T.C.

MARMARA ÜNİVERSİTESİ

EĞİTİM BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR VE ÖĞRETİM TEKNOLOJİLERİ EĞİTİMİ ANABİLİM DALI

BİLGİSAYAR VE ÖĞRETİM TEKNOLOJİLERİ ÖĞRETMENLİĞİ BİLİM DALI

NESNE YÖNELİMLİ PROGRAMLAMA DERSİNİN GELİŞTİRİLMESİ ÜZERİNE BİR EYLEM ARAŞTIRMASI

OSMAN GAZİ YILDIRIM

(Doktora Tezi)

İstanbul, 2022

T.C.

MARMARA ÜNİVERSİTESİ

EĞİTİM BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR VE ÖĞRETİM TEKNOLOJİLERİ EĞİTİMİ ANABİLİM DALI

BİLGİSAYAR VE ÖĞRETİM TEKNOLOJİLERİ ÖĞRETMENLİĞİ BİLİM DALI

NESNE YÖNELİMLİ PROGRAMLAMA DERSİNİN GELİŞTİRİLMESİ ÜZERİNE BİR EYLEM ARAŞTIRMASI

AN ACTION RESEARCH STUDY ON THE DEVELOPMENT OF OBJECT-ORIENTED PROGRAMMING COURSE

OSMAN GAZİ YILDIRIM

(Doktora Tezi)

Danışman

Prof. Dr. Nesrin Özdener Dönmez

İstanbul, 2022

ETİK BEYAN

Marmara Üniversitesi Eğitim Bilimleri Enstitüsü Müdürlüğü Lisansüstü Tez Yazım Kılavuzuna uygun olarak hazırladığım çalışmamda;

- Sunduğum bilgileri, dokümanları ve verileri akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Çalışmamda yararlandığım eserlerin tamamına atıfta bulunarak kaynak gösterdiğimi,
- Elde ettiğim verilerde ve sonuçlarda herhangi bir değişiklik yapmadığımı bildirir;

Aksi bir durumda aleyhimde doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

29.06.2022

Osman Gazi YILDIRIM

ÖZGEÇMİŞ

2004	Maltepe Askeri Lisesinden (İzmir) Mezuniyet
2008	Orta Doğu Teknik Üniversitesi, Eğitim Fakültesi, Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü'nden Mezuniyet
2008	Kara Kuvvetleri Astsubay Meslek Yüksekokulu, Bilgisayar Teknolojisi Bölümü'nde Öğretim Görevlisi Olarak Göreve Başlangıç
2010	Ege Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar ve Öğretim Teknolojileri Eğitimi Bilim Dalı Yüksek Lisans Programına Giriş
2013	Ege Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar ve Öğretim Teknolojileri Eğitimi Bilim Dalı Yüksek Lisans Programından Mezuniyet
2016	Marmara Üniversitesi, Eğitim Bilimleri Enstitüsü, Bilgisayar ve Öğretim Teknolojileri Eğitimi Bilim Dalı Doktora Programına Giriş
2016	Milli Savunma Üniversitesi Kara Astsubay Meslek Yüksekokulu, Bilgisayar Teknolojisi Bölümü'nde Öğretim Görevlisi Olarak Göreve Başlangıç

ÖN SÖZ

Bu tez çalışması ilham kaynağını, Yarının Sınırında (Edge of Tomorrow) filminden almıştır. Filmin kahramanı William Cage'in bir zaman döngüsü içinde sürekli daha iyiye ulaşmak için çabalaması; filmin başlarında savaşmak konusunda hiçbir deneyimi yokken neden sonra kendini işine adaması; gün be gün daha azimli, güçlü ve donanımlı olmayı başarması bizzat bu tezin yazarının da deneyimlediği bir süreç olmuştur. Her daim daha mükemmele ulaşma yolculuğumda sadece akademik bilgi ve birikimleriyle değil; aynı zamamda nadide kişiliği, güçlü karakteri ve engin hoşgörüsüyle bana destek olmaktan biran bile imtina etmeyen danışmanım sayın Prof. Dr. Nesrin Özdener Dönmez hocama teşekkürlerimi ve saygılarımı sunarım. Yolumu kaybettiğimi hissettiğim, yorulduğum ve umutsuzluğa düştüğüm anlarda kendisinin bir kutup yıldızı gibi parlaması, beni her daim cesaretlendirmiş ve yolumu bulmama yardımcı olmuştur.

Tez izleme komitelerimde yer alan, bu çalışma sayesinde kendisini tanıma fırsatına eriştiğim sayın Prof. Dr. Serhat Bahadır KERT hocama ve kendisiyle seneler sonra karşılaştığım çok değerli bölüm hocalarımdan sayın Dr. Öğr. Üyesi Feride Karaca hocama katkıları, yardımları ve dönütlerinden dolayı teşekkür ederim. Tez jürimde yer alan kıymetli hocam sayın Doç. Dr. Buket Doğan ve Dr. Öğr. Üyesi Aliye Saraç katkılarından ve dönütlerinden hocalarıma şükranlarımı sunarım. Araştırma süresince desteğini her daim yanımda hissettiğim sevgili dostum Dr. Ali GERİŞ'e teşekkür etmeyi bir borç bilirim.

Ve son olarak; canım eşim Fadime ve sevgili oğlum Batuhan. Bu süreçte uykusuz gecelerimin yükünü hafiflettiğiniz, zorlukla katlanmam için beni cesaretlendirdiğiniz ve umut ışığım olduğunuz için sizlere sonsuz teşekkürler. Sizin desteğiniz zaman ve mekânın çok ötesinde. Varlığınız sıcaklığı en derinlerde; evvelde ve ahirde.

Osman Gazi YILDIRIM, 2022

ÖZET

Oyun geliştirmeye dayalı öğrenme stratejisi, öğrencilerin oyunlara karşı olan ilgilerini sıfırdan bir oyun geliştirmeye veya mevcut bir oyuna eklentiler yapmaya yönlendirerek programlamayı öğrenmelerini sağlamayı amaçlamaktadır. Bu stratejide öğrenciler yazılım mühendisliği yöntem ve tekniklerini oyun geliştirme projeleri vasıtasıyla deneyimlemektedir. Bu tez çalışmasında araştırmacının kendisinin vermekte olduğu Nesne Yönelimli Programlama dersinde yaşanan sorunları en aza indirebilmek ve dersin verimliliğini artırabilmek için oyun geliştirmeye dayalı programlama öğretiminin mevcut derse bütünleşmiş edildiği bir eylem araştırması yürütülmüştür. Bu bağlamda araştırmanın amacı; mevcut Nesne Yönelimli Programlama dersinin sorunlarının tespit edilmesi; mezun öğrencilerin, uzmanların ve alanyazındaki çalışmaların önerileri ışığında mevcut dersin oyun geliştirmeye dayalı öğretim stratejisini temel alarak yeniden tasarlanması; geliştirilen dersin uygulanması ve bu süreçte yaşanan deneyimlerin raporlanması olarak belirlenmiştir.

Belirlenen amaç ve hedefler doğrultusunda bu tez çalışması, iki araştırma döngüsünü kapsayacak şekilde tasarlanmış ve uygulanmıştır. Birinci döngü analiz, tasarım, uygulama ve değerlendirme basamaklarını içerirken ikinci döngü tasarım, uygulama ve değerlendirme basamaklarını içermiştir. Birinci döngünün analiz aşamasında daha önce nesne yönelimli programlama dersini almış ve okullarından mezun olmuş 59 katılımcı ile nesne yönelimli programlama dersini vermekte olan altı öğretim elemanından veriler toplanarak mevcut dersin sorunlarının detaylı bir şekilde tespit edilmesi sağlanmıştır. İlave olarak analiz aşamasında öğrencilerin derste geliştirmek istedikleri yazılım projesi ve oyun türleri nedenleriyle incelenmiştir. Tasarım aşamasında; analiz aşamasından elde edilen verilerin çözümlenmesinden ve alanyazından yararlanılarak mevcut nesne yönelimli programlama dersi Morrison, Ross ve Kemp öğretim tasarım modeli ışığında oyun geliştirmeye dayalı öğretim stratejisini temel alacak şekilde yeniden tasarlanmıştır. Birinci döngünün uygulama aşaması sekiz hafta sürecek şekilde Türkiye'deki bir meslek yüksekokulunun bilgisayar teknolojisi programında öğrenim görmekte olan 29 önlisans öğrencisi ile yürütülmüştür. Birinci döngünün uygulama aşamasından sonra uygulanan eylem planının etkinliği değerlendirilmiş ve birinci döngüde karşılaşılan sorunların giderilmesi amacıyla ikinci döngünün tasarım aşamasında bazı revizyonlar gerçekleştirilmiştir. Revize edilen nesne yönelimli programlama dersi acil uzaktan öğretim yöntemi kullanılarak 15 hafta sürecek şekilde Türkiye'deki bir meslek yüksekokulunun bilgisayar teknolojisi programında öğrenim görmekte olan 30 önlisans öğrencisi ile yürütülmüştür. Bu aşamadan sonra

uygulanan eylem planının değerlendirilmesine yönelik yansıtma yapılmıştır. Araştırmada hem nitel hem de nicel araştırma tekniklerine uygun olarak toplam 14 adet veri toplama aracı kullanılmıştır.

Araştırmanın birinci döngü bulgularına göre oyun geliştirmeye dayalı stratejisi temel alınarak sunulan öğretimin katılımcıların nesne yönelimli programlama bilgi ve becerilerini anlamlı şekilde geliştirdiği tespit edilmiştir. Bunun yanında katılımcıların en çok çokbiçimlilik, kurucu metot ve kapsülleme konularında zorlandıkları da ortaya çıkmıştır. İkinci döngüde gerçekleştirilen revizyonlarla birlikte bu konularda gelişim sağlanmıştır. Araştırmanın diğer sonuçlarına göre oyun geliştirmeye dayalı programlama öğretiminin her iki döngüde de katılımcıların çoğunun programlama dersine yönelik motivasyonlarını ve programlama öz yeterlilik algılarını olumlu etkilediği görülmüştür. Ancak programlama kaygısı bağlamında değerlendirildiğinde, katılımcıların programlama kaygılarının çalışmanın başlarında arttığı ancak süreç içinde azaldığı; yine de programlama kaygılarının devam ettiği sonucuna varılmıştır.

Bu çalışma kapsamında gerçekleştirilen öğretim tasarımın ve uygulama sonucunda elde edilen deneyimlerin, geliştirilen "Programlama Kaygı Ölçeğinin" alanyazına katkı sunması açısından önemli görülmektedir. Oyun geliştirmeye dayalı öğretim stratejisinin nesne yönelimli programlama dersine entegrasyonu açısından değerlendirildiğinde, mevcut çalışmanın öğrencilerin nesne yönelimli programlama dersinde karşılaştıkları sorunları en aza indirmeye çalışması; öğrencilerin programlamaya yönelik ilgi ve motivasyonlarını artırarak programlama öz yeterlilik algılarında olumlu değişimler yaratmayı amaçlaması bakımından güncel, işlevsel ve özgündür.

Anahtar Kelimeler: Nesne Yönelimli Programlama, Oyun Geliştirmeye Dayalı Programlama Öğretimi, Oyun Programlama, Programlama Dersine Yönelik Motivasyon, Programlama Öz Yeterliliği, Programlama Kaygısı.

ABSTRACT

The game development-based learning strategy seeks to assist students in learning programming by directing their interest in games towards creating games from scratch or adding features to existing games. Students gain exposure to software engineering principles and techniques through game development projects in this strategy. This participatory action research aims to redesign the Object-Oriented Programming course in which the first author is both the instructor and researcher by integrating game development-based programming instruction to make it more effective and efficient.

In line with the determined goals and objectives, this dissertation was designed and implemented to cover two research cycles. While the first cycle included the analysis, design, implementation, and evaluation stages, the second cycle comprised the design, implementation, and evaluation steps. In the analysis phase of the first cycle, data were collected from 59 participants who had previously taken the object-oriented programming course and graduated from their schools, and six instructors teaching the object-oriented programming course, and the problems of the current course were determined in detail. In addition, the software project and game genres that the students desired to develop in the course were investigated during the analysis phase. The present object-oriented programming course was redesigned in light of the Morrison, Ross, and Kemp instructional design model, based on the study of the data obtained from the analysis phase and the literature. The implementation phase of the first cycle was carried out with 29 associate degree students studying in the computer technology program of a vocational school in Turkey, for eight weeks. The effectiveness of the action plan adopted after the first cycle's implementation phase was assessed, and some adjustments were conducted during the second cycle's design stage to address the issues that emerged during the first cycle. For 15 weeks, the revised object-oriented programming course was taught to 30 associate degree students in a vocational school's computer technology program in Turkey, utilizing the emergency remote learning technique. Following this stage, an evaluation of the action plan was carried out. The study employed a total of 14 data collection tools, which included both qualitative and quantitative research methods.

According to the first cycle findings of the research, it was determined that the instruction based on the game-development-based strategy significantly improved the object-oriented programming knowledge and skills of the participants. In addition, it was

revealed that the participants had the most difficulties in polymorphism, constructor methods, and encapsulation. Progress has been made in these areas as a result of the adjustments made in the second cycle. According to the research's other findings, game-based programming education had a good impact on the motivation of the majority of the participants toward the programming course and their perceptions of programming self-efficacy in both cycles. However, when the participants' programming anxiety was assessed in the context of the study, it was discovered that their programming anxiety increased at the start of the study but decreased over time; however, it was established that programming concerns persisted.

The instructional design employed in this study, as well as the experience gained as a result of the implementation, are regarded as significant contributions to the literature. Regarding the endeavor to integrate a game development-based teaching technique into an object-oriented programming course, the current study is up-to-date, functional, and distinctive. It aims to reduce the problems schools encounter in the object-oriented programming course and to increase students' interest and motivation for programming by increasing their programming self-efficacy perceptions. In addition, the Programming Anxiety Scale was developed as part of the research.

Keywords: Object-Oriented Programming, Game Development Based Learning, Game Programming, Motivation for Programming, Programming Self-Efficacy, Programming Anxiety.

İÇİNDEKİLER

Sayfa No.

ETİK BEYAN i
ÖZGEÇMİŞ ii
ÖN SÖZ iii
ÖZET iv
ABSTRACT vi
İÇİNDEKİLER viii
TABLOLAR LİSTESİ xiii
ŞEKİLLER LİSTESİ xviii
KISALTMALAR xxiii
BÖLÜM I: GİRİŞ 1
1.1. Problem 1
1.2. Amaç 9
1.2.1. Araştırma Soruları 11
1.3. Önem 12
1.4. Sınırlılıklar 13
1.5. Varsayımlar 14
1.6. Tanımlar 14
BÖLÜM II: KAVRAMSAL ÇERÇEVE 16
2.1. Programlama Öğretimi, Önemi ve Güçlükleri 16
2.2. Programlama Derslerinde Öğrencilerin Başarılarını Etkileyen Faktörler 21
2.2.1. Programlama Deneyimi 22
2.2.2. Programlamaya Yönelik Öz Yeterlilik Algısı 24
2.2.3. Programlamaya Yönelik Motivasyon 26
2.2.4. Programlama Kaygısı 27
2.2.5. Diğer Faktörler 29
2.3. Programlama Öğretimini Kolaylaştırma Stratejileri 30
2.4. Oyun Geliştirmeye Dayalı Öğrenme 33
2.4.1. Tarihçesi ve Sınıflandırması 35

2.4.1.1. Oyuna benzer geliştirme ortamlarının kullanımı 36
2.4.1.2. Oyun geliştirme veya mevcut bir oyuna eklentiler yapma 38
2.4.2. Dayandığı Pedagojiler 41
BÖLÜM III: YÖNTEM 44
3.1. Araştırma Modeli 44
3.2. Eylem Araştırması Süreci 45
3.2.1. Birinci Döngü 46
3.2.1.1. Araştırma süreci 46
3.2.1.2. Uygulama ortamı 49
3.2.1.3. Çalışma grubu 50
3.2.2. İkinci Döngü 55
3.2.2.1. Araştırma süreci 55
3.2.2.2. Uygulama ortamı 56
3.2.2.3. Çalışma grubu 57
3.3. Veri Toplama Araçları 59
3.3.1. Katılımcı Bilgi Formu 59
3.3.2. NYP Başarı Sınavı 60
3.3.3. Proje Değerlendirme Rubriği 61
3.3.4. Dijital Öğrenci Günlükleri 63
3.3.5. Araştırmacı Günlüğü 63
3.3.6. Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği 64
3.3.7. Programlama Öz Yeterlilik Ölçeği 65
3.3.8. Programlama Kaygı Ölçeği 67
3.3.9. Odak Grup Görüşmesi 71
3.3.10. Analiz Aşaması Katılımcı Anketi 72
3.3.11. Analiz Aşaması Öğretim Elemanı Görüşme Formu 72
3.3.12. Unity 3D Başarı Sınavı 73
3.4. Veri Toplama Süreci 74
3.4.1. Birinci Döngü 74
3.4.2. İkinci Döngü 75
3.5. Verilerin Analizi 76
3.5.1. Nitel Tekniklerle Elde Edilen Verilerin Analizi 78
3.5.2. Nicel Tekniklerle Elde Edilen Verilerin Analizi 79

3.6. Araştırmacının Rolü 79
3.7. Geçerlik ve Güvenirlik 80
3.7.1. Nitel Araştırma Teknikleri Boyutu 80
3.7.2. Nicel Araştırma Teknikleri Boyutu 82
3.8. Araştırmanın Etiği 83
BÖLÜM IV: BULGULAR VE YORUMLAR 85
4.1. Birinci Döngü 85
4.1.1. Analiz Aşaması 85
4.1.1.1. Mevcut NYP dersinin karakteristik özellikleri 85
4.1.1.2. Araştırma Sorusu 1: Öğrencilerin mevcut NYP dersi hakkındaki görüşleri nelerdir? 88
4.1.1.3. Araştırma Sorusu 2: Mevcut NYP dersinde öğrencilerin ve öğretim elemanlarının yaşadıkları sorunlar nelerdir? 91
4.1.1.4. Araştırma Sorusu 3: Öğrenciler mevcut NYP dersinde ne tür programlama uygulamalarını geliştirmenin kendileri için daha faydalı olacağını değerlendirmektedir? 95
4.1.1.5. Araştırma Sorusu 4: Mezun öğrencilerin mevcut NYP dersinde oyun geliştirmeye dayalı öğretimin uygulanabilirliği konusundaki değerlendirmeleri nelerdir? 106
4.1.2. Tasarım Aşaması 107
4.1.2.1. Araştırma Sorusu 5: NYP dersi Türkiye'deki diğer MYO'larda hangi içeriklerle verilmektedir? 107
4.1.2.2. Araştırma Sorusu 6: Mevcut problemleri ortadan kaldırmak için oyun geliştirmeye dayalı öğretim NYP dersine nasıl entegre edilebilir? 112
4.1.2.3. Öğretim tasarımı süreci 122
4.1.3. Uygulama Aşaması 137
4.1.3.1. Birinci döngüde uygulanan öğretim programı 137
4.1.3.2. Araştırma Sorusu 7: Eylem planının uygulanması sürecinde katılımcıların ve araştırmacının yaşadığı sorunlar nelerdir? 139
4.1.4. Değerlendirme Aşaması 145
4.1.4.1. Araştırma Sorusu 8: Eylem planının uygulanmasından sonra katılımcıların nesne yönelimli programlama bilgi ve beceri düzeylerinde nasıl bir değişim meydana gelmiştir? 145
4.1.4.2. Araştırma Sorusu 9: Geliştirilen NYP dersine yönelik katılımcıların görüşleri nelerdir? 161
4.2. İkinci Döngü 165
4.2.1. Tasarım Aşaması 165

4.2.1.1. Araştırma Sorusu 10: Birinci döngüden elde edilen deneyimler ışığında tasarlanan NYP dersinde ne gibi değişiklikler yapılabilir? 165
4.2.1.2. Revize edilen NYP dersinin öğretim programı 173
4.2.2. Uygulama Aşaması 176
4.2.2.1. Araştırma Sorusu 11: Eylem planının uygulanması sürecinde katılımcıların ve araştırmacının yaşadığı sorunlar nelerdir? 176
4.2.3. Değerlendirme Aşaması 183
4.2.3.1. Araştırma Sorusu 12: Eylem planının uygulanmasından sonra katılımcıların nesne yönelimli programlama bilgi ve beceri düzeyleri nasıl bir değişim göstermiştir? 183
4.2.3.2. Araştırma Sorusu 13: Eylem planının uygulanmasıyla birlikte katılımcıların psikolojik faktörlerinde (motivasyon, kaygı, öz yeterlilik algısı vb.) nasıl bir değişim meydana gelmiştir? 199
4.2.3.3. Araştırma Sorusu 14: Geliştirilen NYP dersine yönelik katılımcıların görüşleri nelerdir? 212
4.3. Birinci ve İkinci Döngü Karşılaştırması 226
4.3.1. Akademik Başarı Açısından Karşılaştırma 226
4.3.2. Algılanan Programlama Bilgi ve Zorluk Seviyesi Açısından Karşılaştırma 230
4.3.3. Psikolojik Faktörler Açısından Karşılaştırma 231
BÖLÜM V: SONUÇ, TARTIŞMA VE ÖNERİLER 233
5.1. Sonuç ve Tartışma 233
5.1.1. NYP Dersinin Yeniden Tasarlanması Sürecine İlişkin Sonuç ve Tartışma 234
5.1.2. Mevcut NYP Dersinin Sorunlarına Yönelik Üretilen Çözümlerin Etkinliğine İlişkin Sonuç ve Tartışma 237
5.1.2.1. Öğretim programı kategorisindeki problemlerin çözümüne ilişkin sonuç ve tartışma 237
5.1.2.2. Psikolojik faktörler kategorisindeki problemlerin çözümüne ilişkin sonuç ve tartışma 244
5.1.2.3. Ders etkinlikleri ve materyalleri kategorisindeki problemlerin çözümüne ilişkin sonuç ve tartışma 249
5.2. Öneriler 251
BÖLÜM VI: KAYNAKLAR 254
EKLER 289
Ek 1. Katılımcı Bilgi Formu 289
Ek 2. NYP Başarı Sınavı Madde Belirtke Tablosu 292
Ek 3. NYP Başarı Sınavı Kapsam Geçerliliği Uzman Görüş Formu 293

Ek 4. NYP Başarı Sınavı 299
Ek 5. Proje Değerlendirme Rubriği Kapsam Geçerliliği Uzman Görüş Formu 304
Ek 6. Proje Değerlendirme Rubriği 307
Ek 7. Programlama Kaygı Ölçeği 309
Ek 8. Analiz Aşaması Katılımcı Anketi 310
Ek 9. Analiz Aşaması Öğretim Elemanı Görüşme Formu 314
Ek 10. Unity 3D Başarı Sınavı Kapsam Geçerliliği Uzman Görüş Formu 315
Ek 11. Unity 3D Başarı Sınavı 319
Ek 12. Haftalık Ders Planları 323

TABLOLAR LİSTESİ

Sayfa No.

Tablo 1.1. Eylem araştırmasının döngülerine ait araştırma soruları 11
Tablo 2.1. Programlama öğretim sürecinde kullanılan bilgi türleri (Mcgill ve Volet, 1997) 17
Tablo 3.1. Birinci döngünün analiz aşamasında gerçekleştirilen işlemler 47
Tablo 3.2. Birinci döngünün tasarım aşamasında gerçekleştirilen işlemler 48
Tablo 3.3. Birinci döngünün aşamaları ve katılımcıları 50
Tablo 3.4. Mezun katılımcılara ait demografik özellikler 51
Tablo 3.5. Öğretim elemanlarına ait demografik özellikler 52
Tablo 3.6. Birinci döngü uygulama aşaması katılımcılarına ait demografik özellikler 53
Tablo 3.7. İkinci döngü katılımcılarına ait demografik özellikler 57
Tablo 3.8. Araştırma kapsamında kullanılan veri toplama araçları 59
Tablo 3.9. NYP dersi kazanımları 60
Tablo 3.10. Proje Değerlendirme Rubriğinin geçerliliğine ilişkin uzman görüşleri 62
Tablo 3.11. Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeğine ait bilgiler 64
Tablo 3.12. Programlama Öz Yeterlilik Ölçeğine ait bilgiler 65
Tablo 3.13. Programlama Öz Yeterlilik Ölçeği madde faktör yükleri ve madde toplam korelasyonu sonuçları 66
Tablo 3.14. Programlama Öz Yeterliliğine ait hesaplanan uyum indeks değerleri 67
Tablo 3.15. Programlama Kaygı Ölçeğine ait bilgiler 67
Tablo 3.16. Programlama Kaygı Ölçeği madde faktör yükleri 69
Tablo 3.17. Programlama Kaygı Ölçeğine ait hesaplanan uyum indeks değerleri 70
Tablo 3.18. Ölçeğin yakınsama geçerliliği ve güvenirlik analizi sonuçları 71
Tablo 3.19. Ölçeğin ayrıt edici geçerlilik analizi sonuçları 71
Tablo 3.20. Unity 3D Başarı Sınavı madde belirtke tablosu 73
Tablo 3.21. Birinci döngüde izlenen veri toplama adımları 75
Tablo 3.22. İkinci döngüde izlenen veri toplama adımları 76
Tablo 3.23. Araştırma sorularını yanıtlamaya yönelik veri toplama teknikleri, veri türleri ve veri analizi 77
Tablo 3.24. Nitel araştırma teknikleri boyutunda gerçekleştirilen geçerlik ve güvenirlik çalışmaları 80

Tablo 3.25. Araştırma soruları bağlamında araştırmacılar arası uyum ve güvenirlik analizleri 82
Tablo 3.26. Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeğine ait Cronbach Alfa (α) iç tutarlık katsayıları 83
Tablo 3.27. Programlama Öz Yeterlilik Ölçeğine ait Cronbach Alfa (α) iç tutarlık katsayıları 83
Tablo 3.28. Programlama Kaygı Ölçeğine ait Cronbach Alfa (α) iç tutarlık katsayıları 83
Tablo 4.1. Mevcut NYP dersine ait öğretim programı 85
Tablo 4.2. Vize, final ve değerlendirme notlarının ağırlıkları 87
Tablo 4.3. Mezun öğrencilerin mevcut NYP dersinin katkılarına yönelik görüşleri 88
Tablo 4.4. Mezun öğrencilerin mevcut NYP dersindeki projelere yönelik görüşleri..... 90
Tablo 4.5. Mezun öğrencilerin mevcut NYP dersinde yaşadıkları sorunlara ilişkin görüşleri 91
Tablo 4.6. Öğretim elemanlarının mevcut NYP dersinde yaşadıkları sorunlara ilişkin görüşleri 94
Tablo 4.7. Mezun öğrencilerin geliştirmek istedikleri yazılım türlerine ilişkin betimsel analizler 96
Tablo 4.8. Mezun öğrencilerin geliştirmek istedikleri yazılım türlerine ilişkin ortak kodlara yönelik örnek alıntılar..... 98
Tablo 4.9. Mezun öğrencilerin geliştirmek istedikleri yazılım türlerine ilişkin ortak olmayan kodlara yönelik örnek alıntılar..... 100
Tablo 4.10. Oyun/mobil oyunları geliştirmek isteyen mezun öğrencilerin tercih ettikleri dijital oyun türlerine ilişkin betimsel analizler 101
Tablo 4.11. Mevcut NYP dersinde oyun/mobil oyun geliştirmek isteyen mezun öğrencilerin bu oyun türlerini tercih etmelerine ilişkin ortak kodlara yönelik örnek alıntılar..... 103
Tablo 4.12. Oyun/mobil oyun geliştirmek isteyen mezun öğrencilerin bu oyun türlerini tercih etmelerine ilişkin ortak olmayan kodlara yönelik örnek alıntılar..... 105
Tablo 4.13. Mezun öğrencilerin NYP dersinde oyun geliştirmeye dayalı öğretimin uygulanabirliğine yönelik görüşleri..... 106
Tablo 4.14. Farklı üniversitelerde okutulan NYP derslerinin konu kapsamları..... 108
Tablo 4.15. Elde edilen çözüm önerileri doğrultusunda öğretim tasarımında kullanılan stratejiler 120
Tablo 4.16. Geçmiş yıllarında NYP dersini alan öğrencilere ait demografik özellikler ... 125

Tablo 4.17. Geliştirilen NYP dersinin haftalık öğretim programının özeti 131
Tablo 4.18. Kısa sınavların, başarı sınavının ve projelerin ağırlıkları 136
Tablo 4.19. Geliştirilen NYP dersinin haftalık öğretim programının özeti 138
Tablo 4.20. Birinci döngünün uygulanması aşamasında ortaya çıkan sorunlar ve sorunların çözümüne yönelik alınan eylem kararları 139
Tablo 4.21. NYP Başarı Sınavı öntest ve sontest puanlarına yönelik betimsel istatistikler 145
Tablo 4.22. NYP Başarı Sınavı öntest-sontest puanlarının karşılaştırılmasına ilişkin Wilcoxon sıralı işaretler testi sonuçları 145
Tablo 4.23. NYP Başarı Sınavının teorik kısmına ilişkin kazanımlarında başarısız olan katılımcı sayıları 147
Tablo 4.24. NYP Başarı Sınavının uygulama kısmına ilişkin kazanımlarında başarısız olan katılımcı sayıları 150
Tablo 4.25. Sontest teorik sınav ile uygulama sınavı Pearson çarpım moment korelasyon analizi sonuçları 151
Tablo 4.26. Ölçeklere ilişkin betimsel istatistik sonuçları 152
Tablo 4.27. Pearson çarpım moment korelasyon analizi sonuçları 153
Tablo 4.28. Algoritma ve Programlamaya Giriş dersi ile NYP Başarı Sınavı sontest başarı puanları regresyon analiz sonuçları 154
Tablo 4.29. Görsel Programlama dersi ile NYP Başarı Sınavı sontest başarı puanları regresyon analiz sonuçları 154
Tablo 4.30. Katılımcıların algıladıkları programlama bilgi seviyesindeki değişimin yönü 157
Tablo 4.31. Katılımcıların algıladıkları programlama bilgi seviyesindeki değişim durumu 157
Tablo 4.32. Katılımcıların algıladıkları programlama zorluk seviyesindeki değişim durumu 158
Tablo 4.33. Mann Whitney U testi sonuçları 158
Tablo 4.34. Düşük ve yüksek düzeyde başarılı olan öğrencilerin ölçek verilerine ilişkin betimsel istatistik sonuçları 159
Tablo 4.35. Karting Microgame oyun prototipine ilave edilmesi önerilen özelliklerin değerlendirilmesi 165
Tablo 4.36. Öğretim programı temasının alt kategorilerinde gerçekleştirilen değişikliklerle ilgili bilgiler 166

Tablo 4.37. Öğretim materyalleri ve etkinlikleri temasının alt kategorilerinde gerçekleştirilen değişikliklerle ilgili bilgiler 170
Tablo 4.38. Ders için tasarlanan rozetler ve kullanım amaçları 173
Tablo 4.39. Revize edilen NYP dersinin haftalık öğretim programının özeti 174
Tablo 4.40. İkinci döngünün uygulanması aşamasında ortaya çıkan sorunlar ve sorunların çözümüne yönelik alınan eylem kararları 177
Tablo 4.41. Katılımcıların paylaştıkları kodlama hatalarıyla ilgili çözümlemeler 178
Tablo 4.42. NYP Başarı Sınavı öntest ve sontest puanlarına yönelik betimsel istatistikler 183
Tablo 4.43. NYP Başarı Sınavı öntest-sontest puanlarının karşılaştırılmasına ilişkin Wilcoxon sıralı işaretler testi sonuçları 183
Tablo 4.44. NYP Başarı Sınavının teorik kısmına ilişkin kazanımlarında başarısız olan katılımcı sayıları 185
Tablo 4.45. NYP Başarı Sınavının uygulama kısmına ilişkin kazanımlarında başarısız olan katılımcı sayıları 186
Tablo 4.46. Sontest teorik sınav ile uygulama sınavı Pearson çarpım moment korelasyon analizi sonuçları 187
Tablo 4.47. Katılımcıların oyun projelerinin değerlendirilmesi sonuçları 187
Tablo 4.48. Ölçeklere ilişkin betimsel istatistik sonuçları 190
Tablo 4.49. Pearson çarpım moment korelasyon analizi sonuçları 191
Tablo 4.50. Katılımcıların algıladıkları programlama bilgi seyiyesindeki değişimin yönü 193
Tablo 4.51. Katılımcıların algıladıkları programlama bilgi seviyesindeki değişim durumu 195
Tablo 4.52. Katılımcıların algıladıkları programlama zorluk seviyesindeki değişim durumu 196
Tablo 4.53. Mann Whitney U testi sonuçları 196
Tablo 4.54. Düşük ve yüksek düzeyde başarılı olan öğrencilerin ölçek verilerine ilişkin betimsel istatistik sonuçları 196
Tablo 4.55. NYP Başarı Sınavı öntest ve sontest puanlarına yönelik betimsel istatistikler 199
Tablo 4.56. Unity 3D Başarı Sınavı öntest-sontest puanlarının karşılaştırılmasına ilişkin Wilcoxon sıralı işaretler testi sonuçları 199

Tablo 4.57. Uygulamanın öğrencilerin programlama dersine yönelik motivasyonlarına etkilerine yönelik görüşleri 200
Tablo 4.58. Tek faktörlü tekrarlanan ölçümler ANOVA testi sonuçları 205
Tablo 4.59. Uygulamanın öğrencilerin programlama öz yeterlilik algısına etkilerine yönelik görüşleri 205
Tablo 4.60. Uygulamanın öğrencilerin programlama kaygısına etkilerine yönelik görüşleri 209
Tablo 4.61. Tek faktörlü tekrarlanan ölçümler ANOVA testi sonuçları 211
Tablo 4.62. Uzaktan öğretimin öğrencilerin programlama dersine yönelik motivasyonlarına etkilerine yönelik görüşleri 216
Tablo 4.63. Katılımcıların ders materyallerine yönelik görüşleri 222
Tablo 4.64. NYP Başarı Sınavı öntest ve sontest puanlarına yönelik betimsel istatistikler 227
Tablo 4.65. Döngülere ait öntest başarı puan ortalamalarının karşılaştırması 227
Tablo 4.66. Döngülere ait sontest başarı puan ortalamalarının karşılaştırması 227
Tablo 4.67. Algılanan programlama bilgi seviyesindeki değişimlerin karşılaştırılması ... 231
Tablo 4.68. Algılanan programlama zorluk seviyesindeki değişimlerin karşılaştırılması 231
Tablo 4.69. Toplam programlama dersine yönelik motivasyon puanları ortalamaları karşılaştırması 231
Tablo 4.70. Akran kaygısı faktörü ortalamaları karşılaştırması 232
Tablo 4.71. Programlama beceri kaygısı faktörü ortalamaları karşılaştırması 232
Tablo 4.72. Programlama öz yeterlilik algısı puanları ortalamaları karşılaştırması 232

ŞEKİLLER LİSTESİ

Sayfa No

Şekil 2.1. Micro:bits (https://blog.adacore.com/) 32
Şekil 2.2. Lego Mindstroms (https://www.lego.com) 32
Şekil 2.3. Bee-bot (Rose, 2019) 32
Şekil 2.4. Logo (eclipse.org, 2021) 36
Şekil 2.5. Karel the Robot (Roberts, 2005) 36
Şekil 2.6. Alice programı ekran görüntüsü (Abdellatif, 2020) 36
Şekil 2.7. Scratch programı ekran görüntüsü (Stenerson, 2012) 37
Şekil 2.8. Kodu programı ekran görüntüsü (Rose, 2019) 38
Şekil 2.9. Greenfoot programı ekran görüntüsü (Kölling, 2010) 38
Şekil 3.1. Eylem araştırması süreci 45
Şekil 3.2. Birinci döngünün zaman çizelgesi 46
Şekil 3.3. Birinci döngü uygulama ortamı 49
Şekil 3.4. Katılımcıların günlük bilgisayar kullanımı sıklıkları 51
Şekil 3.5. Katılımcıların günlük dijital oyun oynama sıklıkları 51
Şekil 3.6. Katılımcıların algıladıkları NYP bilgi seviyeleri 51
Şekil 3.7. Katılımcıların algıladıkları NYP zorluk seviyeleri 51
Şekil 3.8. Katılımcıların günlük bilgisayar kullanımı sıklıkları 53
Şekil 3.9. Katılımcıların günlük dijital oyun oynama sıklıkları 53
Şekil 3.10. Katılımcıların algıladıkları NYP bilgi seviyeleri 54
Şekil 3.11. Katılımcıların algıladıkları NYP zorluk seviyeleri 54
Şekil 3.12. Katılımcıların ortaöğretimde programlama dersi alma durumları 54
Şekil 3.13. Katılımcıların oyun geliştirme tecrübeleri 54
Şekil 3.14. İkinci döngü araştırma süreci 55
Şekil 3.15. Kullanılan sanal sınıfın akış ekranı (Google Classroom) 56
Şekil 3.16. Kullanılan sanal sınıfın sınıf çalışmaları arayüzü 56
Şekil 3.17. Katılımcıların günlük bilgisayar kullanımı sıklıkları 57
Şekil 3.18. Katılımcıların günlük dijital oyun oynama sıklıkları 57
Şekil 3.19. Katılımcıların algıladıkları NYP bilgi seviyeleri 58
Şekil 3.20. Katılımcıların algıladıkları NYP zorluk seviyeleri 58
Şekil 3.21. Katılımcıların ortaöğretimde programlama dersi alma durumları 58
Şekil 3.22. Katılımcıların oyun geliştirme tecrübeleri 58

Şekil 3.23. Standardize edilmiş sonuçlar ile birinci-düzey DFA 70
Şekil 3.24. Birinci döngüye ait veri toplama süreci 74
Şekil 3.25. İkinci döngüye ait veri toplama süreci 75
Şekil 4.1. Noktalar arası mesafe hesaplama projesi 86
Şekil 4.2. Statik sınıflar kullanarak bazı geometrik şekillerin özelliklerini hesaplama projesi 86
Şekil 4.3. Kalıtım kullanılarak farklı tip araçların otopark ücretlerinin hesaplanması projesi 86
Şekil 4.4. Basit bir hafıza oyunu 86
Şekil 4.5. Ders yönetim sisteminin ekran alıntısı 87
Şekil 4.6. Öğrencilerin mevcut NYP dersinde geliştirmek istedikleri yazılım türlerinin nedenlerine ilişkin görüşleri 97
Şekil 4.7. Mevcut NYP dersinde oyun/mobil oyun geliştirmek isteyen öğrencilerin bu oyun türlerini tercih etmelerine ilişkin görüşleri 102
Şekil 4.8. Öğretim programı sorun kategorisine ait çözüm önerileri 113
Şekil 4.9. Psikolojik faktörler sorun kategorisine ait çözüm önerileri 114
Şekil 4.10. Ders etkinlikleri sorun kategorisine ait çözüm önerileri 115
Şekil 4.11. Dersin işlenişi sorun kategorisine ait çözüm önerileri 116
Şekil 4.12. Ders materyalleri sorun kategorisine ait çözüm önerileri 117
Şekil 4.13. Değerlendirme yöntemi sorun kategorisine ait çözüm önerileri 118
Şekil 4.14. Öğrencilerin hazırbulunuşluk seviyeleri sorun kategorisine ait çözüm önerileri 119
Şekil 4.15. Morrison, Ross ve Kemp Modeli (2004) 124
Şekil 4.16. Ders materyali olarak kullanılacak Karting Microgame oyun prototipi 128
Şekil 4.17. Oyun prototipine eklenecek modlara yönelik hazırlanan UML sınıf diyagramları 128
Şekil 4.18. Geliştirmesi planlanan oyun prototipinin son halinin örnek bir gösterimi 129
Şekil 4.19. Kullan-değiştir-oluştur stratejisi (Lee ve diğerleri, 2011) 134
Şekil 4.20. Geliştirilen NYP dersi için tasarlanan ders portalı 135
Şekil 4.21. Geliştirilen NYP dersinin kavramsal çerçevesi (Thota ve Whitfield'dan (2010) uyarlanmıştır.) 137
Şekil 4.22. Kodlama hatalarının türleri ve dağılımı 140
Şekil 4.23. Doğru olarak kodlanmış Monobehaviour metodu 141

Şekil 4.24. D2_K21 kodlu katılımcının projesinden etiket (tag) eklenmemiş bir bonus nesnesi gösterimi 141
Şekil 4.25. D2_K3 kodlu katılımcının projesinden Is Trigger özelliği eklenmemiş bir capsule collider bileşeni gösterimi 142
Şekil 4.26. D2_K15 kodlu katılımcının projesinden sınıf değişkenlerine değer atanmamış bir sınıf gösterimi 142
Şekil 4.27. NYP Başarı Sınavının teorik kısmına ilişkin kazanımlara ait başarı yüzdeleri ve puan ortalamaları 146
Şekil 4.28. "Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar." (K8) kazanımında başarısız olan sınav kâğıdı örneği (D1_K29) 148
Şekil 4.29. "Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar." (K8) kazanımında başarısız olan sınav kâğıdı örneği (D1_K22) 148
Şekil 4.30. "Kurucu metot kavramını örneklerle açıklar." (K5) kazanımında başarısız olan sınav kâğıdı örneği (D1_K10) 149
Şekil 4.31. NYP Başarı Sınavının uygulama kısmına ilişkin kazanımlara ait başarı yüzdeleri ve puan ortalamaları 150
Şekil 4.32. Uygulama öncesinde algılanan programlama bilgi seviyesi dağılım grafiği .. 155
Şekil 4.33. Katılımcıların uygulama öncesi ve uygulama sonrasında algıladıkları programlama bilgi seviyesindeki değişim 156
Şekil 4.34. Katılımcıların algıladıkları zorluk seviyesindeki değişim grafiği 158
Şekil 4.35. Düşük ve yüksek düzeyde başarılı olan öğrencilerin ölçek alt faktör ortalamalarının karşılaştırma grafiği 159
Şekil 4.36. Düşük başarılı öğrencilerin uygulama öncesi ve uygulama sonrasında algıladıkları programlama bilgi seviyesindeki değişim 160
Şekil 4.37. Yüksek başarılı öğrencilerin uygulama öncesi ve uygulama sonrasında algıladıkları programlama bilgi seviyesindeki değişim 161
Şekil 4.38. Katılımcıların oyun geliştirmeye dayalı öğretim süreciyle ilgili algıları 162
Şekil 4.39. Uzaktan öğretim için oluşturulan google sınıfı (sanal sınıf) 168
Şekil 4.40. Ders için oluşturulan Youtube kanalı 169
Şekil 4.41. Ders için kurulan whatsapp grubu 169
Şekil 4.42. Karting Microgame oyun prototipinin yeni versiyonu 172
Şekil 4.43. Google Classroom'da oluşturulan sıkça sorulan sorular forumu 178
Şekil 4.44. Youtube'da oluşturulan sıkça sorulan sorular videolar çalma listesi 178

Şekil 4.45. “İlgili sınıf kod tamamlama ekranında çıkmıyor.” hatası örnek ekran görüntüsü 179
Şekil 4.46. “Sınıf bileşen olarak eklenemiyor.” hatası örnek ekran görüntüsü 179
Şekil 4.47. “Nesne başvurusu bir nesnenin örneğine ayarlanmadı.” hatası örnek ekran görüntüsü 180
Şekil 4.48. “Tag ismi bulunamıyor.” hatası örnek ekran görüntüsü..... 180
Şekil 4.49. Başarı Sınavının teorik kısmına ilişkin kazanımlara ait başarı yüzdeleri ve puan ortalamaları..... 184
Şekil 4.50. NYP Başarı Sınavının uygulama kısmına ilişkin kazanımlara ait başarı yüzdeleri ve puan ortalamaları 186
Şekil 4.51. Yeni objelerin eklenmesi..... 187
Şekil 4.52. Pist tasarımın değiştirilmesi 188
Şekil 4.53. Çarpışma efektlerinin eklenmesi..... 188
Şekil 4.54. Laserlight efektinin eklenmesi 188
Şekil 4.55. Yeni bonus nesnelerinin eklenmesi..... 189
Şekil 4.56. Uygulama öncesinde algılanan programlama bilgi seviyesi dağılım grafiği .. 192
Şekil 4.57. Katılımcıların uygulama öncesi ve uygulama sonrasında algıladıkları programlama bilgi seviyesindeki değişim..... 194
Şekil 4.58. Katılımcıların algıladıkları zorluk seviyesindeki değişim grafiği..... 195
Şekil 4.59. Düşük ve yüksek düzeyde başarılı olan öğrencilerin ölçek alt faktör ortalamalarının karşılaştırma grafiği 197
Şekil 4.60. Düşük başarılı öğrencilerin uygulama öncesi ve uygulama sonrasında algıladıkları programlama bilgi seviyesindeki değişim..... 198
Şekil 4.61. Yüksek başarılı öğrencilerin uygulama öncesi ve uygulama sonrasında algıladıkları programlama bilgi seviyesindeki değişim..... 198
Şekil 4.62. D2_K3 kodlu katılımcının ilave olarak geliştirdiği oyun projesinin ekran alıntısı 201
Şekil 4.63. D2_K8 kodlu katılımcının ilave olarak geliştirdiği oyun projesinin ekran alıntısı 202
Şekil 4.64. Katılımcıların programlama dersine yönelik motivasyon puanları ortalaması değişim grafiği..... 204
Şekil 4.65. Katılımcıların programlama öz yeterlilik puan ortalamaları değişim grafiği.. 208
Şekil 4.66. Katılımcıların programlama kaygısı puan ortalamaları değişim grafiği 211
Şekil 4.67. Akran kaygısı alt faktörü ortalama puanları değişim grafiği 212

Şekil 4.68. Programlama beceri kaygısı alt faktörü ortalama puanları değişim grafiği 212
Şekil 4.69. Katılımcıların oyun geliştirmeye dayalı öğretim süreciyle ilgili algıları 213
Şekil 4.70. Katılımcıların oyun geliştirmeye dayalı öğretim sürecinin bireysel ve mesleki gelişimlerine katkılarına yönelik görüşleri 219
Şekil 4.71. Ders materyallerinin katkısının değerlendirilmesi 223
Şekil 4.72. Katılımcıları öğretim süreciyle ilgili önerileri..... 224
Şekil 4.73. Teorik sınav başarı yüzdesi açısından karşılaştırma sonuçları..... 228
Şekil 4.74. Teorik sınav puan ortalaması açısından karşılaştırma sonuçları 229
Şekil 4.75. Uygulama sınavı başarı yüzdesi açısından karşılaştırma sonuçları..... 230
Şekil 4.76. Uygulama sınavı puan ortalaması açısından karşılaştırma sonuçları 230

KISALTMALAR

NYP	Nesne Yönelimli Programlama
BÖTE	Bilgisayar ve Öğretim Teknolojileri Eğitimi
IDE	Integrated Development Environment (Tümleşik Geliştirme Ortamı)
MIT	Massachusetts Institute of Technology
MYO	Meslek Yüksekokulu

BÖLÜM I: GİRİŞ

1.1. Problem

Günümüzde bilgisayarları ve bilgisayar yazılımlarını kullanmadığımız alan yok denecek kadar azdır. Alan Turing'in 1936'da Princeton Üniversitesinde doktora öğrencisiyken *"evrensel makine (universal machine)"* olarak alanyazına tanıttığı bu teknoloji; bundan yaklaşık 85 yıl sonra bilim, sanayi, finans, haberleşme, otomasyon, savunma, araştırma-geliştirme, akademik faaliyetler ve günlük hayatından vazgeçilmez bir unsuru haline gelmiştir. Son yirmi yılda internetin ve mobil teknolojilerin de gelişmesiyle büyük veri, yapay zekâ, blok zinciri, biyoinformatik**,** giyilebilir teknolojiler, bulut bilişim, üç boyutlu yazıcılar, robotik ve sanal gerçeklik gibi birçok alanda atılımlar yaşanmış; bu teknolojiler ve yazılımlar günlük hayatta daha sık kullanılır olmuştur. Nitekim Covid-19 salgının başlangıcında Çinli bilim insanlarının virüsün gen dizilimini hızlı bir şekilde çözerek bu gen dizilimini tüm bilim insanlarıyla paylaşması yine bilgisayar yazılımları sayesinde olmuştur (Le-Guillou, 2020). Bu sayede Covid-19 aşısı hızlı bir şekilde geliştirilebilmiş ve insanlığın hizmetine sunulabilmiştir (McGregor, 2020). Pandemi sürecinde yazılımların uzaktan öğretim faaliyetleri açısında sergilediği rolü de unutmamak gerekir. Bu süreçte yazılımlar kullanılarak canlı dersler yapılmış, öğrenme yönetim sistemleri aracılığıyla öğrencilerin ödev ve sınav işlemleri takip edilmiş; bunun yanında öğrencilerle etkileşim sağlanmıştır.

Bilgisayar bilimlerinin üstlendiği süreçleri otomatikleştirmek, iletişimi yaygınlaştırmak, daha iyi ürün ve hizmetler sağlamak, dünyanın daha üretken olmasını kolaylaştırmak gibi sorumlulukları, beraberinde insanoğlunun yazılımlara daha fazla bağımlı olmasına neden olmuştur (Santos, Tedesco, Borba ve Brito, 2020). Bunun bir sonucu olarak tüm gelişmiş ve gelişmekte olan ülkelerin, varlıklarını sürdürebilmeleri ve daha iyi bir konuma yükselebilmeleri için kullanılan yazılımların bakımlarını yapabilecek; aynı zamanda karşılaşılacak yeni sorunlara etkin çözümler üretebilecek nitelikli bireylerin yetiştirilmesi görevini üstlenmesini zorunlu hale getirmiştir (Demirer ve Sak, 2016). Bu görevin başarılı olma koşullarından birisi de öğrencilere Papert'ten (1980) itibaren önemi vurgulanan; iyi eğitimli ve bilgili bir vatandaş olabilmenin gereklerinden birisi olarak kabul edilen (Al-Makhzoomy, 2018; Guzdial ve DiSalvo, 2013) programlama becerisinin kazandırılması ile mümkündür (Horses, 2016; Kert ve Uğraş, 2009). Bu sayede yükseköğretimdeki akademik

iklim öğrencilerin sadece yazılım alanındaki gelişmeleri öğrenmesini kolaylaştırmayacak; aynı zamanda öğrencileri teknolojik gelişmelerin ivmesine ayak uyduran ve inovasyon yapan bireyler olarak donatacaktır (Smaragdina, Nidhom, Soraya, Ningrum ve Akbar, 2020). Bu yükümlülüklerin bilincinde olarak dünyanın birçok ülkesindeki öğretim kurumları müfredat politikalarında programlama öğretiminin önemini vurgulayan girişimlerde bulunmaktadır (Montiel ve Gomez-Zermeño, 2021; Yilmaz Ince ve Koc, 2021). Çağın gerekliliklerine uygun olarak bu girişimlere nesne yönelimli programlama paradigmasının dâhil edilmesi kaçınılmaz bir hal almıştır.

Nesne yönelimli programlama paradigması, odak noktasına nesneleri koyarak gerçek dünyanın bir temsilini oluşturmayı amaçlayan bir yaklaşımdır (Corral, Balcells, Estévez, Moreno ve Ramos, 2014; Wong ve Tan, 2016). Bu paradigma, yaşadığımız gerçek dünyada zaten varlıklarını sürdürmekte olan nesnelerin *özellik (attribute)* ve *davranışlarını (behavior)* programlama dillerini vasıtasıyla modelleyerek mesajlar veya işlevler gibi nesneler arası etkileşimler yoluyla programlama problemlerine çözüm üretmeyi hedeflemektedir (Aniche, Yoder ve Kon, 2019; Hillar 2015). 1960'ların başlarına kadar uzanan ve o zamanlar birkaç MIT projesinde yer alan programlamadaki nesne kavramı giderek önem kazanmış; günümüzde kullanılan en popüler 10 programlama dilinden yedisinin nesne yönelimli olmasına kadar evrilmiştir (Aniche, Yoder ve Kon, 2019). Nesne yönelimli programlama paradigması; (1) programlama genel yazılım geliştirme süresinin azaltılması, (2) önceden yazılan kodların yeniden kullanılabilmesini sağlaması ve (3) kod organizasyonu esnekliği sunması gibi sağladığı pek çok avantajdan dolayı günümüzde hem yazılım endüstrisinde hem de programlama öğretiminde kritik bir role sahip olmuştur (Abbasi, Kazi, Kazi, Khowaja ve Baloch, 2021; Brito ve Medeiros, 2019; Dadson, 2021). Ancak NYP öğretiminin bazı sorunlar barındırdığı bilinen bir gerçektir. Bu sorunlar; (1) genel olarak programlamanın zorluklarından, (2) nesne yönelimli programlamanın kendine özgü özelliklerinden ve gerektirdiği becerilerden, (3) NYP öğretiminde kullanılan yöntem ve materyallerden ve (4) NYP derslerindeki ödev ve projelerin karakteristiklerinden kaynaklanmaktadır.

NYP öğretimindeki sorunların ilki genel olarak bilgisayar programlamanın zorluklarından kaynaklanmaktadır. Programlamayı öğrenmek, bu konuda deneyimi olmayan birçok öğrenci için zordur (Esteves, Fonseca, Morgado ve Martins, 2009). Bir bilgisayar programını oluşturabilmek için öğrencilerin problem çözme, mantıksal ve matematiksel düşünme gibi üst düzey bilişsel becerilere sahip olmaları gerektirmektedir (Korkmaz ve

Altun, 2014; Malik, Mathew, Al-Nuaimi, Al-Sideiri ve Coldwell-Neilson, 2019). Bunun yanında öğrencilerin bir programlama diline ait sözdizimini bilmeleri ve mevcut kodları anlayabilmeleri şarttır (Perera, Tennakoon, Ahangama, Panditharathna ve Chathuranga, 2021). Bir problemi analiz edebilmek, probleme yönelik çözüm üretebilmek, oluşturulan çözümü bir programlama dilini kullanarak ifade edebilmek, oluşturulan çözümü test edebilmek ve ortaya çıkan hataları düzeltebilmek de programlamanın gerekliliklerindendir (Cheah, 2020). Bu gereklilikler neticesinde öğrenciler sürekli çabalamak ve sıkıcı bir deneme yanılma sürecine dâhil olmak zorundadır (Jiau, Chen ve Ssu, 2009). Programlama kavramların karmaşık, bilişsel olarak zorlayıcı ve öğrencilerin alışık oldukları ders içeriklerinden radikal bir şekilde farklı olması nedeniyle alanyazındaki pek çok çalışmada öğrencilerin programlamayı öğrenmede genellikle zorluk yaşadığı belirtilmektedir (Anfurrutia, Álvarez, Larrañaga ve López-Gil, 2017; Korkmaz ve Altun, 2014). Gerçekleştirilen çalışmalar öğrencilerin değişkenler, veri yapıları, karar mekanizmaları veya bellek adresleri gibi en temel programlama kavramları gibi konuları anlamakta zorlandıklarını belirtmektedir (Lahtinen, Mutka ve Jarvinen, 2005; Miliszewska ve Tan, 2007).

NYP öğretimindeki sorunların ikincisi, nesne yönelimli programlamanın kendine has özelliklerinden ve gerektirdiği becerilerden kaynaklanmaktadır. Bu kapsamda NYP paradigmasının sınıf, nesne, kapsülleme, soyutlama, kalıtım, arayüzler, nesneler arası mesaj iletimi ve çokbiçimlilik gibi birbiriyle örtüşen ve öğrenciler açısından kimi zaman birbirinden ayırt edilmesi güç olan pek çok soyut kavramı barındırması, bu disiplinin teorik karmaşıklık ile karakterize edilmesine yol açmıştır (Abbasi, Tabassum, Kazi, Tunio ve Qureshi, 2021; Livovský ve Porubän, 2014). NYP dersini almakta olan öğrencilerin programlama beceri düzeyleri ve sahip oldukları sınırlı zaman çerçevesi göz önüne alındığında, NYP disiplini yüksek derecede soyutlama içeren ve teori ile pratik arasında denge kurmayı zorlaştıran bir alan olarak betimlemektedir (Al-Nuaim, Allinjawi, Krause ve Tang, 2011; Gainutdinova, Denisova ve Shirokova, 2019). Alanyazındaki çalışmalar öğrencilerin NYP derslerinde yer alan bu soyut kavramları anlamada; bu kavramların gerçek hayattaki karşılıklarını özümsemede; büyük resmi görerek sınıflar arasındaki ilişkileri tanımlamada; yazdıkları kodlardaki hataları bulup düzetmede ve NYP'de yer alan yapıları kullanarak düzgün çalışan bir yazılım geliştirmede çeşitli zorluklarla karşılaştıklarını belirtmektedir (Gainutdinova ve diğerleri, 2019; Seng ve Yatim, 2014). Bunun bir sonucu olarak öğrencilerin NYP derslerinde; (1) nesnelerin nasıl ve ne zaman oluşturulduğu (Al-

Nuaim ve diğerleri, 2011), (2) özelliklere ait parametrelerin nasıl değer aldığı (Miller, Settle ve Lalor, 2015), (3) nesne ve metotlar arasında nasıl bir ilişki olduğu (Abbasi ve diğerleri, 2021), (4) nesneler arası mesaj iletiminin nasıl gerçekleştiği (Esteves ve Mendes, 2004), nesne ve sınıf kavramlarının farklılıkları (Biju, 2013), (5) nesne oluşturmanın bir süreci olarak aşırı yüklemeli kurucu metotların nasıl çağrılması gerektiği (Biju, 2013), (6) metotların yalnızca nesneler üzerinden çağrılması gerekliliği (Sajaniemi, Kuittinen ve Tikansalo, 2008) , (7) bir nesnenin durumu (object state) ve bir işlemin nesne durumunu nasıl değiştireceği (Sanders ve Thomas, 2007) ve (8) kalıtım ve kompozisyonu birbirinden ayıran özelliklerin neler olduğu (Alhazbi ve Ismail, 2010) gibi konularda zorlandıkları ve çeşitli kavram yanılgılarına düştükleri yapılan çalışmalar tarafından ifade edilmektedir.

Öğrenciler açısından NYP paradigmasının karakteristik özellikleriyle ilgili bir diğer problemli alanı, NYP kullanılarak gerçekleştirilen yazılımlarda çok fazla sınıf ve nesnenin yer almasıyla ilgilidir (Yang, Lee, Hicks ve Chang, 2015). Ragonis ve Ben-Ari (2005) bu durumu şu şekilde açıklamaktadır: *"NYP paradigmasında her şeyden sorumlu tek bir ana (main) metot yerine birçok sınıf vardır ve program akışı, nesnelerin oluşturulmasını, nesnelere ait referansların diğer sınıfların nesnelerindeki değişkenlere atanmasını ve metotların örtük bir "this" parametresi ile nesneler üzerinden çağrılmasını barındırır. Acemi bir programcı şu gibi sorular sorabilir: Tüm bu karmaşık süreçten kim sorumlu? Programda neler gerçekleşiyor, ne zaman gerçekleşiyor ve neden?"*. Yapılan çalışmalar, NYP derslerinde öğrencilerin net bir zihinsel model elde edene kadar bellekteki nesnelerin birbirleriyle nasıl ilişki kurduğunu anlamada *ve program akışını (program flow)* takip etmede zorlandıklarını göstermektedir (Corral ve diğerleri, 2014; Xinogalos, 2015).

NYP öğretiminde karşılaşılan sorunların üçüncüsü NYP derslerinde kullanılan yöntem ve materyallerle ilgilidir. Öğrencilere programlamayı ilk olarak prosedürel programlama ile öğretme stratejisinin NYP açısından problemli olduğu bilinen bir gerçektir (Corral ve diğerleri, 2014; Kanaki ve Kalogiannakis, 2018). Çalışmalar; programlamaya prosedürel programlama ile başlayan öğrencilerin daha önceki alışkanlıklarının bir sonucu olarak yazılımları yürütülen talimatlardan ve program akışını yönlendiren kontrol yapılarından ibaret olduğunu düşündüklerini belirtmektedir (Denny ve diğerleri, 2022). Bu öğrenciler, kodların bir başlangıcı ve bir sonu olması gerektiğine ve kodları baştan aşağı okuyarak bir yazılımın anlaşılabileceğe inanmaktadır (Qian ve Lehman, 2017). Öğrencilerin önceden elde ettikleri bu prosedürel programlama deneyimi, NYP paradigmasını öğrenmeye geçişte

kavram yanılgılarına neden olmakta ve NYP öğretiminin etkinliğini zayıflatmaktadır (Kanaki, ve Kalogiannakis, 2018; Sorva ve Seppälä, 2014).

NYP öğretiminde kullanılan yöntemin bir diğer problemli alanı, öğretim programında NYP paradigmasının kavram ve mekanizmalarına ağırlık vermek yerine derste kullanılan programlama dilinin sözdizimsel özelliklerine yoğunlaşılmasıyla ilgilidir (Lin, Yeh ve Tan, 2022). Bu durumun doğal bir sonucu olarak NYP konusunda deneyimsiz öğrenciler programlama problemlerinin çözümüne ve kavramların edinilmesine odaklanmak yerine programlama dilinin sözdizimsel ayrıntılarıyla uğraşmak zorunda kalmaktadır. Programlama diline aşırı yoğunlaşılması, öğrencilerin NYP kavramlarının program geliştirme esnasında pratiğe dökülmesiyle ilgili yeterli becerilere sahip olmasını engellemektedir (Kurniawan, Jégourel, Lee, De Mari ve Poskitt, 2021; Özdener, 2008). NYP kullanarak problem çözme becerisi gelişmeyen öğrenciler de karşılaştıkları problemleri çözememe neticesinde derse karşı ilgilerini kaybetmekte ve programlamadan kaçınmaktadır (Hooshyar ve diğerleri, 2021). Yöntem ve materyaller bağlamında diğer problem alanları ise (1) NYP öğretiminde kullanılan yazılım geliştirme ortamlarının profesyonel yazılım mühendisleri için geliştirilmiş olmasından dolayı öğrencilerin bunları kullanırken zorlanmaları (Aleksandrovna ve Nikolaevich, 2019); (2) NYP öğretiminde kullanılan programlama dillerinin karmaşık olmasından dolayı NYP'yi öğrenme sürecini uzatmaları (Livovský ve Porubän, 2014); ve (3) geleneksel programlama öğretiminde kullanılan basılı kitaplar gibi statik öğretim materyallerin NYP'nin dinamik yapısını öğrenmede etkisiz kalmasıdır (Zhang, Zhang, Stafford ve Zhang, 2013)

NYP'nin barındırdığı son sorun kategorisi ise NYP derslerindeki ödev ve projelerle ilgilidir. Günümüzde NYP derslerinde kullanılan pedagojik yaklaşımlar ağırlıklı olarak laboratuvar görevlerinin, sınavların ve projelerin üzerine yoğunlaşmaktadır (Thuné ve Eckerdal, 2019). Ancak bu tür ders etkinliklerinde öğrencilere gerçek hayattaki uygulamalardan kopuk programlama problemleri sunulmaktadır (Vihavainen, Airaksinen ve Watson, 2014). Örneğin Carbonaro, Szafron, Cutumisu ve Schaeffer'e (2010) göre programlama ödevleri sayıların ya da metinlerin sıralanması, algoritmaların test edilmesi ya da dosya oluşturma gibi öğrencilere hiçbir şey ifade etmeyen konular içermektedir. Wood ve Keen (2015) bu sorunla ilgili şu açıklamayı yapmaktadır: *"NYP derslerinde öğrencilerden NYP kavramlarını öğrenmeleri, bunları programlama çözümlerinde kullanmaları, programlama dilinin detaylarına vakıf olmaları, yazdıkları kodlardaki hataları gidermeleri gibi pek çok beceriyi aynı anda kazanmaları beklenmektedir. Bu ezici*

yükü hafifletmek adına öğretmenler derslerde NYP kavramlarının birer birer tanıtıldığı küçük programlama ödevlerine güvenirler. Bu yaklaşım pek çok problemi de beraberinde getirir çünkü bu ödevler genellikle gerçek dünya problemleriyle ilişkili değildir. Bunun bir sonucu olarak öğrencilerin derse karşı motivasyonları düşer ve katılımları azalır." Wood ve Keen'in (2015) açıklamasıyla uyumlu olarak alanyazında yapılan çalışmalar, günlük yaşamlarında gelişmiş arayüzlere ve oyun gibi karmaşık yazılımlara alışkın olan günümüz öğrencilerinin, programlama derslerinde komut satırları içeren ya da basit diyalog pencereleri barındıran programlama ödev ve projeleriyle uğraştıklarında programlamayla ilgili beklentilerinin karşılanmadığını; bu öğrencilerin derste çözülmekte olan örneklerin çok fazla çaba ve ders dışı zaman gerektirdiğini ancak gerçek hayatla çok az ilişkili olduğunu düşündüğünü ve derse karşı ilgilerini yitirdiklerini belirtmektedir (Abbasi ve Kazi, 2020; Wong ve Yatim, 2018).

Yukarıdaki paragraflarda bahsedilen NYP öğretiminin barındırdığı sorunları en aza indirebilmek ve NYP paradigmasını öğrencilere etkili bir şekilde aktarabilmek için alanyazında bazı çözüm önerileri yer almaktadır. "*İlk olarak nesne (objects-first)"* yaklaşımı bu çözüm önerilerinden birisidir. Prosedürel programlamanın temel kavramlarını incelemeden önce nesne ve sınıf kavramlarıyla çalışmayı temel alan (Livovský ve Porubän, 2014) bu yaklaşımın sayesinde, öğrencilerin prosedürel programlamadan NYP'ye geçişini ifade eden "*daha sonra nesne" (objects-later)* paradigmasında yaşadıkları sorunların en aza indirilebileceği savunulmaktadır (Uysal, 2012). Diğer bir öneri ise Karel J. Robot, CoffeeDregs, Bluej, Jeliot 3, AguiaJ, JavelinaCode, Fujaba ve Greenfoot gibi program görselleştirme sağlayan araçlarının kullanılmasıdır (Bakar, Mukhtar ve Khalid, 2019; Kölling, 2018). Çalışmalar, bu araçlar sayesinde programların nasıl davranacakları, nesnelerin ne tür ilişkilere sahip oldukları ve nesnelerin durumlarında hangi tür değişimlerin yaşanacağı gibi konuların UML diyagramları gibi statik diyagramlar aracılığıyla görselleştirilmesinin sağlandığını; bunun da öğrencilerin zihinlerinde gerçekleştirmeleri gereken karmaşık bir simülasyonun yükünü azalttığını belirtmektedir (Koniukhov, 2018; Yang, Lee, Hicks ve Chang, 2015). NYP öğretimin sorunlarını azaltmak için mevcut olan diğer öneriler ise NYP derslerinde *kullanım örneklerinin (use case)* veya *sınıf nesne etkileşimi diyagramlarının* (Thramboulidis, 2003) kullanılması, öğretim materyali ve etkinlikleri olarak eğitici robotik kitlerine başvurulması (Çınar ve Tüzün, 2021) veya programlama dillerinin karmaşıklığından kaçınmak için Scratch gibi görsel programlama dillerinin tercih edilmesidir (Mihci ve Ozdener, 2014). NYP derslerinin verimliliğini

artırmak için alanyazında yer alan bir başka öneri ise NYP'yi oyun programlamayarak öğretme stratejisidir.

Alanyazında oyun geliştirmeye dayalı öğretim (Wang ve Wu, 2009) olarak da anılan bu stratejinin NYP derslerinin başarısını artırdığını, öğrencilerin motive olmalarına yardımcı olduğunu ve öğrenci deneyimini zenginleştirdiğini savunan birçok çalışma bulunmaktadır (Garcia, Pacheco, Méndez ve Calvo-Manzano, 2020; Scherer, Siddiq ve Viveros, 2020). İlave olarak, bu alanda yapılan araştırmalar oyun geliştirmeye dayalı programlama etkinliklerinin dersleri daha dinamik, ilgi çekici ve hedefe yönelik hâle getirdiğini göstermektedir (Khaleel ve diğerleri, 2015; Spieler, Grandl, Ebner ve Slany, 2019). Bunun sonucunda öğrenciler nesne yönelimli tasarım, tasarım kalıpları, yazılım mimarisi, kullanılabilirlik gibi yazılım mühendisliğinin en iyi uygulamalarını uygulayarak, disiplinin doğasında bulunan sorunları ve zorlukları kendi başlarına keşfetme ve çözme şansı elde etmektedir (Souza ve diğerleri, 2017).

Oyun geliştirmeye dayalı öğretim temel çıkış noktası; bir dijital oyunda yer alan tüm bileşenlerin nesnelerden meydana gelmesi; oyun mekanik ve dinamiklerinin bu nesneler arasındaki ilişkiler tarafından yönlendirmesidir (Corral ve diğerleri, 2014; Stoffová, 2019). Alanyazındaki çalışmalar, oyun projelerinin doğası gereği çok sayıda nesne etkileşimi içermesinden dolayı öğrencilere nesne modelleme ve etkileşim modelleme uygulamaları için geniş fırsatlar sunduğunu ve NYP paradigmasının öğrenilmesine yardımcı olduğunu ifade etmektedir (Skraparli, Karavidas ve Tsiatsos, 2022).

Alanyazında her ne kadar NYP derslerindeki güçlüklerin üstesinden gelebilmek için bazı öneriler bulunsa da, öğrenciler için NYP derslerinde anlamlı öğrenme deneyimleri yaratmak zorlu bir görevdir (Eckerdal, 2006; Kotsovoulou, 2020). Bu zorlu görevin ne derecede yerine getirildiği, öğrencilerin NYP derslerindeki başarılarını ve programlamaya karşı tutumlarını doğrudan etkilemektedir. Öğrencilerin NYP derslerinde yaşadıkları zorluklar, zaten problem çözme becerileri tam olarak gelişmemiş olarak dersi almaya başlayan öğrencilerin programlamaya karşı ilgilerini yitirmelerine ve motivasyonlarının düşmesine neden olmaktadır (Qian ve Lehman, 2017; Robins, 2019; Seng ve Yatim, 2014). Bu süreçte bazı öğrenciler programlamadan korkmakta (Rogerson ve Scott, 2010; Woszczynski, Haddad ve Zgambo, 2005), programlama konusunda başarılı olabileceklerine yönelik kendilerine güvenlerini ifade eden öz yeterlilik algılarını kaybetmekte (Chao, 2016), programlamaya karşı olumsuz düşünceler geliştirmekte ve programlama içeren projelerden kaçınmaktadır (Fang, 2012; Miliszewska ve Tan, 2007). Programlama derslerinde

öğrencilerin başarı yüzdelerinin düşük; dersi bırakan öğrenci sayılarının oldukça yüksek olması bu durumun en büyük göstergelerindendir (Dasuki ve Quaye, 2016; Luxton-Reilly ve diğerleri, 2019).

Programlama öğretiminde yaşanan sorunlar öğrencilerin kariyer tercihlerini büyük ölçüde etkilemektedir. Programlamaya karşı olumsuz düşünceler geliştiren öğrenciler, bilgisayar bilimleri alanını seçtikleri takdirde kariyerlerinin kodlamadan başka bir şey içermeyeceğine ve çalışmalarında yaratıcılıklarını yansıtamayacaklarına inanmaktadır (Gomes ve Mendes 2007; Yan, 2009). Bunun bir sonucu olarak öğrenciler diğer meslek gruplarına göre yıllık istihdam olanağı daha fazla olmasına rağmen veri analisti, veri madencisi, yazılım proje yöneticisi, yazılım mühendisi, yazılım testçisi, sistem analisti, sistem tasarımcısı, veritabanı yöneticileri ve ağ yöneticisi gibi programlama becerileri içeren kariyer seçeneklerini tercih etmekten kaçınmaktadırlar (Abdunabi, Hbaci, ve Ku, 2019). Günümüzde bilgisayar bilimleri bu kadar önemliyken ve bilişim sektörünün ekonomideki payı artarken, bu bilimi okumak isteyen öğrencilerin sayısındaki azalış (Böttcher, Thurner, Hafner ve Hertle, 2021; Tayebi, Gomez ve Delgado, 2021) ironik bir duruma işaret etmektedir. Gelecekte bu alanda çalışacak kalifiye eleman açığı doğması son derece muhtemeldir. Yazılım mühendisliği alanının geleceği programlamayı tam olarak anlamayan kişilere bırakılamayacak kadar önemli bir alandır.

NYP derslerinin başarılarını artırabilmek, bu derslerde öğrencilerin motivasyonlarını yükseltebilmek ve NYP öğretiminin sorunlarını en aza indirebilmek için NYP derslerinin günümüz öğrencilerin altyapılarını, karakterlerini ve öğrenme tercihlerini dikkate alarak tasarlanması gerekmektedir (Spieler ve Slany; 2018). Geçmiş dönemlerdeki öğrencilerin aksine günümüz öğrencileri öğrenmeyi farklı şekilde deneyimlemekte ve öğrenme hakkında farklı inançlara sahiptir (Matwyczuk, 2013). Günümüz dijital çağ öğrenenleri günlük hayatlarında çok çeşitli teknolojik araçları kullanan; bununla birlikte videoların, dijital oyunların ve sosyal medyanın dijital dillerini konuşan bireylerden oluşmaktadır. Bu öğrenciler basılı kaynaklar yerine dijital kaynakları; düz metinler yerine görselleri; ciddi çalışmalar yerine oyunlarla keşfederek öğrenmeyi tercih etmektedirler (Çimen ve Hangül, 2021). Bu bireyler için öğrenme, bilgi aktarımı ya da ne düşünmeleri gerektiklerinin söylenmesinden çok, yaşam boyu öğrenme becerilerinin edinilmesi ya da nasıl düşünmesi gerektiklerinin öğretilmesiyle ilgidir (Matwyczuk, 2013).

Tüm bunların ışığında, bu eylem araştırmasını gerçekleştirilmesinin temel nedeni; araştırmacının vermekte olduğu NYP dersinde alanyazında bahsedilen sorunlara benzer

sorunlar yaşaması ve derslerinde bizzat yaşadığı bu sorunları en aza indirebilme gayretidir. "*Öğrencilerimin Nesne Yönelimli Programlama dersindeki motivasyonlarını nasıl artırabilirim?",* "*Derste kullanılan ödev ve projeleri öğrenciler açısından nasıl daha anlamlı hale getirebilirim?",* "*Nesne Yönelimli Programlamaya ait kavram ve tekniklerin öğrenilmesini nasıl kolaylaştırabilirim?"* gibi kritik hususlar, NYP dersinin iyileştirilmesi sürecinde araştırmacı için itici güç olmuştur. Bu iyileştirme sürecinin odak noktasını günümüz öğrencilerin özelliklerinin ve öğrenme tercihlerinin dikkate alınması; Garris, Ahlers ve Driskell'in (2002) önerdiği gibi öğrencilerin derslerde ve ders dışı zamanlarda "*coşkulu, odaklanmış ve meşgul*" olmalarının sağlanması; öğrencilerin "*kasıtlı uygulama (deliberate practice)"* (Ericsson, Krampe ve Tesch-Römer, 1993) yapmaya yönlendirilmesi teşkil etmiştir. Bunu gerçekleştirebilmek için araştırmacının öğretim görevlisi olarak vermekte olduğu mevcut NYP dersi oyun geliştirmeye dayalı programlama öğretimi stratejisi barındıracak şekilde revize edilmiştir. Mevcut NYP dersinde öğrencilere zengin bir NYP deneyimi yaşatabilmek maksadıyla dijital oyunların (1) doğası gereği çok sayıda etkileşimli nesne içererek nesne ve etkileşim modelleme konusunda zengin bir NYP deneyimi sunması, (2) günümüz öğrencileri tarafından sıklıkla oynanması sebebiyle bilinen bir proje alanı olması, (3) içerdiği eğlence unsurunun yazılım projesi boyunca yüksek düzeyde bir coşkuyu beslemeye ve öğrencilerin ilgilerini sürdürmeye yardımcı olması gibi karakteristik özelliklerinden yararlanılması amaçlanmıştır. Bu iyileştirme sürecinin sonunda öğrencilerin NYP paradigmasını öğrenmelerinin kolaylaşması; NYP dersine karşı olan ilgi ve motivasyonlarının yükselmesi; başarmışlık duyguları ve memnuniyetlerinin artması ve programlamaya yönelik kaygılarının azalması öngörülmüştür. Bu eylem araştırmasında elde edilen deneyimlerin Nesne Yönelimli Programlama dersini veren alan uzmanlarına katkı sağlaması, bu alanda ileride yapılacak çalışmalara ışık tutması beklenmektedir.

1.2. Amaç

Bu tez çalışmasının temel amacı; araştırmacının öğretim elemanı olduğu Türkiye'deki bir meslek yüksekokulunun Bilgisayar Teknolojisi bölümü ikinci sınıfında okutulan NYP dersinde yaşanan güçlükleri oyun geliştirmeye dayalı öğretim stratejisi kullanarak en aza indirmek; bu bağlamda mevcut NYP dersini uzman görüşleri ve alanyazın desteği ile yeniden tasarlamak, geliştirilen dersin ne derecede etkili olduğunu belirlemek ve bu süreçte elde edilen deneyimleri paylaşmaktır. Tez çalışmasının amaçları doğrultusunda;

- Mevcut NYP dersinin iyileştirilmesi amacıyla daha önce dersi alan mezun öğrencilerin ve derse giren öğretim elemanlarının yaşadıkları güçlüklerin tespit edilmesi,
- Mevcut NYP dersine oyun geliştirmeye dayalı öğretim stratejisinin nasıl entegre edilebileceği konusunda daha önce dersi alan mezun öğrencilerden ve derse giren öğretim elemanlarından önerilerin alınarak hangi tür oyunların ve ne tür platformların derste kullanılabileceğine yönelik görüşlerin elde edilmesi,
- Elde edilen öneriler, uzman görüşleri ve alanyazın taraması rehberliğinde mevcut NYP dersi oyun geliştirmeye dayalı programlama öğretiminin bir türü olan mevcut bir oyuna eklentiler yapma (game modding) stratejisini temel alacak şekilde yeniden tasarlanması,
- Bu öğretim tasarımı aşamasında öğrencilerin önceki programlama deneyimleri (bildikleri programlama dilleri, kullandıkları yazılım geliştirme ortamları vb.) dikkate alınarak bir oyun prototipinin ders materyali olarak belirlenmesi,
- Tasarlanan NYP dersinde gerçekleştirilecek programlama etkinliklerinin, bu oyun prototipine yapılacak eklentiler (puan sistemi, nesne toplama, engellerden kaçma vb.) bağlamında oluşturulması,
- Tasarlanan NYP dersinin birinci döngüde uygulanması,
- Birinci döngünün uygulanması aşamasında karşılaşılan sorunlar, bunun yanında öğrencilerden elde edilen görüşler doğrultusunda tasarlanan NYP dersinin revize edilmesi,
- Revize edilen NYP dersinin ikinci döngüde tekrar uygulanması,
- Her iki döngüde uygulanan öğretim programlarının etkinliği ve kullanılabilirliğinin, NYP dersinin kazanımları, bununla birlikte programlamaya yönelik motivasyon, programlama kaygısı ve programlama öz yeterlilik algısı bağlamında değerlendirilmesi,
- Katılımcıların geliştirilen NYP dersi hakkındaki görüşlerinin belirlenmesi,
- Revize edilen NYP dersinin geliştirilmesi gereken yönleri betimlenerek ileride yapılabilecek çalışmalar için önerilerin sunulması hedeflenmiştir.

Bu tez çalışmasının temel amacı ve hedefleri doğrultusunda cevapları aranan araştırma soruları aşağıda sunulmuştur.

1.2.1. Araştırma Soruları

Bu tez çalışması, iki eylem araştırması döngüsü kapsayacak şekilde gerçekleştirilmiştir. Eylem araştırmasının döngüleri *analiz, tasarım, uygulama ve değerlendirme* aşamalarını kapsayacak şeklinde kategorize edilmiştir. Araştırmacının döngülerine ait araştırma soruları Tablo 1.1'de sunulmuştur.

Tablo 1.1. Eylem araştırmasının döngülerine ait araştırma soruları

Araştırma Döngüsü	Eylem Araştırması Aşaması	Eylem Araştırması Planlama Basamağı	Araştırma Sorusu
BİRİNCİ DÖNGÜ	***Analiz***	Adım 1. Problemin Belirlenmesi	1.1. Öğrencilerin mevcut NYP dersi hakkındaki görüşleri nelerdir?
			1.2. Mevcut NYP dersinde öğrencilerin ve öğretim elemanlarının yaşadıkları sorunlar nelerdir?
		Adım 2. Bilgilerin Toplanması	1.3. Öğrenciler mevcut NYP dersinde ne tür programlama uygulamalarını geliştirmenin kendileri için daha faydalı olacağını değerlendirmektedir?
		Adım 3. Bilgilerin Yorumlanması	1.4. Mezun öğrencilerin mevcut NYP dersinde oyun geliştirmeye dayalı öğretimin uygulanabilirliği konusundaki değerlendirmeleri nelerdir?
	Tasarım	Adım 4. Eylem Planının Oluşturulması	1.5. NYP dersi Türkiye'deki diğer MYO'larda hangi içeriklerle verilmektedir?
			1.6. Mevcut problemleri ortadan kaldırmak için oyun geliştirmeye dayalı öğretim NYP dersine nasıl entegre edilebilir?
	Uygulama	Adım 5. Eylem Planının Uygulanması	1.7. Eylem planının uygulanması sürecinde katılımcıların ve araştırmacının yaşadığı sorunlar nelerdir?
	Değerlendirme	Adım 6. Sonuçların Değerlendirilmesi	1.8. Eylem planının uygulanmasından sonra katılımcıların nesne yönelimli programlama bilgi ve beceri düzeylerinde nasıl bir değişim meydana gelmiştir?
			1.9. Geliştirilen NYP dersine yönelik katılımcıların görüşleri nelerdir?

(Devam ediyor)

Tablo 1.1. Eylem araştırmasının döngülerine ait araştırma soruları (Devam)

İKİNCİ DÖNGÜ	***Tasarım***	Adım 7. Eylem Planının Revize Edilmesi	2.1. Birinci döngüden elde edilen deneyimler ışığında tasarlanan NYP dersinde ne gibi değişiklikler yapılabilir?
	Uygulama	Adım 8. Revize Edilen Eylem Planının Uygulanması	2.2. Eylem planının uygulanması sürecinde katılımcıların ve araştırmacının yaşadığı sorunlar nelerdir?
	Değerlendirme	Adım 9. Sonuçların Değerlendirilmesi	2.3. Eylem planının uygulanmasından sonra katılımcıların nesne yönelimli programlama bilgi ve beceri düzeyleri nasıl bir değişim göstermiştir? 2.4. Eylem planının uygulanmasıyla birlikte katılımcıların psikolojik faktörlerinde (motivasyon, kaygı, öz yeterlilik algısı vb.) nasıl bir değişim meydana gelmiştir? 2.5. Geliştirilen NYP dersine yönelik katılımcıların görüşleri nelerdir?

1.3. Önem

Yazılımlar günlük hayatımızda sıklıkla kullanılmaktadır. Günümüzde internet üzerinden alış veriş yapılmakta, dizi/film seyredilmekte, çok kullanıcılı oyunlar oynanmakta, haberler takip edilmekte ve haberleşme sağlanmaktadır. Yazılımların giderek artan önemiyle uyumlu olarak gelişen teknolojileri barındıran programlar geliştirebilecek nitelikli bireylere ihtiyaç bulunmaktadır. Günümüzde yazılımların ağırlıklı olarak NYP paradigması kullanılarak geliştirilmesi nedeniyle bilgisayar bilimleri öğrencilerin NYP disiplini konusunda beceriler kazanması kritik öneme sahiptir. Ancak alanyazındaki çalışmalar öğrencilerin NYP derslerinde pek çok zorluğun üstesinden gelmek zorunda olduklarından dolayı aşırı zorlandıklarını; bunun bir sonucu olarak dersi bırakma oranlarının yüksek, başarı oranlarının düşük olduğunu belirtmektedir (Susanti, 2021; Wilcox ve Lionelle, 2018).

Programlama derslerinin öğrencilerin ilgisini çekmekte zorlanması, dersten başarısız olan öğrencilerin sayısının oldukça fazla olması ve öğrencilerin derse karşı motivasyonlarının düşük olması gibi sorunlar (Beaubouef ve Mason, 2005; Fletcher ve Lu, 2009), NYP öğretiminde doğru bir öğretim stratejisinin kullanılıp kullanılmadığı konusundaki tartışmaların hala güncel olduğunu göstermektedir. Alanyazında yer alan tartışmalar doğrultusunda bu tez çalışmasında; NYP öğretimini kolaylaştırması, öğrencilerin derse karşı ilgi, motivasyon ve programlama öz yeterlilik algılarını artırması, aynı zamanda programlama kaygılarını azaltması beklenen oyun geliştirmeye dayalı programlama öğretiminin mevcut NYP dersine entegrasyonunu içeren bir çözüm önerilmektedir. Bu

araştırma, öğretiminde çok çeşitli zorluklar barındıran ancak aynı zamanda tüm dünyada önemi giderek vurgulanan NYP öğretimini konu alıyor olması; öğrencilerin NYP dersinde karşılaştıkları sorunları en aza indirmeye çalışması; NYP öğretimini oyun geliştirmeye dayalı öğretim stratejisi kullanılarak kolaylaştırmayı hedeflemesi ve bu stratejinin sonuçlarını yansıtması bakımından güncel, işlevsel ve özgündür.

Bu doktora tezinde geliştirilen etkinliklerin ve elde edilen bulguların, programlama eğitimcilerine, araştırmacılara, eğitim kurumları ve öğretim tasarımcılarına oyun geliştirmeye dayalı programlama öğretiminin bir öğretim stratejisi olarak uygulanabilirliği ve etkinliği konusunda ışık tutacağı düşünülmektedir. Yükseköğretimde oyun geliştirmeye dayalı programlama öğretimi nispeten yeni bir alandır ve bu yaklaşımın etkililiğini değerlendirmek için daha fazla araştırmaya ihtiyaç olduğu aşikârdır (Chen, Siau ve Nah, 2012; Hewett, 2016). Türkçe alanyazında da bu yöntemin kullanımına yönelik çalışmalar sınırlı sayıdadır. Çalışmadan elde edilen deneyimler vasıtasıyla çalışmada kullanılan oyun geliştirmeye dayalı programlama öğretimi yönteminin öğrencilerin NYP dersine yönelik motivasyon, öz yeterlilik algısı ve programlama kaygısı gibi psikolojik değişkenlerine etkisi ortaya çıkarılacak ve bu programlama alanyazınına katkı sağlayacaktır. Bununla birlikte elde edilen bulguların bu yöntemin sağladığı üstünlükler ve sınırlılıklar konusunda nitel veriler sağlayacak olması; programlama öğretiminde sıklıkla kullanılan yöntemlere farklı bir bakış açısı kazandırması, öğrencilerin ve araştırmacının elde ettikleri deneyimleri aktarması açısından oldukça değerlidir.

1.4. Sınırlılıklar

Nesne Yönelimli Programlama dersinin geliştirilmesini hedefleyen bu çalışmanın sınırlılıkları aşağıda sıralanmıştır:

- Araştırmanın gerçekleştirildiği dönemde yaşanan Covid-19 pandemisi çalışmanın en önemli sınırlılığını oluşturmaktadır. Normalde 15 hafta olarak planlanan araştırmanın birinci döngüsü pandemi sebebiyle kısaltılarak sekiz hafta ile sınırlandırılmıştır. Bu sebeple her ne kadar planlandığı gibi yüz yüze eğitimle tamamlansa da araştırmanın birinci döngüsünde “Kalıtım” ile “Hata Kotarımı” konuları işlenememiştir. Bu konuların işlenememesi nedeniyle bu konularla ilgili kazanımlara ait sorular, birinci döngüde kullanılan NYP Başarı Sınavında yer almamıştır.

- Pandemi sebebiyle yüz yüze derslerde uygulanan fiziksel mesafe ve maske kullanımı kuralları öğrencilerle araştırmacı arasındaki sınıf içi etkileşime etki eden bir sınırlılıktır.

- Yüz yüze gerçekleştirilmesi planlanan araştırmanın ikinci döngüsü pandemi sebebiyle acil uzaktan öğretim yöntemiyle gerçekleştirilmiştir.

- Katılımcı öğrencilerin meslek yüksekokulu öğretimleri sürecinde aldıkları internet programcılığı, bilişim sistemleri güvenliği, veri tabanı yönetim sistemleri gibi diğer mesleki alan dersleriyle NYP dersinin ortak herhangi bir kazanımı olmamakla birlikte; öğrencilerin bu derslerde genel kodlama ile alakalı uygulamalar yapmaları, katılımcıların olgunlaşmasında etkili olabileceği nedeniyle sınırlılık olarak tanımlanmıştır.

- Araştırmanın verileri, birinci döngü için Türkiye'deki bir meslek yüksekokulunun Bilgisayar Teknolojisi bölümü ikinci sınıfında öğrenim görmekte olan 29; ikinci döngü için ise 30 öğrenci ile sınırlıdır.

- Araştırma, birinci döngü için sekiz hafta (24 saat), ikinci döngü için ise 15 hafta (60) saat öğretim etkinliklerinin verileri ile sınırlıdır.

- Çalışmada varılan yargı ve sonuçların kapsamı; oyun geliştirme aracı olarak Unity 3D editörü, programlama dili olarak C# ile sınırlıdır.

1.5. Varsayımlar

- Katılımcıların ölçme araçlarını yanıtlarken gerçek bilgi, beceri, duygu ve düşüncelerini yansıttıkları varsayılmaktadır.

- Çalışmaya katılan öğrenciler, daha önce Algoritma ve Programlamaya Giriş ile Görsel Programlama derslerini almışlardır. Katılımcıların bu derslerden aldıkları dönem sonu notlarının gerçeği yansıttığı varsayılmıştır.

1.6. Tanımlar

Nesne Yönelimli Programlama: Bilgisayar yazılımlarını metotlar ve prosedürler yerine birbirleriyle mesajlaşan nesneler bağlamında modelleyen bir programlama yaklaşımıdır. Bu yaklaşımda yazılım dahilindeki her işlev nesneler olarak soyutlandırılır ve bu şekilde kullanılır.

Dijital Oyun: Bilgisayar, oyun konsolu, tablet veya akıllı telefonlar gibi cihazlar kullanılarak tek kullanıcılı ya da internet bağlantısı vasıtasıyla çok kullanıcılı olarak oynanan oyun türüne verilen isimdir.

Eğitsel Oyun: Belirli bir eğitim veya öğrenme amacı içeren ve esas olarak eğlence amacıyla tasarlanmayan bir oyun kategorisini ifade etmektedir.

Oyun Motoru: Kişilerin veya şirketlerin oyun yapmak amacıyla kullandığı ücretli veya ücretsiz araçlara verilen isimdir. Daha önce başka yazılımcılar tarafından oluşturulmuş kütüphanelerden meydana gelen bu yapı oyun geliştiricileri sınıfları, metotları ve nesneleri yeniden kodlama zahmetinden kurtarır. Bu sayede daha az kod yazılmasına ve zamandan tasarruf edilmesine olanak tanır.

Oyun Prototipi: Oyun prototipi, oyun motoru kullanılarak bir programlama dilinde oluşturulmuş çalışan bir oyunu ifade etmektedir. Bu çalışan oyunu oyun geliştiricileri kendi ihtiyaçlarına ve hayal güçlerine bağlı olarak istedikleri gibi biçimlendirebilirler; oyuna yeni bileşenler ekleyip oyunu değiştirebilirler.

IDE: Integrated Development Environment (Tümleşik Geliştirme Ortamı) yazılımcıların daha hızlı, verimli ve hatasız kod yazmalarına olanak tanıyan yazılım geliştirme araçlarıdır.

BÖLÜM II: KAVRAMSAL ÇERÇEVE

2.1. Programlama Öğretimi, Önemi ve Güçlükleri

Pea ve Kurland (1983) bilgisayar programlamayı "*bir bilgisayarın belirli görevleri başarabilmesi için bir dizi yazılı talimattan oluşan yeniden kullanılabilir bir ürün geliştirmeyi içeren faaliyetlerin bütünü"* olarak tanımlamaktadır (s. 5). Başka bir ifade ile programlama, programlama dilleri vasıtasıyla bilgisayarlara çeşitli durumlarda nasıl davranacağını tanıtma işidir. Bilgisayarların belirli görevleri başarabilmesi için gerçekleştirilen programlama faaliyeti algoritma oluşturma, bir programlama dilinde kod yazma, yazılan kodları araçlar sayesinde makine diline çevirme, kodlardaki hataları giderme ve oluşturulan yazılımı test etme işlemlerini barındırmaktadır (Kert ve Uğraş, 2009; Papadopoulos ve Tegos, 2012).

21.yüzyılda gerçekleşen dijital dönüşümün çok hızlı ilerlemesi ve teknolojinin hayatımızdaki etkisini giderek artırması öğrencilerin sahip olması gereken bilgi ve becerileri değiştirerek bilişim becerilerinin öğrenme ortamında öğretilmesi ve değerlendirilmesi ihtiyacını kabul eden girişimlere neden olmuştur (Montiel ve Gomez-Zermeño, 2021). Ülkeler, öğrencilerin bu yeni becerileri kazanmalarını sağlamak için eğitim sistemlerinde değişiklik yapma ihtiyacı hissetmişlerdir (Oluk, Korkmaz ve Oluk, 2018). Tıpkı günlük hayatta sürekli kullanılan aritmetik, okuma, yazma vb. becerilerin kazandırılması gerekliliği gibi, günümüz şartlarına uyum sağlayabilmeleri için tüm öğrencilere kendi yaş gruplarına uygun şekilde programlama eğitiminin verilmesi gerektiğini savunan araştırmacılar vardır (Guzdial ve Disalvo, 2013; Kalelioğlu ve Gülbahar, 2014).

Programlama öğretimi öğrencilere problem çözme becerilerinin kazandırıldığı; değişkenler, döngüler, diziler, metotlar, veri yapıları, nesne yönelimli programlama gibi temel programlama kavramlarının tanıtıldığı; programlama dillerinin sözdiziminin gösterildiği ve bir programlama dili kullanarak programlama problemlerinin çözüldüğü eğitime verilen isimdir (Medeiros, Ramalho ve Falcão, 2018). Bunun yanında Seralidou ve Douligeris'e (2021) göre programlama öğretimi birçok problemi çözebilmek için bir programlama diline ait kontrol ve veri yapılarını öğrenmeyi; program oluştururken hafıza yönetimi, veri giriş/çıkışı vb. konularda etkili stratejileri kullanmayı kapsamaktadır. Benzer şekilde Lahtinen ve diğerlerine (2005) göre programlamayı öğrenmek; soyut kavramları

doğru bir şekilde kavramayı, doğal dilden farklı olan programlama dillerinin sözdizimine hâkim olmayı, üretilecek çözümü gerekli bilgisayar eylemlerine dönüştürmek için mantıksal akıl yürütmeyi gerektirir. Bilişim sistemleri açısından değerlendirildiğinde bilgisayar programlamanın pratik bilgisi, üst düzey bilgisayar dersleri ve konuları için bir ön koşuldur (Ayalew, Tshukudu ve Lefoanea, 2018).

McGill ve Volet (1997) programlama öğretimi sürecinde öğrencilerin edinmesi gereken birbiriyle ilişkili üç temel programlama bilgisini; *a) kavramsal bilgi (conceptual knowledge), b) sözdizimsel bilgi (syntactic knowledge),* ve *c) stratejik bilgi (strategic knowledge)* olarak açıklamaktadır (Tablo 2.1). *Kavramsal bilgi*, programlamada kullanılan yapıların ve ilkelerinin anlaşılmasıyla ilgilidir ve zihinsel modellerinin geliştirilmesinin yanı sıra program tarafından yürütülen eylemlerin ne anlama geldiğini kavramayı içerir. *Sözdizimsel bilgi*, programlama diline has özellikleri ve bu özelliklerin kullanım kuralları hakkındaki bilgilerin bütünüdür. *Stratejik bilgi* ise programlama problemlerini çözebilmek amacıyla kavramsal ve sözdizimsel bilgiyi en etkili şekilde kullanma yeteneği olarak tanımlanmaktadır. Bu üç programlama bilgisi kendi arasında *bildirimsel (declarative)* ve *prosedürel (procedural)* olmak üzere ikiye ayrılmaktadır (Mcgill ve Volet, 1997).

Tablo 2.1. Programlama öğretim sürecinde kullanılan bilgi türleri (Mcgill ve Volet, 1997)

	Bildirimsel Bilgi	**Prosedürel Bilgi**
Kavramsal Bilgi	Bir program yürütülürken gerçekleşen eylemlerin anlamlarını anlama ve açıklama becerisi	Programlama problemlerine çözüm tasarlama becerisi
Sözdizimsel Bilgi	Bir programlama diline ait sözdizimsel öğelerin bilgisi	Kod yazarken programlama diline ait sözdizimi kurallarını uygulayabilme becerisi
Stratejik Bilgi	Yeni bir problemi çözmek için bir programı tasarlama, kodlama, test etme ve hata giderme becerisi	

Yurdugül ve Aşkar'a (2013) göre bildirimsel bilgi; belirli gerçekler, kavramlar ve ilkeler hakkında bilgilerin tamamıdır. Bunun yanında prosedürel bilgi; bu bildirimsel bilgiyi kullanarak problem çözebilme işlemidir. Başka bir ifade ile prosedürel bilgi bir bakıma *"nasıl yapılır (know how)"* bilgisidir. Yurdugül ve Aşkar'a (2013) göre en önemli bilgi stratejik bilgidir çünkü bu bilgi, programlama problemlerine çözüm oluşturabilmek için sözdizimsel ve kavramsal bilgiyi en uygun ve etkili şekilde kullanma becerisini barındırmaktadır (Mcgill ve Volet, 1997).

Programlama, öğrenilecek farklı bilgi türlerini barındırmasına ilave olarak bir programlama problemini çözebilmek için bazı işlem basamaklarını da içermektedir. Dale ve

Weems'e (2005) göre programlama iki aşamadan oluşmaktadır. Bunlar *problem çözme (problem-solving)* ve *uygulama (implementation)* aşamalarıdır. Her bir aşama kendine has bildirimsel ve prosedürel bilgiler gerektirmektedir (Cheah, 2020). Programlama öğretiminde öğrencilere istenilen bilgi ve becerilerin kazandırılabilmesi için öğrencilerin öncelikle problem çözme basamaklarını oluşturması; daha sonra ise uygulama aşamasına geçmesi gerekmektedir.

Problem çözme aşamasında öğrencilerden beklenen problemleri ayrıntılı bir şekilde analiz etmeleri ve problemlerin çözümüne yönelik algoritmaların tasarlamalarıdır. Bu aşamada ne tür veri yapıları kullanılacağı, programın hangi karar mekanizmaları ve döngüleri barındıracağı, doğru sonuçlar üretebilmek için fonksiyonların hangi sırada çalıştırılacağı belirlenir (Cheah, 2020). *Uygulama aşamasında ise* problem çözme aşamasında gerçekleştirilen analizler ve toplanan bilgiler doğrultusunda kod yazma işlemi gerçekleştirilir (Dale ve Weems, 2005). Kodların yazılıp, bilgisayar programının oluşturulmasına müteakip programın verimliliğini ölçmek için kapsamlı testler yapılır. Testlerin sonuçlarına göre oluşturulan program amaçlanan hedefleri karşılamıyorsa, daha fazla ince ayarın yapılması ve problem çözme aşamasının tekrarlanması gerekmektedir (Dale ve Weems, 2005).

Gerek programlama öğretiminde öğrenilmesi gereken bilgiler, gerekse öğrencilerin bir programı oluştururken tamamlaması gereken süreçler değerlendirildiğinde programlama öğretimi hem öğrenciler hem de öğretmenler açısından zorlayıcı bir süreçtir (Tan, Ting ve Ling, 2009). Programlama öğretimiyle ilgili gerçekleştirdikleri sistematik derleme çalışmasında Robins, Rountree ve Rountree (2003) programlama öğretimini şu şekilde tanımlamaktadır: "*Programlamayı öğrenmek güçtür. Bu alana yeni başlayanlar çok çeşitli zorluklarla karşılaşırlar. Programlama dersleri genellikle zor olarak kabul edilir ve en yüksek ders bırakma oranlarına sahiptir* (s. 137).

Tan ve diğerlerine (2009) göre bu zorluklar birkaç nedenden oluşmaktadır. Bunlardan ilki program tasarım sürecidir. Bu aşamada öğrencilerden belirli bir programlama problemini çözebilmek için algoritma tasarlamaları beklenmektedir ancak öğrenciler, farklı işlevleri prosedürlere ayıramadıkları için bu tasarım aşamasında oldukça zorlanmaktadırlar. Öğrencilerin zorlanmasının diğer bir nedeni programlama dillerinin yapısından kaynaklanmaktadır. Öğrenciler, doğal dillere uzak olan programlama dillerini hatırlamakta ve anlamakta zorlanmaktadırlar (Bosse ve Gerosa, 2017). Son olarak oluşturulan

programlardaki hataları bulmak ve gidermek zaman alıcı ve meşakkatli bir süreç olup öğrencilerin motivasyonlarını olumsuz yönde etkilemektedir.

Benzer şekilde bu konu ile ilgili Ismail, Ngah ve Umar'ın (2010) yaptığı çalışmada programlama öğretiminde karşılaşılan dört temel sorundan bahsedilmektedir. Bu sorunların ilki öğrencilerin problemleri analiz etme ve programlama kavramlarını anlama becerisinin eksikliğidir. İkinci sorun, derslerde programlamada gerçekleştirilen problem çözme faaliyetlerinde etkisiz sunum tekniklerinin kullanımıyla ilgilidir. İsmail ve diğerlerine (2010) göre sözde kod ve akış şemaları gibi geleneksel tekniklerin kullanımı yalnızca yapılandırılmış programlamayı öğretmek için uygundur. Nesne yönelimli programlama öğretimi söz konusu olduğunda, geleneksel teknikler öğrencilere yeterli açıklama ve anlayış sağlamakta yetersiz kalmaktadır. Üçüncü sorun, problem çözme ve kodlama teknikleri için öğretim stratejilerinin etkisiz kullanımından kaynaklanmaktadır. Nesne yönelimli programlama öğretiminde geleneksel öğretim stratejileri kullanımı etkili olamamaktadır. İsmail ve diğerlerine (2010) göre öğretim elemanlarını bilişsel strateji açısından farklı paradigmaların kullanılması gerektiği konusunda hemfikirdir. Öğrencilerin kontrol ve veri akışı sürecini anlamasına yardımcı olabilmek maksadıyla uzamsal ve görselleştirme yeteneklerini destekleyen öğretim materyaline ihtiyaç vardır. Son sorun ise öğrencilerin programlama sözdizimini anlamamaları ve bunlara hâkim olmamalarıdır (Qian ve Lehman, 2017).

Gomes ve Mendes (2007) yaptıkları çalışmanın sonucunda programlama öğretiminde öğrencilerin yaşadıkları zorlukların sebeplerini şu şekilde özetlemişlerdir:

- Programlamanın yüksek bir seviyede soyut düşünme becerisi gerektirmesi,
- Programlamanın, öğrencilerin hem bilgi hem de problem çözme teknikleri konusunda iyi düzeyde olmalarını gerektirmesi,
- Programlamanın daha fazla teorik bilgiye dayanan, kapsamlı okuma veya ezber içeren derslere kıyasla farklı alanlarda yoğun bir çalışma gerektirmesi,
- Programlama derslerinin kalabalık sınıf mevcutları nedeniyle bireyselleştirilememesi ve öğrenciler kendi anlama hızlarında dersleri takip edememesi,
- Programlamanın çoğunlukla dinamik bir süreç olması ancak genellikle statik ders materyalleri kullanılarak öğretilmesi,

- Programlama derslerinde öğretmenlerin metodolojilerinin genellikle öğrencilerin öğrenme stillerini dikkate almaması,
- Programlama dillerinin karmaşık bir sözdizimine sahip olmasının yanında pedagojik motivasyonlarla oluşturulmamış olup profesyonel kullanım için olmasıdır.

Benzer problemlerin nesne yönelimli programlama öğretiminde de mevcut olduğu birçok araştırmacı tarafından ortaya konulmuştur. Yapılan çalışmalar, prosedürel programlama paradigmasından nesne yönelimli programlamaya paradigmasına geçişin öğrenciler için zorlayıcı olduğu göstermektedir (Bennedsen ve Schulte, 2007). İlave olarak, her ne kadar nesne yönelimli programlama gerçek hayattaki nesnelerle ilgili modellemeler içerdiği için doğal bir alan gibi kabul edilebilse de, bu gerçek hayat senaryolarının nesne yönelimli programlamanın sınıf, nesne, özellik, sınıf metodu, mesaj aktarımı, kalıtım, çokbiçimlilik ve kapsülleme gibi temel kavramlarıyla nasıl eşleştirileceğini öğrencilerin anlaması zordur (Abbasi ve diğerleri, 2021; Livovský ve Porubän, 2014). Benzer şekilde Sheetz, Irwin, Tegarden, Nelson ve Monarchi'nin (1997) nesne yönelimli programlama tekniklerinin öğretimde karşılaşılan güçlüklerle ilgili gerçekleştirdikleri çalışmanın sonuçlarına göre, öğrenciler nesne yönelimli programlamanın temel kavramlarını anlama konusunda algoritma tasarlama ve programlama tekniklerini konusundan daha fazla zorlanmaktadır.

Her ne kadar öğrenilmesi ve uzmanlaşması öğrenciler için zor olsa da programlama öğrenmenin öğrencilere çok sayıda bilişsel faydalar sağladığı bilinmektedir. Scherer ve diğerlerinin (2019) 105 araştırma ile yaptıkları meta-analiz çalışmasının sonuçlarına göre programlamayı öğrenmenin yaratıcı düşünme, matematiksel beceriler ve üst biliş gerektiren durumlar için olumlu transfer etkileri olduğunu göstermiştir. Aynı şekilde alanyazında programlama öğrenmenin öğrenenlerin bilişsel gelişimlerine, problem çözme, bilgisayarca düşünme ve plan yapma becerilerine olumlu katkılar sağladığını belirten çalışmalar bulunmaktadır (Gomes ve Mendes, 2007; Grover ve Pea, 2013). Akcaoglu'nun (2013) gerçekleştirdiği çalışmanın sonuçları, bilişsel beceriler açısından daha önce programlama öğrenen öğrencilerin öğrenmeyenlere göre daha iyi performans gösterdiklerini vurgulamaktadır.

Sonuç olarak programlama, belirli seviyeye ulaşabilmek için sürekli pratik gerektiren ve yaratıcılık, ayrıntılara dikkat, problem çözme becerileri ve temel mantığın benzersiz bir karışımını gerektiren farklı bir zanaattır (Oram ve Wilson, 2010). Bunun yanında

programlamayı öğrenmek, bilişsel açıdan yorucu ve zorlayıcı bir süreçtir (O'Grady, 2012). İyi bir programcı olabilmeleri için öğrencilerin programlama dilinin özelliklerini öğrenmenin ötesinde çok farklı becerileri kazanmaları gerekmektedir (Esteves ve Mendes, 2004; Lahtinen ve diğerleri, 2005). Tıpkı yazılı notaları anlayabilmesi ve müzik besteleyebilmesi için müzik teorisi dersini alan, bunun yanında öğrendiği bilgileri bir müzik aleti çalarak pekiştiren müzisyenler gibi; bilgisayar bölümü öğrencileri de programlama derslerinde edindikleri teorik bilgileri kendilerine sunulan problemler aracılığı ile bir programlama dilini kullanarak pekiştirmek zorundadır (Bayliss ve Strout, 2006). Bu süreçte öğrencilerin hem programlama diline has sözdizimi ve programlama yapıları hakkında bilgi sahibi olmaları gerekirken hem de problemleri bilgisayar kodları ile çözmek için hızlı bir şekilde problem çözme stratejileri oluşturmak zorundadır (Soares, Martin ve Fonseca, 2015). Bunları aynı anda gerçekleştirmeye çalışmak öğrenciler için zorlayıcı bir süreçtir. Yapılan çalışmalar, öğrencilerin karşılaştığı sorunların büyük bir bölümünün programlama öğretiminin başlarında ortaya çıktığını göstermektedir (Cheah, 2020). İlave olarak programlama öğretimi boyunca öğrenciler engellerle karşılaştıklarında kolayca pes edebilmektedirler (Cheah, 2020). Üniversitelerde okutulan "programlamaya giriş" dersine kayıt yaptıran öğrencilerin neredeyse üçte birinin dersi bırakması, bu dersin öğrencilerin açısından oldukça zorlayıcı olduğunun en önemli kanıtlarındandır (Yan, 2009).

2.2. Programlama Derslerinde Öğrencilerin Başarılarını Etkileyen Faktörler

Öğrenme ile ilgili alanyazın, öğrencilerin bilişsel katılımını (engagement) ve öğrenme süreçlerini pek çok faktörün etkilediğini belirtmektedir. Bunlardan bazıları öğrenenlerin bireysel özellikleri, öğrenme ortamının yapısı ve hedeflenen kazanımlardır (Carbone, Hurst, Mitchell ve Gunstone, 2009). Helme ve Clarke (2001) bireylerin bilgi, beceriler, beklentiler, algılar, ihtiyaçlar, değerler ve hedefler gibi öğrenme ortamlarına bilişsel katılımlarını etkileyen çok sayıda özelliği beraberinde getirdiğini şu şekilde ifade etmektedir: "Öğrencilerin başarılı olabilmeleri için hem iradeye (motivasyon) hem de beceriye (yetenek) sahip olmaları gerekir. Öğrenmeye motive olan ve öğrendikleri hakkında dikkatlice düşünen öğrencilerin, kapsanan materyal hakkında daha derin bir anlayış geliştirmesi, öğretmenlerin deneyimidir" (s. 136). Alanyazında bireylerin akademik performanslarını etkileyen faktörler sıklıkla araştırılmıştır (Carbone ve diğerleri, 2009). Benzer araştırmalar programlama öğretimi alanyazınında mevcuttur ve programlama derslerinde öğrencilerin başarılarını

etkileyen faktörler sıklıkla incelenen konulardan birisidir. Bu bölümde bu faktörler incelenmiştir.

2.2.1. Programlama Deneyimi

Programlama konusunda deneyimli olup olmama durumu, öğrencilerin programlama derslerindeki başarılarını etkileyen faktörlerden en sık incelenenlerden birisidir. Alanyazında programlama konusunda acemi olanlarla deneyimli olanları karşılaştıran; kullanılan programlama paradigmasının bu tür öğrencilere etkilerini inceleyen; ayrıca bu iki farklı kategoride yer alan öğrencilerin özelliklerini belirlemeye çalışan çeşitli çalışmalar bulunmaktadır. Programlama konusunda acemi olanlarla deneyimli olanları karşılaştıran çalışmalar, daha önce programlama dersi almanın üniversitedeki programlama dersi başarısı üzerinde olumlu etkiye sahip olduğunu göstermektedir (Barlow-Jones, van der Westhuizen ve Coetzee, 2014; Watson, Li ve Godwin 2014).

Kullanılan programlama paradigmasının acemi ve deneyimli öğrenciler açısından karşılaştıran çalışmaların sonuçlarında zıtlıklar bulunmaktadır. Nesne yönelimli programlama ile ilgili yapılan çalışmalar, bu paradigmanın programlamaya giriş açısından eşik kavram (treshold concept) olduğunu (Boustedt ve diğerleri, 2007) ve acemi öğrencilerin nesne ve sınıf gibi temel nesne yönelimli programlama kavramları konusunda kavram yanılgılarının varlığını göstermektedir (Boustedt ve diğerleri, 2007; Eckerdal ve Thuné, 2005). İlave olarak öğrencilerin nesne yönelimli programlama kavramlarını değişkenler, metotlar ve diziler gibi temel programlama kavramlarına göre daha zor bulduklarını gösteren çalışmalar mevcuttur (Butler ve Morgan, 2007). Bu sonuçlara zıt olarak Ventura ve Ramamurthy (2004) ile Rountree, Rountree, Robins ve Hannah'ın (2004) çalışmalarının sonuçlarına göre prosedürel dillerdeki önceki programlama deneyimi ile nesne yönelimli programlama başarısı arasında bir ilişki bulunmamaktadır. Çalışmanın yazarları, nesne yönelimli programlama dersinin başarı faktörlerinin prosedürel dillerden farklı olabileceğini ve bu etkenlerin yeniden değerlendirilmesi gerektiğini öne sürmüşlerdir.

Alanyazında acemi programcıların özelliklerini belirlemeye çalışan çalışmalar incelendiğinde; programlama ifadelerine yönelik sözdizimi ve semantiğini bilseler de acemi öğrencilerin bu yapıları kullanarak geçerli bir program kodunu yazma konusunda zorlandıklarını gösteren çalışmalar bulunmaktadır (Spohrer ve Soloway, 1986). İlave olarak acemi programcıların yazdıkları kodlardaki hataları bulmayı ve gidermeyi başarmakta zorlandıkları bilinmektedir (Lahtinen ve diğerleri, 2005). Bu durum, acemi öğrencilerin bir

programı bütünüyle anlayamadıklarını, yalnızca kod parçalarını anlamaya çabaladıklarını göstermektedir (Mannila, 2009). Benzer çalışmalar (Kölling ve Rosenberg, 1996; Lister, Simon, Thompson, Whalley ve Prasad, 2006) acemilerin program kodları hakkında yüzeysel bilgilere sahip olduklarını ve bir kodu anlamaya çalışırken "satır satır" yaklaşımını kullandıklarını belirtmektedir. İlave olarak acemi öğrenciler, yazacakları kodları planlama ve yazdıkları kodları test etmeye yeterince zaman ayırmamaktadır. Yazdıkları kodlar hatalıysa problemi çözmek için başka bir çözüm aramamakta, programı yeniden formüle etmek yerine yalnızca yerel çözümlerle düzeltmeye çalışmaktadır. Muller'e (2005) göre acemi programcıların hataların çoğu genellikle organize bilgi eksikliği ve problem çözme stratejilerinin tam olarak gelişmemiş olmasından kaynaklanmaktadır. Bunun yanında problemler arasındaki içsel benzerlikleri görme ve fikirleri bir problemden farklı bir bağlamda benzer bir probleme aktarma yeteneklerindeki eksiklikten kaynaklanmaktadır. Sonuç olarak acemi programcıların doğru ve/veya yüksek kaliteli kod yazamamasının popüler bir açıklaması, öğrencilerin bir problem tanımını soyutlama, onu alt problemlere ayırma ve parçaları eksiksiz bir çözümde yeniden birleştirme yeteneğinden yoksun olmalarıdır (Kasto, 2016).

Araştırmacılar, bir acemi programcının uzman sayılabilmesi için 10 yıllık bir deneyime sahip olması gerektiği konusunda hemfikirdirler (Robins ve diğerleri, 2003). Bu deneyim kazanma sürecinde öğrenenlerin geçirdiği aşamaların tanımlanması konusunda alanyazında çeşitli fikirler bulunsa da en çok belirtilen Dreyfus ve Dreyfus'un (1986) sınıflandırmasıdır (Robins ve diğerleri, 2003). Bu sınıflandırmaya göre öğrenciler başlangıç (novice), ileri düzeyde başlangıç (advanced beginner), yetkinlik (competence), yeterlilik (proficiency) ve uzman (expert) basamaklarından geçerek programlama konusunda acemiden uzmana olan serüvenlerini tamamlamaktadır.

Dreyfus ve Dreyfus'un (1986) sınıflandırmasına ek olarak Perkins, Hancock, Hobbs, Martin ve Simmons (1989) acemi programcılarında çeşitli sınıflandırmalara sahip olabileceğini belirtmektedir ve iki tür kategori tanımlamaktadır: durdurucular (stoppers), ve hareket edenler (movers). "Durdurucular", bir problemde nasıl ilerleyeceklerini hemen göremedikleri zaman durma ve pes etme eğiliminde olan öğrencilerdir. Perkins ve diğerleri (1989) durdurucuların özelliklerini şu şekilde açıklamaktadır:

"Acemi programcılar nasıl ilerleyeceklerini oldukça hızlı bir şekilde gördüklerinde, doğal olarak düşündükleri bu çözümü uygularlar. Bununla birlikte, net bir hareket tarzı kendini göstermediğinde, genç programcı çok önemli bir kırılma noktasıyla karşı karşıyadır: Bundan sonra ne yapmalı? ... Bazı öğrenciler

oldukça tutarlı bir şekilde en basit çözümü benimser ve sadece durur. Sorunu kendi başlarına çözme umutlarını bırakmış görünürler." (s. 265)

Durduruculara zıt olarak "hareket edenler" hatalarla karşılaştıklarında çözüm bulmayı denemeye, kodlarını değiştirmeye ve hata geri bildirimini etkin bir şekilde kullanmaya devam eder. Perkins ve diğerlerine (1989) göre hareket edenler de ikiye ayrılabilir. "Karıştırıcılar (tinkerers)" olarak da adlandırılan "aşırı hareket edenler (extreme movers)" bir hata ile karşılaştıklarında çözüm aramaya ve değişik yapmaya devam ederler ancak bunu rastgele bir şekilde yaparlar. Perkins ve diğerleri (1989) karıştırıcıların özelliklerini şu şekilde açıklamaktadır:

"Bu tür öğrenciler genellikle kurcalama dediğimiz bir yaklaşımla program oluştururlar. Bir programlama problemini bir kod yazarak ve sonra onu çalıştırma umuduyla küçük değişiklikler yaparak çözmeye çalışırlar." (s. 272)

Perkins ve diğerlerinin (1989) acemi öğrencilerle ilgili yaptığı bu çalışma öğrencilerin programlama öğrenme metot ve stillerinin başarılarını etkilediğini ortaya koyması açısından son derece değerlidir. Robins ve diğerleri (2003) göre acemi öğrencileri kapsayan bu tür sınıflandırmaların belirli bir öğrenci için geçerli olduğu düşünüldüğünde, öğrencilere sunulacak geri bildirim ve yardımın niteliği hedefe yönelik olacak; bu da öğrencilerin başarısına katkı sunacaktır.

2.2.2. Programlamaya Yönelik Öz Yeterlilik Algısı

Bandura (1986) öz yeterlilik algısını "insanların, belirlenmiş performans türlerini elde etmek için gerekli olan eylemleri organize etme ve yürütme yeteneklerine ilişkin yargıları" olarak tanımlamaktadır (s. 391). Başka bir ifade ile öz yeterlilik algısı, bireylerin bir hedefi başarabilmek için sahip oldukları yeteneklere olan inançlarını temsil etmektedir. Bandura'ya (2001) göre öz yeterlilik algısının bireyin kendi yaşantıları, dolaylı öğrenme yaşantıları, sözel ikna ve fizyolojik ve duygusal durum olmak üzere dört kaynağı mevcut olup bireylerin gerçekleştirmek istedikleri etkinlik seçimlerini, amaçlarını, hedefleri doğrultusunda gösterdikleri çabayı ve zorluklar karşısındaki direnme gücünü etkilemektedir.

Wiedenbeck'e (2005) göre öz yeterlik algısı genel öz güven gibi bir karakter özelliği veya genelleştirilmiş bir inanç olmayıp belirli bir görev veya faaliyete özgüdür. Bu açıdan değerlendirildiğinde *"bir kişi araba kullanmak gibi bir alanda yüksek öz yeterliliğe sahip olabilirken, bilgisayar programlama gibi başka bir alanda düşük öz yeterliliğe sahip olabilmektedir."* (s. 2). Bununla birlikte, birbirine yakın faaliyetler arasında, tıpkı farklı

programlama dilleri arasında olduğu gibi, öz yeterlik inançlarının transferi gerçekleşebilmektedir (Wiedenbeck, 2005).

Öz yeterlilik kavramıyla ilgili en önemli hususlardan birisi, bireylerin herhangi bir aktivitedeki performansını, bireylerin becerileri veya o alandaki önceki başarılarından çok kendi yeteneklerine olan inancı belirlemesidir (Pajares, 1996). Bu bağlamda bireylerin belirli görevleri başarabilmek için tüm becerilere sahip olmaları, başarılı olacakları anlamına gelmemektedir. Ne kadar becerili olurlarsa olsunlar bireyler bir alanda düşük öz yeterlilik algısına sahiplerse başarısız olma ihtimalleri yüksektir (Askar ve Davenport, 2009; Korkmaz ve Altun, 2014).

Yapılan çalışmalar, öz yeterlilik algısı ile akademik başarı arasında anlamlı bir ilişkinin varlığını belirtmektedir. Örneğin Robbins ve diğerlerinin (2004) 1973 ile 2002 yılları arasında gerçekleştirilen 109 bilimsel çalışmayı inceledikleri meta-analiz çalışmasının sonuçları, okudukları bölümlerden bağımsız olarak öz yeterlilik algısı yüksek olan öğrencilerin, öz yeterlilik algısı düşük olan öğrencilere göre zorluklar karşısındaki dirençlerinin daha yüksek olduğunu ifade etmektedir. İlave olarak bu tür öğrenciler öğrenmede daha iyi performans göstermekte ve derslerine daha yüksek ilgi gösterme eğilimindedirler. Bu durum, başarılı olmak için önce başarıya inanmanın gerekliliğini ortaya koymaktadır.

Ketelhut'un (2007) öz yeterlilik algısı ile ilgili yapılan önceki araştırmaları analiz ettiği çalışmanın sonuçlarına göre, yüksek öz yeterliliğe sahip öğrencilerin zorlukları bir meydan okuma olarak algıladıkları ve zor konuları öğrenmeye daha istekli olduklarını ifade etmektedir. İlave olarak bu tür öğrenciler başarısızlığı daha fazla çabaya ihtiyaç olduğunun bir göstergesi olarak kabul etmektedir ve öğrenmelerini artırmak için özel stratejiler kullanmaktadır. Buna zıt olarak öz yeterlilik algısı düşük olan öğrenciler ise başarısızlığı şanssızlıklarının ya da yeteneklerinin kısıtlı olmasının bir sonucu olarak görmektedir. Bu tür öğrenciler genellikle sorunlarının zorluk seviyesini tam olarak kestiremeyip sorunlarını gözlerinde büyütmektedir.

Benzer sonuçlar programlama öğretimi için de geçerlidir. Bu alanda yapılan çalışmalar, programlama derslerinde öz yeterlilik algısı yüksek olan öğrencilerin daha düşük olan öğrencilere göre, karşılaştıkları programlama problemleri çözmek için daha istekli ve dirençli olduklarını ortaya koymaktadır (Cigdem ve Yildirim, 2014; Ramalingam, LaBelle ve Wiedenbeck, 2004). Lin, Chang ve Tsai'nın (2016) çalışmalarının sonuçları,

programlama öz yeterlilik algısı yüksek olan öğrencilerin düşük olanlara göre temel programlama kavramlarını daha iyi anladıklarını göstermektedir. Başka bir çalışmaya göre bilişim alanında öğrenim gören öğrencilerin öz yeterlilik algılarının hem bilişim sistemleri ile ilgili bir konuyu öğrenme karşı isteklerini hem de ileride bu alanda alanında çalışıp çalışmamak istemelerini doğrudan etkilemektedir (Abdunabi ve diğerleri, 2019).

Wiedenbeck, Labelle ve Kain (2004) öğrencilerin sahip oldukları beceriler ile kendilerine olan inançlarının fazlasıyla iç içe geçmiş olmasından dolayı programlama derslerinde öğrencilerin performanslarını yükseltmenin bir yolunun da öğrencilerin öz yeterlilik algılarını geliştirmek olduğunu savunmaktadır. Özellikle programlama gibi fazla çaba gerektiren ve öğrenilmesi zor konularda öğrencilere uygulamalı deneyimler sağlamak, öğrencilerin öz yeterlilik algılarını geliştirmek açısından oldukça önemlidir (Wiedenbeck ve diğerleri, 2004). Bunun yanında gerçekleştirilen çalışmalar, programlama derslerine yönelik motivasyon ile öz yeterlilik arasında anlamlı bir ilişkinin varlığını göstermektedir (Tsai, 2019; Yukselturk ve Altiok, 2017).

2.2.3. Programlamaya Yönelik Motivasyon

Motivasyon; biyolojik, bilişsel ve sosyal düzenleme süreçlerinin merkezinde yer alması nedeniyle bireylerin davranışlarının incelenmesinde önemli bir konu olmuştur (Serrano-Cámara, Paredes-Velasco, Alcover ve Velazquez-Iturbide, 2014). Eğitim açısından değerlendirildiğine, öğrencileri aktif bir öğrenen olarak tanımlayan yapılandırmacılık kuramı bireylerin bilgilerini özümseyebilmeleri için motivasyonun varlığını en temel koşul olarak belirtmektedir (Petrovskaya, 2019). Motivasyon kavramı, bireyin enerjisini belirli bir amaca yönlendirmek ve bir performansı gerçekleştirmek için harekete geçirmek olarak tanımlanmaktadır (Ryan ve Deci, 2000). Başka bir ifade ile motivasyon, bir amaç için bireyi harekete geçiren güç olarak açıklanmaktadır (Covington ve Dray, 2002).

Alanyazında motivasyon türleri çeşitli araştırmacılar tarafından farklı şekillerde ele alınmıştır. En bilinen sınıflandırmasıyla motivasyon, yönelimiyle itibariyle içsel (intrinsic) ve dışsal (extrinsic) olmak üzere ikiye ayrılmaktadır. İçsel motivasyon bir şeyi doğası gereği ilginç ve zevkli olduğu için yapmayı istemek olarak açıklanırken, dışsal motivasyon ödül, ceza vb. çevresel etmenler nedeniyle bir amacı gerçekleştirmeye yönelmek olarak tanımlanmaktadır (Ryan ve Deci, 2000).

Benzer şekilde Entwistle (2014) motivasyonu üç ana başlık altında incelemektedir:

- *İçsel (intrinsic) motivasyon*: Öğrencilerin ders konularına olan ilgisinden kaynaklanan motivasyon,
- *Dışsal (extrinsic*) *motivasyon*: Öğrencilerin ileride bir ödülü (genellikle finansal) elde etmek için dersi geçme arzusundan kaynaklanan motivasyon,
- *Başarı-Rekabet (achievement – competitive) motivasyonu:* Akademik olarak iyi olmaya ve (bazen) akranlardan daha iyi olmaya yönelik motivasyon.

Sahip olduğu motivasyon türünden bağımsız olarak motive olmuş bir öğrencinin, amaçlarını gerçekleştirmek için daha fazla çaba ve zaman sarf edeceği bilinen bir gerçektir (Chen ve Jang, 2010, Simpson, 2013). Ancak yine de Feldgen ve Clúa'ya (2004) göre motivasyon soyut bir kavramdır ve her ne kadar öğrencilerin davranışlarından yola çıkarak motive olup olmadıkları anlaşılabilse de kesin olarak ölçülmesi mümkün değildir.

Programlama öğretimi değerlendirildiğinde, araştırmacıların motivasyonla ilgili farklı türlerde çalışmalar yaptıkları görülmektedir. Örneğin bazı araştırmacılar, programlama öğretiminde motivasyonun önemi üzerinde durmuştur (Alaoutinen ve Smolander, 2010; Hauswirth ve Adamoli, 2013). Bazı araştırmacılar, öğrencilerin programlama dersini seçme veya bırakma nedenlerine odaklanmıştır. Örnek vermek gerekirse Bergin ve Reilly'nin (2005) yaptıkları çalışmanın sonuçlarına göre öğrencilerin %22'sinin programlama derslerini kişisel ilgilerinden dolayı, %40'ının kariyer nedenleriyle ve %35'inin zorunlu olduğu için seçtiklerini göstermiştir. Bunun yanında motivasyon eksikliği, öğrencilerin programlama dersini bırakmalarının başlıca nedenlerinden biri olarak görülmektedir (Kinnunen ve Malmi, 2006).

Bazı araştırmacılar öğrencileri daha çok motive edebilecek teknolojilere ve öğrenme aktivitelerine yoğunlaşmışlardır. Programlama giriş dersi için geleneksel programlama örnekleri yerine web ve oyun programlama örneklerinin kullanılması daha motive edici olduğunu ifade eden çalışmalar mevcuttur (Çelik, 2020; Kurkovsky, 2013). Hangi açıdan yaklaşılırsa yaklaşılsın programlama öğretiminde motivasyon kritik bir role sahiptir ve programlama derslerinin öğrencilerin motivasyonlarını artıracak şekilde tasarlanması gerekmektedir.

2.2.4. Programlama Kaygısı

Spielberger (2013) kaygıyı sinir sisteminin uyarılmasıyla ilişkili öznel bir gerginlik, korku, sinirlilik ve endişe duygusu olarak tanımlamaktadır. Kaygı bazı bireyler için

genelleştirilmiş kişilik özelliği olmakla birlikte bazı bireyler için duruma bağlı olarak ortaya çıkan bedensel, duygusal ve zihinsel uyarılmış haline verilen isimdir (Malim, 2017). Korku ile kaygı birbirine karıştırılabilen ancak farklı kavramlardır; korku bilişsel iken kaygı duygusal bir süreci ifade etmektedir (Beck, Emery ve Greenberg, 2005). Alanyazında kaygı, durumluk kaygı (state anxiety) ve sürekli kaygı (trait anxiety) olarak ikiye ayrılmaktadır. Durumluk kaygı, bireyin endişeyi belirli bir zaman diliminde veya o anda hissetmesini ifade eden geçici bir duygusal durum olarak nitelendirilirken; sürekli kaygı nispeten istikrarlı bireysel farklılıkları ifade eden kalıcı kişilik özelliğidir (Vitasari, Wahab, Othman, Herawan ve Sinnadurai, 2010).

Programlama kaygısı Connolly, Murphy ve Moore (2007) tarafından olumsuz deneyimler veya beklentilerden kaynaklanan programlama ortamına özgü psikolojik bir durum olarak tanımlamaktadır. Connolly ve diğerleri (2007) ayrıca programlama kaygısının, programlamayı öğrenirken öğrencilerin yeteneklerini yanlış değerlendirmelerinden kaynaklandığını iddia etmektedir. Scott'a (2015) göre, programlama dersleri öğrenciler için oldukça yeni kavram ve materyalleri içerdiğinden, öğrenciler genellikle programlama derslerinin ilk aşamalarında programlama kaygısı ile karşılaşmaktadırlar. Choo ve Cheung'a (1991) göre programlama kaygısı çeşitli unsura karşı kaygıyı barındırmaktadır. Bunlar; (1) programlama derslerinin içeriği ve problem çözme aktivitelerine yönelik kaygı, (2) programı test etme, hata mesajlarını anlama ya da hata giderme gibi programlamaya özgü entelektüel aktivitelere yönelik kaygı, (3) programlama sınavları ya da ödevleri aracılığıyla değerlendirilmeye yönelik kaygıdır.

Programlama derslerinde algılanan kaygı öğrencilerin programlama derslerinde aktif olmalarını ve zihinsel şemaları oluşturmalarını engellemekte olup programlama problemlerine çözüm üretmelerini zorlaştırmaktadır (Demir, 2021). Bunun neticesinde programlama kaygısı Huggard (2004) tarafından "programlama travması" olarak adlandırılan ve kafa karışıklığı, hayal kırıklığı ve can sıkıntısı (Bosch, D'Mello ve Mills, 2013) gibi yoğun olumsuz duygular uyandıran bir fenomene dönüşmektedir.

Yapılan çalışmalar, öğrencilerin derslerde yaşadıkları kaygı düzeyleri ile akademik performans arasında güçlü bir ilişkinin varlığını göstermektedir. Öğrencilerin kaygı düzeyleri arttıkça performansları düşmektedir (Vitasari ve diğerleri, 2010). Bilgisayar bilimlerine giriş dersinin başarısını etkileyen faktörler üzerine yapılan bir boylamsal çalışma, öğrencilerin bu dersteki performansının en önemli yordayıcısının, öğrencinin ders hakkında hissettiği kaygı derecesi ile belirlenen konfor seviyesi olduğunu göstermektedir

(Wilson ve Shrock, 2001). Benzer bir durum programlama dersleri için de geçerlidir. Maguire, Maguire ve Kelly (2017) programlama kaygısının öz güven eksikliğine neden olduğunu ve öğrencileri bağımsız olarak kodlama yapmaktan alıkoyduğunu belirtmektedir. Özmen ve Altun'un (2014) gerçekleştirdiği araştırmanın sonuçları, düşük düzeyde programlama kaygısı olan öğrencilerin programlamaya fazladan zaman ayırıp daha nitelikli programlar kodlarken, kaygı düzeyi yüksek öğrencilerin programlama uygulamalarına sınırlı zaman ayırdıklarını ve programlama öğrenmekten kaçındıklarını belirtmektedir. Benzer şekilde Scott'ın (2015) çalışmasının sonuçları, programlama kaygısının öğrencilerin programlamaya ayırdıkları süreyi azalttığını ve derse katılımlarını düşürdüğü göstermektedir.

2.2.5. Diğer Faktörler

Alanyazında, programlama deneyimi, programlama öz yeterlilik algısı, programlamaya yönelik motivasyon ve programlama kaygısına ek olarak programlama öğretiminde öğrencilerin başarılarını etkileyen farklı faktörlerden de bahsedilmektedir. Örneğin Bergin ve Reilly (2005) cinsiyetin programlama derslerindeki başarıyı etkileyip etkilemediğini araştırmıştır. Bu çalışmanın sonucunda cinsiyetin programlama becerisini etkilediğini bulunmuştur. Buna zıt olarak Doyle, Kasturiratna, Richardson ve Soled (2009) cinsiyetin programlama dersinin başarısı üzerinde bir etkisinin olmadığını vurgulamaktadır. Sharma ve Shen'in (2018) bir Hint ve Avustralya üniversitesini karşılaştırdıkları çalışmanın sonuçlarına göre programlama performansı ile cinsiyet arasında Avustralyalı üniversite öğrencileri için ilişki bulunmazken, Hint üniversitesindeki erkek ve kız öğrencilerin performansları arasında istatistiksel olarak anlamlı bir fark ortaya çıkmıştır.

Öğrencilerin öğrenme stilleri ile programlama başarısını arasında anlamlı bir ilişki hem Byrne ve Lyons (2001) hem de Thomas, Woodbury ve Jarman (2002) tarafından bulunmuştur. Benzer şekilde öğrencilerin programlama dersinden beklentilerinin ve programlama dersine karşı tutumlarının programlama başarılılarını etkilediği ortaya konulmuştur (Baser, 2013; Tüysüz, 2010). Bennedsen ve Caspersen (2007), gerçekleştirdiği çalışmada programlamaya giriş dersi için öğrencilerin başarısını etkileyen faktörleri araştırmış ve lise eğitimindeki matematik başarısının önemli bir gösterge olduğunu bulmuştur. Benzer şekilde Bergin ve Reilly'nin 2005 yılında gerçekleştirdikleri çalışmada öğrencilerin lise eğitimlerinde aldıkları matematik ve fen dersi puanlarının programlama derslerindeki performanslarıyla güçlü bir ilişkisi olduğunu bulunmuştur. Porter, Guzdial,

McDowell ve Simon (2013) ile Wilson ve Shrock'un (2001) çalışmalarının sonuçları bu ilişkiyi doğrular niteliktedir. Alanyazında yukarıdaki bulgularla uyumsuzluk içeren farklı çalışmalar da mevcuttur. Örneğin Ventura (2005) öğrencilerin programlama derslerindeki başarısı ile programlama deneyimi, matematiksel yetenek, akademik ve psikolojik değişkenler, cinsiyet, derslerdeki rahatlık ve öğrenci çabası değişkenler arasındaki ilişkiyi incelemiştir. Bu çalışmanın sonucuna göre programlama deneyimi, matematiksel yetenek ve cinsiyet öğrencilerin programlama başarılarının yordayıcısı olarak bulunmazken öğrenci çabası ve rahatlık düzeyinin başarının en güçlü yordayıcıları olduğu bulunmuştur.

Alanyazında yer alan birçok çalışma öğrencilerin programlama derslerindeki başarılarını etkileyen faktörleri araştırmış olsa da, bulgular farklı faktörler ve hatta bazı durumlarda aynı faktörler için farklı sonuçlar göstermektedir (Ayalew ve diğerleri, 2018). Ayalew ve diğerlerine (2018) göre alanyazındaki çalışmaların sonuçlarının değişiklik göstermesinin nedeni çevre, kültür, öğretim yöntemi, dersin yapısı, değerlendirme stratejisi, öğretim elemanı, öğrencilerin içsel yapısı gibi birçok değişkenden kaynaklanmaktadır.

2.3. Programlama Öğretimini Kolaylaştırma Stratejileri

Programlama öğretimi alanyazınında öğrencilerin başarılarını yükseltmek ve derse karşı motivasyonlarını artırmak için pek çok çalışmanın yapıldığı görülmektedir. Bu çalışmaların temel eleştirisi geleneksel programlama öğretiminin etkin ve verimli olmamasına yöneliktir. Kısaca bahsetmek gerekirse geleneksel programlama öğretimi, tipik bir "Java veya C# Programlamaya Giriş" dersinin yapısındadır ve alanyazında *"Dersler ve laboratuvarlar (Lectures and labs)"* ismiyle sınıflandırılabilmektedir. Bu yaklaşımda öğrenciler Eclipse ya da Visual Studio gibi bir IDE kullanarak kodlarını yazmakta olup, yazdıklarını kodların sonuçlarını konsolda ya da form üzerinde görmektedirler (Forte ve Guzdial, 2004). Santos ve diğerlerine (2020) göre geleneksel programlama öğretimi yöntemi davranışsal bir öğrenme teorisini benimsemektedir. Bu yöntemde öğrenciler, öğretmenin mizahı, coşkusu ve kendilerinin edindikleri bilgi, beceri ve anlayışla motive edilebilirler. Günümüzde en temel ve yaygın programlama öğretim yöntemi geleneksel programlama öğretimidir (Foster, 2015; Santos ve diğerleri, 2020).

Son zamanlarda, geleneksel programlama öğretim yöntemlerine yönelik bazı eleştiriler yapılmaktadır. Eleştirilerin ana nedeni bu yöntemin öğrencilerin bilgilerini uygulamalarına izin vermemesi ve öğrencilerin edindikleri programlama bilgilerinin kalıcı olmaması

yönündedir (Abdellatif, 2020). Alanyazında geleneksel programlama öğretimine alternatif olabilecek çok çeşitli yöntemin denendiği ve öğrencilerin geleneksel programlama öğretiminde yaşadıkları güçlükleri ortadan kaldırmak için pek çok çalışmanın yapıldığı görülmektedir. Örneğin Galves, Guzman ve Conejo (2009) yaptıkları çalışmada öğrencilerin nesne yönelimli programlama problemlerini çözebilecekleri ve anında geri bildirim alabilecekleri bir problem çözme ortamı geliştirmişlerdir.

Bazı çalışmalar öğrencilerin devam durumlarına ve ders faaliyetlerinden aldıkları notlara odaklanmıştır. Green, Chan ve Plant (2016) programlama dersinden başarısız olma ihtimali yüksek öğrencileri tespit ve takip etmek için bir sistem geliştirmişlerdir. Porter ve diğerleri (2013) öğrencilerin programlama derslerine katılımlarını sağlamak amacıyla *eşli programlama (pair programming)* yöntemini denemiştir. Alanyazında öğrencilere etkili geri bildirim sunmak için öğrencilerin yazdıkları kodları otomatik olarak değerlendiren sistemlerin de kullanıldığı görülmektedir (Char, 2016; Shaffer ve Rosson, 2013). İlave olarak programlama öğretimini daha etkili ve verimli yapmak için etkileşimli kitapların ve öğrenme nesnelerinin kullanıldığı görülmektedir (Vincenti, Braman ve Hilberg, 2013).

Robotlar ve robotik kitler de programlama derslerinin etkinliğini artırmak maksadıyla kullanılmaktadır (Bers, Flannery, Kazakoff ve Sullivan, 2014). Micro:bits (Şekil 2.1), Lego Mindstorms (Şekil 2.2) ve Bee-bots (Şekil 2.3) bu araçlara örnek gösterilebilir. Bu araçlar, somut bir nesne üzerinde çalışmanın öğrenmeyi daha az soyut hale getireceği düşünülerek genellikle görsel programlama dilleriyle birlikte derse entegre edilmektedir (Rose, 2019). Bu yöntemde, öğrencilerin yazdıkları kodların sonuçları fiziksel ve somut bir sistem olan bir robota yansıtılmaktadır (Sullivan, 2008). Barnes'a göre (2002) sadece bilgisayar ekranlarına odaklanmaya kıyasla gözlemlenebilir fiziksel davranışlar (ışık, ses, hareket vb.) gösteren ve anlamlı aktiviteler gerçekleştiren fiziksel bir model, öğrencilerin daha çok ilgisini çekmektedir.

Şekil 2.1. Micro:bits (https://blog.adacore.com/)

Şekil 2.2. Lego Mindstroms (https://www.lego.com)

Şekil 2.3. Bee-bot (Rose, 2019)

Programlama öğretimini geliştirmeyi amaçlayan bir başka yöntem ise programlama derslerinde *bağlantısız aktiviteler (unplugged activities)* olarak isimlendirilen materyallerin kullanılmasıdır. Bu yöntemde, öğrencilere programlama bilgisinin bilgisayar kullanmadan mantık oyunları, kartlar veya fiziksel hareketler yoluyla kazandırılması amaçlanmaktadır (Bell, Alexander, Freeman ve Grimley, 2009). Yapılan araştırmalar, programlama öğretimine bu yöntemle başlamanın bilgisayar tabanlı uygulamalar veya dillerle başlamaya kıyasla özellikle ilköğretim çağındaki öğrencilerin güvenini artırdığına dair sonuçlar sunmaktadır (Hermans ve Aivaloglou, 2017).

Alanyazındaki bahsedilen tüm girişimlerin yanında, programlama öğretimini daha etkili ve verimli yapmak için en çok kullanılan stratejilerden birisi de *ciddi oyunların (serious games)* programlama müfredatına entegre edildiği yöntemdir. Ciddi oyunlarla ilgili yapılan araştırmalar, oyunları ders içerikleri ve öğrenme etkinlikleriyle bütünleştirmenin öğrencilerin akademik başarıları, motivasyonları ve sınıf dinamikleri üzerinde olumlu etkilerinin olduğunu göstermektedir (Rosas ve diğerleri, 2003). Bunun yanında dijital oyunlarının çekiciliği, sürükleyiciliği ve etkileşimli özellikleri sayesinde bu tür öğrenme ortamlarında öğrencilerin soyut düşünme ve problem çözme becerilerinin geliştirdiği, aktif öğrenme süreçlerinin hızlandığı, alternatif öğrenme stillerinin desteklendiği ve ders memnuniyetlerinin arttığı bilinmektedir (Abbasi ve diğerleri, 2021; Peña Miguel, Corral Lage ve Mata Galindez, 2020).

İlgilerini çeken etkinliklerin öğrenciler tarafından iyi karşılandığı; bu tür etkinliklerin öğrencilerin yaratıcılıklarına, üretkenliklerine ve işbirliği yapma becerilerine katkıda bulunduğu bilinen bir gerçektir (Khaleel ve Meriam, 2015; Spieler ve Slany, 2018). Bununla birlikte iyi tasarlanmış dijital oyun içeren etkinlikler, öğrencilere pratik becerilerin yanı sıra kavramsal anlayış da kazandırmaktadır (Singer ve Schneider, 2012). Oyunların öğrenme üzerindeki tüm olumlu yansımaları düşünüldüğünde Kerr'e (2006) göre oyunları ders

müfredatına katmak çok doğal bir tepki olarak karşılanmalıdır. Kucher'e (2021) göre günümüzde birçok akademisyen, oyun ve benzeri araçları ders müfredatlarına uyarlayarak öğrencilerin motivasyonlarını yükseltmeye ve ders kazanımlarını artırmaya çalışmaktadır.

Alanyazında oyunların programlama öğretimi müfredatına; *(1) geleneksel alıştırmalar yerine oyunları kullanma, (2) derslerde oyun oynayarak öğrenme ve (3) dijital oyun geliştirme veya mevcut bir oyuna eklentiler yapma* olmak üzere üç şekilde entegre edildiği söylenebilir (Sung, Hillyard, Angotti, Panitz, Goldstein ve Nordlinger, 2010; Wang ve Wu, 2009). Bu yöntemler aşağıda açıklanmıştır.

- *Geleneksel alıştırmalar yerine oyunları kullanma*: Bu yöntemde geleneksel ders etkinlikleri yerine ders aktiviteleri olarak oyunlar kullanılmaktadır. Bu sayede öğrencilerin ders etkinliklerini ve alıştırmalarını yaparken ekstra çaba sarf etmeleri sağlanmaktadır. Bunun yanında bu yöntem öğretim elemanlarına öğrencilerin alıştırmalarla gerçek zamanlı olarak nasıl çalıştıklarını izleme fırsatı vermektedir (Wang ve Wu, 2009).

- *Derslerde oyun oynayarak öğrenme:* Bu yöntemde, öğrencilerin katılımını ve motivasyonlarını artırmak için derslerde dijital oyunların oynanmaktadır. Bu yaklaşımda öğrenciler ve öğretim elemanları derslerde bilgi temelli eğitsel oyunlar oynamakta ve bu etkinliğin sonuçlarını tartışmaktadır.

- *Dijital oyun geliştirme veya mevcut bir oyuna eklentiler yapma*: Bu yöntemde öğrenciler, bilgisayar bilimlerine ait bilgi ve becerileri elde etmek için dijital oyun geliştirir veya mevcut bir oyuna eklentiler yaparak oyunları kendilerine göre değiştirirler. Alanyazında oyun geliştirmenin İngilizcesi *"game programming"*, *"game development"*, *"game design"* olarak tanımlanmakla birlikle mevcut bir oyuna eklentiler yaparak oyunu değiştirmek *"game modding"* olarak tanımlanmaktadır. Kullanılan kavramdan bağımsız olarak akademik amaçlarla bir oyunun geliştirildiği veya değiştirildiği öğrenme stratejisine "*Oyun Geliştirmeye Dayalı Öğrenme-Game Development Based Learning*" denilmektedir.

2.4. Oyun Geliştirmeye Dayalı Öğrenme

Oyun geliştirmeye dayalı öğrenme stratejisinde, öğrenciler yazılım mühendisliği yöntem ve tekniklerini oyun geliştirme projeleri vasıtasıyla deneyimler. Bu stratejinin kullanılmasındaki temel amaç, öğrencilerin programlama ders içeriğine ve öğrenme etkinliklerine yönelik motivasyonlarını artırmak, programlama dersine yönelik tutumlarını iyileştirmek, derse katılımlarını ve işbirlikçi çalışmalarını teşvik etmek, teori ile uygulama

arasındaki süreyi kısaltmak ve soyut kavramlarla somut etkinlikleri bir araya getirmektir (Çelik, 2020; Jafari ve Abdollahzade, 2019). Başka bir ifade ile öğrencilerin oyunlara karşı olan ilgilerini oyun geliştirmeye veya mevcut bir oyuna eklentiler yapmaya yönlendirerek programlamayı öğrenmelerini sağlamaktır.

Oyunların programlama öğretimi amacı doğrultusunda kullanılmasının Souza, Veado, Moreira, Figueiredo ve Costa'ya göre (2017) göre üç nedeni vardır. Bunlardan ilki, oyunların öğrenciler için iyi bilinen bir alan olmasıdır. Bu nedenle geliştirilecek bir oyun projesinin gereksinimleri öğrenciler tarafından kolay anlaşılabilmektedir. İkincisi, oyunların ilgi çekici, motive edici ve bilişsel kapılmayı kolaylaştırıcı unsurlar barındırmasıdır. Son olarak, oyunların barındırdığı çeşitli karmaşık seviyeleri, öğreticilere sınıf ödev ve projeleri için geniş bir seçenekler seti sağlamaktadır. Bu yöntemin bir başka yenilikçi uygulaması, oyun geliştirmek veya değiştirmek için çeşitli oyun geliştirme araçlarının kullanımını da öğretiyor olmasıdır. Bu sayede öğrenciler hem yeni bir araç kullanma becerisi kazanırken aynı zamanda her şey için sıfırdan kod yazmak yerine yazılım mühendisliği kavram ve uygulamalarına yoğunlaşabilmektedir (Souza ve diğerleri, 2017).

Yapılan çalışmalar, oyun geliştirmeye dayalı öğrenme ortamlarının dersler için ilgi çekici, hedefe yönelik ve etkileşimli görevler sağladığını; bunun da dersleri daha dinamik, işbirlikçi ve çekici hale getirdiğini ifade etmektedir (Khaleel ve diğerleri, 2015; Spieler, Grandl, Ebner ve Slany, 2019). Bazı çalışmalar eğlenceli olarak nitelendirilen oyun programlama etkinliklerinin öğrencilerin yaratıcılık, problem çözme, mantıksal düşünme, sistem tasarımı ve işbirliği becerilerinin gelişimine olumlu etkilerinin olduğunu göstermektedir (Backlund ve Hendrix, 2013). Bunun yanında Wu ve Wang'e (2012) göre oyun geliştirmeye dayalı öğrenme, öğrenciler için kişisel olarak anlamlı olan yazılımları kullanma deneyimi sağlayarak bir ürün oluşturmalarına ve yeni bilgileri yapılandırmalarına yardımcı olmaktadır. Bu yöntem sayesinde öğrenciler nesne yönelimli tasarım, tasarım kalıpları, yazılım mimarisi, kullanılabilirlik gibi yazılım mühendisliğinin en iyi uygulamalarını uygulayarak, disiplinin doğasında bulunan sorunları ve zorlukları kendi başlarına keşfetme ve çözme şansı elde etmektedir (Souza ve diğerleri, 2017).

Oyun geliştirmeye dayalı öğrenme stratejisinin bilgisayar bilimlerinde kullanıldığı belirli konular veya alanlar değerlendirildiğinde, Wang ve Wu'nun (2015) gerçekleştirdiği çalışma değerlidir. Wang ve Wu (2015) oyun geliştirmenin büyük oranda programlamayı öğretmek için kullanıldığını ifade etmektedir. Oyun geliştirmenin kullanıldığı ikinci en büyük konu ise yapay zekâyı öğretmektir. İlave olarak bilgisayar bilimlerinde oyun

geliştirmenin kullanıldığı diğer konular ise algoritma tasarımı, paralel hesaplama, veri yapıları ve boole cebiridir. Bununla birlikte, alanyazında oyun geliştirmeye dayalı öğrenmenin derslerde nasıl kullanıldığı konusunda büyük bir çeşitlilik mevcuttur (Wang ve Wu, 2015). Bazı çalışmalarda bu strateji sadece öğrencilerin ödevleri için kullanılmış olsa da bazı çalışmalarda tüm dersin bir oyun geliştirme projesi etrafında tasarlandığı görülmektedir. Yapılan çalışmalar dijital oyunlarının görsel doğasının, nesne yönelimli programlamanın öğretme-öğrenme sürecini geliştirmeye katkıda bulunduğunu ve öğrencilerin sıklıkla öğrenmekte zorlandıkları soyut kavramları anlamalarına olanak tanıdığını göstermektedir (López, Duarte, Gutiérrez ve Valderrama, 2019; López, Duarte ve Gutiérrez, 2020).

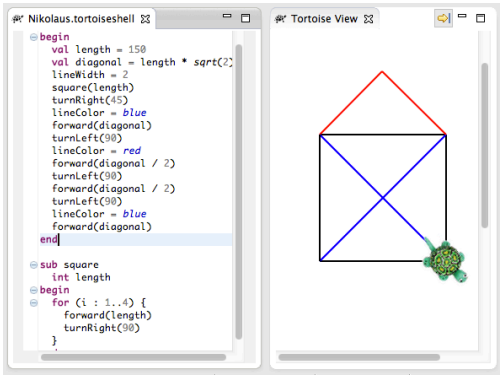
2.4.1. Tarihçesi ve Sınıflandırması

Programlama öğretiminde oyun geliştirmeye dayalı öğrenmenin kullanılması fikri yeni bir uygulama değildir (Vassilev, 2015). Oyun benzeri bir ortamda programlama yoluyla öğrenmenin en eski uygulamaları 1970'lerin başlarına dayanmaktadır. Alanyazındaki ilk örnekleri *Logo* (Lukas, 1972) ve *Karel the Robot*'tur (Pattis, Roberts ve Stehlik, 1995). *Logo*, metin tabanlı basit komutlarını kullanarak farklı geometrik desenler çizen programlanabilir bir kaplumbağa konsepti üzerine kuruludur (Şekil 2.4). Logo kullanılarak öğrencilerin prosedürel işlemler, yineleme ve özyineleme gibi genel hesaplama konularında becerilerinin geliştirilmesi amaçlanmaktadır.

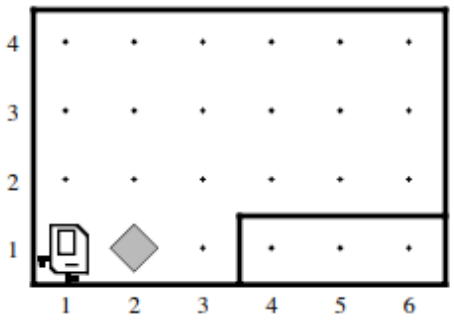
Logo ile benzer amaçlarla geliştirilen *Karel the Robot* 1981 yılında yayımlanmıştır (Papert, 1980). İki boyutlu bir ızgara üzerinde hareket edebilen bir robotu kontrol etmek için Pascal tabanlı basit bir programlama dilini kullanır. Bu robot hareket, dönüş ve sesli uyarı gibi talimatları anlayıp tepki verebilmektedir (Şekil 2.5). Günümüzde halen kullanılmakta olan birçok farklı sürüme ilham vermiştir. Karel the Robot kullanılarak öğrenciler tarafından oluşturulan nesneler (robotlar, bipleyiciler, duvarlar vb.) kodlanarak bu nesnelere ait programlanmış davranışlar kolayca gözlemlenebilmektedir (Henriksen ve Kölling, 2004).

Logo ve Karel the Robot'un kodlamayı daha erişilebilir hale getirmeyi amaçlayan programlama araçları oluşturma geleneği başlatmıştır ve bu araçlar için temel teşkil etmiştir (Foster, 2015; Rose, 2019). Günümüzde programlama öğretimini kolaylaştırmak için çok sayıda aracın geliştirildiği görülmektedir. Alanyazında programlama öğretiminde kullanılan oyun geliştirmeye dayalı öğrenme stratejisi iki farklı şekilde kullanılmaktadır. Bunlar; (1)

Oyuna benzer geliştirme ortamlarının kullanımı ve (2) oyun geliştirme veya mevcut bir oyuna eklentiler yapmadır (Foster, 2015; Souza ve diğerleri, 2017).



Şekil 2.4. Logo (eclipse.org, 2021)



Şekil 2.5. Karel the Robot (Roberts, 2005)

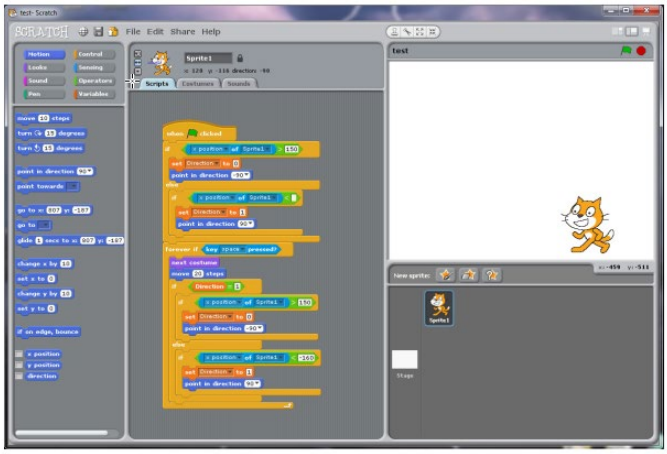
2.4.1.1. Oyuna benzer geliştirme ortamlarının kullanımı

Bu yaklaşımında öğrenciler, programlarını oyuna benzer geliştirme ortamlarını kullanarak oluşturmaktadırlar (Foster, 2015). Bunlara *Alice, Scratch, Kodu ve Greenfoot* gibi geliştirme ortamları örnek gösterilebilir. Öğrenciler yazılım geliştirmek için kod yazmak yerine blok tabanlı işlemler gerçekleştirmektedir. Bu araçlar aşağıda kısaca açıklanmıştır.

Alice, Carnegie Mellon Üniversitesindeki bir ekip tarafından geliştirilmiştir. Java programlama dili ile oluşturulmuş bir blok tabanlı programlama ortamıdır (Abdellatif, 2020). Arayüzü, program bileşenlerinin sürükle ve bırak manipülasyonuna dayanmaktadır (Şekil 2.6). Kullanıcıların kolayca animasyonlar oluşturmasına ve animasyonları kontrol etmesine olanak tanıyan Alice, öğrencilerin mantıksal ve hesaplamalı düşünme becerilerini geliştirmek, bunun yanında nesne yönelimli programlamanın temel ilkelerini öğretmek için tasarlanmıştır (Hainey, Connolly, Stansfield ve Boyle, 2011).

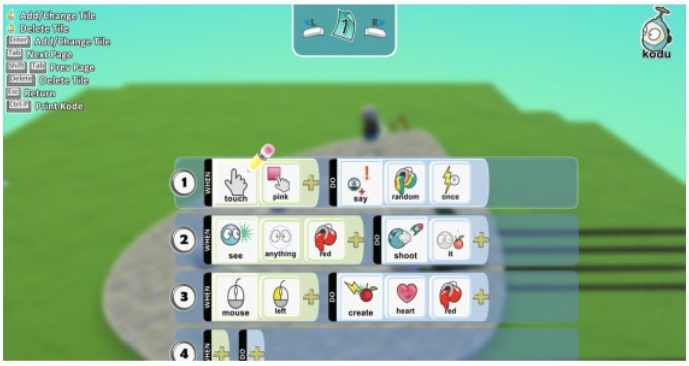
Şekil 2.6. Alice programı ekran görüntüsü (Abdellatif, 2020)

Scratch, Massachusetts Institute of Technology'deki medya laboratuvarı tarafından geliştirilen bir multimedya ortamıdır (Şekil 2.7). Bu aracın yaratıcıları çocukların yapı bloklarıyla nasıl oynadığına odaklanarak Scratch'i yapı bloklarına benzeyen bileşenler içerecek şekilde geliştirmiştir (Resnick ve diğerleri, 2009). Scratch çeşitli renkleri kullanarak farklı eylemleri kategorize etmekte ve bu eylemlere uyması için bulmaca benzeri bir şema kullanmaktadır (Stenerson, 2012). Scratch, öğrencilerin değişkenler, koşullar, döngüler ve nesneler gibi programlama kavramlarını öğrenmeleri için blokları sürükleyip bırakmalarına olanak tanımaktadır. Scratch de Alice gibi öğrencilerin kendi oyun ve animasyonlarını oluşturmalarını sağlamaktadır.



Şekil 2.7. Scratch programı ekran görüntüsü (Stenerson, 2012)

Kodu, Microsoft firması tarafından geliştirilen ve dokuz yaş ve üzeri çocuklar için tasarlanmış yüksek seviyeli, görsel ve yorumlanmış bir programlama dilidir (Rose, 2019; Stolee, 2010). Alanyazında Kodu Game Lab olarak da isimlendirilmektedir. Kodu programlama dili "*bağımsız keşif*" yoluyla dijital oyunlar oluşturmak için bir üç boyutlu oyun ortamına entegre edilmiştir (MacLaurin, 2011). Kodu, Scratch ve Alice'in aksine öğrenilmesi daha kolay bir kullanıcı arayüzüne (Şekil 2.8) ve daha ilgi çekici grafiklere sahiptir (Stenerson, 2012). Kodu olay tabanlı bir programlama dilidir ve kullanıcılar her bir karakteri (ağaç, bulut, elma, kaya vb.) ayrı ayrı programlayarak hem nesnelere davranış kazandırmakta hem de olayların akışını biçimlendirmektedir (Fowler, Fristce ve MacLauren, 2012).



Şekil 2.8. Kodu programı ekran görüntüsü (Rose, 2019)

Greenfoot, nesne yönelimli programlama öğretimine etkileşimli, görsel ve ilgi çekici (Şekil 2.9) bir yaklaşım sağlayan entegre bir yazılım geliştirme ortamıdır (Paraskeva, Mysirlaki ve Papagianni, 2010). Hedef kullanıcı kitlesi 14 yaş ve üzeri öğrenciler olmakla beraber üniversite öğretimi için de uygundur (Kölling, 2010). Öğrenciler Greenfoot'u kullanarak temel programlama kavramlarını öğrenirken, oyunlar ve simülasyonlar gibi ilgi çekici programları hızlı ve kolay bir şekilde geliştirebilmektedir (Fincher, Cooper, Kölling ve Maloney, 2010). Kölling'e (2010) göre Greenfoot, genellikle nesne yönelimli programlama ile ilişkilendirilen karmaşıklığın çoğunu ortadan kaldırmakta ve özel olarak tasarlanmış bir ortam sağlayarak Java'nın kullanımını kolaylaştırmaktadır. Greenfoot'un en önemli özelliklerinden birisi nesne yönelimli programlama kavramlarını görselleştirmesidir.

Şekil 2.9. Greenfoot programı ekran görüntüsü (Kölling, 2010)

2.4.1.2. Oyun geliştirme veya mevcut bir oyuna eklentiler yapma

Oyun geliştirme; programlama derslerinde öğrencilerin derse katılımı ve derse yönelik motivasyonlarını artırmak için basit oyunlar tasarlama ve programlamayı içeren

programlama öğretimi yaklaşımıdır (Ouahbi, Kaddari, Darhmaoui, Elachqar ve Lahmine, 2015). Öğrenciler bu yaklaşımda gerçek hayattaki profesyonel bir yazılım geliştiricisi rolüne benzeyen oyun geliştiricisi rolünü üstlenirler (Teodosieva ve Teodosiev, 2021). Bununla birlikte öğrenciler, çeşitli araçları kullanarak problem tabanlı bir öğrenme ortamında kendi oyunlarını oluşturmak için programlama kavramlarını nasıl uygulayacaklarını öğrenmeye yönlendirilirler (Soares ve diğerleri, 2015). Kafai (1995) yıllar önce gerçekleştirdiği çalışmada oyun geliştirmenin programlama öğretimi açısından önemi şu şekilde açıklamaktadır: "*Öğrenciler tüm tasarım kararlarına dâhil olmakta ve teknolojik akıcılığı geliştirmeye başlamaktadır. Nasıl dilde akıcılık, dille ilgili gerçekleri bilmekten çok daha fazlasını ifade ediyorsa; teknolojik akıcılık da yalnızca yeni teknolojik araçların nasıl kullanılacağını değil, aynı zamanda bu araçlarla önemli şeylerin nasıl oluşturulacağını ve en önemlisi, bu araçların kullanımına dayalı yeni düşünme biçimleri geliştirmeyi de içermektedir.*" (s. 39). Teknolojik imkânların artmasıyla birlikte günümüzde bu yaklaşımın nesne yönelimli programlama öğretimine entegre edilebilmesini sağlayan çeşitli araçlar bulunmaktadır (Chursin ve Semenov, 2020; Robertson, 2012). Bu araçlar alanyazında *oyun motoru* olarak isimlendirilmekte olup Game Maker, JMonkey, Marmalade, Ogre3D, Shive, Sio2, Turbulenz, Unreal Engine 4 ve Unity 3D gibi yazılımlar en modern oyun motorları arasında gösterilmektedir (Chursin ve Semenov, 2020).

Yapılan çalışmalar, programlamayı oyun geliştirerek öğrenmenin hem öğrencilerin nesne yönelimli programlama dersine karşı olan ilgilerini; hem de bu derslerindeki başarılarını artırdığını göstermektedir (Law, 2020; Renton, 2016). Wong, Hayati, Yatim ve Hoe'nun (2017) gerçekleştirdiği çalışmanın sonuçlarına göre oyun geliştirme tabanlı öğrenme yaklaşımının nesne yönelimli programlama için faydalı bir öğrenme yaklaşımı olduğu göstermiştir. Bu çalışma, öğrencilerin tanımlanmış bir oyun geliştirme projesinde nesne yönelimli programlama paradigmasıyla aktif olarak ilgilendiklerini, bu paradigmayı kullandıklarını ve uygun oyun mekanikleri tasarladıklarını göstermektedir. Benzer şekilde Lee, Kurniawan ve Choo'nun (2021) derslerinde oyun geliştirme içeren dönem sonu projesi bazlı değerlendirme sistemini kullandıkları çalışmanın sonuçlarına göre öğrenciler oyun projelerine karşı yüksek ilgi göstermiştir. Bu durum öğrencilerin nesne yönelimli programlama dersi başarıları olumlu yönde etkilemiştir.

Programlama öğretimine oyun programlamayı dâhil etmenin bir diğer olumlu yanı ise öğrencilere sağladığı yüksek seviyede analiz etme ve yaratıcı düşünme becerileri ile ilgilidir. Carbonaro ve diğerleri (2010) programlama dersinde öğrencilerinden ScpritEase aracını

kullanarak oyun geliştirmelerini istemiştir. Çalışmanın sonuçlarına göre oyun programlama aktiviteleri öğrencilerin yüksek seviyede düşünme becerilerini harekete geçirerek daha fazla yaratıcı düşünme sürecine girmelerini sağlamıştır. Bu bağlamda değerlendirildiğinde, oyun programlayarak programlama öğretiminin öğrencileri dijital üretici olmaya teşvik ederek (Kafai, 2006) programlama dersi verimliliğini artırdığı söylenebilmektedir (Garneli ve Chorianopoulos, 2018).

Mevcut bir oyuna eklentiler yapma; bir oyunun belirli öğelerini değiştirerek oyunun biraz veya tamamen farklı bir sürümünü oluşturma sürecini ifade etmektedir (El-Nasr ve Smith, 2006). Bu oyun modifikasyonu işlemi alanyazında genellikle "*oyun modları (game mods)*" veya kısaca "*modlar (mods)*" olarak adlandırılmaktadır. Mevcut bir oyundan yeni bir oyun oluşturma işlemi genellikle orijinal oyundan zevk aldıkları veya geliştirmek istedikleri için harici geliştiriciler veya oyun meraklıları tarafından gerçekleştirilmektedir (Cheung, Zimmermann ve Nagappan, 2014). Bu oyun meraklıları orijinal oyunlara yeni karakterler, düşmanlar, silahlar, modeller, seviyeler, hikâyeler ve müzikler gibi yeni unsurlar ekleyerek oyunu tamamen veya kısmen değiştirebilirler. Bu durum orijinal bir oyunun başlangıçta amaçlandığından daha uzun süre tekrar oynanabilir kalmasına olanak sağladığı gibi geliştiricilerin kişisel fikirlerini oyuna yansıtmalarına olanak tanır (Cheung ve diğerleri, 2014; Grizioti ve Kynigos, 2021). İlgili alanyazın, oyun modları oluşturmanın hemen hemen her oyunu kapsayabilen ve 80'lerden beri gerçekleştirilmekte olan bir işlem olduğunu belirtmektedir (Vagle, Palmason, Dautaj, ve Lysø, 2021). Half Life, Warcraft 3, NeverWinter Nights, Unreal Tournament, The Sims 3, Command and Conquer, Civilization, Dragon Age, Minecraft, Garry's Mod gibi birçok ünlü oyun kendi oyun geliştirme araç setiyle birlikte satın alınabildiği bilinmektedir (Foster, 2015; Loh ve Byun, 2009).

Oyun geliştirmeyi ders müfredatına uyarlamak isteyen araştırmacılar açısından mevcut bir oyuna eklentiler yapma etkinliği değerlidir çünkü hem sıfırdan bir oyun tasarlamak zordur hem de bu işlem çok fazla zaman ve kaynak gerektirmektedir (Bellotti ve diğerleri, 2011; Sousa, 2021). Sıfırdan sofistike bir oyun oluştururken karşılaşılan öğrenme eğrisi ve teknik sorunları ortadan kaldırması açısından oyun modu oluşturma, programlama öğretiminde birçok çalışmada kullanılmıştır (Grizioti ve Kynigos, 2018; Sotamaa, 2010). Yapılan araştırmalar, oyun modu oluşturmanın öğrencilerin değişkenler, olay işleme, koşullu yapılar gibi programlama kavramları ile nesne yönelimli tasarım sürecini anlamalarını kolaylaştırdığını ve programlamayı öğrenmeye motive ettiğini göstermektedir (El-Nasr ve Smith, 2006). Bunun yanında oyun modu oluşturmanın bir yazılımcının sahip

olması gereken beceriler olan eleştirel düşünme, hata giderme, karmaşıklığı yönetme, yineleme ve iyileştirme gibi bir dizi becerinin gelişimini teşvik ettiği ifade edilmektedir (Grizioti ve Kynigos, 2018; Moshirnia, 2007).

El-Nasr ve Smith (2006) mevcut oyunları yenilerini oluşturmak için değiştirmeyi *"öğretim açısından pek çok faydalar barındıran bir tasarım etkinliği*" olarak tanımlamaktadır (s. 2). El-Nasr ve Smith'e (2006) göre bu etkinlikler, öğrencilerin beceri ve kavramları keşfetmeleri; bunun yanında bunları gerçek dünyada nasıl uygulanabileceklerini anlamaları için anlamlı bağlamlar sağlamaktadır. Öğrenciler tasarım sürecinde analiz, sentez, değerlendirme, gözden geçirme gibi becerileri; bunun yanında planlama ve izleme gibi üstbilişsel becerileri kullanmaktadır. Bu süreçte öğrenciler kendi iç dünyalarını ve fikirlerini yansıtan ürünler oluştururken akranlarından ve uzmanlardan sürekli geri bildirim alabilmektedir. Tasarımlarının nasıl ve neden başarısız olduğunu, nasıl daha başarılı olabileceklerini anlamak için çalışabilirler. Son olarak, karşılaştıkları sorunlar için alternatif çözümler üretebilmektedirler.

Benzer şekilde Loh ve Byun (2009) mevcut bir oyuna eklentiler yaparak programlama öğrenmenin üniversite ve lise öğrencileri için uygun bir yöntem olduğunu belirtmektedir. McArthur ve Teather (2015) bu yöntemin bireysel ya da grup olarak çalışmaya olanak tanıdığını; bununla birlikte sınıf içi kısa süreli etkinlikler ya da yarıyıl boyu süren bir proje etkinliği olarak tasarlanabileceğini belirtmektedir. Scacchi'ye (2011) göre bu yöntem oldukça kıymetlidir çünkü öğrencilerin mod oluşturma, oyun geliştiricisi olma ve bilgisayar oyunları bilimini öğrenmesine yardımcı olmaktadır.

2.4.2. Dayandığı Pedagojiler

Oyun geliştirmeye dayalı öğrenme stratejisinin iki farklı pedagojiyi temel aldığı söylenebilir (O'Grady-Jones, 2020). Bunlar *inşacılık (constructionism)* ve *proje tabanlı öğrenmedir (project-based learning)*. *İnşacılık,* Jean Piaget'nin yapılandırmacı öğrenme teorisini temel alır. Yapılandırmacı (constructivist) öğrenme teorisi, bilginin öğrenenler tarafından çoğunlukla sosyal bir bağlamda bireysel olarak oluşturulduğunu savunmaktadır (Vygotsky, 1978). İnşacılık öğrenme teorisi de bilginin bir öğretici tarafından iletilmek yerine öğrenenler tarafından aktif olarak inşa edildiğini kabul etmektedir.

İnşacılıkta öğrenme bireysel bir bilgi alanı oluşturma etkinliğidir (Pellas, 2014). Bu etkinlikte bilginin inşa edilebilmesi için bireyin önce düşünmesi, sonra inşa etmesi ve ardından düşüncelerini başkalarıyla paylaşması gerekir. Öğrenen, akranlarının veya

öğreticilerin geri bildirimleri ve tepkileri yoluyla yeni bir bilgi alanı inşa eder. Bilginin inşası sırasında *metin, öykü, oyun, model, makine, bilgisayar yazılımı ve hatta oyuncaklar* gibi öğrenenler için kişisel olarak anlamlı ve paylaşılabilir eserlerin oluşturulması çok önemlidir çünkü bilgi bu ürünlerin oluşturulması vasıtasıyla kazanılmaktadır (Harel ve Papert, 1991). İnşacılığı merkezine alan bir programlama öğretiminde de öğrenciler Papert'ın (1980) deyimiyle üzerinde düşünecekleri bir nesneye (object-to-think-with) sahip olmakta; eserlerini ve kendi öğrenme süreçlerini bu nesne üzerinde yönetme fırsatı bulmaktadırlar (Özmen, 2020).

Öğrenciler, inşacılık üzerine kurulu öğrenme ortamlarında sınıflarının ötesine geçerek kendilerini gerçek dünyaya bağlayan anlamlı projelerde görev alırlar ve "*yaparak öğrenme"* stratejisini kullanarak ürünler oluştururlar (Togashi, 2019). Bu tür öğrenme ortamlarında teknoloji, içerik sağlama aracı olmaktan ziyade Hay ve Barab'in (2001) tabiriyle "*entelektüel ifade ve keşif için bir araç*" görüntüsündedir. İlave olarak bu öğrenme ortamlarında eğlenceli deneyler teşvik edilir. Sürekli test etme ve tekrarlı denemeler, öğrencilerin tasarımlarını geliştirmeye teşvik edildiği öğrenme sürecinin değerli bir parçasıdır (Resnick, 2014). Oyun geliştirmeye dayalı öğrenme bağlamında öğrenciler, sosyal değerin yanı sıra kişisel değeri de olan oyunlar oluşturmakta ve paylaşmaktadırlar (O'Grady-Jones, 2020).

Brennan'ın (2015) *tasarım, kişiselleştirme, paylaşım* ve *yansıtma* olmak üzere dört basamakta kategorize ettiği inşacı öğrenme süreci, oyun geliştirmeye dayalı öğrenme açısından önemli adımları kapsamaktadır. Nitekim tasarım aşamasında öğrencilerin tarafından gerçekleştirilen problem analizi ve olası çözüm yollarının araştırılması gibi adımlar, oyun projesi üzerinde gerçekleştirilmesi planlanan eklentilerin belirlenmesinde kritik bir öneme sahiptir. Oyuna hangi eklentilerin yapılacağı, bu eklentilerin hangi sınıf ve nesneleri içereceği, hangi oyun mekanik ve dinamiklerin oyuna yön vereceği tasarım aşamasında belirlendiği için bu aşama bilginin inşasına katkıda bulunacaktır. Tasarım aşamasından sonra kişiselleştirme aşaması gerçekleştirilecektir ve bu aşamada bireysel öğrenme faaliyetleri önem kazanacaktır. Bireysel oyun projelerinde gerçekleştirilmek istenen eklentilerin tamamlanması, kişisel öğrenme sürecine destek sağlayarak öğrenenlerin bilgilerini genişletmelerine olanak tanıyacaktır. Geliştirilen bireysel oyun projeleri paylaşım aşamasında sosyal olarak paylaşılacak ve kişilerin kendi fikirlerinden öğeler taşıyan ürünler akranların karşısına çıkarılacaktır. Son aşama olan yansıtma aşamasında ise; bu ürün geliştirme sürecinde elde edilen deneyimler değerlendirilerek sürecin olumlu ve geliştirilmesi gereken yönleri üzerinde değerlendirilmelerde bulunulacaktır. Bu basamakları

tamamlamalarının neticesi olarak öğrenciler, bireysel öğrenme hedefleri doğrultusunda programlama teknikleri ile problemlerin çözümü arasındaki ilişkiyi keşfetme yolunda adımlar atmış olmaktadırlar (Ezeamuzie, 2022).

Proje tabanlı öğrenme (PTÖ), öğrencilerin bilgiyi doğrudan almak yerine bireysel ya da gruplar halinde gerçek hayatla ilişkili bir proje üzerinde çalışarak yapılandırdıkları ve ürüne dönüştürdükleri bir öğretim modelidir (Yılmaz ve Gültekin, 2007). Bu model öğrencileri merkeze alarak öğrencilerin özgün bir meydan okuma içeren karmaşık görevlere katılımını sağlamaya çalışır (Holm, 2011). Bu modele göre öğrenme, öğrenciler kişisel olarak anlamlı buldukları projeler üzerinde çalışırken gerçekleşmektedir (O'Grady-Jones, 2020). Proje oluşturma sürecinde yeni fikirler üretme, prototipler tasarlama ve tekrarlanan iyileştirme yapma, öğrencilerin problem çözme ve düşünme becerilerini geliştirmektedir (Grant ve Branch, 2005). Proje tabanlı öğrenme ortamlarında, öğrenciler kendi öğrenmelerini kurgulayıp yönlendirirler, proje geliştirme sürecinde karşılaştıkları engelleri iş birliği içinde aşmaya çalışırlar ve başarılı olabilmek için kendi kararlarını kendileri alırlar (Erdem, 2002). Dijital bir oyun geliştirme veya değiştirme projesi çeşitli becerilere sahip üyelerden oluşan bir ekibin koordineli bir çabasını gerektirdiğinden, proje tabanlı öğrenme oyun geliştirmeye dayalı öğretme açısında uygun bir yöntemdir (Sillaots ve Fiadotau, 2018).

Programlama öğretimi bağlamında değerlendirildiğinde PTÖ, öğrencilerin anlamlı buldukları gerçek hayatla alakalı programlama problemleriyle yüzleşmelerini, bunları nasıl ele alacaklarını belirlemelerini ve daha sonra problem çözümleri oluşturacak şekilde hareket etmelerini temel alan bir öğretim modelidir (Bender, 2012). Linares-Pellicer ve diğerlerine (2020) göre bu yaklaşımda programlama projelerinin iki temel kriteri karşılaması beklenir. Bunlardan ilki, programlama projelerinin öğrenciler açısından anlamlı olması gerekliliğidir. Öğrenciler projeleri kişisel olarak iyi yapmak istedikleri bir şey olarak algılamalıdır çünkü projeler önemli ve zorludur. İkinci olarak, projenin öğretim amaçlı olması ve öğrencilerin öğrenme hedeflerine odaklanması gerekliliğidir. Oyun geliştirmeye dayalı programlama öğretiminde kullanılan projelerin hem öğretim hedeflerini barındırdığı hem de öğrenciler açısından anlamlı ve zorlu görevleri içerdiğini söylemek mümkündür.

BÖLÜM III: YÖNTEM

Bu bölümde; araştırmanın modeline, araştırma sürecine, çalışma grubuna, araştırmada kullanılan veri toplama araçlarına, araştırmacının rolüne, verilerin analizine ve araştırmanın geçerliğine - güvenirliğine yönelik açıklamalara yer verilmiştir.

3.1. Araştırma Modeli

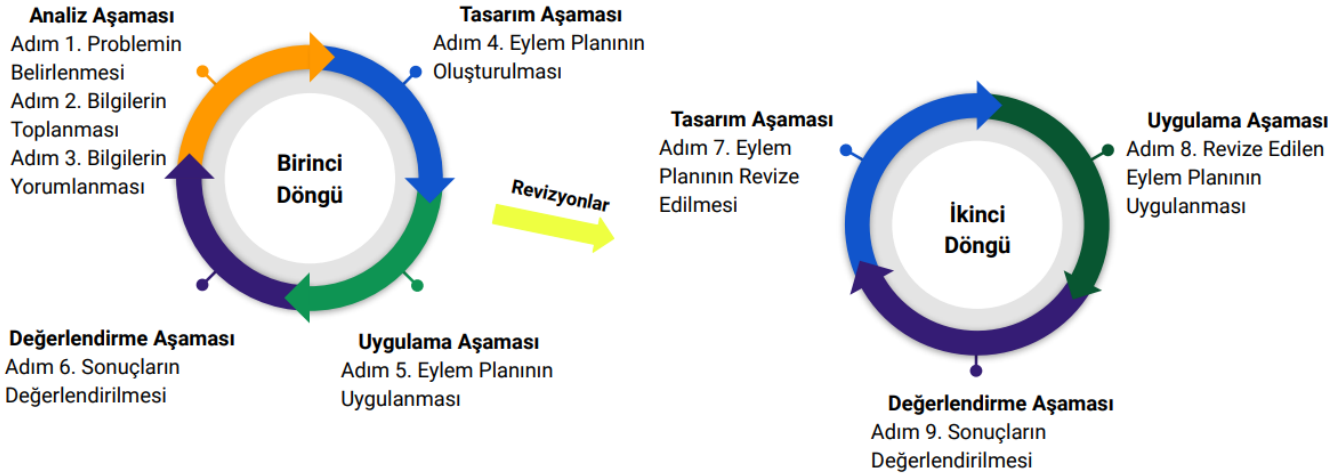
Bu tez çalışması eylem araştırması olarak tasarlanmıştır. Her ne kadar alanyazında eylem araştırmaları için farklı tanımlamalar bulunsa da eylem araştırmaları "*bir grup insanın bir problemi tanımlaması, problemi çözmek için bir şeyler yapması, çabalarının ne kadar başarılı olduğunu görmesi, sonuçtan tatmin olmazlarsa yeni çözüm yolları araması; özetle yaparak ve yaşayarak öğrenmesi"* basamaklarını içermektedir (O'Brien, 1998). Bu basamaklar ve prensipler, bu tez çalışmasının tasarlanması, uygulanması ve değerlendirmesi aşamalarında yol gösterici olmuştur. Bu çerçevede; (1) araştırmacının vermekte olduğu NYP dersinde öğrencilerin ve öğretim elemanlarının yaşadıkları sorunlar ortaya çıkarılmış, (2) mevcut sorunların en aza indirilebilmesi ve dersin verimliliğinin artırılması amacıyla mezun öğrencilerin ve derse giren öğretim elemanlarının sorunların çözümüne yönelik önerileri alınmış ve (3) NYP dersine oyun geliştirmeye dayalı öğretim stratejisinin entegre edildiği bir eylem planı geliştirilerek NYP dersinin "*geliştirilmesi ve iyileştirilmesi*" amaçlanmıştır. Bu çalışma; araştırmacının kendi dersinde karşılaştığı sorunlara çözüm yollarının aranması, dersin geliştirme ve iyileştirme sürecinin sistematik bir döngü içerisinde gerçekleştirilmesi ve dersin verimliliğinin artırılması amacıyla sürekli daha yararlı olana ulaşmayı amaçlaması sebebiyle eylem araştırması olarak tasarlanmış ve uygulanmıştır (Mertler, 2006).

Alanyazında eylem araştırmalarının uygulama süreçleri bağlamında her ne kadar farklı yaklaşımların olduğu ve takip edilmesi gereken basamaklar için evrensel bir sıralamanın olmadığı görülse de (Fraenkel ve Wallen, 2003; Johnson, 2014) yine de eylem araştırmalarında bazı temel süreçlerin takip edildiği görülmektedir. Bu tez çalışması kapsamında Kuzu'nun (2009) eylem araştırmalarının basamakları olarak belirttiği; (1) problemin ve araştırma sorularının belirlenmesi, (2) problemin çözümüne yönelik eylemlerin planlanması, (3) eylem planlarının uygulanması, (4) uygulanan eylem planlarının etkinliğine yönelik verilerin toplanması ve (5) bu sürece yönelik yansıtmaların yapılması süreçleri uygulanmıştır. Bunun yanında eylem planın etkinliğinin değerlendirilmesi bağlamında birçok araştırmacının (Kuzu, 2009; Mertler, 2009; Yıldırım ve Şimşek, 2011)

uygulanabilirliğini belirttiği gibi anket ve ölçekler, günlükler, yarı-yapılandırılmış veya yapılandırılmamış görüşmeler, odak grup görüşmeleri, başarı sınavları, öğrenci ürünleri, rubrikler gibi hem nitel hem de nicel veri toplama araçlarından yararlanılmıştır. Bu tez çalışması kapsamında toplanan verilerin analizinde hem nitel hem de nicel analiz yöntemleri kullanılmıştır.

3.2. Eylem Araştırması Süreci

Bu çalışma kapsamında iki döngüden oluşan ve Ferrance (2000) ile Mertler'den (2009) uyarlanan bir eylem araştırması yürütülmüştür (Şekil 3.1). Bu uyarlama; *sistematik, dinamik, döngüsel ve işbirliğine dayalı* olarak gerçekleştirilen eylem araştırmalarının tek bir döngüden oluşabileceği gibi birden fazla döngü de içerebileceği prensibini temel almıştır (Mertler, 2009; Stringer, 2004).



Şekil 3.1. Eylem araştırması süreci

Şekil 3.1'de belirtildiği gibi çalışmanın birinci döngüsü *analiz, tasarım, uygulama* ve *değerlendirme* basamaklarını içermiştir. *Analiz aşamasında* mevcut NYP dersinde öğrencilerin ve öğretim elemanlarının yaşadıkları sorunlar tespit edilmiştir. Bunun yanında öğrencilerin NYP dersi hakkındaki görüşleri; hangi türlerde yazılım projesi geliştirmenin kendileri için daha yararlı olacağı yönündeki düşünceleri elde edilmiştir. *Tasarım aşamasında* mevcut NYP dersindeki sorunlara etkili çözümler üretebilmek için NYP dersi yeniden tasarlanmıştır. Bu süreçte; analiz aşamasında toplanan verilerden, alınan uzman görüşlerinden ve alanyazından destek alınarak NYP dersinin oyun geliştirmeye dayalı olarak öğretilmesi yönünde bir eylem planı oluşturulmuştur. *Uygulama aşamasında*, tasarlanan

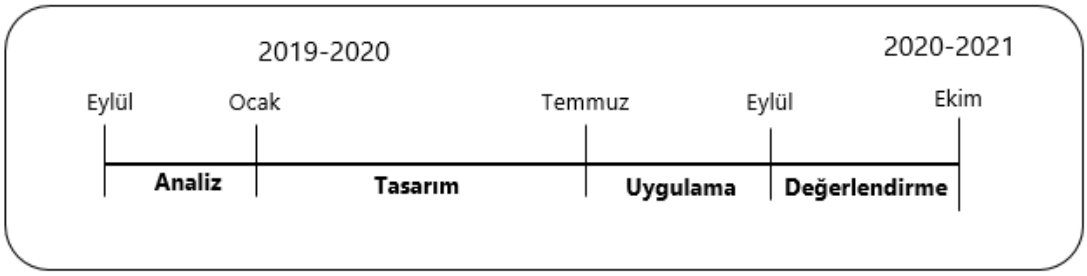
eylem planı hayata geçirilmiştir. *Değerlendirme aşamasında* ise uygulanan eylem planının etkililiği değerlendirilerek yansıtma yapılmıştır.

Birinci döngüsü tamamlandıktan sonra çalışmanın ikinci döngüsüne geçilmiştir. Bu döngü *tasarım*, *uygulama* ve *değerlendirme* basamaklarını içermiştir (Şekil 3.1). *Tasarım aşamasında*, birinci döngüden elde edilen deneyimler ışığında geliştirilen NYP dersinde bazı revizyonlar gerçekleştirilmiştir. *Uygulama aşamasında*, gerçekleştirilen değişiklikleri barındıran revize edilmiş eylem planı tekrar uygulanmıştır. *Değerlendirme aşamasında* ise uygulanan eylem planının başarısı değerlendirilmiş ve yansıtma yapılmıştır. Her iki döngüde tamamlanan aşamaların detaylı açıklamaları aşağıda yer alan alt maddelerde verilmiştir.

3.2.1. Birinci Döngü

3.2.1.1. Araştırma süreci

Çalışmanın birinci döngüsü 2019-2020 Eğitim-Öğretim yılında tamamlanmıştır. 2019 yılı eylül ayında analiz aşaması ile başlayan bu süreç, 2020 yılının eylül ayında değerlendirme aşaması ile son bulmuştur. Eylem araştırmasının birinci döngüsünün zaman çizelgesi Şekil 3.2'de verilmiştir. Mevcut NYP dersinin ihtiyaç analizlerinin gerçekleştirildiği analiz aşaması bu tez çalışması için kritik derecede önemlidir çünkü bu aşama NYP dersinin sorunlarını çözebilecek etkili bir eylem planının geliştirilebilmesinin temelini oluşturmaktadır (Ferrance, 2000; Schmuck, 2006). Analiz aşamasında gerçekleştirilen işlemler Tablo 3.1'de verilmiştir.



Şekil 3.2. Birinci döngünün zaman çizelgesi

Tablo 3.1. Birinci döngünün analiz aşamasında gerçekleştirilen işlemler

Gerçekleştirilen İşlemler	**İşlem Açıklamalar**
Analiz Aşaması Veri Toplama Araçlarının Hazırlanması	▪ Analiz Aşaması Katılımcı Anketinin hazırlanması. ▪ Analiz Aşaması Öğretim Elemanı Görüşme Formunun hazırlanması.
Analiz Aşaması Verilerinin Toplanması	▪ Daha önce NYP dersini almış mezun öğrencilerin mevcut NYP dersi hakkındaki görüşlerinin elde edilmesi. ▪ Mevcut NYP dersinde öğrencilerin karşılaştıkları sorunlara, bu sorunların çözümüne ve öğrencilerin dersten beklentilerine yönelik verilerin toplanması. ▪ Mezun öğrencilerin NYP dersinde ne tür yazılım projeleri geliştirmenin kendileri için daha yararlı olacağına yönelik görüşlerinin elde edilmesi. ▪ Mezun öğrencilerin NYP dersinde yazılım projesi olarak geliştirmeyi tercih ettikleri dijital oyun türlerine yönelik görüşlerinin alınması. ▪ NYP dersini vermekte olan öğretim elemanlarının derste yaşadığı sorunlara ve bu sorunların çözümüne yönelik öğretim elemanları ile yarı yapılandırılmış görüşmelerin yapılması. ▪ Mevcut NYP dersine oyun geliştirmeye dayalı programlama öğretimi stratejisinin nasıl entegre edilebileceği konusunda mezun öğrenci ve öğretim elemanlarının görüşlerinin elde edilmesi.
Analiz Aşaması Verilerinin Analizi	▪ Mezun öğrencilerin ve öğretim elemanlarının mevcut NYP dersinde yaşadıkları sorunların ortaya çıkarılması. ▪ Mezun öğrencilerin ve öğretim elemanlarının mevcut NYP dersindeki sorunlara yönelik çözüm önerilerinin belirlenmesi. ▪ Mezun öğrencilerin mevcut NYP dersinde geliştirmeyi tercih ettikleri yazılım türlerinin ve bunları tercih etme nedenlerinin belirlenmesi. ▪ Mevcut NYP dersine oyun geliştirmeye dayalı programlama öğretimi stratejisinin nasıl entegre edilebileceği konusunda mezun öğrenci ve öğretim elemanlarının görüşlerinin ortaya çıkarılması. ▪ Öğrencilerin mevcut NYP dersinden beklentilerinin belirlenmesi.

Analiz aşamasından sonra tasarım aşamasına geçilmiştir. *Tasarım aşaması* 2019-2020 Eğitim-Öğretim Yılı aralık ile temmuz ayları arasında gerçekleşmiştir. Bu aşamada gerçekleştirilen işlemler Tablo 3.2'de verilmiştir.

Tablo 3.2. Birinci döngünün tasarım aşamasında gerçekleştirilen işlemler

Gerçekleştirilen İşlemler	İşlem Açıklamaları
Mevcut NYP Dersinde Yaşanan Sorunlara Yönelik Çözüm Önerilerinin Belirlenmesi	▪ Mevcut NYP dersine oyun geliştirmeye dayalı programlama öğretimi stratejisinin nasıl entegre edilebileceği yönünde mezun öğrenci ve öğretim elemanlarından elde edilen görüşlerin çözümlenmesi. ▪ Alanyazın taraması.
Eylem Planının Geliştirilmesi	▪ NYP dersinin öğretim tasarımın gerçekleştirilmesi.
Programlama Kaygı Ölçeğinin Geliştirilmesi	▪ Ölçek taslak madde havuzunun oluşturulması. ▪ Taslak maddeler kullanılarak verilerin toplanması. ▪ Toplanan veriler vasıtasıyla Açımlayıcı Faktör Analizinin yapılması. ▪ Taslak ölçek kullanılarak yeniden veri toplanılarak Doğrulayıcı Faktör Analizinin yapılması.
Programlama Öz Yeterlilik Ölçeğinin Uyarlanması	▪ Ölçek maddeleri kullanılarak veri toplanması. ▪ Toplanan veriler vasıtasıyla Açımlayıcı ve Doğrulayıcı Faktör Analizinin yapılması.
Diğer Veri Toplama Araçlarının Belirlenmesi ve Hazırlanması	▪ Katılımcı Bilgi Formunun hazırlanması. ▪ NYP Başarı Sınavının hazırlanması. ▪ Proje Değerlendirme Rubriğinin hazırlanması. ▪ Kullanılacak diğer ölçeklerin belirlenmesi ve kullanım izinlerinin alınması.

Tablo 3.2'de belirtildiği gibi tasarım aşaması mevcut NYP dersinde karşılaşılan sorunları çözmek için bir eylem planının geliştirildiği aşama niteliğindedir. Eylem planının geliştirilmesi sürecinde oyun geliştirmeye dayalı programlama öğretimin mevcut NYP dersine nasıl entegre edilebileceği üzerinde durulmuştur. Bu aşamada analiz aşamasında toplanan verilerden ve alanyazından faydalanılmıştır. Tasarım aşamasında bir eylem planı geliştirilmiş ve bu eylem planına uygun olarak NYP dersi yeniden tasarlanmıştır. Bu kapsamda Morrison, Ross ve Kemp Modeli (2004) kullanılarak NYP dersinin öğretim tasarımı (instructional design) gerçekleştirilmiştir. Tasarım aşamasında gerçekleştirilen son etkinlik ise geliştirilen eylem planının etkinliğinin değerlendirilmesinde kullanılacak veri toplama araçlarının belirlenmesi ve hazırlanması olmuştur. Bu süreçte kullanılacak veri toplama araçlarından bazıları araştırmacı ve tez danışmanı tarafından hazırlanmış; alanyazında mevcut olanların ise kullanım izinleri alınmıştır.

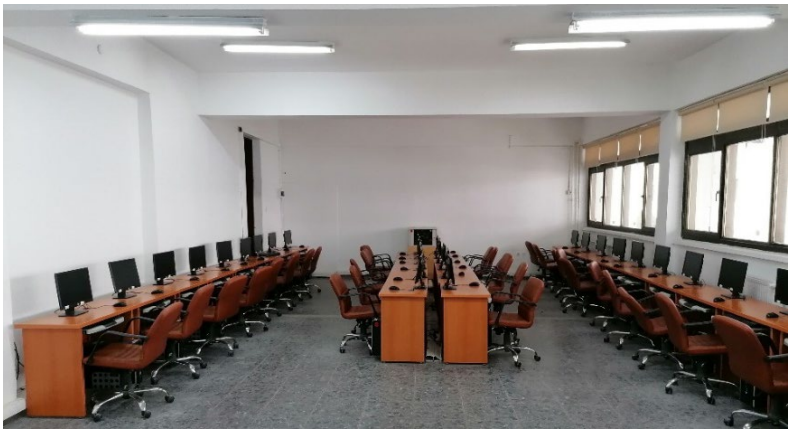
Birinci döngü *uygulama aşaması* 2019-2020 Eğitim-Öğretim Yılı temmuz ve ağustos ayları arasında sekiz hafta sürecek şekilde gerçekleşmiştir. Eylem planının uygulanması kapsamında her hafta yüz yüze olmak üzere üç saat ders işlenmiş; bu derslerin ilk saati teorik,

diğer saatleri uygulamalı olarak icra edilmiştir. Bu süreçte araştırmacının ve katılımcıların karşılaştıkları sorunlar kayıt altına alınmıştır.

Birinci döngü *değerlendirme aşaması* 2019-2020 Eğitim-Öğretim Yılı eylül ayında gerçekleşmiştir. Bu aşamada eylem planının ne kadar başarılı olduğu konusunda yansıtmalar yapılmıştır. Değerlendirme aşamasında eylem planının uygulanması esnasında karşılaşılan sorunlara yönelik çözüm önerileri geliştirilmiştir. Bunun yanında geliştirilen NYP dersinde yapılabilecek iyileştirmeler belirlenmiş ve derste bazı revizyonlara gidilmiştir.

3.2.1.2. Uygulama ortamı

Bu tez çalışmasının benzer koşullarda yapılacak araştırmalara kaynak teşkil edebileceği değerlendirildiğinde, eylem araştırmasının gerçekleştirildiği ve verilerin toplandığı ortamının detaylı tasviri önemli bir konudur (Ekiz, 2004). Araştırmanın birinci döngüsü Türkiye'deki bir meslek yüksekokulunun bilgisayar teknolojisi bölümünün bilgisayar sınıfında yüz yüze sınıf ortamında gerçekleştirilmiştir. Bilgisayar sınıfının unsurları; (1) öğretim elemanı ve öğrenci bilgisayarları, (2) bu bilgisayarlara ait bilgisayar masaları ve (3) yazı tahtasıdır. Bilgisayar sınıfında yer alan tüm bilgisayarlar bir anahtarlama cihazına (switch) bağlıdır ve aralarında haberleşme mevcuttur. Bu haberleşme sayesinde tüm bilgisayarlardan okulun ders yönetim sistemine ulaşılabilmektedir. Bilgisayar sınıfında herhangi bir projeksiyon cihazı bulunmamaktadır ancak öğretim elemanları dershanelerde Netsupport yazılımından faydalanarak ekran paylaşımı yapabilmektedir. Şekil 3.3'te bilgisayar sınıfının genel görünümü tasvir edilmiştir.



Şekil 3.3. Birinci döngü uygulama ortamı

3.2.1.3. Çalışma grubu

Bu tez çalışmasının birinci döngüsünde; analiz/tasarım aşamasında iki çalışma grubu; esas uygulamanın yapıldığı uygulama aşamasında ise bir çalışma grubu olmak üzere toplamda üç farklı çalışma grubu yer almıştır. Bu çalışma gruplarına ait bilgiler Tablo 3.3'te sunulmuştur.

Tablo 3.3. Birinci döngünün aşamaları ve katılımcıları

Eylem Araştırması Aşaması	**Katılımcı Türü**	**N**
Analiz / Tasarım	Mezun Öğrenci	59
	Öğretim Elemanı	6
Uygulama	Uygulama Katılımcısı Ön lisans Öğrencisi	29

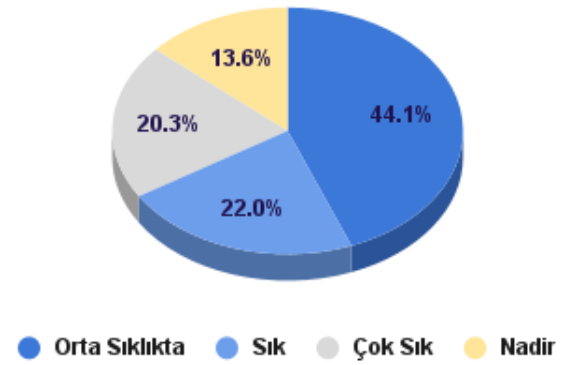
Tablo 3.3'te gösterildiği üzere, araştırmanın birinci döngüsünün analiz/tasarım aşamasının çalışma grubunu daha önce NYP dersini almış ve okullarından mezun olmuş 59 öğrenci ile NYP dersini vermekte olan altı öğretim elemanı oluşturmuştur. Mezun öğrenci grubunu bu eylem araştırmasının uygulandığı meslek yüksekokulundan mezun olmuş öğrencilerle Türkiye'deki bir devlet üniversitesinin BÖTE bölümünden mezun olmuş öğrenciler oluşturmuştur. Öğretim elemanı çalışma grubu ise bu uygulamanın gerçekleştirildiği meslek yüksekokulunda NYP dersini vermekte olan öğretim elemanları oluşturmuştur. Analiz/tasarım aşamasında yer alan iki çalışma grubuna ilave olarak birinci döngünün eylem planının uygulandığı basamak olan uygulama aşamasının çalışma grubunu 29 ön lisans öğrencisi oluşturmuştur. Tüm bu katılımcılara ait bilgiler izleyen maddelerde detaylı bir şekilde açıklanmıştır.

3.2.1.3.1. Mezun öğrencilerden oluşan çalışma grubu

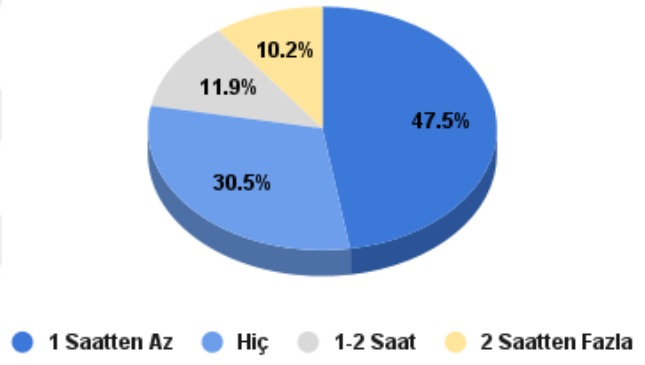
Analiz/tasarım aşamasının ilk çalışma grubunu oluşturan mezun öğrencilere ait demografik özellikler Tablo 3.4'te sunulmuştur. Tablo 3.4'e göre 59 mezun öğrencinin büyük çoğunluğunun (N:53; %90) 17-24 yaş aralığında olduğu görülmektedir. Benzer şekilde katılımcıların büyük çoğunluğu (N:54; %91) lise eğitimlerini Anadolu Lisesi ve Mesleki ve Teknik Anadolu Lisesinde tamamlamıştır. Demografik özelliklerine ilave olarak katılımcı mezun öğrencilerin günlük bilgisayar kullanımı sıklıkları, dijital oyun oynama sıklıkları, algıladıkları NYP bilgi ve zorluk seviyeleri Şekil 3.4-Şekil 3.7'de sunulmuştur.

Tablo 3.4. Mezun katılımcılara ait demografik özellikler

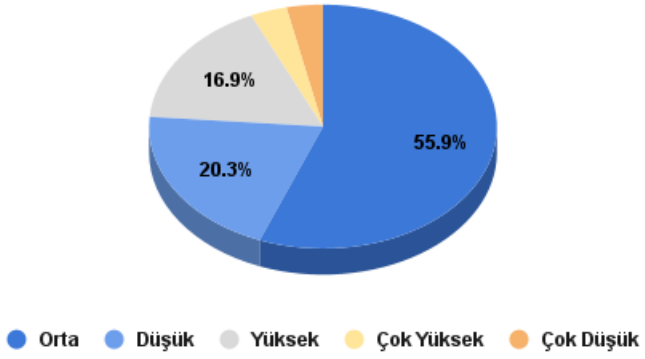
		Frekans (f)	Yüzde (%)
Cinsiyet	Kadın	22	37
	Erkek	37	63
Yaş (Yıl)	17-24	53	90
	25-29	4	7
	30-39	2	3
Eğitim Durumu	Lisans Mezunu	35	60
	Ön lisans Mezunu	24	40
Mezun Olunan Ortaöğretim Kurumu	Anadolu Lisesi	29	49
	Mesleki ve Teknik Anadolu Lisesi	25	42
	Anadolu İmam-Hatip Lisesi	3	6
	Açık Öğretim Lisesi	2	3



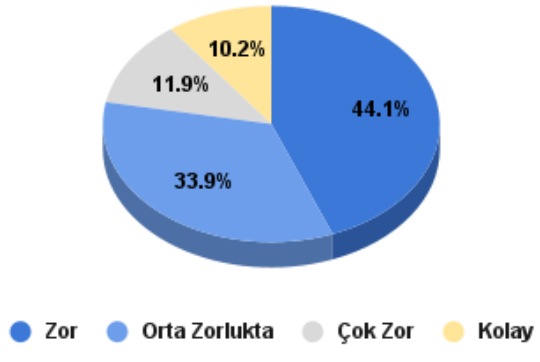
Şekil 3.4. Katılımcıların günlük bilgisayar kullanımı sıklıkları



Şekil 3.5. Katılımcıların günlük dijital oyun oynama sıklıkları



Şekil 3.6. Katılımcıların algıladıkları NYP bilgi seviyeleri



Şekil 3.7. Katılımcıların algıladıkları NYP zorluk seviyeleri

Şekil 3.4'e göre katılımcıların çoğunun (N:39, %66) günlük bilgisayar kullanımı orta sıklıkta ve sıktır. Bunun yanında katılımcıların çoğu (N:46, %78) günlük olarak ya hiç dijital oyun oynamamakta ya da 1 saatten az oynamaktadır (Şekil 3.5). Şekil 3.6'ya göre katılımcıların %76'sının algıladıkları NYP bilgi seviyesi *"düşük"* ve "*orta*" seviyedir. Aynı

zamanda katılımcıların büyük çoğunluğunun (N:46; %78) NYP dersini "*orta zorlukta*" ve "*zor*" olarak nitelendirdiği görülmektedir (Şekil 3.7). Bu tez çalışması boyunca araştırmaya katılan mezun öğrencilerinin isimleri M1, M2, M3,…,M59 şekilde kodlanmıştır.

3.2.1.3.2. Öğretim elemanlarından oluşan çalışma grubu

Birinci döngünün Analiz ve Tasarım aşamasında daha önce NYP dersini almış ve üniversitelerinden mezun olmuş öğrencilere ilave olarak NYP dersini vermekte olan öğretim elemanları da yer almıştır. Araştırmanın bu aşamasında öğretim elemanlarından NYP dersinde karşılaştıkları sorunlar ile bu sorunların çözümüne yönelik önerileri elde edilmiştir. Bununla birlikte öğretim elemanlarından mevcut NYP dersine oyun geliştirmeye dayalı programlama öğretiminin nasıl entegre edilebileceği konusunda uzman görüşü elde edilmiştir. Çalışmaya katılan öğretim elemanlarına ait demografik özellikler Tablo 3.5'te sunulmuştur.

Tablo 3.5. Öğretim elemanlarına ait demografik özellikler

		Frekans (f)
Cinsiyet	Kadın	1
	Erkek	5
Yaş (Yıl)	20-25	1
	26-30	2
	31-35	1
	36-40	2
Eğitim Durumu	Lisans	1
	Yüksek Lisans	3
	Doktora	2
Branş	BÖTE	6
Mesleki Deneyim (Yıl)	1-5	1
	6-10	1
	10-15	2
	15-20	2

Tablo 3.5'e göre çalışmaya katılan çoğunluğu erkek olup tamamı lisans eğitimlerini BÖTE alanında tamamlamıştır. Bu tez çalışması boyunca araştırmaya katılan öğretim elemanlarının isimleri ÖE1, ÖE2, ÖE3, ÖE4, ÖE5 ve ÖE6 şekilde kodlanmıştır.

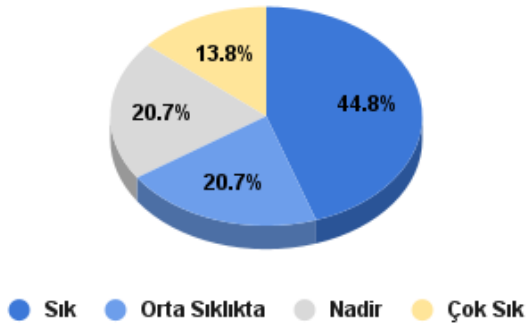
3.2.1.3.3. Uygulama aşaması çalışma grubu

Birinci döngünün tasarım aşamasında belirlenen eylem planın uygulandığı basamak olan uygulama aşamasına bir devlet meslek yüksekokulunun Bilgisayar Teknolojisi Bölümü ikinci sınıfında öğrenim görmekte olan 29 ön lisans öğrencisi katılmıştır. Bu çalışma grubunun tamamı erkek, 17-24 yaş aralığında ve ön lisans öğrencisidir. Katılımcıların büyük çoğunluğunun (N:22; %76) genel akademik not ortalaması üç ve üzeridir (Tablo 3.6). Benzer şekilde katılımcıların büyük çoğunluğunun (N: 28; %97) mezun olduğu orta öğretim kurumu Anadolu Lisesi ve Mesleki ve Teknik Anadolu Lisesidir.

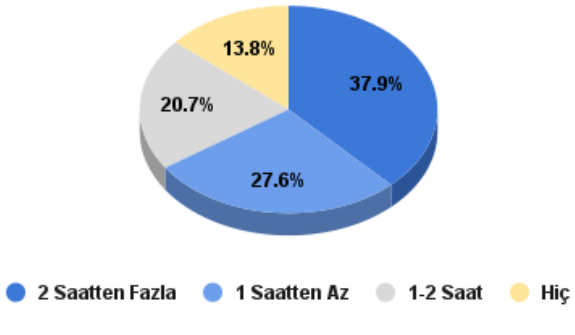
Tablo 3.6. Birinci döngü uygulama aşaması katılımcılarına ait demografik özellikler

		Frekans (f)	Yüzde (%)
Genel Akademik Not Ortalaması (Ön lisans)	0-2.00	-	-
	2.01-2.50	1	3
	2.51-3.00	6	21
	3.01-3.50	15	52
	3.51-4.00	7	24
Mezun Olunan Ortaöğretim Kurumu	Anadolu Lisesi	22	76
	Mesleki ve Teknik Anadolu Lisesi	6	20
	Anadolu İmam-Hatip Lisesi	1	4

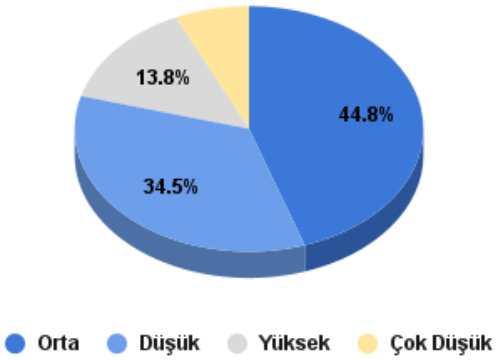
Demografik özelliklerine ilave olarak birinci döngü çalışma grubunun günlük bilgisayar kullanımı sıklıkları, dijital oyun oynama sıklıkları, algıladıkları NYP bilgi ve zorluk seviyeleri Şekil 3.8-Şekil 3.11'de sunulmuştur.



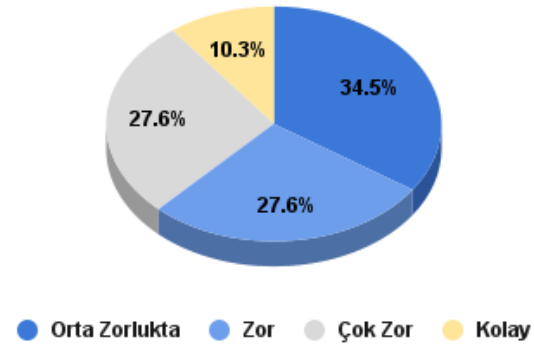
Şekil 3.8. Katılımcıların günlük bilgisayar kullanımı sıklıkları



Şekil 3.9. Katılımcıların günlük dijital oyun oynama sıklıkları

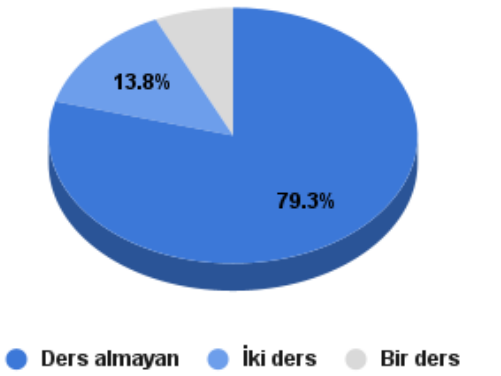


Şekil 3.10. Katılımcıların algıladıkları NYP bilgi seviyeleri



Şekil 3.11. Katılımcıların algıladıkları NYP zorluk seviyeleri

Şekil 3.8'e göre katılımcıların çoğunun (N:19, %66) günlük bilgisayar kullanımı orta sıklıkta ve sıktır. Bunun yanında katılımcıların %58.6'sı (N:17) günlük olarak 1 saat ve üzeri dijital oyun oynamaktadır (Şekil 3.9). Şekil 3.10'a göre katılımcıların %79'unun algıladıkları programlama bilgi seviyesi *"orta"* ve *"düşük"* seviyedir. Aynı zamanda katılımcıların büyük çoğunluğunun (N:26; %90) programlamayı "*orta zorlukta*", "*zor*" ve "çok *zor*" olarak nitelendirdiği görülmektedir (Şekil 3.11). Tüm bu bilgilere ilave olarak birinci döngü uygulama aşaması katılımcılarının programlama ve oyun geliştirme tecrübeleri Şekil 3.12 ve Şekil 3.13'te sunulmuştur.



Şekil 3.12. Katılımcıların ortaöğretimde programlama dersi alma durumları

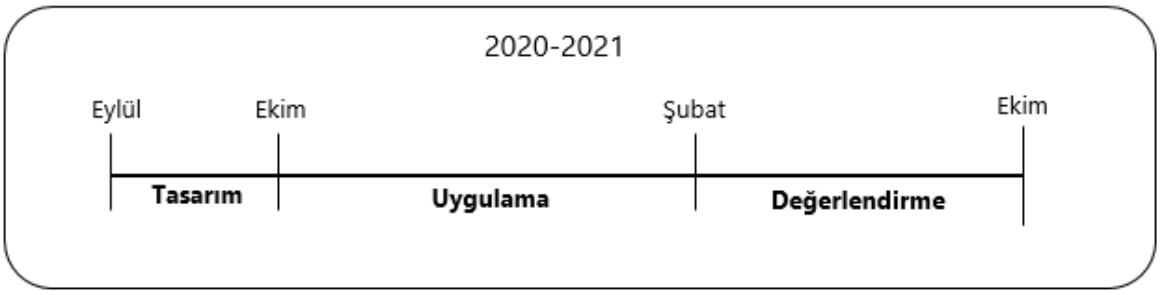
Şekil 3.13. Katılımcıların oyun geliştirme tecrübeleri

Şekil 3.12'e göre katılımcıların çoğunun (N:23; %79) ortaöğretimde programlama dersi almadıkları görülmektedir. Benzer şekilde bir katılımcı hariç olmak üzere katılımcıların daha önce oyun geliştirme deneyiminin olmadığı söylenebilir (Şekil 3.13). Bu tez çalışması boyunca araştırmanın birinci döngüsüne katılan öğrencilerinin isimleri D1_K1, D1_K2, D1_K3,…,D1_K29 şekilde kodlanmıştır.

3.2.2. İkinci Döngü

3.2.2.1. Araştırma süreci

Çalışmanın birinci döngüsü tamamlandıktan sonra ikinci döngüsüne başlanmıştır. Bu süreç 2020 yılının eylül ayında tasarım aşaması ile başlamış; 2021 yılının ekim ayında değerlendirme aşaması ile son bulmuştur. Eylem araştırmasının ikinci döngüsünün zaman çizelgesi Şekil 3.14'te verilmiştir.



Şekil 3.14. İkinci döngü araştırma süreci

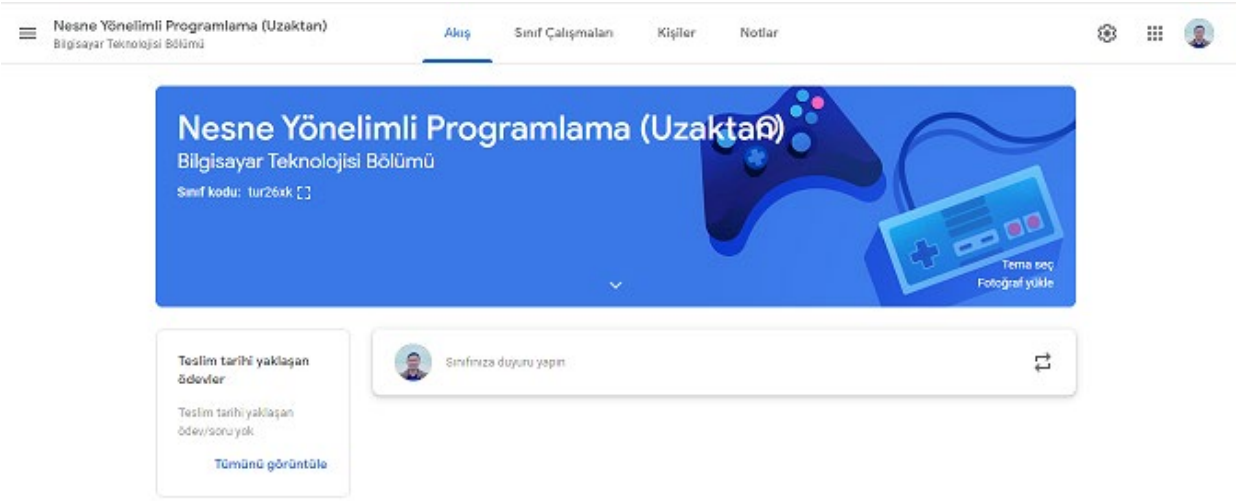
İkinci döngü *tasarım aşaması* 2020-2021 Eğitim-Öğretim Yılı eylül ve ekim aylarını kapsamıştır. Bu aşamada birinci döngüden elde edilen bulgular ve deneyimlerden yararlanılarak geliştirilen NYP dersinde dersin içeriği, dersin ortamı (çevrimiçi öğrenme ortamı) ve ders materyalleri (oyun prototipi vb.) bağlamında bazı değişiklikler yapılmıştır. Buna ilave olarak birinci döngüde yer almayıp ikinci döngüde kullanılması kararlaştırılan veri toplama araçları belirlenmiştir.

İkinci döngü *uygulama aşaması* 2020-2020 Eğitim-Öğretim Yılı ekim ile şubat ayları arasında on beş hafta sürecek şekilde gerçekleşmiştir. Eylem planının uygulanması kapsamında her hafta üç ders saati olacak şekilde canlı dersler işlenmiştir. Bu süreçte araştırmacının ve katılımcıların karşılaştıkları sorunlar kayıt altına alınmıştır.

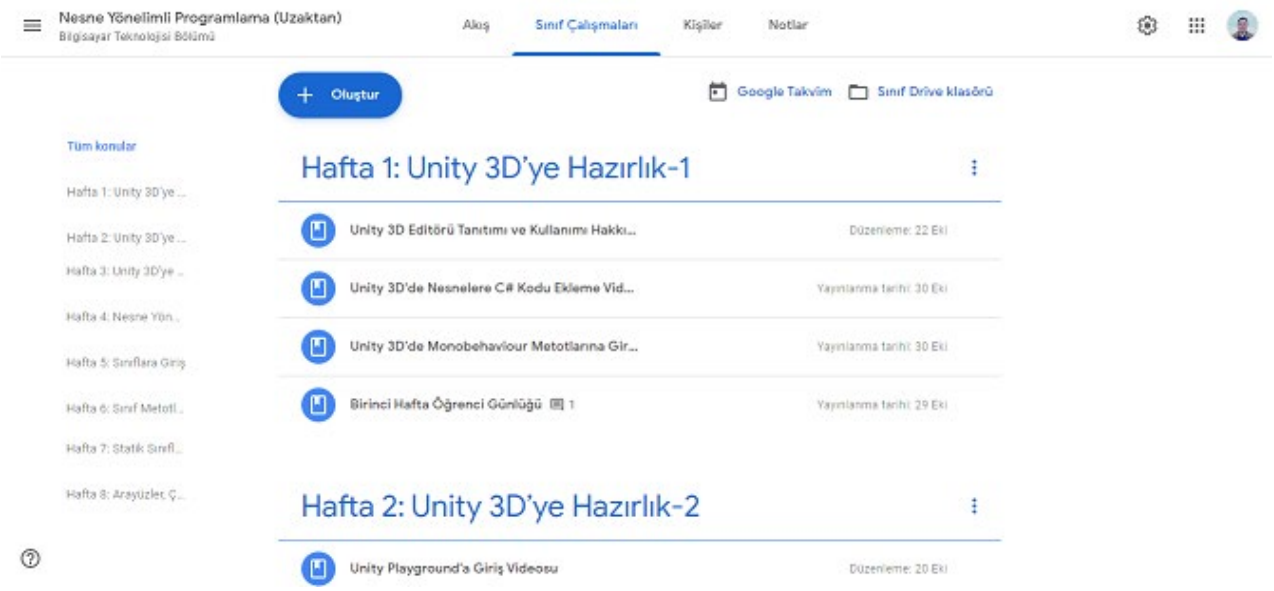
İkinci döngü *değerlendirme aşaması* 2020-2021 Eğitim-Öğretim Yılı şubat ile ekim ayları arasında gerçekleşmiştir. Bu aşamada eylem planının ne kadar başarılı olduğu konusunda yansıtmalar yapılmıştır. Değerlendirme aşamasında eylem planının uygulanması esnasında karşılaşılan sorunlara yönelik çözüm önerileri geliştirilmiştir. Bunun yanında geliştirilen NYP dersinde yapılabilecek iyileştirmeler belirlenmiştir.

3.2.2.2. Uygulama ortamı

Çalışmanın ikinci döngüsünde pandemi nedeniyle acil uzaktan öğretim yöntemine geçilmiş olup dersler çevrimiçi olarak işlenmiştir. Çevrimiçi derslerin işlenmesinde Zoom yazılımı kullanılmıştır. Bunun yanında öğrencilerin ders etkinliklerine, duyurularına ve kaynaklarına erişimini kolaylaştırmak maksadıyla Google Classroom kullanılarak tüm öğrencilerin katılımlarının sağlandığı bir sanal sınıf oluşturulmuştur. Google Classroom sanal sınıfa ait sınıf akış ekranı Şekil 3.15'te, sınıf çalışmaları arayüzü ise Şekil 3.16'da sunulmuştur.



Şekil 3.15. Kullanılan sanal sınıfın akış ekranı (Google Classroom)



Şekil 3.16. Kullanılan sanal sınıfın sınıf çalışmaları arayüzü

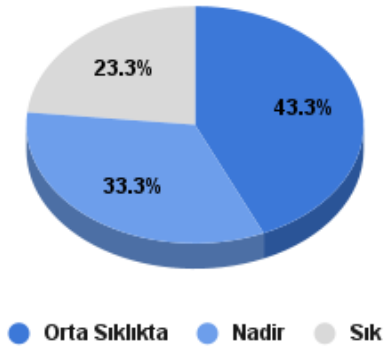
3.2.2.3. Çalışma grubu

İkinci döngünün uygulama aşamasına bir devlet meslek yüksekokulunun Bilgisayar Teknolojisi Bölümü ikinci sınıfında öğrenim görmekte olan 30 ön lisans öğrencisi katılmıştır. Bu çalışma grubunun tamamı erkek, 17-24 yaş aralığında ve ön lisans öğrencisidir. Katılımcıların büyük çoğunluğunun (N:27; %90) genel akademik not ortalaması 2.50 ve üzeridir (Tablo 3.7). Benzer şekilde katılımcıların büyük çoğunluğunun (N: 26; %87) mezun olduğu orta öğretim kurumu Anadolu Lisesi ve Mesleki ve Teknik Anadolu Lisesidir.

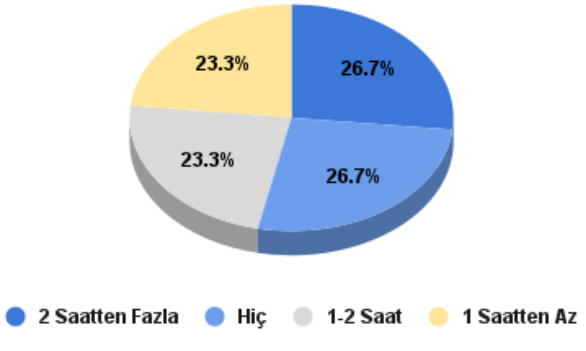
Tablo 3.7. İkinci döngü katılımcılarına ait demografik özellikler

		Frekans (f)	Yüzde (%)
Genel Akademik Not Ortalaması (Ön lisans)	0-2.00	-	-
	2.01-2.50	3	10
	2.51-3.00	12	40
	3.01-3.50	13	43
	3.51-4.00	2	7
Mezun Olunan Ortaöğretim Kurumu	Anadolu Lisesi	19	63
	Mesleki ve Teknik Anadolu Lisesi	7	24
	Açık Öğretim Lisesi	2	7
	Sosyal Bilimler Lisesi	1	3
	Anadolu İmam-Hatip Lisesi	1	3

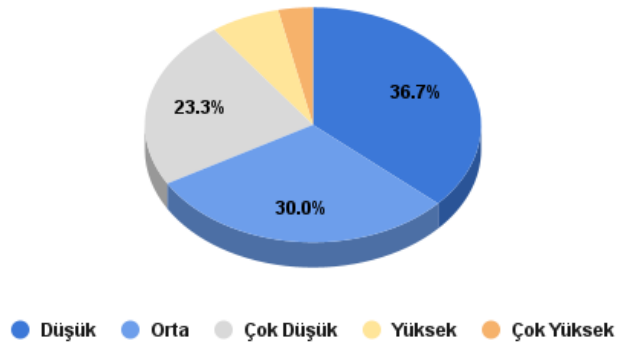
Demografik özelliklerine ilave olarak ikinci döngü çalışma grubunun günlük bilgisayar kullanımı sıklıkları, dijital oyun oynama sıklıkları, algıladıkları NYP bilgi ve zorluk seviyeleri Şekil 3.17-Şekil 3.20'de sunulmuştur.



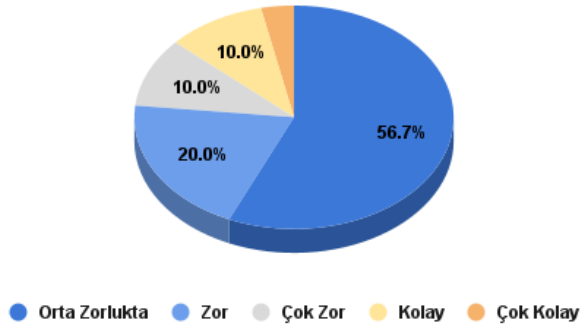
Şekil 3.17. Katılımcıların günlük bilgisayar kullanımı sıklıkları



Şekil 3.18. Katılımcıların günlük dijital oyun oynama sıklıkları

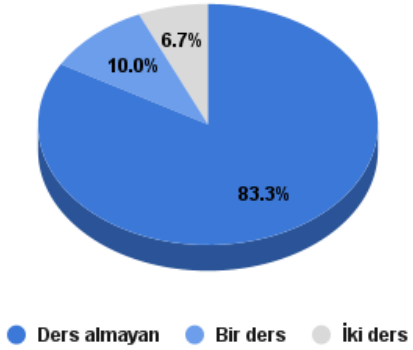


Şekil 3.19. Katılımcıların algıladıkları NYP bilgi seviyeleri



Şekil 3.20. Katılımcıların algıladıkları NYP zorluk seviyeleri

Şekil 3.17'ye göre katılımcıların çoğunun (N:20, %67) günlük bilgisayar kullanımı orta sıklıkta ve sıktır. Bunun yanında katılımcıların %50'si (N:15) günlük olarak 1 saat ve üzeri dijital oyun oynamaktadır (Şekil 3.18). Şekil 3.19'a göre katılımcıların %67'sinin algıladıkları programlama bilgi seviyesi *"orta"* ve "*düşük*" seviyedir. Aynı zamanda katılımcıların büyük çoğunluğunun (N:26; %87) programlamayı "*orta zorlukta*", "*zor*" ve "çok *zor*" olarak nitelendirdiği görülmektedir (Şekil 3.20). Tüm bu bilgilere ilave olarak birinci döngü uygulama aşaması katılımcılarının programlama ve oyun geliştirme tecrübeleri Şekil 3.21 ve 3.22'de sunulmuştur.



Şekil 3.21. Katılımcıların ortaöğretimde programlama dersi alma durumları

Şekil 3.22. Katılımcıların oyun geliştirme tecrübeleri

Şekil 3.21'e göre katılımcıların çoğunun (N:25; %83) ortaöğretimde programlama dersi almadıkları görülmektedir. Benzer şekilde iki katılımcı hariç olmak üzere katılımcıların daha önce oyun geliştirme deneyiminin olmadığı söylenebilir (Şekil 3.22). Bu tez çalışması boyunca araştırmanın ikinci döngüsüne katılan öğrencilerinin isimleri D2_K1, D2_K2, D2_K3,…,D2_K30 şekilde kodlanmıştır.

3.3. Veri Toplama Araçları

Bu tez çalışmasında "veri çeşitlemesi (data triangulation)" ve farklı veri kaynaklarından yararlanmak maksadıyla birçok nitel ve nicel veri toplama aracı kullanılmıştır. Bu araçlardan faydalanılırken Padak ve Padak'ın (1994) "*eylem araştırmalarında soruların yanıtını etkin bir şekilde bulabilmek için kullanılabilecek her bir unsur, bilgi kaynağıdır."* prensibi temel alınmıştır. Araştırmanın her iki döngüsünün araştırma sorusuna yönelik olarak kullanılan veri toplama araçları Tablo 3.8'de sunulmuştur.

Tablo 3.8. Araştırma kapsamında kullanılan veri toplama araçları

Veri Toplama Aracı	**Araştırma Sorusu**	
	Birinci Döngü	**İkinci Döngü**
Literatür taraması	1.5., 1.6.	2.1
Katılımcı Bilgi Formu	1.8.	2.3.
NYP Başarı Sınavı	1.8.	2.3.
Proje Değerlendirme Rubriği	1.8.	2.3.
Dijital Öğrenci Günlükleri	1.7., 1.9.	2.2., 2.4., 2.5.
Araştırmacı Günlüğü	1.7., 1.8., 1.9.	2.2., 2.4., 2.5.
Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği	1.8., 1.9.	2.3., 2.4., 2.5.
Programlama Öz Yeterlilik Ölçeği	1.8., 1.9.	2.3., 2.4., 2.5.
Programlama Kaygı Ölçeği	1.8., 1.9.	2.3., 2.4., 2.5.
Odak Grup Görüşmesi	1.1., 1.2., 1.3., 1.4., 1.6, 1.7., 1.8., 1.9.	2.1., 2.2., 2.3., 2.4., 2.5.
Analiz Aşaması Katılımcı Anketi	1.1., 1.2., 1.3., 1.4., 1.6.	-
Analiz Aşaması Öğretim Elemanı Görüşme Formu	1.2., 1.6.	-
Unity 3D Başarı Sınavı	-	2.3.
WhatsApp Yazışmaları		2.2., 2.5.

3.3.1. Katılımcı Bilgi Formu

Katılımcı bilgi formu, eylem araştırmasının uygulama aşamasına katılan öğrencileri daha iyi tanımak amacıyla oluşturulmuştur. Hazırlanan bu ankette; katılımcıların yaş aralıklarının, mezun oldukları ortaöğretim kurumlarının, bilgisayar sahibi olup olmama durumlarının, günlük olarak bilgisayar kullanma ve dijital oyun oynama sıklıklarının, daha önce programlama alıp almama durumlarının, algıladıkları programlama bilgi ve zorluk seviyelerinin, kullanmayı bildikleri tasarım, geliştirme veya kodlama araçlarının, bildikleri programlama dillerinin, programlama ve oyun programlama hakkındaki düşüncelerinin belirlenmesine yönelik sorulara yer verilmiştir. Katılımcı Bilgi Formu Ek 1'de sunulmuştur.

3.3.2. NYP Başarı Sınavı

Mevcut NYP dersinin en önemli hedeflerinden birisi, öğrencilerin kendilerine verilen programlama problemlerini nesne yönelimli programlama paradigmasını kullanarak çözmeleridir. Öğrenciler bu derste hem nesne yönelimli programlama kavramlarını öğrenirken hem de nesne yönelimli programlama tekniğini uygulamaktadır. Öğrencilerin bu bilgi ve becerileri kazanıp kazanmadığını değerlendirmek için bu tez çalışması kapsamında araştırmacı tarafından geliştirilen bir akademik başarı sınavı kullanılmıştır. NYP Başarı Sınavının geliştirilmesi sürecinin ilk basamağında NYP eğitiminin kazanımlar listesinde yer alan 22 kazanıma uygun olarak teorik ve uygulama sorularını içeren 23 soruluk bir başarı sınavı hazırlanmıştır. NYP dersine ait kazanımlar Tablo 3.9'da verilmiştir. Soruların hazırlanması sürecinde madde belirtke tablosu oluşturularak her bir kazanımı ölçebilmesi sağlanmıştır. NYP Başarı Sınavına ait madde belirtke tablosu Ek 2'de sunulmuştur.

Tablo 3.9. NYP dersi kazanımları

Kazanım Kodu	Kazanımlar
K1	Sınıf kavramını örneklerle açıklar.
K2	Nesne kavramını örneklerle açıklar.
K3	Özellik (attribute) kavramını örneklerle açıklar.
K4	Davranış (behaviour) kavramını örneklerle açıklar.
K5	Kurucu metot kavramını örneklerle açıklar.
K6	Sınıf ile nesne arasındaki ilişkiyi betimler.
K7	Kalıtımın (inheritance) kullanım amaçlarını örnek vererek açıklar.
K8	Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar.
K9	Kapsüllemenin (encapsulation) kullanım amaçlarını örnek vererek açıklar.
K10	Statik sınıflar ile normal sınıfları karşılaştırır.
K11	C# programlama dilinde kullanılan erişim belirleyicilerin kullanım amaçlarını belirtir.
K12	Verilen UML Sınıf diyagramlarında yer alan sınıflara ait özellik ve davranışlarını çözümleyerek bu sınıflar arasındaki ilişkileri açıklar.
K13	Verilen bir C# sınıfını analiz ederek bu sınıfta kullanılan NYP prensiplerini (kalıtım, kapsülleme, çokbiçimlilik vb.) çözümler.
K14	Bir sınıfa ait özelliklerin erişim belirleyicilerini verilen koşula göre değiştirir.
K15	Önceden oluşturulmuş bir C# sınıfını, belirtilen oyun nesnesine bileşen (component) olarak ekler.
K16	UML sınıf diyagramı verilen arayüzü (interface), C# programlama dilinde oluşturur.

(Devam ediyor)

Tablo 3.9. NYP dersi kazanımları (Devam)

Kazanım Kodu	Kazanımlar
K17	UML sınıf diyagramında belirtilen çokbiçimli metotları oluşturur ve kullanır.
K18	Verilen koşullara göre kalıtım mekanizmasını C# programlama dilinde gerçekleştirir.
K19	UML sınıf diyagramı verilen sınıfı C# programlama dilinde oluşturur.
K20	Önceden oluşturulmuş bir C# sınıfına ait özelliklerin değer atamasını verilen koşula göre yapar.
K21	Bir C# sınıfına ait Monobehavior metotlarını, verilen koşula göre oluşturur.
K22	Kullanıcı arayüzü (UI) nesnelerini, oluşturduğu C# sınıfında kullanır.

NYP Başarı Sınavının geliştirilmesi sürecinin ikinci basamağında sınavın kapsam geçerliliği çalışmaları yürütülmüştür. Bu çalışmaların gerçekleştirilmesinde Davis'in (1992) işlem basamakları esas alınmıştır. Bu kapsamda, hazırlanan başarı sınavı NYP Başarı Sınavı Kapsam Geçerliliği Uzman Görüş Formu (Ek 3) kullanılarak Tablo 3.5'te demografik özellikleri açıklanan BÖTE alanında uzman altı öğretim elemanı tarafından değerlendirilmiştir. Başarı sınavında yer alan sorular "*Uygun, Küçük düzeltmeler yapılmalı, Büyük düzeltmeler yapılmalı ve Uygun değil"* kriterlerine göre değerlendirilmiştir. Uzmanlardan elde edilen geri bildirimler doğrultusunda uygun ve küçük düzeltmeler yapılmalı maddeleri 1 puan; uygun değil ve büyük düzeltmeler yapılmalı maddeleri 0 puan olarak kodlanarak her soru için madde geçerlilik indeksi hesaplanmıştır. Bununla birlikte tüm sınav sorularının indeks puanlarının ortalaması alınarak NYP Başarı Sınavına ait kapsam geçerlilik indeksi belirlenmiştir. Bu kapsamda Bölüm-2 Md.f, Bölüm-2 Md.g ve Bölüm-3 Soru-4 kodlu soruların madde geçerlilik indeks değeri 0.83 olarak hesaplanırken diğer soruların tamamında indeks değeri 1.00 olarak hesaplanmıştır. Sınavın tamamı için kapsam geçerlilik indeksi ise 0.98 olarak belirlenmiştir. Bu sonuçlar başarı sınavının kapsam geçerliğinin sağlandığını göstermektedir. Birinci ve ikinci döngü sonunda NYP Başarı Sınavına ait Cronbach Alfa (α) iç tutarlık katsayıları hesaplanarak güvenirlik analizleri yapılmıştır. Analiz sonuçları tezin 3.7. Geçerlik ve Güvenirlik alt başlığında sunulmuştur. NYP başarı sınavı örneği Ek 4'te sunulmuştur.

3.3.3. Proje Değerlendirme Rubriği

Bu tez çalışması kapsamında öğrencilerin hazırladıkları oyun projeleri, öğrencilerin uygulama sürecindeki ilerlemelerini ya da değişimlerini takip edebilmek açısından önemli bir veri kaynağı olmuştur (Hubbard ve Power, 2003). Öğrenci projelerini objektif bir şekilde değerlendirebilmek için araştırmacı tarafından projenin başarı ölçütlerini listeleyen bir proje değerlendirme rubriği (dereceli puanlama anahtarı) geliştirilmiştir. Öğrencilerin

geliştirdikleri oyun prototipleri bu rubrik kullanılarak araştırmacı tarafından değerlendirilmiştir. Kısaca bahsetmek gerekirse Proje Değerlendirme Rubriği on kategoriden oluşmaktadır. Her bir kategori kapsamında "düşük seviye" için bir puan, "orta seviye" için beş puan, "iyi seviye" için sekiz puan ve "ileri seviye" için 10 puan verilmektedir. Dolayısıyla bir öğrencinin proje değerlendirme puanı 10 ile 100 puan arasında değişmektedir.

Bu ölçme aracının geliştirilmesi sürecinde Goodrich'in (2001) belirttiği adımlar takip edilerek (1) proje değerlendirme rubriğinin amacı ve bu amaca bağlı olarak türü belirlenmiş, (2) proje değerlendirme rubriğinin ölçüt ve alt ölçütleri belirlenmiş, (3) belirlenen ölçüt ve alt ölçütlere dönük davranış (performans) düzeyleri tanımlanmış ve (4) hazırlanan proje değerlendirme rubriği uzman görüşleri doğrultusunda gözden geçirilmiş ve taslak rubriğe son şekli verilmiştir. Proje değerlendirme rubriğinin kapsam geçerliliğini belirlemek amacıyla Proje Değerlendirme Rubriği Kapsam Geçerliliği Uzman Görüş Formu (Ek 5) hazırlanmış ve Tablo 3.5 Tablo 3.5'te özellikleri verilen BÖTE alanında uzman dört öğretim elemanı tarafından incelenmiştir. Uzmanlar Proje Değerlendirme Rubriğinde yer alan ölçütleri "*Uygun Değil, Kısmen Uygun ve Uygun"* kriterlerine göre değerlendirilmiştir. Görüşüne başvurulan uzmanların yanıtları ve toplam uzman görüşü puanları Tablo 3.10'da sunulmuştur.

Tablo 3.10. Proje Değerlendirme Rubriğinin geçerliliğine ilişkin uzman görüşleri

Ölçüt No	**1. Uzman**			**2. Uzman**			**3. Uzman**			**4. Uzman**			**Toplam**		
	UD	**KU**	**U**	**UD**	**KU**	**U**	**UD**	**KU**	**U**	**UD**	**KU**	**U**	**UD**	**KU**	**U**
Ölçüt 1			+			+			+			+			4
Ölçüt 2			+			+			+			+			4
Ölçüt 3			+			+			+			+			4
Ölçüt 4			+			+			+			+			4
Ölçüt 5			+			+			+			+			4
Ölçüt 6			+			+			+			+			4
Ölçüt 7			+			+			+			+			4
Ölçüt 8			+		+				+			+		1	3
Ölçüt 9			+			+			+		+			1	3
Ölçüt 10			+			+			+			+			4

UD: *Uygun Değil* ***KU:*** *Kısmen Uygun* ***U:*** *Uygun*

Proje Değerlendirme Rubriğinin kapsam geçerliliği çalışmaları Tablo 3.10'da sunulan geri bildirimler doğrultusunda Davis'in (1992) işlem basamakları esas alınarak

yürütülmüştür. Uzmanlardan elde edilen uygun ve kısmen uygun maddeleri 1 puan, uygun değil maddeleri 0 puan olarak kodlanarak her bir ölçüt için geçerlilik indeksi hesaplanmıştır. Bununla birlikte tüm ölçütlere ait indeks puanlarının ortalaması alınarak Proje Değerlendirme Rubriğine ait kapsam geçerlilik indeksi belirlenmiştir. Bu kapsamda rubrikte yer alan tüm ölçütlerin indeks değeri 1.00 olarak hesaplanmış olup rubriğin tamamı için kapsam geçerlilik indeksi ise 1.00 olarak belirlenmiştir. Bu hesaplamalar doğrultusunda rubriğin daha güvenilir ve yansız değerlendirme yapabilmesi (Kutlu, Doğan ve Karakaya, 2014) sağlanmaya çalışılmıştır. Geliştirilen Proje Değerlendirme Rubriği örneği Ek 6'da sunulmuştur.

3.3.4. Dijital Öğrenci Günlükleri

Alanyazında yer alan diğer eylem araştırmalarında olduğu gibi öğrenci günlükleri bu tez çalışmasında da araştırma sorularını yanıtlayabilmek için sıklıkla başvurulan bir veri toplama aracı olmuştur. Öğrenci günlükleri sayesinde eylem araştırması sürecine katılan bireylerin duygu, düşünce ve deneyimleri ile ilgili detaylı bilgiler edinilmiştir (Mertler, 2009). Araştırmacı, her dersin bitiminde katılımcılardan NYP dersi ile ilgili duygu ve düşüncelerini dijital bir günlüğe yazmalarını istemiştir. Günlük verileri nitel verilerinin analizi ve değerlendirilmesi (Creswell, 2003) sürecine uygun olarak incelenmiş ve çözümlemiştir. Öğrenciler dijital günlüklerini yazarken aşağıdaki konu başlıklarını esas almışlardır:

- Bu derste neyi öğrendim?
- Bu derste neyi başardım?
- Bu derste neyi tam olarak anlayamadım?

Öğrenci günlükleri sayesinde katılımcılar daha yakından tanınmış; katılımcılara ait duygu ve düşünceler daha derinlemesine öğrenilmiş; katılımcıların kendilerini ifade etmelerini sağlanmış ve derslerde (yüz yüze ve canlı dersle) araştırmacının gözden kaçırdığı noktaların açığa çıkarılabilmesi olanaklı hale gelmiştir.

3.3.5. Araştırmacı Günlüğü

Araştırmacı günlüğü, eylem araştırması sürecinde araştırmacı tarafından tutulan notlar olarak tanımlanabilir. Bu tez çalışması kapsamında kullanılan araştırmacı günlüğünde; araştırmacının gözlemleri, kısa notları, görüşleri, izlenim ve fikirleri yer almıştır (Johnson, 2014). Bu noktadan hareketle araştırmacının tüm hazırlıkları, ders etkinlikleri, süreçte

karşılaştığı sorunlar ve denenen çözüm yolları, eylem araştırması sürecine ilişkin yansıtmaları bu günlükte yer almıştır.

3.3.6. Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği

Law, Lee ve Yu (2010) tarafından geliştirilen Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği (BPDÖMÖ), bilgisayar programlama derslerini alan üniversite öğrencilerinin derse yönelik motivasyonlarını ölçmek amacıyla geliştirilmiştir. Orijinal dili İngilizce olan ölçek, Avcı ve Ersoy (2018) tarafından Türkçe'ye uyarlanmıştır. Toplam 19 maddeden oluşan ölçek altılı likert tipinde geliştirilmiştir. Ölçeğe ait bilgiler Tablo 3.11'de sunulmuştur.

Tablo 3.11. Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeğine ait bilgiler

Alt Faktör	Madde Sayısı	İç Tutarlık Katsayısı (α)	Örnek Madde
Tutum ve Beklenti	4	.82	"Program yazmadaki yüksek performansın yüksek not almamda bana yararlı olacağını bilirsem programlamada daha başarılı olurum."
Zorlayıcı Amaçlar	3	.83	"Daha önce hiç karşılaşmadığım bir problemi çözmek için bir program yazmak gerektiğinde, programı tamamlamaya ve bu süreçte yeni öğrenme deneyimleri elde etmeye motive olurum."
Belirgin Hedefler	3	.71	"Programlama çalışmasının amacı hakkında net olduğumda daha iyi yapmaya motive olurum."
Ödül ve Takdir	3	.78	"Yüksek performansım diğerleri (sınıf arkadaşlarım ve öğretmenim) tarafından takdir edilirse, programlama yapma performansım daha da yükselir."
Ceza	2	.71	"Programlarımda hata yaptığımda uygun ceza (örn. not azaltma) verilirse, programlama öğrenmeye daha fazla motive olurum."
Sosyal Baskı ve Rekabet	4	.82	"Sınıf arkadaşlarım ile rekabet etmek beni programlamada daha başarılı olmaya iter."
			Cronbach Alfa Katsayısı (α)=.90

Tablo 3.11'e göre ölçek altı alt faktörden oluşmaktadır. Avcı ve Ersoy'un (2018) gerçekleştirdiği uyarlama çalışmasında ölçek 312 mühendislik fakültesi öğrencisine uygulanmış ve doğrulayıcı faktör analizi yapılmıştır. Avcı ve Ersoy'un (2018) gerçekleştirdiği analiz sonuçlarına göre alt faktörlerin iç tutarlık katsayıları .71 ile .83 arasında olup ölçeğin tamamı için iç tutarlılık katsayısı .90'dır. Bu tez çalışmasının birinci ve ikinci döngülerinde uygulanan ölçeğin alt faktörlerine ve geneline ait Cronbach Alfa (α) iç tutarlık katsayıları hesaplanarak güvenirlik analizleri yapılmıştır. Analiz sonuçları tezin 3.7. Geçerlik ve Güvenirlik alt başlığında sunulmuştur.

3.3.7. Programlama Öz Yeterlilik Ölçeği

Kukul, Gökçearslan ve Günbatar (2017) tarafından tasarlanan ve orijinal adı "*Ortaokul Öğrencileri İçin Programlama Öz Yeterlilik Ölçeği*" olan ölçek 31 maddeden oluşmaktadır. Ölçek, tek bir boyut içerecek şekilde beşli likert tipinde geliştirilmiştir. Ölçeğe ait bilgiler Tablo 3.12'de sunulmuştur.

Tablo 3.12. Programlama Öz Yeterlilik Ölçeğine ait bilgiler

Alt Faktör	Toplam Madde Sayısı	Örnek Maddeler
Programlama Öz Yeterliliği	31	"Programlama problemlerini çözmek için bana verilen çözüm yollarından farklı çözüm yolları önerebilirim." "Karmaşık programlama problemlerini, küçük alt problemlere ayırarak çözebilirim." "Programlama problemini çözmek için tasarladığım işlemlerin sırasını gerektiğinde değiştirebilirim."
		Cronbach Alfa Katsayısı (α)=.95

Orijinal ölçeğin çalışma grubunu Ankara'daki bir devlet ortaokulunda öğrenim gören 233 öğrenci oluşturmuştur. Bu nedenle bu tez çalışması kapsamında kullanılan ölçeğin üniversite öğrencileri için de geçerli ve güvenilir olup olmadığı incelenmiştir. Orijinal ölçeğin uyarlanması kapsamında ilk olarak ölçek maddelerinin dili incelenmiştir. Bu süreçte ölçek maddeleri beş ön lisans öğrencisine tek tek okutulmuş ve öğrencilerden okudukları maddelerden ne anladıkları sorulmuştur. Öğrencilerin ölçekteki maddeleri anlamakta herhangi bir güçlük yaşamadıkları tespit edilmiştir. Daha sonra ölçek, bir devlet meslek yüksekokulunda Programlamaya Giriş dersini almakta olan 343 üniversite öğrencisine çevrimiçi olarak uygulanmıştır. Toplanan verilerin faktör analizine uygun olup olmaması durumu Barlett'in Küresellik değeri ve KMO (Kaiser-Mayer-Olkin) katsayısının hesaplanması ile test edilmiştir. Yapılan analizin sonucunda Barlett normal dağılım testinin anlamlı (p=.000) olduğu tespit edilmiş; bunun yanında KMO katsayısı .964 olarak hesaplanmıştır. Bu sonuçlar, ölçeğin faktör analizi için uygun olduğunu göstermiştir. Bu aşamadan sonra elde edilen veriler, ölçeğin yapı geçerliliğinin incelenmesi amacıyla Açımlayıcı Faktör Analizine (AFA) tabi tutulmuştur. Faktör analizi sonucunda elde edilen madde faktör yükleri ve madde toplam korelasyon katsayıları Tablo 3.13'te sunulmuştur.

Tablo 3.13. Programlama Öz Yeterlilik Ölçeği madde faktör yükleri ve madde toplam korelasyonu sonuçları

Madde No	Madde Faktör Yükü	Madde Toplam Korelasyonu
Madde 1	.544	.595
Madde 2	.576	.667
Madde 3	.540	.685
Madde 4	.714	.617
Madde 5	.687	.663
Madde 6	.780	.646
Madde 7	.747	.614
Madde 8	.675	.679
Madde 9	.570	.722
Madde 10	.532	.704
Madde 11	.604	.652
Madde 12	.491	.588
Madde 13	.521	.785
Madde 14	.609	.767
Madde 15	.663	.765
Madde 16	.606	.776
Madde 17	.611	.772
Madde 18	.526	.700
Madde 19	.751	.525
Madde 20	.513	.747
Madde 21	.592	.740
Madde 22	.644	.724
Madde 23	.795	.712
Madde 24	.792	.733
Madde 25	.575	.775
Madde 26	.604	.740
Madde 27	.702	.565
Madde 28	.739	.483
Madde 29	.790	.621
Madde 30	.745	.713
Madde 31	.679	.701
	Cronbach Alfa Katsayısı (α)=.97	

Tablo 3.13'e göre Programlama Öz Yeterlilik Ölçeğinin her bir maddesine ait madde faktör yükleri .491 ile .795 arasında değişmektedir. Bununla birlikte 31 madde için madde toplam korelasyonu .483 ile .785 arasında değişmektedir. Ölçeğin güvenilirliği için

hesaplanan Cronbach Alfa iç tutarlılık katsayısı .97 olarak hesaplanmıştır. Yapılan analizlerin sonuçları ölçeğin test edilen yapısının Kukul ve diğerlerinin (2017) yapısıyla uyumlu olduğunu; aynı zamanda güvenilir bir ölçme aracı olduğunu göstermektedir (Clark ve Watson, 2016).

AFA'dan sonra ölçeğin psikometrik özelliklerini değerlendirmek amacıyla Doğrulayıcı Faktör Analizi (DFA) yapılmış; bu analiz doğrultusunda ölçeğe ait uyum indeksleri incelenmiştir. Ölçeğe ait hesaplanan uyum indeksleri Tablo 3.14'te sunulmuştur. Hiçbir sınırlama yapılmadan gerçekleştirilen DFA sonucunda ölçeğin iç tutarlığının yeterince yüksek olduğu, bir başka ifadeyle ölçeğin güvenilir ölçümler yapabildiği ortaya çıkmıştır (Kline, 2005). Tüm bu çalışmalara ilave olarak tez çalışmasının birinci ve ikinci döngülerinde uygulanan ölçeğin geneline ait Cronbach Alfa (α) iç tutarlık katsayıları hesaplanarak güvenirlik analizleri yapılmıştır. Analiz sonuçları tezin 3.7. Geçerlik ve Güvenirlik alt başlığında sunulmuştur.

Tablo 3.14. Programlama Öz Yeterliliğine ait hesaplanan uyum indeks değerleri

X^2	X^2/df	p	RMSEA	GFI	NFI	CFI	AGFI	IFI
888.514*	2.205	.000	0.059	.84	.90	.94	.81	.94

* $p<0.01$

3.3.8. Programlama Kaygı Ölçeği

Programlamaya Yönelik Kaygı Ölçeği (PKÖ), öğrencilerin bilgisayar programlama derslerindeki kaygılarını ölçmek amacıyla araştırmacı ve doktora tez danışmanı tarafından geliştirilmiştir. Açımlayıcı faktör analizine (AFA) 392, doğrulayıcı faktör analizine (DFA) 295 lisans ve ön lisans öğrencisinin katılımıyla geliştirilen ölçek, iki faktör ve 11 maddeden oluşmaktadır. Programlama Kaygı Ölçeğine ait bilgiler Tablo 3.15'te sunulmuştur.

Tablo 3.15. Programlama Kaygı Ölçeğine ait bilgiler

Alt Faktör	Madde Sayısı	İç Tutarlık Katsayısı (α)	Örnek Madde
Akran Kaygısı	5	.843	"Daha önce programlama dersi alan arkadaşlarımın seviyesine yetişemeyeceğime inanmak beni kaygılandırır."
Programlama Beceri Kaygısı	6	.882	"Programlarımda hatalarla karşılaşmak benim için büyük bir endişe kaynağıdır."
			Cronbach Alfa Katsayısı (α)=.901

Tablo 3.15'e göre "*akran kaygısı*" faktörü 5 maddeden, "p*rogramlama beceri kaygısı*" faktörü 6 maddeden oluşmaktadır. Beşli likert tipinde geliştirilen ölçekte, her bir madde "Hiçbir zaman" seçeneği için 1 puan, "Her zaman" seçeneği için ise 5 puan olarak değerlendirilmektedir. Dolayısıyla bir katılımcının PKÖ'den alacağı minimum puan 11, maksimum puan 55 olarak hesaplanmaktadır. Öğrencilerin programlamaya yönelik kaygıları; "*ölçekten alınan toplam puan arttıkça programlama kaygısı da artar"* prensibiyle belirlenmektedir. Ölçeğin geliştirilme sürecinin detayları aşağıda açıklanmıştır.

Programlamaya Yönelik Kaygı Ölçeğinin geliştirilmesi sürecinin ilk adımında araştırmacılar tarafından detaylı bir alanyazın taraması gerçekleştirilmiş; *test kaygısı, bilgisayar kaygısı, matematik kaygısı, yabancı dil öğrenme kaygısı* gibi kaygı ölçekleri incelenmiştir. Alanyazın incelemesinden sonra 33 maddeden oluşan madde havuzu oluşturulmuştur. Taslak ölçek formunun *kapsam geçerliliğinin (content validity)* sağlanması amacıyla BÖTE alanında üç, Türk Dili alanında bir ve Psikolojik Danışmanlık ve Rehberlik alanında bir olmak üzere beş uzmandan görüş alınmıştır. Uzmanların değerlendirmeleri doğrultusunda ölçek maddeleri düzenlenmiştir.

Taslak ölçeğin *görünüş geçerliği (face validity)* sağlamak ve ölçek maddelerinin katılımcılar tarafından anlaşılırlığının belirlenmesi amacıyla araştırma örneklemine uygun dokuz katılımcıyla yüz yüze bilişsel görüşmeler yapılmıştır. Her bir madde anlam ve ifade açısından incelenmiştir. Katılımcıların da görüşleri alınarak 24 maddelik taslak Programlamaya Yönelik Kaygı Ölçeğine son hali verilmiştir. Bu işlemlerin ardından ölçeğin geçerlik ve güvenirlik analizleri gerçekleştirilmiştir.

Bu aşamada 24 maddeden oluşan taslak ölçek programlama dersini almakta olan lisans ve ön lisan öğrencileriyle çevrimiçi olarak paylaşılmıştır. Araştırmaya katılan 392 katılımcıdan gelen yanıtlar doğrultusunda taslak ölçeğin geçerlik ve güvenirlik çalışmaları yapılmıştır. Taslak ölçeğin yapı geçerliliğinin belirlenmesi amacıyla AFA gerçekleştirilmiştir. AFA'da örneklemin yeterli olup olmadığı test etmek için Kaiser-Meyer-Olkin (KMO) değeri; verilerin faktör analizine uygun olup olmadığını sınamak için ise Barlett'in Küresellik değeri kullanılmıştır. KMO değeri .951 olarak bulunmuştur. Barlett testi sonucunda anlamlı farklılık tespit edilmiştir (x^2=3790.593, p=.000). Elde edilen sonuçlara göre mevcut veriler ile AFA yapılmasının uygun olduğu görülmüştür (Tabachnick ve Fidell, 2007). AFA'da ölçeğin alt boyutlarının ilişkili olduğu değerlendirilerek Temel Eksenler Metodu (Principal Axis Factoring) ile oblik döndürme (direct oblimin) tekniği kullanılmıştır (Gorsuch, 1983).

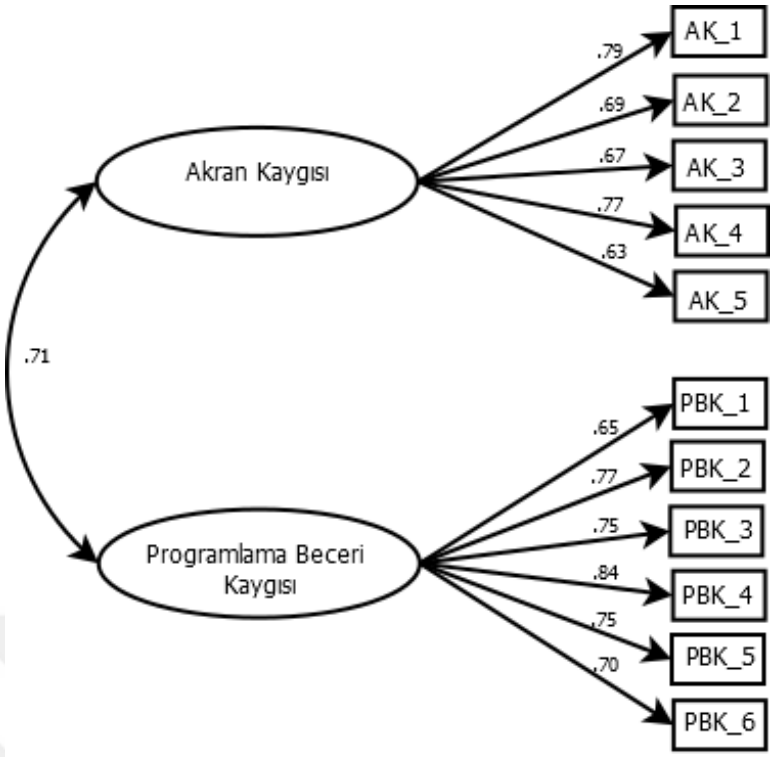
AFA sonucunda ölçekten madde çıkarma sürecinde, maddeler için madde toplam korelasyonu, iç tutarlılık katsayısı, ortak varyans ve alt-üst grup farkları dikkate alınarak işlemler yapılmıştır. Her bir madde çıkarma işleminin ardından madde toplam korelasyonları yeniden hesaplanmış ve ölçeğin faktör yapısı araştırılmıştır. Faktör analizi sonucunda uzmanların fikirleri de dikkate alınarak 10 madde ölçekten çıkarılmıştır. Uygun olmayan on ölçek maddesi silindikten sonra 14 madde ve iki faktörden oluşan Programlamaya Yönelik Kaygı Ölçeği oluşturulmuştur. Programlamaya Yönelik Kaygı Ölçeğinin toplam varyansın %69.4'ünü açıkladığı tespit edilmiştir. Ölçek maddelerine ait faktör yükleri Tablo 3.16'da sunulmuştur.

Tablo 3.16. Programlama Kaygı Ölçeği madde faktör yükleri

Madde No	Akran Kaygısı	Programlama Beceri Kaygısı
AK_1	.633	
AK_2	.806	
AK_3	.918	
AK_4	.643	
AK_5	.634	
AK_6	.877	
PBK_1		.818
PBK_2		.726
PBK_3		.851
PBK_4		.844
PBK_5		.909
PBK_6		.780
PBK_7		.852
PBK_8		.822

AK: *Akran Kaygısı*
PBK: *Programlama Beceri Kaygısı*

AFA'dan sonra ölçeğin psikometrik özelliklerini değerlendirmek amacıyla DFA yapılmış; bu analiz doğrultusunda ölçeğe ait uyum indeksleri incelenmiştir. DFA'da test edilen iki faktörlü ölçek yapısına ait standardize edilmiş sonuçlar Şekil 3.23'te sunulmuştur. Bunun yanında ölçeğe ait hesaplanan uyum indeksleri Tablo 3.17'de sunulmuştur.



Şekil 3.23. Standardize edilmiş sonuçlar ile birinci-düzey DFA

Tablo 3.17. Programlama Kaygı Ölçeğine ait hesaplanan uyum indeks değerleri

X^2	X^2/df	p	RMSEA	SRMR	GFI	NFI	CFI
114.226*	2.79	.000	.071	0.032	.93	.93	.95

* $p<0.01$

Tablo 3.17'de verilen uyum indeks değerlerine göre ölçeğin yapısal modeli iyi bir uyum göstermektedir. DFA sonucunda Şekil 3.23'de verilen modelde ölçeğin her bir maddesinin kendi faktörüne önemli ölçüde yüklendiği ve nispeten büyük olduğu (.50 ve üzeri) belirlenmiştir. DFA'dan sonra ölçeğin geçerlilik ve güvenirlik analizleri gerçekleştirilmiştir. Ölçeğin yakınsama (convergent) geçerliliğini ölçmek amacıyla standardize edilmiş faktör yükleri, faktörlere ait average variance extracted (AVE) ve composite reliability (CR) değerleri hesaplanmıştır. Yakınsama geçerlilik kriteri olarak standardize edilmiş faktör yüklerinin .5; AVE ve CR değerlerinin için sırasıyla .5 and .7'ten yüksek olması göz önünde bulundurulmuştur (Hair, Black, Babin ve Anderson, 2010). Ölçeğin güvenirlik kriteri olarak Cronbach alfa değerlerinin .70'den yüksek olması gözetilmiştir (Taber, 2018). Ölçeğe ait yakınsama geçerliliği ve güvenirlik analizi sonuçları Tablo 3.18'de verilmiştir. Tabloya göre ölçeğin yukarıda belirtilen kriterleri sağladığı görülmektedir. Ölçeği geneli için Cronbach

alfa değerinin .901 olarak hesaplanması da ölçeğin güvenirlik ölçütlerini sağladığını göstermektedir.

Tablo 3.18. Ölçeğin yakınsama geçerliliği ve güvenirlik analizi sonuçları

Faktörler	CR	AVE	Cronbach Alfa
Akran Kaygısı	.835	.506	.843
Programlama Beceri Kaygısı	.881	.554	.882

Ölçeğin ayırt edici (discriminant) geçerliliği Fornell & Larcker kriteri ve Heterotrait-Monotrait (HTMT) korelasyon oranı hesaplanarak ölçülmüştür (Ab Hamid, Sami ve Sidek, 2017). Tablo 3.19'da Fornell & Larcker kriteri için her bir faktöre ait AVE değerinin karekökü hesaplanarak koyu şekilde yazılmış; ilgili satır ve sütunda çapraz (köşegen) olarak verilmiştir. Bunun yanında faktörlere ait korelasyon sonucu da sunulmuştur. Fornell & Larcker kriteri bağlamında faktörlere ait AVE değerlerinin karekökünün korelasyondan büyük olması dikkate alınmıştır. Tablo 3.19'da bu şartın sağlandığı görülmektedir. İlave olarak DFA'da test edilen modelin HTMT katsayısı .705 olarak hesaplanmıştır. Bu değerin .90'dan düşük olması, modelin ayırt edici geçerliliğe sahip olduğunu göstermektedir (Henseler, Ringle ve Sarstedt, 2015). Ölçeğe ait gerçekleştirilen tüm bu geçerlilik ve güvenirlik analizi sonuçları, iki faktör ve 11 maddeden oluşan Programlama Kaygısı Ölçeğinin güvenilir ölçümler yapabildiği göstermektedir (Kline, 2005).

Tablo 3.19. Ölçeğin ayrıt edici geçerlilik analizi sonuçları

Faktörler	Faktörler	
	Akran Kaygısı	Programlama Beceri Kaygısı
Akran Kaygısı	.711	
Programlama Beceri Kaygısı	.705	.745

Tüm bu çalışmalara ilave olarak tez çalışmasının birinci ve ikinci döngülerinde uygulanan ölçeğin alt faktörlerine ve geneline ait Cronbach Alfa (α) iç tutarlık katsayıları hesaplanarak güvenirlik analizleri yapılmıştır. Analiz sonuçları tezin 3.7. Geçerlik ve Güvenirlik alt başlığında sunulmuştur. Programlamaya Yönelik Kaygı Ölçeği örneği Ek 7'de sunulmuştur.

3.3.9. Odak Grup Görüşmesi

Bu tez çalışması kapsamında kullanılan odak grup görüşmeleri, Krueger'in (1994) deyimine uygun olarak "*tanımlanmış bir ilgi alanı hakkında algılar elde etmek için tehdit*

edici olmayacak şekilde tasarlanmış bir tartışma" olarak planlanmış ve uygulanmıştır. Bu sayede katılımcıların her iki döngüde de eylem planının uygulanmasıyla ilgili deneyimleri ve görüşleri elde edilmiş, aynı zamanda katılımcıların bu uygulamalarla ilgili bilişsel ve duygusal tepkileri araştırılmıştır (Vaughn, Schumm ve Sinagub, 1996). Katılımcılara bu görüşmelerde "*NYP dersinde uyguladığımız "programlamayı oyun programlayarak öğrenmenin" sevdiğiniz ve sevmediğiniz yönlerini nelerdir?"*,*"Derslerde hangi konularda zorluk yaşadınız?", "Derste kullandığımız bu yöntem programlama konusundaki kendinize güveninizi nasıl etkiledi?"* gibi sorular yöneltilmiştir.

3.3.10. Analiz Aşaması Katılımcı Anketi

Analiz Aşaması NYP Dersi Katılımcı Anketi, mevcut NYP dersinde öğrencilerin yaşadığı sorunların; bunun yanında öğrencilerin dersten beklentilerinin tespit edilebilmesi maksadıyla oluşturulmuştur. Hazırlanan bu ankette; katılımcıların cinsiyetlerinin, yaş aralıklarının, bilgisayar sahibi olup olmama durumlarının, günlük bilgisayar kullanma sürelerinin, dijital oyunları oynama sıklıklarının, daha önce programlama alıp almama durumlarının, algıladıkları programlama bilgi ve zorluk seviyelerinin, NYP dersinde yaşadıkları sorunların ve bu sorunların çözümüne yönelik önerilerinin belirlenmesine yönelik sorulara yer verilmiştir. Bunun yanında bu anket vasıtasıyla katılımcıların mevcut NYP dersinde geliştirilen yazılım projeleri hakkındaki görüşleri ve derste geliştirmeyi tercih ettikleri yazılım projesi türleri belirlenmiştir. Analiz Aşaması NYP Dersi Katılımcı Anketi Ek 8'de sunulmuştur.

3.3.11. Analiz Aşaması Öğretim Elemanı Görüşme Formu

NYP Dersi Analiz Aşaması Öğretim Elemanı Yarı Yapılandırılmış Görüşme Formu, mevcut NYP dersinde öğretim elemanlarının yaşadıkları sorunların tespit edilebilmesi; bunun yanında dersin daha etkili ve verimli hale getirilebilmesi için önerilerinin elde edilebilmesi maksadıyla oluşturulmuştur. Hazırlanan bu yarı yapılandırılmış görüşme formunda; öğrencilerin NYP derslerinde başarılarını etkileyen faktörlerin, öğretim elemanlarının NYP dersinde yaşadıkları problemlerin, NYP dersini daha ilgi çekici yapabilmek için başvurulabilecek stratejilerin, mevcut NYP dersine oyun geliştirmeye dayalı öğretim stratejisinin nasıl entegre edilebileceğinin belirlenmesine yönelik sorulara yer verilmiştir. Form örneği Ek 9'da sunulmuştur.

3.3.12. Unity 3D Başarı Sınavı

Öğrencilerin kendi oyun prototiplerini geliştirme sürecinde Unity 3D editöründe yer alan araç ve kavramlarla ilgili edindikleri bilgilerini ölçmek amacıyla bir başarı sınavı geliştirilmiştir. Soruların hazırlanması sürecinde ilk olarak madde belirtke tablosu oluşturularak her kazanımın ölçülebilmesi sağlanmıştır. Unity 3D Başarı Sınavına ait madde belirtke tablosu Tablo 3.20'de sunulmuştur.

Tablo 3.20. Unity 3D Başarı Sınavı madde belirtke tablosu

No	Kazanım	Unity 3D Başarı Sınavı Soru Numarası
1	Unity 3D editörünün The Toolbar penceresini tanır.	Bölüm 1-Soru 1- Md. a
2	Unity 3D editörünün The Hierarchy penceresini tanır.	Bölüm 1-Soru 1- Md. b
3	Unity 3D editörünün The Game View penceresini tanır.	Bölüm 1-Soru 1- Md. c
4	Unity 3D editörünün The Scene View penceresini tanır.	Bölüm 1-Soru 1- Md. d
5	Unity 3D editörünün The Inspector penceresini tanır.	Bölüm 1-Soru 1- Md. e
6	Unity 3D editörünün The Project penceresini tanır.	Bölüm 1-Soru 1- Md. f
7	Move aracı ile Move Gizmo'sunu eşleştirir.	Bölüm 1-Soru 2
8	Rotate aracı ile Rotate Gizmo'sunu eşleştirir.	Bölüm 1-Soru 2
9	Scale aracı ile Scale Gizmo'sunu eşleştirir.	Bölüm 1-Soru 2
10	RectTransform aracı ile RectTransform Gizmo'sunu eşleştirir.	Bölüm 1-Soru 2
11	Asset kavramını açıklar.	Bölüm 2-Soru 1- Md. a
12	Material kavramını açıklar.	Bölüm 2-Soru 1- Md. b
13	Shader kavramını açıklar.	Bölüm 2-Soru 1- Md. c
14	Prefab kavramını açıklar.	Bölüm 2-Soru 1- Md. d
15	Rigidbody kavramını açıklar.	Bölüm 2-Soru 1- Md. e
16	Particle kavramını açıklar.	Bölüm 2-Soru 1- Md. f
17	Vektörlerin normalleştirilmesinin gerekliliğini açıklar.	Bölüm 3- Md. a
18	Time.deltaTime ifadesinin kullanılma nedenlerini açıklar.	Bölüm 3- Md. b
19	Monobehavior metotlarından FixedUpdate() ile Update() metotlarını karşılaştırır.	Bölüm 3- Md. c
20	Unity 3D'de yer alan Monobehaviour metotlarının çalışma sırasını düzenler.	Bölüm 3- Md. d
21	Kendisine sunulan bir Monobehavior metodunun içeriğindeki C# kodlarındaki hataları düzeltir.	Bölüm 3- Md. e

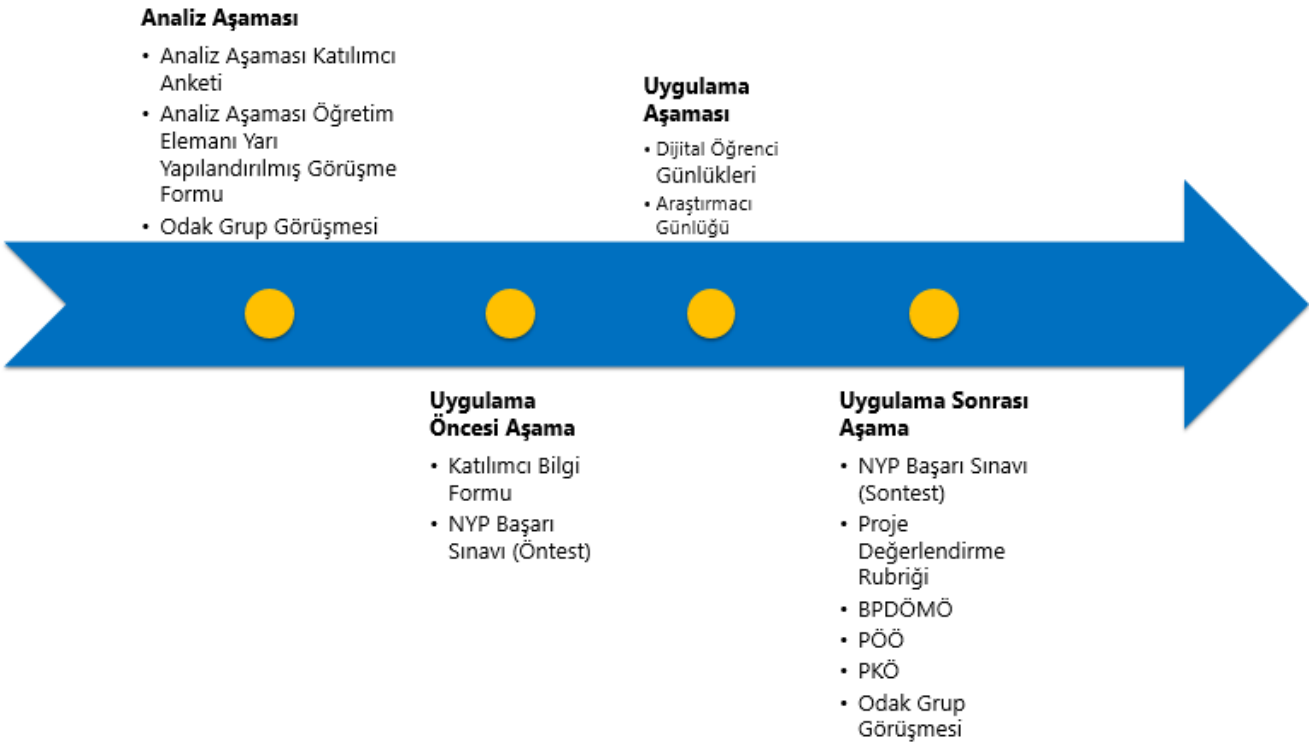
Unity 3D Başarı Sınavının geliştirilmesi sürecinin ikinci basamağında sınavın kapsam geçerliliği çalışmaları yürütülmüştür. Bu çalışmaların gerçekleştirilmesinde Davis'in (1992) işlem basamakları esas alınmıştır. Bu kapsamda, hazırlanan başarı sınavı Unity 3D Başarı Sınavı Kapsam Geçerliliği Uzman Görüş Formu (Ek 11) kullanılarak Tablo 3.5'te demografik özellikleri açıklanan BÖTE alanında uzman üç öğretim elemanı tarafından değerlendirilmiştir. Başarı sınavında yer alan sorular "*Uygun, Küçük düzeltmeler yapılmalı, Büyük düzeltmeler yapılmalı ve Uygun değil"* kriterlerine göre değerlendirilmiştir. Uzmanlardan elde edilen geri bildirimler doğrultusunda uygun ve küçük düzeltmeler yapılmalı maddeleri 1 puan; uygun değil ve büyük düzeltmeler yapılmalı maddeleri 0 puan

olarak kodlanarak her soru için madde geçerlilik indeksi hesaplanmıştır. Bununla birlikte tüm sınav sorularının indeks puanlarının ortalaması alınarak Unity 3D Başarı Sınavına ait kapsam geçerlilik indeksi belirlenmiştir. Bu kapsamda tüm sorular ve sınavın tamamı için kapsam geçerlilik indeksi ise 1.00 olarak belirlenmiştir. Bu sonuçlar başarı sınavının kapsam geçerliğinin sağlandığını göstermektedir. İkinci döngü sonunda Unity 3D Başarı Sınavına ait Cronbach Alfa (α) iç tutarlık katsayısı hesaplanarak analiz sonuçları tezin 3.7. Geçerlik ve Güvenirlik alt başlığında sunulmuştur. Unity 3D başarı sınavı örneği Ek 12'de sunulmuştur.

3.4. Veri Toplama Süreci

3.4.1. Birinci Döngü

Araştırmanın birinci döngüsünün analiz ve uygulama aşamalarında hem nitel hem de nicel veriler toplanmıştır. Analiz aşamasında mevcut NYP dersinde yaşanan problemlere ve bunların çözümlerine yönelik veriler toplanırken uygulama aşamasında, geliştirilen eylem planının etkisine yönelik veriler toplanmıştır. Veri toplama sürecinde kullanılan ölçme araçları Şekil 3.24'te belirtilmiştir.



Şekil 3.24. Birinci döngüye ait veri toplama süreci

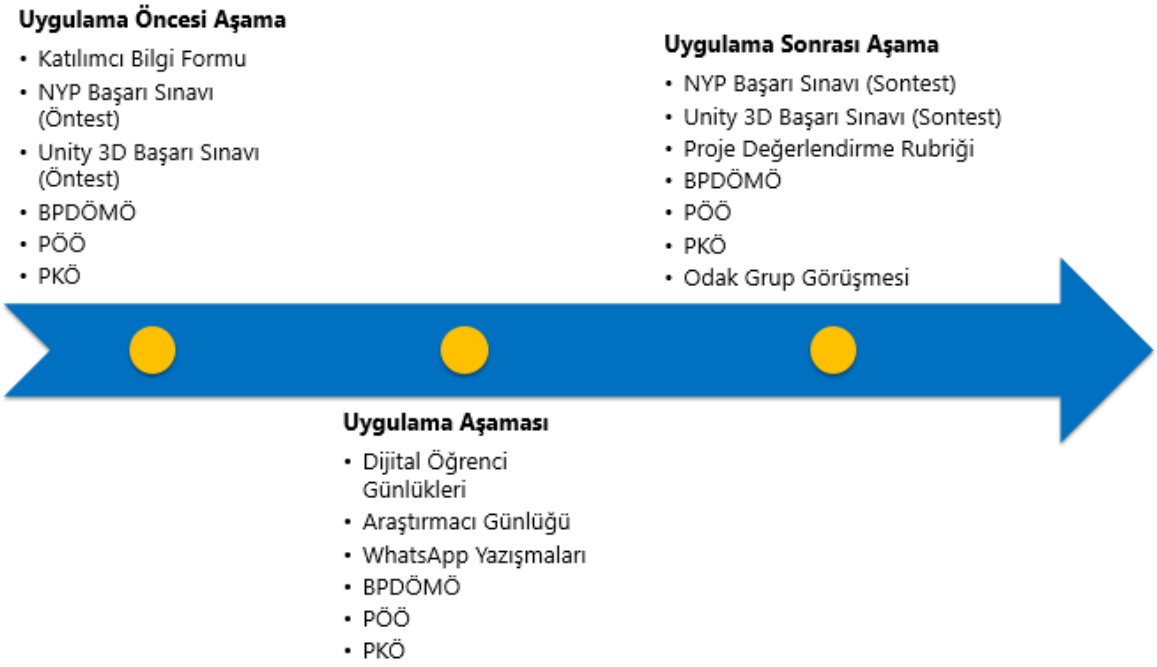
Kullanılan veri toplama araçlarına ilave olarak, araştırmanın birinci döngüsünün veri toplama sürecinde izlenen adımların detayları Tablo 3.21'de sunulmuştur.

Tablo 3.21. Birinci döngüde izlenen veri toplama adımları

Veri Toplama Süreci Aşamaları	İzlenen Adımlar
Analiz Aşaması	▪ Analiz Aşaması Katılımcı Anketi kullanılarak 59 mezun öğrenciden çevrimiçi olarak veri toplanmıştır. ▪ Analiz Aşaması Öğretim Elemanı Görüşme Formu kullanılarak altı öğretim elemanı ile yüz yüze görüşmeler yapılmıştır.
Uygulama Öncesi Aşama	▪ Katılımcı Bilgi Formu elden dağıtılarak 29 katılımcıdan veri toplanmıştır. ▪ NYP Başarı Sınavı öntest olarak yüz yüze derste uygulanmıştır.
Uygulama Aşaması	▪ Katılımcılar uygulama süresince dijital günlük tutmuşlardır. ▪ Araştırmacı uygulama süresince araştırmacı günlüğü tutmuştur.
Uygulama Sonrası Aşama	▪ NYP Başarı Sınavı sontest olarak yüz yüze derste uygulanmıştır. ▪ Katılımcılar oyun projelerini (öğrenci ürünlerini) yüz yüze derste teslim etmişlerdir. ▪ Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği, Programlama Öz Yeterlilik Ölçeği ve Programlama Kaygı Ölçeği sontest olarak elden dağıtılarak uygulanmıştır. ▪ Katılımcılarla yüz yüze odak grup görüşmesi yapılmıştır.

3.4.2. İkinci Döngü

İkinci döngünün veri toplama süreci uygulama öncesinde, uygulama aşamasında ve uygulama sonrasında olmak üzere üç aşamada gerçekleşmiştir. Veri toplama sürecinde kullanılan ölçme araçları Şekil 3.25'te belirtilmiştir.



Şekil 3.25. İkinci döngüye ait veri toplama süreci

Kullanılan veri toplama araçlarına ilave olarak, araştırmanın ikinci döngüsünün veri toplama sürecinde izlenen adımların detayları Tablo 3.22'de sunulmuştur.

Tablo 3.22. İkinci döngüde izlenen veri toplama adımları

Veri Toplama Süreci Aşamaları	İzlenen Adımlar
Uygulama Öncesi Aşama	▪ Katılımcı Bilgi Formu kullanılarak 30 katılımcıdan çevrimiçi olarak veri toplanmıştır. ▪ NYP Başarı Sınavı çevrimiçi olarak öntest olacak şekilde uygulanmıştır. Unity 3D Başarı Sınavı çevrimiçi olarak öntest olacak şekilde uygulanmıştır. ▪ Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği, Programlama Öz Yeterlilik Ölçeği ve Programlama Kaygı Ölçeği çevrimiçi olarak öntest olacak şekilde uygulanmıştır.
Uygulama Aşaması	▪ Katılımcılar uygulama sürecince dijital günlük tutmuşlardır. ▪ Araştırmacı uygulama sürecince araştırmacı günlüğü tutmuştur. ▪ WhatsApp yazışmaları vasıtasıyla katılımcıların oyun projelerinde karşılaştıkları programlama hatalarıyla ilgili veri toplanmıştır. ▪ Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği, Programlama Öz Yeterlilik Ölçeği ve Programlama Kaygı Ölçeği çevrimiçi olarak ara test olacak şekilde uygulanmıştır.
Değerlendirme Aşaması	▪ NYP Başarı Sınavı çevrimiçi olarak sontest olacak şekilde uygulanmıştır. ▪ Unity 3D Başarı Sınavı çevrimiçi olarak sontest olacak şekilde uygulanmıştır. ▪ Katılımcılar oyun projelerini (öğrenci ürünlerini) çevrimiçi olarak teslim etmişlerdir. ▪ Dört oturumda toplam 16 katılımcı ile yüz yüze odak grup görüşmesi yapılmıştır. ▪ Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği, Programlama Öz Yeterlilik Ölçeği ve Programlama Kaygı Ölçeği çevrimiçi olarak sontest olacak şekilde uygulanmıştır.

3.5. Verilerin Analizi

Bu tez çalışması kapsamında toplanan veriler, Clark, Porath, Thiele ve Jobe'nin (2020) önerdiği gibi "*ileri sürülebilecek herhangi bir iddia için güçlü kanıtlar sağlayacak temaları ve kalıpları belirlemek*" amacıyla kullanılmıştır. Bu doğrultuda elde edilen verilerden etkili çıkarımlar yapabilmek için veriler düzenlenmiş, araştırma soruları çerçevesinde betimlenmiş ve yorumlanmıştır (Yıldırım ve Şimşek, 2011). Verilerin analizi süreklilik göstermiş olup veri toplama işlemi ile eşzamanlı olarak yürütülmüştür (Karagöz, 2018). Her bir araştırma sorusu için toplanan veri türü ve bu verilerin çözümlenmesi amacıyla kullanılan veri analiz teknikleri Tablo 3.23'te sunulmuştur.

Tablo 3.23. Araştırma sorularını yanıtlamaya yönelik veri toplama teknikleri, veri türleri ve veri analizi

Araştırma Sorusu	Veri Toplama Tekniği	Veri Analiz Tekniği
1.1. Öğrencilerin mevcut NYP dersi hakkındaki görüşleri nelerdir?	Nitel	İçerik Analizi
1.2. Mevcut NYP dersinde öğrencilerin ve öğretim elemanlarının yaşadıkları sorunlar nelerdir?	Nitel	İçerik Analizi
1.3. Öğrenciler mevcut NYP dersinde ne tür programlama uygulamalarını geliştirmenin kendileri için daha faydalı olacağını değerlendirmektedir?	Nitel	İçerik Analizi
1.4. Mezun öğrencilerin mevcut NYP dersinde oyun geliştirmeye dayalı öğretimin uygulanabilirliği konusundaki değerlendirmeleri nelerdir?	Nitel	İçerik Analizi
1.5. NYP dersi Türkiye'deki diğer MYO'larda hangi içeriklerle verilmektedir?	Nitel	İçerik Analizi
1.6. Mevcut problemleri ortadan kaldırmak için oyun geliştirmeye dayalı öğretim NYP dersine nasıl entegre edilebilir?	Nitel	İçerik Analizi
1.7. Eylem planının uygulanması sürecinde katılımcıların ve araştırmacının yaşadığı sorunlar nelerdir?	Nitel	İçerik Analizi
1.8. Eylem planının uygulanmasından sonra katılımcıların nesne yönelimli programlama bilgi ve beceri düzeylerinde nasıl bir değişim meydana gelmiştir?	Nicel	Betimleyici İstatistik Normallik Testi (Shapiro-Wilk) Wilcoxon Sıralı İşaretler Testi Mann Whitney U Testi Korelasyon Analizi Regresyon Analizi
1.9. Geliştirilen Nesne Yönelimli Programlama dersine yönelik katılımcıların görüşleri nelerdir?	Nitel	İçerik Analizi
2.1. Birinci döngüden elde edilen deneyimler ışığında tasarlanan NYP dersinde ne gibi değişiklikler yapılabilir?	Nitel	İçerik Analizi
2.2. Eylem planının uygulaması sürecinde katılımcıların ve araştırmacının yaşadığı sorunlar nelerdir?	Nitel	İçerik Analizi
2.3. Eylem planının uygulanmasından sonra katılımcıların nesne yönelimli programlama bilgi ve beceri düzeyleri nasıl bir değişim göstermiştir?	Nicel	Betimleyici İstatistik Normallik Testi (Shapiro-Wilk) Wilcoxon Sıralı İşaretler Testi Mann Whitney U Testi Korelasyon Analizi
2.4. Eylem planının uygulanmasıyla birlikte katılımcıların psikolojik faktörlerinde (motivasyon, kaygı, öz yeterlilik algısı vb.) nasıl bir değişim meydana gelmiştir?	Nitel Nicel	İçerik Analizi Betimleyici İstatistik ANOVA testi
2.5. Geliştirilen Nesne Yönelimli Programlama dersine yönelik katılımcıların görüşleri nelerdir?	Nitel Nicel	İçerik Analizi Betimleyici İstatistik

Tablo 3.23'te görüldüğü üzere, toplanan verilerin ve araştırma sorularının doğasına uygun olarak bazen birden fazla veri analiz tekniği kullanılmıştır. Nitel ve nicel verilerin

analizinde kullanılan tekniklere yönelik detaylı açıklamalara aşağıdaki bölümlerde yer verilmiştir.

3.5.1. Nitel Tekniklerle Elde Edilen Verilerin Analizi

Bu tez çalışmasında elde edilen nitel veriler, eylem araştırmalarının dinamik ve döngüsel yapısına uygun olan (Erdoğan ve Akbaba, 2019) "*sistematik analitik analiz*" yaklaşımından ve "*içerik analizinden*" yararlanılarak çözümlenmiştir. Tez çalışması kapsamında gerçekleştirilen sistematik analitik analiz yaklaşımı çerçevesinde toplanan veriler hızla gözden geçirilmiş, araştırma sorularıyla karşılaştırılmış, varılan sonuçlar sınanmış ve elde edilen bulgulara yansıtmalar yapılmıştır (Gürgür, 2016). İçerik analizi çerçevesinde ise Robson'ın (2015) tanımladığı içerik analizi aşamaları olan (1) veri ile tanışmak, (2) ilk kodları üretmek, (3) temaları belirlemek, (4) tematik ağlar oluşturmak, ve (5) bütünleştirme ve yorumlama adımları takip edilmiştir. Bu kapsamda toplanan nitel veriler ilk olarak gözden geçirilmiş, daha sonra özetlenmiş, ardından da ortaklıklarına göre kodlanarak bulgulara dönüştürülmüştür (Karagöz, 2018). Bu süreç, eylem planının etkinliğinin değerlendirilerek ek eylemlere gerek duyulup duyulmadığını belirlemeye olanak tanımıştır (Erdogan ve Akbaba, 2019).

Tüm bu bilgilerin ışığında, nitel verilerin analizi kapsamında (1) açık uçlu soruların bulunduğu anketlerden, (2) dijital öğrenci günlüklerinden ve (3) araştırmacı günlüğünden elde edilen veriler direkt olarak içerik analizi yöntemiyle çözümlenmiştir. Bununla birlikte odak grup görüşmeleri yazılı doküman haline getirilerek çözümlenmiştir. Elde edilen nitel veriler satır satır okuma tekniği ile kodlanmış, bilgi parçalarına ayrılmış, bu parçalar kodlarla etiketlenmiş, gereksiz ya da çakışan kodlar elenmiş, son aşamada tümevarım yaklaşımı ile bu kodlar bir araya getirilerek temalar oluşturulmuştur (Creswell, 2005). Nitel verilerin çözümlenmesi süreci ilk olarak araştırmacı ve uzman bir değerlendirici tarafından ayrı ayrı yürütülmüştür. Daha sonra bu değerlendiriciler bir araya gelerek yapılan analizler üzerinde tartışılmış ve görüş birliğine varmıştır. Kodlayıcılar arası güvenirlik için, Miles ve Huberman'ın (2015) Güvenirlik=Görüş birliği/(Görüş birliği+Görüş ayrılığı)*100 formülünden yararlanılmıştır. Nitel verilerin analizinde MS Excel 2016 ve Atlas 9 yazılımlarından faydalanılmıştır.

3.5.2. Nicel Tekniklerle Elde Edilen Verilerin Analizi

Bu tez çalışması kapsamında kullanılan Katılımcı Bilgi Formu, Uygulama Sonrası Katılımcı Bilgi Formu, Proje Değerlendirme Rubriği, NYP Başarı Sınavı, Unity 3D Başarı Sınavı, Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği, Programlama Öz Yeterlilik Ölçeği, Programlama Kaygı Ölçeği gibi veri toplama araçlarından elde edilen verilerin analizinde Tablo 3.23'de yer alan çeşitli çözümleme tekniklerine başvurulmuştur. Betimsel istatistikler elde edilen verileri hakkında genel bir fikre sahip olmak ve uygulanacak çıkarımsal istatistiklerin varsayımlarını kontrol etmek amacıyla kullanılmıştır. Betimsel istatistiklerin hazırlanmasından ardından, hangi çıkarımsal istatistik analiz tekniklerinin kullanılacağına karar vermek amacıyla normallik testi gerçekleştirilmiştir. Bu amaçla eylem planlarının uygulamasında yer alan katılımcılara ait grup büyüklüğünün 50'den az olması nedeniyle normallik testi olarak Shapiro-Wilk normallik testleri uygulanmıştır (Büyüköztürk, Kılıç-Çakmak, Akgün, Karadeniz ve Demirel, 2008).

Tek grup için öntest-sontest karşılaştırmasının yapıldığı analizlerde normallik varsayımının karşılanmaması nedeniyle Wilcoxon İşaretli Sıralar Testi kullanılmıştır (Büyüköztürk ve diğerleri, 2008). Araştırmanın ikinci döngüsünde kullanılan ölçeklerin öntest, aratest ve sontest olarak uygulanması nedeniyle tek faktörlü tekrarlanan ölçümler ANOVA testi yürütülmüştür (Pallant, 2020). Analiz sonuçlarında p değeri için .05 anlamlılık derecesi kullanılmıştır. Süreç içerisinde kullanılan başarı sınavları, proje değerlendirmeleri, ölçek ve anket verilerinin analizinde SPSS 22.0 (Statistical Programme for Social Science), ile MS Excel 2016 paket programı kullanılmıştır. Programlama Öz Yeterlilik Ölçeğinin uyarlanması ve Programlama Kaygı Ölçeğinin geliştirilmesi aşamalarında bu yazılımlara ilave olarak AMOS 22 paket programı kullanılmıştır.

3.6. Araştırmacının Rolü

Araştırmacı, bir eylem araştırması olarak tasarlanan bu tez çalışmasında çalışmanın sahip olduğu "*deneyime dayalı, eylem odaklı ve karşılaşılan sorunların çözümünü amaçlayan uygulamalı bir araştırma türü"* (Ladkin, 2004) karakteristikleriyle uyumlu olarak hem araştırmacı hem de dersi veren öğretim elemanı olmak üzere ikili bir rol üstlenmiştir (Olsen ve Lindøe, 2004). Bu kapsamda araştırmacı; (1) mevcut NYP dersinde yaşanan sorunların tespit edildiği ve öğrencilerin ders hakkındaki görüşlerinin elde edildiği analiz aşamasında veri toplamış; (2) analiz aşamasında toplanan verilerin çözümlemesini

gerçekleştirmiş, (3) toplanan verilerin, uzman görüşlerinin ve alanyazın desteği ile NYP dersini daha etkili ve verimli kılmak için oyun geliştirmeye dayalı öğretim stratejisini temel alan bir eylem planı hazırlamış, (4) hazırlanan eylem planına uygun olarak NYP dersine yönelik öğretim tasarımını gerçekleştirmiş, (5) eylem planının uygulanmasını iki döngüden oluşacak şekilde planlayarak yüz yüze ve çevrimiçi öğretim etkinliklerini yürütmüş, (5) döngülerin uygulama süreçlerinde ve uygulamaların sonrasında eylem planının etkinliğini değerlendirebilmek amacıyla veri toplayarak yansıtmalarda bulunmuştur.

3.7. Geçerlik ve Güvenirlik

Çalışmanın bu bölümünde, nicel ve nitel araştırma tekniklerine göre toplanan verilerin geçerlik ve güvenirliklerini sağlayabilmeye yönelik izlenen adımlar sunulmaktadır.

3.7.1. Nitel Araştırma Teknikleri Boyutu

Bu eylem araştırmasının nitel yanının yansız bir biçimde sunulabilmesi ve niteliğinin üst düzeye çıkarılabilmesi için *(1) inandırıcılık, (2) aktarılabilirlik, (3) tutarlılık* ve *(4) onaylanabilirlik* olmak üzere dört boyutta çalışmalar yürütülmüştür (Yıldırım ve Şimşek, 2011). Bu kapsamda gerçekleştirilen çalışmalar Tablo 3.24'te sunulmuştur.

Tablo 3.24. Nitel araştırma teknikleri boyutunda gerçekleştirilen geçerlik ve güvenirlik çalışmaları

Boyut	Yapılan Çalışmalar
İnandırıcılık	Uzman incelemesi
	Katılımcılarla uzun süre etkileşim
	Derinlik odaklı veri toplama
	Veri çeşitleme
Aktarılabilirlik	Katılımcıların ayrıntılı tanıtımı
	Ortamın ayrıntılı tanıtımı
	Yürütülen süreçlerin ayrıntılı tanıtımı
Tutarlılık	Araştırma yöntemlerinin ayrıntılı tanıtımı
	Başka araştırmacıların süreç ve sonuçları incelemesi
Onaylanabilirlik	Nesnel yansıtma
	Değerlendiriciler arası uyum
	Verilerin saklanması

İnandırıcılık boyutunun sağlanabilmesi için bu eylem araştırmasının verileri biri analiz ikisi uygulama olmak üzere toplam üç aşamada toplanmıştır. Bu aşamalar uzun bir zaman

dilimine yayılarak katılımcılarla uzun süre etkileşim sağlanmıştır. Araştırma sürecinde elde edilen veriler araştırma soruları bağlamında düzenli olarak çözümlenmiştir. Bu sayede toplanan tüm verilerin sıklıkla kontrolü sağlanarak veri çözümlemelerinden sonra uzman görüşüne başvurulabilmiştir. Araştırmada farklı veri toplama araçlarının kullanımı anlamına gelen veri çeşitlemeye özen gösterilmiş; bu veriler farklı zamanlarda toplanarak verilerin geçerliği sağlanmıştır. Veri kaybının önüne geçilebilmesi için toplanan tüm veriler zamanında arşivlenmiştir.

Aktarılabilirlik boyutunun sağlanabilmesi için araştırmada ayrıntılı betimlemelere başvurulmuştur. Araştırmanın katılımcılarının, uygulama ortamının, araştırmada kullanılan araçların ve araştırma kapsamında yürütülen süreçlerin ayrıntılı tanıtımları gerçekleştirilmiştir. Bu yapılırken açık ve anlaşılır bir dilin kullanımına özen gösterilmiştir.

Tutarlılık boyutunun sağlanabilmesi için programlama öğretimi alanyazınında yer alan farklı araştırmaların eylem araştırmasıyla yapılması bağlamında makul bir benzerlik sağlanmıştır. Araştırma yöntemi ayrıntılı bir şekilde açıklanmış; gerçekleştirilen süreçleri doktora tez danışmanı, tez izleme komitesi üyeleri gibi farklı uzmanların incelemesi sağlanmıştır. Bunun yanında verilerin çözümlenmesi, araştırmacının dışında farklı bir uzman tarafından daha gerçekleştirilmiş, bu sayede çözümlemelerin tutarlı olması sağlanmıştır. Tüm bunlara ilave olarak tutarlılık bağlamında araştırmadan elde edilen tüm veriler araştırma sorularıyla uyumlu olacak şekilde sunumu gerçekleştirilmiştir.

Onaylanabilirlik boyutunun sağlanabilmesi için araştırma kapsamında toplanan veriler nesnel olarak değerlendirilmiştir. Elde edilen sonuçlara ulaştıran kanıtların, sonuç çıkarım süreçlerinin ve değerlendirmelerin mümkün olduğunca açıklanmasına özen gösterilmiştir (Lincon ve Guba, 1985). İlave olarak verilerin çözümlenmesi araştırmacının dışında farklı bir uzman tarafından da gerçekleştirilmiş; değerlendiriciler arası uyum hesaplanarak nesnel değerlendirmelerin yapılması sağlanmıştır (Miles ve Huberman, 2015). Araştırma soruları bağlamında hesaplanan araştırmacılar arası uyum katsayıları, uyum yüzdeleri ve sonuçların yorumlanması Tablo 3.25'te sunulmuştur. Tabloya göre iki değerlendiricinin nitel verilerin çözümlenmesi bağlamında uyumların %85'in üzerinde olması ve hesaplanan Cohen Kappa değerinin .80'den büyük olması nedeniyle bu tez çalışması güvenilir bir çalışma olarak kabul edilebilmektedir (Krippendorff, 2018; Neuendorf, 2017). Son olarak *onaylanabilirlik* bağlamında tüm ham verilerin ve ham verilerden oluşan çözümlemelerin gerektiğinde incelemeye hazır olacak şekilde muhafaza edilmesi çalışmanın güvenilirliğini artırmıştır.

Tablo 3.25. Araştırma soruları bağlamında araştırmacılar arası uyum ve güvenirlik analizleri

Araştırma Sorusu	Uyum (%)	Cohen's Kappa	Yorum
1.1	97	.95	Çok iyi düzeyde uyum
1.2	95	.91	Çok iyi düzeyde uyum
1.3	95	.90	Çok iyi düzeyde uyum
1.4	96	.91	Çok iyi düzeyde uyum
1.5	100	1.00	Çok iyi düzeyde uyum
1.6	90	.82	Çok iyi düzeyde uyum
1.9	92	.85	Çok iyi düzeyde uyum
2.4	92	.83	Çok iyi düzeyde uyum
2.5	93	.86	Çok iyi düzeyde uyum

3.7.2. Nicel Araştırma Teknikleri Boyutu

Nicel veri toplama araçlarının güvenilirlik çalışması kapsamında, araştırmada kullanılan başarı sınavlarının ve likert tipi ölçeklerin iç tutarlılıkları Cronbach Alfa (α) iç tutarlık katsayıları hesaplanarak belirlenmiştir. Bu sayede ölçeklerin ölçmek istediği değişkeni ne tutarlılıkla ölçtüğünün ya da ölçme sonuçlarının hatalardan arındırılmış olmasının derecesi ortaya çıkarılmıştır (Tavşancıl, 2002). İç tutarlılık katsayısının .70'den büyük olmasının ölçeklerin güvenilir olarak kabul edilmesini sağladığı değerlendirildiğinden (Taber, 2018), çalışma kapsamında ölçek alt boyutlarında ve ölçeğin genelinde .70'den büyük iç tutarlılık katsayısı aranmıştır.

NYP Başarı Sınavının birinci ve ikinci döngüde; Unity 3D Başarı Sınavının ikinci döngüde sontest olarak uygulanmasının ardından Cronbach Alfa (α) iç tutarlık katsayıları hesaplanarak güvenirlik analizleri yapılmıştır. NYP Başarı Sınavının birinci döngü için .89; ikinci döngü için .91 olarak hesaplanan Cronbach Alfa (α) iç tutarlık katsayısı sınavın güvenilir sonuçlar ürettiğini göstermektedir. Benzer şekilde Unity 3D Sınavının Cronbach Alfa (α) iç tutarlık katsayısı .92 olarak hesaplanmıştır. Bu sonuç da sınavın sınavın güvenilir sonuçlar ürettiğini göstermektedir.

Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği, Programlama Öz Yeterlilik Ölçeği ve Programlama Kaygı Ölçeği birinci döngüde sontest olarak; ikinci döngüde ise öntest, aratest ve sontest olarak uygulanmıştır. Ölçeklerin her bir uygulanışının ardından alt faktörlerine ve ölçeklerin geneline ait Cronbach Alfa (α) iç tutarlık katsayıları

hesaplanarak güvenirlik analizleri yapılmıştır. Analiz sonuçları Tablo 3.26-Tablo 3.28'de sunulmuştur.

Tablo 3.26. Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeğine ait Cronbach Alfa (α) iç tutarlık katsayıları

Alt Boyut	**Birinci Döngü**	**İkinci Döngü**		
	Sontest	**Öntest**	**Aratest**	**Sontest**
Tutum ve Beklenti	.84	.71	.73	.72
Zorlayıcı Amaçlar	.89	.82	.88	.91
Belirgin Hedefler	.88	.72	.71	.72
Ödül ve Takdir	.73	.73	.80	.85
Ceza	.72	.77	.82	.95
Sosyal Baskı ve Rekabet	.87	.75	.72	.79
Ölçeğin Geneli	.92	.85	.87	.91

Tablo 3.27. Programlama Öz Yeterlilik Ölçeğine ait Cronbach Alfa (α) iç tutarlık katsayıları

	Birinci Döngü	**İkinci Döngü**		
	Sontest	**Öntest**	**Aratest**	**Sontest**
Ölçeğin Geneli	.97	.93	.96	.97

Tablo 3.28. Programlama Kaygı Ölçeğine ait Cronbach Alfa (α) iç tutarlık katsayıları

Alt Boyut	**Birinci Döngü**	**İkinci Döngü**		
	Sontest	**Öntest**	**Aratest**	**Sontest**
Akran Kaygısı	.89	.80	.89	.88
Programlama Beceri Kaygısı	.92	.85	.88	.91
Ölçeğin Geneli	.94	.89	.93	.92

Her üç tablo da incelendiğinde ölçeklere ait alt boyutların ya da ölçeklerin genelinin iç tutarlılık katsayılarının .70'den büyük olduğu görülmektedir. Bu durum ölçeklerin güvenilir sonuçlar ürettiğini göstermektedir. Benzer şekilde ölçeklerin güvenilirliklerinin yüksek olması yapı geçerliliklerinin de yüksek olduğunu göstermektedir (Yaşar, 2014).

3.8. Araştırmanın Etiği

Etik, diğer araştırma türlerinde olduğu gibi eylem araştırmalarında da önemli bir konudur. İyiliği, güvenilirliği ve dürüstlüğü teşvik eden; aynı zamanda katılımcıların zarar

görmelerini önleyen ahlaki ve yol gösterici ilkeleri barındıran etik kuralları (Morrow ve Richards, 1996) bu tez çalışmasında titizlikle uygulanmıştır. Bu kapsamda katılımcılar araştırmanın amaçları konusunda detaylı bir şekilde bilgilendirilmiş, kendilerinden ne tür veriler toplanacağı ve toplanan bu verilerin hangi tür işlemlere tabi tutulacağı konusunda aydınlatılmıştır. Araştırma sürecince gizliliğe önem verilmiş; katılımcılara ait kimlik bilgileri gizli tutularak raporlaştırma aşamasında katılımcılara kodlar verilmiştir. Araştırmada elde edilen ses ve görüntü kayıtlarında katılımcılardan izin alınmıştır. Bu kayıtlar yalnızca araştırmacının veri depolama aygıtlarında saklanmıştır.

Araştırmanın etiği ile ilgili olarak üzerinde titizlikle durulan bir diğer konu ise katılımcıların araştırmanın hiçbir aşamasında zarar görmemelerinin sağlanmasıdır. Araştırmanın birinci döngüsünün uygulama süreci pandemi şartlarında sekiz hafta sürecek şekilde yüz yüze olarak gerçekleştirilmiştir. Ancak bu sekiz haftalık eğitim sadece NYP dersine özgü olmayıp genel bir uygulamayı içermiştir. Katılımcılar bu süreçte NYP dersi ile birlikte diğer ön lisans derslerini almaya devam etmişler; okullarında sadece bu tez çalışması kapsamında bulunmamışladır. Uygulama esnasında Covid-19 tedbirlerine azami dikkat edilmiş; maske ve mesafe kuralları özenle uygulanmıştır. Benzer şekilde öğrencilerin okullarına giriş çıkışları kısıtlı bir şekilde gerçekleştirilmiştir.

BÖLÜM IV: BULGULAR VE YORUMLAR

4.1. Birinci Döngü

4.1.1. Analiz Aşaması

Birinci döngünün analiz aşaması bulguları alt maddelerde açıklanmıştır. Tasarlanan bu eylem araştırmasının temel amacı olan "*NYP dersinin geliştirilmesi ve iyileştirilmesi"* sürecinin daha iyi anlaşılabilmesi için öncelikle mevcut NYP dersinin karakteristik özelliklerinden bahsetmenin uygun olacağı değerlendirilmektedir.

4.1.1.1. Mevcut NYP dersinin karakteristik özellikleri

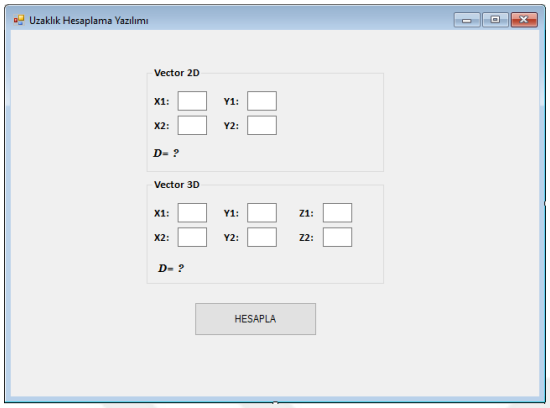
Mevcut NYP dersi haftada bir saat teorik, iki saat uygulama olmak üzere üç saatlik bir derstir. Bir ders saati 40 dakikadır. 13 hafta ders anlatımı ve iki hafta sınav haftası olmak üzere toplam 15 haftalık bir öğretim programına sahiptir. Dersin öğretim programı Tablo 4.1'de sunulmuştur. Bu dersi araştırmacının haricinde altı öğretim görevlisi daha vermekte olup tüm öğretim elemanları derslerinin işlenmesi sürecinde Tablo 4.1'de belirtilen öğretim programını takip etmektedir.

Tablo 4.1. Mevcut NYP dersine ait öğretim programı

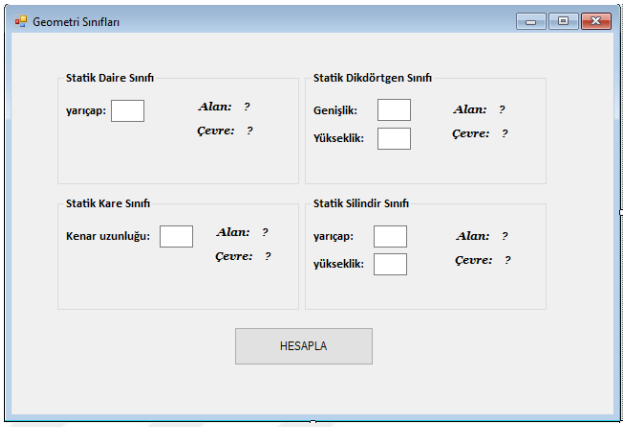
Hafta No	Konu
1	Nesne Yönelimli Programlamanın Temelleri
2-3	Sınıf ve Nesneler
4-5	Sınıf Metotları
6	Veri Yapıları
7	Statik Sınıflar ve Metotlar
8	Vize Haftası
9-11	Kalıtım
12-13	Nesne Yönelimli Proje Geliştirme
14-15	Kurulum Dosyası Oluşturma
16	Final Haftası

Mevcut NYP dersinde öğrenciler Tablo 4.1'de yer alan öğretim programı kapsamında nesne yönelimli programlamanın temelleri, sınıflar ve nesneler, sınıf metotları, kurucu metotlar, UML diyagramları, arayüzler, kalıtım, nesne yönelimli proje geliştirme ve kurulum dosyası oluşturma gibi konularda bilgi sahibi olmaktadır. Dersin uygulama saatlerinde ise

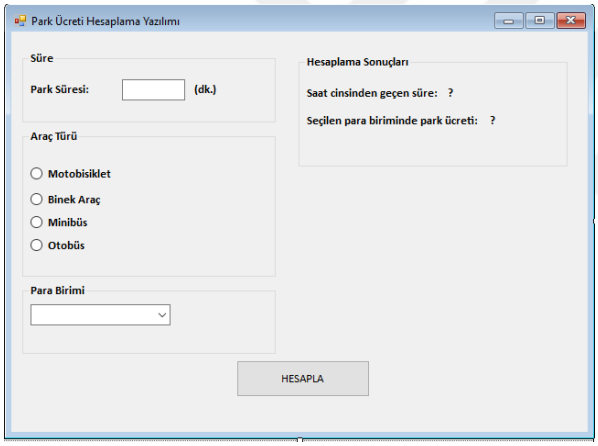
teorik saatlerde edinilen nesne yönelimli programlama teknikleriyle ilgili bilgiler kullanılarak form uygulamaları geliştirilmektedir. Derste programlama dili olarak C# kullanılmakta olup uygulama saatlerinde geliştirilen nesne yönelimli yazılım projelerinden bazılarının ekran alıntısı Şekil 4.1-Şekil 4.4'te sunulmuştur.



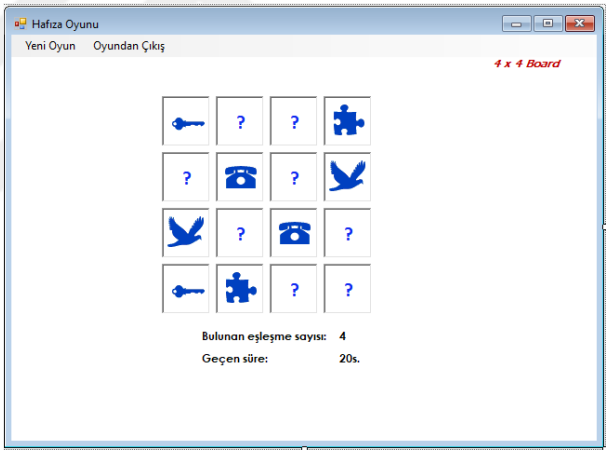
Şekil 4.1. Noktalar arası mesafe hesaplama projesi



Şekil 4.2. Statik sınıflar kullanarak bazı geometrik şekillerin özelliklerini hesaplama projesi



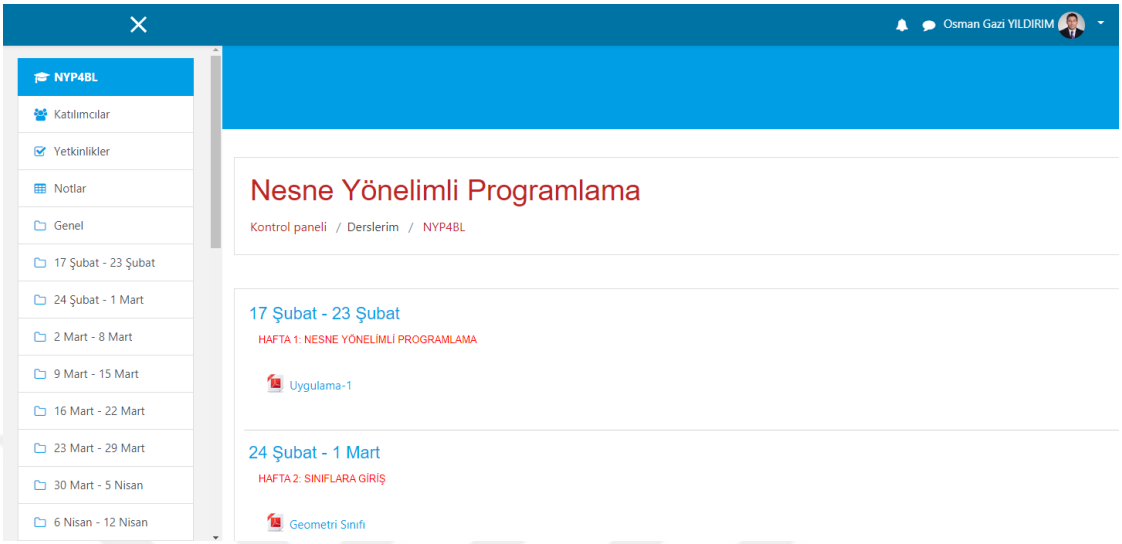
Şekil 4.3. Kalıtım kullanılarak farklı tip araçların otopark ücretlerinin hesaplanması projesi



Şekil 4.4. Basit bir hafıza oyunu

Şekil 4.1'de iki boyutlu ve üç boyutlu uzayda iki nokta arasındaki mesafenin hesaplanmasını sağlayan programın ekran alıntısı verilmiştir. Bu yazılım projesini geliştirebilmek için öğrenciler C#'ta arayüzleri ve kalıtımı ve kullanmaktadır. Şekil 4.2'de geometrik şekillerin özelliklerinin (alan, çevre, hacim vb.) hesaplanmasını sağlayan programın ekran alıntısı sunulmuştur. Bu hesaplamaların yapılabilmesi amacıyla öğrenciler statik geometrik sınıflar oluşturmakta ve statik sınıf metotları tanımlamaktadırlar. Şekil 4.3'te öğrenciler kalıtım kullanarak bir otopark ücreti hesaplama projesi geliştirmektedirler. Son olarak Şekil 4.4'te ise NYP mekanizmalarının kullanıldığı ve 4x4, 6x6 ve 8x8 gibi farklı

oyun tahtalarında oynanabilen basit bir hafıza oyunu geliştirmektedirler. Öğrenciler nesne yönelimli yazılım projelerini geliştirirken kaynak olarak ders kitaplarını ve MOODLE tabanlı ders yönetim sistemini kullanmaktadırlar (Şekil 4.5).



Şekil 4.5. Ders yönetim sisteminin ekran alıntısı

Mevcut NYP dersinde öğrenciler vize, final ve değerlendirme notu olmak üzere üç farklı kritere göre değerlendirilmektedir. Öğrenciler dersin 8.haftasında teorik ve uygulama sınavlarını içeren vize sınavına girmektedir. Benzer şekilde 16.haftada teorik ve uygulama sınavlarını içeren final sınavına girmektedir. Değerlendirme notu ise öğrencilerin derse katılım, ödev, proje gibi aktiviteleri tamamlama durumuna göre derse giren öğretim elemanı tarafından verilmektedir. Vize, final ve değerlendirme notlarının ağırlıkları Tablo 4.2'de sunulmuştur.

Tablo 4.2. Vize, final ve değerlendirme notlarının ağırlıkları

Değerlendirme Türü		**Ders Başarısına Etkisi**
Vize	Teorik	%12
	Uygulama	%20
Final	Teorik	%16
	Uygulama	%24
Değerlendirme Notu (Derse katılım, ödev, proje vb.)	-	%8

4.1.1.2. Araştırma Sorusu 1: Öğrencilerin mevcut NYP dersi hakkındaki görüşleri nelerdir?

Öğrencilerin mevcut NYP dersi hakkındaki görüşlerini elde edebilmek için Analiz Aşaması Katılımcı Anketi kullanılmıştır. Bunun yanında daha önce dersi almış ve okullarından mezun olmuş öğrencilerle odak grup görüşmesi yapılmıştır. Elde edilen verilerin çözümlenmesi sonucunda NYP dersinin öğrenciler açısından katkıları ile derste gerçekleştirilen nesne yönelimli programlama projelerinin faydalı olup olmama durumları ortaya çıkarılmıştır. Bu sonuçlar aşağıdaki alt bölümlerde açıklanmıştır.

4.1.1.2.1. Mevcut NYP dersinin öğrencilere sağladığı katkıların değerlendirilmesi

İlk olarak, daha önce NYP dersini almış öğrencilerin dersi hangi açılardan değerli bulduklarını incelemek için kendilerine "*NYP dersini faydalı buluyor musunuz? Neden?*" sorusu yöneltilmiştir. Bu bağlamda mevcut NYP dersinin öğrencilere hangi açılardan katkı sağladığı analiz edilmiş olup bu temaya ait kategori, frekans ve yüzde bilgileri Tablo 4.3'te sunulmuştur.

Tablo 4.3. Mezun öğrencilerin mevcut NYP dersinin katkılarına yönelik görüşleri*

Tema	Kategori	Kodlar	Frekans (f)	Yüzde (%)
NYP Dersinin Sağladığı Katkıların Değerlendirilmesi	**Yeterliliklere Katkı**	NYP paradigması ile ilgili bilgilerin öğrenilmesini sağlıyor.	19	32
		Nesne yönelimli tasarım yapma becerisi kazandırıyor.	15	25
		Algoritmik/analitik düşünme becerimizin gelişmesini sağlıyor.	12	20
		Problem çözme becerimizin gelişmesini sağlıyor.	10	17
		Yaratıcı düşünme becerimizin gelişmesini sağlıyor.	5	9
	Gelecekte Elde Edilecek Olanaklara Katkı	İstihdam olanağımızı artırıyor.	11	19
		Teknolojik yeniliklere ayak uydurabilmemiz için altyapı oluşturuyor.	5	9
	Katkısız	Gelecekte programlama içeren bir alanda çalışmayacağım için katkısı bulunmuyor.	2	3

Tablo incelendiğinde; NYP paradigması ile ilgili bilgilerin öğrenilmesini ve NYP tekniğini kullanarak yazılım tasarlamaya olanak sağlamasının mevcut NYP dersinin öğrencilerin kişisel yeterlilikleri katkısına yönelik görüşler arasında sayıca ilk sıralarda yer almaktadır. Ayrıca öğrenciler mevcut NYP dersinin algoritmik/analitik düşünme, problem

* Katılımcı sayısı: 59. Bir katılımcının vermiş olduğu cevap birden fazla kod içerebilir.

çözme, yaratıcı düşünme gibi becerilerinin gelişmesine yardımcı olduğunu belirtmektedir. Bu konu ile ilgili katılımcı görüşlerinden bazıları aşağıda verilmiştir.

Bu ders en çok sınıf, nesne, kalıtım gibi NYP ile ilgili bilgilerin öğrenilmesini sağlıyor. Bu öğrendiklerimizi dersin uygulama saatlerinde ders projelerimizde uygulayarak pekiştiriyoruz. Bu sayede Görsel Programlama-1 dersinde öğrendiklerimizin üstüne yeni şeyler ekliyoruz. (M5)

Yararlı olduğunu düşünüyorum çünkü programlama öğrenirken aslında problemleri daha hızlı ve daha farklı yollar bularak çözebilmeyi öğreniyoruz. (M19)

Katılımcılar, NYP dersinin kendilerinin kişisel yeterliliklerine katkısının yanında gelecekte istihdam olanaklarına (N:11) ve teknolojik bilgi ve beceri altyapılarına (N:5) katkıda bulunduğunu belirtmiştir. Bu konu ile ilgili M12 kodlu katılımcının görüşleri aşağıdaki gibidir:

Programlama geleceğin yapı taşlarından biri. Günümüzde birçok insan dijital cihazlar ve birçok sanal uygulama kullanıyor ve öyle görünüyor ki yıllar ilerleyip teknoloji geliştikçe de bu alanda ki ihtiyaç kat sayısı giderek artacaktır. Herkes mutlu olacağı işi yapmalı bende programlamaktan mutlu oluyorum ve bu işi yapmak istiyorum.

Dersin faydalı olduğunu düşünmeyen her iki katılımcı da okullarından mezun olduktan sonra ağ teknikeri olarak çalışmaya başladıklarını, mesleklerinin programlama ilgili hususları barındırmadığını düşündükleri için NYP dersinin kendilerine katkısının olmadığını belirtmişlerdir. Bu konu ile ilgili M27 kodlu katılımcı görüşlerini şu şekilde ifade etmiştir:

Gereksiz bir ders, çünkü bizim işimize yarayan Cisco'da [bilgisayar ağ teknolojileri alanında] kendimizi geliştirmek.

4.1.1.2.2. Derste gerçekleştirilen NYP projelerinin öğrenciler açısından faydalı olup olmama durumunun değerlendirilmesi

NYP dersinin katkılarının yanında, katılımcıların derste gerçekleştirdikleri programlama projelerini faydalı bulup bulmamalarıyla ilgili görüşleri de elde edilmiştir. Mevcut NYP dersindeki projelerin değerlendirilmesi temasına ait kategori, frekans ve yüzde bilgileri Tablo 4.4'te sunulmuştur. Tabloya göre, ders içi projeler en çok teorik derslerde öğrenilen konuların pratiğinin yapılmasını ve konuların tekrar edilmesini sağlamaktadır. Bunun yanında ders için projelerin, öğrencilerin yeni yöntemler denemelerine, problem çözmelerine ve hatalarını giderip hatalarından ders almalarına olanak sağladığı belirtilen faydalar arasındadır. Bu konu ile ilgili katılımcı görüşlerinden bazıları aşağıda verilmiştir.

Ders projeleri derste öğrendiklerimi tek başımıza bir uygulamada hayata geçirmeyi sağlıyor. Bu sayede öğrendiklerimizin kalıcılığının arttığını düşünüyorum. (M23)

[Programlama projelerinin]Programlama bilgime katkısı büyüktür. Projelerde deneme yanılma yapma fırsatı bulduğumuzdan faydalı olduğunu düşünüyorum. Karşılaştığım hatalar sayesinde problemleri çözmekte eskisi kadar zorluk çekmiyorum. (M41)

Tablo 4.4. Mezun öğrencilerin mevcut NYP dersindeki projelere yönelik görüşleri*

Tema	Kategori	Kodlar	Frekans (f)	Yüzde (%)
Mevcut NYP Dersindeki Projelerin Değerlendirilmesi	Olumlu	Teorik konuların uygulanmasını sağlıyor.	21	36
		Öğrendiğimiz konuların tekrar edilebilmesine imkân tanıyor.	16	27
		Yeni yöntemler kullanma ve denemeye olanak sağlıyor.	10	17
		Problem çözme fırsatı sunuyor.	8	14
		Yazılım hatalarımızı çözmeye ve bunlardan öğrenmeye olanak tanıyor.	3	5
	Olumsuz	Günlük hayatla ilişkili projeler yapmıyoruz.	26	44
		Görsel olarak çekici projeler üretmiyoruz.	24	41
		Keyifli projeler gerçekleştirmiyoruz.	21	36
		Projeler yaratıcı düşünmeyi teşvik etmemektedir.	16	27

Olumlu görüşlere ilave olarak bazı katılımcıların mevcut NYP dersindeki projelerle ilgili olumsuz görüşleri de bulunmaktadır. Bu görüşler değerlendirildiğinde, projelerin günlük hayatla ilişkili konular içermemesi, dönem boyu benzer konularda proje geliştirilmesi, projelerin keyifli, profesyonel ve görsel olarak çekici olmaması katılımcıların ders içi projelerle ilgili olumsuz görüşlerini yansıttığı söylenebilir. Bu konu ile ilgili katılımcı görüşlerinden bazıları aşağıda verilmiştir.

Motivasyonu azaltan en büyük etkenin ilgisizlik olması olduğunu düşünüyorum. İlgi her şeyin anahtarı olabilir. Derste birçok noktada beni çözüme veyahut sonuca ulaştıran ilgiydi. Derslerde bir pizza sipariş programı tasarlayıp kodlamak herkes için çok keyifli olmuyor. Bu da ders ilgimizin azalmasına sebebiyet veriyor. (M47)

Derslerin daha zevkli geçmesi ve ilgi çekmesi adına grafiksel ağırlıklı işlenirse ve öğrencinin sevebileceği türde örnekler üzerinden gidilirse daha verimli ve faydalı olabilir. (M18)

Derslerde düzgün bir şey geliştirdiğimizi düşünmüyorum. Sadece temel şeyleri yapıyoruz. (M33)

* Katılımcı sayısı: 59. Bir katılımcının vermiş olduğu cevap birden fazla kod içerebilir.

4.1.1.3. Araştırma Sorusu 2: Mevcut NYP dersinde öğrencilerin ve öğretim elemanlarının yaşadıkları sorunlar nelerdir?

4.1.1.3.1. Mevcut NYP dersinde öğrencilerin yaşadıkları sorunların değerlendirilmesi

Mevcut NYP dersinde öğrencilerin yaşadıkları sorunlar, tıpkı öğrencilerin NYP dersi ve derste geliştirilen projeler hakkındaki görüşlerinin ortaya çıkarılmasında olduğu gibi NYP Dersi Analiz Aşaması Katılımcı Anketi ile NYP Dersi Analiz Aşaması Katılımcı Yarı Yapılandırılmış Görüşme Formu kullanılarak belirlenmiştir. Elde edilen verilerin çözümlenmesinden sonra öğrencilerin mevcut NYP dersinde yaşadıkları sorunlar; *(1) öğretim programı, (2) dersin işlenişi, (3) ders etkinlikleri, (4) ders materyalleri, (5) değerlendirme yöntemi ve (6) teknolojik altyapı* olmak üzere altı alt kategoride gruplandırılmıştır. İlk beş alt kategoriye ait kodlar Tablo 4.5'te sunulmuştur. Ders etkinlikleri alt kategorisine ait kodlar ise öğrencilerin derste geliştirilen nesne yönelimli programlama projeleri hakkındaki görüşlerinin sunulduğu Bölüm 4.1.1.2.2'de daha önce açıklanmıştır.

Tablo 4.5. Mezun öğrencilerin mevcut NYP dersinde yaşadıkları sorunlara ilişkin görüşleri*

Tema	Kategori	Kodlar	Frekans (f)	Yüzde (%)
Öğrencilerin Mevcut NYP Dersinde Yaşadıkları Sorunların Değerlendirilmesi	**Öğretim Programı**	NYP'de yer alan kavramları anlamada ve yazılım geliştirme sürecinde kullanmada güçlük yaşıyorum.	25	42
	Dersin İşlenişi	Derste öğrencilerin programlama hazırbulunuşluk seviyesi dikkate alınmıyor.	15	10
		Derste kullanılan kodlar detaylı açıklanmıyor.	12	7
	Ders Etkinlikleri	Günlük hayatla ilişkili projeler yapmıyoruz.	26	44
		Görsel olarak çekici projeler üretmiyoruz.	24	41
		Keyifli projeler gerçekleştirmiyoruz.	21	36
		Projeler yaratıcı düşünmeyi teşvik etmemektedir.	16	27
	Ders Materyalleri	NYP ile ilgili yeterli Türkçe kaynak bulunmuyor.	19	32
	Değerlendirme Yöntemi	Proje ve ödevlerin genel değerlendirmede ağırlıkları düşüktür.	24	41
		Sınavlar konuları ezberlemeye yönlendiriyor.	8	14
	Teknolojik Altyapı	Kullandığımız bilgisayarların donanımları güncel değil.	20	34
		İnternete bağlanmada sorunlar bulunuyor.	22	37

* Katılımcı sayısı: 59. Bir katılımcının vermiş olduğu cevap birden fazla kod içerebilir.

Öğretim programı ile ilgili olarak katılımcıların en sık karşılaştıkları zorluk, ders içeriğinin kendileri açısından zorlayıcı olmasıdır. Bunun sebeplerinin başında NYP'de görsel programlamaya (prosedürel programlama) göre çok fazla kavramın yer almasıdır. M33 kodlu katılımcı bu durumu aşağıdaki şekilde ifade etmiştir:

Benim NYP dersinde zorlanmamdaki en büyük sebep nesne yönelimli programlama dersinin içeriğinin görsel programlamaya göre çok daha fazla ayrıntıları olmasıdır. Dersin içeriğinde nesne, sınıf, kalıtım gibi öğrenmem gereken çok fazla şey olduğu için dersin başlarında çok zorlandım. Görsel programlama dersinden nesne yönelimli programlamaya ilk geçtiğimiz zamanlarda bu kavramlar çok karışık ve soyut gelmişti. Bunların birbiriyle ilişkilerini ve kullanımlarını anlamam zaman aldı. Fakat bir kere oturtunca programlamayı ne kadar kolaylaştırdığını gördüm.

Dersin işlenişi ile ilgili olarak katılımcılar, programlama dersini ilk defa alan öğrencilerin dersin işleniş hızından dolayı sorun yaşadığını ve bu öğrencilerin dersi daha önce programlama dersi alan öğrencilerle aynı hızda takip etmek zorunda kaldıklarını ifade etmişlerdir. Bunun yanında katılımcılardan bazıları dersi ezber mantığı ile öğrenmek zorunda kaldıklarını, bunun da öğrenmeleri açısından bir problem teşkil ettiğini bildirmişlerdir. Bu konu ile ilgili katılımcı görüşlerinden bazıları aşağıda verilmiştir.

Ders bence çok hızlı işleniyor. Daha önceden bilgisayar ile aşina olan arkadaşlarımız dersi daha iyi yapabiliyor iken ben daha zor anlıyorum. Dersin daha basit düzeyde anlatılması daha iyi olabilirdi. (M19)

Bu derslerde öğrendiğim bilgiler programlama bilgisi açısından bana katkısı olduğunu düşünüyorum. Fakat kısa zamanda ve yoğun bir tempoda çok konu ve çok kod gördüğümüzü düşünüyorum. Bu durum da öğrencileri ister istemez zorluyor. (M32)

Değerlendirme yöntemi ile ilgili olarak bazı katılımcılar, derste uygulanan sınav sisteminin kendilerini ezber yöntemine yönlendirdiğini; bu durumun da bilgileri öğrenilmesini engellediğini ve kalıcılık için bir tehdit olduğunu ifade etmişlerdir. Bunun yanında bazı öğrenciler dönem sonu notlarında sınavların ağırlığının çok yüksek olduğunu, buna zıt olarak ödev ve projelerin ağırlığının çok düşük olduğunu belirtmişleridir. Bu konu ile ilgili katılımcı görüşlerinden bazıları aşağıda verilmiştir.

Üzülerek söylüyorum ki derste öğrendiklerimiz kalıcı değil. Tüm öğrendiklerimizi unutuyoruz çünkü ezber yaparak sınava çalışıyoruz. (M3)

Bence burada öğrenmiş olduğumuz programlama dillerini gerçekten öğrenilmesini istiyorsanız proje üzerinden notlandırma olmalı ve öğrencilerin sınırları zorlanmalı. Bu sayede hem bölümümüz hem de öğrenciler başarıyı elde etmiş olur. Oysa biz sınav için çalışıyoruz. Proje ve ödev notları düşük olduğu için sınav daha önemli hale geliyor. (M41)

Katılımcılar, öğretimle ilişkili sorunların yanında teknolojik altyapı ve ders materyalleri ile ilgili de sorun yaşadıklarını ifade etmişlerdir. Bu sorunların temelinde okul bilgisayarlarının donanım özelliklerinin güncel olmaması, öğrencilerin sınırlı bir internet bağlantısı olması ve dersler ilgili yeterli kaynağın olmaması yer almaktadır. Bu konu ile ilgili katılımcı görüşlerinden bazıları aşağıda verilmiştir.

Dersin oldukça yararlı olduğunu düşünüyorum fakat okulumuzda internet bağlantımız kısıtlı olduğu için öğrenme kapasitemiz doğal olarak azalıyor. İnternetten faydalanabilirsek eğer daha çok öğrenebiliriz. (M6)

Derste geliştirdiğimiz örnek projelere benzeyen, detaylı açıklamalar içeren video derslere ihtiyacımız olduğunu düşünüyorum. Ne kadar çok örnek proje görürsek o kadar çok konular akılda kalır. (M49)

Öğrencilerin yukarıda açıklanan sorunlarının yanında, *ders etkinlikleri* alt kategorisinde de bazı sorunları bulunmaktadır. Bu sorunların kodları "Günlük hayatla ilişkili projeler yapmıyoruz.", "Görsel olarak çekici projeler üretmiyoruz.", "Keyifli projeler gerçekleştirmiyoruz.", "Projeler yaratıcı düşünmeyi teşvik etmemektedir." şeklindedir. Bu kodlara ait açıklamalar Bölüm 4.1.1.2.1'de daha önce sunulduğu için bu bölümde detaya girilmemiştir.

4.1.1.3.2. Mevcut NYP dersinde öğretim elemanlarının yaşadıkları sorunların değerlendirilmesi

Öğretim elemanlarının mevcut NYP dersinde yaşadıkları sorunların belirlenmesi amacıyla NYP dersine giren beş öğretim elemanı ile yarı yapılandırılmış görüşmeler yapılmıştır. Elde edilen verilerin çözümlenmesi sonucunda oluşturulan kategori, frekans ve yüzde bilgileri Tablo 4.6'da sunulmuştur.

Tablo 4.6'ya göre öğretim elemanlarının yaşadıkları ilk sorun kategorisi psikolojik faktörlerle ilgilidir. Öğretim elemanlarının tamamı bazı öğrencilerin derse karşı isteksizlik duyduğunu, bazı öğrencilerin dersi faydalı bulmadıklarını ve yeteri kadar uygulama yapmadıklarını belirtmektedir. Bu konu ile ilgili ÖE2 kodlu katılımcının görüşleri aşağıdaki gibidir:

Genel anlamda birçok öğrenci derse karşı ilgili başlıyor. Ancak aradan birkaç hafta geçtikten sonra dersi yapabilen öğrenciler derse karşı motivasyonlarını korurken yapamayanlar kopuyor ve bu benim ne işime yarayacak sorularını sormaya başlıyorlar. Bazı öğrenciler ise sadece dersi geçmek için çalışıyorlar. Öğrencilerin düşündükleri tek şey bu dersten nasıl geçerim. Programı nasıl yaparım çalıştırırım tamamlarım soruları yerine bu dersten nasıl geçerim sorularıyla meşguller.

Tablo 4.6. Öğretim elemanlarının mevcut NYP dersinde yaşadıkları sorunlara ilişkin görüşleri*

Tema	Kategori	Kodlar	Frekans (f)	Yüzde (%)
Öğretim Elemanlarının Mevcut NYP Dersinde Yaşadıkları Sorunların Değerlendirilmesi	Psikolojik Faktörler	Dersi yararlı bulmadıkları için bazı öğrencilerin motivasyonları düşüktür.	5	100
		Bazı öğrenciler sadece sınava yönelik çalışma eğilimi göstermektedir.	3	60
		Bazı öğrenciler programlama konusunda kendilerine güvenmemektedir.	3	60
		Bazı öğrencilerin programlama konusundaki kaygısı yüksektir.	2	40
	Öğretim Programı	Bazı öğrenciler NYP dersinin içeriğini anlamada ve yazılım geliştirirken kullanmada güçlük yaşamaktadır.	5	100
	Öğrencilerin Hazırbulunuşluk Seviyeleri	Bazı öğrencilerin NYP dersi için programlama hazırbulunuşluk seviyeleri yeterli düzeyde değildir.	3	60
		Bazı öğrencilerin matematiksel bilgi ve algoritmik düşünme seviyeleri yeterli düzeyde değildir.	3	60
		Bazı öğrencilerin yabancı dil bilgisi yeterli seviyede değildir.	2	40
	Ders Etkinlikleri	Derste yapılan projeler bazı öğrencilerin ilgisini çekmekte yetersiz kalmaktadır.	5	100
	Teknolojik Altyapı	İnternete bağlanmada sorunlar bulunmaktadır.	5	100
		Kullanılan bilgisayar donanımları güncel değildir.	4	80

Öğretim elemanlarının sorun yaşadığı ikinci kategori bazı *öğrencilerin NYP dersinin içeriğini anlama ve kullanmada güçlük yaşamalarıyla* ilgilidir. Öğretim elemanlarının tamamına göre bazı öğrenciler NYP dersi içeriğinde yer alan sınıf, nesne, kapsülleme, kalıtım ve çokbiçimlilik gibi konuları anlamada ve bunları bir yazılım projesi geliştirirken kullanmada güçlük yaşamaktadır. Bu konu ile ÖE1 kodlu katılımcı görüşlerini şu şekilde ifade etmiştir:

> Dersin ilk haftalarında sınıf ve nesne kavramlarını anlamak bazı öğrenciler için güç olabilmektedir. Bu öğrenciler sınıfın ne olduğunu ve neden tanımlandığını; nesnenin ne olduğunu ve o sınıftan nasıl nesne oluşturulduğunu; oluşturulan nesne üzerinden metotlar yoluyla nasıl işlemler yaptırıldığını kavramakta zorlanmaktadır. Bazen davranış, özellik, kurucu metot kavramları bile anlaşılmamaktadır. En basit örneğiyle tanımlanan kare sınıfının özellikleri olan bir kenar, alan ve çevre bile bazı öğrenciler tarafından nesne gibi algılanabilmektedir. Yani sınıfın özellikleri ile nesne kavramı bile karıştırılabilmektedir. Dersin ilerleyen haftalarında yer alan kalıtım ve çokbiçimlilik konularının anlaşılması ve kullanılması da zorlayıcı olabilmektedir. Öğrencilerin zorluklarının temel nedenini NYP'de çok fazla kavramın yer almasına bağlıyorum. Yazılımları NYP kullanarak geliştirmek belirli bir çaba ve zaman gerektiren bir süreç diye düşünüyorum.

* Katılımcı sayısı: 5. Bir katılımcının vermiş olduğu cevap birden fazla kod içerebilir.

Öğretim elemanlarının sorun yaşadığı üçüncü kategori *öğrencilerin hazırbulunuşluk seviyeleri* ile ilgilidir. Öğretim elemanlarından bazıları, bazı öğrencilerin temel programlama konularında bilgilerinin eksik olması, bunun yanında matematik ve yabancı dil konularında seviyelerinin istenilen düzeyde olmaması nedeniyle derste zorlandıklarını belirtmektedir. Bu konu ile ilgili katılımcı görüşlerinden bazıları aşağıda verilmiştir.

Öğrencilerimiz NYP dersinden önce Algoritma ve Programlamaya Giriş dersi ile Görsel Programlama dersi alıyorlar. Normal şartlarda değişken, döngü, diziler, listeler, metotlar gibi konularda sıkıntı yaşamamaları beklenir. Ancak hem bu konularda hem de listbox, combobox gibi form elemanlarının kullanımında sıkıntı yaşayan öğrenciler bulunuyor. Tabii bu durum NYP dersini yavaşlatıyor. Eksik konulara değinmek zorunda kalıyoruz. (ÖE5)

…Bazı öğrenciler karşılaştıkları hataları anlamakta güçlük çekiyorlar. Bunda İngilizce bilgi seviyelerinin düşük olması da bir etken. (ÖE3)

Bir diğer sorun kategorisi olan *ders etkinlikleri* kategorisi, derste gerçekleştirilen nesne yönelimli programlama projelerinin yapısıyla ilgilidir. Bazı öğretim elemanları ders içi uygulamaların öğrencilerin ilgilerini çekmediğini, bununla birlikte kendilerini ifade edebilecekleri ve yeteneklerini gösterebilecekleri uygulamaların sınırlı sayıda olduğunu belirtmişlerdir. Bu konu ile ilgili ÖE6 kodlu katılımcının görüşleri aşağıda verilmiştir:

Örneğin ne uygulaması yapıyoruz, kare, dikdörtgen, üçgen gibi geometrik sınıfları içeren ve bunların alan ve çevrelerini sınıf ve nesnelerle hesaplayan bir form uygulaması. Bunu başlangıç seviyesi bir uygulama olarak düşünelim. Öğrenciler diyorlar ki biz bu uygulamayı ileride nerede kullanacağız, ne işimize yarayacak. Bu durumda da derse yapılan bazı uygulamalar öğrencilerimizin dikkatini çekmiyor. Çünkü yaptığımız bazı uygulamalar öğrencilerin hiçbir zaman kullanmayacağı şeyler oluyor.

Son sorun kategorisi olan *teknolojik altyapı*, okulun internet bağlantısının sınırlı olması ve bilgisayar dershanelerinde yer alan bilgisayarların donanım özelliklerinin güncel olmamasıyla ilgilidir. Öğretim elemanlarından bazıları bu sorunların dersin işleniş hızını, verimliliğini etkilediğini belirtmektedir. Bunun yanında özellikle internet bağlantısı sınırlı olması sebebiyle öğrencilerin kaynak tarama ve örnek proje inceleme olanaklarının sınırlı olduğunu belirtmişlerdir.

4.1.1.4. Araştırma Sorusu 3: Öğrenciler mevcut NYP dersinde ne tür programlama uygulamalarını geliştirmenin kendileri için daha faydalı olacağını değerlendirmektedir?

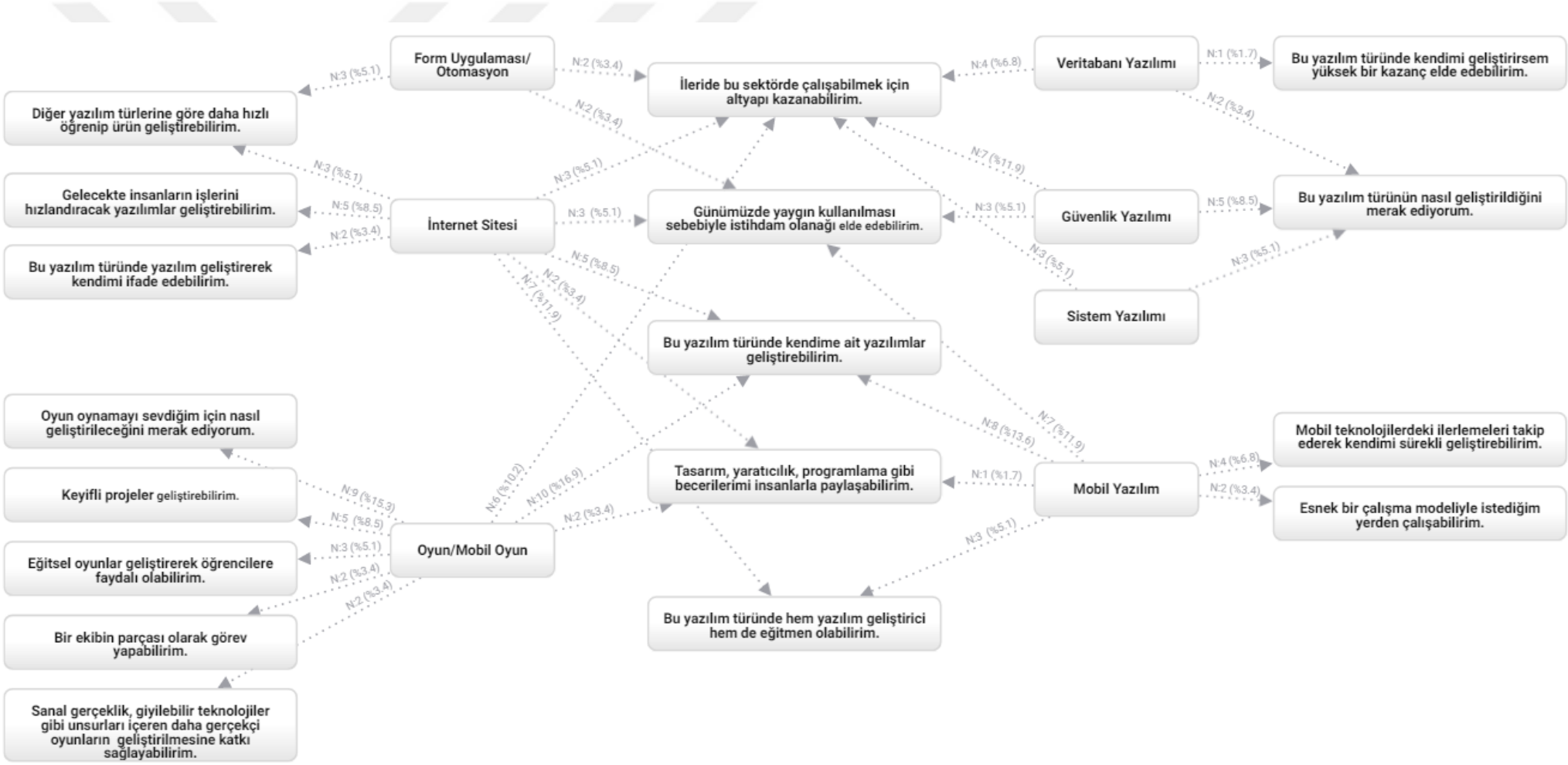
Katılımcıların mevcut NYP dersinden beklentilerinin değerlendirilmesi kapsamında katılımcıların derste geliştirmek istedikleri yazılım türleri belirlenmiştir. Katılımcıların geliştirmek istedikleri yazılım türlerine yönelik görüşleri elde edilirken "*NYP dersinde size*

en çok fayda sağlayacağını düşündüğünüz sadece bir yazılım türünde yazılım geliştirecek olsanız hangisini tercih edersiniz? Neden?" sorusu yöneltilmiştir. Verilerin çözümlenmesi ile elde edilen betimsel analiz sonuçları Tablo 4.7'de sunulmuştur. Tabloya göre, *oyun/mobil oyunlar, internet siteleri* ve *mobil yazılımlar* mezun öğrencilerin NYP dersinde en çok geliştirmek istedikleri yazılım türleri arasındadır.

Tablo 4.7. Mezun öğrencilerin geliştirmek istedikleri yazılım türlerine ilişkin betimsel analizler

	Frekans (f)	**Yüzde (%)**
Oyun/Mobil Oyun	15	25
İnternet Sitesi	11	17
Mobil Yazılım	9	15
Güvenlik Yazılımı	7	12
Form Uygulaması/Otomasyon	7	12
Sistem Yazılımı	6	10
Veritabanı Yazılımı	4	8
Toplam Görüş Sayısı	**59**	**100**

Mezun öğrencilerin ilgili yazılım türünü neden geliştirmek istedikleri de elde edilen verilerin çözümlenmesi ile değerlendirilmiş olup, katılımcıların verdiği cevaplara göre oluşturulan kategoriler ve kodlar şematize edilerek Şekil 4.6'da sunulmuştur. Şekle göre öğrencilerin belirttikleri yazılım türlerini tercih etmelerinde; (1) "*Bu yazılım türünde kendime ait yazılımlar geliştirebilirim*.", (2) "*İleride bu sektörde çalışabilmek için altyapı kazanabilirim*.", (3) "*Günümüzde yaygın kullanılması sebebiyle istihdam olanağı elde edebilirim*.", (4) "*Bu yazılım türünde kendimi geliştirirsem yüksek bir kazanç elde edebilirim*.", (5) "*Bu yazılım türünde hem geliştirici hem de eğitmen olabilirim*." ve (6) "*Tasarım, yaratıcılık, programlama gibi becerilerimi insanlarla paylaşabilirim*." ortak nedenler etkili olmuştur. Örnek vermek gerekirse; oyun/mobil oyun yazılım türünü tercih etmelerinin nedenini "*Bu yazılım türünde kendime ait yazılımlar geliştirebilirim.*" olarak belirten katılımcılar (N:10) bulunurken, mobil yazılım (N:8) veya web sitesi (N: 5) geliştirmek isteyen katılımcılar da aynı sebeple derste bu yazılım türlerini geliştirmek istemektedir. Ortak olan kodlara ait öğrenci alıntılarından bazı örnekler Tablo 4.8'de sunulmuştur.



Şekil 4.6. Öğrencilerin mevcut NYP dersinde geliştirmek istedikleri yazılım türlerinin nedenlerine ilişkin görüşleri

Tablo 4.8. Mezun öğrencilerin geliştirmek istedikleri yazılım türlerine ilişkin ortak kodlara yönelik örnek alıntılar

Kod	Kategori	Örnek Alıntı
İleride bu sektörde çalışabilmek için altyapı kazanabilirim.	Oyun/Mobil Oyun	*M14: Günümüzde oyun sektörünün bu kadar gelişmesi ve benim oyun oynamayı çok sevmem nedeniyle oyunları geliştirmeyi öğrenmek isterdim. Oyunları boş zaman geçirilecek bir şey değil profesyonel bir meslek olarak görüyorum. Gelecekte bu alanda çalışmak istediğim için NYP dersi bu konuda altyapı oluşturmak için yardımcı olabilir.*
Günümüzde yaygın kullanılması sebebiyle istihdam olanağı elde edebilirim.	Güvenlik Yazılımı	*M21: Güvenlik yazılımları geliştirmek isterim çünkü zaman ilerledikçe devir tamamen teknoloji üzerinden ilerliyor. Gelecekte şirketler, iş insanları, kurumlar için yeteri kadar iyi güvenlik yazılımı yazabilirsem hem aranan insan olurum hem de güzel kazançlar elde edebilirim."*
Bu yazılım türünde kendime ait yazılımlar geliştirebilirim.	Mobil Yazılım	*M8: Günümüzde akıllı telefonlar herkesin elinde. Bu yüzden hem insanlara ulaşmak, hem de reklam almak için mobil yazılım türünü tercih ederdim. Bu sayede kendime ait bir yazılımın olurdu.*
Tasarım, yaratıcılık, programlama gibi becerilerimi insanlarla paylaşabilirim.	İnternet Sitesi	*M40: Günlük yaşantımızda web siteleri ile sürekli iç içeyiz. Ben de programlama ve tasarım becerilerimi gösterebileceğim yenilikçi internet sitesi projelere katılmak isterdim.*
Bu yazılım türünün nasıl geliştirildiğini merak ediyorum.	Sistem Yazılımı	*M11: Geleceğin dünyasında bilgisayarlar ve sistemler önemli bir rol oynayacak. Ben sistem yazılımları geliştirmek isterdim çünkü bu yazılım türünün nasıl geliştirildiğini merak ediyorum.*
Bu yazılım türünde hem yazılım geliştirici hem de eğitmen olabilirim.	Mobil Yazılım	*M58: Günümüzde hepimiz akıllı cihazları kullanıyoruz. Tüm işlerimizi oradan hallediyoruz. Bu alanda program ihtiyacı artarken bunu öğrenmek isteyenlerin sayısı da artıyor. Ben derste bu kişilere eğitim verebilecek altyapıyı kazanmak isterdim. Böylece bu tür yazılımların nasıl geliştirileceği konusunda eğitmen olabilirdim.*
Diğer yazılım türlerine göre daha hızlı öğrenip ürün geliştirebilirim.	Form Uygulaması/ Otomasyon	*M44: Okulda aldığımız C# eğitimi form uygulamaları geliştirme üzerine. Birinci sınıfta aldığımız iki programlama dersi de bu proje türünde. NYP dersinde de form uygulamaları geliştirmek isterdim çünkü çok fazla zorlanmadan daha hızlı proje geliştirirdim.*

Ortak kodlara ilave olarak, katılımcılar belirtilen yazılım türüne özgü sebeplerle de tercihte bulunmuşlardır. Örneğin NYP dersinde mobil yazılım geliştirip bu yazılım türünün nasıl geliştirilebileceği hakkında bilgi sahibi olmak isteyen katılımcılardan bazıları (N:4) bunun nedenini "*Mobil teknolojilerdeki ilerlemeleri takip ederek kendimi sürekli geliştirebilirim.*" şeklinde fikir beyan ederken bazı katılımcılar ise (N:2) mobil yazılım geliştiricisi olarak istihdam edilmeleri durumunda "*Esnek bir çalışma modeliyle istediğim yerden çalışabilirim.*" şeklinde fikir belirtmiştir. Katılımcıların ders kapsamında geliştirmek istedikleri yazılım türlerine ait ortak olmayan kodlara yönelik örnek alıntılar Tablo 4.9'da sunulmuştur.

Tablo 4.9. Mezun öğrencilerin geliştirmek istedikleri yazılım türlerine ilişkin ortak olmayan kodlara yönelik örnek alıntılar

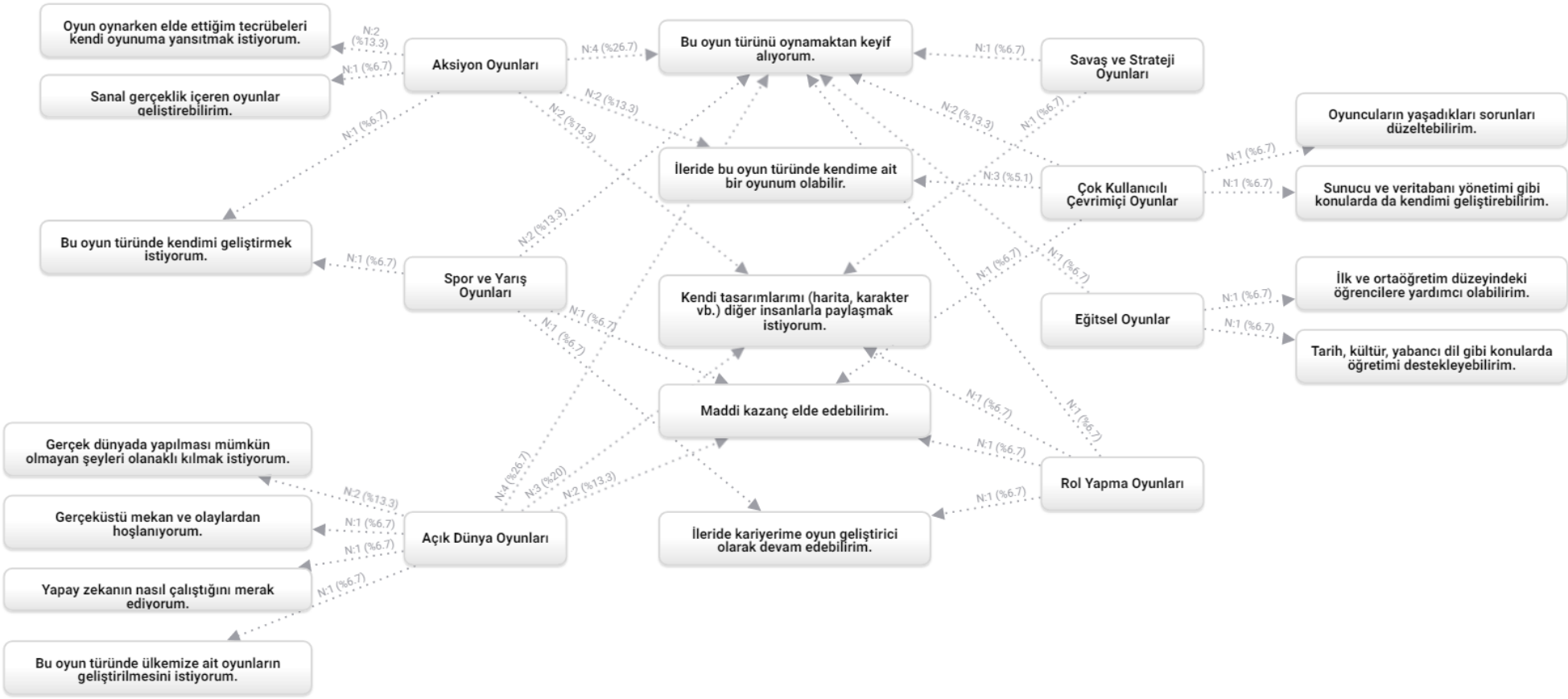
Kategori	Kod	Örnek Alıntı
Oyun/Mobil Oyun	Oyun oynamayı sevdiğim için nasıl geliştirileceğini merak ediyorum.	*M45: Mobil ve oyun geliştirmek isterim çünkü her boş anımda telefon kullanırım, telefondan da anlarım, oyun oynamayı severim ve her ayrıntısına kadar öğrenmek isterim.*
	Keyifli projeler geliştirebilirim.	*M6: Oyun vb. uygulamalar geliştirmeyi isterdim. Yaş grubumca ve çağımız bilgisayar çağı yani oyun çağı olduğundan bu yönde bir gelişim daha eğlenceli olacaktır. Yaptığım projelerden keyif alırdım.*
	Eğitsel oyunlar geliştirerek öğrencilere faydalı olabilirim.	*M29: Daha çok oyunlar geliştirmek isterdim çünkü çocuklar adına daha yararlı, daha zekâ geliştirici ve eğitici, onları geleceğe hazırlayabilmek adına faydalı işler yapmış olurdum.*
	Bir ekibin parçası olarak görev yapabilirim.	*M36: ...Ben yalnız çalışmak yerine ekiple çalışmaktan hoşlanıyorum. İnsanlarla etkileşim içinde çalışabileceğim proje türlerini tercih ederim. Oyun geliştirme genelde tek başına başarılması zor bir süreç olduğu için ekip ruhu ve çalışması gerektiriyor. Ben de böyle bir ekipte yer almak isterdim.*
	Sanal gerçeklik, giyilebilir teknolojiler gibi unsurları içeren daha gerçekçi oyunların geliştirilmesine katkı sağlayabilirim.	*M49: Oyunu daha gerçekçi yapıp giyilebilir kıyafetlerle hisleri, acıları hissetmek isterim. Oyunda vurulduğumuz zaman o bölgede acı hissetmek oyunun gerçekliğini yükseltir. Bunu askeri açıdan da kullanabiliriz. Riske girmeden ve maliyeti düşürerek eğitimler verilebilir.*
Mobil Yazılım	Mobil teknolojilerdeki ilerlemeleri takip ederek kendimi sürekli geliştirebilirim.	*M2: ...Mobil teknolojiler sürekli gelişim içinde. Ben de bu alandaki gelişmeleri yakından takip etmek ve kendimi güncellemek için bu tür yazılımları tercih ederdim. Yeni şeyler öğrenmek, bunları da uygulamak benim için keyifli olmuştur her zaman.*
	Esnek bir çalışma modeliyle istediğim yerden çalışabilirim.	*M33: Meslek hayatımda sürekli bir yere gidip orada belirli saatlerde çalışmayı düşlemiyorum. Daha çok kendimi projeye adayıp onunla istediğim yerde ve istediğim kadar uğraşmayı seviyorum. Bu imkâna mobil yazılımlar geliştirerek kavuşabileceğimi düşünüyorum. Evden çalışıp yazılım ekibindeki görevlerimi başarmak isterdim.*
İnternet Sitesi	Gelecekte insanların işlerini hızlandıracak yazılımlar geliştirebilirim.	*M25:Ben oyunlardan ziyade günlük hayatı kolaylaştıracak ve işlerini hızlandıracak bir şeylerin yazılımını yapmak istiyorum.*
	Bu yazılım türünde yazılım geliştirerek kendimi ifade edebilirim.	*M38: Bizim de kendi başımıza bir şeyler başardığımızı gösterebileceğimiz tarzda internet siteleri geliştirmek isterdim. Bu sayede hem kendimi ifade ederdim hem de sevdiklerime ürünümü gösterirdim.*
Veritabanı Yazılımı	Bu yazılım türünde kendimi geliştirirsem yüksek bir kazanç elde edebilirim.	*M17: Günümüzde verilerin saklanması, güvenliği ve analizi önemli hale geldi. Bu nedenle bu sektörde çalışacak yetenekli insanlara ihtiyaç bulunuyor. Ben de bu sektörde kendimi geliştirip yüksek bir gelir elde edebileceğime inanıyorum. O yüzden NYP dersinde veritabanı yazılımları geliştirmek isterdim.*

Tüm katılımcıların NYP dersinde geliştirmek istedikleri yazılım türü ve bunun nedenleri ile ilgili veri toplama işlemi gerçekleştirildikten sonra, oyun/mobil oyun geliştirmek isteyen 15 katılımcıya "*NYP dersinde geliştirmek istediğiniz oyun türü için tek bir tür seçecek olsaydınız hangi türü seçerdiniz? Neden?"* sorusu yöneltilmiştir. Verilerin çözümlenmesi ile elde edilen betimsel analiz sonuçları Tablo 4.10'da sunulmuştur. Tabloya göre, bir oyuncunun sanal bir dünyayı serbestçe dolaştığı *açık dünya oyunları* (örn. GTA, Grand Theft Auto) ile *aksiyon oyunları* (örn. Call of Duty, MineCraft, Rust) oyun/mobil oyun geliştirmek isteyen mezun öğrencilerin en çok tercih ettikleri oyun türleri arasındadır.

Tablo 4.10. Oyun/mobil oyunları geliştirmek isteyen mezun öğrencilerin tercih ettikleri dijital oyun türlerine ilişkin betimsel analizler

	Frekans (f)	**Yüzde (%)**
Aksiyon Oyunları	4	27
Açık Dünya Oyunları	4	27
Spor ve Yarış Oyunları	2	13
Çok Kullanıcılı Çevrimiçi Oyunlar	2	13
Eğitsel Oyunlar	1	7
Savaş ve Strateji Oyunları	1	7
Rol Yapma Oyunları	1	7
Toplam Görüş Sayısı	**15**	**100**

İlgili oyun türünü neden geliştirmek istedikleri de elde edilen verilerin çözümlenmesi ile değerlendirilmiş olup, katılımcıların verdiği cevaplara göre oluşturulan kategoriler ve kodlar şematize edilerek Şekil 4.7'de sunulmuştur. Şekle göre öğrencilerin belirttikleri oyun türlerini tercih etmelerinde; (1) "*Bu oyun türünü oynamaktan keyif alıyorum*.", (2) "*İleride bu oyun türüne ait bir oyunum olabilir*.", (3) "*Kendi tasarımlarımı (harita, karakter vb.) diğer insanlarla paylaşmak istiyorum*.", (4) "*Maddi kazanç elde edebilirim*." ve (5) "*İleride kariyerime oyun geliştirici olarak devam edebilirim.*" ortak nedenler etkili olmuştur. Ortak olan kodlara ait öğrenci alıntılarından bazı örnekler Tablo 4.11'de sunulmuştur.



Şekil 4.7. Mevcut NYP dersinde oyun/mobil oyun geliştirmek isteyen öğrencilerin bu oyun türlerini tercih etmelerine ilişkin görüşleri

Tablo 4.11. Mevcut NYP dersinde oyun/mobil oyun geliştirmek isteyen mezun öğrencilerin bu oyun türlerini tercih etmelerine ilişkin ortak kodlara yönelik örnek alıntılar

Kod	Kategori	Örnek Alıntı
Bu oyun türünü oynamaktan keyif alıyorum.	Açık Dünya Oyunları	*M14: Ben GTA serisinde olduğu gibi açık dünya oyunu geliştirmek isterdim. Çünkü herhangi bir kural olmadan oynanan oyunları çok severim. Gerçek dünyada yapamadığım legal/illegal her şeyi bu dünyada yapardım. İstediğim hayatı yaşardım.*
İleride bu oyun türünde kendime ait bir oyunum olabilir.	Aksiyon Oyunları	*M2: ...Kendime ait bir aksiyon tarzı oyunum olsun isterdim. Bunu tamamen kendi gayretlerimle başarmak isterdim. Oyuna kendimi temsil eden karakterler ekleyip yakınlarımın benimle gurur duymalarını isterdim.*
Kendi tasarımlarımı (harita, karakter vb.) diğer insanlarla paylaşmak istiyorum.	Savaş ve Strateji Oyunları	*M7: Ben strateji türündeki oyunları severim. Haritaları ve tarihi çok seven birisi olarak eski devletlerin bulunduğu bölgelerin haritalarını içinde barındıran, kendi tasarladığım haritalarda geçen oyunlar tasarlamak ve bunu başkalarıyla paylaşmak isterim.*
Maddi kazanç elde edebilirim.	Açık Dünya Oyunları	*M14: ...Derste oyun yapmayı öğrenip, hatta yapıp, gerekli platformlarda bu oyunların satışını gerçekleştirebilirsek hem derse karşı olan ilgimiz artar hem de yapabileceğimiz ekstra bir uğraşı [yeteneği] envanterimize eklemiş oluruz. Böyle olunca bunaltıcı günlerin sonunda hedefine bir adım daha yaklaşmanın verdiği azimle birkaç satır daha kod yazmak için can atacağımızdan eminim.*
İleride kariyerime oyun geliştirici olarak devam edebilirim.	Spor ve Yarış Oyunları	*M24: ...Bir bilgisayarcı olarak hem merakımı giderecek hem de kendi kariyer planlarım için bu tür bir oyun geliştirmek isterdim. Açık dünya ya da FPS oyunlarında rekabet çok olduğu için kariyerime bu tür oyun geliştirerek devam etmeyi düşünürdüm. Bu sayede hem mesleki çalışmalarımdan keyif alıp hem de para kazanırdım.*
Bu oyun türünde kendimi geliştirmek istiyorum.	Aksiyon Oyunları	*M18: Programlama derslerinde aklımdan geçen aslında en çok bilgisayar oyunları üzerine yapılan kodlamalardır. Çünkü küçüklüğümden beri ben bilgisayar oyunlarıyla hep iç içe olmuşumdur. Özellikle FPS tarzı oyunların nasıl yapıldığını çocukluktan beri düşünürüm. Lisenin sonlarına doğru bu konuya merakım iyice arttı. Bu merakla hep oyun tasarlamayı ve yapmayı istedim. Zevkli olmasının yanında bir o kadar da zahmetli olacağını düşündüm hep. Oyunun içinde binlerce fonksiyon var ve hepsi birbiriyle bağlantılı. Bunun zor olması içimdeki ateşi hiç dindirmedi. Ben bu karmaşıklığı hiç düşünmeden bu oyun türünde kendimi geliştirmek ve iyi şeyler başarmak istiyorum.*

Ortak kodlara ilave olarak, katılımcılar belirtilen oyun türüne özgü sebeplerle de tercihte bulunmuşlardır. Örneğin aksiyon oyununu tercih eden katılımcılardan bazıları (N:2) bunun nedenini "*Oyun oynarken elde ettiğim tecrübeleri kendi oyunuma yansıtmak istiyorum*." şeklinde fikir beyan ederken bazı katılımcılar ise (N:1) bu oyun türünün nasıl geliştirildiğini öğrendiği zaman "*Sanal gerçeklik içeren oyunlar geliştirebilirim.*" şeklinde fikir belirtmiştir. Oyun/mobil oyun geliştirmek isteyen katılımcıların tercih ettikleri oyun türlerine ait ortak olmayan kodlara yönelik örnek alıntılar Tablo 4.12'de sunulmuştur.

Tablo 4.12. Oyun/mobil oyun geliştirmek isteyen mezun öğrencilerin bu oyun türlerini tercih etmelerine ilişkin ortak olmayan kodlara yönelik örnek alıntılar

Kategori	Kod	Örnek Alıntı
Açık Dünya Oyunları	Gerçek dünyada yapılması mümkün olmayan şeyleri olanaklı kılmak istiyorum.	*M36: …Hayatımızda belki de hiçbir zaman sahip olamayacağımız imkanlara sahip olmayı sağlayabilirdim. Örneğin hiç gidemeyeceğimiz bir şehre sanal ortamda gitmek ya da hiçbir zaman binemeyeceğimiz bir arabaya binip kullanmak gibi.*
	Gerçeküstü mekan ve olaylardan hoşlanıyorum.	*M14: Ben bir oyun tasarlayacak olsam kesinlikle aşırı mitolojik bir evrende geçmesini isterdim. Tabii bu oyun açık dünya tarzında olurdu. Oyunda hiç kimsenin hayal bile edemeyeceği mekân ve kişiler olurdu.*
	Yapay zekânın nasıl çalıştığını merak ediyorum.	*M36: …Ayrıca bu tür oyunlarda [açık dünya oyunlarında] yapay zekâ ile çalışmak isterdim. NCO'ların nasıl zorluklar çıkaracağını, nasıl araba kullanacağını, nasıl yürüyeceğini programlamayı öğrenmek isterdim.*
	Bu oyun türünde ülkemize ait oyun geliştirilmesini istiyorum.	*M41: Bir açık dünya tarzı oyun geliştirmek isterdim. Ülkemiz ilgili oyun konusuna yabancı. Herkes bu oyunları bilgisayarına sadece oynamak için indiriyor. Ben bu türde bir oyun geliştirilmesine katkıda bulunmak isterdim. Hem ülkemize maddi açıdan gelir sağlamak hem de ülkemizin açık dünya türünde parmakla gösterilecek olmasını isterim.*
Aksiyon Oyunları	Oyun oynarken elde ettiğim tecrübeleri kendi oyunuma yansıtmak istiyorum.	*M2: Ben "survival [hayatta kalma]" tarzı bir oyun geliştirirdim. Küçüklüğümden beri oynadığım bütün survival oyunlarını bir araya getirerek, onlardan elde ettiğim tecrübeleri bu tarzda bir oyuna yansıtırdım.*
	Sanal gerçeklik içeren oyunlar geliştirebilirim.	*M49: Bir oyun yapacak olsaydım bu FPS tarzı olurdu ve meta verse [sanal gerçeklik içeren] bir ortamda geçerdi. Orada oyun deneyimi daha gerçekçi olurdu.*
Çok Kullanıcılı Çevrimiçi Oyunlar	Oyuncuların yaşadıkları sorunları düzeltebilirim.	*M13: Ben 6-7 yıl Metin2 oyununu oynadım. Bu oyunu çok sevdim ancak oyunda pekçok sıkıntı yaşadım. Zaten oyun da şu anki yenilikler açısından çok geride kaldı. Ben bu tarzda bir oyun geliştirip yaşadığım sıkıntıları düzeltmeye çalışırdım. Bu sayede oyun kendime ve kullanıcılara daha uygun hale gelebilir.*
	Sunucu ve veritabanı yönetimi gibi konularda da kendimi geliştirebilirim.	*M57: …Bu tür bir oyun yapmayı öğrenmek bana birçok konuda bilgi sağlardı. Örneğin üç boyutlu grafik oyun motorlarını kullanmayı öğrenmek, online olduğu için sunucular ve veritabanı gibi konularda gelişmemi sağlardı. Zor ve komplike bir kodlama yapmak basit şeyler öğrenmekten çok daha iyi olacaktır.*
Eğitsel Oyunlar	İlk ve orta öğretim düzeyindeki öğrencilere yardımcı olabilirim.	*M29: …Geliştirdiğim oyunlar hem üniversite öncesi öğrencilerin derslerine yardımcı olur hem de eğlenerek öğrenmelerine sağlar.*
	Tarih, kültür, yabancı dil gibi konularda öğretimi destekleyebilirim.	*M29: Ben eğitsel bir oyun geliştirmek isterdim. Mesela Türk tarihi ve kültürünün anlatılması konusunda. Ya da yabancı dil eğitimi ile ilgili. Programlama derslerinde yabancı dil konusunda oldukça zorlandım. Eğer küçükken ben bu tür bir oyun oynamış olsaydım belki de dil konusunda bu kadar çok zorlanmayacaktım ve bilgisayarda yabancı dilde bir şeyler yazdığında ne demek istediğini anlayacaktım.*

4.1.1.5. Araştırma Sorusu 4: Mezun öğrencilerin mevcut NYP dersinde oyun geliştirmeye dayalı öğretimin uygulanabilirliği konusundaki değerlendirmeleri nelerdir?

Mezun öğrencilerden NYP dersinde oyun geliştirmeye dayalı programlama öğretimi yapılması durumunda ortaya çıkacak olası avantaj ve dezavantajları değerlendirmeleri istenmiştir. Başka bir ifade ile mevcut NYP dersinde oyun geliştirmeye dayalı öğretimin ne tür faydalar sağlayacağı ve ne tür mahsurları olacağı konusunda fikirlerini belirtmeleri istenmiştir. Mezunların büyük bir çoğunluğunun (N:52) NYP dersinde oyun geliştirmeye dayalı programlama öğretiminin öğrenenler açısından avantajlar sağlayacağı yönünde fikir beyan ettikleri, az sayıda katılımcının (N:7) ise dezavantajlı olabileceği konusunda görüş bildirdikleri görülmüştür. Mezun öğrencilerin değerlendirmelerine ait kategori, frekans ve yüzde bilgileri Tablo 4.13'te sunulmuştur.

Tablo 4.13. Mezun öğrencilerin NYP dersinde oyun geliştirmeye dayalı öğretimin uygulanabirliğine yönelik görüşleri*

Tema	Kategori	Kodlar	Frekans (f)	Yüzde (%)
Mevcut NYP Dersinde Oyun Geliştirmeye Dayalı Öğretimin Uygulanabirliğinin Değerlendirilmesi	Dezavantajlar	Zorlayıcı olur.	7	12
		Sıkıcı olur.	4	7
		Yararlı olmaz.	1	2
	Avantajlar	Yararlı olur.	52	88
		Dersin verimliliğini artırır.	21	36
		Motivasyonu artırır.	15	25
		Keyifli olur.	12	20
		Kalıcılığı artırır.	5	9
		Dersi sevdirir.	3	5

Tablo 4.13'e göre, oyun geliştirmeye dayalı programlama öğretiminin uygulanması durumunda katılımcılar bunun sağlayacağı avantajlar olarak yararlı olacağı, dersin verimliliğini artıracağı, öğrencilerin derse karşı motivasyonunu artıracağı, keyifli olacağı, kalıcılığı artıracağı ve dersi sevdireceği yönünde fikirleri bulunmaktadır. Bu konu ile ilgili katılımcı görüşlerinden bazıları aşağıda verilmiştir.

"Oyun geliştirmek üzere kurgulanmış bir ders almak". Bu cümleyi duymak bile beni çok heyecanlandırdı. Derslerin bu şekilde geçmesi için buraya en güzel yazıyı yazmam gerekiyor diye düşünüyorum şuan. Kesinlikle

* Katılımcı sayısı: 59. Bir katılımcının vermiş olduğu cevap birden fazla kod içerebilir.

yararlı olur. Sadece ders içerisinde değil, ders dışında dâhi programlamayı araştıracağımıza, yeni bilgiler öğrenmek isteyeceğimize eminim biz öğrenciler olarak. Tekrar söylüyorum ki kesinlikle ve kesinlikle bu şekilde ders işlenmesi büyük bir oranda başarı elde edilmesi demektir. (M14)

Yararlı olur çünkü uzun soluklu ve birçok aşaması olan bir uygulama. Bu yüzden bu uygulamayı yaparken birçok yeni bilgi edinebilir ve farklı sorunlarla karşılaşacağımızdan faydalı olacağını düşünüyorum. (M28)

Yararlı olabileceğini düşünüyorum çünkü benim gibi çevremdeki çoğu insan da oyun oynamaya meraklı ve böyle bir oyun geliştirmek içsel motivasyonu arttırabilir. (M38)

Oyun geliştirmeye dayalı öğretimin dezavantajları olarak yedi katılımcı bunun öğrenciler açısından zorlayıcı olacağını, dört katılımcı sıkıcı olacağını ve bir katılımcı ise yararlı olmayacağını belirtmiştir. Bu konu ile ilgili katılımcı görüşlerinden bazıları aşağıda verilmiştir.

Bence yararlı olmaz çünkü aynı şeyleri kodlamak öğrencileri bir süreden sonra sıkmaya başlayacaktır ve öğrenciler bundan veri alamaz hale gelecektir. (M16)

Derste oyun geliştirmek yararlı olmayacaktır. Çünkü üç boyutlu ortam ve fizik bilgisi işin içine girdiği için öğrenci pek çok şeyi bilmek zorunda kalacak ve zorlanacaktır. Kodlama öğrenmekten uzaklaşacaktır. (M44)

4.1.2. Tasarım Aşaması

4.1.2.1. Araştırma Sorusu 5: NYP dersi Türkiye'deki diğer MYO'larda hangi içeriklerle verilmektedir?

Mevcut NYP dersinin oyun geliştirmeye dayalı öğretim stratejisi çerçevesinde geliştirilmesi sürecinde, gerçekleştirilecek öğretim tasarımına kaynak teşkil edecek şekilde mevcut NYP dersinin konu kapsamıyla Türkiye'deki diğer MYO'larda okutulan NYP derslerinin konu kapsamları karşılaştırılmıştır. Bu karşılaştırmadaki temel amaç, diğer MYO'larda okutulan NYP dersi ile geliştirilecek NYP dersinin konu kapsamlarında uyumun sağlanması; bu sayede öğrencilerin diğer MYO'lardan mezun olan öğrencilerle benzer kazanımlar edinerek mezun olmalarının temin edilmesidir. Bu kapsamda mevcut NYP dersinin konu kapsamı ile Türkiye'deki üniversitelerin 2019 yılında en yüksek puanla öğrenci kabul eden Bilgisayar Programcılığı ön lisans programları Yükseköğretim Program Atlası (https://yokatlas.yok.gov.tr/) kullanılarak karşılaştırılmıştır. Bu üniversitelerde okutulan NYP derslerinin ders saatleri ve ders konu kapsamları Tablo 4.14'te sunulmuştur. Tabloda yer alan derslerin Mevcut NYP dersiyle ortak olan konuları koyu olarak yazılmıştır.

Tablo 4.14. Farklı üniversitelerde okutulan NYP derslerinin konu kapsamları

Üniversite Adı	Ders Kodu ve Adı	Teorik+Uygulama	Ders Konu Kapsamı
İstanbul Medipol Üniv.	BPR1210943 Nesne Yönelimli C# Programlama	2+2	▪ Visual Studio kurulumu ve tanıtımı. ▪ C# dili hakkında bilgiler. ▪ Değişken. ▪ Operatörler ve tür dönüşümleri. ▪ Karar yapıları (if, else, switch). ▪ Döngüler (while, for, do while, foreach). ▪ **Diziler ve çok boyutlu diziler.** ▪ **Metotlar.** ▪ **Sınıflar.** ▪ **Kalıtım.** ▪ **Arayüzler.** ▪ **Soyut sınıflar.** ▪ Veritabanı işlemleri.
	BPR1210945 Nesne Yönelimli Java Programlama	2+2	▪ Yazılım geliştirme döngüsü. ▪ Flowchart. ▪ **Nesne yönelimli programlamanın prensipleri.** ▪ **UML ve Use Case diagram'ları.** ▪ **Java diline giriş (primitive veri tipleri, nesne, sınıf, metot, operatörler, karar yapıları).** ▪ Döngüler. ▪ Hata yönetimi. ▪ **Nesne yönelimli programlama kavramları (kalıtım,** çokbiçimlilik, kapsülleme, **soyutlama, arayüzler).** ▪ Operatörlerin aşırı yüklenmesi.
İstanbul Aydın Üniv.	BUE241 Nesne Tabanlı Programlama-I BUE242 Nesne Tabanlı Programlama-II	3+0	▪ **Nesneye yönelik programlama.** ▪ Algoritma analizi ve tasarımı. ▪ **Sınıflar ve nesneler.** ▪ Değişkenler. ▪ **Metotlar, kurucu metotlar ve yıkıcı metotlar.** ▪ **Kalıtım, arayüzler,** çokbiçimlilik, **çoklu görev ve yöntemler.** ▪ **Soyut, statik ve sabit sınıflar.** ▪ İstisnai durum yönetimi.
İstanbul Bilgi Üniv.	BIL209 Nesne Yönelimli Yazılım - Tasarım Desenleri	3+0	▪ **Nesne yönelimli yazılım geliştirme ve temel kavramlar.** ▪ **Sınıf ve nesneler.** ▪ **Kalıtım.** ▪ Çokbiçimlilik. ▪ Dinamik bağlama. ▪ Soyut sınıflar. ▪ **Arayüzler.** ▪ Tasarım desenleri.

(Devam ediyor)

Tablo 4.14. Farklı üniversitelerde okutulan NYP derslerinin konu kapsamları (Devam)

Üniversite Adı	Ders Kodu ve Adı	Teorik+Uygulama	Ders Konu Kapsamı
Nişantaşı Üniv.	OBLG255 Nesne Tabanlı Programlama I	2+1	▪ Sabit, değişken ve nesne kullanımı. ▪ Operatörlerin kullanımı. ▪ Karar kontrol deyimleri. ▪ Döngü kontrol deyimleri. ▪ Kullanıcı tanımlı fonksiyonlar. ▪ Hazır fonksiyonlar. ▪ Dosya işlemleri. ▪ **Sınıf, alan ve metot kullanımı.** ▪ Lokal ve global referanslar. ▪ **Diziler ve çok boyutlu diziler.** ▪ Standart bileşenler. ▪ Gelişmiş bileşenler. ▪ Veritabanı bağlantısı ve sorguları.
	OBLG256 Nesne Tabanlı Programlama II	2+1	▪ Karar kontrol deyimleri. ▪ Döngü kontrol deyimleri. ▪ **Kontrol nesneleri ve diziler.** ▪ **Nesne kullanımı ve operatörler.** ▪ Bileşen kütüphanesi. ▪ Standart ve kullanıcı tanımlı fonksiyonlar. ▪ İşletim sistemi nesneleri.
Yaşar Üniv.	MBBP2213 Nesne Yönelimli Programlama	3+2	▪ **Nesne yönelimli programlamanın temel kavramları.** ▪ Fonksiyonlar ve döngüler. ▪ Sorgular. ▪ **Sınıflar ve nesneler.** ▪ **Kalıtım** ve çokbiçimlilik. ▪ Metotlar ve metotların aşırı yüklenmesi. ▪ İstisna durumları işleme.
İstanbul Gedik Üniv.	BLP205 Nesne Yönelimli Programlama	2+2	▪ **(Ders içeriğine erişilememiştir.)**
Ufuk Üniv.	MYP 122 Nesneye Yönelik Programlama	2+2	▪ **Nesne tanımı ve genel kavramlar.** ▪ **C# ile nesne tanımı.** ▪ **Nesne özellikleri (metotlar, özellikler).** ▪ **Nesne kullanarak soyutlama.** ▪ **Kalıtım kavramı.** ▪ **C# ile kalıtım kavramı uygulamaları.** ▪ Soyutlama (encapsulation) kavramı. ▪ C# ile soyutlama uygulamaları. ▪ Çokbiçimlilik (polymorphism) ve uygulamaları. ▪ **Nesne metotlarının yeniden tanımlanması** ▪ Soyut nesneler. ▪ SOLID Prensipler ve C# dilinde uygulamalar.

(Devam ediyor)

Tablo 4.14. Farklı üniversitelerde okutulan NYP derslerinin konu kapsamları (Devam)

Üniversite Adı	Ders Kodu ve Adı	Teorik+Uygulama	Ders Konu Kapsamı
Başkent Üniv.	BİLP104 Nesneye Yönelik Programlama I	2+2	▪ Java platformu ve bileşenleri. ▪ Java diline giriş. ▪ Java'da değişkenler ve temel veri tipleri. ▪ String sınıfı. ▪ Main denetimi. ▪ Program akış denetimi ve operatörler, döngüler. ▪ **Diziler.** ▪ **Sınıf ve nesne kavramı.** ▪ **Paketler, metotlar, erişim denetimi.** ▪ **Nesneye yönelik programlamanın temel ilkeleri:** Kapsülleme (encapsulation), **kalıtım (inheritance).** çokbiçimlilik (polymorphism) **ve arayüzler (interfaces).** ▪ Dâhili sınıflar.
	BİLP225 Nesneye Yönelik Programlama II	2+2	▪ **Sınıf ,nesne.** ▪ **Erişim belirleyicileri.** ▪ Kapsülleme. ▪ **Yapılandırıcı kavramları.** ▪ Yapılandırıcı aşırı yükleme ve metot aşırı yükleme. ▪ Miras (kalıtım), alt sınıf, üst sınıf, super yapılandırıcı. ▪ Metot örtme, soyut sınıflar ve soyutlama. ▪ Çokbiçimlilik. ▪ Java'da bellek kullanımı, Genetic kavramı. ▪ Collections framework'e giriş. ▪ List arayüzü, Arraylist yapısı. ▪ Sıralama algoritmaları. ▪ Comparatör arayüzü, Map Arayüzü, HashMap, TreeMap, LinkedHashMap, Set arayüzü, HasSet, TreeSet, LinkedHasSet. ▪ Dosya okuma yazma. ▪ Hata Kontrolü.
İzmir Ekonomi Üniv.	MBP106 Nesne Tabanlı Programlama I	2+2	▪ Giriş, çalışma ortamı ve kurulumu. ▪ Temel sözdizimi ve Java programlama kuralları. ▪ Değişkenler, veri tipleri ve operatörler. ▪ Kontrol yapıları, döngüler, diziler, metotlar. ▪ **Sınıflar ve nesneler.** ▪ Hata yakalama. ▪ Math sınıfı ile matematiksel işlemler. ▪ String sınıfı ile metin işlemleri. ▪ Dosya işlemleri.
	MBP205 Nesne Tabanlı Programlama II	2+2	▪ **Nesne tabanlı programlamaya giriş.** ▪ **Sınıflar, nesneler ve metotlar.** ▪ **Sınıflar üzerinde aşırı yükleme ve kurucu metotlar.** ▪ **Nesne tabanlı programlamanın temelleri: kalıtım,** çokbiçimlilik, soyutlama ve kapsülleme. ▪ Soyut sınıflar. ▪ **Arayüzler.** ▪ Hata ayıklama. ▪ Generic sınıf ve metotlar. ▪ Tasarım desenleri.

(Devam ediyor)

Tablo 4.14. Farklı üniversitelerde okutulan NYP derslerinin konu kapsamları (Devam)

Üniversite Adı	Ders Kodu ve Adı	Teorik+Uygulama	Ders Konu Kapsamı
Marmara Üniv.	BLY2003 Nesne Yönelimli Programlama I	3+1	▪ **Nesne tabanlı (Object-oriented) temel ilke ve kavramlar.** ▪ Java programlama dilinin temel ilke ve kavramları. ▪ İfadeler ve işlem önceliği. ▪ Veri dönüşümleri. ▪ Kullanıcıdan girdileri alma. ▪ Grafiğe giriş. ▪ Package ve Import kavramı. ▪ **Nesne oluşturulması ve nesne referansları.** ▪ String sınıfı ve metotları. ▪ Random ve Math sınıfları. ▪ Giriş-çıkış (I/O) işlemleri. ▪ Kontrol yapıları, tekrarlı yapılar. ▪ Döngüler. ▪ **Dizi kavramı, iki ve çok boyutlu diziler.** ▪ Sıralama algoritmaları. ▪ Arama algoritmaları. ▪ **UML diyagramları.** ▪ **Sınıf, nesne ve nesne üyelerine erişim belirteçleri.** ▪ **Metot kavramı**. ▪ **Kalıtım.** ▪ Kapsülleme, çokbiçimlilik. ▪ Applet ve grafik.
	BLY2004 Nesne Yönelimli Programlama II	3+1	▪ Alt programlar. ▪ **Fonksiyonlar.** ▪ Applet kavramı. ▪ Grafik uygulamaları. ▪ Dosya işlemleri ve ikili dosyalar. ▪ Hata yakalama. ▪ Yığın ve kuyruk yapısı. ▪ Tek bağlı doğrusal ve dairesel listeler. ▪ Çift bağlı doğrusal ve dairesel listeler. ▪ Veritabanı bağlantısı.

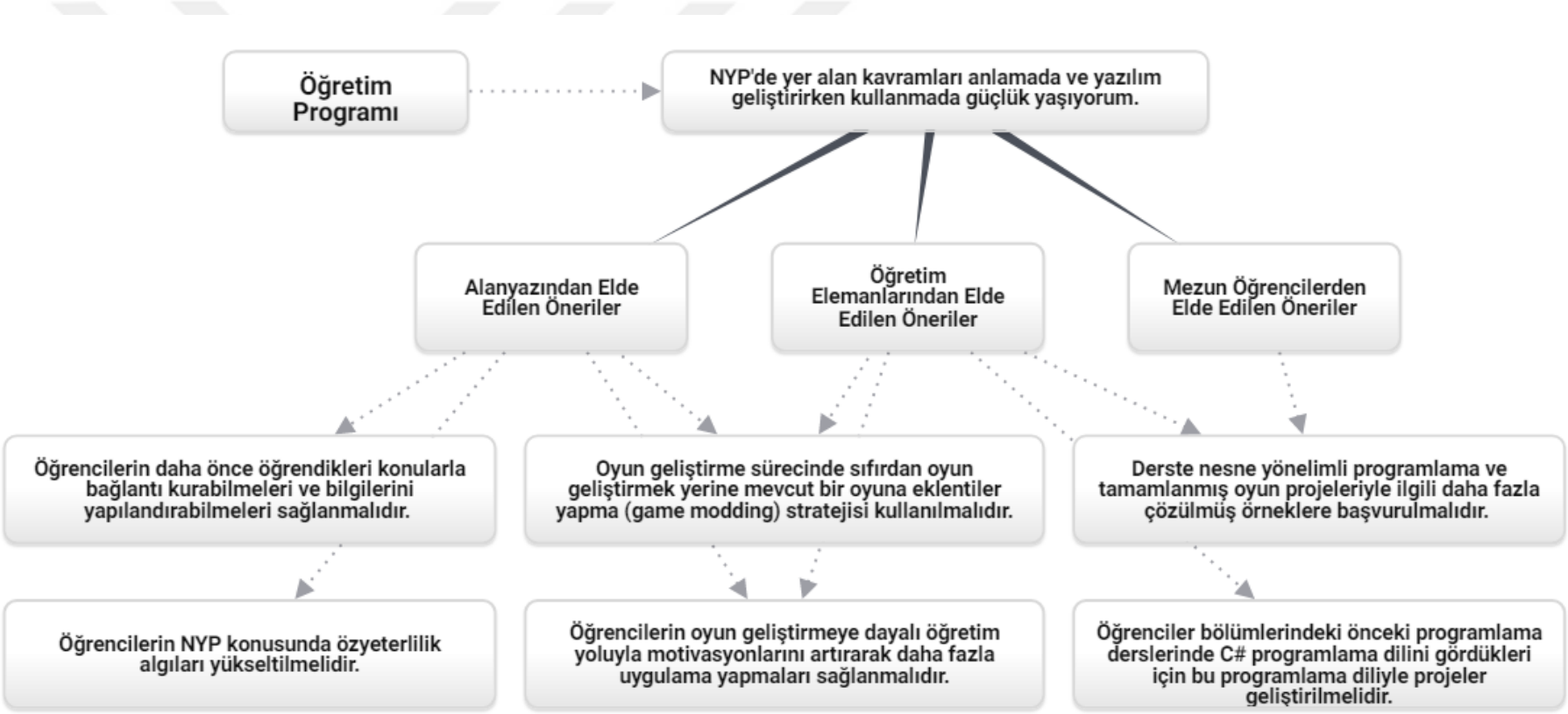
Tablo 4.14'te görüldüğü üzere MYO'larda verilen NYP dersinin isimlerinde, ders sayılarında ve ders saatlerinde farklılıklar bulunmaktadır. Örneğin NYP dersi Nişantaşı Üniversitesinde Nesne Tabanlı Programlama ismi ile verilirken Yaşar Üniversitesinde Nesneye Yönelik Programlama ismi ile verilmektedir. Benzer şekilde NYP dersi İstanbul Medipol Üniversitesinde haftada iki teorik iki uygulama olmak üzere toplam dört saat verilirken İstanbul Aydın Üniversitesinde haftada üç saat teorik olarak verilmektedir. NYP dersi Nişantaşı Üniversitesi gibi bazı üniversitelerde Nesne Tabanlı Programlama I ve II olmak üzere iki farklı derse ayrılabilmektedir.

Tablo 4.14 incelendiği zaman her ne kadar ismi NYP olsa da bazı derslerin içeriklerinde IDE kurulumu, değişkenler, matematiksel ve mantıksal operatörleri karar yapıları, döngüler gibi

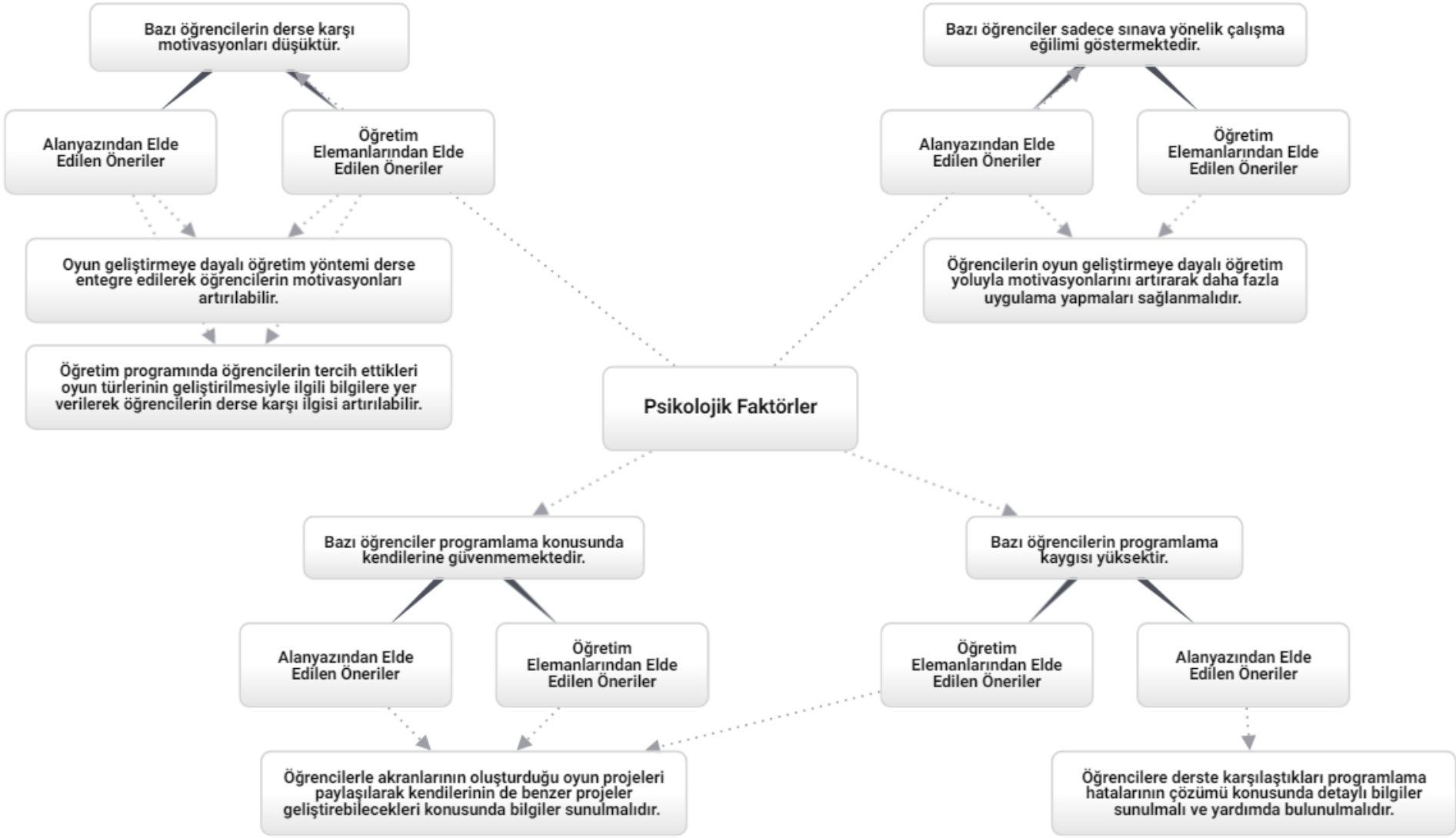
prosedürel programlama içeriklerinin yer aldığı görülmektedir. Bu da MYO'larda okutulan NYP dersi için bütüncül bir çerçevede ortak bir ders içeriği standardının olmadığını; NYP derslerinin ağırlıklı olarak prosedürel programlamanın unsurlarının tanıtımından sonra NYP paradigmasına geçişi ifade eden *"daha sonra nesne (objects-later)"* yaklaşımını barındırdığını göstermektedir. Bu durumdan farklı olarak araştırmanın gerçekleştirildiği MYO'da prosedürel programlamayı barındıran içerikler Algoritma ve Programlamaya Giriş dersi ile Görsel Programlama derslerinde verilmekte olup, NYP dersi sadece NYP paradigmasının tanıtılması ve uygulanması üzerine yoğunlaşmaktadır. Gerçekleştirilen analizler sonucunda mevcut NYP dersinde *soyut sınıflar, metotların aşırı yüklenmesi, çokbiçimlilik, kapsülleme ve hata kotarımı (istisnai durum yönetimi)* konularının olmadığı tespit edilmiştir. Bu konuların tasarlanan NYP dersine ilave edilip edilemeyeceği derse giren öğretim elemanlarının (Tablo 3.5) katılımıyla gerçekleştirilen odak grup görüşmesinde tartışılmıştır. Bu görüşmenin sonucunda tasarlanan NYP dersine *çokbiçimlilik, kapsülleme ve hata kotarımı* konularının eklenmesinin elzem olduğu değerlendirilmiştir. Tasarlanan dersin ilk üç haftasının Unity 3D editörü tanıtımına ayrılacak olmasının oluşturacağı zaman kısıtlaması nedeniyle *soyut sınıflar ve metotların aşırı yüklenmesi* konuları öğretim programına dâhil edilmemiştir. İlave olarak gerçekleştirilen odak grup görüşmesi neticesinde tasarlanan NYP dersinde zaten bir oyun projesinin oluşturulacak olması ve bu projenin kurulum işlemlerinin Unity 3D'de yapılacak olması nedeniyle mevcut NYP dersindeki *nesne yönelimli proje oluşturma* ve *kurulum dosyası oluşturma* konularının tasarlanan NYP dersinin öğretim programına dâhil edilmemesi kararlaştırılmıştır.

4.1.2.2. Araştırma Sorusu 6: Mevcut problemleri ortadan kaldırmak için oyun geliştirmeye dayalı öğretim NYP dersine nasıl entegre edilebilir?

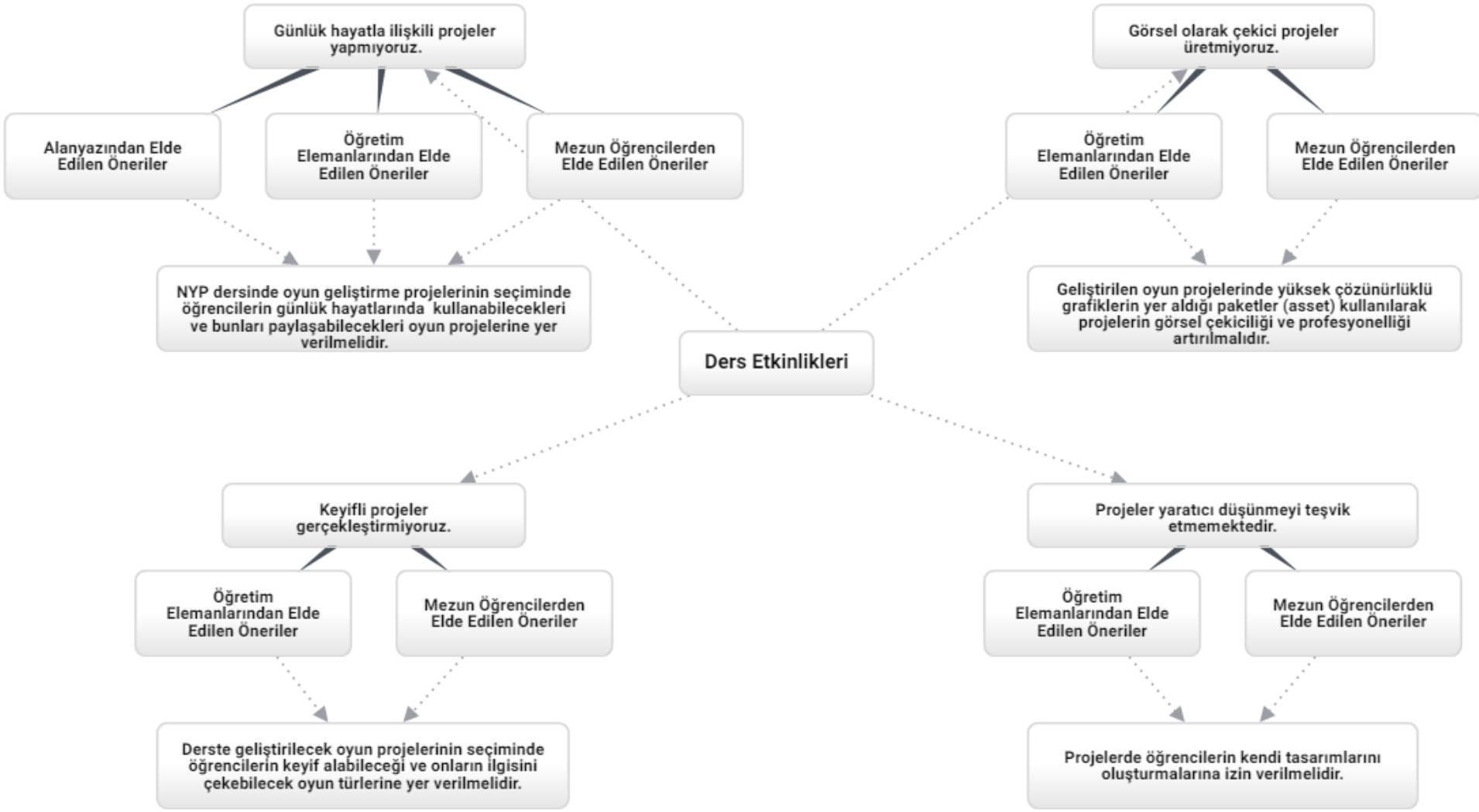
Araştırmanın tasarım aşamasında, mevcut NYP dersinin oyun geliştirmeye dayalı öğretim stratejisini barındıracak şekilde yeniden tasarlanabilmesi için kullanılacak stratejiler belirlenmiştir. Bununla birlikte bu aşamada; geliştirilen stratejiler doğrultusunda mevcut NYP dersinin öğretim tasarımı, öğrencilerin ve öğretim elemanlarının yaşadıkları sorunlara (Tablo 4.5 ve Tablo 4.6) yönelik çözüm önerilerini de kapsayacak şekilde yeniden yapılmıştır. Bu süreçte analiz aşamasında toplanan verilerden, uzman görüşlerinden ve programlama alanyazınından yararlanılmıştır. Mevcut NYP dersinde öğrencilerin ve öğretim elemanlarının yaşadıkları sorunlara yönelik çözüm önerileri Şekil 4.8-Şekil 4.14'te sunulmuştur.



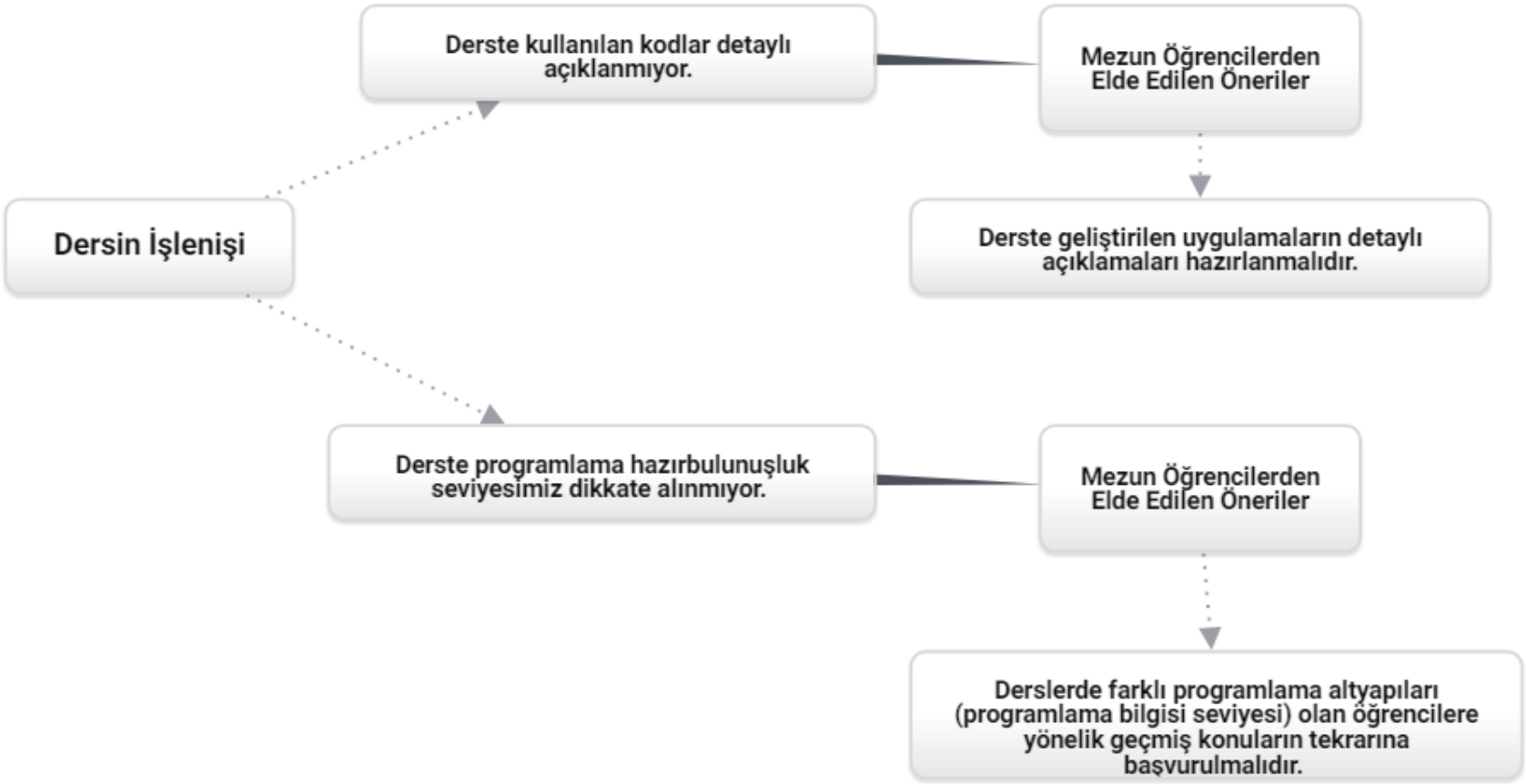
Şekil 4.8. Öğretim programı sorun kategorisine ait çözüm önerileri



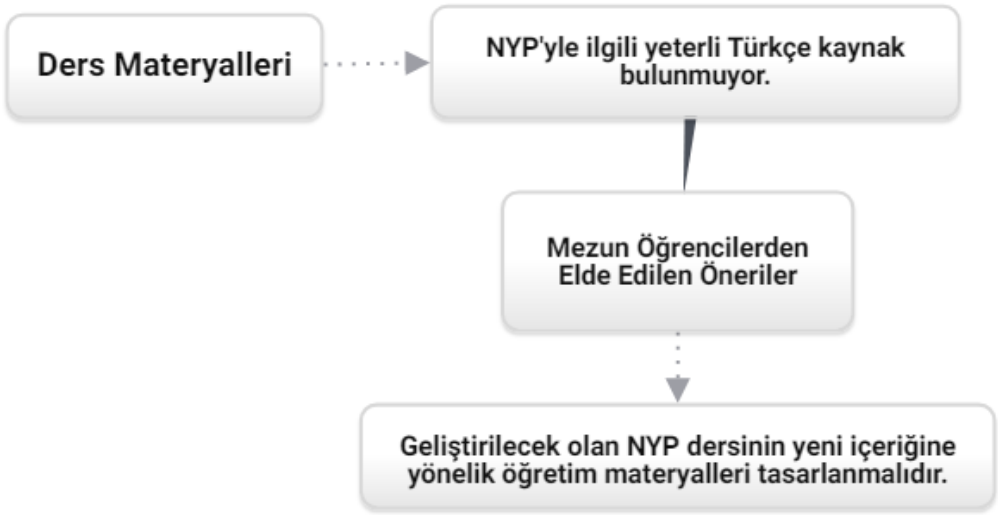
Şekil 4.9. Psikolojik faktörler sorun kategorisine ait çözüm önerileri



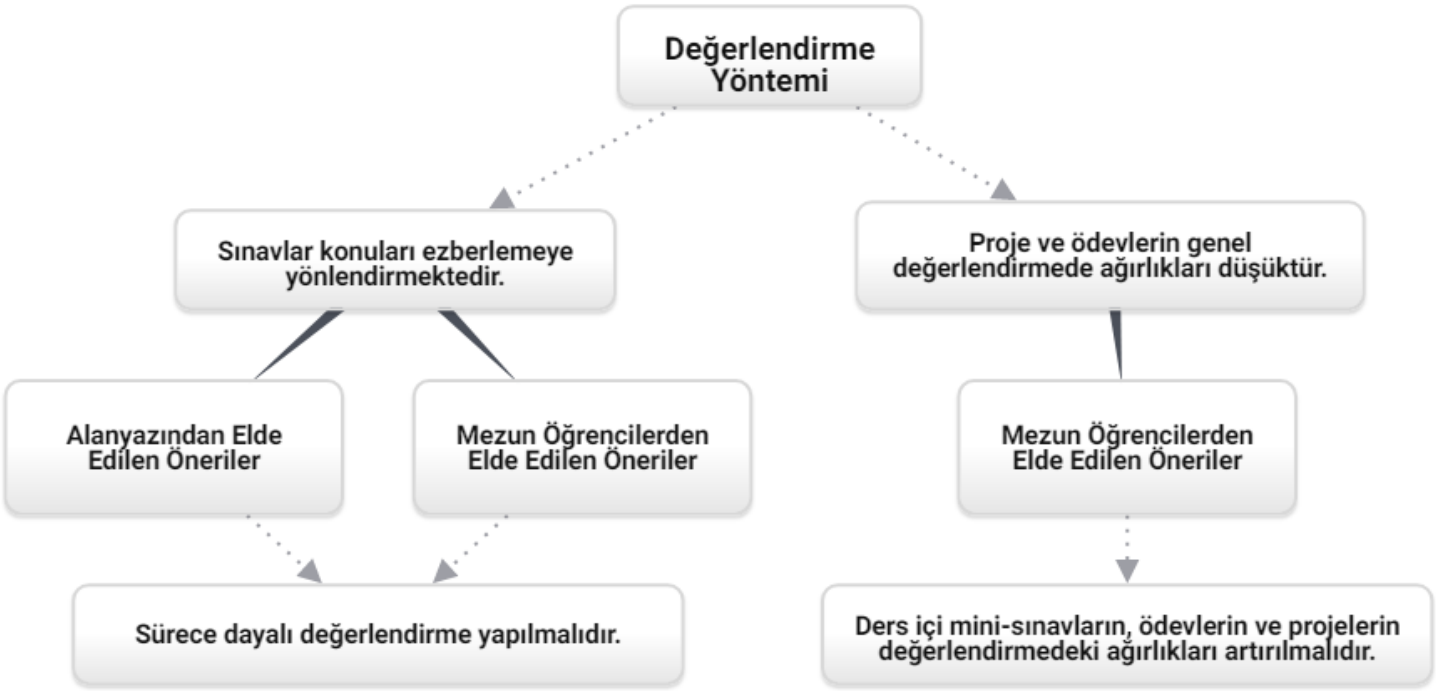
Şekil 4.10. Ders etkinlikleri sorun kategorisine ait çözüm önerileri



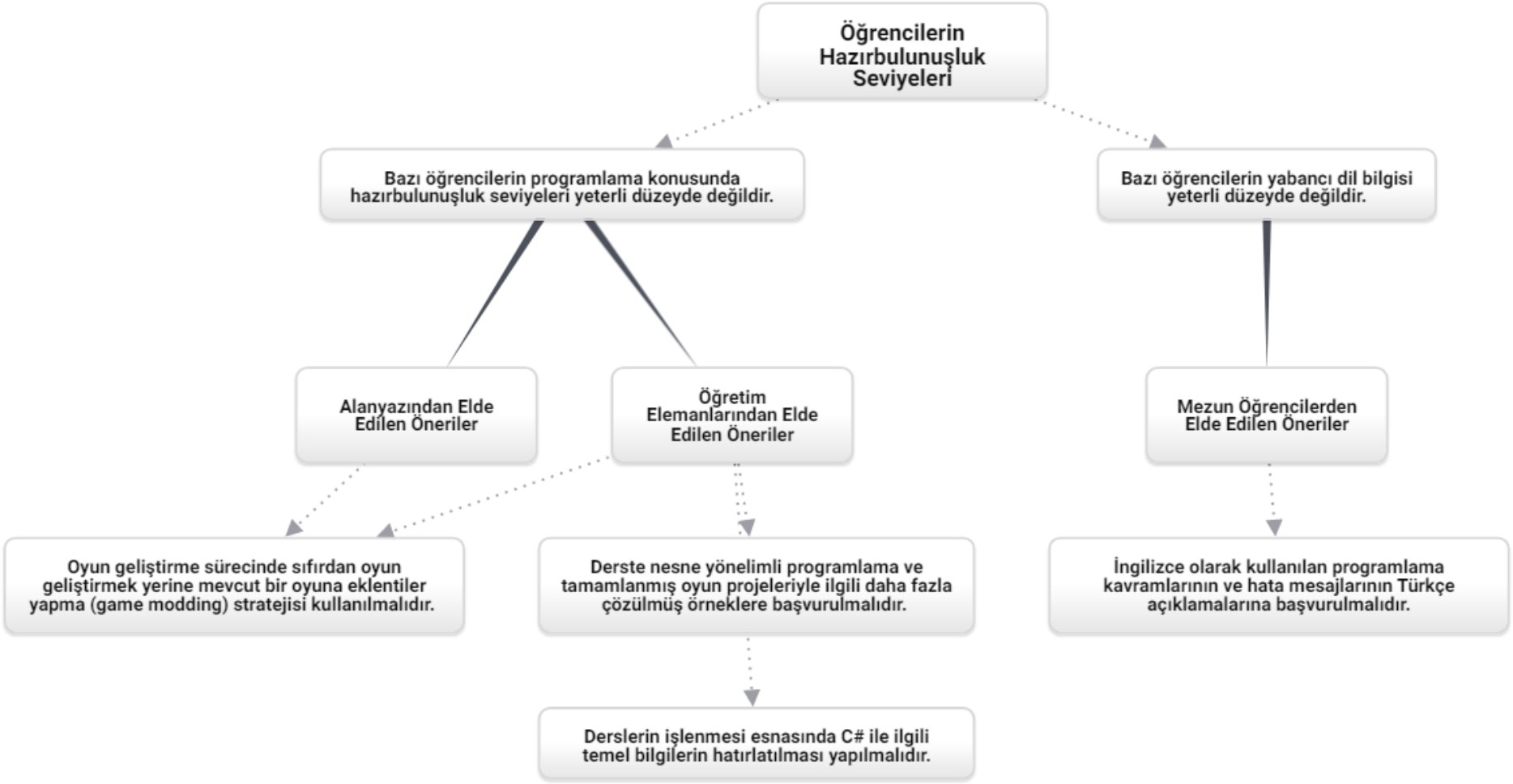
Şekil 4.11. Dersin işlenişi sorun kategorisine ait çözüm önerileri



Şekil 4.12. Ders materyalleri sorun kategorisine ait çözüm önerileri



Şekil 4.13. Değerlendirme yöntemi sorun kategorisine ait çözüm önerileri



Şekil 4.14. Öğrencilerin hazırbulunuşluk seviyeleri sorun kategorisine ait çözüm önerileri

Elde edilen öneriler doğrultusunda mevcut NYP dersinin sorunlarını en aza indirmek maksadıyla belirlenen çözüm stratejileri Tablo 4.15'te verilmiştir.

Tablo 4.15. Elde edilen çözüm önerileri doğrultusunda öğretim tasarımında kullanılan stratejiler

Kategori	Mevcut NYP Dersinde Karşılaşılan Sorunlar	Sorunların Çözümüne Yönelik Öğretim Tasarımında Yer Alan Stratejiler
Öğretim Programı	NYP'de yer alan kavramları anlamada ve yazılım geliştirme sürecinde kullanmada güçlük yaşıyorum.	▪ Oyun projelerini geliştirme sürecinde sıfırdan oyun hazırlamak yerine mevcut bir oyuna eklentiler yapma (game modding) stratejisinin kullanılmasına karar verilmiştir. ▪ Oyun projelerini geliştirirken Kullan-Değiştir-Oluştur modeline (Lee ve diğerleri, 2011) başvurulmasına karar verilmiştir. ▪ Öğrenciler önceden aldıkları programlama dili derslerinde C# programlama dilini kullandıkları için derste bu programlama dilinin kullanılmasına karar verilmiştir. ▪ Oyun projelerinin geliştirilmesi sürecinde C# programlama diliyle uyumlu bir oyun motoru olan Unity 3D editörünün kullanılmasına karar verilmiştir. ▪ NYP ve oyun geliştirme ile ilgili çözülmüş örnekler hazırlanmıştır.
Psikolojik Faktörler	Bazı öğrencilerin derse karşı motivasyonları düşüktür.	▪ Dersin öğretim programında oyun geliştirmeyle ilgili kazanımlara ve örnek projeler yer verilerek öğrencilerin derse karşı motivasyonlarının yükseltilmesi amaçlanmıştır. ▪ Dersin öğretim programında Unity 3D ile geliştirilmiş oyun projeleriyle ilgili bilgilere yer verilmiştir. Kendi oyunlarını da çeşitli platformlarda paylaşabilecekleri konusunda bilgiler hazırlanmıştır.
	Bazı öğrenciler sadece sınava yönelik çalışma eğilimi göstermektedir.	▪ Değerlendirmelerde ders içi mini-sınavların, ödevlerin ve bireysel oyun projesinin ağırlığı artırılmıştır.
	Bazı öğrenciler programlama konusunda kendilerine güvenmemektedir.	▪ Dersin öğretim programına öğrencilerin akranlarının Unity 3D ile geliştirmiş oldukları oyun projeleriyle ilgili bilgilere yer verilerek kendilerinin de benzer oyunları geliştirebilecekleri konusunda bilgiler hazırlanmıştır.
	Bazı öğrencilerin programlama konusundaki kaygısı yüksektir.	▪ Oyun projelerinin geliştirilmesi esnasında öğrencilere akran ve öğretmen yardımının sunulması kararlaştırılmıştır. ▪ Öğrencilerin gelişim aşamaları takip edilerek genel durumlarıyla ilgili bireysel geribildirimlerde bulunulmasına karar verilmiştir. ▪ Karşılaşılan hatalarla ilgili Türkçe kaynaklar hazırlanmıştır.

(Devam ediyor)

Tablo 4.15. Elde edilen çözüm önerileri doğrultusunda öğretim tasarımında kullanılan stratejiler (Devam)

Kategori	Mevcut NYP Dersinde Karşılaşılan Sorunlar	Sorunların Çözümüne Yönelik Öğretim Tasarımında Yer Alan Stratejiler
Ders Etkinlikleri	Günlük hayatla ilişkili projeler yapmıyoruz.	▪ Dersin öğretim programında Unity 3D gibi profesyonel bir oyun motoru kullanılarak geliştirilen oyun projesi örneklerine yer verilmiştir. Bununla birlikte bu derste edinilen neticesinde kendilerinin de Unity FSP Microgame, Lego Microgame gibi oyun prototiplerini değiştirebilecekleri gibi bilgilere yer verilmiştir. ▪ Dersin öğretim programında öğrencilerin geliştirdikleri oyunları mobil platformlar ya da VR gibi ortamlara aktarabilecekleriyle ilgili bilgilere yer verilmiştir.
	Görsel olarak çekici projeler üretmiyoruz.	▪ Oyun geliştirme sürecinde öğrencilerin hazırbulunuşluk seviyelerine uygun olarak kullanabilecekleri Unity mikro oyunlarından Karting Microgame'in dersin öğretim programına entegre edilmesi kararlaştırılmıştır. Üç boyutlu grafiklere sahip olan bu oyunun görsel olarak ilgi çekici olduğuna karar verilmiştir.
	Keyifli projeler gerçekleştirmiyoruz. Projeler yaratıcı düşünmeyi teşvik etmemektedir.	▪ Kendi oyun projelerini istedikleri gibi tasarlamalarını sağlamak ve kendi senaryolarını oyunlarına uygulamalarına olanak tanımak amacıyla çeşitli Unity paketlerinin öğrencilere tanıtılması kararlaştırılmıştır. Bu paketler sayesinde öğrencilerin kendi tasarım anlayışlarını yansıtan bir oyun projesi oluşturabilecekleri değerlendirilmiştir.
Dersin İşlenişi	Derste öğrencilerin programlama hazırbulunuşluk seviyesi dikkate alınmıyor.	▪ Öğrencilerin kendi öğrenme hızlarında öğrenebilmeleri için ders materyalleri hazırlanmıştır. ▪ Derste farklı programlama altyapıları (programlama bilgisi seviyesi) olan öğrencilere yönelik geçmiş konuların tekrarına başvurulması kararlaştırılmıştır.
	Derste kullanılan kodlar detaylı açıklanmıyor.	▪ Derste geliştirilen uygulamaların detaylı açıklamaları hazırlanmıştır.
Ders Materyalleri	NYP ile ilgili yeterli Türkçe kaynak bulunmuyor.	▪ Geliştirilen NYP dersinin yeni içeriğine ve kullanılan oyun projesinin geliştirilmesine yönelik Türkçe kaynaklar hazırlanmıştır.
Değerlendirme Yöntemi	Proje ve ödevlerin genel değerlendirmede ağırlıkları düşüktür. Sınavlar konuları ezberlemeye yönlendiriyor.	▪ Değerlendirmelerde ders içi mini-sınavların, ödevlerin ve bireysel oyun projesinin ağırlığı artırılmıştır.

(Devam ediyor)

Tablo 4.15. Elde edilen çözüm önerileri doğrultusunda öğretim tasarımında kullanılan stratejiler (Devam)

Kategori	Mevcut NYP Dersinde Karşılaşılan Sorunlar	Sorunların Çözümüne Yönelik Öğretim Tasarımında Yer Alan Stratejiler
Öğrencilerin Hazırbulunuşluk Seviyeleri	Bazı öğrencilerin NYP dersi için programlama hazırbulunuşluk seviyeleri yeterli düzeyde değildir. Bazı öğrencilerin matematiksel bilgi ve algoritmik düşünme seviyeleri yeterli düzeyde değildir.	▪ Öğrencilerle algoritma tasarımı, C#'ta yer alan değişkenler, aritmetiksel ve mantıksal operatörler, karar yapıları, döngüler, diziler ve metotlar gibi konularla ilgili kaynakların paylaşılması kararlaştırılmıştır. ▪ Öğrencilerin derste yer alan konuları daha kolay anlayabilmeleri için detaylı açıklamalara ve çözülmüş örneklere başvurulması kararlaştırılmıştır.
	Bazı öğrencilerin yabancı dil bilgisi yeterli seviyede değildir.	▪ Dersin öğretim programında yer alan İngilizce terimlerin Türkçe açıklamaları içerek kaynak hazırlanmıştır. ▪ Unity 3D yer alan İngilizce terimlerin ders esnasında açıklanması kararlaştırılmıştır.

4.1.2.3. Öğretim tasarımı süreci

NYP dersinin iyileştirilmesi ve geliştirilmesi kapsamında gerçekleştirilen öğretim tasarımı, nesne yönelimli programlama kavram ve tekniklerinin ders materyallerine ve öğrenim planına sistematik olarak aktarılması süreci olarak tanımlanabilir (Gustafson ve Branch, 1997). Bu süreç, Morrison, Ross ve Kemp'in (2004) öğretim tasarım modeli (Şekil 4.15) temel alınarak gerçekleştirilmiştir. Eylem araştırmasının karakteristik özellikleri değerlendirildiğinde bu modelin; öğretim sorunlarından kaynaklanan sorunlara çözümler içermesi, lineer olarak ilerlemeyip esnek bir yapıya sahip olması ve etkili, güvenilir ve verimli bir öğretim programı geliştirmek için teknoloji, pedagoji ve içeriği harmanlaması gibi özellikleri sebebiyle NYP dersinin öğretim tasarımı modeli olarak seçilmiştir. Nitekim iki döngü şekilde tamamlanan bu eylem araştırmasında da birinci döngü sonunda elde edilen deneyimler sonucunda revizyonlar gerçekleştirilmiş; öğretim tasarım modelinin esnek yapısı öğretim içeriği, içeriğin sıralaması, ders etkinlikleri ve materyalleri bağlamında gerçekleştirilen değişikliklere olanak tanımıştır.

Programlama bağlamında ise bu modelin seçilmesin ilk nedeni; modelin öğrenen merkezli olup öğretim tasarımına içerik açısından bakmak yerine öğrenen ve bağlama yoğunlaşmasıdır. Bu eylem araştırmasına katılan öğrenenlerin geçmiş programlama bilgi ve becerileri gerek öğretimin hedeflerinin ve görevlerinin, gerekse öğretim içeriğinin belirlenmesinde etkin bir rol oynamıştır. Örneğin katılımcıların NYP dersini almadan önce bir dönem Algoritma ve Programlamaya Giriş, diğer bir dönem ise Görsel Programlama dersini almış olmaları NYP dersinin öğretim tasarımının "daha sonra nesne" stratejisine uygun olarak gerçekleştirilmesini

zorunlu kılmıştır. Benzer şekilde katılımcıların büyük bir çoğunluğunun daha önce oyun geliştirmeyle ilgili bir deneyime sahip olmamaları sıfırdan bir oyun geliştirmek yerine oyuna eklentiler yapma yoluyla oyun programlamayı gerektirmiştir. Bu etkenler dersin hedeflerini, görevlerini ve içeriğini doğrudan etkilemiştir. Programlama bağlamında bu modelin seçilmesin ikinci nedeni ise oyun geliştirmeye dayalı öğretimin temel aldığı proje tabanlı öğrenmenin bu öğretim tasarım modeline uygunluğuyla ilgilidir. Bu modelin programlama projelerinin belirlenmesi, planlanması, yönetilmesi ve süreç içinde değerlendirmesine kolaylık sağladığı söylenebilir (Cano, Ruiz, ve Garcia, 2015). Nitekim NYP dersinde de katılımcıların mevcut bir oyuna eklentiler yapma yoluyla NYP prensiplerini öğrenmeleri hedeflenmiş; öğretim programında yer alan ders içerikleri ile oyun projesi geliştirme basamakları birbirlerine entegre edilerek dersin görevleri bu bağlamda hazırlanmıştır.

Tüm bu bilgilerin ışığında öğretim tasarımı süreci, aşağıdaki dört sorunun yanıtlarını aramakla başlamıştır:

1. Öğretim programının hedef kitlesinde yer alan öğrencilerin karakteristik özellikleri nelerdir? *(Öğrenenlerin doğası)*
2. Öğretim sürecinin sonunda öğrencilerin hangi konularda bilgi sahibi olmaları ve hangi becerileri kazanmaları beklenmektedir? *(Hedefler)*
3. Ders içeriği en iyi nasıl öğrenilir ve istenen beceriler en iyi nasıl kazanılır? *(Öğretim/öğrenme yöntemleri, ders materyalleri ve etkinlikleri)*
4. Öğrenmenin hangi ölçüde gerçekleştiği nasıl belirlenebilir? *(Değerlendirme)*

Bu soruların yanıtlarını bulma motivasyonuyla gerçekleştirilen öğretim tasarımının detayları, aşağıdaki alt maddelerde açıklanmıştır. Morrison ve diğerlerinin (2004) sunduğu model bağlamında açıklanan bu sürecin, geliştirilen NYP dersinin karakteristik özelliklerinin daha iyi anlaşılmasına katkıda bulunacağı değerlendirilmektedir.

Şekil 4.15. Morrison, Ross ve Kemp Modeli (2004)

4.1.2.2.1. Öğretim problemleri

Gerçekleştirilen öğretim tasarımı sürecinin ilk adımında, mevcut NYP dersinde öğrencilerin ve öğretim elemanlarının yaşadıkları problemler açık bir şekilde tespit edilmiştir. Bu tespitin öğretim tasarımı açısından kritik öneme sahip olduğu değerlendirilmektedir çünkü bir öğretim tasarımcısının öğretim ortamındaki eksik ve sorunlu unsurları görememesi, bu hususları iyileştirmesini olanaksız kılmaktadır (Uzun, 2012). Nitekim Bölüm 4.1.1.3.'te açıklanan bu problemler, bu tez çalışmasının gerçekleştirilmesinin de itici gücü olmuş; bu problemleri en aza indirilerek NYP dersinin etkinliğinin ve verimliliğinin artırılması tezin temel amaçlarından birisini teşkil etmiştir. Bu nedenle hem öğretim tasarımı sürecinde hem de eylem araştırmasının yürütülmesi aşamasında NYP dersinin problemli alanları sürekli göz önünde bulundurulmuştur. Bölüm 4.1.1.3.'te detayları sunulan bu problemler şunlardır:

- Öğrencilerin Yaşadıkları Sorunlar
  - Öğretim Programı
  - Dersin İşlenişi
  - Ders Etkinlikleri

- Ders Materyalleri
- Değerlendirme Yöntemi
- Teknolojik Altyapı

▪ Öğretim Elemanlarının Yaşadıkları Sorunlar
- Psikolojik Faktörler
- Öğretim Programı
- Ders Etkinlikleri
- Öğrencilerin Hazırbulunuşluk Seviyeleri
- Teknolojik Altyapı

4.1.2.2.2. Öğrenen özellikleri

Öğretim tasarımın sürecinde öğrenenlerin yaşlarının, hazırbulunuşluk seviyelerinin, önceki bilgi ve becerilerinin önem arz etmesi (Sisovic, Matetic ve Bakaric, 2015) sebebiyle 2017, 2018 ve 2019 yıllarının bahar döneminde NYP dersini alan öğrencilerin profilleri incelenmiş ve gerçekleştirilen öğretim tasarımında öğrencilerin bu özellikleri dikkate alınmıştır. Adı geçen yıllarda NYP dersini alan öğrencilere ait demografik özellikler Tablo 4.16'da sunulmuştur.

Tablo 4.16. Geçmiş yıllarında NYP dersini alan öğrencilere ait demografik özellikler

		Frekans (f)	**Yüzde (%)**
Mezun Olunan Ortaöğretim Kurumu	Anadolu Lisesi	37	50
	Mesleki ve Teknik Anadolu Lisesi	30	40
	Açık Öğretim Lisesi	4	6
	Anadolu İmam-Hatip Lisesi	3	4
Ortaöğretimde Programlama Dersi Alma Durumu	Alan	10	14
	Almayan	64	87

Tablo 4.16 incelendiğinde 2017, 2018 ve 2019 yıllarında NYP dersini alan 74 ön lisans öğrencisinin tamamının erkek ve 17-21 yaş aralığında olduğu görülmektedir. İlave olarak bu öğrencilerin büyük çoğunluğunun (N:67; %91) mezun olduğu orta öğretim kurumu Anadolu Lisesi ve Mesleki ve Teknik Anadolu Lisesidir. Benzer şekilde öğrencilerin büyük çoğunluğunun (N:64; %87) ortaöğretimde programlama dersi almadıkları görülmektedir. 2017, 2018 ve 2019 yıllarında NYP dersini alan öğrencilerin özelliklerinin benzerlikler barındırması,

her yıl Bilgisayar Teknolojisi Programına kayıt yaptıran öğrenci profillerinin benzer olacağını garanti etmemektedir ancak bu öğretim tasarımıyla hazırlanan NYP dersini alacak öğrencilerin genel özelliklerinin diğer yıllarla benzerlikler taşıyacağı beklenmektedir.

Öğrenci profillerine ilave olarak, kayıtlı oldukları Bilgisayar Teknolojisi Programında aldıkları Görsel Programlama ve Algoritma ve Programlama Giriş derslerinde edindikleri bilgi ve becerilerin öğrencilerin NYP dersindeki hazırbulunuşluk seviyelerini etkileyeceği değerlendirilmektedir. Bu bağlamda öğrencilerin ön lisans eğitimleri sürecinde aldıkları önceki programlama derslerinin içeriğinden ve kazanımlarından bahsetmek uygun olacaktır. Algoritma ve Programlamaya Giriş dersinin sonunda öğrencilerin;

- Algoritma kavramını açıklamaları,
- Algoritmaların kullanım amaçlarını kavramaları,
- Bir problemin çözümüne yönelik algoritma tasarlayabilmeleri,
- Sözde kod (pseudo code) ile algoritma oluşturabilmeleri,
- Akış şemalarını tanıyabilmeleri,
- Tasarladıkları algoritmaları akış şemalarıyla gösterebilmeleri,
- Visual Studio yazılım geliştirme ortamını (IDE) kullanabilmeleri,
- Tasarladıkları algoritmaları C# programlama dili kullanarak konsol uygulamalarına dönüştürebilmeleri,
- Değişken, veri tipleri, aritmetik ve mantıksal operatörler, karar yapıları, döngüler, metotlar gibi programlama kavramlarını kavrayabilmeleri hedeflenmektedir.

Benzer şekilde Görsel Programlama dersinin sonunda öğrencilerin;

- Visual Studio yazılım geliştirme ortamını kullanarak form uygulamaları oluşturmaları,
- Visual Studio araç kutusunda yer alan form nesnelerinin kullanım amaçlarını açıklayabilmeleri,
- Geliştirdikleri form uygulamalarında Visual Studio araç kutusunda yer alan form nesnelerini kullanabilmeleri,
- C# programlama dilinde kullanılan veri tipi, değişken ve sabit kavramlarını açıklamaları,
- Genel ve yerel değişkenler arasındaki farkları açıklayabilmeleri,

- C# programlama dilinde değişken ve sabit tanımlayabilmeleri,
- C# programlama dilinde karar yapılarını kullanabilmeleri,
- C# programlama dilinde *for, while* ve *foreach* döngülerini kullanabilmeleri,
- Değer döndüren ve döndürmeyen metotların farklarını açıklayabilmeleri,
- C# programlama dilinde metot tanımlayabilmeleri,
- Tanımladıkları metotları geliştirdikleri yazılımlarda kullanabilmeleri,
- C# programlama dilinde dizi tanımlayabilmeleri,
- Tanımladıkları dizileri geliştirdikleri yazılımlarda kullanabilmeleri,
- Form uygulamaları geliştirirken karşılaştıkları istisnai durumları yönetebilmeleri hedeflenmektedir.

Görsel Programlama ve Algoritma ve Programlamaya Giriş derslerinin kazanımları incelendiğinde NYP dersini almadan önce öğrencilerin algoritma kavramı, akış şeması, algoritma tasarımı, C# programlama dilinde veri tipleri, değişkenler, karar yapıları, döngüler, metotlar, diziler ve istisnai durum yönetimi gibi konularda bilgi ve beceri sahibi oldukları görülmektedir. Bu eğitimlerin sayesinde öğrencilerin belirli bir programlama altyapısı kazanarak NYP dersinde sınıf, nesne, özellik, davranış, kurucu metot, kapsülleme, arayüz, kalıtım ve çokbiçimlilik gibi nesne yönelimli programlama kavramlarını öğrenebilmeleri ve bunları programlama problemlerinin çözümünde kullanabilmeleri beklenmektedir.

4.1.2.2.3. Görev analizi

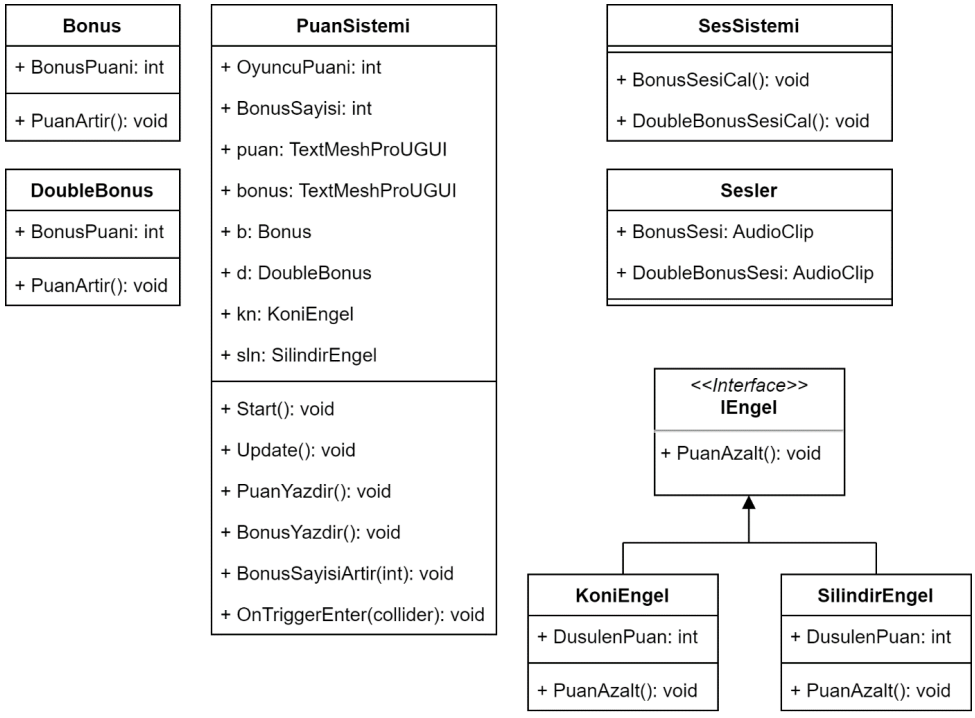
Görev Analizi aşamasında NYP dersinin hangi içerikleri kapsayacağı ve bu içeriklerin oyun geliştirmeye dayalı öğretim stratejisine uyumlu olarak hangi tür görevleri içereceği konusunda detaylı araştırmalar yapılmıştır. Bu aşamada alanyazın taraması yapılarak ders konu kapsamı üzerinde çalışmalar yapılmıştır. Alanyazın taramasına ilave olarak mevcut NYP dersinin içeriği, Türkiye'deki üniversitelerin Bilgisayar Programcılığı ön lisans programlarında benzer isimlerde okutulan NYP derslerinin içeriğiyle karşılaştırılmıştır. Bu karşılaştırmanın detayları Bölüm 4.1.1.1.'de sunulmuştur. Bu karşılaştırmanın sonunda mevcut NYP dersinin ders içeriğinin diğer üniversitelerin ders içerikleriyle büyük oranda uyumlu olduğu görülmüştür.

İçerik analizinin yanında dersin oyun geliştirmeye dayalı öğretim stratejisine nasıl entegre edileceği konusunun üzerinde de durulmuştur. Bu bağlamda, ders materyali olarak uzman görüşü alınarak öğrencilerin hazırbulunuşluk seviyelerine uygun ve Unity 3D oyun motoru

kullanılarak geliştirilmiş Karting Microgame isminde bir oyun prototipi belirlenmiştir (Şekil 4.16).

Şekil 4.16. Ders materyali olarak kullanılacak Karting Microgame oyun prototipi

Oyun prototipinin belirlenmesinden sonra öğrencilerin dersin içeriğine ve hedeflerine uygun olarak oyun prototipinde hangi değişiklikleri gerçekleştirecekleri ve oyun prototipine hangi modları ekleyecekleri *(puan sistemi, ses sistemi vb.)* belirlenmiştir. Gerçekleştirilecek bu modlara ait UML sınıf diyagramları Şekil 4.17'de sunulmuştur.



Şekil 4.17. Oyun prototipine eklenecek modlara yönelik hazırlanan UML sınıf diyagramları

Şekil 4.17'de görüldüğü üzere oyun prototipine *bonus sınıfları, engel sınıfları, puan sistemi sınıfı, sesler ve ses sistemi sınıflarının* eklenmesi planlanmıştır. Bunun yanında oyun prototipine *IEngel* isminde bir arayüz de eklenecek; kalıtım ve çokbiçimlilik konuları bu arayüz kullanılarak anlatılacaktır. Bu oyun modları eklendikten sonra oyun prototipinin Şekil 4.18'de bir örneği gösterilen son halini alması tasarlanmıştır.

Şekil 4.18. Geliştirmesi planlanan oyun prototipinin son halinin örnek bir gösterimi

4.1.2.2.4. Öğretimsel hedefler

Öğretimsel hedefler; NYP dersinin içeriğine yönelik çerçevenin çizilmesi, dersin sonunda elde edilecek bilgi ve becerilere yönelik sonuçlara rehberlik edilmesi ve başarı için standartların tanımlanması açısından önemlidir. NYP dersinin sonunda öğrencilerin;

- Nesne yönelimli programlamanın kullanım amaçlarını açıklayabilmeleri,
- NYP için mevcut olan programlama araçlarını listeleyebilmeleri,
- Sınıf, nesne, özellik, davranış, kurucu metot, nesneler arası mesajlaşma, erişim belirleyicisi gibi NYP'nin temel kavramlarını ifade edebilmeleri,
- Sınıf ile nesne arasındaki ilişkiyi betimleyebilmeleri,
- Sınıf ile nesne arasındaki ilişkiye yönelik örnekler verebilmeleri,
- UML diyagramlarının kullanım amaçlarını belirtebilmeleri,
- Oluşturdukları sınıfların UML diyagramlarını çizebilmeleri,
- UML diyagramı verilen bir sınıfı oluşturabilmeleri,
- Kapsülleme, arayüz, kalıtım ve çokbiçimlilik gibi NYP kavramlarını açıklayabilmeleri,

- Çokbiçimlilik prensibine yönelik örnekler verebilmeleri,
- Kapsüllemenin kullanım amaçlarını listeleyebilmeleri,
- Statik sınıfların kullanım amaçlarını ifade edebilmeleri,
- Statik sınıflarla statik olmayan sınıfları karşılaştırabilmeleri,
- C#'ta kullanılan erişim belirleyicileri listeleyebilmeleri,
- Farklı erişim belirleyicilerinin görevlerini açıklayabilmeleri,
- Dijital oyun türlerini sınıflandırabilmeleri,
- Oyun motorlarının kullanım amaçlarını belirtebilmeleri,
- Unity 3D editörünün temel kavramlarını (GameObject, Component, Collider, Rigidbody, Camera, Scene, Prefab, Hierarchy, Parenting, Tag vb.) açıklayabilmeleri,
- Unity 3D'de üç boyutlu bir oyun projesi oluşturabilmeleri,
- Oluşturdukları oyun projesine mevcut Unity 3D asset'lerini ekleyebilmeleri,
- Unity 3D'de sınıf oluşturabilmeleri,
- Oluşturdukları sınıflara ait statik olan ve olmayan özellikleri ekleyebilmeleri,
- Özelliklerin (sınıf değişkenlerinin) tanımlanmasında kapsülleme mekanizmasını kullanabilmeleri,
- Tanımladıkları sınıf değişkenlerinin erişim belirleyicilerini belirleyebilmeleri,
- Oluşturdukları sınıflara ait sınıf metotlarını tasarlayabilmeleri,
- Oluşturdukları sınıflara Monobehaviour metotları ekleyebilmeleri,
- Önceden oluşturulmuş sınıfları oyun projelerinde kullanabilmeleri,
- Oluşturdukları sınıfları oyun nesnelerine bileşen (component) olarak ekleyebilmeleri,
- Önceden oluşturulmuş sınıfların sınıf değişkenlerine ait erişim belirleyicilerini değiştirebilmeleri,
- UML diyagramı verilen bir arayüzü oluşturabilmeleri,
- Oluşturduğu arayüzlere ait çokbiçimli metotları tasarlayabilmeleri,
- Oyun projesinde arayüz aracılığı ile kullanılacak kalıtım mekanizmasını tasarlayabilmeleri,
- UML diyagramı verilen kalıtım mekanizmasını oyun projesinde kullanabilmeleri,
- Oyun projesi geliştirme aşamasında karşılaştıkları hataları giderebilmeleri,
- Proje sunma bilgi ve becerisi edinebilmeleri hedeflenmektedir.

4.1.2.2.5. İçeriğin düzenlenmesi

Öğrencilerin NYP dersinde hangi içerikler hakkında bilgi sahibi olacakları, hangi becerileri kazanacakları ve görevleri başaracakları belirlendikten sonra, bu içeriklerin sunumu için en uygun sıra uzman görüşleri alınarak belirlenmiştir. NYP dersinin haftalık öğretim programının özeti Tablo 4.17'de sunulmuştur. Geliştirilen NYP dersi, mevcut NYP dersinde olduğu gibi 40 dakikalık bir teorik ve iki uygulama olmak üzere haftada toplam üç ders olacak şekilde tasarlanmıştır. Tablo 4.17'de sunulduğu üzere geliştirilen NYP dersinin ilk üç haftasında öğrencilerin kendi oyun prototiplerini geliştirmek için dönem boyunca kullanacakları Unity 3D oyun motorunun kullanımına yönelik konular yer almaktadır. Dördüncü haftadan itibaren NYP'nin temelleri ve paradigması ile ilgili konulara yer verilmiştir. Teorik saatlerde ilgili haftanın konusunun ve alt konu başlıkları anlatılması; uygulama saatlerinde ise NYP ile ilgili tekniklerin ders materyali olarak kullanılan oyun prototipi üzerinde gerçekleştirilmesi planlanmıştır. Tablo 4.17'de koyu olarak yazılan alt konu başlıkları, öğrencilerin oyun prototipi üzerinde gerçekleştireceği programlama etkinliklerini ifade etmektedir.

Tablo 4.17. Geliştirilen NYP dersinin haftalık öğretim programının özeti

Hafta No	**Konu**	**Alt Konu Başlıkları**
1	Unity 3D Editörü Tanıtımı	▪ Unity 3D ve Unity Hub Kurulumu ▪ Unity Hub Yazılımı Tanıtımı ▪ Unity 3D'de Yeni Proje Oluşturma ▪ Unity 3D Editörü Tanıtımı ▪ Unity'de Layout'lar ▪ Transform Araçları ▪ Component, Camera, Scene, Asset, Material, Prefab ve Rigidbody Kavramları ▪ Nesnelere C# Kodu Ekleme ▪ Monobehaviour Metotlarının Tanıtımı
2	Unity 3D Playground Projesi Oluşturma-1	▪ Unity Asset Store Tanıtımı ▪ Unity Playground Platformunun Tanıtımı ▪ Unity Playground Kod Parçacıklarının (Script) Tanıtımı ▪ Yeni Bir Playground Projesi Oluşturma ▪ Sorting Layer Kavramı ▪ Nesnelere Hazır Kod Parçaları Ekleme ▪ Parenting ve Childing Kavramları ▪ Particle Kavramı ▪ Collider'ların Kullanımı ▪ FPS Kavramı

(Devam ediyor)

Tablo 4.17. Geliştirilen NYP dersinin haftalık öğretim programının özeti (Devam)

3	Unity 3D Playground Projesi Oluşturma-2	▪ Tag, Projectile, Trigger, Attribute Kavramları ▪ Prefab Oluşturma ve Kullanma ▪ Can Sistemi (Health System) Kullanımı ▪ Lazer Sistemi (Laser System) Kullanımı ▪ Puan Sistemi (Point System) Kullanımı ▪ Kullanıcı Arayüz (UI) Nesnelerinin Tanıtımı ve Kullanımı ▪ Level Tasarımı
4	Nesne Yönelimli Programlamanın Temelleri	▪ Nesne Yönelimli Programlama Paradigmasının Amaçları ▪ Sınıf ve Nesne Kavramları ▪ Özellik (Attribute) ve Davranış (Behaviour) Kavramları ▪ Kurucu Metot Kavramı ▪ C#'ta Kullanılan Erişim Belirleyicileri ▪ Nesneler Arası Mesajlaşma Kavramı ▪ UML Sınıf Diyagramları
5	Sınıflar ve Nesneler	▪ Unity 3D'de Sınıf ve Nesnelerin Kullanımı ▪ Roll-A-Ball Oyununun Sınıf ve Nesneler Açısından Çözümlenmesi ▪ Karting Microgame Prototipi Tanıtımı ▪ Yeni Bir Unity 3D Projesi Oluşturma ▪ **Karting Microgame Oyun Prototipini Projeye Ekleme** ▪ **Karting Microgame Oyun Prototipine Çeşitli Unity 3D Assetleri Ekleme**
6	Sınıflar ve Nesneler	▪ Oyun Prototipinde Sınıf Oluşturma **(Puan Sistemi Sınıfı)** ▪ Oluşturulan Sınıfların UML Sınıf Diyagramlarını Çizme
7	Sınıflar ve Nesneler	▪ Oyun Prototipinde Sınıf Oluşturma **(Bonus ve Double Bonus Sınıfları)** ▪ **Oyun Prototipine UI Elementleri Ekleme** Oluşturulan Sınıfların UML Sınıf Diyagramlarını Çizme
8	Sınıf Metotları	▪ Oyun Prototipinde Sınıf Metotları Oluşturma ve Kullanma **(PuanYazdir() ve BonusYazdir() Metotları)** ▪ Başka Sınıfların Metotlarını Kullanma (**PuanArtir() Metodu**)
9	Sınıf Metotları	▪ Oyun Prototipinde Sınıf Metotları Olarak Monobehavior Metotlarını Oluşturma ve Kullanma **(Start(), Update(), FixedUpdate(), OnTriggerEnter() ve BonusYazdir() Metotları)**
10	Veri Yapıları	▪ Statik Özellik ve Sınıf Kavramları ▪ Statik Sınıflarla Normal Sınıfların Karşılaştırması ▪ Oyun Prototipinde Statik Metotların Tanımlanması ve Kullanılması **(Ses Sistemi ve Sesler Sınıfları)**

(Devam ediyor)

Tablo 4.17. Geliştirilen NYP dersinin haftalık öğretim programının özeti (Devam)

11	Arayüzler ve Kalıtım	▪ Kalıtım (Inheritance) Kavramı ▪ Üst-Sınıf (Superclass) ve Alt-Sınıf (Subclass) Kavramları ▪ Soyut (Virtual) Sınıflar ▪ Oyun Prototipinde Arayüz Tanımlama **(IEngel Sınıfı)** ▪ Oyun Prototipinde Kalıtım Kullanma **(Koni Engel ve Silindir Engel Sınıfları)**
12	Kapsülleme ve Çokbiçimlilik	▪ Veri Gizleme İlkesi ▪ Görünebilirlik Kuralları ▪ Çokbiçimlilik Kavramı ▪ Oyun Prototipinde Çokbiçimli Metotları Tanımlama ve Kullanma **(PuanAzalt() Metodu)**
13	Hata Kotarımı Görsel Efektler ve Pist Tasarımı	▪ Oyun Prototipinde Karşılaşılan Hatalar ve Çözümleri ▪ Oyun Prototipine Görsel Efekt Ekleme **(Bounce, SpeedPad, Particle)** ▪ **Pist Tasarımını Değiştirme**
14	Öğrenci Proje Sunumları	▪ Öğrencilerin Hazırladıkları Oyun Prototiplerinin Tanıtımı
15	Öğrenci Proje Sunumları	▪ Öğrencilerin Hazırladıkları Oyun Prototiplerinin Tanıtımı

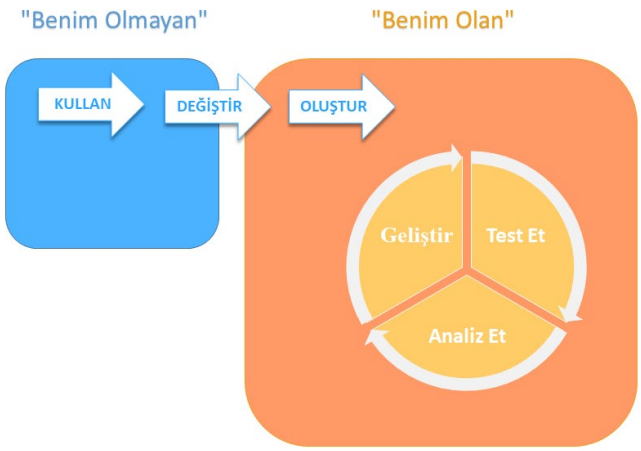
4.1.2.2.6. Öğretim stratejileri

Geliştirilen NYP dersine ait hedeflerin başarılabilmesi, ders içeriğinin etkili ve verimli bir şekilde sunulabilmesi, bunun yanında öğrencilerin dersle ilgili görevleri başarabilmeleri için alanyazın ve uzman görüşlerini doğrultusunda bazı öğretim stratejileri belirlenmiştir. Bunlar (1) Oyun modu oluşturma (game modding) stratejisi, (2) kullan-değiştir-oluştur (use-modify-create) stratejisi ve (3) proje tabanlı öğrenmedir.

Oyun Modu Oluşturma: NYP dersinde oyun modu oluşturma (game modding) stratejisinin kullanılmasının temel nedenlerin birisi öğrencilerin sıfırdan bir oyun geliştirirken karşılaşacakları öğrenme eğrisi ve teknik sorunları ortadan kaldırmaktır (Grizioti ve Kynigos, 2018; Sotamaa, 2010). Bu strateji sayesinde öğrenciler daha önce çalışan bir oyun prototipini kullanarak kendi oyunlarını tasarlayacaklar ve nesne yönelimli programlama paradigmasını bu tasarım sürecinde kullanabileceklerdir.

Kullan-Değiştir-Oluştur: Bu strateji sayesinde öğrencilerin daha önce geliştirilen bir oyunun kullanıcısı olma profilinden kendi oyunlarının yaratıcısı olma profiline dönüşmeleri amaçlanmıştır (Martin ve diğerleri, 2020). Şekil 4.19'da bir modeli sunulan bu stratejinin *kullan*

aşamasında öğrenciler başka yazılımcılar tarafından daha önce oluşturulmuş olan Karting Microgame oyun prototipini kullanmaya; başka bir ifade ile Karting Microgame oyununu oynamaya başlamaktadır. Öğrencilerin bu oyunu oynarken aynı zamanda oyunun nasıl programlandığını, hangi nesne ve sınıflara sahip olduğunu inceleme şansı edinmektedir. Bu sayede oyun dinamikleri ve kodlarını daha yakından tanımakta, oyunla ilgili kendilerine güvenlerini kazanmaktadırlar (Boulden ve diğerleri, 2021). *Değiştir* aşaması, öğrencilerin oyun prototipine ekleyeceği sınıflar ve nesneler vasıtasıyla yaptıkları küçük iyileştirmelerin oyunlarını nasıl etkileyeceğini keşfettikleri geçiş aşaması olarak tanımlanabilir. *Oluştur* aşamasında ise öğrenciler oyun prototiplerine dersin amaçları ve kendi istekleri doğrultusunda puan sistemi, engel sistemi, ses sistemi gibi yeni modlar eklemektedir. Bu süreçte öğrenciler "*Benim Olmayan"* bir oyundan *"Benim Olan"* (Lee ve diğerleri, 2011) bir oyuna doğru geçiş yapmaktadır.

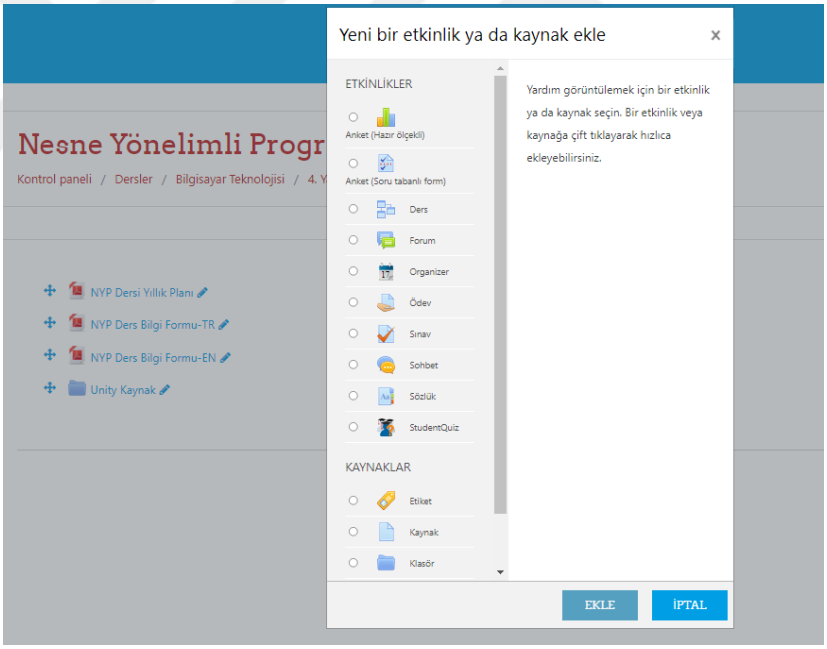


Şekil 4.19. Kullan-değiştir-oluştur stratejisi (Lee ve diğerleri, 2011)

Proje Tabanlı Öğrenme: Proje tabanlı öğrenme stratejisi kullanılarak NYP dersinde öğrencilerin kişisel olarak anlamlı buldukları oyun projeleri üzerinde çalışmaları esnasında özgün meydan okumalar içeren karmaşık görevlere katılmalarının sağlanması amaçlanmaktadır (Holm, 2011). Bu strateji kullanılarak öğrencilerin oyun projelerini geliştirmeleri aşamasında otantik öğrenme çerçevesinde yeni fikirler üretmelerine, problem çözme ve hata giderme becerilerinin geliştirilmesine katkıda bulunulacaktır.

4.1.2.2.7. Öğretimin sunumu

Geliştirilen NYP dersinin, mevcut NYP dersinde olduğu gibi yüz yüze olarak sunulması planlanmıştır. Benzer şekilde mevcut NYP dersinde olduğu gibi öğretim materyallerinin ve ders etkinliklerinin öğrencilerle paylaşılmasında MOODLE tabanlı öğrenme yönetim sisteminin (Şekil 4.20) kullanımına devam edilmesine karar verilmiştir. Ancak öğrenme yönetim sisteminde sunulan ders içerikleri, geliştirilen NYP dersi öğretim programı dikkate alınarak yeniden düzenlenmiştir. Düzenlenen bu öğrenme yönetim sisteminde ders videoları, ders notları, örnek programlama projeleri ve sunumları gibi ders materyalleri, ders izlencesi (syllabus), derse ait mini sınavlar (quizler), ödevler, dış bağlantı kaynakları, dersin katılımcıları, takvim, öğrenci notları gibi bölümler yer almaktadır. Bu içeriklere ilave olarak, ders portalına öğrencilerin dijital günlüklerini öğretim elemanına gönderebilecekleri bir forum da eklenmiştir.



Şekil 4.20. Geliştirilen NYP dersi için tasarlanan ders portalı

4.1.2.2.8. Değerlendirme araçları

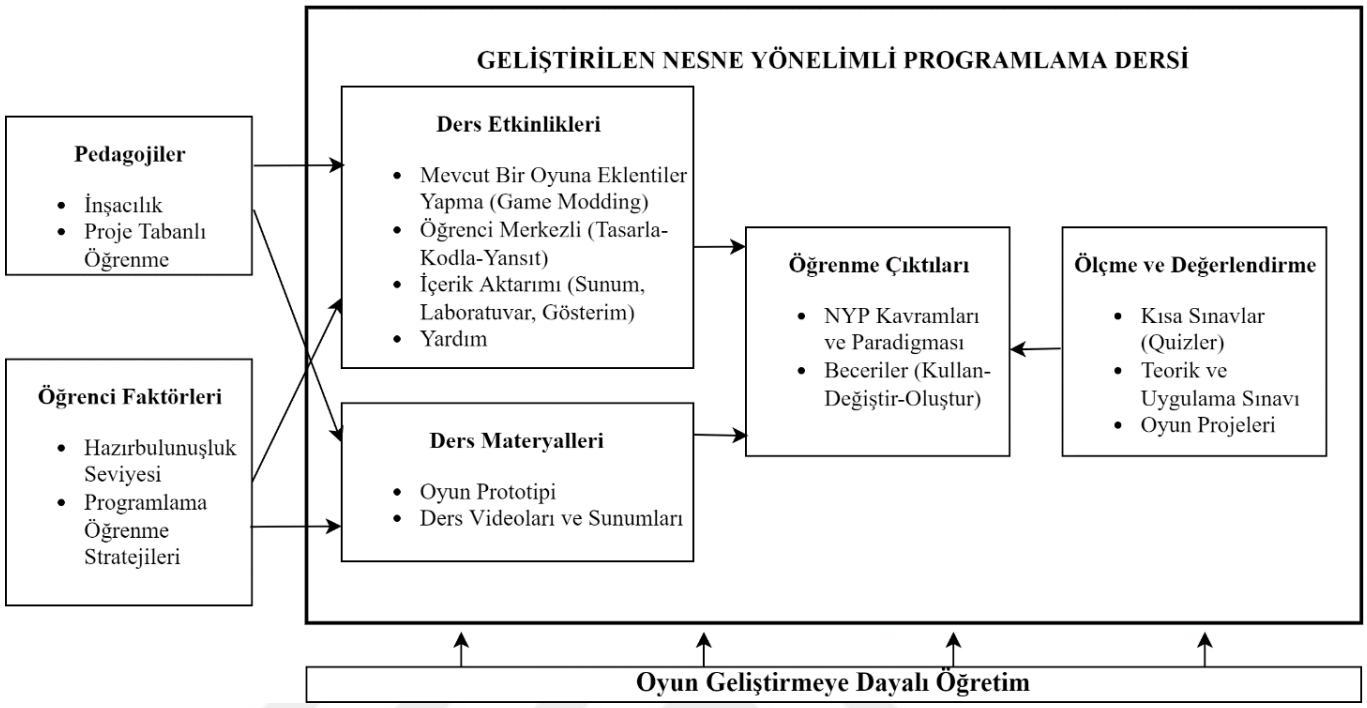
Dersin amaçlarına ulaşılıp ulaşılmadığı konusunda yargıda bulunmak amacıyla derse ait değerlendirme araçları yeniden tasarlanmıştır. Geliştirilen NYP dersinde öğrencilerin kısa sınavlar (quizler), başarı sınavı ve bireysel oyun projeleri olmak üzere üç farklı kritere göre değerlendirilmesi planlanmıştır. Bu ölçme araçlarının ağırlıkları Tablo 4.18'de sunulmuştur.

Tablo 4.18. Kısa sınavların, başarı sınavının ve projelerin ağırlıkları

Değerlendirme Türü		Ders Başarısına Etkisi
Kısa Sınavlar	Quiz-1	%5
	Quiz-2	%5
Başarı Sınavı	Teorik Sınav	%20
	Uygulama Sınavı	%20
Bireysel Oyun Projeleri	-	%40
Derse Katılım	-	%10

4.1.2.2.9. Geliştirilen NYP dersinin kavramsal çerçevesi

Yukarıda açıklanan öğretim tasarımı süreci, geliştirilen NYP dersinin kavramsal çerçevesinin Şekil 4.21'de yer aldığı biçimde ortaya çıkmasını sağlamıştır. Bu kapsamda, öğrenci faktörleri ile inşacılık ve proje tabanlı öğrenme pedagojileri ders etkinliklerinin ve materyallerinin karakteristik özelliklerinin çerçevesini çizmiştir. Ders etkinlikleri ve materyallerinin inşacılık temelinde hazırlanması sürecinde Brennan'ın (2015) *tasarım, kişiselleştirme, paylaşım ve yansıtma* aşamalarını içeren öğrenme süreci tasarımı modeli yol gösterici olmuştur. Bununla birlikte ders etkinlikleri ve materyalleri öğrenme çıktılarının yapısını oluşturmuş; aynı zamanda öğrencilerinden edinmeleri beklenen kavramsal bilgilerle programlama becerilerine temel teşkil etmiştir. Bu bütünsel öğretim tasarımı yaklaşımı; dersin öğrenme çıktılarını, ölçme ve değerlendirme yaklaşımını, ders etkinliklerini ve medyasını öğrencilerin hazırbulunuşluk seviyeleri ve programlamayı nasıl öğrendiklerine dair bildiklerimizle birleştirerek bunları oyun geliştirmeye dayalı öğrenme yaklaşımı çerçevesinde temellendirmeyi amaçlayan bir girişim niteliği taşımaktadır. Bu yeni ders tasarımıyla birlikte öğrencilerin programlama öğrenme yaklaşımlarının etkilenmesi, programlamaya yönelik algılarının olumlu yönde değişmesi ve NYP dersi deneyimlerinde farklı türde varyasyonlar sağlayan bir öğrenme bağlamının yaratılması planlanmıştır.



Şekil 4.21. Geliştirilen NYP dersinin kavramsal çerçevesi (Thota ve Whitfield'dan (2010) uyarlanmıştır.)

4.1.3. Uygulama Aşaması

4.1.3.1. Birinci döngüde uygulanan öğretim programı

NYP dersinin haftalık öğretim programı, öğretim tasarım sürecinin bir parçası olarak pandemiden önce oluşturulmuş ve özeti daha önce Tablo 4.17'de sunulmuştur. Ancak pandemi şartları nedeniyle hazırlanan bu öğretim programının sekiz hafta ile sınırlandırılması gerekmiştir. Bu açıdan değerlendirildiğinde, daha önce Morrison ve diğerlerinin (2004) öğretim tasarımı modeline uygun olarak gerçekleştirilen öğretim tasarımının *öğretimsel hedefler, içeriğin düzenlenmesi ve görev analizi* aşamalarında uzman görüşleri ışığında değişikliklerin yapıldığı söylenebilir. Araştırmanın birinci döngüsünde uygulanan bu öğretim programının özeti Tablo 4.19'da verilmiştir.

Tablo 4.19. Geliştirilen NYP dersinin haftalık öğretim programının özeti

Hafta No	Konu	Alt Konu Başlıkları
1	Unity 3D Editörü Tanıtımı	▪ Unity 3D ve Unity Hub Kurulumu ▪ Unity Hub Yazılımı Tanıtımı ▪ Unity 3D'de Yeni Proje Oluşturma ▪ Unity 3D Editörü Tanıtımı ▪ Unity'de Layout'lar ▪ Transform Araçları ▪ Component, Camera, Scene, Asset, Material, Prefab ve Rigidbody Kavramları ▪ Nesnelere C# Kodu Ekleme ▪ Monobehaviour Metotlarının Tanıtımı
2	Nesne Yönelimli Programlamanın Temelleri	▪ Nesne Yönelimli Programlama Paradigmasının Amaçları ▪ Sınıf ve Nesne Kavramları ▪ Özellik (Attribute) ve Davranış (Behaviour) Kavramları ▪ Kurucu Metot Kavramı ▪ C#'ta Kullanılan Erişim Belirleyicileri ▪ Nesneler Arası Mesajlaşma Kavramı ▪ UML Sınıf Diyagramları
3	Sınıflar ve Nesneler	▪ Unity 3D'de Sınıf ve Nesnelerin Kullanımı ▪ Roll-A-Ball Oyununun Sınıf ve Nesneler Açısından Çözümlenmesi ▪ Karting Microgame Prototipi Tanıtımı ▪ Yeni Bir Unity 3D Projesi Oluşturma ▪ **Karting Microgame Oyun Prototipini Projeye Ekleme** ▪ **Karting Microgame Oyun Prototipine Çeşitli Unity 3D Assetleri Ekleme**
4	Sınıflar ve Nesneler	▪ Oyun Prototipinde Sınıf Oluşturma **(Puan Sistemi, Bonus ve Double Bonus Sınıfları Sınıfları)** ▪ Oluşturulan Sınıfların UML Sınıf Diyagramlarını Çizme
5	Sınıf Metotları	▪ Oyun Prototipinde Sınıf Metotları Olarak Monobehavior Metotlarını Oluşturma ve Kullanma **(Start(), Update(), FixedUpdate(), OnTriggerEnter() ve BonusYazdir() Metotları)**
6	Sınıf Metotları	▪ Oyun Prototipinde Sınıf Metotları Oluşturma ve Kullanma **(PuanYazdir() ve BonusYazdir() Metotları)** ▪ Başka Sınıfların Metotlarını Kullanma (**PuanArtir() Metodu**)
7	Veri Yapıları	▪ Statik Özellik ve Sınıf Kavramları ▪ Statik Sınıflarla Normal Sınıfların Karşılaştırması ▪ Oyun Prototipinde Statik Metotların Tanımlanması ve Kullanılması **(Ses Sistemi ve Sesler Sınıfları)**
8	Kapsülleme ve Çokbiçimlilik	▪ Veri Gizleme İlkesi ▪ Görünebilirlik Kuralları ▪ Çokbiçimlilik Kavramı ▪ Oyun Prototipinde Çokbiçimli Metotları Tanımlama ve Kullanma **(PuanAzalt() Metodu)**

4.1.3.2. Araştırma Sorusu 7: Eylem planının uygulanması sürecinde katılımcıların ve araştırmacının yaşadığı sorunlar nelerdir?

Hazırlanan eylem planının uygulanması aşamasında öğrenciler ve öğretici olan araştırmacı bazı sorunlarla karşılaşmıştır. Öğrencilerin karşılaştıkları sorunların tespitinde derslerdeki gözlemlerin yanında dijital öğrenci günlüklerinden de faydalanılmıştır. Uygulanma aşamasında karşılaşılan sorunlara yönelik anlık çözümler üretebilmek için uzman görüşleri doğrultusunda bazı eylem kararları alınmıştır. Eylem planının uygulanması aşamasında karşılaşılan sorunlar ve bu sorunların çözümüne yönelik alınan eylem kararları Tablo 4.20'de verilmiştir. Tablo 4.20'de gösterildiği üzere eylem planının uygulanması aşamasında karşılaşılan sorunlar (1) kodlama hataları, (2) zaman kısıtlaması, (3) konuların karmaşıklığı, (4) öğrenci günlükleri, (5) yazılım uyumsuzluğu, (6) öğrenci devamsızlığı ve (7) konu dizilimi olmak üzere yedi kategori altında toplanmıştır.

Tablo 4.20. Birinci döngünün uygulanması aşamasında ortaya çıkan sorunlar ve sorunların çözümüne yönelik alınan eylem kararları

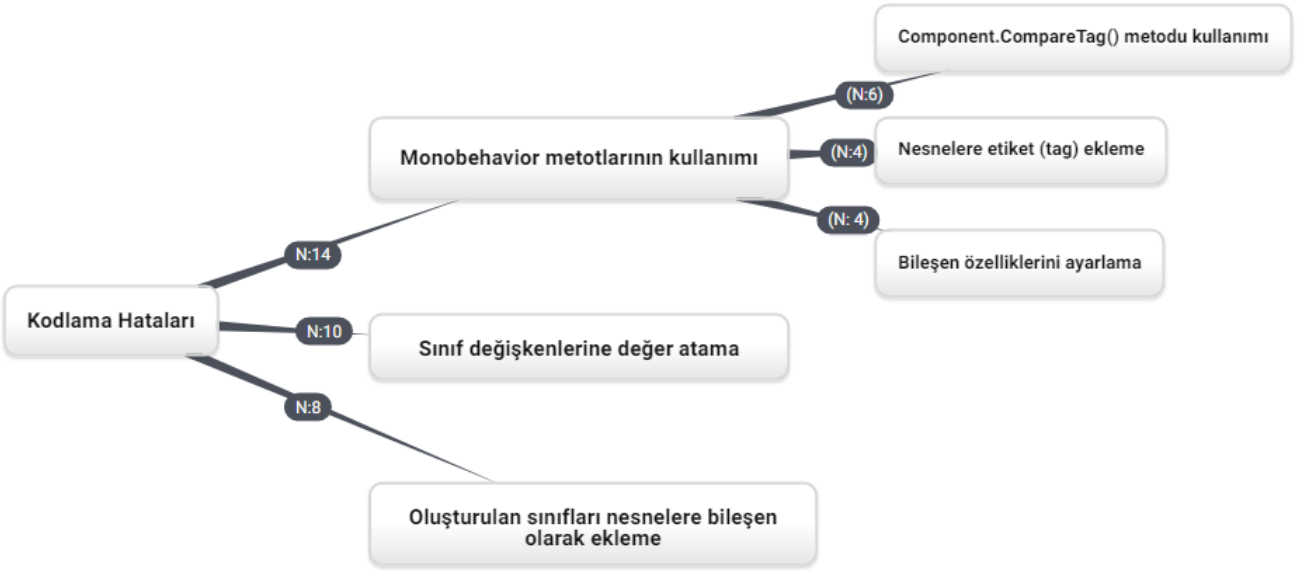
Kategori	**Uygulama Aşamasında Ortaya Çıkan Sorunlar**	**Sorunların Çözümüne Yönelik Alınan Eylem Kararları**
Kodlama Hataları	Öğrencilerin Monobehavior metotlarını kullanırken metot parametreleri konusunda hatalarla karşılaştıkları fark edilmiştir.	Monobehavior metotlarının parametreleriyle ilgili çözülmüş örnekler hazırlanarak derste canlı kod gösterimleri yapılmıştır.
	Sınıf değişkenlerine değer ataması ve nesneleri ilişkilendirme konularında öğrencilerin sıklıkla hatalarla karşılaştıkları tespit edilmiştir.	Öğrencilere oyun projelerinde kullandıkları sınıfların UML diyagramları hazırlatılarak sınıflar arasındaki ilişkilerin gösterildiği bir etkinlik düzenlenmiştir.
	Öğrencilerin sınıfları oluşturdukları ancak bunu ilgili nesnelere bileşen (component) olarak eklemeyi unuttukları tespit edilmiştir.	Hangi oyun nesnesinin hangi sınıfa (bileşene) sahip olması gerektiğini içeren nesne-bileşen ilişki modellerinin hazırlanmasını içeren bir etkinlik düzenlenmiştir.
Zaman Kısıtlaması	Unity 3D'de nesnelere C# kodu ekleme ve Monobehaviour metotları konuları için ayrılan iki ders saati yeterli olmamıştır.	▪ Ders portalına Monobehaviour metotlarının kullanımı konusunda eğitsel videolar eklenmiştir. ▪ İkinci döngü için revizyon kararı alınmıştır.
Konuların Karmaşıklığı	Unity 3D'de yer alan İngilizce kavramlar öğrenciler tarafından karmaşık olarak algılanmıştır.	▪ Proje geliştirme sürecinde Unity 3D'de yer alan İngilizce kavramlar sıklıkla açıklanmıştır. ▪ Unity 3D'de yer alan kavramlarla ilgili e-kitap öğrencilerle paylaşılmıştır.
	D1_K7, D1_K18 ve D1_K29 kodlu katılımcılar (N:3) projelerini bireysel olarak bitirmelerinin mümkün olmadığını belirtmişlerdir.	Bu öğrenciler, geçmiş programlama derslerindeki başarı durumuna göre başarılı olan öğrencilerle eşleştirilmiş ve projelerini birlikte tamamlamaları teşvik edilmiştir.

(Devam ediyor)

Tablo 4.20. Birinci döngünün uygulanması aşamasında ortaya çıkan sorunlar ve sorunların çözümüne yönelik alınan eylem kararları (Devam)

Kategori	Uygulama Aşamasında Ortaya Çıkan Sorunlar	Sorunların Çözümüne Yönelik Alınan Eylem Kararları
Öğrenci Günlükleri	Bazı öğrencilerin günlüklerinde yeterince detaya girmediği belirlenmiştir.	Öğrenciler günlükleri detaylı bir şekilde doldurmaları konusunda uyarılmıştır.
	Bazı öğrencilerin günlüklerini doldurmadıkları ya da önceki haftanın aynısını gönderdiği tespit edilmiştir.	Öğrencilere kendilerinden günlüklerle ilgili beklenenler konusunda açıklamalarda bulunulmuştur.
Yazılım Uyumsuzluğu	Visual Studio 2015 ile Unity 3D 2018 arasında uyum sorunları yaşanmıştır.	Bilgisayar dershanelerine Visual Studio Community 2015 yerine Visual Studio Community 2017 yazılımları kurulmuştur.
Öğrenci Devamsızlığı	Geçmiş haftalardaki dersleri kaçıran öğrencilerin projelerini tamamlarken zorlandıkları gözlenmiştir.	Ders portalına her hafta gerçekleştirilen ders uygulamalarının videolarının eklenmesi
Konu Dizilimi	Roll-A-Ball oyun prototipi bir hafta geç tanıtılmıştır.	İkinci döngü için revizyon kararı alınmıştır.

Bu sorun kategorilerinden ilki kodlama hatalarıdır. Kodlama hatalarının türleri ve bu hataların dağılımları Şekil 4.22'de verilmiştir. Şekil 4.22'de görüldüğü üzere kodlama hataları üç kategoriye ayrılmıştır.



Şekil 4.22. Kodlama hatalarının türleri ve dağılımı

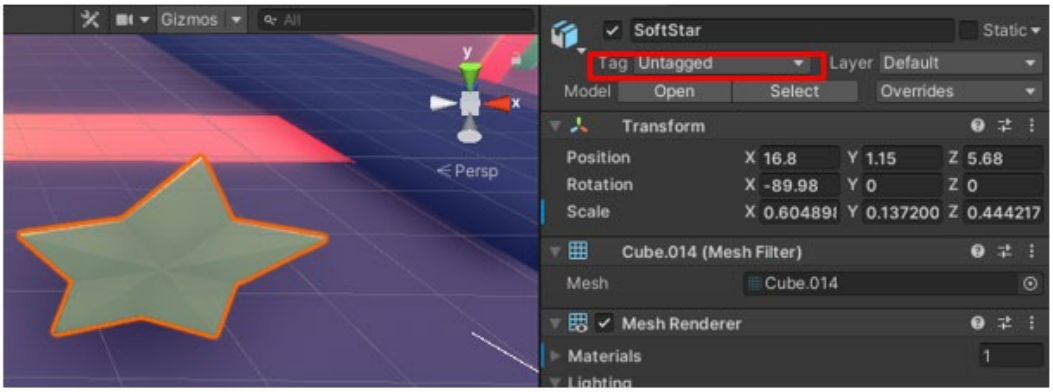
Şekil 4.22'de belirtildiği üzere katılımcıların sıklıkla yaptıkları ilk hata türü Monobehaviour metotlarının kullanımıyla ilgilidir. Bu metodun doğru kodlanmış hali Şekil 4.23'te sunulmuştur. Monobehaviour metotlarının kullanımında da üç farklı türde hatalar tespit edilmiştir. Bunlardan ilki Component.CompareTag() kullanımının yapılmamasıyla ilişkilidir. Uygulamanın beşinci

haftasında beş katılımcı, sürücü ile bonus ya da double bonus nesnelerinin çarpışıp çarpışmadığını kontrol ettirmek için hangi metodu kullanacaklarını, bu metodun hangi parametrelere sahip olduğunu hatırlayamamışlardır. Dolayısıyla bu katılımcılar Component.CompareTag() metodunu kullanmayı unuttukları için sürücü ile bonus nesneleri arasındaki çarpışmaları kontrol ettirememişlerdir. Bunun bir sonucu olarak bonus ve double bonus sınıflarına ait PuanArtir() metodu çalışmamıştır. İlgili öğrenciler araştırmacıdan yardım isteyerek çarpışma durumu kontrolü için kullanılacak metot hakkında bilgi talebinde bulunmuşlardır.

```
27      private void OnTriggerEnter(Collider other)
28      {
29          if (other.CompareTag("Bonus"))
30          {
31              b.PuanArtir();
32              other.gameObject.SetActive(false);
33          }
34
35          if (other.CompareTag("DoubleBonus"))
36          {
37              db.PuanArtir();
38              other.gameObject.SetActive(false);
39          }
40      }
41  }
```

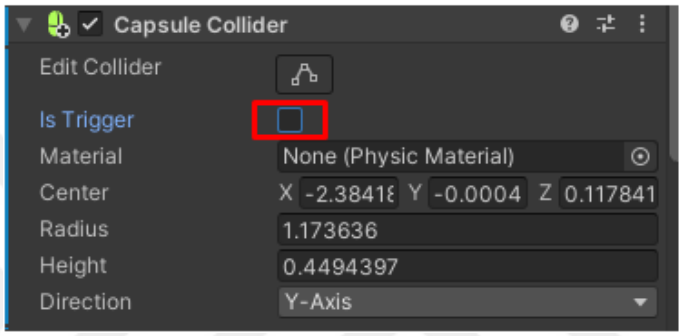
Şekil 4.23. Doğru olarak kodlanmış Monobehaviour metodu

Monobehaviour metotlarının kullanımında karşılaşılan ikinci hata türü nesnelere etiket (tag) eklenmemesinden kaynaklanmıştır. Uygulamanın beşinci haftasında dört katılımcı nesnelere tag eklemeyi unuttuğu için bonus ve double bonus sınıflarına ait PuanArtir() metodu çalışmamıştır. Nitekim bu durum D2_K21 kodlu katılımcının oyun projesinde Şekil 4.24'teki gibi yansımıştır.



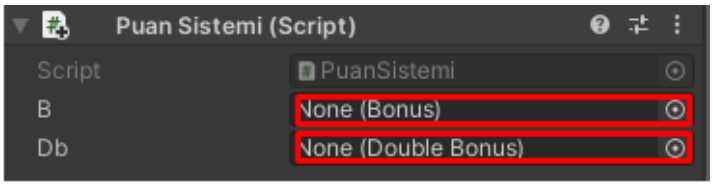
Şekil 4.24. D2_K21 kodlu katılımcının projesinden etiket (tag) eklenmemiş bir bonus nesnesi gösterimi

Monobehaviour metotlarının kullanımında karşılaşılan son hata türü nesnelerin bileşen özelliklerinden kaynaklanmıştır. Uygulamanın dördüncü haftasında dört katılımcı her ne kadar nesneler arasındaki çarpışmaların algılanabilmesi için bonus nesnelerine *capsule collider* bileşeni eklemiş olsalar da bu bileşenlere *Is Trigger* özelliği atamamışlardır. Nitekim bu durum D2_K3 kodlu katılımcının oyun projesinde Şekil 4.25'te belirtildiği gibi görülmüştür. Nesneler trigger özelliğine sahip olmadıkları için OnTriggerEnter metodu sürücü ile bonus nesneleri arasındaki çarpışmayı algılayamamış ve oyuncu puanını artıramamıştır.



Şekil 4.25. D2_K3 kodlu katılımcının projesinden Is Trigger özelliği eklenmemiş bir capsule collider bileşeni gösterimi

Öğrencilerin kodlama hataları kategorisinde sıklıkla yaptıkları ikinci hata türü sınıf değişkenlerine (özellik) değer atanmasıyla ilgilidir (Şekil 4.22). Uygulamanın dördüncü haftasında katılımcılar puan sistemi sınıfını oluşturmuşlardır. Ancak katılımcıların %33'ünün (N:10) bu sınıfın değişkenleri olan bonus ve double bonus türündeki değişkenlere değer atamayı unutmuşlar; dolayısıyla nesne referans hatasıyla karşılaşmışlardır. Nitekim bu durum D2_K15 kodlu katılımcının projesinde Şekil 4.26'daki gibi yer almıştır.



Şekil 4.26. D2_K15 kodlu katılımcının projesinden sınıf değişkenlerine değer atanmamış bir sınıf gösterimi

Katılımcıların yaygın olarak karşılaştıkları son hata kategorisi, oluşturulan sınıfların oyun nesnelerine bileşen olarak eklenmemesinden kaynaklanmıştır. Uygulamanın beşinci haftasında katılımcıların %27'si (N:8) bonus ve double bonus sınıflarını oyun nesnelerine bileşen olarak

eklememişlerdir. Bunun bir sonucu olarak bu katılımcıların oyunlarında çarpışma yaşanmasına rağmen bonus ve double bonus sınıflarına ait PuanArtir() metotları çağrılmadığı için oyuncu puanında herhangi bir değişiklik olmamıştır.

Uygulama sürecinde yaşanan sorunların ikincisi *zaman kısıtlamasıyla* ilgili olmuştur (Tablo 4.21). Bu sorun araştırmacının günlüğünde aşağıdaki şekilde yer almıştır:

> Eylem planının uygulanması aşaması yedi hafta süreceği için Unity 3D editörü tanıtımına ve Monobehavior metotlarına (start, update, fixedupdate vb.) yalnızca bir hafta ayırmak zorunda kaldım. Diğer haftalarda nesne yönelimli programlamanın temel kavramlarına geçip bunları oyun programlayarak uygulayacağız ve öğreneceğiz. Bu hafta her şeyi çok hızlı anlatmak zorunda kaldım. Hatta Unity 3D'de nesnelere C# kodu ekleme ve Monobehaviour metotları konuları kapsamında bir küp nesnesini her frame'de döndürme işlemini gerçekleştirdik. Ancak öğrencilerin bunu uygulamaları için sadece 15 dakikaları vardı. Bence bu süre yetersiz. (Araştırmacı günlüğü, 02.07.2020)

Araştırmacı günlüğünde de yer aldığı üzere pandemi sebebiyle eylem araştırmasının birinci döngüsünün uygulama aşaması sekiz hafta sürmüştür. Bu sebepten ötürü normalde üç hafta süre ayrılan Unity 3D editörü tanıtımı ve Unity 3D metotlarının (Monobehavior metotları) kullanımı konusu için sadece bir haftalık zaman ayrılabilmiştir. Ancak pandemi koşullarından önce hazırlanan dersin haftalık planında "Unity 3D'de nesnelere C# kodu ekleme ve Monobehaviour metotları" konuları için iki ders saati ayrılmış olmasına rağmen bu süre yeterli olmamıştır. Bu kapsamda bu konular için ayrılan süre bağlamında araştırmanın ikinci döngüsü için revizyon kararı alınmıştır.

Konuların karmaşıklığı konusu ile ilgili olarak araştırmacı Unity 3D'de yer alan kavramların açıklamaları içeren bir e-kitap hazırlayarak öğrencilerle paylaşmıştır. Buna ilave olarak bu kavramlar derslerde ve proje geliştirme sürecinde sıklıkla açıklanmıştır. Konuların karmaşıklığı temasıyla ilgili olarak alınan bir diğer eylem kararı ise projelerini tek başlarına bitirmelerinin mümkün olmayacağını ifade eden öğrencilerle geçmiş programlama derslerindeki başarı durumu yüksek başarılı olan öğrencilerin eşleştirilmesi ve akran öğrenmesine fırsat tanınması olmuştur. Bu sayede geçmiş programlama derslerindeki başarı durumu düşük başarılı olan öğrenciler başarılı öğrencilerle birlikte çalışmış, karşılaştıkları hataları birlikte çözerek projelerini birlikte tamamlamışlardır. Bu husus araştırmacı günlüğünde aşağıdaki gibi yer almıştır:

Acemi öğrenciler benden ve arkadaşlarından çok sık yardım talebinde bulunuyorlar. Konuları daha iyi öğrenebilsinler ve projelerini tamamlayabilsinler diye programlama konusunda deneyimli akranlarından yardım alabilmelerine olanak sağladım. Bazı öğrenciler arkadaşlarıyla birlikte çalışıyor, onların anlatımlarından faydalanıyorlar. (Araştırmacı günlüğü, 23.07.2020).

Dersin verimliliğini etkileyen bir diğer sorun teması olan *öğrenci devamsızlığı* konusudur. Bu sorun araştırmacı günlüğünde aşağıdaki şekilde ifade edilmiştir:

Bazı öğrencilerim geçen dersi kaçırmıştı. Bu durum benim için handikap çünkü bu haftaki dersi öğrenebilmeleri için geçen haftaki dersi bilmeleri gerekiyordu. Onları etütte diğer arkadaşları ile çalışmaları gerektiği konusunda yönlendirdim ancak bazılarını bunu gerçekleştirmemiş. Bu engeli aşmak için her dersin ekran görüntüsünü kaydetmeli ve okulun portalına yüklemeliyim. (Araştırmacı günlüğü, 16.07.2020).

Araştırmacı bu sorunun çözümüne yönelik olarak her hafta gerçekleştirilen ders uygulamalarının video kayıtlarını ders portalında öğrencilerle paylaşmıştır.

Uygulama esnasında karşılaşılan bir diğer sorun ise *öğrenci günlükleri* ile ilgilidir. Günlüklerin incelenmesi sonucunda bazı öğrencilerin günlüklerinde yeterli detaya yer vermedikleri veya geçen hafta yazdıklarının aynısını gönderdikleri tespit edilmiştir. Bu durumda yönelik örnekler aşağıda verilmiştir:

Dersten çok fazla bir şey anlamıyorum (D1_K29, 09.07.2020).

Bu derste neyi başardım? Başarı sayabileceğim bir şey olmadı. Bu derste neyi tam olarak anlayamadım? Anlamadığımı söyleyebileceğim bir şey olmadı (D1_K18,16.07.2020).

Öğrencilerin günlükleri araştırmacı tarafında haftalık olarak incelenmiş ve öğrenciler günlüklerini detaylı bir şekilde doldurmaları konusunda uyarılmıştır.

Yazılım uyumsuzluğu sorun temasıyla ilgili olarak uygulamanın ikinci haftasından itibaren Visual Studio Community 2015 yerine Visual Studio Community 2017 yazılımının kullanılması kararı alınmıştır ve bu sorun çözülmüştür. *Teknik imkânlar* sorunun ana nedeni bilgisayar dershanelerinin internet bağlantılarının yetersiz olması ve bazı güvenlik kısıtlamalarıdır. Her ne kadar araştırmacı asset veya kaynak paylaşımı gibi konularda çözüm üretse de öğrencilerin ders dışı zamanlarda dershaneleri daha fazla kullanma taleplerini karşılayacak çözüm üretememiştir.

Sonuç olarak Tablo 4.21'de yer alan sorunların bazıları için çözüm kararları alınmış, bazıları için araştırmanın ikinci döngüsünü kapsayacak şekilde revizyon kararları almış ancak bazıları için çözüm üretilememiştir. Revizyon kararı alınan sorunlar tabloda yer alan sekiz sorun

temasından *zaman kısıtlaması* ve *konu dizilimi* temalarında yer almıştır. Zaman kısıtlaması temasında Monobehavior metotları ve kullanıcı arayüzü nesnelerinin kullanımı konularının; konu dizilimi temasında ise Roll-A-Ball oyun prototipinin incelenmesi konusunun işlenmesiyle ilgili revizyon kararı alınmıştır.

4.1.4. Değerlendirme Aşaması

4.1.4.1. Araştırma Sorusu 8: Eylem planının uygulanmasından sonra katılımcıların nesne yönelimli programlama bilgi ve beceri düzeylerinde nasıl bir değişim meydana gelmiştir?

Katılımcıların uygulama öncesi ve sonrasına ait programlama bilgi ve becerileri NYP Başarı Sınavı kullanılarak belirlenmiştir. Katılımcıların NYP Başarı Sınavı öntest ve sontest puanlarına yönelik betimsel istatistikler Tablo 4.21'de sunulmuştur.

Tablo 4.21. NYP Başarı Sınavı öntest ve sontest puanlarına yönelik betimsel istatistikler

	Öntest			Sontest		
	N	**$\bar{x}$**	**ss**	**$\bar{x}$**	**ss**	**Fark**
NYP Başarı Sınavı	29	5.18	4.11	67.56	19.83	62.38

Buna ilave olarak katılımcıların öntest ve sontest puanları istatistiksel olarak karşılaştırılmıştır. Bu karşılaştırmayı gerçekleştirmek maksadıyla öncelikle test sonuçlarının normal dağılıp dağılmadığı kontrol edilmiştir. Shapiro-Wilk testi sonuçlarına göre katılımcıların öntest ve sontest puanları normal dağılmamaktadır ($p<.05$). Bu sebeple katılımcıların NYP Başarı Sınavı öntest ve sontest puanları karşılaştırmak amacıyla Wilcoxon sıralı işaretler testi kullanılmıştır (Tablo 4.22).

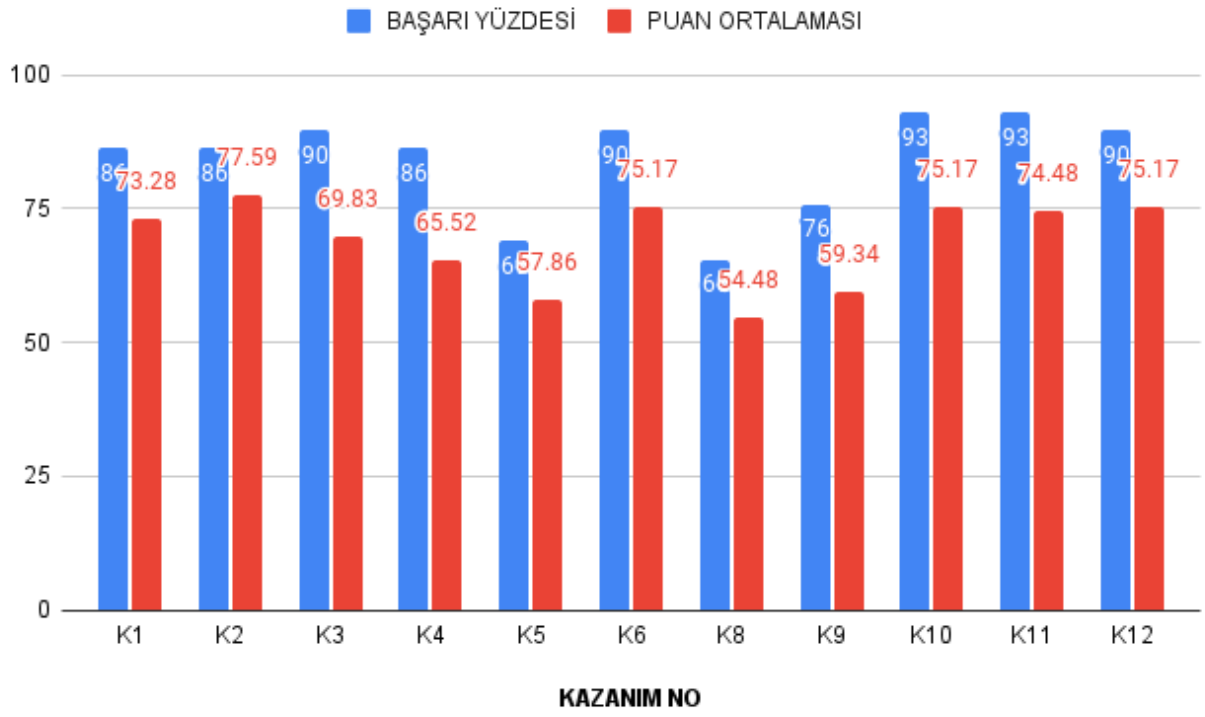
Tablo 4.22. NYP Başarı Sınavı öntest-sontest puanlarının karşılaştırılmasına ilişkin Wilcoxon sıralı işaretler testi sonuçları

Puan	Sıralar	N	Sıra $\bar{x}$	Sıra $\sum$	z	p
NYP Başarı Sınavı	Negatif sıralar	0	0	0	-4.70	.00
	Pozitif sıralar	29	15.00	435.00		
	Eşit	0				
	Toplam	29				

Tablo 4.22'ye göre NYP Başarı Sınavı için gerçekleştirilen Wilcoxon sıralı işaretler testi sonuçları, katılımcıların uygulamaya katıldıktan sonra başarı puanlarında büyük bir etki

büyüklüğü ile (r=.87) istatistiksel olarak anlamlı değişiklikler olduğunu ortaya koymuştur (z=-4.70, p=.00). Bu sonuca göre gerçekleştirilen NYP eğitimi öğrencilerin NYP bilgi ve becerilerini anlamlı bir şekilde etkilemektedir.

Öntest-sontest karşılaştırmasına ilave olarak NYP Başarı Sınavına ait kazanımlar incelenmiş olup, katılımcıların her bir kazanımdan elde ettikleri ortalama başarı yüzdeleri belirlenmiştir. Şekil 4.27'de NYP Başarı Sınavının teorik kısmına ilişkin kazanımlara ait başarı yüzdeleri ve puan ortalamaları sunulmuştur. Şekle göre başarı yüzdeleri 66 ile 93 arasında; puan ortalamaları 54.48 ile 77.59 arasında değişmektedir.



Şekil 4.27. NYP Başarı Sınavının teorik kısmına ilişkin kazanımlara ait başarı yüzdeleri ve puan ortalamaları

Şekil 4.27 incelendiğinde; NYP Başarı Sınavının teorik kısmına ilişkin kazanımlardan *"Kurucu metot kavramını örneklerle açıklar. (K5)"*, *"Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar. (K8)"* ve *"Kapsüllemenin (encapsulation) kullanım amaçlarını örnek vererek açıklar. (K9)"* kazanımlarının diğer kazanımlara kıyasla puan ortalamaları ve başarı yüzdelerinin daha düşük olduğu görülmektedir. Nitekim bu durumu teorik

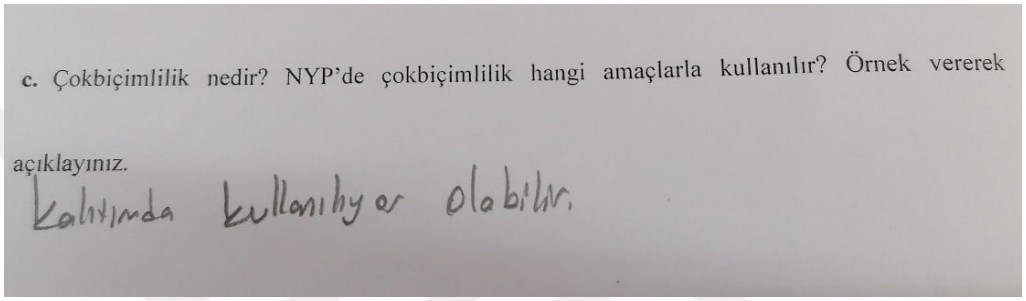
sınava ait kazanımlarda başarısız olan (hiç yanıt vermeyen ya da yanlış yanıt veren) katılımcı sayılarının verildiği Tablo 4.23 teyit etmektedir.

Tablo 4.23. NYP Başarı Sınavının teorik kısmına ilişkin kazanımlarında başarısız olan katılımcı sayıları

Kazanım No	Kazanım	Frekans (f)	Katılımcı Kodları
K8	Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar.	10	D1_K1, D1_K4, D1_K8, D1_K9, D1_K10, D1_K11, D1_K16, D1_K23, D1_K27, D1_K29
K5	Kurucu metot kavramını örneklerle açıklar.	9	D1_K2, D1_K8, D1_K9, D1_K10, D1_K12, D1_K23, D1_K27, D1_K28, D1_K29
K9	Kapsüllemenin (encapsulation) kullanım amaçlarını örnek vererek açıklar.	7	D1_K8, D1_K9, D1_K10, D1_K13, D1_K23, D1_K27, D1_K29
K1	Sınıf kavramını örneklerle açıklar.	4	D1_K7, D1_K10 D1_K18, D1_K29
K2	Nesne kavramını örneklerle açıklar.	4	D1_K7, D1_K10 D1_K18, D1_K29
K4	Davranış (behaviour) kavramını örneklerle açıklar.	4	D1_K9, D1_K10 D1_K23, D1_K29
K3	Özellik (attribute) kavramını örneklerle açıklar.	3	D1_K8, D1_K23, D1_K29
K6	Sınıf ile nesne arasındaki ilişkiyi betimler.	3	D1_K8, D1_K23, D1_K29
K12	Verilen UML Sınıf diyagramlarında yer alan sınıflara ait özellik ve davranışlarını çözümleyerek bu sınıflar arasındaki ilişkileri açıklar.	3	D1_K7, D1_K18 D1_K29
K10	Statik sınıflar ile normal sınıfları karşılaştırır.	2	D1_K18, D1_K29
K11	C# programlama dilinde kullanılan erişim belirleyicilerin kullanım amaçlarını belirtir.	2	D1_K23, D1_K29

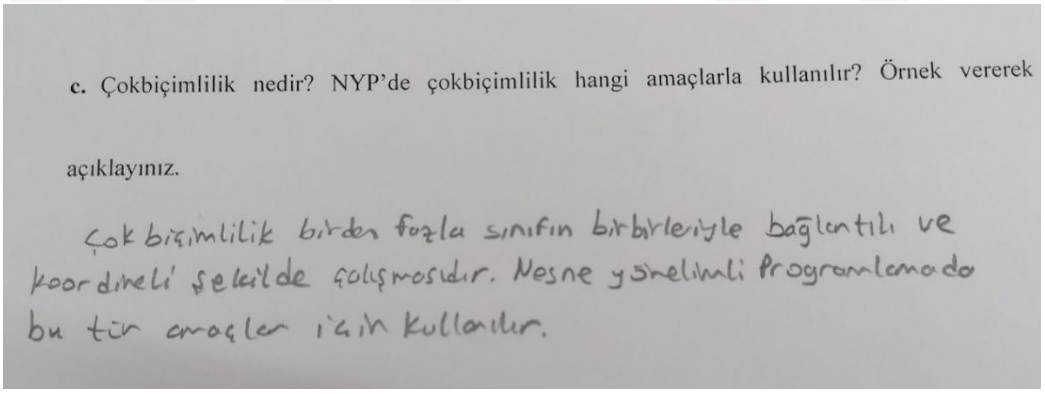
Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar. (K8) kazanımındaki başarısızlığın nedenleri için öğrencilerin sınav kağıtları incelendiği zaman bu kazanımda başarısız olan öğrencilerin %70'inin (N:7) çokbiçimlilik kavramı hakkında hiçbir

bilgi sahibi olmadıkları görülmüştür. Nitekim bu sorunla alakalı D1_K29 kodlu katılımcının sınav kağıdına bakıldığı zaman bu durum Şekil 4.28'deki gibi görülmektedir. İlave olarak katılımcıların %30'u (N:3) ise çokbiçimliliği sınıfların koordineli olarak çalışması şekilde nitelendirdikleri tespit edilmiştir. Nitekim bu sorun D1_K22 kodlu katılımcının sınav kağıdına Şekil 4.29'daki gibi görülmektedir. Bu duruma öğrencilerin çokbiçimlilik mekanizmasını tam olarak anlayamamalarının yanında çokbiçimlilikle dinamik bağlanmayı karıştırmalarının sebep olabileceği değerlendirilmektedir.


c. Çokbiçimlilik nedir? NYP'de çokbiçimlilik hangi amaçlarla kullanılır? Örnek vererek açıklayınız.

Kalıtımda kullanılıyor olabilir.

Şekil 4.28. "Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar." (K8) kazanımında başarısız olan sınav kâğıdı örneği (D1_K29)

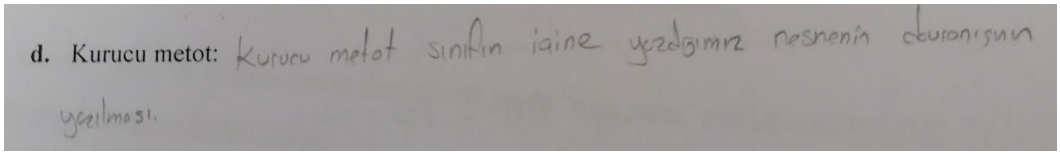

c. Çokbiçimlilik nedir? NYP'de çokbiçimlilik hangi amaçlarla kullanılır? Örnek vererek açıklayınız.

Çokbiçimlilik birden fazla sınıfın birbirleriyle bağlantılı ve koordineli şekilde çalışmasıdır. Nesne yönelimli Programlamada bu tür amaçlar için kullanılır.

Şekil 4.29. "Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar." (K8) kazanımında başarısız olan sınav kâğıdı örneği (D1_K22)

Kurucu metot kavramını örneklerle açıklar. (K5) kazanımındaki başarısızlığın nedenleri için öğrencilerin sınav kâğıtları incelendiği zaman öğrencilerin büyük bir kısmının kurucu metotla sınıf metotları arasındaki farkları anlayamadıkları ve bu iki metot türünü karıştırdıkları tespit edilmiştir. Nitekim bu sorunla alakalı D1_K10 kodlu katılımcının sınav kâğıdına bakıldığı zaman bu karmaşıklık Şekil 4.28'deki gibi görülmektedir. Bunun yanında Unity 3D'de kurucu metot kullanımı yerine Monobehaviour metotlarından Start() metodunun kullanılmasının

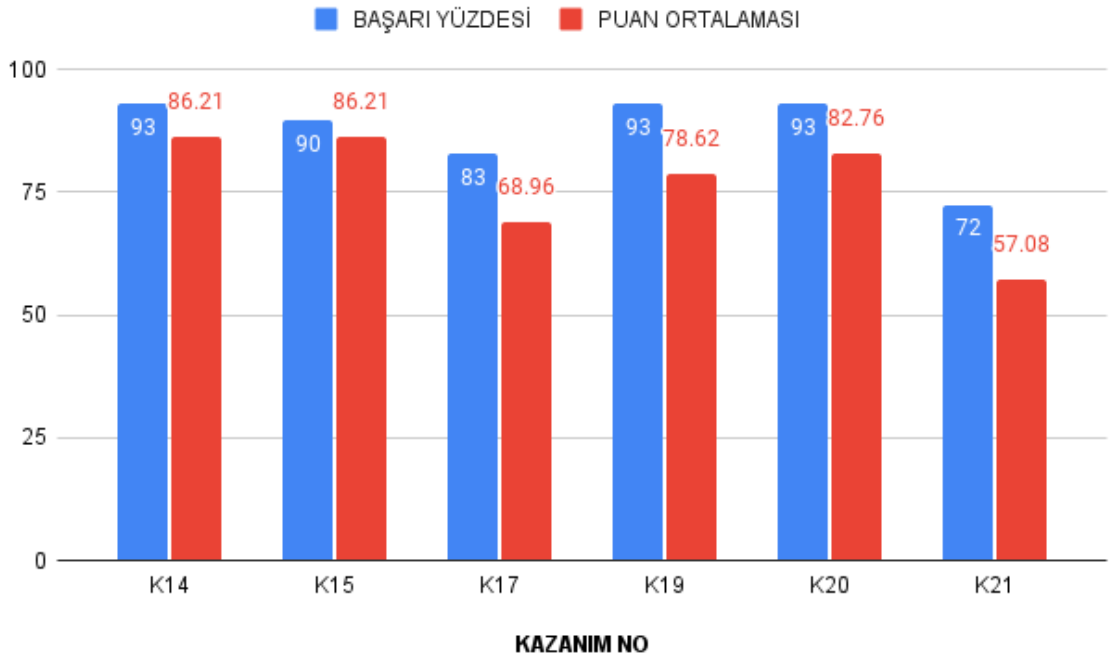
tavsiye edilmesi nedeniyle birinci döngüde kurucu metotlarla ilgili bir herhangi bir uygulamanın yapılmamasının da bu duruma neden olabileceği değerlendirilmektedir.



Şekil 4.30. "Kurucu metot kavramını örneklerle açıklar." (K5) kazanımında başarısız olan sınav kâğıdı örneği (D1_K10)

*"Kapsüllemenin (encapsulation) kullanım amaçlarını örnek vererek açıklar. (K9)"*kazanımında başarısız olan öğrencilerin sınav kâğıtları incelendiğinde, D1_K9, D1_K13, D1_K23 ve D1_K27 kodlu katılımcıların kapsüllemeye örnek veremedikleri görülmüştür. İlave olarak D1_K8, D1_K10 ve D1_K29 kodlu katılımcıların soruya herhangi bir yanıt yazmadıkları görülmüştür. D1_K8, D1_K10 ve D1_K29 kodlu katılımcılarla yapılan odak grup görüşmesinde bu durumun nedeni sorulduğunda, D1_K8 kodlu katılımcı kapsüllemeyle ilgili sadece "veri saklama işlemi" olduğunu hatırladığını; ancak bunun ne şekilde ve hangi amaçlarla yapıldığını bilmediğini belirtmiştir. D1_K10 ve D1_K29 kodlu katılımcı ise kapsülleme kavramıyla ilgili herhangi bir fikirlerinin olmadığını beyan etmişlerdir.

Teorik kazanımlara ilave olarak NYP Başarı Sınavının uygulama kısmına ait kazanımlara ilişkin başarı yüzdeleri ve puan ortalamaları da belirlenmiştir (Şekil 4.31). Şekil 4.31'e göre başarı yüzdeleri 72 ile 93 arasında; puan ortalamaları ise 57.08 ile 86.21 arasında değişmektedir. En düşük başarı yüzdesi "*Bir C# sınıfına ait Monobehavior metotlarını, verilen koşula göre oluşturur. (K21)"*kazanımında gerçekleşmiştir. NYP Başarı Sınavının uygulama kısmına ilişkin kazanımlardan başarısız olan katılımcı sayıları Tablo 4.24'te sunulmuştur. Sekiz katılımcının *"Bir C# sınıfına ait Monobehavior metotlarını, verilen koşula göre oluşturur. (K21)"* kazanımında; beş katılımcının "*UML sınıf diyagramında belirtilen çokbiçimli metotları oluşturur ve kullanır. (K17)*" kazanımında başarısız olması, katılımcıların en fazla Monobehavior metotları ve çokbiçimlilik konularında zorlandıklarını göstermektedir. Nitekim teorik sınavda da çokbiçimlilik kazanımında başarısız olan katılımcı sayısı, diğer kazanımlara göre daha fazla olmuştur.



Şekil 4.31. NYP Başarı Sınavının uygulama kısmına ilişkin kazanımlara ait başarı yüzdeleri ve puan ortalamaları

Tablo 4.24. NYP Başarı Sınavının uygulama kısmına ilişkin kazanımlarında başarısız olan katılımcı sayıları

Kazanım No	Kazanım	Frekans (f)
K21	Bir C# sınıfına ait Monobehavior metotlarını, verilen koşula göre oluşturur.	8
K17	UML sınıf diyagramında belirtilen çokbiçimli metotları oluşturur ve kullanır.	5
K15	Önceden oluşturulmuş bir C# sınıfını, belirtilen oyun nesnesine bileşen (component) olarak ekler.	3
K14	Bir sınıfa ait özelliklerin erişim belirleyicilerini verilen koşula göre değiştirir.	2
K19	UML sınıf diyagramı verilen sınıfı C# programlama dilinde oluşturur.	2
K20	Önceden oluşturulmuş bir C# sınıfına ait özelliklerin değer atamasını verilen koşula göre yapar.	2

Katılımcıların sontest teorik sınav notları ile uygulama sınav notları arasında bir ilişki olup olmadığının belirlenebilmesi amacıyla Pearson çarpım moment korelasyon analizi gerçekleştirilmiştir (Tablo 4.25). Tabloya göre sontest teorik sınav notları ile uygulama sınav

notları arasında p<.01 düzeyinde .805'lık pozitif yönde çok yüksek düzeyde bir ilişki bulunmuştur.

Tablo 4.25. Sontest teorik sınav ile uygulama sınavı Pearson çarpım moment korelasyon analizi sonuçları

	Sontest Teorik Sınav	**Sontest Uygulama Sınavı**
Sontest Teorik Sınav	—	.805**
Sontest Uygulama Sınavı	.805**	—

** p< .01.

Katılımcıların programlama dersine karşı motivasyonlarının, programlama kaygılarının ve programlamaya yönelik öz yeterlilik algılarının sontest başarı puanları ile ilişkisini incelemek amacıyla korelasyon analizi gerçekleştirilmiştir. Ölçeklere ait betimsel istatistikler Tablo 4.26'da, korelasyon analizi sonuçları Tablo 4.27'de sunulmuştur. Pearson Çarpım Moment Korelasyon analizi sonucunda Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeğinin altı alt faktörü ile katılımcıların sontest başarı puanları arasında anlamlı bir ilişki bulunmamıştır. Benzer şekilde katılımcıların programlamaya yönelik öz yeterlilik algıları ile sontest başarı puanları arasında anlamlı bir ilişki bulunmamıştır (r=.279; p>.05). Her ne kadar Programlama Kaygı Ölçeğinin Akran Kaygısı alt faktörü ile katılımcıların sontest başarı puanları arasında anlamlı bir ilişki bulunamasa da (r=-.37; p>.05) ölçeğin Programlama Beceri Kaygısı alt faktörü ile katılımcıların sontest başarı puanları arasında anlamlı bir ilişki bulunmuştur (r=-.48; p<.05). İlave olarak katılımcıların programlamaya yönelik öz yeterlilik algıları ile Programlama Kaygı Ölçeğinin Akran Kaygısı ve Programlama Beceri Kaygısı alt faktörleri arasında istatistiksel açıdan p<.01 düzeyinde negatif yönde anlamlı bir ilişki saptanmıştır (sırasıyla r=-.588 ve r=-.676).

Tablo 4.26. Ölçeklere ilişkin betimsel istatistik sonuçları

	Sontest		
	N	**x̄**	**ss**
Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği			
Tutum ve Beklenti	27	4.44	.73
Zorlayıcı Amaçlar	27	3.27	1.33
Belirgin Hedefler	27	4.02	1.09
Ödül ve Takdir	27	4.20	1.23
Ceza	27	3.06	.88
Sosyal Baskı ve Rekabet	27	2.60	1.34
Toplam Puan	27	3.63	.95
Programlama Kaygı Ölçeği			
Akran Kaygısı	27	2.37	1.06
Programlama Beceri Kaygısı	27	2.87	1.06
Toplam Puan	27	2.65	1
Programlama Öz Yeterlilik Ölçeği			
Toplam Puan	27	3.57	.83

Tablo 4.27. Pearson çarpım moment korelasyon analizi sonuçları

	Tutum ve Beklenti	Zorlayıcı Amaçlar	Belirgin Hedefler	Ödül ve Takdir	Ceza	Sosyal Baskı ve Rekabet	Akran Kaygısı	Programlama Beceri Kaygısı	Programlama Öz yeterlilik Algısı	NYP Başarı Sınavı (Sontest)
Tutum ve Beklenti	—									
Zorlayıcı Amaçlar	.535**	—								
Belirgin Hedefler	.550**	.743**	—							
Ödül ve Takdir	.570**	0.30	.397*	—						
Ceza	.559**	0.34	.311	.614**	—					
Sosyal Baskı ve Rekabet	.507**	.641**	.370	.320	.665**	—				
Akran Kaygısı	-0.07	-0.09	-.383*	-.227	.037	.010	—			
Programlama Beceri Kaygısı	-0.24	-0.37	-.593**	-.318	-.006	-.018	.760**	—		
Programlama Öz yeterlilik Algısı	.503**	0.34	.549**	.477*	.217	.154	-.588**	-.676**	—	
NYP Başarı Sınavı (Sontest)	0.06	0.36	.310	-.009	-.284	-.134	-.370	-.480*	.279	—

* p< .05, ** p< .01.

Ölçeklerden elde edilen verilerin analizlerine ilave olarak, katılımcıların daha önce aldıkları programlama derslerindeki başarı durumlarının NYP Başarı Sınavı sontest puanlarına etkisi de istatistiksel olarak analiz edilmiştir. Bu analizlerde örneklem büyüklüğü de dikkate alınarak Algoritma ve Programlamaya Giriş dersi ile Görsel Programlama dersinin akademik başarı puanları, NYP Başarı Sınavı sontest puanlarına ile ayrı ayrı regresyon analizi işlemlerine tabi tutulmuştur.

Algoritma ve Programlamaya Giriş dersinin akademik başarı puanları ile katılımcıların sontest başarı puanlarının regresyon analizi sonuçlarına göre (Tablo 4.28) bu iki değişken arasında istatistiksel olarak anlamlı bir ilişki vardır (R^2=.285, F(1,27)=10.751, p<.05). Katılımcıların daha önce aldıkları Algoritma ve Programlamaya Giriş dersinin akademik başarı puanları, son test başarı puanlarını anlamlı bir şekilde (t=3.279, p=.003) etkilemektedir.

Tablo 4.28. Algoritma ve Programlamaya Giriş dersi ile NYP Başarı Sınavı sontest başarı puanları regresyon analiz sonuçları

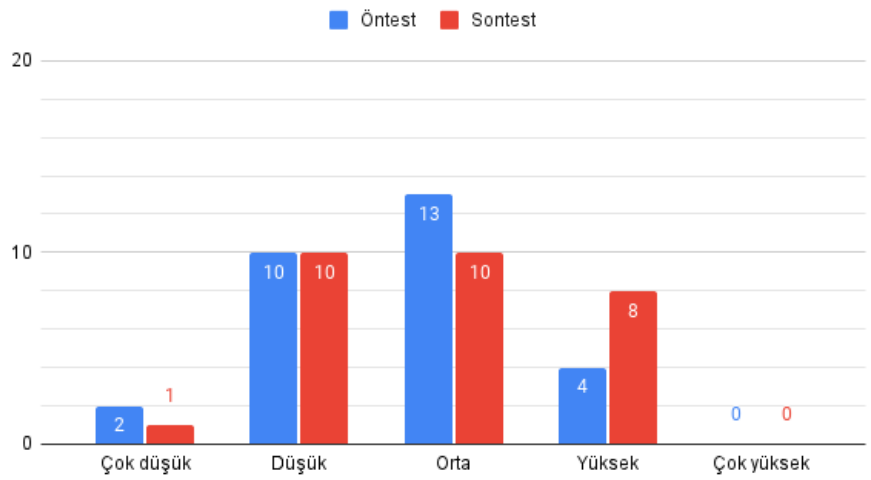
Değişken	B	Standart Hata	β	t	p
Algoritma ve Programlamaya Giriş	.65	.198	.534	3.279	.003

Görsel Programlama dersinin akademik başarı puanları ile katılımcıların sontest başarı puanlarının regresyon analizi sonuçlarına göre (Tablo 4.29) bu iki değişken arasında istatistiksel olarak anlamlı bir ilişki vardır (R^2=.605, F(1,27)=41.330, p<.05). Katılımcıların daha önce aldıkları Görsel Programlama dersinin akademik başarı puanları, son test başarı puanlarını anlamlı bir şekilde (t=6.429, p=.000) etkilemektedir.

Tablo 4.29. Görsel Programlama dersi ile NYP Başarı Sınavı sontest başarı puanları regresyon analiz sonuçları

Değişken	B	Standart Hata	β	t	p
Görsel Programlama	.861	.134	.778	6.429	.000

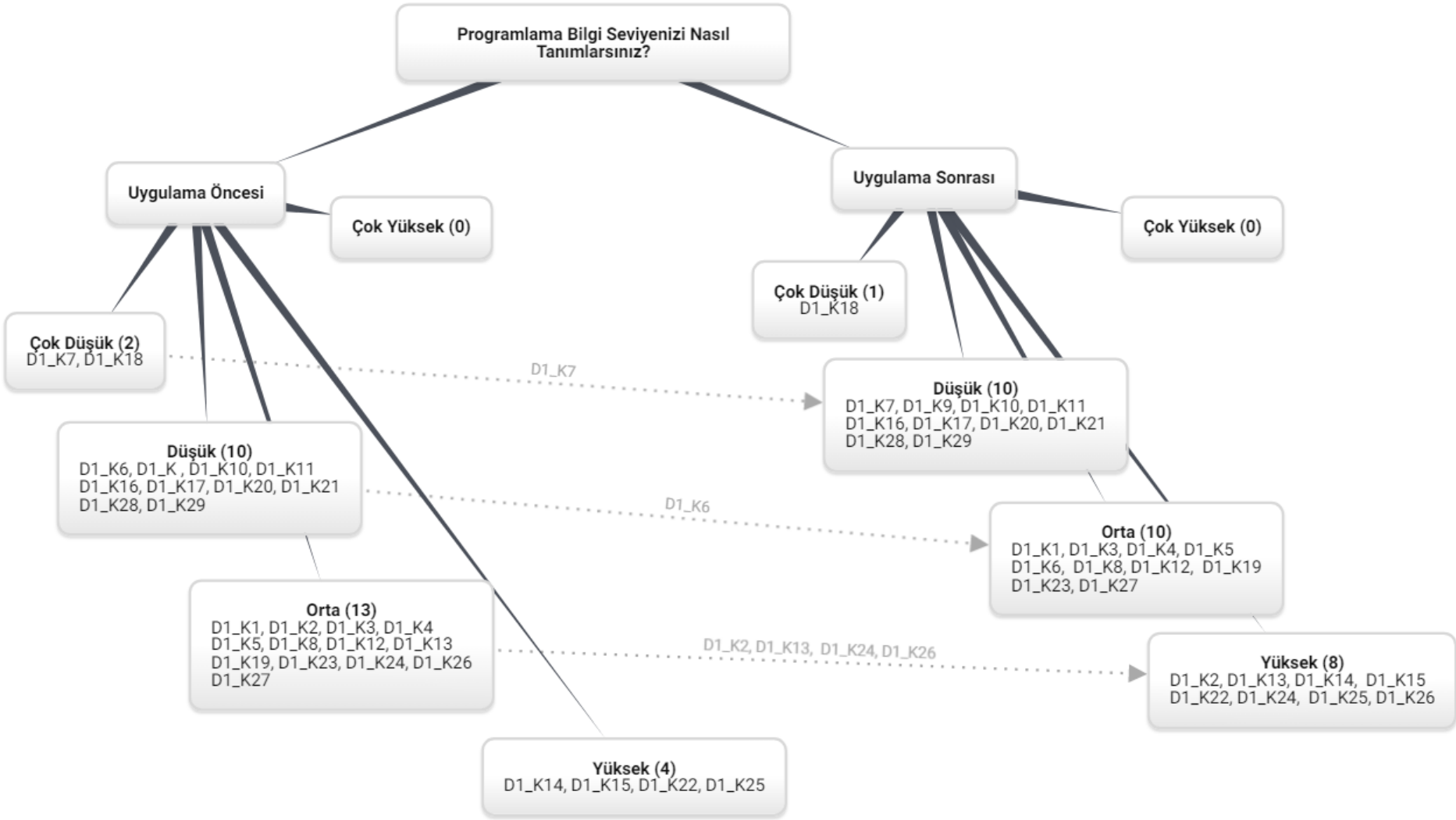
NYP Başarı Sınavının ayrıntılı analizlerine ilave olarak katılımcıların uygulama öncesinde ve sonrasında algıladıkları programlama bilgi seviyesindeki değişim, katılımcı bilgi formu kullanılarak incelenmiştir. Katılımcıların algıladıkları programlama bilgi seviyeleri karşılaştırma grafiği Şekil 4.32'de verilmiştir.



Şekil 4.32. Uygulama öncesinde algılanan programlama bilgi seviyesi dağılım grafiği

Şekil 4.32'ye göre katılımcılar uygulama öncesinde algıladıkları programlama bilgi seviyelerini ağırlıklı olarak (N:27, %93) düşük, orta ve yüksek olarak nitelendirmiştir. İlave olarak kendilerini programlama bilgisi seviyesi açısından çok yüksek seviye olarak nitelendiren hiç katılımcı olmamıştır. Benzer şekilde uygulama sonrasında kendilerini programlama bilgisi seviyesi açısından düşük, orta ve yüksek seviye olarak nitelendiren katılımcıların yüzdesi biraz artarak %97 (N:28) seviyesine ulaşmıştır. Uygulama öncesi ve sonrası kıyaslamasında en önemli değişim yaşanan iki seviye çok düşük ve yüksek seviyeler olmuştur. Uygulama öncesinde katılımcıların %7'si (N:2) kendilerini çok düşük seviye olarak nitelendirirken bu oran uygulama sonrasında %3'e (N:1) gerilemiştir. Uygulama öncesinde kendisini yüksek seviye olarak nitelendiren katılımcıların oranı %14 (N:4) iken bu oran uygulama sonrasında %28'e (N:8) çıkmıştır.

Katılımcıların algıladıkları programlama bilgi seviyesindeki değişimler, katılımcı bazında da incelenmiştir (Şekil 4.33). Benzer şekilde uygulama öncesine göre kendilerini düşük ve orta seviye olarak nitelendiren katılımcıların sayısı azalmış ve kendilerini orta ve yüksek seviye olarak nitelendiren katılımcıların sayısı artmıştır. Algılanan programlama seviyesinde değişim yaşayan altı katılımcı değerlendirildiğinde, değişimin en çok orta seviyeden iyi seviyeye doğru gerçekleştiği sonucuna ulaşılmaktadır (N:4). Bununla birlikte katılımcılardaki algılanan programlama bilgi seviyesindeki değişimlerin bir üst basamağa doğru gerçekleştiği (örn. düşükten ortaya); hiçbir katılımcının değişiminin iki üst basamağa (örn. düşükten yükseğe) gerçekleşmediği görülmektedir.



Şekil 4.33. Katılımcıların uygulama öncesi ve uygulama sonrasında algıladıkları programlama bilgi seviyesindeki değişim

Programlama bilgi seviyesindeki değişimin özeti Tablo 4.30'da sunulmuştur. Tabloya göre çok düşükten düşüğe, düşükten ortaya ve ortadan yükseğe olmak üzere üç kategoride değişim yaşanmış olup en çok değişim ortadan yükseğe yönünde olmuştur. İlave olarak, katılımcıların %79'unun algıladıkları programlama bilgi seviyesinde değişim yaşanmazken %21'ininde değişim yaşanmıştır (Tablo 4.31).

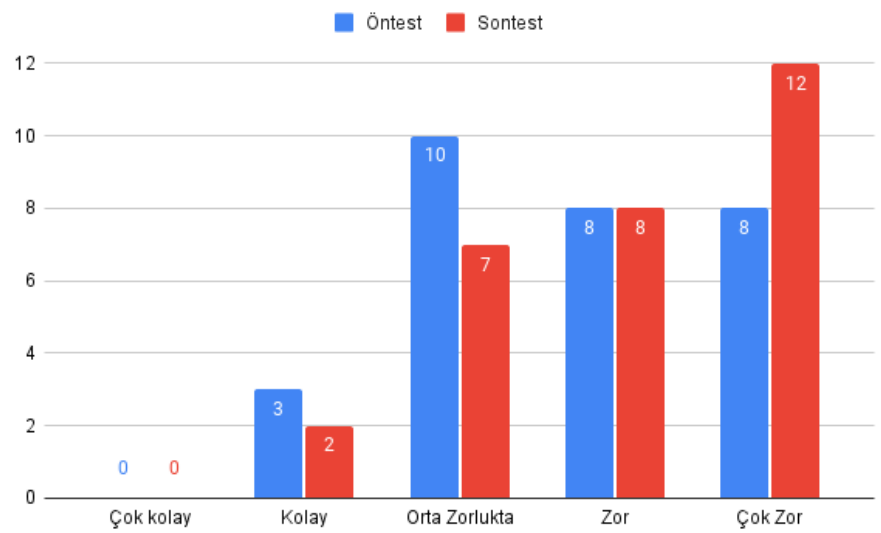
Tablo 4.30. Katılımcıların algıladıkları programlama bilgi seviyesindeki değişimin yönü

Değişimin Yönü		**Frekans (f)**
Öntest	**Sontest**	
Çok Düşük	Düşük	1
Düşük	Orta	1
Orta	Yüksek	4

Tablo 4.31. Katılımcıların algıladıkları programlama bilgi seviyesindeki değişim durumu

Değişim Durumu	**Frekans (f)**	**Yüzde (%)**
Algılanan Programlama Bilgi Seviyesi Artan	6	21
Algılanan Programlama Bilgi Seviyesi Azalan	-	-
Algılanan Programlama Bilgi Seviyesi Değişmeyen	23	79

Algılanan programlama bilgi seviyesindeki değişimlere ek olarak katılımcıların algıladıkları programlama zorluk seviyesindeki değişimler de incelenmiştir. Uygulama öncesi ve sonrası algılanan zorluk seviyesi değişim grafiği Şekil 4.34'te sunulmuştur. Şekle göre algıladıkları zorluk seviyesi "orta zorlukta" ve "zor" kategorisinde olan katılımcıların sayısı azalırken "çok zor" kategorisinde algılanan zorluk seviyesi artmıştır. Dolayısıyla katılımcıların algıladıkları programlama zorluk seviyesinin arttığı söylenebilir. Nitekim algılanan zorluk seviyesinin 5'li likert tipinde değerlendirildiği hesaba katıldığında; uygulama öncesinde 3.72 olarak ölçülen bu değerin uygulama sonrasında 4.03 olarak ölçülmesi katılımcıların algıladıkları programlama zorluk seviyesinin arttığını teyit etmektedir. Benzer şekilde, katılımcıların %34'ünün algıladıkları programlama zorluk seviyesi artarken %10'unun azalması, bu durumu doğrular niteliktedir (Tablo 4.31). Son olarak, katılımcıların %55'inin algıladıkları programlama zorluk seviyesinde herhangi bir değişimin yaşanmadığı ortaya çıkmıştır. (Tablo 4.32).



Şekil 4.34. Katılımcıların algıladıkları zorluk seviyesindeki değişim grafiği

Tablo 4.32. Katılımcıların algıladıkları programlama zorluk seviyesindeki değişim durumu

Değişim Durumu	Frekans (f)	Yüzde (%)
Algılanan Programlama Zorluk Seviyesi Artan	10	34
Algılanan Programlama Zorluk Seviyesi Azalan	3	11
Algılanan Programlama Zorluk Seviyesi Değişmeyen	16	55

NYP Başarı Sınavı, ölçek verileri ve katılımcıların algıladıkları programlama bilgi seviyeleri incelendikten sonra uygulanan eylem planının geçmiş programlama derslerindeki başarı durumu açısından düşük ve yüksek düzeyde olan öğrenciler üzerinde etkisini incelemek maksadıyla bazı analizler yapılmıştır. İlk olarak, düşük ve yüksek düzeyde olan katılımcıların NYP Başarı Sınavı sontest puanları karşılaştırılmıştır (Tablo 4.33). Gerçekleştirilen Mann Whitney U testi sonuçlarına göre geçmiş programlama derslerindeki başarı durumu açısından düşük ve yüksek düzeyde olan öğrencilerin sontest başarı puanları arasında anlamlı bir ilişki bulunmaktadır U=.000, p=.001.

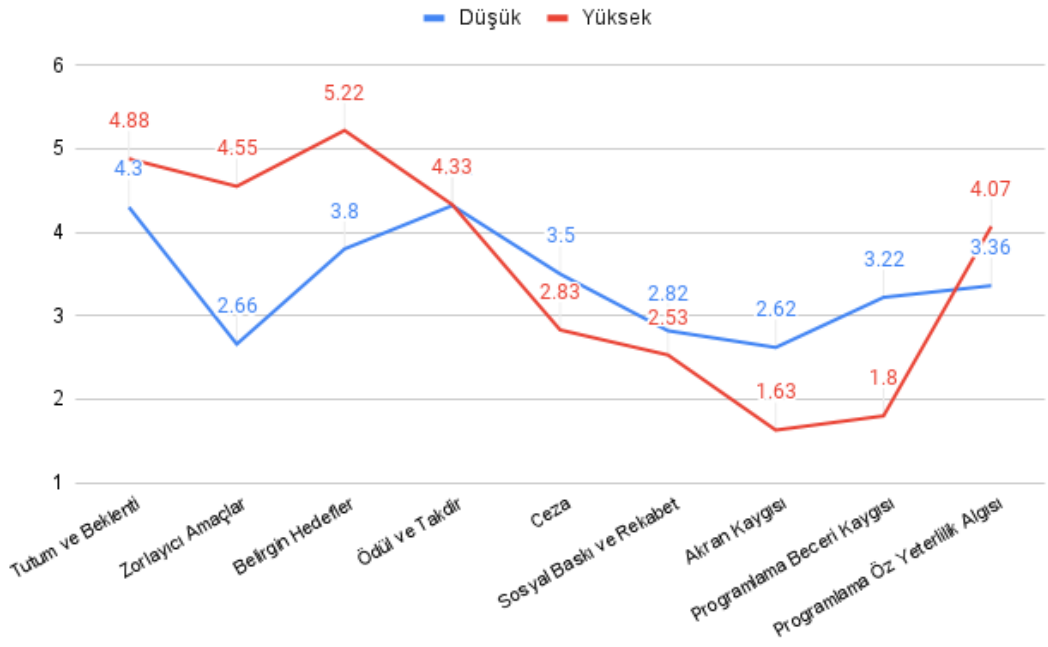
Tablo 4.33. Mann Whitney U testi sonuçları

Grup	N	Sıra $\bar{x}$	Sıra $\sum$	U	z	p
Düşük	7	4.00	28.00	.000	-3.004	.001
Yüksek	6	10.50	63.00			

NYP Başarı Sınavına ilave olarak geçmiş programlama derslerindeki başarı durumu açısından düşük ve yüksek düzeyde olan öğrencilerin programlama dersine karşı motivasyonları, programlama kaygıları ve programlamaya yönelik öz yeterlilik algıları karşılaştırılmıştır. Ölçeklere ait betimsel istatistikler Tablo 4.34'te verilmiştir. Buna ilave olarak geçmiş programlama derslerindeki başarı durumu açısından düşük ve yüksek düzeyde olan öğrencilerin Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeğinin altı alt faktörü, Programlama Kaygı Ölçeğinin iki alt faktörü ile Programlama Öz Yeterlilik Ölçeği puan ortalamalarının karşılaştırmaları Şekil 4.35'te verilmiştir.

Tablo 4.34. Düşük ve yüksek düzeyde başarılı olan öğrencilerin ölçek verilerine ilişkin betimsel istatistik sonuçları

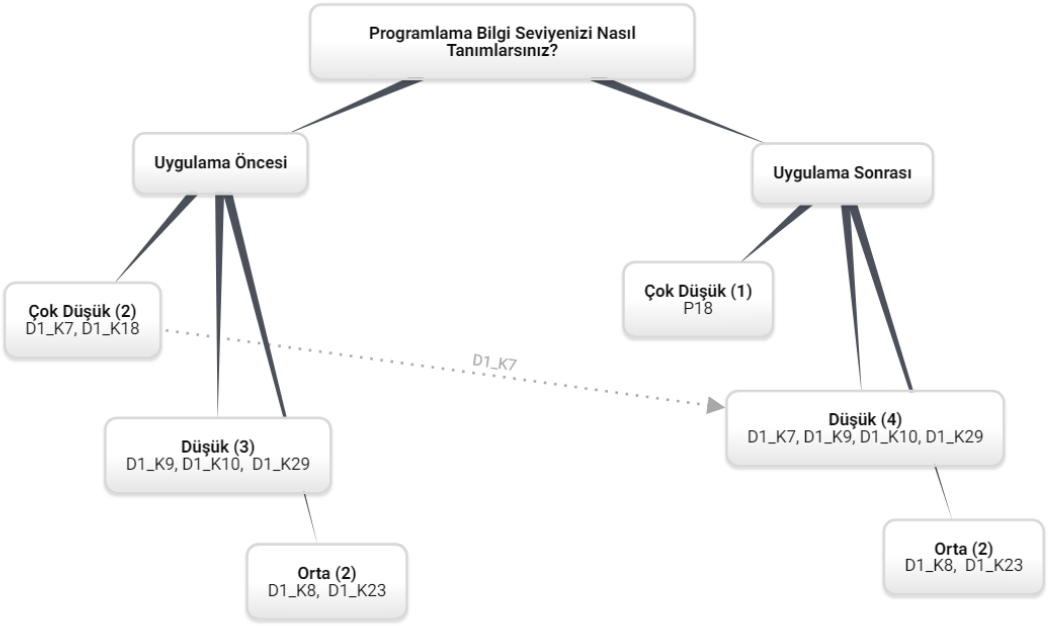
Ölçek	Grup	N	Min.	Maks.	x̄	ss
Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği	Düşük	7	2.37	5.21	3.7	1.3
	Yüksek	6	2.95	5.4	4.836	.95
Programlama Kaygı Ölçeği	Düşük	7	1.71	5	2.96	1.26
	Yüksek	6	1	2.14	1.72	.49
Programlama Öz Yeterlilik Ölçeği	Düşük	7	1.6	4.7	3.36	1.22
	Yüksek	6	3.6	5	4.07	.49



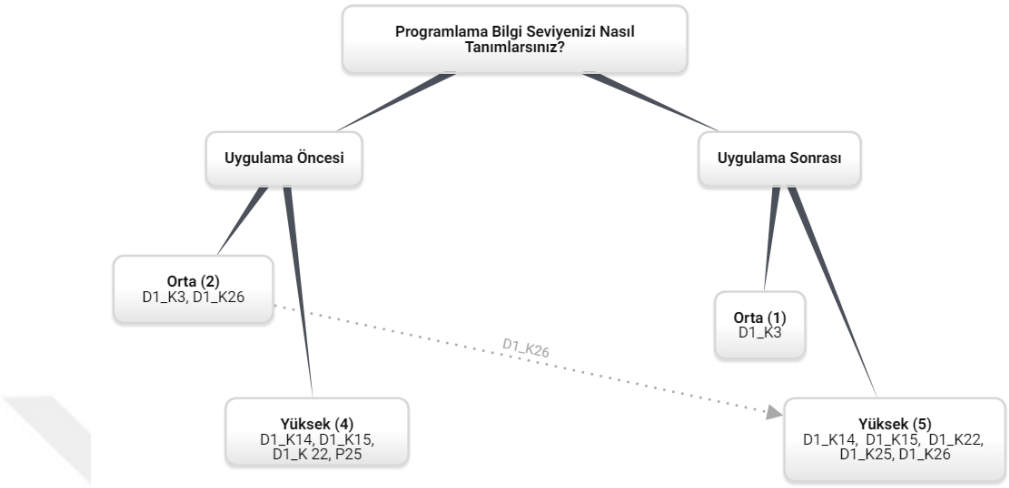
Şekil 4.35. Düşük ve yüksek düzeyde başarılı olan öğrencilerin ölçek alt faktör ortalamalarının karşılaştırma grafiği

Şekil 4.35'e göre, yüksek başarılı öğrencilerin Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeğinin "*tutum ve beklenti*", "*zorlayıcı amaçlar*", ve "*belirgin hedefler*" alt faktörlerinde düşük başarılı öğrencilere göre daha yüksek puan ortalamalarına sahip oldukları görülmektedir. Ancak düşük başarılı öğrenciler "ceza" ile "sosyal baskı ve rekabet" alt faktörlerinde daha yüksek puan ortalamalarına sahiptir. "*Ödül ve takdir*" alt faktör ortalama puanları her iki grubun da eşittir. Programlama Kaygı Ölçeği değerlendirildiğinde, düşük başarılı öğrencilerin hem akran hem de programlama beceri kaygıları daha yüksek çıkmıştır. Ancak yüksek başarılı öğrencilerin Programlama Öz Yeterlilik Ölçeği puan ortalamaları düşük başarılı öğrencilerden daha yüksektir.

Ölçek verilerinin analizine ilave olarak, önceki programlama derslerindeki başarı durumu açısından düşük ve yüksek başarılı öğrencilerin uygulama öncesi ve sonrasında algıladıkları programlama bilgi seviyesindeki değişim de incelenmiştir. Bu kapsamda düşük başarılı öğrencilerin uygulama öncesinde ve sonrasında algıladıkları programlama bilgi seviyesi çok düşük, düşük ve orta seviyede gerçekleşmiştir (Şekil 4.36). Bundan farklı olarak yüksek başarılı öğrencilerin uygulama öncesi ve sonrasında algıladıkları programlama bilgi seviyeleri "orta" ve "yüksek" seviyede gerçekleşmiştir (Şekil 4.37).



Şekil 4.36. Düşük başarılı öğrencilerin uygulama öncesi ve uygulama sonrasında algıladıkları programlama bilgi seviyesindeki değişim



Şekil 4.37. Yüksek başarılı öğrencilerin uygulama öncesi ve uygulama sonrasında algıladıkları programlama bilgi seviyesindeki değişim

4.1.4.2. Araştırma Sorusu 9: Geliştirilen NYP dersine yönelik katılımcıların görüşleri nelerdir?

Katılımcıların oyun geliştirmeye dayalı programlama öğretimi süreciyle ilgili algıları öğrenci günlükleri ve odak grup görüşmesinden elde edilen veriler doğrultusunda çözümlenerek Şekil 4.38'de sunulmuştur. Katılımcıların oyun geliştirmeye dayalı olarak verilen NYP öğretim sürecine yönelik olumlu algıları *keyifli, faydalı, motive edici ve ilgi çekici* şeklinde listelenebilir.

Katılımcılar öğretim süreciyle ilgili olumlu algılarını şu cümlelerle ifade etmişlerdir:

"Normal programlama derslerine göre daha zevkli bir öğretim süreci yaşadığımızı düşünüyorum. Nesne Yönelimli Programlamayı özellikle oyun programlayarak öğrenmek ve bir oyun geliştirmek çok daha eğlenceli oldu." (D1_K12).

"Unity motoru, görsel programlamaya karşı[görsel programlamaya göre] daha eğlenceli ve öğretici." (D1_K2).

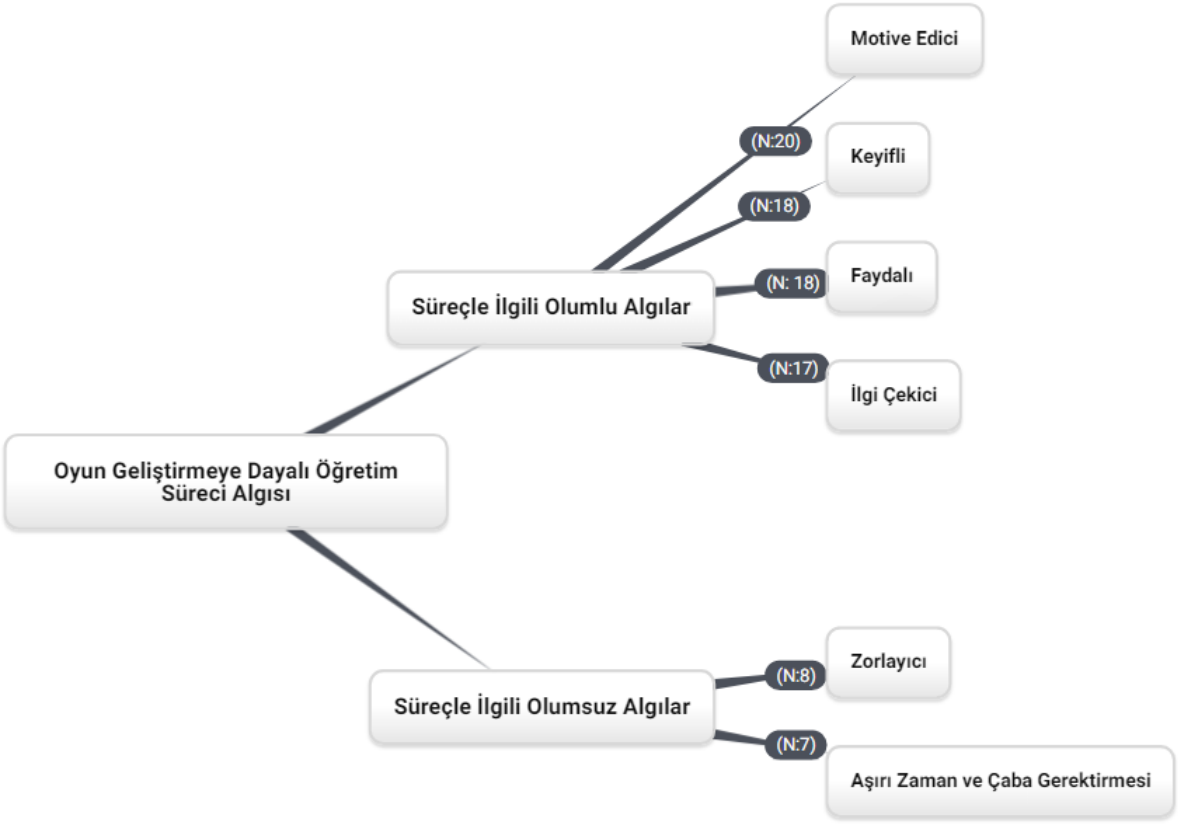
"Oyun programlaması yaparak programlama yapmayı öğrenmek açık bir deyişle daha eğlenceli. Farkındalık yaratmak için gayet iyi oldu farklılık değişik eklentiler merak ettirici ve kamçılayıcı şekilde olduğu için bana bu şekilde faydaları oldu." (D1_K5).

"Bu şekilde devam etmesi benim hoşuma gidiyor. Yoktan bir şeyler yaratmak ve görsellik kullanmak hoşuma gidiyor." (D1_K15).

"Derste kullandığımız yöntem akılda daha kalıcı, mantıklı olduğunu ve daha da öğretici olduğunu düşünüyorum. Form ve konsollar sıkıcı bence. Motivasyon sağlıyor ve ilgi çekiyor." (D1_K27).

Araştırmacı da katılımcıların olumlu algılarını gözlemlemiş ve deneyimlerini araştırmacı günlüğüne şu cümlelerle aktarmıştır:

Bugün ilk defa Unity 3D editörünü kullanmaya başladık. Kullanmaya başlamadan önce oyun nedir, çeşitleri nelerdir, oyun motoru kavramı gibi kavramlardan bahsettim. Scene, asset, component, texture, material vb. temel kavramlardan bahsettim. Unity 3D editör pencerelerinden (scene, game, hierarchy view vb.) bahsettim. Öğrencilerimin ilgilerini çektiğini düşünüyorum. Bana Unity skybox'ını nasıl değiştirebiliriz, nesneleri nasıl çarpıştırabiliriz, nasıl daha hızlı döndürebiliriz gibi sorular soruyorlar. Bu beni gayet memnun ediyor. Derse karşı olan ilgilerinin ve motivasyonlarının arttığını gözlemliyorum. Hemen hemen tüm soruları yanıtlamaya çalışıyorum. Küçük notlar tutmaları için kendilerini teşvik ediyorum. (Araştırmacı günlüğü, 02.07.2020).



Şekil 4.38. Katılımcıların oyun geliştirmeye dayalı öğretim süreciyle ilgili algıları

Katılımcıların öğretim sürecine yönelik olumsuz algıları ise sürecin *zorlayıcı olması* ve *aşırı zaman ve çaba gerektirmesiyle* ilgilidir (Şekil 4.38). Katılımcıların NYP öğretimini neden zorlayıcı bulduklarıyla ilgili detaylı bilgi edinebilmek maksadıyla NYP öğretim sürecini

zorlayıcı bulan öğrencilerin sekizinin dâhil olduğu bir odak grup görüşmesi gerçekleştirilmiştir. Öğrencilerin NYP öğretimini zorlayıcı bulmalarının temel nedeni NYP paradigmasının program çözümüne sınıf ve nesneleri koyarak öğrencilerin alışkın oldukları prosedürel programlama dinamikleriyle benzememesiyle ilgilidir. Öğrenciler önceden aldıkları Görsel Programlama dersinde form uygulamaları geliştirmişlerdir. Formun gerçekleştirmesini istedikleri yetenekleri button, radiobutton, checkedbox, listbox gibi form elemanlarını kullanarak gerçekleştirmişlerdir. Öğrenciler özellikle formda yer alan nesnelerin olaylarına (events) kod yazarak olay tabanlı programlama uygulamaları yapmaya alışkındırlar. Ancak NYP dersiyle birlikte öğrenciler problem çözümleri için oyun nesnelerine davranışlar kazandırmayı içeren sınıflar tanımlamışlar; oyun mekanizmalarını oyun nesnelerin davranışlarıyla yönlendirmişlerdir. Öğrencilerin hangi sınıfların tanımlanacağı; bu sınıfların hangi özellikler ve davranışları içereceğini belirlemek öğrenciler için zorlayıcı olmuştur. Çünkü bu süreç bir bakıma birbirleriyle metotları üzerinden mesajlaşacak nesnelerin ilişkilerini tasarlamak ve bunu C# koduna dökmeyi içermektedir. D1_K9 kodlu katılımcının yaşadığı bu zorlukla ilgili düşünceleri şu şekildedir:

> Ben C# yazılım dilini görsel programlama dersiyle öğrenmeye başlamıştım ve o şekilde alışmıştım. Fakat bir anda çok farklı içeriklerle karşılaşınca iyice kafam karıştı. Görsel programlamadan NYP'ye geçmek insanda alışmış olduğu gerçekleri değiştirmek zorunda bırakıyor; insanın zihni de doğal olarak buna kolayca alışamıyor. Görsel Programlama dersinde somut nesneler üzerinden (textbox button label gibi) çalıştığımız için her şeyi görerek yapıyordum; mesela sadece button'ın click olayına kod yazıyordum. Nesne yönelimli programlamaya geçtiğimizde ise önümde programlanmayı bekleyen boş bir sayfa gördüm; yapmam gereken şey oyun nesneleri için sınıflar ve metotlar yaratmaktı ancak nereden başlayacağımı başlarda bilemedim. Zamanla sıfırdan bir dünya oluşturur gibi yaratmaya başladık ve derse olan ilgim arttı. Bir kere oturtunca programlamayı ne kadar kolaylaştırdığını gördüm.

Katılımcıların öğretim sürecini zorlayıcı bulmalarının bir diğer nedeni, NYP'de yer alan kavramların ne işe yaradığı ve nasıl kullanılacağını öğrenmenin; bunun yanında sınıflar arasındaki ilişkiyi kavramanın fazla çaba gerektirmesiyle ilgilidir. Nitekim D1_K10 kodlu katılımcının bu zorlukla ilgili düşünceleri şu şekildedir:

> "Programlama dersinde oyun programlamak eğlenceli. Bu süreçte kendi oyun projem üstünde sıkılmadan çalıştım. Ancak sınıf, nesne, özellik, davranış gibi kavramların ne olduğunu ve programlamada nasıl kullanıldığını anlamak; bunun yanında projemde yer alan sınıflar arasındaki ilişkiyi kavramak için üzerinde çok çalışmam gerekti. Konuları anlamaya çalışmak zaman aldığı için projem yavaş ilerledi."

Katılımcılardan bazılarının ise oyun projelerini geliştirirken problemleri alt problemlere bölemedikleri ve bunu başaramamanın performanslarını olumsuz yönde etkilediği ortaya çıkmıştır. Nitekim D1_K7 kodlu katılımcının bu zorlukla ilgili düşünceleri şu şekildedir:

"Açıkçası önceki programlama derslerim de çok iyi değildi. Bu derste de çok zorlandım çünkü öğrenecek çok şey vardı. Hem programlama öğrendik hem de tasarım yapmaya çalıştık. Bu arada Unity'yi de öğrenmek zorundaydık. Yani bu durum benim için zorlayıcı oldu. Örneğin projeye nereden başlayacağımı bilemedim. Nasıl ilerleyeceğim, neleri gerçekleştireceğim konusunda kafamda net bir kurgu olmadı." (D1_K7).

Katılımcıların zorlanmasının son nedeni ise hiçbirinin NYP dersinden önce Unity 3D editörünü kullanmamış olmalarıyla ilgilidir. Bu bağlamda Unity 3D'yi ilk defa bu derste kullanmaya başladıkları için bazı katılımcılar editörün kullanımını başlangıçta karmaşık bulmuşlardır. Katılımcıların bu konu hakkındaki düşünceleri şu şekildedir:

"Unity'ye giriş yaptık. Daha ilk defa gördüğüm bir program olduğum için biraz zorlandım anlamakta." (D1_K20, birinci hafta günlüğü).

"Unity'yi ilk defa kullanıyorum. Bu dersi almadan önce hiç kullanmamıştım. Bana karışık geldi. Dersteki örnek oyun da hiç Unity kullanmamış biri için zor. Oyuna bir şeyler eklerken projem bozulur diye korkuyorum." (D1_K23, üçüncü hafta günlüğü).

Yaşadıkları tüm güçlüklere rağmen katılımcılardan, programlama öz yeterlilik algılarının öğretim sürecinden nasıl etkilendiğine yönelik görüş alınmıştır. Katılımcıların %66'sı (N:19) öz yeterlilik algılarının olumlu etkilendiğini; programlama konusundan kendilerine güvenlerinin geliştiğini ve bir oyunu programlayabilecekleri konusundan alt yapı kazandıklarını belirtmişlerdir. Ancak katılımcıların geri kalan %34'ü (N:10) programlama konusundaki kendilerine güvenlerinin olumsuz etkilenerek becerileri konusunda güvenlerini sorguladıklarını ifade etmişlerdir.

Tüm bu verilere ilave olarak katılımcılardan ders materyali olarak kullanılan Karting Microgame oyun prototipini kullanılan sınıflar, nesneler ve kodların karmaşıklığı bağlamında değerlendirmeleri; bunun yanında prototipe eklenebilecek ilave özellikleri belirtmeleri istenmiştir. Katılımcıların %13'ü (N:4) kullanılan oyun prototipini çok karmaşık, %66'sı (N:20) orta düzeyde karmaşık, geri kalan %20'si (N:5) ise basit olarak nitelendirmişlerdir. İlave olarak bazı katılımcılar oyun prototipine farklı özellikler eklenerek oyunun görsel ve işlevsel olarak kalitesinin artabileceğinden bahsetmişlerdir. Katılımcılara göre bu özelikler; (i) oyunun çok oyunculu (multiplayer) olabileceği, (ii) oyunun başlangıcında farklı pistlerin veya farklı

araçların (farklı modellerde yarış arabaları, yarış motorları vb.) seçilebilmesi ve (iii) oyuna görsel efektler (bonus toplayınca ışıltı çıkması vb.) şeklindedir (Tablo 4.35). Tabloda yer alan öneriler değerlendirildiğinde, sunulan önerilerin katılımcıların başarı durumlarıyla herhangi bir ilişki göstermediği fark edilmiştir. Örneğin *çok oyunculu (multiplayer) oyun özelliği* önerisinde bulunan D1_K10 ve D1_K18 kodlu katılımcılar NYP Başarı Sınavı sonuçlarına göre sınıfın alt %25'lik bölümünde yer alırken D1_K10 kodlu katılımcı üst %25'lik bölümünde yer almıştır. Bu bağlamda değerlendirildiğinde, her ne kadar katılımcılar kullanılan oyun prototipinin daha görsel, ilgi çekici ve fonksiyonel olması için önerilerde bulunsalar da bu önerileri gerçekleştirmek için ihtiyaç bulunan programlama altyapısı ve yeterliliklere sahip olup olmadıkları konusunda herhangi bir öz değerlendirme sürecine girmedikleri belirlenmiştir.

Tablo 4.35. Karting Microgame oyun prototipine ilave edilmesi önerilen özelliklerin değerlendirilmesi

Kategori	Öneren Katılımcılar	Frekans (f)	Yüzde (%)
Çok Oyunculu (Multiplayer) Oyun Özelliği	D1_K4, D1_K5, D1_K8, D1_K10, D1_K14, D1_K18, D1_K20, D1_K23, D1_K26	9	31
Oyunu Farklı Araçlarla (Motosiklet, yarış arabası, 4x4 vb.) Oynama Seçenekleri	D1_K2, D1_K14, D1_K15, D1_K16, D1_K19, D1_K23	6	21
Oyunu Farklı Pistlerde Oynama Seçeneği	D1_K1, D1_K8, D1_K9, D1_K22	4	14
Efektler (Bonus toplama, çarpışma vb.)	D1_K5, D1_K17, D1_K27, D1_K28	4	14

4.2. İkinci Döngü

4.2.1. Tasarım Aşaması

4.2.1.1. Araştırma Sorusu 10: Birinci döngüden elde edilen deneyimler ışığında tasarlanan NYP dersinde ne gibi değişiklikler yapılabilir?

Eylem araştırmasının ikinci döngüsüne kaynak teşkil edecek şekilde; (i) birinci döngünün uygulama aşamasında karşılaşılan sorunlar, (ii) birinci döngünün değerlendirilmesinden sonra elde edilen sonuçlar ve (iii) pandemi koşulları dikkate alınarak geliştirilen Nesne Yönelimli Programlama dersinde bazı değişiklikler yapılmıştır. Bu değişiklikler;

- *Öğretim Programı,*

- *Öğretim Ortamı,*
- *Öğretim Materyalleri ve Etkinlikleri,*
- *Psikolojik Faktörler* temaları bağlamında gerçekleştirilmiştir.

4.2.1.1.1. Öğretim programı temasında gerçekleştirilen değişiklikler

Öğretim programı temasında gerçekleştirilen değişiklikler, *konulara ayrılan süre, konu dizilimi ve konu içeriği* olmak üzere üç kategori altında toplanmıştır. Öğretim programı temasında gerçekleştirilen değişikliklerle ilgili bilgiler Tablo 4.36'da sunulmuştur. Bununla birlikte öğretim programında gerçekleştirilen değişiklikler neticesinde NYP dersinin ikinci döngü için revize edilmiş hali Bölüm 4.2.1.2'de açıklanmıştır.

Tablo 4.36. Öğretim programı temasının alt kategorilerinde gerçekleştirilen değişikliklerle ilgili bilgiler

Kategori	**Tespit Edilen Durum**	**Gerçekleştirilen Değişiklik**
Konulara Ayrılan Süre	Unity 3D'de nesnelere C# kodu ekleme ve Monobehaviour metotları konuları için ayrılan iki ders saati yeterli olmamıştır.	Çalışmanın birinci döngüsünde "Unity 3D'de nesnelere C# kodu ekleme ve Monobehaviour metotları konuları" için iki ders saati olarak ayrılan süre, öğrencilerin yeterince uygulama yapamaması nedeniyle üç derse çıkarılmıştır.
Konu Dizilimi	Roll-A-Ball oyun prototipi bir hafta geç tanıtılmıştır.	İkinci döngünün konu diziliminde değişiklik yapılmıştır.
Konu İçeriği	Geçmiş dönemlerde alınan Algoritma ve Programlama Giriş ile Görsel Programlama dersleri, öğrencilerin sontest başarı puanlarını anlamlı bir şekilde etkilemiştir.	İkinci döngünün konu içeriğine, iki ders saati olmak üzere genel C# revizyon saati eklenmiştir.
	Katılımcılar tarafından odak grup görüşmesinde oyun prototipine farklı araç seçenekleri, pist seçimi, çok oyunculu oyun özelliği gibi özelliklerin eklenebileceği ifade edilmiştir.	Katılımcılardan elde edilen öneriler doğrultusunda mevcut oyun prototipine farklı pist seçimi ile başlama özelliği eklenmiştir. Buna ilave olarak oyuna görsel efektler ekleme konusu için öğrencilere video dersler hazırlanmıştır.

Tablo 4.36'da görüldüğü üzere gerçekleştirilen değişikliklerin ilki konulara ayrılan sürelerle ilgilidir. Birinci döngünün uygulama aşamasında zaman kısıtlaması nedeniyle yaşanan bazı sorunların dersin verimliliğini düşürmesi, Unity 3D'de nesnelere C# kodu ekleme ve Monobehaviour metotları konularına ayrılan sürenin bir saat artırılmasını gerekli kılmıştır. Zira uygulama sınavında "*Bir C# sınıfına ait Monobehavior metotlarını, verilen koşula göre oluşturur. (K21)"* kazanımının başarı yüzdesinin %72 çıkması (Şekil 4.31) ve sekiz öğrencinin bu konuda başarılı olamaması da (Tablo 4.24) bu bulguyu destekler niteliktedir.

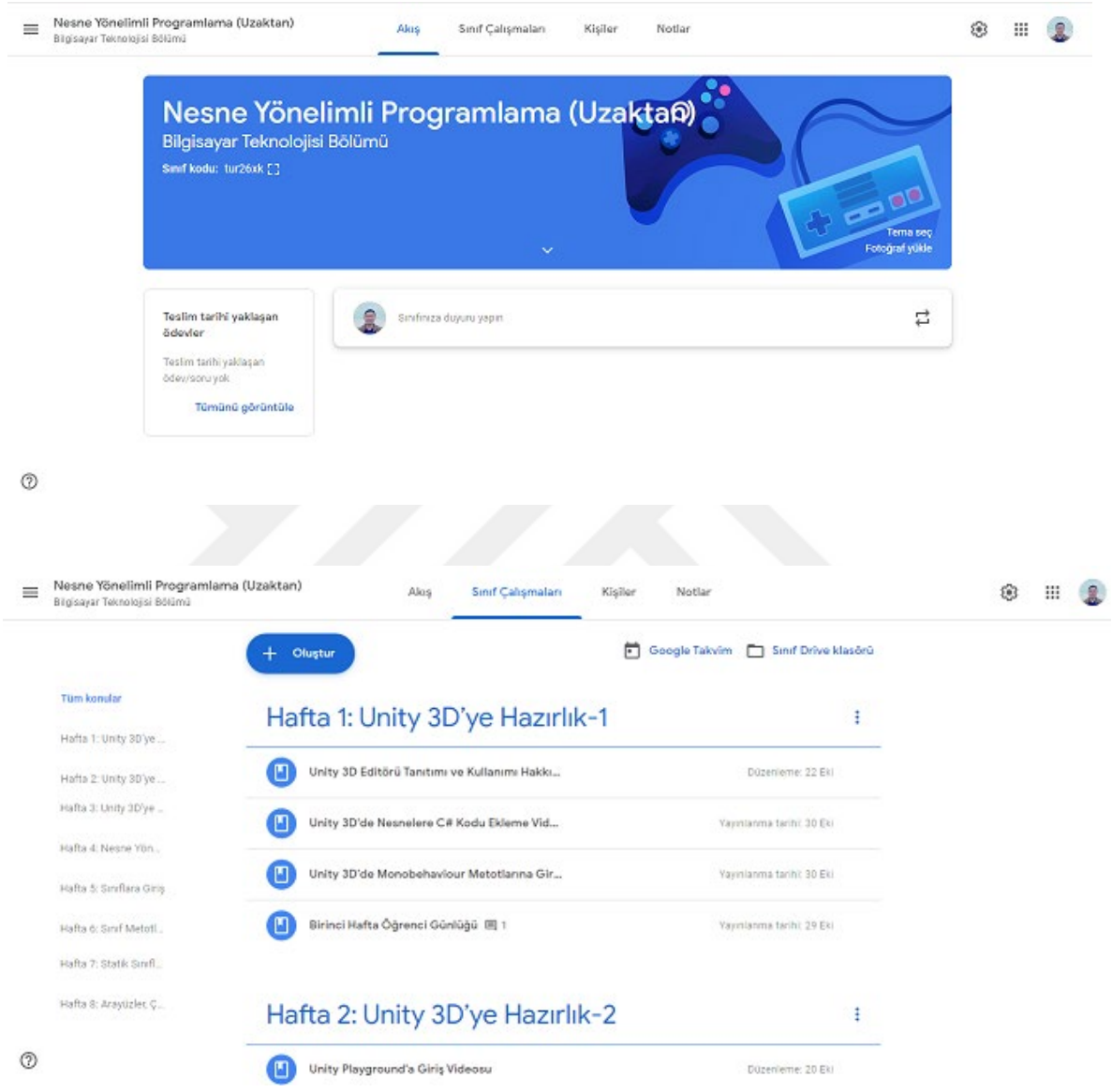
Öğretim programında gerçekleştirilen değişikliklerin ikincisi konuların dizilimleriyle ilgilidir. Araştırmacı günlüğünde Roll-A-Ball oyun prototipinin tanıtımın, "sınıf ve nesneler" konusundan bir önceki konu olan "nesne yönelimli programlamanın temelleri" konusunda yapılmasının gerektiği; bu sayede NYP kavramlarının Unity 3D'de kullanımına yönelik somut örnekler paylaşılabileceği ve dersin verimliliğinin artacağı raporlanmıştır. Bu sebeple uzman görüşleri doğrultusunda ikinci döngüde konu diziliminde değişiklik yapılmıştır.

Öğretim programında gerçekleştirilen değişikliklerin sonuncusu konu içerikleri ile ilgilidir. Birinci döngünün değerlendirilmesi aşamasının sonuçlarına göre, öğrencilerin geçmiş dönemlerde aldıkları Algoritma ve Programlama Giriş ile Görsel Programlama derslerinin öğrencilerin sontest başarı puanlarını anlamlı bir şekilde etkilediği bulunmuştur. Buna ilave olarak geçmiş programlama derslerindeki başarı durumu düşük olan öğrencilerin daha fazla dış desteğe ihtiyaç duydukları, oyunlarında karşılaştıkları hatalarını giderme konusunda daha fazla zorlandıkları gözlenmiştir. Bu bulgulara dayanarak ikinci döngünün öğretim programına *C#'ta değişkenler, karar yapıları, döngüler ve metotlar* konularını kapsayan iki saatlik genel revizyon saati eklenmiştir. Konu içeriğinde yapılan bir diğer değişiklik ise oyun prototipinde gerçekleştirilecek eklentilerle ilgilidir. Odak grup görüşmesine katılan öğrenciler, oyun prototipine farklı araç seçenekleri, pist seçimi, çok oyunculu oyun özelliği, görsel efektler gibi özelliklerin eklenebileceğini; bunların da oyunun kalitesini artırabileceğini ifade etmişlerdir. Öğrencilerin odak grup görüşmesinde ifade ettiği öneriler doğrultusunda mevcut oyun prototipine farklı pist seçimi ile başlama özelliği eklenmiştir. Buna ilave olarak oyuna görsel efektler ekleme konusu için öğrencilere video dersler hazırlanmıştır. Mevcut oyun prototipine farklı araçların (araba, kamyon, motosiklet vb.) ya da çok oyuncu ile oynama özelliğinin eklenmesi zaman yetersizliği sebebiyle mümkün olamamıştır.

4.2.1.1.2. Öğretim ortamı temasında gerçekleştirilen değişiklikler

Çalışmanın birinci döngüsü 2020 yılının eylül ayında, değerlendirme aşaması ile son bulmuştur. Birinci döngünün uygulama aşaması yüz yüze öğretim ile gerçekleştirilmiştir. Ancak çalışmanın ikinci döngüsü pandemi nedeniyle "Acil Uzaktan Öğretim – Emergency Remote Teaching" yöntemi ile gerçekleştirilmiştir. Çalışmanın öğretim ortamında yapılan en köklü değişiklik, derslerin uzaktan işlenmesi ile ilgilidir. Bu kapsamda bir Google sınıfı oluşturulmuştur (Şekil 4.39). Oluşturulan bu sanal sınıf ikinci döngü boyunca ders yönetim

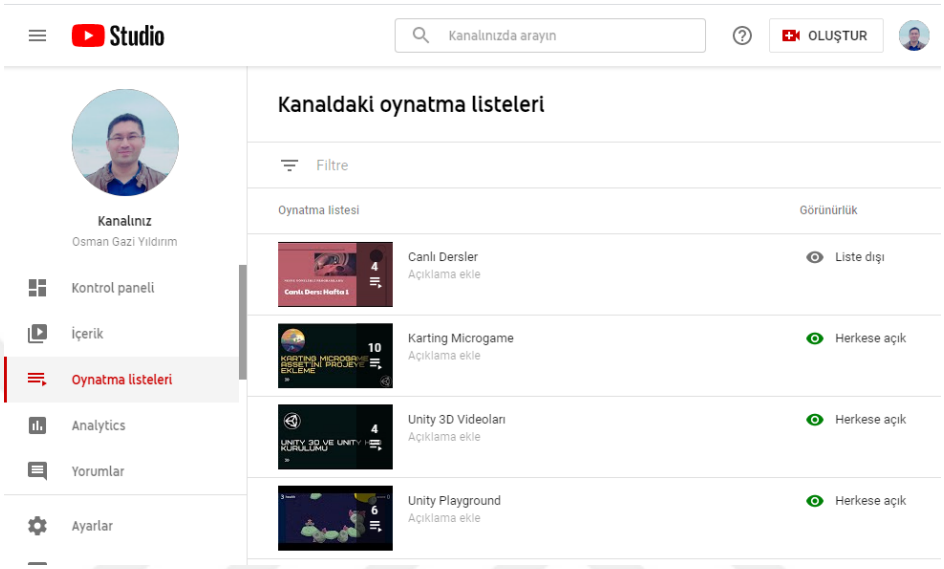
sistemi olarak kullanılmıştır. İkinci döngünün katılımcıları bu sınıfa kaydedilmiş; kullanılan tüm ders materyalleri ve etkinlikleri bu sanal sınıfa aktarılmıştır.



Şekil 4.39. Uzaktan öğretim için oluşturulan google sınıfı (sanal sınıf)

İkinci döngüde dersler Zoom programı kullanılarak işlenmiştir ve canlı dersler kaydedilmiştir. İlave olarak bir YouTube kanalı oluşturulmuş (Şekil 4.40); ders videolarının ve canlı derslerin kayıtları bu kanalda paylaşılmıştır. Öğrencilerle anlık haberleşmek ya da ders hakkında duyurular yayımlamak maksadıyla bir WhatsApp grubu kurulmuş (Şekil 4.41) ve dersi alan tüm öğrenciler bu gruba dâhil edilmişlerdir. Bu WhatsApp grubunun kurulmasındaki bir

diğer amaç ise öğrencilerin karşılaştıkları hatalarla ilgili detaylı bilgi edinmeyi sağlamaktır. Başka bir ifade ile bu WhatsApp grubu öğrencilerin karşılaştıkları hatalarla ilgili bir *veri toplama aracı* olarak kullanılmıştır.



Şekil 4.40. Ders için oluşturulan Youtube kanalı

Şekil 4.41. Ders için kurulan whatsapp grubu

4.2.1.1.3. Öğretim materyalleri ve etkinlikleri temasında gerçekleştirilen değişiklikler

Öğretim materyalleri ve etkinlikleri temasında gerçekleştirilen değişiklikler; (1) kazanımlar, (2) kodlama hataları, (3) konuların karmaşıklığı, (4) ders materyalleri, (5) geçmiş

programlama derslerindeki başarı durumuna göre düşük/yüksek başarılı karşılaştırması, (7) öğrenci günlükleri, (7) yazılım uyumsuzluğu ve (8) öğrenci devamsızlığı olmak üzere sekiz kategoride toplanmıştır. Öğretim materyalleri ve etkinlikleri temasının kategorileri bağlamında gerçekleştirilen değişikliklerle ilgili bilgiler Tablo 4.37'de sunulmuştur.

Tablo 4.37. Öğretim materyalleri ve etkinlikleri temasının alt kategorilerinde gerçekleştirilen değişikliklerle ilgili bilgiler

Kategori	**Tespit Edilen Durum**	**Gerçekleştirilen Değişiklik**
Kazanımlar	*"Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar. (K8)"* kazanımının başarı düzeyi beklenenin altında çıkmıştır.	▪ Karting Microgame oyun prototipinde tüm bonus ve engel sınıflarına çokbiçimlilik mekanizmasının kullanıldığı yeni metotlar eklenmiştir. ▪ İkinci döngünün 13.haftasına "Çokbiçimlilik Uygulaması" etkinliği eklenmiştir. ▪ Dersin sanal sınıfında NYP'de çokbiçimliliğin kullanımına yönelik örneklerin yer aldığı sunum paylaşılmıştır.
	"Kurucu metot kavramını açıklar. (K5)" kazanımının başarı düzeyi beklenenin altında çıkmıştır.	▪ Karting Microgame oyun prototipinde ses sisteminin oluşturulmasında kurucu metotların kullanılmasına yönelik revizyonlar yapılmıştır. ▪ Unity 3D'de kurucu metotların kullanılmasına yönelik örneklerin yer aldığı kod demoları hazırlanmıştır.
	"Kapsüllemenin (encapsulation) kullanım amaçlarını örnek vererek açıklar. (K9)*"* kazanımının başarı düzeyi beklenenin altında çıkmıştır.	İkinci döngünün 12.haftasına "UML Sınıf Diyagramı Analizi" etkinliği eklenmiştir.
Kodlama Hataları	Katılımcıların Monobehaviour metotlarının kullanımı, sınıf değişkenlere değer ataması ve bileşen özelliklerinin ayarlanması konularında hatalarla karşılaştıkları tespit edilmiştir.	Oyun projesi geliştirme esnasında sıklıkla karşılaşılabilecek hatalarla ilgili videolar hazırlanarak dersin sanal sınıfında paylaşılmıştır.
Konuların Karmaşıklığı	Bazı öğrenciler oyun projelerini gerçekleştirirken nereden başlayacaklarını ve nasıl bir yol haritası çizmeleri gerektiğini bilmediklerini belirtmişlerdir.	Öğrencilerin projelerinde ilerlemelerine yol göstermesi amacıyla oyun projesinde yer alan tüm sınıfların UML diyagramları oluşturularak sınıf etkileşimlerini içeren detaylı bir ders kaynağı oluşturulmuştur.
	Bazı öğrenciler tarafından Unity 3D'de yer alan İngilizce kavramların karmaşık olduğu ifade edilmiştir.	▪ Unity 3D'de yer alan kavramlarla ilgili e-kitap hazırlanarak dersin sanal sınıfında paylaşılmıştır. ▪ Canlı derslerin işlenişi esnasında karşılaşılan Unity 3D kavramları sıklıkla hatırlatılmıştır ve açıklanmıştır.

(Devam ediyor)

Tablo 4.37. Öğretim materyalleri ve etkinlikleri temasının alt kategorilerinde gerçekleştirilen değişikliklerle ilgili bilgiler (Devam)

Kategori	Tespit Edilen Durum	Gerçekleştirilen Değişiklik
Ders Materyalleri	Öğrenciler kullanılan Karting Microgame oyun prototipiyle ilgili yeterince Türkçe kaynağın olmadığını belirtmişlerdir.	Karting Microgame oyun prototipi ile ilgili videolar hazırlanarak dersin Youtube kanalında paylaşılmıştır.
	Öğrenciler Unity 3D ile ilgili daha fazla ders materyaline ihtiyaç duyduklarını ifade etmişlerdir.	Dersin sanal sınıfında Unity 3D ile ilgili Türkçe e-kitaplar, sunumlar ve videolar paylaşılmıştır.
	Birinci döngüde kullanılan Karting Microgame oyun prototipi, 15 Haziran 2020 tarihi itibariyle Unity 3D ekibi tarafından güncellenmiştir.	▪ İkinci döngü için Karting Microgame prototipinin yeni versiyonunun (Şekil 4.42) kullanılması kararlaştırılmıştır. ▪ Ders anlatımı ve proje geliştirme videoları yeni Karting Microgame prototipi kullanılarak yeniden hazırlanmıştır.
Düşük/Yüksek Başarılı Karşılaştırması	Geçmiş programlama derslerindeki başarı durumu düşük seviyede olan üç öğrenci projelerini bireysel olarak bitirmelerinin mümkün olmadığını belirtmişlerdir.	▪ Düşük başarılı öğrencilerin günlükleri haftalık olarak takip edilerek günlüklerinde bahsettikleri sorunlarla ilgili çözüm üretilmesi planlanmıştır.
	Geçmiş programlama derslerindeki başarı durumu göre düşük seviyede olan öğrencilerin tamamının sontest başarı kategorileri "düşük" olarak gerçekleşmiştir.	▪ Unity 3D'de hazırlanmış basit seviyede oyun projelerinin nasıl hazırlandığına yönelik ders videoları paylaşılarak düşük başarılı öğrencilerin paylaşılan projeleri hazırlamaları teşvik edilmiştir.
	Geçmiş programlama derslerindeki başarı durumu göre düşük seviyede olan öğrencilerin programlama dersine yönelik motivasyonları ve öz yeterlilik algıları yüksek seviyede olan öğrencilere göre düşük; kaygıları ise yüksek çıkmıştır.	▪ Bireysel oyun projelerinin geliştirilmesi sürecinde öğrencilerin yararlanabileceği videolar hazırlanarak dersin Youtube kanalında paylaşılmıştır.
Öğrenci Günlükleri	Bazı öğrencilerin günlüklerinde yeterince detaya girmediği belirlenmiştir.	Öğrencilerin kendilerini daha iyi ifade edebilmeleri için öğrenci günlüklerine yönergeler eklenmiştir.
	Bazı öğrencilerin günlüklerini doldurmadıkları ya da önceki haftanın aynısını gönderdiği tespit edilmiştir.	Rastgele örneklem usulü her hafta üç öğrencinin günlüğü seçilerek canlı derslerde bu günlüklerin içeriği ile ilgili geri bildirimlerde bulunulmuştur.
Yazılım Uyumsuzluğu	Visual Studio 2015 ile Unity 3D 2018 arasında uyum sorunları yaşanmıştır.	İkinci döngüde ilk döngüden farklı olarak Visual Studio Community 2015 yerine Visual Studio Community 2017 yazılımı kullanılmıştır.
Öğrenci Devamsızlığı	Önceki dersleri kaçıran öğrencilerin projelerini tamamlarken zorlandıkları belirlenmiştir.	Proje geliştirme sürecinde öğrencilerin yararlanabileceği videolar hazırlanarak dersin Youtube kanalında paylaşılmıştır.

Şekil 4.42. Karting Microgame oyun prototipinin yeni versiyonu

4.2.1.1.4. Psikolojik faktörler temasında gerçekleştirilen değişiklikler

Birinci döngünün değerlendirme aşamasında katılımcıların programlama beceri kaygısı puanları ile NYP Başarı Sınavı puanları; programlama kaygısı puanları ile programlama öz yeterlilik algısı puanları arasında negatif yönde anlamlı bir ilişki bulunmuştur. Bu bağlamda, öğrencilerin programlama kaygılarını en aza indirebilmek amacıyla; oyun projelerinde karşılaştıkları hataları düzeltemedikleri zaman akranlarına ulaşabilmeleri maksadıyla bir WhatsApp grubu kurulmuştur. Bunun yanında öğrencilerin oyunlarını geliştirirken karşılaştıkları güçlüklerin üstesinden gelebilmek ve olası sorularını yanıtlayabilmek maksadıyla çevrimiçi görüşme saati (office hour) planlanmıştır. Son olarak derslerde öğrencilerin motivasyonlarının artırtılması maksadıyla oyunlaştırma unsurlarından faydalanılmasına karar verilmiştir. Öğrencilerin motivasyonlarını artırmak maksadıyla çeşitli rozetler (badges) tasarlanmıştır ve derse ilerleme çubuğu (progress bar) eklenmiştir. Tasarlanan bu rozetlerin kullanım amaçları Tablo 4.38'de açıklanmıştır.

İkinci döngüde gerçekleştirilen *öğretim programı*, *öğretim ortamı*, *öğretim materyalleri ve etkinlikleri* ve *psikolojik faktörler* temaları bağlamında gerçekleştirilen tüm bu revizyonlar değerlendirildiğinde, daha önce Morrison ve diğerlerinin (2004) öğretim tasarımı modeline uygun olarak gerçekleştirilen öğretim tasarımının *içeriğin düzenlenmesi ve öğretimin sunumu* aşamalarında uzman görüşleri ışığında değişikliklerin yapıldığı söylenebilir.

Tablo 4.38. Ders için tasarlanan rozetler ve kullanım amaçları

Sıra No	Rozet	Kullanım Amacı
1		Canlı derslere düzenli bir şekilde katılan ve derse katkıda bulunan öğrenciler için tasarlanmıştır.
2		İçinde bulunulan ayda tüm canlı derslere katılan, tüm günlükleri dolduran ve dersin gerekliliklerini yerine getiren öğrenciler için tasarlanmıştır.
3		Oyun projesine yeni özellikler ekleyen (araçlar, menüler vb.) öğrenciler için tasarlanmıştır.
4		Gerçekleştiren Unity 3D mini sınavında tüm soruları doğru yanıtlayan öğrenciler için tasarlanmıştır.
5		Oyun projelerinde yaratıcı tasarımlar (pist, yeni bonuslar vb.) ve efekler kullanan öğrenciler için tasarlanmıştır.
6		Öğrenci günlüklerinde detaylı tasvirler ve anlatımlar yapan öğrenciler için tasarlanmıştır.

4.2.1.2. Revize edilen NYP dersinin öğretim programı

Geliştirilen NYP dersinin daha etkin ve verimli olabilmesi için gerçekleştirilen revizyonlar doğrultusunda, ikinci döngünün uygulama aşaması için hazırlanan NYP dersi haftalık öğretim programının özeti Tablo 4.39'da sunulmuştur. Bu öğretim programında en dikkat çekici olan

unsur programın ilk haftasına genel C# revizyonu konusunun eklenmesi olmuştur. Bunun yanında Unity 3D'de nesnelere kod ekleme konusu için bir hafta süre ayrılmıştır. Bu sayede birinci döngüden farklı olarak hem Unity 3D'de C#'ın kullanımı hem de Monobehaviour metotlarının tanıtımı için ayrılan süre uzatılmıştır. Son olarak Roll-A-Ball oyununun sınıf ve nesneler açısından çözümlenmesi aktivitesi, "Nesne Yönelimli Programlamanın Temelleri" konusunun işlendiği haftaya alınarak, NYP prensiplerinin ve mekanizmalarının bir oyun prototipindeki örneklerin çözümlenmesi vasıtasıyla açıklanması hedeflenmiştir. Haftalık ders planları Ek 12'de sunulmuştur.

Tablo 4.39. Revize edilen NYP dersinin haftalık öğretim programının özeti

Hafta No	**Konu**	**Alt Konu Başlıkları**
1	Genel C# Revizyonu & Unity 3D Editörü Tanıtımı	▪ C#'ta Değişkenler, Matematiksel ve Mantıksal İfadeler ▪ C#'ta Karar Yapıları, Döngüler ve Metotlar ▪ Unity 3D ve Unity Hub Kurulumu ▪ Unity Hub Yazılımı Tanıtımı ▪ Unity 3D'de Yeni Proje Oluşturma ▪ Unity 3D Editörü Tanıtımı ▪ Unity'de Layout'lar ▪ Transform Araçları ▪ Component, Camera, Scene, Asset, Material, Prefab ve Rigidbody Kavramları
2	Unity 3D'de Nesnelere Kod Ekleme	▪ Nesnelere C# Kodu Ekleme ▪ Monobehaviour Metotlarının Tanıtımı
3	Unity 3D Playground Projesi Oluşturma-1	▪ Unity Asset Store Tanıtımı ▪ Unity Playground Platformunun Tanıtımı ▪ Unity Playground Kod Parçacıklarının (Script) Tanıtımı ▪ Yeni Bir Playground Projesi Oluşturma ▪ Sorting Layer Kavramı ▪ Nesnelere Hazır Kod Parçaları Ekleme ▪ Parenting ve Childing Kavramları ▪ Particle Kavramı ▪ Collider'ların Kullanımı ▪ FPS Kavramı
4	Unity 3D Playground Projesi Oluşturma-2	▪ Tag, Projectile, Trigger, Attribute Kavramları ▪ Prefab Oluşturma ve Kullanma ▪ Can Sistemi (Health System) Kullanımı ▪ Lazer Sistemi (Laser System) Kullanımı ▪ Puan Sistemi (Point System) Kullanımı ▪ Kullanıcı Arayüz (UI) Nesnelerinin Tanıtımı ve Kullanımı ▪ Level Tasarımı

(Devam ediyor)

Tablo 4.39. Revize edilen NYP dersinin haftalık öğretim programının özeti (Devam)

Hafta No	Konu	Alt Konu Başlıkları
5	Nesne Yönelimli Programlamanın Temelleri	▪ Nesne Yönelimli Programlama Paradigmasının Amaçları ▪ Sınıf ve Nesne Kavramları ▪ Özellik (Attribute) ve Davranış (Behaviour) Kavramları ▪ Kurucu Metot Kavramı ▪ C#'ta Kullanılan Erişim Belirleyicileri ▪ Nesneler Arası Mesajlaşma Kavramı ▪ UML Sınıf Diyagramları ▪ Roll-A-Ball Oyununun Sınıf ve Nesneler Açısından Çözümlenmesi
6	Sınıflar ve Nesneler	▪ Unity 3D'de Sınıf ve Nesnelerin Kullanımı ▪ Karting Microgame Prototipi Tanıtımı ▪ Yeni Bir Unity 3D Projesi Oluşturma ▪ **Karting Microgame Oyun Prototipini Projeye Ekleme** ▪ **Karting Microgame Oyun Prototipine Çeşitli Unity 3D Assetleri Ekleme**
7	Sınıflar ve Nesneler	▪ Oyun Prototipinde Sınıf Oluşturma **(Puan Sistemi Sınıfı)** ▪ Oluşturulan Sınıfların UML Sınıf Diyagramlarını Çizme
8	Sınıflar ve Nesneler	▪ Oyun Prototipinde Sınıf Oluşturma **(Bonus ve Double Bonus Sınıfları)** ▪ **Oyun Prototipine UI Elementleri Ekleme** Oluşturulan Sınıfların UML Sınıf Diyagramlarını Çizme
9	Sınıf Metotları	▪ Oyun Prototipinde Sınıf Metotları Oluşturma ve Kullanma **(PuanYazdir() ve BonusYazdir() Metotları)** ▪ Başka Sınıfların Metotlarını Kullanma (**PuanArtir() Metodu**)
10	Sınıf Metotları	▪ Oyun Prototipinde Sınıf Metotları Olarak Monobehavior Metotlarını Oluşturma ve Kullanma **(Start(), Update(), FixedUpdate(), OnTriggerEnter() ve BonusYazdir() Metotları)**
11	Veri Yapıları	▪ Statik Özellik ve Sınıf Kavramları ▪ Statik Sınıflarla Normal Sınıfların Karşılaştırması ▪ Oyun Prototipinde Statik Metotların Tanımlanması ve Kullanılması **(Ses Sistemi ve Sesler Sınıfları)**
12	Arayüzler ve Kalıtım	▪ Kalıtım (Inheritance) Kavramı ▪ Üst-Sınıf (Superclass) ve Alt-Sınıf (Subclass) Kavramları ▪ Soyut (Virtual) Sınıflar ▪ Oyun Prototipinde Arayüz Tanımlama **(IEngel Sınıfı)** ▪ Oyun Prototipinde Kalıtım Kullanma **(Koni Engel ve Silindir Engel Sınıfları)**

(Devam ediyor)

Tablo 4.39. Revize edilen NYP dersinin haftalık öğretim programının özeti (Devam)

Hafta No	Konu	Alt Konu Başlıkları
13	Kapsülleme ve Çokbiçimlilik	▪ Veri Gizleme İlkesi ▪ Görünebilirlik Kuralları ▪ Çokbiçimlilik Kavramı ▪ Oyun Prototipinde Çokbiçimli Metotları Tanımlama ve Kullanma **(PuanAzalt() Metotları)**
14	Hata Kotarımı Görsel Efektler ve Pist Tasarımı	▪ Oyun Prototipinde Karşılaşılan Hatalar ve Çözümleri ▪ Oyun Prototipine Görsel Efekt Ekleme **(Bounce, SpeedPad, Particle)** ▪ **Pist Tasarımını Değiştirme**
15	Öğrenci Proje Sunumları	▪ Öğrencilerin Hazırladıkları Oyun Prototiplerinin Tanıtımı

4.2.2. Uygulama Aşaması

4.2.2.1. Araştırma Sorusu 11: Eylem planının uygulanması sürecinde katılımcıların ve araştırmacının yaşadığı sorunlar nelerdir?

İkinci döngünün uygulanması aşamasında karşılaşılan sorunlara ait veriler, (1) katılımcıların haftalık dijital günlüklerinden, (2) öğrencilerin canlı derslerdeki beyanlarından ve (3) katılımcılarla yapılan WhatsApp yazışmalarından faydalanarak elde etmiştir. Uygulanma aşamasında karşılaşılan sorunlara yönelik anlık çözümler üretebilmek için uzman görüşleri doğrultusunda bazı eylem kararları alınmıştır. İkinci döngünün uygulanması aşamasında karşılaşılan sorunlar ve bu sorunların çözümüne yönelik alınan eylem kararları Tablo 4.40'da verilmiştir.

Tablo 4.40'da görüldüğü üzere ikinci döngünün uygulama aşamasında ortaya çıkan sorunların ilk kategorisi, öğrencilerin oyun projelerini geliştirirken karşılaştıkları *kodlama hatalarıyla* ilgilidir. Katılımcılar internette aramalarına ve kendi bireysel çabalarına rağmen hatalarını gidermekte başarısız olmaları durumunda, dersin WhatsApp grubundan ve araştırmacıya gönderdikleri özel mesajlar yoluyla karşılaştıkları hatalarla ilgili yardım talebinde bulunmuşlardır. Katılımcılar projelerinde karşılaştıkları hatalarla ilgili düşüncelerini şu cümlelerle aktarmışlardır:

"Bu derste en çok zorlandığım husus çözemediğim bir sorunun ben çözmeye çalıştıkça onun daha çok sorun vermesi, kapatıp açtıkça sorunları tekrarlaması ve hiç çözülmemiş gibi olması. Bu durum beni adeta zorladı. Ama hocamızın yardımıyla ve araştırmalarım sonucunda bu sorunları da çözdüm açıkçası. Bir rahatlama üstümden bir

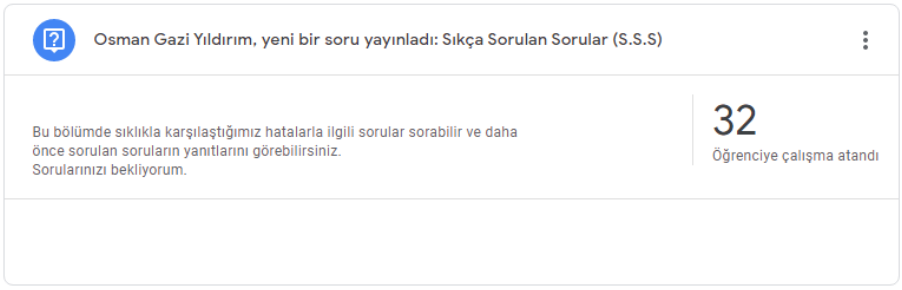
yük kalkması oldu çünkü bunu yapamamak beni sıkıntıya sokmasından ziyade motivasyonumu da düşürürdü. Ama çözebildik, sorunumun kalmaması beni çok rahatlattı." (D2_K8).

"Kodları yazarken ufak tefek yanlışlarım oldu bu hataları görmekte çok zorlandım tek tek kontrol etmem gerekti." (D2_K10).

Tablo 4.40. İkinci döngünün uygulanması aşamasında ortaya çıkan sorunlar ve sorunların çözümüne yönelik alınan eylem kararları

Tema	Uygulama Aşamasında Ortaya Çıkan Sorunlar	Sorunların Çözümüne Yönelik Alınan Eylem Kararları
Kodlama Hataları	Katılımcılar Karting Microgame prototipinin geliştirilmesi projesinde kodlama hatalarıyla karşılaşmışlardır.	Dersin Youtube kanalına bir dakikalık açıklamalar içeren Sıkça Sorulan Sorular videoları eklenerek katılımcılara karşılaştıkları hatalarla ilgili destek sağlanmıştır. Diğer öğrencilerin benzer sorunlarla ilgili bilgi sahibi olabilmeleri için ders portalına (Google Classroom) Sıkça Sorulan Sorular forumu eklenmiştir.
Yazılım Hataları	Bazı öğrenciler Unity 3D, Unity Hub ve Visual Studio 2017 yazılımlarının kurulumlarında hatalarla karşılaşmışlardır. Bazı öğrenciler Unity 3D, Unity Hub ve Visual Studio 2017 yazılımlarının lisanslamasında sorun yaşamışlardır.	Yazılımların kurulması sırasında karşılaşılan hataların çözümüne yönelik videolar hazırlanarak öğrencilerle paylaşılmıştır. Yazılımların lisanslanması konusunda videolar hazırlanarak öğrencilerle paylaşılmıştır.
Bağlantı Sorunları	Canlı dersin ilk haftasında araştırmacının internet bağlantısından kaynaklı bağlantı problemleri yaşanmıştır.	Araştırmacı bağlantı problemi olmayan bir konumda canlı derslere devam etmiştir.
Öğrenci-Öğretmen Etkileşimi	Canlı derslerde katılımcı-araştırmacı etkileşiminin sınırlı olduğu gözlemlenmiştir.	Öğrencilerle etkileşimi artırmak için soru-cevap ve tartışma seansları gerçekleştirilmiştir. Öğrencilerden kameralarını canlı dersler boyunca açık tutmaları istenmiştir.
Donanımsal Hatalar	Unity 3D editörü ve Visual Studio IDE'si belirli donanım konfigürasyonlarına gereksinim duyduğu için bazı katılımcılar bunu sağlamakta güçlük çekmişlerdir.	Bu sorun kategorisiyle ilgili eylem kararı alınamamıştır.

Öğrencilerin hatalarını çözmelerine yardımcı olabilmek maksadıyla ders portalında *"Sıkça Sorulan Sorular"* isminde bir bölüm hazırlanmıştır (Şekil 4.43). Katılımcılardan forum şeklinde olan bu bölümde karşılaştıkları sorunları paylaşmalarını istemiştir. Bununla tüm katılımcıların diğerlerinin deneyimlerinden istifade etmeleri amaçlanmıştır. Bununla birlikte karşılaşılan hataların çözümüne yardımcı olmak maksadıyla araştırmacı tarafından bir dakikalık çözüm videoları hazırlamıştır (Şekil 4.44). Her ne kadar araştırmacı kendisine özel mesajlarla aktarılan hata iletilerinin diğer katılımcıların da faydalanabilmesi için dersin Whatsapp grubundan ya da forumundan paylaşılmasını istese de öğrencilerin çoğu özel mesajlaşmayı tercih etmişlerdir.



Şekil 4.43. Google Classroom'da oluşturulan sıkça sorulan sorular forumu

Şekil 4.44. Youtube'da oluşturulan sıkça sorulan sorular videolar çalma listesi

Katılımcıların gerek WhatsApp grubundan gerekse özel mesajlarla aktardıkları kodlama hatalarına yönelik çözümlemeler Tablo 4.41'de verilmiştir. Katılımcıların paylaştıkları kodlama hataları incelendiğinde hataların kimi zaman Unity 3D editörü ayarlarından kaynaklandığı, kimi zaman ise Karting Microgame prototipine eklentiler yaparken (puan sistemi, ses sistemi, bildirim sistemi vb.) yapılan kodlama hatalarından kaynaklandığı görülmektedir.

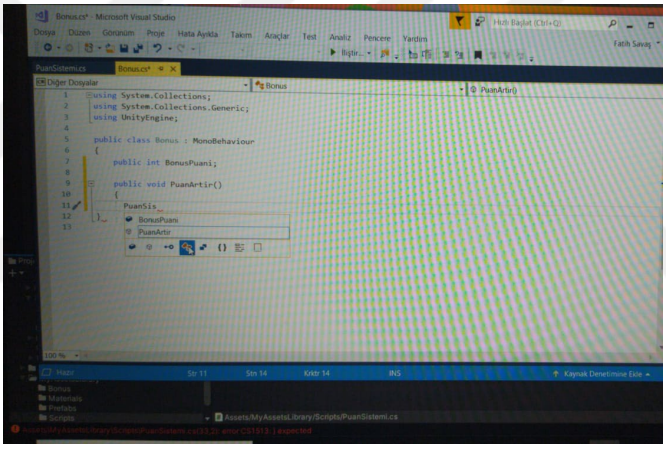
Tablo 4.41. Katılımcıların paylaştıkları kodlama hatalarıyla ilgili çözümlemeler

Hata Mesajı	Hata Açıklaması	Frekans (f)
Hata mesajı bulunmuyor ancak puan sistemi sınıfı kod tamamlama ekranında çıkmıyor (Şekil 4.45).	Puan sistemi sınıfı diğer sınıflar tarafından kullanılamıyor çünkü Visual Studio IDE'si Unity 3D için kod geliştirme ortamı olarak seçilmemiş.	8
Sınıf bileşen olarak eklenemiyor çünkü sınıf bulunamıyor (Can't add script component because the script class could not be found). (Şekil 4.46)	Visual Studio ile oluşturulurken sınıfa verilen isim ile sınıfın dosya sistemindeki ismi birbirinden farklılık içeriyor.	7

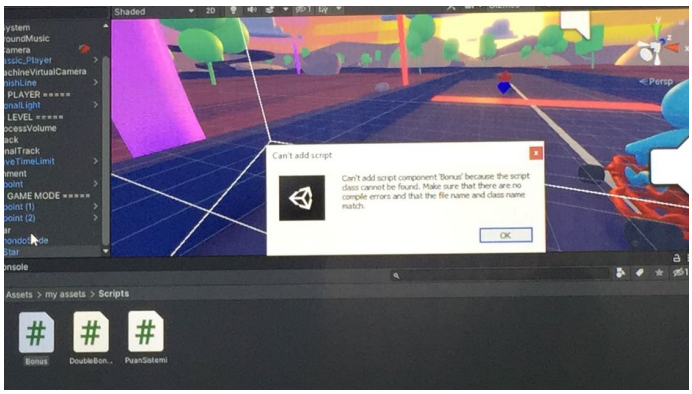
(Devam ediyor)

Tablo 4.41. Katılımcıların paylaştıkları kodlama hatalarıyla ilgili çözümlemeler (Devam)

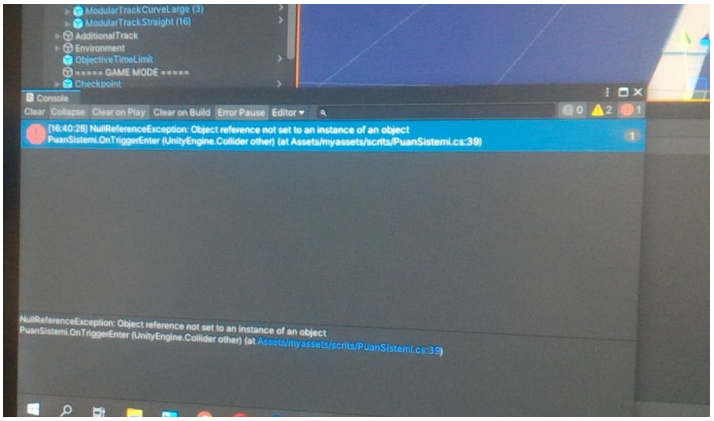
Hata Mesajı	Hata Açıklaması	Frekans (f)
Nesne başvurusu bir nesnenin örneğine ayarlanmadı (Object referans not set to an instance of an object). (Şekil 4.47)	Oluşturulan bir sınıfın üyesine herhangi bir oyun nesnesi ataması yapılmamış.	4
Tag ismi bulunamıyor (The name otherCompareTag does not exist in the current context). (Şekil 4.48)	Oyun nesnesinin etiketi (tag) tanımlanmadığı için Monobehaviour metodunda hata oluşuyor.	3
İki parametre alan aşırı yüklenmiş bir metot bulunmuyor (No overload for method takes two arguments).	Sınıf metodu için aşırı yükleme tanımlanmamış; dolaysıyla bu metodun mevcut halindeki parametre sayısı geçerli değil.	2
Hata mesajı bulunmuyor ancak statik sınıf metoduna erişilemiyor.	Sınıf metodu private erişim belirleyicisine sahip olduğu için diğer sınıflar tarafından erişilemiyor.	2



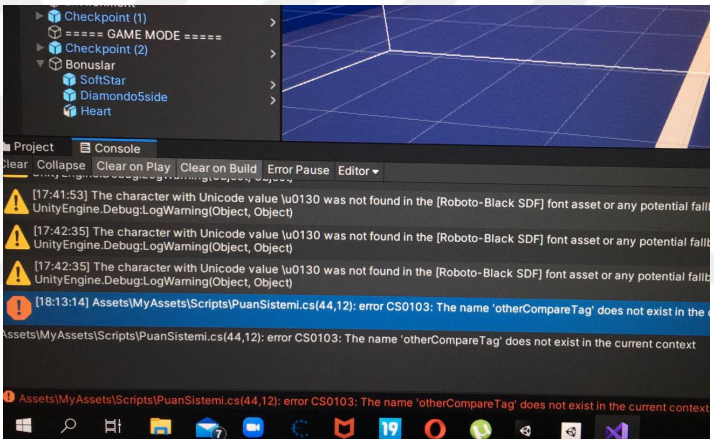
Şekil 4.45. "İlgili sınıf kod tamamlama ekranında çıkmıyor." hatası örnek ekran görüntüsü



Şekil 4.46. "Sınıf bileşen olarak eklenemiyor." hatası örnek ekran görüntüsü



Şekil 4.47. "Nesne başvurusu bir nesnenin örneğine ayarlanmadı." hatası örnek ekran görüntüsü



Şekil 4.48. "Tag ismi bulunamıyor." hatası örnek ekran görüntüsü

İkinci döngünün uygulanması aşamasında karşılaşılan bir diğer problem alanı, derste kullanılan oyun motoru olan Unity 3D editörü ile program geliştirme ortamı olan Visual Studio ile ilgilidir. Bu hatalar Unity 3D editörü ve Visual Studio IDE'sinin kurulumu, lisanslaması ve projelerde kullanılması esnasında katılımcıların karşılaştıkları sorunları kapsamaktadır. Bazı yaşadıkları sorunları dersin WhatsApp grubunda paylaşmışlardır. Bu sorunlarla ilgili çözüm önerileri araştırmacı tarafından ders portalında paylaşılmıştır. Katılımcılar yaşadıkları sorunlarla ilgili yaşadıkları deneyimleri şu cümlelerle aktarmışlardır:

"Bugün bu ders için gerekli olan programları bilgisayarıma indirdim. İndirmede birkaç küçük sorunla karşılaştım bu sorunlardan beni en çok yoran ise bilgisayarımın işletim sisteminin temelinde yatan bir sorundu. Bu yüzden bilgisayarıma yeni bir işletim sistemi kurdum." (D2_K9., ilk hafta günlüğü).

“Unity programında lisans sorunu ile karşılaştım. Bu sorunumun çözülmesinde internetten aratarak ve hocamızdan yardım alarak nasıl çözüleceğini öğrendim, uyguladım ve sorunum ortadan kalktı.” .” (D2_K28., ilk hafta günlüğü).

“Projemde puan sistemlerinin ilk aşamasını yeni bitirdim çünkü Visual Studio ile ilgili bir sorunla karşılaştım. Halletmem biraz uzun sürdü fakat hallettim ve şuan projemi hızlı bir şekilde bitirmeye odaklandım.” (D2_K13., Sekizinci hafta günlüğü).

Bağlantı hataları sorun kategorisi, canlı derslerin işlenmesi esnasında karşılaşılan bağlantı kopmalarını kapsamaktadır. Özellikle dersin birinci haftasında araştırmacının internet bağlantısından kaynaklı kopmalar yaşanmış ve dersin akışında aksaklıklar meydana gelmiştir. Katılımcılar bu sorunu şu cümlelerle ifade etmişlerdir:

“Derste bağlantının sürekli kopmasından ötürü ders verimli işlenemedi. Bu sıkıntı bir daha olmazsa önümüzdeki haftalarda derslerden daha çok verim alacağımı düşünüyorum.” (D2_K26., ilk hafta günlüğü).

“Dersim'de [canlı derste] internet kopmaları nedeniyle pek fazla verim alamadım ama kesinlikle verim alacağıma inanıyorum isteyerek katılıyorum derslere ve çabalıyorum anlamak için yapılan videoları tek tek izliyorum diğer derslerimizde daha da akıcı şekilde ders işleyebilirsek teknik arızalar çıkmazsa mükemmel olur bunun dışındaki her şey yolunda gidiyor.” (D2_K13., ilk hafta günlüğü).

Canlı derslerin ilk haftasında yaşanan bağlantı problemi ile ilgili araştırmacı gerekli önlemi almış ve canlı dersi yaptığı konumu değiştirmiştir. Yapılan bu değişiklikle canlı derslerin diğer haftalarında araştırmacıdan kaynaklı herhangi bir internet problemi yaşanmamıştır. Bunun yanında bazı katılımcıların uygulama sürecinde kendi internet bağlantılarından kaynaklı benzer sorunlar yaşadığı gözlemlenmiştir. Katılımcılardan bazıları ise canlı derslerde özellikle ekran paylaşımı esnasında görüntülerde donmaların yaşandığını ve bu donmalar nedeniyle araştırmacıyı takip edemediklerini şu cümlelerle bildirmişlerdir:

“Dersin sevmediğim tek sıkıntısı bazen internetten dolayı yapılan şeyi görmememiz. Tabii video çektiğiniz için açıp Youtube’dan tekrar bakabiliyorduk.” (D2_K20)

“Derste en çok zorlandığım durum ekran paylaşımında görüntünün kasmasıydı. Fakat kendi yaptığım projede derste oturmayan her şeyin oturduğunu söyleyebilirim.” (D2_K4).

Bu sorunla ilgili araştırmacı canlı derslerin işlenmesi esnasında sık sık katılımcılara görüntü kaybı yaşayıp yaşamadıklarını sormuş ve yaşanması durumunda canlı derste yapılan işlemleri (kod yazma, tasarım oluşturma, hata giderme vb.) tekrar gerçekleştirmiştir.

Canlı derslerde yaşanan bir diğer sorun ise *öğretmen-öğrenci etkileşimiyle* ilgilidir. Araştırmacı, canlı derslerin okul ortamında gerçekleşen yüz yüze derslerdeki etkileşimi sağlayamadığını şu cümlelerle ifade etmiştir:

"Yüz yüze ders yapmak benim için her açıdan daha verimli ve avantajlı idi. Yüz yüze derslerde öğrencilerim uygulama yapabiliyorlar ve ben laboratuvarda aralarında dolaşarak neyi yapıp yapamadıklarını kontrol edebiliyorum. Jest ve mimiklerinden, sorularından, yaptıklarından dersin gidişatı hakkında bilgi sahibi olabiliyordum. Pandemi nedeniyle dersi uzaktan yapmak zorunda kaldım, bu etkileşimi biraz etkiledi. Sadece konuya ve kendime yoğunlaşıyormuşum gibi hissediyorum. Buna alışmak zaman alabilir." (İlk hafta günlüğü).

"Genel olarak dersten ve öğrencilerimden memnunum. Ancak etkileşimin derinliği ve sıklığı açısından yüz yüze dersin yerini tutması zor görünüyor. Dersin verimliliği konusunda endişelerim bulunuyor. Ben dersi anlatırken ekran paylaşımı esnasında katılımcıların çoğunun kameralarını ve seslerini kapattığını gözlemliyorum; bazen sorduğum sorulara yanıt alamıyorum. Bu nedenle dersin tüm bölümlerinde (hazırlık, anlatım, ekran paylaşımı, soru cevap vb.) kameraların açılmasını istedim. Bu isteği sık sık yinelemek zorunda kalmak dersin akışını bozuyor. Başka bir handikap ise Zoom programında Unity'yi etkin bir şekilde kullanabilmek için ekran paylaşımı esnasında öğrencilerin birkaçının dahi olsa gözüktüğü küçük resimleri kapatıyorum çünkü sağda inspector window var ve tam ekranda öğrencilerin resimleri açıkken kullanamıyorum." (İkinci hafta günlüğü).

İkinci döngünün uygulanması aşamasında yaşanan sorunlarla ilgili son kategori, *donanımsal hatalarla* ilgilidir. Unity 3D editörü ve Visual Studio IDE'si belirli donanım konfigürasyonlarına gereksinim duyduğu için bazı katılımcılar bunu sağlamakta güçlük çekmişlerdir. Bazı katılımcılar bilgisayarlar temin etmek zorunda kalmış, bunu gerçekleştiremeyenler ise ya derse mevcut düşük donanımlı bilgisayarlarıyla devam etmişler ya da ders uygulamalarını ve projesini tamamlayamamışlardır. Katılımcılar bu konu hakkındaki deneyimlerini şu cümlelerle aktarmışlardır:

"Derse hazırlık için öncelikle bilgisayarımı değiştirdim uygulamaları kullanabilmek için." (D2_K21., ilk hafta günlüğü).

"Bilgisayarım biraz kassa da [zorlansa da] ufak bir bekleme süresi ile (kahve koyma) işim çözülüyor. Ama sistemim eski olduğu için uygulamalar arası geçişte ve az da olsa FPS [dakikadaki frame sayısı] düşüşü yaşıyorum." (D2_K3., altıncı hafta günlüğü).

"Bilgisayarımda yeterli özellik bulunmadığı için projemde yavaş ilerlemek zorunda kalıyorum." (D2_K20., sekizinci hafta günlüğü),

"Bilgisayarım olmadığı için uygulamaları yapamadım ve bazı bilgiler tabiri caizse havada kaldı." (D2_K19).

4.2.3. Değerlendirme Aşaması

4.2.3.1. Araştırma Sorusu 12: Eylem planının uygulanmasından sonra katılımcıların nesne yönelimli programlama bilgi ve beceri düzeyleri nasıl bir değişim göstermiştir?

Katılımcıların uygulama öncesi ve sonrasına ait programlama bilgi ve becerileri NYP Başarı Sınavı kullanılarak belirlenmiştir. Katılımcıların NYP Başarı Sınavı öntest ve sontest puanlarına yönelik betimsel istatistikler Tablo 4.42'de sunulmuştur.

Tablo 4.42. NYP Başarı Sınavı öntest ve sontest puanlarına yönelik betimsel istatistikler

	Öntest			Sontest		
	N	**x̄**	**ss**	**x̄**	**ss**	**Fark**
NYP Başarı Sınavı	30	4.56	3.3	85.92	13.04	81.36

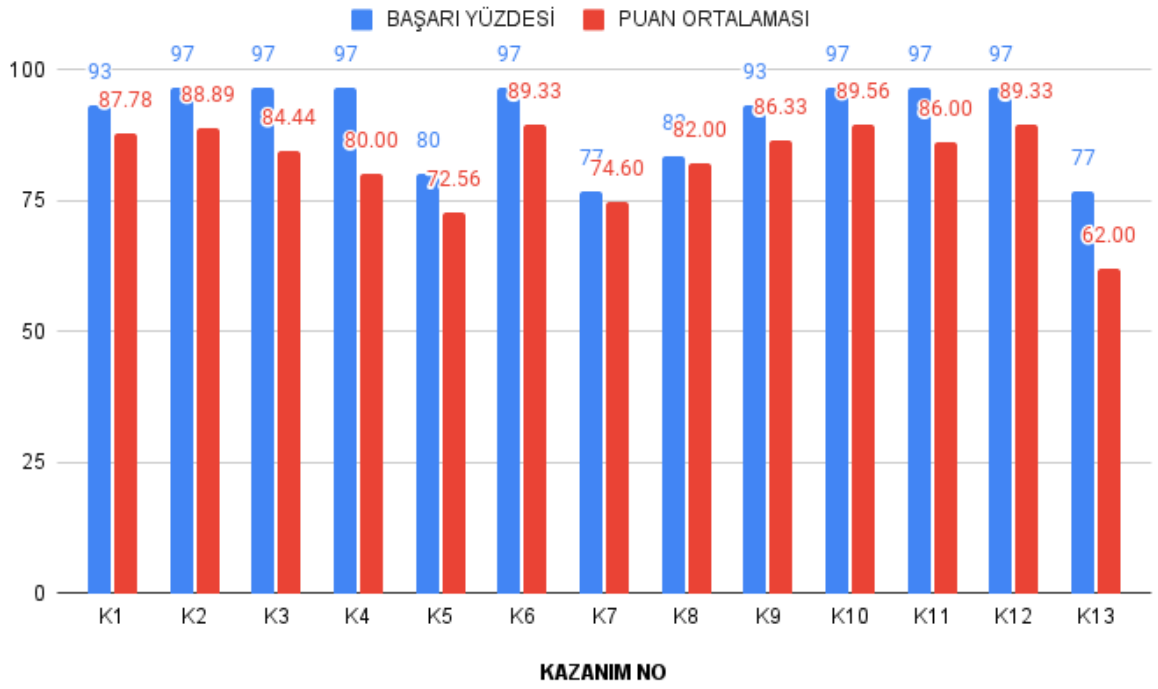
Buna ilave olarak katılımcıların öntest ve sontest puanları istatistiksel olarak karşılaştırılmıştır. Bu karşılaştırmayı gerçekleştirmek maksadıyla öncelikle test sonuçlarının normal dağılıp dağılmadığı kontrol edilmiştir. Shapiro-Wilk testi sonuçlarına göre katılımcıların öntest ve sontest puanları normal dağılmamaktadır ($p<.05$). Bu sebeple katılımcıların NYP Başarı Sınavı öntest ve sontest puanları karşılaştırmak amacıyla Wilcoxon sıralı işaretler testi kullanılmıştır (Pallant, 2020). Test sonuçları Tablo 4.43'te verilmiştir. Tablo 4.43'e göre NYP Başarı Sınavı için gerçekleştirilen Wilcoxon sıralı işaretler testi sonuçları, katılımcıların uygulamaya katıldıktan sonra başarı puanlarında büyük bir etki büyüklüğü ile ($r=.78$) istatistiksel olarak anlamlı değişiklikler olduğunu ortaya koymuştur ($z=-4.29$, $p=.00$). Bu sonuca göre gerçekleştirilen NYP eğitimi öğrencilerin NYP bilgi ve becerilerini anlamlı bir şekilde etkilemektedir.

Tablo 4.43. NYP Başarı Sınavı öntest-sontest puanlarının karşılaştırılmasına ilişkin Wilcoxon sıralı işaretler testi sonuçları

Puan	**Sıralar**	**N**	**Sıra x̄**	**Sıra ∑**	**z**	**p**
NYP Başarı Sınavı	Negatif sıralar	0	0	0	-4.29	.00
	Pozitif sıralar	30	13.00	355.00		
	Eşit	0				
	Toplam	30				

Öntest sontest karşılaştırmasına ilave olarak NYP Başarı Sınavına ait kazanımlar incelenmiş olup katılımcıların her bir kazanımdan elde ettikleri ortalama başarı yüzdeleri belirlenmiştir. Şekil 4.49'da NYP Başarı Sınavının teorik kısmına ilişkin kazanımlara ait başarı

yüzdeleri sunulmuştur. Şekle göre başarı yüzdeleri 77 ile 97 arasında değişmektedir. En düşük başarı yüzdeleri "*Verilen bir C# sınıfını analiz ederek bu sınıfta kullanılan NYP prensiplerini (kalıtım, kapsülleme, çokbiçimlilik vb.) çözümler. (K13)", "Kalıtımın (inheritance) kullanım amaçlarını örnek vererek açıklar (K7)"* ile *"Kurucu metot kavramını örneklerle açıklar. (K5)"* kazanımlarında gerçekleşmiştir.



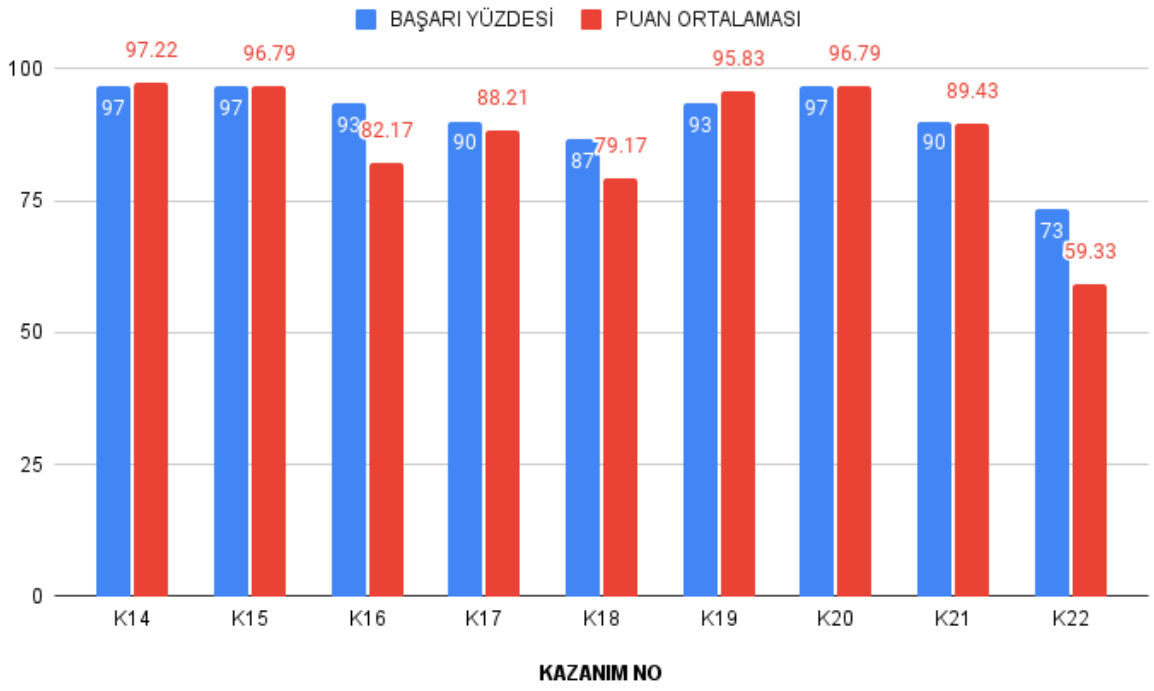
Şekil 4.49. Başarı Sınavının teorik kısmına ilişkin kazanımlara ait başarı yüzdeleri ve puan ortalamaları

NYP Başarı Sınavının ayrıntılı incelenmesi sonucunda sınavın teorik kısmına ilişkin kazanımlardan başarısız olan (hiç yanıt vermeyen ya da yanlış yanıt veren) katılımcı sayıları da tespit edilmiştir (Tablo 4.44). Tabloya göre katılımcılar en çok "*Verilen bir C# sınıfını analiz ederek bu sınıfta kullanılan NYP prensiplerini (kalıtım, kapsülleme, çokbiçimlilik gibi) çözümler. (K13)*" kazanımında başarısız olmuştur.

Tablo 4.44. NYP Başarı Sınavının teorik kısmına ilişkin kazanımlarında başarısız olan katılımcı sayıları

Kazanım No	Kazanım	Frekans (f)
K7	Kalıtımın (inheritance) kullanım amaçlarını örnek vererek açıklar.	7
K13	Verilen bir C# sınıfını analiz ederek bu sınıfta kullanılan NYP prensiplerini (kalıtım, kapsülleme, çokbiçimlilik vb.) çözümler.	7
K5	Kurucu metot kavramını örneklerle açıklar.	6
K8	Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar.	5
K1	Sınıf kavramını örneklerle açıklar.	2
K9	Kapsüllemenin (encapsulation) kullanım amaçlarını örnek vererek açıklar.	2
K2	Nesne kavramını örneklerle açıklar.	1
K3	Özellik (attribute) kavramını örneklerle açıklar.	1
K4	Davranış (behaviour) kavramını örneklerle açıklar.	1
K6	Sınıf ile nesne arasındaki ilişkiyi betimler.	1
K10	Statik sınıflar ile normal sınıfları karşılaştırır.	1
K11	C# programlama dilinde kullanılan erişim belirleyicilerin kullanım amaçlarını belirtir.	1
K12	Verilen UML Sınıf diyagramlarında yer alan sınıflara ait özellik ve davranışlarını çözümleyerek bu sınıflar arasındaki ilişkileri açıklar.	1

Teorik kazanımlara ilave olarak, NYP Başarı Sınavının uygulama kısmına ait kazanımlara ilişkin başarı yüzdeleri de belirlenmiştir (Şekil 4.50). Şekle göre başarı yüzdeleri 73 ile 97 arasında değişmektedir. En düşük başarı yüzdeleri *"Kullanıcı arayüzü (UI) nesnelerini, oluşturduğu C# sınıfında kullanır. (K22)"* ile *"Verilen koşullara göre kalıtım mekanizmasını C# programlama dilinde gerçekleştirir. (K18)"* kazanımlarında gerçekleşmiştir. Uygulama sınavının kazanımlarında başarısız olan katılımcıların sunulduğu Tablo 4.45 bu durumu teyit etmektedir. Bu durumun ortaya çıkmasında, birinci döngüde zaman yetersizliği sebebiyle kullanıcı arayüzü (UI) nesnelerinin katılımcıların kullanımına hazır olarak verilmiş olması rol oynamış olabilir. Nitekim birinci döngüde katılımcılardan bu nesneleri oluşturup oyun projelerinde kullanmaları istenmemiştir. Ancak ikinci döngüde kullanıcı arayüzü nesneleri katılımcılar tarafından oluşturulmuş ve kullanılmış; dolayısıyla bu kazanımın başarısı sadece bu döngüde ölçülmüştür.



Şekil 4.50. NYP Başarı Sınavının uygulama kısmına ilişkin kazanımlara ait başarı yüzdeleri ve puan ortalamaları

Tablo 4.45. NYP Başarı Sınavının uygulama kısmına ilişkin kazanımlarında başarısız olan katılımcı sayıları

Kazanım No	Kazanım	Frekans (f)
K22	Kullanıcı arayüzü (UI) nesnelerini oluşturduğu C# sınıfında kullanır.	8
K18	Verilen koşullara göre kalıtım mekanizmasını C# programlama dilinde gerçekleştirir.	4
K17	UML sınıf diyagramında belirtilen çokbiçimli metotları oluşturur ve kullanır.	3
K21	Bir C# sınıfına ait Monobehavior metotlarını, verilen koşula göre oluşturur.	3
K16	UML sınıf diyagramı verilen arayüzü (interface), C# programlama dilinde oluşturur.	2
K19	UML sınıf diyagramı verilen sınıfı C# programlama dilinde oluşturur.	2
K14	Bir sınıfa ait özelliklerin erişim belirleyicilerini verilen koşula göre değiştirir.	1
K20	Önceden oluşturulmuş bir C# sınıfına ait özelliklerin değer atamasını verilen koşula göre yapar.	1
K15	Önceden oluşturulmuş bir C# sınıfını, belirtilen oyun nesnesine bileşen (component) olarak ekler.	1

Katılımcıların sontest teorik sınav notları ile uygulama sınav notları arasında bir ilişki olup olmadığının belirlenebilmesi amacıyla Pearson çarpım moment korelasyon analizi gerçekleştirilmiştir (Tablo 4.46). Tabloya göre sontest teorik sınav notları ile uygulama sınav

notları arasında p<.01 düzeyinde .752'lık pozitif yönde çok yüksek düzeyde bir ilişki bulunmuştur.

Tablo 4.46. Sontest teorik sınav ile uygulama sınavı Pearson çarpım moment korelasyon analizi sonuçları

	Sontest Teorik Sınav	**Sontest Uygulama Sınavı**
Sontest Teorik Sınav	—	.752**
Sontest Uygulama Sınavı	.752**	—

** p< .01.

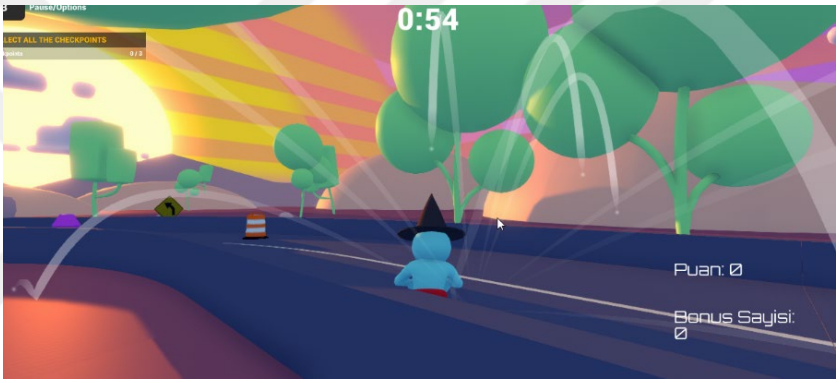
NYP Başarı Sınavının yanında, öğrencilerin geliştirdikleri oyun projeleri de değerlendirilmiştir (Tablo 4.47). Bu değerlendirmeye göre katılımcılarının tamamının oyun prototipinin görünüm özelliklerine yeni objeler (resimler, levhalar, engel nesneleri vb.) eklediği (örn. Şekil 4.51); çoğunun pist tasarımını kendi isteklerine göre değiştirdiği (örn. Şekil 4.52), ve oyuna görsel efektler ekledi (örn. Şekil 4.53 ve Şekil 4.54) görülmüştür. Ancak hiçbir katılımcının oyun projesine derste uygulaması yapılan puan sistemi, ses sistemi, bildirim sistemi gibi sınıfların dışında yeni sınıflar eklemediği de tespit edilmiştir.

Tablo 4.47. Katılımcıların oyun projelerinin değerlendirilmesi sonuçları

Oyun Projeleri Üzerinde Gerçekleştirilen İşlem	**Frekans (f)**
Oyunun görünüm özelliklerini değiştirecek şekilde yeni objelerin (resimler, levhalar, engel nesneleri vb.) eklenmesi	30
Pist tasarımının değiştirilmesi	28
Oyuna görsel efektlerin (Çarpışma efekti, laserlight vb.) eklenmesi	27
Yeni bonus nesnelerinin (sınıflarının) eklenmesi (Şekil 4.55)	19
Yeni engel nesnelerinin (sınıflarının) eklenmesi	13

Şekil 4.51. Yeni objelerin eklenmesi

Şekil 4.52. Pist tasarımın değiştirilmesi



Şekil 4.53. Çarpışma efektlerinin eklenmesi

Şekil 4.54. Laserlight efektinin eklenmesi

Şekil 4.55. Yeni bonus nesnelerinin eklenmesi

Katılımcıların programlama dersine karşı motivasyonlarının, programlama kaygılarının ve programlamaya yönelik öz yeterlilik algılarının sontest başarı puanları ile ilişkisini incelemek amacıyla korelasyon analizi gerçekleştirilmiştir. Ölçeklere ait betimsel istatistikler Tablo 4.48'de, korelasyon analizi sonuçları Tablo 4.49'da sunulmuştur. Pearson çarpım moment korelasyon analizi sonucunda, Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeğinin sadece "Ödül ve Takdir" alt faktörü ile katılımcıların sontest başarı puanları arasında anlamlı bir ilişki bulunmuştur ($r=.572$, $p<.001$). Benzer şekilde Programlama Kaygı Ölçeğinin Programlama Beceri Kaygısı alt faktörü ile katılımcıların sontest başarı puanları arasında negatif yönde anlamlı bir ilişki bulunmuştur ($r=-.345$; $p<.05$). Buna zıt olarak, Programlama Kaygı Ölçeğinin Akran Kaygısı alt faktörü ve katılımcıların programlamaya yönelik öz yeterlilik algıları ile sontest başarı puanları arasında anlamlı bir ilişki bulunmamıştır (sırasıyla $r=-.297$ ve $r=-.055$). Ancak katılımcıların programlamaya yönelik öz yeterlilik algıları ile Programlama Kaygı Ölçeğinin Akran Kaygısı ve Programlama Beceri Kaygısı alt faktörleri arasında istatistiksel açıdan negatif yönde anlamlı bir ilişki saptanmıştır (sırasıyla $r=-.456$ ve $r=-.635$).

Tablo 4.48. Ölçeklere ilişkin betimsel istatistik sonuçları

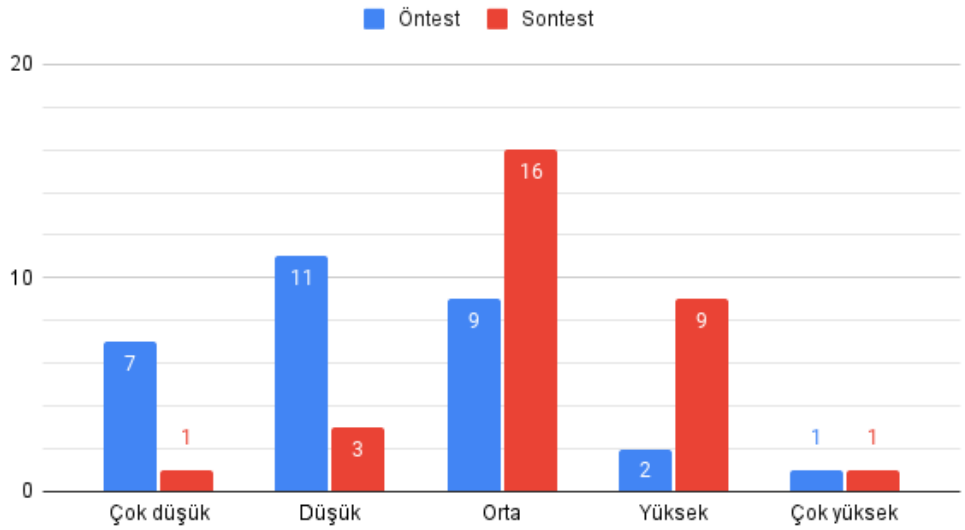
		Sontest	
	N	**x̄**	**ss**
Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği			
Tutum ve Beklenti	30	5.18	.76
Zorlayıcı Amaçlar	30	4.28	1.14
Belirgin Hedefler	30	5.02	.83
Ödül ve Takdir	30	5.11	1.03
Ceza	30	2.97	1.06
Sosyal Baskı ve Rekabet	30	4.18	1.13
Toplam Puan	30	4.56	.80
Programlama Kaygı Ölçeği			
Akran Kaygısı	30	2.38	1.1
Programlama Beceri Kaygısı	30	2.75	.94
Toplam Kaygı Puanı	30	2.59	.91
Programlama Öz Yeterlilik Ölçeği			
Toplam Puan	30	4.08	.63

Tablo 4.49. Pearson çarpım moment korelasyon analizi sonuçları

	Tutum ve Beklenti	Zorlayıcı Amaçlar	Belirgin Hedefler	Ödül ve Takdir	Ceza	Sosyal Baskı ve Rekabet	Akran Kaygısı	Programlama Beceri Kaygısı	Programlama Öz yeterlilik Algısı	NYP Başarı Sınavı (Sontest)
Tutum ve Beklenti	—									
Zorlayıcı Amaçlar	.536**	—								
Belirgin Hedefler	.859**	.626**	—							
Ödül ve Takdir	.749**	.345	.734**	—						
Ceza	.134	.485**	.180	.192	—					
Sosyal Baskı ve Rekabet	.451*	.578**	.568**	.490**	.435*	—				
Akran Kaygısı	-.193	-.424*	-.261	-.008	-.023	-.068	—			
Programlama Beceri Kaygısı	-.397*	.672**	-.417*	-.073	-.264	-.378*	.631**	—		
Programlama Öz yeterlilik Algısı	.784**	.807**	.761**	.466**	.285	.496**	-.456*	-.635**	—	
NYP Başarı Sınavı (Sontest)	.328	-.035	.380	.572**	-.013	-.004	-.297	-.345*	.055	—

* p< .05, ** p< .01.

NYP Başarı Sınavının ayrıntılı analizlerine ilave olarak katılımcıların uygulama öncesinde ve sonrasında algıladıkları programlama bilgi seviyesindeki değişim, katılımcı bilgi formu kullanılarak incelenmiştir. Katılımcıların algıladıkları programlama bilgi seviyeleri karşılaştırma grafiği Şekil 4.56'da verilmiştir.



Şekil 4.56. Uygulama öncesinde algılanan programlama bilgi seviyesi dağılım grafiği

Şekil 4.56'ya göre katılımcıların uygulama öncesinde algıladıkları programlama bilgi seviyeleri ağırlıklı olarak (N: 27, %90) çok düşük, düşük ve orta olarak nitelendirilmiştir. Ancak uygulama sonrasında kendilerini programlama bilgisi seviyesi açısından çok düşük, düşük ve orta olarak nitelendiren katılımcıların yüzdesi önemli derecede azalarak %66 (N:20) seviyesine düşmüştür. Uygulama öncesinde katılımcıların %23'ü kendilerini çok düşük seviye olarak nitelendirirken uygulama sonrasında bu seviyede bir katılımcı kalmıştır. Benzer şekilde uygulama öncesinde katılımcıların %36'sı kendilerini düşük seviye olarak nitelendirirken uygulama sonrasında bu oran %10'a düşmüştür. Ancak bu oranlar azalırken uygulama sonrasında algıladıkları programlama bilgi seviyesi orta ve yüksek olan katılımcıların oranı %36'dan %83'e çıkmıştır.

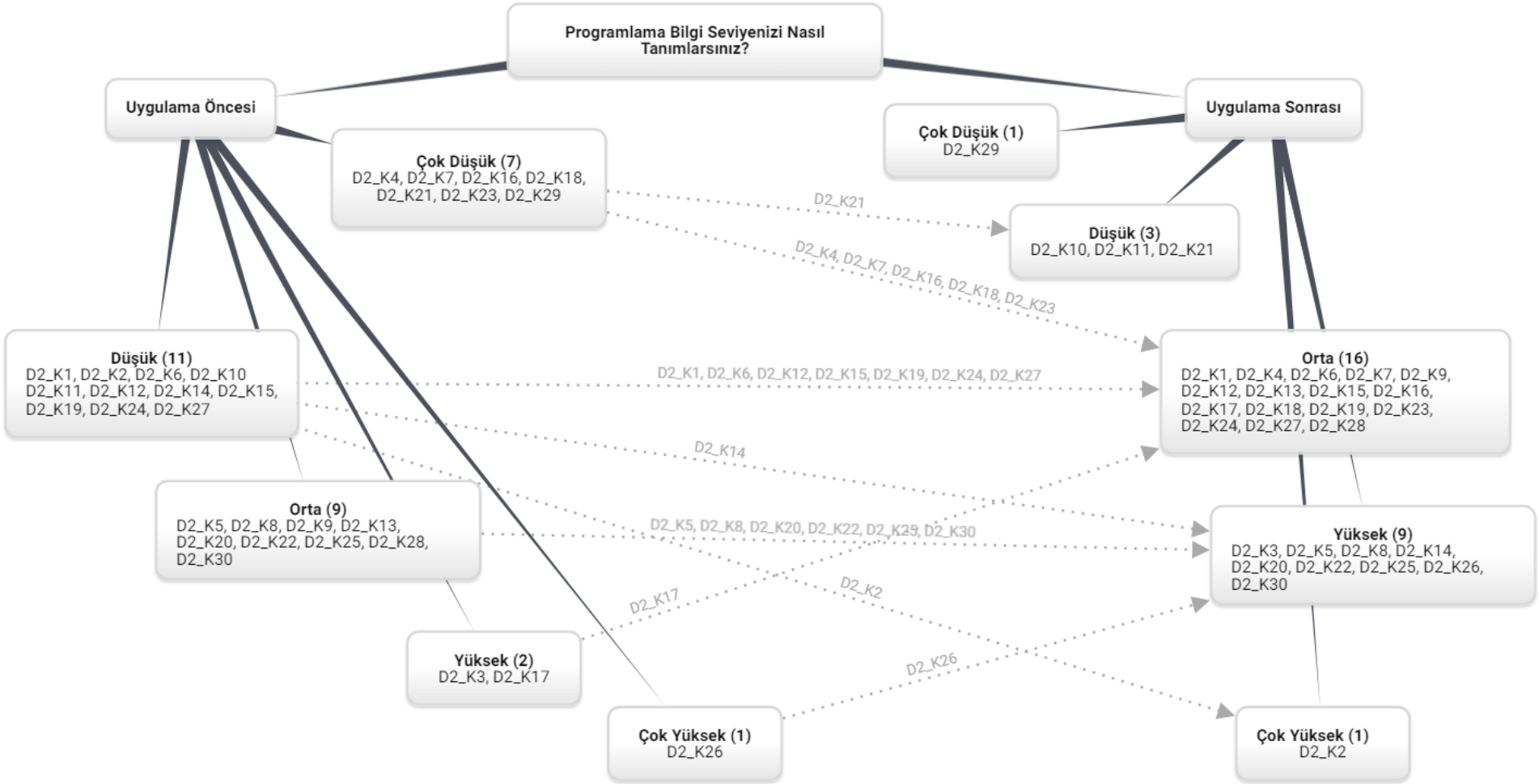
Katılımcıların algıladıkları programlama bilgi seviyesindeki değişimler katılımcı bazında da incelenmiştir (Şekil 4.57). Şekle göre katılımcıların %76'sının (N:23) algıladıkları programlama bilgisi seviyesinde değişiklik yaşanmıştır. İlave olarak uygulama sonrasında, uygulama öncesine göre kendilerini çok düşük ve düşük seviye olarak nitelendiren katılımcıların

sayısı önemli ölçüde azalırken kendilerini orta ve yüksek seviye olarak nitelendiren katılımcıların sayısı önemli derecede artmıştır. Algılanan programlama seviyesinde değişim yaşayan 23 katılımcı değerlendirildiğinde, değişimin en çok düşük seviyeden orta seviyeye gerçekleştiği söylenebilir (N:7, %23). İki katılımcının algıladığı programlama seviyesinde düşüş yaşanmıştır. Bu düşüş bir katılımcı için çok yüksek seviyeden yükseğe; diğer bir katılımcı için ise yüksek seviyeden orta seviyeye şeklindedir.

Katılımcıların programlama bilgi seviyesindeki değişim durumunun özeti Tablo 4.50'de sunulmuştur. Tabloya göre yedi kategoride değişim yaşanmış olup en fazla değişim düşük seviyeden orta seviyeye (N:7) ve orta seviyeden yüksek seviye (N:6) yönünde olmuştur. İlave olarak, katılımcıların %67'sinin algıladıkları programlama bilgi seviyesi artmış; %7'sinin azalmış ve %26'sının değişmemiştir (Tablo 4.51).

Tablo 4.50. Katılımcıların algıladıkları programlama bilgi seviyesindeki değişimin yönü

Değişimin Yönü		**Frekans (f)**
Öntest	**Sontest**	
Düşük	Orta	7
Orta	Yüksek	6
Çok Düşük	Orta	5
Çok Düşük	Düşük	1
Düşük	Yüksek	1
Yüksek	Orta	1
Çok Yüksek	Yüksek	1

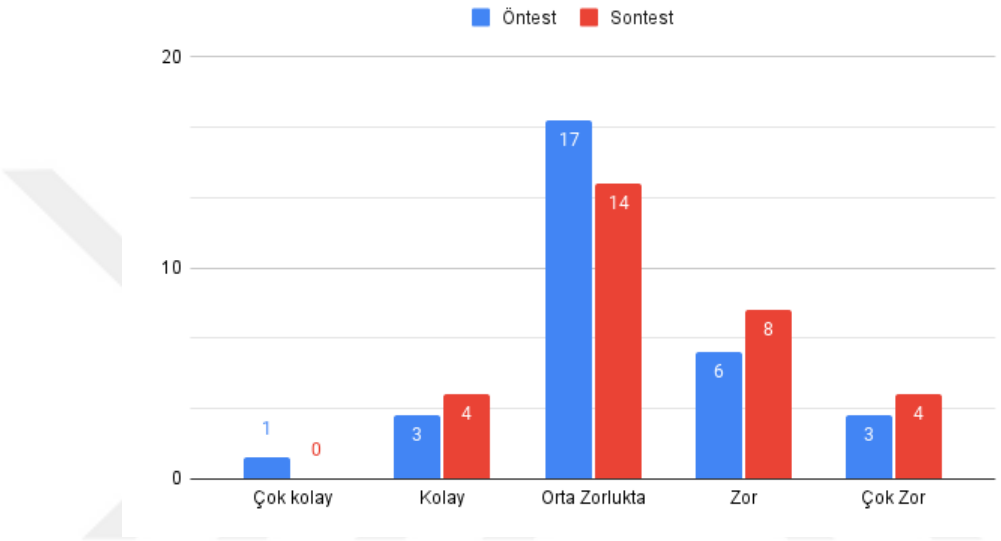


Şekil 4.57. Katılımcıların uygulama öncesi ve uygulama sonrasında algıladıkları programlama bilgi seviyesindeki değişim

Tablo 4.51. Katılımcıların algıladıkları programlama bilgi seviyesindeki değişim durumu

Değişim Durumu	Frekans (f)	Yüzde (%)
Algılanan Programlama Bilgi Seviyesi Artan	20	67
Algılanan Programlama Bilgi Seviyesi Azalan	2	7
Algılanan Programlama Bilgi Seviyesi Değişmeyen	8	26

Algılanan programlama bilgi seviyesindeki değişimlere ek olarak katılımcıların algıladıkları programlama zorluk seviyesindeki değişimler de incelenmiştir. Uygulama öncesi ve sonrası algılanan zorluk seviyesi değişim grafiği Şekil 4.58'de sunulmuştur.



Şekil 4.58. Katılımcıların algıladıkları zorluk seviyesindeki değişim grafiği

Şekle göre algıladıkları zorluk seviyesi "çok kolay" ve "orta zorlukta" kategorisinde olan katılımcıların sayısı azalırken, diğer kategorilerde artmıştır. Dolayısıyla katılımcıların algıladıkları programlama zorluk seviyesinin arttığı söylenebilir. Nitekim algılanan zorluk seviyesinin 5'li likert tipinde değerlendirildiği hesaba katıldığında; uygulama öncesinde 3.21 olarak ölçülen bu değerin uygulama sonrasında 3.38 olarak ölçülmesi katılımcıların algıladıkları programlama zorluk seviyesinin arttığını teyit etmektedir. Benzer şekilde, katılımcıların %37'sinin algıladıkları programlama zorluk seviyesi artarken %16'sının azalması, bu durumu doğrular niteliktedir (Tablo 4.52). Son olarak, katılımcıların %47'sinin algıladıkları programlama zorluk seviyesinde herhangi bir değişimin yaşanmadığı ortaya çıkmıştır. (Tablo 4.52).

Tablo 4.52. Katılımcıların algıladıkları programlama zorluk seviyesindeki değişim durumu

Değişim Durumu	Frekans (f)	Yüzde (%)
Algılanan Programlama Zorluk Seviyesi Artan	11	37
Algılanan Programlama Zorluk Seviyesi Azalan	5	16
Algılanan Programlama Zorluk Seviyesi Değişmeyen	14	47

NYP Başarı Sınavı, ölçek verileri ve katılımcıların algıladıkları programlama bilgi seviyeleri incelendikten sonra uygulanan eylem planının geçmiş programlama derslerindeki başarı durumu açısından düşük ve yüksek düzeyde olan öğrenciler üzerinde etkisini incelemek maksadıyla bazı analizler yapılmıştır. İlk olarak, düşük ve yüksek düzeyde olan katılımcıların NYP Başarı Sınavı sontest puanları karşılaştırılmıştır (Tablo 4.53). Test sonuçlarına göre geçmiş programlama derslerindeki başarı durumu açısından düşük ve yüksek düzeyde olan öğrencilerin sontest başarı puanları arasında anlamlı bir ilişki bulunmaktadır U=5.000, p = .041. Başka bir ifade ile geçmiş dönemlerdeki programlama derslerinde başarılı olup olmamak, sontest başarı puanını anlamlı olarak etkilemektedir.

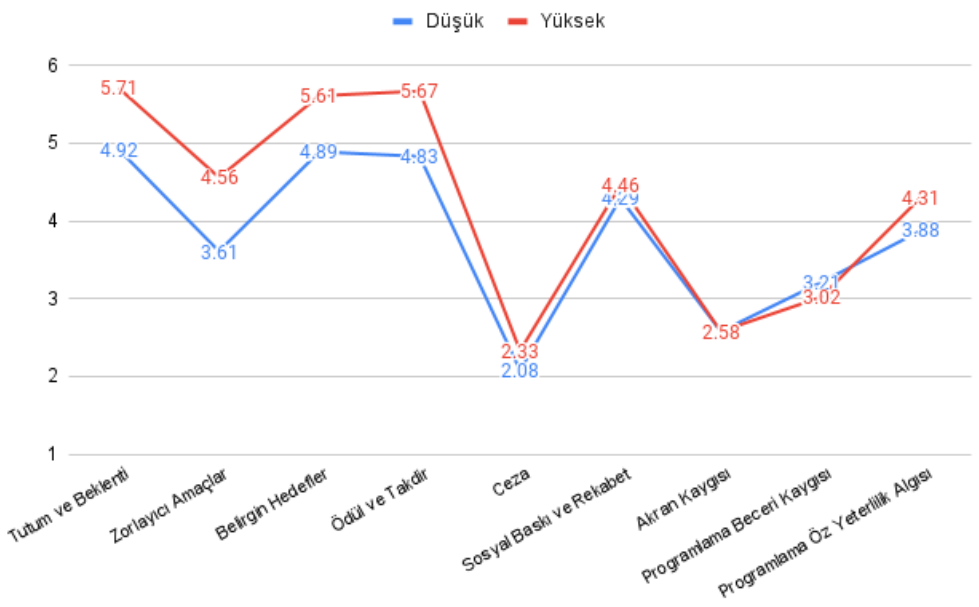
Tablo 4.53. Mann Whitney U testi sonuçları

Grup	N	Sıra $\bar{x}$	Sıra $\sum$	U	z	p
Düşük	6	4.33	26.00	5.000	-2.09	.041
Yüksek	6	8.67	52.00			

NYP Başarı Sınavına ilave olarak geçmiş programlama derslerindeki başarı durumu açısından düşük ve yüksek düzeyde olan öğrencilerin programlama dersine karşı motivasyonları, programlama kaygıları ve programlamaya yönelik öz yeterlilik algıları karşılaştırılmıştır. Ölçeklere ait betimsel istatistikler Tablo 4.54'de verilmiştir. Buna ilave olarak önceki programlama derslerindeki başarı durumu açısından düşük ve yüksek başarılı olan öğrencilerin Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeğinin altı alt faktörü, Programlama Kaygı Ölçeğinin iki alt faktörü ile Programlama Öz Yeterlilik Ölçeği puan ortalamalarının karşılaştırmaları Şekil 4.59'da verilmiştir.

Tablo 4.54. Düşük ve yüksek düzeyde başarılı olan öğrencilerin ölçek verilerine ilişkin betimsel istatistik sonuçları

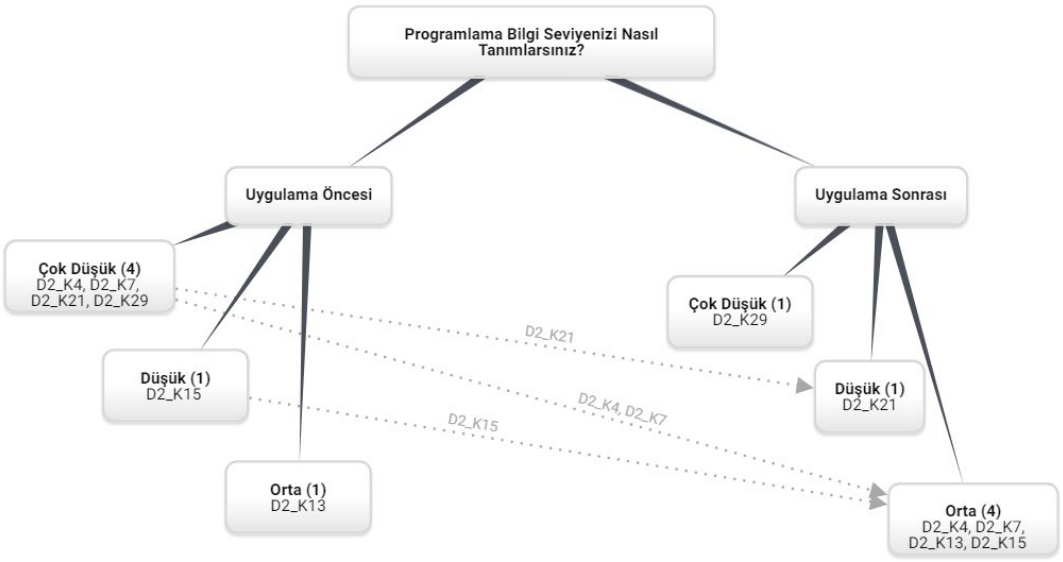
Ölçek	Grup	N	Min.	Maks.	$\bar{x}$	ss
Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeği	Düşük	6	3.74	4.79	4.26	.42
	Yüksek	6	4.26	5.53	4.89	.53
Programlama Kaygı Ölçeği	Düşük	6	2	3.93	2.94	.82
	Yüksek	6	2.14	3.43	2.83	.58
Programlama Öz Yeterlilik Ölçeği	Düşük	6	3.13	4.52	3.88	.59
	Yüksek	6	3.94	4.74	4.3	.28



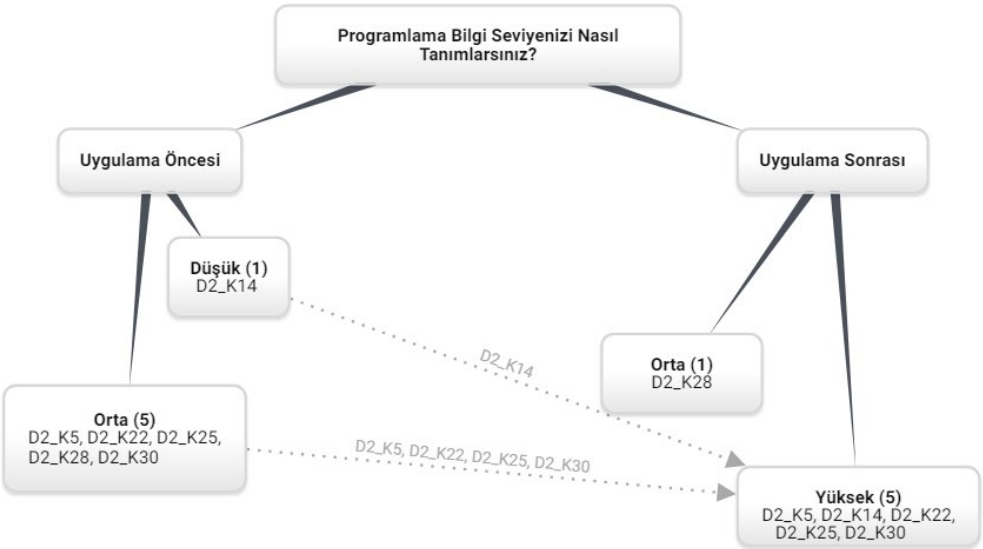
Şekil 4.59. Düşük ve yüksek düzeyde başarılı olan öğrencilerin ölçek alt faktör ortalamalarının karşılaştırma grafiği

Şekil 4.59 incelendiğinde, yüksek başarılı öğrencilerin Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeğinin altı alt faktörünün tamamında düşük başarılı öğrencilere göre daha yüksek puan ortalamalarına sahip oldukları görülmektedir. Benzer durum Programlama Öz Yeterlilik Ölçeği puan ortalamaları için de geçerlidir. Buna karşın, düşük başarılı öğrencileri yüksek Programlama Kaygı Ölçeğine ait programlama beceri kaygıları yüksek başarılı öğrencilere göre daha yüksek çıkarken akran kaygıları eşit çıkmıştır.

Ölçek verilerinin analizine ilave olarak, önceki programlama derslerindeki başarı durumu açısından düşük ve yüksek başarılı öğrencilerin uygulama öncesi ve sonrasında algıladıkları programlama bilgi seviyesindeki değişim de incelenmiştir. Bu kapsamda düşük başarılı öğrencilerin uygulama öncesinde ve sonrasında algıladıkları programlama bilgi seviyesi çok düşük, düşük ve orta seviyede gerçekleşmiştir (Şekil 4.60). Uygulama sonrasında bir öğrencinin algıladığı programlama bilgi seviyesinde değişiklik yaşanmış; bu değişiklik de "çok düşük" seviyesinden "düşük" seviyeye doğru gerçekleşmiştir. Bundan farklı olarak yüksek başarılı öğrencilerin uygulama öncesi ve sonrasında algıladıkları programlama bilgi seviyeleri "orta" ve "yüksek" seviyede gerçekleşmiştir (Şekil 4.61). Uygulama sonrasında bir yüksek başarılı öğrencinin algıladığı programlama bilgi seviyesinde değişiklik yaşanmış; bu değişiklik de "orta" seviyesinden "yüksek" seviyeye doğru gerçekleşmiştir.



Şekil 4.60. Düşük başarılı öğrencilerin uygulama öncesi ve uygulama sonrasında algıladıkları programlama bilgi seviyesindeki değişim



Şekil 4.61. Yüksek başarılı öğrencilerin uygulama öncesi ve uygulama sonrasında algıladıkları programlama bilgi seviyesindeki değişim

NYP Başarı Sınavı analizlerine ilave olarak, katılımcıların uygulama öncesi ve sonrasındaki Unity 3D bilgi seviyelerindeki değişim Unity 3D Başarı Sınavı kullanılarak

belirlenmiştir. Katılımcıların Unity 3D Başarı Sınavı öntest ve sontest puanlarına yönelik betimsel istatistikler Tablo 4.55'te sunulmuştur.

Tablo 4.55. NYP Başarı Sınavı öntest ve sontest puanlarına yönelik betimsel istatistikler

	Öntest			Sontest		
	N	**$\bar{x}$**	**ss**	**$\bar{x}$**	**ss**	**Fark**
Unity 3D Başarı Sınavı	30	4.4	2.7	93.2	7.31	88.8

Buna ilave olarak katılımcıların öntest ve sontest puanları istatistiksel olarak karşılaştırılmıştır. Bu karşılaştırmayı gerçekleştirmek maksadıyla öncelikle test sonuçlarının normal dağılıp dağılmadığı kontrol edilmiştir. Shapiro-Wilk testi sonuçlarına göre katılımcıların öntest ve sontest puanları normal dağılmamaktadır ($p<.05$). Bu sebeple katılımcıların Unity 3D Başarı Sınavı öntest ve sontest puanları karşılaştırmak amacıyla Wilcoxon sıralı işaretler testi kullanılmıştır (Pallant, 2020). Test sonuçları Tablo 4.56'da verilmiştir. Tablo 4.56'ya göre Unity 3D Başarı Sınavı için gerçekleştirilen Wilcoxon sıralı işaretler testi sonuçları, katılımcıların uygulamaya katıldıktan sonra başarı puanlarında büyük bir etki büyüklüğü ile ($r=.83$) istatistiksel olarak anlamlı değişiklikler olduğunu ortaya koymuştur ($z=-4.29$, $p=.000$). Bu sonuca göre oyun geliştirmeye dayalı tasarlanan ve gerçekleştirilen NYP öğretimi, öğrencilerin Unity 3D bilgilerini anlamlı bir şekilde etkilemektedir.

Tablo 4.56. Unity 3D Başarı Sınavı öntest-sontest puanlarının karşılaştırılmasına ilişkin Wilcoxon sıralı işaretler testi sonuçları

Puan	**Sıralar**	**N**	**Sıra $\bar{x}$**	**Sıra $\sum$**	**z**	**p**
NYP Başarı Sınavı	Negatif sıralar	0	0	0	-4.80	.00
	Pozitif sıralar	30	15.50	465.00		
	Eşit	0				
	Toplam	30				

4.2.3.2. Araştırma Sorusu 13: Eylem planının uygulanmasıyla birlikte katılımcıların psikolojik faktörlerinde (motivasyon, kaygı, öz yeterlilik algısı vb.) nasıl bir değişim meydana gelmiştir?

Uygulanan eylem planının katılımcıların programlama dersine yönelik motivasyon, öz yeterlilik algısı ve programlama kaygısı gibi psikolojik faktörlerini nasıl etkilediğine yönelik veriler öğrenci günlükleri, odak grup görüşmeleri, anketler ve ölçekler kullanılarak toplanmıştır. Bu bölümde katılımcılara ait psikolojik faktörlerdeki değişime yönelik bulgular sunulmuştur.

4.2.3.2.1. Programlamaya dersine yönelik motivasyon

Araştırmanın ikinci döngüsünde uygulanan eylem planının katılımcıların programlama dersine yönelik motivasyonları üzerindeki etkisini incelemek amacıyla nitel ve nicel veriler analiz edilmiştir. Nitel veri analizi sonucunda derste kullanılan yöntemin katılımcıların %90'ının (N:27) programlama dersine yönelik motivasyonunu olumlu yönde etkilediği, buna zıt olarak %10'unun motivasyonunu etkilemediği (N:3) görülmüştür. Gerçekleştirilen uygulamanın katılımcıların motivasyonlarını neden artırdığına ya da değiştirmediğine yönelik gerçekleştirilen çözümlemeler Tablo 4.57'de verilmiştir.

Tablo 4.57. Uygulamanın öğrencilerin programlama dersine yönelik motivasyonlarına etkilerine yönelik görüşleri*

Tema	Kategori	Kodlar	Frekans (f)	Yüzde (%)
Uygulamanın Programlama Motivasyonu Üzerindeki Etkilerinin Değerlendirilmesi	**Programlama Motivasyonunun Artmasının Nedenleri**	Bir oyun projesi geliştirmek ilgi çekici olduğu için motivasyonum arttı.	17	57
		Adım adım kendi oyunumu geliştirebilme imkânı elde etmek motivasyomu artırdı.	12	40
		Yazdığım kodların etkilerini proje üzerinde hemen görebilmek motivasyonumu artırdı.	8	27
		Dijital oyunları keyifle oynadığım için oyun projesi geliştirmek motivasyonumu artırdı.	7	23
		Kod yazmayı sevdirdiği için motivasyonum arttı.	4	13
	Programlama Motivasyonunun Değişmemesinin Nedenleri	Kod yazmaktan hoşlanmadığım için uygulama motivasyonumu etkilemedi.	3	10
		Programlama benim için zorlayıcı olduğu için uygulama motivasyonumu etkilemedi.	2	7
		Sıklıkla hatalarla karşılaştığım için uygulama motivasyonumu etkilemedi.	2	7
		Bilgisayarım olmadığı için uygulama yapamadım; bu nedenle motivasyonum etkilenmedi.	2	7

Uygulamanın katılımcıların programlama dersine yönelik motivasyonlarını artırmasının ilk nedeni, NYP dersinde yazılım geliştirme etkinliği olarak bir oyun geliştirmenin *keyifli ve ilgi çekici* olmasıyla ilgilidir. Bazı katılımcıların bu konu hakkındaki düşünceleri şu şekildedir:

"Bu yöntem diğer yöntemlere göre daha zevkli, eğlenceli, teşvik edici oluyor, daha az sıkıcı. Oyun üzerinde uğraşmak ilgi çekici insanın üzerinde çalışası geliyor." (D2_K25).

* Katılımcı sayısı: 30. Bir katılımcının vermiş olduğu cevap birden fazla kod içerebilir.

"Programlamayı oyun programlayarak öğrenmenin faydası oldu. Çünkü başka bir proje yapsaydık bu kadar öğrenme hevesimiz olmazdı. Oyun gibi ilgi çekici konular üzerinden gitmek daha başarılı olmamızı sağladı." (D2_K5).

Hatta D2_K3 ve D2_K8 kodlu katılımcıların kendi Karting Microgame oyun projelerinin yanında örnek ekran alıntıları Şekil 4.62 ve Şekil 4.63'de sunulan başka bir oyun projesi de geliştirmeleri, oyun geliştirmenin bazı öğrenciler için keyifli bir programlama etkinliği olmasının ve bu durumun öğrencilerin motivasyonlarını artırmasının bir göstergesi olarak kabul edilebilir. Kendilerinden ders kapsamında istenmemesine ve değerlendirme kriterleri arasında yer almamasına rağmen bu iki katılımcının neden ilave bir oyun projesi geliştirdikleri sorgulandığında; katılımcıların bu tür projelerden keyif alarak uzun süre üzerinde çalışabilmeleri, oyun projelerine istedikleri özellikleri ekleyebilmek için sarf ettikleri gayretin sonuçlarını hem kendilerinin hem de başkalarının görebilmeleri, bu tür projelerin hem eğlenip hem de öğrenmelerine olanak tanımaları sayılabilir. D2_K8 kodlu katılımcı bu konu ile ilgili düşüncelerini şu cümlelerle ifade etmiştir:

"Aslında bu dersin en çok sevdiğim yönü proje geliştirmenin, kod yazmanın ve ürün oluşturmanın zevkli olmasıydı. İnsan kodlarını yazarken de, tasarımını yaparken de sıkılmıyor. Saatlerce hiç ara vermeden çalıştığım oldu. Dahası hem zevk aldım hem öğrendim. Ben de bu nedenlerle internet üzerinden bir oyun prototipi bularak derste yaptığımız oyuna benzer bir proje oluşturdum. Derste öğrendiklerimi diğer projemde uyguladım. Zorlandığım olmadı değil. Ancak araştırarak öğrenerek bu sorunları aştım. Her iki çalışmamda da ilerlemek beni mutlu etti. Ve sonunda iki projem oldu. Bu ders hayatıma bir yenilik kattı açıkçası. Bu oyunları yaptım. Bundan sonra Unity'yi hiç kullanmadım demem. Dahası başka bir yerde gördüğümde buna dair bir fikrimin olduğunu belirtebilirim ve ileriki dönemlerde eğer gerekirse veya istersem bu tarz farklı oyunlar da yapabilirim."

Şekil 4.62. D2_K3 kodlu katılımcının ilave olarak geliştirdiği oyun projesinin ekran alıntısı

Şekil 4.63. D2_K8 kodlu katılımcının ilave olarak geliştirdiği oyun projesinin ekran alıntısı

Katılımcıların kendi oyunlarını adım adım geliştirmeleri, oyunlarına her geçen hafta yeni eklentiler üretmeleri katılımcıların motivasyonlarını artıran bir diğer unsur olmuştur. Bazı katılımcıların bu konu hakkındaki düşünceleri şu şekildedir:

"Oyun sayesinde kodlama daha çekici bir hale geldi ve bu kodlamaların sonunda elimizde tamamen kendi düşüncelerimizle elde ettiğimiz bir oyun olması, kodlamayı bize daha çekici kıldı." (D2_K5).

"Çalışan bir oyun üzerine eklenti yapmayı oyunun üzerine yeni kodlar yazmayı oyuna yeni eklentiler ve görseller eklemek oyunun geliştiğini görmek beni programlamayı daha iyi anlamak için motive etti." (D2_K18).

"Kod yazdıkça projede bir şeylerin değiştiğini görmek çok motive edici geliyor bana." (D2_K10).

Katılımcıların programlama motivasyonlarını etkileyen başka bir unsur ise yazdıkları kodların etkilerini bizzat oyun projesi üzerinde deneyimlemek olmuştur. Bu sayede katılımcılar yazdıkları kodların oyun projelerini nasıl etkilediğini gözlemleme şansı elde etmişlerdir. Bazı katılımcıların bu konu hakkındaki düşünceleri şu şekildedir:

"Programlamada yaptığımız değişiklikleri [oyun üzerinde yapılan değişiklikleri] bizzat görmek [oyun üzerinde görmek] işin daha da motive edici olmasını sağladı." (D2_K1).

"Oyun yapmamız bizim biraz dâhil, ilgili olmamızı sağladı. Ve internetten de araştırmamızı sağladı. Başaralı sonuçları elde ettikçe yazdığım kodların uygulamadaki etkisini gördükçe merakım ve hevesim arttı." (D2_K23).

Katılımcıların programlamaya yönelik motivasyonlarını artıran diğer unsurlar ise zaten dijital oyunlardan hoşlanmaları ya da bu uygulamanın kod yazmaktan hoşlanmayan öğrencilere kod yazmayı sevdirmesiyle ilgilidir. Bazı katılımcıların bu konu hakkındaki düşünceleri şu şekildedir:

"Zaten oyun oynamayı severdim, bu da dikkatimi daha da çekti ve kendine yöneltti." (D2_K30).

"Açıkçası yazılım geliştirme ve kod yazmayı çok sevmezdim. Artık sevmeye başladım. Derste kullandığımız bu yöntem buna sebep oldu." (D2_K4).

"Eskiden kod yazmayı sevmezdim yani okuldayken ama şimdi kod yazdıkça oyunda bir şeylerin değiştiğini bir şeylerin hareketlendiğini görmek çok motive ediyor artık daha sıcak bakıyorum." (D2_K9).

Uygulanan eylem planı bazı katılımcıların programlamaya yönelik motivasyonlarını olumlu yönde etkilerken, bazı katılımcıların (N:3) programlama dersine yönelik motivasyonlarını değiştirmemiştir. Bu durumun ortaya çıkmasındaki ilk neden katılımcıların kod yazmaktan hoşlanmamalarıyla ilgilidir. Uygulamanın programlama dersine yönelik motivasyonlarını etkilemediğini belirten katılımcıların her üçü de zaten kod yazmaktan hoşlanmadıklarını, programlama projelerine karşı ilgi duymadıklarını ve bu durumun oyun geliştirme projesi gerçekleştirmekle değişmediğini belirtmişlerdir. D2_K27 kodlu katılımcının bu konu hakkındaki düşünceleri şu şekildedir:

"Bun uygulama ilgimi pek değiştirmedi çünkü kod yazmayı pek sevmiyorum. Dersi almadan önce de sevmiyordum, hala da kod yazmaktan hoşlanmıyorum. Her ne kadar kod yazdıktan sonra görsel tasarım yapmamız biraz ilgimi çekse de programlama ve kodlar işin içine girince ilgim azalıyor."

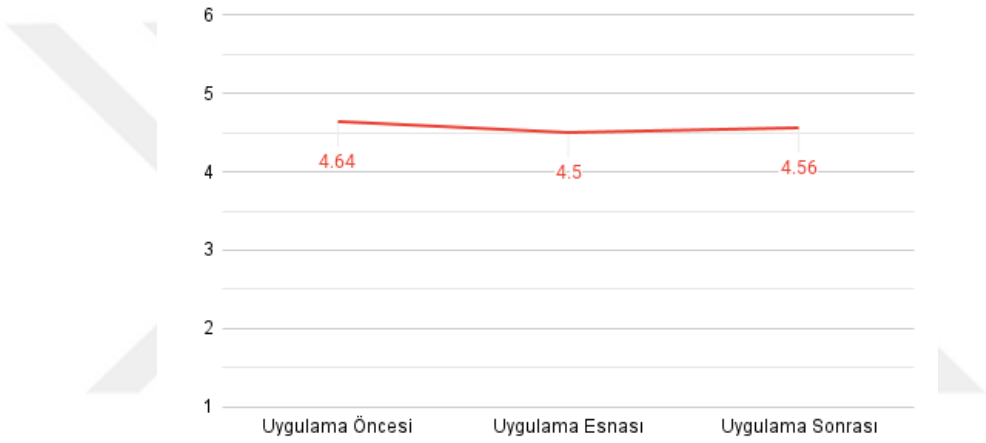
Katılımcıların programlama derslerine yönelik motivasyonlarının etkilenmemesinin bir diğer nedeni ise programlamanın zorlayıcı yapısından kaynaklanmaktadır. Karşılaşın hataları düzeltmekte yetersiz kalmak, bunun yanında bazı konuları anlamakta güçlük çekmek gibi sebepler katılımcıların motivasyonlarının değişmemesine neden olan unsurlardandır. D2_K27 kodlu katılımcının bu konu hakkındaki düşünceleri şu şekildedir:

"Bu uygulama ilgimi ve motivasyonumu değiştirmedi. Benim programlama altyapım yeterli değildi. Bazen izlediğim videolarda anlamadığım yerler oluyordu. Yapamadığım zaman moralim bozuluyordu. Derste WhatsApp'tan aldığım desteğin bana çok katkısı oldu. İnternetten kaynak araştırdım ancak bizim yaptığımız projeye benzemiyordu. Yazılım zor ve ezber gerektiriyor ve görsel olmadığı için beni zorluyor. En ufak bir yerde hata vermesi beni gerçekten sıkıntıya sokuyordu. Saatlerce takıldığım yerlerde oyalandığım oldu. Bir de bilgisayar sıkıntısı oldu, ben sonradan aldım. Tabi zaman kaybettim bir miktar. Bu yüzden zorlandım. Kısacası programlamayı hala sevmiyorum, bu yüzden motivasyonum az."

İki katılımcı ise kişisel bilgisayarları olmadığı için projelerini tamamlayamadıklarını, uygulama yapmamaları nedeniyle programlama dersine karşı olan motivasyon ve ilgilerinin değişmediğini belirtmişlerdir. D2_K11 kodlu katılımcı düşüncelerini şu cümlelerle ifade etmiştir:

"Bu ders programlamaya karşı olan motivasyon ve ilgimi değiştirmedi. Bilgisayarım yoktu biliyorsunuz. Sadece canlı dersleri takip edip ders videolarını izledim. Ondan dolayı bir proje oluşturamadım. Programlamaya karşı olan ilgim ve motivasyonum değişmedi bu yüzden."

Nitel analizlere ilave olarak katılımcıların programlama dersine yönelik motivasyonları, uygulama öncesinde, uygulama esnasında ve uygulama sonrasında olmak üzere üç farklı zamanda Programlama Derslerine Yönelik Motivasyon Ölçeği kullanılarak ölçülmüş ve katılımcıların motivasyonlarındaki değişim incelenmiştir. Katılımcıların toplam motivasyon puan ortalamalarının değişim grafiği Şekil 4.64'te sunulmuştur. Buna ek olarak katılımcıların Programlama Derslerine Yönelik Motivasyon Ölçeğinden zaman 1 (uygulama öncesi), zaman 2 (uygulama esnası) ve zaman 3 (uygulama sonrası) için elde edilen puanları kıyaslamak için tek faktörlü tekrarlanan ölçümler ANOVA testi yürütülmüştür. Test sonucuna göre zaman için istatistiksel olarak anlamlı bir sonuç bulunmamıştır, Wilks's Lambda= .952, F(2, 28)=.704, p>.05.



Şekil 4.64. Katılımcıların programlama dersine yönelik motivasyon puanları ortalaması değişim grafiği

Toplam programlama dersine motivasyon puanlarına ilave olarak, Programlama Derslerine Yönelik Motivasyon Ölçeğinin altı alt faktörünün zaman 1 (uygulama öncesi), zaman 2 (uygulama esnası) ve zaman 3 (uygulama sonrası) puanları tek faktörlü tekrarlanan ölçümler ANOVA testi ile kıyaslanmıştır. Test sonuçları Tablo 4.58'de verilmiştir. Tablo 4.58'e göre *zorlayıcı amaçlar* alt faktöründe zaman için istatistiksel olarak anlamlı bir sonuç bulunmuştur. *Zorlayıcı amaçlar* alt faktörüne ait Bonferroni post hoc testi sonuçlarına göre katılımcıların puan ortalamaları uygulama öncesi ($\bar{x}$ = 4.77; ss =.90) ile uygulama sonrası için anlamlı bir şekilde değişmektedir ($\bar{x}$= 4.28; ss= 1.14; p= .029).

Tablo 4.58. Tek faktörlü tekrarlanan ölçümler ANOVA testi sonuçları

Faktör	N	Wilks's Lambda	F	p
Tutum ve Beklenti	30	.978	.309	>.05
Zorlayıcı amaçlar	30	.697	6.076	**<.05**
Belirgin hedefler	30	.916	1.285	>.05
Ödül ve Takdir	30	.975	.357	>.05
Ceza	30	.92	1.223	>.05
Sosyal Baskı ve Rekabet	30	.99	.141	>.05

4.2.3.2.2. Programlamaya yönelik öz yeterlilik algısı

Uygulanan eylem planının katılımcıların programlamaya yönelik öz yeterlilik algıları üzerindeki etkisini incelemek amacıyla toplanan nitel ve nicel veriler analiz edilmiştir. Nitel veri analizi sonucunda derste kullanılan yöntemin katılımcıların %93'ünün (N:28) programlamaya yönelik öz yeterlilik algısını olumlu yönde etkilediği, buna zıt olarak bir katılımcının (%3) programlama öz yeterlilik algısını olumsuz olarak etkilediği, bir katılımcının (%3) ise programlama öz yeterlilik algısını etkilemediği sonucuna varılmıştır. Gerçekleştirilen uygulamanın katılımcıların programlama öz yeterlilik algılarını neden artırdığına ya da değiştirmediğine yönelik gerçekleştirilen çözümlemeler Tablo 4.59'da verilmiştir.

Tablo 4.59. Uygulamanın öğrencilerin programlama öz yeterlilik algısına etkilerine yönelik görüşleri*

Tema	Kategori	Kodlar	Frekans (f)	Yüzde (%)
Uygulamanın Programlama Öz yeterlilik Algısı Üzerindeki Etkilerinin Değerlendirilmesi	**Programlama Öz yeterlilik Algısının Gelişmesinin Nedenleri**	Oyun geliştirmeyi başarabildiğimi görmek programlama konusunda kendime güvenimi artırdı.	22	77
		Karşılaştığım hataları kendi başıma çözebileceğimi fark ettim.	5	17
		Sanılanın aksine programlamanın başarılması çok zor olmadığını fark ettim.	4	13
	Programlama Öz yeterlilik Algısının Değişmemesinin Nedenleri	Programlama benim için hala zor bir alan, o yüzden programlama öz yeterlilik algım değişmedi.	1	3
	Programlama Öz yeterlilik Algısının Azalmasının Nedenleri	Programlama konusunda başarılı olamayacağım düşüncesine kapılmama neden olduğu için öz yeterlilik algım azaldı.	1	3
		Programlama konusunda arkadaşlarımın seviyesine ulaşamayacağım düşüncesine kapılmama neden olduğu için öz yeterlilik algım azaldı.	1	3

* Katılımcı sayısı: 30. Bir katılımcının vermiş olduğu cevap birden fazla kod içerebilir.

Programlama öz yeterlilik algıları olumlu yönde etkilenen katılımcılar, kullanılan yöntemin bir oyunu kendi hayal güçleri doğrultusunda geliştirebilecekleri konusunda kendilerine güven sağladığını ve programlamayı başarabildiklerini hissettiklerini ifade etmişlerdir. Bazı katılımcıların bu konu hakkındaki düşünceleri şu şekildedir:

"Hali hazır bir oyuna kod, nesne ve efekt ekleyip daha iyi çalışır bir hale getirdim. Bu benim kodlama konusunda kendime güvenimi arttırdı. Oyun yapmanın herkesin hayali olduğunu düşünüyorum. Ufak da olsa bunu başardım." (D2_K18).

"Hem öğrendik hem bir şeyler yapabildiğimizi fark ettik. Oynadığımız oyunları kendimizin yapabildiğimizi görmek cesaretlendirici oldu. Kendimin böyle şeyleri yapabileceğini düşünmezdim." (D2_K17).

"Daha yeni öğrendiğimiz için elbette unuttuğumuz yerler olacak lakin sürekli olarak bir proje içerisinde olursak gözü kapalı oyun programlayacağımıza inanıyorum. İnsanın yaptıkça bir şeyleri başardıkça onu bırakası gelmiyor." (D2_K26).

"Kendime güvenimi olumlu yönde değiştirdiğini düşünüyorum çünkü bir şeyleri yapabildiğimi ve kurgulayabildiğimi gördüm." (D2_K21).

Yapılan veri analizi sonucuna göre katılımcıların programlamaya yönelik öz yeterlilik algılarını etkileyen faktörlerden birisi de üzerinde çalıştıkları projelerde ortaya çıkan hatalarla ilgilidir. Bazı katılımcıların projelerinde çıkan hataları düzelttikçe kendilerine olan güvenlerinin arttığını şu cümlelerle aktarmışlardır:

"Kendime güvenimi aşırı derecede iyi etkiledi sebebine gelecek olursak bu süreçte karşılaştığım sorunlar beni zorladı ama bunu çözebilmem. Bu durum dersteki ve programlamadaki kendime güvenimi arttırdı, ondan ziyade böyle bir oyunun hazır assets de olsa benim katkılarım ve emeklerimle olması beni hem sevindirdi hem de daha iyisini yapabileceğim konusunda kendime bir özgüven geldi. Eğer ki ileriki dönemde bu tür bir oyun yapmak istersem buna en büyük katkıyı bu ders sağlamış olur." (D2_K8).

"Kendime güvenim arttı. Çünkü karşılaştığım sorunların birçoğunu kendim hallettim." (D2_K28).

Katılımcılardan bazıları dersten önce programlama derslerini zor olarak değerlendirdiklerini, ancak dersten sonra bu algılarının değiştiğini aktarmışlardır. Katılımcılar düşüncelerini şu cümlelerle ifade etmişlerdir:

"Kendimi daha iyi hissediyorum çünkü programlama ve oyun geliştirme sandığım kadar zor değilmiş." (D2_K15).

Belirli bir IQ seviyesinden üstte olan her insanın oyun yapabileceği kanaatine vardım. Karmaşık gibi görünen alt yapının aslında çok kolay ve basit bir mantığı var." (D2_K3).

"Kod yazmanın o kadar da zor olmadığını anladım." (D2_K6).

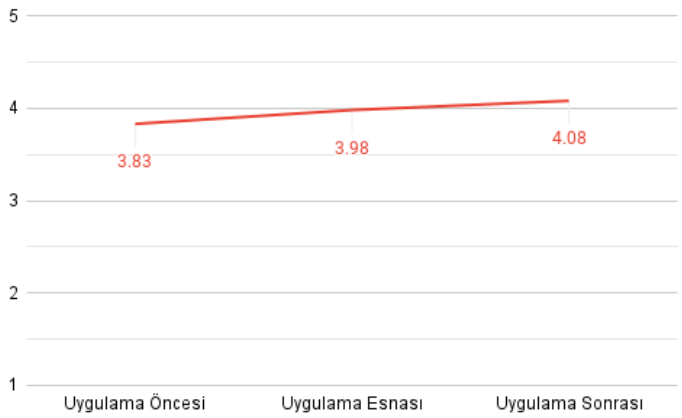
Uygulanan eylem planından olumlu yönde etkilenen katılımcıların yanında, bir katılımcının programlamaya yönelik öz yeterlilik algısı etkilenmemiştir. D2_K29 isimli katılımcı bunun sebebini şu cümlelerle ifade etmiştir:

"Programlama konusunda kendime güvenim değişmedi. Yazılım benim için hala zor. Algoritma kurmak, yazılım tasarlamak, oyun geliştirmek benim için oldukça güç. Yabancı dilim de kötü. Zaten altta yatan en büyük sorunlardan birisi de bu. Yani programlama konusunda kendime güvenimde herhangi bir değişiklik olmadı."

Yukarıda bahsedilen durumlara ilave olarak, bir katılımcının programlamaya yönelik öz yeterlilik algısının uygulanan eylem planından olumsuz yönde etkilendiği görülmüştür. Bu durumu yaratan temel neden, ilgili katılımcının bu süreçte programlamayı başaramayacağına, oyun projesini tamamlayamayacağına ve arkadaşlarının seviyesine ulaşamayacağına inanmasıdır. Bunun sebebini D2_K21 isimli şu cümlelerle açıklamıştır:

"Açıkçası sadece dersten derse programlama çalıştım. Öyle olunca çok başarılı olamadım. Karamsarlıkta kaldım. Canlı derste öğrendiğimi anladığımı sanıyordum. Sonrasında ise yapamayınca kendime güvenimi sorguluyordum. Günü gününe çalışmayınca ve programlamayı yapamayınca anlayamadığımı düşünüyordum. Arkadaşlarım gibi proje geliştirip dersi başarabileceğime inancım gün gün azaldı. Bu da beni karamsarlıkta bıraktı."

Nitel analizlere ilave olarak katılımcıların programlamaya yönelik öz yeterlilik algıları, uygulama öncesinde, uygulama esnasında ve uygulama sonrasında olmak üzere üç farklı zamanda Programlama Öz yeterlilik Algısı Ölçeği kullanılarak ölçülmüş ve katılımcıların öz yeterlilik algılarındaki değişimler incelenmiştir. Katılımcıların programlama öz yeterlilik puan ortalamalarının değişim grafiği Şekil 4.65'de sunulmuştur. Buna ek olarak Programlama Öz yeterlilik Algısı Ölçeğinden zaman 1 (uygulama öncesi), zaman 2 (uygulama esnası) ve zaman 3 (uygulama sonrası) için elde edilen puanları kıyaslamak için tek faktörlü tekrarlanan ölçümler ANOVA testi yürütülmüştür. Test sonucunda zaman için istatistiksel olarak anlamlı bir sonuç bulunmuştur, Wilks's Lambda =.793, $F(2, 28)=3.658$, $p<.05$. Bonferroni post hoc testi sonuçlarına göre katılımcıların programlama öz yeterlilik algısı puan ortalamaları uygulama öncesi ($\bar{x}$ = 3.83; ss= .53) ile uygulama sonrası için anlamlı bir şekilde değişmektedir ($\bar{x}$ = 4.08; ss = .63; p= .035).



Şekil 4.65. Katılımcıların programlama öz yeterlilik puan ortalamaları değişim grafiği

4.2.3.2.3. Programlama kaygısı

Araştırmanın ikinci döngüsünde uygulanan eylem planının katılımcıların programlama kaygısı üzerindeki etkisini incelemek amacıyla nitel ve nicel veriler analiz edilmiştir. Toplanan verilerin analizi sonucunda katılımcıların %63'ünün (N:19) uygulama esnasında programlama kaygısının arttığı ancak zamanla azaldığı, %37'sinin (N:11) ise programlama kaygısının değişmediği belirlenmiştir. Buna ilave olarak her ne kadar programlama kaygıları süreç içinde azalsa da, katılımcıların %57'si (N:17) hala programlama konusunda kaygılarının olduğunu belirtmişlerdir. Gerçekleştirilen uygulamanın katılımcıların programlama kaygısını neden artırdığı ya da değiştirmediğine yönelik gerçekleştirilen çözümlemeler Tablo 4.60'da verilmiştir.

Uygulamanın başlarında katılımcıların programlama kaygısının artmasının nedenleri olarak katılımcıların bireysel programlama yeteneklerine güvenmemesi, daha önce bir oyun geliştirme deneyimlerinin olmaması ve karşılaştıkları hataları giderme konusunda duydukları endişeler sayılabilir. Bazı katılımcıların bu konu hakkındaki düşünceleri şu şekildedir:

"Dönemin başlarında dersi geçmek, projeyi bitirmekle ilgili korkularım vardı. Ya bunları başaramazsam diye içimden geçiriyordum. Zamanla bir şeyleri yapabileceğimi gördüm ve bu konu kod yazamayacağım endişesini azalttı diyebilirim." (D2_K21).

Bilgisayar bölümü okuyan birisi bile o kodları, programları görünce içimde yapamam, öğrenemem korkusu oluyordur. Bende oluyordu. Bu dersi aldıktan sonra korkularım zamanla azaldı. Artık çok karışık oyunlar, temalı programlar kurabilmemiz için bu ders bize bir referans oldu. Özgüvenimize bilgilerimize çok pozitif yönde katkı sağladı." (D2_K17).

"Başta bir hata çıktığında nasıl çözeceğim hakkında endişelerim vardı. Fakat bunların hepsini yendim. Özellikle kod yazma konusundaki korkularımı ve önyargılarımı yenmiş oldum. Artık korkmadan ve sıkılmadan kod yazabiliyorum." (D2_K4).

Tablo 4.60. Uygulamanın öğrencilerin programlama kaygısına etkilerine yönelik görüşleri*

Tema	Kategori	Kodlar	Frekans (f)	Yüzde (%)
Uygulamanın Programlama Kaygısı Üzerindeki Etkilerinin Değerlendirilmesi	**Programlama Kaygısının Artmasının Nedenleri**	Uygulamanın başlarında programlama yeteneklerime çok fazla güvenemediğim için kaygım vardı.	13	43
		Daha önce bir oyun geliştirme deneyimim olmadığı için uygulamanın başlarında endişelerim vardı.	12	40
		Uygulamanın başlarında hatalarla karşılaşınca nasıl çözeceğim konusunda endişelerim vardı.	7	23
	Programlama Kaygısının Zamanla Azalmasının Nedenleri	Programlama yeteneklerime daha fazla güvenmeye başladığım için kaygılarım azaldı.	15	50
		Karşılaştığım hatalar konusunda öğretmen ve arkadaşlarımın desteği kaygılarımı azalttı.	5	17
		Ders kaynakları yeterince açıklama içerdiği için kaygılarım azaldı.	2	7
	Programlama Kaygısının Devam Etmesinin Nedenleri	Programlamanın yapısı gereği zor olmasından dolayı kaygılarım devam ediyor.	8	27
		Programlama sınavlarının zor olmasından dolayı endişelerim devam ediyor.	7	23
		Programlama derslerinde mutlaka bir değerlendirmeye tabi tutulacağımı bildiğim için kaygım devam ediyor.	5	17
		Farklı yazılım projelerinde programlamayla ilgili yeni şeyler öğrenmek zorunda kalacağım için kaygım devam ediyor.	4	13
		Hatalarla karşılaşmakla ilgili hala endişelerim bulunuyor.	4	13

Katılımcılar, programlama kaygılarının azalmasının etmenlerini programlama konusunda kendilerine güvenlerinin artması, karşılaşılan hatalar konusunda arkadaş ve öğretmen desteği ile ders kaynaklarının yeterince açıklama içermesi olarak belirtmişlerdir. Bazı katılımcıların bu konu hakkındaki düşünceleri şu şekildedir:

"Açıkçası ilk başlarda nasıl yapacağım hakkında bir korkum vardı ama ders sadece canlı derslerle kalmayıp videolarla ve yardımlarla desteklendi. Hocamız takıldığımız yerlerde cevaplamalar yaptı bu da benim korkularımı yenmemi sağladı. Artık bu konuda hiç bir sorunum yok." (D2_K8).

"Başlarda uygulamayı [dönem sonu projesini] yapamayacağım için biraz endişelendim ama arkadaşlarımın yardımı ve videolarla hallettim." (D2_K18).

"Aslında bazı kalıpları öğrendikten sonra çok da korkutucu ve zor olmadığını öğrendim." (D2_K19).

Bazı katılımcılar uygulanan yöntemin programlama kaygılarını azalttığını ancak programlama kaygılarının devam ettiğini belirtmişlerdir. Bunun sebebini programlamanın

* Katılımcı sayısı: 30. Bir katılımcının vermiş olduğu cevap birden fazla kod içerebilir.

doğası gereği zor olmasına, kod yazma esnasında sıklıkla hatalarla karşılaşmalarına, farklı yazılım projelerinde farklı araç ve teknolojileri öğrenmeleri gerekeceğine ve programlama derslerinde mutlaka bir değerlendirmeye (sınav, ödev, quiz vb.) tabi tutulacaklarını bilmelerine bağlamışlardır. κatılımcılar bu konu hakkındaki düşüncelerini şu cümlelerle aktarmışlardır:

"Programlama çok fazla yapıdan, komuttan ve kod satırından meydana gelmekte. Bu durum nesne yönelimli programlamaya yeni başlayan birisi için kafa karıştırıcı bir şey. Daha sonradan öğrendikçe ve nesne yönelimli programlamanın mantığını anlayınca basitleşiyor. Genellikle ben programlamayı anlamıyorum. Ona karşı bir önyargım var, bir korkum var. Ancak çalıştıkça yapabiliyorum. Şuan programlamaya karşı hala bir kaygım var, aldığımız Nesne Yönelimli Programlama dersinden sonra aldığımız yani şuanda almakta olduğumuz yeni programlama derslerinde bunu hissediyorum. Ama başardıkça bu kaygımın azalacağına inanıyorum. Her programlama dersinde bunları yaşadığım için, her programlama dersinde üzerime yeni bir sorumluluk yüklendiği için kaygılarımı değiştirmedi. Programlama yine bir uğraş gerektirecek, yine bir çalışma gerektirecek, belki yapacağım belki yapamayacağım. Nerede hata verecek, hatamı bulabilecek miyim? Bu gibi düşüncelerden dolayı kaygımı olumlu ya da olumsuz etkilemedi diyebilirim. " (D2_K11).

"Endişelerimin bir kısmını yendim ama kodlamada hata yapıp ne de olsa bir öğrenimde bulunduğumuz için not olarak değerlendirileceğimi düşününce istemsiz korku ve endişeye sebep olabiliyor." (D2_K12).

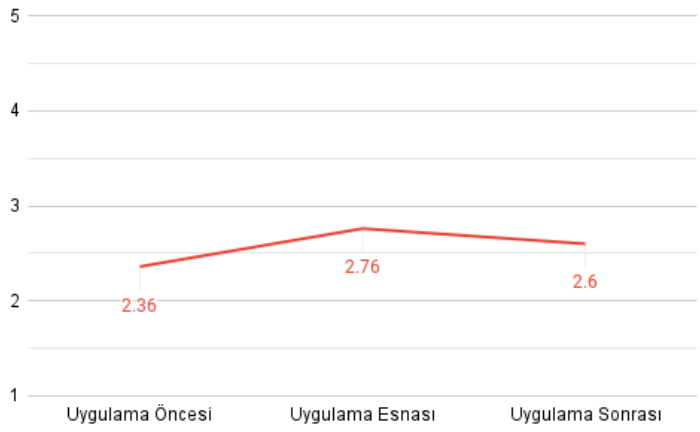
"Programlama kaygım hep vardı. Derste uyguladığı yöntem evet azalttı ama hala o kaygı var. Çünkü program yazmak çok meşakkatli bir iş." (D2_K7).

"En başta çok endişeliydim fakat belirli bir aşamaya geldikten sonra benim için eğlenceli bir hale dönüştü. Endişelerim ortadan kalktı ama hala kodlama için farklı bir şekilde karşımıza geldiğinde, farklı tür projeler geliştirdiğimde korkularım olacaktır. Çünkü yeni projelerde yeni teknolojiler kullanmak zorunda kalacağım. Örneğin sanal gerçeklik tabanlı bir oyun geliştirsem bununla ilgili pek çok yeni şeyle karşılaşacağım. Yeni bileşenler, hatalar gibi. Bu durum beni endişelendirecektir. " (D2_K13).

"Bu uygulamanın sevmediğim yönü hata yapıp hatamı göremediğimde, problemimi çözemediğimde sorun yaratması. Hevesimi kırması ve bunu yapamayınca iyi bir programcı olup olmayacağım hakkımda şahsi endişelerim." (D2_K12).

Nitel analizlere ilave olarak katılımcıların programlama kaygıları, Programlama Kaygı Ölçeği kullanılarak uygulama öncesinde, uygulama esnasında ve uygulama sonrasında olmak üzere üç farklı zamanda ölçülmüş ve katılımcıların programlama kaygılarındaki değişimler incelenmiştir. . Katılımcıların programlama kaygısı puan ortalamalarının değişim grafiği Şekil 4.66'da sunulmuştur. Buna ek olarak Programlama Kaygı Ölçeğinden Zaman 1 (uygulama öncesi), Zaman 2 (uygulama esnası) ve Zaman 3 (uygulama sonrası) için elde edilen puanları kıyaslamak için tek faktörlü tekrarlanan ölçümler ANOVA testi yürütülmüştür. Test sonucunda zaman için istatistiksel olarak anlamlı bir sonuç bulunmuştur, Wilks's Lambda=.700, $F(2, 28)= 5.998$, $p<.05$. Bonferroni post hoc testi

sonuçlarına göre katılımcıların programlama kaygısı toplam puan ortalamaları uygulama öncesi ($\bar{x}$ = 2.36; ss= .80) ile uygulama esnası için anlamlı bir şekilde değişmektedir ($\bar{x}$= 2.76; ss= .90; p= .006).



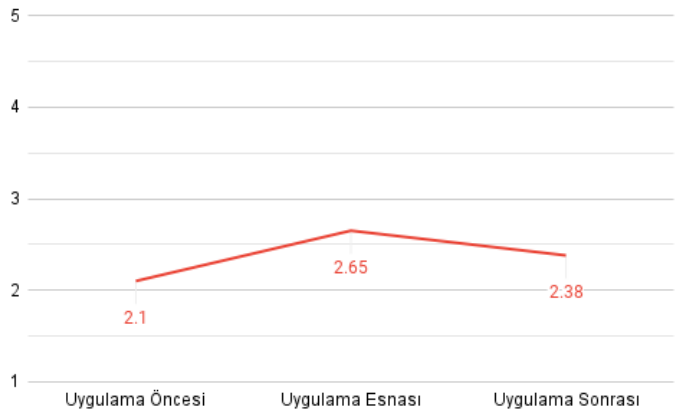
Şekil 4.66. Katılımcıların programlama kaygısı puan ortalamaları değişim grafiği

Toplam programlama kaygısı puanlarına ilave olarak, Programlama Kaygı Ölçeğinin akran ve programlama beceri kaygısı alt faktörünün zaman 1 (uygulama öncesi), zaman 2 (uygulama esnası) ve zaman 3 (uygulama sonrası) puanları tek faktörlü tekrarlanan ölçümler ANOVA testi ile kıyaslanmıştır. Test sonuçları Tablo 4.61'de verilmiştir. Tabloya göre *akran kaygısı* alt faktöründe zaman için istatistiksel olarak anlamlı bir sonuç bulunmuştur. Bu alt faktörüne ait Bonferroni post hoc testi sonuçlarına göre katılımcıların puan ortalamaları uygulama öncesi ($\bar{x}$= 2.10; ss =.91) ile uygulama esnası için anlamlı bir şekilde değişmektedir ($\bar{x}$= 2.65; ss= 1.05; p= .001).

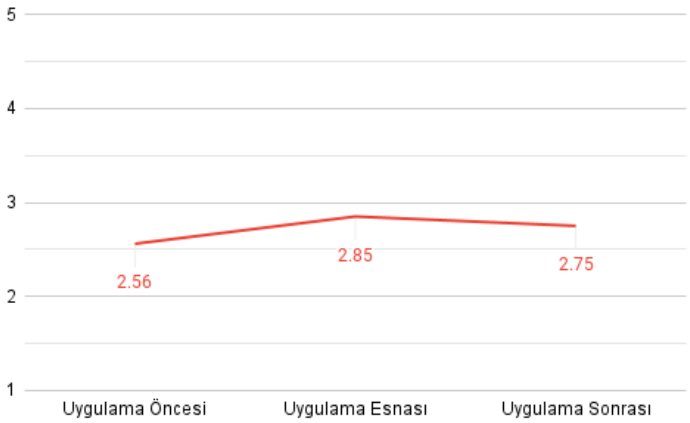
Tablo 4.61. Tek faktörlü tekrarlanan ölçümler ANOVA testi sonuçları

Faktör	N	Wilks's Lambda	F	p
Akran Kaygısı	30	.608	9.008	<.05
Programlama Beceri Kaygısı	30	.861	2.253	>.05

Tüm bu analizlere ek olarak, katılımcıların Programlama Kaygı Ölçeğinin alt faktör ortalama puanları değişim grafikleri Şekil 4.67 ve Şekil 4.68'de sunulmuştur.



Şekil 4.67. Akran kaygısı alt faktörü ortalama puanları değişim grafiği



Şekil 4.68. Programlama beceri kaygısı alt faktörü ortalama puanları değişim grafiği

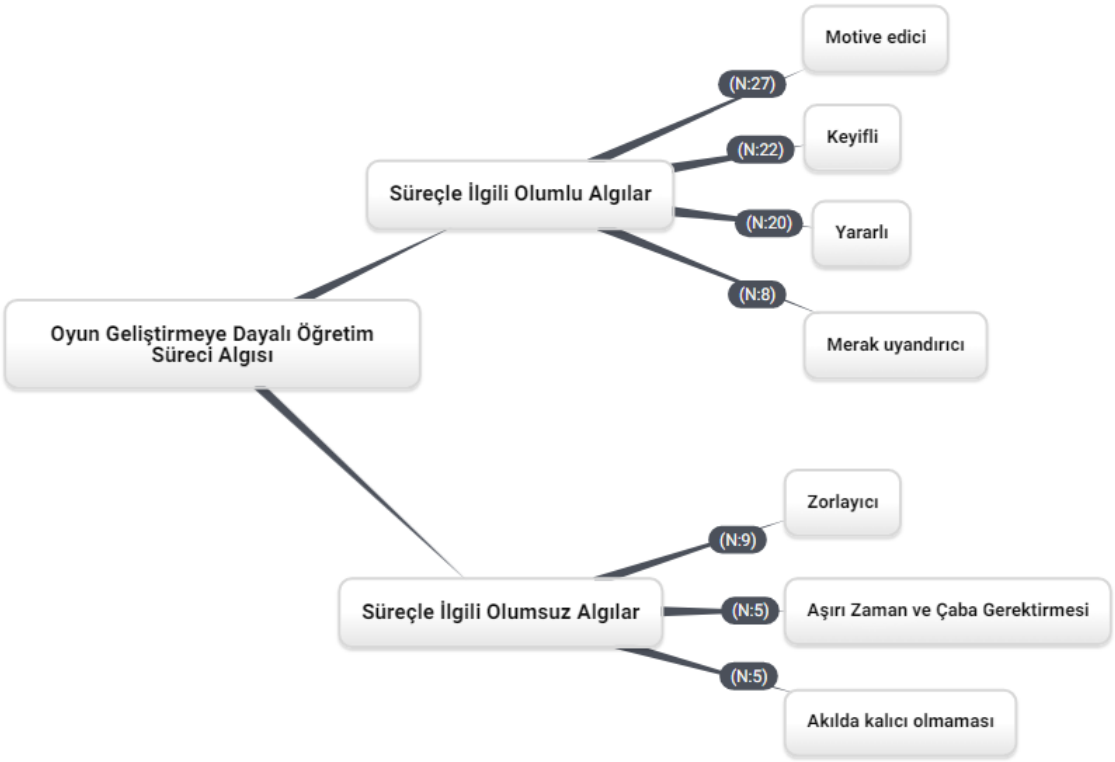
4.2.3.3. Araştırma Sorusu 14: Geliştirilen NYP dersine yönelik katılımcıların görüşleri nelerdir?

Katılımcıların ikinci döngüde gerçekleştirilen NYP öğretim sürecine yönelik görüşleri; (1) katılımcıların oyun geliştirmeye dayalı öğretim süreci algıları, (2) uzaktan öğretim süreci algıları, (3) öğretim sürecinin katılımcıların bireysel ve mesleki gelişimlerine katkıları,(4) katılımcıların ders materyalleri hakkındaki görüşleri ve (5) katılımcıların uygulamanın geliştirilmesine yönelik önerileri bağlamında çözümlenerek alt maddelerde sunulmuştur.

4.2.3.3.1. Katılımcıların oyun geliştirmeye dayalı öğretim süreci algıları

Katılımcıların mevcut bir oyuna eklentiler yapmaya dayalı öğretim sürecine yönelik algıları olumlu ve olumsuz olmak üzere ikiye ayrılmıştır (Şekil 4.69). Şekilde görüldü üzere katılımcılar olumlu algıları *keyifli, yararlı, motive edici ve merak uyandırıcı* olarak

nitelendirilirken olumsuz algıları *zorlayıcı, aşırı zaman ve çaba gerektirmesi* ve *akılda kalıcı olmaması* şeklindedir.



Şekil 4.69. Katılımcıların oyun geliştirmeye dayalı öğretim süreciyle ilgili algıları

Öğretim sürecinin keyifli ve yararlı olduğunu belirten katılımcılar düşüncelerini şu cümlelerle ifade etmişlerdir:

"Zevkli bir eğitim süreciydi benim için. Zaten özel hayatımda da resim çizmeyi, tasarım yapmayı sevdiğim için, oyun tasarlama kısmı benim için baya baya zevkliydi." (D2_K15).

"Özellikle o oyun tasarlama zamanları gayet güzeldi benim için. Eğitim de genel olarak gayet verimli ve yeterliydi. Yorucu oldu ama sıkıcı değildi. Eğlenerek öğrenme diyebilirim. Bundan sonrası tamamen bizim kendi çabamıza bağlı. Eğer istersek kendimizi geliştirebiliriz bu aldığımız eğitimin üzerine." (D2_K2).

"Eğitim genel olarak iyiydi. Sıkılmadık, derslerden zevk aldık. İlgimizi çekti, oyun üzerinde çalıştığımız için kısacası güzel ve başarılıydı." (D2_K23).

"Programlamayı oyun programlayarak öğrenmek daha zevkli ve akıcı oldu. Öğrenmemi kolaylaştırdı. Programlamayı sevmeme ve anlamama yardımcı oldu." (D2_K14).

Katılımcılar programlamayı oyun programlayarak öğrenmenin kendilerini yeni şeyler öğrenmek ve denemek için motive ettiğini ve ilgilerini çektiğini belirtmişlerdir. Katılımcılar bir ürün elde etmenin, mevcut bir oyun prototipine kendi hayal güçleri doğrultusunda özellikler eklemenin kendilerinde başarmışlık hissi oluşturduğunu, bunun da nesne

yönelimli programlama üzerinde çalışmaları için motivasyon sağladığını belirtmişlerdir. Katılımcılar bu konu hakkındaki düşüncelerini şu cümlelerle ifade etmişlerdir:

"Direkt kod yazarak bir şeyler yapmaktansa bize hitap eden bir oyun yapmak ve bunu kodlamak bizleri daha çok motive etti ve öğrenme isteği oluşturdu. Sonunda daha iyi bir oyun oluşturmak istediğimiz için başarma isteğimiz arttı." (D2_K5).

"Programlamayı oyun yaparak öğrenmek zevkli olduğu için daha çok yapma isteği ve sürekli projeyle ilgilenme isteği oluşturdu. Oyun yaparak hem daha rahat hem de daha kolay öğrendik. Yaptığım projenin kodlarının oyunun içinde uygulandığını görmek benim ilgimi artırdı ve oyunu güzelleştirmek için daha çok çalıştım." (D2_K13).

"İlk başlarda yazdığımız bu kodlar benim için pek bir şey fark etmiyordu ama oyuna dönüşünce yazdığımız kodlar insanı daha çok öğrenme ye ve çalışmaya yöneltiyor." (D2_K6).

"Bu haftayla birlikte Unity ile neler yapacağımı görünce Unity kullanmaya hevesim artmaya başladı. Yapabileceklerimizi gördükçe heyecanlanıyorum. İleriki haftalarda hevesimin artacağını düşünüyorum."((D2_K26, Dördüncü hafta günlüğü).

"Programlamayı oyun programlayarak öğrenmek beni daha çok derse çekti. Kendi tasarladığım oyunla vakit geçirmek onu geliştirmek benim ilgilimi daha çok çekiyordu. Başardığım bir şeyleri görünce mutlu olup daha da üst seviyelere çıkmak istiyordum." (D2_K8).

"Bu dersi sevdim çünkü düz anlatmak yerine oyun ve renkli olunca bizi daha çok derse çekti." (D2_K10).

Katılımcıların öğretim sürecine yönelik algılarını olumlu olarak etkileyen bir diğer etmen ise proje geliştirme sürecinde sıfırdan oyun oluşturmak yerine mevcut bir oyuna eklentiler yapma yoluyla programlamayı NYP paradigmasını öğrenmek olmuştur. Katılımcılardan sekizi derste çalışan bir oyun prototipinin kullanılmasından dolayı oyun projelerinin detaylarıyla fazla uğraşmak zorunda kalmadıklarını, sadece gerçekleştirilmesi gereken işleme (göreve) yöneldiklerini belirtmişlerdir. Katılımcılar bu konu hakkındaki düşüncelerini şu cümlelerle ifade etmişlerdir:

"Benim için dersi kolaylaştıran ve öğrenmemi hızlandıran şeylerden birisi de derste oyun prototipi kullanmak oldu. Gördüğümüz prototip üzerinde çalışmak, yeniden bir oyun geliştirmeye göre çok daha kolay. Üzerinde istediğimiz eklentileri yapıp oyunu daha kolay ve hızlı şekillendirebildik. Bu sayede dersin konularına da yoğunlaşma fırsatı bulduk. " (D2_K17).

"Böyle bir prototip üzerinde çalışmak daha motive edici oldu. Bu sayede sadece belli başlı şeylerle uğraştık. Diğer türlü sıfırdan oyun yapmak için birçok konuya el atmak lazım gelirdi, bu da çok sıkıcı olurdu. Ayrıca hazır oyunun benim sayemde daha çok geliştiğini, daha stabil, daha iyi bir hale büründüğünü görmek güzel hissettirdi." (D2_K25).

Katılımcıların ders uygulaması hakkındaki olumsuz algıları ise NYP ve oyun programlamanın zor olmasından, aşırı zaman ve çaba gerektirmesinden, bunun yanında

akılda kalıcı olmamasından kaynaklanmaktadır. Katılımcılar bu konu hakkındaki görüşlerini şu cümlelerle aktarmışlardır:

"Oyun içindeki sesler ve oyun akışı içinde puan yazdırmayı derste gayet iyi öğrendim fakat bunları dersten sonra denediğimde başaramadım videoları birkaç kere izleyerek bu sorunu çözeceğimi düşünüyorum." (D2_K13, yedinci hafta günlüğü).

"Dersler öğretici fakat karışık mesela o gün öğrendiğim bir şeyi 2-3 gün sonra tekrar yapmak istediğimde komutanımızın attığı videoları izleyerek yani yardım alarak yapmak zorunda kalıyorum fakat bu benim için bir sorun değil." (D2_K10, altıncı hafta günlüğü).

"Haftalar geçtikçe anlamakta zorlanıyorum ama anlamadığım kısımları gerek attığınız videolar olsun gerekse Youtube'dan başka videolar olsun destekleyerek anlamaya çalışacağım." (D2_K26, beşinci hafta günlüğü).

"Ders eğlenceli ama ilerledikçe karıştırabiliyorum sınıfları." (D2_K14, altıncı hafta günlüğü).

"Bu hafta görmüş olduğumuz dersleri tekrar ettim ve program üzerinde denedim fakat hala videoları izlemeden proje hazırlayamıyorum. Biraz daha tekrar yaptıktan ve videoları izledikten sonra projeyi yardımsız tekrar yapmaya çalışacağım." (D2_K23, on birinci hafta günlüğü).

Katılımcılardan beşi dersin başlarında hem NYP paradigmasını hem de Unity 3D editörünü kullanmayı aynı anda öğrenmenin kendileri açısından güçlük yarattığını; ancak zaman içinde Unity editörünü kullanma becerileri geliştikçe bu durumun sorun olmaktan çıktığını belirtmişlerdir. Katılımcılar bu konu hakkındaki görüşlerini şu cümlelerle aktarmışlardır:

"Başlarda Unity'de çalışmakta zorlanıyordum. Bana çok karışık bir program gibi geldi, sadece hocamızın paylaştığı videoları saniye saniye izleyip aynısını yapabiliyordum. Hatta sadece nesneleri ve sınıfları oluşturuyordum, bunun ilerisindeki uygulamaların beni zorlayacağını düşünüyordum. Hangi nesneye ne tür kodları ekleyeceğim konusunda kafam karışıyordu. Zaman içinde temelini öğrendim. Artık Unity zor gelmiyor bana. Ancak programlama hala zorluk içeriyor." (D2_K13).

"Unity'yi öğrenmek biraz zaman aldı benim için. Açıkçası başlarda oyunla birlikte oluşturduğumuz dosyalar arka planda karışıklık yaratıyordu. Örneğin çok nesne vardı ve kodları birbirinden bağımsız birçok nesneye yazmak durumunda kalıyordunuz. Geliştirdiğimiz oyunun nesne ve sınıflarını da kavramak gerekiyordu. Bu yüzden dersin ilk haftalarında kapsamlı olarak öğrenemedim, ama idare etmeye çalıştım. Sonrasında daha iyi seviyeye geldim. Unity'yi bilseydim sanırım projemde çok daha hızlı yol alırdım ve daha az zorlanırdım." (D2_K9).

4.2.3.3.2. Katılımcıların uzaktan öğretim süreci algıları

Araştırmanın ikinci döngüsü acil uzaktan öğretim yöntemi kullanılarak tamamlanmıştır. Uzaktan öğretim sürecinin katılımcıların motivasyonlarına etkisini inceleyebilmek

maksadıyla odak grup görüşmesi gerçekleştirilmiştir. Nitel veri analizi sonucunda dersin uzaktan işlenmesinin katılımcıların %60'ının (N:18) programlama dersine yönelik motivasyonunu olumlu yönde etkilediği, %33'ünün (N:10) olumsuz etkilediği ve %7'sinin (N:2) motivasyonunu etkilemediği tespit edilmiştir. Gerçekleştirilen uygulamanın katılımcıların motivasyonlarını neden artırdığına ya da değiştirmediğine yönelik gerçekleştirilen çözümlemeler Tablo 4.62'de verilmiştir.

Tablo 4.62. Uzaktan öğretimin öğrencilerin programlama dersine yönelik motivasyonlarına etkilerine yönelik görüşleri*

Tema	Kategori	Kodlar	Frekans (f)	Yüzde (%)
Uygulamanın Programlama Motivasyonu Üzerindeki Etkilerinin Değerlendirilmesi	Motivasyon Artışının Nedenleri	İstediğim zamanda çalışabilme imkânı motivasyonumu artırdı.	12	40
		Evde kendimi rahat hissetmem motivasyonumu artırdı.	8	27
		Proje ve ödevlerin geçme notuna etkisi uzaktan öğretimde daha yüksek olduğu için motivasyonum arttı.	3	10
	Motivasyon Düşmesinin Nedenleri	Anlık olarak iletişim kurmada sorunlar yaşadığımın için motivasyonum düştü.	6	20
		Uzaktan öğretim derse yönelik konsantrasyonumu olumsuz etkilediği için motivasyonum düştü.	3	10
		Ders işliyormuş gibi hissedemediğim için motivasyonum düştü.	2	7
		Yüz yüze öğretime göre derse daha fazla zaman ayırmak zorunda kaldığım için motivasyonum düştü.	1	3
	Motivasyonu Etkilememesinin Nedenleri	Anlatılan içeriklerin aynı olması nedeniyle dersin uzaktan işlenmesi motivasyonumu etkilemedi.	2	7

Uzaktan öğretim sürecinin katılımcıların programlama dersine yönelik motivasyonlarını artırmasının ilk nedeni, dersin uzaktan işlenmesinin sağladığı zaman esnekliğiyle ilgilidir. Öğrenciler tercih ettikleri zaman aralığında ve istedikleri sıklıkla derse çalışabilme imkânı elde etmenin motivasyonları üzerinde olumlu etkilerinin olduğunu belirtmişlerdir. Bazı katılımcıların bu konu hakkındaki düşünceleri şu şekildedir:

Uzaktan eğitim süreci benim motivasyonumu artırdı. Uzaktan eğitim esnasında programlama dillerini daha iyi benimsediğimi söyleyebilirim. Uzaktan eğitimde hem internet yönünden hem de bilgisayarın başına oturma yönünden bir kısıtlama yok. İstediğin zaman başına oturup istediğin zaman kalkabiliyorsun. Bir sorunla karşılaşınca internette derin araştırma yapılabiliyorsun. Uzaktan eğitimde gece çalışmalarım yoğunlaştı; bu yüzden fazlasıyla motive oldum diyebilirim. (D2_K5).

Uzaktan öğretim süreci motivasyonumu artırdı. Zaman zaman derse vaktinde katılamasam da sonradan izleyebildiğimden kaçırdığım konu olmadı. Bazen konuları anlamak için tekrar tekrar izleme şansım da oldu.

* Katılımcı sayısı: 30. Bir katılımcının vermiş olduğu cevap birden fazla kod içerebilir.

Bu sayede ödevleri ve projemi düzgün yapabildim. Bu da motivasyonumun yüksek olmasını sağladı. (D2_K20).

Uzaktan öğretim sürecinin programlama dersine yönelik motivasyonlarını artırmasının bir diğer nedeni, bir bilgisayar dershanesine nazaran katılımcıların ev ortamında kendilerini rahat hissederek kendi bilgisayarlarıyla çalışma imkânı elde etmeleriyle ilgilidir. İlave olarak öğrencilerin yüz yüze derslerdeki gibi diğer arkadaşlarına yetişme gibi bir kaygılarının olmayışı, motivasyonlarının artmasına neden olmuştur. D2_K12 kodlu katılımcının bu konu hakkındaki düşünceleri şu şekildedir:

Kendi evimde, kendi bilgisayarımda çalışmak bana huzurlu hissettirdi. Çünkü evde rahat bir şekilde çalışabildim. Sınıftan geride kalma korkusu daha az olduğu için üzerimde baskı da hissetmedim. Bu da beni motive etti.

Uzaktan öğretim sürecinin öğrencilerin motivasyonlarını artırmasının son nedeni, uzaktan öğretim sürecinde proje ve ödevlerin geçme notuna etkisinin daha fazla olmasıyla ilgilidir. Her ne kadar araştırmanın birinci ve ikinci döngüsünde kullanılan değerlendirme sistemi değişmemiş olsa da, katılımcıların meslek yüksekokulu birinci sınıfı öğretim sürecinden alışkın oldukları değerlendirme sistemine nazaran uzaktan öğretimde ödev ve proje notlarının ağırlığının artması, üç katılımcının motivasyonlarının artmasına neden olmuştur. D2_K3 kodlu katılımcının bu konu hakkındaki düşünceleri şu şekildedir:

Geçme notunda ödev ve projelerin payının yüksek olması beni rahatlattı. Daha keyif alarak öğrenebilmek beni mutlu etti. Bu da motivasyonumu artıran nedenlerdendir.

Motivasyon artışına zıt olarak uzaktan öğretim süreci 10 katılımcının motivasyonlarını olumsuz etkilemiştir. Bunun ilk nedeni uzaktan öğretimde etkileşimin sınırlı olmasından kaynaklanmıştır. Her ne kadar karşılaşılan kodlama hatalarının paylaşımı ve çözümüne yönelik bir WhatsApp grubu kurulmuş olsa da, bazı katılımcılar yüz yüze öğretiminde hatalar ve anlaşılamayan noktalarla ilgili daha etkili sonuçlar aldıklarını belirtmişlerdir. D2_K15 kodlu katılımcının bu konu hakkındaki düşünceleri şu şekildedir:

Yüz yüze derslerde mesela bir sorunla karşılaştığımızda hocamız anında yardımcı olabiliyordu; [bilgisayar dershanesindeki] bilgisayarımıza [öğretmen bilgisayarından] bağlanıp düzeltmeler ve tavsiyeler yapabiliyordu. Ama bu süreçte bu imkân olmadı. Geç saatlerde kimseyi rahatsız etmemek için [WhatsApp] gruba da mesaj yazamadım. Hataları çözmekte zorlandım, bazen bir sonraki güne bıraktım. Bu da beni rahatsız ediyordu. Kimi zaman hatayı çözemediğim için rüyalarıma giriyordu ve beni uykusuz bırakıyordu.

Katılımcıların motivasyonlarının olumsuz etkilenmesinin bir diğer nedeni ise uzaktan öğretim sürecinin bazı katılımcıların konsantrasyonlarını olumsuz etkilemesiyle ilgilidir. D2_K24 kodlu katılımcının bu konu hakkındaki düşünceleri şu şekildedir:

Dersi evde dinlemekte konsantre güçlüğü yaşadım. Çünkü evde kafamı dağıtabilecek birçok etken olduğundan odaklanma problemim oldu. Konsantre olup çalışmak gayret gerektirdi.

Katılımcıların motivasyonlarının olumsuz etkilenmesinin bir diğer nedeni ise canlı derslerin bazı katılımcılar tarafından etkisiz bulunmasıyla ilgilidir. Canlı derslerde hem öğretmeni uygulama yapmadan dinlemek hem de etkileşime girmekte zorlanmak iki katılımcıyı olumsuz etkilemiştir. D2_K26 kodlu katılımcının bu konu hakkındaki düşünceleri şu şekildedir:

Uzaktan öğretimde ders işliyormuş gibi hissedemedim. Derslere katılma şevkim gün geçtikçe azalıyordu. Ders içerisindeki gibi etkileşimi vermiyor, bu yüzden derse olan ilgim azalıyordu. Slayt okuma gibi oluyordu dersler. Hocamız ne kadar güzel ve etkileşimli anlatsa da yüz yüze dersteki gibi olamıyordu.

Uzaktan öğretim süreci iki katılımcının derse yönelik motivasyonunu etkilememiştir. Dersin konu içeriğinin sunum ortamından bağımsız olarak aynı olacağını düşünen bu katılımcılar, zaten NYP dersinin uygulama ağırlıklı olacağını ve ders zamanları haricinde kendilerinin uygulama yapmaları gerekeceği nedeniyle motivasyonlarının etkilenmediğini belirtmişlerdir. D2_K27 kodlu katılımcının bu konu hakkındaki düşünceleri şu şekildedir:

Dersin uzaktan eğitimde olması motivasyonumu etkilemedi. Anlatılan içerikler zaten yüz yüze ders olsaydı da aynı olacaktı. Programlama dersleri uygulamalı öğrenilmesi gerek dersler. Yüz yüze de olsa kodları kâğıda yazınca hatam oldu mu olmadı mı kâğıt bana söylemiyor. Mutlaka uygulama yapılması gerekiyor. Ancak derste uygulama için sınırlı süre var. Öyle olunca mutlaka ders dışı çalışmak gerekiyor.

4.2.3.3.3. Öğretim sürecinin katılımcıların bireysel ve mesleki gelişimlerine katkıları

Oyun geliştirmeye dayalı öğretim sürecinin katılımcıların bireysel ve mesleki gelişimlerine katkıları Şekil 4.70'de sunulmuştur. Bu katkılar "*teknik ve mesleki yeterliliklere katkılar"* ve *"istihdam olanağı"* olmak üzere ikiye ayrılmıştır. Teknik ve mesleki yeterliliklere katkıları değerlendirildiğinde; öğretim sürecinin katılımcıların NYP paradigmasını öğrenmelerine, oyun programlama deneyimi kazanmalarına ve projelerinde karşılaştıkları hataları düzeltme becerilerinin gelişmesine olanak tanıdığı görülmektedir. Katılımcılar bu konu hakkındaki düşüncelerini şu cümlelerle ifade etmişlerdir:

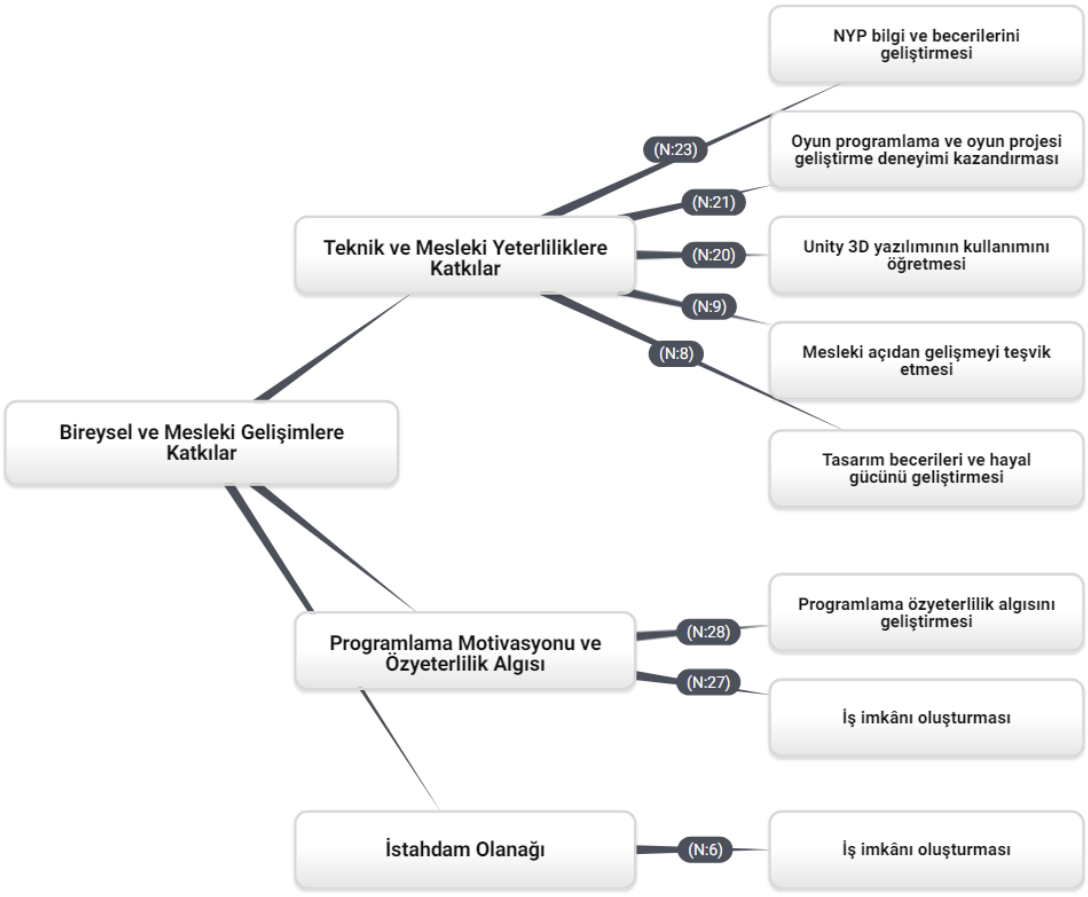
"Kaliteli bir eğitimdi. Nesne Yönelimli Programlama kavramlarını anlamama ve kullanmama yardımcı oldu. Oyun üretmeyi, nesnelere özellik kazandırmayı ve oyuna C# kodlarını bağlamayı öğrendim. Sınıflar oluşturarak C#'ta NYP uygulamaları yapmayı pekiştirdim. Programlamayı hızlandırdım ve akıcı geldi" (D2_K14).

"Programlamayı oyun programlayarak öğrenmek aslında çok güzel oldu. Oyun yapmak istediğim bir şeydi, bunu nasıl yapacağımı bilmiyordum. Okulda görsel programlama dersinde öğrendiğimiz C# kodlarını nasıl oyunlarda kullanabileceğim konusunda bilgim yoktu. Şimdi bu derste hem NYP tasarımını hem de oyun

programlamayı öğrenmek beni her yönden olumlu etkiledi. Ayrıca daha çok uğraş, daha çok öğretici video, yazılım ve kod gerektirdi, bu da bizim bilgilerimizi artırdı.” (D2_K8).

“Şu anda günümüz neslinin en popüler aktivitesi olan bilgisayar oyunları hakkında bir bilgi sahibi olmak, en azından nasıl yapıldığını, yaparken hangi süreçlerden geçildiğini bilmek bir bilgisayar bölümü öğrencisi olarak benim için önemli bir tecrübe idi.” (D2_K1).

“Dersin bana katkısı şekilde olmuştur: Kodlama yardımı ile nesne yönelimli programlama yapabilmeyi ve bunu yaparken karşılaştığım sorunları kendi başıma çözmeyi öğrendim. Sonuçta hepimiz hatalarla karşılaştık ve bunları çözmek için daha çok kodlarla içli dışlı olduk. Daha çok yazıp denemek zorunda kaldık. Hatalarla karşılaştık ve çözümler ürettik. Bu da öğretici oldu.” (D2_K2).



Şekil 4.70. Katılımcıların oyun geliştirmeye dayalı öğretim sürecinin bireysel ve mesleki gelişimlerine katkılarına yönelik görüşleri

Katılımcılar, öğretim sürecinin sonunda NYP paradigmasıyla ilgili beceriler kazanmanın yanında Unity 3D editörünü kullanma becerisi de kazandıklarını belirtmişlerdir. Katılımcılar bu konu hakkındaki düşüncelerini şu cümlelerle ifade etmişlerdir:

“Açıkçası bu dersi almak benim hiç aklımda yoktu. Ama bunun olması çok güzel oldu. Benim Unity hakkındaki bilgilerim ve yeteneklerim sıfır diyebilirdim [dersten önce], bu dersten sonra şöyle bir geriye dönüp baktığımda arada dağlar kadar farkın olduğunu söyleyebilirim.” (D2_K15).

“Bu derste Unity kullanmayı öğrendim, bu da çok yararlı oldu. Bu dersin bir diğer avantajı ise deneme yanılma yapmaya müsait olması idi. Kodları bazen belli bir kalıbın içine sokmadan deneme yanılma yapabildim. Örneğin oyunun içinde kullanacağımız objemiz araba olsun. Buna yerçekimini vs. ekleriz. Gerçekçi bir araba sürme vs. yaparız. Bir şekilde işlerimizi hallederiz. Ancak hiçbir zaman yerçekimine bağlı olmayan bir obje yapmazsak, objenin yerçekimsiz ortamda neler yapabileceğini asla tecrübe etmemiş oluruz. Unity ortamı bize yerçekimsiz ortamda çalışsak neler olduğunun cevabını bulmak için olanak sağladı. Bazen nesneler üstünde deneme yanılma ile sonuçları görmeye çalıştım. Zevkli idi.” (D2_K12).

“Artık oyun oynarken Unity’de nasıl yapıldığını gözümde canlandırıyorum.” (D2_K2).

“Unity uygulaması ve C# hakkında daha çok öğrenimim arttı. Ders beklentilerimi karşıladı, hatta beklentilerimin üstünde oldu. Bu kadar oyun tasarlayacağımızı sanmıyordum. Bu ders sayesinde Unity hakkında birçok bilgi öğrenim elde ettim, Unity’de oyun tasarlama kodlama açısından yeni şeyler öğrendim.” (D2_K28).

Öğretim sürecinin katılımcılara katkı sağladığı bir diğer alan, mesleki açıdan yeni bilgiler öğrenmelerini teşvik etmesidir. Katılımcılardan dokuzu öğretim sürecinin meslek derslerine olan ilgilerini artırdığını ve bölümü ile ilgili yeni şeyler öğrenmeyi teşvik ettiğini belirtmiştir. Katılımcılar bu katkılarını şu cümlelerle ifade etmişlerdir:

“Derste kullandığımız oyun yöntemi bölüm derslerime ve mesleki bilgilere olan ilgimi etkiledi. Çünkü oyun benim hayatımın bir parçası ve oyun yapmak beni doğru yönde etkiliyor. Bu dersi almaya başladıktan sonra gerek merakımdan gerekse kendimi derse daha hazır hale getirmek için Udemy üzerinden dersler satın aldım. Ve videolarla birlikte orada denilenleri komut şeklinde kendimde yapmaya çalıştım. NYP dersi haricinde boş zamanlarımda Udemy'i kullanarak kendime yeni şeyler buluyordum ve bu ders sayesinde daha doğrusu bu ders bana farklı şeylere merak açtı. Blender programı, Home Design3D programını kullandım. Belki bu alakasız, kod sistemi yoktu ama bu ders beni bu şekilde de etkiledi ve ileriki hayatımda Movavi Editor, Movavi Editör+ programlarını kullanmak istiyorum. Bunları kullanmak isteme sebebim daha çok iç mimar gibi tasarımdan zevk almam. Hatta derste Karting Microgame tasarımımızdaki bana en keyif veren nokta oyuna bir takım şeyler şerit, bariyer vs. şeyler eklemektir.” (D2_K10).

“Bu ders bilgisayara ve bölüm derslerime olan ilgimi arttırdı. Şöyle ki okulda yaptıklarımıza bakınca bölümümüzün sadece bizi dersten geçirecek kadar bilgi edindirmesi gibi geliyordu bana. Ama bu süreçteki yaptıklarımızın bana katkısı çok oldu. Gerçekten kod yazdığımı, bölümümün hakkını verdiğimi düşündüm. Diğer üniversitelerde aynı bölümü okuduğum arkadaşıma baktığımda biz baya iyi işler yapıyorduk. Bunu arkadaşıma anlattığımda ise arkadaşım bana bizim daha iyi işler yaptığımızı ve gerçek bilgisayarcının bunu yapması gerektiğini söyledi. O da gerçekten yaptıklarımıza özendi. Keşke biz de bunları yapsak çok boş şeyler yapıyoruz dedi. Bunları duymak yaptıklarımıza daha çok bağlanmamı sağladı.” (D2_K8).

“Programlama adına aslında bu zamana kadar ciddi olarak çok fazla bir şey üretmemiş olduğumuzu bu dersle kavradım. Yani bilgisayar çağında büyüdük ancak çok fazla programlama araştırmadık, yeni şeyler öğrenmek için gayret etmedik. Bu ders bize bu alanda da gelişmemiz gerektiğini fark ettirdi.” (D2_K17).

Katılımcılar, oyun prototipi üzerinde yaptıkları görsel değişikliklerin hayal güçlerine ve tasarım yeteneklerine de katkıda bulunduğunu belirtmişlerdir. Katılımcılar bu durum hakkındaki düşüncelerini şu cümlelerle ifade etmişlerdir:

"Oyun projesi üzerinde çalışmak tasarım anlamında ufkumu genişletti, hayal gücümü daha çok canlandırdı. Oyun içinde yaptığımız pist, engeller vb. tasarımlar bence bize tecrübe kazandırdı. Daha önce tasarım yapmamıştım." (D2_K23).

"Dersin sevdiğim özelliklerinden bir tanesi her zaman bizi daha meraklı kılması ve geliştirici seçenleri olması, hayal gücümüze dayalı bir şekilde ilerlemesi ve hayal gücümüzü zorlamasıydı. Hayal dünyasını geliştiren çoğu şeyi özgür kılan bir ortamdı. Kısaca bizi düşünmeye ve hayal etmeye sevk etti diyebilirim." (D2_K13).

"Kodlama bilgimin ötesinde tasarım açısından da bu dersin bana çok katkısı oldu. Oyunuma daha fazla ne ekleyebilirim diye üzerinde düşündüm, oyuna eklemek için asset store'dan eklentiler araştırdım. Ağaçlar eklemek, panolar eklemek, binalar eklemek, pist tasarımını değiştirmek. Bunlar hep katkı sağladı." (D2_K5).

Öğretim sürecinin sağladığı bir diğer katkı, derste gerek NYP paradigması gerekse oyun programlamayla ilgili kazanılan becerilerin katılımcıların istihdam olanaklarını artırabilecek olmasıyla ilgilidir. Katılımcılardan dokuzu bu dersi almanın kendilerinin istihdam olanaklarını artırabileceğini, bilgi ve beceri seviyeleri itibariyle akranlarının bir adım önünde olabileceklerini belirtmişlerdir. D2_K8 kodlu katılımcı bu konu hakkındaki düşüncelerini şu cümlelerle ifade etmişlerdir:

"Bu derste edindiğim deneyimler benim iş başvurularında elimi kuvvetlendirecektir. Bir oyun firmasına başvurursam ya da bu sektörde bir işe girmek istersem derste öğrendiklerimden bahsedebilirim."

Öğretim sürecinin katkı sağladığı son alan katılımcıların programlamaya yönelik motivasyon ve öz yeterlilik algılarıyla ilgilidir. Bu katkılarla ilgili detaylı analiz sonuçları, 4.2.3.2.1 ve 4.2.3.2.2 bölümlerinde sunulmuştur.

4.2.3.3.4. Katılımcıların ders materyalleri hakkındaki görüşleri

Katılımcıların ders materyalleri hakkındaki görüşleri, Karting Microgame oyun prototipi ve ders videoları bağlamında elde edilerek çözümlenmiştir. Analiz sonuçları Tablo 4.63'te verilmiştir. Tabloya göre katılımcıların çoğu (N:22) ders materyali olarak kullanılan Karting Microgame oyun prototipinin öğrenilmesinin kolay olduğunu düşünürken sekiz katılımcı öğrenilmesini orta zorlukta ve zor olarak nitelendirmektedir. Katılımcılar bu konu hakkındaki düşüncelerini şu cümlelerle aktarmışlardır:

"Derste kullandığımız prototip kolay anlaşılır olduğu için bizlere faydalı oldu. Oyunun temel olarak altyapısı olduğu için öğrenilmesi kolay oldu." (D2_K5).

"Bence hiç öğrenilmesi güç değil. Araba yarışı sevenler için ilgi çekici olabilir. Unity'nin en temeli olabilir öğrenmek için. Çünkü zaten hazır olan bir oyunda tasarım değişikliği yaptık ve kodlar yazdık. Bu yüzden daha çabuk anladık." (D2_K2).

"Karting Microgame orta zorlukta ve ilgi çekici bir yarış oyunu idi. Geliştirmeye değiştirmeye çok açık basit bir oyundu, üzerinde değişiklikler yapmak hem kolay hem de zevkliydi." (D2_K18).

"Karting Microgame bana zor geldi. Nesnelerin ve sınıfların ne işlevleri olduğunu anlamam zaman aldı." (D2_K21).

Tablo 4.63. Katılımcıların ders materyallerine yönelik görüşleri*

Tema	Kategori	Alt Kategori	Kodlar	Frekans (f)	Yüzde (%)
Katılımcıların Ders Materyallerine Yönelik Görüşlerinin Değerlendirilmesi	Karting Microgame Oyun Prototipi	Zorluk Seviyesi	Öğrenilmesi kolaydır.	22	73
			Öğrenilmesi orta zorluktadır.	7	23
			Öğrenilmesi zordur.	1	3
		İlgi Çekicilik	İlgi çekicidir.	8	27
			İlgi çekici değildir.	3	10
	Ders Videoları	Olumlu	Konuyu anlamak için yararlıdır.	23	77
		Olumsuz	Uzun anlatımlara sahiptir.	6	20
			Arka plan müziği yoktur.	4	13
			Ses kalitesi yeterli değildir.	2	7

Karting Microgame oyun prototipinin ilgi çekiciliği konusunda katılımcıların birbirinden farklı düşündükleri sonucuna varılmıştır. Bazı katılımcılar göre (N:8) kullanılan oyun prototipi ilgi çekici bulurken bazı katılımcılara göre (N:3) ise kullanılan oyun prototipi yaş gruplarına uygun değildir. Katılımcılar bu konu hakkındaki düşüncelerini şu cümlelerle ifade etmişlerdir:

"Derste yapılan oyun günümüz şartlarında oyun dünyasında üst sıralarda olan bir oyun türündeydi ve herkesin ilgisini çekecek cinstendi. Parkur olsun arka fondaki haritalar olsun araçlar olsun amaçlar olsun hepsi ilgi çekiciydi ve öğrenmek açısından da seçilecek en iyi oyunlardan biriydi. Çünkü hem ilgi gören bir oyun türüydü hem de yapımı biraz daha kolay ve temel atmak için gayet mantıklı bir seçimdi." (D2_K14).

"İlgi çekici çok bir yanı olmadığını düşünüyorum. Benim için sadece sürüş deneyimi vermesi güzel. Yaş Aralığı 4-8 olan bir oyun tasarladık, bu yüzden ilgimi fazla çekmedi." (D2_K28).

Katılımcıların ders videoları hakkındaki görüşleri olumlu ve olumsuz olmak üzere ikiye ayrılmaktadır (Tablo 4.63). Katılımcıların %77'sine (N:23) göre NYP dersi için hazırlanan ders videoları faydalıdır. Ancak bazı katılımcılar videoların süresini uzun bulmuş (N:6); ses kalitesini de yeterli olarak görmemiştir (N:2). Dört katılımcı ise videolarda arka plan

* Katılımcı sayısı: 30. Bir katılımcının vermiş olduğu cevap birden fazla kod içerebilir.

müziğinin olmamasını bir eksiklik olarak belirtmiştir. Katılımcılar ders videoları hakkındaki görüşlerini şu cümlelerle ifade etmişlerdir:

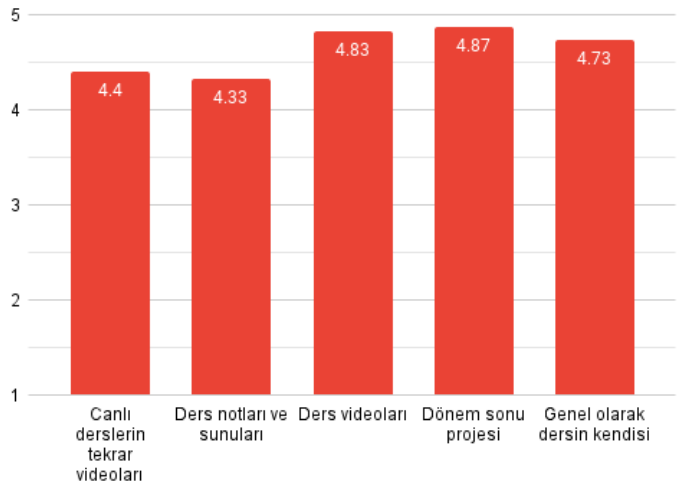
"Her hafta yaptığımız çalışmanın videolarla tekrarını seyretme imkânı sunmanız bizim için haftanın her saati sizden ders alma imkânı sundu. Geride kalırım korkusunu yenmemizi sağladı. Bu durum bizi daha motive şekilde derslere yönlendirdi. Hocam yüklediğiniz videolardan çok etkili çünkü internetteki videolar hep ağır yabancı diller var anlaşılması zor." (D2_K16).

"Geçen haftadaki dersimizde bize çok yardımcı olacak bilgiler öğrendik fakat tekrar etmediğim için bir kısmı kalıcı olmadı. Fakat bunları videoları izleyerek kapatabileceğimi düşünüyorum." (D2_K23, üçüncü hafta günlüğü).

"Ders videolarının bazıları gerçekten uzundu. Bazen izlerken ara vermek zorunda kaldığım oldu." (D2_K19).

"Videolar bazen monotonlaştı. Videolarda arka plan müziği olsaydı daha fazla yoğunlaşma şansımız olabilirdi." (D2_K22).

Son olarak, katılımcılardan derste kullanılan materyalleri katkı sağlama durumuna göre *bir (hiç katkısı olmadı)-beş (çok katkısı oldu)* skalasında değerlendirmeleri istenmiştir. Derste kullanılan materyallerle ilgili katkı durumu grafiği Şekil 4.71'de sunulmuştur. Şekle göre en çok katkıyı *dönem sonu projesi* ve *ders videoları*; en az katkıyı ise *ders notları ve sunuları* sağlamıştır.



Şekil 4.71. Ders materyallerinin katkısının değerlendirilmesi

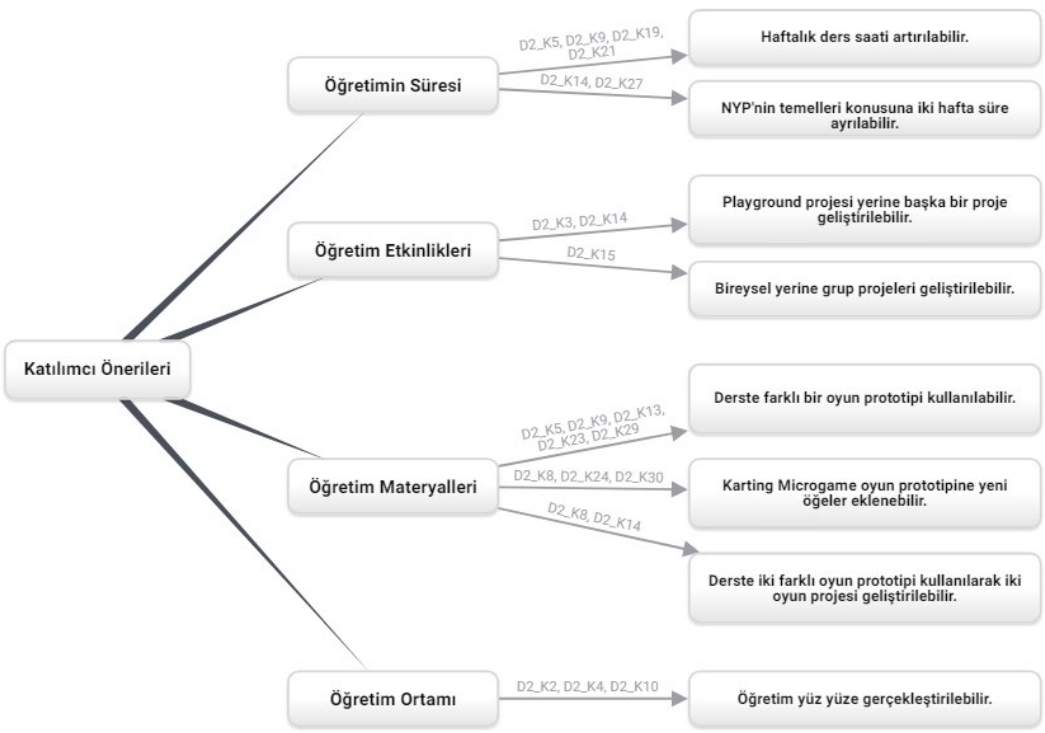
4.2.3.3.5. Katılımcıların uygulamanın geliştirilmesine yönelik önerileri

Katılımcıların oyun geliştirmeye dayalı programlama öğretim sürecine yönelik önerileri; (1) öğretim süresi, (2) öğretim etkinlikleri, (3) öğretim materyalleri ve (4) öğretim

ortamı olmak üzere dört kategoride çözümlenerek Şekil 4.72'de sunulmuştur. *Öğretim süresiyle* ilgili olarak dört katılımcı dersin haftalık ders saatinin bir saat artırılarak üç saat yerine dört saat işlenebileceğini; bunun da dersin verimliliğini artırabileceğini belirtmişlerdir. Bunun yanında iki katılımcı ise öğretim programının beşinci haftasında işlenen *"Nesne Yönelimli Programlamanın Temelleri*" konusunun iki hafta süreyle işlenebileceğini belirtmiştir. Katılımcılar bu konu hakkındaki görüşlerini şu cümlelerle ifade etmişlerdir:

"Ben dersin gayet verimli geçtiğini düşünüyorum ama dersin süresi bir saat artırılarak kod kısımlarının üstünde biraz daha durulabilir. Bileşen çoktu, oluşturduğumuz sınıf da çoktu. Bazen neyi nereye yazacağımızı da karıştırabiliyoruz. Uzun ders saati sayesinde kodların daha detaylı açıklanması sağlanabilir. İngilizce terimlerin ve hataların çok daha detaylı açıklaması yapılabilir. Bu sayede daha çok akılda kalabilir." (D2_K21).

"Nesne yönelimli programlamanın temelleri konusu için ayrılan süre daha fazla olabilirdi, biz sadece bir hafta işledik. Bence daha fazla detaya girilebilirdi, teorik kısımdan bahsediyorum. İki hafta sürebilirdi [konu]. Üzerinde çok fazla konuşup konuyu detaylı işleyebilirdik." (D2_K14).



Şekil 4.72. Katılımcıları öğretim süreciyle ilgili önerileri

Öğretim etkinlikleriyle ilgili olarak iki katılımcı, Unity 3D editörü kullanım becerisinin kazandırılması için gerçekleştirilen ve hiçbir C# kodunun yazılmasını gerektirmeyen Unity Playground projesinin yerine kod yazma aktiviteleri içeren bir başka proje geliştirilebileceğini, bu sayede Karting Microgame projesine geçilince daha kolay kod

yazabileceklerini belirtmişlerdir. D2_K14 kodlu katılımcı bu konuyla ilgili düşüncelerini şu cümlelerle ifade etmiştir:

"Playground'da iki hafta çalıştık ancak hiç kod yazmadık. Bence onun yerine kod yazabileceğimiz, bizi kodlama yapmaya alıştıracak küçük bir oyun tasarlayabilirdik. Bu sayede esas prototipe geçince daha kolay ve akıcı kod yazabilirdik. Playground'da kod yazmadık, Karting prototipinde yazdık, biraz zorlandık."

Öğretim etkinlikleriyle ilgili D2_K15 kodlu katılımcı ise bireysel projeler yerine grup projeleri geliştirmenin daha faydalı olabileceğini; bunun akran öğrenmesine fırsat tanıyabileceğini şu cümlelerle ifade etmiştir:

"Prototip üzerinde çalışırken hatalarla karşılaşıyorduk ve hatanın nedenini bazen bulamıyorduk. İnternetten araştırınca da hataların nedenini bulamıyorduk, zaten kaynakların çoğu yabancıydı. Ya size [araştırmacı] soruyorduk ya da arkadaşlarımıza soruyorduk. Mesela D2_21 isimli arkadaşım bana birkaç defa uzaktan bağlandı, hatalarımın kaynağını gösterdi. Bu da oldukça öğretici oldu. Bu proje grup projesi olarak yapılabilir. Bu sayede daha güzel ürünler ortaya çıkabilir."

Öğretim materyalleriyle ilgili olarak beş katılımcı mevcut oyun prototipinin kendileri için basit olduğunu belirtmiştir. Bu katılımcılar daha ilgi çekici ve yaş gruplarına uygun bir oyun prototipi kullanmanın dersin verimliliğini artırabileceğini şu cümlelerle ifade etmişlerdir:

"Prototipimiz çok basitti. Üzerinde eklentiler yapmak da kolaydı. Görsel açıdan daha ilgi çekici bir prototip kullanılabilir. Bizi biraz daha zorlasa daha iyi olabilirdi. Daha öğretici olabilirdi. Uygulama, yani oyunda gerçekleştirdiğimiz uygulama biraz daha zorlayıcı seviyede olabilirdi." (D2_K9).

"Günümüzde daha çok FPS [birinci şahıs nişancı] tarzı oyunlar daha çok oynanıyor. Tamam, yarış oyunları falan da güzel ancak FPS tarzı oyunlar daha dikkat çekici. Yapımı daha zevkli olabilir. Mesela level'ları [aşamalar] olabilir. Oyun bir karakter içerebilir. Amaçlar daha değişik, daha gerçekçi olabilir." (D2_K23).

"Bizim kullandığımız prototipin grafikleri basit kaçıyordu, daha görsel, grafik açısından daha üstün bir prototip kullanılabilir. Yalnız burada şöyle bir sorun var. Benim bilgisayar sistemim daha karmaşık bir oyuna yeterdi bazı arkadaşlarımın yetmeyebilirdi. Bunu da göz önüne almak gerekir." (D2_K5).

Öğretim materyalleriyle ilgili olarak üç katılımcı mevcut Karting Microgame oyun prototipine canavarlar, yeni araçlar, sürücüler, düşmanlar vb. öğelerin eklenerek oyunun hedeflerinin geliştirilebileceğini ve oyunun daha sürükleyici hale gelebileceğini belirtmişlerdir. Katılımcılar bu konu hakkındaki görüşlerini şu cümlelerle ifade etmişlerdir:

"Mevcut oyun prototipine farklı sürücüler, düşmanlar, canavarlar, uçma efektleri kazandırılabilirdi. Arabamız sağa sola hareket etmek yerine uçabilirdi. Yok etmesi gereken bir şey olabilirdi. Objeleri topluyorduk biz. Toplamanın yanında yok edilecek nesneleri gördüğünde onları yok edebilirdik. Sürücümüz bazı hareketli nesnelerden kaçabilirdi. Daha güçlü hedefler ve daha güçlü bir senaryo eklenebilir." (D2_K24).

"Daha renkli uygulamalar derken daha çok dış mekân, daha farklı unsurlar eklenebilirdi. Mesela bir canavar. Onu öldürdükçe puanımız artabilirdi. Ondan kaçmaya çalışabilirdik, o bize vurdukça puanımız azalabilirdi. Yani oyunu biraz daha çekici kılmak için geliştirebilirdik." (D2_K30).

Öğretim materyalleriyle ilgili olarak iki katılımcı ise öğretim materyali olarak hem mevcut Karting Microgame oyun prototipinin kullanılmasını, hem de ikinci bir oyun prototipinin öğretim sürecinde yer almasını önermiştir. Bu katılımcılar öğretim sürecinin başlarında Karting Microgame oyun projesinin, devamında daha gelişmiş bir oyun projesinin geliştirilmesinin dersin verimliliğini artıracağını belirtmişlerdir. D2_K14 kodlu katılımcı bu konu hakkındaki düşüncelerini şu cümlelerle aktarmıştır:

"Mevcut prototip basitti, nesne yönelimli programlamayı anlamamızı sağladı. Programlamaya karşı olan önyargımızın kırılmasını sağladı. Bizi programlamaya yönlendirdi ve ilgimizi çekti. Düşüncem şuydu. Bu oyun prototipinde eklentiler yaptıktan sonra kendimi yeni şeyler yapmak için elverişli hissettim. Yeni bir şeyler yapmak için kendime güvenim geldi. Yapabileceğim potansiyele sahibim. Benim teklifim şu yönde. Nesne Yönelimli Programlama dersinin temeli için bu prototiple başlanabilir. Mesela sınıflar, nesneler, davranışlar için bu prototipi kullanıp kalıtım vb. konular için daha gelişmiş bir prototipe geçilebilir."

Katılımcıların öğretim süreciyle ilgili önerilerinin son kategorisi, *öğretim ortamıyla* ilgilidir. Üç katılımcı dersin uzaktan yerine yüz yüze öğretimle yapılmasının verimliliğini artıracağını ifade etmişlerdir. Katılımcıların bu konu hakkındaki düşünceleri şu şekildedir:

"Dersi laboratuvar ortamında yapsaydık daha verimli geçeceğine inanıyorum. O zaman karşılaştığımız hatalara daha hızlı çözümler bulabilirdik, hem öğretim görevlisinden hem de arkadaşlarımızdan daha hızlı yanıtlar alabilirdik. Evde olunca ister istemez bazen ders çalışmayı erteleyebiliyoruz, okulda olsaydı mecburen çalışırdık. (D2_K2).

"Canlı derste sınıfta olduğumuz kadar etkileşim halinde olamıyoruz. Herkes aynı anda konuşmak isteyebiliyor mesela. Bir de internet sıkıntısı da yaşanabiliyor, bazılarımızın interneti iyi oluyor bazılarımızın kötü. Mesela benim iki hafta internet hızım çok kötü olduğu için çok fazla verim alamadım. Görüntüde çok sıkıntı olmadı ama seste sıkıntı oldu, verimli olmadı Ders uzaktan olduğu için yüzde yüz bir verim alındığı söylenemez. Yüz yüze daha etkili olacağı kanaatindeyim." (D2_K4).

4.3. Birinci ve İkinci Döngü Karşılaştırması

4.3.1. Akademik Başarı Açısından Karşılaştırma

Birinci ve ikinci döngünün akademik başarı açısından karşılaştırılması, hem NYP Başarı Sınavından elde edilen toplam puan ortalamaları hem de her bir kazanımdan elde edilen puan ortalamaları ve başarı yüzdeleri dikkate alınarak gerçekleştirilmiştir. Akademik başarı açısından karşılaştırmanın ilk adımında, her iki döngüde katılımcıların NYP Başarı

Sınavından elde ettikleri öntest ve sontest puanları karşılaştırılmıştır. Bu karşılaştırmaya yönelik betimsel istatistikler Tablo 4.64'te sunulmuştur. Tabloda belirtildiği üzere her ne kadar katılımcıların öntest puan ortalamaları birbirine yakınken ikinci döngünün sontest puan ortalaması birinci döngü sontest puan ortalamasından 18.36 puan fazladır. Bu farkın istatistiksel olarak anlamlı olup olmadığını belirlemek için her iki döngüye ait öntest ve sontest puanları karşılaştırılmıştır.

Tablo 4.64. NYP Başarı Sınavı öntest ve sontest puanlarına yönelik betimsel istatistikler

		Öntest		**Sontest**	
	N	**x̄**	**ss**	**x̄**	**ss**
Birinci Döngü	29	5.18	4.11	67.56	19.83
İkinci Döngü	30	4.56	3.3	85.92	13.04

İlk olarak her iki döngünün öntest puanları karşılaştırılmıştır. Her iki döngünün öntest puan ortalamaları normal dağıldığı için (Shapiro-Wilk testi sonuçlarına göre p>.05) karşılaştırma için t testi kullanılmıştır (Pallant, 2020). Test sonuçları Tablo 4.65'te verilmiştir. Tabloya göre katılımcıların birinci ve ikinci döngüdeki NYP Başarı Sınavı öntest puan ortalamalarında istatistiksel açıdan anlamlı bir fark çıkmamıştır ($t_{(57)}$=1.976; p=.06).

Tablo 4.65. Döngülere ait öntest başarı puan ortalamalarının karşılaştırması

Uygulama	**N**	**x̄**	**ss**	**sd**	**t**	**p**
Birinci Döngü	29	5.18	4.11	57	1.976	.06
İkinci Döngü	30	4.56	3.3			

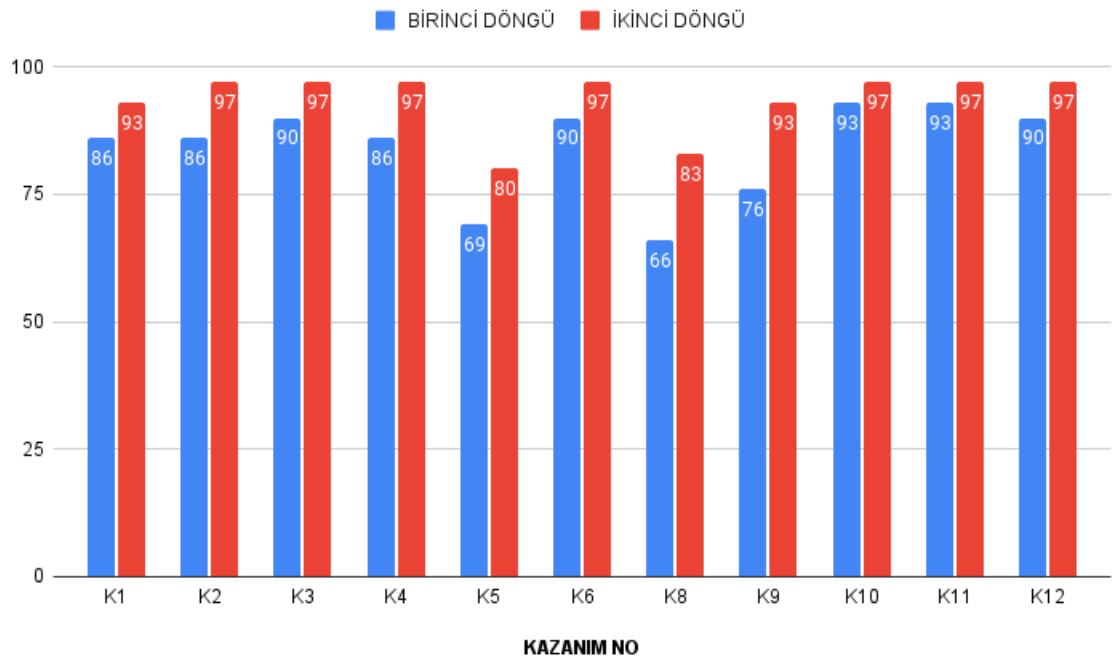
Öntest puan ortalamalarının karşılaştırılmasının ardından döngülere ait sontest puanları karşılaştırılmıştır. Her iki döngünün sontest puan ortalamaları normal dağılmadığı için (Shapiro-Wilk testi sonuçlarına göre p<.05) karşılaştırma için Mann Whitney U testi kullanılmıştır (Pallant, 2020). Test sonuçları Tablo 4.66'da verilmiştir. Tabloya göre birinci ve ikinci döngü sontest başarı puanları arasında ikinci döngü lehine anlamlı bir ilişki bulunmaktadır U=284.000, p=.018.

Tablo 4.66. Döngülere ait sontest başarı puan ortalamalarının karşılaştırması

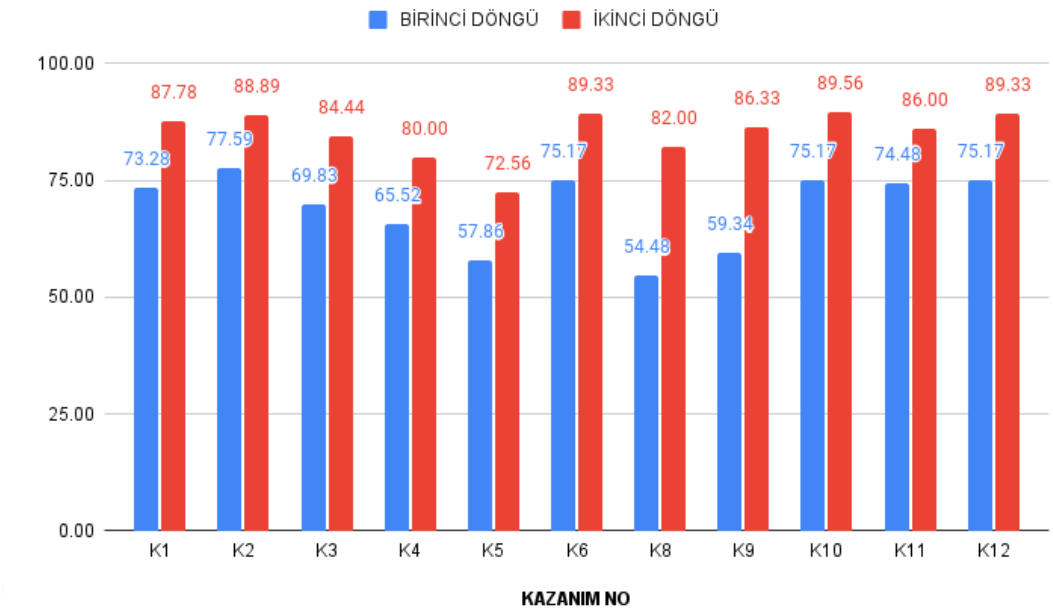
Değişken	**N**	**Sıra x̄**	**Sıra ∑**	**U**	**z**	**p**
Birinci Döngü	29	35.21	1021.00	284.000	-2.37	.018
İkinci Döngü	30	24.97	749.00			

Akademik başarı karşılaştırmasının ikinci adımında, kazanım bazında başarı yüzdesi ve puan ortalamaları karşılaştırması gerçekleştirilmiştir. Teorik sınav karşılaştırma sonuçları

Şekil 4.73 ve Şekil 4.74'de sunulmuştur. Teorik sınav başarı yüzdesi açısından karşılaştırma sonuçları değerlendirildiğinde, tüm kazanımlarda başarı yüzdesi ikinci döngü lehine olumlu sonuçlanmıştır (Şekil 4.73). Bununla birlikte birinci döngüde başarı yüzdesi en düşük olan "*Kurucu metot kavramını örneklerle açıklar." (K5)* ve "*Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar." (K8)* kazanımlarının başarı yüzdelerinde sırasıyla 11 ile 17 puanlık bir artış yaşanmıştır. İlave olarak bu kazanımların puan ortalamalarında sırasıyla 14.7 ile 27.52 puanlık yüksek bir artış yaşanmıştır (Şekil 4.74). Buna rağmen K5 kazanımının puan ortalaması 72.56'da kalmış ve K5 kazanımının puan ortalaması ikinci döngüde diğer kazanımlara göre düşük kalmıştır.

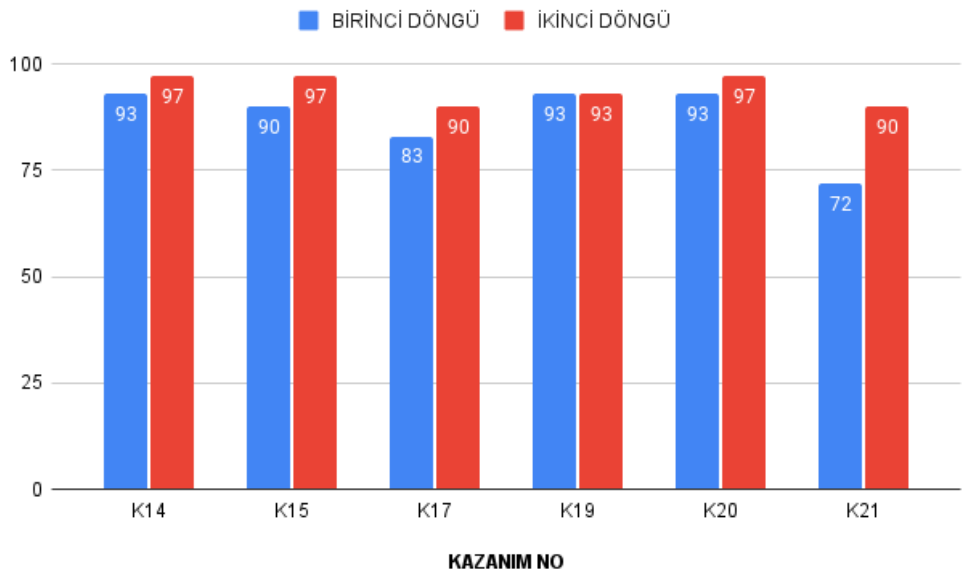


Şekil 4.73. Teorik sınav başarı yüzdesi açısından karşılaştırma sonuçları

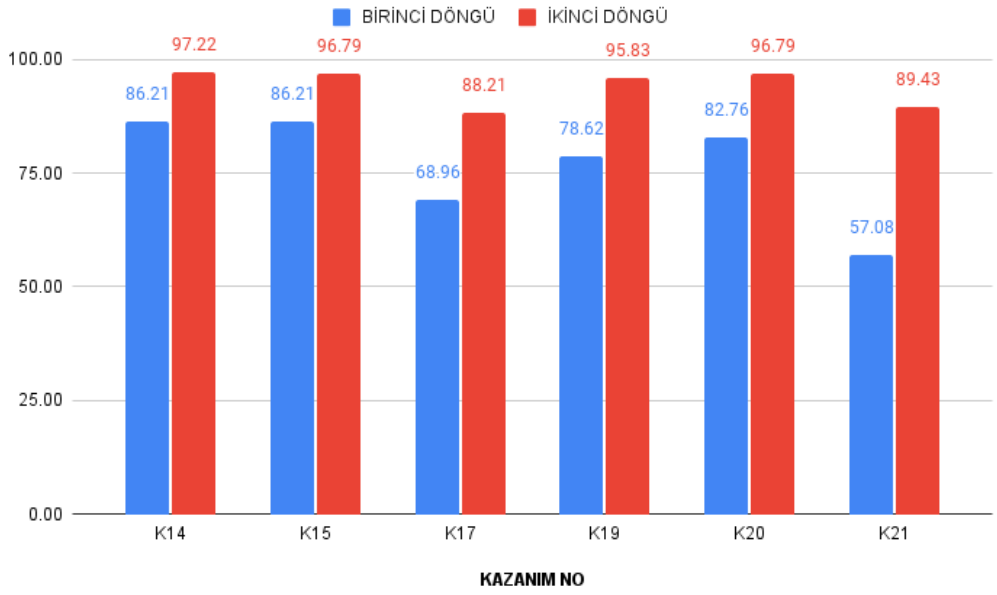


Şekil 4.74. Teorik sınav puan ortalaması açısından karşılaştırma sonuçları

Teorik sınav sonuçlarının karşılaştırmasına ilave olarak, uygulama sınav karşılaştırma sonuçları Şekil 4.75 ve Şekil 4.76'da sunulmuştur. Uygulama sınavı başarı yüzdesi açısından karşılaştırma sonuçları değerlendirildiğinde, "UML sınıf diyagramı verilen sınıfı C# programlama dilinde oluşturur." (K19) kazanımı hariç tüm kazanımlarda başarı yüzdesi ikinci döngü lehine olumlu sonuçlanmıştır (Şekil 4.75). Bununla birlikte birinci döngüde başarı yüzdesi en düşük olan "UML sınıf diyagramında belirtilen çokbiçimli metotları oluşturur ve kullanır." (K17) ve "Bir C# sınıfına ait Monobehavior metotlarını, verilen koşula göre oluşturur." (K21) kazanımlarının başarı yüzdelerinde sırasıyla 7 ile 18 puanlık bir artış yaşanmıştır. K21 kazanımdaki artışın K17 kazanımına oranla daha fazla olduğu görülmüştür. İlave olarak bu kazanımların puan ortalamalarında sırasıyla 19.25 ile 32.35 puanlık yüksek bir artış yaşanmıştır (Şekil 4.76). Bu artışlarla birlikte bu iki kazanımın puan ortalamaları, birinci döngüde diğerlerinden düşük olmasına rağmen ikinci döngüde diğer kazanımların puan ortalamaları seviyesine ulaşmıştır. Son olarak; "Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar." (K8) kazanımı ile "UML sınıf diyagramında belirtilen çokbiçimli metotları oluşturur ve kullanır." (K17) kazanımlarının puan ortalamalarının ilk döngüde diğer kazanımlara kıyasla görece düşük çıkıp ikinci döngüde yükselmesi; teorik sınav ile uygulama sınav sonuçlarının birbirini desteklediğini göstermektedir.



Şekil 4.75. Uygulama sınavı başarı yüzdesi açısından karşılaştırma sonuçları



Şekil 4.76. Uygulama sınavı puan ortalaması açısından karşılaştırma sonuçları

4.3.2. Algılanan Programlama Bilgi ve Zorluk Seviyesi Açısından Karşılaştırma

Birinci ve ikinci döngü uygulamalarını neticesinde katılımcıların algıladıkları programlama bilgi seviyelerindeki değişimler Tablo 4.67'te sunulmuştur. Tabloya göre ikinci döngüde katılımcıların %67'sinin algıladıkları programlama bilgi seviyesinde artış yaşanırken bu durum birinci döngüde %21 ile sınırlı kalmıştır. Benzer şekilde birinci döngüde katılımcıların %79'unun algıladıkları programlama bilgi seviyelerinde herhangi bir

değişim yaşanmazken bu oran ikinci döngüde önemli derecede azalarak %26 seviyesine düşmüştür.

Tablo 4.67. Algılanan programlama bilgi seviyesindeki değişimlerin karşılaştırılması

Değişim Durumu	Birinci Döngü		İkinci Döngü	
	Frekans (f)	Yüzde (%)	Frekans (f)	Yüzde (%)
Algılanan Programlama Bilgi Seviyesi Artan	6	21	20	67
Algılanan Programlama Bilgi Seviyesi Azalan	-	-	2	7
Algılanan Programlama Bilgi Seviyesi Değişmeyen	23	79	8	26

Algılanan programlama bilgi seviyesindeki değişime ilave olarak birinci ve ikinci döngü uygulamalarını neticesinde katılımcıların algıladıkları programlama zorluk derecesindeki değişimler de karşılaştırılmış ve sonuçlar Tablo 4.68'de sunulmuştur. Tabloya göre birinci ve ikinci döngü uygulamaları sonucunda algılanan zorluk seviyesindeki değişimler büyük oranda benzerlik göstermiştir.

Tablo 4.68. Algılanan programlama zorluk seviyesindeki değişimlerin karşılaştırılması

Değişim Durumu	Birinci Döngü		İkinci Döngü	
	Frekans (f)	Yüzde (%)	Frekans (f)	Yüzde (%)
Algılanan Programlama Zorluk Seviyesi Artan	10	34	11	37
Algılanan Programlama Zorluk Seviyesi Azalan	3	11	5	16
Algılanan Programlama Zorluk Seviyesi Değişmeyen	16	55	14	47

4.3.3. Psikolojik Faktörler Açısından Karşılaştırma

Birinci ve ikinci döngü sontest puan ortalamaları, programlama dersine yönelik motivasyon, programlama kaygısı ve programlama öz yeterlilik algısı bağlamında karşılaştırılmıştır. Katılımcıların birinci ve ikinci döngü programlama dersine yönelik motivasyon sontest puan ortalamaları karşılaştırma sonuçları Tablo 4.69'da sunulmuştur. Tablo 4.66'ya göre katılımcıların ikinci döngüdeki programlama dersine yönelik motivasyon sontest puan ortalamaları birinci döngüye göre istatistiksel açıdan anlamlı şekilde yüksek çıkmıştır ($t_{(56)}$=4.31; p=.00) .

Tablo 4.69. Toplam programlama dersine yönelik motivasyon puanları ortalamaları karşılaştırması

Uygulama	N	$\bar{x}$	ss	sd	t	p
Birinci Döngü	28	3.58	.92	56	-4.31	.00
İkinci Döngü	30	4.55	.80			

Katılımcıların birinci ve ikinci döngü programlama kaygı sontest puanları, Programlama Kaygısı Ölçeği faktörleri olan "akran kaygısı" ve "programlama beceri kaygısı" bağlamında karşılaştırılmıştır. Bu faktörlere yönelik karşılaştırma sonuçları Tablo 4.70 ve Tablo 4.71'de sunulmuştur. Tablo 4.70'e göre katılımcıların birinci ve ikinci döngüdeki akran kaygısı sontest puan ortalamalarında istatistiksel açıdan anlamlı bir fark çıkmamıştır ($t_{(55)}$=-.045; p=.96). Benzer şekilde Tablo 4.71'e göre katılımcıların birinci ve ikinci döngüdeki programlama beceri kaygısı sontest puan ortalamalarında istatistiksel açıdan anlamlı bir fark çıkmamıştır ($t_{(55)}$=-.437; p=.66).

Tablo 4.70. Akran kaygısı faktörü ortalamaları karşılaştırması

Uygulama	N	$\bar{x}$	ss	sd	t	p
Birinci Döngü	27	2.37	1.06	55	-.045	.96
İkinci Döngü	30	2.38	1.1			

Tablo 4.71. Programlama beceri kaygısı faktörü ortalamaları karşılaştırması

Uygulama	N	$\bar{x}$	ss	sd	t	p
Birinci Döngü	27	2.87	1.06	55	.437	.66
İkinci Döngü	30	2.75	.94			

Son olarak katılımcıların birinci ve ikinci döngü programlama öz yeterlilik algısı sontest puan ortalamaları karşılaştırma sonuçları Tablo 4.72'de sunulmuştur. Tablo 4.72'ye göre katılımcıların ikinci döngüdeki programlama öz yeterlilik algısı sontest puan ortalamaları birinci döngüye göre istatistiksel açıdan anlamlı şekilde yüksek çıkmıştır ($t_{(55)}$=2.66; p=.01).

Tablo 4.72. Programlama öz yeterlilik algısı puanları ortalamaları karşılaştırması

Uygulama	N	$\bar{x}$	ss	sd	t	p
Birinci Döngü	27	3.57	.83	55	-2.66	.01
İkinci Döngü	30	4.08	.63			

BÖLÜM V: SONUÇ, TARTIŞMA VE ÖNERİLER

Bu araştırma kapsamında Türkiye'deki bir meslek yüksekokulunun Bilgisayar Teknolojisi bölümü ikinci sınıfında okutulan NYP dersinde yaşanan güçlüklerin en aza indirilebilmesi ve dersin verimliliğinin artırılabilmesi için oyun geliştirmeye dayalı öğretim stratejisinin kullanıldığı bir eylem planı geliştirilmiş, mevcut NYP dersi bu stratejiye uygun olarak yeniden tasarlanmış ve iki döngü içerecek şekilde uygulanmıştır. Bu bağlamda, uygulanan eylem planının katılımcıların NYP bilgi ve beceri düzeyleri üzerindeki etkileri araştırılırken aynı zamanda programlama dersine yönelik motivasyon, programlama öz yeterlilik algısı ve programlama kaygısı açısından katılımcılarda oluşan psikolojik etkiler üzerine incelemelerde bulunulmuştur. Son olarak araştırma soruları kapsamında, katılımcıların gerçekleştirilen NYP öğretim sürecine yönelik görüşleri, oyun geliştirmeye dayalı öğretimle ilgili deneyimleri, bu sürecin katılımcılara olan katkısı ile ilgili görüşleri değerlendirilmiştir. Bu bölümde araştırma süresince elde edilen bulgular alanyazınla tartışılarak yorumlanmış, sonuçlar ortaya konmuş ve gelecek çalışmalar için önerilerde bulunulmuştur.

5.1. Sonuç ve Tartışma

Bu eylem araştırmasının temel motivasyonu; araştırmacının öğretim elemanı olarak vermekte olduğu NYP dersinde yaşanan sorunları en aza indirebilmek olmuştur. Bu bağlamda "*Öğrencilerimin Nesne Yönelimli Programlama dersindeki motivasyonlarını nasıl artırabilirim?"*, "*Derste kullanılan ödev ve projeleri öğrenciler açısından nasıl daha anlamlı hale getirebilirim?"*, "*Nesne Yönelimli Programlamaya ait kavram ve tekniklerin öğrenilmesini nasıl kolaylaştırabilirim?"* gibi kritik hususlar, NYP dersinin iyileştirilmesi sürecinde araştırmacı için temel hedefler olmuştur. Bu iyileştirme sürecinde NYP dersi oyun geliştirmeye dayalı programlama öğretimi stratejisi temel alınarak yeniden tasarlanmış; iki döngüden oluşacak şekilde uygulanmış ve bu uygulamalarla ilgili değerlendirmeler yapılmıştır. Araştırmanın sonuç ve tartışma bölümünde mevcut NYP dersinin yeniden tasarlanma süreci üzerinde yansıtma yapılmış; bununla birlikte hazırlanan eylem planının mevcut NYP dersinde yaşanan sorunların çözümünde ne derecede etkili olduğuyla ilgili elde edilen sonuçlar alanyazında yer alan çalışmalarla karşılaştırılmıştır. Bu kapsamda, elde edilen sonuçlar alanyazın ile tartışılırken (1) NYP dersinin yeniden tasarlanması süreci ve

(2) mevcut NYP dersinin sorunlarına yönelik üretilen çözümlerin etkinliği özelinde yorumlarda bulunulmuştur.

5.1.1. NYP Dersinin Yeniden Tasarlanması Sürecine İlişkin Sonuç ve Tartışma

Bu tez çalışmasının ilk basamağı olan analiz aşamasında, mevcut NYP dersinde öğrencilerin ve derse giren öğretim elemanlarının yaşadıkları güçlükler tespit edilmiştir. Bu kapsamda NYP dersini almış mezun öğrencilerden veri toplanmış; aynı zamanda NYP dersine giren öğretmenlerle görüşmeler yapılmıştır. Veri analizi sonucunda öğrencilerin sınıf, nesne, soyutlama ve kalıtım gibi NYP'nin temel kavramlarını anlamakta güçlük yaşadıkları bulunmuştur. Bunun yanında öğrencilerin NYP'de yer alan mekanizmaları bir yazılım projesi geliştirirken kullanmakta zorlandıkları; projelerini NYP kullanarak tasarlamakta sorun yaşadıkları sonucuna ulaşılmıştır. Bu sonuç, NYP kavramlarının soyut yapısının (Abbasi ve diğerleri, 2021; Akkaya ve Akpinar, 2022) öğrencilerin NYP mekanizmalarını hangi amaçla ve ne tür tekniklerle kullanmaları gerektiğine yönelik zihinsel modellerin oluşumunu güçleştirdiğini belirten çalışmalarla uyumludur (Abbasi, Kazi ve Khowaja, 2017; Brito ve Medeiros, 2019; Sunday, Wong, Samson ve Sanusi, 2022). Nitekim yapılan çalışmalar, öğrencilerin zihinsel modellerinin tam olarak oluşmamasının NYP kullanarak geliştirecekleri yazılım projelerinin isterlerini anlamalarını ve projelerinde kullanacakları sınıfları ve modülleri belirlemelerini zorlaştırdığını göstermektedir (Gainutdinova, Denisova ve Shirokova, 2019; Parastar, 2021).

Tüm bu güçlükleri en aza indirebilmek için uzman görüşleri ve alanyazın rehberliğinde mevcut NYP dersi oyun geliştirmeye dayalı programlama öğretiminin bir türü olan mevcut bir oyuna eklentiler yapma (game modding) stratejisini temel alacak şekilde yeniden tasarlanmıştır. Bu bağlamda üç boyutlu bir oyun prototipi, profesyonel bir oyun geliştirme aracı olan Unity 3D kullanılarak C# programlama diliyle öğrencilerin kendi hayal güçlerine bağlı olarak uyarlanmıştır. Oyunlarda yer alan mekanik ve dinamiklerin öğrenciler açısından bilinen bir alan olması; bunun yanında oyunlarda yer alan bileşenlerin tamamının nesnelerden oluşması (Wong ve Yatim, 2018) nedeniyle bu strateji benimsenmiştir. Böylece alanyazında NYP projelerinde kullanılan örneklerin öğrencilerin günlük hayatlarıyla ilişkilendirilerek zihinsel modellerin oluşumuna katkı sağlayacak yöntem ve etkinliklerin kullanılmıyor olması (Stoffová, 2019; Kotsovoulou, 2020) probleminin çözümüne yönelik bir adım atılmıştır.

Alanyazın incelendiğinde NYP paradigmasının oyun programlama yerine eğitsel bir oyunu oynama yoluyla öğrencilere benimsetildiği çalışmalar da mevcuttur (Abbasi ve diğerleri, 2021; Bouali, Nygren, Oyelere, Suhonen ve Cavalli-Sforza, 2019; Wong ve diğerleri, 2017; Smaragdina ve diğerleri, 2020). Bu açıdan değerlendirildiğinde, bu tez çalışmasının oyun oynayarak NYP'nin öğretildiği çalışmalardan ayrıldığı görülmektedir. Nitekim öğrencilerin NYP dersi kapsamında kendi oyunlarını geliştirme etkinlikleri; analiz, tasarım, kodlama ve test gibi yazılım geliştirme süreçlerini içermiştir. Öğrencilerin bu oyun geliştirme sürecinden elde ettikleri deneyimlerin, oyun oynarken bir içeriği öğrenmekten oldukça farklı olabileceği değerlendirilmektedir.

NYP dersinin yeniden tasarlanması sürecinde Türkiye'deki MYO'larda okutulan NYP derslerinin konu kapsamları incelenmiştir. Benzer şekilde alanyazındaki çalışmalarda belirtilen NYP konu kapsamları ve kazanımları araştırılmıştır. Bu araştırma neticesinde MYO'larda okutulan NYP dersleri için bütüncül bir çerçevede ortak bir ders içeriği standardının olmadığı; NYP derslerinin ağırlıklı olarak prosedürel programlamanın unsurlarının tanıtımından sonra NYP paradigmasına geçişi ifade eden "en son nesne (object last)" yaklaşımını barındırdığı sonucuna varılmıştır. Her ne kadar diğer MYO'lar için ortak bir standart olmamakla birlikte alanyazındaki çalışmalarda NYP dersi öğretim programının *sınıf, nesne, kapsülleme, kalıtım ve çokbiçimlilik* gibi konuları içerdiği görülmektedir (Kaila, Kurvinen, Lokkila ve Laakso, 2016; Salleh, Drus, Gunasekaran ve Dewi, 2021). Bu tez çalışması kapsamında geliştirilen NYP dersinin öğretim programında bahsedilen konu kapsamlarının yer alıyor olması, geliştirilen NYP dersinin içeriğinin alanyazınla uyumlu olduğunu göstermektedir.

Bu çalışma kapsamında geliştirilen NYP dersinin öğretim programının, oyun geliştirmeye dayalı öğretim stratejisinin kullanıldığı bir NYP dersi için örnek bir müfredat olarak alanyazına katkı sağlaması amaçlanmıştır. Morrison ve diğerlerinin (2004) öğretim tasarım modelini temel alacak şekilde gerçekleştirilen öğretim tasarımı, tüm detaylarıyla aktarılmaya çalışılmıştır. Böylelikle, Akkaya ve Akpinar (2022) ile Laporte ve Zaman'nın (2018) da çalışmalarında bir eleştiri olarak sunduğu oyun tabanlı öğrenmeyle ilgili yapılan çalışmaların sadece dersin kazanımlarına ulaşıp ulaşmadığına yönelik sonuçlara yoğunlaşıp öğretim tasarım süreciyle ilgili detayları içermemesi probleminin çözümüne katkı sunulmaya çalışılmıştır.

NYP dersi müfredatı, NYP kavramlarının mümkün olduğunca erken tanıtılacak şekilde tasarlanmıştır. Nitekim Livovský ve Porubän da (2014) NYP kavramlarının dersin

olabilecek en erken zamanlarında tanıtılmasını ve dönem boyunca sıklıkla bu kavramlara değinilmesini önermektedir. Bunun yanında alanyazında NYP kavramlarının geliştirilen yazılım projelerinde kullanılması ve uygulanması tavsiye edilmektedir. Bu öneriyle uyumlu olarak geliştirilen NYP müfredatında yer alan NYP kavramları, öğrenciler tarafından üç boyutlu bir oyun prototipi üzerinde uygulanarak teorik konuların pratiğe dönüştürülmesi sağlanmıştır. Bu sayede alanyazında belirtilen teori ile uygulama arasındaki boşluk doldurulmaya çalışılmıştır (Poolsawas, Chaipornkeaw ve Suparkawat, 2016).

Geliştirilen NYP dersi, öğrencilerin dönem boyunca tek bir proje üzerinde çalışarak tüm NYP tekniklerini bu oyun prototipine eklentiler yapma yoluyla deneyimlemelerine olanak tanıyacak şekilde tasarlanmıştır. Nitekim alanyazında da NYP dersinin daha verimli ve etkili olabilmesi için Wallinga'nın (2019) tabiriyle *basit projeler (toy projects)* yerine öğrencilerin NYP'yle ilgili büyük resmi görmelerini sağlayacak; grafik ve nesne açısından zengin ve kapsamlı projelerin geliştirilmesi önerilmektedir (Abbasi ve diğerleri, 2021). Bu oyun modu oluşturma deneyimi sonucunda öğrencilerin günümüzde hiçbir yazılımın sıfırdan başlamayacağını; yazılımlara fonksiyonel özellikler kazandırabilmek için mutlaka bir platforma ve önceden yazılmış kodları içeren kütüphanelere ihtiyaç duyulacağını fark etmelerini amaçlanmıştır (Dwarika ve De Villiers, 2015). Böylelikle öğrencilerin NYP paradigmasının neden önemli olduğunu ve son zamanlarda yazılım sektöründe neden yaygın olarak tercih edildiğini kavramalarını kolaylaştırmaya yönelik bir adım atılmıştır (Aniche, Yoder ve Kon, 2019).

Öğrenciler, geliştirilen NYP dersini bölümlerinde okutulan Algoritma ve Programlamaya Giriş ile Görsel Programlama derslerinin devamındaki yarıyılda almışlardır. NYP dersinden önce öğrenciler algoritma kavramı ve tasarlanması, akış şemaları, IDE kullanımı, C#'ta değişkenler, matematiksel ve mantıksal ifadeler, karar yapıları ve döngüler, diziler, metotlar gibi prosedürel programlamanın temelleriyle ilgili bilgi sahibi olmuşlardır. Bu yönüyle geliştirilen NYP dersinin "*daha sonra nesne (objects-later)*" yaklaşımıyla tasarlandığı söylenebilir. Çalışmanın gerçekleştirildiği MYO'nun öğretim programı değerlendirildiğinde, NYP dersinin daha sonra nesne yaklaşımını benimsemesi bir zorunluluktan kaynaklanmıştır. Alanyazında NYP dersinin *"ilk önce nesne (objects-first)"* yaklaşımıyla tasarlandığı çalışmalar da bulunmaktadır (Dolgopolovas, Jevsikova ve Dagiene, 2018; Kunkle ve Allen, 2016). Alanyazında her iki yaklaşımın kendine ait avantaj ve dezavantajlarından bahsedilmektedir. İlk olarak nesne yaklaşımında öğrencilerin hem programlamanın temellerini hem de nesne yönelimli programlamanın kavramlarını

öğrenmeleri gerekirken bu süreç çok fazla bilişsel yüke neden olmaktadır (Perez-Gonzalez ve diğerleri, 2021). Ancak daha sonra nesne yaklaşımında ise öğrencilerin önceden elde ettikleri prosedürel programlama deneyimi, NYP paradigmasını öğrenmeye geçişte kavram yanılgılarına neden olmakta ve NYP öğretiminin etkinliğini zayıflatmaktadır (Kanaki ve Kalogiannakis, 2018). Nitekim bu çalışmada da öğrenciler prosedürel programlamadan NYP paradigmasına geçişte zorlanmış; öğrencilerin nesne yönelimli düşünme tarzını benimsemesi zaman almıştır (Thuné ve Eckerdal, 2019).

Son olarak, NYP dersinin yüz yüze sunulacak şekilde tasarlandığı söylenebilir. Her ne kadar çalışmanın birinci döngüsü yüz yüze gerçekleştirilmiş olsa da COVİD-19 salgını nedeniyle ikinci döngü acil uzaktan öğretim stratejisi kullanılarak gerçekleştirilmiştir. Bewer ve Gladkaya'nın (2022) belirttiği gibi oyun geliştirmeye dayalı öğretim programlama alanyazınında büyük çoğunlukla yüz yüze gerçekleştirildiği görülmektedir. Bu tez çalışmasının ikinci döngüsünden elde edilen bulguların, oyun geliştirmeye dayalı öğretimin çevrimiçi kullanımına yönelik deneyimleri yansıtarak alanyazına katkı sağlaması amaçlanmıştır.

5.1.2. Mevcut NYP Dersinin Sorunlarına Yönelik Üretilen Çözümlerin Etkinliğine İlişkin Sonuç ve Tartışma

NYP dersinin geliştirilmesi sürecinde öncelikle ihtiyaç analizi yapılmış ve mevcut NYP dersinin problemleri tespit edilmiştir. Araştırmanın tasarım aşamasında ise tespit edilen problemlerin çözümüne yönelik bir eylem planı oluşturulmuş; uygulama aşamasında ise hazırlanan eylem planı hayata geçirilmiştir. Bu bölümde; uygulanan eylem planının mevcut NYP dersinin sorunlarının çözümünde ne derece etkili olduğu, alanyazınla tartışılarak sunulmuştur. Bu tartışma; mevcut NYP dersinin problem kategorilerinden (1) öğretim programı, (2) psikolojik faktörler ve (3) ders etkinlikleri ve materyalleri bağlamında gerçekleştirilmiştir.

5.1.2.1. Öğretim programı kategorisindeki problemlerin çözümüne ilişkin sonuç ve tartışma

Oyun geliştirmeye dayalı öğretim yönteminin kullanıldığı bu çalışmanın her iki döngüsünde de öğrencilerin NYP bilgi ve becerileri, öğretim öncesine göre anlamlı ve yüksek düzeyde artmıştır (Tablo 4.22 ve Tablo 4.43). Öğrencilerin NYP bilgi ve becerilerinde gerçekleşen artış, oyun geliştirmeye dayalı programlama öğretiminin NYP teknik ve yöntemlerinin öğrenilmesine katkıda bulunduğu sonucuna ulaşan alanyazındaki

birçok çalışmayla uyumludur (Brito ve Medeiros, 2019; Tomaiuolo, Angiani, Ferrari, Mordonini ve Poggi, 2018). İlave olarak birinci ve ikinci döngüde katılımcıların elde ettikleri akademik başarıları karşılaştırılmıştır. Katılımcıların her iki döngüde de NYP Başarı Sınavından elde ettikleri puanların karşılaştırılmasına göre, iki döngünün öntest puan ortalamalarında istatistiksel olarak anlamlı bir fark bulunmazken (Tablo 4.65) sontest puan ortalamalarında anlamlı bir fark tespit edilmiştir (Tablo 4.66). Bu durumun ortaya çıkmasında; birinci döngü sonrasında gerçekleştirilen revizyonların ve ikinci döngünün birinci döngüye göre daha uzun bir öğretim sürecini kapsamasının etkili olabileceği değerlendirilmektedir. Bu bulgunun ortaya çıkmasında, öğrencilerin NYP kavramlarını öğrenebilmeleri ve bunları yazılım geliştirmede kullanabilmek maksadıyla zihinsel modellerinin oluşması için yeterli bir süreye ihtiyaç duymaları etkili olmuş olabilir (Ragonis, ve Shmallo, 2021).

Bu tez çalışmasının NYP Başarı Sınavının uygulama kısmıyla ilgili sonuçları incelendiğinde, katılımcıların teorik sınav puanlarıyla uygulama sınavı puanları arasında pozitif yönde ve yüksek düzeyde bir ilişki bulunmuştur (Tablo 4.25 ve Tablo 4.46). Bu durum bilişsel düzeyde edinilen bilgilerin beceri düzeyinde de kullanılabildiğini göstermektedir. Benzer şekilde öğrenciler, derste öğrendikleri NYP mekanizmalarını geliştirdikleri bireysel oyun prototiplerine puan sistemi, bonus ve engel sınıfları, ses sistemi, bildirim sistemi gibi yeni bileşenler ekleme aşamalarında kullanmışlardır (Şekil 4.51 ve Şekil 4.55). Bu sonuçlar, alanyazındaki çalışmalarla uyumlu olarak oyun geliştirmeye dayalı öğretim etkinlikleri vasıtasıyla öğrencilerin sınıf oluşturma, nesneleri kullanma, kalıtım ve çokbiçimlilik gibi konularda programlama becerisi kazandıklarını göstermektedir (Chatain, Bitter, Fayolle, Sumner ve Magnenat, 2019; López ve diğerleri, 2020; Panskyi ve Rowinska, 2021). Öğrencilerin bu becerileri kazanmalarında; öğrencilerin oyun geliştirme sürecinde aktif bir rol oynamasının ve bu sürecin bir yaparak öğrenme deneyimi barındırmasının önemli bir unsur olduğu değerlendirilmektedir (Theodoropoulos ve Lepouras, 2022). Öğrencilerin NYP bilgi ve becerilerindeki gelişim; meslek yüksekokullarında benzer isimlerde okutulan NYP dersleri için anlamlı bir ilerlemeyi ifade etmekle birlikte bilişim teknolojileri veya bilgisayar mühendisliği öğretiminde de NYP derslerinin verimliliğini artırabileceği söylenebilir.

Çalışmanın sonuçları kazanım bazında incelendiğinde, "*Nesne kavramını örneklerle açıklar." (K2)* kazanımının başarı yüzdesi birinci döngüde %86 (Şekil 4.27), ikinci döngüde %97 (Şekil 4.49) olarak gerçekleşmiştir. Böylelikle birinci döngüde katılımcıların büyük bir

kısmının; ikinci döngüde ise neredeyse tamamının nesne kavramını kavradıkları ortaya çıkmıştır. Ancak Tomaiuolo ve diğerlerinin (2018) çalışmasında ise öğrencilerin nesne kavramını anlamakta sorun yaşadığı sonucuna ulaşılmıştır. Her ne kadar Tomaiuolo ve diğerlerinin (2018) çalışmasında da bu tez çalışmasında olduğu gibi öğrenciler tüm mekanik ve dinamiklerinin nesnelerden oluştuğu bir oyun geliştirmiş olsalar da, bu farklılığın ortaya çıkma nedenleri incelemeye değerdir. Bu uyumsuzluğun temel sebepleri arasında Tomaiuolo ve diğerlerinin (2018) çalışmasının katılımcı grubunun daha önce programlama dersi almamış bireylerden oluşması ve oyun geliştirme etkinliklerinin bir hafta sonu süren bir çalışma kampında gerçekleştirilmiş olması yer alıyor olabilir. Nitekim bu durumdan farklı olarak bu tez çalışmasının her iki döngüsünde de yer alan katılımcıları daha önce iki tane programlama dersi almış olmaları bakımından programlama konusunda tecrübeli sayılabilecek durumdadırlar. İlave olarak katılımcılar birinci döngüde sekiz hafta; ikinci döngüde ise 15 haftalık bir öğretim faaliyetine katıldıkları için daha uzun süren bir proje geliştirme sürecine dâhil olmuşlardır.

Kazanım bazında değerlendirme kapsamında çalışmanın sonuçlarına göre "*Verilen UML Sınıf diyagramlarında yer alan sınıflara ait özellik ve davranışlarını çözümleyerek bu sınıflar arasındaki ilişkileri açıklar." (K12)* kazanımında öğrencilerin başarı yüzdeleri birinci döngüde %90 (Şekil 4.27), ikinci döngüde %97 (Şekil 4.49) olarak gerçekleştiği görülmektedir. Bu bulgu, Al-Fedaghi'ye (2021) göre nesne yönelimli sistemleri modellemede en yaygın yöntem olarak kullanılan ve yazılım belirtimi ile gerçekleştirme arasında bir "köprü" görevi gören UML sınıf diyagramlarının öğrencilerin büyük bir çoğunluğu tarafından öğrenildiğini göstermektedir. Bu sayede alanyazında programlama derslerinde UML sınıf diyagramlarına yeterince başvurulmadığı probleminin (Reuter ve diğerleri, 2020) çözümüne yönelik bir adım atılmıştır.

UML sınıf diyagramlarının kullanılmasının, birinci döngüde diğer kazanımlara göre görece düşük çıkan ve revizyona başvurulan bazı kazanımların başarı yüzdelerinin ikinci döngüde artmasını sağlamış olabileceği söylenebilir. Çalışmanın bulgularına göre *Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar." (K8)* ve "*Kapsüllemenin (encapsulation) kullanım amaçlarını örnek vererek açıklar." (K9)* kazanımlarının başarı yüzdelerinde, ikinci döngüye eklenen "*Çokbiçimlilik Uygulaması*" ve "*UML Sınıf Diyagramı Analizi*" etkinlikleri sonucunda sırasıyla 17 ve 27 puanlık bir artış görülmüştür (Şekil 4.49). Bu etkinlikler vasıtasıyla öğrenciler oyun prototiplerinin hangi sınıfları içereceği, bu sınıfların hangi özellik ve davranışlara sahip olacağı, oyunda kalıtım

mekanizmasının nasıl işleyeceği, sınıfların hangi çokbiçimli metotlara sahip olacağı gibi unsurları yorumlama şansına sahip olmuşlardır. Bu durumun grafiklerin sözcüklerden çok daha fazla anlam içerebileceği prensibiyle öğrencilerin zihinsel modellerini oyun prototiplerine aktarmalarını kolaylaştırdığı söylenebilir.

Bu çalışmanın her iki döngüsünde de öğrencilerin daha önce aldıkları Algoritma ve Programlamaya Giriş ile Görsel Programlama derslerindeki başarı durumlarının NYP dersindeki başarılarını anlamlı derecede etkilediği görülmüştür. Nitekim önceki programlama derslerinde yüksek başarılı olan öğrenciler NYP dersinde de yüksek başarılı olmayı sürdürmüştür (Tablo 4.33 ve Tablo 4.53). Bu bulgu, öğrencilerin önceki programlama derslerindeki başarı durumunun mevcut derslerindeki başarıları için kritik öneme sahip olduğu sonucuna ulaşan alanyazındaki çalışmalarla uyumludur (Denner, Werner ve Ortiz, 2012; Garcia Ramirez, 2022; Toukiloglou ve Xinogalos, 2022). İlave olarak çalışmanın bulguları incelendiğinde, önceki programlama derslerinde yüksek başarılı öğrencilerin oyun prototiplerine ders videolarında belirtilenlerin haricinde farklı bonus ve engel sınıfları ekledikleri görülürken, düşük başarılı öğrenciler dersin asgari gerekliliklerini yerine getirmekle yetindikleri ortaya çıkmıştır. Benzer sonuçlar Brito ve Medeiros'un (2019) gerçekleştirdiği çalışmada da bulunmuştur. Brito ve Medeiros'un (2019) çalışmasının sonuçlarına göre programlama konusunda daha düşük başarılı öğrenciler üç boyutlu nesneleri sadece önceden öğretim elemanı tarafından tanımlanmış metotları kullanarak oluştururken yüksek başarılı öğrenciler ise kendi metotlarını tanımlamayı başarabilmişlerdir. Bu açıdan değerlendirildiğinde geçmiş dönemlerde aldıkları derslerde düşük başarılı olan öğrencilerin diğerlerinin seviyesine ulaşabilmeleri için teknik destek sağlamanın, gelişimlerini takip ederek onlara geri bildirim sunulmasının önemli olduğu değerlendirilmektedir.

Bu tez çalışmasının sonuçlarına göre her ne kadar öğrencilerin NYP bilgi ve becerilerinin gelişmiş olsa da, öğrencilerin bazı konularda zorlandıkları da ortaya çıkmıştır. Öğrencilerin yaşadıkları zorlukların ilki prosedürel programlamadan NYP'ye geçişte yaşanan güçlüklerle ilgilidir. Geliştirilen NYP dersinin katılımcı grubu daha önce prosedürel programlamayla ilgili iki ders aldıkları için zorunlu olarak NYP dersini "daha sonra nesne" yaklaşımıyla almışlardır. Bu durumun bir neticesi olarak daha önce alıştıkları form uygulamalarında gerçekleştirdikleri olay tabanlı programlamanın aksine NYP dersinde geliştirdikleri algoritmaların merkezine nesneyi koymak durumunda kalmışlar; gerçekleştirdikleri oyun projelerinde oyun nesnelerinin davranışlarını değiştirmek suretiyle

oyun mekanizmalarını yönlendirmişlerdir. Bu sürece alışmak; hangi sınıfı tanımlayacaklarını; hangi oyun nesnesine ne tür özellik ve davranışlar kazandıracaklarını belirlemek; aynı zamanda bunu C# kodlarıyla programlamak bazı öğrenciler için zorlayıcı olmuştur. Nitekim bu bulguyla uyumlu olarak, alanyazında prosedürel programlamadan NYP paradigmasına geçişte öğrencilerin güçlük yaşadığı ve NYP düşünme tarzını benimsemeleri için belirli bir tecrübe kazanma sürecine ihtiyacı olduklarını belirten çalışmalar bulunmaktadır (Abbasi ve diğerleri, 2017; Akkaya, 2018). Bu açıdan değerlendirildiğinde; NYP paradigması içinde yer alan kavram ve tekniklerin anlaşılabilirliğini kolaylaştırmak için öğrencilere çözülmüş örneklerin, sınıflar arasındaki ilişkilerin betimlendiği UML diyagramlarının ve canlı kod gösterimlerinin sunulmasının önemli olduğu değerlendirilmektedir.

Öğrencilerin yaşadıkları bir diğer güçlük sınıf metotlarının geliştirilmesi ve kullanılmasıyla ilgilidir. Bu çalışmada bazı öğrenciler oyun prototiplerini geliştirmeleri esnasında C#'ta metotlar, metot parametrelerinin tanımlanması, metotların çağrılması, sınıf metotlarının tanımlanması ve Monobehavior metotlarının kullanılması gibi konularda zorlanmışlardır. Benzer sonuçlar alanyazındaki çalışmalarda da belirtilmiştir (Wong ve diğerleri, 2017; Miller ve diğerleri, 2015). Her ne kadar metotların nasıl tanımlanacağı ve çağrılacağı, metot parametrelerinin nasıl belirleneceği gibi konular NYP dersinin öğretim programında yer almayıp prosedürel programlamanın konuları dâhilinde bulunsa da; bu konuların önceki programlama derslerinde tam olarak öğrenilmemesi NYP dersinin başarısını olumsuz etkileyebilecektir. Nitekim Tomaiuolo ve diğerlerinin (2018) gerçekleştirdiği çalışmada NYP dersini alan öğrenciler için en zorlayıcı konunun metot parametreleri ve yerel değişkenler olduğu sonucunun ortaya çıkması, programlamanın temel konularının NYP kazanımlarında etkin bir rol oynadığını vurgular niteliktedir.

Bu çalışmanın sonunda öğrencilerin kurucu metot konusunda zorluk yaşadıkları görülmüştür. Bu zorluk özellikle kurucu metotla sınıf metotlarının (davranışlar) birbirine karıştırılmasından kaynaklanmıştır. Birinci döngü sonunda bu durumla ilgili revizyonlara başvurulmasına rağmen ikinci döngüde de diğer konulara göre kurucu metot konusu öğrenciler açısından zorlayıcı olmuştur. Bunun olası nedenlerinin birisinin Unity 3D'nin bileşen tabanlı (component-based) mimariyi kullanması olabilir. Unity 3D'de yer alan bileşen tabanlı mimari, oyun geliştirme maliyetlerini en aza indirebilmekte ve oyun geliştirme süresini azaltmakla birlikte alışılagelmiş NYP uygulamalarından kurucu metot kullanımı konusunda ayrışmaktadır (Holopainen, 2016). Bu mimaride bir oyun nesnesi için

tanımlanan diğer her davranış, bileşen (component) adı verilen programlama komut dosyaları tarafından uygulanan bir kompozisyon rutinine atfedilmektedir (Holmstedt ve Mengiste, 2017). Ancak bu durum sınıflardan nesne türetmeyi new operatörü ile yapmak yerine Monobehaviour metotlarından olan Awake() ve Start() metotlarıyla yapmayı gerekli kılmaktadır. Nitekim Unity 3D'de kurucu metotların kullanılması önerilmemektedir (Unity, 2022). Öğrencilerin birinci döngü sonunda kurucu metot konusunda güçlük yaşadıkları tespit edilmesi üzerine ikinci döngüde bu konuyla ilgili revizyonlar gerçekleştirilmiş; her ne kadar kullanılması önerilmese de öğrencilere new operatörü ile nesne üretmeyi içeren ilave uygulamalar yaptırılmıştır. Bu sayede kurucu metot konusunda daha az öğrenci güçlük yaşamıştır.

Geliştirilen NYP dersinde öğrencilerin zorlandıkları bir diğer husus çokbiçimlilik konusudur. Alanyazında belirtildiği gibi çokbiçimlilik konusu NYP'nin nispeten ileri seviye konularının arasında yer alarak öğrenciler açısından zorlayıcı olabilmektedir (Guenaga, Eguíluz, Garaizar ve Gibaja, 2021; Sunday ve diğerleri, 2022). Nitekim ikinci döngüde bu konuyla ilgili gerçekleştirilen örnek uygulamalar ve detaylı açıklamalar bu konuda daha çok öğrencinin başarılı olmasını sağlamıştır. Çalışmanın sonucunda bazı öğrencilerin kalıtım konusu anlamakta da güçlük yaşadığı görülmüştür (Şekil 4.49). Benzer sonuçlar alanyazındaki çalışmalarda da (Abbasi ve diğerleri, 2021; Lian, Varoy ve Giacaman, 2022). Bu güçlükleri en aza indirebilmek için kalıtım hiyerarşileri UML diyagramları üzerinden açıklanmalı; süper sınıfların sahip olduğu hangi özellik ve davranışların alt sınıflara aktarılacağı ve bunların hangilerinin alt sınıflar tarafından kullanılabileceği detaylı bir şekilde açıklanmalıdır.

Çalışmanın birinci döngüsünde bazı katılımcıların oyun prototiplerini geliştirirken projelerine nasıl devam edecekleri konusunda fikir yürütememişler, oyun prototipinin sınıf modelini zihinlerinde kurgulayamamaları sebebiyle sınıf tasarımlarını gerçekleştirememişler ve projelerinin başlarında çabalamalarına rağmen projelerini geliştirmekten vazgeçmişlerdir. Başka bir ifade ile birinci döngünün üç katılımcısı karşılaştıkları güçlükler sebebiyle oyun geliştirme konusunda pes etmişlerdir. Bu bulgu, NYP'nin yeni başlayanlar için oldukça zorlayıcı olduğu; yeni başlayanların karşılaştıkları zorluklar karşısından sıklıkla bunalabilecekleri ve programlamaya karşı ilgililerini kaybedebilecekleri sonucuyla uyumludur (Poolsawas ve diğerleri, 2016; Tanielu, 'Akau'ola, Varoy ve Giacaman, 2019). Bu doğrultuda NYP paradigmasını öğrenmeye yeni başlayan öğrencilerin benzer problemleri yaşamamaları için öğrenciler sıklıkla desteklenmeli, gelişim

aşamaları takip edilmeli ve karşılaştıkları hataların çözümleri konusunda kendilerine yardımcı olunmalıdır. Nitekim çalışmanın ikinci döngüsünde öğrenciler daha fazla takip edilmiş; karşılaşılan hatalarla ilgili daha fazla açıklamalar yapılarak öğrencilerin projeleri üzerinde çalışmaya devam etmeleri teşvik edilmiştir. Bunların neticesinde projesini tamamlayamayan öğrenci olmamıştır.

Çalışmanın sonunda bazı öğrencilerin hata mesajlarının ne anlama geldiğini bilmedikleri, nasıl çözmeleri gerektiği konusunda fikir yürütmekte zorlandıkları görülmüştür. Bu durumu yaratan nedenlerden birisi oyun geliştirmekte kullanılan IDE'lerin profesyoneller için geliştirilmesinden dolayı NYP'ye yeni başlayanlar için detaylı açıklamalar içeren ve onları yönlendiren hata mesajlarını barındırmamasından kaynaklanmaktadır (Watson, Li ve Lau, 2011). Bu güçlüğü aşmak için derslerde öğrencilere hataların nedenlerinin detaylı bir şekilde açıklanarak olası çözüm yollarının gösterilmesinin önemli olacağı değerlendirilmektedir. Karşılaşılan hataların anlaşılmamasının bir diğer nedeni ise öğrencilerin yaşadıkları yabancı dil bariyeriyle ilgilidir. Çalışmanın sonucunda bazı öğrencilerin yabancı dil sorunu yaşadıkları; oyunlarını geliştirirken kullandıkları programlama dilinin ve Unity 3D editörünün yabancı dilde geliştirilmiş olması nedeniyle kavramları anlamakta güçlük çektiği görülmüştür. Bu sonuç alanyazınla uyumlu olarak yabancı dil bariyerinin öğrencilerin programlamayı öğrenmelerinde güçlüklere neden olabileceğini göstermektedir (Groher ve diğerleri, 2022) Her ne kadar günümüzde bazı IDE'ler ana dil desteği sağlasa da Unity 3D'de böyle bir destek henüz bulunmamaktadır. Bu açıdan değerlendirildiğinde gerek Unity 3D'de yer alan terimlerin gerekse NYP yapılarının Türkçe karşılıklarının açıklanmasının öğrencilerin bu kavramları daha hızlı öğrenmelerini sağlayacağı değerlendirilmektedir.

Son olarak; NYP öğretimiyle ilgili sonuçlara ilave olarak öğrencilerin oyun programlama becerilerinde anlamlı bir ilerleme görülmüştür. Tasarlanan NYP dersi İstanbul Medipol Üniversitesi, Nişantaşı Üniversitesi, İzmir Ekonomi Üniversitesi, Süleyman Demirel Üniversitesi gibi Türkiyedeki birçok üniversitenin meslek yüksekokulunda ayrı bir ders olarak okutulan oyun programlama derslerine (Sunday ve diğerleri, 2022) bir alternatif olmamakla birlikte NYP öğretim sürecinin sonunda öğrenciler bir Unity 3D gibi profesyonel bir oyun motorunu kullanma; oyun mekanikleri, dinamikleri, oyun fiziği gibi oyun programlamayla ilgili kavramlara aşina olmuşlar ve oyun programlamayla ilgili becerilerini geliştirmişlerdir. Öğrenciler bu oyun geliştirme deneyimini ve becerilerini NYP dersinin tek dönemlik bir ders olmasına rağmen başarabilmişlerdir. Bu durumu, katılımcılara ait Unity

3D Başarı Sınavı puanlarının öntest-sontest karşılaştırma sonuçları teyit etmiştir (Tablo 4.56). Bu sonuç, Akkaya ve Akpinar'ın (2022) belirttiği gibi NYP dersinin oyun programlama dersine katkı sağlayacağı yönündeki sonuçlarıyla uyumludur. Ancak oyun programlama dersinin zaman kısıtlaması nedeniyle bu çalışmanın yürütüldüğü meslek yüksekokulundaki bilgisayar teknolojisine dâhil edilememesinden dolayı oyun programlamanın tüm boyutları NYP dersine entegre edilememiştir. Öğretim programının imkân vermesi koşuluyla NYP dersine ilave olarak oyun programlama dersinin ayrı bir ders olarak okutulmasının; öğrencilerin oyun geliştirme platform ve tekniklerinin kullanımına yönelik becerilerini artıracağından dolayı NYP dersinin daha verimli işlenmesini sağlayacağı değerlendirilmektedir. Böylelikle NYP dersinin öğretim programında yer alan Unity 3D editörü tanıtımı ve Monobehaviour metotlarının kullanımı gibi konular oyun programlama dersinde gösterilecek; NYP dersinde ise sadece NYP tekniklerine yönelik programlama uygulamaların gerçekleştirilmesi sağlanmış olacaktır.

5.1.2.2. Psikolojik faktörler kategorisindeki problemlerin çözümüne ilişkin sonuç ve tartışma

Psikolojik faktörler kategorisindeki problemler; öğrencilerin NYP dersine karşı motivasyonu, programlama öz yeterlilik algıları ve programlama kaygıları bağlamında incelenmiştir. Bu tez çalışmasının nitel çözümleme tekniklerinden elde edilen sonuçlarına göre, çalışmanın birinci döngüsünde öğrencilerin %69'u oyun geliştirmeye dayalı programlama öğretiminin motive edici olduğunu belirtmiştir (Şekil 4.38). Benzer şekilde çalışmanın ikinci döngüsünde öğrencilerin %90'ı kullanılan yöntemi motive edici bulmuştur (Şekil 4.69). Her iki döngüde de katılımcıların çoğunluğunun derse karşı ilgili ve motive olmaları, araştırmacının gözlemleriyle de teyit edilmiştir. Bu bağlamda değerlendirildiğinde, oyun geliştirmeye dayalı öğretim yöntemi kullanmanın öğrencilerin programlamaya yönelik motivasyonlarına olumlu katkılar sunduğu söylenebilir. Öğrencilerin motivasyonlarındaki artışa birkaç unsur neden olmuş olabilir.

Öğrencilerin bu süreçte motivasyonlarının artmasına neden olan ilk unsur; oyun geliştirmeye dayalı öğretimin öğrencilerin ilgisini çekmesi olabilir (Şekil 4.38 ve Şekil 4.69). Nitekim alanyazındaki çalışmalarda da oyun geliştirmenin öğrencilerin ilgilerini çekerek onları derse karşı motive ettiği belirtilmektedir (Akkaya ve Akpinar, 2022; Tomaiuolo, Angiani, Ferrari, Mordonini ve Poggi, 2018). Öğrencilerin derse karşı ilgilerinin çekilmesi oldukça önemlidir çünkü öğrenci ilgisi ders başarısının temel unsurlarından birisidir (Sabri, Fakhri ve Moumen, 2022). İkinci döngünün sonunda gerçekleştirilen odak grup

görüşmesinden elde edilen çözümlemelere göre öğrencilerin oyun geliştirme yoluyla derse karşı ilgilerinin çekilmesi, öğrencilerin ders dışı zamanlarda da dersle ilgili çalışmalar yapmalarını sağlamıştır. Bu odak grup görüşmesinde katılımcılar kendi düşüncelerini oyun prototiplerine aktarabilmek için saatlerce gerekli teknikleri araştırdıklarını; yeni paketleri ve kodları incelediklerini ve karşılaştıkları hataları düzeltmek için uğraş verdiklerini belirtmişlerdir. Bu durumu öğrencilerin araştırmacıya sundukları oyun projelerinde farklı sınıf, nesne ve Unity 3D paketlerini kullanmış olmaları teyit etmektedir (örn. Şekil 4.52). Bu durumun bir diğer göstergesi ise ikinci döngü sonunda iki katılımcının kendi oyun prototiplerine ilave olarak farklı bir oyun geliştirip araştırmacıyla paylaşmalarıdır (Şekil 4.62 ve Şekil 4.63). Tüm bu bulgular değerlendirildiğinde, oyun geliştirmeye dayalı öğretimin öğrencilerin ilgilerini çekerek derse daha fazla çalışmalarını sağladığını sonucuna ulaşılabilir. Nitekim alanyazındaki çalışmalar da bu durumu teyit etmekte olup oyun programlamanın öğrencilerin programlama derslerine daha fazla çalışmalarını sağladığını belirtmektedir (Bewer ve Gladkaya, 2022).

Çalışmanın bulgularına göre öğrencilerin yazdıkları kodların kendi oyun projelerinde ne tür değişikliklere neden olduğunu hemen görebilmeleri, motivasyonlarını artıran bir diğer unsur olmuştur (Tablo 4.57). Benzer sonuçlar projeleri üstünde gerçekleştirdikleri işlemlerin hangi değişikliklere neden olduğunu görmenin öğrenciler açısından önemli ve motive edici olduğu sonucuna ulaşılan López, Duarte ve Gutiérrez'in (2020) çalışmasında da bulunmuştur. İlave olarak gerçekleştirilen odak grup görüşmesinde öğrenciler üç boyutlu bir evrende geliştirdikleri oyun projelerinde kendi ürünlerinin gelişim süreçlerini takip edebilmelerinin; yeni eklentilerle oyunlarına farklı mekanizmalar kazandırabilmelerinin motive ederek oyun projeleri üzerinde daha fazla çalışmaya teşvik ettiğini belirtmişlerdir. Benzer sonuçlar Brito ve Medeiros'un (2019) çalışmasında da ortaya çıkmıştır. Bu bağlamda öğrencilerin kendi projelerinin adım adım geliştiğini gözlemlemesinin NYP dersi için motive edici bir unsur olduğunu göstermektedir.

Öğrencilerin birinci ve ikinci döngü sontest motivasyon puanları kıyaslandığında, ikinci döngü lehine istatistiksel olarak anlamlı bir fark bulunmuştur (Tablo 4.69). Her iki döngüde de farklı versiyonları olsa da aynı oyun prototipinin kullanılmış olması, oyun prototipinin motivasyon açısından bir farklılığa neden olmadığını göstermektedir. Ancak iki döngü arasındaki farklılık ikinci döngünün birinci döngüye kıyasla daha uzun sürmesi ve ikinci döngünün öğretim programında daha fazla konunun yer almasından kaynaklanıyor olabilir. Nitekim ikinci döngüde öğrenciler daha fazla kazanım elde ederek bunları oyun projelerinde

uygulama başarısını elde etmişlerdir. Oysa birinci döngü için ayrılan süre katılımcılar tarafından yetersiz bulunmuş ve bu durumun öğrencilerin proje geliştirme süreçlerini olumsuz etkilediği tespit edilmiştir (Şekil 4.38). Birinci döngüde zaman kısıtlamasının olması, öğrencilerin projelerini tamamlamak için ne kadar süreye ihtiyaçları olduğunu tam olarak tespit edememelerine neden olarak motivasyonlarını düşüren bir unsur olmuş olabilir.

Nitel bulgulara zıt olarak, çalışmanın ikinci döngüsünde nicel çözümleme teknikleriyle elde edilen veriler, istatistiksel olarak anlamlı olmasa da şaşırtıcı bir şekilde öğrencilerin motivasyonlarının süreç içinde düştüğünü göstermektedir (Şekil 4.64). Bu durumun ortaya çıkmasının sebeplerinin başında, öğrencilerin ikinci döngüde NYP dersini uzaktan alması olabilir. Çalışmanın ikinci döngüsünde uygulanan acil uzaktan öğretim yöntemi, araştırmacı-öğrenci ve öğrenci-öğrenci bağlamında sınırlı bir etkileşim barındırdığı; uygulamaları laboratuvar ortamından ziyade bireysel olarak evde yapmak zorunluluğu taşıdığı için öğrencilerin motivasyon puanlarını olumsuz etkilemiş olabilir. Nitekim ikinci döngü sonunda gerçekleştirilen odak grup görüşmesinde bazı katılımcıların motivasyonlarının "*anlık iletişim yetersizliği*", "*uzaktan öğretimin konsantrasyonu olumsuz etkilemesi*", "*ders işliyormuş gibi hissedememek*" ve "*yüz yüze öğretime göre derse daha fazla zaman ayırmak zorunda kalmak*" gibi nedenlerle olumsuz etkilendiği tespit edilmiştir (Tablo 4.62). Bunun bir diğer nedeni ise oyun geliştirmenin öğrencilerin alışkın oldukları form uygulamalarına göre yeni bir alan olması nedeniyle uygulamanın başlarında öğrencilerin oyun geliştireceklerinden dolayı motivasyonlarının yüksek olup zaman içinde bu uygulama türüne alışarak motivasyonlarında azalma yaşanmış olabileceğidir. Bu durum yenilik etkisinden (novelty effect) ileri gelmiş olabilir. Nitekim alanyazındaki çalışmalar yeni bir teknoloji ile karşılaşılması durumunda bireylerin ilgilerinin artabileceğini ancak zamanla azalabileceğini göstermektedir (Huang, 2020; Tsay, Kofinas, Trivedi ve Yang, 2020).

Motivasyona ilave olarak, çalışmanın sonucunda katılımcıların öz yeterlilik algılarına yönelik sonuçlar da elde edilmiştir. Programlama alanyazını incelendiğinde, öz yeterlilik algısıyla ilgili gerçekleştirilen çalışmaların sonuçlarının uyumlu olmadığı ve birbiriyle çeliştiği görülmektedir. Bu tez çalışmasında da oyun geliştirmeye dayalı öğretimin öğrencilerin öz yeterlilik algılarını farklı şekilde etkilediği ortaya çıkmıştır. Her iki döngüdeki öğretim süreci katılımcıların bazılarının programlama öz yeterlilik algılarını olumlu etkilerken düşük sayıda da olsa bazı katılımcılarınkini olumsuz etkilemiş; bazılarının öz yeterlilik algılarında ise herhangi bir etkiye neden olmamıştır.

Öncelikle belirtmek gerekir ki çalışmanın sonuçlarına göre her iki döngüde de katılımcıların çoğunluğunun programlama öz yeterlilik algıları oyun geliştirmeye dayalı öğretim stratejisinden olumlu etkilenmiştir. Bu durumu birinci döngü sonunda gerçekleştirilen odak grup görüşmesinden elde edilen veriler ile ikinci döngü sonunda hem nitel hem de nicel çözümleme teknikleriyle elde edilen sonuçlar teyit etmektedir (Tablo 4.59, Şekil 4.65). Programlama öz yeterlilik algıları olumlu etkilenen katılımcılar, NYP dersinde edindikleri bilgi ve becerileri kendilerine ait bir oyunu geliştirirken, proje geliştirme aşamasında güçlükleri aşarken ve hatalarını bireysel olarak düzeltirken kullanabilmelerinin başarmışlık hislerine katkı sağladığını belirtmişlerdir. Bu başarmışlık hissinin katılımcıların programlama öz yeterlilik algılarını olumlu etkilemiş olabileceği düşünülebilir. Bu sonuçlar alanyazındaki bazı çalışmalarla örtüşmektedir (De Oliveira ve diğerleri, 2017; Wong, Liu, Yin, Yang ve Chao, 2016).

Olumlu etkilere zıt olarak bazı katılımcıların programlama öz yeterlilik algıları oyun geliştirmeye dayalı öğretim sürecinden olumsuz etkilenmiştir (Tablo 4.59). Bu katılımcılar oyun geliştirme sürecinde programlama becerileri konusunda kendilerini sorgulamış; benzer şekilde arkadaşlarının seviyesine ulaşamayacaklarına yönelik inançlara sahip olmuşlardır (Tablo 4.59). Bu sonuçlar, Leonard ve diğerlerinin (2016) gerçekleştirdiği çalışmanın sonuçlarıyla uyumludur. Leonard ve diğerlerinin (2016) çalışmasında da katılımcıların öz yeterlilik algıları MINDSTORMS programlama etkinliklerinin karmaşıklığından ve sıfırdan bir oyun tasarlamanın güçlüklerinden olumsuz etkilenmiştir.

Bu çalışmanın bulgularına göre ikinci döngüde bir katılımcının öz yeterlilik algısı oyun geliştirmeye dayalı öğretim sürecinden etkilenmemiştir. Bu sonuç Korkmaz (2016) ve Seaborn, Seif El-Nasr, Milam ve Yung'un (2012) çalışmalarıyla örtüşmektedir. Bu tez çalışmasında katılımcıların çoğunun programlama öz yeterlilik algılarında olumlu değişimler yaşanırken Korkmaz (2016) ve Seaborn ve diğerlerinin (2012) çalışmasında etkilenmemiş olmasının nedenleri incelenmeye değerdir. Bu farklılık bu çalışmalarda yer alan katılımcı grubunu ve oyun geliştirmek için kullanılan araçların karakteristiklerinden kaynaklanıyor olabilir. Nitekim Korkmaz'ın (2016) çalışmasında öğrenciler Scratch kullanarak oyun geliştirmişlerdir ancak bu oyun geliştirme sürecinin öğrencilerin C++ öz yeterlilik algılarına etkisi incelenmiştir. Bu kapsamda; blok tabanlı bir oyun geliştirme sürecinde öğrencilerin kullandıkları programlama yapılarının ve karşılaştıkları hataların sınırlı olabileceği değerlendirildiğinde, Scratch kullanımının C++ programlama dili özelinde öğrencilerin programlama öz yeterlilik algılarını etkilememesi şaşırtıcı değildir. Oysa bu tez

çalışmasında oyun geliştirmek için profesyonel bir programlama dili olan C# kullanılmış; katılımcıların bu süreçte hatalarla karşılaşmış ve bunları gidermek için çaba sarf etmiştir. Benzer şekilde Seaborn ve diğerlerinin (2012) çalışmasında yer alan katılımcı grubu daha önce herhangi bir programlama dersi almamış öğrencilerden oluşmuştur. Bu durum katılımcıların programlama bilgi seviyelerini tam olarak değerlendirmemelerine neden olmuş ve bu sebeple programlama öz yeterliliklerini değerlendirmede güçlük yaşamışlardır. Ancak bu tez çalışmasındaki öğrenciler daha önce iki tane programlama dersi aldıkları için programlama konusunda yeterince deneyim sahibi olup kendi programlama bilgi ve beceri seviyelerinin farkındadır.

Bu çalışmanın sonucunda alanyazınla uyumlu olarak öğrencilerin programlama öz yeterlilik algılarıyla programlama dersine yönelik motivasyonları arasında pozitif ve istatistiksel olarak anlamlı bir ilişki bulunmuştur (Tsai, 2019; Yukselturk ve Altiok, 2017). Bu durum öğrencilerin sahip oldukları beceriler ile kendilerine olan inançlarının fazlasıyla iç içe geçmiş olmasından dolayı programlama derslerinde öğrencilerin performanslarını yükseltmenin bir yolunun da öğrencilerin öz yeterlilik algılarını geliştirmek olduğunu göstermektedir (Wiedenbeck ve diğerleri, 2004).

Bu çalışmanın sonucunda öğrencilerin programlama öz yeterlilik algılarıyla NYP başarıları arasında anlamlı bir ilişki tespit edilmemiştir. Bu sonuç alanyazındaki bazı çalışmalarla zıtlık içermektedir (Quille ve Bergin, 2016). Her ne kadar alanyazında programlama öz yeterlilik algısı yüksek olan öğrencilerin daha fazla çabaladığı belirtilse de NYP akademik başarısı birden fazla faktörden etkilendiği için tüm çalışkan öğrencilerin daha yüksek notlar alacağı garanti değildir (Tek, Benli ve Deveci, 2018). Bu durum da Scott ve Ghinea'nın (2013) belirttiği gibi programlama öz yeterlilik algısı ile ders başarısının arasında neden sonuç ilişkisinin olmadığını göstermektedir. Nitekim Tek ve diğerlerinin (2018) çalışmasının sonuçları öğrencilerin programlama başarılarının öz yeterlilik algıları kullanılarak tahmin edilmesinin en azından tüm öğrenciler için anlamlı olmayacağını göstermektedir.

Psikolojik faktörler bağlamında son olarak programlama kaygısıyla ilgili bazı sonuçlara ulaşılmıştır. Bu çalışmanın sonucunda katılımcıların çoğunluğunun uygulamanın başlarında programlama kaygılarının arttığı ancak zaman içinde azaldığı; bazı öğrencilerin ise programlama kaygılarının değişmediği sonucuna ulaşılmıştır (Şekil 4.66 ve Tablo 4.60). Çalışmanın sonuçlarına göre öğrencilerin programlama kaygılarındaki artışa neden olan en önemli faktör, uygulamanın başlarında katılımcıların programlama becerilerine

güvenmemeleri olduğu ortaya çıkmıştır. Ancak zaman içinde gerek sınıf oluşturma ve nesnelere davranış kazandırma gibi NYP prensiplerini kavradıkça, gerekse Unity 3D'nin kullanarak oyun geliştirme konusunda tecrübe kazandıkça katılımcıların kaygı durumlarının azaldığı görülmüştür. Bu sonuç NYP dersinde tecrübe kazanmanın ve projelerde NYP paradigması konusunda zihinsel modellerin geliştirilmesinin öğrencilerin kaygılarının azaltılması için önemli olduğunu göstermektedir.

Çalışmanın sonucunda öğrencilere karşılaştıkları hatalar konusunda destek sağlamanın programlama kaygılarının azaltılmasına yardımcı olduğu bulunmuştur. Nitekim alanyazında da projelerinin geliştirilmesi esnasında öğrencilere yapılan yardımların programlama kaygılarını azalttığı belirtilmektedir (Falkner, Vivian, Piper ve Falkner, 2014; Maguire ve diğerleri, 2017). Bu çalışmanın sonucunda bazı öğrencilerin programlama kaygılarının devam ettiği ve yeni programlama ödev ve projelerinde devam edebileceği bulunmuştur (Tablo 4.60). Bu durum, öğrencilerin programlama becerilerine tam olarak güvenememeleri nedeniyle olabilir (Scott, 2015). Nolan ve Bergin'in (2016) belirttiği gibi her ne kadar aldıkları her programlama dersinde öğrenciler becerilerine yenilerini eklese de programlamanın yapısı gereği zor olmasından dolayı programlama becerileri konusunda yeterince kendilerine güvenmemekte ve mezun olduklarında programlama konusunda kaygılı bireyler olmaktadırlar.

Bu çalışmanın sonunda öğrencilerin programlama beceri kaygılarının programlama öz yeterlilik algıları ve başarılarıyla istatistiksel olarak anlamlı negatif bir ilişkiye sahip olduğu bulunmuştur. Bu sonuç, programlama kaygısının programlama performansı ve başarıyı etkileyen önemli faktörlerden olduğunu ve öğrencilerin öz yeterlilik algılarını olumsuz etkilediğini belirten çalışmalarla uyumludur (Askar ve Davenport, 2009; Kırcaburun, Baştuğ ve Bahtiyar, 2017).

5.1.2.3. Ders etkinlikleri ve materyalleri kategorisindeki problemlerin çözümüne ilişkin sonuç ve tartışma

Çalışmanın analiz aşaması bulgularına göre mevcut NYP dersinde öğrencilerin ders etkinlikleri kategorisinde yaşadıkları sorunların temelinde *günlük hayatla ilişkili, görsel olarak ilgi çekici, keyifli* ve *yaratıcı düşünmeyi teşvik eden* projelerin gerçekleştirilmemesi yatmaktadır (Tablo 4.5). Bu sorunların çözümü için NYP dersi oyun geliştirmeye dayalı öğretim stratejisini temel alarak yeniden tasarlanmış ve ders etkinlikleri mevcut bir oyuna eklentiler yapma bağlamında yeniden düzenlenmiştir. Bu çalışmanın bulgularına göre

çalışmanın birinci döngüsünün sonunda gerçekleştirilen odak grup görüşmesinde katılımcıların %62'si; ikinci döngüde ise %73'ü oyun geliştirmeye dayalı öğretimi keyifli bulmuşlardır (Şekil 4.38 ve Şekil 4.69). Benzer sonuçlar Stoffova'nın (2019) yaptığı çalışmada da ortaya çıkmıştır. Stoffova'nın (2019) yaptığı çalışmanın sonucunda da NYP'yi oyun geliştirerek öğrenen öğrenciler geliştirmeyen gruba göre daha fazla dersten keyif almışlar, daha fazla derse çalışmışlar ve derse yönelik fazla fikir üretmişlerdir.

Çalışmanın bulgularına göre araştırmanın her iki döngüsünde de öğrencilerin çoğunluğu nesne yönelimli programlamayı oyun geliştirerek öğrenmeyi yararlı bulmuştur (Şekil 4.41, Şekil 4.78). Nitekim her iki döngünün sonunda gerçekleştirilen odak grup görüşmelerinde katılımcılar oyun geliştirmeye dayalı öğretimi gerçek hayatla ilişkili bir etkinlik olarak gördüklerini, bunun da kendilerini motive ettiğini belirtmişlerdir. Alanyazındaki çalışmalar da oyun geliştirme projelerinin öğrenciler tarafından gerçek hayatla ilişkili bir süreç olarak algılanarak anlamlı öğrenmeye olanak tanıdığını; bunun da öğrencilerin motivasyonunu artırdığını belirtmektedir (Brito ve Medeiros, 2019; Paiva, Leal ve Queirós, 2020).

Çalışmanın bulgularına göre NYP öğretimi, öğrencilerin mesleki bilgi ve beceri kazanımları açısından da yararlı bulunmuştur (Tablo 4.57). Gerçekleştirilen odak grup görüşmelerinde öğrenciler oyun geliştirme konusunda ilk adımı bu ders sayesinde attıklarını ve oyun geliştirmeye devam edebileceklerini belirtmişlerdir. Bununla birlikte çalışmanın bulgularına göre NYP dersinin öğrencileri diğer bölüm derslerine çalışmak konusunda motive ettiğini ve yeni beceriler kazanmaları için teşvik ettiğini göstermektedir. Benzer sonuçlar Tervo (2019) ve Xu, Wolz, Kumar ve Greenburg'ün (2018) çalışmasında da bulunmuştur. İlave olarak ikinci döngünün sonunda gerçekleştirilen odak grup görüşmesinde NYP dersinde edindikleri bilgi ve becerilerin öğrencilerin meslek hayatlarında kendilerine avantaj sağlayabileceğini düşündükleri; bu düşüncelerin öğrencilerin programlamaya karşı olumlu görüşlere sahip olmalarını sağladığı görülmüştür. Bu bulgular, öğrencilerin programlamaya karşı olumlu tutum sergilemiş olabileceklerini gösteriyor olsa da bunun bir tutum ölçeği kullanılarak incelenmesinin faydalı olabileceği değerlendirilmektedir.

Çalışmanın bulgularına göre oyun geliştirmeye dayalı öğretim stratejisi, katılımcıların tasarım becerilerini özgün projelerini oluşturma sürecinde kullanmalarına olanak tanımıştır. Öğrencilerin oyun projelerinin tasarım özelliklerini değiştirmeye yönelik girişimleri, özellikle ikinci döngüde görülmüştür. Yapılan çalışmalarda öğrencilerin oyun projelerinde kendi tasarım becerilerini kullanmak istedikleri ve oyunlarını kendi fikirlerine göre

tasarlamak istedikleri görülmektedir. Wallinga'nın (2019) yaptığı çalışmada öğrenciler standart bir proje geliştirmiş ve bu durum öğrencilerin özgün fikirlerini kullanmalarını engelleyerek motivasyonlarını olumsuz etkilemiştir. Wallinga'nın (2019) çalışmasının aksine bu tez çalışmasında öğrenciler kendi özgün fikirlerini oyun projelerine aktarma şansına sahip olmuşlar ve kullan-değiştir-oluştur modeli bağlamında kendi projelerini oluşturmuşlardır. Böylelikle, mevcut NYP dersinde geliştirilen projelerde öğrencilerin tasarım becerilerini kullanmalarına olanak tanımaması sorununun çözümü için bir adım atılmıştır. Tüm bu bulgular değerlendirildiğinde, her ne kadar gerçekleştirilen oyun geliştirmeye dayalı programlama öğretiminin öğrencilerin yaratıcı düşünmelerine olanak sağlayabileceği düşünülse de bunun yaratıcı düşünce ölçekleriyle ölçülmesinin yararlı olabileceği değerlendirilmektedir.

Ders materyalleri sorun kategorisinde, mevcut NYP dersinde yeterince Türkçe kaynağın ve kod açıklamalarının bulunmaması problemi için çözümler üretilmiştir. Bu bağlamda ders videoları, sunumları ve kod açıklamaları hazırlanmıştır. Her iki döngü sonunda gerçekleştirilen odak grup görüşlerinden elde edilen görüşler doğrultusunda ders için geliştirilen proje demo videolarının öğrenciler tarafından faydalı bulunduğu sonucuna ulaşılmıştır. Bu sonuç Wallinga'nın (2019) belirttiği gibi ders kitaplarının öğrencilerin hayal güçlerini çekmekte sorunlu olduğu, onun yerine daha görsel ders materyalleri kullanmanın gerekliliği ile ilgili önerisiyle uyumlu olarak alanyazındaki statik ders materyallerinin öğrencilerin programlamaya çalışmasına yardımcı olmaması sorunun çözümüne yönelik bir adım olarak değerlendirilebilir. Bu ders videolarında NYP mekanizmalarının örnek kullanımlarına yer verilerek Herala, Vanhala ve Nikula'nın (2015) önerdiği gibi öğrencilerin geliştirdikleri projelerle ilgili detaylı açıklamalara yer verilmiştir. Bu süreç öğrencilerin NYP düşünme becerilerine ve zihinsel modellerinin gelişmesine yardımcı olmuştur.

5.2. Öneriler

Araştırmanın sonuçları doğrultusunda oyun geliştirmeye dayalı programlama öğretimi stratejisinin kullanılarak gerçekleştirilecek NYP öğretim süreci bağlamında araştırmacılara ve uygulayıcılara yönelik öneriler aşağıdaki maddelerde sunulmuştur.

- Bu çalışma kapsamında öğrenciler oyun prototiplerini bireysel olarak geliştirmişlerdir. Ancak alanyazında öğrencilerin projelerini takım olarak geliştirdikleri

çalışmalar bulunmaktadır. Gelecekte yapılacak çalışmalarda, NYP başarısı ve psikolojik faktörlerin takım çalışmasıyla gerçekleştirilmiş projelerde incelendiği çalışmalar yapılabilir.

- Bu çalışmada tek tip bir oyun projesi geliştirilmiştir. Ancak alanyazındaki bazı çalışmalarda gerçek zamanlı strateji, yarış oyunları, rol yapma oyunları gibi oyunların da öğrenciler tarafından gerçekleştirildiği görülmektedir. Bu yönüyle değerlendirildiğinde öğrencilerin geliştirmek istedikleri oyun türünü seçme imkânına sahip olmalarının programlama dersine yönelik motivasyonları üzerinde etkili olabileceği değerlendirilmektedir. Gelecekteki çalışmalarda öğrencilere geliştirmek istedikleri oyun türünü seçme imkânı verilebilir.

- Bu çalışmada kullanılan oyun prototipinin sanal gerçeklik (VR) türüne dönüşümü gerçekleştirilmemiştir. Gelecekteki çalışmalarda öğrencilerin geliştirdikleri oyun prototiplerinin sanal gerçeklik türüne dönüşümü sağlanarak bunun öğrencilerin psikolojik faktörlerine etkisi incelenebilir.

- Bu çalışma kapsamında her ne kadar UML sınıf diyagramları öğretim programına eklenmiş olsa da zaman kısıtlamasından dolayı tasarım kalıpları (design patterns) konuları detaylı olarak anlatılamamıştır. Gelecekteki çalışmalarda NYP dersinin öğretim programına oyun geliştirmede tasarım kalıpları eklenerek bunun NYP paradigmasını öğrenmeyi kolaylaştırıp kolaylaştırmadığı incelenebilir.

- Bu tez çalışması kapsamında NYP kavramlarının öğretilmesi ve uygulanması C# programlama dili kullanılarak gerçekleştirilmiştir. Gelecekteki çalışmalarda Unity 3D'nin görsel programlama dili olan Bolt kullanılarak NYP paradigmasının uygulandığı çalışmalar önerilebilir.

- Bu çalışma daha önce prosedürel programlama deneyimi olan öğrencilerle yürütülmüştür. "İlk önce nesne" yaklaşımı ile kurgulanan bir çalışma kurgulanarak programlama deneyimi olmayan bir öğrenci grubuyla yürütülmesi önerilebilir.

- Gelecekteki çalışmalarda dijital oyun sektöründe görev yapan uzmanların derse davet edilmesi ve onların oyun geliştirme deneyimlerinin öğrencilerle paylaşılması sağlanabilir.

- Bu çalışmada öğrencilerin projelerinde hangi aşamada olduklarının, hangi eklentileri projelerine dâhil ettiklerinin takibinde zorluklar yaşanmıştır. Her ne kadar projelerinin gelişim süreçleriyle ilgili öğrencilerden günlüklerine öz değerlendirmelerde bulunmaları istenmiş olsa da öğrenci günlüklerinin güvenilir bir veri toplama aracı olmadığı

değerlendirilmektedir. Gelecekteki çalışmalarda öğrencilerin gerçek ilerlemelerinin izlenebilmesine yardımcı olabilecek github.com gibi kaynak kodu kontrol sistemleri içeren araçlar kullanılabilir.

- Bu çalışmanın ikinci döngüsünde ders yönetim sistemi olarak Google Classroom kullanılmıştır. Google Classroom her ne kadar kaynak paylaşımına izin verse de özellikle ders videolarının izlenip izlenmediği ya da ders içeriklerine hangi sıklıkla erişimin sağlandığı gibi analitik veriler sağlamamaktadır. Gelecekteki çalışmalarda analitik veriler sağlayan bir ders yönetim sistemi kullanılarak ders materyallerinin kullanımına yönelik detaylı bilgiler elde edilebilir. Bunun yanında bu verilerin ders başarısına etkisiyle ilgili çıkarımlar yapılabilir.

- Bu çalışma daha önce programlama dersi almış ve önlisans düzeyindeki katılımcılarla gerçekleştirilmiştir. İlerideki çalışmalarda programlama konusundan deneyimi olmayan katılımcı gruplarıyla çalışma yapılabilir. İlave olarak oyun geliştirmeye dayalı öğretim stratejisinin daha düşük yaş gruplarındaki etkilerinin incelenebileceği çalışmalar tasarlanabilir.

- Bu çalışmada tasarlanan eylem planı, katılımcıların NYP bilgi ve becerilerine katkıda bulunulmasını; aynı zamanda derse karşı motivasyonlarının, programlama öz yeterlilik algılarının artırılarak programlama kaygılarının azaltılmasını amaçlayan basamaklar içermiştir. Yapılacak yeni çalışmalarda oyun geliştirmeye dayalı öğretim stratejisinin katılımcıların bilgi işlem düşünme (computational thinking) becerilerine etkileri incelenebilir.

- Bu çalışmanın her iki döngüsünde erkek katılımcıların yer aldığı bir örneklem grubuyla çalışılmıştır. İlerideki çalışmalara cinsiyet değişkeni eklenerek oyun geliştirmeye dayalı öğretimin cinsiyet bağlamında etkileri incelenebilir.

BÖLÜM VI: KAYNAKLAR

Ab Hamid, M. R., Sami, W. ve Sidek, M. M. (2017). Discriminant validity assessment: Use of Fornell & Larcker criterion versus HTMT criterion. In *Journal of Physics: Conference Series (Vol. 890, No. 1, p. 012163)*. IOP Publishing.

Abbasi, S., Kazi, H. ve Khowaja, K. (2017). A systematic review of learning object oriented programming through serious games and programming approaches. In *2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)* (pp. 1-6). IEEE.

Abbasi, S. ve Kazi, H. (2020). Stealth assessment in serious games to improve OO learning outcomes. In *2019 International Conference on Advances in the Emerging Computing Technologies (AECT)* (pp. 1-5). IEEE.

Abbasi, S., Kazi, H., Kazi, A. W., Khowaja, K. ve Baloch, A. (2021). Gauge Object Oriented Programming in Student's Learning Performance, Normalized Learning Gains and Perceived Motivation with Serious Games. *Information, 12*(3), 101.

Abbasi, S., Tabbassum, K., Kazi, H., Tunio, S. ve Qureshi, S. (2021). Investigating Student's Obstacles While Learning Object Orientation. *Internatıonal Journal of Scientific & Technology Research, 10*(6), 7-11.

Abdellatif, A. (2020). *Serious Games in Teaching Computer Programming: Usage and Evaluation* (Doctoral dissertation, Queen's University Belfast).

Abdunabi, R., Hbaci, I. ve Ku, H. Y. (2019). Towards Enhancing Programming Self-Efficacy Perceptions among Undergraduate Information Systems Students. *Journal of Information Technology Education*, *18*.

Akcaoglu, M. (2013). *Cognitive and motivational impacts of learning game design on middle school children.* Michigan State University.

Akkaya, A. (2018). *The effects of serious games on studentsconceptual knowledge of object-oriented programming and computational thinking skills* (Master's thesis, Sosyal Bilimler Enstitüsü).

Akkaya, A. ve Akpinar, Y. (2022). Experiential serious-game design for development of knowledge of object-oriented programming and computational thinking skills. *Computer Science Education*, 1-26.

Alaoutinen, S. ve Smolander, K. (2010). Student self-assessment in a programming course using bloom's revised taxonomy. *In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (pp. 155-159).*

Aleksandrovna, D. T. ve Nikolaevich, P. A. (2019). Two-level study of object-oriented programming by university students. *Современные информационные технологии и ИТ-образование, 15*(1), 200-206.

Al-Fedaghi, S. (2021). Classes in Object-Oriented Modeling (UML): Further Understanding and Abstraction. *arXiv preprint arXiv:2106.00267*.

Al-Makhzoomy, A. K. (2018). *Effect of Game Development-Based Learning on the Ability of Information Technology Undergraduates to Learn Computer and Object-Oriented Programming* (Doctoral dissertation, Wayne State University).

Al-Nuaim, H., Allinjawi, A., Krause, P. ve Tang, L. (2011). Diagnosing student learning problems in object oriented programming. *Computer Technology and Application, 2*(11), 25-32.

Alhazbi, S. ve Ismail, L. S. (2010, December). Supportive online learning environment to improve students' satisfaction in object-oriented programming courses. In *2010 2nd International Congress on Engineering Education* (pp. 89-93). IEEE.

Anfurrutia, F. I., Álvarez, A., Larrañaga, M. ve López-Gil, J. M. (2017). Visual Programming Environments for Object-Oriented Programming: Acceptance and Effects on Student Motivation. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, 12*(3), 124-131.

Aniche, M., Yoder, J. ve Kon, F. (2019). Current challenges in practical object-oriented software design. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)* (pp. 113-116). IEEE.

Askar, P. ve Davenport, D. (2009). An investigation of factors related to self-efficacy for Java programming among engineering students. *TOJET: The Turkish Online Journal of Educational Technology, 8(1).*

Avcı Yucel, U. ve Ersoy, H. (2018). Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeğinin Türkçe Uyarlaması: Geçerlik ve Güvenirlik Çalışması. *Yükseköğretim ve Bilim Dergisi, 8(1)*, 73.

Ayalew, Y., Tshukudu, E. ve Lefoanea, M. (2018). Factors affecting programming performance of first year students at a University in Botswana. *African Journal of Research in Mathematics, Science and Technology Education, 22(3)*, 363-373.

Backlund, P. ve Hendrix, M. (2013) Educational Games-Are They Worth the Effort? A Literature Survey of the Effectiveness of Serious Games. In: *Proceedings of the 5th International Conference on Games and Virtual Worlds for Serious Applications*, pp. 1-8

Bakar, M. A., Mukhtar, M. ve Khalid, F. (2019). The development of a visual output approach for programming via the application of cognitive load theory and constructivism. *Development, 10*(11), 305-312.

Bandura, A. (1986). The explanatory and predictive scope of self-efficacy theory. *Journal of social and clinical psychology, 4*(3), 359-373.

Bandura, A. (2001). Social cognitive theory: An agentic perspective. *Annual review of psychology, 52(1)*, 1-26.

Barnes, D. J. (2002). Teaching introductory Java through LEGO MINDSTORMS models. *In Proceedings of the 33rd SIGCSE technical symposium on Computer science education (pp. 147-151).*

Baser, M. (2013). Attitude, Gender and Achievement in Computer Programming. *Online Submission*. https://doi.org/10.5829/idosi.mejsr.2013.14.2.2007

Barlow-Jones, G., van der Westhuizen, D. ve Coetzee, C. (2014). An investigation into the performance of first year programming students in relation to their grade 12 computer subject results. *In EdMedia+ Innovate Learning (pp. 77-83). Association for the Advancement of Computing in Education (AACE).*

Bayliss, J. D. ve Strout, S. (2006). Games as a" flavor" of CS1. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education* (pp. 500-504).

Beaubouef, T. ve Mason, J. (2005). Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin, 37*(2), 103-106.

Beck, A. T., Emery, G. ve Greenberg, R. L. (2005). *Anxiety disorders and phobias: A cognitive perspective.* Basic Books.

Behnke, K. A. (2015). *Gamification in introductory computer science* (Doctoral dissertation, University of Colorado at Boulder).

Bell, T., Alexander, J., Freeman, I. ve Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology, 13(1)*, 20-29.

Bellotti, F., Ott, M., Arnab, S., Berta, R., de Freitas, S., Kiili, K. ve De Gloria, A. (2011). Designing serious games for education: from pedagogical principles to game mechanisms. *In Proceedings of the 5th European Conference on Games Based Learning (pp. 26-34).* Greece: University of Athens.

Bender, W. N. (2012). *Project-based learning: Differentiating instruction for the 21st century*. Corwin Press.

Bennedsen, J. ve Caspersen, M.E. (2007). Failure rates in introductory programming. *SIGCSE Bulletin*, *39*(2), 32–36.

Bennedsen, J. ve Schulte, C. (2007). What does" objects-first" mean? An international study of teachers' perceptions of objects-first. *In Proceedings of the Seventh Baltic Sea Conference on Computing Education Research-Volume 88 (pp. 21-29).*

Bergin, S. ve Reilly, R. (2005). Programming: Factors that in*fl*uence success. *SIGCSE Bulletin, 37*(1), 411–415

Bers, M. U., Flannery, L., Kazakoff, E. R. ve Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. Computers & Education, 72, 145-157.

Bewer, N. ve Gladkaya, M. (2022). Game Development Based Approach for Learning to Program: A Systematic Literature Review.

Biju, S. M. (2013). Difficulties in understanding object oriented programming concepts. In I*nnovations and Advances in Computer, Information, Systems Sciences, and Engineering* (pp. 319-326). Springer, New York, NY.

Bosch, N., D'Mello, S. ve Mills, C. (2013). What emotions do novices experience during their first computer programming learning session?. *In International Conference on Artificial Intelligence in Education (pp. 11-20).* Springer, Berlin, Heidelberg.

Bosse, Y. ve Gerosa, M. A. (2017). Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage. *ACM SIGSOFT Software Engineering Notes, 41*(6), 1-6.

Bouali, N., Nygren, E., Oyelere, S. S., Suhonen, J. ve Cavalli-Sforza, V. (2019, November). Imikode: A VR game to introduce OOP concepts. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research* (pp. 1-2).

Boulden, D. C., Rachmatullah, A., Hinckle, M., Bounajim, D., Mott, B., Boyer, K. E., ... ve Wiebe, E. (2021). Supporting Students' Computer Science Learning with a Game-based Learning Environment that Integrates a Use-Modify-Create Scaffolding Framework. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education* V. 1 (pp. 129-135).

Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., Sanders, K. ve Zander, C. (2007). Threshold concepts in computer science: do they exist and are they useful?. *ACM SIGCSE Bulletin, 39*(1), 504-508.

Böttcher, A., Thurner, V., Häfner, T. ve Hertle, J. (2021, April). A data science-based approach for identifying counseling needs in first-year students. In *2021 IEEE Global Engineering Education Conference (EDUCON) (pp. 420-429)*. IEEE.

Brennan, K. (2015). Beyond technocentrism. *Constructivist Foundations, 10(3)*, 289-296.

Brito Jr, A. D. M. ve Medeiros, A. A. D. D. (2019). A motivating approach to introduce object-oriented programming to engineering students. *The International Journal of Electrical Engineering & Education*, 0020720919856247.

Butler, M. ve Morgan, M. (2007). Learning challenges faced by novice programming students studying high level and low feedback concepts. *Proceedings ascilite Singapore, (99-107).*

Büyüköztürk, Ş., Kılıç-Çakmak, E., Akgün, Ö., Karadeniz, Ş. ve Demirel, F. (2008). *Bilimsel araştırma yöntemleri.*

Byrne, P. ve Lyons, G. (2001) The effect of student attributes on success in programming. *ITiCSE: Proceedings of the 6th annual conference on Innovation and technology in computer science education*. ACM Press, NY, 49-52

Cano, E. M., Ruiz, J. G. ve Garcia, I. A. (2015). Integrating a learning constructionist environment and the instructional design approach into the definition of a basic course

for embedded systems design. Computer Applications in Engineering Education, 23(1), 36-53.

Carbonaro, M., Szafron, D., Cutumisu, M. ve Schaeffer, J. (2010). Computer-game construction: A gender-neutral attractor to computing science. *Computers & Education, 55*(3), 1098–1111.

Carbone, A., Hurst, J., Mitchell, I. ve Gunstone, D. (2009). An exploration of internal factors influencing student learning of programming. *In Proceedings of the Eleventh Australasian Conference on Computing Education-Volume 95 (pp. 25-34).*

Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education, 95*, 202-215.

Char, B. (2016). Automatic feedback systems for code: what can they tell the busy instructor?. *Journal of Computing Sciences in Colleges, 31*(4), 87-93.

Chatain, J., Bitter, O., Fayolle, V., Sumner, R. W. ve Magnenat, S. (2019). A creative game design and programming app. In *Motion, Interaction and Games* (pp. 1-6).

Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology, 12*(2), ep272.

Chen, X., Siau, K. ve Nah, F. F. H. (2012). Empirical comparison of 3-D virtual world and face-to-face classroom for higher education. *Journal of Database Management (JDM), 23*(3), 30-49.

Chen, W. K. ve Cheng, Y. C. (2007). Teaching object-oriented programming laboratory with computer game programming. *IEEE Transactions on Education, 50*(3), 197-203.

Chen, K. C. ve Jang, S. J. (2010). Motivation in online learning: Testing a model of self-determination theory. *Computers in Human Behavior*, *26*(4), 741–752.

Cheung, G. K., Zimmermann, T. ve Nagappan, N. (2014). The first hour experience: how the initial play can engage (or lose) new players. *In Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play (pp. 57-66).*

Choo, M. L. ve Cheung, K. C. (1991). On meaningful measurement: Junior college pupils' anxiety towards computer programming. *Journal of Educational Technology Systems, 19*(4), 327-343.

Chover, M., Marín, C., Rebollo, C. ve Remolar, I. (2020). A game engine designed to simplify 2D video game development. *Multimedia Tools and Applications, 79*(17), 12307-12328.

Chursin, G. ve Semenov, M. (2020). Learning game development with Unity3D engine and Arduino microcontroller. In *Journal of Physics: Conference Series* (Vol. 1488, No. 1, p. 012023). IOP Publishing.

Cigdem, H. ve Yildirim, O. G. (2014). Predictors of C# programming language self-efficacy among vocational college students. *International Journal on New Trends in Education and Their Implications*, *5*(3), 145-153.

Clark, L. A. ve Watson, D. (2016). Constructing validity: Basic issues in objective scale development. In A. E. Kazdin (Ed.), *Methodological issues and strategies in clinical research* (pp. 187–203). American Psychological Association.

Clark, J. S., Porath, S., Thiele, J. ve Jobe, M. (2020). *Action research*. New Prairie Press.

Connolly, C., Murphy, E. ve Moore, S. (2007). Second chance learners, supporting adults learning computer programming. In *international conference on engineering education–ICEE.*

Corral, J. M. R., Balcells, A. C., Estévez, A. M., Moreno, G. J. ve Ramos, M. J. F. (2014). A game-based approach to the teaching of object-oriented programming languages. *Computers & Education, 73*, 83-92.

Covington, M. V. ve Dray, E. (2002). The developmental course of achievement motivation: A need-based approach. *In Development of achievement motivation* (pp. 33-56). Academic Press.

Creswell, J. W. (2003). *Research Design: Qualitative, Quantitative, And Mixed Methods Approaches*. California: Sage Publications.

Creswell, J. W. (2005). *Educational research: planning, condacting, and evaluating quantitative and qualitative research (2nd ed.).* Upper Saddle River, New Jersey, Pearson Education, Inc.

Çelik, H. C. (2020). The effect of modeling, collaborative and game-based learning on the geometry successof third-grade students. *Education and Information Technologies, 450*(25), 449–469

Çınar, M. ve Tüzün, H. (2021). Comparison of object-oriented and robot programming activities: The effects of programming modality on student achievement, abstraction, problem solving, and motivation. *Journal of Computer Assisted Learning, 37*(2), 370-386.

Çimen, B. ve Hangül, Ş. (2021). Digital immigrant teachers' perceptions about digital native students: an investigation into Turkish school context. *European Journal of Education and Psychology, 14*(2), 1-21.

Dadson, M. (2021). *Using an Interactive Narrative App on Object Oriented Programming to Teach and Interest Students in Coding* (Doctoral dissertation).

Dale, N. B. ve Weems, C. (2005). Programming and problem solving with C++. *Jones & Bartlett Learning.*

Dasuki, S. ve Quaye, A. (2016). Undergraduate students' failure in programming courses in institutions of higher education in developing countries: A Nigerian perspective. *The Electronic Journal of Information Systems in Developing Countries, 76*(1), 1-18.

Davis, L. L. (1992). Instrument review: Getting the most from a panel of experts. *Applied nursing research, 5*(4), 194-197.

Demir, F. (2021). The effect of different usage of the educational programming language in programming education on the programming anxiety and achievement. *Education and Information Technologies*, 1-24.

Demirer, V. ve Sak, N. (2016). Dünyada ve Türkiye'de programlama eğitimi ve yeni yaklaşımlar. *Eğitimde Kuram ve Uygulama, 12*(3), 521-546.

Denner, J., Werner, L. ve Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education, 58*(1), 240–249.

Denny, P., Becker, B. A., Bosch, N., Prather, J., Reeves, B. ve Whalley, J. (2022). Novice Reflections During the Transition to a New Programming Language.

De Oliveira, E. D., Camargo, M. C., Barbosa, C. R. S., Brancher, J. D., de Oliveira Barros, V. T., Campos, V. V. D. S. ve De Barros, R. M. (2017). The power of the game as a mediator tool paradigm of object oriented teaching-learning process. In *2017 IEEE Frontiers in Education Conference (FIE)* (pp. 1-8). IEEE.

Dolgopolovas, V., Jevsikova, T. ve Dagiene, V. (2018). From Android games to coding in C—An approach to motivate novice engineering students to learn programming: A case study. *Computer Applications in Engineering Education, 26*(1), 75-90.

Doyle, M., Kasturiratna, D., Richardson, B. D. ve Soled, S. W. (2009). Computer Science and Computer Information Technology majors together: Analyzing factors impacting students' success in introductory programming. *In 2009 39th IEEE Frontiers in Education Conference (pp. 1-6). IEEE*.

Dreyfus, H. ve Dreyfus, S. (1986). *Mind over machine: The power of human intuition and expertise in the era of the computer.* New York: Free Press.

Dümmel, N., Westfechtel, B. ve Ehmann, M. (2020). MuLE: a Multiparadigm Language for Education. The Object-Oriented Part of the Language. In *Proceedings of the 4th European Conference on Software Engineering Education* (pp. 32-41).

Dwarika, J. ve De Villiers, M. R. (2015). Use of the Alice visual environment in teaching and learning object-oriented programming. In *Proceedings of the 2015 Annual Research Conference on South African Institute of Computer Scientists and Information Technologists* (pp. 1-10).

Eckerdal, A. (2006). *Novice students' learning of object-oriented programming* (Doctoral dissertation, Uppsala University).

Eckerdal, A. ve Thuné, M. (2005). Novice Java programmers' conceptions of" object" and" class", and variation theory. *ACM SIGCSE Bulletin, 37*(3), 89-93.

El-Nasr, M. S. ve Smith, B. K. (2006). Learning through game modding. *Computers in Entertainment (CIE), 4*(1), 7-es.

Entwistle, N. (2014). Motivation and approaches to learning: motivating and conceptions of teaching. *In Motivating students (pp. 25-34). Routledge.*

Erdem, M. (2002). Proje Tabanlı Öğrenme. *Hacettepe Üniversitesi Eğitim Fakültesi Dergisi. 22*:172-179.

Erdoğan, E. ve Akbaba, B. (2019). Ters yüz edilmiş sınıf modeliyle ortaokul öğrencilerinin sosyal bilgiler dersi akademik başarılarının geliştirilmesi. *Cumhuriyet Uluslararası Eğitim Dergisi, 8*(1), 193-213.

Ericsson, K. A., Krampe, R. T. ve Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological review, 100*(3), 363.

Esteves, M., Fonseca, B., Morgado, L. ve Martins, P. (2009). Using Second Life for problem based learning in computer science programming. *Journal for Virtual Worlds Research, 2*(1).

Esteves, M. ve Mendes, A. J. (2004). A simulation tool to help learning of object oriented programming basics. *In Frontiers in Education, 2004. FIE 2004. 34th Annual (pp. F4C-7). IEEE.*

Ezeamuzie, N. O. (2022). Project-first approach to programming in K–12: Tracking the development of novice programmers in technology-deprived environments. *Education and Information Technologies*, 1-31.

Falkner, N., Vivian, R., Piper, D. ve Falkner, K. (2014). Increasing the effectiveness of automated assessment by increasing marking granularity and feedback units. In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 9-14).

Fang, X. (2012). Application of the participatory method to the computer fundamentals course, Affective Computing and Intelligent Interaction. *Advances in Intelligent and Soft Computing, 137*, 185-189.

Feldgen, M. ve Clúa, O. (2004). Games as a motivation for freshman students to learn programming. *34th ASEE. In IEEE Frontiers in Education Conference (pp. 1079-1084).*

Ferrance, E. (2000). *Action research*. LAB, Northeast and Island Regional Education Laboratory at Brown University.

Fincher, S., Cooper, S., Kölling, M. ve Maloney, J. (2010). Comparing alice, greenfoot & scratch. *In Proceedings of the 41st ACM technical symposium on Computer science education (pp. 192-193).*

Fletcher, G. H. ve Lu, J. J. (2009). Education Human computing skills: rethinking the K-12 experience. *Communications of the ACM, 52*(2), 23-25.

Foster, S. R. (2015). *Three Paradigms for Mixing Coding and Games: Coding in a Game, Coding as a Game, and Coding for a Game.* University of California, San Diego.

Fowler, A., Fristce, T. ve MacLauren, M. (2012). Kodu Game Lab: a programming environment. *The Computer Games Journal, 1*(1), 17-28.

Forte, A. ve Guzdial, M. (2004). Computers for communication, not calculation: Media as a motivation and context for learning. *In 37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the (pp. 10-pp). IEEE.*

Fraenkel, J. R. ve Wallen, N. E. (2003). *How to design and evaluate research in education (5th Ed.).* New York: Mac Graw Hill, Inc.

Gainutdinova, T., Denisova, M. ve Shirokova, O. (2019). Current Trends in the Study of Object-Oriented Programming in Higher Education. *ARPHA Proceedings, 1*, 247.

Garcia, I., Pacheco, C., Méndez, F. ve Calvo-Manzano, J. A. (2020). The effects of game-based learning in the acquisition of "soft skills" on undergraduate software engineering courses: A systematic literature review. *Computer Applications in Engineering Education, 28*(5), 1327-1354.

Garcia Ramirez, J. (2022). The Impacts of a New Programming Course in a First-Year Engineering Experience.

Garneli, V. ve Chorianopoulos, K. (2018). Programming video games and simulations in science education: exploring computational thinking through code analysis. *Interactive Learning Environments, 26*(3), 386-401.

Garris, R., Ahlers, R. ve Driskell, J. E. (2002). Games, motivation, and learning: A research and practice model. *Simulation & gaming, 33*(4), 441-467.

Gomes, A. ve Mendes, A. J. N. (2007). Learning to program-difficulties and solutions. In *International Conference on Engineering Education* (pp. 1–5). Coimbra, Portugal.

Goodrich, H. (2001). The effects of instructional rubrics on learning to write. *Current Issues in Education, 4*(4), 1-21.

Grant, M. M. ve Branch, R. M. (2005). Project-based learning in a middle school: Tracing abilities through the artifacts of learning. *Journal of Research on technology in Education, 38*(1), 65-98.

Grover, S. ve Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43.

Galves. J., Guzman. E. ve Conejo. R., (2009). "A blended E-learning experience in a course of object oriented programming fundamentals". *Knowledge-Based Systems, Vol. 22, pp. 279-286*

Gomes, A. ve Mendes, A. J. (2007). An environment to improve programming education. *In Proceedings of the 2007 international conference on Computer systems and technologies(p. 88). ACM.*

Green, S., Chan, C. ve Plant, N., (2016). "Student at Risk: Identification and Remedial Action System for Improving Retention on Computer Science Programmes". *New Directions in the Teaching of Physical Sciences, Vol. 11, Issue. 1.*

Grizioti, M. ve Kynigos, C. (2018). Game modding for computational thinking: an integrated design approach. *In Proceedings of the 17th ACM Conference on Interaction Design and Children (pp. 687-692).*

Grizioti, M. ve Kynigos, C. (2021). Children as players, modders, and creators of simulation games: A design for making sense of complex real-world problems: Children as players, modders and creators of simulation games. In *Interaction Design and Children* (pp. 363-374).

Groher, I., Vierhauser, M., Sabitzer, B., Kuka, L., Hofer, A. ve Muster, D. (2022). Exploring diversity in introductory programming classes: an experience report. In 44th International Conference on Software Engineering: Software Engineering Education and Training.

Guenaga, M., Eguíluz, A., Garaizar, P. ve Gibaja, J. (2021). How do students develop computational thinking? Assessing early programmers in a maze-based online game. *Computer Science Education, 31*(2), 259-289.

Guzdial, M. ve DiSalvo, B. (2013). Computing education: Beyond the classroom. Computer, 46(9), 30-31.

Gürgür, H. (2016). Eylem araştırması. *Eğitimde nitel araştırma desenleri içinde*, 3-50.

Hainey, T., Connolly, T. M., Stansfield, M. ve Boyle, E. A. (2011). Evaluation of a game to teach requirements collection and analysis in software engineering at tertiary education level. *Computers & Education, 56*(1), 21-35.

Hair, J., Black, W., Babin, B. ve Anderson, R. (2010). *Multivariate data analysis (7th ed.).* Upper Saddle River, NJ: Prentice-Hall.

Harel, I. ve Papert, S. (1991). *Constructionism*. Westport, CT, US: Ablex Publishing.

Hauswirth, M. ve Adamoli, A. (2013). Teaching Java programming with the Informa clicker system. *Science of Computer Programming, 78*(5), 499-520.

Hay, K. E. ve Barab, S. A. (2001). Constructivism in practice: A comparison and contrast between apprenticeship and constructionist learning environments. *The Journal of the Learning Sciences, 10*(3), 281–322.

Helme, S. ve Clarke, D. (2001). Identifying cognitive engagement in the mathematics classroom. *Mathematics Education Research Journal, 13(*2), 133-153.

Henseler, J., Ringle, C. M. ve Sarstedt, M. (2015). A new criterion for assessing discriminant validity in variance-based structural equation modeling. *Journal of the academy of marketing science, 43(1)*, 115-135.

Henriksen, P. ve Kölling, M. (2004). Greenfoot: combining object visualisation with interaction. *In Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications (pp. 73-82).*

Herala, A., Vanhala, E. ve Nikula, U. (2015). Object-oriented programming course revisited. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research* (pp. 23-32).

Hermans, F. ve Aivaloglou, E. (2016,). Do code smells hamper novice programming? A controlled experiment on Scratch programs. *In 2016 IEEE 24th International Conference on Program Comprehension (ICPC) (pp. 1-10). IEEE.*

Hewett, K. J. E. (2016). *The Minecraft project: Predictors for academic success and 21 st century skills gamers are learning through video game experiences* (Doctoral dissertation, Texas A&M University-Corpus Christi).

Holm, M. (2011). Project-based instruction: A review of the literature on effectiveness in prekindergarten through 12th grade classrooms. *InSight: Rivier Academic Journal*, *7*(2), 1–13.

Holmstedt, V. ve Mengiste, S. A. (2017). Why and how Practice impacts Confidence in introductory object oriented programming Courses.

Holopainen, T. (2016). Object-oriented programming with Unity: Inheritance versus composition.

Hooshyar, D., Pedaste, M., Yang, Y., Malva, L., Hwang, G. J., Wang, M., ... & Delev, D. (2021). From gaming to computational thinking: An adaptive educational computer game-based learning approach. *Journal of Educational Computing Research, 59*(3), 383-409.

Horses, I. H. M. (2016). *From lived experiences to game creation: How scaffolding supports elementary school students learning computer science principles in an after school setting* (Doctoral dissertation, University of Colorado at Boulder).

Hu, M. ve Steele, A. (2017). Reducing the Gap between OOP Design and Development. In *Proceedings of the 8th Annual Conference of Computing and Information Technology Research and Education New Zealand (CITRENZ)* (pp. 70-75).

Huang Y, Zhang T ve Xu, L. (2018). The Development of a Game with Applications of Object-oriented Programming Concepts. *American Journal of Advanced Research*, 2018, 2–1.

Huang, W. (2020). *Investigating the novelty effect in virtual reality on stem learning* (Doctoral dissertation, Arizona State University).

Hubbard, R. S. ve Power, B. M. (2003). *The art of classroom inquiry: A handbook for teacher-researchers*. Portsmouth, NH: Heinemann.

Huggard, M. (2004). Programming trauma: can it be avoided. *Proceedings of the BCS Grand Challenges in Computing: Education, 50-51*.

Hussain, A., Shakeel, H., Hussain, F., Uddin, N. ve Ghouri, T. L. (2020). Unity Game Development Engine: A Technical Survey. *Univ. Sindh J. Inf. Commun. Technol*, 4, 73-81.

Ismail, M. N., Ngah, N. A. ve Umar, I. N. (2010). Instructional strategy in the teaching of computer programming: a need assessment analyses. *TOJET: The Turkish Online Journal of Educational Technology, 9*(2).

Jafari, S. M. ve Abdollahzade, Z. (2019). Investigating the relationship between learning style and game type in the game-based learning environment. *Education and Information Technologies, 24*(5), 2841-2862.

Jiau, H. C., Chen, J. C. ve Ssu, K. F. (2009). Enhancing self-motivation in learning programming using game-based simulation and metrics. *IEEE Transactions on Education, 52*(4), 555-562.

Johnson, A. P. (2014). *Eylem araştırması el kitabı* (Y. Uzuner & M. Ö. Anay, Çev.). Ankara: Anı Yayıncılık.

Kafai, Y. (1995). Making Game Artifacts To Facilitate Rich and Meaningful Learning.

Kafai, Y. B. (2006). Playing and making games for learning instructionist and constructionist perspectives for game studies. *Games and Culture, 1*(1), 36–40.

Kaila, E., Kurvinen, E., Lokkila, E. ve Laakso, M. J. (2016). Redesigning an object-oriented programming course. *ACM Transactions on Computing Education (TOCE), 16*(4), 1-21.

Kalelioglu, F. ve Gülbahar, Y. (2014). The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective. *Informatics in Education, 13*(1), 33-50.

Kanaki, K. ve Kalogiannakis, M. (2018). Introducing fundamental object-oriented programming concepts in preschool education within the context of physical science courses. *Education and Information Technologies, 23*(6), 2673-2698.

Karagöz, M. (2018). *Türk ve dünya klasikleriyle yapılandırılmış dil ve edebiyat öğretimi tasarısı: okulöncesinden yükseköğretime (bir eylem araştırması).* Yayımlanmamış doktora tezi, Ankara Üniversitesi.

Karram, O. (2021). The Role of Computer Games in Teaching Object-Oriented Programming in High Schools-Code Combat as a Game Approach. *WSEAS Transactions on Advances in Engineering Education*, 18, 37-46.

Kasto, N. (2016). *Learning to Program: The development of knowledge in Novice Programmers* (Doctoral dissertation, Auckland University of Technology).

Kerr, A. (2006). *The Business of Making Games. In Understanding Digital Games*. Sage Publications Ltd.

Kert, S. B. ve Uğraş, T. (2009). Programlama eğitiminde sadelik ve eğlence: Scratch örneği. In *The First International Congress of Educational Research, Çanakkale, Turkey*.

Ketelhut, D. J. (2007). The impact of student self-efficacy on scientific inquiry skills: An exploratory investigation in river city, a multi-user virtual environment. *Journal of Science Education and Technology, 16*(1), 99-111

Khaleel, F. L., Ashaari, N. S., Meriam, T. S., Wook, T. ve Ismail, A. (2015). The study of gamification application architecture for programming language course. *In Proceedings of the 9th international conference on ubiquitous information management and communication (pp. 1-5).*

Kırcaburun, K., Baştuğ,İ. ve Bahtiyar, M. (2017). Modeling the psychological factors affecting computer programming self-efficacy. *AJELI-Anatolian Journal of Educational Leadership and Instruction, 5*(1), 17-27.

Kinnunen, P. ve Malmi, L. (2006). Why students drop out CS1 course?. *In Proceedings of the second international workshop on Computing education research (pp. 97-108).*

Kline, T. (2005). *Psychological testing: A practical approach to design and evaluation*. Sage.

Koniukhov, S. (2018). Methods and Means of Training Object-Oriented Programming in Higher Education Institutions. *Ukrainian Journal of Educational Studies and Information Technology, 6*(1), 103-113.

Korkmaz, Ö. ve Altun, H. (2014). Adapting computer programming self-efficacy scale and engineering students' self-efficacy perceptions. *Participatory Educational Research, 1*(1), 20-31.

Korkmaz, Ö. (2016). The Effect of Scratch-Based Game Activities on Students' Attitudes, Self-Efficacy and Academic Achievement. *Online Submission, 8*(1), 16-23.

Kotsovoulou, M. (2020). *Exploring student perceptions about the use of visual programming environments, their relation to student learning styles and their impact on student motivation in undergraduate introductory programming modules.* Lancaster University (United Kingdom).

Kölling, M. ve Rosenberg, J. (1996). An object-oriented program development environment for the first programming course. *ACM SIGCSE Bulletin, 28*(1), 83-87.

Kölling, M. (2010). The greenfoot programming environment. *ACM Transactions on Computing Education (TOCE), 10*(4), 1-21.

Kölling, M. (2018). Blue, bluej, greenfoot: Designing educational programming environments. In *Innovative Methods, User-Friendly Tools, Coding, and Design Approaches in People-Oriented Programming* (pp. 42-87). IGI Global.

Krippendorff, K. (2018). *Content analysis: An introduction to its methodology*. Sage publications.

Krueger, R. A. (1994). *Focus groups: A practical guide for applied research*. Thousand Oaks
CA: Sage Publications.

Kucher, T. (2021). Principles and best practices of designing digital game-based learning environments. *International Journal of Technology in Education and Science*, *5*(2), 213-223.

Kukul, V. (2018). *Programlama Öğretiminde Farklı Yapılandırılan Süreçlerin Öğrencilerin Bilgi İşlemsel Düşünme Becerilerine, Özyeterliliklerine ve Programlama Başarılarına Etkisi.* Yayımlanmamış doktora tezi, Gazi Üniversitesi.

Kukul, V., Gökçearslan, Ş. ve Günbatar, M. S. (2017). Computer programming self-efficacy scale (CPSES) for secondary school students: Development, validation and reliability. *Eğitim Teknolojisi Kuram ve Uygulama, 7*(1), 158-179.

Kunkle, W. M., ve Allen, R. B. (2016). The impact of different teaching approaches and languages on student learning of introductory programming concepts. *ACM Transactions on Computing Education (TOCE), 16*(1), 1-26.

Kurkovsky, S. (2013). Mobile game development: improving student engagement and motivation in introductory computing courses. *Computer Science Education*, *23*(2), 138-157.

Kurniawan, O., Jégourel, C., Lee, N. T. S., De Mari, M. ve Poskitt, C. M. (2021). Steps Before Syntax: Helping Novice Programmers Solve Problems using the PCDIT Framework. *arXiv preprint arXiv:2109.08896.*

Kutlu, Ö., Doğan, C. ve Karakaya, İ. (2014). *Ölçme ve değerlendirme: performansa ve portfolyoya dayalı durum belirleme*. Pegem Akademi. Ankara. Beşinci baskı.

Kuzu, A. (2009). Öğretmen Yetiştirme ve Mesleki Gelişimde Eylem Araştırması. *Journal of International Social Research, 1*(6).

Ladkin, D. (2004). Action research. In Seale, C., Gobo, G., Gubrium, J. F. ve Silverman, D. (Eds.), *Qualitative research practice* (pp. 478-490). SAGE Publications Ltd, https://www.doi.org/10.4135/9781848608191

Lahtinen, E., Ala-Mutka, K. ve Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *ACM Sigcse Bulletin, 37*(3), 14-18.

Laporte, L. ve Zaman, B. (2018). A comparative analysis of programming games, looking through the lens of an instructional design model and a game attributes taxonomy. *Entertainment Computing, 25*, 48-61.

Law, R. (2020). A pedagogical approach to teaching game programming: Using the PRIMM approach. In *Proceedings of the 14th European Conference on Games Based Learning* (pp. 816-819). Academic Conferences International.

Law, K. M., Lee, V. C. ve Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education, 55*(1), 218-228.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., ... ve Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32-37.

Lee, N. T. S., Kurniawan, O. ve Choo, K. T. W. (2021, June). Assessing Programming Skills and Knowledge During the COVID-19 Pandemic: An Experience Report. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education* V. 1 (pp. 352-358).

Le-Guillou, I. (2020). *Covid-19: How unprecedented data sharing has led to faster-than-ever outbreak research*. Horizon.

Leonard, J., Buss, A., Gamboa, R., Mitchell, M., Fashola, O. S., Hubert, T. ve Almughyirah, S. (2016). Using robotics and game design to enhance children's self-efficacy, STEM attitudes, and computational thinking skills. *Journal of Science Education and Technology, 25*(6), 860-876.

Lian, V., Varoy, E. ve Giacaman, N. (2022). Learning Object-Oriented Programming Concepts Through Visual Analogies. *IEEE Transactions on Learning Technologies*.

Liberman, N., Beeri, C. ve Ben-David Kolikant, Y. (2011). Difficulties in learning inheritance and polymorphism. *ACM Transactions on Computing Education (TOCE), 11*(1), 1-23.

Lin, Y. P., Chang, R. K. ve Tsai, C. Y. (2016). *The relationship between students' self-efficacy and basic programming concepts*. Paper presented at the Conference of Asia-Pacific Educational Research Association & Taiwan Education Research Association, Kaohsiung, Taiwan

Lin, Y. T., Yeh, M. K. C. ve Tan, S. R. (2022). Teaching Programming by Revealing Thinking Process: Watching Experts' Live Coding Videos With Reflection Annotations. *IEEE Transactions on Education.*

Linares-Pellicer, J., Orta-López, J., Salavert-Torres, J., Segura Flor, M. J., Silvestre Cerdà, J. A., & Sanchis, R. (2020). Towards Inter-Subject Project-Based Learning in Programming-Related Courses at Computer Science Studies. *EDULEARN Proceedings (Internet)*, 3973-3978.

Lincoln, Y. S. ve Guba, E. G. (1985). *Naturalistic Inquiry*. Beverly Hills, CA: Sage

Lister, R., Simon, B., Thompson, E., Whalley, J. L. ve Prasad, C. (2006). Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *ACM SIGCSE Bulletin, 38*(3), 118-122.

Livovský, J. ve Porubän, J. (2014). Learning object-oriented paradigm by playing computer games: concepts first approach. *Central European Journal of Computer Science, 4*(3), 171-182.

Loh, C. S. ve Byun, J. H. (2009). Modding Neverwinter Nights into serious games. In Digital simulations for improving education: *Learning through artificial teaching environments (pp. 408-426). IGI Global.*

López, M. A., Duarte, E. V., Gutiérrez, E. C. ve Valderrama, A. P. (2019). Teaching based on ludic environments for the first session of computer programming-Experience with digital natives. *IEEE Revista Iberoamericana de Tecnologias Del Aprendizaje, 14*(2), 34-42.

López, M. A., Duarte, E. V. ve Gutiérrez, E. C. (2020). *Experience in Teaching Object-Oriented Programming and Advanced Topics of Programming through the Development of a Video Game Project.*

Lukas, G. (1972). Uses of the LOGO programming language in undergraduate instruction. *In Proceedings of the ACM annual conference-Volume 2 (pp. 1130-1136).*

Luxton-Reilly, A., Ajanovski, V. V., Fouh, E., Gonsalvez, C., Leinonen, J., Parkinson, J., ... ve Thota, N. (2019). Pass rates in introductory programming and in other STEM disciplines. *In Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (pp. 53-71).*

Malik, S. I., Mathew, R., Al-Nuaimi, R., Al-Sideiri, A. ve Coldwell-Neilson, J. (2019). Learning problem solving skills: Comparison of E-Learning and M-Learning in an introductory programming course. *Education and Information Technologies, 24*(5), 2779-2796.

Mannila, L. (2009). *Teaching mathematics and programming: new approaches with empirical evaluation* (Doctoral dissertation, Åbo Akademi University).

MacLaurin, M. B. (2011). The design of Kodu: A tiny visual programming language for children on the Xbox 360. *In Proceedings of the 38th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages (pp. 241-246).*

Maguire, P., Maguire, R. ve Kelly, R. (2017). Using automatic machine assessment to teach computer programming. *Computer Science Education, 27*(3-4), 197-214.

Malim, T. (2017). *Introductory psychology*. Macmillan International Higher Education.

Martin, F., Lee, I., Lytle, N., Sentance, S. ve Lao, N. (2020, February). Extending and evaluating the use-modify-create progression for engaging youth in computational thinking. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 807-808).

Matwyczuk, R. (2013). *Epistemic learning: Game programming learned from the lens of professionals*. University of Manitoba (Canada).

McArthur, V. ve Teather, R. J. (2015). Serious mods: A case for modding in serious games pedagogy. *In 2015 IEEE Games Entertainment Media Conference (GEM) (pp. 1-4). IEEE.*

McGill, T. J. ve Volet, S. E. (1997). A conceptual framework for analyzing students' knowledge of programming. *Journal of research on Computing in Education, 29*(3), 276-297.

McGregor, G. (2020). *How an overlooked scientific feat led to the rapid development of COVID-19 vaccines.* Fortune.

Medeiros, R. P., Ramalho, G. L. ve Falcão, T. P. (2018). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education, 62*(2), 77-90.

Mertler, C. A. (2009). *Action research: Teachers as researchers in the classroom*. Sage.

Mihci, C. ve Ozdener, N. (2014). Programming Education with a Blocks-Based Visual Language for Mobile Application Development. *International Association for the Development of the Information Society.*

Miles, M. ve Huberman, A. (2015). Nitel veri analizi: genişletilmiş bir kaynak kitap (1. Baskı). *S. Akbaba Altun ve A. Ersoy (Çev. Eds). Ankara: Pegem Akademi*.

Miliszewska, I. ve Tan, G. (2007). Befriending computer programming: A proposed approach to teaching introductory programming. *Informing Science: International Journal of an Emerging Transdiscipline*, *4*(1), 277-289.

Miller, C. S., Settle, A. ve Lalor, J. (2015). Learning object-oriented programming in python: Towards an inventory of difficulties and testing pitfalls. In *Proceedings of the 16th annual conference on information technology education* (pp. 59-64).

Montiel, H. ve Gomez-Zermeño, M. G. (2021). Educational challenges for computational thinking in K–12 education: A systematic literature review of "Scratch" as an innovative programming tool. *Computers, 10*(6), 69.

Morrow, V. ve Richards, M. (1996). The ethics of social research with children: An overview. *Children & Society, 10*(2), 90-105.

Muller, O. (2005). Pattern oriented instruction and the enhancement of analogical reasoning. *In Proceedings of the first international workshop on Computing education research* (pp. 57-67).

Morrison, G. R., Ross, S.M. ve Kemp, J. E. (2004). Designing Effective Instruction. Hoboken, NJ: John Wiley & Sons, Inc.

Moshirnia, A. (2007). The educational potential of modified video games. *Issues in Informing Science & Information Technology, 4*.

Neuendorf, K. A. (2017). *The content analysis guidebook*. Sage publications.

Nolan, K. ve Bergin, S. (2016). The role of anxiety when learning to program: a systematic review of the literature. In *Proceedings of the 16th koli calling international conference on computing education research* (pp. 61-70).

Nolan, K. (2019). *Mental Health in Computer Science. An Investigation of How Mental Health Affects Learning Computer Science* (Doctoral dissertation, National University of Ireland, Maynooth (Ireland)).

O'Brien, R. (1998). *An overview methodological approach of action research.*

O'Grady-Jones, M. K. (2020). *Ready Coder One: Action Research Exploring the Effects of Collaborative Game Design-based Learning on Gifted Fourth Graders' 21st Century Skills and Science Content Knowledge* (Doctoral dissertation, University of South Carolina).

O'Grady, M. (2012). Practical problem-based learning in computing education. *Transactions on Computing Education, 12*(3), 1-16.

Olsen, O. E. ve Lindøe, P. (2004). Trailing research based evaluation; phases and roles. *Evaluation and Program Planning, 27*(4), 371-380.

Oluk, A., Korkmaz, Ö. ve Oluk, H. A. (2018). Effect of scratch on 5th graders' algorithm development and computational thinking skills. *Turkish Journal of Computer and Mathematics Education (TURCOMAT), 9*(1), 54-71.

Oram, A. ve Wilson, G. (2010). *Making software: What really works, and why we believe it.* O'Reilly Media, Inc.

Owens, B. B. ve Stephenson, C. (2011). *CSTA K – 12 Computer Science Standards*. New York, NY.

Özdener, N. (2008). A comparison of the misconceptions about the time-efficiency of algorithms by various profiles of computer-programming students. *Computers & Education, 51(3)*, 1094-1102.

Özmen, B. ve Altun, A. (2014). Undergraduate students' experiences in programming: Difficulties and obstacles. *Turkish Online Journal of Qualitative Inquiry*, *5*(3), 1-27

Özmen, B. (2020). Programlama Öğretiminde Bilgisayımsal Düşünme Becerilerinin Geliştirilmesine Yönelik Oyun Tabanlı Bir Tasarım Modeli Önerisi.

Sajaniemi, J., Kuittinen, M. ve Tikansalo, T. (2008). A study of the development of students' visualizations of program state during an elementary object-oriented programming course. *Journal on Educational Resources in Computing (JERIC), 7*(4), 1-31.

Sanders, K. ve Thomas, L. (2007). Checklists for grading object-oriented CS1 programs: concepts and misconceptions. *ACM SIGCSE Bulletin, 39*(3), 166-170.

Susanti, W. (2021). An overviuw of the teaching and learning process basic programming in algorithm and programming courses. *Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(2)*, 2934-2944.

Pallant, J. (2020). *SPSS survival manual: A step by step guide to data analysis using IBM SPSS*. Routledge.

Paiva, J. C., Leal, J. P. ve Queirós, R. (2020). Fostering programming practice through games. *Information, 11*(11), 498.

Panskyi, T. ve Rowińska, Z. (2021). A holistic digital game-based learning approach to out-of-school primary programming education. *Informatics in Education, 20*(2), 255-276.

Papadopoulos, Y. ve Tegos, S. (2012). Using Microworlds to Introduce Programming to Novices. In *16th Panhellenic Conference on Informatics* (pp. 180–185).

Papert, S. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, NY, USA.

Pajares, F. (1996). Self-efficacy beliefs in academic settings. *Review of Educational Research, 66*, 543–578.

Paraskeva, F., Mysirlaki, S. ve Papagianni, A. (2010). Multiplayer online games as educational tools: Facing new challenges in learning. *Computers & Education, 54*(2), 498-505.

Parastar, M. (2021). *Characteristics of mental representations in novices and experts in Java* (Doctoral dissertation, University of New Brunswick.).

Parkinson, J. ve Cutts, Q. (2022). Relationships between an Early-Stage Spatial Skills Test and Final CS Degree Outcomes. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education* V. 1 (pp. 293-299).

Pattis, R. E., Roberts, J. ve Stehlik, M. (1995). *Karel the Robot: A Gentle Introduction to The Art of Programming.*

Pea, R. D. ve Kurland, D. M. (1983). On the cognitive prerequisites of learning computer programming. *AEDS Journal*, *18*(3), 183-194.

Pea, R. D. ve Kurland, D. M. (1984). On the cognitive and educational benefits of teaching children programming: A critical look. *New ideas in psychology, 2*, 137-168.

Pellas, N. (2014). The development of a virtual learning platform for teaching concurrent programming languages in the Secondary Education: The use of Open Sim and Scratch4OS. *Journal of E-learning and Knowledge Society, 10*(1).

Peña Miguel, N., Corral Lage, J. ve Mata Galindez, A. (2020). Assessment of the development of professional skills in university students: Sustainability and serious games. *Sustainability, 12*(3), 1014.

Perera, P., Tennakoon, G., Ahangama, S., Panditharathna, R. ve Chathuranga, B. (2021). A Systematic Review of Introductory Programming Languages for Novice Learners. *IEEE* Access.

Perez-Gonzalez, H. G., Nunez-Varela, A. S., Martinez-Perez, F. E., Nava-Munoz, S. E., García, C. G., Kalita, J. ve Juárez-Ramírez, R. (2021, October). A 3D Metaphor for Software Code Visualization to Help Students to learn Object-Oriented Concepts. In *2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT)* (pp. 261-267). IEEE.

Perkins, D. N., Hancock, C., Hobbs, R., Martin, F. ve Simmons, R. (1989). Conditions of Learning in Novice Programmers. Studying the Novice Programmer. E. Soloway and JC Spohrer.

Petrovskaya, E. (2019). *Not All Fun and Games: The Design and Evaluation of a Game to Increase Intrinsic Motivation in Learning Programming*. (HCI-E MSc Final Project Report). University College, London

Poolsawas, B., Chaipornkeaw, P. ve Suparkawat, S. (2016). Platform-based for Game Development to Improve The Object-oriented Programming Skills. In *30th Natl. Conf. Educ. Technol. Naresuan Univ. Phitsanulok 65000 Thailand* (Vol. 30, p. 6).

Porter, L., Guzdial, M., McDowell, C. ve Simon, B. (2013). Success in introductory programming: What works?. Communications of the ACM, 56(8), 34-36.

Porubän, J., Bačíková, M. ve Št'astná, J. (2016, November). Motivating students in component-based programming courses. In *2016 International Conference on Emerging eLearning Technologies and Applications (ICETA)* (pp. 289-294). IEEE.

Qian, Y. ve Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE), 18*(1), 1-24.

Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A. ve Lahmine, S. (2015). Learning basic programming concepts by creating games with scratch programming environment. *Procedia-Social and Behavioral Sciences, 191*, 1479-1482.

Quille, K. ve Bergin, S. (2016). Does Scratch improve self-efficacy and performance when learning to program in C#? An empirical study. In *International Conference on Enguaging Pedagogy (ICEP).*

Ragonis, N. ve Ben-Ari, M. (2005). A long-term investigation of the comprehension of OOP concepts by novices. *Computer Science Education, 15*(3), 203-221.

Ragonis, N. ve Shmallo, R. (2021). The Application of Higher-Order Cognitive Thinking Skills to Promote Students' Understanding of the Use of static in Object-Oriented Programming. *Informatics in Education*.

Ramalingam, V., LaBelle, D. ve Wiedenbeck, S. (2004). Self-efficacy and mental models in learning to program. *ACM SIGCSE Bulletin, 36*(3), 171-175

Renton, D. (2016). The Importance of Computer Games Development in the Computing Curriculum in Schools.*The Computer Games Journal volume 5*, pages1–5

Renzella, J., Cummaudo, A., Cain, A., Grundy, J. ve Meyers, J. (2018). SplashKit: A development framework for motivating and engaging students in introductory programming. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 40-47). IEEE.

Resnick, M., Maloney, J., Monroy-Hern´andez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. ve Kafai, Y. (2009). Scratch: programming for all. *Commun. ACM*, 52(11):60{67.

Resnick, M. (2014). Give P'sa chance: Projects, peers, passion, play. In *Constructionism and creativity: Proceedings of the Third International Constructionism Conference. Austrian Computer Society, Vienna* (pp. 13-20).

Reuter, R., Stark, T., Sedelmaier, Y., Landes, D., Mottok, J. ve Wolff, C. (2020). Insights in Students' Problems during UML Modeling. In 2020 IEEE Global Engineering Education Conference (EDUCON) (pp. 592-600). IEEE.

Roberts, E. (2005). *Karel the robot learns java.* Department of Computer Science Stanford University.

Robertson, J. (2012). Making games in the classroom: Benefits and gender concerns. *Computers & Education, 59*(2), 385-398.

Robins, A. V. (2019). Novice Programmers and Introductory Programming. *The Cambridge handbook of computing education research, 327.*

Robins, A., Rountree, J. ve Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education, 13*(2), 137-172.

Robbins, S. B., Lauver, K., Le, H., Davis, D., Langley, R. ve Carlstrom, A. (2004). Do psychosocial and study skill factors predict college outcomes? A meta-analysis. *Psychological Bulletin, 130*(2), 261–288.

Robson, C. (2015). *Bilimsel araştırma yöntemleri: Gerçek dünya araştırması* (Ş. Çınkır ve N. Demirkasımoğlu, Çev. Ed.). Ankara: Anı.

Rogerson, C. ve Scott, E. (2010). The fear factor: How it affects students learning to program in a tertiary environment. *Journal of Information Technology Education: Research, 9*, 147-171.

Rosas, R., Nussbaum, M., Cumsille, P., Marianov, V., Correa, M., Flores, P., ... ve Salinas, M. (2003). Beyond Nintendo: design and assessment of educational video games for first and second grade students. *Computers & Education, 40*(1), 71-94.

Rose, S. (2019). *Developing children's computational thinking using programming games*. Sheffield Hallam University (United Kingdom).

Rountree, N., Rountree, J., Robins, A. ve Hannah, R. (2004). Interacting factors that predict success and failure in a CS1 course. *ACM SIGCSE Bulletin, 36*(4), 101-104.

Ryan, R. M. ve Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist, 55*(1), 68.

Sabri, Z., Fakhri, Y., ve Moumen, A. (2022). The Effects of Gamification on E-learning Education: Systematic Literature Review and Conceptual Model. *Statistics, Optimization & Information Computing, 10*(1), 75-92.

Salleh, F. H. M., Drus, S. M., Gunasekaran, S. S. ve Dewi, D. A. (2021, May). Evaluating the Effectiveness of Course Assessments Method in Determining Student's Competency Level in Programming. In *Journal of Physics: Conference Series* (Vol. 1878, No. 1, p. 012067). IOP Publishing.

Santana, A. D. O. ve Aranha, E. (2019). An approach to generate virtual tutors for game programming classes. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (pp. 246-252).

Santos, S. C., Tedesco, P. A., Borba, M. ve Brito, M. (2020). Innovative Approaches in Teaching Programming: A Systematic Literature Review. In *Proceedings of the 12th International Conference on Computer Supported Education* (pp. 205-2014).

Scacchi, W. (2011). Modding as a basis for developing game systems. In *Proceedings of the 1st international workshop on Games and software engineering* (pp. 5-8).

Scherer, R., Siddiq, F. ve Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of Educational Psychology, 111*(5), 764.

Schmuck, R. A. (2006). *Practical action research for change*. Corwin Press.

Seng, W. Y. ve Yatim, M. H. M. (2014). Computer game as learning and teaching tool for object oriented programming in higher education institution. *Procedia-Social and Behavioral Sciences, 123*, 215-224.

Scott, M. J. ve Ghinea, G. (2013). On the domain-specificity of mindsets: The relationship between aptitude beliefs and programming practice. IEEE Transactions on Education, 57(3), 169-174.

Scott, M. (2015). *Self-Beliefs in the Introductory Programming Lab and Games-based Fantasy Role-Play* (Doctoral dissertation, Brunel University).

Seaborn, K., Seif El-Nasr, M., Milam, D. ve Yung, D. (2012). Programming, PWNed: using digital game development to enhance learners' competency and self-efficacy in a high school computing science course. In *Proceedings of the 43rd ACM technical symposium on computer science education* (pp. 93-98).

Seralidou, E. ve Douligeris, C. (2021, September). Motivating students in distance programming learning using games. In *2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM) (pp. 1-7)*. IEEE.

Serrano-Cámara, L. M., Paredes-Velasco, M., Alcover, C. M. ve Velazquez-Iturbide, J. Á. (2014). An evaluation of students' motivation in computer-supported collaborative learning of programming concepts. *Computers in human behavior, 31*, 499-508.

Shaffer, S. C. ve Rosson, M. B. (2013). Increasing student success by modifying course delivery based on student submission data. *ACM inroads, 4*(4), 81-86.

Sharma, R. ve Shen, H. (2018). Does education culture in*fl*uence factors in learning programming: A comparative study between two universities across continents. *International Journal of Learning, Teaching and Educational Research, 17*(2), 1–24.

Sheetz, S. D., Irwin, G., Tegarden, D. P., Nelson, H. J. ve Monarchi, D. E. (1997). Exploring the difficulties of learning object-oriented techniques. *Journal of Management Information Systems, 14*(2), 103-131.

Sillaots, M. ve Fiadotau, M. (2018). Using project-based learning to teach learning game design: The example of LIFE project. In *Proceedings of the 12th European Conference on Game-Based Learning, ACPI, Reading* (pp. 590-599).

Simpson, O. (2013). *Supporting students for success in online and distance education.* Routledge.

Singer, L. ve Schneider, K. (2012) It was a bit of a race: Gamification of version control. In: *International Workshop on Games and Software Engineering* 2, pp. 5–8.

Sisovic, S., Matetic, M. ve Bakaric, M. B. (2015). Mining student data to assess the impact of moodle activities and prior knowledge on programming course success. In *Proceedings of the 16th International Conference on Computer Systems and Technologies* (pp. 366-373).

Skraparli, G., Karavidas, L. ve Tsiatsos, T. (2022). Dynamic Serious Game for Developing Programming Skills. In *Interactive Mobile Communication, Technologies and Learning* (pp. 580-592). Springer, Cham.

Smaragdina, A. A., Nidhom, A. M., Soraya, D. U., Ningrum, G. D. K. ve Akbar, M. I. (2020). Educational Platformer Game (OOP-EduGame) to Enhance Student Motivation

and Understanding in the Object Oriented Programming Subject. *International Journal of Applied Business and Information Systems, 4*(2), 132-141.

Soares, A., Martin, N. L. ve Fonseca, F. (2015). Teaching introductory programming with game design and problem-based learning. *Issues in Information Systems, 16*(3).

Sorva, J. ve Seppälä, O. (2014, November). based design of the first weeks of CS1. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research* (pp. 71-80).

Sotamaa, O. (2010). When the game is not enough: Motivations and practices among computer game modding culture. *Games and Culture, 5*(3), 239-255.

Sousa, M. (2021). Serious board games: modding existing games for collaborative ideation processes. *International Journal of Serious Games, 8*(2), 129-146.

Souza, M. R. D. A., Veado, L. F., Moreira, R. T., Figueiredo, E. M. L. ve Costa, H. A. X. (2017, May). Games for learning: Bridging game-related education methods to software engineering knowledge areas. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)* (pp. 170-179). IEEE.

Spielberger, C. D. (Ed.). (2013). Anxiety and behavior. *Academic Press*.

Spieler, B. ve Slany, W. (2018). Game development-based learning experience: Gender differences in game design. *arXiv preprint arXiv*:1805.04457.

Spieler, B., Grandl, M., Ebner, M. ve Slany, W. (2019). "Computer Science for all": Concepts to engage teenagers and non-CS students in technology. *arXiv preprint arXiv*:1908.06637.

Spohrer, J.C. ve Soloway E. (1986). Novice mistakes: Are the folk wisdoms correct? *Communications of the ACM, 29*(7), 624-632.

Stenerson, M. E. (2012). *Academic game development: practices and design strategies for creating STEM games* (Doctoral dissertation, Iowa State University).

Stoffová, V. (2019). Learning Object-Oriented Programming by Creating Games. In *The International Scientific Conference eLearning and Software for Education* (Vol. 1, pp. 20-29). " Carol I" National Defence University.

Stringer, E.T. (2004). *Action research in education*. Upper Saddle River, NJ: Pearson Prentice Hall.

Stolee, K.T. (2010). Kodu language and grammar specification. *Microsoft Research whitepaper*.

Sullivan, F. R. (2008). Robotics and science literacy: Thinking skills, science process skills and systems understanding. Journal of Research in Science Teaching: *The Official Journal of the National Association for Research in Science Teaching, 45*(3), 373-394.

Sunday, K., Wong, S. Y., Samson, B. O. ve Sanusi, I. T. (2022). Investigating the effect of imikode virtual reality game in enhancing object oriented programming concepts among university students in Nigeria. *Education and Information Technologies*, 1-27.

Sung, K., Hillyard, C., Angotti, R. L., Panitz, M. W., Goldstein, D. S. ve Nordlinger, J. (2010). Game-themed programming assignment modules: A pathway for gradual integration of gaming context into existing introductory programming courses. *IEEE Transactions on Education, 54*(3), 416-427.

Taber, K. S. (2018). The use of Cronbach's alpha when developing and reporting research instruments in science education. *Research in science education, 48*(6), 1273-1296.

Tan, P. H., Ting, C. Y. ve Ling, S. W. (2009, November). Learning difficulties in programming courses: undergraduates' perspective and perception. In *2009 International Conference on Computer Technology and Development* (Vol. 1, pp. 42-46). IEEE.

Tanielu, T., 'Akau'ola, R., Varoy, E. ve Giacaman, N. (2019). Combining analogies and virtual reality for active and visual object-oriented programming. In *Proceedings of the acm conference on global computing education* (pp. 92-98).

Tavşancıl, E. (2002). *Tutumların Ölçülmesi ve SPSS İle Veri Analizi*. Nobel Yayıncılık: Ankara.

Tayebi, A., Gómez, J. ve Delgado, C. (2021). Analysis on the lack of motivation and dropout in engineering students in Spain. *IEEE Access, 9*, 66253-66265.

Tek, F. B., Benli, K. S. ve Deveci, E. (2018). Implicit theories and self-efficacy in an introductory programming course. *IEEE Transactions on Education, 61*(3), 218-225.

Teodosieva, G. ve Teodosiev, T. (2021). Developing games while learning programming. *Mathematical and Software Engineering, 7(1-2)*, 1-6.

Tervo, J. (2019). Development of supplementary material for teaching game development.

Theodoropoulos, A. ve Lepouras, G. (2022). Game design, gender and personalities in programming education. *Frontiers in Computer Science*, 5.

Thomas, L., Woodbury, J. ve Jarman, E. (2002). Learning styles and performance in the introductory programming sequence. *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education.* ACM Press, NY, 33-37

Thota, N ve Whitfield, R. (2010). Holistic approach to learning and teaching introductory object-oriented programming. *Computer Science Education*, *20*(2), 103-127.

Thramboulidis, K. (2003). A constructivism-based approach to teach object-oriented programming. Journal of Informatics Education and Research, 5(1), 1-12.

Thuné, M. ve Eckerdal, A. (2019). Analysis of Students' learning of computer programming in a computer laboratory context. *European Journal of Engineering Education, 44*(5), 769-786.

Togashi, G. (2019). *Motivating Programming Learners through Game Development*.

Tomaiuolo, M., Angiani, G., Ferrari, A., Mordonini, M. ve Poggi, A. (2018). A Week of Playing with Code, the Object-Oriented Way. In *International Conference in Methodologies and intelligent Systems for Techhnology Enhanced Learning* (pp. 62-69). Springer, Cham.

Topîrceanu, A. (2017). Gamified learning: A role-playing approach to increase student in-class motivation. *Procedia Computer Science,* 112, 41-50.

Toukiloglou, P. ve Xinogalos, S. (2022). Ingame worked examples support as an alternative to textual instructions in serious games about programming. *Journal of Educational Computing Research*, 07356331211073655.

Tsai, C. Y. (2019). Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy. *Computers in Human Behavior, 95*, 224-232.

Tsay, C. H. H., Kofinas, A. K., Trivedi, S. K. ve Yang, Y. (2020). Overcoming the novelty effect in online gamified learning systems: An empirical evaluation of student engagement and performance. *Journal of Computer Assisted Learning*, *36*(2), 128-146.

Tüysüz, C. (2010). The Effect of the Virtual Laboratory on Students' Achievement and Attitude in Chemistry. *International Online Journal of Educational Sciences*, *2*(1).

Unity Documantation. (2022). MonoBehaviour.Awake(). https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html. Erişim Tarihi: 20.04.2022

Uzun, E. (2012). *An Action research on design, delivery and evaluation of a distance course in a vocational higher education institution* [Ph.D. - Doctoral Program]. Middle East Technical University.

Vagle, O., Palmason, V. J., Dautaj, G. ve Lysø, O. K. L. (2021). *Video Game Modding's Potential for Teaching Programming* (Bachelor's thesis, NTNU).

Vassilev, T. I. (2015). An Approach to Teaching Introductory Programming for IT Professionals Using Games. *International Journal of Human Capital and Information Technology Professionals (IJHCITP), 6*(1), 26-38.

Vaughn, S., Schumm, J. S. ve Sinagub, J. (1996). *Focus group interviews in education and psychology*. London: Sage.

Ventura, P.R. (2005). Identifying predictors of success for and objects-*f*irst CS1. *Computer Science Education, 15* (3), 223–243

Ventura, P. ve Ramamurthy, B. (2004). Wanted: CS1 students. No experience required. *ACM SIGCSE Bulletin, 36*(1), 240-244.

Vihavainen, A., Airaksinen, J. ve Watson, C. (2014). A systematic review of approaches for teaching introductory programming and their influence on success. In *Proceedings of the tenth annual conference on International computing education research* (pp. 19-26). ACM.

Vincenti, G., Braman, J. ve Hilberg, J. S. (2013). Teaching introductory programming through reusable learning objects: a pilot study. *Computing Sciences in Colleges*, 1-6.

Vitasari, P., Wahab, M. N. A., Othman, A., Herawan, T. ve Sinnadurai, S. K. (2010). The relationship between study anxiety and academic performance among engineering students. *Procedia-Social and Behavioral Sciences, 8*, 490-497.

Vygotsky, L. S. (1978). *Mind and society: The development of higher mental processes*. Cambridge, MA: Harvard University Press.

Wallinga, M. (2019). Engaging CS2 students via a semester-long in-class game project. *Journal of Computing Sciences in Colleges, 35*(2), 77-88.

Wang, A. I. ve Wu, B. (2009). An application of a game development framework in higher education. *International Journal of Computer Games Technology*, 2009.

Wang, A. I. ve Wu, B. (2015). Use of game development in computer science and software engineering education. CooperK. ML ScacchiW.(Eds.), *Computer Games and Software Engineering,* 31-58.

Watson, C., Li, F. W. ve Lau, R. W. (2011). Learning programming languages through corrective feedback and concept visualisation. In *International Conference on Web-Based Learning (pp. 11-20)*. Springer, Berlin, Heidelberg.

Watson, C., Li, F. W. ve Godwin, J. L. (2014). No tests required: comparing traditional and dynamic predictors of programming success. In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 469-474).

Wekerle, C., Daumiller, M. ve Kollar, I. (2022). Using digital technology to promote higher education learning: The importance of different learning activities and their relations to learning outcomes. *Journal of Research on Technology in Education, 54*(1), 1-17.

Wiedenbeck, S. (2005). Factors affecting the success of non-majors in learning to program. In *Proceedings of the first international workshop on Computing education research* (pp. 13-24).

Wiedenbeck, S., Labelle, D. ve Kain, V. N. (2004). Factors affecting course outcomes in introductory programming. In *PPIG* (p. 11).

Wilcox, C. ve Lionelle, A. (2018, February). Quantifying the benefits of prior programming experience in an introductory computer science course. In *Proceedings of the 49th acm technical symposium on computer science education* (pp. 80-85).

Wilson, B. C. ve Shrock, S. (2001). Contributing to success in an introductory computer science course: a study of twelve factors. *ACM sigcse bulletin, 33*(1), 184-188.

Wong, Y. S., Hayati, M., Yatim, M. ve Hoe, T. W. (2017). A propriety game based learning mobile game to learn object-oriented programming—Odyssey of Phoenix. In *2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering* (TALE) (pp. 426-431). IEEE.

Wong, W. K., Liu, W. T., Yin, S. K., Yang, H. H. ve Chao, T. K. (2016) Collaboration of Multidisciplinary Students on the Development of Mobile Games.

Wong, Y. S. ve Tan, W. H. (2016). Examining Effectiveness of Learning Object-Oriented Programming Paradigm Through Propriety Game-Based Learning Games. In *European Conference on Games Based Learning (p. 796).* Academic Conferences International Limited.

Wong, Y. S. ve Yatim, M. H. M. (2018). A Propriety Multiplatform Game-Based Learning Game to Learn Object-Oriented Programming. In *2018 7th International Congress on Advanced Applied Informatics (IIAI-AAI)* (pp. 278-283). IEEE.

Wood, Z. ve Keen, A. (2015, February). Building worlds: Bridging imperative-first and object-oriented programming in cs1-cs2. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 144-149).

Woszczynski, A. B., Haddad, H. M. ve Zgambo, A. (2005). An IS students worst nightmare: Programming courses. *In Proceedings of the Southern Association of Information Systems Conferences (pp. 130-133).*

Wu, B. ve Wang, A. I. (2012). A guideline for game development-based learning: a literature review. *International Journal of Computer Games Technology*, 2012.

Xinogalos, S. (2015). Object-oriented design and programming: an investigation of novices' conceptions on objects and classes. *ACM Transactions on Computing Education (TOCE), 15*(3), 1-21.

Xu, D., Wolz, U., Kumar, D. ve Greenburg, I. (2018). Updating introductory computer science with creative computation. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 167-172).

Yan, L. (2009). Teaching object-oriented programming with games. In *Procs 6th Int Conf on Information Technology: New Generations.*

Yang, J., Lee, Y., Hicks, D. ve Chang, K. H. (2015). Enhancing object-oriented programming education using static and dynamic visualization. In *2015 IEEE Frontiers in Education Conference (FIE) (pp. 1-5)*. IEEE.

Yang, J., Lee, Y. ve Chang, K. H. (2018). Evaluations of JaguarCode: A web-based object-oriented programming environment with static and dynamic visualization. *Journal of Systems and Software*, 145, 147-163.

Yaşar, M. (2014). İstatistiğe Yönelik Tutum Ölçeği: Geçerlilik ve Güvenirlik Çalışması. *Pamukkale Üniversitesi Eğitim Fakültesi Dergisi*, *36*, 59-75

Yilmaz Ince, E. ve Koc, M. (2021). The consequences of robotics programming education on computational thinking skills: An intervention of the Young Engineer's Workshop (YEW). *Computer Applications in Engineering Education, 29*(1), 191-208.

Yıldırım, A. ve Şimşek, A. (2011). *Sosyal bilimlerde nitel araştırma yöntemleri (8. Ed.).* Ankara: Seçkin Yayınevi.

Yılmaz, F. ve Gültekin, M. (2007). Proje tabanlı öğrenmenin beşinci sınıf fen bilgisi dersinde öğrenme ürünlerine etkisi. *İlköğretim online, 6*(1), 93-112.

Yukselturk, E. ve Altiok, S. (2017). An investigation of the effects of programming with Scratch on the preservice IT teachers' self-efficacy perceptions and attitudes towards computer programming. *British Journal of Educational Technology, 48*(3), 789-801.

Yurdagül, C. (2018). *The Effect of flipped classroom as a teaching strategy on undergraduate students' self-efficacy, engagement and attitude in a computer programming course* [Ph.D. - Doctoral Program]. Middle East Technical University.

Yurdugül, H. ve Aşkar, P. (2013). Learning programming, problem solving and gender: A longitudinal study. *Procedia-Social and Behavioral Sciences, 83*, 605-610.

Zhang, X., Zhang, C., Stafford, T. F. ve Zhang, P. (2013). Teaching introductory programming to IS students: The impact of teaching approaches on learning performance. *Journal of Information Systems Education, 24*(2), 147-155.Z. Sayin and S. Seferoglu, Coding education as a new 21st

EKLER

Ek 1. Katılımcı Bilgi Formu

Sevgili öğrenciler,
Nesne Yönelimli Programlama dersini daha verimli ve eğlenceli hale getirmek için derste oyun programlamayı içeren bir çalışma yapmaktayım. Sorulara vereceğiniz yanıtlar araştırmada kullanılacak olup başka bir kimse ile paylaşılmayacaktır. Katkılarınız için teşekkür ederim.

Osman Gazi YILDIRIM
Öğretim Görevlisi

1. Kişisel Bilgiler:

1. Adınız Soyadınız:
2. Yaş Aralığınız:

   ( ) 17-24

   ( ) 25-29

   ( ) 30-39

   ( ) 40 ve üzeri
3. Mezun Olduğunuz Lise Türü:

2. Bilgisayar Kullanımı ve Bilgisayar Oyunları:

1. Kişisel bilgisayarınız var mı?

   ( ) Evet

   ( ) Hayır
2. Bilgisayarı hangi sıklıkla kullanırsınız?

   ( ) Nadir

   ( ) Orta sıklıkta

   ( ) Sık

   ( ) Çok sık

3. Dijital oyunları hangi sıklıkla oynarsınız?

( ) Hiç

( ) Günde 1 saatten az

( ) Günde 1-2 saat

( ) Günde 2 saatten fazla

3. Programlama Bilgisi:

1. Bölümünüze kayıt olmadan önce programlama dersi aldınız mı?

( ) Hayır

( ) Bir tane

( ) İki tane

( ) Üç ve daha fazla

2. Şu anki programlama bilgi ve becerinizi hangi seviyede tanımlarsınız?

( ) Çok düşük

( ) Düşük

( ) Orta

( ) Yüksek

( ) Çok yüksek

3. Programlama öğrenmenin zorluk seviyesini nasıl tanımlarsınız?

( ) Çok kolay

( ) Kolay

( ) Orta zorlukta

( ) Zor

( ) Çok zor

4. Unity 3D ve Visual Studio dışında bildiğiniz tasarım, geliştirme veya kodlama

programlarını yazınız (Örn. Photoshop, Android Studio vb.)

5. C# dışında bildiğiniz programlama dillerini (mobil ve web yazılımları dâhil) yazınız.

4. Programlama Hakkındaki Düşünceler:

1. Programlama öğrenmenin yararlı olduğunu düşünüyor musunuz? Neden?
2. Programlama öğrenmeyi keyifli buluyor musunuz? Neden?
3. Programlama derslerinde öğrendiklerinizin akılda kalıcı olduğunu düşünüyor musunuz? Neden?

5. Oyun Programlama Hakkındaki Düşünceler:

1. Daha önce Unity 3D'yi herhangi bir amaçla kullandınız mı? Kullandıysanız hangi amaçlarla kullandınız?
2. Daha önce herhangi bir oyun geliştirdiniz mi? Cevabınız evet ise geliştirdiğiniz oyunu ya da oyunları kısaca açıklar mısınız?
3. Programlamayı oyun programlayarak öğrenmenin yararlı olacağını düşünüyor musunuz?

( ) Kesinlikle katılmıyorum

( ) Katılmıyorum

( ) Kararsızım

( ) Katılıyorum

( ) Kesinlikle Katılıyorum

Ek 2. NYP Başarı Sınavı Madde Belirtke Tablosu

Boyut	Kazanımlar	NYP Başarı Sınavı Soru Numarası
TEORİK SINAV	Sınıf kavramını örneklerle açıklar.	Bölüm 1-Md. a
	Nesne kavramını örneklerle açıklar.	Bölüm 1-Md. b
	Özellik (attribute) kavramını örneklerle açıklar.	Bölüm 1-Md. c
	Davranış (behaviour) kavramını örneklerle açıklar.	Bölüm 1-Md. d
	Kurucu metot kavramını örneklerle açıklar.	Bölüm 1-Md. e
	Sınıf ile nesne arasındaki ilişkiyi betimler.	Bölüm 2-Md. a
	Kalıtımın (inheritance) kullanım amaçlarını örnek vererek açıklar.	Bölüm 2-Md. b
	Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar.	Bölüm 2-Md. c
	Kapsüllemenin (encapsulation) kullanım amaçlarını örnek vererek açıklar.	Bölüm 2-Md. d
	Statik sınıflar ile normal sınıfları karşılaştırır.	Bölüm 2-Md. e
	C# programlama dilinde kullanılan erişim belirleyicilerin kullanım amaçlarını belirtir.	Bölüm 2-Md. f
	Verilen UML Sınıf diyagramlarında yer alan sınıflara ait özellik ve davranışlarını çözümleyerek bu sınıflar arasındaki ilişkileri açıklar.	Bölüm 2-Md. g
	Verilen bir C# sınıfını analiz ederek bu sınıfta kullanılan NYP prensiplerini (kalıtım, kapsülleme, çokbiçimlilik vb.) çözümler.	Bölüm 2-Md. h
UYGULAMA SINAVI	Bir sınıfa ait özelliklerin erişim belirleyicilerini verilen koşula göre değiştirir.	Bölüm 3-Soru 1
	Önceden oluşturulmuş bir C# sınıfını, belirtilen oyun nesnesine bileşen (component) olarak ekler.	Bölüm 3-Soru 2
	UML sınıf diyagramı verilen arayüzü (interface), C# programlama dilinde oluşturur.	Bölüm 3-Soru 3
	UML sınıf diyagramında belirtilen çokbiçimli metotları oluşturur ve kullanır.	Bölüm 3-Soru 4
	Verilen koşullara göre kalıtım mekanizmasını C# programlama dilinde gerçekleştirir.	Bölüm 3-Soru 4
	UML sınıf diyagramı verilen sınıfı C# programlama dilinde oluşturur.	Bölüm 3-Soru 5/6
	Önceden oluşturulmuş bir C# sınıfına ait özelliklerin değer atamasını verilen koşula göre yapar.	Bölüm 3-Soru 6- Md. a
	Bir C# sınıfına ait Monobehavior metotlarını, verilen koşula göre oluşturur.	Bölüm 3- Soru 6- Md. b/c/d
	Kullanıcı arayüzü (UI) nesnelerini, oluşturduğu C# sınıfında kullanır.	Bölüm 3- Soru 6- Md. e

Ek 3. NYP Başarı Sınavı Kapsam Geçerliliği Uzman Görüş Formu

Sayın uzman,

Bu kapsam geçerliliği uzman görüş formu, doktora tezim kapsamında veri toplama aracı olarak kullanılacak NYP Başarı Sınavının kapsam geçerliliğini kontrol edebilmek amacıyla hazırlanmıştır. NYP Başarı Sınavı toplam 22 sorudan oluşmakta ve 21 farklı kazanımı kontrol etmektedir.

İkinci sayfadan itibaren göreceğiniz tablonun ilk sütununda NYP Başarı Sınavının soruları yer almaktadır. İkinci sütunda ilgili sorunun ölçmeye amaçladığı kazanımlar yer almaktadır. Başarı sınavında yer alan bir soru yalnızca bir kazanımı test edebileceği gibi birden fazla kazanımı da test edebilmektedir. Tablonun üçüncü sütunundan itibaren ilgili kazanımın uygunluğunu değerlendireceğiniz bölüm yer almaktadır.

NYP Başarı Sınavında yer alan sorunun ilgili kazanımı ölçtüğünü düşünüyorsanız "Uygun (U)" sütununa çarpı işareti koyabilirsiniz. Başarı sınavında yer alan sorunun düzenlenmesi gerektiğini düşünüyorsanız "Küçük Düzeltilmeler Yapılmalı (KDY) ya da Büyük Düzeltmeler Yapılmalı (BDY)" sütununa çarpı koyabilirsiniz. Başarı sınavında yer alan sorunun ilgili kazanımı test etmediğini ya da sorunun gereksiz olduğunu düşünüyorsanız "Uygun Değil (UD)" sütununa çarpı koyabilirsiniz. BDY, KDY ya da UD sütunlarına işaret koymanız durumunda Açıklama bölümüne önerinizi ya da düşüncelerinizi yazmanızı rica ederim.

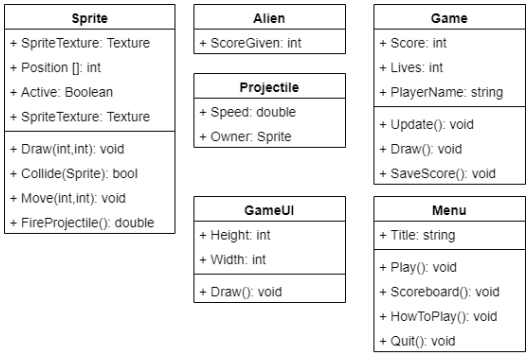
Değerli katkılarınızdan dolayı teşekkürlerimi sunarım.

Öğr.Gör. Osman Gazi YILDIRIM

Kısaltmalar ve Renklendirme

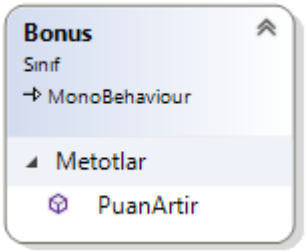
Uygun	U.
Küçük Düzeltmeler Yapılmalı	KDY.
Büyük Düzeltmeler Yapılmalı	BDY.
Uygun Değil	UD.
Açıklama	-

NYP Başarı Sınavı Sorusu	Kazanım	UYGUNLUK DURUMU				
		U.	KDY.	BDY	UD.	Açıklama
Aşağıda Nesne Yönelimli Programlama ile ilgili bazı kavramlar verilmiştir. Bu kavramları birkaç cümle ile açıklayınız.						
1A) Sınıf:	Sınıf kavramını örneklerle açıklar.					
1B) Nesne:	Nesne kavramını örneklerle açıklar.					
1C) Özellik (attribute):	Özellik (attribute) kavramını örneklerle açıklar.					
1D) Kurucu metot:	Kurucu metot kavramını örneklerle açıklar.					
1E) Davranış (behaviour):	Davranış (behaviour) kavramını örneklerle açıklar.					
2A) Sınıf ile nesne arasında nasıl bir ilişki bulunmaktadır? Örnek vererek açıklayınız.	Sınıf ile nesne arasındaki ilişkiyi betimler.					
2B) Kalıtım nedir? NYP'de kalıtım hangi amaçlarla kullanılır? Örnek vererek açıklayınız.	Kalıtımın (inheritance) kullanım amaçlarını örnek vererek açıklar.					
2C) Çokbiçimlilik nedir? NYP'de çokbiçimlilik hangi amaçlarla kullanılır? Örnek vererek açıklayınız.	Çokbiçimliliğin (polymorphism) kullanım amaçlarını örnek vererek açıklar.					
2D) Kapsülleme nedir? Ne amaçla kullanılır? Açıklayınız.	Kapsüllemenin (encapsulation) kullanım amaçlarını örnek vererek açıklar.					
2E) Nesne Yönelimli Programlamada statik sınıflar ne amaçla kullanılır? Statik ile statik olmayan sınıfların arasında ne tür farklılıklar bulunmaktadır? Açıklayınız.	Statik sınıflar ile normal sınıfları karşılaştırır.					
2F) C#'ta kullanılan erişim belirleyicileri listeleyerek kullanım amaçlarını açıklayınız.	C# programlama dilinde kullanılan erişim belirleyicileri sınıflandırır.					
2G) Aşağıda bir C# oyun projesinin UML sınıf diyagramları yer almaktadır. Bu projede yer alan	Verilen UML Sınıf diyagramlarında yer alan sınıflara					

sınıf diyagramları modellerini açıklayarak sınıflar arasındaki ilişkileri belirtiniz.  	ait özellik ve davranışlarını çözümleyerek bu sınıflar arasındaki ilişkileri açıklar.
2G) Aşağıda bir C# projesinin sınıf tanımlamaları yer almaktadır. Bu projede Nesne Yönelimli Programlama prensiplerinden hangileri kullanılmıştır? Açıklayınız. `abstract class Figure { public abstract double CalcSurface(); } class Rectangle : Figure { public double Width { get; set; } public double Height { get; set; } public override double CalcSurface() { return this.Width * this.Height; } } class Square : Figure { public double Size { get; set; } public override double CalcSurface() { return this.Size * this.Size; } } class Geometry {`	Verilen bir C# sınıfını analiz ederek bu sınıfta kullanılan NYP prensiplerini (kalıtım, kapsülleme, çokbiçimlilik vb.) çözümler.

```
        static void Main()
        {
            Figure[] figures =
new Figure[] {
                new Square() {
Size = 3 },
                new Rectangle()
{ Width = 2, Height = 3 },
            };

            foreach (Figure
figure in figures)
            {

Console.WriteLine("Figure = {0},
Surface = {1}",

figure.GetType().Name,

figure.CalcSurface());
            }
     }
    }
```

3A) Assets\Scripts klasöründeki PlayerController adındaki sınıfa ait speed isimli sınıf değişkeninin erişim belirleyicisini, diğer sınıfların erişemeyeceği şekilde ayarlayınız.	Bir sınıfa ait özelliklerin erişim belirleyicilerini verilen koşula göre değiştirir.
3B) Ana kameranın Kamyon isimli nesneyi takip edebilmesi için Assets\Scripts klasöründeki FollowPlayer isimli sınıfı, bileşen olarak ekleyiniz.	Önceden oluşturulmuş bir C# sınıfını, belirtilen oyun nesnesine bileşen (component) olarak ekler.
3C) Aşağıdaki UML sınıf diyagramı verilen arayüzü, Assets\Scripts klasöründe oluşturunuz.	UML sınıf diyagramı verilen arayüzü (interface), C# programlama dilinde oluşturur.
3D) Aşağıdaki UML sınıf diyagramları verilen sınıfları Assets\Scripts klasöründe oluşturarak gerekli kalıtım mekanizmasını gerçekleştiriniz.	UML sınıf diyagramı verilen sınıfı C# programlama dilinde oluşturur.

	Verilen koşullara göre kalıtım mekanizmasını C# programlama dilinde gerçekleştirir.
3E) Aşağıda UML sınıf diyagramı verilen Bonus ismindeki sınıfı Assets\Scripts klasöründe oluşturunuz. 	UML sınıf diyagramı verilen sınıfı C# programlama dilinde oluşturur.
Aşağıda yer alan sorular, UML sınıf diyagramı verilen ve Assets\Scripts klasöründe yer alan PuanSistemi ismindeki sınıf kullanılarak gerçekleştirilecektir.	
3F-i) Oyun başladığında bir defaya mahsus olarak çalışan Monobehaviour metodunda Puan ve bonusSayisi değişkenlerinin değerlerinin sıfır olmasını sağlayınız.	Önceden oluşturulmuş bir C# sınıfına ait özelliklerin değer

	atamasını verilen koşula göre yapar.
3F-ii) OnTriggerEnter() sınıf metodunda Bonus nesneleriyle çarpışma durumunda bonusSayisi değişkeninin değerinin bir, Puan değişkenin değerinin 10 puan artırılmasını sağlayınız.	Bir C# sınıfına ait Monobehavior metotlarını, verilen koşula göre oluşturur.
3F-iii) OnCollisionEnter() sınıf metodunda Bidon nesnesiyle çarpışma durumunda Puan değişkeninin değerinin 20 puan azalmasını sağlayınız.	Bir C# sınıfına ait Monobehavior metotlarını, verilen koşula göre oluşturur.
3F-iv) OnCollisionEnter() sınıf metodunda Bariyer nesnesiyle çarpışma olması durumunda Puan değişkeninin değerinin 30 puan azalmasını sağlayınız.	Bir C# sınıfına ait Monobehavior metotlarını, verilen koşula göre oluşturur.
3F-v) Update() sınıf metodunda Puan ve bonusSayisi değişkenlerinin PuaniYazdir() ve BonusYazdir() sınıf metotları vasıtasıyla ekrana yazdırılmasını sağlayınız.	Kullanıcı arayüzü (UI) nesnelerini, oluşturduğu C# sınıfında kullanır.

Ek 4. NYP Başarı Sınavı

Talimatlar: Sınav süreniz **120 dakikadır**. Sınav kâğıdınıza kimlik bilgilerinizi doğru olarak yazınız. Sınav kitapçığınız toplam üç bölümden ve altı sayfadan oluşmaktadır. Her bölümün karşısında puan değeri yer almaktadır. Sınav kitapçığını ve soruları kontrol ediniz.

Bölüm 1: Tanımlamalar (10 Puan)

Talimatlar: Aşağıda Nesne Yönelimli Programlama ile ilgili bazı kavramlar verilmiştir. Bu kavramları örneklerle açıklayınız. **Her soru 2 puandır.**

a. Sınıf:

b. Nesne:

c. Özellik (attribute):

d. Kurucu metot:

e. Davranış (behaviour):

Bölüm 2: Açık Uçlu Sorular (40 Puan)

Talimatlar: Aşağıda yer alan soruları yanıtlayınız. **Her soru 5 puandır.**

a. Sınıf ile nesne arasında nasıl bir ilişki bulunmaktadır? Örnek vererek açıklayınız.

b. Kalıtım nedir? NYP'de kalıtım hangi amaçlarla kullanılır? Örnek vererek açıklayınız.

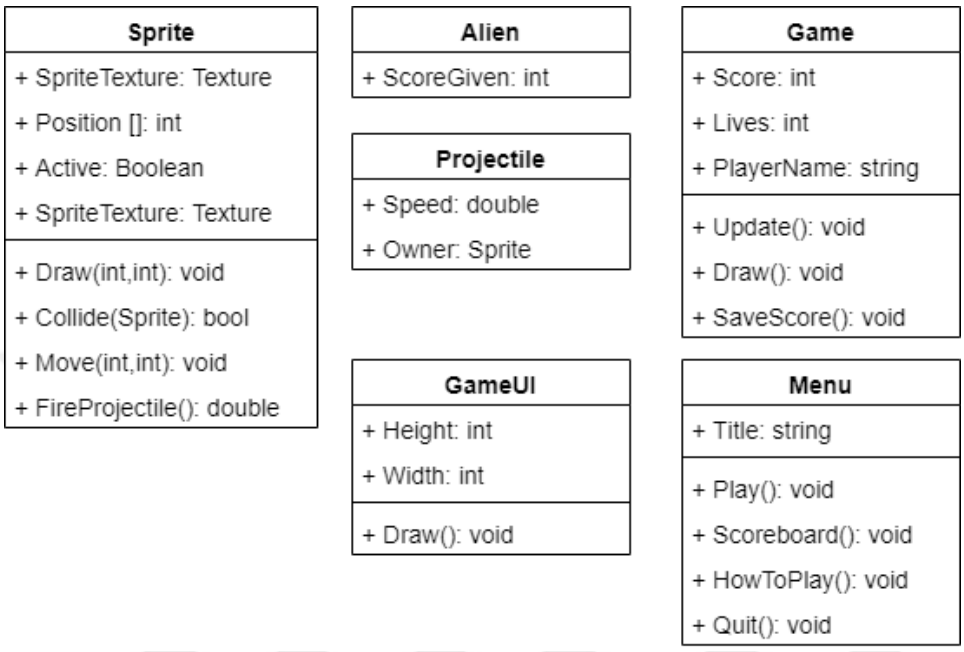
c. Çokbiçimlilik nedir? NYP'de çokbiçimlilik hangi amaçlarla kullanılır? Örnek vererek açıklayınız.

d. Kapsülleme nedir? Ne amaçla kullanılır? Açıklayınız.

e. Nesne Yönelimli Programlamada statik sınıflar ne amaçla kullanılır? Statik ile statik olmayan sınıfların arasında ne tür farklılıklar bulunmaktadır? Açıklayınız.

f. C#'ta kullanılan erişim belirleyicileri listeleyerek kullanım amaçlarını açıklayınız.

g. Aşağıda bir C# oyun projesinin UML sınıf diyagramları yer almaktadır. Bu projede yer alan sınıfların modellerini açıklayarak sınıflar arasındaki ilişkileri belirtiniz.



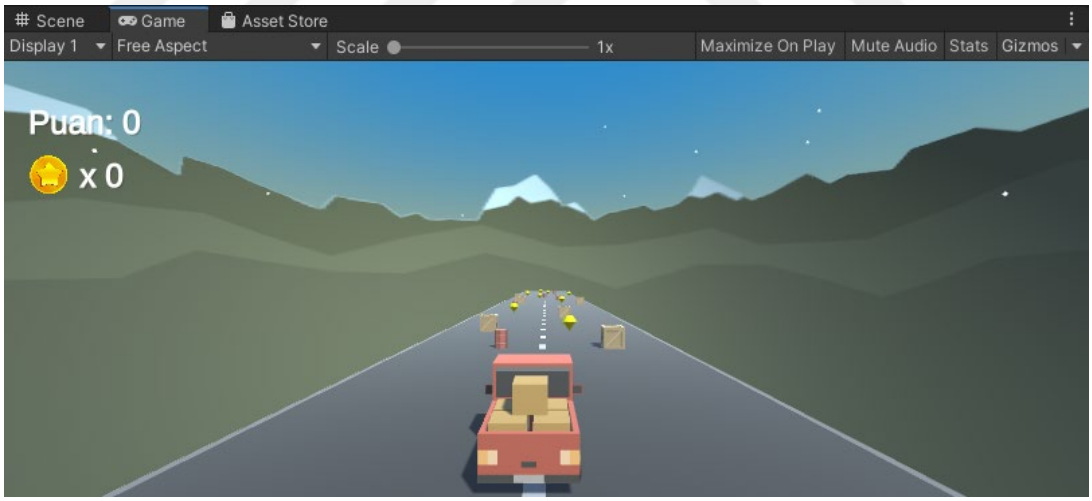
h. Aşağıda bir C# projesinin sınıf tanımlamaları yer almaktadır. Bu projede Nesne Yönelimli Programlama prensiplerinden hangileri kullanılmıştır? Açıklayınız.

```
abstract class Figure
{
    public abstract double CalcSurface();
}


class Rectangle : Figure
{
    public double Width { get; set; }
    public double Height { get; set; }

    public override double CalcSurface()
    {
        return this.Width * this.Height;
    }
}

class Square : Figure
{
    public double Size { get; set; }

    public override double CalcSurface()
    {
        return this.Size * this.Size;
```

```
    }
}

class Geometry
{
    static void Main()
    {
        Figure[] figures = new Figure[] {
            new Square() { Size = 3 },
            new Rectangle() { Width = 2, Height = 3 },
        };

        foreach (Figure figure in figures)
        {
            Console.WriteLine("Figure = {0}, Surface = {1}",
                figure.GetType().Name,
                figure.CalcSurface());
        }
 }

}
```

Bölüm 3: Uygulama Sınavı (50 Puan)

Talimatlar: Size Unity 3D'de oluşturulmuş bir oyun prototipi verilmiştir. Sizden bu oyun prototipini aşağıdaki görevler kapsamında değiştirmeniz istenmektedir. Oyun prototipinin arayüzü aşağıda sunulmuştur.



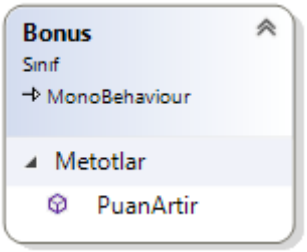
1) Assets\Scripts klasöründeki PlayerController adındaki sınıfa ait speed isimli sınıf değişkeninin erişim belirleyicisini, diğer sınıfların erişemeyeceği şekilde ayarlayınız. **(2 Puan)**

2) Ana kameranın Kamyon isimli nesneyi takip edebilmesi için Assets\Scripts klasöründeki FollowPlayer isimli sınıfı, bileşen olarak ekleyiniz. **(3 Puan)**

3) Aşağıdaki UML sınıf diyagramı verilen arayüzü, Assets\Scripts klasöründe oluşturunuz. **(5 Puan)**

4) Aşağıdaki UML sınıf diyagramları verilen sınıfları Assets\Scripts klasöründe oluşturarak gerekli kalıtım mekanizmasını gerçekleştiriniz. **(10 Puan)**

5) Aşağıda UML sınıf diyagramı verilen Bonus ismindeki sınıfı Assets\Scripts klasöründe oluşturunuz. **(5 Puan)**



6) Aşağıda yer alan sorular, UML sınıf diyagramı verilen ve Assets\Scripts klasöründe yer alan PuanSistemi ismindeki sınıf kullanılarak gerçekleştirilecektir.

a) Oyun başladığında bir defaya mahsus olarak çalışan Monobehaviour metodunda Puan ve bonusSayisi değişkenlerinin değerlerinin sıfır olmasını sağlayınız. **(5 Puan)**

b) OnTriggerEnter() sınıf metodunda Bonus nesneleriyle çarpışma durumunda bonusSayisi değişkeninin değerinin bir, Puan değişkenin değerinin 10 puan artırılmasını sağlayınız. **(5 Puan)**

c) OnCollisionEnter() sınıf metodunda Bidon nesnesiyle çarpışma durumunda Puan değişkeninin değerinin 20 puan azalmasını sağlayınız. **(5 Puan)**

d) OnCollisionEnter() sınıf metodunda Bariyer nesnesiyle çarpışma olması durumunda Puan değişkeninin değerinin 30 puan azalmasını sağlayınız. **(5 Puan)**

e) Update() sınıf metodunda Puan ve bonusSayisi değişkenlerinin PuaniYazdir() ve BonusYazdir() sınıf metotları vasıtasıyla ekrana yazdırılmasını sağlayınız. **(5 Puan)**

Ek 5. Proje Değerlendirme Rubriği Kapsam Geçerliliği Uzman Görüş Formu

Sayın uzman,

Bu kapsam geçerliliği uzman görüş formu, doktora tezim kapsamında veri toplama aracı olarak kullanılacak Proje Değerlendirme Rubriğinin kapsam geçerliliğini kontrol edebilmek amacıyla hazırlanmıştır. Bu rubrik 10 ölçüte sahip olup her bir ölçüt 5 farklı başarı derecesine sahiptir.

İkinci sayfadan itibaren göreceğiniz tablonun ilk sütununda Proje Değerlendirme Rubriğine ait ölçütler yer almaktadır. Tablonun altıncı sütunundan itibaren ilgili başarı kriterinin uygunluğunu değerlendireceğiniz bölüm yer almaktadır. Proje Değerlendirme Rubriğinde yer alan ölçütlerin ilgili başarı kriterini ölçtüğünü düşünüyorsanız "Uygun (U.)" sütununa; rubrikte yer alan başarı kriterinin düzenlenmesi gerektiğini düşünüyorsanız "Kısman Uygun (KU.) sütununa çarpı koyabilirsiniz. İlgili ölçütün uygun olmadığını düşünüyorsanız "Uygun Değil (UD.)" sütununa çarpı koyabilirsiniz. KU. ya da UD. sütunlarına işaret koymanız durumunda açıklama bölümüne önerinizi ya da düşüncelerinizi yazmanızı rica ederim.

Değerli katkılarınızdan dolayı teşekkürlerimi sunarım.

Öğr.Gör. Osman Gazi YILDIRIM

Kısaltmalar ve Renklendirme		
Uygun	U.	
Kısmen Uygun	KU.	
Uygun Değil	UD.	
Açıklama	-	

KATEGORİLER	BAŞARI KRITERLERI				UYGUNLUK DURUMU			
	Düşük Seviye 1 Puan	**Orta Seviye 5 Puan**	**İyi Seviye 8 Puan**	**İleri Seviye 10 Puan**	**U.**	**KU.**	**UD.**	**Açıklama**
Puan Sistemi	Düşük Seviye Oyunda puan sistemi bulunmuyor	Orta Seviye Oyunda puan sistemi bulunuyor ancak hatalardan dolayı çalışmıyor	İyi Seviye Oyunda çalışan bir puan sistemi bulunuyor ancak eksik sınıflar mevcut	İleri Seviye Oyunda çalışan bir puan sistemi bulunuyor ve tüm sınıflar mevcut				
Kalıtım Mekanizması	Düşük Seviye Oyunda kalıtım mekanizması kullanılmamış	Orta Seviye Oyunda kalıtım mekanizması kullanılmış ancak hatalar mevcut	İyi Seviye Oyunda çalışan bir kalıtım mekanizması bulunuyor ancak eksik sınıflar mevcut	İleri Seviye Oyunda çalışan bir kalıtım mekanizması bulunuyor ve tüm sınıflar mevcut				
Ses Sistemi	Düşük Seviye Oyunda ses sistemi bulunmuyor	Orta Seviye Oyuna ses sistemi eklenmiş ancak çalışmıyor	İyi Seviye Oyunda ses sistemi tek bir nesne için çalışıyor	İleri Seviye Oyunda ses sistemi tüm nesneler için çalışıyor				
Bildirim Sistemi	Düşük Seviye Oyunda bildirim sistemi bulunmuyor	Orta Seviye Oyuna bildirim sistemi eklenmiş ancak çalışmıyor	İyi Seviye Oyunda bildirim sistemi tek nesne için çalışıyor	İleri Seviye Oyunda bildirim sistemi tüm nesneler için çalışıyor				
Pist Tasarımı	Düşük Seviye Oyunda hiçbir pist tasarımı gerçekleştirilmemiş	Orta Seviye Oyuna sadece engeller ve levhalar eklenmiş	İyi Seviye Oyuna sadece engeller, levhalar ve yeni kontrol noktaları eklenmiş.	İleri Seviye Oyuna engeller, levhalar ve yeni kontrol noktaları eklenmiş. Oyundaki pist değiştirilmiş ve yeni bir pist tasarımı gerçekleştirilmiş				
Pist Seçim Ekranı	Düşük Seviye Oyunda pist seçim ekranı bulunmuyor	Orta Seviye Oyuna pist seçim ekranı eklenmiş ancak çalışmıyor	İyi Seviye Oyuna pist seçim ekranı eklenmiş ancak geçiş butonları ve resimleri bulunmuyor	İleri Seviye Oyuna pist seçim ekranı eklenmiş. Geçiş butonları ve resimleri mevcut				

Tablonun devamı…

KATEGORİLER	BAŞARI KRITERLERI				UYGUNLUK DURUMU			
	Düşük Seviye 1 Puan	**Orta Seviye 5 Puan**	**İyi Seviye 8 Puan**	**İleri Seviye 10 Puan**	**U.**	**KU.**	**UD.**	**Açıklama**
Hata Giderme (Debugging)	Düşük Seviye Oyunun oynanmasını etkileyen birçok hata mevcut: Oyun tamamlanmamış	Orta Seviye Oyunun oynanmasını etkileyen birkaç hata mevcut: Oyun neredeyse tamamlanmış	İyi Seviye Oyunda çok az ya da hiç hata bulunmuyor: Oyun neredeyse tamamlanmış	İleri Seviye Oyunda hiç hata bulunmuyor: Oyun tamamlanmış				
Tasarım ve Yaratıcılık	Düşük Seviye Oyuna hiçbir görsel unsur eklenmemiş	Orta Seviye Oyuna bir adet görsel unsur eklenmiş	İyi Seviye Oyuna birkaç görsel unsur eklenmiş	İleri Seviye Oyuna tüm görsel unsurlar eklenmiş				
Oyun Bileşenlerinin Organizasyonu	Düşük Seviye Hiyerarşi penceresinde oyun bileşenleri hiçbir şekilde gruplandırılmamış	Orta Seviye Hiyerarşi penceresinde sadece birkaç oyun bileşeni gruplandırılmış	İyi Seviye Hiyerarşi penceresinde oyun bileşenlerinin çoğu gruplandırılmış	İleri Seviye Hiyerarşi penceresinde tüm oyun bileşenleri gruplandırılmış ve hiyerarşi penceresi düzenli tutulmuş				
Proje Teslimi ve Sunumu	Düşük Seviye Proje teslimi geç yapılmış ve proje sunumu bulunmuyor	Orta Seviye Proje teslimi geç yapılmış ancak proje sunumu mevcut	İyi Seviye Proje teslimi zamanında yapılmış ancak proje sunumu bulunmuyor	İleri Seviye Proje teslimi zamanında yapılmış ve proje sunumu mevcut				

Ek 6. Proje Değerlendirme Rubriği

DÖNEM SONU PROJESİ DEĞERLENDİRME ÇİZELGESİ: KARTING MICROGAME				
KATEGORİLER	**Düşük Seviye 1 Puan**	**Orta Seviye 5 Puan**	**İyi Seviye 8 Puan**	**İleri Seviye 10 Puan**
Puan Sistemi	Düşük Seviye Oyunda puan sistemi bulunmuyor	Orta Seviye Oyunda puan sistemi bulunuyor ancak hatalardan dolayı çalışmıyor	İyi Seviye Oyunda çalışan bir puan sistemi bulunuyor ancak eksik sınıflar mevcut	İleri Seviye Oyunda çalışan bir puan sistemi bulunuyor ve tüm sınıflar mevcut
Kalıtım Mekanizması	Düşük Seviye Oyunda kalıtım mekanizması kullanılmamış	Orta Seviye Oyunda kalıtım mekanizması kullanılmış ancak hatalar mevcut	İyi Seviye Oyunda çalışan bir kalıtım mekanizması bulunuyor ancak eksik sınıflar mevcut	İleri Seviye Oyunda çalışan bir kalıtım mekanizması bulunuyor ve tüm sınıflar mevcut
Ses Sistemi	Düşük Seviye Oyunda ses sistemi bulunmuyor	Orta Seviye Oyuna ses sistemi eklenmiş ancak çalışmıyor	İyi Seviye Oyunda ses sistemi tek bir nesne için çalışıyor	İleri Seviye Oyunda ses sistemi tüm nesneler için çalışıyor
Bildirim Sistemi	Düşük Seviye Oyunda bildirim sistemi bulunmuyor	Orta Seviye Oyuna bildirim sistemi eklenmiş ancak çalışmıyor	İyi Seviye Oyunda bildirim sistemi tek nesne için çalışıyor	İleri Seviye Oyunda bildirim sistemi tüm nesneler için çalışıyor
Pist Tasarımı	Düşük Seviye Oyunda hiçbir pist tasarımı gerçekleştirilmemiş	Orta Seviye Oyuna sadece engeller ve levhalar eklenmiş	İyi Seviye Oyuna sadece engeller, levhalar ve yeni kontrol noktaları eklenmiş.	İleri Seviye Oyuna engeller, levhalar ve yeni kontrol noktaları eklenmiş. Oyundaki pist değiştirilmiş ve yeni bir pist tasarımı gerçekleştirilmiş
Pist Seçim Ekranı	Düşük Seviye Oyunda pist seçim ekranı bulunmuyor	Orta Seviye Oyuna pist seçim ekranı eklenmiş ancak çalışmıyor	İyi Seviye Oyuna pist seçim ekranı eklenmiş ancak geçiş butonları ve resimleri bulunmuyor	İleri Seviye Oyuna pist seçim ekranı eklenmiş. Geçiş butonları ve resimleri mevcut
Hata Giderme (Debugging)	Düşük Seviye Oyunun oynanmasını etkileyen birçok hata mevcut: Oyun tamamlanmamış	Orta Seviye Oyunun oynanmasını etkileyen birkaç hata mevcut: Oyun neredeyse tamamlanmış	İyi Seviye Oyunda çok az ya da hiç hata bulunmuyor: Oyun neredeyse tamamlanmış	İleri Seviye Oyunda hiç hata bulunmuyor: Oyun tamamlanmış
Tasarım ve Yaratıcılık	Düşük Seviye Oyuna hiçbir görsel unsur eklenmemiş	Orta Seviye Oyuna bir adet görsel unsur eklenmiş	İyi Seviye Oyuna birkaç görsel unsur eklenmiş	İleri Seviye Oyuna tüm görsel unsurlar eklenmiş

Oyun Bileşenlerinin Organizasyonu	Düşük Seviye Hiyerarşi penceresinde oyun bileşenleri hiçbir şekilde gruplandırılmamış	Orta Seviye Hiyerarşi penceresinde sadece birkaç oyun bileşeni gruplandırılmış	İyi Seviye Hiyerarşi penceresinde oyun bileşenlerinin çoğu gruplandırılmış	İleri Seviye Hiyerarşi penceresinde tüm oyun bileşenleri gruplandırılmış ve hiyerarşi penceresi düzenli tutulmuş
Proje Teslimi ve Sunumu	Düşük Seviye Proje teslimi geç yapılmış ve proje sunumu bulunmuyor	Orta Seviye Proje teslimi geç yapılmış ancak proje sunumu mevcut	İyi Seviye Proje teslimi zamanında yapılmış ancak proje sunumu bulunmuyor	İleri Seviye Proje teslimi zamanında yapılmış ve proje sunumu mevcut

Ek 7. Programlama Kaygı Ölçeği

Sevgili öğrenciler,
Programlama Kaygı Ölçeği bilimsel bir araştırma için geliştirilmiştir. Sorulara vereceğiniz yanıtlar araştırmada kullanılacak olup başka bir kimse ile paylaşılmayacaktır. Her maddeyi dikkatle okuyup belirtilen görüşün sizin görüşünüze ne derece uyduğunu ya da uymadığını belirleyiniz. Soruların doğru ya da yanlış cevapları yoktur. Lütfen tüm soruları yanıtlayınız ve her madde için bir seçeneği işaretleyiniz. Katkılarınız için teşekkür ederim.

Osman Gazi YILDIRIM
Öğretim Görevlisi

ÖLÇEK MADDELERİ	**1: Hiçbir Zaman** **2: Çok Nadir** **3: Bazen** **4: Sıklıkla** **5: Her Zaman**				
	1	**2**	**3**	**4**	**5**
1. Program yazarken sınıf arkadaşlarım gibi sakin kalamamaktan endişe duyarım.					
2. Daha önce programlama dersi alan arkadaşlarımın seviyesine yetişemeyeceğime inanmak beni kaygılandırır.					
3. Daha önce programlama dersi alan sınıf arkadaşlarımın varlığı beni tedirgin eder.					
4. Benim yazamadığım bir kodu çoğu sınıf arkadaşım yazabilmesi beni kaygılandırır.					
5. Arkadaşlarım benim anlamadığım programlama konularında konuştuğunda gergin hissederim.					
6. Program yazamayıp sınıf arkadaşlarımın önünde gülünç duruma düşeceğimden endişelenirim.					
7. Programlamayı iyi anlayamadığımı düşünürüm.					
8. Mantığını öğrenmek yerine programlama konularını ezberlediğimi hissetmek beni kaygılandırır.					
9. Programlama derslerinde öğrendiklerimi çabuk unuttuğumu hissetmek beni kaygılandırır.					
10. Program yazarken çözüm için gerekli olan basamakları (algoritmayı) doğru oluşturabileceğim konusunda şüphelerim var.					
11. Program satırları karmaşık olmaya başladığında aklımın karıştığını hissederim.					
12. Program yazma konusunda kendime güvenmem.					
13. Programlama dersinde öğrenilecek çok fazla konunun olması beni korkutur.					
14. Programlarımda hatalarla karşılaşmaktan dolayı endişe duyarım.					

Ek 8. Analiz Aşaması Katılımcı Anketi

Sevgili öğrenciler,
Nesne Yönelimli Programlama dersini daha etkili ve verimli hale getirebilmek maksadıyla bilimsel bir çalışma yapmaktayım. Sizden aşağıdaki soruları ayrıntılı bir şekilde yanıtlamanız beklenmektedir. Sorulara vereceğiniz yanıtlar araştırmada kullanılacak olup başka bir kimse ile paylaşılmayacaktır. Katkılarınız için teşekkür ederim.

Osman Gazi YILDIRIM
Öğretim Görevlisi

1. Kişisel Bilgiler:

1.1. Adınız Soyadınız:

1.2. Yaş Aralığınız:

( ) 17-24

( ) 25-29

( ) 30-39

( ) 40 ve üzeri

1.3. Mezun Olduğunuz Lise Türü:

2. Bilgisayar Kullanımı ve Bilgisayar Oyunları:

2.1. Kişisel bilgisayarınız var mı?

( ) Evet

( ) Hayır

2.2. Bilgisayarı hangi sıklıkla kullanırsınız?

( ) Nadir

( ) Orta sıklıkta

( ) Sık

( ) Çok sık4

2.3. Dijital oyunları hangi sıklıkla oynarsınız?

( ) Hiç

( ) Günde 1 saatten az

( ) Günde 1-2 saat

( ) Günde 2 saatten fazla

3. Programlama Bilgisi:

3.1. Bölümünüze kayıt olmadan önce programlama dersi aldınız mı?

( ) Evet

( ) Hayır

3.2. Şu anki programlama bilgi ve becerinizi hangi seviyede tanımlarsınız?

( ) Çok düşük

( ) Düşük

( ) Orta

( ) Yüksek

( ) Çok yüksek

3.3. Programlama öğrenmenin zorluk seviyesini nasıl tanımlarsınız?

( ) Çok kolay

( ) Kolay

( ) Orta zorlukta

( ) Zor

( ) Çok zor

4. Programlama Hakkındaki Düşünceler:

4.1. Programlama öğrenmenin yararlı olduğunu düşünüyor musunuz? Neden?

4.2. Programlama öğrenmeyi keyifli buluyor musunuz? Neden?

4.3. Programlama derslerinde öğrendiklerinizin akılda kalıcı olduğunu düşünüyor musunuz? Neden?

5. Derste Geliştirilen Uygulamalar Hakkındaki Düşünceler:

5.1. Programlama derslerinde geliştirdiğiniz programlama uygulamalarının, programlama bilginize katkısını nasıl tanımlarsınız?

5.2. Programlama dersinde ne tür uygulamalar geliştirmenin öğrencilerin programlama bilgi ve becerilerine en çok katkı sağlayacağına inanıyorsunuz? Neden? [Birden fazla seçebilirsiniz.]

( ) Sistem yazılımları

( ) Güvenlik yazılımları

( ) Bilgisayar oyunları

( ) Otomasyon yazılımları

( ) Form uygulamaları

( ) Diğer:

5.3. NYP dersinde görev tabanlı öğrenme kapsamında sadece bir yazılım türünde yazılım geliştirecek olsanız hangisini tercih edersiniz? Neden?

( ) Sistem yazılımı

( ) Güvenlik yazılımı

( ) Oyun/Mobil Oyun

( ) Form Uygulamaları/Otomasyon

( ) İnternet Sitesi

( ) Veritabanı Yazılımları

( ) Mobil Yazılım

( ) Diğer

6. Derste Yaşanan Sorunlar:

6.1. Nesne Yönelimli Programlama dersinde ne tür sorunlar yaşıyorsunuz? Sebepleri nelerdir?

6.2. Size göre Nesne Yönelimli Programlama dersinde yaşadığınız sorunlar nasıl çözülebilir? Neden?

6.3. Nesne Yönelimli Programlama dersinin işlenişi ya da derste kullanılan örneklerle ilgili paylaşmak istediğiniz düşünceniz var mı?

Ek 9. Analiz Aşaması Öğretim Elemanı Görüşme Formu

Merhaba hocam. Bu görüşmeyi, Nesne Yönelimli Programlama dersinde öğretim elemanı olarak yaşadığınız sorunlar ve bu sorunların çözümüne yönelik düşüncelerinizi öğrenmek amacıyla yapıyorum. Görüş ve önerilerinizin Nesne Yönelimli Programlama dersinin daha etkili ve verimli hale getirilmesine katkıda bulunacağına inanıyorum.

İzin verirseniz görüşmeyi kaydetmek istiyorum. Görüşmemize geçmeden önce, görüşmemizin gizli olduğunu belirtmek isteriz. Görüşmede konuşulanları biz ve bazı araştırmacılar dışında başka kimsenin bilmeyeceği konusunda emin olabilirsiniz. Bunun yanında araştırma raporumuzda isimleriniz kullanılmayacaktır. Görüşmemiz sırasında ses kayıt cihazını istediğiniz zaman durdurabilirsiniz. Görüşmemizin yaklaşık 15 dakika süreceğini tahmin ediyorum. Görüşmeye başlamadan önce bana sormak istediğiniz soru ya da belirtmek istediğiniz bir düşünceniz var mı?

Görüşme sorularına başlamamızı ister misiniz?

Osman Gazi YILDIRIM
Öğretim Görevlisi

1. Size göre Nesne Yönelimli Programlama dersinde öğrenciler ne tür sorunlar yaşamaktadır?
2. Öğrencilerin bu dersteki başarısını en çok hangi unsurlar etkilemektedir?
3. Öğretim elemanı olarak Nesne Yönelimli Programlama dersinde ne tür sorunlar yaşadınız?
4. Nesne Yönelimli Programlama dersinde yaşadığınız sorunlar için ne tür çözümler ürettiniz?
5. Size göre programlama derslerinde hangi sebeplerden dolayı öğrencilerin motivasyonları ve derse karşı olan ilgileri azalmaktadır?
6. Nesne Yönelimli Programlama dersini daha kalıcı, eğlenceli ve ilgi çekici hale getirebilmek için derste ne tür değişikliklere başvurulabilir?

Ek 10. Unity 3D Başarı Sınavı Kapsam Geçerliliği Uzman Görüş Formu

Sayın uzman,

Bu kapsam geçerliliği uzman görüş formu, doktora tezim kapsamında veri toplama aracı olarak kullanılacak Unity 3D Başarı Sınavının kapsam geçerliliğini kontrol edebilmek amacıyla hazırlanmıştır. Unity 3D Başarı Sınavı toplam 21 sorudan oluşmakta ve 21 farklı kazanımı kontrol etmektedir.

İkinci sayfadan itibaren göreceğiniz tablonun ilk sütununda Unity 3D Başarı Sınavının soruları yer almaktadır. İkinci sütunda ilgili sorunun ölçmeye amaçladığı kazanımlar yer almaktadır. Başarı sınavında yer alan bir soru yalnızca bir kazanımı test edebileceği gibi birden fazla kazanımı da test edebilmektedir. Tablonun üçüncü sütunundan itibaren ilgili kazanımın uygunluğunu değerlendireceğiniz bölüm yer almaktadır.

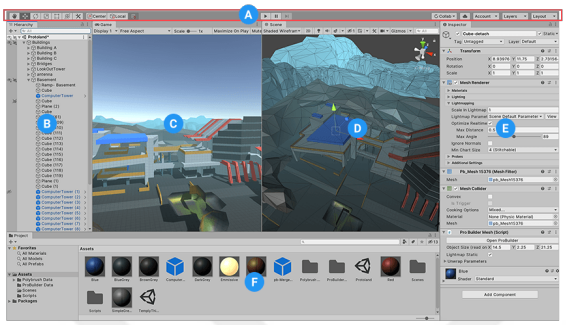
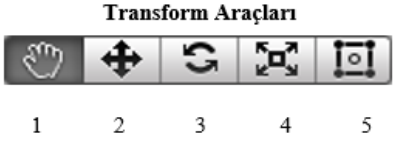
Unity 3D Başarı Sınavında yer alan sorunun ilgili kazanımı ölçtüğünü düşünüyorsanız "Uygun (U)" sütununa çarpı işareti koyabilirsiniz. Başarı sınavında yer alan sorunun düzenlenmesi gerektiğini düşünüyorsanız "Küçük Düzeltilmeler Yapılmalı (KDY) ya da Büyük Düzeltmeler Yapılmalı (BDY)" sütununa çarpı koyabilirsiniz. Başarı sınavında yer alan sorunun ilgili kazanımı test etmediğini ya da sorunun gereksiz olduğunu düşünüyorsanız "Uygun Değil (UD)" sütununa çarpı koyabilirsiniz. BDY, KDY ya da UD sütunlarına işaret koymanız durumunda Açıklama bölümüne önerinizi ya da düşüncelerinizi yazmanızı rica ederim.

Değerli katkılarınızdan dolayı teşekkürlerimi sunarım.

Öğr.Gör. Osman Gazi YILDIRIM

Kısaltmalar ve Renklendirme

Uygun	U.
Küçük Düzeltmeler Yapılmalı	KDY.
Büyük Düzeltmeler Yapılmalı	BDY.
Uygun Değil	UD.
Açıklama	-

Unity 3D Başarı Sınavı Sorusu	Kazanım	UYGUNLUK DURUMU				
		U.	KDY.	BDY	UD.	Açıklama
Aşağıda Unity 3D editöründe yer alan bazı bölümler verilmiştir. Bu öğelerin isimlerini yazarak birkaç cümle ile açıklayınız. 						
A:	Unity 3D editörünün The Toolbar penceresini tanır.					
B:	Unity 3D editörünün The Hierarchy penceresini tanır.					
C:	Unity 3D editörünün The Game View penceresini tanır.					
D:	Unity 3D editörünün The Scene View penceresini tanır.					
E:	Unity 3D editörünün The Inspector penceresini tanır.					
F:	Unity 3D editörünün The Project penceresini tanır.					
Aşağıda Unity 3D editöründe yer alan Transform Araçları ve bu araçlara ait gizmolar verilmiştir. Bu gizmolarla transfer araçlarının numaralarını eşleştirerek transform araçlarının işlevlerini açıklayınız. 						
	Move aracı ile Move Gizmo'sunu eşleştirir.					

Transform Gizmosu	Araç Numarası	Transform Aracı İşlevi Açıklaması	
			Rotate aracı ile Rotate Gizmo'sunu eşleştirir.
			Scale aracı ile Scale Gizmo'sunu eşleştirir.
			RectTransform aracı ile RectTransform Gizmo'sunu eşleştirir.
			

Aşağıda Unity 3D ile ilgili bazı kavramlar verilmiştir. Bu kavramları birkaç cümle ile açıklayınız.	
2A) Asset:	Asset kavramını açıklar.
2B) Material:	Material kavramını açıklar.
2C) Shader:	Shader kavramını açıklar.
2D) Prefab:	Prefab kavramını açıklar.
2E) Rigidbody:	Rigidbody kavramını açıklar.
2F) Particle:	Particle kavramını açıklar.
3A) Vektörlerin bir nesneyi hareket ettirmek için kullanılması durumunda normalleştirilmesi (normalization) gerekmektedir. Sebebini açıklayınız.	Vektörlerin normalleştirilmesinin gerekliliğini açıklar.
3B) Zamana bağlı olarak çalışan oyun mekanizmalarının doğru çalışabilmesi için Time.deltaTime kullanılması gerekmektedir. Sebebini açıklayınız.	Time.deltaTime ifadesinin kullanılma nedenlerini açıklar.
3C) Kullanım amaçları bakımından FixedUpdate() ile Update() metotları arasındaki farkı açıklayınız.	Monobehavior metotlarından FixedUpdate() ile Update() metotlarını karşılaştırır.

3D) Yan tarafta Unity 3D'de yer alan Monobehaviour metotlarının çalışma sırası verilmiştir ancak sıralamada bazı hatalar bulunmaktadır. Hataların kaynağını açıklayarak doğru sıralamayı oluşturunuz.

Unity 3D'de yer alan Monobehaviour metotlarının çalışma sırasını düzenler.

3E) Aşağıda yer alan kod bloğunda bazı hatalar bulunmaktadır. Hataların kaynağını açıklayarak Start() metodunun doğru çalışması için gereken kodları yazınız.

```
using UnityEngine;
using System.Collections;

public class TEST : MonoBehaviour {
    void Start () {
        transform.position.x = 10;
    }
}
```

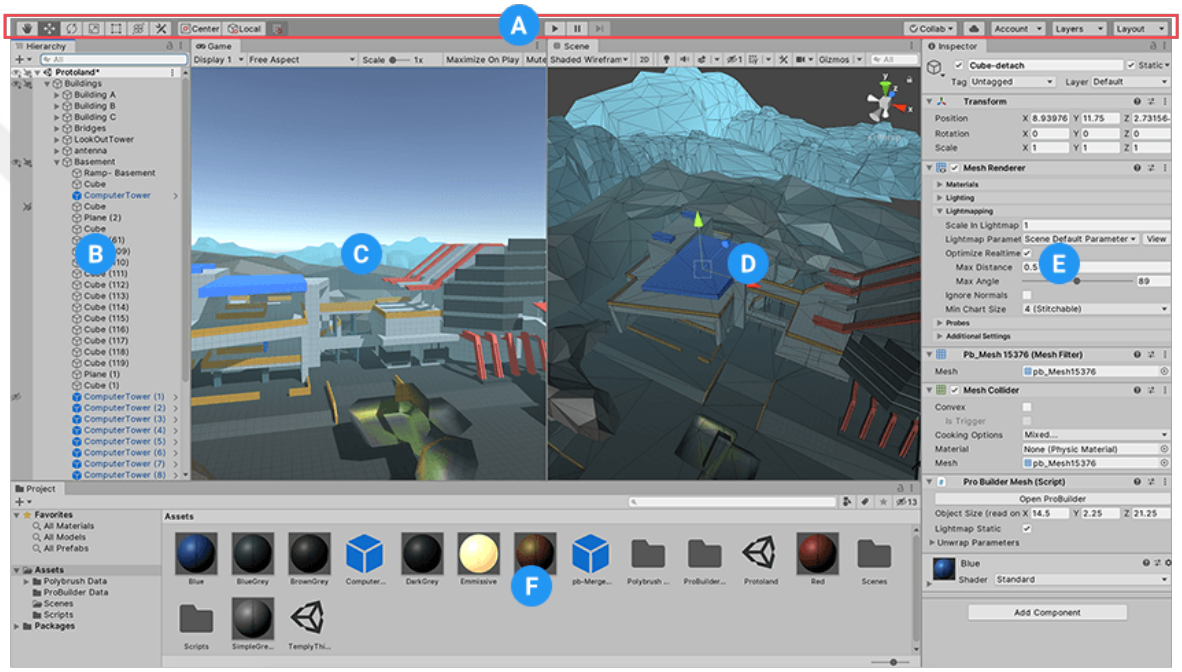
Kendisine sunulan bir Monobehavior metodunun içeriğindeki C# kodlarındaki hataları düzeltir.

Ek 11. Unity 3D Başarı Sınavı

Talimatlar: Sınav süreniz **40 dakikadır.** Sınav kâğıdınıza kimlik bilgilerinizi doğru olarak yazınız. Sınav kitapçığınız toplam iki bölümden ve altı sayfadan oluşmaktadır. Her bölümün karşısında puan değeri yer almaktadır. Sınav kitapçığını ve soruları kontrol ediniz.

Bölüm 1: Unity 3D Editörü (30 Puan)

1. Aşağıda Unity 3D editöründe yer alan bazı bölümler verilmiştir. Bu öğelerin isimlerini yazarak birkaç cümle ile açıklayınız. **Her öğe 3 puandır.**



A:

B:

C:

D:

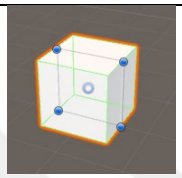
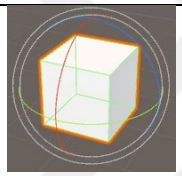
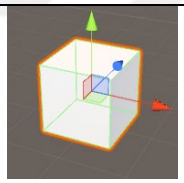
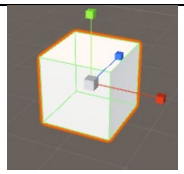
E:

F:

2. Aşağıda Unity 3D editöründe yer alan Transform Araçları ve bu araçlara ait gizmolar verilmiştir. Bu gizmolarla transfer araçlarının numaralarını eşleştirerek transform araçlarının işlevlerini açıklayınız. **Her öğe 3 puandır.**

Transform Araçları

1 2 3 4 5

Transform Gizmosu	Araç Numarası	Transform Aracı İşlevi Açıklaması
		
		
		
		

Bölüm 2: Tanımlamalar (30 Puan)

1. Aşağıda Unity 3D ile ilgili bazı kavramlar verilmiştir. Bu kavramları birkaç cümle ile açıklayınız. Her kavram **5 puandır.**

a. Asset:

b. Material:

c. Shader:

d. Prefab:

e. Rigidbody:

f. Particles:

Bölüm 3: Açık Uçlu Sorular (40 Puan)

Talimatlar: Aşağıda yer alan soruları kısaca yanıtlayınız. Her soru 8 puandır.

a. Vektörlerin bir nesneyi hareket ettirmek için kullanılması durumunda normalleştirilmesi (normalization) gerekmektedir. Sebebini açıklayınız.

b. Zamana bağlı olarak çalışan oyun mekanizmalarının doğru çalışabilmesi için Time.deltaTime kullanılması gerekmektedir. Sebebini açıklayınız.

c. Kullanım amaçları bakımından FixedUpdate() ile Update() metotları arasındaki farkı açıklayınız.

d. Yan tarafta Unity 3D'de yer alan Monobehaviour metotlarının çalışma sırası verilmiştir ancak sıralamada bazı hatalar bulunmaktadır. Hataların kaynağını açıklayarak doğru sıralamayı oluşturunuz.

e. Aşağıda yer alan kod bloğunda bazı hatalar bulunmaktadır. Hataların kaynağını açıklayarak Start() metodunun doğru çalışması için gereken kodları yazınız.

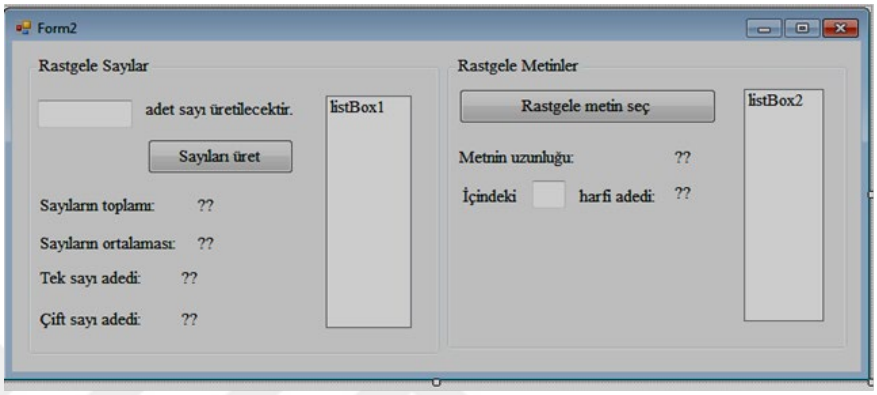
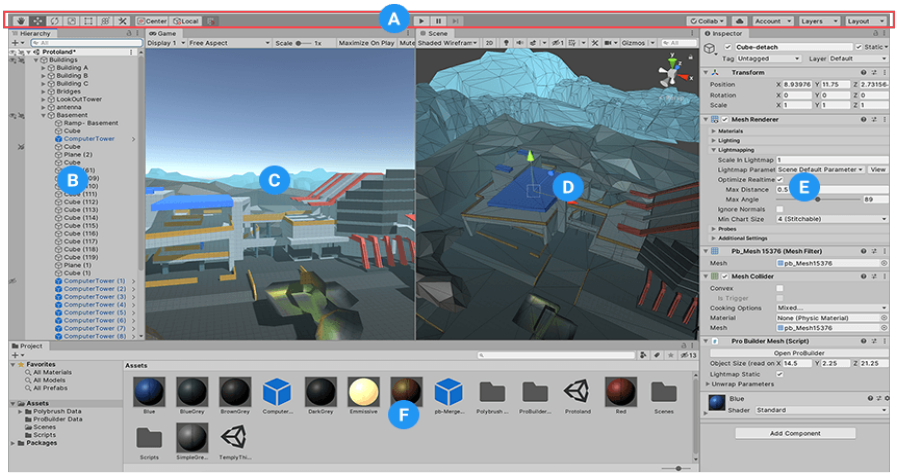
```
using UnityEngine;
using System.Collections;

public class TEST : MonoBehaviour {
    void Start () {
        transform.position.x = 10;
    }
}
```

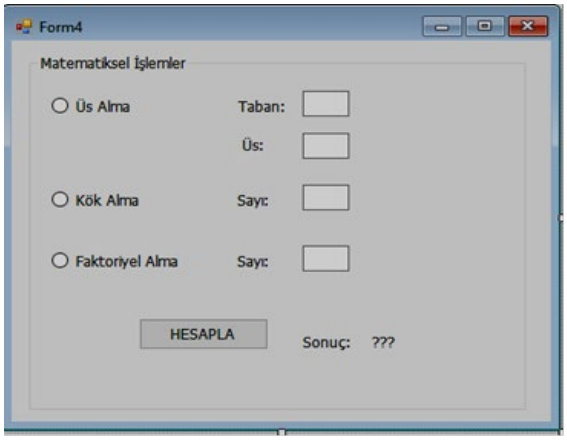
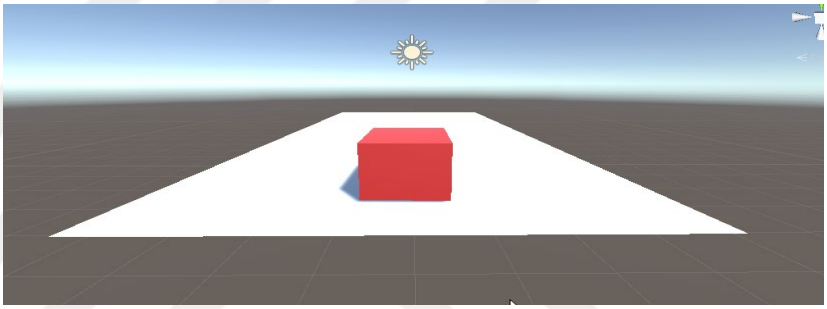
Ek 12. Haftalık Ders Planları

1'İNCİ HAFTA DERS PLANI

Konu:	Genel C# Revizyonu & Unity 3D Editörü Tanıtımı
Tarih ve Saat:	27 Ekim 2020 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) Visual Studio 2017 IDE'sini kullanarak bir C# projesi oluşturur. 2) C#'ta değişken türlerini listeler. 3) C#'ta değişken tanımlar. 4) C#'ta matematiksel ve mantıksal ifadeler listeler. 5) C#'ta karar yapılarını kullanır. 6) C#'ta for ve foreach döngüsünü kullanır. 7) C#'ta metot tanımlar. 8) C#'ta tanımladığı metodu kullanılır. 9) Unity 3D'ye kayıt olur. 10) Unity 3D'de yeni bir proje oluşturur. 11) Projeye yeni bir sahne ekler. 12) Sahneleri kaydeder. 13) Unity 3D'de oluşturulabilecek iki boyutlu ve üç boyutlu projelerin arasındaki farkı bilir. 14) Unity 3D editörüne ait Scene View, Game View, Inspector Window, Hierarchy Window, Toolbar, Project Window gibi bölümlerini tanır. 15) Unity 3D layout'unu değiştirir. 16) Asset, Material, Component, Transform, Prefab, Rigidbody, Collider, Tag, Game Object, UI gibi Unity 3D kavramlarının ne işe yaradığını bilir. 17) Unity 3D'de kullanılabilecek üç boyutlu cisimlerin (küp, küre, düzlem, arazi vb.) ne işe yaradığını bilir. 18) Sahneye üç boyutlu bir cisim ekler. 19) Eklediği üç boyutlu nesnenin konumunu değiştirir. 20) Eklediği üç boyutlu nesnenin boyutlarını değiştirir. 21) Eklediği üç boyutlu nesneyi X,Y ve Z ekseninde döndürür. 22) Yeni materyal oluşturur. 23) Oluşturduğu materyali eklediği üç boyutlu nesneye uygular. 24) Sahneye eklediği üç boyutlu nesnenin etrafında kamerayı döndürerek gezinti yapar. 25) WASD tuşlarını ve fareyi kullanarak sahnede gezinti yapar.
Açıklamalar:	Öğretim programının ilk haftasında genel C# programlama dili revizyonu gerçekleştirilir. Bunu gerçekleştirmek için Visual Studio 2017 IDE'si kullanılarak bir form uygulaması oluşturulur. Bu form uygulaması vasıtasıyla C#'ta değişkenler, matematiksel ve mantıksal ifadeler, karar yapıları, döngüler ve metotlar konularıyla ilgili genel tekrar yapılır. Devamında Unity 3D editörü tanıtımıyla ilgili bilgiler edinilir. Canlı ders esnasında çok fazla zaman alacağı için dersten önce Unity 3D editörü, Unity Hub ve Visual Studio Community 2017 yazılımlarının kurulumu ile ilgili öğrencilerle Google Classroom'da videolar paylaşılır. Öğrencilerden bu yazılımları dersten önce kurmaları istenir. Bunun yanında Unity 3D ve Visual Studio Community 2017 yazılımlarına nasıl lisans temin edileceği ders öncesinde öğrencilere aktarılır. Unity 3D

	editörü arayüzü, pencereleri ve Unity 3D'de yer alana temel kavramlar anlatılır. Ders anlatımı esnasında sahneye bir üç boyutlu nesne eklenir ve taşıma, boyutlandırma, döndürme, pencerelerin değişimi, sahnede ve nesne etrafında gezinme gibi unsurlar bu üç boyutlu nesne kullanılarak anlatılır.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	C# revizyon konuları canlı kod gösterimleriyle anlatılır. Unity 3D editörü tanıtımı konuları demonstrasyon yöntemi ile anlatılır.
Ders İçi Uygulamalar: (Senkron)	**GENEL C# REVİZYONU**  Rastgele Sayılar örneğinde klavyeden girilen adet kadar rastgele sayı üretilip bir diziye aktarılır; daha sonra bu tamsayı dizisinin elemanları liste kutusuna ekletilir. Bu dizinin elemanlarının toplamı; ortalaması; çift ve tek eleman adedi hesaplatılır. Rastgele Metinler örneğinde ise bir metin dizisinden rastgele bir eleman seçtirilir ve bu eleman liste kutusuna aktarılır. Tanımlanacak metotlar vasıtasıyla rastgele seçilen metnin uzunluğu ve klavyeden girilen karakterin adedi hesaplatılır. **UNITY 3D EDİTÖRÜ TANITIMI**  Unity 3D editörü tanıtımı kapsamında Unity'de geliştirilebilecek proje türlerinden bahsedilir. Üç boyutlu bir oyun projesi oluşturularak kaydetme seçeneklerinden bahsedilir. Editörün arayüzü ve yukarıdaki görselde belirtilen project view, hierarchy, toolbar, game view gibi bölümleri tanıtılır. Devamında move, scale, hand tool gibi gizmolar tanıtılır. Son olarak sahneye eklenebilecek üç boyutlu nesneler gösterilir ve sahneye eklenerek renk, konum, boyut gibi özellikleri değiştirilir.

Araştırma ve Uygulamalar: (Asenkron)	**GENEL C# REVİZYONU**  Asenkron uygulamalar kapsamında öğrencilerden yukarıdaki örneği tamamlamaları beklenir. Matematiksel İşlemler örneğinde öğrencilerin tanımlayacağı metotlar kullanılarak üs, kök ve faktöriyel hesaplamaları yapılır. **UNITY 3D EDİTÖRÜ TANITIMI**  Unity 3D'de üç boyutlu bir oyun projesi oluşturularak sahneye bir adet küp ile plane eklenir. Bu nesnelerin konu, renk ve boyut gibi özellikleri değiştirilir. Nesnelere rigidbody, collider gibi özellikler eklenerek nesnelerin aralarındaki ilişkiler incelenir.

2’NCİ HAFTA DERS PLANI

Konu:	Unity 3D’de Nesnelere Kod Ekleme
Tarih ve Saat:	3 Kasım 2020 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) Unity 3D’de yeni bir proje oluşturur. 2) Projeye düzlem (plane), küp gibi nesneler ekler. 3) Nesnelere rigidbody özelliği ekler. 4) Nesnelere bileşen olarak C# script’i ekler. 5) C# sınıflarında (C# script) kullanılabilecek Monobehaviour metotlarını listeler. 6) Awake(), OnEnable(), Start(), Update(), FixedUpdate(), LateUpdate(), OnDisable(), OnDestroy(), OnTriggerEnter() ve OnCollisionEnter() monobehaviour metotlarının kullanım amaçlarını açıklar. 7) Projede yer alan oyun nesnelerinde Monobehaviour metotlarını kullanarak oyun mekanizmalarının işletilmesini sağlar.
Açıklamalar:	Bu haftaki derste oyun projesinde yer alan nesnelere C# kodu (C# script) ekleme anlatılır. Unity 3D’de üç boyutlu bir oyun projesi oluşturularak projeye küp ve plane nesneleri eklenir. Küp nesnesine MovementController isminde bir C# sınıfı eklenerek bu nesnenin her bir frame’de z ekseninde hareket etmesi sağlanır. Ayrıca yön tuşlarına basılınca küp nesnesinin sağa ve sola hareket etmesi sağlanır. CameraFollow isminde bir sınıf oluşturularak kameranın küp nesnesini takip edebilmesi sağlanır. Son olarak küp nesnesinin başka bir nesneyle çarpışıp çarpışmadığı OnCollisionEnter() metodu kullanılarak kontrol ettirilir.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Nesnelere C# kodu ekleme konuları demonstrasyon yöntemi ile anlatılır.
Ders İçi Uygulamalar: (Senkron)	**NESNELERE C# KODU EKLEME**  Oluşturalan küp nesnesi plane’nin üzerine konuşlandırılır. Küp nesnesinin hareket edebilmesi için MovementController isminde bir sınıf tanımlanarak bu sınıf, küp nesnesine bileşen olarak eklenir. MovementController sınıfına ait örnek kodlar aşağıdaki gibidir:

```
Unity Betiği | 0 başvuru
public class MovementController : MonoBehaviour
{
    public Rigidbody rb;

    public float ileriKuvvet = 2000f;
    public float yanKuvvet = 500f;

    Unity İletisi | 0 başvuru
    private void FixedUpdate()
    {
        rb.AddForce(0, 0, ileriKuvvet);

        if (Input.GetKey("d"))
        {
            rb.AddForce(yanKuvvet, 0, 0);
        }

        if (Input.GetKey("a"))
        {
            rb.AddForce(-yanKuvvet, 0, 0);
        }
    }
}
```

Küp nesnesinin hareketinin sağlanmasının ardından kameranın bu nesneyi takip edebilmesi için CameraFollow() isminde bir sınıf tanımlanır ve küp nesnesine bileşen olarak eklenir. CameraFollow sınıfına ait örnek kodlar aşağıdaki gibidir:

```
Unity Betiği | 0 başvuru
public class CameraFollow : MonoBehaviour
{

    public Transform player;
    public Vector3 offset = new Vector3(0,1,-5);

    // Update is called once per frame
    Unity İletisi | 0 başvuru
    void Update()
    {
        transform.position = player.position + offset;
    }
}
```

Son olarak küp nesnesinin başka bir nesneyle çarpışıp çarpışmadığı OnCollisionEnter() metodu ile kontrol ettirilir. Bu metoda ait örnek kodlar aşağıdaki gibidir:

```
Unity Betiği | 0 başvuru
public class PlayerCollision : MonoBehaviour
{
    Unity İletisi | 0 başvuru
    private void OnCollisionEnter(Collision collision)
    {
        Debug.Log(collision.collider.tag);
    }
}
```

Araştırma ve Uygulamalar: (Asenkron)	Canlı derste gösterilen konular ve metotlar bireysel olarak uygulanır.

3’NCÜ HAFTA DERS PLANI

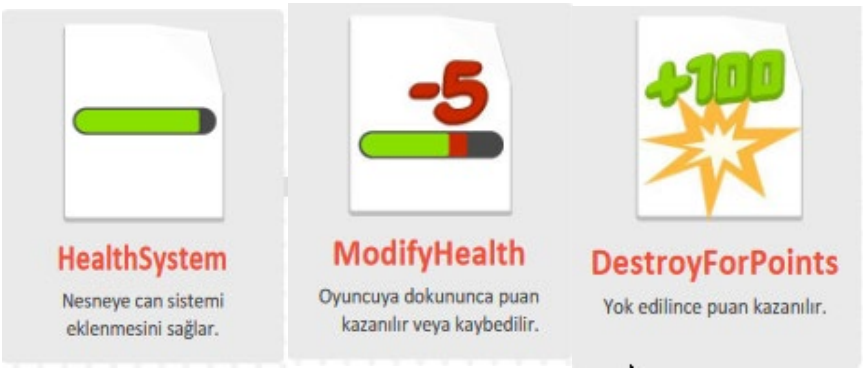
Konu:	Unity 3D Playground Projesi Oluşturma-1
Tarih ve Saat:	10 Kasım 2020 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) Playground platformunu tanır. 2) Playground script’lerinin kullanım amaçlarını açıklar. 3) Yeni bir iki boyutlu Unity 3D projesi oluşturur. 4) Oluşturduğu projeye Playground platformunu ekler. 5) Projeye arka plan resmi ekler. 6) Unity 3D’de Draw Mode özelliğinin ne işe yaradığını bilir. 7) Arka plan resminin Draw Mode özelliğini değiştirir. 8) Unity 3D’de Sorting Layer özelliğinin ne işe yaradığını bilir. 9) Arka plan resminin Sorting Layer’ını değiştirir. 10) Hiyerarşiye Spaceship ekler. 11) Spaceship’e Push ve Rotate script’leri ekler. 12) Push ve Rotate script’lerin çeşitli özelliklerini değiştirir. 13) Parenting ve Childing’in ne işe yaradığını açıklar. 14) Particle’ın kullanım amaçlarını ifade eder. 15) Spaceship’e particle ekler. 16) Eklediği particle’ı child object yapar. 17) Kameraya CameraFollow script’i ekler. 18) Takip edilecek nesnenin atamasını yapar. 19) Sahneye astroid ekler. 20) Collider’ın hangi amaçlarla kullanıldığını ifade eder. 21) Astroid’e Polygon Collider ekler. 22) Spaceship ile astroid’in çarpışmasını sağlar.
Açıklamalar:	Bu haftaki derste Unity 3D editörü bileşenlerinin detaylı açıklamalarının yapılabilmesi maksadıyla Unity Playground platformunun tanıtımı yapılır. İlave olarak bu platformda hazırlanacak uzay oyununun ilk kısmı proje olarak hazırlanır.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Playground oyun projesinin ilk kısmı demonstrasyon yöntemi ile anlatılır.
Ders İçi Uygulamalar: (Senkron)	**UNITY PLAYGROUND PROJESİ BİRİNCİ KISIM**  Projenin bu haftaki konuları olarak projenin arka planını değiştirme, nesneleri Push ve Rotate scriptler kullanarak hareket ettirme, parent-child object

	ilişkileri, kameranın nesneleri takip etme script'i gibi konular işlenir. Playground scriptlerinden bahsedilir. Bu hafta kullanılan scriptler aşağıdaki gibidir: 
Araştırma ve Uygulamalar: (Asenkron)	Bireysel olarak iki boyutlu bir proje oluşturulur. Bu projeye Unity Playground asset'i eklenir. Canlı derste gösterilen konular, ders videolarından da faydalanılarak uygulanır.

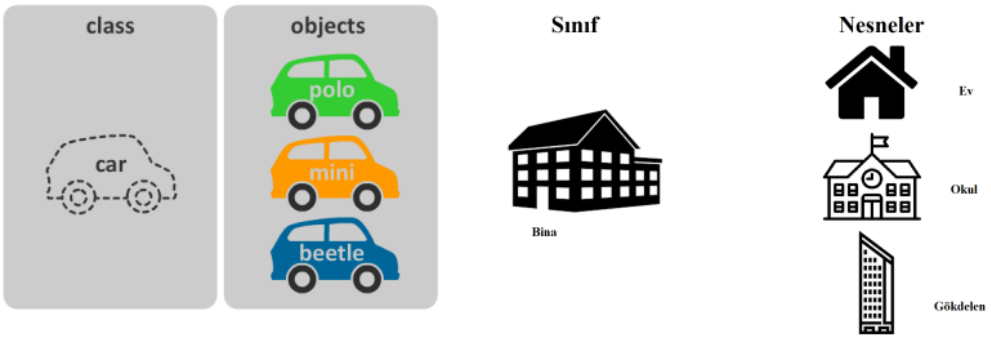
4'ÜNCÜ HAFTA DERS PLANI

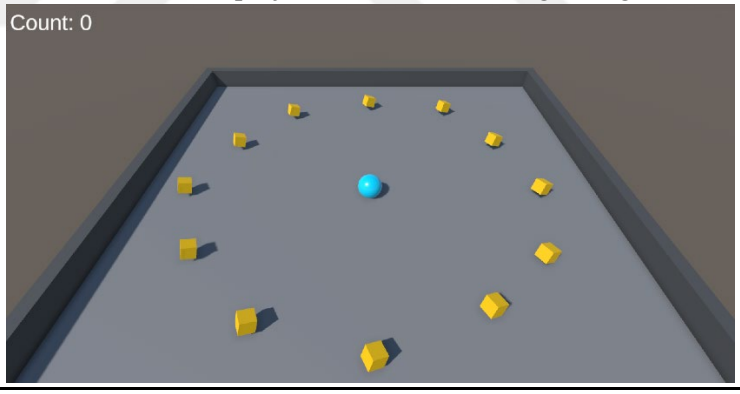
Konu:	Unity 3D Playground Projesi Oluşturma-2
Tarih ve Saat:	17 Kasım 2020 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) Tag, Projectile, Trigger, Attribute kavramlarını açıklar. 2) Oyuncuya tag ve attribute ekler. 3) Prefab oluşturur ve kullanır. 4) Can Sistemi (Health System), Lazer Sistemi (Laser System) ve Puan Sistemi (Point System) ekler. 5) Kullanıcı Arayüz (UI) nesnelerinin kullanım amaçlarını açıklar. 6) Level tasarımı yapar.
Açıklamalar:	Bu haftaki derste Unity Playground platformunda geçen hafta birinci kısmı tamamlanan oyun projesinin ikinci kısmı tamamlanır.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Playground oyun projesinin ikinci kısmı demonstrasyon yöntemi ile anlatılır.
Ders İçi Uygulamalar: (Senkron)	**UNITY PLAYGROUND PROJESİ İKİNCİ KISIM**   Bu hafta, geçen hafta başlanan Unity Playground projesinin ikinci kısmı tamamlanır. Projeye can, lazer ve puan sistemi eklenir. Bunun yanında UI nesneleri ve çeşitli attribute'lar eklenir. Son olarak level tasarımı konularında çalışmalar yapılır. Bu hafta kullanılan attribute'lar aşağıdaki gibidir: 
Araştırma ve Uygulamalar: (Asenkron)	Canlı derste gösterilen konular, ders videolarından da faydalanılarak uygulanır.

5'İNCİ HAFTA DERS PLANI

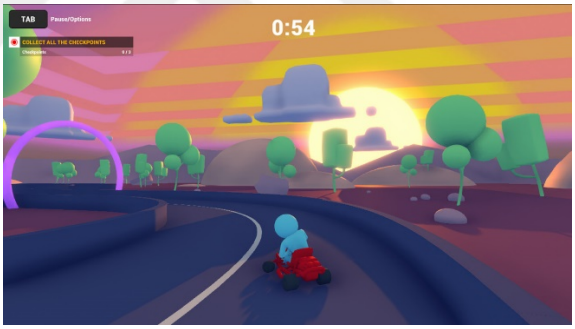
Konu:	Nesne Yönelimli Programlamanın Temelleri
Tarih ve Saat:	24 Kasım 2020 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) Nesne yönelimli programlama paradigmasının kullanılmasının amaçlarını açıklar. 2) Sınıf, nesne, özellik (attribute) ve davranış (behaviour) kavramlarını açıklar. 3) Kurucu metotların kullanılmasının nedenlerini listeler. 4) C#'ta kullanılan erişim belirleyicileri listeler. 5) Nesneler arası mesajlaşma kavramını açıklar. 6) UML sınıf diyagramlarını oluşturur. 7) Bir oyun projesinde kullanılan sınıf ve nesneler arasındaki ilişkiyi çözümler.
Açıklamalar:	Bu haftaki derste Nesne Yönelimli Programlamanın temelleri üzerinde durulur. Dersin uygulama saatinde Roll-A-Ball oyun projesi incelenir. Bu projede yer alan NYP mekanizmaları çözümlenir.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	NYP'nin temelleri demonstrasyon yöntemi ile anlatılır.
Ders İçi Uygulamalar: (Senkron)	Bu hafta, NYP paradigmasının temelleri anlatılır. Sınıf ve nesne kavramlarının üzerinde durulur. Bunun yanında özellik ve davranış kavramları ile erişim belirleyicilerden bahsedilir. Son olarak sınıflara ait UML sınıf diyagramlarının oluşturulması konusu anlatılır. Sınıf ve nesne açısından kullanılabilecek örnekler attribute'lar aşağıdaki gibidir: 

	Araba Sınıfının Davranış ve Özellikleri  **Özellikler** Renk Marka Model Vites Tipi Yakıt Tipi Motor Hacmi Ek Donanımlar **Davranışlar** Çalıştır Vites Değiştir Hızlan Fren Yap Durdur Derste kullanılabilecek UML sınıf diyagramı örneği aşağıdaki gibidir: **Oyuncu** + Adi: string + CanSayisi: int + BonusSayisi: int + Hizi: float + Oyuncu (): void + AdiYazdir(): string + CanSayisiniAzalt(): void + BonusSayisiniArtir(): void + HiziDegistir(float): void Son olarak Roll-A-Ball oyun projesi incelenerek bu projede yer alan NYP mekanizmaları çözümlenir. Bu projenin ekran alıntısı aşağıdaki gibidir: 
Araştırma ve Uygulamalar: (Asenkron)	Roll-A-Ball oyun projesi incelenerek bu projede yer alan NYP mekanizmaları çözümlenir.

6’NCI HAFTA DERS PLANI

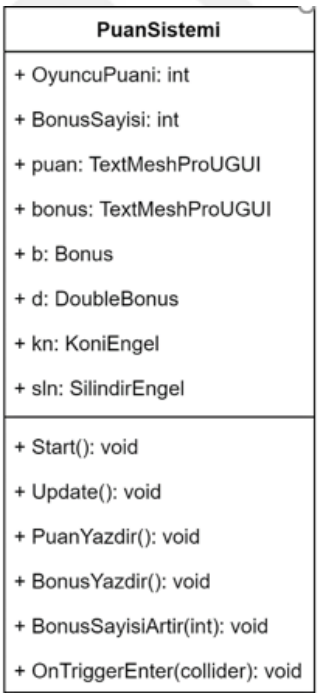
Konu:	Sınıflar ve Nesneler
Tarih ve Saat:	1 Aralık 2020 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) Unity 3D’de oluşturduğu oyun projesine Karting Microgame assetini ekler. 2) Karting Microgame oyun prototipine çeşitli Unity 3D assetlerini ekler.
Açıklamalar:	Bu haftaki derste yeni bir Unity 3D oyun projesi oluşturulur. Bu projeye Karting Microgame asset’i eklenir. Gelecek haftalarda bu oyun prototipi üzerinde ne tür eklentiler oluşturulacağı hakkında bilgi verilir. Bu oyun prototipinde kullanılacak assetler projeye eklenir.

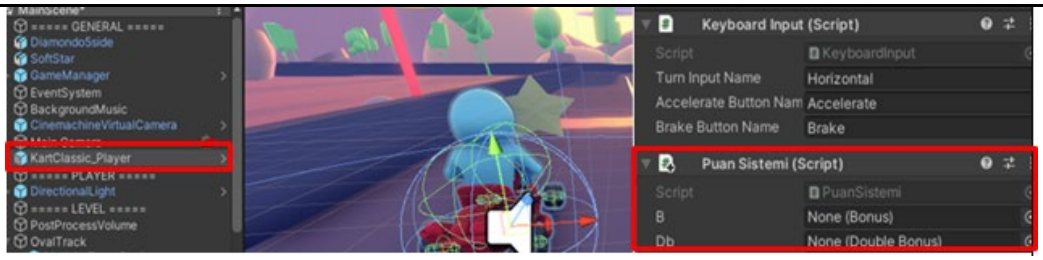
Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Karting Microgame oyun prototipi demonstrasyon yöntemi ile anlatılır.
Ders İçi Uygulamalar: (Senkron)	Bu hafta, dönem boyu geliştirilecek Karting Microgame hakkında bilgi verilir. Bu prototipe ne tür eklentiler yapılacağı tartışılır. Yeni bir Unity 3D oyun projesi oluşturulur ve Karting Microgame asset’i bu projeye eklenir. Karting Microgame assetinin ekran alıntısı aşağıdaki gibidir:  Bu oyun prototipinde kullanılacak assetler tanıtılır ve projeye nasıl ekleneceği konusunda bilgi verilir. Kullanılacak assetlerin ekran alıntıları aşağıdaki gibidir:  Traffic Pack  Bonus Pack

	 **Sound Pack**
Araştırma ve Uygulamalar: (Asenkron)	Her bir öğrenci dönem boyu üzerinde çalışacağı oyun projesini Unity 3D'de oluşturur. Bu projeye Karting Microgame asset'ini ekler. İlave olarak prototipe eklentiler yapılırken kullanılacak assetleri projeye ekler.

7’NCİ HAFTA DERS PLANI

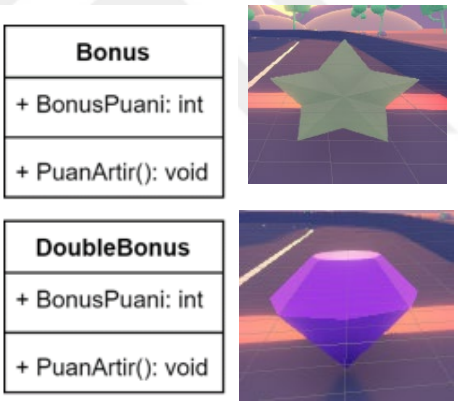
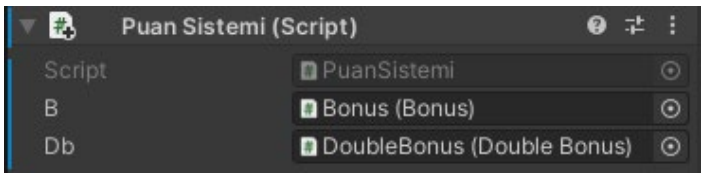
Konu:	Sınıflar ve Nesneler
Tarih ve Saat:	8 Aralık 2020 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) Karting Microgame oyun projesinde PuanSistemi sınıfını oluşturur. 2) Sınıf değişkenlerinin erişim belirleyicilerini ayarlar. 3) Oluşturduğu sınıfı Karting Player nesnesine bileşen(component) olarak ekler. 4) Oluşturulan sınıfın UML sınıf diyagramını çizer.
Açıklamalar:	Bu haftaki derste bireysel oyun projelerine sınıfların eklenmesine başlanır. Bu kapsamda Puan Sistemi sınıfı oluşturulur ve oyuncuya bileşen olarak eklenir.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Puan Sistemi sınıfının oluşturulması canlı kod yöntemi ile anlatılır.
Ders İçi Uygulamalar: (Senkron)	Bu hafta, dönem boyu geliştirilecek Karting Microgame projesine modların (eklentilerin) oluşturulmasına başlanır. NYP’de yer alan sınıf ve nesne kavramları bu hafta Puan Sistemi sınıfının oluşturulmasıyla anlatılır. Puan Sistemi sınıfına ait örnek bir UML sınıf diyagramı aşağıdaki gibidir:  Bu sınıf tanımlanarak Karting Player nesnesine bileşen olarak eklenir. Bu işleme ait örnek bir ekran alıntısı aşağıdaki gibidir:

	
Araştırma ve Uygulamalar: (Asenkron)	Her bir öğrenci Puan Sistemi sınıfını bireysel oyun projesinde oluşturur.

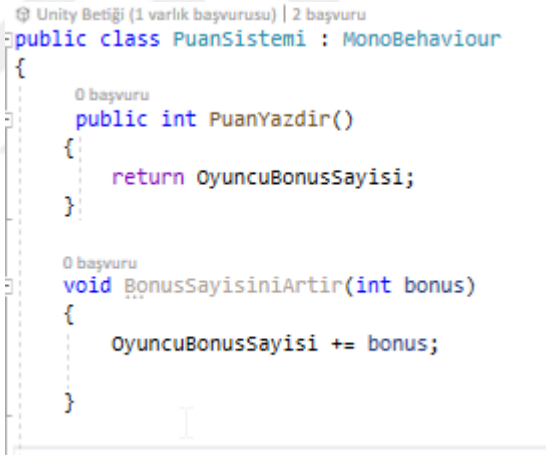
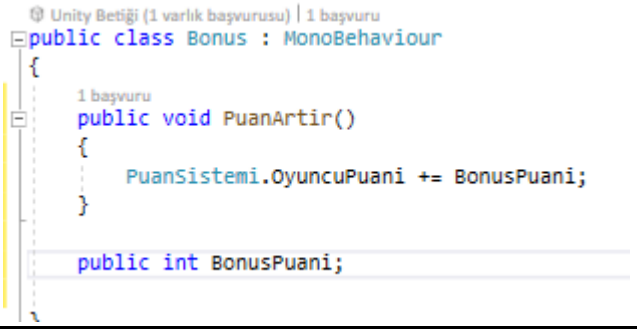
8’İNCİ HAFTA DERS PLANI

Konu:	Sınıflar ve Nesneler
Tarih ve Saat:	15 Aralık 2020 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) Karting Microgame oyun projesinde bonus sınıflarını oluşturur. 2) Oluşturduğu sınıfları bonus nesnelerine bileşen(component) olarak ekler. 3) Oluşturduğu nesneleri prefab’e dönüştürür. 4) Oluşturulan sınıfların UML sınıf diyagramını çizer. 5) Oyun projesine UI nesnesi ekler.
Açıklamalar:	Oyuncunun bonus nesnelerinden puan toplayabilmesini sağlayan puan sisteminin kodlanmasına kapsamında bonus sınıfları oluşturulur. Bu sınıflar ilgili nesnelere bileşen olarak eklenir. Daha sonra kullanılabilmesi maksadıyla bonus nesneleri prefab’e dönüştürülür.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Bonus sınıflarının oluşturulması canlı kod yöntemi ile anlatılır.
Ders İçi Uygulamalar: (Senkron)	Bonus sınıflarına ait örnek bir UML sınıf diyagramı aşağıdaki gibidir:  Bonus sınıfları oluşturulduktan sonra ilgili nesnelere bileşen olarak eklenir. Daha sonra kullanılabilmesi maksadıyla bonus nesneleri prefab’e dönüştürülür. Puan Sistemi sınıfında bu sınıflara referansların yer alması sebebiyle bu prefabler Puan Sistemi sınıfındaki değişkenlere atanır. Bununla ilgili örnek bir gösterim aşağıdaki gibidir: 
Araştırma ve Uygulamalar: (Asenkron)	Her bir öğrenci bonus sınıflarını bireysel oyun projesinde oluşturur.

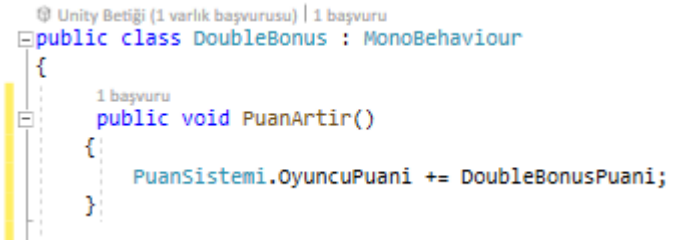
9'UNCU HAFTA DERS PLANI

Konu:	Sınıf Metotları
Tarih ve Saat:	22 Aralık 2020 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) NYP'de davranış kavramının kullanım amaçlarını açıklar. 2) Nesneler arası mesajlaşmanın çalışma prensiplerini açıklar. 3) Puan Sistemi sınıfının PuanYazdir() ve BonusYazdir() metotları ile bonus sınıflarının PuanArtir() metodlarını tanımlar.
Açıklamalar:	Bu hafta NYP'de kullanılan davranış kavramının üzerinde durulur. Nesnelerin birbirleriyle nasıl haberleştiğine yönelik anlatımlarda bulunulur.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Puan Sistemi ve bonus sınıflarının davranışlarının oluşturulması canlı kod yöntemi ile anlatılır.
Ders İçi Uygulamalar: (Senkron)	Bu haftadaki derste Puan Sistemi sınıfının PuanYazdir() ve BonusYazdir() metotları ile bonus sınıflarının PuanArtir() metodları tanımlanır. Puan Sistemi sınıfına ait davranış (sınıf metodu) tanımlamalarına ait örnek bir kodlama aşağıdaki gibidir:  Benzer şekilde bonus sınıflarına ait davranış (sınıf metodu) tanımlamalarına ait örnek bir kodlama aşağıdaki gibidir: 

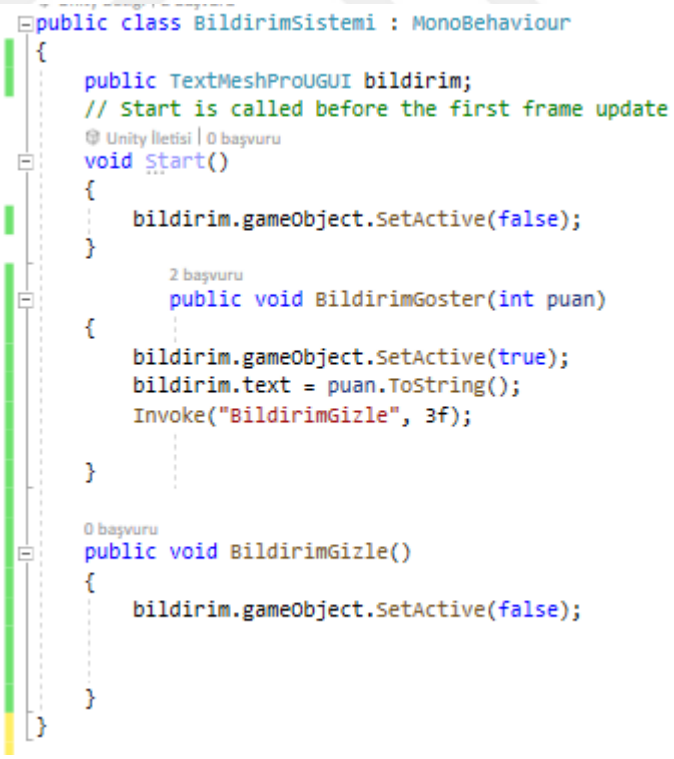
```
Unity Betiği (1 varlık başvurusu) | 2 başvuru
public class PuanSistemi : MonoBehaviour
{
    0 başvuru
    public int PuanYazdir()
    {
        return OyuncuBonusSayisi;
    }

    0 başvuru
    void BonusSayisiniArtir(int bonus)
    {
        OyuncuBonusSayisi += bonus;

    }
```

```
Unity Betiği (1 varlık başvurusu) | 1 başvuru
public class Bonus : MonoBehaviour
{
    1 başvuru
    public void PuanArtir()
    {
        PuanSistemi.OyuncuPuani += BonusPuani;
    }

    public int BonusPuani;
}
```

	
Araştırma ve Uygulamalar: (Asenkron)	Her bir öğrenci sınıf metotlarını bireysel oyun projesinde oluşturur.

10'UNCU HAFTA DERS PLANI

Konu:	Sınıf Metotları
Tarih ve Saat:	29 Aralık 2020 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) NYP'de davranış kavramının kullanım amaçlarını açıklar. 2) Nesneler arası mesajlaşmanın çalışma prensiplerini açıklar. 3) Tanımladığı sınıfta Monobehaviour metotlarını kullanır. 4) Puan Sistemi sınıfının Update() ve OnTriggerEnter() metodlarını tanımlar. 5) Nesnelerin aktif ve pasif olmasını sağlar. 6) Sınıf metotların otomatik çalışmasını (invoke) sağlar. 7) Oyuna bildirim sistemi ekler.
Açıklamalar:	Bu hafta NYP'de kullanılan davranış kavramının üzerinde durulur. Nesnelerin birbirleriyle nasıl haberleştiğine yönelik anlatımlarda bulunulur.

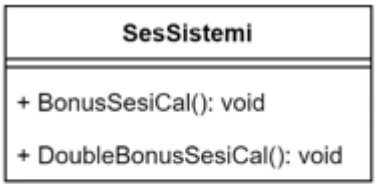
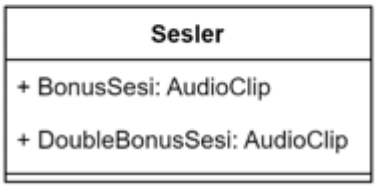
Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Puan Sistemi sınıfının Monobehaviour davranışlarının ve oyuna bildirim sisteminin oluşturulması canlı kod yöntemi ile anlatılır.
Ders İçi Uygulamalar: (Senkron)	Bu haftadaki derste Puan Sistemi sınıfının Update() ve OnTriggerEnter() metotları tanımlanır. Bununla birlikte oyuna bildirim sistemi mod olarak eklenir. Bildirim sistemi için BildirimSistemi sınıf tanımlanır. Bu sınıfın tanımlamalarına ait örnek bir kodlama aşağıdaki gibidir:  Puan Sistemi sınıfına ait davranış (sınıf metodu) tanımlamalarına ait örnek bir kodlama aşağıdaki gibidir:

```
public class BildirimSistemi : MonoBehaviour
{
    public TextMeshProUGUI bildirim;
    // Start is called before the first frame update
    Unity İletisi | 0 başvuru
    void Start()
    {
        bildirim.gameObject.SetActive(false);
    }

        2 başvuru
        public void BildirimGoster(int puan)
    {
        bildirim.gameObject.SetActive(true);
        bildirim.text = puan.ToString();
        Invoke("BildirimGizle", 3f);


    }

    0 başvuru
    public void BildirimGizle()
    {
        bildirim.gameObject.SetActive(false);


    }
}
```

	`` public class PuanSistemi : MonoBehaviour { Unity İletisi | 0 başvuru private void OnTriggerEnter(Collider other) { if (other.CompareTag("Bonus")) { b.PuanArtir(); GameObject.FindObjectOfType<BildirimSistemi>().BildirimGoster(b.BonusPuani); other.gameObject.SetActive(false); } if (other.CompareTag("DoubleBonus")) { db.PuanArtir(); GameObject.FindObjectOfType<BildirimSistemi>().BildirimGoster(db.DoubleBonusPuani); other.gameObject.SetActive(false); } } ```
Araştırma ve Uygulamalar: (Asenkron)	Her bir öğrenci bildirim sistemini projesine ekler. Bunun yanında Puan Sistemi sınıfına ait sınıf metotlarını bireysel oyun projesinde oluşturur.

11'İNCİ HAFTA DERS PLANI

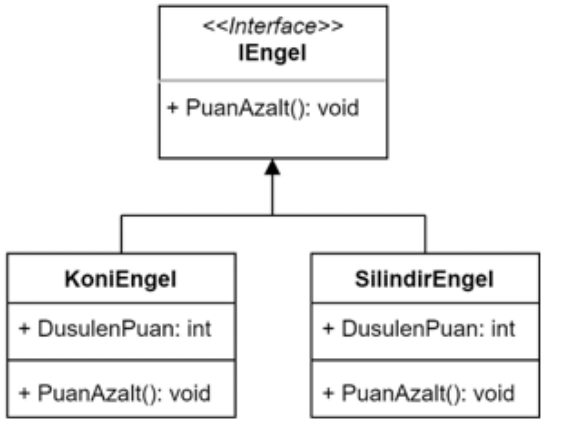
Konu:	Veri Yapıları
Tarih ve Saat:	5 Ocak 2021 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) Statik sınıf, özellik ve davranış kavramlarını açıklar. 2) Statik sınıflarla normal sınıfları karşılaştırır. 3) Oyun prototipinde statik metotları tanımlar. 4) Oyun prototipinde statik sınıf tanımlar. 5) Oyuna ses sistemi modu ekler.
Açıklamalar:	Bu hafta NYP'de kullanılan veri yapıları kavramının üzerinde durulur. Statik sınıf ve metotların kullanımına yönelik açıklamalarda bulunulur.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Statik sınıf, özellik ve davranışların NYP'de hangi amaçlarla kullanıldığına yönelik öğrenciler bilgilendirilir. Uygulama saatinde ise oyuna ses sistemi eklenir. Bu sistem eklenirken hem statik metotlar; hem de kurucu metotlar kullanılır.
Ders İçi Uygulamalar: (Senkron)	Statik metotların kullanılması için SesSistemi ve Sesler ismine iki adet sınıf tanımlanır. Bu sınıflara ait örnek bir UML sınıf diyagramları aşağıdaki gibidir:   Sesler sınıfına ait örnek kodlamalar aşağıdaki gibidir: `public class Sesler : MonoBehaviour` `{` `  public AudioClip BonusSesi;` `  public AudioClip DoubleBonusSesi;` `  public AudioClip EngelSesi;` `}` Bonus ya da engel nesneleriyle çarpışma durumunda seslerin çalınabilmesi için, statik metotlar içeren ve kendisi de statik bir sınıf olan SesSistemi isminde bir sınıfa ihtiyaç bulunmaktadır. Bu sınıfa ait örnek kodlamalar aşağıdaki gibidir:

	0 başvuru public static class SesSistemi { 0 başvuru public static void BonusSesiCal() { GameObject ses = new GameObject("Sound"); AudioSource sesKaynagi = ses.AddComponent<AudioSource>(); sesKaynagi.PlayOneShot(GameObject.FindObjectOfType<Sesler>().BonusSesi); } 0 başvuru public static void DoubleBonusSesiCal() { GameObject ses = new GameObject("Sound"); AudioSource sesKaynagi = ses.AddComponent<AudioSource>(); sesKaynagi.PlayOneShot(GameObject.FindObjectOfType<Sesler>().DoubleBonusSesi); } }
Araştırma ve Uygulamalar: (Asenkron)	Her bir öğrenci ses sistemini projesine ekler.

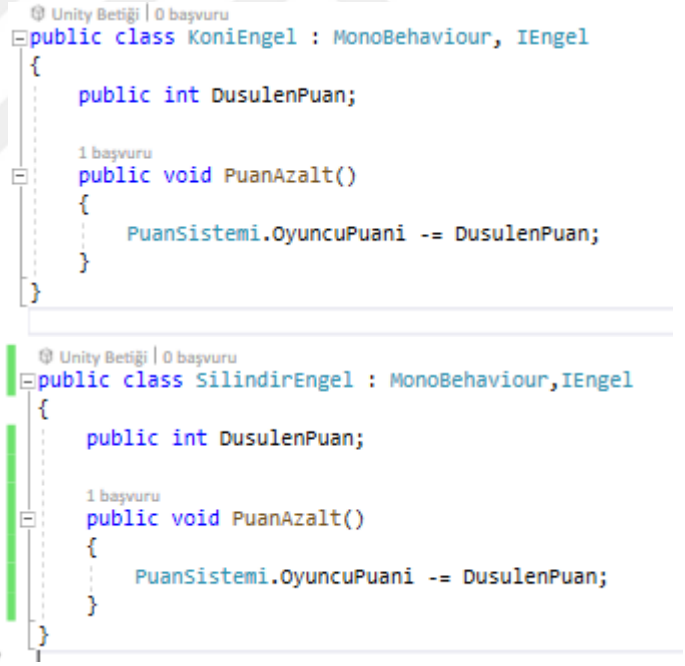
12’NCİ HAFTA DERS PLANI

Konu:	Arayüzler ve Kalıtım
Tarih ve Saat:	12 Ocak 2021 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) Kalıtım (inheritance) kavramını açıklar. 2) NYP’de kalıtımın kullanım amaçlarını listeler. 3) Üst-sınıf (superclass) ve alt-sınıf (subclass) kavramlarını açıklar. 4) Arayüz (interface) kavramını açıklar. 5) Soyut (virtual) sınıfların özelliklerini listeler.
Açıklamalar:	Bu hafta; NYP’de kalıtım mekanizmasının kullanım amaçları anlatılır. Kalıtımın çalışma prensiplerinin anlatılmasından sonra oyun prototipinde engel sınıflarının kalıtımla oluşturulması sağlanır.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Canlı ders saatlerinde kalıtım kavramı, kullanım amaçları, süt-sınıf/alt-sınıf kavramları üzerinde durulur. Soyut sınıflar ve arayüzler hakkında bilgilendirmede bulunulur. Aryayüzlerin kullanım amaçları aktarılır.
Ders İçi Uygulamalar: (Senkron)	Uygulama olarak oyun prototipinde IEngel isminde bir arayüz oluşturulur. KoniEngel ve SilindirEngel isminde iki sınıfın IEngel arayüzünden türetilmesi sağlanır. Bu sınıflara ait örnek bir UML sınıf diyagramları aşağıdaki gibidir:   **KoniEngel**  **SilindirEngel**
Araştırma ve Uygulamalar: (Asenkron)	Her bir öğrenci IEngel arayüzünü oluşturur. KoniEngel ve SilindirEngel sınıflarını bu arayüzden kalıtımla türeterek oyun prototipine ekler.

13'ÜNCÜ HAFTA DERS PLANI

Konu:	Arayüzler ve Kalıtım
Tarih ve Saat:	19 Ocak 2021 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) Veri Gizleme İlkesi açıklar. 2) Görünebilirlik kurallarını listeler. 3) Çokbiçimlilik kavramını açıklar. 4) Oyun Prototipinde çokbiçimli metotları tanımlar ve kullanır.
Açıklamalar:	Bu hafta; NYP'de çokbiçimliliğin kullanım amaçları anlatılır. Veri gizleme ilkesi, görünebilirlik kuralları üzerinde durulur. Geçen hafta oluşturulan KoniEngel ve SilindirEngel sınıflarının çokbiçimli davranışlarının oluşturulmasına devam edilir.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Canlı ders saatlerinde çokbiçimliliğin kullanım amaçları, veri gizleme ilkesi, görünebilirlik kuralları kavramları üzerinde durulur.
Ders İçi Uygulamalar: (Senkron)	Uygulama olarak; daha önce oluşturulan KoniEngel ve SilindirEngel sınıflarının PuanAzalt() metotları oluşturulur. Bu çokbiçimli metotlara ait örnek kodlamalar aşağıdaki gibidir:  KoniEngel ve SilindirEngel sınıflarının tanımlanmasının ardından bu sınıflar koni ve silindir oyun nesnelerine bileşen olarak eklenir. Aynı zamanda bu nesnelerle çarpışma durumuna özel olarak oyunun ses sistemi kodları güncellenir. Bu sayede oyuncunun bu nesnelere çarpması durumunda bonus nesnelerinden farklı sesler çıkarılmış olur.
Araştırma ve Uygulamalar: (Asenkron)	Her bir öğrenci IEngel arayüzünü oluşturur. KoniEngel ve SilindirEngel sınıflarını bu arayüzden kalıtımla türeterek oyun prototipine ekler.

14'ÜNCÜ HAFTA DERS PLANI

Konu:	Hata Kotarımı, Görsel Efektler ve Pist Tasarımı
Tarih ve Saat:	26 Ocak 2021 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	1) Oyun projesinde en yaygın karşılaşılan hataları listeler. 2) Oyun projesindeki hataları giderir. 3) Pist tasarımını değiştirir. 4) Oyuna görsel efektler ekler.
Açıklamalar:	Bu hafta Karting Microgame oyun prototipinde en yaygın karşılaşılan hatalar üzerinde durulur. Bunun yanında pist tasarımı, oyuncu tasarımı değiştirilir. İlave olarak oyuna görsel efektler eklenir.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Canlı ders saatlerinde oyunj projesinde en yaygın karşılaşılan hatalardan olan nesne başvuru hataları, sınıfın bileşen olarak eklenememesi, metot parametre hataları, statik metotlara ve değişkenlere erişilememe gibi yazılım hatalarının nedenlerinden bahsedilir. Bu hataların çözümüne yönelik atılması gereken adımlar açıklanır. İlave olarak pist tasarımı ve oyuncu tasarımı değiştirilir. İlave olarak oyuna görsel efektler eklenir.
Ders İçi Uygulamalar: (Senkron)	Uygulama olarak oyun prototipinin görsel öğelerinin değiştirilmesi uygulanır. Pist tasarımının nasıl değiştirilebileceği gösterilir. Farklı pist tasarım öğeleri tanıtılır. Bunlardan bazıları aşağıda verilmiştir:  Pist tasarımına ilave olarak oyuncuya eklenebilecek görsel efektlerden bahsedilir. Örnek efektler aşağıda verilmiştir: 
Araştırma ve Uygulamalar: (Asenkron)	Her bir öğrenci oyun projesinin pist tasarımını bireysel isteklerine göre değiştirir. Bunun yanında oyuna görsel efektler ekler.

15’İNCİ HAFTA DERS PLANI

Konu:	Öğrenci Proje Sunumları
Tarih ve Saat:	2 Şubat 2021 Salı 15:00-17:00
Süre:	3 saat
Kazanımlar:	Bireysel olarak hazırladığı oyun projesinin tanıtımını yapar.
Açıklamalar:	Bu hafta öğrenciler hazırladıkları oyun projelerinin tanıtımını yaparlar.

Öğretme ve Öğrenme Süreci:	
Canlı Ders Seansı: (Senkron)	Öğrenciler bireysel oyun projelerinin tanıtımı yapar. Bununla birlikte öğrencileri yaşadıkları deneyimleri ve gelecekte başarmak istediklerini aktarırlar.
Ders İçi Uygulamalar: (Senkron)	-
Araştırma ve Uygulamalar: (Asenkron)	-