

T.C.
Marmara Üniversitesi
Eğitim Bilimleri Enstitüsü
Bilgisayar ve Öğretim Teknolojileri Eğitimi Anabilim Dalı
Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Bilim Dalı

PROGRAMLAMA EĞİTİMİNDE GÖRSEL BLOK
PROGRAMLAMA VE MOBİL UYGULAMA GELİŞTİRME
ARAÇLARININ KARŞILAŞTIRILMASI

Can MIHCI
(Yüksek Lisans Tezi)

İstanbul – 2014

T.C.

Marmara Üniversitesi
Eğitim Bilimleri Enstitüsü
Bilgisayar ve Öğretim Teknolojileri Eğitimi Anabilim Dalı
Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Bilim Dalı

BİLİŞİM ÖĞRETMENİ ADAYLARINA YÖNELİK
PROGRAMLAMA EĞİTİMİNDE GÖRSEL BLOK
PROGRAMLAMA VE MOBİL UYGULAMA GELİŞTİRME
ARAÇLARI

Can MIHCI
(Yüksek Lisans Tezi)

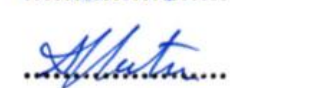
Danışman:
Doç. Dr. Nesrin ÖZDENER

İstanbul – 2014

**Tüm kullanım hakları
M.Ü. Eğitim Bilimleri Enstitüsü'ne aittir.
© 2011**

ONAY

Can Mıhçı tarafından hazırlanan “Programlama Eğitiminde Görsel Blok Programlama ve Mobil Uygulama Geliştirme Araçlarının Karşılaştırılması” konulu bu çalışma, 16.06.2014 tarihinde yapılan savunma sınavı sonucunda jüri tarafından başarılı bulunmuş ve yüksek lisans tezi olarak kabul edilmiştir.

	Adı Soyadı	İmza
TEZ DANIŞMANI	Doç. Dr. Nesrin Özdener	
JÜRİ ÜYESİ	Doç. Dr. Ali Buldu	
JÜRİ ÜYESİ	Yrd. Doç. Dr. Ahmet Feyzi Satıcı	

ÖZGEÇMİŞ

2001	Adana Seyhan ÇEAŞ Anadolu Lisesi
2009	Çukurova Üniversitesi Eğitim Fakültesi Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü'nden mezuniyet
2010	ÖYP çerçevesinde Trakya Üniversitesi Eğitim Fakültesi'nde Araştırma Görevlisi görevine başlangıç
2010	Marmara Üniversitesi Eğitim Bilimleri Enstitüsü Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Bilim Dalı Yüksek Lisans Programına giriş.
2011	Marmara Üniversitesi Eğitim Bilimleri Enstitüsü'ne Araştırma Görevlisi olarak naklen atanma

İLETİŞİM BİLGİLERİ

Görev Yaptığı Kurum	Marmara Üniversitesi Atatürk Eğitim Fakültesi Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü
E-posta	can.mihci@marmara.edu.tr
Web Sitesi	www.canmihci.com
Telefon	+90 541 3754315

ÖNSÖZ

İçinde bulunduğumuz yüzyılda insanlığın tamamı için önem arz eden ve giderek medeniyetin ayrılmaz bir parçası hale gelen bilgisayar bilimleri, erken çocukluk yıllarımdan beri büyük ilgi duyduğum bir alan oldu. Yaygın bilişim devrimine tanıklık ettiğimiz bu günlerde ise, gerek genç neslin bilgisayarın karmaşık dünyasında kaybolmadan ilerleyebilmesi için; gerekse makine ve insan arasındaki uçurumun kapatılabilmesi için yapılması gereken çok iş olduğu ise muhakkak. Acemilere yönelik bilgisayar programlama eğitimi konusunda nice saygın araştırmacının çizdiği yolu takip etmeye çalışan bu araştırma ile, bu çabaya bir nebze olsun katkıda bulunabilmiş olmayı umuyorum.

Bu bağlamda; başlangıçtan bu yana hem akademik hem de insani vasıflarıyla bana örnek teşkil eden ve çalışmada en az benim kadar emeği bulunan çok değerli danışmanım Doç. Dr. Nesrin Özdener'e içtenlikle teşekkür ediyorum. Marmara Üniversitesi Atatürk Eğitim Fakültesi Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü'ndeki yüksek lisans eğitimim boyunca bana değerli fikirleriyle katkıda bulunan, akademisyenliği sevdiren ve bu esnada esirgemedikleri dostluklarıyla destek olan tüm diğer hocalarıma ve öğrenci arkadaşlarıma da ayrıca şükranlarımı sunuyorum. En nihayet, anmadan geçemeyeceğim ve hayatımın en zorlu dönemlerinde dahi beni hep desteklemiş olan anneme, babama ve kardeşime her şey için teşekkür ediyorum.

Bu çalışma, Marmara Üniversitesi Bilimsel Araştırma Projeleri Komisyonu EGT-C-YLP-150513-0211 numaralı projesi kapsamında yürütülmüştür.

ÖZET

Her üç öğrenciden birinin başarısız bulunduğu saptanan üniversite seviyesinde ilk programlama dersinin; konuları açısından öğrenilmesi en zor derslerden biri olduğunu savunulmaktadır. Son zamanlarda bu problemin üstesinden gelinebilmesi için, blok temelli görsel programlama dillerinin eğitsel faaliyetleri kolaylaştıran araçlar olarak kullanıldığı eğitim araştırmalarına rastlanır olmuştur. Bilhassa yakın zamanda, günümüzde popülerliği artan mobil akıllı cihazlarda uygulama geliştirmeye dönük blok-temelli programlama dillerinin eğitimde kullanımı önerilmiştir.

Bu çalışmada ise amaç, üniversite seviyesinde GUI-programlama eğitiminde eğitsel araç olarak mobil platformlar için GUI uygulama geliştirmeye yönelik blok-temelli bir görsel programlama dili kullanmanın; masaüstü GUI uygulama geliştirmeye yönelik metin tabanlı bir programlama dili kullanmaya kıyasla öğrencilerin GUI programlama dersine yönelik akademik başarı ve motivasyonu üzerindeki etkisini deneysel olarak incelemektir. Araştırmanın çalışma grubunu Marmara Üniversitesi Bilgisayar ve Öğretim Teknolojileri Öğretmenliği (BÖTE) ikinci ve üçüncü sınıf öğrencileri oluşturmuştur. 12 Haftalık eğitim uygulaması boyunca öğrencilerin akademik başarı ve derse yönelik motivasyonu; eğitimde kullanılan araç, araçların kullanım sırası, üniversite öncesi GUI-programlama deneyimi, Programlama Dilleri 1 dersi başarı ve akıllı mobil cihaz aidiyeti faktörlerine göre istatistiksel olarak incelenmiştir. Uygulamanın üzerinden geçen 1 yıllık sürenin ardından yapılan bir anketle, blok-temelli araçla eğitim almış olmanın bilişim öğretmeni adayı gruplarda; mesleki açıdan nasıl bir etki bırakmış olabileceği anlaşılmaya çalışılmıştır.

Sonuç olarak, mobil uygulama geliştirmeye yönelik blok-temelli aracın; metin temelli programlamada başarısız bulunmuş öğrencilerin başarısını arttırmada etkili olduğu; ancak metin temelli dillerin kullanımına alışmış öğrencilerin blok-temelli programlamaya adapte olamadığı ve başarılarının düştüğü görülmüştür. Bilgisayar öğretmeni adayları açısından ise, blok-temelli aracın kişisel kullanım için benimsenmediği gibi; mesleki açıdan da rağbet görmediği anlaşılmıştır.

Anahtar Kelimeler: Programlama eğitimi, görsel programlama, blok-temelli programlama, mobil uygulama geliştirme, GUI programlama, App Inventor

ABSTRACT

With a failure rate of one out of three students, it has been suggested that the CS course at college level is one of the most difficult subjects at undergraduate level. Literature suggests that one of the recent approaches for overcoming this problem has been to employ blocks-based visual programming languages as educational tools that make learning programming easier. Especially, the use of a particular blocks-based visual language for developing applications that run on today's highly popular smart mobile devices has been emphasized for use in college education with novices and seasoned students alike.

The aim of this study is to experimentally compare the use of a conventional text-based programming language typically used for desktop application development; with a blocks-based programming language used for mobile application development in terms of impact upon student academic success and motivation in a college level GUI-programming course aimed at preservice computer science teachers. The study group for the research has been the junior and sophomore year students at Marmara University Ataturk Faculty of Education Department of Computer Education and Instructional Technologies. The academic success and course motivation of students has been assessed statistically throughout a 12-week course as influenced by factors such as the programming language used; the consecutive usage order of the said programming languages, pre-college experience in GUI-programming concepts, success in a previous CS1 course at college level and the ownership of smart mobile devices. A survey conducted 1 year after the course has finished has also been used in order to understand how learning programming through a blocks-based visual programming language affects the professional capacities of preservice teachers.

Consequently, it has been understood that a blocks-based visual programming language for developing mobile applications can be beneficial in increasing the success rate of students who have failed to adapt a programming course given with text-based languages. However, it has also been observed that students who are proficient in text-based programming either fail to adapt or perform poorly in using a blocks-based language. As for preservice computer teachers; it has been observed that the blocks-

based language for mobile application development is not preferred for either personal or professional use.

Keywords: Programming education, visual programming, block based programming, mobile application development, GUI programming, App Inventor

İÇİNDEKİLER

1	ONAY	Error! Bookmark not defined.
2	ÖZGEÇMİŞ.....	ii
3	İLETİŞİM BİLGİLERİ.....	ii
4	ÖNSÖZ.....	iii
5	ÖZET.....	iv
6	ABSTRACT	v
8	TABLolar LİSTESİ	xi
9	ŞEKİLLER LİSTESİ.....	xiv
10	GRAFİKLER LİSTESİ	xv
11	KISALTMALAR VE SEMBOLLER	xvi
1	BÖLÜM I: GİRİŞ	1
1.1.	Problem	1
1.2.	Amaç / Hipotezler	5
1.2.1.	Hipotezler.....	6
1.2.2.	Araştırma Soruları.....	9
1.3.	Önem.....	9
1.4.	Sınırlılıklar	10
1.5.	Sayıtlılar	11
1.6.	Tanımlar	11
1.6.1.	Konsol-Uygulama	11
1.6.2.	Grafik Kullanıcı Arayüzü (GUI) ve GUI-Programlama.....	12
1.6.3.	Görsel Programlama Dili	12
1.6.4.	Metin tabanlı Programlama Dili.....	12
1.6.5.	Microsoft Visual C# (MTPD)	12
1.6.6.	Mobil Akıllı Cihaz.....	12
1.6.7.	Android İşletim Sistemi	13
1.6.8.	MIT App Inventor (MBTGPD)	13
1.6.9.	Programlama Dilleri 1 Dersi (PD1)	13
1.6.10.	Deneyimli Öğrenciler.....	13
1.6.11.	Acemi Öğrenciler	14
2	BÖLÜM II: İLGİLİ ARAŞTIRMALAR / ALANYAZIN.....	15
2.1	Üniversite Düzeyinde Programlama Eğitimi	15
2.2	Üniversite Öncesinde Programlama Eğitimi.....	16
2.3.	Programlama Paradigmaları.....	17

2.3.1.	Yordamsal Programlama	17
2.3.2.	Nesne-Yönelimli Programlama	18
2.3.3.	Kaynak kod ve Sözdizimi	19
2.3.4.	Konsol-Uygulamalar	20
2.3.5.	Programlamada Grafik Kullanıcı Arayüzleri	20
2.3	Acemilere Yönelik Programlama Eğitimi - Doğru Programlama Dilinin Seçimi.....	21
2.4	Nesne Yönelimli Dillerle Acemilere Yönelik Programlama Eğitimi – Programlama Paradigmalarının Sunum Sırası.....	22
2.4.1	Temel-yapı öncelikli yaklaşım	25
2.4.2	Nesne-Öncelikli Yaklaşım.....	26
2.4.3	GUI-Öncelikli Yaklaşım.....	27
2.5	Görsel Blok Programlama ve Programlama Eğitiminde Kullanımı.....	28
2.5.1	Pedagojik Temel: Eğitimde Yapımcılık (Constructionism).....	28
2.5.2	Görsel Blok Programlama Dilleri	29
2.5.3	Mobil Cihazlarla Programlama Eğitimi	30
2.5.4	Mobil-Uygulama Geliştirmeye Yönelik Görsel Programlama Dilleri	33
3	BÖLÜM III: YÖNTEM.....	35
3.1	Araştırma Modeli	35
3.2	Çalışma Grubu	38
3.3	Veri Toplama Araçları	38
3.3.1	Ön Bilgi Testi 1	38
3.3.2	GUI Programlama Eğitimi / Akıllı Mobil Cihaz Bilgi Formu	39
3.3.3	Ön Bilgi Testi 2	39
3.3.4	Uygulama Sınavı 1	40
3.3.5	Uygulama Sınavı 2.....	40
3.3.6	Motivasyon Ölçeği.....	41
3.3.7	Uygulama Sonrası Anketi	42
3.3.8	Mikroöğretim Uygulaması	42
3.4	Verilerin Toplanması	43
3.5	Verilerin Çözümlemesi.....	43
3.5.1	Nicel Veriler Üzerinde Yürütülen İstatistikler.....	44
3.5.2	Faktörler Arası Etkileşim: Tek Değişkenli Faktöriyel Varyans Analizi	44
3.6	Uygulama	45
4	BÖLÜM IV: BULGULAR.....	54
4.1	Uygulama Öncesi Bulgular	54
4.1.1	Gruplarda Karşılaştırılabilirlik Testleri	54

4.1.2	Faktör Gruplarının Oluşturulması	55
4.2	Akademik Başarıya Yönelik Hipotezlere Ait Bulgular	58
4.2.1	Kullanılan Aracın GUI Programlama Akademik Başarısına Etkisi	58
4.2.2	Kullanılan Araç ve Üniversite Öncesi GUI Programlama Eğitimi Faktörlerinin Akademik Başarıya Etkileri Arasında Etkileşim.....	59
4.2.3	Kullanılan Araç ve Programlama Dilleri 1 Dersi Başarısı Faktörlerinin Akademik Başarıya Etkileri Arasında Etkileşim	61
4.2.4	Kullanılan Araç ve MTPD Kullanımında Deneyimli Olma Faktörleri Arasında Akademik Başarıya Etki Açısından Etkileşim	63
4.2.5	Çapraz Uygulama Sonunda Nihai Akademik Başarı	65
4.2.6	Çapraz Uygulama Sonunda Nihai Başarı: Üniversite Öncesinde GUI Programlama Eğitimi Almamış Öğrenciler	66
4.3	Motivasyona yönelik hipotezlere ait bulgular	66
4.3.1	Kullanılan Aracın Motivasyona Etkisi (Hipotez 2.1)	66
4.3.2	Kullanılan Araç ve Üniversite Öncesi GUI Programlama Eğitimi Faktörlerinin Motivasyona Etkileri Arasında Etkileşim	67
4.3.3	Kullanılan Araç ve Programlama Dilleri 1 Başarısı Faktörlerinin Motivasyona Etkileri Arasında Etkileşim	68
4.3.4	Kullanılan Araç ve MTPD Kullanımında Deneyimli Olma Faktörleri Arasında Motivasyona Etki Açısından Etkileşim	70
4.3.5	Aracın Değiştirilmesi Durumunda Motivasyonda Değişim: Deneyimli Öğrenciler	72
4.3.6	Aracın Değiştirilmesi Durumunda Motivasyonda Değişim: Acemi Öğrenciler .	74
4.4	Cihaz aidiyetine yönelik hipotezlere dair bulgular	75
4.4.1	Mobil araçla öğrenim gören öğrencilerde cihaz aidiyetinin derse yönelik görev değeri algısına etkisi.....	75
4.4.2	Mobil araçla öğrenim gören öğrencilerde cihaz aidiyetinin akademik başarıya etkisi	76
4.5	Araştırma Sorularına ait Bulgular	76
4.5.1	Kullanılan aracın, öğrencilerin kişisel amaçla geliştirmiş olduğu projelerin sayısına etkisi	77
4.5.2	Öğrencilerin yeni bir programlama dilini kolaylıkla öğrenebileceklerine dair inançları.....	78
4.5.3	Öğrencilerin acemilere yönelik programlama eğitimini başarıyla verip veremeyeceklerine dair görüşleri	80
4.5.4	Öğrencilerin ortaokul öğrencilerine GUI-programlama dersi verilmesi hususunda görüşleri	82
4.5.5	Öğrencilerin programlama eğitiminin verilmeye başlanması gerektiği yaşa ilişkin görüşleri	85
4.5.6	Mikro-öğretim uygulamasının analizi (Araştırma Sorusu 7)	86

5	BÖLÜM V: SONUÇ.....	88
5.1	Sonuç ve Tartışma	88
5.2	Öneriler	97
6	KAYNAKLAR.....	98
7	EKLER.....	106
	EK 1: ÖN BİLGİ TESTİ 1	107
	EK 2: ÖN BİLGİ TESTİ 2	109
	EK 3: UYGULAMA SINAVI 1	110
	EK 4: UYGULAMA SINAVI 2	111
	EK 5: UYGULAMA SONU ANKETİ.....	112
	EK 6: ÖRNEK APP INVENTOR PROJESİ: HESAP MAKİNESİ.....	113
	EK 7: MBTGPD KULLANAN ÖĞRENCİ PROJELERİ	114

TABLÖLER LİSTESİ

Tablo 1.1 Hipotezlerin ilişkin bağımlı ve bağımsız değişkenlerine göre dağılımı.....	8
Tablo 3.1 Çalışma Grupları.....	38
Tablo 3.2 Ön Bilgi Testi 1 Madde Belirtke Tablosu.....	39
Tablo 3.3 Ön Bilgi Testi 2 Madde Belirtke Tablosu.....	40
Tablo 3.4 Uygulama Sınavı 1 Madde Belirtke Tablosu	40
Tablo 3.5 Uygulama Sınavı 2 Madde Belirtke Tablosu	41
Tablo 3.6 - Güdülenme ve Öğrenme Stratejileri Alt Ölçeklerinin Cronbach Alfa Katsayıları....	42
Tablo 4.1 Deney gruplarının Ön Bilgi Testi 1 sonuçlarına göre karşılaştırılması – Bağımsız örneklem t-testi sonuçları.....	54
Tablo 4.2 Deney gruplarının Ön Bilgi Testi 1 sonuçlarına göre karşılaştırılması – Mann Whitney U Testi sonuçları	55
Tablo 4.3 Öğrencilerin Yordamsal Programlama Becerisi faktör grubuna tahsis edilmesi.....	56
Tablo 4.4 Faktör gruplarının son görünümü.....	58
Tablo 4.5 Deney gruplarında GUI-programlama açısından “Deneyimli” ve “Acemi” alt gruplar	58
Tablo 4.6 Kullanılan araca göre Uygulama Sınavı 1 ortalamalarında farka ilişkin t-testi sonuçları	59
Tablo 4.7 Üniversite Öncesi GUI-Programlama Eğitimi ve Kullanılan Araç Faktörlerinin Uygulama Sınavı 1 Sonuçlarına etkileri arasında etkileşim - Betimsel İstatistikler ve Normal Dağılım Testleri	59
Tablo 4.8 Araç ve Üni. Öncesi GUI Programlama Eğitimi faktörlerinin Uygulama Sınavı 1 sonuçlarına etkileri arasında etkileşim - 2x2 Faktöriyel Varyans Analizi Sonuçları, Gruplar Arası Etkiler.....	60
Tablo 4.9 Programlama Dilleri 1 Dersi Başarısı ve Kullanılan Araç Faktörlerinin Uygulama Sınavı 1 Sonuçlarına etkileri arasında etkileşim - Betimsel İstatistikler ve Normal Dağılım Testleri	61
Tablo 4.10 Araç ve Programlama Dilleri 1 Dersi Başarısı faktörlerinin Uygulama Sınavı 1 sonuçlarına etkileri arasında etkileşim - 2x2 Faktöriyel Varyans Analizi Sonuçları, Gruplar Arası Etkiler.....	61
Tablo 4.11 MTPD Kullanımında Deneyim ve Kullanılan Araç Faktörlerinin Uygulama Sınavı 1 Sonuçlarına etkileri arasında etkileşim - Betimsel İstatistikler ve Normal Dağılım Testleri	63

Tablo 4.12 Araç ve MTPD kullanımında deneyim faktörlerinin Uygulama Sınavı 1 sonuçlarına etkileri arasında etkileşim - 2x2 Faktöriyel Varyans Analizi Sonuçları, Gruplar Arası Etkiler	63
Tablo 4.13 Kullanılan araca göre acemi ve deneyimli öğrencilerde başarı ortalamalarının farkına ilişkin t-testi sonuçları.....	65
Tablo 4.14 Kullanılan araç sırasına göre nihai başarı değişkeni için U-testi sonucu	65
Tablo 4.15 Kullanılan araç sırasına göre nihai başarı değişkeni için U-testi sonucu	66
Tablo 4.16 Kullanılan Aracın Motivasyon Ölçeği 1. Ölçüm Alt Boyutlarına Etkisine ilişkin t-testi sonuçları	67
Tablo 4.17 Motivasyon Testi 1. ölçüm içsel hedef düzenleme, görev değeri ve özyeterlilik alt boyutlarında Deney Gruplarının üni. Öncesi GUI-eğitimi faktör gruplarına ait skorlarda betimleyici istatistikler ve normal dağılım testleri.....	67
Tablo 4.18 Kullanılan araç x Üni. Öncesi GUI-eğitimi faktörlerinin motivasyon alt boyutları üzerindeki etkilerini inceleyen 2x2 Faktöriyel Varyans Analizleri Sonuçları – Gruplar Arası Etkiler.....	68
Tablo 4.19 Motivasyon Testi 1. ölçüm içsel hedef düzenleme, görev değeri ve özyeterlilik alt boyutlarında Deney Gruplarının PD1 Başarı faktör gruplarına ait skorlarında betimleyici istatistikler ve normal dağılım testleri.....	69
Tablo 4.20 Kullanılan araç x PD1-başarısı faktörlerinin motivasyon alt boyutları üzerindeki etkilerini inceleyen 2x2 Faktöriyel Varyans Analizleri Sonuçları – Gruplar Arası Etkiler.....	69
Tablo 4.21 Motivasyon Testi 1. ölçüm içsel hedef düzenleme, görev değeri ve özyeterlilik alt boyutlarında Deney Gruplarının MTPD Kullanımında deneyim faktör gruplarına ait skorlarında betimleyici istatistikler ve normal dağılım testleri.....	70
Tablo 4.22 Kullanılan araç x MTPD kullanımında deneyim faktörlerinin motivasyon alt boyutları üzerindeki etkilerini inceleyen 2x2 Faktöriyel Varyans Analizleri Sonuçları – Gruplar Arası Etkiler.....	71
Tablo 4.23 Deneyimli Öğrencilerde Motivasyon Testi 1. ve 2. Ölçüm alt boyutu puanları – betimleyici istatistikler ve normallik testi sonuçları	73
Tablo 4.24 Deneyimli Öğrencilerde Motivasyon Testi 2. Ölçümü Puanlarına Yönelik Kovaryans Analizi Sonuçları – Gruplar Arası Etkiler Testleri (Kovaryant = 1. Ölçüm)	73
Tablo 4.25 Acemi Öğrencilerde Motivasyon Testi 1. ve 2. Ölçüm alt boyutu puanları – betimleyici istatistikler ve normallik testi sonuçları	74
Tablo 4.26 Acemi Öğrencilerde Motivasyon Testi 2. Ölçümü İlişkin Alt Boyutlarına Yönelik Kovaryans Analizi Sonuçları – Gruplar Arası Etkiler Testleri (Kovaryant = 1. Ölçüm) ..	75

Tablo 4.27 Kendi akıllı mobil cihazına sahip olmanın MBTGPĐ aracı ile GUI Programlama eđitiminde grev deđeri algısına etkisi - t-testi sonuları	76
Tablo 4.28 Kendi akıllı mobil cihazına sahip olmanın MBTGPĐ aracı ile GUI Programlama eđitiminde akademik bařarıya etkisin - t-testi sonuları	76

ŞEKİLLER LİSTESİ

Şekil 3.1 Araştırma İzlenesi.....	37
Şekil 3.2 “Merhaba” uygulaması. (a) MBTGPD ile, (b) MTPD ile.....	46
Şekil 3.3 Basit Hesap Makinesi uygulaması – (a) MBTGPD ile, (b) MTPD ile	47
Şekil 3.4 Tombala uygulaması – (a) MBTGPD ile, (b) MTPD ile.....	48
Şekil 3.5 Hızlı Yaz uygulaması – (a) MBTGPD ile, (b) MTPD ile	49
Şekil 3.6 Resimli Sözlük Uygulaması – (a) MBTGPD ile, (b) MTPD ile	50
Şekil 3.7 Öğrenci Otomasyonu uygulaması ana menü– (a) MBTGPD ile, (b) MTPD ile.....	51
Şekil 3.8 Otomasyonu uygulaması kayıt ekleme – (a) MBTGPD ile, (b) MTPD ile	52
Şekil 3.9 Hafıza oyunu uygulaması– (a) MBTGPD ile, (b) MTPD ile.....	53
Şekil 4.1 Ön Bilgi Testi 1 skorlarına göre Deney Grubu 1 üyelerinin (a) ve Kontrol grubu 1 üyelerinin (b) 0,25 standart sapma alt ve üst sınıra göre ayrımlanması.....	56
Şekil 4.2 Üniversite Öncesi GUI Programlama Eğitimi Faktör Gruplarının Oluşturulma Şeması	57
Şekil 4.3 Yeni bir görsel programlama dilini kolaylıkla öğrenebileceğinize inanıyor musunuz?” sorusu için kelime bulutları - (a) kontrol grubu 2 (b) deney grubu 2.....	78
Şekil 4.4 Acemilere programlama dersi verebilir misiniz? sorusu için kelime bulutları (a) kontrol grubu 2 (b) deney grubu 2	80
Şekil 4.5 Ortaokul öğrencilerine programlama dersi verilmeli mi? sorusu için kelime bulutları (a) kontrol grubu 2 (b) deney grubu 2.....	83

GRAFİKLER LİSTESİ

Grafik 4-1 Araç ve Üni. Öncesi GUI Programlama Eğitimi faktörlerinin Uygulama Sınavı 1 sonuçlarına etkileri arasında etkileşim - 2x2 Faktöriyel Varyans Analizi Sonuçları, Profil Grafiği.....	60
Grafik 4-2 Araç ve Programlama Dilleri 1 dersi başarısı faktörlerinin Uygulama Sınavı 1 sonuçlarına etkileri arasında etkileşim - 2x2 Faktöriyel Varyans Analizi Sonuçları, Profil Grafiği.....	62
Grafik 4-3 Araç ve MTPD kullanımında deneyim faktörlerinin Uygulama Sınavı 1 sonuçlarına etkileri arasında etkileşim - 2x2 Faktöriyel Varyans Analizi Sonuçları, Profil Grafiği	64
Grafik 4-4 Kullanılan araç x MTPD kullanımında deneyim faktörlerinin motivasyon alt boyutları üzerindeki etkilerini inceleyen 2x2 Faktöriyel Varyans Analizleri Sonuçları – Profil Grafikleri.....	72
Grafik 4-5 Kullanılan aracın, öğrencilerin kişisel amaçla geliştirmiş olduğu programların sayısına etkisi	77
Grafik 4-6 Kullanılan aracın, öğrencilerin yeni bir programlama dilini kolaylıkla öğrenebileceklerine dair inançlarına etkisi, karşılaştırmalı grafik	79
Grafik 4-7 öğrencilerin acemilere yönelik programlama eğitimi verebileceklerine dair inançlarına etkisi, karşılaştırmalı grafik.....	82
Grafik 4-8 Kullanılan aracın, öğrencilerin ortaokulda programlama eğitim verilmesi fikrine dair görüşlerine etkisi, karşılaştırmalı grafik	84
Grafik 4-9 Öğrencilerin programlama eğitiminin verilmeye başlanması gerektiğini düşündükleri seviyelerin gruplara göre dağılımları	85
Grafik 4-10 Deney Grubu 2 öğrencilerinin FOR Döngüsü konulu Mikroöğretim Videosunun hazırlanmasında tercih etmiş olduğu eğitsel araçlar (n= 101)	86
Grafik 4-11 Kontrol Grubu 2 öğrencilerinin FOR Döngüsü konulu Mikroöğretim Videosunun hazırlanmasında tercih etmiş olduğu eğitsel araçlar (n= 76)	87

KISALTMALAR VE SEMBOLLER

AIA:	MIT (A)pp (I)nventor for (A)ndroid
BÖTE:	(B)ilgisayar ve (Ö)ğretim (T)eknolojileri (E)ğitimi
DG1:	(D)eney (G)rubu (1)
KG1:	(K)ontrol (G)rubu (1)
GD:	(G)örev (D)eğeri
GUI:	(G)raphical (U)ser (I)nterface – (Grafik Kullanıcı Arabirimi)
İHD:	(İ)çsel (H)edef (D)üzenleme
MBTGPD:	(M)obil Akıllı cihazlar için uygulama geliştirmeye yönelik (B)lok (T)emelli (G)örsel (P)rogramlama (D)ili
MTPD:	(M)etin (T)emelli (P)rogramlama (D)ili
ÖY:	(Ö)z(y)eterlilik
PD1:	(P)rogramlama (D)illeri (1) Dersi
PD2:	(P)rogramlama (D)illeri (2) Dersi
XML:	E(x)tensible (M)arkup (L)anguage – Genişletilebilir İşaretleme Dili

BÖLÜM I: GİRİŞ

1.1. Problem

Günümüz eğitim dünyasında bilgisayar programlama eğitiminin, bilgisayar bilimleri alanında öğrenim görmeye başlayan öğrencilerin neredeyse tamamına zorlu bir engel teşkil ettiği iddia edilmiştir (Cornforth, 2014; Fujiwara, Fushida, Tamada, Igaki, & Yoshida, 2012; Lahtinen, Ala-Mutka, J., \#228, & rvinen, 2005; Proulx, 2000). Bu konuda görüş bildiren kimi araştırmacılar, her üç öğrenciden birinin kaldığını (Bennedsen & Caspersen, 2007) saptadıkları üniversite seviyesinde ilk programlama dersinin, konuları açısından öğrenilmesi en zor derslerden biri olduğunu savunmaktadır (Jenkins, 2002). Dolayısıyla, bilgisayar bilimleri üzerine eğitim veren üniversite bölümlerinin çoğunda öğrencilerin programlama ile ilk kez tanışacağı derslerin nasıl verileceği konusunda sorusuna cevap aranmaktadır (Allison, Orton, & Powell, 2002). Bu nedenle, hangi yaklaşım izlenecek olursa olsun, acemilere yönelik eğitimde atılacak her adımın son derece kritik olduğunu söylemek yanlış olmayacaktır.

Programlama eğitiminde daha büyük başarı elde edebilmek için, takip edilecek öğretim yaklaşımının yanı sıra, inceleme altına alınan ve başarıyı etkilediği düşünülen faktörlerden bir diğeri ise, eğitim süresince öğrencilere sunulacak programlama dilidir. Bu bağlamda, eğitimde kullanım için tercih edilecek programlama dilinin hangisi olması gerektiği sorusu, geçen zamana rağmen bilgisayar eğitimi literatüründe popülerliğini korumaktadır.

Programlama eğitiminde daha büyük başarı sağlayabilmek için görsel programlama dillerinin kullanılabileceği önerilmiş (Navarro-Prieto & Cañas, 2001); bilhassa geçtiğimiz yıllar içerisinde, acemilere yönelik eğitimde blok-temelli görsel programlama dillerinin uygun olacağı ileri sürülmüştür (John Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010). Zira blok-temelli görsel programlama dillerinin genel özelliği, metin tabanlı programlama dillerinin vazgeçilmez bir parçası olan, ancak dilden dile çeşitlilik gösteren sözdizimi öğelerinin yerini doğal dilde yazılmış ve sürükle-bırak yöntemiyle bir araya getirilen program bloklarının almasıdır. Sözdiziminin acemilere zaman zaman karmaşık gelebilen yapısı, böylelikle doğal dile veya en azından “yalancı koda” olabildiğince yakın hale getirilmiştir. Bu sayede,

öğrencilerin sözdizimi kurallarını ezberlemek zorunda kalmasının, sözdizimi yazımında hatalarla karşılaşmasının ve kod satırlarında boğulmasının önüne geçilmek istenmiştir (Klassen, 2006). Üstelik, yalnızca doğru kombinasyonlarla bir araya getirilebilecek şekilde düzenlenmiş blok parçalarından oluşan yap-boz şeklindeki düzenek, öğrencinin program yapısı ve onu oluşturan elemanların görevi hakkında daha somut bir fikir elde edinebilmesini sağlamaktadır. Ancak bu güne dek eğitimde kullanım için geliştirilen blok temelli görsel programlama dillerinin genellikle yaşça küçük öğrenciler için hazırlanmış olduğu (Kelleher, Pausch, & Kiesler, 2007; Utting, Cooper, Kölling, Maloney, & Resnick, 2010) dikkat çekmektedir.

Eğitimde kullanılacak programlama dili kadar, hangi yöntem, teknik ve araçlara başvurulacağı da bu alanda faaliyet gösteren araştırmacıların üzerinde durduğu konulardandır. Bu açıdan, programlama eğitimi için literatürde yer almış önerilerden biri de, yakın zamanda baş döndürücü bir hızla popülerlik kazanan taşınabilir akıllı cihazları programlama eğitiminde kullanmaktır (Burd ve diğ., 2012; Fenwick Jr, Kurtz, & Hollingsworth, 2011; Qusay H Mahmoud & Popowicz, 2010; Riley, 2012). Zira geçtiğimiz yüzyılın sonunda evlere giren; yepyeni ve büyümlü cazibesıyla insanları programlama öğrenme yönünde motive eden masaüstü kişisel bilgisayarın, günümüzde özellikle genç nüfus tarafından kanıksanmış olması muhtemeldir. Ancak, teknoloji sahnesine henüz çıkmış mobil akıllı cihazlara yönelik yükselen trendin, belki de geçmişteki bu büyümlü atmosferin tekrar yakalanmasında ve genç nüfusun programlama eğitime yönelik motivasyonunun artırılmasında kullanılması mümkün olabilir.

Mobil akıllı cihazlarda uygulama geliştirme amacıyla yaygın olarak kullanılan programlama dillerinin (Java, Objective C, Visual C#, vs.) metin-temelli diller olduğu ve nesne-yönelimli paradigmayı takip ettiği görülmektedir. Nesne-yönelimli programlama eğitiminin, acemi öğrencilerde bilişsel yükü arttırdığı ve eğitimi zorlaştırdığı (Wiedenbeck, Ramalingam, Sarasamma, & Corritore, 1999) düşünüldüğünde, acaba eğitime yönelik yüksek motivasyon vaat ettiği savunulan ancak günümüzde çoğunlukla nesne-yönelimli yöntemlerle programlanabilen mobil cihazların, başlangıç seviyesindeki programlama eğitiminde kullanımı da tartışmalı hale geliyor olabilir mi?

Bu noktada MIT App Inventor for Android'den bahsetmek yerinde olacaktır. App Inventor, Google firması tarafından programlama bilgisi olmayan son kullanıcıların Android işletim sistemine sahip mobil akıllı cihazlarda uygulama geliştirebilmesi amacıyla üretilmiş bir programlama dilidir. 2011 yılında ücretsiz olarak kamu erişimine sunulan App Inventor, görsel blok-temelli bir programlama dili olup, bu güne dek rastlanan diğer blok-temelli dillerden farkları; a) yetişkinler için de kullanıma uygun olması b) akıllı mobil cihazları hedef alması ve c) bu cihazlar üzerinde çalışan ve günlük hayatta gerçekten ihtiyaç duyulabilecek uygulamalar hazırlamaya olanak tanınmasıdır (Spertus, Chang, Gestwicki, & Wolber, 2010; Wolber, 2011b). Bu özellikleriyle App Inventor'ın programlama açısından eğitsel değere sahip olabileceği kısa sürede keşfedilmiştir, hatta Web-tabanlı bir uygulama olan App Inventor for Android projesi Google firması tarafından bir yüksek eğitim kurumu olan Massachusetts Teknoloji Enstitüsü Medya Laboratuvarına devredilmiştir. Yine, son birkaç yıl içinde bilgisayar eğitimi literatüründe App Inventor'ın programlama eğitiminde kullanımıyla ilgili çeşitli yayınlara rastlanır olmuştur (Gray, Abelson, Wolber, & Friend, 2012; Roy, 2012; Spertus ve diğ., 2010; Uludag, Karakus, & Turner, 2011; Wolber, 2011b). Bu yayınlarda App Inventor'ın; gerek görsel blok-temelli yapısı gerek ise mobil cihazlarla kullanım boyutuyla sağlayabileceği motivasyon sayesinde; yeni başlayanlara yönelik programlama eğitiminde kullanılabilir bir araç olduğundan bahsedilmiştir. Hatta Ahmad ve Gestwicki (Ahmad & Gestwicki, 2013), App Inventor'ın üniversite seviyesinde programlama eğitimi için gayet uygun olduğunu belirtirken, Karakuş ve arkadaşları (Karakus, Uludag, Guler, Turner, & Ugur, 2012) ise App Inventor'ın ileri seviye programlama derslerinde ve deneyimli öğrencilerle de kullanılabilirliğini iddia etmiştir. Ancak literatürde bu aracın programlama eğitiminde kullanımına ilişkin somut sonuçlar sunan, karşılaştırmalı ve deneysel çalışmalara dair eksiklik göze çarpmaktadır.

Dikkat edilmesi gereken bir diğer nokta ise, aslen blokların ardında nesne-yönelimli Java programlama diline ait kodları gizleyen ve mobil akıllı cihazlar için GUI-uygulamalar geliştirilmesini sağlayan App Inventor'ın da, nesne-yönelimli programlama paradigmasını takip eden, GUI özellikli uygulamalar geliştirmeye yönelik bir dil olduğudur. Dolayısıyla, App Inventor'ın programlamayla ilk kez tanışacak öğrencilerle kullanılması şeklinde verilen eğitimin nesne-yönelimli olacağı ve “GUI-öncelikli” (Yau,

2004) yaklaşımları izleyeceği sonucuna işaret etmektedir. Bu da, her ne kadar programlama bloklarının kullanımı ile sözdiziminin oluşturacağı bilişsel yük azaltılmış olsa da, “merhaba dünya”nın ötesine geçildiği andan itibaren acemilerin bir yandan programlamanın temel konseptleri ve algoritma mantığını öğrenirken, diğer yandan da nesneler, GUI-öğeler ve bunlara ilişkin özelliklerle çalışmayı öğrenmesi gerekliliği sonucunu doğuracaktır.

Bu noktada, App Inventor’ın öğrencilere programlama temellerini yordamsal paradigma uyarınca, metin tabanlı programlama dilleri ile konsol-uygulamalar geliştirme suretiyle öğretmeyi hedefleyen üniversite seviyesindeki “temel-yapı öncelikli” bir programlama müfredatında da kullanılıp kullanılamayacağı sorusu akıllara gelmektedir. Zira akıllı mobil cihazlarda uygulama geliştirmeyi kolaylaştıran blok temelli yapıya sahip App Inventor’ın programlama derslerinde bir eğitsel araç olarak; konsol uygulamalardan GUI uygulamalara geçiş basamağında kullanılması durumunda da fayda sağlaması beklenebilir. Bu sayede öğrencilerin konsol-uygulamalardan GUI-uygulamalara ve yordamsal programlamadan nesne-yönelimli programlamaya daha yumuşak bir geçiş yapabileceği düşünülmüştür. Zira sözdizimini bloklar ardında gizleyen sistem, halihazırda temel programlama kavramlarına aşina öğrencilerin nesne-yönelimli GUI-programlama kavramlarına bilişsel olarak daha kolay yoğunlaşmasına olanak tanıyabileceği, geliştirilen uygulamaların akıllı mobil cihazlara yönelik oluşu öğrencilerin motivasyonunda olumlu bir etki sağlayabileceği düşünülmektedir. Hatta bu sayede, ilk basamak eğitimde belki de sözdizimi mantığına alışmamak ya da konsol-temelli uygulamaları gerçek hayatla bağlantısız bulmak gibi gerekçelerle zorlanan veya derse karşı olumsuz bir tavır takınan öğrencilerin tekrar kazanılması mümkün olabilir. Diğer yandan, öğrencilerin birinci basamak eğitimde öğrendikleri temel programlama kavramlarını farklı bir platformda uygulayarak, programlamaya daha geniş bir perspektiften bakabilme kabiliyeti kazanabileceğine de inanılmaktadır.

Ancak; literatürde, özellikle üniversite seviyesi programlama eğitiminde kullanılan metin tabanlı dillerin endüstride istihdam sağlama kapasitesi ölçütünde değerlendirildiği rapor edilmişken; bir yandan da programlamada nispeten başarılı öğrencilerin blok-temelli dilleri “fazla basit ve kolay” bulduğuna da değinilmiştir (Kasurinen, Purmonen, & Nikula, 2008). Bunlar, deneyimli öğrencilerde blok temelli aracın kullanımına ilişkin motivasyonu olumsuz etkileyebilecek hususlar olarak düşünülebilir. Dolayısıyla; blok

temelli dillerle yapılan uygulamada başarı ve motivasyonunun; öğrencilerin metin tabanlı dilin kullanımında deneyimli olma yahut bu dilleri benimseme gibi faktörlerden etkilenip etkilenmediği, cevap aranması gereken bir soru olabilir. Ayrıca, öğrencilerin mobil cihaza yönelik blok temelli araçla çalışırken motivasyonlarına olumlu etkide bulunması beklenen bir diğer faktör olan “akıllı mobil cihaz aidiyetinin”, derste başarı ve motivasyon açısından herhangi bir etkisi olup olmadığının da deneysel olarak incelenmesinin de yararlı olacağı düşünülmektedir.

Bunların yanı sıra; yükselen mobil öğrenme trendiyle birlikte akıllı mobil cihazların eğitimde kullanımı giderek yaygınlaşmış ve bu platformlar üzerinde çalışan eğitsel yazılım uygulamalarına duyulan ihtiyaç da artmıştır. Ülkemizde 1998 yılında eğitim fakültelerinin yeniden yapılandırılması kapsamında açılmış olan BÖTE (Bilgisayar ve Öğretim Teknolojileri) bölümlerinin genel olarak iki ana amacı bulunduğu ve bunların (1) ilk ve orta öğretimde görev alacak olan, temel mesleki bilgi ve becerileri kazanmış bilgisayar öğretmenleri yetiştirmek ve (2) öğrenci kitlesinin pedagojik düzeyine ve müfredata uygun öğretim materyali ve eğitim yazılımı tasarlama, geliştirme, uygulama ve değerlendirme alanlarında donanımlı BDÖ (Bilgisayar Destekli Öğretim) uzmanları yetiştirmek olduğu unutulmamalıdır (Gerek & Kurt, 2010). Bu bilginin ışığında, BÖTE bölümü mezunlarının gerek mobil platformlarda çalışacak eğitsel yazılımları karmaşık programlama dillerini kullanmak zorunda kalmadan, daha kolay geliştirilebilmesi; gerekse K12 seviyesinde programlama eğitiminde kullanım için yeni bir araçla tanışabilmesi için, App Inventor ve benzeri görsel blok-programlama dillerinin BÖTE bölümü müfredatının bir noktasında öğrencilere sunulması faydalı olabilir. Ancak bu iddia uygulamalı çalışmalarla desteklenmelidir.

1.2. Amaç / Hipotezler

Bu çalışmada amaç, üniversite seviyesinde GUI-programlama eğitiminde eğitsel araç olarak mobil GUI uygulama geliştirmeye yönelik blok-temelli bir görsel programlama dili kullanmanın; masaüstü GUI uygulama geliştirmeye yönelik metin tabanlı bir programlama dili kullanmaya kıyasla öğrencilerin GUI programlama dersine yönelik akademik başarı ve motivasyonu üzerindeki etkisini incelemektir. Araştırma; öğrenci başarı ve motivasyonu üzerinde rol oynayacağı düşünülen, “yordamsal programlama

başarısı”, “üniversite öncesinde GUI programlama eğitimi almış olma” ve “akıllı mobil cihaz aidiyeti” faktörleri bağlamında yürütülmüştür.

Araştırmada Bir diğer amaç ise, çalışma grubunu BÖTE bölümü öğrencilerinin oluşturduğu düşünüldüğünde; geleceğin öğretim teknolojisi ve bilişim öğretmeni adayları olan; yani gelecekte hem mobil platformlar için eğitsel içeriği üretme hem de K12 seviyesinde programlama eğitimi verme görevini üstlenecek bu bireylerin akıllı mobil cihaza yönelik blok-temelli bir dili, metin tabanlı bir dile kıyasla mesleki yönden nasıl değerlendireceklerini incelemektir.

Araştırma kapsamında incelenmiş olan hipotezler ve araştırma sorularının listesi aşağıda verilmiştir. Tablo 1-1’de ise hipotezler ile bunları ilgilendiren bağımlı ve bağımsız değişkenlerin dağılımı gösterilmiştir.

1.2.1. Hipotezler

1. Akademik Başarı ile ilgili hipotezler

- 1.1.GUI-programlama eğitiminde akademik başarı açısından karşılaştırıldıklarında, eğitimi MBTGPD ile alan öğrenci grubu, eğitimi MTPD ile alan öğrenci grubuna göre daha başarılı olacaktır.
- 1.2.GUI-programlama eğitiminde MTPD yerine MBTGPD kullanmanın akademik başarıya etkisi üzerinde; metin temelli dille üniversite öncesinde GUI programlama eğitimi almış olma faktörü de rol oynamaktadır.
- 1.3.GUI-programlama eğitiminde MTPD yerine MBTGPD kullanmanın akademik başarıya etkisi üzerinde; üniversitede daha önce alınan yordamsal programlama eğitiminde başarı faktörü de rol oynamaktadır.
- 1.4.GUI programlama eğitiminde MTPD ve MBTGPD araçlarıyla elde edilen başarı seviyeleri üzerinde, öğrencilerin MTPD kullanımında deneyimli yahut acemi olması faktörü de rol oynamaktadır.
- 1.5.GUI programlama eğitiminde MBTGPD ve MTPD araçlarının kullanım sırası, öğrencilerin nihai akademik başarısı üzerinde etkiye sahip olacaktır.

- 1.6.GUI programlama eğitiminde MBTGPD ve MTPD araçların kullanım sırası, GUI programlama eğitimi almamış öğrencilerin nihai akademik başarısı üzerinde etkiye sahip olacaktır.
2. Motivasyona ile ilgili hipotezler
 - 2.1.GUI-programlama eğitiminde MTPD yerine MBTGPD kullanmak, öğrencilerin derse yönelik motivasyonu üzerinde olumlu etki yaratacaktır.
 - 2.2.GUI-programlama eğitiminde MTPD yerine MBTGPD kullanmanın derse yönelik motivasyona etkisi üzerinde; metin temelli dille üniversite öncesinde GUI-programlama eğitimi almış olma faktörü de rol oynamaktadır.
 - 2.3.GUI-programlama eğitiminde MTPD yerine MBTGPD kullanmanın dersedersine yönelik motivasyona etkisi üzerinde; üniversitede daha önce alınan yordamsal programlama eğitiminde başarı faktörü de rol oynamaktadır.
 - 2.4.GUI programlama eğitiminde MTPD ve MBTGPD araçlarının kullanıldığı gruplarda elde edilen motivasyon seviyeleri üzerinde, öğrencilerin MTPD kullanımında deneyimli yahut acemi olması faktörü de rol oynamaktadır.
 - 2.5.GUI programlama eğitiminde; deneyimli öğrencilerde kullanılan aracın çapraz uygulama sonucu değiştirilmesi, motivasyonda değişime neden olacaktır.
 - 2.6.GUI programlama eğitiminde; acemi öğrencilerde kullanılan aracın çapraz uygulama sonucu değiştirilmesi, motivasyonda değişime neden olacaktır.
3. Mobil akıllı cihaz aidiyetinin motivasyon ve başarıya etkisi ile ilgili hipotezler
 - 3.1.MBTGPD ile öğrenim gören öğrencilerde, kendi mobil cihazlarına sahip olan öğrenci grubu ve derse cihaz emülatörü kullanarak çalışan öğrenci grubunun derse yönelik görev değeri algısında fark olacaktır.
 - 3.2.MBTGPD ile öğrenim gören öğrencilerde, kendi mobil cihazlarına sahip olan öğrenci grubu ve derse cihaz emülatörü kullanarak çalışan öğrenci grubu arasında GUI programlama akademik başarısı açısından fark olacaktır.

Tablo 1.1 Hipotezlerin ilişkin bağımlı ve bağımsız değişkenlerine göre dağılımı

Bağımlı Değişken Bağımsız Değişken	Akademik Başarı	Motivasyon	Hipotez
A	H _{1.1}	H _{2.1}	
Faktörler A x G	H _{1.2}	H _{2.2}	
Faktörler A x P	H _{1.3}	H _{2.3}	
Faktörler A x D	H _{1.4}	H _{2.4}	
S (D ₁)	-	H _{2.5}	
S (D ₀)	-	H _{2.6}	
S	H _{1.5}	-	
S (G ₀)	H _{1.6}	-	
M	H _{3.2}	H _{3.1}	

A: Yordamsal programlama eğitimini takip eden GUI programlama eğitiminde kullanılan araç
(A₀ = MTPD, A₁= MBTGPD)

G: Üniversite Öncesinde GUI Programlama Eğitimi
(G₀ = Eğitim Almamış, G₁= Eğitim Almış)

P: Üniversitede Daha Önce Alınan Programlama Dilleri 1 Dersinde Başarı Faktörü
(P₀ = Başarısız, P₁= Başarılı)

D: MTPD kullanımı konusunda deneyim (D₁ = G₁ ∩ P₁ ; D₀ = G₀ ∩ P₀)

S: Yordamsal programlama eğitimini takip eden GUI programlama eğitiminde araçların kullanım sırası
(S₀ = önce MTPD sonra MBTGPD, S₁= önce MBTGPD sonra MTPD)

M: Akıllı mobil cihaz aidiyeti
(M₀= Akıllı Mobil Cihaz Sahibi Değil, M₁= Akıllı Mobil Cihaz Sahibi)

1.2.2. Araştırma Soruları

Araştırma kapsamında aşağıdaki araştırma sorularına cevap aranacaktır.

1. GUI Programlama dersi almış öğrencilerin, söz konusu eğitimi almış oldukları sönestrın sonundan itibaren geen 1 yıllık süre zarfında kişisel amaçlarla geliřtirmiş oldukları masaüstü ve mobil uygulama sayısı; ders kapsamında kullanılmış araçlara göre nasıl deęiřecektir?
2. alıřmaya katılan biliřim öęretmeni adaylarının, acemi bir hedef kitleye programlama eğitimi başarıyla verebileceklerine dair inanları ders kapsamında kullanılmış araçlara göre nasıl deęiřecektir?
3. alıřmaya katılan biliřim öęretmeni adaylarının, yeni bir programlama dilini kolayca öęrenebileceklerine dair inanları ders kapsamında kullanılmış araçlara göre nasıl deęiřecektir?
4. alıřmaya katılan biliřim öęretmeni adaylarının, ortaokul öęrencilerine GUI programlama eğitimi verilmesine iliřkin görüřleri ders kapsamında kullanılmış araçlara göre nasıl deęiřecektir?
5. alıřmaya katılan biliřim öęretmeni adaylarının, programlama eğitiminin hangi yařtan/seviyeden itibaren verilmesinin uygun olduęunu düřünce si ders kapsamında kullanılmış araçlara göre nasıl deęiřecektir?
6. Programlama eğitimi ni MBTGPĐ kullanarak ve yalnızca MTPĐ kullanarak eğitim almış öęretmen adaylarından oluřan iki gruptan, ortaokul öęrencilerine yönelik programlama eğitiminde seilen örnek bir konu için 20 dakikadan oluřan bir ders planı hazırlayarak mikroöęretim uygulaması yapmaları istendięinde öęretmen adayları hangi araçları tercih edeceklerdir?

1.3. Önem

Biliřim dünyasında masaüstünden yaygın biliřime geiř sürecinde olduęumuz bu kritik dönemde, aęın gerisinde kalmamak ve kullanımı gitgide artan mobil akıllı cihazlara yönelik yeni bir programlama eğitimi vizyonu geliřtirebilmek adına bu cihazlar özelinde yürütölmüş arařtırmalara ve uygulamalara ihtiya duyulmaktadır. Bu arařtırma, mobil cihazlara yönelik programlama eğitimi uygulamaları için pilot

niteliğinde olması açısından önemlidir. Diğer yandan bu araştırmanın; literatürde mobil-uygulama geliştirme odaklı görsel blok programlama aracının, lisans seviyesinde programlama eğitiminde motivasyon ve başarıya etkisini inceleyen kapsamlı deneysel araştırmalara yönelik ihtiyacı gidermeye katkıda bulunacağı umulmaktadır. Çalışma grubunun K12 seviyesinde bilişim öğretmeni adaylarından oluşuyor olması ise, aracın kullanışlı bulunması durumunda toplum tabanına yayılabilmesi olanağı tanınması açısından önemlidir. Zira araştırmaya konu edilmiş tipte uygulamalarla, yani bilişim öğretmenlerinin bu tip araçlarla tanıştırılması yoluyla; geleceğin teknolojisi olan mobil akıllı cihazlara yönelik yazılımların geliştirilmesi süreci üniversitelerden ilk ve orta öğretim seviyesine inebilir.

1.4. Sınırlılıklar

- Deney için tasarlanmış çapraz uygulamada eğitim içeriğinin kullanılan programlama dili (ve buna ilişkin IDE) haricinde birbirine tamamen denk kalabilmesi için hem masaüstü hem mobil platformlarda geliştirilen uygulamaların aynı fonksiyona ve yapıya sahip olmasına dikkat edilmiştir. Örneğin, “hesap makinesi” adlı uygulama kontrol grubu öğrencileri tarafından metin tabanlı programlama diliyle masaüstü platforma yönelik olarak geliştirilirken; deney grubu öğrencileri aynı fonksiyona sahip ve aynı isimli uygulamayı görsel blok temelli programlama diliyle, mobil platforma yönelik olarak geliştirmiştir. Dolayısıyla, müfredat çerçevesinde geliştirilen uygulamalarda mobil bilişimin doğasında bulunan ve masaüstü bilişim vizyonunda takip edilmesi ya görece daha zor olan ya da ilave çevresel donanım gerektiren “taşınabilirlik”, “mekansal farkındalık” ve “çevresel algılama” gibi konseptlere yer verilmemiştir. Bu durum, bu uygulamada mobil cihazların rolünün “cebe sığacak kadar küçük ve dokunmatik ekranlı masaüstü bilgisayarlar” seviyesinde kalmasına neden olmuştur ve mobil cihaz kullanımının öğrenci motivasyonu üzerindeki potansiyelini kısıtlayıcı bir etkiye yol açmış olabilir.

- Bu arařtırmada MBTGPD hakkında varılan yargı ve sonuçların kapsamı, MIT App Inventor Beta dili ve IDE’si; MTPD hakkında varılan yargı ve sonuçlarını kapsamı Visual C# ve ilişkin Visual Studio IDE’si ile sınırlıdır.

1.5. Sayıtlar

- Deney için tasarlanmış apraz uygulamada eğitim ieriğinin kullanılan araç haricinde birbirine tamamen denk kalabilmesi için hem masaüstü hem mobil platformlarda geliştirilen uygulamaların aynı fonksiyona sahip olmasına dikkat edilmiştir. Bu bakış açısıyla geliştirilen öğretim programında, deney ve kontrol gruplarında her hafta işlenen derslerin, hedef platform ve kullanılan araçtan bağımsız olarak birbirleriyle tamamen aynı öğrenme çıktılarına sahip olduğu varsayılmıştır.
- Uygulama Sınavı 1 ve Uygulama Sınavı 2 isimli ölçme araçlarında bulunan soruların; sınavın MBTGPD yahut MTPD kullanılarak özölmesi gibi iki durumda da, sınav geçerliliği üzerinde etki yaratmayacağı varsayılmıştır.
- “Programlama Dilleri 1 dersi başarısı” faktörü oluşturulurken, deney gruplarında bulunan öğrencilerin Ön Bilgi Testi 1 ölçümü puanları baz alınmıştır. Öğrencileri bu puanlara göre başarılı/başarısız şeklinde ayırmak için; her iki grup arasında 0,5 standart sapmalı bir aralık bırakılmasının yeterli olduğu varsayılmıştır.

1.6. Tanımlar

1.6.1. Konsol-Uygulama

Konsol uygulaması, yalnızca metinlerden oluşmuş bir arayüze sahip olan yazılım programlarına verilen addır. Bunlara örnek olarak yalnızca metin görüntüleyen terminaller yahut komut istemi arayüzlerinde çalışan programlar verilebilir.

1.6.2. Grafik Kullanıcı Arayüzü (GUI) ve GUI-Programlama

Grafik Kullanıcı Arayüzü, kullanıcıların elektronik cihazlarla etkileşimlerinde grafiksel ikonların ve görsel belirteçlerin, metinlerin yanı sıra ikincil notasyon olarak kullanıldığı arayüzlere verilen isimdir. GUI-Programlama (yahut GUI-uygulama geliştirme) ise, GUI özelliklerine sahip yazılım uygulaması geliştirme pratiklerine verilen addır.

1.6.3. Görsel Programlama Dili

Görsel programlama dilleri, bilgisayar programı geliştirmek için oluşturulan kaynak kodunun, belli bir sözdizimi uyarınca oluşturulan metin girdilerinden ziyade; bloklar, düğümler, akış diyagramları gibi görsel nesnelerin çoğunlukla “sürükle-bırak” şeklinde bir araya getirilerek oluşturulduğu dillerdir.

1.6.4. Metin tabanlı Programlama Dili

Kaynak kodunun yalnızca metin komutları ve buna ilişkin sözdizimi uyarınca oluşturulduğu programlama dilidir.

1.6.5. Microsoft Visual C# (MTPD)

C#, Microsoft tarafından .NET projesi kapsamında geliştirilmiş çok-paradigmalı ve sade, modern, çok amaçlı ve aslen nesne-yönelimli ve metin tabanlı bir programlama dilidir. Çalışmada MTPD şeklinde kastedilen eğitsel araç, bu dil ve birlikte kullanıldığı IDE’si olan Microsoft Visual Studio’dur.

1.6.6. Mobil Akıllı Cihaz

Mobil akıllı cihaz; GSM, GPRS, WiFi, 3G gibi çok çeşitli protokoller kullanarak mobil bilişime uyacak şekilde sürekli olarak ağ bağlantısı ve iletişim sunmayı amaçlayan, pil ömrü, fiziksel boyut ve ağırlık gibi taşınabilirliğe özgü kriterler ve bunların gerektirdiği girdi yöntemleri göz önünde bulundurularak tasarlanmış, yaygın bilişim paradigması uyarınca entegre olarak bulundurduğu çeşitli algılayıcılar sayesinde ortam farkındalığı kazandırılmış kişisel bilgisayar ve iletişim sistemleridir. Örnek olarak akıllı telefonlar ve tabletler gösterilebilir.

1.6.7. Android İşletim Sistemi

Dokunmatik ekran girdisine sahip ve tablet yahut akıllı telefon gibi küçük-orta form faktörüne göre tasarlanmış mobil akıllı cihazlarda kullanım için Google firması tarafından geliştirilmiş olup, bu alanda günümüzde dünyanın en çok kullanım gören işletim sistemidir (Mansfield-Devine, 2012).

1.6.8. MIT App Inventor (MBTGPD)

Akıllı mobil araçlar için uygulama geliştirme amacına sahip ve görsel blok programlama mantığıyla çalışan bir programlama dili ve buna ilişkin GUI tasarım özellikli IDE'dir. İşbu çalışmada mobil cihazlarda uygulama geliştirmeye yönelik blok-temelli görsel programlama dili (MBTGPD) olarak tabir edilen eğitsel araç, MIT App Inventor programlama dili ve öğrencilerin bu dilin eğitimini almak için kullandıkları web-tabanlı ve aynı isimli IDE'ye tekabül etmektedir.

1.6.9. Programlama Dilleri 1 Dersi (PD1)

Bu çalışmada Programlama Dilleri 1 dersi ile kastedilen; üniversite seviyesinde öğrencilerin aldıkları ve bir sömestr boyunca süren programlama dilleri dersi olup, söz konusu derste yordamsal programlama kavramları, metin tabanlı ve nesne-yönelimli bir programlama dili olan C# kullanılarak öğretilmiştir.

1.6.10. Deneyimli Öğrenciler

Çalışmada deney gruplarının “Deneyimli Öğrenciler” tabir edilen alt grupları; üniversite öncesinde metin tabanlı bir programlama dili ile GUI-programlama eğitimi almış olan; bu eğitimin çıktılarına ölçen bir testte (Ön Bilgi Testi 2) ortalamanın üzerinde not almış; ve üniversitede almış oldukları Programlama Dilleri 1 dersi çıktılarına ölçen bir diğer testte (Ön Bilgi Testi 1) ise notları ortalamanın en az 0,25 standart sapma üzerinde kalan öğrencilere tekabül etmektedir. Dolayısıyla bu grupta yer alan öğrenciler, GUI uygulama geliştirme açısından üniversite öncesi deneyimi bulunan ve metin tabanlı dille alınan yordamsal programlama eğitiminde başarılı bulunmuş öğrencilerdir.

1.6.11.Acemi Öğrenciler

Çalışmada deney gruplarının “Acemi Öğrenciler” tabir edilen alt grupları; üniversite öncesinde metin tabanlı bir programlama dili ile GUI-programlama eğitimi almamış olan ve üniversitede almış oldukları Programlama Dilleri 1 dersi çıktılarını ölçen bir testte (Ön Bilgi Testi 1) notları ortalamanın en az 0,25 standart sapma kadar altında kalan öğrencilere tekabül etmektedir. Dolayısıyla bu grupta yer alan öğrenciler, GUI uygulama geliştirme açısından deneyimi bulunmayan ve metin tabanlı dille alınan yordamsal programlama eğitiminde başarısız bulunmuş öğrencilerdir.

BÖLÜM II: İLGİLİ ARAŞTIRMALAR / ALANYAZIN

2.1 Üniversite Düzeyinde Programlama Eğitimi

Kişisel bilgisayarın insan hayatındaki önemi gün geçtikçe artmaktadır ve iş dünyası, eğitim, haberleşme, eğlence gibi çok çeşitli ihtiyaçları karşılayan bu cihazların giderek vazgeçilmez bir konum aldığını söylemek yanlış olmayacaktır. Kişisel bilgisayarları, yarı-iletken devre elemanlarından oluşmuş birer parçalar yığını olmaktan kurtarıp, bireyin ihtiyaçlarını yerine getirecek fonksiyonlar yüklü sistemlere dönüştürmek ise, yazılım geliştiricilerin görevidir.

Profesyonel yazılım geliştiricilerin bilişim endüstrisine katılması genellikle yoğun efor sarf edilmesini gerektiren zorlu bir eğitim sürecinin ardından mümkün olabilmektedir. Bilgisayar programlama eğitiminin zorluğu da dikkate alındığında, profesyonel yazılım geliştiricilerin kolay yetişmediği ve sayılarının toplumun ihtiyaçlarını karşılamaya yetecek kadar yüksek olmadığı söylenebilir. Zira Programlama öğretimi bütün dünyada yaygın olarak verilmekle birlikte öğrencilerin programlamaya giriş dersindeki akademik başarılarının istenilen seviyede olmadığı da bilinen bir gerçektir. Bu konuda çeşitli araştırmacılar görüş belirtmişlerdir. Örneğin, Proulx'a göre başlangıç seviyesindeki programlama eğitimi, bilgisayar bilimleri alanında öğrenim görmeye başlayan öğrencilerin tamamının tökezlemesine yol açan etkenlerden biridir (Proulx, 2000). Jenkins, programlamanın öğrenilmesi en zor ders konularından biri olduğunu savunurken (Jenkins, 2002); Allison ve arkadaşları ise bilgisayar bilimleri üzerine eğitim veren üniversite bölümlerinin çoğunun öğrencilerin programlama ile ilk kez tanışacağı derslerin nasıl verileceği konusunda sıkıntılar yaşamakta olduğunu iddia etmektedir (Allison ve diğ., 2002). Programlama eğitiminin zorlukları öğrencilerde motivasyon sorunlarına yol açabilmektedir. Motivasyonunu yitirmiş bu öğrencilerin öğrenmeyi gerçekleştiremeyeceği ve en nihayet başarısız olabileceği belirtilmiştir (Jenkins, 2002). Örneğin, Meisalo ve arkadaşları çalışmalarında uzaktan eğitim öğrencilerinin %32'sinin henüz eğitimin birinci yılında bilgisayar dersini devamsızlık yoluyla bıraktığını tespit etmiştir ve buna sebep olarak öğrencilerin dersteki uygulamalar veya teorik yönlerin çok zor bulmasını göstermiştir (Meisalo, Suhonen, Torvinen, & Sutinen, 2002). Kinnunen ve Malmi ise, dünya genelindeki üniversitelerde programlamaya giriş dersine kayıt olan öğrencilerin %20 - %40 oranlarında bilişime

yönelik dersleri veya bölümü bıraktığını ifade etmektedirler (Kinnunen & Malmi, 2008). McCracken ve arkadaşlarının yürüttükleri çok uluslu ve çok kuruluşlu bir çalışmanın sonucundaysa öğrencilerin programlamaya giriş dersindeki genel başarısının son derece düşük olduğu bulunmuştur (McCracken ve diğ., 2001).

Dünya genelinde programlama eğitimindeki akademik başarının istenilen seviyenin altında olması, üniversite seviyesinde programlama eğitiminin kendine özgü zorlukları olduğunu göstermektedir. Programlama eğitiminde başarılı olmak için öğrencinin iyi bir matematik temeline sahip olması gerektiği, ancak bunun yanı sıra problem çözme becerisi, genel bilgisayar kullanım becerisi ve algoritmik düşünme becerisine sahip olması gerektiği belirtilmiştir (Burton & Bruhn, 2003; Hadjerrouit, 1998). Robins, Rountree ve Rountree ise programlama eğitiminin zorluğunun, programlama öğreniminin karmaşık yeni bilgiler öğrenme ve bu bilgilerle ilgili stratejiler geliştirme ve pratik program yazma becerileri kazanmayı gerektirmesinden kaynaklandığını ifade etmektedirler (Robins, Rountree, & Rountree, 2003). Zira bu araştırmacılara göre bu tip becerilerin elde edilmesi için öğrencilerin üst seviyede bilişsel kavrama kapasitesine sahip olmaları ve öğretmenlerin de uygun öğretim stratejileri kullanmaları gerekmektedir.

2.2 Üniversite Öncesinde Programlama Eğitimi

Geleneksel olarak üniversite seviyesinde bireyleri hedef alagelmiş olan programlama eğitimi günümüzde giderek daha erken yaşlarda verilir hale gelmiştir. Ancak programlama eğitiminin, okuma-yazma eğitimi gibi düzgün dağılımlı ve neredeyse dünyanın her yerinde hemen hemen aynı yaşlarda verilen bir eğitim olmadığı da muhakkaktır. Bu durum, üniversite seviyesine gelindiğinde programlama eğitimi açısından öğrencilerin kiminin deneyimli, kimininse tamamen deneyimsiz olduğu durumlar oluşmasına sebep olmaktadır. Programlama eğitiminin, programlama becerisi açısından geniş bir skalada bulunan öğrencilere verilmesi ise eğitim açısından sorunlara yol açmaktadır (Davis, Carr, Cooke, & White, 2001; Jenkins & Davy, 2002; Röbling & Freisleben, 2000). Bu durumda her bir öğrenciye ayrı ayrı vakit ayrılması gerekliliği ortaya çıkabilmektedir ancak bu tip uygulamalar için zaman ve kaynak yaratmak her zaman için mümkün olamamaktadır. Programlama hususunda tecrübeli ve tecrübesiz öğrencilerin aynı ortamda eğitim gördüğü senaryolar giderek artmaktadır (Hagan &

Markham, 2000) ancak tecrübeli ve tecrübesiz öğrenciler arasında beceri yönünden görülen derin uçurum; her iki öğrenci tipi için de farklı yönden problemlere neden olmaktadır: deneyimli öğrenciler dersin çok kolay olduğu yanılgısına kapılıp dersi boşlama eğilimi gösterebilecekken; deneyimsiz öğrenciler ise kendilerini deneyimli öğrencilere kıyasla yetersiz hissedip motivasyon sorunu yaşayabilecektir. Bu tip sorunların üzerinden gelebilmek için Sanders, Mueller ve Conrad, programlama dersini vermeden önce deneyimsiz öğrencileri destekleyebilmek ve deneyimlilerin seviyesine taşıyabilmek için ilaveten “bilgisayara giriş” niteliğinde bir ders verilmesini önermiştir (Sanders & Mueller, 2000). Benzer şekilde, Shackelford ve LeBlanc da kendi çalıştıkları kurumda öğrencilere “programlamaya giriş” dersinden önce “bilgisayara giriş” ismini taşıyan ancak öğrencileri bilgisayarın algoritmik işleyişini öğrenmeye hazırlayan 10-haftalık bir öğretim programı sunmuşlardır (Shackelford & LeBlanc Jr, 1997).

2.3. Programlama Paradigmaları

Programlama paradigması, genel anlamda bilgisayar programlarının, tüm altyapı ve elemanlarıyla birlikte oluşturulması esnasında izlenen temel tasarım yaklaşımına verilen isimdir (Van Roy, 2009). Çeşitli programlama dillerinin sahip olduğu özellikler ve kullanım alanları; bu dillerin desteklediği paradigmalar sayesinde tanımlanır. Ancak özellikle günümüzde bazı programlama dillerinin birden fazla paradigmaya destek sağladığı da görülmektedir.

2.3.1. Yordamsal Programlama

Bilgisayar programının geliştirilmesi esnasında kod kalitesinin ve verimliliğinin artırılarak, hataların ve geliştirme için geçen sürenin azaltılmasını amacıyla prosedürler, metotlar, ve bunlara ait alt rutinlerden oluşan blok yapıları ile *for* ve *while* gibi döngülerin sıkça kullanılmasını savunan bir paradigmadır (B. Stroustrup, 1988; Van Roy, 2009). Bu yaklaşım, her satırda belli başlı temel sınamaların gerçekleştirildiği ve satırdan satıra *goto* komutuyla sıkça atlama yapılmasını gerektiren; bu şekilde *spagetti kod* tabir edilen karmaşık ve hatalara gebe programlar kodlarının oluşmasına neden olan yaklaşımlara cevap olarak gelişmiştir.

Modern programlama eğitiminde yordamsal programlama paradigması genellikle, programlama eğitiminde algoritma geliştirme ve kodda hata bulma gibi temel kavramların öğrenciye kazandırılması amacıyla kullanılmaktadır.

2.3.2. Nesne-Yönelimli Programlama

Nesne-yönelimli Programlama, 1960'lardan itibaren geliştirilmiş olan bir bilgisayar programlama paradigmasıdır. Yazılım sistemlerinin karmaşıklığı arttıkça; yazılımlara ait yaşamsal döngüde rastlanan tasarım, yürütme ve bakım gibi çeşitli hususlarda gözlemlenen kimi kısıtlama ve problemlerin üstesinden gelinebilmesi için sürekli olarak bu gibi yeni yazılım geliştirme metodolojileri üretilmektedir. Bunlardan biri olan Nesne-Yönelimli Programlama, programa ilişkin durum ve işlemleri "nesne" tabir edilen yapılara sabitleyen (enkapsülasyon) bir yaklaşımdır (Ambler, Burnett, & Zimmerman, 1992; B. Stroustrup, 1988; Van Roy, Armstrong, Flatt, & Magnusson, 2003). Bu yaklaşım sayesinde üstesinden gelinmesi hedeflenen problemler şunlardır:

- **Karmaşıklık:** Günümüz yazılım mimarileri, genellikle paralel ve eşzamanlı çalışma yapıları bulunduran son derece karmaşık yapılara evrilmiştir. Bu yapıların bünyesinde bulunan soyut elemanların; "sınıflar" ve "nesneler" şeklinde kategorilere ayrılarak somutlaştırılması sağlanmaktadır.
- **Tasarım kolaylığı:** Bilgisayarlara ait görüntü işleme kapasitesi yıllar içerisinde arttıkça; yazılım uygulamalarında bu gelişmelerden faydalanacak şekilde daha zengin görsel öğeler barındıran grafik arayüzler tercih edilmeye başlanmıştır. İnsan-bilgisayar etkileşiminde kullanışlılığı arttıran grafik kullanıcı arayüzlerine sahip bu uygulamaların geliştirilmesinde nesne-yönelimli yaklaşımın son derece uygun olduğu tespit edilmiştir. Zira bu tip bir yazılım uygulamasının tasarımında programcının kullandığı menü, pencere, ikon ve düğme gibi tüm bileşenler aslen birer programlama nesnesi olarak ele alınmaktadır.
- **Yeniden kullanılabilirlik:** Nesne-yönelimli programlama yaklaşımı, yazılımı oluşturan bileşenlerin bir kez üretilerek hazır hale getirildikten sonra tekrar tekrar kullanılmasını öngörmektedir. Hazır bileşenlerin yanı sıra, kalıtım özelliği sayesinde de kullanıcının bir programlama sınıfından, benzer özelliklere sahip alt sınıfları kolaylıkla türetmesi kolaylaşmıştır. Bu sayede herkesçe yaygın

olarak kullanılan kimi bileşenlerin tekrar tasarlanması yahut yeni sınıfların sıfırdan geliştirilmesi gerekliliği ortadan kalkmakta; zamandan tasarruf edilmektedir.

Bunların yanı sıra, verilere erişime ilişkin sorunlara yönelik geliştirilmiş veri soyutlama (data abstraction) kavramı ve aynı isme sahip fonksiyonların ve metotların farklı görevler yerine getirebilmesini sağlayan çok biçimlilik (polymorphism) özellikleri de yazılım projelerinde kolaylık sağlayan nesne-yönelimli programlama özelliklerindendir.

En nihayet, nesne-yönelimli programlama yaklaşımında değişkenlerin genel veya özel olarak tanımlanabilmesi imkanı, zaman zaman milyonlarca satır koda ulaşan günümüzün karmaşık yazılım projeleri üzerinde birden çok yazılım geliştiricisinin çok daha rahat çalışabilmesini sağlamıştır.

Nesne yönelimli programlama günümüzde dünyanın en çok rağbet gören programlama paradigması olup; endüstriyel anlamda en fazla kullanılan programlama dilleri bu yapıda çalışacak şekilde geliştirilmiştir.

2.3.3. Kaynak kod ve Sözdizimi

Kaynak kodu, toplamda bir bilgisayar programını oluşturacak talimatları bilgisayara verebilmek için insanlar tarafından belli bir programlama diline uygun olarak yazılan metinlerdir (Van Roy, 2009). Kaynak kodu bilgisayar tarafından alınıp “derleyici” denilen arayüzden geçirilmek suretiyle bilgisayarın anlayabileceği düşük seviyeli makine koduna dönüştürülür. Sözdizimi ise, belirli bir programlama dilinde kullanılabilecek metin komutları ve sembollerinin ne şekilde kullanıldığında anlamlı hale geleceğini belirleyen kurallar bütünüdür. Programlama dilinde yazılmış kaynak kodunun o dile ait derleyici tarafından makine koduna dönüştürülmesi işleminde, öncelikle sözdizimi kurallarının sağlanıp sağlanmadığı kontrol edilir ve ancak bundan sonra kod semantik anlamda ele alınmaya başlanır. Sözdizimi, belli bir programlama diline ait “dilbilgisi” kuralları olarak düşünülebilir ve bilgisayarın programa ait kaynak kodunu anlayabilmesi için kodda herhangi bir sözdizimi hatası bulunmaması gerekmektedir. Bu durum, özellikle başlangıç seviyesinde programlama eğitiminde sorunlara yol açmaktadır zira dil kullanımının kusursuz biçimde gerçekleşmesi gereği acemi öğrenciler seviyesinde zorluk yaşanmasına neden olmaktadır.

2.3.4. Konsol-Uygulamalar

Konsol uygulaması, yalnızca metinlerden oluşmuş bir arayüze sahip olan programlama verilen addır (Liberty, 2005). Bunlara örnek olarak yalnızca metin görüntüleyen terminaller yahut komut istemi arayüzlerinde çalışan programlar verilebilir, ve tabi ki söz konusu terminaller yahut komut istemlerinin kendileri de bizzat birer konsol uygulamasıdır. Bu uygulamalarda ekranda metin dışında herhangi bir görsel öge bulunmazken, kullanıcıların programla etkileşimi genellikle yalnızca klavye girdi birimi üzerinden yollanan metin girdileri aracılığıyla gerçekleşir. Grafik kullanıcı arayüzlerine sahip uygulamaların gitgide daha kolay geliştirilebilir hale gelmesi ile son kullanıcıya yönelik konsol uygulamaların günümüzde büyük ölçüde kullanımdan kalktığı söylenebilir. Ancak halen profesyonel amaçlarla bu tip uygulamaları kullanan azımsanamayacak sayıda sektör ve kuruluş bulunduğu gibi, GUI öğelerinin doğurabileceği ilave bilişsel yükü ortadan kaldıran konsol uygulama uygulama geliştirme pratiklerinin acemilere yönelik programlama eğitiminde kullanımı da halen yaygındır.

2.3.5. Programlamada Grafik Kullanıcı Arayüzleri

Grafik Kullanıcı Arayüzü, kullanıcıların elektronik cihazlarla etkileşimlerinde grafiksel ikonların ve görsel belirteçlerin, metinlerin yanı sıra ikincil notasyon olarak kullanıldığı arayüzlere verilen isimdir (Liberty, 2005). Bu arayüzler, konsol uygulamaların son kullanıcılar tarafından işletilmesinin zor bulunması ve bilgisayarlarda zamanla gelişen grafiksel gösterim özellikleri ile günümüzde son derece yaygınlaşmıştır. Grafik Kullanıcı Arayüzleri'nin (GUI) yaygınlaşması ile, programlama dilleri ve entegre geliştirme ortamları da bu tip arayüze sahip programların daha kolay geliştirilmesini sağlayacak şekilde evrimleşmiştir. Bilhassa, GUI elemanlarının görsel “nesneler” olarak algılanmasına olanak tanıyan nesne-yönelimli programlama dilleri de bu GUI-tasarım odaklı programlama (kısaca, GUI programlama veya GUI uygulama geliştirme) süreçleri için uygun zemini oluşturmuştur. GUI programlama sayesinde, program arayüzüne ilişkin kodların kaynak kodundan ayrımlanması sağlanmıştır. Programlama dilinde GUI öğelerinin çalışmasını kontrol eden kısımlar genellikle olay güdümlü mimariye sahiptir.

2.3 Acemilere Yönelik Programlama Eğitimi - Doğru Programlama Dilinin Seçimi

Acemilere yönelik programlama eğitiminde yaşanan zorlukların en aza indirilebilmesi için; uygun bir programlama dilinin seçilmesi çok önemlidir. Bu dilin seçiminde; giriş seviyesinde programlama eğitiminin; öğrencilere problem çözme, algoritmik düşünme ve yordamsal programlama mantığı gibi genel programlama prensipleri doğrultusunda beceriler kazandırmayı amaçladığı gerçeği kadar (Hadjerrouit, 1998), kullanılacak dilin pedagojik uygunluk hususu da kesinlikle göz önüne alınmalıdır (Jenkins, 2002). Zira her ne kadar programlama derslerinde temel amaç öğrencilere bir “dili” öğretmekten ziyade “programlamayı” öğretmek olsa da öğrenciler, ders boyunca genellikle tek bir dil öğreniyor oldukları için öğrendikleri hususların yalnızca belli bir dile özgü olduğunu düşünme eğilimindedir. Bu nedenle Jenkins (Jenkins, 2002), öğrencilere sunulacak dilin pedagojik unsurlar göz önüne alınarak ve yalnızca programlama eğitiminde kullanım amacıyla geliştirilmiş olan; profesyonel ve ticari değer taşıyan daha karmaşık dillere geçiş kolaylığı tanıyacak nispeten daha kolay bir dille verilmesi gerektiğini savunmaktadır. Ancak aynı araştırmacı göre, öğretilecek dilin öğrenciler tarafından benimsenmesinde pedagojik hususların yanı sıra, söz konusu dilin ticari anlamda kazanç getirme potansiyelinin de rol oynayabileceğini belirtmiştir. Jenkins her ne kadar eğitim amacıyla seçilecek dilin “endüstride popülerlik” hususu göz önüne alınmadan ve yalnızca eğitsel amaçlarla seçilmesi gerektiğini iddia ediyor olsa da; bu durum günümüzde çoğu öğrencinin ticari açıdan potansiyel vaat eden ve birçok meslek alanında kullanım gören ancak pedagojik yaklaşımlar göz önüne alınmadan üretilmiş popüler programlama dillerini öğrenmek konusunda daha istekli olabileceği gerçeğinin de habercisidir (Vilner, Zur, & Gal-Ezer, 2007). Başka bir çalışmasında öğrencilerin, üniversite sonrası kariyerlerinde elde edilebilecek fırsatları göz önüne aldığı ve bu anlamda kazanç elde etme vaadi bulunmayan programlama dilleri üzerinde çalışmak için efor sarfetmediğinden bahseden Jenkins, programlama eğitiminde öğrencilere motivasyon sağlamak için acemilere yönelik eğitimde seçilecek dilin pedagojik hususlara ek olarak ticari popülerliğe de sahip olması gerektiğini de belirtmiş; bu açılardan gereksinimleri karşıladığını düşündüğü nesne-yönelimli bir programlama dili olan Python’un kullanımını önermiştir (Jenkins, 2004). Fekete, Kay ve diğerleri (Fekete, Kay, Kingston, & Wimalaratne, 2000) ise, üniversite seviyesinde programlama

eğitimi verilebilmesi için, yine bilhassa acemilere yönelik pedagojik hususlar göz önüne alınarak tasarlanmış Blue isimli programlama dilini önerirken, Hadjerrouit (Hadjerrouit, 1998), kolay sözdizimi ve programlama mantığını daha uygun olarak öğrettiği gerekçesiyle, acemiler için eğitimde Simula isimli programlama dilinin kullanımını önermiştir. Resnick, Maloney ve diğerleri (Resnick ve diğ., 2009) ise yaşça nispeten daha küçük acemilere yönelik programlama eğitim için Scratch isimli programlama dilini önermişlerdir.

Ticari anlamda popüler ve nesne-yönelimli programlama dillerinin eğitimde kullanılması hususunda ise belli başlı şartlar koşulmuştur. Örneğin Gibbons (Gibbons, 1998), ticari olarak son derece popüler ve çok farklı platformlar üstünde çalışacak şekilde benimsenmiş nesne-yönelimli Java programlama dilinin, ilk olarak asıl paradigmasının dışına çıkılarak yordamsal/yapısal programlama prensiplerinin öğretiminde kullanılması şartıyla, acemilere yönelik programlama derslerinde kullanılabileceğini belirtmiştir. Buna göre, öğrenciler yordamsal programlama becerilerini kazandıktan sonra aynı dille nesne-yönelimli becerileri kazanacakları ikinci bir dersle programlama eğitime devam edebilecektir; zira Gibbons'a göre algoritma mantığı ve yordamsal yapılar gibi programlama prensipleri eğitimde nesne-yönelimli yapılardan önce öğretilmelidir. Benzer biçimde, Burton ve Bruhn (Burton & Bruhn, 2003) da başlangıçta öğrencilerin nesne-yönelimli bir dille eğitim görmekten ziyade, nispeten daha basit olan hem yordamsal paradigmayı hem de GUI uygulama tasarımını öğrenebilecekleri dillerle eğitim görmeleri gerektiğini savunmaktadır. Bu nedenle öğrenimi daha kolay bir yordamsal dil olan C'nin başlangıç dili olarak verilmesini, ardından da GUI uygulama geliştirme imkanına sahip Visual Basic diliyle eğitime devam edilmesini önermektedirler. Bu sayede öğrenciler, gelecekte karşılaşılabilecekleri tüm programlama dillerine uygulayabilecekleri temel bilgilerle donanabilecek ve nesne-yönelimli programlama kavramını daha kolay öğrenebileceklerdir.

2.4 Nesne Yönelimli Dillerle Acemilere Yönelik Programlama Eğitimi – Programlama Paradigmalarının Sunum Sırası

Bilgisayarın tarihi boyunca, programlama dillerinin tasarlanması ve kullanımı için çok sayıda paradigma geliştirilmiştir. Bunların her birinin avantaj sağladığı kullanım alanları bulunuyor olsa da; son yıllar içerisinde, değişkenlerin özel olarak tanımlanması

sayesinde özellikle büyük ölçekli yazılım projelerinde avantaj sunması yönüyle, nesne-yönelimli programlama paradigması diğerlerine kıyasla büyük ölçüde popülerlik kazanmıştır (Capretz, 2003). Günümüzde, son kullanıcıya yönelik yazılım uygulamalarının geliştirilmesinde nesne-yönelimli paradigmaya sık sık başvurulmaktadır ve günden güne daha fazla yazılımcı, önceden alışkın oldukları diğer paradigmalardan koparak, programlama yaklaşımlarını nesne-yönelimli paradigmaya uyacak şekilde yeniden düzenlemeye başlar olmuştur.

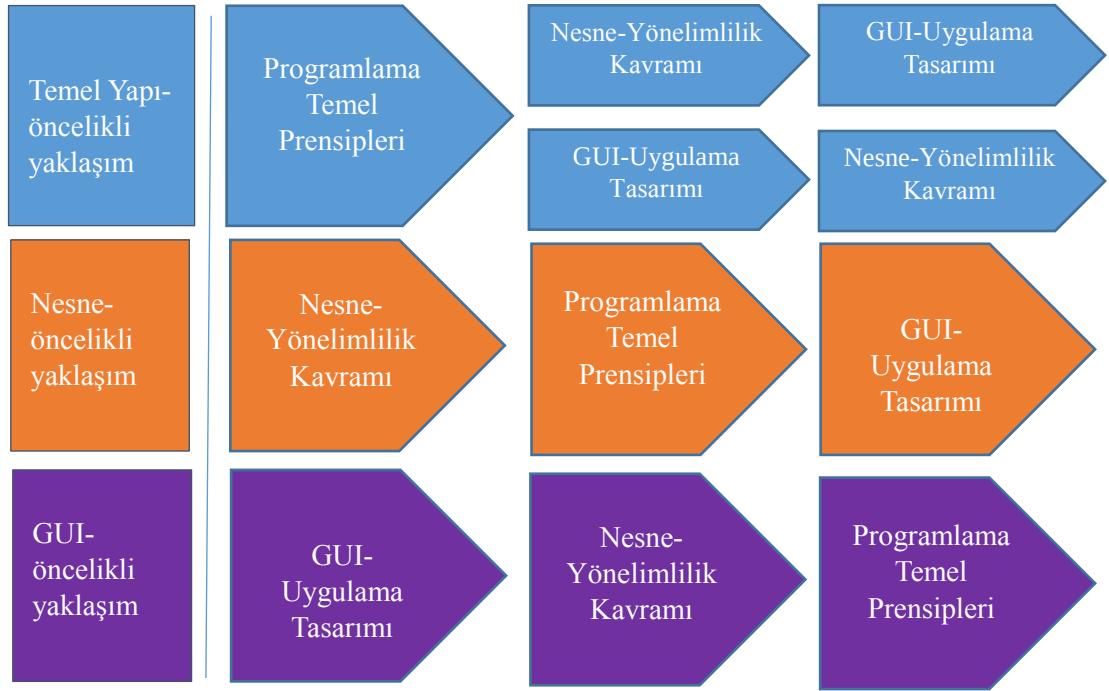
Bu durum, başlangıç seviyesindeki programlama eğitimi açısından da yankılar uyandırmıştır. Örneğin son yıllarda, nesne-yönelimli yazılım geliştiricilere yönelik artan ihtiyacın karşılanabilmesi ve/veya yetiştirilen yazılımcıların ileriki kariyerlerinde kendilerini paradigmalarını yeniden düzenlemek zorunda kalacakları bir noktada bulmamaları için, acemilere yönelik programlama derslerinde “nesne-öncelikli” yaklaşımlar kullanılmaya başlanmıştır (Cooper, Dann, & Pausch, 2003; Decker & Hirshfield, 1993; Kölling, 1999; Kölling & Rosenberg, 1996). Nesne-öncelikli yaklaşıma göre verilen başlangıç seviyesinde programlama eğitiminde ise sıklıkla GUI özelliklerine sahip form uygulamalarına yönelik geliştirme pratikleri ele alınmaktadır (Decker & Hirshfield, 1999; Jiang, Maniotes, & Kamali, 2004; Proulx, Raab, & Rasala, 2002).

Ancak daha önce programlama açısından herhangi bir deneyimi olmayan - henüz algoritma, değişken, operatör, sınıf, metod gibi kavramlara tamamen yabancı olan acemilere yönelik “programlamaya giriş” dersi, halihazırda öğrenciler üzerinde büyük bir bilişsel yoğunluk oluşturmaktadır (Buck & Stucki, 2000). Bu durumu göze alan kimi çevreler ise, değişkenlerin farklı seviyelerde tanımlanmasını gerektiren nesne yönelimli ve nesne özelliklerini manipüle etmeye dayalı form-uygulaması geliştirmeye dayanan yaklaşımların, öğrencilere hedef olarak sunulan öğrenmeler bazında mevcut bilişsel yükün daha da ağır hale getirdiğini iddia etmektedir (Bruce, 2004). Bu nedenle de programlamaya giriş dersinde, temel programlama konseptleri ve algoritma eğitimi daha uygun olan yöntemlerin, ders içi pratiklerin de buna uygun olarak, nispeten daha basit konsol-tabanlı uygulamalar geliştirilmesi şeklinde ilerlemesi şeklinde bir karşı görüş de bulunmaktadır (Jane & Mike, 2004). Bu yaklaşımın başlangıçta öğrencilere daha büyük başarı sağlayabilme potansiyeli bulunsa da, programlamayı bu şekilde öğrenen öğrencilerin daha ileriki aşamalarda modern ve kapsamlı bilgisayar uygulamalarını daha kolay geliştirmek için nesne-yönelimli paradigmaya bir şekilde

alışması gerekebileceği -hatta bu alışma sürecinin en azından 6 ila 18 ay sürebileceği iddia edildiği (Bjarne Stroustrup, 1994) de unutulmamalıdır.

Günümüzde nesne-yönelimli programlama dillerinin, diğer paradigmaları takip eden dillere kıyasla; hem ticari hem de eğitsel açıdan büyük ölçüde popülerlik kazanmış olduğu aşikardır. Hatta kimi çevreler, nesne-yönelimli paradigmanın günümüzde dünyada en fazla rağbet gören programlama yöntemi olduğunu iddia etmektedir (Dovey & Furrow, 2012). Dolayısıyla günümüzde lisans seviyesinde verilen çoğu modern programlama eğitimi programlarının ortak paydası; öğrencilere nesne-yönelimli bir dili kullanarak GUI özelliğine sahip uygulama geliştirme becerisi kazandırma hedefidir. Ancak bu nihai hedefe varmak için takip edilen adımlar çeşitlilik göstermekte olup; öğrencilere programlama kavramlarının öğretimi sırasına göre üç adet ana yaklaşıma rastlanmaktadır. Bu yaklaşımlar ve ilişkin yaklaşımda öğrencilere programlama kavramlarının sunum sırası şekil 2-1’de gösterilmiştir.

Yani resmin geneline bakıldığında, acemilere yönelik programlama eğitiminde nesne-yönelimli dillerin kullanıldığı ancak bu noktada kimi çevrelerin programlama eğitiminde nesne-yönelimli programlama paradigmasına öncelik veren “nesne-öncelikli” ve kimi çevrelerinse aynı dille form-tabanlı uygulamalar geliştirmek yoluyla ve yordamsal programlama paradigmasına öncelik veren “temel yapı öncelikli” yaklaşımları önerdiği görülmektedir. Bilhassa programlama eğitiminin acemilere yönelik aşamasında ise hangi paradigmaya öncelik verileceği hususunda bir fikir birliği bulunmamakta ve öğrenciye daha yüksek fayda sağlanabilmesi için hangisinin önce kullanılacağı konusunda eğitim dünyasında bir tartışma süregelmektedir (Astrachan ve diğ., 2005).



Şekil 2.1 Acemilere yönelik nesne yönelimli programlama eğitiminde önerilmiş yaklaşımlar ve eğitsel kavramların bunlara ilişkin sunum sıraları

2.4.1 Temel-yapı öncelikli yaklaşım

Temel yapı öncelikli (structures-first) yaklaşım, nesne-yönelimlilik kavramı yahut GUI uygulama hususlarından ziyade programlamanın temel kavramlarına öncelik tanımaktadır. Bu yaklaşımı savunan Smolarski, belli bir programlama diline özgü teknik özellikleri öğrenmeye geçmeden önce tüm öğrencilerin programlamaya dair giriş niteliğindeki kavramları iyice anlamış olması gerektiğini iddia etmektedir (Smolarski, 2003). Zira bu araştırmacıya göre, programlama açısından “*dilden bağımsız temel becerileri güçlü bir biçimde kavramış olmak*”, öğrenciye yeni programlama dillerine ve paradigmalarına daha kolay adapte edebilme olanağı sunacaktır. Ayrıca bu yaklaşım; diğer yaklaşım türlerine göre öğrencilere nispeten daha kolay başlangıç konuları sunuyor olduğu için, bu sayede öğrencinin kendine güveni tazelenabilir ve motivasyonu yükselebilir. Bu yaklaşım, öncelikle temel programlama prensiplerini öğretip ancak bunun ardından program tasarımı ve problem çözme hususlarına geçmeyi öneren Sanders ve Mueller tarafından da savunulmuştur (Sanders & Mueller, 2000). Söz konusu yaklaşım, programlama eğitiminde Klasik Öğretim Metodolojisi tarafından önerilen; önce basit program örnekleriyle başlayarak yavaşça daha karmaşık

programlama hususlarına değinilmesi ve bu sayede öğrencinin her kavramı sindirerek bilgisini birikimli olarak artırabilmesi hususuna da paralellik göstermektedir (Cooper ve diğ., 2003). Pedagojik açıdan daha derinlemesine incelemek gerekirse; Bloom'un yeniden gözden geçirilmiş taksonomisinde öğrenmeye ilişkin bilişsel süreçlerin altı (hatırlama, anlama, uygulama, analiz etme, değerlendirme ve yaratma), bilginin ise dört (olgusal, kavramsal, yordamsal ve üstbilişsel) basitten karmaşığa ilerleyen kategoride bulunduğu hatırlanabilir (Fink, 2013; Forehand, 2010; Fuller ve diğ., 2007; Tutkun, Guzel, Koroğlu, & İlhan, 2012). Bu taksonomiye atıfta bulunan Sanders ve Mueller'a göre, bilişsel olarak daha basit ve bilgi olarak daha somut olan "değişken atama" gibi bir evrensel programlama kavramı, programlama eğitiminde "Sınıflar" gibi nesne-yönelimli paradigmaya özgü ve hem bilişsel olarak daha karmaşık ve soyut, hem de analitik düşünme becerisi gerektiren bir konudan önce gelmelidir (Sanders & Mueller, 2000). Aynı araştırmacılara göre, bu bağlamda yordamsal programlama eğitimi de yine nesne-yönelimli eğitimden önce gelmelidir; zira yordamsal programlama deneyimi bulunan öğrenciler, nesne-yönelimli programlama eğitiminde çok daha etkin ve başarılı olmaktadır.

2.4.2 Nesne-Öncelikli Yaklaşım

Nesne-öncelikli yaklaşım, adından da anlaşılabilceği gibi, programlamada *nesneler* kavramının henüz başlangıçta öğretilmesini şart koşturma ve nesne-yönelimli düşüncenin gelişmesini sağlamayı amaçlamaktadır (Cooper ve diğ., 2003). Nesne yönelimli programlama prensipleri üzerine yoğunlaşan bu yaklaşımda, nesneler ve kalıtım hususları, yordamsal kavramlardan önce öğrenciye sunulurken; söz konusu yordamsal kavramların öğretilmesinde ise yine nesne-yönelimli tasarım sınırları içerisinde kalınır (Van Roy ve diğ., 2003). Yani, Sınıflar ve Metotlar gibi kavramlar, değişken atama gibi yordamsal konulardan önce öğretilir ve öğrencilerden daha eğitimin başlangıcında sınıflar ve metotlar oluşturmaya öğrenmeleri beklenir (Lewis, 2000). Bu aşamada nesneler ve bunların birbirleriyle ilişkilerini açıklayabilmek adına genellikle görsel ve kolay anlaşılır bir gösterim yöntemi olan Birleşik Modelleme Dili (Unified Modeling Language, UML) kullanılır; iyice anlaşılmasının ardından da gösterim program koduna dökülür (Burton & Bruhn, 2003).

Nesne-öncelikli yaklaşımın önerilmesinde asıl sebep; öğrencilerin zaman içerisinde bir paradigmadan diğerine geçmeleri durumunda yaşayacakları uyum sorununu ortadan kaldırma çabasıdır (Vilner ve diğ., 2007). Programlamayı temel yapı öncelikli yaklaşım ile öğrenen öğrencilerin daha ileriki aşamalarda modern ve kapsamlı bilgisayar uygulamalarını daha kolay geliştirmek için nesne-yönelimli paradigmaya bir şekilde alışması gerekmektedir ve bu alışma sürecinin en azından 6 ila 18 ay sürebileceği iddia edilmiştir (Bjarne Stroustrup, 1994).

Ancak yine de, bu yaklaşımın acemi öğrencilerde programlama eğitiminde kullanımına karşı çıkan araştırmacılar olmuştur. Sanders ve Mueller, nesne yönelimli yaklaşıma göre tasarladıkları ve bunu uygulamalı olarak denedikleri eğitim programlarında, öğrenme potansiyeli yüksek öğrencilerin dahi zorluk çektiğini saptamış ve temel yapı öncelikli yaklaşıma dönüş yapmışlardır (Sanders & Mueller, 2000). Dünyanın bilgisayar bilimleri açısından en büyük bilimsel kuruluşlarından olan Association for Computing Machinery'ye ait Bilgisayar Bilimleri Eğitimi Özel İlgi Grubu (ACM SIGCSE) üyeleri arasında e-posta aracılığıyla geçen ve Bruce tarafından aktarılan tartışmada da konuyla ilgili pek çok araştırmacının nesne-öncelikli yaklaşımı dezavantajlı bulduğu görülmektedir (Bruce, 2004). Hatta bu tartışmanın sonucunu kimi araştırmacılar nesne-öncelikli yaklaşımın, temel yapı öncelikli yaklaşıma göre tamamen başarısız olduğu şeklinde yorumlamıştır (Astrachan ve diğ., 2005).

2.4.3 GUI-Öncelikli Yaklaşım

GUI-Öncelikli yaklaşım, programlama eğitiminin ilk basamağında yordamsal, yordamsal yahut nesne yönelimli kavramların, program kodunun konsol uygulamalarına dönüştürülerek eğitimini es geçerek; öğrencilerin daha en baştan grafik kullanıcı arayüzüne (Graphical User Interface – GUI) sahip uygulamalar geliştirmeyi öğrenmesini şart koşturmaktadır. Bu sayede öğrenciler, ilk adımdan itibaren sınıflar ve bunlara ilişkin görsel bileşenlerle çalışarak; nesne-yönelimli programlama mantığına alışabilmekte; ancak bundan sonra sırasıyla nesne-yönelimlilik kavramı ve temel yordamsal programlama prensiplerine geçebilmektedirler. (Decker & Hirshfield, 1999; Proulx ve diğ., 2002). GUI bileşenler, statik metin kaplı siyah ekranlardan ibaret konsol uygulamalara nazaran daha gelişmiş bir arayüz sunabilmektedir (Martinez, 2011) ve bu da; yarattığı programın modern bir arayüzle çalıştığını gören öğrenciye doyum ve

motivasyon olarak geri dönmektedir (Hadjerrouit, 1998). Zira bu yaklaşımı izleyen bir programlama dersine devam eden öğrenciler, dersin içeriğini yalnızca teorik ve soyut kavramlar olarak görmekten ziyade somut ve pratikte işe yarayabilecek uygulamalar oluşturuyor olduklarını düşünmektedirler (Gibbons, 1998). Ancak aynı araştırmacıya göre bu yaklaşımın birtakım dezavantajları da mevcuttur, çünkü algoritmik düşünce becerisi ve yordamsal programlama hususlarının yanı sıra, nesne-yönelimli tasarım hususlarına bile başlangıçta yeterince odaklanılmamaktadır. Diğer yandan, GUI oluşturmak zaman zaman öğrencinin dikkatini hem programlamanın dışında yönlere çekmekte, hem de zamanından çalmaktadır.

2.5 Görsel Blok Programlama ve Programlama Eğitiminde Kullanımı

2.5.1 Pedagojik Temel: Eğitimde Yapımcılık (Constructionism)

Bireylere bilgisayar eğitimini en temel seviyede kazandırmanın hedeflendiği bu aşamada, bilgisayar programlamanın bir amaçtan ziyade araç niteliğinde olduğu hususuna dikkat çekilmelidir. Bu tablo, eğitimde Piaget tarafından başlatılan Yapılandırmacılık (Constructivism) kavramının bir uzantısı olan Yapımcılık (Constructionism) akımının yaratıcısı Seymour Papert'ın 1980'lerde Japonya'daki bir konferansta yaptığı konuşmada sanki çok önceden çizilmiştir (Seymour Papert, 1980) :

“... Bir şaire gidip, ‘Ne yapıyorsun?’ diye sormayı deneyin. Şairden ‘Kalem kullanıyorum’ diye bir yanıt alamayacağınız muhakkaktır. Aksine, şair size ‘Şiir yazıyorum’ veya sadece ‘şu an meşgulüm’ diye bir yanıt verecektir. Şairin şiir yazmak için kalem kullanıyor olduğu gayet aşikar olup, bahsetmeye bile değmeyecek bir noktadır. Burada kalem için geçerli olan neyse, bilgisayarlar için de tamamen aynı şey geçerlidir.

Bizler, bilgisayarın öğrenciler tarafından yapılan işler bünyesinde görünmez, telaffuz edilmez bir hal almasını istemekteyiz. Tıpkı matematik gibi. Matematik de insanlar tarafından yapılan işlerin görünmez bir parçası olarak kalmalıdır. Ancak çok sonra, tamamen içselleştirildikten ve bilinç-dışı zihnin bir parçası olduktan sonra, matematik ile ilgili formal dersler almanın ve matematiği soyut bir formal konu olarak öğretmenin zamanı gelecektir. Tabi bu öğretim sürecinde de konu, gerçek dünyaya ait öğelerle her daim ilişkili tutulmalıdır – kullanışlı ve anlamlı öğelerle.”

Bilgisayar programlama eğitimi ile ilgili fikirlerini bu şekilde sözcüğe döken Papert; öğrenmek için en etkin yöntemin, birey için anlam ifade eden ve sosyal yönü bulunan somut objelerin üretilmesi olduğunu öne sürmüştü; ve Piaget'nin “bilişsel yapıların bireyin özelinde inşa edilmesi” fikri üzerine oturtulmuş Yapımcı öğrenme teorisini (S. Papert & Harel, 1991) yaratmıştır. Bu teori doğrultusunda programlama eğitimini başlangıç aşamasında daha basit hale getirmek ve son-kullanıcı yazılım geliştiricilerine daha kolay kullanılabilir bir araç sunabilmek adına ortaya “görsel blok-programlama” isimli bir yaklaşım üretilmiştir (Mohamad, Patel, Tew, Latih, & Qassim, 2011).

2.5.2 Görsel Blok Programlama Dilleri

Görsel blok programlama ortamları, “görsel” (visual) ve “bileşen-temelli” (component-based) programlama mantığını izlemektedir ve sözdizimi (syntax) yerine hazır görsel bloklar şeklinde gelen programlama ifadelerinin “sürükle-bırak” yoluyla bir araya getirilerek yazılımların geliştirilmesine olanak tanımaktadır (Ismail ve diğ. 2012).

Bu yaklaşımın en önemli amacıcı, son-kullanıcıların daha rahat ve pratik yazılım geliştirebilmesine olanak tanımak olsa dahi (Mohamad ve diğ., 2011), blok-programlamanın yazılım geliştirme açısından bir eğitim aracı olarak da kullanılabileceğini öneren belli başlı araştırmalar vardır. Bunlara radikal bir örnek olarak, Wyeth tarafından yürütülen ve 3 ila 8 yaş arası çocuklara temel programlama eğitimini somut objeler seviyesinde, “bilgisayarsız” olarak verebilmeyi amaçlayan, bunun için de öğrencilere birbirine eklenebilen elektronik blok objeleri dağıtılmasını öneren çalışma gösterilebilir (Wyeth & Purchase, 2000). Bunu takip eden benzer çalışmalarda, somut blokların (tangible blocks) küçük yaş çocuklarda programlama eğitimine yönelik ilgiyi ve başarıyı arttırması üzerinde sıkça durulmuş (Horn & Jacob, 2006; McNeerney, 2004), ancak zaman zaman olumlu deneysel sonuçlar elde edilmiş olsa da (Horn & Jacob, 2007; Wang, Zhang, & Wang, 2011) bu konuda daha somut sonuçlara ulaşılmış, deneysel çalışmalara ihtiyaç vardır.

2.5.2.1. Algoritmayı-Hikayeleştiren Görsel Blok Programlama Dilleri

Bilgisayar cephesinde programlama eğitiminde blok yaklaşımını benimsemiş, algoritmayı hikayeleştiren dillere örnek olarak son yıllarda popülerliği yakalamış olan ve Papert'ın uzun bir dönem çalıştığı MIT Media Lab tarafından geliştirilmiş Scratch

verilebilir (J. Maloney ve diğ., 2004; John Maloney ve diğ., 2010). Scratch, yine genç öğrencileri hedefleyen (genellikle 8-16 yaş) ve programlama eğitiminde ilk basamağı teşkil etmesi amacıyla geliştirilmiş bir programlama dilidir. Öğrenciler bu dili kullanarak etkileşimli hikayeler, animasyonlar, oyunlar, müzik ve diğer ürünler ortaya koyabilmekte, bu esnada da temel programlama becerilerini edinmektedir. Ancak Scratch her ne kadar çok sayıda okulda programlama eğitimine giriş basamağında kullanılan bir araç olsa da, (Resnick ve diğ., 2009) bu aracın net faydası ile ilgili yürütülmüş ve nicel verilere dayalı deneysel çalışmaların sayısı fazla değildir.

2.5.3 Mobil Cihazlarla Programlama Eğitimi

Tarihsel sıraya göre, kişisel bilgisayarların maliyet açısından ucuzlayarak çok daha erişilebilir hale gelmesi, kullanım paradigmasının masaüstünden “yaygın bilişime” (Weiser, 1993) doğru evrilmesi, “mobil akıllı cihazların” doğuşu ve ticari yazılımlara yönelik trendlerin 20. yüzyıl sonuna ait kapsamlı ve çok-amaçlı paket programlardan daha yalın, küçük-boyutlu ve tek amaca yönelik “uygulamalara” (*App*) doğru değişmesine paralel olarak, zaman içerisinde yazılım ürünlerine yönelik talebin büyük bir ivmeyle arttığı söylenebilir. Örneğin, araştırma verilerine göre 2011 yılı itibariyle her iki erişkin Amerikan vatandaşından birinin mobil telefonunda en az bir bilgisayar uygulaması (app) bulunmaktadır (Purcell, 2011).

Profesyonel yazılım geliştiricilerin, artan yazılım ihtiyaçlarını karşılamakta yetersiz kaldığı bu gibi bir ortamda, bu kadar fazla sayıda yazılımı geliştirmek için yeni bir kaynağa ihtiyaç duyulmakta olup, bu kaynak, son-kullanıcı yazılım geliştiricileridir.

Son kullanıcı yazılım geliştiricileri (End-user developers), genellikle bilgisayar dışında alanlarda uzmanlık sahibi olup, mesleki veya diğer ihtiyaçlarından ötürü bilgisayar yazılımlarını kendileri geliştirmek zorunda kalan ve bu işten herhangi bir maddi kazanç sağlamayan bireylerdir (Ko ve diğ., 2011). Örnek olarak, bir muhasebecinin yalnızca kendi ofis-içi kullanımı için bir bilgisayar yazılımı geliştirmesini verebiliriz. Ve 2005 yılı itibariyle elde edilen verilere göre, Amerika Birleşik Devletlerinde 2.75 milyon profesyonel yazılım geliştiriciye karşılık 55 milyon son kullanıcı yazılım geliştirici bulunmaktadır (Sarif, Idris, & Zin, 2011). Bunu destekler nitelikte, bilişim endüstrisinde yazılım şirketlerinin, yazılım geliştirme sürecine hitap ettikleri kullanıcıları da dahil etmesi ve ortaya konacak ürünü ortak olarak geliştirmesine dayanan bir model de ortaya

atılmıştır (Han Lun, Chin Chin, Kho, & Hoh, 2008). Diğer yandan, bilgisayar erişimi arttıkça teknolojiyle giderek daha da erken yaşlarda maruz kalmaya başlayan genç nüfus da, bilgisayar üzerinde kendi ilgileri doğrultusunda uygulamalar yaratma hususuna giderek ısınmaktadır (Sandoval-Reyes, Galicia-Galicia, & Gutierrez-Sanchez, 2011). Bu durumda, yazılım geliştirme işinin bu konuda özelleşmiş eğitime sahip seçkin bir azınlığın tekelinde olmaktan çıkma ve insanlığın genelini ilgilendiren bir uğraş halini alma yolunda olduğunu söylemek yanlış olmayacaktır. Dolayısıyla, toplumun her kesiminden insanın en azından temel seviyede bilgisayar programlama becerisine sahip olması gerekliliği, günümüzde kendini her zamankinden fazla hissettirmektedir.

Son-kullanıcı yazılım geliştiriciler, her ne kadar bilgisayar programlama açısından profesyonel geliştiricilerin almış olduğu kapsamlı eğitimi almamış olsalar da, iş yazılım geliştirme noktasına geldiğinde en nihayetinde her iki geliştirici grubu da kendini benzer sorunlarla boğuşurken bulmaktadır (Ko ve diğ., 2011). Ancak her son-kullanıcı geliştiriciye, profesyonellerin almış olduğu kadar kapsamlı bir bilgisayar eğitimi vermek mümkün olmayacağı için, toplumun artan yazılım geliştirme ihtiyacını karşılama noktasında, hem son-kullanıcı geliştiricilerin motivasyonunu ve dolayısıyla sayılarını arttırabilecek hem de bunların karşılaştıkları sorunları azaltabilecek temel eğitim yaklaşımlarının ve yazılım geliştirme araçlarının belirlenmesi gerekmektedir. Ancak bu sayede, toplumun çok sayıda üyesi, bilgisayar bilimlerinin karmaşık dünyasında boğulmadan bilgisayar üzerinde gereksinimlerini karşılayabilecek yazılım ürünleri oluşturabilecektir.

Temel seviyede yazılımlar üretecek geliştiricilere yönelik ihtiyacının karşılanması bir yana dursun, bu tip bir altyapının oluşturulması, toplum bireylerini aynı zamanda bilgisayar bilimleri üzerine kapsamlı bir eğitim sürecine katılmaya daha hazır hale getirebilir ve bir bilişim devriminin kapılarını açabilir.

Bu bağlamda, güçlülere gebe olan ve öğrencilerin eksiksiz konsantrasyonunu gerektiren programlama eğitiminde güdülenmeyi ve buna bağlı başarıyı arttırabilmek için son yıllarda yükselişe geçen yeni bir teknolojiden de faydalanılmaya başlanmıştır: mobil akıllı-cihazlar. Mobil akıllı cihazları kullanarak programlama eğitiminde motivasyon ve buna bağlı başarı elde edilmeye çalışılması gerçekten de son yıllarda göze çarpan bir konudur. Zira Mobil cihazların yüksek kullanım oranları ve genç nüfus

seviyesindeki popülerliği göz önüne alındığında, bu platformları hedef alan eğitsel uygulamaların öğrencilerin ilgisini çekmesi beklenecektir (Rau, Gao, & Wu, 2008). Dvorak’a göre de öğrenciler “yeni” şeyler denemeyi ve yeni araçlarla öğretilen derslere çalışmak için daha fazla vakit ayırmaktadır (Dvorak & Buchanan, 2002). Tam da bu nedenle, Mahmoud ve Dyer, Guelph Üniversitesi’nde yürüttükleri çalışmaya paralel olarak sundukları modelde, öğrencilere programlama eğitimini BlackBerry akıllı telefon cihazları üzerinde vermeyi önermiştir (Qusay H. Mahmoud & Dyer, 2007). Ancak ikili, bu uygulamanın teknik altyapısını detaylandırmış olsa dahi, sistemin öğrenci motivasyonu yahut akademik başarısına yönelik ne gibi etkileri olduğunu açıklamamıştır. Microsoft şirketinin Redmond kampüsünden Tillmann ve arkadaşları ise öğrencilerin programlama eğitimini tamamen mobil cihazları üzerinden alacağı bir sistem geliştirmekte ve tasarım aşamasındaki bu sistemi henüz uygulamalı olarak denememiş olsalar dahi, hayli iddialı konuşarak, programlama eğitiminin geleceğinin bütünüyle “mobil programlamada” yattığını öne sürmüştür (Tillmann ve diğ., 2012). Yine bu doğrultuda Kurkovsky ve Bhagi, ayrı ayrı yürüttükleri çalışmalarda genç öğrencilere programlama eğitimini mobil platformlar için oyun geliştirme yoluyla vermeyi deneyen uygulamalarında, elde edilen olumlu öğrenci motivasyonunun altını çizmişlerdir, ancak keşif amaçlı bu nitel çalışmalarında söz konusu motivasyonun kaynağını derinlemesine incelememişlerdir. Bu durum çalışmalarında, elde edilen motivasyon mobil platformlarda geliştirilen uygulamaların masaüstü çağdaşlarına göre daha basit ve dolayısıyla programlama açısından daha kolay olmasına bağlamıştır, ancak etkinliğin akademik başarıya yahut öğrencilerin gelecekteki programlama becerisine etkisi incelenmemiştir. (Bhagi, 2012; Kurkovsky, 2009). Motivasyon hususunda Jones ve arkadaşları ise, mobil cihazlarla eğitimin öğrencinin derse yönelik motivasyonuna etkisi ile ilgili literatürü inceleyen bir çalışma yürütmüş ve bu cihazların ne gibi özellikleriyle motivasyon sağlayabileceğini belirlemeye çalışmıştır (Jones, Issroff, Scanlon, Clough, & Mcandrew, 2006). Araştırmada, cihazların sunduğu yüksek iletişim kapasitesi, cihazın sunduğu özgürlük hissinin eğitime yönelik motivasyonu olumlu etkileyen faktörleri olduğu, ve akıllı mobil cihaz sahibi olmanın kullanıcılarda yüksek bir “aidiyet” hissi yarattığı ileri sürülmüştür. Yani eğitimde kullanılan akıllı mobil cihazın öğrenciye ait olması durumunun bile motivasyona olumlu etki yapacağı

iddiası bulunmaktadır, ancak bu iddiaları destekleyecek tarama veya uygulama çalışması yürütülmemiştir.

Yapılan alan yazı taramasında görülmektedir ki, öğrencilerin derslere karşı ilgisini çekmek için yeni bir teknoloji olan mobil cihazların ve yaygın bilişim kavramının popülerliğinden faydalanmak, araştırmalara konu olmuş ancak deneme yoluyla kesinliğe kavuşturulmamış bir husustur. Bu nedenle, programlama öğrencileri için günümüzün popüler mobil cihazlarına uygulama geliştirme fikrinin programlama eğitimine yönelik sağladığı motivasyon unsurunun doğasını daha somut bir biçimde kavrayabilmek ve bu motivasyonun akademik başarıya nasıl yansıdığını görmek şeklinde bir ihtiyacın mevcut olduğu söylenebilir

2.5.4 Mobil-Uygulama Geliştirmeye Yönelik Görsel Programlama Dilleri

Google firması tarafından akıllı telefon cihazları için geliştirilmiş Android işletim sisteminde uygulamaların son-kullanıcılar tarafından üretilebilmesinin yolunu açmak üzere ortaya atılmış bir proje olan ve ardından MIT Media Lab'e devredilmiş App Inventor for Android (AIA), geçtiğimiz birkaç yılda ses getirmiş bir görsel blok-programlama dilidir. AIA, arayüz ve kullanım biçimi olarak Scratch ile oldukça benzerlik gösteren bir programlama aracı olup, her yaştan kullanıcıya hitap ettiği iddia edilmiştir. Mobil uygulamalar geliştirme yönünden vaat ettiği motivasyon potansiyeli ve blok-programlama mantığı yoluyla sunduğu kavraması kolay yapısı, araştırmacıları bu programlama ortamının kıymetli bir eğitim aracı olabileceği fikrine götürmüştür. Nitekim, bu mobil cihaz odaklı görsel-blok-programlama dilinin, programlama eğitimine giriş basamağında kullanılabileceği önerilmiş olup (Karakus ve diğ., 2012), öğrencilere motivasyon sağlama açısından yararlı bir araç olduğundan literatürde bahsedilmiştir (Wolber, 2011a).

Blok programlama mantığı ile işletilen ve günümüzün son derece popüler hale gelmiş mobil aygıtları için uygulama geliştirmeye olanak tanıyan bu yeni nesil aracın eğitsel potansiyeli ile ilgili çok sayıda araştırmacı fikir öne sürmüş olsa dahi, tıpkı diğer blok programlama dilleri gibi, ilgili literatürde rastlanan yayınların, uygulama ve istatistiki sonuçlar yönünden eksik kaldığı görülmektedir. Konu hakkında yayında bulunmuş araştırmacıların genel kabulü, bu araçların eğitsel, programlama eğitimine girişte ilk basamakta, programlama becerisi yönünden acemi öğrencilerle kullanımda fayda

sağlayacağı yönündedir. Böylelikle, tüm blok programlama araçlarında olduğu gibi, program geliştirmenin karmaşık yönlerinden biri olan sözdizimi kısmı resmin dışında bırakılarak, öğrencilerin zorlandığı algoritma geliştirme, değişkenler, vs. konulara daha kolaylıkla yoğunlaşılabilceğı öne sürölmüştür. Ancak bu noktada, bilhassa App Inventor aracı etrafında yoğunlaşan iddialara karşın, bu veya benzer araçlarla yürütölmüş yeterli sayıda uygulamalı araştırma olmadığı için, söz diziminin basit bloklar ardında gizlenmesinin programlama eğitimini kolaylaştırma açısından ne denli başarılı olabileceğı ve bunun hangi tip öğrencilere ne kadar yardımcı olabileceğı konusu kesinleşmiş değildir. Örneğın bu tip bir aracın üniversite seviyesinde programlama eğitiminin ileri basamaklarında kullanımından bile fayda görebileceğı iddia edilmiştir. Bu noktada, ortaya fikir öne sürmeye bir süre ara vererek resme daha geniş bir açıdan bakmak, ardından sağlam ve uygulamalı adımlarla ilerlemek daha yararlı olabilir.

BÖLÜM III: YÖNTEM

3.1 Araştırma Modeli

Yarı-deneysel olarak desenlenmiş ve iki aşamadan oluşan çalışmanın ilk aşamasında nicel ve ikinci aşamasında nitel araştırma tekniklerinden faydalanılmıştır. Çalışmanın ilk aşamasında gruplar, deney ve kontrol gruplarına tahsis edilirken kura yoluyla seçkisiz atama (N) yapılmıştır. Bu aşamada deney grubuna “deney grubu 1” ve kontrol grubuna ise” kontrol grubu 1” isimleri verilmiştir. Son test kontrol gruplu deneme modeli uyarınca, gruplar araştırmanın altı haftalık ilk safhasında MTPD ve MBTGPD şeklinde iki farklı araçla (X_1 ve X_2) eğitsel uygulamaya tabi tutulmuştur. Öğrencilere GUI-programlama eğitimi vermeyi amaçlayan uygulama sonunda bağımlı değişken olarak GUI programlama akademik başarısını inceleyen 1.1, 1.2, 1.3 ve 1.4, numaralı hipotezlerin sınanması için Uygulama sınavı 1 (O_2) sonuçları nicel yöntemlerle analiz edilmiştir. Grupların akademik başarı açısından eşliği uygulama öncesi yürütülen ön bilgi testleri (O_1) ile kontrol altına alınmıştır.

N	$O_{1.1}$	X_1	$O_{2.1}$
N	$O_{1.2}$	X_2	$O_{2.2}$

Uygulamanın ilk 6 haftalık safhasında MTPD yerine MBTGPD aracını kullanmanın motivasyon bağımlı değişkeni üzerindeki etkisini inceleyen 2.1, 2.2, 2.3 ve 2.4 numaralı hipotezler ile akıllı mobil cihaz aidiyetine göre görev değeri algısını inceleyen 3.1 ve akademik başarıyı inceleyen 3.2 numaralı hipotezlerinin sınanmasında, son test kontrol gruplu ön-deneme modeli kullanılmıştır. Daha önceden kurayla belirlenmiş gruplarda (N) uygulanan farklı yöntemler (X_1 , X_2) sonrası elde edilen motivasyon testi 1. ölçüm sonuçları (O_1) kıyaslanmıştır.

N	X_1	$O_{1.1}$
N	X_2	$O_{1.2}$

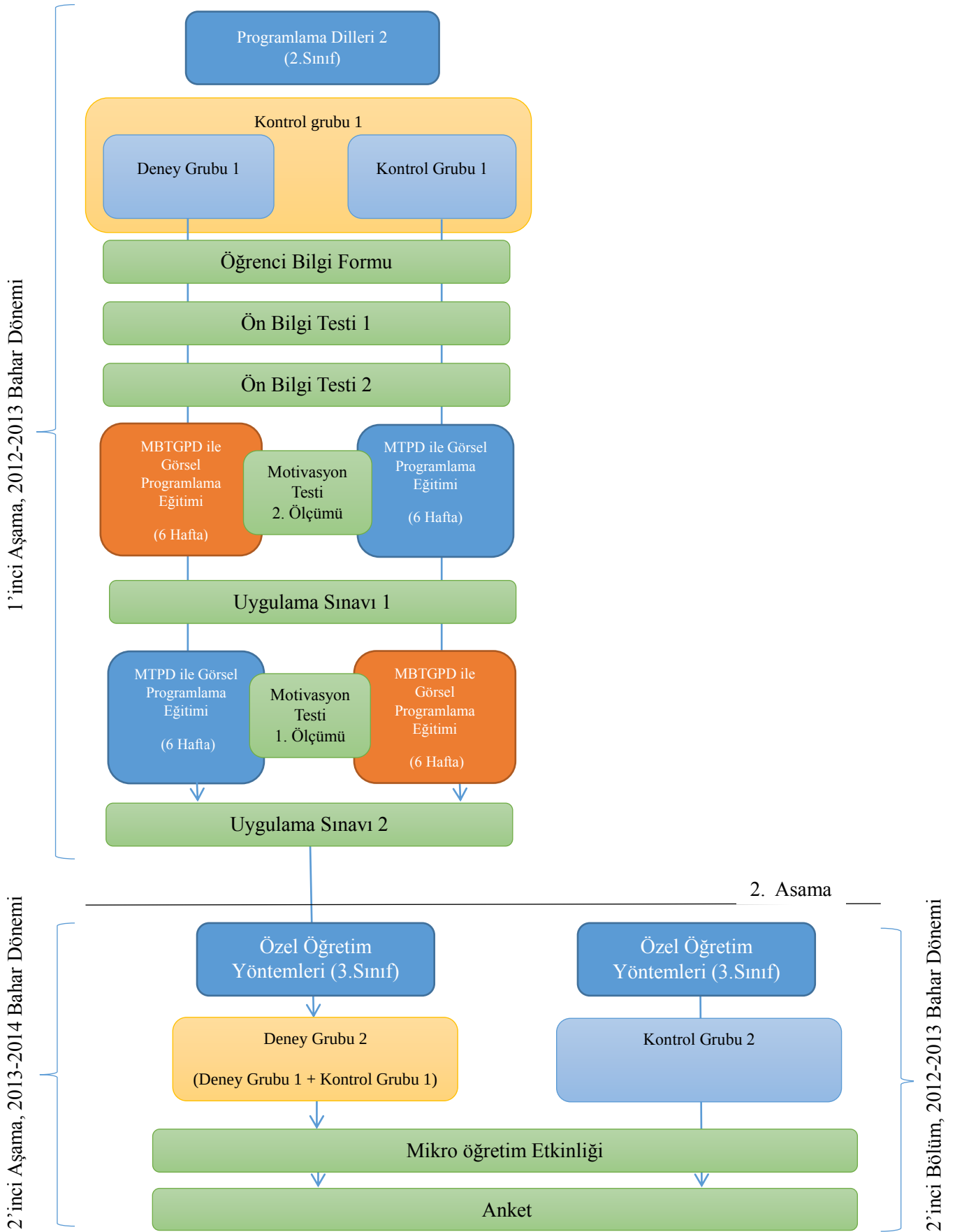
6 haftalık ilk safhanın ardından, yine 6 hafta sürecek ikinci safha başlamış ve bu sürede grupların eğitiminde kullanılan araçlar çapraz biçimde değiştirilmiştir. Böylelikle oluşturulan 2x2 boyutlu bir çapraz (Armaneri & Armaneri, 2006) yarı-deneysel desende ön test son test kontrol gruplu model uyarınca yapılan tekrarlı ölçümler (O_1) ile, eğitimde kullanılan aracın değişimine bağlı olarak motivasyonda değişimi inceleyen 2.5 ve 2.6 numaralı hipotezler test edilmiştir.

N	X ₁	O _{1.1}	X ₂	O _{1.1}
N	X ₂	O _{1.2}	X ₁	O _{1.2}

Nihayet, 12 haftalık GUI-programlama eğitimi sonunda, eğitimde her iki aracı da kullanmış gruplarda MBTGPD ve MTPD araçlarının uygulanma sırasının akademik başarıya etki edip etmediğini inceleyen Hipotez 1.5 ve 1.6 için yine son test kontrol gruplu deneme modeline başvurulmuştur. Akademik başarı açısından eşliği uygulama öncesi yürütülen ön bilgi testleri (O₁) ile kontrol altına alınan gruplar Uygulama Sınavı 2 (O₂) sonuçlarına göre karşılaştırılmıştır

N	O _{1.1}	(X ₁ , X ₂)	O _{2.1}
N	O _{1.2}	(X ₂ , X ₁)	O _{2.2}

Araştırmanın ikinci aşamasında, her iki grubun her iki araçla (MTPD ve MBTGPD) uygulamaya maruz kaldığı deneysel uygulamanın üzerinden 1 yıllık süre geçmesinin ardından ilişkisel tarama modeli (Karasar, 2012) uyarınca uygulamanın yaratmış olabileceği etkileri nitel olarak betimlemeyi hedefleyen bir analiz yürütülmüştür. Bu nedenle, deneysel uygulamanın gerçekleşmesinden bir sene önce programlama dilleri 2 dersinde GUI-programlama konusunu yalnızca MTPD kullanarak öğrenen ikinci bir grup, “Kontrol Grubu 2” olarak belirlenmiştir. Deneysel uygulama kapsamında hem MTPD hem MBTGPD ile eğitim görmüş Deney Grubu 1 ve Kontrol Grubu 1 grupları ise “Deney Grubu 2” ismiyle tek bir grup olarak ele alınmıştır. Ardından, Araştırma Soruları 1, 2, 3, 4, 5 ve 6’nın cevaplanabilmesi için; Deney Grubu 2 ve Kontrol Grubu 2 öğrencilerinin uygulama sonrası anketine verdiği cevaplar ve mikro-öğretim etkinliği sonucu ürettiği videolar nitel yöntemle analiz edilmiştir. Ardından bu veriler, deney grupları ile aynı eğitim aşamalarından geçmiş ancak eğitimde MBTGPD aracını hiç kullanmamış diğer bir gruptan (Kontrol Grubu 2) bir sene önce elde edilmiş verilerle karşılaştırılmıştır. Araştırmaya ait izlençe diyagramı Şekil 3-1’de verilmiştir.



Şekil 3.1 Araştırma İzlenesi

3.2 Çalışma Grubu

Araştırmanın çalışma grubunu, Marmara Üniversitesi Atatürk Eğitim Fakültesi Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü'nde 2012-2013 akademik yılında lisans seviyesinde eğitim gören 2. ve 3. sınıf öğrencileri oluşturmaktadır. Tablo 3.1'de çalışmanın deney ve kontrol grupları ile bunlara yönelik eğitimde kullanılan araçlar paylaşılmıştır.

Tablo 3.1 Çalışma Grupları

Grup Adı		Çalışmanın Yürütüldüğü Dersin Adı		Öğrenci Sayısı	1.Kullanılan Araç	2.Kullanılan Araç
		1. Aşama	2. Aşama			
Deney Grubu 2	Deney Grubu 1	2012-2013 Programlama Dilleri II (2. Sınıf 2.Eğitim)	2013-2014 Özel Öğretim Yöntemleri II (2. Sınıf 2. Egt)	54	MBTGPD	MTPD
	Kontrol Grubu 1	2012-2013 Programlama Dilleri II (2. Sınıf 1.Eğitim)	2013-2014 Özel Öğretim Yöntemleri II (2. Sınıf 1. Egt)	49	MTPD	MBTGPD
Kontrol Grubu 2		-	2012-2013 Özel Öğretim Yöntemleri II (3. Sınıf 1 ve 2. Egt)	76	Yalnızca MTPD	

3.3 Veri Toplama Araçları

3.3.1 Ön Bilgi Testi 1

Ay (Ay, 2011) tarafından C programlama dili için geliştirilen Son Test sorularının .NET C# programlama dilinde yeniden düzenlenmesi ile hazırlanmış 20 soruluk bu test, konsol uygulama geliştirmeye yönelik yordamsal programlama kavramlarına ilişkin bilgi ve beceriyi ölçmektedir. Düzenlemenin ardından geçerlilik-güvenilirlik analizi yeniden yapılmış ve 100 puan üzerinden değerlendirilen testin madde iç tutarlılık katsayısı (Cronbach Alpha) 0,81 olarak bulunmuştur. Bu test ile öğrencilerin PD1 dersi kapsamında edinmiş oldukları yordamsal programlama becerileri ölçülmüş ve grupların bu açıdan eş gruplar olup olmadıkları belirlenmeye çalışılmıştır. Ek 1'de paylaşılmış olan teste ait madde belirtke tablosu, Tablo 3-2'de verilmiştir.

Tablo 3.2 Ön Bilgi Testi 1 Madde Belirtke Tablosu

Hedefler	Sorular
Değişken Kullanımı ve Atama İşlemleri	1, 2, 3, 4, 5, 6, 7, 8, 13, 19, 20
Operatör Kullanımı	1, 5, 6, 7, 9, 19, 20
Karar Yapıları Kullanımı	1, 3, 5, 6, 7, 9, 11, 12, 14, 16, 17, 19, 20
Döngü Yapıları Kullanımı	7, 9, 15, 19, 20
Fonksiyon Oluşturma ve Kullanımı	3, 4, 9, 10, 16, 19
Tek Boyutlu Dizilerin Kullanımı	13, 15, 18, 20

3.3.2 GUI Programlama Eğitimi / Akıllı Mobil Cihaz Bilgi Formu

Öğrencilerin üniversite öncesindeki dönemde metin tabanlı bir dil ile GUI programlama dersi alıp almadıklarını ve uygulamaya konu olan Programlama Dilleri 2 dersi süresince kullanabilecekleri Android işletim sistemine sahip bir akıllı mobil cihaza sahip olup olmadığını belirlemek üzere kullanılmıştır. Öğrencilerin GUI programlama açısından “deneyimli” yahut “acemi” gruplara ayrılmasında kısmen bu formdan elde edilen bilgilerden faydalanılmıştır (Lise seviyesinde görsel programlama eğitimi aldığını beyan eden öğrenciler, bu konuda seviyelerini ölçmek için ayrıca Ön Bilgi 2 testine tabii tutulmuştur).

3.3.3 Ön Bilgi Testi 2

Araştırmacı tarafından oluşturulan bu test, üniversite öncesindeki dönemde GUI programlama eğitimi aldığını beyan etmiş öğrencilerin, bu eğitim sonucu GUI-programlama kavramlarına ne denli aşina olduklarını belirlemek üzere geliştirilmiştir. Testin iki adet kullanım amacı bulunmaktadır:

- Çalışma gruplarının GUI-programlama becerisi açısından eş gruplar olup olmadığının belirlenmesi
- Üniversite öncesinde görsel programlama dersi almış olan öğrenciler arasında, yalnızca bu testte ortalamanın üzerinde skor elde etmiş olan öğrenciler GUI programlama açısından eğitilmiş olarak kabul edilmiştir.

Araştırmacı tarafından geliştirilen, her biri 7.5 puan değerinde 13 sorudan oluşan ve iç tutarlılık katsayısı (Cronbach Alpha) 0,94 olan test Ek 2’de paylaşılmıştır. Bilgisayar laboratuvarında uygulamalı olarak gerçekleştirilmiş teste ait madde belirtke tablosu, Tablo 3-3’te verilmiştir.

Tablo 3.3 Ön Bilgi Testi 2 Madde Belirtke Tablosu

Hedefler	Sorular
Metin Kutusu bileşeninin kullanımı	1, 5
Buton bileşeninin kullanımı	1, 3, 4, 5, 6
Etiket bileşeninin kullanımı	2, 4, 6, 12
Bileşen özelliklerinin değiştirilmesi (tasarım aşamasında)	1, 2, 3, 5
Bileşen özelliklerinin değiştirilmesi (çalışma esnasında)	1, 5, 7, 9
Karar yapılarının kullanımı	6, 7, 8, 9, 10, 11, 12
Mesaj Penceresi kullanımı	8, 12
Aritmetik İşlemlerin kullanımı	6, 7
Aşağı kayan menü bileşeninin kullanımı	10, 11
Bir formdan diğerine veri aktarımı	13

3.3.4 Uygulama Sınavı 1

Öğrencilerin, 6 haftalık eğitim sonunda o ana kadar kullanmış oldukları yazılım geliştirme platformunu kullanarak (Deney grubu için MBTGPD ve Kontrol grubu 1 için MTPD) program geliştirmedeki akademik başarılarını ölçmek amacı ile kullanılmıştır. Sınav, bilgisayar laboratuvarında uygulamalı olarak gerçekleştirilmiştir. Tüm gruplara uygulanan sorular aynı olup, sorular her iki yazılım geliştirme platformunda gerçekleştirilebilecek şekilde hazırlanmıştır. Araştırmacı tarafından geliştirilen sınav, her biri 5 puan değerinde 20 sorudan oluşmaktadır. Ek 3'te paylaşılmış sınavın madde iç tutarlılık katsayısı (Cronbach-Alpha) 0,87 olarak bulunmuştur. Teste ait madde belirtke tablosu, Tablo 3-4'te verilmiştir.

Tablo 3.4 Uygulama Sınavı 1 Madde Belirtke Tablosu

Hedefler	Sorular
Görsel bileşeninin kullanımı	1
Form bileşeni özelliklerine erişim	2
Buton bileşeninin kullanımı	3, 4, 7, 8, 12, 19, 20
Etiket bileşeninin kullanımı	4, 5, 13, 14, 15, 16, 20
Metin kutusu bileşeninin kullanımı	4, 12, 20
Bileşen özelliklerinin değiştirilmesi (tasarım aşamasında)	1, 2, 3, 5
Bileşen özelliklerinin değiştirilmesi (çalışma esnasında)	4, 9, 12, 13, 14
Karar yapılarının kullanımı	7, 8, 10, 15, 17
Aşağı kayan menü bileşeninin kullanımı	6
Mesaj Penceresi kullanımı	8
Onay kutusu bileşeninin kullanımı	10
Zamanlayıcı bileşeninin kullanımı	13, 17
Bir formdan diğerine veri aktarımı	17, 18, 19

3.3.5 Uygulama Sınavı 2

12 hafta boyunca her iki yazılım geliştirme aracı (MBTGPD ve MTPD) kullanılarak alınan eğitim sonunda, deney ve kontrol 1 gruplarında GUI-programlama akademik başarısını ölçmek amacı ile kullanılmıştır. Bilgisayar laboratuvarında uygulamalı olarak gerçekleştiren sınav, 100 puan üzerinden değerlendirilmiştir. Tüm gruplara uygulanan

sorular aynı olup, sorular her iki yazılım geliştirme platformunda da cevaplanabilecek şekilde hazırlanmıştır. Madde iç tutarlılık katsayısı (Cronbach-Alpha) 0,86 olarak bulunan sınav, Ek 4’te paylaşılmıştır. Teste ait madde belirtke tablosu, Tablo 3-5’te verilmiştir.

Tablo 3.5 Uygulama Sınavı 2 Madde Belirtke Tablosu

Hedefler	Sorular
Görsel bileşenin kullanımı	6, 14, 17, 18
Form bileşeni özelliklerine erişim	1, 11
Buton bileşenin kullanımı	2, 4, 7, 10, 13
Etiket bileşenin kullanımı	3, 5, 8, 9
Metin kutusu bileşenin kullanımı	3, 4, 10
Bileşen özelliklerinin değiştirilmesi (tasarım aşamasında)	1, 2, 3, 6, 9
Bileşen özelliklerinin değiştirilmesi (çalışma esnasında)	5, 7, 10, 15, 17, 18
Veri tabanı bağlantısı oluşturulması	4, 5
Bileşenlerin dinamik olarak konumlandırılması	15
Karar yapılarının kullanımı	7, 14, 16, 17
Tek boyutlu dizilerin kullanımı	10
Uygulama alanı bileşenin kullanımı	12, 16
Uyarı penceresi bileşenin kullanımı	19
Zamanlayıcı bileşenin kullanımı	8
Bir formdan diğerine veri aktarımı	10

3.3.6 Motivasyon Ölçeği

Ders kapsamında kullanılan araçların öğrencilerin derse yönelik motivasyonları üzerinde yarattığı etkiyi ölçebilmek için, Güdülenme ve Öğrenme Stratejileri Ölçeği’nin (GÖSÖ) kullanılmıştır. Söz konusu ölçek, Pintrich, Smith, Garcia ve McKeachie’nin (Garcia & Pintrich, 1996) geliştirdikleri bir ölçeğin (Motivated Strategies for Learning Questionnaire) Türkçe’ye uyarlanmış halidir (Büyüköztürk, Akgün, Özkahveci, & Demirel, 2004). Güdülenme Ölçeği (GÖ) ve öğrenme stratejileri ölçeği (ÖSÖ) olmak üzere iki ana bölümden oluşan ölçeğin geçerlik ve güvenirlik çalışmaları Büyüköztürk ve Akgün tarafından ikinci, üçüncü ve son sınıfta okuyan 852 üniversite öğrencisinden elde edilen veriler kullanılarak iki ana bölüm için ayrı ayrı yapılmıştır. Bu araştırmayı ilgilendiren motivasyona yönelik ölçümlerin elde edilebilmesi için, ölçeğin yalnızca Güdülenme Ölçeği (GÖ) ana bölümü öğrencilere uygulanmıştır. Söz konusu ana bölüme ait olan ve Büyüköztürk ve Akgün tarafından bulunmuş Cronbach Alfa geçerlilik katsayıları Tablo 3-6’da verilmiştir.

Tablo 3.6 - Gdlenme ve ğrenme Stratejileri Alt leklerinin Cronbach Alfa Katsayıları

lek	Madde Sayısı	Cronbach Alfa
İsel Hedef Dzenleme	4	0.59
Dıřsal Hedef Dzenleme	4	0.63
Grev Deęeri	6	0.80
ğrenmeye İliřkin Kontrol İnanı	4	0.52
ğrenme ve Performansla İlgili z-Yeterlik	8	0.86
Sınav Kaygısı	5	0.69
Toplam	31	

Gdlenme ve ğrenme Stratejileri leęinin 31 maddelik Gdlenme leęi ana blmn cevaplayan ğrenciler, lekte yer alan her bir ifadeye iliřkin katılma dzeylerini “benim iin kesinlikle yanlış” (1) ile “benim iin kesinlikle doęru” (7) arasında deęiřen Likert tipi yedili derecelendirme leęi zerinde iřaretlemiřtir. Bu uygulamanın ardından, modler yapıdaki leęin yalnızca arařtırmayı ilgilendiren “isel hedef dzenleme”, “grev deęeri” ve “ğrenme ve performansla ilgili z-yeterlilik” alt leklerinden elde edilen ve 1-7 aralıęında deęiřen puanlar kullanılmıřtır.

3.3.7 Uygulama Sonrası Anketi

Deney ve Kontrol gruplarının ders sonrası geen bir yıllık sre ierisinde kiřisel amalarla herhangi bir proje (masast veya mobil) geliřtirip geliřtirmediklerini belirlemek amacı ile kullanılmıřtır. Bylece ğrencilerin ders kapsamında edinmiř oldukları bilgileri gnlk hayatta ne kadar kullanabildikleri de belirlenmiřtir. Ek 5’te paylařılmıř olan anket kapsamında ayrıca, grsel blok programlama ile eęitim almıř ve almamıř ğrencilerin kendilerini aldıkları programlama eęitimi konusunda ne kadar iyi hissettikleri, bu konuda eęitim vermeye hazır hissedip hissetmedikleri ve yeni bir programlama dili ğrenme konusunda ne kadar yeterli hissettikleri gibi aık ulu sorular da sorulmuřtur. Anketin sonuları, dersten sorumlu ğretim elemanının yardımıyla deęerlendirilmiř ve nitel veri olarak analiz edilmiřtir.

3.3.8 Mikroğretim Uygulaması

Her iki aracı (Windows form-tabanlı ve Blok programlama) kullanarak eęitim almıř ğretmen adaylarının, sz konusu eęitimin zerinden 1 sene gemesinin ardından, “Programlamada Dng yapıları” řeklinde seilmiř rnek bir programlama dersi konusu zerinde eęitim vermeleri istendięinde hangi aracı seerek, eęitimi nasıl vereceklerini

belirlemek amacı ile kullanılmıştır. Bu uygulama kapsamında öğrenciler, seçilmiş olan konuyu hangi öğrenme stratejileri ve araçları kullanarak anlatacaklarını sundukları bir günlük ders planı hazırlayarak, bu plan doğrultusunda 20 dakikalık mikro-öğretim videoları çekmişlerdir. Çekilen videolar dersten sorumlu öğretim elemanı ile birlikte analiz edilmiştir.

3.4 Verilerin Toplanması

Araştırma verileri 2012-2013 ve 2013-2014 eğitim yılları boyunca Marmara Üniversitesi BÖTE bölümünde Programlama Dilleri 2 ve Özel Öğretim Yöntemleri 2 derslerinde, ilişkin ölçme araçlarının derslere katılan öğrencilere uygulanması aracılığıyla toplanmıştır. Detaylı bilgi için Şekil 3.1'deki Araştırma İzlenesine bakılabilir. Veriler, derslerden sorumlu öğretim elemanının kılavuzluğunda, araştırmacı tarafından elde edilmiştir. Ön Bilgi Testi 1 ve GUI Programlama Eğitimi / Akıllı Mobil Cihaz Bilgi Formu öğrencilere dersten önce, sınıfta ve kağıt üzerinde uygulanmıştır. Ön Bilgi Testi 2, Uygulama Sınavı 1 ve Uygulama Sınavı 2, bilgisayar programlama becerisini ölçen uygulamalı testler olup, Marmara Üniversitesi BÖTE bölümü bilgisayar laboratuvarlarında gerçekleştirilmiştir. Motivasyon testleri ve Uygulama Sonrası Anketleri, Google Docs isimli web-tabanlı anket sistemi kullanılarak öğrencilere uygulanmıştır. Uygulama sonrası anketi öğrencilerden 1 haftalık süre içerisinde toplanırken, motivasyon testleri her ölçümde yine bölüm bilgisayar laboratuvarlarında ve tek seferde toplanmıştır. Ön Bilgi Testleri, Uygulama Sınavları ve Motivasyon Ölçümleri Programlama Dilleri 2 dersi kapsamında elde edilirken; uygulama sonrası anketi ve mikroöğretim uygulama videoları, öğrencilerden Özel Öğretim Yöntemleri 2 dersi kapsamında elde edilmiştir. Öğrenciler mikroöğretim uygulamalarını 2 hafta içerisinde tamamlayarak, proje raporlarını kağıt üzerinde ve uygulama videolarını ise DVD ortamında teslim etmişlerdir.

3.5 Verilerin Çözümlemesi

Nicel verilerin istatistiksel olarak çözümlenebilmesi için Statistics Package for Social Sciences (SPSS) programının 15'inci sürümü kullanılmıştır. Nitel veriler ise uygulamanın yürütüldüğü dersten sorumlu öğretim elemanının kılavuzluğunda

değerlendirilmiş, QSR Nvivo 10 nitel veri analiz yazılım paketi ile kodlanarak analiz edilmiştir.

3.5.1 Nicel Veriler Üzerinde Yürütülen İstatistikler

3.5.1.1 İki Gruba Ait Ortalamaların Karşılaştırılması: Bağımsız Örneklem T-Testi ve Parametrik Olmayan Eşleniği Mann-Whitney U Testi

Eğitim araştırmalarında ilişkisiz gruplara ait ölçüm verilerinin istatistiki olarak karşılaştırılması için, bağımsız örneklem t-testi önerilmiştir (Büyüköztürk, 2002). Dolayısıyla, iki gruba ait verilerin karşılaştırılmasını gerektiren tüm hipotezlerde ($H_{1.1}$, $H_{1.4}$, $H_{2.1}$, $H_{2.4}$, $H_{3.1}$) bu testten faydalanılmış; ancak verilerde normal dağılım şartının ihlal edildiği durumlarda ($H_{1.5}$, $H_{1.6}$, $H_{3.2}$) testin parametrik olmayan eşleniği olan Mann-Whitney U testine başvurulmuştur (Büyüköztürk, 2002).

3.5.2 Faktörler Arası Etkileşim: Tek Değişkenli Faktöriyel Varyans Analizi

Hipotez 1.2, 1.3, 1.4, 2.2, 2.3 ve 2.4'ün test edilmesinde, bağımsız değişken niteliğindeki iki faktörün bir bağımlı değişken üzerindeki etkileri arasındaki etkileşimi belirlemek için, dikey olmayan (*non-orthogonal*) verilere sahip 2x2 faktör hücreleriyle varyans analizi gerçekleştirilmiştir (Jaccard, 1998; Keselman ve diğ., 1998)

3.5.2.1 Tek Değişkenli Kovaryans Analizi

Motivasyona yönelik bir aracın tekrarlanan ölçümleri üzerinde çalışılacak Hipotez 2.5 ve 2.6'nın test edilebilmesi için, ikinci ölçümde grup ortalamaları arasındaki farkı incelerken ilk ölçüm değerlerini istatistiksel olarak kontrol alındığı tek değişkenli kovaryans analizi tekniğinden faydalanılmıştır (Keselman ve diğ., 1998).

3.5.2.2 Ki-Kare Testi

Araştırma sorusu 1'de iki gruptan elde edilmiş kategorik veriler arasında istatistiksel fark gözlenip gözlenmediğinin test edilebilmesi için, Pearson Ki-kare istatistiksel testinden faydalanılmıştır.

3.6 Uygulama

Araştırmada öğrencilere GUI-programlama eğitiminin, yarı-deneysel desen kapsamında verildiği eğitim uygulaması, 2012-2013 akademik yılı bahar döneminde Marmara Üniversitesi BÖTE bölümü Programlama Dilleri 2 dersi kapsamında gerçekleştirilmiştir. Söz konusu ders, 3'ü teorik ve 2'si uygulama olmak üzere her hafta birbirini izleyen 5 saat boyunca işlenmiş olup, 12 hafta devam etmiştir. Dersin teori kısmı, BÖTE bölümü dersliklerinde işlenmiş ve bu saatlerde dersten sorumlu öğretim elemanı öğrencilere bir bilgisayar ve projektör sistemi yoluyla o haftaya tekabül eden yazılım projesini anlatmıştır. Bunu takip eden uygulamalı kısımda ise bilgisayar laboratuvarları kullanılmış ve öğrenciler öğretim elemanının rehberliğinde, teori saatlerinde anlatılan yazılımı kendi başlarına geliştirmeye ve yeni özellikler eklemeye çalışmıştır.

Uygulamanın ilk altı haftası boyunca, Deney Grubu 1 öğrencileri MBTGPD ile, Kontrol grubu 1 öğrencileri ise MTPD aracı ile öğrenim görmüştür. Bu sürenin sonundaki 7. haftada her grup, önceki 6 haftalık süre boyunca gördükleri konuları kapsayan ve öğrenim gördükleri araç ile cevaplamaları beklenen Uygulama Sınavı 1'e tabi tutulmuştur. Bundan sonra deney gruplarının derste kullandığı araçlar çapraz şekilde değiştirilmiştir. Buna göre, 7. Haftadan uygulamanın sonuna kadar Deney Grubu 1 öğrencileri eğitime MTPD ile, Kontrol grubu 1 öğrencileri ise MBTGPD ile devam etmiştir.

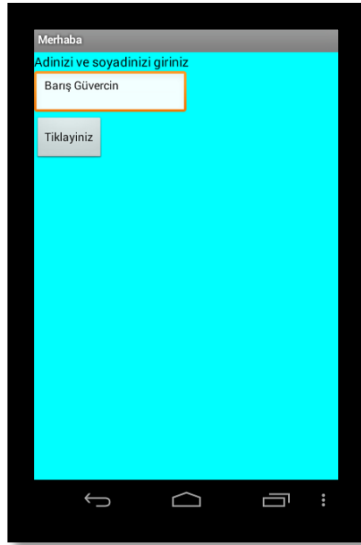
11. haftada öğrencilerden, MTPD veya MBTGPD araçlarından herhangi birini kullanarak geliştirecekleri ödev niteliğinde bir projeyi bir hafta içerisinde teslim etmeleri istenmiştir.

Nihayet 12. Haftada öğrenciler yine, MTPD veya MBTGPD araçlarından herhangi birini kullanarak cevaplayabilecekleri sorulardan oluşan ve bundan önceki haftalar boyunca görülen ders konularını kapsayan Uygulama Sınavı 2'ye tabi tutulmuştur.

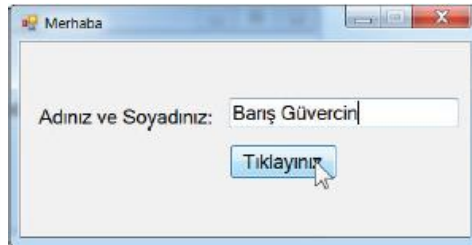
Öğrencilerin haftalık ders içerikleri ve her hafta üzerinde çalışılmış olan yazılım projelerinin detayları aşağıda verilmiştir.

1. Hafta – “Merhaba”: Birinci derste, App Inventor ile öğrenim gören Deney Grubu 1 öğrencilerine, App Inventor’ın bilgisayara kurulumu gösterilmiş ve IDE genel hatlarıyla tanıtılmıştır. Visual C# öğrenen Kontrol grubu 1 öğrencilerine ise, bu dile ilişkin IDE (Visual Studio) içerisinde Form uygulamalarının oluşturulması ve IDE’nin GUI uygulama geliştirme arayüzü tanıtılmıştır. Ardından her iki gruba da, Şekil 3-2’de örnek ekranları görülen, kullanıcının ekrandaki bir metin kutusuna ismini yazmasını ve butona bastığında, kullanıcıyı ismiyle selamlayan bir mesaj penceresinin oluşmasını sağlayacak bir programı nasıl geliştirecekleri gösterilmiştir. Bu derste öğrencilere aşağıdaki hedef davranışların kazandırılması planlanmıştır:

- IDE’yi kurma, kullanıma hazır hale getirebilme, arayüzü tanıma
- Yeni boş GUI uygulama projesi oluşturabilme
- Metin kutusu, buton, etiket ve uyarı mesajı bileşenlerini kullanabilme



(a)



(b)

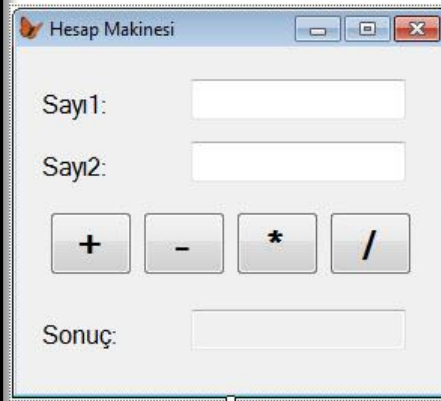
Şekil 3.2 “Merhaba” uygulaması. (a) MBTGPD ile, (b) MTPD ile

2. **Hafta - Basit hesap makinesi:** İkinci derste her iki grubun öğrencilerine de Şekil 3-3'te örnek ekranları görülen; kullanıcının girdiği iki sayıyı, tercih edilen dört matematik işlemden birine tabi tutan ve sonucu ekrana yazdıran basit bir hesap makinesi uygulamasının nasıl hazırlanacağı gösterilmiştir. Bu derste öğrencilere aşağıdaki hedef davranışların kazandırılması planlanmıştır:

- GUI uygulamalarda aritmetik operatörleri kullanabilme
- Koşullu ifadeleri kullanabilme
- Metin kutusu bileşenini uygulama çalışırken temizleyebilme
- Form/Ekran özelliklerini değiştirebilme
- Bileşenleri ekranda konumlandırabilme



(a)



(b)

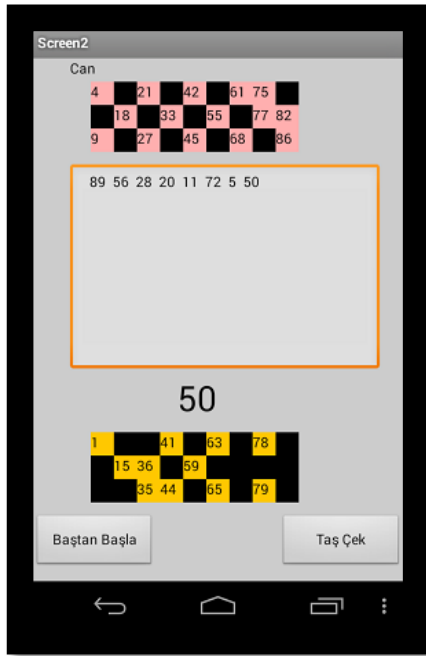
Şekil 3.3 Basit Hesap Makinesi uygulaması – (a) MBTGPD ile, (b) MTPD ile

3. **Hafta – Tombala:** Üçüncü derste her iki grubun öğrencilerine de, bilgisayara karşı oynanan klasik bir tombala oyunu uygulamasının nasıl hazırlanacağı anlatılmaya başlanmıştır. İlk aşamada, combobox ve listbox bileşenleri, tombala kartlarının oluşturulması, renklerinin değiştirilmesi ve taşların seçkili olarak çekilmesi işlemleri kapsamında aşağıdaki hedef davranışlar üzerinde durulmuştur:

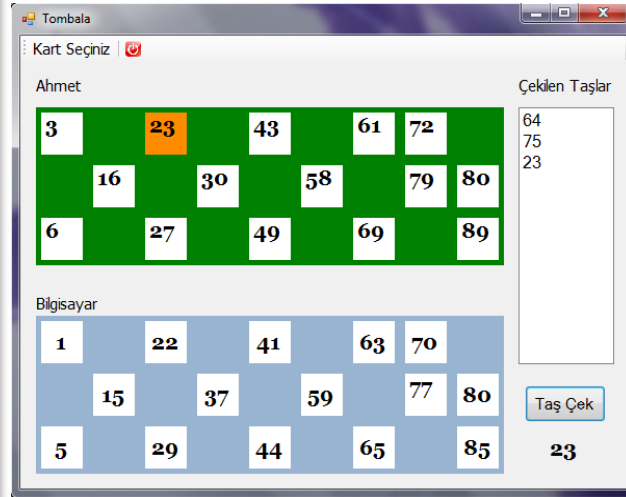
- “Combobox”bileşenini kullanabilme
- “Listbox” bileşenini kullanabilme
- Tek boyutlu dizi oluşturabilme ve bunun elemanlarına erişebilme
- Ekrana çok sayıda bileşeni düzgün biçimde yerleştirebilme, panel kullanımı
- Çok sayıda bileşen özelliğini döngü yoluyla tek seferde değiştirebilme

4. Hafta – Tombala: Dördüncü derste, Şekil 3-4’te örnek ekranları görülen tombala oyunu uygulaması tamamlanmıştır. Bu noktada çekilen taşların rastgele gelmesini sağlamak için Fisher-Yates shuffle algoritmasından bahsedilmiştir. Diğer yandan, bir formdan diğerine veri aktarılması da gösterilerek, oyunun galibinin kim olduğunun ikinci bir ekranda sunulması fonksiyonu da eklenmiştir. Bu amaçla aşağıdaki hedef davranışlar üzerinde durulmuştur:

- Birden çok form kullanabilme
- Formlar arası bilgi geçişini sağlayabilme
- Dizi elemanlarını karıştırabilmek için Fisher-Yates Shuffle Algoritması kullanabilme



(a)

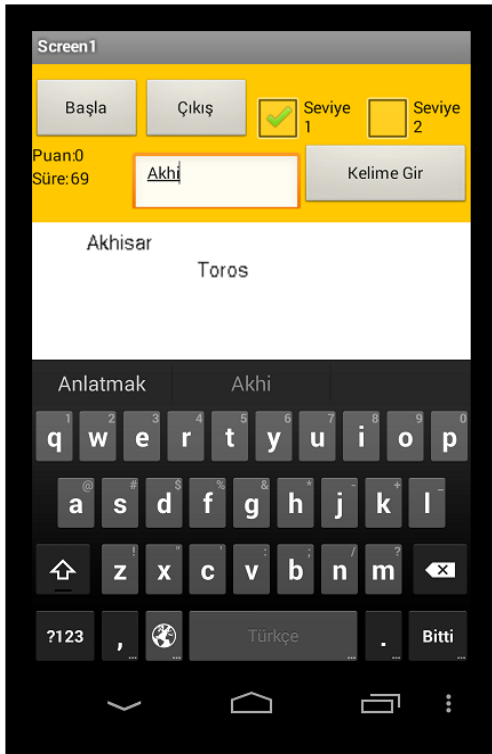


(b)

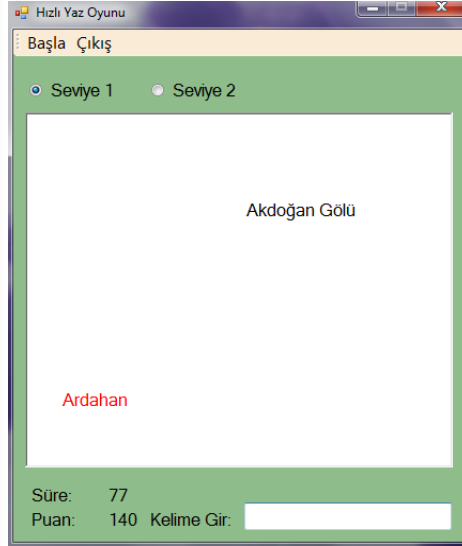
Şekil 3.4 Tombala uygulaması – (a) MBTGPD ile, (b) MTPD ile

5. **Hafta – Hızlı Yaz:** Beşinci derste her iki gruba da, Şekil 3-5’te örnek ekranları görülen soldan sağa yönde ekrandan hızla geçen kelimeleri süre bitmeden doğru biçimde yazmayı amaçlayan “Hızlı Yaz” isimli küçük bir oyun uygulamasının nasıl geliştirileceği gösterilmiştir. Bu oyunun geliştirilmesi esnasında öğrencilere aşağıdaki hedef davranışların kazandırılması öngörülmüştür:

- “Checkbox” ve “RadioButton” bileşenlerini kullanabilme
- “Canvas” ve “Panel” bileşenlerini kullanabilme
- Zamanlayıcı bileşenini kullanabilme
- GUI objelerini dinamik olarak konumlandırabilme



(a)



(b)

Şekil 3.5 Hızlı Yaz uygulaması – (a) MBTGPD ile, (b) MTPD ile

6. **Hafta – Uygulama Sınavı 1:** 6. Haftada uygulanan bu sınavda, her iki gruba da aynı sınav uygulanmış ancak sınav sorularını, öğrenmiş oldukları dili kullanarak cevaplamaları istenmiştir.

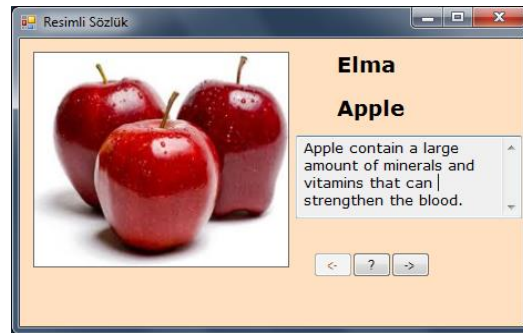
7. Hafta – Yeni IDE’nin tanıtımı: : 7. Haftadan itibaren, çapraz uygulama deseni doğrultusunda bu ana kadar MBTGPD ile öğrenim görmüş olan Deney Grubu 1 öğrencileri eğitime MTPD ile ve MTPD ile öğrenim görmüş Kontrol grubu 1 öğrencileri ise MBTGPD ile devam etmiştir. Bu hafta boyunca, öğrencilere “Merhaba” ve “Hesap Makinesi” uygulamaları gösterilerek çalışacakları yeni dil ve IDE kullanılarak en temel biçimiyle gösterilmiş, “Tombala” ve “Hızlı Yaz” uygulamalarının yeni geçilen IDE’ye göre hazırlanmış ders notları dağıtılmıştır.

8. Hafta – İngilizce-Türkçe Sözlük: 8. Haftada, her iki grubun öğrencilerine de ilişkin araç kullanılarak, Şekil 3-6’da görülen basit bir İngilizce-Türkçe sözlük uygulamasının nasıl oluşturulacağı gösterilmiştir. Sözlük uygulamasında erişilmesi istenen veriler XML formatında saklanmış ve öğrencilere uygulamanın bu XML dosyasını nasıl çözümleneceği izah edilmiştir. Bu uygulamanın geliştirilebilmesi için öğrencilere aşağıdaki konularda eğitim verilmiştir:

- Resim bileşeninin kullanımı
- Çok satırlı metin kutusu bileşeninin kullanımı
- Projeye XML dosyasının eklenmesi ve çözümlenmesi



(a)

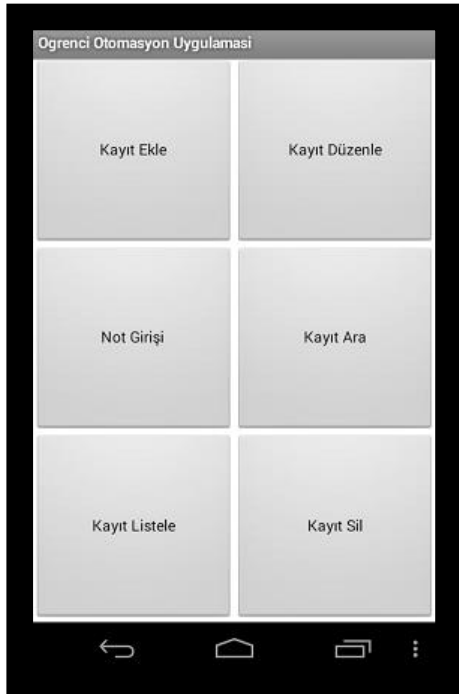


(b)

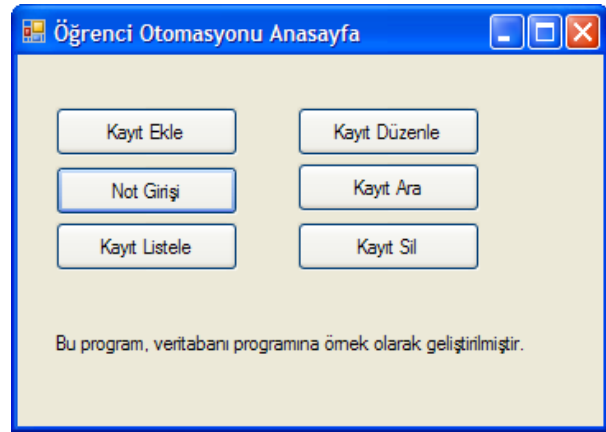
Şekil 3.6 Resimli Sözlük Uygulaması – (a) MBTGPD ile, (b) MTPD ile

9. Hafta ve 10. Hafta Öğrenci Otomasyon Uygulaması – 9. ve 10. haftalarda, her iki grubun öğrencilerine de ilişkin araç kullanılarak, Şekil 3-7 ve 3-8’de örnek ekranları görünen bir “Öğrenci Otomasyon Sistemi” uygulamasının nasıl hazırlanacağı gösterilmiştir. Söz konusu uygulama, veritabanı bağlantısı kurulmasını ve SQL yönetim dili ile söz konusu veritabanındaki verilerin görüntülenebilmesi veya üzerinde değişiklik yapılabilmesini gerektirmektedir. MTPD ile çalışan öğrenciler uygulamaya bağlayacakları veritabanını Microsoft Access 2010 programı ile oluşturmuş, MBTGPD ile çalışan öğrenciler ise bulut-temelli Google Fusiontablos veritabanını kullanmıştır. İki hafta boyunca öğrenciler, veritabanına bağlanmayı öğrenmiş ve SQL sorguları göndererek veritabanına girdi ekleme / çıkartma / düzenleme / listeleme işlemlerini gerçekleştirmiştir. Bu uygulamanın geliştirilebilmesi için öğrencilere aşağıdaki konularda eğitim verilmiştir:

- Birden çok form kullanabilmek
- Formlar arası bilgi geçişini sağlamak
- Veritabanı bağlantısı ve SQL diliyle yönetimi



(a)



(b)

Şekil 3.7 Öğrenci Otomasyonu uygulaması ana menü– (a) MBTGPD ile, (b) MTPD ile

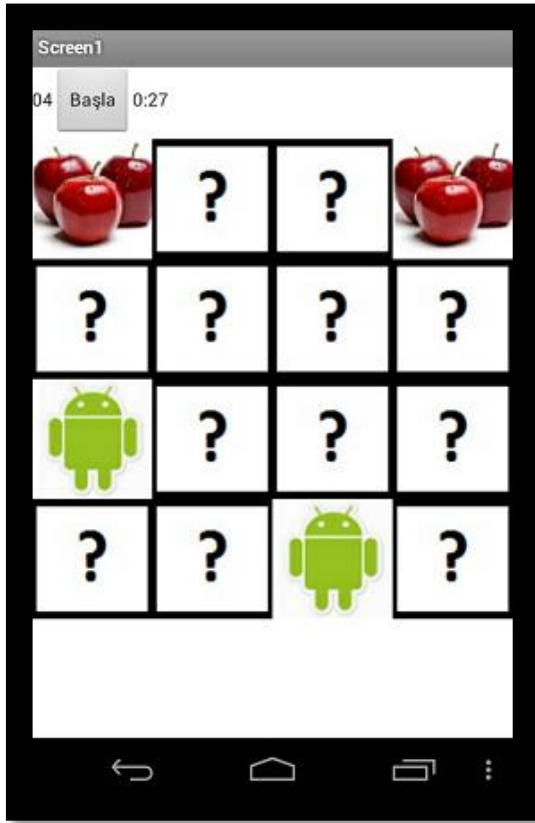
(a)

(b)

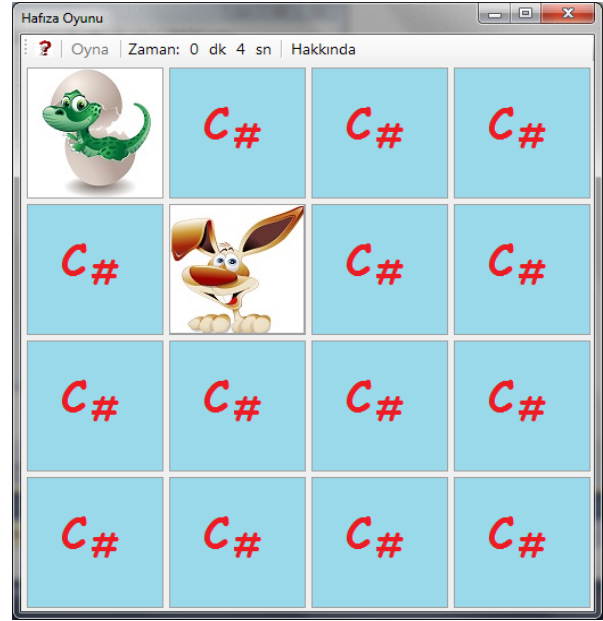
Şekil 3.8 Otomasyonu uygulaması kayıt ekleme – (a) MBTGPD ile, (b) MTPD ile

11. Hafta – Hafıza Oyunu, Öğrenci proje ödevlerinin verilmesi: 11. Haftada, her iki grubun öğrencilerine de ilişkin araç kullanılarak, Şekil 3-9’da örnek ekranları görülen “Hafıza Oyunu” uygulamasının nasıl geliştirileceği gösterilmiştir. Oyunda, ekranda 16 adet kart bulunmaktadır ve oyuncu herhangi bir karta dokunduğunda, kartın arkasında bulunan resim 2 saniye süreyle oyuncuya açık edilmekte ve tekrar kapatılmaktadır. Bu şartlar altında oyuncudan, rastgele düzene göre yerleşmiş kartların arkasında bulunan resimlerin eşlerinin bulunması istenmektedir. Eşi bulunan kartların resim tarafları oyunun sonuna kadar açık kalmaktadır. Bu oyunun geliştirilebilmesi için öğrencilere aşağıdaki konularda eğitim verilmiştir:

- Dinamik kontrollerin oluşturulması
- İki boyutlu dizi kullanımı
- Rastsal sayı üretimi
- Resim dosyalarının kullanımı
- Ses dosyalarının çalıştırılması
- Etiket, Buton, Resim, Zamanlayıcı bileşenlerinin kullanımı



(a)



(b)

Şekil 3.9 Hafıza oyunu uygulaması– (a) MBTGPD ile, (b) MTPD ile

Bunun yanı sıra, aynı hafta içerisinde öğrencilere, bir hafta içerisinde teslim edilmek üzere bir yazılım projesi ödevi verilmiştir. Öğrencilere bu yazılım projesi ödevini, MBTGPD veya MTPD arasından kendi tercih ettikleri araç ile yapabilecekleri duyurulmuştur.

12. Hafta – Uygulama Sınavı 2, Öğrenci projelerinin toplanması: Nihayet 12. haftada öğrencilere, kendi tercihleri doğrultusunda istedikleri soruyu MBTGPD veya MTPD araçlarından birini kullanarak cevaplayabilecekleri yapıda hazırlanmış olan Uygulama Sınavı 2'ye tabi tutulmuştur. Aynı hafta içerisinde, bir önceki hafta verilmiş olan proje ödevleri toplanmış ve değerlendirilmiştir.

BÖLÜM IV: BULGULAR

4.1 Uygulama Öncesi Bulgular

4.1.1 Gruplarda Karşılaştırılabilirlik Testleri

Çalışma gruplarının programlama becerileri açısından eş grup olarak kabul edilip edilemeyeceğini belirlemek amacıyla, bu becerileri ölçen ön bilgi testlerinin (sırasıyla, Ön Bilgi Testi 1 ve Ön Bilgi Testi 2) sonuçları istatistiki olarak karşılaştırılmıştır.

4.1.1.1 Deney gruplarının yordamsal programlama becerisi açısından eşliği

Deney grubu 1 ve Kontrol grubu 1'nin, Programlama Dilleri 2 dersi öncesinde yordamsal programlama becerileri açısından eş grup olup olmadığının anlaşılması için, Ön Bilgi Testi 1 sonuçları karşılaştırılmıştır. Sonuçların normal dağılım sergileyip sergilemediğini belirleyebilmek adına Shapiro-Wilk normallik testine başvurulmuştur. Her iki grupta da normal dağılım gösteren (Deney Grubu 1, $p = .17$; Kontrol grubu 1, $p = .51$) Ön Bilgi Testi 1 sonuçlarının birbirlerinden istatistiki ölçüde farklı olup olmadığını anlayabilmek adına Bağımsız Örneklem t-testine başvurulmuş, sonuçlar Tablo 4.1'de verilmiştir.

Tablo 4.1 Deney gruplarının Ön Bilgi Testi 1 sonuçlarına göre karşılaştırılması – Bağımsız örneklem t-testi sonuçları

Grup	n	$\bar{x}$	ss	t-testi		
				sd	t	p
Deney Grubu 1	54	45,83	14,77	99	,084	.93
Kontrol grubu 1	47	46,10	17,78			

Buna göre Ön Bilgi Testi 1 sonuçları açısından deney grupları arasında anlamlı bir farka rastlanmamıştır ($p = .93$) ve neticede deney gruplarının Programlama Dilleri 1 dersi kapsamında öğrenilmiş yordamsal programlama becerileri açısından eş gruplar olduğu kabul edilmiştir.

4.1.1.2 Deney gruplarının GUI-programlama becerisi açısından eşliği

Deney grubu 1 ve Kontrol grubu 1'de bulunan kimi öğrencilerin üniversite öncesinde aldıkları görsel programlama derslerinde edinmiş olduğu GUI-uygulama geliştirme becerileri açısından eş grup olup olmadığının anlaşılması için, Ön Bilgi Testi 2 sonuçları

karşılaştırılmıştır. Sonuçların normal dağılım sergileyip sergilemediğini belirleyebilmek adına Shapiro-Wilk testi yürütülmüştür. Buna göre gerek Deney Grubu 1'in ($p = ,00$) gerekse Kontrol grubu 1'nin ($p = ,00$) Ön Bilgi Testi 2 sonuçları açısından normal dağılım sergilemediği görülmüştür. Bu nedenle gruplara ait Ön Bilgi Testi 2 sonuçlarının birbirlerinden istatistiki ölçüde farklı olup olmadığını anlayabilmek Mann Whitney U non parametrik istatistik testine başvurulmuş ve sonuçlar Tablo 4.2'de verilmiştir.

Tablo 4.2 Deney gruplarının Ön Bilgi Testi 1 sonuçlarına göre karşılaştırılması – Mann Whitney U Testi sonuçları

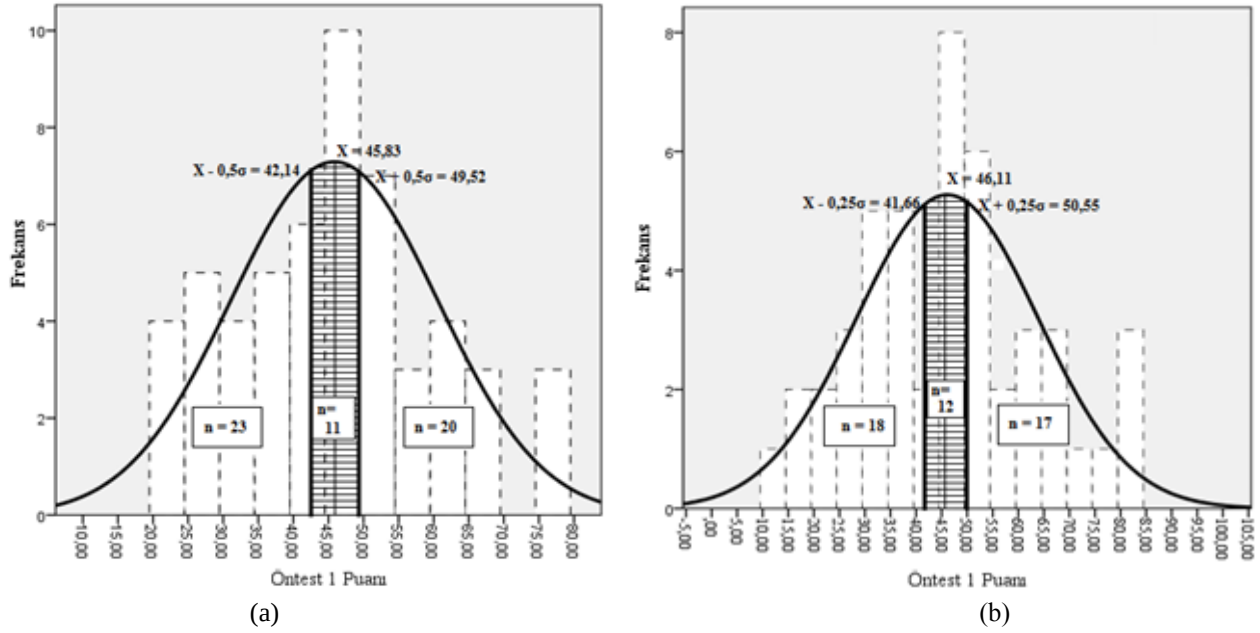
Grup	N	Sıra Ortalaması	Sıra Toplamı	U	p
Deney Grubu 1	54	52,04	2810,00	1213,00	.69
Kontrol grubu 1	47	49,81	2341,00		

Buna göre Ön Bilgi Testi 2 sonuçları açısından deney grupları arasında anlamlı bir fark bulunmamaktadır ($U=1212,00$ $p>.05$). Dolayısıyla deney gruplarının programlama dilleri 2 dersi öncesinde GUI-programlama eğitim seviyeleri açısından eş gruplar olduğu kabul edilmiştir.

4.1.2 Faktör Gruplarının Oluşturulması

4.1.2.1 Programlama Dilleri 1 Dersi Başarısı Faktörü

Öğrencilerin, üniversitede daha önce gördükleri Programlama Dilleri 1 dersi kapsamında öğrenilen yordamsal programlama becerileri açısından başarılı ve başarısız faktör gruplarına ayrılmasına ihtiyaç duyulduğu için, yordamsal programlama becerisini ölçen Ön Bilgi Testi 1 skorlarının dağılımı incelenmiştir. Neticede, başarılı öğrenciler ile başarısızlar arasında bir ayrım oluşturabilmek için ortalamanın 0,25 standart sapma altında kalan öğrenciler Programlama Dilleri 1 dersi konuları açısından başarısız, 0,25 standart sapma üzerinde kalan öğrenciler ise başarılı kabul edilmiştir. Böylece öğrenciler arasında 0.5 standart sapmalık aralık bırakılmış, bu esnada gruplardaki öğrenci sayısının istatistiki sonuçları etkileyecek kadar düşmesi önlenmiştir. Bu işlemin detayları grafiksel olarak Şekil 4.1a ve 4.1b'de gösterilmiştir.



Şekil 4.1 Ön Bilgi Testi 1 skorlarına göre Deney Grubu 1 üyelerinin (a) ve Kontrol grubu 1 üyelerinin (b) 0,25 standart sapma alt ve üst sınıra göre ayrılması

Buna göre, Programlama Dilleri 1 dersi başarıları faktör gruplarına öğrenciler Tablo 4.3'te görüldüğü şekilde tahsis edilmiştir.

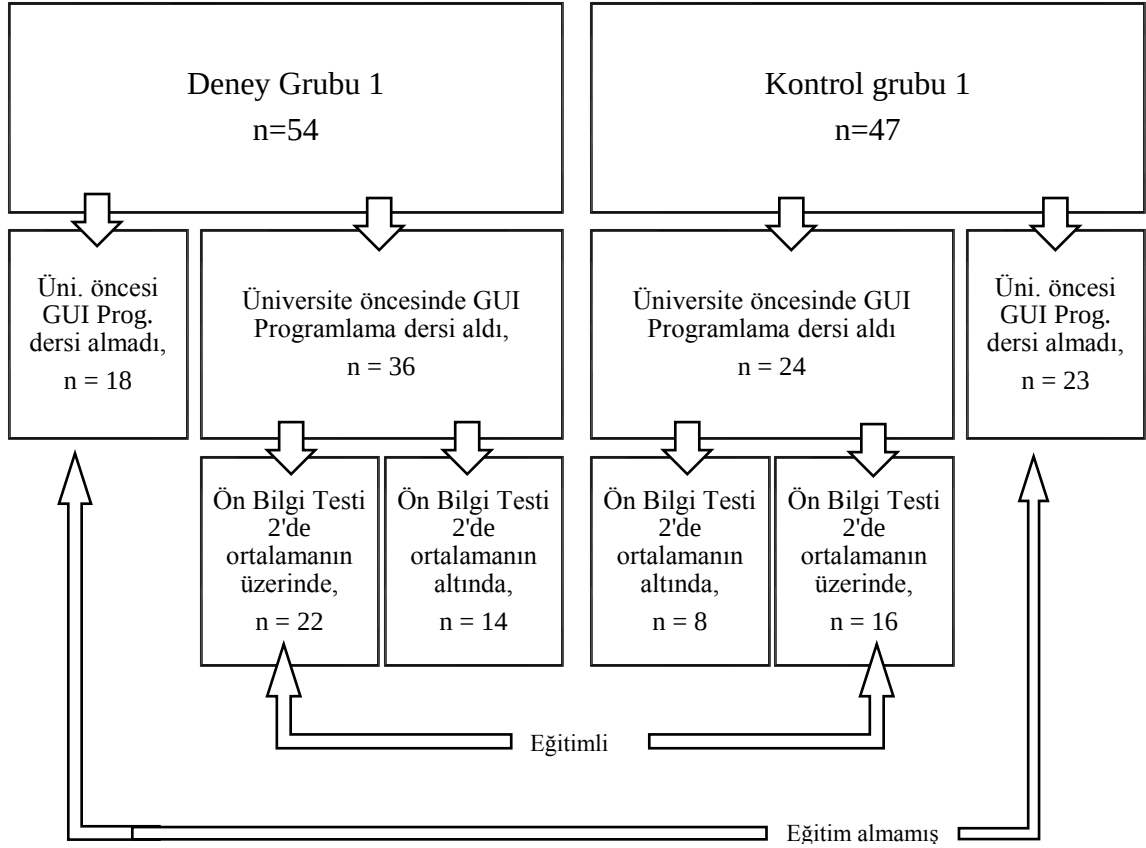
Tablo 4.3 Öğrencilerin Yordamsal Programlama Becerisi faktör grubuna tahsis edilmesi

Grup	“Başarısız” grubuna tahsis edilen öğrenci sayısı	Ortalamanın 0,25 standart sapma alt ve üstünde kalan öğrenci sayısı	“Başarılı” grubuna tahsis edilen öğrenci sayısı
Deney Grubu 1	23	11	20
Kontrol grubu 1	18	12	17

4.1.2.2 Üniversite Öncesi GUI-Programlama Eğitimi Faktörü

GUI-uygulama geliştirme eğitimi bilgi formu kullanılarak, öğrencilerin üniversite öncesi dönemde edinmiş olabilecekleri GUI-uygulama geliştirme bilgisi açığa çıkarılmaya çalışılmıştır. Bu formda belirtilen beyanlara göre, deney grubu 1 ve kontrol grubu 1 öğrencilerinden belli bir kısmının üniversite öncesinde okudukları lisede bilgisayar programlama eğitimi aldığı ve lisede programlama eğitimi almış bu öğrencilerin bir kısmının ise, lisedeki programlama derslerinde nesne yönelimli programlama yaklaşımıyla GUI-uygulama geliştirme eğitimi aldığı anlaşılmıştır. En nihayetinde uygulanan Ön bilgi Testi 2 sayesinde, lise eğitimi boyunca GUI-uygulama geliştirme eğitimi almış öğrencilerin bu konudaki bilgi seviyesi ölçülmüştür. Testte ortalamanın üzerinde kalan öğrenciler, Üniversite öncesi GUI-uygulama geliştirme

eğitimi faktörü açısından eğitilmiş faktör grubuna tahsis edilirken, lise döneminde GUI-uygulama geliştirme eğitimi almamış öğrenciler eğitim almamış faktör grubuna tahsis edilmiştir. Bu uygulamanın ayrıntılı gösterimi Şekil 4.2’de verilmiştir.



Şekil 4.2 Üniversite Öncesi GUI Programlama Eğitimi Faktör Gruplarının Oluşturulma Şeması

4.1.2.3 Faktör gruplarının son görünümü

Programlama Dilleri 1 dersi başarıları ve Üniversite Öncesi GUI-Uygulama Geliştirme Eğitimi faktörlerine göre öğrencilerin tahsis edildiği grupların nihai görünümü Tablo 4.4’te görüldüğü şekildedir.

Tablo 4.4 Faktör gruplarının son görünümü

Deney Grubu 1		Üniversite Öncesi GUI-Programlama Eğitimi	
		Eğitim almamış	Eğitimli
PD1 Dersi Başarısı	Başarısız	12	3
	Başarılı	4	11

Kontrol grubu 1		Üniversite Öncesi GUI-Programlama Eğitimi	
		Eğitim Almamış	Eğitimli
PD1 Dersi Başarısı	Başarısız	12	6
	Başarılı	3	10

Buna göre, öğrencilerin üniversite öncesinde GUI programlama eğitimi almamış olan ve Programlama Dilleri 1 dersi konuları açısından başarısız bulunmuş alt grupları, “Acemi” ve üniversite öncesinde GUI programlama eğitimi almış olan ve Programlama Dilleri 1 dersi konuları açısından başarılı bulunmuş alt grupları da “Deneyimli” alt grupları oluşturacak şekilde tahsis edilmiştir. Bu dağılım Tablo 4.5’te gösterilmiştir.

Tablo 4.5 Deney gruplarında GUI-programlama açısından “Deneyimli” ve “Acemi” alt gruplar

	Alt Grup	
	Acemi (Lisede eğitim almamış x PD1 Başarısız)	Deneyimli (Lisede eğitim almış x PD1 Başarılı)
Deney Grubu 1	12	11
Kontrol grubu 1	12	10

4.2 Akademik Başarıya Yönelik Hipotezlere Ait Bulgular

4.2.1 Kullanılan Aracın GUI Programlama Akademik Başarısına Etkisi

Hipotez 1.1’in test edilebilmesi için, deney gruplarının Uygulama Sınavı 1’de elde ettiği skorların kıyaslanması öngörülmüştür. Bu maksatla kullanılmak istenen Bağımsız Örneklem t-testi için gereken normal dağılım şartının sağlanıp sağlanmadığını sınavan Shapiro-Wilk testi yürütülmüştür. Verilerin normal dağılım gösterdiğinden emin

olunmasının ardından (Deney Grubu 1, $p = .22$; Kontrol grubu 1, $p = .19$) yürütülen bağımsız örneklem t-testi sonuçları Tablo 4.6’da verilmiştir.

Tablo 4.6 Kullanılan araca göre Uygulama Sınavı 1 ortalamalarında farka ilişkin t-testi sonuçları

Gruplar	N	$\bar{x}$	S	t-testi		
				sd	t	p
Deney Grubu 1	54	37,96	18,23	99	,711	.48
Kontrol grubu 1	47	41,06	25,38			

Buna göre, eğitimde kullanılan araç son test puanları üzerinde istatistiki açıdan anlamlı bir etki yaratmamaktadır $t(99) = .71$, $p > .05$.

4.2.2 Kullanılan Araç ve Üniversite Öncesi GUI Programlama Eğitimi Faktörlerinin Akademik Başarıya Etkileri Arasında Etkileşim

Hipotez 1.2’nin test edilebilmesi için Uygulama Sınavı 1’de elde edilen puanları etkilediği düşünülen, kullanılan araç ve üniversite öncesi GUI programlama eğitimi faktörleri arasındaki etkileşimi inceleyen 2x2 faktöriyel varyans analizi gerçekleştirilmiştir. Bu analizin gereksinimlerini karşılamak adına yürütülmüş normal dağılım testi sonuçları Tablo 4.7’de verilmiştir.

Tablo 4.7 Üniversite Öncesi GUI-Programlama Eğitimi ve Kullanılan Araç Faktörlerinin Uygulama Sınavı 1 Sonuçlarına etkileri arasında etkileşim - Betimsel İstatistikler ve Normal Dağılım Testleri

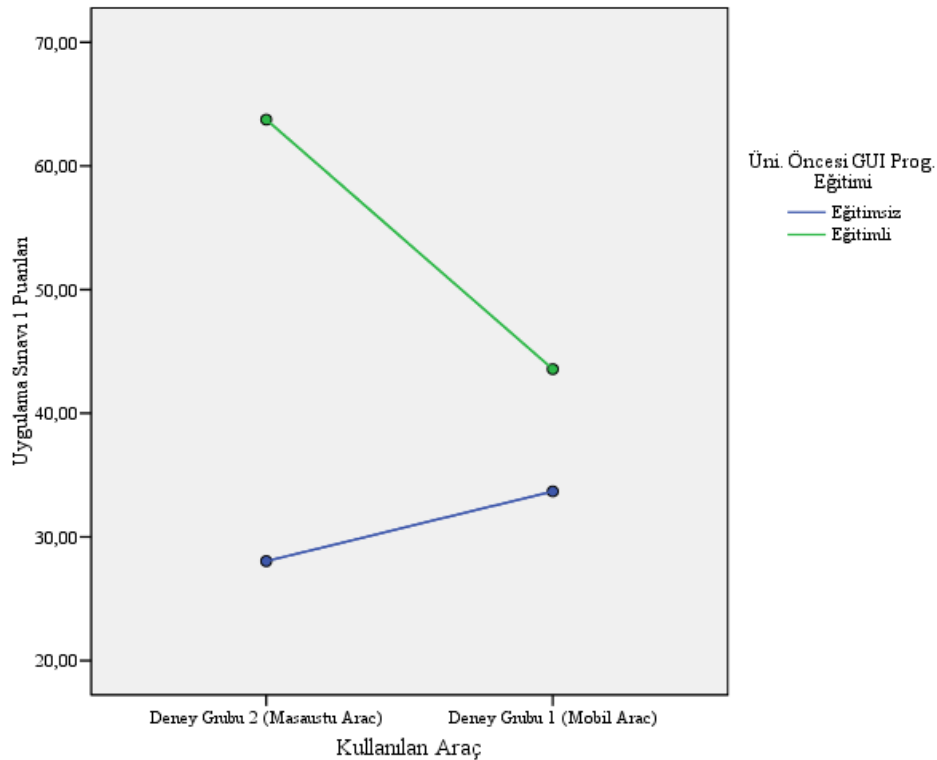
Faktör grupları		n	$\bar{x}$	ss	Shapiro-Wilk		
					İstatistik	sd	p
GUI-programlama eğitimine göre	Deney Grubu 1, GUI Eğitimi Almamış	19	33,68	18,91	,953	19	,45
	Deney Grubu 1, GUI Eğitimli	21	43,57	17,18	,927	21	,06
	Kontrol grubu 1, GUI Eğitimi Almamış	23	28,04	17,69	,955	23	,38
	Kontrol grubu 1, GUI Eğitimli	16	63,75	21,79	,969	16	,83

Hata varyansları homojen dağılan (Levene Hata Varyansları Eşitlik Testi $p > .05$) ve normal dağılım gösterdiğinden emin olunan veriler 2x2 faktöriyel varyans analizine tabi tutulmuş ve gruplar arası etkilere ait test verileri ise tablo 4-8’de sunulmuştur.

Tablo 4.8 Araç ve Üni. Öncesi GUI Programlama Eğitimi faktörlerinin Uygulama Sınavı 1 sonuçlarına etkileri arasında etkileşim - 2x2 Faktöriyel Varyans Analizi Sonuçları, Gruplar Arası Etkiler

Etkili Faktörler	Varyansın kaynağı	Kareler Toplamı	sd	Kareler Ortalaması	F	p
Araç	Araç	1024,82	1	1024,82	2,92	,09
	GUI-D	10080,01	1	10080,01	28,68	,00
Üni. Öncesi GUI-Programlama Eğitimi	AxG	3232,50	1	3232,50	9,20	,00
	Hata	26361,20	75	351,48		
	Toplam	39654,43	78			

Bu analize göre kullanılan araç ve üniversite öncesi GUI programlama eğitimi faktörlerinin GUI-programlama akademik başarısına etkileri arasında ($F(1,75) = 9,18$, $p = .00$) anlamlı bir iki yönlü ilişki vardır. Bunun yanı sıra; beklendiği üzere, üniversite öncesi GUI programlama eğitimi almanın, üniversitede alınan GUI-programlama eğitimi akademik başarısı üzerinde tek başına anlamlı ana etkiye sahip olduğu görülmüştür ($F(1,75) = 28,68$ $p = .000$). Testa ait profil grafiği Grafik 4-1 'de verilmiştir.



Grafik 4-1 Araç ve Üni. Öncesi GUI Programlama Eğitimi faktörlerinin Uygulama Sınavı 1 sonuçlarına etkileri arasında etkileşim - 2x2 Faktöriyel Varyans Analizi Sonuçları, Profil Grafiği

Grafikten anlaşıldığı üzere, Üniversite öncesi GUI programlama deneyimi faktörü ile kullanılan araç faktörleri arasında bir etkileşim bulunmakta olup; bu eğitimi almış öğrenciler MTPD ile daha başarılı olurken, eğitimi almamış öğrenciler MBTGPD ile nispeten daha başarılıdır.

4.2.3 Kullanılan Araç ve Programlama Dilleri 1 Dersi Başarısı Faktörlerinin Akademik Başarıya Etkileri Arasında Etkileşim

Hipotez 1.3'ün test edilebilmesi için Uygulama Sınavı 1'de elde edilen puanları etkilediği düşünülen, kullanılan araç ve Programlama Dilleri 1 dersi başarısı faktörleri arasındaki etkileşimi inceleyen 2x2 faktöriyel varyans analizi gerçekleştirilmiştir. Bu analizin gereksinimlerini karşılamak adına yürütülmüş normal dağılım testi sonuçları Tablo 4-9'de verilmiştir.

Tablo 4.9 Programlama Dilleri 1 Dersi Başarısı ve Kullanılan Araç Faktörlerinin Uygulama Sınavı 1 Sonuçlarına etkileri arasında etkileşim - Betimsel İstatistikler ve Normal Dağılım Testleri

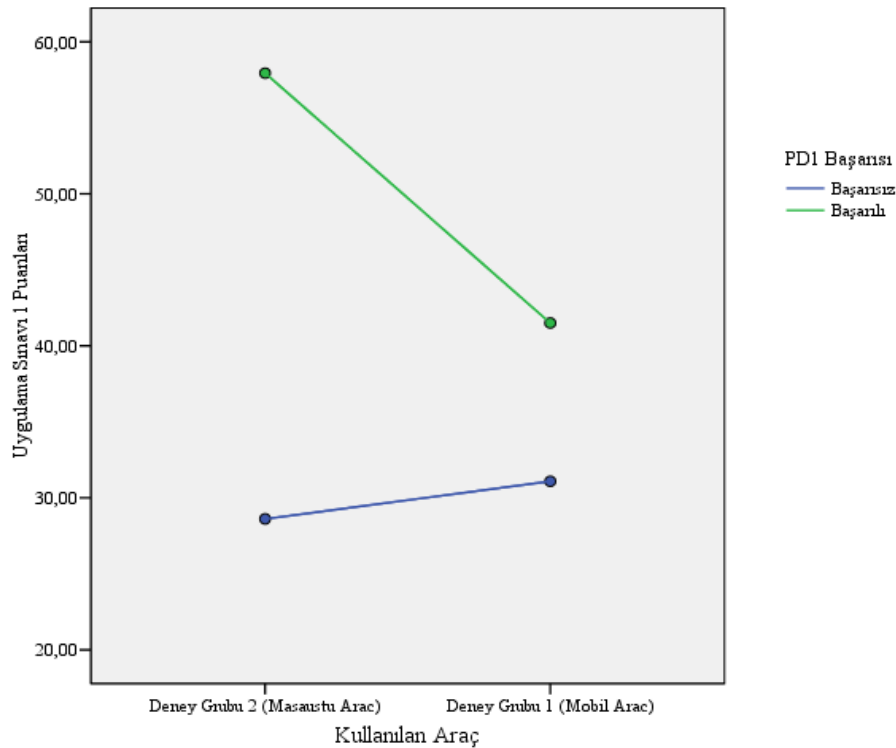
Faktör grupları		n	$\bar{x}$	ss	Shapiro-Wilk		
					İstatistik	sd	p
PD1-Başarısına göre	Deney Grubu 1, PD1 Başarısız	23	31,08	16,16	,929	23	,10
	Deney Grubu 1, PD1 Başarılı	20	41,50	19,60	,987	20	,99
	Kontrol grubu 1, PD1 Başarısız	18	28,61	17,13	,951	18	,44
	Kontrol grubu 1, PD1 Başarılı	17	57,94	24,24	,957	17	,57

Hata varyansları homojen dağılan (Levene Hata Varyansları Eşitlik Testi $p > ,05$) ve normal dağılım gösterdiğinden emin olunan veriler 2x2 faktöriyel varyans analizine tabi tutulmuş ve gruplar arası etkilere ait test verileri ise tablo 4-10'da sunulmuştur.

Tablo 4.10 Araç ve Programlama Dilleri 1 Dersi Başarısı faktörlerinin Uygulama Sınavı 1 sonuçlarına etkileri arasında etkileşim - 2x2 Faktöriyel Varyans Analizi Sonuçları, Gruplar Arası Etkiler

Etkili Faktörler	Varyansın kaynağı	Kareler Toplamı	sd	Kareler Ortalaması	F	p
Araç X PD1-Başarısı	Araç	938,29	1	938,29	2,53	,12
	PD1-Başarısı	7599,03	1	7599,03	20,49	,00
	AxP	1721,63	1	1721,63	4,64	,03
	Hata	27446,04	74	370,89		
	Toplam	37052,88	77			

Bu analize göre, kullanılan araç ve Programlama Dilleri 1 dersi başarısı faktörlerinin GUI-programlama akademik başarısına etkileri arasında anlamlı bir iki-yönlü ilişki vardır ($F(1,74) = 4,64$, $p = .03$). Ve beklendiği üzere, araç ve üniversitede alınan Programlama Dilleri 1 dersindeki başarı faktörünün, GUI-programlama akademik başarısı üzerinde tek başına anlamlı ana etkiye sahip olduğu tespit edilmiştir ($F(1,74) = 20,49$ $p = .000$). Testa ait profil grafiği Grafik 4-2’de verilmiştir.



Grafik 4-2 Araç ve Programlama Dilleri 1 dersi başarısı faktörlerinin Uygulama Sınavı 1 sonuçlarına etkileri arasında etkileşim - 2x2 Faktöriyel Varyans Analizi Sonuçları, Profil Grafiği

Grafikten anlaşıldığı üzere, Programlama Dilleri 1 dersi başarısı faktörü ile kullanılan araç faktörleri arasında bir etkileşim bulunmakta olup; PD1’de yüksek başarılı öğrenciler öğrenciler MTPD ile çalıştıklarında MBTGPD ile çalışan akranlarına göre daha başarılı olurken, PD1’de başarısız olmuş öğrenciler MBTGPD ile çalıştıklarında nispeten daha büyük başarı elde etmektedir.

4.2.4 Kullanılan Araç ve MTPD Kullanımında Deneyimli Olma Faktörleri Arasında Akademik Başarıya Etki Açısından Etkileşim

Üniversite öncesi GUI programlama eğitimi ve Programlama Dilleri 1 dersi başarıları faktörlerinin kesişimi ile, MTPD kullanımında deneyim faktörü oluşturulmuştur. Bu faktörün Uygulama Sınavı 1 skorları üzerindeki etkisi ile aynı skorlar üzerinde kullanılan aracın etkisi arasında etkileşim bulunup bulunmadığını sorgulayan Hipotez 1.4'ün test edilmesi için ise, 2x2 faktöriyel varyans analizi gerçekleştirilmiştir. Bu analizin gereksinimlerini karşılamak adına yürütülmüş normal dağılım testi sonuçları ve betimsel istatistikler Tablo 4-11'de verilmiştir.

Tablo 4.11 MTPD Kullanımında Deneyim ve Kullanılan Araç Faktörlerinin Uygulama Sınavı 1 Sonuçlarına etkileri arasında etkileşim - Betimsel İstatistikler ve Normal Dağılım Testleri

	Faktör grupları	n	$\bar{x}$	ss	Shapiro-Wilk		
					İstatistik	sd	p
MTPD Deneyimine göre	Deney Grubu 1, Deneyimli	11	46,36	16,74	,94	11	,58
	Deney Grubu 1, Deneyimsiz	12	34,58	19,70	,94	12	,53
	Kontrol grubu 1, Deneyimli	10	67,00	23,11	,92	10	,37
	Kontrol grubu 1, Deneyimsiz	12	20,00	11,87	,87	12	,06

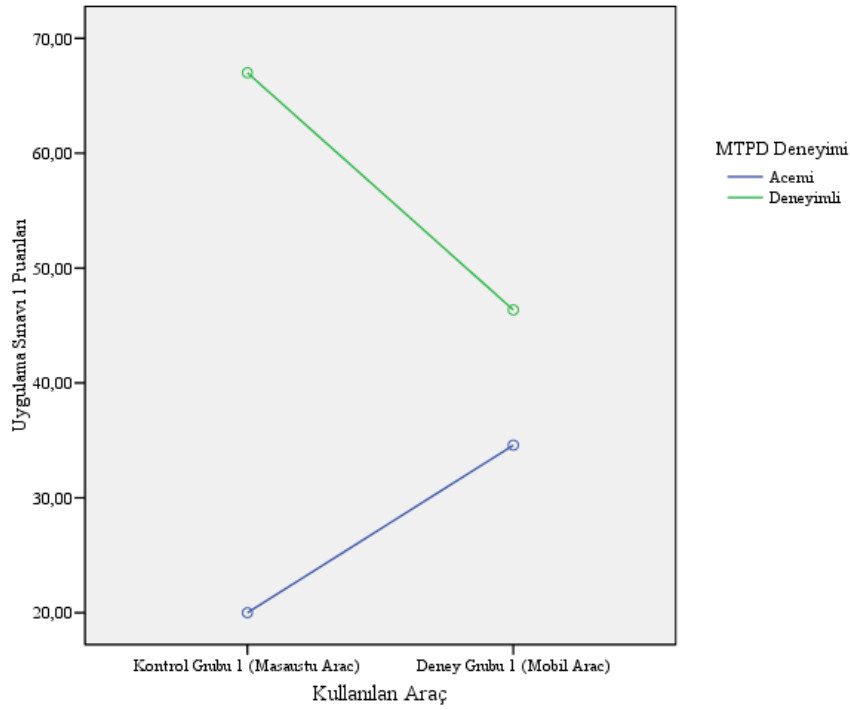
Hata varyansları homojen dağılan (Levene Hata Varyansları Eşitlik Testi $p > ,05$) ve normal dağılım gösterdiğinden emin olunan veriler 2x2 faktöriyel varyans analizine tabi tutulmuş ve gruplar arası etkilere ait test verileri ise tablo 4-12'de sunulmuştur.

Tablo 4.12 Araç ve MTPD kullanımında deneyim faktörlerinin Uygulama Sınavı 1 sonuçlarına etkileri arasında etkileşim - 2x2 Faktöriyel Varyans Analizi Sonuçları, Gruplar Arası Etkiler

Etkili Faktörler	Varyansın kaynağı	Kareler Toplamı	sd	Kareler Ortalaması	F	p
Araç X Deneyim	Araç	102,46	1	102,46	,31	,57
	Deneyim	9662,63	1	9662,63	29,48	,00
	AxD	3468,99	1	3468,99	10,58	,00
	Hata	13437,46	41	327,74		
	Toplam	26297,77	44			

Bu analize göre, kullanılan araç ve öğrencinin MTPD ile deneyimi faktörlerinin GUI-programlama akademik başarısına etkileri arasında anlamlı bir iki-yönlü ilişki vardır

($F(1,44) = 10,58$, $p = .00$). Deneyim faktörünün, GUI-programlama akademik başarısı üzerinde tek başına anlamlı ana etkiye sahip olduğu da görülmektedir ($F(1,44) = 29,48$ $p = .00$). Sonuçların yorumlanabilmesi açısından çizilmiş profil grafiği, Grafik 'da verilmiştir.



Grafik 4-3 Araç ve MTPD kullanımında deneyim faktörlerinin Uygulama Sınavı 1 sonuçlarına etkileri arasında etkileşim - 2x2 Faktöriyel Varyans Analizi Sonuçları, Profil Grafiği

Grafikten anlaşıldığı üzere, deneyim faktörü ile kullanılan araç faktörleri arasında bir etkileşim bulunmakta olup; deneyimli öğrenciler MTPD ile daha başarılı olurken, deneyimsiz öğrenciler MBTGPD ile daha başarılıdır. Bu grupların t-testi ile istatistiki olarak karşılaştırılmaları ise Tablo 4-13'te verilmiştir.

Tablo 4.13 Kullanılan araca göre acemi ve deneyimli öğrencilerde başarı ortalamalarının farkına ilişkin t-testi sonuçları

Gruplar	n	$\bar{x}$	ss	t-testi		
				sd	t	p
Deney Grubu 1 Acemi	12	34,58	19,70	22	-2,20	.039
Kontrol grubu 1 Acemi	12	20,00	11,87			
Gruplar	n	$\bar{x}$	ss	t-testi		
				sd	t	p
Deney Grubu 1 Deneyimli	10	67,00	23,11	19	2,35	.03
Kontrol grubu 1 Deneyimli	11	46,36	16,74			

Buna göre, Uygulama Sınavı 1 puanları açısından öğrenci alt grupları arasında görülen farklar hem acemi ($t(22) = -2.20$, $p < .05$) hem deneyimli ($t(19) = 2,35$, $p < .05$) öğrenciler için istatistiki önem taşımaktadır.

4.2.5 Çapraz Uygulama Sonunda Nihai Akademik Başarı

Hipotez 1.5'in test edilebilmesi için, deney gruplarındaki tüm öğrencilere ait Uygulama Sınavı 2 sonuçlarının karşılaştırılması öngörülmüştür. Bu maksatla yürütülmesi planlanan Bağımsız örneklem t-testi için normal dağılım koşullarının sağlanıp sağlanmadığını sınavan Shapiro-Wilk normallik testleri yürütülmüştür. Buna göre Kontrol grubu 1 skorlarında normallik şartı ihlal edildiğinden ötürü ($p = .03$) deney gruplarının sınav sonuçlarının karşılaştırılmasında Mann Whitney U non-parametrik testinden faydalanılmıştır ve sonuçlar Tablo 4-14'te verilmiştir.

Tablo 4.14 Kullanılan araç sırasına göre nihai başarı değişkeni için U-testi sonucu

Grup	n	Sıra Ortalaması	Sıra Toplamı	U	p
Deney Grubu 1 (Önce Mobil sonra Masaüstü)	54	51,56	2784,50	1238,50	.84
Kontrol grubu 1 Önce Masaüstü sonra Mobil	47	50,35	2366,50		

Buna göre, araçların kullanım sırası, GUI-programlama akademik başarısı üzerinde anlamlı bir etkiye sahip değildir $U=1238,50$ $p > .05$.

4.2.6 Çapraz Uygulama Sonunda Nihai Başarı: Üniversite Öncesinde GUI Programlama Eğitimi Almamış Öğrenciler

Hipotez 1.6'nın test edilebilmesi için, deney gruplarının üniversite öncesi GUI-uygulama geliştirme eğitimi almamış faktör gruplarında elde edilen Uygulama Sınavı 2 skorlarının karşılaştırılması öngörülmüştür. Bu amaçla yürütülmesi düşünülen t-testine yönelik normal dağılım koşullarının sağlanıp sağlanmadığını sınavan Shapiro-Wilk normallik testleri yürütülmüştür. Buna göre Kontrol grubu 1'ye ait faktör alt grubu skorlarında normallik şartı ihlal edildiğinden ötürü ($p = .01$) deney gruplarının sınav sonuçlarının karşılaştırılmasında Mann Whitney U non-parametrik testinden faydalanılmıştır ve sonuçlar Tablo 4-15'te verilmiştir.

Tablo 4.15 Kullanılan araç sırasına göre nihai başarı değişkeni için U-testi sonucu

Grup	n	Sıra Ortalaması	Sıra Toplamı	U	p
Deney Grubu 1, GUI-Eğitimsiz (Önce Mobil sonra Masaüstü)	19	21,29	404,50	214,50	.92
Kontrol grubu 1, GUI-Eğitimsiz (Önce Masaüstü sonra Mobil)	23	21,67	498,50		

Buna göre, araçların kullanım sırasının, üniversite öncesinde GUI programlama eğitimi bulunmayan öğrenci alt gruplarında GUI programlama akademik başarı üzerinde anlamlı bir etkisi yoktur ($U=214,50$ $p>.05$).

4.3 Motivasyona yönelik hipotezlere ait bulgular

4.3.1 Kullanılan Aracın Motivasyona Etkisi (Hipotez 2.1)

Hipotez 2.1'in test edilebilmesi için, deney gruplarına ait Motivasyon Testi 1. Ölçüm İçsel Hedef Düzenleme, Görev Değeri ve Özyeterlilik alt boyutları skorlarının tek tek kıyaslanması öngörülmüştür. Bu amaçla yürütülecek Bağımsız Örneklem t-testleri'ne yönelik normal dağılım şartlarının sağlanıp sağlanmadığını sınavan Shapiro-Wilk testleri yürütülmüştür. Bu verilere göre deney gruplarının motivasyon alt boyutlarına yönelik ölçümlerinde veriler normal dağılım sergilemektedir (DG1; İHY $p = .11$, GD $p = .33$, ÖY $p = .37$; KG1; İHY $p = .69$, GD $p = .21$, ÖY $p = .79$). Bunun neticesinde yürütülen Bağımsız Örneklem t-testlerine ilişkin sonuçlar Tablo 4-16'de verilmiştir.

Tablo 4.16 Kullanılan Aracın Motivasyon Ölçeği 1. Ölçüm Alt Boyutlarına Etkisine ilişkin t-testi sonuçları

		n	$\bar{x}$	ss	t-testi		
					sd	t	p
İçsel Hedefe Yönelim Alt Boyutu	Deney Grubu 1	54	5,23	1,12	99	-1,572	.12
	Kontrol grubu 1	47	5,00	1,01			
Görev Değeri Alt Boyutu	Deney Grubu 1	54	4,77	,89	99	-,332	.74
	Kontrol grubu 1	47	4,70	1,09			
Özyeterlilik Alt Boyutu	Deney Grubu 1	54	3,97	1,19	99	1,658	,10
	Kontrol grubu 1	47	4,37	1,20			

Buna göre, Motivasyon Ölçeği İçsel Hedef Düzenleme alt boyutu puanları ($t(99) = -1.572$, $p > .05$), Görev Değeri alt boyutu puanları ($t(99) = -.332$, $p > .05$) ve Özyeterlilik alt boyutu puanları ($t(99) = 1.658$, $p > .05$) açısından deney ve kontrol grupları arasında anlamlı bir fark yoktur.

4.3.2 Kullanılan Araç ve Üniversite Öncesi GUI Programlama Eğitimi Faktörlerinin Motivasyona Etkileri Arasında Etkileşim

Hipotez 2.2'nin test edilebilmesi için, Motivasyon Ölçeği Birinci Ölçüm İçsel Hedef Düzenleme, Görev Değeri ve Özyeterlilik alt boyutlarından elde edilen puanlar üzerinde kullanılan araç ve üniversite öncesi GUI programlama eğitimi faktörlerinin etkileşimini inceleyen üç adet 2x2 faktöriyel varyans analizi yürütülmüştür. Bu veriler için gerçekleştirilen normal dağılım testleri sonuçları Tablo 4-17'de verilmiştir.

Tablo 4.17 Motivasyon Testi 1. ölçüm içsel hedef düzenleme, görev değeri ve özyeterlilik alt boyutlarında Deney Gruplarının üni. Öncesi GUI-eğitimi faktör gruplarına ait skorlarda betimleyici istatistikler ve normal dağılım testleri

		Gruplar (GUI Progr. Eğitimi faktörü)	n	$\bar{x}$	ss	Shapiro-Wilk		
						İstatistik	sd	p
İçsel Hedefe Yönelim	Deney Grubu 1, GUI Eğitimi Almamış		19	5,29	,65	,931	19	,18
	Deney Grubu 1, GUI Eğitimli		21	5,31	1,04	,949	21	,32
	Kontrol grubu 1, GUI Eğitimi Almamış		23	5,04	,83	,971	23	,72
	Kontrol grubu 1, GUI Eğitimli		16	4,81	1,33	,963	16	,71
Görev Değeri	Deney Grubu 1, GUI Eğitimi Almamış		19	3,74	1,44	,954	19	,45
	Deney Grubu 1, GUI Eğitimli		21	4,14	1,10	,945	21	,28
	Kontrol grubu 1, GUI Eğitimi Almamış		23	4,58	1,23	,983	23	,95
	Kontrol grubu 1, GUI Eğitimli		16	4,73	1,09	,890	16	,06
Özyeterlilik	Deney Grubu 1, GUI Eğitimi Almamış		19	3,43	1,04	,969	19	,76
	Deney Grubu 1, GUI Eğitimli		21	4,51	1,07	,945	21	,27
	Kontrol grubu 1, GUI Eğitimi Almamış		23	4,06	1,02	,971	23	,72
	Kontrol grubu 1, GUI Eğitimli		16	4,96	1,33	,948	16	,46

Her biri Levene Hata Varyansları Eşitlik testini geçen ($p>.05$) ve normal dağılım gösterdiği anlaşılan veriler neticede üç adet 2x2 faktöriyel varyans analizine tabi tutulmuştur ve bu analizlerde gruplar arası etkilere ait test verileri tablo 4-18’de verilmiştir.

Tablo 4.18 Kullanılan araç x Üni. Öncesi GUI-eğitimi faktörlerinin motivasyon alt boyutları üzerindeki etkilerini inceleyen 2x2 Faktöriyel Varyans Analizleri Sonuçları – Gruplar Arası Etkiler

Motivasyon Altboyutu	Varyansın kaynağı	Kareler Toplamı	sd	Kareler Ortalaması	F	p
İçsel Hedef Düzenleme	Araç	2,816	1	2,816	2,944	,09
	GUI-D.	,218	1	,218	,228	,64
	AxG	,303	1	,303	,317	,58
	Hata	71,751	75	,957		
	Toplam	74,829	78			
Görev Değeri	Araç	9,919	1	9,919	6,540	,01
	GUI-D	1,467	1	1,467	,968	,33
	AxG	,329	1	,329	,217	,64
	Hata	113,751	75	1,517		
	Toplam	125,043	78			
Özyeterlilik	Araç	5,719	1	5,719	4,647	,03
	GUI-D.	18,970	1	18,970	15,413	,00
	AxG	,171	1	,171	,139	,71
	Hata	92,306	75	1,231		
	Toplam	115,308	78			

Bu analizlere göre, kullanılan araç ve üniversite öncesi GUI programlama eğitimi faktörlerinin derse yönelik içsel hedef düzenleme algısına etkileri arasında ($F(1,75) = ,32$, $p > .05$), görev değeri algısına etkileri arasında ($F(1,75) = ,217$, $p > .05$) ve özyeterlilik algısına etkileri arasında ($F(1,75) = .139$, $p > .05$) anlamlı iki yönlü ilişkiler bulunamamıştır.

4.3.3 Kullanılan Araç ve Programlama Dilleri 1 Başarısı Faktörlerinin Motivasyona Etkileri Arasında Etkileşim

Hipotez 2.3’ün test edilebilmesi için, Motivasyon Ölçeği Birinci Ölçüm İçsel Hedef Düzenleme, Görev Değeri ve Özyeterlilik alt boyutlarından elde edilen puanlar üzerinde kullanılan araç ve Programlama Dilleri 1 dersi başarısı faktörlerinin etkileşimini inceleyen üç adet 2x2 faktöriyel varyans analizi yürütülmüştür. Bu veriler için gerçekleştirilen normal dağılım testleri sonuçları Tablo 4-19’da verilmiştir.

Tablo 4.19 Motivasyon Testi 1. ölçüm içsel hedef düzenleme, görev değeri ve özyeterlilik alt boyutlarında Deney Gruplarının PD1 Başarı faktör gruplarına ait skorlarında betimleyici istatistikler ve normal dağılım testleri

	Gruplar (PD1 Dersi Başarısı Faktörü)	n	$\bar{x}$	ss	Shapiro-Wilk		
					İstatistik	sd	p
İçsel Hedefe Yönelim	Deney Grubu 1, PD1 Başarısız	23	5,12	,94	,972	23	,73
	Deney Grubu 1, PD1 Başarılı	20	5,28	,85	,954	20	,43
	Kontrol grubu 1, PD1 Başarısız	18	4,79	,88	,956	18	,54
	Kontrol grubu 1, PD1 Başarılı	17	5,42	1,04	,956	17	,56
Görev Değeri	Deney Grubu 1, PD1 Başarısız	23	4,44	,92	,925	23	,08
	Deney Grubu 1, PD1 Başarılı	20	5,03	,85	,912	20	,06
	Kontrol grubu 1, PD1 Başarısız	18	4,39	1,22	,973	18	,84
	Kontrol grubu 1, PD1 Başarılı	17	5,06	1,06	,939	17	,31
Özyeterlilik	Deney Grubu 1, PD1 Başarısız	23	3,65	1,04	,971	23	,70
	Deney Grubu 1, PD1 Başarılı	20	4,32	1,12	,975	20	,85
	Kontrol grubu 1, PD1 Başarısız	18	3,85	1,06	,966	18	,71
	Kontrol grubu 1, PD1 Başarılı	17	5,00	1,11	,923	17	,17

Her biri Levene Hata Varyansları Eşitlik testini geçen ($p>,05$) ve normal dağılım gösterdiği anlaşılan veriler neticede üç adet 2x2 faktöriyel varyans analizine tabi tutulmuştur ve bu analizlerde gruplar arası etkilere ait test verileri tablo 4-20’de verilmiştir

Tablo 4.20 Kullanılan araç x PD1-başarısı faktörlerinin motivasyon alt boyutları üzerindeki etkilerini inceleyen 2x2 Faktöriyel Varyans Analizleri Sonuçları – Gruplar Arası Etkiler

Motivasyon Altboyutu	Varyansın kaynağı	Kareler Toplamı	sd	Kareler Ortalaması	F	p
İçsel Hedef Düzenleme	Araç	,178	1	,178	,205	,65
	PD1-B	3,01	1	3,013	3,469	,07
	AxP	1,10	1	1,100	1,267	,264
	Hata	64,25	74	,868		
	Toplam	68,23	77			
Görev Değeri	Araç	,002	1	,002	,002	,97
	PD1-B	7,59	1	7,598	7,332	,01
	AxP	,039	1	,039	,037	,85
	Hata	76,68	74	1,036		
	Toplam	84,29	77			
Özyeterlilik	Araç	3,79	1	3,792	3,215	,08
	PD1-B	16,03	1	16,037	13,597	,00
	AxP	1,11	1	1,118	,947	,33
	Hata	87,28	74	1,179		
	Toplam	107,67	77			

Bu analizlere göre, kullanılan araç ve Programlama Dilleri 1 dersi başarısı faktörlerinin derse yönelik içsel hedef düzenleme algısına etkileri arasında ($F(1,74) = 1.267, p > .05$), görev değeri algısına etkileri arasında ($F(1,74) = .037, p > .05$) ve özyeterlilik algısına etkileri arasında ($F(1,74) = .947, p > .05$) herhangi bir anlamlı iki yönlü ilişki tespit edilmemiştir.

4.3.4 Kullanılan Araç ve MTPD Kullanımında Deneyimli Olma Faktörleri Arasında Motivasyona Etki Açısından Etkileşim

Hipotez 2.3'ün test edilebilmesi için, acemi öğrencilerin Motivasyon Testi 1. Ölçüm İçsel Hedef Düzenleme, Görev Değeri ve Özyeterlilik Alt Boyutu skorlarına etki ettiği düşünülen MTPD kullanımında deneyimli olma ve kullanılan araç faktörleri arasında etkileşim bulunup bulunmadığı sorusu; her bağımlı değişken için ayrı ayrı gerçekleştirilmiş üç adet 2x2 faktöriyel varyans analizi kullanılmıştır. Grup skorlarında normallik koşullarının sağlanıp sağlanmadığını ölçmek için Shapiro-Wilk testi kullanılmıştır. Bu bağlamda yürütülmesi planlanan faktöriyel varyans analizine ait betimsel istatistikler ve normal dağılım testi sonuçları Tablo 4-21'de verilmiştir.

Tablo 4.21 Motivasyon Testi 1. ölçüm içsel hedef düzenleme, görev değeri ve özyeterlilik alt boyutlarında Deney Gruplarının MTPD Kullanımında deneyim faktör gruplarına ait skorlarında betimleyici istatistikler ve normal dağılım testleri

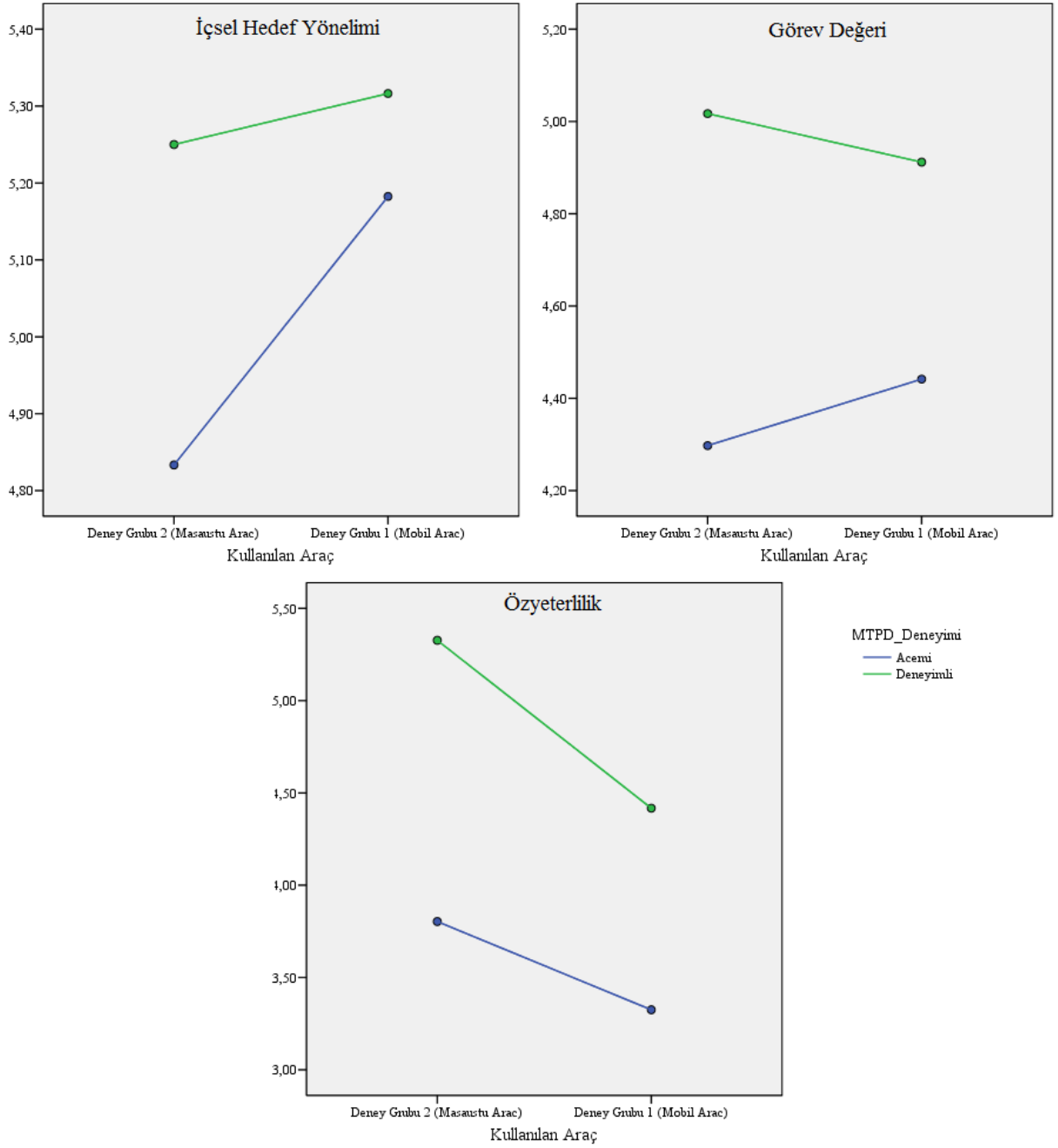
	Gruplar (PD1 Dersi Başarısı Faktörü)	n	$\bar{x}$	ss	Shapiro-Wilk		
					İstatistik	sd	p
İçsel Hedefe Yönelim	Deney Grubu 1, PD1 Acemi	12	5,18	,75	,927	12	,35
	Deney Grubu 1, PD1 Deneyimli	11	5,31	1,15	,963	11	,80
	Kontrol grubu 1, PD1 Acemi	12	4,83	,98	,951	12	,64
	Kontrol grubu 1, PD1 Deneyimli	10	5,25	1,30	,922	10	,37
Görev Değeri	Deney Grubu 1, PD1 Acemi	12	4,44	,84	,974	12	,94
	Deney Grubu 1, PD1 Deneyimli	11	4,91	,97	,919	11	,30
	Kontrol grubu 1, PD1 Acemi	12	4,29	1,33	,978	12	,97
	Kontrol grubu 1, PD1 Deneyimli	10	5,01	1,21	,887	10	,15
Özyeterlilik	Deney Grubu 1, PD1 Acemi	12	3,32	,94	,962	12	,80
	Deney Grubu 1, PD1 Deneyimli	11	4,41	1,09	,947	11	,61
	Kontrol grubu 1, PD1 Acemi	12	3,80	,96	,917	12	,26
	Kontrol grubu 1, PD1 Deneyimli	10	5,32	,99	,931	10	,46

Her biri Levene Hata Varyansları Eşitlik testini geçen ($p>.05$) ve normal dağılım gösterdiği anlaşılan veriler neticede üç adet 2x2 faktöriyel varyans analizine tabi tutulmuştur ve bu analizlerde gruplar arası etkilere ait test verileri Tablo 4-22’de, profil grafikleri Grafik 4-4’te verilmiştir

Tablo 4.22 Kullanılan araç x MTPD kullanımında deneyim faktörlerinin motivasyon alt boyutları üzerindeki etkilerini inceleyen 2x2 Faktöriyel Varyans Analizleri Sonuçları – Gruplar Arası Etkiler

Motivasyon Altboyutu	Varyansın kaynağı	Kareler Toplamı	sd	Kareler Ortalaması	F	p
İçsel Hedef Düzenleme	Araç	,483	1	,483	,436	,51
	MTPD Deneyim	,848	1	,848	,765	,38
	AxD	,224	1	,224	,202	,65
	Hata	45,440	41	1,108		
	Toplam	47,053	44			
Görev Değeri	Araç	,004	1	,004	,003	,95
	MTPD Deneyim	3,958	1	3,958	3,233	,08
	AxD	,174	1	,174	,142	,70
	Hata	50,188	41	1,224		
	Toplam	54,300	44			
Özyeterlilik	Araç	5,388	1	5,388	5,368	,02
	MTPD Deneyim	19,138	1	19,138	19,067	,00
	AxD	,520	1	,520	,519	,47
	Hata	41,152	41	1,004		
	Toplam	65,391	44			

Bu analizlere göre, kullanılan araç ve öğrencinin MTPD ile deneyimi faktörlerinin derse yönelik içsel hedef düzenleme algısına etkileri arasında ($F(1,44) = .202, p > .05$), görev değeri algısına etkileri arasında $F(1,44) = .142, p > .05$ ve özyeterlilik algısına etkileri arasında ($F(1,44) = .519, p>.05$) herhangi bir anlamlı iki yönlü ilişki tespit edilmemiştir.



Grafik 4-4 Kullanılan araç x MTPD kullanımında deneyim faktörlerinin motivasyon alt boyutları üzerindeki etkilerini inceleyen 2x2 Faktöriyel Varyans Analizleri Sonuçları – Profil Grafikleri

4.3.5 Aracın Değiştirilmesi Durumunda Motivasyonda Değişim: Deneyimli Öğrenciler

Hipotez 2.5'in test edilebilmesi için, deneyimli öğrencilerde; ikinci aracın kullanımı esnasında yürütülmüş Motivasyon Ölçeği İkinci Ölçümü'nde içsel hedef düzenleme, görev değeri ve özyeterlilik alt boyutu skorları incelenmiştir. Bu bağlamda Motivasyon Ölçeği Birinci Ölçümü skorlarının kovaryant ve kullanılan araç sırasının faktör olarak belirlendiği kovaryans analizleri kullanılmıştır. Analizlere konu olan kovaryant ve

bağımlı değişkenlere ait betimleyici istatistikler ve normallik testleri Tablo 4-23'te verilmiştir.

Tablo 4.23 Deneyimli Öğrencilerde Motivasyon Testi 1. ve 2. Ölçüm alt boyutu puanları – betimleyici istatistikler ve normallik testi sonuçları

Gruplar		1. Ölçüm						2. Ölçüm					
		n	$\bar{x}$	ss	Shapiro-Wilk			n	$\bar{x}$	ss	Shapiro-Wilk		
					İst.	sd	p				İst.	sd	p
İçsel Hedef Düzenleme	Deney Grubu 1 Deneyimli	11	5,25	1,30	,963	11	,81	11	5,40	1,03	,966	11	,85
	Kontrol grubu 1 Deneyimli	10	5,31	1,15	,922	10	,37	10	5,00	1,35	,882	10	,14
Görev Değeri	Deney Grubu 1 Deneyimli	11	4,91	,97	,919	11	,31	11	5,18	,93	,936	11	,48
	Kontrol grubu 1 Deneyimli	10	5,01	1,21	,887	10	,16	10	4,93	1,20	,869	10	,10
Özyeterlilik	Deney Grubu 1 Deneyimli	11	4,16	1,05	,942	11	,55	11	4,63	1,19	,964	11	,82
	Kontrol grubu 1 Deneyimli	10	5,32	,99	,931	10	,46	10	4,91	1,14	,928	10	,43

Bu verilere göre deney gruplarına mensup faktör gruplarının skor dağılımlarında normallik şartı sağlanmaktadır. Buna göre yürütülen kovaryans analizlerinde (Levene Hata Varyansları Eşitlik Testleri $p > ,05$) bağımlı değişkenlere yönelik gruplar arası etkiler testleri Tablo 4-24'te verilmiştir.

Tablo 4.24 Deneyimli Öğrencilerde Motivasyon Testi 2. Ölçümü Puanlarına Yönelik Kovaryans Analizi Sonuçları – Gruplar Arası Etkiler Testleri (Kovaryant = 1. Ölçüm)

Varyansın kaynağı		Kareler Toplamı	sd	Kareler Ortalaması	F	p
İçsel Hedef Düzenleme	1. Ölçüm	16,486	1	16,486	27,166	,00
	Araç Değişimi	,673	1	,673	1,109	,31
	Hata	10,923	18	,607		
	Toplam	28,286	20			
Görev Değeri	1. Ölçüm	12,927	1	12,927	26,202	,00
	Araç Değişimi	,558	1	,558	1,131	,30
	Hata	8,880	18	,493		
	Toplam	22,129	20			
Özyeterlilik	1. Ölçüm	6,207	1	6,207	5,586	,03
	Araç Değişimi	,527	1	,527	,474	,50
	Hata	20,000	18	1,111		
	Toplam	26,608	20	26,608		

Bu analize göre, deneyimli öğrenciler için, Motivasyon Ölçeği 1. Ölçüm puanları istatistiksel olarak kontrol altına alındığında, derste kullanılan aracın değiştirilmesi 2. Ölçüme ait İçsel Hedef Düzenleme ($F(1,18) = 1.109, p > .05$), Görev değeri ($F(1,18) =$

1,131, $p > .05$) ve Özyeterlilik ($F(1,18) =$, $p > .05$) alt boyutu skorları üzerinde anlamlı bir etki yaratmamaktadır.

4.3.6 Aracın Değiştirilmesi Durumunda Motivasyonda Değişim: Acemi Öğrenciler

Hipotez 2.6'nın test edilebilmesi için, acemi öğrencilerde ikinci aracın kullanımı esnasında yürütülmüş Motivasyon Ölçeği 2. Ölçümü içsel hedef düzenleme, görev değeri ve özyeterlilik alt boyutu skorları bağımlı değişken olarak incelenmiştir. Bu bağlamda, Motivasyon Ölçeği Birinci Ölçümü skorlarının kovaryant ve kullanılan araç sırasının faktör olarak kullanıldığı kovaryans analizleri kullanılmıştır. Analize konu olan kovaryant ve bağımlı değişkene ait betimleyici istatistikler ve normallik testleri Tablo 4-25'te sunulmuştur.

Tablo 4.25 Acemi Öğrencilerde Motivasyon Testi 1. ve 2. Ölçüm alt boyutu puanları – betimleyici istatistikler ve normallik testi sonuçları

Gruplar		1. Ölçüm						2. Ölçüm					
		n	$\bar{x}$	ss	Shapiro-Wilk			n	$\bar{x}$	ss	Shapiro-Wilk		
					İst.	sd	p				İst.	sd	p
İçsel Hedef Düzenleme	Deney Grubu 1 Acemi	12	4,83	,98	,927	12	,35	12	5,60	,95	,912	12	,23
	Kontrol grubu 1 Acemi	12	5,18	,75	,951	12	,65	12	4,44	1,18	,923	12	,31
Görev Değeri	Deney Grubu 1 Acemi	12	4,44	,84	,974	12	,95	12	4,72	1,04	,970	12	,91
	Kontrol grubu 1 Acemi	12	4,29	1,33	,978	12	,98	12	4,32	1,33	,920	12	,28
Özyeterlilik	Deney Grubu 1 Acemi	12	3,32	,94	,962	12	,81	12	3,90	1,24	,893	12	,13
	Kontrol grubu 1 Acemi	12	3,80	,96	,917	12	,26	12	3,83	1,37	,933	12	,42

Bu verilere göre deney gruplarına mensup faktör gruplarının skor dağılımlarında normallik şartı sağlanmaktadır. Buna göre yürütülen Kovaryans Analizinde (Levene Hata Varyansları Eşitlik Testleri $p > ,05$) gruplar arası etkiler Tablo 4-26'da gösterilmiştir.

Tablo 4.26 Acemi Öğrencilerde Motivasyon Testi 2. Ölçümü İlişkin Alt Boyutlarına Yönelik Kovaryans Analizi Sonuçları – Gruplar Arası Etkiler Testleri (Kovaryant = 1. Ölçüm)

	Varyansın kaynağı	Kareler Toplamı	sd	Kareler Ortalaması	F	p
İçsel Hedef Düzenleme	1. Ölçüm	3,316	1	3,316	3,126	,09
	Araç Değişimi	5,757	1	5,757	5,426	,03
	Hata	22,278	21	1,061		
	Toplam	33,598	23			
Görev Değeri	1. Ölçüm	8,024	1	8,024	7,135	,02
	Araç Değişimi	,600	1	,600	,533	,47
	Hata	23,616	21	1,125		
	Toplam	32,577	23			
Özyeterlilik	1. Ölçüm	11,297	1	11,297	8,934	,01
	Araç Değişimi	1,041	1	1,041	,823	,36
	Hata	26,556	21	1,265		
	Toplam	37,885	23			

Bu analize göre, 1. Ölçüm skorları kontrol altına alındığında, derste kullanılan aracın değiştirilmesinin, ilişkin alt gruplardaki öğrencilere ait 2. Ölçüm İçsel Hedef Düzenleme Altboyutu Skorları üzerinde anlamlı bir etkisi vardır ($F(1,21) =$, $p < .05$). Acemi öğrencilerin İçsel Hedef Düzenleme algısı, MTPD’den MBTGPD’ya geçen grupta azalırken, MBTGPD’den MTPD’ye geçen grupta artmıştır.

Diğer yandan, 1. Ölçüm skorları istatistiksel yolla kontrol altına alındığında, derste kullanılan aracın değiştirilmesinin, ilişkin alt gruplarda 2. Ölçüm Görev değeri ($F(1,21) = .533$, $p > .05$) ve Özyeterlilik ($F(1,21) = .823$, $p > .05$) algıları üzerinde anlamlı bir etkisi olmadığı görülmüştür.

4.4 Cihaz aidiyetine yönelik hipotezlere dair bulgular

4.4.1 Mobil araçla öğrenim gören öğrencilerde cihaz aidiyetinin derse yönelik görev değeri algısına etkisi

Hipotez 3.1’in test edilebilmesi için birinci uygulama dönemi boyunca mobil cihazla öğrenim gören öğrencilerin Motivasyon Testi 1. Ölçüm Görev Değeri alt boyutu skorlarının mobil cihaz aidiyeti faktörüne göre karşılaştırılması öngörülmüştür. Bu maksatla kullanılmak istenen Bağımsız Örneklem t-testi için gereken normal dağılım şartının sağlanıp sağlanmadığını sınavan Shapiro-Wilk testi yürütülmüştür. Buna göre deney gruplarına mensup faktör gruplarının skor dağılımında normallik şartı sağlanmaktadır (DG1 $p = .397$, KG1 $p = .391$). Bu bağlamda yürütülen Bağımsız Örneklem t-testine ilişkin sonuçlar Tablo 4-27’de verilmiştir.

Tablo 4.27 Kendi akıllı mobil cihazına sahip olmanın MBTGPD aracı ile GUI Programlama eğitiminde görev değeri algısına etkisi - t-testi sonuçları

Gruplar	n	$\bar{x}$	ss	t-testi		
				sd	t	p
Deney Grubu 1 (mobil cihaza sahip alt grup)	19	5,00	1,01	50	-1,562	,13
Deney Grubu 1 (mobil cihazı olmayan alt grup)	33	4,57	,89			

Buna göre, Motivasyon Ölçeği Görev Değeri alt boyutu puanları açısından kendi mobil cihazı olan ve olmayan öğrenciler arasında anlamlı bir fark yoktur $t(50) = -1,562$, $p > .05$

4.4.2 Mobil araçla öğrenim gören öğrencilerde cihaz aidiyetinin akademik başarıya etkisi

Hipotez 3.2'nin test edilebilmesi için birinci uygulama dönemi boyunca mobil cihazla öğrenim gören öğrencilerin Uygulama Sınavı 1 skorlarının mobil cihaz aidiyeti faktörüne göre karşılaştırılması öngörülmüştür. Bu maksatla kullanılmak istenen Bağımsız Örneklem t-testi için gereken normal dağılım şartının sağlanıp sağlanmadığını sınavın Shapiro-Wilk testi yürütülmüştür ve cihazı olmayan alt grubun skor dağılımında normallik şartının ihlal edildiği görülmüştür ($p = ,038$). Bu nedenle Bağımsız Örneklem t-testinin non-parametrik eşleniği olan Mann Whitney U testi yürütülmüş olup bu teste ilişkin sonuçlar Tablo 4-28'de verilmiştir.

Tablo 4.28 Kendi akıllı mobil cihazına sahip olmanın MBTGPD aracı ile GUI Programlama eğitiminde akademik başarıya etkisinin - t-testi sonuçları

Grup	n	Sıra Ortalaması	Sıra Toplamı	U	p
Deney Grubu 1 (mobil cihaza sahip alt grup)	19	25,37	482,00	292,00	.68
Deney Grubu 1 (mobil cihazı olmayan alt grup)	33	27,15	896,00		

Buna göre, Uygulama Sınavı 1 skoru açısından kendi mobil cihazı olan ve olmayan öğrenciler arasında farka rastlanmamıştır $U=292,00$ $p>.05$.

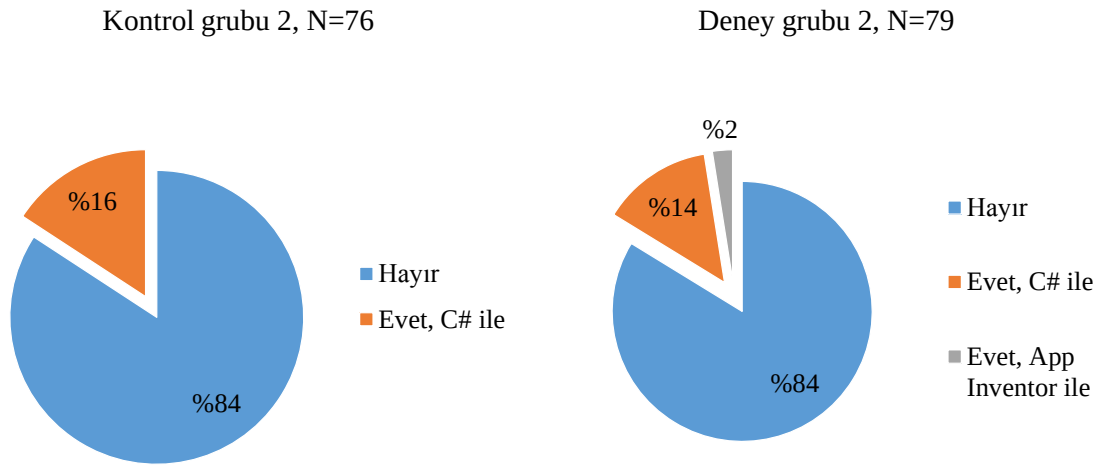
4.5 Araştırma Sorularına ait Bulgular

Deney Grubu 2 ve Kontrol Grubu 2 öğrencilerine, Ek 5'te paylaşılmış olan Uygulama Sonrası Anketi, Programlama Dilleri 2 dersini almış olmaya başladıkları tarihten 1 sene sonra uygulanmıştır. Bu ankette yer alan açık uçlu sorulara verilen cevaplar; araştırmacı ve dersten sorumlu öğretim elemanı tarafından Nvivo 10 nitel veri analiz programı

kullanılarak ayrı ayrı değerlendirilmiş; kategorilere ayrılmış ve ortaya çıkan temalar (varsa) incelenmiştir.

4.5.1 Kullanılan aracın, öğrencilerin kişisel amaçla geliştirmiş olduğu projelerin sayısına etkisi

GUI Programlama eğitiminde MBTGPD kullanmış deney grubu 2 öğrencileri ve aynı eğitimde MBTGPD kullanmamış olan kontrol grubu 2 öğrencilerine, programlama eğitiminin sona ermesinden itibaren geçen 1 yıllık süre içerisinde kişisel amaçlarla geliştirdikleri herhangi bir programlama projesi olup olmadığı sorusu yöneltilmiştir. Cevaplar üzerinde yürütülen analizin sonuçları Grafik 4-5'te verilmiştir.



Grafik 4-5 Kullanılan aracın, öğrencilerin kişisel amaçla geliştirmiş olduğu programların sayısına etkisi

Buna göre kontrol grubu 2 öğrencilerinin %84'ü, PD2 dersini almanın üzerinden geçen 1 yıllık süre içerisinde kişisel amaçlı herhangi bir yazılım geliştirmemişken, %16'sı MTPD (C#) kullanarak kişisel amaçlı yazılım geliştirmişlerdir. Deney grubu 2'de ise yine öğrencilerinin %84'ü 1 yıllık süre içerisinde kişisel amaçlı yazılım geliştirmemişken, %14'ü MTPD (C#) ve %2'si ise MBTGPD (App Inventor) ile kişisel amaçlı yazılımlar geliştirmişlerdir. Bu veriler üzerinde yürütülen Pearson Ki-Kare testi sonucunda MTPD yerine MBTGPD ile eğitim görmenin, öğrencilerin 1 yıl boyunca kişisel amaçla geliştirmiş oldukları yazılım uygulamalarının varlığı ve bu amaçla

kullanılan programlama dili üzerinde anlamlı bir etkisi bulunmadığı görülmüştür ($\chi^2(2) = 1.043, p = 0.59$).

4.5.2 Öğrencilerin yeni bir programlama dilini kolaylıkla öğrenebileceklerine dair inançları

GUI Programlama eğitiminde MBTGPD kullanmış deney grubu 2 öğrencileri ve aynı eğitimde MBTGPD kullanmamış olan kontrol grubu 2 öğrencilerine, yeni bir programlama dilini kolaylıkla öğrenip öğrenemeyecekleri sorusu yöneltilmiştir. Öğrencilerin soruya verdikleri cevaplarda en sık kullandıkları kelimeleri gösteren kelime bulutu şemaları Şekil 4-3'te verilmiştir.

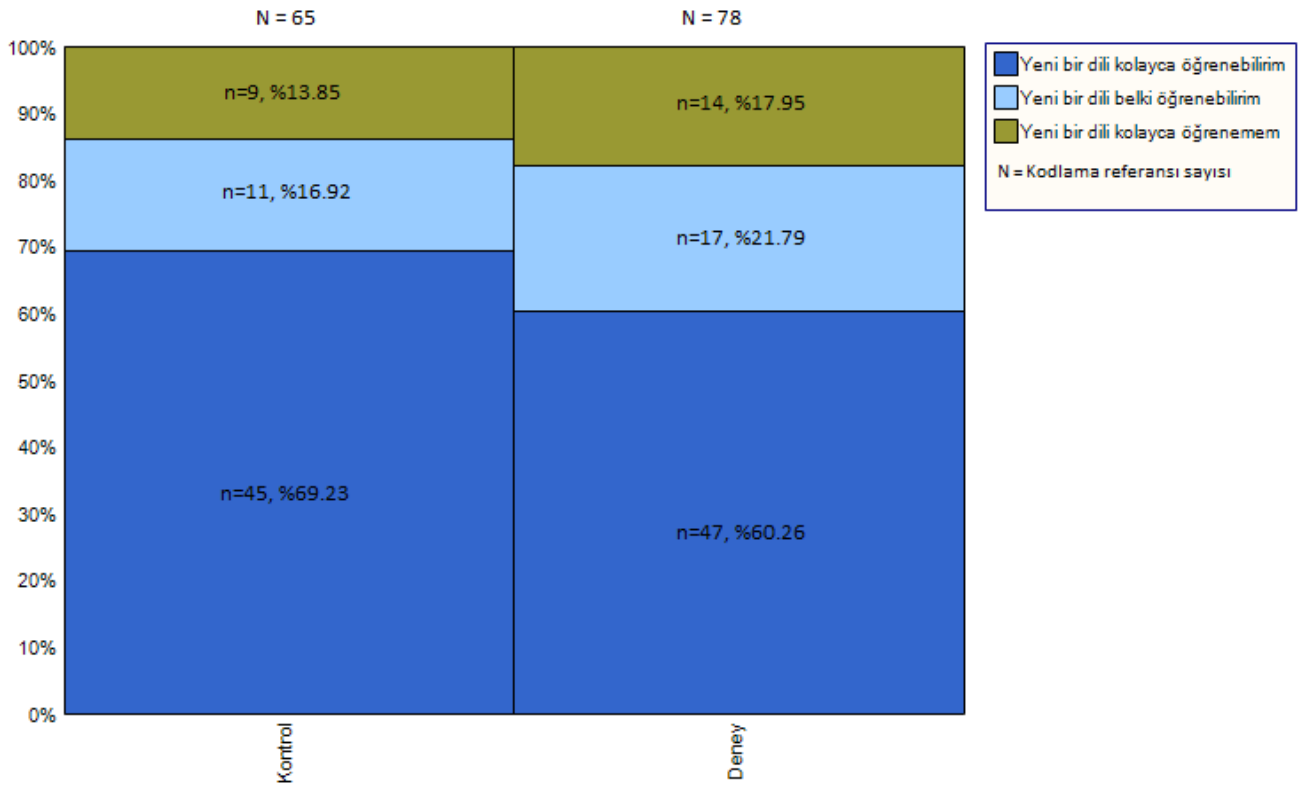


Şekil 4.3 Yeni bir görsel programlama dilini kolaylıkla öğrenebileceğinize inanıyor musunuz?" sorusu için kelime bulutları - (a) kontrol grubu 2 (b) deney grubu 2

Kelime bulutları incelendiğinde, her iki gruptaki öğrencilerin de “evet”, “inaniyorum”, “öğrenebilirim” şeklinde kelimeler kullanarak soruya olumlu yanıt verdiği, ve bu olumlu yanıtların “hayır” şeklinde verilen olumsuz cevaplara nazaran çoğunlukta olduğu görülmüştür. Yine her iki grupta da kimi öğrenciler; dile özgü “kod” yapısı (ikinci grupta “syntax” kelimesi de bu bağlamda kullanılmıştır) “fark”lılık gösterse de, aslında her dil için aynı olan programlama “mantığını” “kavramış” olduklarını belirtmişlerdir. Bilhassa Kontrol Grubu 2’de, kimi öğrenciler diller arasında “küçük” farklar olduğunu, bunun haricinde “hepsi”nin aynı olduğunu ve dolayısıyla yeni bir dili “rahatlıkla” öğrenebileceklerini vurgulamışlardır. Öte yandan; kontrol grubunda bahsi geçmemiş olmasına rağmen deney grubu’nda kullanıldığı görülen “çaba” kelimesi ile öğrenciler;

yeni bir dili öğrenebileceklerini ancak bunun için çaba sarf etmeleri gerektiğini ifade etmişlerdir.

Soruya veren cevaplar üzerinde yapılan kodlamalara göre, soruyu cevaplayan kontrol grubu 2 öğrencilerinin %69.23'ü yeni bir dili kolayca öğrenebileceğini düşünürken (n=45), %16.92'si emin olamadıklarını (n=11) ve %13.85'i ise yeni bir dili kolayca öğrenemeyeceğini (n=9) belirtmiştir. Deney grubu 2 öğrencilerinde ise %60.26'lık kesim yeni bir dili kolayca öğrenebileceğini ifade ederken (n=47), öğrencilerin %21.79'u (n=17) emin olamadıklarını, %17.95'i ise (n=14) yeni bir dili kolayca öğrenemeyeceklerini belirtmişlerdir. Bu verilerin karşılaştırmalı gösterimi Grafik 4-6'da verilmiştir.



Grafik 4-6 Kullanılan aracın, öğrencilerin yeni bir programlama dilini kolaylıkla öğrenebileceklerine dair inançlarına etkisi, karşılaştırmalı grafik

bir dili öğrenebileceklerine dair inancın, her kod kategorisi için kontrol grubuna kıyasla nispeten azalmış olduğu görülmektedir. Bu grubun kelime bulutu analizinde beliren “çaba” kelimesi ile öğrencilerin yeni bir dili ancak çaba göstererek öğrenebilecekleri iması da bu durumu destekler niteliktedir. Neticede, MTPD’nin yanı sıra MBTGPD ile çalışmış olmak öğrencilerin yeni bir dili rahatlıkla öğrenebilecekleri yönündeki inançlarını göreceli olarak azaltmıştır.

4.5.3 Öğrencilerin acemilere yönelik programlama eğitimini başarıyla verip veremeyeceklerine dair görüşleri

GUI Programlama eğitiminde MBTGPD kullanmış deney grubu 2 öğrencileri ve aynı eğitimde MBTGPD kullanmamış olan kontrol grubu 2 öğrencilerine, acemilerden oluşan bir gruba programlama dersi vermeleri istendiğinde bu görevi başarıyla yerine getirip getiremeyecekleri sorusu yöneltilmiştir. Öğrencilerin bu soruya verdikleri cevaplarda en sık kullandıkları kelimeleri gösteren kelime bulutu şemaları Şekil 4-4'te verilmiştir.



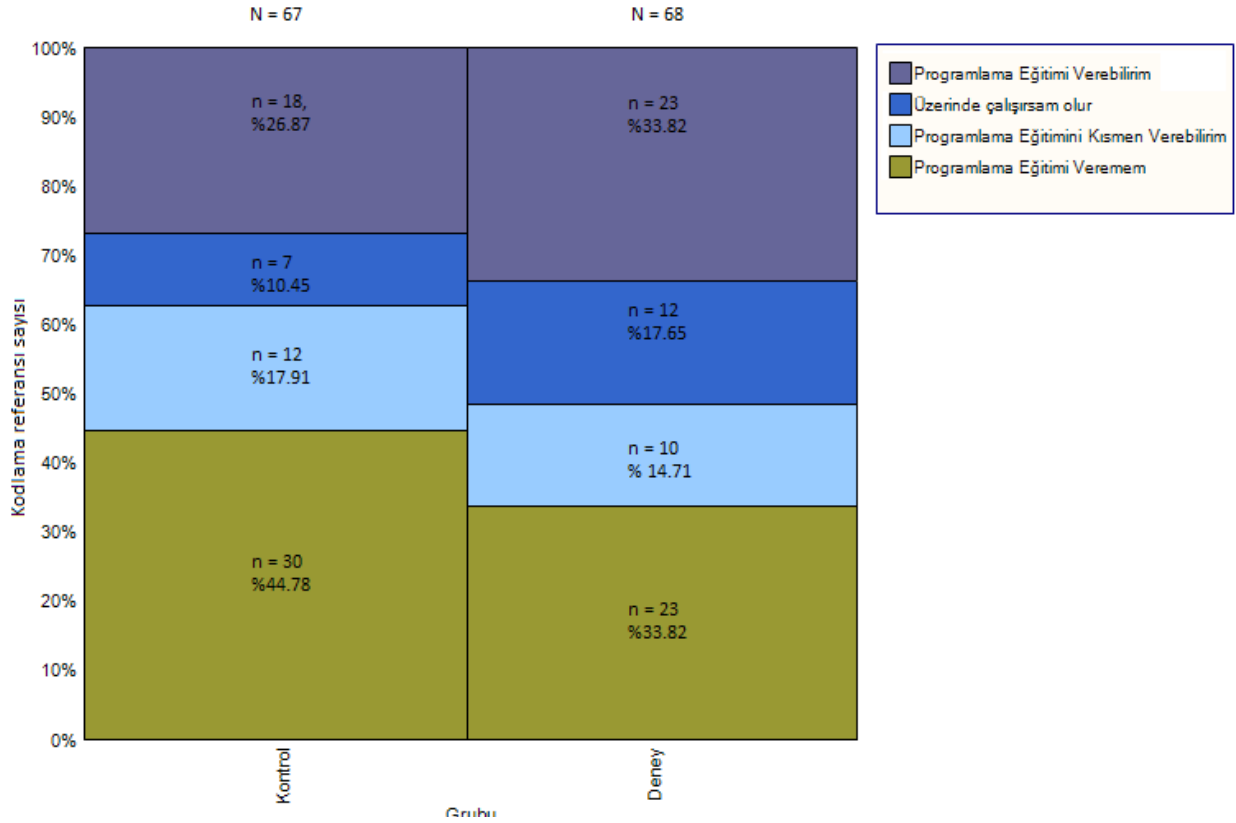
Şekil 4.4 Acemilere programlama dersi verebilir misiniz? sorusu için kelime bulutları (a) kontrol grubu 2 (b) deney grubu 2

Deney ve kontrol grupları için kelime bulutları incelendiğinde; her iki grupta da soruya verilmiş “evet”, “öğretebilirim” şeklinde kelimeler kullanarak olumlu yanıtlara rastlandığı; ancak olumlu yanıtların deney grubunda daha yaygın olduğu ve kontrol grubunda en çok öne çıkan kelimenin “hayır” olduğu görülmektedir. Diğer yandan, olumlu yanıt veren öğrencilerden bir kısmı, bilhassa deney grubunda, “seviye” kelimesini kullanarak sıklıkla, eğitimi ancak “temel” düzeyde verebileceklerini beyan etmişlerdir. “Hayır”, “öğretemem” gibi kelimelerle olumsuz yanıtlar ortaya koyan öğrencilerin de yine “seviye” ve benzeri kelimeler kullanarak, “kapsamlı” bir eğitim veremeyeceklerini beyan ettiği gözlemlenmiştir. Diğer yandan, gruplarda kimi öğrencilerin “henüz” kelimesini sıklıkla kullanarak, kendilerini programlama eğitimi vermeye henüz hazır hissetmediklerini vurguladıkları göze çarpmıştır.

Kontrol grubunda kimi öğrenciler “işin ‘mantığını’ kavradıklarını” dolayısıyla eğitim verebileceklerini, yahut öğrencilere programlamanın temel “mantığını” öğretebileceklerini beyan ederek programlama mantığına gönderme yapmıştır.

Deney grubunda, öğrenciler “görsel” kelimesi ile bilhassa GUI programlama eğitimine atıfta bulunmuştur. MBTGPD gibi ikinci bir dille GUI programlama Zaman zaman ise “App Inventor” ve “ortam” kelimelerini birlikte kullanarak direkt olarak MBTGPD’ne atıfta bulunmuş ve programlama eğitimini bu dili kullanarak verebileceklerinden de bahsetmişlerdir. Ancak rastlanan “kaynak” ve “yardım” gibi kelimeler bağlamında detaylı bir analiz yapıldığında öğrencilerin eğitim vermek için öncelikle bir kaynaktan yardım almaları gerektiğini vurguladıkları sonucuna ulaşılmıştır. Kontrol grubunda gözlemlenmeyen bu durum, öğrencilerin bilhassa MBTGPD açısından eğitsel kaynak eksikliği çekmiş olduğu ipucunu veriyor olabilir. Yine bu grupta, kontrol grubunun aksine, “algoritma” kelimesi kullanıldığı görülmüştür. Bu durum, bu öğrencilerin programlama kavramına daha geniş bir bakış açısıyla yaklaşmaya başlıyor olması şeklinde yorumlanabilir.

Soruya veren cevaplar üzerinde yapılan kodlamalara göre, soruyu cevaplayan kontrol grubu 2 öğrencilerinin %26.87’si kendilerini acemilere yönelik programlama eğitimi verecek yetkinlikte hissederken (n=18), %17.91’i acemilere yönelik eğitimi verip veremeyeceklerinden emin olmadıkları (n=12); %10.45’i bu eğitimi ancak öncelikle hazırlık çalışması yaptıkları takdirde verebileceklerini (n=7) ve %44.78’i ise acemilere yönelik programlama eğitimi veremeyeceklerini (n=30) belirtmiştir. Deney grubu 2 öğrencilerinin ise %33.82’lik kesimi acemilere yönelik programlama eğitimi verebileceklerini ifade ederken (n=23), öğrencilerin %14.71’i programlama eğitimini kısmen verebileceklerini (n=10); öğrenebileceklerini ve %17.65’i eğitimi verebileceklerini ancak öncelikle hazırlık çalışması yapmaları gerektiğini (n=12) belirtmiş; öğrencilerin %33.82’si ise acemilere yönelik programlama eğitimi veremeyeceklerini (n=23) söylemiştir. Bu verilerin karşılaştırmalı gösterimi Grafik 4-7’de verilmiştir.



Grafik 4-7 öğrencilerin acemilere yönelik programlama eğitimi verebileceklerine dair inançlarına etkisi, karşılaştırmalı grafik

Grafik incelendiğinde, MBTGPD ile eğitim görmüş deney grubu öğrencilerinin, programlama eğitimi verme konusundaki inançlarının, yalnızca MTPD ile eğitim görmüş kontrol grubu öğrencilerine kıyasla göreceli olarak yükselmiş olduğu yorumunda bulunulabilir.

4.5.4 Öğrencilerin ortaokul öğrencilerine GUI-programlama dersi verilmesi hususunda görüşleri

GUI Programlama eğitiminde MBTGPD kullanmış deney grubu 2 öğrencileri ve aynı eğitimde MBTGPD kullanmamış olan kontrol grubu 2 öğrencilerine, ortaokul öğrencilerine GUI programlama eğitimi verilmesi fikri konusundaki düşünceleri sorulmuştur. Öğrencilerin bu soruya verdikleri cevaplarda en sık kullandıkları kelimeleri gösteren kelime bulutu şemaları Şekil 4-5'te verilmiştir.



(a)



(b)

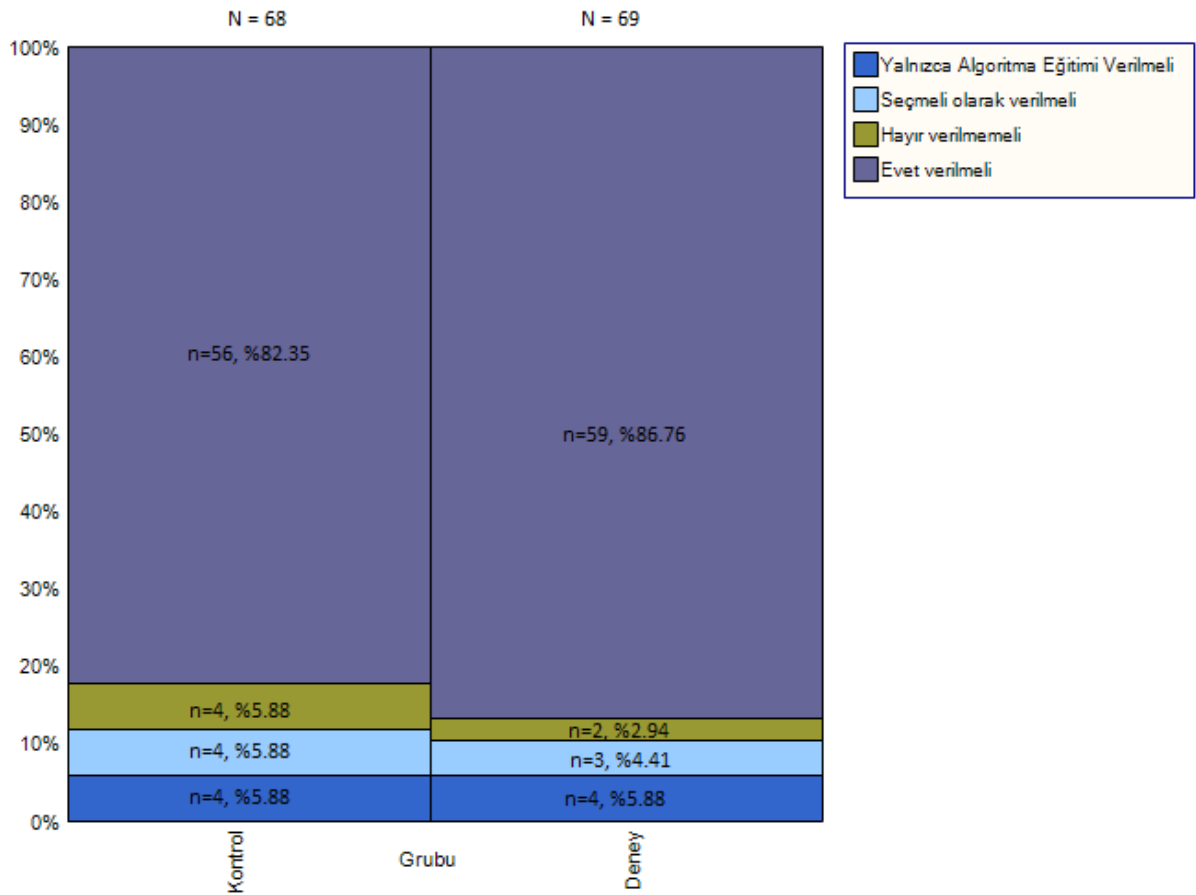
Şekil 4.5 Ortaokul öğrencilerine programlama dersi verilmeli mi? sorusu için kelime bulutları (a) kontrol grubu 2 (b) deney grubu 2

Gerek kelime bulutlarının analizi, gerekse öğrencilerin cevaplarının kodlanması yoluyla; her iki grupta da ortak olarak tespit edilen temalar aşağıda verilmiştir:

1. Programlama ile ortaokul gibi erken bir yaşta tanışmanın öğrenciyi “yönlendirmesi”; ileriki hayatındaki meslek seçiminde kolaylık sağlaması açısından “faydalı/yararlı” olacağı (Deney Grubu 2: n=9, %34.62, Kontrol Grubu 2: n=8, %38.10) :
2. Programlama eğitiminin GUI uygulamalarla verilmesinin öğrenci için “enteresan” ve “eğlenceli” olabileceği (Deney Grubu 2: n=4, %19.23, Kontrol Grubu 2: n=2, %9.52):
3. Programlama eğitiminin öğrenciye “zeka” gelişimi, gibi farklı açılardan da faydalı olabileceği (Deney Grubu 2: n=8, %30.77, Kontrol Grubu 2: n=9, %42.86):
4. Özellikle günümüz koşulları ele alındığında programlama eğitiminin ortaokul yaşında verilmeye başlanmasının uygun olduğu (Deney Grubu 2: n=4, %15.38, Kontrol Grubu 2: n=2, %9.52).

Cevaplar üzerinde yapılan kodlamalara göre; kontrol grubu 2 öğrencilerinden soruyu cevaplayanların %82.35’i Ortaokul öğrencilerine GUI programlama eğitimi verilmesi fikri konusunda olumlu görüşe sahipken (n=56), öğrencilerin %5.88’i dersin seçmeli

olarak verilmesi gerektiğini (n=4), %5.88'i yalnızca algoritma eğitimi verilmesinin daha doğru olacağını (n=4) ve %5.88'i ise bu seviyedeki öğrencilere programlama eğitimi verilmesine olumlu bakmadıklarını (n=4) belirtmişlerdir. Deney grubu 2'de ise öğrencilerin %86.76'sı soruya olumlu yanıtlar verirken (n=59); %2.94'lük bir kesim bu yaşta programlama eğitimi verilmemesi gerektiğini savunmuş (n=2); %4.41'lik bir kesim dersin seçmeli olması gerektiğini belirtmiş (n=3) ve %5.88'lik bir kesim ise bu yaşta yalnızca algoritma eğitimi verilmesinin daha iyi olacağını (n=4) savunmuştur. Bu verilerin karşılaştırmalı gösterimi Grafik 4-8'de verilmiştir.

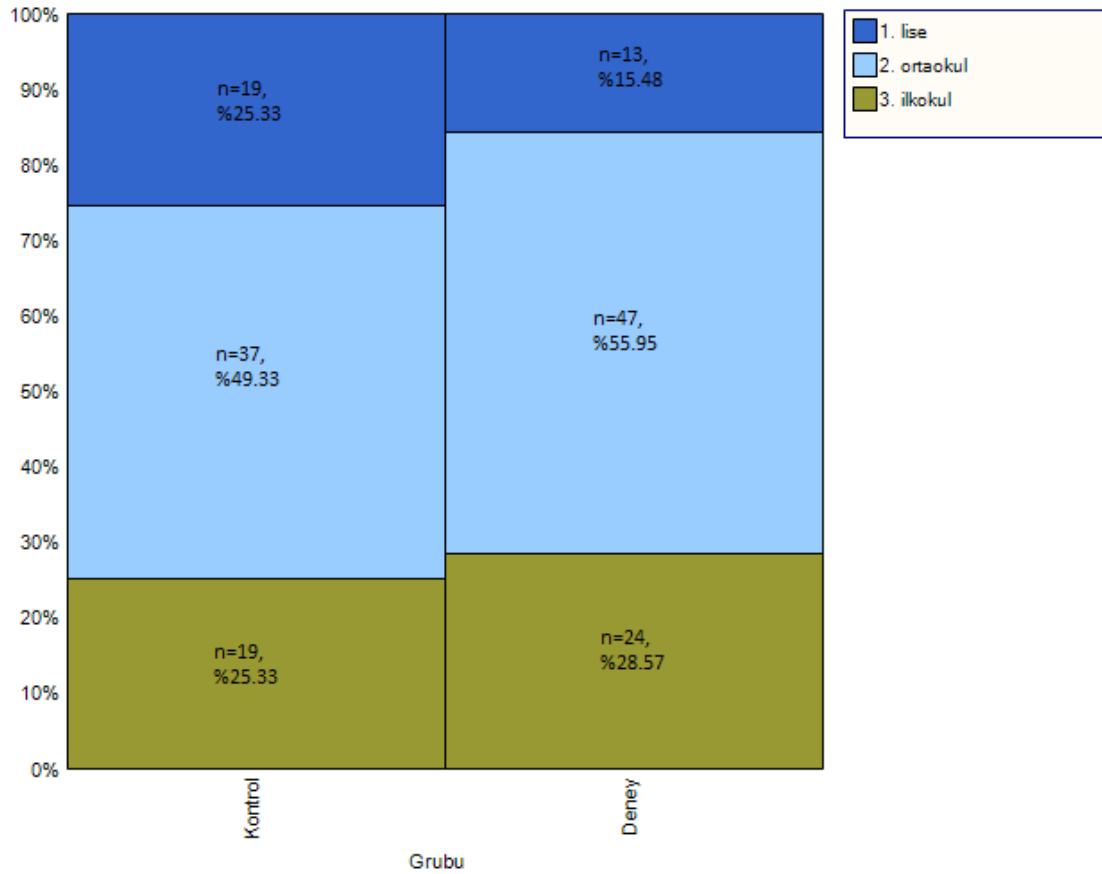


Grafik 4-8 Kullanılan aracın, öğrencilerin ortaokulda programlama eğitim verilmesi fikrine dair görüşlerine etkisi, karşılaştırmalı grafik

Grafik incelendiğinde görülmektedir ki; her iki grupta da öğrenciler benzer oranlarda ve büyük çoğunlukla ortaokul seviyesinde programlama eğitimi verilmesi hususuna olumlu yaklaşmaktadırlar.

4.5.5 Öğrencilerin programlama eğitiminin verilmeye başlanması gerektiği yaşa ilişkin görüşleri

GUI Programlama eğitiminde MBTGPD kullanmış deney grubu 2 öğrencileri ve aynı eğitimde MBTGPD kullanmamış olan kontrol grubu 2 öğrencilerine, programlama eğitiminin hangi yaşta verilmeye başlanması gerektiğini düşündükleri sorulmuştur. Soruya veren cevaplar üzerinde yapılan kodlamalar “Liseden itibaren” (Deney Grubu 2: n=13, %15.48, Kontrol Grubu 2: n=19, %25.33), “Ortaokuldan itibaren” (Deney Grubu 2: %55.95, Kontrol Grubu 2: n=37, %49.33) ve “İlkokuldan itibaren” (Deney Grubu 2: n=24, %28.57, Kontrol Grubu 2: n=19, %25.33) şeklinde gruplandırılmıştır. Cevaplarda kodlamalara ilişkin karşılaştırılmalı grafik, Grafik 4-9’da verilmiştir.

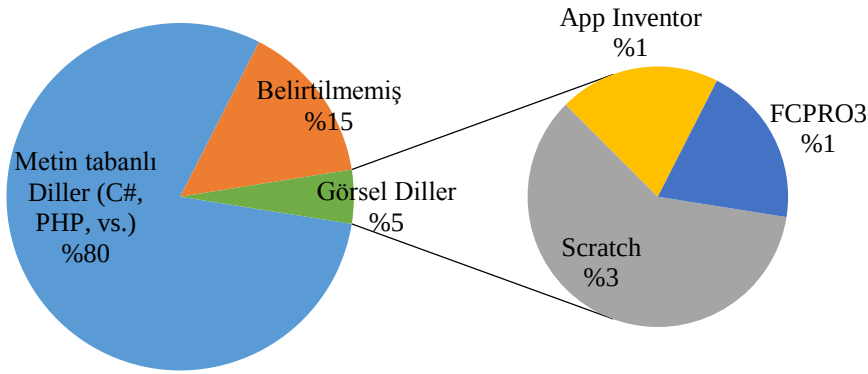


Grafik 4-9 Öğrencilerin programlama eğitiminin verilmeye başlanması gerektiğini düşündükleri seviyelerin gruplara göre dağılımları

Grafik incelendiğinde; deney grubu öğrencilerinin, programlama eğitiminin verilmeye başlanabileceği yaşla ilgili görüşlerinin kontrol grubuna kıyasla az da olsa düşme eğiliminde olduğu görülmüştür. Yani MBTGPD ile eğitim almış olmak, programlama eğitiminin nispeten daha erken yaşta alınmaya başlanabileceğini düşünmeye itmiştir.

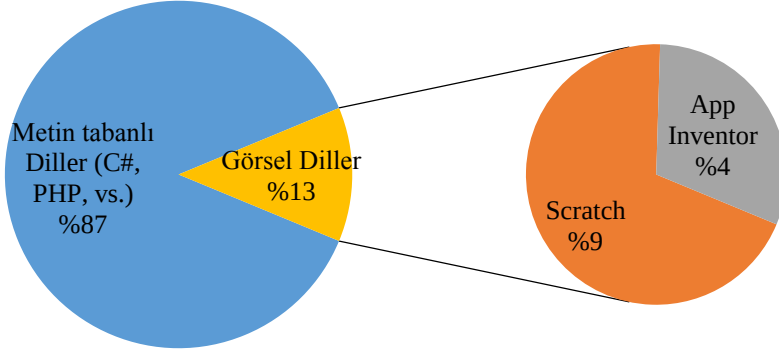
4.5.6 Mikro-öğretim uygulamasının analizi (Araştırma Sorusu 7)

Öğrencilerden “Programlamada FOR döngüsü” isimli konunun nasıl öğretilbileceği konusunda 20-dakika uzunluğunda bir mikro-öğretim videosu çekmeleri istenmiştir. Söz konusu videolar incelenerek, öğrencilerin dersi hangi aracı kullanarak anlatmayı tercih ettiği belirlenmiştir. Buna göre, hem MBTGPD hem de MTPD ile eğitim almış Deney Grubu 2 öğrencilerinin tercihleri Grafik 4-10’da gösterilmiştir.



Grafik 4-10 Deney Grubu 2 öğrencilerinin FOR Döngüsü konulu Mikroöğretim Videosunun hazırlanmasında tercih etmiş olduğu eğitsel araçlar (n= 101)

Buna göre Deney Grubu 2’deki öğrenciler toplamının (n = 101) %80’i dersi metin tabanlı diller ile anlatmayı seçerken; %5’i görsel dilleri tercih etmiştir. Bu %5’lik kısımda ise %1 oranla akış şeması programı FCPRO3, %3 oranla MIT Scratch ve %1 oranla MIT App Inventor bulunmaktadır. Öğrencilerin %15’i ise videolarında konuyu anlatırken spesifik bir dil veya IDE kullanmamıştır. Yalnızca MTPD ile eğitim almış Kontrol Grubu 2 öğrencilerinin tercihleri ise Grafik 4-11’de gösterilmiştir.



Grafik 4-11 Kontrol Grubu 2 öğrencilerinin FOR Döngüsü konulu Mikroöğretim Videosunun hazırlanmasında tercih etmiş olduğu eğitsel araçlar (n= 76)

Buna göre Kontrol Grubu 2'deki öğrenciler toplamının (n = 76) %87'si dersi metin tabanlı diller ile anlatmayı seçerken; 13'ü görsel dilleri tercih etmiştir. %13'lük oranın %9'u MIT Scratch'ten, %4'ü ise MIT App Inventor'dan oluşmaktadır.

Grafik verileri karşılaştırıldığında; daha önceki ankette eğitimde App Inventor'ın kullanılabileceğine yönelik kimi görüşler beyan etmiş olan deney grubu öğrencilerinin; uygulamaya gelindiğinde kontrol grubu öğrencilerine kıyasla mikroöğretim etkinliği için App Inventor'a daha az rağbet etmiş olması durumu çarpıcı bir bulgudur.

BÖLÜM V: SONUÇ

5.1 Sonuç ve Tartışma

Bu çalışma üniversite seviyesinde GUI-programlama eğitiminde eğitsel araç olarak mobil uygulama geliştirmeye yönelik blok-temelli bir görsel programlama dili kullanmanın; metin tabanlı bir programlama dili kullanmaya kıyasla öğrenci akademik başarısı üzerindeki etkisini incelemeyi amaçlayan yarı-deneysel bir uygulamanın sonuçlarını sunmaktadır.

Araştırma bulgularına göre; farklı araçlarla öğrenim gören deney gruplarının akademik başarıları karşılaştırıldığında üniversite seviyesindeki GUI programlama eğitiminde MTPD yerine MBTGPD kullanmanın; genel anlamda akademik başarı üzerinde farklılık oluşturmadığı görülmüştür.

Ancak, mevcut senaryoda programlama açısından acemi ve deneyimli öğrencilerin birlikte eğitim gördüğü gerçeği de hesaba katıldığında; kullanılan aracın başarıya ve motivasyona etkisi üzerinde bu faktörlerin de rol oynayabileceği de düşünülmüş ve sonuçlar bu perspektiften değerlendirilmiştir.

Bu bağlamda yürütülen analizler göstermiştir ki GUI-programlama eğitiminde MTPD yerine MBTGPD kullanmanın akademik başarıya etkisi üzerinde üniversite öncesi GUI programlama deneyimi faktörü rol oynamaktadır. Ayrıca, üniversitede daha önce alınmış olan Programlama Dilleri 1 dersi başarısı da başarıyı etkileyen başka bir faktördür.

Analize ait profil grafikleri incelendiğinde görülmüştür ki; PD1 dersinde MTPD ile eğitim görmüş ve nispeten başarısız bulunmuş öğrencilerin GUI programlama eğitimini MBTGPD ile alanlar, aynı dersi MTPD ile alan akranlarından daha yüksek not ortalamasına sahiptir. Benzer şekilde; lisede MTPD ile GUI programlama eğitimi almamış öğrencilerden üniversitedeki GUI programlama eğitimini MBTGPD ile alanlar; MTPD ile alan akranlarına göre daha yüksek ortalamaya sahiptir.

Bunun tam tersi de geçerli olup; PD1’de başarılı veya lisede MTPD ile GUI programlama eğitimi almış öğrenciler, üniversitedeki GUI programlama eğitimine MTPD ile devam etme durumunda, aynı eğitimi MBTGPD ile alan akranlarına göre nispeten daha yüksek ortalamaya sahip olduğu görülmüştür.

Durumu daha detaylı inceleyebilmek için; gruplar GUI-programlama başarısına etkisi olacağı düşünülen faktörlerin kombinasyonlarına göre metin tabanlı dillerin kullanımında “Deneyimli” ve “Acemi” alt gruplarına ayrılarak; her alt grupta derste ilk kullanılan aracın akademik başarı etkisi ayrı ayrı incelenmiştir.

Bu bağlamda yürütülen analizler sonucu; acemi öğrenci alt grupları arasında akademik başarı açısından kullanılan araca bağlı bir fark tespit edilmiştir. Bu öğrencilerden MBTGPD kullananlar, MTPD kullanan eşleniklerine göre anlamlı ölçüde daha başarılıdır.

Bu durum programlamayla ilk kez üniversite seviyesinde ve metin tabanlı bir programlama dili aracılığıyla tanışmış olan öğrencilerin; söz konusu dile sözdiziminin karmaşıklığı yüzünden öğrenme güçlüğü yaşıyor ve bu nedenle başarısızlığa uğruyor olabileceği ihtimalini işaret etmektedir. Bu noktada MBTGPD aracı öğrencilerin programlama dersinde başarılarını arttırmalarını sağlamak ve muhtemelen bir kurtarıcı rolü üstlenmektedir.

Diğer yandan, deneyimli öğrenci alt gruplarında kullanılan aracın akademik başarı üzerinde etkisi incelendiğinde yine anlamlı bir istatistiksel farka rastlanmıştır. Bu alt gruplarda ise tam tersine, MBTGPD kullananlar, MTPD kullanan akranlarına göre daha başarısız olmuştur.

Literatürde mobil cihazlarda uygulama geliştirmeye yönelik blok-temelli bir görsel programlama dili olan App Inventor’ın programlama eğitimine başlangıçta kullanılmasının, özellikle motivasyonda yarattığı artış sayesinde fayda getirebileceğinden bahsedilmiştir (Ahmad & Gestwicki, 2013). Bilhassa Karakuş ve arkadaşlarının (2012) çalışmasında, App Inventor’ın üniversite seviyesindeki öğrencilerle kullanımı hususunda iddialı önerilerde bulunulmuş ve bu programlama dilinin bir eğitsel araç olarak yalnızca giriş niteliğindeki değil, ileri düzeydeki programlama derslerinde de kullanılabileceğinden bahsedilmiştir. Zira bu araştırmacılara göre App Inventor mevcut öğrencilerin programlamaya olan ilgisini canlı tutmasını sağlayabilme potansiyeline sahiptir. Ancak App Inventor’ın, tamamı daha önce metin tabanlı dillerle programlama eğitimi almış öğrencilerle kullanıldığı bu araştırmaya ait bulgular, Karakuş ve arkadaşlarının deneysel olarak ispatlanmamış bu iddialarıyla kısmen çelişmektedir. Öğrencilerin programlamaya metin-temelli bir dille

tanışmaları durumunda ileri düzey programlama derslerinde App Inventor'ın kullanımını inceleyen bu araştırmanın bulguları göstermektedir ki; söz konusu blok programlama aracının bu noktada fayda sağlayabildiği öğrenci grubu, yalnızca metin-temelli programlama dilinin öğretildiği daha önceki programlama derslerinde başarısız bulunmuş öğrencilerle sınırlıdır. Zira, metin tabanlı bir dil kullanarak hem GUI-programlama kavramıyla üniversite öncesinde tanışmış, hem de üniversite seviyesinde programlama dersinde başarılı olmuş öğrencilerin, MBTGPD ile çalışırken en az metin tabanlı dille olduğu kadar başarı göstermesi beklenmiş olsa da; çalışmada elde edilen sonuçlar bu öğrencilerin bilgilerini blok programlamanın kullanıldığı ortama aktarırken zorluk çektiğini göstermektedir. Bu iddiayı destekler nitelikte bir diğer bulgu da; motivasyon açısından incelendiklerinde bu alt gruplarda yine içsel hedef düzenleme ve görev değeri algıları arasında anlamlı bir fark bulunamamış olsa da; özyeterlilik algısı açısından anlamlı bir fark keşfedilmiş olmasıdır. Beklendiği üzere, deneyimli öğrenci alt gruplarından MTPD ile eğitim alanların özyeterlilik algısı, MBTGPD ile eğitim alan akranlarına kıyasla daha yüksektir. Dolayısıyla bu öğrencilerin MBTGPD kullanırken kendilerini yeni ve anlaşılması güç bir ortama giriş yapmış gibi hissediyor olabilecekleri sonucuna varılabilir. Bir diğer ihtimal ise, aynı öğrenciler kullanımına aşina oldukları MTPD ile çalışırken kendilerini daha rahat hissetmiş olabileceğidir. Blok temelli dillerin, metin tabanlı dillere göre daha kolay anlaşılabilir olmasına rağmen böyle bir durumun gözlemlenmesi yine; öğrencilerin metin tabanlı bir dile ait sözdizimini öğrenmeye çabalamak yüzünden programlamaya yönelik genellenebilir bir bakış açısı geliştirememiş olabilecekleri ihtimalini kuvvetlendirmektedir.

Bu noktada araştırmanın ikinci amacı olan motivasyon boyutuna yönelik incelemelerden bahsetmek faydalı olacaktır. Eğitsel aracın akademik başarı açısından yaratabileceği etkiye yönelik sebeplerin anlaşılabilmesi amacıyla; aynı uygulama kapsamında öğrencilerde motivasyona ait alt boyutlar olan içsel hedef düzenleme, görev değeri ve özyeterlilik algıları incelenmiştir. Bu bağlamda; farklı araçlarla öğrenim gören deney gruplarında motivasyon testlerinden alınan puanlar karşılaştırılmış ve yürütülen analize göre üniversite seviyesindeki GUI programlama eğitiminde MTPD yerine MBTGPD kullanmanın; genel anlamda öğrencilerin içsel hedef düzenleme, görev değeri ve özyeterlilik algıları üzerinde anlamlı bir etki yaratmadığı görülmüştür.

Bunun yanı sıra, akademik başarı üzerindeki etkililiği anlamlı bulunmuş PD1 Başarısı ve üniversite öncesi deneyim faktörlerinin; kullanılan aracın içsel hedef düzenleme, görev değeri ve özyeterlilik algılarına etkileri üzerinde açıklayıcı roller oynamadığı anlaşılmıştır. Yani derste MTPD yerine MBTGPD kullanmanın öğrenci motivasyonu üzerinde genel bir etkisi olmadığı gibi; öğrencilerin üniversitede veya öncesinde metin tabanlı programlama diliyle eğitim almış olması durumları da MBTGPD ile çalışma açısından motivasyon dağılımları üzerinde anlamlı bir desen meydana getirmemiştir. Bu aynı zamanda; acemi/deneyimli grupların başarısındaki olumlu/olumsuz etkiler ile motivasyon arasında genel bir ilişki kurulamadığı anlamına gelmektedir.

Bu durum, App Inventor’ı acemilere yönelik programlama eğitiminde kullanan ve motivasyonu yine MSLQ (Büyüköztürk ve diğ., 2004; Garcia & Pintrich, 1996) ölçeği ile inceleyen ve App Inventor’ın öğrenci motivasyonuna olumlu katkılarda bulunduğunu betimleyen benzer çalışmanın (Ahmad & Gestwicki, 2013) sonuçlarıyla çelişmektedir. Ancak Ahmad ve Gestwicki’nin söz konusu araştırmasında, eldeki çalışmadan farklı olarak; App Inventor’ın “Stüdyo Temelli Öğrenme” yöntemiyle öğrencilere sunulmuş olduğu ve bahsi geçen çalışmada öğrencilerin programlamaya sıfırdan başlamış olduğu gibi etkenlerin varlığı da not düşülmelidir.

Bir diğer araştırma bulgusuna göre; MBTGPD ve MTPD araçlarının GUI programlama eğitiminde kullanım sırası ne deney gruplarındaki öğrencilerin tamamı açısından, ne de üniversite öncesi GUI-programlama deneyimi bulunmayan alt gruplar açısından GUI-programlamada nihai akademik başarı üzerinde anlamlı bir etkiye sahip değildir. Ancak bu sonucu yorumlarken; nihai akademik başarıyı ölçen ve soruların ister MTPD ister MBTGPD kullanılarak cevaplanabileceği uygulamalı sınavda öğrencilerin neredeyse tamamının soruları MTPD kullanarak cevaplamayı tercih etmiş olduğu not düşülmelidir. Dolayısıyla, öğrencilerin bu sınavdaki başarısı neredeyse tamamen MTPD kullanımı üzerinden ölçülmüş olup, öğrencilerin MBTGPD ile nasıl bir akademik performans sergileyecekleri belirlenememiştir.

Derste kullanılan aracın dönem ortasında değiştirilmesinin, deneyimli ve deneyimsiz alt gruplar üzerinde motivasyon açısından etki yaratıp yaratmadığı da incelemeye konu edilmiştir. Yürütülen analize göre, derste kullanılan aracın değiştirilmesi, deneyimsiz öğrencilerin derse yönelik içsel hedef düzenleme algısındaki değişimler üzerinde

anlamalı bir etkiye sahiptir. Bu öğrencilerde içsel hedef düzenleme algısı, MTPD'dan MBTGPD'ya geçen grupta azalırken; MBTGPD'dan MTPD'ya geçen grupta artmıştır. İçsel hedef düzenleme algısının, öğrencinin derste öğrenilen konulara yönelik kendisiyle rekabet algısı olarak da yorumlanabileceği düşünüldüğünde; bu durum bu öğrencilerin MTPD'ni MBTGPD'ne kıyasla daha zorlayıcı olarak algıladığı sonucuna götürebilir ki bu da beklenen bir durumdur. Fakat; kullanılan aracın değiştirilmesi durumu aynı öğrencilerin Görev değeri ve Özyeterlilik algılarında değişim üzerinde anlamalı bir etki yaratmamıştır.

Deneyimli öğrenciler özelinde ise kullanılan aracın değiştirilmesi içsel hedef düzenleme, görev değeri ve özyeterlilik algılarında değişimler açısından herhangi bir fark yaratmamıştır. Dolayısıyla; metin tabanlı programlama dilini öğrenmede nispeten daha başarılı öğrencilerin de motivasyona yönelik olumlu etkileri sonucu “derse yönelik ilgisini” arttıracak iddia edilen (Ahmad & Gestwicki, 2013; Karakus ve diğ., 2012) App Inventor'ın; öğrencilerde motivasyon açısından gözle görülür bir artışa yol açmadığı gibi; bilakis, akademik başarının düşmesine sebep olduğu görülmektedir. App Inventor'ın programlamada deneyimli öğrencilerle kullanımı üzerinde yaptığı çalışmada Soares'ın da ipuçlarını verdiği üzere (Soares, 2013), metin tabanlı programlama ile çalışmış öğrenciler App Inventor'da rastlanan blok yapılarıyla çalışmayı hantal buluyor olabilir. Diğer araştırmacılar da görsel programlama araçları ile yürüttükleri araştırmalarda benzer sonuçlar elde etmiş, blok-programlamanın deneyimli öğrenciler tarafından kullanışlı olarak algılanmadığını (Tanrikulu & Schaefer, 2011), yahut bir “oyuncak” gibi görüldüğünü (Kasurinen ve diğ., 2008) iddia etmişlerdir. Bhagi ise yaptığı çalışmada App Inventor'ın kendisine özgü birtakım eksiklikleri ve kullanılabilirlik sorunlarına değinmiştir (Bhagi, 2012). Yani, gerek öğrencilerin görsel blok programlamaya yönelik genel yaklaşımları, gerekse App Inventor'ın spesifik olarak sahip olduğu özellikler; programlama eğitimini bu güne dek ticari değere sahip IDE'ler üzerinde almış öğrencilerin App Inventor ile çalışırken sorunlar yaşamasına neden olmuş olabilir. Zira, App Inventor IDE'sinin serbest yazılım lisanslı, kullanımı ücretsiz ve en önemlisi, amatör uygulamalar geliştirmekte kullanılan bir ortam olduğu, üstüne üstlük beta aşamasında bulunduğu unutulmamalıdır. Bütün bu özellikleri, bir web tarayıcı içerisinde çalışan App Inventor'ı, bilgisayara yerel olarak kurulan profesyonel bir IDE'ye kıyasla hantal görünmesine sebep olmaktadır. Nitekim

uygulamaya katılan öğrencilerin; eğitimde kullanılan her iki aracı da kullanarak cevaplayabilecekleri bir sınavda, %99 oranında App Inventor yerine metin tabanlı dili (Visual C#) kullanmayı tercih etmemesi bu şekilde açıklanabilir. Programlama eğitiminde kullanılan IDE'nin de eğitimde başarıya etkisinin olduğu gösterilmiştir (Vogts, Calitz, & Greyling, 2008) ve dolayısıyla öğrenciler daha hızlı çalışabilecekler veya daha büyük başarı elde edebileceklerini düşündükleri IDE'ye yönelmiş olabilirler.

Akıllı mobil cihazlarla çalışmanın motivasyonu arttırması bekleniyor olmasına (Ahmad & Gestwicki, 2013) rağmen, öğrencilerin App Inventor'a olumlu olarak yaklaşmıyor olması durumunu açıklayabileceği düşünülen bir diğer husus ise, bu uygulamada öğrencilere alternatif olarak sunulan metin-temelli programlama dili Visual C#'ın çok çeşitli endüstrilerde geniş kullanım alanına sahip, profesyonel bir dil olmasıdır. Öğrenciler sağlayabileceği iş imkanları dolayısıyla Visual C#'ı iyiden iyiye öğrenmeye motive olurken, App Inventor'ın istihdam piyasası tarafından talep edilen bir dil olmaması durumu, kişisel amaçlı uygulama geliştirmek için mobil araçlarla çalışmanın yaratabileceği isteğe darbe vurmuş olabilir. Bu durum, Soares'in çalışmasında (Soares, 2013) aslen programlama bloklarının ardında gizlenmiş Java kodları barındıran App Inventor'ın teknik imkanlarının arttırılmasıyla metin-temelli bu dile geçiş için bir köprü olarak kullanılabileceği fikrini doğrular niteliktedir, zira Sun Microsystems'a ait Java günümüz bilişim endüstrisinde yaygın kullanım alanına sahip profesyonel bir dildir. Dolayısıyla, mobil cihazlara yönelik blok temelli bu aracın, metin tabanlı bir dile doğrudan geçiş için bu şekilde kullanılması durumunda öğrencilerin eğitsel uygulamalara daha büyük ilgiyle yaklaşması sağlanabilir. Böylelikle, başlangıç seviyesinde kullanılacak dilin tercihinde pedagojik hususların birincil öncelikle göz önüne alınmasını, ancak dilin profesyonel anlamda bir değerinin de bulunmasını savunan Jenkins'in (Jenkins, 2002, 2004) fikriyle tutarlı App Inventor – Java melezi (Soares, 2013) bir eğitim modeli oluşabilir.

Tartışılan tüm bu hususlar, App Inventor kullanan deneyimli öğrencilerde motivasyon açısından neden herhangi bir artış görülmediği durumuna açıklık getirmeye yöneliktir. Ancak bunların hiçbiri, hem üniversite öncesinde, hem de üniversitede metin-temelli programlama dillerinin kullanımında başarılı bulunmuş öğrencilerin, blok temelli aracı kullanırken en az metin-temelli dili kullanan akranları kadar başarılı olamadığı sorusunu cevaplamamaktadır. İşte tam da bu noktada, karmaşık sözdizimine sahip metin tabanlı

dillerin programlama eğitimine etkisi hususundan bahsetmek faydalı olacaktır. Lise seviyesindeki programlama eğitiminde kullanılan dilin karmaşık bir sözdizimine sahip olması, öğrencilerin programlama temel mantığını öğrenmesini güçleştirmekte (Grandell, Peltomäki, Back, & Salakoski, 2006) ve daha ziyade sözdizimine dair kuralları “ezberlemeye” eğilim göstermesine neden olmaktadır. Acemilere yönelik eğitimde kullanılan programlama dillerinin sahip olması gereken özellikler incelendiğinde (McIver & Conway, 1996) Visual C# da dahil olmak üzere çoğu popüler nesne-yönelimli dilin nispeten karmaşık bir sözdizimine sahip olduğu fakat buna rağmen ısrarla programlama eğitiminde kullanıldığı görülmektedir. Dolayısıyla, metin tabanlı Visual C# dilinin kullanımında başarılı öğrencilerin, mevcut bilgilerini App Inventor’a aktarmakta güçlük çekmesi ve neticede başarısız oluşu bundan kaynaklanıyor olabilir. Bu sonuçlar Winslow’un tanımlamış olduğu programlamada acemi öğrenciler tarafından yaşanan sorunlarla büyük benzerlik göstermektedir (Winslow, 1996). Hem üniversite öncesinde hem üniversite seviyesinde programlama eğitimi almış bu öğrencilerin, halen acemilerde görülen sorunlarla boğuşuyor olması ise programlama eğitiminin başarısızlığa uğradığının göstergesi olabilir.

Dahası, Milne ve arkadaşlarının çalışmasında programlama öğrencilerinin, bilhassa deneyimleri arttıkça, kendi kavrama becerilerini aslında olduğundan daha iyi olarak algıladığı; bu nedenle kendilerine sunulan programlama dersi konularını küçümsediği ve tam olarak anlamadığı halde anladığını zannettiği rapor edilmiştir (Milne & Rowe, 2002). Blok-programlamanın deneyimli öğrenciler tarafından “fazla kolay” olarak algılanmış olmasına rağmen başarının düşük bulunması bu fenomenin varlığına işaret ediyor olabilir.

Öğrencilerin MBTGPD ile çalışırken motivasyonlarına olumlu etkide bulunması beklenen bir diğer faktör olan “akıllı mobil cihaz aidiyetinin”, akademik başarı ve motivasyona ait “görev değeri” alt boyutu üzerinde herhangi bir etkisi olup olmadığının da incelenmesi öngörülmüştür. Ancak yürütülen analize göre, MBTGPD ile eğitim alan öğrencilerde kendine ait mobil cihazı olan ve olmayanlar arasında ne görev değeri algısı ne de akademik başarı açısından anlamlı bir fark yoktur. Yani beklenenin aksine; öğrencinin kendi mobil cihazına sahip olması onun; mobil platformda uygulama geliştirmeye yönelik MBTGPD ile verilen dersi kendisi için daha değerli olarak görmesine neden olmamıştır. Akademik başarı açısından da; MBTGPD kullanan

öğrencilerin başarısını cihaz aidiyeti ile anlamlı şekilde ilişkilendirecek bir veriye rastlanamamıştır. Bu durum; Jones ve arkadaşlarının mobil cihazlarda aidiyet hususunun derse dönük motivasyonu arttırdığını iddia eden çalışmasına ait sonuçlarla ters düşmektedir (Jones ve diğ., 2006). Ancak bu noktada, öğrencilerin akıllı mobil cihaz sahibi oldukları halde derse çalışmak için App Inventor özelinde kendi cihazlarını mı yoksa IDE'nin bir parçası olan emülatörü mü kullandıkları sorusuna kesin bir cevap verilemeyeceği için bu hususun daha derinlemesine incelenmesi gerekebilir.

Araştırmanın bir diğer amacı; çalışma grubunu BÖTE öğrencilerinin oluşturduğu düşünüldüğünde; geleceğin öğretim teknolojisi ve bilişim öğretmeni adayları olan; yani gelecekte hem mobil platformlar için eğitsel içeriği üretme hem de K12 seviyesinde programlama eğitimi verme görevini üstlenecek bireylerin blok-temelli aracı mesleki yönden nasıl algıladıklarını belirlemektir.

Bu doğrultuda uygulanan anketin sonuçları incelendiğinde; MBTGPD ile alınan dersin üzerinden geçen bir yıllık süre içerisinde öğrencilerin çok az bir kısmının kişisel uygulamalar veya projeler geliştirmek için MTPD yerine MBTGPD kullandığı anlaşılmıştır. Bu noktada kişisel uygulamalar geliştirme ve günlük hayatta kullanım açısından öğretmen adaylarının App Inventor aracını benimsemediği görülebilir.

İkinci bir dil olan MBTGPD ile eğitim alan öğrencilerin, yeni bir dili öğrenebileceklerine dair inançlarının ise görece azaldığı tespit edilmiştir. Yalnızca metin temelli dil ile eğitim görmüş öğrencilerin ise “bir dili bilen hepsini öğrenir, çünkü sözdizimi kuralları dışında pek farkları yok” şeklinde özetlenebilecek beyanatlarda bulunduğu durumu da not düşülmelidir. Buradan çıkarılabilecek sonuç, MBTGPD'nin, mantık olarak metin temelli programlama dillerinden farklı olarak algılanıyor olduğu ve bu farklılığın, öğrencilerin yeni bir dili öğrenebileceklerine dair inançlarına olumsuz etkide bulunma ihtimalinin varlığıdır.

Öğrencilerin, acemilere yönelik programlama eğitimi verip veremeyecekleri sorusuna cevapları incelendiğinde ise; MBTGPD ile eğitim almış olmanın öğrencilerin bu konudaki inançlarını nispeten pekiştirdiği şeklinde bir sonuç göze çarpmıştır. MBTGPD ile çalışan öğrencilerin cevaplarında; GUI-programlama kavramına daha büyük bir vurgu yapılmıştır. Bu durumda, App Inventor eğitiminin GUI öncelikli bir yapıda olması etkili olduğu söylenebilir. Nitekim bu grupta öğrenciler yine zaman zaman, söz

konusu eğitimi App Inventor kullanarak verebileceklerinden bahsetmiş, ancak kaynak eksikliği hususuna değinmişlerdir. App Inventor'ın yeni bir dil olduğu ve ticari popülerliğe sahip diller kadar dokümantasyon desteğine sahip olmadığı gibi gerçekler bu durumda rol oynamış olabileceği düşünülmektedir

Son olarak, öğrencilerden “Programlamada FOR döngüsü” isimli konunun ortaokul öğrencilerine nasıl öğretilabileceği konusunda 20-dakika uzunluğunda bir günlük ders planı hazırlayarak buna uygun mikro-öğretim videosu çekmeleri istenmiştir. Öğrencilerin bu uygulamada kullanmak için hangi dili seçtikleri incelendiğinde ise; programlama dersinde MBTGPD kullanmış deney grubu öğrencilerinin, App Inventor'ı tercih etme eğiliminin kontrol grubu öğrencilerine düştüğü gözlenmiştir. Dolayısıyla bu öğrenciler her ne kadar ankette “acemilere yönelik ders verebileceklerini” vurgulamış ve bu dersi “App Inventor” kullanarak verebileceklerinden bahsetmiş olsalar da, iş uygulamaya geldiğinde yine metin temelli dilleri tercih etmişlerdir.

Bu sonuçların ortaya koyduğu tablo incelendiğinde, MIT App Inventor ile eğitim görmüş olmanın, öğrencilerin bu programlama dilini başkalarına programlama eğitimi verebilecekleri bir araç olarak algılamak gibi etkisi bulunduğu, ancak pratikte bu aracı tercih etmemek gibi bir durum olduğu görülmüştür. Bu yargı; yürütülen benzer çalışmaların sonuçlarıyla çelişmektedir. Örneğin Fesakis ve Serafeim, bilişim öğretmeni adaylarının Scratch isimli blok programlama dilini, K12 seviyesinde eğitim için mesleki anlamda nasıl algıladığını incelemişlerdir ve Scratch ile çalışmış öğrencilerin, eğitimde blok-programlamanın kullanılması durumuna olumlu baktığı sonucuna varmışlardır (Fesakis & Serafeim, 2009). Yine Scratch kullanarak bu çalışmanın bir benzerini yürüten Choi de aynı sonucu doğrulayan sonuçlar elde etmiştir (Choi, 2012). Ancak mevcut çelişki; çalışmalara konu olan blok programlama dili Scratch'in App Inventor ile hangi noktalarda benzer olduğu / farklılaştığı hususu derinlemesine incelenerek yorumlanmalıdır. Diğer yandan, literatürde bulunan bu çalışmaların mikroöğretim gibi uygulamalara yer vermediğine ve söz konusu blok programlama dilini başka bir dille karşılaştırma halinde olmadığına dikkat edilmelidir. Burada çıkarılması muhtemel sonuç; başlangıç seviyesinde programlama eğitimi için görsel blok temelli dillerin kullanımını bilişim öğretmeni adayları özelinde değerlendiren çalışmaların; uygulamalı ve karşılaştırmalı çalışmalar olması gerektiğidir.

5.2 Öneriler

Programlama eğitiminde acemilere sunulan ilk araç olarak App Inventor kullanmanın, eğitimde “GUI-öncelikli” yaklaşımı kullanmayı kabul etmek anlamına geldiği unutulmamalıdır.

Acemilere yönelik programlama eğitiminde metin tabanlı bir dilin kullanımı tercih edilmesi durumunda; eğitim için kullanılacak dilin olabildiğince basit bir sözdizimine sahip olması ve öğrencilerin bu sayede dile özgü kurallardan ziyade programlamanın genel mantığını öğrenmeye yoğunlaşabilmesi gerektiği hususuna dikkat edilmelidir.

MIT App Inventor Beta, metin tabanlı bir dili öğrenmede güçlük çekmiş öğrencilere alternatif bir araç olarak sunulabilir. Ancak App Inventor’ın, üniversite döneminde veya öncesinde metin tabanlı bir programlama diliyle eğitim görmüş ve başarılı bulunmuş öğrencilere bir eğitsel araç olarak sunulmasının, adaptasyon sorunlarından yahut öğrencilerin aracı benimsememesinden ötürü akademik başarıda bir düşüşe neden olabileceği not düşülmelidir. Bu hususun önüne geçilebilmesi için, App Inventor IDE’sinin kullanılabilirliğinin artırılması ve teknik yapısının Java gibi endüstri açısından benimsenmiş bir dile geçiş yapmayı olanak tanıyacak şekilde değiştirilmesi önerilmektedir.

Bu çalışmada, App Inventor’ın öğrencilerde derse yönelik motivasyon açısından yaratabileceği olumlu veya olumsuz etkiler kullanılan ölçü aracıyla tam olarak belirlenememiştir. Bu nedenle bu programlama dilinin motivasyon üzerindeki etkisinin inceleme amacıyla gelecekte yürütülebilecek araştırmalarda farklı araçlar ve yöntemler kullanılması önerilmektedir.

Diğer yandan; başlangıç seviyesinde programlama eğitimi için görsel blok temelli dillerin kullanımının bilişim öğretmeni adayları tarafından nasıl algılandığını inceleyen araştırmaların; uygulamalı ve karşılaştırmalı çalışmalar olması gerektiği söylenebilir.

KAYNAKLAR

- Ahmad, K., & Gestwicki, P. (2013). *Studio-based learning and app inventor for android in an introductory CS course for non-majors*. Paper presented at the Proceeding of the 44th ACM technical symposium on Computer science education, Denver, Colorado, USA.
- Allison, I., Orton, P., & Powell, H. (2002). *A virtual learning environment for introductory programming*. Paper presented at the Proceedings of the 3rd Conference of the LTSN-ICS.
- Ambler, A. L., Burnett, M. M., & Zimmerman, B. A. (1992). Operational versus definitional: a perspective on programming paradigms. *Computer*, 25(9), 28-43. doi: 10.1109/2.156380
- Armaneri, Ö., & Armaneri, S. (2006). Çapraz Tasarımın Klinik Araştırmalarda Uygulanması. *DEÜM Fakültesi FENVEM Dergisi*, 8(3), 51-64.
- Astrachan, O., Bruce, K., Koffman, E., K, M., #246, lling, & Reges, S. (2005). Resolved: objects early has failed. *SIGCSE Bull.*, 37(1), 451-452. doi: 10.1145/1047124.1047359
- Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *SIGCSE Bull.*, 39(2), 32-36. doi: 10.1145/1272848.1272879
- Bhagi, A. (2012). *Android Game Development with AppInventor*. (Master of Engineering in Electrical Engineering and Computer Science), Massachusetts Institute of Technology, Massachusetts, USA.
- Bruce, K. B. (2004). Controversy on how to teach CS 1: a discussion on the SIGCSE-members mailing list. *SIGCSE Bull.*, 36(4), 29-34. doi: 10.1145/1041624.1041652
- Buck, D., & Stucki, D. J. (2000). Design early considered harmful: graduated exposure to complexity and structure based on levels of cognitive development. *SIGCSE Bull.*, 32(1), 75-79. doi: 10.1145/331795.331817
- Burd, B., Jo, #227, Barros, o. P., Johnson, C., Kurkovsky, S., . . . Tillman, N. (2012). *Educating for mobile computing: addressing the new challenges*. Paper presented at the Proceedings of the final reports on Innovation and technology in computer science education 2012 working groups, Haifa, Israel.
- Burton, P. J., & Bruhn, R. E. (2003). Teaching programming in the OOP era. *ACM SIGCSE Bulletin*, 35(2), 111-114.
- Büyüköztürk, Ş. (2002). Sosyal bilimler için veri analizi el kitabı. *Ankara: PegemA Yayıncılık*.
- Büyüköztürk, Ş., Akgün, Ö. E., Özkahveci, Ö., & Demirel, F. (2004). Güdülenme ve öğrenme stratejileri ölçeğinin Türkçe formunun geçerlik ve güvenirlik çalışması. *Kuram ve Uygulamada Eğitim Bilimleri*, 4(2), 207-239.
- Capretz, L. F. (2003). A brief history of the object-oriented approach. *SIGSOFT Softw. Eng. Notes*, 28(2), 6. doi: 10.1145/638750.638778

- Choi, H. (2012). Learners' reflections on computer programming using Scratch: Korean primary pre-service teachers' perspective. [Conference Paper]. *ICoME2012 Proceedings*.
- Cooper, S., Dann, W., & Pausch, R. (2003). Teaching objects-first in introductory computer science. *SIGCSE Bull.*, 35(1), 191-195. doi: 10.1145/792548.611966
- Cornforth, D. (2014). Teaching Mobile Apps for Windows Devices Using TouchDevelop.
- Davis, H., Carr, L., Cooke, E., & White, S. (2001). Managing diversity: Experiences teaching programming principles.
- Decker, R., & Hirshfield, S. (1993). Top-down teaching: object-oriented programming in CS 1. *SIGCSE Bull.*, 25(1), 270-273. doi: 10.1145/169073.169495
- Decker, R., & Hirshfield, S. (1999). *Programming Java: An Introduction to Programming Using Java*: Brooks/Cole Publishing Co.
- Dovey, J., & Furrow, A. (2012). Object-Oriented Programming *Beginning Objective-C* (pp. 23-41): Apress.
- Dvorak, J. D., & Buchanan, K. (2002). *Using Technology To Create and Enhance Collaborative Learning*.
- Fekete, A., Kay, J., Kingston, J., & Wimalaratne, K. (2000). Supporting reflection in introductory computer science. *SIGCSE Bull.*, 32(1), 144-148. doi: 10.1145/331795.331844
- Fenwick Jr, J. B., Kurtz, B. L., & Hollingsworth, J. (2011). *Teaching mobile computing and developing software to support computer science education*. Paper presented at the Proceedings of the 42nd ACM technical symposium on Computer science education.
- Fesakis, G., & Serafeim, K. (2009). *Influence of the familiarization with scratch on future teachers' opinions and attitudes about programming and ICT in education*. Paper presented at the ACM SIGCSE Bulletin.
- Fink, D. T. (2013). Bilişsel Alan Sınıflamasında Yenilenmiş Bloom, SOLO, Fink, Dettmer Taksonomileri ve Uluslararası Alanda Tanınma Durumları Asım ARI.
- Forehand, M. (2010). Bloom's taxonomy. *Emerging perspectives on learning, teaching, and technology*, 41-47.
- Fujiwara, K., Fushida, K., Tamada, H., Igaki, H., & Yoshida, N. (2012, 26-27 Oct. 2012). *Why Novice Programmers Fall into a Pitfall?: Coding Pattern Analysis in Programming Exercise*. Paper presented at the Empirical Software Engineering in Practice (IWESEP), 2012 Fourth International Workshop on.
- Fuller, U., Johnson, C. G., Ahoniemi, T., Cukierman, D., Hern, I., #225, . . . Thompson, E. (2007). Developing a computer science-specific learning taxonomy. *SIGCSE Bull.*, 39(4), 152-170. doi: 10.1145/1345375.1345438
- Garcia, T., & Pintrich, P. R. (1996). Assessing students' motivation and learning strategies in the classroom context: The motivated strategies for learning questionnaire *Alternatives in assessment of achievements, learning processes and prior knowledge* (pp. 319-339): Springer.

- Gerek, S., & Kurt, A. A. (2010). Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümlerinde Ekonomi Okuryazarlığına İlişkin Göstergeler. *University of Gaziantep Journal of Social Sciences*, 9(1).
- Gibbons, J. (1998). Structured programming in Java. *ACM SIGPLAN NOTICES*, 33, 40-43.
- Grandell, L., Peltomäki, M., Back, R.-J., & Salakoski, T. (2006). *Why complicate things?: introducing programming in high school using Python*. Paper presented at the Proceedings of the 8th Australasian Conference on Computing Education-Volume 52.
- Gray, J., Abelson, H., Wolber, D., & Friend, M. (2012). *Teaching CS principles with app inventor*. Paper presented at the Proceedings of the 50th Annual Southeast Regional Conference.
- Hadjerrouit, S. (1998). Java as first programming language: a critical evaluation. *ACM SIGCSE Bulletin*, 30(2), 43-47.
- Hagan, D., & Markham, S. (2000). Does it help to have some programming experience before beginning a computing degree program? *SIGCSE Bull.*, 32(3), 25-28. doi: 10.1145/353519.343063
- Han Lun, T., Chin Chin, W., Kho, J., & Hoh, S. (2008, 12-14 Oct. 2008). *Simple Mobility Configurator: Block Programming for the Non-Experts*. Paper presented at the Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on.
- Horn, M. S., & Jacob, R. J. K. (2006). *Tangible programming in the classroom: a practical approach*. Paper presented at the CHI '06 Extended Abstracts on Human Factors in Computing Systems, Montré#233;al, Qu#233;bec, Canada.
- Horn, M. S., & Jacob, R. J. K. (2007). *Designing tangible programming languages for classroom use*. Paper presented at the Proceedings of the 1st international conference on Tangible and embedded interaction, Baton Rouge, Louisiana.
- Jaccard, J. (1998). *Interaction effects in factorial analysis of variance*: Sage.
- Jane, Y., & Mike, J. (2004). *Introducing Java: the Case for Fundamentals-first*. Paper presented at the Proceedings of the International Conference on Education and Information Systems, Technologies and Applications (EISTA 2004), Orlando, USA.
- Jenkins, T. (2002). *On the difficulty of learning to program*. Paper presented at the Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences.
- Jenkins, T. (2004). The First Language-A Case for Python? *Innovation in Teaching and Learning in Information and Computer Sciences*, 3(2).
- Jenkins, T., & Davy, J. (2002). Diversity and motivation in introductory programming. *Innovation in Teaching and Learning in Information and Computer Sciences*, 1(1).

- Jiang, K., Maniotes, J., & Kamali, R. (2004). *A different approach of teaching introductory visual basic course*. Paper presented at the Proceedings of the 5th conference on Information technology education, Salt Lake City, UT, USA.
- Jones, A., Issroff, K., Scanlon, E., Clough, G., & Mcandrew, P. (2006). *Using mobile devices for learning in Informal Settings: Is it Motivating*. Paper presented at the IADIS International conference Mobile Learning.
- Karakus, M., Uludag, S., Guler, E., Turner, S. W., & Ugur, A. (2012, 21-23 June 2012). *Teaching computing and programming fundamentals via App Inventor for Android*. Paper presented at the Information Technology Based Higher Education and Training (ITHET), 2012 International Conference on.
- Karasar, N. (2012). *Bilimsel araştırma yöntemi: kavramlar, ilkeler, teknikler*: Nobel Yayın Dağıtım.
- Kasurinen, J., Purmonen, M., & Nikula, U. (2008). *A study of visualization in introductory programming*. Paper presented at the Proc. 20th annual Meeting of Psychology of Programming Interest Group, Lancaster, UK.
- Kelleher, C., Pausch, R., & Kiesler, S. (2007). *Storytelling alice motivates middle school girls to learn computer programming*. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.
- Keselman, H. J., Huberty, C. J., Lix, L. M., Olejnik, S., Cribbie, R. A., Donahue, B., . . . Levin, J. R. (1998). Statistical Practices of Educational Researchers: An Analysis of their ANOVA, MANOVA, and ANCOVA Analyses. *Review of Educational Research*, 68(3), 350-386. doi: 10.3102/00346543068003350
- Kinnunen, P., & Malmi, L. (2008). *CS minors in a CSI course*. Paper presented at the Proceedings of the Fourth international Workshop on Computing Education Research.
- Klassen, M. (2006). *Visual approach for teaching programming concepts*. Paper presented at the Proceedings of the 9th International Conference on Engineering Education (ICEE 2006).
- Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., . . . Wiedenbeck, S. (2011). The state of the art in end-user software engineering. *ACM Comput. Surv.*, 43(3), 1-44. doi: 10.1145/1922649.1922658
- Kölling, M. (1999). The problem of teaching object-oriented programming, Part 1: Languages. *Journal of Object-oriented programming*, 11(8), 8-15.
- Kölling, M., & Rosenberg, J. (1996). An object-oriented program development environment for the first programming course. *SIGCSE Bull.*, 28(1), 83-87. doi: 10.1145/236462.236514
- Kurkovsky, S. (2009). Engaging students through mobile game development. *SIGCSE Bull.*, 41(1), 44-48. doi: 10.1145/1539024.1508881
- Lahtinen, E., Ala-Mutka, K., J, H.-M., & rvinen. (2005). A study of the difficulties of novice programmers. *SIGCSE Bull.*, 37(3), 14-18. doi: 10.1145/1151954.1067453

- Lewis, J. (2000). Myths about object-orientation and its pedagogy. *ACM SIGCSE Bulletin*, 32(1), 245-249.
- Liberty, J. (2005). *Programming C#: Building .NET Applications with C#*. " O'Reilly Media, Inc."
- Mahmoud, Q. H., & Dyer, A. (2007). *Integrating BlackBerry wireless devices into computer programming and literacy courses*. Paper presented at the Proceedings of the 45th annual southeast regional conference, Winston-Salem, North Carolina.
- Mahmoud, Q. H., & Popowicz, P. (2010). *A mobile application development approach to teaching introductory programming*. Paper presented at the Frontiers in Education Conference (FIE), 2010 IEEE.
- Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004, 29-30 Jan. 2004). *Scratch: a sneak preview [education]*. Paper presented at the Creating, Connecting and Collaborating through Computing, 2004. Proceedings. Second International Conference on.
- Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *Trans. Comput. Educ.*, 10(4), 1-15. doi: 10.1145/1868358.1868363
- Mansfield-Devine, S. (2012). Paranoid Android: just how insecure is the most popular mobile platform? *Network Security*, 2012(9), 5-10.
- Martinez, W. L. (2011). Graphical user interfaces. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(2), 119-133. doi: 10.1002/wics.150
- McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., . . . Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, 33(4), 125-180.
- McIver, L., & Conway, D. (1996). *Seven deadly sins of introductory programming language design*. Paper presented at the Software Engineering: Education and Practice, 1996. Proceedings. International Conference.
- McNerney, T. (2004). From turtles to Tangible Programming Bricks: explorations in physical language design. *Personal and Ubiquitous Computing*, 8(5), 326-337. doi: 10.1007/s00779-004-0295-6
- Meisalo, V., Suhonen, J., Torvinen, S., & Sutinen, E. (2002). *Formative evaluation scheme for a web-based course design*. Paper presented at the ACM SIGCSE Bulletin.
- Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. *Education and Information technologies*, 7(1), 55-66.
- Mohamad, S. N. H., Patel, A., Tew, Y., Latih, R., & Qassim, Q. (2011, 20-23 March 2011). *Principles and dynamics of block-based programming approach*. Paper presented at the Computers & Informatics (ISCI), 2011 IEEE Symposium on.

- Navarro-Prieto, R., & Cañas, J. J. (2001). Are visual programming languages better? The role of imagery in program comprehension. *International Journal of Human-Computer Studies*, 54(6), 799-829.
- Papert, S. (1980). Part 2: LEGO/LOGO Projects. *Constructionism vs. Instructionism*. Retrieved 01.17, 2013, from http://www.papert.org/articles/const_inst/const_inst2.html
- Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, 1-11.
- Proulx, V. K. (2000). Programming patterns and design patterns in the introductory computer science course. *SIGCSE Bull.*, 32(1), 80-84. doi: 10.1145/331795.331819
- Proulx, V. K., Raab, J., & Rasala, R. (2002). Objects from the beginning - with GUIs. *SIGCSE Bull.*, 34(3), 65-69. doi: 10.1145/637610.544436
- Purcell, K. (2011). Half of adult cell phone owners have apps on their phones. *Pew Research Center's Internet & American Life Project*. Accessed January, 9, 2012.
- Rau, P.-L. P., Gao, Q., & Wu, L.-M. (2008). Using mobile communication technology in high school education: Motivation, pressure, and learning performance. *Computers & Education*, 50(1), 1-22. doi: <http://dx.doi.org/10.1016/j.compedu.2006.03.008>
- Resnick, M., Maloney, J., Andr, #233, Monroy-Hern, s., #225, . . . Kafai, Y. (2009). Scratch: programming for all. *Commun. ACM*, 52(11), 60-67. doi: 10.1145/1592761.1592779
- Riley, D. (2012). *Using mobile phone programming to teach Java and advanced programming to computer scientists*. Paper presented at the Proceedings of the 43rd ACM technical symposium on Computer Science Education.
- Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 137-172.
- Roy, K. (2012). *App inventor for android: report from a summer camp*. Paper presented at the Proceedings of the 43rd ACM technical symposium on Computer Science Education.
- Röbling, G., & Freisleben, B. (2000). *Experiences in using animations in introductory computer science lectures*. Paper presented at the ACM SIGCSE Bulletin.
- Sanders, I., & Mueller, C. (2000). *A fundamentals-based curriculum for first year computer science*. Paper presented at the ACM SIGCSE Bulletin.
- Sandoval-Reyes, S., Galicia-Galicia, P., & Gutierrez-Sanchez, I. (2011, 15-18 Nov. 2011). *Visual Learning Environments for Computer Programming*. Paper presented at the Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2011 IEEE.
- Sarif, S. N., Idris, S., & Zin, A. M. (2011). *The design of blocks integration tool to support end-user programming*. Paper presented at the Electrical Engineering and Informatics (ICEEI), 2011 International Conference on.
- Shackelford, R. L., & LeBlanc Jr, R. J. (1997). *Introducing computer science fundamentals before programming*. Paper presented at the Frontiers in Education

- Conference, 1997. 27th Annual Conference. Teaching and Learning in an Era of Change. Proceedings.
- Smolarski, D. C. (2003). A first course in computer science: Languages and goals. *Teaching Mathematics and Computer Science*, 1(1), 137-152.
- Soares, A. (2013). *Reflections on Teaching App Inventor for Non-Beginner Programmers: Issues, Challenges and Opportunities*. Paper presented at the Proceedings of the Information Systems Educators Conference ISSN.
- Spertus, E., Chang, M. L., Gestwicki, P., & Wolber, D. (2010). *Novel approaches to CS 0 with app inventor for android*. Paper presented at the Proceedings of the 41st ACM technical symposium on Computer science education.
- Stroustrup, B. (1988). What is object-oriented programming? *Software, IEEE*, 5(3), 10-20. doi: 10.1109/52.2020
- Stroustrup, B. (1994). *The design and evolution of C++*: ACM Press/Addison-Wesley Publishing Co.
- Tanrikulu, E., & Schaefer, B. C. (2011). The users who touched the ceiling of scratch. *Procedia-Social and Behavioral Sciences*, 28, 764-769.
- Tillmann, N., Moskal, M., Halleux, J. d., Fahndrich, M., Bishop, J., Samuel, A., & Xie, T. (2012). *The future of teaching programming is on mobile devices*. Paper presented at the Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education, Haifa, Israel.
- Tutkun, O. F., Guzel, D., Koroğlu, M., & Ilhan, H. (2012). Bloom's Revized Taxonomy and Critics on It. *The Online Journal of Counselling and Education*, 23.
- Uludag, S., Karakus, M., & Turner, S. W. (2011). *Implementing IT0/CS0 with scratch, app inventor for android, and lego mindstorms*. Paper presented at the Proceedings of the 2011 conference on Information technology education.
- Utting, I., Cooper, S., Kölling, M., Maloney, J., & Resnick, M. (2010). Alice, Greenfoot, and Scratch--a discussion. *ACM Transactions on Computing Education (TOCE)*, 10(4), 17.
- Van Roy, P. (2009). Programming paradigms for dummies: What every programmer should know. *New Computational Paradigms for Computer Music*, 9.
- Van Roy, P., Armstrong, J., Flatt, M., & Magnusson, B. (2003). *The role of language paradigms in teaching programming*. Paper presented at the ACM SIGCSE Bulletin.
- Vilner, T., Zur, E., & Gal-Ezer, J. (2007). Fundamental concepts of CS1: procedural vs. object oriented paradigm - a case study. *SIGCSE Bull.*, 39(3), 171-175. doi: 10.1145/1269900.1268835
- Vogts, D., Calitz, A., & Greyling, J. (2008). *Comparison of the effects of professional and pedagogical program development environments on novice programmers*. Paper presented at the Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology.

- Wang, D., Zhang, C., & Wang, H. (2011). *T-Maze: a tangible programming tool for children*. Paper presented at the Proceedings of the 10th International Conference on Interaction Design and Children, Ann Arbor, Michigan.
- Weiser, M. (1993). Hot topics-ubiquitous computing. *Computer*, 26(10), 71-72. doi: 10.1109/2.237456
- Wiedenbeck, S., Ramalingam, V., Sarasamma, S., & Corritore, C. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting with Computers*, 11(3), 255-282.
- Winslow, L. E. (1996). Programming pedagogy—a psychological overview. *SIGCSE Bull.*, 28(3), 17-22. doi: 10.1145/234867.234872
- Wolber, D. (2011a). *App inventor and real-world motivation*. Paper presented at the Proceedings of the 42nd ACM technical symposium on Computer science education, Dallas, TX, USA.
- Wolber, D. (2011b). *App inventor and real-world motivation*. Paper presented at the Proceedings of the 42nd ACM technical symposium on Computer science education.
- Wyeth, P., & Purchase, H. C. (2000, 2000). *Programming without a computer: a new interface for children under eight*. Paper presented at the User Interface Conference, 2000. AUIC 2000. First Australasian.
- Yau, J. (2004). Learning Objects and the teaching of Java programming. *Unpublished thesis, University*.

EKLER

- Ek 1: Ön Bilgi Testi 1
- Ek 2: Ön Bilgi Testi 2
- Ek 3: Uygulama Sınavı 1
- Ek 4: Uygulama Sınavı 2
- Ek 5: Uygulama Sonu Anketi
- Ek 6: Örnek App Inventor Projesi
- Ek 7: App Inventor ile Hazırlanmış Öğrenci Projeleri

EK 1: ÖNBİLGİ TESTİ

BÖLÜM A: Aşağıdaki programların çıktılarını yazınız.

1.

```
static void Main(string[] args)
{
    int i, a = 3;
    for (i = 1; i < 10; i++)
        if (i % a == 0)
            Console.WriteLine("{0} ", i);
}
```

2.

```
static void Main(string[] args)
{
    int a=0, b=1, c=2;
    a = a+c; b=a+c; c=a+b;
    Console.WriteLine("{0}{1}{2}", c, b, a);
}
```

3.

```
static void Main(string[] args)
{
    Char a = 'b', b = 'A', c = 'c';
    if (Char.IsLower(a))
        a = Char.ToUpper(b)
    else if (Char.IsUpper(b))
        b = Char.ToUpper(a);
    c = Char.ToUpper(c);
    Console.WriteLine("{0}{1}{2}", a, c, b);
}
```

4.

```
class Program
{
    static public int abc(int a, int b)
    {
        return a * b;
    }
    static void Main(string[] args)
    {
        int a = 2, b = 3;
        Console.WriteLine("{0}", abc(a, b) + abc(b, b));
    }
}
```

5.

```
static void Main(string[] args)
{
    int a = 3, b = 3, c = 4;
    if (a >= b)
        if (a <= c)
            Console.WriteLine("{0}{1}", a, c);
        else if (b < c)
            if (a < b)
                Console.WriteLine("{0}{1}", b, a);
            else Console.WriteLine("{0}{1}", c, b);
    }
}
```

6.

```
static void Main(string[] args)
{
    int a = 2, b = 1, c = 3;
    if (a >= b)
        Console.WriteLine("{0}{1}", a, b);
    if (a >= c)
        Console.WriteLine("{0}{1}", a, c);
}
```

BÖLÜM B: Aşağıdaki programlar çalışmakta fakat istenilen işlevi yerine getirmemektedir. Programların hatalarını düzelterek çalışır hale getiriniz.

1. Verilen üç sayının bir üçgenin kenar uzunlukları olup olamayacağını test eden program. (İpucu: Bir üçgende herhangi iki kenarın uzunlukları toplamı üçüncü kenarın uzunluğundan daha büyük olmalıdır.)

```
static void Main(string[] args)
{
    int a, b, c;
    a = int.Parse(Console.ReadLine());
    b = int.Parse(Console.ReadLine());
    c = int.Parse(Console.ReadLine());
    if (a + b > c && a + c > b)
        Console.WriteLine("Evet");
    Console.WriteLine("Hayır");
}
```

2. Girilen sayı sıfırdan büyükse 'Acik', sıfır ve sıfırdan küçükse 'Kapali' yazan program.

```
static void Main(string[] args)
{
    int a;
    a = int.Parse(Console.ReadLine());
    if (a < 0)
        Console.WriteLine("Kapali");
    Console.WriteLine("Acik");
}
```

3. Girilen iki sayının ikisinin de sıfıra eşit olup olmadığını kontrol eden program.

```
static void Main(string[] args)
{
    int a, b;
    a = int.Parse(Console.ReadLine());
    b = int.Parse(Console.ReadLine());
    if (a * b == 0)
        Console.WriteLine("Esit");
    else
        Console.WriteLine("Esit Degil");
}
```

4. Esc tuşuna basılana kadar klavyeden basılan her tuş için haftanın bir gününü random üreten ve ekrana yazan program.

```
static void Main(string[] args)
{
    Random rand = new Random();
    string str = "";
    while (str != "0")
    {
        str = Console.ReadLine();
        int s = rand.Next(1, 8);
        switch (s)
        {
            case 1: Console.WriteLine("Pazartesi"); break;
            case 2: Console.WriteLine("Sal "); break;
            case 3: Console.WriteLine("carsamba"); break;
            case 4: Console.WriteLine("Persembe"); break;
            case 5: Console.WriteLine("Cuma"); break;
            case 6: Console.WriteLine("Cumartesi"); break;
            case 7: Console.WriteLine("Pazar"); break;
        }
    }
}
```

5. Üç basamaklı bir sayının basamakları toplamını yazan program.

```
static void Main(string[] args)
{
    int a;
    a = int.Parse(Console.ReadLine());
    Console.WriteLine("{0}", a/100 + (a/10)%100 + a/10);
}
```

6. Girilen bir sayının kaç basamaklı olduğunu yazan program.

```

static void Main(string[] args)
{
    int a, b=1;
    a=int.Parse(Console.ReadLine());
    while (a > 0)
    {
        a = a / 10;
        ++b;
    }
    Console.WriteLine("{0}", b);
}

```

BÖLÜM C: Aşağıdaki programlar çalışmakta ve istenilen işlevi yerine getirmektedir. Fakat her bir program daha hızlı çalışabilir veya daha kısa yazılabilir. Programlarda yapılan mantık hatalarını yazınız.

1. Bir cümlede kaç adet 'a' ve kaç adet 'b' harfi olduğunu ayrı ayrı yazan program.

```

static void Main(string[] args)
{
    string str;
    int i, top=0;
    Console.WriteLine("Cümleyi giriniz:");
    str = Console.ReadLine();
    for (i=0; i<str.Length; i++)
        if (str[i]=='a')
            top++;
    Console.WriteLine("{0} adet a vardır.\n", top);
    top = 0;
    for (i = 0; i < str.Length; i++)
        if (str[i] == 'b')
            top++;
    Console.WriteLine("{0} adet b vardır.\n", top);
}

```

2. Girilen üç sayıdan en küçüğünü yazdıran program.

```

static void Main(string[] args)
{
    int a,b,c;
    a = int.Parse(Console.ReadLine());
    b = int.Parse(Console.ReadLine());
    c = int.Parse(Console.ReadLine());
    if (a<b && a<c)
        Console.WriteLine("{0}", a);
    else if (b < a && b < c)
        Console.WriteLine("{0}", b);
    else if (c < a && c < b)
        Console.WriteLine("{0}", c);
}

```

3. 10 elemanı bulunan a ve b dizilerinin birbirinin tersi olup olmadığını test eden program.

```

static void Main(string[] args)
{
    int[] a = new int[10] {1,3,2,4,3,5,1,0,9,2};
    int[] b = new int[10] {2,9,1,0,5,3,4,2,3,1};
    int i, flag = 1;
    for (i = 0; i < 10; i++)
        if (a[i] != b[9 - i])
            flag = 0;
    if (flag == 1)
        Console.WriteLine("Evet");
    else Console.WriteLine("Hayır");
}

```

4. Klavyeden girilen iki sözlü ve üç yazılı notundan en yüksek sözlü ve en yüksek yazılı notunu yazan program.

```

class Program
{
    static int max1(int a, int b)
    {

```

```

        if (a > b)
            return a;
        return b;
    }
    static int max2(int a, int b, int c)
    {
        if (a > b && a > c)
            return a;
        if (b > c)
            return b;
        return c;
    }
}

```

```

static void Main(string[] args)
{
    int soz1,soz2,yaz1,yaz2,yaz3;
    Console.WriteLine("İki sözlü notu:");
    soz1 = int.Parse(Console.ReadLine());
    soz2 = int.Parse(Console.ReadLine());
    Console.WriteLine("Üç yazılı notu:");
    yaz1 = int.Parse(Console.ReadLine());
    yaz2 = int.Parse(Console.ReadLine());
    yaz3 = int.Parse(Console.ReadLine());
    Console.WriteLine("En büyük sözlü notu: {0}\n", max1(soz1,soz2));
    Console.WriteLine("En büyük yazılı notu: {0}\n", max2(yaz1, yaz2, yaz3));
}

```

5. Girilen bir sayının mutlak değerini yazan program. Bir sayının mutlak değeri sıfır veya sıfırdan büyük bir sayıdır. ($|-5| = 5$; $|5| = 5$; $|0| = 0$)

```

static void Main(string[] args)
{
    int sayi,sonuc;
    sayi = int.Parse(Console.ReadLine());
    if (sayi < 0)
        sonuc = 0;
    else if (sayi < 0)
        sonuc = -sayi;
    else sonuc = sayi;
    Console.WriteLine("{0}", sonuc);
}

```

6. Klavyeden girilen sayıları tersten yazdıran program.

```

static void Main(string[] args)
{
    int[] a = new int[100];
    int i, n,temp;
    Console.WriteLine("Kaç sayı?");
    n = int.Parse(Console.ReadLine());
    Console.WriteLine("Sayıları giriniz:");
    for (i = 0; i < n; i++)
        a[i] = int.Parse(Console.ReadLine());
    for (i=0; i<n/2; i++)
    {
        temp = a[i];
        a[i] = a[n-i-1];
        a[n-i-1] = temp;
    }
    for (i=0; i<n; i++)
        Console.WriteLine("{0}", a[i]);
    Console.ReadKey();
}

```

EK 2: ÖN BİLGİ TESTİ 2

Bu oyun oyuncu ile bilgisayar arasında geçmektedir. Oyuncuya 1-13 aralığındaki sayılardan oluşan 3 adet kart verilir. Benzer şekilde bilgisayara da 3 adet kart verilir. Oyuncu isterse kart al butonu ile bir kart daha alabilir. Oyuncu aç butonuna bastığında hem oyuncunun hem de bilgisayar tüm kartları açılır. Kart toplamı 19 sayısına yakın olan oyuncu oyunu kazanır.

- 1)Oyuncunun ismi sorulmalı, İsmi Gir butonuna basılarak ekrana yazdırılabilmeli ancak bu aşamada ekranda herhangi bir kart görülmemelidir (Şekil 1)
- 2)Oyuncunun kartları yeşil, bilgisayarın kartları mavi olmalı.
- 3)Oyunda Kullanılan tüm butonların yazı fontu 15 ve yazı rengi kırmızı olmalı.
- 4)**Kart Dağıt** butonuna tıklandığında 3’ er adet kart dağıtılmalı(Oyuncunun kartları üzerindeki numaralar görünüyor, yani kartlar açık ancak bilgisayarın kartlarının numaraları görünmüyor, yani bilgisayarın kartları kapalı). (Şekil 2)
- 5)Oyuncu **İsmi Gir** butonuna basılarak isim belirlenmediği sürece **kart dağıt** butonuna basılamamalı.
- 6)Oyuncu bu esnada **Kart Al** butonuna bastığında isterse bir kart daha alabilir. Oyuna 3 kart ile başlayan oyuncu böylece yeni kart çekerek toplamda en fazla 5 karta sahip olabilecektir. (Şekil 3)
- 7)**Kartları Aç** butonuna basıldığında oyuncu ve bilgisayarın kartları açılmalı ve karşılaştırılmalı. Hangi oyuncunun kart toplamı 19’a daha yakınsa o oyuncu kazanmalı. (Şekil 3)
- 8) Oyuncu kazandıysa bir açılan pencere ile tebrik edilmeli, kaybettiyse üzüntülerimiz belirtilmelidir.
- 9) Kartlar açıldıktan ve oyunun galibi belli olduktan sonra, kartlar yeniden dağıtılana dek oyuncu yeni kart al butonunu kullanamamalıdır.
- 10)Kazanılan her oyun için oyuncu, önceden “**Seçilen Puan**” ögesi ile belirlenmiş olan miktarda puan almalı, oyun kaybedilmişse aynı miktarda puan kaybetmelidir. Kartların her açılmasında biriken puanlar ekranın sağ üst köşesindeki “**toplam puan**” alanında görülebilmelidir. (Şekil 2 & 3)
- 11) Kartların her açılmasından sonra, bir kez daha kart dağıtılmadan önce “Seçilen Puan” değiştirilebilmelidir, ancak kartlar dağıtıldıktan sonra ve açılmadan önce “Seçilen Puan” kısmı değiştirilememeli, yani bu bileşen kullanılamamalıdır.
- 12) **Tekrar Oyna** butonuna basıldığında ekrana “Emin Misiniz?” şeklinde uyarı mesajı gelmeli ve “Evet” cevabı alınması durumunda kazanılan toplam puan, oyuncu ismi ve seçilen puan haneleri sıfırlanmalı ve oyun en başa dönmelidir.
- 13)Oyunun **Hakkında** ekranı olmalı ve bu ekrana geçiş mümkün olmalıdır.

Oyuncu İsmi İsmi Gir

Puan Seç Kart Dağıt

Toplam Puan: 0

Oyuncu İsmi

Hakkında Yeniden Başla Kartları Aç Kart Al

Şekil 1

Oyuncu İsmi İsmi Gir

Seçilen Puan: 50 Kart Dağıt

Toplam Puan: 0

Oyuncu İsmi

3 6 1

Kart Toplamı: 10

Hakkında Yeniden Başla Kartları Aç Kart Al

Şekil 2

Oyuncu İsmi İsmi Gir

Seçilen Puan: 50 Kart Dağıt

Toplam Puan: 0

Oyuncu İsmi

3 6 1 12

Kart Toplamı: 22

2 11 13

Kart Toplamı: 26

Hakkında Yeniden Başla Kartları Aç Kart Al

Şekil 3

EK 3: UYGULAMA SINAVI 1

Tasarım resmi aşağıda verilen programı geliştiriniz. Programın aşağıdaki özelliklere sahip olmasını sağlayınız.

- 1-)Program “**Çarpım Tablosu Oyunu**” isimli bir resim içermeli (<http://xxx.xxx.xxx> adresinden indirilebilir).
- 2-)Oyunda Arka plan rengi açık sarı olmalı.
- 3-)Kullanılan butonlar yeşil, yazı renkleri beyaz olmalı.
- 4)Oyuncu ismini girerek, “**ismi gir**” butonuna bastığında, form üzerinde isim görülebilmeli
- 5-) Oyuncunun ismi her zaman Büyük harflerle ve kırmızı renkte görülmeli.
- 6-) “**Çarpan Basamağı**” ile açılan listeden oyuncu çalışmak istediği sayıyı seçebilmeli(2-9).
- 7-)”**Başla**” butonuna basıldığında isim girilmediği sürece oyun başlayamamalı.
- 8-) Başla butonuna basıldığında çalışılmak istenen sayı seçilmediyse, kullanıcı bir mesaj ile bu konuda uyarılmalı.
- 9) Oyun başladığında, “**İsmi gir**”, “**Başla**” ve “**Çarpan Basamağı**” öğelerine artık tıklanamamalı.
- 10-)Çarpanı sor seçeneği seçildiğinde oyuncuya çarpan sorulmalı($5 \times ?=35$), çarpanı sor seçeneği seçilmediğinde sonuç ($5 \times 7=?$)sorulmalıdır.
- 11-)Oyunda sayılar rastlantısal (Random) gelmeli.
- 12-)Cevapla butonuna her tıklandığında, metin kutusundaki metin temizlenmeli
- 13-)Oyun süresi 40 saniye olup, “Kalan süre:” kısmı, 20. Saniyeden sonra geriye doğru ilerleyen bir kronometre gibi her saniyeyi göstermeli. (20.. 19.. 18.. 17.. şeklinde)
- 14-)Sürenin bitmesine 10 saniye kala süreyi gösteren rakamlar kırmızı renkte görülmeli.
- 15-)Oyuncu cevabı yazıp, “**Cevapla**” butonuna bastığında; cevap doğru ise 5 puan kazanmalı yanlış ise 5 puan kaybetmeli.
- 16-)Oyuncu puanını her zaman görebilmeli.
- 17-)Oyun süresi bittiğinde (40 sn) “**Sonuç**” isimli yeni bir ekranda oyuncunun puan durumu açıklanmalı.
- 18-)”**Sonuç**” isimli ekranda oyuncunun doğru ve yanlış cevap sayıları görülebilmeli.
- 19-) “**Sonuç**” ekranında “**Tamam**” butonu ile ekran kapanıp, birinci ekrana geri dönülebilmeli
- 20-) Ana ekrana döndüğünde, oyuncu ismi halen aynı kalmalı, puan sıfırlanmış olmalı, “çarpan basamağı:” ögesi üstündeki sayı değeri temizlenmeli, oyuncu ismi yeniden belirlenebilir olmalıdır. (Dikkat: oyuncu ismi aynı kalmıştır ancak, “çarpan değeri” seçilmemiş olduğu için “başla” butonu aktif değildir)

Logo Resmi

Çarpım Tablosu Oyunu

Oyuncu İsmi

İsmi Gir

Çarpan Basamağı: 5 ▼

☒ Çarpanı Sor

Başla

İsim

Kalan Süre: 15

Puan: 0

5 x = 30

Cevapla

EK 4: UYGULAMA SINAVI 2

I-Tasarımı aşağıda verilen bir kelimenin Türkçe ve İngilizce karşılıklarını bir veri tabanı dosyasına yazdırmak üzere geliştirilen programı oluşturunuz.

- 1.) Form rengi sarı olmalı
- 2.) Form üzerindeki tüm butonlar mavi olmalı
- 3.) Girilen kelimeler her zaman büyük harf görünmeli.
- 4.) **Ekle** butonuna basıldığında kelimeler veri tabanına eklenmeli.
- 5.) **Kelime Sayısı**, bilgisi veri tabanına eklenen kelime sayısını göstermektedir.

II-Tasarımı aşağıda verilen bir kronometre tasarlayınız

- 1.) Formun arka planında resim olmalı.
- 2.) **Başla** butonuna basıldığında buton **Dur** butonuna dönüşmeli (Başla ve Dur için tek buton olmalı)
- 3.) **Başla** butonuna basıldığında 10 dakikadan başlayarak **Dur** butonuna basılana kadar geçen süreyi geri saydırarak ekrana yazmalı.
- 4.) Zaman; 0:00(dakika:saniye) formatında , 16 punto ve mavi renkte görülmeli
- 5.) **Kayıt Listele** butonuna basıldığında açılan yeni form üzerinde **Dur** butonuna basılan tüm zamanlar listelenmeli.

III-) Sincap, kedi, fare ve kaplumbağa bir zindana düşmüşler. Kapıyı açık buldukları bu vakitte onların kaçmasına lütfen yardımcı ol! Uygulama için ihtiyacınız olacak tüm resimler ve soruyu daha iyi anlayabilmeniz için hazırlanan video, masa üstünde yer almaktadır.

- 1.) Ekranın başlığı, açıklama resminde görüldüğü gibi, “Büyük Kaçış” olmalı.
- 2.) Ekranda, içerisinde kapı resmi bulunduran 250 x 200 boyutlarında bir oyun alanı olmalı. **DİKKAT:** şekilde oyun alanı üzerinde görülen (1-2-3-4) numaralar yalnızca açıklama amacıyla verilmiştir.
- 3.) Ekranda dört adet yön butonu olmalı.)
- 4.) Ekran ilk açıldığında, size verilmiş olan “fare.png”, “kaplumbaga.png”, “sincap.png” ve “kedi.png” resimlerinden **rastgele** bir tanesi, oyun alanındaki “1” ile gösterilen konumunda bulunmalı (sol alt köşe).
- 5.) Yön butonlarına her basıldığında, ekrana gelen hayvan resmi belirlenen yönde 10 piksel hareket etmeli.
- 6.) Hareket eden hayvan resmi, butonlara basılarak oyun alanının dışına çıkamamalı.
- 7.) Hareket eden hayvan resmi, butonlar aracılığıyla kapıya varmayı başardığında kaybolmalı, oyun alanındaki “2” ile gösterilen konumda rastgele ikinci bir hayvan resmi belirmeli, bu da ilki gibi butonlarla kontrol edilebilmelidir. Benzer durum diğer hayvanlar için “3” ve “4” konumlarında gerçekleşmeli.
- 8.) Rastgele olarak gelen hayvan resimlerinde, bir gelen resim bir daha gelmemelidir.
- 9.) 4. hayvan da kapıya ulaştığında ekrana “Tebrikler, oyunu kazandınız!” şeklinde bir mesaj gelmelidir.

EK 5: UYGULAMA SONU ANKETİ

Lütfen aşağıdaki soruları kısaca ve uygun biçimde cevaplayınız.

Adınız Soyadınız:

Devreniz:

[] 1. Öğretim

[] 2. Öğretim

- 1) Hangi programlama dillerini biliyorsunuz?
- 2) Bu güne dek hangi yazılım geliştirme ortamlarını (IDE) kullandınız?
Örn: Visual Studio, Eclipse, Netbeans, Adobe Flash, Macromedia Authorware gibi
- 3) Bölümde görmüş olduğunuz Programlama Dilleri derslerinde hangi yazılım geliştirme ortamlarını kullanarak proje yaptınız?
- 4) Programlama Dilleri 2 dersini hangi notla geçtiniz?
Lütfen harf notu olarak belirtiniz. (AA, BA, BB, vb.. bilmiyorsanız boş bırakabilirsiniz)
- 5) Programlama Dilleri (hem 1, hem 2) dersleri boyunca hangi yazılım projelerini yaptınız?
Her iki ders için de ayrı ayrı projelerinizi çok kısaca açıklayınız.
- 6) Programlama Dilleri 2 dersini geçtiğiniz belli olduğu andan itibaren (örneğin, geçen sene temmuz ayından) bu güne dek, bildiğiniz programlama dilleri ile herhangi bir uygulama geliştirdiniz mi? Cevabınız evet ise, hangi dili kullanarak ne amaçlı uygulama geliştirdiğinizi kısaca tanımlayınız.
- 7) Kullanmayı bildiğiniz programlama dilleri / yazılım geliştirme ortamları düşünüldüğünde, kendinizi bu konularda bir okul bünyesinde ve daha önce programlama eğitimi almamış öğrencilere yönelik dersler verebilecek yetkinlikte hissediyor musunuz? Açıklayınız.
Cevabınız evet ise, hangi dil ile, ne ölçüde ders verebilirsiniz?
- 8) Ortaokul öğrencilerine görsel programlama eğitimi verilmesi fikri konusundaki düşünceleriniz nedir?
(görsel programlama eğitimi ile kast edilen, PD2 ders içeriğinde olduğu gibi GUI form uygulamaları geliştirilmesi eğitimidir)
- 9) Programlama eğitimi sizce hangi yaştan / seviyeden itibaren verilmeli?
- 10) Bildiğiniz programlama dillerinden farklı, yeni bir görsel programlama dilini kolaylıkla öğrenebileceğinize inanıyor musunuz?

EK 6: ÖRNEK APP INVENTOR PROJESİ: HESAP MAKİNESİ

Aşağıda, deneysel uygulama kapsamında öğrencilere verilen ders projelerine örnek olarak, Deney Grubu 1'e 2. Hafta boyunca gösterilmiş olan "Hesap Makinesi" uygulamasının ekran görüntüsü ve kod bloklarının görünümü paylaşılmıştır.

Kod bloklarının son hali:

```
when ToplaButonu.Click
do
  set SecilenIslemLabel.Text to text +
  set IslemSonucuTextBox.Text to BirinciSayiTextBox.Text + IkinciSayiTextBox.Text

when CikartButonu.Click
do
  set SecilenIslemLabel.Text to text -
  set IslemSonucuTextBox.Text to BirinciSayiTextBox.Text - IkinciSayiTextBox.Text

when CarpButonu.Click
do
  set SecilenIslemLabel.Text to text x
  set IslemSonucuTextBox.Text to BirinciSayiTextBox.Text * IkinciSayiTextBox.Text

when BolButonu.Click
do
  ifelse test IkinciSayiTextBox.Text = number 0
  then-do set SecilenIslemLabel.Text to text Sıfır ile bölmeye çalıştınız.
  else-do set SecilenIslemLabel.Text to text %
  set IslemSonucuTextBox.Text to call format as decimal number BirinciSayiTextBox.Text / IkinciSayiTextBox.Text places number 5

when TemizleButonu.Click
do
  set SecilenIslemLabel.Text to text
  set IslemSonucuTextBox.Text to text
  set BirinciSayiTextBox.Text to text
  set IkinciSayiTextBox.Text to text
```

EK 7: MBTGPD KULLANAN ÖĞRENCİ PROJELERİ

7.7.1. “Shoot em Up” Oyunu (örümcek vurma)



Proje tanımı: Ekranın üst kısmında sağa-sola hareket eden örümcekleri vurmaya ve onların karşılık olarak verdiği atışlardan kaçınma amacına sahip bir Shoot –em up oyunudur. Fikir <http://goo.gl/fM1XKH> kaynağından alınmıştır. Oyuncunun ekranın alt kısmında bulunan “gemisini” hareket ettirmek için sağ ve sol butonları var, bunun yanı sıra bir de ateş etme butonu var. Oyunun ilk seviyesinde ekranda düşman olarak sadece bir örümcek hareket ediyor, ancak örümceği vurmak suretiyle elde edilen puanlar belirli bir seviyeyi aşınca ikinci bir örümcek de beliriyor. Örümcekler oyuncuya ateş ederek karşılık veriyorlar, bu atışların oyuncuyu 3 defa vurması durumunda oyun sona eriyor.

Kullanılan bileşenler:

- Canvas
- ImageSprite
- Ball
- Horizontal Arrangement
- Vertical Arrangement
- Label
- Timer

7.7.2. NECS – İngilizce Kelime ve Telaffuz Öğrenme Programı

Proje tanımı: NECS programı İngilizce eğitiminde öğrencilere pratik yapma imkanı tanımak amacıyla oluşturulmuştur. Program, 4 modülden oluşmaktadır. İlk modülde, öğrencilere İngilizce kelimeler, ilişkin resimleriyle birlikte gösterilmekte ve telaffuzları da sesli olarak okunmaktadır. İkinci modülde, öğrenciye resim sunulurken hangi kelimeye tekabül ettiğini doğru tercih etmesi gereken çoktan seçmeli bir test uygulanmaktadır. Üçüncü modülde, resmi gösterilen veya telaffuzu okunan kelimeyi öğrenciden klavye kullanarak düzgün biçimde yazması istenmektedir. Son olarak dördüncü modülde ise, yazı ile verilen ve ilişkin resmi gösterilen kelimeyi öğrencinin doğru telaffuzla okuması istenmektedir.

Kullanılan bileşenler:

- Label
- Button
- Canvas
- Image
- Screen
- Arrangement
- Speech Recognizer
- Text to Speech
- Sound



7.7.3. Sinek Yakalama Oyunu

Proje tanımı: Orada burada rastgele uçan sinir bozucu sineklere gününü göstermenin zamanı geldi. Oyun ekrandaki sineği parmağınızla vurarak öldürmeyi amaçlamaktadır. Sineğin her yakalanmasında puan artacaktır ve ileriki seviyelere ulaşılacaktır. Oyuncunun kullanabileceği, biri ekrandaki tüm sinekleri anında yok eden, diğeri de sinekleri yavaşlatan iki adet “joker” hakkı bulunmaktadır.

Kullanılan Bileşenler

- Label
- Button
- Textbox
- Canvas
- ImageSprite
- Image
- Timer
- Screen
- Arrangement

