T.C.

DOKUZ EYLÜL ÜNİVERSİTESİ

GÜZEL SANATLAR ENSTİTÜSÜ

GRAFİK ANASANAT DALI

Yüksek Lisans Tezi

SHADER SİSTEMLERİNİN OYUN TASARIMINDA KULLANIMININ GRAFİK AÇIDAN İNCELENMESİ VE UYGULANMASI

Hazırlayan

Muharrem Deniz ÇİÇEK

Danışman

Doç. Ceren BULUT YUMRUKAYA

İZMİR / 2024

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum "SHADER SİSTEMLERİNİN OYUN TASARIMINDA KULLANIMININ GRAFİK AÇIDAN İNCELENMESİ VE UYGULANMASI" adlı çalışmanın, tarafımdan, bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurmaksızın yazıldığını ve yararlandığım eserlerin kaynakçada gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve bunu onurumla doğrularım.

Tarih

… / … / …

Deniz ÇİÇEK

TUTANAK

Dokuz Eylül Üniversitesi Güzel Sanatlar Enstitüsü' nün …… / …… / ..... tarih ve …… sayılı toplantısında oluşturulan jüri, Lisanüstü Öğretim Yönetmeliği'nin …….. maddesine göre …….……. öğrencisi ..... 'nın

……
……

konulu tezi incelenmiş ve aday …… /……. /……... tarihinde, saat.....'da jüri önünde tez savunmasına alınmıştır.

Adayın kişisel çalışmaya dayanan tezini savunmasından sonra ..... dakikalık süre içinde gerek tez konusu, gerekse tezin dayanağı olan anasanat dallarından jüri üyelerine sorulan sorulara verdiği cevaplar değerlendirilerek tezin ……. olduğuna oy ..... ile karar verilmiştir.

BAŞKAN

ÜYE ÜYE

ÖZET

Bu araştırma, gölgelendirici programlarının kullanımını odağına alan bir uygulama projesi aracılığıyla mekanik oyunlardan elektromekanik ve dijital oyunlara oyun tasarımının gelişimini inceler. Gölgelendiricilerin sağladıkları araçlar çizgi, iki ve üç boyutlu form, ışık-gölge ve doku gibi geleneksel tasarım unsurlarının bilgisayar diline (vektör, ağ, model vb.) çevrilmiş dijital versiyonları olarak değerlendirilebilirler. Uygulama projesi olarak geliştirilen Mini Island, 2010'lu yıllarda büyük oyun şirketlerinin sürekli olarak daha yüksek çözünürlüklü grafikler, aşırı detay, gerçeğe yakınlık ve karmaşık oynanış özelliği ve estetiğini destekleyen ve dayatan kalıplaşmış üretimlerine alternatif olarak ortaya çıkan bağımsız oyun türüne dâhildir. Bu alanda yok denecek kadar az basılı kaynağın bulunduğu ülkemizde, böylesi bir tez çalışması hem tez araştırmasının genel kapsamı hem de uygulama projesinin teknik boyutu bakımlarından İngilizce literatür taramasına dayalıdır.

ABSTRACT

This research examines the evolution of game design from mechanical games to electromechanical and digital games through an application project focusing on the use of shader programs. The tools provided by shaders can be considered as digital versions of traditional design elements such as line, two and three dimensional forms, light and shadow and texture, translated into computer language (vector, mesh, model, etc.). Developed as an application project, Mini Island belongs to the independent game genre that emerged in the 2010s as an alternative to the stereotypical productions of big game companies that constantly support and impose higher resolution graphics, extreme detail, realism and complex gameplay features and aesthetics. In our country, where there are almost no printed sources in this field, such a thesis study is based on English literature review in terms of both the general scope of the thesis research and the technical dimension of the application project.

ÖNSÖZ

Günümüzde grafik tasarımın sacayaklarını bilgisayar, iletişim ve sanat alanları oluşturmaktadır. İletişim ve tasarım alanları bilginin olduğu kadar duygunun da aktarımı ile ilgilidirler. Her tür kültürel içeriğin oluşturulduğu ve aktarıldığı dijital ortam ise beraberinde kendi düşünme ve hissetme biçimini getirmektedir. Dolayısıyla çağdaş grafik tasarımcısının, ilgi alanı ne olursa olsun, bu üç alanı sentezleyebilen bir bilgi ve pratiğe sahip olması gerekir. Bu anlamda hem böylesi bir eğitim sürecinde hem de bu tezi oluşturmamda değerli yorum ve katkılarıyla beni yönlendiren danışman hocam sayın Doç. Ceren Bulut Yumrukaya'ya ve ayrıca "Mini Island" başlıklı uygulama projemin programlarının yazılımında son derce önemli yardımlarını esirgemeyen Damla Alim'e teşekkürü bir borç bilirim.

Deniz ÇİÇEK

İÇİNDEKİLER

GÖRSEL LİSTESİ

GİRİŞ

İletişim, grafik tasarım ve bilgisayar bilimi alanları birbirlerinden farklı olmakla birçok ortak özellikleri vardır. Bunlardan bazıları bilgi iletimi, elektronik medya kullanımı ve bilginin temsiline ilişkindir. Bilgisayar bilimi bilginin temsili, manipülasyonu, iletimi ve tüketilmesinin teknik yönlerine odaklanırken, iletişim bilimi ve grafik tasarım aynı şeyin insani ve sanatsal yönlerine odaklanırlar. Bilgisayar bilimleri görüntülerin pikseller cinsinden temsili ve veri iletimiyle, iletişim bilimleri ve grafik tasarım ise sırasıyla içeriğin teorik ve görsel düzeyde temsili ve bilgi ve duygu iletimiyle ilgilenirler.

Dijital devrimle birlikte mevcut tüm medya, bilgisayar için erişilebilir sayısal verilere dönüşmüştür. Sayısal kodlama [numerical coding] grafikleri, görüntüleri, sesleri, şekilleri, mekanları ve metinleri hesaplanabilir veri kümelerine dönüştürürken medyayı da programlanabilir kılmıştır. Dijital koddan oluşan, sayısal olarak temsil edilen yeni medya (mültimedya) nesneleri biçimsel olarak, yani matematiksel bir işlev kullanılarak tanımlanabilir ve algoritmik manipülasyona tabi tutulabilirler. Bilgisayar ortamına aktarılmış entegre, etkileşimli, çok katmanlı, çeşitli, değişken, modüler ve hiyerarşiden yoksun medya artık bilgisayarın veri organizasyonunun yerleşik kurallarını takip eder. Yapısal olarak, yeni medya nesneleri diğer bilgisayar dosyalarıyla uyumludur ve onlara dönüştürülebilir. Bu anlamda yeni medya aracılığıyla bilgisayarlaşan, yani bilgisayarın işleme biçimine dönüştürülen kültürün kendisi, sadece uygulama ve sistem yazılımları ve bilgisayar programlama araçlarıyla değil, aynı zamanda sosyal ağ hizmetleri ve sosyal medya teknolojileriyle bir arayüz haline gelir. Yüz milyonlarca insan tarafından doğrudan kullanılan "kültürel yazılım" çok daha büyük bir yazılım evreninin yalnızca görünen kısmıdır. Bilgisayarlar ve yazılımlar basitçe bir "teknoloji" değil, içinde farklı düşünebilen ve hayal kurulabilen yeni ortamlardır. Dolayısıyla çağdaş *kontrol*, *iletişim*, *temsil*, simülasyon, *analiz*, *karar verme*, *hafıza*, *görme*, *yazma* ve *etkileşim* teknikleri yazılımın rolünü ve etkilerini hesaba katmadan anlaşılamaz.

Bu anlamda günümüzde özellikle dijital oyun tasarımı ile ilgilenen grafik tasarımcılar hem bilgisayar hem iletişim hem de güzel sanatlar alanlarının kesişme

noktasında var olmak ve işlev görmek durumundadırlar. Bu durum mesleki açıdan avantajlar sunmakla birlikte bazı güçlükleri de beraberinde getirir. Bir grafik tasarımcı sıradan bir teknoloji kullanıcısı ve tüketicisi olmanın ötesinde, içinde faaliyet gösterdiği bilginin kodlanmasına dayalı sistemin ve teknolojinin yapısı, işleyişi ve özelliklerine ilişkin teorik ve pratik bir kavrayışa da sahip olmalıdır. Elinizdeki bu araştırma, dijital oyun yapımında kullanılan ve geleneksel grafik tasarım kavramlarına ve uygulamalarına dayalı ancak onların sayısallaştırılmış bir biçimi olduğu söylenebilecek gölgelendirici (shader) programlarını bir uygulama projesi örneğinde inceler.

"Kültürel, Teknolojik ve Tarihsel Bağlamda Dijital Oyunların Gelişimi" başlıklı birinci bölüm, daha geniş bir enformasyon sistemi ve teknolojileri bağlamında bilgisayarın sayısal kodlamaya dayalı ortamına aktarılmış bir kültürün önemli bir parçası olan dijital oyunların gelişimini; "Oyun Motorları ve Gölgelendiriciler" başlıklı ikinci bölüm dünyada en yaygın kullanıma sahip Unity oyun motorunun görüntü işleme iş akışında (render pipeline) çalışan gölgelendirici programlarını; üçüncü bölüm ise "Mini Island" başlıklı proje özelinde, 21. Yüzyılın ikinci on yılında büyük oyun firmalarının oyun anlayışına ve estetiğine alternatif bir mecra olarak ortaya çıkan ve gelişen bağımsız oyun türünü ve onun alt kategorisi olan inşa oyunlarını ele alır.

1. BÖLÜM

KÜLTÜREL, TEKNOLOJİK VE TARİHSEL BAĞLAMDA DİJİTAL OYUNLARIN GELİŞİMİ

1. BÖLÜM

KÜLTÜREL, TEKNOLOJİK VE TARİHSEL BAĞLAMDA DİJİTAL OYUNLARIN GELİŞİMİ

1.1. Enformasyon Sistemi

Enformasyon Sistemi (ES) 1950'lerde bilgisayarların veri işlemek için yaygın olarak kullanılmaya başlanması ile ortaya çıkan görece yeni bir disiplindir (Elliot, 2006). Bilgisayarlar olmadan da enformasyon sisteminden bahsedilebilir ancak uygulamada, bilgi sistemleri artık neredeyse her zaman bilgisayarlıdır ve teknoloji kullanımlarında çok gelişmiş olabilirler. Bu sistemde bilgisayarlar kadar insanların da sürece dahil olduğu, dolayısıyla tüm parçalarının otomatik olmadığı söylenebilir. Bilgi işleme döngüsü olarak tanımlanan ES, verileri (olguları) işleyerek bilgi üreten bilgisayarlı bir sistemdir ve bu bilgi işleme ve üretme süreci *girdi*, *işlem*, *çıktı* ve *depolama* olarak dört adımdan oluşur (Siegel, 2005). Çevreden alınan ve bilgisayara iletilen ham verilere girdi denir. Bilgisayar giriş aygıtından verileri aldıktan sonra, kullanıcılar için yararlı bilgiler üretmek üzere verileri manipüle edecek, rafine edecek ve işleyecektir. Bu adım işleme olarak adlandırılır. Veriler rafine edildikten ve yararlı bilgilere dönüştürüldükten sonra, çıktı olarak son kullanıcılara gösterilir. Son olarak, bilginin gelecekte kullanılmak üzere saklanması gerekir. Dört süreç de *bilgi işleme döngüsünü* oluşturur. Girdi ham olgulardan (facts) oluşurken, bilgi (information) düzenlenmiş (organized) veya işlenmiş (processed) olguların bir toplamıdır.

Eksiksiz bir enformasyon sisteminin içerdiği unsurlar şunlardır: donanım, yazılım, veri, eğitimli personel ve prosedürler. Bilgisayar donanımı beş kategoride sınıflandırılabilir: kişisel (mikro) bilgisayarlar, sunucular, mini bilgisayarlar, ana bilgisayarlar ve süper bilgisayarlar. Taşınabilir ya da taşınabilir olmayan kişisel bilgisayarlar mikroişlemci içerirler ve bireysel veya kişisel kullanım için tasarlanmışlardır. En popüler mikrobilgisayar türü masaüstü bilgisayardır. İş istasyonları

(Workstations), kişisel bilgisayarlar arasında en güçlü bilgisayarlardır ve çoğu zaman bir ağ ortamında dosya sunucusu olarak kullanılırlar. Birçok mühendis ürün tasarımı ve testlerine yardımcı olmak için iş istasyonları kullanır. İş istasyonları, hesaplamalarda ve grafiklerde yüksek performans gösterir. Taşınabilir bilgisayarlar arasında dizüstü bilgisayar, not defteri (notebook), alt not defteri (subnotebook) ve kalem tabanlı bilgisayarlar bulunmaktadır. Dizüstü bilgisayarlar sabit bir sürücüye, CD-ROM sürücüsüne ve diğer ekipmanlara sahiptirler. Not Defteri (Notebook) bilgisayarlar, dizüstü bilgisayarların daha küçük versiyonlarıdır. Alt not defteri bilgisayarlar, not defteri bilgisayardan bile daha küçüktür. Kalem tabanlı bilgisayarlar en küçük bilgisayarlardır ve bilgisayara elle bilgi girmek veri girmek için tasarlanmış özel bir yazılımdır.

Sunucu (server) bilgisayarlar, kullanıcıların dosyaları, uygulamaları ve donanım kaynaklarını paylaşmasına olanak tanıyan bir bilgisayar ağını desteklemek üzere tasarlanmıştır. Bir sunucu bilgisayar normalde ağdaki diğer bilgisayarlara dosya depolama ve kaynak yönetimi, veri iletişimi, yazdırma yönetimi ve diğer bilgisayar işlevleri açısından hizmet etmek için kullanılır. Minibilgisayarlar çoklu kullanıcı ortamları açısından mikrobilgisayarlardan daha güçlüdür. Bilimsel hesaplamalar, mühendislik tasarımı, uzay araştırmaları ve karmaşık işlem gerektiren vb. görevler için kullanılan en güçlü mini bilgisayarlar ise süper mini bilgisayarlar olarak adlandırılır.

Bir yazılım programı aslında programcılar tarafından çeşitli bilgisayar dillerinde yazılmış bir dizi talimattır. Yazılım, bilgisayarın izleyeceği işlem dizilerini içerir. Bir programın çalışabilmesi veya yürütülebilmesi için önce programın bilgisayarın ana belleğine yüklenmesi gerekir. Bundan sonra, programlar nasıl tasarlandıklarına bağlı olarak belirli işlevleri yerine getirmek için çalıştırılabilir. Örneğin, kelime işlem (Word processing) programı kullanıcıların yazdıklarını girmelerine ve içerikleri düzenlemelerine olanak tanır. Bir grafik tasarım programı grafik tasarımları gerçekleştirmek için kullanılır. Bilgisayar programlarının çoğu, Cobol, Basic ya da C++ gibi programlama dillerinde gerekli talimatları yazan ve bilgisayar programcısı olarak adlandırılan özel eğitimli kişiler tarafından yazılır. İki tür yazılım programından bahsedilebilir: sistem yazılımı ve uygulama yazılımı. Sistem yazılımı, bilgisayar donanımını kontrol etmek ve çalıştırmak için kullanılan programlardan oluşur. Sistem

yazılımında üç bileşen vardır: işletim sistemi, yardımcı programlar ve dil işlemcileri. İşletim sistemi bilgisayara programların nasıl yükleneceği, saklanacağı ve yürütüleceği, giriş/çıkış aygıtları arasında verilerin nasıl aktarılacağı ve mevcut kaynakların (CPU zamanı) nasıl yönetileceği gibi işlevlerin nasıl yerine getirileceğini söyler. Bilgisayarın çalışabilmesi için işletim sisteminin ana belleğe yüklenmesi gerekir. Yardımcı programlar, disket biçimlendirme ve dizin oluşturma gibi uygulama yazılımlarında bulunmayan işlevleri yerine getirmek üzere tasarlanmıştır. Uygulama yazılımı, belirli bir kullanıcının görevini yerine getirmek için oluşturulan programlardan oluşur daha sonra işletim sisteminin yardımıyla bilgisayara yüklenebilir. Uygulama yazılımı bir kullanıcının bir belge hazırlamasına, bir çalışma sayfası tasarlamasına veya faydalı bir veritabanı oluşturmasına olanak tanır.

"Veri" terimi genellikle bir yönetim bilgi sisteminin girdisini ifade eder. Veriler ES tarafından işlendikçe bilgi üretilir. Bu bilgiler daha sonra karar vermek için kullanılabilir. Veriler normalde dosyalara veya tablolara girilir ve bunlar daha sonra bir veri tabanında düzenlenir. Kullanıcılar uygulama yazılımı aracılığıyla girdi verilerini alabilir ve çıktı olarak bilgi üretebilir.

ES uzmanları ve programcılar sistemin tasarlanması ve programlanmasından sorumludur, bilgisayar operatörleri ise sistemi bilgi üretmek için kullanır. Yeterli eğitimle, operatörler ES uzmanları tarafından tasarlanan istenen işlevleri yerine getirebilirler. Deneyimli bir kullanıcı da ES uzmanlarına değerli önerilerde bulunabilir veya ES'nin geliştirilmesine dahil olabilir.

1.1.1. Mültimedya, İletişim ve Enformasyon Teknolojisi

Yaklaşık 1995 yılından bu yana bilgisayarlar dijital video, ses, animasyon ve metni tek bir donanım ve yazılım paketine entegre etme kabiliyetine sahiptir (Burnett vd., 2003). Depolama ve hızdaki artışlar ile boyut ve maliyetteki düşüşler sayesinde multimedya günümüzde ekonomik hale gelmiştir. İletişim yeteneklerine ve İnternet gibi ağlar üzerinden bilgi paylaşımına giderek daha fazla önem verilmektedir.

Bilgisayar herhangi bir ortam olabilen bir araçtır, dolayısıyla bugün multimedya olarak adlandırılan şeyin temelidir. Hem multimedya deneyimimizin üretilmesine yardımcı olur hem de ona erişmemizi sağlar. Dijk (2005) multimedyanın üç temel özelliğini katmanlaşma, modülerlik ve bilginin manipüle edilebilirliği olarak sınıflandırır:

> Bunlardan ilki bilginin katmanlaştırılmasıdır. Kullanıcılar, açıklamalar, şekiller, illüstrasyonlar, fotoğraflar, videolar, animasyonlar, sesler ve benzeri şekillerde alınan bir olgu hakkında daha fazla bilgi bulabilirler. Böylece, aynı bilgi çeşitli şekillerde tasvir edilebilir. İkinci özellik modülerliktir: bir bilgi veri tabanı, ayrı ayrı alınabilecek ve kullanıcının istediği şekilde birleştirilebileceği parçalardan oluşur. Son özellik ise kullanıcının "kesip dijital bilgi parçalarını yapıştırmasını sağlayan multimedyadaki bilginin manipüle edilebilirliğidir. (s. 56)

Multimedya aynı zamanda eş zamanlı çalışan beş süreçle tanımlanabilir: entegrasyon, etkileşimlilik (interactivity), hipermedya, tutulma (immersion) ve anlatısallık (Burnet vd., 2003). Entegrasyon, sanatsal formların ve teknolojinin melez bir ifade biçiminde birleştirilmesidir. Etkileşimlilik (kullanıcının medya deneyimini doğrudan manipüle etme ve etkileme yeteneği), kullanıcının bilgi sunumu üzerinde sahip olduğu kontrol miktarıdır. Etkileşimin en yaygın üç sınıflandırması şunlardır: doğrusal, dallanan ve hipermedya.

Etkileşimli doğrusal bir sunum, bilginin sunulma sırasına ve şekline yazarın karar verdiği bir sunumdur. Kullanıcı sadece hızı kontrol eder. Etkileşimli bir dallanma programı, kullanıcının bir ana menü gibi bir grup seçenek arasından seçim yaparak sunum sırası üzerinde bir miktar kontrole sahip olduğu bir programdır. Yazar, programın herhangi bir noktasında mevcut olan seçeneklere nelerin dahil edileceğine karar verme kontrolünü hala elinde tutmaktadır. Etkileşimli hipermedya, kullanıcının sunumun hızı, sırası ve içeriği üzerinde neredeyse tamamen kontrol sahibi olduğu, birbiriyle ilişkili bir bilgi ağı olarak düşünülebilir. Bağlantılar bilgiye rastgele erişim sağlar. 'Hipermedya' ile yazarlar ve sanatçılar, geleneksel hiyerarşilere alternatifler sunan bir dizi kışkırtıcı yan yana gelişte kullanıcıyı bir fikirden diğerine sıçramaya teşvik eden eserler yaratabilirler.

‘Tutulma’ üç boyutlu bir ortamın simülasyonuna veya önerisine girme deneyimidir ve multimedya bilgilerinin oluşturulmasını ve kullanımını destekleyen zengin multimodal (görsel, işitsel, dokunsal) arayüz tekniklerinin kullanılmasıyla elde edilir. Son olarak anlatısallık, yukarıdaki kavramlardan türeyen ve doğrusal olmayan hikâye formları ve medya sunumuyla sonuçlanan estetik ve biçimsel stratejileri kapsar.

Multimedyanın temelinin iki farklı bilime dayandığı iddia edilebilir: iletişim bilimi ve bilgisayar bilimi. İletişim biliminin tanımları ve kökleri çoğunlukla 'yumuşak bilimler' olarak adlandırılan sosyal bilimler ve beşerî bilimlerden türetilirken, bilgisayar biliminin tarihi çoğunlukla 'sert bilimlerde bulunur. Bazen bu ayrım Sanat ve Bilim arasındaki fark olarak tanımlanır (s. 3).

İletişim ve bilgisayar bilimleri birbirinden farklı olmakla birlikte birçok benzerliği de paylaşmaktadır. Bunlardan bazıları bilgi iletimi, elektronik medya kullanımı ve bilgi temsilidir (örneğin, bilgi teorisi; bilgiyi kayıpsız temsil etmek için gereken minimum kod). Bilgisayar bilimi, bilginin temsili, manipülasyonu, iletimi ve alımının teknik yönlerine odaklanırken, iletişim bilimi aynı şeyin insani yönlerine odaklanır. Bilgisayar bilimleri görüntülerin pikseller cinsinden temsili ile ilgilenirken, iletişim bilimleri içeriğin teorik düzeyde, daha çok sunum açısından temsili ile ilgilenir. Genel olarak, bilgisayar bilimleri veri iletimiyle ilgilenirken, iletişim bilimleri bilgi/duygu/bilgi iletimiyle ilgilenir. Bilgisayardan bilgisayara ‘teknik’ iletişim (communications) ile insandan insana, insandan bilgisayara ve bilgisayardan insana ‘insani’ iletişim (communication) arasındaki önemli bir ayrım söz konusudur (s. 3-4).

Dijital devrimle birlikte mevcut tüm medya, bilgisayar için erişilebilir sayısal verilere dönüştürülmüştür. Bunun sonucunda grafikler, hareketli görüntüler, sesler, şekiller, mekanlar ve metinler hesaplanabilir, yani basitçe bilgisayar verisi kümeleri haline gelir. Medya yeni medya olur (Manovich, 2001, aktaran Burnet, 2003, s. 10-11). Medya ve bilgisayarın buluşması ve kültürün bir bütün olarak bilgisayarlaşması hem medyanın hem de bilgisayarın kimliğini değiştirir. Medyanın kimliği bilgisayarınkinden bile daha dramatik bir şekilde değişmiştir. Yeni medya'yı önceki sanat formlarından kavramsal olarak ayıran beş özellik şunlardır:

> … medyanın programlana bilirliğini kolaylaştıran sayısal kodlama [numerical coding]; parçalarında yapısal bir ayrıklık yaratan modülerlik; üretim ve erişimin otomasyonu; değişkenlik [variability], yani medyanın 'tamamlanmasından' sonra da değişken format ve versiyonlarda sunulmaya devam edebilmesi; ve son olarak, kodlarının farklı sistemler arasında işlediği ve dolayısıyla farklı sistemler arasında aktarılabilir olduğu ölçüde kod dönüştürme [transcoding] (s. 10).

Dijital koddan oluşan, sayısal olarak temsil edilen yeni medya nesneleri biçimsel olarak, yani matematiksel bir işlev kullanılarak tanımlanabilir ve algoritmik manipülasyona tabi tutulabilir. Medya böylece programlanabilir hale gelir. *Modüler* yeni medya nesneleri daha büyük nesneler halinde birleştirildiklerinde bile bağımsızlıklarını korudukları için temelde hiyerarşik olmayan bir organizasyona vurgu yapar. Örneğin bir Word belgesi ve WWW her zaman kendi başlarına erişilebilen ayrı nesnelerden oluşur. Bu iki özellik birlikte, medya yaratımı, manipülasyonu ve erişimiyle ilgili birçok işlemin otomasyonuna izin verir. Böylece, 'insanın niyetliliği yaratıcı süreçten en azından kısmen çıkarılabilir. *Otomasyon* örnekleri görüntü düzenleme, sohbet robotları, bilgisayar oyunları, arama motorları, yazılım ajanları vb. alanlarda bulunabilir. Yeni medya nesneleri potansiyel olarak sonsuz versiyonlarda var olabilen şeylerdir. Örneğin, öğelerinin sırası bir kez ve herkes için belirlenen film, öğelerinin sırası esasen *değişken* olan yeni medya ile taban tabana zıttır. Değişkenliğe örnek olarak özelleştirme (customization) ve ölçeklenebilirlik (scalability) verilebilir. *Kod dönüştürme* (transcoding) temel olarak bir şeyi başka bir formata çevirmek anlamına gelir. Ancak en önemli husus, bilgisayar ortamına aktarılmış medya artık bilgisayarın veri organizasyonunun yerleşik kurallarını takip eder. Yapısal olarak, yeni medya nesneleri diğer bilgisayar dosyalarıyla uyumludur ve onlara dönüştürülebilir. Bu anlamda yeni medya aracılığıyla bilgisayarlaşan, yani bilgisayarın işleme biçimine (sayısal mantık) dönüştürülen kültürün kendisi bir arayüz haline gelir.

1.1.2. Yazılım Kültürü ve Oyun

Manovich'e (2013) göre genişletilmiş anlamıyla *yazılım* (softwear) sadece uygulama yazılımları, sistem yazılımları ve bilgisayar programlama araçlarını değil, aynı zamanda sosyal ağ hizmetleri ve sosyal medya teknolojilerini de içerir (s. 6).

> “Arama motorları, tavsiye sistemleri, haritalama uygulamaları, blog araçları, açık artırma araçları, anlık mesajlaşma müşterileri ve elbette insanların yeni yazılımlar yazmasına olanak tanıyan OS, Android, Facebook, Windows, Linux gibi platformlar küresel ekonominin, kültürün, sosyal yaşamın ve giderek siyasetin merkezinde yer almaktadır. Ve bu yüz milyonlarca insan tarafından doğrudan kullanılması ve kültürün "atomlarını" taşıması anlamında "kültürel yazılım" çok daha büyük bir yazılım evreninin yalnızca görünen kısmıdır.” (Manovich, 2001, s. 7)

Bilgisayarlar ve yazılımlar sadece "teknoloji" değil, daha ziyade farklı düşünebileceğimiz ve hayal edebileceğimiz yeni ortamlardır. Çağdaş *kontrol*, *iletişim*, *temsil*, simülasyon, *analiz*, *karar verme*, *hafıza*, *görme*, *yazma* ve *etkileşim* teknikleri yazılımın rolünü ve etkilerini hesaba katmadan anlaşılamaz (s. 15). Öyle ki, Alman medya ve edebiyat kuramcısı Friedrich Kittler, öğrencilerin günümüz kültürü hakkında bir şeyler söyleyebilmek için en az iki yazılım dili bilmesi gerektiğini söyler (s. 20).

1980 ile 1990 arası bir on yıl gibi kısa bir zaman diliminde, bilgisayar görünmez bir teknoloji olmaktan çıkıp kültürün yeni motoru haline gelmiştir. Bu on yıl içinde aynı zamanda bilgisayarların piyasaya sürüldüğü, masaüstü yayıncılığın ortaya çıktığı ve yaygınlaştığı ve bazı edebiyat akademisyenlerinin hipermetin kullandığı bir dönemdir. Apple 1984 yılında Macintosh için tasarladığı Grafik Kullanıcı Arayüzü’nü, 1987'de Adobe Systems Illustrator'ı ve 1989’da Photoshop’ı piyasaya sürer. Aynı yıl James Cameron tarafından yönetilen, ilk karmaşık sanal karakteri yaratmak için öncü CGI’ın kullanıldığı The Abyss filmi gösterime girer. 1990 yılına gelindiğinde Tim Berners-Lee, bugün var olan World Wide Web'in tüm bileşenlerini (web sunucusu, web sayfaları ve web tarayıcısı) yaratmıştır. En önemlisi, teknik olmayan kullanıcılara yönelik Grafik Kullanıcı Arayüzü (GUI), kelime işlemci, çizim, boyama, 3D modelleme, animasyon, müzik besteleme ve düzenleme, bilgi yönetimi, hipermedya ve multimedya yazma (HyperCard, Director) ve küresel ağ (World Wide Web) uygulamaları içeren yazılımların piyasaya sürülmesidir. Kullanımı kolay yazılımların devreye girmesiyle 1990'lar, grafik tasarım, mimari, ürün tasarımı, mekân tasarımı, film yapımı, animasyon, medya tasarımı, müzik, yüksek öğrenim ve kültür yönetimi gibi alanlarda yazılım araçlarının kademeli

olarak kullanılmaya başlandığı bir on yıl olacaktır. Bu tez çalışmasının merkezinde olan Shaderlar, Manovich'in deyimiyle, 'medya yazılımları' kategorisine dahil Photoshop, Illustrator, InDesign, Final Cut, After Effects, Maya, Blender, Dreamweaver ve Apertur gibi medya içeriği oluşturmak, düzenlemek ve organize etmek için kullanılan uygulamaların temel bir bileşenidir.

> Sicart (2023) enformasyon çağının kültürünü anlamak için, neden yazılımla oynadığımızı anlamamız gerektiğini söyler.
>
> Yazılımla oynarız çünkü yazılım da oyun gibi dünyalar yaratır. Yazılımla oynarız çünkü oynamak diğer faillik biçimleriyle ilişki kurmanın bir yoludur ve yazılım insani ve yapay faillik biçimleri yaratır. Yazılımla oynarız çünkü eğlencelidir - verimli bir şekilde işlevsel olması gereken bir şeyi sadece işe yaramaz şekillerde kullanmak ve geçici olarak başka bir dünyada başkalarıyla tanışmak ve başkaları olmak eğlencelidir. (s. 10)

Dijital oyun analog oyundan farklıdır çünkü dijital oyun, yazılım tarafından ve yazılım için yaratılan dünyalarda yazılım failleriyle ilişkisel bir bağlanma biçimidir. Bu anlamda dijital oyun, enformasyon çağının bir olgusudur. İnfosferdeki birçok insan-yazılım ilişkisi oyunsaldır. Güç elde etme ve aidiyet mantığıyla sürükleyici sentetik dünyaların zevklerini vaat eden video oyunları; Minecraft'tan Snapchat'e, Facebook'a ve Instagram'a kadar, başkalarıyla ilişki kurmanın yeni dolayımlı biçimleriyle yaşanacak yeni yerleri kolaylaştıran dijital oyun alanları; saygıdeğer Tamagotchi'den ya da her zaman hoşa giden Alexa'dan, bankacılık yaparken, randevulaşırken ya da hayatlarımızı çevrimiçi yayınlarken eğlenmemizi isteyen uygulamaların daha akıcı, oyuncak benzeri deneyimlerine kadar dijital oyuncaklar söz konusudur. İnfosfer eğlenceli bir yer olabilir ve oyun yazılımı yeni kültür, sanat ve sosyal ilişki biçimleri yaratır. Sicart "dünya" kavramını, belirli eylemlilik biçimlerinin varlık deneyimini şekillendirdiği bir ortam olarak tanımlar.

> "Dijital oyun, öznellikleri şekillendirmenin, sınırları çizmenin ve yazılım unsurlarıyla ilişki kurarken kültürel formları ifade etmenin ve yaratmanın bir yoludur. Oyun ve yazılım, dünyalar yaratmak için kurallar ve süreçsel mantık

> kullanır. Bir oyunun kuralları, yürürlüğe girdiğinde, o oyunun dünyasını yaratır. Belirli bir yazılım parçasının prosedürleri, o yazılımın etmen olduğu dünyayı yaratır.
>
> …. Eylemlilik ortamları olarak dünyaların bu çokluğu yeni değildir: iş dünyası sosyal dünyadan, o da aile dünyasından farklıdır. Bunlar birbiriyle örtüşebilir, ancak içinde farklı failler olduğumuz farklı dünyalardır. Bu dünyalar eylem yoluyla, bir şeyler yaparak ve aynı zamanda failliği anlamlı bir şekilde sınırlandıran sınırlar ve prosedürler belirleyerek yaratılır. Bir iş sözleşmesi iş yerindeki işlevimizi tanımlar. Birini sevmek de eylemlerimizi şekillendirir. Hepimiz birçok dünyada yaşıyoruz ve bu dünyaların bazıları yazılım için ve yazılım tarafından yaratılıyor. Ve bu yazılım dünyalarının bazılarında oyun oynuyoruz." (s. 12)

Hukuk, şiir, savaş, sanat ve dil gibi olguların kaynağında oyun olduğu doğruysa, enformasyon çağı kültürünün birincil kaynağı oyun yazılımıdır. Ancak Sicart oyun yazılımının, video oyunlarının ötesinde dünyada var olmanın ilişkisel bir biçimi, yazılım ve insan eylemlikleri arasında bir ilişki kurma, şekillendirme, yeniden şekillendirme ve bunlara boyun eğmenin bir yolu olduğunu söyler (s. 14). Oyun yazılımları yapay ve insan eylemliliğinin ilişki kurabileceği yeni dünyalar yaratırlar. Bu dünyalardan bazıları bizi bilgisayarların süreçlerine tutsak ederken, diğerleri verili olan değil mümkün olan bir yaşama açılacaktır.

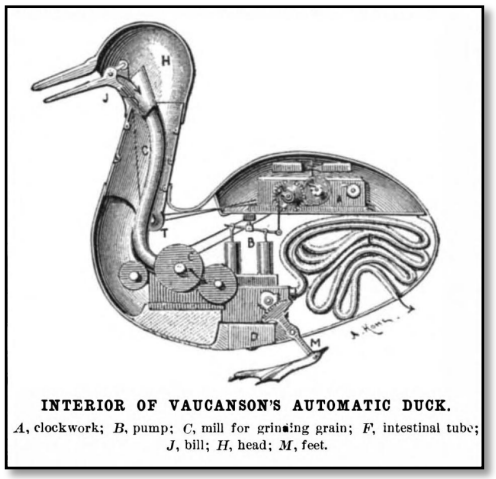
1.2. Atari Salonlarından Konsollara Dijital Oyunların Tarihi

Dijital Oyunların Tarihi: Sanat, Tasarım ve Etkileşimdeki Gelişmeler başlıklı kitabında Andrew Williams (2017), dijital oyunların tarihini "üç ana tarihsel oyun bağlamının/platformunun -oyun salonları, ev bilgisayarları ve konsolların- bir bileşimi olarak" değerlendirir. Dijital oyunların gelişimi, bilgisayar donanımı ve yazılımındaki gelişmelerle yakından bağlantılı olsa da oyun tasarımına ilişkin fikirler teknolojik sınırlamaların ötesinde süreklilik gösterir. Dijital atari oyunlarının ilk ortaya çıkışından neredeyse yüz yıl önce hem makine mühendisleri hem de sanatçılar görsel olarak çekici, anlaşılması kolay ve ustalaşması zor veya tamamen imkânsız olacak şekilde tasarlanmış

oyunlar yaratırlar. Bu tasarımlar çok sayıda müşterinin oyun başına düşük fiyatlar ödemesini sağlayarak sürümden kazanmaya odaklanan bir iş modeline dayanmaktadır. Oyunun hemen başlayıp hızlı bir şekilde sonlanması durumunda finansal açıdan başarılı olabilecek bu modelde oyunlar, karnaval ve panayırlar gibi kamusal alanlarda oyun oynama geleneğiyle ve yenilgiden sonra kazanma veya kendini kurtarma yönündeki insani arzuyla yakından bağlantılıdır.

1.2.1. Jetonlu Atari (Arcade) Makinelerinin Gelişimi

Madeni parayla çalışan makinelerin başlangıcı, İskenderiyeli mucit Heron'un, arınma ritüelleri için su dağıtan bir mekanizmayı tetiklemek için düşen madeni paranın kuvvetini kullanan bir cihaz tasarladığı MS birinci yüzyıla kadar uzanır (s. 2). Yirminci yüzyılın son çeyreğinin klasik jetonla çalışan video oyun salonları, 1800'lerin sonu ve 1900'lerin başındaki İkinci Sanayi Devrimi'nin getirdiği teknolojik ve ekonomik değişikliklerle doğrudan ilişkilidir. Bu dönemde jetonla çalışan eğlence makinelerinin üç ana türü ortaya çıkar: ilki, izleyicilere kinetik zevk getiren etkileşimli olmayan çalışan modeller; ikincisi, kamusal alanlarda kullanılmak üzere tasarlanmış test cihazlarının para kazandıran versiyonları; üçüncüsü ise bireylerin bir dizi iki boyutlu (2D) hareketsiz görüntüye, üç boyutlu (3D) stereoskopik görüntülere veya eski hareketli görüntülere bakmalarına olanak tanıyan görüntüleyicilerdir.



Şekil 1. Canard Digérateur (Sindiren Ördek). (https://archive.org/details/scientific-american-1899-01-21/page/n9/mode/2up

Orta çağdan itibaren Avrupalı mühendisler, şarkı söyleyen kuşlardan dinsel toplantılardaki grotesk gösterilerde kullanılan otomatik şeytanlara kadar geniş bir yelpazede karşımıza çıkan ve mekanik olarak hareket eden eğlence araçları geliştirmişlerdir. Birkaç yüzyıl içerisinde bu tip araçlar, hareket ve davranışları gerçeğine son derece benzer şekilde yeniden yaratacak kadar gelişkin hale gelir. Özellikle İsviçreli, Alman ve Fransız saat yapımcıları salt okunur belleğin öncülü ve daha sonraki mekanik eğlence cihazlarının önemli bir bileşeni olacak olan, "cam" adı verilen, düzensiz şekilli disklerde saklanan mekanik programlar oluştururlar örneğin, 1739'da Jacques de Vaucanson, kanatlarını çırpan, yiyen, içen ve hatta dışkılamayı simüle eden Canard Digérateur (Sindiren Ördek) adlı otomatı yaratmıştır. 1785'te ise Peter Kinzing ve David Roentgen, bir çekiçle tellere vurarak minyatür bir santur çalan ve aynı zamanda başı ve gözleriyle ince hareketler yapan otomatlar tasarlamışlardır. Bu otomatlar arasında en etkileyici olanı Henri Maillardet'in 1805 yılı civarlarında yarattığı, dört ayrıntılı sahne çizen ve ikisi Fransızca biri İngilizce üç şiir yazan Ressam Yazar'ıdır (Draughtsman Writer). Başlangıçta sadece zengin Avrupalı soyluların eğlence aracı olan bu otomatik cihazların geniş halk kitlelerinin kullanımına sunulması önce İngiltere'de ortaya çıkan sonra Avrupa'nın diğer ülkeleri ve Amerika Birleşik Devletleri'ne yayılan jetonla çalışan modeller vasıtasıyla olur. Bu gelişmelerle eş zamanlı olarak, yine madeni para ile çalışan, çoğu kaldırma, itme, çekme, kavrama ve yumruklama gibi eylemlerin sonuçlarını ölçerek rekabeti teşvik eden veya sosyal etkileşimi ve izleyiciliği kolaylaştıran deneme cihazları kulüp ve salonlarda yaygınlaşır.

Şekil 2. The Locomotive, 1885 by William T. Smith. *History of Digital Games Developments in Art, Design and Interaction* (s. 3).

Şekil 3. Electricity is Life, 1904 Mills Novelty. *History of Digital Games Developments in Art, Design and Interaction* (s. 4).

Şekil 4. Perfect Muscle Developer. *History of Digital Games Developments in Art, Design and Interaction* (s. 4).

Edison’un başlangıçta iş toplantılarını kaydetmek amacıyla geliştirdiği fonograf, bir yatırımcısının, kullanıcının bir kuruş karşılığında müzik kayıtlarını, tanınmış kişilerin konuşmalarını veya ses efektleriyle dolu dramatik anlatıları dinleyebildiği borular ve bozuk para yuvası eklemesiyle on dokuzuncu yüzyılın sonlarında tren istasyonlarında, otel lobilerinde ve tatil yerlerinde sayıları giderek artan mekanik ve elektromekanik yeniliklerle uyumlu hale gelir (s. 6). Yine Edison'un asistanı William Dickson’ın geliştirdiği, 1880'lerin sonlarında prototipi hazırlanan ve 1894'te üretime sunulan Kinetoskop, bir şeritten hareketli görüntüleri oynatma yeteneğine sahiptir. Bir gözetleme deliği ile pille çalışan bir ampul arasında uzanan 35 mm'lik bir film vasıtasıyla boks maçları, sirk sanatçıları veya tarihsel olayların yeniden canlandırmaları 30-40 saniyelik sürelerle izlenebilmektedir. Kısa süre içinde Kinetoskop salonları adı verilen özel eğlence alanları ilk olarak New York'ta ortaya çıkar ve Chicago, San Francisco, Atlantic City, Londra ve Paris'e yayılır. Başlangıçta kullanıcının bilet alarak kullandığı bu cihazlara bozuk para yuvası eklenmesiyle iş gücünden tasarruf edilir. Böylece kinetoskop salonları, jetonla çalışan eğlence cihazlarının bulunduğu bir alan olarak "atari salonu" konseptinin yaratılmasına yardımcı olacaktır. On dokuzuncu yüzyılın sonlarında Amerika ve İngiliz oyunlarının pek çoğu kuruşla çalıştığı için ucuz eğlence oyunlarının ve araçlarının toplandığı yerler kuruş atari (penny arcade) olarak adlandırılmaya başlanır. 1900'lerin başında kineteskopun yerini yine Edison'un eski iş arkadaşı William Dickson tarafından geliştirilen, jetonla çalışan yeni bir film görüntüleyici olan Mutoskop alır. 1907 yılına gelindiğinde ise, Amerika Birleşik Devletleri'ndeki paralı oyun salonları odak noktalarını daraltırlar ve filmlere daha az yer vermeye başlarlar.

Görüntüleyicilere ve test cihazlarına ek olarak, jetonlu makine üreticileri organize sporlara dayalı oyunlar da üretirler. İlk spor tabanlı jetonlu makineler, hedef vurma ve güç testi oyunlarının unsurlarını, yirminci yüzyılın sonları ve yirmi birinci yüzyılın başlarındaki elektromekanik ve dijital silah oyunlarının bir öncülü olan, ayrıntılı ortamlar ve çalışma modellerinde görülen hareketli karakterlerle birleştirirler. Bu, oyunlarla etkileşim hakkında düşünmeye yönelik farklı bir yaklaşıma işaret eder, çünkü oyuncular doğrudan rekabet etmek yerine oyuncunun oyun alanındaki varlığını ve eylemlerini temsil eden, dijital oyunlarda avatar olarak bilinen, vekil bir varlık

aracılığıyla yaparlar. Bu spor tabanlı oyunların önemli bir kısmı iki oyuncu için tasarlanmıştır; bu da kamusal sosyal alanlarda kullanılan oyunlar için büyük önem taşıyan bir tasarım özelliğidir. Birçok makinede ayrıca büyük miktarda cam bulunmaktadır ve bu kalabalık seyirci kitlelerin oyunun tadını çıkarmasını sağlar.

Şekil 5. Play Football (1926, Chester-Pollard Co.). *History of Digital Games Developments in Art, Design and Interaction* (s. 12).

1930'larda büyük kalabalıkları çeken spor etkinliklerini canlı yayınlayan radyonun popülerleşmesi ve elektriğin eğlence cihazlarında kullanılmaya başlanması gibi gelişmeler jetonlu atari ve langırt oyunlarının tasarımında etkili olur. İkinci Dünya Savaşı'ndan sonraki yirmi beş yıllık dönemde, tüm türlerdeki atari oyunlarında büyük ölçüde mekanik sistemden elektromekanik sisteme geçilir ve bu değişikliğin getirdiği yenilik zamanlayıcı (timer) olacaktır. Zamana bağlı oyun, acemi ve yetenekli oyuncuların öngörülebilir süreler içinde oynaması sayesinde makinenin jeton toplama hızını eşitlemeye yardımcı olur. Ayrıca oyuncuların yalnızca bir görevi sınırlı bir süre içinde yapma zorunlulukları oyun deneyimini daha stresli ve heyecanlı hale getirir.

1950'lerden 1970'lere kadar, jetonlu oyun üreticileri, direksiyon ve pedallar tarafından işletilen zaman ayarlı oyunlara sahip önemli sayıda araç tabanlı eğlence üretir. Başlangıçta, yarış yaygın olarak uygulanan bir konsept değildir. Daha yaygın olarak, oyuncunun güvenli sürüş becerilerini ölçmeyi amaçlayan oyunlardır; sürücü ne kadar

iyiyse, puan o kadar yüksek olacaktır. Elektromekanik dönemin en sofistike yarış oyunlarından biri, oyuncunun direksiyonu olan bir model arabayı kontrol ettiği ve bir yarış alanında diğer arabaları geçerek puan topladığı Road Runner'dır. 1960 ve 70'lerin sonunda yaygınlaşan füze fırlatan oyunların ilk örneği, oyuncunun motorlu bir konveyör bantla çalışan model gemi setlerine torpido fırlattığı Japon yapımı *Periscope* adlı oyundur.

Şekil 6. Road Racer, (1962, Williams Electric Mfg. Co.). *History of Digital Games Developments in Art, Design and Interaction* (s. 22).

1970'lerin sonlarında dijital jetonlu oyunların gelişmeye başlamasıyla atari oyunlarının da niteliği değişir. İlk dijital atari oyunları elektromekanik atari oyunlarında bulunan görsel detay ve derinlik hissinden yoksun olsa da karmaşık hareketli parça dizileri ve kir birikiminin neden olduğu bakım sorunlarından da yoksundur. Daha ayrıntılı elektromekanik langırt ve atari oyunlarının boyutlarına kıyasla, jetonlu video oyunları daha küçüktür, bu da atari salonlarına daha fazla makinenin sığabilmesine ve daha fazla jeton toplanabilmesine olanak sağlar. Dijital olarak tasarlanabilen oyun türleri, elektromekanik oyunlara kıyasla daha dinamik ve aşamalı olarak zorluk derecesi artan oynanış olanakları sunar.

1.2.2. Bilgisayar Oyunlarının Gelişimi

1930'lu ve 40'lı yıllarda Almanya, İngiltere ve Amerika Birleşik Devletleri'nden mühendisler bağımsız olarak çeşitli matematiksel hesaplamalar ve problem çözme işlevleri gerçekleştirebilen elektromekanik ve elektronik hesaplama cihazları geliştirirler. 14 Şubat 1946'da Japonya'nın teslim olmasından tam altı ay sonra, Pennsylvania Üniversitesi ilk programlanabilir bilgisayar ENIAC'ı (Elektronik Sayısal Bütünleştirici ve Hesaplayıcı) geliştirir (Donovan, 2010). Bu son teknoloji ürünü bilgisayarın yapımı üç yıl sürer ve ABD ordusuna 500.000 dolara mal olur. 30 ton ağırlığındaki ordu için topçu atış tablolarını hesaplamak üzere geliştirilen bu cihaz 63 metrekarelik bir alana ihtiyaç duymaktadır. İç kısımlarında 1.500'den fazla mekanik röle ve 17.000 vakum tüpü bulunmaktadır. Ekranı ya da klavyesi olmadığı için talimatlar delikli kartlar kullanılarak girilmekte ve ENIAC da kendi delikli kartlarını basarak yanıt vermektedir. Bu kartların daha sonra bir IBM hesap makinesine girilerek bir anlama dönüştürülür.

Silah ateşleme ve şifreli mesajları çözmek gibi askeri ihtiyaçlara yönelik geliştirilen, üniversite ya da hükümet binalarında özel laboratuvarlarda bulundurulan ve tüm bir odayı kaplayacak büyüklükte bu cihazlar, küçük transistorların kullanımı girmesiyle yerlerini buzdolabı büyüklüğünde "mini" bilgisayarlara bırakırlar (s. 29). 1960 ve 70'li yıllarda boyut ve maliyetteki bu azalma bir yandan bilgisayarlara ulaşımı kolaylaştırırken diğer yandan bilgisayar korsanlığı kültürünün gelişmesine yol açar. Ticari olmayan böylesi bir ortamda, özellikle satranç gibi, yapay zekâ teorilerini örnekleyecek ve diğer alanlara da uygulanacak radikal buluşlara odaklanmayı sağlayacak olan bilgisayar oyunlarının gelişimi, video atari (oyun) salonları ve ev bilgisayarları için oyun tasarlayanların yararlanacakları bir bilgi temeli oluşturur. İlk gerçekten otonom satranç makinesi, yirminci yüzyılın başlarında zeki elektromekanik araçlar geliştirmekte uzmanlaşmış İspanyol mucit Leonardo Torres y Quevedo (1852–1936) tarafından geliştirilir. Satranç bilgisayarların teorik ve hakiki kapasitesini test etmek için II. Dünya savaşından sonra da kullanılmaya devam eder. 1950'lerin sonuna gelindiğinde bilgisayarlar tüm bir satranç oyunu başından sonuna oynayabilir hale gelirler. Satrancın yanında "Nim" gibi oyunlar, bilgisayarların olumsuz sonuçları olan belirli kararları eleyerek daha verimli hamlelere odaklanmasını sağlayan hesaplama tasarrufu

stratejilerini geliştirmelerine yardımcı olur. Bu çalışmaların doruk noktası, 1997'de IBM süper bilgisayarı "Deep Blue"nun son satranç şampiyonu Garry Kasparov'u yendiği ünlü satranç maçı olur.

Yapay zekâ testleri Alan Turing tarafından geliştirilen Taklit Oyunu (Imitation Game) gibi oyun benzeri sistemleri de içerir. Bu testin birinci versiyonu bir erkek ve bir kadın katılımcının cinsiyetlerinin üçüncü bir kişi tarafından sorulara verdikleri cevaplar temelinde tahmin edilmesini içerir. Katılımcılardan biri cinsiyeti tahmin edecek olanı yanıltmak üzere cevaplar verecektir. İkinci versiyonunda yargıyı verecek kişi bilgisayarla yer değiştirecektir. Bilgisayar ilk versiyondaki kişinin bildiği ya da yanıldığı oranda bilir ya da yanılırsa zekaya sahip olduğu söylenebilecektir (s. 33). Turing'den sonra araştırmacılar sorulara cevap veren bilgisayar programları geliştireceklerdir. Bunların arasında Massachusetts Teknoloji Enstitüsü'nde Joseph Wiezenbaum tarafından 1964-66 arası geliştirilen ve bir dil analizcisi ve metin içeren ELIZA adlı program genel bir söyleşiyi devam ettirebilen ilk program olacaktır.

Bilimsel bir bağlamda geliştirilen önceki oyunlardan önemli oranda ayrılan ilk oyun, II. Dünya savaşı sırasında atom bombası üzerinde çalışan nükleer fizikçi William Higinbotham tarafından geliştirilen ve gençlerin bilim alanında kariyer yapmalarını daha fazla özendirmeyi ve aynı zamanda Amerikan teknolojik üstünlüğünü güçlendirmeyi amaçlayan İki Kişilik Tenis (Tennis For Two) oyunu olacaktır. Bu basit ama ilgi çekici oyun, dikey bir çizginin fileyi ve yatay bir çizginin oyun alanını temsil ettiği yandan bir perspektiften görülen bir tenis maçını gösterir. İki Kişilik Tenis oyunu, balistik füzelerin ve diğer füzelerin yörüngesini simüle edebilen analog bir bilgisayara dayanır. Oyuncular topun yörüngesini düğmeli kumanda aracılığıyla ayarlar ve kumandanın tek düğmesine basarak topu rakibine gönderirler. Oyuncu topa her vole vurduğunda, bilgisayarın elektromekanik anahtarları yüksek bir tıkırtı sesi çıkarır, bu da istenmese de sanal eylemler için sesli bir geri bildirim unsuru sağlıyordu. Oyuncular sırayla topa ileri geri vurarak rakiplerini topa garip bir açıyla vurmaya zorlarlar.

Şekil 7. William Higinbotham'ın 1958 yılında New York Brookhaven Ulusal Laboratuvarı ziyaretçi gününde sergilenen İki Kişilik Tenis oyunu (soldan ikinci makine). *History of Digital Games Developments in Art, Design and Interaction* (s. 35).

İki kişilik tenis oyununda da oyuncular, top sahanın kendi taraflarındayken kumandanın düğmesine bastıklarında asla ıskalamazlar. Gerçekliğin bu ihlali, 1960'lar ve 1970'lerde bilgisayar korsanları tarafından yaratılan oyunlarda daha büyük ölçüde geliştirilen ve sadece elektronik formatta mümkün olan bir deneyime yol açar. İki Kişilik Tenis'in değiştirilmiş bir versiyonu "Bilgisayar Tenisi" adı altında 1961 yılı ziyaretçi gününde sergilenir.

Williams (2017) bilgisayar korsanlığını (Hacking), “elektronik bilgisayarların programlanmasında doğal olarak kendini ifade eden yaratma, bozma ve tamir etme döngülerine odaklanan bir deney ve çözüm bulma biçimi” olarak tanımlar (s. 36). Bilgisayar oyunu geliştirmek bu deney ve çözümlerin mantıksal bir sonucu olacaktır. Üniversitelerin ve araştırma enstitülerinin laboratuvarlarına giren bilgisayarlardan anlayan bir grup insanın alandaki egemenliğine bir tepki olarak bilgisayarların çalışma biçimleri ve imkanları üzerine kafa yoran bilgisayar korsanlarının, 1960 ve 70’lerin karşı-kültür hareketlerinin ruhunu paylaştığı söylenebilir. 1960'lar ve 1970'lerdeki pek çok bilgisayar korsanı için bilgisayar oyunları kendinde bir amaç, yeteneklerini zorlayan ve

bilgisayarlar hakkındaki bilgilerini ilerleten etkileşimli bir öğrenme projesiyken bazıları için bir meslek haline gelmiştir. Dolayısıyla ilk bilgisayar korsanlarının kolektif çalışması, ticari dijital oyun endüstrisinin kurulmasında çok önemli bir rol oynamıştır.

1.2.3. Bilgisayar Oyunlarının Ticarileşmesi

1970'ler tüketici sektöründe eğlence ve üretkenliğin biçimini yeniden şekillendiren ve fiyatlarda önemli bir düşüşe ve toplam donanımda fiziksel bir azalmaya yol açan teknolojik devrimlerin yaşandığı bir on yıl olmuştur. Mikroişlemciler ve mikrodenetleyiciler hesap makinelerinde, mikrodalga fırınlarda, otomobillerde ve daha birçok alanda ortaya çıkarak yeni hız ve otomasyon imkanları sağlamıştır. Bilgisayar tarafından manipüle edilen ve bilgisayar tarafından üretilen görüntüler ilk olarak Willy Wonka ve Çikolata Fabrikası (1971), Westworld (1973) ve Yıldız Savaşları (1977) gibi filmlerde ortaya çıkmıştır (s.51). İlk taşınabilir bireysel müzik dinleme cihazı Walkman 1979 yılında Sony tarafından piyasaya sürülmüştür. Yine 1970'lerde hobi amaçlı programcılığı başlatan "mikrobilgisayarlar" vasıtasıyla tümüyle elektronik video oyunlarının ilk defa oyun salonlarına ve evlere girdiği görülmüştür. Yirminci yüzyılın sonuna gelindiğinde, video oyunları giderek daha karmaşık hale gelir ve film mecrasıyla daha yakından ilişkilendirilir (Potter ve Romano, 2012, s. 204). Oyuncuların sanal kürekleri veya tenis raketlerini manipüle ettiği Pong'un ilk günlerinden itibaren, video oyunları oyuncuların bir karakter için hareket etmesine izin verecek şekilde gelişmiştir. Video oyunları, oyuncuları daha önce sinemada canlandırılmış karakterlerle kaynaştıran anlatı odaklı deneyimler haline gelmiştir. Böylece, büyük ölçüde Hollywood'un teknik bilgisine dayanan ve sinema endüstrisinin devlerinden esinlenerek şekillenen video oyunu endüstrisi sinema endüstrisi simbiyotik bir ilişki kurmuştur.

İlk ticari bilgisayar oyunlarının bazıları tasarım olarak mekanik ve elektromekanik oyunları elektronik bir araca uyarlar ve dolayısıyla oyuncuyu kavramsal olarak çok zorlamazken, diğer bir yaklaşım, kabul edilmiş norm ve geleneklerin ötesine geçmekten sakınca görmemiştir. Buna karşın tümüyle elektronik bilgisayar oyunlarının ticari açıdan en başarılıları her ikisini de birleştiren, yani aşina olunanı yeni bir biçimde sunan oyunlar olmuştur. Ekranda iki ışı karesi ve bir mermi sergilemek için televizyon ev televizyon setini kullanan Magnavox tarafından Odyssey adıyla 1972 yılında piyasaya

sürülen cihaz ilk kuşak ev videolarının başlangıcı olmuştur (Williams, 2017, s. 55). Odyssey ile oynanan oyunların her biri bir ya da daha fazla oyun kartı kullanır. 1970'lerin sonunda Asteroids14, Pac-Man15 ve Space Invaders16 gibi çok daha gelişmiş oyunlar piyasaya çıkar ve tam anlamıyla kült statüsü kazanırlar (Rokošný, 2018, s. 52) Genellikle atari oyunlarının altın çağı olarak adlandırılan bu dönem, nihayet oyunları ticarileştirdi ve genel halkı kendine çekti. Space Invaders'ın gelişiyle birlikte oyunlar o kadar popüler hale gelir ki, ABD'de alışveriş merkezlerinin yanı sıra neredeyse her yerde -alkollü içki satan dükkanlarda, benzin istasyonlarında, havaalanlarında ve oldukça tuhaf bir şekilde cenaze levazımatçılarında- bulunabilir hale gelmiştir. Altın Çağ olarak adlandırılan bu dönemde Pseudo (sözde) 3D görüntüler ve raster grafik yerine vektör grafiği ile ilgili çalışmalar artar ve bu da teknolojik ilerlemeyle neden olmuştur.

Şekil 8. Magnavox Odyssey tüketici modeli (1962, Williams Electric Mfg. Co.).
History of Digital Games Developments in Art, Design and Interaction

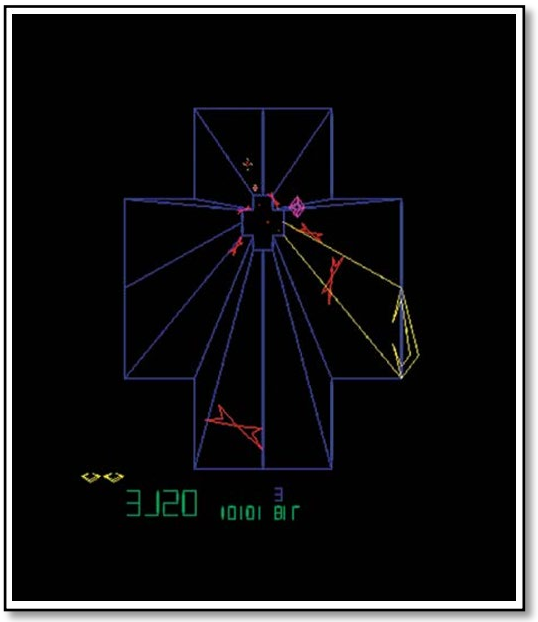
En eski dijital atari oyunları elektrik mühendisleri tarafından geliştirilen transistör-transistör mantığına dayanmakta, ekrandaki oyun kuralları ve davranışları, her oyun için özel olarak yerleştirilen ayrı fiziksel devrelerden oluşmaktaydı. Buna karşın mikroişlemci tabanlı bir oyunda, oyunun kuralları ve davranışları, daha standartlaştırılmış bir donanıma talimatlar veren yazılım olarak bulunur. Transistör-transistör mantığı ile karşılaştırıldığında, mikroişlemciler önemli ölçüde daha hızlıdır ve bu daha karmaşık oyunlar geliştirilmesine olanak tanımaktadır. Mikroişlemcilerin benimsenmesi, oyunların fiziksel tasarımında ve geliştirilmesinde bir dizi değişiklik getirecektir. Üreticiler tek tip

kart setleri oluşturabilecekleri için oyunları daha hızlı üretilmiştir. Bunun önemli bir sonucu olarak da oyun tasarımındaki ana rol mühendislerden bilgisayar programcılarına geçmiştir. 1970'lerin sonlarından 1980'lere kadar, programcılar genellikle sadece oyunu değil, aynı zamanda grafikler ve animasyonlar tasarlayabilirler. Mikro işlemciler ekonomikleştikçe, onlarca büyük ve küçük oyun şirketi 1970'lerin ortalarından sonlarına kadar bir dizi özel konsolla yeni oluşan ev pazarına girer. Daha sonraki kartuş tabanlı konsolların aksine, özel konsollar sabit sayıda yerleşik oyun içermektedir. 1975 yılında Magnavox Odyssey'i üretimden kaldırır ve yerine televizyondan bağımsız, özel bindirmeleri ve temel ses özelliklerini içeren Odyssey 100'ü getirir. Magnavox, 1975 ve 1976 yılları arasında beş özel konsol çıkarır. 1977 ve 1978 yıllarında Japon oyuncak üreticisi Nintendo, ilk ev oyun sistemleri olan Color TV Game 6 ve Color TV Game 15'i piyasaya sürer.

Şekil 9. Odyssey 300 (1962, Williams Electric Mfg. Co.). *History of Digital Games Developments in Art, Design and Interaction*

1970'lerin sonu ve 1980'lerin başında atari oyunları sanat ve tasarımda istisnai bir yaratıcı döneme girmektedir. Aynı zamanda ikinci nesil ev konsollarında kartuş kullanılmaya başlanılması neredeyse sınırsız bir oyun kütüphanesi oluşturma imkânı oluşturmuştur (Williams, 2017, s. 70). Oyun tasarımcıları Space Invaders'da uzaylılardan biri olan Crab, "Jump Man" karakteri ve çok renkli vektör ışın grafikleri gibi dijital oyun tarihinin en ikonik konseptlerinden, karakterlerinden ve görsellerinden bazılarını yaratılır. Oyun tasarımcıları, Qix'te (1981, Taito) doğrusal hatlar çizerek rastgele hareket eden bir düşmana karşı alan kazanmak ve Joust'ta (1982, Williams Electronics) devekuşuna binmiş şövalyelere karşı gerçeküstü yarışmalar gibi soyut ve alışılmadık kavramlar

geliştirirler. Bu yaratıcı gelişmeler ve keşifler, bir bakıma kamunun dijital oyunlara ilişkin gelişen kavrayışı, oyun üreticilerinin hazırladıkları etkili oyun kılavuzları ve *Yıldız Savaşları* (Star Wars) gibi bilim kurgu filmlerinin hem oyun tasarımcılarına hem de oyunculara sağladığı yeni referansla bağlantılıdır (s. 71). Bu dönemin oyunlarıyla erken dijital oyunlar arasındaki temel fark, öncekilerin belirli bir süre içinde performansa bakmaksızın yekpare ve sürekli oluşları, yeni oyunların ise kısa fakat daha yoğun ve zorluk dereceleri gittikçe artan ve yeni kurallar, düşmanlar ve mekanlar sunan aşamalara ayrılmalarıdır. Altın Dönem (Golden Age) olarak adlandırılan bu dönemdeki önemli gelişmelerden biri, vektör grafiklerinde görüntü oluşturmak için yeni bir yöntemin yaygın kullanımıdır. Vektör grafikleri bilgisayarlar için yeni olamamakla birlikte oyun salonları için yenidir. Vektör ekranlar, şekillerin ana hatlarını oluşturmak için yüksek hızda yenilenen ince bir ışık demetiyle birbirine bağlanan nokta kümelerini kullanmaktadır. Görsel olarak vektör grafikleri, önceleri baskın olan ve 200 × 180 piksel çözünürlüğe sahip raster tipi grafiklerden öncelikle netlikleri ve yaklaşık 1024×768 piksel çözünürlükte jilet keskinliğinde çizgileri ve yoğun parlak görüntüler oluşturma yetenekleriyle ayrılmaktadır. Başlangıçta siyah beyaz olan vektör grafikleri de 1980'lerin başında renkliye dönüşür.



Şekil 10. Tempest (Atari, 1981)

Genel olarak, sadece 90 saniyelik bir oyun için tasarlanmış olan jetonlu oyunların tarihi, olay örgüsü ve karakter gelişiminin ortaya çıkması için genellikle daha fazla zaman gerekmesinden dolayı karakterler ve anlatılar açısından çok az şey içermektedir. Buna karşın 1980'lerin jetonlu oyun tasarımcıları karakterlerine ve ortamlarına daha fazla kişilik ve farklılık kazandırmayı amaçlarlar. Bunun için *Robotron 2084* gibi bazı oyunlar metin içerirken, diğerleri etkileşimli olmayan animasyonlar kullanırlar. Anlatının dahil edilmesi nihayetinde tasarımcıların ortamın etkileyici niteliklerini daha fazla keşfetmelerine ve daha önceki jetonlu elektromekanik oyunların kısıtlamalarından kurtulmalarına olanak sağlar. Nitekim dört bölümden oluşan bir seri anlatısal kavrama dayanan *Donkey Kong* adlı oyun, "Nintendo'nun hakimiyetinin başlangıcını işaret ederek ve oyunlarda daha önce çok az ilgi gören iki alan olan sanat ve anlatının önemini göstererek dünya çapında başarılı olur" (s. 83).

Şekil 11. Robotron 2084, (1962, Williams Electric Mfg. Co.). *History of Digital Games Developments in Art, Design and Interaction*

1980'lerin başındaki atari salon oyunları, daha fazla içerik çeşitliliği ile daha uzun süreli oyun sunabilen ev konsolları ve ev bilgisayarlarının artan rekabetiyle karşı

karşıya kalmıştır. Arcade oyunları, özellikle 1980 sonrasında, ikinci nesil ev konsolu donanımı ve oyunlarının tasarımcıları üzerinde en büyük etkiye sahip olmuştur. Bunun büyük bir kısmı, belirli jetonlu oyunların kendini kanıtlanmış tasarımlarını kopyalama arzusundan kaynaklanmaktadır. Buna karşın makineye bir çeyreklik atma ihtiyacının olmadığı ev video oyunlarında oyuncu sahip olduğu en büyük seçim hakkını, oyunu bir daha oynayıp oynamama seçimini yitirir. Tekrar oynatma özelliği sayesinde ev oyunları daha sık oynanacaktır. Hızlı ölümler ve zorluk derecesindeki hızlı artışlar, oyun kartuşu satın alanların yüksek maliyetini haklı çıkarmayacaktır. Bu nedenle tasarımcılar, atari oyunlarını keyifli kılan nitelikleri kullanmaya devam ederken oyuncunun etkileşim süresini uzatacak bir yola ihtiyaç duyarlar. Oyunu uzatmaya yönelik en yaygın çözüm, oyuncunun oyunun zorluk seviyesini seçmesine izin vermek olacaktır. Bu özellik tipik olarak alınan can sayısını, düşmanların hızını veya oyuncunun performansına yardımcı olan veya engelleyen diğer değişkenleri yönetmektedir. Bir diğer tipik yaklaşım ise kuralları değiştirerek oyunun doğasını ve elde edilen deneyimi önemli ölçüde değiştirebilen bir "oyun seçme" seçeneğidir. Örneğin *Space Invaders*'tan esinlenen *Alien Invasion* (1981), oyuncunun hem oyuncu hem de uzaylılar tarafından bir seferde yapılan atış sayısını kontrol etmesine izin veren 10 varyasyon içermektedir.

1.2.4. Mikrobilgisayar Devrimi ve Bilgisayar Oyunları

1970'lerin sonu ve 1980'lerin başı, macera oyunları, rol yapma oyunları ve simülasyon oyunları gibi, 1970'lerde bilgisayar korsanları tarafından minibilgisayarlar için üretilen oyunlarla bağlantılı üç benzersiz ticari bilgisayar oyunu türünün gelişimine tanıklık eder. 1970'lerin ortalarından sonlarına doğru ilk "mikrobilgisayarlar" ortaya çıkar (s. 113). PDP-1 gibi mini bilgisayarlara bağlı zaman paylaşımlı terminallerin aksine, mikrobilgisayarlar bireysel kullanım için tasarlanmıştır.

Şekil 12. PDP-1 mini bilgisayarda oynanan Spacewar adlı oyun (History of Digital Games, s. 39).

Şekil 13. Spacewar'ın yaratıcıları Dan Edwards (solda) ve Peter Samson, 1962 dolaylarında

Yeni sınıf mikrobilgisayarlar hem işletmelere hem de bireylere pazarlandıkça, bilgisayar bilgisi gelecekteki siyasi, ekonomik ve kültürel gelişim için belirleyici olur. Üniversiteler ve ilkokullar, eğitim yazılımları, kelime işlemcileri ve hatta programlama dilleri aracılığıyla bilgisayarların nasıl kullanılacağını öğrenmeye başlanmıştır. Mikrobilgisayarların yetenekleri daha büyük öncüllerine kıyasla bellek ve işlem gücü açısından sınırlı olduğundan, programcılar için özellikle gelişmiş grafikler için bellek tasarrufu sağlayan hileler kullanmak çok önemli hale gelmiştir. Yaygın bir yöntem olarak, programcının işlemcinin bireysel eylemlerini kısaltılmış kelimeler ve sayılar aracılığıyla yönlendirdiği assembly programlama dili kullanılmıştır. Bu durum, açık kelime komutlarını otomatik olarak bilgisayarın anlayabileceği çoklu eylemlere çeviren ve

yazması çok daha ekonomik olan BASIC gibi üst düzey kodlama dilleriyle tezat oluştursa da -örneğin BASIC ile tek satır kodla yazılan “merhaba dünya” yazısının görüntüsünü oluşturmak için assembly dilinde 12 satır kod yazmak gerekir- assembly ile yazılan oyunlar BASIC ile yazılanlardan hızlı olduğu için çok daha iyi performans gösterirler ve bu da önemli bir kalite farkına yol açar (s. 115). İlk interaktif kurgu oyunu, 1975 yılında Will Crowther tarafından bir üniversite ana bilgisayarında geliştirilen ve Kentucky'deki Mamut Mağarası'nın keşfini simüle eden *Adventure* ya da *Colossal Cave Adventure* adlı oyundur.

1980'lerin ortalarından sonlarına doğru piyasaya sürülen, daha güçlü multimedya özellikleriyle tasarlanmış yeni nesil 16 ve 32 bit bilgisayarlar grafik tasarım, video prodüksiyonu, animasyon ve diğer yaratıcı alanlardaki devrimi hızlandırarak profesyonel uygulamaları temelden dönüştürür. Yeni bilgisayarlar daha yüksek ekran çözünürlükleri ile özel olarak tasarlanmış bilgisayar monitörlerini gerekli hale getirerek oturma odası televizyonunu bir bilgisayar ekranı olmaktan çıkarır. Ev bilgisayarlarının yetenekleri arttıkça, daha büyük depolama kapasitesine olan talep de artar ve programların ve dosyaların boyutu sabit diskleri bir diğer önemli bileşen haline getirir. Bilgisayarlar ve oyun tasarımı için asıl önemli değişiklikler Apple'ın Macintosh'u ile gelir. Daha geniş bir kitle tarafından kullanılmak üzere tasarlanan Macintosh, fare ve grafik kullanıcı arayüzü (GUI) eşleştirmesiyle ulaşılabilir hale gelir. Talimatları vermek için fareyi kullanmak, bilgisayar etkileşimlerine uzamsal bir boyut kazandırarak çok sayıda metin komutunu ezberleme ihtiyacını ortadan kaldıracaktır. Infocom ve Sierra Online fare etkileşimi için macera oyunları geliştirler. Sierra On-Line'da Roberta Williams'ın King's Quest V: Absence Makes the Heart Go Yonder! (1990) adlı oyunu metin kullanımını tamamen terk ederek karakteri hareket ettiren ve oyun dünyasıyla etkileşime giren simgeler aracılığıyla yalnızca fare girişine odaklanır (s. 134). 1990'ların başlarından itibaren tüketici bilgisayar pazarı, Apple'ın Macintosh bilgisayar serisi ve yalnızca belleği değil, aynı zamanda işlemci ve anakart gibi donanım bileşenlerini de yükseltme olanağı veren açık bir mimari tasarıma sahip IBM PC etrafında toplanır. Compaq, Gateway, Dell ve Hewlett-Packard gibi şirketler tarafından satılan "IBM uyumlu" bilgisayarlar 1980'ler ve 1990'lar boyunca bu tasarım özelliğini tekrarlarlar.

Şekil 14. 1990'ların ortalarından IBM uyumlu bir bilgisayarın iç kısmı.

1990'ların ortalarından sonlarına doğru oyun teknolojilerinde yaşanan hızlı evrim, gerçek zamanlı çokgen tabanlı 3D oyunlarda mümkün olan uygulama yelpazesini ve ayrıntı düzeyini büyük ölçüde geliştirir. Oyun geliştiriciler, 2D görüntülere bağımlı kalmadan tam 3D oyunlar yaratırlar ve gerçek zamanlı 3D geliştikçe, oyunlarda hem oyun hem de anlatı amaçlı film kullanımına olan ilgi azalarak milenyumun başında büyük ölçüde ortadan kalkar. IBM uyumlu bilgisayarlar için yeni donanım yükseltmeleri, PC oyunlarının temel doğasını değiştirerek ve onu metodik, sıra tabanlıdan (turn-based) hızlı tempolu 3D oyunlar için önde gelen bir platforma dönüştürür (s.185). Bu gelişmelerle paralel olarak ev konsollarının donanımı da değişecektir. 1994 ve 1996 yılları arasında Sega, Sony ve Nintendo'nun her biri yeni bir "beşinci nesil" ev konsolu piyasaya sürerler. Başlangıçta Dural ve Black Belt kod adlarıyla anılan Sega'nın yeni 128-bit sistemi 1998'in başlarında resmi olarak Katana olarak adlandırılır (Leonard Herman, 2002, s. 22). Sega aynı zamanda yeni sistemin Microsoft Windows CE işletim sistemi kullanacağını, bunun da PC'ye ve PC'den daha kolay oyun dönüştürme anlamına geleceğini açıklar. 1999 yılında Nintendo, Dolphin kod adlı yeni konsolunu duyurur. Konsol, IBM tarafından üretilecek olan Gekko adlı 400MHz bakır mikroçip teknolojisi üzerine inşa edilecektir. Yine aynı dönemde Microsoft, işletim sistemi olarak Windows CE'nin bir versiyonunu kullanacak olan X-Box kod adlı bir ev konsolu sistemi üzerinde çalıştığını duyurur. 2000

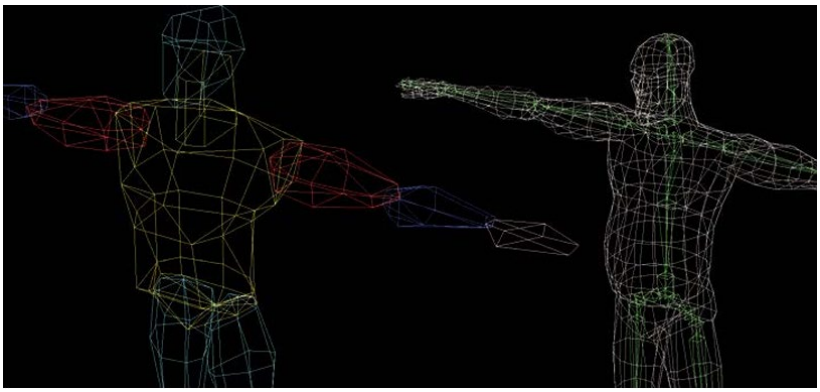
yılında Sony PlayStation 2'yi Japonya'da piyasaya sürer. Yine Indrema adlı bir yeni bir şirket 2001 yılında yeni bir oyun konsolu çıkarmayı vaat eder. Linux işletim sistemi kullanan Indrema L600, oyun, DVD ve CD oynatacak ve hatta sabit diskine TV şovları kaydedecektir. Microsoft söz verdiği gibi 2001 yılında xbox'ı dünya çapında piyasaya sürer (Kent, 2001). Aynı yıl Sony dünya çapında 80 milyondan fazla PlayStation ve Nintendoh 110 milyondan fazla GameBoy satmıştır (s. 590).

Nintendo'nun o zamanki başkanı Iwata Satoru, mühendislerinden üç DVD kutusundan daha kalın olmayan ve kalabalık bir oturma odası eğlence merkezine kolayca yerleştirilebilecek bir konsol geliştirmelerini ister (World Video Game Hall of Fame, 2018). En önemlisi bu konsol kendilerini asla oyuncu olarak görmeyen insanlara, örneğin bir anneye de hitap etmelidir. Konsolda devrimin kumandayla başlayıp kumandayla biteceğine inan Satoru, yeni nesil konsolun kod adı "Devrim" koyar. Aksiyonu kontrol etmek için D-pad'leri, joystick'leri, düğmeleri ve tetikleyicileri kullanmak yerine, oyuncunun birincil giriş cihazı kontrol cihazının kendisi olur. Nintendo sonunda yeni konsoluna, oyunların toplumsal yönünü çağrıştırmak için "Wii" adını verir- iki "i" karakteri yan yana duran iki oyuncuya gönderme yapar.

Nintendo mühendisleri ivmeölçerler kullanarak bir oyuncunun elini yukarı, aşağı, sola ve sağa hareket ettirdiğinde ve herhangi bir yönde hızlandığında bunu kaydedebilen bir "uzaktan kumanda" geliştirirler. Nintendo'nun ünlü NES Zapper tabancasının manevi halefi olan uzaktan kumanda, oyuncunun sanal nesneleri sadece ekranda işaret ederek manipüle etmesine olanak tanıyacaktır. Ancak Wii uzaktan kumandasının gerçek dehası, ekranda kopyalanması amaçlanan nesnenin şeklini alabilmesinde yatmaktadır. Mario Kart ve diğer yarış oyunları için, uzaktan kumanda yana doğru çevrilerek bir direksiyon gibi kullanılabilmektedir. Uçuş simülatörleri için bir joystick olurken, Fishing Master'da, bir sinek oltası olarak ikiye katlanabilecektir. Basit, sezgisel oyunlarla oyuncu tabanını genişleten Nintendo, yüz milyondan fazla Wii konsolu satarak Microsoft Xbox 360 ve Sony PlayStation 3'ü geride bırakır ve rakip şirketleri kendi hareket algılayıcı kontrol cihazlarını tanıtmaya zorlar.

2003 yılında Activision, Amerika Birleşik Devletleri'nin Irak'ı işgal etmesinin üzerinden den henüz birkaç ay geçmişken ve savaş haberleri her gece televizyonlardan verilirken, Medal of Honor'a benzeyen ancak Amerikan, İngiliz ve Sovyet bakış açılarından birden fazla hikâye içeren birinci şahıs nişancı oyunu Call of Duty'yi piyasaya sürer. Video oyunları genellikle çocuklarla ilişkilendirilse de Eğlence Yazılımları Derneği'ne (Entertainment Software Association) göre video oyunu oynayanların yaş ortalaması otuz beştir. Oyuncular reşit yaşa geldikçe, oynamayı seçtikleri oyunlar da reşit olmaktadır. Sevimli Mario ve Pac-Man video oyunlar popülaritelerini devam ettirmekle birlikte, Call of Duty: Modern Warfare 2 (2009), The Walking Dead (2012), The Last of Us (2013), ve Grand Theft Auto V (2013) gibi popüler oyunlar yetişkin oyunculara hitap etmektedir.

21. yüzyılın ilk on yılı, donanımdaki gelişmelere paralel olarak oyun grafiklerinde hızlı bir evrime tanıklık eder (Williams, 2017, s. 209) . Bazı oyunlar tamamen foto-gerçekçiliğin peşinde koşarken, çoğu gerçekçi ışıklandırma ve doku efektleriyle stilize görsellerin bir karışımını kullanır. Geliştiriciler daha fazla ayrıntı sağlamak ve yeni donanım özelliklerine uymak için poligon sayısını artırdıkça, 3D oyun modellerinin karmaşıklığı her iki durumda da önemli ölçüde artacaktır. Örneğin 90'larda karakter modelleri birkaç yüz poligonla elle çizilirken, 2010'larda kullanılan poligon sayısı on binler ila yüz binlere arasındadır. Bu geometrik karmaşıklıktaki artış, 3D modellerin canlandırılış biçimlerini de etkileyecektir.



Şekil 15. 1990'ların sonları (sol) ve 2000'ler (sağ)

Soldaki düşük poligonlu figür tek tek geometrik parçalardan oluşturulurken, sağdaki yüksek poligonlu figür süreklidir ve iç kemiklerin manipüle edilmesiyle canlandırılır. Animasyon üretmek için karakterler, yüksek çokgen yüzeyleri düzgün bir şekilde deforme etmek ve dirsek ve diz bükme gibi eylemleri simüle etmek için modelin içinde bulunan kemiklerle donatılır. Düşük poligonlu 3D modellerdeki sınırlı sayıda yüzey olarak yeterince hassas bir biçimde bükülmediği için istenmeyen veya doğal olmayan etkiler yaratmaktadır. Parçalar halinde inşa edilen düşük poligonlu modeller ile tek bir yüksek poligonlu model arasındaki görsel farklar, aralarında sadece 6 yıl bulunan iki oyun olan Star Wars: Jedi Knight-Dark Forces 2 (Görsel …) ve Knights of the Old Republic (Görsel …) oyunlarında görülebilir. 2000'li ve 2010'lu yılların sonlarında, karakter modelleri genellikle bir aktörün vücudunun 3D taramalarıyla oluşturulurken, hareket yakalama verileri daha akıcı ve gerçekçi animasyonlar sağlayacaktır.

Şekil 16. Yıldız Savaşları: Jedi Şövalyesi-Karanlık Güçler II (1997, LucasArts).

Şekil 17. Yıldız Savaşları: Eski Cumhuriyet Şövalyeleri (2003, BioWare Corporation).

2000'lerden önce 3D oyun görselleri, belirli bir malzemeyi simüle etmek için 2D görüntüleri 3D yüzeylere saran ve örneğin aynı basit küpü tahta bir sandığa ya da metal bir kutuya dönüştürebilen renk haritalarına dayanmaktadır. Ancak tek başına bu renk haritaları ikna edici değildir, çünkü ilk 3D modellerin düşük poligon sayısı nedeniyle yüzeyler düz ve nispeten özelliksiz kalır. Ancak gölgelendiricilerin ortaya çıkışı, 3D modellerdeki yüzeylerin görünümünü değiştirebildikleri için oyunlardaki görsel gerçekçiliği önemli ölçüde artırır.

Şekil 18. Saints Row IV (2013). Soldan sağa: (1) tel kafes model, (2) normal harita, (3) renk haritası ve (4) yansımalar ve son detaylarla birlikte kompozit model.

2. BÖLÜM

OYUN MOTORLARI VE GÖLGELENDİRİCİLER

2. BÖLÜM

OYUN MOTORLARI VE GÖLGELENDİRİCİLER

2.1. Oyun ve Görüntü İşleme Motorları ve Unity

Oyun motorları programcıların, tasarımcıların ve sanatçıların video oyunları da dahil olmak üzere (ancak bunlarla sınırlı olmaksızın) iş birliği içinde gerçek zamanlı etkileşimli dijital içerik oluşturmalarını ve bu içeriği konsollar, akıllı telefonlar ve sanal gerçeklik cihazları gibi farklı platformlarda çalıştırmalarını sağlayan bir kod çerçevesi sağlarlar (Nicoll & Keogh, 2019, s. 3-9). Ancak bu kod oyunla ilgili herhangi bir kod içermemeli ve yeniden kullanılabilir olmalıdır (Düvel & Oliver, 2004, s. 6). Bu anlamda bir motor projeden bağımsız olmalı ve motorun kodunu değiştirmeye gerek kalmadan diğer video oyunu projeleri ve oyun dışı multimedya yazılım projeleri ile çalışabilmelidir.

Bir anahtar vasıtasıyla çalıştırıldığında abayı itme işlevi gören motora benzer bir biçimde, bir 3D motoru grafik adaptörünü çalışmaya hazır hale getirir ve 3D modellerin ekranda görünmesini sağlar. Motorun görevi, grafik adaptörüyle iletişim kurmak, görüntü işleme işlemlerini ayarlamak, modelleri dönüştürmek ve abartılı matematikle (rotasyon matrisleri gibi) uğraşmak gibi temel işleri yapmaktır (s. 3-4). İster yazılım ister donanım tabanlı olsun, gerçek zamanlı bir oyun motoru, sahne grafiği yönetimi konularıyla ilgilenmenin yanında, karmaşık ve hareketli nesneleri fiziksel olarak gerçekçi bir şekilde işleme, çarpışmayı algılama, kavisli yüzeyleri, poligonal modelleri, karakterlerin animasyonunu, geometrik ayrıntı düzeyini, arazi yönetimini ve uzamsal sıralamayı destekleme özelliğine sahip olmalıdır (Eberly, 1999, s. xxvii)

Gölgelendiricilerin ortaya çıkışı, grafik uygulamaları yazma şeklinde değişikliklere neden olur, çünkü artık grafik adaptörünün sürücüsünün kullanıcıya ve ana işlemcide çalışan programlara sunduğu donanımsal işlevleri kullanmak yerine, grafik adaptöründe bulunan işlemci için özel programlar yazma imkânı bulunmaktadır. Bu bakımdan günümüzde sorun iş yükünü merkezi işlem birimi (CPU) olarak adlandırılan ana kart işlemcisiyle grafik adaptörü üzerinde bulunan grafik işlem birimi (GPU) arasında eşit olarak dağıtma meselesidir (Düvel & Oliver, 2004, s. 8).

Oyun motorları tipik olarak türe özgüdürler (Gregory, 2019, s. 13) Örneğin iki kişilik bir dövüş oyunu için tasarlanan bir motor, devasa çok oyunculu çevrimiçi oyun motorundan veya birinci şahıs nişancı (first person shooter) motorundan veya gerçek zamanlı strateji (real time strategy) motorundan çok farklı olacaktır. Bununla birlikte, büyük ölçüde örtüşme de söz konusudur -türü ne olursa olsun tüm 3B oyunlar, joypad, klavye ve/veya fareden bir tür düşük seviyeli kullanıcı girişi, bir tür 3B ağ oluşturma, çeşitli yazı tiplerinde metin oluşturma dahil bir tür baş üstü ekranı (HUD), güçlü bir ses sistemi gerektirir. Her biri kendine özgü teknoloji gereksinimlerine sahip oyun türleri arasında birinci şahıs nişancı oyunları (Quake, Unreal Tournament, Half-Life, Battlefield, Destiny, Titanfall ve Overwatch), platform oyunları ve diğer üçüncü şahıs oyunları (Space Panic, Donkey Kong, Pitfall!, Super Mario Brothers, Super Mario 64, Crash Bandicoot, Rayman 2, Sonic the Hedgehog, Jak and Daxter serisi, Ratchet & Clank serisi, Super Mario Galaxy), kavga oyunları (Soul Calibur, Tekken), yarış oyunları (Gran Turismo, San Francisco Rush, Cruis'n USA, Hydro Thunder, Need for Speed, Juiced, Mario Kart, Jak X, Freaky Flyers), strateji oyunları (Dune II: The Building of a Dynasty, Warcraft, Command & Conquer, Age of Empires ve Starcraf), devasa çok oyunculu çevrimiçi oyunlar (Guild Wars 2, EverQuest, World of Warcraft, Star Wars Galaxies), içeriği oyuncu tarafından yazılan oyunlar (LittleBigPlanet, LittleBigPlanet 2, LittleBigPlanet 3, Dreams) ve sanal, artırılmış ve karma gerçeklik oyunları, konum tabanlı oyunlar (Pokemon Go), spor oyunları (futbol, beyzbol, futbol, golf, vb.), rol yapma oyunları (RPG), tanrı oyunları (Populous, Black & White), çevresel/sosyal simülasyon oyunları (SimCity, The Sims), bulmaca oyunları (Tetris) elektronik olmayan oyunların dönüşümleri (satranç, kart oyunları, go) web tabanlı oyunlar (Electronic Arts'ın Pogo sitesinde sunulanlar) bulunur (Gregory, 2019, s. 14-31).

Yukarıda türlerini belirttiğimiz oyunların geliştirildiği belli başlı motorları söyle sıralayabiliriz: The Quake Family of Engines, Unreal Engine, The Half-Life Source, Frostbite (DICE), Rockstar Advanced Game Engine (RAGE), PhyreEngine (Sony), XNA Game Studio (Microsoft) ve Unity. Bunların yanında çeşitli ticari oyun motorları, açık kaynak (open source) motorlar, tescilli şirket içi motorlar ve ayrıca programcı olmayanlar için 2B oyun motorları bulunur (s. 31-37). Bir oyun motoru genellikle bir araç takımı ve

bir çalışma zamanı bileşeninden oluşur. Tüm yazılım sistemleri gibi, oyun motorları da katmanlar halinde inşa edilir. Normalde üst katmanlar alt katmanlara bağlıdır ve döngüsel bağımlılık (circular dependency) olarak adlandırılan tersi durum sistemde karışıklığa, istenmeyen eşleşmelere, dolayısıyla yazılımda kararsızlığa neden olabilir.

Hedef donanım (target hardware) katmanı, oyunun üzerinde çalışacağı bilgisayar sistemini veya konsolu temsil eder. Tipik platformlar arasında Microsoft Windows, Linux ve MacOS tabanlı PC'ler; Apple iPhone ve iPad, Android akıllı telefonlar ve tabletler, Sony PlayStation Vita ve Amazon Kindle Fire gibi mobil platformlar; Microsoft'un Xbox, Xbox 360 ve Xbox One, Sony PlayStation, PlayStation 2, PlayStation 3 ve PlayStation 4 ve Nintendo'nun DS, GameCube, Wii, Wii U ve Switch gibi oyun konsolları yer alır (s. 38). *Aygıt sürücüleri* (device drivers), işletim sistemi veya donanım satıcısı tarafından sağlanan düşük seviyeli yazılım bileşenleridir. Sürücüler donanım kaynaklarını yönetir ve işletim sistemi ile üst motor katmanlarını sayısız donanım cihazı çeşidiyle iletişim kurma gibi ayrıntılardan korur. Bilgisayarda sürekli çalışmakta olan, örneğin Microsoft Windows gibi, bir işletim sistemi (operation system) önleyici çoklu görev olarak bilinen zaman dilimli bir yaklaşımla oyun dahil birden fazla programın yürütülmesini düzenler. Bu, bir PC oyununun asla donanımın tam kontrolüne sahip olmadığı anlamına gelir. Çoğu oyun motoru, DirectX, OpenGL, Vulkan, Havox, PhysX, Ode, Boost, Folly, Kynepse, Granny, Havok Animation, Euphoria v.b. bir dizi *üçüncü taraf yazılım geliştirme kiti*nden (software development kits) ve *ara yazılım*dan yararlanır. Bir yazılım geliştirme kiti tarafından sağlanan işlevsel veya sınıf tabanlı arayüz genellikle *Uygulama Programlama Arayüzü* (Application Programming Interface ya da kısaca API) olarak adlandırılır.

Sony'nin Naughty Dog ve Insomniac stüdyoları gibi birinci parti stüdyolar dışında çoğu şirket (örneğin Electronic Arts ve ActivisionBlizzard Inc.), oyunlarını mümkün olan en geniş pazara sunabilmek için, birden fazla donanım platformunda çalışabilecek oyun motorları geliştirmeyi hedeflerler. Bu nedenle çoğu oyun motoru, sürücülerin, işletim sisteminin ve diğer üçüncü taraf yazılımların üzerine oturan ve belirli arayüz işlevlerini oyun geliştiricisi olarak her hedef platformda kontrol sahibi olmanızı sağlayacak özel işlevlere "sararak" motorun geri kalanını altta yatan platforma ilişkin

bilgilerin çoğundan koruyan, platformdan bağımsız bir katmanla tasarlanmıştır (Platform Independence Layer).

Modern oyun motorları (duman, ateş, su sıçramaları vb.) parçacık sistemleri; (kurşun delikleri, ayak izleri vb.) çıkartma sistemleri; ışık eşleme ve ortam eşleme; dinamik gölgeler ve 3B sahne ekran dışı bir tampona işlendikten sonra uygulanan tam ekran son efektler gibi çok çeşitli görsel efektleri destekler. Tam ekran son efektlerin bazı örnekleri şunlardır: yüksek dinamik aralık (HDR) ton eşleme ve parıldama; tam ekran kenar yumuşatma (Full Screen Anti-aliasing) ve ağartma baypası, doygunluk ve doygunluk giderme efektleri vb. dahil olmak üzere renk düzeltme ve renk kaydırma efektleri. Oyun morları ayrıca, *bellek yönetimi* (memory menagement), *matematik kütüphanesi* (math library), *özel veri yapıları ve algoritmalar* (custom data structures and algorithms) gibi çekirdek sistemlere, *kaynak yöneticisi* gibi katmanlara ve *görüntü işleme motoru* (rendering engine) gibi bileşenlere sahiptir (s. 45).

2.1.1 Görüntü İşleme Motoru (Rendering Engine)

Görüntü işleme motoru, herhangi bir oyun motorunun en büyük ve en karmaşık bileşenlerinden biridir ve birçok farklı şekilde tasarlanabilir. Çoğu modern görüntü işleme motoru, büyük ölçüde bağlı oldukları 3D grafik donanımının tasarımından kaynaklanan bazı temel tasarım felsefelerini paylaşmaktadır. Görüntü işleme motoru tasarımına yönelik yaygın ve etkili bir yaklaşım, katmanlı bir mimari kullanmaktır. *Düşük seviyeli görüntü işleme motoru* (low-level renderer), motorun tüm ham görüntü işleme olanaklarını kapsar. Bu seviyede tasarım, bir sahnenin hangi bölümlerinin görünür olabileceğine fazla aldırmadan, geometrik ilkellerin bir koleksiyonunu mümkün olduğunca hızlı ve zengin bir şekilde oluşturmaya odaklanır. Bu bileşen, aşağıda ele alınan çeşitli alt bileşenlere ayrılmıştır. Düşük seviyeli görüntü işleme motorunun Malzemeler ve Gölgelendiriciler, Statik ve Dinamik Aydınlatma, kameralar, Metin ve Yazı Tipleri, ilkel teslim (primitive submission), Görünüm Alanları ve Sanal Ekranlar alt bileşenleri, Doku ve Yüzey Yönetimi, Hata Ayıklama Çizimi (Çizgiler vb.), Grafik Aygıt Arayüzü gibi alt bileşenleri bulunur.

Farklı oyun motorları farklı türde içerik oluşturma ve üretim iş akışları için optimize edilmiştir, ancak günümüz oyun motorlarının birçoğu bir dizi farklı kültürel yazılım, programlama dili, üretim iş akışı ve ara yazılım ve eklenti özellikleriyle birlikte çalışabilir. Unity bu birlikte çalışabilirliğin paradigmatik bir örneğidir. Örneğin bir sanatçı, Blender veya Maya gibi bir yazılım aracını kullanarak 2B ve 3B modeller, görsel efektler veya dokular yaratabilir ve ardından yarattıklarını Unity'ye aktarabilir (Nicoll & Keogh, 2019, s. 10)

2.1.2. Görüntü İşleme İş akışı (Rendering Pipeline)

İş akışı (pipeline), her biri oyun motoruna 'beslenen' ve motorun temel kod çerçevesi aracılığıyla birlikte çalışabilir hale getirilen tasarım konseptleri, varlıklar, animasyonlar, komut dosyaları, ses ve daha fazlasının üretim zincirine verilen addır. Günümüzün oyun motorları aynı zamanda hızlı yinelemeyi teşvik etme eğilimindedir, yani geliştiricilerin oluşturma, test etme ve düzenleme modları arasında sorunsuz bir şekilde geçiş yaparak içeriği prototip haline getirmelerini (ve zaten var olan içeriği yinelemelerini) mümkün kılar. Oyun motorları çeşitli tasarım metodolojilerini ve proje türlerini kolaylaştırmak için özelleştirilebilen modüler araç setleri olma eğilimindedir.

Görüntü işleme motoru (rendering engine) karelerin (frame) işlenmesinden sorumludur ve örgü (mesh) verilerini yeni karelere dönüştürmek için kullandığı adımlar dizisine *görüntü işleme iş akışı* (rendering pipeline) ya da *grafik iş akışı* (graphics pipeline) denir. Bir oyun motoru (game engine) olan Unity'nin resmî web sayfası (Unity Documentation) grafik (graphics) linki altında Unity'nin 2022.3 versiyonu için bu adımları söyle sıralar: 1) Hangi nesnelerin görüntüleneceğine karar verilen ayıklama işlemi (culling). Bu genellikle kamera görüntüsünün dışında kalan (frustum culling) veya diğer nesnelerin arkasına gizlenen (occlusion culling) nesnelerin kaldırılması anlamına gelir. 2) Nesnelerin doğru ışıklandırmayla piksel tamponlarına çizilme işlemi. 3) Ekran için son çıktı karesini oluşturmak üzere piksel tamponlarının değiştirildiği işlem. Değişikliklere örnek olarak renk tonlaması, parıldama (bloom) ve alan derinliği

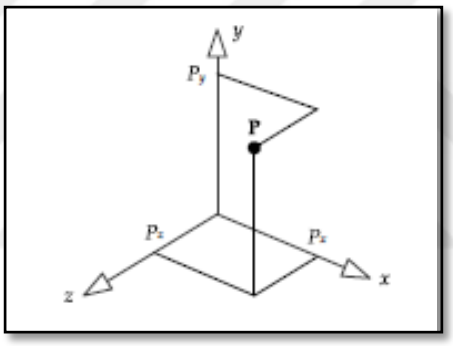
verilebilir. Oyun motoru her yeni kare oluşturduğunda bu adımlar tekrarlanır (Documentation, 2024)

Web sayfası, proje geliştirme sürecinin başında doğru Unity görüntü işleme iş akışını seçmenin önemli olduğunu vurgulayarak kullanım amaçlarına bağlı olarak seçenekleri söyle sıralar: 1) Yerleşik Görüntü İşleme İş Akışı (Built-In Render Pipeline); 2) Evrensel Görüntü İşleme İş Akışı (Universal Render Pipeline, URP); 3) Yüksek Çözünürlüklü Görüntü İşleme İş Akışı (High Definition Render Pipeline, HDRP). Tüm platformlarda görüntü işleme ölçeklenebilirliğine ihtiyaç duyan projeler için Yerleşik Görüntü İşleme İş Akışı; Tüm platformlarda, özellikle de döşeme tabanlı ertelenmiş işleme platformlarında ve bağlı olmayan VR platformlarında görüntü işleme ölçeklenebilirliğine ihtiyaç duyan projeler için Evrensel Görüntü İşleme İş Akışı ve üst düzey platformlarda fotogerçekçilik ve yüksek doğrulukta görüntü işleme gerektiren projeler için Yüksek Çözünürlüklü Görüntü İşleme İş Akışı kullanılmalıdır. Web sayfası ayrıca bu çeşitli görüntü işleme boru hatlarının kullandıkları platformlar, karşılaştırmalı özellikleri, özel bir görüntü işleme hattı oluşturma, ayrıca kameralar, ışıklar, modeller, ağlar (meshes), dokular, gölgelendiriciler, malzemeler, görsel etkiler, gökyüzü, renk, grafik uygulama arayüz desteği, grafik performansı ve profil oluşturma, dünya inşası (world building), ses, animasyon vb. Unity oyun motorunun özelliklerine ilişkin bilgi içerir.

Bir Unity sahnesi, Oyun Nesnelerini (GameObjects) üç boyutlu bir alanda temsil eder. İzleyicinin ekranı iki boyutlu olduğundan, Unity'nin bir görünüm yakalaması ve görüntülemek için onu "düzleştirmesi" gerekir. Bunu kameralar kullanarak yapar. Bir kameranın gördüğü şey, dönüşüm ve kamera bileşeni tarafından tanımlanır. Dönüşüm konumu bakış açısını tanımlar, ileri (Z) ekseni görünüm yönünü tanımlar ve yukarı (Y) ekseni ekranın üst kısmını tanımlar. Kamera bileşenindeki ayarlar, görünümün içine düşen bölgenin boyutunu ve şeklini tanımlar. Bu parametreler ayarlandığında, kamera o anda "gördüğü" şeyi ekranda görüntüleyebilir. GameObject hareket ettikçe ve döndükçe, görüntülenen görünüm de buna uygun olarak hareket eder ve döner.

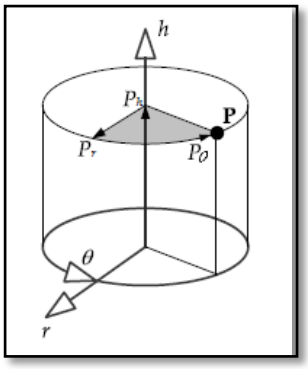
Modern 3B oyunların çoğu sanal bir dünyadaki üç boyutlu nesnelerden oluşur ve bu nesneler neredeyse her zaman köşeleri noktalarla temsil edilen üçgenlerden meydana gelir. Bir nokta, 2 ya da 3 boyutlu uzamda bir konumu temsil eder. Bir oyun motorunun tüm bu nesnelerin konumlarını, yönlerini ve ölçeklerini takip etmesi, onları oyun dünyasında canlandırması ve ekranda görüntülenebilmeleri için ekran alanına dönüştürmesi gerekir. (Gregory, 2019, s. 360)

Kartezyen koordinat sistemi, oyun programcıları tarafından kullanılan en yaygın koordinat sistemidir. İki veya üç boyutlu uzayda bir konumu belirtmek için birbirine dik iki veya üç eksen kullanır. Yani, bir P noktası ikili veya üçlü gerçek sayı ile temsil edilir, (Px, Py) veya (Px, Py, Pz)

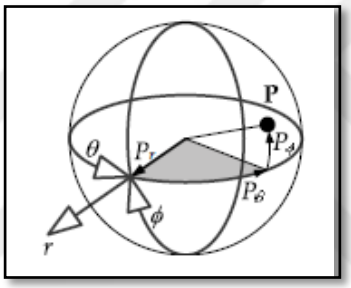


Şekil 19. Kartezyen koordinat sistemi örneği

Kartezyen koordinat sisteminin dışında dikey bir "yükseklik" ekseni h, dikeyden çıkan bir radyal eksen r ve bir sapma açısı teta (q) kullanan Silindirik ve bir yunuslama açısı phi (ϕ), bir sapma açısı theta (q) ve bir radyal ölçüm r kullanan Küresel koordinat sistemleri de kullanılır.



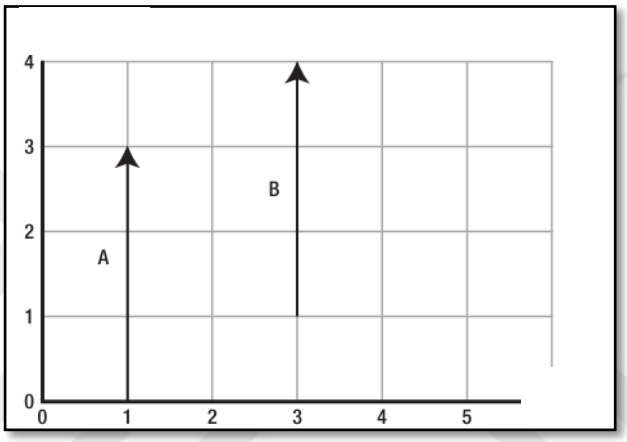
Şekil 20. Silindirik koordinat sistemi



Şekil 21. Küresel koordinat sistemi

Vektör, n boyutlu uzayda hem büyüklüğü hem de yönü olan bir niceliktir. Bir vektör, kuyruk adı verilen bir noktadan baş adı verilen bir noktaya uzanan yönlendirilmiş bir doğru parçası olarak görselleştirilebilir. Bunu, bir büyüklüğü temsil eden ancak yönü olmayan bir skaler (yani sıradan bir gerçek değerli sayı) ile karşılaştırdığımızda genellikle skaler sayılar italik yazılırken (örn. *v*), vektörler kalın harflerle yazılır (örn. v). Bir 3B vektör, aynen bir nokta gibi üçlü skaler (x, y, z) ile temsil edilebilir. Noktalar ve vektörler arasındaki ayrım aslında oldukça hassastır. Teknik olarak, bir vektör bilinen bir noktaya göre sadece bir ofsettir. Bir vektör 3B uzayda herhangi bir yere taşınabilir -büyüklüğü ve

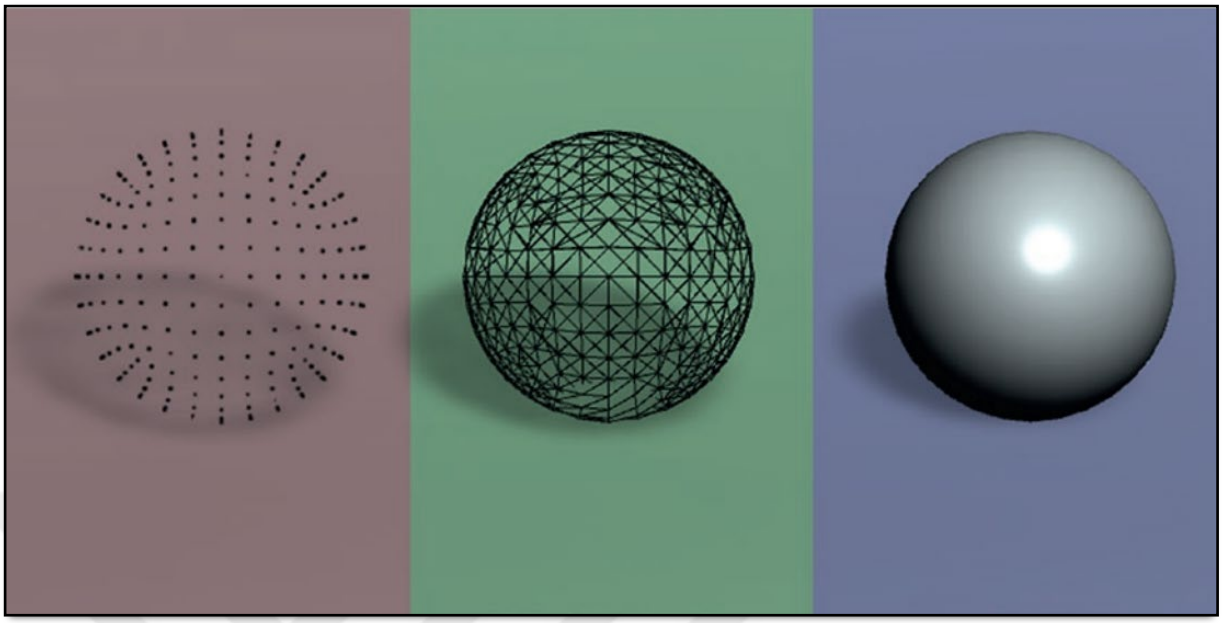
yönü değişmediği sürece aynı vektördür. Bir vektör, kuyruğu koordinat sisteminin orijinine sabitlenmesi koşuluyla, bir noktayı temsil etmek için kullanılabilir. Oyun programcılarının büyük çoğunluğu "vektör" terimini hem noktaları (konum vektörleri) hem de vektörleri ifade etmek için kullanır. Vektörler bir yolculuktaki yönler gibi düşünülebilir. Bir sonraki benzin istasyonuna ulaşmak için 3 mil kuzeye gitmek gerekiyorsa, bu "kuzey" yönü ve 3 mil büyüklüğü veya uzunluğu olan bir vektör olarak düşünülebilir. (Kyle Halladay, 2019, s. 4)



Şekil 22. Aynı vektörün iki farklı konum örneği

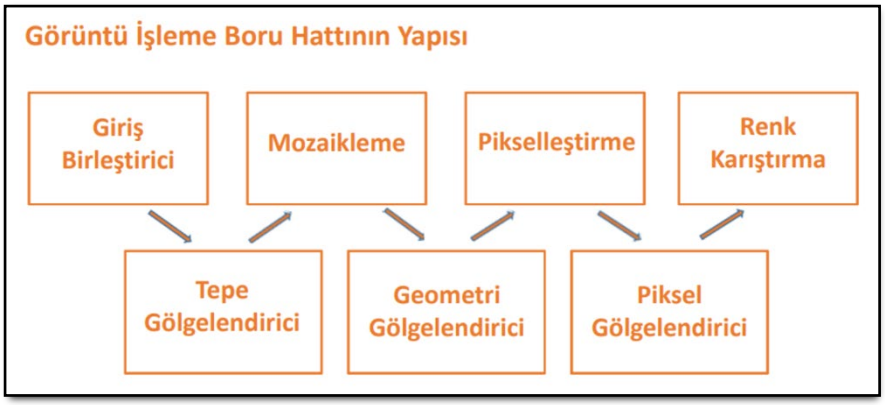
Bilgisayar grafiklerinde görüntü işleme, bir 3B oyun veya uygulamada 3B gölgelendiriciyi tanımlamak için kullanılan tepe noktası (vertex) konumlarını içeren bir dizi 2B veya 3B örgüden (mesh) ve ışıkların konumu veya oyun kamerasının yönü gibi bir oyun sahnesiyle ilgili bilgilerden görüntü oluşturma işlemidir. Bu işlem tarafından oluşturulan görüntü render veya frame (kare) olarak adlandırılır ve bu görüntü kullanıcıya bilgisayar ekranında saniyede birçok kez sunulur. Aslında, çoğu oyun saniyede 30 ila 60 kez yeni bir kare oluşturur (s. 1).

Örgü, şekilleri bir bilgisayar için anlamlı olacak şekilde tanımlamak için sahip olduğumuz yöntemlerden biridir. Bir şekli tanımlamak için bir örgünün üç şey hakkında bilgi depolaması gerekir: *köşeler* (vertices), *kenarlar* (edges) ve *yüzler* (faces).



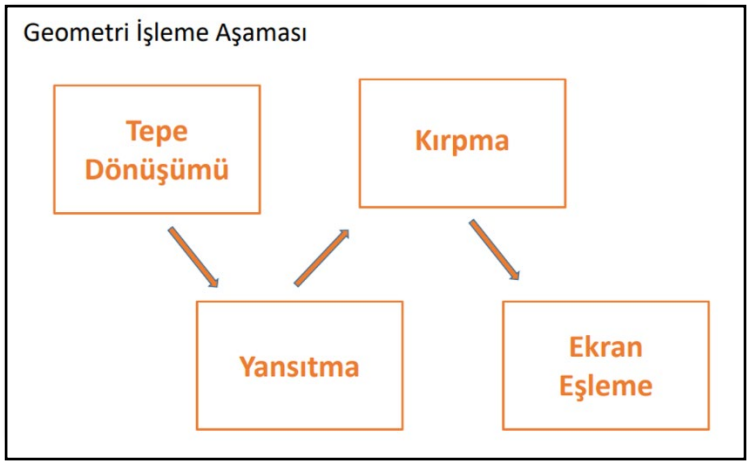
Şekil 23. Bir örgünün farklı bileşenleri – sol: köşeler, orta: kenarlar, sağ: yüzeyler görebiliriz

Yalnızca bir örgünün köşeleri bellekte saklanır ve kenarları ve yüzleri, köşelerin bulunduğu sıraya göre dolaylı olarak tanımlanır. Bazen bu köşe sıralaması basitçe köşelerin örgünün şekil verilerinde saklanma sırasıdır, bazen de *dizin tamponu* (index buffer) adı verilen bir veri yapısı tarafından tanımlanır. Unity oyun motoru, görüntü işleme İş Akışının (rendering pipeline) temel yapısını 1) gerçek zamanlı görüntü işleme motorları için bir görüntü işleme iş akışının temel modeline karşılık gelen uygulama, 2) geometri işleme, 3) rasterleştirme, 4) piksel işleme şeklinde dört aşamaya ayırır (Espíndola & Yeber, 2023). Bu aşamalar gerçek zamanlı görüntü işleme motorları için görüntü işleme iş akışının temel modeline karşılık gelir.



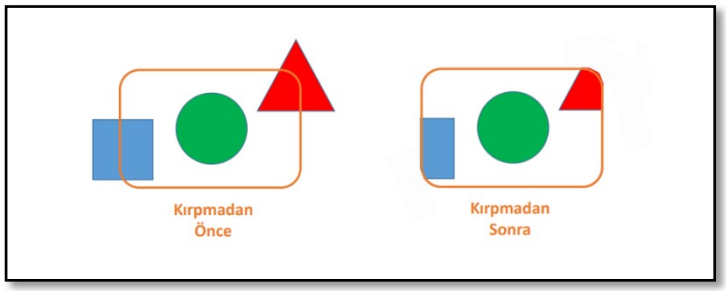
Şekil 24. Mantıksal Görüntü İşleme İş Akışı

Uygulama aşaması CPU'da başlar ve bir sahne içinde gerçekleşen çarpışma algılama, doku animasyonu, klavye ve fare girişi gibi çeşitli işlemlerden sorumludur, Uygulama aşamasının işlevi, daha sonra ilkelleri (örneğin üçgenler, çizgiler, köşeler) oluşturmak için bellekte depolanan verileri okumaktır. Bu aşamanın sonunda, tüm bu bilgiler matris çarpımı yoluyla köşelerin dönüşümünü oluşturmak için geometri işleme aşamasına gönderilir. CPU, bilgisayar ekranımızda gördüğümüz görüntüleri GPU'dan talep eder. Bu istekler iki ana adımda gerçekleştirilir: Önce geometri işlemeden piksel işlemeye kadar olan aşamalar kümesine karşılık gelen görüntü işleme durumu yapılandırılır ve sonra, nesne ekrana çizilir. Geometri işleme aşaması GPU'da gerçekleşir ve nesnenin vertex işleminden sorumludur. Bu aşama tepe gölgelendirme, projeksiyon, kırpma ve ekran eşleme olmak üzere dört alt sürece ayrılır.



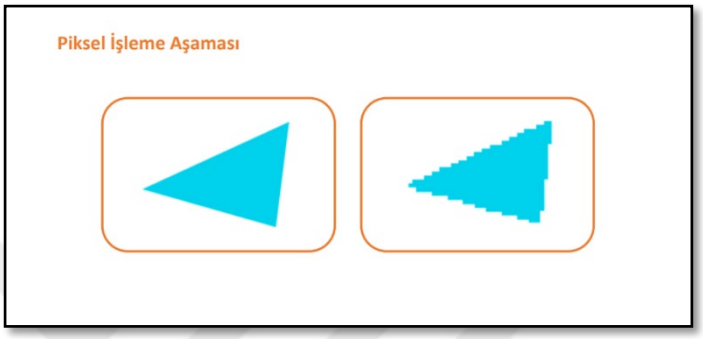
Şekil 25. Geometri işleme aşaması

İlkel öğeler uygulama aşamasında bir araya getirildiğinde, daha yaygın olarak tepe noktası gölgelendirici aşaması (vertex shader stage) iki ana görevi yerine getirir: 1) Nesnenin köşelerinin (vertex) konumunu hesaplar, 2) Nesnenin konumunu bilgisayar ekranına yansıtılabilmesi için farklı uzay koordinatlarına dönüştürür (s. 21). İşte bu aşamada, normaller (Bir çokgenin yüzeyinde, bir yüzün veya tepe noktasının yönünü veya yönelimini belirlemek için kullanılan dik vektörler), teğetler (yatay doku yönü boyunca ağ yüzeyini takip eden bir birim uzunluğunda vektörler), UV koordinatları (nesnenin ağının düzleştirilmiş, iki boyutlu bir temsili) vb. sonraki aşamalara aktarmak istediğimiz özellikleri seçebiliriz. Projeksiyon ve kırpma, sahnedeki kameranın özelliklerine göre değişen sürecin bir parçası olarak ve tüm görüntü işleme işlemi de yalnızca kameranın görüş alanı içindeki öğeler için gerçekleşir.



Şekil 26. Kırpma aşaması

Belleğe alınan kırpılmış nesneler daha sonra ekran haritasına gönderilir (ekran eşleme). Bu aşamada, sahnede bulunan üç boyutlu nesneler, pencere koordinatları olarak da bilinen ekran koordinatlarına dönüştürülür. Pikselleştirme (rasterization) bu 2B ekran koordinatlarına sahip nesnelerin ekranda işgal edecekleri pikselleri bulma aşamasıdır. Bu işlem, sahnedeki nesneler ile ekrandaki pikseller arasında bir senkronizasyon adımıdır.

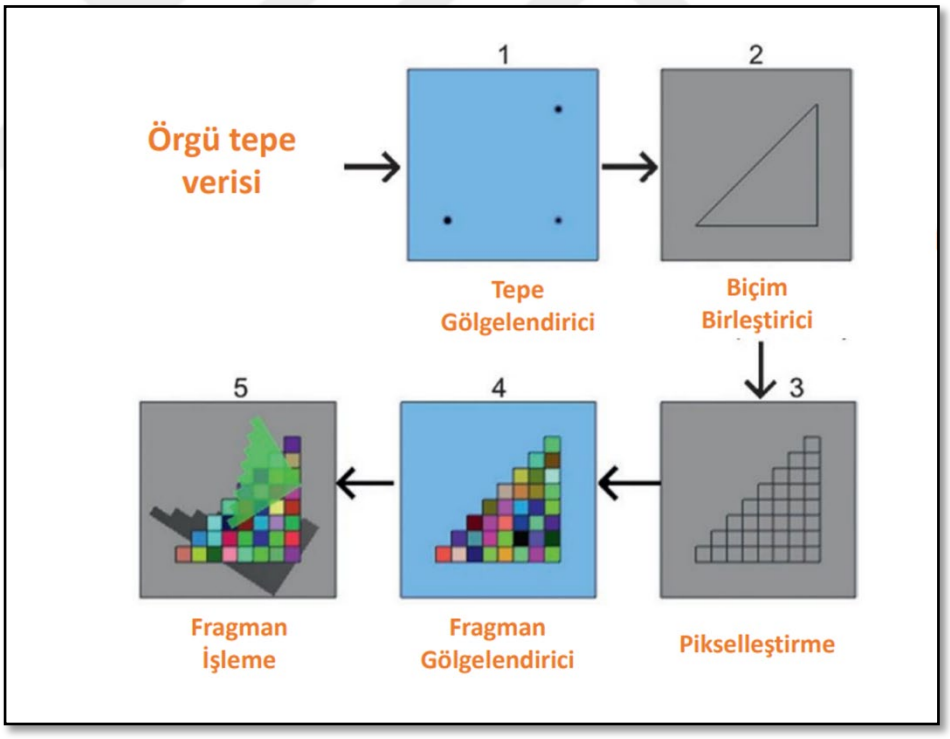


Şekil 27. Bir geometrik biçimin kapladığı alan ekranda piksellere dönüştürülür.

Bu aşamada her nesne için iki işlem gerçekleşir: 1) Üçgen kurulumu, 2) Üçgen geçişi. Üçgen kurulumu, üçgen geçişine gönderilecek verilerin oluşturulmasından sorumludur ve ekrandaki bir nesnenin kenarları için denklemleri içerir. Üçgen geçişi, çokgen nesnesinin alanı tarafından kapsanan pikselleri listeler. Bu şekilde "fragment" adı verilen bir piksel grubu oluşturulur. Ancak, bu kelime çoğu kez tek bir piksele atıfta bulunmak için kullanılır. Önceki işlemlerden gelen enterpolasyonlu değerleri kullanan bu son aşama, tüm pikseller ekrana yansıtılmaya hazır olduğunda başlar. Bu noktada, piksel gölgelendirici aşaması olarak da bilinen fragment shader aşaması başlar ve her pikselin görünürlüğünden sorumludur. Temel olarak yaptığı şey, bir pikselin son rengini hesaplamak ve ardından bunu renk tamponuna göndermektir.

Görüntü işleme iş akışı ya da grafik işleme sürecini özetlemek gerekirse, Vertex gölgelendiricisi bir örgüdeki (mesh) tüm köşeleri işlemeyi bitirdikten sonra, tüm bu verileri işlem hattının "şekil birleştirme" (shape assembly) adı verilen bir sonraki adımına gönderir (Kyle Halladay, 2019, s. 13). Bu aşama, tüm köşeleri çizgilerle birbirine bağlamaktan, esasen, örgünün kenarlarını ekrana getirmekten sorumludur. Basitleştirilmiş işleme hattındaki bir sonraki aşama pikselleştirmedir (rasterization). Bu

aşamada GPU, örgünün ekranda hangi pikselleri kaplayabileceğini belirler ve bu potansiyel piksellerin her birinin ekranda çizilmesi için gereken tüm bilgileri içeren bir fragman oluşturur. Her bir fragman potansiyel bir piksel olarak düşünülebilir, ancak tüm fragmanlar ekranda piksel olarak sonuçlanmayacaktır. Görüntü işleme iş akışının "Fragman işleme" aşamasının iki önemli rolü vardır: fragman testi ve harmanlama (blending) işlemleri. Fragman testi, ekrana hangi parçaların yerleştirileceğine ve hangilerinin atılacağına karar verir. Önceki işlemlerden gelen enterpolasyonlu değerleri kullanan bu son aşama, tüm pikseller ekrana yansıtılmaya hazır olduğunda başlar (Espíndola & Yeber, 2023, s. 23). Bu noktada, piksel gölgelendirici aşaması olarak da bilinen fragment shader aşaması başlar ve her pikselin görünürlüğünden sorumludur. Temel olarak yaptığı şey, bir pikselin son rengini hesaplamak ve ardından bunu renk tamponuna göndermektir.



Şekil 28. Grafik işlem hattının basitleştirilmiş bir görünümü. Adımların sırası hem oklarla hem de her kutunun üzerinde bulunan sayılarla gösterilmektedir. (Kyle Halladay, 2019)

2.2. Gölgelendiriciler (Shaders)

Kullanılan türe bağlı olarak her görüntü işleme iş akışının kendine has özellikleri vardır. Malzeme özellikleri, ışık kaynakları, dokular ve gölgelendirici içinde dâhili olarak gerçekleşen tüm işlevler, ekrandaki nesnelerin görünümünü ve optimizasyonunu etkileyecektir (Espíndola & Yeber, 2023). Grafik kartında çalıştırılan gölgelendiriciler işlem yaptıkları aşamanın adını alırlar ve CPU tarafından çalıştırılan çoğu normal programın yapamadığı bazı şeyleri yapabilirler -örneğin görüntü işleme iş akışının bazı önemli kısımlarını kontrol edebilirler. Bu anlamda gölgelendiriciler, grafik kartının, örgü verilerini ekranda bir görüntüye dönüştürmek için attığı, görüntü işleme iş akışı (rendering pipeline) olarak adlandırılan bir dizi adıma -örneğin .fbx uzantılı bir poligon nesnenin bilgisayar ekranına işlenmesi için geçmesi gereken sürece - müdahale ederek ilginç efektler elde etmek için kullanılan "shader" uzantılı (örn. color.shader) küçük programlardır. (Halladay, 2019, s. 12; Espíndola & Yeber, 2023, s. 19). Gölgelerndiricilerde, bilgisayar ekranındaki bir nesneyi kapsayan alandaki her piksel için renk işlemeye izin veren matematiksel hesaplamalar ve talimat listeleri (komutlar) vardır (Espíndola & Yeber, 2023, s. 33). Gölgelendirici programları, poligonal bir nesnenin özelliklerine dayalı olarak (koordinat sistemlerini kullanarak) öğeler çizmemizi sağlar. Gölgelendiriciler, görevleri aynı anda çözmek için tasarlanmış binlerce küçük, verimli çekirdekten oluşan paralel bir mimariye sahip oldukları için GPU tarafından yürütülürken, CPU sıralı seri işleme için tasarlanmıştır.

Unity oyun motorunda gölgelendiricilerle ilişkili üç tür dosya vardır: 1) farklı görüntü işleme iş akışı türlerinde derlenebilen ".shader" uzantılı programlar; 2) Yalnızca evrensel görüntü işleme iş akışı veya yüksek çözünürlüklü görüntü işleme iş akışında derlenebilen ".shadergraph" uzantılı programlar: 3) Genellikle Shader Graph'ta bulunan Özel İşlev adlı bir düğüm türünde kullanılan ve özelleştirilmiş işlevler oluşturmamızı sağlayan ".hlsl" uzantılı dosyalar. Ayrıca Unity'de, gölgelendiriciler oluşturmak için tanımlanmış en az dört tür yapı vardır; bunlar arasında vertex shader ve fragment shader kombinasyonunu, ardından otomatik aydınlatma hesaplaması için surface shader'ı ve daha gelişmiş kavramlar için compute shader'ı bulabiliriz. Bu yapıların her biri, derleme

işlemini kolaylaştıran önceden tanımlanmış özelliklere ve işlevlere sahiptir; yazılım bu yapıları otomatik olarak eklediğinden, işlemler de kolayca tanımlanabilir.

Unity'de gölgelendirici geliştirme ile ilişkili üç programlama dili vardır: 1) HLSL (High-Level Shader Language - Microsoft), 2) Cg (C for Graphics - NVIDIA) gölgelendiriciye derlenir ancak artık yazılımın mevcut sürümlerinde kullanılmaz, 3) ShaderLab (bildirimsel dil - Unity) (s. 34). “Cg”, çoğu GPU'da derlenmek üzere tasarlanmış yüksek seviyeli bir programlama dilidir. NVIDIA tarafından Microsoft ile iş birliği içinde geliştirilmiştir ve HLSL'ye çok benzer bir sözdizimi kullanır. Gölgelendiricilerin Cg dili ile çalışmasının nedeni hem HLSL hem de GLSL'yi (OpenGL Gölgelendirme Dili) derleyebilmeleri, video oyunları için materyal oluşturma sürecini hızlandırmaları ve optimize etmeleridir.

Bir gölgelendirici oluşturulduğunda, kod CGPROGRAM adlı bir alanda derlenir. 2019'dan günümüze güncel versiyonlarının resmi gölgelendirici programlama dili HLSL olan Unity, şu anda Cg ve HLSL arasında daha fazla destek ve uyumluluk sağlamak için çalışmaktadır ve dolaysıyla bu blokların yerini HLSLPROGRAM ve ENDHLSL'nin alması oldukça muhtemeldir (s. 35). Shader Graph ve Compute hariç Unity'deki tüm gölgelendiriciler ShaderLab adı verilen bildirimsel (declerative) bir dille yazılır. Bu dilin sözdizimi, Unity denetçisinde gölgelendiricinin özelliklerinin görüntülemesini sağlar ve bu da değişkenlerin ve vektörlerin değerlerinin gerçek zamanlı olarak değiştirilerek gölgelendiricinin istenen sonucu elde edecek şekilde ayarlanmasını mümkün kılar.

2.2.1. Gölgelendiriclerin Temel Yapısı ve Özellikleri

Gölgelendiriciler, denetçide bir yol (InspectorPath) ve bir adla (shaderName) başlar, ardından özellikler (ör. dokular, vektörler, renkler vb.) daha sonra ShaderLab'de komutların bildirilmesine ve geçişler oluşturulmasına olanak tanıyan “SubShader” ve hepsinin sonunda isteğe bağlı olan ve hata oluşturan bir gölgelendirici yerine farklı bir gölgelendirici derleyerek grafik donanımın çalışmasına devam etmesini sağlayan “Fallback” bulunur (Espíndola & Yeber, 2023, s. 41). Bir gölgelendirici doğrudan poligonal bir nesneye uygulanamaz, önceden oluşturulmuş bir malzeme aracılığıyla

yapılması gerekir. "InspectorPath", gölgelendiriciyi bir malzemeye uygulamak için seçilecek yeri ifade eder ve bu seçim Unity Inspector aracılığıyla yapılır. Kodla yazılan gölgelendiricilerin çoğu, “Gölgelendirici”nin bildirimi, ardından Unity “Inspector”daki yolu ve son olarak ona atanan “adla” başlar (örneğin, Gölgelendirici "shader inspector path/shader name"). SubShader ve Fallback gibi özelliklerin her ikisi de ShaderLab bildirim dilinde "Gölgelendirici" alanının içine yazılır (s. 43).

Bir Gölgelendirici, 1) ışık kaynaklarının açılarını, görüş açısını ve diğer ilgili hesaplamaları içerebilecek kod ve matematiksel hesaplamaları içeren bir nesneyi oluşturma yöntemini 2) Doku haritaları, renkler ve sayısal değerler gibi malzeme denetçisinde özelleştirilebilen parametreleri tanımlar. Gölgelendiriciler, son kullanıcının grafik donanımına bağlı olarak farklı yöntemler de belirleyebilir. Bir Malzeme ise malzemeyi işlemek için hangi gölgelendiricinin kullanılacağını gölgelendiricinin parametreleri için hangi doku haritalarının, renk ve sayısal değerlerin kullanılacağı gibi belirli değerleri tanımlar.

Özel Gölgelendiriciler grafik programcıları tarafından yazılmak üzere tasarlanmıştır. Oldukça basit olan ShaderLab dili kullanılarak oluşturulurlar. Bununla birlikte, bir gölgelendiricinin çeşitli grafik kartlarında iyi çalışmasını sağlamak karmaşık bir iştir ve grafik kartlarının nasıl çalıştığı hakkında oldukça kapsamlı bir bilgi gerektirir. Bir Materyalin denetçisinin görüntülediği özellikler, Materyalin kullandığı Gölgelendirici tarafından belirlenir. Bir Gölgelendirici, ekranda işlenen GameObject'in piksellerini oluşturmak için doku ve aydınlatma bilgilerinin nasıl birleştirileceğini belirleyen özel bir grafik programı türüdür.

ShaderLab, Unity'nin iş akışı ayarlarını, gölgelendirici kodunu ve bir gölgelendiricinin beklediği üniformalar hakkındaki bilgileri kapsayan özel veri formatıdır (Kyle Halladay, 2019, s. 300). ShaderLab programının özellikleri, Unity denetçisinden manipüle edilebilen parametrelerin bir listesine karşılık gelir. Dinamik olarak veya çalışma zamanında oluşturulmak veya değiştirilmek istenen gölgelendiriciyle ilgili olarak kullanılan doku, rakam, kaydırıcı (slider), renk, vektör, maintex, malzeme özelliği

çekmecesi gibi hem değer hem de kullanışlılık açısından farklı özellikler mevcuttur (s. 44-50).

Tekrarlamak gerekirse, Unity'de gölgelendiriciler üç geniş kategoriye ayrılır. Grafik işlem hattının bir parçası olan gölgelendiriciler en yaygın gölgelendirici türüdür. Ekrandaki piksellerin rengini belirleyen hesaplamalar yaparlar. Unity'de, genellikle Shader (gölgelendirici) nesnelerini kullanarak bu tür gölgelendiricilerle çalışır. Compute Shaders (Hesaplama Gölgelendiricileri), normal grafik işlem hattının dışında GPU üzerinde hesaplamalar gerçekleştirir. Ray Tracing Shaders (Işın İzleme Gölgelendiricileri), ışın izleme ile ilgili hesaplamaları gerçekleştirir.

Güncel Unity web sayfası temel gölgelendirici kavramlarını şöyle tanımlar: shader veya shader programı, GPU üzerinde çalışan ve aksi belirtilmedikçe grafik işlem hattının bir parçası olan bir programdır; Shader nesnesi, aynı dosyada birden fazla gölgelendirici programı tanımlamaya ve Unity'ye bunları nasıl kullanılacağını söylemeye olanak tanır ve shader programları ve diğer bilgiler için bir sarmalayıcıdır; ShaderLab, gölgelendiriciler yazmak için Unity'ye özgü bir dil; Shader Graph, kod yazmadan gölgelendiriciler oluşturmaya yarayan bir araç; shader asset, bir gölgelendirici nesnesini tanımlayan .shader uzantılı bir dosya ve Shader Graph asset ise, bir Unity projesinde gölgelendirici nesnesi tanımlayan bir dosyadır.

2.2.2. Unity Gölgelendirici Tipleri ve Grafik Özellikleri

Unity’nin kullanılan versiyonuna bağlı olarak Yerleşik Görüntü İşleme İş Akışı (Built-in Rendering Pipeline) programında, Standart Yüzey Gölgelendirici (Standard Surface Shader), Işıksız Gölgelendirici (Unlit Shader), Görüntü Efekti Gölgelendiricisi (Image Effect Shader), Hesaplama Gölgelendiricisi (Compute Shader) ve Işın İzleme Gölgelendiricisi (Ray Tracing Shader) gibi gölgelendirici seçenekleri mevcuttur (s. 36). Proje Evrensel İş Akışı (Universal Rendering Pipeline) veya Yüksek Çözünürlüklü Görüntü İşleme İş Akışında (High Definition Rendering Pipeline) yaratılmışsa Gölgelendirici Grafiği (Shader Graph) paketiyle birlikte gelebilecek ilave gölgelendiriciler kullanılabilecek seçenek sayısını artırabilir.

Standart Yüzey Gölgelendirici (Standard Surface Shader) temel bir aydınlatma modeliyle etkileşime giren ve yalnızca Yerleşik Görüntü İşleme İş Akışında (Built-in RP) çalışan kod yazma optimizasyonu ile karakterize edilir. Işıkla etkileşime giren bir gölgelendirici oluşturmak için iki seçenek söz konusudur: 1) Işıksız bir Gölgelendirici kullanmak ve malzeme üzerinde aydınlatma oluşturmaya izin veren matematiksel işlevler eklemek. 2) veya bazı durumlarda albedo, specular ve diffuse içeren temel bir aydınlatma modeline sahip bir Standart Yüzey Gölgelendirici kullanmak (S.37). Hem Built-in hem de Scriptable görüntü işleme motorlarıda çalışan Işıksız Gölgelendirici (unlit sahder) birincil renk modelini ifade eder ve genellikle efektleri oluşturmak için kullanılacak temel yapıdır. Düşük seviye donanımlar (low-end hardware) için ideal olan bu program türünün kodunda herhangi bir optimizasyon olmadığı için tüm yapısı görülebilir ve ihtiyaca göre değiştirilebilir. "OnRenderImage" (C #) işlevini gerektiren Görüntü Efekti Gölgelendiricisi (image effect sahader) yapısal olarak Işıksız Gölgelendiriciye çok benzer ve esas olarak Yerleşik Görüntü İşleme İş Akışı (Built-in RP) programında işlem sonrası efektlerde kullanılır. Yaygın gölgelendiricilerin aksine, uzantısı ".compute" ve programlama dili HLSL olan ve normal görüntü işleme iş akışının dışında, grafik kartında çalışan Hesaplama Gölgelendiricisi (compute shader) bu özellikleri ile yapısal olarak yukarıda bahsedilen gölgelendiricilerden çok farklıdır. Son olarak ".raytrace" uzantılı Işın İzleme Gölgelendiricisi (Ray Tracing Shader), yalnızca Yüksek Çözünürlüklü Görüntü İşleme İş Akışında (high resolution render pipeline) çalışır ve bazı teknik sınırlamaları vardır.

Unity oyun motorunun resmî web sayfası görüntü işleminin temelde örgüler (Meshes), Malzemeler (Materials), Gölgelendiriciler (Shaders) ve Dokuların (Textures) yakın bir ilişki içerisinde kullanılmalarıyla gerçekleştiğini belirtir. Modeller, karakterler, arazi veya ortam nesneleri gibi 3B nesnelerin şekli ve görünümü hakkında veriler içeren dosyalardır. Model dosyaları, örgüler, malzemeler ve dokular dahil olmak üzere çeşitli veriler içerebilir. Ayrıca animasyonlu karakterler için animasyon verileri de içerebilirler. Modeller genellikle harici bir uygulamada oluşturulur ve ardından Unity'ye aktarılır. Örgüler Unity'nin ana grafik primitifleridir. Bir nesnenin şeklini tanımlarlar. Malzemeler, kullanılan Dokulara referanslar, döşeme bilgileri, Renk tonları ve daha fazlasını dahil

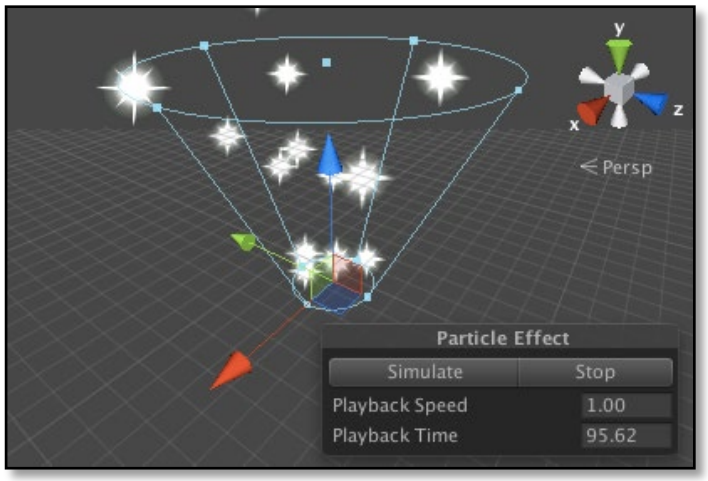
ederek bir yüzeyin nasıl işlenmesi gerektiğini tanımlar. Bir Malzeme için mevcut seçenekler, Malzemenin hangi Gölgelendiriciyi kullandığına bağlıdır. Gölgelendiriciler, aydınlatma girdisine ve Malzeme yapılandırmasına dayalı olarak işlenen her pikselin Rengini hesaplamak için matematiksel hesaplamaları ve algoritmaları içeren küçük komut dosyalarıdır. Dokular bitmap görüntülerdir. Bir Materyal dokulara referanslar içerebilir, böylece Materyalin Shader'ı bir GameObject'in yüzey rengini hesaplarken dokuları kullanabilir. Bir GameObject'in yüzeyinin temel Rengine (Albedo) ek olarak, Dokular bir Materyalin yüzeyinin yansıtıcılığı veya pürüzlülüğü gibi diğer birçok yönünü temsil edebilir.

Bir Materyal kullanılacak belirli bir Gölgelendiriciyi belirtir ve kullanılan Gölgelendirici Materyalde hangi seçeneklerin mevcut olduğunu belirler. Bir Gölgelendirici, kullanmayı beklediği bir veya daha fazla Doku değişkenini belirtir ve Unity'deki Malzeme Denetçisi, bu Doku değişkenlerine kullanıcının kendi Doku Varlıklarını atamasına olanak tanır. Normal görüntü işleme işlemlerinin çoğu için (karakterlerin, manzaraların, ortamların, katı ve saydam GameObject'lerin, sert ve yumuşak yüzeylerin işlenmesi edilmesi gibi) Standart Gölgelendirici genellikle en iyi seçimdir. Bu, birçok yüzey türünü son derece gerçekçi bir şekilde oluşturabilen, son derece özelleştirilebilir bir gölgelendiricidir. Farklı bir yerleşik Gölgelendiricinin (Built-in Sahader) veya hatta özel yazılmış bir gölgelendiricinin uygun olabileceği başka durumlar da vardır (örneğin sıvılar, yapraklar, kırılgan cam, parçacık efektleri, karikatürize, illüstratif veya diğer sanatsal efektler veya gece görüşü, ısı görüşü veya x-ışını görüşü gibi diğer özel efektler).

Standart Gölgelendirici bir malzeme parametreleri listesi sunar. Bu parametreler neredeyse tüm gerçek dünya yüzeylerinin görünümünü yeniden oluşturmak için kullanılabilir. Unity'de Standart Gölgelendiriciye ek olarak, özel amaçlar için bir dizi başka yerleşik gölgelendirici kategorisi de söz konusudur. Bunlar aydınlatma ve cam efektleri için FX; Kullanıcı arayüzü grafikleri için GUI ve UI; Mobil cihazlar için basitleştirilmiş yüksek performanslı gölgelendirici Mobil; Ağaçlar ve arazi için Nature; Parçacık sistemi efektleri için Particles; Tüm geometrilerin arkasındaki arka plan ortamlarını oluşturmak için Skybox; 2B sprite sistemi ile kullanım için Sprites - Işıksız:

Tüm ışık ve gölgelendirmeyi tamamen atlayan işlem için Unlit; Standart Gölgelendirici tarafından yerini alan eski gölgelendiricilerin geniş koleksiyonu Legacy'dir (Documentation, 2024).

2D oyunlarda Sprite'lar, nesnelerin şekillerine yaklaşan düz örgülere uygulanan dokular kullanılarak gerçekleştirilir. Bir oyunun grafik kullanıcı arayüzü (GUI), skor ekranı ve seçenekler menüsü gibi, doğrudan oyun sahnesinde kullanılmayan ancak oyuncunun seçim yapmasına ve bilgileri görmesine izin vermek için orada bulunan grafiklerden oluşur. Bu grafikler, bir ağ yüzeyini detaylandırmak için kullanılan türden açıkça farklı olmasına rağmen standart Unity dokuları kullanılarak işlenirler. Örgü katı nesneleri temsil etmek için idealdir, ancak alevler, duman ve bir büyünün bıraktığı parıltılar gibi şeyler için daha az uygundur. Bu tür efektler Parçacık Sistemleri tarafından çok daha iyi işlenir. Parçacık, duman bulutu gibi temelde akışkan veya gaz halinde olan bir şeyin küçük bir bölümünü temsil eden küçük bir 2B grafiktir. Bu parçacıkların birçoğu aynı anda oluşturulduğunda ve isteğe bağlı olarak rastgele varyasyonlarla harekete geçirildiğinde, çok ikna edici bir etki yaratabilirler. Örneğin, merkezi bir noktadan büyük bir hızla ateş dokusuna sahip parçacıklar göndererek bir patlama gösterilebilir. Bir şelale, su parçacıklarının sahnenin yüksek bir çizgisinden aşağıya doğru hızlandırılmasıyla simüle edilebilir.



Şekil 29. Yıldız parçacık sistemi.
https://docs.unity3d.com/2023.2/Documentation/Manual/Textures.html (2024)

Unity, uygulamanın görünümünü çok az kurulum süresiyle büyük ölçüde iyileştirebilecek bir dizi işlem sonrası efekt ve tam ekran efekti sağlar. Bu efektleri fiziksel kamera ve film özelliklerini simüle etmek veya stilize görseller oluşturmak için kullanılabilir. Hangi son işlem efektlerinin mevcut olduğu ve bunları nasıl uygulayacağınız, hangi render işlem hattını kullandığınıza bağlıdır. Bir görüntü işleme hattındaki post-processing çözümü diğer bru hatlarıyla hatlarıyla uyumlu değildir.

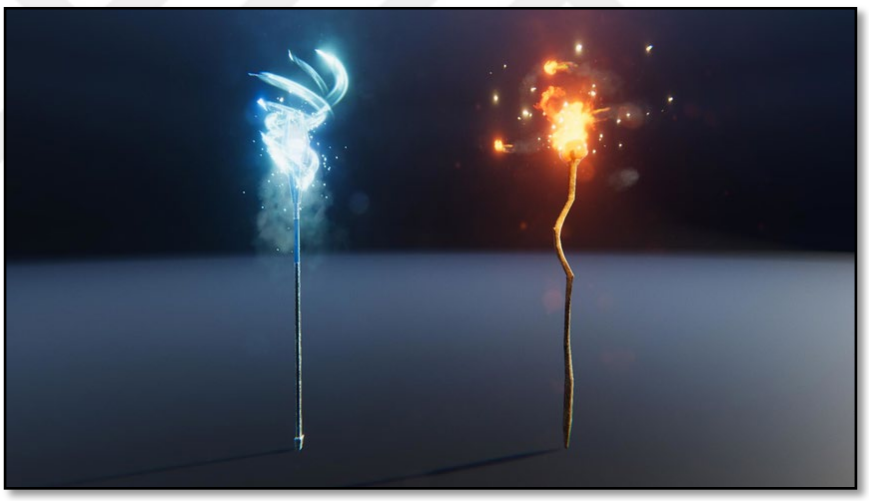
Şekil 30. Post-processing yapılmamış sahne.
https://docs.unity3d.com/2023.2/Documentation/Manual/PostProcessingOverview.html (2024)

Şekil 31. Post-processing ile sahne Post-processing ile sahne. https://docs.unity3d.com/2023.2/Documentation/Manual/PostProcessingOverview.html (2024)

Görsel etkiler Unity'nin son versiyonlarında performans, görünüm ve yapılandırma açısından paketler arasında farklılık gösterir. Son versiyonun listelediği görsel etkiler şunlardır: Ambient Occlusion (sahnedeki ortam aydınlatmasına maruz kalmayan alanları koyulaştırır), Anti-aliasing (sahnedeki kenarların görünümünü yumuşatır), Auto Exposure (Bir görüntünün pozlamasını orta tonuna uyacak şekilde dinamik olarak ayarlar), Bloom (görüntüdeki parlak alanların parlamasını sağlar), Channel Mixer (her bir giriş renginin dengesini ayarlamanızı sağlar), Chromatic Aberration (renkleri görüntünün koyu ve açık alanları arasındaki sınırlar boyunca dağıtır), Color Adjustments (son işlenen görüntünün genel tonunu, parlaklığını ve kontrastını değiştirmeyi sağlar), Color Curves (renk tonu, doygunluk veya parlaklıkta belirli aralıkları ayarlanmasını sağlar), Fog (Dış ortamlarda sis veya buğu görünümünü simüle eder), Depth of Field (ön plandaki nesneler odakta kalırken görüntünün arka planını bulanıklaştırır), Grain (görüntünüzün üzerine film gürültüsü bindirir), Lens Distortion (gerçek dünyadaki bir kamera lensinin şeklinden kaynaklanan bozulmayı simüle eder), Lift-Gamma-Gain (üç yönlü renk tonlaması yapmanıza olanak tanır), Motion Blur (görüntüyü kameranın hareketi yönünde bulanıklaştırır), Panini Projection (geniş bir görüş alanının neden olduğu görüntünün kenarındaki bozulmayı düzeltir), Screen Space Reflection (ıslak zemin yüzeylerini veya su birikintilerini simüle eden ince yansımalar

oluşturur), Shadows Midtones Highlights (görüntüdeki gölgelerin, orta tonların ve vurguların renk tonunu ve parlaklığını ayrı ayrı kontrol eder), Split Toning (görüntüdeki iki farklı tonu iki belirli renkle eşleştirir), Tonemapping (bir görüntünün değerlerini yüksek dinamik aralık (HDR) renklerine yeniden eşler), Vignette (görüntünün kenarlarını koyulaştırır), White Balanc (görüntüdeki beyaz alanları korur ve beyaz alanların etrafındaki diğer tonları dengeler). (Documentation, 2023)

Buit-in Partical System (Yerleşik Parçacık Sistemi), Unity'nin desteklediği her platform için efektler oluşturmayı sağlar. Bu sistem, CPU üzerindeki parçacık davranışını simüle ederek, bir sistemle ve içindeki parçacıklarla etkileşim kurmak için C# komut dosyaları kullanma ve Unity'nin temel fizik sistemini kullanarak sahnedeki Çarpıştırıcılarla (colliders) etkileşime girme imkânı sağlar.

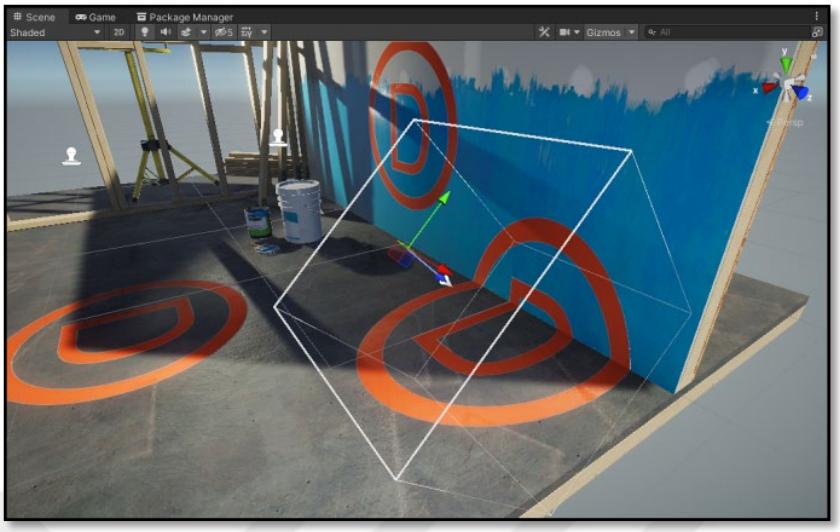


Şekil 32. Built-in Parçacık Sistemi ile yapılan efektler.
https://docs.unity3d.com/2023.2/Documentation/Manual/Built-inParticleSystem.html (2024)

Unity’de büyük ölçekli görsel efektler oluşturmak için kullanabilen Görsel Efekt Grafiği, GPU'daki parçacık davranışını simüle eder, bu da Yerleşik Parçacık Sisteminden çok daha fazla parçacığın simüle edilmesini sağlar. Çok sayıda parçacık içeren ve son derece özelleştirilebilir davranışa ihtiyaç duyan görsel efektler oluşturmak isteniyorsa Yerleşik Parçacık Sistemi yerine Görsel Efekt Grafiği kullanılmalıdır.

Şekil 33. Görsel Efekt Grafiği ile yapılan efektler.
https://docs.unity3d.com/2023.2/Documentation/Manual/VFXGraph.html (2024)

Decals (Çıkartmalar), diğer malzemelerin yüzeyini süsleyen malzemelerdir. En yaygın olarak boya sıçramaları ve çıkartmalar gibi efektler için kullanılırlar. Unity, bu malzemeleri diğer sahne geometrisinin etrafını saracak şekilde bir sahneye yansıtır. Bu, örneğin blob gölgeleri gibi, onları yansıtılan ışığı simüle eden efektler için de kullanabileceği anlamına gelir. Mercek parlamaları ve haleler, sahneye atmosfer katabilen ışık efektleridir.



Şekil 34. Evrensel Render İşlem Hattındaki Decals (Çıkartmalar). https://docs.unity3d.com/2023.2/Documentation/Manual/visual-effects-decals.html (2024)

Şekil 35. Evrensel Render İşlem Hattındaki Decals (Çıkartmalar). https://docs.unity3d.com/2023.2/Documentation/Manual/visual-effects-lens-flares.html

Sky (Gökyüzü), bir Kameranın bir kareyi oluşturmadan önce çizdiği bir arka plan türüdür. Bu tür bir arka plan 3B oyunlara ve uygulamalara büyük fayda sağlar çünkü derinlik hissi verir ve ortamın gerçekte olduğundan çok daha büyük görünmesini sağlar. Gökyüzünün kendisi, uzak üç boyutlu çevre yanılsaması yaratmak için bulutlar, dağlar, binalar ve diğer ulaşılamayan nesneler gibi her şeyi içerebilir. Unity, Sahnede gerçekçi ortam aydınlatması oluşturmak için bir gökyüzü de kullanabilir.

Unity Editör, geleneksel gama renk uzayının yanı sıra doğrusal renk uzayında da çalışmaya olanak tanır. Gama renk uzayı tarihsel olarak standart format olsa da doğrusal renk uzayı oluşturma daha kesin sonuçlar verir. İnsan gözü ışık yoğunluğuna

doğrusal bir tepki vermez. Bazı ışık parlaklıklarını diğerlerinden daha kolay görürüz- siyahtan beyaza doğrusal bir şekilde ilerleyen bir gradyan gözümüze doğrusal bir gradyan gibi görünmeyecektir.

Şekil 36. Doğrusal bir gradyan. Sağ: Gözlerimizin bu degradeyi nasıl algıladığı. https://docs.unity3d.com/2023.2/Documentation/Manual/LinearLighting.html (2024)

Tarihsel nedenlerden dolayı, monitörler ve ekranlar aynı karakteristiğe sahiptir. Bir monitöre doğrusal bir sinyal göndermek, yukarıdaki resimde sağdaki gradyan gibi görünen bir şeyle sonuçlanır ve gözümüze yanlış görünür. Bunu telafi etmek için, monitörün doğal görünen bir görüntü gösterdiğinden emin olmak amacıyla düzeltilmiş bir sinyal gönderilir. Bu düzeltme gama düzeltmesi olarak bilinir. Hem gama hem de doğrusal renk uzaylarının var olmasının nedeni, aydınlatma hesaplamalarının matematiksel olarak doğru olması için doğrusal uzayda yapılması, ancak sonucun gözümüze doğru görünmesi için gama uzayında sunulması gerektiğidir.

3. BÖLÜM

INDIE OYUNLARI VE BİR UYGULAMA PROJESİ

3. BÖLÜM

INDIE OYUNLARI VE BİR UYGULAMA PROJESİ

Indie oyun, genel olarak 2000'li yılların ortalarında öne çıkan, bağımsız bir geliştirici ekibi tarafından, genellikle sınırlı kaynaklarla, bir yayıncı olmadan veya yayıncının sanatsal sürece önemli bir katılımı olmadan oluşturulan bir oyun olarak tanımlanır (Fiadotau, 2018). "Indie", "independent" (bağımsız) kelimesinin kısaltılmış halidir ve bağımsız üreticilerin sahip olduğu düşünülen yaratıcı özerkliğe işaret eder. Başlangıçta müzik endüstrisinde ortaya çıkan bu kavram daha sonra sinema, moda ve nihayetinde oyun alanına yayılır. Bilgisayar laboratuvarlarında öğrenciler ve evlerinde yatak odası geliştiricileri tarafından tasarlanan "bağımsız oyunlar" onlarca yıldır var olsa da daha dar kapsamlı "indie" etiketi çoğunlukla 21. yüzyılda geliştirilen oyunlara atıfta bulunur. Bağımsız oyunların ortaya çıkışı, GameMaker, RPGMaker, Unreal Engine ve Unity 3B gibi tüketici sınıfı oyun motorlarının yanı sıra geniş bant internete yaygın erişim ve Web 2.0'a geçişin sağladığı dijital dağıtımın yükselişi ile mümkün olmuştur. Yaratılmak istenen oyun, PC'ler, konsollar ve cep telefonları da dâhil olmak üzere çok sayıda farklı cihazda piyasaya sürülebilir (Hill-Whittall, 2015).

Deneysel platform oyunlarından, sıkça tartışılan "sanat oyunlarına" kadar, bir yayıncılık anlaşmasının baskısı olmadan geliştirilen indie oyunlar benzersiz ve bazen de alışılmadık deneyimler sunarlar (Rose, 2011, s. viii). Indie oyunların geliştiricilerine göre, bu oyun türünün son yıllarda popüler olmasının nedenleri arasında büyük oyun firmalarının yaratıcı projeler üretememeleri; internetin yarattığı kendin yap anlayışının gelişmesi (klavyesi olan herkes köşe yazarı veya yazar, cep telefonu olan herkes film yönetmeni ve boş saati olan herkes bir video oyunu yapımcısı ya da yayıncısı olabilir); araçların ve mevcut platformların sayısının artması ve kullanımlarının daha kolay hale gelmesi; video oyunlarla büyüyen kuşağın yetişkin yaşa gelmesi; birkaç kişi tarafından geliştirilen indie oyunların çok daha kalabalık bir grup tarafından geliştirilen oyunlardan daha samimi ve insani bir deneyim sunmaları bulunmaktadır (s. xii-xiii).

3.1. Oyun Araçlarından Oyun Tarzlarına Indie Uygulamaları ve Estetiği

Apple'ın App Store'u ve Valve'ın Steam'i gibi dijital vitrinlerin ve Unity ve GameMaker gibi daha erişilebilir geliştirme araçlarının yükselişi, video oyunlarının yaratıldığı, dağıtıldığı, finanse edildiği, satın alındığı ve oynandığı küresel ve yerel bağlamların büyük ölçüde çeşitlendiği bir döneme denk düşer (Keogh, 2019). Üstte hem kapı bekçileri hem de kâr edenler olarak konsol üreticileri ve çok uluslu yayıncılar, altta ise ezici ve boğucu gizlilik anlaşmaları gibi iyi bilinen bir dizi emek sorunuyla mücadele eden, sömürülen oyun yapımcıların bulunduğu video oyunu endüstrisinin geleneksel, hegemonik ve hiyerarşik konfigürasyonları, çok daha geniş bir sosyo-ekonomik bağlamda video oyunu eserleri yaratan ve dağıtan yerel stüdyolar ve bireyler tarafından sorgulanmaktadır.

Enformel ve formel medya ekonomilerinin iç içe geçişi, Web 2.0 ve özellikle katılımcı sosyal medya platformlarının yükselişinin ardından özellikle yoğunlaşmıştır. Bireysel içerik yaratıcıları, yaratıcı faaliyetlerini dağınık niş kitlelere dağıtmak için resmi kurumsal platformlardan giderek daha fazla faydalanmakta ve bu platformlara güvenmekteyken, aynı zamanda bu resmi platformlar ticari olarak enformel içerik yaratıcıları tarafından üretilen içeriğe bağımlı hale gelmektedir (s. 18). 2000'li yılların sonlarında bağımsız geliştiriciliğin görünürlüğünün artması yeni oyuncu demografilerinin gelişmesine yardımcı olurken, konsol üreticilerinden ve gişe rekortmeni yayıncılardan bağımsız ve küçük ölçekli video oyunu geliştirme hem yeni hem de yerleşik geliştiriciler için giderek daha cazip hale gelir (s. 25). Ardından resmi video oyunu endüstrisinin daha önce pek bilinmeyen bir sektörü, üçüncü taraf (third party) oyun motorları ortaya çıkar.

Oyun motorları, sanatçıların, programcıların, ses mühendislerinin, tasarımcıların ve yazarların çeşitli varlıklarının ve emeklerinin bir araya getirildiği yazılım ortamını sağlayarak bir oyun projesinin iş akışının temelini oluşturur. Bir oyun motoru ve oyun geliştirme hattı için gerekli araçları geliştirmek son derece teknik ve pahalı bir süreçtir. 2000'li yılların sonlarında daha küçük ölçekli geliştirmenin yükselişiyle birlikte, büyük programlama ekiplerinin eksikliğini hafifletmek için erişilebilir oyun motorlarına yönelik talepler önemli ölçüde artar. Son on yılda ortaya çıkan çok çeşitli üçüncü taraf oyun

motorları arasında hiçbiri Unity kadar önemli olmamıştır. Unity oyun motoru hem birçok küçük ve orta ölçekli stüdyo tarafından kullanılabilecek kadar sağlam hem de amatörler ve hobiciler tarafından erişilebilecek kadar erişilebilirdir. Ekstra özelleştirme içeren bir 'Pro' Unity lisansı aylık nispeten yüksek bir maliyete mal olurken, motorun geliştirme araçlarının çoğuna erişim sağlayan bir 'Kişisel' lisans ücretsizdir. Ayrıca, geniş bir geliştirici topluluğu Unity Varlık Mağazası (Unity Asset Store) aracılığıyla dağıtılan ve daha sonra diğer geliştiriciler tarafından kendi oyun projelerinde kullanılabilecek uzantılar oluşturur. Böylece, Unity kullanıcıları Unity Varlık Mağazası için binlerce varlık ve eklenti geliştirecek ve bunların birçoğu motorun varsayılan araç setine dahil edecektir (Nicoll & Keogh, 2019, s. 35). Kullanıcılar ayrıca topluluk forumları ve bloglarında Unity hakkında gönüllü olarak bilgi, tavsiye ve öğreticiler sağlayarak bir kolektif zekâ havuzu yaratacaklardır.

> Apple'ın App Store'una çok benzer şekilde, Unity'nin Asset Store'u kullanıcıların çeşitli kullanıcı tarafından geliştirilen (bazen 'prefabrik' olarak adlandırılan) varlıkları ve eklentileri satın almasına, satmasına ve/veya serbestçe edinmesine olanak tanır. Bir eklenti örneği, Unity tarafından geliştirilen içerikte görüntülerin nasıl işleneceğini belirleyen bir 'gölgelendirici' aracıdır. Geçmişte, geliştiricilerin özel gölgelendiriciler yazması veya bir motorun varsayılan gölgelendiricileriyle idare etmesi gerekirdi. Asset Store aracılığıyla kullanıcılar Shader Forge ve Shader Amplify gibi birçoğu ücretsiz olan veya tek seferlik bir ücret karşılığında satılan, kullanıcı tarafından geliştirilmiş çok sayıda gölgelendirici eklentisi oluşturabilir, paylaşabilir ve edinebilir (s.39).

Unity'nin (ve daha az ölçüde Epic'in benzer Unreal motorunun) video oyunu geliştirme uygulamaları üzerindeki etkisi oldukça fazladır. Kaynaklar profesyonel ve amatör geliştiriciler arasında hala önemli ölçüde farklılık gösterse de, birçoğu artık yaratım için aynı araçları kullanıyor ve gayri resmi (in/formal) geliştiricileri katı bir şekilde düzenlenmiş son kullanıcı 'modder' konumundan kurtararak bunun yerine resmi geliştiricilerle aynı ara katman aktörleri tarafından düzenlenmelerini sağlar. Sonuç olarak, beceri setleri standart hale gelmekte ve aktörlerin in/formal ayrımında geçiş yapabilmeleri daha mümkün hale gelmektedir. Oyun endüstrisinde inovasyonu engellediği düşünülen

gizlilik kültürü ve bunu takip eden temel sistemlerin yeniden icat edilmesine yönelik bitmek bilmeyen döngü, ticari sırları saklamaktan daha az endişe duyan bağımsız ve gayri resmi geliştiricilerden oluşan bu daha geniş, daha işbirlikçi topluluk tarafından sorgulanır hale gelir. Hatta bazı Unity geliştiricileri senaryolarını çevrimiçi olarak yayınlamakta ve diğer geliştiricileri bunları kullanmaya ve/veya yinelemeye davet etmektedir. Los Angeles, Haydarabad, Berlin, Melbourne ve Singapur'da düzenlenen yıllık 'Unite' konferansları gibi Unity etkinliklerinde şirket temsilcileri, geliştiriciler, bağımsızlar ve öğrenciler ağ kurmak, bilgi paylaşmak ve en son Unity özelliklerini tartışmak için bir araya gelirler (s.42). Unity kullanıcı tabanı sadece bilginin açık paylaşımını benimsemekle kalmaz, aynı zamanda kolektif zekasının motorun varsayılan araç setine dahil edilmesini bekler. Bu durum, bir zamanlar ana akım stüdyo gelişimini karakterize eden kapalı tescilli yapılardan ve hiyerarşik iş modellerinden çok uzaktır.

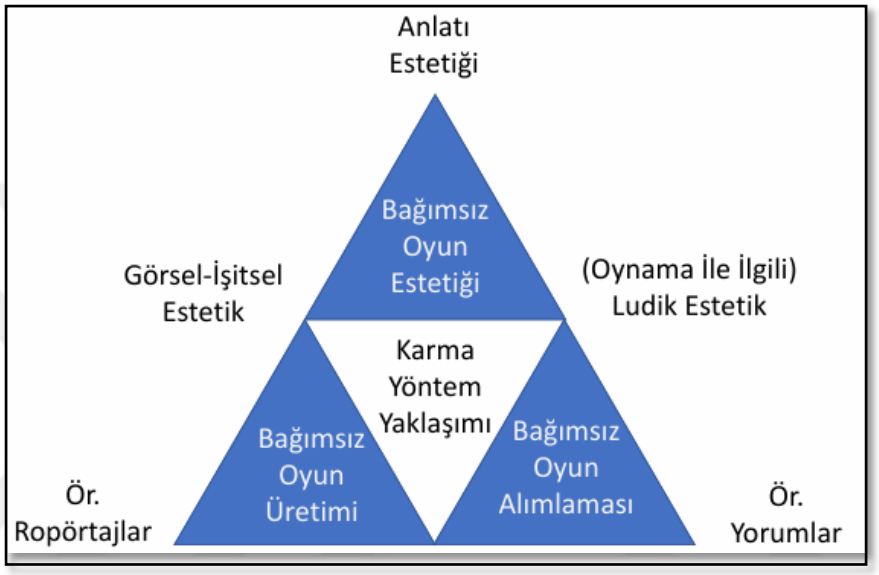
Küçük bir stüdyo tarafından geliştirilmesi gerektiği hala genel olarak kabul görse de (ancak büyük bir bütçeyle ve büyük bir yayıncıyla çalışabilir), günümüzde indie oyun genel bir "indie duyarlılığı" ve indie estetiği ile tanımlanır. Ana akım oyunlara karşı söylemsel muhalefetinin yanında, eski video oyunlarının görünüm ve hissini çağrıştıran pikselli retro grafikleri ve chiptune müzik kullanımları bu oyun türünü niteleyen estetik özelliklerdir. Yazılım araçları ve internet tabanlı topluluklar tarafından mümkün kılınan indie uygulamalarının birçoğu, araç kullanıcılarını, informal geliştiricileri ve hatta ticari olarak başarılı bağımsız oyunları etkileyecek farklı stiller ve estetik yaklaşımlar gelişmesine neden olur. (Reed, 2020, s. 119). Örneğin, ilk olarak 2016'nın sonlarında piyasaya sürülen ve geliştiricilerin 8 × 8 karolar ve sınırlı renk şemalarıyla keşfedilecek piksel sanat dünyaları oluşturmaya hemen başlamalarına olanak tanıyan Adam LeDoux'un Bitsy adlı tarayıcı içi uygulaması, daha sonra bir itch.io sayfasına gömülebilen ve çevrimiçi olarak kolayca paylaşılabilen bir HTML dosyası olarak dışa aktarılabilir. Bir diğer örnek, Fernando Ramallo tarafından yakın zamanlarda üretilen ve kullanıcıların doğrudan Doodle Studio 95 adlı motorda sprite, animasyon ve shader efektleri çizmelerine olanak tanıyan ve arayüzünde görsel olarak 1990'ların Kid Pix ve Klik n Play gibi "yaratıcılık yazılımlarına" atıfta bulunan bir Unity eklentisidir. “Bu eklenti,,, dijital

yaratımı yalnızca mümkün değil, aynı zamanda görsel olarak çekici, davetkar ve erişilebilir kılmak isteyen diğer programlara bir geri dönüş çağrısı yapar" (s. 119).

Maria B. Garda ve Paweł Grabarczyk (Is Every Indie Game Independent? Towards the Concept of Independent Game, 2016) "indie" teriminin sadece "bağımsız" teriminin bir kısaltması olmadığını, aslında, daha geniş bir kavram olan "bağımsız" video oyunu içinde ayrı bir tarihsel kavram olarak anlaşılması gerektiğini ileri sürerler. "Bağımsız oyun" kavramı, üç tür bağımsızlığın kesişimi olarak açıklanabilir: 1) geliştirici ve yatırımcı ilişkisi açısından finansal bağımsızlık; 2) geliştirici ve hedef kitle ilişkisi açısından yaratıcı bağımsızlık; 3) geliştirici ve yayıncı ilişkisi açısında yayıncılık bağımsızlığı. Bu üç bağımsızlık türü, oyun ile ilgili bir dış faktör arasındaki ilişki olarak tanımlanabilir çünkü hepsi bir şeyden bağımsızlığı temsil eder. Oyun bağımsızlığı açısından bu bağımsızlık türlerinden her biri tek başlarına ele alındıklarında zorunlu, birlikte ele alındıklarında yeterli koşulu sağlarlar. Dolayısıyla, Garda ve Grabarczyk'a göre "indie oyun" terimi, Batı dünyasında 2000'li yılların ortalarında ortaya çıkan belirli bir tür bağımsız oyunla ilişkili "indie belirteçleri" ifade eden ve (dijital dağıtım, retro tarzı veya küçük ekip vb.) dönemin video oyun kültürünün kültürel, sosyal, ekonomik ve teknolojik koşulları tarafından belirlenen bir dizi koşullu özelliğe gönderme yapan tarihsel bir kavramdır.

Buna karşın, "Analyzing Indie Aesthetics" başlıklı makalesinde Jan-Noël Thon (Thon, 2020). Jesper Juul'un "bir temsilin temsili" olarak tanımladığı, daha önceki zamanların stillerini taklit etmek için çağdaş teknolojiyi kullanan "bağımsız stil" kavramına müracaat etmenin daha elverişli olacağını söyler. "Bağımsız üslup" dolaysızlık (immediacy) mantığının aksine aracılık işaretlerini çoğaltan" ve böylece bizi ortamın veya medyanın farkına vardıran hiper aracılık mantığına bağlı kalma eğilimindedir. Thon'a göre indie oyunlar ve estetikleri (ve aslında diğer video oyunları ve estetikleri) şu üç boyutta verimli bir şekilde analiz edilebilir: görsel-işitsel (audio-visual), ludik (ludic) ve anlatı (narrative) estetiği. Görsel-işitsel estetik, bir bağımsız oyunun uzamsal perspektifler, görsel stiller ve resimsel malzemeler yanında ses tasarımı, müzik ve/veya seslendirmenin de dahil olduğu görsel-işitsel tasarımına atıfta bulunur. Ludik estetik, bir bağımsız oyunun oyuncularına sunduğu etkileşim olanaklarını, oyun mekaniklerini ve

oyun hedeflerini ve sonuçta ortaya çıkan deneyimsel kaliteyi, kısacası oyunun "hissini" ifade eder. Anlatı estetiği ise yalnızca oyunda anlatılan hikâye veya hikayeleri değil, aynı zamanda doğrusal olmayan anlatı biçimleri, anlatıcı-karakterlerin kullanımı ya da karakterlerin öznelliğine dolayımlı (mediated) "doğrudan erişim" gibi anlatı stratejilerini de içerir.



Şekil 37. Thon'un Söylemsel bir yapı olarak indie estetiğinin analizine yönelik karma bir yöntem yaklaşımı.

Büyük şirketlerin ürettikleri modern ana akım video oyunları, sürekli olarak daha yüksek çözünürlüklü grafikler, hiper detay, gerçeğe yakınlık ve karmaşık oynanış özelliği ve estetiğini destekleyen homojen başlıklar üretmektedir (Dolan, 2021, s. 18). Ancak, tüm bu teknik ilerlemelere rağmen, türler, temalar, anlatılar ve oynanışlar büyük ölçüde aynı kalmaktadır (s. 19). 2020'nin en çok satan beş oyunu Call of Duty: Black Ops Cold War (Treyarch, Raven Software), Call of Duty: Modern Warfare (Infinity Ward 2019), Animal Crossing: New Horizons (Nintendo EPD), Madden NFL 21 (EA Tiburon) ve Assassin's Creed: Valhalla (Ubisoft Montreal) (birinci şahıs nişancı, çiftçilik simülatörü / rahat oyun, spor simülatörü ve açık dünya macerası gibi) yerleşik türlere uyan, Animal Crossing: New Horizons hariç tümü en üst düzey, sinematik grafiklere sahip oyunlardır. Bu duruma karşı bir "estetik güce" ihtiyaç vardır ve bu güçlerden biri

piksel grafiklere ve basitleştirilmiş oynanışa sahip video oyunları ya da post-retro oyunlardır. Dolan'a göre "post-retro" terimi, ister görsellerde ister oynanışta olsun retro estetiği benimseyen ve uyarlayan, geçmiş oyunların öz farkındalığına sahip, ancak aynı zamanda doğrudan nostaljiden ayrılan oyunları tanımlar (s. 21). Post-retro oyunlar "tamamen yeni bir deneyim yaratmak için hem retro hem de modern oyun öğeleriyle karıştırılmış retro bir estetik kullanır" (Fulton and Fulton, 2010, akt. Dolan s. 21). Braid (Number None 2009), Super Meat Boy (Team Meat 2010) ve Fez (Polytron Corporation 2013) gibi post-retro oyunlar 2010'lu yılların başında yaygın ticari ilgi görmeye başlar ve bu oyunlarla ilgili söylemlerin çoğu nostaljiyi çağrıştırır. Post-retro oyunlar geçmiş formları diriltirken, AAA endüstrisi tarafından üretilmeyen yeni tür oyunlar ve hikayeler yaratırlar.

3.2. Sandbox İnşa Oyunları

Strateji, macera, nişancı, spor veya sürüş oyunları gibi geleneksel türlerden herhangi birine girmeyen, çoğu zaman farklı oyun deneyimlerinin bir araya getirilmesinden oluşan Sandbox oyunlar doğrusal olmayan ve oyunculara, oyun boyunca ilerlemek için yüksek derecede özgürlük sağlayan büyük, açık, hayat dolu dünyalar sunan, doğrusal hikâyeyi takip etmek zorunda kalmadan istedikleri her şeyi yapmalarına izin veren oyun sistemleridir (Ocio & Brugos, 2009).

Çok oyunculu birinci şahıs (first-person) bir sandbox oyunu olan Minecraft, oyuncuların avatarları bir ızgara boyunca hareket ederken, 3B nesneleri toplarken, yerleştirirken ve inşa ederken sanal ortamlar tasarlamalarına olanak tanır. Oyun, oyuncuların hayatta kalmak için sağlıklarını korumaları gereken *hayatta kalma* ve oyuncuların mallara sınırsız erişimine izin veren *yaratıcı* olmak üzere iki mod kullanır (Hebert & Jenson, 2020). Son zamanların en önemli ve popüler indie video oyunlarından biri haline gelen Minecraft, Dwarf Fortress ve Infiniminer gibi oyunlardan etkilenmiştir. Oyun, sınırlı bir alanda geçen bir sandbox inşa oyunu olarak başlar ancak kısa süre içinde prosedürel olarak oluşturulan rastgele arazileri içerecek şekilde büyür. Ardından düşmanlar dahil edilir, yiyecek gibi öğeler hayatta kalmak için gerekli hale gelir ve oyuncunun kaynak elde etmesi için blokların çıkarılması gerekir.

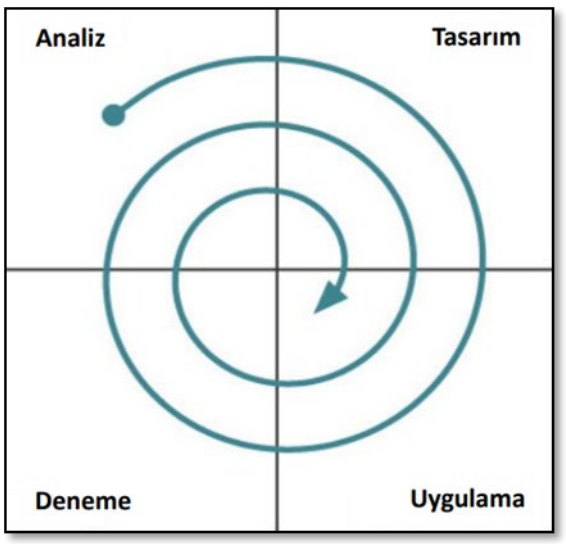
MineCraft gibi açık dünya sandbox oyunları, oyuncuyu yapılandırılmış doğrusal oyunun dışına çıkmaya, yaratıcı olmaya ve sanal dünya ile özgürce etkileşime girmeye teşvik etmek için tasarlanmıştır (Kleanthous, Christodoulou, Papadopoulos, & Samaras, 2018, s. 69-74). Sandbox oyunları, geleneksel hedef veya skor odaklı oyunlardan daha açık uçlu ve oyunculara oyunu nasıl oynayacaklarına karar verme ve kendi hedeflerini belirleme konusunda daha fazla özerklik verme eğiliminde olduğundan pek çok disiplinde oyun temelli, açık uçlu yaratıcı eğitimin bir aracı ve modeli olarak araştırılmakta, ders planlarına entegre edilmektedir. Ayrıca Minecraft sınıf kullanımı için özel olarak tasarlanan Minecraft Eğitim Edisyonunu piyasaya sürer. Minecraft'taki sanal dünya, oyuncuların oyun içindeki blokları yok edebilmesi ve yerleştirebilmesi açısından şekillendirilebilir. Bu öğretmenlerin kendi ders planlarını oluşturmalarına olanak tanıyan bir özelliktir. Eğitim faaliyetlerini desteklemek için eğitim sürümünde belirli öğeler ve özellikler benzersiz bir şekilde mevcuttur. Minecraft Eğitim Sürümünü kullanan öğretmenler, öğrencileri için sanal dünyaları, en büyüğü Minecraft Eğitim web sitesi olmak üzere çeşitli kaynaklardan indirebilmektedirler. (Bar-El & Ringland, 2021). Minecraft'ta oyuncular, kendi doğal kurallarını takip eden ve gerçek dünyayı çeşitli derecelerde yansıtan dijital bir dünya ile etkileşime girerler. Oyun dünyası, yeryüzündeki coğrafi çeşitliliği dar bir şekilde temsil eden ovalar, çöller ve dağlar gibi çeşitli biyomlardan oluşur. Minecraft'ın bu özelliği öğretmenlerin, öğrencilerden Minecraft dünyası ile gerçek dünya arasında karşılaştırmalar yapmalarını istemelerine olanak sağlamaktadır. Ayrıca öğretmenler oluşturdukları öğrenci gruplarından, belirlenmiş bir dünyayı virtuel olarak birlikte inşa etmelerini isteyebilmektedirler.

Bir diğer popüler sandbox oyunu olan The Sims (2008 Electronic Arts, Inc.) oyununda oyuncu, önceden inşa edilmiş ya da oyuncu tarafından inşa edilmiş ve döşenmiş bir evde yaşayan Sims adı verilen simüle edilmiş insanları kontrol eder. Her Sim'in açlık, enerji, sosyal, eğlence, mesane ve hijyen gibi tatmin edilmesi gereken bir dizi ihtiyacı vardır ve tatmin seviyeleri zamanla azalır. Mobilya gibi nesnelerle ve diğer Sim'lerle etkileşimler ihtiyaç tatmin seviyeleri üzerinde olumlu ve olumsuz etkilere sahip olabilir. Oyuncunun temel görevi, Sim'in ihtiyaçlarının kritik derecede azalmasını önleyecek eylemleri seçmektir. Oyuncunun belirleyebileceği kişisel hedeflerin ötesinde

bir "kazanma" koşulu yoktur. Bununla birlikte, belirli ihtiyaçlar çok uzun süre karşılanmazsa bir Sim ölebilir. Bu nedenle, The Sims iki ayrı ancak birbiriyle ilişkili oyun zorluğu içerir: 1) bir Sim'in ihtiyaçlarının kritik derecede azalmasını önlemek için yeterli nesneye sahip bir ev tasarlamak ve 2) bu nesnelerle etkileşimleri Sim'in azalan ihtiyaç tatmin seviyelerinin yeterince yenilenmesini sağlayacak şekilde seçmek (Charity, Rajesh, Ombok, & & Soros, 2020, s. 182-188).

İlkokul, orta ve üniversite düzeyindeki resmi sınıf ortamlarında, SimCity gibi şehir inşa oyunlarının değerli pedagojik araçlar olduğu, öğrencilerin bütüncül düşünmelerine, şehirleri birbiriyle bağlantılı ve birbirine bağımlı birçok parçası olan karmaşık bir sistem olarak anlamalarına yardımcı olduğu ve eleştirel düşünme becerilerini güçlendirmede ve öğrencileri coğrafi örüntü ve süreçlerle tanıştırmada etkin olduğu ileri sürülmektedir (Bereitschaft, 2016, s. 51-60). Ayrıca oyun öğrenciler bir şehrin trafik, kirlilik, suç, doğal afetler, negatif nakit akışı, atık birikimi, vatandaş huzursuzluğu gibi birçok zorluğuyla yüzleşirken eleştirel muhakemeyi güçlendirmeye de yardımcı olabilir. Sonuç olarak, resmi bir pedagojik araç olarak sınırlılıklarına rağmen, şehir inşa oyunları ve diğer video oyunları öğrenme ve bilişsel gelişim için hala etkili ve benzersiz bir şekilde avantajlı bir ortam sunabilirler.

3.3. Mini Island'ın Tasarım, Prototip ve Gelişim Süreci



Şekil 38. Yinelemeli tasarım süreci

İyi bir oyun tasarım ve geliştirme sürecinin temeli yinelemeli test ve iyileştirme döngüleridir (Bond, 2018). Yinelemeli tasarım sürecinin dört aşaması bulunur: analiz, tasarım, uygulama ve deneme. Analiz aşaması tasarımın hangi aşamada olduğunu, neyin elde edilmek istendiğini anlamakla ilgilidir. Bu aşamada tasarımda çözülmeye çalışılan sorun, projeye aktarabilecek kaynaklar ve tasarımı uygulamak için sahip olunan süre net bir şekilde anlaşılmalıdır. Analiz aşamasını, eldeki kaynakları kullanarak sorunu çözmek üzere beyin fırtınası ile başlayıp uygulama için somut bir planla sona erecek olan tasarım aşaması izler. Bir sonraki aşama oyun tasarımı fikrinden oynanabilir prototipe mümkün olduğunca çabuk ulaşmayı amaçlayan uygulama aşamasıdır. Bir prototipin asgari düzeyde çalışması sağlandıktan sonra, sıra onun oyuncu tarafından test edilmesine gelir.

Prototip, oyunun formal sisteminin görsel, işitsel ve diğer özelliklerin kaba bir modeli ve oyun mekaniğinin nasıl işlediğini görmeye olanak sağlayan oynanabilir bir versiyondur (Tracy Fullerton, 2004). İki tür prototip vardır: fiziksel prototipler ve yazılım prototipleri. Fiziksel prototipler kalem ve kâğıt veya diğer gerçek dünya malzemeleri kullanılarak oluşturulurken, yazılım prototipleri tamamen dijitaldir ve bilgisayar koduna dayanır. Oyun tasarımcılarının tasarım fikirlerini yalnızca kâğıt üzerinde değil, aynı zamanda çalışan dijital prototipler aracılığıyla da taslak haline getirebilmeleri giderek daha önemli hale gelmektedir. Unity gibi gelişmiş, ancak ulaşılabilir oyun geliştirme motorlarının ortaya çıkması, oyun tasarım fikirlerini ve bu fikirlerin temel fikirden çalışan dijital prototipe gelişimini başkalarına ifade eden oynanabilir prototipler oluşturmayı kolaylaştırmıştır.

Grafik kartının, örgü verilerini ekranda bir görüntüye dönüştürmek için attığı, görüntü işleme iş akışı (rendering pipeline) olarak adlandırılan bir dizi adıma müdahale ederek ilginç efektler elde etmek için kullanılan gölgelendirici programlarını ve uygulamalarını odağına alan bu tez kapsamında Unity oyun motoru kullanılarak geçekleştirilen “Mini Island” uygulama projesi, indie (bağımsız) oyun türü içinde açık dünya sandbox inşa oyunları kategorisine dahildir.

Şekil 39. Oyun ekran görüntüsü

Projeye esin kaynağı olan oyunlardan bazıları The Block, Hidden Farm Top-Down 3D, Toem, Dorfromantik, Let's Build a Zoo, StardewValley olarak sıralanabilir. Minik bir ada içerisinde geçen bir inşa simülasyonu (building simulation) olarak planlanan ve adanın içerisinde çiftçilik ve balıkçılık yapma ve kaynakları doğru kullanma gibi görece sade ve gündelik strateji ve keşif özelliklerini içeren oyun, oynama aracını fare ile sınırlandırmasıyla da oyuncuya keyifli ve huzurlu bir deneyim yaşatmayı amaçlar. Ağırlıklı olarak Amplify Shader Editörün kullanıldığı projede ayrıca Dotween, Zenject, UnmaskForUGUI, UIEffect v.b eklentilerden yararlanılmıştır (UI eklentili programlar için 2. Bölüme bakınız).

3.3.1. Mini Island Oyun Tasarım Belgesi

Bir oyun üretim sürecinin ilk aşamasını Oyun Tasarım Dokümanı (OTB) oluşturmaktır. Bu belge oyunun temel kurallarının, konseptinin vb. kararların alınması için yapılır. Oyun tasarım belgesi oyunun konsepti, türü, oynanış, mekanik, modeller, bölümler, müzikler, hangi programların kullanılacağı gibi unsurlar içerebilirken aynı zamanda sadece temel mekaniği içeren versiyonları da vardır. Bu şekilde büyük veya orta boy ekipler aynı oyun üzerinde çalışırken karmaşıklığın önüne geçilmiş olur. Bu anlamda Mini Island oyunun temel tasarımına ait başlıklar şu şekildedir:

Oyun Türü: Sandbox, İnşa simulasyonu, Indie

Oyun motoru: Unity

Oyun Mekaniği: Oyun bir ada kurmak üzerine işler. Kendi adamızı büyütüp geliştirerek oynanır. Bunu yaparken çiftçilik, balıkçılık gibi para kazandıracak ve oyunun ilerlemesini sağlayacak zorluklar vardır. Adamızı büyütmek için ek blok satın alma ve araştırma yaparak yeni buluşlar elde etmek. Bunların yanında adanın fotoğraflarını çekmek gibi unsurlar vardır. Oyunda iki farklı mod bulunur. Biri sınırsız para ve gelişme ile sadece modelleri yerleştirme, diğeri ise oyunda ilerleme yapmaya çalışma. Kamera hareketi: Oyunda "q" ve "e" tuşları ile doksan derece sola ve sağa giden bir kamera vardır. Mause kaydırma yoluyla yakınlaşıp uzaklaşılabilir.

Oyun tasarım: Tasarımlar 3B modeller ile elle çizilmiş hissiyatı yaratmayı hedefler. Bu şekilde oyuncuların kendi adalarının fotoğrafını çekip güzel görseller yakalamasına olanak tanır.

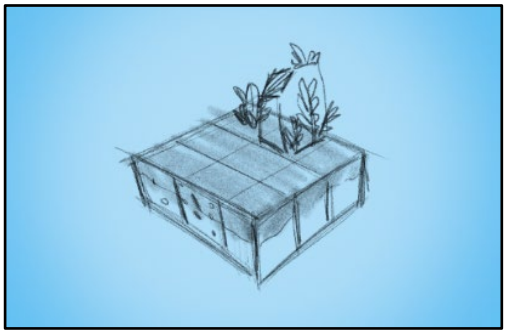
Arayüz Ekranları Listesi: Start Game, InGame, End Game, Settings, Task, Photo, Load, Camera, Store, Upgrade

3.3.2. İlk Çizimler

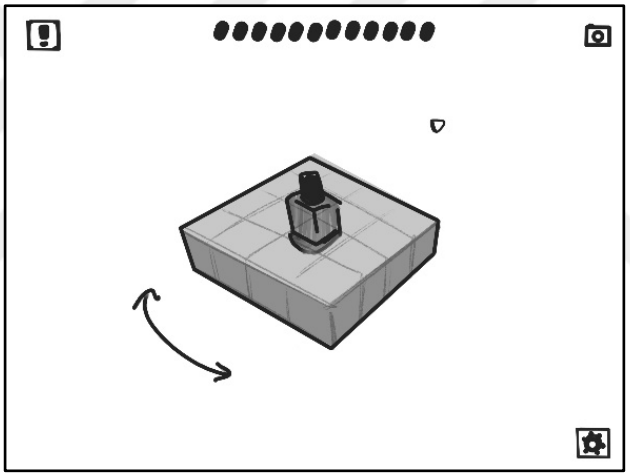
Oyun üretimi içerisinde en kreatif olunabilecek ve teknik tarafı hiç düşünmeden konsept üzerine çalışılması gereken aşama skeç aşamasıdır. Bu aşamada oyun karakterleri, arayüzler, görsellik gibi unsurların tasarımları için ön çizimler yapılır. Böylece oyun yapımına geçilince tasarımlar belli bir netlik kazanmış olur. Prototip oluşturma sürecinde ilk çizimlerden hareketle görsellik ve tekniğe ilişkin problemler bütüncül bir perspektiften kavramsallaştırılır ve çözümlenir.

Mini Island oyunu içerisinde çizimler aşamalı olarak gelişti. Öncelikle temel eskizler yapılırken kamera açısı, blokların ekranda ne kadar yer kapladığı, arayüz tasarımları, ek blok yerleştirme meseleleri ele alındı. Bu aşamada görsellikten çok mekanik ve kullanılabilirlik dikkate alındı. Oyun tasarımı geliştirilmesi sürecinde çizim aşaması tekrar tekrar geri dönülen bir aşama olarak modellerin ve görsel tasarımın denenmesinde vaz geçilmez bir araç olma özelliğini devam ettirdi. Bu nedenle oyun içerisinde farklı gelişim aşamaları için yapılmış için farklı çizimler söz konusudur.

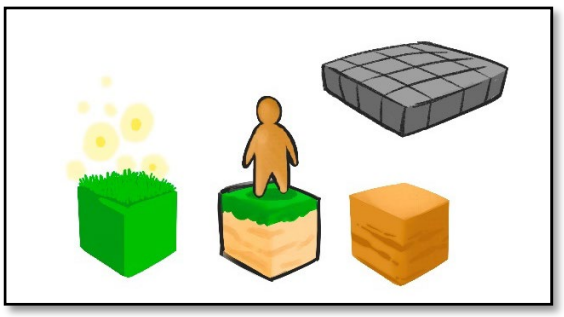
Bazıları görsel açıdan oldukça uğraşılmış iken not düşmek amaçlı olan diğerleri daha basit çizimlerdir.



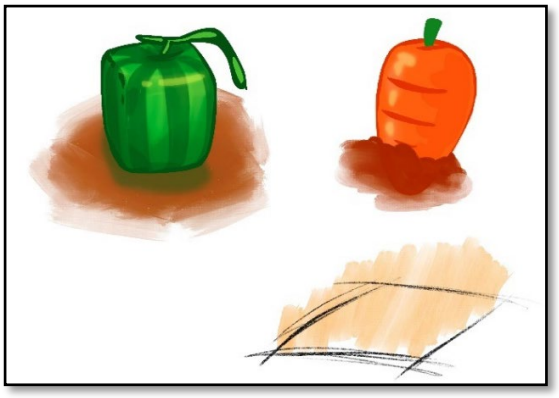
Şekil 40. Eskiz



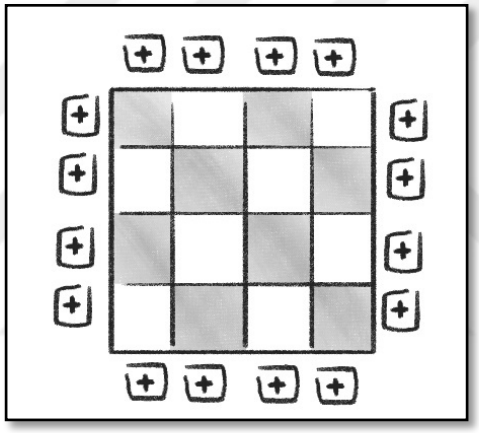
Şekil 41. Eskiz



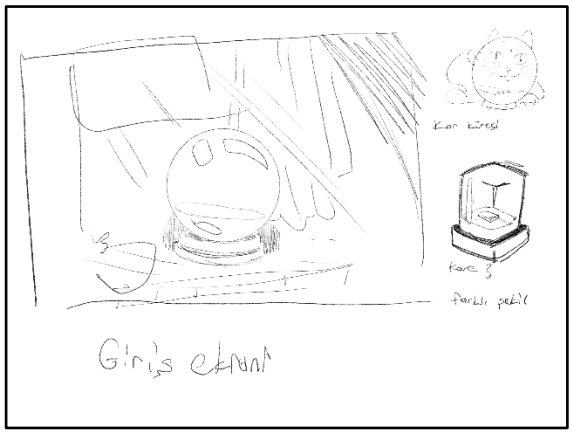
Şekil 42. 3B Blok doku denemeleri.



Şekil 43. 3B Ekilen ürünler neler olabilir

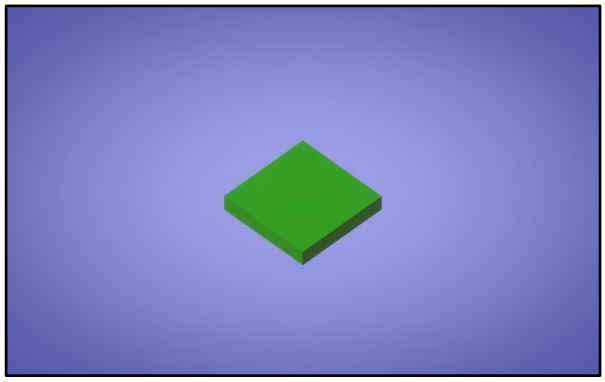


Şekil 44. Ada büyütme sistemi tasarımı



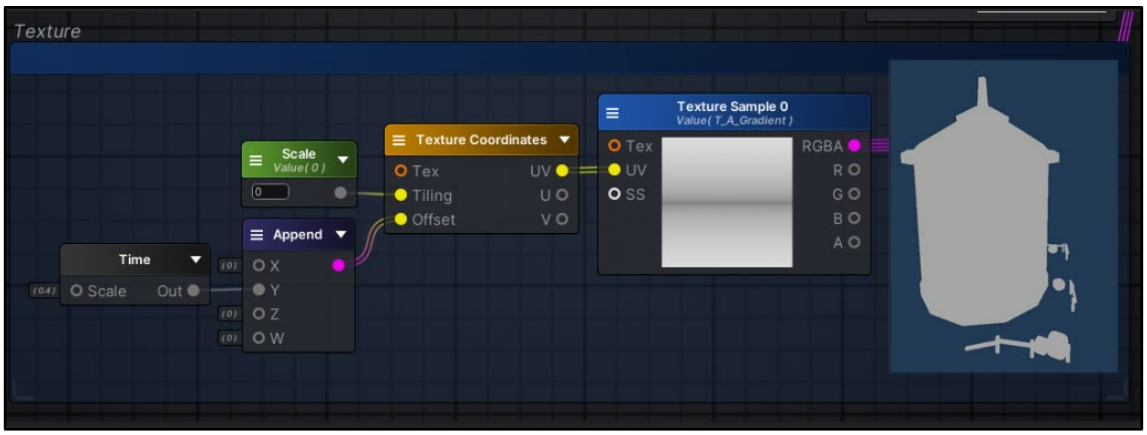
Şekil 45. Eskiz

3.3.3. Prototipler ve Gölgelendiriciler



Şekil 46. İlk Prototip

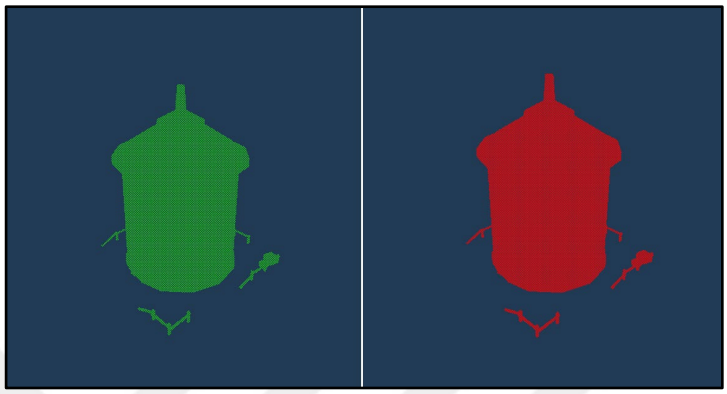
İlk prototipte belirli bir alanın üzerine fare getirildiğinde yerleştirilmek istenen nesne henüz tıklamadan görülemiyor. Oyuncu seçtiği bir nesneyi yerleştirmek istediğinde müsait ve müsait olmayan alanlara göre nesnenin sırasıyla yeşil ve kırmızı ve halihazırda yerleştirilmiş opak nesnelere göre şeffaf görünmesini sağlamak ve ayrıca fare belli bir alan üzerine geldiğinde oyuncuya bir şey yapması beklendiğini işaret edecek bir animasyon gerçekleştirmek üzere bir gölgelendirici oluşturmak gerekiyor.



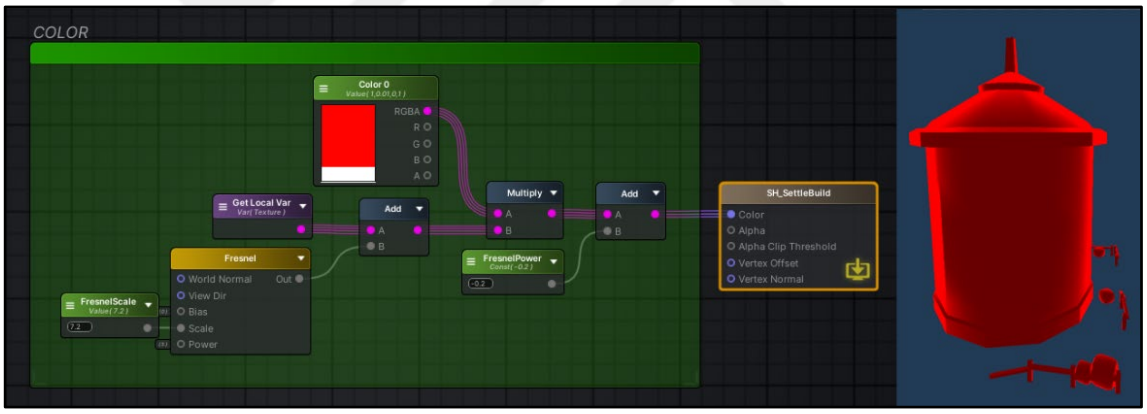
Şekil 47. Gölgelendirinin doku ile zaman nodları eklenmiş hali

Bu gölgelendirici üç farklı parçası var. Renk, dither ve animasyon. Öncelikle doku içerisinde basit bir opaklığı artırıp azaltma efekti eklemek için bir zaman (time) ve doku (texture) nodları kullanıldı. Doku nodu'nu "y" ekseninde zaman içerisinde kaydırarak bir efekt oluşturuldu. Kullanılan doku içerisine bağlı olarak efekt

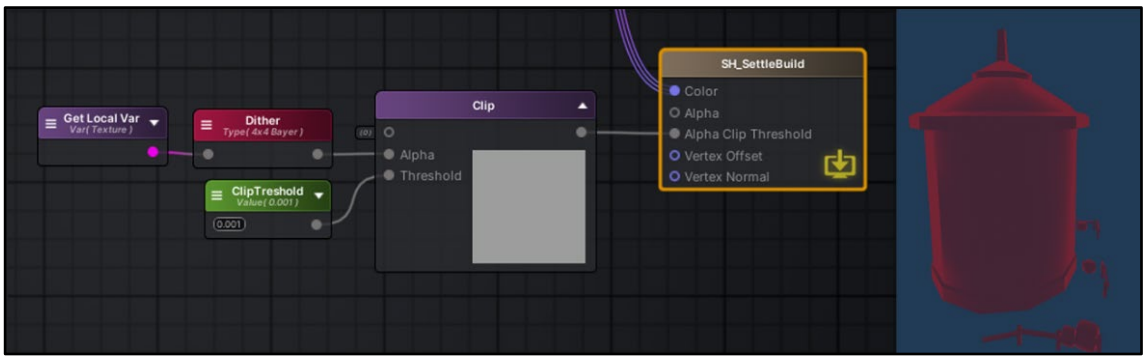
değişebiliyor. Hemen arkasından Frenel efekt ve renk eklendi. Bu şekilde üç boyutlu nesnenin yumuşak bir geçiş efekti elde edildi. Son olarak dither ve clip kullanılarak nesnenin opaklığı azaltıldı.



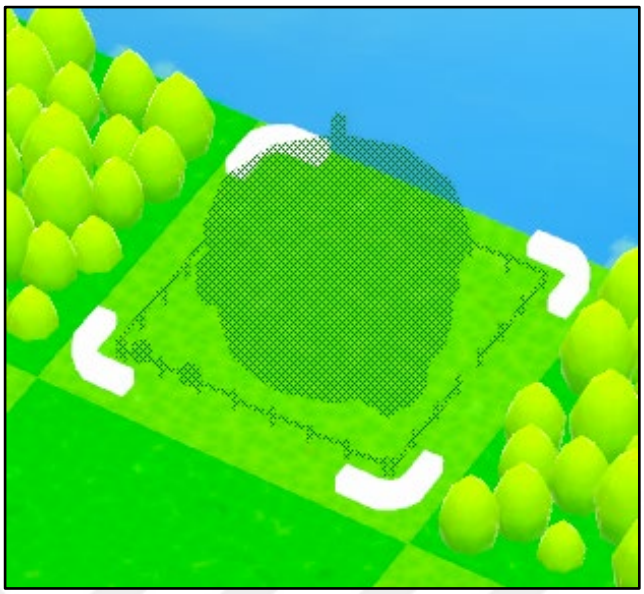
Şekil 48. Kırmızı-yeşil renk değişkenleri gölgelendirici çıktısı



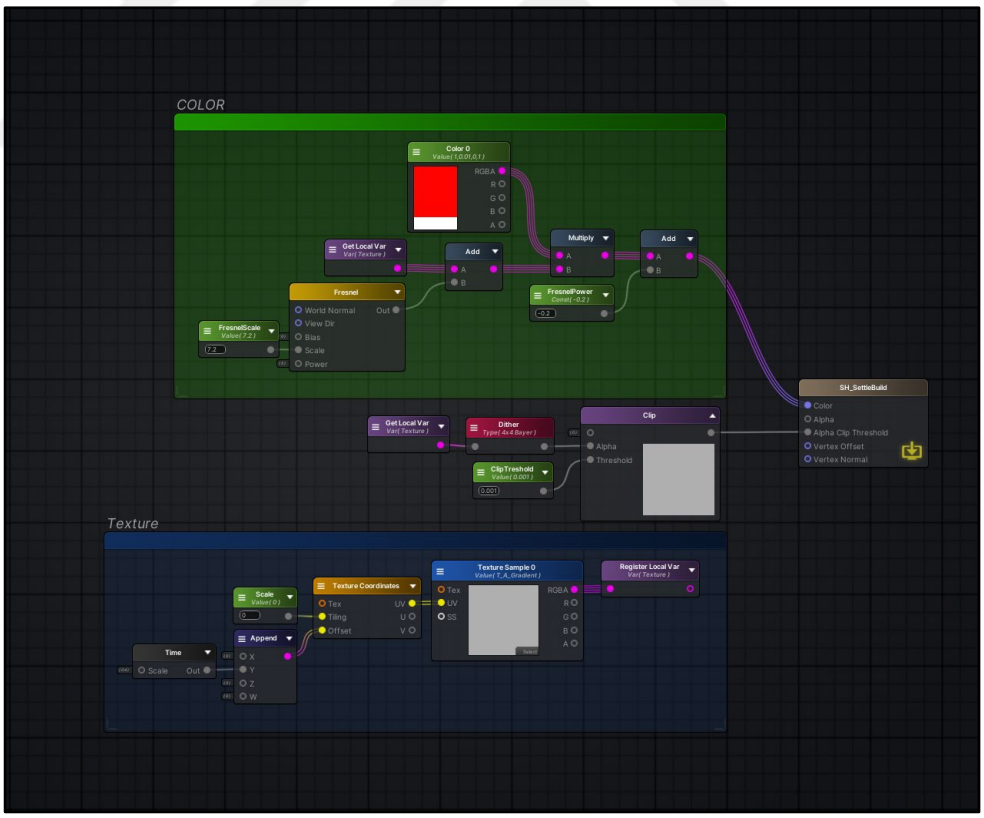
Şekil 49. Kırmızı renk ve frenel eklenmiş hali



Şekil 50. Dither efekti eklenmiş hali



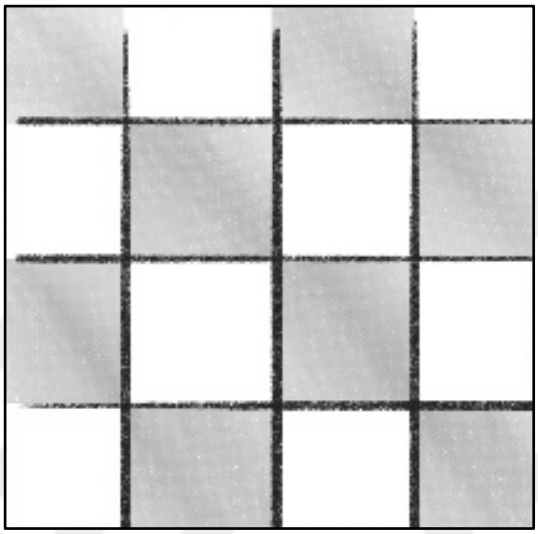
Şekil 51. Sahnede dither efekti görünümü.



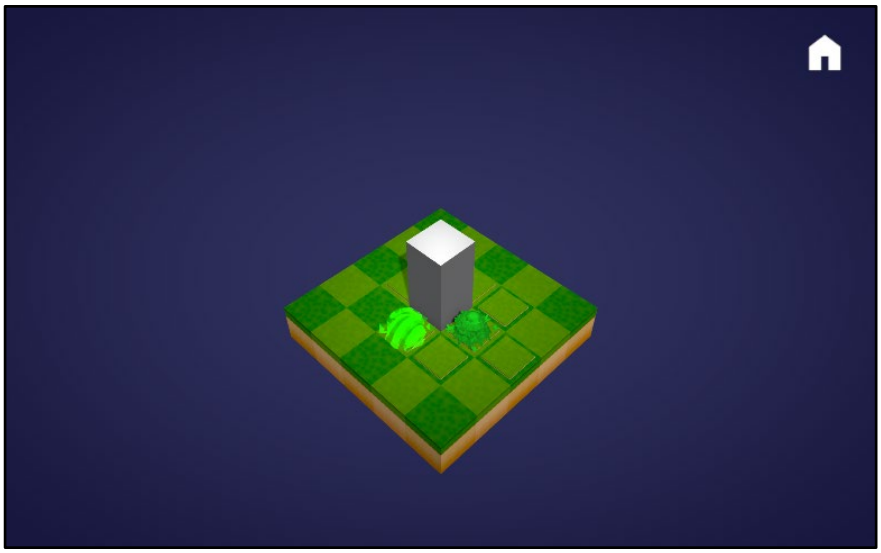
Şekil 52. Dither gölgelendiricisinin bitmiş hali

Şekil. Renk, dither ve animasyon için oluşturulmuş gölgelendiricinin son hali

Mekanik açısından yan yana dizilmiş bloklardan oluşan ada sathı nesne yerleştirme alanı olarak anlaşılamıyor. Ada zeminini oluşturan yan yana dizilmiş blokları nesne yerleştirme alanları olarak vurgulamak için bir dama tahtası görünüşü faydalı olacaktır. Bunun için renkleri bir miktar multiply yaparak koyulaştıran bir gölgelendirici yapıldı.



Şekil 53. Dama tablası tasarımı

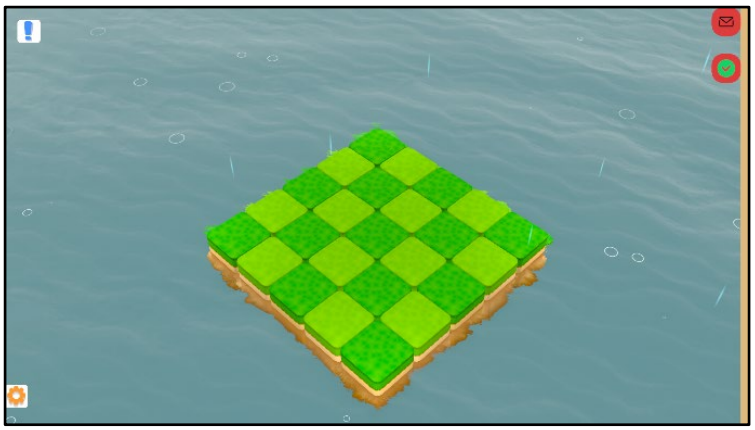


Şekil 54. Prototip 2

İkinci prototipe baktığımızda her ne kadar mekanik olarak daha anlaşılabilir bir versiyon olsa da görsel olarak fazla koyu ve ışıklandırma konusunda geliştirilmeye muhtaç.

Şekil 55. Çapraz ışık eklenmiş hali

Bu durumu düzeltmek için ilk olarak ışıklandırma üzerine çalışıldı. Tepeden gelen ışığın sert etkisi eklenen çapraz bir ışık kaynağı ile yumuşatıldı. Bunun yanında Skybox daha açık bir mavi rengi ile değiştirildi. Post proses yardımı ile Ekranın etrafına bir vinyet efekt eklendi. Oyunu test ederken boşlukta (hiçlikte) duran blokların rahatsız edici olduğu fark edildi. Bunun yerine deniz için su gölgelendiricisi yaparak oyuna hareket getirme kararı verildi. Deniz efekti olduğunda bir anda farklı hava durumları, su efektleri, balıklar, yağmur, kar gibi hava durumu efektleri eklenebilir bir hale geliyor. Daha sonra bu efektleri mekaniğe dahil edilebilir.



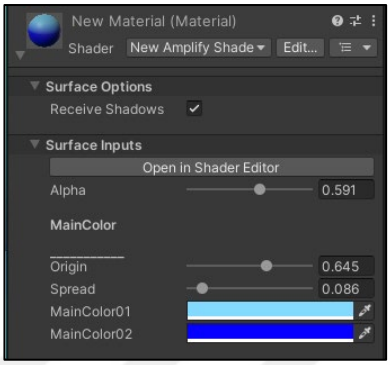
Şekil 56. Deniz gölgelendiricisi birinci deneme

İlk denemede suyun içinin görünmemesinin kapalı ve boğucu bir atmosfer yarattığı düşüncesiyle tasarım tekrar ele alınarak daha canlı renkleri olan ve içini görebildiğimiz deniz oluşturuldu. Denizin içi görülebildiği için su altına kayalık yerler ve balıklar eklendi. Bu da oyunun atmosferini hafifleştirdi. Adayı oluşturan küp blokların sert gözükmemesi için yapılan kenar yumuşatma yakalamak istenen elle çizilmişlik ve 3B görsellik hissini bozmaya başladığı için keskin köşeli modeller ile devam edildi. Ek olarak bloklarda doku değişikliğine gidildi. Sonuç olarak su gölgelendiricisinin eklenmesiyle proje boşlukta oynanan bir oyun olmaktan çıkarak görsel olarak tatmin edici bir ada fikrine doğru evirildi. Böylece su dalgalanması, rüzgâr, yağmur, kar yağması gibi efektler eklenebilecek bir ortam yaratılmış oldu.

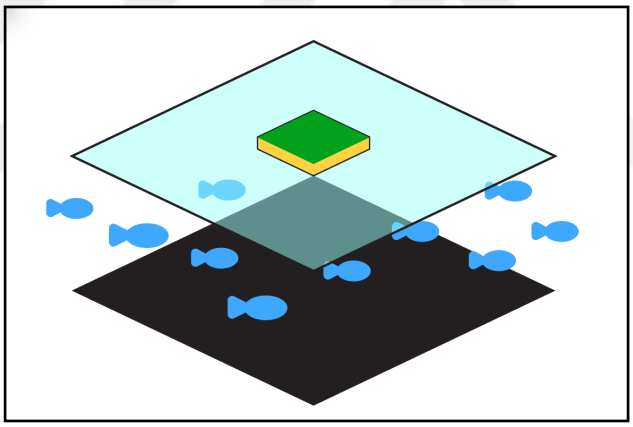
Şekil 57. Deniz gölgelendiricisi ikinci deneme

Mini Island projesinde hem oyunun içeriği hem görsel hissiyatı hem de görüntüde kapladığı alan açısından en önemli eklenti su gölgelendiricisidir. Su efekti üç ayrı parçadan oluşmaktadır. Su, Hareketli efektler (balık vb.) ve zemin. Bunun için biri birkaç farklı yerde kullanılmak üzere iki farklı gölgelendirici oluşturuldu. Bunlardan biri su gölgelendirici olarak kullanılırken diğer gölgelendirici su altındaki bütün modeller için kullanıldı. Su gölgelendiricisi, hava durumuna ve mevsimlere göre değişen deniz (ör. düz, dalgalı ve buzlanmış su) ve adadaki yaşam koşullarına (kar yağdığında tarım yapmanın zorlaşması vb.) ilişkin etkileri, görsel bir etkinin yanında oyuncunun oynarken dikkat etmesi gereken unsurları (ör. buzlanma oluştuğunda ekinlerinin zarar göreceğini fark etmesi gibi) görmesine yardımcı olacak şekilde mekanik olarak verebilmek için tasarlandı. Öncelikle iki tane üst üste düzlem oluşturup, üstekine su gölgelendiricisini alttakine ise istenilen rengin eklenebileceği bir gölgelendirici eklendi. Ara kısım için balıklar, kayalar veya gemi kalıntısı gibi modeller için ayrı bir gölgelendirici yapıldı. Gölgelendirici içerisinde opaklık değişkeni, iki farklı renk, y ekseninde iki rengin nerede olacağını belirleyecek bir değişken ve su altındaki bütün modellerin siluetinin istendiğinde değiştirilebilmesi amacıyla, iki renk arasındaki geçişin keskinliğinin

belirlenebileceği bir değişken eklendi. Bu işlemlerden sonra hem içerisindeki objeleri gösterip hem onları farklı kırılmalarla su hissi vermek için bir ayrı gölgelendirici oluşturuldu.

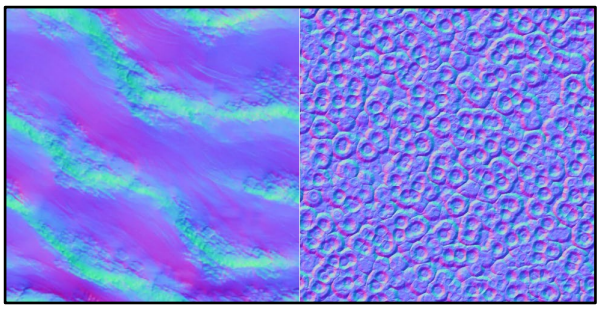


Şekil 58. Su altı gölgelendiricisi



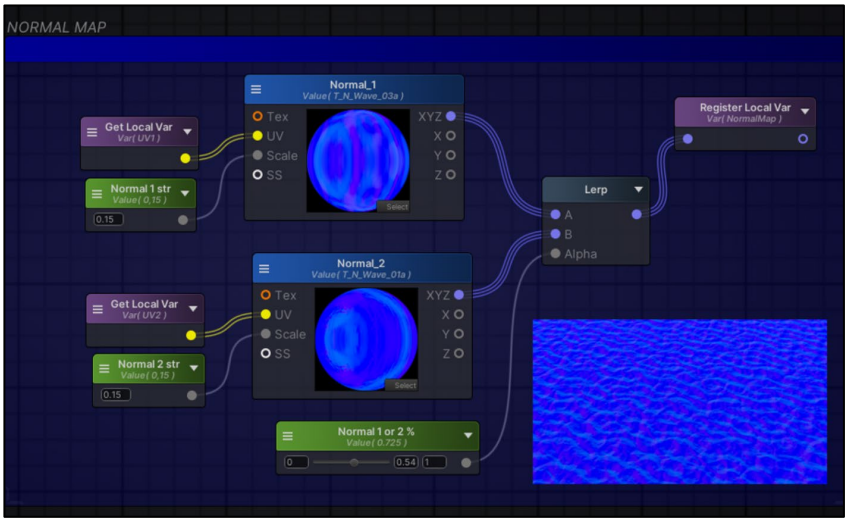
Şekil 59. Mekân yapılırken kullanılacak modeller

Su dalgalanması efekti için oyuna dışarıdan normal dokusu eklendi. Görüntü işleme motoru, aydınlatma hesaplamalarını gerçekleştirirken ağın gerçek yüzey normalleri yerine normal haritanın değerlerini okur. Bu sayede işleme sürelerini kısaltmaya olanak tanır. Normal içerisinde RGB renk değerleri vektörün X,Y,Z yönünü depolamak için kullanılır. Bunun yanında #8080FF modelin yüzeyinde hiçbir değişikliği temsil etmez.



Şekil 60. Dalga görüntüsü oluşturmak için kullanılan iki farklı dokuda normal map

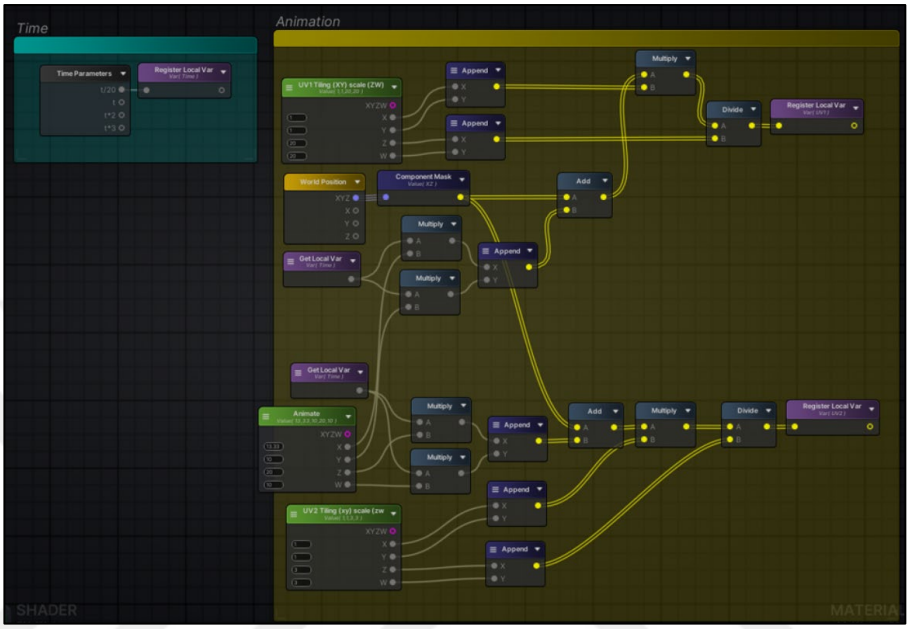
Mini Islan'da doğal bir dalgalanma hissi yaratmak için iki farklı normal doku (texture) kullanıldı. Bunun için normal haritaları doğrusal ara değer hesabı (Linear Interpolation - LERP) kullanarak bir değişkene bağlanır ve bu değişken, iki normal haritası arasında hangisinin daha baskın olacağını belirler. Böylece farklı değerler vererek çeşitli dalgalanma etkileri elde edilebilmektedir. Ayrıca hareket oluşturmak için zaman değişkeni ve dokunun baskınlığını değiştirmek için bir "Normal str" değişkeni eklenir. Doku düğümünün (texture node) içerisinde görünürlüğü değiştirme imkanı olduğu için sadece bir değişken eklemek yeterli olmaktadır. Ancak zaman ile değişecek bir animasyon oluşturmak için biraz daha düzenleme yapmak gerekir.



Şekil 61. Normal dokularının, gölgelendirici içerisindeki konumu

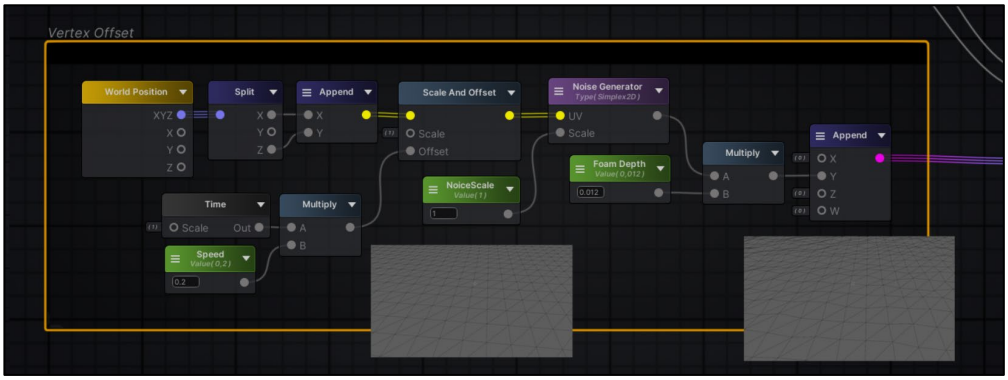
Zaman Parametreleri düğümü, Unity'nin dahili geçen süre çıktısını saniye cinsinden ve farklı sabit miktarlarla ölçeklendirilmiş olarak verir (Unity Products:

Amplify Shader Editor/Time Parameters - Amplify Creations Wiki). Bu çıktı, hız, büyüklük ve konumu yatay ve dikey olarak ayrı ayrı değiştirebilecek şekilde düzenlenir. Bu sayede, ihtiyaç duyulduğunda farklı efektler elde edilebilir.

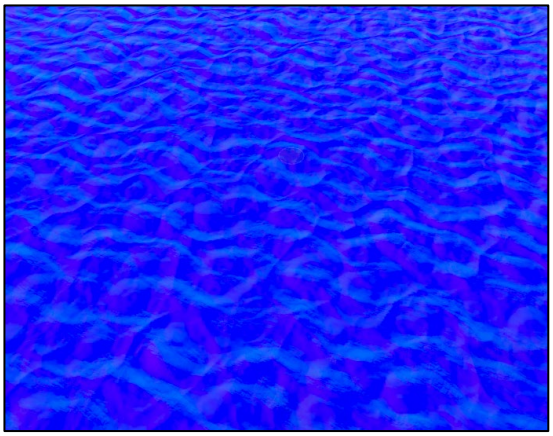


Şekil 62. Gölgelendirici içerisindeki animasyon bölümü

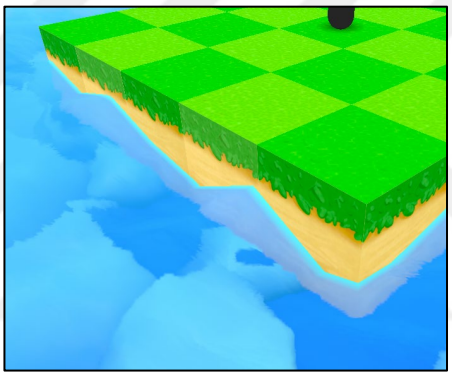
Dalgalanma efektini daha etkili kılmak amacıyla, modelin içindeki köşeler tepe noktası uzaklığı (vertex offset) kullanılarak hareket animasyonu elde edildi. Bu yöntem, özellikle gölgelendiricinin ada ile temas ettiği alanlarda etkili olmaktadır. Bu etkiyi oluşturmak için bir gürültü üretici (noise generator) kullanıldı. Dalgalanma efektinden sonra renk ve saydamlık özellikleri oluşturuldu.



Şekil 63.



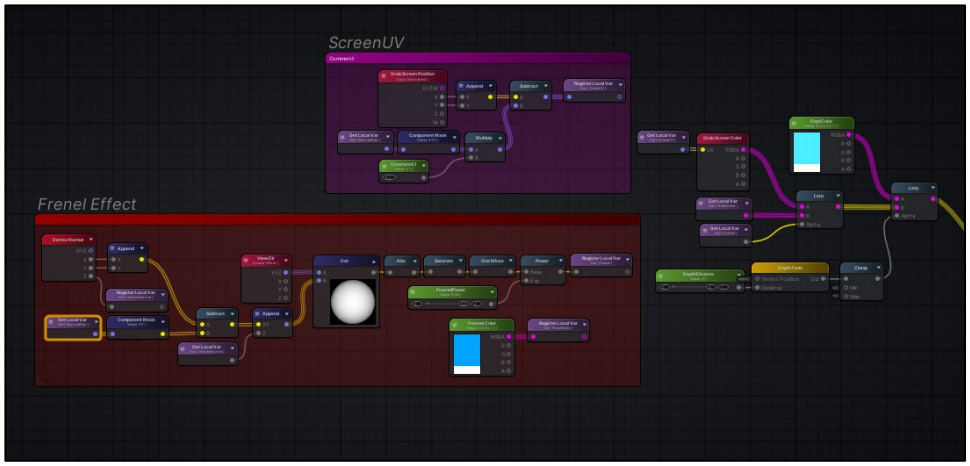
Şekil 64. Normal ve vertex offset bittiğinde gölgelendiricinin görünüşü



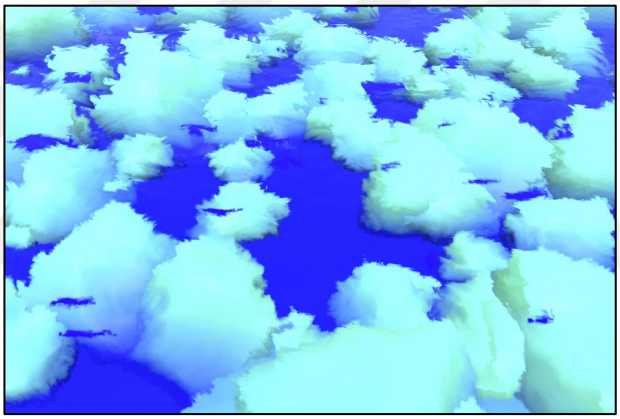
Şekil 65. Vertex Ofset Oyun içinde nasıl gözüküyor

Öncelikle normal map, frenel ve screen uv ile iletişimi olacak şekilde kullanılmaktadır. Fresnel efekti, yüzeyin normaline (yüzeye dik olan doğrultu) bakış açısına bağlı olarak yüzeyin yansıma ve kırılma oranlarının değişmesini tanımlar. Özellikle su, cam gibi yüzeylerde daha gerçekçi görsellik elde etmek için yapılır.

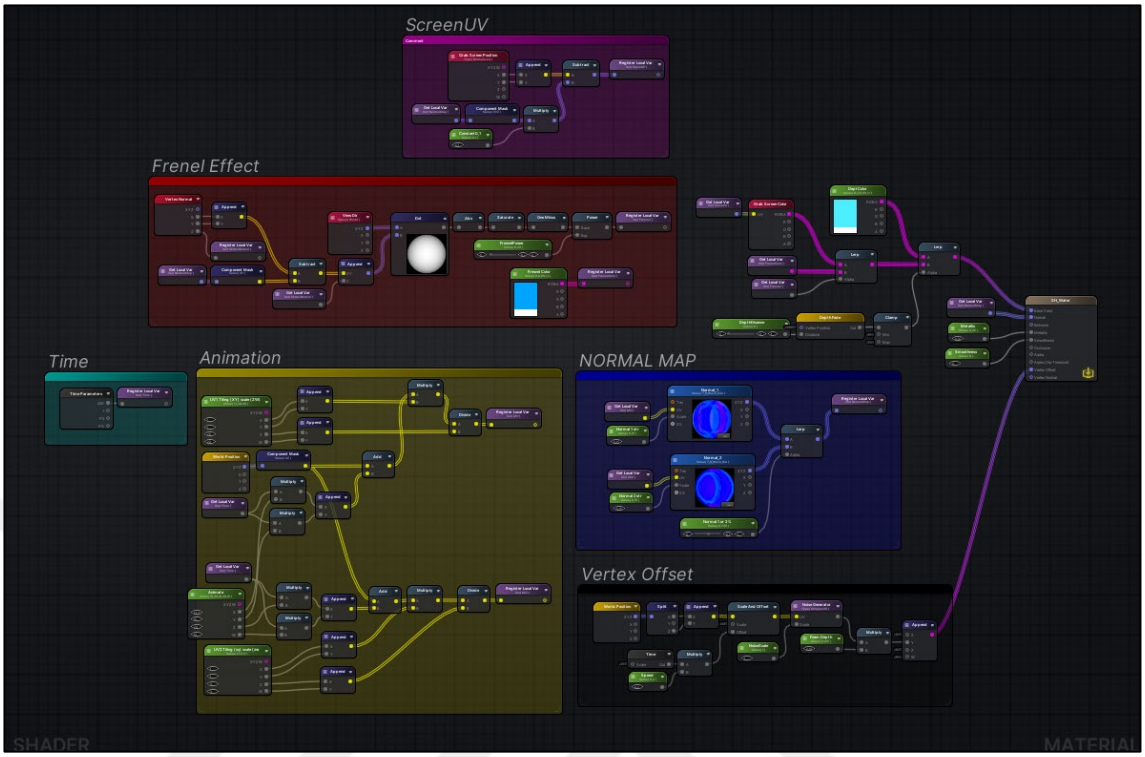
Şekil 66. Frenel efekt eklendiğinde gölgelendirici



Şekil 67. Gölgelendirici renk aşaması

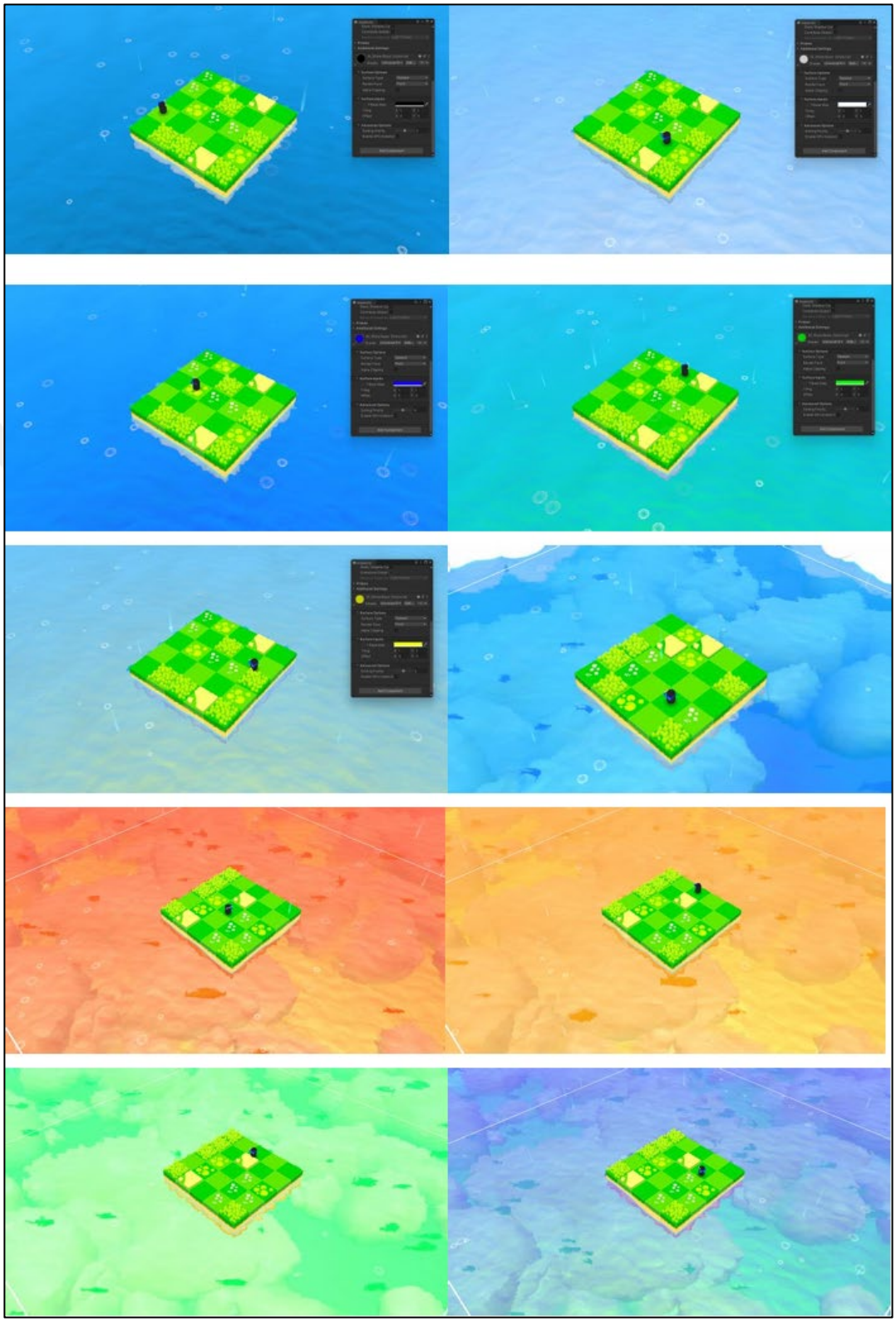


Şekil 68. Screen Uv

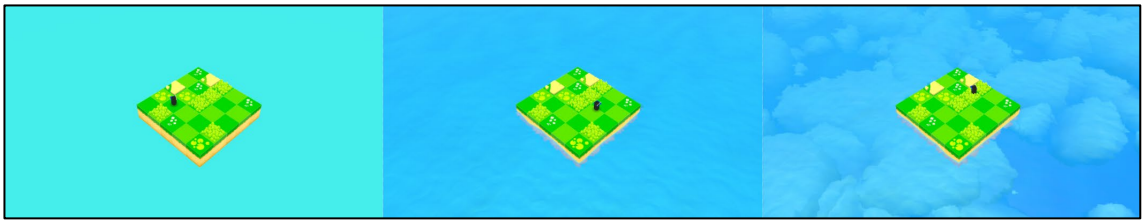


Şekil 69. Su gölgelendiricisi

Gölgelendirici içerisinde kullanılan değişkenler sayesinde farklı hava durumu, renk gibi çeşitli durumlar oluşturulabilmektedir. Bunun için materialler kullanılarak değişkenlerin değerleri kaydedilir.

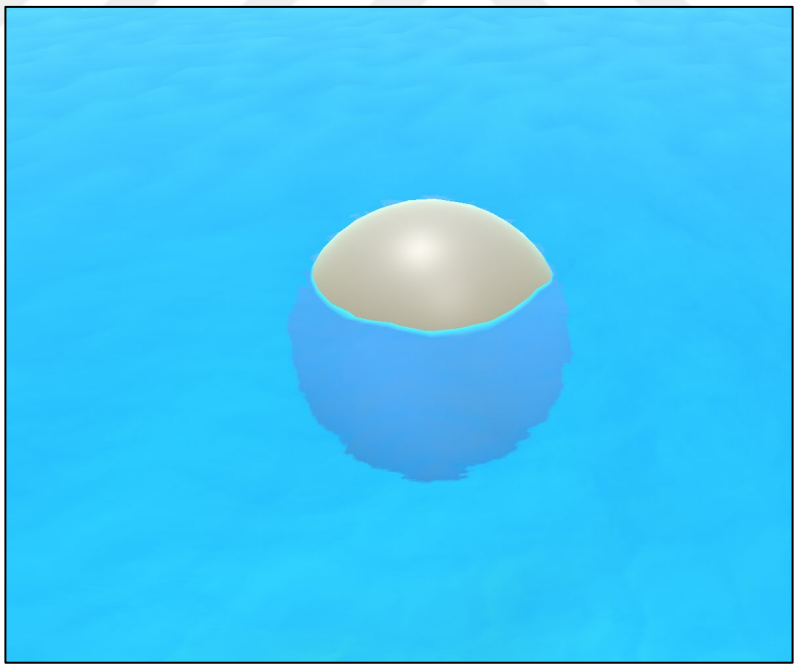


Şekil 70. Gölgelendirici içerisindeki bazı değişkenler ile ede edilen farklı etki örnekleri.



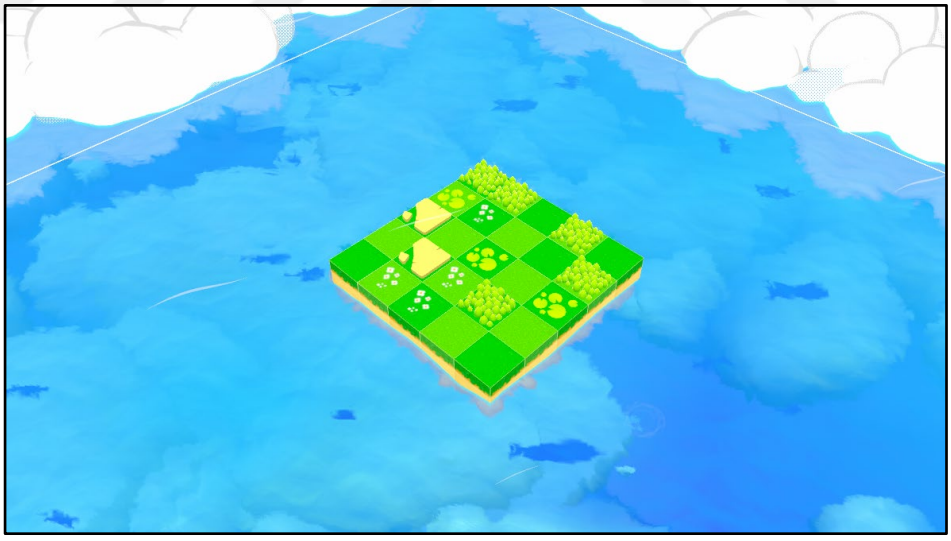
Şekil 71: Gölgelendirici aşamaları

Şekil 72: Yağmırlu ve yağmursuz hali



Şekil 73: Gölgelendirici model ile etkileşime girdiğindeki görünüş

Şekil 74: Su gölgelendiricisinin sahne içerisindeki görünüşü

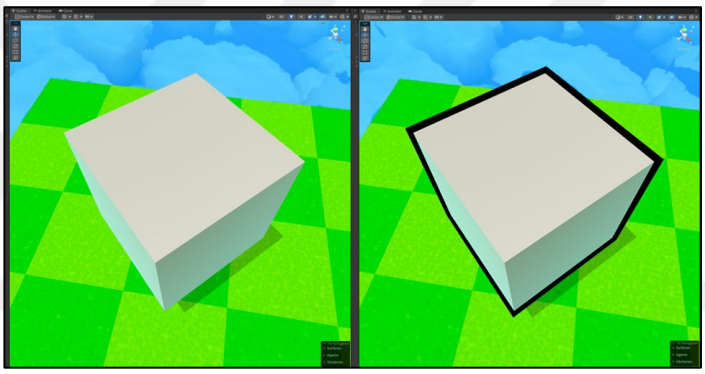


Şekil 75. Blokların değişimi ve bulutlar

Oyuncu ada dışında keşfedebileceği alanlar olarak bulutlar eklendi. Kamera hareketi yaparak göremeyeceği ancak oyunun ileriki seviyelerinde çeşitli olaylar (event)

vasıtasıyla farklı oyuncu olmayan karakterler ile karşılaşacağı ve farklı seçimlerde bulunacağı tasarımlar yapıldı.

Oyun geliştirme veya grafik tasarımında nesneleri vurgulamak, belirli bir stil veya estetik oluşturmak veya oyun mekaniğine ek özellikler eklemek için kontur gölgelendiricisi kullanılabilir. Örneğin, bir nesne etrafında bir kenarlık oluşturularak daha belirgin hale getirebilir veya özel bir efekt yaratılabilir. Bu, oyunculara önemli nesneleri veya karakterleri daha kolay fark ettirme veya oyun mekaniğini vurgulama şansı verir. Genellikle karakterler, nesneler veya diğer oyun öğeleri gibi 3B nesnelerin kenarlarını renkli veya siyah bir çizgiyle çevrelemek için kullanılır.



Şekil 76. Sol tarafta Mini Island 3B küp nesne kontursuz ve sağ taraf konturlu görünüşü

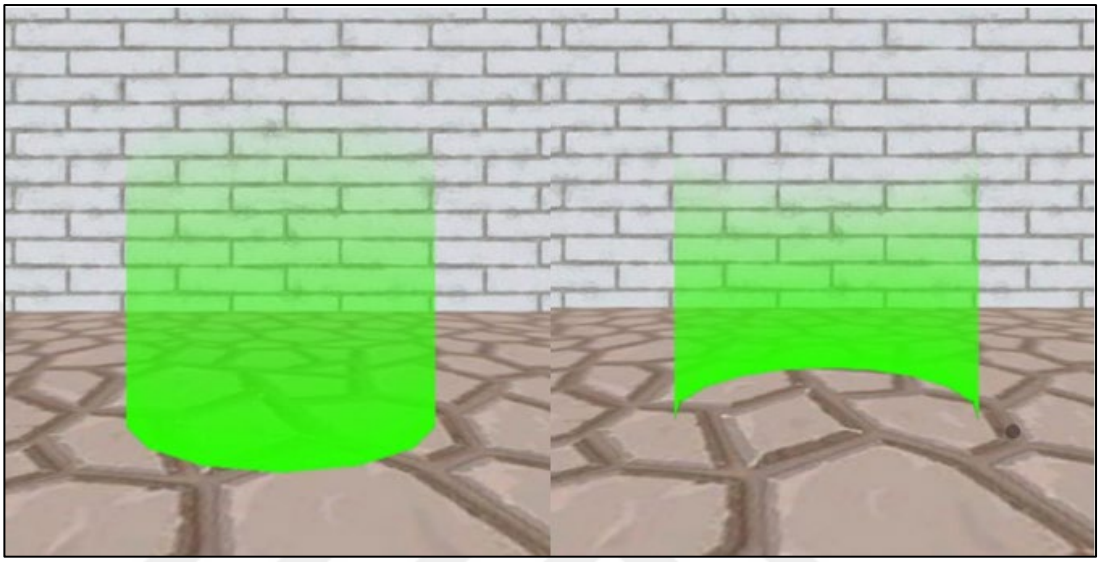
Kontur gölgelendiricisi, oyun içerisindeki 3B objelerin, arayüz (user interface) ile ilişkisini desteklemek ve modeller arasında oyuncunun doğru 3B objeyi seçtiğini fark etmesini sağlama işlevini de görür. Bu proje özelinde oyun içerisinde 3B objelere tıklandığında, üzerlerinde bir arayüz açılır. Bu arayüz oyuncuya oyun içerisinde kendi eklediği modelleri silme, kopyalama, döndürme vb. etkileşimleri yapma olanağı sağlar.

Şekil 77. Oyun içerisinde arayüz ile 3b modellerin ilişkilendirilmesi için kullanılır.

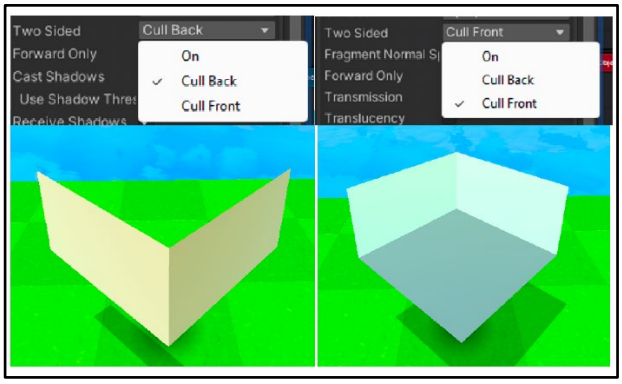
Mini Island için hazırlanan kontur gölgelendiricisi içerisine kontur kalınlığı belirleme, renk değiştirme, açılma kapanma animasyonu ve ihtiyaç olunması durumunda farklı yerlerde ve farklı renk ve kalınlıklarda kontur kullanabilmek için bir degrade efekt özelliği eklendi. Gölgelendirici, çalışma prensibi gereği eklendiği modelin ikinci materyali olarak işlev görür. İlk materyal objenin rengi, dokusu gibi nitelikleri verirken, kontur gölgelendiricisi ikinci materyal olarak eklenir. İş akışı bu iki materyali iki kere işler ve bu ikinci işlenme sırasında model büyütülüp normallerini tersine çevrilerek kontur etkisi elde edilir. Modeli seçtikten sonra inspector (denetim paneli) ekranından mesh renderer (örgü işleme) bileşeni içindeki artı butonuna tıklayarak ikinci bir materyal penceresi yaratılır ve buraya hali hazırda oluşturulmuş ve sonra oluşturulacak gölgelendirici ile ilişkili materyal eklenir.

Şekil 78. Bir model içerisine iki materyal eklemek için kullanılan arayüz

Bu işlemden sonra modelin normallerinin ters çevrilmesi gerekir. Bu şekilde ilk işlendiğinde modelin normalleri dışa dönükken ikinci işlemede içe dönük olur. İçe dönük kısım dışa göre her zaman arkada kalacağı için kontur efekti oluşur.



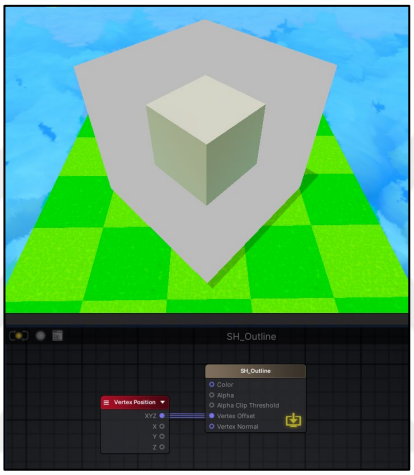
Şekil 79. Sol taraf cull-back, sağ taraf cull-front (https://forum.godotengine.org/t/how-to-double-face-culling-for-circular-effect/17394)



Şekil 80. Küpün işlenme modu (normal)

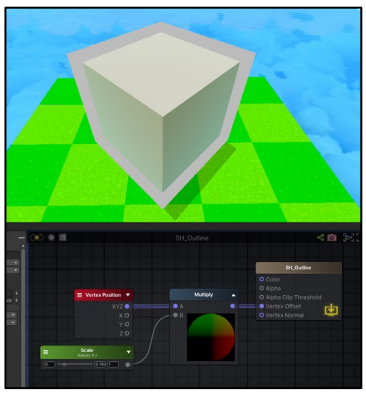
Gölgelendirici içerisinde vertex object kullanılarak modelin daha geniş olması sağlanır. Tepe noktası düğümü (vertex position node), nesne uzayında köşelerin konumunu verir. Bu veri doğrudan ağdan çıkarılır ve nesne orijinine göre köşe konumunu

içerir, bu da konum değerlerinin oyun nesnesinin (gameobject) sahip olduğu dönüşüm değeri ne olursa olsun değişmeyeceği anlamına gelir. Bu genellikle Yerel Tepe Noktası Ofseti çıktısında kullanılmak veya nesneye bağlı olan ve oyun nesnesinin konum, dönüş veya boyut gibi özellikleri değişse bile aynı kalan efektleri oluşturmak için kullanışlıdır. Bu düğümün kullanıldığı yere bağlı olarak (tepe noktası veya parça/yüzey işlevi) ya doğrudan bir değer (tepe noktası işlevlerinde) ya da köşeler arasında enterpolasyonlu bir değer (parça/yüzey işlevlerinde) döndürür. (wiki.amplify, 2024)

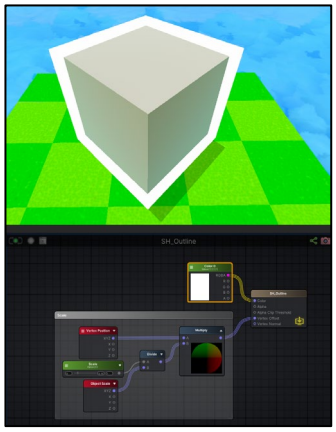


Şekil 81. Vertex position eklendiğinde

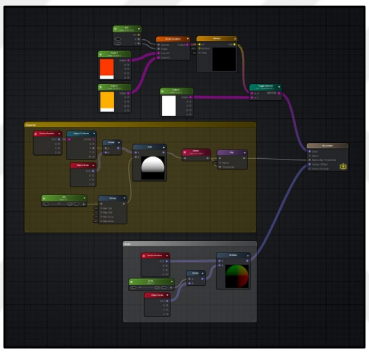
Projede, gölgelendiriciye eklenen değişken (float) ve çarpma (multiply) düğümleri (nodes) vasıtasıyla, modelin vertex konumu 0.5 ile çarpılarak (culling işlemleri sonucu ortaya çıkan) konturun kalınlığı belirlenmiştir.



Şekil 82. Tepe noktası konumu ile bölme işlemi yapılıp kalınlık ayarı eklendikten sonra



Şekil 83. Renk değişkeni eklendikten sonra

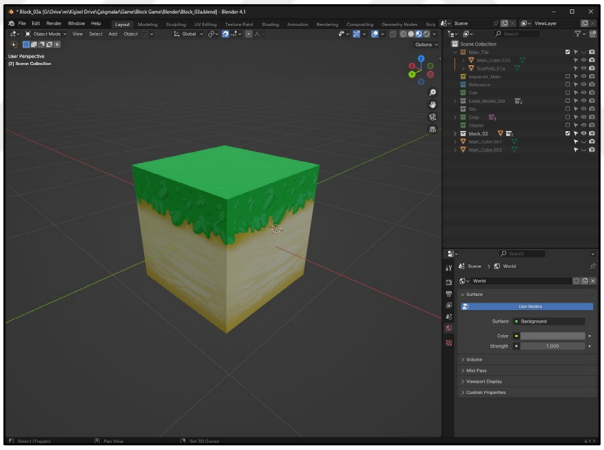


Şekil 84. Kontur gölgelendiricisinin son hali

3.3.4. 3B Modeller ve Arayüz Tasarımları

Oyun boyunca üretilen prototipler için ihtiyacımız olacak bazı ek sprite ve modeller gerekiyor. Hem gölgelendiriciyi kullanmak hem de oyuncunun mekanikleri anlayabilmesi için oyunun içerisine 3b ve 2b nesneler yerleştirmek gerekiyor. Oyun için 3b nesneler blender üzerinden yapıldı. Arayüz için yapılan tasarımlar ise Adobe illistrator kullanılarak yapıldı. Daha sonra 3B nesneler ve arayüz tasarımları Unity üzerinde birleştirildi.

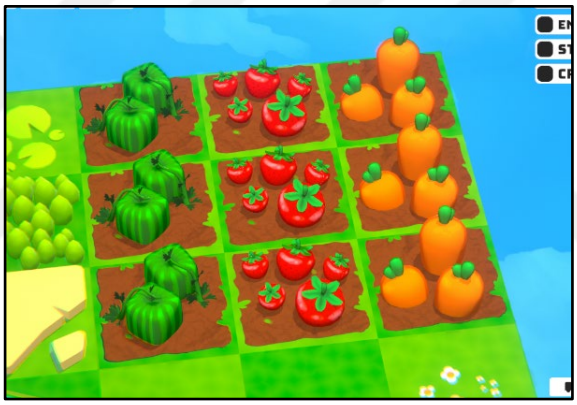
Oyun boyunca oyun motoru içerisine sürekli olarak model desteği sağlandı. Bunların içerisinde kimi zaman küpler gibi modüler modeller kimi zaman ise deniz gibi gölgelendirici üzerinden geliştirilecek ürünler için alt bölümlere ayrılmış bir kare eklendi. Modeller arasında binalar, ağaçlar, bitkiler, vb. bulunur.



Şekil 85. Blender içerisinde model görünümü

Modelleri üretilirken örgüler blender, dokular photoshop ve procreat kullanılarak üretildi. Bunun dışında modellerin oyuna uyum sağlayıp sağlayamadıklarını, sahnelerin nasıl göründüğünü, ışıklandırmanın modeller ve dokular üzerinde nasıl bir etki bıraktığını test etmek için Unity kullanıldı.

Şekil 86. 3b model

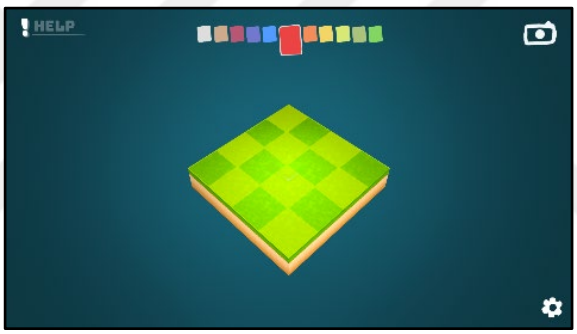


Şekil 87. 3b model

Şekil 88. 3b model

Arayüz tasarımları oyuncunun, oyun ile en çok iletişime geçtiği alandır. Özellikle inşa simülasyonu (building simulation) oyunlarında temel oynanış bir ikonu seçimi ve 3B nesneyi yerleştirmeye dayalı oluğu için arayüz tasarımları estetik ve mekanik açıdan oyunun en önemli öğelerinden biri haline gelir. Oyunlarda ayarlar, oyun içi, giriş ekranı ve bitiş ekranı vb. farklı arayüzlere ihtiyaç olabilir. Oyuncu bu ekranlar yardımı ile oyunu oynar.

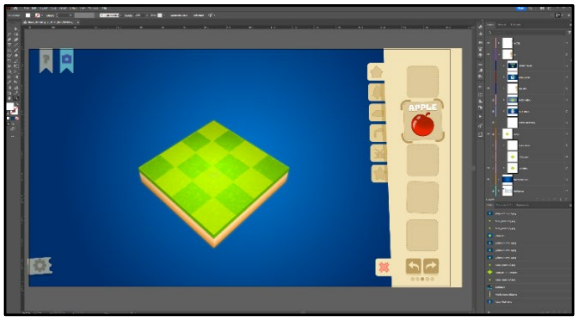
Mini Island oyunu için bu arayüz tasarımı yaparken giriş ekranı, oyun içi ekranı, ayarlar ekranı, fotoğraf çekme ekranı, albüm ekranı, pazaryeri ekranı, yetenek geliştirme ekranı gibi ekranlar üzerinde çalışıldı. Oyun çoğunlukla oyun içi ekranında oynandığı için öncelikli olarak oyun içi ekranı geliştirildi.



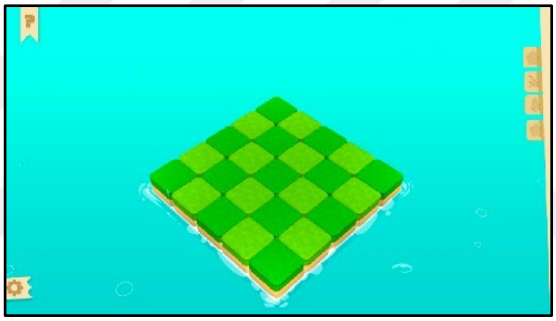
Şekil 89. İlk arayüz tasarımı

Başlangıçtaki arayüz tasarımı oyunun başlangıçtaki hali gibi daha sade bir yapıya sahipti. Birkaç buton ve bunların açtığı birkaç yeni ekrandan oluşmaktaydı. Bu tasarım oyunun temel işlevini yapabilmesi üzerine kurgulanmıştı. İçerisinde ayarlar butonu, fotoğraf butonu, yardım butonu, renk seçim kartları bulunmaktaydı. Bu tasarımda renk seçim kartları dışındaki bütün butonlar bir başka arayüz açmak için kullanılmaktaydı. Oyunun genel sadeliğine uygun olarak tek yapılması gereken fare yardımı ile ortadaki küplerin üzerine tıklayarak nesne eklemekti. Tam bu kısımda arayüz olarak ihtiyacımız olan bir model listesi söz konusudur. Model listesi genel olarak fare kullanarak tıkladığımızda hangi modeli yerleştirdiğimizi oyuncunun anlaması için kullanılır. Tasarımın başlangıç aşamasında fare aracılığıyla küplerin üzerine tıklandığında yerleştirilecek nesne görünmemekteydi. Model listesi eklendiğinde oyuncu

önce liste içerisinden bir model seçip sonra küpün üzerine tıklayarak yerleştireceği modeli görür hale geldi.

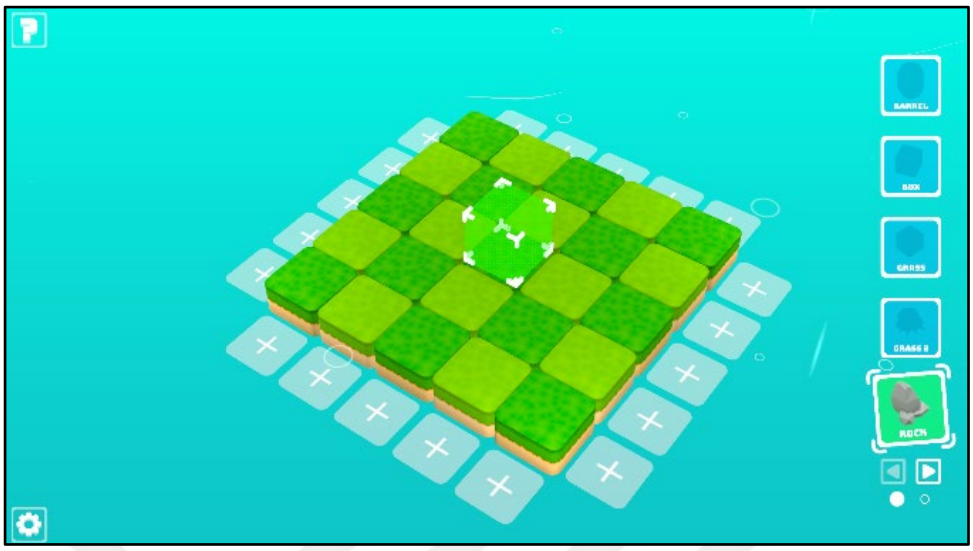


Şekil 90. Arayüz tasarımı II. Aşama tasarımı



Şekil 91. Arayüz tasarımı II. Aşama oyun içi görünümü

Arayüz tasarımının ikinci denemesinde nesneler klasör içerisindeki listeler gibi sıralandı. Bu şekilde daha çok nesne eklenebilir, gruplandırma yapabilir ve sayfalar arasında geçiş yapılır hale geldi. Ancak arayüzün oyun ekranında fazla yer kaplaması ve oyunun oluşturmak istediği ferah atmosferi engellemesi nedeniyle arayüz tasarımında değişikliğe gidildi.

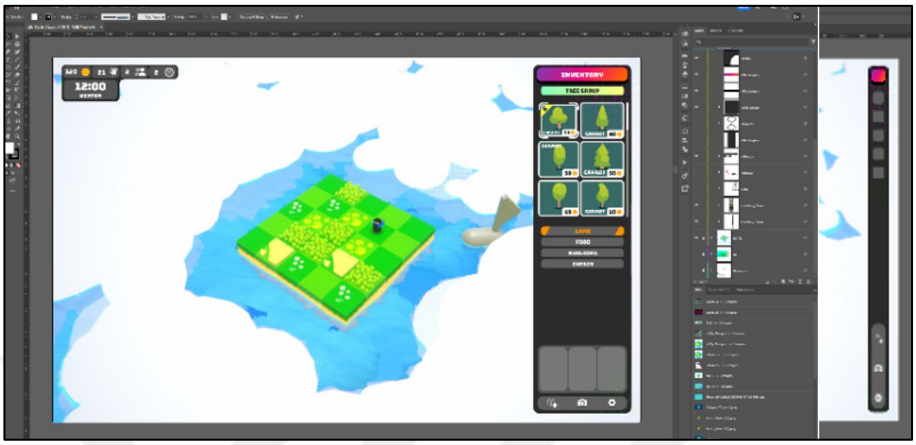


Şekil 92. Arayüz tasarımı III. aşama

Üçüncü arayüz tasarımı denemesi için oyunun görselliğini bozmayacak ve gereğinden fazla yer kaplamayacak bir deneme yapıldı. Öncelikle sahnenin içinde oluşan aurayı bozmamak için ekranı bölecek bir arayüz tasarımı yerine parçalı ve mekân ile ilişkili bir tasarıma geçildi. Oyun içinde beyaz rengini sadece arayüz üzerinde kullanılarak sade bir tasarım oluşturuldu. Hover (havada asılı) efekti korunarak seçim yaptığında renk değiştirme özelliği eklendi. Ancak bu tasarımdan alan içerisinde çok model bulundurma imkânı vermediği için vazgeçildi.

Şekil 93. Arayüz tasarımı IV. Aşama

Dördüncü arayüz tasarımı denemesi için telefon uygulamasını andıran bir üslup oluşturmaya çalışıldı.



Şekil 94. Arayüz tasarımı V. Aşama

Oyunun içerisine bulutlar eklendiğinde oluşan atmosfere benzer, html ve css kullanarak yapılmış bir internet sitesi gibi hissini veren bir üslup üzerine çalışıldı. Böylece yalnızca nesneler ve gruplar değil aynı zamanda ayarlar, fotoğraf ve blok ekleme modları da aynı alan içerisine dahil edildi. Bulutların beyazlığı ile kontrast oluşturmayı sağlarken bir yandan da siyah arka plan sayesinde oyun içerisine yerleştirilen nesnelerin konturları daha okunabilir bir hale gelmiş oldu. Sol üst köşedeki saat, mevsim, para gibi göstergeleri içeren arayüz öğesi içinde aynı üslup kullanıldı.

SONUÇ

İçinde yaşadığımız dünya analogdur. Algılayabildiğimiz tüm girdiler, sesler ve görüntüler analogdur; yani sürekli zaman ve değere sahip sinyallerdir. Bizler analog canlılar olarak analog sinyalleri algılar ve analog sinyalleri işleriz. Örneğin kalp atışımızı ölçmek, aktivitelerimizi takip etmek, analog sensör bilgilerinin işlenmesini gerektirir. Fiziksel dünyadan gelen ve onları algılayabilmemiz için fiziksel dünyaya geri dönmeleri gereken bilgiyi (information), ayrık zaman ve niceliksel büyüklüğe sahip sinyallerle temsil eden dijital elektronik cihazlarımız dahi bizimle ve kendi aralarında konuşurken analog arayüzlere ihtiyaç duyarlar. Bilgisayarlar kaynağında (fiziksel dünya) analog olan ve vericide ve alıcıda da analoğa dönüşmesi gereken bilgiyi (information) dijital formatta işler ve depolar.

Bu araştırmanın ana konusunu oluşturan shader (gölgelendirici) uygulamaları nokta, çizgi, iki ve üç boyutlu form, ışık-gölge, valör, renk ve doku gibi sanat ve tasarımın temel biçimlendirme öğelerinin, kavramlarının, araçlarının ya da olanaklarının dijital ortama, bilgisayarın yazılım diline çevrilmiş versiyonları olarak düşünülebilir. Örneğin bir vektör, kuyruğu koordinat sisteminin orijinine sabitlenmesi koşuluyla, bir noktayı temsil etmek için kullanılabilir. Oyun programcılarının büyük çoğunluğu "vektör" terimini hem noktaları (konum vektörleri) hem de vektörleri ifade etmek için kullanır. Bilgisayar dilinde vektör, uzayda hem büyüklüğü hem de yönü olan bir niceliktir. Bir vektör, kuyruk adı verilen bir noktadan baş adı verilen bir noktaya uzanan yönlendirilmiş bir doğru parçası olarak görselleştirilebilir. Bir 3B vektör, aynen bir nokta gibi üçlü skaler (x, y, z) ile temsil edilebilir. Noktalar ve vektörler arasındaki ayrım aslında oldukça hassastır. Teknik olarak, bir vektör bilinen bir noktaya göre sadece bir ofsettir. Bir vektör 3B uzayda herhangi bir yere taşınabilir -büyüklüğü ve yönü değişmediği sürece aynı vektördür. 2B ve 3B forma gelince, örgü (mesh), şekilleri bir bilgisayar için anlamlı olacak şekilde tanımlamak için sahip olduğumuz yöntemlerden biridir. Bir şekli tanımlamayan örgü üç şey hakkında bilgi içerir: köşeler (boyutsuz nokta), kenarlar (tek boyutlu uzamda çizgi) ve yüzler (iki boyutlu uzamda düzlem). Ve düzlemin üçüncü boyutta sürüklenmesinden oluşan 3B form, Gölgelendiriciler, analog tasarımın bu dört

unsurunun yanında renk, evrensel ışık-gölge ve doku gibi unsurları da bilgisayar ortamında üretme ve uygulama imkânı sağlarlar.

Indie (bağımsız) oyun türü içerisinde Sandbox olarak adlandırılan inşa simülasyonu (building simulation) kategorisine dahil Mini Island başlıklı oyun projesini yönlendiren temel motivasyon ve amaç, oyun tasarımı alanında çalışmak isteyen grafik tasarımcıları grafik ve bilgisayar tasarımı alanlarının kesişme noktasında bulunan gölgelendirici (shader) programlarını öğrenmeye ve kullanmaya teşvik etmek olmuştur. Bu proje, ilk çizimlerden kâğıt ve dijital prototiplerin, 3B modellerin ve arayüzlerin oluşturulmasına, denemelere, ardından tekrar çizimlere ve yeni prototiplerin, 3B modellerin ve arayüzlerin oluşturulmasına uzanan bir analiz, tasarım, uygulama, denememe döngüsü içerisinde gerçekleştirilmiştir. Görsel ve mekanik açıdan sade bir tasarımla başlayan oyun geliştirme süreci, müteakip aşamalarda modeller, arayüzler ve oyun mekaniği bakımlarından zenginleşen bir içeriğe evirilmiştir. Işık, renk, doku, malzeme ve animasyon etkileri için yapılan farklı gölgelendiriciler vasıtasıyla şeffaflık, opaklık, devinim vb. etkiler oluşturularak oyunun gerektirdiği oranda görsel ve mekanik inandırıcılık sağlanmaya çalışılmıştır.

Blok yerleştirmeye dayalı oyunun mekanik açıdan şu ana kadar geçirdiği gelişim altı aşamada şöyle özetlenebilir: Birinci aşamada sadece kamera açısını Q tuşu ile sola E tuşu ile sağa çevirerek elde edilen 90 derece dönüş imkânı söz konusudur. Bu durumda henüz sahneyi 360 derece açıdan görmek mümkün değildir. İkinci olarak fare ile her yöne döndürme ve yakınlaşma ve uzaklaşma özelliği oluşturulmuştur. Üçüncü aşamada adayı büyütmek için bir blok ekleme sistemi ve dördüncü aşamada modelleri seçmek için bir klasör arayüzü eklenmiştir. Beşinci aşamada, oluşturulan mekanların fotoğrafını çekmek için bir fotoğraf sistemi ve fotoğraflara bakmak için bir albüm oluşturulmuştur. Altıncı ve şu ana kadarki son aşamada oyunun ekonomisini oluşturmak için pazaryeri ve tarım yapmak için ekme ve toplama mekanikleri geliştirilmiştir. Mini Island oyunu bir parçası olduğu, temel olarak gölgelendirici programlarının konu alan tez çalışmasının sınırları içerisinde bir örnek olarak tamamlanmıştır. Ancak oyun, kendi türü içinde geliştirilmeye açık ve elverişlidir.

KAYNAKÇA

Bar-El, D., & Ringland, K. (2021). Teachers Designing Lessons with a Digital Sandbox Game: The Case of Minecraft Education Edition. *European Conference on Games Based Learning. Academic Conferences International Limited.*

Bereitschaft, B. (2016). Gods of the city? Reflecting on city building games as an early introduction to urban systems. *Journal of Geography*, 51-60.

Bond, J. G. (2018). *Introduction to Game Design, Prototyping, and Development.* Addison-Wesley.

Burnet, R. B. (2003). *Perspectives On Multimedia.* Sweden.

Burnett, R. (2003). *Perspectives on Multimedia Communication, Media and Information Technology.*

Charity, M., Rajesh, D., Ombok, R., & & Soros, L. B. (2020). Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment . *Say "sul sul!" to simsim, a sims-inspired platform for sandbox game ai*, (s. 182-188).

Claire Bond Potter, R. C. (2012). *Histories of Contemporary America.*

Dijk, J. v. (2005). *The Network Society Social Aspects of New Media.*

Documentation. (2023, 02 20). Unity3d: https://docs.unity3d.com/2023.2/Documentation/Manual/PostProcessingOverview.html adresinden alındı

Documentation. (2024, 02 06). Unity3d: https://docs.unity3d.com/Manual/render-pipelines-overview.html adresinden alındı

Dolan, P. R. (2021). 16-bit dissensus: post-retro aesthetics, hauntology, and the emergency in video games. 18.

Düvel, S. Z., & Oliver. (2004). *3D Game Engine Programming.* Boston: Premier Press.

Eberly, D. H. (1999). *3D GAME DESIGN: A Practical Approach to Real-Time Computer Graphics.*

Elliot, D. A. (2006). *The State of the Field, Edited by John Leslie King University of Michigan Kalle Lyytinen Case Western Reserve University, Scoping the Discipline of Information Systems.*

Espíndola, F., & Yeber, P. (2023). *The unity shaders bbbible.*

Fiadotau, M. (2018). Indie Game. *Encyclopedia of Computer Graphics and Games.* içinde Estonia: Springer.

Gregory, J. (2019). *Game Engine Architecture.* CRC Press.

Hebert, C., & Jenson, J. (2020). eaching with sandbox games: Minecraft, game-based learning, and 21st century competencies. *Canadian journal of learning and technology*, 46.

Hill-Whittall, R. (2015). *Indie Game Developer Handbook.* Newyork-London: Focal Press.

Is Every Indie Game Independent? Towards the Concept of Independent Game. (2016). *The International Journal of Computer Game Research*.

Keogh, B. (2019). From aggressively formalised to intensely in/ formalised: accounting for a wider range of videogame development practices. *CREATIVE INDUSTRIES JOURNAL* , 14–33.

Kleanthous, S., Christodoulou, D., Papadopoulos, G. A., & Samaras, G. (2018). *Towards Modelling the User Creative Process in a Sandbox Game.*

Kyle Halladay. (2019). *Practical Shader Development: Vertex and Fragment Shaders for Game Developers.* Bristol: Apress.

Leonard Herman, J. H. (2002). *The History of Video Games.*

Manovich, L. (2001). *The Language of New Media.* London.

Nicoll, B., & Keogh, B. (2019). *The Unity Game Engine.*

Ocio, S., & Brugos, J. A. (2009). Multi-agent Systems and Sandbox Games.

Reed, E. (2020). The Influence Of Software Tools On Game Development Communities And Aesthetics. *Indie games in the digital age* (s. 119). içinde New York: Bloomsbury Publishing Inc.

Rokošný, I. (2018). Digital Games as a Cultural. *Acta Ludologica*, 48-61.

Rose, M. (2011). *Indie Games You Must Play.* CRC Press .

Sicart, M. (2023). *Playing Software.* england: The MIT Press.

Siegel, j. K. (2005). *THE VEST POCKET GUIDE TO INFORMATION TECHNOLOGY.*

Thon, J.-N. (2020). Analyzing Indie Aesthetics. *DiGRA*.

Tracy Fullerton, C. S. (2004). *Game Design Workshop: Designing, Prototyping, and Playtesting Games.* Focal Press.

wiki.amplify. (2024, 06 05). https://wiki.amplify.pt/index.php?title=Unity_Products:Amplify_Shader_Editor/Vertex_Position adresinden alındı

Williams, A. (2017). *History of Digital Games: Developments in Art, Design and Interaction* .

ÖZGEÇMİŞ

Ad, Soyad: Muharrem Deniz Çiçek

Eğitimi:

Yüksek Lisans: 2020, Dokuz Eylül Üniversitesi, Güzel Sanatlar Fakültesi, Grafik Anasanat dalı (Devam ediyor)

Lisans: 2015, Dokuz Eylül Üniversitesi, Güzel Sanatlar Fakültesi, Grafik Tasarım Bölümü, Grafik Resimleme ve Baskı Anasanat dalı

Lise: 2010, Ümran Baradan Güzel Sanatlar Lisesi

İş Tecrübesi:

2018 Jeton Studio, İzmir (Staj çalışması)

2019 Blay Games Oyun Stüdyosu, İzmir (Staj çalışması)

2020 – 2023 Blay Games Oyun Stüdyosu, İzmir