SON RAPOR

Unity Oyun Motoru ile Windows Tabanlı 2D Platform Oyunu

171180068 Mehmet Oğuz TOZKOPARAN

161180029 Ramazan Emre ERKAN

BM495 Bilgisayar Projesi I

Ocak 2021

İçindekiler

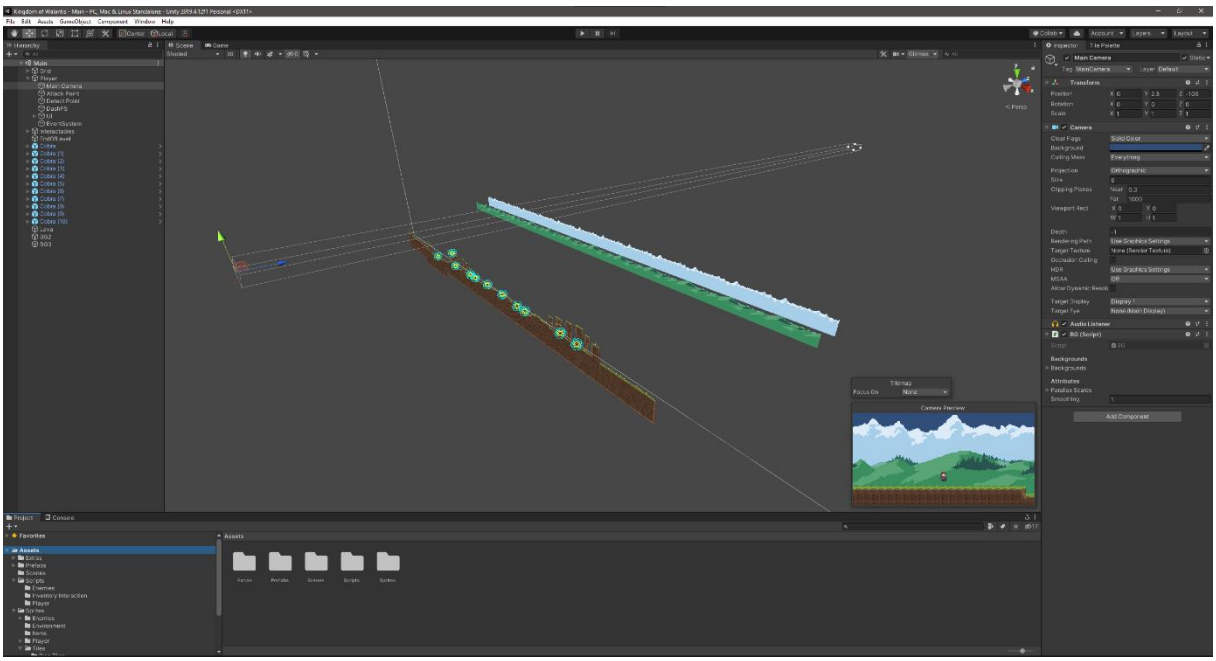
Şekillerin Listesi

ÖZET

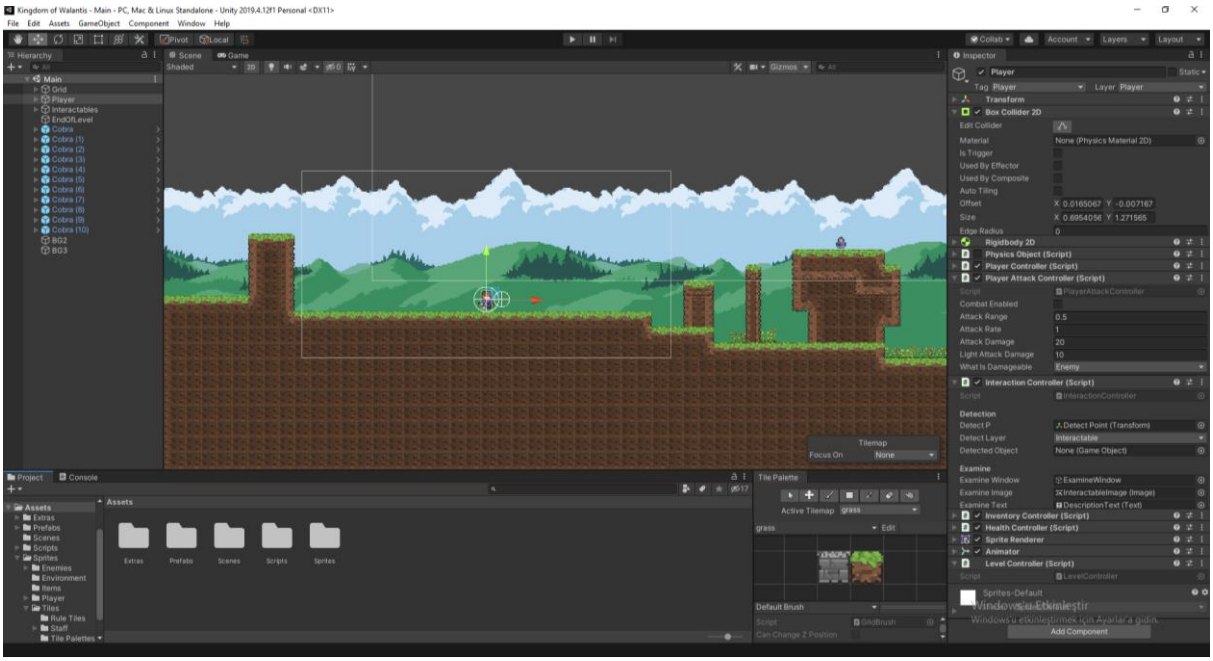
Bir oyun projesi geliştirmek sanat ve mühendisliğin birleşiminden oluşmaktadır. Oyun geliştirmek son zamanlarda oldukça popüler bir alandır. Çok büyük şirketler büyük ve uzun çaplı oyun geliştiriciliği yapmaktadırlar. Fakat son zamanlarda bağımsız geliştiricilerin başardıkları oyun projeleri insanların daha çok ilgisini çekmektedir. Bağımsız geliştirilmiş olan oyunlar büyük şirketlerin onlara olanak sağlayıp yatırım yapmasını sağlayabilecek bir hale gelmiştir. Özellikle son yıllarda geliştirilen birçok bağımsız oyun Unity Oyun Motoru 'nu kullanmaktadır. Bu sebepler doğrultusunda ve oyun geliştiriciliği öğrenmek amacıyla Kingdom of Walantis, Unity Oyun Motoru ile Windows Tabanlı 2D Platform Oyunu olarak geliştirilmiştir.

1. GİRİŞ

Bu proje Unity Oyun motoru kullanılarak, 16 piksel resimler aracılığıyla yazılmış, 2 boyutlu bir piksel oyunudur. Unity motorunun 2 boyutlu oyun geliştirme kütüphaneleri kullanılmıştır. Bu sayede Z eksenini kullanmadan sadece X ve Y ekseni ile bir harita oluşturulmuştur. Kamera oluşturulan haritaya tam karşıdan bakacak şekilde yerleştirilmiştir. Arka plan ise haritanın arkasına yerleştirilmiştir. Sonuç olarak kamera, harita ve arka plan; X ve Y ekseni üzerinde paralel olacak şekilde Z ekseninde derinlikli olarak sıralanmıştır. Oyunun 3 boyutlu ve 2 boyutlu görünümü aşağıda belirtilmiştir.



Şekil 1. 2 Boyutta nesnelerin yerleşimi



Şekil 2. Unity arayüzü

Haritada olduğu gibi geri kalan her nesne, tasarım veya özellik 2 boyutlu olarak oluşturulmuştur. Karakter ve düşmanlar art arda 16x16 piksel resimlerin ekranda görüntülenmesi sonucu sanki hareketliymiş gibi gözükmeleri sağlanır. Bu işlemi Unity programının animatörleri gerçekleştirir. Yalnızca piksel resimlerle bir oyun oluşturmak elbet ki mümkün değildir. Haritanın içerisinde karakter nesnesinin aktif olarak hareket edebilmesi ve diğer nesneler ile etkileşime girebilmesi için gerekli kodların yazılması gereklidir. Bu bağlamda C# dili ile birbirinden farklı görevlere sahip birçok farklı sınıf kodlanmıştır. Bu kodların sınıflarının birbirleriyle uyum içerisinde çalışmaları sonucu ortaya bütün bir oyun çıkmıştır. Yazılan C# kodlarının belirli bir kısmı Unity oyun motorunun C# dili için geliştirdiği kütüphaneler yardımıyla geliştirilmiştir.

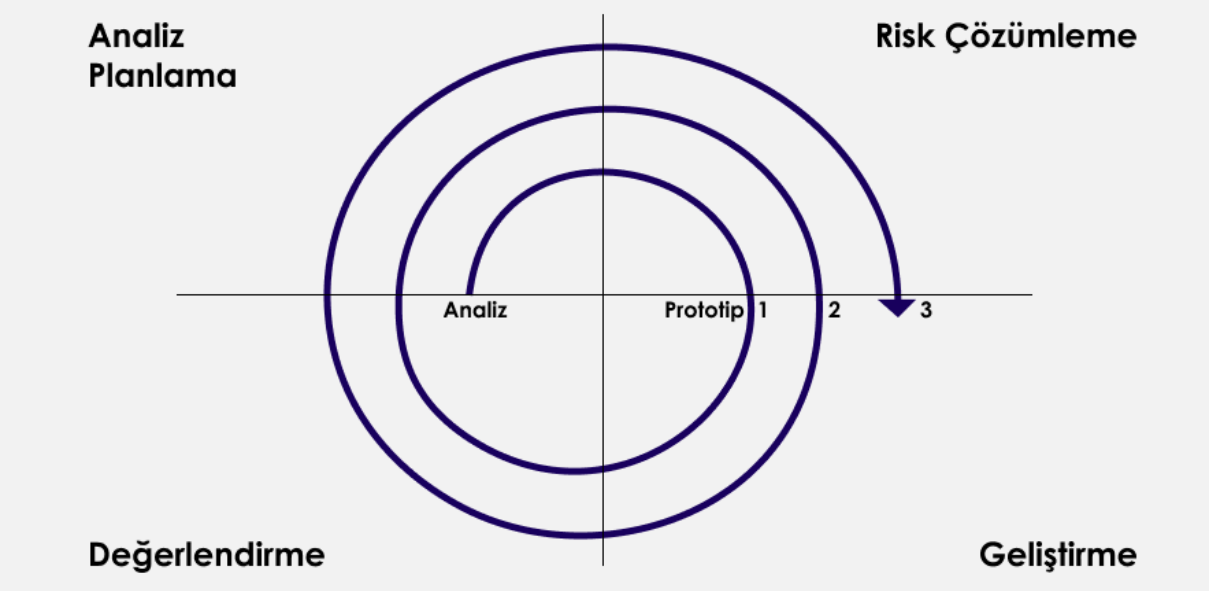
Oyun içerisinde kullanılacak olan görsellerin bir kısmı geliştiriciler tarafından piksel resim çizmeye yarayan programlar sayesinde oluşturulurken diğer bir kısmı, piksel resimler çizmek iyi derecede resim bilgisi isteyen bir sanat olduğundan, bazı profesyonel artistlerin internette ücretsiz olarak paylaştıkları resimlerdir. Hazır alınan bu kaynaklar, raporun ilerleyen kısımlarında belirtilecektir.

2. SİSTEM TASARIMI

2.1. Tanım

2.1.1. Yazılım Süreç Modeli

Projenin hayata geçirilmesinde spiral süreç modeli uygulanmıştır. Şekil 3 ’te modelin görsel olarak anlatımı sunulmuştur.



Şekil 3. Spiral süreç modeli

2.1.2. Tasarım

Projenin tasarımı 2 aşamada gerçekleşmiştir. Projenin ilk aşamasında hikayenin tasarımı ve kurgulanması gerçekleşmiştir. Daha sonraki aşama da ise oynanış ve oyun öğeleri tasarımı hikayeye bağlı kalınarak gerçekleşmiştir.

2.1.3. Kodlama

Projemizin kaynak kodları GitHub platformunda özel bir depoda kaydedilmiştir. Geliştiriciler Git ile erişip kendi bölümlerindeki eklemelerini ve versiyon güncellemelerini yüklenmesi kararlaştırılmıştır. Proje Unity 2D oyun motorunda C# programlama dili ile geliştirilmiştir.

2.1.4. Test

GitHub ile entegre bir şekilde geliştiricilerin sorunsuz bir şekilde aynı anda kod geliştirebileceği bir sistem oluşturulduktan sonra bir geliştirici tarafından yapılan güncellemelerin sorunsuz bir şekilde diğer geliştiriciye de etki etmesi için *“TestField”* adlı bir deneme projesi oluşturulmuştur. Yapılan değişiklikler, eklenen yenilikler GitHub aracılığıyla

depoda kaydedilip güncellemelerin bütün geliştiricilerde sıkıntısızca çalıştığından emin olunduktan sonra asıl oyunun geliştirileceği proje olan “Kingdom of Walantis” projesi açılıp kaynak kodları GitHub’a eklenmiştir. Oyun bu proje üzerinde geliştirilmiştir.

2.2. Rol ve Sorumluluklar

Projenin geliştirilmesinde takım üyeleri ortak çalışacaktır. Bu sebeple görevlerde her iki üye yer almaktadır. Takım üyeleri görevlerin alt işlemlerinde farklı konularda yardımlaşarak çalışacaktır.

2.3 Araç ve Teknikler

Projede, kullanılacak olan teknolojilere en stabil sürümler seçilerek karar verilmiştir. Proje, Unity Oyun Motoru ile C# programlama dili kullanılarak geliştirilmiştir. Unity versiyonu 2019.4.12f olarak seçilmiştir. Buna ek olarak programlama Visual Studio 2019 üzerinden yapılmıştır. Arayüz ve grafik tasarımları Aseprite uygulaması üzerinden tasarlanmıştır.

2.4 Araç ve Teknolojilerin Temini

Kullanılan Unity Oyun Motoru, C# programlama dili ve Visual Studio 2019 geliştirme ortamı ücretsiz olarak temin edilmiştir. Aseprite uygulamasının lisansı hali hazırda takım üyelerinde mevcuttur.

3. HİKAYE TASARIMI

3.1 Tanım

Hikaye, oyun projelerinde, projenin temeli olarak kabul edilebilir. Hikaye, hitap ettiği oyuncunun kendini oyuna daha fazla dahil olmasını ve oyuna dalmış hissetmesine yardımcıdır. Hikaye, oyuna anlam kazandırır ve oyuncunun yapması gerekenleri oyuncuya sunar. Oyuncu, oyunu oynayarak ana karakterin başarılarını ve başarısızlıklarını deneyimler. Bu yüzden yapılacak olan oyun projesinde hikaye çok önemlidir.

3.2 Hikayenin Geliştirilmesi

Hikaye geliştirilirken kullanılan öğeler tamamen hayali öğelerdir ve gerçeklikle bağlantısı yoktur. Hikaye geliştirilirken eski orta çağ zamanlarından ve Avrupa krallıklarından esinlenilmiştir. Ana karakter savaşçı *“Eliot”*, babası İmparator *“Bernard”*, kardeşi *“Lancelot”* olarak seçilmiştir.

3.3 Hikaye

Orta çağda *"Walantis"* imparatorluğu şanlı bayrağını kıtanın tamamında dalgalandırmak adına büyük bir adım atmaktaydı. Bunun üzerine gözünü hırs bürümüş İmparator "*Bernard"*, kıtaya yayıldıkça yönetimin zorlaşmaması için topraklarını 11 hanedanlığa böldü ve her bir hanedanlığı kendisine bağladı.

Bu sırada imparatorun iki erkek çocuğu oldu. Babası kendi yozlaşmış fikirlerinde kullanabilmek için oğulları *"Lancelot"* ve *"Eliot"* kendisinin de orada eğitim aldığı gizli bir dini tarikat olan *"Faith of Balance"* 'a yolladı. Eğitimler acımasız ve ağırdı. Sadece çelik gibi iradesi ve sarsılmayan inancı olan birinin geçebileceği sınavlardan geçtiler. Zaman ilerledikçe *"Lancelot"* bazı şeyleri sorgulamaya başladı ve buraya geliş amacını fark etti. Onun bu aydınlanmasından *"Faith of Balance"* da haberdardı. Ne yazık ki *"Lancelot"* ormandaki bir eğitimde şüpheli bir şekilde kayboldu ve bir daha ondan haber alınamadı. Kardeşi "*Eliot*" ise bu karanlık olayın perde arkasından bihaber bir şekilde eğitimlerini tamamlayıp farkında olmadan babasının olmasını hayal ettiği canavar haline geldi.

"Eliot" eğitimini tamamlamış ve tüm kıta ele geçirilmiş olmasına rağmen bir türlü doyuma ulaşamayan İmparator *"Bernard"*, gözü halkın varlıklarına dikti ve hanedanlıkların yönettiği halktan korkunç vergiler almaya başladı. Birkaç verimli toprağa sahip veya ticaret merkezi bulunan hanedanlar hariç çoğu hanedanlık fakirleşti. Verimli topraklarını hasat ederek vergilere rağmen gücüne güç katan adil ve eşitlikten yana olan, halkına önem veren 3 hanedan imparatorun bu davranışından memnun değillerdi ve artık işlerin değişmesi gerektiğine karar verdiler. Bunun için bazı devlet meselelerinde imparatora karşı çıktılar. Bu durumdan rahatsız olan imparator "*Faith of Balance"* tarikatından bu meseleyi sessizce çözmesini ister ve tarikat imparatora oğlunu göndermesini tavsiye eder. Bu şekilde *"Eliot"* kaderi şekillenmiş olur ve uzun bir serüvene başlar.

4. OYNANIŞ VE OYUN ÖĞELERİ TASARIMI

Oyun temelde klavye ve fare ile oynanmaktadır. "A" ve "D" tuşu ile sağa ve sola hareketi sağlanırken "Boşluk" tuşu ile zıplama hareketi gerçekleştirilir. Oyunda bulunan çift zıplama özelliği ile ilk zıplamadan sonra havadayken ikinci bir zıplama gerçekleştirilir. "Tab" tuşu ile oyun içi karakter envanter arayüzüne ulaşılır. "J" ve "K" tuşları temel saldırı tuşlarıdır. "J" tuşu ile hafif saldırı yapılabilirken "K" tuşu ile ağır saldırı gerçekleştirilir. "L" tuşu ile "Dash" özelliği aktive edilir. Bu özellik sayesinde karakterin hızı kısa bir süre boyunca artar. Böylece

sadece zıplama kullanılarak ulaşılamayacak yerlere ulaşılabilir ya da düşmanların saldırılarından kaçınılabilir. Bu özelliğin kullanılabilmesi için önceki kullanılma zamanının üzerinden belirli bir süre geçmiş olması gerekir.

4.1 Envanter ve Etkileşecek Nesneler

Oyunda karakterin etkileşime geçebileceği iksirler ve benzeri eşyalar vardır. Bu eşyalardan bazıları karakter tarafından tüketilebilirken bazıları da yalnızca eşya olarak kullanılabilir. Kuşanılan veya tüketilen nesnelerin özellikleri, karakter yeni bölüme geçtiğinde etkilerini yitirmez. Nesneler kullanılmayıp envanterde bekletildiği takdirde karakter yeni bölüme geçtiğinde envanterden silinmez. Envanter, 16 birimlik bir eşya alanına sahiptir ve her bir eşya 1 birim alan kaplamaktadır. Alınan eşyaların farklı özellikleri mevcuttur. Aşağıda iki ana başlık ve alt başlıkları olarak açıklanmaktadır:

1. **Etkileşim türü (Interaction Type):** Etkileşim türü, eşyaların oyuncu ile nasıl etkileşeceğine karar verir. Bu özellik üç alt başlıkla açıklanır.
    a. **Türsüz (NONE):** Türsüz etkileşim türü, eşyanın oyuncu ile etkileşime girmeyeceğini işaret etmektedir. Hatalar alınmamak için bu tür var olmak zorundadır.
    b. **Alınabilir (Pick Up):** Alınabilir etkileşim türü, eşyanın oyuncu tarafından envantere alınabileceğini belirtmektedir.
    c. **İncelenebilir (Examine):** İncelenebilir etkileşim türü, eşyanın oyuncu tarafından incelenip istenilen olayları aktif edebilmesini sağlar.
2. **Eşya türü (Item Type):** Eşya türü, eşyaların envantere alındıktan sonra hikaye eşyası veya tüketilebilir bir eşya olduğunu belirler.
    a. **Statik (Static):** Statik eşya türü, eşyanın envanterde tüketilemez bir eşya olduğunu belirtir. Bu tarz eşyalar hikaye belirten eşyalardır.
    b. **Tüketilebilir (Consumable):** Tüketilebilir eşya türü, eşyanın envanterde tüketilebilir olduğu bilgisini eşyaya kazandırır. Can iksirleri, güç iksirleri, atılma (dash) iksiri ve güçlendirme kartları bu türe örnektir.

Oyunda 11 adet eşya bulunmaktadır. İksir eşyaları geçici güçlendirmeler sunarken kart eşyaları kalıcı güçlendirmeler sunmaktadır. Bu eşyalar sırasıyla aşağıda açıklanmıştır.

1. **Can İksirleri (Poiton of Health, Large Potion of Health):** Bu eşyalar can azaldığında kullanıldıklarında oyuncuya can puanı sağlarlar.

2. **Güç İksirleri (Potion of Power, Large Potion of Power):** Bu eşyalar kullanıldıkları zaman oyuncunun vuruşlarını güçlendirerek düşmanlarına daha fazla hasar vurmasını sağlar.
3. **Atılma İksirleri (Potion of Dash, Large Potion of Dash):** Bu eşyalar kullanıldıkları zaman oyuncunun atılma bekleme süresini kısaltmaktadır.
4. **Güçlendirme Kartları (Card of Armor, Card of Power, Card of Dash):** Bu eşyalar oyuncuda büyük güçlendirmeler oluşturmaktadır. Bu güçlendirmeler Unity 'nin sunduğu PlayerPrefs ile kalıcı güçlendirmeler olarak tasarlanmıştır.
    a. **Zırh Güçlendirme Kartı (Card of Armor):** Bu eşya kullanıldığı zaman oyuncunun zırhını arttır. Böylece oyuncun aldığı hasarlar azalmaktadır.
    b. **Güçlendirme Kartı (Card of Power):** Bu eşya kullanıldığı zaman oyuncunun vuruşları yüksek miktarda artmaktadır. Böylece oyuncu gittikçe güçlenmektedir.
    c. **Atılma Güçlendirme Kartı (Card of Dash):** Bu eşya kullanıldığında kullanıcının atılma bekleme süresi iksirlere göre daha yüksek miktarda kısalmaktadır. Bu sayede oyuncu daha kısa sürede atılma kullanabilecektir.
5. **Hikaye Parşömenleri (Scroll of Walantis):** Bu eşyalar envanterde hikaye belirten eşyalardır. Herhangi bir güçlendirme veya kullanma söz konusu değildir.

Şekil 4. Oyun eşyaları

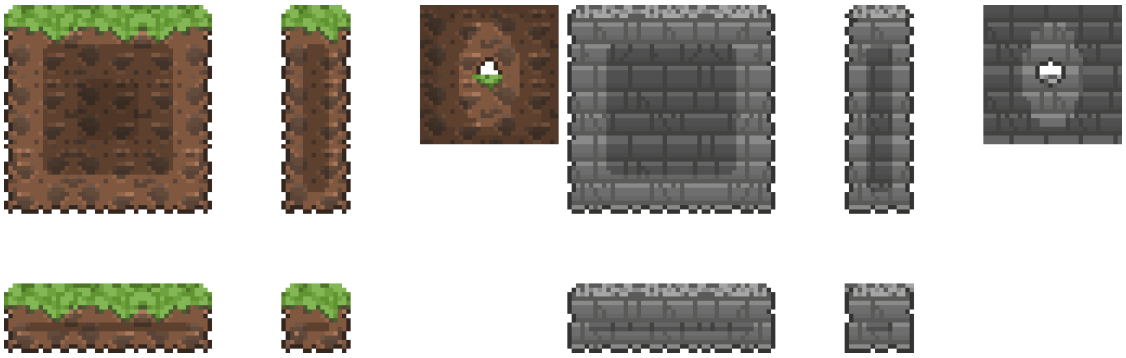
4.1.1. Arayüz Tasarımı

Envanter ve genel arayüz tasarımı için grafik tasarımları yapılmıştır. Oyuncunun canını oyuncuya sunmak için geliştirilen can barı ve can yüzde kısmı, envanterdeki her bir eşya alanını belirtmek için geliştirilen tasarımlar Şekil 5 'te sunulmaktadır.

Şekil 5. Can barı ve eşya arka planı

4.2 Zemin ve Arka Plan Tasarımı

Oyunun zemin grafikleri *"Tile"* olarak tasarlanmıştır. Oyun platform oyunu olduğu için platform öğelerini yansıtması adına *"Tile"* sisteminin ızgara şeklinde kullanılması uygun görülmüştür. Izgara sistemi ile harita tasarımları geliştirilmesi hızlı olması adına Unity 'e bir eklenti kullanılmıştır. Bu eklenti detaylı bir şekilde 5.2 Unity 2D Ekstraları başlığı altında sunulmaktadır. Geliştirilen grafikler aşağıdaki Şekil 6 'da sunulmaktadır. Ayrıca ilk iki bölümün arka planı hazır olarak alınmıştır [1].



Şekil 6. Zemin

4.3 Bölüm ve Düşman Tasarımları

Oyunda 2 farklı bölüm ve 1 tane güvenli köy olmak üzere 3 sahne vardır. İlk bölüm açık havada vadide geçer. Burada saldırgan yılanlar ve etkileşime geçilebilecek eşyalar bulunur. Tabana eklenen çiçek ve ağaçlar sayesinde orman havası verilmeye çalışılmıştır.

İlk bölümü başarıyla tamamlandıktan sonra güvenli köy bölümüne gelinir. Bu güvenli köy *"dungeon"* 'a girmeden önceki son duraktır. Yine açık havada olup ormanın kenarına kurulmuş bir köy görüntüsü oluşturur. İçerisinde evler, su kuyusu, ufak bir tezgâh ve çeşitli kutular vardır. Karakter bir sonraki bölüme geçmeden evvel burada canını doldurur ve son hazırlıklarını gerçekleştirir.

Bir sonraki bölüm *"dungeon"* kısmının ilk katıdır. Karanlık bir ambiyansa sahiptir ve etrafı çepeçevre beton duvar ile örülüdür. *"Mini golem"* ve *"dwarf"* gibi çeşitli düşmanlara ev sahipliği yapar. Ancak bu bölümde önceki bölümlerdeki mutlu ve aydınlık havaya nazaran daha ürkütücü ve korkutucu bir hava vardır. İçerisinde daha önceden canlıların üzerine düşüp can verdiği kazıklar, insanların içerisine hapsedildiği kafesler, bu kafeslerin sallandırıldığı ve

hatta bazen insanların bağlandığı zincirler ve fokurdayan lavlar vardır. Kazık ve lavların içerisine düşüldüğü takdirde karakter ölür ve bölüme yeniden başlar.

Düşmanlar Unity 'nin sağladığı ve işi kolaylaştırdığı *"Prefab"* olarak adlandırılan tür ile oluşturulmuştur. Bu sayede tek bir *"Prefab"* oluşturup birçok düşmanı baştan oluşturma gereksinimi duyulmamıştır. Düşmanlar temel olarak şu özelliklere sahiptirler:

1. Etrafta saldıracakları kimse yokken bulundukları zemin üzerinde hareketlerine devan etme.
2. Karakter görüş mesafelerindeyse karaktere doğru yürüme
3. Karakter düşmanın arkasındaysa ancak yine de belirli bir mesafenin içerisindeyse karaktere doğru dönme
4. Karaktere saldırabilecek mesafedeyse durup saldırı eylemini gerçekleştirme.

Düşmanların sahip oldukları animasyonlar ise saldırı, yaralanma, yürüme, sabit durma ve ölme animasyonlarıdır.

Oyunda *"yılan"*, *"mini golem"* ve *"dwarf"* olmak üzere 3 çeşit düşman vardır.

- Yılan; ormanda bulunur.
    - Canları azdır.
    - Az hasar verir.
    - İki saldırı arası bekleme süresi azdır. Bu sayede sık sık saldırır.
- "*Dwarf"* yalnızca *"dungeon"* 'da bulunur.
    - Canları yılanlara oranla daha fazladır.
    - Saldırıları orta güçlüdür.
    - İki saldırı arası bekleme süreleri yılanlara oranla daha fazladır.
- *"Mini golem"* hem ormanda hem de *"dungeon"* 'da bulunur.
    - Canları çok fazladır.
    - Saldırıları çok güçlüdür.
    - İki saldırı arası bekleme süresi çok uzundur.

5. KODLAMA

5.1 Oyunun Kodlanması

Öncelikle Github içinde gizli bir depo olan *"Kingdom of Walantis"* deposu açılarak takım üyelerinin bu depoya katılması sağlanmıştır. Geliştiriciler Git kullanarak bu depoyu edindikten sonra depoda yeni bir Unity 2D projesi kurulumu yapılarak projeye başlanmıştır. Yüklemelerde gerekli olmayan ve Unity tarafından oluşturulabilecek dosyalar için *".gitignore"* dosyası oluşturulmuştur [2]. Önemli her değişiklik Github üzerinde yeni bir *"branch"* açılarak yüklenmiş olup hatalardan kaçınılmıştır.

Yazılan Kodlar:

PhysicsObject, LevelController, BackGround, PlayerController, PlayerAttackController, HealthController, Interactable, InteractionController, InventoryControlller, EnemyController.

- PhysicsObject:

Bu sınıf, oyun içerisindeki nesnelerin fiziksel mekaniklerinin daha akıcı ve güzel hissiyatlı olmasını sağlar. Unity programının içerisinde hazır olarak tanımlı olan RigidBody2D kütüphanesi ile nesnelerin mekanikleri hissiyat olarak gerçekçi gelmediğinden geliştirici ekip mekanikleri kendisi oluşturmuştur. Sınıfın içerisinde RigidBody2D kütüphanesi sadece yer çekimi özelliğini almak için kullanılmıştır. İçerisindeki update() fonksiyonu sayesinde nesnelerin fizik kuralları çerçevesinde hareket edebilmesi için her saniye üzerlerine etki eden kuvvet vektörleri yeniden hesaplanır. Özellikle yer çekimi her saniye ekip üyeleri tarafından ortak bir kararla seçilmiş bir ivme ile artar.

Karakter hareket halindeyken gerçekçiliği korumak için karakterin anlık olarak çarptığı (değdiği) her bir yüzey liste ile tutulmuştur. Aynı zamanda her saniye yataydaki hız vektörü ile o anki hız çarpılarak yatay düzlemdeki hız saklanır. Böylelikle karakter tavana çarptığı anda yataydaki hızını kaybetmez ve yataydaki hızına bir ölçüde devam eder [3].

- LevelController:

Bu sınıf sahneler arası geçişlerden sorumludur. İçerisinde bulunan metodlar sayesinde aynı bölümü (sahneyi) yeniden başlatabilir, bölüm tamamlandıktan sonra karakteri yeni bölüme geçirebilir. Yeni bölüme geçmek için karakterin her levelin sonunda bulunan görünmez bir nesne ile çarpışması (collision) sonucu bu sınıf içerisindeki ilgili metod harekete geçer ve yeni sahne yüklenir. Düşmanlar, lav veya kazıklar yüzünden karakter ölürse, bölümün yeniden yüklenmesinden bu sınıf sorumludur.

- PlayerController:

Bu sınıf, karakterin temel hareketlerinin tamamının tanımlandığı sınıftır. Karakterin zıplama ile ulaşabileceği maksimum yükseklik, havadayken gerçekleştirebileceği ikinci bir zıplama hakkının olup olmadığı, ileri doğru atılma sağlayan dash hareketinin tekrar kullanılabilme süresi, karakterin temel yön hareketleri için gerekli fiziksel kuvvetler ve karakterin temel animasyonları burada kontrol edilir. Aynı zamanda karakterin incelenebilir nesneler ile etkileşime gireceği zaman veya envanteri açıldığı zaman hareketini durdurulması, lav veya kazık gibi öldürücü nesnelere değdiği zaman ölmesi, yaralandığı zaman gerekli animasyonla beraber o anki gerçekleştirdiği saldırının sonlandırılması ve canının kendisine gelen saldırı kadar azaltılması burada sağlanır. PhysicsObject sınıfının özellikeri bu sınıfa override edilir.

- PlayerAttackController:

Bu sınıf, karakterin tüm saldırı mekaniklerinden sorumlu sınıftır. Karakterin saldırısının ne kadar uzağa erişebileceği, hangi sıklıkla saldırı yapılabileceği, saldırı sonucu karakterin silahının düşmana değip değmediği, zıplama esnasında karakterin saldırı gerçekleştirememesi gibi kontroller sağlanır. Saldırı için gerekli klavye girdileri de burada kontrol edilir. Karakterin 2 adet saldırı türü mevcuttur. Bunlardan ilki hızlı ama az etkili bir hafif saldırıdır. Diğeri ise yavaş ama çok etkili bir ağır saldırıdır. Ağır saldırı fazla vurmasına rağmen saldırı süresi uzun olduğundan bu esnada düşmandan bir hasar alınması fazla olasıdır.

- HealthController:

Bu sınıf, karakterin düşman tarafından aldığı saldırılar sonucu PlayerController sınıfından çağrılır. İçerisinde karakterin canının; alınan saldırının kuvveti kadar azaltılmasını sağlayan metodla birlikte güncel can durumunu arayüzde gösteren can barını güncelleyen bir metoda sahiptir.

- Interactable:

Bu sınıf, oyundaki oyuncu ile etkileşime girecek tüm nesneler için oluşturulmuştur. Sınıf, nesnelerin Etkileşim Türü, Eşya Türü ve Olaylar gibi özelliklerini barındırmaktadır. Etkileşim Türü ve Eşya Türü özellikleri ayrıntılı olarak 4.1 Envanter ve Etkileşecek Nesneler başlığı altında anlatılmıştır. Olaylar ise Unity 'nin içinde olan *UnityEngine.Events* modülü ile gerçekleştirilmiştir. Olaylar, oyunda olacak olayların gerçekleşmesini ve tüketilecek olan nesnelerin kontrol edilmesini sağlar.

- InteractionController:

Bu sınıf, oyuncu etkileşebilecek nesnelerine üzerine geldiğinde onlarla etkileşimin kurulmasını sağlayan ana sınıftır. Eğer nesne incelenebilir bir nesne türüyse incelemesini sağlar.

- InventoryControlller:

Bu sınıf, envanterin açılmasını, nesnelerin envantere alınmasını, eşyaların tüketilmesini ve güçlendirmeleri gerçekleştirmeyi sağlayan sınıftır. Liste veri yapısı kullanılarak envantere alınan eşyalar nesne olarak tutulmaktadır.

- EnemyController:

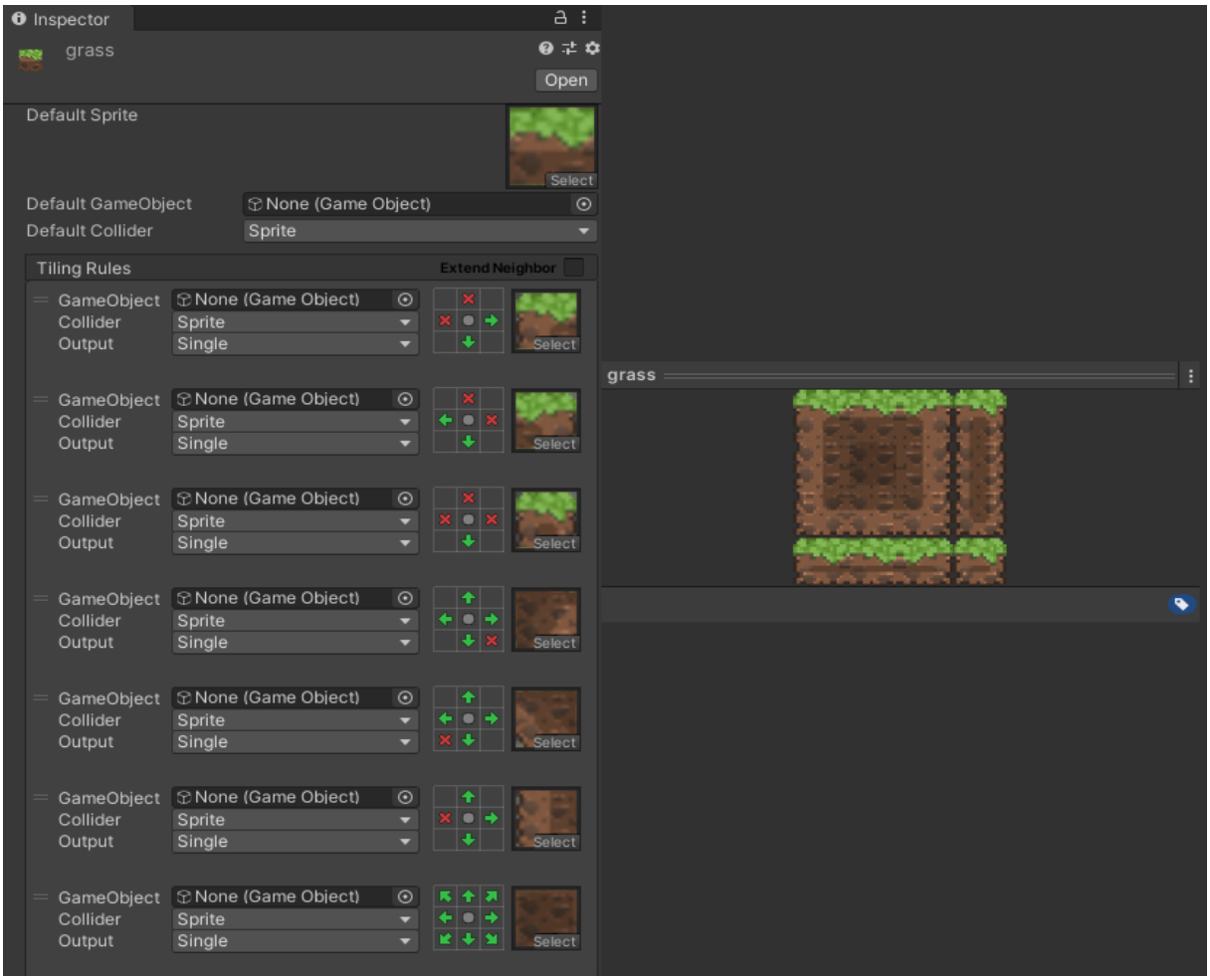
Bu sınıf, düşmanların kontrollerinin sağlandığı sınıftır. İçerisinde düşmanların yürüme hızı, saldırılarının ne kadar uzağı etkileyeceği, bir sonraki saldırı için gerekli bekleme süreleri,

verdikleri hasarın boyutu ve canları tutulur. Bu sınıf da PhysicsObject sınıfından gerekli mekanikleri override eder. Buna ek olarak düşmanların karakter belirli bir sınırın dışında ise kendilerine has bir kalıp ile sola ve sağa hareket etmeleri, karaktere doğru yüzleri dönükse ve görme mesafesinin içerisindeyse karaktere doğru yürümeleri, karaktere arkaları dönükse ancak karakter bu düşmanlara belirli bir mesafeden daha yakınsa düşmanların karaktere doğru dönüp saldırmak üzere karaktere doğru yaklaşması, saldırı mesafesi sağlandığı takdirde hızlarını sıfırlayıp saldırı eylemine geçmeleri sağlanır.

Bunlara ek olarak düşmanların karakter tarafından bir saldırı alması sonucu canını azaltmak ve yaralanma animasyonları çalıştırmak ve eğer canı tamamen bitmişse düşmanın ölmesi için gerekli fonksiyonlar yine bu sınıf içerisinde tanımlanmıştır. Bu sınıf bütün farklı düşman tiplerine uygulanabilir. İstenildiği takdirde farklı düşmanlara farklı can, saldırı gücü veya saldırı sıklığı değerleri verilebilir.

5.2 Unity 2D Ekstraları

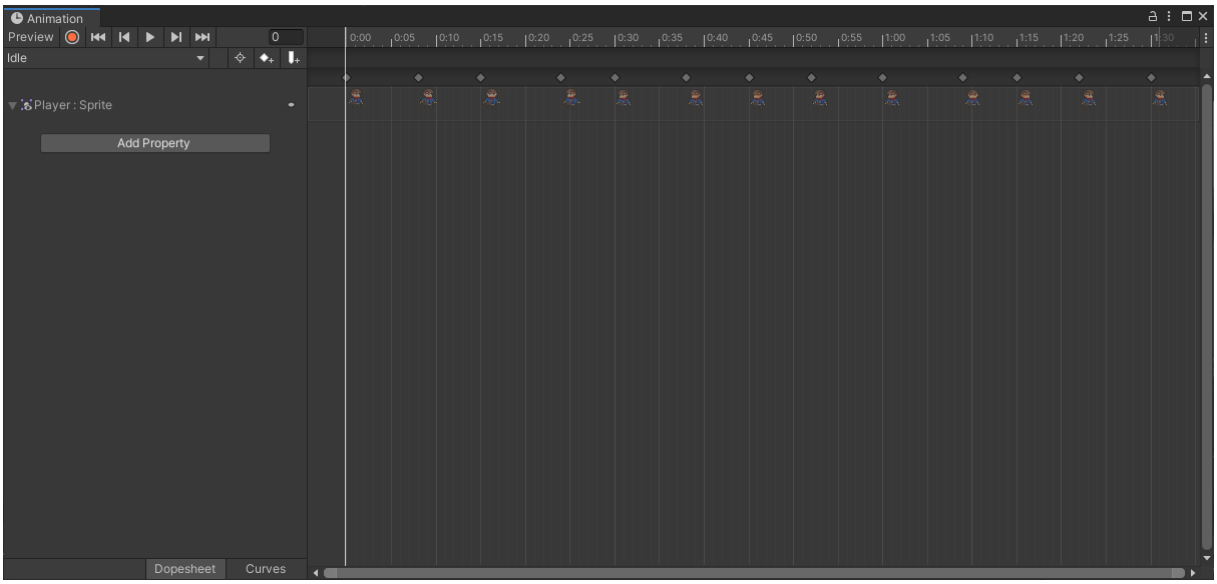
Unity 2D ekstraları, Unity içinde kullanılabilen bir eklenti olarak Unity geliştiricileri tarafından sunulmaktadır [4]. Unity 2D ekstraları *"Rule Tile"* kullanılması için projeye eklenmiştir. *"Rule Tile"* özelliği *"Tilemap"* ızgara yapısı kullanan oyunlarda geliştirilen grafiklerin belirli bir kurala göre çizilmesine olanak tanır. 4.2 Zemin Tasarımı başlığı altında sunulan grafikler kullanılarak kurallar oluşturulmuştur. Kurallar, kırpılan bir *"Tile"* 'ın sağına, soluna, altına veya üstüne bir grafik gelip gelmeyeceğini ayarlamayla oluşturulmaktadır. Oluşturulan "*Rule Tile"*, kaydedildikten sonra *"Tile Palette"* olarak kullanılacak olan palet oluşturulur ve içerisine bu *"Rule Tile"* eklenir. Böylece hızlı bir şekilde çizilen veya tasarlanan bölümler oyun projesine her bir grafiği teker teker seçerek değil de tek bir kurallı grafik kullanılarak hızlı bir şekilde eklenebilir. Paletten alınan fırça ile ızgaraya bölüm çizilirken hangi grafiklerin nereye gelmesi gerektiği Unity tarafından anlaşılmış olur [5].



Şekil 7. Rule Tile

5.3 Animasyonlar ve Animatörler

Animasyon ve Animatörler, oyuna eklenecek karakterlerin animasyonlarını ve animasyon ağaçlarını barındırmaktadır. 2 boyutlu bir oyun projesi karakter animasyonları yeni çerçeveler (*"frame"*) üzerine çizilerek yapılmaktadır. Her bir çerçevede karakterin yeni pozisyonu çizilir. Tüm çerçeveyi içeren bir dosya Unity içine aktarıldıktan sonra Unity 'nin içinde olan *"Sprite Editor"* ile bu dosyadaki çerçeveler kırpılır. Kırpılma işlemi sonucunda ortaya üst üste konularak animasyon elde edilebilecek görüntüler açığa çıkar. Unity içinde olan Şekil 8 'de da arayüzü gösterilen *"Animation"* özelliği ile animasyon oluşturulur [6].



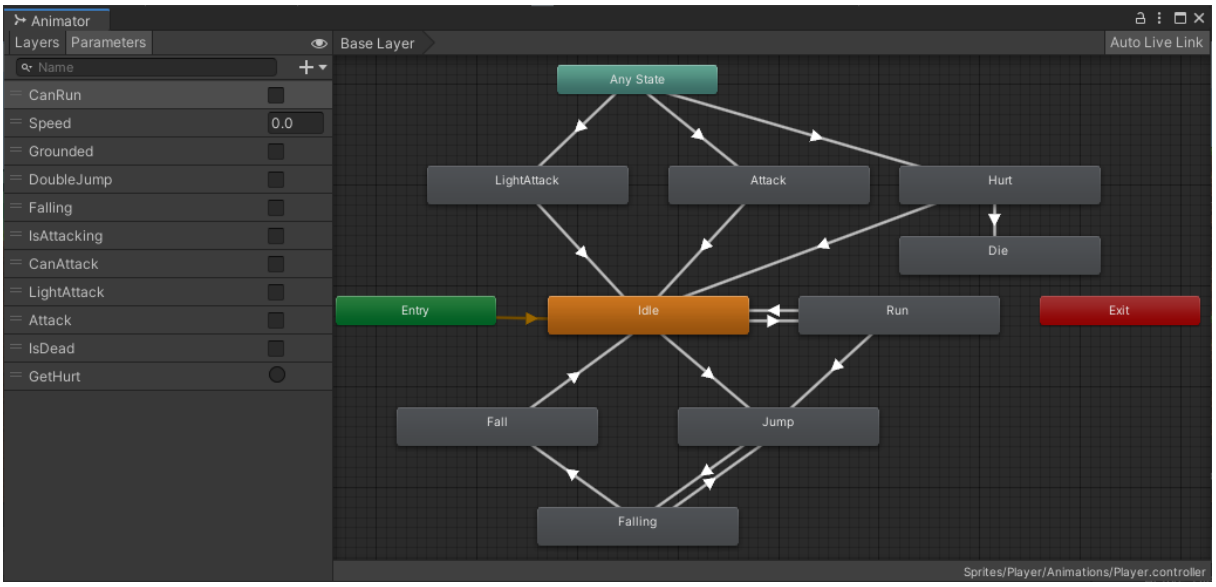
Şekil 8. Animasyon penceresi

Animasyonlar oluşturulduktan sonra bir animasyon kontrolcüsü olan animatör gereklidir. Oyunda saldırı, yürüme, sabit durma gibi animasyonları olan karakterler için kesinlikle animatör gerekmektedir. Animatörler, animasyonların geçiş durumlarını ve bu durumların hangi zamanlarda gerçekleşeceğini belirtmekte olup animasyon ağacı şeklinde var olmaktadır. Öncelikle animatörlerde bir tane başlangıç durumu bulunmaktadır. Bu durum hiçbir girdi gelmediğinde oynatılacak olan animasyondur. Bu projede başlangıç durumu sabit durma animasyonu olarak tasarlanmıştır. Oyuncu veya düşmanlarda herhangi bir girdi yok ise sabit durma animasyonu çalışmaktadır. Saldırı gibi girdiler *"Any State"* adlı durumdan alınmaktadır. Bu durumdan çalışacak olan animasyonlar herhangi bir durumdan tetiklenerek çalışmaktadır. Başlangıç durumundan bu durumlara geçişlere gerek yoktur fakat bu durumlardan başlangıç durumuna geçişe gerek vardır. Ve geçiş yapılırken çıkış süresi özelliği verilmesi gerekir. Çıkış süresi özelliği animasyonun duracağını belirtir.

Şekil 9. Ana karakter sabit duruş çizimleri

5.3.1 Ana Karakter Animasyonu ve Animatörü

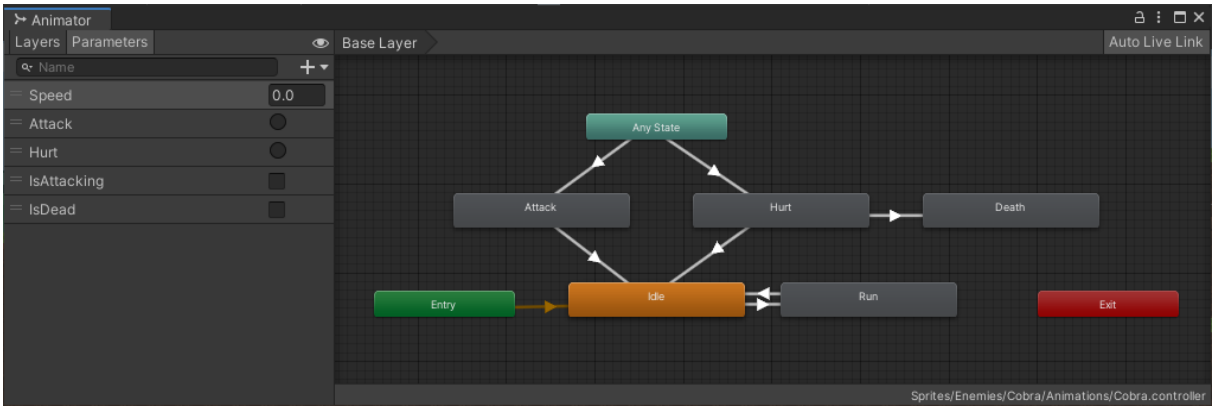
Ana karakter animasyonu ve animatörü oyun projesinin en önemli kısımlarından birisidir. Ana karakterin sabit durma, koşma, zıplama, hafif saldırı, saldırı, yara alma ve ölme animasyonları mevcuttur. Bu animasyonlar arası geçişler animatörde parametreler tutularak sağlanmaktadır. Ana karakterin çizimleri ve animasyonları hazır olarak alınmıştır [7].



Şekil 10. Ana karakterin animatörü

5.3.1 Düşmanların Animasyonu ve Animatörü

Düşmanların animasyonları ve animatörü ana karakterin animasyonları ve animatörüne daha temel düzeyde olmaktadır. Düşmanlar sabit durma, koşma, saldırı, yaralanma ve ölme animasyonlarını içermektedir. Düşmanların animatörü, ana karakterin animatörüne benzer fakat zıplama ve ekstra saldırı animasyonu olmadan tasarlanmıştır. Düşman karakterlerin çizimleri ve animasyonları hazır olarak alınmıştır [7].



Şekil 11. Düşmanların animatörü

6. UNITY OYUN MOTORU İLE WİNDOWS TABANLI 2D PLATFORM OYUNU

6.1 Kullanıcılar

6.1.1 Admin

Oyun içerisinde yönetici yetkisine sahip herhangi bir kullanıcı bulunmamaktadır.

6.1.2 Danışman Hoca

Dr. Öğr. Üyesi Öner BARUT

6.1.3 Öğrenciler

Mehmet Oğuz TOZKOPARAN, Ramazan Emre ERKAN

6.1.4 Kullanıcılar

Kullanıcılar, geliştirilen Unity Oyun Motoru ile Windows Tabanlı 2D Platform Oyunu 'nu oynayabilecekler.

7. TEST AŞAMASI

7.1 Mekaniklerin Testi

Oyunun geliştirilmeye başlamasından sonra eklenen her yeni özellikle birlikte gerekli testler sağlanmıştır. Karakterin duvarlarla olan etkileşimi, koşarken veya zıplarken saldırı yapabilmesi, karakterin sola veya sağa doğru hareketi sırasında animasyonların doğru tarafa doğru bakıp bakmadığı kontrol edilmiş ve fark edilen hatalar giderilmiştir.

Rule tile özelliği eklenirken oluşturulan haritalardaki yanlış yerleştirmeler, düşmanların öldükten sonra deaktive edilmesi, düşmanların olması gerektiği gibi karakteri takip edebilmesi, etkileşime geçilecek nesnelerin yerden alınabilmesi ve alındıktan sonra tüketilebilmesi ile birlikte bu kazanılan özelliklerin yeni bölüme taşınabilmesi, eklenen kazık ve lavların karakteri direkt olarak öldürebilmesi, bölüme yeniden başlanılması veya bir sonraki bölüme geçilmesi gereken durumlarda karakterin ve envanterinin doğru bir şekilde taşınabilmesi gibi özellikler test edilmiştir.

Karşılaşılan tüm bu hatalar Unity programının debug sisteminden de faydalanılarak giderilmeye çalışılmıştır. Oyunun çalışmasına engel olmayan ancak oyunun gerçekçi ve akıcı bir şekilde oynanmasına engel olan mantıksal hatalar; kod içerisinden eklenen ve değişkenlerin Unity programının konsol ekranından yazdırılmasını sağlayan Unity yardımcı

kütüphaneleriyle takip edilmiş olup bu hatalar giderilmiştir. Hataların çözümünde internetten yardım alınmıştır [8].

7.2 Grafiklerin Testi

Oyun yapısı itibariyle 16 pixel bir oyun olduğundan hem kendi oluşturduğumuz hem de internetten hazır olarak aldığımız çizimleri eklerken Unity programının kullandığı filtreleme ve sıkıştırma fonksiyonlarını deneyerek projemize uygun olan kombinasyonu seçtik. Böylelikle eklemek istediğimiz nesnelerin oyun içindeki boyutlarını ve netliklerini tam olarak ayarlayabildik.

Animasyonlar devamlı gelen frame ‘lerin (çerçeve) kısa aralıklarla sırayla ekrana verilmesi sonucu oluşur. Bu bağlamda eklenen her bir frame ‘in ekranda gösterilme sırası ve zamanını düzenlendikten sonra animasyonların doğru çalışıp istenilen görüntüyü sağlayıp sağlayamadıkları kontrol edildi.

Bunlara ek olarak animasyonların çalıştığı anlarda eklenen etkinlikler (event) ile animasyonun belirli saniyelerinde istenilen fonksiyonların çalıştırılması sağlanmıştır. Örneğin karakterin saldırıya başladığı animasyonda silah tam olarak en uç noktadayken saldırı kodunun metodu çağrılır böylelikle animasyonlar ile kodların senkronizasyonları tam olarak eşleştirilir. Bu bağlamda eklediğimiz animasyonların doğru zamanda doğru metotları çağırıp çağırmadığını da kontrol ettik.

Oyun içi envanter arayüzünde envanterdeki eşyaların ekranda gösterilirken tam olarak envanter kutucuklarının içerilerinde olması gerektiği gibi durmaları, seçilen envanter öğesinin açıklamasının yine aynı envanter arayüzünde doğru bir şekilde yazdırılabilmesi kontrol edildi ve bu sebepten ortaya çıkan hatalar giderildi.

8. SONUÇ

Oyunlar ölçülü ve seçici oynandığında, stres atma kaynağı olmanın yanında kişinin zihinsel sağlığının iyileşmesi ve sosyal becerilerinin gelişmesi için bir katalizördür. Geleneksel olan masa oyunlarından ve diğer eğlence türlerinden farklı bir seviyede olan video oyunları ilgi çekici ve sürükleyicidirler. Oyunculara sunulan hikaye ile onların duygularına dokunabilirler. Görselleri ile oyuncuların yaratıcılıklarını geliştirebilirler. Bu çıkarımların yanı sıra sadece oyuncuların değil oyun geliştiricilerin de çok fazla deneyim kazanmasını sağlar. Ayrıca oyun geliştirirken öğrenilen ve geliştirilen birçok yetenek vardır. Bunlar geliştirici takım üyeleri için oldukça iyi deneyimlerdir. Kingdom of Walantis oyun projesini geliştirmek biz ekip üyelerine oyun geliştirirken planlama ve analiz, tasarım, oyunlarda bölüm tasarımları, karakter tasarımları, envanter ve eşya tasarımları, Unity Oyun Motoru, çeşitli algoritmalar, animasyonlar, risk değerlendirmesi, bağımsız oyun geliştiriciliği, geliştirme aşamasında olan oyunun testi ve oyun geliştirme sektörü hakkında bilgi sağlamıştır.

9. KAYNAKLAR

[1] İnternet: Free Pixelart Tileset - Cute Forest by aamatniekss. https://aamatniekss.itch.io/free-pixelart-tileset-cute-forest (27.01.2021).

[2] İnternet: Unity Gitignore. https://github.com/github/gitignore/blob/master/Unity.gitignore (27.01.2021).

[3] İnternet: 2D Physics - Unity Learn. https://learn.unity.com/tutorial/2d-physics (27.01.2021).

[4] İnternet: Unity-Technologies/2d-extras: Fun 2D Stuff that we'd like to share. https://github.com/Unity-Technologies/2d-extras (27.01.2021).

[5] İnternet: Introduction to Tilemaps – Unity. https://learn.unity.com/tutorial/introduction-to-tilemaps (27.01.2021).

[6] İnternet: Working with Animations and Animation Curves - Unity Learn. https://learn.unity.com/tutorial/working-with-animations-and-animation-curves (27.01.2021).

[7] İnternet: Elthen's Pixel Art Shop - itch.io. https://elthen.itch.io/ (27.01.2021).

[8] İnternet: The best place for answers about Unity - Unity Answers. https://answers.unity.com/ (28.01.2021).