ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM 4062-R PROJE RAPORU

Unity ile İzometrik ARPG Oyun geliştirme

Şuayip Deniz Talha TOPAY(18290059)

Alcad ALİ(18290992)

Doç.Dr. Gazi Erkan BOSTANCI

20 Mayıs 2022

ÖZET

Bu proje kapsamında Unity üzerinde 2 dönem boyunca Isometric ARPG (Isometric Action Role Playing Game) türünde bir oyun geliştirmekteyiz. Bu raporda güz döneminde anlattığımız, üstünde çalışmalarımızı sonuçlandırdığımız ARPG oyunumuzun son geldiği noktadan ve bu noktaya geliş aşamalarından bahsedeceğiz.

İÇİNDEKİLER

1. GİRİŞ

Geçen dönem raporumuz sonunda 4 tane hedef listelemiştik. Bu hedefler şu şekilde idi:

1)Karakter haraketinin ve animasyonunun oturtulması.

2)Düşman karakterlerin oyuna eklenmesi ve dövüş sistemi.

3)Harita tasarımının çeşitlendirilerek 3-4 haritaya çıkılması.

4)Karakter ve kılıcı için gelişme aşamaları oyuna entegre edilmesi.

Bunlardan 3 tanesini tamamen, 4. Hedefimizi ise kısmi olarak elde etmeyi başardık.

Karakter animasyonlarımız tamamen düzeltildi ve karaktere oturtuldu. Oyun içersinde 3 farklı düşman ve 3 farklı harita bulunmakta. 4. Hedef için ise karakterimiz kısmi olarak gelişebilmekte ancak kılıcı gelişmemektedir.

2. DÖNEM İÇİ İLERLEMELER

2.1 Plastic SCM'e Geçiş

Geçen dönem versiyon kontrolü yazılımı olarak Unity'nin kendi içinde bulunan Unity Collaborate kullandığımızdan bahsetmiştik. Unity çalıştığımız dönem içerisinde versiyon kontrolu yazılımını değiştirme kararı aldı ve Unity Collaborate için destek vermeyi bıraktı. Bunun yerine artık tamamen Unity içerisinde bir arayüz olarak çalışabilen ve gerektiğinde detaylı işlemler için ayrı bir masaüstü yazılımı bulunan Plastic SCM'e geçiş yaptı. Biz de Unity Collaborate'i daha fazla kullanamadığımız ve daha önceki versiyon kontrolü kurulumumuz artık çalışmadığı için projemizi Plastic SCM'e taşıma kararı aldık.

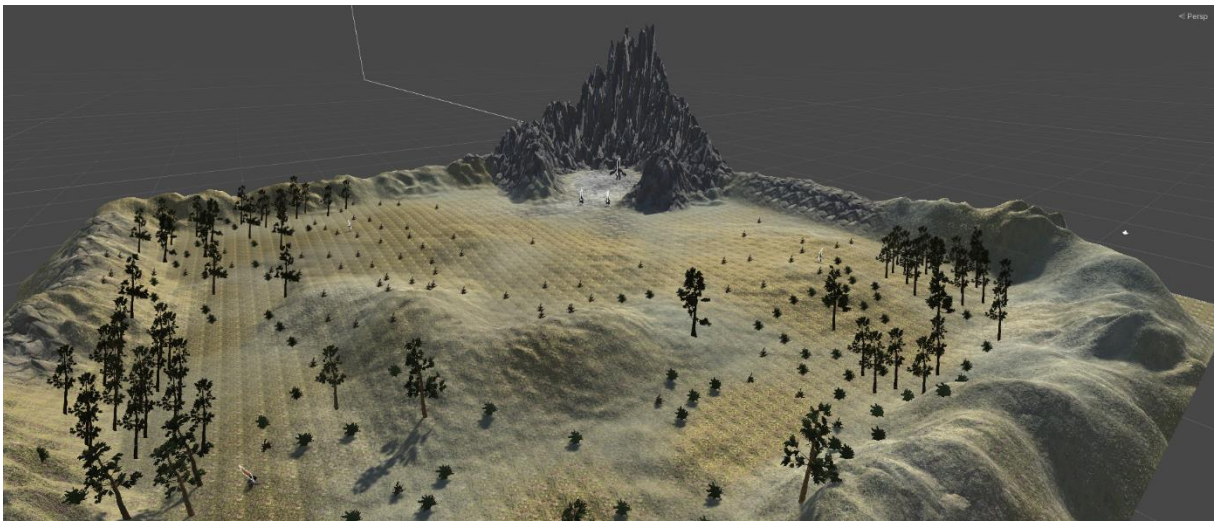
Plastic SCM'in bize sağladığı bağzı kolaylıklar ve Unity Collaborate içerisinde bulunmayan bazı özellikleri var. Öncelikle Unity içerisindeki arayüz detaylandırılmış, eskisi gibi sadece push ve pull seçeneklerinden oluşmamakta. Yapılan commitlere yorum yazılabilmekte, istenildiği takdirde ise projenin daha eski bir haline dönülebilmekte. Bundan daha detaylı çalışılmak istenirse ise Unity'den tamamen ayrı bir desktop uygulaması geliştirilmiş ve bu uygulama Unity içerisindeki değişiklikleri gerçek zamanlı olarak görüntüleyebilmekte. Projemizi Plastic SCM'e geçirmek için ise Unity sitesindeki adımları uyguladık. Plastic SCM uygulamasını indirip bunu Unity hesabımıza bağladık, projemizi ise Unity'i güncelledikten sonra yeni versiyonu ile tekrar açtık. Dipnot olarak bu geçişin tek yönlü olduğu da belirtimektedir. Bir proje Plastic SCM'e geçtikten sonra Unity Collaborate'e geri dönememekte.

2.2 Yeni Haritalar ve İyileştirmeler

Oyunumuzun son halinde 3 harita bulunmakta. Bunlardan ilki geçen dönem üzerinde durduğumuz dağ haritası. Bu haritanın üzerinde yer şekli ve doku değişikliği yapmadık. İkincisi ise yeni eklediğimiz haritalardan olan karlı bir mekan. Bu haritada eski haritaya benzer bir yer yüzü yapısı kullanıldı, doku olarak ise normal toprak ya da çim dokusu üzerine kar dokusu kaplandı. Kar dokusu olabildiğince gerçeğe yakın olması için yüksek yerlerde daha yoğun, alçak yerler ve patika gibi alanlarda ise daha

seyrek olacak şekilde yerleştirildi. Ayrıca default şekilde gelen kar dokusu istediğimiz görüntüyü vermediği için bu dokunun ayarlarından smootness değerini arttırarak daha parlak bir görüntü elde ettik.

Üçüncü ve son haritamız ise diğer haritalarımıza göre daha az lineer, düz ve açık bir alan. Orta kısmında hafif bir tepe ve oyunun son boss odası bulunmakta. Boss odası tasarımı olarak etrafında keskin kayalıklar olan bir alan tasarladık. Orman haritası içerinde ise bu alana yaklaşıldıkça yavaş yavaş bir geçiş olacak şekilde bitki örtüsünü azattık ve yer kaplamasını ise topraktan kayaya doğru geçirdik. Bu son odaya yaklaşırken oyuncunun bunu önceden fark etmesini sağladı ve tamamen etrafı kapalı keskin bir kayalık çukur olduğu için son boss dövüşü yapılırken oyuncu üzerinde daha heyecan uyandırıcı bir tecrübe haline geldi.



Şekil 2.2.1 3.Harita ve Boss Odası

3 haritada da ortak yaptığımız bir değişiklik ise üç boyutlu bir bitki örtüsü eklemek oldu. Bunun için ücretsiz bir orman asseti kullandık. Assetin içinde 5 ayrı tipte ağaç ve 2 ayrı tipte küçük çalı bulunmakta. Bu nesneleri Unity içine import ettikten sonra prefab dosyalarını terrain editor içerisine kaydediyoruz. Bu adımı her ağaç için tekrarladıktan sonra bir ağaç havuzumuz oluyor. Bundan sonra ise fırça kullanarak hangi ağaç türünü haritada nereye yerleştireceğimize karar veriyoruz. Aynı ağaç havunda çalıştığımız için yeni eklenecek ağaç önceki nesneleri de dikkate alarak ekleniyor ve çarpışmaların önüne geçmiş oluyor. Rastgele rotasyon ve boyut seçenekleri de aynı ağaçların tekrar tekrar kullanıldığını gizlemekte kullandığımız önemli bir araç. Bu aşamalar sonucunda zaten iyi bir noktaya gelmiş oluyoruz. Son

olarak elde edebileceğimiz en iyi görüntüyü elde etmek için bazı uygun yerde olmayan ağaçları ve çalıları kaldırdık, haritanın uc noktalarındaki bazı alanlarda ise sıklaştırarak daha canlı bir orman görünümü elde ettik.

2.3 Düşmanlar

Oyunda 3 türden farklı düşmanlar olacak; iskelet, titan ve wizard. Oyunun 1. seviyesinde sadece iskelet düşman ile başlancaktır ve bu düşman küçük boyutunda olacak, ancak seviyeler ileriye geçtikçe farklı türler ve boyutlardan bazı düşmanlar olacak. Bu düşmanları seçmek için Unity Asset Store sitesinden arama yaparak ARPG oyununa en uygun 3D düşmanları ve istediğimiz özellikler ile birlikte bulduk. Aşağıda resimi gösterilecektir.

Şekil 2.3.1. Oyun Düşmanları

Bu düşmanlar silahları ve animasyonları bulunur. Biz onları import ederek Unity üzerine indirebildik ve modelini sahneye koyarak ekleyebildik. Bu modeller Enemy objesini adlandırdık ve Inspector kısmındaki bazı componentleri ekledik, onlara bahsedecek olursak; NavMeshAgent component, Bu bileşen, kullanarak sahnede gezinmesine izin vermek için oyundaki objeye eklenir. Box Collider ve Rigidbody component'ler, Unity ortamında bir nesnenin başka bir nesneyle çarpıştığını algılamak için nesnelerin üzerine bu componentleri eklememiz lazımdır. Enemy adlı bileşeni, script olarak kodlama yapmak için ekledik. Bunları ekleyip gerekli adımlar

yaparak istediğimiz hareketleri oluşturabiliriz. Oyunun düşmanları seçtikten sonra onların hareketini nasıl ayarladığımızı bahsedeceğiz.

2.3.2 Düşman Hareketi:

Düşmanların hareketi player'e göre ayarladık, ve bunu yapmak için aralarındaki mesafe hesaplayarak gerçekleştirdik. Enemy script'in içinde Transform türden bir tane player varaible'ı tanımladık, çünkü unity sahnedeki player objesine ulaşmak isteyip arasındaki mesafa hesaplayacağız. Aralarındaki mesafa kaç olduğunu öğrenmek için Vector3.Distance fonkisyonu çağırarak içinde player'in pozisyonu ve trasform pozisyonu (enemy script'i kendi enemy objesine eklediğimiz için transform olacak) yaptık.

Elde ettiğimiz değeri haspladıktan sonra başka değerlerden karşılaştık; bu değerlere belli mesafeleri belirledik; 1. değer bize kovalama sınırı ve 2. değer ise saldırma sınırı, kovalamak için mesafeyi 8, eğer hesapladığımız değeri 8 den küçük yada eşit ise düşmanlar palyer'e doğru kovalamaya başlayacaklar, saldırmak için mesafeyi 3.5, o yüzden eğer düşman player'e çok yakın olursa saldıracak.

Kovalama ve saldırma için düşmanların animasyonları kullandık. Bunu yapmak için her bir düşman objesine Animator component'i atadık, bu Animator içinde kullanmak istediğimiz animasyonları ayarladık ve birbiriyle transfom bağlantıları var, yani Idle durumundayken kovalama durumuna geçecek, kovalama durumundayken saldırma durumuna geçecek. Bu şekilde hareketi ve animasyonları beraber çalıştırdık. Düşmanlar kovalarken sadece palyer'e takip edecekler ama saldırma yaparken kendi pozisyonlarında durup palyer'in tam önünde olup saldıracaklar.

Tabiki düşmanlar kovalama yada saldırma yaparken player de kaçmaya veya saldırmaya başlayacak. O nedenle palyer'in animasyonları burada çalışacak ve aynı şekilde çok yakın bir mesafedeyken düşmanın önüne bakıp saldıracak fakat burada saldırma yapmak için mouse'un sağ butona basıp gerçekleştirilecek, düşmanların animasyonları ise otomatik olarak mesafeye göre çalışacak. Eğer player'in ve düşmanların aralarındaki mesafe 8 den büyük olursa düşmanlar kendi pozisyonlarında duracaklar.

Bu şekilde hareket, kovalama ve saldırma animasyonları çalışmaya karar verdik.

2.4 Animasyonlar ve Can Barları:

Can barları yapmak için palyer ve bütün düşmanlara Unity üzerinden bir tane Canvas ekledik, bu Canvas'ın içinde Slider türden bir obje oluşturduk ve bu slider'ı bizim karakterlerimizin can barları olacaktır. Tüm can barlarının yönü kamera yönüne ait, bu şekilde Canvas içinde bir tane Bilboard script'i ekledik, bu script'te sadece kamera pozisyonu alıp LookAt fonkisyonun içine attık ve böylece can barları kameraya bakıp onun dönmesi ile hareket etmektedir.

Can barlarının değerleri 0-1 arasındaki tüm sayılara denk gelecektir. Oyun başladığında can barları 1'den başlayıp ve her karakter hasar aldığı zaman bu değeri azalacak ve böylece can barları hareket edecetir.

Ayrıca 2 tane daha düşmanların animasyonları var ve onları hasar alma ve ölme animasyonlarıdır. Aynı bir şekilde Animator kısmında ekleyip Enemy script'ten kontrol ettirdik. Can barları ve bu animasyonları aynı anda değiştirmek ve çalıştırmak için EnemyTakeDamage adlı bir tane fonksiyon kullandık. Bu fonksiyon bir tane integer tipinden bir değer alıyor, biz bu fonkiyonun AnimationStateController adlı bir tane scriptten çağırıp bu değer belirliyoruz çünkü palyer'in saldırma animasyonu çalışırken düşmanlar hasar alacaklar. Bu fonksiyon içinde bir tane değer tanımladık; bu değer oyun başladığında düşmanın maksimum canı temsil edecek, bu değeri ve fonksiyonu çağırıldığı yerden gönderilen değeri kullanarak can barının değeri azalacaktır.

Can barının değeri azalma olayı anlatacağımız şekilde yapılacak; player saldırma her seferde yaparken bir tane foreach loop çalışmaya başlayacak, bunu kullanma sebebi ise player'in etrafındaki düşman olup olmadığını kontrol etmektedir. Düşman olduğunu öğrenmek için Unity'de inspector kısmındaki Layer'e bir tane Enemy Layer ekleyip kontrol ederken kullandık.

O zaman player'in etrafındaki düşman varsa bahsettiğimiz fonksiyon çalışacak ve bir tane değer alacak, bu değeri düşmanın can değerinden azaltacağız. Eğer düşman canı 0' dan büyük ise sadece hasar animasyonu çalışacak, böylece devam ediyor ve canı 0' a eşit ise o zaman bu düşman ölme animasyonu çağırarak ölecektir.

Her bir düşman öldükten sonra düşmanların sayısı 1 azalacak ve düşman objesini yok edilecek. Yok etmek için Unity'de bir tane fonksiyon vardır. Bu fonksiyonun adı Destroy ve bir tane Rigidbody ve float tipinden 2 tane parametre alıyor, düşman objesi olduğu için bu fonksiyon içinde gameobject yazdık, ama float değeri 1 saniye olarak yaptık, bu float değeri kaç süreden sonra belirlediğimiz objeyi yok etme için kullanılıyor, böylece düşmanın canı 0 olunca ölme animasyonu çalışıp 1 saniye sonra yok edilecektir.

Player'e aynı şekilde can barı da ekledik ve onun içinde TakeDamage bir tane parametre alan bir fonksiyonu tanımladık. Bu fonksiyonun içinde aynı şekilde palyer'in canından gelen değeri azaltıp canı düşecek. Fakat bu fonksiyon düşmanlar saldırma yaparken çağrılmıyor, düşmanlar palyer' ile çarpışınca çağrılacaktır.

Unity'de üzerinden her bir düşmanın objesine bir tane Box Collider component'i ekldiğimizi daha önceki Düşmanlar Seçimi bölümünde bahsettik. Bu Box Collider'in içinde bir tane Is Trigger seçeniği var, bu seçeneği aktifleştirince OnTriggerEnter bir tane fonksiyon kullanabildik. Bu fonksiyon bir tane Collider tipinden bir nesne alıyor. Bu nesneyi bizim player'imizi temsil edecek, o yüzden palyer'e işaretlemek için tag kullandık çünkü Collider tipteki objelerin bir tag'ı oluyor, aynı zamanda Unity üzerinden inspector kısmındaki player objesine player tag'ı atadık.

Bahsedeceğimiz şekilde OnTriggerEnter foksiyonu çalışacak; düşmanlar player' ile çarpışınce otomatik olarak bu fonksiyon çağıracak ve eğer çarpıştıkları objenin tag'ı player ise TakeDamage fonksiyonu çalışıp player hasar alarak canı azalacak, canı 0' a denk gelirse o zaman ölecek yani oyun sonuna geldiğini anlamına demektir.

2.5 Oyun Arayüzü:

Oyun içinde 3 tane ekran ve menüleri olacak. Bu ekranlar görüntülemek için Unity'de bir tane Canvas oluşturduk, bu Canvasın içinde 3 tane panel atadık, her bir panel bir tane ekran gibi olacaktır.

Bu ekranları kontrol etmek için Canvas'ın içinde PlayerManager adlı bir tane script oluşturduk. PlayerManager script'i 3 tane Gameobject (yazdırmak istediğimiz ekranları) alıyor, 1.si başlangıç ekranı, 2.si sonraki aşama ekranı ve 3.sü ise oyun

bitti ekranıdır.

Şekil 2.5.1 Oyun arayüzü

Oyun başladığında oyun başlama(Start Game) ekranı gözükecek, bu ekranda 2 buton olacak, bir buton oyunu başlatmak için, diğer buton ise oyundan çıkış yapmak için ayarladık. PlayerManager script'in içinde bir tane StartGame fonksiyonu tanımladık, bu fonksiyon oyun başlatma ekranındaki başlat butonun içine ekledik, bu sayede butona basınca oyunu başlatacak, oyun başlama ekranı silinecek ve oyun aktif hale getirilecektir. Oyun başlamadan önce çıkış yapmak isternirse script'in içinde ExitGame fonksiyonu belirledik, bu fonksiyon oyun başlatma ekranındaki çıkış butonuna ekledik, butona basına çıkış yapılacak. Unity'de belirli bir fonksiyon var bu fonksiyon çıkış yapmak için kullanılıyor, Application.Quit() şeklindedir.

Her bir seviyeden sonraki seviyeye geçmek için sonraki seviye (Next Level) ekranı gözükecek, bu ekranda 2 tane buton bulunur. Bir buton sonraki seviyeye geçme butonu, diğeri ise oyun çıkış butonudur. Çıkış butonu bahsettiğimiz ExitGame fonksiyonu aynı şekilde çalışıyor. Sonraki seviyeye geçme butonu ise içinde bir tane NextLevel fonksiyonu tanımladık, bu fonksiyonda bir tane variable kullandık, bu variable bulunduğumuz seviyenin index'e temsil ediyor, onu arttırınca sonraki seviyenin index'e eşit olacaktır. 3 tane haritamız olduğu için 3 seviye anlamına demek, o yüzden bu index 3' e eşit olursa 0' a eşitledik. Bu nedenle 3. seviye bitince sonraki seviye butona basınca 1. seviyeden oyun yineden başlanacaktır.

Sonraki seviyeye geçmek için tüm düşmanların ölmesni bekliyoruz, o yüzden PlayerManager script'i her seferde düşmanların sayısını kontrol edecek, ve eğer bu sayı 0 olursa yani tüm düşmanlar öldü demek ve böylece sonraki seviye ekranı görünecektir.

Son oyun ekranımız oyun bitti (Game Over) ekranıdır. Bu ekranda aynı şekilde 2 tane buton olacak, oyun yineden başlatma ve oyun çıkış butonlarıdır. Oyun çıkış butonu aynı mekanizma ile çalışıyor, ama oyun yineden başlatma butonu ise oyunda kaybedince yineden oynamak istenirse ona basıp oyun başlatacaktır. Oyunu yineden oynamak için PlayerManager script'in içinde RestartGame fonksiyonu oluşturduk, bu fonksiyonda bulunduğumuz seviye tekrardan yüklenecek, bunu yapmak için Unity'deki SceneManager yardımı ile bulunduğumuz seviyenin index'e ulaşarak seviye yineden aktif hale getirilecektir.

Oyun bitmesi için bizim player'imizin canı 0' a eşit olunca olacaktır. Böylece player'in içindeki AnimationStateController script'inden palyer'in canını kontrol ettik. Bahsettiğimiz gibi player'in canı kontrol etmek ve azalmak için bir tane TakeDamage fonksiyonu kullandık, bu fonksiyonda player'in canı 0 düşünce PlayerManager script'in içindeki bir tane bool variable çağırılacak ve true değere eşitlenecek, bu variable true olursa player ölmüş anlamına gelecek, oyun bitti ekranı çalıştırılacak ve oyun aktif halini iptal edilecektir.

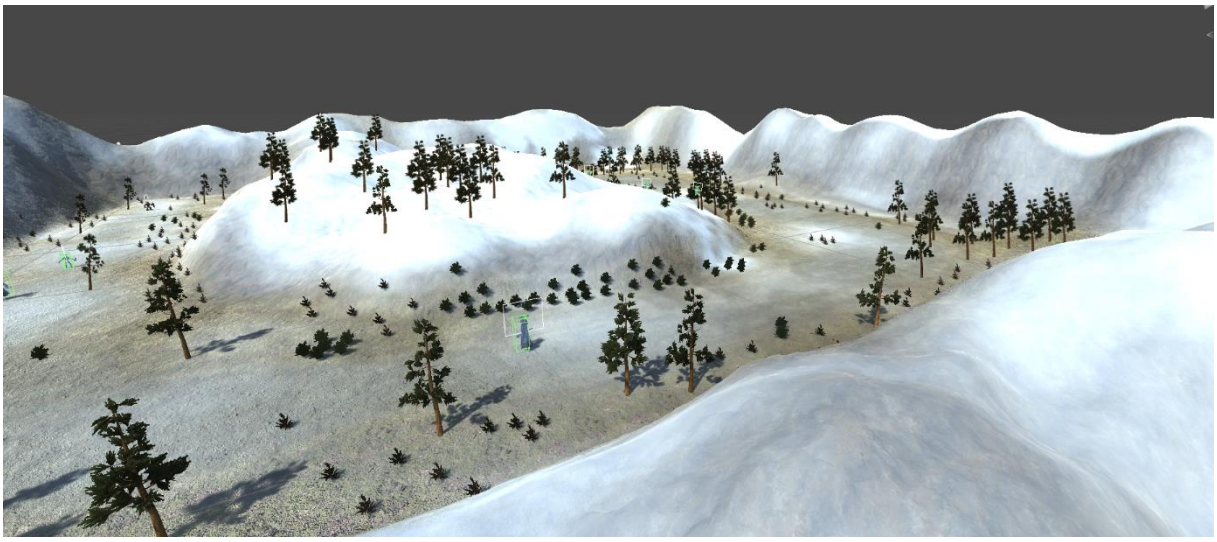
2.6 Can İksiri ve Zorluk

Dövüş sistemi bittikten sonra oyun zorluğunu denemek adına testler yaptık. Oyunumuzun bir bölüm başında tam canda doğup bölüm sonuna kadar herhangi bir iyileşme metodu ya da herhangi bir hata örtücü method olmadan bitirmek istemedik. Bunun için oyuncuya can iksiri seçeneğini vermeyi seçtik. Oldukça basit bir mekanik olacak şekilde oyuna uyumlu bir biçimde tasarladık. 'F' tuşuna basıldığında anında tam cana ulaşıyoruz. Dövüşürken anlık olarak kullanılabilmesi için herangi bir animasyon ya da bekleme süresi yok. Ancak oyundaki tüm zorluğu yok etmemek için bunu bölüm başına 5 defa kullanılabilecek şekilde sınırlandırdık.

Basitçe nasıl çalıştığından bahsedecek olursak ise; oyun durumlarına direkt erişimimiz bulunan AnimationStateController adlı script içine implemente ettik. Bu

script içinde zaten can barının grafik kısmını sergilemek için içerisinde tuttuğumuz bir Health değeri bulunmaktaydı. Basit bir kontrol mekanizması yazarak (5'ten fazla içilmemeli, can tam dolu ise içilmemeli vs.) tamamen çalışır duruma getirdik. Bunun sonucunda oyuncuya ekstra seçenek vererek yeni bir oynanış mekaniği eklemiş olduk. Artık ne kadar az canda kullanırsa o kadar çok can dolduracak bir mekanik olduğu için oyuncuyu risk almaya ve bunu ölmeye en yakın anda kullanmaya teşvik ediyoruz. Bu durumda oyuncu ölme riski ile sürekli karşılaşmakta ve oyun süresi boyunca ilgisini kaybetmemekte. Daha az risk almak isteyen biri ise bunu dövüşe girmeden hemen önce kullanmak isteyebilir ancak böylelikle bölümün son kısımlarında ise daha çok zorlanmayı göze alacaktır.

Oyun zorluğu ise genele hitap etmesi için ayarlandı. Hali hazırda oyun oynayan ve bu türde oyunlarda tecrübeli olan insanlara basit gelecektir. Çok oyun oynamamış ya da yeni başlayanlar insanlar için ortalama bir zorluk seviyesi olacaktır. Can barı dışında düşman sayıları ve zorluklarını da buna uyumlu olarak ayarladık. Oyun başlarında tek tük kolay düşmanlarla karşılaşırken oyun ilerledikçe sayı artmakta ve düşmanlar zorlaşmakta.



Şekil 2.6.1 Karlı harita üzerinde düşmanlar

Dövüş sistemi oturduktan sonra ise karakter hızı, dönüş hızı, saldırı hızı gibi çeşitli parametreleri de tekrar gözden geçirerek en uygun olan aralığı seçmeye çalıştık. Karakteri biraz daha hızlandırarak harita üzerinde yürümenin can sıkmasını engelleyerek saldırı hızını ise vuruş hissiyatını iyileştirmek için biraz yavaşlattık

3. SONUÇ

3.1 Sonuç

Geçtiğimiz yıl içerisinde baştan bir oyun yaptık ve süreç içerisinde bir sürü zorlukla karşılaştık. Takım çalışması, zaman planlaması, projeden beklentilerimiz gibi kavramlar zaman içerisinde daha da iyi oturdu ve gelişimimize büyük katkısı oldu. GBYF standına duyulan ilgi ise bizi çok gururlandırdı, hevesle gelen herkese elimizden geldiği kadar yaptıklarımızı açıkladık. Pozitif bir tecrübe elde etmiş olarak yolumuza devam edeceğiz.

4. KAYNAKÇA:

1. Unity Documentation : https://docs.unity3d.com/Manual/index.html
2. Unity Asset Store : https://api.unity.com/v1/oauth2/authorize?client_id=asset_store_v2&locale=en_US&redirect_uri=https%3A%2F%2Fassetstore.unity.com%2Fauth%2Fcallback%3Fredirect_to%3D%252F&response_type=code&state=53998af6-6a72-4fd2-afa4-2b96cb5544cf
3. Unity Collaboration : https://unity.com/unity/features/collaborate
4. Unity Dashboard : https://dashboard.unity3d.com/login
5. Introduction to Game Development , Terrain Generation and Navigation: https://books.google.com.tr/books?id=saq9AAAAQBAJ&printsec=frontcover&dq=Beginning+3D+Game+Development+with+Unity+4:+All-in-one,+multi-platform+game&hl=en&sa=X&redir_esc=y#v=onepage&q=Beginning%203D%20Game%20Development%20with%20Unity%204%3A%20All-in-one%2C%20multi-platform%20game&f=false
6. Unity Components, Camera and Player Characters : https://books.google.com.tr/books?hl=en&lr=&id=WfAWzVW9IK0C&oi=fnd&pg=PT16&dq=Unity+Game+Development+Essentials&ots=AV6NAyOLAm&sig=Gd5snzveC-i_RPTYz-_tTy_ca_4&redir_esc=y#v=onepage&q&f=false
7. Unity 3D Game Development and Animations : https://books.google.com.tr/books?hl=en&lr=&id=vFMggPqVnhcC&oi=fnd&pg=PT9&dq=how+to+animate+characters+%C4%B1n+unity&ots=zbyhY-aJPv&sig=7XWPaIpqJD1WAC5P9jqQ6i_k72A&redir_esc=y#v=onepage&q=how%20to%20animate%20characters%20%C4%B1n%20unity&f=false
8. Designing And Animating Characters: https://tigerprints.clemson.edu/cgi/viewcontent.cgi?referer=https://scholar.google.com/&httpsredir=1&article=3386&context=all_theses

9. Setting Up Animations On The Player Character And RPG Projects : https://books.google.com.tr/books?hl=en&lr=&id=LzgzEAAAQBAJ&oi=fnd&pg=PT11&dq=Unity+in+Action:+Multiplatform+game+development+in+C%23&ots=EUJR97P6Qx&sig=6Dr77018P_3CYhLtxG6BVvmuG20&redir_esc=y#v=onepage&q=Unity%20in%20Action%3A%20Multiplatform%20game%20development%20in%20C%23&f=false
10. Collaborative And Plastic SCM In Unity : https://ieeexplore.ieee.org/abstract/document/8402188
11. Creating Health Bars : https://books.google.com.tr/books?hl=en&lr=&id=1M8GBgAAQBAJ&oi=fnd&pg=PT5&dq=+health+bar+in+a+game+in+unity&ots=mii8Yve_mS&sig=Q8Oaljq_vhC8-xBmdiT9qXK8c9U&redir_esc=y#v=onepage&q=health%20bar%20in%20a%20game%20in%20unity&f=false
12. UI In Unity : https://books.google.com.tr/books?hl=en&lr=&id=oPBZDwAAQBAJ&oi=fnd&pg=PP1&dq=canvas+and+panel+in+unity&ots=e9BNnJ00vs&sig=NsXiv7A5D32eOxWcU_nUmyZE5pQ&redir_esc=y#v=onepage&q=canvas%20and%20panel%20in%20unity&f=false
13. Levels And How To Get Close To Player: https://books.google.com.tr/books?id=kcC9AAAAQBAJ&lpg=PP3&ots=YLobIDrAXK&dq=how%20to%20get%20close%20to%20player%20in%20unity&lr&pg=PA206#v=snippet&q=get%20close%20to%20player&f=false