T.C.

FIRAT ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

YMH313 – İşletim Sistemleri

Ders Proje Dokümantasyonu

Düzensiz Hayat

Proje Ekibi

170541034 – Meliha GÜLBAY

Ekim – 2019

1. Giriş

1.1 Projenin Amacı

Oyun insanlara gerçek bir deneyim sunarak oyunun daha sürdürülebilir olmasını sağlar. Bu oyunu oynayan kişiler karakterin yerine bizzat kendini koyacaklar. Bir zamandan sonra karakterin hayatı düzene girmeye başladıkça, bunun insanlar içinde teşvik niteliğinde olması amaçlanmaktadır. Oyunun kişinin odaklanma kabiliyetini arttırması ve refleksleri geliştirmesi amaçlanmaktadır.

1.2 Projenin Kapsamı

Oyunumuzda kullandığımız karakterin düzensiz bir hayatı vardır. Hiçbir plan yapmaz ve sürekli olarak gideceği yerlere geç kalır. Bu düzensizliği, iş hayatını da olumsuz etkilemektedir. Bu durumu, evinin iş yerine uzak olmasını bahane ederek geçiştirmektedir ve bu yüzden iş yerine ne kadar yakın bir eve taşınırsa, o kadar işlerinin düzene gireceğine inanmaktadır. İşe sürekli geç kalan karakterimiz artık durumu abartmaktadır. Ve sabahları servisi de kaçırmaktadır. Patronundan son uyarısını alan karakterimiz tekrar işe geç kalırsa kovulacaktır.

Karakterimiz için hem para biriktirip iş yerine yakın bir yerden ev almamız gerekiyor. Hem de sabahları servisi kaçırdığımız için işyerine koşarak gitmemiz gerekiyor. Tabi ki de koştuğumuz için enerjimiz tükenmektedir. Koşu esnasında yol üzerinden su, enerji ve yiyecek gibi elementleri toplayarak karakterimizin enerjisini doldurabiliriz.

Bu tür elementlerin türüne göre parası da değişmektedir. Hem para biriktirip iş yerine yakın ev alma çabası hem de işe geç kalmamak için yolda yiyecek, içecek alarak paranın azalması ile durumun korunması gerekmektedir.

2. Proje Planı

2.1 Giriş

Proje bitişi tahmini olarak iki ay da bitirilmesi planlanmıştır. Bu süre içerisinde Unity öğrenilmesi, araştırılması ve nasıl bir yol izlenilmesi gerekildiği araştırılıp uygulamaya başlanılması planlanmıştır.

2.2 Zaman - İş Planı

Tarih	Versiyon	Yapılanlar
27 Ekim – 03 Kasım / 2019	v1	Başlangıç, Bitiş, Hakkında, Kazanma, Kaybetme sahnelerinin 2 boyutlu tasarımları ve BolumYoneticisi.css kodu yazıldı ve bunun sayesinde sahneler arası geçiş işlemi gerçekleştirildi.
03 Kasım – 10 Kasım / 2019	v2	Oyun için prefablar bulundu ve projeye entegre edildi. Bulunan prefablar ile oyunun ilk görünüm için ekranı tasarlandı. Karakter seçildi ve projeye eklendi.
10 Kasım – 17 Kasım / 2019	v3	Karakterin durma animasyonu, yürüyüş animasyonu ve koşma animasyonu

		dinamik 0 ile 1 değerler arasında geçişli animasyon şeklinde AnimatorController içinde tasarlandı ve karakterin ana animasyonu olarak seçildi. Bunun ile ilgili testler gerçekleştirildi.
24 Kasım – 1 Aralık / 2019	v4	Karakterin yürüdüğü zaman karşısına çıkacak yol haritaları için 6 çeşit haritası tasarlandı. Bu tasarlanan haritaların durumlara göre hangisinin görüneceği dinamik olarak kodları yazıldı.
8 Aralık – 15 Aralık / 2019	v5	Zıplama, Dans etme, ölme gibi animasyonlar karakterin AnimatorController'ına eklendi ve bunun için KarakterKontrolcüsü kodu düzenli şekilde yazıldı. Bu animasyonlar oynatılırken ses efektleride animasyonlara dahil edildi.
15 Aralık – 22 Aralık / 2019	v6	Grafiklerin birbiri ile bağlantısı sağlanarak oyun olarak görünmesi için güç özelliği eklenerek zamana bağlandı. Bu zamana göre de oyunun bitiş haritası görüntülenerek oyunun bitişi sağlandı. Oyun esnasında engellerin ve güç elementlerininde yönetilmesi kodlara eklendi.

3. Genel Açıklamalar

3.1 Karakter Özellikleri

"Düzensiz Hayat" oyunundaki karakterin yapabildikleri şu şekilde sıralanabilir:

- Zıplayabilmek
- Sağa geçmek
- Sola geçmek
- Koşabilmek
- Yürüyebilmek
- Durabilmek
- Dans edebilmek
- Enerji Toplayabilmek

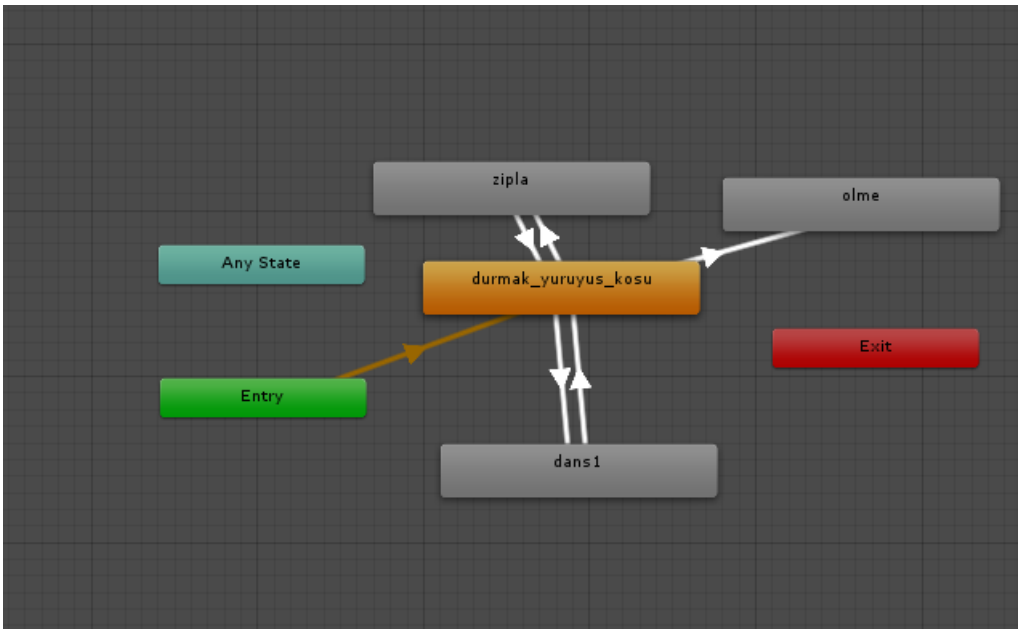
3.2 Proje Efektleri

Oyunda kullanılan ses efektleri şu şekildedir:

- Rüzgar sesi
- Zıplama sesi
- Yürüme sesi
- Koşma sesi
- Kar temasında yere basma sesi
- Çöl temasında yere basma sesi

3.3 Proje Animasyonları

Karakter animasyonlarının nasıl çalıştıkları:



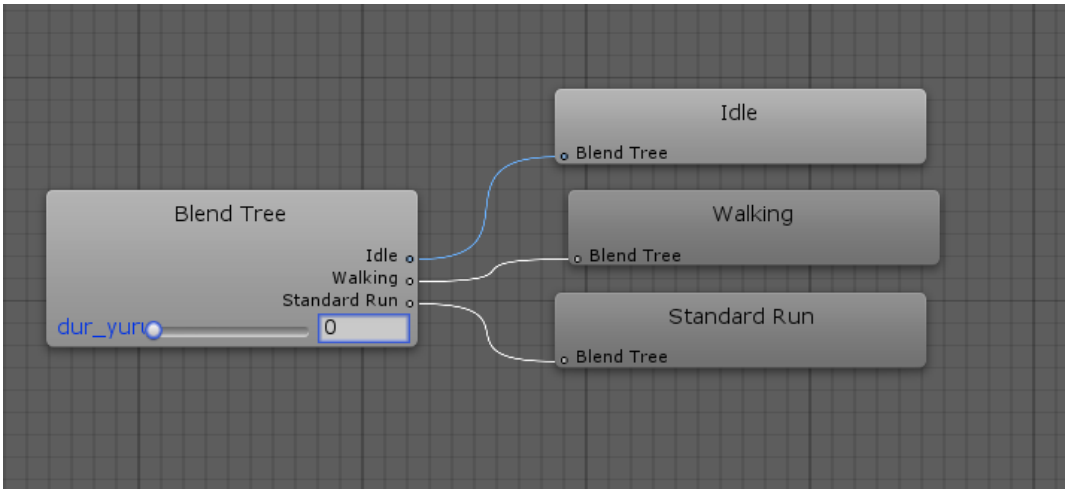
Şekil -1

Oyun başladığında karakterin Şekil-1 de göründüğü gibi AnimatorController özelliği çalışmaya başlar. Bu özellik şu şekilde çalışmaktadır. Başlangıçta durmak_yuruyus_kosu blend tree si sonsuz döngü halinde çalışır.

Bu blend tree sonsuz olarak bu parametreye göre çalışır. Eğer AnimatorController da istenilecek farklı durumlar olursa Şekil-1 de göründüğü gibi animasyon geçişi de gerçekleşecektir. Örneğin zıplamak için click özelliği olur ise blend tree animasyonu durur zıplama animasyonu çalışır ve animasyon işlemi bittikten sonra ana animasyona geçiş yapar.

Eğer oyun içerisinde ölme işlemi olursa öldüğünü belirten checkbox işaretlenir. Bu işlem yapılırsa ölme animasyonu çalıştırılır. Fakat bu animasyondan blend tree animasyonuna geçiş oku bulunmamaktadır. Yani animatorController un animasyon işlemi biter.

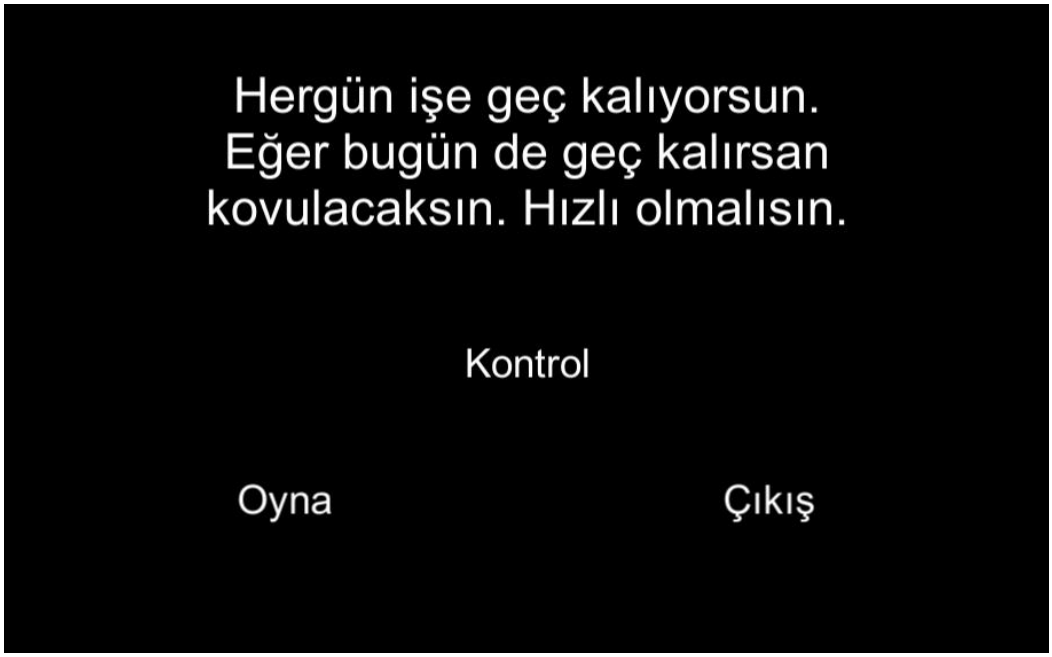
Eğer oyun içerisinde dans animasyonu çalıştırılırsa çalışır ve dans işlemi bitene kadar beklemektedir. Dans işlemi bittikten sonra blend tree animasyonuna geri geçiş yapar ve o esnada değer ne ise o değer ile blend tree animasyonu devam eder. Örneğin önceden koşuyor ise koşmaya devam eder.



Şekil - 2

Bu blend tree Şekil – 2 de göründüğü gibi 0 ile 1 arasında parametre alır. 0 değeri var ise idle animasyonu yani normal duruş animasyonu çalışır. 0 dan 0.5 e geçerken hafiften yürüme animasyonu devreye girer ama durma animasyonu da orana göre çalışır. Parametre eğer 0.5 değer alır ise sadece yürüme animasyonu çalışır. 0.5 ile 1 arasında değer alır ise koşma animasyonu çalışır ve yürüme animasyonu hafif orana göre çalışır durumdadır. Parametre eğer 1 değeri alır ise sadece koşma animasyonu çalışır.

3.4 Oyun Nasıl Çalışır ?



Şekil-3:Oyun başlangıç ekranı

Oyunu başlattığımızda karşımıza Şekil-3 te göründüğü gibi Oyun başlangıç ekranı çıkıyor. Bu ekranda oyun hakkında kısa bilgi görünmektedir.

Şekil-4: Hakkında ekranı

Şekil-4 te görünen Hakkında ekranında Oyun kontrol bilgileri görünmektedir.

Şekil-5 : Karakter

Şekil-6 : Oyun sahnesi

Oyunu başlattığımız zaman Şekil-6 da ki sahne gibi görünmektedir ve oynamaya hazır durumdadır.

Şekil-7 : Oyun sahnesinde dans animasyonu

Oyun içerisinde T tuşuna bastığımız zaman karakterimiz dans eder ve dans animasyonu bitene kadar herhangi bir animasyon tetiklemesi çalışmamaktadır.

Şekil-8 : Oyun sahnesinde engeller ve güç elementi görünümleri

Oyunu oynamaya başladığımız zaman ilerledikçe karşımıza engeller çıkmaktadır. Engellerden kurtulabilmek için sağa sola geçmemiz gerekir ya da zıplamamız gerekir. Gücümüz azaldığı zaman karşımıza çıkan güçleri almamız gerekir. Çünkü güç ile zaman birbiri ile uyumlu çalışır.

Eğer güç azalırsa, zaman geri sayımı durduracaktır. Geri sayım olmazsa karşımıza bitiş çizgisi gelmeyecektir. Bu yüzden hem zaman azalana kadar engellerden kurtularak koşmamız, hem de gücümüzü toplamamız gerekir.

Şekil-9 : Koşu animasyonu

W tuşu ile hızımızı arttırır. S tuşu ile hızımızı azaltırız. Hızımız 0 ile 1 arasında değişebilmektedir. 0 hızında ise durma animasyonu, 0.5 hızında ise yürüyüş animasyonu, 1 hızında ise Şekil-9 da göründüğü gibi koşma animasyonun çalışmaktadır.

Şekil-10 : Ölme animasyonu

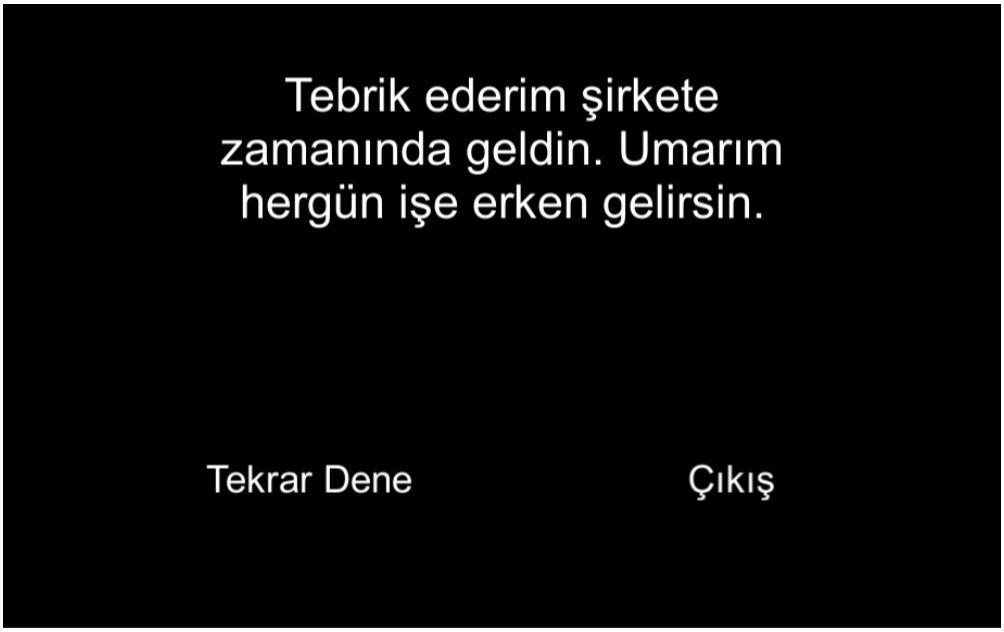
Karşımıza çıkan engellerin ortak tag değeri vardır. Oyunumuzun çalışma mantığında karakterin çarptığı cisimin tag ifadesi istediğimiz özellik ise karakterimizin Şekil-10 da göründüğü gibi ölme animasyonu çalışmaktadır.

Şekil-11 : Karakter öldüğü zaman karşısına çıkan ekran

Ölme animasyonu çalıştıktan belirlediğimiz süre sonra karşımıza öldüğümüze dair sahne gelmektedir. Bu sayede sahnemiz değiştiği için oyunun bütün işleyişi bitmektedir.

Şekil-12 : Oyun bitişe yakın sahne görüntüsü

Eğer oyunu oynarken gücümüzü toplayarak zamanımızı bitirene kadar engellere çarpmadan koşma animasyonu ile ilerlenirse karşımıza finish haritası yüklenir ve bayrağı geçtikten sonra belirlediğimiz süre sonra oyun bitecektir. Oyunun biteceği süre koşma animasyonu ile hesaplanıp şirketin kapısına gelene kadar ki süre test edilerek belirlenmiştir. Ve karakter kapıya geldiğinde oyun bitmiş gibi görünmektedir.



Şekil-13 : Oyun bitiş ekranı

Oyun bittiğinde karşımıza çıkan Şekil-13 teki sahne sayesinde oyunu kazandığımızı ve oyunun bittiği bilgisini görebilmekteyiz.

3.5 Oyun kod bloklarının incelenmesi:

```
void KarakterZemineDegiyor()
{
    float hiz = karakterKontrolcusu.animasyonHizi; // karakterKontrolcusu class ın içindeki animasyonHizi değişkeninin değerini hafızaya aktarıyoruz.

    if (hiz > 0) // eğer bir hıza sahip ise
    {
        if (hiz > 0.5f) // eğer hızı yarının üstünde ise
        {
            mevcutAudioClip = kosmaSesi; // kosma sesi yap
        }
        else
        {
            mevcutAudioClip = yuruyusSesi; // yürüme sesi yap
        }

    }
    else // eğer bir hıza sahip değil ise ve Space tuşuna basılmadı ise
    {
        mevcutAudioClip = null; // sesi yok et
    }

    if (Input.GetKeyDown(KeyCode.Space))mevcutAudioClip = ziplamaSesi; // zıpladıysa sesi güncelle

    if (audioSource.clip != mevcutAudioClip) // eğer o an çalan clip ile mevcut değişen ses aynı değil ise
    {
        SesKlibiniTetiklet(mevcutAudioClip); // sesi güncelle
    }
}
```

Şeki-14

KarakterKontrolcusu classından aldığımız animasyon hızına göre karakterimizin ses efektlerini yönetiyoruz. Eğer hız 0 dan büyükse ve 0.5 ten de büyükse koşu sesi, 0 dan büyükse ve 0.5 ten küçük ise yürüyüş sesi çalışsın. Eğer 0 ise sesi yok eder. Space tuşuna basar ise zıplama sesi yüklenir.

```
public class KarakterDusmanIliskisi : MonoBehaviour
{
    Animator animator;
    KarakterKontrolcusu karakterKontrolcusu;
    OyunBitirmeIslemi oyunBitirmeIslemi;

    // Start is called before the first frame update
    void Start()
    {
        animator = gameObject.GetComponent<Animator>();
        karakterKontrolcusu = gameObject.GetComponent<KarakterKontrolcusu>();
        oyunBitirmeIslemi = gameObject.GetComponent<OyunBitirmeIslemi>();
    }

    // Update is called once per frame
    void Update()
    {

    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.tag=="Dusman")
        {
            animator.SetBool("olsun_mu", true);
            karakterKontrolcusu.KarakterKontrolEtme = false;
            StartCoroutine(oyunBitirmeIslemi.OyunuBitir("Dusman", true,4));//4 saniye sonra ölme sebebi "Dusman" olarak oyunu bitir (true)
        }
    }
}
```

Şekil – 15

Eğer karakterimiz bir cisime değerse ve cisimin tag özelliği “Dusman” ise karakterin animatorController’ına olsun_mu değişkenini true olarak gönderiyoruz. KarakterKontrolEtme özelliğini false yapıyoruz ve bu sayede karakter artık kontrol edilemiyor. OyunBitirme fonksiyonunu “Dusman” özelliği ile 4 saniye olarak çalıştırıyoruz ve 4 saniye sonra “Dusman” özelliğinden dolayı oyun kaybetme sahnesi yükleniyor.

```
private void OnTriggerExit(Collider other)
{
    if(other.tag == "YolunBasi")
    {
        if (geriSayim <= 0)
        {
            yeniYolEkle(level1Bitis, other);
        }
        else
        {
            GameObject yol = yeniYolEkle(level1[Random.Range(0, level1.Length)], other);
            yolaDusmanlariEkle(yol);
        }
    }

    if (other.tag == "YolunSonu")
    {
        Destroy(other.transform.parent.gameObject);
    }
}
```

Şekil – 16

Eğer değdiğimiz trigger "YolunBasi" ise zamanımız da 0 oldu ise karşımıza çıkacak harita bitiş haritası olacaktır. Fakat zamanımız henüz 0 olmadı ise karşımıza çıkacak harita, haritalar listesinde ne kadar harita mevcut ta var ise o kadarı rastgele seçilerek karşımıza çıkar ve bu sayede gerçekçi oyun deneyimi yaşamış oluruz. "YolunSonu" tagı her haritanın sonunda bulunmaktadır. Bu yere geldiğimiz zaman bir önceki haritanın yok edilmesini sağlayabiliriz. Bu sayede oyunumuzda gereksiz objeler kalkmış olur ve hiçbir şekilde yavaşlama yaşamayız.

4. Sonuç

- Sahneler arası geçişin nasıl olduğunu öğrendim.
- Hazır bir karakteri projeye dahil etmeyi öğrendim.
- Karaktere animasyonun nasıl eklendiğini öğrendim.
- Birden fazla haritanın nasıl bir şekilde, hangi durumlarda karşımıza çıkartıp, gereksiz yerlerin ise hangi durumlarda silinmesi gerektiğini öğrendim.
  Tag: Bilgi sistemlerinde bir etiket, bir bilgi parçasına atanan bir anahtar kelime ve ya terimdir.
  Collider: Her nesnenin fiziksel olarak algılanması için collider componentine ihtiyaç vardır. Bu özellik sayesinde örneğin karakterin bir cisime fiziksel temas uygulaması sağlanır ve bu fiziksel temas sonucu hangi nesnenin hangi nesneye değdiği algılanır.
  Trigger: Collider özelliği ile algılaması aynıdır. Fakat nesneye fiziksel temas uygulanamaz. Örneğin nesnenin üzerine gittiğinizde nesnenin içerisinden geçersiniz. Bu özellik genellikle görünmeyen fakat algılanmak istenen durumlarda kullanılır.
  Component: Her nesnenin bir özelliği olur. Örneğin kod özelliği, fiziksel özelliği, transform özelliği, collider özelliği, ses özelliği gibi özellikleri vardır. Bu özelliklerin her biri bir Component'tir.

5. Kaynaklar

- https://docs.unity3d.com/ScriptReference/GameObject-tag.html
- https://assetstore.unity.com/
- https://www.mixamo.com/#/?page=1&type=Character