



Beykent Üniversitesi  
Yazılım Mühendisliği Bölümü  
Yazılım Mühendisliği Tasarım Projesi

2024 – BAHAR  
WitDark Dökümantasyonu

Whispers in the Dark (WitDark)  
Grup Üyeleri

2003013062 - Beyzanur Yakut  
2003013024 - Gizem Güven  
2003013014 - Mustafa Örnek  
2103013800 - Ömer Faruk Dinçaslan  
2003013036 - Zeynep Ciplak

# İÇİNDEKİLER

<b>İÇİNDEKİLER</b>	<b>1</b>
<b>ŞEKİLLER LİSTESİ</b>	<b>3</b>
1. Şekil - 1: Giriş Sahnesindeki Oynanışa Dair Görünüm	3
2. Şekil - 2: Giriş Sahnesi Tasarımı	3
3. Şekil - 3: Radar Nesnesinin Görünümü	3
4. Şekil - 4: Radar Nesnesinin Diğer Nesnelerle Etkileşimine Ait Bir Görüntü	3
5. Şekil - 5 : Öğretici Modun Alanının Tasarımı	3
6. Şekil - 6: Envanter menüsünün görünümü sağ üstte	3
7. Şekil - 7: Oyuncunun Envanter Menüyü Kullanabilmesini Sağlayan Kod	3
8. Şekil - 8: Coin(Para)'lerin Toplanarak Envantere Eklenmesini Sağlayan Kod (Bu kod başka bir eşyaya da uyarlanabilir)	3
9. Şekil - 9: Envanter Menüsünde Coin(Para)lerin Sayısını Değiştirmeyi Sağlayan Kod (Bu kod başka bir eşyaya da uyarlanabilir)	3
10. Şekil - 10: Envanter Menüsünde Sword(Kılıç)'ların Sayısını Değiştirmeyi Sağlayan Kod (Bu kod başka bir eşyaya da uyarlanabilir)	3
11. Şekil - 11: Ayak Seslerinin FMOD for Unity ile Gösterimi	3
12. Şekil - 12: Ayak Seslerinin Bulunulan Bölgeye Göre Değişmesini Sağlayan Kod	3
13. Şekil - 13: Ayak Sesleri Kod ( Devamı - 1)	3
14. Şekil - 14: Ayak Sesleri Kod ( Devamı - 2)	3
15. Şekil - 15: Seslerin Yönetilmesini Sağlayan Kod	3
16. Şekil - 16: Seslerin Yönetilmesini Sağlayan Kod(Devamı)	3
17. Şekil - 17: Genel Ses Ayarlarını Yapan Kod	3
18. Şekil - 18: FMOD Tasarım Arayüzü	3
19. Şekil - 19: Köy Alanının Genel Görünümü	3
20. Şekil-20: Köy Alanında Oyuncunun Demirci Karakteri İle Etkileşimi	3
21. Şekil-21: Köy Alanında Oyuncunun Demirci Karakteri İle Diyaloğu	3
22. Şekil - 22: Köy Alanında Oyuncunun Marangoz Karakteri İle Etkileşimi	3
23. Şekil - 23: Köy Alanında Oyuncunun Marangoz Karakteri İle Diyaloğu	4
24. Şekil - 24: Oyuncunun NPC İle Etkileşime Girme Ve Devam Ettirme Kodu	4
25. Şekil - 25: Oyuncunun NPC İle Etkileşime Girme Ve Devam Ettirme Kodu	4
26. Şekil - 26: Yan Oyun Alanı Tasarımı	4
27. Şekil - 27: Yan Oyun Alanı Eşyaya ait Kod	4
28. Şekil - 28: Gardiyen Sistemi	4
29. Şekil - 29: Gardiyen Sistemi Kodu	4
30. Şekil - 30: WitDark Ana Menü	4
31. Şekil - 31: WitDark Ayarlar Menüsü	4
32. Şekil - 32: WitDark Ana Menü Kodu	4
33. Şekil - 33: WitDark Ayarlar Menüsü Kodu	4
34. Şekil - 34: WitDark Duraklatma Menüsü	4
35. Şekil - 35: WitDark Duraklatma Menüsü Kodu	4
36. Şekil - 36: WitDark Açık Dünya Alanı	4

37. Şekil - 37: Kale Bölgesinin Oyun İçi Gösterimi	4
38. Şekil - 38: Kılıç, Ok ve Asa Kullanımına Ait Oynanış Görüntüleri	4
39. Şekil - 39: Save Sistemi Kodu	4

## **RESİM LİSTESİ**

1. Şekil - 1: Giriş Sahnesindeki Oynanışa Dair Görünüm	5
2. Şekil - 2: Giriş Sahnesi Tasarımı	5
3. Şekil - 3: Radar Nesnesinin Görünümü	5
4. Şekil - 4: Radar Nesnesinin Diğer Nesnelerle Etkileşimine Ait Bir Görüntü	5
5. Şekil - 5 : Öğretici Modun Alanının Tasarımı	5
6. Şekil - 6: Envanter Menüsünün Görünümü	5
7. Şekil - 11: Ayak Seslerinin FMOD for Unity ile Gösterimi	5
8. Şekil - 19: Köy Alanının Genel Görünümü	5
9. Şekil-20: Köy Alanında Oyuncunun Demirci Karakteri İle Etkileşimi	5
10. Şekil-21: Köy Alanında Oyuncunun Demirci Karakteri İle Diyaloğu	5
11. Şekil - 22: Köy Alanında Oyuncunun Marangoz Karakteri İle Etkileşimi	5
12. Şekil - 23: Köy Alanında Oyuncunun Marangoz Karakteri İle Diyaloğu	5
13. Şekil - 26: Yan Oyun Alani Tasarımı	5
14. Şekil - 28: Gardiyen Sistemi	5
15. Şekil - 30: WitDark Ana Menü	5
16. Şekil - 31: WitDark Ayarlar Menüsü	5
17. Şekil - 34: WitDark Duraklatma Menüsü	5
18. Şekil - 36: WitDark Açık Dünya Alanı	5
19. Şekil - 37: Kale Bölgesinin Oyun İçi Gösterimi	5

## **KISALTMA LİSTESİ**

### **ÖZET**

### **1. GİRİŞ**

1.1 MOTİVASYON	7
1.2 HİKAYE	7

### **2. LİTERATÜR TARAMASI**

2.1 EUROFLY	9
2.2 MIST WORLD	9

### **3. YÖNTEM**

3.1 GİRİŞ SAHNESİ TASARIMI (Beyzanur YAKUT, Ömer Faruk DİNÇASLAN)	10
Şekil - 1: Giriş Sahnesindeki Oynanışa Dair Görünüm	10
Şekil - 2: Giriş Sahnesi Tasarımı	11

3.2 TEMEL OYNANIŞ MEKANİKLERİ VE RADAR NESNESİNİN OLUŞTURULMASI (Beyzanur YAKUT, Ömer Faruk DİNÇASLAN)	11
---	----

Şekil - 3: Radar Nesnesinin Görünümü	12
--------------------------------------	----

Şekil - 4: Radar Nesnesinin Diğer Nesnelerle Etkileşimine Ait Bir Görüntü	12
---	----

3.3 TUTORİAL SAHNESİ TASARIMI (Beyzanur YAKUT, Ömer Faruk DİNÇASLAN)	13
--	----

Şekil - 5 : Öğretici Modun Alanının Tasarımı	14
--	----

3.4 ENVANTERİN OLUŞTURULMASI (Beyzanur YAKUT)	15
---	----

Şekil - 6: Envanter Menüsünün Görünümü	15
--	----

Şekil - 7: Oyuncunun Envanter Menüyü Kullanabilmesini Sağlayan Kod	16
--	----

Şekil - 8: Coin(Para)'lerin Toplanarak Envantere Eklenmesini Sağlayan Kod (Bu kod başka bir eşyaya da uyarlanabilir)	17
Şekil - 9: Envanter Menüsünde Coin(Para)lerin Sayısını Değiştirmeyi Sağlayan Kod (Bu kod başka bir eşyaya da uyarlanabilir)	18
Şekil - 10: Envanter Menüsünde Sword(Kılıç)'ların Sayısını Değiştirmeyi Sağlayan Kod (Bu kod başka bir eşyaya da uyarlanabilir)	19
<b>3.5 SES TASARIMI (Mustafa ÖRNEK)</b>	<b>20</b>
<b>3.5.1 AYAK SESLERİ</b>	<b>20</b>
Şekil - 11: Ayak Seslerinin FMOD for Unity ile Gösterimi	20
Şekil - 12: Ayak Seslerinin Bulunulan Bölgeye Göre Değişmesini Sağlayan Kod	21
Şekil - 13: Ayak Sesleri Kod ( Devamı - 1)	22
Şekil - 14: Ayak Sesleri Kod ( Devamı - 2)	23
<b>3.5.2 AUDIO MANAGER</b>	<b>23</b>
Şekil - 15: Seslerin Yönetilmesini Sağlayan Kod	24
Şekil - 16: Seslerin Yönetilmesini Sağlayan Kod(Devamı)	25
<b>3.5.3 FMOD EVENTS</b>	<b>26</b>
Şekil - 17: Genel Ses Ayarlarını Yapan Kod	26
Şekil - 18: FMOD Tasarım Arayüzü	27
<b>3.6 KÖY ALANI TASARIMI (Gizem GÜVEN)</b>	<b>28</b>
Şekil - 19: Köy Alanının Genel Görünümü	28
<b>3.7 NPC'LERİN OLUŞTURULMASI (Gizem GÜVEN)</b>	<b>29</b>
Şekil-20: Köy Alanında Oyuncunun Demirci Karakteri İle Etkileşimi	30
Şekil-21: Köy Alanında Oyuncunun Demirci Karakteri İle Diyalogu	30
Şekil - 22: Köy Alanında Oyuncunun Marangoz Karakteri İle Etkileşimi	31
Şekil - 23: Köy Alanında Oyuncunun Marangoz Karakteri İle Diyalogu	31
Şekil - 24: Oyuncunun NPC İle Etkileşime Girme Ve Devam Ettirme Kodu	32
Şekil - 25: Oyuncunun NPC İle Etkileşime Girme Ve Devam Ettirme Kodu	33
<b>3.8 YAN OYUN ALANI TASARIMI (Zeynep CIPLAK)</b>	<b>34</b>
Şekil - 26: Yan Oyun Alanı Tasarımı	34
Şekil - 27: Yan Oyun Alanı Eşyaya ait Kod	35
<b>3.9 GARDİYAN SİSTEMİNİN EKLENMESİ (Zeynep CIPLAK)</b>	<b>36</b>
Şekil - 28: Gardıyan Sistemi	36
Şekil - 29: Gardıyan Sistemi Kodu	37
<b>3.10 ANA MENÜ TASARIMI (Gizem GÜVEN - Zeynep CIPLAK)</b>	<b>38</b>
Şekil - 30: WitDark Ana Menü	38
Şekil - 31: WitDark Ayarlar Menüsü	39
Şekil - 32: WitDark Ana Menü Kodu	39
Şekil - 33: WitDark Ayarlar Menüsü Kodu	40
<b>3.11 DURAKLATMA MENÜSÜ TASARIMI (Gizem GÜVEN - Zeynep CIPLAK)</b>	<b>41</b>
Şekil - 34: WitDark Duraklatma Menüsü	41
Şekil - 35: WitDark Duraklatma Menüsü Kodu	42
<b>3.12 AÇIK DÜNYA HARİTASI (Ömer Faruk DİNÇASLAN)</b>	<b>43</b>
Şekil - 36: WitDark Açık Dünya Alanı	43
<b>3.13 KALE BÖLGESİ (Ömer Faruk DİNÇASLAN)</b>	<b>44</b>
Şekil - 37: Kale Bölgesinin Oyun İçi Gösterimi	44

3.14 DÖVÜŞ MEKANIĞI (Ömer Faruk DİNÇASLAN)	46
Şekil - 38: Kılıç, Ok ve Asa Kullanımına Ait Oynanış Görüntüleri	47
3.15 SAVE SİSTEMİ (Gizem GÜVEN - Zeynep CIPLAK)	48
Şekil - 39: Save Sistemi Kodu	49
<b>4. SONUÇLAR VE TARTIŞMA</b>	<b>49</b>
<b>5. KAYNAKÇA</b>	<b>51</b>

## **ŞEKİLLER LİSTESİ**

1. *Şekil - 1: Giriş Sahnesindeki Oynanışa Dair Görünüm*
2. *Şekil - 2: Giriş Sahnesi Tasarımı*
3. *Şekil - 3: Radar Nesnesinin Görünümü*
4. *Şekil - 4: Radar Nesnesinin Diğer Nesnelerle Etkileşimine Ait Bir Görüntü*
5. *Şekil - 5 : Öğretici Modun Alanının Tasarımı*
6. *Şekil - 6: Envanter menüsünün görünümü sağ üstte*
7. *Şekil - 7: Oyuncunun Envanter Menüsü Kullanabilmesini Sağlayan Kod*
8. *Şekil - 8: Coin(Para)'lerin Toplanarak Envantere Eklenmesini Sağlayan Kod (Bu kod başka bir eşyaya da uyarlanabilir)*
9. *Şekil - 9: Envanter Menüsünde Coin(Para)lerin Sayısını Değiştirmeyi Sağlayan Kod (Bu kod başka bir eşyaya da uyarlanabilir)*
10. *Şekil - 10: Envanter Menüsünde Sword(Kılıç)'ların Sayısını Değiştirmeyi Sağlayan Kod (Bu kod başka bir eşyaya da uyarlanabilir)*
11. *Şekil - 11: Ayak Seslerinin FMOD for Unity ile Gösterimi*
12. *Şekil - 12: Ayak Seslerinin Bulunulan Bölgeye Göre Değişmesini Sağlayan Kod*
13. *Şekil - 13: Ayak Sesleri Kod ( Devamı - 1)*
14. *Şekil - 14: Ayak Sesleri Kod ( Devamı - 2)*
15. *Şekil - 15: Seslerin Yönetilmesini Sağlayan Kod*
16. *Şekil - 16: Seslerin Yönetilmesini Sağlayan Kod(Devamı)*
17. *Şekil - 17: Genel Ses Ayarlarını Yapan Kod*
18. *Şekil - 18: FMOD Tasarım Arayüzü*
19. *Şekil - 19: Köy Alanının Genel Görünümü*
20. *Şekil-20: Köy Alanında Oyuncunun Demirci Karakteri İle Etkileşimi*
21. *Şekil-21: Köy Alanında Oyuncunun Demirci Karakteri İle Diyalogu*
22. *Şekil - 22: Köy Alanında Oyuncunun Marangoz Karakteri İle Etkileşimi*

23. *Şekil - 23: Köy Alanında Oyuncunun Marangoz Karakteri İle Diyalogu*
24. *Şekil - 24: Oyuncunun NPC İle Etkileşime Girme Ve Devam Ettirme Kodu*
25. *Şekil - 25: Oyuncunun NPC İle Etkileşime Girme Ve Devam Ettirme Kodu*
26. *Şekil - 26: Yan Oyun Alanı Tasarımı*
27. *Şekil - 27: Yan Oyun Alanı Eşyaya ait Kod*
28. *Şekil - 28: Gardiyen Sistemi*
29. *Şekil - 29: Gardiyen Sistemi Kodu*
30. *Şekil - 30: WitDark Ana Menü*
31. *Şekil - 31: WitDark Ayarlar Menüsü*
32. *Şekil - 32: WitDark Ana Menü Kodu*
33. *Şekil - 33: WitDark Ayarlar Menüsü Kodu*
34. *Şekil - 34: WitDark Duraklatma Menüsü*
35. *Şekil - 35: WitDark Duraklatma Menüsü Kodu*
36. *Şekil - 36: WitDark Açık Dünya Alanı*
37. *Şekil - 37: Kale Bölgesinin Oyun İçi Gösterimi*
38. *Şekil - 38: Kılıç, Ok ve Asa Kullanımına Ait Oynanış Görüntüleri*
39. *Şekil - 39: Save Sistemi Kodu*

## **RESİM LİSTESİ**

1. *Şekil - 1: Giriş Sahnesindeki Oynanışa Dair Görünüm*
2. *Şekil - 2: Giriş Sahnesi Tasarımı*
3. *Şekil - 3: Radar Nesnesinin Görünümü*
4. *Şekil - 4: Radar Nesnesinin Diğer Nesnelerle Etkileşimine Ait Bir Görüntü*
5. *Şekil - 5 : Öğretici Modun Alanının Tasarımı*
6. *Şekil - 6: Envanter Menüsünün Görünümü*
7. *Şekil - 11: Ayak Seslerinin FMOD for Unity ile Gösterimi*
8. *Şekil - 19: Köy Alanının Genel Görünümü*
9. *Şekil-20: Köy Alanında Oyuncunun Demirci Karakteri İle Etkileşimi*
10. *Şekil-21: Köy Alanında Oyuncunun Demirci Karakteri İle Diyalogu*
11. *Şekil - 22: Köy Alanında Oyuncunun Marangoz Karakteri İle Etkileşimi*
12. *Şekil - 23: Köy Alanında Oyuncunun Marangoz Karakteri İle Diyalogu*
13. *Şekil - 26: Yan Oyun Alanı Tasarımı*
14. *Şekil - 28: Gardiyen Sistemi*
15. *Şekil - 30: WitDark Ana Menü*
16. *Şekil - 31: WitDark Ayarlar Menüsü*
17. *Şekil - 34: WitDark Duraklatma Menüsü*
18. *Şekil - 36: WitDark Açık Dünya Alanı*
19. *Şekil - 37: Kale Bölgesinin Oyun İçi Gösterimi*
20. *Şekil - 38: Kılıç, Ok ve Asa Kullanımına Ait Oynanış Görüntüleri*

## **KISALTMA LİSTESİ**

WitDark: Whispers In The Dark

RPG: Role Playing Game(Rol Yapma Oyunu)

MMORPG: Massively Multiplayer Online Role-Playing Game(Devasa Çok Oyunculu  
Çevrimiçi Rol Yapma Oyunu)

NPC: Non Player Character(Oyuncu Olmayan Karakter)

## **ÖZET**

WitDark(Whispers in the Dark) projesinin motivasyon kaynağı, görme engelli bireylerin rekreatif aktivitelere erişimlerini artırmak ve onların dijital oyun deneyimlerini geliştirebilmek, içerik bakımından engelin varlığı ortadan kaldırılarak genel kitleye hizmet eden bir hikaye oluşturarak görme engelli olmayan bireylerin de ilgisine sunulacak tarzda bir oyun tasarlarkarak görme engelli olmayan oyuncu kitlesine bu engel hakkında empati yapabilme gücü kazandırabilmek olan, özünde ise görme engelli bireyler için geliştirilmiş bir oyun projesidir. Projemizde, erişilebilirlik standartlarına bağlı olarak işitsel geri bildirim ve kullanıcı arayüzü optimizasyonu gibi yöntemler kullanılarak, oynanışı uygun bir oyun tasarımları gerçekleştirilmiştir. Görme engelli kullanıcıların geri bildirimleri düzenli olarak alınmış ve projede sürekli olarak iyileştirme yapılabilmesi mümkün hale gelmiş, bu geri bildirimlere dayanılarak geliştirme süreci gerçekleştirilmiştir. Sonuç olarak görme engelli bireylerin sese olan duyarlığını göz önünde bulundurularak hassaslaşdırılmış geliştirme yöntemlerine başvurulması gereği sonucuna varılmış olup proje bu yönde geliştirilmiştir.

## **1. GİRİŞ**

WitDark, görme engelli bireyler için geliştirilmiş bir video oyunu sunmaktadır. Amacımız, görme engelli bireylerin dijital eğlence dünyasında daha fazla yer almasını sağlamak ve onlara keyifli, erişilebilir bir oyun deneyimi sunmaktır. Projemiz, RPG(Rol yapma oyunu) tarzda ve açık dünya türünde bir oyun olarak tasarlanmıştır. Çalışma konusu olarak tercih edilen alan, uygulama zorluğu açısından ilk olması bakımından mantıklı bir seçim gibi görünmese de oluşturulan hikaye nezdinde oyuncuya verilen etkinin tesirini artırmak amacıyla bu tarzda tercih edilmiştir.

### **1.1 MOTİVASYON**

Görme engelli bireyler için oluşturulmuş siyah ekran ve ses tasarımları ile geliştirilen oyunların aksine hikayeyi baz alarak farkındalık kazandırbilmek amacıyla bütün oyuncu kitlesine odaklanılmıştır. Bu şekilde görme engelli bireylerin hayat kalitesine olumlu anlamda etki etmesi amaçlanmıştır.

## 1.2 HİKAYE

Ana karakter bir ormanda oyuna başlar. Hava kararmak üzereyken kardeşinin hastalığı için bitki toplamaya çıktığı ormandan geri dönmeye çalışmaktadır. Şiddetli yağmurun başlaması üzerine yolunu kestiremez ve öünü görmeden koşmaya başlar. Ancak saat çok geç olmuştur ve dışarıdaki insanların eve girmesi için yapılan uyarı olan çan sesini duymaya başlamıştır. Çan sesinden sonra yaratıklar ortaya çıkmaya başlar ve karakterimiz duyduğu yaratık seslerinden, kopan firtınadan etkilenerek güçsüz düşer ve daha fazla devam edemeyeceğini anlayarak bir ağaca çarpar ve olduğu yerde bayılır. Karakterimiz bayıldığında karakterin ağızından hikaye anlatımı başlar. “ Her şey geceleri ortaya çıkan yaratıklarla başları. Git gide yaratıklara karşı verdığımız kayıplar artmaya başladı. Ve biz de geceleri evlerimize kapanmak zorunda kaldık, gündüz ise kısıtlı bir hayatın içine hapsolduk. Onlarla nasıl savaşacağımızı bilemez hale geldik. Korku insanları ele geçirdi. Bu yaratıklar zihnimize girip bizimle oyun oynadılar. Kimimiz delirdi, kimimiz de ortadan kayboldu. Kardeşim de delirenlerden biriydi....” Karakterimiz tekrar kendine gelip uyandığında gündüz olmuştur. Olduğu yerden kalkar ve evine gitmek için yola düşer. Bu sırada oyun öğretici kısma başlar. Yürüme, eşya toplama, envanter kullanımı gibi bilgiler verilir. Gece yaratıklardan etkilendiği için uyandığında belirli sesler duyarak evine ulaşmaya çalışır. Evine ulaştığında ise kardeşini evde bulamaz. Kardeşinin zihninin artık tamamen yaratıkların kontrolüne geçtiğini ve onların peşinde ortadan kaybolduğunu anlar. Kardeşini bulup iyileştirme umuduyla yola çıkar ve bir arayışa girer. Bu sırada yolda birkaç yeni yan görevler, yeni tanımlarla karşılaşır. Oyunun devamında karakterimiz Tanrı ile iletişime geçer ve onun yanına gider. Tanrıdan yardım ister. Yaratıkların güçleri yüzünden tanrı tüm gücünü kullanamadığı için belirli görevler vererek bazı şartlarla ana karakterimize yardım etmeye başlar. Kardeşi için yapması gereken ilacın gerektirdiği şeylerin bulunmasında, savaşçı veya büyütücü olarak kazandığı yeteneklerin verilmesinde tanrıının verdiği görevlerin başarılı şekilde geçilmesi rol oynar. Tanrı kendi yardımlarının, kendi verdiği görevlerin yanı sıra ana karakterimize yardımcı karakter de verir. Bu yolculuk sürecinde sesli betimlemeler gibi yererde yardımcı karakter ana karaktere tanımlama yaparak veya bazen ipucu vererek yardımcı olur. Yaşanan birçok çalışma, yan görevler gibi aşamalardan sonra karakterimiz belirli bir bilgiye ve yeteneğe sahip olduğunda Zihin hırsızı yaratığın inine ulaşır. Orda kardeşiyle birlikte deliren birçok insan daha olduğunu görür. Yaratıkla savaşır ve yaratığı alt etmesi sonucunda yaratıktan aldığı bir totemle ilacın son aşamasını tamamlar ve kardeşine vererek onu normal haline döndürmeye başlar. Böylece gece dünyasını zihin hırsızı yaratığın elinden kurtarmış olur. Şimdi ise sırada

Tanrılarının çalınan güçlerinin yeniden gelmesi için ana karakterimizle tanrıının yaptığı anlaşmalar vardır...

## **2. LİTERATÜR TARAMASI**

Seçilen konuya ilgili olarak yeterli sayıda örnek bulunamadığından görme engelli bireylerle iletişim kurulabilmesi açısından İstanbul Diyalog Müzesi’nde yer alan “Karanlıkta Diyalog” oluşumuyla ilgilenen oyuncu kitleyle iletişime geçilmiştir ve bu şekilde görme engelli oyuncuların oynadıkları oyunlara erişim mümkün hale gelmiştir. Bu bağlamda incelenen oyunlar; Eurofly ve MistWorld olarak seçilmiştir.

### **2.1 EUROFLY**

Eurofly, görme engelli bireyler için geliştirilmiş bir uçuş simülasyon oyunudur. Oyuncular, sanal bir pilot olarak çeşitli görevleri tamamlayarak ve dünya çapında uçuşlar yaparak deneyim kazanırlar. Bu oyun, görme engelli bireylerin dijital dünyada uçuş simülasyonu deneyimini yaşayabilmeleri için özel olarak tasarlanmıştır. Görme engelli bireyler için geliştirilen en bilindik oyunlardan biri olduğu bilinmektedir. Bir simülasyon oyunu olması sebebiyle kısıtlı bir bölgeye hakim olabilmenin getirdiği deneyimle WitDark’ın açık dünya olarak tasarlanması hedeflenmiştir. Eurofly, klavye kullanımı dışında joystick ile de oynanabilmektedir. Görme engelli bireylerin mouse olmadan klavye kullanıyor olmaları göz önünde bulundurulduğunda, oyunumuzun ilk çıkış platformunun bilgisayar olmasına karar verilmesi bağlamında klavye kullanımında Eurofly’dan ilham alınmıştır.

### **2.2 MIST WORLD**

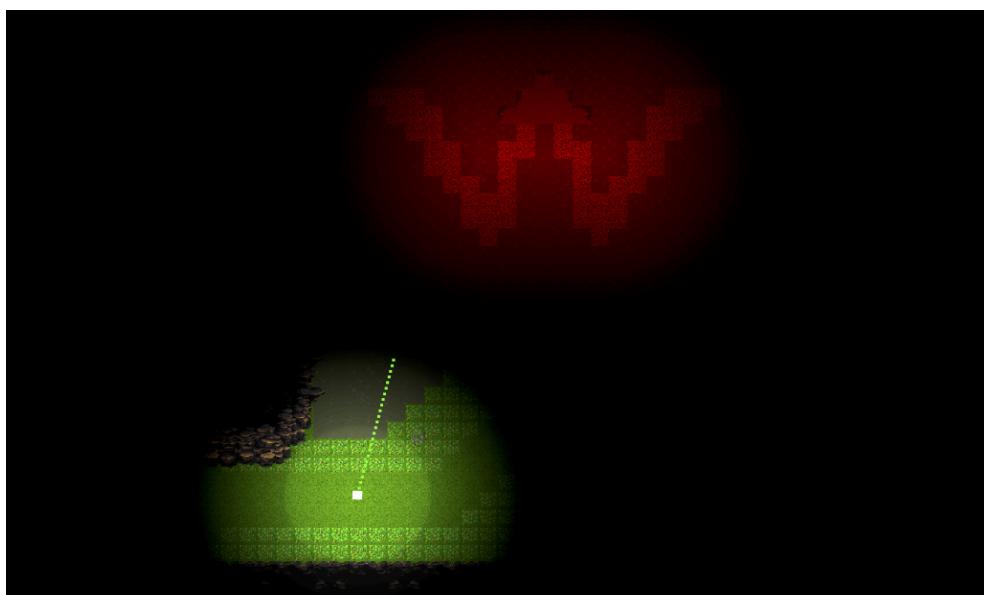
Mist World, bir MMORPG (Devasa Çok Oyunculu Çevrimiçi Rol Yapma Oyunu) olarak tanımlanan büyük çaplı çok oyunculu bir çevrimiçi rol yapma oyunudur. Oyunda, oyuncular fantastik bir dünyada çeşitli karakterleri kontrol ederler ve birbirleriyle etkileşime girerler. Oyunda oyuncular birer elf, rahip, avcı, büyüğü, ork veya avcısı gibi sınıfı sahip olabilmenin yanı sıra oyuncular istedikleri takdirde tehlikeli insan dünyasını korumak için oyundaki canavarlarla veya iblislerle savaşabilmektedirler. Oyuncular ayrıca bu canavarlar ile savaşmanın yanı sıra oyundaki arenayı dövüş becerilerini göstermek veya takımlar arasında rekabet etmek için kullanabilirler. Dövüşmenin yanı sıra, oyunda savaşmanın dışında ilginç

oynanış şekilleri de vardır: madencilik, ot toplama, balık tutma, düğün yapma, öğretmenlerden eğitim alma ve çırak alma vb. böylece oyuncular deneyimlerini daha da zenginleştirebilirler. Bu oyunun bizim projemize nazaran öne çıkan en büyük özelliği MMORPG olmasıdır. Bizim projemiz, bir rol yapma oyunu olmasının yanında single player(tek-oyunculu) bir oynanış sunmaktadır. Mist World ve WitDark oyunları, karakter seçimi gibi daha gelişmiş özelliklerin dışında neredeyse aynı potada eritilebilecek oyunlardır.

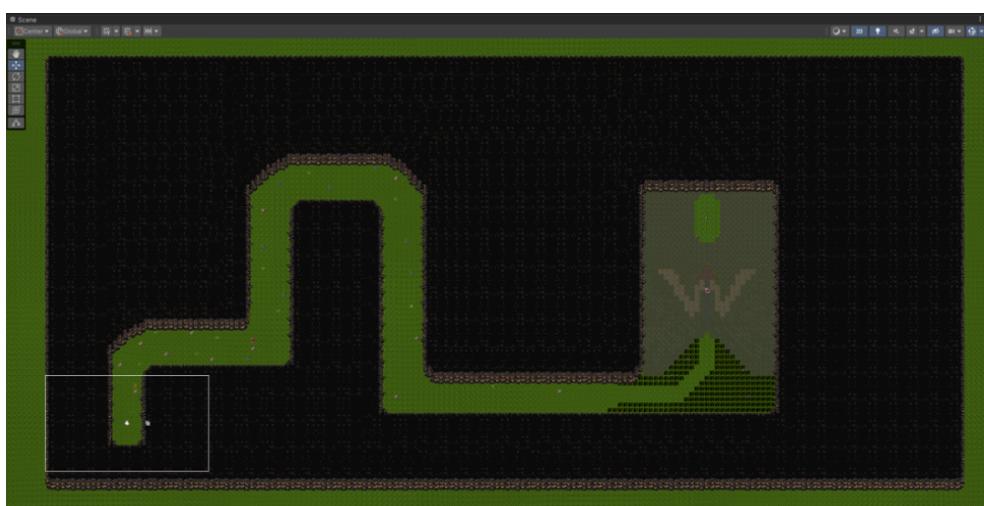
### 3. YÖNTEM

#### 3.1 GİRİŞ SAHNESİ TASARIMI (Beyzanur YAKUT, Ömer Faruk DİNÇASLAN)

Projemize Giriş Sahnesini tasarlayarak başladık, bu sahneyi hazırlarken de diğer adımlarda bahsedeceğimiz çeşitli ihtiyaçlar doğdu, bunlar özellikle karakterin tasarımını, oynanış mekaniği, oyuncunun çevresindeki nesneler ile iletişimini gibi birkaç temel unsurdan oluşmuştur. Giriş sahnesi, oyuncunun X ve Y koordinatları sırasıyla 0'a 0 başlangıç konumundan (ekranda sol altta göründüğü üzere) WitDark Logosunun olduğu bölgeye doğru oyuncunun hikayeyle bağlantılı olan bir nesneyi aramak üzere yolculuğunu konu alır.



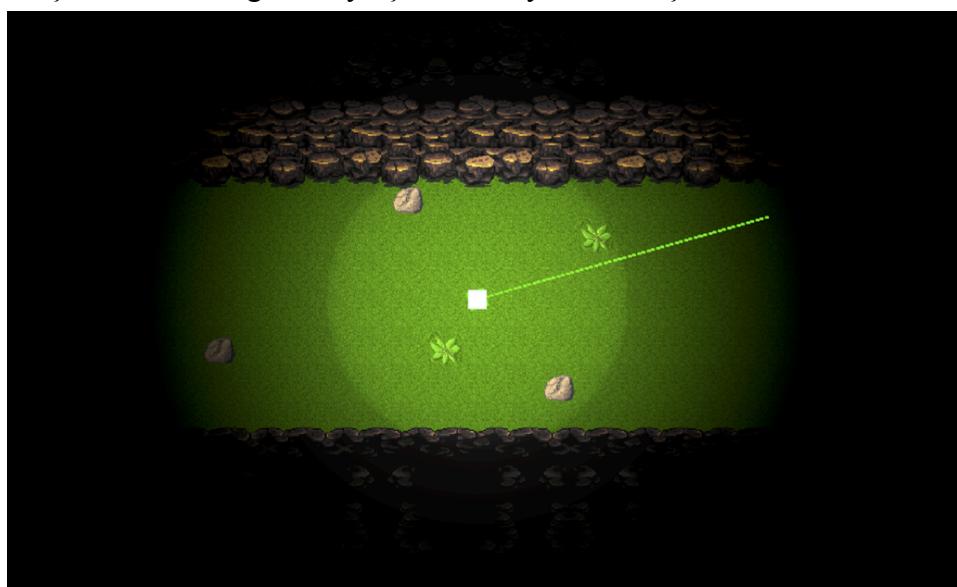
*Şekil - 1: Giriş Sahnesindeki Oynanışa Dair Görünüm*



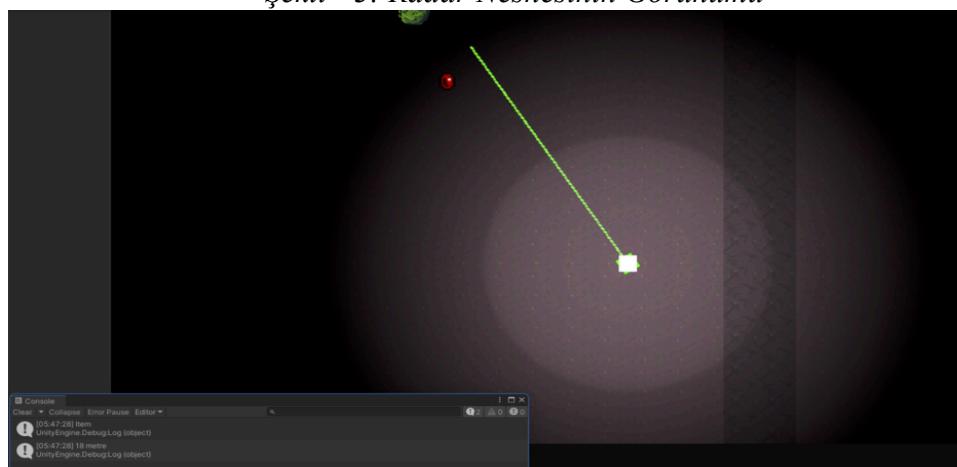
*Şekil - 2: Giriş Sahnesi Tasarımı*

### **3.2 TEMEL OYNANIŞ MEKANİKLERİ VE RADAR NESNESİNİN OLUŞTURULMASI (Beyzanur YAKUT, Ömer Faruk DİNÇASLAN)**

Bir sonraki adımda projemizdeki ana karaktere yürümesini, koşmasını sağlayacak bir hareket komutu sağlayıp bunun ardından ona “Radar” adını verdığımız bir nesne ekledik. Bu “Radar” nesnesinin temel işlevi karakterimizin 20 metre çapındaki nesneler ile etkileşime girip oyuncuya ilgili objelere ait anlık genel bilgi ve özellikle konum bilgisi sağlamaktır. Radarı tasarlarken bir objeden ziyade ışık kullandık(Unity’de Light nesnesi olarak geçiyor) bu sayede genel oynanıştaki görsellerin karanlık görünümünden etkilenmeyecek olup nesnelerin içerisinde gecebilecek ve hedeflenen bilgiyi oyuncuya sağlayacaktır. Yukarıda bahsettiğimiz radarın etkileşimi ile alakalı görüntüye Şekil -4’te yer verilmiştir.



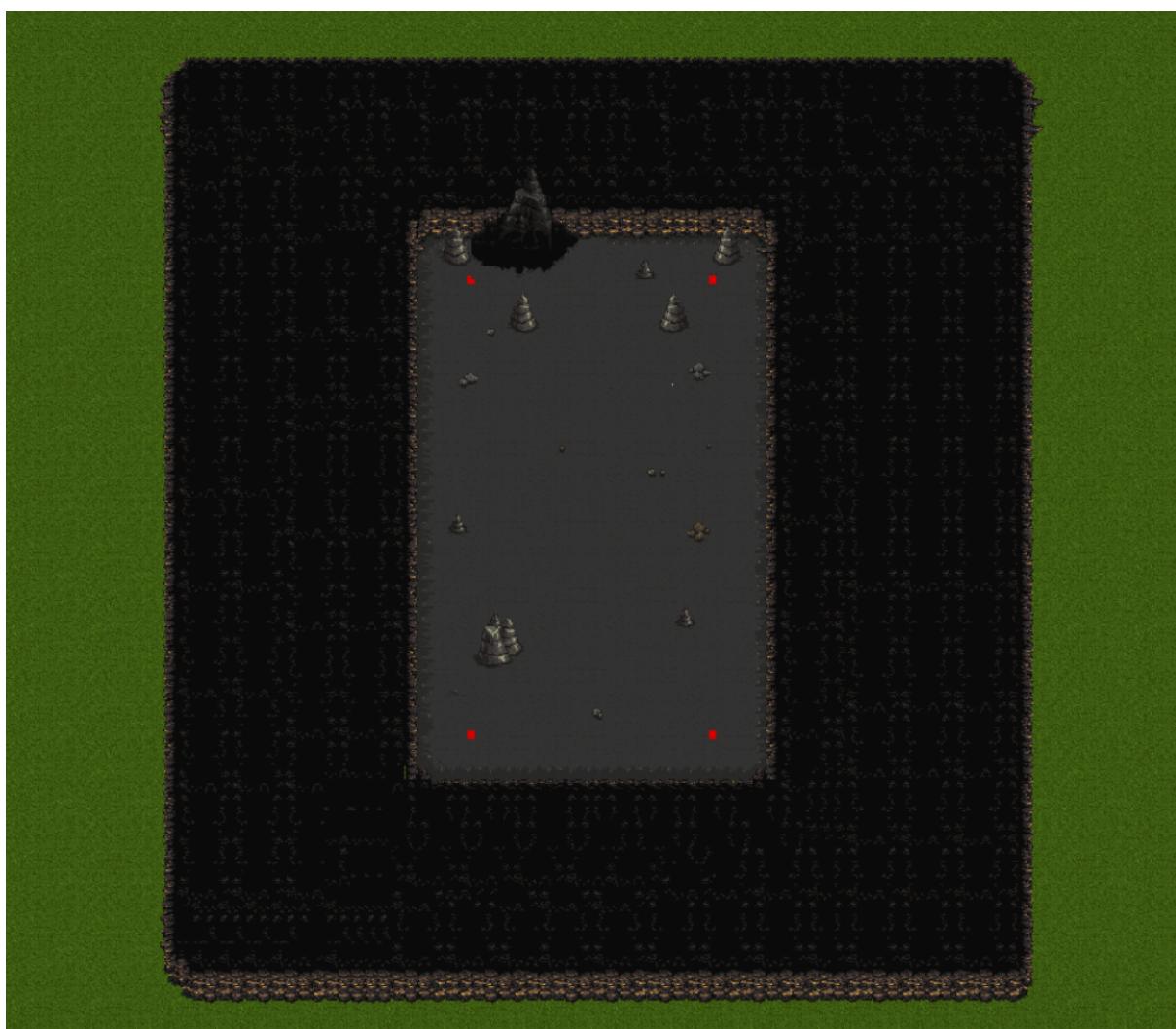
*Şekil - 3: Radar Nesnesinin Görünümü*



*Şekil - 4: Radar Nesnesinin Diğer Nesnelerle Etkileşimine Ait Bir Görüntü*

### 3.3 TUTORİAL SAHNESİ TASARIMI (Beyzanur YAKUT, Ömer Faruk DİNÇASLAN)

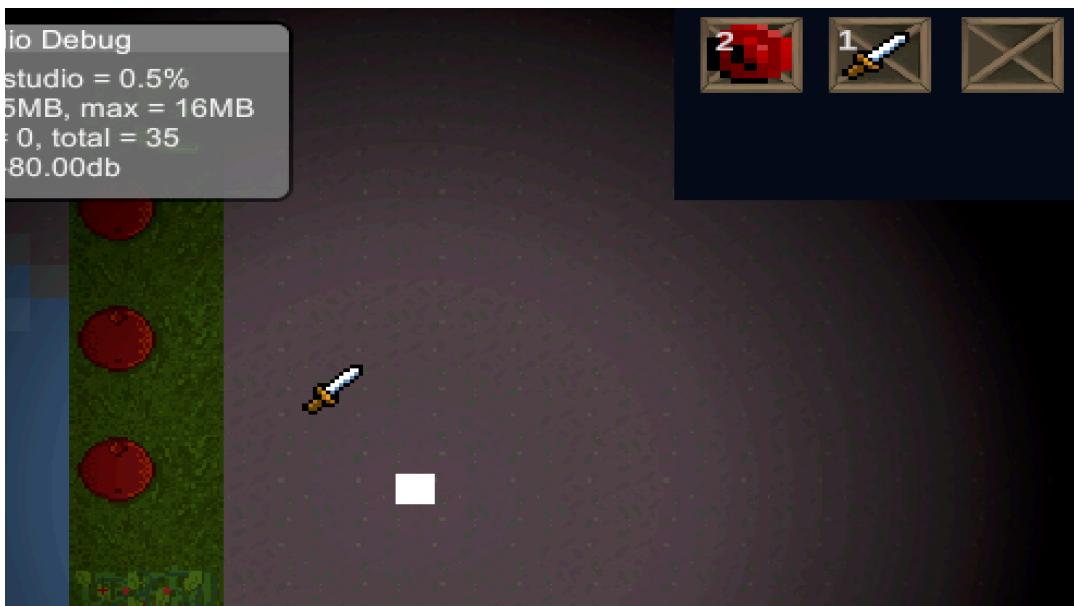
İlk sahneyi ve temel mekanikleri tasarladıkten hemen sonra bu mekanikleri oyuncuya öğretebileceğimiz bir tutorial(öğretici kısım) alanı oluşturmak istedik. Bu aynı zamanda oyuncunun oyunun hikayesini öğreneceği kısımdan hemen sonra Player(oyuncu) nesnesi Öğretici Modu alanına ışınlanmasıyla erişebileceği bir sahnedir. Bu alanda bir NPC(oyuncu olmayan - yan karakter) ile etkileşime girecek ve oyuna devam etmek için gerekli bilgileri NPC'den edinecektir. Bu sahnenin tasarımı yapılrken alışlagelmiş RPG oyunlarından ilham alarak bir yeraltı zindani, bir maden ambiyansı yakalanması amaçlanmıştır. Bu doğrultuda karakterimiz mağaradan çıkacak ve yolculuğuna başlayacaktır.



*Şekil - 5 : Öğretici Modun Alanının Tasarımı*

### **3.4 ENVANTERİN OLUŞTURULMASI (Beyzanur YAKUT)**

Önceki adımlarda oyuncuya hikaye ve oynanış konusunda destek olduktan hemen sonra bir diğer temel oynanış mekaniklerinden biri olan envanter sistemini geliştirmeyi kendimize görev belirledik. Çünkü envanter, oyuncunun nesneler, yan karakterler ve diğer öğeler ile olan etkileşimi zenginleştirmede önemli bir rol oynar. Projemizde geliştirdiğimiz oyunun envanter menüsü, oyuncuların sahip oldukları eşyaları yönetmelerine olanak tanıyan kullanıcı dostu bir arayüz sunar. Envanter menüsü, oyunun oynanışını ve kullanıcı deneyimini geliştiren kritik bir bileşendir. Envanter menüsü, görsel olarak çekici ve işlevsel bir arayüze sahiptir. Menü, oyun ekranının sağ üst köşesinde bir canvas ile temsil edilir. Envanterin oyun içinde kullanımını genel olarak ifade etmek gerekirse, oyuncu, klavyeden “I” tuşuna bastığında envanter açılır ve yine “I” tuşuna basıldığı takdirde envanter kapatılır. Envanter açıldığında ekranın belirli bir kısmını kaplar. “A” ve “D” tuşları envanter menüsü içindeki nesneler arasında dolaşmayı sağlar. Hangi envanter seçilmiş ise bu envanterin ismi konsola yazdırılarak oyuncunun envanterleri istediği gibi yönetmesi sağlanır. Envanter seçildikten sonra “Enter” tuşuna tıklanarak bu envanterin sayısı azaltılır ve var olan envanter sayısı konsola yazdırılarak oyuncu bilgilendirilir. Geliştirdiğimiz envanter menüsü, kullanıcı dostu tasarımlı ve işlevsel özellikleri ile oyuncuların oyundan maksimum keyif almalarını sağlamaktadır. Oyuncuların eşyalarını kolayca yönetebilmeleri, oyunun akıcılığını ve oynamabilirliğini artıran önemli bir unsurdur.



*Sekil - 6: Envanter Menüsünün Görünümü*

```

1  using UnityEngine;
2  public class PlayerCtrl : MonoBehaviour
3  {
4      public GameObject inventory;
5      bool invIsActive = false;
6      [SerializeField] float speed;
7      public static int swordAmount = 0;
8      public GameObject player;
9      public GameObject inventoryMenu;
10     public GameObject[] inventorySlots;
11     public int selectedSlotIndex = 0;
12
13     void Start()
14     {
15         inventory.SetActive(false);
16     }
17     void Update()
18     {
19         if (!invIsActive)
20         {
21             if (Input.GetKeyDown(KeyCode.I))
22             {
23                 inventory.SetActive(true);
24                 invIsActive = true;
25             }
26         }
27         else
28         {
29             PlayerMovement.movSpeed = 0;
30             PlayerMovement.speedX = 0;
31             PlayerMovement.speedY = 0;
32             PlayerMovement.rb.velocity = Vector2.zero;
33
34             if (Input.GetKeyDown(KeyCode.A))
35             {
36                 selectedSlotIndex = (selectedSlotIndex - 1 + inventorySlots.Length) % inventorySlots.Length;
37                 Debug.Log("Seçilen envanter: " + inventorySlots[selectedSlotIndex].name);
38             }
39             else if (Input.GetKeyDown(KeyCode.D))
40             {
41                 selectedSlotIndex = (selectedSlotIndex + 1) % inventorySlots.Length;
42                 Debug.Log("Seçilen envanter: " + inventorySlots[selectedSlotIndex].name);
43             }
44
45             if (Input.GetKeyDown(KeyCode.I))
46             {
47                 inventory.SetActive(false);
48                 invIsActive = false;
49             }
50         }
51     }
52 }
```

Ln 1,

*Şekil - 7: Oyuncunun Envanter Menüyü Kullanabilmesini Sağlayan Kod*

```

1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using FMODUnity;
5   [RequireComponent(typeof(StudioEventEmitter))]
6
7   public class coin : MonoBehaviour
8   {
9       private bool _collected = false;
10      private Transform playerTransform;
11      private StudioEventEmitter emitter;
12
13      void Start()
14      {
15          playerTransform = GameObject.FindGameObjectWithTag("Player").transform;
16          emitter = AudioManager.instance.InitializeEventEmitter(FMDEvents.instance.coinIdle, this.gameObject);
17          emitter.Play();
18      }
19      private void OnTriggerEnter2D(Collider2D collision)
20      {
21          if (collision.gameObject.CompareTag("Player"))
22          {
23              if (!_collected)
24              {
25                  ScoreText.coinAmount += 1;
26                  _collected = true; // Not needed if destroying the game object
27                  Destroy(this.gameObject); // Only use if you intend to destroy the game object
28                  emitter.Stop();
29                  AudioManager.instance.PlayOneShot(FMDEvents.instance.coinCollected, this.playerTransform.position);
30              }
31          }
32          if (collision.gameObject.CompareTag("PlayerLight"))
33          {
34              Debug.Log("Item");
35              float distance = Vector3.Distance(transform.position, playerTransform.position);
36
37              if (distance > 0.8f && distance <= 1.2)
38              {
39                  Debug.Log("2 metre");
40              }
41              else if (distance > 0.4f && distance <= 0.8)
42              {
43                  Debug.Log("1 metre");
44              }
45              else
46              {
47                  Debug.Log("Distance to Player: " + distance + " units");
48              }
49          }
50      }
51  }

```

Lp 14\_Coin

*Şekil - 8: Coin(Para)'lerin Toplanarak Envantere Eklenmesini Sağlayan Kod (Bu kod başka bir eşyaya da uyarlanabilir)*

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using TMPro;
4  using UnityEngine;
5  using UnityEngine.UI;
6
7  public class ScoreText : MonoBehaviour
8  {
9      public TextMeshProUGUI text;
10     public static int coinAmount = 0;
11     public int i = 0;
12     GameObject player;
13
14
15     void Start()
16     {
17         player = GameObject.FindGameObjectWithTag("Player");
18         text = GetComponent<TextMeshProUGUI>();
19     }
20
21
22     void Update()
23     {
24         i = player.GetComponent<PlayerCtrl>().selectedSlotIndex;
25         if (text != null)
26         {
27             text.text = coinAmount.ToString();
28
29             if (Input.GetKeyDown(KeyCode.Return))
30             {
31                 if (i == 0)
32                 {
33                     if (coinAmount > 0)
34                     {
35                         coinAmount--;
36                         Debug.Log("\nPara envanteri azaltıldı. Yeni envanter sayısı: " + coinAmount);
37
38                     }
39                 }
40             }
41         }
42     }
43 }

```

*Sekil - 9: Envanter Menüsünde Coin(Para)lerin Sayısını Değiştirmeyi Sağlayan Kod (Bu kod başka bir eşyaya da uyarlanabilir)*

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class sword : MonoBehaviour
6  {
7      private bool _collected = false;
8      private Transform playerTransform;
9
10     void Start()
11     {
12         playerTransform = GameObject.FindGameObjectWithTag("Player").transform;
13     }
14     private void OnTriggerEnter2D(Collider2D collision)
15     {
16         if (collision.gameObject.CompareTag("Player"))
17         {
18             if (!_collected)
19             {
20
21                 swordText.swordAmount += 1;
22                 _collected = true;
23                 Destroy(this.gameObject); ⚡
24             }
25         }
26         if (collision.gameObject.CompareTag("PlayerLight"))
27         {
28
29             float distance = Vector3.Distance(transform.position, playerTransform.position);
30
31             if (distance > 0.8f && distance <= 1.2)
32             {
33                 Debug.Log("Kılıc " + "2 metre");
34             }
35             else if (distance > 0.4f && distance <= 0.8)
36             {
37                 Debug.Log("Kılıc, " + "1 metre");
38             }
39             else
40             {
41                 Debug.Log("Distance to Player: " + distance + " units");
42             }
43         }
44     }
45 }

```

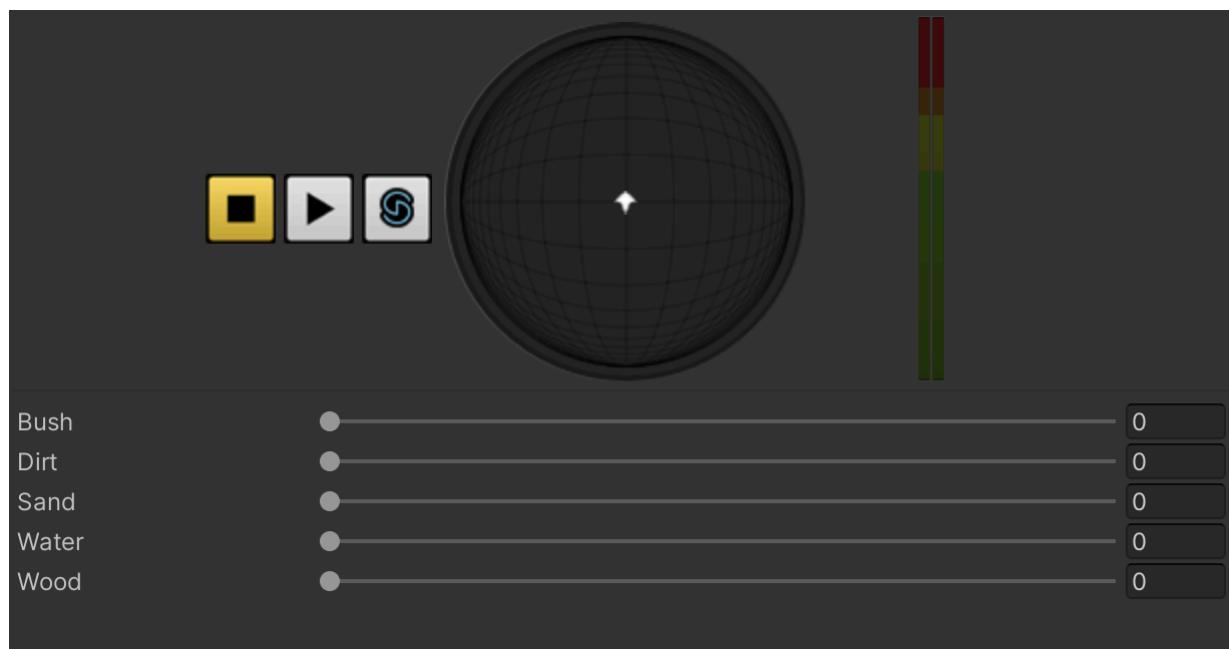
*Şekil - 10: Envanter Menüsünde Sword(Kılıç)’ların Sayısını Değiştirmeyi Sağlayan Kod (Bu kod başka bir eşyaya da uyarlanabilir)*

### 3.5 SES TASARIMI (Mustafa ÖRNEK)

Projenin ses tasarımı kısmına geldiğimizde ise bu kısım her ne kadar projenin geneline dağılmış ve diğer yapılan adımlardan bağımsız da olsa bu tasarımın bu aşamada gerçekleştirmeyi uygun gördük. Projenin ana hedef kitlesinin görme engelli bireyler olduğu düşünüldüğünde, en önemli unsurun ses olduğu bir geçektir. Detaylı araştırmalarımız sonucu elde edilen verilere bakılarak FMOD Studio ile ses tasarımı gerçekleştirildi. 3D olarak tasarlanan sesler oyunda gerekli kodlamalar ile belirli bölgelere konumlandırıldı.

#### 3.5.1 AYAK SESLERİ

Karakterin bulunduğu bölgedeki ortamla bağlantı kurabilmesi açısından birçok farklı ayak sesi FMOD sayesinde 3D Timeline olarak tasarlandı.



Şekil - 11: Ayak Seslerinin FMOD for Unity ile Gösterimi

```

1  using UnityEngine;
2  using System.Collections;
3
4
5  public class Footsteps : MonoBehaviour
6  {
7
8      public FMODUnity.EventReference m_EventPath;
9
10
11     public float m_Wood;
12     public float m_Water;
13     public float m_Dirt;
14     public float m_Sand;
15
16
17     public float m_StepDistance = 2.0f;
18     float m_StepRand;
19     Vector3 m_PrevPos;
20     float m_DistanceTravelled;
21
22
23     public bool m_Debug;
24     Vector3 m_LinePos;
25     Vector3 m_TrianglePoint0;
26     Vector3 m_TrianglePoint1;
27     Vector3 m_TrianglePoint2;
28
29
30     void Start()
31     {
32         //Initialise random, set seed
33         Random.InitState(System.DateTime.Now.Second);
34
35         //Initialise member variables
36         m_StepRand = Random.Range(0.0f, 0.5f);
37         m_PrevPos = transform.position;
            m_LinePos = transform.position;

```

*Sekil - 12: Ayak Seslerinin Bulunulan Bölgeye Göre Değişmesini Sağlayan Kod*

```

38     }
39
40     void Update()
41     {
42         m_DistanceTravelled += (transform.position - m_PrevPos).magnitude;
43         if(m_DistanceTravelled >= m_StepDistance + m_StepRand)//TODO: Play footstep sound based on position from here
44         {
45             PlayFootstepSound();
46             m_StepRand = Random.Range(0.0f, 0.5f); //Adding subtle random variation to the distance required before
47             m_DistanceTravelled = 0.0f;
48         }
49
50         m_PrevPos = transform.position;
51
52         if(m_Debug)
53         {
54             Debug.DrawLine(m_LinePos, m_LinePos + Vector3.down * 1000.0f);
55             Debug.DrawLine(m_TrianglePoint0, m_TrianglePoint1);
56             Debug.DrawLine(m_TrianglePoint1, m_TrianglePoint2);
57             Debug.DrawLine(m_TrianglePoint2, m_TrianglePoint0);
58         }
59     }
60
61
62
63     void PlayFootstepSound()
64     {
65         //Defaults
66         m_Water = 0.0f;
67         m_Dirt = 1.0f;
68         m_Sand = 0.0f;
69         m_Wood = 0.0f;
70
71         RaycastHit hit;
72         if(Physics.Raycast(transform.position, Vector3.down, out hit, 1000.0f))
73         {
74
75             if(m_Debug)
76                 m_LinePos = transform.position;
77
78             if(hit.collider.gameObject.layer == LayerMask.NameToLayer("MainGround"))
79             {
80                 int materialIndex = GetMaterialIndex(hit);
81                 if(materialIndex != -1)
82                 {
83                     Material material = hit.collider.gameObject.GetComponent<Renderer>().materials[materialIndex];

```

*Sekil - 13: Ayak Sesleri Kod ( Devami - I)*

```

if(m_Debug)
    Debug.Log("Wood: " + m_Wood + " Dirt: " + m_Dirt + " Sand: " + m_Sand + " Water: " + m_Water);

//if(m_EventPath != null)
{
    FMOD.Studio.EventInstance e = FMODUnity.RuntimeManager.CreateInstance(m_EventPath);
    e.set3DAttributes(FMODUnity.RuntimeUtils.To3DAttributes(transform.position));

    e.setParameterByName("Wood", m_Wood);
    e.setParameterByName("Dirt", m_Dirt);
    e.setParameterByName("Sand", m_Sand);
    e.setParameterByName("Water", m_Water);

    e.start();
    e.release(); //Release each event instance immediately, there are fire and forget, one-shot instances.
}

```

1 reference

```

GetMaterialIndex(RaycastHit hit)

Mesh m = hit.collider.gameObject.GetComponent<MeshFilter>().mesh;
int[] triangle = new int[]
{
    m.triangles[hit.triangleIndex * 3 + 0],
    m.triangles[hit.triangleIndex * 3 + 1],
    m.triangles[hit.triangleIndex * 3 + 2]
};
for(int i = 0; i < m.subMeshCount; ++i)
{
    int[] triangles = m.GetTriangles(i);
    for(int j = 0; j < triangles.Length; j += 3)
    {
        if(triangles[j + 0] == triangle[0] &&
           triangles[j + 1] == triangle[1] &&
           triangles[j + 2] == triangle[2])
            return i;
    }
}
return -1;

```

*Sekil - 14: Ayak Sesleri Kod ( Devami - 2)*

### 3.5.2 AUDIO MANAGER

Oyun içi ses yönetimini gerçekleştiren sınıfıtır. Ses olaylarının(event) oluşturulması, çalınması ve durdurulması gibi işlemleri yönetir. Ses yayımlayıcısının çalışmasını kontrol eder. Oyundaki ayarların ayarlanması sağlar.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using FMODUnity;
5  using FMOD.Studio;
6  using System;
7
8  public class AudioManager : MonoBehaviour
9  {
10    private List<EventInstance> eventInstances;
11    private List<StudioEventEmitter> eventEmitters;
12    public static AudioManager instance { get; private set; }
13
14    private void Awake()
15    {
16      if (instance != null)
17      {
18        Debug.LogError("Found more than one Audio Manager in the scene");
19      }
20      instance = this;
21
22      eventInstances = new List<EventInstance>();
23      eventEmitters = new List<StudioEventEmitter>();
24    }
25
26    public void PlayOneShot(EventReference sound, Vector3 worldPos)
27    {
28      RuntimeManager.PlayOneShot(sound, worldPos);
29    }
30
31    public EventInstance CreateInstance(EventReference eventReference)
32    {
33      EventInstance eventInstance = RuntimeManager.CreateInstance(eventReference);
34      eventInstances.Add(eventInstance);
35      return eventInstance;
36    }
37
38    public StudioEventEmitter InitializeEventEmitter(EventReference eventReference, GameObject emitterGameObject)
39    {
40      StudioEventEmitter emitter = emitterGameObject.GetComponent<StudioEventEmitter>();
41      emitter.EventReference = eventReference;
42      eventEmitters.Add(emitter);
43    }
}
```

*Şekil - 15: Seslerin Yönetilmesini Sağlayan Kod*

```

43     // Hoparlör simgesini devre dışı bırak
44     Renderer renderer = emitterGameObject.GetComponent<Renderer>();
45     if (renderer != null)
46     {
47         renderer.enabled = false;
48     }
49
50     return emitter;
51 }
52
53
54     1 reference
55     private void CleanUp()
56     {
57         // Stop and release any created instances
58         foreach (EventInstance eventInstance in eventInstances)
59         {
60             eventInstance.stop(FMOD.Studio.STOP_MODE.IMMEDIATE);
61             eventInstance.release();
62         }
63
64         // Stop all of the event emitters because if we don't they may hang around in other scenes
65         foreach (StudioEventEmitter emitter in eventEmitters)
66         {
67             emitter.Stop();
68         }
69
70     0 references
71     private void OnDestroy()
72     {
73         CleanUp();
74     }
75
76     0 references
77     internal StudioEventEmitter InitializeEventEmitter(object swordIdle, GameObject gameObject)
78     {
79         throw new NotImplementedException();
80     }
81
82     // Ses olayını durdurmak için metot
83     0 references
84     public void StopEvent(GameObject gameObject)
85     {
86
87         // Eğer bu nesne bir StudioEventEmitter bileşeni içeriyorsa, onun oynatılan sesini durdur
88         StudioEventEmitter emitter = gameObject.GetComponent<StudioEventEmitter>();
89         if (emitter != null)
90         {
91             emitter.Stop();
92         }

```

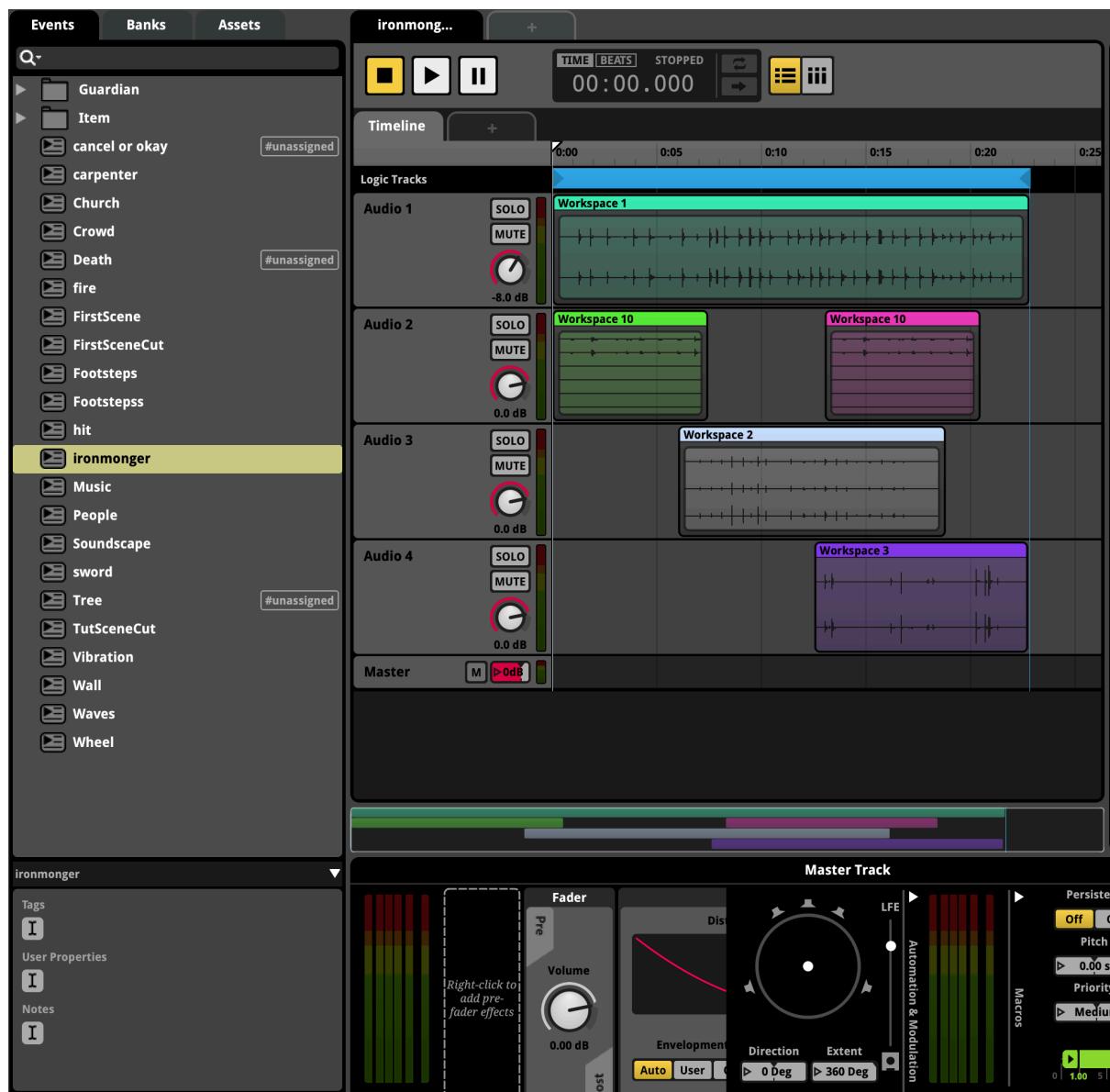
*Şekil - 16: Seslerin Yönetilmesini Sağlayan Kod(Devamı)*

### 3.5.3 FMOD EVENTS

Bu sınıf, belirli ses olaylarına referansları tutar ve bu referansları diğer sınıfların erişimine sunar. Aynı zamanda oyun içindeki materyallerin efektlerinin temel ses seviyelerini ayarlamaya yardımcı olur. FMOD Studio kısmında oluşturulan ses tasarımlarının hassaslık ayarları gerçekleştirildikten sonra entegreleri sağlanmıştır. Coin gibi oyun içi diğer materyallerin entegresi diğer kısımlarda eklenmiştir.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using FMODUnity;
5
6  public class FMODEvents : MonoBehaviour
7  {
8
9      [field: Header("Coin SFX")]
10     [field: SerializeField] public EventReference coinCollected {get; private set; }
11
12     [field: Header("Player SFX")]
13
14     [field: SerializeField] public EventReference playerFootsteps{ get; private set;}
15
16     [field: SerializeField] public EventReference coinIdle { get; private set; }
17     public static FMODEvents instance { get; private set; }
18
19     private void Awake()
20     {
21         if (instance != null)
22         {
23             |   Debug.LogError("Found more than one FMOD Events instance in the scene.");
24         }
25         instance = this;
26     }
27 }
```

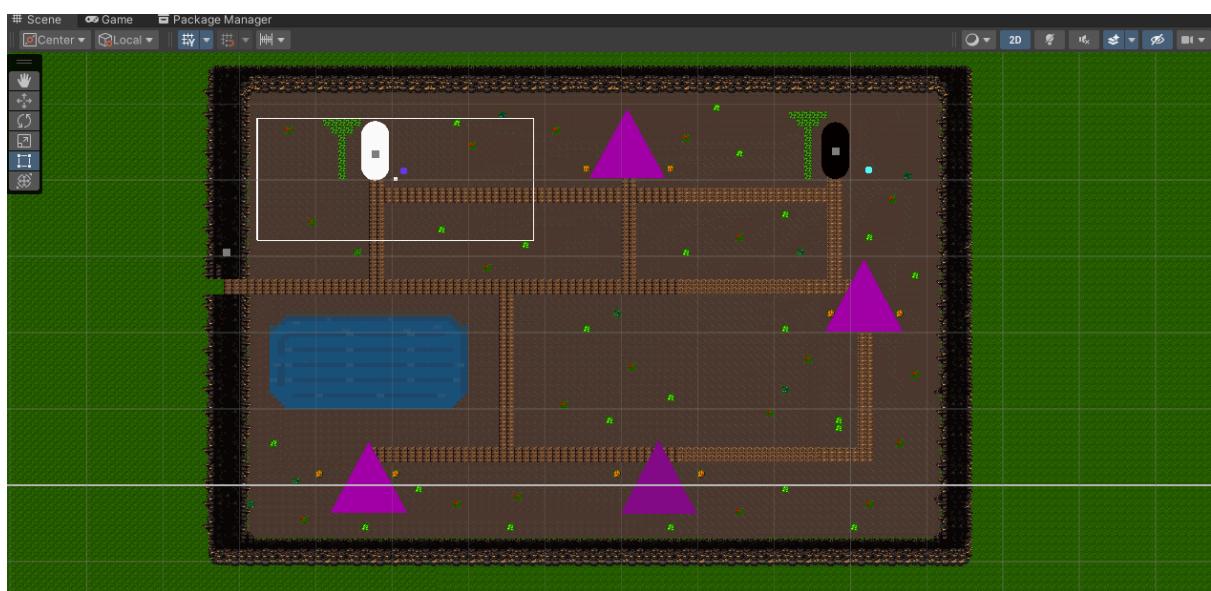
*Sekil - 17: Genel Ses Ayarlarını Yapan Kod*



*Şekil - 18: FMOD Tasarım Arayüzü*

### 3.6 KÖY ALANI TASARIMI (Gizem GÜVEN)

Ses tasarımlına devam ederken ilk sahneler ve öğretici sahnelerin ardından karakterimizin(oyuncunun) gerekiğinde başvuracağı, tabiri caizse onun evi olan köy alanının tasarımını yapmaya karar verdik. Projenin bu kısmında köy alanının arayüz tasarımı gerçekleştirildi. Oyunun açık dünya imkanı kullanılarak köy alanında oyuncunun ortamı keşif teması, farklı alanlarda göstereceği ses özellikleri göz önüne alınarak birbirinden farklı ortamlar oluşturuldu. Köy alanı diğer alanlardan etrafi çevrilerek ayrıldı bu sayede köye giren oyuncunun güvenli alana girişi gösterildi. Oyuncunun etkileşime girip diyalog kurabileceği NPC'ler oluşturuldu.



Şekil - 19: Köy Alanının Genel Görünümü

### **3.7 NPC'LERİN OLUŞTURULMASI (Gizem GÜVEN)**

Önceki adımda oluşturduğumuz köy alanına canlılık katmak amacıyla köy tasarımine birkaç NPC(oyuncu olmayan karakter) eklemeyi uygun gördük. Bu köy alanında, iki önemli NPC (Non-Player Character) oluşturulmuştur: biri demirci, diğer ise marangoz olarak adlandırılmıştır. Oyuncu, belirli bir seviyeye göre ayarlanmış bu NPC'lerin bulunduğu alana geldiğinde, "E" tuşuna basarak bu karakterlerle etkileşime girebilir ve konuşma başlatılmaktadır. Konuşmanın devam etmesi "Space" tuşuna basarak yönlendirilmektedir. Bu etkileşim sürecinde, her NPC'nin kendine özgü özellikleri ve kişilikleri doğrultusunda oyuncuya kurdukları diyaloglar, karakterlerin üzerindeki beliren konuşma baloncukları aracılığıyla görüntülenmektedir. Görme engelli kullanıcılar için ise bu konuşma baloncuklarındaki her cümle sesli betimleme yoluyla aktarılmaktadır. Böylece, tüm oyuncuların oyun içi diyalogları tam anlamıyla deneyimlemeleri sağlanmaktadır. NPC'lerle kurulan bu diyaloglar, oyunun hikayesine önemli katkılar sunar. Oyuncular, bu karakterlerle etkileşime girerek oyunun ana hikayesi hakkında daha fazla bilgi edinirler ve oyunun atmosferine daha derinlemesine dalarlar. Ayrıca, NPC'ler oyunculara belirli görevler vererek, ana oyundan yan oyumlara ve ek hikaye parçalarına geçiş imkanı sağlamaktadır. Bu görevler, oyunun genel hikaye yapısını ve oynanışını zenginleştirir, oyunculara daha geniş bir deneyim sunar. Bu şekilde, oyuncunun oyundaki farklı karakterlerle etkileşim kurması ve onlarla iletişimde bulunması sağlanmış olur. Oyunun hikayesi, NPC'ler aracılığıyla genişletilerek, oyunculara daha derin ve anlamlı bir oyun deneyimi sunulmaktadır. Görevler ve yan hikayelerle desteklenen bu yapı, oyuncuların oyunda geçirdikleri zamanı daha keyifli ve verimli hale getirmektedir.



Şekil-20: Köy Alanında Oyuncunun Demirci Karakteri İle Etkileşimi



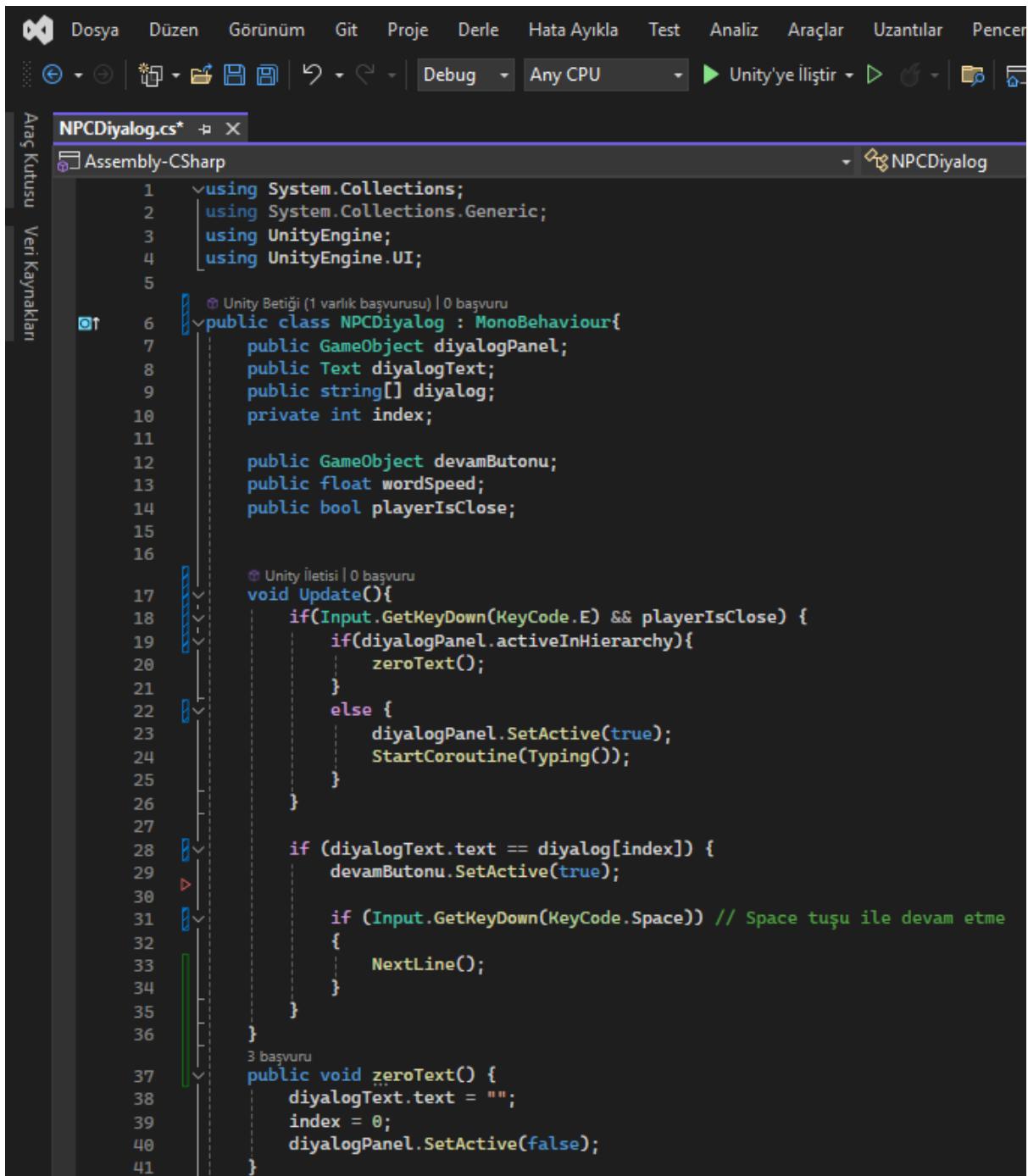
Şekil-21: Köy Alanında Oyuncunun Demirci Karakteri İle Diyalogu



Şekil - 22: Köy Alanında Oyuncunun Marangoz Karakteri İle Etkileşimi



Şekil - 23: Köy Alanında Oyuncunun Marangoz Karakteri İle Diyaloğu

A screenshot of the Visual Studio IDE showing the NPCDialog.cs script. The window title is "NPCDialog.cs". The code implements a MonoBehaviour for NPC dialog. It uses Unity's Input system to handle key presses (E for start, Space for next line) and a coroutine to type text at a specified speed. It also includes a method to clear the text field.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  @Unity Betiği (1 varlık başvurusu) | 0 başvuru
7  public class NPCDialog : MonoBehaviour{
8      public GameObject diyalogPanel;
9      public Text diyalogText;
10     public string[] diyalog;
11     private int index;
12
13     public GameObject devamButonu;
14     public float wordSpeed;
15     public bool playerIsClose;
16
17     @Unity iletişı | 0 başvuru
18     void Update(){
19         if(Input.GetKeyDown(KeyCode.E) && playerIsClose) {
20             if(diyalogPanel.activeInHierarchy){
21                 zeroText();
22             }
23             else {
24                 diyalogPanel.SetActive(true);
25                 StartCoroutine(Typing());
26             }
27         }
28         if (diyalogText.text == diyalog[index]) {
29             devamButonu.SetActive(true);
30
31             if (Input.GetKeyDown(KeyCode.Space)) // Space tuşu ile devam etme
32             {
33                 NextLine();
34             }
35         }
36     }
37     3 başvuru
38     public void zeroText() {
39         diyalogText.text = "";
40         index = 0;
41         diyalogPanel.SetActive(false);
42     }
43 }
```

Şekil - 24: Oyuncunun NPC İle Etkileşime Girme Ve Devam Ettirme Kodu

The screenshot shows the Visual Studio IDE interface with the following details:

- Menu Bar:** Dosya, Düzen, Görünüm, Git, Proje, Derle, Hata Ayıkla, Test, Analiz, Araçlar, Uz.
- Toolbar:** Back, Forward, Search, Build, Run, Debug dropdown set to "Debug", Platform dropdown set to "Any CPU", Unity integration button.
- Code Editor:** File name: NPCDialog.cs\*.cs. Language: C#. The code implements a coroutine for typing text and handles player interactions via OnTriggerEnter2D and OnTriggerExit2D events.
- Solution Explorer:** Shows "Assembly-CSharp".
- Toolbars:** Araç Kutusu (Toolbox) and Veri Kaynakları (Data Sources).

```
41     }
42     2 başvuru
43     IEnumerator Typing() {
44         foreach (char letter in diyalog[index].ToCharArray()) {
45             diyalogText.text += letter;
46             yield return new WaitForSeconds(wordSpeed);
47         }
48     1 başvuru
49     public void NextLine() {
50         devamButonu.SetActive(false);
51         if(index < diyalog.Length -1) {
52             index++;
53             diyalogText.text = "";
54             StartCoroutine(Typing());
55         }
56         else {
57             zeroText();
58         }
59     }
60     ④ Unity iletişı | 0 başvuru
61     private void OnTriggerEnter2D(Collider2D other) {
62         if (other.CompareTag("Player")) {
63             playerIsClose = true;
64         }
65     ④ Unity iletişı | 0 başvuru
66     private void OnTriggerExit2D(Collider2D other) {
67         if (other.CompareTag("Player"))
68         {
69             playerIsClose = false;
70             zeroText();
71         }
72     }
```

Şekil - 25: Oyuncunun NPC İle Etkileşime Girme Ve Devam Ettirme Kodu

### 3.8 YAN OYUN ALANI TASARIMI (Zeynep CIPLAK)

Köy alanının tasarımını zenginleştirdikten hemen sonra oyunumuzu da zenginleştirmek adına karakterimize kendini geliştirmesi ve oynanışa adapte olabilmesi için yan görev alanı oluşturduk. Bu yan görevde oyuncumuz, nehirlerin yakınındaki eşyaları toplayarak görevini takip eder. 20 adet eşya toplanıldığından görevi tamamlanır. Oyuncu, eşyalara temas ettiğinde eşyalar birkaç önceki adımda geliştirdiğimiz envantere alınır ve skor ekranın sol üst köşesindeki text üzerinde görüntülenen. Her item toplandığında konsola mesajlar gönderilir.



*Sekil - 26: Yan Oyun Alani Tasarimi*

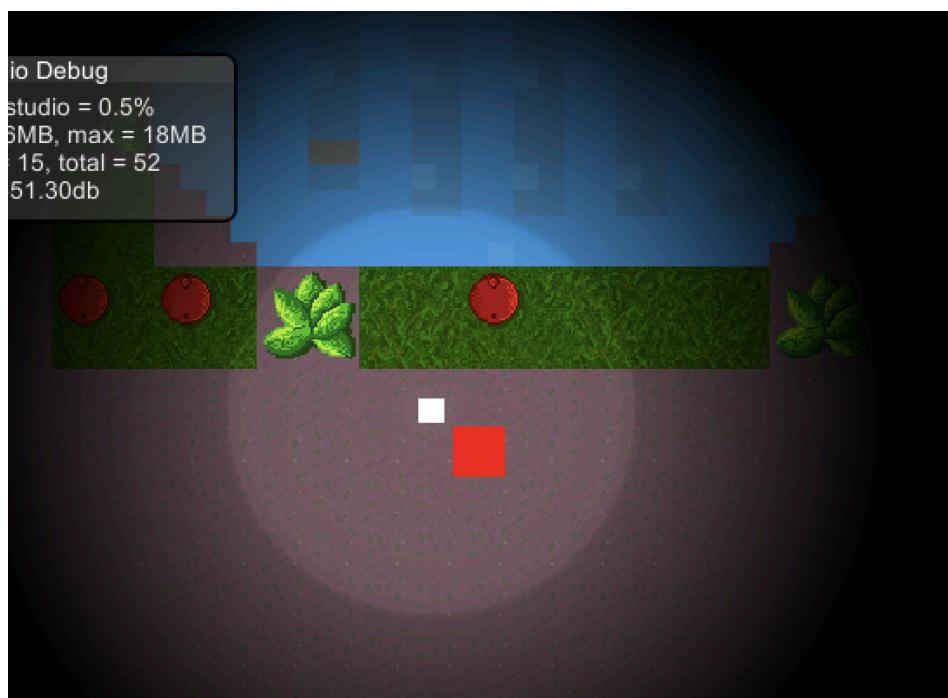
The screenshot shows a Unity code editor window with the file 'coin.cs' open. The code is a C# script for a Unity MonoBehaviour. It includes imports for System.Collections, System.Collections.Generic, UnityEngine, and FMODUnity. The script contains a Start() method that initializes an AudioManager instance, finds the player's transform, and initializes an event emitter. It also includes an OnTriggerEnter2D() method that checks if the player has collected the coin by comparing the tag of the colliding object.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using FMODUnity;
5
6  [RequireComponent(typeof(StudioEventEmitter))]
7
8  public class coin : MonoBehaviour
9  {
10
11
12
13      private bool _collected = false;
14
15      //private AudioManager audioManager;
16
17      private Transform playerTransform; // Store Player's Transform component
18
19      private StudioEventEmitter emitter;
20
21
22  void Start()
23  {
24
25      // audioManager = AudioManager.instance;
26      // Find the GameObject with "Player" tag (assuming it has a Transform component)
27      playerTransform = GameObject.FindGameObjectWithTag("Player").transform;
28
29      emitter= AudioManager.instance.InitializeEventEmitter(FMODEvents.instance.coinIdle, this.gameObject);
30      emitter.Play();
31
32
33
34
35  private void OnTriggerEnter2D(Collider2D collision)
36  {
37
38      /*
39      if (collision.gameObject.CompareTag("Player")) // Check if colliding with "Player" tag
40      {
41          ScoreText.coinAmount += 1;
42          Destroy(gameObject);
43      }
44      */
45      if (collision.gameObject.CompareTag("Player"))
46      {
47          if (!_collected)
48          {
49              /* // coinAS sesini AudioManager üzerinden çal
if (audioManager != null && audioManager.coinAS != null)
```

Sekil - 27: Yan Oyun Alanı Eşyaya ait Kod

### **3.9 GARDİYAN SİSTEMİNİN EKLENMESİ (Zeynep CIPLAK)**

Oyunumuzun yan görev alanını tasarladıkten hemen sonra bu alanda, oyunun zorluk seviyesini artırmak ve oyuncuya dikkat gerektiren bir engel sunmak adına bazı mekanikler ekledik. Bu yan görevde iki adet gardiyan bulunmaktadır. Gardiyanlar, belirli aralıklarla ve belirli bir devriye rotasında ileri geri hareket ederler. Bu, oyunculara gardiyanların hareketlerini tahmin etme ve doğru zamanda doğru hamleyi yapma konusunda meydan okuma sunar. Oyuncu karakter, gardiyan karakterlerine çarptığında, yan oyun alanının başlangıç noktasına geri atılır. Bu geri atılma, oyuncunun ilerlemesini kısmen geri alır ve tekrar denemeye zorlar. Böylece oyuncu, gardiyanlardan kaçınmak ve görevini tamamlamak için daha dikkatli olmak zorunda kalır. Gardiyanların bu devriye sistemi, oyuna stratejik derinlik kazandırırken, oyuncuların reflekslerini ve dikkatlerini test eder. Aynı zamanda, oyuncuların gardiyanların devnimlerini gözlemlemeleri ve doğru zamanda doğru adım atmaları gerektiği için oyunun zorluk seviyesi artar. Bu dinamik, oyuncuların oyundan aldıkları keyfi ve tatmini artırır. Bu yan görevdeki gardiyan sistemi, sadece oyuncunun dikkat ve strateji yeteneklerini geliştirmekle kalmaz, aynı zamanda oyunun genel atmosferine ve heyecanına da katkıda bulunur. Oyuncular, bu tür engellerle başa çıkarken daha fazla odaklanır ve oyunun sunduğu zorlukları aşmanın tatminini yaşırlar.



*Şekil - 28: Gardiyan Sistemi*

The screenshot shows a Unity code editor window with the following details:

- Title Bar:** GuardController.cs
- Class Name:** GuardController
- Method:** Update()
- Code Content:** The code defines a class GuardController that inherits from MonoBehaviour. It contains variables for moveSpeed (2f), moveDistance (10f), startPos (Vector3), movingForward (bool), and currentDistance (float). The Start() method initializes startPos to transform.position. The Update() method checks if movingForward is true. If true, it moves forward by moveAmount (Mathf.Min(moveSpeed \* Time.deltaTime, moveDistance - currentDistance)) and updates currentDistance. If currentDistance reaches moveDistance, movingForward is set to false. If movingForward is false, it moves back by moveBackAmount (Mathf.Min(moveSpeed \* Time.deltaTime, currentDistance)) and updates currentDistance. If currentDistance reaches 0f, movingForward is set to true and currentDistance is reset to 0f.

```
1  using UnityEngine;
2
3  public class GuardController : MonoBehaviour
4  {
5      public float moveSpeed = 2f; // Gardiyanın hareket hızı
6      public float moveDistance = 10f; // İleri gidilecek mesafe
7
8      private Vector3 startPos; // Gardiyanın başlangıç pozisyonu
9      private bool movingForward = true; // Gardiyanın ileri mi geri mi hareket ettiğini kontrol eden flag
10     private float currentDistance = 0f; // Gardiyanın ilerlediği toplam mesafe
11
12     void Start()
13     {
14         startPos = transform.position; // Başlangıç pozisyonunu kaydet
15     }
16
17     void Update()
18     {
19         if (movingForward)
20         {
21             // İleri hareket
22             float moveAmount = Mathf.Min(moveSpeed * Time.deltaTime, moveDistance - currentDistance);
23             transform.Translate(Vector2.right * moveAmount);
24             currentDistance += moveAmount;
25
26             // Belirli mesafeye ulaşıldığında dur
27             if (currentDistance >= moveDistance)
28             {
29                 movingForward = false;
30             }
31         }
32         else
33         {
34             // Geri hareket
35             float moveBackAmount = Mathf.Min(moveSpeed * Time.deltaTime, currentDistance);
36             transform.Translate(Vector2.left * moveBackAmount);
37             currentDistance -= moveBackAmount;
38
39             // Başlangıç noktasına geri dönündüğünde tekrar ileri harekete geç
40             if (currentDistance <= 0f)
41             {
42                 movingForward = true;
43                 currentDistance = 0f; // Mesafeyi sıfırla
44             }
45         }
46     }
47 }
48 }
```

*Şekil - 29: Gardiyan Sistemi Kodu*

### **3.10 ANA MENÜ TASARIMI (Gizem GÜVEN - Zeynep CIPLAK)**

Ses tasarımlı ile aynı şekilde çalışmamızdaki bir başka gidişattan bağımsız bir kısım olan Ana Menü Tasarımı bölümünü buraya yerleştirmeyi uygun gördük. Bu adımda oyunumuzun kullanıcı dostu bir arayüze sahip olması için ana menü sayfası ekledik. Ana menü, oyuncuların oyuna başlamadan önce çeşitli seçeneklere erişimini sağlamaktadır. Bu sayfadaki üç temel buton: 'Oyna', 'Ayarlar' ve 'Oyundan Çık'. 'Oyna' butonuna tıklandığında, oyuncu oyuna kaldığı yerden devam eder. Bu özellik, oyuncuların en son kaydettiği noktadan oyuna hızlıca geri dönmelerini sağlar ve oyun akışının kesintisi ugramamasını sağlar. Böylece, oyuncular oyun deneyimlerine kaldıkları yerden hızlıca devam edebilirler. 'Ayarlar' butonuna tıklandığında, oyuncular ses ayarları gibi gerekli oyun içi ayarlara erişebilirler. Ses ayarları menüsünde, müzik ve efekt ses seviyeleri gibi seçenekler bulunmaktadır. Bu bölümde oyuncular, oyun deneyimlerini kişisel tercihlerine göre özelleştirebilirler. Müzik sesini açma veya kapatma, efekt, genel ses, müzik gibi seçeneklere ayrılmış sesleri azaltma veya artırma bunlara örnek verilebilir. Ses ayarlarının yanı sıra, varsa diğer ayar seçeneklerine de bu menüden ulaşmak mümkündür. 'Oyundan Çık' butonuna tıklandığında ise, oyuncu oyundan tamamen çıkar ve ana ekrana döner. Oyundan çıkmadan önce, oyuncuların ilerlemelerinin kaydedildiğinden emin olunması önemlidir, böylece daha sonra oyuna döndüklerinde kaldıkları yerden devam edebilirler. Ana menü, oyuncuların oyunla etkileşime geçmesini kolaylaştırın ve onlara çeşitli seçenekler sunan bir yapıya sahiptir. Bu kullanıcı dostu tasarım, oyuncuların oyunu daha rahat ve keyifli bir şekilde deneyimlemelerini amaçlamaktadır.



*Şekil - 30: WitDark Ana Menü*



Şekil - 31: WitDark Ayarlar Menüsü

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour
{
    public void PlayGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1); //butona basıldığında sahne yükler.
    }

    public void QuitGame()
    {
        Debug.Log("Oyunu Kapattın!");
        Application.Quit();
    }
}
```

Şekil - 32: WitDark Ana Menü Kodu

The screenshot shows the Unity Editor's code editor window with the file `SettingsMenu.cs` open. The code implements a settings menu with a mute toggle.

```
1  using UnityEngine;
2  using TMPro;
3  using UnityEngine.UI;
4  public class SettingsMenu : MonoBehaviour{
5      [SerializeField] private Toggle muteToggle;
6      private void Awake(){
7          if (!PlayerPrefs.HasKey("Mute")){
8              PlayerPrefs.SetInt("Mute", 0);
9          }
10         else{
11             LoadMuteToogle();
12         }
13     }
14
15
16
17     private void LoadMuteToogle()
18     {
19         muteToggle.isOn = PlayerPrefs.GetInt("Mute") == 1;
20     }
21
22
23     public void MuteToggle()
24     {
25         PlayerPrefs.SetInt("Mute", muteToggle.isOn ? 1 : 0);
26         if (muteToggle.isOn)
27         {
28             AudioListener.pause = true;
29         }
30         else
31         {
32             AudioListener.pause = false;
33         }
34     }
35 }
```

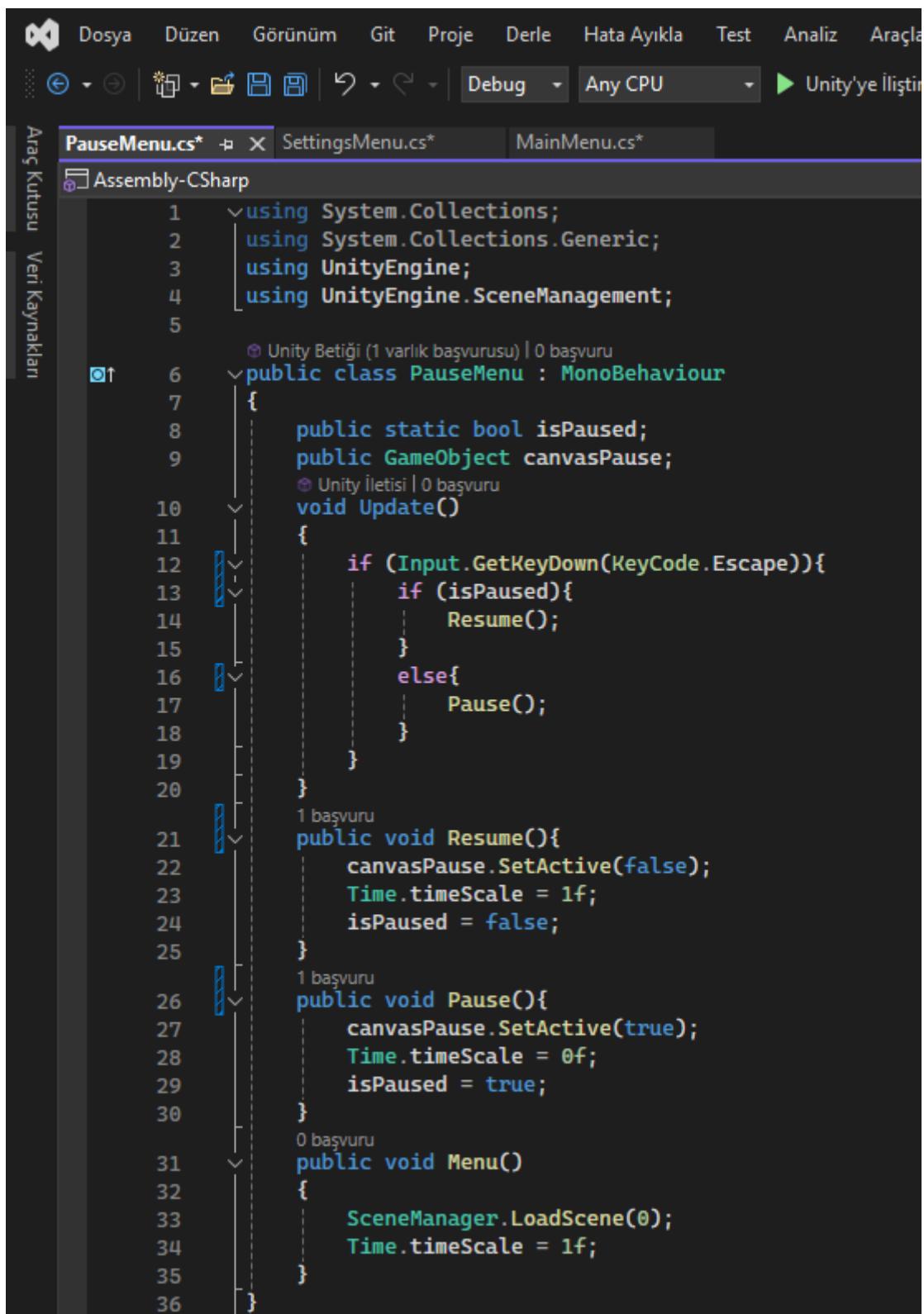
Sekil - 33: WitDark Ayarlar Menüsü Kodu

### **3.11 DURAKLATMA MENÜSÜ TASARIMI (Gizem GÜVEN - Zeynep CIPLAK)**

Ana menü tasarımlının hemen ardından menüler arası geçiş sa�lamak için ve özellikle oyun içerisinde bir oyun için olmazsa olmaz eylemleri oyuncuya sunmak için duraklatma menüsünü tasarlamaya başladık. Oyuncu oyun içerisindeyken 'Escape' tuşuna basarak mevzu bahis duraklatma menüsüne ulaşmaktadır. Bu menüde oyun duraklatılır ve oyunculara üç seçenek sunulur: 'Devam Et', 'Kaydet' ve 'Ana Menü'. 'Devam Et' butonuna tıklandığında oyun kaldığı yerden devam eder, böylece oyuncular oyun akışını kesintiye uğratmadan kaldıkları yerden oyuna dönebilirler. 'Kaydet' butonu, oyuncuların ilerlemelerini kaydetmelerini sağlar. Bu özellik, oyuncuların istedikleri zaman oyunu kaydetmelerine ve daha sonra kaldıkları yerden devam etmelerine olanak tanır. 'Ana Menü' butonuna tıklandığında ise oyuncular ana menüye geri dönerler, bu seçenekle oyunu duraklattıkları yerden çıkararak ana menüye dönme imkanı sunar. Duraklatma menüsü, oyuncuların oyunla etkileşime geçmesini kolaylaştırın ve onlara çeşitli seçenekler sunan bir yapıya sahiptir. Bu kullanıcı dostu tasarım, oyuncuların oyunu daha rahat ve keyifli bir şekilde deneyimlemelerini amaçlamaktadır.



*Sekil - 34: WitDark Duraklatma Menüsü*



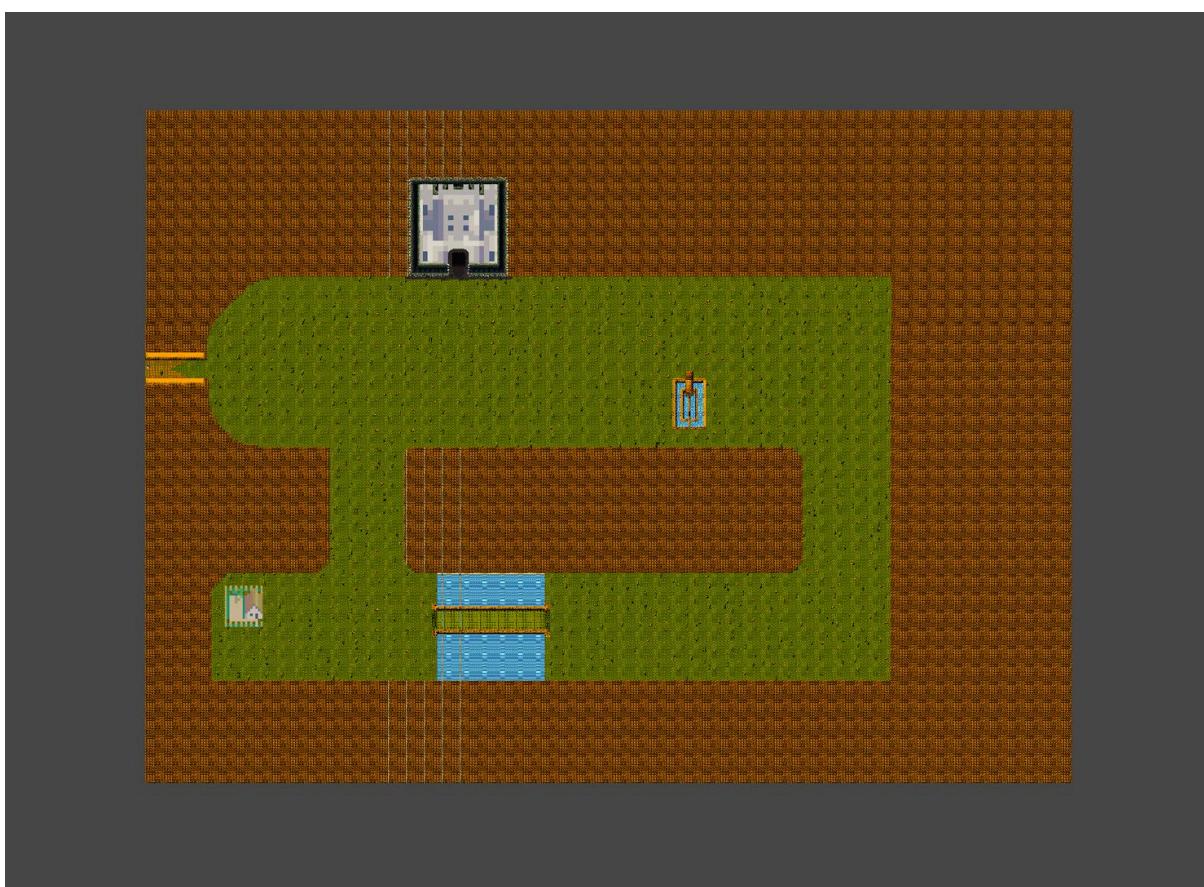
The screenshot shows the Unity Editor's code editor with the file `PauseMenu.cs` open. The code implements a pause menu system for a Unity game. It includes methods for pausing and resuming the game, and a menu method that loads a new scene.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class PauseMenu : MonoBehaviour
7  {
8      public static bool isPaused;
9      public GameObject canvasPause;
10
11     void Update()
12     {
13         if (Input.GetKeyDown(KeyCode.Escape))
14         {
15             if (isPaused)
16                 Resume();
17             else
18                 Pause();
19         }
20     }
21
22     public void Resume()
23     {
24         canvasPause.SetActive(false);
25         Time.timeScale = 1f;
26         isPaused = false;
27     }
28
29     public void Pause()
30     {
31         canvasPause.SetActive(true);
32         Time.timeScale = 0f;
33         isPaused = true;
34     }
35
36     public void Menu()
37     {
38         SceneManager.LoadScene(0);
39         Time.timeScale = 1f;
40     }
41 }
```

Şekil - 35: WitDark Duraklatma Menüsü Kodu

### 3.12 AÇIK DÜNYA HARİTASI (Ömer Faruk DİNÇASLAN)

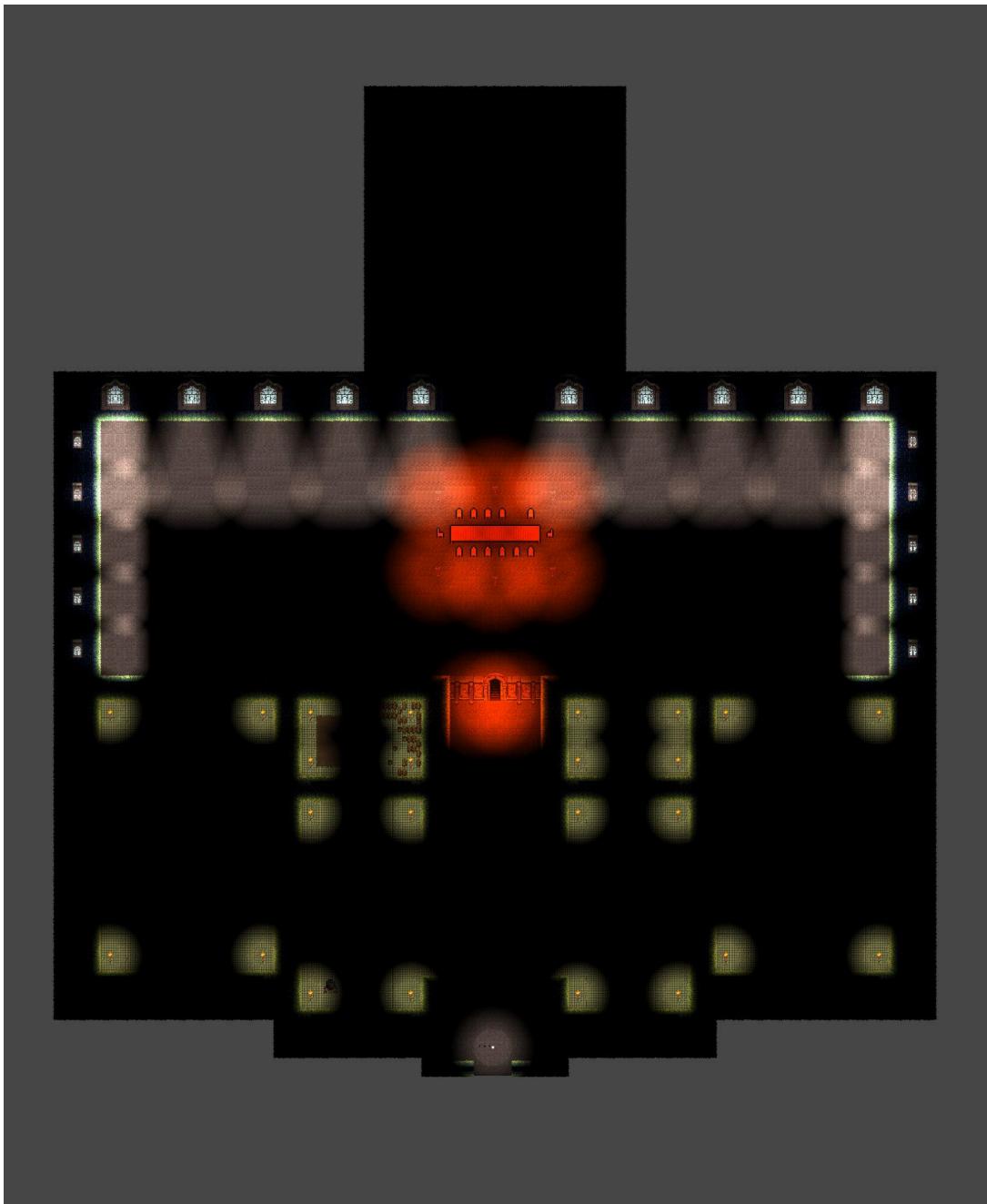
Oyunumuzun temel mekaniklerini oluşturduktan, oynanışı hedeflediğimiz tür olan open-world(açık dünya) oynanışına yaklaştırdıktan hemen sonra, bu oynanış türü için en temel adımlardan biri olan oyunun evreninin haritasını oluşturmaya başladık. Bu alanı, oyunun girişindeki temel oynanışı takiben karakterimizin maceraya başladığı yer olarak tayin ettik. Aynı zamanda oyuncumuz bu alanda iken, lineer olmadan istediği herhangi bir bölgeye erişim sağlayabilecektir.



*Şekil - 36: WitDark Açık Dünya Alanı*

### 3.13 KALE BÖLGESİ (Ömer Faruk DİNÇASLAN)

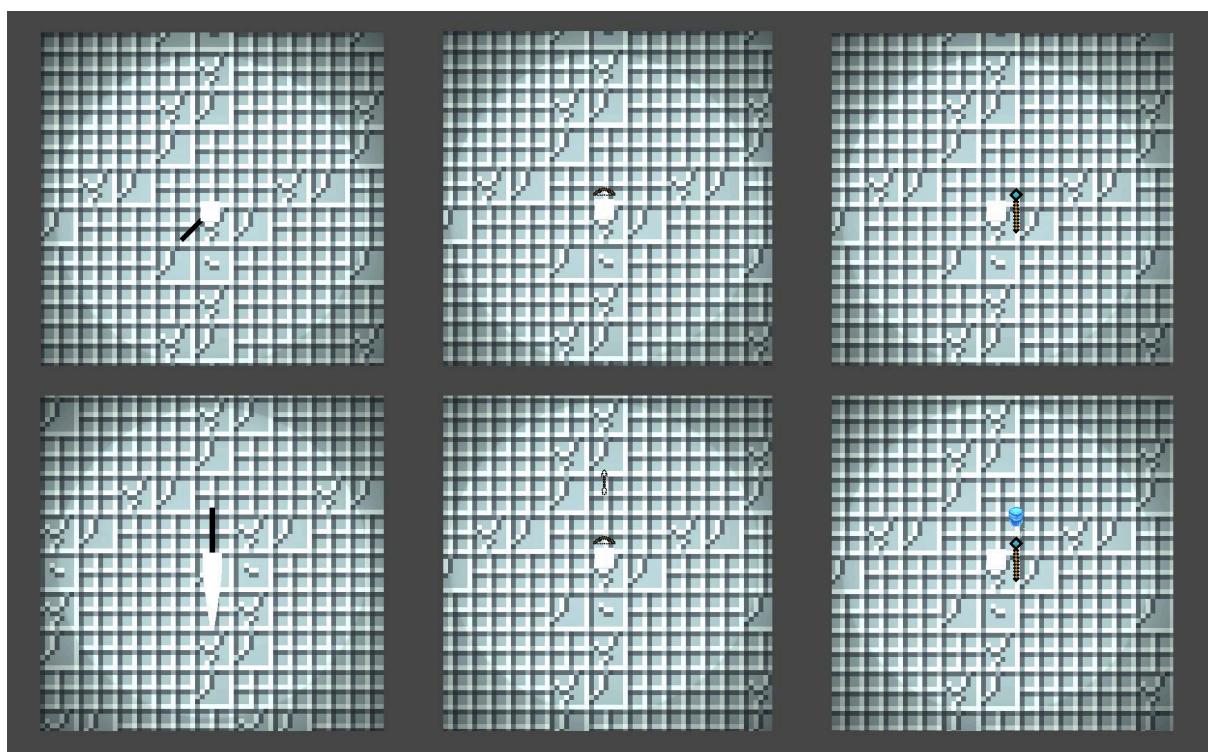
Önceki adımda belirttiğimiz gibi, temel mekanikleri tamamladığımız için oyunumuzu zenginleştirmek için bir diğer alan oluşturduk. Bu alana ise “kale” ismini verdik. Kale bölgesini oluştururken temel amacımız bir çok türdeş oyunun yaptığı gibi bir savaş alanı oluşturmaktır. Bu bölgenin, özellikle oyuncunun dövüş kabiliyetini gösterebileceği bir yer olmasını hedefledik.



*Sekil - 37: Kale Bölgesinin Oyun İçi Gösterimi*

### 3.14 DÖVÜŞ MEKANIĞI (Ömer Faruk DİNÇASLAN)

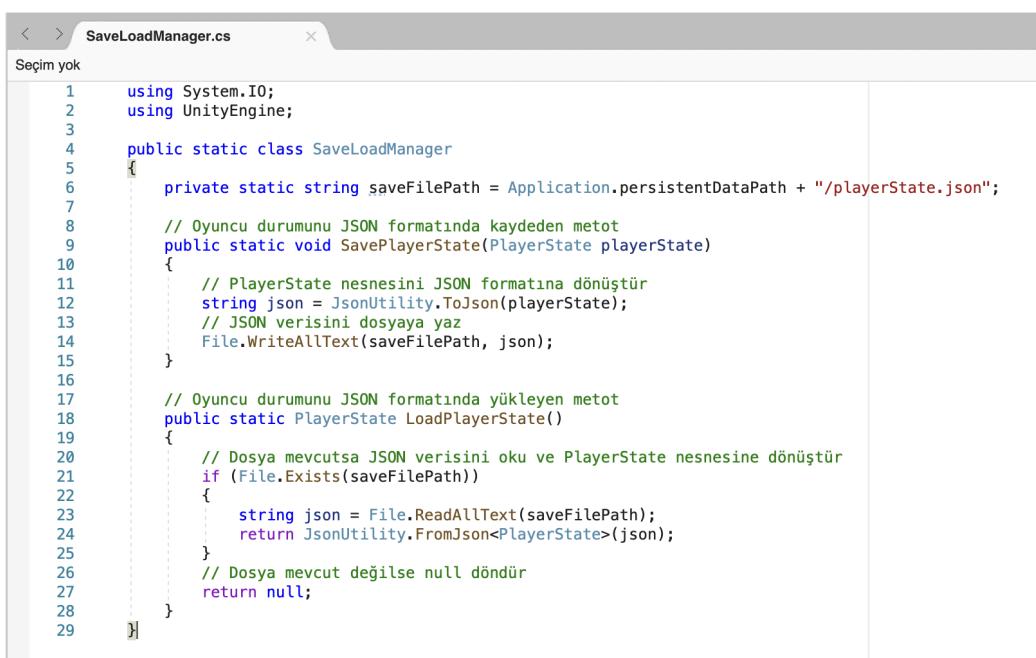
Projemize kale bölgesini eklememizin ardından, bu adımda ise ilgili bölgenin temel amacı olan savaş kabiliyetlerinin tanıtımı ve oynanışı amacıyla daha sağlam bir dövüş mekanığı oluşturduk. Bunun için oyunu muza birkaç silah ekledik, bunlar: kılıç, ok, asa. Her bir silahımızın hasarına ait ayrı bir kod yazdık ve her bir silaha ait farklı animasyonlar oluşturduk. Bu animasyonlar vuruşlardaki ilgili şartlar dahilinde oluşturuldu. Bahsettiğimize örnek olarak; kılıç eşyası oyuncu tarafından kullanıldığı zaman her bir üçüncü vuruşunda daha fazla hasar veriyor, bir oka sahip olmadan yay kullanılamıyor, asa ileride geliştirmeyi hedeflediğimiz mana olmadan kullanılamıyor vb. Bu aşama, aynı zamanda karakterimizin canını ve oluşturduğumuz düşmanların canlarını tanımladığımız kısımdır. Aşağıda, oyuncunun hem silahlara sahip iken hem de silahları kullanırken oyundaki animasyonun nasıl değiştiğini gösteren bir görsel bulunmaktadır.



*Şekil - 38: Kılıç, Ok ve Asa Kullanımına Ait Oynanış Görüntüleri*

### 3.15 SAVE SİSTEMİ (Gizem GÜVEN - Zeynep CIPLAK)

Oyunumuzda yapılan eylemlerin, edinilen eşyaların, varılan noktaların neticesinde elbetteki verilerin bir kaydı tutulması gerekiyordu, bu nedenle oyunumuzun son aşamasında yer verdığımız geliştirme Save Sistemi oldu. Geliştirdiğimiz save sistemi, oyuncuların ilerlemelerini kaydederek oyunu istedikleri zaman terk edip daha sonra kaldıkları yerden devam etmelerine olanak sağlar. Bu sistem, ilk olarak oyuncunun, oyunun ilk sahnesindeki ilerlemesini kaydederek devam etmesi için sunulmuştur. Oyuncu, oyunun ilk sahnesinde ilerlediği noktaya geldiğinde veya bir görevi tamamladığında oyun otomatik olarak ilerlemeyi kaydetmektedir. Bu kayıt, oyuncunun karakterinin konumunu, envanterini ve diğer önemli oyun verilerini içerir. Oyuncu, oyunu kaydetmek için menü seçeneğini kullanabilir. Oyuncu, oyundan çıktıığında veya oyunu kapatıp daha sonra tekrar başlattığında, save sistemi kaldığı noktayı hatırlar. Oyuncu oyunu yeniden başlattığında, save dosyasındaki bilgileri kullanarak oyuna kaldığı yerden devam edebilir. Bu sayede, oyuncular herhangi bir ilerlemeyi kaybetmeden oyunu istedikleri zaman bırakabilir ve daha sonra kolayca devam edebilirler. Save sistemi, oyunculara oyun deneyimlerini daha esnek ve özelleştirilebilir hale getirme imkanı sunar. Oyuncular, oyunun keyfini çıkarırken ilerlemelerini kaydetmek ve istedikleri zaman oyunu devam ettirmek konusunda daha fazla kontrol sahibi olurlar. Bu da oyuncuların oyunu daha keyifli bir şekilde deneyimlemelerini sağlar.



```
< > SaveLoadManager.cs
Seçim yok
1  using System.IO;
2  using UnityEngine;
3
4  public static class SaveLoadManager
5  {
6      private static string saveFilePath = Application.persistentDataPath + "/playerState.json";
7
8      // Oyuncu durumunu JSON formatında kaydeden metot
9      public static void SavePlayerState(PlayerState playerState)
10     {
11         // PlayerState nesnesini JSON formatına dönüştür
12         string json = JsonUtility.ToJson(playerState);
13         // JSON verisini dosyaya yaz
14         File.WriteAllText(saveFilePath, json);
15     }
16
17     // Oyuncu durumunu JSON formatında yükleyen metot
18     public static PlayerState LoadPlayerState()
19     {
20         // Dosya mevcutsa JSON verisini oku ve PlayerState nesnesine dönüştür
21         if (File.Exists(saveFilePath))
22         {
23             string json = File.ReadAllText(saveFilePath);
24             return JsonUtility.FromJson<PlayerState>(json);
25         }
26         // Dosya mevcut değilse null döndür
27         return null;
28     }
29 }
```

Şekil - 39: Save Sistemi Kodu

## **4. SONUÇLAR VE TARTIŞMA**

WitDark adlı projemizde, sesli diyalogların ön plana çıktığı, oyuncuya çevresindeki nesnelerin ve ortamların kendisine anlık olarak sunulduğu bir evren oluşturulmuş, oyuncunun ise bu bilgilerden yararlanarak oyunda edindiği ilerlemede ona zengin bir deneyim sunulabilmesi için açık bir dünya oluşturularak görme engelli bireylerin rahatlıkla oynayabileceği şekilde çeşitli hikaye kurgusu ile desteklenmiş bir oyun geliştirilmiştir.

Projemiz, sesli geri bildirimler ve yönlendirmeler ile görme engelli oyuncuların oyunu rahatça oynamasını sağlamaktadır. Projemiz bünyesinde ana menü, ayarlar ve duraklatma menüsü, envanter menüsü gibi kullanıcı arayüzü bileşenlerini barındırmayan yanı sıra bu bileşenleri sesli olarak desteklemiştir. Yan görevler ve sesli yönlendirmelerle desteklenen ana oyun mekaniği, oyuna stratejik bir derinlik katmıştır. Sesi ön plana çıkaran oyun mekaniklerinin ve arayüz tasarımının, bu alandaki diğer çalışmalar için bir ön ayak olması da bu projenin bir başka misyonudur. Pratik anlamda ise, oyunumuz görme engelli bireylerin oyun oynama deneyimini iyileştirmek ve bu bireylerin dijital oyun dünyasına erişimini artırmayı hedefler. Sonuçlarımızı etkileyen birkaç önemli faktör bulunmaktadır. İlk olarak, sesli geri bildirimlerin kalitesi ve doğru zamanlaması, oyuncunun oynanabilirliğini doğrudan etkilemiştir. Sesli geri bildirimlerin net ve anlaşılır olması, oyuncuların oyunda doğru kararlar verebilmeleri için kritik öneme sahiptir. İkinci olarak, save sistemi oyuncuların ilerlemelerini sorunsuz bir şekilde kaydedip sürdürmeleri açısından çok önemlidir. Oyuncuların oyun içi etkileşimlerinde gardiyanlarının hareketlerini doğru tahmin edebilmeleri için sesli yönlendirmelerin net ve zamanında iletilmesi de önemli bir faktördür. Ayrıca, oyunda bulunan radar sistemi de oyuncuların etraftaki nesneleri algılayabilmesi açısından kritik bir role sahiptir. Radar sistemi, oyunculara çevrelerindeki nesnelerin konumlarını ve hareketlerini anlamalarına yardımcı olur, bu da oyun deneyimini zenginleştirir. Envanter menüsünde, oyuncular görevlerdeki başarı oranlarını görebilirler. Bu özellik, oyuncuların görevlerini ne kadar başarıyla tamamladıklarını takip etmelerini sağlar ve oyuna stratejik bir derinlik katar.

Projede şu anki eksikliklerden birkaçı oyuncunun daha geniş bir görme engelli oyuncu kitlesi tarafından test edilmesi ve geri bildirimlerin toplanması gereksinimine yönelikir. Bu geri bildirimler, oyuncunun erişilebilirliğini ve oynanabilirliğini artırmak için kullanılabilir. Ayrıca, oyuncunun sesli geri bildirimleri daha da geliştirilebilir ve çeşitlendirilebilir.

Gelecekte, oyunumuza daha fazla sesli yönlendirme ve geri bildirim ekleyerek oyuncuların oyunu daha rahat oynamalarını sağlayabiliriz. Ayrıca, oyunun farklı platformlarda da çalışmasını sağlamak için optimizasyon çalışmaları yapılabilir. Son olarak, görme engelli bireyler için diğer oyun türlerinde de benzer erişilebilirlik çözümleri geliştirilebilir ve bu alanda daha fazla araştırma yapılabilir.

## 5. KAYNAKÇA

- KAYA,FATİH, 2017,  
Unity 2D Oyun Tasarımı, herseyimi, <https://www.herseyimi.com/2017/03/21/platform/>
- KALAÇ, MUSTAFA ÖZHAN, TECİM , VAHAP , 2021, Engelsiz Bilişim, Kriter, İstanbul.
- 2023,Non-functional requirement, Wikipedia,  
[https://en.wikipedia.org/wiki/Non-functional\\_requirement](https://en.wikipedia.org/wiki/Non-functional_requirement)
- Univera Bilgisayar Sistemleri, NİSAN 24, 2010, UML ve Modelleme - Bölüm 10  
(Component ve Deployment Diyagramlar),  
<https://univera-ng.blogspot.com/2010/04/uml-ve-modelleme-bolum-10-component-ve.html>
- Sightless Kombat, HAZİRAN 18, 2020, The Last of Us 2 - Blind Accessibility Review,  
<https://caniplaythat.com/2020/06/18/the-last-of-us-2-review-blind-accessibility/>
- MEHMET, 23 KASIM, 2017, Fonksiyonel ve Fonksiyonel Olmayan Gereksinim Nedir?,  
<https://agilebusinessthink.wordpress.com/2017/11/23/fonksiyonel-ve-fonksiyonel-olmayan-gereksinim-nedir/#:~:text=Fonksiyonel%20gereksinim%2C%20bir%20sistemin%20neyi,%E2%80%9Cmust%20do%E2%80%9D%20ile%20kullan%C4%B1l%C4%B1r>