

Beykent Üniversitesi  
Yazılım Mühendisliği Bölümü  
Yazılım Mühendisliği Tasarım Projesi

Rapor 2

2023 – GÜZ

***“THE MIST”***

2003013140 FARANGİS SAKAYİ

2003013066 İREM DEMİREĞEN

2103013802 KARDELEN ŞAŞKIN

### III. Tasarım Dokümantasyonu

#### 8. Tasarım Hedeflerinin Tanımlanması

Oyun projemizin tasarım hedeflerini belirlemek, projeyi başarıyla tamamlamak ve kullanıcı beklentilerini karşılamak için kritik öneme sahiptir. Bu hedefler, kullanıcı deneyimini iyileştirmeye odaklanmayı içerir ve kullanıcı dostu bir ara yüz, çarpıcı grafikler ve ses efektleri gibi özellikleri içermektedir. Ayrıca belirli bir türe girmek veya belirli bir platformu hedeflemek gibi tüm kampanya hedeflerimiz de tutarlı olmalıdır. Bir proje üzerinde çalışırken tasarım hedefleri projenin teknik yapısının belirlenmesine yardımcı olur. Etkin bütçe planlaması ve kaynak yönetimi için de önemlidir. Pazarlama stratejimizi optimize ederek ve kendimize rekabet avantajı kazandırarak başarı şansımızı artırır. Ayrıca proje programlarının belirlenmesi, zamanın yönetilmesi, geliştirici ekip motivasyonunun artırılması gibi konularda da rehberlik sağlar. Sonuçta tasarım hedefleri oyun projemizin başarısını belirler. İlerleme ve çözüme yönelik temel ve odaklanmış bir rehber sağlar. Bizim “THE MIST” oyun projemizde de bunlara nasıl dikkat ettik ve uyguladık onları açılacağız;

#### RELIABILITY (GÜVENİLİRLİK):

Güvenilirlik, bir oyun projesi için oldukça önemlidir çünkü kullanıcıların oyunu sorunsuz bir şekilde deneyimlemelerini sağlar. İşte bu hedefin oyun projemizde nasıl entegre edilebileceği ve neden önemli olduğu:

##### □ Oyun İçi Hata Kontrolü:

Açıklama: Oyuncular, oyun içinde hata yapabilirler. Güvenilir bir sistem, bu hataları kontrol eder ve gerekirse düzeltir.

Uygulama: Kullanıcıdan gelen verileri dikkatlice işleyerek ve hata durumları için uygun geri bildirimler sağlayarak oyun içi güvenilirliği arttırabiliriz.

##### □ Kararlı Oyun Motoru:

Açıklama: Oyunun temelini oluşturan motorun güvenilir ve kararlı olması önemlidir.

Uygulama: Oyun motorunu sürekli olarak güncelleyerek ve test ederek, performans sorunları ve hataları minimize edebiliriz. Buda oyundaki karakterin veya canavarların sürekli olarak buglara veya oyunun hatalarına takılmasını engeller.

##### □ Veri Güvenliği:

Açıklama: Oyuncuların ilerlemeleri, başarıları ve diğer verileri güvenli bir şekilde saklanmalıdır.

Uygulama: Güvenli veri tabanı yönetimi ve şifreleme yöntemleri kullanarak oyuncu verilerini korumalıyız. Kullanıcı oyunda nerde kalmış ne kadarını çözmüş vs bunları görmek ister.

##### □ Hata Günlüğü ve İzleme:

Açıklama: Oyunun çalışması sırasında oluşan hataları izlemek ve kaydetmek.

Uygulama: Hata günlükleri (log) oluşturarak, geliştiricilere sorunları tespit etme ve düzeltme konusunda yardımcı olacak veriler sağlamayı hedefleriz çünkü oyunda hata sürekli takip edilmesi gereken bir durumdur.

## **MODIFIABILITY (DEĞİŞTİRİLEBİLİRLİK):**

Modifiability, oyunumuzun yaşam döngüsü boyunca gelişmeye ve güncellemeye devam edebilmesi için kritik bir unsurdur. Bu tasarım hedefi sayesinde, oyunumuzun gelecekteki değişimlere daha iyi adapte olması mümkün olacaktır. İşte bu hedefin oyun projemize nasıl entegre edilebileceği ve neden önemli olduğu:

### **□ Esnek Oyun Mekanikleri:**

Açıklama: Oyun mekaniklerini esnek ve modüler bir şekilde tasarlamak, gelecekteki güncellemelerde ve eklemelerde daha fazla esneklik sağlar.

Uygulama: Oyun mekaniklerini modüller halinde tasarlayarak, yeni özellikleri kolayca ekleyebilir veya mevcutları değiştirmeyi planladık. Böylelikle Eva karakterimiz yükseltme yeni güç veya bir durum eklendiğinde kolaylıkla güncelleyebiliriz.

### **□ Oyun Dünyası ve Level Tasarımı:**

Açıklama: Oyunun dünyası ve seviye tasarımları, kolayca değiştirilebilir olmalıdır.

Uygulama: Level tasarımlarını ve dünya yapılarını açık kaynak veya modüler bir yapıda düzenleyerek, yeni seviyeler veya bölgeler eklemek daha kolay olmalıdır çünkü bizim oyunumuzda da fazla level ve içinde tonlarca uygulama bulmaca olduğu için tasarım yapmaya açık olmalıdır.

### **□ Grafik ve Görsel Öğeler:**

Açıklama: Grafik öğeleri, karakter tasarımları ve diğer görsel unsurların değiştirilebilir olması, oyunun görünümünü ve tarzını güncellemeyi kolaylaştırır.

Uygulama: Grafik ve görsel unsurları ayrı dosyalarda saklayarak, yeni tasarımları eklemek veya mevcutları değiştirmek için daha esnek bir yapıda yapmaya özen gösterdik çünkü canavarın herhangi bir öğesini değiştirmek, sisin rengini şeklini değiştirmek, tüm aktörlere hızlı ve kolay şekilde ulaşabilmek değiştirilebilirlik açısından önemlidir.

### **□ Oyun Motoru Modifikasyonu:**

Açıklama: Oyun motorunun modifiye edilebilir olması, özel özelliklerin eklenmesini veya mevcut özelliklerin değiştirilmesini sağlar.

Uygulama: Open-source bir oyun motoru kullanarak veya kendi motorunuzu modüler bir şekilde tasarlayarak, ihtiyaca göre modifikasyonları kolayca gerçekleştirebiliriz. Tüm saydığım özellikler oyun motorunun kolay güncellenebilir olmasına bağlıdır buda oyun projelerinde önemli bir özelliktir.

### **□ Kod Modüleritesi ve Yeniden Kullanılabilirlik:**

Açıklama: Kodun modüler olması, belirli bir özelliği değiştirmeyi veya güncellemeyi kolaylaştırır.

Uygulama: Oyun içi sistemleri ve özellikleri modüler sınıflar ve fonksiyonlar kullanarak düzenleyerek, daha kolay modifikasyon imkanı sağlarız. Yani her karakterin her kısmın ayrı kodlanması güncellemeyi oyun motorunun işini kolaylaştırır.

## MAINTAINABILITY (BAKIM KOLAYLIĞI):

Maintainability, bir oyun projesinin uzun ömürlü ve geliştirilebilir olmasını sağlamak için kritik bir tasarım hedefidir. Bu hedefi başarılı bir şekilde uygulayarak, oyunumuzun bakımını kolaylaştırabilir ve geliştirme ekibimizin verimliliğini artırabiliriz. İşte bu tasarım hedefinin oyun projemize nasıl entegre edilebileceği ve neden önemli olduğu:

### □ Kod Okunabilirliği ve Dökümantasyon:

Açıklama: Kodunuzun okunabilir olması ve kapsamlı dökümantasyon içermesi, geliştiricilerin hızlı ve sorunsuz bir şekilde müdahale etmelerini sağlar.

Uygulama: Kod yazarken anlaşılır isimlendirme standartlarına uyarak ve her sınıf, fonksiyon veya modül için açıklayıcı yorumlar ekleyerek okunabilirlik ve anlaşılabilirlik sağlamalıyız.

### □ Modüler Kod Tasarımı:

Açıklama: Oyun içindeki farklı sistemleri modüler bir şekilde düzenlemek, bakım sürecinde herhangi bir sistemi güncelleme veya değiştirme işlemlerini kolaylaştırır.

Uygulama: Oyun içindeki farklı özellikleri modüler sınıflar veya bileşenler halinde organize ederek, her birini bağımsız olarak güncelleme yeteneği kazanabiliriz, çünkü oyun uzun ve bir bütün olduğundan hepsini tüm halde düşünürsek bakımı çok fazla uzun ve zor yapılıdır.

### □ Oyun Kaynaklarının Yönetimi:

Açıklama: Oyun içinde kullanılan kaynaklar (grafikler, sesler, vs.) düzenli bir şekilde yönetildiğinde, yeni kaynak eklemek veya mevcutları değiştirmek daha kolay olur.

Uygulama: Kaynakları belirli kategorilere göre düzenleyerek ve yönetim sistemleri kullanarak, bakım sürecinde kaynakları hızlı bir şekilde güncelleyebiliriz. Oyunumuz biraz gerilim biraz korku olduğundan dolayı ses ve efektler ekstra önemli oluyor bu sebeple kolaylıkla yönetilebilir olmalıdır.

### □ Hata İzleme ve Loglama:

Açıklama: Hataları izlemek ve kaydetmek, bakım sürecinde hataları hızlı bir şekilde tanımlamanıza ve düzeltmenize yardımcı olur.

Uygulama: Hata izleme araçları ve loglama sistemleri kullanarak, oyunun çalışması sırasında ortaya çıkan hataları kaydedebilir ve analiz edebiliriz ve sonrasında da oyunu sürekli güncel tutmamız mümkün olur.

### □ Otomatik Testler:

Açıklama: Otomatik testler, her güncelleme sonrasında oyunun temel özelliklerini test etmenizi sağlar, böylece hataları erken aşamada yakalayabilirsiniz.

Uygulama: Unit testleri ve otomasyon testleri kullanarak, oyunumuzun kritik özelliklerini düzenli olarak test edebiliriz. Ve bu sayede hata ayıklamak, bulmak çok daha kolay bir hale gelir.

## UNDERSTANDABILITY (ANLAŞILABİLİRLİK):

Understandability tasarım hedefi, projenin karmaşıklığını yönetmek ve ekibin projeyi daha etkili bir şekilde geliştirmesini sağlamak için kritiktir. Bu hedefi başarılı bir şekilde uygulayarak, geliştirme ekibimizin uyum sağlaması ve verimliliği artabilir.

İşte bu tasarım hedefinin oyun projemize nasıl entegre edilebileceği ve neden önemli olduğu:

### □ Temiz ve Anlaşılır Kod Yapısı:

Açıklama: Kodunuzun temiz, düzenli ve anlaşılır bir yapıda olması, diğer geliştiricilerin veya ekibe yeni katılanların kodu hızlıca anlamalarını sağlar.

Uygulama: Anlamlı değişken isimleri kullanmalıyız, kodunuzu modüler hale getirmeliyiz. ve ilgili kod bloklarını mantıklı bir şekilde düzenleyerek anlaşılabilir bir kod tabanı oluşturulmalıyız çünkü sürdürülme anlaşılma açısından önemli bir etkidir.

### □ Dokümantasyon:

Açıklama: Projenin genel mimarisi, kullanılan algoritmalar ve önemli kararlar hakkında kapsamlı dokümantasyon, ekibin projeyi anlamasını kolaylaştırır.

Uygulama: README dosyaları, inline yorumlar ve belgelendirme araçları kullanarak, projemizin yapısını ve çalışma şeklini detaylı bir şekilde açıklamalıyız ve bu şekilde oyunumuzun anlaşılması artar ve herkes verimli bir şekilde çalışabilir.

### □ Modülerlik:

Açıklama: Oyunumuzdaki farklı özellikleri modüler bir şekilde düzenlemek, her bir modülü izole etmek ve bu modüller arasındaki ilişkileri açıklamak, projenin anlaşılabilirliğini artırır.

Uygulama: Ana oyun motoru, grafik işleme, oyun mekaniği gibi farklı modülleri ayrı sınıflar veya paketler halinde düzenleyerek, her modülün görevini açıkça belirtmeliyiz. Bu şekilde yaptığımız oyunun her noktasına kolayca ulaşma ve anlamamızı sağlar.

## ADAPTABILITY (ADAPTASYON YETENEĞİ):

Adaptasyon yeteneği, oyun projemizin uzun ömürlü ve başarılı olabilmesi için kritik bir faktördür. Değişen teknolojik, pazarlama ve kullanıcı beklentilerine hızlıca adapte olabilen bir oyun, rekabet avantajı sağlayabilir ve oyuncuların sürekli ilgisini çekebilir.

Bu tasarım hedefini projemize entegre etmek, oyunumuzu uzun vadede sürdürülebilir kılmak ve değişen ihtiyaçlara hızlı bir şekilde cevap verebilmek açısından önemlidir. İşte bu tasarım hedefinin oyun projemize nasıl entegre edilebileceği ve neden önemli olduğu:

### □ Modüler ve Esnek Mimariler:

Açıklama: Oyunumuzdaki farklı özellikleri ve bileşenleri modüler bir şekilde tasarlamak, gelecekte eklenmesi veya değiştirilmesi gereken özelliklere uyum sağlamayı kolaylaştırır.

Uygulama: Her oyun mekaniği, grafik işleme veya yapay zeka bileşeni gibi farklı özellikleri ayrı modüller halinde tasarlamalıyız ve bu modüller arasındaki bağlantıları esnek hale getirmeliyiz bu sayede oyuna güncelleme ve yeni özellik getirmek istediğimizde bu esneklik bize önemli bir kolaylık sağlar.

#### □ **Veri Yönetimi ve Ayarlar:**

Açıklama: Oyunumuzdaki ayarları, konfigürasyonları ve dinamik verileri dışa aktarılabilir ve değiştirilebilir bir şekilde yönetmek, oyunumuzu daha adapte edilebilir hale getirir.

Uygulama: Oyun içi ayarları dış bir konfigürasyon dosyasından okuma, oyun içindeki değişkenleri dinamik olarak güncelleme ve içerik güncellemelerini kolayca entegre etme mekanizmaları oluşturmamızdır. Ayarlar kolay yönetilebilir hale gelmesi gerekir projemizde.

#### □ **Güncelleme ve Yama Mekanizmaları:**

Açıklama: Oyunumuzu güncel tutmak ve hataları düzeltmek için etkili bir güncelleme veya yama mekanizması, oyunun hızlı bir şekilde uyum sağlamasını sağlar.

Uygulama: Otomatik güncelleme sistemleri veya kullanıcıya basit yükleme seçenekleri sunarak, oyunumuzu sürekli olarak güncel ve sorunsuz tutmalıyız, çünkü oyunun teknolojinin gerisinde kalmaması çok önemlidir. Ve bizim oyunumuz geleceği hastalığı hedef aldığı için teknolojiye daha fazla ayak uydurmalıdır.

#### □ **Community ve Kullanıcı Geri Bildirimi:**

Açıklama: Oyunumuzu sürekli olarak geliştirmek ve kullanıcı beklentilerine uyum sağlamak için bir community ve geri bildirim mekanizması kurun.

Uygulama: Kullanıcı geri bildirimlerini düzenli olarak değerlendirmeliyiz, popüler taleplere öncelik verin ve toplulukla etkileşimde bulunarak oyununuzu kullanıcı taleplerine uyumlu hale getirmeliyiz mesela kullanıcının buna benzer bir oyun çıktığında veya hikâyenin içinde beğenmediği bir şey olduğunda bunu dikkate almalıyız.

### **REUSABILITY (YENİDEN KULLANILABİLİRLİK) :**

Yeniden kullanılabilirlik, bir oyun projesinde kullanılan bileşenlerin, modüllerin veya kod bloklarının başka yerlerde tekrar kullanılabilir olma yeteneğini ifade eder. Bu tasarım hedefi, geliştirme sürecini hızlandırabilir, bakım maliyetlerini azaltabilir ve kodun daha tutarlı olmasını sağlayabilir. İşte bu tasarım hedefinin oyun projemize nasıl entegre edilebileceği ve neden önemli olduğu:

#### □ **Oyun Motoru ve Framework Kullanımı:**

Açıklama: Eğer geniş bir oyuncak kutusu gibi kullanılabilen bir oyun motoru veya framework kullanılıyorsa, bu, projemizdeki birçok temel bileşeni içerir ve geliştirme sürecinizde yeniden kullanılabilirliği artırır.

Uygulama: Unity, Unreal Engine veya diğer popüler oyun motorları ve framework'leri gibi araçları projemize entegre edin ve bu araçların sunduğu standart özellikleri kullanın. Bizde oyunumuzu bu platformlarda yaparak daha çok olanak ve kullanım sağlayacağız.

#### □ **Modüler Oyun Mekanikleri:**

Açıklama: Oyunumuzun farklı bölümlerinde veya seviyelerinde kullanılabilecek modüler oyun mekaniği parçaları oluşturmak, yeniden kullanılabilirliği artırır.

Uygulama: Örneğin, bir platformer oyunu geliştiriyoruz zıplama, koşma veya düşman yapısı gibi temel oyun mekaniği modüllerini oluşturduk ve bunları farklı seviyelerde kullanım sağlayacağız.

#### □ **Kütüphane ve Arayüzler:**

Açıklama: Sık kullanılan işlevleri içeren bir kütüphane oluşturmak veya genel arayüzleri standartlaştırmak, kodumuzu başka projelerde veya farklı bileşenlerde kullanılabilir kılar.

Uygulama: Oyun içi ses yönetimi, animasyon kontrolü veya kullanıcı arayüzü elemanları gibi genel işlevleri içeren bir kütüphane oluşturmak sonrasında bunları kullanmak bize önemli bir kolaylık sağlar.

#### □ **Grafik ve Tasarım Varlıkları:**

Açıklama: Oyun içi grafik, animasyonlar veya ses efektleri gibi varlıkları standart bir formatta ve katalogta düzenleyerek, farklı projelerde veya oyun seviyelerinde yeniden kullanabilir hale getirmeliyiz.

Uygulama: Oyun içi karakter modelleri, arka planlar veya efektler gibi varlıkları genel bir kütüphane veya asset havuzu içinde organize etmeliyiz böylelikle üst levellerde veya oyun içindeki ilerideki tasarımlarda bunu kolaylıkla kullanabiliriz.

#### □ **Dış Kaynak Kullanımı:**

Açıklama: Harita tasarımı, karakter animasyonları veya ses tasarımı gibi belirli uzmanlık alanlarında dış kaynakları kullanmak, projemizdeki belirli bileşenlerin yeniden kullanılabilir olmasını sağlayabilir.

Uygulama: Uzmanlık gerektiren alanlarda profesyonel dış kaynakları projenize entegre edin ve bu kaynakları farklı projelerde veya seviyelerde kullanabilmeliyiz. Mesela yaratacağımız canavar aktörünü her levelde kullanabilmek için uyumlu ve tam bir entegre sağlanmalıdır.

### **EFFİCIENCY (VERİMLİLİK) :**

Verimlilik, bir oyunun kaynakları (donanım, yazılım ve diğer) ne kadar etkili bir şekilde kullandığını ve oyunun düzgün bir şekilde çalıştığını belirten bir tasarım hedefidir. Verimlilik, oyunumuzun genel performansını artırırken, aynı zamanda daha geniş bir oyuncu kitlesine hitap etmesini sağlar. Bu nedenle, projemizde verimliliği artırmak için çeşitli stratejiler kullanmak, oyunumuzun daha iyi çalışmasına ve daha olumlu bir kullanıcı deneyimine yol açabilir. İşte bu hedefin projemizde nasıl uygulanabileceği ve neden önemli olduğu:

#### □ **Oyun Motoru Optimizasyonu:**

Açıklama: Oyun motorunun ve kullanılan diğer yazılım araçlarının performansını optimize etmek, oyunun düşük sistem gereksinimleriyle daha geniş bir oyuncu kitlesine ulaşmasını sağlar.

Uygulama: Grafik ayarları, gölgeleme efektleri ve fizik motoru gibi oyun motoru özelliklerini optimize etmeli ve Gereksiz işlemleri ve kaynakları kullanmayan bir oyun motoru seçmeliyiz çünkü oyunumuzda efektler ve görseller çok önemli olduğundan gereksiz hiçbirşey olmamalı.

#### □ **Grafik ve Animasyon Optimizasyonu:**

Açıklama: Oyun içi grafik ve animasyonların efektif bir şekilde kullanılması, oyunun akıcı bir şekilde çalışmasını sağlanmalı.

Uygulama: LOD (Level of Detail) sistemleri kullanarak uzak nesnelerin ayrıntı seviyelerini azaltın. Tek bir animasyonu birden çok nesne için paylaşın ve gereksiz detayları ortadan kaldırmalıyız. Çünkü genel olarak oyunumuz karanlık ve kasvetli geçeceğinden uzaktakilerin gereksiz detaylarına ihtiyaç olmamalıdır.

#### □ **Yüksek Performanslı Grafik ve Ses Motorları:**

Açıklama: Grafik ve ses işleme motorlarının performansını artırmak, oyunun daha etkileyici ve akıcı bir deneyim sunmasına yardımcı olur.

Uygulama: GPU (Graphics Processing Unit) için özel optimizasyonlar kullanarak grafik işlemleri optimize etmeliyiz. SES işleme için hafif ve hızlı kütüphaneleri tercih etmeliyiz. Canavar aktörümüz bulunduğu için ses efektleri o gerilimiz vermek oldukça önemlidir.

#### **TRACEABILITY OF REQUIREMENTS (GEREKSİNİM TAKİP EDİLEBİLİRLİĞİ):**

"Traceability of requirements" bir oyun projesi için önemli bir tasarım hedefi olabilir. Bu hedef, proje sürecindeki her adımın başlangıcından itibaren belirlenen gereksinimlere olan uyumu izlemeyi ve sürdürmeyi amaçlar. İşte bu hedefi projemize uygun şekilde detaylandırmak için bazı açıklamalar:

#### □ **Gereksinimlerin Belirlenmesi:**

Açıklama: Oyunumuzun başlangıcında, müşteri ihtiyaçlarını ve proje hedeflerini belirleyin. Bu, oyununuzun temel gereksinimlerini anlamaya yardımcı olacaktır.

Uygulama: Kapsamlı bir gereksinim analizi yapmalıyız. Müşteri toplantıları, kullanıcı anketleri ve iş analizi gibi yöntemleri kullanarak gereksinimleri belirlemeliyiz, böylelikle oyunumuzun yaş grubunu vs öğrenebiliriz.

#### □ **Gereksinimlerin Belgelendirilmesi:**

Açıklama: Belirlenen gereksinimleri detaylı bir şekilde belgeleyin. Bu belgeler, geliştirme ekibinin ve diğer paydaşların gereksinimleri anlamalarına yardımcı olur.

Uygulama: Gereksinim belgesi oluşturulmalı. Gereksinimlerin öncelik sırasına göre düzenlenmiş, anlaşılır ve izlenebilir olmasını sağlamalıyız.

#### □ **Test Sürecinde Gereksinim İzlemesi:**

Açıklama: Test aşamasında, her gereksinimin doğru bir şekilde test edildiğinden emin olun. Bu, yazılımın gereksinimlere uygunluğunu doğrulamaya yardımcı olur.

Uygulama: Test senaryolarını ve test durumlarını belirlemeliyiz. Her biri belirlenen gereksinimle eşleşen testlerin sonuçlarını izlemeliyiz.

#### **FAULT TOLERANCE (HATA TOLERANSI):**

"Fault tolerance" (Hata Toleransı), bir oyun projesi için önemli bir tasarım hedefi olabilir. Bu hedef, oyununuzun kullanıcılar veya dış etkenler tarafından meydana gelebilecek hatalara karşı dayanıklı olmasını amaçlar. İşte bu hedefi projemize uygun şekilde detaylandırmak için bazı açıklamalar:

#### □ **Kullanıcı Hatalarına Karşı Dayanıklılık:**

Açıklama: Oyunumuzu kullanırken oyuncuların yapabileceği hatalara karşı dirençli olun. Bu, yanlış girişler, hatalı tıklamalar veya beklenmedik kullanım durumları gibi hataları içerir.

Uygulama: Giriş kontrolleri ve doğrulamaları eklemeliyiz. Oyuncunun yanlış giriş yapmasına veya beklenmeyen durumlara maruz kalmasına karşı önlemler alınmalı çünkü kullanıcı bulmaca çözerken yanlış bir şeye basabilir bunun için direk iptal olmamalıdır.



#### □ **Oyun İçi Hatalara Karşı Planlama:**

Açıklama: Oyun içi hataların (örneğin, programlama hataları) meydana gelmesi durumunda sisteminizin stabil kalmasını sağlayacak bir strateji belirlenmeli.

Uygulama: Hata günlükleri (log) oluşturun ve bu günlükleri düzenli olarak kontrol etmeliyiz. Hata durumlarını otomatik olarak tespit etmek ve düzeltmek için önlemler almalıyız çünkü oyunda oluşan hatalar kullanıcıyı kötü etkilemektedir.

#### □ **Oyun Sürekliliği:**

Açıklama: Oyunumuzun sürekli olarak çalışabilir olmasını sağlamak için hata durumlarında sistemlerin otomatik olarak kurtarılmasını planlamalıyız.

Uygulama: Yedekleme sistemleri ve otomatik kurtarma mekanizmaları eklemeliyiz. Oyun içindeki hata durumlarında sistemleri otomatik olarak yeniden başlatma veya kurtarma planları oluşturmalıyız.

#### □ **Kullanıcıya Dostu Hata Mesajları:**

Açıklama: Kullanıcıların karşılaştığı hataları anlamalarına ve çözmelerine yardımcı olacak dostane hata mesajları sağlayın.

Uygulama: Kullanıcı dostu ve açıklayıcı hata mesajları oluşturun. Oyunculara sorunlarını anlamaları ve çözmeleri için rehberlik etmeliyiz, çünkü kullanıcı herhangi bir sebepten çökme veya bağlanamama hatası alabilir bir pc oyunu olacağı için internete bağlı olabilir kullanıcının anlaması için bunları açıklamalıyız.

### **COST-EFFECTIVENESS (MALİYET ETKİNLİĞİ):**

"Cost-effectiveness" (Maliyet Etkinliği), bir oyun projesi için önemli bir tasarım hedefi olabilir. Bu hedef, projenin bütçe sınırları içinde kalmasını ve kaynakların en etkili şekilde kullanılmasını amaçlar. İşte bu hedefi projenize uygun şekilde detaylandırmak için bazı açıklamalar:

#### □ **Maliyet Analizi:**

Açıklama: Her bir geliştirme aşamasında ve projenin farklı bileşenlerinde oluşan maliyetleri ayrıntılı bir şekilde analiz edin. Bu, geliştirme sürecinde belirli bir aşamada veya bileşende maliyet artışları veya azalmalarını izlemenize yardımcı olacaktır.

Uygulama: Geliştirme aşamaları, personel maaşları, donanım, yazılım lisansları, pazarlama ve diğer gider kalemleri üzerinde düzenli olarak maliyet analizi yapın. Bütçe sınırları içinde kalmak için her aşamada ve bileşende maliyet etkin çözümleri değerlendirin.

#### □ **Yeniden Kullanılabilirlik:**

Açıklama: Daha önce geliştirilen modüllerin veya bileşenlerin yeniden kullanılabilir olması, geliştirme sürecinde zaman ve maliyet tasarrufu sağlar. Aynı zamanda, daha önce test edilmiş ve sağlam bileşenlerin kullanılması, hata oranını azaltabilir.

Uygulama: Modüler bir yaklaşım benimseyerek, geliştirilen kodun mümkün olduğunca yeniden kullanılabilir olmasını sağlanmalı. Bu hem maliyet hem de zaman açısından avantaj sağlar.

#### □ Performans ve Kaynak Kullanımı Optimizasyonu:

Açıklama: Oyunumuzun performansını artırmak için optimize edilmiş kodlar ve kaynak kullanımı önemlidir. Bu, hem oyunun daha akıcı çalışmasını sağlar hem de gereksiz kaynak tüketimini önler.

Uygulama: Geliştirme sürecinde düzenli olarak performans testleri yapılmalı. Gereksiz kod parçalarını ve kaynak tüketen işlemleri belirleyip optimize ederek maliyeti düşürebiliriz.

#### ROBUSTNESS (SAĞLAMLIK):

"Robustness" (Sağlamlık), bir oyun projesi için önemli bir tasarım hedefi olabilir. Bu, oyunun çeşitli koşullar altında stabil, güvenilir ve hataya dayanıklı olmasını sağlamayı amaçlar. İşte bu hedefi projenize uygun şekilde detaylandırmak için bazı açıklamalar:

#### □ Veri Güvenliği ve İyileştirme:

Açıklama: Oyun içindeki oyuncu verilerinin güvenliği önemlidir. Veri kaybını önlemek, güvenilir bir şekilde oyuncu ilerlemesini ve başarılarını saklamak için uygun mekanizmalar sağlanmalıdır.

Uygulama: Veri tabanı güvenlik önlemleri alınmalı, düzenli olarak veri yedeklemeleri yapılmalı ve veri kaybını önlemek için hata durumlarında geri dönüşler sağlanmalıdır. Çünkü bu oyunumuzda kaldığımız yerden devam etme olduğu için verilerin saklanması önemlidir.

#### □ Hata Yakalama ve İzleme:

Açıklama: Oyun içindeki hataların yakalanması, izlenmesi ve kaydedilmesi, geliştiricilere hızlı bir şekilde müdahale etme imkanı tanır. Bu, kullanıcıların olası hatalarla karşılaştıklarında daha iyi destek almasını sağlar.

Uygulama: Hata izleme araçları kullanılmalı, kullanıcıların karşılaştığı hatalar kaydedilmeli ve bu hataların geliştiricilere bildirilmesi otomatikleştirilmelidir.

#### □ Dinamik Oyun Senaryolarına Uyum:

Açıklama: Oyunun, oyuncuların beklenmedik veya hatalı girişlere karşı dirençli olması önemlidir. Bu, kullanıcıların oyun sırasında beklenmeyen durumlarla karşılaştığında oyunun çökmesini önler.

Uygulama: Kullanıcıların beklenmedik girişlere karşı nasıl tepki verdiğini gözlemleyin ve bu senaryoları ele alacak şekilde oyununuzu güncelleyin.

#### □ Düşük Kaynak Kullanımı ve Performans Optimizasyonu:

Açıklama: Oyunun düşük kaynaklarda (RAM, CPU, GPU) çalışabilmesi, cihazların sınırlı kaynaklarına karşı dirençli olması gerekir. Bu, geniş bir kullanıcı kitlesine ulaşmayı sağlar.

Uygulama: Oyunun optimize edilmiş kodlar içermesi, düşük performanslı cihazlarda test edilmesi ve gerektiğinde performans optimizasyonlarının yapılması sağlanmalıdır.

#### Neden Sağlamlık Önemlidir:

Kullanıcı Memnuniyeti: Oyunun stabil ve hatasız olması, kullanıcıların memnuniyetini artırır.

Oyunun Güvenilirliği: Oyuncuların oyunun güvenilir olduğuna inanması, uzun vadeli bir başarı için önemlidir.

Müşteri Desteği Azaltır: Sağlam bir oyun, kullanıcı hataları ve şikayetleri nedeniyle müşteri desteği ihtiyacını azaltır.

Uzun Vadeli Başarı: Sağlamlık, oyununuzun uzun vadeli başarısını sürdürmek için önemlidir.

## **HIGH-PERFORMANCE (YÜKSEK PERFORMANS) :**

"High-performance" (Yüksek Performans), bir oyun projesi için önemli bir tasarım hedefi olabilir. Bu, oyunun hızlı çalışması, akıcı bir oyun deneyimi sunması ve yüksek grafik veya hesaplama gereksinimlerini karşılaması anlamına gelir. İşte bu hedefi projenize uygun şekilde detaylandırmak için bazı açıklamalar:

### **□ Frame Rate Optimizasyonu:**

Açıklama: Oyunun herhangi bir cihazda yüksek frame rate (kare hızı) sunabilmesi önemlidir. Bu, oyunun daha akıcı ve hızlı görünmesini sağlar.

Uygulama: Oyunun grafik ve animasyon unsurları optimize edilmeli, düşük performanslı cihazlarda bile yüksek frame rate elde edilmelidir.

### **□ Düşük Gecikme Süresi:**

Açıklama: Oyuncuların girişlerine hızlı tepki vermek, düşük gecikme süresi gerektirir. Bu, oyun kontrolünün daha duyarlı ve gerçek zamanlı hissedilmesini sağlar.

Uygulama: Oyunun giriş işleme süreleri optimize edilmeli, ağ iletişimi ve oyun motoru gecikmeleri minimize edilmelidir.

### **□ Yüksek Çözünürlük ve Grafik Kalitesi:**

Açıklama: Oyunun yüksek çözünürlükte ve kaliteli grafiklerle görsel olarak etkileyici olması önemlidir. Bu, oyunculara görsel bir zevk sunar.

Uygulama: Grafik motoru ve çizim algoritmaları optimize etmeliyiz, yüksek kaliteli tekstürler ve efektler kullanılmalıdır. Çünkü oyunumuz grafik ve gerilime dayalı olduğu için çözünürlük vs yüksek olmalıdır.

### **□ Paralel İşleme ve Çoklu İşlemci Desteği:**

Açıklama: Oyunun çeşitli işlemleri paralel olarak işleyebilmesi, çoklu işlemci sistemlerinden yararlanması performansı artırır.

Uygulama: Çoklu iş parçacığı (thread) desteği eklenmeli, hesaplama yoğun işlemler paralel olarak çalışacak şekilde tasarlanmalıdır.

### **□ Optimize Edilmiş Bellek Yönetimi:**

Açıklama: Oyunun bellek kullanımının etkili bir şekilde yönetilmesi, performansı artırır ve bellek sızıntıları önler.

Uygulama: Bellek kullanımı izlenmeli, gereksiz bellek aşırı kullanımı önlenmeli ve bellek sızıntıları düzeltilmelidir.

## **Neden Yüksek Performans Önemlidir:**

Akıcı Oyun Deneyimi: Yüksek performans, oyunun akıcı bir şekilde çalışmasını sağlar, bu da daha keyifli bir oyun deneyimi sunar.

Geniş Kitleye Ulaşma: Düşük ve yüksek performanslı cihazlarda çalışabilen oyunlar, geniş bir oyuncu kitlesine ulaşabilir.

Teknolojik Standartları Karşılama: Yüksek performans, oyunun güncel teknolojik standartları karşılamasını sağlar ve oyununuzun gelecekte de rekabetçi kalmasına yardımcı olur.

## **GOOD DOCUMENTATION (İYİ DOKÜMANTASYON):**

"Good documentation" (İyi Dokümantasyon), bir oyun projesi için oldukça önemli bir tasarım hedefidir. İyi dokümantasyon, oyun geliştirme sürecini düzenler, ekip içi iletişimi güçlendirir ve proje süresince karşılaşılabilecek sorunlara çözümler sunar. İşte bu hedefi projenize uygun şekilde detaylandırmak için bazı açıklamalar:

### **□ Proje Genel Bakış Dokümanı:**

Açıklama: Oyunun genel tasarımı, oynanış mekaniği, karakterler, dünya tasarımı ve hikaye özeti gibi genel bilgileri içeren bir doküman.

Uygulama: Ekip üyelerinin projenin genel yapısını anlamalarına yardımcı olacak detaylı bir genel bakış dokümanı hazırlanmalıdır. Şuan yaptığımız bu rapor gibi ön hazırlama dokümantasyonu oyun projeleri için çok önemlidir.

### **□ Teknik Dokümantasyon:**

Açıklama: Oyunun teknik yapısı, kullanılan oyun motoru, programlama dilleri, algoritmalar ve oyun içi sistemlerin detaylı bir açıklamasını içeren dokümantasyon.

Uygulama: Kod içerikleri, sınıflar, fonksiyonlar ve kullanılan teknolojilerin belgelenmesi gerekmektedir.

### **□ Oyun Tasarım Dokümanları:**

Açıklama: Oyunun genel tasarımı, seviye tasarımları, karakter tasarımları, oyun içi etkinlikler ve kullanıcı ara yüzü tasarımını içeren belgeler.

Uygulama: Seviye tasarım dokümanları, karakter dosyaları, oyun içi etkinlik akışları ve kullanıcı ara yüzü prototipleri, use case diyagramları gibi birçok dokümanlar önceden hazırlanmalıdır.

### **□ Test ve Kalite Güvence Dokümantasyonu:**

Açıklama: Test senaryoları, hata raporları, performans test sonuçları ve güvenlik testleri için belgeler.

Uygulama: Hata takip sistemleri, test planları ve kalite güvence standartlarına uygun belgeler yapılmalı ve oyun içindeki hatalar direk giderilmelidir.

### **□ Proje Takip ve İlerleme Raporları:**

Açıklama: Projenin ilerleme durumu, hedeflere ne kadar yaklaşıldığı, belirli aşamalarındaki gelişmeleri takip eden raporlar.

Uygulama: Haftalık veya aylık raporlar, ilerleme gösteren grafikler ve takvimler yapılmalıdır yani projenin ağ diyagramı çıkartılmalıdır bizim rapor 1 de yaptığımız gibi.

#### □ **Kullanıcı El Kitabı:**

Açıklama: Oyunu oynayanlar için kullanıcı dostu bir el kitabı veya rehber, oyunun nasıl oynanacağı, kontroller ve ipuçları içerir.

Uygulama: Oyunun içinde veya ayrı bir belge olarak kullanıcıya sunulan detaylı bir el kitabı olmalıdır. Yapılan oyunda bir hikayemiz olduğu için bu detaylıca kullanıcıya aktılmalıdır.

#### **Neden İyi Dokümantasyon Önemlidir:**

Ekip İçi İletişim ve İşbirliği: İyi dokümantasyon, ekip üyeleri arasında bilgi paylaşımını artırır ve işbirliği için sağlam bir temel oluşturur.

Proje Süresince Sorunların Çözümü: Dokümantasyon, proje sırasında karşılaşılan sorunların çözümüne yönelik rehberlik sağlar.

Proje Devralma ve Gelecekteki Geliştirmeler: İyi dokümantasyon, projenin gelecekteki geliştirmeler veya yeni ekibin projeyi devralması için temel oluşturur.

Proje Anlayışını Artırma: Dokümantasyon, projenin genel anlayışını artırır ve herkesin projenin büyük resmini görmesine yardımcı olur.

### **WELL-DEFINED INTERFACES (İYİ TANIMLANMIŞ ARAYÜZLER) :**

"Well-defined interfaces" (İyi Tanımlanmış Arayüzler), bir oyun projesi için önemli bir tasarım hedefidir. İyi tanımlanmış arayüzler, oyun içindeki bileşenlerin (modüllerin, sistemlerin, özelliklerin) birbirleriyle etkileşimini düzenler ve ekip içinde daha düzenli bir işbirliği sağlar. İşte bu hedefi projenize uygun şekilde detaylandırmak için bazı açıklamalar:

#### □ **Modül Arayüzleri:**

Açıklama: Oyun içindeki farklı modüller (grafik motoru, ses motoru, oyun mekaniği modülü, vs.) arasındaki etkileşimleri tanımlayan açık ve anlaşılır arayüzler.

Uygulama: Modül arayüzleri, her modülün diğerleriyle nasıl iletişim kurduğunu belirten belgeler ve protokoller olmalıdır component diyagramında anlatıldığı gibi detaylıca açıklanmalıdır.

#### □ **Kullanıcı Arayüzü Arayüzleri:**

Açıklama: Oyunun kullanıcı arayüzü (UI) bileşenleri arasındaki etkileşimleri ve veri akışını tanımlayan arayüzler.

Uygulama: Oyun menüleri, ekranlar ve kullanıcı etkileşimleri arasındaki arayüz belgeleri, kullanıcının başlangıç menüsü ile etkileşimi herşeyi anlaşılır ve akıcı olmalıdır.

#### □ **Oyun Mekaniği Arayüzleri:**

Açıklama: Oyunun temel mekaniği, karakter kontrolü, düşman yapısı ve diğer oyun özellikleri arasındaki etkileşimleri tanımlayan arayüzler.

Uygulama: Karakter kontrolü, çarpışma algılama, oyun içi etkileşimler gibi oyun mekaniği arayüz belgeleri. Karakterimiz Eva'nın her düşmanla karşılaşması, kontrolleri hepsi tanımlanmalıdır.

#### □ **Harita ve Seviye Tasarımı Arayüzleri:**

Açıklama: Oyun haritası, seviye tasarımı ve oyun içindeki farklı seviyeler arasındaki geçişleri tanımlayan arayüzler.

Uygulama: Harita formatları, seviye tasarım protokolleri, seviye yükleme ve geçiş arayüzleri.

## **Neden İyi Tanımlanmış Arayüzler Önemlidir:**

Ekip İçi İletişim ve İşbirliği: Herkesin aynı arayüzleri kullanması, ekip içinde daha düzenli bir işbirliği sağlar.

Modüler Geliştirme: Modüler yapılar, bağımsız olarak geliştirilebilir ve test edilebilir. İyi tanımlanmış arayüzler, modüler geliştirmeyi destekler.

Hata Ayıklama ve Bakım: İyi tanımlanmış arayüzler, hata ayıklama sürecini kolaylaştırır ve bakımı daha etkili hale getirir.

Gelecekteki Geliştirmeler: Oyunumuzun genişletilmesi veya yeniden tasarlanması durumunda, iyi tanımlanmış arayüzler gelecekteki geliştirmeleri destekler.

## **USER-FRIENDLINESS (KULLANICI DOSTU) :**

"User-friendliness" (Kullanıcı Dostu) bir oyun projesi için son derece önemli bir tasarım hedefidir. Bu hedef, oyunumuzun kullanıcılarının deneyimini olumlu yönde etkilemeyi amaçlar ve oyunu daha çekici ve erişilebilir hale getirir. İşte bu hedefi projenize uygun şekilde detaylandırmak için bazı açıklamalar:

### **□ Kolay Anlaşılır Kullanıcı Arayüzü:**

Açıklama: Oyunumuzun menüleri, kontrolleri ve diğer kullanıcı arayüzü elemanları, oyuncuların hızlı bir şekilde anlamalarını sağlamak üzere basit ve berrak bir şekilde tasarlanmalıdır.

Uygulama: Temiz ve düzenli menü düzenleri, basitleştirilmiş kontroller, anlaşılır simgeler, oyunun hikayesi karışık olduğundan dolayı daha fazla simge ve kontrol anlatımı olmalıdır.

### **□ Kolay Kontroller:**

Açıklama: Oyun kontrolleri, oyuncuların oyunu etkili bir şekilde yönetmelerini kolaylaştırmak için basit ve duyarlı olmalıdır.

Uygulama: Kolayca öğrenilebilen temel kontroller, özelleştirilebilir kontrol seçenekleri.

### **□ Eğitim ve Rehberlik:**

Açıklama: Oyunculara oyunun temel özellikleri, stratejiler ve mekaniği konusunda açık ve kullanıcı dostu rehberlik sağlamak.

Uygulama: İlk oyun başlangıcında interaktif eğitimler, kullanıcı dostu ipuçları ve bilgi ekranları. Eğer oyuncuya hikâyeyi tam olarak anlatmazsak bir şekilde eksiklik olur.

### **□ Geribildirim ve İletişim:**

Açıklama: Oyunculara oyun içindeki başarıları, hataları ve gelişmeleri bildirmek için kullanıcı dostu geribildirim mekanizmaları.

Uygulama: Oyun içi başarı ekranları, açıklayıcı hata mesajları, kullanıcı geri bildirim formları. Kullanıcının oyun hakkında geri bildirimini dikkate almamız önemlidir.

### **□ Bariz ve İntuitif Oyun Mekaniği:**

Açıklama: Oyun içindeki mekanikler, oyuncuların doğal bir şekilde anlamalarını sağlamak üzere bariz ve sezgisel olmalıdır.

Uygulama: Oyun içi eylemler ve tepkiler arasındaki tutarlılık, sezgisel oyun mekanikleri.

## **Neden Kullanıcı Dostuluğu Önemlidir:**

Geniş Kitle Erişimi: Kullanıcı dostu bir oyun, geniş bir oyuncu kitlesine hitap edebilir.

Oyuncu Memnuniyeti: Kolay anlaşılır ve kullanımı basit bir oyun, oyuncuların memnuniyetini artırabilir.

Başlangıçta Oyun İlgisi: Kullanıcı dostu bir oyun, oyuncuların hızlı bir şekilde oyunu sevmelerini ve başlamalarını sağlar.

Rekabet avantajı: Kullanıcı dostu bir tasarım, diğer oyunlara göre rekabet avantajı sağlar.

## **RAPİD DEVELOPMENT (HIZLI GELİŞİM) :**

"Rapid development" (Hızlı Gelişim), bir oyun projesi için önemli bir tasarım hedefi olabilir, ancak her proje farklı olduğu için bu hedefin projenize uygun olup olmadığını değerlendirmeniz önemlidir. İşte bu hedefi projenize uygun şekilde detaylandırmak için bazı açıklamalar:

### **□ Kısa Geliştirme Süreleri:**

Açıklama: Oyunun prototip aşamalarının hızlı bir şekilde geliştirilmesi, değişikliklere hızlı cevap verme ve yeni özellikleri hızla entegre etme yeteneği.

Uygulama: Hızlı prototipleme araçları kullanma, basit özellikleri kısa sürede test etme gibi özellikler oyunların daha hızlı güncellenmesine yardımcı olur.

### **□ Esneklik ve İteratif Gelişim:**

Açıklama: Projenin geliştirme sürecinin esnek ve iteratif olması, tasarımdaki değişikliklere ve geliştirmelere hızlı bir şekilde yanıt verebilme yeteneği.

Uygulama: Sık sık geri bildirim almak, prototipleri hızlı bir şekilde revize etmemiz gerekir.

### **□ Minimum Viable Product (MVP) Odaklı Gelişim:**

Açıklama: Projenin temel işlevselliğine odaklanma ve bu temel işlevselliği hızlı bir şekilde sunma.

Uygulama: Ana oyun mekaniğine odaklanma, gereksinimler listesini MVP ilkelerine göre belirlenmelidir.

## **Neden Hızlı Gelişim Önemlidir:**

Piyasa Rekabeti: Hızlı gelişim, oyununuzu rakiplerinizden önce piyasaya sürmenize ve trendlere hızlı bir şekilde uyum sağlamanıza yardımcı olabilir.

Kullanıcı Geri Bildirimi: Hızlı gelişim, kullanıcı geri bildirimlerine daha hızlı yanıt vermenizi sağlar, bu da oyununuzu kullanıcı beklentilerine daha iyi uyarlamak anlamına gelir.

Değişen Gereksinimlere Uyum: Oyun projelerinde gereksinimler sıkça değişebilir. Hızlı gelişim, bu değişikliklere daha kolay adapte olmanıza yardımcı olabilir.

Motivasyon ve İlerleme: Hızlı gelişim, ekip üyelerinin motivasyonunu artırabilir çünkü daha sıkı çalışmanın ve başarıların hemen görülmesinin bir sonucudur.

## **EASE OF LEARNING (ÖĞRENME KOLAYLIĞI):**

"Ease of learning" (Öğrenme Kolaylığı), Bu tasarım hedefi, genellikle geniş bir oyuncu kitlesiyle etkileşimde bulunan oyun projeleri için büyük önem taşır. Oyuncuların oyunu hızlıca kavramaları ve keyif alabilmeleri, projenizin başarısını olumlu yönde etkileyebilir.

Bu tasarım hedefini projenize uygun şekilde detaylandırmak için aşağıdaki açıklamaları inceleyebilirsiniz:

### **□ Kullanıcı Dostu Kontroller:**

Açıklama: Oyunun temel kontrollerinin basit, anlaşılır ve kullanıcı dostu olması.

Uygulama: Ana kontrollerin oyun içinde doğal ve sezgisel bir şekilde kullanılması, oyuncuların oyun mekaniğini hızlıca öğrenmelerine yardımcı olur.

### **□ Eğitim Seviyeleri:**

Açıklama: Oyun içinde kullanıcıya temel mekaniği öğreten eğitim seviyelerinin bulunması.

Uygulama: Oyunculara oyun içinde interaktif eğitimler veya rehberlik sunarak temel konseptleri öğretmek.

### **□ Görsel İşaretler ve İpuçlar:**

Açıklama: Oyunculara yönlendirmeler için görsel işaretler, ipuçları ve animasyonların kullanılması.

Uygulama: Oyuncuların dikkatini çekmek ve önemli noktalara odaklanmalarını sağlamak için görsel unsurların etkili bir şekilde kullanılması, karakterimizin çevreyi keşfedip ipuçlarını toplamamız gerekir.

### **□ Oyun İçi Yardım Menüsü:**

Açıklama: Kullanıcıların oyun içinde yardım alabilecekleri bir menünün bulunması.

Uygulama: Oyunculara oyun içinde herhangi bir aşamada yardım ve açıklamalar sunan bir menü sistemi.

## **Neden Ease of Learning Önemlidir:**

Yeni Oyuncular İçin Çekici: Oyunun hızlıca anlaşılabilir olması, yeni oyuncuları cezbetmek için önemlidir.

Kullanıcı Deneyimi: Kolay öğrenilen oyunlar, kullanıcı deneyimini artırır ve oyuncuların oyunu daha uzun süre oynamasını sağlar.

Oyuncu Memnuniyeti: Oyunun öğrenme süreci, oyuncuların memnuniyetini artırır ve olumlu bir oyun deneyimi sağlar.

## **READABILITY (OKUNABİLİRLİK):**

Bu tasarım hedefi, oyun içindeki her türlü bilgi ve içeriğin oyuncular tarafından rahatça okunabilir olmasını hedefler. Oyuncuların oyun dünyasıyla etkileşimde bulunurken bilgileri anlamalarını ve bu bilgileri doğru bir şekilde kullanmalarını sağlamak, oyunun genel başarısını artırabilir.



#### □ **Metin ve Bilgi Düzeni:**

Açıklama: Oyun içindeki metinlerin düzenli ve anlaşılır bir şekilde sunulması.

Uygulama: Oyun içindeki bilgilerin, menülerin ve diğer metin tabanlı içeriklerin düzenli bir şekilde yerleştirilmesi ve okunabilir olması.

#### □ **Görsel Temsil:**

Açıklama: Oyun içindeki görsel unsurların, ikonların ve simgelerin anlamlarının açık olması.

Uygulama: Oyuncuların oyun içindeki nesneleri, karakterleri ve diğer önemli öğeleri kolayca tanıyabilmeleri için net ve açık görsel temsillerin kullanılması.

#### □ **Menü Navigasyonu:**

Açıklama: Oyun içindeki menülerin basit, doğrudan ve kullanıcı dostu bir navigasyona sahip olması.

Uygulama: Oyuncuların oyun içinde hızlıca dolaşabilmeleri için menülerin anlaşılır bir sıralama ve yapıya sahip olması.

#### □ **Oyun İçi İpuçları:**

Açıklama: Oyunculara oyun içinde doğrudan veya dolaylı yoldan bilgi veren okunabilir ipuçlarının kullanılması.

Uygulama: Oyuncuların oyun mekaniği, hikaye veya diğer önemli unsurlar hakkında daha fazla bilgi alabilmeleri için okunabilir ipuçlarının stratejik bir şekilde yerleştirilmesi.

### **Neden Readability Önemlidir:**

Kullanıcı Deneyimi: Oyun içindeki bilgilerin anlaşılır bir şekilde sunulması, oyuncuların deneyimini olumlu yönde etkiler.

Yönlendirme: Okunabilir bir tasarım, oyuncuları doğru yönlendirebilir ve hikaye, görev veya bulmacaları daha iyi anlamalarına yardımcı olabilir.

Erişilebilirlik: Oyunun geniş bir oyuncu kitlesi tarafından oynanabilir olması için bilgilerin okunabilir ve anlaşılır olması önemlidir.

Karmaşıklık Azaltma: Readability, oyun içindeki karmaşıklığı azaltabilir ve oyuncuların oyunu daha rahat anlamalarını sağlayabilir.

### **EASE OF REMEMBERİNG (HATIRLANABİLİRLİK):**

Bu tasarım hedefi, oyuncuların oyun içindeki önemli bilgileri hatırlamalarını ve bu bilgileri kullanarak oyun dünyasında etkili bir şekilde ilerlemelerini sağlamayı hedefler. Hatırlanabilir bir oyun, oyuncuların oyun deneyimini daha etkileyici ve ödüllendirici hale getirebilir.

#### □ **Oyun İçi Bilgilerin Tutarlılığı:**

Açıklama: Oyun içinde sunulan bilgilerin tutarlı bir şekilde yer alması ve oyuncuların önceki bilgileri hatırlamalarını kolaylaştırması.

Uygulama: Hikaye öğeleri, görevler ve karakterlerle ilgili bilgilerin oyun boyunca tutarlı bir biçimde sunulması, oyunların verilen hikayesinde herşey net açıklanmalı ki tutarlı olsun oyuncu gözünde.

#### □ **Oyuncu İlerlemesinin İzlenmesi:**

Açıklama: Oyuncuların kendi ilerlemelerini ve başarılarını hatırlamalarını sağlayacak bir sistem.

Uygulama: Oyuncuların kazandıkları ödüller, tamamladıkları görevler ve elde ettikleri yeteneklerin bir takip sisteminden geçirilmesi, oyuncu her level yükseldiğinde hangi levelde vs olduğunu görmelidir.

#### □ **Karakter ve Mekan İsimleri:**

Açıklama: Oyuncuların oyun içindeki karakterlerin, yerlerin ve önemli nesnelerin isimlerini hatırlamalarını kolaylaştırmak.

Uygulama: Oyun içindeki karakter ve mekan isimlerinin belirli aralıklarla oyunculara hatırlatılması veya bu bilgilerin oyunculara kolayca erişilebilir bir şekilde sunulması, biz karakterimize eva adını verip tüm hikayeyi anlaşılır yaptık ki kullanıcının aklında soru işareti kalmaması içindir.

#### □ **Oyun Mekaniğinin Anımsanabilirliği:**

Açıklama: Oyun içindeki temel mekaniklerin oyuncular tarafından hızlı bir şekilde hatırlanabilir olması.

Uygulama: Oyunculara oyun mekaniğiyle ilgili önemli ipuçları ve hatırlatıcılar sunulması, tutorial seviyelerinin etkili bir şekilde tasarlanması.

#### □ **Hikaye Akışının Kolay Hatırlanabilirliği:**

Açıklama: Oyunun hikayesinin ana hatlarıyla oyuncular tarafından hatırlanabilir olması.

Uygulama: Oyuncuların kararlarının hikaye üzerindeki etkilerinin vurgulanması ve önemli olayların görsel ve işitsel olarak öne çıkarılması.

### **Neden Ease of Remembering Önemlidir:**

Kullanıcı Katılımı: Oyuncuların oyun içindeki önemli bilgileri hatırlamaları, onların daha fazla katılım göstermelerine ve oyun dünyasına daha derinlemesine bağlanmalarına yardımcı olabilir.

Oyun Zevki: Kolay hatırlanabilirlik, oyunun keyifli olmasına katkıda bulunabilir çünkü oyuncular olayları ve karakterleri daha iyi takip edebilir.

Zorluk Seviyesi: Oyuncuların oyun içindeki zorlukları aşmaları ve hedeflere ulaşmaları için bilgileri hatırlamaları önemlidir.

### **EASE OF USE (KULLANIM KOLAYLIĞI):**

Bu tasarım hedefi, kullanıcı deneyimini ön planda tutarak oyununuzu daha çekici ve erişilebilir hale getirmeyi amaçlar. Kullanıcıların oyununuzu kolayca anlamaları ve keyif almaları, başarılı bir oyun deneyimi için kritik öneme sahiptir.

#### □ **Kullanıcı Arayüzü Tasarımı:**

Açıklama: Oyun içindeki menüler, düğmeler ve kontrollerin sezgisel bir şekilde tasarlanması.

Uygulama: Ana menü, alt menüler ve oyun içi kontrollerin kullanıcıların beklediği şekilde düzenlenmesi. Kontrollerin ve butonların açık ve anlaşılır simgelerle desteklenmesi.

#### □ **Tutorial ve Eğitim Seviyeleri:**

Açıklama: Oyun başında veya yeni bir özellik tanıtıldığında kullanıcılara interaktif eğitim seviyeleri sunulması.

Uygulama: Oyunculara oyun içindeki temel mekanikleri, kontrolleri ve stratejileri öğreten kullanıcı dostu eğitim seviyelerinin oluşturulması.

#### □ **Kolay Erişim ve Menü Yönetimi:**

Açıklama: Oyuncuların oyun içindeki bilgilere ve seçeneklere kolayca erişmelerini sağlayan bir menü yönetimi.

Uygulama: Oyuncuların envanterlerine, görevlerine ve oyun içi istatistiklere hızlıca ulaşmalarını sağlayan bir menü tasarımı.

#### **Neden Ease of Use Önemlidir:**

Geniş Kitlelere Ulaşma: Kullanımı kolay bir oyun, geniş bir oyuncu kitlesine ulaşma şansını artırabilir.

Kullanıcı Memnuniyeti: Kullanıcılar, oyunu anlamak ve oynamak konusunda zorluk yaşamadıklarında daha memnun olurlar.

Kullanıcının Odağını Sürdürme: Kullanım kolaylığı, oyuncuların oyun dünyasına odaklanmalarını ve oyunun keyfini çıkarmalarını sağlar.

#### **INCREASED PRODUCTIVITY (ARTAN ÜRETKENLİK):**

Bu tasarım hedefi, ekip içindeki iş akışını optimize etmeyi ve geliştirme süreçlerini daha verimli hale getirmeyi amaçlar. Bu, hem projenin hızlı bir şekilde ilerlemesine hem de ekibin daha tatmin edici bir çalışma deneyimi yaşamasına katkı sağlar.

#### □ **Geliştirici Araçları ve Otomasyon:**

Açıklama: Geliştiricilerin sıkça kullanılan görevleri otomatikleştiren ve geliştirme sürecini hızlandıran araçların entegre edilmesi.

Uygulama: Kod derleme, hata ayıklama, test süreçleri gibi geliştirme görevlerini kolaylaştıran otomasyon araçlarının kullanılması.

#### □ **Modüler ve Yeniden Kullanılabilir Kod Yapısı:**

Açıklama: Kodun modüler olması ve yeniden kullanılabilir öğeler içermesi, geliştiricilerin daha hızlı ve verimli çalışmalarına olanak tanır.

Uygulama: Oyun içindeki farklı öğelerin (örneğin karakterler, düşmanlar, araçlar) modüler bileşenlere ayrılması ve bu bileşenlerin yeniden kullanılabilir olması.

#### □ **Hızlı Prototipleme ve Test:**

Açıklama: Tasarımcıların ve geliştiricilerin hızlı prototipler oluşturarak oyun mekaniğini test etmelerini sağlayan süreçlerin benimsenmesi.

Uygulama: Oyun mekaniği, grafikler ve ses öğelerinin hızlı bir şekilde prototip haline getirilerek test edilmesi.

#### □ İyi Belgeleme Pratikleri:

Açıklama: Kodun ve proje süreçlerinin detaylı ve anlaşılır belgelere sahip olması, ekibin daha etkili bir şekilde çalışmasına katkı sağlar.

Uygulama: Kod belgeleri, proje süreçleri, kullanım kılavuzları gibi belgelerin düzenli ve anlaşılır olması.

#### **Neden Increased Productivity Önemlidir:**

Hızlı Geliştirme Süreci: Daha etkili bir geliştirme süreci, oyunun daha hızlı bir şekilde tamamlanmasını sağlar.

Azalan Maliyetler: Daha üretken bir ekip, proje süresince ortaya çıkabilecek maliyetleri azaltabilir.

Yenilik ve Değişikliklere Hızlı Adaptasyon: Hızlı çalışma süreçleri, yeni fikirleri veya değişiklikleri daha çabuk adapte etmeyi sağlar.

Çalışan Memnuniyeti: Geliştiricilerin ve diğer ekip üyelerinin etkili çalışma koşulları, genel memnuniyeti artırır.

#### **FLEXİBİLİTY (ESNEKLİK):**

Bu tasarım hedefi, oyun projesinin değişen gereksinimlere, teknolojik yeniliklere ve piyasa koşullarına uyum sağlamasını amaçlar. İşte bu hedefin oyun projeniz için detaylıca açıklanması:

#### □ Dinamik Geliştirme Süreci:

Açıklama: Projenin geliştirme sürecinin esnek olması, yeni gereksinimlere veya değişen koşullara hızlı bir şekilde adapte edilebilmesi.

Uygulama: Sık sık güncellenen bir geri bildirim döngüsü, esnek planlama ve sürekli iyileştirme uygulamaları.

#### □ Modüler Mimariler:

Açıklama: Projedeki bileşenlerin ve modüllerin bağımsız ve değiştirilebilir olması.

Uygulama: Oyunun farklı bölümleri arasında bağımsız modüller kullanarak, bir bölümü değiştirirken diğerlerine müdahale etmeden esneklik sağlama.

#### □ Uyumlu Oyun Motorları:

Açıklama: Esnek ve geniş bir yelpazede kullanılabilen oyun motorlarının tercih edilmesi.

Uygulama: Oyun motoru seçimi, projenin ihtiyaçlarına uygun olacak şekilde esneklik sağlayan bir temel oluşturmali.

#### □ Kolay Genişletme ve Güncelleme:

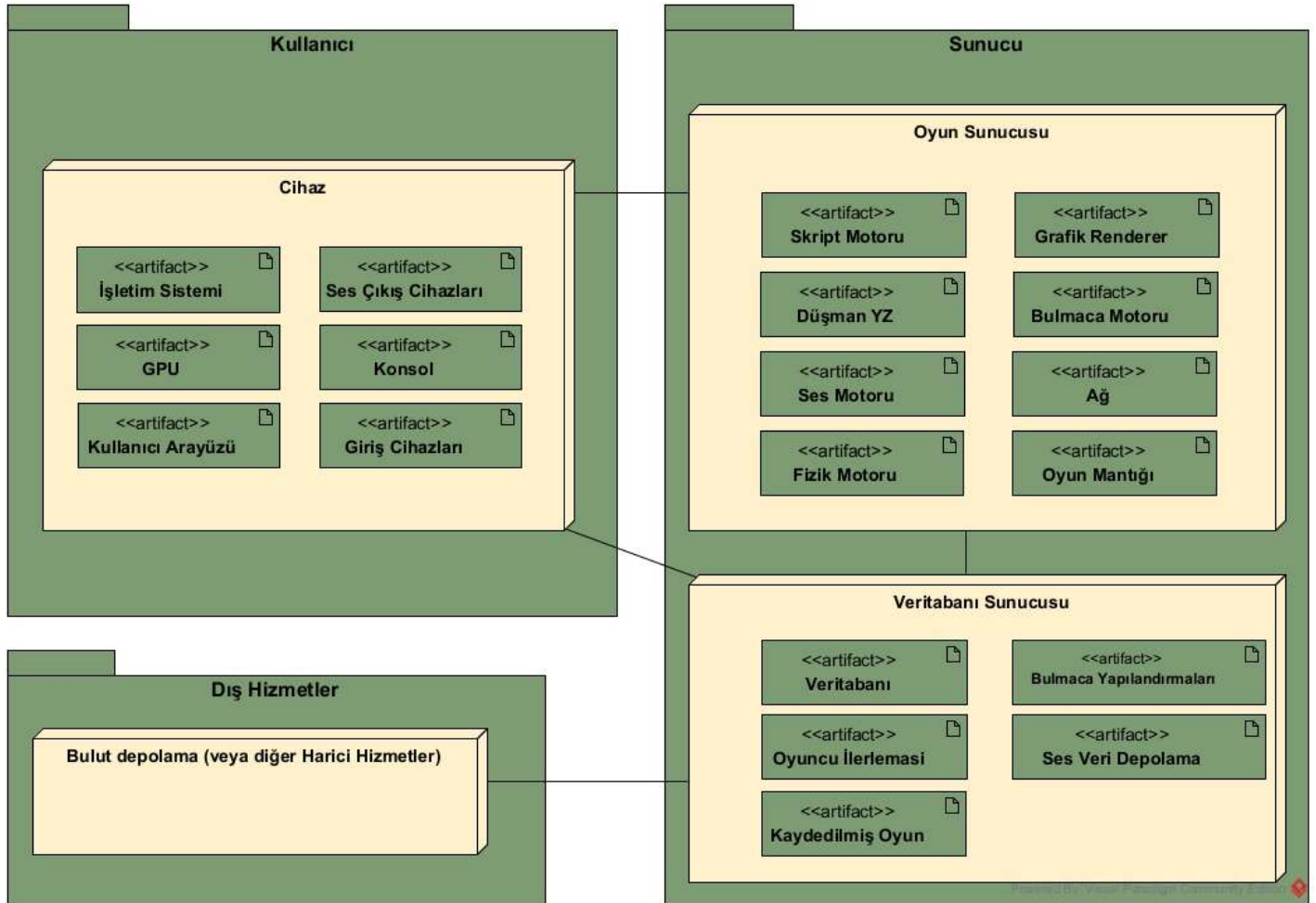
Açıklama: Oyunun yeni içerik, özellik veya güncellemelerle kolayca genişletilebilir olması.

Uygulama: Oyun içi içerik eklentileri, düzenli güncellemeler ve kullanıcı taleplerine hızlı yanıt verme.

## 9. Önerilen Yazılım Mimarisi

Çözüm önerisi, oyun projenizin temel mimarisini belirleyerek, her bir bileşenin görevlerini ve aralarındaki etkileşimi tanımlamaktadır. Oyun İstemcisi, kullanıcının bilgisayarını veya oyun konsolunu temsil eder ve grafiklerin oluşturulmasından, kullanıcı girişinin yönetilmesinden sorumlu birimleri içerir. İşletim sisteminden ses çıkış cihazlarına kadar çeşitli alt bileşenleri içerir. Oyun Sunucusu, oyun mantığını, düşman yapay zekasını, bulmaca oluşturmayı ve genel oyun durumunu yönetir; skript motoru, grafik renderer, düşman yapay zekası ve diğer bileşenleri içerir. Veri tabanı Sunucusu, oyuna ilişkin verileri depolar ve güvenlik önlemleri alınmış bir veri tabanı yönetim sistemine sahip olmalıdır. Dış hizmetler, bulut depolama gibi harici servislerle entegre olmalıdır, oyuncu verileri için yedekleme ve senkronizasyon sağlar. Bu önerilen mimari, genel bir çerçeve sunarken, projenizin özel gereksinimlerine uygun olarak adapte edilebilir.

Örneğin, çoklu oyuncu özelliklerine sahipse, ağ altyapısı daha da ölçeklendirilebilir hale getirilebilir. Her bir bileşenin detayları, proje gereksinimlerinize uyacak şekilde daha fazla ayrıntıya sahip olmalıdır.



Şekil 1 THE MIST DEPLOYMENT DIAGRAM

## A. KULLANICI:

### Kullanıcı Cihazı:

Oyuncunun bilgisayarını veya oyun konsolunu temsil eder.

Grafiklerin oluşturulmasından, kullanıcı girişinin yönetilmesinden ve kullanıcı arayüzünün görüntülenmesinden sorumlu Oyun İstemcisini içerir.

- **İşletim Sistemi:** Donanımı yöneten ve bilgisayar programları için hizmet sağlayan yazılım.
- **Ses Çıkış Cihazları:** Hoparlörler, kulaklıklar veya diğer ses çıkışı desteği.
- **GPU (Grafikler için):** Sürükleyici bir deneyim için yüksek performanslı grafik işleme.
- **Konsol:** “Konsol” terimi, video oyunları oynamak için tasarlanmış özel bir bilgisayar sistemi türü olan oyun konsolunu ifade eder. Oyun konsolları, genellikle bir televizyona veya monitöre bağlanan ve oyun kumandaları gibi kendi giriş cihazlarıyla birlikte gelen özel cihazlardır.
- **Kullanıcı Arayüzü:** Oyuncuların oyun ortamında gördüğü ve kullandığı görsel ve etkileşimli öğeler.
- **Giriş Cihazları:** Fare, klavye veya denetleyici gibi çeşitli giriş aygıtları için destek.

## B. SUNUCU:

### Oyun Sunucusu:

Oyun mantığını, düşman yapay zekasını, bulmaca oluşturmayı ve genel oyun durumunu yönetir.

Oynatıcı cihazıyla iletişim kurmak için ağ oluşturma bileşenlerini içerir.

Skript (Komut Dosyası) Motoru: Komut dosyasıyla oluşturulan olayların ve dizilerin uygulanmasına izin verir.

- **Grafik Renderer:** 2B ve 3B grafiklerin oluşturulmasını yönetir.
- **Düşman YZ (Yapay Zekası):** Düşman davranışlarını kontrol etmek için.
- **Bulmaca Motoru:** Dinamik bulmaca üretimi ve çözümü için.
- **Ses Motoru:** Sürükleyici bir deneyim için ses öğelerinin oynatılmasını kontrol eder.
- **Ağ:** Oyun istemcisi ile iletişim için TCP/IP’yi kullanır.
- **Fizik Motoru:** Oyun dünyasındaki gerçekçi fiziksel etkileşimleri simüle eder.
- **Oyun Mantığı:** Oyun mantığı ve genel yönetim için.

## C. VERİTABANI SUNUCUSU:

Oyuncu ilerlemesi, kaydedilen oyunlar, bulmaca yapılandırmaları ve düşman yapay zeka davranışları dahil olma üzere oyunla ilgili verileri depolar.

Oyun verilerini almak ve güncellemek için Oyun Sunucusu tarafından erişilir.

- **Veritabanı:** Oyunla ilgili verileri depolar.
- **Bulmaca Yapılandırmaları:** Dinamik bulmaca üretimi ve çözümü için.

- **Oyuncu İlerlemesi:** Ayrıntılı oyuncu ilerlemesini, toplanan öğeleri ve mevcut oyun durumunu saklar.
- **Ses Veri Depolama:** Oyuna ilişkin ses dosyalarının ve ilgili verilerin depolanmasını yönetir.
- **Kaydedilmiş Oyun:** Oyuncuların ilerlemelerine devam edebilmesi için kayıtlı oyun verilerini tutar.

#### **D. DIŞ HİZMETLER:**

##### **Bulut Depolama (veya diğer Harici Hizmetler):**

Oyuncu verileri için ek yedekleme ve senkronizasyon sağlar.

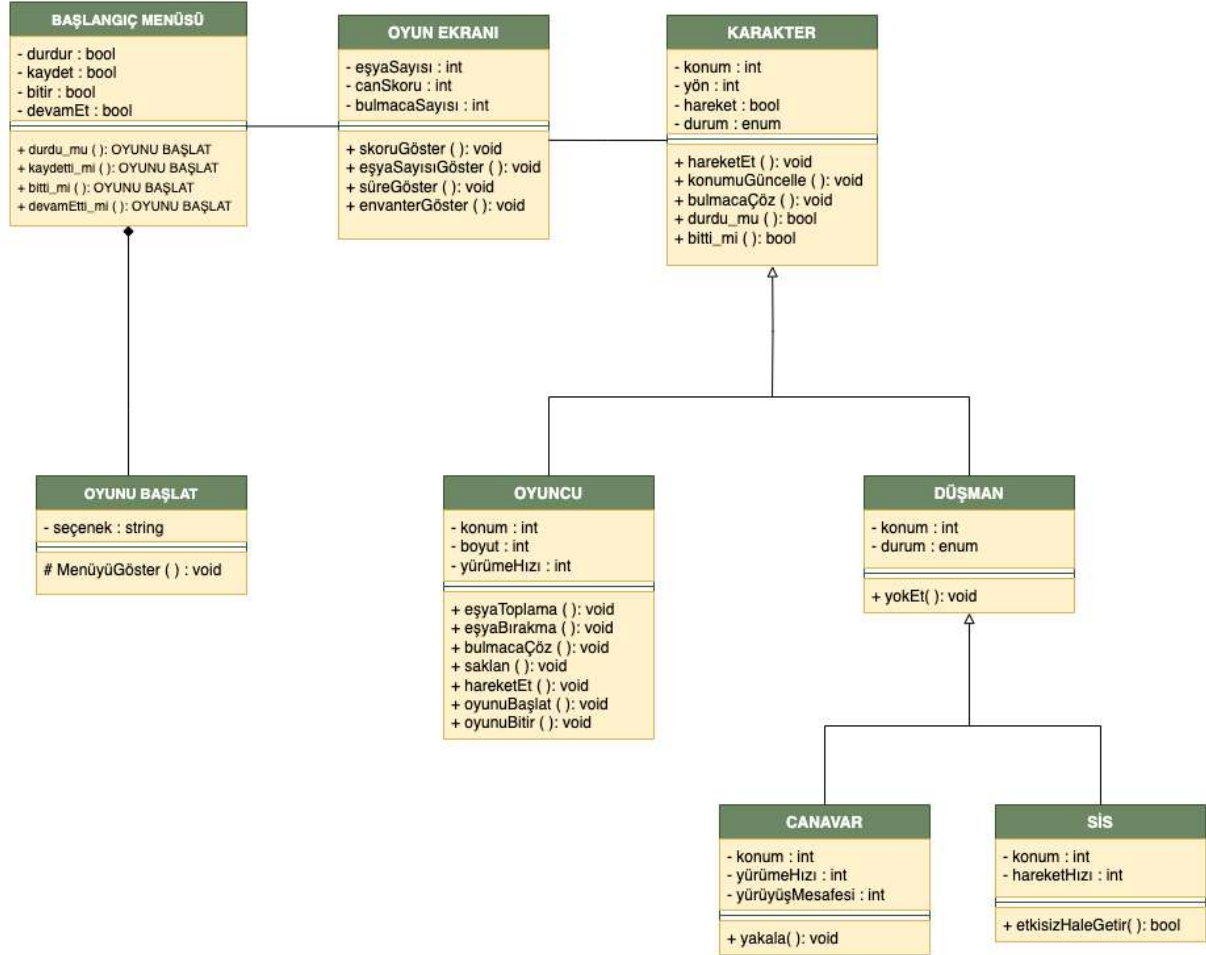
Kaydedilen oyunlar ve tercihler için bulut tabanlı depolamayı destekler.

**Diyagram visual paradigim ile çizilmiştir.**

## 10. Sınıf Diyagramları

Projenin sınıf diyagramı, oyun sistemindeki sınıfların ve bu sınıflar arasındaki ilişkilerin görsel bir temsili oluşturmak için kullanılan UML diyagramlarıdır. Sınıf diyagramları, sınıfların özelliklerini ve bu sınıfların nasıl birbiriyle iletişim kurduklarını gösterir. Ayrıca, Sınıflar arasındaki türetme ilişkisini de gösterir.

### 1. KISIM

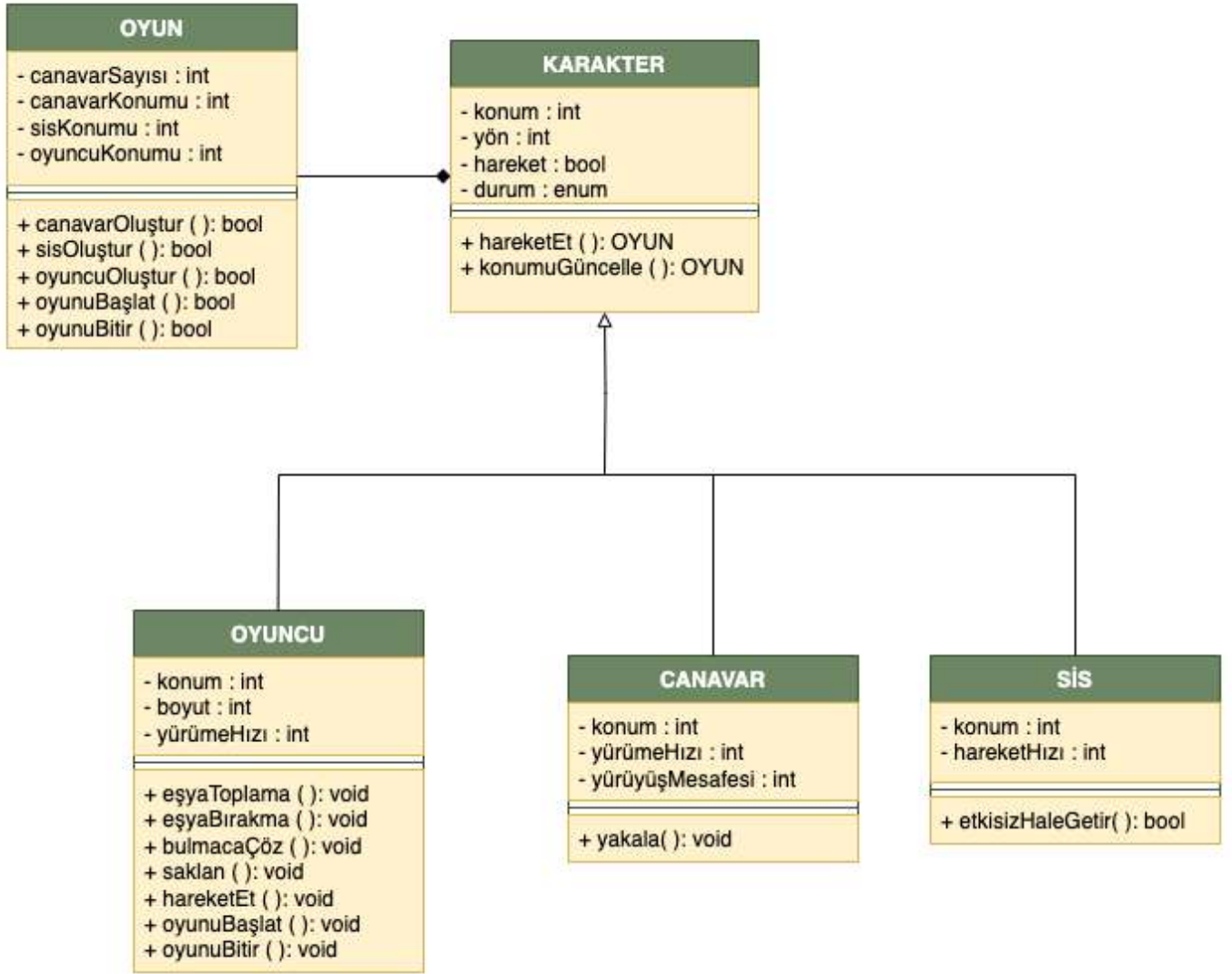


Şekil 2 Sınıf Diyagramı 1. Kısım

- Oyun sürecini gösteren bu sınıf diyagramında ilk adım olarak “Oyunu Başlat” sınıfı yer alır. Bu Sınıf menüyü gösterme metodunu içerir.
- Ardından “Başlangıç Menüsü” sınıfı yer alır. Menüde bulunan seçenekleri içerir. “Başlat” sınıfıyla arasında “Composition” ilişkisi bulunur. Yani “Başlangıç Menüsü” sınıfı “Başlat” sınıfı olmadan var olamaz, ona bağlıdır.
- “Oyun Ekranı” sınıfı ise oyun ekranında gösterilecek olan özellikleri içerir. “Başlangıç Menüsü” ile aralarında “Association” ilişkisi bulunur.
- “Karakter” sınıfı oyun içinde bulunan aktörlerin çeşitli özelliklerini ve metotlarını gösterir. “Oyun Ekranı” ile aralarında “Association” ilişkisi bulunur.
- “Karakter” sınıfı ise üst sınıf olup “Oyuncu”, “Düşman”, “Canavar”, “Sis” alt sınıflarını kapsar. Bu ilişki “Inheritance” olarak adlandırılır.
- “Düşman” sınıfı ise ayrı bir üst sınıf olarak “Canavar” ve “Sis” alt sınıflarını kapsar. “Inheritance” ilişkisi vardır.



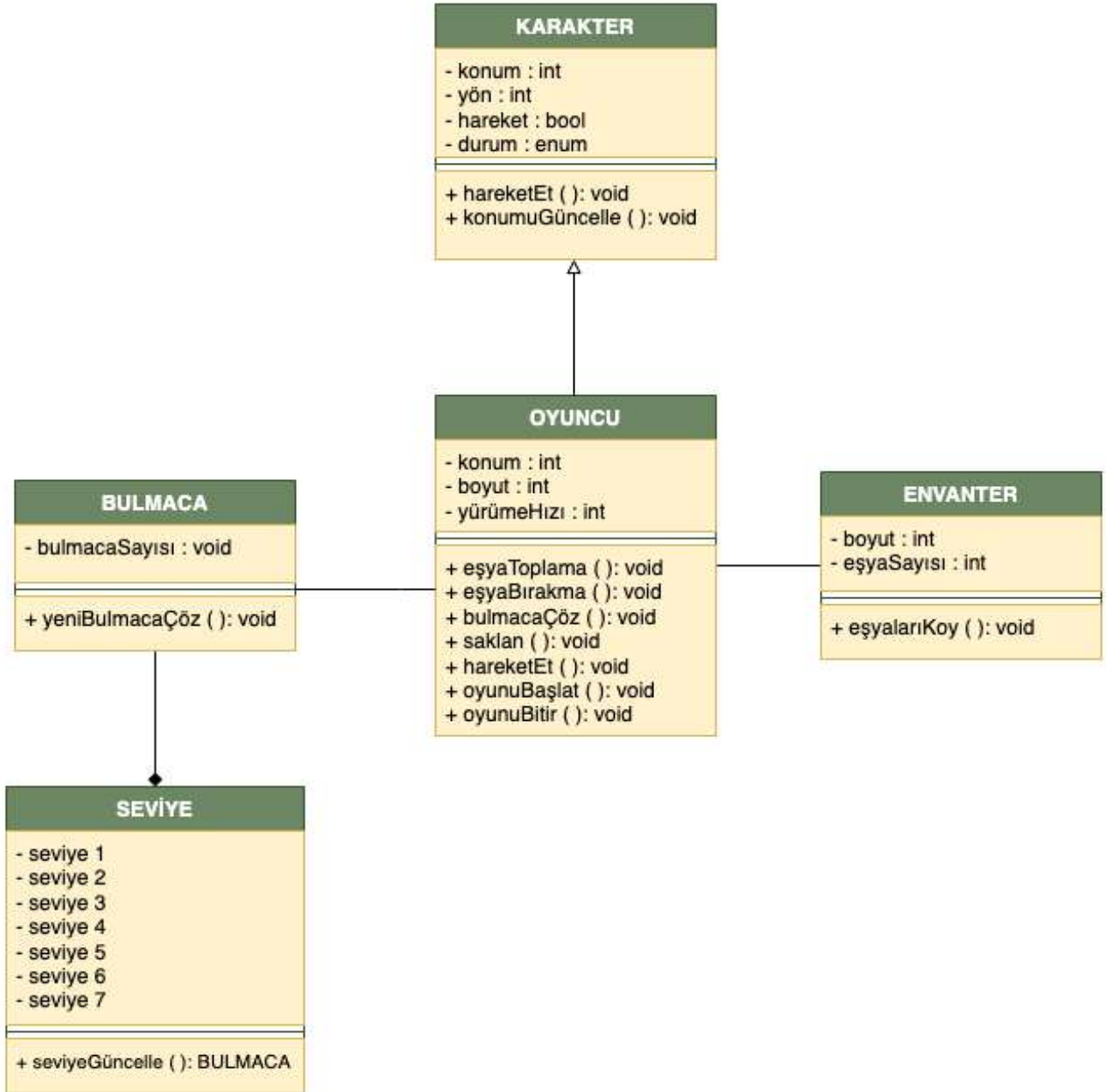
## 2. KISIM



Şekil 3 Sınıf Diyagramı 2. Kısım

- “Oyun” sınıfı ise oyun içerisinde oluşturulacak aktörleri ve özelliklerini içerir.
- “Karakter” sınıfı ile “Oyun” sınıfının arasında “Composition” ilişkisi bulunur. “Karakter” sınıfı “Oyun” sınıfına bağlıdır aksi takdirde tek başına var olamaz.
- “Karakter” sınıfı üst sınıf olup “Oyuncu”, “Canavar”, “Sis” alt sınıflarını kapsar. Bu ilişki “Inheritance” olarak adlandırılır.

### 3. KISIM



Şekil 4 Sınıf Diyagramı 3. Kısım

- “Karakter” sınıfı üst sınıf olup “Oyuncu” alt sınıfını kapsar. Bu sınıfların arasında “Inheritance” ilişkisi bulunur.
- “Envanter” sınıfı oyundaki eşyalar hakkındaki özellikleri ve metotları kapsar. “Oyuncu” sınıfıyla aralarında “Association” ilişkisi bulunur.
- “Bulmaca” sınıfı ise oyunda çözülmesi gereken bulmaca hakkındaki özellikleri ve metotları gösterir. “Oyuncu” sınıfı ile aralarında “Association” ilişkisi bulunur.
- “Seviye” sınıfı ise oyundaki levelleri ve metotlarını gösterir. Bulmacalar çözülmeden level atlanamaz bu yüzden “Bulmaca” sınıfı ile aralarında “Composition” ilişkisi bulunur.

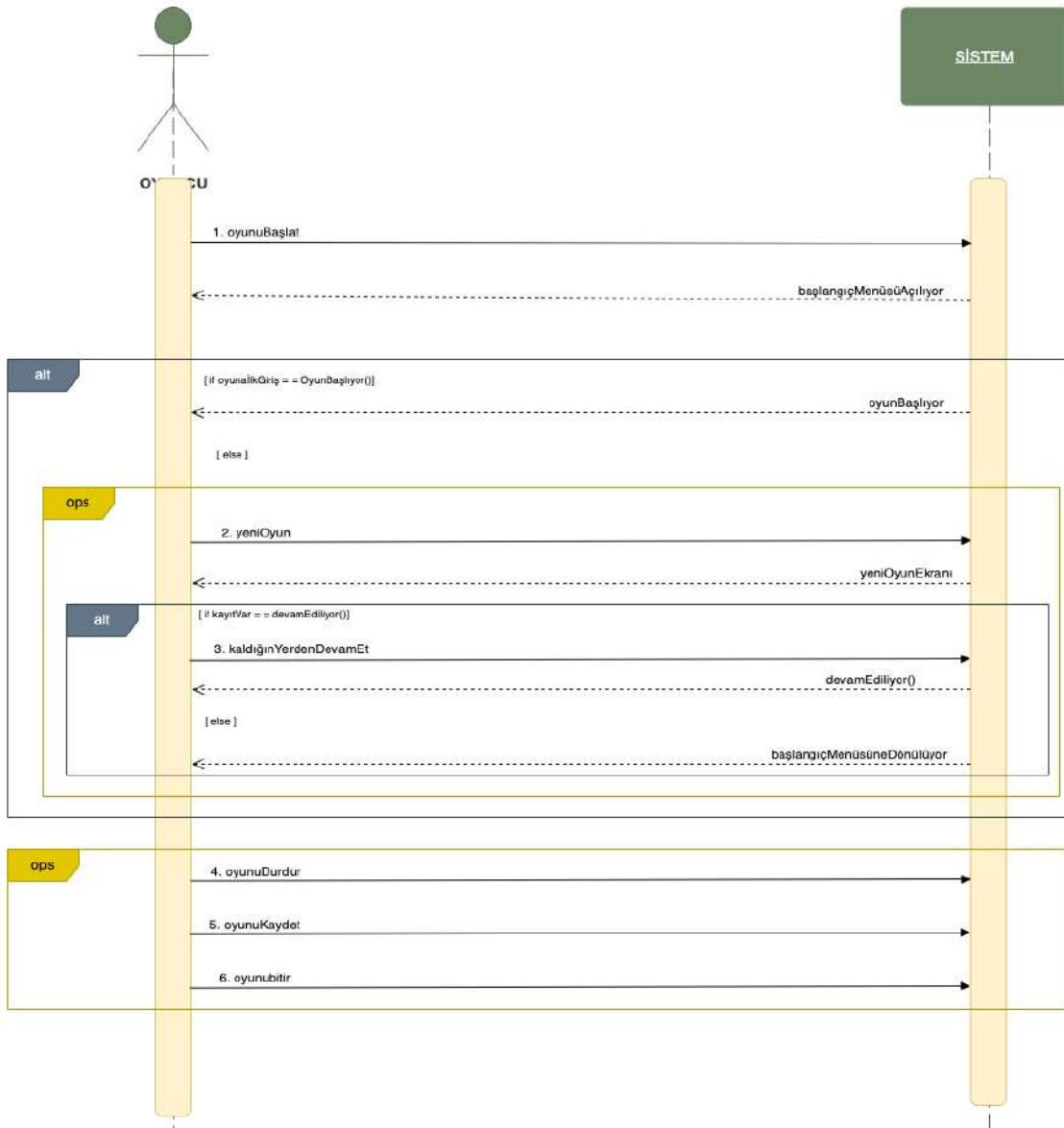
Diyagramlar draw.io ile çizilmiştir.

## 11. Dinamik Model

Projenin sequence diyagramı, oyunun temel etkileşimlerini ve bileşenler arası mesajlaşma akışını görselleştirmektedir. Oyuncunun bir hamle yapma isteği, kullanıcı ara yüzü üzerinden Oyun İstemcisine iletilir. Oyun İstemcisi, bu isteği oyun mantığında işleyerek gerekli verileri Oyun Sunucusuna iletir. Oyun Sunucusu, aldığı bu verilere göre oyunun genel durumunu günceller ve gerektiğinde Dış Hizmetler ile etkileşime geçer, örneğin bulut depolama üzerinden oyuncu verilerini günceller. Veri tabanı Sunucusu, oyun verilerini depolar ve gerektiğinde bu verilere erişim sağlar.

Oyun Sunucusu, sonuçları Oyun İstemcisine iletir, bu sonuçlar kullanıcı ara yüzü üzerinden oyuncuya gösterilir. Sequence diyagram, bu süreçleri zaman sırasında nasıl gerçekleştiklerini anlamak için bir rehber sağlar, böylece yazılım ekibi arasında bir anlayış birliği oluşturulabilir ve olası problemler önceden tespit edilebilir.

### □ BAŞLANGIÇ MENÜSÜ :

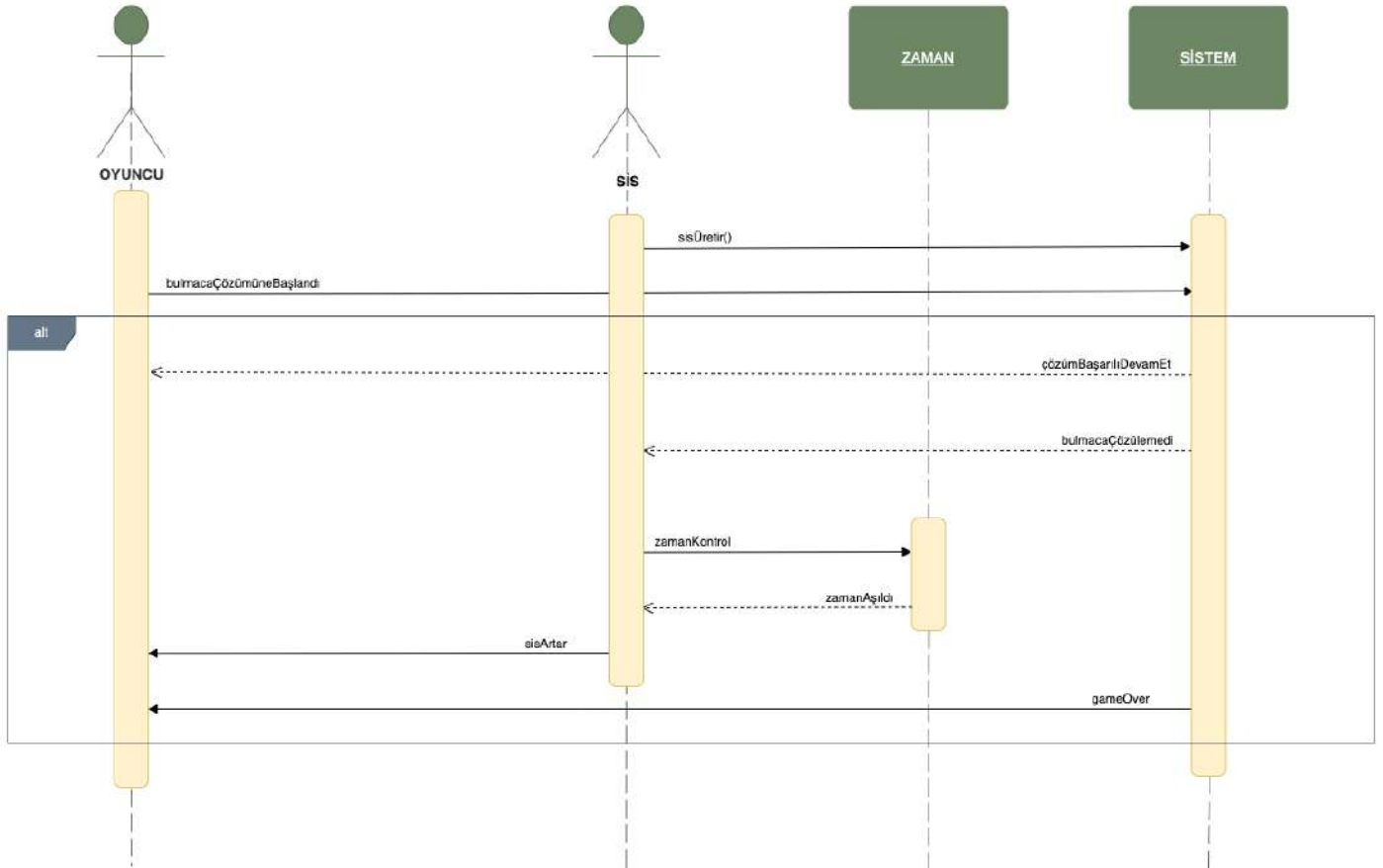


Şekil 5 Sequence Diagram Başlangıç Menüsü

- Oyuncunun oyunu başlatma sürecini temsil eden bu sequence diagramda, ilk adım olarak oyuncunun "Oyunu Başlat" butonuna tıklaması yer alır.
- Ardından, sistem geri bildirim olarak bize başlangıç menüsünü açar.
- Oyuncu, ara yüzdeki "Yeni Oyun" seçeneğini seçer ve oyun sistemi yeni oyunu başlatmak için gerekli işlemleri gerçekleştirir. Ve sistem geri bildirim olarak yeni oyun açıldı mesajını iletir
- Diğer seçeneklerde ise oyuncunun kaydedilmiş bir oyunu bulamaması durumu, oyunu kaydetme, oyunu durdurma ve bitirme senaryoları ele alınır. Örneğin, oyuncu "Kaldığın Yerden Devam Et" butonuna tıklarsa, oyun sistemi kaydedilmiş bir oyun var mı diye kontrol eder. Eğer varsa, oyuncuyu son kaldığı yerden devam ediliyor mesajı iletir
- Aksi takdirde, kaydedilmiş bir oyun bulunamazsa oyuncu başlangıç menüsüne geri döndürülür.

Ayrıca, oyuncunun oyunu başlatıp durdurma veya bitirme seçenekleri de ele alınmıştır. Örneğin, oyuncu oyunu başlattıktan sonra durdurabilir veya tamamen bitirebilir. Bu durumlar, kullanıcının oyun deneyimini yönlendirmek ve kontrol etmek için sağlanan alternatif yolları temsil eder. Bu şekilde, sequence diagram kullanıcı ile oyun sistemi arasındaki etkileşimi ve mesaj alışverişini anlatan bir bütün olarak tasarlanmış olur.

#### □ OYUNCU VE SİS ARASINDAKİ İLİŞKİ



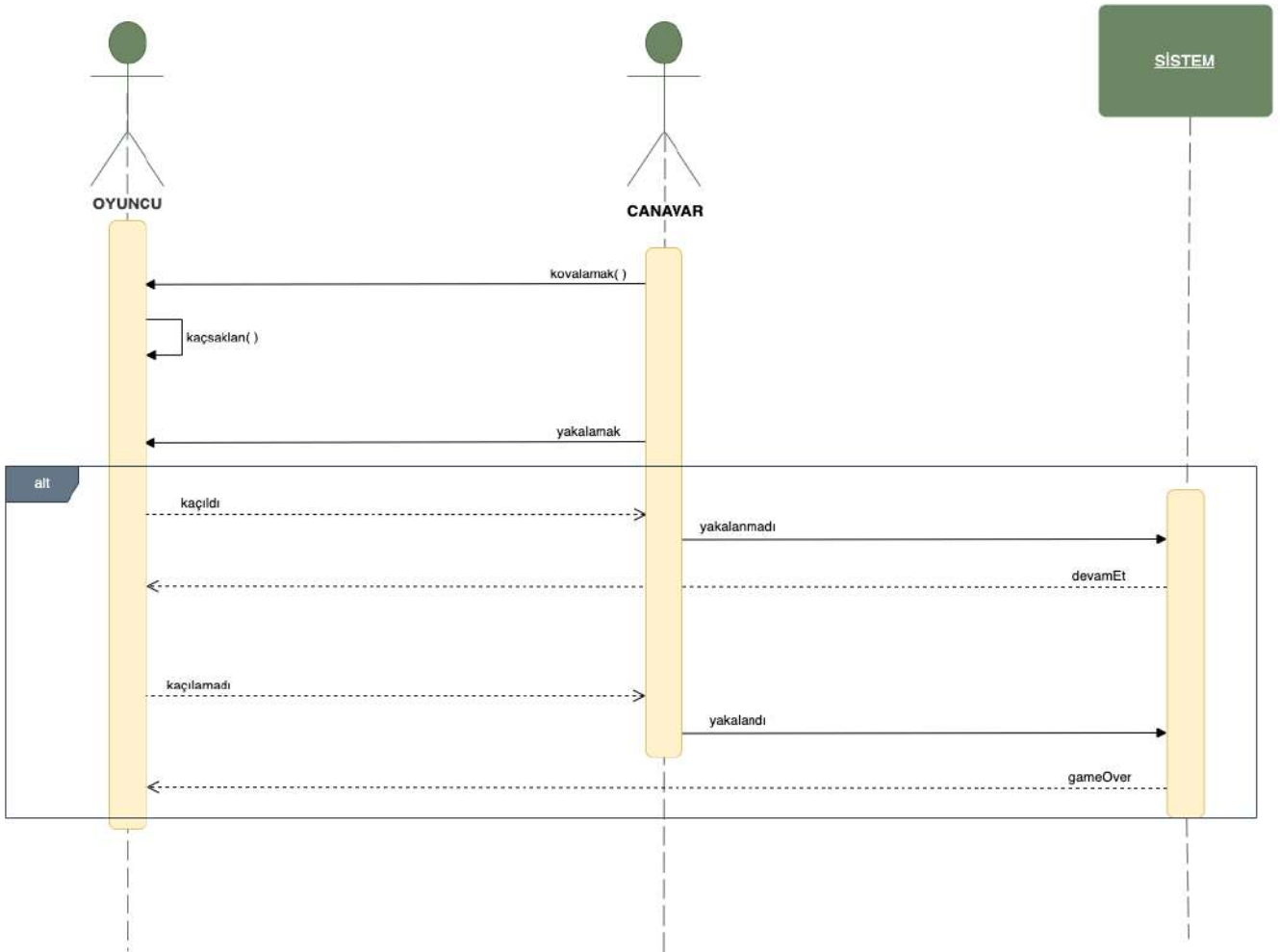
Şekil 6 Sequence Diagram Oyuncu Sis İlişkisi

Sis Aktörü ile Oyuncu Aktörü arasındaki ilişkiyi temsil eden bu sequence diyagram, oyuncunun belirli bir süre içinde bulmacayı çözmemesi durumunda ortaya çıkan senaryoyu ele alır.

- Oyuncu, oyun oynamaya devam et eylemi gerçekleştirir kendi içinde ve ardından "Bulmaca Çöz" eylemini gerçekleştirip bunu sisteme gönderir.
- Sistem geri bildirim olarak bulmacanın çözülemediğini sis aktörüne aktarır.
- Sis aktörü zaman durumuna zaman kontrolü mesajını gönderip zamanın aşılp aşılmadığını kontrol eder. Ve zaman durumundan zaman aşımı mesajını alır.
- Sis Aktörü 'nün etkisinin arttığına dair bir durum ortaya çıkar. Bu durumu oyuncu aktörüne uyarı mesajı olarak gönderir.
- Ve ardından sistemde game over mesajını iletir oyuncu aktörüne.
- Diğer bir yol ise oyuncu aktöründen sisteme bulmaca çözüldü mesajı iletilecek ve sistemden devam et mesajı anında oyuncuya iletilecektir bir sonraki görevler için.

Bu sequence diagram, oyuncunun bulmaca çözme eylemi sonucunda ortaya çıkabilecek farklı senaryoları ve bu senaryolara bağlı olarak Sis Aktörü'nün etkisinin nasıl değişebileceğini görselleştirir.

#### □ OYUNCU VE CANAVAR ARASINDA İLİŞKİ



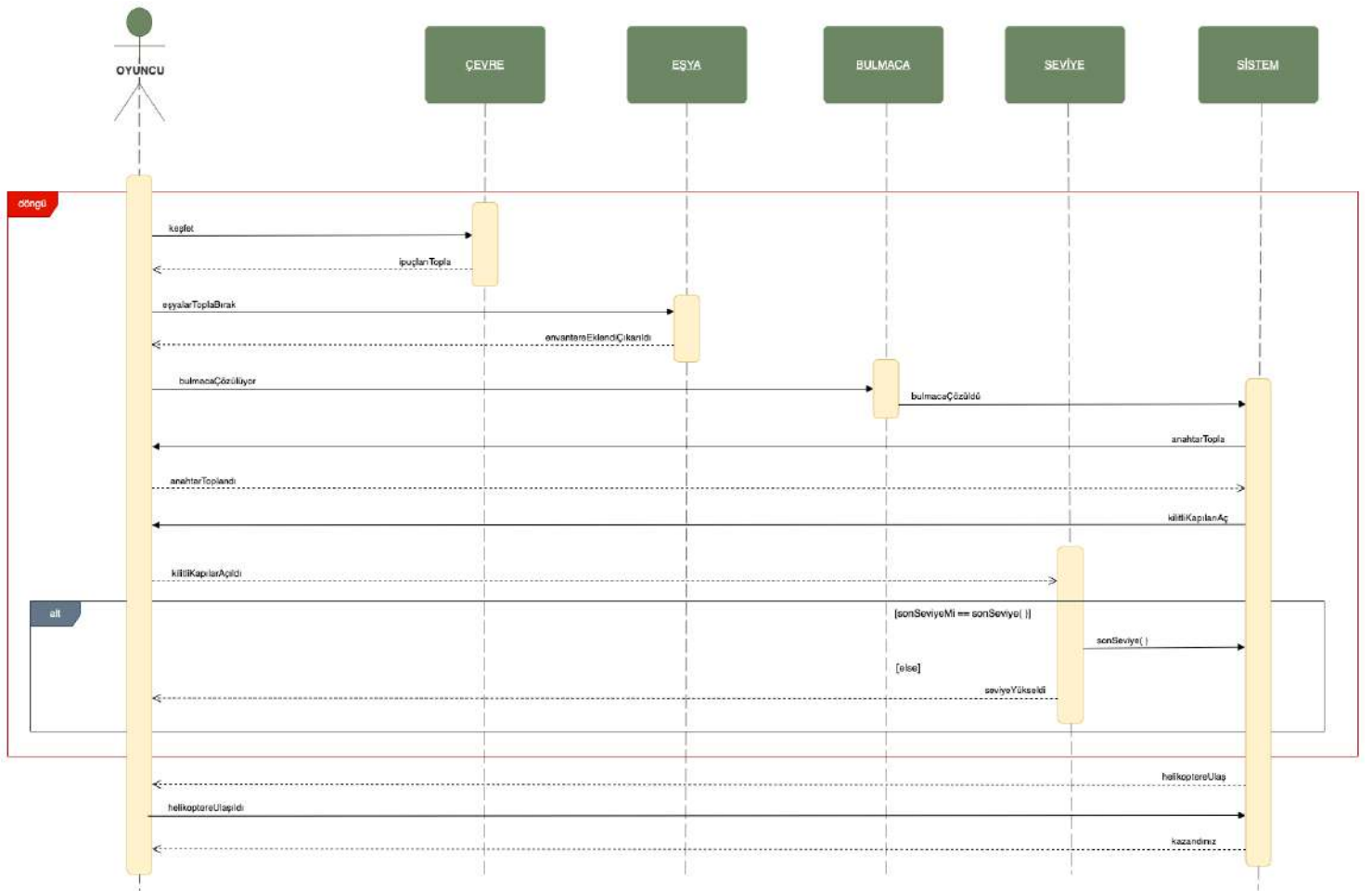
Şekil 7 Sequence Diagram Oyuncu Canavar İlişkisi

Bu sequence diagram, "Canavar Aktörünün Oyuncu Aktörü İle İlişkisi" use case'ini temsil eder ve oyuncu ile canavar arasındaki etkileşimi gösterir.

- Oyuncu, oyun oynamaya devam et eylemi gerçekleştirir kendi içinde
- Canavar kendi eylemi olarak kovalamak eylemini ara sıra gerçekleştirmektedir oyuncuya bu mesajı gönderir.
- Yakalamak aşamasına geldiğinde yakalamak mesajı oyuncuya iletilir.
- Oyuncudan yakalandı mesajı sisteme iletilir ve geri mesaj olarak game over mesajı oyuncuya iletilir
- Bir diğer ihtimal ise oyuncu kaçıp saklanırsa yakalanmadı mesajını sisteme iletilir ve sistemden geri mesaj olarak devam et diye bildirim alır.

Bu sequence diagram, canavarın oyuncuyla etkileşimini ve bu etkileşim sonucunda ortaya çıkabilecek farklı senaryoları görselleştirir. Bu senaryolar, oyuncunun başarı ve başarısızlık durumlarına göre oyunun nasıl ilerleyebileceğini açıklayan bir yapı sunar.

## □ OYUNCU İLİŞKİLERİ



Şekil 8 Sequence Diagram Oyuncu İlişkileri

Bu sequence diagram, "Oyuncunun oyun içindeki durumları" use case'ini temsil eder ve oyuncunun oyun içindeki farklı durumlarını gösterir.

- Oyuncu, oyun oynamaya devam et eylemi gerçekleştirir kendi içinde
- Ana karakterin uyanması durumu ele alınır. Önce karakter çevre durumuna keşfet mesajını gönderir geri bildirim olarak ipuçlarını topla mesajını alır.
- Sonra eşya durumuna eşyaları topla / bırak mesajını iletir. Eşya durumundan ise envantere eklendi veya çıkarıldı mesajını alır.
- Oyuncu bulmaca çöz mesajını bulmaca durumuna iletir.
- Bulmaca durumu sisteme bulmaca çözüldü mesajını iletir. Sistem geri mesaj olarak anahtarları topla iletir.
- Oyuncudan anahtar toplandı metodu sisteme gider ve sistemden kilitli kapıları aç mesaj oyuncuya gelir.
- Kilitli kapılar açıldı diye seviye durumuna mesaj iletilir. Seviyeden oyuncuya seviye yükseldi diye mesaj iletilir.
- Seviye durumu her seferinde kendi içinde son seviyede olup olmadığının kontrolünü yapar ve tüm bu durumlar döngü içerisinde son seviyeye gelinene kadar tekrarlanır.
- Eğer son seviye kontrolünde son seviyeye gelindi ise seviye durumundan sisteme son seviye mesajı iletilir. Geri bildirim olarak oyuncuya helikoptere ulaş mesajı gelir.
- Oyuncudan sisteme helikoptere ulaşıldı mesajı iletilir. Geri bildirim olarak sistemden oyuncuya kazandınız mesajı iletilir ve diyagram sonlanır.

Bu sequence diagram, oyuncunun oyun içinde farklı senaryolarına göre nasıl hareket ettiğini ve oyunun nasıl ilerlediğini görselleştirir.

**Diyagramlar draw.io ile çizilmiştir.**



## 12. Kullanıcı Arayüzü

### LOGO VE FONT



Oyunumuz için kanlı ve korkunç bir logo tasarladık sadece isminden oluşan çünkü korku oyunu olduğu için karşı tarafa kullanıcıya bunu hissettirmesini amaçladık. Font olarak them people kullandık.

### 1. KISIM



Bu kısımda oyun ekranımıza tıkladığımızda ekranda yüklenme süresi oyunun arka planda açılması için olan bir süre vardır o ekranı göreceğiz ve buraya logo ve kullanacağımız hastaneye benzer bir hastane koymayı tercih ettik bunu editleyip sis görüntüsü verim bilimsel içerikler olduğu için yeşil kanlı bir oyun olduğundan da kırmızı tonlarını kurmayı tercih ettik.

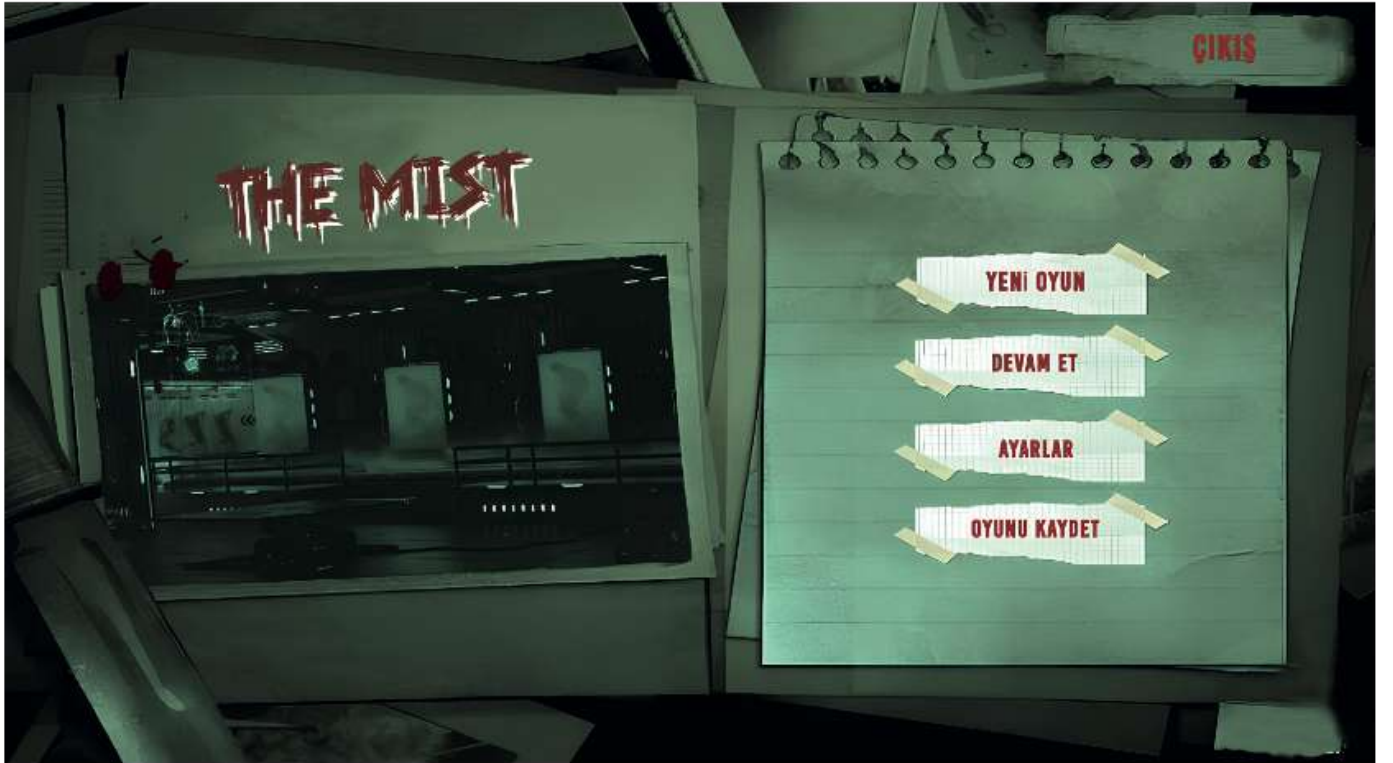


## 2. KISIM



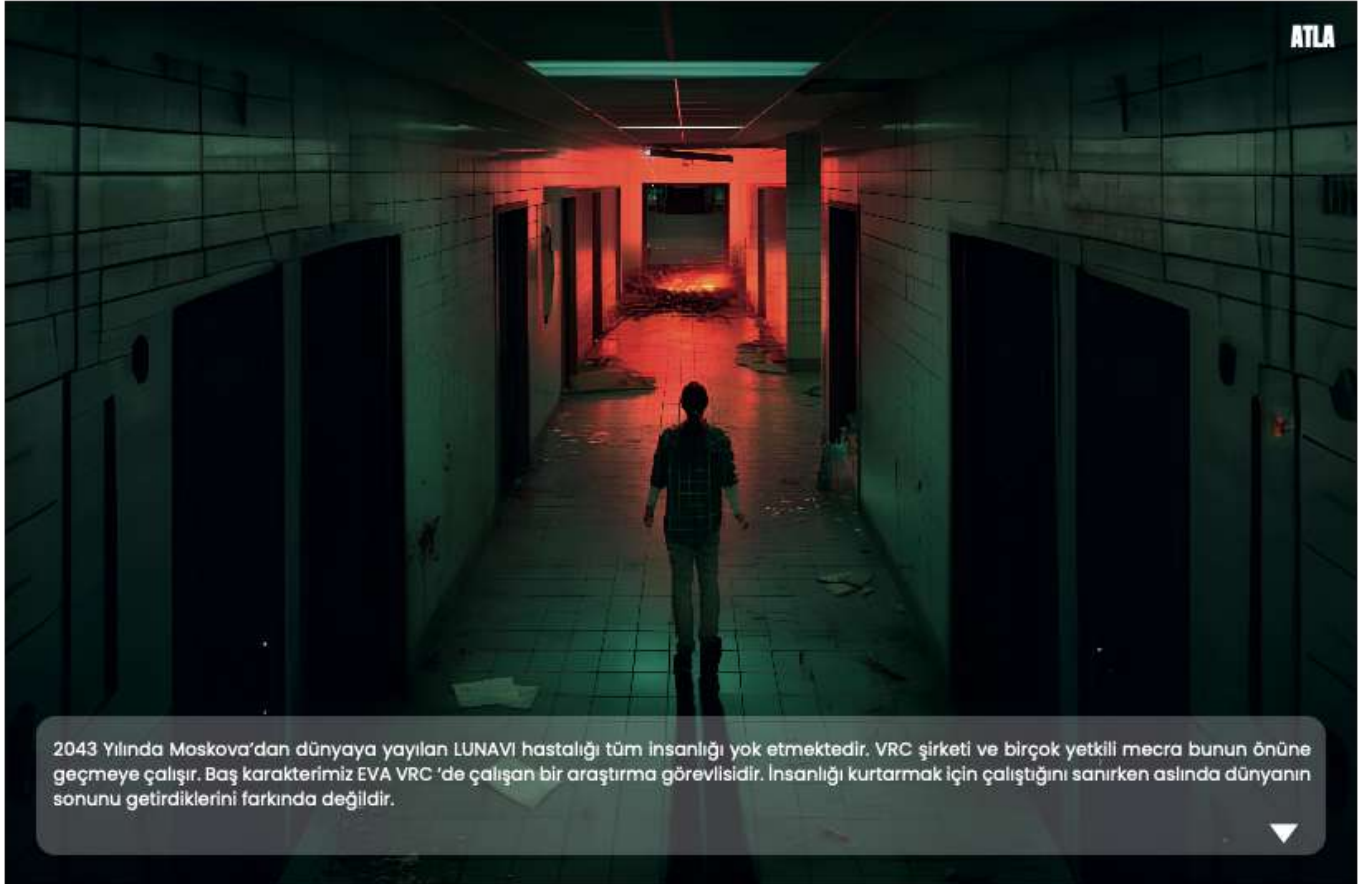
Bu kısımda yükleniyor ekranından sonra oyuncunun oyunu görmesi için oyunu başlatması gerekir bu yüzden oyunu başlat butonunu görüyoruz.

## 3. KISIM



Bu kısımda oyuncu oyunu başlat butonuna tıkladıktan sonra önüne başlangıç menüsü ekranı açılır. Başlangıç menüsü ekranımızda ipuçları ve bulmacalarla dolu bir oyun olduğundan dava dosyası gibi yapmayı planladık. Sol tarafta oyunumuzda kullandığımız canavarlarında olduğu bir laboratuvarın fotoğrafını görüyoruz onun üstünde oyunumuzun logosu ve etraf yeşil fon hâkim çünkü ana ekrandaki font ve paletlerimiz bu şekilde. Daha sonra sağ üstte oyuncunun esc bastıktan sonra direk çıkabileceği ‘ÇIKIŞ’ butonumuz var. Diğer seçenekler ise diyagramlarımızda sıklıkla bahsettiğimiz ‘YENİ OYUN’, ‘DEVAM ET’, ‘OYUNU KAYDET’ ve her oyunda olduğu gibi müzik kapatma açma vs. işlemleri için ‘AYARLAR’ butonu bulunmaktadır.

#### 4. KISIM



Bu kısımda oyuncu yeni oyuna başladığında karakterimizi başta kullandığımız hastanenin içinde görüyoruz zaten planlara göre 1. Şahıs görüşlü bir oyun olacağı için karakteri görülecek yerler kısıtlı olacak burada belirli kesitler ile hikâyenin konusunu anlatmayı planlıyoruz. Ama dinlemek veya okumak istemeyenler için sağ üstte bir atla yani direk oyuna geç butonu bulunmaktadır. Okumak isteyenler ve görmek isteyenler içinde yazıların yazılırken sağda bir ok tuşu ile okumak için kendiliğinden geçme butonu bulunmaktadır. Bu şekilde kutucukta tüm hikâye anlatılıp kullanıcıya aktarılıp oyuna başlanılacaktır.

- Oyuna dair diğer sahneler düşünülerek ve aşama aşama tasarlanacaktır.

**Kullanıcı Ara yüzler adobe illüstratör,ve photoshop ile çizilmiştir.**

## IV Test Planları

"The Mist" adlı oyun projemizin test dokümantasyonu, yazılım test sürecini yönlendirmek ve belgelemek için detaylı bir çözüm sunar. Bu doküman, IEEE 829-Test Plan Dokümantasyonu standartlarına uygun olarak yazılım testinin başarılı bir şekilde gerçekleştirilmesi için ihtiyaç duyulan adımları içerir.

### i) Neler Yapılması Gerektiği:

Test sürecinin başarılı bir şekilde yönetilebilmesi için öncelikle proje gereksinimleri ve kullanıcı beklentileri dikkate alınacak, test stratejisi ve test planı belirlenecek ve test ekipleri bu plana göre organize edilecektir. Ayrıca, testlerin otomasyonu ve manuel test süreçleri belirlenen standartlara göre uygulanacaktır.

### ii) Hangi Kalite Standardına Göre Yapılacağı:

Yazılım testleri, oyunun hedeflediği kalite standartlarına uygun olarak gerçekleştirilecektir. Bu kapsamda kullanılacak olan test araçları, yöntemler ve süreçler belirlenen kalite standartlarına uygun olacaktır.

### iii) Test İşlemlerinin Hangi Zaman Ölçeğinde Gerçekleştirileceği:

Test işlemleri, yazılım geliştirme sürecinin ilerleme aşamalarına paralel olarak planlanacak ve uygulanacaktır. Bu, geliştirme sürecinin farklı aşamalarında gerçekleştirilecek olan çeşitli test türlerini içerecektir. Özellikle, yazılımın belli başlı dönemlerdeki sürümleri üzerinde yapılan testlerin zamanlaması projenin dinamiklerine uygun olacaktır.

### iv) Test Prosesini Hedeflenenler Doğrultusunda Sağlamakla Alınan Riskler ve Bunların Nasıl Gerçekleştirileceği:

Test sürecinde karşılaşılabilecek riskler belirlenecek, bu risklere karşı önleyici adımlar alınacak ve var olan risklerle başa çıkabilmek adına acil durum planları oluşturulacaktır. Ayrıca, sürekli test izleme ve raporlama mekanizmaları kullanılarak test sürecinin sağlıklı bir şekilde ilerlediğinden emin olunacaktır.

Bu test dokümantasyonu, The Mist oyun projesinin başarılı bir şekilde teslim edilebilmesi için yazılım test sürecinin etkili bir şekilde yönetilmesine ve belgelenmesine katkıda bulunacaktır.

## 13. Test Edilebilecek ve Test Edilemeyecek Özellikler

The Mist oyunu için test edilebilecek ve test edilemeyecek özelliklerin belirlenmesi, oyunun kalitesini ve performansını değerlendirmek amacıyla kritik bir adımdır. Test edilebilecek özellikler arasında oyunun başarılı bir şekilde başlatılması, karakterin hareket kabiliyeti, görsel ve işitsel efektlerin düzgün çalışması gibi teknik özellikler yer alır.

Bununla birlikte, oyunun estetik tercihleri, kullanıcı deneyimi üzerindeki duygusal etki, ve oyuncular arasındaki rekabet gibi faktörler test edilemeyecek özelliklere örnek olarak verilebilir. Bu testler, oyunun kullanıcı dostu, sorunsuz ve tatmin edici bir deneyim sunup sunmadığını değerlendirerek oyunun genel kalitesini artırmak için önemlidir.

Tablo 13.1 Sistemin Test Edilecek Alt Sistemleri

TEST EDİLEN ALT SİSTEMLER	TEST EDİLEN SENARYOLAR
<b>BAŞLANGIÇ MENÜ SİSTEMİ</b>	Başlangıç menüsü kullanılarak oyunun başlatılması, durdurulması, kaydedilmesi, yeni oyun istenmesi gibi durumlar.
<b>OYUNCU AKTÖRÜNÜN SİSTEMİ</b>	Oyuncu karakteri Eva'nın temel mekaniklerini, Hareket, zıplama, koşma gibi temel kontrollerin doğruluğu.
<b>CANAVAR AKTÖRÜNÜN SİSTEMİ</b>	Canavar aktörünün temel komutları gerçekleştirip gerçekleştirmediği, kovalamak, yakalamak eylemlerinin kontrolleri.
<b>SİS AKTÖRÜNÜN SİSTEMİ</b>	Sis aktörünün temel komutları, sürekli sis üretip üretmediği, zamana bağlı bir şekilde çalışıp çalışmadığı, oyuncuyu etkileme biçimi.
<b>ARA YÜZ SİSTEMİ</b>	Oyun içindeki menüler, pencereler ve kullanıcı ara yüz elemanlarının doğru çalışmasının kontrolü. Ara yüzdeki butonların işlevselliğinin test edilmesi.
<b>VERİ TABANI VE KAYIT SİSTEMİ</b>	Oyuncu ilerlediği kısımları eğer kaydetmek isterse kayıt yapmak için doğruluğunun kontrolü.
<b>İLERLEME VE SEVİYE SİSTEMİ</b>	Oyuncunun bir seviyeyi başarıyla tamamlaması durumunda oyunun doğru bir şekilde bir sonraki seviyeye geçip geçmediğinin kontrolü. Seviye tasarımındaki olası hataların test edilmesi.
<b>OYUN SES SİSTEMİ</b>	Oyundaki seslerin aktörlerinin seslerinin düzgün bir şekilde çalışıp çalışmadığı, Canavarların veya korku seslerinin düzgün entegre edilip edilmediği

Tablo 13.2 Sistemin Test Edilemeyen Alt Sistemleri

TEST EDİLEMİYEN ALT SİSTEMLER	TEST EDİLEMİYEN SENARYOLAR
<b>OYUN MEKANİĞİ</b>	Karakterin fiziksel görünümü: Karakter tasarımı, genellikle estetik ve kullanıcı tercihlerine dayandığı için objektif bir testle değerlendirilemez.
<b>GRAFİK VE ARAYÜZ SİSTEMİ</b>	Grafik tasarımı ve estetik değerlendirmesi: Oyunun grafik tasarımı, genellikle sanatsal tercihlere ve estetik algılara dayandığı için objektif bir test sürecine tabi tutulamaz.
<b>VERİTABANI</b>	Veri tabanı güvenliği: Veri tabanı güvenliği genellikle güvenlik önlemleri ve kod incelemeleriyle değerlendirilir. Ancak, bu değerlendirme tam anlamıyla test sürecine tabi tutulamaz.
<b>KONTROL FARKLILIKLARI SİSTEMİ</b>	Oyuncunun kullandığı kontrol cihazı tipine göre yaşanabilecek olan girdi gecikmeleri
<b>OYUNCU BECERİSİ</b>	Oyuncuların beceri seviyeleri ve oyun içindeki gelişimleri, oyunun zorluk seviyelerini ve dengeyi etkileyebilir. Her oyuncunun beceri düzeyi farklı olduğundan, bu özellik önceden tam olarak tahmin edilemez.
<b>OYUNCU KARARLARI</b>	Oyuncuların oyun içinde hangi seçimleri yapacakları ve bu seçimlerin nasıl bir etki yaratacağı önceden belirlenemez. Oyuncuların stratejileri, tercihleri ve davranışları oyunu beklenmedik şekillerde etkileyebilir.

## TEST PLANI

### Test Seviyeleri:

#### Birim Testleri:

- ☐ Grafik motoru fonksiyonları test edilecek.
- ☐ Oyun mekaniği işlevleri, doğruluk ve performans açısından kontrol edilecek.

#### Entegrasyon Testleri:

- ☐ Grafik motoru ile oyun mekaniği arasındaki entegrasyon test edilecek.
- ☐ Oyunun farklı bileşenlerinin birleşimi ve birbirleriyle uyumu kontrol edilecek.

#### Sistem Testleri:

- ☐ Oyunun genel performansı, kullanıcı ara yüzü ve tüm özellikleri test edilecek.

### Test Türleri:

#### Fonksiyonel Testler:

- ☐ Oyunun temel özellikleri, karakter kontrolü, düşman etkileşimleri test edilecek.
- ☐ Görevler, seviye geçişleri ve oyun içi etkileşimler kontrol edilecek.

#### Performans Testleri:

- ☐ Oyunun hızı, yüksek grafikler altında performansı test edilecek.
- ☐ Tepki süresi ve yüksek kullanıcı trafiği durumunda performans testleri yapılacaktır.

#### Güvenlik Testleri:

- ☐ Oyunun güvenlik önlemleri, kullanıcı verilerinin şifreleme durumu kontrol edilecek.
- ☐ Oyunun sunucu tarafında güvenlik açıkları taranacaktır.

### Test Yöntemleri:

#### El İle Yapılan Testler:

- ☐ Kullanıcı ara yüzü testleri, oyun içi deneyim kontrolü manuel olarak yapılacaktır.
- ☐ Oyunun farklı platformlarda (PC, konsol) manuel testleri gerçekleştirilecektir.

#### Otomatik Testler:

- ☐ Oyunun belirli senaryoları, otomatik test araçları kullanılarak sürekli test edilecektir.
- ☐ Performans testleri otomatik test senaryoları ile tekrarlı olarak yürütülecektir.

### Beyaz Kutu / Kara Kutu / Gri Kutu Testleri:

#### Beyaz Kutu Testleri:

- ☐ Grafik motoru kodları üzerinde detaylı testler gerçekleştirilecektir.
- ☐ Oyun mekaniği fonksiyonlarının iç yapısı detaylı olarak kontrol edilecektir.

#### Kara Kutu Testleri:

- ☐ Oyunun kullanıcıdan gelen girişlere nasıl tepki verdiği kontrol edilecektir.
- ☐ İlk defa kullanıcı tarafından görülen hataların kara kutu testleri yapılacaktır.

#### Gri Kutu Testleri:

- ☐ Oyunun veri tabanı ile etkileşimi, iç yapısının bazıları kontrol edilecektir.
- ☐ Kullanıcı deneyimi ve oyunun iç dinamikleri gri kutu testleri ile kontrol edilecektir.

Bu test planı, The Mist oyununun geliştirme sürecinde kalite kontrolünü sağlamak için kullanılabilir. Testlerin kapsamı, belirli senaryolara odaklanarak yazılımın güvenilir ve performanslı bir şekilde çalışmasını sağlamayı hedefler.

Tablo 13.3 Sistemin Elle Gerçekleştirilen Testleri

ELLE GERÇEKLEŞTİRİLEN TEST	TEST EDİLEN ÖZELLİK	TEST TÜRÜ
Oyunu Başlatmak İçin ‘Oyunu Başlat’ Tuşuna Bas	Kullanıcının yeni oyunu başlatması	Fonksiyonellik / Kullanılabilirlik
Oyunu Başlatmak İçin ‘Oyuna Devam Et’ Butonuna Bas	Kullanıcının kayıtlı oyuna devam etmesi	Fonksiyonellik / Kullanılabilirlik
Oyundur Durdurmak İçin ‘Oyunu Durdur’ Butonuna Bas	Kullanıcının oyunu durdurmasını sağlamak	Fonksiyonellik / Kullanılabilirlik
Oyundan Çıkmak İçin ‘Oyunu Bitir’ Tuşuna Bas	Kullanıcının oyundan başarılı bir şekilde çıkması	Fonksiyonellik / Kullanılabilirlik
Karakteri Hareket Ettirmek İçin Atanmış Olan Yön Tuşlarına Bas	Kullanıcının karakteri hareket ettirmesi	Fonksiyonellik / Kullanılabilirlik
Karakterin Kaç / Saklan Aksiyonu İçin Atanmış Tuşlara Bas	Karakterin düşman karşısında kaçıp saklanmasını sağlar	Fonksiyonellik / Kullanılabilirlik
Karakteri Kullanıcı Olmayan Düşmanlara Yönlendir	Karakterin oyun içerisindeki kullanıcı olmayan düşmanlar ile etkileşimi	Fonksiyonellik / Kullanılabilirlik
Karakterin Eşyaları Toplamak Ve Envantere Ekleme İçin Olan Tuşa Bas	Karakter eşyaları toplayıp envantere ekler veya çıkarır	Fonksiyonellik / Kullanılabilirlik
Oyun İçindeki Bulmacaların Doğru Bir Şekilde Çözülüp Çözülemediğini Kontrol Etmek	Oyuncunun doğru cevapları girdiğinde bulmacanın başarıyla tamamlandığı ve ilerleme kaydedildiği.	Fonksiyonellik / Kullanılabilirlik
Görev Tamamlama Testi	Oyuncunun oyun içindeki görevleri tamamlayabilmesi için görev sistemi test edilecek.	Fonksiyonellik / Kullanılabilirlik
Canavarla İteraksiyon Testi	Oyuncunun canavarlarla etkileşimi test edilecek. Oyuncu, bir canavarla karşılaşacak ve etkileşime geçmeye çalışacak.	Fonksiyonellik / Kullanılabilirlik
Zaman Kontrol Testi	Oyun içindeki zaman sınırlı görevlerin test edilmesi. Oyuncu, belirli bir süre içinde bir görevi tamamlamaya çalışacak.	Fonksiyonellik / Kullanılabilirlik



## 14. Test Cases

### □ Test Kimliği ve Alt Sistemler:

The Mist oyunu için belirlenen test kimlikleri, oyunun farklı alt sistemlerinde gerçekleştirilecek testleri tanımlar. Bu kimlikler, her bir testin özel bir alt sistemle ilişkilendirilmesini sağlar.

Örneğin, karakter kontrolleri için "CHAR-CTRL-TEST", grafik ve arayüz için "GRAPH-UI-TEST" gibi.

### □ Test Türleri ve Özellikler:

Oyunun alt sistemlerine özgü olarak belirlenen test türleri, oyunun işlevselliği, birim testi, entegrasyon testi ve diğer özel test türlerini içerir. Örneğin, görsel efektlerin testi için "VISUAL-EFFECT-TEST", oyun ilerleme sistemi için "GAME-PROGRESS-TEST" gibi.

### □ Giriş Betimlemeleri ve Eylemler:

Her bir test case için belirlenen giriş betimlemeleri, kullanıcının oyun içindeki etkileşimini simgeler. Bu eylemler, klavye girişleri, fare tıklamaları veya oyun kontrol cihazları üzerinden yapılan eylemleri içerir.

Örneğin, karakterin silah kullanma eylemi için "WEAPON-USE-ACTION", harita üzerinde hareket etme eylemi için "MAP-NAVIGATION" gibi.

### □ Beklenen Çıktılar ve Hata Yönetimi:

Her test case için belirlenen beklenen çıktılar, oyunun başarılı bir şekilde tamamlanması durumunda elde edilmesi gereken sonuçları tanımlar. Ayrıca, hata yönetimi için olası hataların kaynakları, önem dereceleri ve hata tipleri belirlenir.

Örneğin, oyun içinde beklenmeyen bir çökmeye neden olan hataların yönetimi için "CRASH-MANAGEMENT", kullanıcının hareket etme sırasında karşılaştığı hatalar için "NAVIGATION-ERROR-MANAGEMENT" gibi.

Bu şekilde belirlenen test kimlikleri, test türleri, giriş betimlemeleri, beklenen çıktılar ve hata yönetimi, The Mist oyun projesinin alt sistemlerinin test süreçlerini açıklamak ve belgelemek için kullanılır.

## AYRINTILI TEST DURUMLARI VE SONUÇLARI

TEST CASE ID	TCH-01
Test Amaçları	Karakter kontrol testi.
Test Prosedürleri ve Sonuçları	Klavye tuşlarına basılarak karakterin sağa, sola, yukarı ve aşağı hareketi kontrol edilir
Test Verisi	Klavyeden girilen tuş bilgileri
Beklenen Sonuç	Karakterin istenilen yönlere doğru hareket etmesi
Gerçek Sonuç	Beklenen sonuç ile aynı
Hata Türü	Hareket hatası
Hata Önemi	Önemli

TEST CASE ID	TCH-02
Test Amaçları	Görev Tamamlama Testi
Test Prosedürleri ve Sonuçları	Belirli bir göreve tıklanarak görevin başarıyla tamamlanması kontrol edilir
Test Verisi	Görev seçim bilgileri
Beklenen Sonuç	Görevin başarıyla tamamlanması ve bilgilendirmenin alınması
Gerçek Sonuç	Beklenen sonuç ile aynı
Hata Türü	Görev tamamlama hatası
Hata Önemi	Önemli



TEST CASE ID	TCH-03
Test Amaçları	Seviye Geçiş Testi
Test Prosedürleri ve Sonuçları	Bir Seviyeden Diğerine Geçiş Mekanizması Test Edilir.
Test Verisi	Seviye Tamamlama Bilgileri
Beklenen Sonuç	Oyuncunun Bir Sonraki Seviyeye Başarıyla Geçmesi
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Geçiş Hatası
Hata Önemi	Yüksek

TEST CASE ID	TCH-04
Test Amaçları	Oyun İçi Etkileşim Testi
Test Prosedürleri ve Sonuçları	Diğer Aktörlerle Etkileşim Test Edilir.
Test Verisi	Etkileşim Bilgileri
Beklenen Sonuç	Başarılı Etkileşim Sonucunda Bilgi Alınması
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Etkileşim Hatası
Hata Önemi	Orta

TEST CASE ID	TCH-05
Test Amaçları	Grafik Ve Ara yüz Kontrol Testi
Test Prosedürleri ve Sonuçları	Oyunun Grafikleri Ve Ara yüzü Test Edilir.
Test Verisi	Ekran Gezinme Bilgileri
Beklenen Sonuç	Grafik Ve Ara yüzün Sorunsuz Görünmesi
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Görüntü Hatası
Hata Önemi	Orta

TEST CASE ID	TCH-06
Test Amaçları	Performans Testi
Test Prosedürleri ve Sonuçları	Oyunun Performansı, Hızı ve Tepki Süresi Test Edilir
Test Verisi	Performans Testi İçin Uygun Durumlar
Beklenen Sonuç	Tatmin Edici Performans
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Performans Düşüklüğü
Hata Önemi	Yüksek

TEST CASE ID	TCH-07
Test Amaçları	Hata Kontrol Testi
Test Prosedürleri ve Sonuçları	Oyun İçinde Karşılaşılan Hatalar Kontrol Edilir
Test Verisi	Hata Taklit Etme Bilgileri
Beklenen Sonuç	Hata Durumlarında Uygun Hata Mesajlarının Görüntülenmesi
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Hata Mesajı Eksikliği
Hata Önemi	Orta

TEST CASE ID	TCH-08
Test Amaçları	Bulmaca Çözme Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Oyun İçindeki Bulmacaları Başarıyla Çözmesi Kontrol Edilir
Test Verisi	Bulmaca Çözme Bilgileri
Beklenen Sonuç	Bulmacanın Başarıyla Çözülmesi
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Çözüm Hatası
Hata Önemi	Orta

TEST CASE ID	TCH-09
Test Amaçları	Zaman Kontrol Testi
Test Prosedürleri ve Sonuçları	Zaman Sınırlı Görevlerin Belirli Süre İçinde Tamamlanması Kontrol Edilir.
Test Verisi	Zaman Kontrolü Bilgileri
Beklenen Sonuç	Görevin Belirtilen Süre İçinde Tamamlanması
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Zaman Aşımı
Hata Önemi	Önemli

TEST CASE ID	TCH-10
Test Amaçları	Gelişmiş Bulmaca Çözme Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Karmaşık Bulmacaları Çözmesi Kontrol Edilir
Test Verisi	Karmaşık Bulmaca Çözme Bilgileri
Beklenen Sonuç	Zorlu Bulmacanın Başarıyla Çözülmesi
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Çözüm Zorluğu
Hata Önemi	Yüksek

TEST CASE ID	TCH-11
Test Amaçları	Özel Yetenek Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Karakterinin Özel Yeteneklerini Doğru Bir Şekilde Kullanması Kontrol Edilir.
Test Verisi	Özel Yetenek Kullanım Bilgileri
Beklenen Sonuç	Özel Yeteneklerin Doğru Bir Şekilde Çalışması
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Yetenek Hatası
Hata Önemi	Orta

TEST CASE ID	TCH-12
Test Amaçları	Bölge Keşif Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Oyun Dünyasında Gizli Bölgeleri Keşfetmesi Kontrol Edilir
Test Verisi	Keşif Bilgileri
Beklenen Sonuç	Gizli Bölgelere Başarıyla Erişim Sağlamak
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Keşif Eksikliği
Hata Önemi	Orta

TEST CASE ID	TCH-13
Test Amaçları	Etkileşimli Hikâye Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Etkileşimli Hikaye Unsurlarını Doğru Anlaması Kontrol Edilir.
Test Verisi	Hikaye Etkileşim Bilgileri
Beklenen Sonuç	Oyuncunun Hikayeyi Doğru Bir Şekilde Anlaması
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Hikaye Anlama Hatası
Hata Önemi	Orta

TEST CASE ID	TCH-14
Test Amaçları	Kritik Hasar Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Kritik Hasar Sistemini Doğru Bir Şekilde Denemesi Kontrol Edilir
Test Verisi	Kritik Hasar Bilgileri
Beklenen Sonuç	Kritik Vuruşlardaki Artışın Doğru Bir Şekilde Uygulanması
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Kritik Hasar Eksikliği
Hata Önemi	Orta

TEST CASE ID	TCH-15
Test Amaçları	NPC'lerden Kaçış Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Oyun Dünyasındaki Npc'lerden Kaçması Kontrol Edilir
Test Verisi	NPC Kaçış Bilgileri
Beklenen Sonuç	Npclerden Zarar Almadan Kaçmak
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Kaçış Hatası
Hata Önemi	Orta

TEST CASE ID	TCH-16
Test Amaçları	Mücadele Beceri Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Oyun İçindeki Mücadele Becerilerini Kontrol Etmek
Test Verisi	Mücadele Becerisi Bilgileri
Beklenen Sonuç	Düşmanlarla Başarılı Bir Şekilde Savaşmak
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Mücadele Hatası
Hata Önemi	Orta

TEST CASE ID	TCH-17
Test Amaçları	Canavar Davranış Testi
Test Prosedürleri ve Sonuçları	Oyun İçindeki Canavarın Doğru Davranışlarını Kontrol Etmek
Test Verisi	Davranış Bilgileri
Beklenen Sonuç	Yaratıkların Doğru Bir Şekilde Tepki Vermesi
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Davranış Hatası
Hata Önemi	Orta

TEST CASE ID	TCH-18
Test Amaçları	Kamera Kontrol Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Oyun İçindeki Kamera Kontrollerini Test Etmek
Test Verisi	Kamera Kontrol Bilgileri
Beklenen Sonuç	Kameranın Doğru Bir Şekilde Hareket Etmesi
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Kamera Kontrol Hatası
Hata Önemi	Orta



TEST CASE ID	TCH-19
Test Amaçları	Fizik Motoru Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Oyun İçindeki Fizik Motorunu Kontrol Etmek
Test Verisi	Fizik Bilgileri
Beklenen Sonuç	Nesnelerin Doğru Bir Şekilde Hareket Etmesi ve Etkileşime Girmesi
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Fizik Hatası
Hata Önemi	Önemli

TEST CASE ID	TCH-20
Test Amaçları	Oyun İçi Ses Efekt Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Oyun İçindeki Ses Efektlerini Kontrol Etmek
Test Verisi	Ses Bilgileri
Beklenen Sonuç	Ses Efektlerinin Doğru Bir Şekilde Çalışması
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Ses Eksikliği
Hata Önemi	Orta

TEST CASE ID	TCH-21
Test Amaçları	İlerleyen Seviyelerde Zorlaşan Görev Testi
Test Prosedürleri ve Sonuçları	Oyuncunun İlerleyen Seviyelerde Karşılaştığı Görevlerin Zorluk Seviyesinin Artması Ve Başarılı Bir Şekilde Tamamlanması.
Test Verisi	Görev Zorluk Bilgileri
Beklenen Sonuç	Oyuncunun Seviyeye Uygun Görevleri Başarılı Bir Şekilde Tamamlaması.
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Görev Zorluk Hatası
Hata Önemi	Yüksek

TEST CASE ID	TCH-22
Test Amaçları	Oyun İçi İleri Düzey Grafik Testi
Test Prosedürleri ve Sonuçları	Oyunun Yüksek Grafik Ayarlarında Sorunsuz Bir Şekilde Çalışması.
Test Verisi	Grafik Ayarları
Beklenen Sonuç	Oyunun Yüksek Grafik Ayarlarında Sorunsuz Bir Şekilde Çalışması Ve Görsel Kalitenin Yüksek Olması.
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Grafik Performans Hatası
Hata Önemi	Yüksek

TEST CASE ID	TCH-23
Test Amaçları	Oyun İçi Başarı Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Belirli Görevleri Tamamlayarak Başarı Elde Etmesi.
Test Verisi	Başarı Bilgileri
Beklenen Sonuç	Oyuncunun Belirli Görevleri Başarıyla Tamamlayarak Oyun İçi Başarı Elde Etmesi.
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Başarı Elde Etme Hatası
Hata Önemi	Orta

TEST CASE ID	TCH-24
Test Amaçları	Yeniden Bağlanma Testi
Test Prosedürleri ve Sonuçları	Oyuncunun İnternet Bağlantısının Kesilmesi Durumunda Oyuna Tekrar Bağlanabilmesi.
Test Verisi	Bağlantı Bilgileri
Beklenen Sonuç	Oyuncunun İnternet Bağlantısı Kesildikten Sonra Tekrar Bağlanabilmesi Ve Oyun Durumunu Koruması.
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Bağlantı Hatası
Hata Önemi	Yüksek

TEST CASE ID	TCH-25
Test Amaçları	Oyun İçi İstatistik Kontrolü
Test Prosedürleri ve Sonuçları	Oyuncunun Oyun İçi İstatistiklerinin Doğru Bir Şekilde Kaydedilmesi.
Test Verisi	İstatistik Bilgileri
Beklenen Sonuç	Oyuncunun Oyun İçi Performansının Doğru Bir Şekilde Kaydedilmesi Ve İstatistiklerinin Güncellenmesi.
Gerçek Sonuç	Beklenen Sonuç ile Aynı
Hata Türü	İstatistik Kaydetme Hatası
Hata Önemi	Düşük

TEST CASE ID	TCH-26
Test Amaçları	Oyun İçi İstatistik Kontrolü
Test Prosedürleri ve Sonuçları	Oyuncunun Oyun İçi İstatistiklerinin Doğru Bir Şekilde Kaydedilmesi.
Test Verisi	İstatistik Bilgileri
Beklenen Sonuç	Oyuncunun Oyun İçi Performansının Doğru Bir Şekilde Kaydedilmesi Ve İstatistiklerinin Güncellenmesi.
Gerçek Sonuç	Beklenen Sonuç ile Aynı
Hata Türü	İstatistik Kaydetme Hatası
Hata Önemi	Düşük

TEST CASE ID	TCH-27
Test Amaçları	Sis Aktörü Kontrol Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Oyun İçinde Karşılaştığı Sis Efektinin Doğru Bir Şekilde Çalışması.
Test Verisi	Sis Aktörü Bilgileri
Beklenen Sonuç	Sis Aktörünün Sis Efektinin Atmosferi Etkileyici Bir Şekilde Değiştirmesi.
Gerçek Sonuç	Beklenen Sonuç ile Aynı
Hata Türü	Sis Aktörü Hatası
Hata Önemi	Yüksek

TEST CASE ID	TCH-28
Test Amaçları	Kilitli Kapı Açma Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Bir Görevi Tamamladıktan Sonra Kilitli Bir Kapıyı Başarılı Bir Şekilde Açması.
Test Verisi	Kapı Açma Bilgileri
Beklenen Sonuç	Oyuncunun Görevi Tamamladıktan Sonra Kilitli Kapının Başarılı Bir Şekilde Açılması.
Gerçek Sonuç	Beklenen Sonuç ile Aynı
Hata Türü	Kapı Açma Hatası
Hata Önemi	Yüksek

TEST CASE ID	TCH-29
Test Amaçları	Anahtar Toplama Kontrolü
Test Prosedürleri ve Sonuçları	Oyuncunun Bir Görevi Tamamladıktan Sonra Anahtar Toplaması Ve Bu Anahtarın Doğru Bir Şekilde Kaydedilmesi.
Test Verisi	Anahtar Toplama Bilgileri
Beklenen Sonuç	Oyuncunun Anahtarı Başarılı Bir Şekilde Toplaması Ve İlerleyen Görevlerde Kullanabilmesi.
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Anahtar Toplama Hatası
Hata Önemi	Orta

TEST CASE ID	TCH-30
Test Amaçları	Kaçıp Saklanma Yeteneği Testi
Test Prosedürleri ve Sonuçları	Oyuncunun Tehlikeli Durumlarla Karşılaştığında Kaçıp Saklanabilme Yeteneğini Kontrol Etme.
Test Verisi	Kaçıp Saklanma Bilgileri
Beklenen Sonuç	Oyuncunun Düşmanlardan Kaçıp Saklanma Yeteneğini Başarılı Bir Şekilde Kullanabilmesi.
Gerçek Sonuç	Beklenen Sonuç İle Aynı
Hata Türü	Kaçıp Saklanma Hatası
Hata Önemi	Orta

# V Sözlük

## ▪ Abstraksiyon:

Karmaşık bir sistemi basitleştirmek amacıyla, detaylardan arındırarak yalnızca önemli özelliklere odaklanma süreci.

## ▪ Ağ Şeması:

Bir sistemdeki cihazların veya fiziksel veya değiştirme bağlantılarının bir diyagram tipini gösteren. Bu, ağ mimarisini yönetmek için kullanılır.

## ▪ Arayüz Tasarımı:

Kullanıcıların bir yazılım ürünü ile etkileşimde bulunduğu grafiksel veya metinsel arayüzlerin planlanması ve oluşturulması.

## ▪ Microservices Architecture:

Yazılımın küçük, bağımsız ve ölçeklenebilir hizmetlere bölündüğü bir mimari yaklaşım. Her bir hizmet, belirli bir işlevselliği temsil eder.

## ▪ Modülerlik:

Yazılımın bağımsız ve yeniden kullanılabilir parçalara bölünmesi, böylece sistemin daha kolay bakımını ve geliştirmesini sağlama.

## ▪ Nesne Yönelimli Tasarım (OOP):

Yazılım tasarımının bir paradigması; programın nesnelerle (veri yapıları ve metodları) modellenmesi ve bu nesneler arasındaki ilişkilerin kullanılması.

## ▪ Performans Optimizasyonu:

Yazılımın çalışma hızını, bellek kullanımını ve diğer performans metriklerini iyileştirmek için yapılan tasarım ve geliştirme optimizasyonları.

## ▪ Sequence Diagram:

Sistemdeki nesnelerin ve bu nesneler arasındaki etkileşimlerin zamansal sıralamalarını gösteren bir UML diyagram türüdür.

## ▪ Sistem Entegrasyonu:

Farklı alt sistemlerin veya bileşenlerin birleştirilerek tek bir çalışan sistem oluşturulması süreci.

## ▪ Test Otomasyonu:

Yazılım testlerinin otomatikleştirilmesi, böylece tekrar eden test süreçleri daha hızlı ve güvenilir bir şekilde gerçekleştirilebilir.

## ▪ UML (Birleşik Modelleme Dili):

Yazılım tasarımını görsel olarak temsil etmek için kullanılan standart bir anlatım dilidir. Sıklıkla kullanım durumu, sıra, sınıf diyagramları gibi unsurları içerir.

- **Veri tabanı Tasarımı:**

Verilerin etkili bir şekilde depolanması, yönetilmesi ve erişilmesini sağlamak için veri tabanı şemalarının oluşturulması.

- **Yazılım Güvenliği:**

Yazılımın, siber saldırılara karşı dayanıklılığını sağlamak için güvenlik önlemlerinin tasarımı ve uygulanması.

- **Yeniden Kullanılabilirlik:**

Yazılım bileşenlerinin veya modüllerin farklı projelerde veya farklı bağlamlarda tekrar kullanılabilme yeteneği.