EVRY İŞ BAŞVURUSU PROJESİ

Mustafa KIR

Mesut KARADAĞ

İSTANBUL

2023

ÖZET

EVRY İŞ BAŞVURUSU PROJESİ

Mustafa KIR

Bu projede, bir blok kırma oyunu geliştirilmiştir. Oyun, kullanıcıların blokları yok ederek seviyeleri geçtiği bir oyundur. Bu proje asıl amacının oyunun görüntüsü ve oyun mekaniği eklemek olamdığını, oyunun backend'ini solid prensibler, event sistemi ve desing patternlerine uygulayabildiği göstermeyi amaçladım. Bu kapsamda oyunda 3 ana tasarım desenine uyguladım. Bu kalıplar:

Singleton Tasarım Kalıbı: Oyunun çeşitli bileşenlerinin tek bir örneğinin oluşturulmasını sağladım.

Koddan bir örnek,

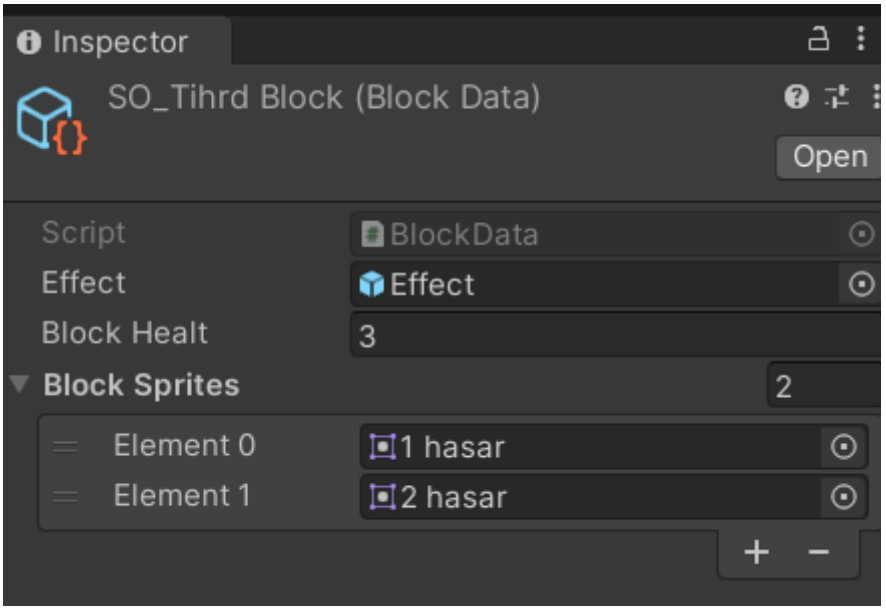
```
11 başvuru
public static LevelManager Instance { get; set; }
Unity İletisi | 0 başvuru
private void Awake()
{
    #region Singloton
    if (Instance == null)
    {
        Instance = this;
        DontDestroyOnLoad(gameObject);
    }
    else
    {
        Destroy(this.gameObject);
        return;
    }
    #endregion
}
```

Object Pool Tasarım Kalıbı: Oyunda sürekli olarak oluşturulan ve yok edilen nesnelerin yönetimini kolaylaştırmak için Object Pool tasarım kalıbı kullanılmıştır. Blok veya efekt nesneleri gibi sıkça kullanılan nesneler, önceden oluşturulmuş havuzlarda saklanarak performans artırılmıştır.

Oluşturulan havuz:

Flyweight Tasarım Kalıbı: Oyundaki blokların görsel ve temel özelliklere sahip nesneleri Flyweight tasarım kalıbı kullanarak bellek kullanımı optimize yapmayı amaçladım. Bu kapsamda blok nesnelerin efekleri, sağlık ve sprite nesnelerini tutan scritable objects'ler oluşturdum.

Örnek bir Scritable Object:



Event Sistemi: Oyunuma event sistemine entegre ettim. Oyun başlaması, oyun sonlanması, oyunun durması, oyunun devam etmesi, levlin tamamlanması ve sonraki level geçiş gibi işlemleri eventi sistemi ile yaptım. böylelikle hem temiz ve okunabilir bir kod yazmayı hem de performs optimizasyon yapmayı amaçladım. Koddan örnek

```
public delegate void GameEvent();
public static event GameEvent OnGameStart;
public static event GameEvent OnGameEnd;
public static event GameEvent OnGamePause;
public static event GameEvent OnGameContinue;
public static event GameEvent OnComplatedLevel;
public static event GameEvent OnNextLevel;
```

Proje Detayları

Proje Konusu: Unity ile 2 boyutlu blok kırma oyunu.
Proje Amacı: İş başvurusu değerlendirmesi için bir 2 boyutlu blok kırma oyununun geliştirilmesi.

Proje Yapılışı ve İçeriği:

Bu proje, kullanıcı deneyimini ve oyun performansını optimize etmek amacıyla tek bir sahne üzerinde geliştirildi. Sahneler arası geçişlerde karşılaşılabilecek potansiyel sıkıntıları ve veri kaybını engellemeği amaçladım.

Oyunun kullanıcı arayüzü için "IMenuItem" adında bir interface tanımlanmıştır. Bu interface sayesinde, menü tuşları arasında tutarlılık sağlanmış ve her bir tuş, kalıtım mantığı kullanılarak geliştirilmiştir. Bu yaklaşım, kod tekrarını azaltmaya ve sistemin genişletilebilirliğini artırmaya yardımcı olur.

Oyun içi olayların yönetimi için bir Event sisteminden yararlanılmıştır. Oyun başlatma, duraklatma, devam ettirme, seviye tamamlama, sonraki seviye geçişi ve oyun sonu gibi önemli olaylar bu sistem üzerinden yönetilir. Bu sistem, oyunun farklı bölümleri arasında düzgün bir iletişim kurulmasını sağlayarak, oyun mantığının daha temiz ve yönetilebilir olmasına olanak tanır.

Oyunda bar kontrolü için de kalıtım yaklaşımı benimsenmiştir. Kullanıcılar, barı iki farklı yöntemle kontrol edebilirler: birinci yöntemde fare kullanılırken, ikinci yöntemde bir yapay zeka algoritması devreye girer. Bu amaca yönelik olarak, "MoveWithAI" ve "MoveWithMouse" adında iki sınıf oluşturulmuştur. Her iki sınıf da "IGameBarMove" interface'inden miras alır. Bu interface, barın hareket etme mekanizmasını standart bir şekilde tanımlar, böylece farklı kontrol mekanizmaları aynı arayüzü uygulayarak oyunun genel yapısına entegre edilebilir.
NOT: Yapay zeka kontörlünü aktif etmek için oyundaki GameBar nesnesine gidip GameBarController scriptindeki AI oyun başlatmadan önce aktif etmeniz gerekir.

Oyun içindeki top kontrolü, topun Rigidbody bileşeni üzerinden gerçekleştirilmiştir. Topun hareketi, Rigidbody'nin velocity özelliğine hız değeri verilerek sağlanır. Bu yaklaşım, topun fiziksel hareketlerinin gerçekçi ve tutarlı olmasını garanti eder, çünkü Unity'nin fizik motoru bu hareketleri doğal yasalara uygun şekilde işler.
Oyunun sonlandırılması ise bir çarpışma algılama sistemi üzerinden yönetilir. Çarpışma sırasında, çarpışan nesnelerden birinin etiketi (TAG) kontrol edilir. Örneğin, topun alt sınır ile olan çarpışması bir oyun kaybı durumunu tetikler. Bu durumda, belirli bir TAG'e sahip nesne ile çarpışma algılandığında oyun sonlandırılır veya belirli bir olay tetiklenir.

Oyundaki blokların özelliklerini yönetmek için Scriptable Objects kullanıldı. Bu objeler, oyunda kullanılan blokların çeşitli özelliklerini saklamak için tasarlandı. Scriptable Objects, blokların özelliklerini merkezi bir konumda tutarak, fazladan veri tasarrufu sağlar ve oyunun performansını optimize eder. Bu yöntem, oyun içinde tekrar tekrar kullanılacak blok türlerinin özelliklerini tanımlamak için idealdir, çünkü blok özellikleri tek bir yerde tanımlandığı için, oyun sahnesine eklenen her blok için bu özellikler kolayca yeniden kullanılabilir. Ayrıca, bu yapının kullanımı, oyun içindeki veri yönetimini daha düzenli ve anlaşılır hale getirirken, aynı zamanda oyunun genel bellek kullanımını azaltır ve performansı artırır. Bu, özellikle büyük ve karmaşık oyunlar için kritik bir avantaj sağlar.

Oyundaki seviye yüklemeleri ve mevcut seviyenin ilerlemesinin kaydedilmesi, JSON dosyaları kullanılarak gerçekleştirilmektedir. Seviye bilgileri, bloksData.json dosyasında saklanmakta, burada her bir seviyedeki blok sayısı ve konumları tanımlanmıştır. Bu dosya, oyunda yüklenecek olan seviyelerin blok konumları saklamaktadır. Oyuna yeni bir seviye eklemek istendiğinde, bu dosyanın güncellenmesi yeterli olur; böylece Unity ortamında ya da kodda herhangi bir değişiklik yapılmasına gerek kalmaz. Bu yaklaşım, kodun bakım kolaylığını ve modüler yapıldığını gösterir.
Oyuncunun mevcut ilerlemesinin kaydedilmesi ise level.json adlı başka bir JSON dosyası üzerinden yapılır. Bu dosya, oyuncunun hangi seviyede olduğunu, bu seviyede ne kadar ilerlediğini içerir.