Proje Dokümantasyonu

Bu dokümantasyon, Unity oyun motoru kullanılarak geliştirilmiş modüler bir projenin temel bileşenlerini, işlevselliğini ve tasarım prensiplerini detaylandırmaktadır. Proje; oyuncu ilerlemesi, kaynak yönetimi, dinamik üretim mekanikleri ve envanter sistemi gibi ana unsurları kapsamaktadır. Tüm kaydet/yükle işlemleri, bileşenlerin Inspector üzerinden sürükle bırak yöntemiyle merkezi SaveLoadManager'a entegre edilebilmesini sağlayan **ISaveable** arayüzü ile standart hale getirilmiştir.

İçindekiler

---

1. Genel Bakış

Proje, Unity motoru ile geliştirilmiş modüler bir yapıyı ortaya koymaktadır. Oyuncu ilerlemesi, kaynak yönetimi, üretim mekanikleri ve envanter sistemi gibi kilit unsurlar; bağımsız olarak çalışacak, ancak birbirleriyle sorunsuzca entegre olabilecek şekilde tasarlanmıştır. Kaydet/yükle işlemleri için **ISaveable** arayüzü kullanılarak her bileşenin durumu JSON formatında saklanıp geri yüklenebilmektedir. Bu sayede geliştirme sırasında, bileşenler Inspector üzerinden sürükle bırak yöntemiyle merkezi SaveLoadManager'a kolayca eklenebilmektedir.

2. Kod Yapısı ve Bileşenler

ISaveable Arayüzü

- **Amaç:**
  Durumlarını JSON formatında kaydedip geri yüklemesi gereken sınıflar için gerekli metotları tanımlar.
- **Ana Metotlar:**
  - **CaptureState():** Mevcut durumu JSON string olarak döndürür.
  - **RestoreState(string state):** Sağlanan JSON string'inden nesnenin durumunu geri yükler.
  - **ClearAll():** Nesnenin durumunu sıfırlar.
- **Özellik:**
  - **UniqueIdentifier:** Her kaydedilebilir nesne için benzersiz bir tanımlayıcı sağlar.

> **Not:** Kaydedilmek istenen tüm sınıflar, verilerini oluşturmak amacıyla ISaveable arayüzünü uygulamalıdır. Bu bileşenler daha sonra Inspector üzerinden SaveLoadManager'a sürükle bırak yöntemiyle eklenebilir.

LevelManager (Seviye Yönetimi)

- **Amaç:**
  Oyuncunun XP'sini ve seviyesini yönetmek.
- **Ana Fonksiyonlar:**
  - **XP Takibi ve Seviye Atlaması:**
    XP, AddXP(float xp) metodu ile eklenir; CheckLevelUp() metodu, yeterli XP biriktiğinde seviye atlamayı gerçekleştirir.
    RecalculateXPRequirement() metodu, bir sonraki seviye için gereken XP'yi hesaplar.
  - **UI Güncellemesi:**
    XP ve seviye değişiklikleri, UIManager aracılığıyla ekranda yansıtılır.
- **Kaydet/Yükle Desteği:**
  Durum, CaptureState(), hata yönetimi için try-catch ile genişletilmiş RestoreState(string state) ve ClearAll() metotları kullanılarak saklanır ve geri yüklenir.

PlayerManager (Oyuncu Yönetimi)

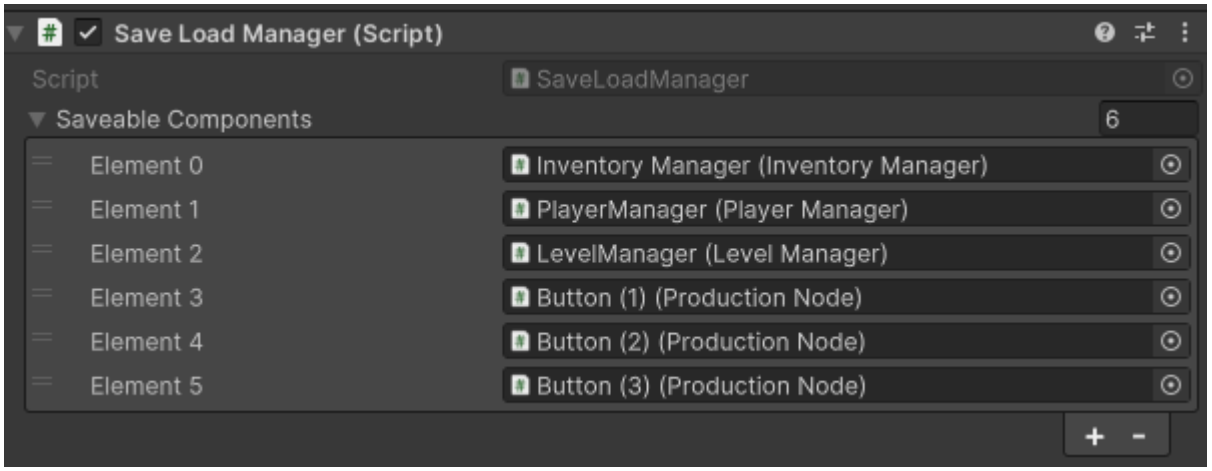
- **Amaç:**
  Oyuncunun adını ve seviyesini korur, günceller ve bu değişiklikleri UI üzerinde gösterir.
- **Ana Fonksiyonlar:**
  - **Oyuncu Verisi:**
    PlayerData sınıfı, oyuncunun adını ve seviyesini tutar.
  - **Veri Güncellemesi:**
    UpdatePlayerName() ve UpdatePlayerLevel(int levelIncrement) gibi metotlar, oyuncu verilerini değiştirir.
  - **UI Güncellemesi:**
    Değişiklikler, UIManager aracılığıyla ekranda anında yansıtılır.
- **Kaydet/Yükle Desteği:**
  Oyuncu verileri JSON formatında saklanır ve try-catch bloklarıyla çevrelenmiş RestoreState metodu kullanılarak geri yüklenir.

ProductionNode (Üretim Düğümü)

- **Amaç:**
  Belirlenen aralıklarda öğe üretir; üretim işlemi hem oyun sırasında (runtime) hem de offline modda gerçekleşir.
- **Ana Fonksiyonlar:**
  - **Üretim Süreci:**
    UpdateProduction() metodu, belirli zaman aralıklarında üretim işlemini gerçekleştirir.
    ComputeOfflineProduction() metodu, oyunun kapalı olduğu sürede üretilen miktarı hesaplar.
  - **Toplama Süreci:**
    CollectProducedItems() metodu, üretilen öğeleri toplayarak envantere aktarır.
- **Kaydet/Yükle Desteği:**
  Üretilen öğe sayısı ve son üretim zaman damgası, JSON formatında kaydedilip try-catch ile çevrelenmiş RestoreState(string state) metodu kullanılarak geri yüklenir.

SaveLoadManager (Kaydet/Yükle Yönetimi)

- **Amaç:**
  Oyundaki tüm ISaveable bileşenlerinin durumunu merkezi olarak yönetir.
- **Ana Fonksiyonlar:**
  - **Kaydetme:**
    SaveAll() metodu, her ISaveable bileşenin durumunu alır ve JSON formatında bir dosyaya yazar.
  - **Yükleme:**
    LoadAll() metodu, JSON dosyasını okuyarak her bileşenin durumunu geri yükler.
  - **Kayıtların Temizlenmesi:**
    ClearSaveFile() metodu, saklanan verileri temizler.
- **Entegrasyon:**
  - Saklanması gereken veriler, bir sınıf içinde toplanır ve ardından ISaveable arayüzü kullanılarak CaptureState() fonksiyonu ile JSON formatında elde edilir.
  - Tüm ISaveable bileşenler, Inspector üzerinden sürükle bırak yöntemiyle SaveLoadManager'a eklenir.



Inventory & Envanter Yönetimi

Inventory (Envanter Modeli)

- **Amaç:**
  Oyuncunun para ve öğelerini saklamak.
- **Ana Fonksiyonlar:**
  - **Para Yönetimi:**
    Para, AddCurrency(float amount) ile eklenir ve SpendCurrency(float amount) ile harcanır.
  - **Öğe Yönetimi:**
    Öğeler, AddItem(ItemSO itemData, int qty) metodu ile eklenir; RemoveItem(ItemSO itemData, int qty) metodu ile çıkarılır.
  - **Veri Sıfırlama:**
    ClearAllData() metodu, envanteri sıfırlar.

InventoryManager (Envanter Yöneticisi)

- **Amaç:**
  Envanter verisini yönetir, günceller ve UI güncellemelerini tetikleyen event mekanizmalarını kullanır.
- **Ana Fonksiyonlar:**
  - **İşlemler:**
    Para ve öğe ekleme/çıkarma işlemleri gerçekleştirilir.
  - **Event Sistemi:**
    OnInventoryChanged ve OnInventoryReset event'leri ile değişiklikler diğer bileşenlere bildirilir.
- **Kaydet/Yükle Desteği:**
  Envanter durumu, JSON formatında saklanır ve try-catch bloklarıyla korunan RestoreState metodu ile geri yüklenir.

InventoryUI (Envanter UI)

- **Amaç:**
  Envanterdeki öğelerin ve para bilgisinin kullanıcı arayüzünde görsel temsilini sağlamak.
- **Ana Fonksiyonlar:**
  - **Slot Oluşturma:**
    Envanterdeki her öğe için dinamik olarak UI slot'ları oluşturulur.
  - **UI Güncelleme:**
    RefreshUI() metodu, para ve öğe adetlerindeki değişiklikleri ekrana yansıtır.
  - **Slot Yenileme:**
    ResetSlots() metodu, mevcut slotları temizleyip yeniden oluşturur.

UIManager (Kullanıcı Arayüzü Yönetimi)

- **Amaç:**
  Oyuncu adı, XP, seviye ve para gibi genel UI elemanlarını yönetip günceller.
- **Ana Fonksiyonlar:**
  - **Oyuncu Adı Güncellemesi:**
    ApplyNameChange() metodu, input alanından alınan değeri kullanarak oyuncu adını günceller.
  - **Görsel Güncellemeler:**
    SetPlayerName(), SetXpText(), SetLevelText() ve SetCurrencyText() metotları, UI bileşenlerini anlık olarak günceller.

3. Test ve Mobil Uyumluluk

- **Test:**
  Geliştiriciler, ana ekranda XP, seviye, para ve envanter öğeleri gibi değerleri manipüle edebilmek için test tuşları eklemiştir. Yapılan değişiklikler anında ekranda görünür ve oyundan çıkıldığında otomatik olarak kaydedilip, yeniden girişte geri yüklenir.
- **Mobil Uyumluluk:**
  Kullanılan canvas, tüm mobil cihazlarda ölçeklenebilir şekilde yapılandırılmıştır. Mevcut UI düzeni, sadece test amaçlı hazırlanmış olup, üretim ortamında son kullanıcı deneyimine göre ayarlanabilir.

4. Tasarım Prensipleri

- **Tek Sorumluluk Prensibi:**
  Her sınıf, belirli bir işlevselliğe odaklanır (örneğin; seviye yönetimi, oyuncu yönetimi, envanter yönetimi, UI güncellemesi), bu da kodun okunabilirliğini ve bakımını artırır.
- **Açık/Kapalı Prensibi:**
  Yeni kaydedilebilir sınıflar, mevcut koda dokunulmadan eklenebilir; tek yapmanız gereken ISaveable arayüzünü uygulamaktır.
- **Liskov Yerine Koyma Prensibi:**
  ISaveable arayüzünü uygulayan herhangi bir bileşen, SaveLoadManager'da sorunsuzca birbirinin yerine kullanılabilir.
- **Arayüz Ayrımı Prensibi:**
  ISaveable arayüzü, veri kalıcılığı için gerekli temel metotları içerir ve gereksiz metotlardan arındırılmıştır.
- **Bağımlılıkların Tersine Çevrilmesi:**
  Gerekli bileşenler (örneğin; UIManager, InventoryManager) Inspector üzerinden dışarıdan enjekte edilerek sıkı bağımlılıkların önüne geçilmiş ve test edilebilirlik artırılmıştır.

---

5. Kullanım ve Genişletilebilirlik

- **Modüler Yapı:**
  Yeni bir kaydedilebilir bileşen eklemek için, ilgili sınıfın ISaveable arayüzünü uyguladığından emin olun ve ardından bileşeni Inspector üzerinden SaveLoadManager'a sürükle bırak yöntemiyle ekleyin.
- **Üretim Mekanikleri:**
  ProductionNode, hem oyun sırasında hem de offline modda dinamik olarak öğe üretimini hesaplar. Bu mekanizma, farklı üretim hızları veya türleri için kolayca genişletilebilir.
- **Test Edilebilirlik:**
  Ana ekrandaki test tuşları, XP, seviye, para ve envanter öğeleri gibi değerlerin dinamik olarak değiştirilebilmesine olanak tanır. Yapılan değişiklikler, oyundan çıkıldığında kaydedilir ve yeniden girişte geri yüklenir.
- **Mobil Uyumluluk:**
  Ölçeklenebilir canvas, farklı ekran boyutlarına uyum sağlayacak şekilde ayarlanmıştır. Mevcut UI düzeni, yalnızca test amaçlı olup, nihai kullanıcı deneyimi için özelleştirilebilir.

6. Sonuç

Proje, Unity kullanılarak modüler tasarım prensiplerine dayalı sağlam bir mimari sunmaktadır.

- **ISaveable Arayüzü:**
  Tüm bileşenlerde kaydet/yükle işlemlerini standartlaştırarak sistemin genişletilebilir ve bakımının kolay olmasını sağlar.
- **SaveLoadManager:**
  Inspector üzerinden sürükle bırak yöntemiyle eklenen bileşenlerin durumunu merkezi olarak JSON dosyasında saklar ve geri yükler.
- **Test & Mobil Uyumluluk:**
  Entegre test tuşları ve ölçeklenebilir canvas, geliştirme sürecinde esneklik sağlar ve nihai kullanıcı deneyimini optimize eder.

Bu dokümantasyon, projenin genel yapısını, bileşenlerin işlevselliğini ve uygulanan tasarım prensiplerini detaylı olarak açıklamaktadır. Ek bilgi veya değişiklik gereksinimi olursa, lütfen iletişime geçiniz.

7. URL

Test Videosu:
https://youtu.be/bpMEEzqT4wk

Github:
https://github.com/YAMTAR-98/InventorySaveAndAutoGenerateSystem.git

Portföy:
https://muhammednafikirman.info/