Stack Attack Clone - Case Dökümantasyonu

Performans Odaklı Geliştirmeler

- Düzenli olarak oluşturulan tüm objeler **Pooling** tekniği ile yönetilmektedir.
  Bu sayede, ihtiyaç duyulmadıkça yeni nesneler oluşturulmaz veya mevcutlar **destroy** edilmez.

- **Pooling** sistemi aşağıdaki nesneler için aktif olarak kullanılmaktadır:

    - Tüm **Projectile** nesneleri

    - **Görsel efektler (VFX)**

    - **Kırılabilen hücreler / Stack'ler**

- **Viewport** dışına çıkan oyun elementleri (varsayılan davranış olarak, ayrı bir şekilde handle edilebilir) otomatik olarak ait oldukları pool'a geri dönerler.

    - Projectile'lar

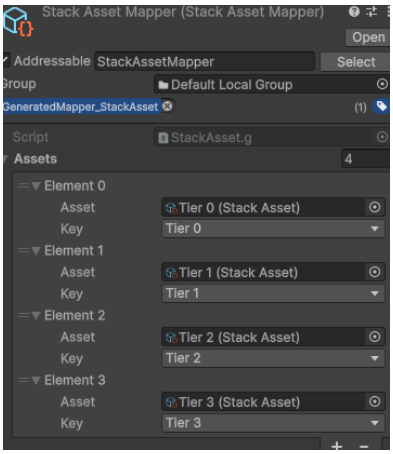
    - Stack'ler *(Rotator'lar ayrı olarak handle edilir)*

Asset Management

- **Single Scriptable Asset** (tarafımdan geliştirilmiş özel bir araç) sayesinde, proje genelinde tek örnekli Scriptable Object'lere **TypeName.Default** şeklinde erişim sağlanmaktadır.
  Bu sistem, altyapısında **Addressables** kullanmaktadır.

- **O2 Auto Mapper** (yine tarafımdan geliştirilen bir araç – GitHub: OxygenButBeta/O2-Unity-Auto-Asset-Mapper)
  Bu araç sayesinde, proje genelinde hedef asset'lere **editor üzerinden drag & drop** yapmaya gerek kalmadan otomatik bir **map** oluşturulmaktadır.
  Geliştirme sürecinde bu map üzerinden **enum** tabanlı erişim sağlanır.

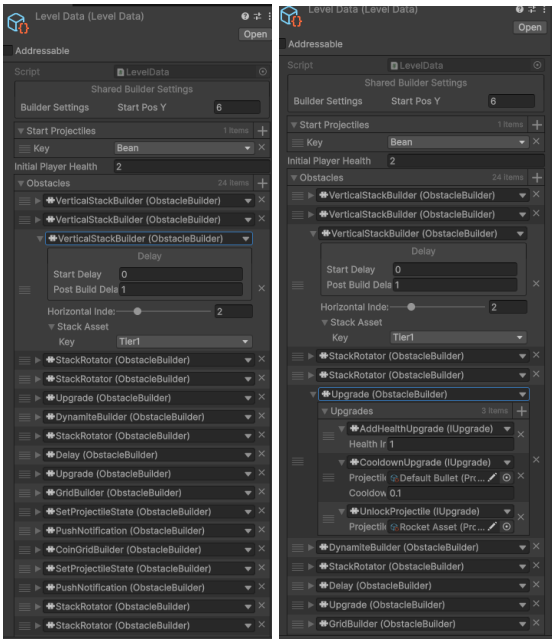
- **Örnek:**



Bu map yapısı sayesinde **Level Tool** içerisinde Stack asset'leri bir **enum** üzerinden tek tıkla seçilebilmektedir.
Böylece hem geliştirme kolaylığı sağlanmakta hem de atanmayan veya eski referansların kalması gibi hataların önüne geçilmektedir.

Level Tooling

- Level geliştirmeyi kolaylaştırmak amacıyla oluşturulmuş özel bir sistemdir.
Bu yapı **SerializeReference** tabanlı olup, **Serialize Reference Editor** desteği sunmaktadır.
(Şu anda GUI tarafı **Odin Inspector** tarafından sağlanmaktadır; ancak bağımsız araçlarla da çalışabilir.)



- **ScriptableObject** tabanlı **Level Data** yapısında, önceden tanımlı elemanların eklendiği bir liste üzerinden tüm level'lar oluşturulabilmektedir.
  Bu sayede **level designer'lar**, tek bir satır kod yazmadan yalnızca editör üzerinden listeye eleman ekleyerek yeni level'lar oluşturabilirler.

- Tüm level elementleri, temel sınıf olan **ObstacleBuilder** sınıfından türetilmektedir ve doğrudan kullanılabilirler.

Notlar

- Üçüncü parti bir **Dependency Injection** kütüphanesi kullanma gereği duymadım.
  Oyunun yapısı tek sahnede geçen basit bir kurgudan oluştuğu için, bağımlılıkların manuel olarak yönetilmesi bir dezavantaj yaratmamaktadır.

- **Odin Inspector**, endüstri standardı bir araç olarak tercih edilmiştir.
  Daha düzenli ve kolay **Editor Scripting** imkânı sağlamaktadır.
  Editor özelleştirmeleri dışında zorunlu olduğu bir alan bulunmamaktadır.

- **Stack**'lerin üst üste gelmesi durumunda, **Draw Layer** ayarları hem kendi aralarında hem de diğer stack'lerle olacak şekilde yapılandırılmıştır.
  Böylece birbirlerini engellemeden doğru şekilde görüntülenirler.

- **Stack efektleri**, **SerializeReference** ile bağımsız biçimde tanımlanmıştır ve **DoTween** kullanılarak sağlanmaktadır.
  Bu sayede animasyonlar kolayca eklenip çıkarılabilir.