**Proje ve Kodların Genel Hiyerarşisi**

Bu projede modüler ve sürdürülebilir bir yapı kurmaya özen gösterdim. Kod yapısını **SOLID prensiplerine** göre oluşturmaya çalıştım ve bağımlılık yönetiminde **Zenject** framework’ünden faydalandım. Dosya ve klasör yapısını da olabildiğince domain bazlı şekilde düzenledim.

**1. ScriptableObjects**

Oyun içi verileri (kule, düşman, dalga vs.) doğrudan koddan bağımsız şekilde kontrol edebilmek için **ScriptableObject** yapısını kullandım.

* **EnemyData** klasöründe iki farklı düşman türü için veri tanımları var:  
  NewAttackEnemyData, NewRunnerEnemyData
* **TowerData** klasöründe iki farklı kuleye ait veri var:  
  Tower1Data, Tower2Data
* **WaveData** ise düşman dalgalarının kontrolünü sağlıyor.

**2. Scripts**

Kodları farklı sorumluluklara göre parçaladım:

**a) Core:**  
Oyun akışını ve genel sistemleri yöneten sınıflar burada.

* GameManager, MapManager, UIManager, WaveManager

**b) Data:**  
ScriptableObject’lerin kullandığı veri sınıfları burada yer alıyor.

* EnemyData, AttackEnemyData, RunnerEnemyData, TowerData, WaveData

**c) Enemies:**  
Düşmanlarla ilgili tüm davranışlar ve türler burada.

* EnemyBase, RunnerEnemy, AttackEnemy, EnemySpawner, IEnemy, EnemyType

**d) Health:**  
Can sistemini yöneten yapı.

* HealthSystem ve UI tarafında HealthBar

**e) Installers:**  
Zenject ile dependency'leri bağladığım yerler.

* GameInstaller (Scriptable Object üzerinden),
* SceneInstaller (sahne üzerindeki referansları bind eder)

**f) Projectile:**  
Kulelerin gönderdiği mermiler ve havuzlama sistemi.

* Projectile, ProjectilePool

**g) Towers:**  
Kulelerin mantığı ve yerleştirme sistemi burada.

* TowerBase, TowerPlacementArea, TowerSelectionPanel
* Ayrıca saldırı stratejileri: SingleTargetAttackStrategy, MultiTargetAttackStrategy

**Kullandığım Framework: Zenject**

Bu projede **Zenject** kullanmayı tercih ettim. Sebebi; bağımlılıkları dışarıdan inject edebilmek, daha esnek ve test edilebilir bir yapı kurabilmekti. Ayrıca sahne üzerindeki referansları da kolayca yönetebilmek adına Zenject oldukça pratik.

**Zenject Kullanımı**

* GameInstaller: TowerData, WaveData, ProjectilePool, HealthBar gibi değerleri sahne dışı ScriptableObject üzerinden bind ediyorum.
* SceneInstaller: Sahne üzerindeki MapManager, Camera, EnemySpawner gibi objeleri burada bağlıyorum.
* Kule, düşman, sağlık sistemi gibi sınıflarda **field injection** ya da **constructor injection** kullanarak Zenject üzerinden bağımlılıkları alıyorum.
* TowerSelectionPanel, WaveManager, EnemySpawner gibi sistemlerin birbirinden kopuk şekilde çalışmasını sağlayan şey de yine Zenject oldu.

Zenject sayesinde sistemler arasında doğrudan bir bağlantı kurmam gerekmediği için kod daha okunabilir, sürdürülebilir ve kolay test edilebilir hâle geldi.

**Proje Nasıl Çalıştırılır.**

**ScriptableObject klasörün içindeki EnemyData, TowerData ve WaveData objeleri düzenlendikten sonra play tuşuna basılır. Oyun başladıktan sonra sahnede duvarların üstünde mavi renkli dikdörtgen alanlar vardır o alanlara tıklandığında kule kurma ekranı karşımıza çıkar. Bu ekran kule sayısına göre otomatik güncellenmektedir. UI’ da seçilen kule oyunda aktif olur ve npcler ile savaş başlar. Dalga bittikten sonra WaveDatada belirtilen Preparation Duration süresi kadar bekleyip sonraki dalgayı başlatır.**