2D Uzay Nişancı (Shooter) Oyunu – Ders Raporu

Bu rapor; düşman ve bonus (power-up) üretimi, merminin hareketi, düşmanın davranışı ve çarpışma mantığı üzerine işlediğimiz dersi özetler. Kodlar C# ile yazılmıştır ve Unity'nin temel bileşenleri olan GameObject, Transform, Collider, Rigidbody2D, Coroutine ve Instantiate/Destroy gibi kavramları kullanır.

Github:https://github.com/Oxhi1/Oyun-programlama-haftalik-kayit

---

1) Oyunun Genel Fikri

- Ekranın üst kısmından düzenli aralıklarla "düşmanlar" (Enemy) ve "bonuslar" (ör. Triple Shot Power-up) aşağı doğru düşer.

- Oyuncu uzay gemisi (Player) mermi (Laser) ateşler. Mermi yukarı doğru gider, ekrandan çıkarsa kaybolur.

- Mermi düşmana çarparsa mermi ve düşman yok olur.

- Oyuncu bonusa çarparsa oyuncuya bir süreliğine güç verir (ör. üçlü atış).

- Oyuncu ölünce yeni düşman/bonus üretilmesi durur.

Ekran görüntüsü : 

2) Kullanılan Scriptler ve Sorumlulukları

A) SpawnManager.cs

- Görev: Belirli aralıklarla sahneye düşman ve bonus prefab'larını "spawn" (üretmek) eder.

- Önemli noktalar:

 - Start() içinde iki coroutine başlatır:

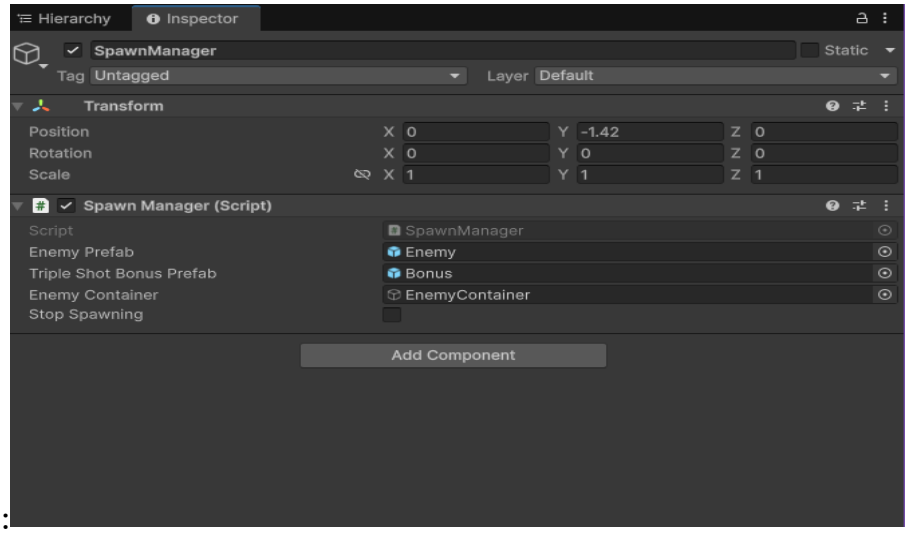
  - SpawnEnemyRoutine(): 5 saniyede bir düşman üretir.

  - SpawnBonusRoutine(): 3–7 saniye arasında rastgele sürede bonus üretir.

 - stopSpawning değişkeni true olursa üretimi durdurur (ör. oyuncu ölünce).

 - Üretilen düşmanları düzenli tutmak için EnemyContainer adlı bir boş (Empty) GameObject'in altına atar.

Ekran görüntüsü :



B) LaserMove.cs

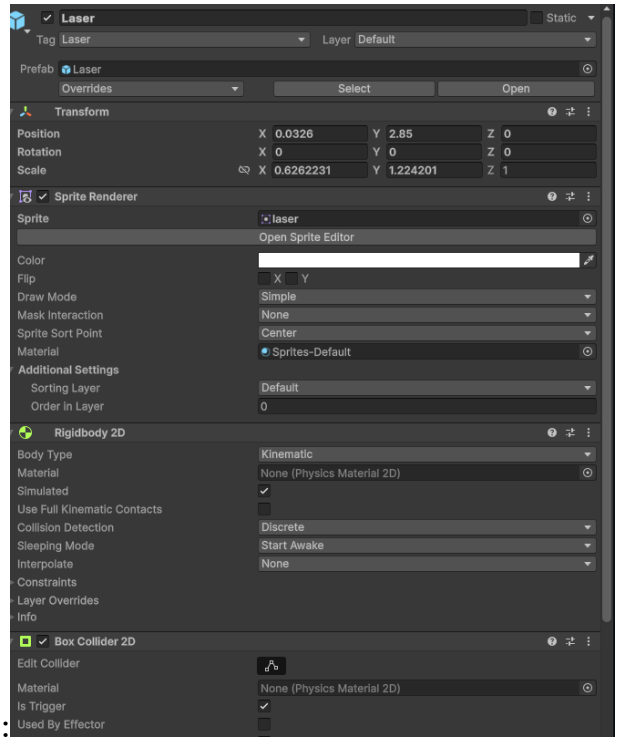
- Görev: Mermiyi (Laser) yukarı hareket ettirmek ve ekrandan çıkınca yok etmek.

- Önemli noktalar:

 - Update() içinde `transform.Translate(Vector3.up * speed * Time.deltaTime)` ile mermiyi hareket ettirir.

 - Y konumu > 7 olunca kendini (ve varsa parent'ını) Destroy eder. Bu, sahnede gereksiz obje birikmesini engeller.

Ekran görüntüsü :



C) Enemy.cs

- Görev: Düşmanı aşağı hareket ettirmek, ekranın altından çıkınca tekrar üstte rastgele x konumundan sahneye sokmak ve çarpışmaları yönetmek.

- Önemli noktalar:

 - Update() içinde `Translate(Vector3.down * speed * Time.deltaTime)`.

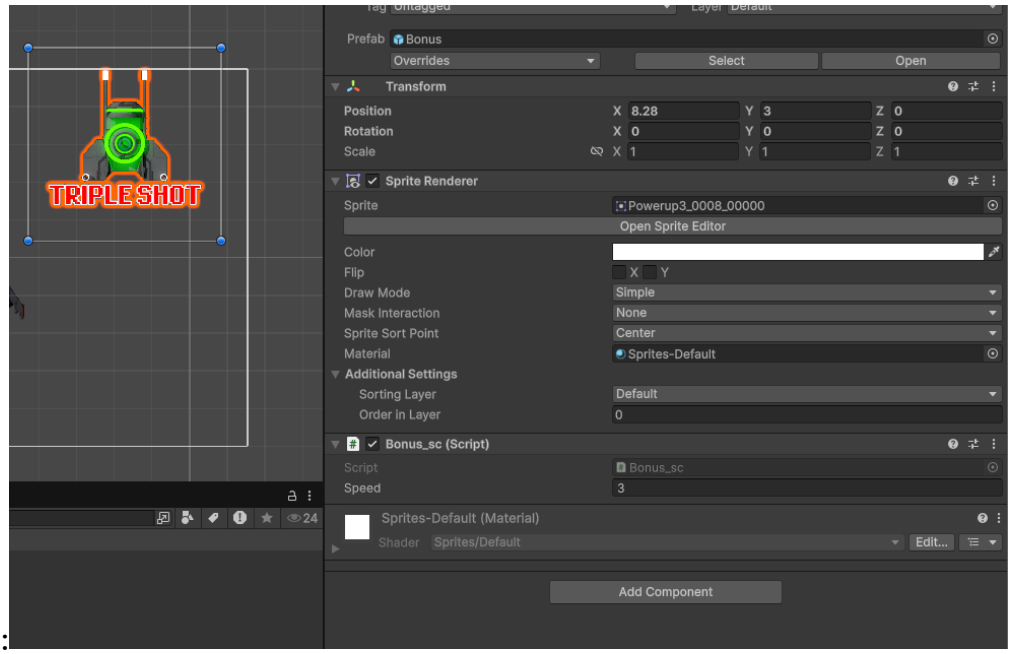
 - Y < -5.5 olduğunda `new Vector3(Random.Range(-9.5f, 9.5f), 7.4f, 0)` konumuna sıfırlar (yukarıdan tekrar düşer).

 - OnTriggerEnter2D(Collider2D other):

  - other.tag == "Player" ise PlayerController.Damage() çağırır, düşmanı yok eder.

  - other.tag == "Laser" ise lazeri ve düşmanı yok eder.

Ekran görüntüsü:



D) Bonus_sc.cs (Triple Shot Bonus)

- Görev: Bonus objesi aşağı düşer, oyuncuya temas edince PlayerController.TripleShotActive() çağırır ve kendini yok eder.

- Önemli noktalar:

  - Update() içinde `Translate(Vector3.down * speed * Time.deltaTime)`.

  - Y < -5.8 olunca Destroy(this.gameObject).

  - OnTriggerEnter2D() içinde Player ile temas ederse bonusu uygular.

3) Unity Kurulumu – Adım Adım

Bu kısım, hiç bilmeyen biri için “ne, nasıl yapılır?” rehberidir.

1) Sahne Hazırlığı

- Yeni bir 2D sahne oluşturun.

- Player (oyuncu gemisi) objesini yerleştirin.

- Camera ve arka planı ayarlayın.

2) Tag ve Layer Ayarları

- Tags üzerinden “Player” ve “Laser” tag’lerini oluşturun (Edit > Project Settings > Tags and Layers).

- Player objesine “Player”, mermi prefab’ına “Laser” tag’ini atayın.

3) Collider ve Rigidbody2D

- Çarpışmaların algılanması için ilgili objelere 2D collider ekleyin:

- Player: ör. CapsuleCollider2D veya BoxCollider2D (Is Trigger: tercihinize göre; bu projede OnTriggerEnter2D kullanıldığı için Is Trigger önerilir).

- Enemy: ör. BoxCollider2D (Is Trigger: true).

- Laser: ör. BoxCollider2D (Is Trigger: true).

- Bonus: ör. CircleCollider2D (Is Trigger: true).

- Rigidbody2D ekleyin:

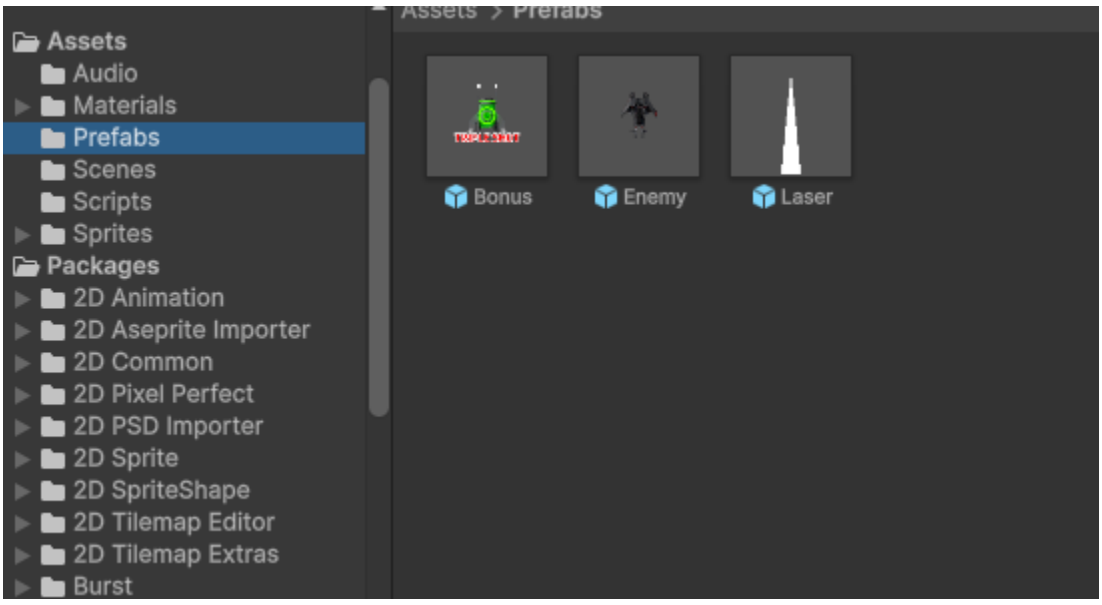
- Çoğu durumda “Body Type: Kinematic” ve Gravity Scale = 0 kullanılır (hareketi script yönetiyor).

4) Prefab’ları Hazırlama

- Enemy, Laser, TripleShot Bonus gibi objelerden prefab oluşturun (Project penceresine sürükleyip bırakın).

- EnemyContainer adında boş bir GameObject oluşturun (sahne düzeni için).

Ekran görüntüsü :



5) SpawnManager Kurulumu

- SpawnManager adında boş bir GameObject oluşturun ve SpawnManager.cs scriptini ekleyin.

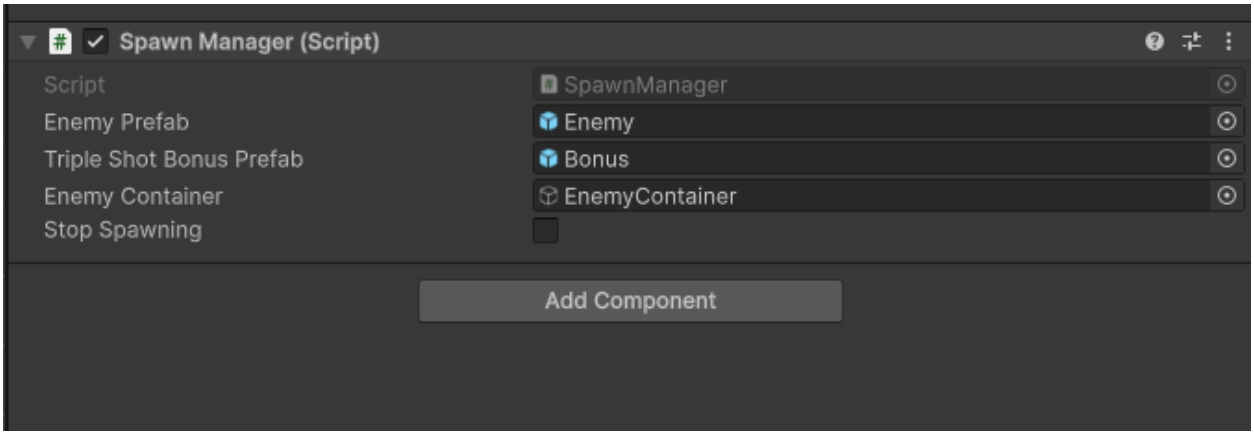
- Inspector'da:

 - enemyPrefab alanına Enemy prefab'ını sürükleyin.

 - tripleShotBonusPrefab alanına TripleShot Bonus prefab'ını sürükleyin.

 - enemyContainer alanına sahnedeki EnemyContainer objesini sürükleyin.

Ekran görüntüsü :



6) PlayerController Entegrasyonu

- Enemy ve Bonus script'leri PlayerController'a bağlanıyor:

 - Enemy.cs: `playerController.Damage();`

 - Bonus_sc.cs: `playerController.TripleShotActive();`

- Bu yüzden Player'da bir PlayerController.cs olmalı; içinde en azından Damage() ve TripleShotActive() metodları bulunmalı.

- Oyuncu öldüğünde SpawnManager.OnPlayerDeath() çağrılarak spawn işlemleri durdurulmalı.

7) Oyun Sınırları ve Değerler

- LaserMove: speed varsayılan 3f; y > 7 olunca yok ediliyor. Kamera ortalaması ve oyun alanına göre sayıları uyarlayabilirsiniz.

- Enemy: y < -5.5 iken tekrar yukarı alınır; x: -9.5 ile 9.5 arası rastgele.

- Bonus: y < -5.8 iken yok olur.

4) Koddaki Önemli Kavramlar – Basit Açıklamalar

- Instantiate(prefab, position, rotation): Sahneye yeni bir nesne kopyası oluşturur.

- Destroy(gameObject): O nesneyi sahneden siler (bir süre sonra bellekten de temizlenir).

- Coroutine ve IEnumerator:

  - `StartCoroutine(...)` ile zaman kontrollü tekrarlı işler yapılır.

  - `yield return new WaitForSeconds(5.0f)` gibi beklemeler sağlanır.

- Random.Range(min, max):

  - float için min ve max kapsayıcıdır (dahil).

  - int için üst sınır hariçtir (ör. Random.Range(3, 8) => 3,4,5,6,7 döner).

- OnTriggerEnter2D(Collider2D other):

  - “Is Trigger” aktif collider’lar arasında tetiklenen temas olayını yakalar.

  - `other.tag` ile temas edilen objenin Tag’ine göre davranış değiştirilir.

- transform.Translate(v):

  - Objenin konumunu çevirir. `Time.deltaTime` ile çarpmak, FPS’ten bağımsız sabit hız sağlar.

##) Sonuç

Bu derste, sahneye periyodik olarak düşman/bonus eklemeyi (Coroutine + Instantiate), mermiyi hareket ettirmeyi ve çarpışmaları yönetmeyi öğrendik. Oyuncu ölünce spawn'ı durdurma gibi oyun döngüsünün temelini oluşturan kontrol akışlarını gördük. Bu altyapı, basit ama genişletilebilir bir 2D shooter oyununun temelini oluşturur.
