BLM0364 OYUN PROGRAMLAMA – HAFTA 8 RAPORU

Yasin Ekici

21360859029

Rapor İçeriği

- **Kullanıcı ara yüzü geliştirme.**
- **Puan sistemi**
- **Can sistemi**
- **Game Over yazısı ve oyunu yeniden başlatmak için yönerge.**
- **Ana menü**

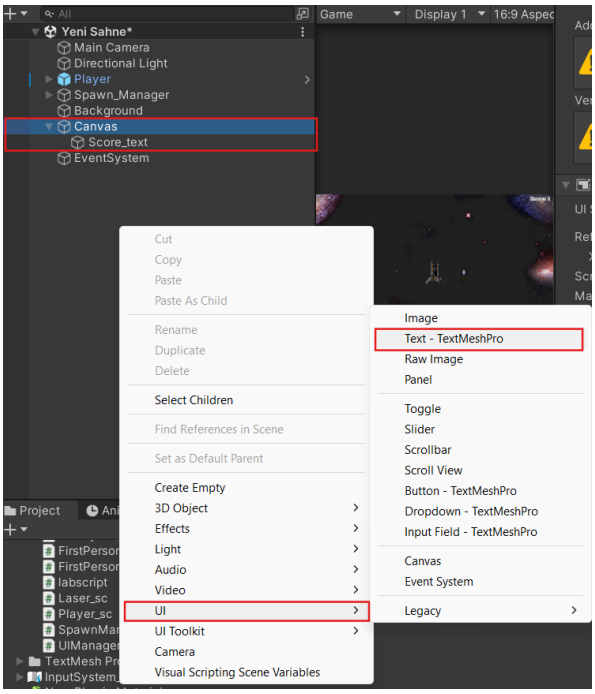
Bu raporda anlatılan kodlar için: https://github.com/YasinEkici/game-programming

Asset'ler için:
https://e.pcloud.link/publink/show?code=XZhFVJZuzLw5aWEuizh3kDHE43iNVyak6l7

1. Puan Sistemi

- Bir düşmanı öldürdüğünüzde belirli bir miktar puan alın.
- Sahip olduğunuz puanları ekranda görüntüleyin.
- Bir UI Canvas ve Metin oluşturun.

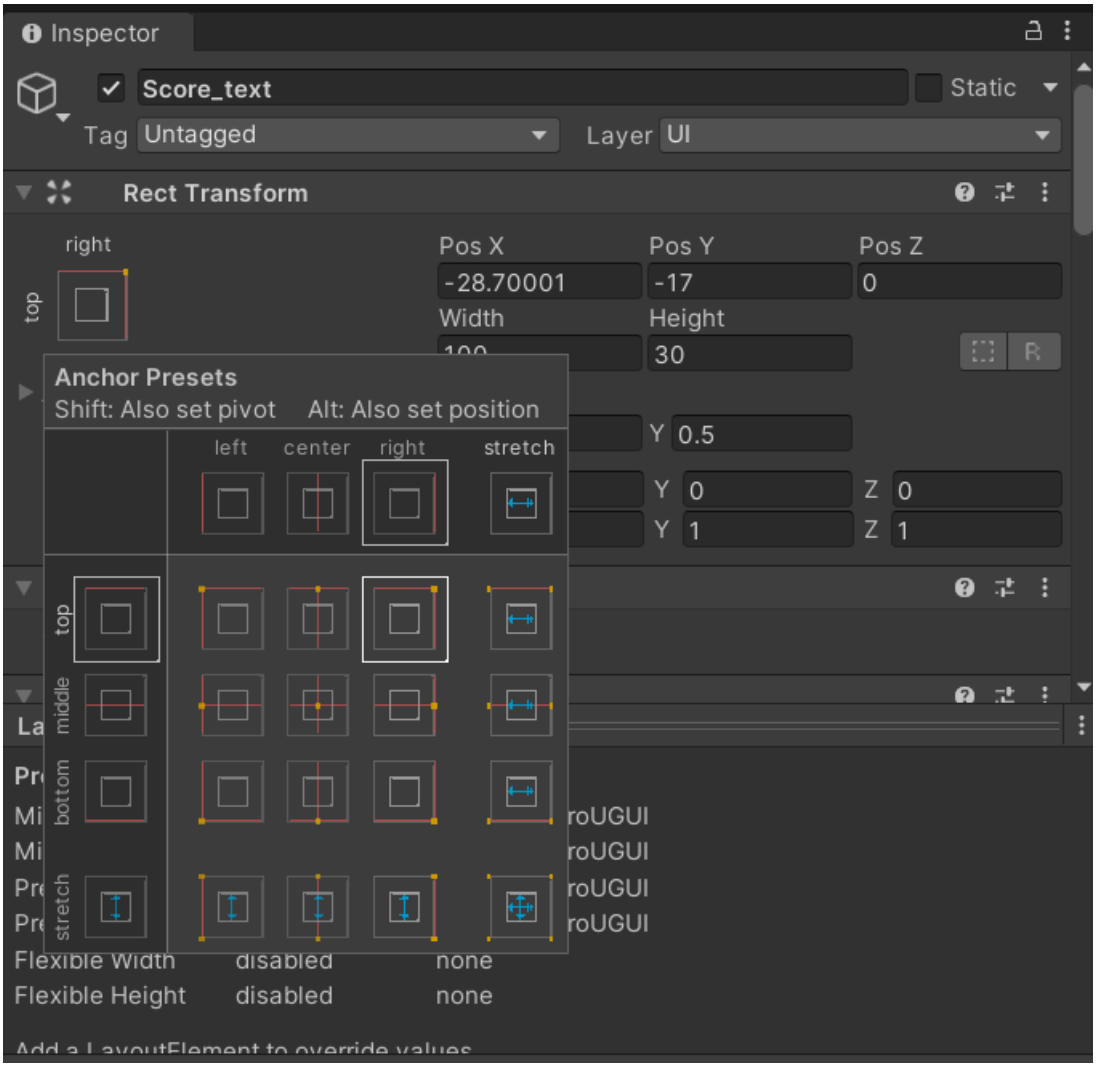
1. Hiyerarşi penceresine sağ tıklayın ve UI -> Text seçin.



2. UI Canvas otomatik olarak eklenir ve Metin Canvas'ın alt öğesi olur.
3. Olay sistemi (Event System) otomatik olarak eklenir.

- Canvas, ekrandaki beyaz öğe olup UI nesnelerini barındırır.
- Event System, UI ile etkileşim için kullanılır.
- Metin nesnesinin adını “Score_text” olarak değiştirin.
- Rengi beyaz yapın.
- Yazı tipi boyutunu 20 olarak ayarlayın.
- Yazıyı “Score: 0” olarak değiştirin.
- UI öğesini ekranın üst sağ köşesine yerleştirin.

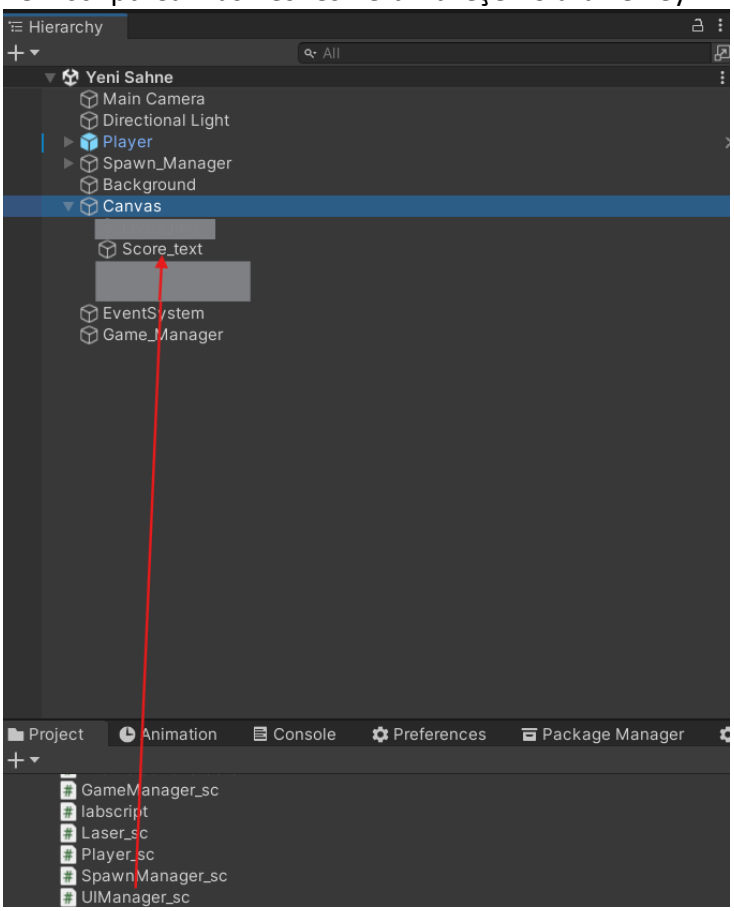
1. x = -100, y = -50 ayarlayın.
2. Genişlik = 100, Yükseklik = 30 ayarlayın.
3. Doğru ölçeklendirme için Rect Transform bileşenini kullanın ve üst sağ anker ön ayarını seçin.



- Nesnenin boyutunu ekran boyutuyla ölçeklendirin.

1. Canvas'ı seçin. Canvas Scaler bileşeninde Ölçek Modunu “Constant Pixel Size” yerine “Scale with Screen Size” olarak değiştirin.

- “UIManager_sc” adında yeni bir C# scripti oluşturun.
- Yeni scripti Canvas nesnesine bir bileşen olarak ekleyin.



- Oyuncunun puanını takip etmek için özel bir değişken oluşturun.

  Player nesnesinde: [SerializeField] int score = 0;

- Player scriptinde puan artırmak için bir genel metod oluşturun.

```
public void UpdateScore(int points){

    score += points;

}
```

- Enemy scriptinde, Enemy’nin Laser tarafından vurulup vurulmadığını kontrol edin.
- Enemy scripti Player scriptiyle iletişim kurabilmelidir.
- Player scripti için özel bir handler değişkeni oluşturun.
- Start fonksiyonunda Player bileşenini bulup Player script bileşenini alın.

```
Player_sc player = GameObject.Find("Player").GetComponent<Player_sc>();
```

- Enemy nesnesini yok etmeden önce Player puanını artırın.

```
    void OnTriggerEnter2D(Collider2D other){
  if(other.tag == "Player"){
    Player_sc player = other.transform.GetComponent<Player_sc>();
    if(player != null){
      player.Damage();
    }
    Destroy(this.gameObject);
  }
  else if (other.tag == "Laser"){
    Destroy(other.gameObject);
    if(player != null){
      player.UpdateScore(10);
    }
    Destroy(this.gameObject);
  }
}
```

- Skoru ekranda güncellemek için;
- Player scripti, UI Manager scriptine erişebilmelidir;
- UI Manager scripti için Player scriptinde özel bir handler değişkeni oluşturun.

```
UIManager_sc uiManager_sc;
```

- Start fonksiyonunda UI Manager script bileşenini bulup null olup olmadığını kontrol edin;

```
uiManager_sc = GameObject.Find("Canvas").GetComponent<UIManager_sc>();
    if(uiManager_sc == null){
```

```
            Debug.LogError("UI Manager bulunamadi");
        }
```

- AddScore fonksiyonunu güncelleyin;

```
public void UpdateScore(int points){
    score += points;
    if(uiManager_sc != null)
    {
       uiManager_sc.UpdateScoreTMP(score);
    }

}
```

- UI Manager tarafından skorun ayarlanması için yeni bir fonksiyon gerekli. Adı UpdateScoreTMP olacak.
- UI Manager scriptini güncelleyin.

    1. Canvas üzerindeki Metin UI öğesine erişin.
    2. UI öğelerine erişebilmek için önce UnityEngine.UI kütüphanesini ekleyin.
    3. Özel bir Metin handler değişkeni oluşturun ve kaydedin.

        ```
        [SerializeField] TextMeshProUGUI scoreTMP;
        ```

    4. Hiyerarşide Score_text'i UI_Manager scriptindeki Score Text alanına sürükleyip bırakın.

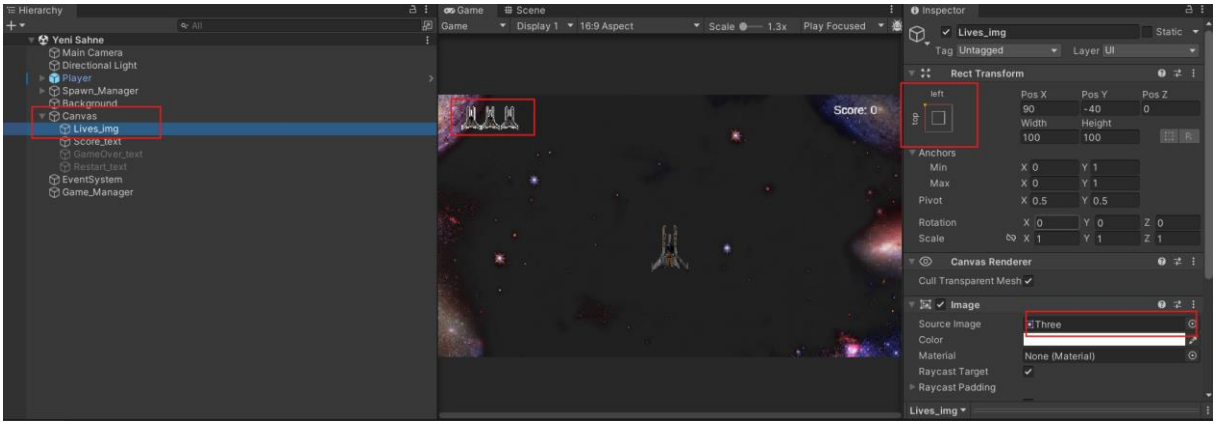
- Başlangıçta ekranda başlangıç skor değerini ayarlayın.

    Start fonksiyonuna: `scoreTMP.text = "Score: " + 0;`

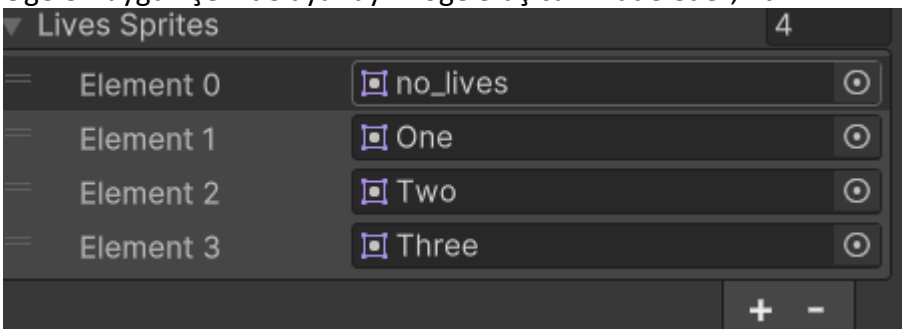
- Player scripti, skoru güncellediğinde UI görüntüsünü güncellemek için UpdateScoreTMP fonksiyonunu tanımlayın.

    ```
    public void UpdateScoreTMP(int score){
        scoreTMP.text = "Score: " + score;
    }
    ```

2. Can Sistemi

- **Oyuna varsayılan olarak üç can ile başlayın.**
- **Ekranda can göstergesi olarak bir resim bileşeni ekleyin.**
- **Canvas'a sağ tıklayın ve bir Resim bileşeni oluşturun.**
- **Resim bileşenine "Lives_img" adını verin.**
- **Varsayılan olarak "Three" can sprite'ını ayarlayın.**
- 

- **Preserve Aspect seçeneğini aktif edin.**
- **Üst sol ankere sabitleyin ve x = 90, y = -40 pozisyonlarını ayarlayın.**
- UI Manager scriptinde bir can sprite'ı dizisi tanımlayın;
  [SerializeField] private Sprite[] livesSprites;
- Canvas'ta dizinin boyutunu 4 olarak ayarlayın.
  lives_img.sprite = livesSprites[3];
- Öğeleri uygun şekilde ayarlayın: öğe 3 üç canı ifade eder, vb.



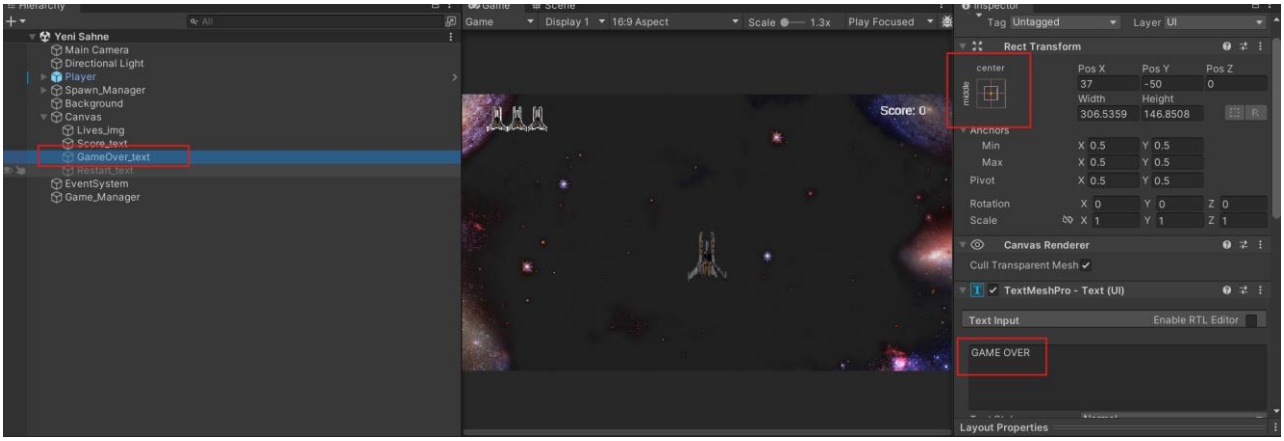
- Ekranda görüntülenen Resim bileşenine erişmek için bir değişken tanımlayın.
- Bu değişkeni Canvas üzerinden atayın;

  [SerializeField] private Image lives_img;

- Can sprite'ını güncellemek için bir fonksiyon tanımlayın.
  public void UpdateLivesIMG(int lives){
  lives_img.sprite = livesSprites[lives];
  }
- UpdateLivesIMG fonksiyonunu Player scriptinde can güncellendiğinde çağıracağız
  Damage fonksiyonu içerisinde: if(uiManager_sc != null){
  uiManager_sc.UpdateLivesIMG(lives);
  }

3. Game Over Ekranı

- Canvas altında bir Metin öğesi oluşturun.
- Bu öğeye “GameOver_text” adını verin.
- Metni “GAME OVER” olarak ayarlayın.
- Rengi beyaz yapın
- Yazı tipi boyutunu 50 olarak ayarlayın.
- Metni ekranın ortasına yerleştirin.
- Varsayılan olarak “Game Over” metni kapalı olacak.
- Inspector'dan nesneyi devre dışı bırakın.



- UI Manager scriptinde “Game Over” metnine referans için özel bir değişken tanımlayın.

```
[SerializeField] TextMeshProUGUI gameOverTMP;
```

- Değişkenin değerini Canvas altında ayarlayın.
- Metin nesnesinin başlangıçta devre dışı olduğundan emin olun.
  Start fonksiyonuna: gameOverTMP.gameObject.SetActive(false);
- Oyuncunun canı bittiğinde nesneyi etkinleştirin.
- UpdateLivesIMG fonksiyonunda mevcut can değerini kontrol edin.
- Mevcut can 1’den az olduğunda metin nesnesini aktif hale getirin;

```
public void UpdateLivesIMG(int lives){

    lives_img.sprite = livesSprites[lives];

    if(lives == 0){

       gameOverTMP.gameObject.SetActive(true);

    }
```

```
}
```

- Game Over Flicker Efekti;
    - UI Manager scriptinde bir coroutine tanımlayın.
    - Game Over metnini açıp kapatın.
    - Metni “Game Over” olarak ayarlayın.
    - Yarım saniye bekleyin.
    - Metni boş yapın.
    - Yarım saniye bekleyin.

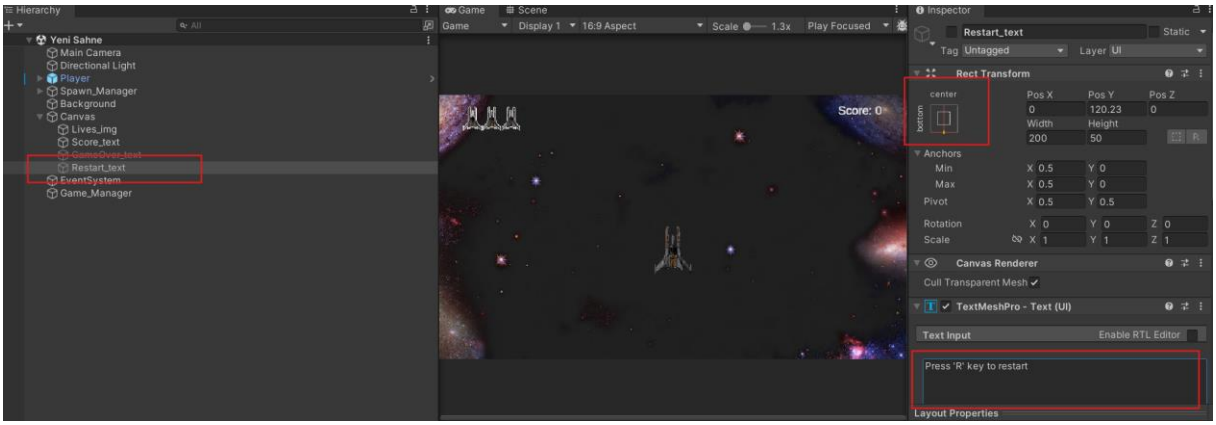
```
IEnumerator GameOverFlickerRoutine(){
    while(true){
        gameOverTMP.text = "GAME_OVER";
        yield return new WaitForSeconds(0.5f);
        gameOverTMP.text = "";
        yield return new WaitForSeconds(0.5f);
    }
}
```

- Oyun bittiğinde yanıp sönme coroutine fonksiyonunu çağırın.

```
public void UpdateLivesIMG(int lives){

lives_img.sprite = livesSprites[lives];

if(lives == 0){

gameOverTMP.gameObject.SetActive(true);

}

}
```

4. Oyunu Yeniden Başlatma

- Oyunu yeniden başlatma talimatlarını vermek için Canvas altında yeni bir Metin nesnesi oluşturun.
- Metin nesnesine “Restart_text” adını verin.
- Metni “Press 'R' key to restart” olarak ayarlayın.
- Rengi beyaz yapın.
- Yazı tipi boyutunu 28 olarak ayarlayın.
- Metni bir satır sığacak şekilde genişletin ve ortalayın.
- Metin oyun nesnesini devre dışı bırakın, böylece başlangıçta kapalı olur.



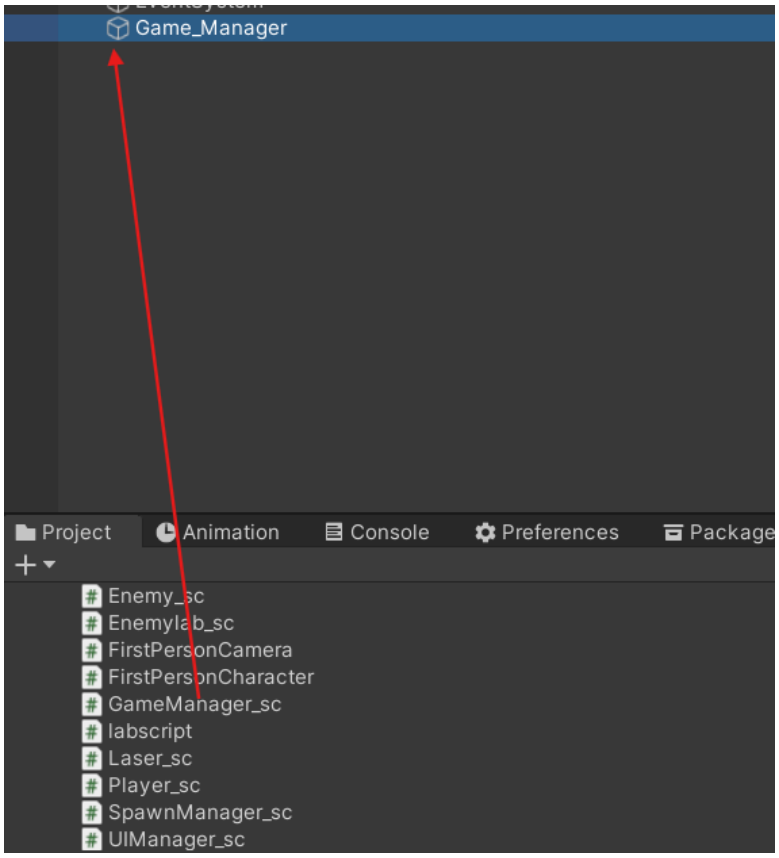
- Canvas'ta yeni bir özel değişken tanımlayın ve Restart metni için referans oluşturun.
- Değişkeni Inspector'dan ayarlayın.
- [SerializeField] TextMeshProUGUI restartTMP;
- Oyuncunun canı bittiğinde bu metni etkinleştirin.
- Bunu yapmak adına bir fonksiyon tanımlayacağız;

```
void GameOverSequence(){
if(gameManager_sc != null){
gameManager_sc. GameOver();
}
gameOverTMP.gameObject.SetActive(true);
restartTMP.gameObject.SetActive(true);
StartCoroutine(GameOverFlickerRoutine());
}
```

- Şimdi UpdateLivesIMG fonksiyonunu düzenleyelim;

```
public void UpdateLivesIMG(int lives){
    lives_img.sprite = livesSprites[lives];
    if(lives == 0){
      GameOverSequence();
    }
  }
```

- Yeni bir boş nesne oluşturun ve adına "Game_Manager" deyin.
- "GameManager_sc" adında yeni bir C# scripti oluşturun ve Game_Manager nesnesine bileşen olarak ekleyin.



- Game Manager scriptinde oyunun bitip bitmediğini kontrol eden bir Boolean değişken tanımlayın;

```
bool isGameOver;
```

- Boolean değişkeni true olarak ayarlamak için genel bir fonksiyon tanımlayın;

```
public void GameOver(){
    isGameOver = true;
}
```

- UI Manager scriptinde Game Manager için bir handler değişkeni tanımlayın;

```
GameManager_sc gameManager_sc;
```

- Start fonksiyonunda bu componenti bulun;

```
gameManager_sc =
GameObject.Find("Game_Manager").GetComponent<GameManager_sc>();
```

- Game Manager scriptinin GameOver fonksiyonunu GameOverSequence'da çağırın;

```
void GameOverSequence(){
    if(gameManager_sc != null){
```

```
        gameManager_sc. GameOver();
    }
    gameOverTMP.gameObject.SetActive(true);
    restartTMP.gameObject.SetActive(true);
    StartCoroutine(GameOverFlickerRoutine());
  }
```

Game Manager scriptinde SceneManagement kütüphanesini dahil edin ve sahne yönetim fonksiyonlarına erişim sağlayın;

```
using UnityEngine.SceneManagement;
```

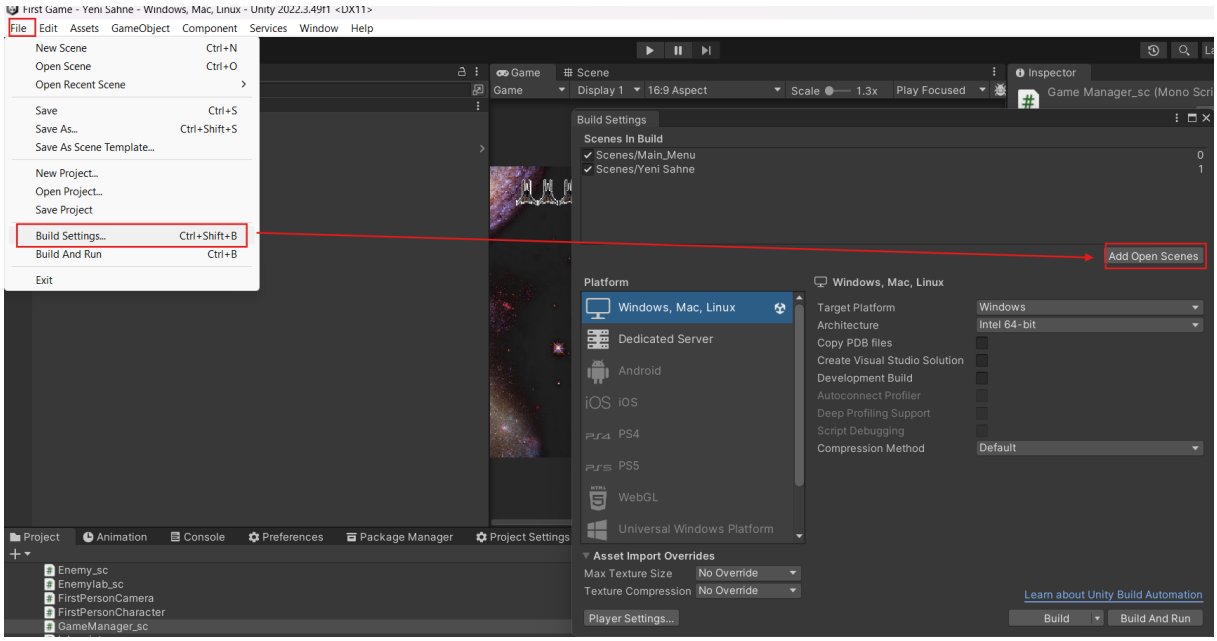
Tuş girişini kontrol edin ve koşullar sağlanıyorsa sahneyi yeniden yükleyin.

```
void Update(){
    if(Input.GetKeyDown(KeyCode.R) && isGameOver == true){
      SceneManager.LoadScene("Yeni Sahne");
    }
  }
```

Sahne adı veya diziniyle sahne yüklenebilir.

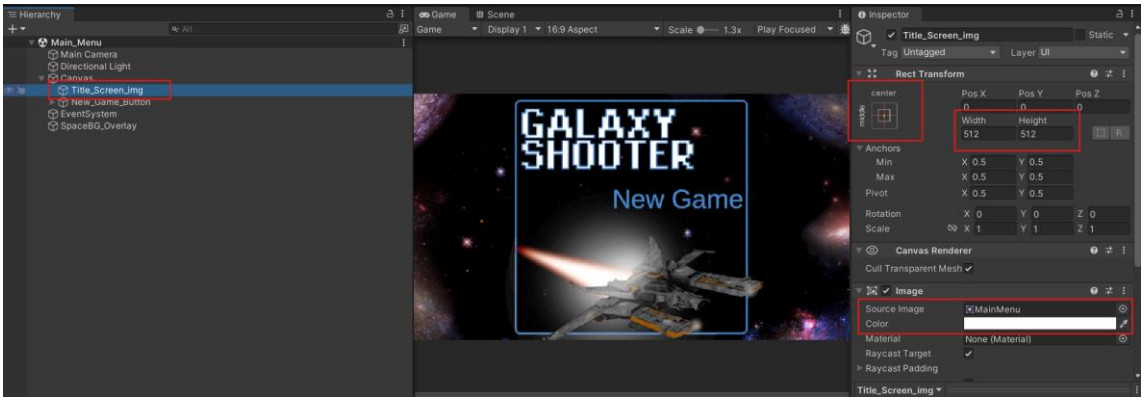
Daha hızlı olması için dizin kullanılabilir.

File -> Build Settings -> Add Open Scenes seçeneğini kullanarak sahneleri ekleyin.

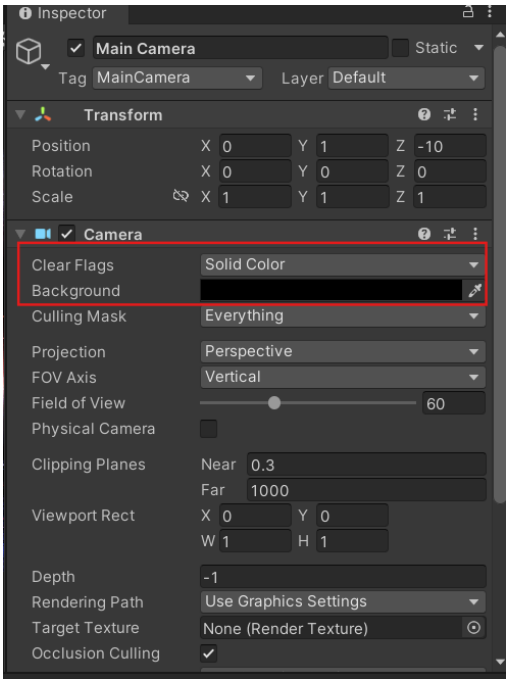


5. Ana Menü

- Yeni bir sahne oluşturun.
  - File -> New Scene seçeneğini kullanın.
  - Sahneyi Assets/Scenes altında "Main_Menu" olarak kaydedin.
- Arka planı ayarlayın.
  - UI -> Image oluşturun ve adına "Title_Screen_img" deyin.
  - Kaynak görüntüsünü Assets/UI altındaki "MainMenu" sprite olarak ayarlayın.
  - x ve y pozisyonlarını 0 olarak ayarlayın.
  - Aspect Ratio'yu koruyun.
  - Yükseklik ve genişlik değerlerini 512 olarak değiştirin.

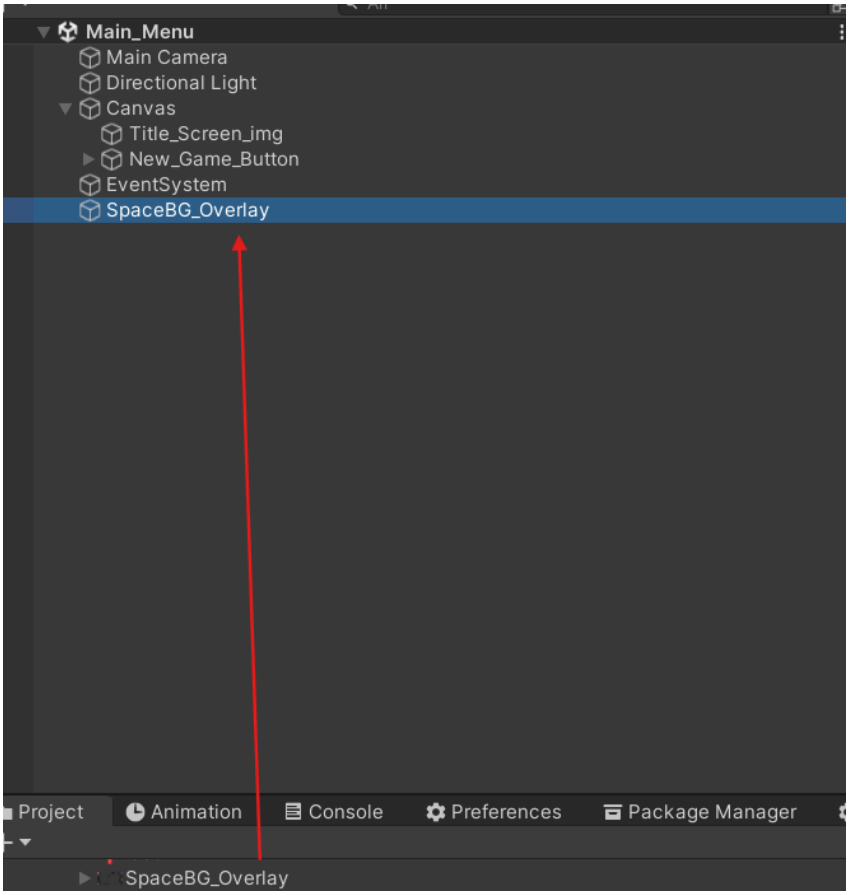


- Arka planı karartın.
  - Ana Kamerayı seçin.
  - Arka plan rengini Siyah olarak ayarlayın.
  - Clear Flags özelliğini Solid Color olarak değiştirin.



Nebula arka planını ekleyin.

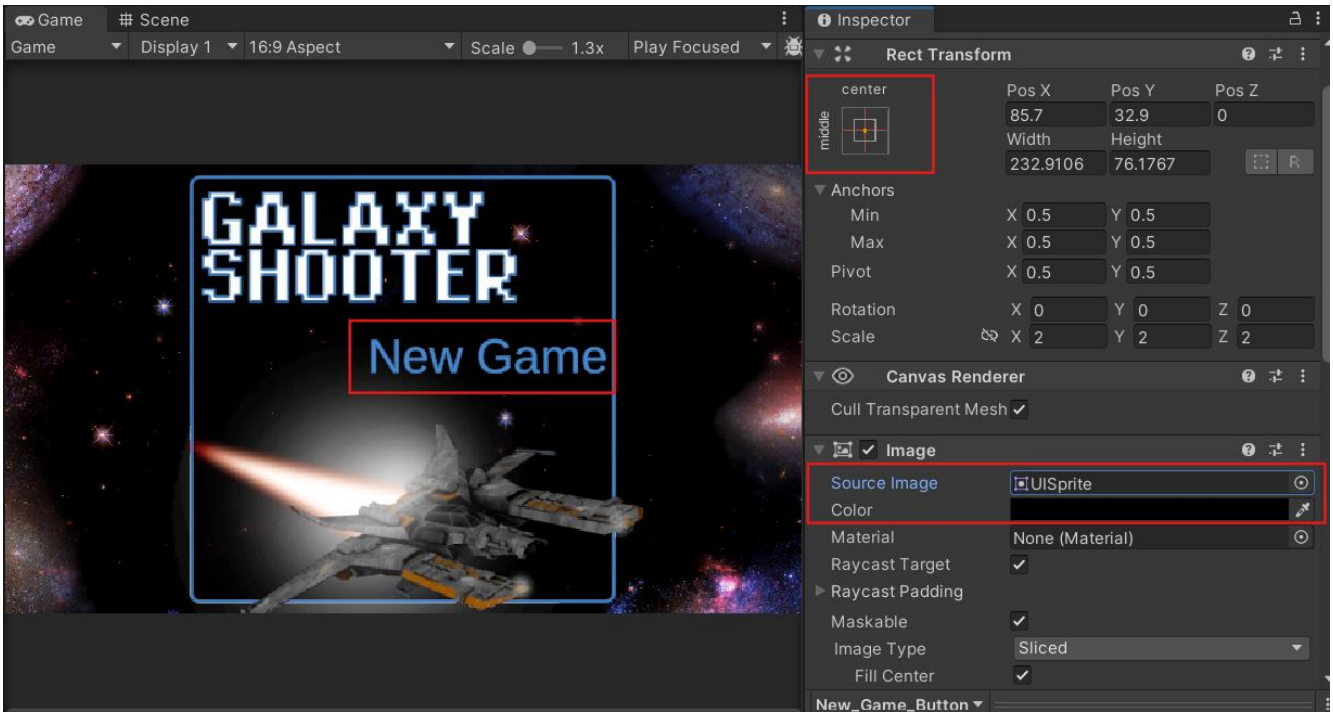
- **"SpaceBG_Overlay" öğesini seçin ve sahneye sürükleyin.**



- **Nesneye odaklanın ve kenarlarına kadar genişletin, böylece ekrana sığar.**

- Yeni bir buton oluşturun.

- Canvas'a sağ tıklayın -> UI -> Button seçeneğini seçin.
- Butona "New_Game_button" adını verin.
- Ekranın ortasına taşıyın.
- Yazıyı "New Game" olarak değiştirin.
- Vurgulanan rengi açık mavi yapın.
- Arka plan rengini siyah yapın.



- Canvas'a yeni bir script ekleyin.

➢ Scripts altında "Main_Menu" adında bir klasör oluşturun.
➢ "MainMenu_sc" adında yeni bir C# scripti oluşturun ve bu scripti Canvas'a ekleyin.

- Canvas scaler ayarını değiştirin.

➢ Canvas'ı seçin ve Canvas Scaler'ı Scale with Screen Size olarak değiştirin.

➢ Butonu ekranın ortaya sabitleyin.

Build Settings'e gidin ve Game sahnesini silin.

Add Open Scenes seçeneğiyle Ana Menü sahnesini ekleyin.

Daha sonra Game sahnesini yapı ayarlarına sürükleyin.

MainMenu scriptinde SceneManagement kütüphanesini dahil edin ve Game sahnesini yüklemek için bir genel fonksiyon tanımlayın;

```
using UnityEngine.SceneManagement;

public void LoadGame()
  {
     SceneManager.LoadScene(1);
  }
```

LoadGame fonksiyonunu MainMenu scriptinde çağırmamız gerekiyor.

New Game butonunu seçin ve “On Click” kısmında + butonuna tıklayın.

Canvas'ı tıklama olayına sürükleyin ve MainMenu_sc -> LoadGame fonksiyonunu seçin.