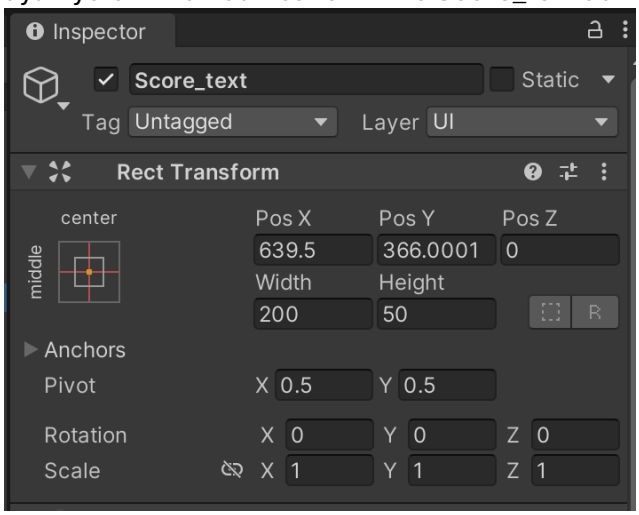
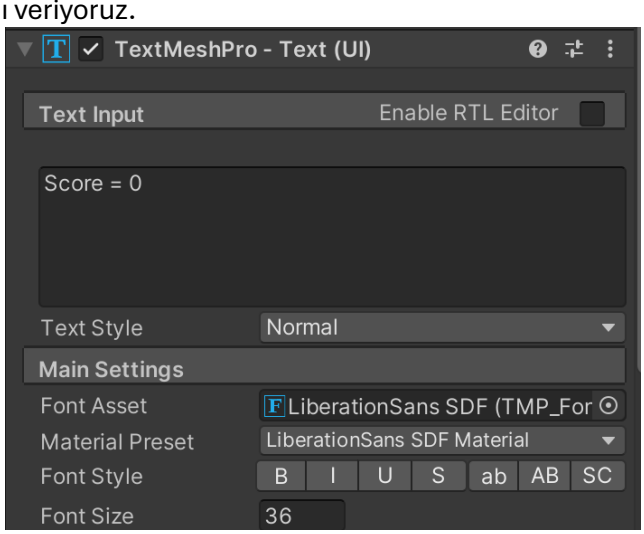
Oyun Programlama Ödevi – Hafta 8

Ad: Eren
Soyad: Köse
Numara: 22360859075
GitHub Kod Linki: https://github.com/erennkose/btu-oyun-programlama/tree/main/Hafta8

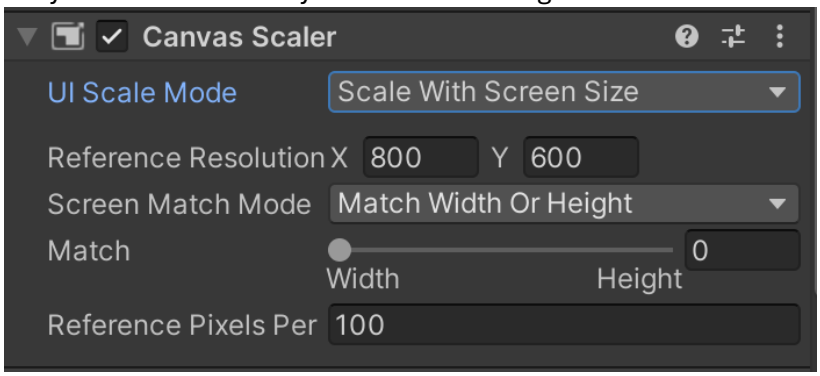
- ## Kullanıcı Arayüzü Geliştirme
  - ### Düşmanı öldürdükçe skorumuzu arttırma:

Öncelikle skorumuzu ekranda göstermek için **Hierarchyde** sağ tık atıp UI seçeneği altındaki **TextMeshPro** seçeneğini seçiyoruz. Bu sayede Canvasımız da otomatik olarak eklenecek. Aynı zamanda EventSystem de gelecek. Canvası görmek için **Scene** kısmında **Zoom Out** yapıyoruz ve Textimizi burada görebiliyoruz. Ekran boyutu değiştikçe metnin kaymaması için **Inspector** kısmındaki **Rect Transform** bölümünün sol üstündeki kısımda bulunan kare şekline tıklıyoruz ve sağ üstü seçiyoruz. Font büyüklüğünü ayarlayıp, metnin ne yazmasını istediğimizi yazıyoruz. X ve Y konumlarını Canvasın içerisinde kalacak şekilde ayarlıyoruz. Width'i ve Height'ı da ekrandan taşmayacak şekilde ayarlıyoruz. Ardından bu textimize **Score_text** adını veriyoruz.





Canvasa tıklayıp Inspector kısmından UI Scale Mode'umuzu **Scale With Screen Size** seçeneğine tıklıyoruz. Bu metnin boyutunu ekranımıza göre düzenler.



Gerekli ayarlamaları yaptıktan sonra Scripts klasörüne sağ tık atıp **Create > C# Script** seçeneğini seçiyoruz. Adını **UIManager_sc** yapıyoruz. Bu scripti sürükleyip **Canvas** nesnesine bırakıyoruz. Ardından scriptimize giriyoruz. Burada önce bazı değişkenler tanımlayacağız. Bunun için oyunumuzdaki Canvasın altında bulunan **Score_text** nesnemizi göstermemiz gerek. Kodumuzda **TextMeshProUGUI** türünde bir **scoreTMP** adında değişken oluşturuyoruz. Buna kodda erişebilmek için **Score_text** nesnemizi sürükleyip **Canvas** nesnemizin sahip olduğu **script componentinin** altındaki **Score TMP** seçeneğine sürükleyip bırakıyoruz. **Start** fonksiyonun altında **scoreTMP.text = "Score :" + 0;** diyerek metnimizi kod üzerinden başlangıç için hazırlayabiliyoruz..

```
[SerializeField]
2 başvuru
TextMeshProUGUI scoreTMP;
```

Öncelikle **Player_sc** içerisinde farklı scriptlerde ihtiyacımız olan fonksiyonlara erişmek için **Start** içerisinde bazı değişken tanımlamaları yapıyoruz.

```
void Start()
{
    spawnManager_Sc = GameObject.Find("Spawn_Manager").GetComponent<SpawnManager_sc>();
    uiManager_sc = GameObject.Find("Canvas").GetComponent<UIManager_sc>();
    if(spawnManager_Sc == null){
        Debug.Log("Spawn_Manager oyun nesnesi bulunamadı.");
    }
    if(uiManager_sc == null){
        Debug.Log("UIManager nesnesi bulunamadı.");
    }
}
```

Düşman gemilerini yok ettikçe ekranda görünecek skorun artmasını istiyoruz. Bunun için **Player_sc**'de **int score = 0;** şeklinde **score** değişkenimizi tanımlıyoruz.

```
[SerializeField]int score = 0;
```

Bu değişkeni değiştirmek için **UpdateScore** adında bir fonksiyon tanımlıyoruz. Bu fonksiyon **score'u parametre olarak aldığı int points kadar arttıracak**. Ardından ise **uiManager_sc** değişkenimiz **null değilse** uiManager_sc değişkeninin tuttuğu UIManager_sc scriptinin altındaki **UpdateScoreTMP** fonksiyonuna bu yeni **score** değerimizi verip sahnedeki metinde güncellenmesini sağlayacağız.

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

Enemy_sc scriptine gidiyoruz ve çarpışmanın gerçekleştiği **OnTriggerEnter2D** fonksiyonunu buluyoruz. **Laser** ile çarpışma durumunda oyuncunun skorunu arttırmamız gerekecek. **Destroy** ile **laseri** yok ettikten sonra Player scriptinde bulunan **UpdateScore** fonksiyonunu burada parametreye kaç puan kazandırmak istediğimizi girerek çağırıyoruz. Bu sayede skorumuzu arttırmış oluyoruz.

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

Score değişkenimizi 10 arttırma işini yaptıktan sonra bunu ekranımızda da göstermek için **UIManager_sc** içerisinde **public** bir **UpdateScoreTMP(int score)** fonksiyonu oluşturuyoruz. Buradaki Start fonksiyonunda bulunan metin gibi **scoreTMP.text = "Score:" + score;** diyoruz. Bu sayede ekranda skor güncellenmesini sağlayan fonksiyonu oluşturuyoruz.

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

Şimdi Player_sc scriptine geri dönüyoruz ve **Start** içinde bir **UIManager_sc** nesnesi oluşturuyoruz. Burada **Canvası** bulduruyoruz.

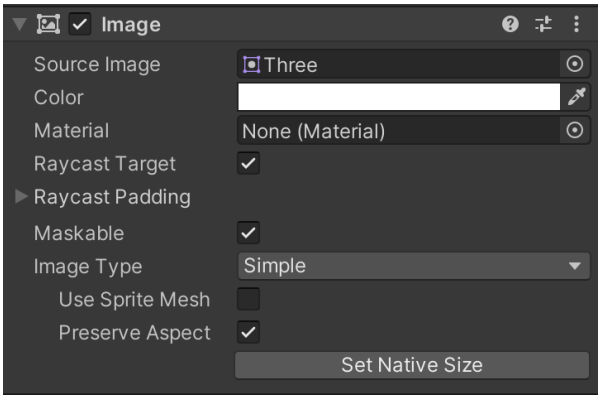
```
uiManager_sc = GameObject.Find("Canvas").GetComponent<UIManager_sc>();
```

UpdateScore içerisinde uiManager_sc değişkeninin tuttuğu UIManager_sc scriptinin altındaki **UpdateScoreTMP** fonksiyonuna bu yeni **score** değerimizi verip sahnedeki metinde güncellenmesini sağlıyoruz.

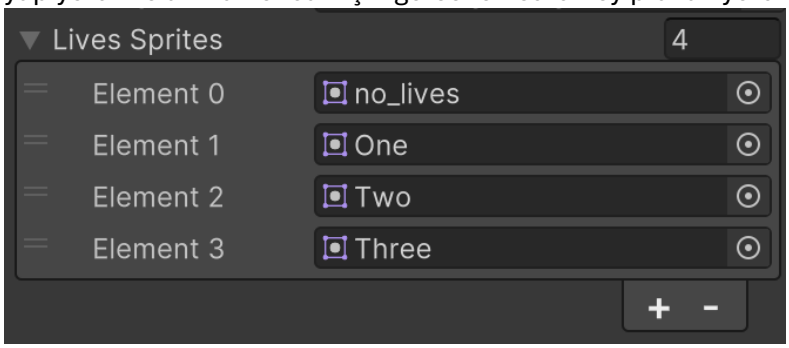
```
public void UpdateScore(int points){
    score += points;
    if(uiManager_sc != null){
        uiManager_sc.UpdateScoreTMP(score);
    }
}
```

- **Kalan canı düzenleme ve ekranda gösterme:**

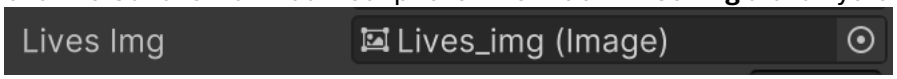
Şimdi ise ekranda ne kadar can kaldığını göstereceğiz. **Canvasa** sağ tıklayıp **UI** seçeneğinin altından **Image** seçeneğini seçiyoruz. Adını **Lives_img** yapıyoruz. Bunu da scoreu sağ üstte gösterdiğimiz şekilde fakat bu kez sol üstte gösteriyoruz. Componentlerden Source Image seçeneğine **UI** klasörünün altındaki **Lives** klasöründe bulunan **Three** olan görseli sürükleyip bırakıyoruz. X ve Y pozisyonlarını ekranda görünecek şekilde ayarlıyoruz.



Şu anda sadece canımızın tam dolu olduğu durumu gösteriyor. Dinamik olarak can görselini de güncellemek için **UIManager_sc** scriptine gidiyoruz. 4 sprite saklayacak bir **array** tanımlamalıyız. **Sprite[] liveSprites;** şeklinde **[SerializedField]** olarak tanımlıyoruz ve Unity'e geri dönüyoruz. Canvas altındaki Scriptin özelliklerine gidiyoruz. Burada Live Sprites arrayini görebiliriz. Eleman sayısını **4** yapıyoruz ve altına her can için görselleri sürükleyip bırakıyoruz.



Ardından koda geri dönüyoruz ve **UIManager_sc** içerisine Image **livesImg** adında **[SerializeField]** olarak bir değişken oluşturuyoruz. Unity ekranına dönüyoruz ve değer atamasını **Lives_img**'ı sürükle bırak ile Canavsın altındaki script özelliklerinden **Lives Img**'a bırakıyoruz.



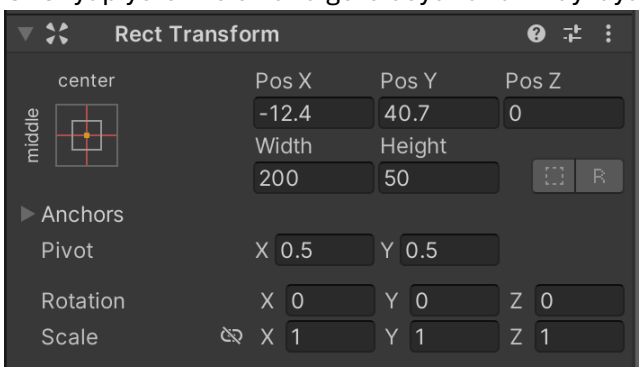
Koda geri dönüp **Start** içerisnde **livesImg.sprite = liveSprites[3]** diyoruz. Bu sayede oyun başladığında ekranda 3 canımız olduğunu göstermiş oluyoruz. Can miktarı azaldıkça livesImg.sprite'ın güncellenmesi gerekiyor. Bunun için **UpdateLivesImg** adında **public** bir fonksiyon oluşturuyoruz. **Parametre** olarak **int currentLives** alıyor. İçinde ise cana göre **livesImg.sprite** güncelleniyor. Bu fonksiyonu ise **Player_sc** içerisinde **Damage** fonksiyonunda bu fonksiyonu çağırmamız gerekiyor (Canımızı Damage fonksiyonunda değiştirdiğimizden dolayı). Bu sayede canın güncellenmesini de sağlamış oluyoruz.

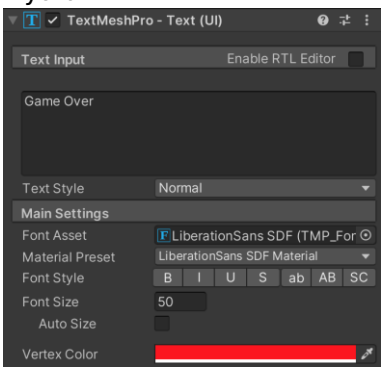
```
public void Damage(){
    if (isShieldBonusActive){
        isShieldBonusActive = false;
        shieldVisualizer.SetActive(false);
        return;
    }
    else {
        lives--;
        if(uiManager_sc != null){
            uiManager_sc.UpdateLivesImg(lives);
        }
    }

    if(lives < 1){
        if(spawnManager_Sc != null){
            spawnManager_Sc.OnPlayerDeath();
        }
        Destroy(this.gameObject);
    }
}
```

- **<u>Canımız bitince Game Over yazısını yazdırma:</u>**

Canımız bittiği zaman oyunun bittiğini belirtmemiz gerek. Canvasa sağ tıklayıp **UI** altındaki seçeneklerden TextMeshPro'yu seçiyoruz. **GameOver_text** adında isimlendiriyoruz. **Rect Transform** bölümünün sol üstündeki kısımda bulunan kare şekline tıklıyoruz ve **merkezi** seçiyoruz. Metni **Game Over** yapıyoruz ve ekrana göre boyutlandırmayı ayarlıyoruz.





Oyun başladığında bu yazının gözükmemesini istiyoruz. Bunun için **UIManager_sc** scriptine gidiyoruz. Burada **TextMeshProUGUI gameOverTMP** adında **[SerializeField]** bir değişken oluşturup Unity ekranımızda Canvasın scriptinin özellikleribdeb **Game Over TMP** seçeneğine sürükleyip **GameOver_text** nesnemizi bırakıyoruz.

Arından **UIManager_sc** scriptindeki **Start** içerisinde **gameOverTmp.gameObject.SetActive(false)** yapıyoruz. Bu sayede oyun başlayınca bu texti göremeyeceğiz.

```
gameOverTMP.gameObject.SetActive(false);
```

Canımız bitince bu texti göstermek için ise **UpdateLivesImg** altında canımızın 0'a eşit olup olmadığını kontrol edip 0 olduysa **gameOverTmp.gameObject.SetActive(true)** yaparak bu texti görünür hale getiriyoruz. Bu texte flipper efekti eklemek için ise UIManager_sc scriptinde IEnumerator tipinde bir coroutine fonksiyonu tanımlıyoruz. Adını **GameOverFlickerRoutine** olarak tanımlıyoruz. İçerisinde **while(true)** olduğu sürece önce **SetActive** fonksiyonuna **true** verip textimizin görünür hale olmasını sağlıyoruz, ardından **yield return new WaitForSeconds(0.5f)** yazıp sonrasında tekrardan **SetActive** fonksiyonuna **false** verip görünürlüğünü tekrar kapatıp ardından bir kez daha **yield return new WaitForSeconds(0.5f)** yazarak fonksiyonun tekrar çalışmasını sağlıyoruz. Bu fonksiyonu da GameOverSequence adında bir fonksiyonda **coroutine olarak** çağıracağız. Bu sayede ekrandaki **Game Over** metninde **flipper** efekt yapmış oluyoruz.

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

Oyun bitişinde çok fazla işlem yaptığımız için **UIManager_sc** içerisinde **GameOverSequence** adında bir fonksiyon oluşturuyoruz. İçerisine gerekli kodları alıyoruz ve canımızın 0 olması halinde bu fonksiyonu çağırıyoruz. Yukarıda oluşturduğumuz **GameOverFlickerRoutine** coroutine fonksiyonunu da burada **StartCoroutine** ile çağırıyoruz. Bu sayede oyun bittiğinde gerekli metinleri ekranda yazdırmış oluyoruz.

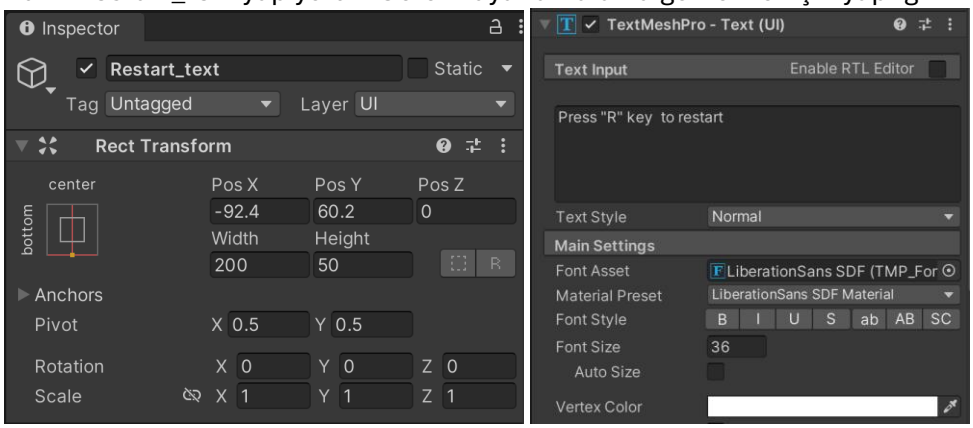
```
void GameOverSequence(){
    if(gameManager_sc != null){
        gameManager_sc.GameOver();
    }
    gameOverTMP.gameObject.SetActive(true);
    StartCoroutine(GameOverFlickerRoutine());
    restartTMP.gameObject.SetActive(true);
}
```

Yukarıda anlattığımız **GameOver** fonksiyonu ise GameManager_sc scriptinde bool isGameOver = false değişkenini true'ya çeviriyor.

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

- **<u>Restart için yönerge ve R tuşuna basınca oyunu yeniden başlatma:</u>**

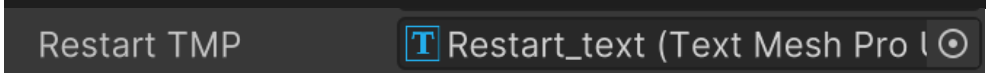
Restart tuşunu belirtmek amacıyla yeni bir **TextMeshPro** nesnesini yine **Canvas** altında oluşturuyoruz. Adını **Restart_text** yapıyoruz. Gerekli ayarlamaları diğer textler için yaptığımız gibi yine ayarlıyoruz.



Bunun da oyunun başında görünmesini istemediğimiz için Game Over yazısına yaptığımız gibi aynı şekilde görünürlüğünü **Start** içerisinde kapatıyoruz. Bunun için önce **restartTMP** adında diğer TMP değişkenlerimiz gibi bir değişken oluşturup Unity ekranından nesne atamasını yapıyoruz.

```
[SerializeField]
2 başvuru
TextMeshProUGUI restartTMP;
```

```
restartTMP.gameObject.SetActive(false);
```



R tuşuna basınca oyunu yeniden başlatmak için önce bir oyun nesnesi oluşturacağız. Hierarchyde sağ tıklayıp **Game_Manager** adında bir boş oyun nesnesi oluşturuyoruz. **Scripts** klasöründe **GameManager_sc** adında bir script oluşturup sürükle bırak yöntemiyle **Game_Manager** nesnemize bırakıyoruz. Ardından koda dönüyoruz. **UIManager_sc** içerisinde **GameManager_sc** tipinde bir **gameManager_sc** değişeni tanımlıyoruz. Start içerisinde GameObject.Find ile Game_Manager nesnesini bulup bu değişkene atıyoruz. Ardından null kontrolü yapıyoruz ve null ise hata mesajı yazdırıyoruz.

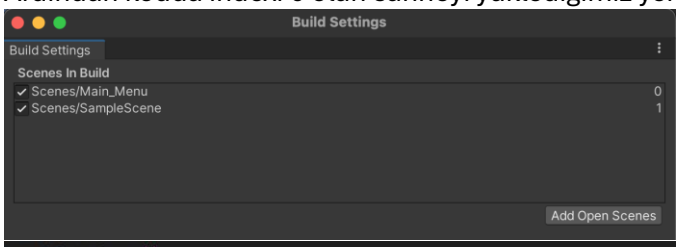
```
gameManager_sc = GameObject.Find("Game_Manager").GetComponent<GameManager_sc>();
if(gameManager_sc == null){
    Debug.LogError("UIManager_sc::Start gameManager_sc is NULL");
}
```

Ardından GameOverSequence içerisinde gameManager_sc.GameOver yazarak **GameOver** fonksiyonunu çağırıyoruz. **GameManager_sc** scriptine dönüyoruz ve **Update** içerisinde **Input.GetKeyDown(KeyCode.R)** ile R tuşuna basılma durumunu kontrol ettiriyoruz. Eğer R tuşuna basıldıysa indexi 0 olan sahnemizin tekrar yüklenmesini sağlayacağız. **SceneManager.LoadScene(0)** diyerek bu işlemi de tamamlıyoruz.

```
void Update()
{
    if(Input.GetKeyDown(KeyCode.R) && isGameOver == true){
        SceneManager.LoadScene(0);
    }
}
```

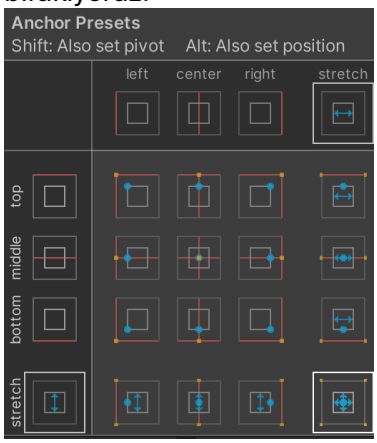
- **<u>Ana Menü için yeni sahne ve düzenlemeler:</u>**

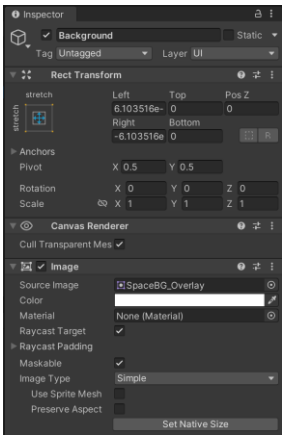
Scenes klasörü altında Create >Scene seçeneklerini seçerek Main_Menu adında bir sahne ekliyoruz. Main_Menu sahnesini ekledikten sonra indexini 0 yapıyoruz ve oyunumuzun indexini ise 1 yapıyoruz. Ardından kodda indexi 0 olan sahneyi yüklediğimiz yerdeki 0 olan indexleri 1 yapıyoruz.



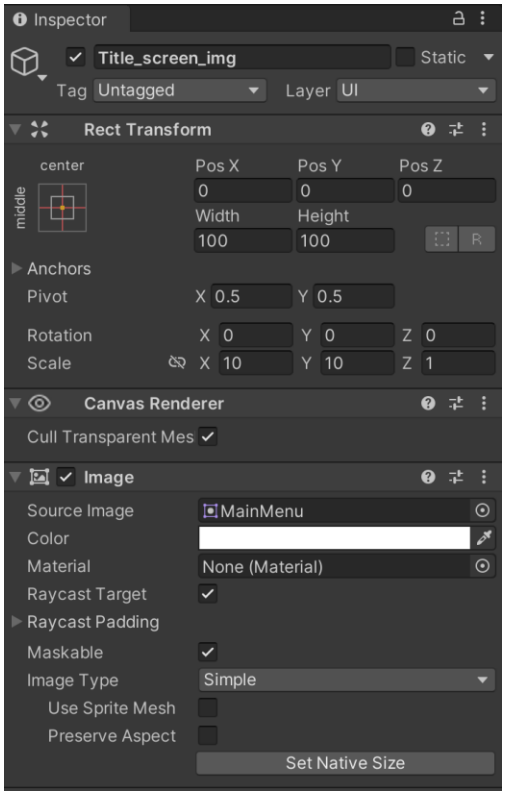

```
void Update()
{
    if(Input.GetKeyDown(KeyCode.R) && isGameOver == true){
        SceneManager.LoadScene(1);
    }
}
```

Ardından bu sahneyi Unity'de açıyoruz. Hierarchy kısmına sağ tıklayıp bir Image nesnesi oluşturuyoruz. Oluşturduğumuzda Canvas ve EventSystem de o nesneyle birlikte geliyor. Image nesnesinin adını Background yapıyoruz. Background nesnemize tıklıyoruz ve Rect Transform kısmının sol üstündeki konumlandırma kısmına tıklayıp Alt+Shift (Option+Shift) basılı tutarak sağ alttaki seçeneği seçiyoruz. Ardından Source Image kısmına Sprites klasörü altındaki Space BG_Overlay görselini sürükleyip bırakıyoruz.





Ardından Canvas'ın altında bir **Image** nesnesi daha oluşturup **Title_screen_img** adını veriyoruz. Burada da Inspector kısmına gelip **Rect Transform** kısmının sol üstündeki konumlandırma kısmına tıklayıp merkeze ayarlıyoruz. Bundan sonra ise Source Image kısmına **Sprites > UI** klasörü altındaki Main Menu görselini sürükle-bırak yapıp bırakıyoruz. Gerekli boyut ayarlamalarını da **Rect Transform** kısmından yapıyoruz.



Şimdi bir **MainMenu_sc** scripti oluşturuyoruz. Burada butona tıklanması halinde oyunun başlamasını istediğimiz için oyun sahnesini yükletecek bir kod yazıyoruz.

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

Ardından ise Canvas altında **Create > UI > Button – TextMeshPro** seçeneğini seçerek bir buton oluşturuyoruz. Bu butonu da merkeze sabitliyoruz. Ardından **Button** componentine geliyoruz ve **OnClick ()** kısmıda **+** butonuna basıyoruz. **None** yazan yere **MainMenu_sc** scriptimizi sürükle-bırak yapıp bırakıyoruz.