Ad: Yusuf

Soyad: Güney

Numara: 22360859041

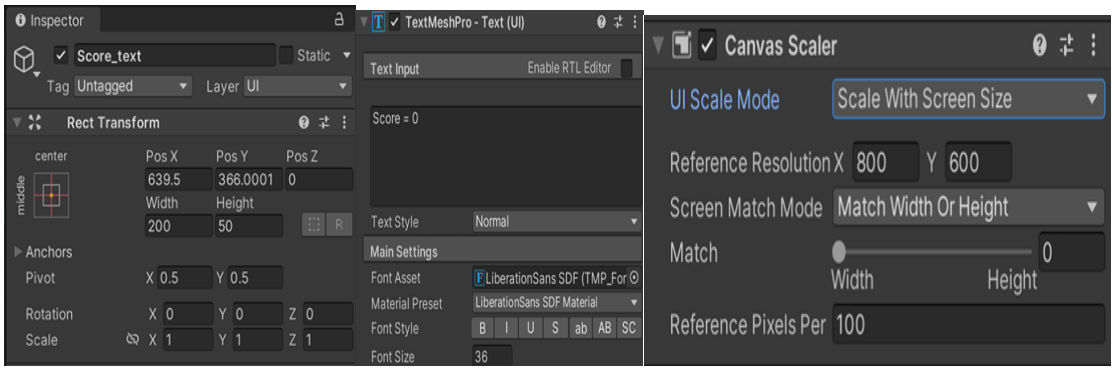
Ders: Oyun Programlama Hafta 8 Rapor

Kullanıcı ara yüzü geliştirme

1-Düşman gemileri yok ettiğinizde puan kazandıracak sistem geliştirin ve anlık puan durumunu ekranda gösterin

Skoru ekranda göstermek için öncelikle Hierarchy'den sağ tıklayarak UI altınkadi TextMeshPro seçeneğini seçiyoruz. Bu işlem, otomatik olarak Canvas'ın eklenmesini sağlıyor. Ayrıca EventSystem de otomatik olarak oluşturuluyor. Canvas'ı görüntülemek için Scene kısmında uzaklaştırma (Zoom Out) yapıyoruz ve metni burada görebiliyoruz. Metnin ekran boyutu değiştiğinde kaymasını önlemek için, Inspector'da Rect Transform bölümünün sol üst köşesindeki kare simgesine tıklayıp sağ üst köşeyi seçiyoruz. Yazı boyutunu ayarlıyor, metin içeriğini belirliyor ve X ile Y pozisyonlarını Canvas sınırları içinde kalacak şekilde düzenliyoruz. Genişlik (Width) ve yükseklik (Height) değerlerini ise ekran dışına taşmayacak şekilde ayarlıyoruz. Son olarak, bu metne Score_text adını veriyoruz.

Canvas'a tıkladıktan sonra Inspector panelinde **UI Scale Mode** ayarını **Scale With Screen Size** olarak seçiyoruz.



Script dosyası içerisinde UIManager_sc adında yeni bir script dosyası oluşturuyoruz Ardından bu scripti sürükleyerek Canvas nesnesine ekliyoruz.

Scripti açtığımızda, öncelikle bazı değişkenler tanımlıyoruz. Bunun için, oyunumuzda Canvas altında yer alan Score_text nesnesine erişmemiz gerekiyor. Kodda, TextMeshProUGUI türünde bir scoreTMP adlı değişken tanımlıyoruz. Bu değişkene erişim sağlamak için, Score_text nesnesini sürükleyerek Canvas nesnesindeki script bileşeninde görünen Score TMP alanına bırakıyoruz.

Daha sonra, Start fonksiyonunun altında şu kodu yazıyoruz:

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

Bu sayede, metni kod üzerinden başlangıçta "Score : 0" şeklinde hazırlamış oluyoruz.

```
[SerializeField]
0 references
TextMeshProUGUI scoreTMP;
```

Player_sc içerisinde, farklı scriptlerdeki fonksiyonlara erişim sağlamak için öncelikle **Start** fonksiyonunda gerekli değişkenleri tanımlıyoruz.

```
0 references
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.");
    }
}
```

Skor değişkenimizi
int Score = 0;

Olacak şekilde tanımlıyoruz.

Skor güncellemesi yapabilmek için **UpdateScore** adında bir fonksiyon tanımlıyoruz. Bu fonksiyon, bir parametre olarak aldığı **int points** değeri kadar skoru artıracak. Daha sonra, **uiManager_sc** değişkenimiz boş (null) değilse, bu değişkenin işaret ettiği **UIManager_sc** scriptindeki **UpdateScoreTMP** fonksiyonuna yeni skor değerimizi ileteceğiz. Böylece sahnedeki metin de güncellenmiş olacak.

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

Enemy_sc scriptine giderek, çarpışma olaylarının işlendiği **OnTriggerEnter2D** fonksiyonunu buluyoruz. Burada, düşmanın **Laser** ile çarpıştığı durumu kontrol ederek oyuncunun skorunu artırmamız gerekiyor. **Laser** yok edildikten sonra, **Player** scriptindeki **UpdateScore** fonksiyonunu çağırıyoruz ve parametre olarak kazandırmak istediğimiz puan miktarını iletiyoruz. Bu işlem, oyuncunun skorunu artırmamızı sağlıyor.

```
0 references
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);
    }
}
```

Skor güncellemesini ekranda göstermek için **UIManager_sc** içerisinde **public UpdateScoreTMP(int score)** adında bir fonksiyon tanımlıyoruz. Bu fonksiyon, parametre olarak aldığı **score** değerini kullanarak metni güncelliyor. İçeride, **Start** fonksiyonundaki metni ayarladığımız gibi, **scoreTMP.text** değerini değiştiriyoruz.

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

Player_sc scriptindeki Start içinde bir UIManager_sc nesnesi oluşturuyoruz. Burada Canvası bulduruyoruz.

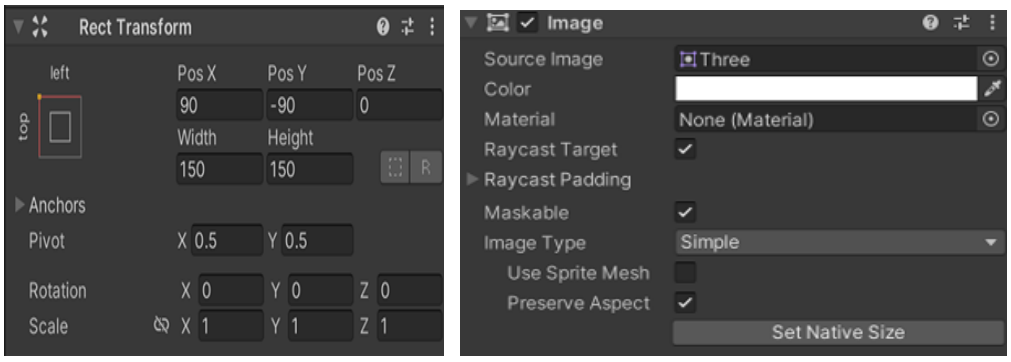
```
uiManager_sc = GameObject.Find("Canvas").GetComponent<UIManager_sc>();
```

UpdateScore fonksiyonunda, güncellenen skoru sahnedeki metinde göstermek için, **uiManager_sc** değişkeni üzerinden **UIManager_sc** scriptindeki **UpdateScoreTMP** fonksiyonunu çağırıyoruz ve yeni skor değerimizi veriyoruz.

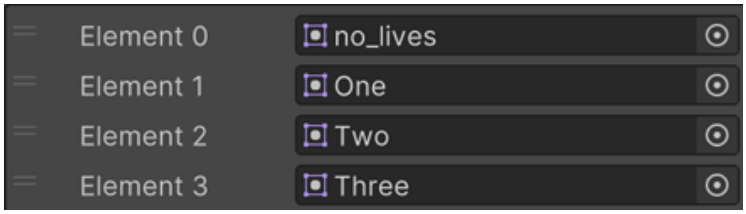
```
0 references
public void UpdateScore(int points){
    score += points;
    if(uiManager_sc != null){
        uiManager_sc.UpdateScoreTMP(score);
    }
}
```

2- Oyuna üç canla başlayın ve mevcut canı göstermek için lives sprite'ı kullanın

Ekranda kalan can miktarını göstermek için ilk adım olarak canvas üzerinde sağ tıklayıp **UI** seçeneği altından **Image** öğesini seçiyoruz. Bu yeni öğeye **Lives_img** adını veriyoruz. Score'u sağ üst köşede gösterdiğimiz yöntemi takip ederek, bu kez sol üst köşede görünmesini sağlıyoruz. **Source Image** bileşenine, **UI** klasörünün içindeki **Lives** klasöründe yer alan **Three** görselini sürükleyip bırakıyoruz. Son olarak, X ve Y pozisyonlarını ayarlıyoruz.



Şu anda yalnızca canın tamamen dolu olduğu durumu gösteriyoruz. Dinamik olarak can görselini güncelleyebilmek için **UIManager_sc** scriptine gidiyoruz. Bu script içerisinde, 4 sprite saklayacak bir array tanımlamamız gerekiyor. **Sprite[] liveSprites;** şeklinde **[SerializedField]** olarak tanımlıyoruz ve ardından Unity'e geri dönüyoruz. Canvas altındaki scriptin özelliklerine gidiyoruz ve burada **Live Sprites** arrayini görebiliyoruz. Eleman sayısını 4 olarak ayarlıyoruz ve her can için uygun görselleri sürükleyip array'e ekliyoruz.



UIManager_sc scriptine **Image livesImg** adında bir değişken ekliyoruz. Ardından Unity ekranına dönüp, **Lives_img** görselini **Canvas** altındaki script özelliklerinden **Lives Img** değişkenine sürükleyip bırakıyoruz. Koda geri dönüp, **Start** fonksiyonu içinde **livesImg.sprite = liveSprites[3];** şeklinde bir satır ekliyoruz. Bu sayede oyun başladığında ekranda 3 canımız olduğunu gösterebiliyoruz. Can miktarı azaldıkça **livesImg.sprite**'ının güncellenmesi gerekiyor. Bunun için **UpdateLivesImg** adında public bir fonksiyon oluşturuyoruz. Bu fonksiyon, **int currentLives** parametresi alacak ve içinde cana göre **livesImg.sprite**'ı güncelleyecek. Son olarak, bu fonksiyonu **Player_sc** scriptinin **Damage** fonksiyonu içinde çağırmamız gerekiyor, çünkü can miktarını **Damage** fonksiyonu içinde değiştiriyoruz.

```
0 references
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);
    }
}
```

3-Game Over yazısı oluşturun ve oyun bittiğinde ekranda gösterin

Canımız bittiği zaman oyunun bittiğini belirtmek için, **Canvas**'a sağ tıklayıp **UI** altındaki seçeneklerden **TextMeshPro**'yu seçiyoruz. Bu na da **GameOver_text** adını veriyoruz. Ardından, **Rect Transform** bölümündeki sol üst kısımdaki kare şekline tıklayarak merkezi seçiyoruz. Metin kısmını **Game Over** olarak değiştiriyoruz ve ardından ekranda uygun şekilde görünmesi için boyutlandırmayı ayarlıyoruz. Bu şekilde, oyunun bittiğini belirten bir **Game Over** mesajı eklemiş oluyoruz.

Oyun başladığında **Game Over** yazısının görünmemesini sağlamak için **UIManager_sc** scriptine gidiyoruz. Burada **TextMeshProUGUI gameOverTMP** adında bir değişken oluşturuyoruz. Ardından, Unity ekranına dönüp, **Canvas**'ın script özelliklerinden **Game Over TMP** seçeneğine **GameOver_text** nesnemizi sürükleyip bırakıyoruz.

Arından UIManager_sc scriptindeki Start içerisinde gameOverTmp.gameObject.SetActive(false) yapıyoruz. Bunu yapma sebebimiz oyun başladığında Game Over textini görmek istemediğimizdendir.

Canımız bitince **Game Over** yazısını göstermek için, **UpdateLivesImg** fonksiyonunda canımızın 0'a eşit olup olmadığını kontrol ediyoruz. Eğer canımız 0 olduysa, **gameOverTMP.gameObject.SetActive(true);** komutunu kullanarak **Game Over** metnini görünür hale getiriyoruz. Bu metne bir **flipper efekti** eklemek için, **UIManager_sc** scriptinde **IEnumerator** tipinde bir coroutine fonksiyonu tanımlıyoruz. Fonksiyonun adını **GameOverFlickerRoutine** olarak belirliyoruz. İçerisinde, **while(true)** döngüsü kullanarak aşağıdaki adımları takip ediyoruz:

1. **gameOverTMP.gameObject.SetActive(true);** Bu komut ile metni görünür hale getiriyoruz.
2. **yield return new WaitForSeconds(0.5f);** 0.5 saniye bekliyoruz.
3. **gameOverTMP.gameObject.SetActive(false);** Bu komut ile metnin görünürlüğünü kapatıyoruz.
4. **yield return new WaitForSeconds(0.5f);** Tekrar 0.5 saniye bekliyoruz.

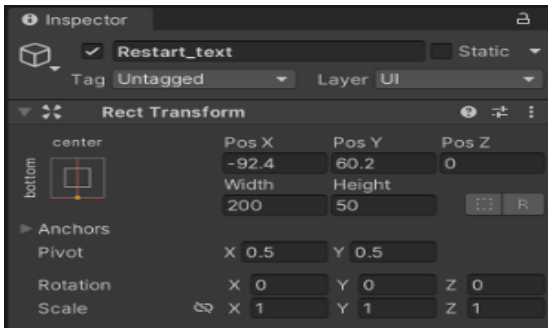
Bu döngü sürekli olarak çalışacak ve **Game Over** metni ekranda yanıp sönecektir.

Son olarak, bu coroutine fonksiyonunu **GameOverSequence** adında bir fonksiyon içerisinde çağırıyoruz. **StartCoroutine(GameOverFlickerRoutine());** komutunu kullanarak **flipper efekti**'ni başlatıyoruz. Bu şekilde, ekrandaki **Game Over** metnine **flipper efekti** eklemiş oluyoruz.

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

4-Oyunu yeniden başlatmak için yönerge oluşturun. R tuşuna basıldığında oyun yeniden başlasın

Restart tuşunu belirtmek amacıyla yeni bir **TextMeshPro** nesnesi oluşturuyoruz. Bunun için **Canvas** altında sağ tıklayıp **UI** altından **TextMeshPro**'yu seçiyoruz. Bu yeni öğeye **Restart_text** adını veriyoruz. Ardından, diğer metinlerde yaptığımız gibi gerekli ayarlamaları yapıyoruz.



Oyunun başında görünmesini istemediğimiz için, Game Over yazısında yaptığımız gibi, görünürlüğünü Start fonksiyonu içinde kapatıyoruz. Bunu gerçekleştirmek için, önce "restartTMP" adında, diğer TMP değişkenlerimize benzer bir değişken tanımlayıp, Unity editöründen bu nesneyi atıyoruz.

```
[SerializeField] TextMeshProUGUI restartTMP;
```

Hierarchy penceresinde sağ tıklayarak "Game_Manager" adında boş bir oyun nesnesi ekliyoruz. Scripts klasöründe "GameManager_sc" adında bir script oluşturup, sürükle-bırak yöntemiyle bu scripti Game_Manager nesnesine bağlıyoruz. Daha sonra koda geri dönüyoruz. UIManager_sc scripti içerisinde, GameManager_sc tipinde bir "gameManager_sc" değişkeni tanımlıyoruz. Start fonksiyonunda, GameObject.Find ile Game_Manager nesnesini bulup, bu nesneyi gameManager_sc değişkenine atıyoruz. Ardından null kontrolü yaparak, eğer nesne 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");
}
```

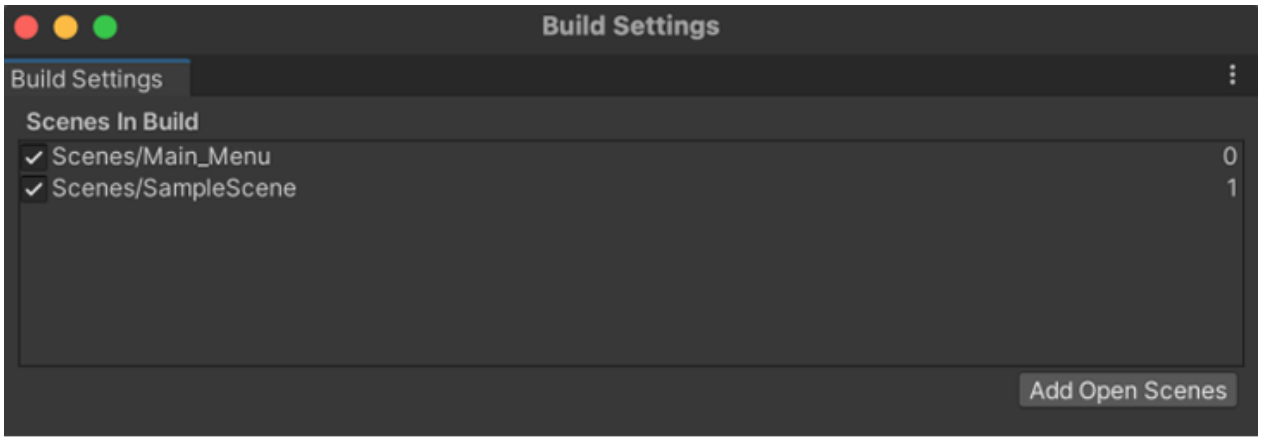
Sonrasında, GameOverSequence içerisinde gameManager_sc.GameOver yazarak GameOver fonksiyonunu çağırıyoruz. Ardından, GameManager_sc scriptine dönüyoruz ve Update fonksiyonunda Input.GetKeyDown(KeyCode.R) ile R tuşuna basılma durumunu kontrol ediyoruz. Eğer R tuşuna basıldıysa, indexi 0 olan sahnemizin tekrar yüklenmesini

sağlamak için SceneManager.LoadScene(0) kodunu ekliyoruz. Bu şekilde oyunun yeniden başlamasını sağlıyoruz.

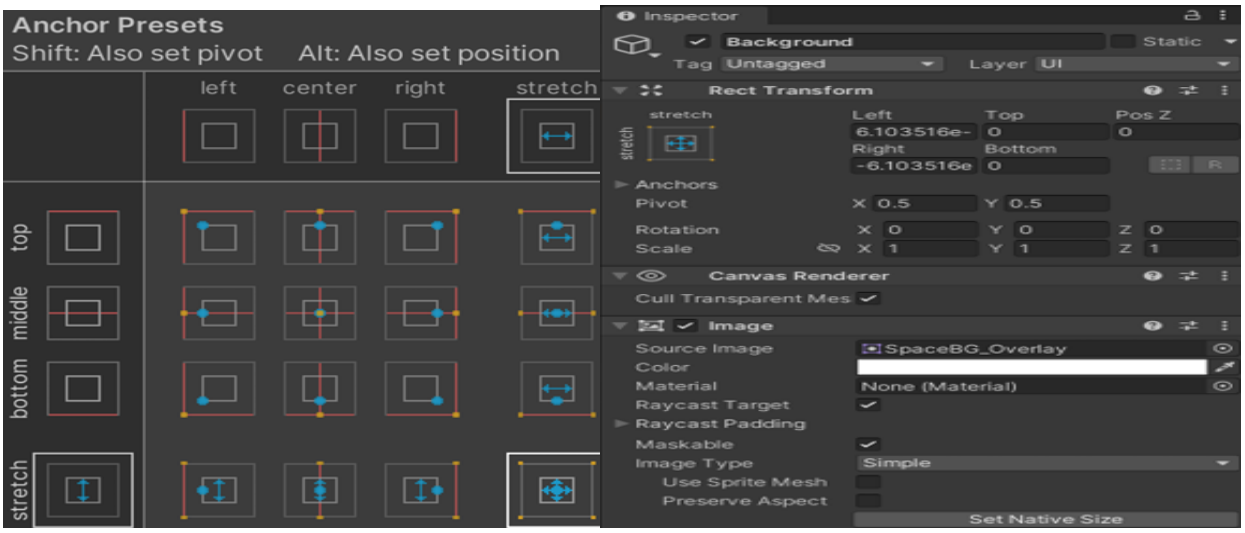
```
0 references
void Update()
{
    if(Input.GetKeyDown(KeyCode.R) && isGameOver == true){
        SceneManager.LoadScene(1);
    }
}
```

5-Ana menü için yeni bir sahne oluşturun. Arka planı ayarlayın. Yeni oyun başlatmak için buton ekleyin.

Scenes klasörü altında Create içerisinde Scene seçeneklerini seçerek "Main_Menu" adında bir sahne oluşturuyoruz. Main_Menu sahnesini ekledikten sonra, sahne indexini 0 yapıyoruz ve oyunumuzun sahnesinin indexini ise 1 olarak ayarlıyoruz. Ardından, kodda indexi 0 olan sahneyi yüklediğimiz yerlerde, bu indexleri 1 olarak değiştiriyoruz. Böylece, oyuna başladığımızda Main_Menu sahnesi ilk olarak yüklenecek, oyun sahnesine geçiş ise indexi 1 olan sahneye yapılacak.



Ardından bu sahneyi Unity'de açıyoruz. Hierarchy kısmına sağ tıklayarak bir Image nesnesi oluşturuyoruz. Oluşturduğumuzda, Canvas ve EventSystem de otomatik olarak bu nesneyle birlikte eklenir. Image nesnesinin adını "Background" olarak değiştiriyoruz. Background nesnesine tıklayarak, Rect Transform kısmının sol üstündeki konumlandırma butonuna tıklıyoruz. Alt+Shift (Option+Shift) tuşlarına basılı tutarak sağ alttaki köşe seçeneğini seçiyoruz. Sonrasında, Source Image kısmına Sprites klasörü altındaki "Space BG_Overlay" görselini sürükleyip bırakıyoruz. Bu şekilde Background görselini sahnede ayarlamış olduk.

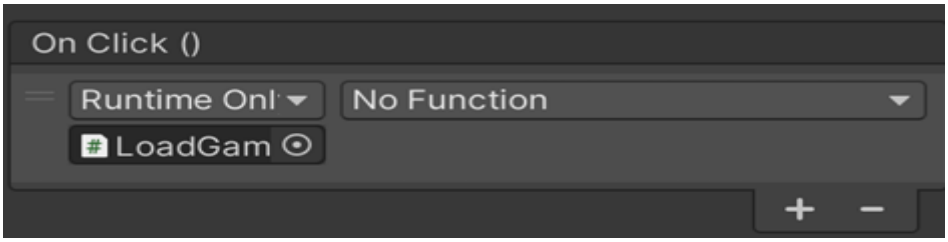


Canvas altına yeni bir Image nesnesi ekleyip, adını "Title_screen_img" olarak belirliyoruz. Ardından, Inspector penceresinde Rect Transform bölümündeki sol üst konumlandırma seçeneğine tıklayarak görseli merkeze yerleştiriyoruz. Sonrasında, Source Image kısmına gidip, Sprites > UI klasörü altındaki "Main Menu" görselini sürükleyip bırakıyoruz. Son olarak, gerekirse boyut ayarlamalarını Rect Transform bölümünden yapıyoruz.

Şimdi, bir "MainMenu_sc" scripti oluşturuyoruz. Bu script içinde, butona tıklanması durumunda oyunun başlamasını sağlamak için oyun sahnesini yükleyecek bir kod yazıyoruz

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

Son olarak, Canvas altında "Create > UI > Button – TextMeshPro" seçeneğini seçerek bir buton oluşturuyoruz. Bu butonu merkeze yerleştiriyoruz. Ardından, Button bileşenine gidip, OnClick() kısmında "+" butonuna tıklıyoruz. Burada, None yazan alana "MainMenu_sc" scriptini sürükleyip bırakıyoruz. Bu işlemle, butona tıklandığında belirttiğimiz fonksiyonun çalışmasını sağlıyoruz.



Kodların tamamına Github sayfasından ulaşabilirsiniz.

Github sayfası linki: