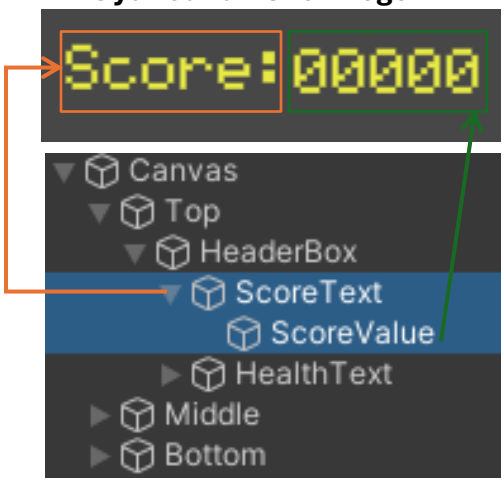
8.Hafta Ders Raporu – Batuhan Şengül

UI- Canvas

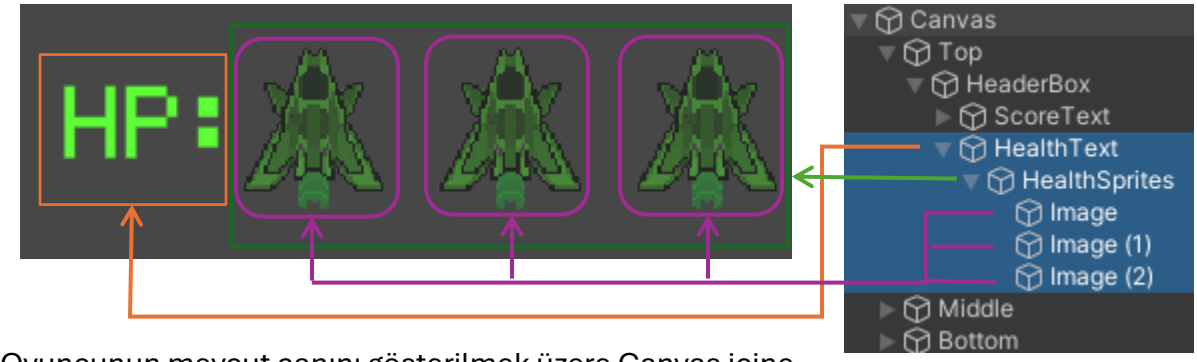
Oyun Arayüzü

- **Oyuncunun Skor Değeri**



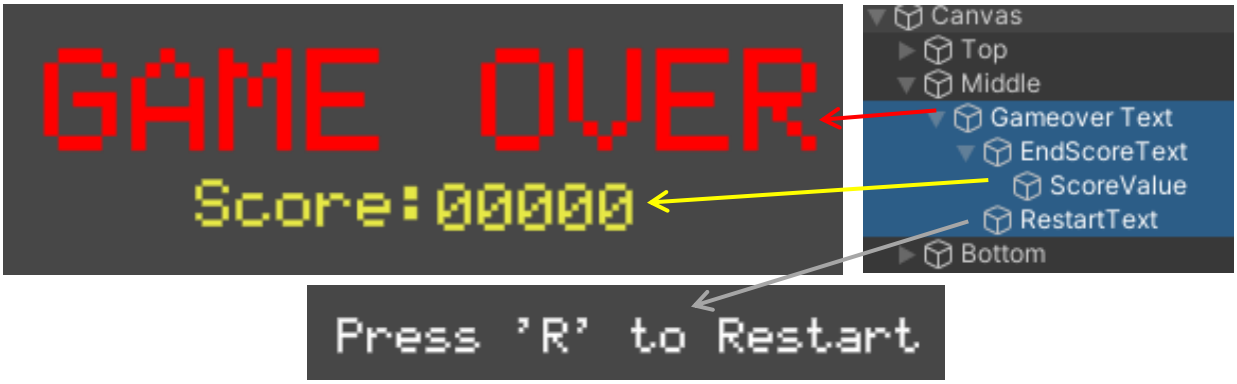
Oyuncunun mevcut skorunu göstermek için iki tane text nesnesi oluşturulup renk ve font ayarları yapılıp bu oyun nesnelerin konumları oyun ekranının orta üstünde olacak şekilde ayarlanmıştır. Düşman yok edildikçe skor değerinin güncelleyen kod mantığı yazılmıştır. (Bkz. AddScore(int amount) Fonksiyonu)

- **Oyuncunun Can Değerleri**



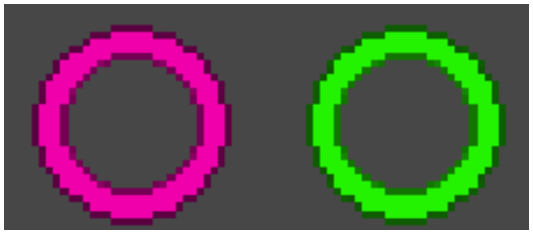
Oyuncunun mevcut canını gösterilmek üzere Canvas içine spirite atılıp bu oyun nesnelerinin konumları oyun ekranının sol üstünde olacak şekilde ayarlanmıştır. Oyun oynanırken mevcut canlar yeşil, kaybedilen canları kırmızı renkle gösterecek kod yazılmıştır. (Bkz. DamagePlayer(int healthPoint))

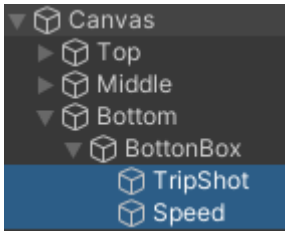
- **Oyun Sonu Yazısı**



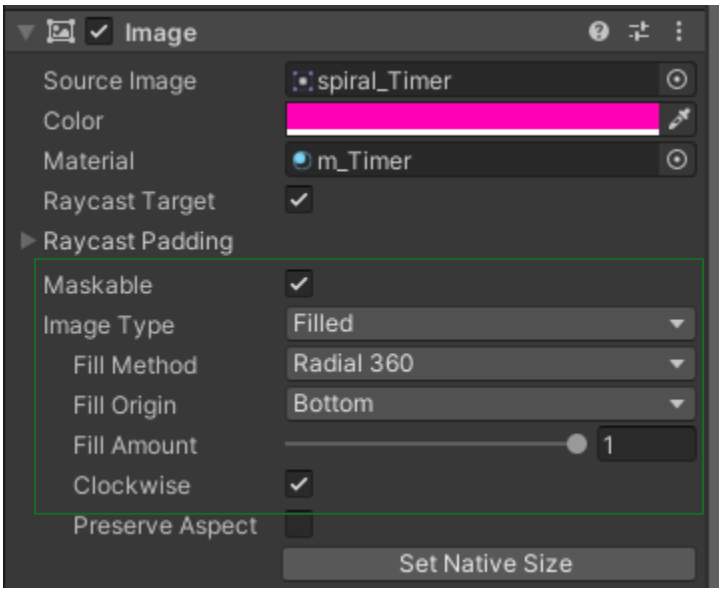
Oyuncunun mevcut canı 0 olduğu zaman gözükecek oyun nesneleri ayarlanmıştır. Bu yazı nesnelerinin yanıp sönmesini sağlayacak fonksiyon kod ile ayarlanmıştır. (Bkz. GameoverUIUpdates() Fonksiyonu)

- **Güçlendirme Sayaç Göstergeleri(Ekstra)**





Bu sayaçlar ile oyuncunun mevcut sahip olduğu güçlendirmenin bitmesine ne kadar kaldığı görselleştirilmiştir.



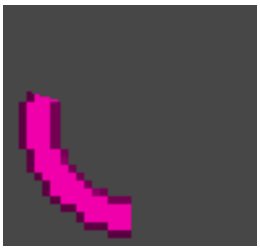
Bu resimlerin(spritelerın):

- Image Type-> Filled
- Fill Method -> Radial 360
- Fill Origin -> Bottom

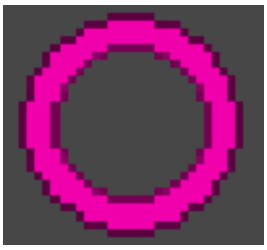
Olarak ayarlanmıştır. Kod ile fill amount 1 ile 0 arasında değiştirilerek sayaç haline getirilmiştir. (Bkz. IEnumerator PowerupUIUpdate(Image powerUpImage, float duration) Fonksiyonu)

Fill amount farklı değer örnekleri:

Fill Amount = 0.65



Fill Amount = 0.30



Fill Amount = 1.00

TextManager Scripti

Arayüz güncellemelerini genel kontrolünü sağlayacak bir script(TextManager.cs) oluşturulmuştur.

- **Değişkenler**

```
public TextMeshProUGUI scoreValueText;
public float incrementSpeed = 0.05f;

private int currentScore = 0;
private int targetScore = 0;
private Coroutine incrementCoroutine;

[Header("Health Settings")]
public GameObject healthSpritesParent;
public Color damageColor = Color.red;
private Color defaultHealthColor=Color.green;

[Header("UI Settings")]
public GameObject top;
public GameObject middle,bottom, gameoverGameObject;
public TextMeshProUGUI endScoreValueText;
public float flickerSpeed = 0.2f;

private Coroutine flickerCoroutine;
private bool gameEnded= false;
```

public TextMeshProUGUI scoreValueText: Oyuncunun skorunu gösteren bileşen.

public float incrementSpeed = 0.05f : Skor artış animasyon hızının tutan değişken.

private int currentScore = 0: Mevcut skorun değerini tutan değişken.

private int targetScore = 0: Skorun değişimle ulaşacağı değeri tutan değişken.

private Coroutine incrementCoroutine: Skor artış efektinin halledildiği coroutine.

public GameObject healthSpritesParent: Oyuncunun can spritelarının tutulduğu parent oyun nesnesi.

public Color damageColor = Color.red: Hasar alındığında oyuncunun canının güncelleneceği renk.

private Color defaultHealthColor=Color.green: Hasar alınmadan ayarlanan can rengi.

public GameObject top,middle,bottom, gameoverGameObject: Arayüz nesneleri

public TextMeshProUGUI endScoreValueText:Oyun sonu skorunu tutan değişken.

public float flickerSpeed = 0.5f: Oyun sonundaki yanıp sönme sıklığını tutan değer.

private Coroutine flickerCoroutine: Oyun sonundaki yanıp sönme efektini tutan corotine.

private bool gameEnded= false: Oyunun bitip bitmediğini tutan değişken.

- **Fonksiyonlar**

1. **Başlangıç Ayarları (Awake/Start)**

```
Unity Message | 0 references
private void Awake()
{
    if (gameoverGameObject.activeSelf)
    {
        gameoverGameObject.SetActive(false);
    }

}
Unity Message | 0 references
private void Start()
{

    scoreValueText.text = "000000";

    defaultHealthColor = healthSpritesParent.transform.GetChild(0).GetComponent<Image>().color;

}
```

1. Oyun başlangıcında GameOver ekranı saklanır.
2. Skor 000000'a güncellenir.
3. Can spritelarının renkleri alınır.

2. **Update() Fonksiyonu**

```
Unity Message | 0 references
private void Update()
{
    if(gameEnded&& Input.GetKeyDown(KeyCode.R))
    {
        SceneManager.LoadScene("MainScene");
    }
}
```

Eğer oyuncunun canı kalmayıp oyun bittiyse R tuşuna basılıp basılmadığını kontrol eder, tuşa basıldığında oyunu baştan başlatır.

3. **AddScore(int amount) Fonksiyonu**

```
public void AddScore(int amount)
{
    targetScore += amount;


    if (incrementCoroutine != null)
    {
        StopCoroutine(incrementCoroutine);
    }

    incrementCoroutine = StartCoroutine(UpdateScore());
}
```

Skoru belirlenen miktar kadar arttırılmasını sağlayan fonksiyondur.

4. **IEnumerator UpdateScore()**

```
1 reference
private IEnumerator UpdateScore()
{
    while (currentScore < targetScore)
    {
        currentScore+=5;
        scoreValueText.text = currentScore.ToString("D6");

        yield return new WaitForSeconds(incrementSpeed);
    }

    incrementCoroutine = null;
}
```

Değişen skorun arayüzde güncellemesini yapan fonksiyondur.

5. **DamagePlayer(int healthPoint)**

```
1 reference
public void DamagePlayer(int healthPoint)
{

    Transform damagedSprite = healthSpritesParent.transform.GetChild(healthPoint-1);
    Image spriteImage = damagedSprite.GetComponent<Image>();

    if (spriteImage != null)
    {

        spriteImage.color = damageColor;

    }

}
```

Oyuncu hasar aldığında arayüz güncellemelerinin yapıldığı fonksiyondur.

6. **IEnumerator PowerupUIUpdate(Image powerUpImage, float duration) Fonksiyonu**

```
4 references
public IEnumerator PowerupUIUpdate(Image powerUpImage, float duration)
{

    powerUpImage.gameObject.transform.SetSiblingIndex(1);
    powerUpImage.gameObject.SetActive(true);
    powerUpImage.fillAmount = 1;

    float elapsedTime = 0;

    while (elapsedTime < duration)
    {
        elapsedTime += Time.deltaTime;
        powerUpImage.fillAmount = 1 - (elapsedTime / duration);
        yield return null;
    }

    powerUpImage.fillAmount = 0;
    powerUpImage.gameObject.SetActive(false);
}
```

Güçlendirmelerin kalan süresinin görselleştirildiği fonksiyondur. Herhangi bir güçlendirmeyle etkileşime girildiğinde çağrılır.

7. **GameoverUIUpdates() Fonksiyonu**

```
1 reference
public void GameoverUIUpdates()
{
    top.SetActive(false);
    gameoverGameObject.SetActive(true);
    bottom.SetActive(false);
    endScoreValueText.text = currentScore.ToString("D6");
    if (flickerCoroutine != null)
    {
        StopCoroutine(flickerCoroutine);
    }
    flickerCoroutine = StartCoroutine(FlickerText());
}
```

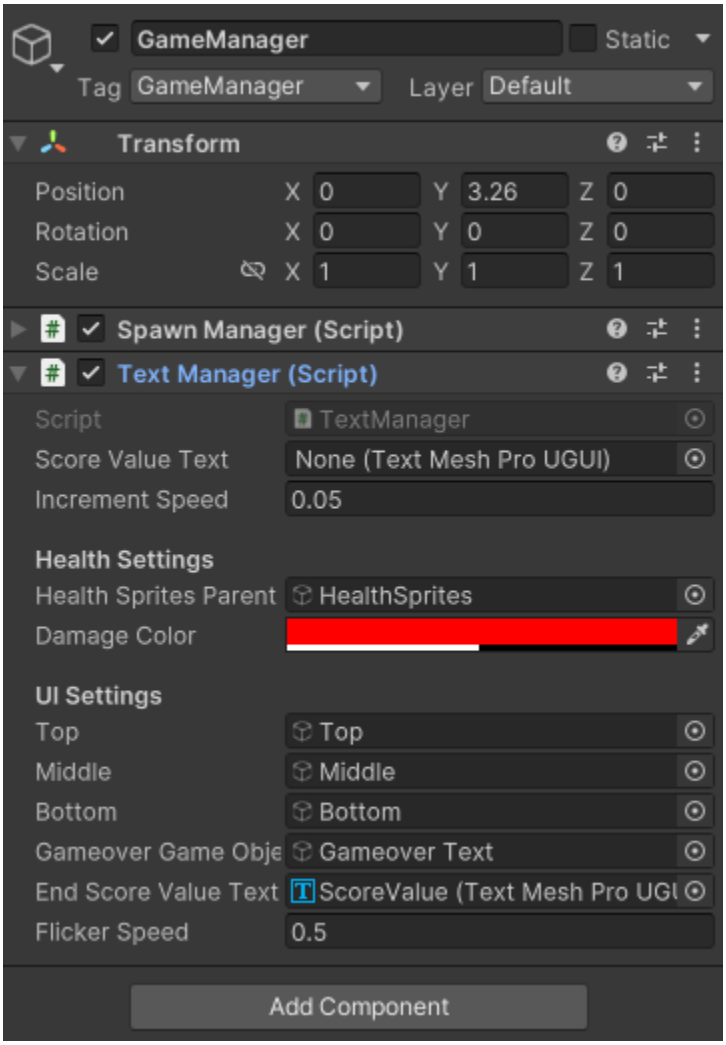
Oyun sonunda arayüz güncellemelerinin yapıldığı fonksiyondur. Oyuncunun canı 0 olduğu zaman çağırılır.

8. **IEnumerator FlickerText() Fonksiyonu**

```
1 reference
private IEnumerator FlickerText()
{
    TextMeshProUGUI[] texts = middle.GetComponentsInChildren<TextMeshProUGUI>();
    while (true)
    {
        foreach (TextMeshProUGUI text in texts)
        {
            Color color = text.color;
            color.a = color.a == 1f ? 0f : 1f;
            text.color = color;
        }
        yield return new WaitForSeconds(flickerSpeed);
    }
}
```

Oyun sanında ekranda gözüken yazıların yanıp sönmesini sağlayan corotinedir. Oyuncunun canı 0 olduğu zaman çağırılır.

Oluşturulan bu script GameManager Oyun nesnesinin içine eklenip gerekli değişkenlerin atamaları yapılıp düzenlenmiştir.



Enemy Script Güncellemeleri

```
private TextManager tm;
```

- TextManager türünden bir değişken oluşturulmuştur.

```
Unity Message | 0 references
void Start()
{
    pm = FindObjectOfType<PlayerMovement>();

    tm = GameObject.FindGameObjectWithTag("GameManager").GetComponent<TextManager>();

    xPos = Random.Range(-pm.xBorderValue, pm.xBorderValue);
    transform.position = new Vector3(xPos, 6, 0);
    speed = speed+ Random.Range(0, speedRandomMultiplier);

}
```

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Player"))
    {
        pm.SpawnExp(transform.position);
        pm.LowerLives();
        Destroy(gameObject);
    }
    else if (other.CompareTag("Laser"))
    {
        tm.AddScore(100);
        pm.SpawnExp(transform.position);
        Destroy(other.gameObject);
        Destroy(gameObject);
    }
}
```

- Oluşturulan değişkene oyun başlangıcında gerekli atama yapılmıştır.
- Düşman öldürüldüğünde skoru arttıracak fonksiyon tm(TextManager) üzerinden çağırılmıştır (bkz. AddScore(int amount) Fonksiyonu).

PlayerMovement Script Güncellemeleri

```
private TextManager tm;
```

- TextManager türünden bir değişken oluşturulmuştur.

```
void Start()
{
    Debug.Log("Game Started");
    timer= fireCooldown;
    cameraShake = Camera.main.GetComponent<CameraShake>();
    currentSpeed=regularSpeed;
    animator = transform.GetChild(0).GetComponent<Animator>();
    tripTimerImage.gameObject.SetActive(false);
    speedTimerImage.gameObject.SetActive(false);

    tm=GameObject.FindGameObjectWithTag("GameManager").GetComponent<TextManager>();
}
```

```
1 reference
public void LowerLives()
{
    tm.DamagePlayer(lives);
    lives--;
    if (lives <= 0)
    {
        Destroy(gameObject);
        tm.GameoverUIUpdates();

    }

}
```

- Oluşturulan değişkene oyun başlangıcında gerekli atama yapılmıştır.
- Oyuncunun canı azaltıldığında arayüz güncellemelerinin yapılacağı fonksiyon tm(TextManager) üzerinden çağırılmıştır. (bkz. DamagePlayer(int healthPoint))
- Oyuncunun canı 0 olduğunda arayüz güncellemelerini yapılacağı fonksiyon tm(TextManager) üzerinden çağırılmıştır. (Bkz. GameoverUIUpdates() Fonksiyonu)

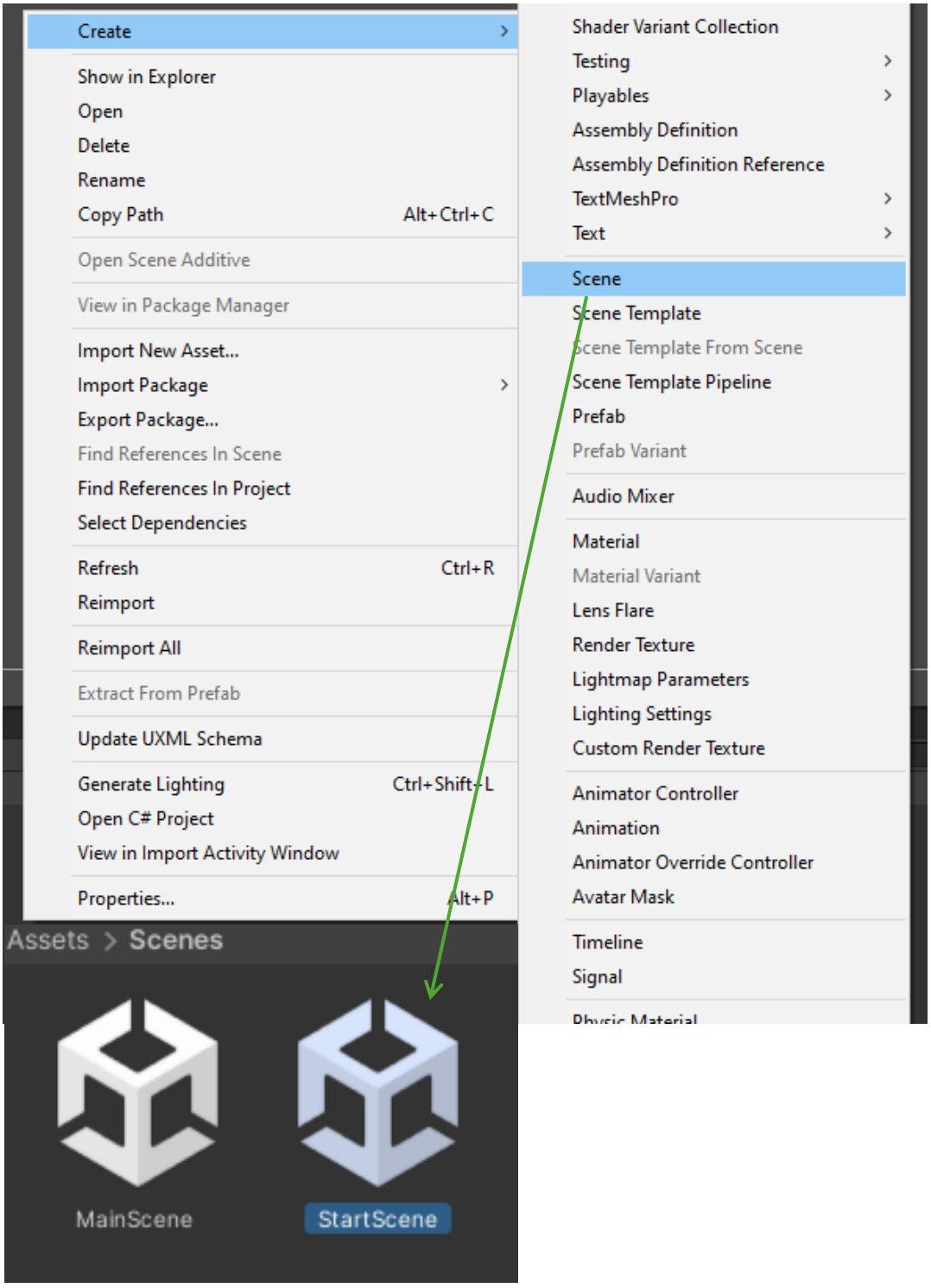
```
2 references
public IEnumerator TripleEffect()
{
    if (tripUIUpdateCoroutine == null)
    {
        tripUIUpdateCoroutine = StartCoroutine(tm.PowerupUIUpdate(tripTimerImage, tripleShotDuration));
    }
    else
    {
        tripTimerImage.gameObject.SetActive(false);
        StopCoroutine(tripUIUpdateCoroutine);
        tripUIUpdateCoroutine = StartCoroutine(tm.PowerupUIUpdate(tripTimerImage, tripleShotDuration));

    }

    yield return new WaitForSeconds(tripleShotDuration);
    tripleShotIsActive = false;
}
```

```
2 references
public IEnumerator SpeedEffect()
{
    currentSpeed = regularSpeed * speedMultiplier;
    animator.SetBool("SpeedIsOn", true);

    if (speedUIUpdateCoroutine == null)
    {
        speedUIUpdateCoroutine = StartCoroutine(tm.PowerupUIUpdate(speedTimerImage, speedDuration));
    }
    else
    {
        speedTimerImage.gameObject.SetActive(false);
        StopCoroutine(speedUIUpdateCoroutine);
        speedUIUpdateCoroutine = StartCoroutine(tm.PowerupUIUpdate(speedTimerImage, speedDuration));

    }

    yield return new WaitForSeconds(speedDuration);

    currentSpeed = regularSpeed;
    speedIsActive = false;
    animator.SetBool("SpeedIsOn", false);
}
```

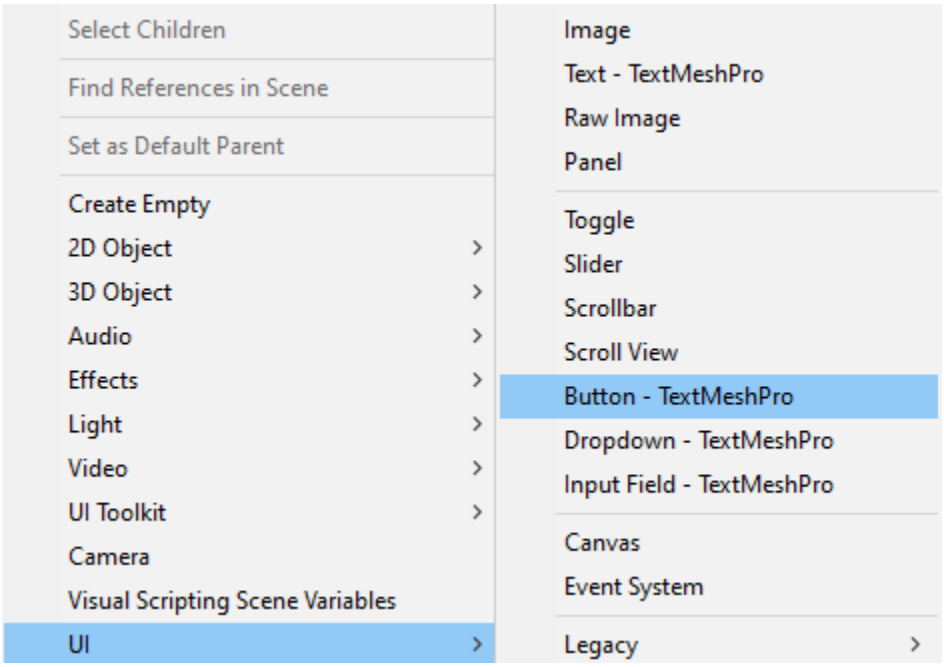
- Güçlendirmelerle etkileşime girildiğinde arayüzde sayaç güncellemelerinin yapılacağı fonksiyonlar tm(TextManager) üzerinden çağırılmıştır ve olası durumların mantığına uygun kod blokları yazılmıştır.

Main Menu-Başlangıç Ekranı

Oyunun ana ekranı için Assets -> Scenes -> Sağ tıklama ->Create ->Scene seçilerek yeni bir sahte oluşturulmuştur ve adı “StartScene” olarak değiştirmiştir. Oluşturulan sahneye çift tıklanarak geçiş yapılır.



Önceki haftalarda anlatıldığı gibi kamera ve sahne arkaplan ayarları yapılır. Hiearchyde Sağ Tıklama->UI içerisinden oyunu başlanacak buton ile oyun ismini yazacağı text eklenir .



Konum ve renk yapıldıktan sonra oyun ekranı bu şekilde gözükmektedir.

Kaynakça

https://docs.unity3d.com/Manual/

https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UIBasicLayout.html

https://chatgpt.com/

Proje Kodu ve Github Repo

Kod: https://github.com/bathuchan/btu-gameprogramming-BatuhanSengul/tree/main/Reports/8.Hafta

Proje Repo: https://github.com/bathuchan/btu-gameprogramming-BatuhanSengul

Hazırlayan

Batuhan Şengül – 20360859008- bathu.sengul@gmail.com