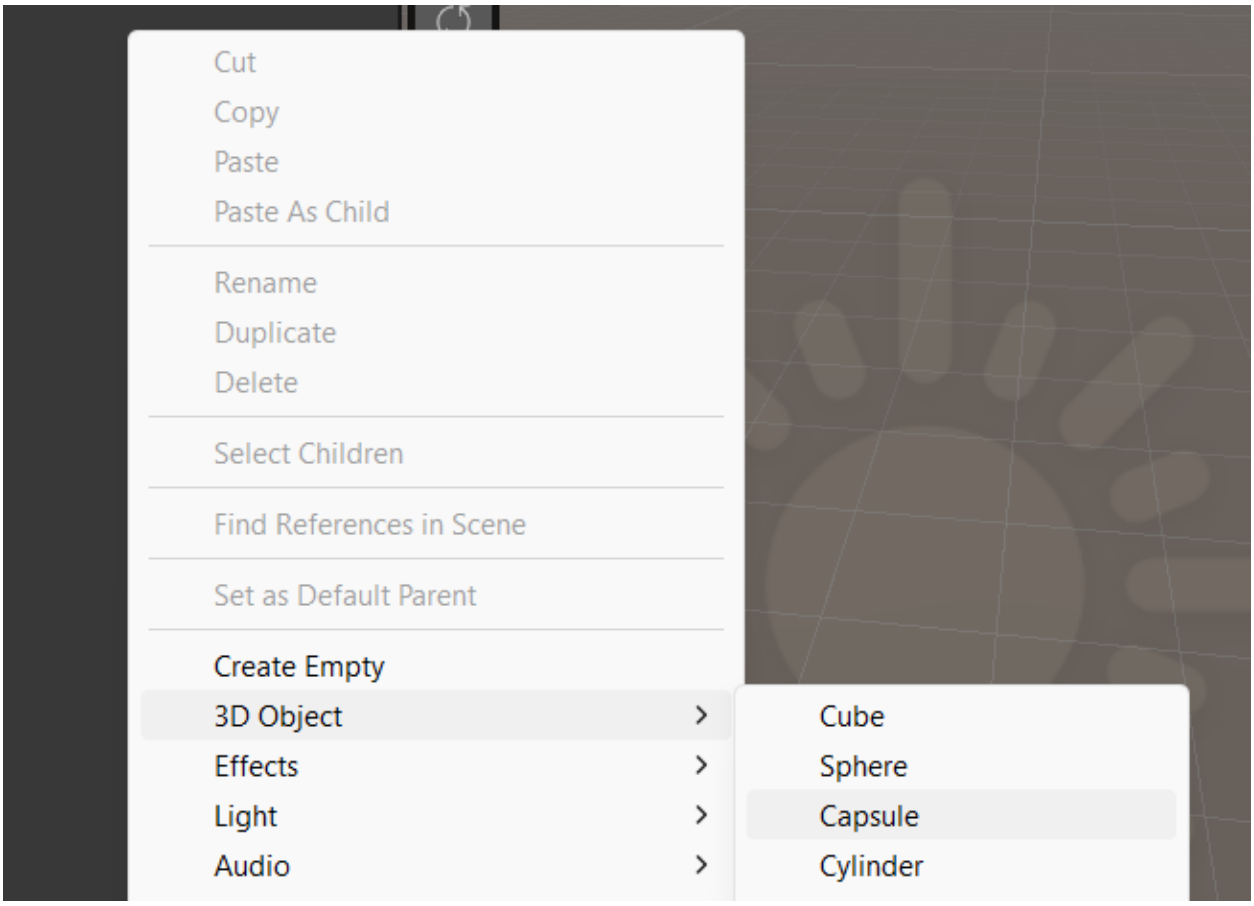
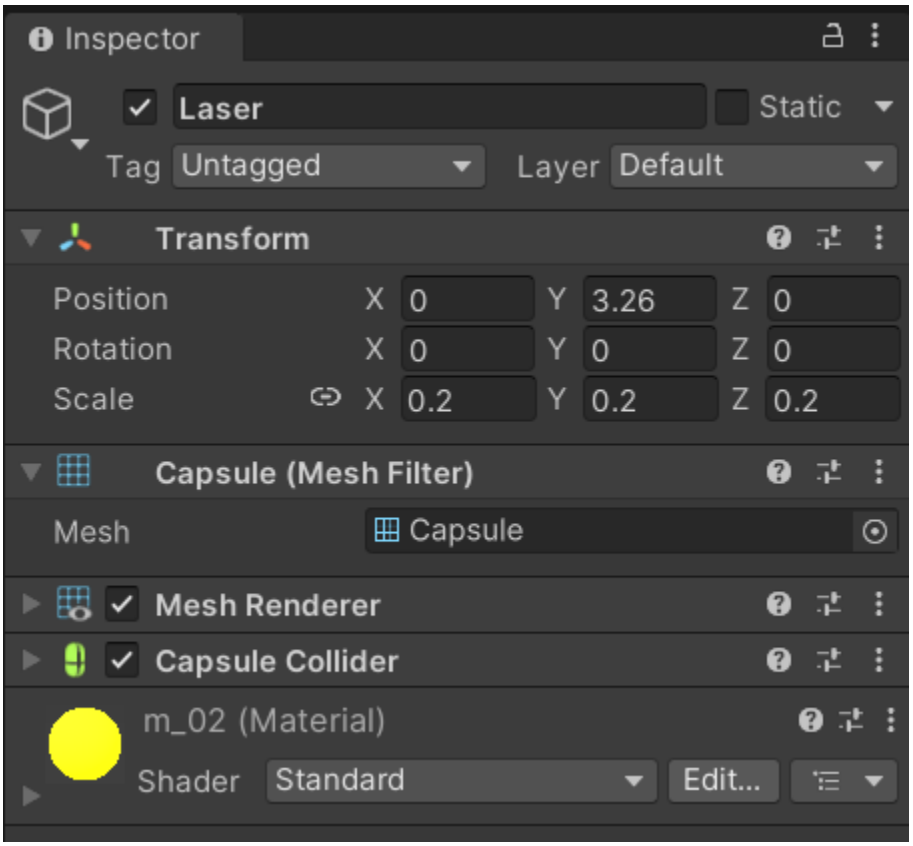
2.Hafta Ders Raporu – Batuhan Şengül

Prefab Oluşturma – “Lazer”

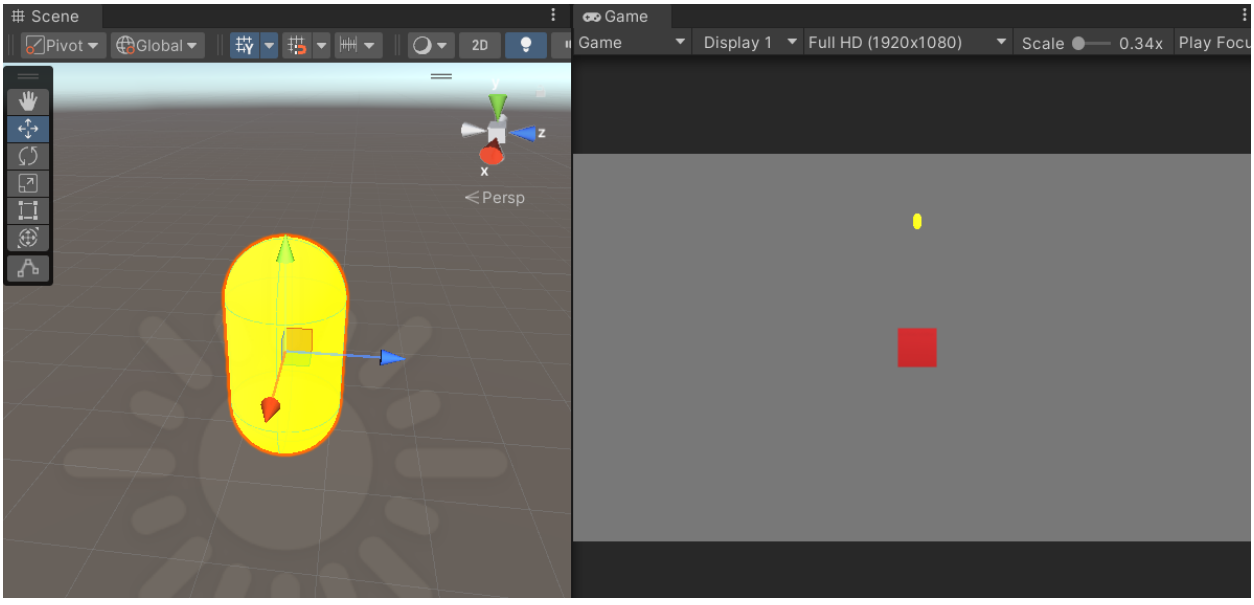
Hierchy sağ tıklama -> 3D object -> Capsule seçilir. Bu örnekte yapacağımız lazerin modelini oluşturmuş oluruz.

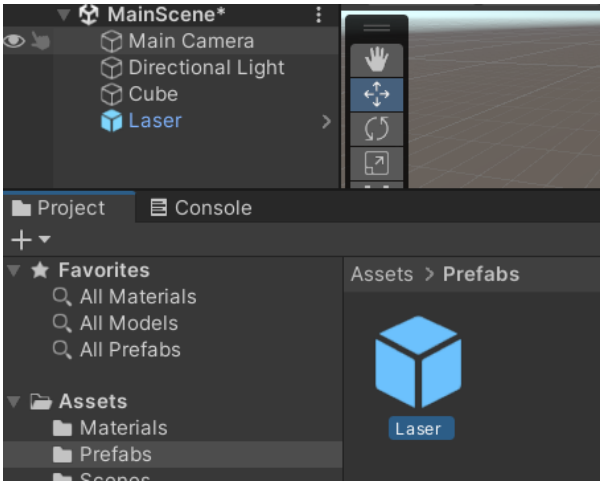


Oluşturulan kapsül oyun nesnesini Inspectordan ismi ve boyutları(scale) düzenlenir ayrıca önceki haftada gösterildiği gibi bir materyal oluşturulup-düzenlenip bu oyun objesine atanır.



Oluşturulan nesne game ve scene pencerelerinde aşağıdaki gibi gözükmektedir.





Oluşturulan oyun nesnesi Hierchy den Project penceresinde önceden oluşturulan "Prefabs" klasörüne sürüklenerek prefab haline getirilir. Eğer Hierchy de oyun nesnesi mavi renkle yazmaya başladıysa başarılı bir şekilde prefab oluşturulmuştur demektir.

Sahne üstündeki lazer oyun nesnesinde yapacağımız değişiklikler prefabde istenirse güncellenebilir ya da geri prefabdaki gibi eski haline getirilebilir.

Lazer

Rigidbody Bileşeni

Rigidbody bileşeni, bir nesnenin fizik motoruyla etkileşime geçmesini sağlayarak, onu gerçek dünya fizik kurallarına (kütle, yerçekimi, kuvvetler, sürtünme vb.) tabi tutar. Bu bileşen, 3D ve 2D oyunlarda fiziksel etkileşimlerin yönetilmesi için çok önemli bir rol oynar. Eğer bir oyun nesnesine **Rigidbody** eklenmişse, bu nesne fiziksel hareketleri, çarpışmaları ve kuvvetleri simüle eder.

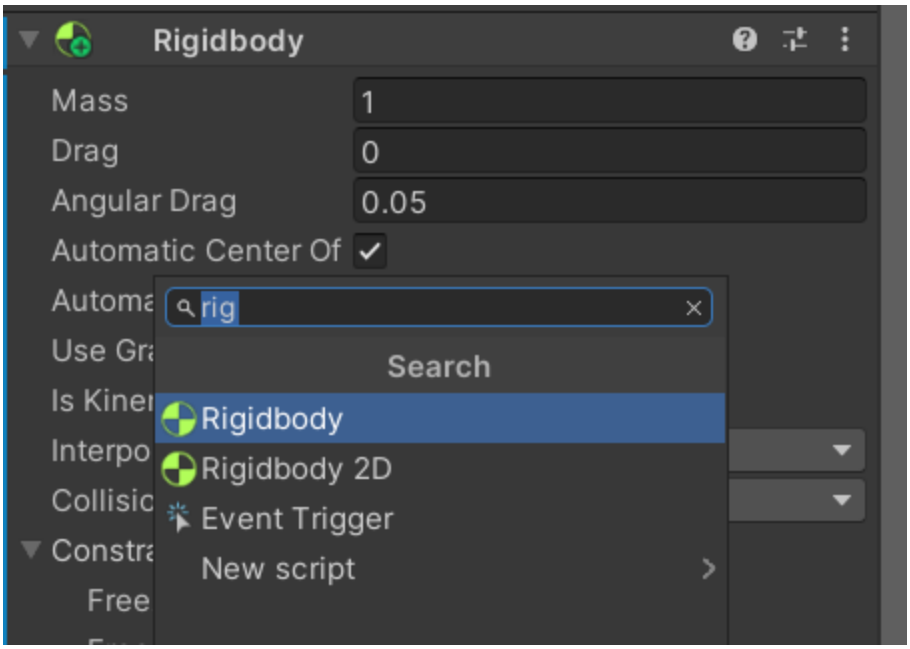
- **Rigidbody İçindeki Ayarlar:**

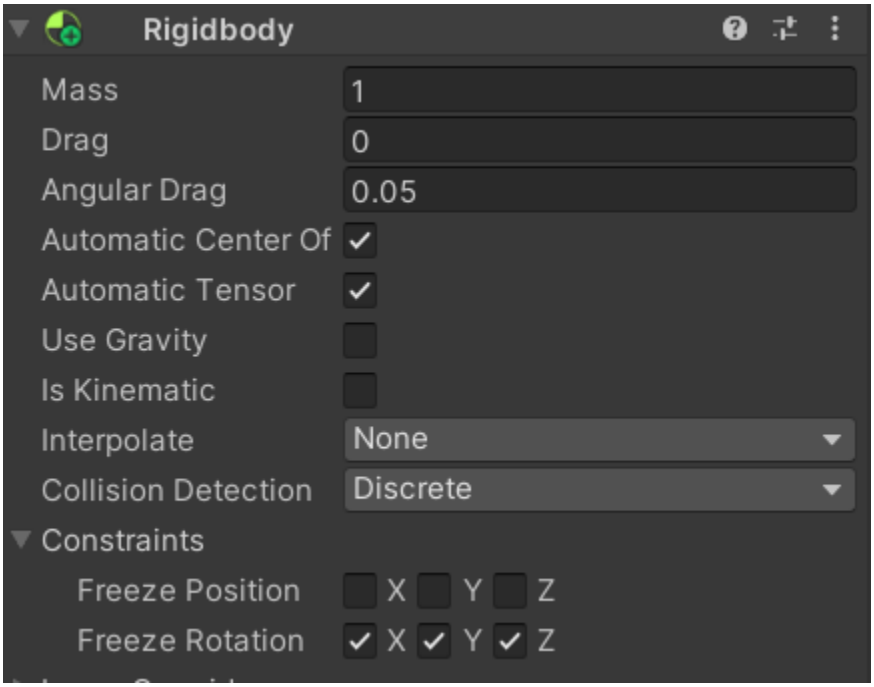
1. **Mass (Kütle):**
    - Nesnenin ağırlığını belirler. Daha büyük kütleli nesneler daha zor hızlanır ve kuvvetlerden daha az etkilenir.
2. **Drag (Sürüklenme):**
    - Nesnenin hızının azalmasını sağlar. Hava direnci veya sürtünmeye benzer. Drag ne kadar yüksekse, nesne o kadar hızlı durur.
3. **Angular Drag (Açısal Sürüklenme):**
    - Nesnenin dönme hızının azalmasını sağlar. Yüksek değerler, nesnenin dönmesinin daha hızlı durmasını sağlar.
4. **Use Gravity (Yerçekimi Kullan):**
    - Bu ayar etkinleştirildiğinde, nesne yerçekimi kuvvetine maruz kalır ve aşağı doğru düşer.

5. **Is Kinematic (Kinematik):**
    - Kinematik hale getirildiğinde nesne, fizik motoru tarafından etkilenmez. Yani çarpışmalar ve kuvvetler ona etki etmez, ancak kod ile kontrol edilebilir.
6. **Interpolate (Enterpolasyon):**
    - Hareketin yumuşaklığını artırır. Nesneler düşük kare hızlarında bile düzgün hareket eder.
        - **None**: Enterpolasyon yok.
        - **Interpolate**: Geçmiş karelerden enterpolasyon yaparak hareketi yumuşatır.
        - **Extrapolate**: Gelecek kareyi tahmin eder ve buna göre hareket eder.
7. **Collision Detection (Çarpışma Algılama):**
    - Nesnelerin çarpışmalarını nasıl algılayacağını ayarlar. Yüksek hızda hareket eden nesnelerin çarpışmaları kaçırmaması için önemlidir.
        - **Discrete**: Varsayılan ayar, sadece çarpışmalar fiziksel olarak temas ettiğinde algılanır.
        - **Continuous**: Yüksek hızlı nesneler için sürekli çarpışma algılaması yapılır. Daha fazla işlem gücü gerektirir.
        - **Continuous Dynamic**: Daha hassas sürekli çarpışma algılaması sağlar, özellikle hızlı hareket eden nesneler için kullanılır.
        - **Continuous Speculative**: Daha az işlemci gücü tüketir, ancak sürekli algılama sunar.
8. **Constraints (Kısıtlamalar):**
    - Nesnenin hareketini veya dönmesini belirli eksenlerde sınırlandırmanı sağlar.
        - **Freeze Position**: X, Y veya Z ekseninde hareketi durdurur.
        - **Freeze Rotation**: X, Y veya Z ekseninde dönmeyi durdurur

Lazerin Fiziği

Örneğe geri dönecek olursak lazer prefabine rigidbody bileşeni eklenir. (Add Component : Rigidbody) Bu bileşen lazere hareket kazandırmak için kullanılacaktır.





Mermier yerçekiminden etkilenmemesi için “Use Gravity” kapatılır. Merminin düz bir şekilde kalması için Rotation (x/y/z) değerleri kısıtlanır.

Script

- **Kullanılan Değişkenler**

```
[SerializeField]
private GameObject laserPrefab;
[SerializeField]
private float projSpeed=10f, projLifespan=1f;
[SerializeField]
float fireCooldown = 0.5f, timer = 0.5f;
```

Private GameObject laserPrefab: Space tuşuna basınca oluşturulacak lazer prefabı.

Private float projSpeed: Lazerin hız çarpanı.

Private float projLifeSpan: Lazerin yok olmadan önce oyunda kalacağı süre.

Private float fireCooldown: Ateş etmeden önceki beklenecek süre.

Private float timer: Sayaç.

- **Start Fonksiyonu**

```
void Start()
{
    Debug.Log("Game Started");
    timer= fireCooldown;
}
```

Sayaç oyun aşladığında beklenecek süreye eşitlenir.

- **Update Fonksiyonu**

```
Unity Message | 0 references
void Update()// fpse bagli calisir
{
    if (fireCooldown >= 0)
    {
        fireCooldown -= Time.deltaTime;
    }

    HandleMovement();//Hareket fonksiyonu
    HandleShoot();//ates etme fonksiyonu
}
```

If bloğu içinde sayaç sürekli azaltılır. Hareket ve Ateş etmeyle ilgilenen fonksiyonlar her karede çağırılır.

- **HandleMovement() Fonksiyonu ve Border**

```
public void HandleMovement()
{

    horizontalVal = Input.GetAxis("Horizontal");
    verticalVal = Input.GetAxis("Vertical");


    //Oyun nesnesinin Sag veya Sola haraketi

    transform.Translate(new Vector3(horizontalVal * speed * Time.deltaTime, verticalVal * speed * Time.deltaTime, transform.position.z));

    //Ekran keanar sinirlari
    transform.position = new Vector3(Math.Clamp(transform.position.x, -xBorderValue, xBorderValue),
        Math.Clamp(transform.position.y, -yBorderValue, yBorderValue),
        transform.position.z);

}
```

Yatay ve dikey klavye girdileri okunur. Oyuncunun pozisyonu okunan değerlere göre güncelenir. Bu yapılan pozisyon değişikliklerinin istenen oyun alanında olup olmadığı Math.Clamp() fonksiyonuyla ayarlanan “BorderValue” değerleriyle kontrol edilir o aralıkta tutulur ve güncellenir.

(BorderValue’lar oyun başlatılarak incelenip ayarlanmıştır.)

```
[SerializeField]
private float xBorderValue,yBorderValue;
```

- **HandleShoot() Fonksiyonu**

```
private void HandleShoot()
{

    if(Input.GetKey(KeyCode.Space) && fireCooldown <= 0)
    {
        fireCooldown = timer;
        GameObject go= Instantiate(laserPrefab, transform.position+Vector3.up, Quaternion.identity);

        if(go.TryGetComponent<Rigidbody>(out Rigidbody rb))
        {
            rb.AddForce(Vector3.up * projSpeed,ForceMode.Impulse);
            Destroy(go,projLifespan);

        }
    }

}
```

If bloğu içinde “Space”e basılıyor mu ve ateş edilme zamanı uygun mu diye kontrol edilir. Eğer iki durumda sağlanırsa, zamanlayıcı sıfırlanır. Oyuncunun pozisyonunun biraz üstünde (transform.position+Vector3.up) mermi nesnesi yaratılır ve “go” GameObjectine atanır.

2. If bloğunda go GameObject’inde rigidbody var mı diye kontrol edilir varsa rb Rigidbody nesnesi oluşur(rb). Oluşan rigidbodye 1 anlığına yüksek bir hız çarpanı uygulanır. Bu hız çarpanı sayesinde lazer belirlenen yöne(Vector3.up) hareket eder.

Destroy içinde de go GameObject’i (yani oluşmuş lazer) projLifespan kadar bir süre sonra yok edilir ve lazer nesnelerinin çok fazla birikerek optimizasyon sıkıntısı çıkarması engellenir.

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/blob/main/Reports/2.Hafta/PlayerMovement.cs

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

Hazırlayan

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