İSKENDERUN TEKNİK ÜNİVERSİTESİ
RAPOR

192523217

GORKEM ARSLANBOGAN

OYUN PROJESİ
BİLGİAYAR MÜHENDİSLİĞİ BÖLÜMÜ

İSKENDERUN TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMERİ FAKÜLTESİ

MAYIS 2023

OYUN KONUSU:

Oyunun konusu diğer uzay araçlarıyla savaşma ve onları silahlarla patlatma.

Oyun ilk açıldığında bizi kaya parçası karşılıyor kaya parçası patladığında ise

uzay araçları gelmeye başlıyor ve birbirlerine ateş edip savaşmaya başlıyorlar,

sağa sola hareket ederek kullanıcı yön tuşlarıyla mermilerden kaçmaya çalışıyor.

Skor ekranı sağ üstte kullanıcı uzay araçlarını patlattıkça skoru artıyor,

kullanıcının 3 adet canı var canı bittiğinde oyun bitiyor ve tekrar başlama ekranına

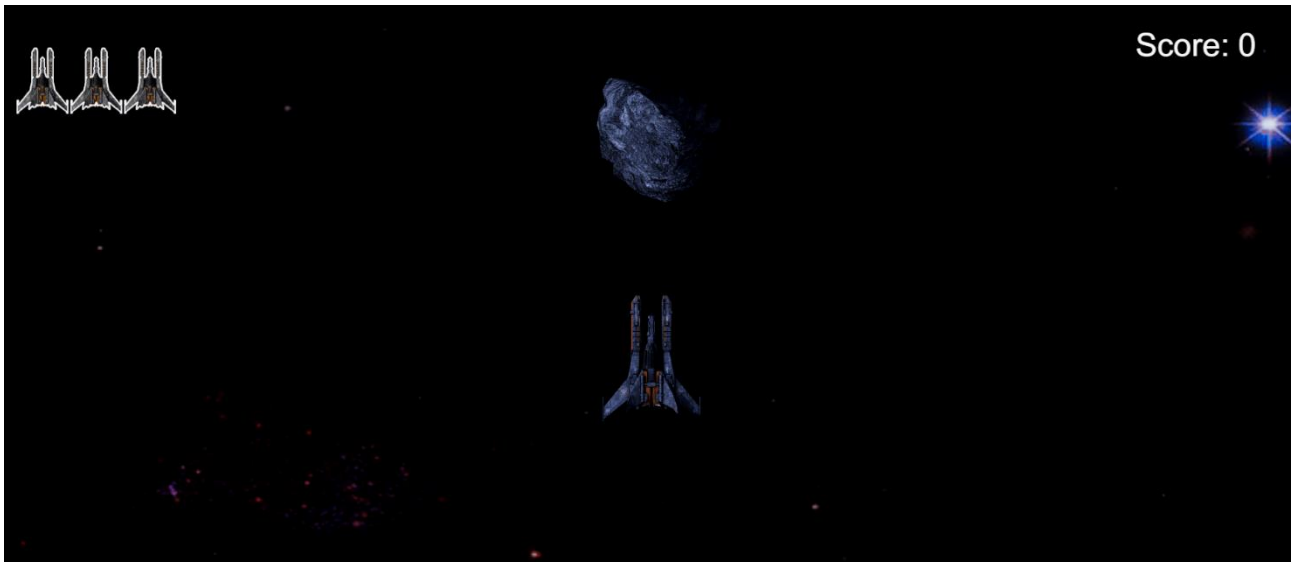
yönlendiriyor oyun.

'R' tuşuna basıldığında ise kullanıcı oyuna baştan başlıyor.

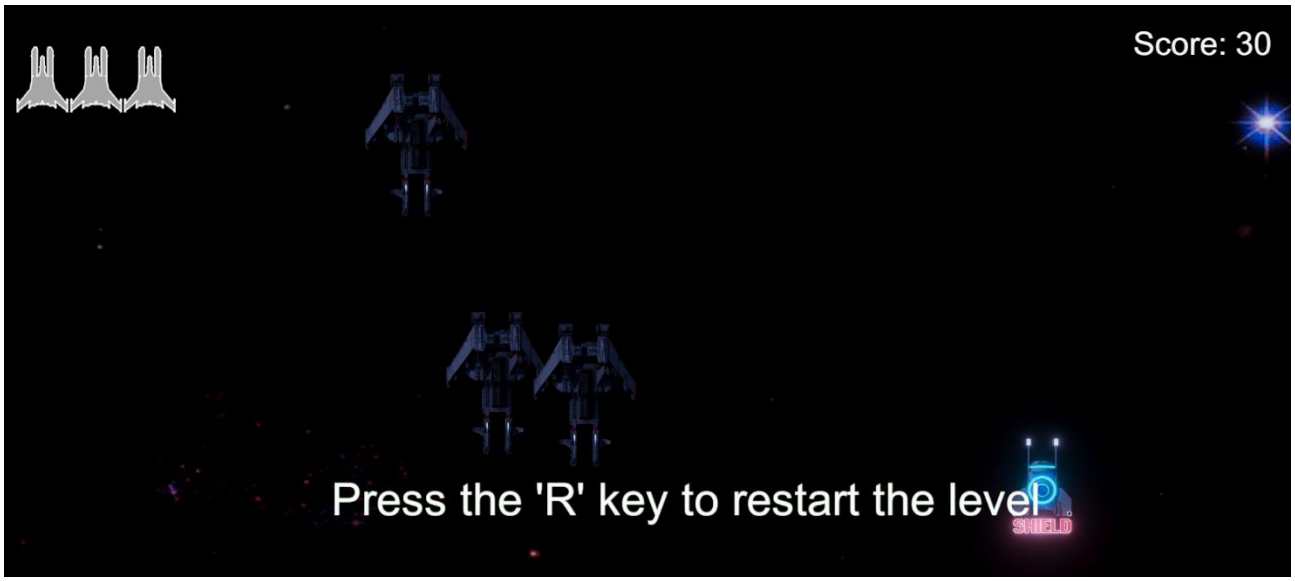
KULLANILAN TEKNOLOJİLER:

- **UnityHub**
- **Visual Studio 2022**
- **Git (Version Control System)**
- **Github (Remote presenter)**

OYUN İÇİ GÖRSELLER




Score: 30
SHIELD
SPEED


Score: 30
Press the 'R' key to restart the level
SHIELD


Score: 30
SPEED


Press the 'R' key to restart the level
SHIELD

Kodlar:

Astroid Taşı kodu:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Asteroid : MonoBehaviour
{
    [SerializeField]
    private float _rotateSpeed = 3.0f;
    [SerializeField]
    private GameObject _explosionPrefab;
    private SpawnManager _spawnManager;
    // Start is called before the first frame update
    private void Start()
    {
        _spawnManager = GameObject.Find("Spawn_Manager").GetComponent<SpawnManager>();
    }

    // Update is called once per frame
    void Update()
    {
        transform.Rotate(Vector3.forward * _rotateSpeed * Time.deltaTime);
    }


    private void OnTriggerEnter2D(Collider2D other)
    {

        if(other.tag == "Laser")
        {

            Instantiate(_explosionPrefab, transform.position, Quaternion.identity);
            Destroy(other.gameObject);
            _spawnManager.StartSpawning();
            Destroy(this.gameObject,0.25f);

        }

    }

}
```

Rakip – Düşman Kodları:

```
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;

public class Enemy : MonoBehaviour
{
    // Start is called before the first frame update
    [SerializeField]
    private float _speed = 4.0f;


    private Player _player;
    private Animator _anim;
    private AudioSource _audioSource;

    void Start()
    {
        _player = GameObject.Find("Player").GetComponent<Player>();
        _audioSource = GetComponent<AudioSource>();
        if(_player==null)
        {

            Debug.LogError("The Player is NULL");

        }
        _anim = GetComponent<Animator>();
        if(_anim == null)
        {

            Debug.LogError("The Animator is NULL");
        }
    }

    // Update is called once per frame
    void Update()
    {
        transform.Translate(Vector3.down * _speed * Time.deltaTime);


        if(transform.position.y < -5f)
        {
            float ramdomx = Random.Range(-8f, 8f);
            transform.position = new Vector3(ramdomx, 7, 0);
        }
    }

    private void OnTriggerEnter2D(Collider2D other)
    {
        if(other.tag == "Player")
        {
            Player player = other.transform.GetComponent<Player>();

            if(player != null)
            {
                player.Damage();
            }
```

```
            _anim.SetTrigger("OnEnemyDeath");
            _speed = 0;
            _audioSource.Play();
            Destroy(this.gameObject,2.8f);

        }

        if(other.tag == "Laser")
        {
            Destroy(other.gameObject);
           if(_player != null)
            {
                _player.AddScore(10);

            }
            _anim.SetTrigger("OnEnemyDeath");
            _speed = 0;
            _audioSource.Play();
            Destroy(this.gameObject,2.8f);

        }

    }
}
```

Player Oyuncu Kodları :

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Player : MonoBehaviour
{
    // Start is called before the first frame update
    [SerializeField]
    private float _speed = 3.5f;
    private float _speedMultiplier = 2;
    [SerializeField]
    private GameObject _laserPrefab;
    [SerializeField]
    private GameObject _tripleShotPrefab;
    [SerializeField]
    private float _fireRate = 0.5f;
    private float _canFire = -1f;
    [SerializeField]
    private int _lives = 3;
    [SerializeField]
    private SpawnManager _spawnManager;




    private bool _isTripleShotActive = false;

    private bool _isSpeedBoostActive = false;

    private bool _isShieldsActive = false;
```

```
[SerializeField]
private GameObject _shieldVisualizer;

[SerializeField]
private GameObject _leftEngine, _rightEngine;

[SerializeField]
private int _score;

private UIManager _uiManager;
[SerializeField]
private AudioClip _laserSoundClip;

private AudioSource _audioSource;

public float horizontalInput;
void Start()
{
   transform.position = new Vector3(0, 0, 0);
   _spawnManager = GameObject.Find("Spawn_Manager").GetComponent<SpawnManager>();
   _uiManager = GameObject.Find("Canvas").GetComponent<UIManager>();
   _audioSource = GetComponent<AudioSource>();

   if(_spawnManager == null)
   {
      Debug.LogError("the spawn Manager is Null .");
   }
   if(_uiManager == null)
   {
      Debug.LogError("The UI Manager is Null .");
   }
   if(_audioSource == null)
   {
      Debug.LogError("AudioSource on the player is NULL");

   }
   else
   {

      _audioSource.clip = _laserSoundClip;
   }

}

// Update is called once per frame
void Update()
{
   CaLculateMovement();

   if (Input.GetKeyDown(KeyCode.Space) && Time.time > _canFire)
   {
      _canFire = Time.time + _fireRate;
      Instantiate(_laserPrefab, transform.position + new Vector3(0, 0.8f, 0), Quaternion.identity);
   }


}
void CaLculateMovement()
{
```

```
    float horizontalInput = Input.GetAxis("Horizontal");
    float verticalInput = Input.GetAxis("Vertical");
    //transform.Translate(Vector3.right * horizontalInput * _speed * Time.deltaTime);
    //transform.Translate(Vector3.up * verticalInput * _speed * Time.deltaTime);

    Vector3 direction = new Vector3(horizontalInput, verticalInput, 0);

        transform.Translate(direction * _speed * Time.deltaTime);

    transform.position = new Vector3(transform.position.x, Mathf.Clamp(transform.position.y, -3.8f, 0), 0);

    if (transform.position.x > 11.3f)
    {
        transform.position = new Vector3(-11.3f, transform.position.y, 0);
    }
    else if (transform.position.x < -11.3f)
    {
        transform.position = new Vector3(11.3f, transform.position.y, 0);
    }
}
 void FireLaser()
{
    _canFire = Time.time + _fireRate;

    if(_isTripleShotActive ==true )
    {
        Instantiate(_tripleShotPrefab, transform.position, Quaternion.identity);

    }
    else
    {
        Instantiate(_laserPrefab, transform.position + new Vector3(0, 1.05f, 0), Quaternion.identity);

    }
    _audioSource.Play();
}



public void Damage()
{
    if(_isShieldsActive == true)
    {
        _isShieldsActive = false;
        _shieldVisualizer.SetActive(false);

    }
    _lives--;
    if(_lives == 2)
    {

        _leftEngine.SetActive(true);

    }
    else if(_lives == 1)
    {

        _rightEngine.SetActive(true);
```

```
        }
        _uiManager.UptadeLives(_lives);

        if(_lives < 1)
        {
            _spawnManager.OnPlayerDeath();
            Destroy(this.gameObject);
        }
    }
    public void TripleShotActive()
    {
        _isTripleShotActive = true;
        StartCoroutine( TripleShotPowerDownRoutine());
    }

   IEnumerator TripleShotPowerDownRoutine()
    {
        yield return new WaitForSeconds(5.0f);
        _isTripleShotActive = false;

    }
    public void SpeedBoostActive()
    {
        _isSpeedBoostActive = true;
        _speed *= _speedMultiplier;
        StartCoroutine(SpeedBoostPowerDownRoutine());


    }
    IEnumerator SpeedBoostPowerDownRoutine()
    {

        yield return new WaitForSeconds(5.0f);
        _isSpeedBoostActive = false;
        _speed /= _speedMultiplier;

    }
    public void ShieldsActive()
    {
        _isShieldsActive = true;
        _shieldVisualizer.SetActive(true);
    }
    public void AddScore(int points)
    {

        _score += points;
        _uiManager.UpdateScore(_score);


    }

}
```

DEVAMI GELECEK..

AÇIKLAMALAR KOD BLOGLARI ÜZERİNDE, ÖNCESİNDE VEYA SONRASINDA YAPILMIŞTIR

KAYNAKÇA

https://www.youtube.com/c/Arrimus3D
https://www.youtube.com/c/gurkankaraman
https://docs.unity3d.com/Manual/index.html
https://www.udemy.com/
https://www.btkakademi.gov.tr/