Android Game

Luis Rodriguez Varona 2018-6309

Tomas Hernández 2018-5509

Programación aplicada.

Informe 3.

Para este proyecto teníamos que realizar una aplicación para Android de un juego simple de un alien que reventaba globos de helio para sobrevivir. Aunque nosotros modificamos un poco esta sinopsis, logramos de manera correcta. Lo implementamos de manera parecida al movimiento de flappy bird ya que solo podíamos movernos en un solo eje(Eje y).

```
//[SerializeField] private Transform arriba = null;
//[SerializeField] private Transform abajo = null;

//private float currentPosition = 0.5f;
//private float speed = 1f;

private const float JUMP_AMOUNT = 2f;

private float verticalInput;

private Rigidbody2D naveRigidBody;


0 referencias
private void Awake()
{
    naveRigidBody = GetComponent<Rigidbody2D>();
}

// Update is called once per frame
0 referencias
private void Update()
{
    if (Input.GetAxis("Fire1") == 1)
    {
        Jump();
    }
}

//private void FixedUpdate()
//{
//    MovePlayer();
//}

1 referencia
private void Jump()
{
    naveRigidBody.velocity = Vector2.up * JUMP_AMOUNT;
    //currentPosition += verticalInput * speed * Time.deltaTime;
    //currentPosition = Mathf.Clamp01(currentPosition);
    //transform.position = Vector3.Lerp(abajo.position, arriba.position, currentPosition);
}
```

Este es el código del movimiento donde se declara un float con el valor de cuanto se va a saltar, luego en una función aparte llamada jump(); se realiza el calculo en el **vertor y *** la cantidad que se había asignado. Luego en Update lo que se hace es llamar esta función constante mente cada vez que se haga tap en la pantalla.

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

1 referencia
public class Globos : MonoBehaviour
{
    public float speed = 1;
    public bool run;
    public GameObject Globo;

    0 referencias
    void Start()
    {
        //transform.Rotate(0f, 0f, -180f, Space.Self);
    }

    // Update is called once per frame
    0 referencias
    void Update()
    {


        transform.Translate(new Vector2(0f, speed * Time.deltaTime));

        gameObject.transform.position = gameObject.transform.position;

    }

    0 referencias
    private void OnTriggerEnter2D(Collider2D col)
    {

        if (col.tag == "DEL")
        {
            Destroy(this.gameObject);
        }

    }

    0 referencias
    public void TakeDamage()
    {
        this.gameObject.SetActive(false);
    }
}
```

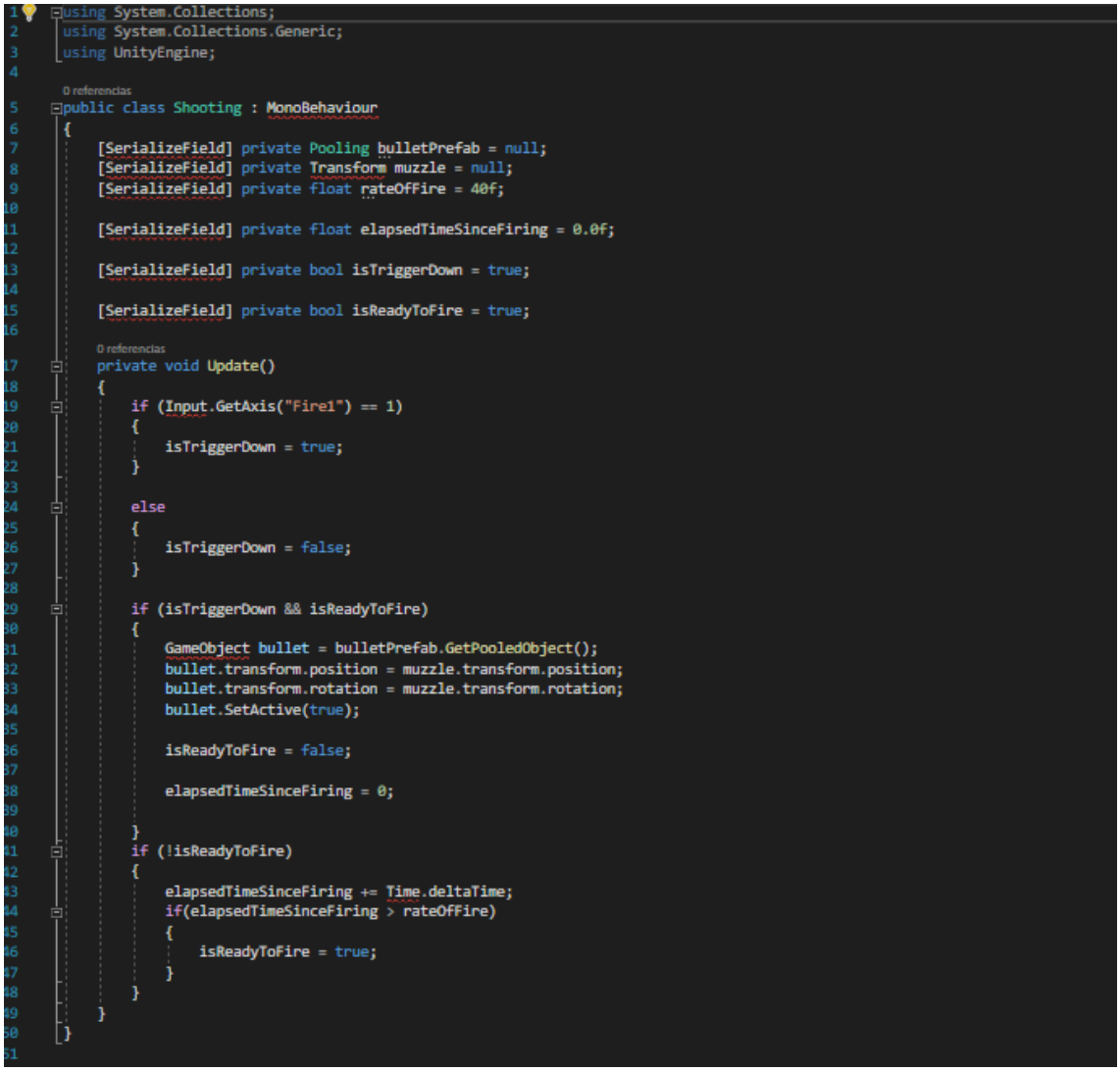
En esta parte simplemente se declaran los globos y su comportamiento. Dígase la velocidad con la que se mueven y se establece cuando se deben destruir al estar en contacto con los proyectiles.

```
0 referencias
public class GenerateBallooon : MonoBehaviour
{
    public Rigidbody2D[] balloons;
    public Transform balloonSpawner;

    private float[] spawnTime = { 1, 1.1f, 1.2f, 1.3f, 1.4f, 1.5f };

    0 referencias
    void Start()
    {
        StartCoroutine("CreateBalloon");
    }

    0 referencias
    IEnumerator CreateBalloon()
    {
        int randomSpawn = Random.Range(0, 6);
        yield return new WaitForSeconds(spawnTime[randomSpawn]);

        while (true)
        {
            Rigidbody2D BalloonInstance;
            int randomNum = Random.Range(0, 5);

            BalloonInstance = Instantiate(balloons[randomNum], balloonSpawner.position, balloonSpawner.rotation) as Rigidbody2D;
            //BalloonInstance.AddForce(-balloonSpawner.right * 750f);

            randomSpawn = Random.Range(0, 6);
            yield return new WaitForSeconds(spawnTime[randomSpawn]);
        }

    }
}
```

Este es el código simple de generación de los globos donde se generan de manera totalmente aleatoria.

```
5 referencias
public class Hellium : MonoBehaviour
{
    public static int heliumLevel;

    0 referencias
    void Start()
    {
        heliumLevel = 30;

        StartCoroutine("DecreaseHeliumLevel");
    }

    0 referencias
    void Update()
    {
        gameObject.GetComponent<Text>().text = heliumLevel.ToString();

        if (heliumLevel <= 0)
        {
            Controller.gameOver = true;
        }
        else if (heliumLevel >= 500)
        {
            Controller.won = true;
        }
    }

    0 referencias
    IEnumerator DecreaseHeliumLevel()
    {
        while (true)
        {
            yield return new WaitForSeconds(10);

            heliumLevel -= 5;
        }
    }
}
```

En esta clase se declaran las condiciones de victorias en base al puntaje, de que a los 500 puntos se gana el juego y al pasar 10 segs el nivel de helio decrece de 5 en 5.



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

0 referencias
public class Shooting : MonoBehaviour
{
    [SerializeField] private Pooling bulletPrefab = null;
    [SerializeField] private Transform muzzle = null;
    [SerializeField] private float rateOfFire = 40f;

    [SerializeField] private float elapsedTimeSinceFiring = 0.0f;

    [SerializeField] private bool isTriggerDown = true;

    [SerializeField] private bool isReadyToFire = true;

    0 referencias
    private void Update()
    {
        if (Input.GetAxis("Fire1") == 1)
        {
            isTriggerDown = true;
        }

        else
        {
            isTriggerDown = false;
        }

        if (isTriggerDown && isReadyToFire)
        {
            GameObject bullet = bulletPrefab.GetPooledObject();
            bullet.transform.position = muzzle.transform.position;
            bullet.transform.rotation = muzzle.transform.rotation;
            bullet.SetActive(true);

            isReadyToFire = false;

            elapsedTimeSinceFiring = 0;

        }
        if (!isReadyToFire)
        {
            elapsedTimeSinceFiring += Time.deltaTime;
            if(elapsedTimeSinceFiring > rateOfFire)
            {
                isReadyToFire = true;
            }
        }
    }
}
```

Esta parte es la encargada de manejar las ballas llamando la clase pool que se encarga de spawnear la balas en la posición necesaria y asi poder llevar el conteo en base a que globo impacta y demás.