Ciclo: Animaciones 3D, Juegos y Entornos Interactivos **Curso:** 2020/21

Módulo: Desarrollo de Entornos Interactivos Multidispositivo

Nombre y apellidos: **Iratxe Domínguez Andrés**

EXAMEN TEÓRICO **Grupo A**

Escribe tu nombre y apellidos en la cabecera de este documento, y a continuación explica qué herramientas vistas a lo largo del curso utilizarías para lograr los objetivos planteados en el ejercicio práctico, explicando por qué esas y no otras:

Interactividad y restricción de movimiento

```
    Primero hay que crear una variable que guarda la velocidad máxima del objeto
    private float velocidad = 3f;

    Segundo hay que crear variables que muestran si el objeto esta dentro de limites y se
puede mover (valor 1) o fuera de los limites y no se puede mover (valor 0)
    private float limX = 1f;
    private float limY = 1f;

En el Update se llama al método LimitadorMovimiento.

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

Se crea un método que va a hacer que la esfera esté limitada en el plano.
    void LimitadorMovimiento()
    {
       En desplX se guarda el desplazamiento horizontal que se genera en el controlador
(teclado o mando)
        float desplX = Input.GetAxis("Horizontal");

        Este "if" significa que estás cerca del límite izquierdo (en este caso -4f) y
sigues intentando moverte a la izquierda (desplX < 0)
        Por lo tanto no puedes moverte (limX = 0f)
        if (transform.position.x <= -4f && desplX < 0)
        {
            limX = 0f;
        }
```

```
        Este "else if" significa que estás cerca del límite derecho (en este caso 4f) y
sigues intentando moverte a la derecha (desplX > 0)
        // Por lo tanto no puedes moverte (limX = 0f)
        else if (transform.position.x >= 4f && desplX > 0)
        {
            limX = 0f;
        }
        En el caso de que no intentes atravesar los limites puedes moverte libremente
(limX = 1f)
        else
        {
            limX = 1f;
        }
```

El metodo transform.Translate() mueve el objeto una pequeña cantidad para dar ilusión de movimiento fluido.

Se utiliza el "Vector3.right" para moverlo a derecha y a izquierda porque Unity utiliza el signo de desplX (positivo o negativo.

"velocidad" indica la velocidad del movimiento y se multiplica por Time.deltaTime para que el objeto no salga disparado.

transform.Translate(Vector3.right * Time.deltaTime * velocidad * desplX * limX);

Para que se mueve alante y atrás se hace igual poniendo Vector3.forward, pero me fallan los números de dentro. No sé muy bien cómo cambiarlos para z.

```
    // Para el eje Y se realiza de la misma forma cambiando las variables
    float desplY = Input.GetAxis("Vertical");

    if (transform.position.y <= 0f && desplY < 0)
    {
      limY = 0f;
    }
    else if (transform.position.y >= 4f && desplY > 0)
    {
      limY = 0f;
    }
    else
    {
      limY = 1f;
    }
    transform.Translate(Vector3.forward * Time.deltaTime * velocidad * desplY * limY);
  }
  IEnumerator Tiempo()
  {
    int n;
    for(n=0; ; n++)
    {
      myText.text = "Tiempo transcurrido:" + n;

      yield return new WaitForSeconds(1f);
    }
  }


    }
```

Seguimiento del jugador con la cámara

Hay que asociar un script a la cámara.

Se crea una variable tipo GameObject que va a hacer referencia al objeto que va a seguir (en este caso a la esfera) y una tipo Vector3 privada para el desplazamiento de la cámara.

En el método Start el desplazamiento (la he definido antes) va a ser igual a la posición que ya existe menos la posición de la esfera.

En el método Update he metido que la posición inicial va a ser igual a la posición de la esfera más el desplazamiento. Metiéndolo aquí, conseguiré que la cámara siga en todo momento a la esfera.

Para terminar hay que ir a Unity e ir a Main Camera. Ahí aparece Esfera (variable pública) y ahí tengo que asociar el objeto Sphere para que así funcione.

Creación de elementos (columnas) de forma aleatoria

Instanciar un prefab (Script Prefab_creator): Es el codigo para crear un número determinado de prefabs dentro de un bucle for y les hemos cambiado ligeramente su posición en X e Y utilizando el indice (n) del bucle for y un número aleatorio (randomPosY).

```
    {

    El Script estaría asociado a un empty object y solo sirve para crear prefabs.
    //En esta variable se va a guardar desde unity (hay que meterlo en el empty object, llamado instanciador) el prefab que voy a instanciar.
    [SerializeField] GameObject myPrefab;

    //Esta variable nos indica la posicion inicial en la que se instancia el prefab, hay que enlazar el instanciador para que sea la misma posicion
    [SerializeField] Transform initPos;

    //Esta variable va a guardar el cambio de posicion respecto a initPos que va a tener cada prefab.
    Vector3 newPos;

    // Start is called before the first frame update
    void Start()
    {
        //Esto es un bucle for para generar 10 prefabs.
        for(int n = 0; n <10; n++)
        {
            //Generacion de un numero aleatorio para la posicion Y del prefab.
            float randomPosY = Random.Range(-1f, 1f);
```

```
            //Se genera un vector que guarda el desplazamiento de cada prefab.
            //En este caso, utiliza el indice del bucle (n) para que cada prefab salga un
poco mas lejos que el anterior (2*n) en el eje X.
            //Tambien se utiliza un numero aleatorio (randomPosY) para cambiar la posicion
vertical del prefab en el eje Y.
            Vector3 desplazarPos = new Vector3(2*n, randomPosY, 0);

            //Se suma la posicion inicial al desplazamiento para calcular la posicion
final de cada prefab.
            newPos = initPos.position + desplazarPos;

            //La funcion Instantiate crea un nuevo prefab cada vez que se la llama.
Utiliza tres parametros (myPrefab, newPos y Quaternion.identity).
            //myPrefab es el objeto que se va a instanciar (siempre es un GameObject)
            //newPos es la posicion donde se va a instanciar cada vez (siempre es un
Transform o un Vector3)
            //Quaternion.identity es la orientacion con la que aparece el prefab, en este
caso ponemos .identity para que nos muestre el objeto sin rotacion.
            Instantiate(myPrefab, newPos, Quaternion.identity);
        }
    }

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

    }
   }

   }
```

User Interface (tanteos)

CORRUTINA TIEMPO

Se crea un texto en Unity.

En el script de la Esfera primero hay que crear una librería (using UnityEngine.UI) para que reconozca el texto. Después hay que crear una variable serializada para así poder luego en Unity poner el texto (Text myText).

En el método Start, hay que llamar a la corrutina (Tiempo) que voyh a crear para que se ejecute.

Creo la corrutina (IEnumerator…) ya que es asíncrona. Dentro de esta corrutina creo un bucle inifinito con for (es infinito porque el segundo espacio lo dejo en blanco).

int n;

for(n=0; ;n++){

Aquí hay que poner el texto o lo que queramos que se vea en pantalla.

MyText.text="Tiempo:"+n;

Y para finalizar quiero que me devuelva el texto cada segundo (1f).

yield return new WaitForSeconds (1f);

}

Colisiones:

Aquí hay que tocar el Mesh Renderer para que no se destruya, que solo desaparezca.

Este código muestra el funcionamiento básico de una colisión con el método OnCollisionEnter(). Es necesario añadir el componente Rigid Body al objeto en cuyo script vaya a ir el método del colisionador (OnCollisionEnter()). Este método se ejecuta automáticamente cuando el objeto colisiona.

{

Este tipo de colisiones presentan mas informacion que otras (OnTriggerEnter).
// El metdodo "OnCollisionEnter" se llama automaticamente cuando el objeto asociado a este script CON RIGID BODY entra en contacto con otro elemento.

//La variable myMesh de la clase MeshRenderer se crea para enlazar el renderizado del objeto y poder desactivarlo cuando colisiona
[SerializeField] MeshRenderer myMesh;

// Start is called before the first frame update
void Start()
{

}

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

}

private void OnCollisionEnter(Collision collision)
{
// Dentro de este metodo se realizan todas las acciones que van a suceder en el momento de la colision

```
    // Desaparece el renderizado 3D del objeto para dar la ilusion de que ha desaparecido el objeto pero sin perder sus propiedades
    myMesh.enabled = false;
  }
    }
```

En este caso, la colisión genera la desaparición del renderizado (Mesh) del GameObject para generar la ilusión de que se ha destruido pero sigue estando en la escena.

Entrega y evaluación

Cuando tengas completo el documento, expórtalo a pdf con este formato:

Apellidos_nombre_ExTco1EV.pdf

Guárdalo en el misma carpeta que el documento Word, dentro del repositorio, y súbelo a un commit de GitHub.