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

Módulo: Desarrollo de Entornos Interactivos Multidispositivo

Nombre y apellidos: **Luna Jiménez Matheu**

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

Para mover la esfera crearemos unas variables al inicio del script. La primera será una privada para restringir su acceso desde otros lugares. Declaramos la nueva variable de Vector3 y metemos la posición de destino con sus coordenadas en el eje que sea (por ejemplo Z) añadiendo 1 a la posición.

Ej:

Vector3 (0, 0, 1);

La segunda es una pública de tipo float con una velocidad ya asignada, por ejemplo:

vel = 5f;

Es la que va a determinar la velocidad del objeto en movimiento.

Interactividad:

En el void Update tendríamos un float llamado despX con un GetAxis que estaría asignado a un control (un mando) dependiendo de cual, sería horizontal, o vertical etc.

(En este caso es horizontal)

float despX = Input.GetAxis ("Horizontal")

Después se le aplica un método transform al objeto (la esfera) en su propiedad de traslación (translate) dentro de void Update para indicarle a Unity que es una acción que debe realizarse de manera constante cada frame si es llamada.

Para evitar que la bola salga "volando" añadimos a la línea de código un Time.deltaTime que se guíe por la velocidad determinada en la variable float para que el desplazamiento del objeto sea constante mientras no se le aplique ninguna modificación más de velocidad. Sería como aplicarle la velocidad "estándar".

Time.deltaTime * vel);

Ahora, para limitar el movimiento del objeto dentro de un escenario acotado, creamos una variable serializada o privada float desde la cual podamos llamar a una booleana yponer un límite de posición en los ejes, de manera que si la esfera se encuentra dentro de los límites, el valor fuera por ej. 1 y se podría desplazar, y si sale de los mismos el valor sería 0, impidiendo el movimiento. Ej:

```
private float  limitX  = 1f;

private float limitZ   = 1f;
```

La booleana estaría hecha con las condiciones siguientes:

Si la posición del objeto es mayor o menor que los valore numéricos establecido como límite en en el eje X entonces el desplazamiento se bloquea.

```
If (transform.position.x <= 5f &&  despX  < 0)

limitX = 0f;

else

limitX = 1f;
```

Seguimiento del jugador con la cámara

Para hacer que la cámara siga al objeto o al personaje, creamos una variable que contenga la posición de la cámara.

```
[SerializedField] Transform cubePos;
```

En el void Update creamos un Vector3 con un targetPos que contenga todos los ejes del cubePos.

Ej:

```
Vector3 targetPosition = new Vector3(cubePos.position.x,cubePos.position.y,
transform.position.z);
```

Si quisiéramos hacer que la cámara lo siga con suavidad, añadiríamos una variable float que determinase un número de desplazamiento concreto y una Vector3 que llamase a la velocidad de la cámara. Ej:

```
[SerializeField] float smoothVelocity = 0.3F;
private Vector3 camaraVelocity = Vector3.zero;
```

Y añadimos esto mismo en el Update justo debajo del Vector3 de antes

```
transform.position = Vector3.SmoothDamp(transform.position, targetPosition, ref
camara  Velocity, smoothVelocity);
```

Creación de elementos (columnas) de forma aleatoria

Para hacer que aparezcan columnas primero creamos un empty object en Unity. Esto sirve como instanciador para los prefabs de las columnas. En ese script aplicamos una variable con una clase serializada que permita acceder a elementos de otros scripts llamando al Game

Object para que instancie el prefab. Algo así:

(Esta clase permitiría acceder a otros scripts públicos y privados)

[SerializeField] GameObject myPrefab;

Después, para hacer que aparezcan varias de forma aleatoria, haremos un bucle dentro de void Start, para que se aplique desde el primer momento en el que inicia el juego.

Crearíamos dos variables al inicio del script, una con la posición inicial del prefab, y otra que sería la nueva posición del mismo.

En el bucle for haríamos aparecer un número determinado de prefabs con números enteros (int) + n.

Después haríamos un float que haga que la posición de los prefabs al aparecer sea aleatoria en el eje X RandomPosX con un número aleatorio ubicado entre dos valores Ej: (-5 f, 5 f).

Añadimos un Vector3 para guardar el desplazamiento de los prefabs y se hace uso de los n del bucle for para que cada columna aparezca a una distancia de la siguiente y se aplica también un randomPos en el eje X para que cada vez que salgan sea en un lugar distinto.

```
Vector3 desplazarPos = new Vector3(2*n, randomPosX, 0);
```

También pueden usarse corrutinas, que son elementos que funcionan en paralelo a otros script y que se repiten cada cierto tiempo.

Para crear una corrutina, hacemos un bucle dentro de un Ienumerator con el nombre de la corrutina y la llamamos en el Start. Ej:

```
StartCoroutine ("Corrutina");
```

User Interface (tanteos)

Podemos crear un contador si usamos variables dentro de la función Update. Al principio del script declararíamos la variable de tipo float que establecerá el margen de tiempo (cada 1 segundo por ej.)

```
private  float     Intervale   = 1.0f;
```

Luego iríamos al Update y restaríamos 1 a ese tiempo con deltaTime

```
Intervale -= Time.deltaTime;
if (Intervale  < 0)
```

Colisiones:

Para hacer una colisión entre dos objetos, haríamos una variable llamada myMesh dentro de la clase MeshRenderer que uniese el render de los items con el código para poder desactivarlo al chocar.

Se pueden hacer con OnTriggerEnter u OnCollisionEnter, este último otorgando más información acerca del GameObject con el que se va a realizar la colisión.

Al hacer una colisión, se puede tanto destruir el objeto (perdiendo sus propiedades) como eliminar su renderizado (conservándolas).

Esto se haría aplicándole a la variable myMesh un OnCollisionEnter

```
private void OnCollisionEnter (Collision collision)
myMesh.enabled = false;
```

Para este método es necesario que el objeto asociado a este script tenga un Rigid Body aplicado al chocar con otro elemento.

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.