Aspectos tecnológicos en Proyectos de Esports

Clase 3

Clase 3

Aspectos tecnológicos en Proyectos de Esports

Proyecto 3D

01

Creación del ambiente 3D
Creación del personaje e incorporación de movilidad

Proyecto RV

02

Configuración SDK Google Cardboard
Selección de objetos con mirada
Interacción con UI

Proyecto RA

03

Unity AR Foundation - ARCore
Configuración ambiente RA

Clase 3

Proyecto 3D 01

Creación del ambiente 3D
Creación del personaje e incorporación de movilidad

Proyecto RV 02

Configuración SDK Google Cardboard
Selección de objetos con mirada
Interacción con UI

Proyecto RA 03

Unity AR Foundation - ARCore
Configuración ambiente RA

Aspectos tecnológicos en Proyectos de Esports

Creación del ambiente 3D

Una vez creado el proyecto 3D, creamos el **terreno** de juego mediante, por ejemplo, la utilización de un plano: **3D Objects → Plane**

Se le da un nombre, se configura su posición (origen) y su tamaño (escala).

Se agregan otros objetos (por ej. cubos) para que actúen como **obstáculos** en la escena, y el personaje pueda moverse entre ellos. Se les da un nombre y se configura su posición (origen) y su tamaño (escala).

También se pueden agregar otros objetos desde el **AssetStore** de Unity.

Creación del ambiente 3D

Es posible aplicar **materiales** a los objetos creados para darles un color o textura.

Para esto creamos una carpeta "**Materials**" y dentro creamos algunos nuevos materiales. A cada material le asignamos un color (Albedo).

También podemos crear un material especial, que llamaremos SkyBox, para usarlo de horizonte.

En el **shader** lo configuramos como **SkyBox→Procedural** (como el que creo Unity) y podemos variarle los colores del cielo y la tierra.

Creación del ambiente 3D

Para finalizar debemos añadir el SkyBox al ambiente.

En el menú **Window→Rendering→Lighting**, pestaña "**Enviroment**", arrastramos nuestro material a **SkyBox Material**.

Ahora si modificamos el material, se verá reflejado en el horizonte.

Para finalizar, acomodamos la **cámara** en el punto de perspectiva deseado.

Creación del personaje e incorporación de movilidad

Creamos el **personaje principal**, que será manejado por el **usuario**. En este caso se usará una forma simple como lo es una cápsula: **3D Objects→Capsule**

Se le da un nombre, se configura su posición (más arriba que el suelo, para que pueda verse completamente) y su tamaño (escala). Si se desea se le aplica también un material.

Agregamos al personaje el componente **RigidBody** (física). Deshabilitamos la rotación (**Constraints → Freeze rotation**) en todos los ejes para que sea más simple. Ahora el personaje puede colisionar contra otros objetos.

Creación del personaje e incorporación de movilidad

Creamos una carpeta “**Scripts**” y añadimos un nuevo script C# para el movimiento del personaje. Luego lo incorporamos al personaje.

Agregamos las variables privadas y públicas que vamos a necesitar:

```
private Rigidbody rb;          // ref. al rigidbody del personaje
public float speed = 10f;
public float rotationSpeed = 300;
public float jumpForce = 7;
```

En **Start**, obtenemos el componente rigidbody y lo guardamos:

```
rb = GetComponent<Rigidbody>();
```

Creación del personaje e incorporación de movilidad

En **Update**, obtenemos la entrada del usuario y luego actualizamos la posición del personaje solo 1 vez por segundo (no por frame).

Además, multiplicamos el traslado por la variable **speed**, así podemos controlar su velocidad.

```
//valores posibles: <- es -1, -> es 1, NADA es 0
float horizontal = Input.GetAxisRaw ("Horizontal");
float vertical = Input.GetAxisRaw ("Vertical");

transform.Translate (new Vector3 (horizontal, 0.0f, vertical) * Time.deltaTime * speed);
```

Creación del personaje e incorporación de movilidad

Para que el personaje este siempre en el centro de la pantalla, debemos ligar la cámara al personaje. Esto se logra simplemente **arrastrando la cámara al personaje**.

Una vez que la cámara esta ligada al personaje, podemos permitirle al jugador **modificar la perspectiva de visualización**.

Para permitir rotar la perspectiva, vamos a tener en cuenta los movimientos del mouse y el valor almacenado en la variable **rotationSpeed**.

Creación del personaje e incorporación de movilidad

Además, le vamos a permitir al jugador **realizar un salto con la barra espaciadora**.

En **Update**, agregamos:

```
float rotationY = Input.GetAxis ("Mouse X");
transform.Rotate (new Vector3 (0, rotationY * Time.deltaTime * rotationSpeed, 0));

if (Input.GetKeyDown (KeyCode.Space)) {
        Jump ();
}
```

Creación del personaje e incorporación de movilidad

La función **Jump()** realizará un salto teniendo en cuenta el valor configurado en la variable **jumpForce**.

```
void Jump()
{
        rb.AddForce(new Vector3(0, jumpForce, 0), ForceMode.Impulse);
}
```

Clase 3

Aspectos tecnológicos en Proyectos de Esports

Proyecto 3D

01

Creación del ambiente 3D
Creación del personaje e incorporación de movilidad

Proyecto RV

02

Configuración SDK Google Cardboard
Selección de objetos con mirada
Interacción con UI

Proyecto RA

03

Unity AR Foundation - ARCore
Configuración ambiente RA

Configuración SDK Google Cardboard

Existe una guía de inicio rápido para aprender a configurar Google Cardboard en Unity, y se puede encontrar en:

https://developers.google.com/cardboard/develop/unity/quickstart

Configuración SDK Google Cardboard

Antes de comenzar, hay que verificar desde el **Unity Hub** si se tiene instalados los módulos necesarios de Android en la versión de Unity que vayan a utilizar.

Esto es porque el proyecto estará destinado a ese tipo de plataforma, ya que utilizaremos el Cardboard.

Los módulos necesarios son tres: **Android Build Support**, **Android SDK/NDK Tools** y **OpenJDK**.

Configuración SDK Google Cardboard

Creamos un **proyecto 3D** en Unity, y lo primero que hay que hacer es integrar el **SDK Google Cardboard**.

Para hacerlo desde Unity, importamos el SDK desde el **Package Manager**, haciendo uso de la siguiente **url de git**: https://github.com/googlevr/cardboard-xr-plugin.git

También es posible elegir y descargar la versión del SDK que deseemos desde la siguiente url: https://github.com/googlevr/cardboard-xr-plugin. Una vez bajado, lo descomprimimos y ponemos dentro de la carpeta **Packages** de nuestro proyecto.

Configuración SDK Google Cardboard

Una vez incorporado el SDK, nos dirigimos al menú **Google Cardboard XR→Samples** dentro de la ventana del "**Package Manager"**, e importamos "**Hello Cardboard**".

Luego, abrimos la escena "**Hello Cardboard**", y guardamos la escena **HelloCardboard** en nuestra carpeta de escenas (**File→Save as...**).

De esta forma, partiendo desde esta escena básica que contiene la funcionalidad de realidad virtual, es posible ir modificando la escena y armando nuestro propio proyecto VR.

Configuración SDK Google Cardboard

Vamos a “**File → Build Settings**”, eliminamos las escenas a buildear que pueden estar agregadas por defecto y agregamos la escena que acabamos de guardar presionando el botón “**Add Open Scenes”**, ya que será la única escena con la que vamos a trabajar. Si nuestra app tuviera varias escenas, deberían estar todas agregadas en esta lista.

A su vez, debemos cambiar ciertas propiedades del proyecto. Seleccionamos la **plataforma** a utilizar, en este caso **Android**, ya que nuestro proyecto es para celulares con pantalla dividida (una vista para cada ojo).

Luego presionamos el botón “**Switch Platform**” para hacer efectivo el cambio.

Configuración SDK Google Cardboard

Antes de continuar, debemos establecer algunas opciones de configuración (ver https://developers.google.com/cardboard/develop/unity/quickstart):

Desde la pantalla de “**Player Settings**”, en la solapa "**Player**":

- Dar un nombre a la **empresa** y el **producto** (necesario para cuando se genere la app)
- **Resolution & Presentation**:
  - Desactivar Optimized Frame Pacing. Orientación = landscape.
- **Other Settings**:
  - Identification: desactivar Override default package name, Min API -> 8 (api 26)
  - Configuration: Scripting backend = IL2CPP (no MONO), API compatibility = .NET framework, activamos arm64.

Configuración SDK Google Cardboard

- **Publishing settings**:
  - Build: activamos CustomMainGradleTemplate, CustomGradlePropertiesTemplate.

También en la ventana de "**Projects Settings**", pero en el menú "**XR Plug-in Management**", debemos modificar lo siguiente:

- En la solapa de "**Android**" debemos elegir el dispositivo "**Cardboard XR Plugin**"

Configuración SDK Google Cardboard

Por último, debemos hacer algunas modificaciones en los archivos de configuración que agregamos manualmente:

- Agregar **dependencias** en "Assets/Plugins/Android/**mainTemplate.gradle**":

```
implementation 'androidx.appcompat:appcompat:1.6.1'
implementation 'com.google.android.gms:play-services-vision:20.1.3'
implementation 'com.google.android.material:material:1.6.1'
implementation 'com.google.protobuf:protobuf-javalite:3.19.4’
```

- Agregar **propiedades** en "Assets/Plugins/Android/**gradleTemplate.properties**":

```
android.enableJetifier=true
android.useAndroidX=true
```

Configuración SDK Google Cardboard

Ahora si, ya esta todo configurado para poder utilizar y buildear la escena **HelloCardboard**.

Esta escena presenta un **mini-juego** de demostración que presenta las funciones principales del SDK Cardboard.

En el juego, los usuarios miran alrededor de un mundo virtual para **encontrar y recolectar objetos**.

Siempre es posible ver un único objeto en pantalla que, al **apuntarlo y clickearlo**, es recolectado. En ese momento aparece el nuevo objeto en una posición aleatoria de la escena.

Configuración SDK Google Cardboard

Si observamos la escena, podemos ver que contiene varios componentes:

1. **CubeRoom**: es el ambiente en donde van apareciendo los objetos a encontrar.

   - Contiene un script llamado “**Cardboard startup**” que se encarga de iniciar y gestionar el plugin de VR.

2. **PointLight**: se encarga de gestionar la iluminación del ambiente. Es posible cambiar los valores para crear otro tipo de iluminación.

Configuración SDK Google Cardboard

3. **Treasure**: contiene los objetos con los cuales es posible interactuar. Notar que todos los componentes están deshabilitados menos uno, que será el que se visualice al iniciar el juego. Esto es para que siempre haya un único objeto a encontrar.

   - Cada objeto tiene el script "**ObjectController**" que gestiona la interacción. Esta preparado para que cuando miramos un objeto cambie de color y si presionamos el botón del cardboard el objeto cambia de posición.

Configuración SDK Google Cardboard

- Si vemos el código del script “**ObjectController**”, podemos ver que realiza las siguientes tareas:
  - **OnPointerEnter()**: esta función maneja lo que se debe hacer cuando un objeto es apuntado. En este caso, solo le cambia el color al objeto.
  - **OnPointerExit()**: esta función maneja lo que se debe hacer cuando un objeto deja de ser apuntado. En este caso, le vuelve a poner el color original al objeto.
  - **OnPointerClick**(): esta función maneja lo que se debe hacer cuando un objeto es seleccionado. Mediante un llamado a la función TeleportRandomly(), se encarga de desactivar al objeto, tomar otro objeto al azar y seleccionar una posición aleatoria para el nuevo objeto.

Configuración SDK Google Cardboard

4. **Player**: es el componente jugador. Dentro de este componente se encuentra la cámara principal (Main Camera).

   - La cámara a su vez contiene un componente **CardboardReticlePointer**, que sirve para mostrar el retículo en el mundo virtual (un punto en el centro del campo de visión del usuario) que sigue el movimiento de la cabeza del usuario y se utiliza para interactuar con los objetos en el entorno VR.

Configuración SDK Google Cardboard

5. **GraphicsAPIText**: es un componente de texto 3D que es usado únicamente para mostrar información sobre la API para renderizar los gráficos 3D utilizada.

    - Contiene un script "**GraphicsAPITextController**" que muestra el nombre de la API o el mensaje "Unrecognized Graphics API" si no la puede reconocer.

Clase 3

Proyecto 3D 01

Creación del ambiente 3D
Creación del personaje e incorporación de movilidad

Proyecto RV 02

Configuración SDK Google Cardboard
Selección de objetos con mirada
Interacción con UI

Proyecto RA 03

Unity AR Foundation - ARCore
Configuración ambiente RA

Aspectos tecnológicos en Proyectos de Esports

Unity AR Foundation - ARCore

AR Foundation es un marco de trabajo creado por Unity para el desarrollo de experiencias con realidad aumentada. Incluye funciones centrales de ARKit, ARCore, Magic Leap y HoloLens, así como otras funciones únicas de Unity.

ARCore es la plataforma creada en 2017 por Google para crear experiencias de RA. ARCore permite que se pueda utilizar un dispositivo móvil para analizar y detectar su entorno. Al entender el mundo que lo rodea, es posible incorporar información aumentada en la escena real.

Unity AR Foundation - ARCore

ARCore

ARCore es capaz de hacer **tres tareas claves** para incorporar RA:

- **Detectar el movimiento**, para entender su posición relativa dentro del entorno.
- **Entender los alrededores**, detectando el tamaño y ubicación de las diferentes superficies horizontales, verticales y anguladas.
- **Estimar las condiciones de iluminación**. De esta manera, las imágenes de realidad aumentada que muestre se relacionan con la luz de una manera lo más realista posible.

Configuración ambiente RA

Para comenzar con el ejemplo debemos crear un **proyecto 3D en plataforma Android** (File → Build Settings → Android → Switch Platform), y luego incorporar el package de Realidad Aumentada desde el Unity Registry.

Vamos a "**Windows→Package Manager**", seleccionamos ver los paquetes del **Unity Registry**, e instalamos los paquetes de la categoría **AR**.

Luego debemos entrar a “Player Settings” para modificar algunas opciones.

Configuración ambiente RA

En la solapa "**Player**“ dentro de “Projects Settings” debemos configurar:

- **Other Settings**:
  - Rendering: desactivar Auto Graphics API, y luego remover Vulkan
  - Identification: Min API -> 8 (api 26)
  - Configuration: Scripting backend = IL2CPP (no MONO), activamos arm64.

En la solapa " **XR Plug-in Management**“ dentro de “Projects Settings” debemos activar el Plug-in Providers de **ARCore** para Android.

Configuración ambiente RA

Para visualizar un elemento 3D en realidad aumentada, debemos agregar dos componentes a la escena:

- **Sesión de RA**. Este componente controla el ciclo de vida de la experiencia en realidad aumentada.

- **Origen de sesión de RA**. Este componente transforma las características rastreables (superficies, planos y puntos) en su posición, orientación y escala con respecto a la escena.

Configuración ambiente RA

Procedemos a crear la sesión de RA mediante “**XR→AR Session**”.

También hacemos lo mismo con el origen de sesión AR, mediante “**XR→AR Session Origin**” (de acuerdo a la versión que usemos, se puede llamar también “XR Origin”)

Este último componente **trae incorporado su propia cámara** para poder realizar las transformaciones necesarias, por lo que no vamos a necesitar la “Main Camera” que viene por default en la escena, y podemos eliminarla.

Configuración ambiente RA

Con estos dos componentes incorporados al proyecto, se visualizará en RA cualquier elemento 3D que incorporemos a la escena.

Agregamos algún objeto a la escena, como un **plano**, un **cubo** o una **esfera**, y si lo deseamos creamos un **material** y se lo asignamos, para que tenga un color determinado.

Configuramos su posición en el **origen**, pero **cambiamos su coordenada Z** con un valor positivo (por ejemplo 1), para que aparezca delante de nuestra posición al iniciar la aplicación. También podemos cambiar sus valores de **escala** y **rotación**.

Configuración ambiente RA

Si no deseamos que se vea siempre un objeto en la escena, sino que el objeto aumentado aparezca al visualizar una imagen o QR, entonces hay que agregar al "**Origin**" un componente "**AR Tracked Image Manager**".

Este componente necesita una lista de imágenes de referencia a ser "**trackeadas**" y los objetos **prefab** que debe mostrar al visualizar dichas imágenes de referencia. Creamos dentro de Assets una lista desde "**XR-> Reference Object Library**" y agregamos en dicho componente las imágenes que deseemos.

Luego, solo falta pasarle al "**AR Tracked Image Manager**" la lista creada y un prefab para visualizar.

Configuración ambiente RA

Para probar la aplicación, debemos generar el archivo instalable del proyecto.

Vamos a “**File→Build Settings**”, agregamos la escena abierta (si aún no está en la lista), presionamos “**Build**”, y guardamos el archivo **APK** que nos generará en algún lugar del disco. Luego copiaremos este archivo en el dispositivo e instalaremos la app.

Otra alternativa es tener el dispositivo conectado mediante USB, y directamente instalar la app desde Unity presionando “**File→Build and Run**”.

Visualización 3D

Vimos tres ejemplos básicos de como crear una **app 3D**, una **app 3D con RV** y una **app 3D con RA**. A partir de estos ejemplos sencillos, pueden empezar a generar sus propios proyectos.