

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

			PRÁCTICA DE LABORATORIO		
CARRERA: COMPUTACION			ASIGNATURA: HIPERMEDIAL		
NRO. PRÁCTICA:	8	TÍTULO PRÁCTICA: Desarrollo de una aplicación de realidad virtual usando la herramienta Unity y desplegada en un dispositivo móvil Android.			
OBJETIVO <ul style="list-style-type: none">• Experimenta con aplicaciones de realidad virtual.• Experimenta con aplicaciones de realidad aumentada.• Distingue la diferencia entre tecnologías de realidad virtual y realidad aumentada.					
INSTRUCCIONES		<p>Este proyecto es una oportunidad para que combine y practique todo lo que aprendió en el capítulo de Realidad Virtual. Creará una experiencia de realidad virtual totalmente interactiva en forma de laberinto.</p> <p>Esto le brinda la oportunidad de aplicar lo que ha aprendido con las secuencias de comandos para brindar una experiencia audiovisual completa.</p> <p>Crear la GVR Camera Rig</p> <p>Durante este paso, crearemos la cámara VR incluyendo el GvrEditorEmulator en la escena y configurando la cámara.</p> <ol style="list-style-type: none">1. Agregue el objeto prefab GvrEditorEmulator a la escena.2. Convierta el objeto Main Camera en un elemento hijo del objeto GvrEditorEmulator.3. Restablece el componente Transformar del objeto Main Camera4. Mueva el objeto GvrEditorEmulator a una ubicación conveniente para el desarrollo, por ejemplo, Posición: 0, 3, 35 y Rotación: 0, 180, 0.5. Ingrese al modo de juego y use Alt + Mouse y Ctrl + Mouse para rotar e inclinar el ángulo de visión de la cámara. <p>Preparando la escena para la interacción</p> <p>Durante este paso, prepararemos la escena para la interacción configurando el puntero, el emisor de rayos y el sistema de eventos, y luego probaremos el sistema de punto de referencia incluido (waypoint).</p> <ol style="list-style-type: none">1. Agregue el objeto prefab GvrReticlePointer a la escena como elemento secundario del objeto del Main Camera.2. Aumente el valor de Distancia máxima de retícula para el GvrReticlePointer del valor predeterminado de 10 a 20.			

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

3. Agregue el script GvrPointerPhysicsRaycaster como componente en el objeto Main Camera.
4. Agregue el objeto prefab GvrEventSystem a la escena.
5. Ingrese al modo de juego y navegue por la escena haciendo clic en los puntos de referencia.

Hacer que los objetos del juego sean interactivos


Durante este paso haremos que la Moneda, la Llave y la Puerta sean interactivas añadiéndoles componentes de disparadores y eventos.

1. Localiza y selecciona el objeto Coin en la jerarquía.
2. Verifique que tenga un componente Collider.
3. Agregue el script Coin proporcionado como componente.
4. Agregue un disparador de evento (Event Trigger) como componente.
5. Agregue el evento PointerClick al componente Event Trigger.
6. Asigne el componente del script Coin al campo de objeto del evento Pointer Click.
7. Asigne el método Coin.OnCoinClicked () como la función para el evento Pointer Click.
8. Ingrese al modo de juego, haga clic en la moneda y verifique que el mensaje 'Coin.OnCoinClicked ()' esté impreso en la ventana de la consola.
9. Repita el mismo proceso para el objeto del juego Key pero use el script Key y el método Key.OnKeyClicked ().
10. Repita el mismo proceso para el primer objeto principal del juego de Puerta, pero use el script de Puerta y el método Door.OnDoorClicked ().

Hacer la interfaz de usuario interactiva

Durante este paso, haremos que el objeto SignPost sea interactivo al agregarle componentes de disparadores y eventos. El proceso es casi idéntico al que hicimos con la moneda, la llave y la puerta en el paso anterior, pero no necesitamos un colisionador para interactuar con los objetos del juego de la interfaz de usuario. En su lugar, debemos verificar que el objeto del juego Canvas tenga un componente Graphic Raycaster, y debido a que estamos usando GVR, reemplazaremos el Graphic Raycaster de Unity con el GvrPointerGraphicRaycaster de Google VR.

1. Localiza y selecciona el objeto del juego SignPost en la jerarquía.
2. Elimine el componente Graphic Raycaster que se agrega automáticamente al crear un nuevo objeto.
3. Agregue el script GvrPointerGraphicRaycaster como componente.
4. Agregue el script SignPost proporcionado como componente.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

5. Agregue un Event Trigger como componente.
6. Agregue el evento PointerClick al componente Event Trigger.
7. Asigne el componente script SignPost al campo de objeto del evento Pointer Click.
8. Asigne el método SignPost.ResetScene () como la función para el evento Pointer Click.
9. Ingrese al modo de juego, haga clic en SignPost y verifique que el mensaje 'SignPost.ResetScene ()' esté impreso en la ventana de la consola.

Programando el comportamiento de la moneda (coin)


Durante este paso, programaremos el comportamiento de la moneda para que, cuando se haga clic en una moneda (coin), se reproduzca un sonido, muestre un efecto de "poof" y desaparezca.

1. Hay un mínimo de cinco monedas en el laberinto. Cuando se hace clic en una moneda, se reproduce un efecto de sonido en la ubicación de esa moneda. Cuando se hace clic en una moneda, esa moneda se elimina de la jerarquía de la escena.
2. Abra el script de Coin y lea todo el script, incluidos todos los comentarios.
3. Dedique un tiempo a comprender el comportamiento del objeto prefab CoinPoof proporcionado. Puede hacer esto, por ejemplo, ingresando al modo de juego y arrastrando un objeto prefab CoinPoof a la escena.
4. Programe el comportamiento de la moneda completando todos los comentarios TODO en el script.

Programando el comportamiento de la llave (key)

Durante este paso, programaremos el comportamiento de la llave (key) para que, cuando se haga clic en la llave, reproduzca un sonido, muestre un efecto de "poof" y desaparezca.

1. Hay un mínimo de una llave en el laberinto. Cuando se hace clic en la llave, se reproduce un efecto de sonido en la ubicación de la llave. Cuando se hace clic en la llave, la llave se elimina de la jerarquía de la escena. Cuando se hace clic en la llave, la puerta se desbloquea.
2. Abra el script key y lea completo, incluidos todos los comentarios.
3. Dedique un tiempo a comprender el comportamiento del objeto prefab KeyPoof proporcionado. Puede hacer esto, por ejemplo, ingresando al modo de juego y arrastrando un objeto prefab KeyPoof a la escena.
4. Programe el comportamiento key completando todos los comentarios TODO en el script.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Programando el comportamiento de la puerta (door)

Durante este paso, programaremos el comportamiento de la puerta (door) para que cuando se haga clic en la llave (key), la Puerta se desbloquee y cuando se haga clic en la Puerta, se escuche un sonido y comience a girar a una posición de apertura.

1. La puerta evita que el usuario navegue al objeto SignPost hasta que se haya abierto. Cuando comienza el juego, la puerta está bloqueada y cerrada. La puerta no se puede abrir cuando está bloqueada. La puerta solo puede desbloquearse haciendo clic en la llave. Cuando se hace clic y se desbloquea la puerta, la puerta comienza a abrirse. Cuando la puerta comienza a abrirse, se reproduce un efecto de sonido en la ubicación de la puerta. La puerta se anima a una posición de abierta solo por código, es decir, no mediante el uso de animación y controlador de animador.
2. Abra el script Door y lea todo el script, incluidos todos los comentarios.
3. Agregue un componente AudioSource al primer objeto padre Door y desactive la propiedad Play On Awake.
4. Asigne el audio Door_Opening al campo AudioClip.
5. Programe el comportamiento de la puerta completando todos los comentarios TODO en el script.

Programando el comportamiento del SignPost

Durante este paso, programaremos el comportamiento de SignPost para que cuando se haga clic en SignPost se reinicie el juego.

1. El SignPost no se puede ver ni interactuar antes de que se abra la puerta. Cuando se hace clic en SignPost, la escena se restablece a su estado inicial para que el juego se pueda volver a jugar.
2. Abra el script SignPost y lea todo el script, incluidos todos los comentarios.
3. Programe el comportamiento SignPost completando todos los comentarios TODO en el script.


Crear la funcionalidad del juego

Durante este paso, armaremos todo y convertiremos nuestro proyecto en un juego real.


Laberinto

- El laberinto está diseñado de tal manera que el usuario no puede identificar una ruta a la llave desde la posición de inicio.

Waypoints

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

	<ul style="list-style-type: none"> Los puntos de referencia se colocan en todo el laberinto de tal manera que los usuarios pueden navegar desde la posición inicial a todos los objetos del juego con los que se puede interactuar, es decir, todas las monedas, la llave, la puerta y el SignPost. <p>Monedas</p> <ul style="list-style-type: none"> Hay un mínimo de cinco monedas en el laberinto. <p>Llave</p> <ul style="list-style-type: none"> Hay un mínimo de una clave en el laberinto. <p>Puerta</p> <ol style="list-style-type: none"> La puerta evita que el usuario navegue hasta SignPost hasta que se haya abierto. <p>SignPost</p> <ul style="list-style-type: none"> El cartel no se puede ver ni interactuar antes de que se abra la puerta. <ol style="list-style-type: none"> Use los objetos de juego Maze para diseñar un laberinto. Mueva el jugador, es decir, el objeto del juego GvrEditorEmulator, a la ubicación de inicio deseada. Agregue más Waypoints para que el jugador pueda navegar a todas las partes del Laberinto. Agrega más monedas para que el jugador tenga muchas monedas para coleccionar. Mueva la llave a una ubicación para que sea un tanto difícil para el jugador encontrarla. Agregue paredes (cubos) para asegurarse de que los usuarios tengan que navegar para encontrar la llave y las monedas.
ACTIVIDADES POR DESARROLLAR	

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

CTIVIDADES DESARROLLADAS

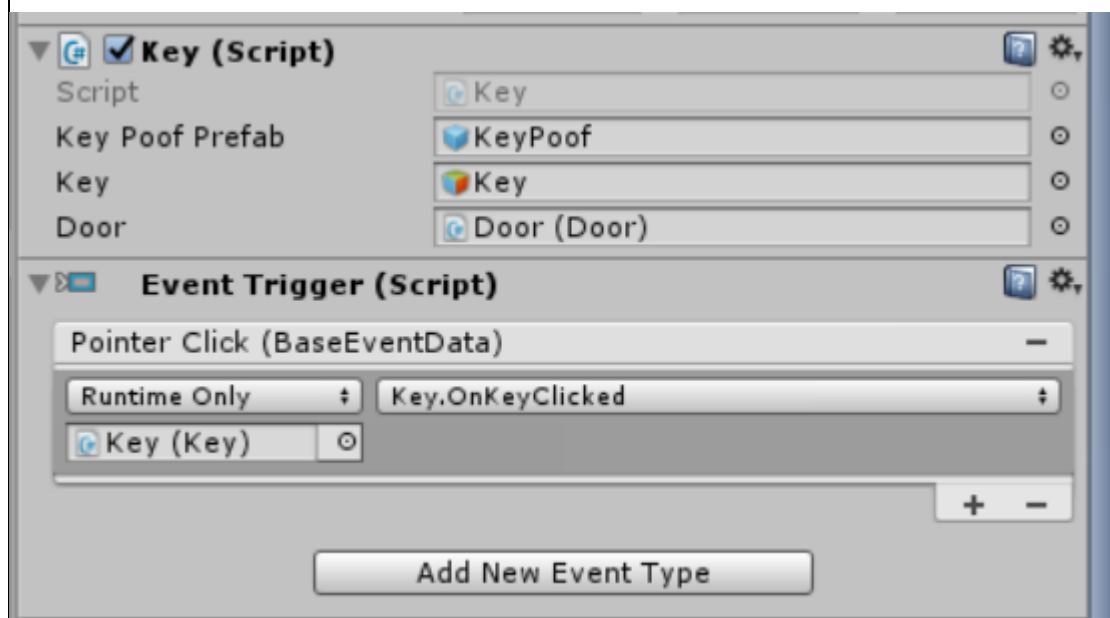
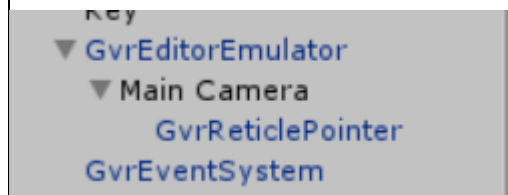
1. Desarrollar cada uno de los requerimientos planteados siguiendo las instrucciones brindadas en la guía de prácticas.


En la guía de prácticas se detalla el proceso a seguir para la elaboración de este juego; en el presente informe se seccionarán las instrucciones en 3 partes: preparación de la escena y objetos, programación de comportamientos y diseño del laberinto. En los siguientes puntos se detallará cada sección.

2. Preparar la escena y los objetos del juego.

En esta parte se realiza la preparación del ambiente; de manera concreta, se puede decir que en esta primera parte se agregan los objetos y componentes necesarios para que el ambiente y los objetos sean interactivos.

En primer lugar, se agrega el GvrEditorEmulator (que vendría siendo la cámara), el GvrReticlePointer (el puntero). Además, se agregan scripts a los objetos que serán animados posteriormente, así como también los triggers que controlarán dichos comportamientos.



	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

3. Programar los comportamientos de los objetos.

En esta sección se explicará acerca de la programación de los comportamientos (animaciones o acciones) que tendrán los objetos principales del juego y con los cuales el jugador va a interactuar.

A. Programación del comportamiento de las monedas – Coins.

En sí, se pide que cuando el jugador “toque” las monedas estas desaparezcan mientras se muestra una animación de “humo” y un sonido. Además, se pide como un reto que las monedas giren sobre su eje mientras se ejecuta el juego. Para ello se programa en el script ya definido en el proyecto, el cual en la sección anterior se agregó como componente al objeto Coin.

```

4
5 public class Coin : MonoBehaviour {
6
7     // TODO: Create variables to reference the game objects we need access to
8     // Declare a GameObject named 'coinPoofPrefab' and assign the 'CoinPoof' prefab to the field in Unity
9     public GameObject coinPoofPrefab;
10    public GameObject coin;
11
12    void Update () {
13        // OPTIONAL-CHALLENGE: Animate the coin rotating
14        // TIP: You could use a method from the Transform class
15        coin.transform.Rotate(0, Time.deltaTime * 180, 0, Space.World);
16    }
17
18
19    public void OnCoinClicked () {
20        /// Called when the 'Coin' game object is clicked
21        /// - Displays a poof effect (handled by the 'CoinPoof' prefab)
22        /// - Plays an audio clip (handled by the 'CoinPoof' prefab)
23        /// - Removes the coin from the scene
24
25        // Prints to the console when the method is called
26        Debug.Log ("Coin.OnCoinClicked() was called");
27
28        // TODO: Display the poof effect and remove the coin from the scene
29        // Use Instantiate() to create a clone of the 'CoinPoof' prefab at this coin's position and with the 'CoinPoof' prefab's rotation
30        // Use Destroy() to delete the coin after for example 0.5 seconds
31        GameObject clonCoinPoof = (GameObject) Object.Instantiate(coinPoofPrefab, coin.transform);
32        Object.Destroy(coin, 0.8f);
33    }
34

```

B. Programación del comportamiento de la llave – Key.

En sí, se pide que cuando el jugador “toque” la llave esta tenga un comportamiento similar al que tienen las monedas. La única diferencia, es que también se llamará a un método de un objeto externo (en este caso el objeto Door). Para ello se programa en el script ya definido en el proyecto, el cual en la sección anterior se agregó como componente al objeto Key.

```

5 public class Key : MonoBehaviour {
6
7     // TODO: Create variables to reference the game objects we need access to
8     // Declare a GameObject named 'keyPoofPrefab' and assign the 'KeyPoof' prefab to the field in Unity
9     // Declare a Door named 'door' and assign the top level 'Door' game object to the field in Unity
10    public GameObject keyPoofPrefab;
11    public GameObject key;
12    public Door door;
13
14    void Update () {
15        // OPTIONAL-CHALLENGE: Animate the key rotating
16        // TIP: You could use a method from the Transform class
17        key.transform.Rotate(0, Time.deltaTime * 180, 0, Space.World);
18    }
19
20
21    public void OnKeyClicked () {
22
23        // Prints to the console when the method is called
24        Debug.Log ("Key.OnKeyClicked() was called");
25        //Debug.Log(keys);
26
27        // TODO: Unlock the door, display the poof effect, and remove the key from the scene
28        // Use 'door' to call the Door.Unlock() method
29        // Use Instantiate() to create a clone of the 'KeyPoof' prefab at this coin's position and with the 'KeyPoof' prefab's rotation
30        // Use Destroy() to delete the key after for example 0.5 seconds
31
32        door.Unlock();
33
34        GameObject clonKeyPoof = (GameObject)Object.Instantiate(keyPoofPrefab, key.transform);
35        Object.Destroy(key, 0.8f);
36    }
37 }

```

C. Programación del comportamiento de la puerta – Door.

La programación de la puerta es la más compleja de todas. Se debe validar que se haya recolectado la llave, para que pueda comenzar con la animación de “apertura”. Además, como retos opcionales se pide reproducir otro sonido si la puerta está bloqueada y al comenzar la animación de apertura se desactiven los “Colliders” para evitar la interacción desde ese punto.


```

43
44 void Start () {
45     AudioSource = GetComponent<AudioSource>();
46     leftDoorStartRotation = leftDoor.transform.rotation;
47     leftDoorEndRotation = leftDoorStartRotation * Quaternion.Euler(0, 0, -90);
48     rightDoorStartRotation = rightDoor.transform.rotation;
49     rightDoorEndRotation = rightDoorStartRotation * Quaternion.Euler(0, 0, 90);
50 }
51
52 void Update () {
53     if (opening){
54         leftDoor.transform.rotation = Quaternion.Slerp(leftDoorStartRotation, leftDoorEndRotation, timer / rotationTime);
55         rightDoor.transform.rotation = Quaternion.Slerp(rightDoorStartRotation, rightDoorEndRotation, timer / rotationTime);
56         timer = timer + Time.deltaTime;
57     }
58 }
59
60 public void OnDoorClicked() {
61     if (locked == false) {
62         opening = true;
63         AudioSource.Play();
64
65         foreach (var collider in door.GetComponentsInChildren<BoxCollider>()){
66             collider.enabled = false;
67         }
68
69     } else {
70         AudioSource.PlayClipAtPoint(lockedAudio, door.transform.position);
71     }
72 }
73
74 public void Unlock () {
75     locked = false;
76 }
77 }
78

```

D. Programación del comportamiento del cartel – SignPost.

El comportamiento de este “cartel”, es la de volver a cargar el juego (la escena) desde el inicio. Para ello, se debe tener en cuenta que el jugador no podrá interactuar con este cartel si es que antes no se ha logrado encontrar la/s llaves.

```

8 public class SignPost : MonoBehaviour {
9
10     public void ResetScene () {
11         /// Called when the 'SignPost' game object is clicked
12         /// - Reloads the scene
13
14         // Prints to the console when the method is called
15         Debug.Log ("SignPost.ResetScene()' was called");
16
17         // TODO: Reset the scene by getting a reference to the scene and reloading it
18         // Declare a Scene named 'scene', then use SceneManager.GetActiveScene () to get the current scene and assign it to 'scene'
19         // Use SceneManager.LoadScene() and the Scene.name property to reload the scene
20         Scene scene = SceneManager.GetActiveScene();
21         SceneManager.LoadScene(scene.name);
22     }
23 }

```

4. Diseñar el escenario e implementar funcionalidades.

En las secciones anteriores se preparaba el ambiente para la interacción entre la escena y el jugador, pero no había una estructura o diseño como tal del laberinto; en otras palabras se realizó la “lógica”. En esta sección se explicará de

manera concreta el cómo se realizó el proceso de diseño y cuáles fueron los requisitos.

Como requisitos principales se plantea lo siguiente:

- Debe de haber un mínimo de 5 monedas en el juego.
- Debe de haber al menos una llave.
- Deben de haber "Waypoints" de manera que se pueda acceder a varios lugares de la escena.
- Se debe de diseñar el laberinto con los objetos que se encuentran en la escena.

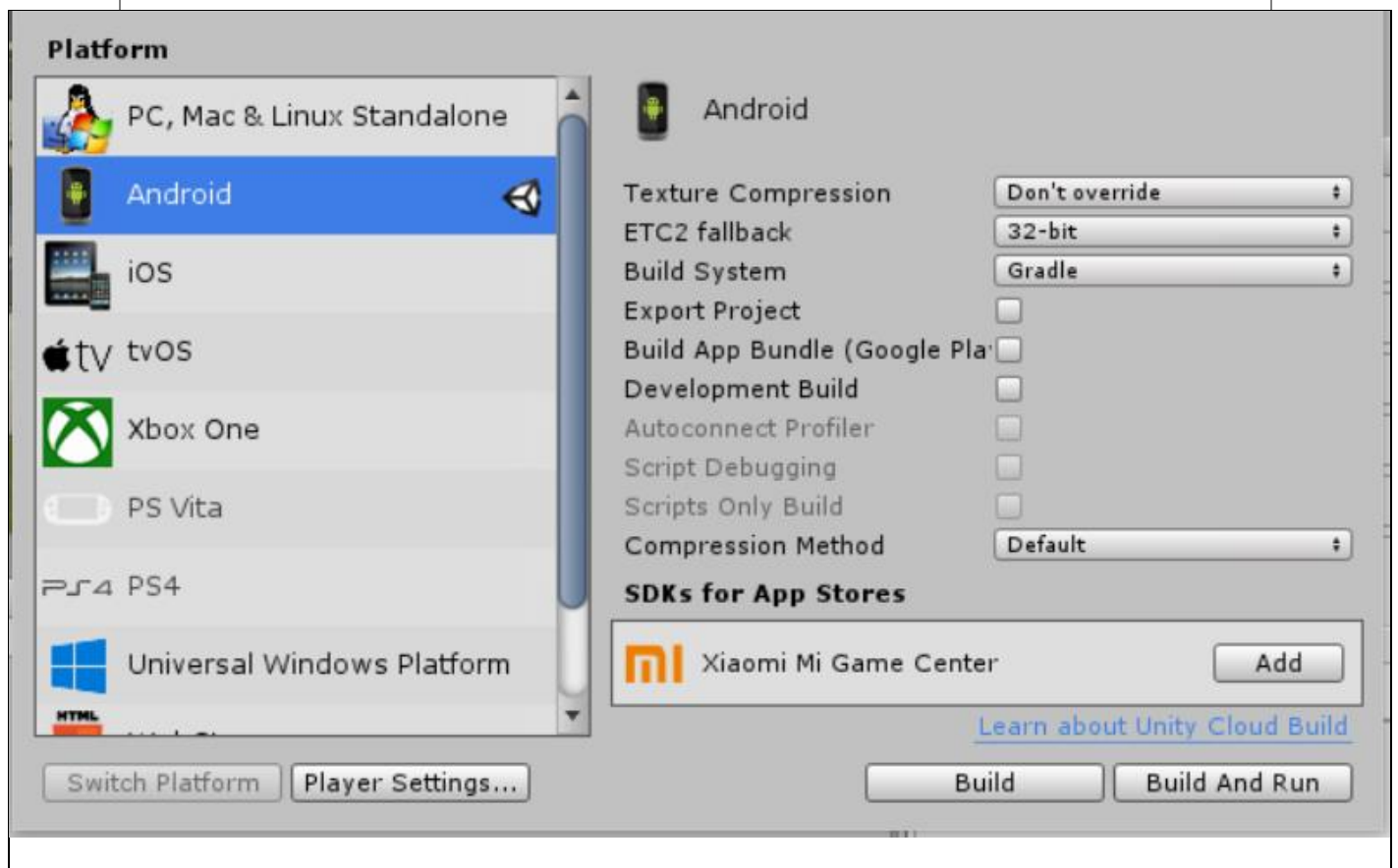
Para cumplir con los requerimientos anteriores se diseñó un laberinto medianamente grande, donde se encuentran distribuidos por todo el laberinto "Waypoints" y 5 monedas. Cerca del templo se encuentran dos monedas más, además de la llave que desbloquea la puerta.

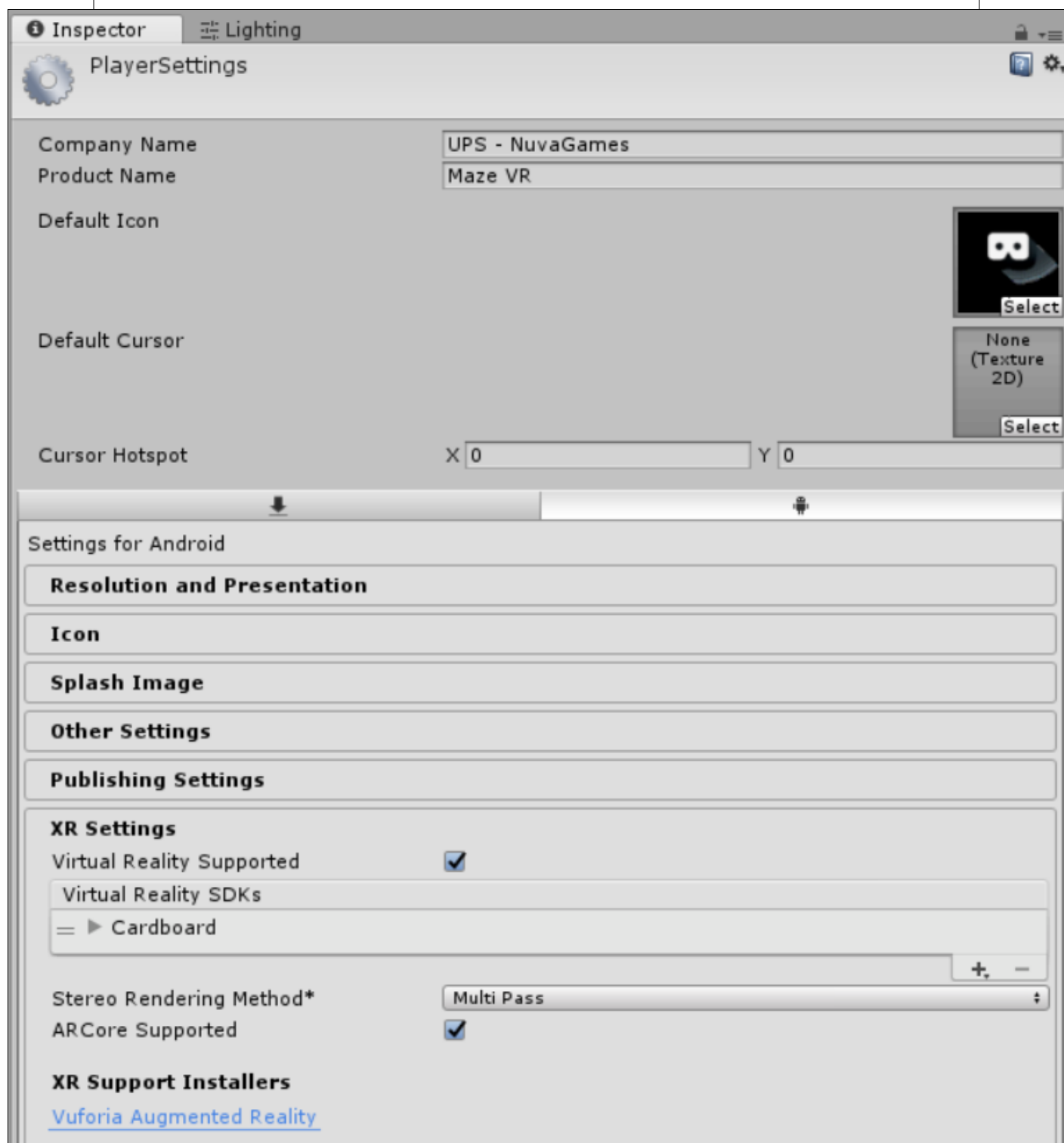
Además, se agregaron ciertos objetos decorativos a través de toda la escena a manera de que se pueda mejorar la presentación del juego. En cuanto a la iluminación se modificó un poco la Luz Direccional para que tenga un ligero aspecto a "atardecer", incluso se agregó un punto de luz dentro del templo para una mejor visualización del interior.



5. Desplegar el proyecto para dispositivos Android.

Otra parte importante de la práctica es desplegar el juego para que pueda ser instalado y ejecutado en dispositivos móviles. Para ello se configuró ciertas opciones de "Player". Además de habilitar el soporte para VR.



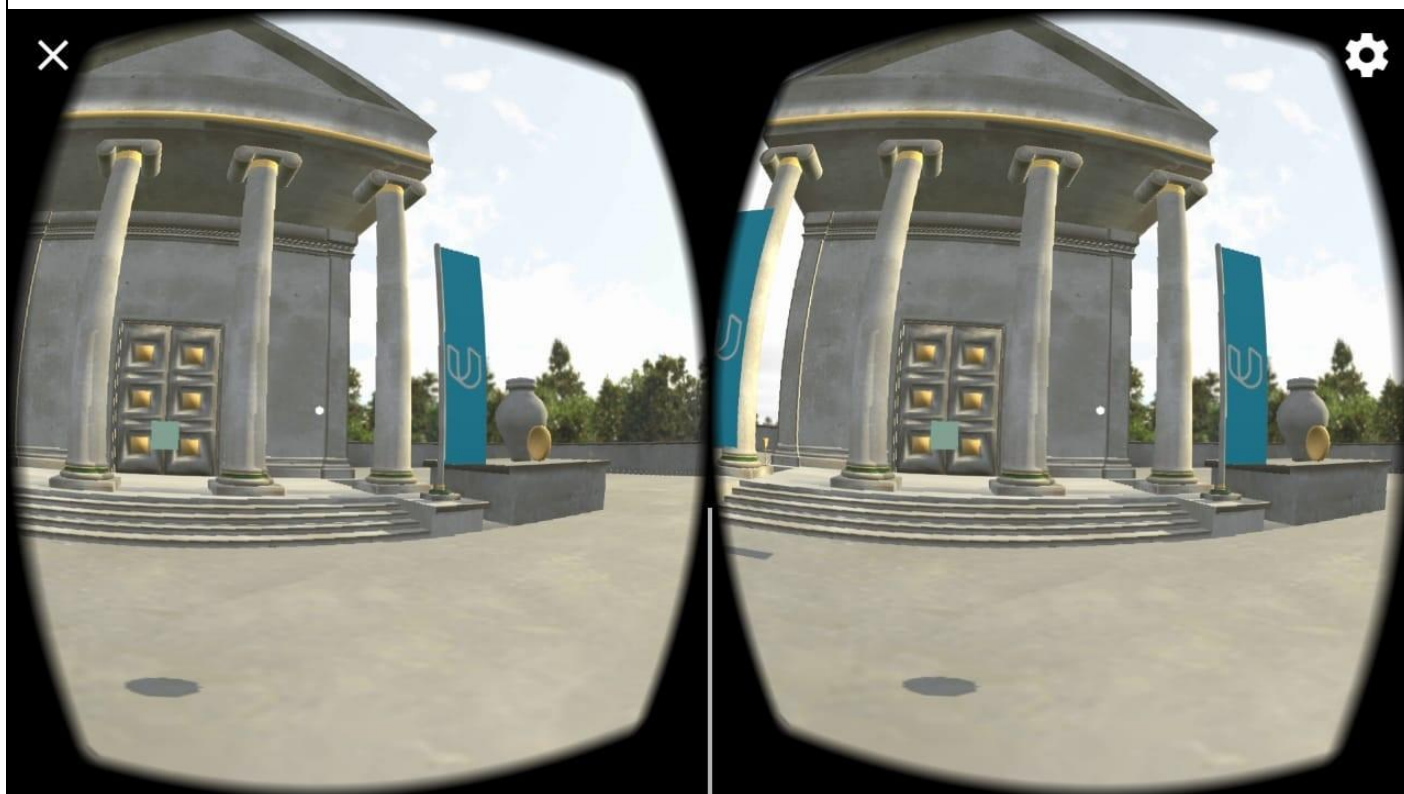



6. Presentar la aplicación en ejecución.

A. Ejecución en Unity.



B. Ejecución en un móvil Android.



	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

RESULTADO(S) OBTENIDO(S):

- Desarrolla e integra módulos interactivos virtuales.
- Se desarrollo la practica de manera exitosa

CONCLUSIONES:

- Se implemento el escenario de realidad virtual usando la herramienta Unity.
- Se ha usado los métodos enseñados en la Clase
- LINK de la Practica: <https://github.com/ingrafa/Practica08-Laberinto-VR>

RECOMENDACIONES:

Crear mas practicas para el desarrollo de mas aplicaciones de realidad virtual.

Estudiante: Rafael Angamarca Muyulema

Firma:

