Comparativa entre Unity3D y OpenAL

Curso 2k14/2k15

Apellidos, nombre	Preciado Sánchez, José Juan (jopresnc@epsa.upv.es)
Titulación	Grado de Ingeniería informática
Fecha	Marzo/2015

UNIVERSIDAD POLITECNICA DE VALENCIA

Índice

UNIVERSIDAD POLITECNICA DE VALENCIA

Índice de figuras

Índice de tablas

1 Resumen de las ideas clave

En este trabajo se va a reproducir la práctica 2 de OpenAL en la plataforma de desarrollo de videojuegos Unity3D, para poder ver las similitudes entre una y otra. Se realizará una explicación de la manera de trabajar con Unity3D y más específicamente en la incorporación y síntesis de sonido, dando como resultado una memoria de trabajo con una forma similar a la de la práctica ya realizada en clase, donde se realizarán los mismos procedimientos aplicados en esta nueva plataforma.

Características de los objetos de aprendizaje
1. Introducción a Unity3D.
2. Aprendizaje de desarrollo de scripts orientados al audio posicional.
3. Puesta en marcha de los scripts.
4. Comparativa con el desarrollo realizado en OpenAL.

Tabla 1: Características de los objetos de aprendizaje.

2 Introducción

Unity3D es una aplicación de desarrollo de videojuegos multiplataforma que facilita el trabajo con aspectos visuales de las aplicaciones y ofrece algún atajo a la hora de programar, aunque esto presenta sus limitaciones, por lo que es aconsejable habituarse al manejo mediante script, lo cual ofrece un mayor número de posibilidades, admite scripts en 3 lenguajes distintos *C#*, *JavaScript* y *Boo*, cuyas características están construidas sobre diferentes tecnologías, en particular, el sistema de audio está construido con la biblioteca *FMOD* [1]. En *Unity3D* es más natural cargar ficheros de audio que generar señales acústicas.

FMOD[1] es una herramienta de desarrollo de sonidos para videojuegos y aplicaciones desarrollado por la empresa *Firelight Technologies*, que reproduce y mezcla ficheros de audio de diversos formatos en diferentes sistemas operativos. Es una herramienta de libre descarga y uso para todo tipo de proyecto [2].

Todas las características en *Unity3D* giran en torno a los *GameObject*, algo similar a lo que pasa en Java con las clases, por lo que el audio no es una excepción y también gira en torno a estos. La configuración de audio en *Unity3D* es similar al de *OpenAL*.

- Una fuente (*AudioSource*) el cual estará unido a un *GameObject* y permitirá reproducir sonidos en un entorno 3D.
- El oyente (*AudioListener*), implementa la función de micrófono, graba los sonidos a su alrededor y los reproduce por los altavoces del usuario. Solo se puede tener un oyente en la escena.
- Los buffers están estructurados en *arrays* de *floats* en los cuales se comprimen los datos de audio.

1 http://www.fmod.org/

Al igual que en *OpenAL*, *Unity3D* realiza todos los cálculos necesarios como la atenuación debida a la distancia, el efecto *Doppler*, etc. Creando un escenario aural por el cual el usuario puede desplazarse en un espacio tridimensional.

Al igual que en *OpenAL*, existen unas relaciones por lo que se pueden hablar de dos objetos más:

1. Un contexto, que es el conjunto de oyente y fuentes de sonido, los cuales modelan las propiedades relativas a la propagación del audio y definen una determinada escena a representar de forma sonora.
2. Un dispositivo que es la representación abstracta que representa el conjunto de hardware y manejador del mismo que finalmente se encargará de generar el sonido.

La figura 1 muestra los objetos básicos tanto de *OpenAL* semejantes en *Unity3D* y sus relaciones con los objetos de contexto y dispositivo.

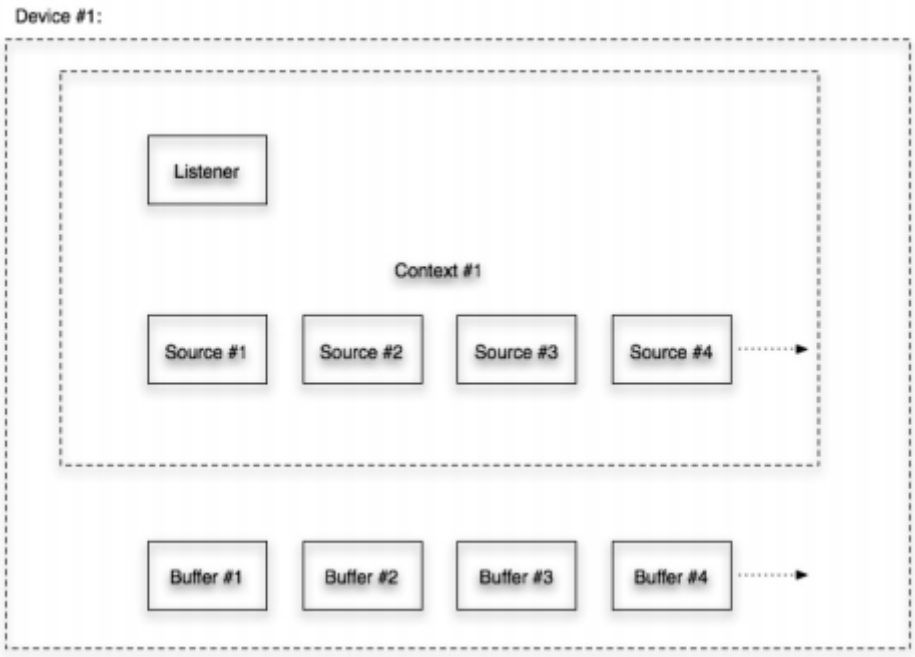

Figura 1: Un ejemplo de jerarquía entre objetos de OpenAL [3]

De forma similar a OpenAL, cuando queremos que reproducir un sonido en Unity3D, por lo menos habrá que tener un dispositivo, dentro de ese dispositivo, al menos debe haber un contexto. De forma implícita al crear una escena (contexto) en Unity3D viene un oyente (AudioListener) como componente del objeto cámara, hay que agregar las diferentes fuentes que vayan a ser necesarias, se puede realizar de diferentes maneras; una forma sencilla es agregar un objeto vacio con la componente de AudioSource, una vez realizado se pueden añadir los distintos scripts con los sonidos a reproducir a cada fuente. estos scripts podrán añadirse tantas veces como se deseen al mismo o distintos objetos. Todo el proceso se explicará más detalladamente en los ejemplos.

Dentro de una escena, los objetos pueden ser trasladados por ella, estando en objetos visibles por el usuario o no, aplicados de forma adecuada pueden lograrse efectos muy realistas añadiendo sonidos de viento, fuego, naturaleza....

Una escena simple en Unity3D que use sonidos empezará con la cámara por defecto, la cual tiene incorporada un oyente, habrá que añadir un objeto cualquiera y situarlo cerca de la cámara, crear un script generador de sonidos y añadirlo al objeto como componente, además de añadir el componente fuente. Se puede hacer variar las distintas propiedades de audio al igual que la posición de la fuente. A diferencia de OpenAL al concluir la ejecución no hay que preocuparse de liberar recursos ya que esto es realizado de forma automática.

En la documentación de Unity3D se encuentran todos los atributos relacionados al audio, tanto un manual como las especificaciones de código:

- Manual Unity3D
  http://docs.unity3d.com/Manual/index.html

- Referencia Script Unity3D
  http://docs.unity3d.com/ScriptReference/index.html

3 Planificación

Para realizar el trabajo se seguirá el siguiente proceso:

1. Lectura de los ejercicios presentados en la práctica de audio.
2. Reproducción de los ejercicios usando Unity3D.
3. Redacción del proceso seguido en Unity3D para la realización de los ejercicios.
4. Comparación entre los pasos seguidos en OpenAL y Unity3D.
5. Desarrollo de diferentes escenarios para ver los diferentes efectos.
6. Realización de una memoria orientada a la reproducción de la práctica de audio en Unity3D.

4 Ejemplos básicos

Los siguientes ejemplos son montajes sencillos para hacerse una idea de cómo incorporar cierto audio a una aplicación. Para empezar se explicará el proceso a seguir para crear un nuevo proyecto, y se realizarán diversos ejemplos de inclusión de audio en la escena.

4.1 Creación de un nuevo proyecto

Para empezar, una vez iniciado Unity3D se dará la opción de abrir un proyecto existente o crear uno nuevo, se va a seleccionar la opción de crear un nuevo proyecto, para el cual se seleccionará la carpeta donde queremos que se guarde, no se importará ningún paquete de los presentes en la lista *Assets* y se dejará la configuración por defecto en 3D.

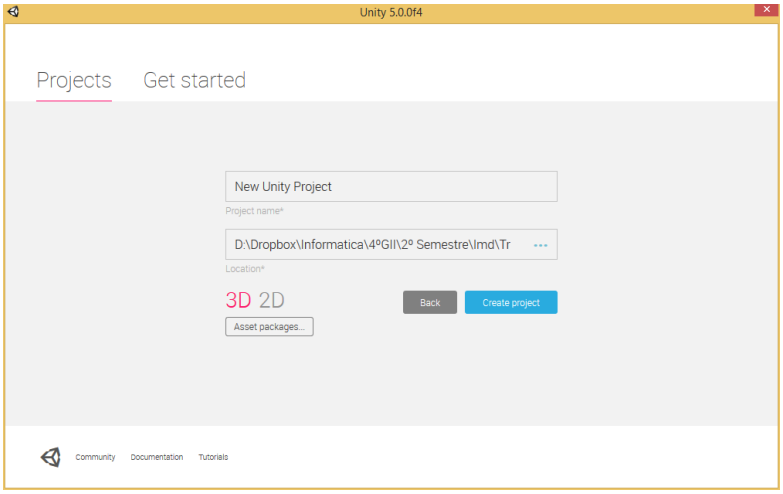

Figura 2: Creando un proyecto nuevo en Unity

Una vez pulsado el botón de crear nos aparecerá una ventana similar a esta

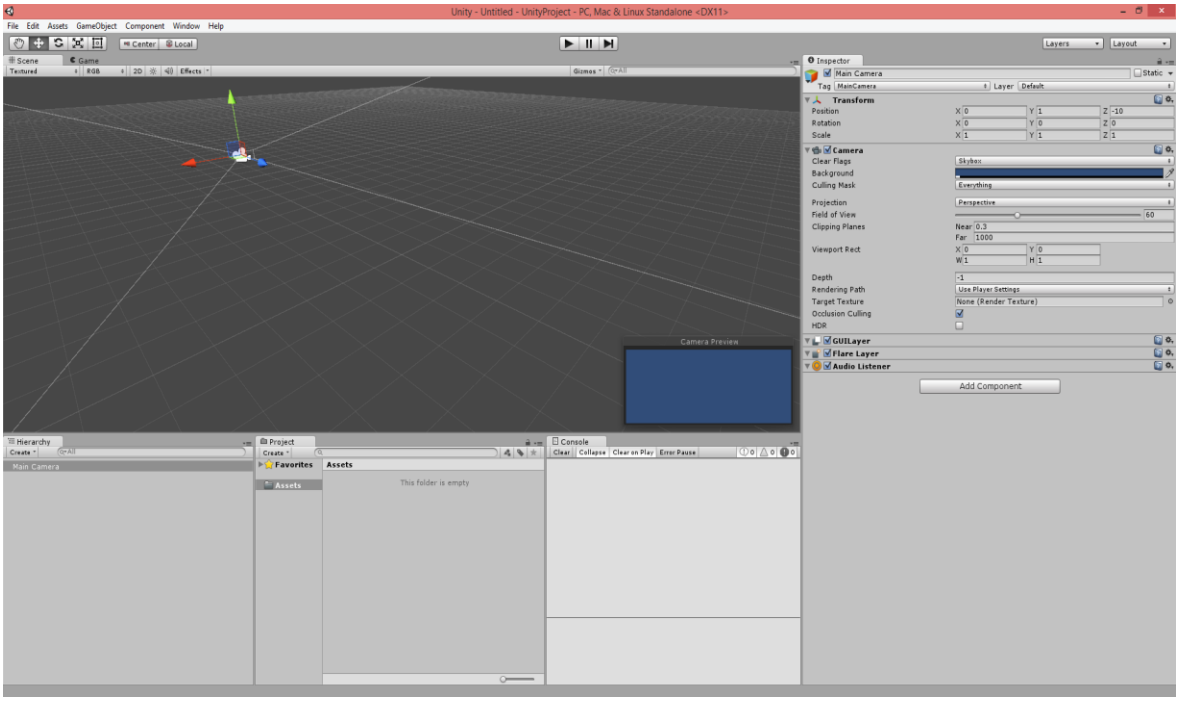

Figura 3: Ventana inicial del proyecto

En ella habrán diferentes secciones, las cuales se explican a continuación

- En la ventana' *Hierachy'* aparecerán todos los objetos presentes en la escena actual, en este caso sólo está la cámara principal.

- En *'Project'* es donde se guardarán los recursos de nuestra aplicación, en ella se pueden crear nuevas carpetas, importar imágenes, audio, generar los scripts necesarios para la aplicación,...

- En *'Console'* se mostrarán los mensajes típicos de un terminal, las advertencias, errores, los textos de *Debug*, etc,

- En 'Inspector' se mostrarán las propiedades de un objeto al ser seleccionados, en este caso está seleccionada la cámara principal y se puede ver las siguientes propiedades.
    - Transformación, las propias de posición, rotación y escalado en sus tres ejes
    - Camera, propiedades propias del visionado de la cámara
    - *GUILayer*, para *renderizar* las texturas y textos en 2D presentes en la escena.
    - *Flare Layer*, para simular el efecto de refracción de las luces en las cámaras.
    - Audio *Listener*, el oyente por defecto al crear un escena.

- La ventana *'Scene'* es donde se trabajará con los objetos en 3D de forma gráfica.

- La ventana *'Game'* mostrará el resultado conseguido hasta el momento según las modificaciones realizadas.

4.2 Trabajando con Unity3D

A la hora de trabajar con *Unity* hay varias aspectos a conocer

- Cuando se está ejecutando la escena hasta ahora realizada pulsando el play se cambiara a la vista de *'Game'*, durante la ejecución puede realizarse cualquier cambio sobre la escena, pero estos NO serán permanentes, una vez se detenga la ejecución volverán a sus valores anteriores a esta.

- Para ver más claramente los objetos es aconsejable agregar una fuente de luz, para ello en la barra de menú superior (*File*, *Edit*, ...) seleccionamos *GameObject -> Light -> Directional Light*. La posición de esta no influye es como la luz emitida por el sol, se puede variar los valores de rotación para ver los efectos producidos sobre el cubo. En *Unity* 5.0 viene una por defecto

- Para centrar la vista sobre un objeto simplemente se realiza doble click sobre él en la ventana *'Hierachy'*.

- Se pueden crear Scripts de 3 tipos *C#, JavaScript* y *Boo*, en los ejemplos se usará *C#.*

- En la ventana 'Inspector' aparece un *CheckBox* al lado de cada propiedad, este indica si está o no habilitado.

4.3 Inclusión de un fichero de audio

4.3.1. Usando las facilidades de Unity3D

Como en Unity3D no se dispone de un *"Hello, world!"* por defecto, para este primer ejemplo se cargara un fichero de audio, lo cual se hace de forma más natural que generar señales simples.

Partiendo del proyecto generado anteriormente, y usando el oyente por defecto, el cual está situado en la cámara, vamos a crear un cubo para poder visualizar de forma sencilla donde estará la fuente, le añadiremos una fuente (*AudioSource*) y un fichero de audio.

Lo primero será importar el fichero de audio a nuestro proyecto

1. Seleccionar el fichero de audio a importar.
2. Hacer *click* derecho sobre la ventana de *Assets*, donde pone *"This folder is empty"*.
3. Crear una nueva carpeta con el nombre Audio (*Create -> Folder*).
4. Realizar doble *click* sobre la nueva carpeta.
5. *Click* derecho "*Import* new *Asset*..."
6. Localizar y seleccionar el fichero de audio a utilizar

una vez realizado, ya tendremos disponible el audio para nuestro proyecto.

El siguiente paso será introducir el objeto cubo al cual agregaremos la fuente y el audio, para seleccionamos *GameObject -> 3D object-> Cube*.
Para situarlo delante de la cámara podemos hacerlo de forma gráfica o escribiendo las coordenadas en la ventana *'Inspector'*, en la propiedad *'Position'* de *'Transform'*, (x=0, y=0, x=-5).

Es aconsejable introducir una luz ambiental de la forma en que se explica en el apartado anterior (3.2).
Una vez realizado esto toca añadir la fuente de audio, para ello seleccionamos el cubo y en su panel de propiedades (*Inspector*), pulsamos *AddComponent -> Audio -> Audio Source*.

Por último arrastramos el audio importado al inspector de propiedades, en la parte de "*Audio Source*" al recuadro de *Audio Clip*, otra forma es pulsando el circulo que hay al lado del recuadro y seleccionando el audio.

Ya tenemos nuestro cubo con el audio, solo queda ver los resultados pulsando el *play*.

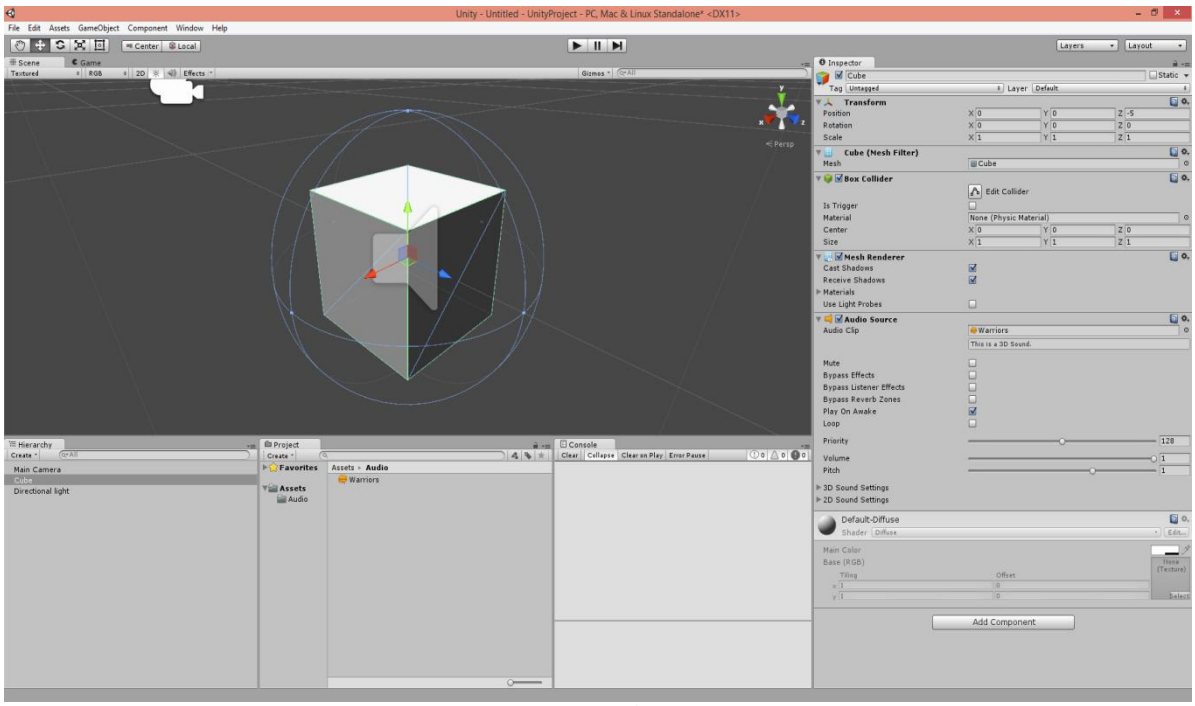

Figura 4: Resultado final escena

El problema de realizar todo el procedimiento visualmente es que no tenemos ningún control del audio durante su ejecución, si compilamos el proyecto en su estado actual tendríamos una aplicación que reproduce continuamente un sonido.

Esta forma de agregar audio es recomendable cuando en una escena tenemos sonidos que se repiten en el tiempo, por ejemplo para el sonido de fuego, podemos agregarlo de esta manera y seleccionar la opción de *'loop'*, por lo que de una forma rápida y simple tendremos configurado el sonido de este.

4.3.2. Usando Scripts

Vamos a desarrollar una escena idéntica a la anterior mediante un script, para ello primero seleccionamos *File -> Save scene*, se guardará en la carpeta *Assets* si no le indicamos otra, damos un nombre a la primera escena y creamos una nueva *File -> New scene.*

Ahora, como ya tenemos disponible el audio en los recursos, podemos empezar por crear el script que usaremos, para ello podemos crear una nueva carpeta 'Scripts' y dentro de ella hacemos *click* derecho -> *Create -> C# Script.*

Para que se ejecute el script al pulsar en 'play' necesitaremos que se encuentre en algún objeto de la escena, para ello crearemos un objeto vacio y se lo agregaremos.

1. *GameObject -> Create Empty*
2. Seleccionamos el nuevo "*GameObject*" y renombramos adecuadamente "*ScriptObject*" (puede llamarse de cualquier forma).

3. Arrastramos nuestro script a su inspector de propiedades o' *Add Component'* -> *Scripts* -> "NuestroScript"

Una vez realizado esto, al ejecutar la aplicación el script realizará las instrucciones implementadas. Agregaremos un objeto del tipo 'Cube' el cual modificaremos con el script.

Hacemos doble *click* sobre el script creado y se abrirá una ventana para modificar su código, por defecto aparecerá lo siguiente:

```
using UnityEngine;
using System.Collections;

public class Script332 : MonoBehaviour {

	// Use this for initialization
	void Start () {

	}

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

	}
}
```

Importa las librerías básicas usadas y crea una clase con 2 métodos, *'Start*()' y *'Update*()', estás funciones responden a distintos acontecimientos durante la ejecución. Las instrucciones de *'Start*()' se ejecutarán una única vez antes de la primera actualización de *frame* si el script está habilitado y las de *'Update*()' se ejecutará una vez por *frame* [4].

Para poder cargar recursos habrá que reorganizar las carpetas, creando una carpeta llamada *'Resources'* y arrastrando la carpeta de *'Audio'* a su interior.

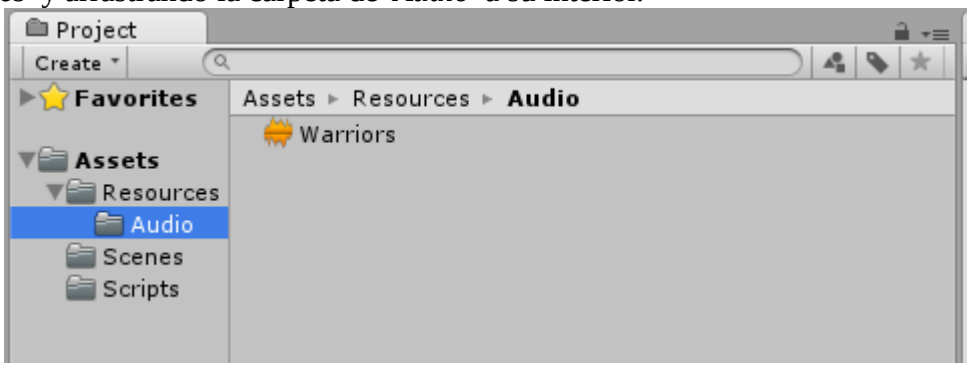

Figura 5: Estructura del proyecto

Lo primero será inicializar las variables que necesitaremos

- *GameObject*, una para la luz y otra para el cubo.
- *Ligth* para la propiedad de la luz.
- *AudioSource* para la fuente.

```
GameObject sol;
Light luzSolar;

GameObject cubo;
AudioSource sonido;
```

En la fución *'Start*()' se inicializarán todos los objetos y modificarán sus propiedades a las por defecto.

```
// Use this for initialization
void Start () {
```

Configuración de la luz ambiental tal y como se encontraba en la anterior escena

```
//Creacion de objeto vacio con el nombre "Sol"
sol = new GameObject ("Sol");
```

La función *GetComponent*<T> () es muy usada en Unity3D ya que permite seleccionar las propiedades de los objetos para poder modificarlas.

```
//Translacion de la poscion de 0, 0, 0 a 0, 0, -5
sol.GetComponent<Transform> ().Translate (0f, 0f, -5f);
//Rotacion de 0, 0, 0, a 50, 330, 0
sol.GetComponent<Transform> ().Rotate (50f, 330f, 0.0f);
```

Con la función *AddComponent*<T> () se pueden agregar nuevas propiedades a los objetos.

```
//Insercion de propiedades luminosas al objeto
//Guardar un puntero a la propiedad luminosa en el objeto
luzSolar = sol.AddComponent<Light> ();
//Cambiar el tipo de luz, por defecto Puntual a Direccional
luzSolar.type = LightType.Directional;
//Cambio de la intensidad de la luz de 1 a 0.5
luzSolar.intensity = 0.5f;
```

La función *Find(string name)* de *GameObject* nos permite localizar a un objeto en la escena por su nombre.

```
//Localizacion del objeto ya creado con nombre "Cube" a la variable cubo
cubo = GameObject.Find ("Cube");
//Translacion del objeto a la posicion 0, 0, -5
cubo.GetComponent<Transform> ().Translate (0f, 0f, -5f);
```

```
        //Inserccion de una fuente de audio al objeto cubo
        //Puntero a la fuente de audio de cubo
        sonido = cubo.AddComponent<AudioSource> ();
```

La función *Load*<T>(*string* Path) nos permite cargar recursos introduciendo el *Path* a partir de la carpeta *'Resources'*

```
        //Asignacion del audio seleccionado anteriormente a la fuente
        sonido.clip = Resources.Load<AudioClip> ("Audio/Warriors");
        //Ejecucion del audio
        sonido.Play ();
}
// Update is called once per frame
void Update () {
        //Deteccion de si es pulsada la barra espaciadora
        if (Input.GetKeyDown (KeyCode.Space))
        {
                //Si el audio se esta reproduciendo
                if(sonido.isPlaying)
                        //Pausar la reproduccion
                        sonido.Pause();
                else
                        //Reproducir audio
                        sonido.Play();
        }
}
```

Al trabajar con script disponemos de una mayor libertad de maniobrabilidad, a diferencia de la anterior escena, en la que no teníamos ningún control sobre ella, en esta tenemos la posibilidad de pausar el audio y volver a reanudarlo, además de otras muchas posibilidades que se irán viendo más adelante.

4.4 Generando señales simples

Ahora que se conoce la dinámica de trabajo con Unity3D realizaremos un script generador de señales simples, estas no pueden ser generadas de otra forma que no sea con el uso de Scripts.

Para ello crearemos una nueva escena idéntica a la anterior del apartado 3.3.2, para ello, una vez guardada la escena actual seleccionamos *Edit -> Save Scene ass...* y le otorgamos un nuevo nombre creando así una copia de la anterior.

En *ScriptObject* deshabilitaremos o borraremos la componente script del apartado anterior y crearemos un nuevo script, el cual añadiremos como componente en *ScriptObject* y comenzaremos a modificar.

Se pueden realizar las modificaciones deseadas ya comentadas anteriormente como posiciones, luminosidad o probar a modificar otras propiedades dentro del script o gráficamente.

Una vez realizado la configuración inicial, podemos comenzar a modificar nuestro script generador de señales simples. Para ello necesitaremos definir las variables que usaremos más adelante.

```
public int position = 0;
public int samplerate = 44100;
public float frequency = 440;
private AudioClip simple;
private AudioSource fuente;
```

Haremos uso de la función AudioClip.Create [5] que crea un clip definiendo los diferentes parámetros, se usará en la funcion Start()

```
simple = AudioClip.Create ("Simple", samplerate * 2, 1, samplerate, true,  OnAudioRead);
fuente = cubo.AddComponent<AudioSource> ();
fuente.clip = simple;
fuente.Play ();
```

Como se puede observar se hace uso de dos funciones por callBack las cuales vienen explicadas en la documentación de Unity y son las siguientes

```
void OnAudioRead(float[] data)
{
	int count = 0;
	while (count < data.Length)
	{
		data[count] = Mathf.Sign(Mathf.Sin(2 * Mathf.PI * frequency * position / samplerate));
		position++;
		count++;
	}
}
```

Una vez realizado esto podemos ejecutar nuestra escena y escuchar una señal de 440Hz. En modo ejecución, podemos cambiar la fuente de audio a modo bucle (dentro de las propiedades del cubo habilitar 'Loop') y pulsar la barra espaciadora, en caso de que tengamos la detección de esta como en el código anterior, y escuchar la señal.

Además podemos modificar su frecuencia seleccionando el ScriptObject. Veremos cómo aparecen unas variables las cuales podemos mantener un click izquierdo sobre el nombre de 'Samplerate' y 'Frequiency' y ver los efectos que se producen en el sonido al mover el cursos manteniendo el click.

A diferencia de OpenAL en Unity3D no se dispone de las opciones de OpenAL para modificar la señal de forma simple, por lo que hay que realizar los cálculos para su realización.

Un ejercicio de OpenAL dice lo siguiente "Realizar una variación de este código que haga sonar una escala musical. Para ello habrá que generar una onda básica del mismo tipo pero con diferentes valores de frecuencias. Por ejemplo, se puede probar con los valores (en Hz) siguientes: 262, 294, 330, 349, 392, 415, 440,494" [3].
Para este ejercicio se puede modificar el Update del código anterior para que dependiendo de la tecla pulsada establezca una frecuencia distinta y reinicie el sonido, consiguiendo así una especie de teclado musical.

```
void Update () {
		//Deteccion de si es pulsada la barra espaciadora
		if(Input.GetKeyDown(KeyCode.Alpha1))
		{
			frequency = 262;
			fuente.Play();
		}
		if(Input.GetKeyDown(KeyCode.Alpha2))
		{
			frequency = 294;
			fuente.Play();
		}
		if(Input.GetKeyDown(KeyCode.Alpha3))
		{
			frequency = 330;
			fuente.Play();
		}
		if(Input.GetKeyDown(KeyCode.Alpha4))
		{
			frequency = 349;
			fuente.Play();
		}
		if(Input.GetKeyDown(KeyCode.Alpha5))
		{
			frequency = 392;
			fuente.Play();
		}
		if(Input.GetKeyDown(KeyCode.Alpha6))
		{
			frequency = 415;
			fuente.Play();
		}
		if(Input.GetKeyDown(KeyCode.Alpha7))
		{
			frequency = 440;
```

```
                fuente.Play();
            }
            if(Input.GetKeyDown(KeyCode.Alpha8))
            {
                frequency = 494;
                fuente.Play();
            }
        }
```

5 Audio Posicional

Para mostrar el efecto de audio posicional se ha recreado el ejemplo del proyecto PIGE visto en clase, podéis desplazaros con las teclas 'W' avanzar, 'A' desplazamiento lateral izquierdo, 'S' desplazamiento lateral derecho y 'D' retroceder o con las flechas de dirección de la misma manera. Para activar y desactivar los sonidos pulsar los números 1, 2 y 3.

6 Actividades

Se han reproducido los ejercicios marcados en el último apartado del boletín adaptados a las características de *Unity*.

- Incorporar la funcionalidad del apartado “Generando señales simples” en el ejemplo de PIGE, para lo que se crearán tantos nuevos cubos como señales básicas se podían crear en aquel caso y que las teclas activen y desactiven cada una de esos sonidos.
- Hacer variar la propiedad de *PITCH* de cada fuente, con la tecla 'i'.
- Hacer variar la propiedad de *GAIN* de cada fuente, con la tecla 'g', así como la de oyente (seleccionándolo con la tecla 'o').
- Cambiar la orientación en un eje, con la tecla 'r' o 'R' (minúscula la decrementa en una cantidad y en mayúscula para incrementarla), del oyente, previamente se habrá seleccionado con la tecla 'o'.
- Mover (cambiar la posición espacial) de cada fuente con las flechas, previamente se habrá seleccionado con la tecla 'f' y una de las teclas de cifras. Así como su orientación, con 'r' o 'R'

Se ha implementado de la siguiente manera:

Primero se debe seleccionar la fuente con las teclas de **'F1' a 'F11'**.
Una vez seleccionada la fuente, para reproducirla se pulsará la **barra espaciadora**.
Para **aumentar el PITCH** se pulsará la tecla **I**.
Para **disminuir el PITCH** se pulsará la tecla **K**.
Para **aumentar el GAIN** se pulsará la tecla **U**.
Para **disminuir el GAIN** se pulsará la tecla **J**.

Para **rotar a la derecha** se pulsará la tecla **O**.
Para **rotar a la izquierda** se pulsará la tecla **L**.
Para **mover una fuente** pulsar **M** y una vez finalizado volver a pulsar **M**.

En *Unity3D* no hay una opción por defecto para modificar el *GAIN* por lo que se ha tenido que hacer la siguiente función

```
public class GainModifier : MonoBehaviour {

	public float gain = 1;

	void OnAudioFilterRead(float[] data,int channels) {
		for (int i = 0; i < data.Length; ++i)
			data[i] = data[i] * gain;
	}
}
```

la cual modifica el GAIN de las fuentes de audio a las cual se le asigne, además no es posible modificar el GAIN del oyente.

7 Uso del micrófono

Se ha generado una escena para mostrar cómo hacer uso del micrófono. Para comenzar a grabar hay que hacer uso de la instrucción

```
Microphone.Start ("Built-in Microphone", true, 360, 44100);
```

La cual grabará clips de hasta un minuto de duración sin detenerse. Para escuchar estas grabaciones se ha creado un cubo con una fuente de audio y como clip se ha asignado el generado por la grabación.

Con la tecla 'R' se comenzará a grabar y al volver a pulsar se forzará a terminar la grabación, lo que a la vez generará un fichero .WAV del clip grabado. Para generar este fichero .WAV se ha recurrido a una solución ya generada savWav.cs[6] a la cual se ha modificado la linea 41 de "`var filepath = Path.Combine(Application.persistentDataPath, filename);`)" a "`var filepath = Path.Combine(Application.dataPath, filename);`", para que guarde el fichero en el mismo directorio que la aplicación. Con la tecla espacio se podrá escuchar la última grabación realizada.

8 Opciones de audio en Unity3D

Para acceder a las opciones de audio se debe acceder a *Edit -> Projects Settings ->* Audio.
En el 'Inspector' se abrirá una ventana donde se podrán cambiar algunas de las características de este como el volumen, el modo de salida de audio, el factor *Doppler*...

9 Generación de un ejecutable

Para elaborar un único ejecutable con las diferentes escenas se han creado unos cargadores de escena presentes en todas ellas. Los que tienen partículas verdes son para avanzar al siguiente nivel y los que tienen partículas rojas para retroceder al anterior.

Para que esto funcione primero hay que añadir las escenas a usar en *File -> Build Settings...* y arrastrar todas las escenas a usar al recuadro titulado *'Scenes* In *Build'*, una vez aquí, se puede seleccionar para que plataforma se va a compilar y qué tipo de arquitectura.

Al seleccionar 'Build' nos creará un ejecutable con todas las escenas que hayamos añadido.

10 Comparativas OpenAL vs Unity3D

	OpenAL	*Unity3D*
Generación de señales simples	Sí	Sí
Trabajo con ficheros de audio	Sí	Sí
Audio posicional	Sí	Sí
Variación del *PITCH*	Sí	Sí
Variación del *GAIN*	Sí	Sí
Variación *GAIN* oyente	Sí	No
Orientación del oyente	Sí	Sí
Mover fuentes y orientación	Sí	Sí
Variar la propiedad *DOPPLER*	Sí	Sí
Captura de audio	Sí	Sí
Creación de ficheros *WAV*	Sí	Sí

Figura 2: Comparativa entre OpenAL y Unity3D

11 Conclusión

Tras presentar algunas características del audio en *Unity3D* y ver su forma de trabajar se ha visto como, aunque a la hora de modificar el audio *OpenAL* presenta métodos más preparados, la mayor parte de estos pueden ser cubiertos mediante el uso de *scripts* en *Unity*, el cual ofrece una manipulación más sencilla que *OpenAL* a la hora de trabajar con ficheros de audio y una dificultad similar cuando se trata de la generación de este. Como se puede comprobar viendo los diferentes códigos generadores de señales simples en el anexo.

12 Bibliografía

Bibliografía y referencias.

12.1 Boletín de práticas:

[3] Manuel Agustí, Curso 2k14/2k15,Grado de Ingeniero en Informática, Escola Tècnica Superior d'Enginyeria Informàtica, DISCA - UPV, "Práctica de procesamiento de audio: Audio posicional con OpenAL", Integración de Medios Digitales

12.2 Referencias de fuentes electrónicas:

[1] Unity (software) http://es.wikipedia.org/wiki/Unity_%28software%29

[2] FMOD Studio audio tools now completely free for indies http://www.gamasutra.com/view/news/212696/FMOD_Studio_audio_tools_now_completely_free_for_indies.php

[4] Execution Order of Event Functions http://docs.unity3d.com/Manual/ExecutionOrder.html

[5] AudioClip.Create http://docs.unity3d.com/ScriptReference/AudioClip.Create.html

[6] Calvin Rien, 2012, Unity3D: script to save an AudioClip as a .wav file. https://gist.github.com/darktable/2317063

13 ANEXO

La diferencia entre la funcionalidad de ambos códigos es que el de *Unity* está haciendo uso de una única fuente a la que le cambia la frecuencia dependiendo de la tecla y el de *OpenAL* hace uso de una fuente por señal.

13.1 Señales simples OpenAL

```
/*
 * (C) The OpenAL Utility Toolkit (ALUT) v1.1
 *
 * Compilar:
 * gcc helloWorld_ALUT.c  -lalut -lopenal -o helloWorld_ALUT
 *
 * Es interesante tener a mano la documentaci ó n sobre :
  'The OpenAL Utility Toolkit', 'OpenAL 1.1 Specification' y 'OpenAL_Programmers_Guide'


 */

#include <stdlib.h>
#include <stdio.h>
#include <AL/alut.h>

int main (int argc, char **argv)
{
 char c;
 ALuint buffers[6], fuente;
 alutInit (&argc, argv);

 buffers[0] = alutCreateBufferHelloWorld();
 buffers[1] = alutCreateBufferWaveform(ALUT_WAVEFORM_SINE, 440.0, 0.0, 1.0);
 buffers[2] = alutCreateBufferWaveform(ALUT_WAVEFORM_SQUARE, 440.0, 0.0, 1.0);
 buffers[3] = alutCreateBufferWaveform(ALUT_WAVEFORM_SAWTOOTH, 440.0, 0.0, 1.0);
 buffers[4] = alutCreateBufferWaveform(ALUT_WAVEFORM_WHITENOISE, 440.0, 0.0, 1.0);
 buffers[5] = alutCreateBufferWaveform(ALUT_WAVEFORM_IMPULSE, 440.0, 0.0, 1.0);

 alGenSources (1, &fuente);
 printf("'h'ellow, 's'ine, sq'u'are, sa'w'tooh, white'n'oise, 'i'mpulse\n");

 do {
  scanf("%c",&c);
  switch ( c )
  {
  case 'h': alSourcei(fuente, AL_BUFFER, buffers[0]); printf("PASO"); break;
```

```
  case 's': alSourcei(fuente, AL_BUFFER, buffers[1]); break;
  case 'u': alSourcei(fuente, AL_BUFFER, buffers[2]); break;
  case 'w': alSourcei(fuente, AL_BUFFER, buffers[3]); break;
  case 'n': alSourcei(fuente, AL_BUFFER, buffers[4]); break;
  case 'i': alSourcei(fuente, AL_BUFFER, buffers[5]); break;
  } // switch
  if(c=='h' || c=='s' || c=='u' || c=='w' || c=='n' || c=='i')
  alSourcePlay (fuente);
  alutSleep (1);
 }while ((c!='q') && (c!='Q'));

 alDeleteBuffers(6, buffers);
 alutExit ();
 return EXIT_SUCCESS;
}
```

13.2 Señales simples Unity3D

```
using UnityEngine;
using System.Collections;

public class Script34 : MonoBehaviour {

        public int position = 0;
        public int samplerate = 44100;
        public float frequency = 440;
        private AudioClip simple;
        private AudioSource fuente;

        private GameObject cubo;
        // Use this for initialization
        void Start () {
                //Localizacion del objeto ya creado con nombre "Cube" a la variable cubo
                cubo = GameObject.Find ("Cube");
                //Translacion del objeto a la posicion 0, 0, -5
                cubo.GetComponent<Transform> ().Translate (0f, 0f, -5f);

                //Inicializacion del clip
                simple = AudioClip.Create ("Simple", samplerate * 2, 1, samplerate, true, OnAudioRead);

                //Inserccion de una fuente de audio al objeto cubo
                //Puntero a la fuente de audio de cubo
                fuente = cubo.AddComponent<AudioSource> ();

                fuente.clip = simple;
        }
```

```
void OnAudioRead(float[] data)
{
	int count = 0;
	while (count < data.Length)
	{
		data[count] = Mathf.Sign(Mathf.Sin(2 * Mathf.PI * frequency * position / samplerate));
		position++;
		count++;
	}
}

// Update is called once per frame
void Update () {
	//Deteccion de si es pulsada la barra espaciadora
	if(Input.GetKeyDown(KeyCode.Alpha1) || Input.GetKeyDown(KeyCode.Keypad1))
	{
		frequency = 262;
		fuente.Play();
	}
	if(Input.GetKeyDown(KeyCode.Alpha2) || Input.GetKeyDown(KeyCode.Keypad2))
	{
		frequency = 294;
		fuente.Play();
	}
	if(Input.GetKeyDown(KeyCode.Alpha3) || Input.GetKeyDown(KeyCode.Keypad3))
	{
		frequency = 330;
		fuente.Play();
	}
	if(Input.GetKeyDown(KeyCode.Alpha4) || Input.GetKeyDown(KeyCode.Keypad4))
	{
		frequency = 349;
		fuente.Play();
	}
	if(Input.GetKeyDown(KeyCode.Alpha5) || Input.GetKeyDown(KeyCode.Keypad5))
	{
		frequency = 392;
		fuente.Play();
	}
	if(Input.GetKeyDown(KeyCode.Alpha6) || Input.GetKeyDown(KeyCode.Keypad6))
	{
```

```
                frequency = 415;
                fuente.Play();
            }
            if(Input.GetKeyDown(KeyCode.Alpha7) || Input.GetKeyDown(KeyCode.Keypad7))
            {
                frequency = 440;
                fuente.Play();
            }
            if(Input.GetKeyDown(KeyCode.Alpha8) || Input.GetKeyDown(KeyCode.Keypad8))
            {
                frequency = 494;
                fuente.Play();
            }

        }
    }
```