**MATERIALES, TEXTURIZADO Y MONETIZACIÓN**

**1 - Monetización y Analytics**

En esta sección revisaremos 2 servicios de Unity: ads y analytics. Se describe cada uno en la sección correspondiente.

Para habilitar cualquiera de estos servicios debemos abrir la ventana de servicios llendo a Window -> General -> Services y seleccionando en esa ventana el servicio que queremos habilitar.

**1.1 - Ads**

Ads es una abreviación de advertisements, o anuncios en español. Unity nos ofrece un servicio en que le muestra ads a nuestros jugadores. Nosotros como desarrolladores recibimos un porcentaje de las ganancias que esos anuncios generan.

Para mostrar un ad primero debemos habilitar el servicio como se describió previamente.

Otra cosa importante, para poder probar los avisos en el editor, es seleccionar una plataforma móvil en los build settings, como iOS o Android. Si esto no podremos ver el preview de anuncios cuando probemos el juego.

Una vez hecho esto podemos traer el namespace Advertisement a cualquier script escribiendo:

using UnityEngine.Advertisements;

Podemos luego escribir código para mostrar ads como se describe a continuación.

**1.1.1 - Avisos simples**

La manera más sencilla de mostrar un aviso es escribiendo:

Advertisement.Show();

Esto detendrá la aplicación para el jugador y le mostrará un video publicitario.

**1.1.2 - Avisos recompensados**

Es muy común en los juegos móviles un sistema de monetización en que se le ofrece al jugador ver un video publicitario a cambio de alguna recompensa en el juego, como monedas o vidas extra.

Cuando queramos recompensar al jugador por ver un video primero escribimos una función para darle la recompensa al jugador. Esta función debe ser void y recibir como parámetro un enum de tipo ShowResult, como se describe a continuación:

void DarRecompensa (ShowResult result)

{

if (result == ShowResult.Finished)

{

coins += 10;

}

}

El parámetro show result puede tomar uno de tres tipos de valores: Finished, Skipped y Failed. El valor que tendrá dependerá de lo que el jugador haya hecho o de si el video falló por algún motivo. Finished significa que el video fue visto hasta el final. Skipped significa que el jugador se saltó el video. Failed significa que algún problema de conexión impidió ver el video.

Creada esta función debemos vincularla con una función para mostrar el video, como en el código que sigue:

ShowOptions options = new ShowOptions();

options.resultCallback += DarRecompensa;

Advertisement.Show (options);

El objeto show options contendrá una referencia a la función que se creó anteriormente. Eso ocurre en las primeras dos líneas. La tercera línea muestra el anuncio y utiliza como parámetro las opciones recién creadas. Esto hará que el jugador sea recompensado si ve el video completo.

**1.2 - Analytics**

Analytics se refiere a obtener información acerca de los usuarios de nuestro juego.

Si habilitamos Analytics en la ventana de servicios ya estaremos leyendo cosas básicas, como cuándo los jugadores juegan nuestro juego y en qué sistemas operativos.

Para ver la información que recopilamos debemos ir a nuestro Dashboard, que se encuentra en nuestra cuenta en el sitio web de Unity. Podemos ir directamente a él a través de la ventana de servicios, seleccionando Analytics y luego Go to Dashboard.

Además de los eventos básicos de comenzar y finalizar la aplicación, podemos leer eventos creados por nosotros. Para crear un evento nuestro escribimos código como el que sigue:

AnalyticsEvent.Custom (“jugador subio de nivel”, new Dictionary <string, object> ()

{

{“nuevo nivel”, level},

{“nombre”, nombreJugador}

});

Lo primero que hacemos es ponerle un nombre al evento, en este caso se llama “jugador subio de nivel”. Podemos imaginarnos que llamamos a este código siempre que el nivel del jugador aumenta.

Luego usamos un diccionario para definir variables, que contienen información adicional acerca del evento. A cada una podemos darle cualquier valor. En el ejemplo se utiliza un entero (level) y un string (nombreJugador) como valores. Es importante saber que estos valores deben ser tipos de dato primitivos para ser visibles en el dashboard de Unity. Los valores primitivos son todos los tipos de número (int, float, double, etc), los booleanos y los strings.

Este diccionario con valores adicionales es opcional. Si queremos podemos lanzar un evento solo con un nombre y nada más.