«Talento Tech»

Desarrollo de Videojuegos

Unity 2D

Clase 15

«Talento Tech»

Buenos Aires Ciudad

Clase N° 15 | Sonidos + Build

Temario:

- Audio handling (Audio Sources, mixers)
- BuildSettings
- Icon

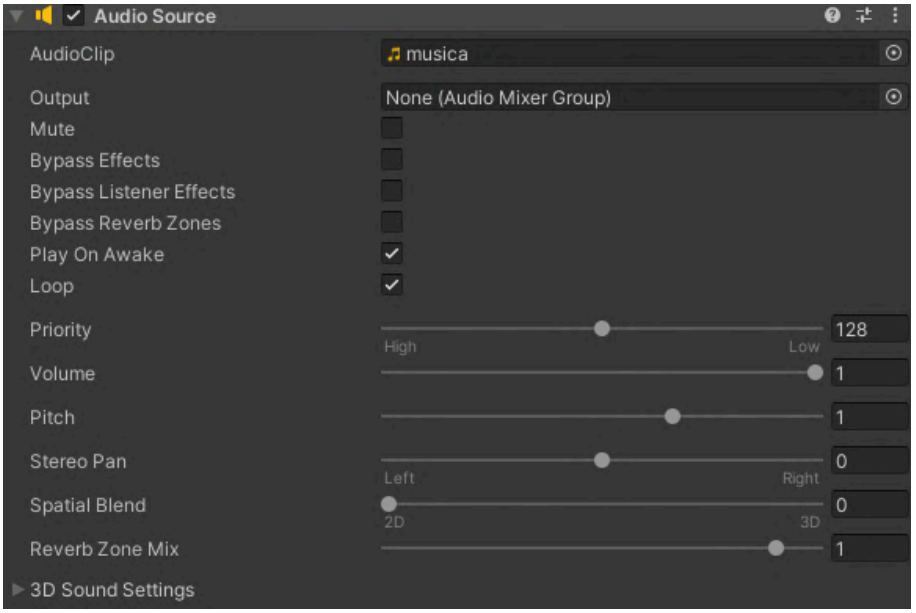
Sonido.

La música y el sonido en videojuegos.

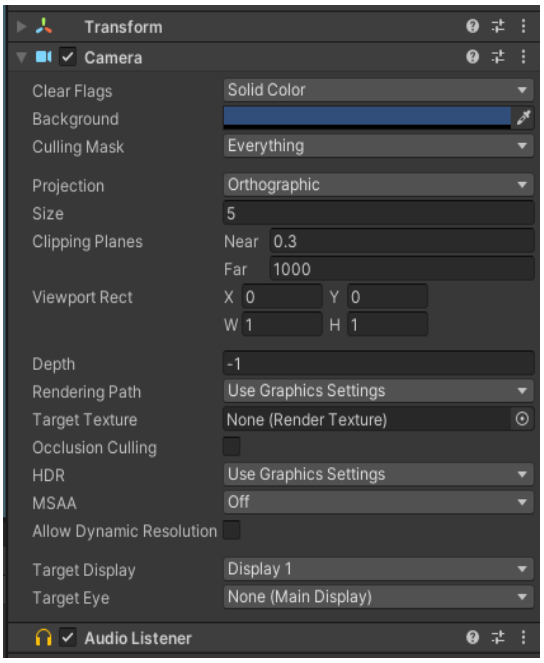
Los sonidos en los juegos a veces parecen estar en un segundo plano cuando se trata de diseño, muchas personas lo dan por sentado y no le prestan atención al diseño de sonido ni a la música, lo que no necesariamente significa algo malo. Un buen diseño de sonido pasa desapercibido porque se siente acorde al mundo que nos propone el juego y sin darnos cuenta nos aporta inmersión y feedback en nuestras acciones.

AudioSource y AudioListeners.

Ya sea para implementar música o sonidos, en Unity vamos a utilizar un componente que se llama *AudioSource,* es decir, fuente de sonido; el cual posee muchas propiedades para facilitar la implementación de efectos de sonido.



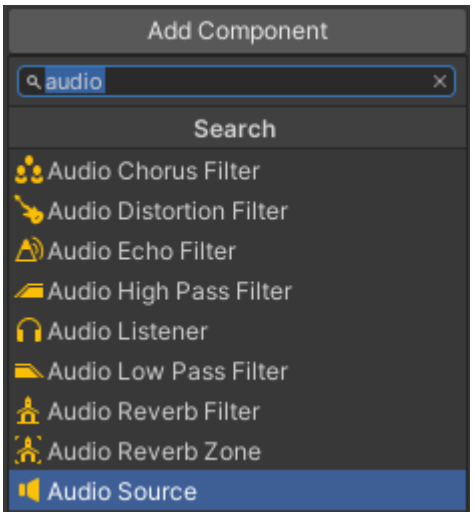
Además de un AudioSource, vamos a necesitar un Listener, este es otro componente que va a escuchar el sonido, y le va a dar una espacialidad a lo que escuchamos. Este componente no tiene ninguna propiedad, simplemente va a ser el que interprete el sonido del juego para que lo podamos escuchar cuando juguemos. En la mayoría de los casos, la cámara es el que poseé el AudioListener, en la siguiente imagen se puede ver que la cámara del proyecto ya tiene uno de forma predeterminada:



Implementando música.

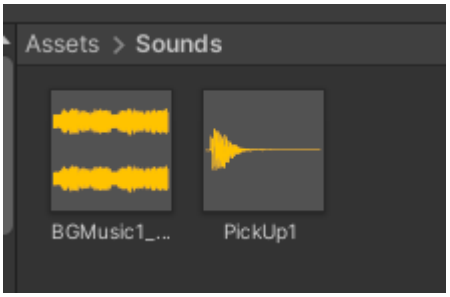
Como siempre aclaramos, existen diversas formas de implementar recursos y funciones en Unity. Si investigan por su cuenta van a aprender nuevas formas de realizarlas. En este curso vamos a aprender a hacerlo de una forma específica, pero no es la forma definitiva.

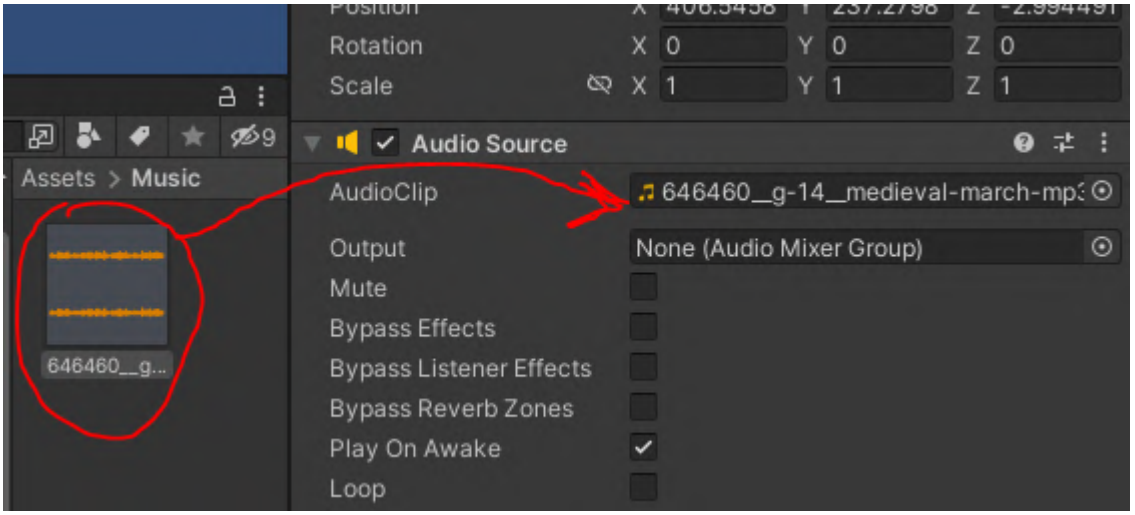
Lo primero que vamos a hacer es un objeto vacío y lo vamos a llamar “Música” (no utilizamos tildes para evitar problemas, ya que el lenguaje de Unity por defecto es el inglés).
Dentro de este objeto vamos a agregar el **AudioSource**:



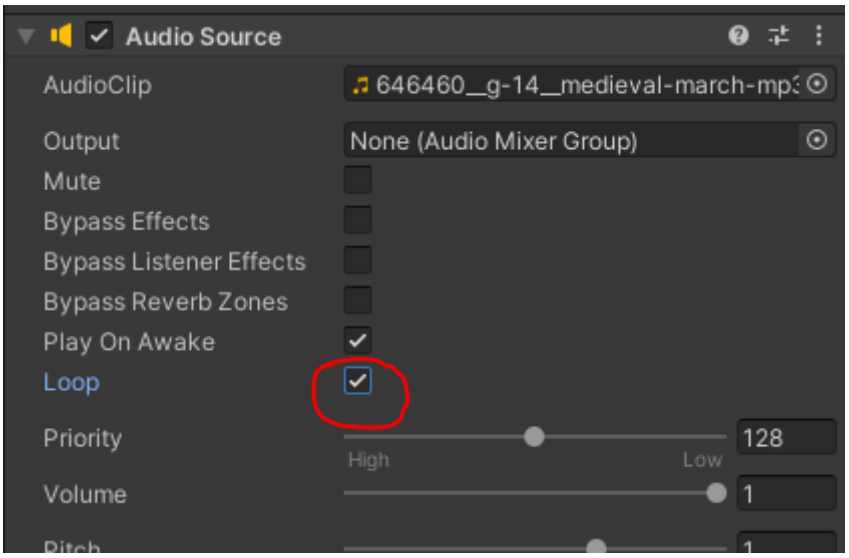
 ¡Excelente! Este objeto se va a encargar de loopear el sonido, es decir que cuando termine la pieza, la va a volver a repetir infinitamente. Vamos a descargar algo de fondo desde la pagina freesound.org

Vamos a subir el archivo de sonido arrastrándolo a Unity y luego lo pondremos en la variable “AudioClip” de nuestro Audio Source.





Es muy importante que la variable *Loop* sea verdadera, para que la música se repita infinitamente.

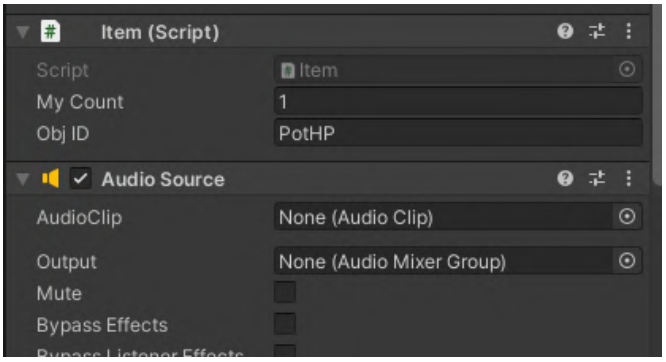


¡Nuestro juego ya tiene música!

Los sonidos que elijamos son muy importantes, ya que buscamos generar distintas emociones según el juego. Acá, por ejemplo, creamos un ambiente competitivo utilizando un clásico sonido de Arcades.

Implementando efectos de sonido desde el código.

Ahora que nuestro juego tiene música, tenemos que agregarle unos efectos de sonido para darle una sensación más responsiva al movimiento. Vamos a agregar un efecto de sonido cuando agarramos algún objeto. Para esto, le añadiremos el componente “Audio Source” a nuestros objetos “**PickUp**”y modificaremos nuestro Script “**Item**” para que pueda reproducir audios desde él.



Primero nos aseguramos de que nuestro Script tenga definidas 2 variables muy importantes, el AudioSource y una del tipo “”AudioClip” que contendrá nuestro sonido a reproducir.

```
private AudioSource aSource;
[SerializeField] private AudioClip clip;
```

Obtendremos nuestro audiosource usando el GetComponent en nuestra función Start().

```
private void Start()
{
    aSource = GetComponent<AudioSource>();
}
```

Y ahora crearemos nuestra situación para reproducir el sonido. ¿En qué momento queremos que suene? En este caso, siguiendo la idea de la clase 8, haremos que se reproduzca cuando agarremos nuestro “pickUp” (la poción).

```
 private void OnTriggerEnter2D(Collider2D collision){
        Pick2(collision);
    }
private void Pick2(Collider2D col){
  if (col.CompareTag("Player")){
      ItemManager inv = col.GetComponent<ItemManager>();

   foreach (KeyValuePair<GameObject, int> items in inv.inventory){
         Item invID = items.Key.GetComponent<Item>();
       if (invID.objID == objID){
          aSource.PlayOneShot(clip);
          Debug.Log("Obtuve un: " + gameObject.tag);
```

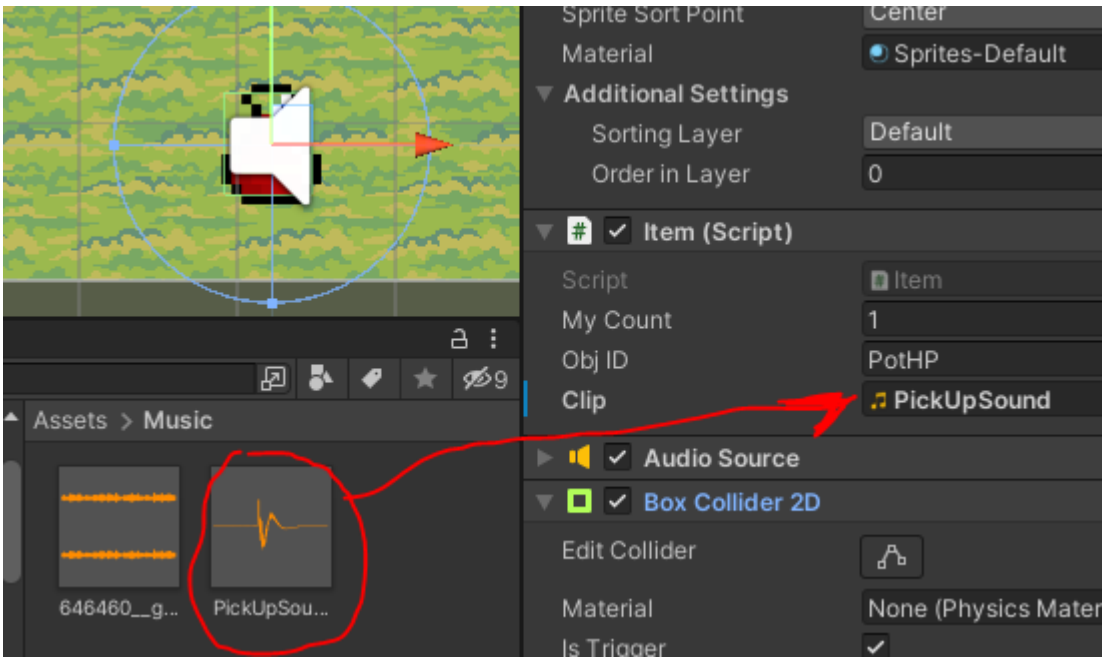
Claro que si quieren hacerlo más sencillo o no poseen el código de la clase 8, pueden simplificarlo de esta manera:

```
 private void OnTriggerEnter2D(Collider2D collision){
   if (collision.CompareTag("Player")){
     aSource.PlayOneShot(clip);
     Destroy(gameObject);
        }
    }
```

 ¡Ya tenemos nuestro código armado! Nos falta decidir qué sonido queremos escuchar.

Para esto, podremos ingresar a la página https://freesound.org/ y buscar el sonido que deseemos.

Una vez descargado, lo pondremos en nuestra carpeta de sonidos dentro de Unity y lo colocaremos en nuestra variable usando el inspector

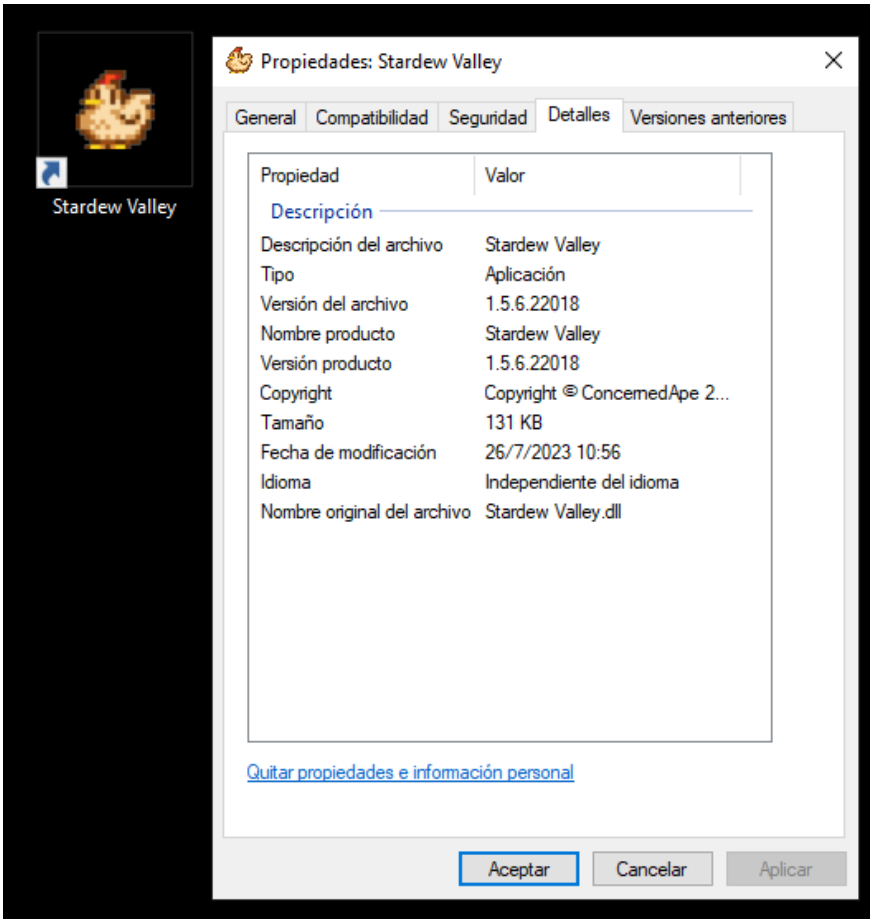


Y listo! Ya tenemos nuestro sonido reproduciéndose cada vez que agarremos un objeto

Build.

Información de nuestro juego.

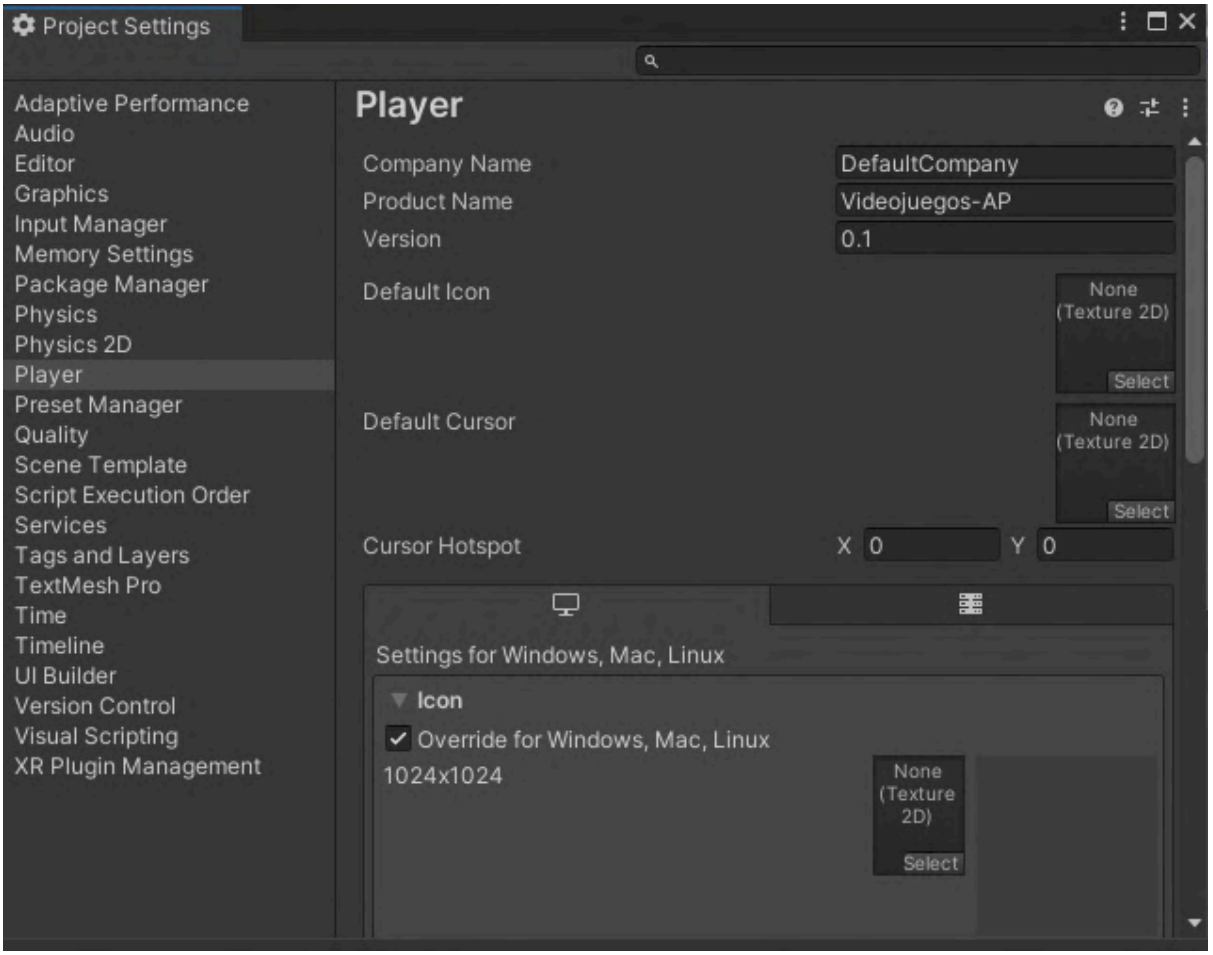
Cuando instalamos un videojuego, sus desarrolladores nos brindan toda la información necesaria para poder identificarlo, desde la versión del juego (que puede variar con el paso del tiempo y los hitos del desarrollo del mismo), su nombre y un ícono. Pero ¿para qué nos sirve reconocer esta información?



Principalmente para poder reconocer sus características sin necesidad de iniciar el juego, identificar compatibilidades, saber cuánto espacio de almacenamiento ocupará en sus equipos, etc.

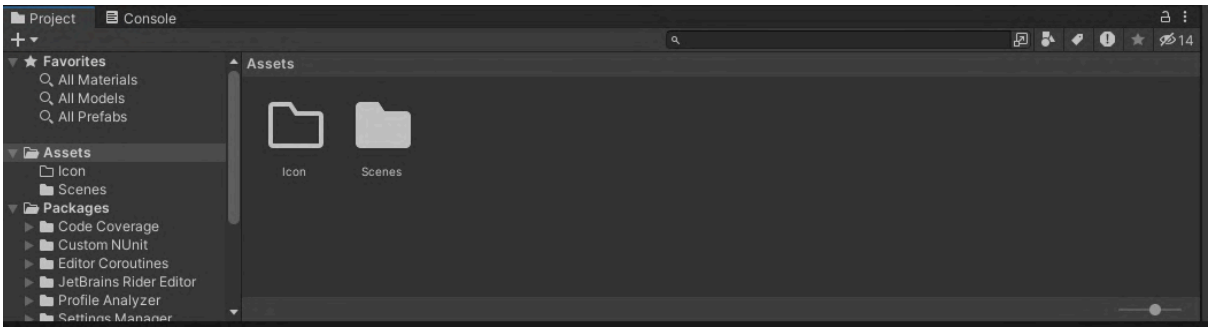
Nuestro rol como game devs nos permite brindar esa información a nuestros usuarios finales. Esto lo podemos hacer dentro del motor de videojuegos que estemos utilizando para desarrollarlo. En nuestro caso vamos a verlo con Unity.

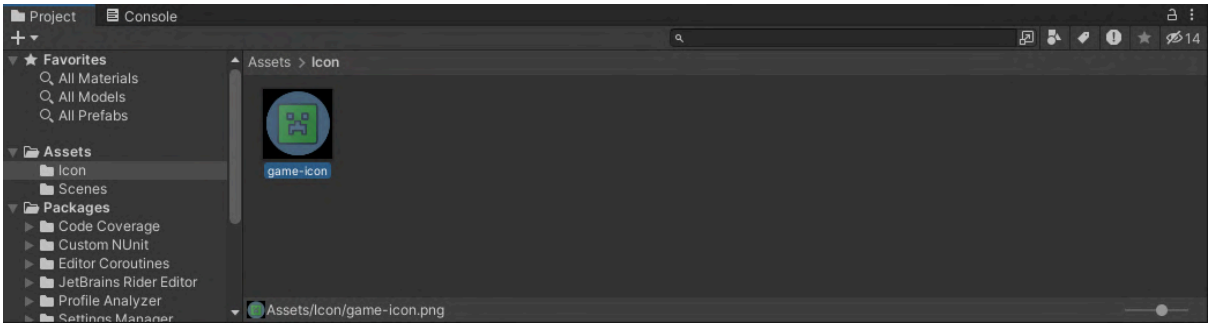
Empezando por los íconos. Para eso vamos a ir a ***Edit*** → ***Project Settings*** → ***Player***



Dentro de *Player* vamos a ir a la opción *Default Icon* y cuando hacemos click en *Select*, el motor va a pedir que carguemos un archivo del tipo *Texture 2D*, es decir, esas imágenes que generalmente utilizamos en formato PNG para cargarlas como texturas a nuestros materiales.

Esas texturas se guardan dentro de nuestra carpeta Assets del proyecto. Si aún no tenemos una imágen cargada para utilizar como ícono, entonces vamos a crear una carpeta dentro de nuestro proyecto llamada *"Icon"*. A la que le vamos a importar la imágen en formato PNG que vayamos a utilizar.



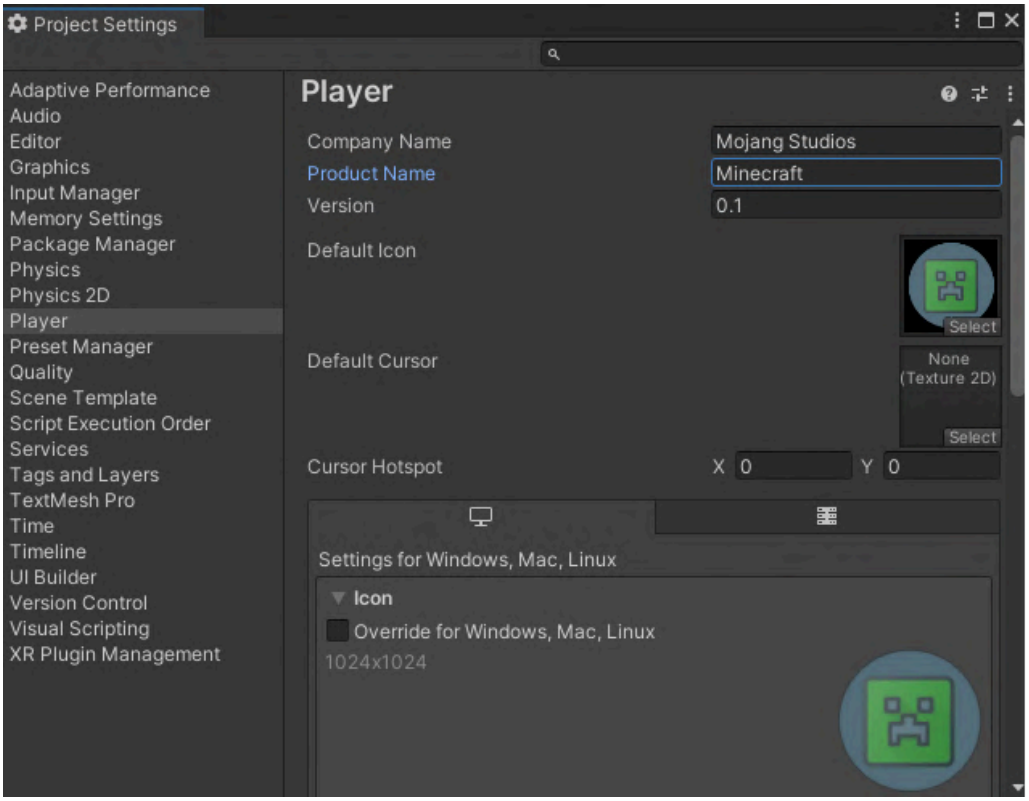


SUPERTIP: Recuerden que tiene que ser distintiva así reconocemos el juego.

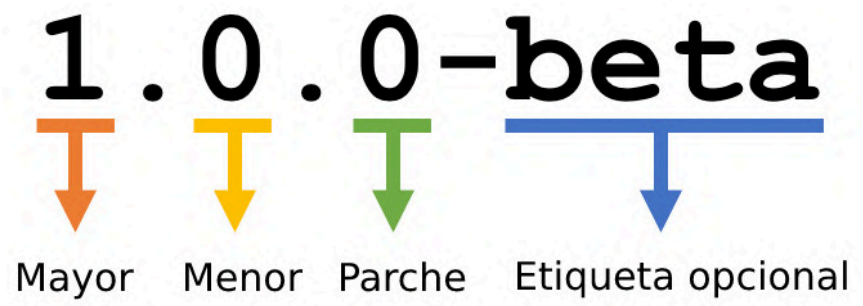
El tamaño de la imágen suele ser estándar: 64 x 64 píxeles o 512 x 512 píxeles, siempre que la imágen esté en formato PNG la calidad se va a conservar.

Ahora que tenemos importada nuestra imágen, volvamos a Player dentro de Project Settings.

Cargamos nuestro ícono y vamos a ver más opciones de personalización como el nombre de nuestra compañía, el nombre del juego y la versión en la que se encuentra.



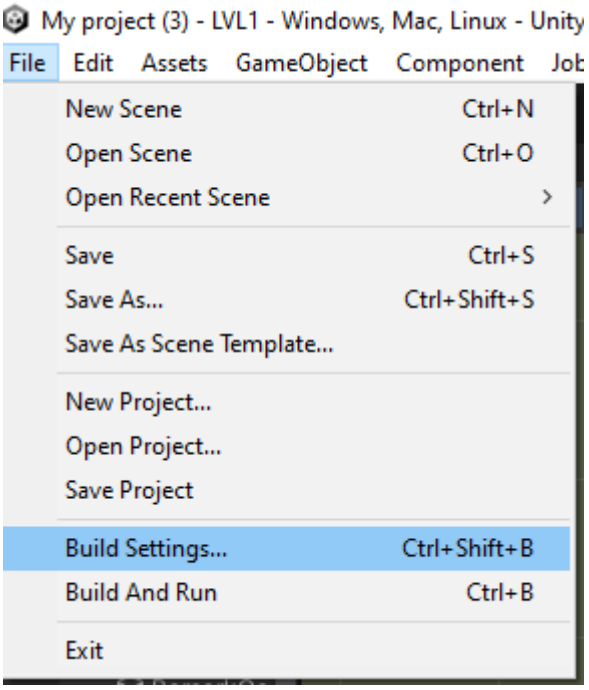
La modificación de versiones de nuestro proyecto dependerá de la magnitud de los cambios que realicemos y se modifican de derecha a izquierda. A esto lo denominamos “sistema de versionado semántico".



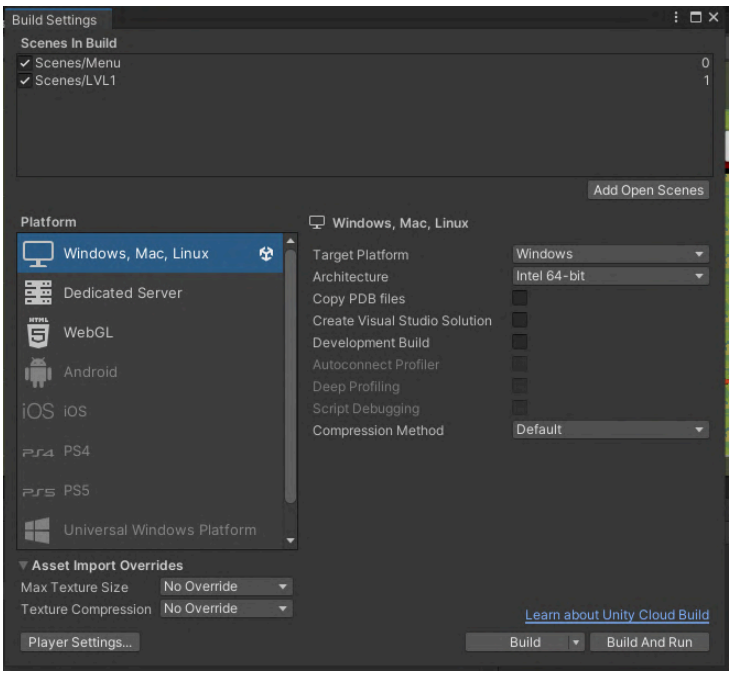
Configuración del proyecto y ejecutable.

Ahora que tenemos toda la información del juego necesaria, vamos a finalizar con la configuración del proyecto y realizar un primer ejecutable.

File → Build Settings

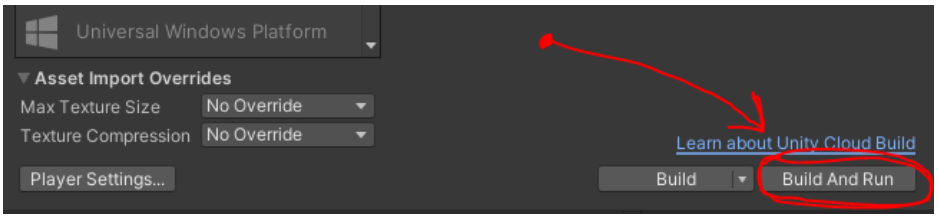


En **Build Settings** vamos a poder modificar la plataforma en la que queremos que nuestro videojuego sea jugado. Podemos elegir PC (con distintos sistemas operativos), Android, iOS y consolas de videojuegos entre otras.



Seleccionamos las escenas que queremos que formen parte del juego y también el orden en el que queremos que se ejecuten.

Hacemos click en el botón ***Build and Run***.



Nos abrirá el explorador de archivos que nos pide seleccionar una carpeta en la que se va a alojar el ejecutable. Podemos guardarlo fuera del proyecto o crear una carpeta dentro del mismo. Hacemos click en “Seleccionar carpeta” y el proyecto comenzará a compilar.

DATO: Si el script tiene errores de compilación el proyecto **no** se va a exportar. Por lo que es importante que revisemos en detalle cada uno de los scripts asociados a nuestros objetos.

Una vez terminado, basta con ingresar a la carpeta creada en el espacio seleccionado y encontrarás el Ejecutable junto con todos los archivos necesarios para inicializar el juego.

¡Listo, ya tenés tu videojuego exportado y listo para jugar!

Ejercicios prácticos:

¡Audios! Assemble!

1) Colocar música de fondo y algunos sonidos a elección según las situaciones. Por ejemplo: cuando atacamos, morimos, recibimos daño, el oponente ataca, caminamos, etc.
2) Crear el Build de tu juego y subirlo a una carpeta de Drive para poder realizar la entrega.

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad