

«Talento Tech»

Videojuegos

Clase 02





Clase N° 2 | Conceptos básicos

TEMARIO

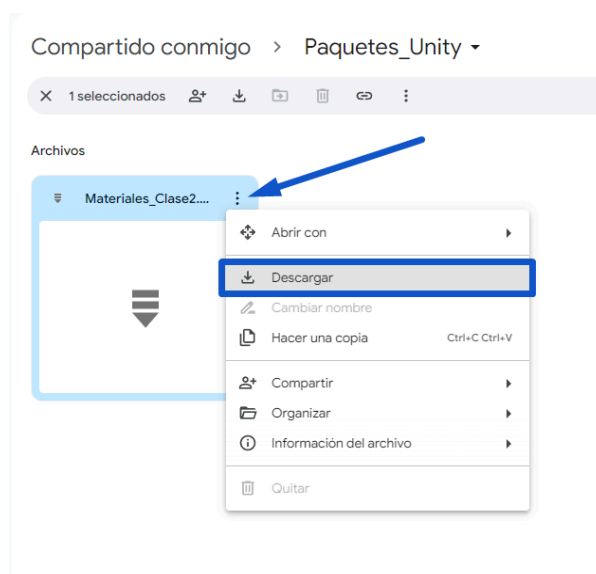
- Unity Packages.
- Materiales: texturas y tiling.
- Componentes: Rigidbody, Colliders
- Script: Funciones y operadores básicos.

Descarga de Unity Packages

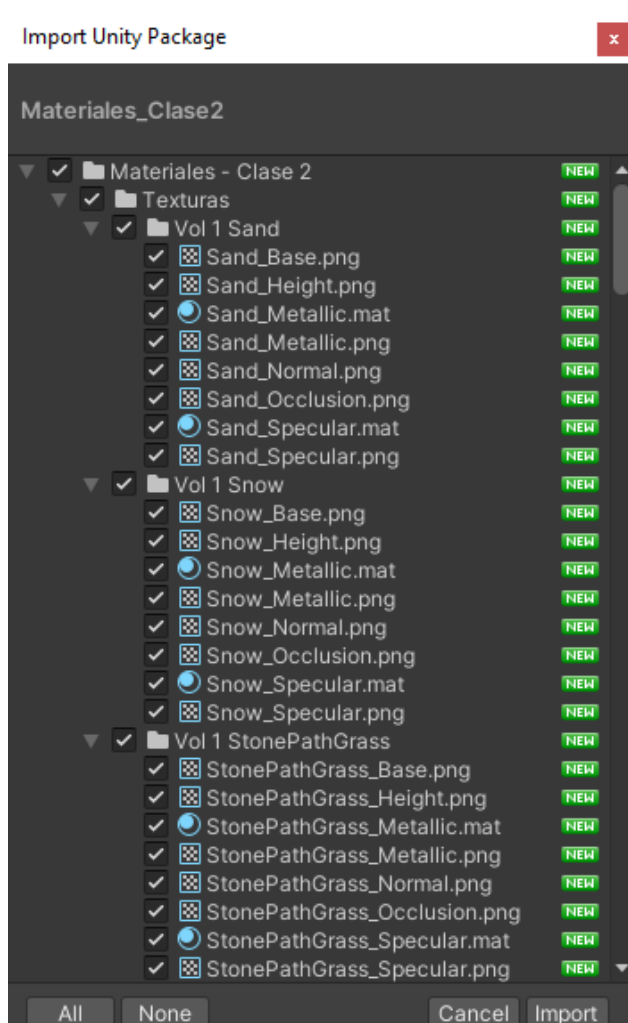
Antes de comenzar la clase, vamos a hablar sobre los **Unity Packages (paquetes de Unity)** que son un **conjunto de assets** que podemos armar y/o descargar para mover datos de nuestro juego con mayor facilidad. En esta clase vamos a descargar un paquete de materiales para que podamos hablar sobre estos.

📁 **Link de la carpeta con los paquetes:** [PAQUETES DE UNITY](#) 📁


Dentro de la carpeta, vamos a descargar el archivo llamado “*Materiales_Clase2*”.



Una vez descargado, simplemente tenemos que hacer doble clic sobre el archivo para que Unity se encargue de importarlo. Cuando nos aparezca la ventana con la lista de materiales, le damos a “Import” y esperamos.

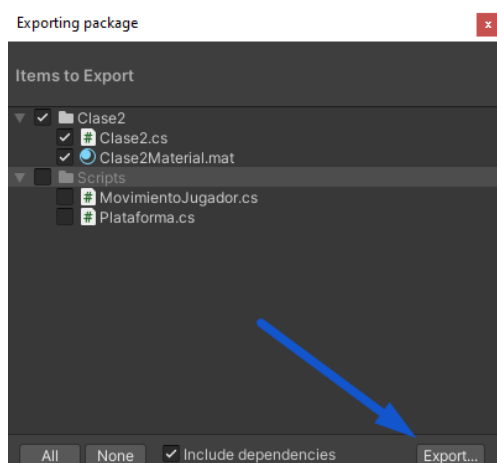


Si hicimos todo bien, deberían aparecer los materiales en nuestra ventana de Project, donde se encuentran todos los assets de nuestro proyecto.

 **Datazo: ¡Te invitamos a explorar *Unity Assets Store* para descargar más paquetes!**



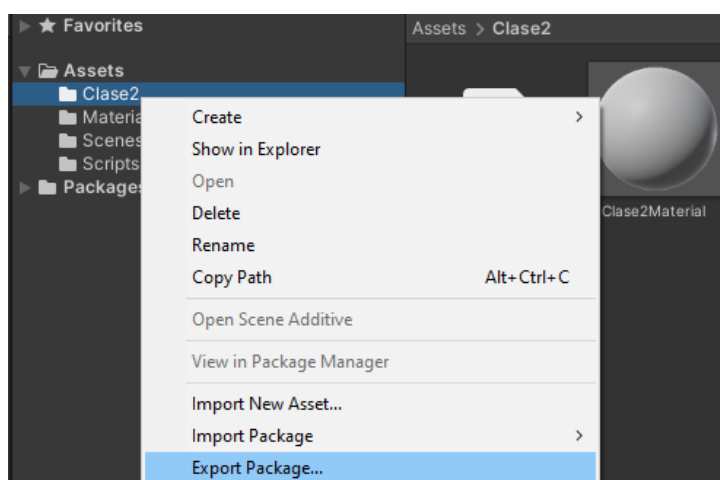
Muchos son pagos pero hay paquetes gratuitos: <https://assetstore.unity.com>



Creación de Unity Packages

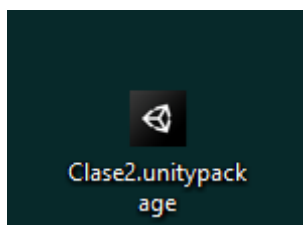
Como vimos, los paquetes son muy útiles para compartir y usar recursos en Unity. Ahora veremos cómo crear nuestros propios paquetes para facilitar la distribución de contenido con compañeros o para entregar desafíos en la plataforma.

En este ejemplo, observamos un material y un script ubicados dentro de una carpeta llamada **'Clase2'**. Para convertirlo en un paquete, simplemente hacemos **clic derecho** en la mencionada carpeta y seleccionamos **'Export Package'**.



Asegurémonos de exportar únicamente la carpeta deseada y hagamos clic en **'Export'**:

Después, seleccionamos una carpeta de destino, ★ **¡Y listo!** ★ Ahora podemos **compartir** nuestro paquete con otras personas, ya sea por Mail, Drive o Discord si el paquete no es muy pesado.

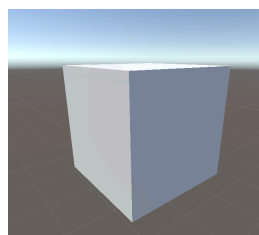
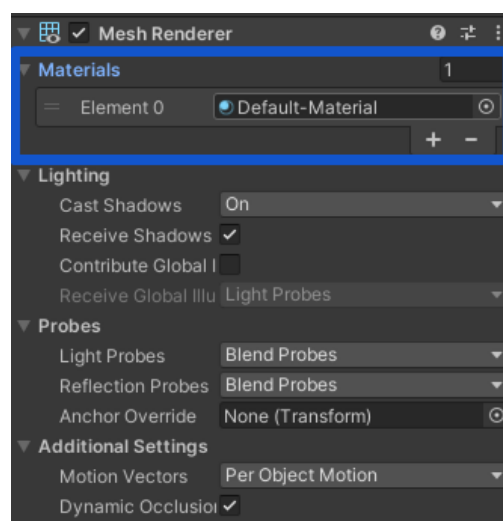


Materiales

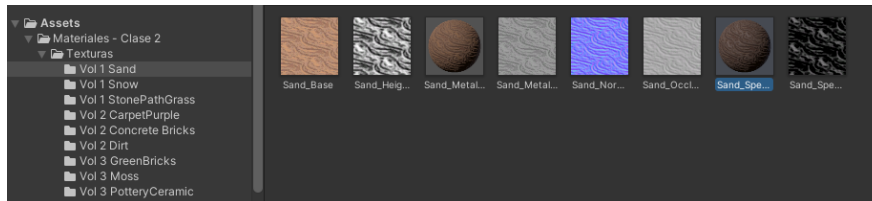
Los materiales en Unity son un **conjunto de texturas** que le podemos **aplicar** a los **objetos**, estos le van a otorgar una estética particular a nuestro juego, dependiendo de qué

queramos lograr. En el paquete que descargamos vamos a encontrar muchos tipos de materiales, tierra, arena, nieve, piedra y muchos más.

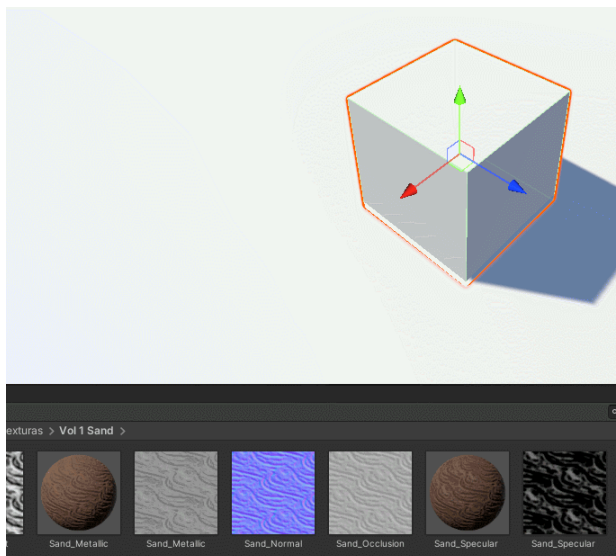
Si seleccionamos el cubo que hicimos la clase pasada, vamos a la ventana **Inspector** y buscamos el **componente MeshRenderer**, vamos a poder encontrar un apartado de **Materials** donde vamos a poder ver el material actual del objeto, en este caso tiene el material predeterminado, **“Default-Material”**.



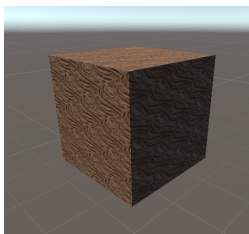
Dentro de nuestra ventana **Project**, vamos a **abrir la carpeta** de los materiales y buscar alguno que nos llame la atención.



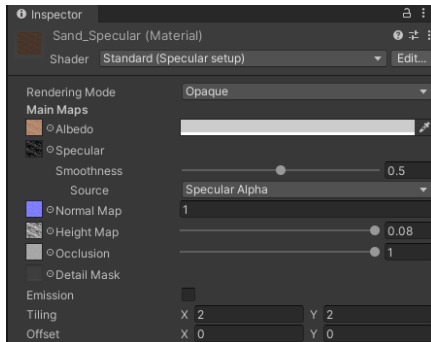
Si se lo queremos **añadir** a nuestro **objeto** lo vamos a pinchar con el mouse y lo vamos a arrastrar hasta el objeto y vamos a soltarlo allí.



Los **materiales** están representados como **esferas**, estos son los que queremos utilizar, el resto son **texturas** que conforman el producto final y que si prestamos atención notaremos que son **imagen en 2D**. En este caso elegimos el material *Sand* y nuestro cubo luce totalmente diferente:



Si seleccionamos un material, vamos a poder ver algunos **parámetros** en la **ventana Inspector** que podemos modificar para lograr distintos resultados:



Algunos de los cambios más interesantes que podemos hacer son:

Albedo: Podemos **cambiar el color** del material, también podemos arrastrar una textura en este apartado.

Smoothness: Que tan **suave** se ve.

HeightMap: Cuanta **sensación de profundidad** refleja.

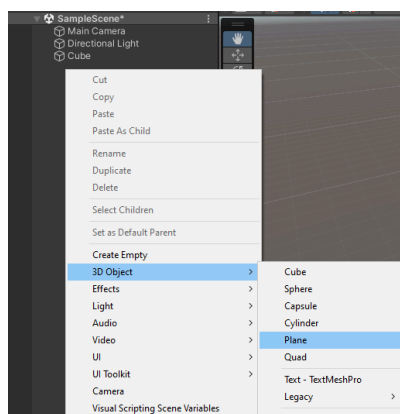
Tiling: Cuantas veces se **repite la textura** en uno de sus ejes (**X**, **Y**)

¡Probemos cambiar algunos valores para ver como queda nuestro material!

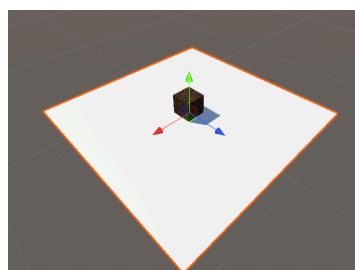
Rigidbody

Ahora vamos a crear un suelo para nuestro cubo, ya que vamos a ver dos **componentes básicos** que conforman las **físicas** de Unity.

Para crear un suelo vamos a hacer lo mismo que cuando hicimos el cubo, solo que esta vez vamos a buscar entre la lista de objetos 3D, un plano (Plane):

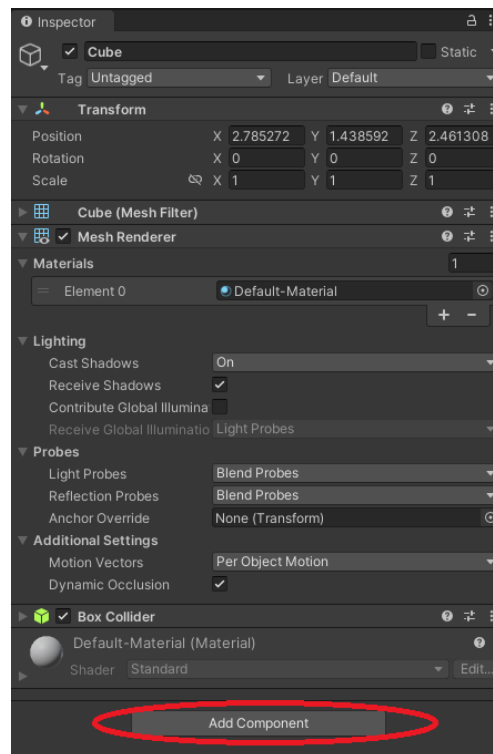


Una vez creado, vamos a acomodarlo para que quede por debajo del cubo, además, vamos a procurar tener la cámara apuntando hacia el cubo de forma que en la **ventana de Game** se pueda ver con claridad lo que pasa en la escena.

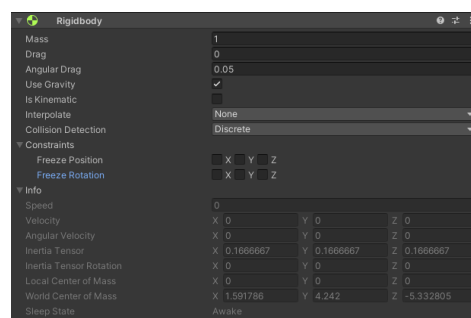


En la situación actual, si le damos **play** al juego, vamos a notar que el cubo queda suspendido en el aire; esto lo queremos cambiar, dándole propiedades físicas al cubo para que pueda ser afectado por la gravedad. **Para eso, tenemos que agregarle un componente llamado *Rigidbody*.**

El **Rigidbody** es un componente, y estos se agregan desde el *Inspector*, seleccionamos nuestro cubo, y buscamos un botón que dice “*Add Component*”, y escribimos en el buscador que nos aparece el nombre del componente que deseamos agregar.



Una vez agregado, nuestro objeto adquiere propiedades físicas. Si probamos darle play al juego, vamos a notar que ahora sí el cubo es afectado por la gravedad.



Dentro del componente vamos a poder modificar varios parámetros para lograr distintos resultados, alguno de los más importantes son:

Mass: Modifica la masa del objeto (más masa, más inercia o resistencia al cambio de movimiento de un cuerpo)

Use Gravity: Podemos o desactivar la gravedad en este objeto.

Is Kinematic: Si lo activamos, el objeto deja de ser afectado por la física pero puede moverse por otros medios.

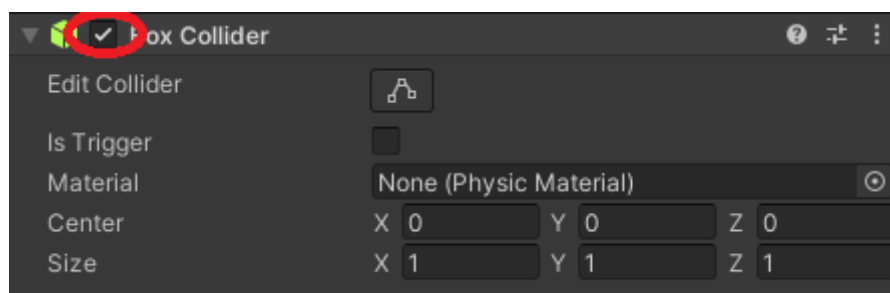
Constraints: En esta sección, podemos “congelar” un eje de rotación o de movimiento, resulta muy útil en múltiples ocasiones.

Colliders

Los colliders son los **componentes** que le agregan **colisión a un objeto**, pueden adquirir distintas formas y cumplen un rol muy importante.

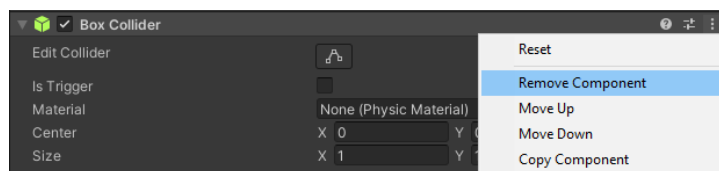
Vamos a notar que nuestro cubo cuando lo creamos ya viene incluido un componente *Box Collider*, es decir una colisión con forma de caja.

Podemos probar que pasaría si no tuviera este componente, para esto podemos desactivarlo haciendo clic en el check que está al lado del nombre del componente, este método sirve para todos los componentes (menos el *Rigidbody*).



De esta forma nuestro cubo va a ser afectado por la gravedad, pero no va a colisionar con el suelo, por lo tanto va a caer hasta el infinito.

También podemos borrar un componente. Para esto hay que ir a los tres puntos que se encuentran a la derecha del componente, y seleccionar *Remove component*:



Si optamos por borrar el componente, asegurémonos de agregarlo de nuevo para que el cubo se comporte como deseamos.

Funciones básicas

Ahora vamos a hablar de programación, veremos un concepto fundamental que nos va a servir para hacer nuestros juegos: **las funciones**.

Vamos a crear un script nuevo, lo vamos a llamar "Clase2" y vamos a echar un vistazo a lo primero que aparece cuando creamos un script:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[Script de Unity | 0 referencias]
public class Clase2 : MonoBehaviour
{
    // Start is called before the first frame update
    [Mensaje de Unity | 0 referencias]
    void Start()
    {
    }

    // Update is called once per frame
    [Mensaje de Unity | 0 referencias]
    void Update()
    {
    }
}
    
```

Acá podemos ver dos funciones: **Start()** y **Update()**, el primero ejecuta código en el primer frame desde que el objeto aparece y el segundo ejecuta código una vez por frame (cuadro).

Pero todavía no explicamos lo que es una función: Las funciones **son bloques de código** que podemos colocar en los programas, para hacer una tarea específica. Para que se entienda vamos a hacer algo práctico. Vamos a hacer una función

cambiar el valor de una variable, que vamos a hacer de cuenta, representa la vida de nuestro personaje.

Declarando una función

Para usar una función **primero tenemos que declararla**, al igual que con las variables, no podemos usar cosas sin crearlas primero, declarar en este caso lo podríamos usar como sinónimo de crear. Vamos a borrar las funciones de *Start()* y *Update()*, y lo vamos a reemplazar vamos a escribir lo siguiente:

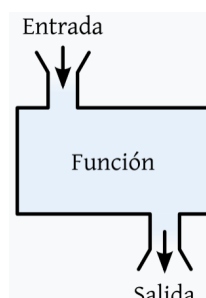
```

Script de Unity | 0 referencias
public class Clase2 : MonoBehaviour
{
    int vida = 0;

    0 referencias
    void CambiarVida(int a)
    {
        vida += a;
    }
}
    
```

No te asustes si no entendés, vamos a repasar el código paso por paso para que podamos entender las distintas partes que conforman una función.

Punto 1: Las funciones como dijimos, son bloques de código que cumplen una función, y tienen un output y un input, es decir, pueden recibir información y devolver otra.



El “void” (vacío en inglés), que declaramos al principio, es el tipo de valor de salida que va a tener la función, es decir, que este no va a devolver un valor, pero podría hacerlo si quisiéramos asignando otro tipo de variable (recordemos los tipos de variables de la clase anterior).

Punto 2: El nombre de nuestra función. Esta parte es muy importante, ya que el nombre que elijamos para nuestra función tiene que ser una abstracción del código que contenga adentro. Si vamos a hacer una función para saltar, lo ideal es que la función se llame “Saltar” y no “DinosaurioRosa”. Por una cuestión de convención, noten que las funciones tienen la primera letra en mayúscula y no usamos espacios, separamos las palabras con otras mayúsculas.

Punto 3: Las funciones siempre terminan con paréntesis, como `Start()` y `Update()`, como dijimos en el **punto 1**, las funciones tienen entrada y salida; los paréntesis representan la entrada, y pueden recibir información o no. La información que recibe se llama **parámetros**, en este caso va a recibir un parámetro, vamos a recibir un “int” que se va a llamar “a” (podría ser cualquier nombre). Cuando utilicemos la función se mostrará cómo ingresamos información a nuestra función.

Punto 4: Nuestra función va a tomar “a” y lo va a sumar a una variable que creamos con anterioridad, que es un número entero (int) y se va a llamar “vida”. Como no le asignamos ningún valor: “vida” es igual a cero.

Utilizando nuestra función

Ya tenemos nuestra función declarada, pero esto **no** significa que **va a ejecutarse sola**, tenemos que llamar la función **dentro de un contexto**. Con contexto nos referimos a en qué momento del juego correríamos esta función; en este caso concreto, lo ideal sería que la vida cambie en función de lo que esté sucediendo en nuestro juego. Si nos atacan o nos curamos podríamos usar esta función para actualizar la vida de nuestro personaje.

Pero no nos adelantemos, vamos a implementar nuestro código al principio de nuestro juego. Para eso existe la función **Start()**, la cual vamos a utilizar para que la vida tenga un x valor al comienzo.

Vamos a hacer lo siguiente:

```

Script de Unity | 0 referencias
public class Clase2 : MonoBehaviour
{
    int vida = 0;

    Mensaje de Unity | 0 referencias
    private void Start()
    {
        CambiarVida(100);
    }

    1 referencia
    void CambiarVida(int a)
    {
        vida += a;
    }
}
    
```

Volvimos a agregar la función de **Start()**, y dentro de las llaves escribimos el nombre de la función que acabamos de declarar.

Nótese que para llamar una función, sólo hay que escribir su nombre, y entre paréntesis le tenemos que dar un **valor de entrada**. El que le otorgamos será “a” y se lo va a sumar a “vida”. Si le pasamos un negativo se va a restar (porque si sumamos un número negativo, se resta).

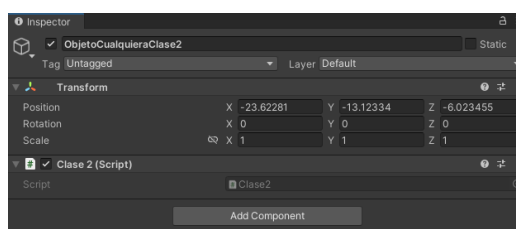
Cómo hicimos la clase anterior, vamos a agregar un **Print()** y vamos a **asignarle nuestro código** a un **objeto** en escena para que se ejecute el código:

```

Script de Unity | 0 referencias
public class Clase2 : MonoBehaviour
{
    int vida = 0;

    Mensaje de Unity | 0 referencias
    private void Start()
    {
        CambiarVida(100);
        print("Vida: " + vida);
    }

    1 referencia
    void CambiarVida(int a)
    {
        vida += a;
    }
}
  
```



Ahora cuando le demos play a nuestro proyecto. **Vamos a poder ver en la ventana Consola el valor de la variable “vida”.**



Desafío N° 2:

Buscar una imagen de tu gusto agregalo a tu proyecto y convertilo en sprite seleccionandolo, yendo a "TextureType" seleccionando "Sprite (2D and UI)

Crear un material en una carpeta aparte y asignarla a nuestro personaje

Asignarle el sprite al "Albedo" de nuestro material

No te olvides de **guardar** todos los cambios.

Tomá una **captura de pantalla** del personaje con el nuevo material

Subilas al espacio correspondiente del Desafío 2.



Buenos Aires
aprende
Agencia de Actividades para el Futuro

BA Buenos
Aires
Ciudad