«Talento Tech»

Desarrollo de Videojuegos

Unity 2D

Clase 01

Buenos Aires Ciudad

Clase N° 1 | ¡Bienvenidos a Unity!

Temario:

- Introducción a Unity
- Interfaz
- Scripts
- Variables
- OOP

Introducción a Unity

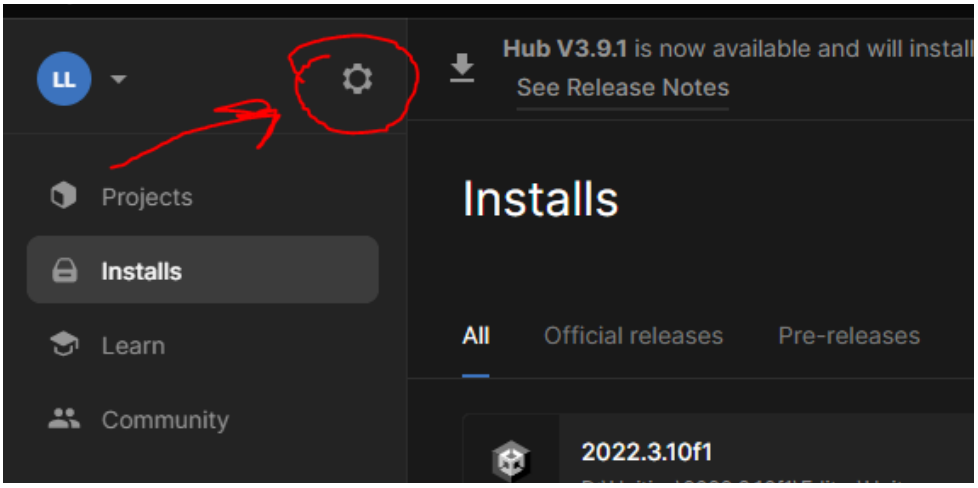
Unity es un **motor gráfico** de desarrollo de videojuegos y aplicaciones interactivas. Permite crear experiencias en 2D y 3D de forma accesible y amigable. Con herramientas intuitivas y una gran comunidad, es ideal para diseñadores y programadores que quieren dar vida a sus ideas, ya sea en videojuegos, simulaciones o realidad virtual. ¡Es como un lienzo mágico donde puedes crear mundos y personajes fascinantes!

Un motor gráfico es un software que facilita la creación de gráficos y animaciones en videojuegos y aplicaciones interactivas. Proporciona herramientas para manejar la física, la iluminación, los sonidos y otros elementos visuales, permitiendo a los desarrolladores centrarse en la creatividad sin preocuparse por los aspectos técnicos.

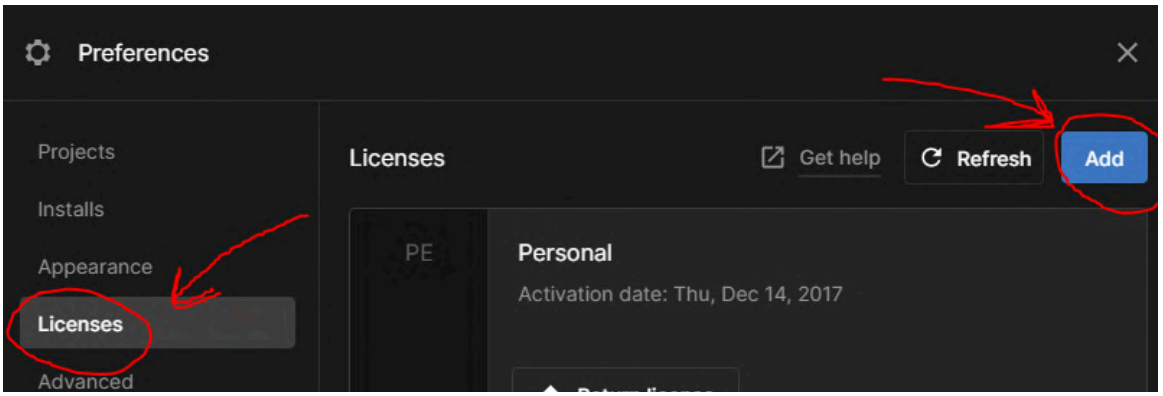
Instalamos Unity

¿Cómo se instala?

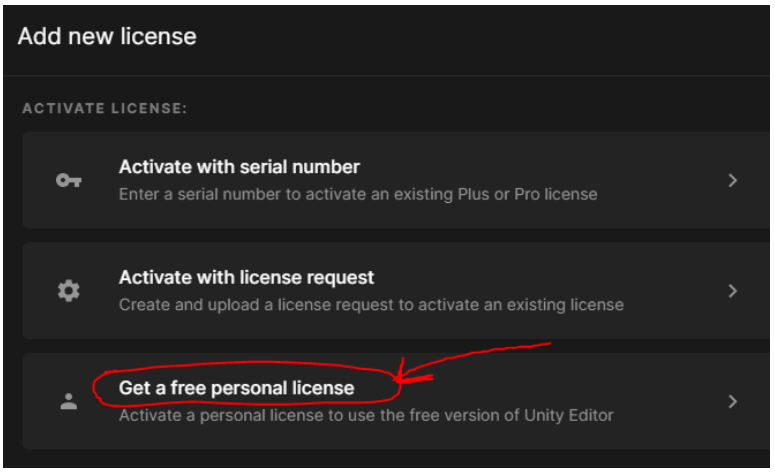
1. Primero tenes que entrar a: https://unity.com/es/download
2. Descarga UnityHub
3. Instalalo
4. Preferencias



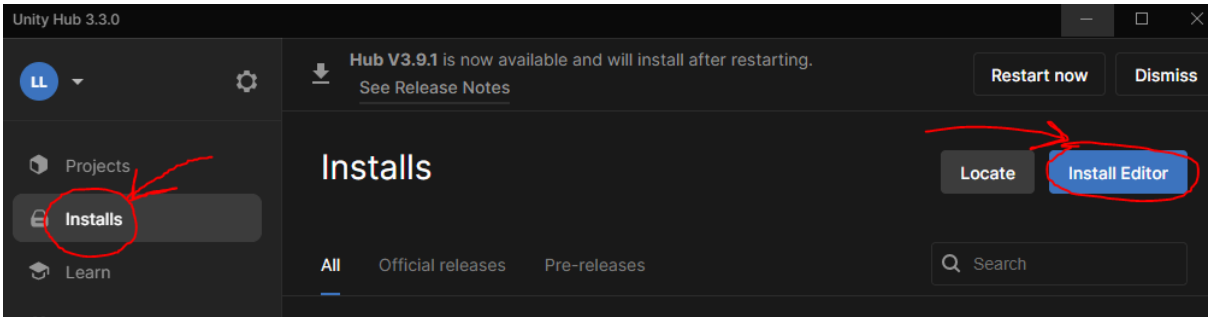
5. Licenses -> Add



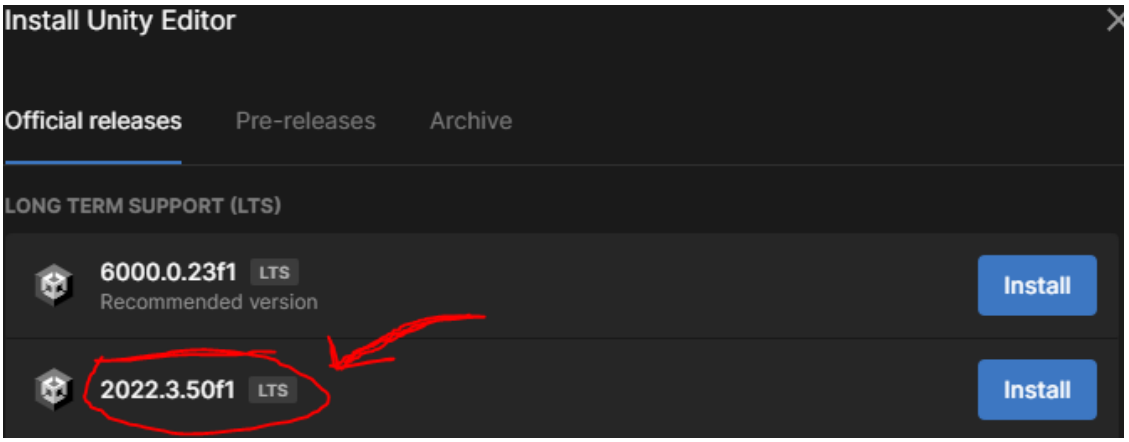
6. Free personal license



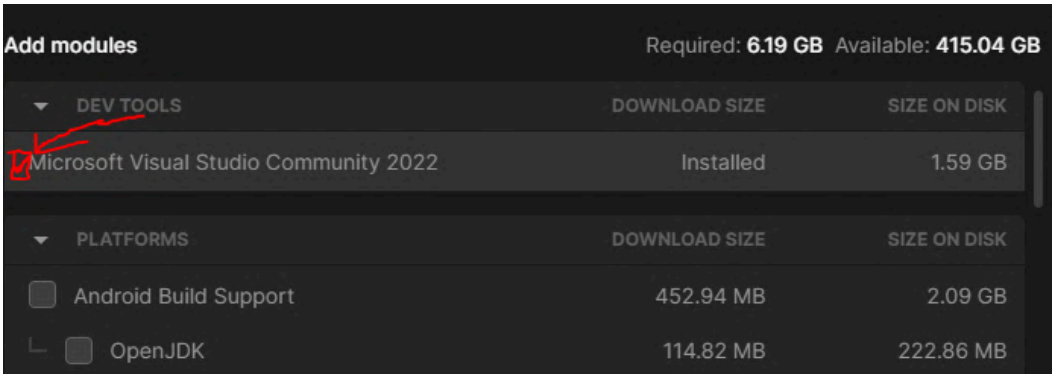
7. Installs -> Install Editor



8. Instalar la V. 2022.3.50f1



9. Al instalar den check al Visual Studio

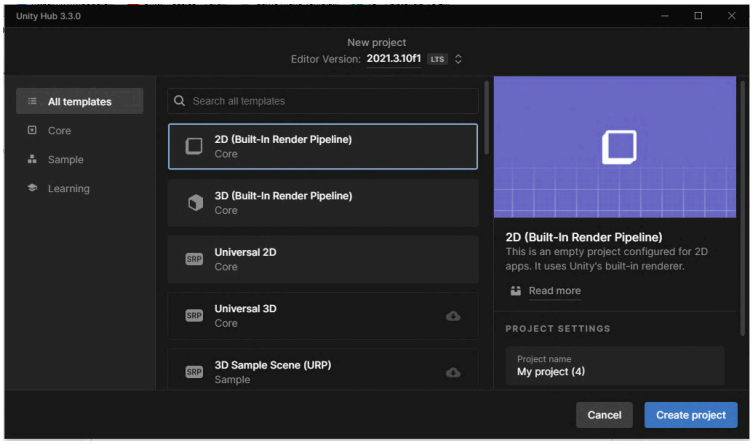


10. Revisen que, cuando se instale el Visual Studio. Tenga esto en Check… Y Listo!

La ventana del Proyecto.

Proyectos y Escenas

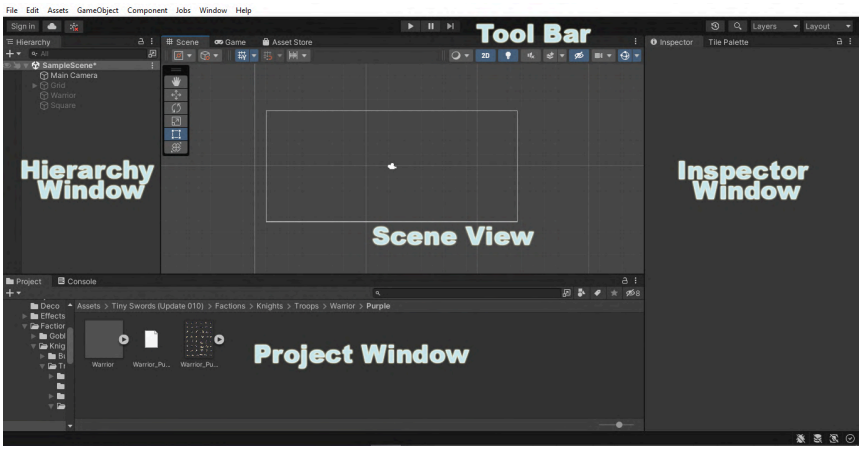
Creá un proyecto 2D: En el Unity HUB, seleccioná “New Project” y luego seleccioná “2D Core”, elegí un nombre para tu proyecto y después seleccioná “Create Project”.



La Interfaz.

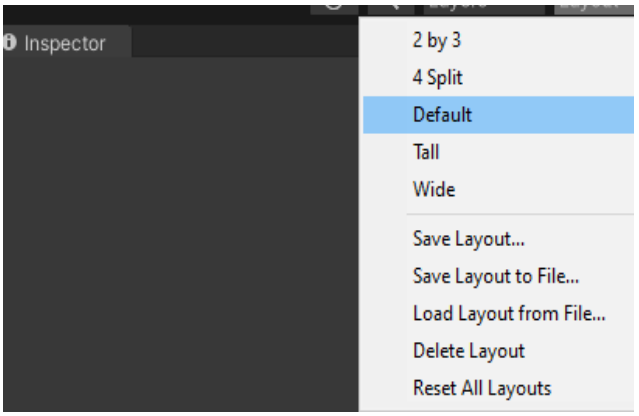
Te proponemos que, junto con este tema, **abras Unity** y repases su interfaz para poder reconocer sus características. Tomate tu tiempo, la **ventana principal del editor** se **compone** de ventanas con **pestañas** que pueden ser **re-arregladas**, agrupadas o desagrupadas y minimizadas. Esto significa que el **aspecto del editor** puede ser **diferente** entre **proyectos** y **desarrolladores/as**, según la preferencia personal y del tipo de trabajo que está haciendo.

El **arreglo predeterminado** de las ventanas te da un **acceso práctico** a las **ventanas** más **comunes**. Si aún no estás familiarizado con las ventanas diferentes en Unity, podes **identificarlas por su nombre** en la pestaña. Las ventanas más comunes y útiles se muestran en sus posiciones por defecto, abajo:



📌 ***TIP importante:***

Si por alguna razón, desordenaste la interfaz y querés volver a dejarla como estaba, podés ir a la esquina superior derecha a donde dice "Layout" y seleccionamos "Default" (predeterminado). Esta acción restaura el orden de las ventanas a su estado inicial.



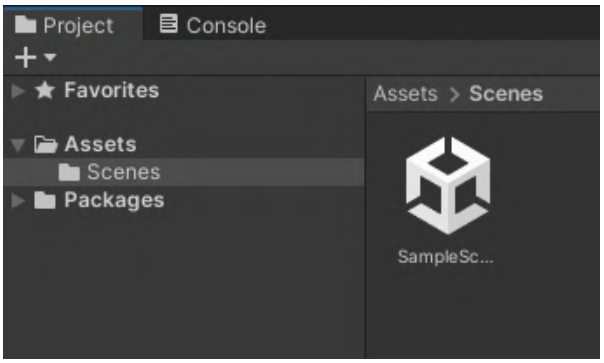
A continuación vamos a empezar a recorrer estas pestañas una por una para comenzar a familiarizarnos con cada una de ellas.

Project Window (ventana del proyecto).

Para arrancar vamos a buscar la pestaña *Project*. En ella se guardan las carpetas de nuestro proyecto.

Esta **pestaña** es el **núcleo** de nuestro proyecto, acá se va **almacenar** toda la **información** que el dispositivo (en este caso una computadora) necesita para **ejecutar** nuestro **juego**, así como todos los recursos como sonido, modelos, código etc. Esta pestaña es una **representación** de una carpeta que existe en tu disco de almacenamiento al cual podemos **acceder** para manipularlo según convenga.

La **carpeta** *Assets* va a contener la mayoría de los elementos que vamos a necesitar para realizar nuestro trabajo.

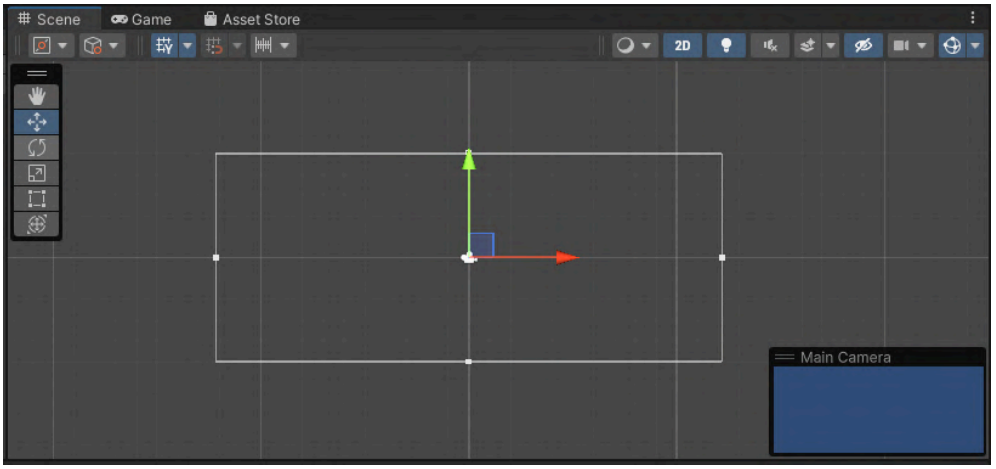


The Scene View (vista de escena).

Para empezar, una **escena** es un **conjunto de objetos** que **interactúan** según dispongamos. Esto se da gracias a **código escrito en C#**, es decir, un lenguaje de programación muy utilizado en desarrollo de software. Una escena **puede ser** un **nivel** de tu juego, o el **menú** principal, o hasta una **pantalla de carga**.

Esta **ventana** permite la **navegación** y **edición** de tu escena. La *scene view* puede mostrar una **perspectiva 2D o 3D** dependiendo del tipo de proyecto en el que estés trabajando.

Es claramente una de las más importantes dentro de nuestro proyecto, ya que vamos a pasar la mayor parte del tiempo acá, porque es la forma en la que **interactuamos** con nuestros **objetos**.



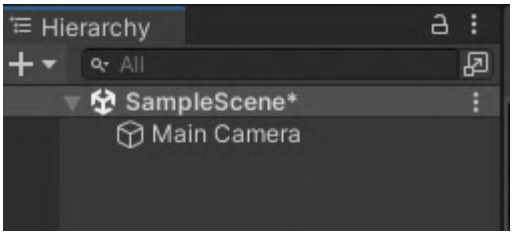
Antes de pasar a otras pestañas, vamos a ver cómo nos movemos por nuestra escena, utilizando una escena de pruebas con un cubo rojo (en las próximas clases veremos más detalles sobre materiales y texturas de objetos).

Manteniendo clic derecho o la **rueda del mouse** podemos **movernos** en los **ejes X/Y.**

Girando la **rueda del mouse** hacemos **zoom**.

Hierarchy Window (ventana de jerarquía)

Es una representación de **texto jerárquico** de cada **objeto** en la **escena**. Cada elemento en la escena tiene una entrada en la **jerarquía**, por lo que **las dos ventanas** están inherentemente **vinculadas**. La jerarquía revela la estructura de cómo los objetos están agrupados el uno al otro.



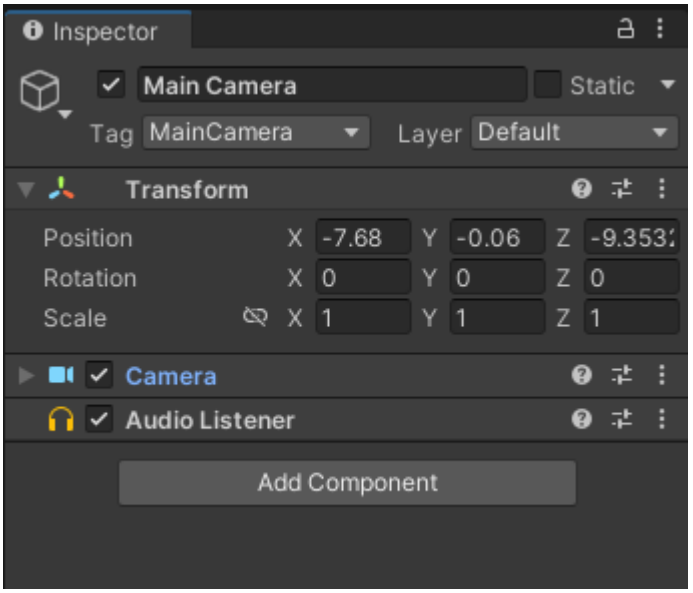
La ventana de jerarquía es una lista de los objetos en escena.

En esta ventana podemos ver **de qué está compuesta la escena actual**, y todos los elementos de esta lista son objetos, todos, aunque sea una cámara, una luz, o un objeto invisible; siempre hablamos de objetos en Unity.

Podemos ver que en esta escena existen dos objetos: la **cámara**, que siempre tiene que estar para que podamos **renderizar** nuestro juego (la renderización es el proceso de generación de imagen), y la **luz** de la escena llamada ***"Directional Light"*.**

Inspector Window (ventana del inspector).

Nos permite **visualizar** y **editar** todas las **propiedades** del **objeto** actualmente **seleccionado**. Entendiendo que diferentes objetos tienen diferentes propiedades.

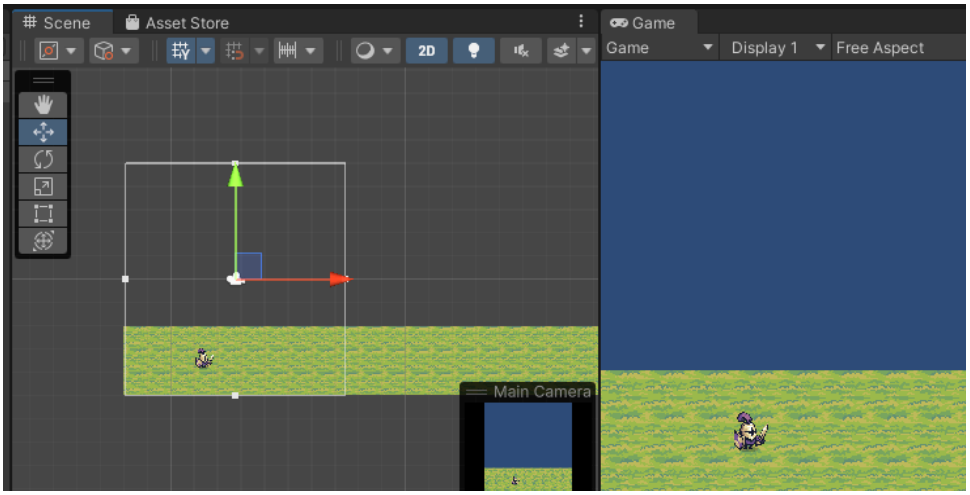


El inspector es una ventana muy importante, ya que acá **vamos a poder crear/editar componentes**. Estos son códigos que cumplen una cantidad infinita de funciones, pueden ser las físicas de un objeto, sus colisiones, su comportamiento, de qué color es y un largo etcétera.

La ventana de Game (o juego)

Esta ventana es el juego en sí, indica cómo se va a ver **nuestra escena cuando le damos play**. En general cuando estamos desarrollando un juego, necesitamos probar que todo funcione. Es por esto que vamos a estar usando esta herramienta para ver todo lo que podemos hacer en nuestra escena a medida que avancemos.

La vista va a ser siempre desde la cámara activa, así que hay que tener cuidado y saber qué es lo que está **renderizando** antes de darle ***play*** a nuestro **proyecto**.



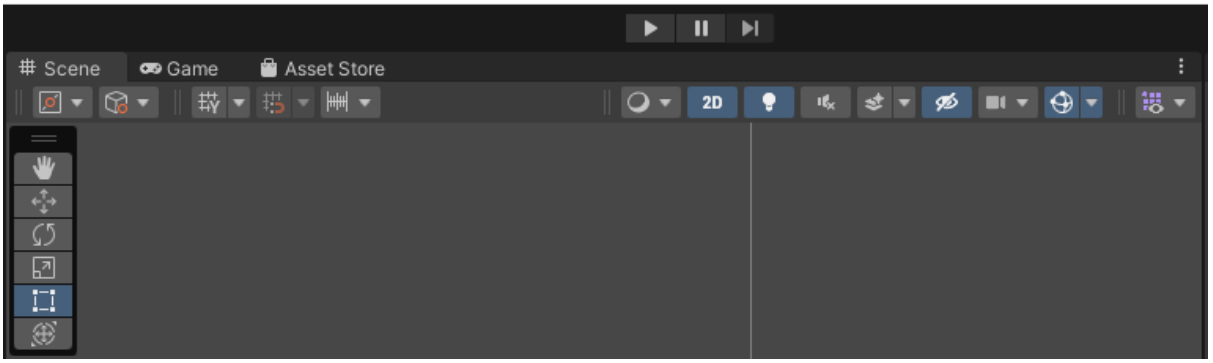
En la imágen podemos ver cómo separamos las pestañas de escena y de juego para poder trabajar de forma cómoda. Recuerden que pueden arrastrar las pestañas a discreción para dejarlo como la imagen anterior.

📌 **Super TIP:**

Si en la Escena nos posicionamos de la forma en que queremos que se vea y luego seleccionamos la cámara y vamos arriba en "GameObject" → "Align with view", la cámara se alineará a nuestra visión actual de la Escena, ahorrándonos el trabajo de estar moviendola a mano.

La barra de herramientas.

La barra de herramientas **proporciona** un acceso a las **características** más esenciales para trabajar. En la izquierda contiene las **herramientas básicas** para manipular la escena y los objetos dentro de esta. En el centro están los **controles de reproducción, pausa, y pasos**.



Cabe destacar que **no es una ventana**, y es una parte de la interfaz de Unity que no se puede reajustar.

Al comienzo, la interfaz de Unity puede resultar un poco compleja.

¡No te frustres! A medida que la utilicemos, se volverá cada vez más familiar y sencilla.

Recordá entonces que cada ventana es muy específica y es importante que reconozcas sus funciones.

Ventana	Características
Project	Donde se **almacenarán** nuestros **assets**.
Scene	Donde vamos a ver y modificar todos los objetos que estarán en el juego.
Hierarchy	Desde acá podemos crear los objetos de la escena.
Inspector	Acá también podemos modificar, pero en este caso, las propiedades de cada objeto por separado, añadirle componentes y cambiar valores a sus "variables".
Game	La ejecución del juego.

Y muy importante a tener en cuenta: la **barra** de herramientas de **arriba**, donde tenemos todo lo necesario para utilizar el motor y manipular la ejecución del juego.

GameObjects y figuras primas

Los **GameObjects** son **objetos** fundamentales en **Unity** que representan personajes, props, y el escenario. Estos no logran nada por sí mismos pero funcionan como contenedoras para **Components**, que implementan la verdadera funcionalidad.

¿Cómo crear un objeto?

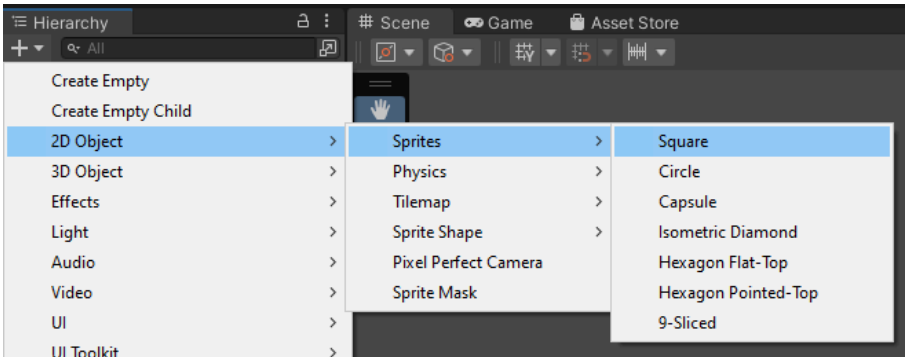
Vamos a seguir una serie de pasos para **crear** nuestro primer **objeto:**

1. Vamos a hacer **clic derecho** sobre el **espacio vacío** de la jerarquía. Luego se desplegará un menú conteniendo distintos objetos que se pueden **colocar dentro de la escena**

2. Seleccionamos **“2D Object”**; nos va a aparecer **Sprites**

3. En Sprites nos aparecerá una lista de **formas geométricas** en 2D, como un cuadrado, un círculo, una cápsula, etc.

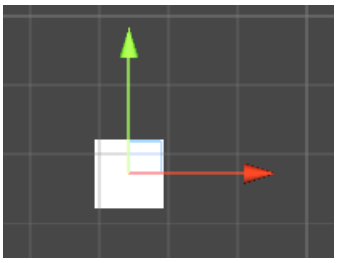
4. Vamos a seleccionar el **Cuadrado (“Square”)**.



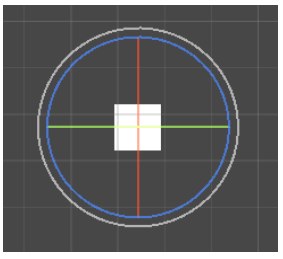
Interactuando con objetos

Volvamos a la ventana de escena, vamos a interactuar un poco con nuestro cuadrado. Existen tres atajos del teclado básicos que utilizaremos para interactuar con los objetos de la escena:

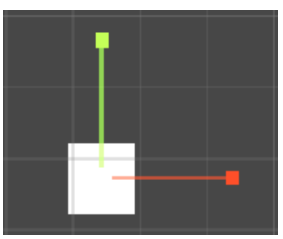
W para **mover objetos** en uno de sus dos ejes **(X,Y)**



E para **rotar objetos** en uno de sus tres ejes **(X,Y, Z)**



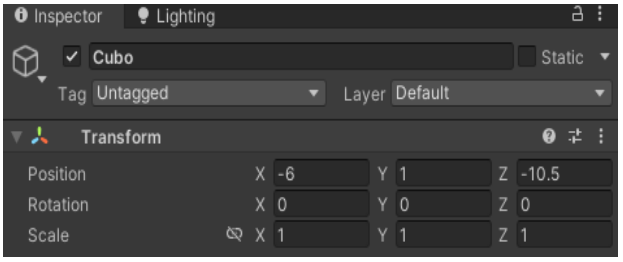
R para **escalar objetos** en uno de sus dos ejes **(X,Y)**



Componente Transform

El **transform** es un **componente** que tienen **todos los objetos del juego**. Transform **se encarga** de **manejar** la **posición**, la **rotación** y la **escala** de un **objeto**, las tres variables que estuvimos manipulando recientemente. En el Inspector vamos a poder **ver el valor** de estas variables **dividido** en sus respectivos **ejes (X,Y, Z)**.

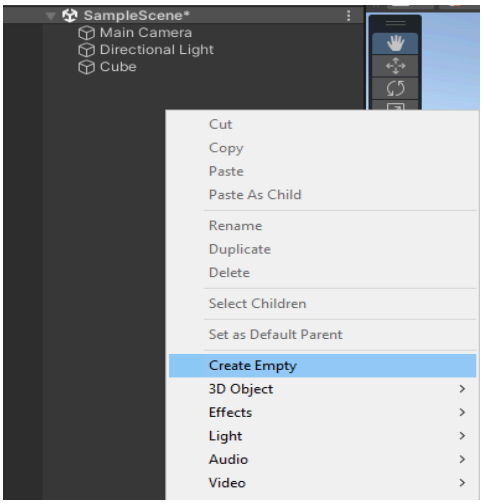
Con *transform* también vamos a poder **manipular** estas **variables** de forma **numérica**, es decir, asignándole un valor a la posición, por ejemplo.



Empty Objects

El propósito de un **Empty Object** es ser un **contenedor** o un **objeto de referencia** en la escena (CheckPoints, Spawners, etc).

Para **crearlo** realizamos el **mismo proceso** que con los **objetos**, pero, en vez de “2D Object”, iremos a **“Create Empty”**. En lo siguiente, verán que se creó un objeto nuevo en la jerarquía con el único componente siendo el “transform”, por lo tanto no tendrá ninguna representación visual. En este caso lo vamos a utilizar para agrupar nuestros objetos creados, dentro del mismo.



Luego de colocarlos dentro del Empty, podrán ver que la variable “Position” de sus objetos fue modificada. Esto se debe a que el mismo depende de su contenedor.

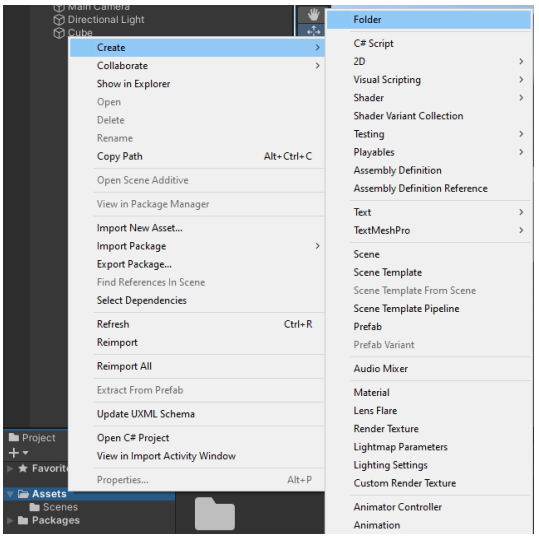
Los scripts

Los scripts son lo más importante de todo nuestro proyecto, ya que contienen **códigos** escritos por nosotros/as que hacen que los **objetos** se **comporten** de una forma u otra. Script significa “guión” en inglés, pero los vamos a llamar por su nombre original para que se vayan familiarizando con el término, ya que se extiende a todas las áreas de la programación.

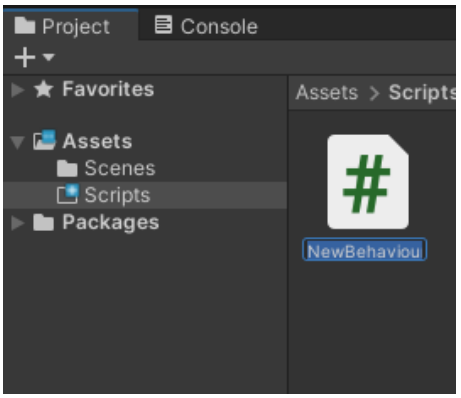
Los scripts en Unity están en un **lenguaje de programación** particular, llamado **C#**, y este lenguaje tiene sus **reglas** y **sintaxis** que tenemos que respetar para que funcione correctamente.

Es un lenguaje de programación creado por Microsoft, basado en Programación Orientada a Objetos (POO) que se utiliza para el desarrollo de diversos tipos de software, incluyendo videojuegos.

Creando nuestro primer script

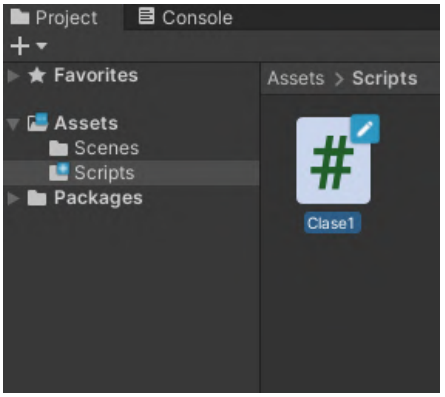


En la pestaña de **Project**, dentro de la carpeta de **“Assets”**, vamos a **crear** nuestra propia **carpeta** para ordenar nuestro proyecto, y la llamaremos **“Scripts”**. Es muy importante la organización para garantizar un mejor resultado.
Para esto vamos a hacer clic derecho dentro de la pestaña **“Project”, yendo a → “Create” → “Folder”**.



Una vez creada nuestra carpeta, repetiremos el mismo proceso, pero esta vez, seleccionando **“C# Script”**. Una vez creado, es muy importante *NO TOCAR NADA*, ya que **antes** debemos **nombrar** nuestro nuevo **archivo**.

Ahora tocará seleccionar el nombre de nuestro script: en todos los casos, **el nombre asignado está relacionado con el comportamiento o la utilidad del mismo.** Por ejemplo, un Script que tenga el código para mover un objeto, podría llamarse *"Movimiento"*.
En este caso, vamos a llamar nuestro archivo *"Clase1"*.



📌 **Super TIP importantísimo:**

Asegurate de que el nombre del Script sea siempre el mismo que el de la "Class".

Visual Studio.

Para poder editarlos vamos a necesitar un programa externo, el Visual Studio, el cual es descargado durante el proceso de instalación de Unity.

Si queremos utilizarlo correctamente, debemos enlazarlo siguiendo estos pasos:
1) Vamos a ir a la esquina superior izquierda de la pantalla y hacemos clic en:

Edit → Preferences → External Tools → External Script Editor: Visual Studio Community.

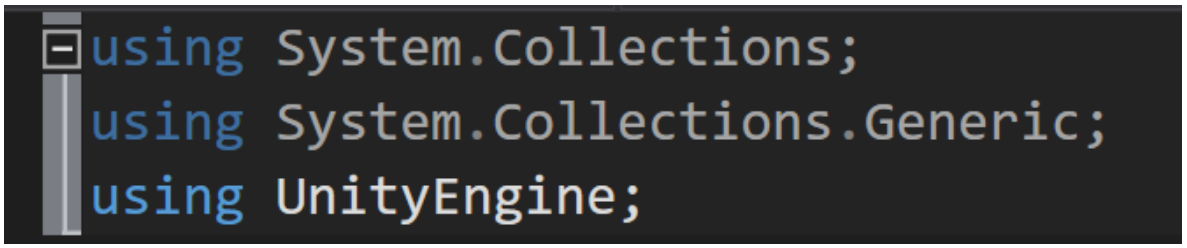
2) Una vez hecho esto, si hacemos doble clic sobre nuestro Script se va a abrir el programa.

¿Qué es un código?

El siguiente paso será realizar nuestro primer código. Pero primero intentemos entender qué es un código.

Por definición, es **una serie de instrucciones** elaboradas en un lenguaje informático/programación dirigidas a la "computadora" para que realice lo que le pidamos. Como todo lenguaje este tiene su estructura y sus reglas e iremos aprendiendo algunas de ellas al pasar las clases.

Para empezar, al abrir el Script en **Visual Studio**, verán en las primeras líneas de código, las llamadas **librerías** que son la "data"/información que **importa** nuestro archivo para poder trabajar. Si borramos "using UnityEngine;" no va a funcionar nuestro script.



Seguimos con la **"Public Class…"**. Esta "Clase" es la **sección** donde estará programado nuestro código, encerrará el **comportamiento del objeto** que queramos programar. Por eso, es muy importante que la **Clase posea un nombre adecuado** al comportamiento que estamos pensando, como por ejemplo "Movement", "Hero", "Enemy", "Potion".

Más abajo nos encontramos con las funciones **"Private Void Start()"** y **"Private Void Update"**. Son parte de la estructura general. Qué son las funciones y para qué sirven, lo veremos en Clase 2, por ahora vamos a enfocarnos en aprender acerca de las variables.

```
public class Clase1 : MonoBehaviour{

void Start(){// Start is called before the first frame update

}

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

}}
```

Una **variable** es un **espacio reservado** en la memoria de la computadora que almacena datos, y estos, justamente, pueden ser modificados y “variar”. A las variables se les coloca un nombre, también asociado a su uso. Por ejemplo, si quiero que contenga el dato de vida de mi héroe y otra de mi enemigo podría llamarlas “hpHero”, “hpEnemy” o “vidaHeroe”, “vidaEnemigo”. Recuerden que es importante **no** usar tildes en los nombres, ya que estas no son reconocidas por estar utilizando formas basadas en el Inglés.

Vamos a empezar **definiendo cinco variables**, llamadas “vida”, “puntaje”, “daño”, “nombre” y “estoyVivo” y les asignaremos :

```
public class Clase1 : MonoBehaviour
{//Llave: Con esto, abrimos y cerramos instrucciones compuestas

   //Variables Globales: Pueden ser usadas por
   //todo el codigo de la clase
   public float vida = 10;//float: Numeros Reales
   private int puntaje = 0;//int: Numero Enteros
   public string nombre = "Jhon Cena"; // string: Texto
   private bool estoyVivo = false;//bool: Solo contiene 2 valores "true" o "false"

   void Start()
   {

      int daño;// Variable Local: Solo puede ser usada
               // dentro de la funcion donde fue creada
      estoyVivo = true;
      puntaje = 0;

   }
```

Presentemos atención, hay varias cuestiones para destacar…

Texto en verde siempre, después de **“//”.** A esto lo llamamos **“comentarios”**. Son **textos** que **no son interpretados** por la computadora, es decir, los ignora completamente. Su rol es el de permitirnos escribir comentarios para realizar **explicaciones** de nuestro código, **ayuda memoria** o hasta **deshabilitar líneas** que no nos interese por ahora que sean ejecutadas.

También vemos que las **cinco variables**, antes de sus nombres, tienen escritas cosas distintas, **“int”**, **“float”**, **“string”**, **“bool”**. Estos son los **tipos de datos** que sirven para **identificar** qué tipo de **información** voy a **guardar** dentro de mi variable.

Notamos que **definimos** cuatro arriba del “Start” y una dentro del mismo. Estas se llaman **variables Globales** y **Locales**, explicadas en la imagen.

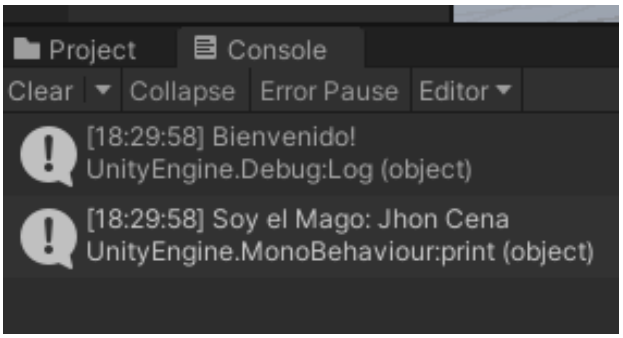
Por último, vemos que les hemos **asignado valores** tanto **cuando** definimos/**creamos** la variable, como cuando la **usamos en** la función ***Start***. Una muestra de que, si lo deseamos, siempre podremos asignar su valor en cualquiera de los 2 momentos, dependiendo lo que pensemos hacer.

La ventana Console.

```
void Start()
{
    Debug.Log("Bienvenido!");
    print("Soy el Mago: " + nombre);
    //A esto se le llama "concatenar", es un proceso que nos permite
    //unir tipos de dato, como un texto prediseñado y una variable
    // o por qué no, varias variables
}
```

Veremos cómo **mostrar** el contenido de una **variable** en Unity, utilizando ***“Print()”*** o ***“Debug.Log()”***, comandos **utilizados** en su mayoría para el “debug”, el procedimiento de **testeo** y la detección de **errores**.

De esta manera podremos mostrar algún dato o mensaje que nos interese en la **ventana Console** (consola) de Unity al ejecutar el juego:

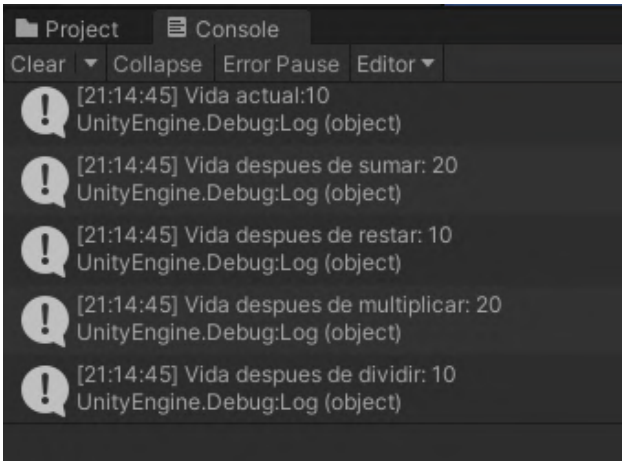


Operadores Aritméticos.

Realizaremos unas cuentas sencillas utilizando **operadores aritméticos**, tales como la **Suma(+)**, **Resta(-)**, **Multiplicación(*)** y **División(/)**

```
void Start()
{
    Debug.Log("Vida actual: " + vida);

    vida += 10;//Sumo 10 a mi variable vida
    Debug.Log("Vida despues de sumar: " + vida);

    vida -= 10;//Resto 10 a mi variable vida
    Debug.Log("Vida despues de restar: " + vida);

    vida *= 2;//Multiplico por 2 a mi variable vida
    Debug.Log("Vida despues de multiplicar: " + vida);

    vida /= 2;//Divido por 2 a mi variable vida
    Debug.Log("Vida despues de dividir: " + vida);
}
```

En la imagen podrán ver que para realizar estas sencillas operaciones nos basta con colocar el **operador** delante del símbolo **“=”**. Esto hará que el **valor** de nuestra **variable** se modifique según la **operación indicada** y el **valor asignado**, dando así, el siguiente resultado:



Conclusión

Durante la clase aprendimos que la organización de tus proyectos y la exploración de nuevas herramientas son esenciales para abordar nuestros videojuegos con Unity.
No solo aprendimos sobre interfaz y elementos visuales, sino que iniciamos el camino en la programación, otro de los pilares esenciales de un videojuego.

Ejercicios prácticos:

¡Bienvenido a Unity!

Si aún no creaste un proyecto;

- **Creá un 2D Project** y utilizando la **"SampleScene"** (recordá cambiarle el nombre) que te proporciona.
- **Creá un objeto** y organizalo dentro de un objeto vacío (Empty Object).
- Para terminar le colocaras un código que sume una variable y muestre el valor en consola.

Empezá a pensar tu **juego**. ¿Qué **datos** usarías? ¿Por qué razones?

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


Buenos Aires
aprende
Agencia de Habilidades para el Futuro
Buenos
Aires
Ciudad