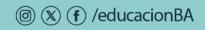
«Talento Tech»

Videojuegos

Clase 01











Clase N° 1 | Conceptos básicos

TEMARIO

- Proyectos y escenas
- Interfaz de Unity.
- GameObjects y figuras primas.
- Primeros pasos con script.







Proyectos y Escenas

Creá un proyecto 3D: En el Unity HUB, seleccioná "New Project" y luego seleccioná "3D Core", elegí un nombre para tu proyecto y después seleccioná "Create Project". img17.gif

Luego de crear nuestro 3D Project

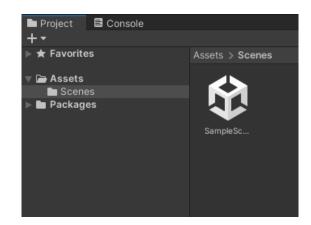
Para arrancar vamos a buscar la pestaña *Project*, que es donde se guardan las carpetas de nuestro proyecto.

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 las cosas que vamos a necesitar para realizar nuestro trabajo.







Una escena es un conjunto de objetos que interactúan según dispongamos gracias a código escrito en C#, 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.



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.

Ahora que sabemos esto, comencemos a recorrer la interfaz de Unity para que podamos ubicar qué hace cada ventana:

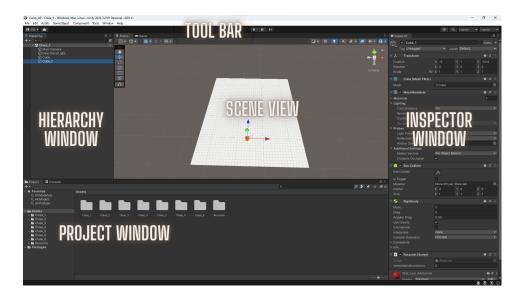
La Interfaz

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**, 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:

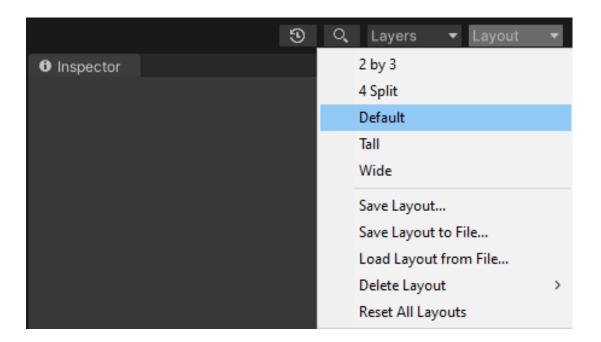


★ Super TIP importantísimo:

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.







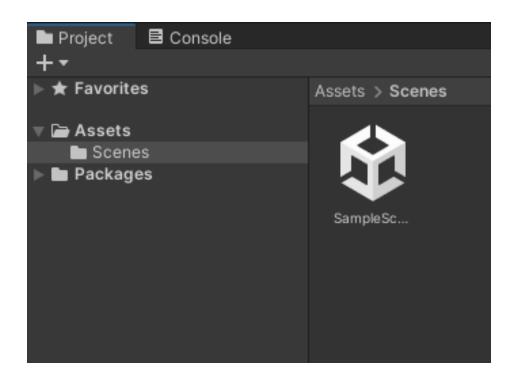
A continuación vamos a empezar a recorrer estas pestañas una por una para que empecemos a familiarizarnos con cada una de ellas.

Project Window (ventana del proyecto)

Como mencionamos al principio de la clase, **muestra sus assets** (modelos 3D, texturas, sonidos, scripts, animaciones, materiales, etc.) de librería que están disponibles para ser usados. Cuando importes alguno a tu proyecto, aparecerá acá.







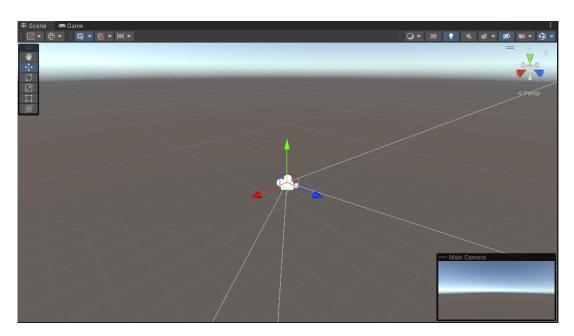
The Scene View (vista de escena)

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.

Esta **ventana** 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 podemos girar la cámara.

Manteniendo la rueda del mouse podemos movernos en los ejes X/Y.

Si mantenemos clic derecho y usamos las teclas WASD, nos podemos mover cómodamente.

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**. Ya 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.

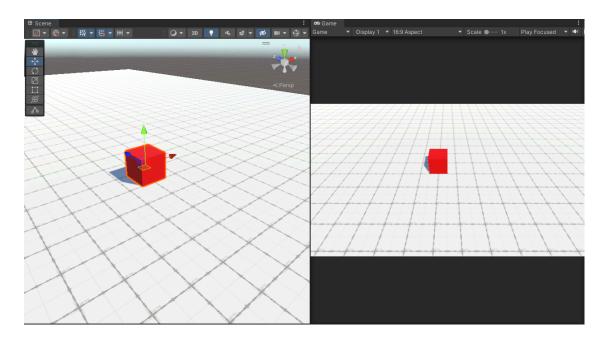
La ventana de Game (o juego)

Esta ventana es el juego en sí, como se va a ver **nuestra escena cuando le damos play**. En general cuando estamos desarrollando un juego, necesitamos probar que todo funcione, así que la vamos a estar utilizando 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 imagen podemos ver cómo separamos las pestañas de escena y de juego para poder trabajar de forma cómoda. Acuérdense 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**.

image18.gif

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.

psyduck.gif

¡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: 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

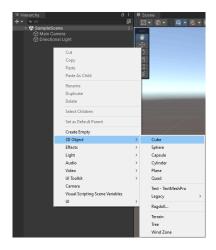
Como crear un objeto

Vamos a **crear** nuestro primer **objeto**, para eso vamos a hacer **clic derecho** sobre la **espacio vacío** de la jerarquía, se desplegará un menú conteniendo distintos objetos que se pueden **colocar dentro de la escena**, seleccionamos **"3D Object"**; nos va a aparecer una





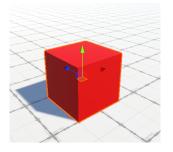
lista de **formas geométricas** en 3D, como un cubo, una esfera, una cápsula, etc. Vamos a seleccionar el **cubo** ("Cube").



Interactuando con objetos

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

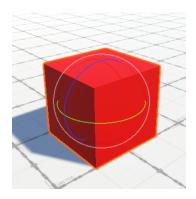
W para mover objetos en uno de sus tres ejes (X,Y,Z)



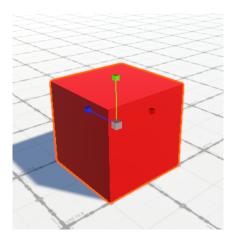




E para rotar objetos en uno de sus tres ejes (X,Y,Z)



R para escalar objetos en uno de sus tres ejes (X,Y,Z)

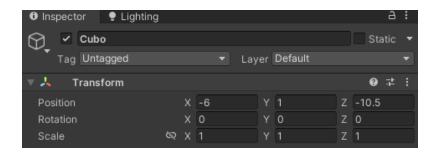


Componente Transform

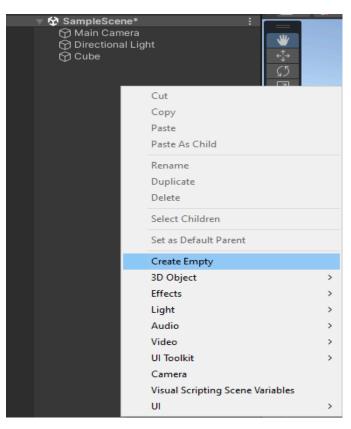
El transform es un componente que tienen todos los objetos del juego, todos los objetos sin ninguna excepción. Transform se encarga de manejar la posición, la rotación y la escala de un objeto, las tres variables que estuvimos manipulando recientemente. En el Inspector vamos a poder ver el valor de estas 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 "3D 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.



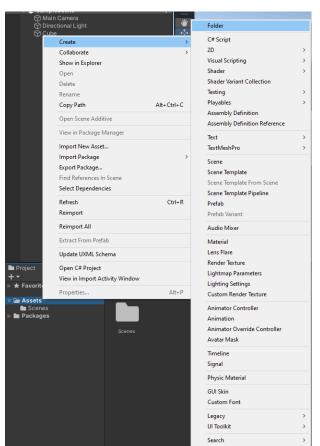
) ; **)**;



Los scripts

Los scripts son lo más importante de todo nuestro proyecto, ya que contienen **códigos** escritos por nosotros 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.

Cómo habíamos mencionado brevemente, 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.



Creando nuestro primer script

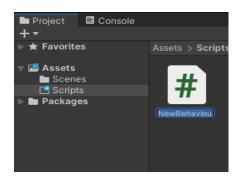
En la pestaña de **Project**, dentro de la carpeta de "**Assets**", vamos a **crear** nuestra propia **carpeta** para ordenar nuestro proyecto, y la llamaremos "**Scripts**". Es muy importante la organización para garantizar un mejor resultado.

Para esto vamos a hacer clic derecho dentro de la pestaña "Project", yendo a

→ "Create" → "Folder".

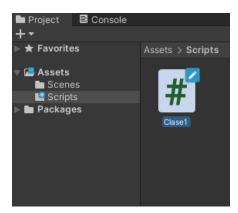






Una vez creada nuestra carpeta, repetiremos el mismo proceso, pero esta vez, seleccionando "C# Script". Una vez creado, es muy importante no tocar nada, ya que antes debemos nombrar nuestro nuevo archivo.

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".

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

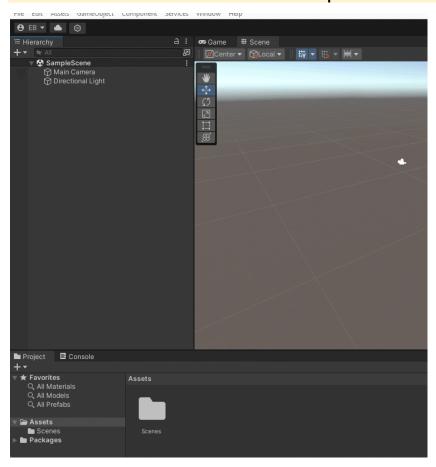
Para poder utilizar correctamente, debemos enlazarlo siguiendo estos pasos:





Vamos a ir a la esquina superior izquierda de la pantalla y haremos clic en Edit →

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



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.

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

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 nos tendremos que conformar aprendiendo sobre variables.

```
public class Clase1 : MonoBehaviour{
    //Start is called before the first frame update
    void Start(){
    }
    //Update is called once per frame
    void Update(){
    }
}
```





Una **variable** es un **espacio reservado** en la memoria de la computadora que almacena datos, y estos, justamente, pueden ser modificados y "variar". A estas 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.

Arrancaremos **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 código de la clase
    public int vida = 10; // int: numero entero
    private float puntaje = 0; // float:numero real
    public string nombre = "Jhon Cena"; // string: texto
    private bool estoyVivo = flase; //bool: solo contiene 2 valores "true" o "false"
    void Start(){
        int daño; //Vaiable local: Solo puede ser usada dentro de la funcion donde fue

creada
        estoyVivo = true;
        puntaje = 0;
    }
}
```

Si prestan atención notarán varias cosas:

Texto en verde siempre, después de "II". 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, justamente, comentarios para realizar explicaciones de nuestro código, ayuda memoria o hasta deshabilitar líneas que no nos interese por ahora que sean ejecutadas.





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.

Que **definimos** cuatro arriba del "Start" y una dentro del mismo. Estas se llaman **variables Globales** y **Locales**, explicadas en la imagen.

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

Por último, mega último, 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**.

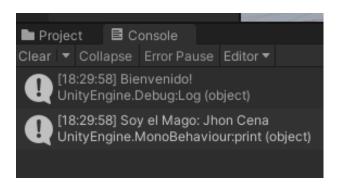
```
void Start(){
    Debug.log("¡Bienvenido!");
    print("Soy el Mago: " + nombre);

// A esto se le llama "concatenar", es un proceso que nos permite unir tipos de dato, como texto y una o varias variables.
}
```

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







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.





Desafío Nº 1:

Si aún no creaste un proyecto; Creá un **3D Project** y utilizando la **"SampleScene"** (recordá cambiarle el nombre) que te proporciona.

Creá un personaje en Unity utilizando figuras primitivas (**3D Objects** simples como cubos, esferas, cilindros) y organizalo jerárquicamente mediante un objeto vacío (Empty Object).

Ajustá las posiciones y escalas de las **figuras primitivas** para lograr la apariencia deseada del personaje.

No es necesario utilizar código en este desafío; enfocate en la manipulación visual dentro del **editor de Unity**, identificando cada ventana y cada herramienta vista en clases.

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

Tomá una **captura de pantalla** del personaje que creaste y su jerarquía **Subilas** al espacio correspondiente del "Desafío 1" en el campus.



