Álvaro Holguera
CIFP José Luis Garci
(Madrid)

UD 01
INTRODUCCIÓN AL MOTOR UNITY3D

Desarrollo de Entornos Interactivos Multidispositivo

Álvaro Holguera
CIFP José Luis Garci
(Madrid)

UNITY3D

Vamos a comenzar a trabajar con el motor de videojuegos Unity 3D

Toda la información sobre Unity la puedes encontrar en la documentación de su página web https://docs.unity3d.com/Manual/index.html

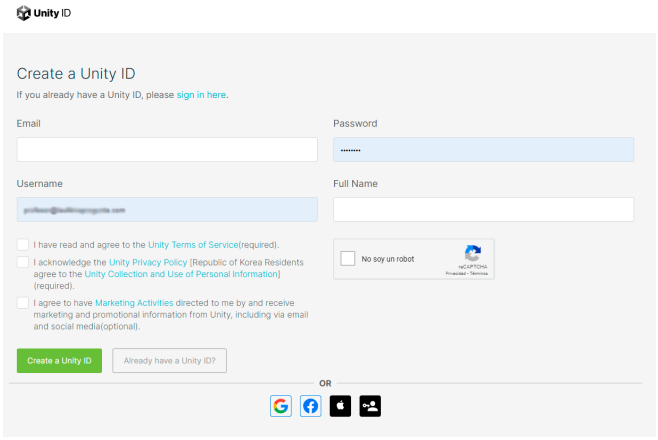
INSTALAR EL PROGRAMA

Unity3D s un software que puede instalarse de forma gratuita con casi todas las prestaciones disponibles siempre que sea para uso personal.

Crear una cuenta en Unity 3D

Antes de instalar el programa, deberemos crear una cuenta para luego poder activar la licencia. Para ello, iremos a la página para crear una ID de Unity: https://id.unity.com/account/new?locale=es

Una vez allí, rellenaremos el formulario:



Creada ya la cuenta, podremos logearnos en la página de Unity: https://unity3d.com/es/user/login?return=es

Unity Hub

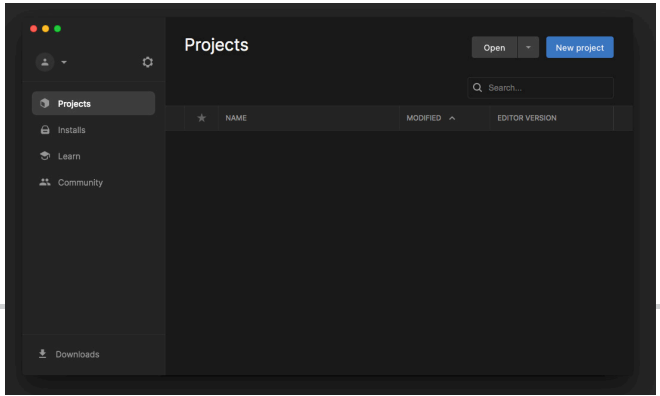
Ahora debemos hacer es instalar el Editor de Unity.

Debido a que existen muchas versiones del editor, y que podemos tener varias instaladas de forma simultánea, usaremos la herramienta "Unty Hub" que nos permite gestionar las instalaciones del editor y los proyectos.

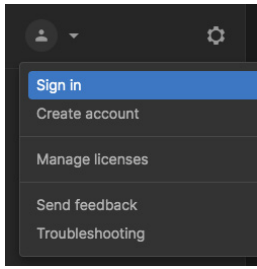
IMPORTANTE: cambiar de versión de Editor en medio de un proyecto puede provocar que ciertos scripts o funcionalidades dejen de funciona. Asegúrate de usar siempre la misma versión del editor de Unity.

Iremos a la página de Unity para descargar e instalar Unity Hub: https://unity3d.com/es/get-unity/download

Una vez instalado, al abrirlo veremos una ventana donde se mostrarán los proyecto abiertos recientemente.



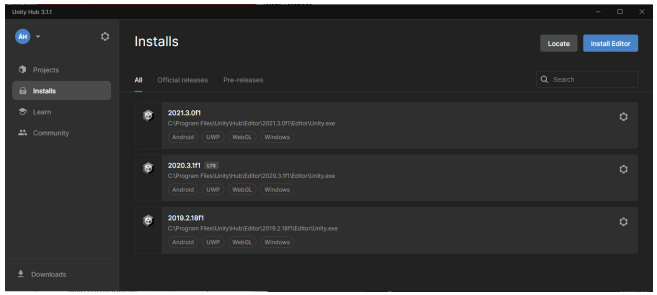
Antes de seguir, nos loguearemos haciendo click en el icono de usuario que aparece arriba a la izquierda (nos abrirá el navegador web donde deberemos iniciar sesión si no lo hemos hecho):



Instalar el editor

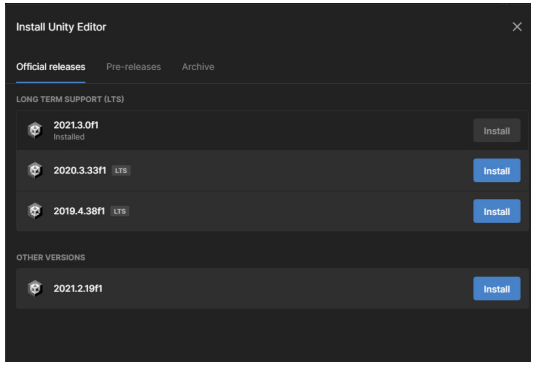
Para poder usar Unity, debemos instalar al menos una versión del editor. Para ello, iremos a la pestaña de "Installs"

deberemos elegir las versiones del editor que queremos tener instaladas en nuestro equipo. Para ello iremos a la pestaña de "Installs". En ella se mostrarán todas las versiones instaladas. En esta imagen por ejemplo, se muestran varias versiones:

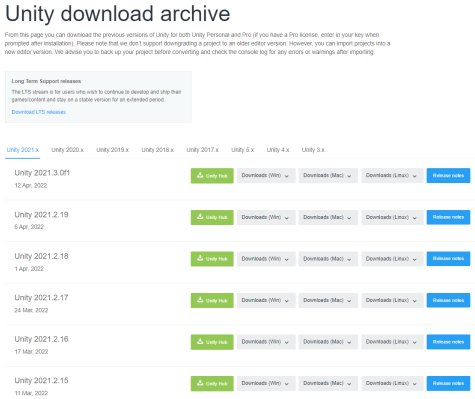


INSTALAR EL PROGRAMA (II)

Para instalar una nueva versión, haremos click en el botón de “Install Editor”. Nos abrirá una ventana donde se muestran las versiones oficiales más recientes:

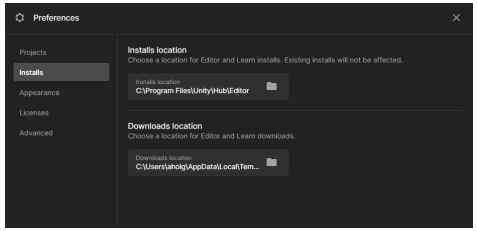


Si necesitamos instalar versiones anteriores, podemos ir a la pestaña de “Archive”, que nos redirigirá a la página de Unity con el historial de versiones, organizadas por años. Podemos descargar el ejecutable y lanzarlo:



Antes de instalar, nos preguntará qué paquetes queremos instalar, que dependerá en gran medida de las plataformas a las que queremos dirigir nuestros videojuegos. Más tarde podemos añadirlas en la pestaña de Installs, donde apare cada versión con los paquetes instalados, en la rueda dentada de cada instalación en la opción “Add modules”.

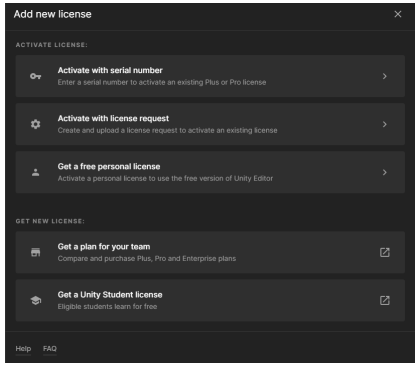
Si necesitas conocer o modificar la carpeta en la que se descargan y se instalan los editores, puedes hacerlo pulsando en la rueda dentada de preferencias, en la pestaña “Installs”:



Activar la licencia

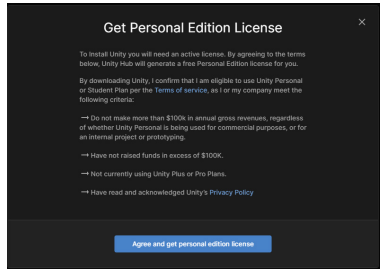
Una vez instalado el editor, es importante activar la licencia para poder usarlo con todas las funcionalidades.

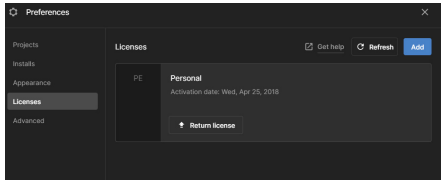
Si nos vamos a “Preferencias>Licenses” podremos añadir una nueva (siempre que hayamos iniciado una sesión):



Si queremos usar Unity para uso personal y sin ánimo de lucro, podemos activarlo en la opción de “Get a free personal license”.

Una vez que hayamos aceptado las condiciones, nos aparecerá an la pestaña de licencias de las preferencias





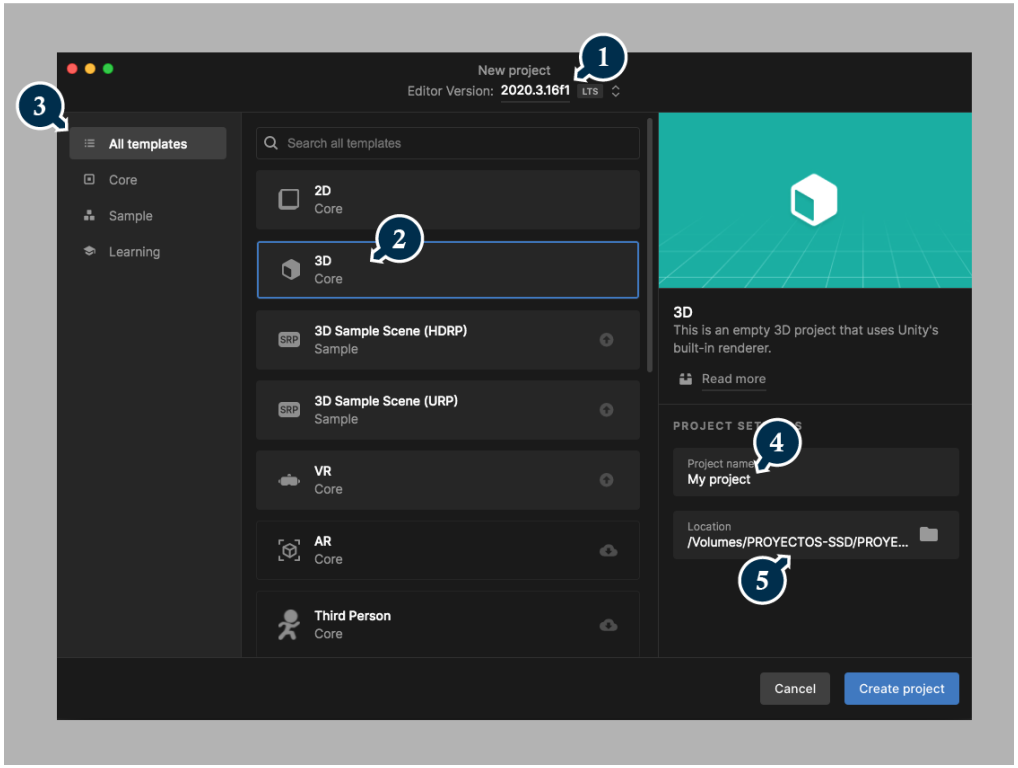
Otras herramientas útiles de Unity Hub son:

1. **Learn**: acceso a los vídeos y tutoriales de la página de https://learn.unity.com/

2. **Comunity**: acceso a blogs, foros, y más.

CREAR UN PROYECTO

Vamos a crear nuestro primer proyecto. Para ello pulsaremos en el botón de "New Project" que aparece en la ventana principal de "Projects". En la ventana emergente, veremos varias opciones:



- La versión del editor que usaremos para ese proyecto (1). Muy importante elegir la correcta y no cambiar de versión salvo que sea estrictamente necesario.
- Disponemos de varias plantillas de proyecto. Las analizaremos en detalle más adelante, de momento eligiremos la plantilla de "3D" básica -o core- (2).

A la izquierda podemos filtrar las plantillas que se muestran (3)

- Asignaremos un nombre al proyecto (4). Es importante elegir un nombre descriptivo pero sin caracteres extraños, ya que se creará una carpeta con ese nombre.
- También debemos indicar dónde se guardará el proyecto en nuestro disco duro (5). Recuerda que en el menú de preferencias, en la carpeta de "Projects", podemos cambiar la ubicación por defecto. NUNCA olvides dónde has guardado tu proyecto.

Puedes cambiar la ruta por defecto que se usa para crear los proyectos.

NOTA: la carpeta que creará con el nombre indicado contendrá todo nuestro proyecto. Si queremos abrirlo en otro ordenador, solo tenemos que copiar esa carpeta y en Unity Hub en lugar de crear un proyecto pulsaremos en "Open" y le indicaremos dónde está la carpeta del proyecto. Consejo: la carpeta "Library" suele pesar mucho, y no es necesario copiarla ya que Unity la regenera automáticamente al abrir el proyecto.

Al crear el proyecto, o abrir un proyecto creado, se lanzará el editor de Unity.

Álvaro Holguera
CIFP José Luis Garci
(Madrid)

LA INTERFAZ

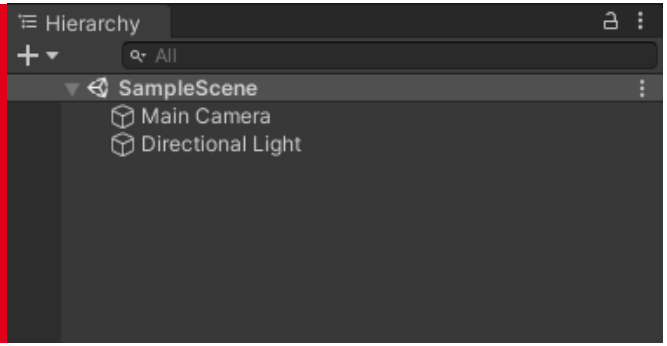
1. Jerarquía
2. Escena
3. Inspector
4. Proyecto

Veremos otros dos paneles que no aparecen aquí: la de **juego** y la de **consola**.

RECUERDA: puedes cambiar de posición los paneles, así como de tamaño, cerrarlos, maximizarlos, pulsando con el botón derecho (RMB) en la pestaña del panel (si los arrastras puedes hacerlos flotantes). Y también puedes crear tus propios espacios de trabajo en "Window > Layout > Save Layout"

1. PANEL DE JERARQUÍA (Hierarchy)

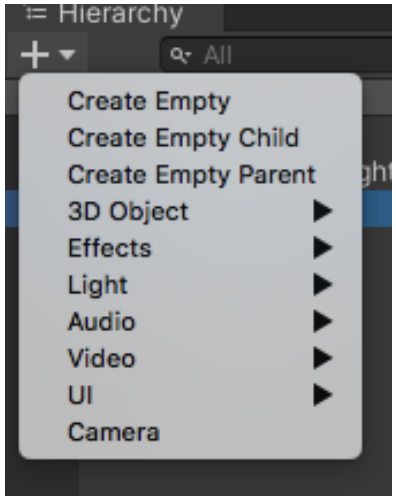
En este panel irán apareciendo los elementos que hay en nuestra escena.



Veremos que Unity ha creado una escena por defecto ("SampleScene"), de la que dependerán de forma jerárquica los elementos que vayamos incorporando.

Por defecto ha puesto una cámara y una luz.

Podemos añadir elementos a la escena mediante el icono de "+" que aparece arriba, o arrastrándolos directamente a la venta de la escena o e el menú de "Game Object".



PRACTICA: añade un elemento a la jerarquía para ver cómo aparece en la escena, por ejemplo un objeto 3D.

Algo muy importante que debemos tener en cuenta en este panel, es que los elementos se organizan de forma jerárquica. Si arrastramos un elemento encima de otro, se convertirá en "hijo" suyo y dependerá de él (si se mueve, rota o se escala el padre, los hijos también).

Cuanto tenemos muchos elementos en la escena, es recomendable organizarlos por ejemplo creando objetos vacíos (Empty) que ejercerán de padres de todos.

También contamos con una herramienta de búsqueda que nos permite localizar los elementos de la escena de forma rápida cuando hay muchos.

Al pasar por encima de un elemento, aparecerán dos botones que nos permiten bloquearlos o esconderlos en la escena.

NOTA: si hacemos doble click en un elemento, se centrará en la vista de la escena (lo mismo que si pulsamos "F" con el ratón encima del panel de la escena.

Una opción muy útil es crear grupos de selección. Para ello, seleccionamos los que queramos y en Edit > Selection guardamos ese grupo, para recuperarlo más tarde. También podemos crearlo pulsando Ctrl. + Alt. + "1/2/3…" y recuperarlos mediante Ctrl. + Shift. + Alt. + "1/2/3".

2 PANEL DE ESCENA (Scene)

Entorno 3D que muestra los elementos de la escena.

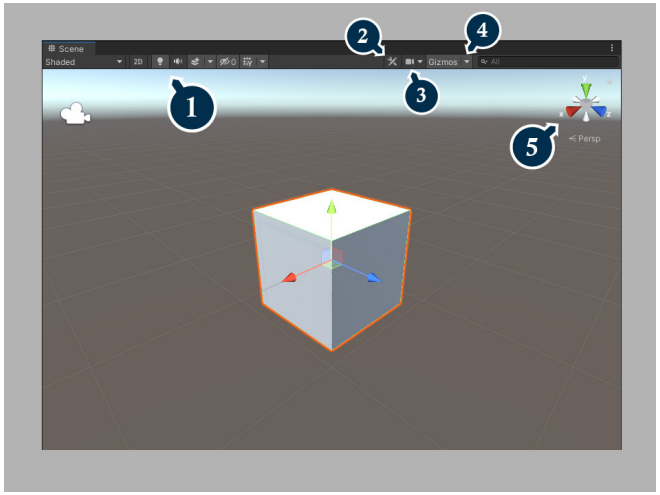
Para movernos entre ellos, deberemos usar los atajos de teclado y ratón:

- **Alt + Botón izquierdo** de ratón: la vista pivota en torno al objeto que estamos mirando.

Si pulsamos la tecla "f" cuando el puntero está sobre el panel de la escena, la vista centra el objeto que tengamos seleccionado en la jerarquía.

- **Botón derecho y arrastrar**: la vista pivota sobre la posición del observador.
- **Botón central del ratón y arrastrar:** nos desplazamos por la escena (también pulsando Ctrl. + Alt.)
- **Rueda del ratón**: se hace Zoom a la escena. También se puede hacer con más precisión pulsando Alt. + Botón Derecho.
- Con los **cursores** podemos desplazar izquierda y derecha, y hacer zoom hacia adelante y hacia atrás.

Veamos algunas de las funciones adicionales del panel:



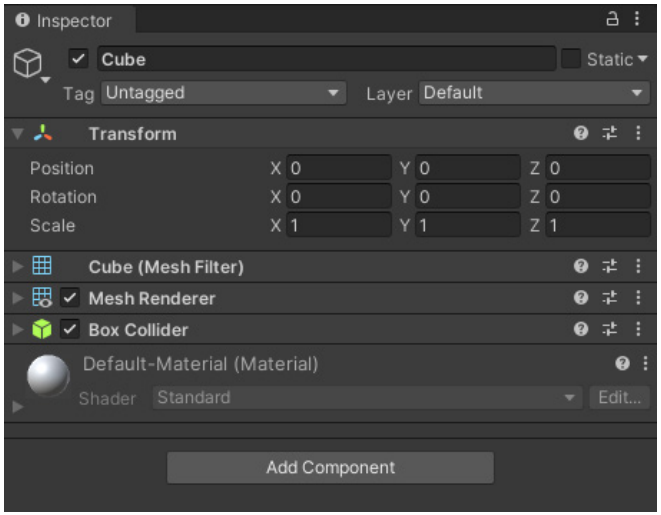
- **Opciones de visualización** (1): nos permite seleccionar cómo queremos ver los objetos, conmutar a un entorno 2D (útil para juegos en 2D), activar y desactivar sonidos, luces o efectos atmosféricos. Mostrar los elementos ocultos y finalmente mostrar u ocultar la malla.
- **Activar panel de herramientas contextual** (2). Dependerá del elemento que tengamos seleccionado en la escena.
- **Configuración de la cámara de la vista de la escena** (3). Es independiente de la cámara del juego. Permite ajustar por ejemplo el ángulo de visión.
- **Gizmos** (4): son los ayudantes en la escena, que no tienen presencia física pero que son útiles (por ejemplo, el icono que indica dónde está la cámara). Podemos ocultarlos todos o, mediante el desplegable, decidir cuáles mostrar y cuáles ocultar.
- **Configurador de la perspectiva** (5). Además de ayudarnos a orientarnos (recuerda, el verde es hacia arriba, el rojo hacia la derecha y el azul hacia el fondo), si haemos click en cualquiera de los conos que indican el eje la vista se alineará con ese eje. Por otro lado, si hacemos click en el cubo del centro, o en el texto "Persp" cambiaremos a una vista isométrica (muy útil para alinear objetos).

NOTA: cada escena tiene su propio entorno, con sus propios elementos. Podemos crear tantas escenas como queramos, en File > New Scene (o pulsando Ctrl. + N). Nada más crearla deberemos guardarla (un asterisco en el nombre de la escena en el panel de jerarquía nos indica que está sin guardar)...

3&4 INSPECTOR & PROYECTO

Panel de Inspector

Muestra las propiedades y los componentes del objeto que tengamos seleccionado.

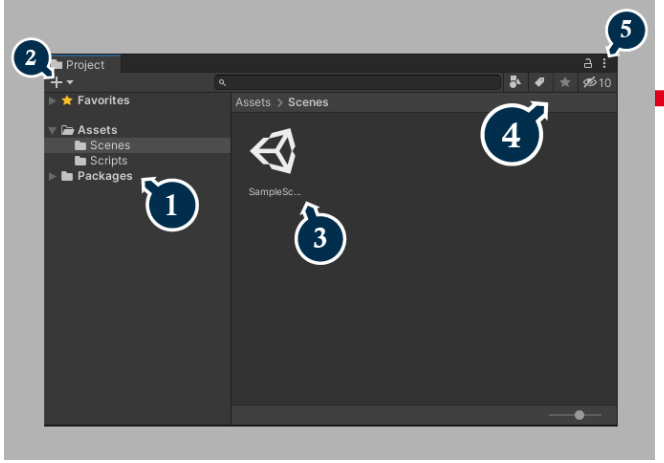


En la parte superior, podemos cambiar al nombre del elemento y desactivarlo por completo mediante la casilla de verificación que hay junto al nombre.

El resto de los elementos los veremos en el siguiente punto cuando hablemos de los "Game Objects".

Panel de Proyecto

En este panel podremos organizar todos los materiales que vamos a usar en las diferentes escenas.



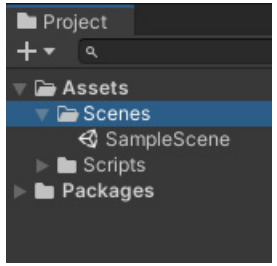
• En la columna izquierda veremos la estructura de carpetas (1). Todo lo que vayaos incorporando se añadirá a la carpeta "Assets". Podemos crear subcarpetas pulsando con el Botón Derechó del ratón y pulsar "Create > Folder". Es esencial llevar un correcto orden de todo el material.

• Podemos añadir elementos a nuestro proyecto usando el icono de "+" (2), o pulsando con el Botón Derecho y "Create". Como veremos, los elementos que se pueden añadir alproyecto no son los mismos que a la escena (si queremos que un elemento de la escena aparezca en el proyecto para usarlo en otras escenas, deberemos crear un "prefab", del que hablaremos más tarde)

• Al seleccionar una carpeta, veremos su contenido en la columna derecha (3).

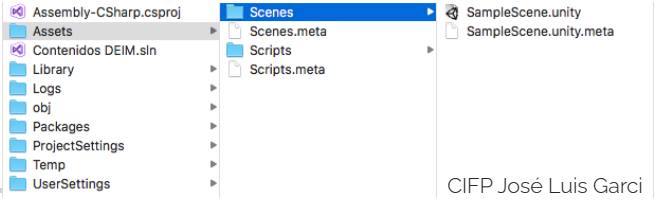
• Tenemos herramientas que nos permiten buscar elementos por su nombre, su tipo, su etiqueta o incluso crear carpetas de favoritos para organizarlo mejor (4)

• En el menú de hamburguesa que tenemos arriba a la derecha (5), podemos hacer que este panel se muestre con una columna solo, lo que puede ser en ocasiones más cómodo, tal y como se muestra en esta imagen.



CONSEJO: Si pulsas "Alt" al hacer click en el triángulo que despliega la carpeta, se despliegan todas las subcarpetas.

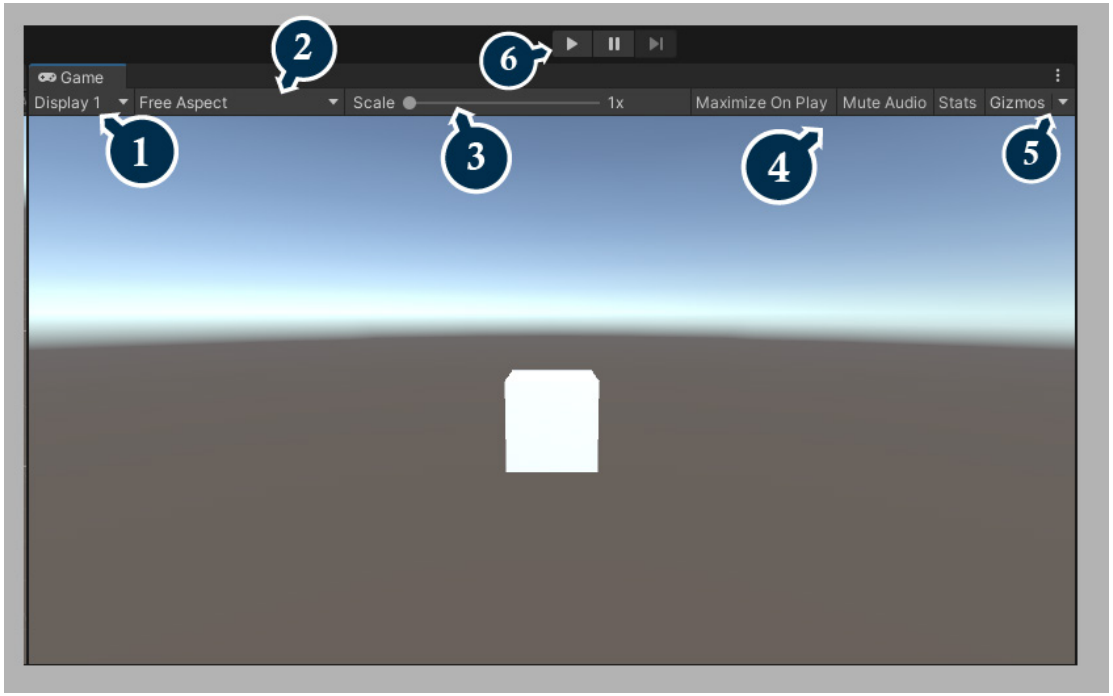
IMPORTANTE: todo lo que muestra el panel de proyecto en la carpeta "Assets", se corresponde con la carpeta "Assets" que encontrarás en la carpeta creada por Unity Hub con el nombre y en la ubicación indicados. Si añades un archivo ahí, aparecerá en el proyecto automáticamente, y si borras un elemento, se borrará también de la carpeta (sin posibilidad de deshacer la acción).



PANEL DE JUEGO

Muestra el resultado final del juego. Veamos algunas de sus funciones:

• El juego muestra lo que capta la cámara de la escena (si la seleccionamos en la jerarquía, nos muestra en una ventana flotante una previsualización). Pero como podemos tener varias cámaras en una escena, cada una la podemos dirigir a un "Display" distinto, y en el panel de juego elegir el "Display" que queramos ver (1)



• La proporción -ó "Aspect Ratio"- de nuestro juego, así como las dimensiones de la pantalla, son datos importantes que debemos conocer de antemano, para seleccionar la previsualización correcta (2), por ejemplo ajustando una proporción fija o incluso un tamaño en píxeles fijos. Por defecto viene seleccionado "Free Aspect" donde vemos todo, independientemente del tamaño del panel de juego. Es útil mientras componemos la escena, pero no es lo que verá el jugador. Más adelante veremos otras opciones.

•Con la rueda del ratón podemos hacer zoom en el panel de juego (3), pero no es recomendable, ya que la imagen se pixela (no es lo mismo que acercar la cámara)

• Tenemos varias opciones de visualización (4): maximizar al lanzar el juego (muy útil cuando estamos testeando el resultado final), mutear el audio y mostrar las estadísticas (las veremos en el tema de optimización del juego)

• Finalmente, igual que en la escena, podemos elegir ver o no los Gizmos (5). En el juego por defecto están desactivados, pero en ocasiones nos interesa verlos mientras jugamos.

En la parte superior (6), tenemos los botones para "lanzar" el juego: Play / Pause / Step (nos permite ir fotorama a fotograma con el juego pausado).

Cuando pulsamos en play (o "Ctrl. + P), el panel de "Game" pasa a primer plano, aunque puedes seguir haciendo cambios en la escena. Quizás te interese reorganizar los paneles para tener los dos visibles de forma simultánea.

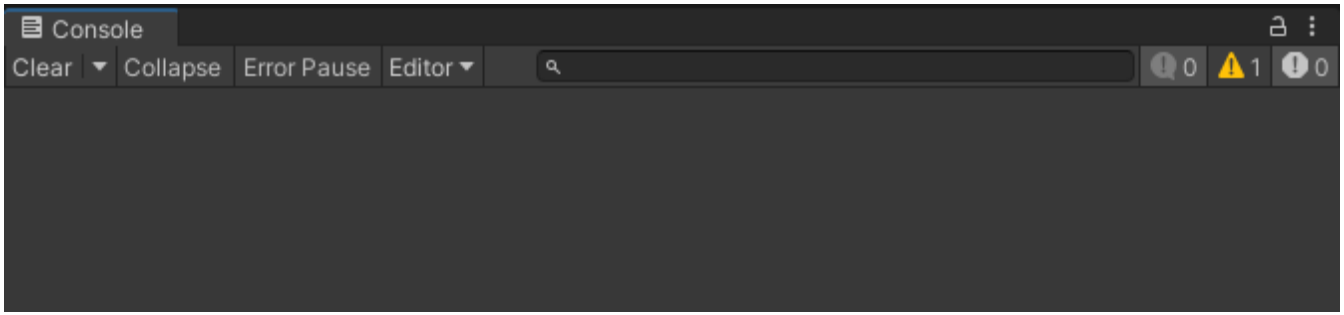
IMPORTANTE: los cambios que se realicen mientras el juego está en ejecución, no se guardan y desaparecen al parar el juego. Por eso, aparecen los paneles tintados. Puedes aumentar la saturación o cambiar el color de ese tinte en "Edit > Preferences > Colors > Playmode tint"

PANEL DE CONSOLA

Una de las herramientas más importantes que tiene un programador para "comunicarse" con su programa, es la consola. En ella, no solo aparecen los mensajes de error o avisos del programa, sino que puede incluir sus propios mensajes en los scripts, ayudando así en el desarrollo.

Por ello, es conveniente (cuando no necesario), tener siempre visible la consola mientras se desarrolla un programa.

Más adelante aprenderemos a mandar mensajes a la consola, de momento veamos las opciones del panel:



• En la opción de "Clear" podemos indicarle que se borren los mensajes cuando se da al Play (muy recomendable para ir limpiando los mensajes antiguos), o cuando se compila el juego

• La opción de Collapse permite que si se repite mucho un mensaje solo aparece uno, indicando cuántas veces se ha repetido.

• "Error Pause" hace que el juego se detenga si aparece un error en la consola.

• En las opciones de la derecha, podemos desactivar ciertos tipos de errores. De izquierda a derecha, son los errores menos graves (por ejemplo mensajes nuestros a consola, que aparecerán con un icono blanco), las advertencias (amarillo) y los mensajes graves (aparecerán con un icono rojo)

Si Unity detecta un error grave, no permitirá ejecutar el juego.

NOTA: Si aparece un error que hace referencia a una línea de código, podemos hacer doble click sobre él y nos llevará al script y a esa línea.

AÑADIENDO OBJETOS

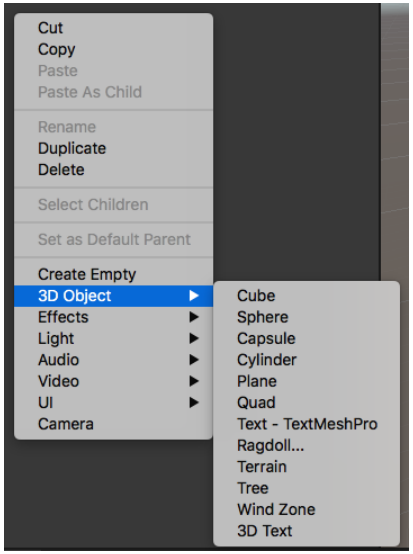
Los elementos que incorporamos a nuestra escena se llaman "Game Objects", y vamos a ver cómo manejarlos.

Primitivas

Aunque lo normal es importar a Unity elementos elaborados en otros programas (modelos, animaciones, materiales y texturas, terrenos, etc.), Unity nos permite crear una serie de objetos 3D sencillos, así como otros elementos, disponibles en el menú "Game Object", así como pinchando con el botón derecho del ratón en la ventana de Jerarquía o en el icono "+" de ese mismo panel.

En esta imagen podemos ver cómo añadir un objeto 3D desde la ventana de jerarquía.

NOTA: si al hacer click con el botón derecho lo hacemos sobre un Game Object existente, el nuevo se creará como hijo de aquel.



Podemos crear:

• Objetos vacíos. Muy útiles para organizar, o para añadir scripts a la escena.

• Objetos 3D: cubos, esferas, cápsulas, planos o incluso terrenos y árboles

Más adelante instalaremos un paquete que nos permitirá añadir Objetos 2D, como sprites o mallas.

• Efectos, como sistemas de partículas que veremos

• Luces

• Audio y vídeo

• Elementos de Interfaz de Usuario (UI) como botones, textos, etc.

• Cámaras.

Al crear un Game Object, deberemos primero darle un nombre. Es importante nombrar y organizar correctamente los elementos de la escena.

MODIFICANDO OBJETOS

Una vez añadido cualquier elemento a la escena, podremos actuar sobre él de varias formas.

En el panel superior, veremos las diferentes opciones con sus correspondientes atajos:

• **Mover escena** (Q): nos permite desplazar la visión de la escena (también pulsando la rueda central del ratón)

• **Desplazar objeto** (W): lo podemos hacer en un eje o en un plano concreto, su hacemos click y arrastramos en las flechas o en los lados central del cubo que aparece

• **Rotar** (E): aparecerán los 4 sentidos de rotación.

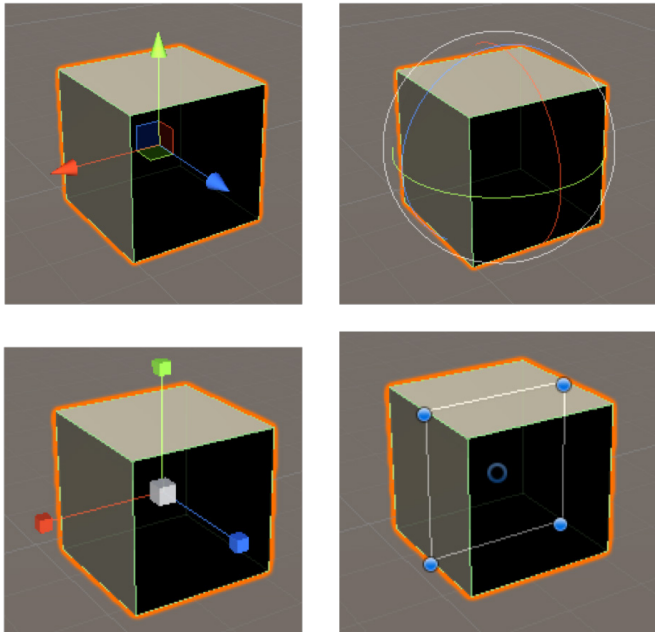
• **Escalar** (R): si lo hacemos en un eje se deformará, pero si lo hacemos pulsando en el cubo central lo hará de forma uniforme.

• "**Rect Tool**" ó deformación planar (T): permite mover y escalar un objeto usando unos ayudantes. Es muy útil cuando trabajamos con objetos 2D, como sprites o textos.

• "**Smart tool**": permite hacer todas las modificaciones a la vez.

• El "**Editor Tool**" nos permite acceder a herramientas específicas del elemento que tenemos seleccionado (igual que la ventana de herramientas de la Escena).

Al seleccionar cualquier objeto, nos mostrará los ejes a través de los cuales podemos mover, rotar o escalar:



NOTA: Si pulsamos "Ctrl." al mismo tiempo que movemos, rotamos o escalamos, se hará en unidades fijas. En Edit-> Snap Settings podemos configurar esas cantidades.

IMPORTANTE: aunque las medidas en Unity son arbitrarias, se acepta por convención que se correponden con metros. Es decir, que un cubo que tenga 1x1x1, entenderemos que tiene 1 metro de ancho, 1 metro de alto y 1 metro de largo. Podemos usar la malla de la escena como referencia, ya que por defecto cada cuadro de la retícula es de 1 unidad.

Punto de pivote y punto de referencia

A la derecha de las herramientas de modificación, encontraremos dos botones más:

• **Pivot/Center**: cuando movemos, rotamos y escalamos un objeto lo podemos hacer en base a su punto de pivote, o a su centro de gravedad. IMPORTANTE: en Unity no podemos cambiar el punto de pivote de un elemento 3D, propio o importado. Para hacerlo debemos ir a programas externos

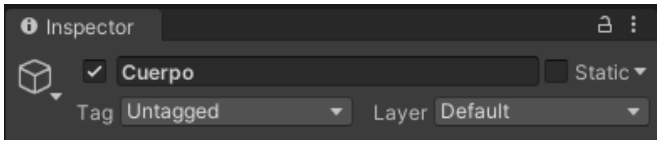
• **Global/Local**: nos permite alternar entre los ejes del escenario, o los del objeto. Por ejemplo, si un objeto está rotado, el eje Z en Global mirará paralelo al suelo, pero en Local mirará hacia adelante del objeto, reproduciendo su inclinación, lo que facilita moverlo en esa dirección.

NOTA: si queremos duplicar un elemento, podemos hacerlo pulsando Ctrl. + D con ese objeto seleccionado.

INSPECTOR Y COMPONENTES

Vamos a analizar en detalle ahora el panel de Inspector, y los elementos que allí aparecen asociados al objeto seleccionado en la escena, llamados "componentes".

Antes de ver esos componentes, veamos la parte superior del panel de "Inspector":



• Casilla de verificación. Si la desactivamos, el objeto dejará de tener presencia en la escena.

• Nombre del Game Object. Podemos también cambiarlo en el panel de Jerarquía, haciendo un click prolongado.

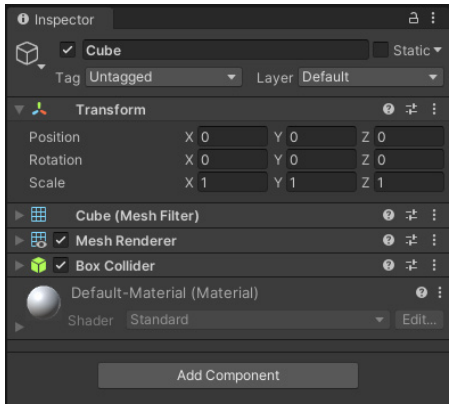
• "Static". Nos permite indicar los objetos que no tendrán movimiento. Lo veremos en detalle cuando veamos iluminación y "Nav Mesh"

• Tag: podemos asignar una "etiqueta" de la lista disponible o crear nuevas. Más adelante veremos que es útil agrupar elementos con la misma etiqueta

• Layer: también podemos agrupar elementos en Layer, útil para evitar colisiones entre capas o para crear profundidad en elementos 2D, entre otras cosas.

Los Componentes

Los componentes son añadidos que incorporan nuevas funcionalidades al Game Object. Además del componente transform, común a todos, hay muchos más componentes. Algunos de ellos vienen por defecto al crear una primitiva en el escenario, como los que tiene un cubo:



Podemos ver que tiene un Mesh Renderer, que permite que la cámara lo "vea", ó un Box Collider para detectar colisiones con otros objetos.

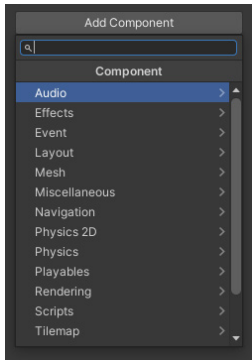
Veremos que casi todos los componentes tienen una casilla de verificación, que permite desactivarlo sin necesidad de borrarlo. Otras opciones que aparecen:

• Botón que nos lleva a la página de documentación de Unity que explica el componente

• Presets: nos permite crear configuraciones preestablecidas de un componente y recuperarlas luego

• Menú de hamburguesa, para borrarlo, moverlo, editarlo, copiarlo, buscar qué elementos lo usan en la escena, etc.

Para añadir un nuevo componente, usaremos el botón de "Add Component". Aparecerá una lista ordenada por categorías, y un campo de búsqueda para agilizar la selección.



Cada componente tendrá sus propios parámetros, como veremos más adelante. Y no todos los objetos soportan todos los componentes.. Iremos viendo más.

NOTA: más adelante descubriremos algo útil, que todos los parámetros de los componentes de nuestro Game Object pueden ser leídos y modificados desde código.

EL COMPONENTE TRANSFORM

Todos los elementos de la escena lo tienen, incluso los “Empty Objects”, ya que determina la posición en el espacio 3D.

Tiene 3 elementos básicos: posición, rotación y escala, cada uno de ellos modificable en los 3 ejes de coordenadas. Permite hacer modificaciones más precisas que con las herramientas vistas antes.

Podemos introducir los valores manualmente, o si nos ponemos encima de la letra del eje, hacer click y arrastrar izquierda y derecha.

RECUERDA: si rotas, escalas o mueves un elemento, se trasladará a sus hijos.

IMPORTANTE: las coordenadas de posición de un objeto son siempre respecto al padre. Si no tiene, es respecto al escenario, pero si es hijo de otro, lo son respecto de ese. Por ejemplo, si está en la posición 0,0,0 se centrará en su objeto padre, esté donde esté del escenario.

TRUCO: Hay dos técnicas útiles que te ayudarán a posicionar un objeto o la cámara:

• Game Object ->Move to View (Ctrl. + Alt. + F): El objeto se seleccionado e ubicará en la posición hacia la que mira la vista.

• Game Object -> Align with View (Ctrl. + Shift. + F) Movemos al objeto para que esté en la posición en la que está la vista de la escena (muy útil para luces y cámaras, para ubicarlas hacia donde estamos mirando en ese momento)

• Game Object > Align View to Selected: moverá la vista de la escena para mirar hacia donde mira el objeto seleccionado.

CREANDO PREFABS

Un prefab es un tipo específico de Game Object que se crea en el proyecto -no depende de una escena-, con sus propios componentes y propiedades, y que puede ser reutilizado en cualquier momento (instanciado).

Cualquier elemento que creemos directamente en nuestro escenario puede ser añadido como prefab, si lo arrastramos al panel de proyecto desde el panel de la jerarquía (cambiará de icono a un cubo que indica que es un prefab).

NOTA: una vez añadido como prefab, lo habitual es borrar el original del escenario, ya que la idea es instanciarlo mediante código.

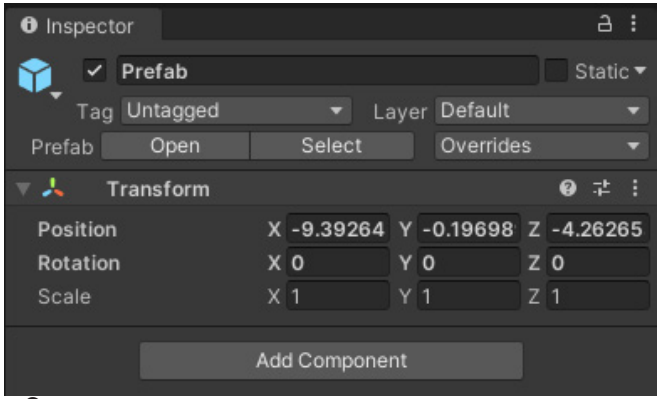
También se puede crear un prefab desde cero, añadiendo en el proyecto un prefab vacío, mediante click derecho del ratón y "Create" o usando el icono de "+", y una vez creado podemos abrirlo y construir lo que queramos.

Podemos modificar el prefab directamente abriéndolo con un doble click en el proyecto, o pulsando en el botón "Open Prefab" de la ventana inspector . Se abrirá una nueva ventana en la escena. En la parte superior de nuestra escena veremos que estamos dentro de un prefab, y para volver a la escena tenemos que hacer click sobre el nombre de la escena.

Operaciones específicas con un prefab

La gran ventaja, es que cualquier modificación que hagamos sobre él se trasladará a todas las instancias.

Si lo seleccionamos, veremos que su ventana de inspector es ligeramente diferente. Veremos las funciones específicas:



1. Como podemos ver, el icono del objeto es un cubo azul, indicando que es un prefab

2. El botón de "Open" lo abrirá en la ventana de la escena para que podamos añadir cambios. Recuerda que cualquier cambio que hagamos en el Prefab original se aplicará a las instancias que haya en cualquier escena

3. El botón de "Select" nos permite buscarlo en el proyecto y seleccionarlo.

4. Overrides: cuando realizamos cambios en un prefab de la escena, es decir en su instancia, estos cambios no se aplican al prefab original. Si desplegamos este menú, aparecerán los cambios realizados y nos da la opción de aplicarlos al original, de forma individual o todos ("Aply all"), o incluso revertirlos.

IMPORTANTE: recuerda que si haces modificaciones en una instancia de un Prefab, SOLO estás modificando ese en esa escena. El resto seguirá sin cambios a menos que apliques los cambios mediante "Overrides".

Podemos duplicar un prefab pulsando con el botón derecho y pulsando en "Create->Prefab Variant"

También podemos convertir un prefab en un elemento normal en la escena, pulsando botón derecho en la jerarquía > Prefab > Unpack.

EJERCICIO

Usando lo que hemos aprendido, vamos a crear un objeto en nuestra escena que combine diferentes primitivas, debidamente jerarquizadas y transformadas. Posteriormente, crearemos un prefab con ese objeto

Instrucciones

1. Crea un proyecto en Unity llamado “Zaxxon”.

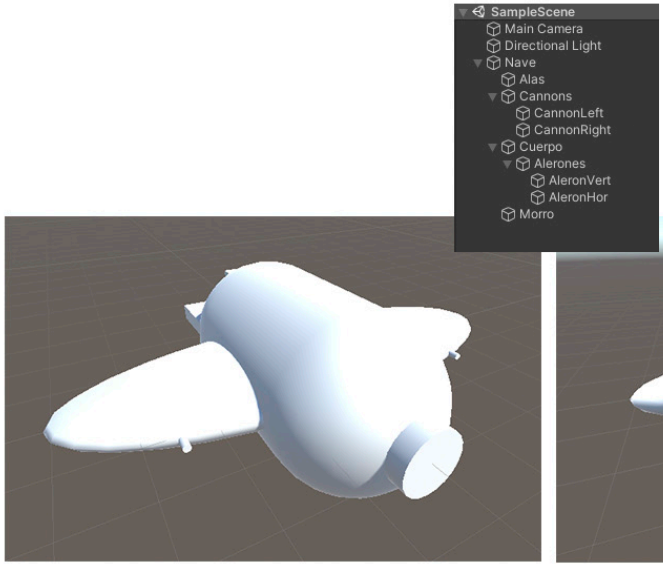
2. En la escena que viene por defecto creada, usa primitivas modificadas para crear el modelo (cubos, esferas, cilindros y una cápsula para el morro). Asegúrate de que la envergadura de la nave es de 4 metros de una punta del ala a la otra (recuerda que alternar con la vista isométrica es muy práctico para hacer mediciones “a ojo”), y respeta los ejes para que luego te sea más fácil moverlo.

3. Utiliza las herramientas vistas: atajos, modificaciones exactas o con valores enteros, duplicar, etc.

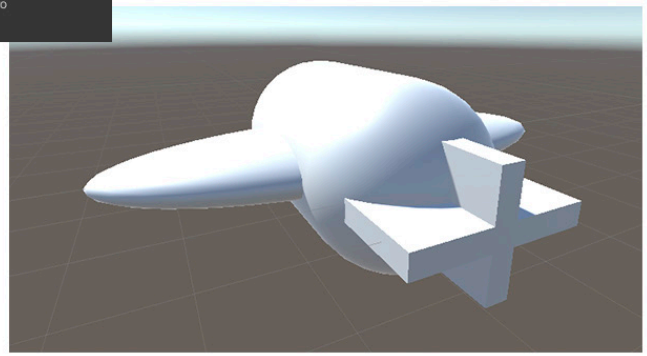
4. En la jerarquía, organízalo para que todos los elementos de la nave dependan del cuerpo (el cubo), y a su vez, que los cañones dependan de un Empty Object llamado “cannons”.

5. Crea un prefab, ábrelo y añade dentro la nave creada. También puedes crear un Empty Object que haga de padre de todos, nombrarlo y arrastrarlo al proyecto, pero si haces eso asegúrate de que está en las coordenadas 0,0,0

IMPORTANE: organiza y nombre correctamente todos los elementos, tanto en el proyecto como en la jerarquía. Y ten en cuenta que la cara frontal del objeto es la que mira hacia el eje Z, en profunidad.



Parte delantera



Parte trasera

AÑADIENDO SCRIPTS

Uno de los componentes que más usaremos y que más posibilidades nos brinda, es el de los scripts, los cuales contedrán nuestro código.

Para llevar a cabo cualquier juego o aplicación es necesario recurrir a la programación para dotar de funcionalidades complejas y controlar la interactividad de nuestros elementos.

Creando nuestro primer script

Un script es un archivo de texto con extensión .cs (de c#) y que se añade como un componente más a los Game Objects de nuestra escena. Veamos los pasos para crearlo:

1. Seleccionamos el objeto de la escena que queremos que contenga el script, y en el panel de Inspector pulsamos el botón de "Add Component"

2. Buscamos un componente de script c# y lo añadimos

3. Le damos un nombre adecuado. Es importante que sea descriptivo, no muy largo, que no contenga caracteres extraños, ni espacios en blanco, ni que empiece con números. Una vez dado un nombre, no se debe cambiar ya que queda asociado a la clase.

4. Se creará un archivo .cs en la carpeta del proyecto, al que podremos acudir con un programa externo. Lo ideal es moverlo a una carpeta específica donde guardemos nuestros scripts.

5. Otra opción es crear el script en la carpeta directamente, pulsando el botón derecho del ratón en la carpeta y Create > C# Script, y posteriormente arrastrarlo al panel de inspector del Game Object o directamente al objeto en la jerarquía. Quedará añadido como componente.

6. Lo podemos abrir desde Unity, con un doble click en la ventana de proyeccto o en el menú de hamburguesa del componente -> editar.

Los scripts funcionan como un componente más del Game Object.

TRUCO: a menudo queremos añadir scripts a la escena que no estén asociados a un Game Object en concreto. Podemos crear un Empty Object y asociarlo a él.

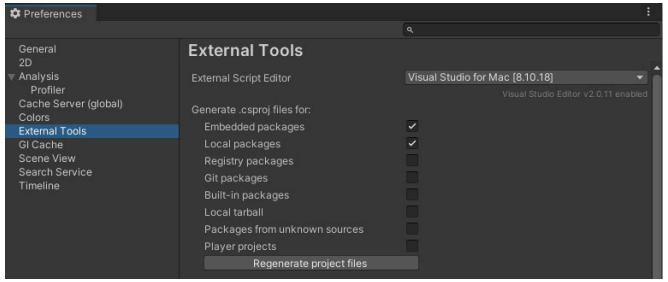
Para abrir el script podemos hacer un doble click en el panel de proyecto, o dar a Editar en el menú de hamburguesa del inspector. Pero antes de hacerlo, tenemos que configurar con qué programa se abrirá.

Abrir el script mediante nuestro editor de código (IDE)

Aunque los scripts son archivos de texto, y el código lo podemos editar con cualquier editor, es necesario vincular Unity con el software de edición de código para que reconozca la sintaxis propia de Unity.

Aunque podemos trabajar con varios, aquí lo haremos con Visual Studio. Puedes descargar las diferentes versiones aquí.

Una vez instalado, nos aseguraremos de que Unity lo usa como editor predeterminado. Para ello, vamos a Edit > Preferences > External Tools

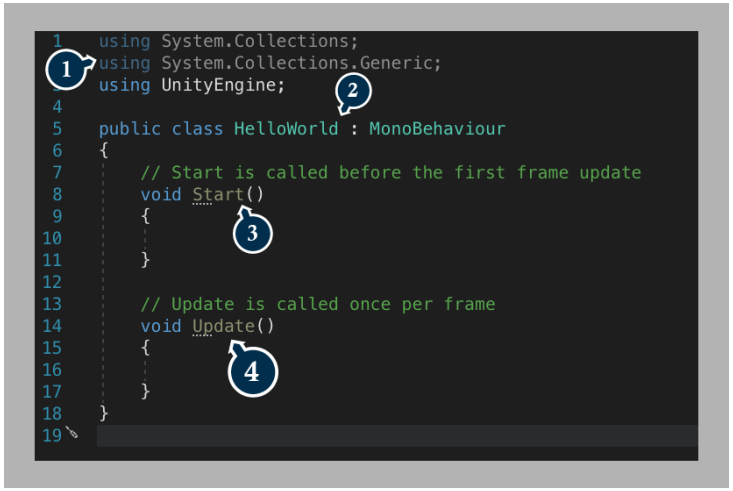


En el desplegable de "External Script Editor" nos aseguramos de seleccionar el programa que tengamos instalado, para que se ejecute al abrir el script.

Sabremos que está correctamente vinculado porque el editor reconocerá sintaxis de Unity. Si en algún momento deja de hacerlo, se recomienda pulsar el botón que se ve en la imagen: "Regenerate Project Files"

ESTRUCTURA DE UN SCRIPT

Para empezar a practicar, vamos a crear un script llamado "HelloWorld" (es importante escribirlo con mayúscula y sin espacios en blanco), que añadiremos como componente a un objeto de la escena (puede ser un Empty Object) y lo abriremos. En la siguiente imagen veremos lo que aparece en ese script recién creado:




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

public class HelloWorld : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

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

    }
}
```

Veamos en detalle los elementos que la componen:

• Lo primero que encontramos son las librerías (1) que necesitamos para acceder a las funcionalidades básicas de Unity. Se incorporan usando la sintaxis "using". Más adelante, para usar funcionalidades específicas, tendremos que incorporar otras librerías (la documentación de Unity especifica en todo momento las librerías que se necesitan)

• A continuación, se declara la clase (2), que como verás tiene el mismo nombre que el archivo, y así debe ser. Como hemos visto en la Unidad anterior, hereda de la clase Monobehaviour de Unity.

• Dentro de el script encontraremos dos métodos (funciones) que vienen por defecto, aunque no son necesarias:

Start (3): se invoca una vez, al cargar por primera vez el script. Recuerda que si ningún objeto de la escena tiene asociado el script, no se ejecuta, y si lo tienen 20 objetos pues se ejecuta 20 veces.

Update (4): se invoca cada fotograma de ejecución del juego. Es decir, que si estamos jugando a 60 fps, se ejecutará 60 veces por segundo.

IMPORTANTE: el archivo con el script debe tener el mismo nombre que la clase publica creada. Si cambiamos el nombre del archivo, lo tenemos también que hacer a la clase, y viceversa.

PARA NOTA: existen otros métodos similares a Start y Update, pero con sutiles diferencias:

• "Awake": Similar al Start, pero se ejecuta antes, al cargarse la escena, lo que permite declarar atributos para que otros objetos accedan a ellos al cargarse. Se ejecuta aunque el objeto esté desactivado.

• "LateUpdate": similar a Update, pero se ejecuta cuando se han cargado el resto de elementos del juego.

• "FixedUpdate": a diferencia de Update que se ejecuta lo antes posible, este se ejecuta a intervalos regulares. Es aconsejable para usar con colisionadores.

• "Corutinas": se ejecutan cada cierto tiempo y en segundo plano. Las veremos más adelante.

HOLA MUNDO

Vamos a realizar un ejercicio básico para ver cómo funciona el script: mandar a consola un mensaje, que no puede ser otro que "Hola Mundo".

Para mandar un texto a consola podemos hacerlo de dos formas:

1. Usando la sintaxis print() y dentro el mensaje. Es el método nativo de C#.

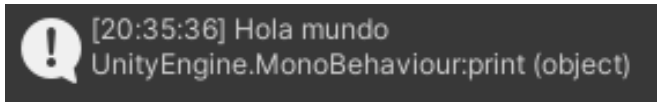
2. O usando el método heredado Debug.Log(). Funciona igual, pero la clase Debug de Unity tiene más funcionalidades. Puedes verlas aquí: https://docs.unity3d.com/ScriptReference/Debug.html

Ahora, crea una variable global de tipo cadena de texto, que contendrá el mensaje, y mándalo a consola en el método Start, así:

```
public class HelloWorld : MonoBehaviour
{
    //Declaramos la variable y le damos un valor
    string saludo = "Hola mundo";

    // Start is called before the first frame update
    void Start()
    {
        //Mandamos el mensaje a consol
        print(saludo);
    }

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

    }
}
```

IMPORTANTE: es necesario guardar los cambios en el script para que Unity lo reconozca, algo que hace automáticamente.

Lanza el juego en Unity (Ctrl. + P) y observa la consola. Recuerda que como programador debes tener siempre el panel de consola visible, y con las alertas activadas, en este caso las "blancas".



Verás que además del mensaje, muestra la hora en la que se ha lanzado y el método usado para enviarlo. Si haces doble click en el mensaje de consola, te llevará a la línea de código que lo ha mandado.

Ahora haz lo mismo, pero en lugar de mandar el mensaje en el método "Start" hazlo en el método "Update", a ver qué ocurre.

PRACTICA

Crea una variable, en este caso de número entero, y dale un valor de cero.

En el método Update, manda un mensaje que concatene el texto "Fotograma: " + la variable creada, pero asegúrate de que en cada ciclo de Update le sume uno. Ejecuta el juego, y habrás creado un contador de fotogramas.

RECOMENDACIÓN: los métodos "Start" y "Update" es recomendable tenerlos lo más limpio de código posible, por lo que se recomienda recopilar el código que aparece y meterlo en métodos a los que llamaremos. Ya veremos cómo.

MÁS INFORMACIÓN: https://docs.unity3d.com/ScriptReference/

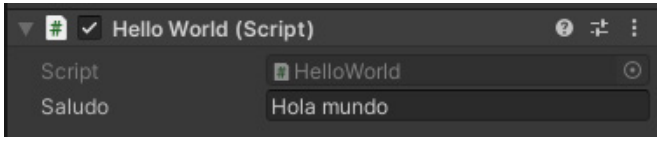
Variables públicas y serializadas

Recuerda que las variables en POO pueden ser privadas (solo accesibles dentro de la clase, y las que asigna por defecto), públicas (accesibles desde otras clases) y estáticas (si cambian su valor, se cambia en todas las instancias de esa clase).

Pero declarar una variable pública en Unity tiene otro efecto. Para comprobarlo, usando el ejemplo anterior, la variable "saludo" que hemos creado, declárala pública, de esta forma:

```
public string saludo = "Hola mundo";
```

Guarda el script, y vuelve a Unity. Fíjate en el componente con el script (debes seleccionar el Game Object de la escena que lo tiene). Verás que ahora puedes cambiar su valor directamente en Unity:



Prueba a poner otro texto y lanza el juego. Verás que se manda a consola el nuevo texto: el valor que le has dado en Unity sobrescribe el que se le ha asignado al declarar la variable. Si quieres sobreescribir a su vez el valor de Unity, deberás dárselo en el método "Start".

TRUCO: si haces click en la casilla con el nombre del script que tiene una diana a la derecha, te señalará en la ventana de proyecto dónde se encuentra tu script.

Hay otra forma de permitir cambiar el valor de las variables en Unity sin declararlas públicas, por motivos de seguridad. Para ello, debemos añadir [SerializeField] al comienzo, lo que permite también acceder desde la interfaz, pero sin hacerla pública. De esta forma:

```
[SerializeField] string saludo = "Hola mundo";
```

RECOMENDACIÓN: si vas a declarar variables globales en la clase (lo que serán sus atributos), lo mejor es hacerlo justo después de declarar la clase, dentro de ella claro, y antes de cualquier método, para tenerlas identificadas al comienzo de todo.

PARA NOTA: ¿Recuerdas los arrays? Prueba a crear un array público o serializado y mira cómo aparece en Unity. Descubrirás que puedes decirle el tamaño y los contenidos.

Convenciones que debemos seguir al escribir scripts:

• Las clases deben escribirse con mayúscula. Eso implica que el nombre de los archivos que los contienen también.

• NUNCA debemos crear archivos (es decir, clases) usando nombres de clases ya existentes y heredadas de Unity. Por ejemplo, si creamos un archivo llamado "Time" creará conflicto con la clase heredada del mismo nombre.

• Los métodos también se escribirán con mayúscula preferiblemente.

• Las variables se escribirán con minúscula.

• Usaremos la técnica de CamelCase para los nombres.

• Las variables y las clases no deben tener espacios, ni caracteres especiales, ni comenzar con un número.

• Es conveniente crear una subcarpeta en nuestro proyecto que contenga todos los scripts

• Deberemos prestar atención a la ventana de consola, donde nos indicará los errores de nuestro código, siempre indicando el archivo, la línea y la columna del error, y el tipo de error.

INSTANCIANDO UN PREFAB

Ya que tenemos creado un prefab de un ejercicio anterior, vamos a aprender cómo llevarlo a escena mediante código.

Para hacerlo, deberemos usar el métdo de "instanciar", cuya sintaxis es:

```
Instatiate(Objeto, posición padre opcional, rotación opcional)
```

Al método tenemos que pasar 3 variables: el objeto a instanciar, su posición y su rotación.

Cuestiones a tener en cuenta:

1. La primera variable es una de tipo "GameObject".

2. Si solo pasamos una variable, la del objeto, instanciará el prefab en las coordenadas 0,0,0.

3. Las variables de posición y rotación puedes reemplazarse por una variable de tipo "Transform", pero si lo hacemos, el prefab se instanciará cómo hijo del elemento del que hemos pasado el componente Transform.

4. Si no queremos indicar una rotación, podemos usar el método "Quaternion.Identyity" que, como veremos más adelante, es como pasar unas coordenadas de rotación nulas.

El código del script, que asociaremos a un objeto de la escena y que analizaremos a continuación será el que se ve a continuación.

```
public class InstatiatePrefab : MonoBehaviour
{
    //Creamos una variable serializada que contendrá el prefab
    [SerializeField] GameObject myPrefab;

    //Creamos otra variable de tipo Transform, que contendrá la posición
    [SerializeField] Transform reference;

    //Veremos varias opciones para instanciar
    void Start()
    {
        //Crea una instancia en la coordenadas 0,0,0
        Instantiate(myPrefab);

        //Crea una instancia en la posición del objeto de referencia y como hijo suyo
        Instantiate(myPrefab, reference);

        //Crea una instancia en el punto de referencia, pero sin rotación
        Instantiate(myPrefab, reference.position, Quaternion.identity);

    }
}
```

• Como verás, hemos borrado el método Update porque no era necesario en esta ocasión.

• Se han creado dos variables serializadas para poder dotarlas de valor directamente en Unity, una de tipo "GameObject" que contendrá el Prefab (lo arrastraremos desde el panel de proyecto hasta la casilla del valor de la variable), y otra de tipo Transform, que indicará dónde queremos instanciar el Prefab.

Para que la variable Transform tenga un valor, crearemos un Empty Object en la escena y lo arrastramos a la casilla de la variable, y automáticamente tomará su componente Transform (es decir, su posición y rotación)

Opcionalmente, podemos arrastrar directamente el componente Transform del mismo objeto que tiene asociado el Script.

• Una vez tenemos la variable Transform, podemos acceder a sus diferentes atributos: position, rotation, scale, ya que funciona como un objeto (recuerda la POO).

• Vemos las tres formas en las que se puede instanciar el Prefab.

Aquí ves el componente del Script (que he llamado InstantiatePrefab) contiene las dos variables serializadas, a las que he arrastrado a una el Prefab y a la otra un EmptyObject de la escena llamado "InitGame".

PARA NOTA: Además, podemos asignar esas instancias a una variable de tipo GameObject, lo que nos será muy útil. Esta sería la forma:

```
GameObject instanc = Instantiate(myPr);
```

PRACTICA: Como imaginarás, podemos crear tantas instancias como queramos, en la posición que queramos. ¿Serías capaz de crear 10 instancias de un prefab mediante un bucle, como los que vimos en la introducción a la programación?

OTRAS HERRAMIENTAS DE UNITY

Vamos a ver algunas herramientas adicionales que nos ofrece Unity que nos serán de utilidad.

Project Settings

Si vamos a Edit->Project Settings, podremos acceder a muchas configuraciones específicas de nuestro proyecto. Algunas son:

• "**Input**": ejes de nuestro juego (podemos añadir más, añadiendo uno a "Size"), y asignarle las teclas y funciones que queramos

• **Tags and Layers**: podemos configurar la lista de etiquetas y capas que, más adelante, podremos asignar a nuestros elementos del juego. Las etiquetas sirven para identificar objetos, y las capas principalmente para crear prioridades de renderizado o evitar colisiones de objetos en capas distintas

• **Audio**: configuraciones básicas del sonido

• **Time**: ajustar los intervalos entre fotogramas (lo veremos cuando apliquemos el método "fixedUpdate"), o incluso cambiar la velocidad del tiempo absoluta

• **Player**: nos permite configurar cuestiones relacionadas con la compilación para ciertas plataformas (como Anrdoid, iOS, Web, etc.). En este apartad pondremos el nombre de nuestra compañía, del producto, así como la imagen del producto.

• **Physics**: aspectos relacionados con la simulación de físicas. Es en esta ventana donde podemos determinar qué capas colisionarán con otras

• **Physics2D**: similar al anterior, pero con opciones propias del 2D.

• **Quality**: podremos configurar la calidad final del proyecto dependiendo de la plataforma a la que es exportada. Entrando en la configuración específica podemos configurar cómo se comportarán las sombras, las textutas, los renderizados, etc.

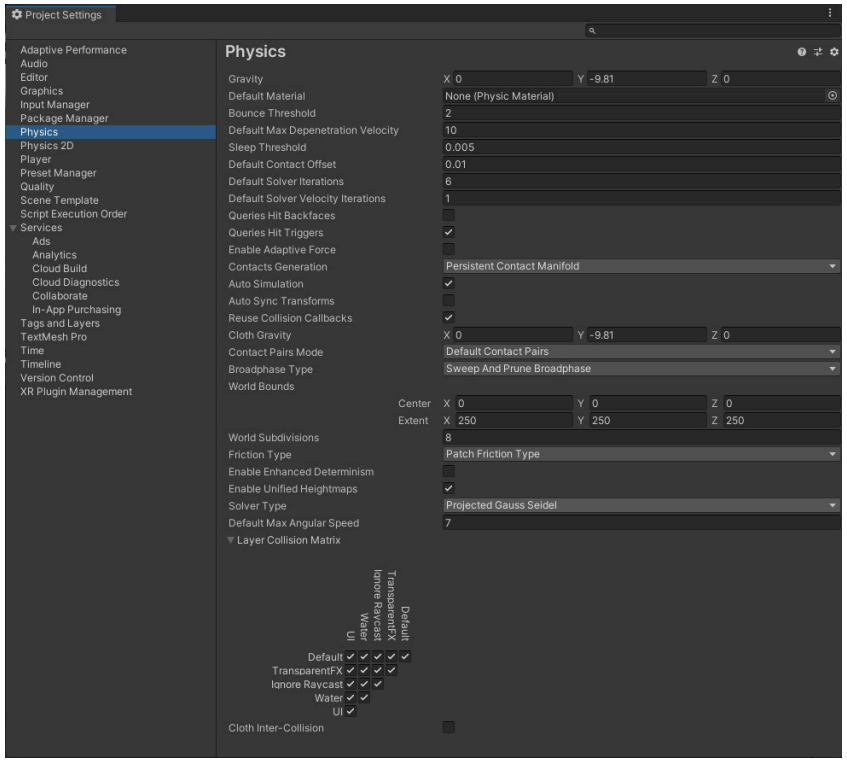
• **Graphics**. Aspectos relacionados con los gráficos de nuestro juego, especialmente a cómo se comportan los shaders.

• **Network**: funcionamiento en red de nuestro juego.

• **Editor**: permite configurar cómo se configurará con respecto al Editor de Unity. Por ejemplo, Unity Remote (testear en móvil), sistema de control de versión, URL del host para pruebas, etc.

• **Script Execution Order**: las funciones Awake, OnEnable y Update se ejecutan según se cargan los scripts, pero aquí se puede alterar ese orden.

En la imagen, vemos un ejemplo de ventana, con la pestaña que gestiona las físicas, donde podemos entre otras cosas indicar la fuerza de la gravedad, o las capas que colisionan entre sí:



OTRAS HERRAMIENTAS DE UNITY (II)

Package Manager

Mediante la herramienta Window->Package Manager, podemos gestionar los paquetes instalados en nuestro proyecto, y también los que están disponibles.

También se puede ver la versión de cada paquete, actualizarlo, desinstalarlo, etc.

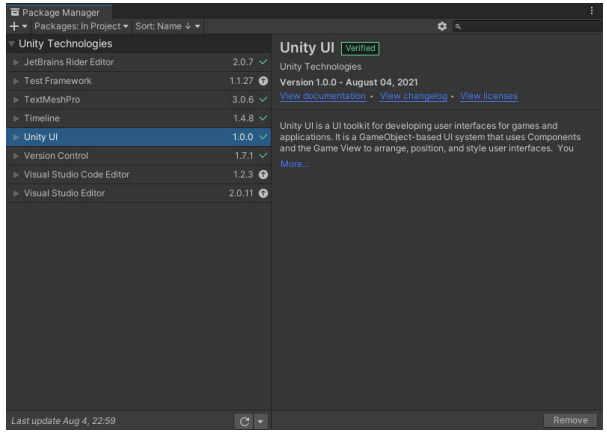
Los paquetes ofrecen una gran variedad de funcionalidades a nuestro proyecto, funcionando como "plugins" de Unity.

Algunos ejemplos que veremos en la siguiente unidad:

- Nuevo "Input System Package" de Unity, que permite un mayor control sobre los dispositivos de entrada y el registro de interactividad.

- El paquete de Cinemachine nos permite crear cinemáticas, y en las versiones más recientes, funciones de seguimiento de cámara.

En la herramienta podemos ver los paquetes que hay en el proyecto, o buscar en las bases de datos de Unity (Unity Registry).



Asset Store

El panel de la tienda de assets ("Asset Store") antiguamente integraba esta herramienta, pero ahora redirige a la página web: https://assetstore.unity.com/

En este sitio podemos buscar recursos, en una gama enorme de precios (algunos gratuitos también), y de muchos tipos.

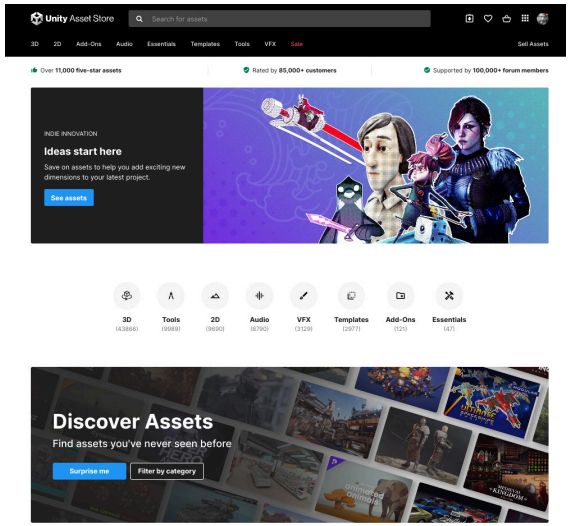
Debemos estar logeados con nuestra cuenta de Unity si queremos adquirir algún asset.

Se pueden realizar búsquedas por distintos criterios, previsualizar los assets, explorar su contenido, etc.

IMPORTANTE: Hay que tener cuidado con las licencias de uso, especialmente en las que son gratuitas.

Una vez adquirido un asset, si nos vamos a la ventana de Package Manager y seleccionamos "My Assets" nos aparecerán allí disponibles para descargar (si no lo hemos hecho ya) o para incorporar a nuestro proyecto.

Cuando importamos un paquete de assets, podemos importar todos los elementos o solo algunos.



ENLACES A VÍDEOS

Aquí podrás acceder a los vídeos que explican y/o profundizan sobre los conceptos vistos, por si te son de utilidad.

Interfaz de Unity

Introducción a la interfaz del programa

https://www.youtube.com/watch?v=TP_qrtchjYA

Añadiendo scripts

Cómo añadir scripts y modificarlos

https://www.youtube.com/watch?v=XaY0RLMSdFg

Instanciando prefabs

Creando e instanciando prefabs

https://www.youtube.com/watch?v=nqgsnGjqYCU

Usar los arrays para instanciar prefabs

https://www.youtube.com/watch?v=L8QpQ_mg2Zk