

# Introducción a Unity



[www.cesarguerra.mx](http://www.cesarguerra.mx)

V1.0



# Acerca de César Guerra

-  [www.cesarguerra.mx](http://www.cesarguerra.mx)
-  [cesar@cesarguerra.mx](mailto:cesar@cesarguerra.mx)
-  [@warderer](https://twitter.com/@warderer)
-  [cesar.guerra.mx](https://facebook.com/cesar.guerra.mx)



Ing. En Computación – UNACAR  
Mtro. En Dirección de Ingeniería de Software – IEU

Fanático de las tecnologías y los nuevos retos:  
Agente de cambio ágil, diseñador, desarrollador,  
project manager y entre ratos gamer.



# Contenido

- Fundamentos de Unity
- Instalación
- Creando nuevos proyectos
- Interfaz
- Programación con C#



# Fundamentos de Unity

## ¿Qué es Unity?

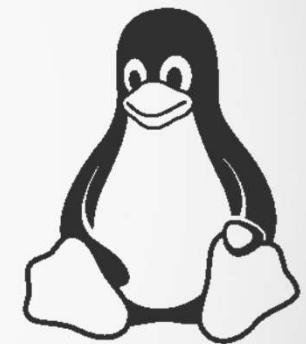
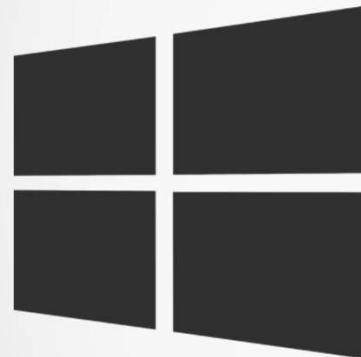
Unity es una plataforma de desarrollo de videojuegos.

Usada para crear juegos 3D y 2D, desplegarlos a través de dispositivos móviles, escritorio, VR/AR, consolas o la web.



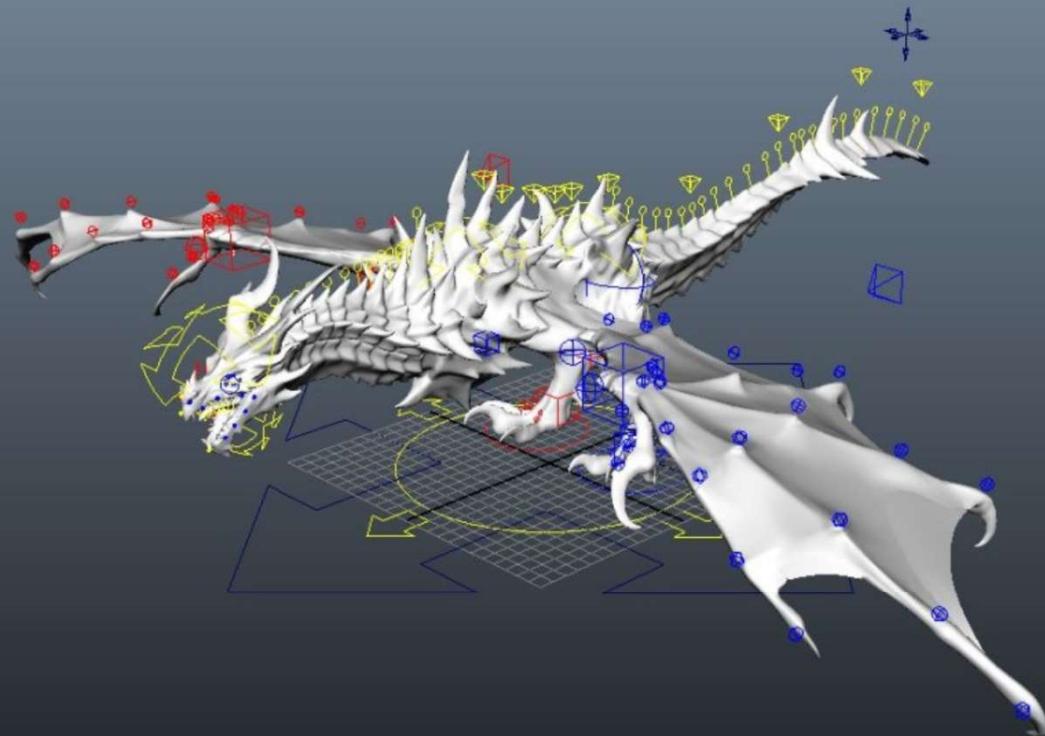
## ¿Qué es Unity?

Unity esta disponible para desarrollar desde plataformas Windows, Mac OS y Linux.



## Interoperabilidad de Unity

Unity puede usarse junto con Blender, 3ds Max, Maya, Softimage, Modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks y Allegorithmic Substance.



# Proyectos Destacados

## Cup Head



# Proyectos Destacados

FireWatch



## Proyectos Destacados

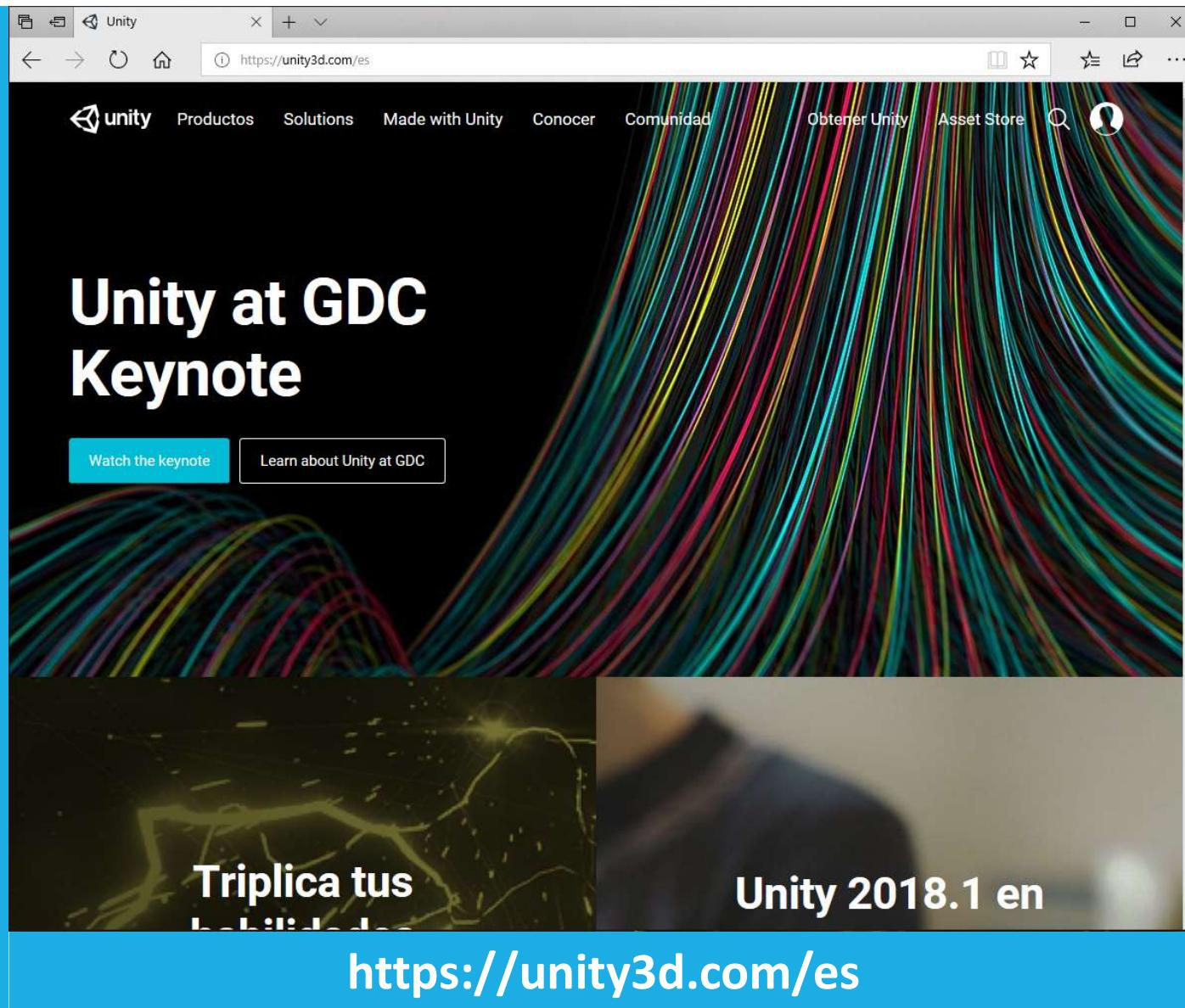
### Super Mario Run



## ¿Cómo consigo Unity?

Se descarga a través de su sitio web, haciendo clic en el enlace de “Obtener Unity”.

Nota: Es necesario crear una cuenta de Unity para poder utilizar la herramienta.



The screenshot shows the Unity website homepage (<https://unity3d.com/es>). The main title "Unity at GDC Keynote" is displayed prominently. Below it are two buttons: "Watch the keynote" (in a blue box) and "Learn about Unity at GDC". The background features a dark, abstract graphic of many colorful, curved lines resembling light or energy streaks. At the bottom of the page, there are two visible text snippets: "Triplica tus habilidades" and "Unity 2018.1 en". The URL <https://unity3d.com/es> is displayed at the bottom of the slide.

# Licenciamiento

**Plan Personal:** Gratuito e ideal para aprender. Podemos usarlo si facturamos menos de 100 mil USD anuales.

**Plan Plus:** Costo mensual, con más herramientas. Se debe de facturar anualmente menos de 200 mil USD para poder acceder a este plan.

**Plan Pro:** Costo mensual, aún más herramientas. Si facturamos más de 200 mil USD, debemos usar este plan.



The screenshot shows the Unity Store homepage with a dark background featuring yellow glowing lines. The main headline reads: "¿Listo para empezar a crear? Ponte serio con Unity Plus o Unity Pro." Below this, a green bar states: "Todos los planes de Unity son libres de regalías e incluyen Todas las plataformas, prestaciones del motor principal, actualizaciones continuas y acceso a versiones beta." Three plans are listed in cards:

- Personal**: Gratis. Para principiantes, estudiantes y aficionados que desean explorar y empezar a trabajar con Unity. Includes: Todas las prestaciones básicas del motor.
- Plus** Best Seller: 35 \$ por mes. Para los creadores que tienen la seria intención de hacer realidad su visión y piensan publicar sus contenidos. Includes:
  - Ahora se incluye:
    - Cursos para desarrolladores de juegos de Unity + herramienta Bolt de Visual Scripting Tool ([valorizados en \\$214](#))
    - Beneficio para suscriptores del 20 % de descuento en la Asset Store
  - Ver oferta
- Pro**: 125 \$ por mes. Para los profesionales que necesitan absoluta flexibilidad y anhelan una personalización avanzada. Includes:
  - Ahora se incluye:
    - Beneficio para suscriptores del 20 % de descuento en la Asset Store
    - Servicios de nivel Pro
  - Todas las prestaciones de la versión Plus
  - Tienes a tu disposición planes con soporte Premium y acceso al código fuente
  - Requisitos del plan: sin límites de ingresos o financiamiento

# Cuenta de Unity

La creación de la cuenta de Unity puede hacerse desde la página web de Unity o bien al abrir por primera vez la aplicación.

unity

### Crear una Unity ID

Si ya tienes una Unity ID, [inicia sesión aquí](#).

Correo electrónico

Contraseña

Nombre de usuario

Nombre completo

Haz clic o toca la Mano



Acepto las [Condiciones de uso](#) y la [Política de privacidad](#)

¡Recibe novedades, descuentos y más de Unity!

[Crear una Unity ID](#)

¿Ya tienes una Unity ID?

0

 [Iniciar sesión con Google](#)

 [Iniciar sesión con Facebook](#)

# Requerimientos Técnicos

## Para desarrollo

**OS:** Windows 7 SP1+, 8, 10, solo versiones de 64 bits; Mac OS X 10.9+.

No se han probado las versiones de servidor de Windows & OS X.

**CPU:** Soporte para el conjunto de instrucciones SSE2.

**GPU:** Tarjeta de video con capacidad para DX10 (shader modelo 4.0).

El resto depende principalmente de la complejidad de sus proyectos.

## Requisitos adicionales para el desarrollo de plataformas:

- iOS:** Computadora Mac con sistema operativo versión OS X 10.9.4 como mínimo y Xcode 7.0 o superior.

- Android:** Kit de desarrollo Android SDK y Java (JDK); el IL2CPP scripting backend requiere Android NDK.

- Windows Store:** Windows 10 (64 bits) y Visual Studio y plataforma SDK correspondiente:

- Universal Windows Platform (UWP): Windows 10 (64 bits), Visual Studio 2015 o posterior y Windows 10 SDK;
- El scripting backend IL2CPP también requiere la instalación de la función de compilador C++ con Visual Studio.

## Para ejecutar juegos de Unity

Por lo general, el contenido desarrollado con Unity puede ejecutarse bastante bien en todas partes.

Qué tan bien se ejecuta depende de la complejidad de su proyecto. Requisitos más detallados:

**Escritorio:**

OS: Windows Vista SP1+, Mac OS X 10.9+, Ubuntu 12.04+, SteamOS+.

Tarjeta de video con capacidad para DX10 (shader modelo 4.0).

CPU: compatible con el conjunto de instrucciones SSE2.

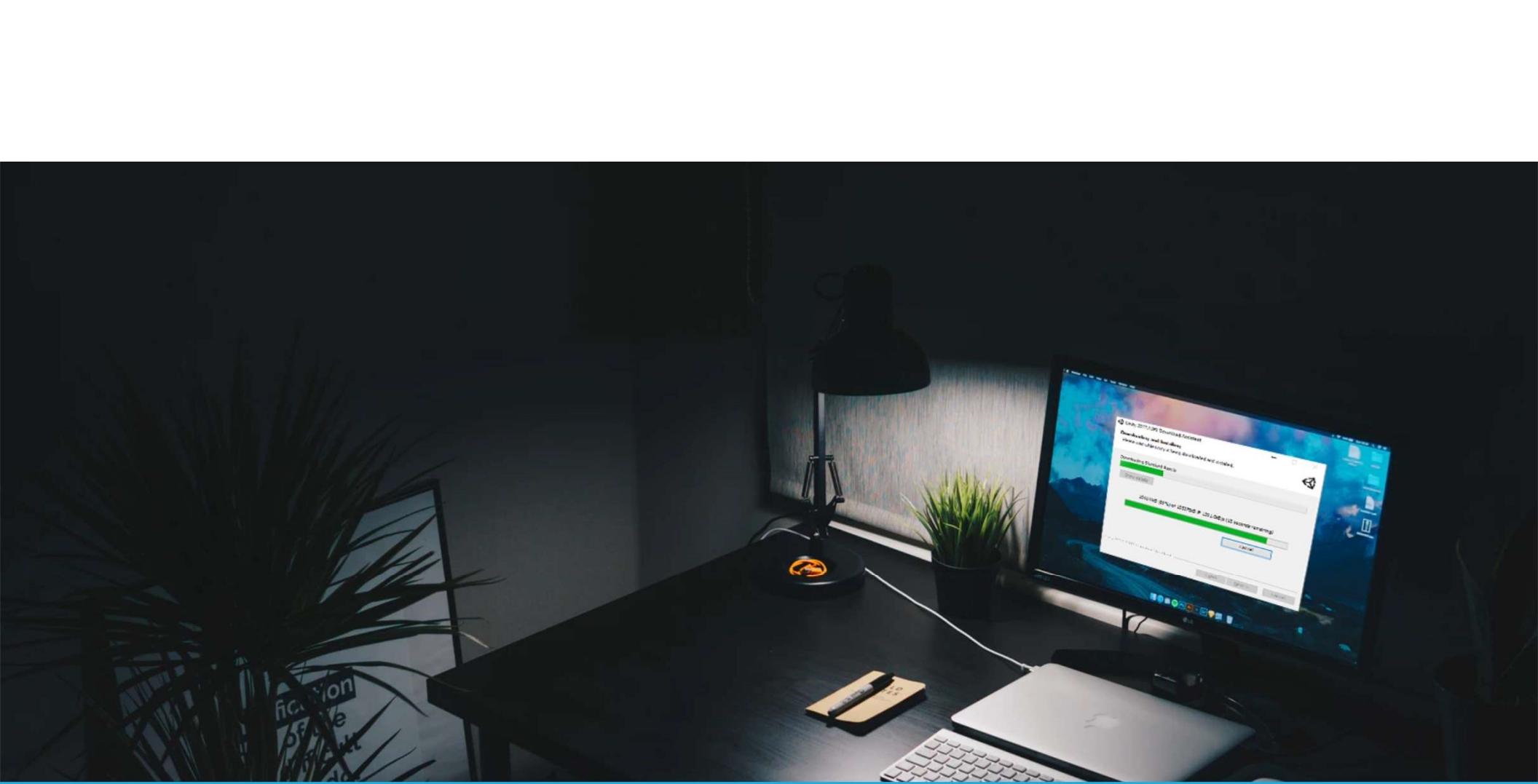
El reproductor de iOS requiere iOS 7.0 o una versión superior.

**Android:** OS 4.1 o posterior; ARMv7 CPU con soporte NEON o CPU Atom; OpenGL ES 2.0 o posterior.

**WebGL:** Cualquier versión de escritorio reciente de Firefox, Chrome, Edge o Safari

**Windows Store Apps:** Windows 10 y tarjeta de video con capacidad para DX10 (shader modelo 4.0).

<https://unity3d.com/es/unity/system-requirements>

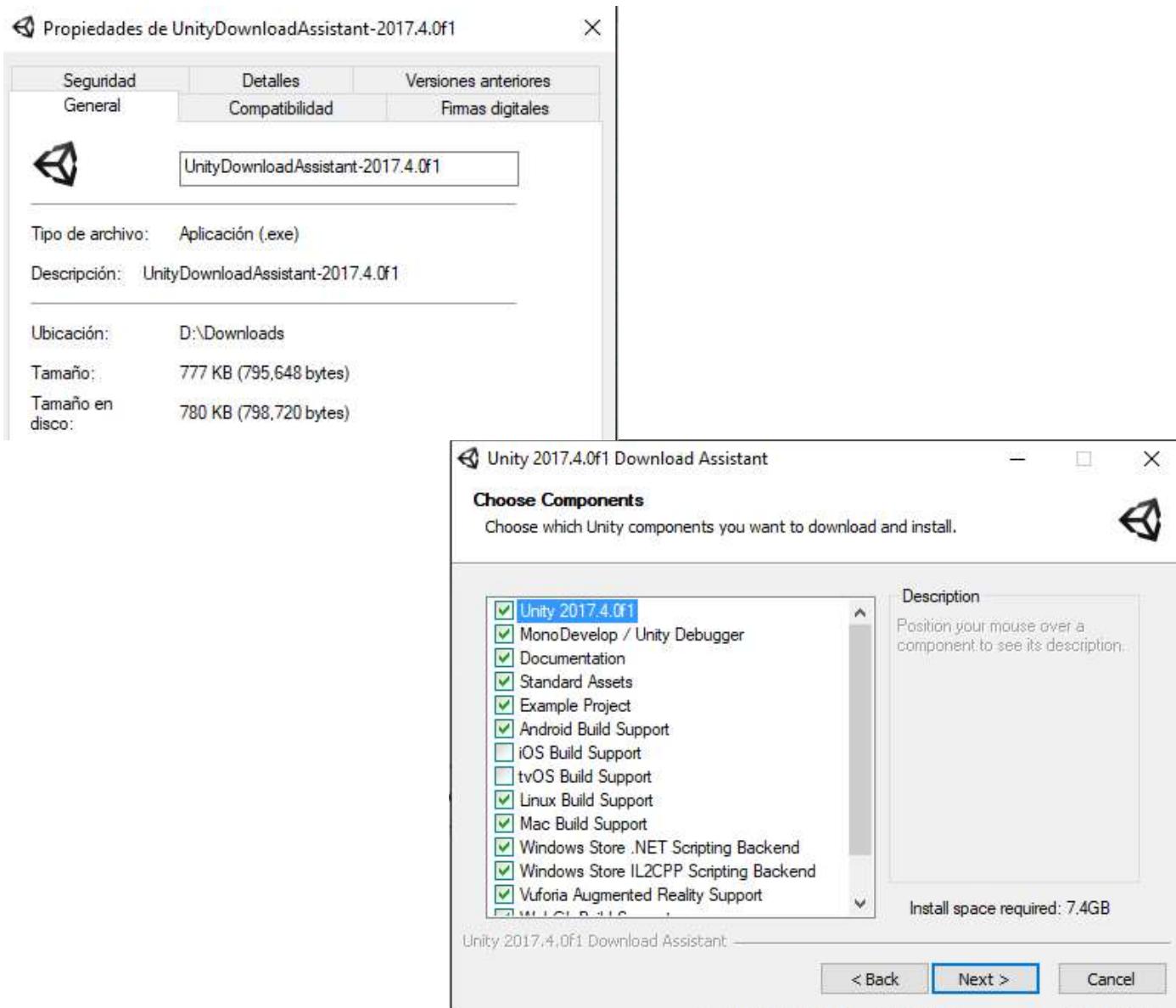


# Instalación

# Unity Download Assistant

Es la herramienta que nos permitirá descargar e instalar Unity y sus componentes.

Debemos seleccionar aquellos componentes que tengan sentido para la plataforma que planeamos desarrollar.



The image shows two windows related to the Unity Download Assistant. The top window is a file properties dialog for 'UnityDownloadAssistant-2017.4.0f1.exe'. It displays the following details:

General	Detalles	Versiones anteriores
Seguridad	Compatibilidad	Firmas digitales
UnityDownloadAssistant-2017.4.0f1		
Tipo de archivo: Aplicación (.exe)		
Descripción: UnityDownloadAssistant-2017.4.0f1		
Ubicación:	D:\Downloads	
Tamaño:	777 KB (795,648 bytes)	
Tamaño en disco:	780 KB (798,720 bytes)	

The bottom window is the 'Choose Components' screen of the Unity Download Assistant. It lists various components with checkboxes, all of which are checked. The checked components include:

- Unity 2017.4.0f1
- MonoDevelop / Unity Debugger
- Documentation
- Standard Assets
- Example Project
- Android Build Support
- iOS Build Support
- tvOS Build Support
- Linux Build Support
- Mac Build Support
- Windows Store .NET Scripting Backend
- Windows Store IL2CPP Scripting Backend
- Vuforia Augmented Reality Support
- Unreal Engine Integration

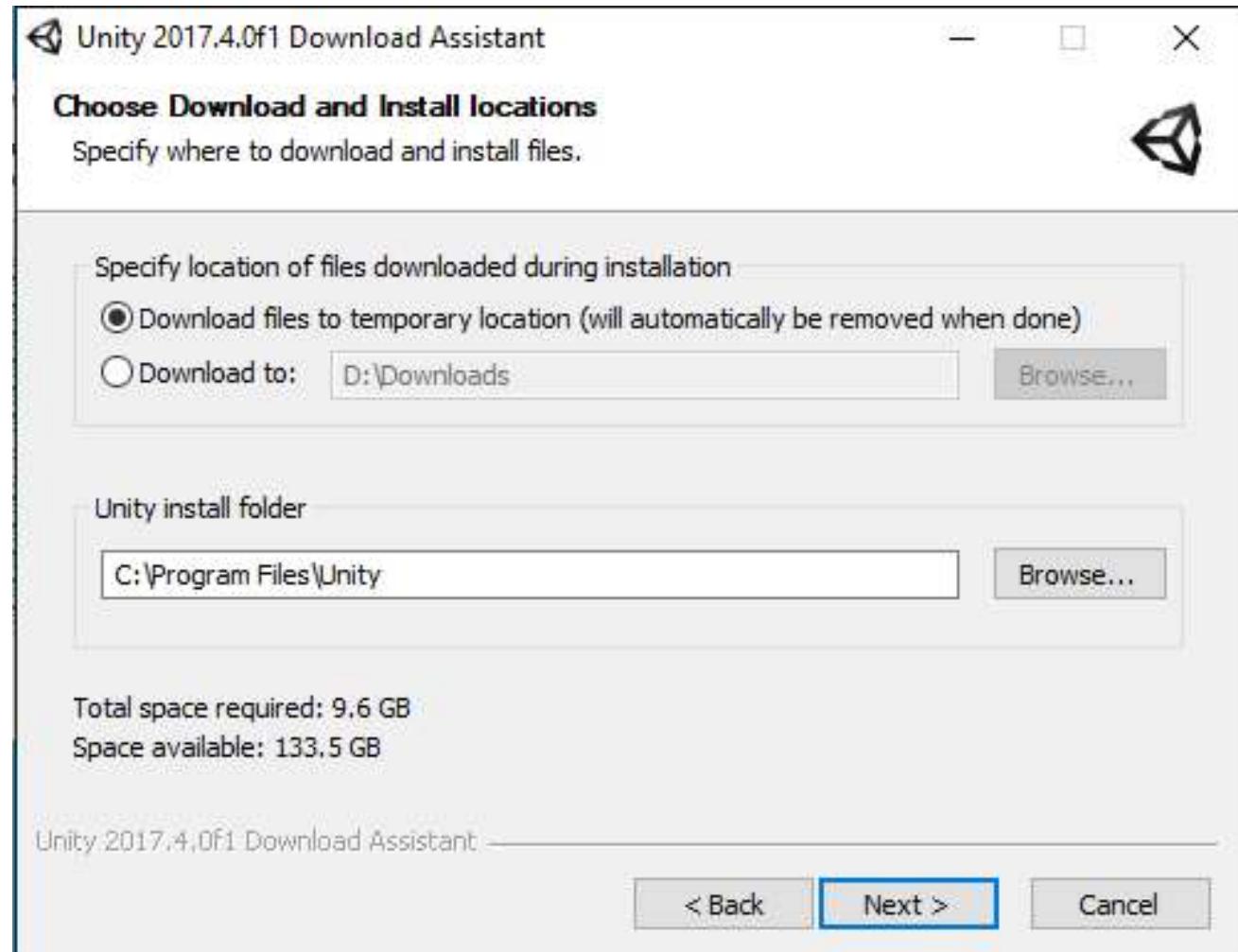
A description panel on the right says: "Position your mouse over a component to see its description." Below the component list, it says "Install space required: 7.4GB". At the bottom, there are buttons for '< Back', 'Next >', and 'Cancel'.

# Unity Download Assistant

Podemos elegir si queremos descargar el instalador de Unity de manera temporal y borrarlo después de instalarlo.

O bien:

Que guarde una copia del instalador en el disco duro.



# Unity Download Assistant

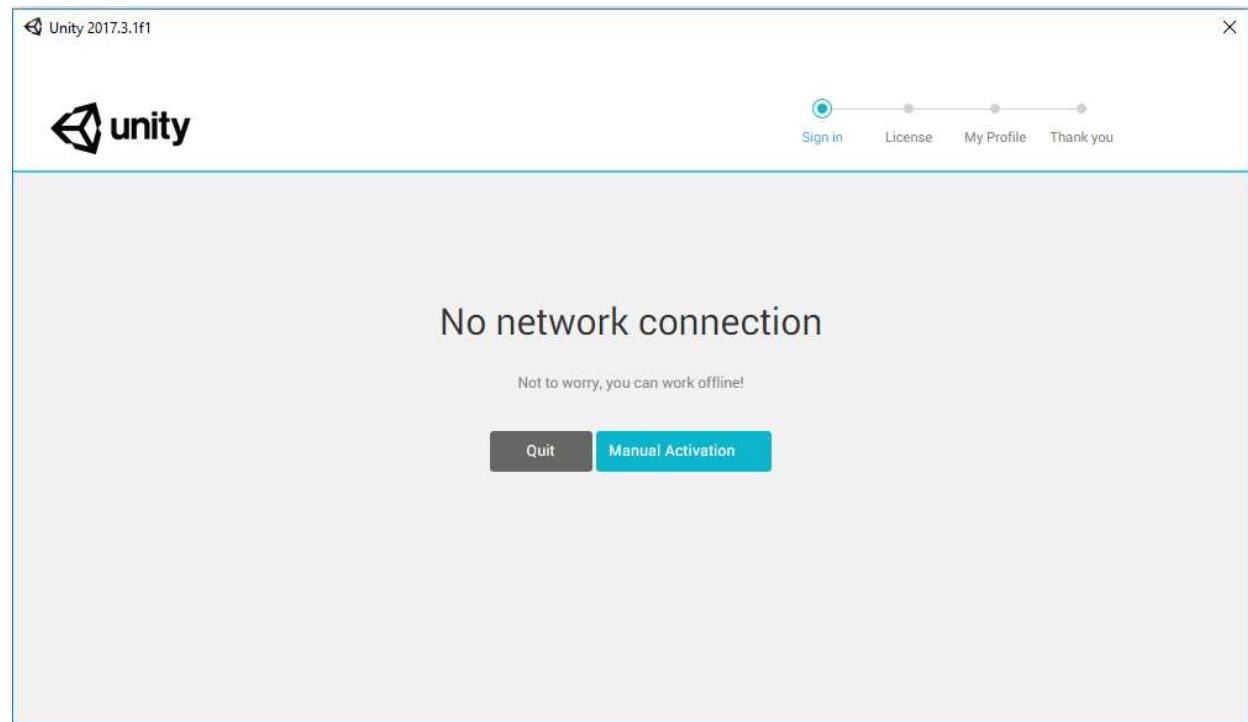
Archivos descargados en nuestro disco duro.

Nombre	Fecha de m...	Tipo	Tamaño
 FacebookGamesArcade	31/03/2018...	Paquete de Windo...	51,342 KB
 install	31/03/2018...	Archivo por lotes ...	2 KB
 UnityDocumentationSetup	31/03/2018...	Aplicación	363,061 KB
 UnityExampleProjectSetup	31/03/2018...	Aplicación	252,765 KB
 UnityMonoDevelopSetup	31/03/2018...	Aplicación	42,698 KB
 UnitySetup64	31/03/2018...	Aplicación	524,408 KB
 UnitySetup-Android-Support-for-Editor-2017.4.0f1	31/03/2018...	Aplicación	189,799 KB
 UnitySetup-Facebook-Games-Support-for-Editor-2017.4.0f1	31/03/2018...	Aplicación	31,391 KB
 UnitySetup-Linux-Support-for-Editor-2017.4.0f1	31/03/2018...	Aplicación	119,478 KB
 UnitySetup-Mac-Support-for-Editor-2017.4.0f1	31/03/2018...	Aplicación	27,449 KB
 UnitySetup-Metro-Support-for-Editor-2017.4.0f1	31/03/2018...	Aplicación	168,378 KB
 UnitySetup-UWP-IL2CPP-Support-for-Editor-2017.4.0f1	31/03/2018...	Aplicación	149,555 KB
 UnitySetup-Vuforia-AR-Support-for-Editor-2017.4.0f1	31/03/2018...	Aplicación	64,403 KB
 UnitySetup-WebGL-Support-for-Editor-2017.4.0f1	31/03/2018...	Aplicación	131,723 KB
 UnityStandardAssetsSetup	31/03/2018...	Aplicación	185,379 KB

# Activación de Unity

Después de instalar Unity, la primera vez que lo usemos nos pedirá activación, bastara con entrar con la cuenta previamente creada y elegir que usaremos la licencia personal.

Este proceso requiere conexión a internet. En caso contrario podemos optar por una activación Manual (offline)



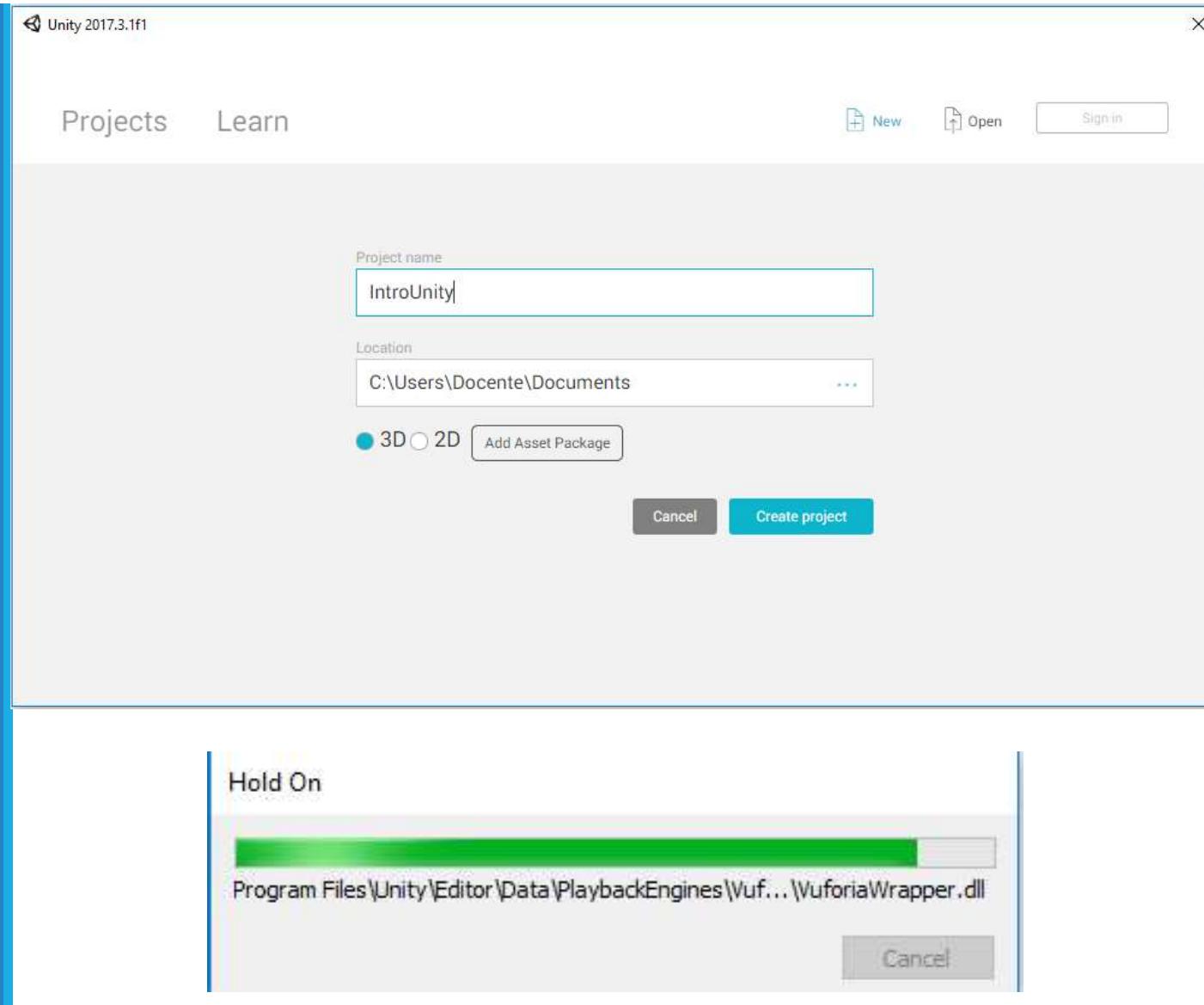


Creando nuevos proyectos

# Creación de un nuevo Proyecto

Es importante especificar la ruta donde se guardará físicamente el proyecto e indicar si lo que haremos será de tipo 2D o 3D.

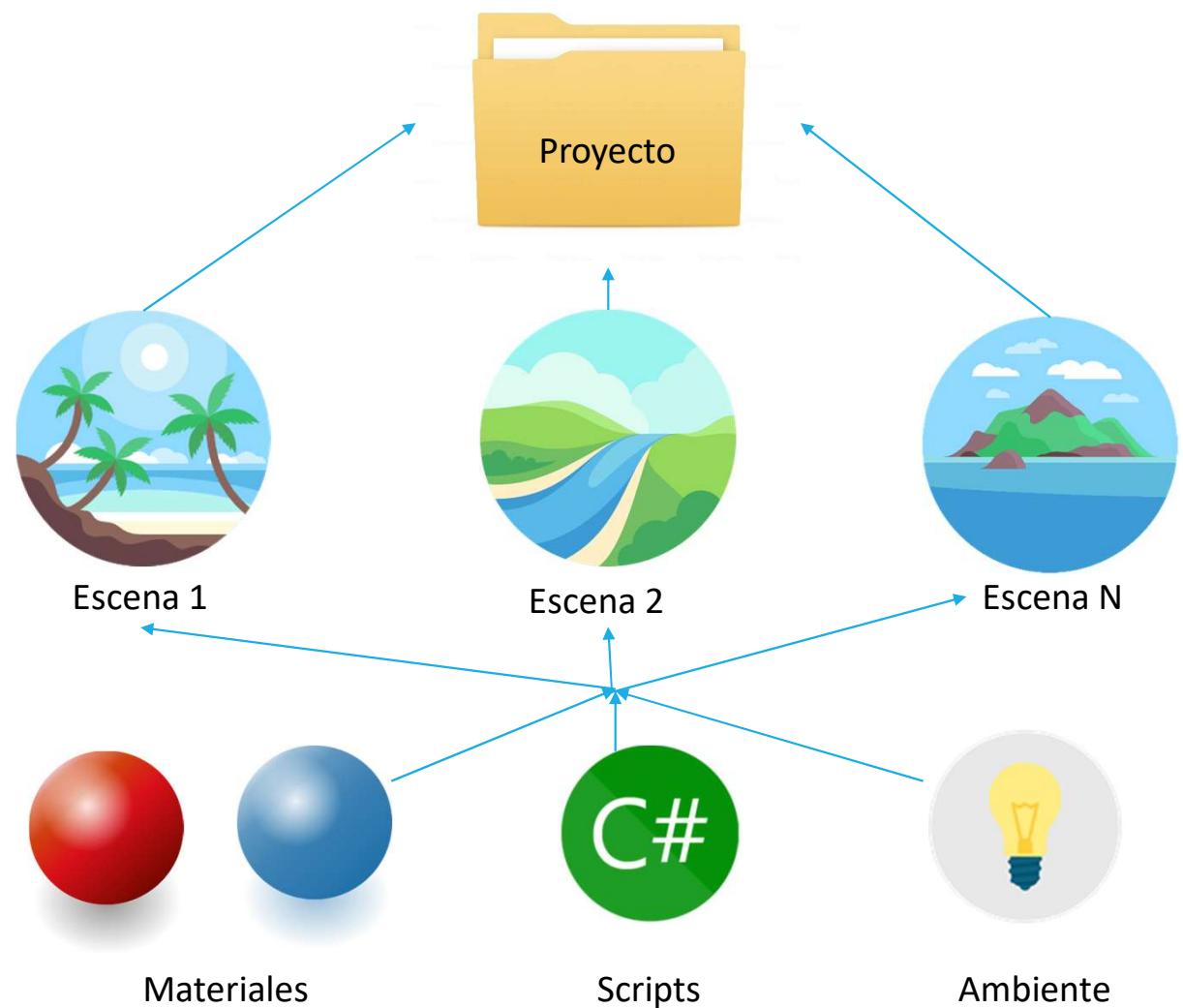
Es recomendable no usar espacios en blanco en el nombre del proyecto.

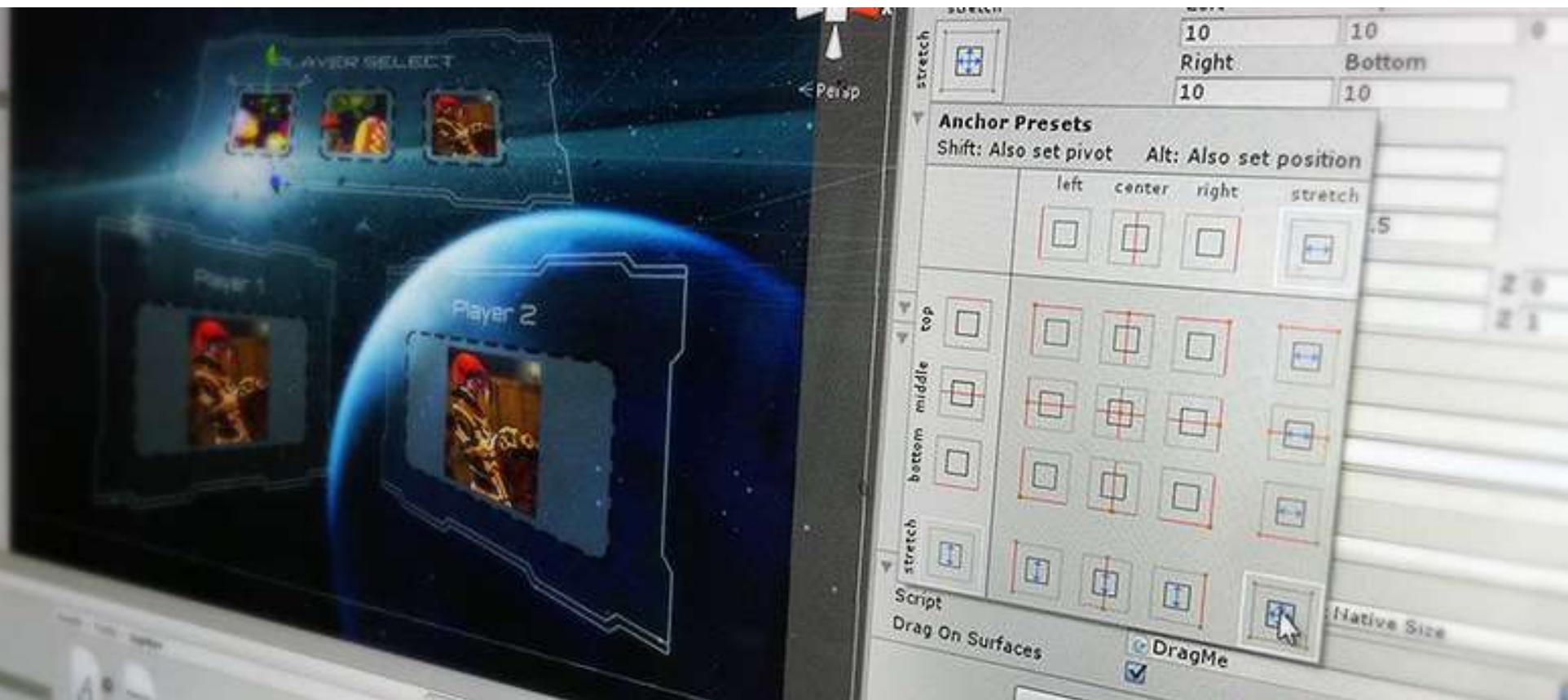


## Esquema de Trabajo de Unity

Un proyecto puede contener varias escenas. La lógica del juego nos permitirá movernos entre diferentes escenas (o niveles).

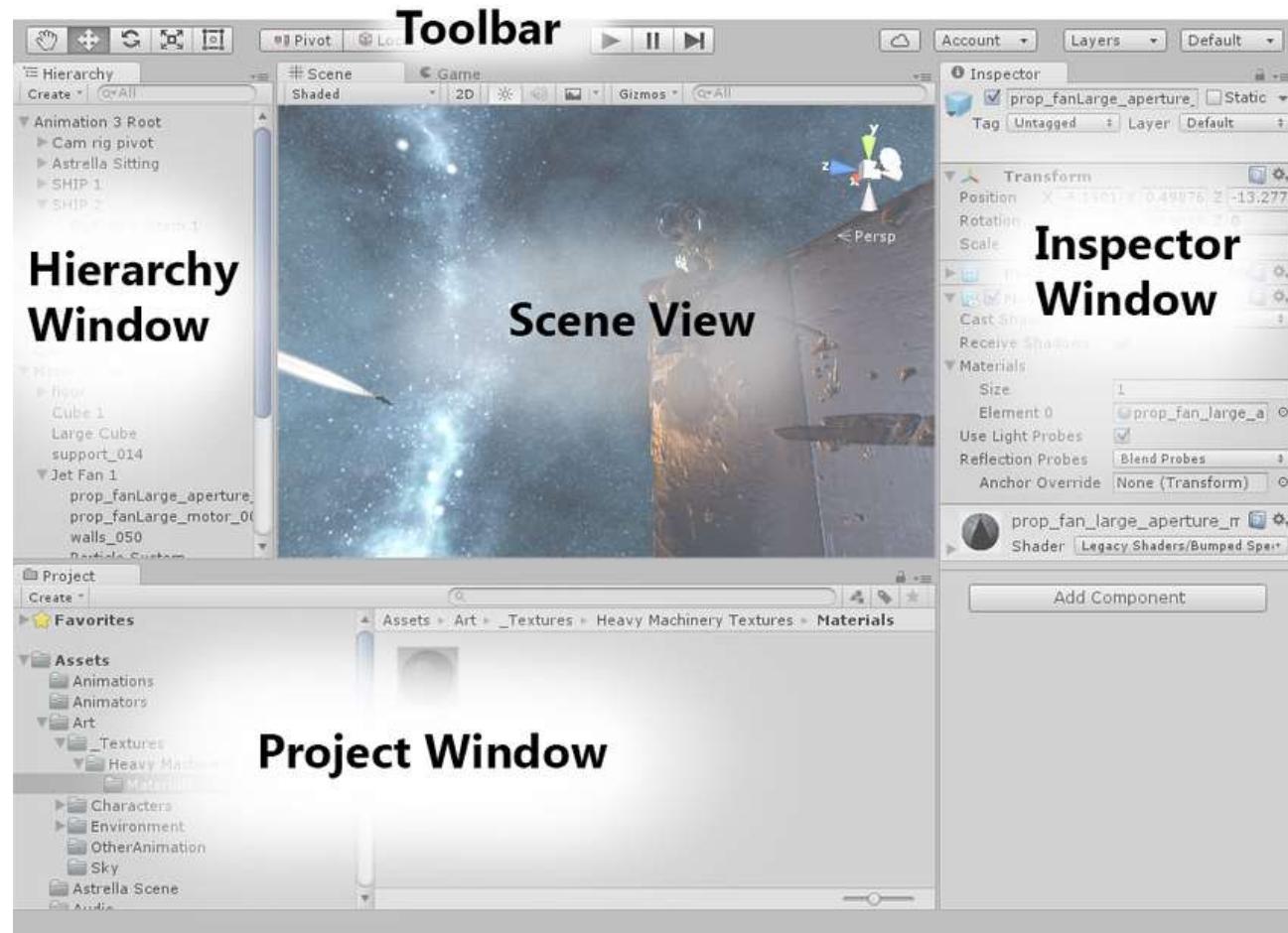
Los materiales, scripts de programación y elementos del ambiente forman parte de una escena.





# Interfaz

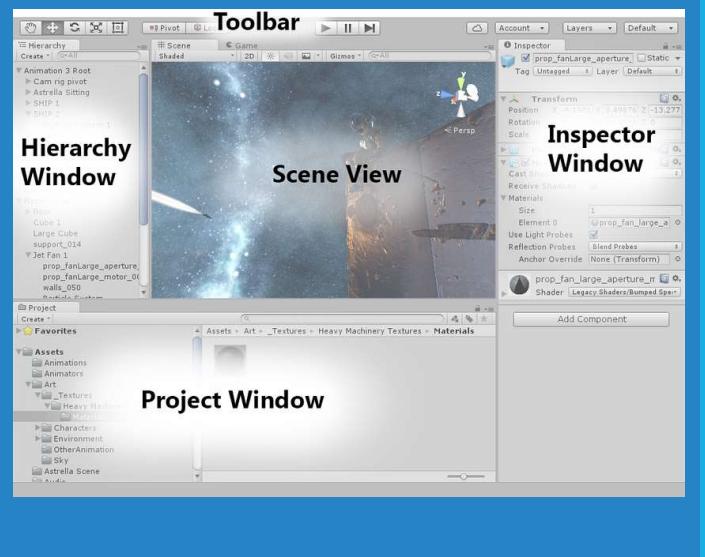
Primeros Pasos y Opciones Habituales



Interfaz General de Unity

# Interfaz

## Principales Elementos



**Toolbar:** Acceso a herramientas básicas como manipular la scene view y los objetos dentro de esta; los controles de reproducción, pausa, y pasos; capas, cuenta etc.

**Scene View:** Escenario virtual donde colocaremos los objetos.

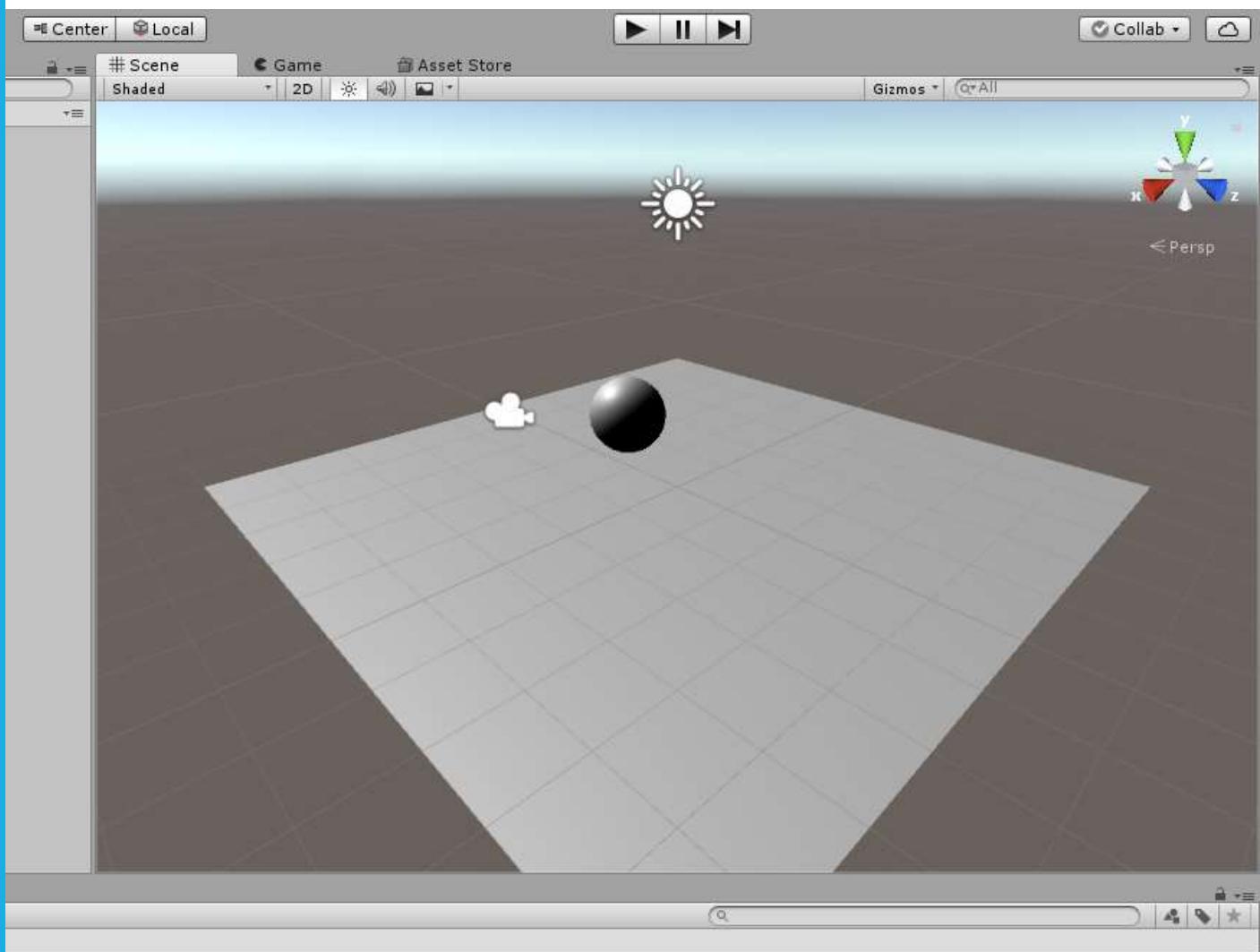
**Project Window:** Muestra assets de librería que están disponibles para ser usados en una o más Escenas. Son archivos físicos.

**Hierarchy Window:** Es una representación jerárquica de cada objeto en la escena. Podria decirse que son “las instancias” de lo que tenemos en el Project Window.

**Inspector Window:** Panel que muestra todas las propiedades del objeto actualmente seleccionado. Ya que diferentes objetos tienen diferentes propiedades, el layout (diseño) y contenido de la ventana del inspector puede variar.

## Escena (Scene)

Nuestro escenario virtual donde colocaremos los objetos visuales.

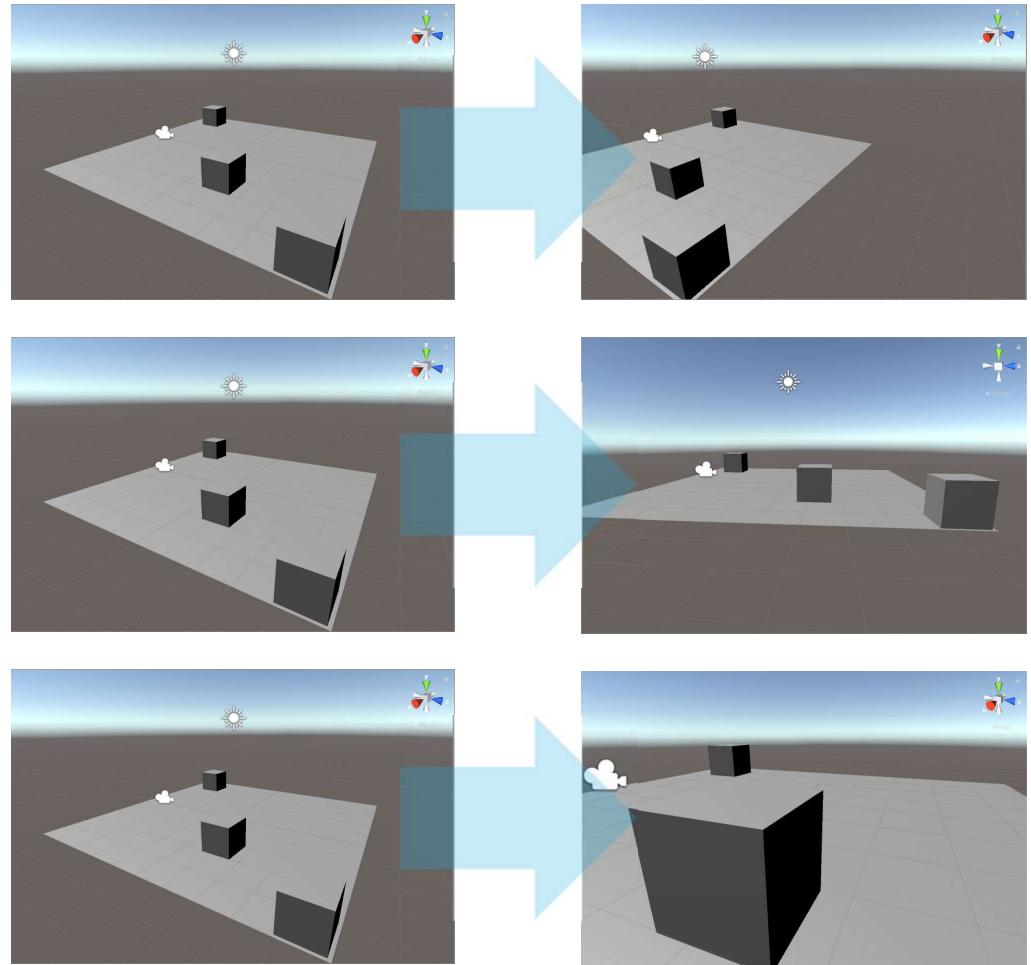


# Movimiento en Escena (Scene)

Paneo (Click Central)

Rotación (Click Izq. +  
Mov. Raton)

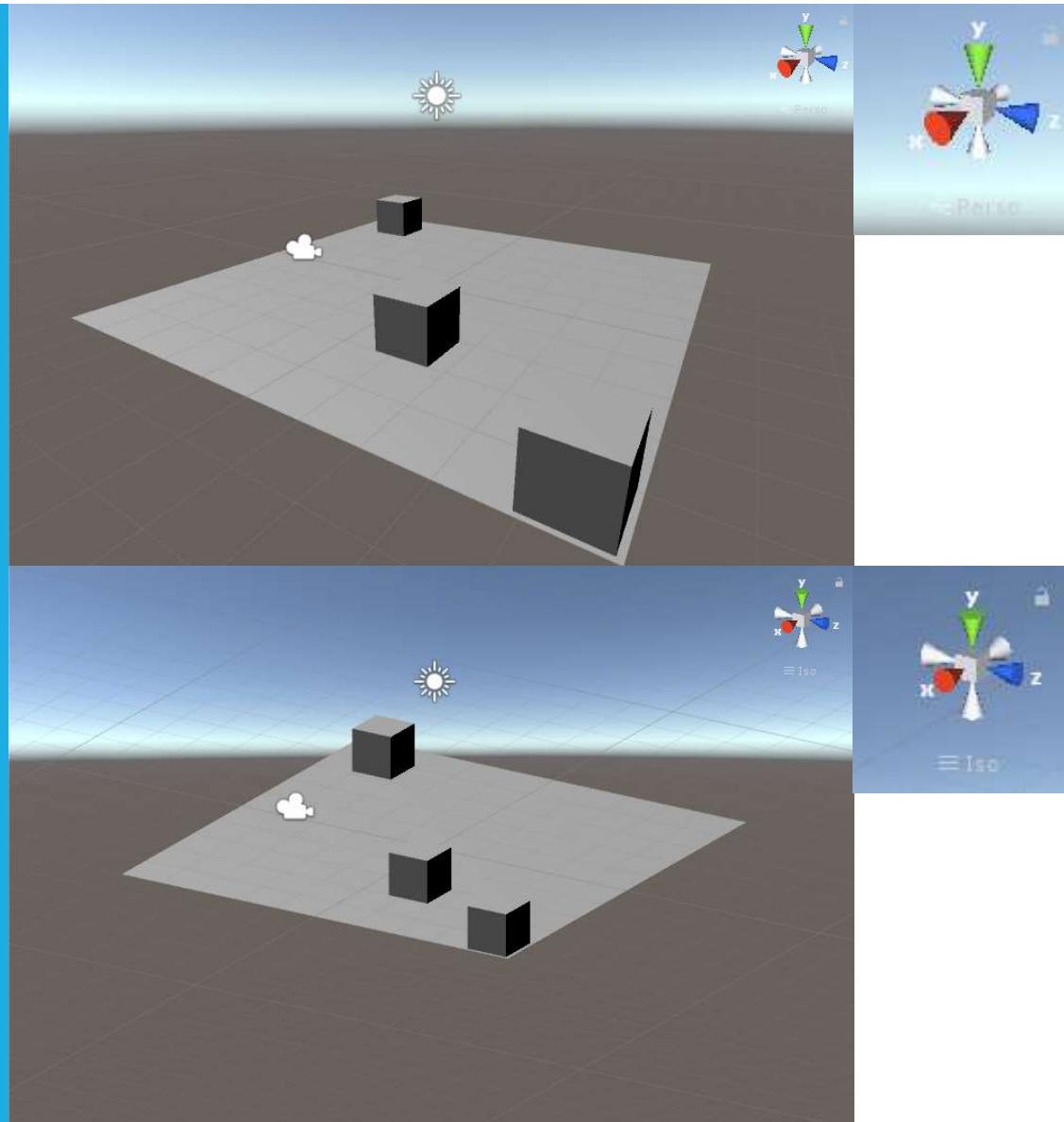
Zoom (Scroll del Raton,  
Arriba y Abajo)



# Escena (Scene)

## Vistas

- **Perspectiva:** Con noción de profundidad.
- **Isográfica:** Sin noción de profundidad.

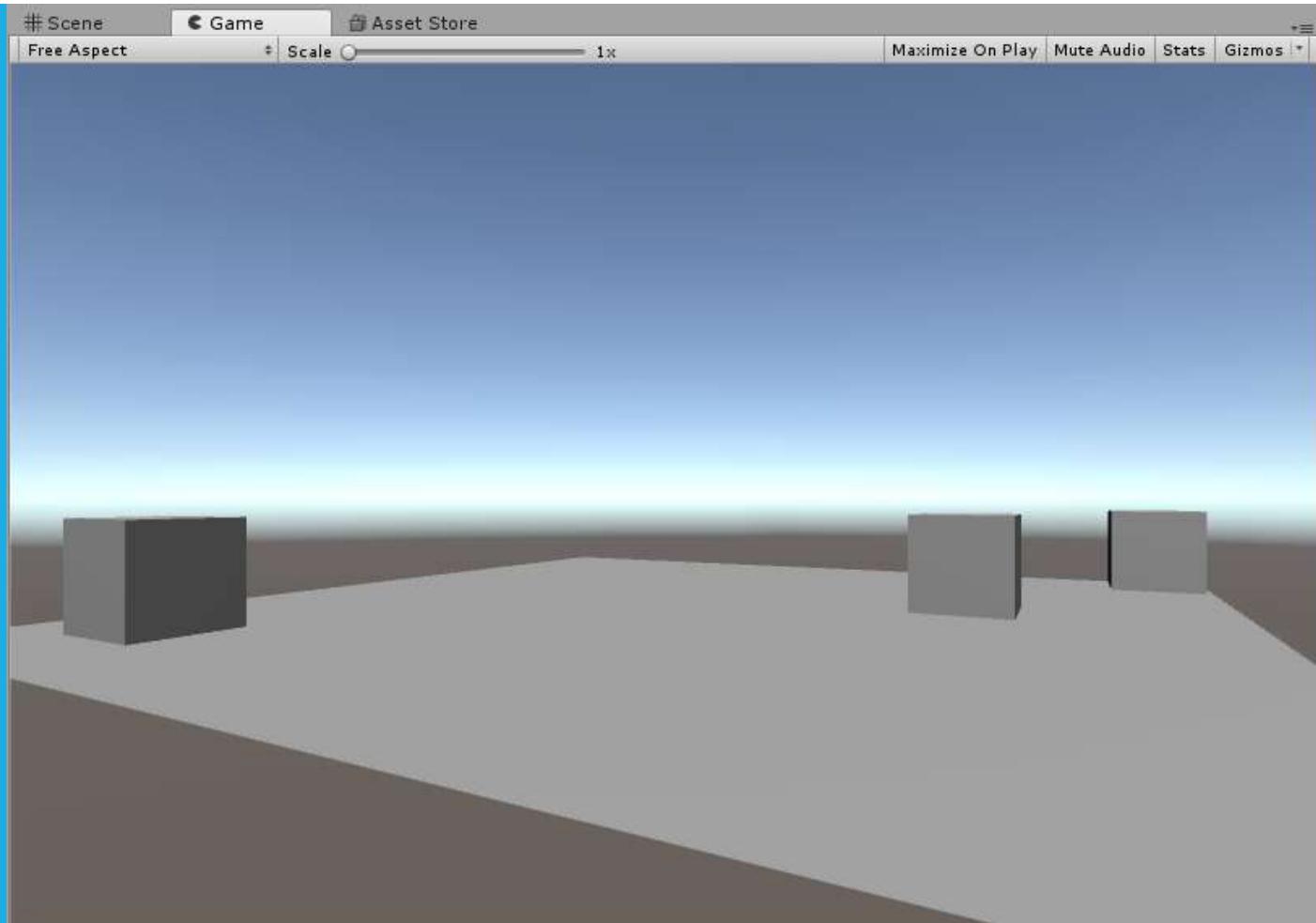


## Juego (Game)

Es el modo de “Ejecución” de nuestra aplicación.

En este modo podremos previsualizar como “corre” nuestra aplicación para probar.

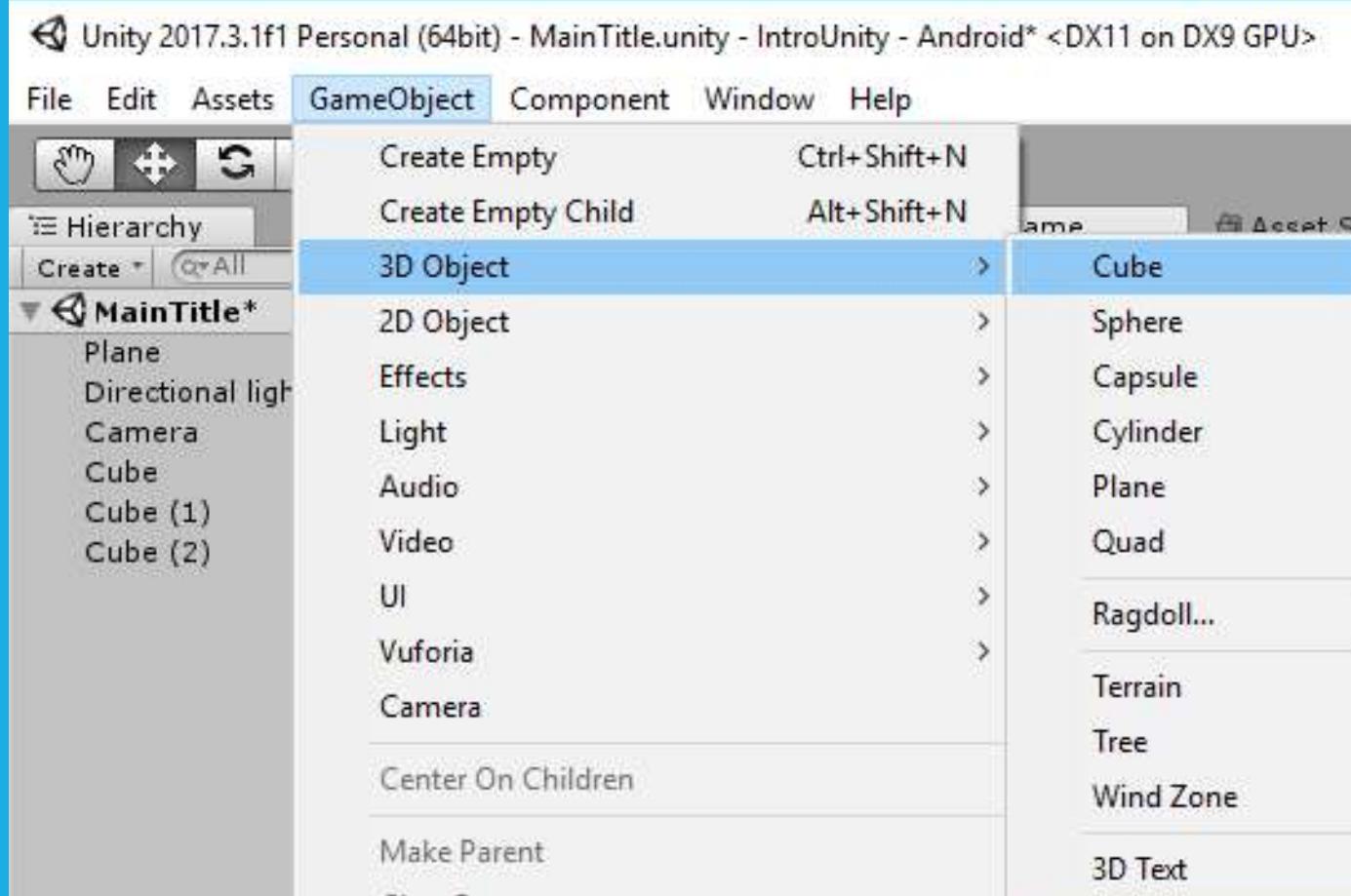
También se ejecutan los scripts y entran en juegos los hilos de ejecución en Unity.



# Game Objects

Los GameObjects son objetos fundamentales en Unity que representan personajes, props, y el escenario.

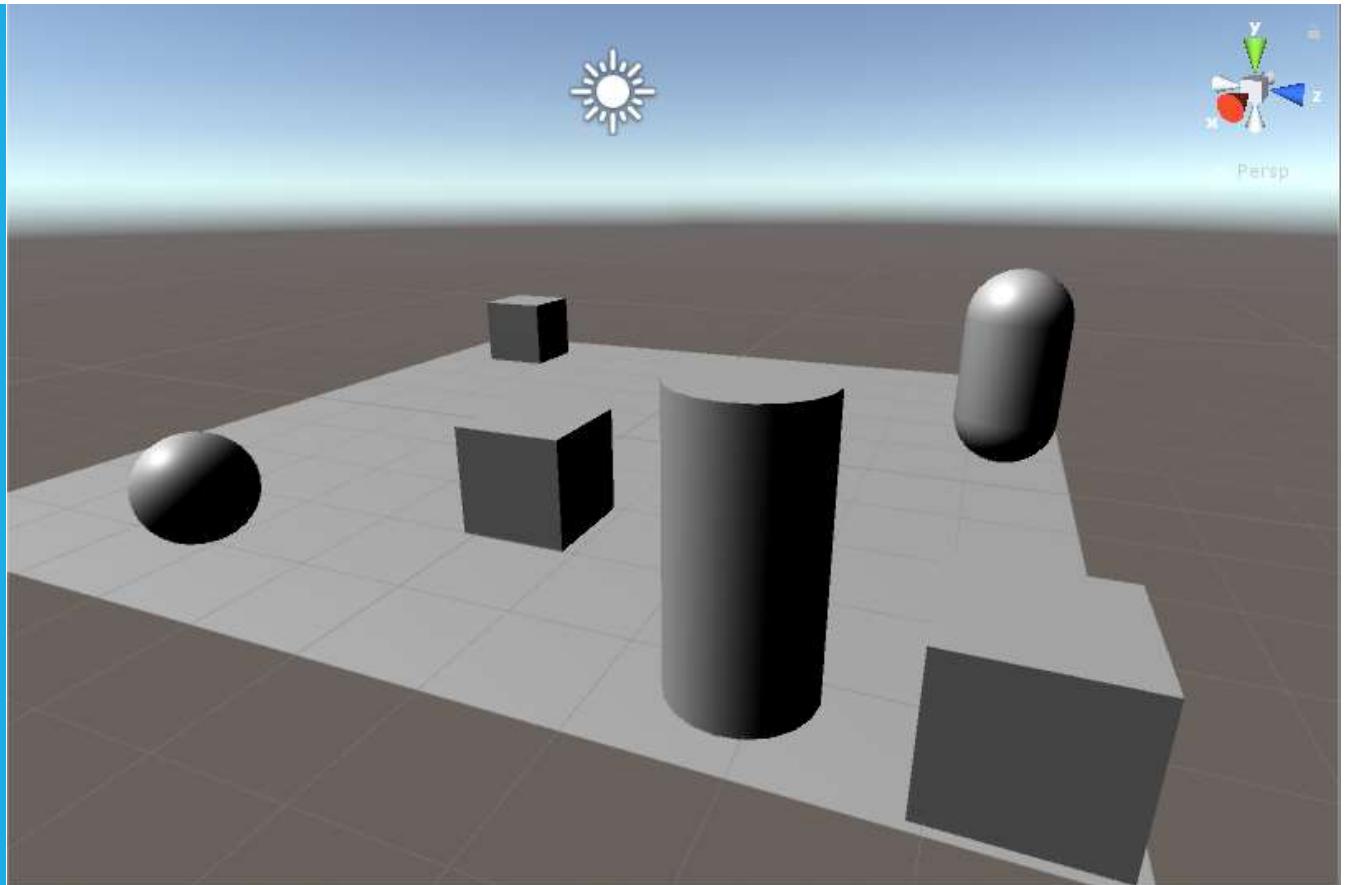
Estos no logran nada por sí mismos pero funcionan como contenedoras para Components, que implementan la verdadera funcionalidad.



# Game Objects (Primitivas 3D)

Son los objetos básicos tridimensionales en Unity.

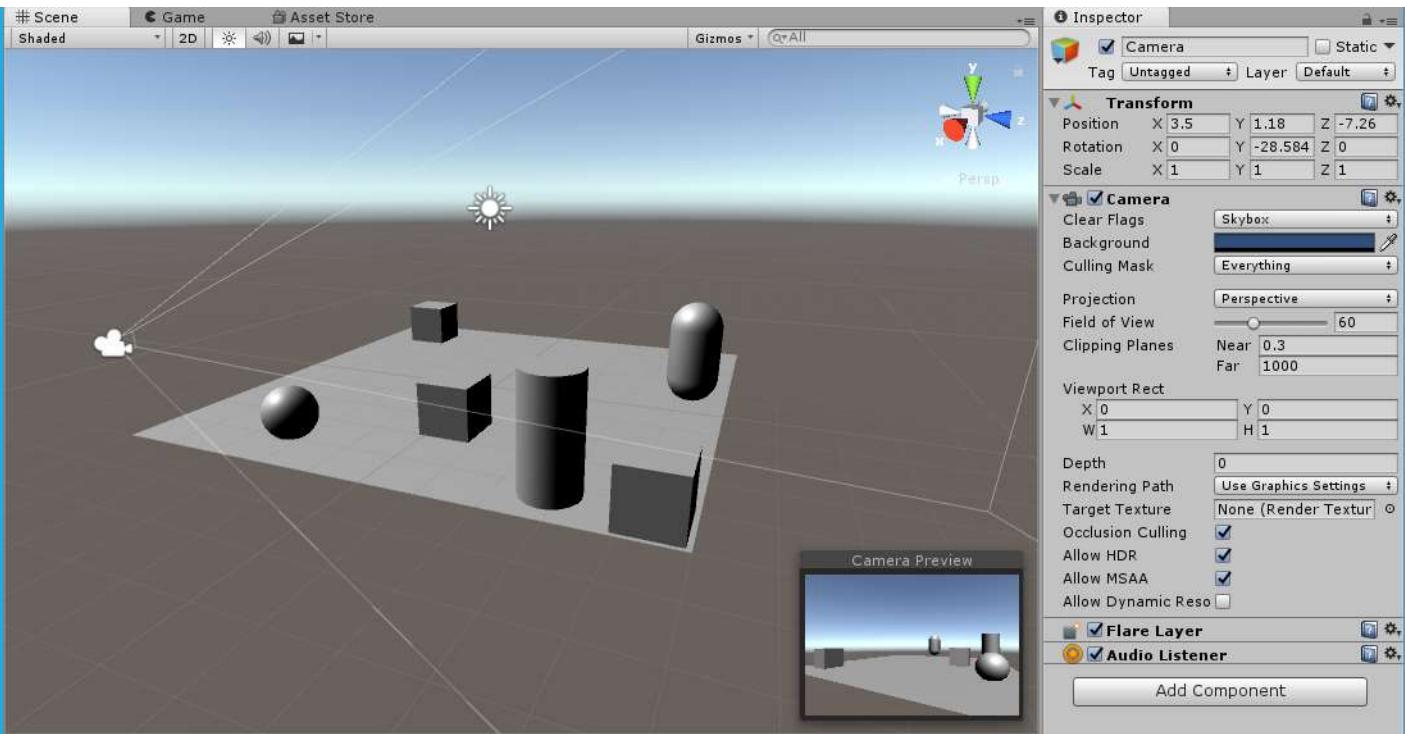
Contiene elementos como cubos, esferas, cilindros, capsulas, planos, etc.



# Camera

Sirven para mostrar el mundo que hemos creado al espectador o jugador.

Siempre debemos tener al menos una cámara. Aunque podemos tener más de una e ir alternando vistas o mostrando 2 diferentes (ejemplo: pantalla compartida en juegos cooperativos).



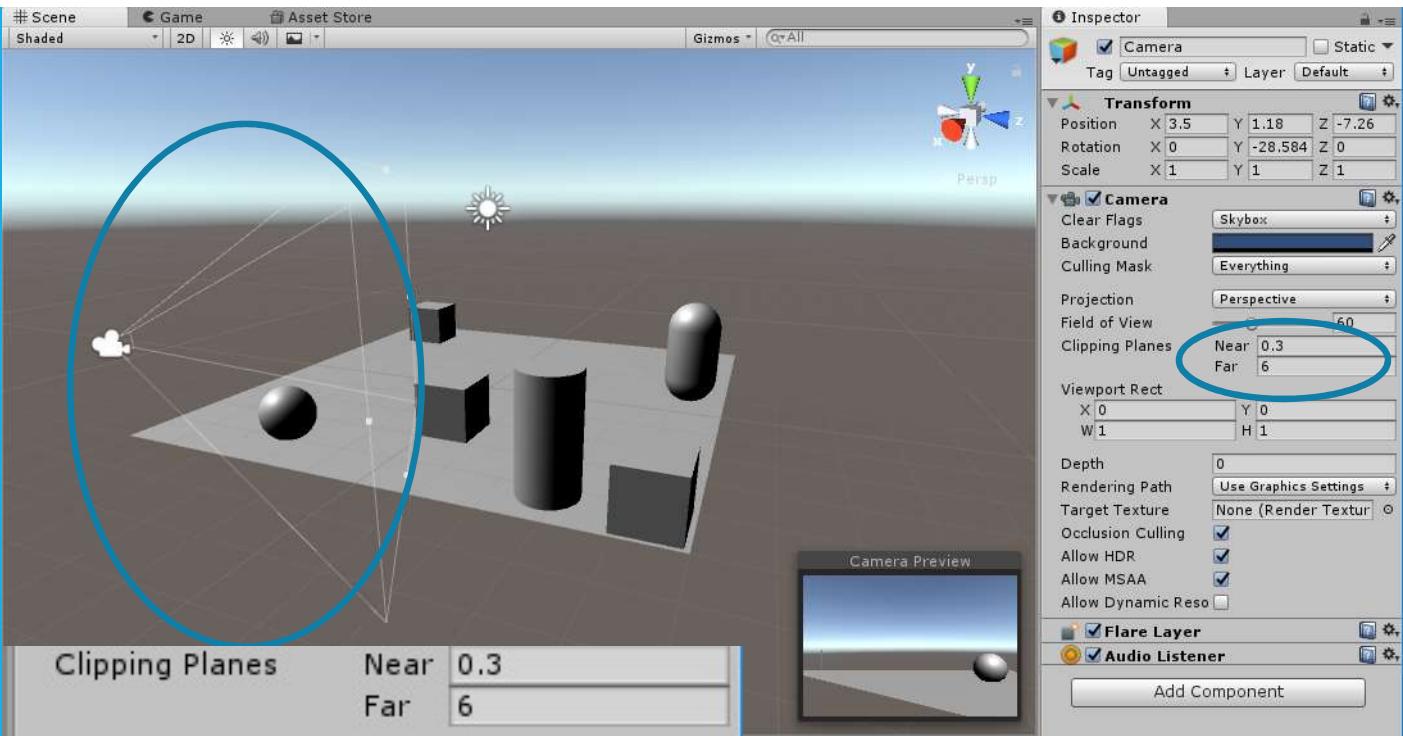
# Clipping Planes en Camaras

Los Clipping Planes en las Camaras tienen que ver con la Distancia de Visión.

En Near configuramos que tan cerca puede estar un objeto de la cámara antes que desaparezca.

En Far configuramos que tan lejos puede ver objetos nuestra cámara.

La correcta configuración de los Clipping Planes influyen tanto en la experiencia de juego/usuario así como en el rendimiento del mismo.



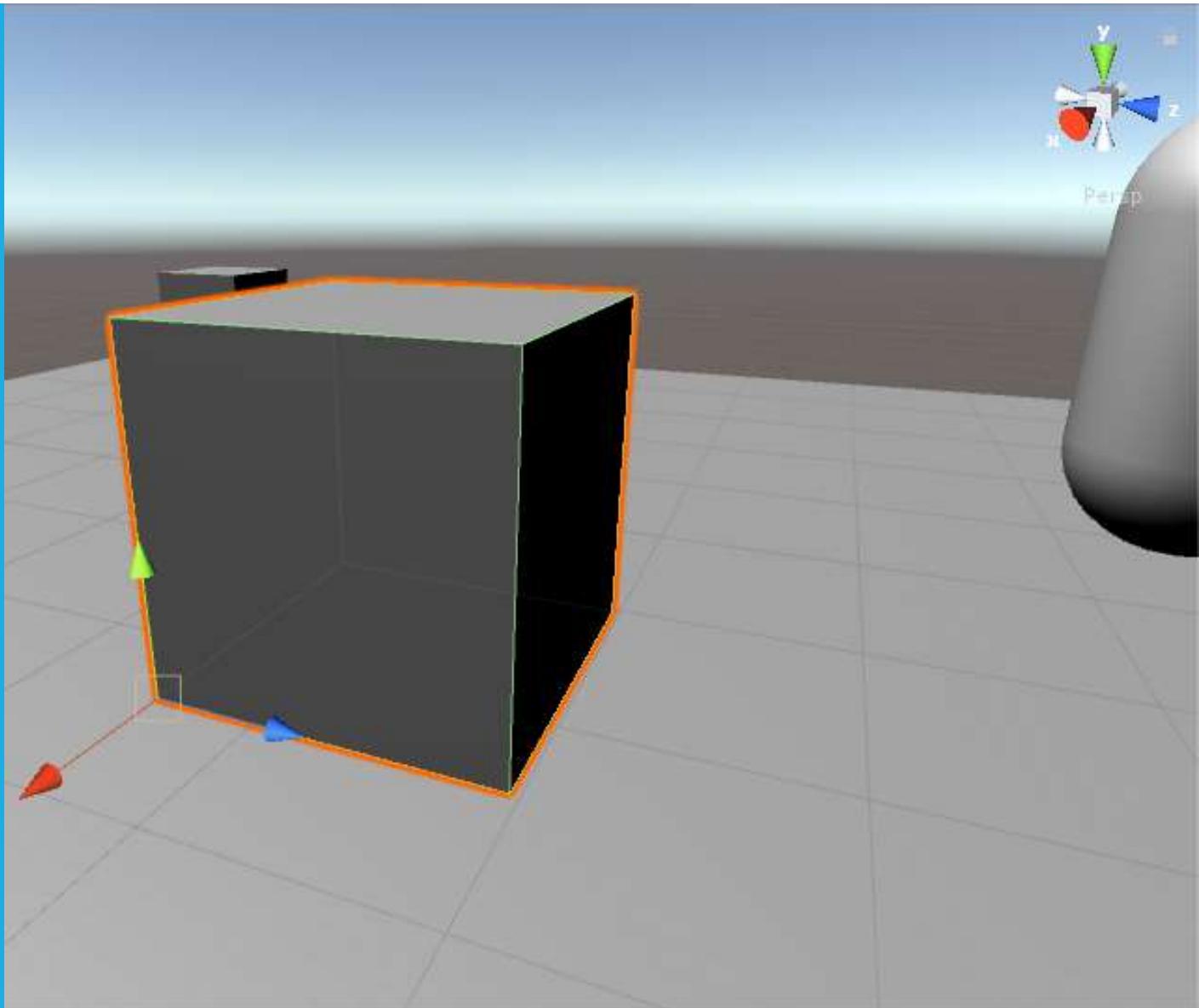
# Pivotes

Manteniendo pulsada la tecla V movemos temporalmente el Pivote.

Los pivotes de los objetos se suelen colocar en la parte de abajo (piso)

Los pivotes se definen desde el programa de modelado, no directamente en Unity.

Si queremos ajustar el Pivote en Unity, es necesario colocar el objeto dentro de un Empty (contenedor) y manipularlo desde el contenedor.

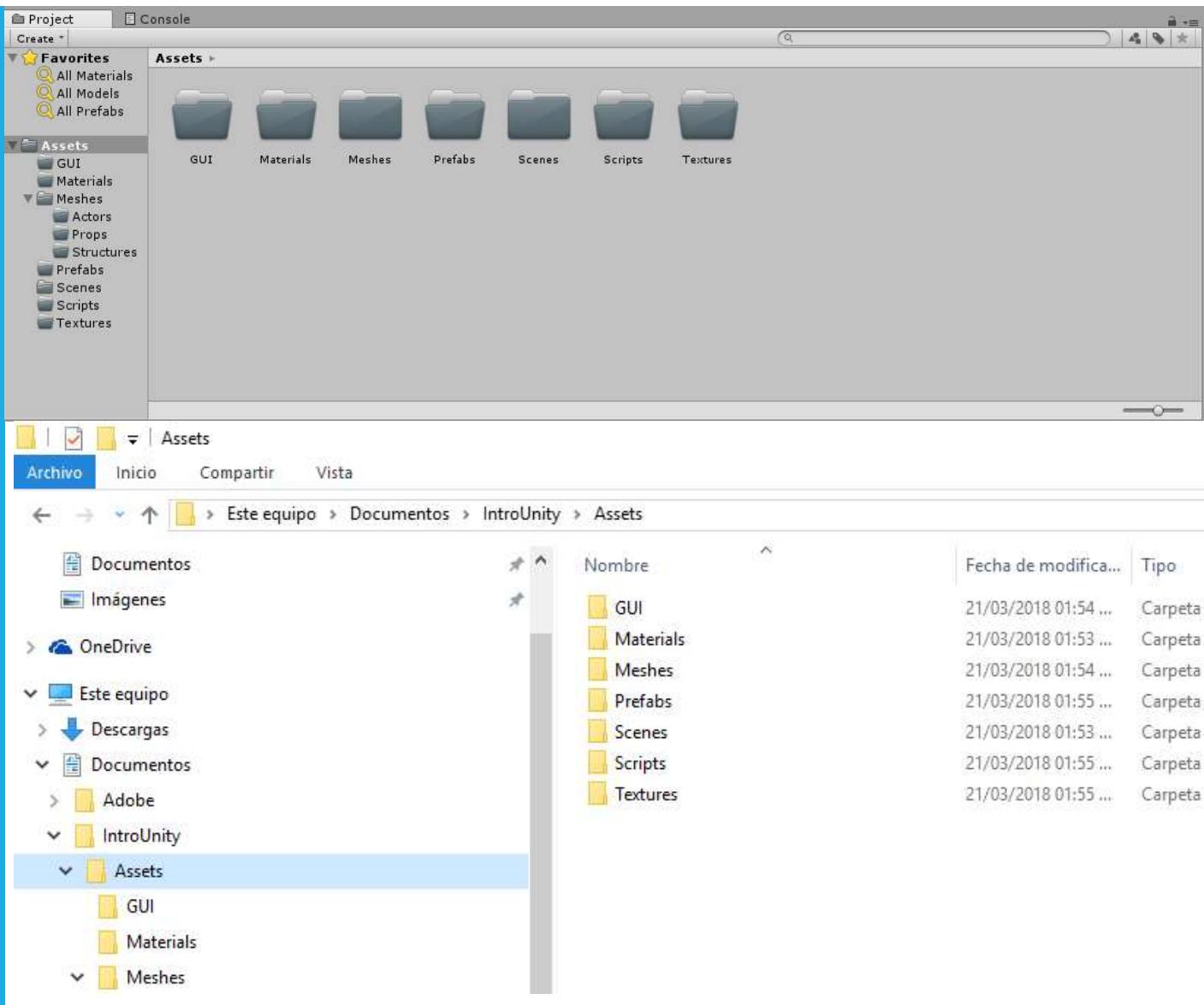


# Sección Proyecto (Project)

Contiene toda la información “física” de nuestro proyecto.

**PRECAUCIÓN: SI  
BORRAMOS UN ASSET, LO  
BORRAMOS DEL DISCO  
DURO.**

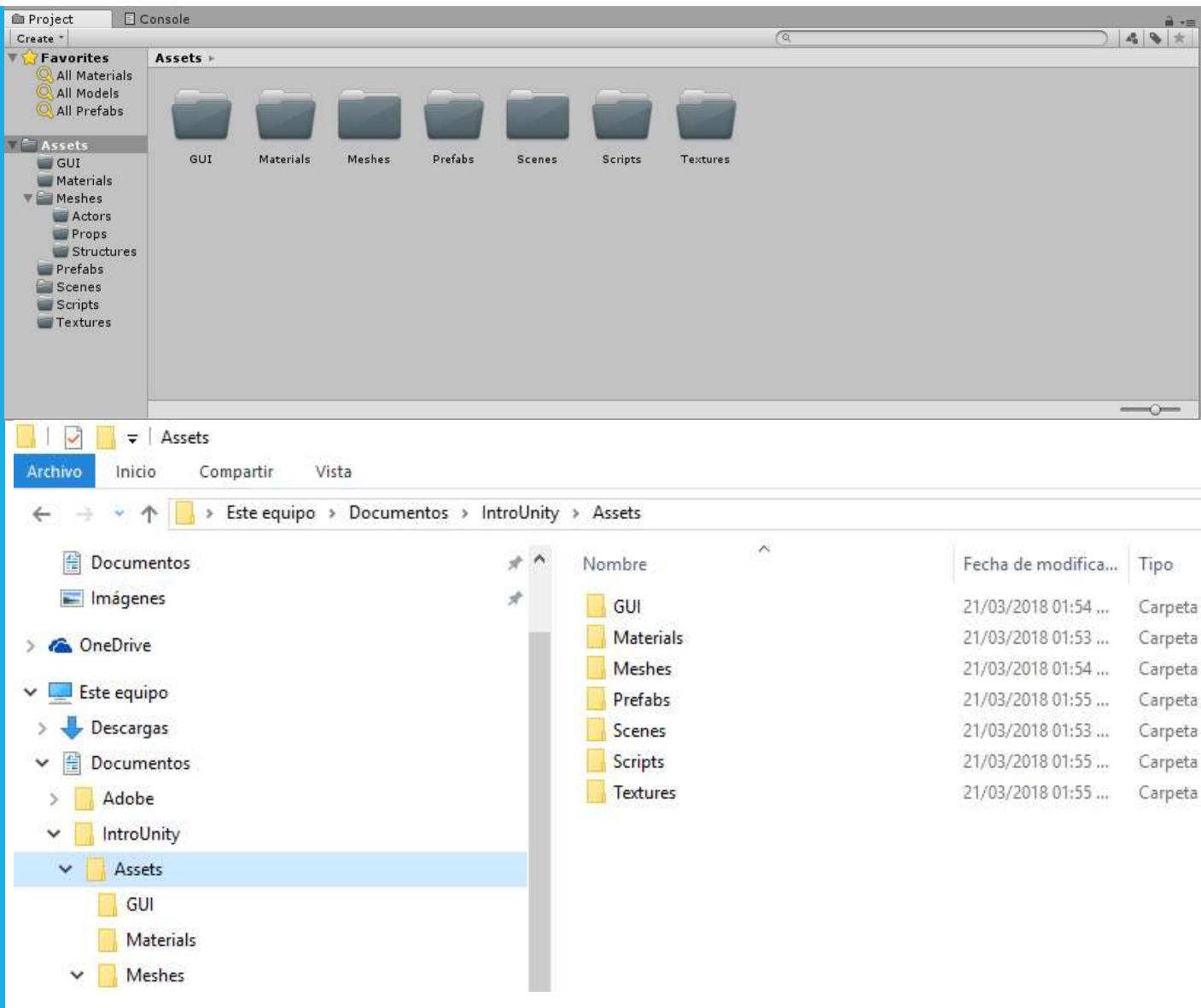
Por defecto todos los elementos se colocan dentro de la carpeta “Assets”.



# Sección Proyecto (Project)

La información de Project  
puede ser utilizado en uno o  
más escenarios.

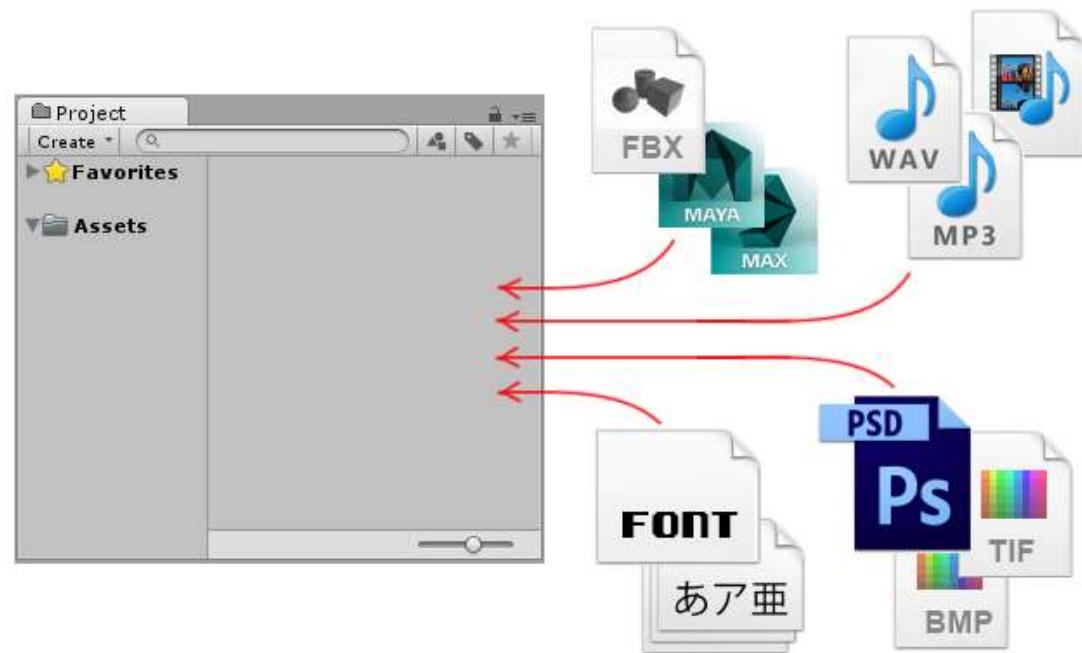
Es recomendable usar  
carpetas y archivos con  
nombres en inglés y  
convención CamelCase.



## Assets

Un asset es una representación de cualquier item que puede ser utilizado en un proyecto de Unity.

Un asset podría venir de un archivo creado afuera de Unity, tal como un modelo 3D, un archivo de audio, una imagen, o cualquiera de los otros tipos de archivos que Unity soporta.

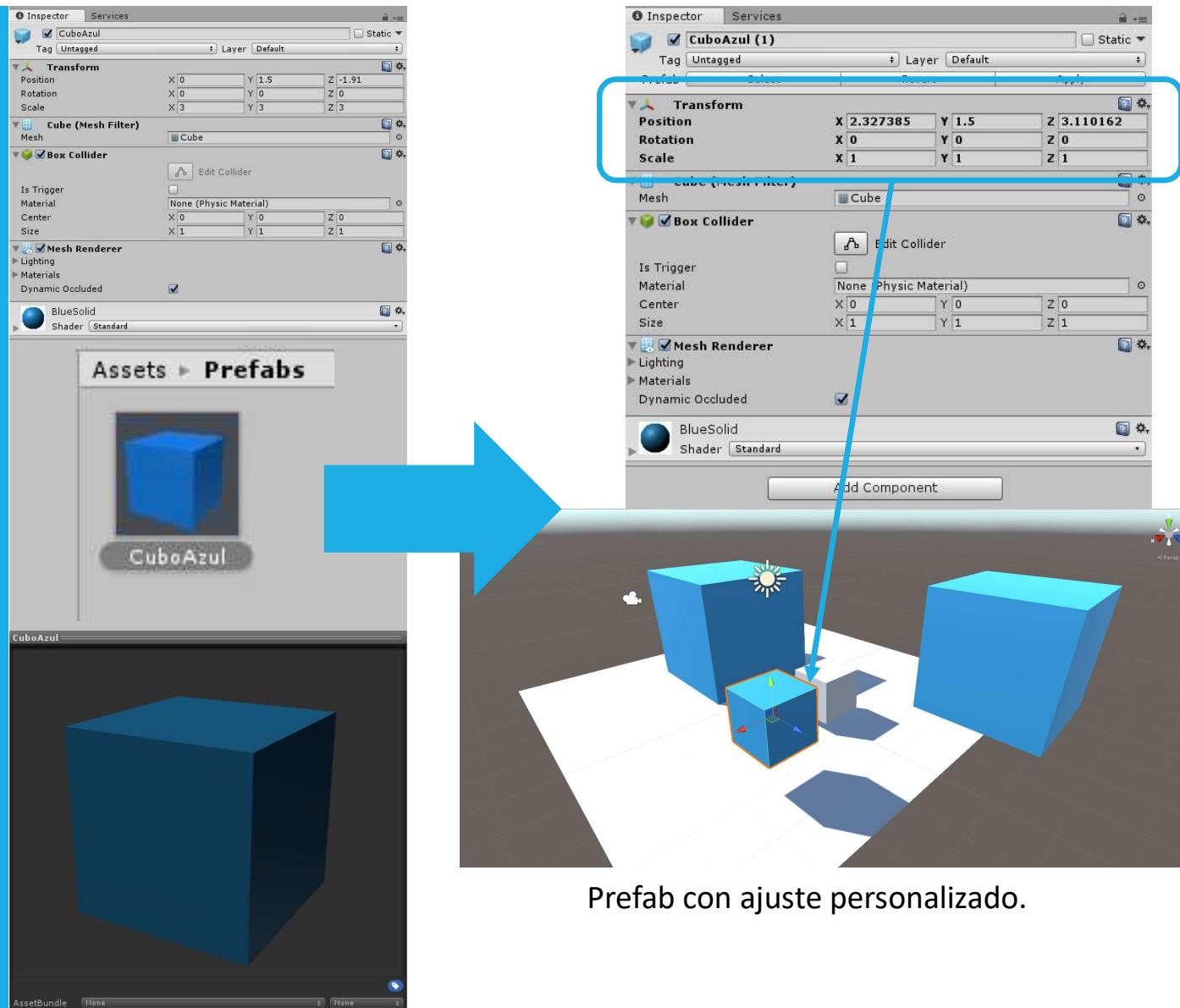


# Prefabs

Es un asset que almacena un objeto GameObject con componentes y propiedades.

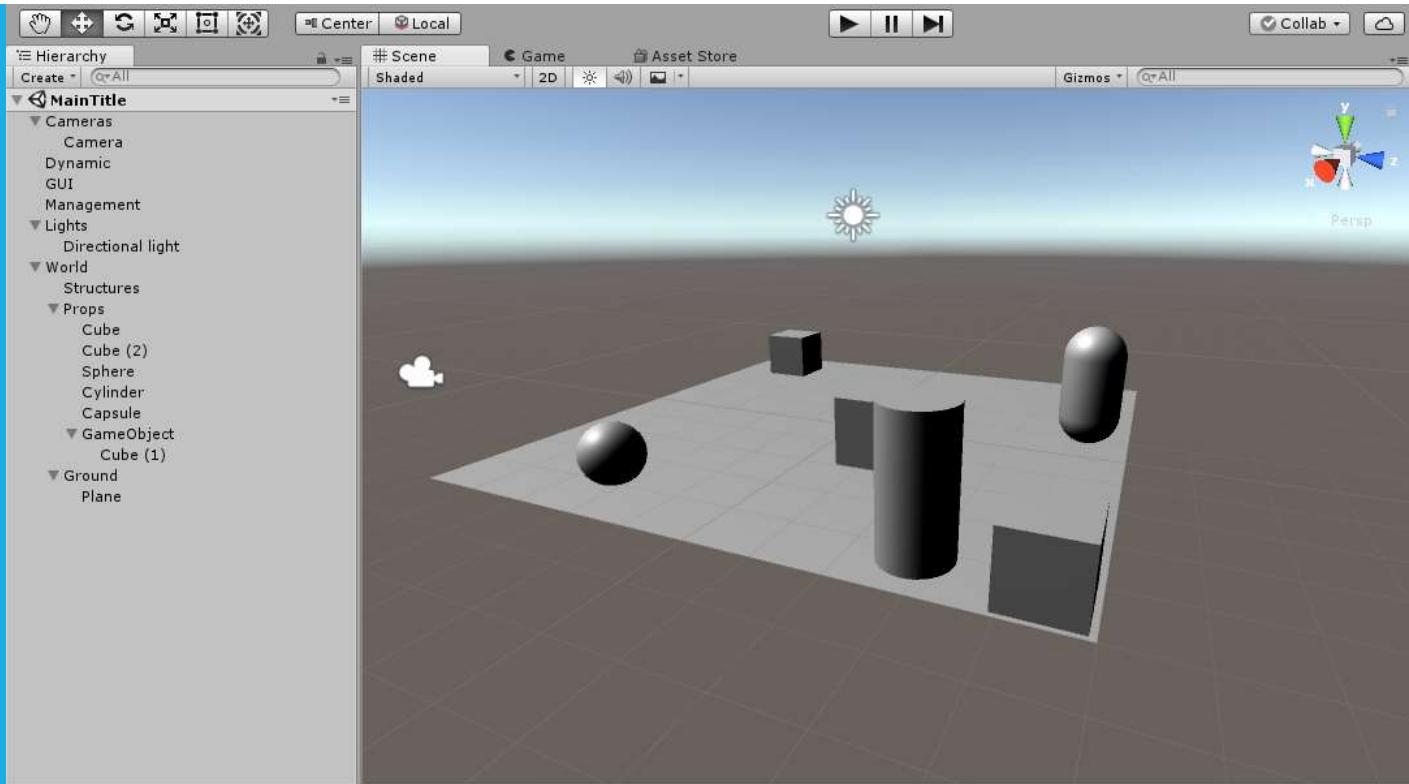
El prefab actúa como una plantilla a partir de la cual se pueden crear nuevas instancias del objeto en la escena.

Cualquier edición hecha a un prefab asset será inmediatamente reflejado en todas las instancias producidas por él, pero, también se puede anular componentes y ajustes para cada instancia individualmente.



# Jerarquía (Hierarchy)

Es donde podremos visualizar en formato de texto todos los elementos presentes en nuestro Escenario actual.



# Sección Jerarquía (Hierarchy)

Una buena práctica consiste en tener una jerarquía estandarizada y entendible para organizar nuestra jerarquía.

Hacerlo nos permitirá trabajar mejor con proyectos complejos.

Adicionalmente nos permitirá compartir el proyecto y que sea más fácilmente entendible por otra persona.

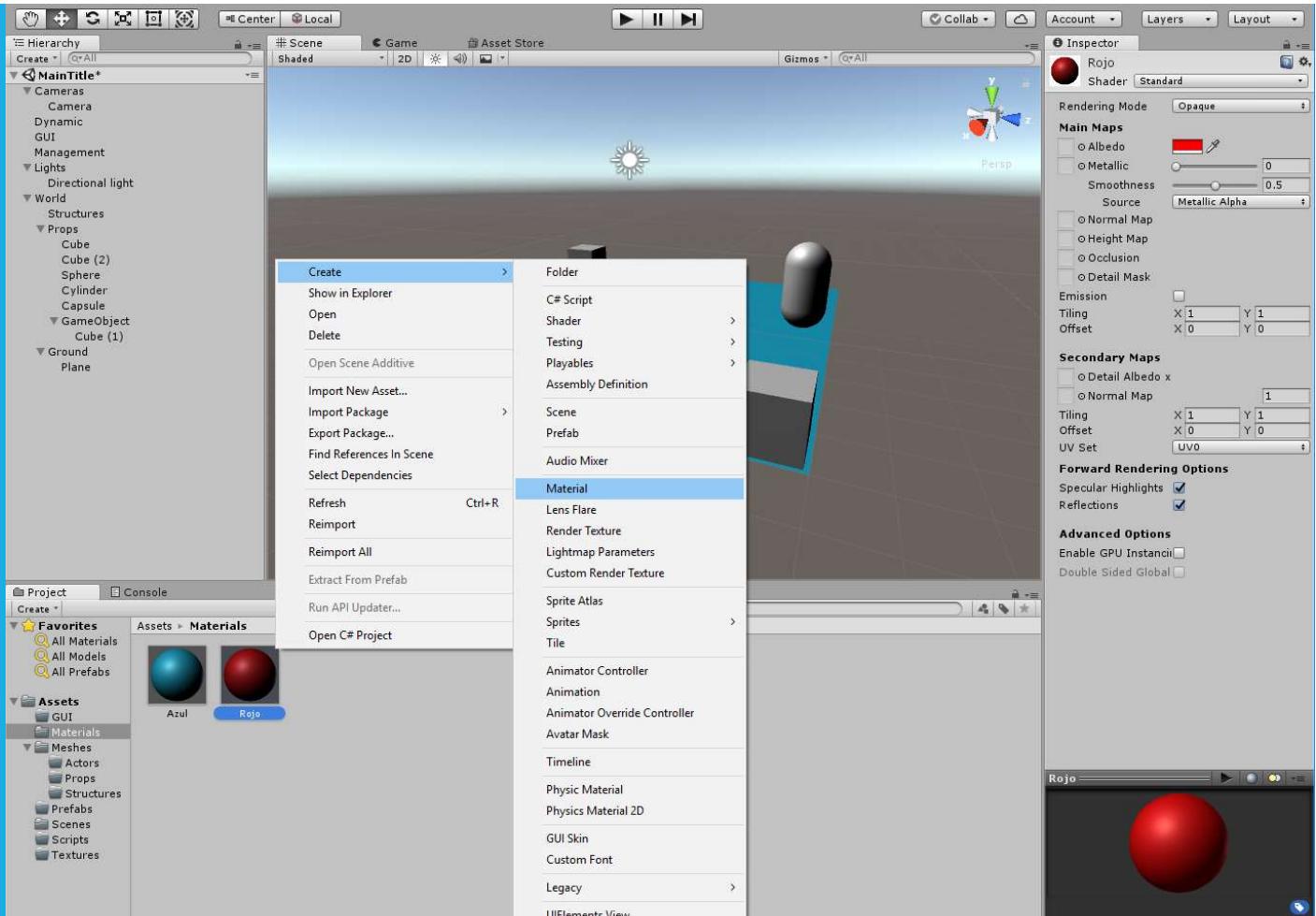


# Materiales (Materials)

Los materiales contienen la información de color/textura e iluminación.

Pueden ser reutilizados y reaplicados a varios objetos en el proyecto.

Para crearlos, en nuestra ventana de proyecto, hacer clic derecho, seleccionar la opción Create y posteriormente Material.

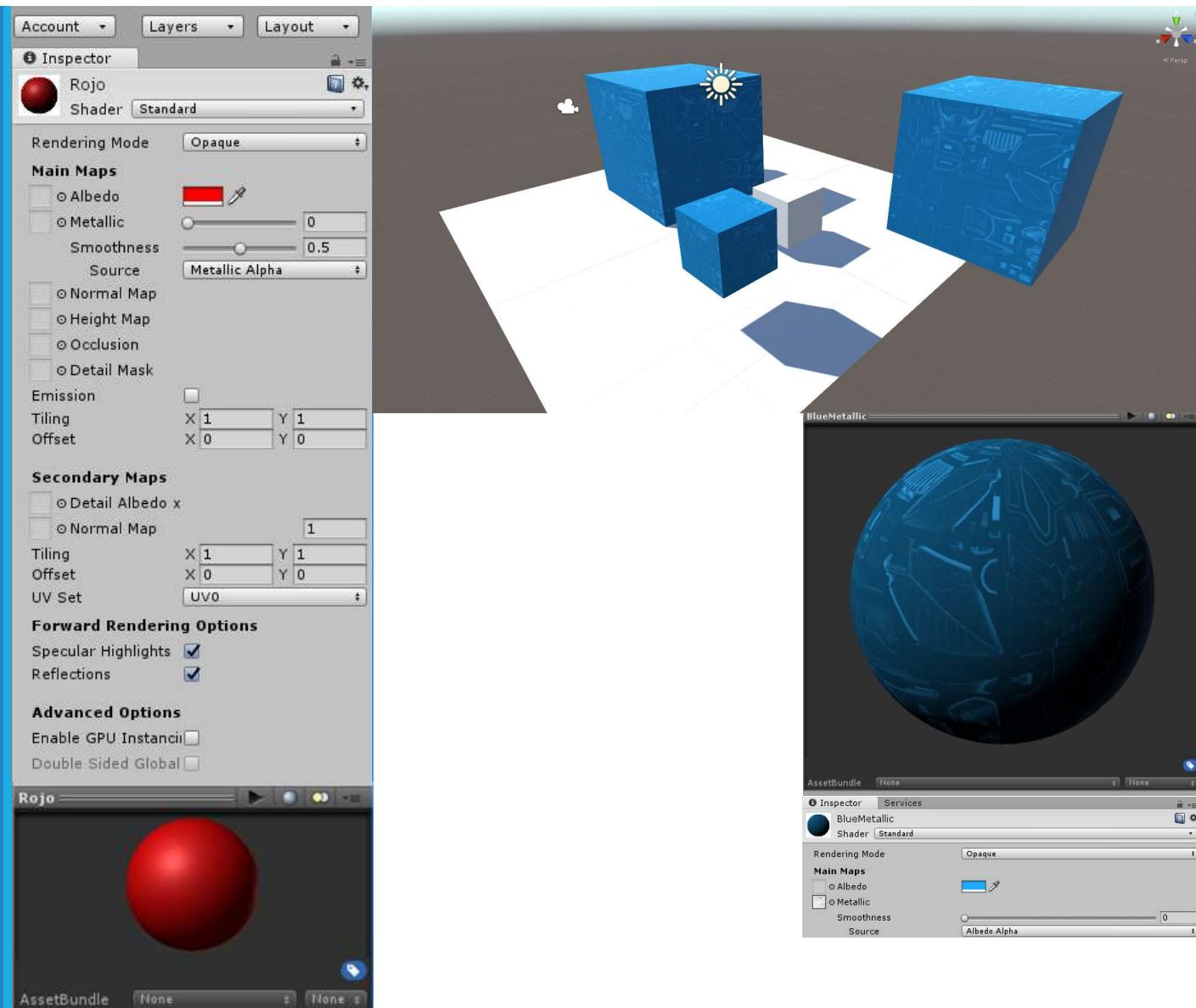


# Materiales (Materials)

En un Material de Shader Standard:

Albedo nos dará el color del material o bien la textura base.

Metallic indicará la reflectividad de la superficie y puede tomar el color del Albedo.



# Illuminación (Illumination)

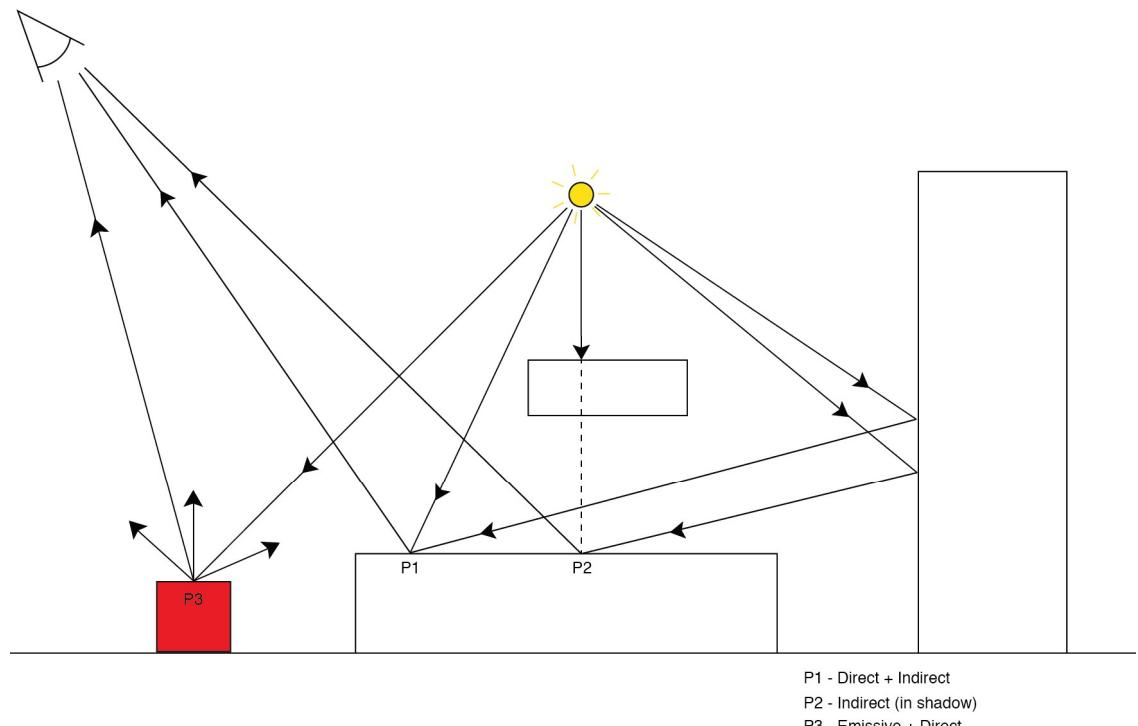
## Terminología

Surface: Todos los triangulos que component todas las mallas en la escena, son llamadas Superficie (surface).

Emitted Light: Es la luz emitida directamente en la superficie de la escena.

Direct Light: Es la luz emitida que llega a una superficie de la escena y es reflejada hacia un sensor (ejemplo: la retina del ojo o una camara).

Indirect Light: Es una luz emitida, que llega a una superficie de la escena al menos 2 veces y es reflejada al sensor.

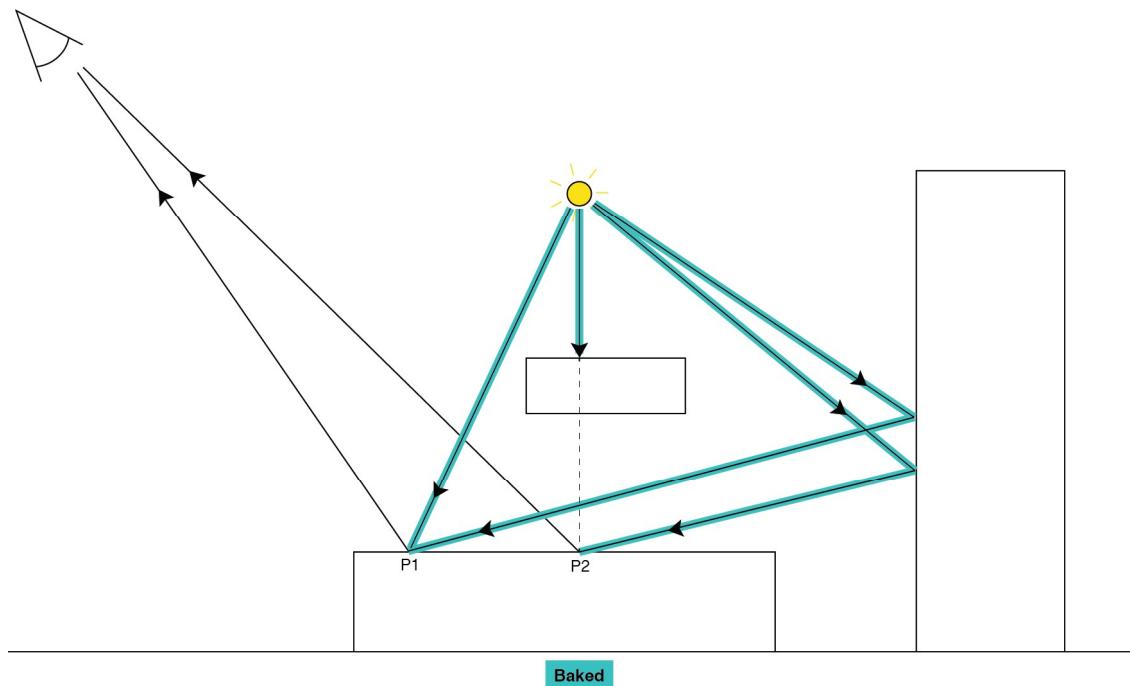


# Iluminación (Illumination)

La iluminación puede darse en 2 modos:

**Realtime:** Iluminación que se genera en tiempo real en ejecución de la aplicación.

**Baked:** Se precalcula la iluminación antes de la ejecución en tiempo real. Útil para mejorar el rendimiento e iluminar ciertas regiones. Como desventaja no genera sombras en tiempo real ni tiene specular.



# Illuminación (Illumination)

Existen 4 tipos fundamentales de iluminación:



Directional



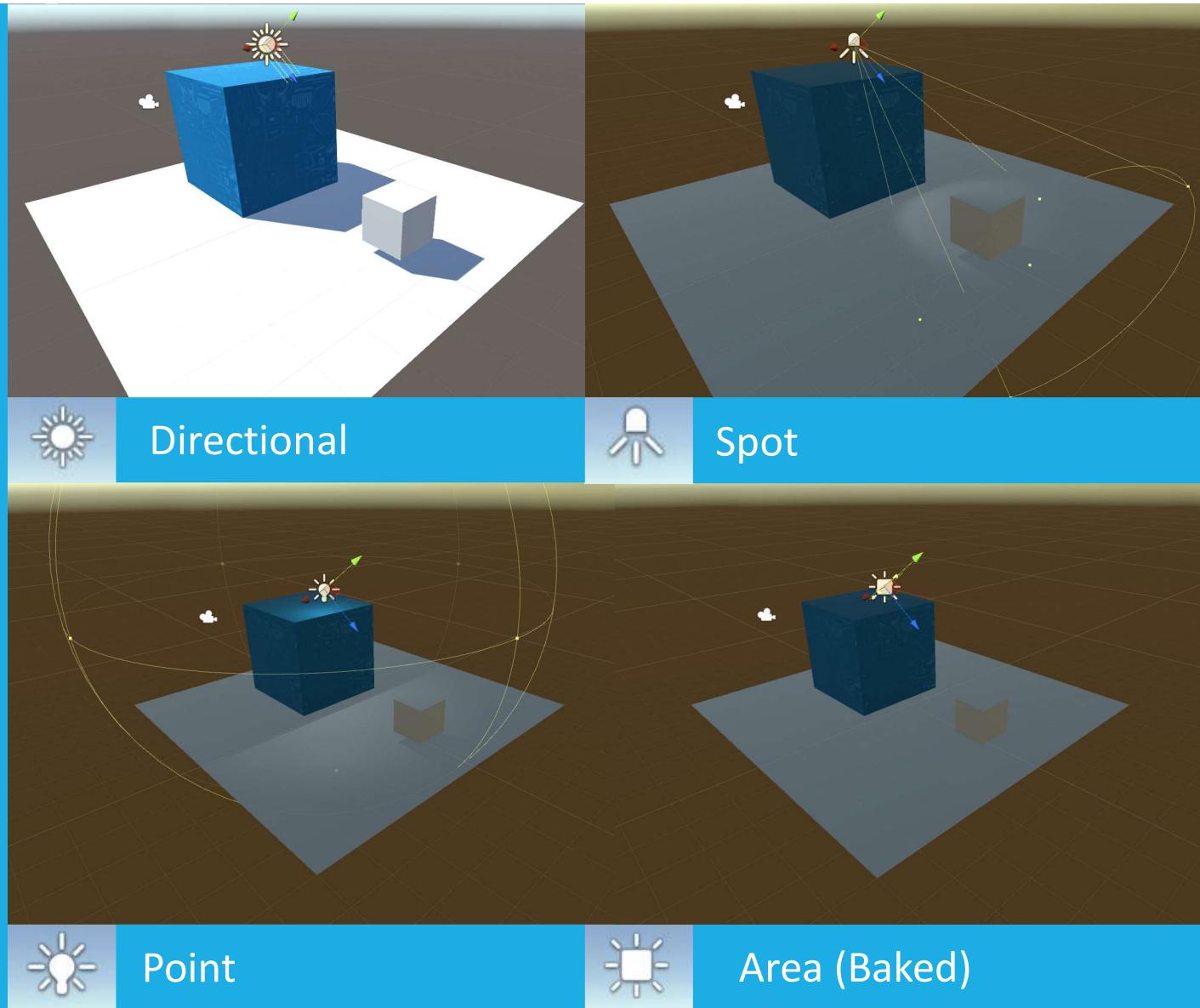
Spot



Point



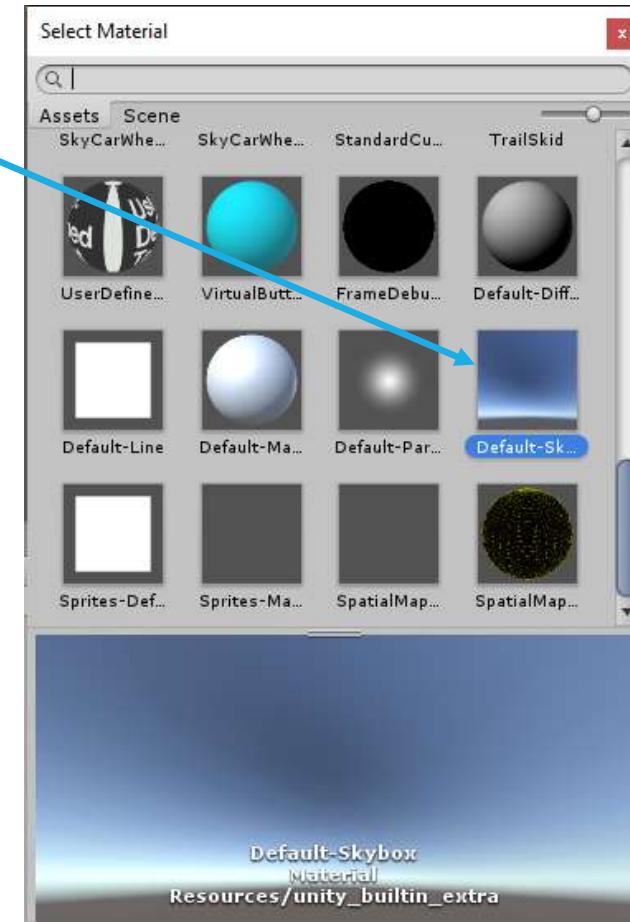
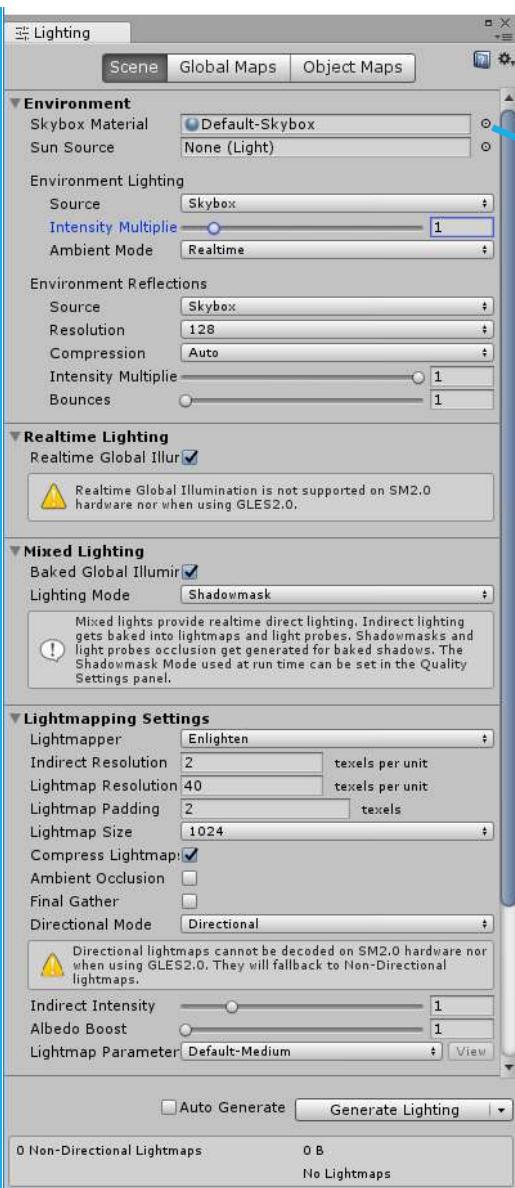
Área (Baked)



# Illuminación (Illumination)

Podemos modificar el Material del cielo (e información de iluminación) de la escena desde la ventana:

Window -> Lighting ->  
Settings





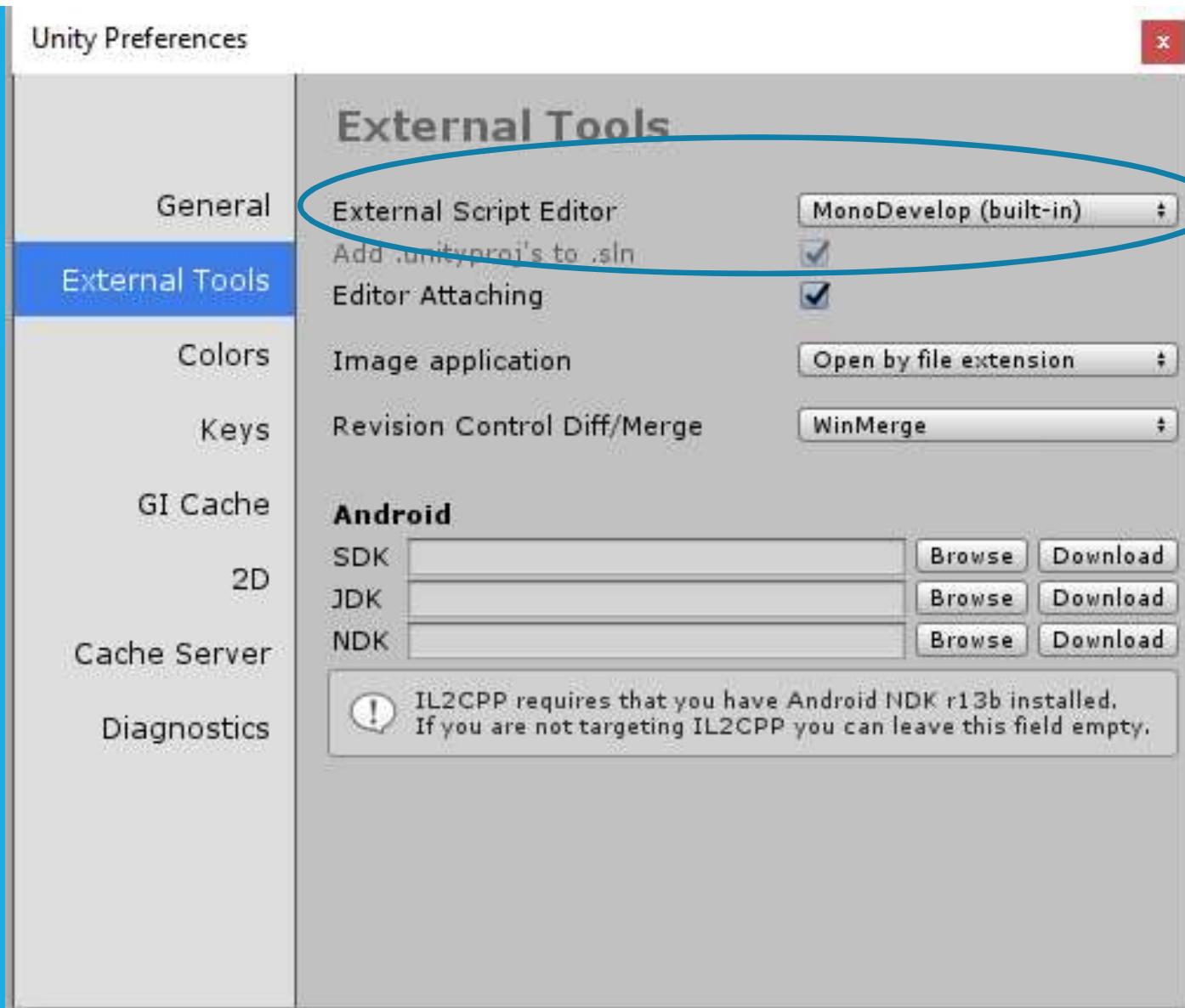
# Programación con C#

# Cambiando el editor por defecto

Es recomendable usar MonoDevelop como editor de código por defecto para Unity.

Para cambiar este comportamiento, hay que ir al menú Edit, luego a Preferences.

En la ventana de Unity Preferences, ir a la sección de External Tools y elegir MonoDevelop (built-in) en la opción de External Script Editor.



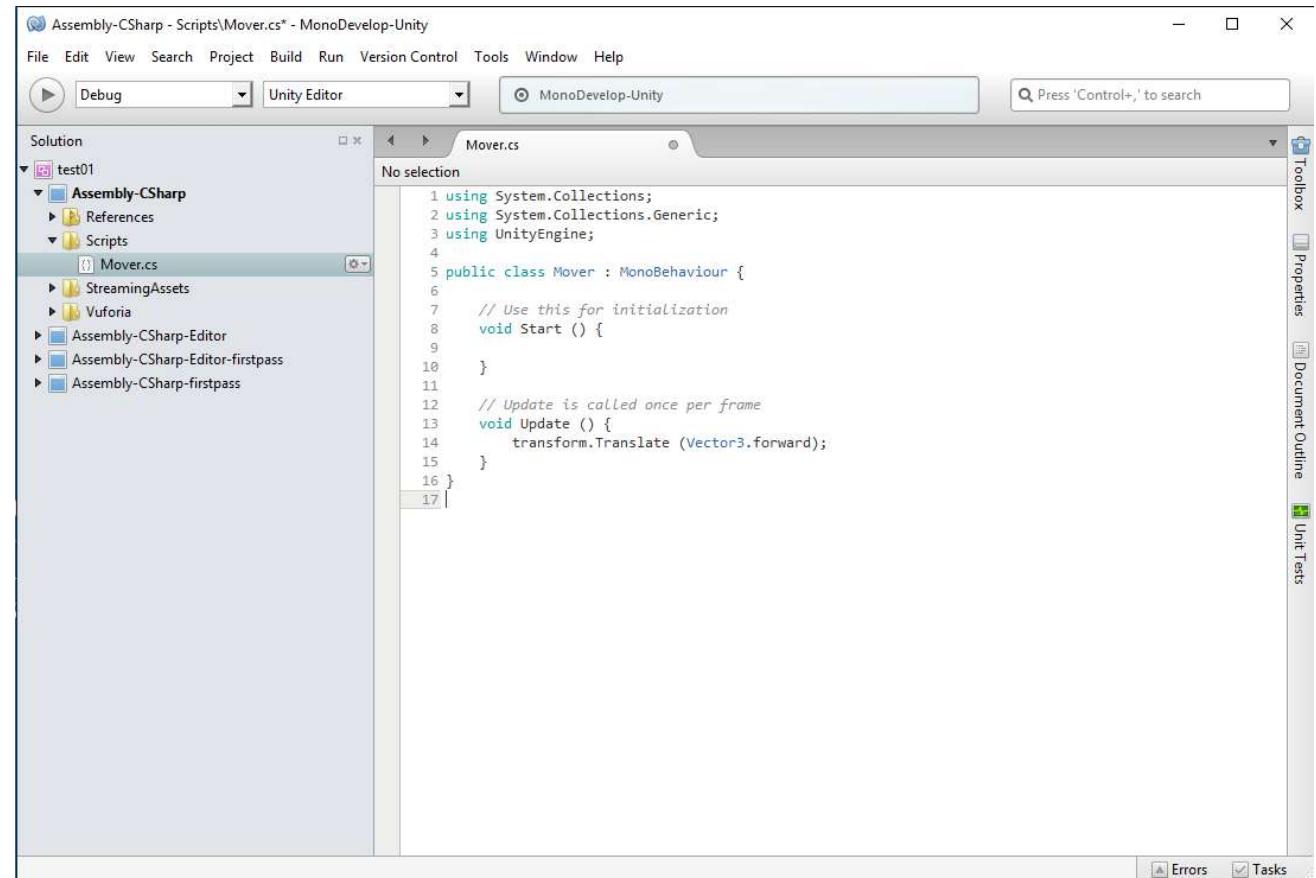
# MonoDevelop

MonoDevelop es un entorno de desarrollo integrado (IDE) libre y gratuito.

Nos permite trabajar con el lenguaje C#, que es el por defecto usado en Unity.

Viene incluido dentro de las opciones a instalar en Unity.

Resulta más ligero y práctico que utilizar Visual Studio.



The screenshot shows the MonoDevelop IDE interface. The title bar reads "Assembly-CSharp - Scripts\Mover.cs\* - MonoDevelop-Unity". The menu bar includes File, Edit, View, Search, Project, Build, Run, Version Control, Tools, Window, Help. The toolbar has a "Debug" button and a "Unity Editor" dropdown set to "MonoDevelop-Unity". A search bar says "Press 'Control+, to search". The left sidebar is the "Solution" browser, showing a project named "test01" with a "Assembly-CSharp" folder containing "References", "StreamingAssets", "Vuforia", and three "Assembly-CSharp-Editor" files. Inside "Assembly-CSharp", there's a "Scripts" folder with "Mover.cs" selected. The main editor window shows the code for "Mover.cs":

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Mover : MonoBehaviour {
6
7     // Use this for initialization
8     void Start () {
9
10    }
11
12     // Update is called once per frame
13     void Update () {
14         transform.Translate (Vector3.forward);
15     }
16 }
17 }
```

The status bar at the bottom right shows "Errors" and "Tasks".

# Script Vacío

Cuando en Unity creamos un nuevo Script, el archivo viene con un contenido base como el siguiente.

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

public class MyFirstScript : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

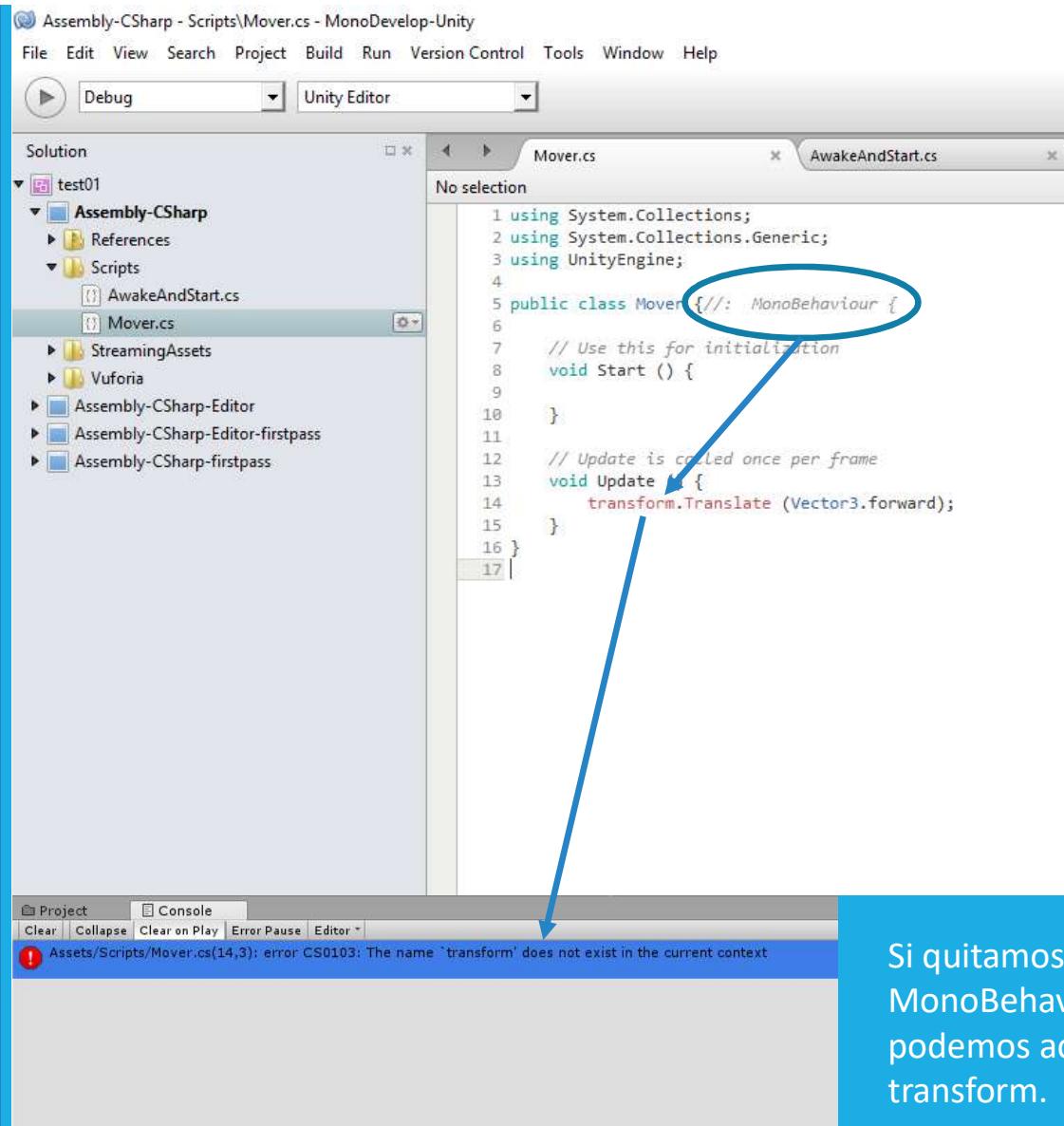
    // Update is called once per frame
    void Update () {

    }
}
```

# :MonoBehavior

La clase **MonoBehaviour** es **una** de las principales de las API de Unity.

Todos los scripts vinculados a objetos (en el caso de Unity son **GameObjects**) deben derivar de esta clase para poder aprovechar eventos y funciones útiles para la manipulación de elementos de nuestro proyecto.



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Mover : MonoBehaviour {
6
7     // Use this for initialization
8     void Start () {
9
10    }
11
12     // Update is called once per frame
13     void Update () {
14         transform.Translate (Vector3.forward);
15    }
16 }
17
```

The screenshot shows the MonoDevelop-Unity interface. The project tree on the left shows a solution named 'test01' with a 'Assembly-CSharp' folder containing 'References', 'Scripts' (with 'AwakeAndStart.cs' and 'Mover.cs'), 'StreamingAssets', 'Vuforia', and several 'Assembly-CSharp-...' files. The 'Mover.cs' file is selected in the editor. The code editor window displays the following C# script:

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Mover : MonoBehaviour {
6
7     // Use this for initialization
8     void Start () {
9
10    }
11
12     // Update is called once per frame
13     void Update () {
14         transform.Translate (Vector3.forward);
15    }
16 }
17
```

A blue oval highlights the line 'public class Mover : MonoBehaviour {'. A blue arrow points from this oval to the error message in the console window at the bottom. The console window has tabs for 'Project' and 'Console'. It contains the following message:

Assets/Scripts/Mover.cs(14,3): error CS0103: The name 'transform' does not exist in the current context

Si quitamos **MonoBehavior**, ya no podemos acceder a **transform**.

# Hilos de Ejecución en Unity

Principales:

- Start
- Update
- Awake

*// Se ejecuta una vez. Ocurre en el frame en que el script es habilitado. Se ejecuta ANTES de cualquier UPDATE. Útil para inicializar variables.*

```
void Start () {  
}
```

*// Se ejecuta cada frame. Esto ocurre después del renderizado de los objetos. Es útil para realizar la mayoría del código de comportamiento (movimientos, vida, controles, etc. ), ya que puede comprobar o realizar acciones hasta que cierto criterio se cumpla.*

```
void Update () {  
}
```

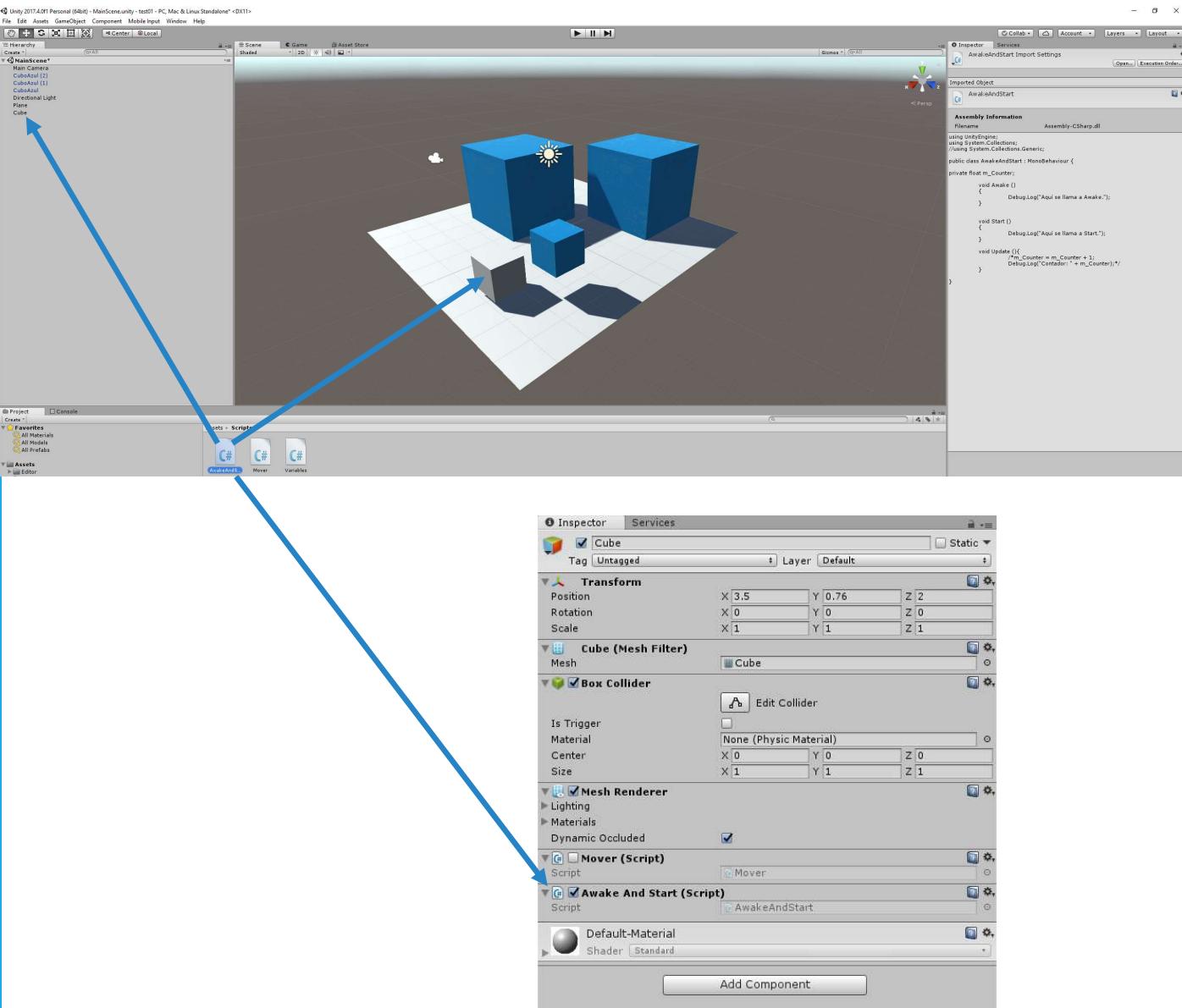
*// Se ejecuta una vez. Ocurre justo después de instanciar un prefab y antes de cualquier Start. Aunque este deshabilitado el Script en el inspector de objetos, la ejecución de Awake ocurre.*

```
void Awake () {  
}
```

# Adjuntando scripts a GameObjects

Un Script puede ser adjuntado a un GameObject y este aparecerá en el Inspector.

Para hacerlo, simplemente hay que arrastrar y soltar el script hacia el Inspector con el elemento deseado (ya sea visualmente en Scene View, textualmente en Hierarchy Window o Inspector Window).



# Probando Hilos de Ejecución Start y Awake

Mediante el siguiente código podemos comprobar cuando es llamado cada elemento y en que orden.

Podemos darnos cuenta que al deshabilitar el script en el inspector, ya no se ejecuta Start, sin embargo Awake si que se ejecuta.

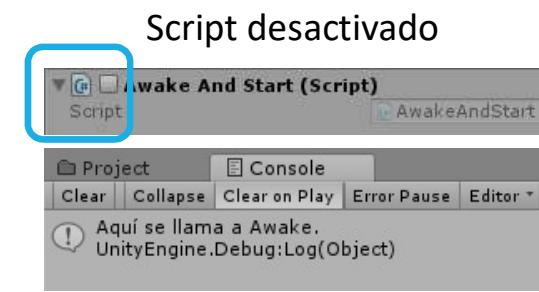
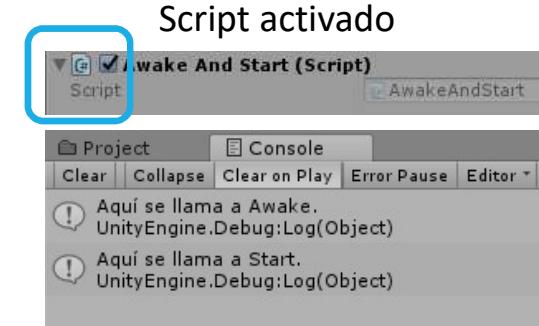
```
using UnityEngine;
using System.Collections;

public class AwakeAndStart : MonoBehaviour {

    void Awake ()
    {
        Debug.Log("Aquí se llama a Awake.");
    }

    void Start ()
    {
        Debug.Log("Aquí se llama a Start.");
    }

    void Update ()
    }
```



# Probando Hilos de Ejecución Update

Modificaremos ligeramente el Script para demostrar la ejecución de Update.

```
using UnityEngine;
using System.Collections;

public class AwakeAndStart : MonoBehaviour {
    private float m_Counter;

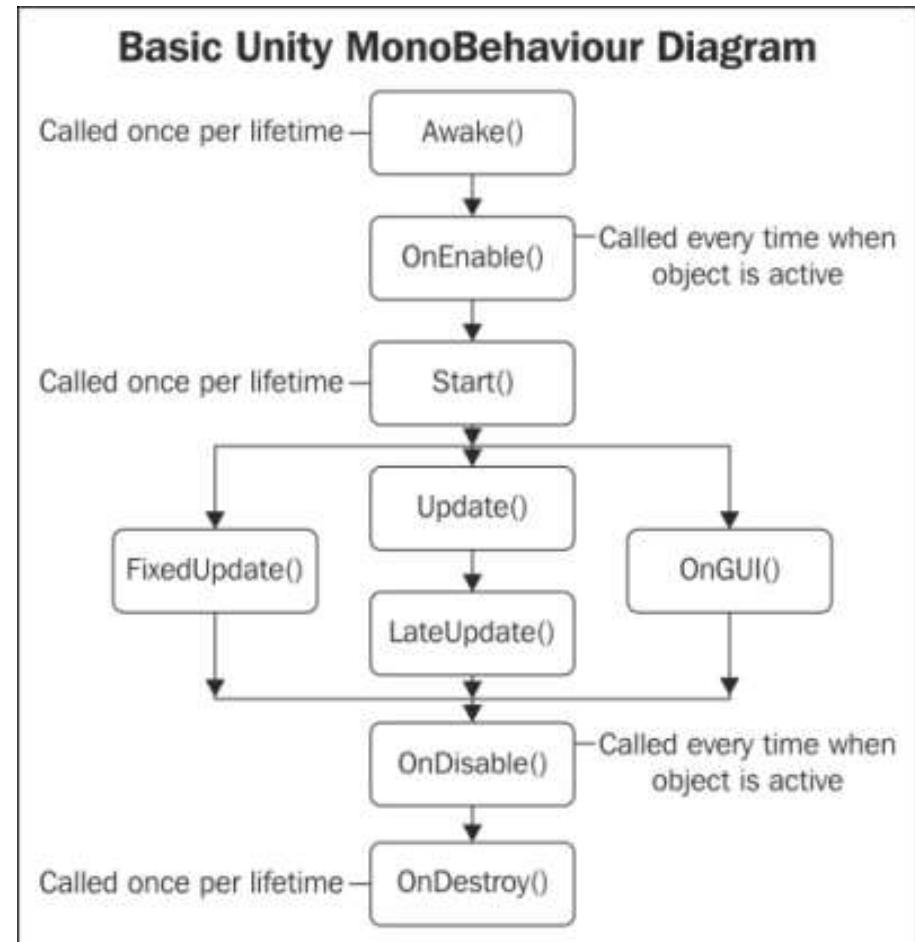
    void Awake ()
    {
        Debug.Log("Aquí se llama a Awake.");
    }

    void Start ()
    {
        Debug.Log("Aquí se llama a Start.");
    }

    void Update (){
        m_Counter = m_Counter + 1;
        Debug.Log("Contador: " + m_Counter);
    }
}
```

# Hilos de Ejecución en Unity

Diagrama más completo de hilos de ejecución.



# Buenas prácticas de nomenclatura de Variables

Como buena practica, usaremos un prefijo para nombrar una variable:

**m\_**

Seguido del nombre de la variable en CamelCase:

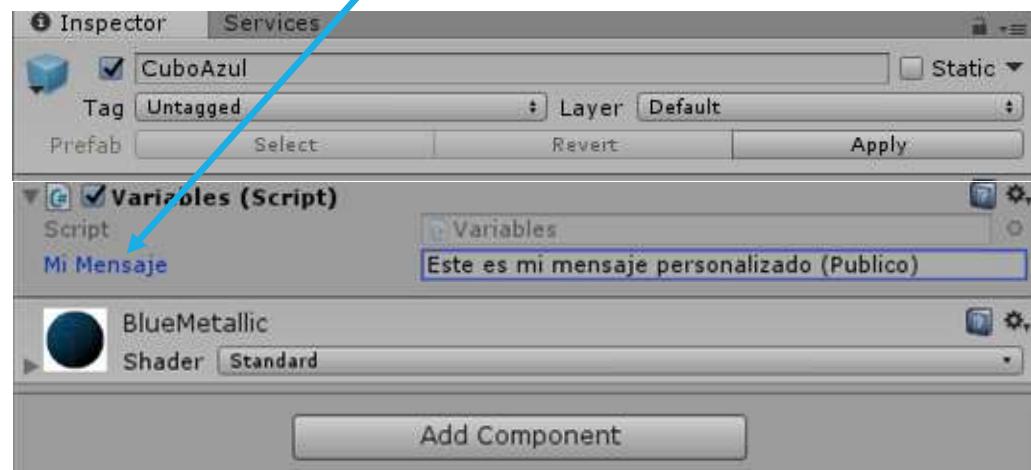
**MiVariable**

Para al final obtener:

**m\_MiVariable**

Si es variable publica, al ser colocado en el Inspector, Unity omitirá el prefijo y colocara los espacios al nombre de la variable.

```
5 public class Variables : MonoBehaviour {  
6  
7     public string m_MiMensaje;
```



# Declaración de Variables

Principalmente usaremos estos 2 tipos de variables:

**Privada (Private):** Donde la variable sólo puede ser usada dentro de nuestra clase.

**Pública (Public):** Donde el valor de la variable puede ser establecido desde el inspector del objeto que tiene el Script.

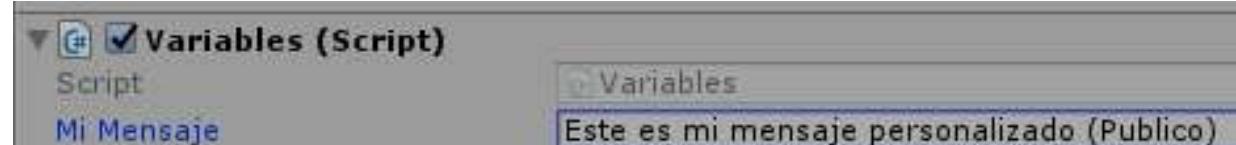
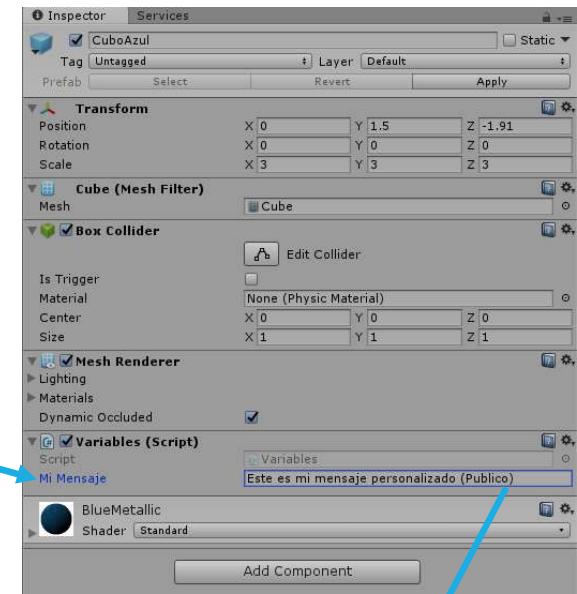
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Variables : MonoBehaviour {

    public string m_Mensaje;
    private string m_VariablePrivada;

    // Use this for initialization
    void Start () {
        m_VariablePrivada = " solo Accesible dentro de la clase";
        Debug.Log(m_Mensaje + m_VariablePrivada);
    }

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



## Ejemplo: Script de Rotación

Script que realiza la rotación de un objeto, controlando gracias a variables publicas la velocidad y dirección de la rotación.

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

public class RotateObject : MonoBehaviour {

    //Public Fields
    public float m_Velocity;
    public Vector3 m_Direction;

    // Use this for initialization
    void Start () {
    }

    // Update is called once per frame
    void Update () {
        transform.Rotate (m_Direction * m_Velocity * Time.deltaTime);
    }
}
```

# Ejemplo: Script de Traslación

Script que realiza la traslación (desplazamiento) de un objeto en orientación global o local, controlando la velocidad, dirección inicial del desplazamiento y distancia máxima, para al alcanzarla, regresar hacia el punto de origen.

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

public class TranslateObject : MonoBehaviour {
    //public fields
    public float m_Velocity;
    public float m_MaxDistance;
    public Vector3 m_Direction;
    public bool m_GlobalOrientation;

    //private fields
    private Vector3 m_Origin;
    private bool m_isGoing;

    // Use this for initialization
    void Start () {
        m_isGoing = true;
        m_Origin = transform.position;
    }

    // Update is called once per frame
    void Update () {
        if (Vector3.Distance (m_Origin, transform.position) < m_MaxDistance) {
            transform.Translate (
                (m_Direction * (m_isGoing ? 1 : -1)) * m_Velocity * Time.deltaTime,
                (m_GlobalOrientation ? Space.World : Space.Self));
        } else {
            m_Origin = transform.position;
            m_isGoing = !m_isGoing;
        }
    }
}
```

## Ejemplo: Script de Escala

Incremento y decremento entre un rango de valores, a velocidad variable, multiplicando Time.deltaTime por la variable m\_Velocity y controlando en que ejes se escala.

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

public class ScaleObject : MonoBehaviour {

    //public
    public float m_MaxScale;
    public float m_MinScale;
    public float m_Velocity;

    public bool m_ScaleX;
    public bool m_ScaleY;
    public bool m_ScaleZ;

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        //Tiempo que lleva ejecutandose el juego -> Time.deltaTime
        float newScale = Mathf.PingPong (Time.deltaTime*m_Velocity, m_MaxScale - m_MinScale) + m_MinScale;
        //Uso de operadores ternarios, para validar en que ejes se incrementa.
        float newScaleX = m_ScaleX ? newScale : transform.localScale.x;
        float newScaleY = m_ScaleY ? newScale : transform.localScale.y;
        float newScaleZ = m_ScaleZ ? newScale : transform.localScale.z;
        transform.localScale = new Vector3 (newScaleX,newScaleY,newScaleZ);
    }
}
```

# Referencias

---

## Unity3D: Awake and Start

<https://unity3d.com/es/learn/tutorials/topics/scripting/awake-and-start>

<https://www.relaxate.com/tutorial-documentacion-manual-programacion-videojuegos-unity-2d-parte-2>

<https://www.deustoformacion.com/blog/diseño-producción-audiovisual/6-mejores-juegos-3d-hechos-con-unity>

<https://docs.unity3d.com/es/current/Manual/class-GameObject.html>

<https://docs.unity3d.com/es/current/Manual/CreatingComponents.html>

<https://docs.unity3d.com/Manual/LightMode-Baked.html>