Informática Gráfica
Transformaciones geométricas

Juan Carlos Torres
Grupos C y D

Dpt. Lenguajes y Sistemas Informáticos
ETSI Informática y de Telecomunicación
Universidad de Granada

—

Curso 2024-25

Transformaciones Geométricas

Transformaciones geométricas

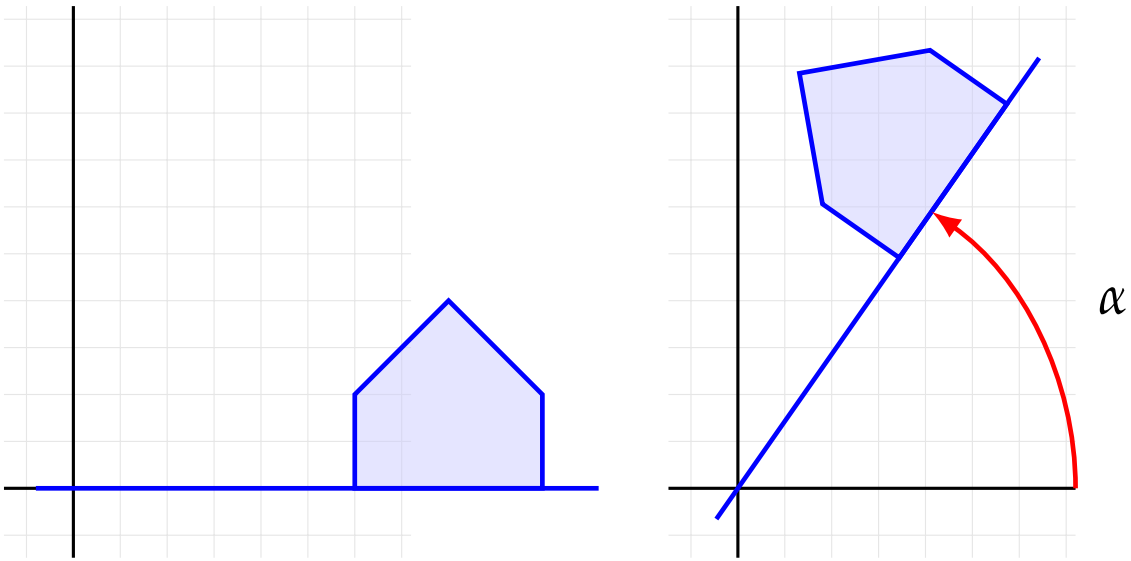
Las transformaciones geométricas son funciones que modifican las coordenadas de los puntos. Existen varias transformaciones geométricas simples que son muy útiles en Informática Gráfica para la definición de escenas y animaciones:

- **Traslación:** desplazar todos los puntos del espacio de igual forma, es decir, en la misma dirección y la misma distancia.
- **Escalado:** estrechar o alargar las figuras en una o varias direcciones.
- **Rotación:** rotar los puntos un ángulo en torno a un eje

Transformación de rotación en 2D

Si α es un valor real cualquiera (que expresa un ángulo en radianes), entonces podemos construir la transformación R_α de **rotación** entorno al origen $\mathbf{0}$, un ángulo α radianes. Para cualquier $\mathbf{p} = (x, y)$:

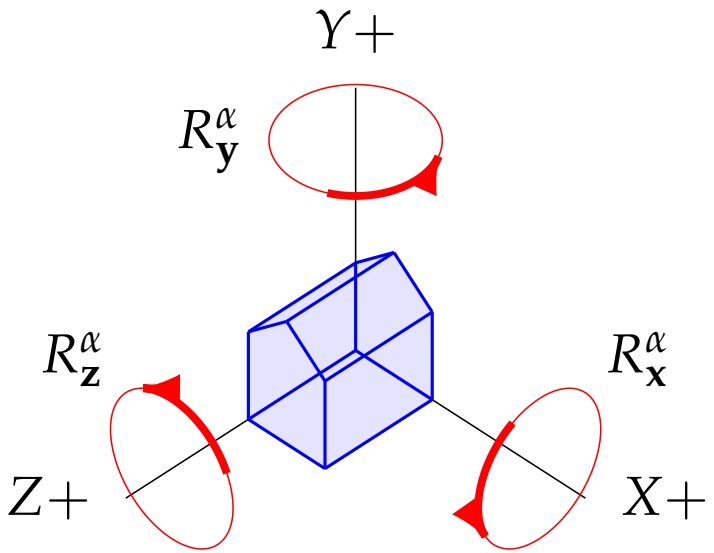
$$R_\alpha(\mathbf{p}) \equiv (x', y') \quad \text{donde:} \quad \begin{cases} x' & \equiv & \cos(\alpha)x - \sin(\alpha)y \\ y' & \equiv & \sin(\alpha)x + \cos(\alpha)y \end{cases}$$



Esta rotación es en el sentido contrario a las agujas del reloj cuando $\alpha > 0$, y en el sentido de las agujas cuando $\alpha < 0$ (se dice que la rotación es en sentido anti-horario).

Transformaciones de rotación en 3D

En $\mathbb{E}_3$ hay tres posibles rotaciones básicas (una para cada eje). Las llamaremos $R^{\alpha}_{\mathbf{x}}$,$R^{\alpha}_{\mathbf{y}}$ y $R^{\alpha}_{\mathbf{z}}$, donde α es un valor real que indica el ángulo de giro (en radianes).



- Cada rotación 3D (entorno a cada eje) es similar a la rotación 2D actuando sobre dos coordenadas, y dejando sin cambiar la coordenada correspondiente al eje de giro.

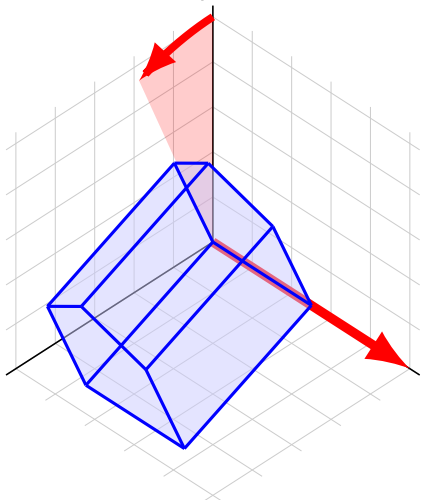
Asumimos que estas rotaciones son **siempre en sentido anti-horario** (cuando se mira hacia el origen desde la rama positiva del eje de giro).

Expresiones y ejemplos de las rotaciones en 3D

Las expresiones de las tres rotaciones son:

Eje X ($R^{\alpha}_{\mathbf{x}}$)

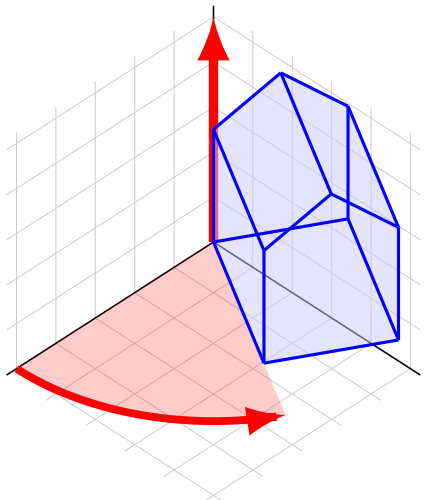
$$\begin{aligned} x' &\equiv x \\ y' &\equiv cy - sz \\ z' &\equiv sy + cz \end{aligned}$$



$R^{23^o}_{\mathbf{x}}$

Eje Y ($R^{\alpha}_{\mathbf{y}}$)

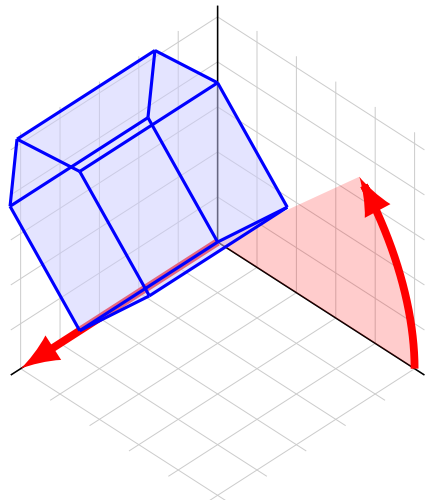
$$\begin{aligned} x' &\equiv cx + sz \\ y' &\equiv y \\ z' &\equiv -sx + cz \end{aligned}$$



$R^{60^o}_{\mathbf{y}}$

Eje Z ($R^{\alpha}_{\mathbf{z}}$)

$$\begin{aligned} x' &\equiv cx - sy \\ y' &\equiv sx + cy \\ z' &\equiv z \end{aligned}$$



$R^{45^o}_{\mathbf{z}}$

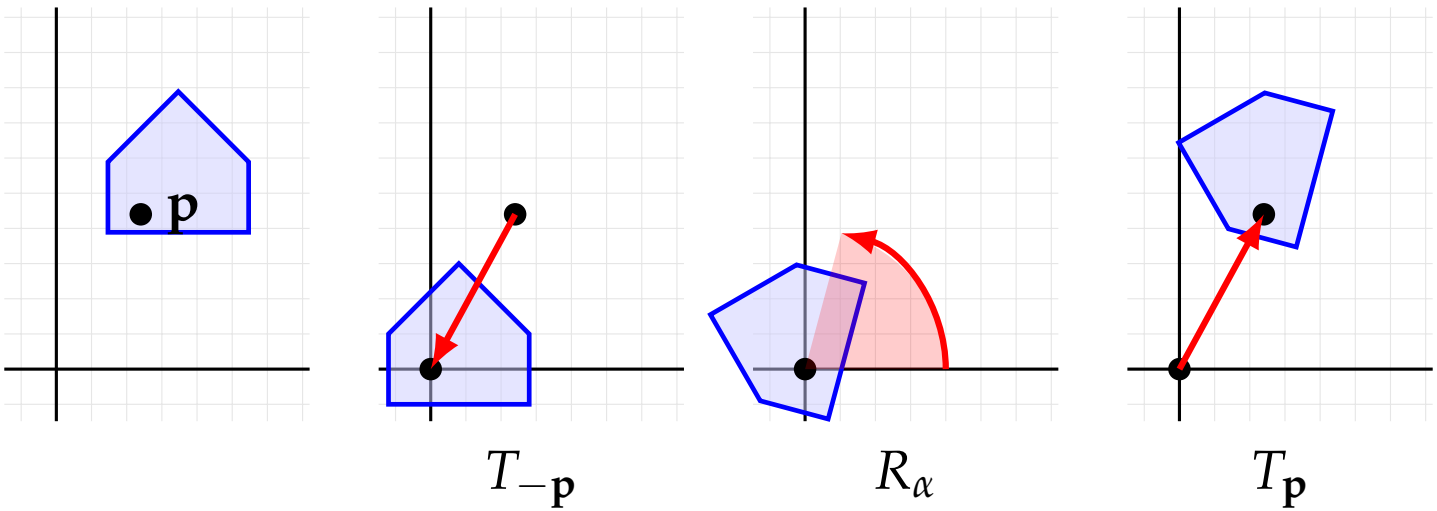
Donde: $c \equiv \cos(\alpha)$ y $s \equiv \sin(\alpha)$

Rotaciones 2D sobre puntos arbitrarios

Un ejemplo de composición es la rotaciones 2D entorno a un punto $\mathbf{p}$ distinto del origen ($Q^{\alpha}_{\mathbf{p}}$), que se consigue componiendo traslación (por $-\mathbf{p}$), rotación y traslación (por $\mathbf{p}$):

$$Q^{\alpha}_{\mathbf{p}} \quad = \quad T_{-\mathbf{p}} \circ R_{\alpha} \circ T_{\mathbf{p}}$$

las transformaciones usadas se aprecian en esta figura:

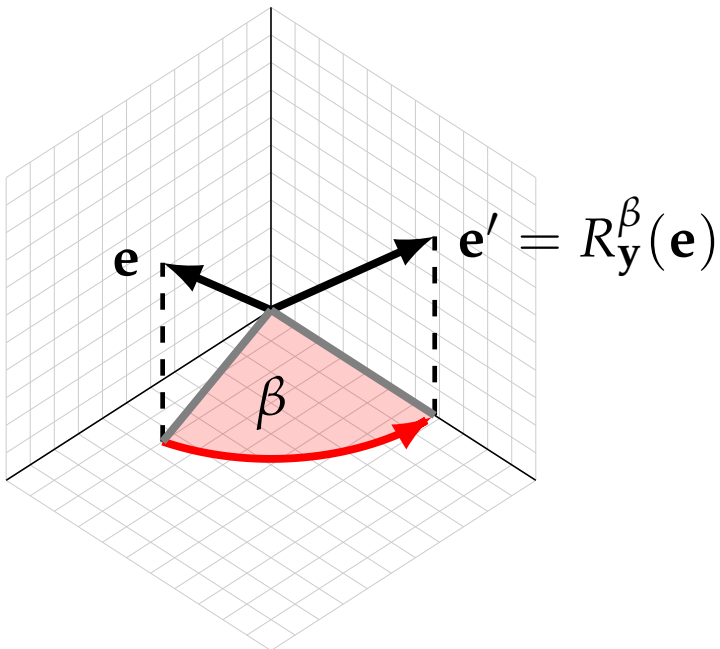


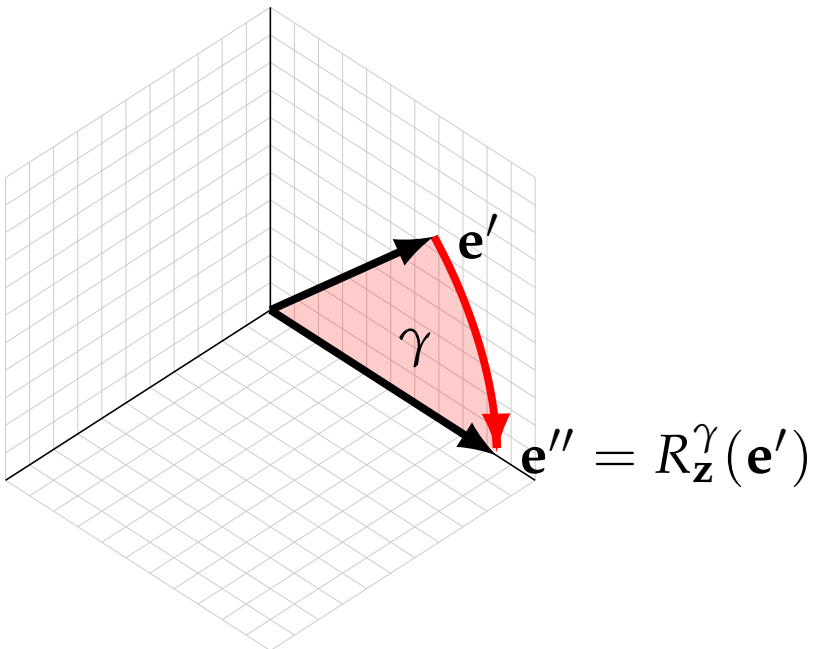
Rotaciones 3D con ejes arbitrarios

Si $\mathbf{e} = (e_x, e_y, e_z)$ es un vector arbitrario (con $\|\mathbf{e}\| > 0$), entonces se puede definir la transformación de rotación α radianes con $\mathbf{e}$ como eje de rotación, que notaremos como $R^{\alpha}_{\mathbf{e}}$:

$$R^{\alpha}_{\mathbf{e}} = A \circ R^{\alpha}_{\mathbf{x}} \circ A^{-1} \qquad \text{donde:} \quad A \equiv R^{\beta}_{\mathbf{y}} \circ R^{\gamma}_{\mathbf{z}}$$

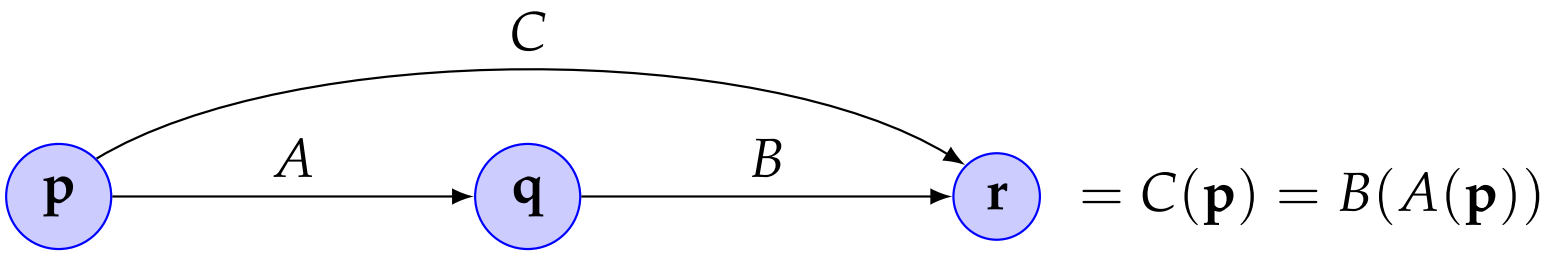
β y γ dependen de $\mathbf{e}$ y se eligen de forma que A alinea $\mathbf{e}$ con $\mathbf{x}$:





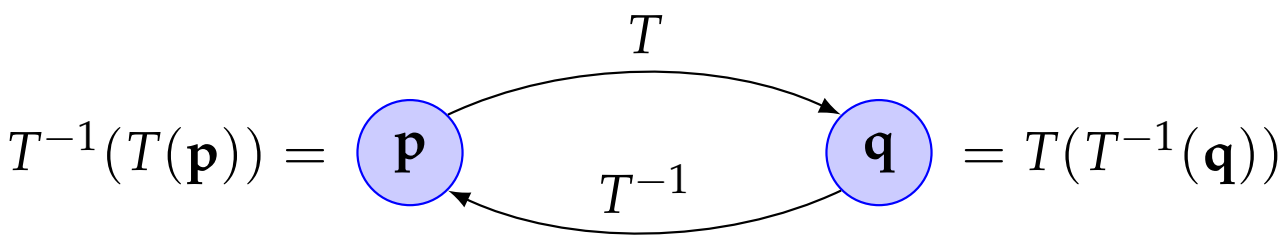
Composición e inversa

Las transformaciones geométrica se pueden **componer**. Así la aplicación de las transformaciones A y B, es una nueva transformación $C = A \circ B$:



Esta composición es, en general, **no conmutativa**.

Algunas transformaciones T tienen una **inversa** T^{-1}:



Transformaciones afines y lineales

Todas las transformaciones que hemos visto (traslación, escalado y rotación) son **transformaciones afines**:

- Esto significa que transforman líneas rectas en líneas rectas, manteniendo las proporciones entre longitudes de segmentos en dichas rectas.

Además, los escalados y las rotaciones son **transformaciones lineales**

- Una transformación T será lineal si y solo si cumple:

$$T(a\mathbf{p} + b\mathbf{q}) \quad = \quad a\,T(\mathbf{p}) \,+\, b\,T(\mathbf{q})$$

  (para cualquiera puntos $\mathbf{p}$ y $\mathbf{q}$, y valores reales a y b)

lineal implica afín, pero no al contrario.

Las **traslaciones** son afines pero **no son lineales**).

Matrices asociadas a transformaciones lineales

Si T es una transformación lineal en $\mathbb{E}_3$, entonces existen nueve valores reales $(m_{00},\ldots,m_{22})$ que determinan la transformación T. Si $\mathbf{p}=(x,y,z)$ y $T(\mathbf{p})=\mathbf{p}'=(x',y',z')$ entonces:

$$\begin{aligned} x' &= m_{00}x + m_{01}y + m_{02}z \\ y' &= m_{10}x + m_{11}y + m_{12}z \\ z' &= m_{20}x + m_{21}y + m_{22}z \end{aligned}$$

Esto es equivalente a decir que T tiene asociada una **matriz de transformación** $M=(m_{ij})$ (3x3) que actua sobre las coordenadas de los puntos (escritas como vectores columna):

$$\mathbf{p}' = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = M\mathbf{p}$$

(en 2D es similar con matrices 2x2).

Coordenadas Homogéneas en 3D y 2D

Para poder representar todas las transformaciones geométricas con matrices se usan coordenadas homogéneas.

El **espacio de coordenadas homogéneas 3D** es el conjunto de tuplas de la forma $(x, y, z, w) \in \mathbb{R}^4$. Se puede visualizar como un espacio de cuatro dimensiones con un nuevo eje W perpendicular a los otros tres. La tupla (x, y, z, w) representa:

- el punto 3D de coordenadas $(x/w, y/w, z/w)$, si $w \neq 0$.
- la dirección del vector que va desde el origen a (x, y, z), si $w = 0$ (a estas tuplas se les llama *puntos en el infinito*).

Los puntos y direcciones 2D en el plano cartesiano también pueden representarse usando el espacio de coordenadas homogéneas 2D, que ahora son ternas de la forma (x, y, w), que representan

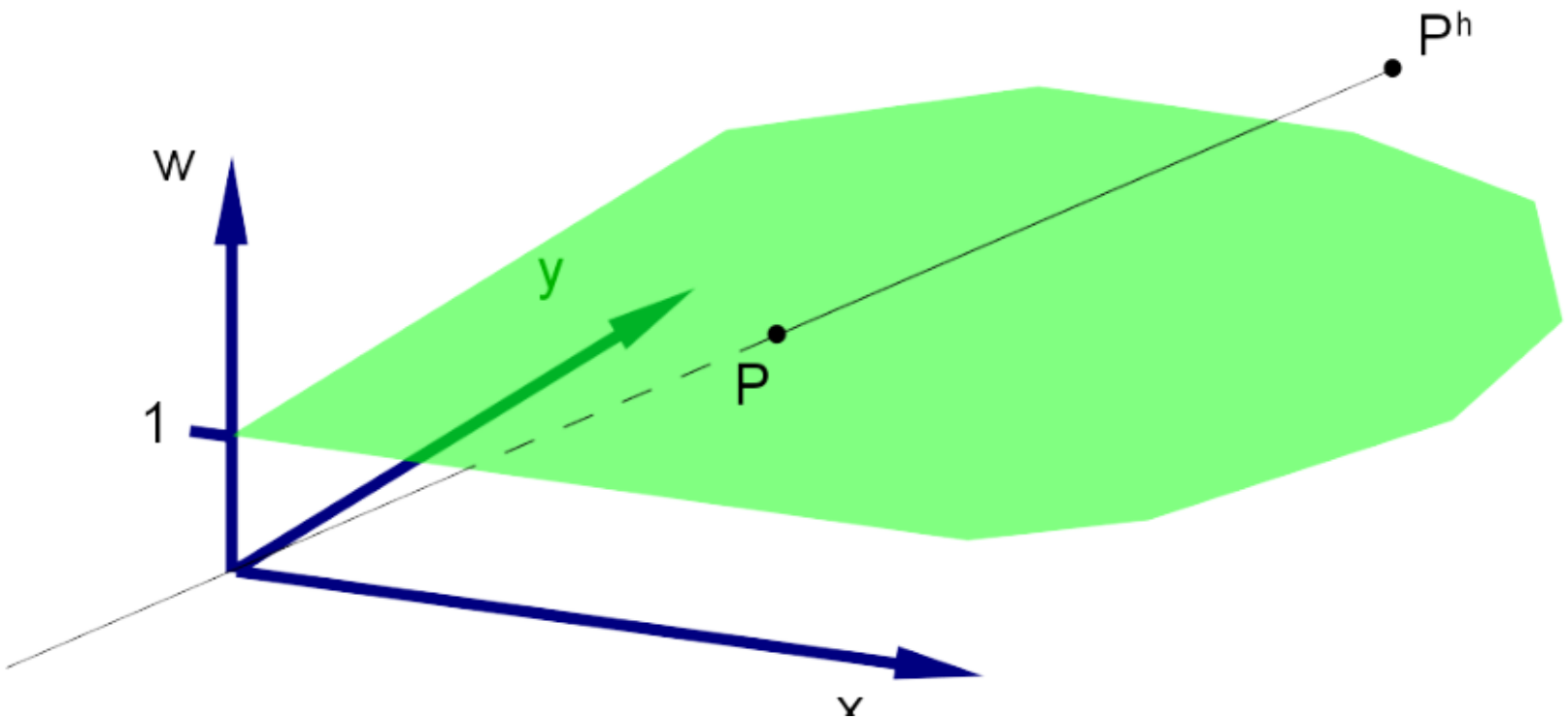
- el punto del plano de coordenadas $(x/w, y/w)$, si $w \neq 0$.
- la dirección del vector que va desde el origen a (x, y), si $w = 0$.

El espacio de coordenadas homogéneas 2D se puede visualizar de forma semejante a $\mathbb{E}_3$.

Coordenadas Homogéneas en 2D

Para poder representar todas las transformaciones geométricas con matrices se usan coordenadas homogéneas.

$$P^h=(x,y,z,w)\rightarrow P=(x',y',z') \quad t.q. \quad x'=\frac{x}{w};y'=\frac{y}{w};z'=\frac{z}{w}$$



Coordenadas homogéneas y transformaciones

Un punto 3D, representado por sus coordenadas homogéneas, puede ser transformado usando exclusivamente multiplicaciones por matrices 4x4:

- Se usan tuplas de la forma $(x, y, z, 1)$ para el punto (x, y, z)
- Una transformación lineal T en coordenadas euclídeas 3D (con matriz M) se puede extender fácilmente a una transformación lineal en coordenadas homogéneas, para ello extendemos la matriz 3x3 $M = (m_{ij})$ añadiendole una fila y una columna:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} m_{00} & m_{01} & m_{02} & 0 \\ m_{10} & m_{11} & m_{12} & 0 \\ m_{20} & m_{21} & m_{22} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = M\mathbf{p}$$

- La composición de transformaciones lineales se puede implementar muy fácilmente usando composición de matrices 4x4.

Matrices 4x4 de transformaciones usuales

Escalado ($E_{\mathbf{s}}$)
$\mathbf{s} = (s_x, s_y, s_z)$

$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Traslación ($T_{\mathbf{d}}$)
$\mathbf{d} = (d_x, d_y, d_z)$

$$\begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotac. X ($R^{\alpha}_{\mathbf{x}}$)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotac. Y ($R^{\alpha}_{\mathbf{y}}$)

$$\begin{pmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotac. Z ($R^{\alpha}_{\mathbf{z}}$)

$$\begin{pmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

donde: $c \equiv cos(\alpha)$ y $s \equiv sin(\alpha)$, y α es el ángulo de rotación, en radianes

Composición como producto de matrices

Si tenemos dos transformaciones lineales A y B, y una tercera transformación $C = A \circ B$, entonces:

- A y B tienen asociadas las matrices 4x4 $M = (m_{ij})$ y $N = (n_{ij})$, respectivamente, luego $\mathbf{p}' = B(A(\mathbf{p}))$ implica $\mathbf{p}' = NM\mathbf{p}$.
- C también es una transformación lineal, y tiene asociada la matriz $O = (o_{ij})$. Por tanto $\mathbf{p}' = C(\mathbf{p})$ implica $\mathbf{p}' = O\mathbf{p}$.

Como consecuencia, para cualquier $\mathbf{p}$, se cumple $NM\mathbf{p} = O\mathbf{p}$, luego:

$$O = NM \quad \Longleftrightarrow \quad o_{ij} = \sum_{k=0}^{3} n_{ik}\, m_{kj}$$

Es decir: **la matriz asociada a la transformación compuesta es el producto de las matrices originales** (en orden inverso).

Secuencias de transformaciones

Una de las grandes ventajas del uso de matrices es para aplicar una secuencia de transformaciones a un punto. Supongamos la transformación C, composición de n transformaciones $T_1, T_2, \ldots, T_n$:

$$C = \underbrace{T_1 \circ T_2 \circ T_3 \circ \cdots \circ T_n}_{\longrightarrow}$$

para aplicar T a un punto, podemos simplemente multiplicarlo por la matriz producto (M) de las matrices originales $N_1, N_2, \ldots, N_n$:

$$M = \underbrace{N_n N_{n-1} N_{n-2} \cdots N_2 N_1}_{\longleftarrow}$$

lo cual simplifica y hace mucho más cortos los cálculos, al poderse precalcular la matriz producto y aplicarse a un conjunto grande de puntos.

Transformaciones geométricas en OpenGL

Rota α grados (en sentido antihorario) respecto al eje (e_x, e_y, e_z)

```
glRotatef( GLfloat α, GLfloat e_x, GLfloat e_y, GLfloat e_z )
```

Traslada según el vector (d_x, d_y, d_z)

```
glTranslatef( GLfloat d_x, GLfloat d_y, GLfloat d_z )
```

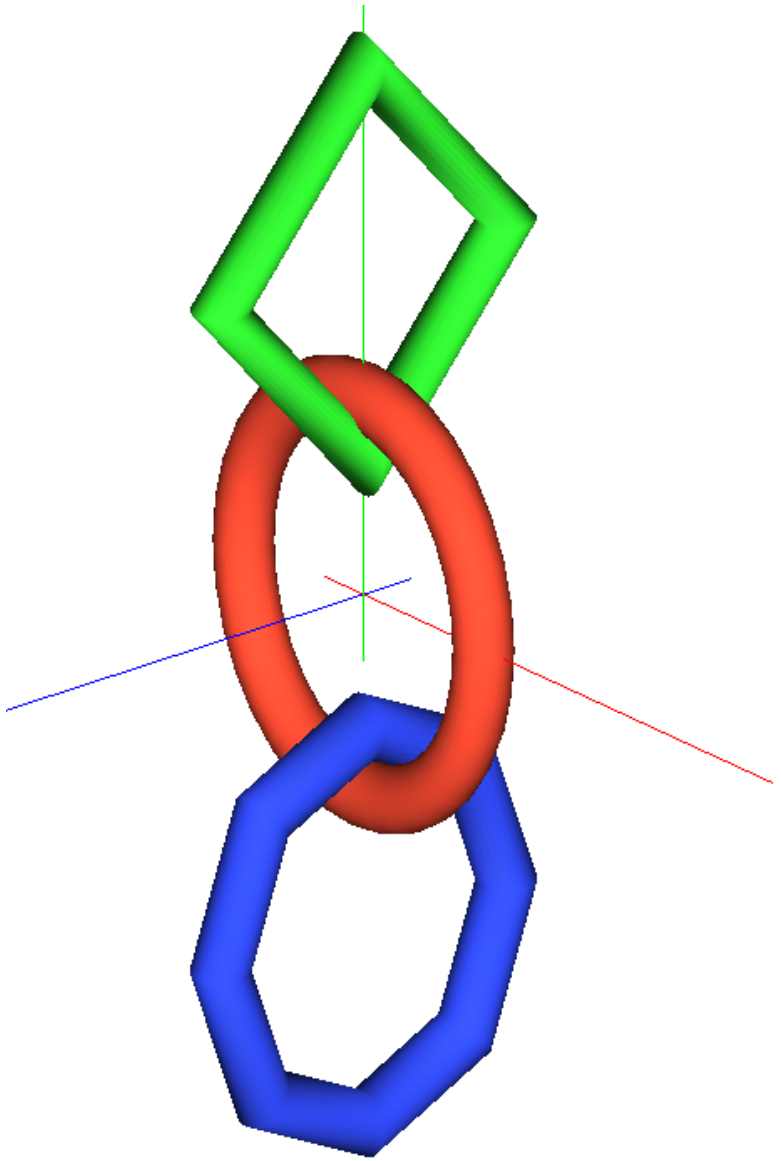
Escala en (s_x, s_y, s_z)

```
glScalef( GLfloat s_x, GLfloat s_y, GLfloat s_z )
```

La transformación se compone con las ya cargadas.

Composición de transformaciones

```
glutSolidTorus(0.5,3,24,32);

glTranslatef(0,5,0);
glRotatef(90,0,1,0);
glutSolidTorus(0.5,3,24,4);

glTranslatef(0,-5,0);
glRotatef(90,0,1,0);;
glutSolidTorus(0.5,3,24,8);
```



Composición de transformaciones

Guarda la transformación actual en la Pila.

```
glPushMatrix();
```

Recupera la transformación actual de la pila.

```
glPopMatrix();
```

```
glutSolidTorus(0.5,3,24,32);

glPushMatrix();
glTranslatef(0,5,0);
glRotatef(90,0,1,0);
glutSolidTorus(0.5,3,24,4);
glPophMatrix();

glTranslatef(0,-5,0);
glRotatef(90,0,1,0);;
glutSolidTorus(0.5,3,24,8);
```

Unity 3D

Unity 3D

Unity 3D es un motor de desarrollo de aplicaciones gráficas interactivas.

- Entorno de desarrollo amigable y versátil
- Permite la creación de experiencias 2D y 3D
- Puede implementar para múltiples plataformas (PC, consolas, móviles y realidad virtual)
- Se programa en C#
- Comunidad de desarrollo activa
- Gran cantidad de recursos y activos disponibles

Instalación

Descarga

- `https://unity.com/es/download`
- Disponible para: Linux, Mac y Windows
- Se instala el Unity Hub, a través del cual se gestionan los proyectos y la instalación del editor
- Manual en `https://docs.unity3d.com/Manual/index.html`

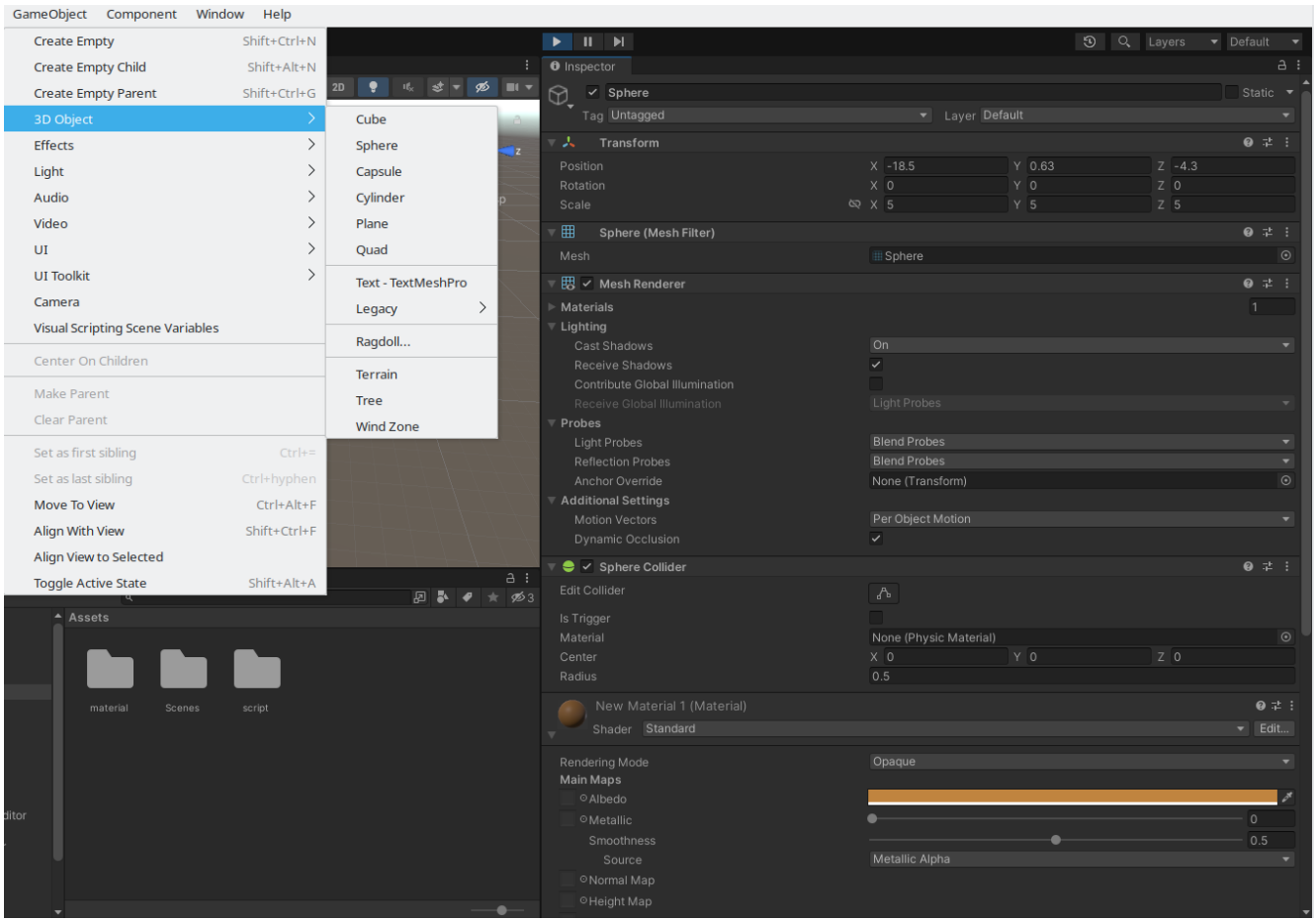
Asegurate de incluir Linux Build Support si quieres generar ejecutables para Linux.

Licencias

- Estudiante (Gratis)
- Personal (Gratis hasta 100.000 año)
- Profesional (1.900 €año)

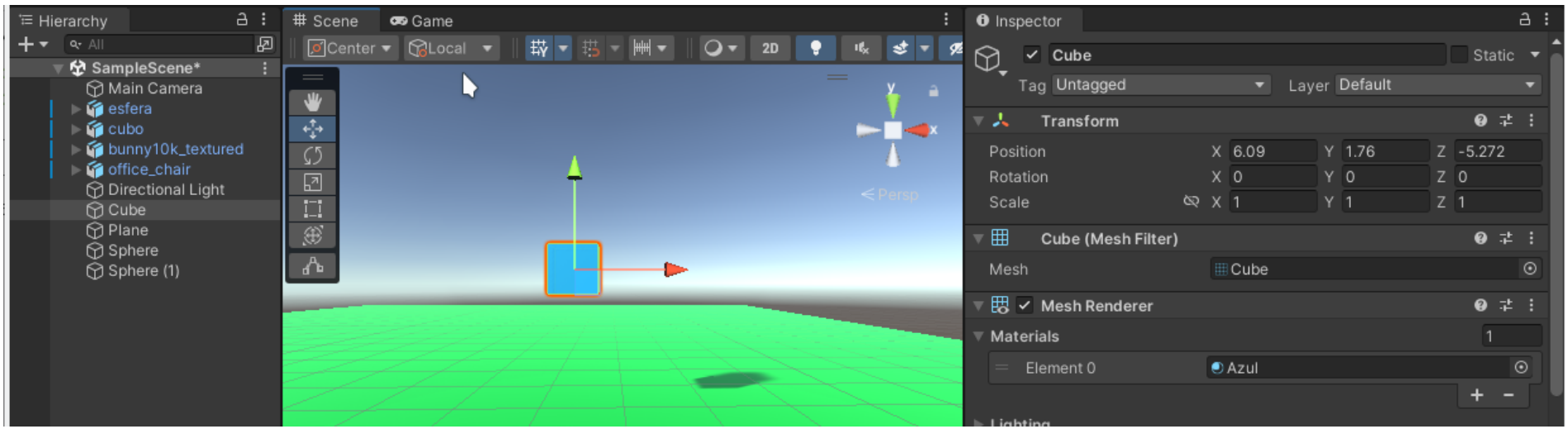
Creación de objetos

1. Primitivas simples
2. Propiedades en el panel inspector



Transformaciones geométricas en Unity

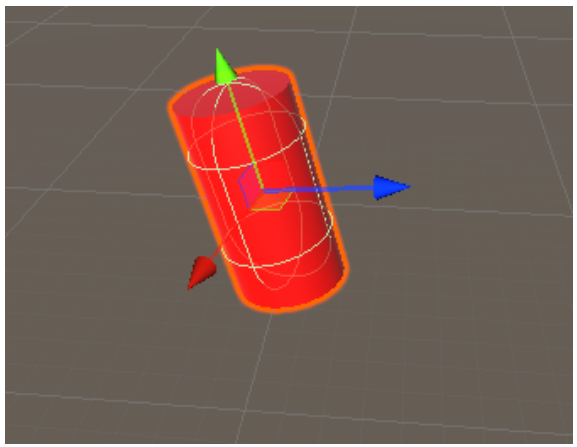
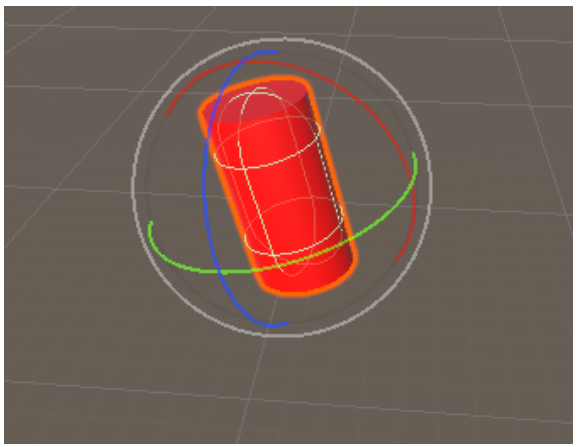
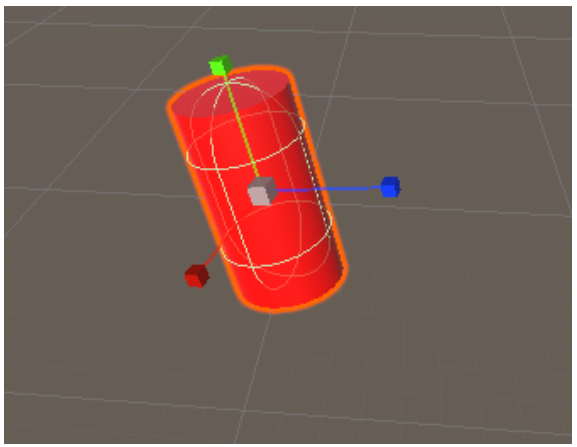
Para transformar un objeto lo seleccionamos y o bien modificamos los parámetros de las transformaciones o lo modificamos interactivamente con las herramientas de edición.



Modificación interactiva de objetos

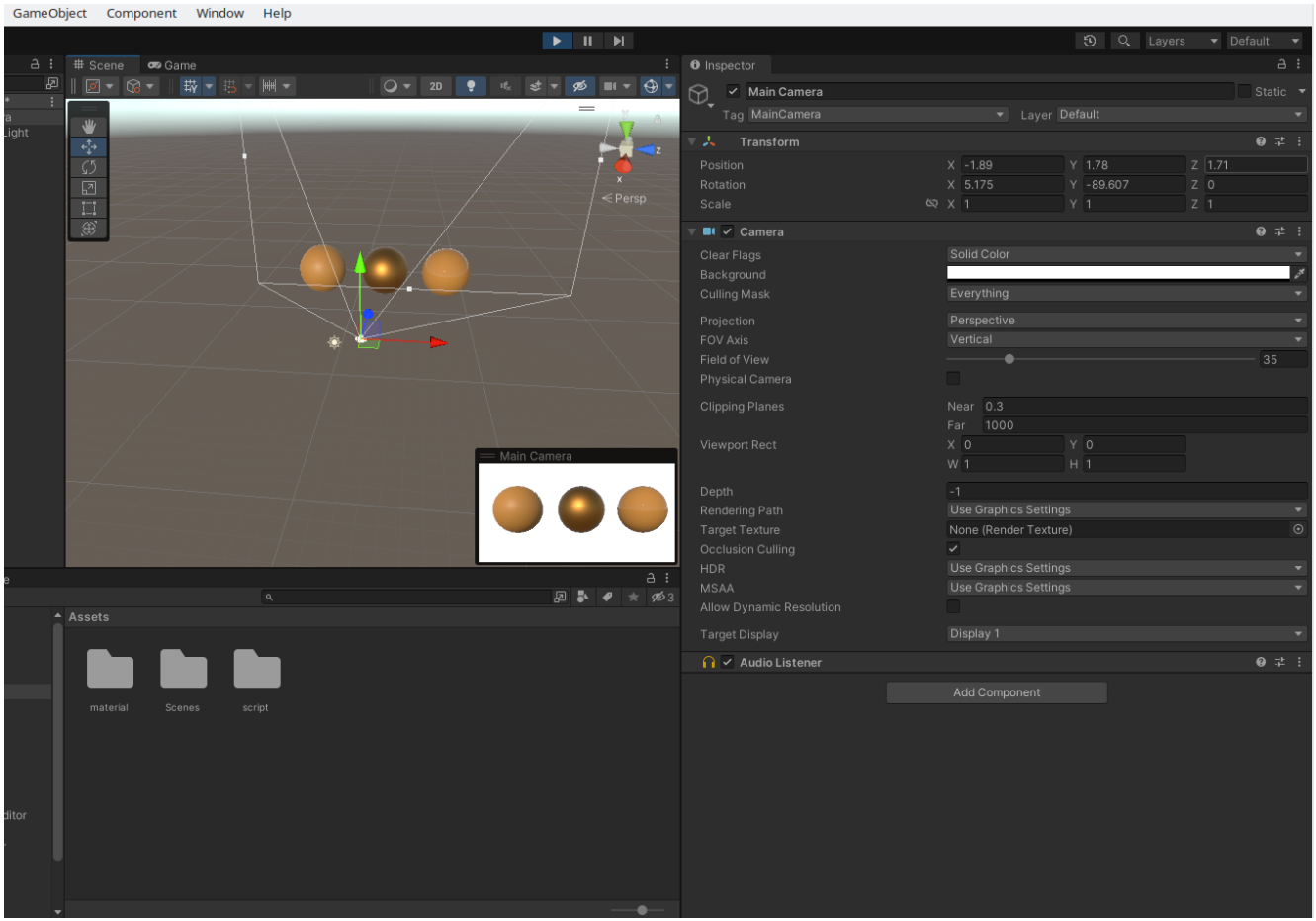
Cuando se selecciona un objeto se puede manipular con controles de

- Posición
- Orientación
- Escalado

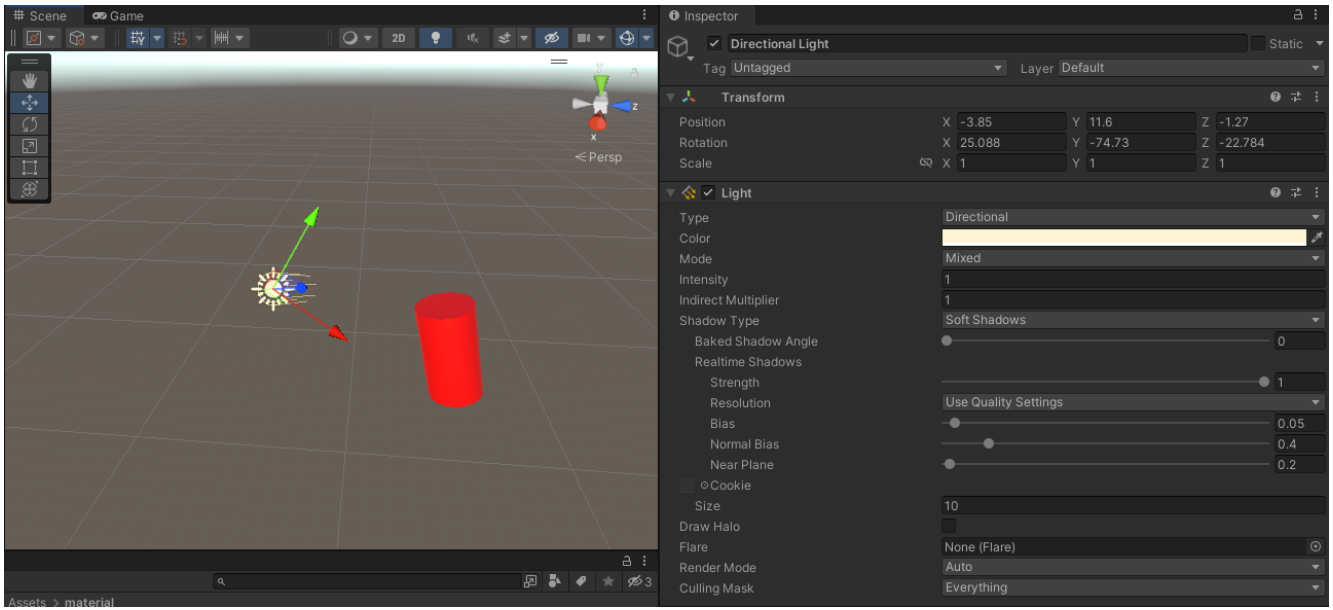
Cámara

- Hay al menos una cámara en la escena
- Cuando se selecciona aparece una ventana con la vista de la cámara
- Propiedades de la cámara aparecen en panel inspector



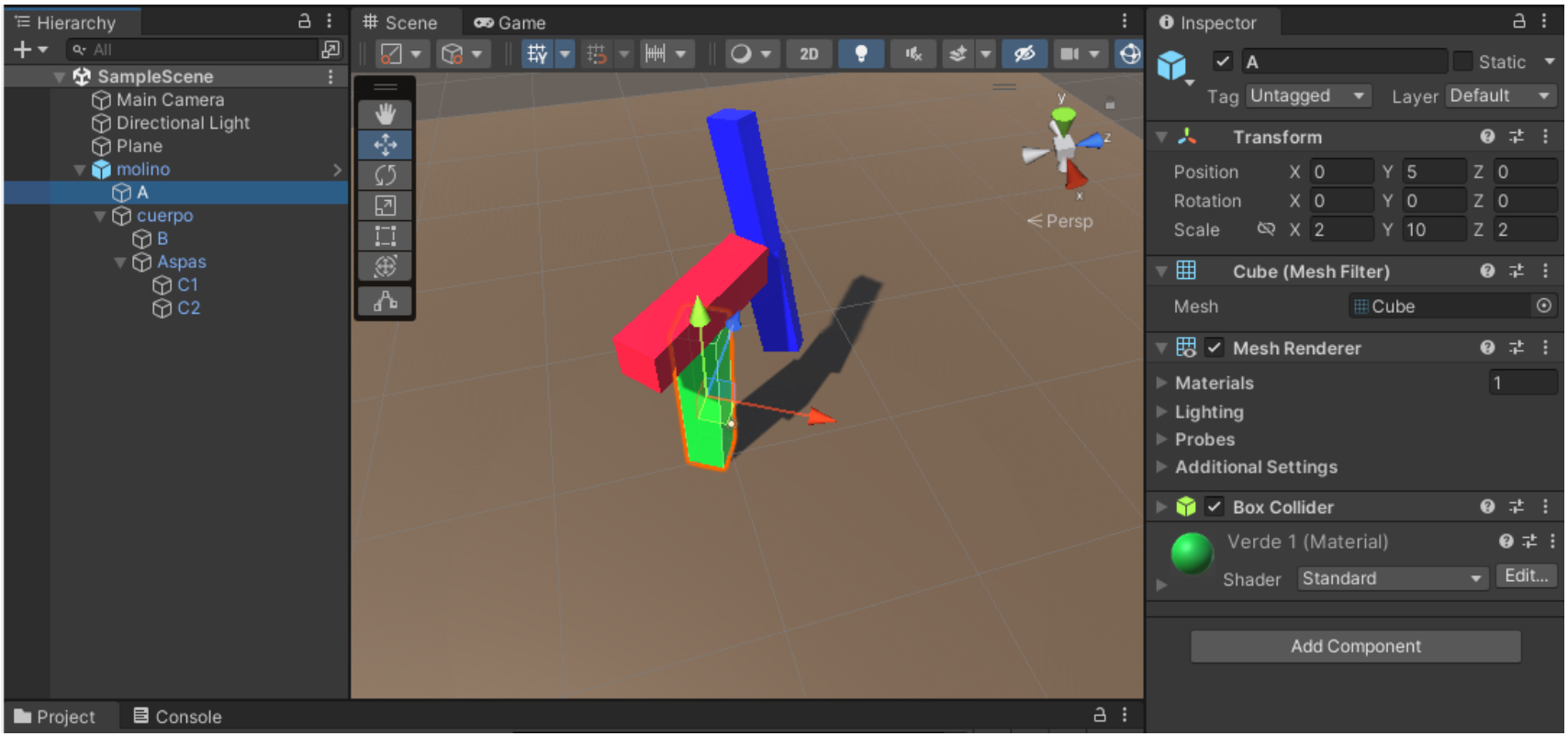
Luces

- Hay al menos una luz en la escena
- Propiedades de las luces aparecen en panel inspector



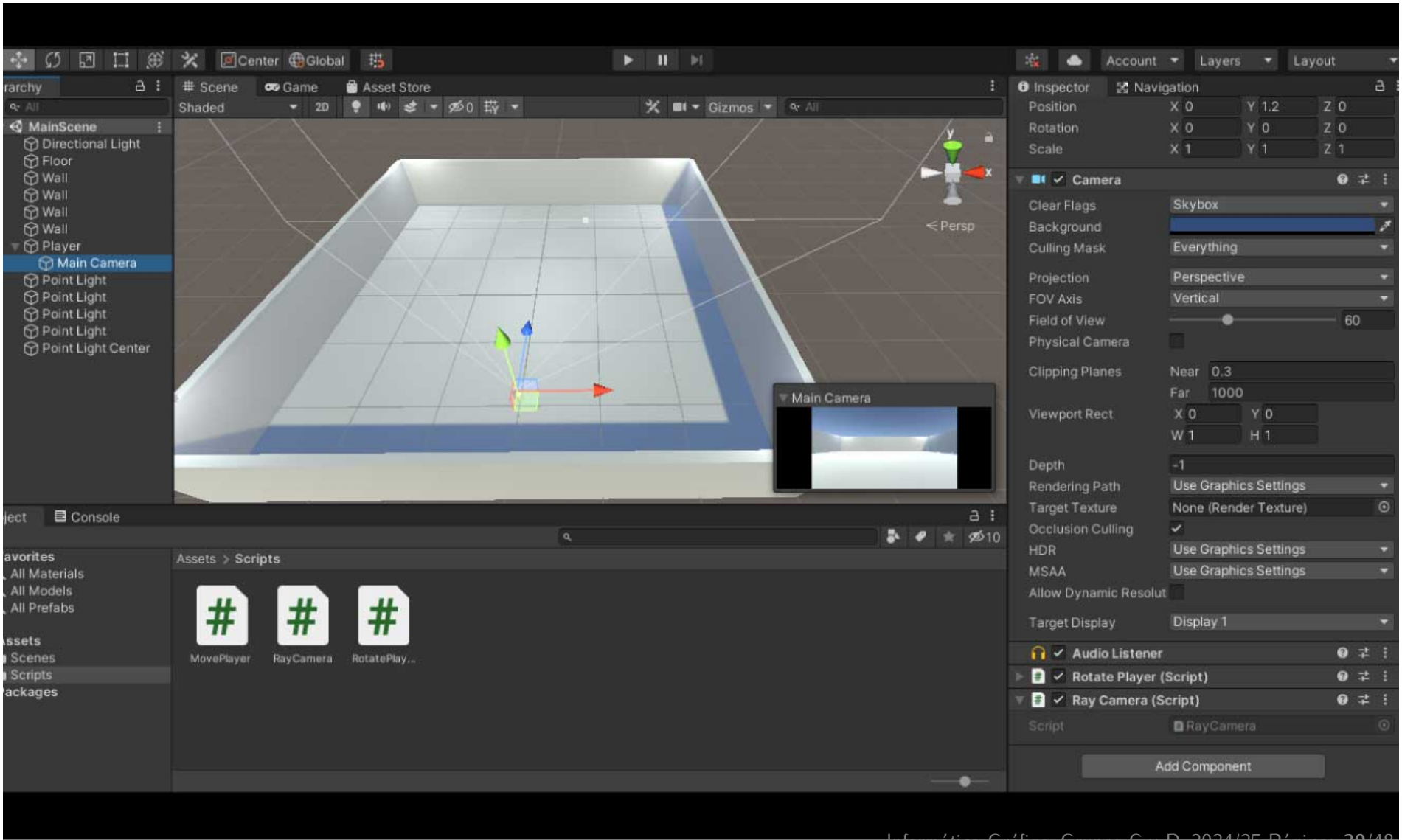
Escena en Unity

El panel jerarquía (Hierarchy) contiene la lista de elementos de la escena



Cámara y luces en la escena

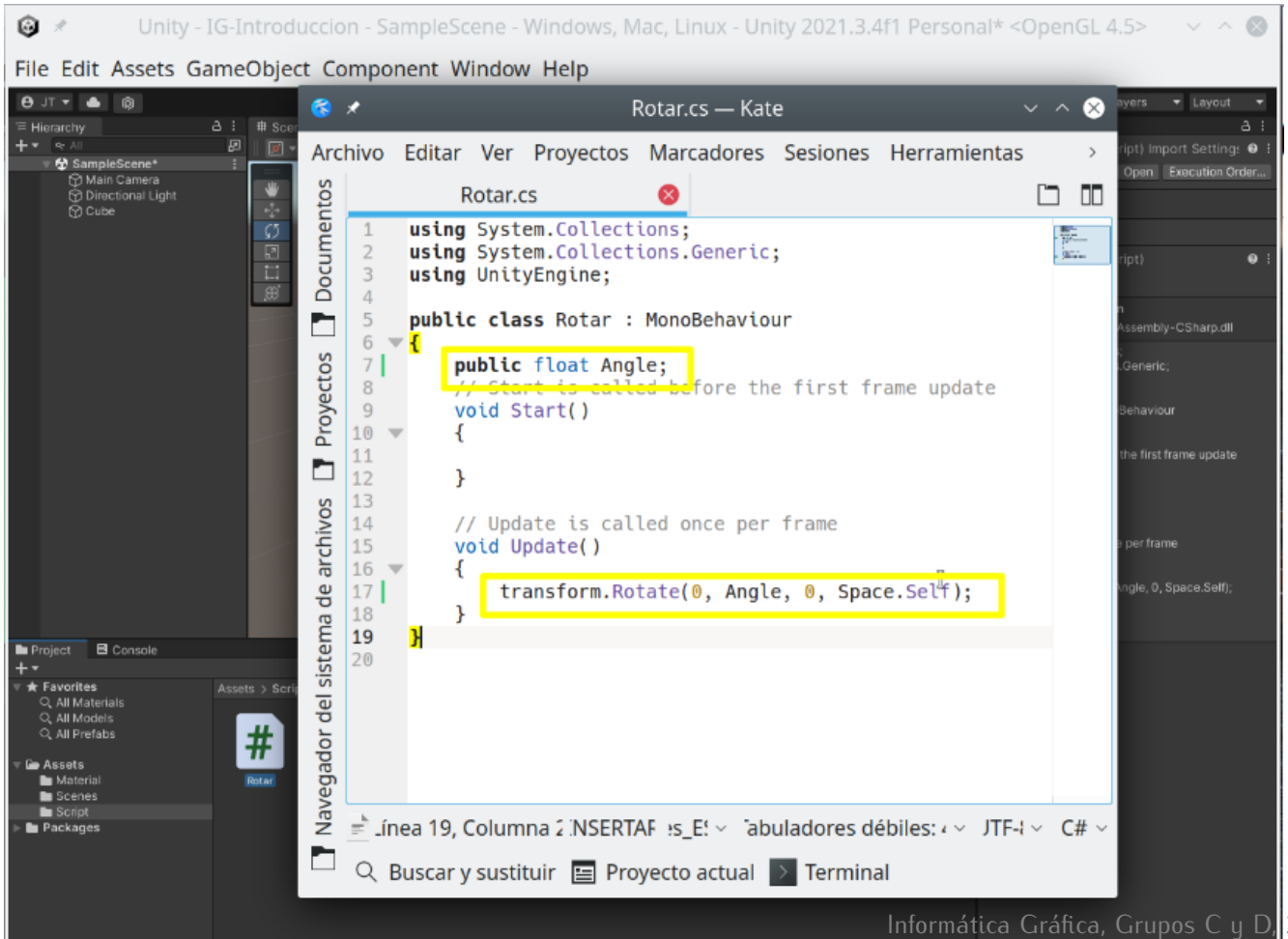
Las cámaras y luces son objetos de la escena.



Interacción

Se pueden asociar scripts a los objetos. Cada script tiene dos métodos:

- Start: Se ejecuta al iniciar el sistema.
- Update: Se ejecuta en cada frame.



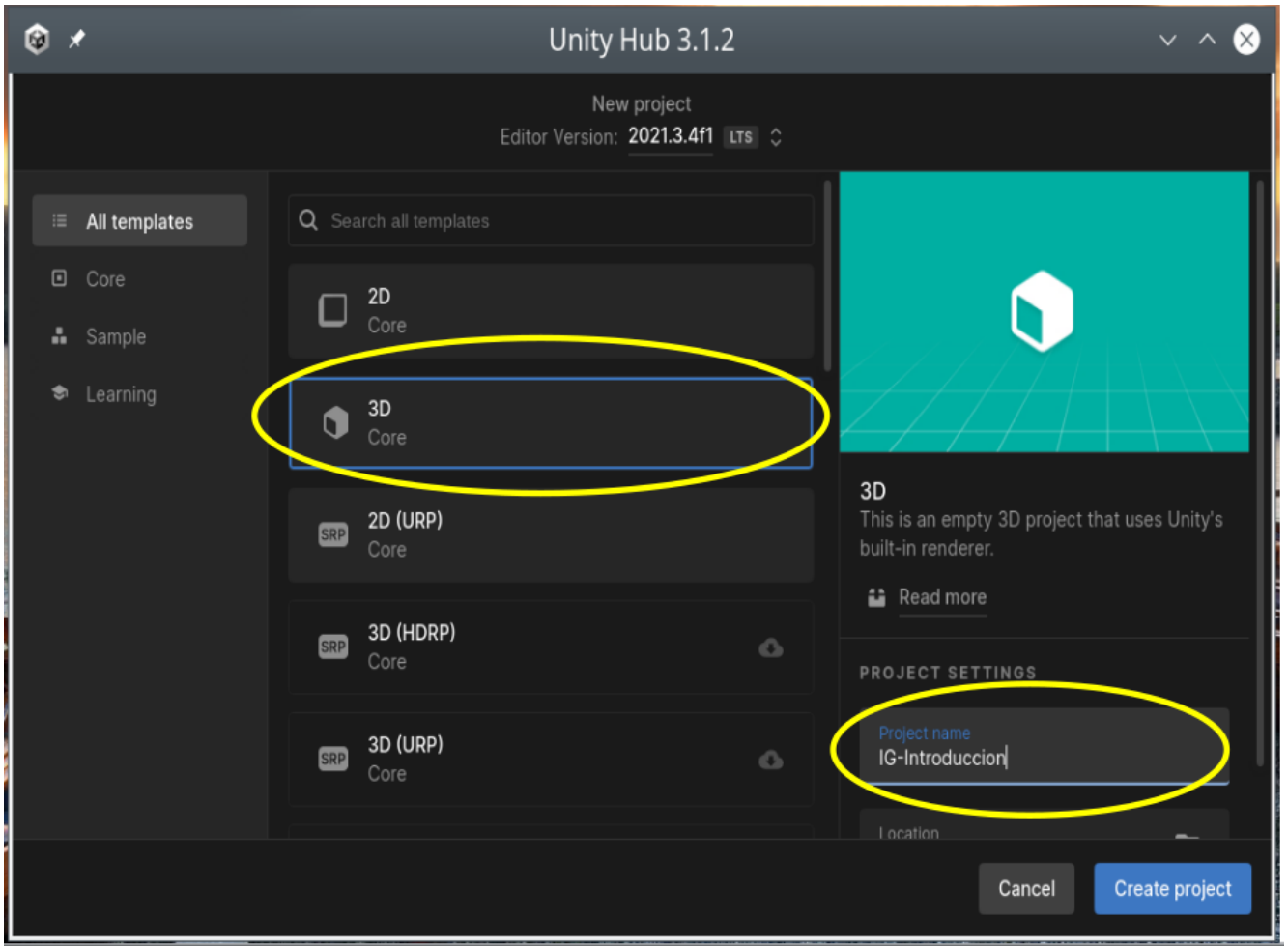
Interacción

Para permite que se interaccione con el programa se utilizan funciones para comprobar y obtener eventos de entrada.

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

public class quit : MonoBehaviour
{
  void Start()
  {
  }

  void Update()
  {
    if (Input.GetKeyDown(KeyCode.Escape))
    {
      Application.Quit();
    }
  }
}
```

Ejemplo básico

Crea un proyecto

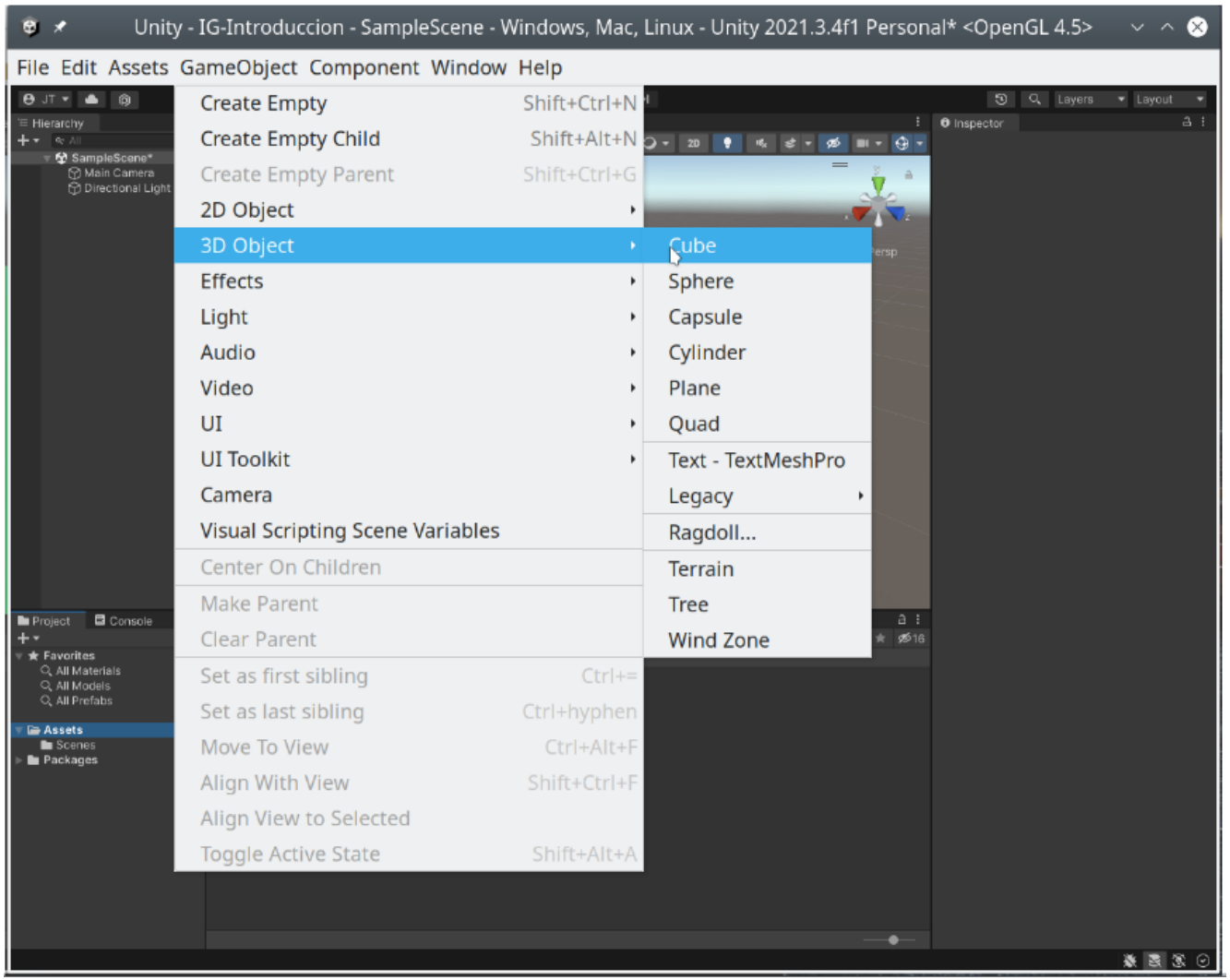


Ejemplo básico

1. Crea un cubo en la escena
2. En Panel Inspector cambia su escala a 10,10,10
3. En panel Project haz Click derecho en Assests y selecciona *CreateFolder*.
4. En el folder creado haz click derecho y selecciona *CreateMaterial*
5. En Inspector cambia el Albedo para asignarle color
6. Arrastra el assests del material al cubo en el panel Hierarchy
7. Selecciona la cámara y orientarla para que se vea el cubo
8. Crea un folder y llámalo script
9. Crea un script en él. Llámalo Rotar y haz doble click para editarlo.
10. Añade la línea $transform.Rotate(0, Angle, 0, Space.Self);$ en el método update y la variable $public float Angle;$
11. Arrastra el script desde el panel Project al cubo en el panel Hierarchy.
12. En el inspector del cubo buscar el script y asignale valor a Angle.
13. Play!

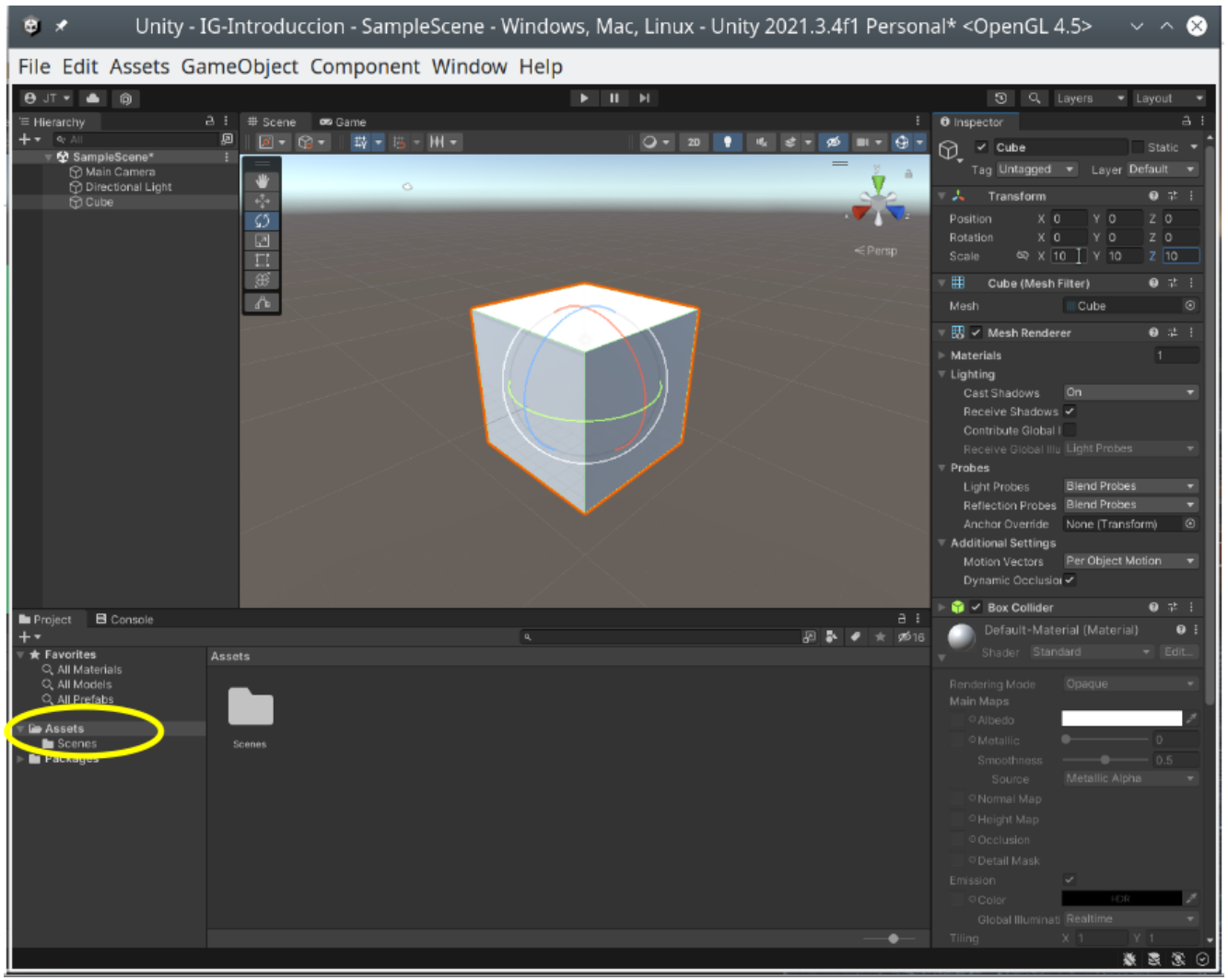
Ejemplo básico

1. Crea un cubo en la escena



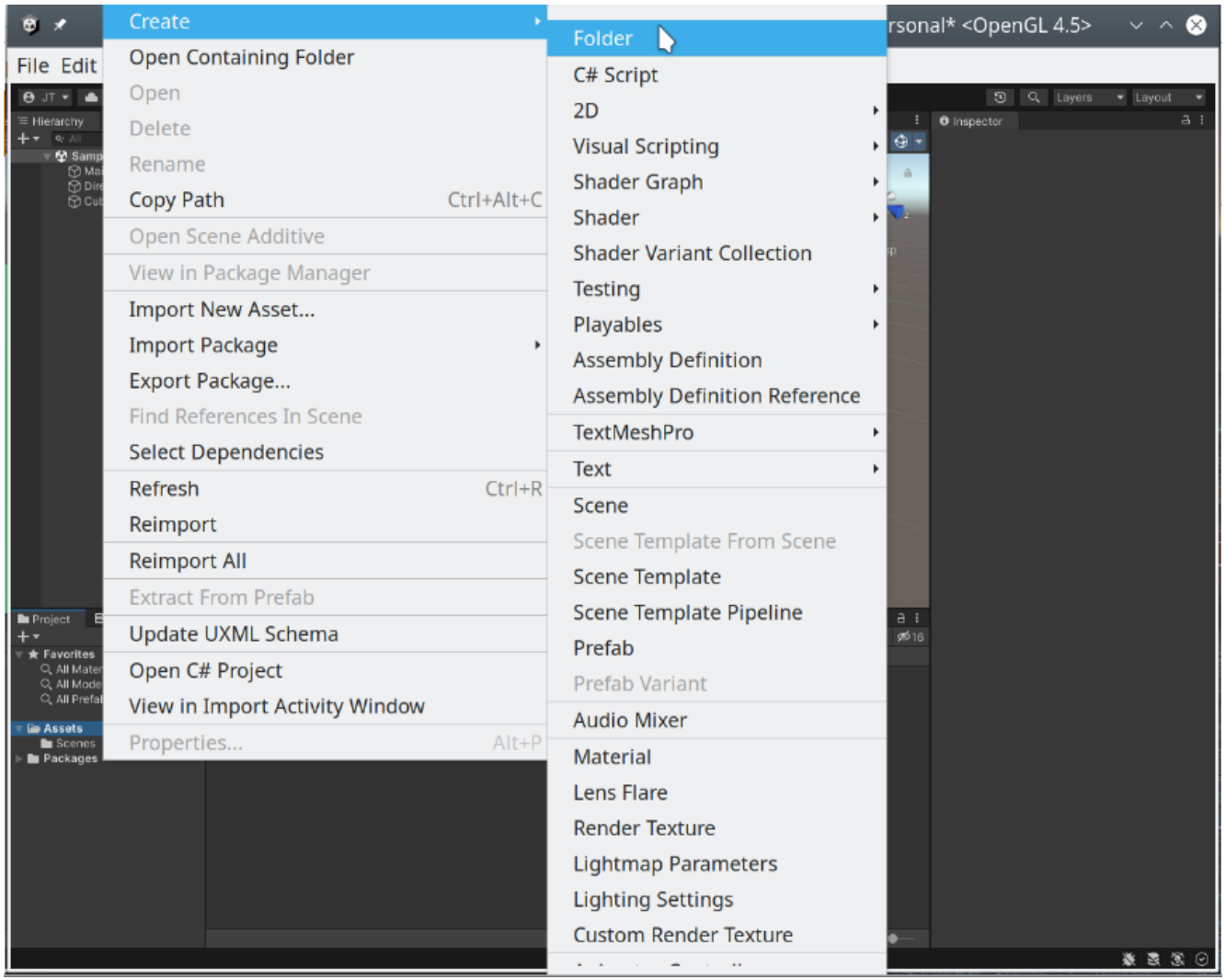
Ejemplo básico

2. En Panel Inspector cambia su escala a 10,10,10



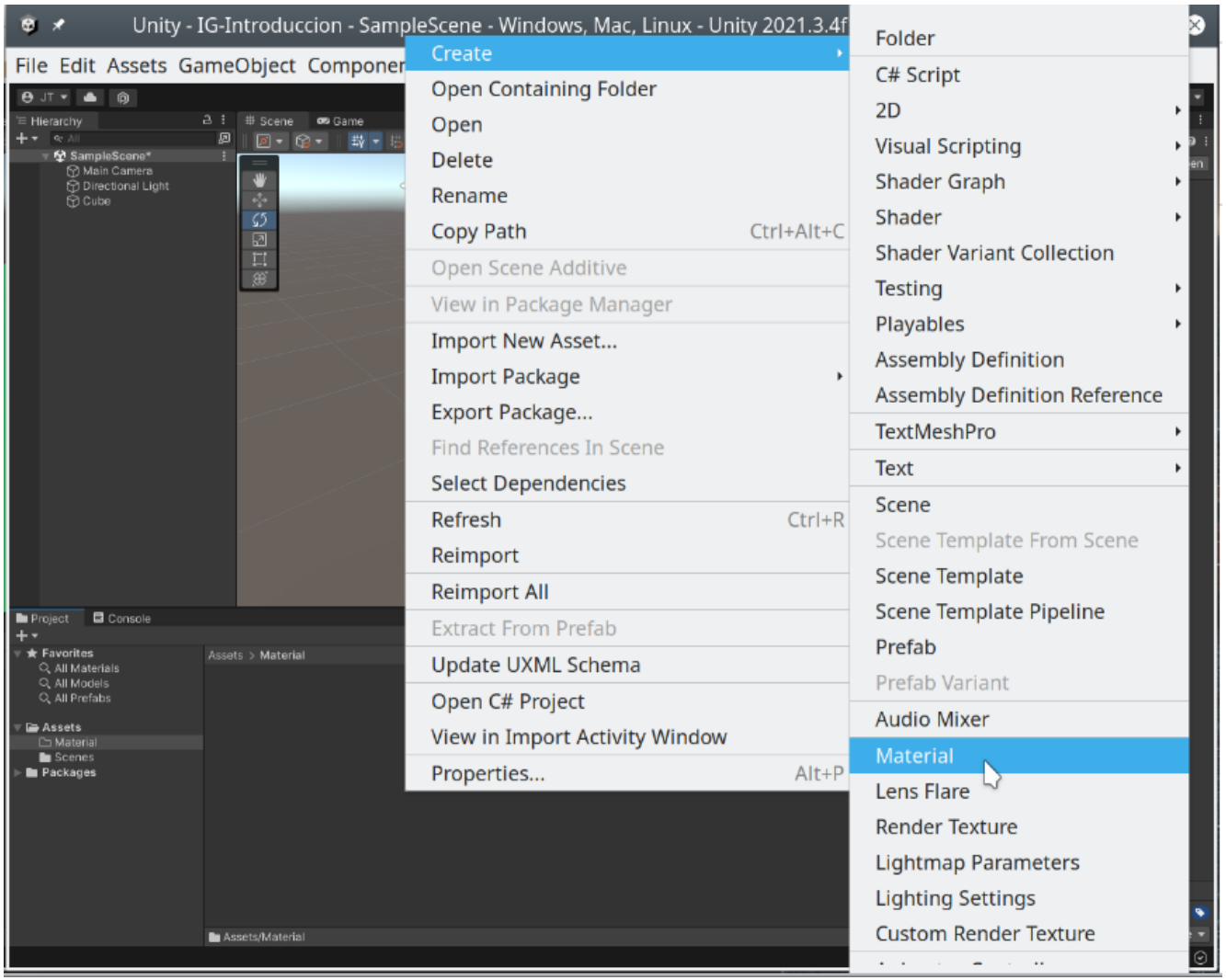
Ejemplo básico

3. En panel Project haz Click derecho en Assests y selecciona *CreateFolder*, llámalo Material



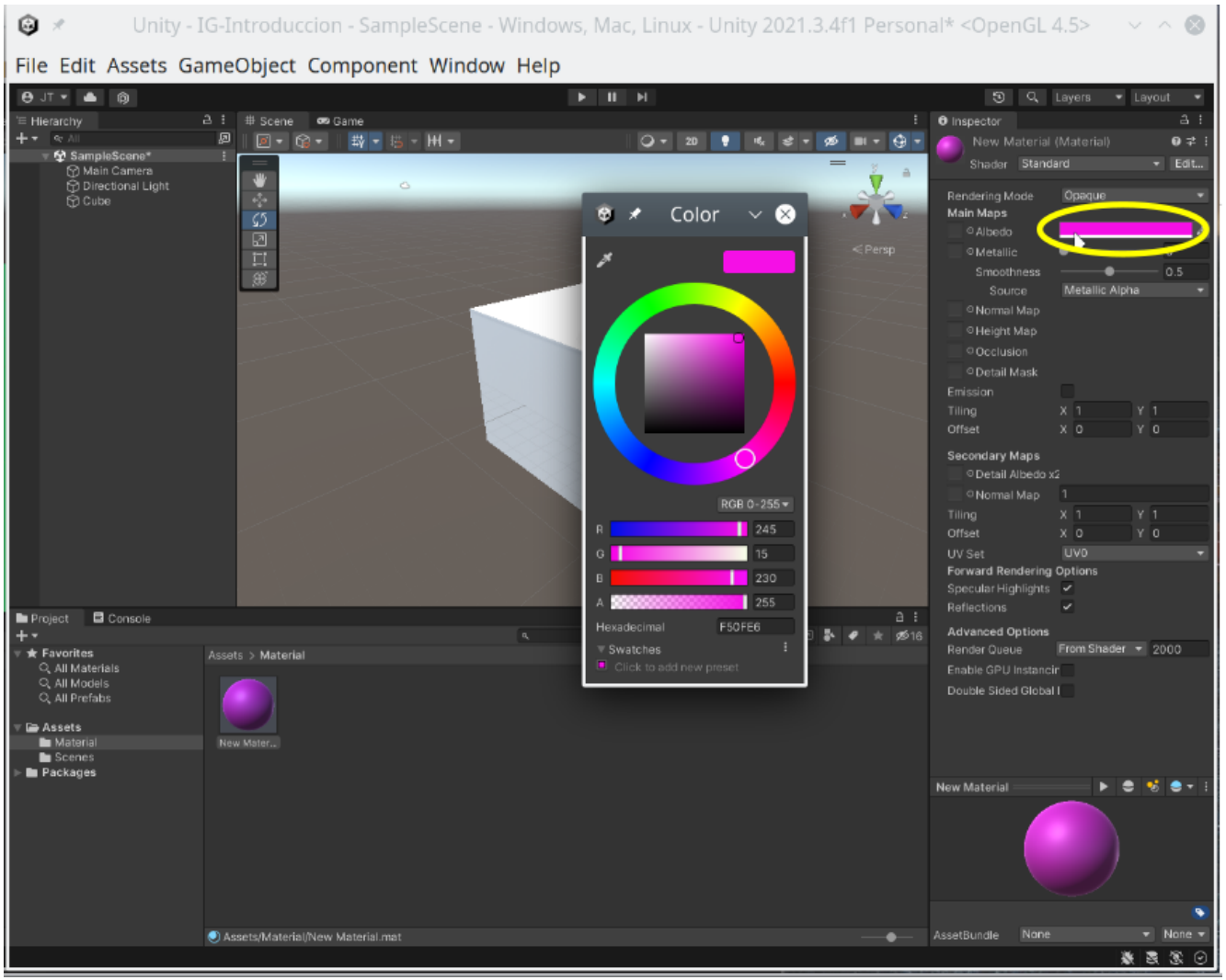
Ejemplo básico

4 En el folder creado haz click derecho y selecciona *CreateMaterial*



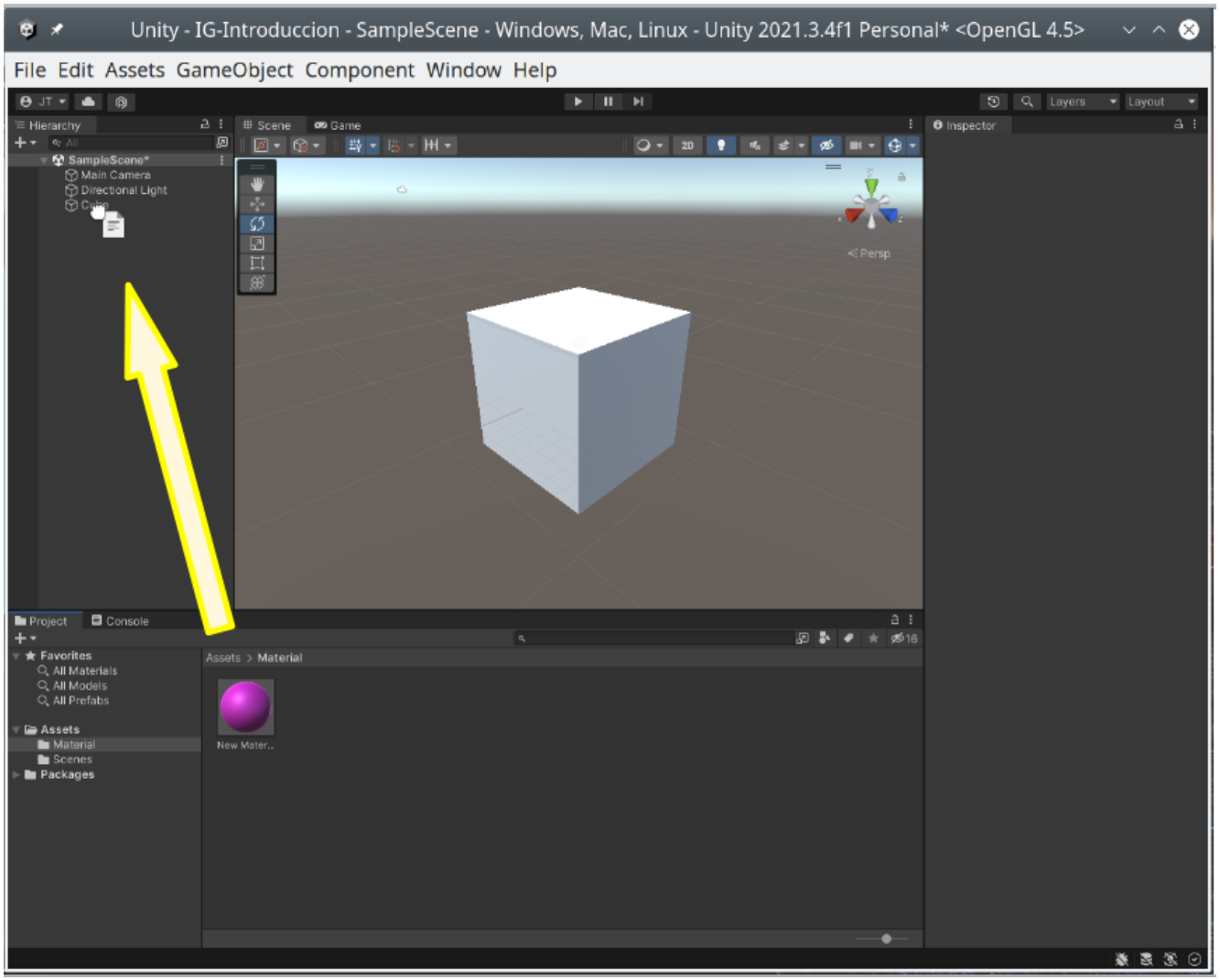
Ejemplo básico

5 En el panel Inspector cambia el Albedo para asignarle color



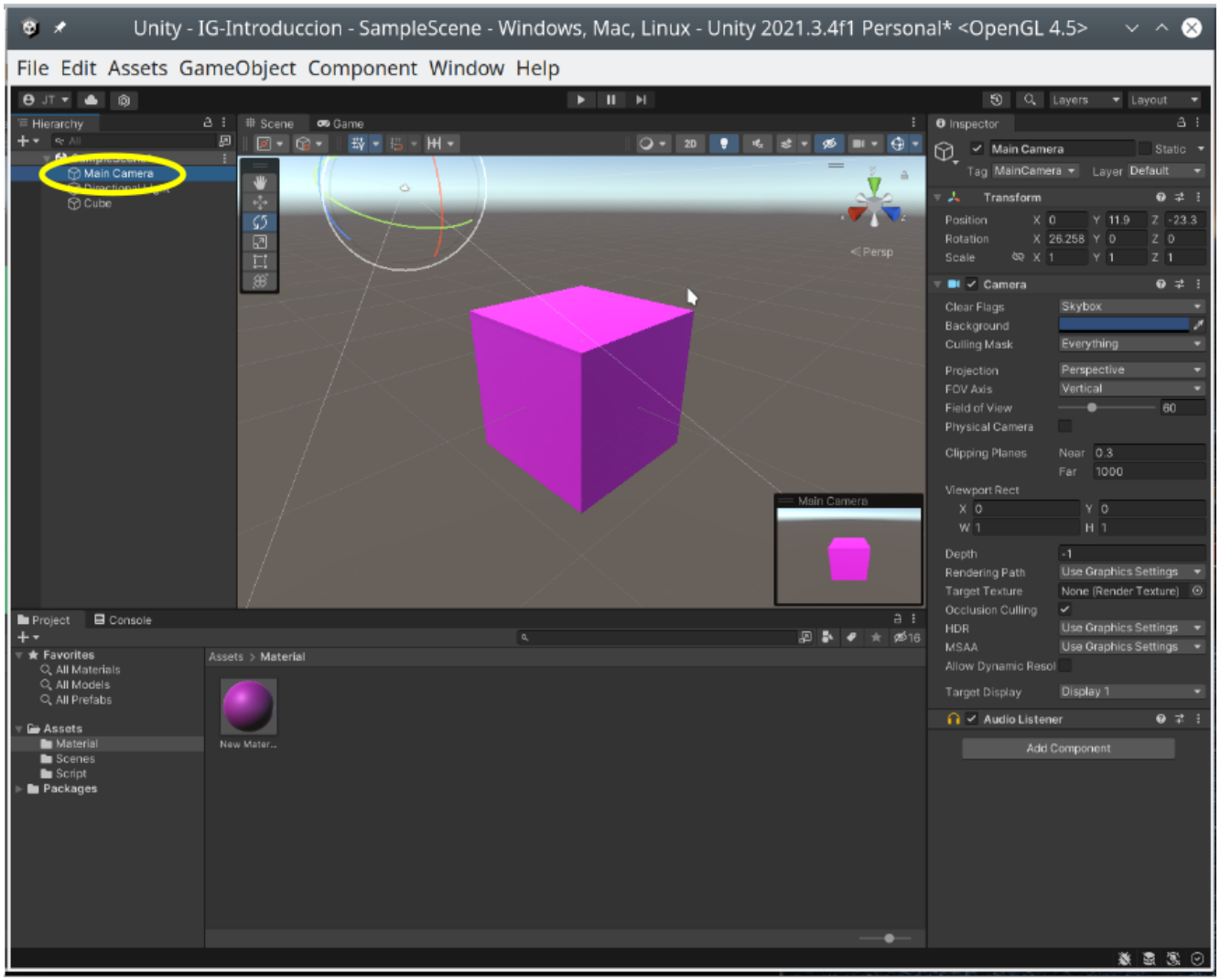
Ejemplo básico

6 Arrastra el assests del material al cubo en el panel Hierarchy



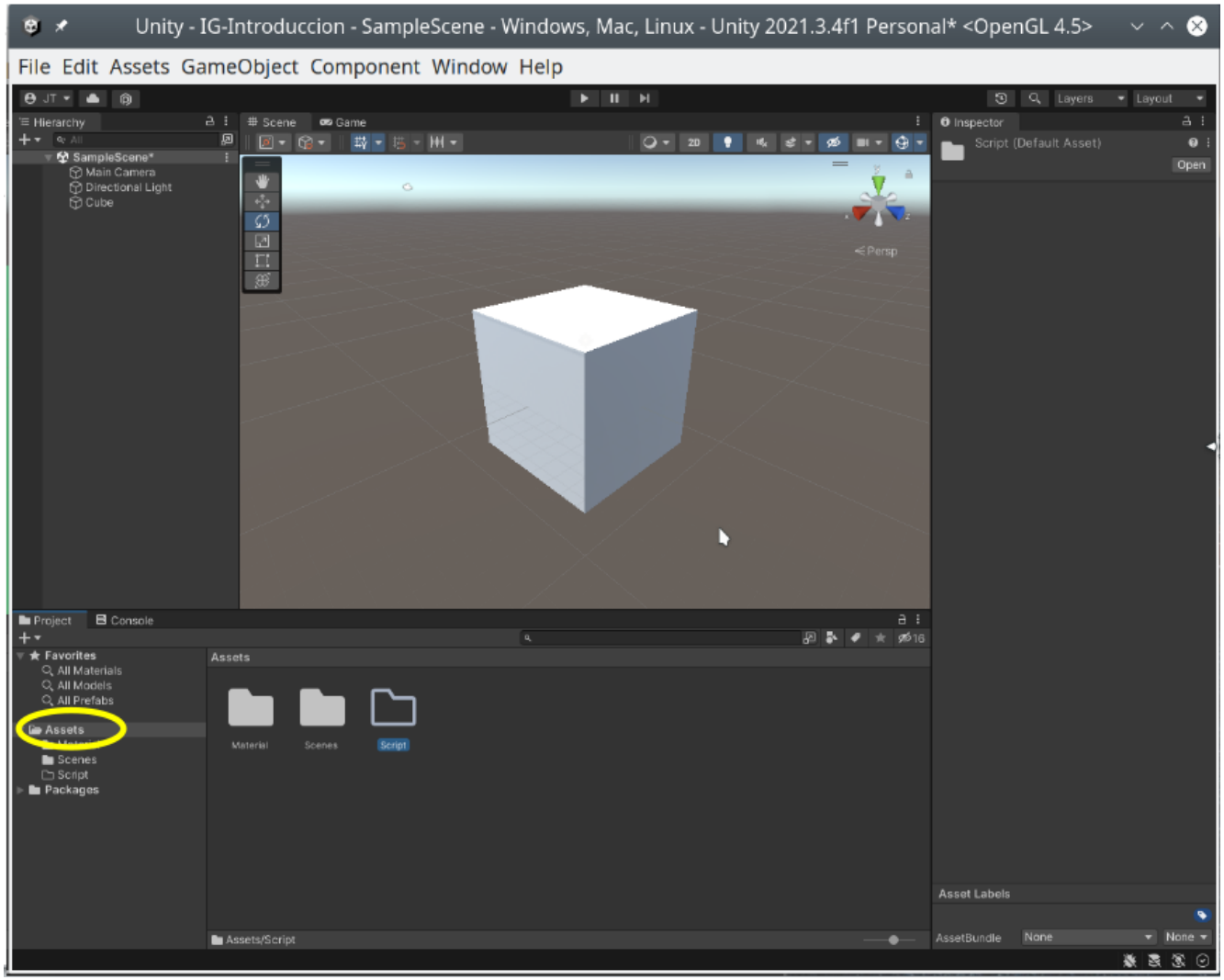
Ejemplo básico

7 Selecciona la cámara y orientarla para que se vea el cubo



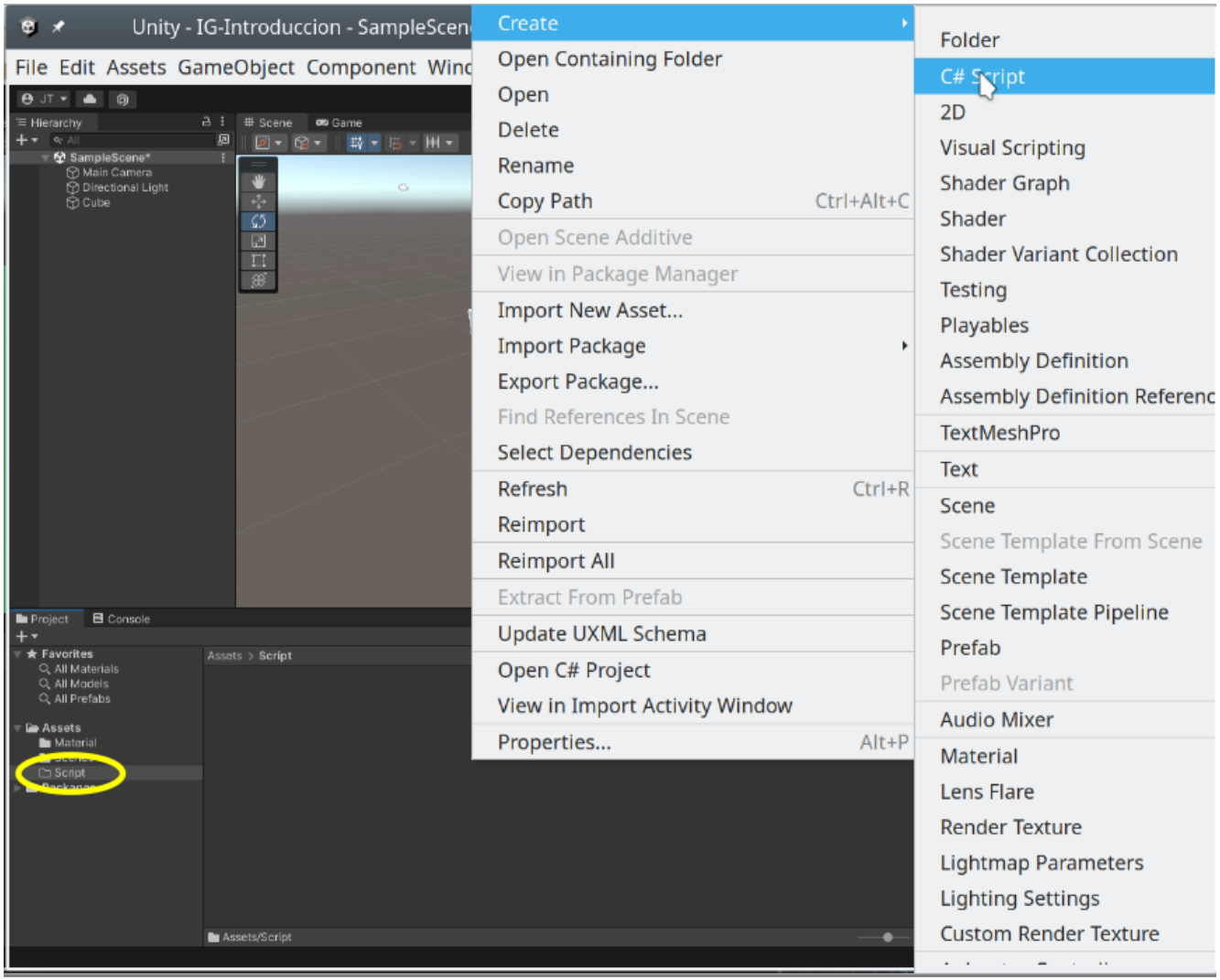
Ejemplo básico

8. Crea un folder en el proyecto y llámalo script.



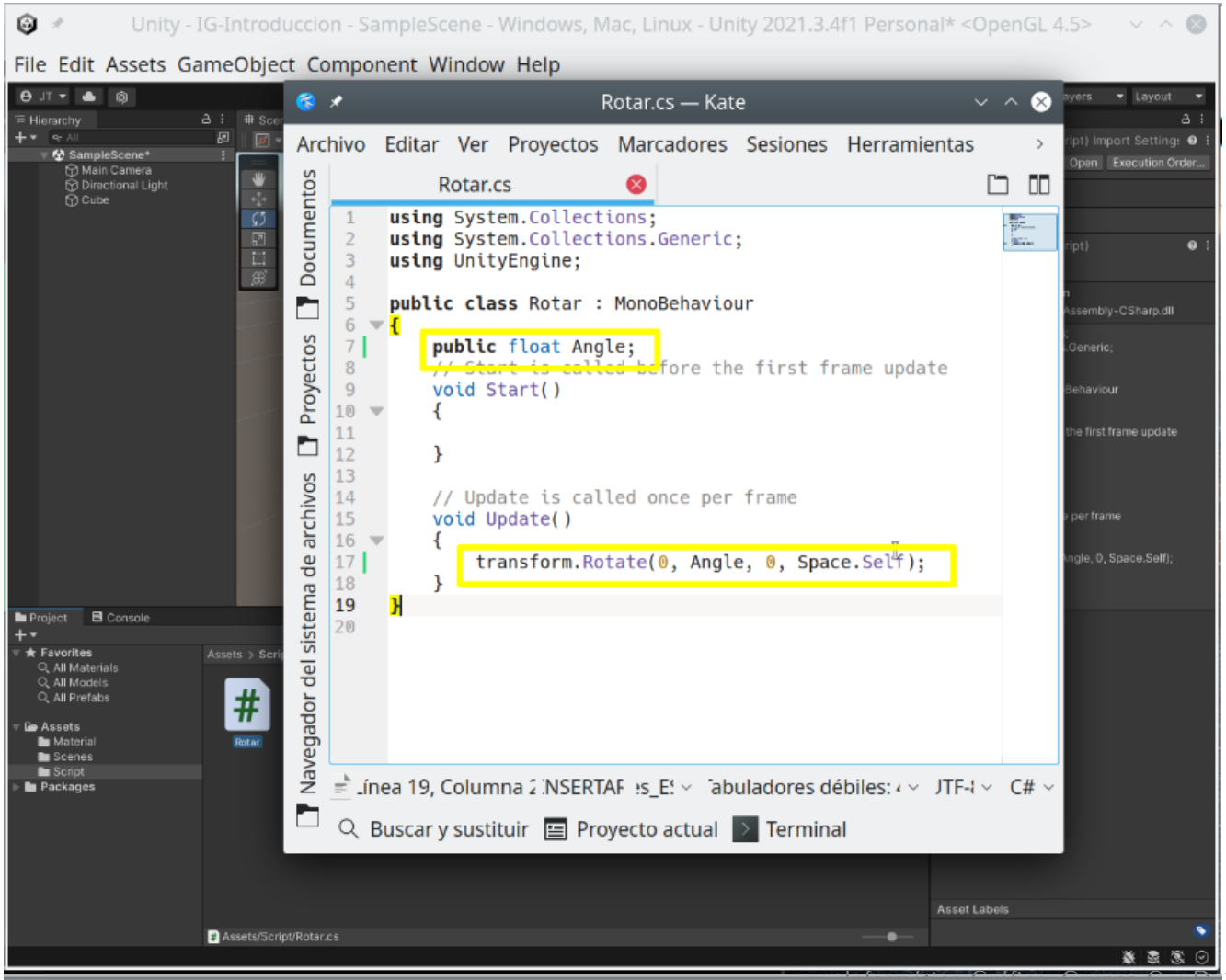
Ejemplo básico

9. Crea un script en él. Llámalo Rotar y haz doble click para editarlo.



Ejemplo básico

10 Añade la línea $transform.Rotate(0, Angle, 0, Space.Self);$ en el método update y la variable $public float Angle;$

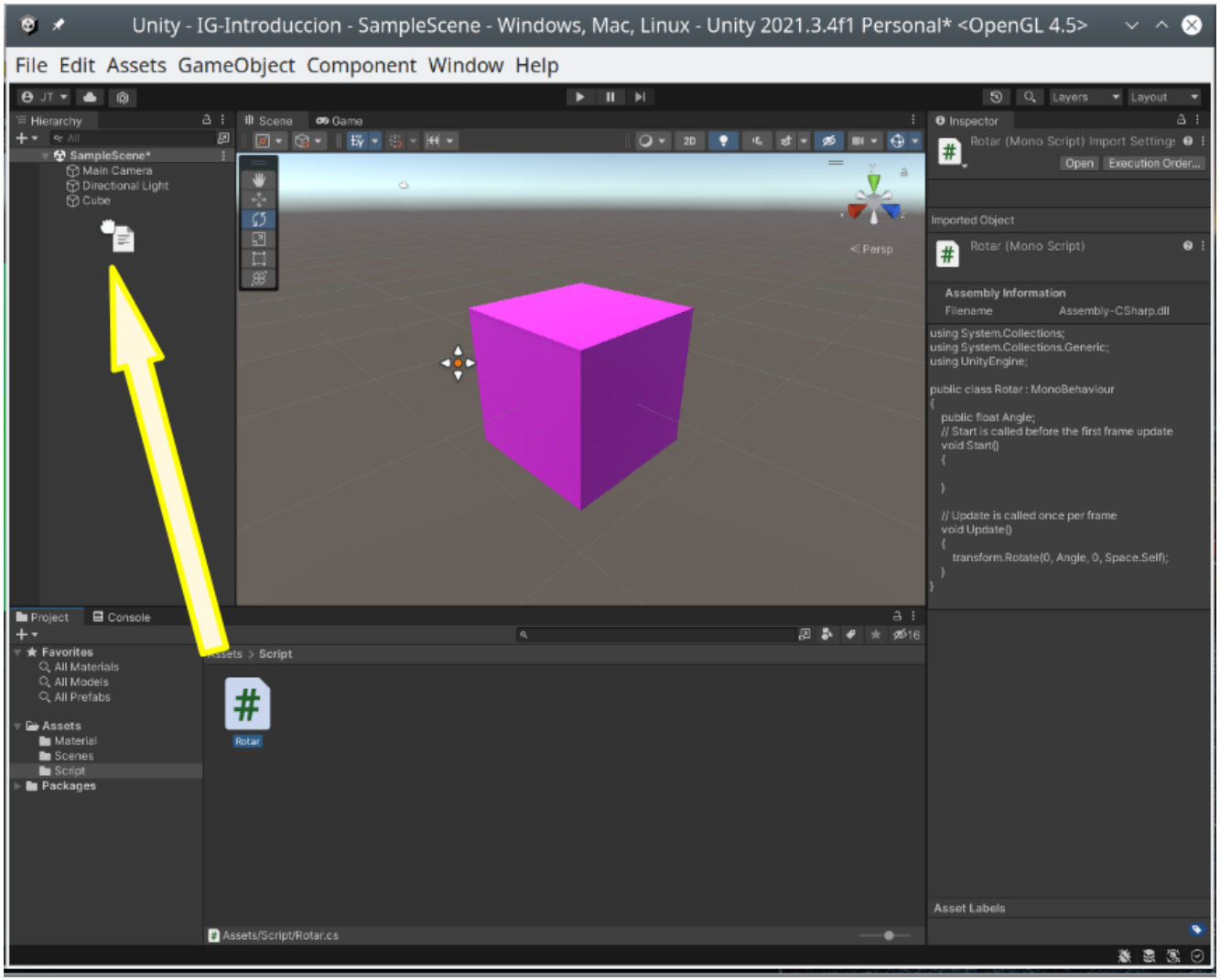


Ejemplo básico

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

public class rotar : MonoBehaviour
{
  public float Angle;

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

  // Update is called once per frame
  void Update()
  {
    transform.Rotate(0, Angle, 0, Space.Self);
  }
}
```

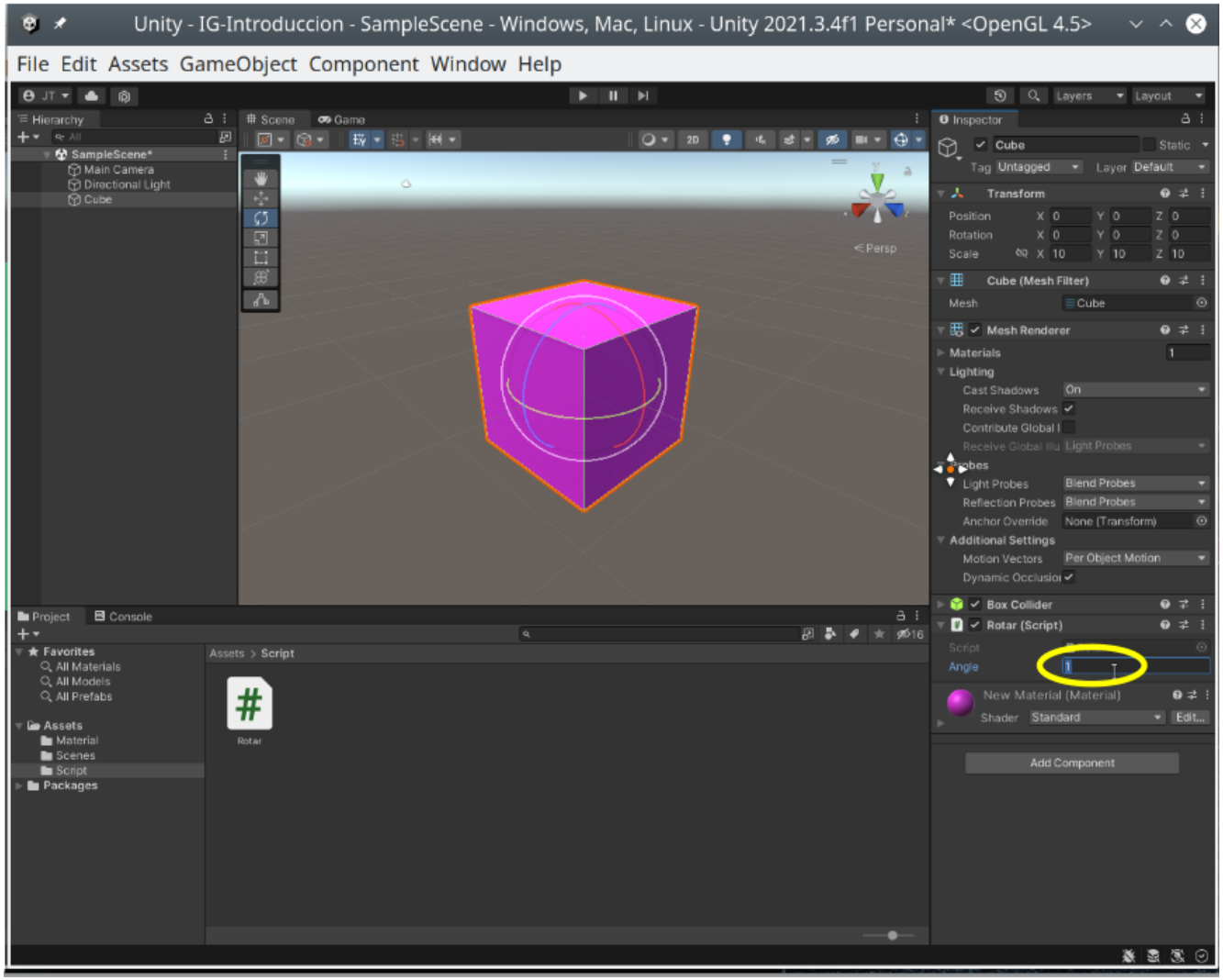
Ejemplo básico

11 Arrastra el script desde el panel Project al cubo en el panel Hierarchy.



Ejemplo básico

12 En el inspector del cubo buscar el script y asignale valor a Angle.



Ejemplo básico

13 Play!