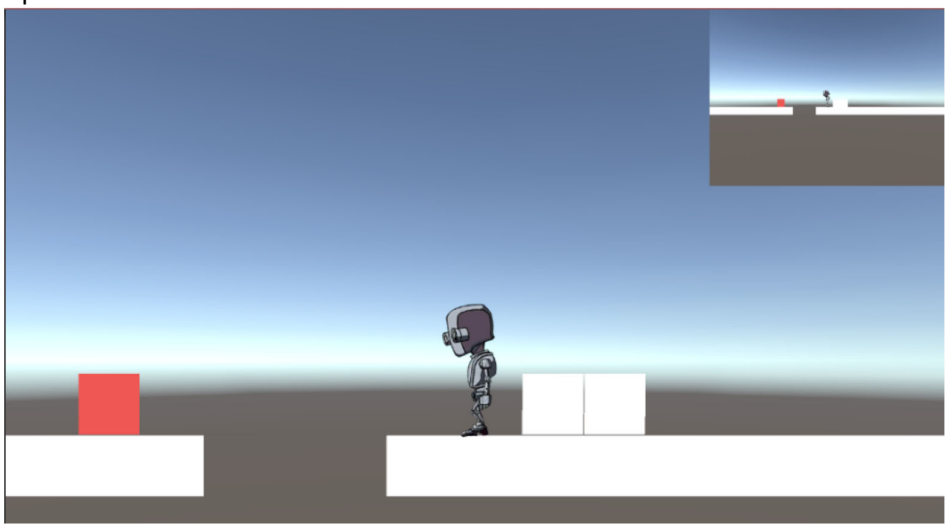
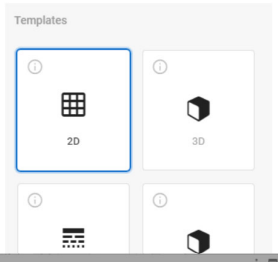
Unity 2020
Escena 2D

1 Introducción

- En este tema vamos a definir una escena 2D muy sencilla para ir introduciéndonos en el uso de Scripts
- Puntos a tratar:
  - Definir escenario 2D
  - Definir Mini Mapa, para visión global del juego
  - Añadir Killzone, para reiniciar la escena (lo copiaremos a la escena de inicio)
  - Definir el comportamiento del objeto respecto a las colisiones y las fuerzas que se le aplican



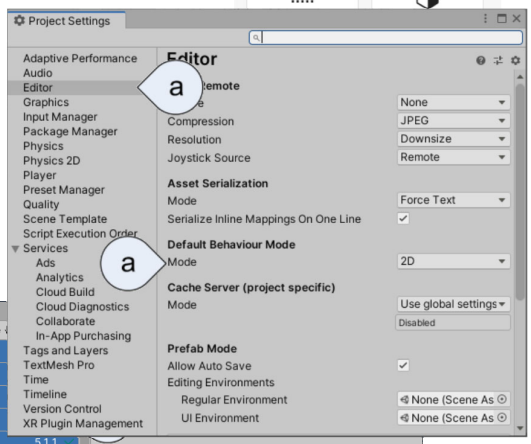
1 Introducción
Entorno de desarrollo 2D



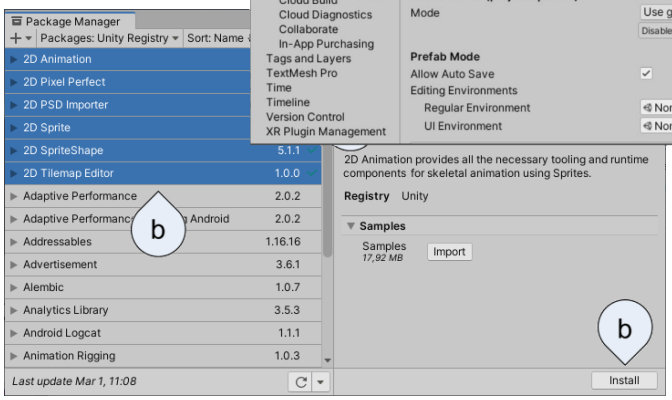
- Si iniciamos un proyecto con la plantilla 2D, no debemos hacer nada de lo que indicamos a continuación.
  Pero, como vamos a añadir una escena 2D a nuestro proyecto 3D, debemos configurar la escena de la siguiente forma:

a) Menú ***Edit*** → ***Project Settings***
   Apartado ***Editor***
   Cambiar ***Default Behaviour Mode*** a ***2D***

   (NOTA: Recuerda cambiar este modo al terminar de trabajar con la escena 2D)



b) Instalar todos los paquetes 2D
   - Menú ***Window*** → ***Package Manager***
   - Seleccionar en el desplegable: ***Packages Unity Registry***
   - Seleccionar los *packages* que empiezan por ***2D*** e instalarlos uno a uno



2 Preparar el escenario
Sprites. Dibujar las plataformas

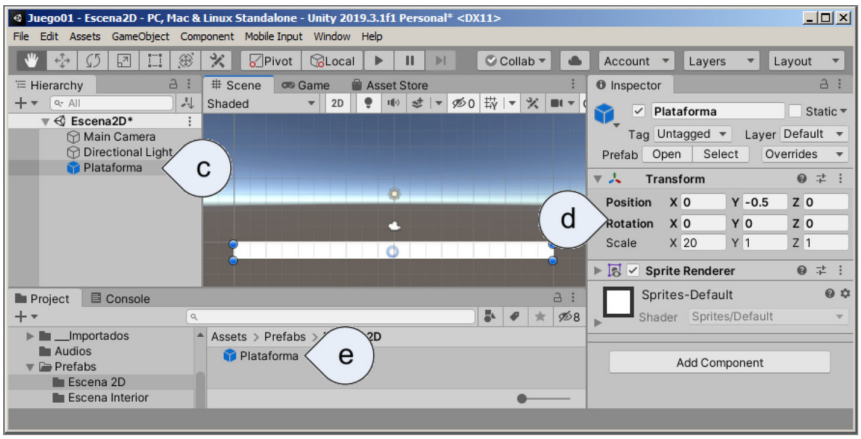
a) Para definir un escenario en 2D es recomendable activar el modo de visualización en 2D

b) Para definir las plataformas de la escena, se utiliza el generador de formas 2D (*sprite*) desde ***Hierarchy***: ***Botón derecho del ratón → 2D Objects → Sprites → Square***

c) Seleccionar el *Sprite* en *Hierarchy* (o *Scene*)

d) Cambiar dimensiones

e) Definir como *Prefab*
   - Assets\Prefabs\Escena 2D



2 Preparar el escenario

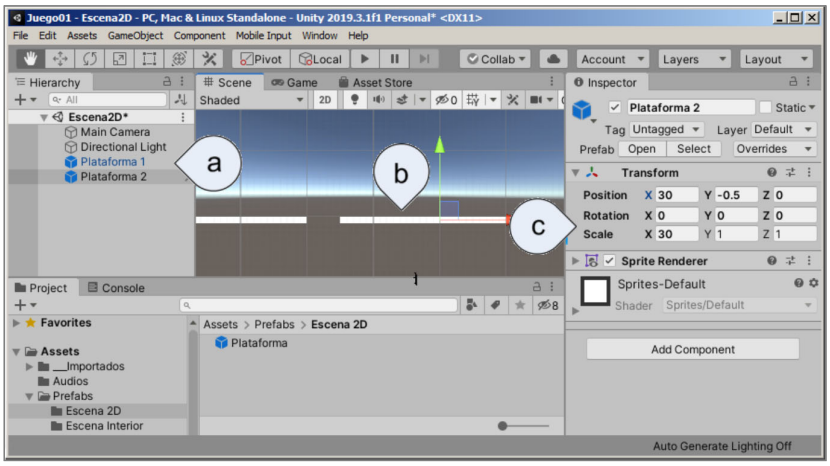
Sprites. Dibujar las plataformas

- Para hacer la otra plataforma

a) Duplicar la plataforma inicial
   - Renombrar las plataformas para diferenciarlas

b) Desplazar la Plataforma 2 (cambiar Position X: 30)

c) Cambiar las dimensiones: Scale X: 30



2 Preparar el escenario

Añadir cámara

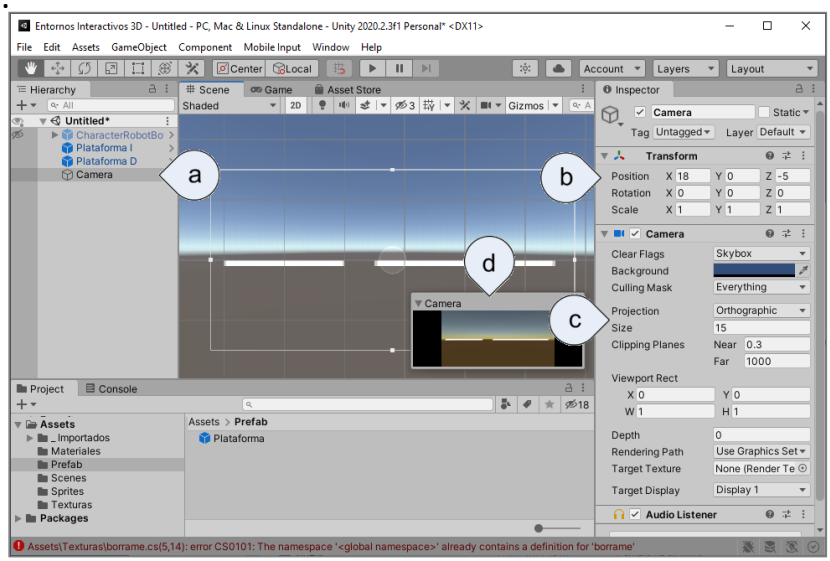
- Si no hay cámara en la escena, añadimos una y establecemos los siguientes parámetros:

a) Seleccionamos la cámara en ***Hierarchy***

b) La posicionamos en X=18; Y=0; Z=-5. ***Rotation*** = 0 y ***Scale*** = 1.

c) Cambiamos los valores:
   - *Projection*: Orthographic
   - *Size* = 15

d) Comprobamos que está bien encuadrado en la previsualización



2 Preparar el escenario

Añadir personaje

- Para añadir el personaje a la escena:

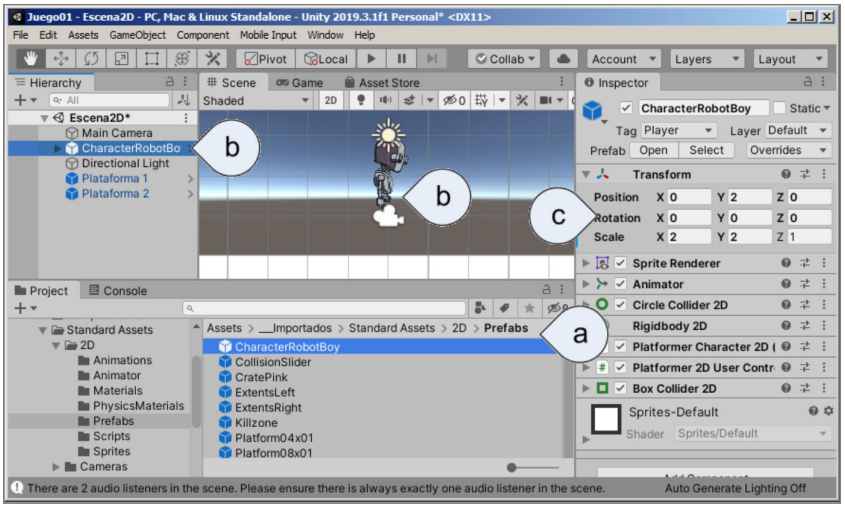
a) Arrastramos a la escena el personaje ***CharacterRobotBoy*** desde la carpeta *Standard Assets\2D\Prefabs*

b) Arrastramos el personaje a *Scene* y lo seleccionamos en *Hierarchy*

c) Cambiamos los valores:
   - *Position Y*: 2
   - *Scale X = Scale Y* = 2

d) Ejecutamos la aplicación

Unity no detecta ninguna colisión con las plataformas
- *El personaje cae*
- *Hay que definir las características de colisión de las plataformas*

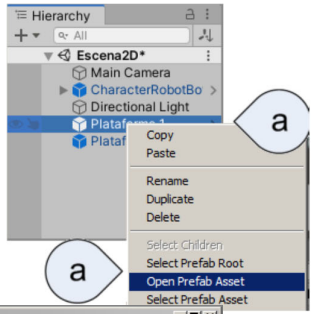


2 Preparar el escenario

Definir colisiones

- Para que los objetos colisionen con el resto de objetos de la escena, es necesario el componente encargado de esa función: *Collider*

a) Editamos el *Prefab* ***Plataforma***, para que afecte a las dos plataformas insertadas
   - *Botón derecho sobre una de las plataformas* → ***Open Prefab Asset***



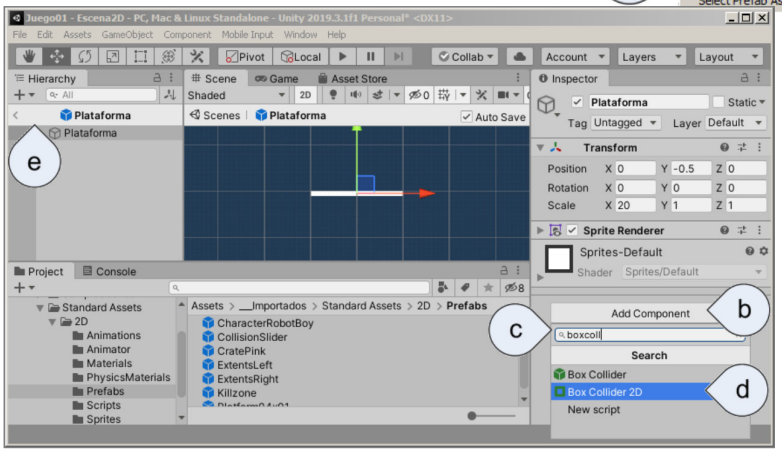
b) Botón ***Add Component***

c) Buscar *Boxcollider*

d) Seleccionar ***Box Collider 2D***

e) Salir del editor del *Asset*

Ejecutar la aplicación
- *El personaje puede andar por las plataformas*



2 Preparar el escenario

Añadir cámara dirigida al personaje

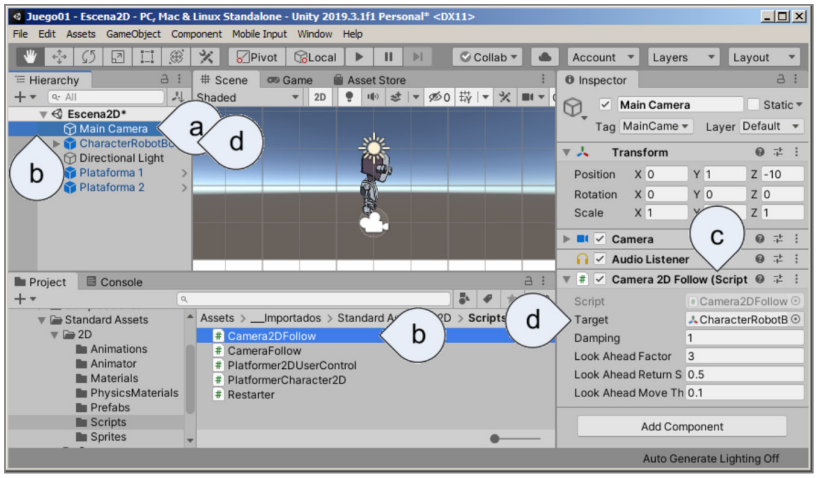
- Vamos a añadir una cámara que nos permita seguir al personaje
  - Para lograrlo, vamos a añadir un *Script* a la *Main Camera,* disponible en *Standard Assets*

a) Seleccionamos ***Main Camera*** en *Hierarchy*

b) Arrastramos el *Script* ***Camera2DFollow*** desde *Project* a ***Main Camera*** en *Hierarchy*
   - *El script se encuentra en Standard Assets\2D\Scripts*

c) Comprobamos que el script se ha añadido como componente de ***Main Camera***

d) Arrastramos ***CharacterRobotBoy*** desde *Hierarchy* hasta ***Target*** en *Inspector*



Ejecutar la aplicación

– *La cámara sigue al personaje*

3 Completar el escenario

Añadir un *Mini Mapa*

- El campo de vista de la cámara no nos permite ver bien el escenario a medida que nos vamos desplazando
  - Vamos a añadir un *Mini Mapa*, que nos permita ver el escenario en su conjunto

a) Creamos una cámara llamada *MiniMap*. *Menú* ***Game Object → Camera***

b) Position *Z = -20* para ver más parte del escenario

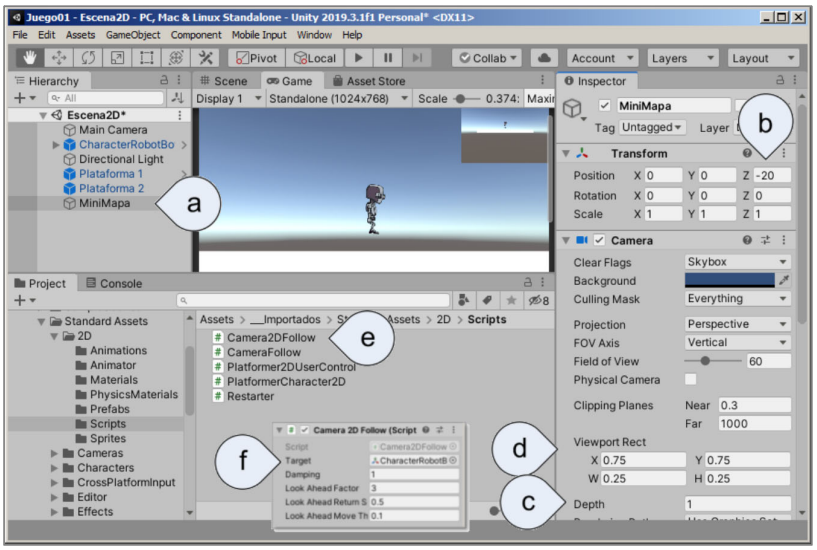
c) Depth = 1
   - *Para que se visualice por encima de Main Camera*

d) Posición y tamaño de la ventana:
   - *X=0.75; Y=0.75*
     Posición del centro de la ventana desde abajo-izda
   - *W=0.25; H=0.25*
     Tamaño de la ventana

e) Arrastrar a la cámara el script ***Camera2DFollow***

f) Arrastrar al Script ***CharacterRobotBoy***

(como apartado anterior)



3 Completar el escenario

Añadir *Killzone*

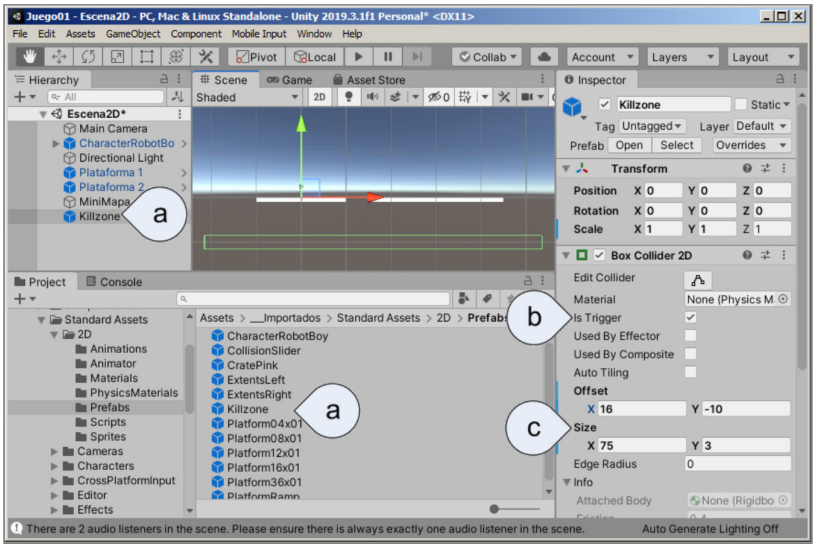
- Si el personaje cae de las plataformas, hay que reiniciar la aplicación
  - Para evitarlo, añadimos una *Killzone*, que reiniciará el escenario cuando se detecte la colisión con el personaje

a) Arrastramos el *Prefab* ***Killzone*** desde *Project* a *Hierarchy*
  - Aparece un rectángulo verde bajo las plataformas, representando la *Killzone*
  - *El script se encuentra en Standard Assets\2D\Prefabs*

b) Comprobamos que está activado el control ***Is Trigger***

c) Cambiamos los parámetros de Offset y Size, para asegurar que el personaje caerá siempre dentro de la zona

Ejecutar la aplicación



4 Añadir Colisiones y Física

Definir colisiones

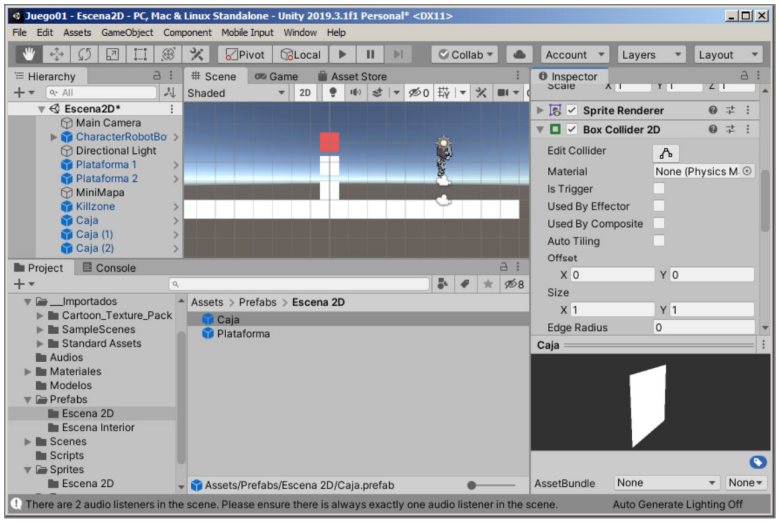
- Para entender el concepto de colisiones:

a) Dibujar y apilar tres cajas
  1. *Crear un Sprite del tipo Square en Hierarchy*
  2. Definirlo *como Prefab*
  3. *Insertar tres cajas*

El personaje atraviesa las cajas

a) Añadir el componente Box Collider 2D al prefab de la caja
  1. Editar el *prefab* ***caja***
  2. *Botón* ***Add component*** → ***Box Collider 2D***

El personaje no atraviesa las cajas y éstas no se mueven



4 Añadir Colisiones y Física

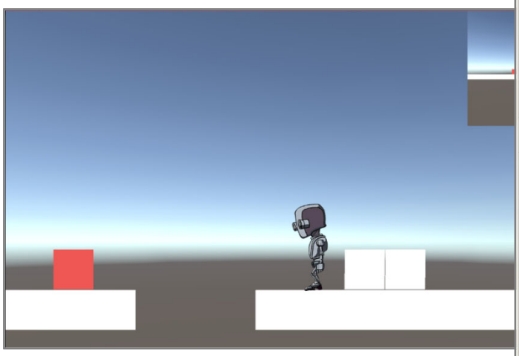
Definir física del objeto

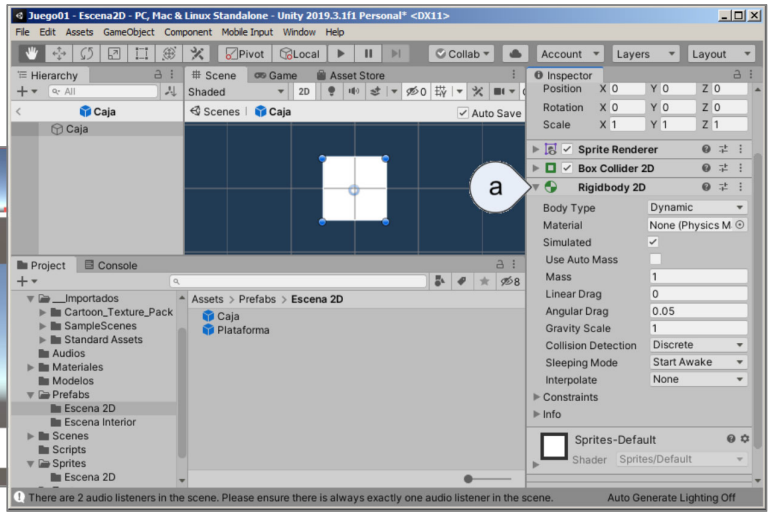
- Para que un GameObject responda a la Física (fuerzas, gravedad…) es necesario añadirle el componente RigidBody ó RigidBody2D

a) Añadir el componente RigidBody2D al prefab de la caja

1. Editar el *prefab* ***caja***
2. *Botón* ***Add component*** → ***RigidBody2D***

a) Ejecutar la aplicación e intentar pasar las tres cajas de una plataforma a otra





5 Añadir *Kill Zone* a la escena de Inicio

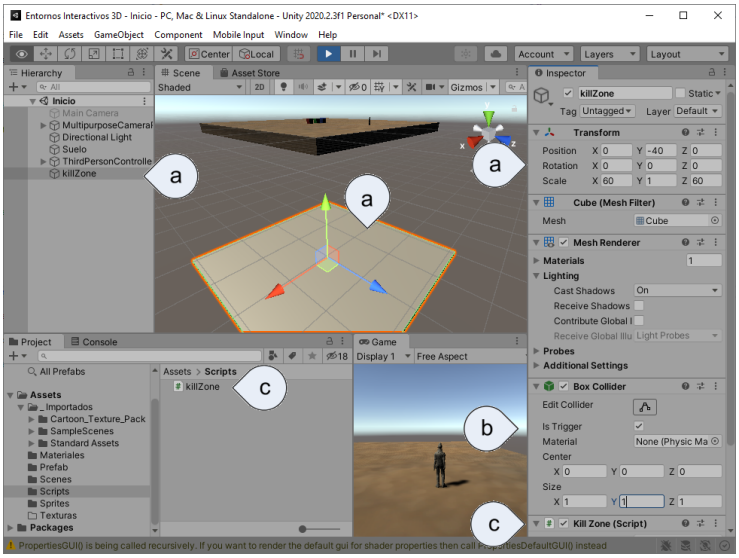
Definir la *Kill Zone*

- Abrimos la escena de Inicio y aprovechamos el script del *prefab* KillZone usado en la escena 2D:

a) Dibujamos el cubo que servirá de Kill Zone
   - *Posición: 0, -40, 0*
   - *Rotación*: 0, 0, 0
   - *Escala: 80, 1, 80*

b) Añadir un *Box Collider* al cubo
   - *Centro*: 0, 0, 0
   - *Dimensiones: 1, 1, 1*
   - Activar *Is Trigger*

c) Crear el script y añadirlo al cubo



NOTAS:

- Las dimensiones del cubo pueden ser menores (1) si las dimensiones del Box Collider son mayores (60)
- Se puede definir como prefab para reutilizarlo

5 Añadir *Kill Zone* a la escena de Inicio

Escribimos el Script a partir del de la escena 2D

- Copiamos y pegamos la función ***OnTriggerEnter2D*** en el script:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;  //  Añadir tipos definidos en el espacio de nombres UnityEngine.SceneManagement


Script de Unity | 0 referencias
public class killZone : MonoBehaviour
{
    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    void Start()
    {
        //  No hace falta para nada, se puede borrar
    }

    // Update is called once per frame
    Mensaje de Unity | 0 referencias
    void Update()
    {
        //  No hace falta para nada, se puede borrar
    }
    Mensaje de Unity | 0 referencias
    private void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Player")
        {
            SceneManager.LoadScene(SceneManager.GetSceneAt(0).name);
        }
    }
}

```

a) Cargamos el "espacio de nombres" ***UnityEngine.SceneManagement***
   - *Así podremos utilizar los tipos definidos en el espacio de nombres sin necesidad de especificarlos de forma explícita en el código*

b) La función Start y Update (que se añaden por defecto en los scripts nuevos) no son necesarias y se pueden borrar

c) Eliminar la referencia a las 2D del nombre de la función y del tipo:
   - *OnTriggerEnter (Collider other)*

d) Comprobar que el *GameObjet* ***FirstPersonController*** tiene el tag ***Player*** asignado