Introduction to Unity

Step 4: more scripting

Piotr Fulmański

Wydział Matematyki i Informatyki,
Uniwersytet Łódzki, Polska

November 18, 2015

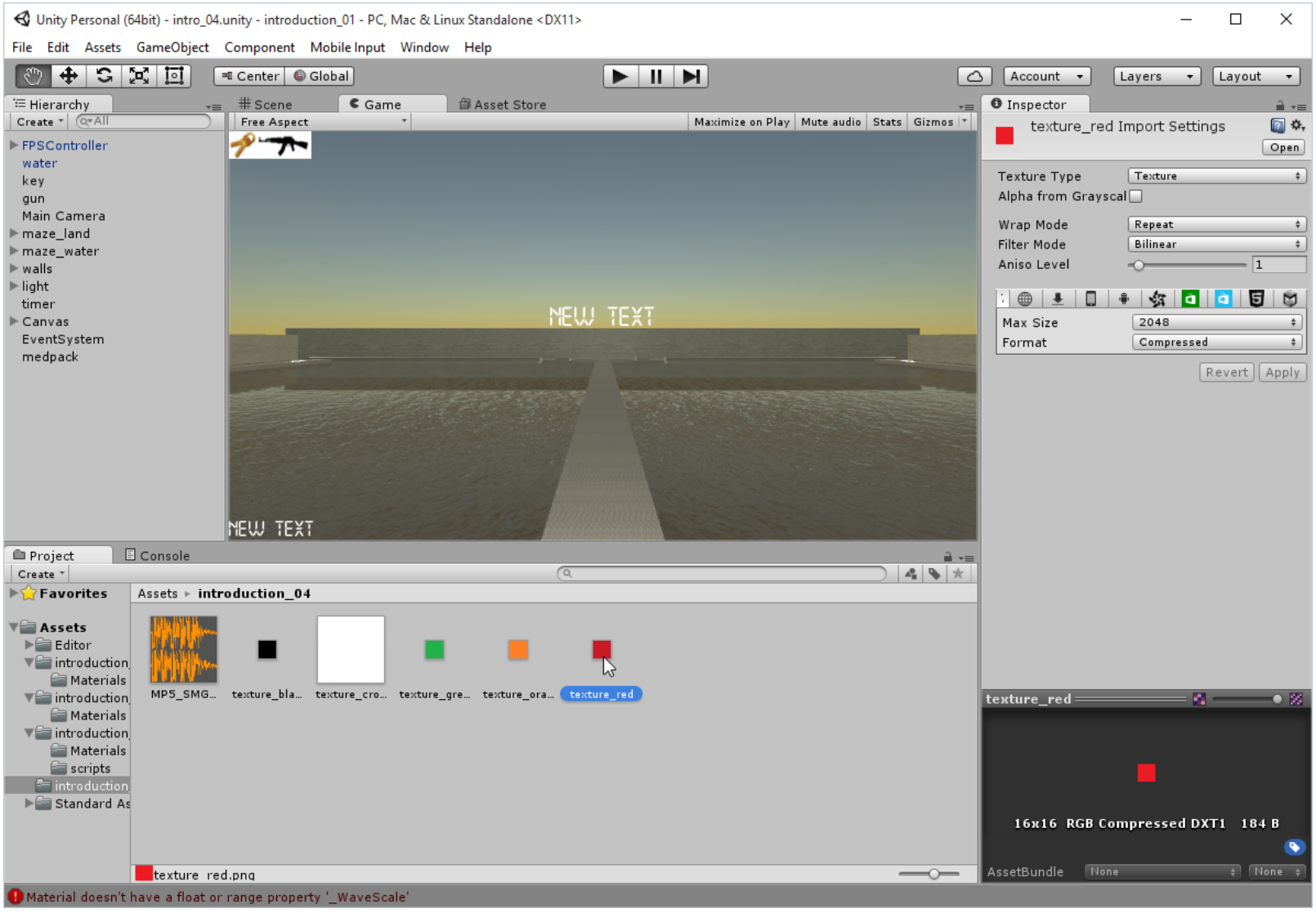
Table of contents

1. Open the previous project (scene `intro_03`).
2. Duplicate the scene we have been working on so far by saving it as `intro_04` **File | Save Scene As**.

1. Create a new folder called `introduction_04`, inside the `Assets` folder.
2. Find / prepare red, orange, green and black texture for health bar.
3. Find / prepare the gun sound (for example: `http://soundbible.com/2091-MP5-SMG-9mm.html`) and name it `gunshot`.
4. Find / prepare the crosshair texture and name it `texture_croshair`.

Preliminaries

Prepare assets



Create folder for the scripts

1. Select the folder `introduction_04` and from the **Project** window, select **Create** | **Folder**.
2. Rename this folder `scripts`.

Create the health bar

Create a new script

1. Check if the folder `scripts` is selected.
2. From the top menu, select **Assets** | **Create** | **JavaScript**.
3. Doing so should create a new JavaScript script within the folder labeled `scripts`.
4. Rename this script `health_bar`.

Create the health bar

Add the code

```
private var currentHealth: int = 50;
private var currentColor: Texture2D;
public var style: GUIStyle;
public var textureRed: Texture2D;
public var textureGreen: Texture2D;
public var textureOrange: Texture2D;
public var textureBlack: Texture2D;
```

Create the health bar

Add the code

```
function OnGUI()
{
  if (currentHealth >= 67)
    currentColor = textureGreen;
  else if (currentHealth >= 34)
    currentColor = textureOrange;
  else
    currentColor = textureRed;

  style.normal.background = textureBlack;
  GUI.Box(Rect(0, 25, 100, 20), "", style);
  style.normal.background = currentColor;
  GUI.Box(Rect(0, 25, currentHealth, 20), "", style);
}
```

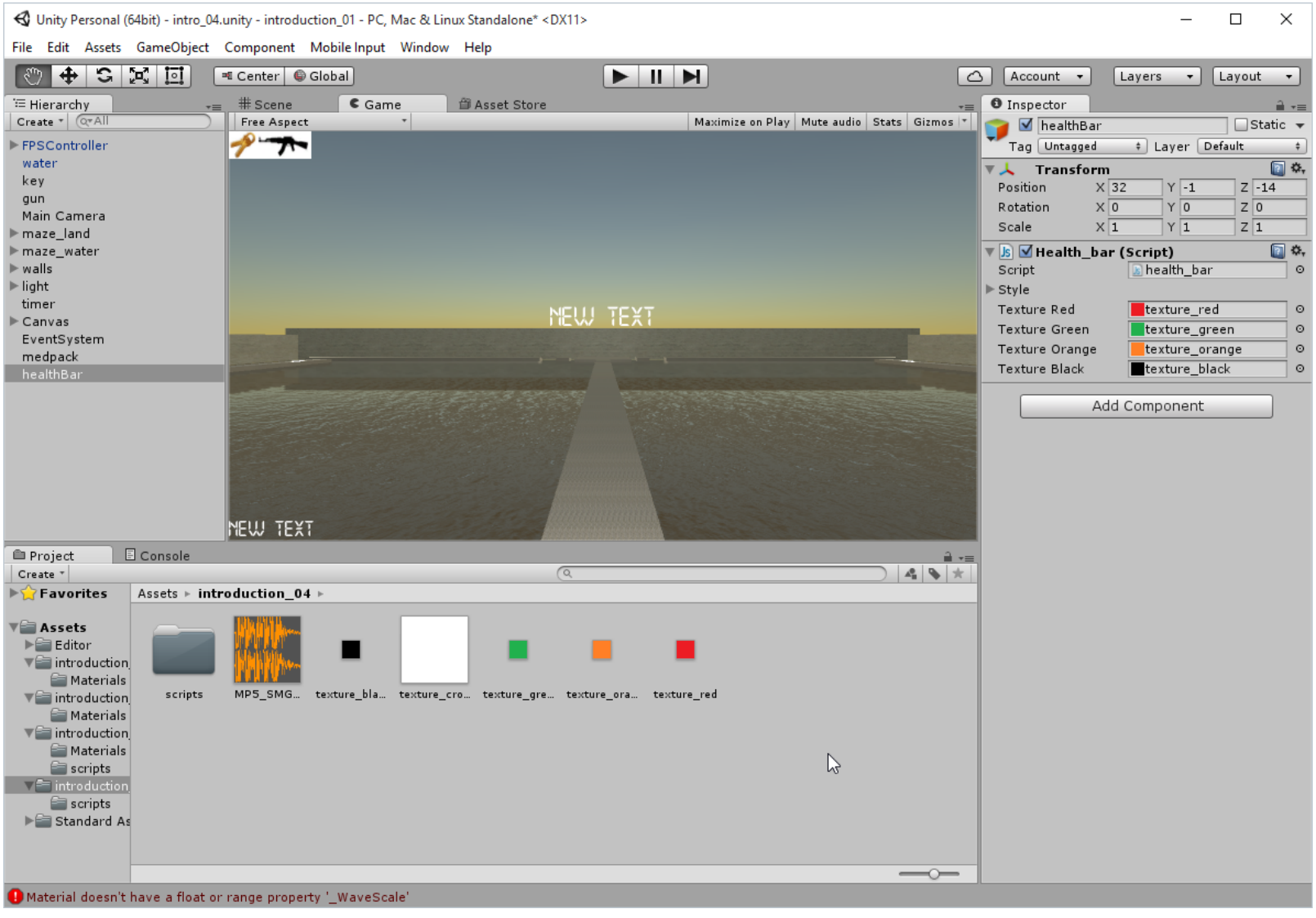
Create the health bar

Create an empty object, link it to the script and bind textures

1. Create an empty object and rename it `healthBar`.
2. Attach the script `healthBar` that we created previously to this object.
3. Locate the **Red** texture by selecting to **Assets** | **chapter4**.
4. Select the object **healthBar** in the **Hierarchy** view, and drag-and-drop the **Red** texture to the right of the variable called `Texture Red` in the component called **Health Bar (Script)**.
5. Repeat the previous two steps for the textures **Green**, **Orange**, and **Black**.

Create the health bar

Create an empty object, link it to the script and bind textures



Create the health bar

Update the health bar when med pack is collected

1. Add the following function to the `healthBar` script

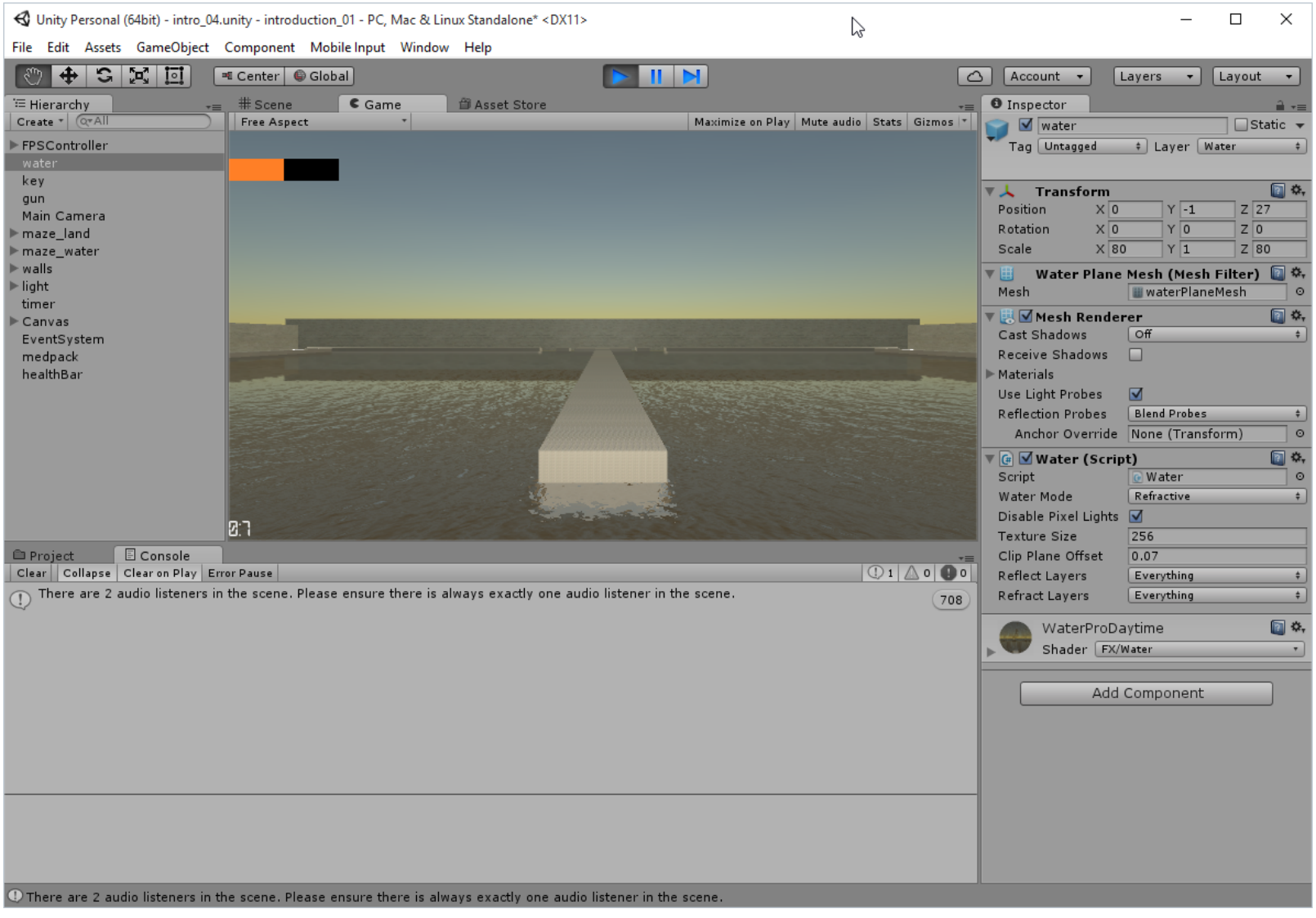
```
public function setHealth(updatedValue: int)
{
  currentHealth = updatedValue;
}
```

2. Modify the `collision_detection` script with highlighted lines

```
if (c.gameObject.tag == "medpack")
{
  health = 100;
  GameObject.Find("healthBar")
    .GetComponent(health_bar)
    .setHealth(health);
}
```

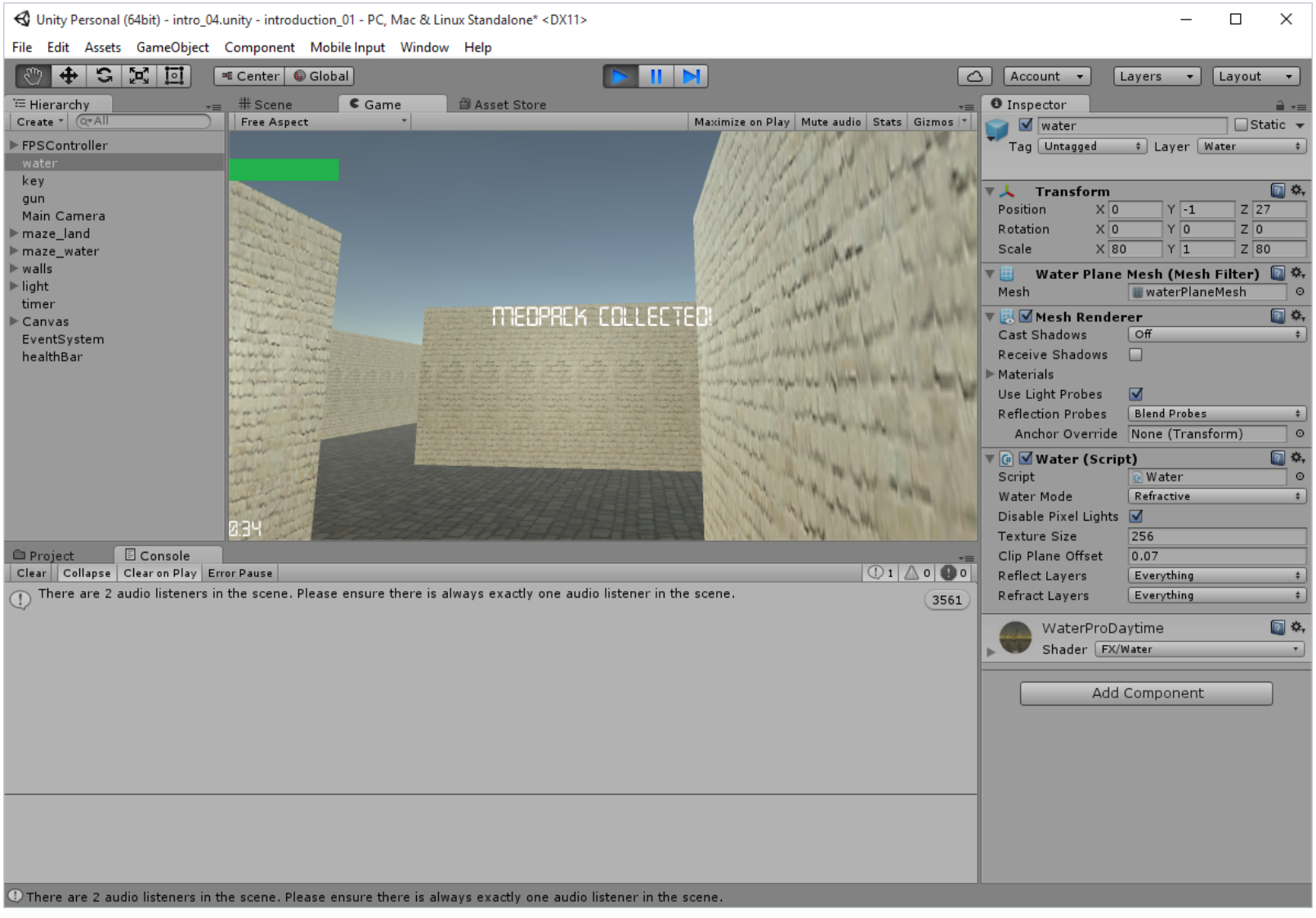
Create the health bar

Update the health bar when med pack is collected



Create the health bar

Update the health bar when med pack is collected



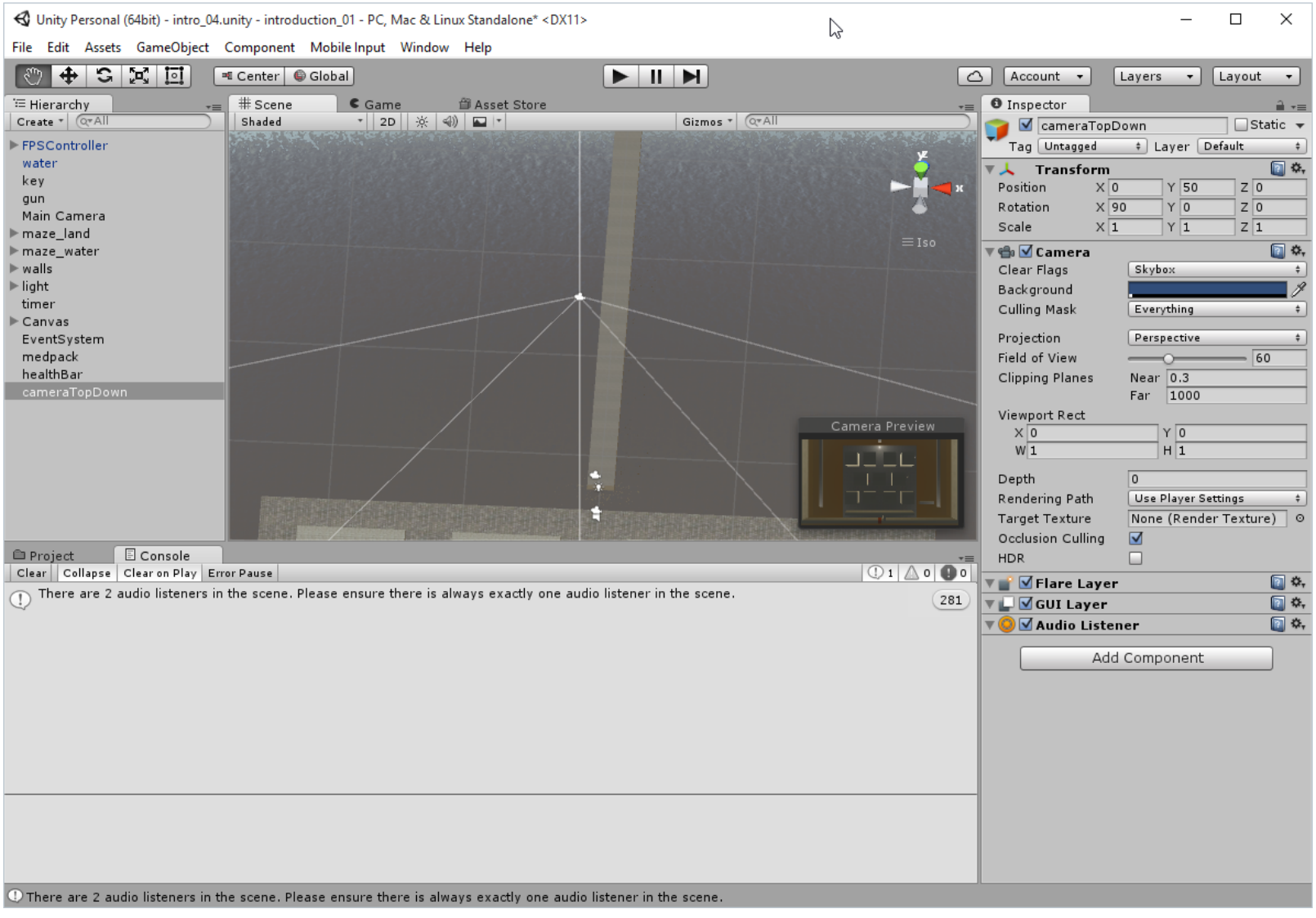
Create a mini-map of the level

Add a camera for a top-down view of the game world

1. Create a new camera (**Game Object** | **Camera**).
2. Rename this camera `cameraTopDown`.
3. Rotate this camera about 90 degrees around the `X` axis, so that its rotation properties are `(x=90, y=0, z=0)`, and change its position to `(x=0, y=30, z=0)`.
4. If we click on this camera in the **Hierarchy** view, and look at the camera preview (the small rectangle in the bottom-right corner of the **Scene** view), we should see the game world from above.

Create a mini-map of the level

Add a camera for a top-down view of the game world



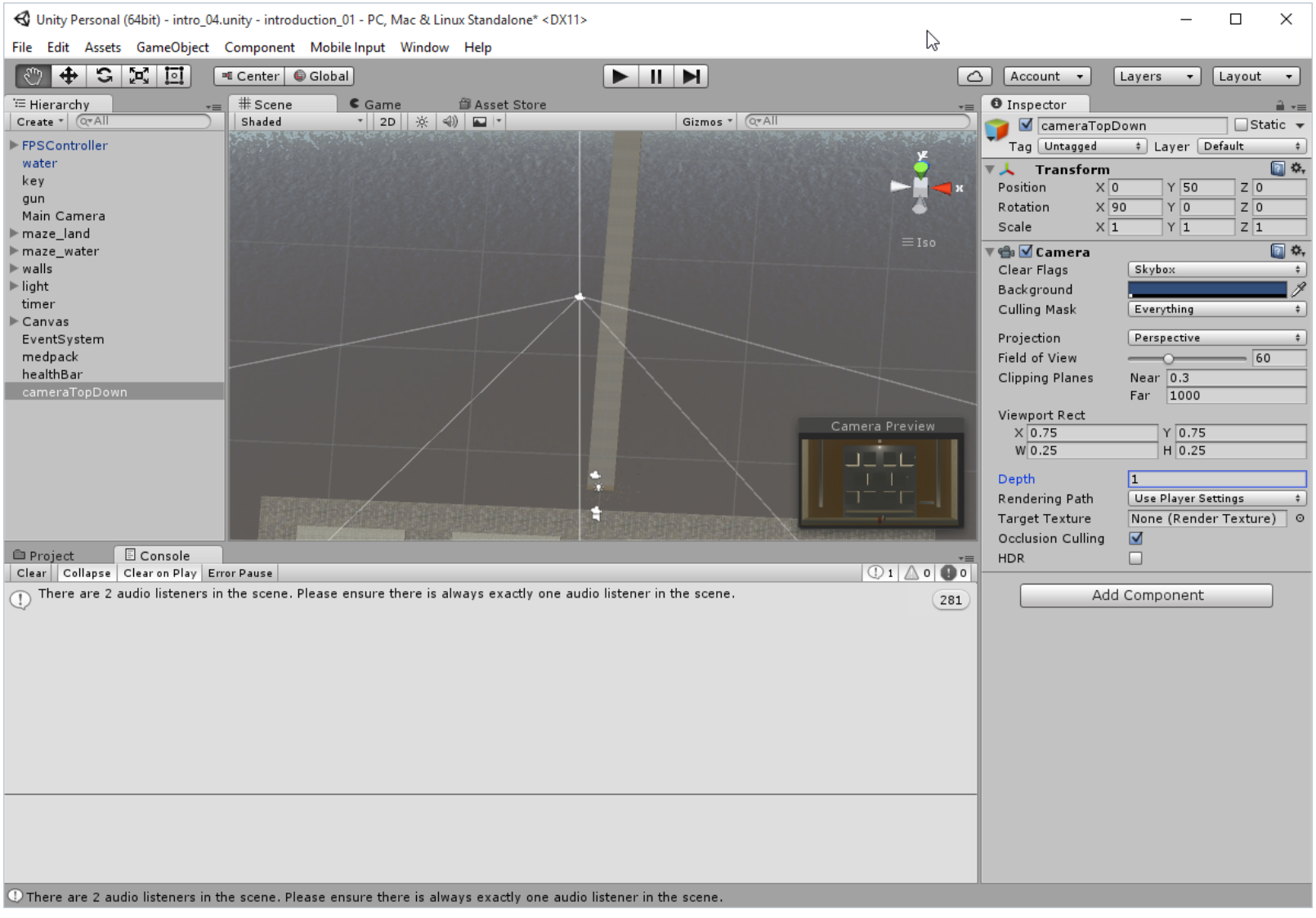
Create a mini-map of the level

Make top view as a part of the user interface

1. Click once on the camera labeled `cameraTopDown`.
2. Look at the **Inspector** window and click on the arrow to the left of the camera component to show its properties.
3. Change the attribute **Normalized View Port Rect**, as follows: `x=0.75`, `y=0.75`, `w=0.25`, and `h=0.25`.
4. Change the attribute **Depth** to `1`.
5. Delete the components **Audio Listener**, **GUILayer**, and **Flare Layer** (right-click on the component and select **Remove Component** from the contextual menu).
6. Play the scene. Test different settings for **Normalized View Port Rect**.

Create a mini-map of the level

Make top view as a part of the user interface



Working with layers

1. Click on the object **cameraTopDown** from the **Hierarchy** window.
2. In the **Inspector** window, click on the drop-down menu to the right of the **Layer**.
3. Select the option **Add Layer** from the drop-down menu. This should open a **Tags & Layers** tab, where in **Layers** section the series of built-in layers (for example, **Builtin Layer 0**) as well as user layers (for example, **User Layer 8**) would be listed.
4. Modify the first user layer by clicking on to the right of the label `User Layer 8`, type `topView` and press **Enter**.
5. Select the object **cameraTopDown** in **Hierarchy**.
6. In the **Inspector** window, within the component camera, modify the attribute **Culling Mask**, so that only the layer labeled `topView` is selected.

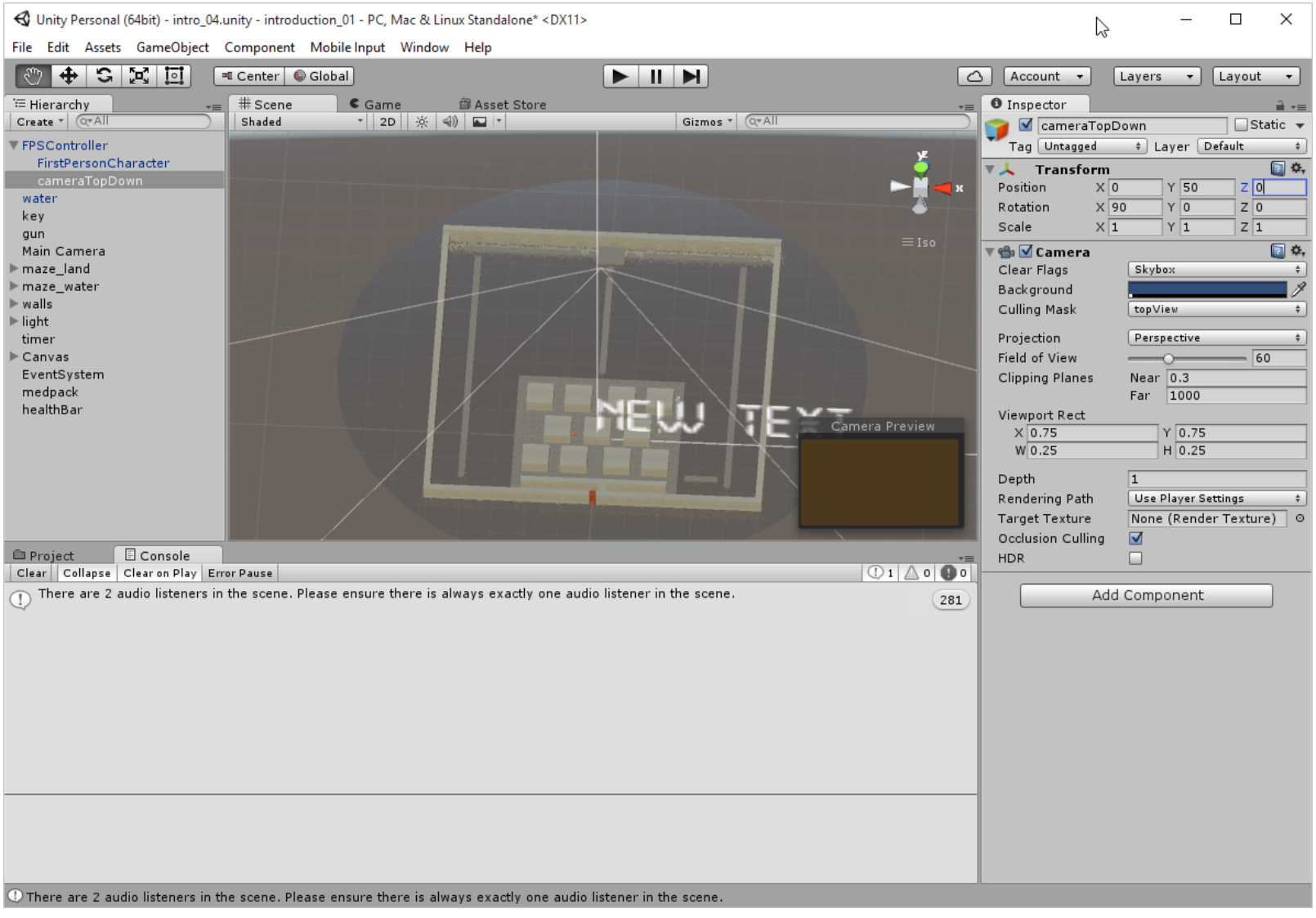
Working with layers

Next, we will make sure that this top-view camera is always above the player

1. Drag-and-drop the camera **cameraTopDown** on the **FPSController**.
2. Change its position to `(x=0, y=30, z=0)`.

Working with layers

Next, we will make sure that this top-view camera is always above the player



Working with layers

Display a simplified representation of the main character

1. Create a new **sphere**.
2. Change its scale to `(x=2, y=2, z=2)`.
3. Rename this object `dot_fpc`.
4. Locate the texture labeled `Green` by selecting **Assets** | **introduction_04** and apply this texture to the sphere.
5. Drag sphere object (`dot_fpc`) on the **FPSController** as we did it previously with the camera **cameraTopDown**.
6. Change its position to `(x=0, y=0, z=0)`.
7. This will include the sphere as a child of the first-person controller. Thanks to this, any transformation applied to the first-person controller will be also applied to the sphere. As a result, the sphere will move along with the character.

Working with layers

Display a simplified representation of the main character

1. Click on the object `dot_fpc` in the **Hierarchy** window to select it.
2. In the **Inspector** window, click on the drop-down menu to the right of the **Layer** label.
3. Select the option **topView** from the list.
4. In the **Inspector** window, right-click on the component **Sphere Collider** for this object, and select the option **Remove Component** from the contextual menu. This will remove the collider from the sphere. We do this because we don't want the player to collide with the sphere.

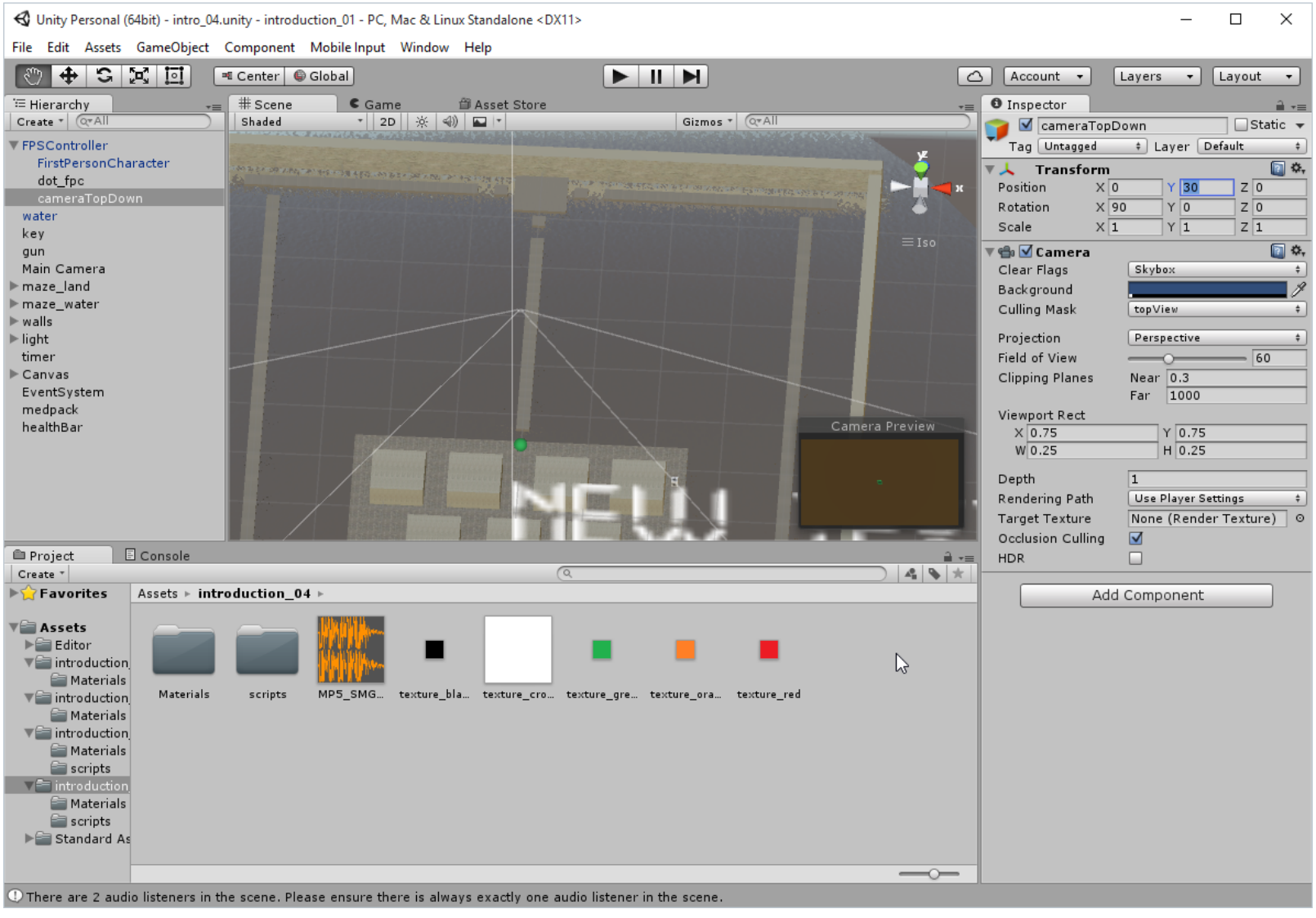
Working with layers

Display a simplified representation of the main character regardless of the light around it

1. Select the object `dot_fpc`.
2. In the **Inspector** window, open the component **Mesh Renderer**, and change its **Shader** property to **Standard**, **Main Maps** | **Metalic** equal to 0 and **Main Maps** | **Smoothness** equal to 0.
3. Leave the other options as default.

Working with layers

Display a simplified representation of the main character regardless of the light around it



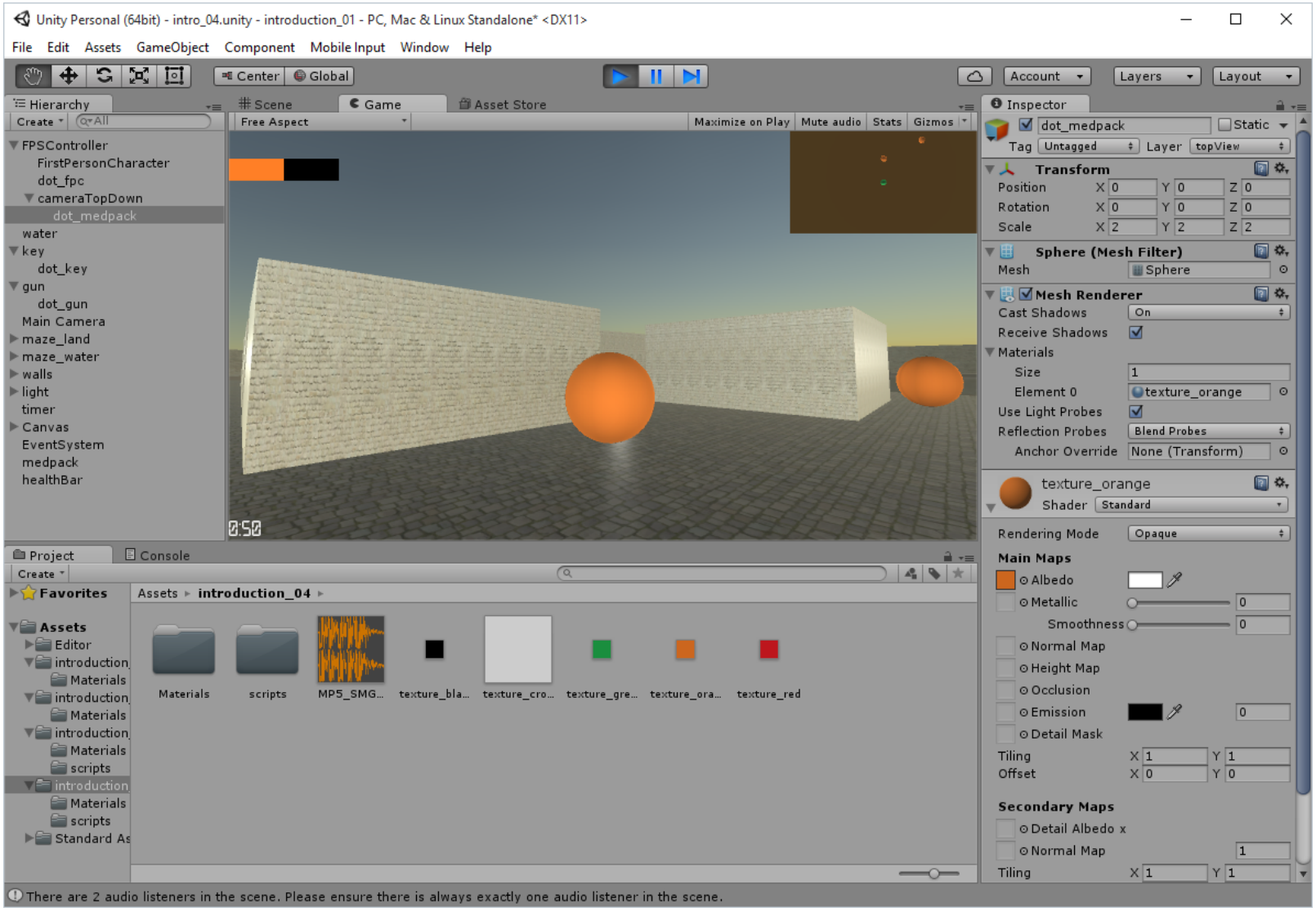
Working with layers

Other objects: create the dots

1. Create a new sphere, change its scale to `(x=2, y=2, z=2)`, and rename it `dot_medpack`.
2. Locate the texture labeled `Orange` by selecting **Assets** | **introduction_04** and apply this texture to the sphere.
3. Change the shader property as we did it before.
4. Remove the **SphereCollider** component from this object.
5. Change the position of this object to `(x=0, y=0, z=0)`.
6. Change its **Layer** property to **topView**.
7. Drag-and-drop the object (**dot_medpack**) on the object labeled `medpack`.
8. Repeat the previous steps to create two other spheres named `dot_key` and `dot_gun` for both the objects labeled `key` and `gun`.

Working with layers

Other objects: create the dots

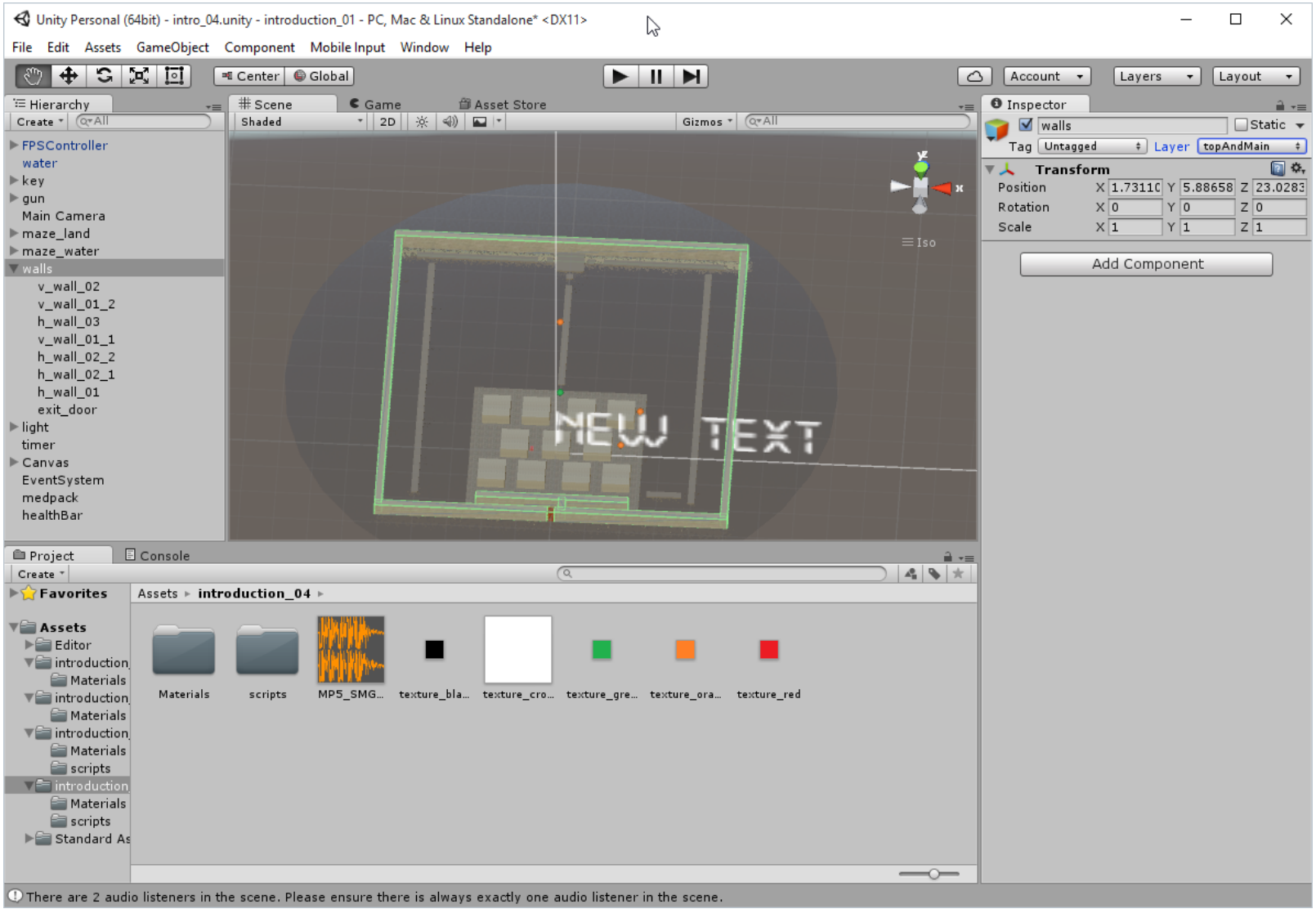


1. Select one of the walls in the scene.
2. In the **Inspector** window, click on the drop-down menu to the right of the label **Layer**.
3. From the drop-down menu, select the option **Add Layer**.
4. Create a layer, to the right of the label **User Layer 9**, that we will label `topAndMain`.

1. Select all the walls in the level (or select them one-by-one if needed) as well as all objects labeled `block`.
2. In the **Inspector** window, click on the drop-down menu to the right of the label **Layer**.
3. From the drop-down menu, select **topAndMain**.
4. We may also apply this layer to other objects such as the rocks and platforms in the water area.

Working with layers

Display part of the environment



1. Click on **cameraTopDown**.
2. In the **Inspector** window, change the **Culling Mask** attribute of its **Camera** component so that it includes both the layers **topView** and **topAndMain**.
3. Select the **FPSController | First Person Character** object.
4. Change the **Culling Mask** attribute of its **Camera** component so that it displays everything but not the **topView** layer.

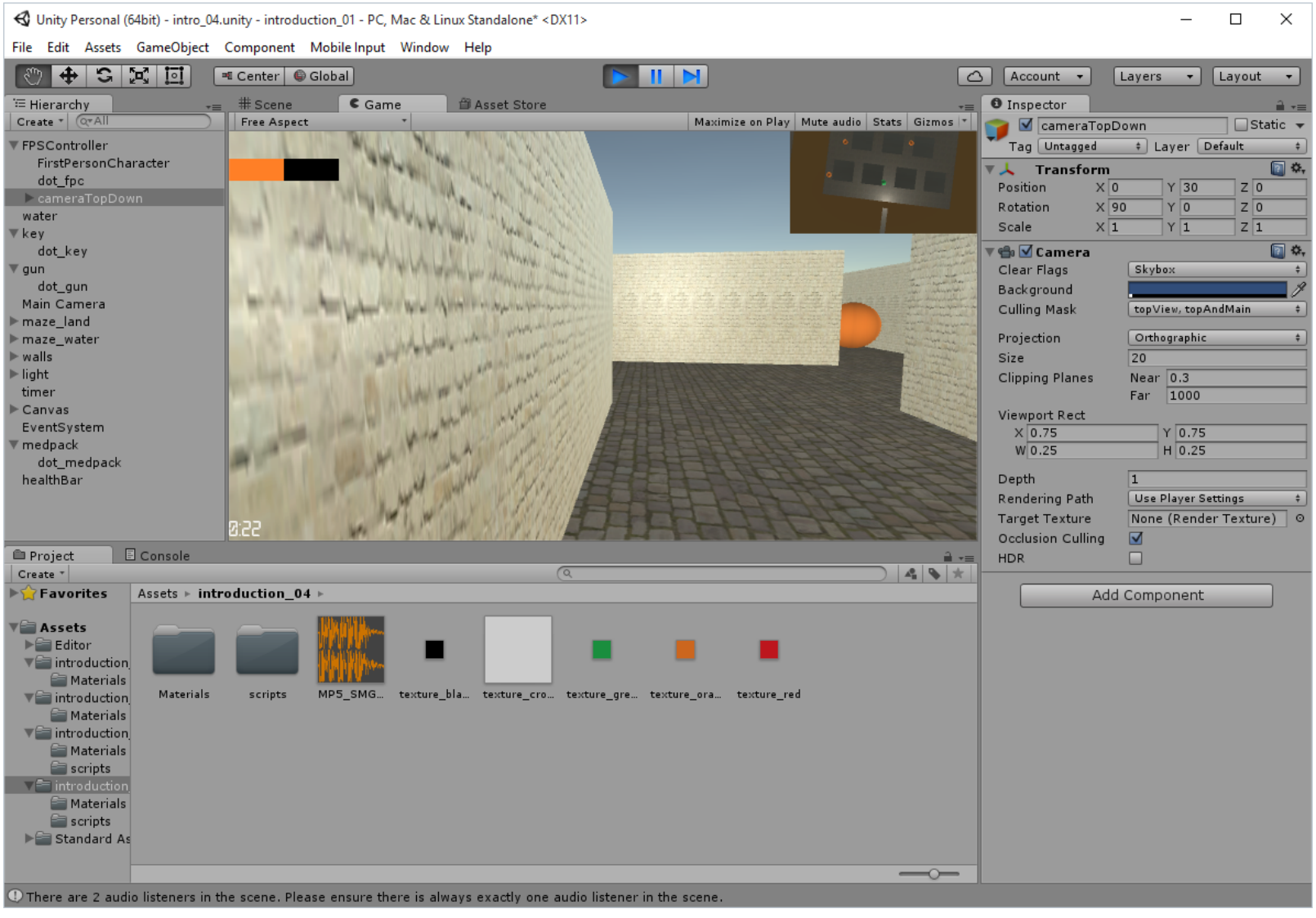
Working with layers

Change settings for top down camera (optional)

1. Click on **cameraTopDown**.
2. In the **Inspector** window, change the **Projection** attribute to **Orthographics**.
3. Change the **Size** attribute to 20.

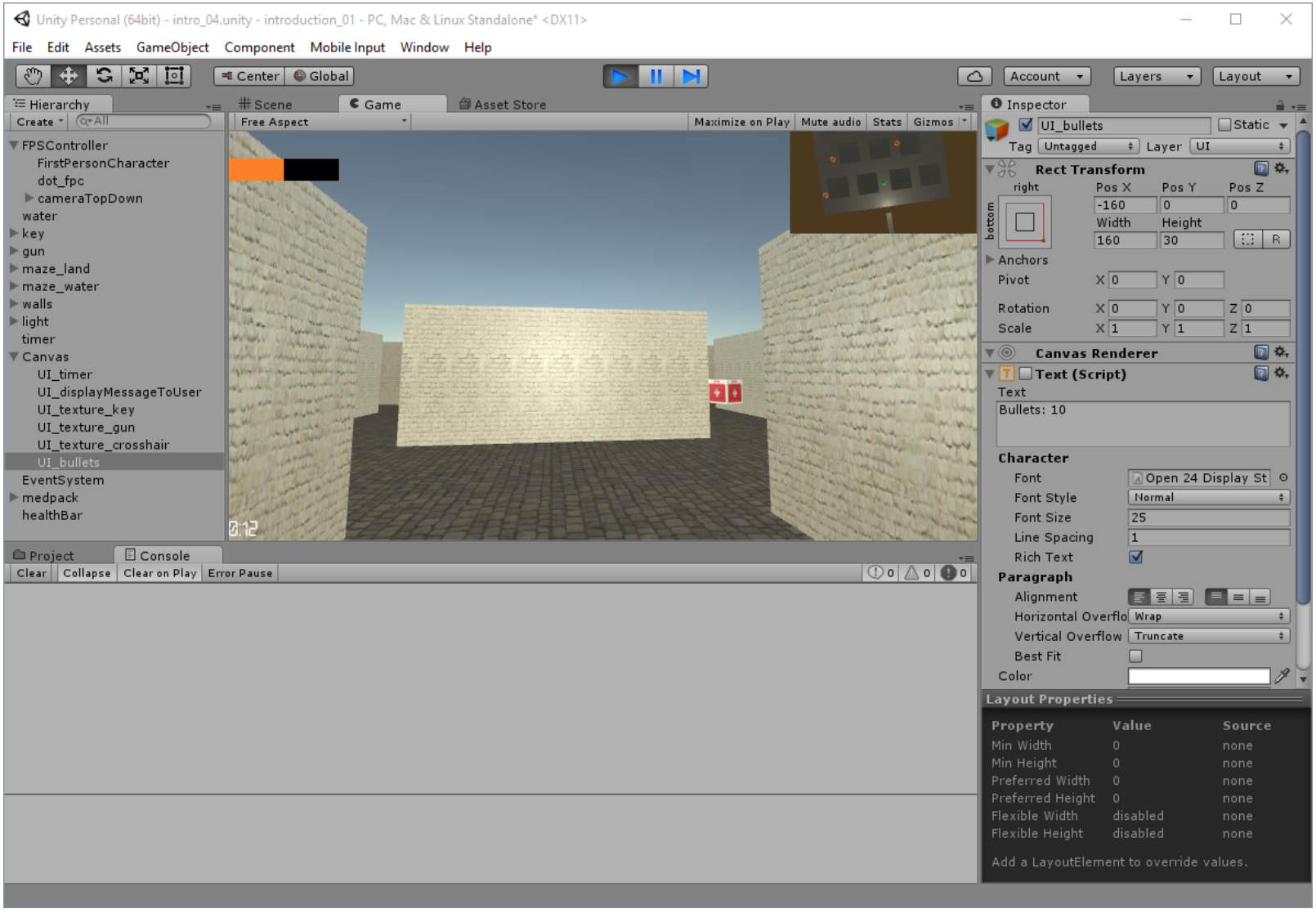
Working with layers

Change settings for top down camera (optional)



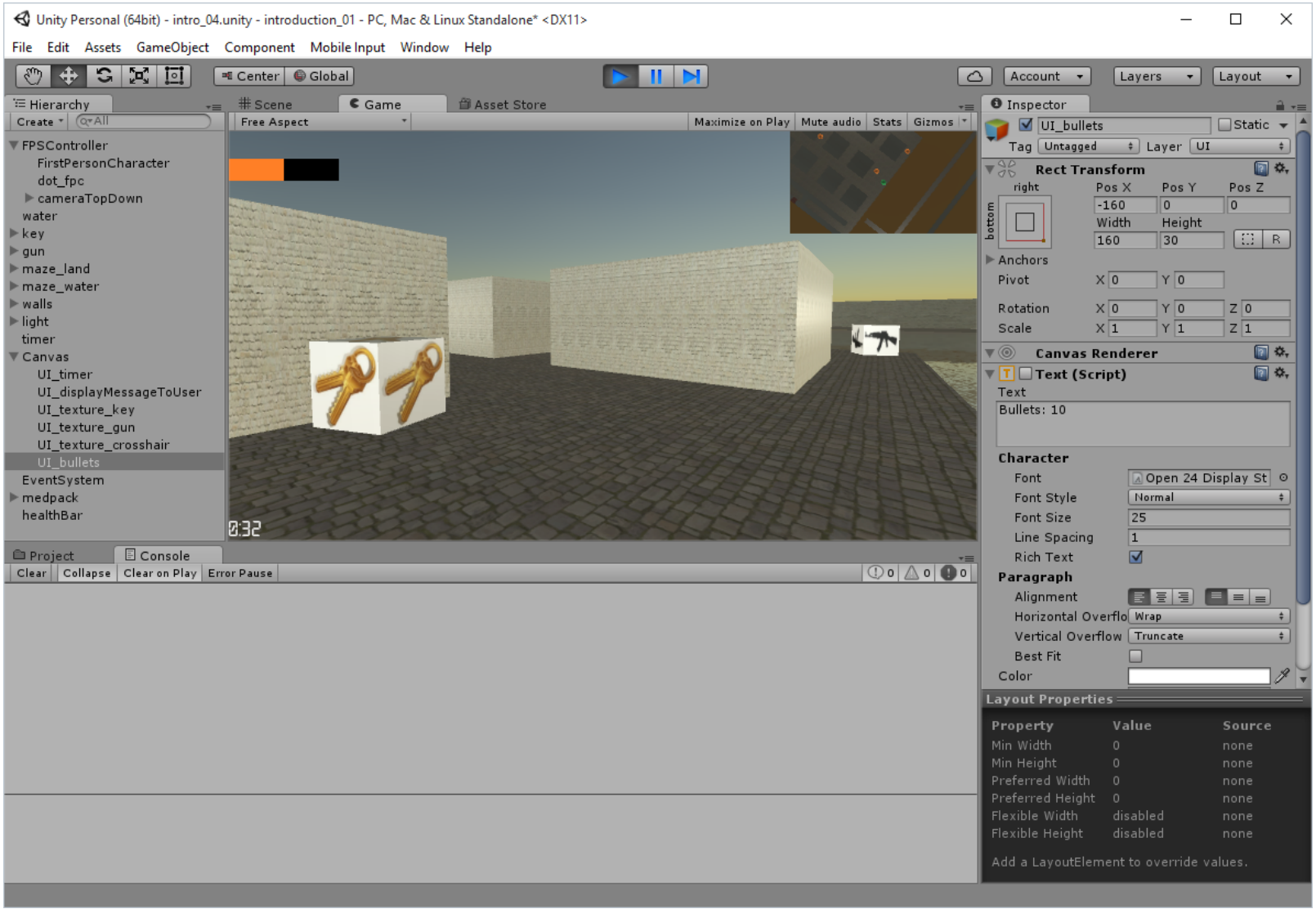
Working with layers

Change settings for top down camera (optional)



Working with layers

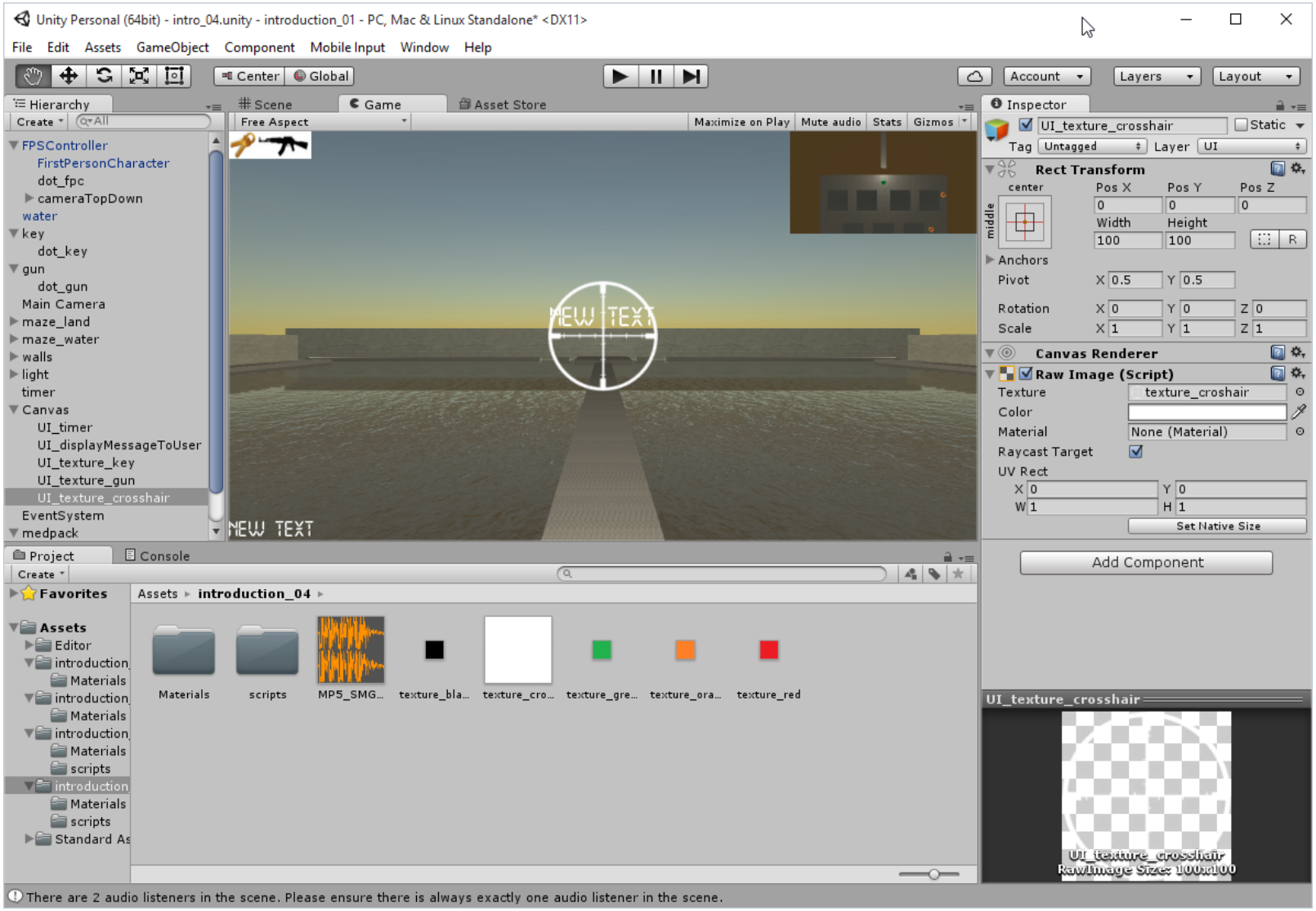
Change settings for top down camera (optional)



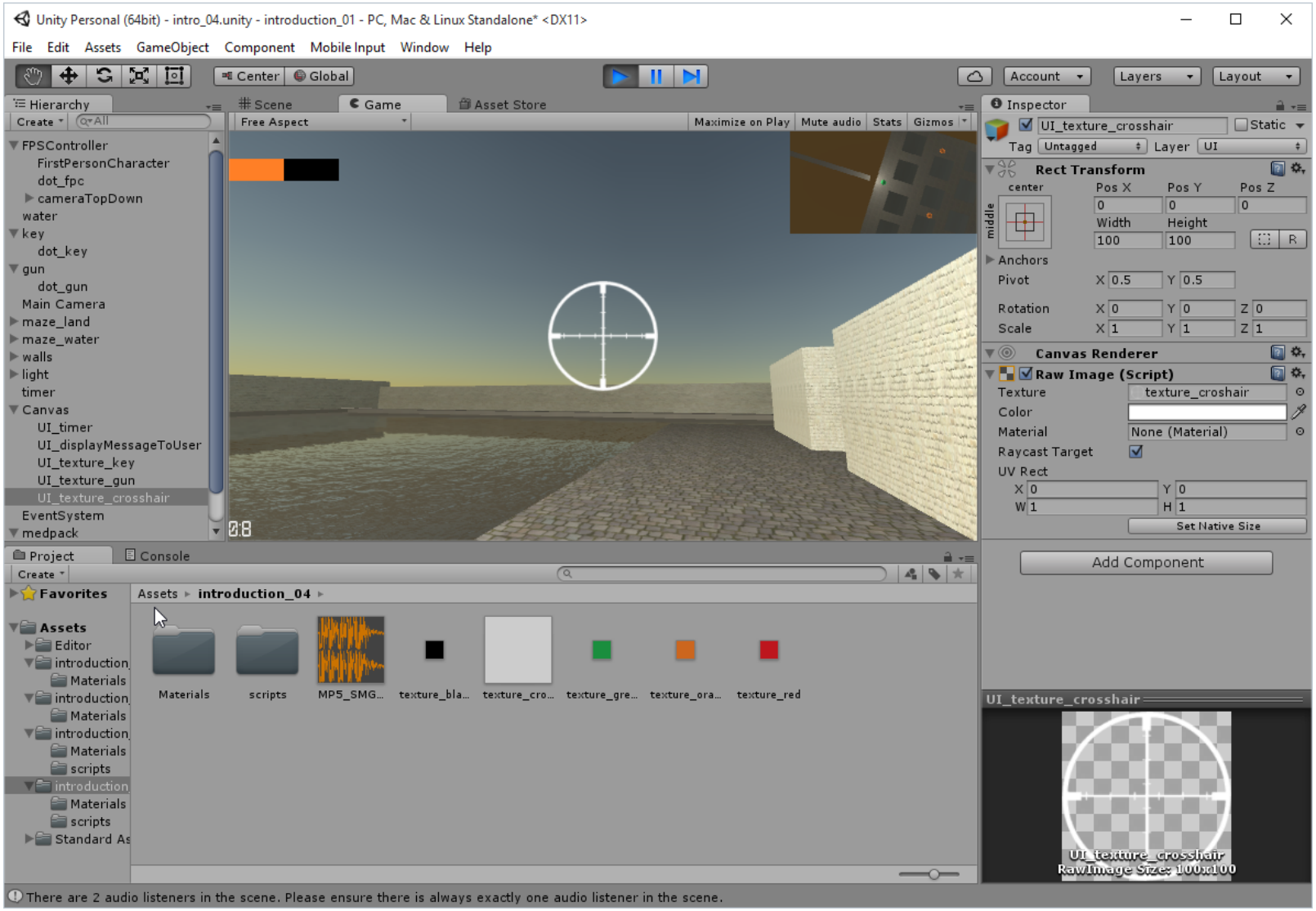
Create a gun

1. Create a new **UI.RawImage**, and rename it `UI_texture_crosshair`.
2. Change its position so that it is displayed in the middle of the screen.
3. Drag-and-drop the texture labeled `texture_crosshair` by selecting **Assets | introduction_04** to the **Raw Image | Texture** component of this object.
4. This should display the crosshair in the middle of the screen in the game view.

Create a gun


Unity Personal (64bit) - intro_04.unity - introduction_01 - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Mobile Input Window Help
Center Global
Account Layers Layout
Hierarchy
Create All
FPSController
FirstPersonCharacter
dot_fpc
cameraTopDown
water
key
dot_key
gun
dot_gun
Main Camera
maze_land
maze_water
walls
light
timer
Canvas
UI_timer
UI_displayMessageToUser
UI_texture_key
UI_texture_gun
UI_texture_crosshair
EventSystem
medpack
Scene
Game
Asset Store
Free Aspect
Maximize on Play
Mute audio
Stats
Gizmos
NEW TEXT
NEW TEXT
Inspector
UI_texture_crosshair
Static
Tag Untagged
Layer UI
Rect Transform
center
middle
Pos X
Pos Y
Pos Z
Width
Height
Anchors
Pivot
Rotation
Scale
Canvas Renderer
Raw Image (Script)
Texture
texture_croshair
Color
Material
None (Material)
Raycast Target
UV Rect
Set Native Size
Add Component
Project
Console
Create
Favorites
Assets
Editor
introduction_
Materials
scripts
Standard As
Assets ▸ introduction_04 ▸
Materials
scripts
MP5_SMG...
texture_bla...
texture_cro...
texture_gre...
texture_ora...
texture_red
UI_texture_crosshair
UI_texture_crosshair
RawImage Size: 100x100
There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.

Create a gun


Unity Personal (64bit) - intro_04.unity - introduction_01 - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Mobile Input Window Help
Center Global
Account Layers Layout
Hierarchy
Create All
FPSController
FirstPersonCharacter
dot_fpc
cameraTopDown
water
key
dot_key
gun
dot_gun
Main Camera
maze_land
maze_water
walls
light
timer
Canvas
UI_timer
UI_displayMessageToUser
UI_texture_key
UI_texture_gun
UI_texture_crosshair
EventSystem
medpack
Scene
Game
Asset Store
Free Aspect
Maximize on Play
Mute audio
Stats
Gizmos
Inspector
UI_texture_crosshair
Static
Tag Untagged
Layer UI
Rect Transform
center
middle
Pos X
Pos Y
Pos Z
Width
Height
Anchors
Pivot
Rotation
Scale
Canvas Renderer
Raw Image (Script)
Texture
texture_croshair
Color
Material
None (Material)
Raycast Target
UV Rect
Set Native Size
Add Component
Project
Console
Create
Favorites
Assets
Editor
introduction_
Materials
scripts
Assets ▸ introduction_04 ▸
Materials
scripts
MP5_SMG...
texture_bla...
texture_cro...
texture_gre...
texture_ora...
texture_red
UI_texture_crosshair
UI_texture_crosshair
RawImage Size: 100x100
There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.

Create a gun

Use ray casting to aim and fire a bullet

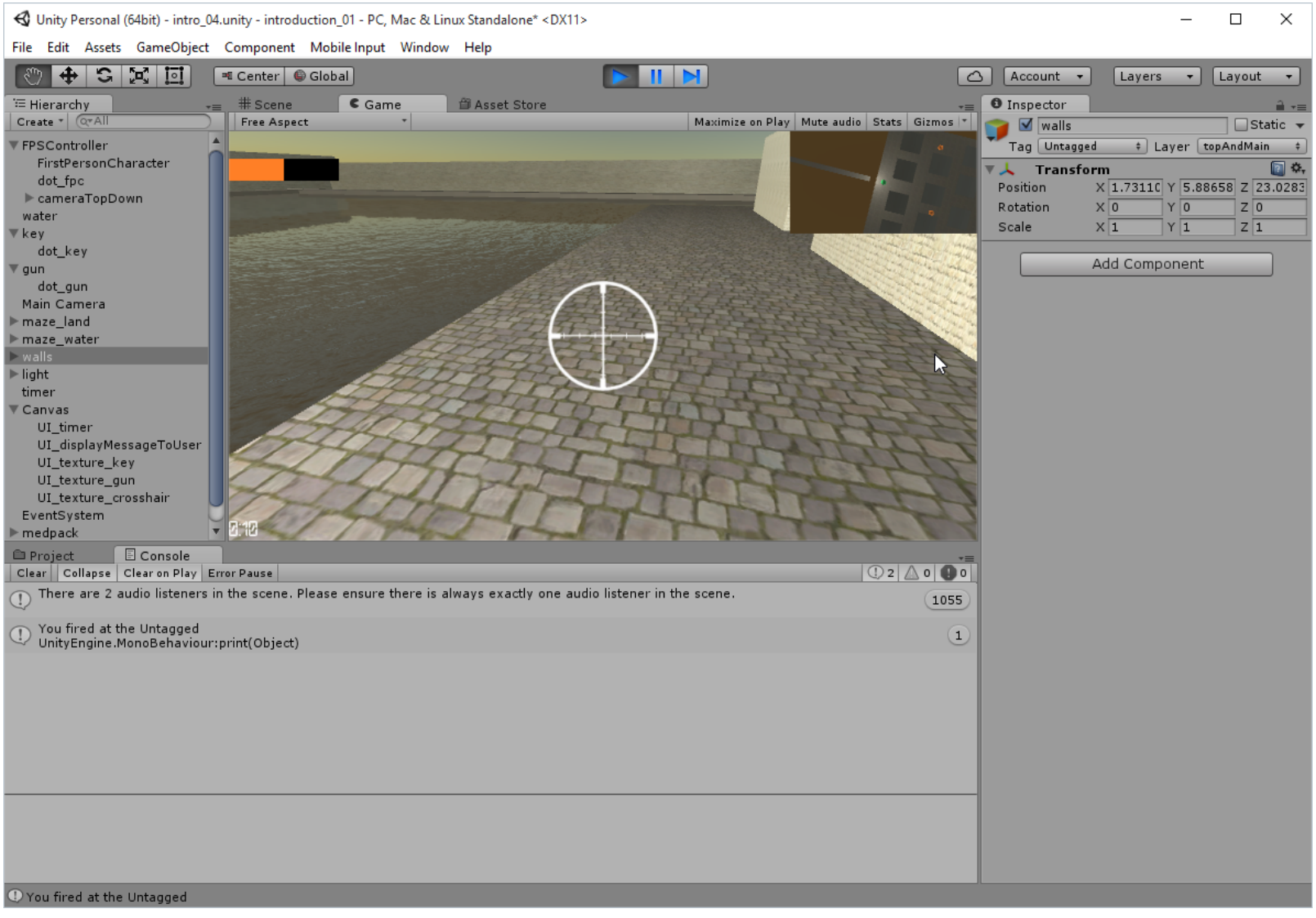
1. Create a new script by selecting **Assets** | **introduction_04** | **Scripts** and rename it `fire_gun`.
2. Modify the script as described in the following code

```
function Update ()
{
  if (Input.GetButtonDown("Fire1"))
  {
    var hit : RaycastHit;
    var ray = Camera.main
              .ScreenPointToRay (Vector3(Screen.width/2,
                                         Screen.height/2));
    if(Physics.Raycast (ray, hit, 100))
    {
      print("You fired at the "+hit.collider.gameObject.tag);
    }
  }
}
```

3. Attach the script to **UI_texture_crosshair** object.

Create a gun

Use ray casting to aim and fire a bullet



Create a gun

Hide the mouse cursor

1. Open the script `fire_gun`.
2. Add the following line within the function `Start`

   ```
   Cursor.visible = false;
   ```
3. Play the scene and check that the mouse cursor is hidden after the first shot.

Create a gun

Display and update the number of ammunitions left

1. Open the script `fire_gun`.
2. Add the following line at the start of the script

```
public var nbBullets: int;
```

3. Add the following code inside the `Start` function

```
nbBullets = 0;
```

4. Modify the function `Update` as highlighted in the following code

```
... see next slide ...
```

Create a gun

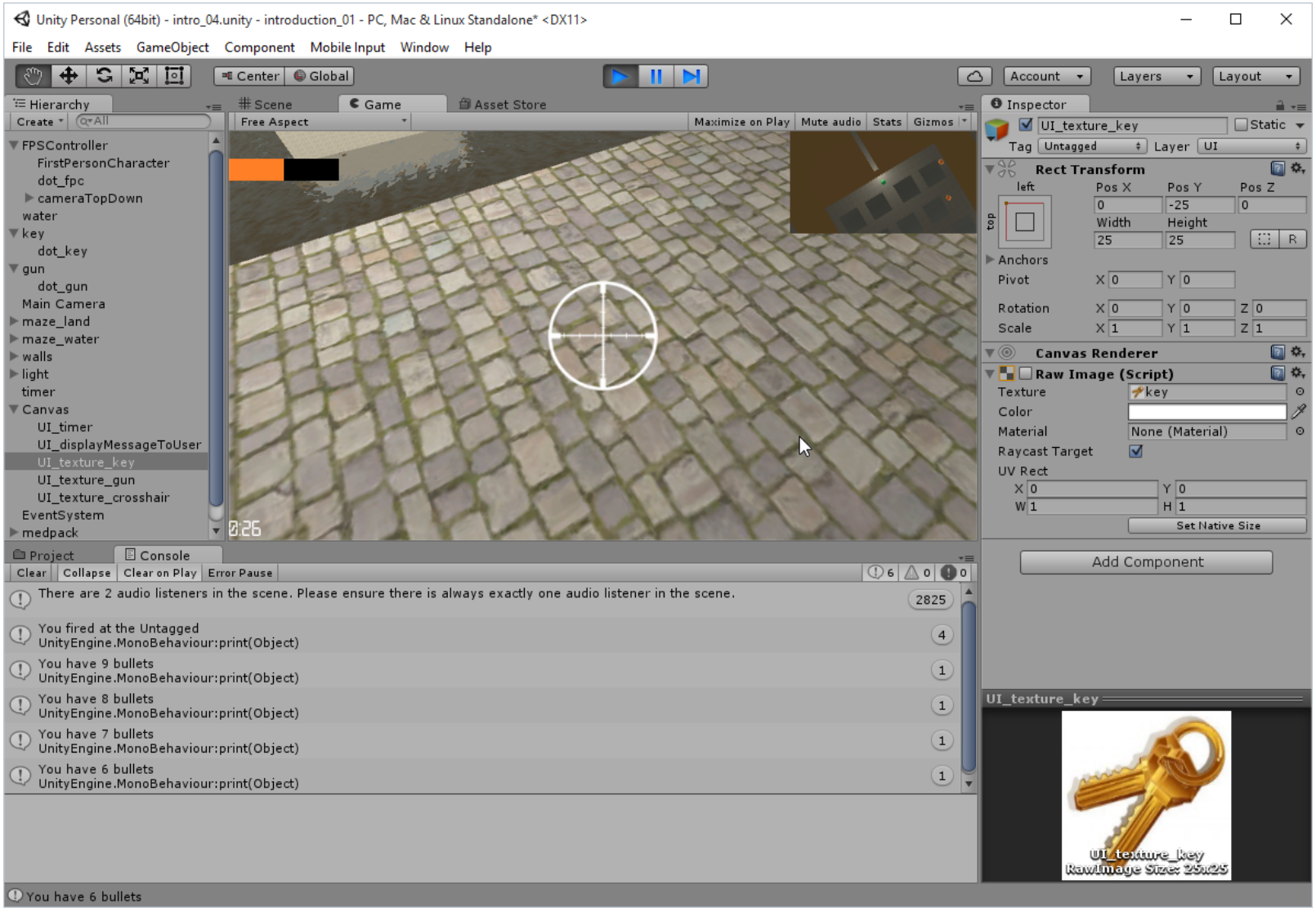
Display and update the number of ammunitions left

1. Modify the function Update as highlighted in the following code

```
function Update ()
{
  if (Input.GetButtonUp("Fire1"))
  {
    if (nbBullets > 0)
    {
      var hit : RaycastHit;
      var ray = Camera.main
                .ScreenPointToRay (Vector3(Screen.width/2,
                                           Screen.height/2));
      if(Physics.Raycast (ray, hit, 100))
      {
        print("You fired at the "+hit.collider.gameObject.tag);
      }
      nbBullets--;
      print("You have " + nbBullets + " bullets");
    }
  }
}
```

Create a gun

Display and update the number of ammunitions left



Create a gun

Display and update the number of ammunitions left

1. Create a new **UI.Text** object, rename it **UI_bullets** and change its position to right bottom corner.
2. Open the script `fire_gun`.
3. Add the following line to the start of the script

```
private var ui_bullets: GameObject;
```

4. Add the following line to the function `Start`

```
ui_bullets = GameObject.Find("UI_bullets").
             GetComponent(UI.Text);
```

5. Add the following line of code to the function `Update`

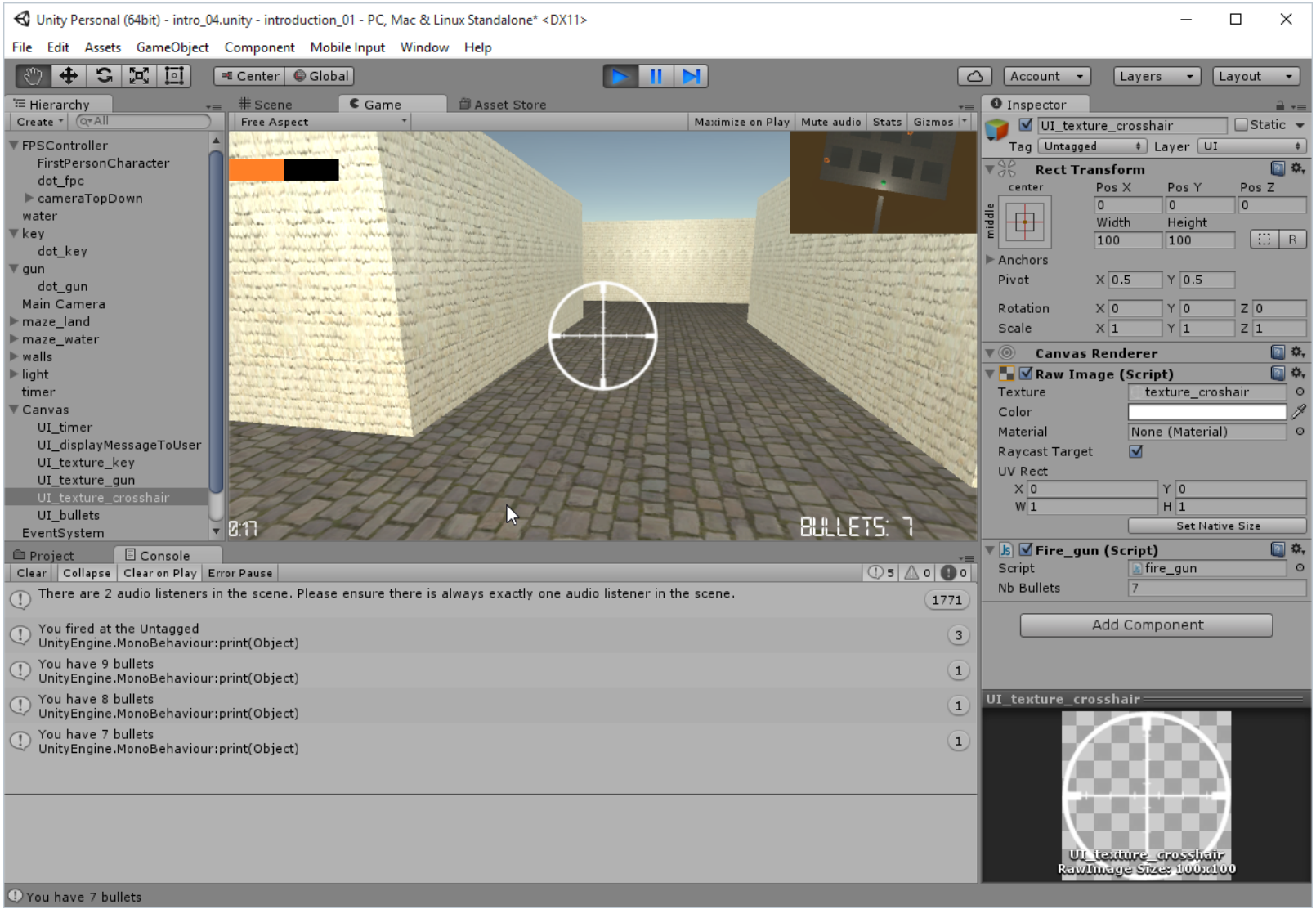
```
ui_bullets.text= getTextForUIBullet(GameObject.
Find("UI_texture_crosshair").
     GetComponent(fire_gun).
     nbBullets);
```

6. Add the function

```
function getTextForUIBullet(nbBullets: int){
return "Bullets: " + nbBullets;
}
```

Create a gun

Display and update the number of ammunitions left



1. Modify the `Start` function in `fire_gun` script adding

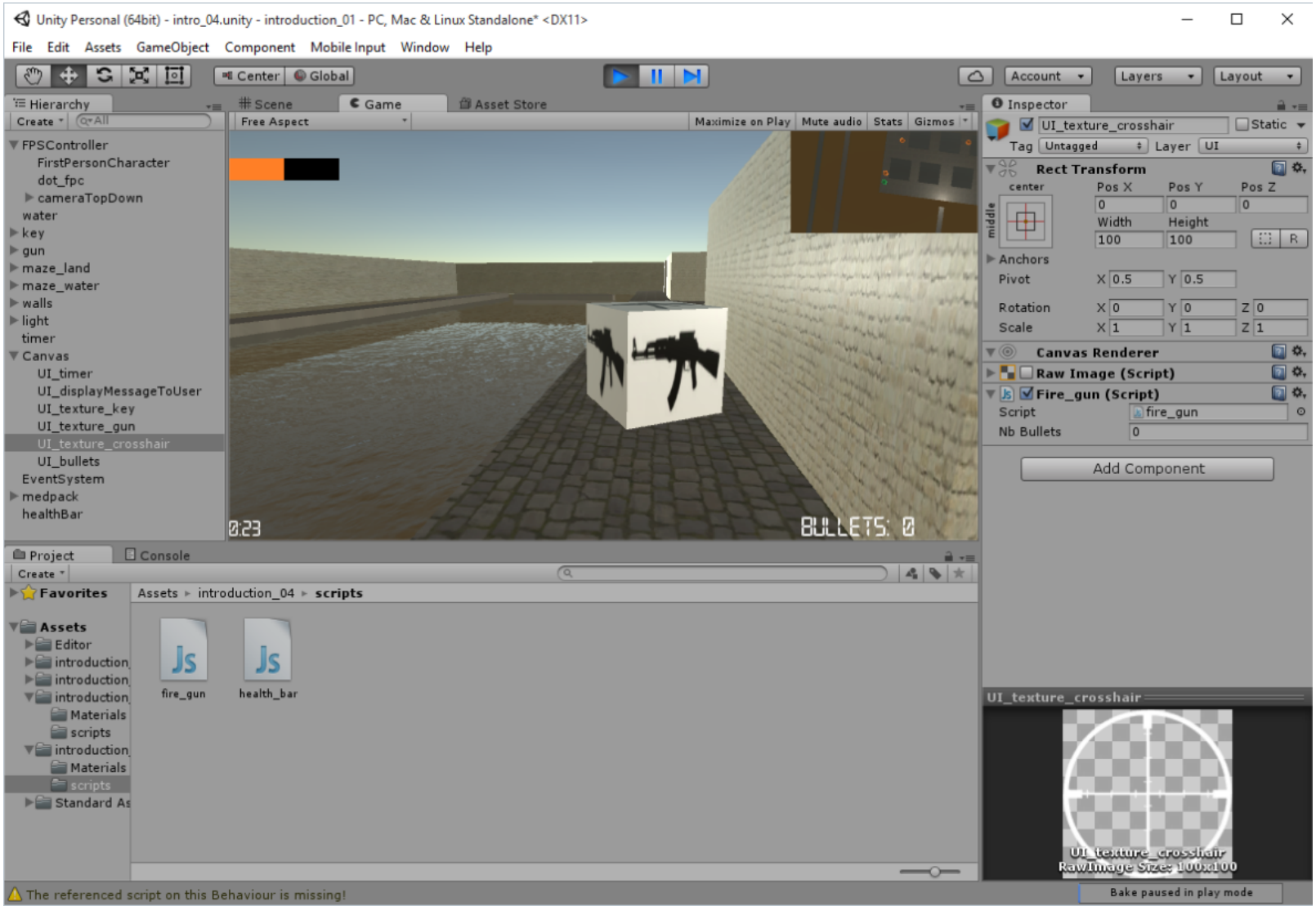
```
function Start ()
{
    ...
GameObject.Find("UI_texture_crosshair").
   GetComponent(UI.RawImage).
   enabled=false;
}
```

1. Open the script `collision_detection`.
2. Add the highlighted code to the script in the section that detects whether the gun has been collected

```
if (c.gameObject.tag == "gun")
{
  hasGun = true;
  changeGUITexture(true, "gun");
  GameObject.Find("GUITexture_crosshair")
    .guiTexture
    .enabled = true;
  GetComponent(shootBullet).nbBullets = 40;
  hasGun = true;
  changeUITexture("gun", hasGun);
  GameObject.Find("UI_texture_crosshair").
       GetComponent(UI.RawImage).
       enabled=true;
  GameObject.Find("UI_texture_crosshair").
        GetComponent(fire_gun).
        nbBullets = 50;
```

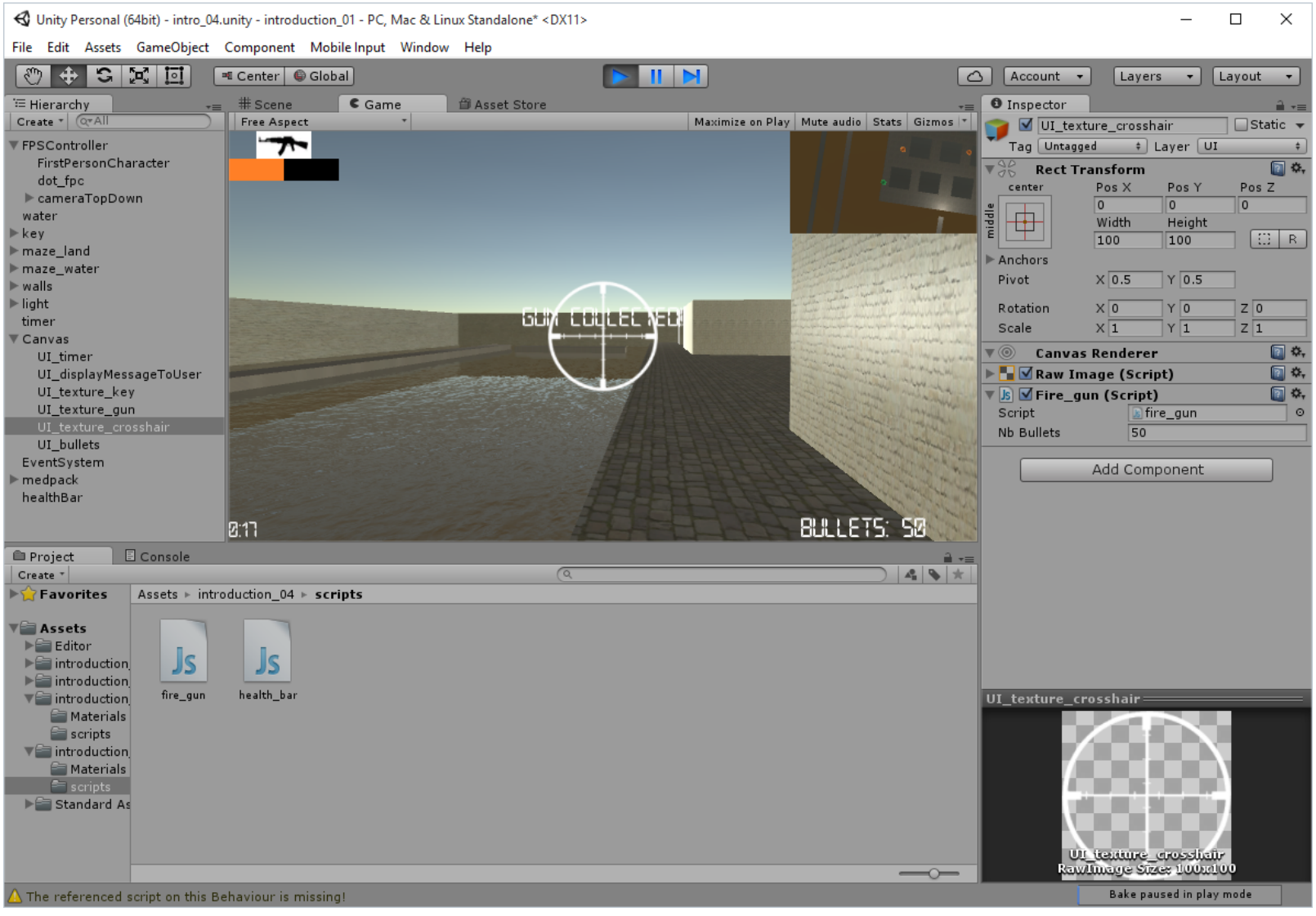
Create a gun

Fine tuning



Create a gun

Fine tuning



Create a gun

Add sound whenever the player fires a shot

1. Open the script `fire_gun`.
2. Add the following line at the start of the script

```
@script RequireComponent (AudioSource)
#pragma strict
public var fire_sound: AudioClip;
```

3. Add the following highlighted code

```
if (nbBullets >= 1)
{
  var audio: AudioSource = GetComponent.<AudioSource>();
  audio.clip = fire_sound;
  audio.Play();
```

4. To be continued...

Create a gun

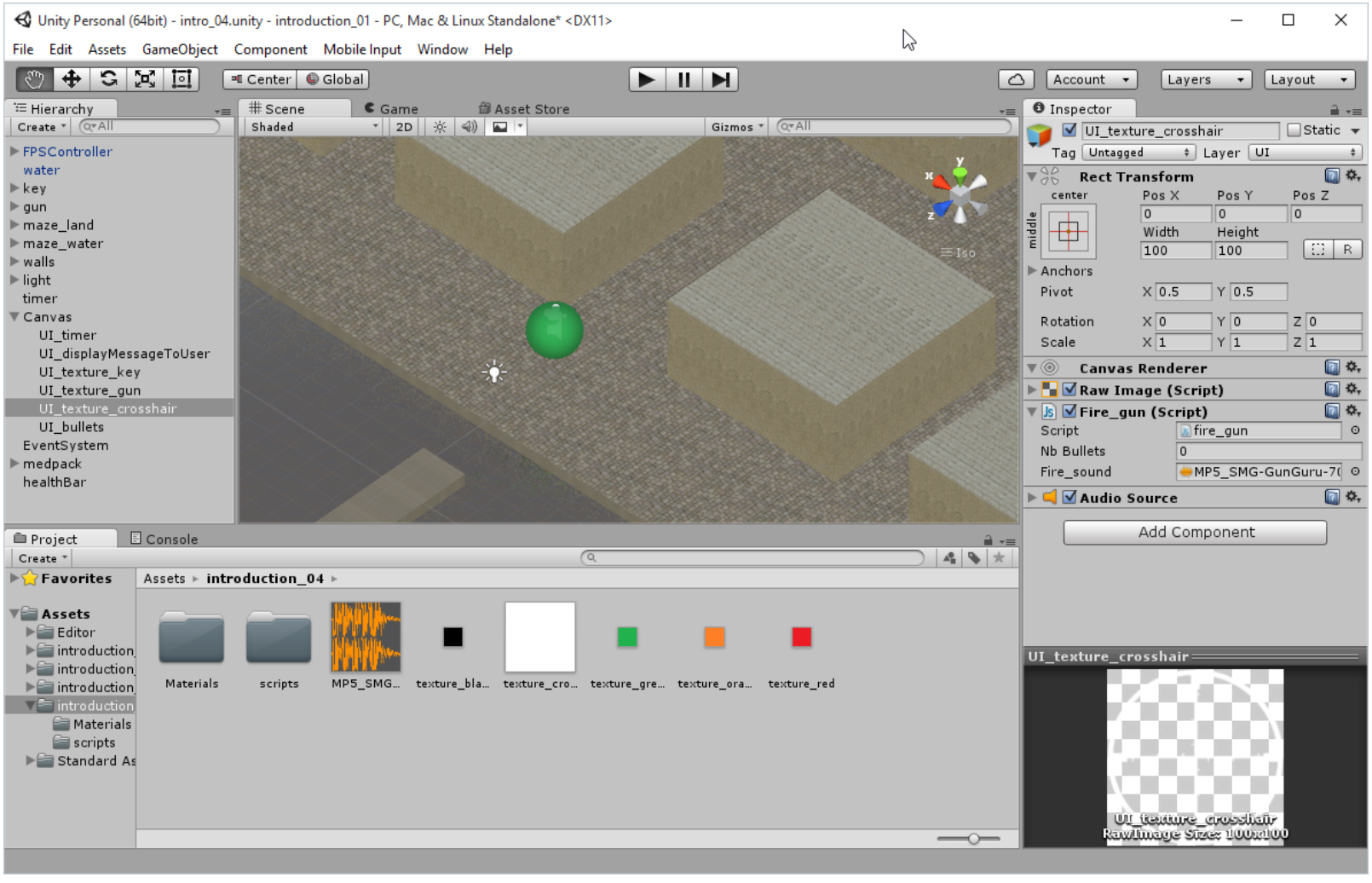
Add sound whenever the player fires a shot

. . . continued

1. Select the object **UI_texture_corsshair** in the **Hierarchy** window.
2. Locate the component **fire_gun** for this object in the **Inspector** window.
3. Drag-and-drop the sound `gunshot` to the variable `fire_sound` within the component **fire_gun**.
4. Press **Add Componen** button and select **Audio** | **Audio source**.
5. Test the scene and check that the sound is played when we fire the gun.

Create a gun

Add sound whenever the player fires a shot



Create a gun

Prepare for a particle emitter

1. Include the following line at the start of the script `fire_gun`

```
public var flash: GameObject;
```

2. In the script `fire_gun`, add the highlighted lines

```
print ("You fired at the" + hit.collider.gameObject.tag):
var spark: GameObject = Instantiate(flash,
                                    hit.point,
                                    Quaternion.identity);
```

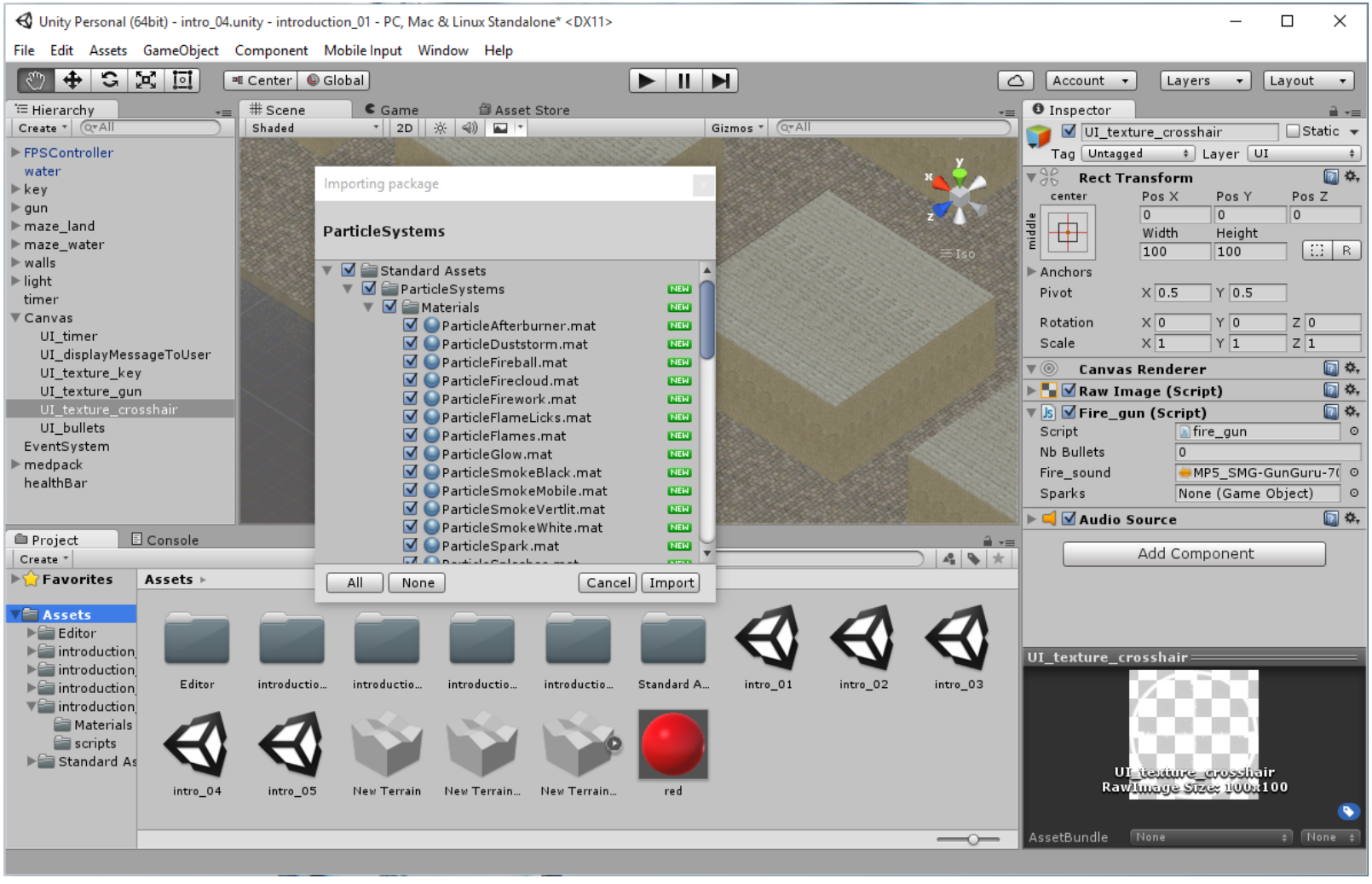
Create a gun

Find the prefab to simulate sparks

1. Select **Assets** | **Import Package** | **ParticleSystems**. This should show a window labeled **Importing** package. As per previous sections, it includes all built-in particles (including legacy particles) that can be used in Unity.
2. Click on **Import**.
3. This will create a new folder labeled `ParticleSystems` in **Assets** | **Standard Assets**.
4. If we select **Assets** | **Standard Assets** | **ParticleSystems** | **Prefabs**, we can find many particle prefabs.
5. Select one of them, for example the prefab `Explosion`.
6. Drag-and-drop the prefab that we have found previously to the variable called `flash` for the script `fire_gun`, which is a component of the **UI_texture_crosshair**.
7. Test the scene: fire shots at objects, and check whether flash appear at the point of impact.

Create a gun

Find the prefab to simulate sparks



Task

Task 04_01: game world task

Modify game world with some new elements like

- rotating walls,
- moving flors;
- moving hot block (collision with hot block kills the player).

Task

Task 04_02: gun task

Add an automatic feature to the gun, so that the player can fire the gun repeatedly by just holding the left mouse button down.

Summary

You should know

- switch between and display multiple camera views
- how to define and apply layers to filter content displayed by a camera,
- how to use special effects like sparks.