Introduction to Unity

Piotr Fulmański

Wydział Matematyki i Informatyki,
Uniwersytet Łódzki, Polska

22 października 2015

Table of contents

Introduction to game engines

Unity3D is a game engine – a type of software which makes possible for everyone to design and develop video games, **focusing essentially on the game mechanics, rather than the underlying layers necessary to build a game**.

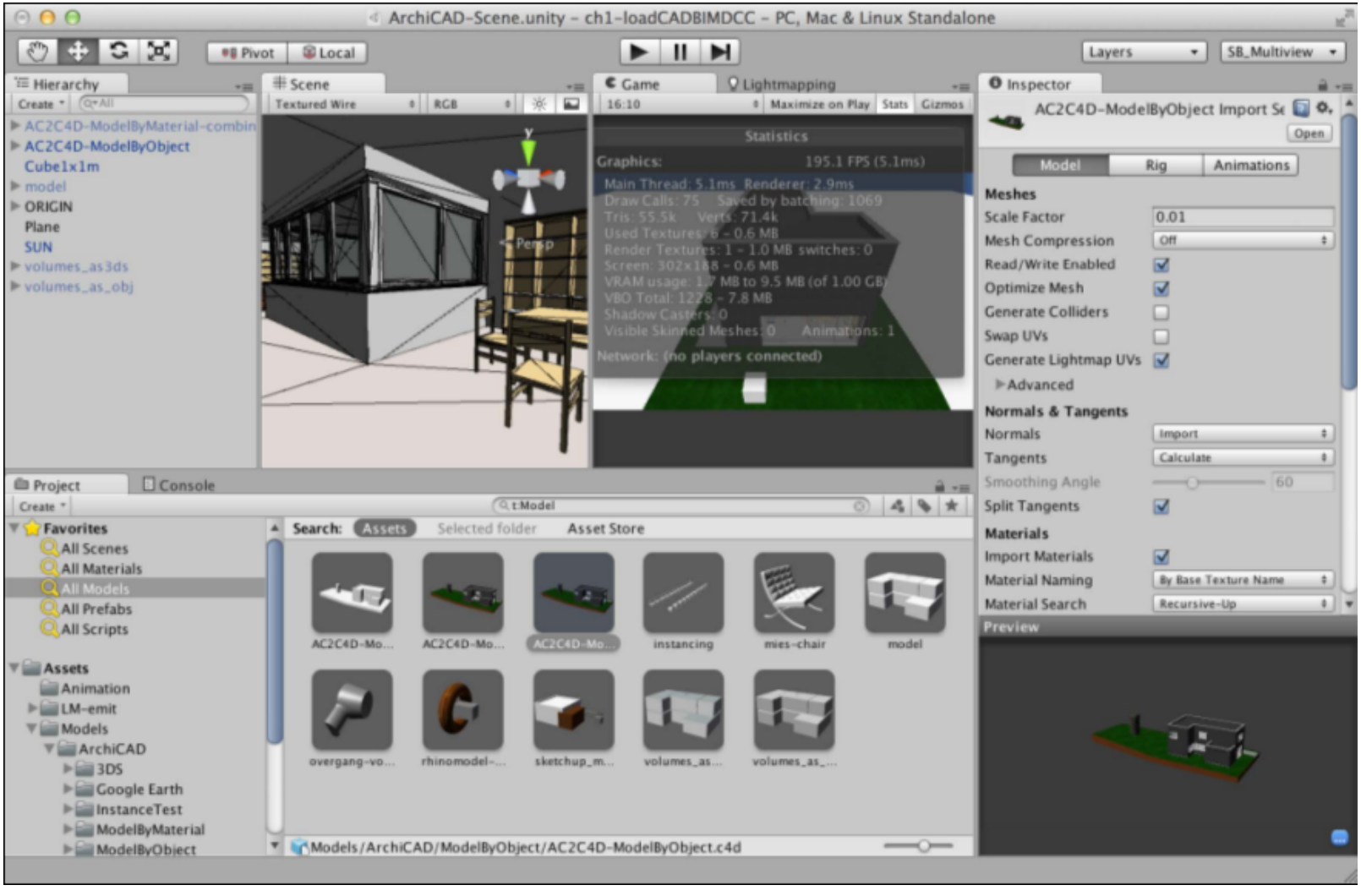
Introduction to game engines

While game engines were initially essential for the production of video games, they are now used for a wide range of applications, with purposes other than gaming. For example, game engines are now employed for

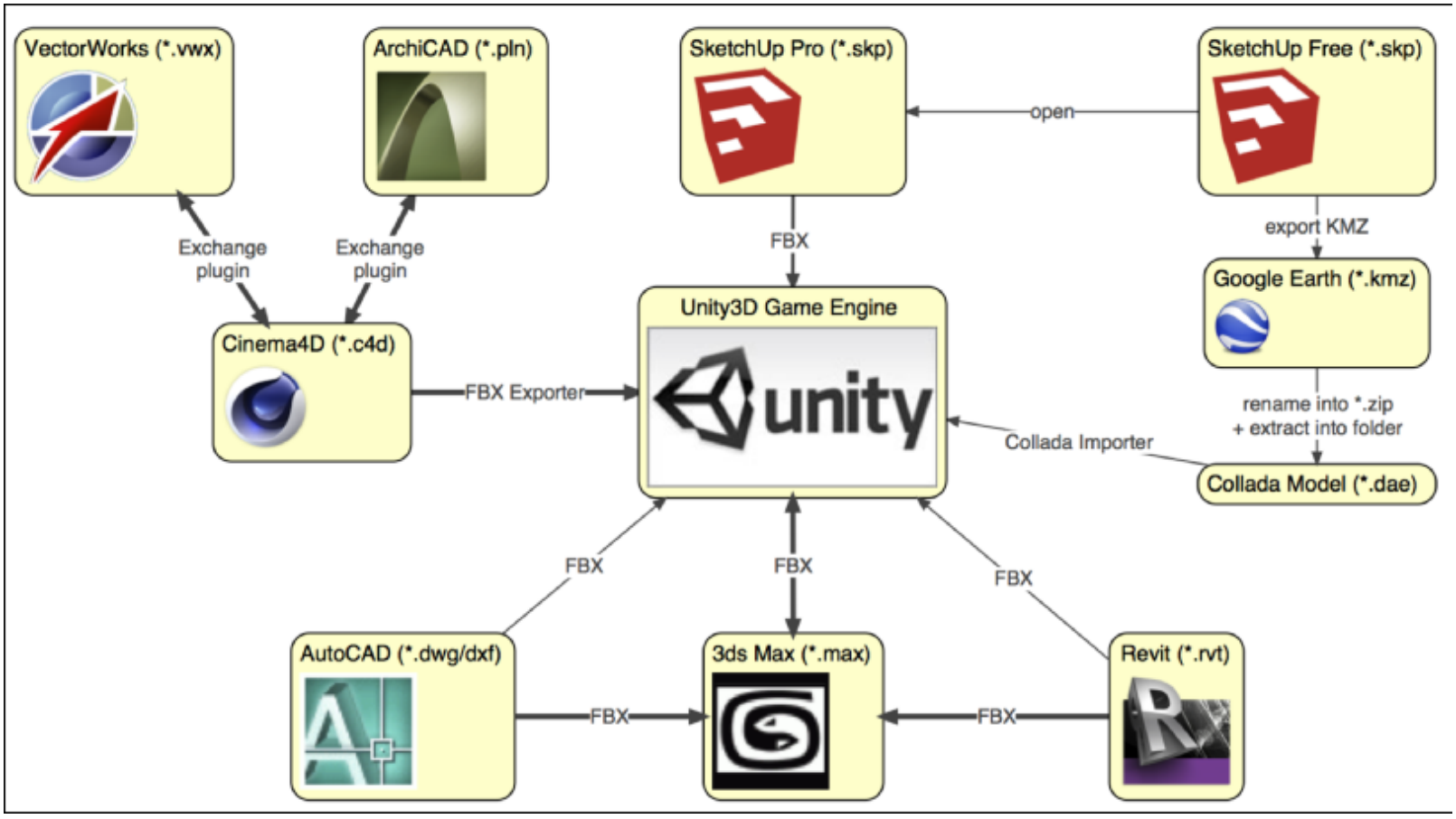
- simulation,
- teaching, and training,
- visualization

as they often make it possible to create and manage very realistic environments easily.

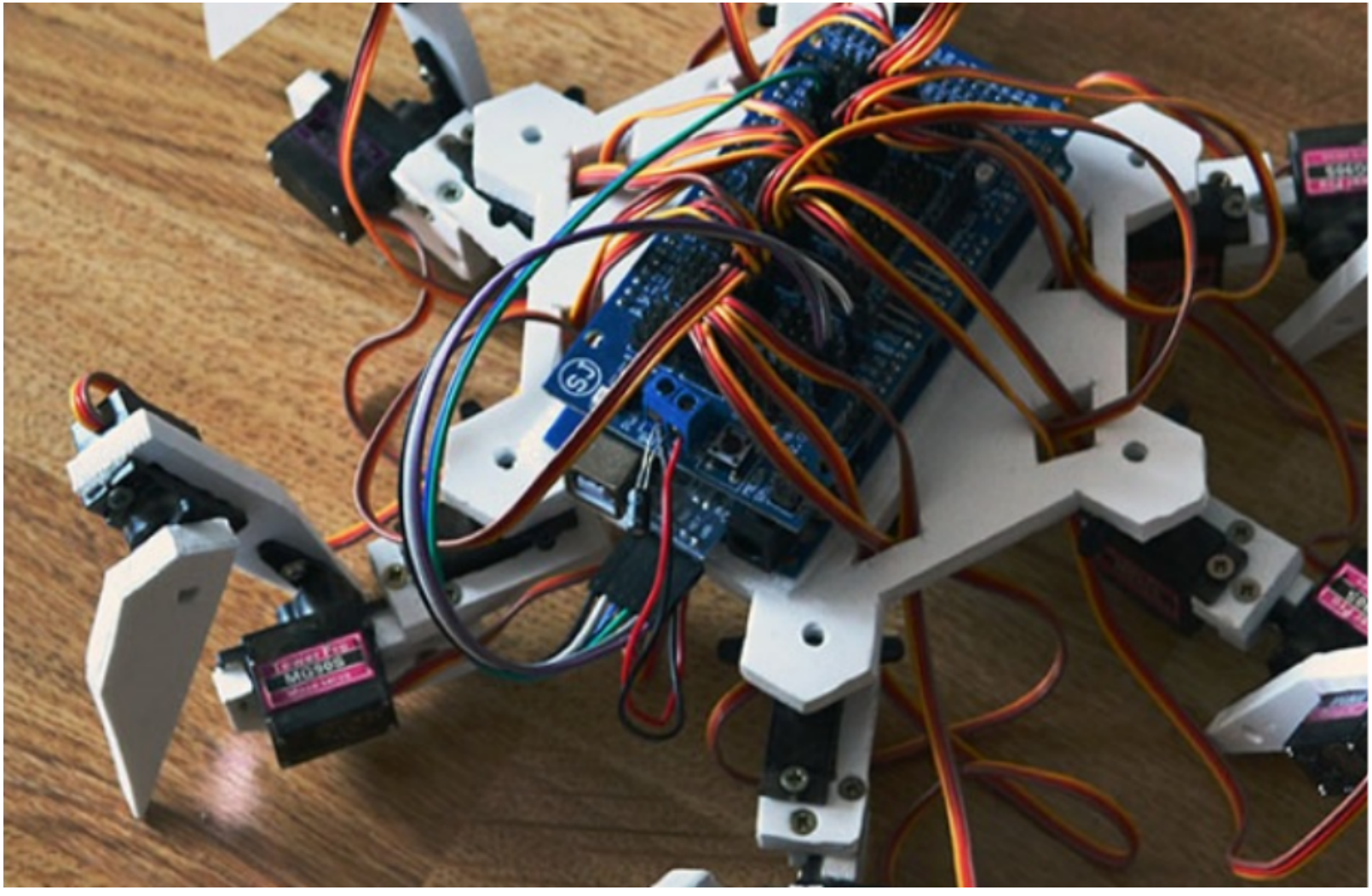
Introduction to game engines



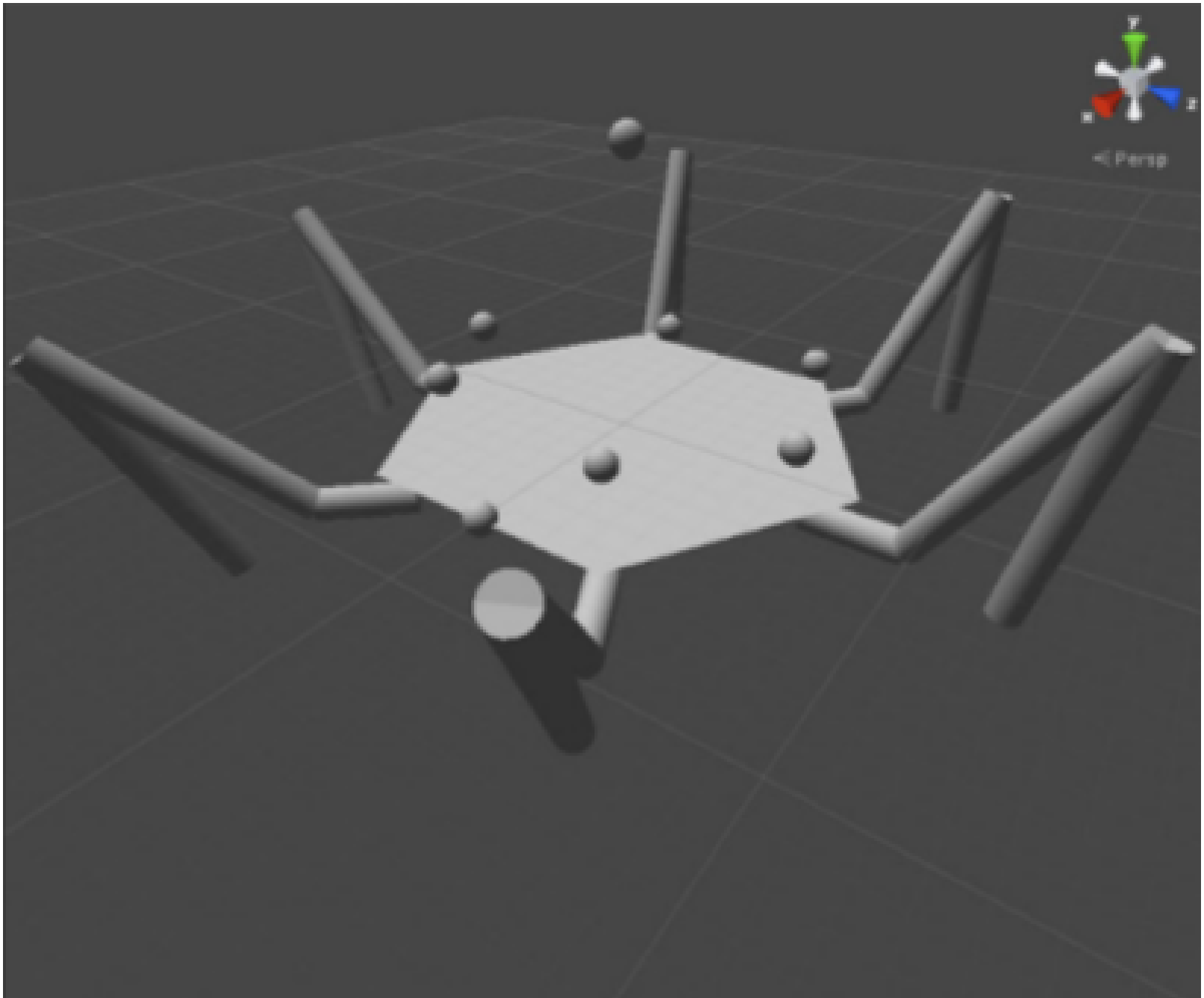
Introduction to game engines



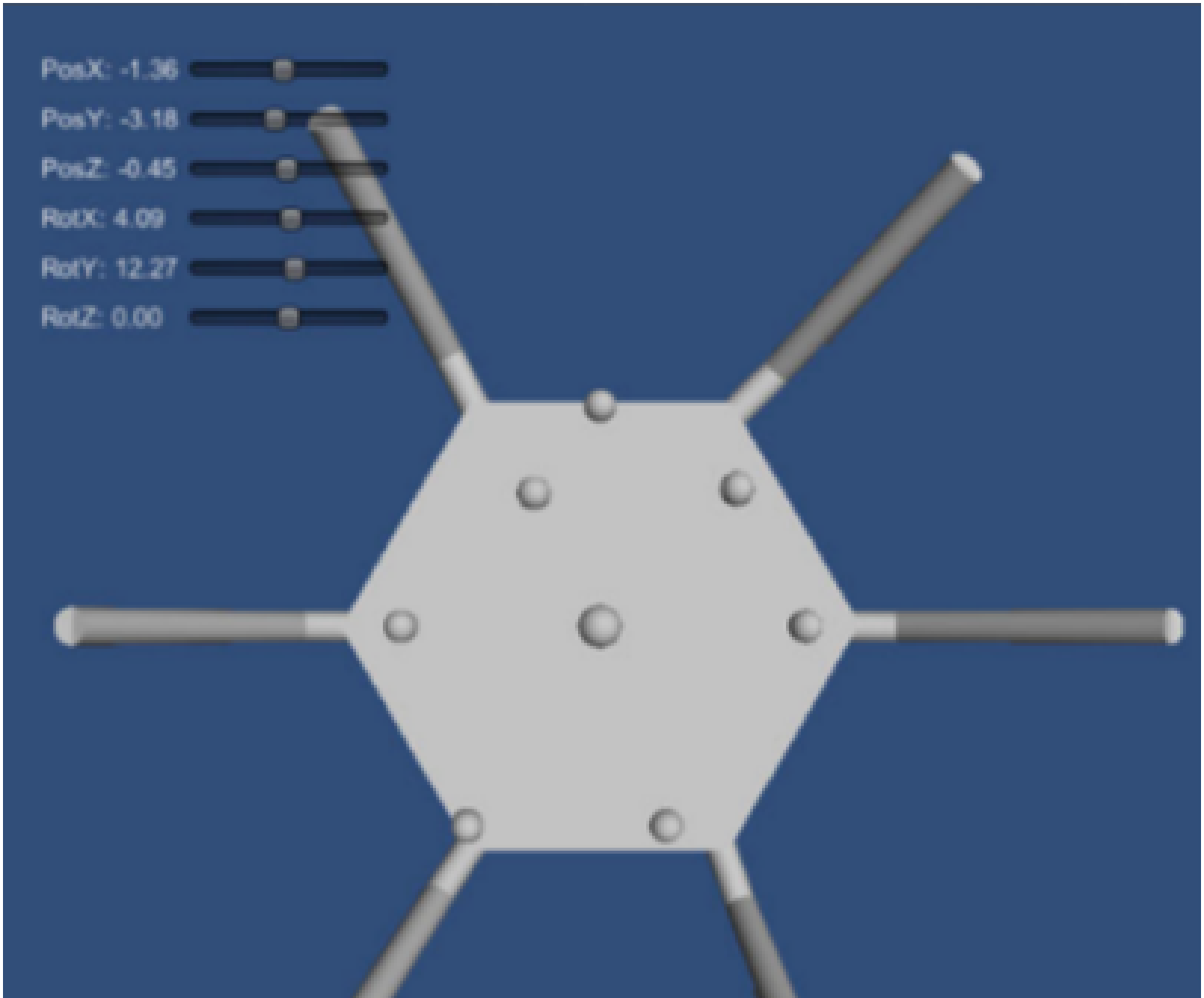
Introduction to game engines



Introduction to game engines



Introduction to game engines

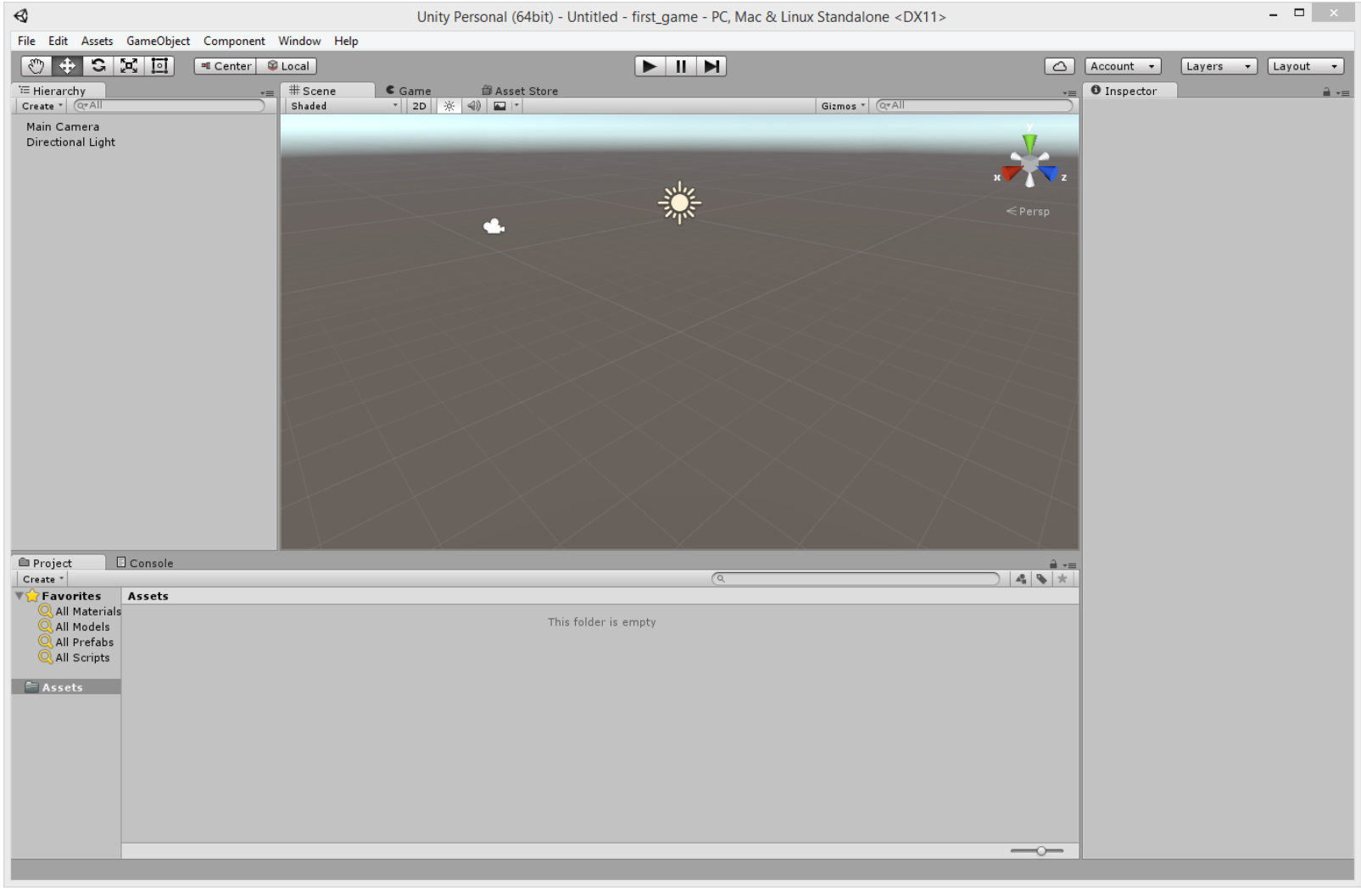


Why choose Unity

- Unity3D is built with simplicity and effectiveness in mind to allow both novice and advanced developers to maximize their game creation experience.
- Unity3D makes it possible to develop games of different genres: platformers, role playing games, first-person shooters, massive multiplayer online role playing games, simulations, or strategy games.
- Unity3D makes it possible to develop games for a comprehensive number of platforms: Android, iOS, Windows Phone 8, PC, Mac, Linux, PS3 and XBOX 360.
- Unity3D makes it possible to code the game using relatively high-level programming and scripting languages, including JavaScript, Boo, or C#.
- Unity3D offers the possibility to employ third-party plugins that greatly enhance the workflow and add some very interesting effects and functionalities.
- Unity3D includes a built-in access to the online assets store that provides material for our Unity projects: textures, characters, GUI systems, scripts and so on.

Unity3D's interface

By default, when we launch Unity3D for the first time, the project AngryBots should be open. The default layout is applied in Unity3D, and you will notice that the screen is divided into several sections or views, including the following:



Unity3D's interface

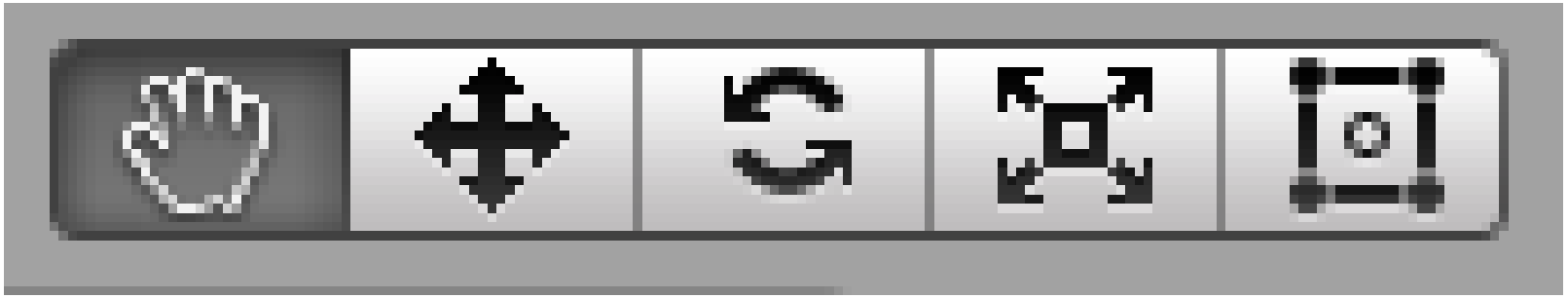
- The Scene view, where we can visualize and modify the scene we have created for our game
- The Hierarchy view, where we can see a list of all the objects included in our scene
- The Project view, which contains all assets used in the current project
- The Inspector view, which displays the properties of the object selected

Navigating in the scene

Arrow Movement

We can navigate through the scene using the arrow keys from the keyboard. Hold down the Shift key with an arrow to move faster.

Using the Hand Tool



- Move: (hand) Click-drag to drag the camera around (left, right, up, down).
- Orbit: (eye) Hold Alt and click-drag to orbit the camera around the current pivot point. This option is not available in 2D mode as the view is orthographic.
- Zoom: (magnifier) Hold Alt and right click-drag to zoom the Scene View. On Mac you can also hold Control and click-drag instead.

Shortcuts Without Using the Hand Tool

- Move: Hold Alt-Control and click-drag.
- Orbit: Hold Alt and click-drag.
- Zoom: Hold Alt and right click-drag.

Navigating in the scene

Flythrough Mode

We can access this mode by holding down the mouse right button inside the scene. We can then

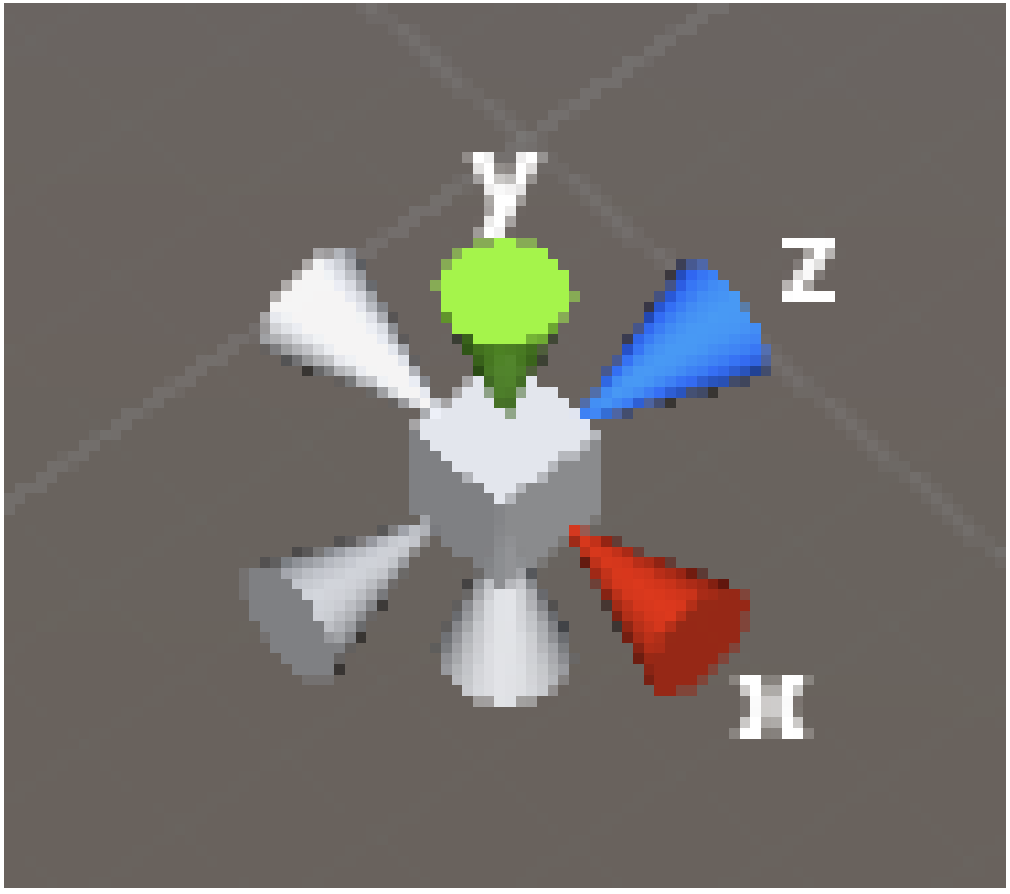
- navigate through the scene using the keys W, A, S, and D;
- float up and down using the keys Q and E,
- or look around by moving the mouse left, right, forward, or back.

In addition to navigating through the scene, we can also focus on one object by pressing the F (focus) key. This will cause Unity3D to move the camera so that the object is displayed on the screen.

Navigating in the scene

Scene Gizmo

We can view the scene from the x, y, or z axes using the gizmo located in the top-right corner of the window.



For example, clicking on the y (green) arrow will display the view from the y-axis. Clicking on the box in the middle of the gizmo will toggle the view between perspective and isometric modes.

Scene View Navigation:
`http://docs.unity3d.com/Manual/SceneViewNavigation.html`

Navigating through the one of complete projects scene

Navigating through the one of complete projects scene:
`http://unity3d.com/learn/resources/downloads`
Navigate through the project scene using the different navigation modes explained earlier. Select some of the items and look at their features using the Inspector. Look at the scene from the x, y, and z axes. Open the Console window and look for any message in this window. Play the scene using the play icon located at the top of the window (that is, black triangle pointing right) or by pressing Ctrl + P. To exit the play mode, we can either click on the play icon or use the shortcut Ctrl + P.

Creating a new project and scene

- Select **File | New Project**. A window labeled **Project Wizard** should appear.
- Choose and set a directory and a name for our project (`<base_dir>/unity/first_game`).
- Do not select any of packages.
- Click on **Create Project**.

Adding objects to a scene - theory

Unity3D makes it possible to add different types of built-in objects, including

- 3D primitives (for example, spheres, cubes, cylinders, or planes),
- lights (for example, point, directional, or area lights),
- Graphical User Interface (GUI) elements (for example, text or textures),
- or cameras.

Each object can be added by selecting: **Game Object** and then object type you want to add from submenu. For example to add cube select: **Game Object** | **3D Object** | **Cube**.

Objects – transformations

Once an object has been created, we can

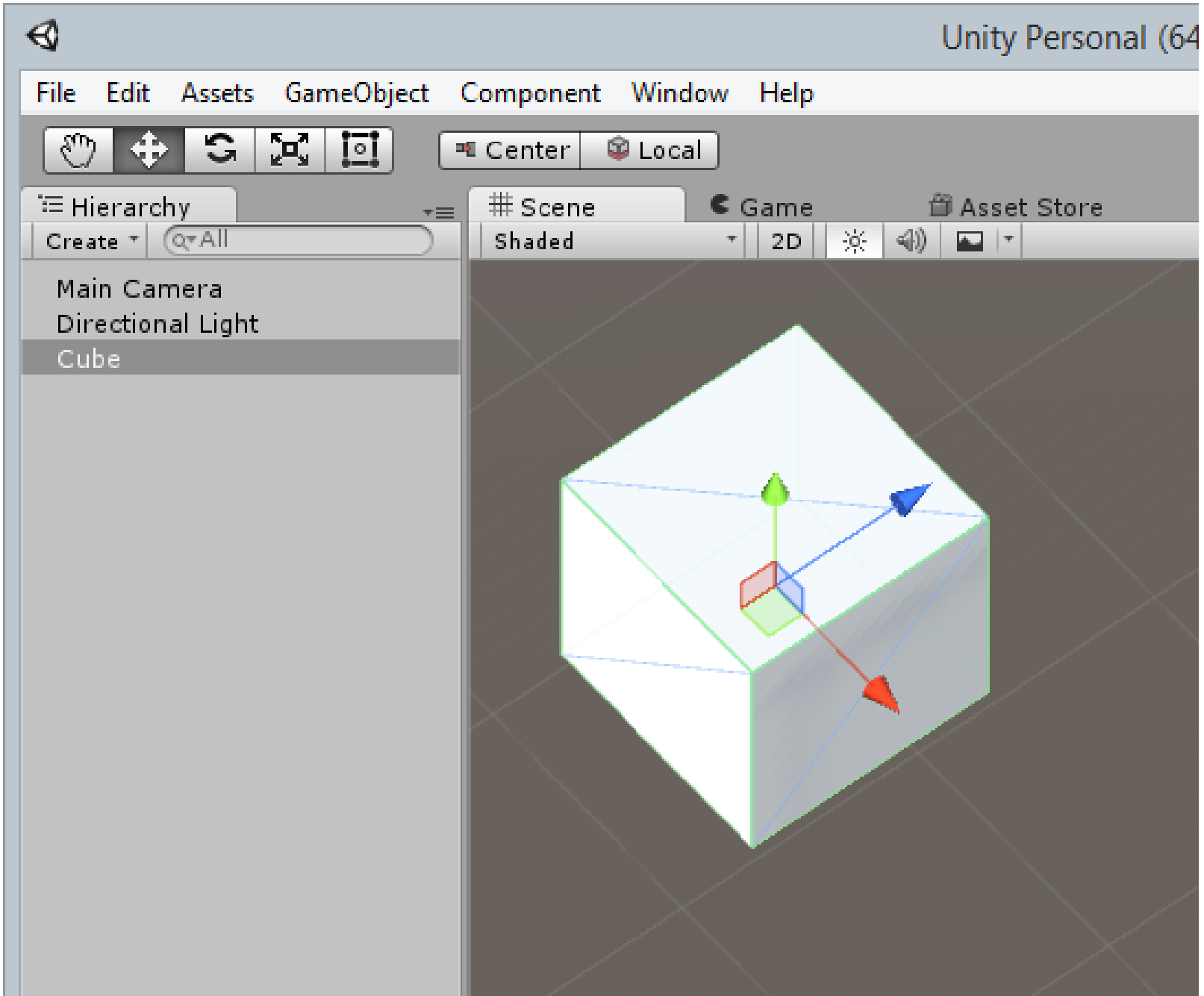
- change its properties using the Inspector
- or by directly moving, rotating, or scaling this object in the scene view.

Having the object, we can use the buttons located at the top-left corner of the scene view to apply transformations. We can also do this using the keys W, E, and R respectively.

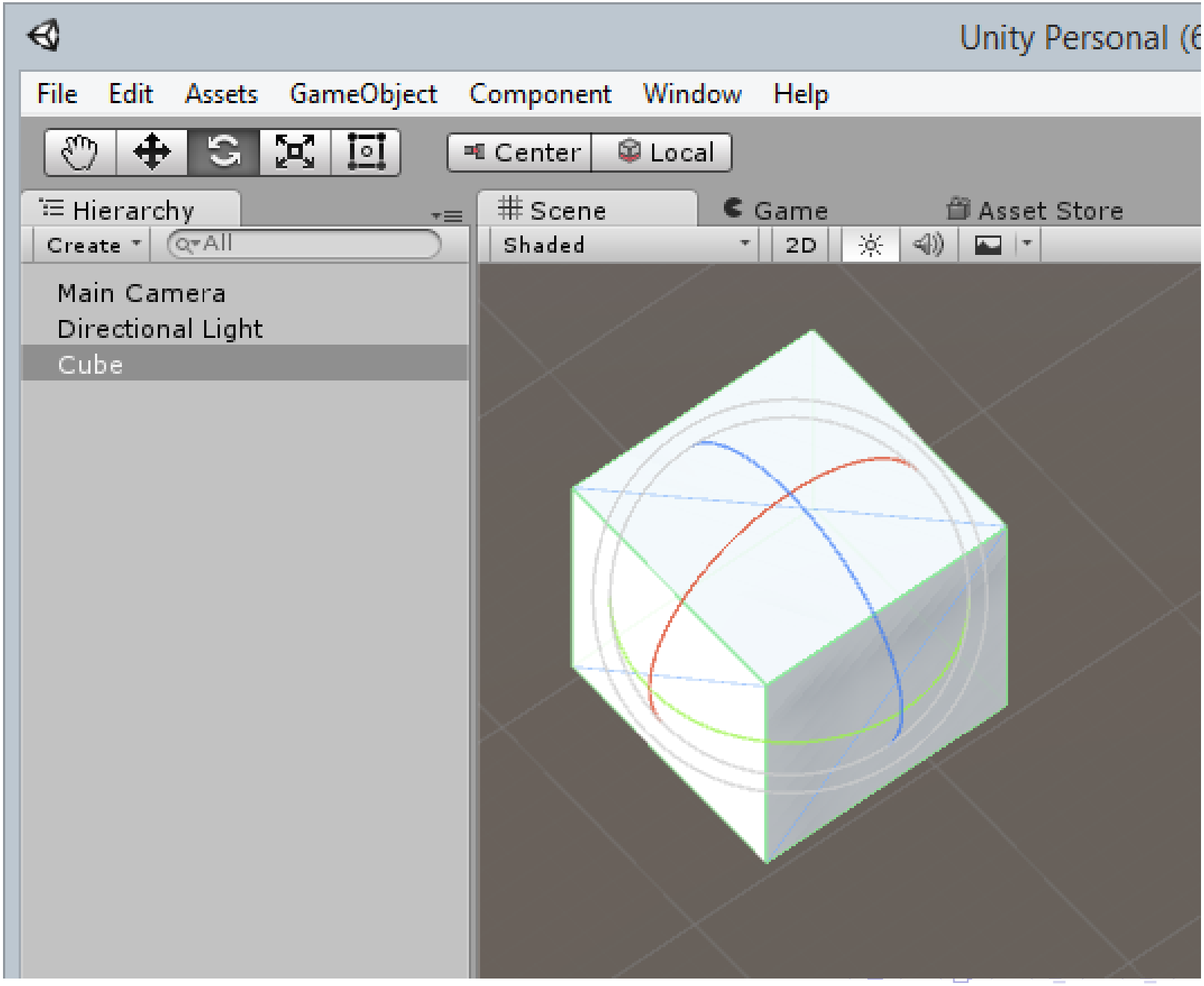
Objects – transformations

For example, if we select an object and click on the crossed-arrows button (or alternatively press the W key), three axes will appear on the object: a green axis (y), a blue axis (z), and a red axis (x). Dragging any of these axes will move the object in the corresponding direction. The same applies to rotating and scaling an object.

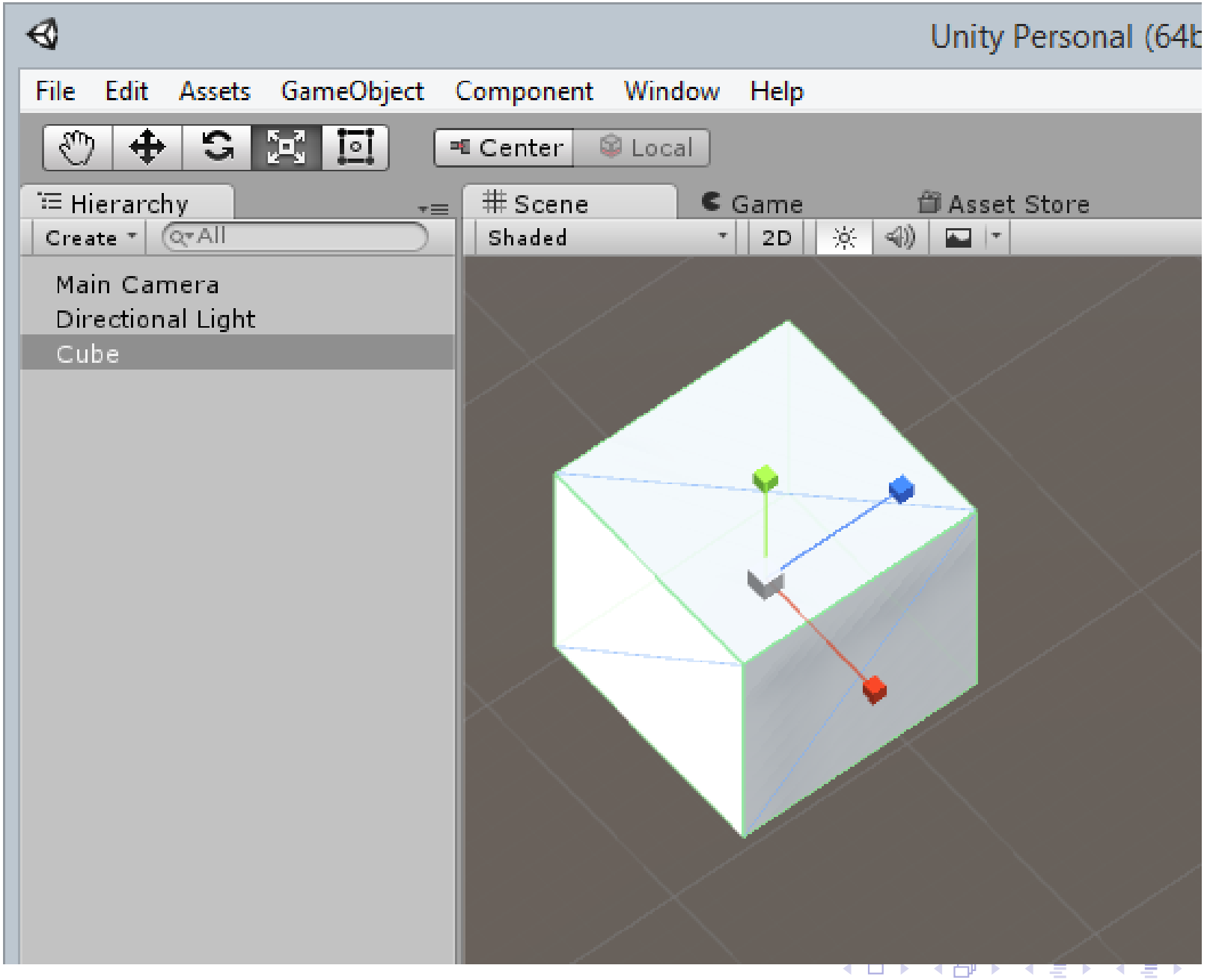
Objects – transformations



Objects – transformations



Objects – transformations



Adding objects to a scene - practice

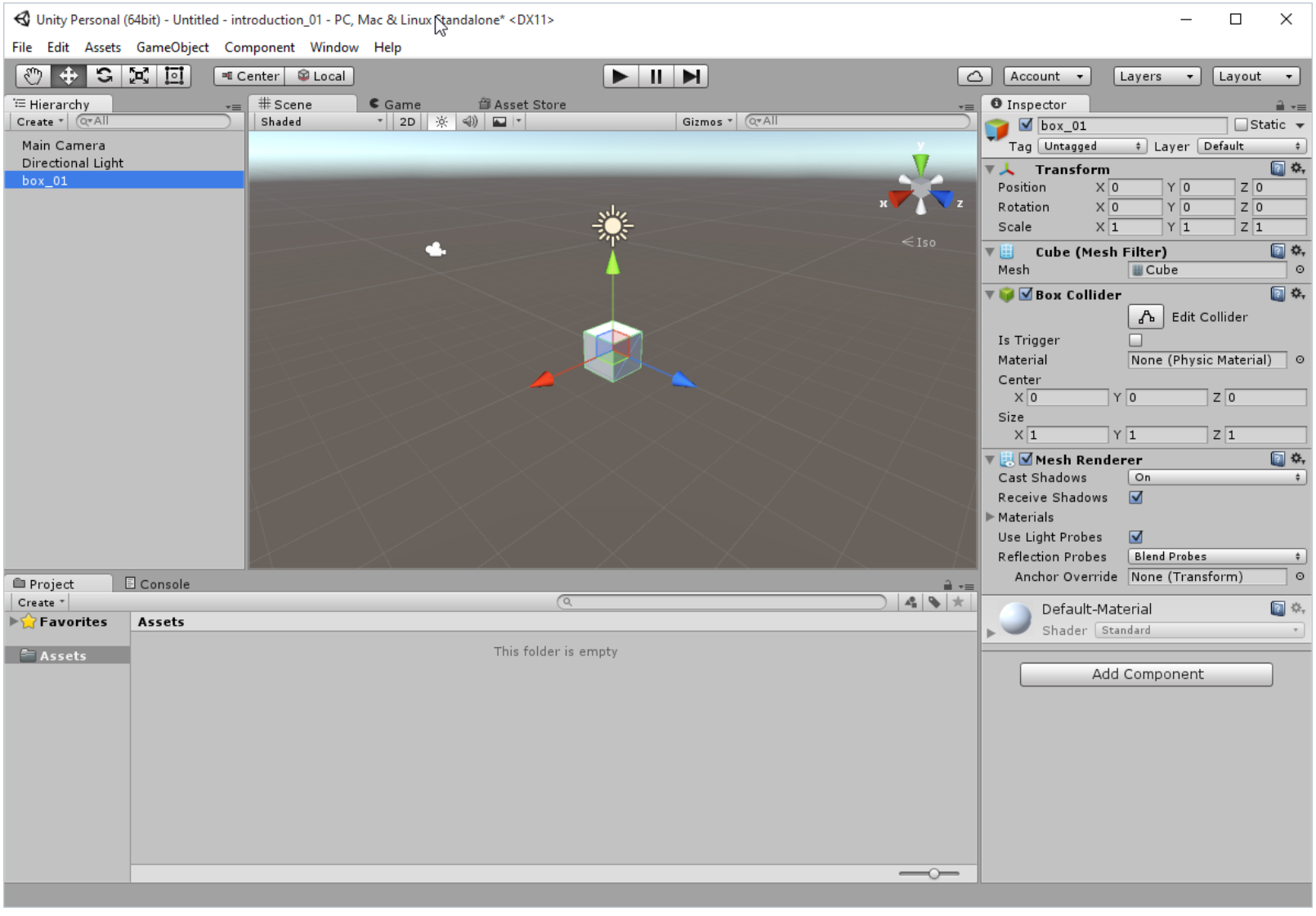
Creating a cube

Add an object

- Select: **Game Object** | **3D Object** | **Cube**.
- In the **Hierarchy** window, change the name of this cube to `box_01`.
- Make sure that the object is selected by clicking on it in the **Hierarchy** or in the **Scene** view.
- In the **Inspector** window, change the x, y, and z position parameters of this object to (x=0, y=0, z=0).
- Double-click on this object in the hierarchy to focus the camera on it.

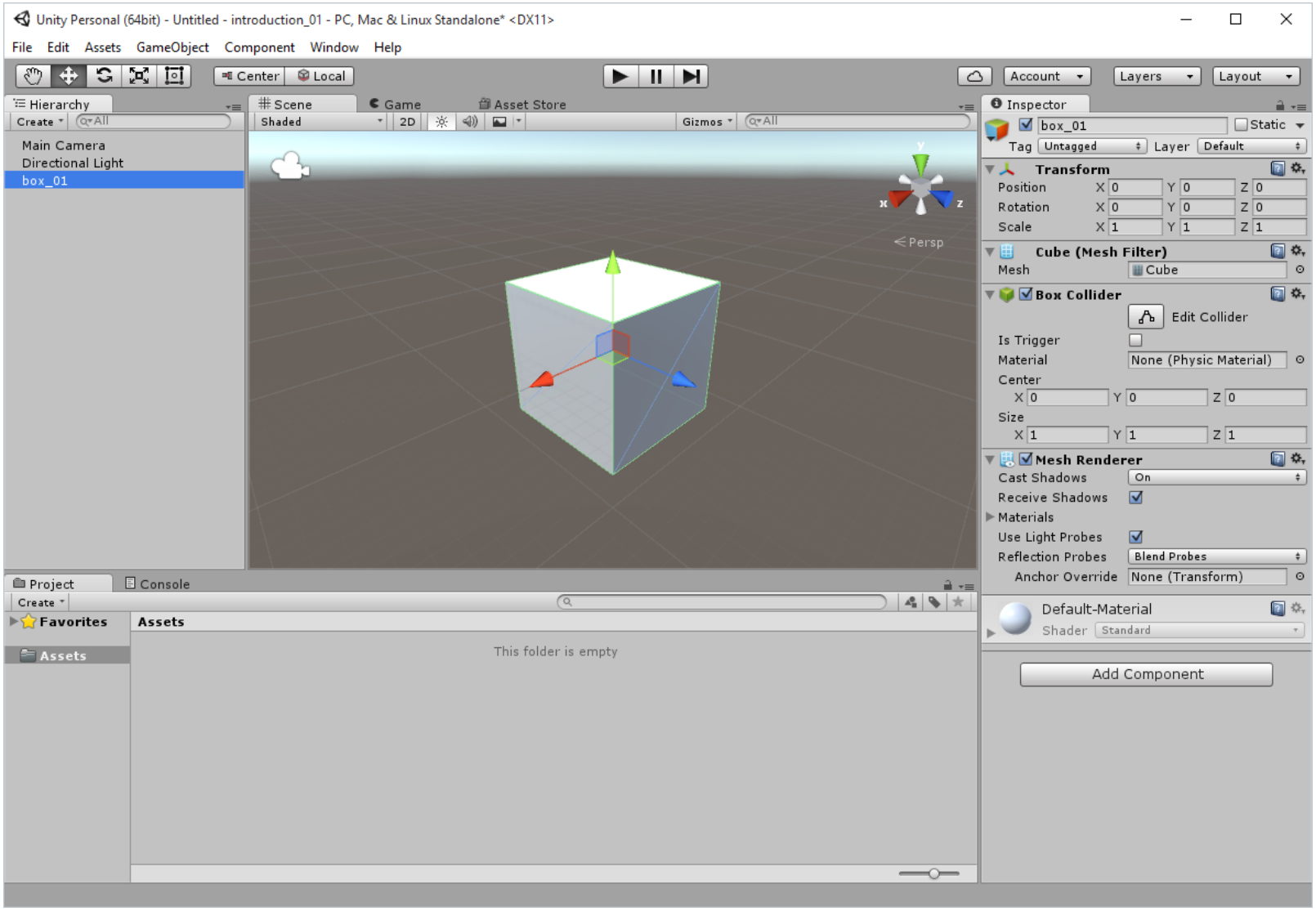
Adding objects to a scene - practice

Creating a cube



Adding objects to a scene - practice

Creating a cube



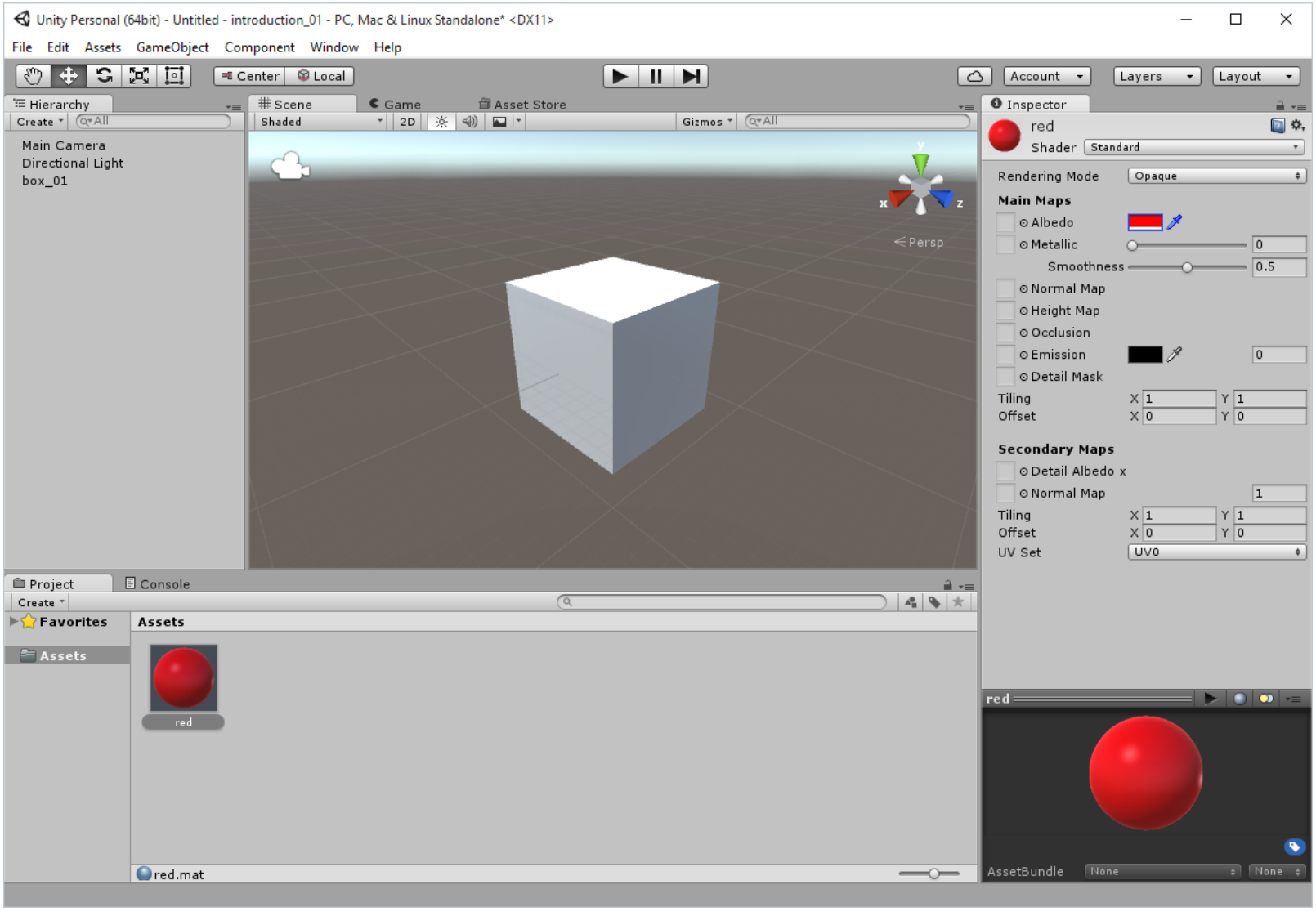
Create a material

Add a color

- To create and apply a material (for example, color or texture) to the box first from the **Project** window, select **Create** | **Material**.
- In the **Assets** window change the name (which is initially `New Material`) of newly created material new material to `red`.
- Make sure that this material is selected, and look at its properties in the **Inspector** window.
- One of the properties of this object is main color — we will modify this property by clicking on the (white) rectangle in section **Main Maps**.
- This should open a window labeled **Color**. This window makes it possible to pick a color for this material.
- Pick a red color and close the **Color** window. We should see that the **Preview** of the material now shows a red sphere.

Create a material

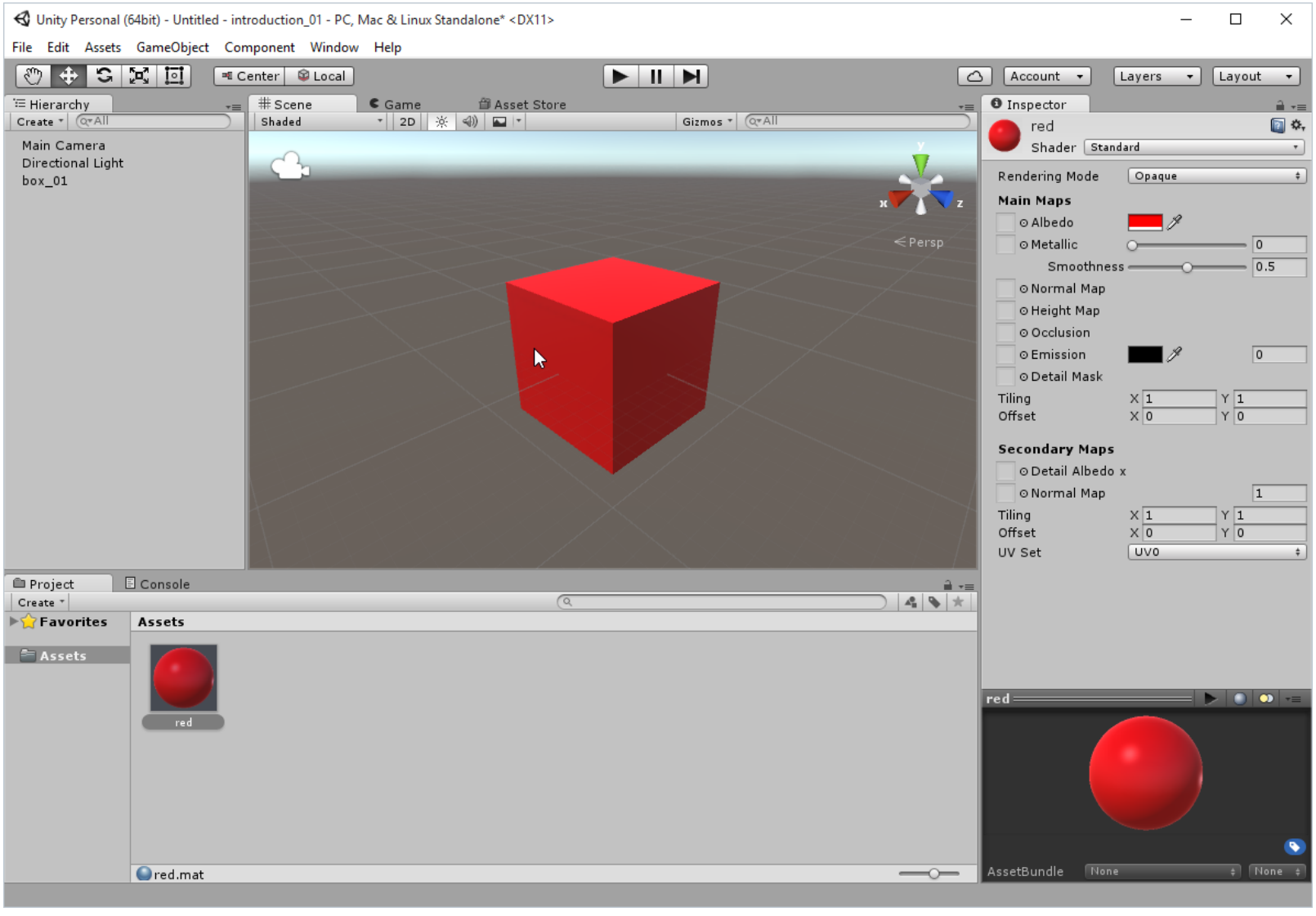
Add a color



- Apply the material to the box. This step can be done in at least two ways.
  - The first way in which we can do is as follows:
    1. Drag-and-drop the material from the **Assets** window to the `box_01` in **Scene** window.
  - The second way is as follows:
    1. Click on the object `box_01`.
    2. Look at the **Inspector** window and click on the **Materials** attribute of the **Mesh Renderer** component for this object.
    3. Click on the circle to the right of the label **Default-Material** so that we can change the material. A new window will appear.
    4. From the new window, click on the tab labeled **Assets** and type the text red in the search field located at the top of this window.
    5. This should return one result, which is the material we have just created. Click on the material red and close the window. The cube should turn to red.

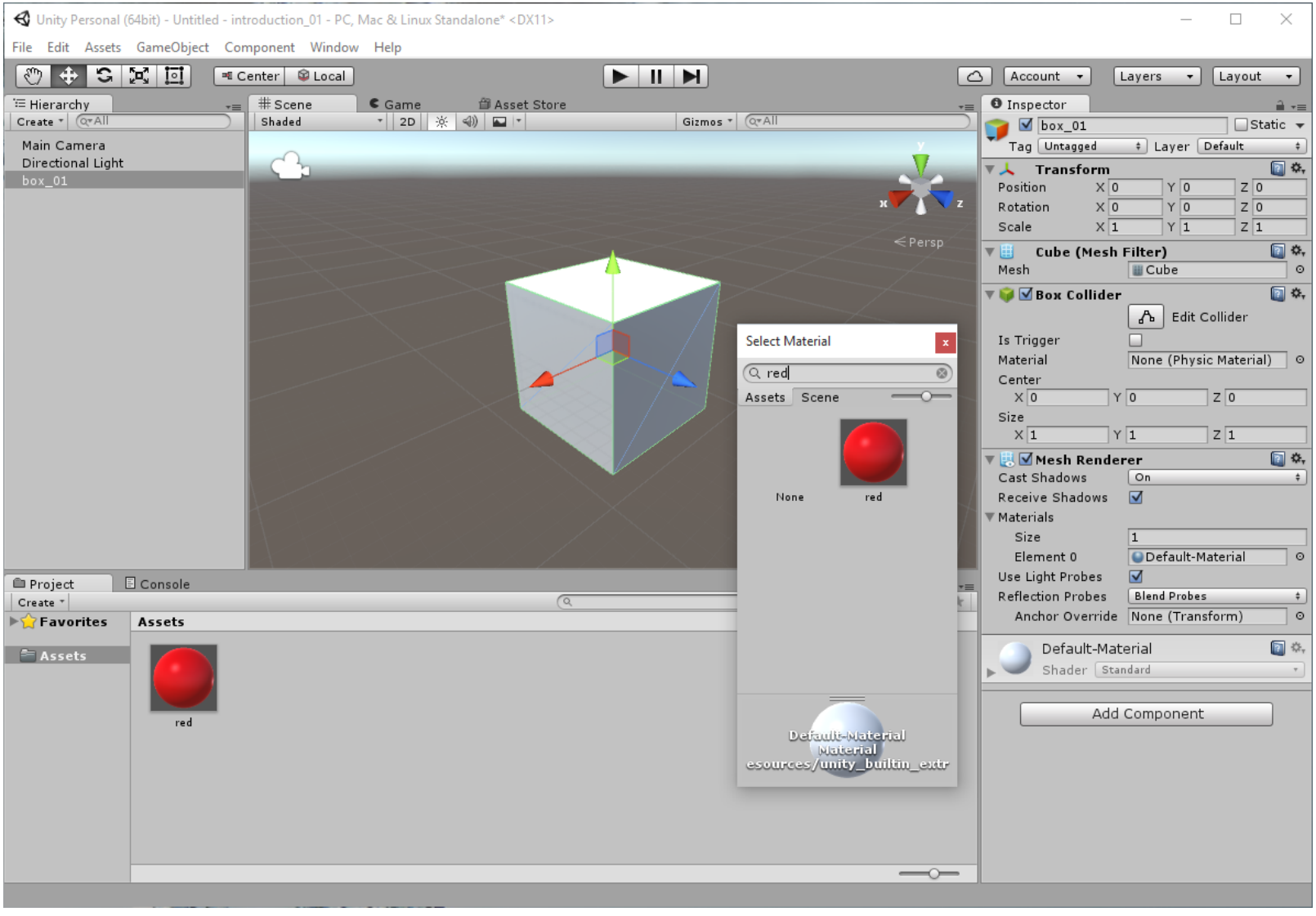
Create a material

Apply a color



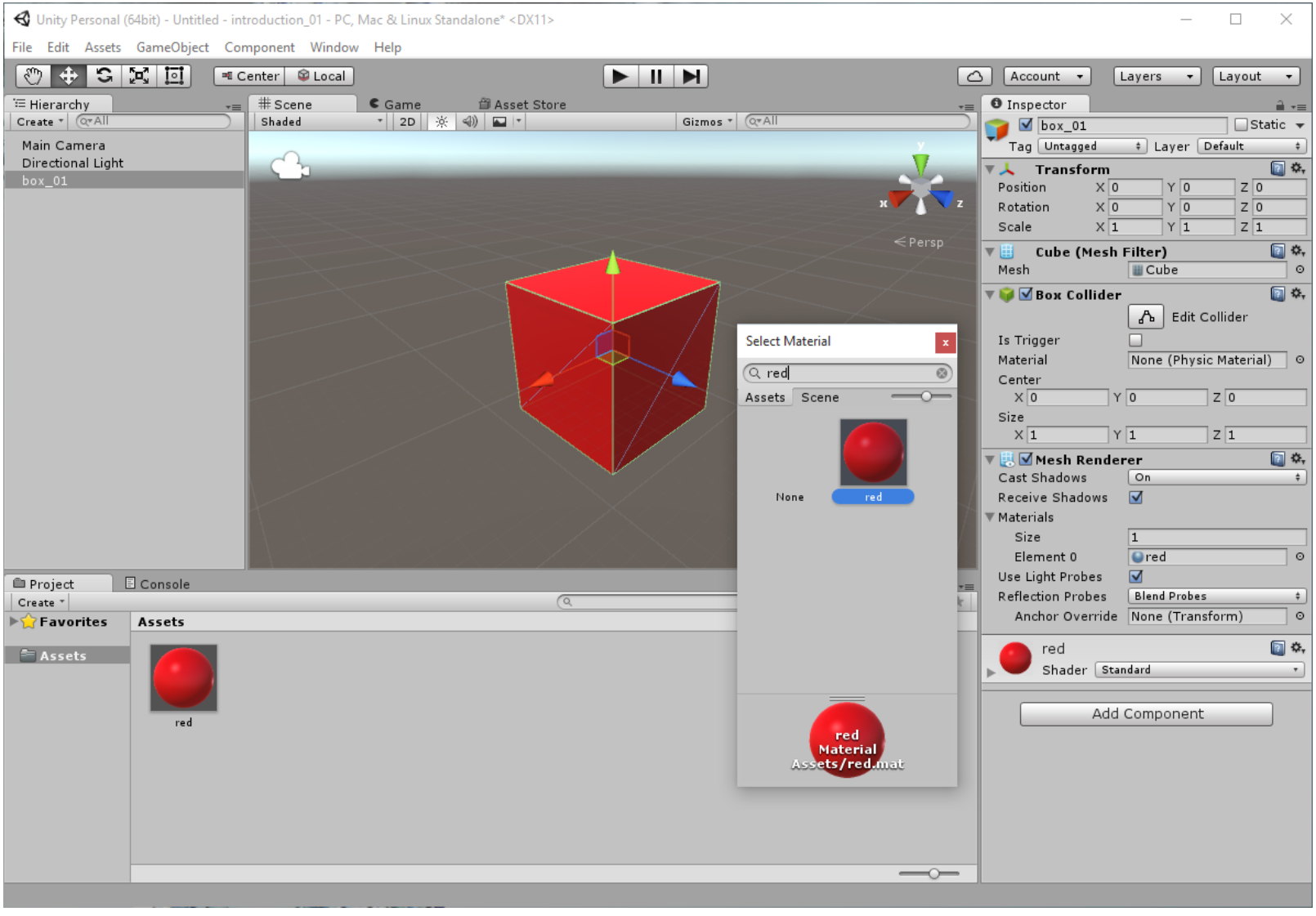
Create a material

Apply a color



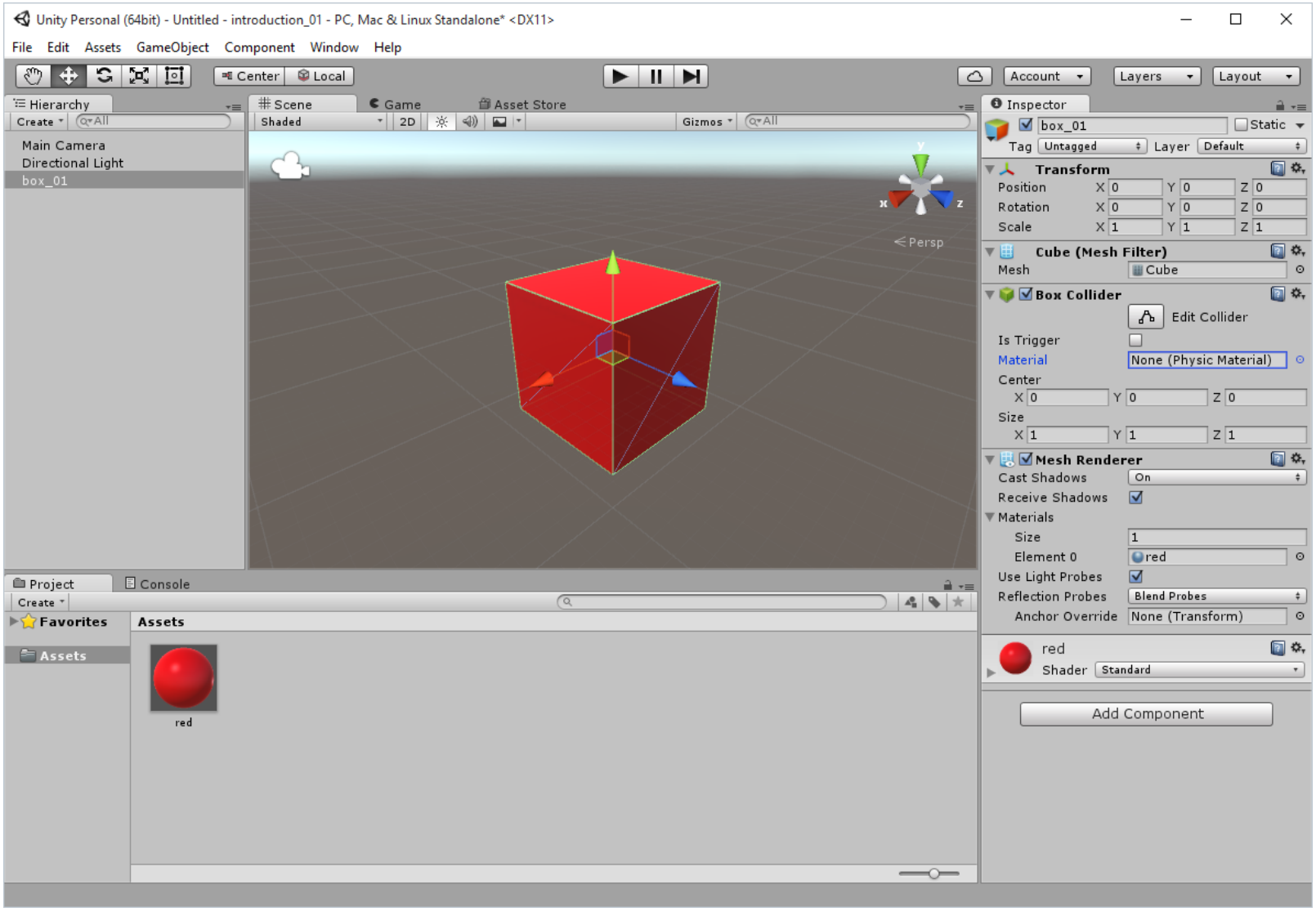
Create a material

Apply a color



Create a material

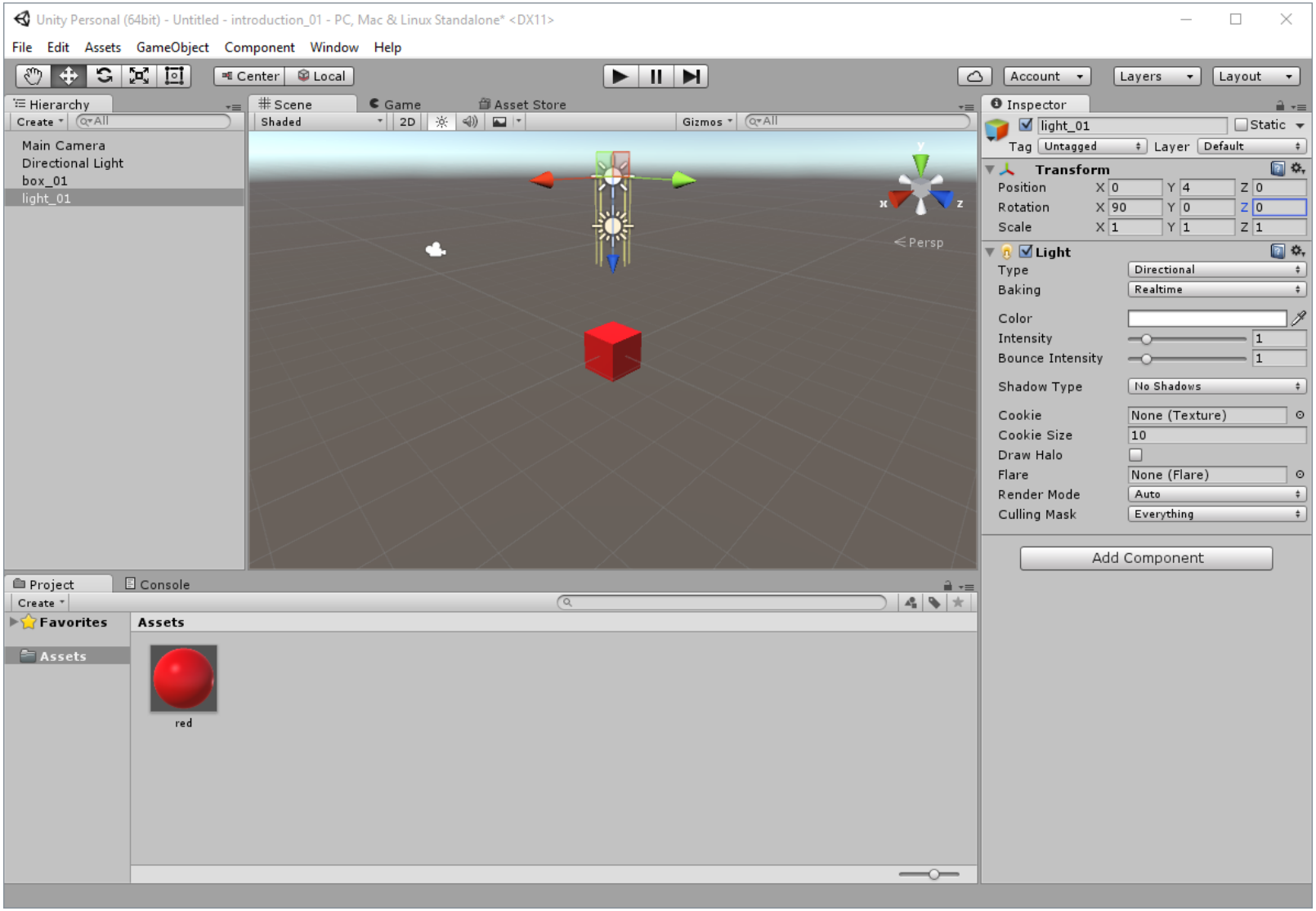
Apply a color



Add a light

- Select: **Game Object** | **Light** | **Directional Light**.
- In the **Hierarchy** view, change the name of this light from `Directional Light` to `light_01`.
- In the **Inspector** window, change the position of this light to (x=0, y=4, z=0).
- In the **Inspector** window, change the rotation parameters of this object to (x=90, y=0, z=0). This will rotate the light around the x axis, so that it points downwards (that is, toward the cube).

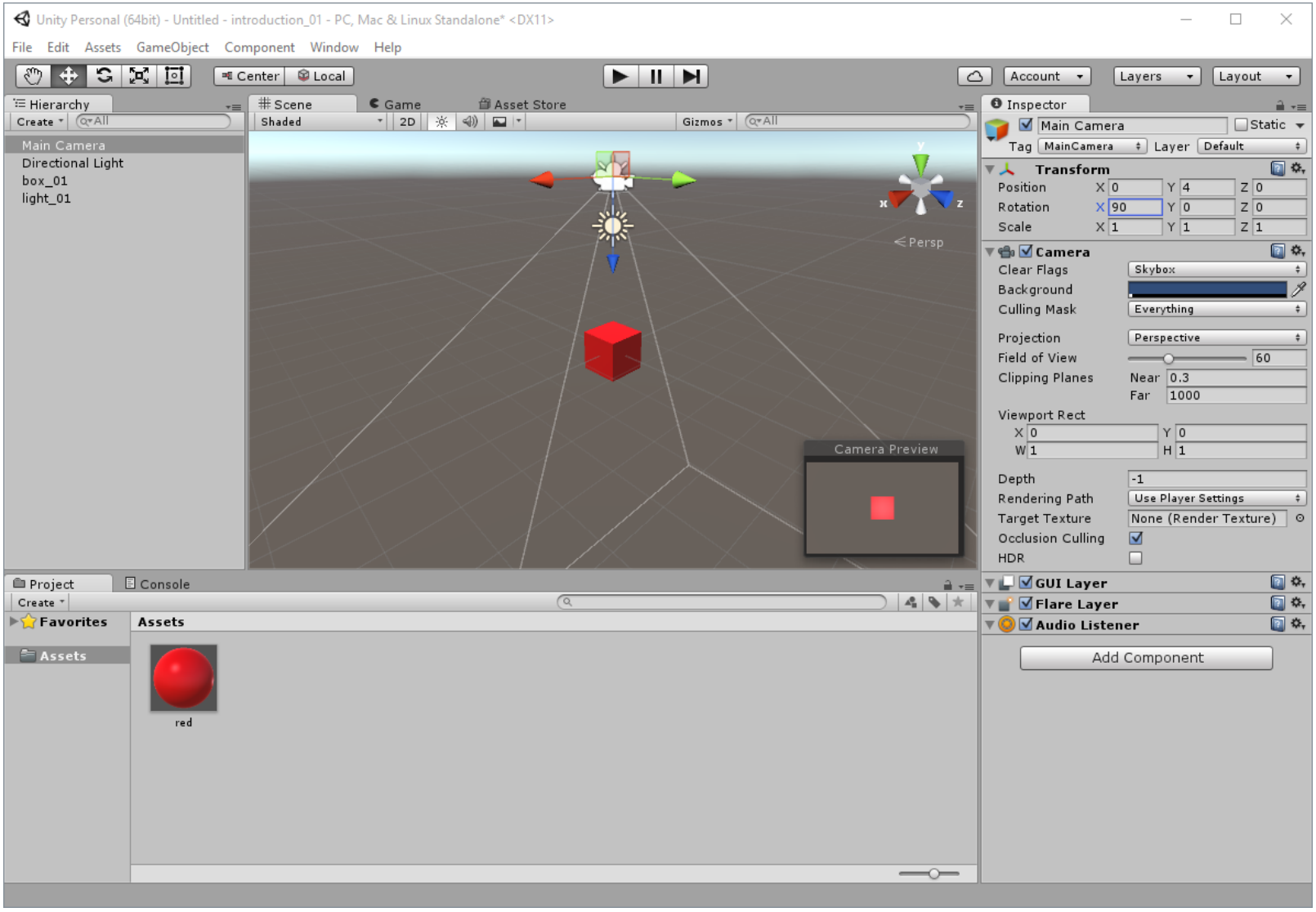
Add a light


Unity Personal (64bit) - Untitled - introduction_01 - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Account Layers Layout
Hierarchy
Create All
Main Camera
Directional Light
box_01
light_01
Scene Game Asset Store
Shaded 2D Gizmos All
Persp
Inspector
light_01 Static
Tag Untagged Layer Default
Transform
Position X 0 Y 4 Z 0
Rotation X 90 Y 0 Z 0
Scale X 1 Y 1 Z 1
Light
Type Directional
Baking Realtime
Color
Intensity 1
Bounce Intensity 1
Shadow Type No Shadows
Cookie None (Texture)
Cookie Size 10
Draw Halo
Flare None (Flare)
Render Mode Auto
Culling Mask Everything
Add Component
Project Console
Create
Favorites
Assets
Assets
red

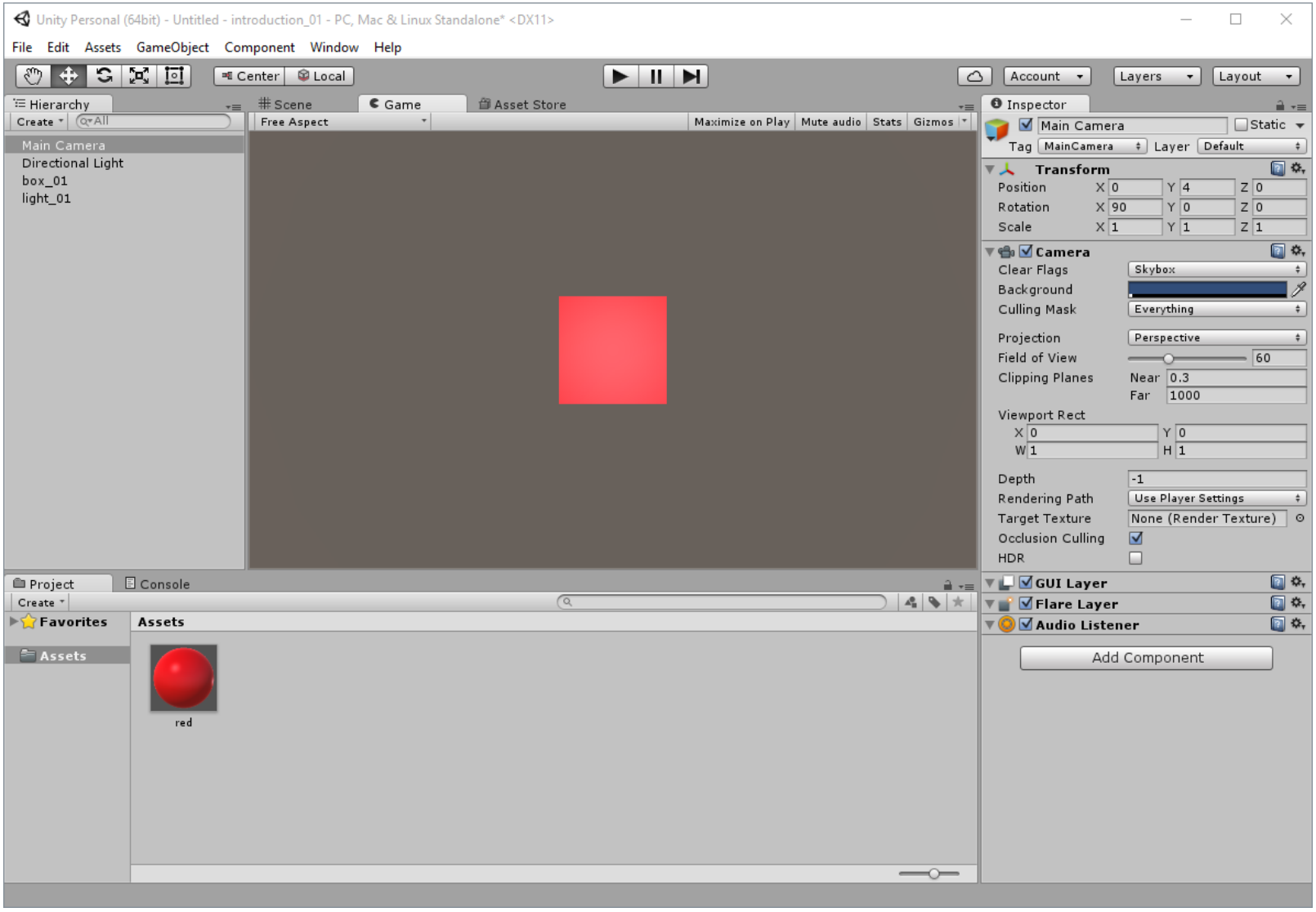
Move the camera so that the objects can be seen from the camera

- The camera is already present in the **Scene** and is labeled as `Main Camera` by default.
- Select this camera (from the **Hierarchy** or **Scene** view).
- In the **Inspector** window, change the position of this camera to (x=0, y=4, z=0).
- In the **Inspector** window, change the rotation of this camera to (x=90, y=0, z=0). This will, as it was done for the light, rotate the camera around the x axis, so that it points downwards (that is, toward the cube). We can check the camera
  - by looking at the camera view (that is, the rectangle located at the bottom-right corner of the **Scene** view),
  - or by clicking on the **Game** tab (to display the game view).

Move the camera so that the objects can be seen from the camera

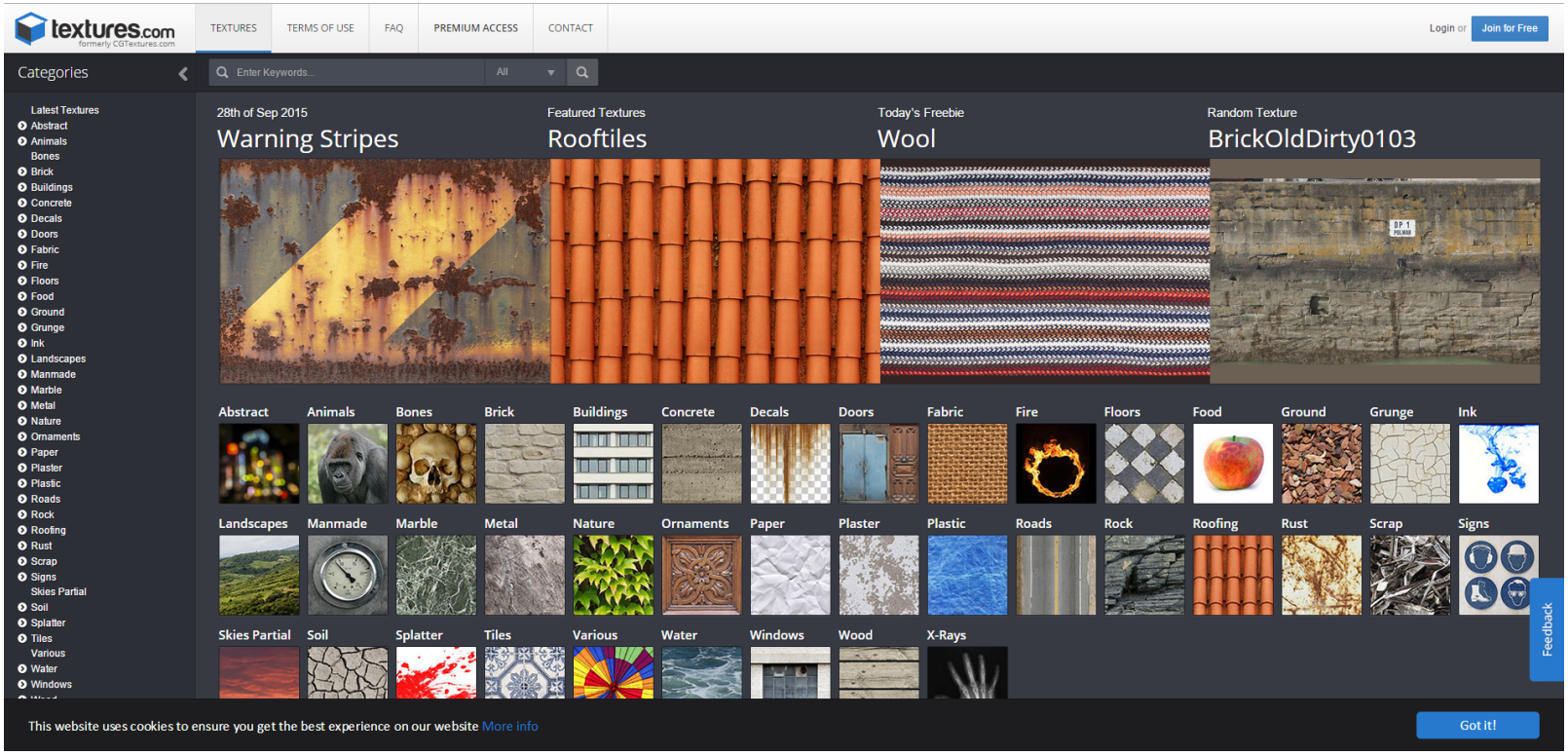


Move the camera so that the objects can be seen from the camera



Get textures

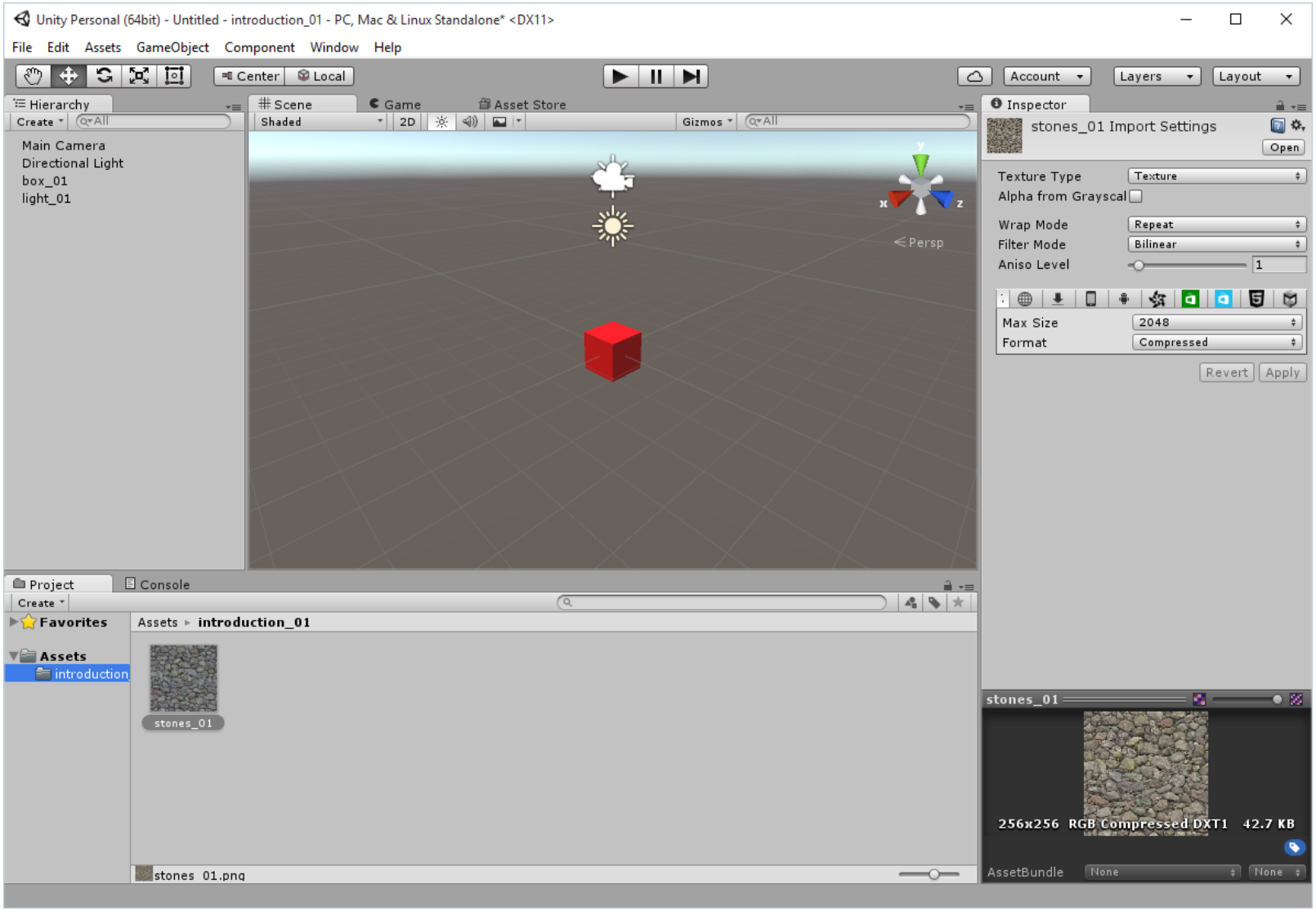
http://www.textures.com/



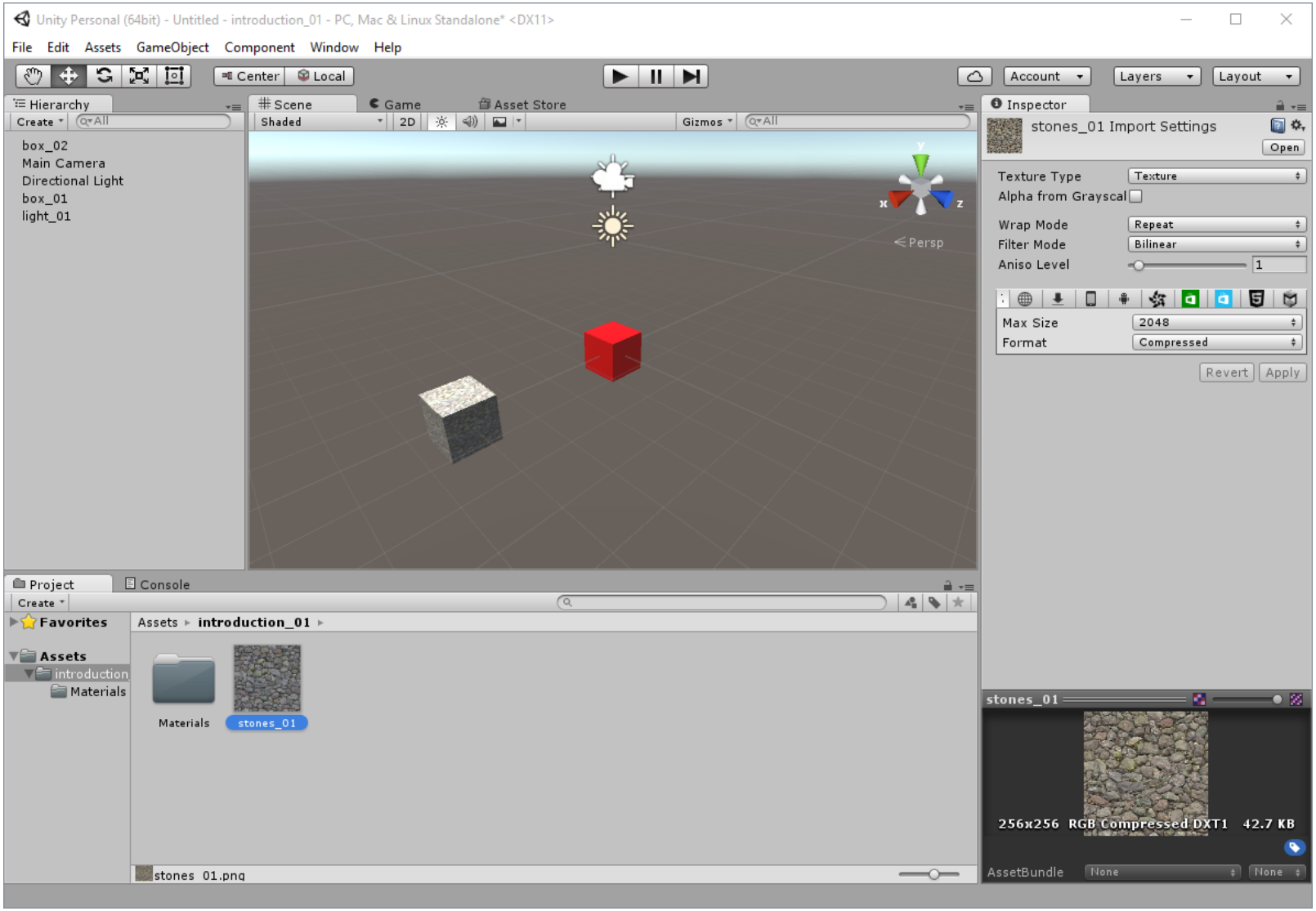
Import texture to the project

- Switch to Unity3D.
- Select the folder labeled **Assets** in the **Project** view (click on it once).
- Create a new folder within this folder (from the **Project** window, select **Create** | **Folder**).
- Rename the new folder `introduction_01`.
- Select this folder (that is, click on this folder once).
- Select: **Assets** | **Import New Asset**.
- Browse to the location where the texture was saved on our computer.
- Select the texture and click on **Import**.
- An asset labeled with the name of the texture should now appear in the folder `introduction_01`.
- In the **Hierarchy** view, duplicate the game object labeled `box_01`.
- Call this new object `box_02`.
- Change this object's position to (x=4, y=0, z=0).
- Drag-and-drop the texture that we have just imported to the object `box_02` either in the **Scene** view or in the **Hierarchy** view.
- The object labeled `box_02` should now feature a texture.

Import texture to the project



Import texture to the project



Import texture to the project

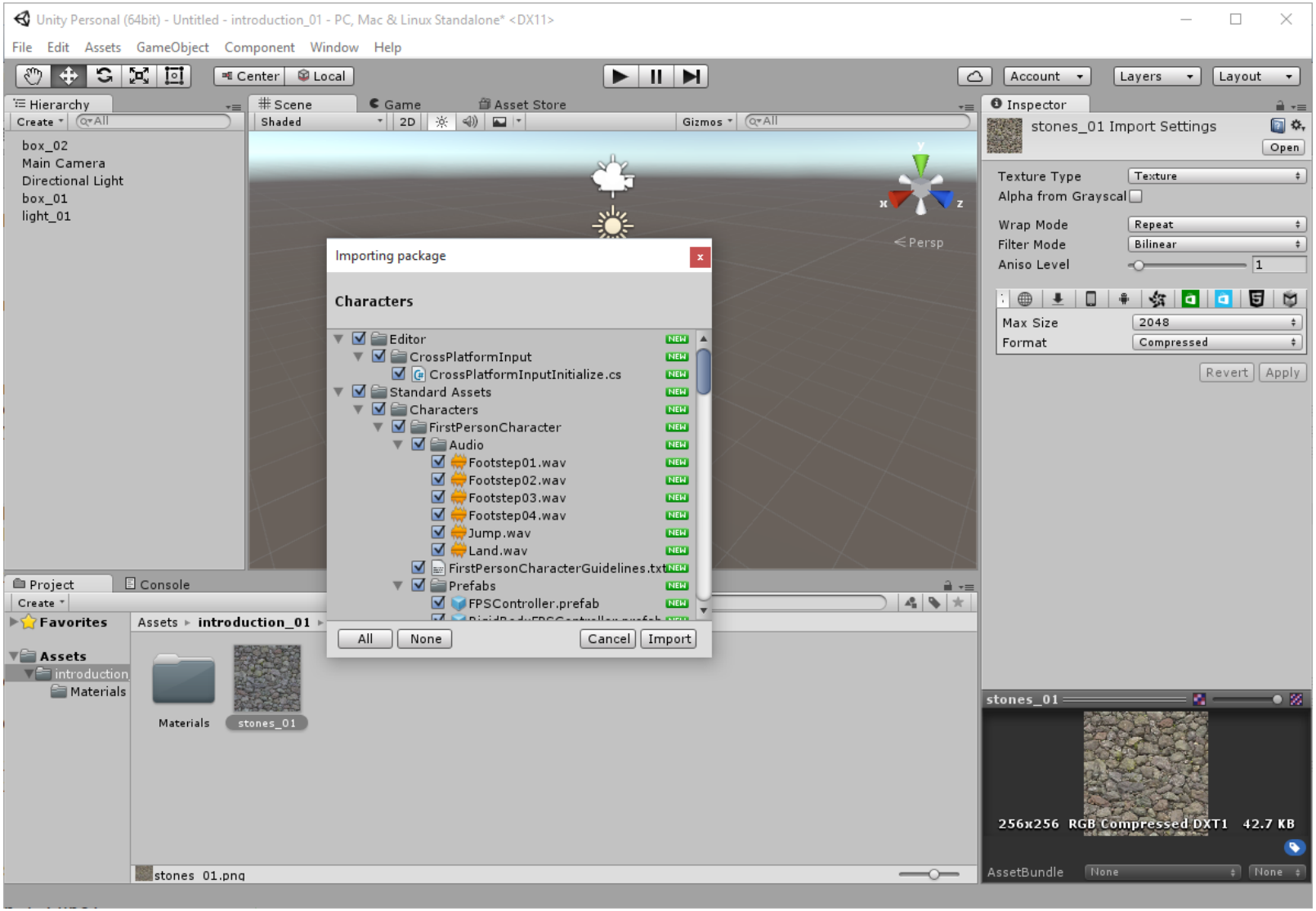
Note that by importing this image, Unity3D has automatically created the corresponding textures that will be used in the project. These textures are located in a folder called **Materials**, which is within the folder `introduction_01`.

Implementing first- and third-person views

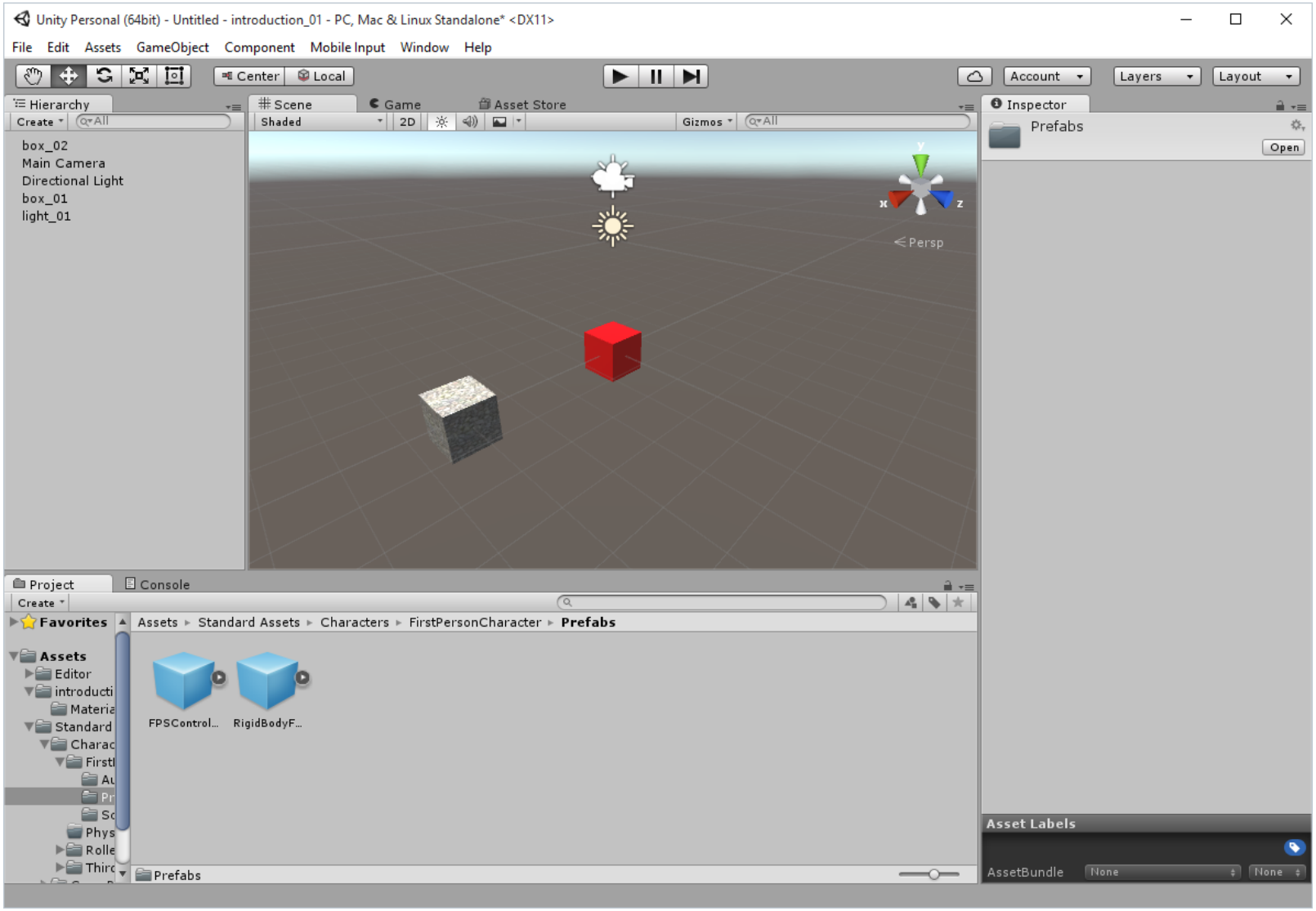
In most cases we need to navigate through the game using either first- or third-person view. This requires using a camera and the ability to move it based on the players' keyboard entries. As you can guess, Unity3D includes built-in objects to implement both types of navigation. If we have chosen to import these assets when creating our project, these assets are located in the **Project** view, inside the folder **Standard Assets | Characters**, and are named **FirstPersonCharacter** and **ThirdPersonCharacter**. If you haven't done so yet, we will need to import the character controller package as follows:

- Select: **Assets | Import Package | Characters**.
- A new window labeled **Importing** package will appear.
- Click on the button labeled **Import**.
- This will import the character controllers in our project.

Implementing first- and third-person views



Implementing first- and third-person views



Prefabs

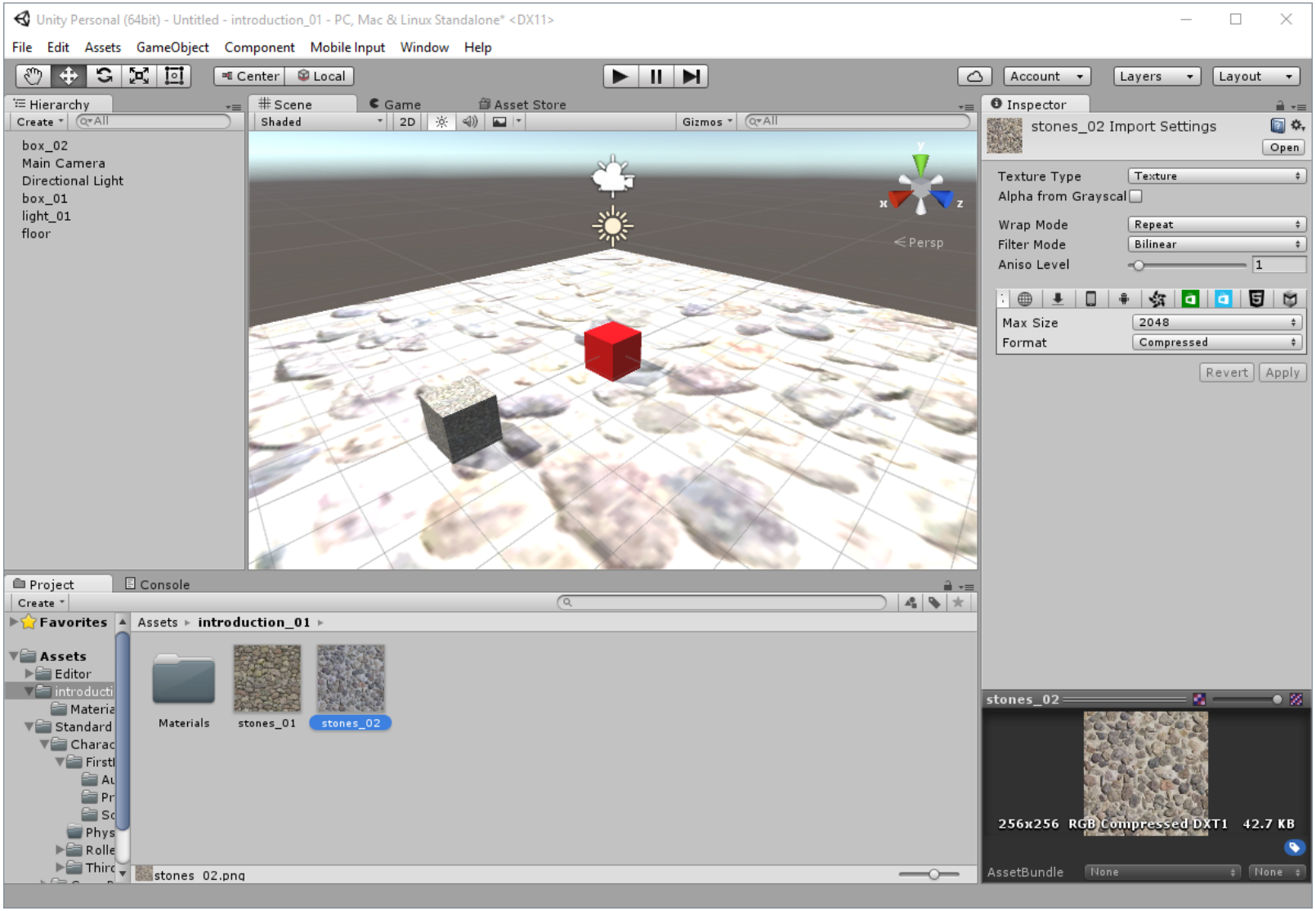
Once we have imported this package, both controllers appear with a blue box. This indicates that they are prefabs. Prefabs are comparable to templates, and make it possible to reproduce similar objects based on the same template, without the need to recreate them. Once a template is created and instantiated, if the prefab is modified, all instances will also be modified accordingly.
If we click on any of these prefabs, we will see in the Inspector window that they include a set of components and attributes such as gravity, walk speed, or run speed, which can be modified and affect the behavior of the controller accordingly.

Add the ground

Before we add our first-person controller to the scene, we will create an object that will act as the ground on which the player will be able to walk or run.

- Create a new box (**Game Object** | **3D Object** | **Cube**).
- Rename this box `floor`.
- In the **Inspector** window, change the position of this object to (x=0, y=-1, z=0) and its scale properties to (x=20, y=1, z=20). This will scale the cube on the x and z axes.
- To apply a texture to floor cube find some nice texture and download it.
- In Unity3D, select the folder `introduction_01` in the **Project** window, so that the image can be imported in this folder.
- Import this texture as explained previously.
- Apply the texture to the object labeled `floor`.
- Click on the object labeled `floor` and look at its properties in the **Inspector** window.

Add the ground

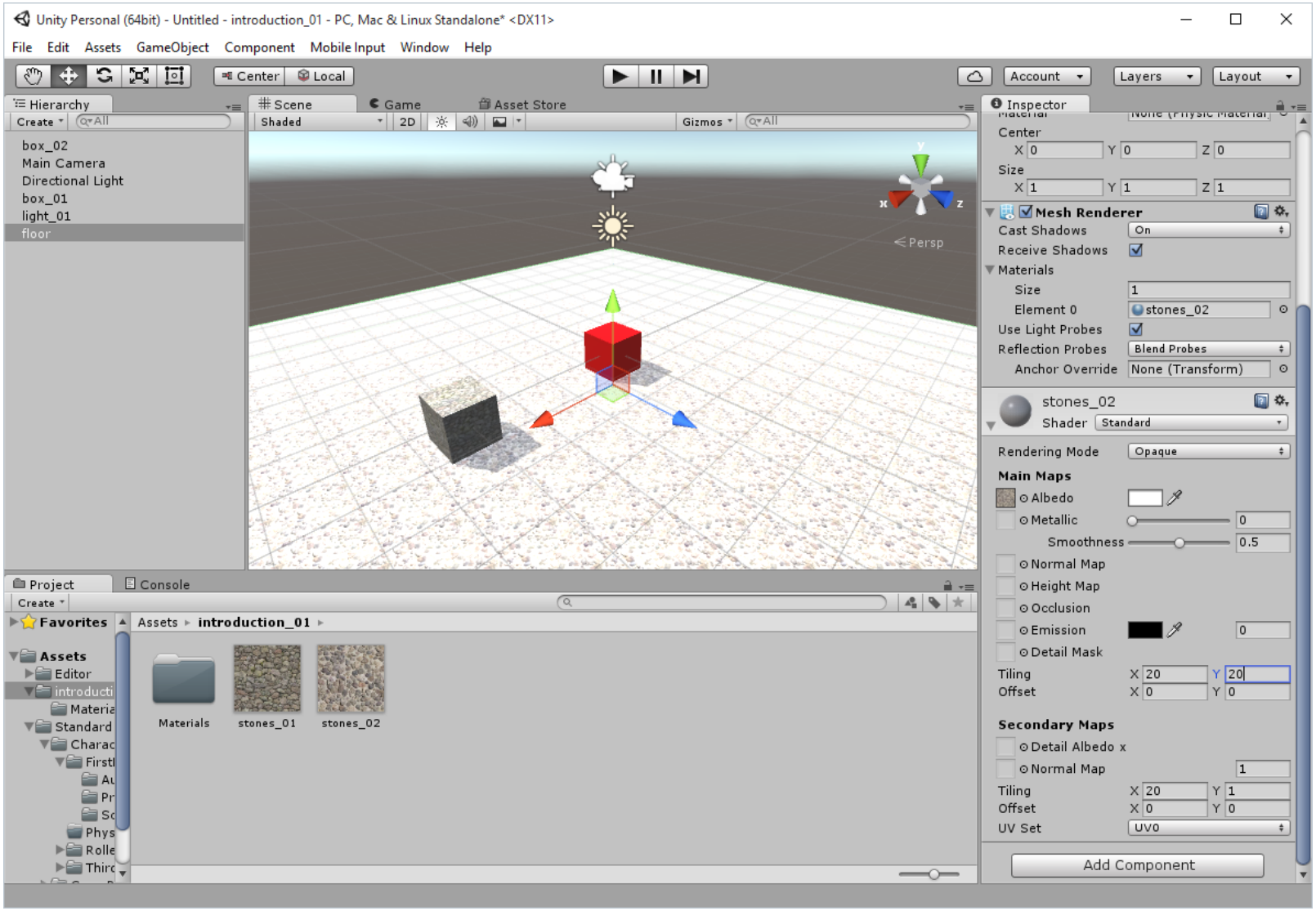


The ground – tuning

If you want, you can change the tiling properties of the texture used for the floor:

- Click once on the object floor in the **Hierarchy** window.
- In the **Inspector** window, click once on the component labeled as your texture. This should display more properties for this texture.
- Change the tiling properties to (x=20, y=20).

The ground – tuning



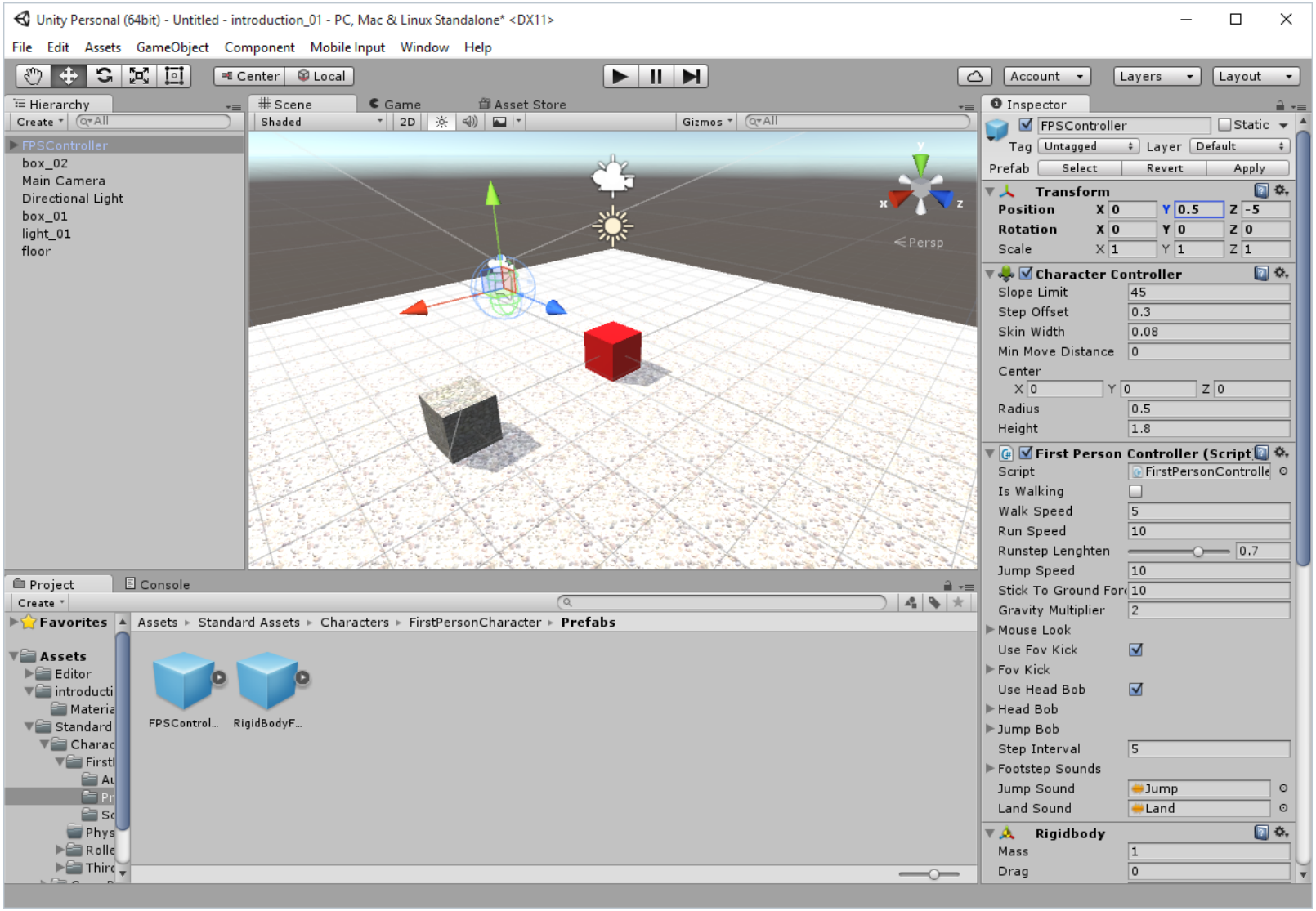
Adding a first-person controller

Add First Person Controller to the scene:

- Drag-and-drop the First Person Controller prefab (that is, blue box labeled `FirstPersonCharacter`) by selecting in **Assets** view **Assets** | **Standard Assets** | **Characters** | **FirstPersonCharacter** | **Prefabs** | **FPSController** onto the **Scene** view (or the **Project** view).
- Move this First Person Controller to the position (x=0, y=0.5, z=-5) using the position properties in the **Inspector** window.

Note that this should place First Person Controller slightly above the ground; if it is too low, the collision may not be detected and the controller will fall indefinitely.

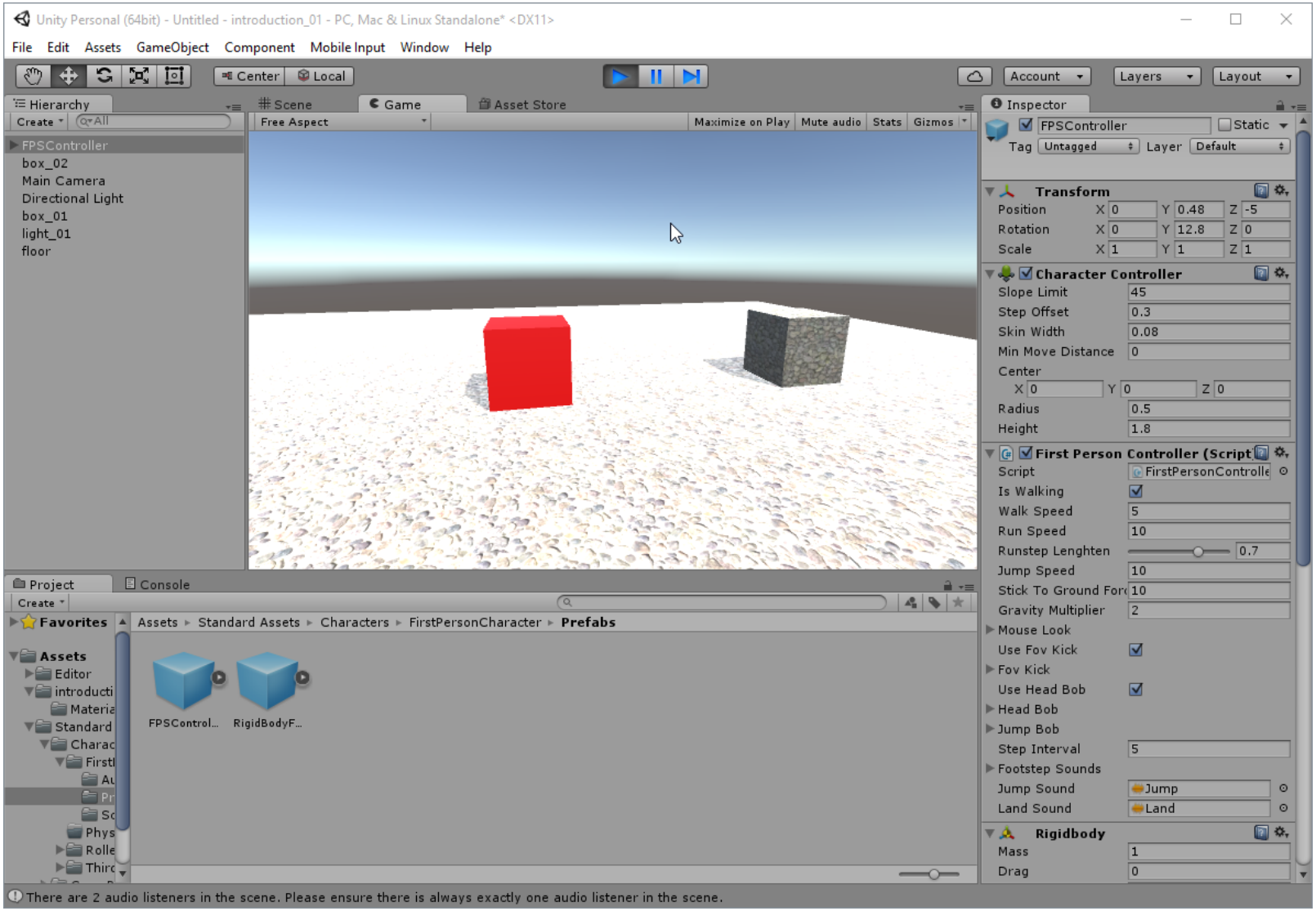
Adding a first-person controller



Test the scene

- Press the **Play** button located at the top of the window (or Ctrl + P).
- Navigate through our scene using the keys W, A, S, and D, the arrow keys, or the mouse.
- Quit the Play mode by clicking on the **Play** button or Ctrl + P.

Test the scene



Adding a third-person controller

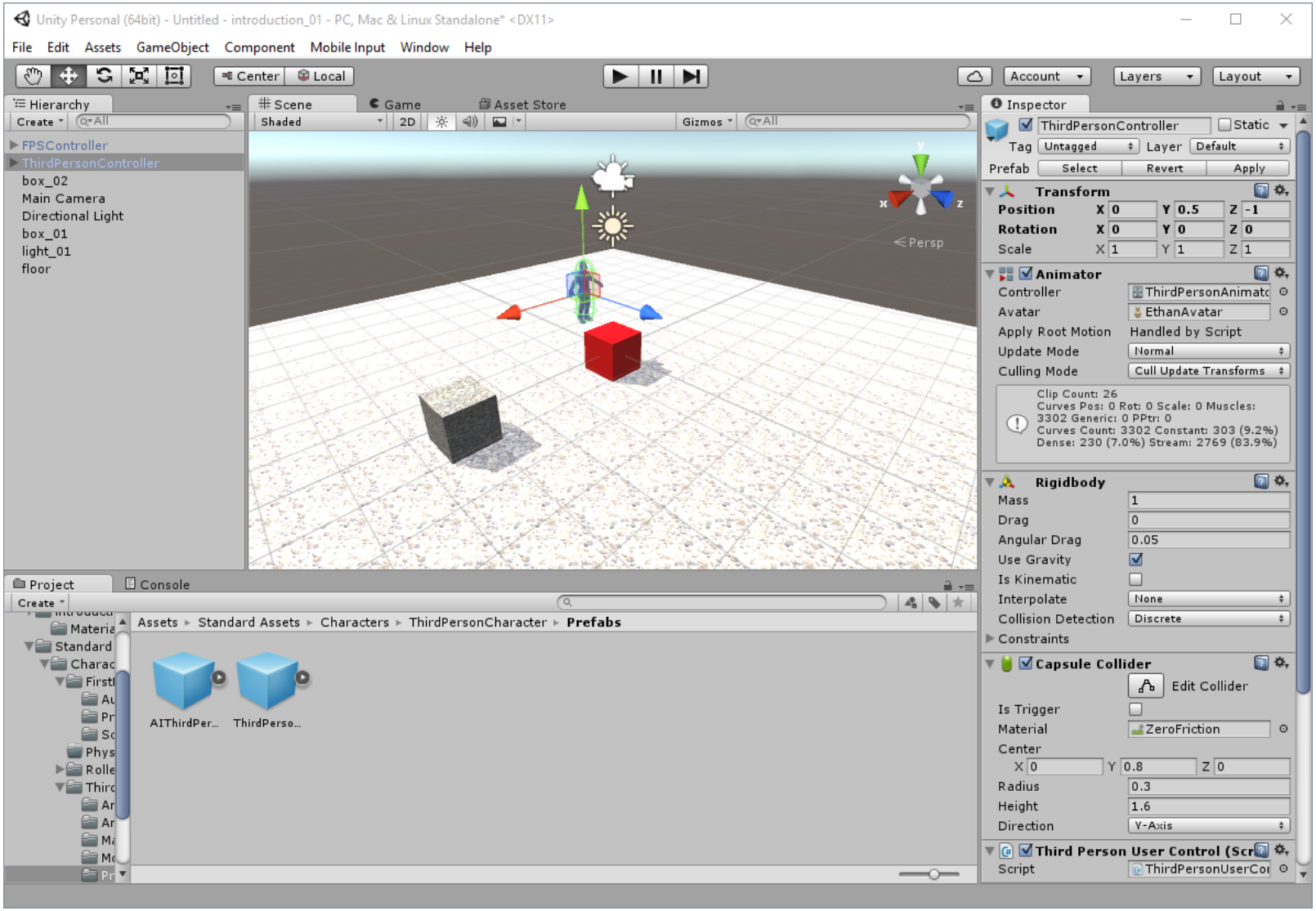
Deactivate the first-person controller:

- The scene already includes a First Person Controller. To avoid any conflict between these, it is better to deactivate the First Person Controller.
- Select First Person Controller in our scene, and uncheck the box to the left of the label First Person Controller (`FPSController`) in the **Inspector** window.
- This will deactivate this object, which means that it will still be present in the project, but not used or seen when the scene is played until it is reactivated.

Add the third-person controller

- Drag-and-drop the Third Person Controller prefab (that is, blue box labeled `ThirdPersonCharacter`) by selecting in **Assets** view **Assets** | **Standard Assets** | **Characters** | **ThirdPersonCharacter** | **Prefabs** | **ThirdPersonController** onto the **Scene** view (or the **Project** view).
- Move this Third Person Controller to the position (x=0, y=0.5, z=-1). This should place the controller slightly above the ground.

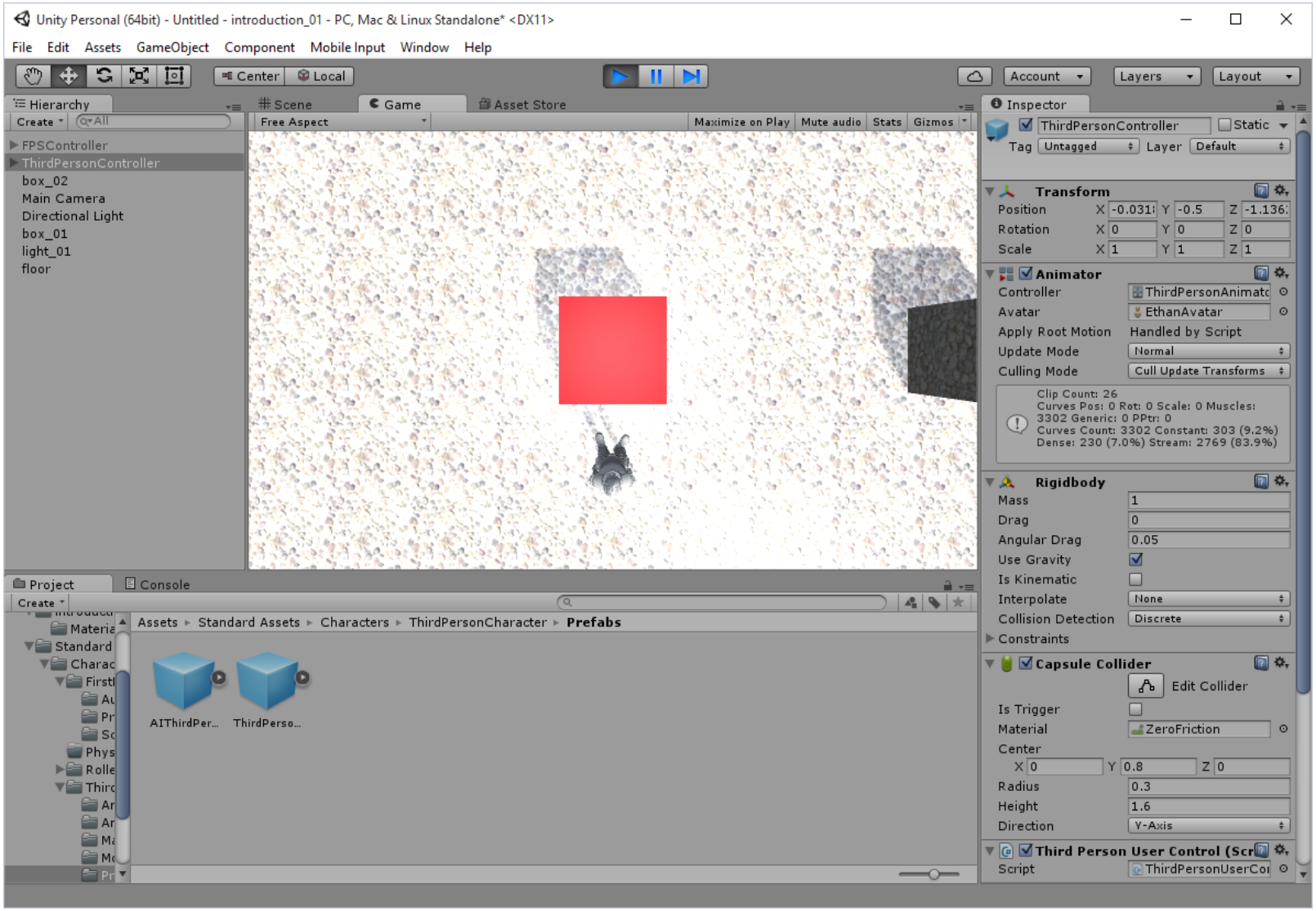
Add the third-person controller



Test the scene

- Press the **Play** button located at the top of the window or (Ctrl + P).
- Navigate through our scene using the keys W, A, S, D, or the arrow keys.
- Quit the Play mode by clicking on the **Play** button or use the corresponding shortcut (Ctrl + P).
- Save our scene (**File** | **Save Scene As**) and name it `intro_01`.

Test the scene



Summary

You should know

- how the user interface looks like,
- useful shortcuts,
- how to create and manipulate built-in objects,
- how to apply colors and textures,
- how to add both first- and third-person controllers for the player to navigate through the scene.