SPACE SHOOTER IN UNITY 5
01 GAME SETUP, PLAYER AND CAMERA

This upgrade guide will help you follow the Space Shooter tutorial series using Unity 5.

The official video tu orials can be found on our Learn page here:
http://unity3d.com/learn/tutorials/projects/space-shooter-tutorial

The official Space Shoo er assets can be found here:
https://www.assetstore.unity3d.com/en/content/13866

For any problems, issues, questions or comments, please use the official Unity Com unity forum thread:
http://forum.unity3d.com/threads/space-shooter-tutorial-q-a.313899

The Space Shooter project was originally recorded under an early version Unity 4. There have been a number of changes to Unity since the original recordings were made. These include an entire new way to create in game user interfaces, an optimization of the underlying physics system, increased modularization of the editor and more. The impact of these changes on the Space Shooter project are, luckily, minimal. This document will go into the details of what has changed, and what changes need to be made in a new project following the existing videos. The upgrade guide is arranged by episode, and then by the time-stamp on the video where the upgrade is required.

Please turn "Annotations" ON when watching these video tutorials, as the annotations will mention these upgrade issues in the video.

Let's get started!

SPACE SHOOTER IN UNITY 5
01 GAME SETUP, PLAYER AND CAMERA

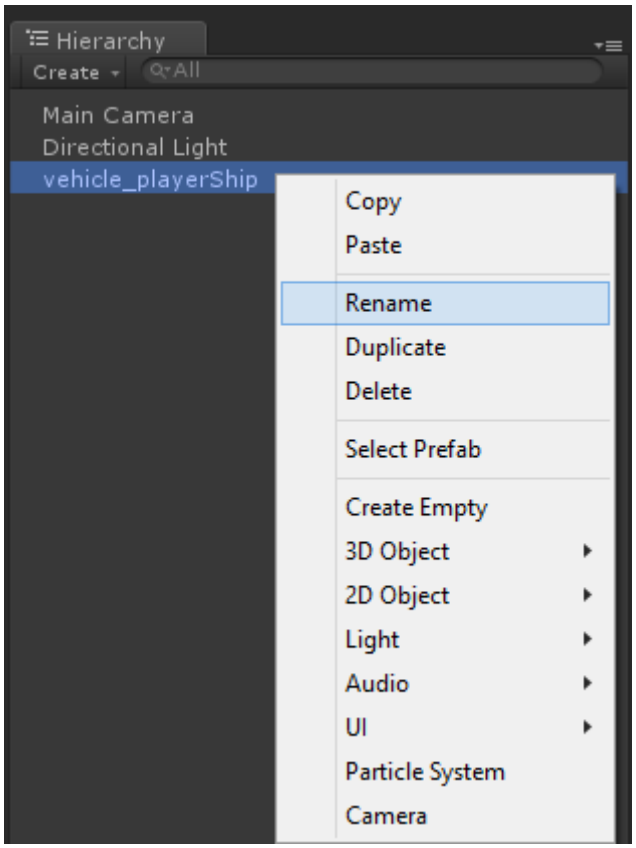
01. SETTING UP THE PROJECT

No known issues.

02. THE PLAYER GAMEOBJECT

01:05 Renaming the Player GameObject

To rename the Player GameObject, the lesson mentions using the <return> or <enter> key to enable editing. This only works on Mac OS and is not true for Windows. For both Mac OS and Windows, clicking on the GameObject twice - slowly, will enable editing. It is also possible, on both operating systems, to Right-Click the GameObject to bring up a context sensitive menu that includes "Rename". This will also enable editing.



06:05 A NOTE ON COMPOUND COLLIDERS

It is worth noting that in Unity 5, due to a change in the underlying PhysX physics engine, compound colliders will now return multiple messages if multiple colliders are hit during the same frame.
For more information, please see the documentation.

02. THE PLAYER GAMEOBJECT (continued)

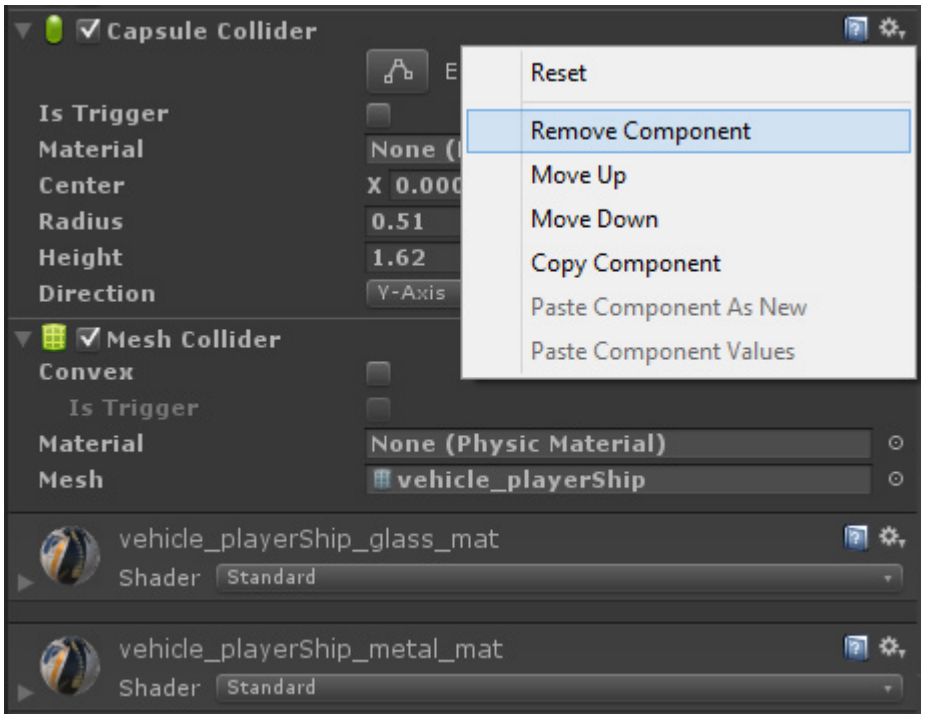
06:20 Replacing the Capsule collider with the Mesh collider

Automatic Collider replacement has been removed from Unity 5. There is no longer a dialogue box asking to replace the existing Collider component with the new Collider being selected from the Add Component menu.

This message is obsolete in Unity 5.

The new Mesh Collider will not replace the existing Capsule Collider automatically. The new Mesh Collider will be added to the Player GameObject.

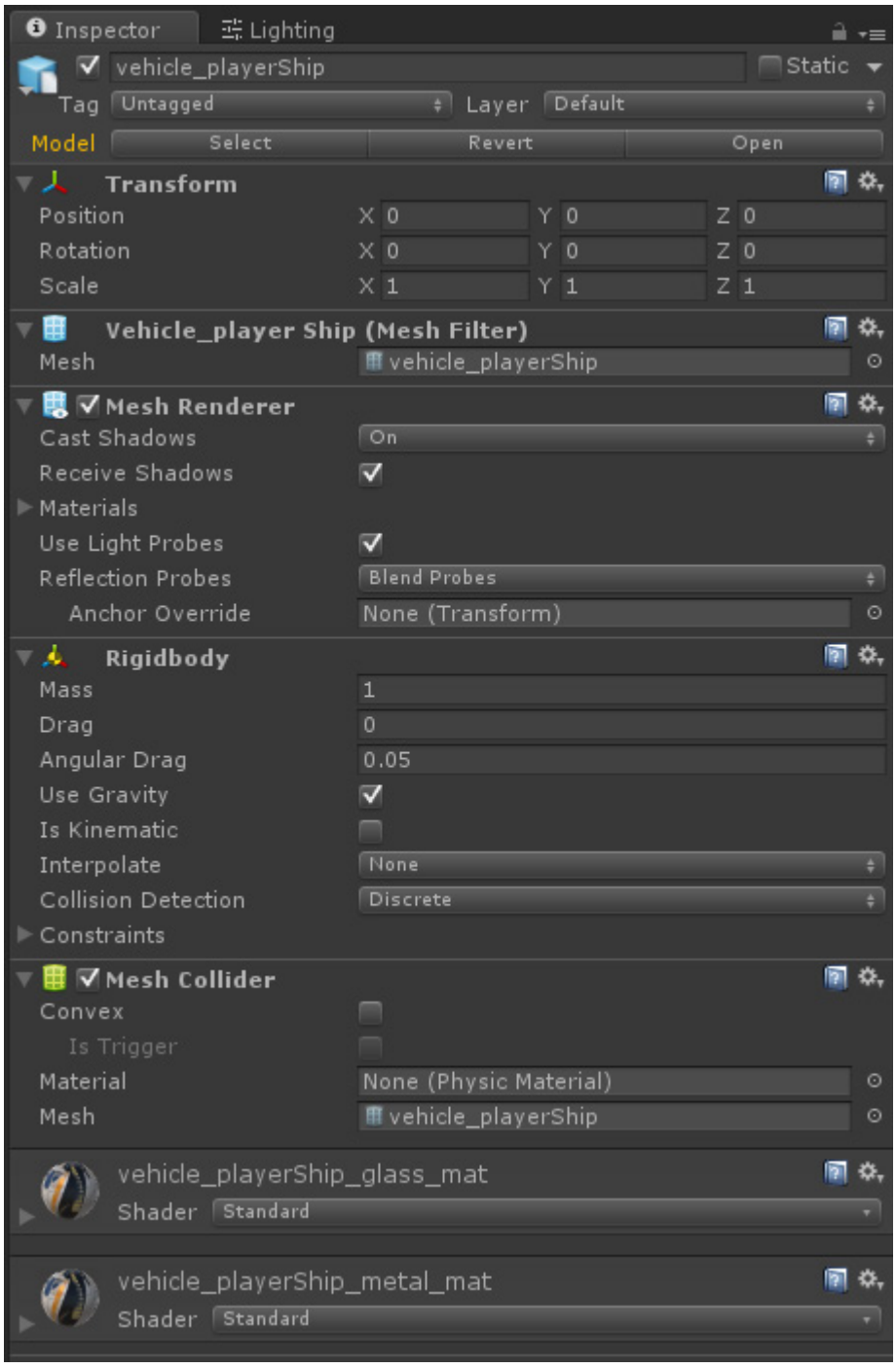
Once the Mesh Collider component has been added to the Player GameObject, the existing Capsule Collider must be removed using the context sensitive "gear" menu in the upper right of the Capsule Collider component.



02. THE PLAYER GAMEOBJECT (continued)

06:20 Replacing the Capsule collider with the Mesh collider (continued)

The Player GameObject should now have the Mesh Collider Only and no Capsule Collider.



02. THE PLAYER GAMEOBJECT (continued)

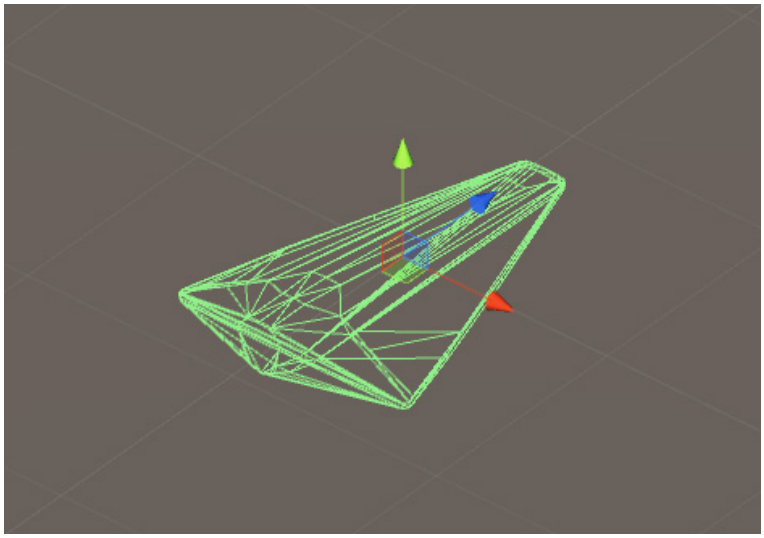
06:35 Using and Visualizing the Mesh Collider

The Mesh Collider component will not participate properly in physics collisions and will not be visible in the scene view unless we select "Convex" on the Mesh Collider Component.

In Unity 5, the Mesh Collider component needs to be Convex to be able to participate properly in physics collisions.

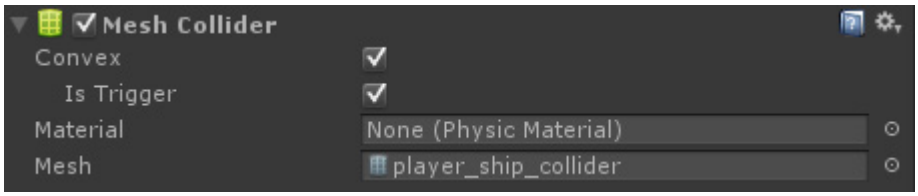
When "Convex" is ***not*** selected, the Mesh Collider will not participate properly in physics collisions and will not be visible in the scene view.

Please note: The "Convex Mesh Collider" will look differently in the scene compared to the video presentation. It should look similar to this:



08:05 Setting the Mesh Collider Component options

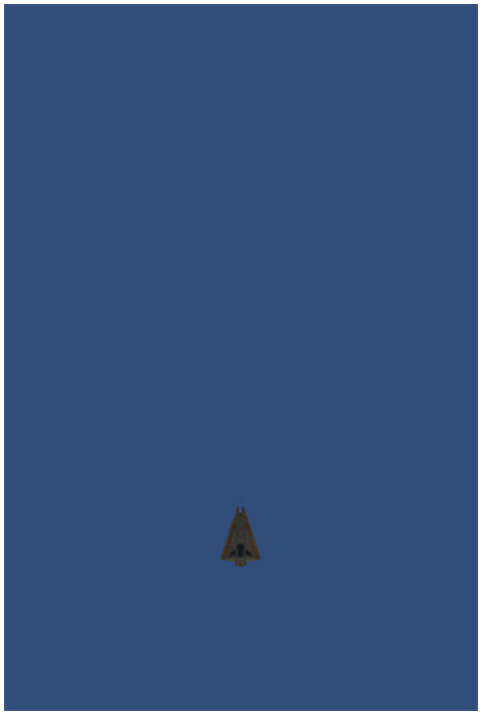
As well as setting "Is Trigger" to true, we must also make sure (as mentioned in the step above) that the "Convex" value is selected as well. The final version of the Mesh Collider component should look like this:



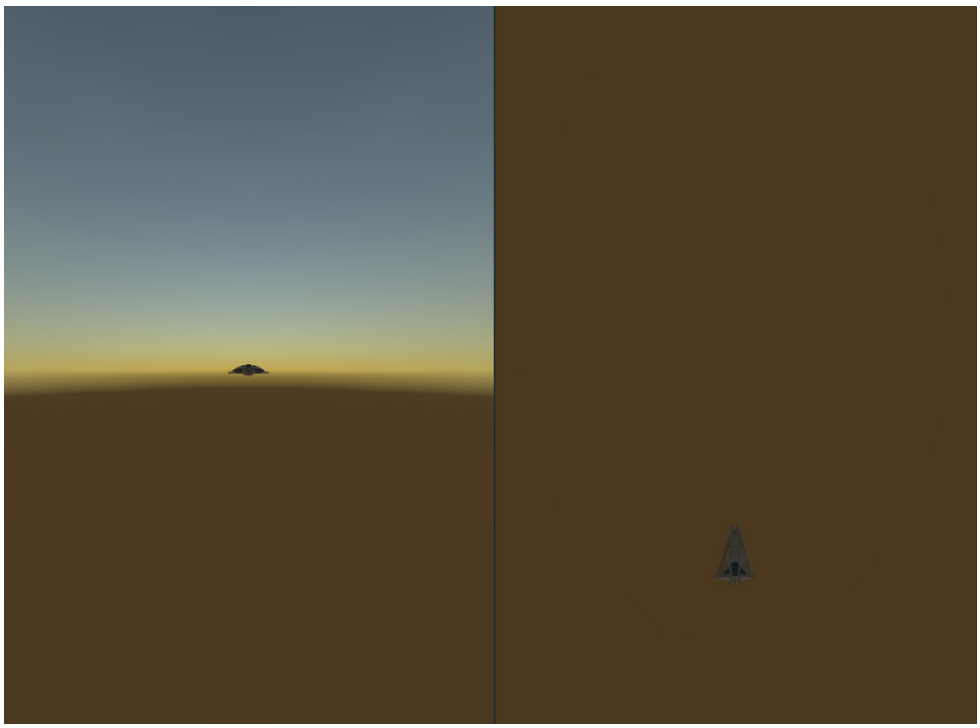
03. CAMERA AND LIGHTING

05:03 The Background, The Skybox and Solid Color

In the video lesson, the background is a uniform blue.



However, in Unity 5, depending upon the direction of the Main Camera's rotation, the background will either be a procedural horizon, or brown:



SPACE SHOOTER IN UNITY 5
01 GAME SETUP, PLAYER AND CAMERA

03. CAMERA AND LIGHTING (continued)

05:03 The Background, The Skybox and Solid Color (continued)

This is because, by default in Unity 5, a Skybox is included in the scene. This skybox will influence the ambient light affecting the ship and will be used as the background image by the Main Camera.

In the video lesson the background is blue because there is no skybox being used, and the Camera is falling-back on the default solid color in the absence of a skybox.

05:48 The Default Directional Light

Unity creates a default "Directional Light" in every scene.

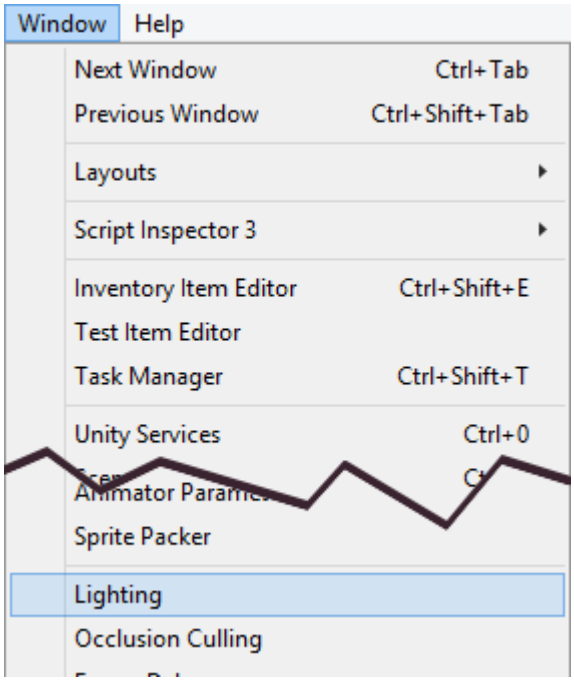
Later in this lesson, new lights will be created as needed.

Please select the default "Directional Light" GameObject and delete it.

06:00 Ambient Light

Lighting has signifi antly changed in Unity 5. This includes how Ambient light works and where to find the settings.

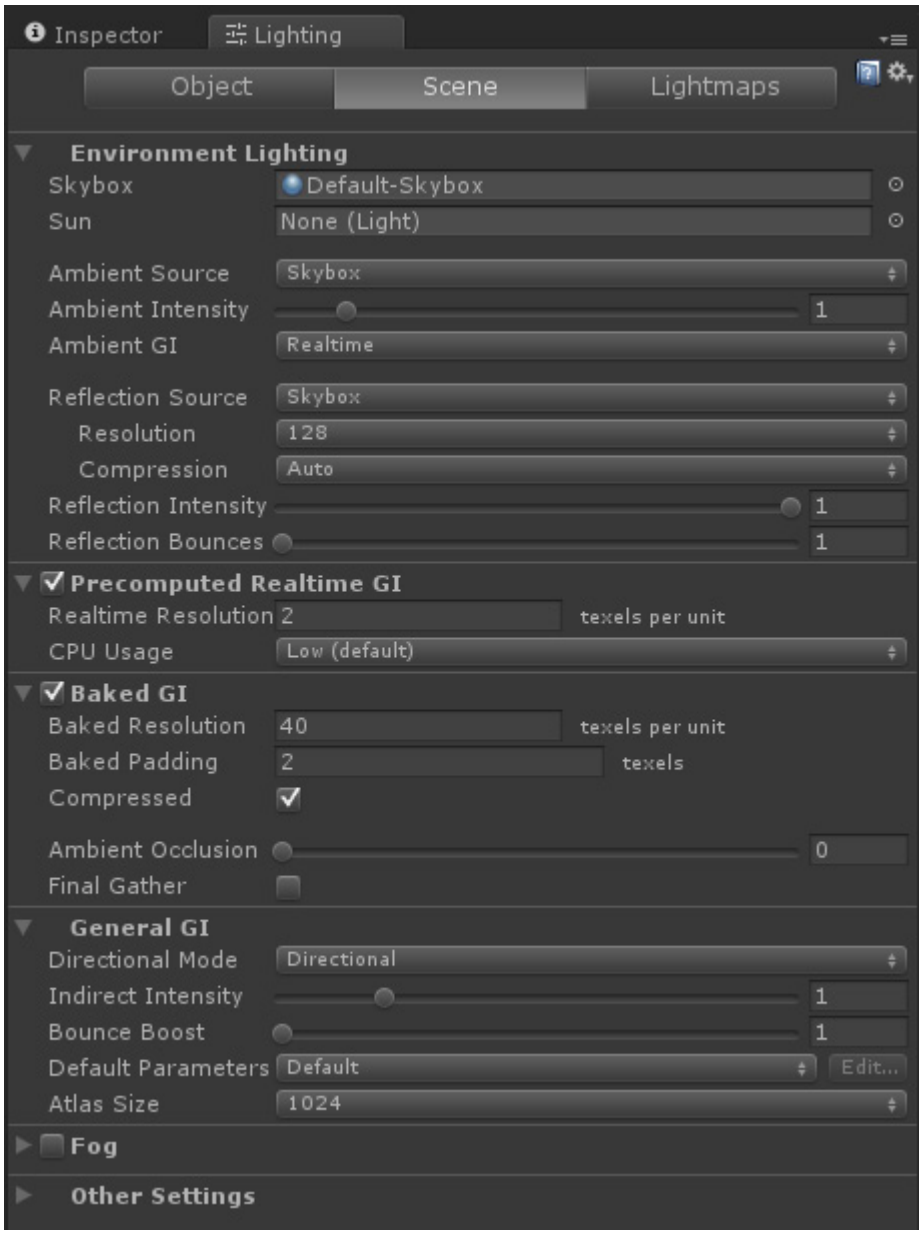
To find the settings for lighting and the related scene settings, open the Lighting Panel. This panel can be found in Window/Lighting. Once the Lighting Panel is open, select the Scene tab.



03. CAMERA AND LIGHTING (continued)

06:00 Ambient Light (continued)

Please note that it is possible to dock this windows in the editor, and it may be wise to do so now.



The current Lighting Panel in Unity 5

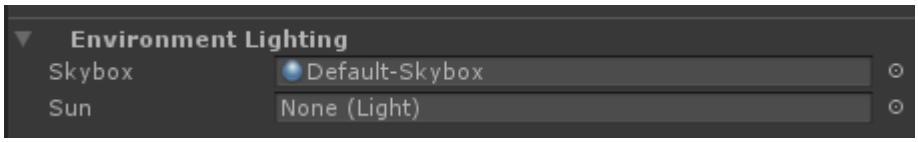
03. CAMERA AND LIGHTING (continued)

06:00 Ambient Light (continued)

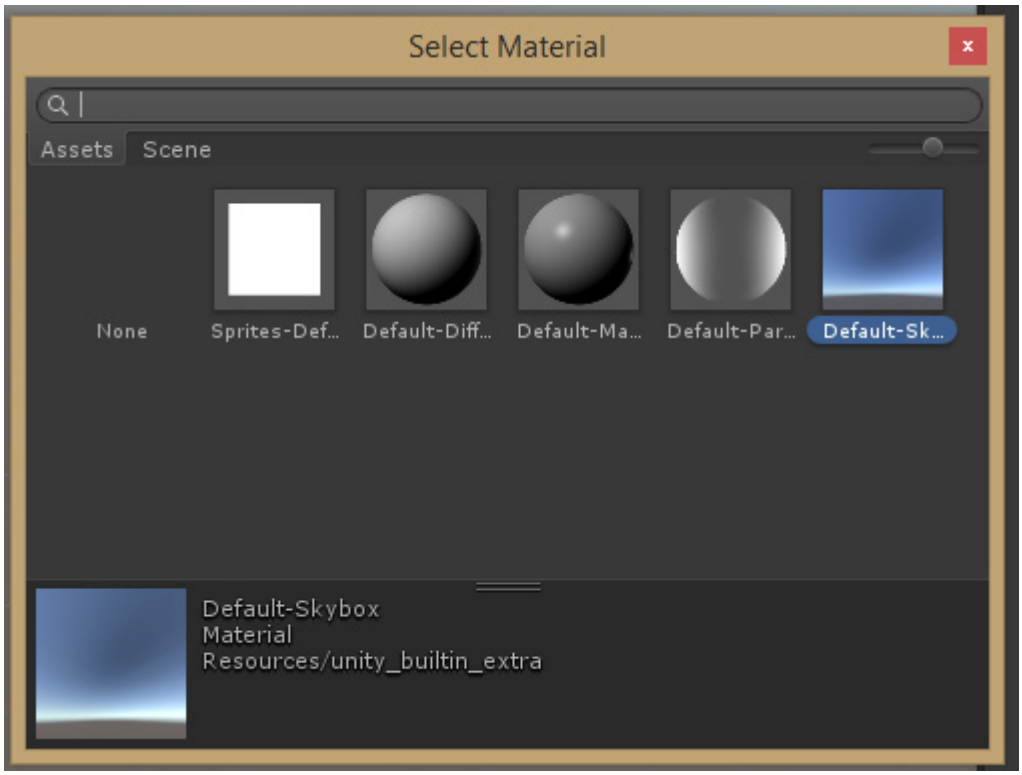
As indicated in the video lesson, Ambient Light can be a constant color, but by default, Ambient Light uses the Skybox to set it's color values:

To have no contribution to the lighting from Ambient Light, the Ambient Light source must have no value. This can be done by either leaving the Ambient Source as Skybox and making sure Skybox is None, or by setting the Ambient Source to Color and making sure the color is Black.

To remove the Skybox, simply delete it. The Skybox field is a asset field like all of the other asset fields in the inspector.



You can either select the target button to the right (the circle with the dot in the middle) and it will open up an asset picker window:

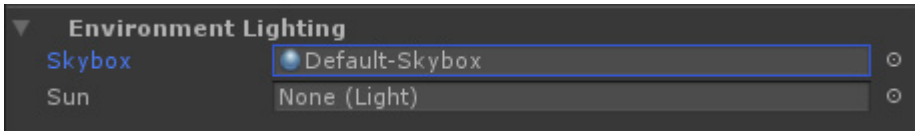


Here you can choose "None", or ...

03. CAMERA AND LIGHTING (continued)

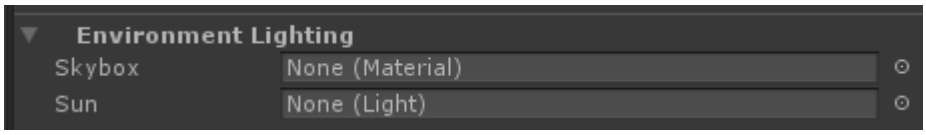
06:00 Ambient Light (continued)

Simply select the skybox field:



... and use the backspace or delete key to remove it.

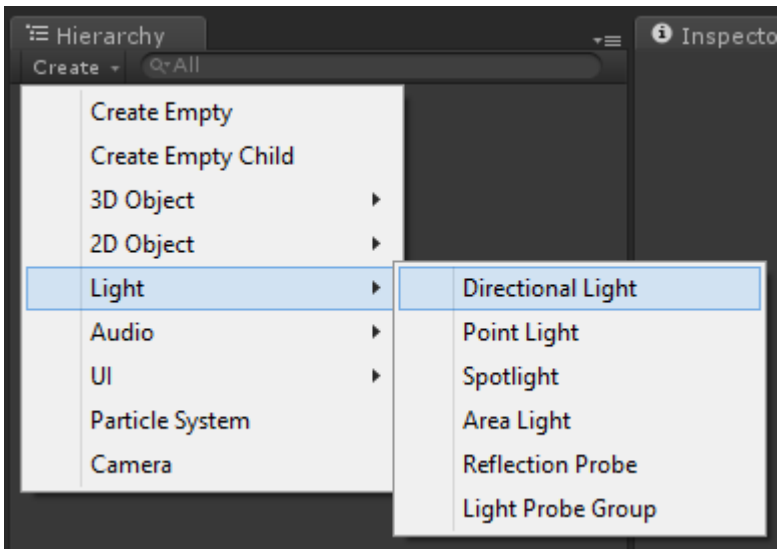
In the end it should look like this:



07:03 Creating a Directional Light

Please make sure the default Directional Light was removed as mentioned above before continuing on.

To create a new Directional Light, use the menu option found under: Create/Light/Directional Light.



SPACE SHOOTER IN UNITY 5
01 GAME SETUP, PLAYER AND CAMERA

03. CAMERA AND LIGHTING (continued)

08:30 Adjusting the Main Light's Intensity

One of the many changes to the Lighting System in Unity 5 is how basic materials respond to lights. Shaders no longer apply a 2x multiply of light intensity:
http://docs.unity3d.com/Manual/UpgradeGuide5-Shaders.html

This means "in general" lights are half as strong as they used to be in Unity 4 and earlier. In general, the Light's Intensity value should be doubled now compared to the values in the video lesson.

In this particular case, it was felt that a value of 2.0 was correct, which is more than twice the existing value. You should balance these lights to your particular taste. If you like your choice of lighting, then it's perfect. It doesn't affect anything later on in this lesson in any technical way.

10:15 Adjusting the Fill Light's Intensity

A suggested value of 1.0 felt correct here. This is twice the value in the video lesson.

12:10 Adjusting the Rim Light's Intensity

A suggested value of 0.5 felt correct here. This is twice the value in the video lesson.

This makes the overall suggested values for the lights:

Main: 2.0
Fill: 1.0
Rim: 0.5

12:34 Hierarchy Sort Order

Please Note: The Hierarchy no longer sorts alphabetically by default, but by Transform Order. By default, the hierarchy order can be reordered at will. Be aware that the hierarchy order can influence the project's behaviour. For example, the render order of UI Elements is influenced by transform order. Please see the documentation for more information:
http://docs.unity3d.com/Manual/Hierarchy.html

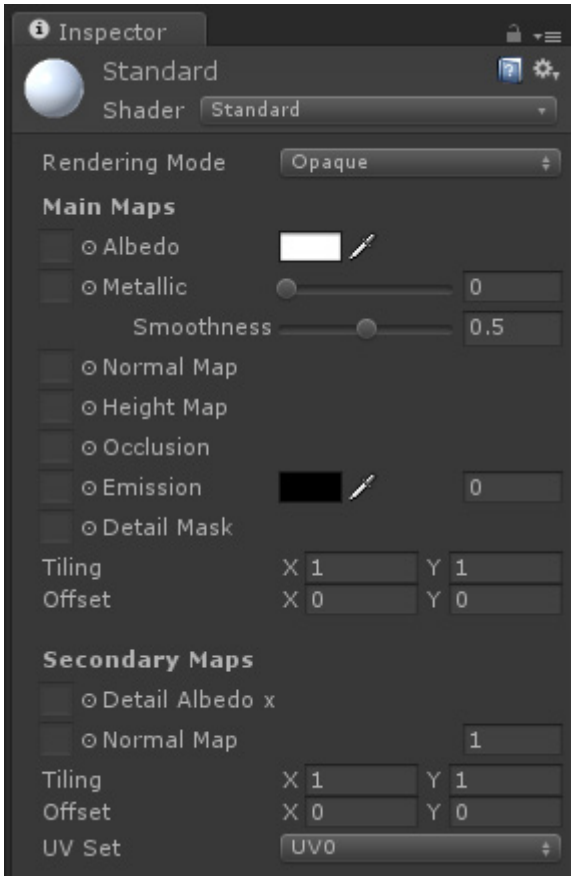
04. CAMERA AND LIGHTING

00:24 Creating a "Quad"

Please Note: This menu item has moved. It can now be found by navigating to "Create/3D object/Quad" the hierarchy's create menu.

04:00 Creating new Materials

Materials are created using the Standard Shader. This new material will be created with a default "opaque" setup very similar to the former "default diffuse".



For more information please see:
• http://docs.unity3d.com/Manual/Shaders.html
• http://unity3d.com/learn/tutorials/modules/beginner/unity-5/unity5-lighting-overview
• http://unity3d.com/learn/tutorials/modules/beginner/graphics/lighting-and-rendering

04. CAMERA AND LIGHTING (continued)

06:04 The Standard Shader

This Material is using the Standard Shader. The Standard Shader contains a number of detailed settings for this material, allowing for a fewer number of different shaders to create materials with.

The Standard Shader is a physicslly based shader (http://docs.unity3d.com/Manual/shader-StandardShader.html). This allows for a more friendly way of achieving a consistent look under different lighting conditions.

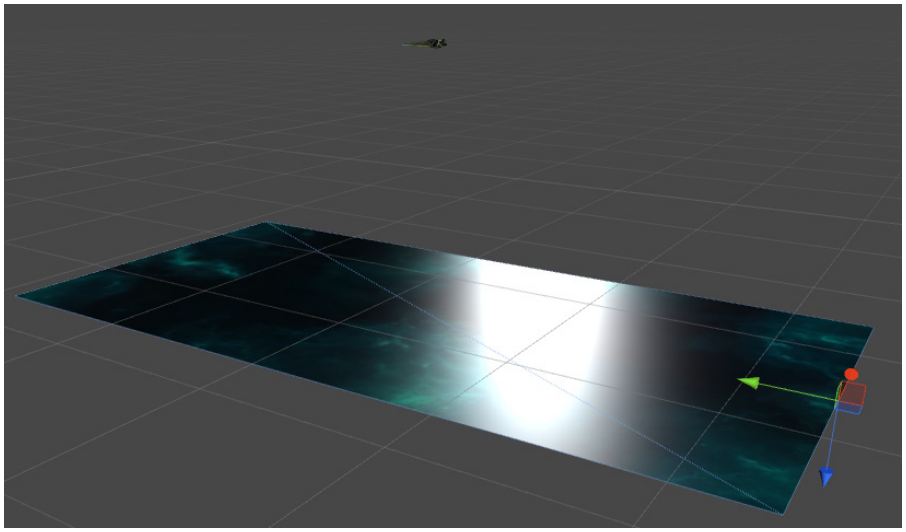
With the current default settings, the material applied to the "Background Quad" is acting a little like a "simple painted background". However, with a Smoothness of 0.5, it may be less matte than the "default diffuse" material shown in the video.

07:30 Changing Shaders on a Material

This Material is using the Standard Shader set to default opaque.

This is very similar to the Default Diffuse shader shown in the video. The settings on the Standard Shader make this Material behave in a way very much like a surface painted with Matte Paint. Due to the Smoothness setting of 0.5, it may behave more like a "semi-gloss" paint. To make the surface behave like Glossy Paint, there is no need to change the shader - simply change the Smoothness setting on the Standard Shader.

Try this: Angle the camera so that one of the directional lights is reflecting off of the surface of the Background and change the smoothness from 0 to 1 and back again while observing how the surface of the quad reacts.



04. CAMERA AND LIGHTING (continued)

07:30 Changing Shaders on a Material (continued)

For more information please see the documentation on the Standard Shader and details on the Smoothness property: http://docs.unity3d.com/Manual/StandardShaderMaterialParameterSmoothness.html.

05. MOVING THE PLAYER

00:00 Changes to Scripting in Unity 5

Unity no longer uses "Helper References" to access common components.

In Unity 5 and later we can no longer access components using their "shorthand helper references" and we must access them directly using "GetComponent".

One example of this is accessing the Rigidbody component attached to the same GameObject as the script. In Unity 4 and earlier, this was simply accessed with "rigidbody."

Now this must be done with "GetComponent<Rigidbody>()."

It is usually a "best practice" to find this Component when the instance of the script initializes, and "cache" the reference in a local variable.

This is commonly written as:

```
private Rigidbody;

void Start ()
{
      rb = GetComponent<Rigidbody>();
}
```

Now, with the reference to the local Rigidbody component saved in the variable "rb", we can use this reference anywhere within the script.

One example would be to add force to the rigidbody with:

```
rb.AddForce (someVector3Value);
```

SPACE SHOOTER IN UNITY 5
01 GAME SETUP, PLAYER AND CAMERA

05. MOVING THE PLAYER (continued)

01:55 Clarifying terminology

The voice over in the lesson mentions "CamelCase". For the sake of accuracy, in computing terms, this should have been"Pascal Case". For more information of this terminology, please see the link below:
https://en.wikipedia.org/wiki/CamelCase

04:35 Updating the code to avoid obsolete references

Please Note: In Unity 5 and later we can no longer access components using their "shorthand helper references" such as "rigidbody." and we must access them directly using "GetComponent".

To access the local Rigidbody component, please use "GetComponent<Rigidbody>()." It is best to "cache" this reference in a local variable when the instance of the script is initialized.

Please declare a local member variable to hold the Rigidbody reference by writing at the top of the class:

```
private Rigidbody rb;
```

… and in void Start() write:

```
rb = GetComponent<Rigidbody>();
```

The top of the script should now read:

```
public class PlayerController : Monobehaviour
{
      private Rigidbody rb;

      void Start ()
      {
            rb = GetComponent<Rigidbody>();
      }
```

From here on, while writing code addressing the Rigidbody component, ***do not use*** *rigidbody*. but ***please use*** *rb*, as *rb* now contains a reference to the local Rigidbody component attached to the Player GameObject.

The line being written in the video should now read as:

```
rb.velocity =  some Vector3 value;
```

SPACE SHOOTER IN UNITY 5
01 GAME SETUP, PLAYER AND CAMERA

05. MOVING THE PLAYER (continued)

09:50 Updating the code to avoid obsolete references

Please Note: In Unity 5 and later we can no longer access components using their "shorthand helper references" such as "rigidbody." and we must access them directly using "GetComponent".

To access the local Rigidbody component, please use "GetComponent<Rigidbody>()." It is best to "cache" this reference in a local variable when the instance of the script is initialized.

Assuming that the reference to the local Rigidbody component was cached as per the previous upgrade note (see Ep 5, 4:35), the line being written in the video should now read as:

```
rb.position = new Vector3 (x, y, z);
```

18:20 Updating the code to avoid obsolete references

Please Note: In Unity 5 and later we can no longer access components using their "shorthand helper references" such as "rigidbody." and we must access them directly using "GetComponent".

To access the local Rigidbody component, please use "GetComponent<Rigidbody>()." It is best to "cache" this reference in a local variable when the instance of the script is initialized.

Assuming that the reference to the local Rigidbody component was cached as per the previous upgrade note (see Ep 5, 4:35), the line being written in the video should now read as:

```
rb.rotation = Quaternion.Euler (x, y, z);
```

06. CREATING SHOTS

01:00 Creating a "Quad"

Please Note: This menu item has moved. It can now be found by navigating to "Create/3D object/Quad" the hierarchy's create menu.

SPACE SHOOTER IN UNITY 5
01 GAME SETUP, PLAYER AND CAMERA

06. CREATING SHOTS (continued)

03:20 The "Default Diffuse" shader has been replaced

For the purposes of this lesson, when using the Standard Shader, the "Main Texture" property of the Default Diffuse shader has been replaced by the "Albedo" property. Please follow the instructions in the video, but use the "Albedo" field rather than the "Main Texture" field.

For more detailed information, please see the documentation on the Standard Shader: http://docs.unity3d.com/Manual/shader-StandardShader.html.

04:10 Using the "Albedo" field ins ead of the "Main Texture" fiel

As we are using the Standard Shader, please drop the Texture into the "Albedo" field on the material.

07:40 Viewing the Quad's collider in the Scene View

Please Note: The collider will only be visible in the scene if "convex" is selected on the collider component.

11:00 Updating the code to avoid obsolete references

Please Note: In Unity 5 and later we can no longer access components using their "shorthand helper references" such as "rigidbody." and we must access them directly using "GetComponent".

To access the local Rigidbody component, please use "GetComponent<Rigidbody>()." It is best to "cache" this reference in a local variable when the instance of the script is initialized.

Please declare a local member variable to hold the Rigidbody reference by writing at the top of the class:

```
private Rigidbody rb;
```

… and in void Start() write:

```
rb = GetComponent<Rigidbody>();
```

From here on, while writing code addressing the Rigidbody component, ***do not use*** *rigidbody*. but ***please use*** *rb*, as *rb* now contains a reference to the local Rigidbody component attached to the Player GameObject.

06. CREATING SHOTS (continued)

11:00 Updating the code to avoid obsolete reference (continued)

The line being written in the video should now read as:

```
rb.velocity =
```

The top of the script should now read:

```
public class Mover : Monobehaviour
{
      private Rigidbody rb;

      void Start ()
      {
            rb = GetComponent<Rigidbody>();
            rb.velocity =
      }
```

12:40 Setting the *Speed* value on the Prefab

Please note that we are setting this value on the PREFAB in the Project View, not the INSTANCE in the Scene.

07. SHOOTING SHOTS

No known issues.

SPACE SHOOTER IN UNITY 5
02 BOUNDARIES, HAZARDS AND ENEMIES

01. BOUNDARY

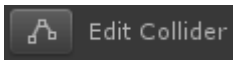
00:52 Creating a "Cube"

Please Note: This menu item has moved. It can now be found by navigating to "Create/3D object/Cube" the hierarchy's create menu.

02. CREATING HAZARDS

02:33 Editing the Collider component's shape in the Scene view

Colliders are now edited by selecting the "Edit Collider" button on the Collider component.



03:45 Update the code to avoid obsolete references

Please Note: In Unity 5 and later we can no longer access components using their "shorthand helper references" such as "rigidbody." and we must access them directly using "GetComponent".

To access the local Rigidbody component, please use "GetComponent<Rigidbody>()." It is best to "cache" this reference in a local variable when the instance of the script is initialized.

Please declare a local member variable to hold the Rigidbody reference by writing at the top of the class:

```
private Rigidbody rb;
```

… and in void Start() write:

```
rb = GetComponent<Rigidbody>();
```

From here on, while writing code addressing the Rigidbody component, ***do not use*** *rigidbody*. but ***please use*** *rb*, as *rb* now contains a reference to the local Rigidbody component attached to the Player GameObject.

The line being written in the video should now read as:

```
rb.angularVelocity =
```

SPACE SHOOTER IN UNITY 5
02 BOUNDARIES, HAZARDS AND ENEMIES

02. CREATING HAZARDS (continued)

03:45 Update the code to avoid obsolete references (continued)

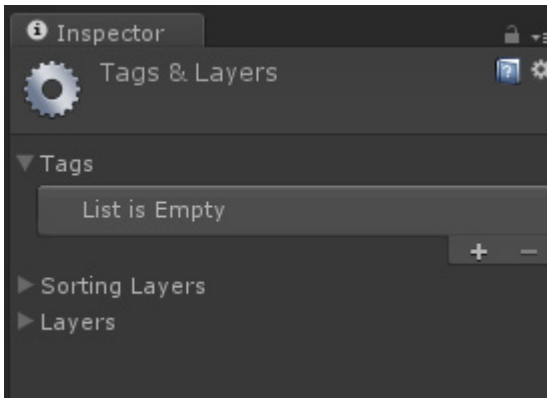
The top of the script should now read:

```
public class RandomRotator : Monobehaviour
{
      private Rigidbody rb;

      void Start ()
      {
            rb = GetComponent<Rigidbody>();
            rb.angularVelocity =
      }
```

11:55 Assigning Tags and Layers

The Tags and Layers panel has been redesigned. To add a new tag, press the '+' button on the tags list:



03. EXPLOSIONS

No known issues.

04. GAME CONTROLLER

No known issues.

05. SPAWNING WAVES

No known issues.

SPACE SHOOTER IN UNITY 5
03 SCORING, FINISHING AND BUILDING THE GAME

01. AUDIO

07:15 Update the code to avoid obsolete references

Please Note: In Unity 5 and later we can no longer access components using their "shorthand helper references" such as "audio." and we must access them directly using "GetComponent".

To access the local AudioSource component, please use "GetComponent<AudioSource>()." It is best to "cache" this reference in a local variable when the instance of the script is initialized.

Please declare a local member variable to hold the AudioSource reference by writing at the top of the class:

```
private AudioSource audioSource ;
```

… and in void Start() write:

```
audioSource = GetComponent<AudioSource >();
```

… then use the line:

```
audioSource.Play();
```

02. COUNTING POINTS AND DISPLAYING SCORE

07:15 Adding a GUIText Component

To Create a GUIText GameObject in 4.6 and later, please do the following:

- Create a new Empty GameObject

- Select the new GameObject

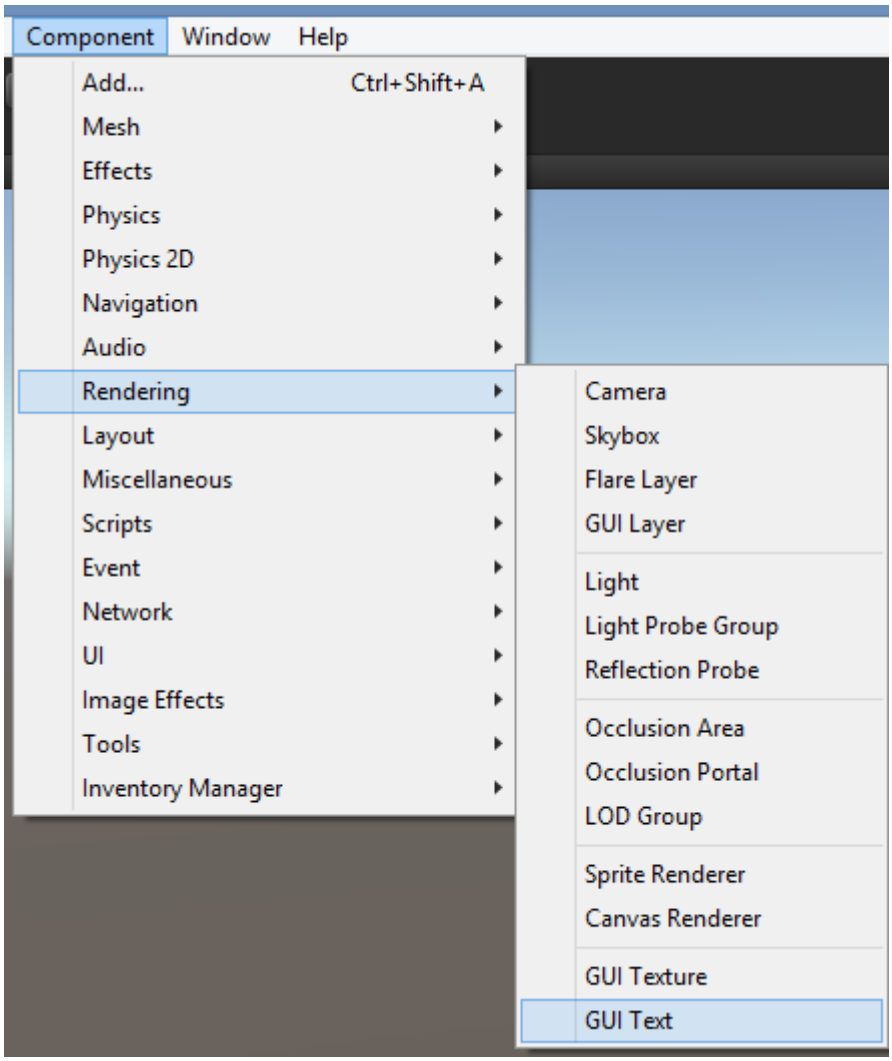
- Add a GUIText component -
  - from the Add Component Button
    or
  - from the Component Menu

    ... found at: Component/Rendering/GUIText

SPACE SHOOTER IN UNITY 5
03 SCORING, FINISHING AND BUILDING THE GAME

02. COUNTING POINTS AND DISPLAYING SCORE (ctd)

07:15 Adding a GUIText Component (continued)



- Update the GUIText's transform position to (0.5, 0.5, 0.0)
  - This will place the GUIText into the middle of the screen.

- Update the GUIText.text with "Score Text".
  - This will make the GUIText object easier to see on screen.

03. ENDING THE GAME

00:55 Adding a GUIText Component

Please see the note above (Sect. 3, Ep. 2, 07:15) for details.

Update the GUIText.text with "Restart Text".

SPACE SHOOTER IN UNITY 5
03 SCORING, FINISHING AND BUILDING THE GAME

03. ENDING THE GAME (continued)

02:51 Adding a GUIText Component

Please see the note above (Sect. 3, Ep. 2, 07:15) for details.

Update the GUIText.text with "Game Over Text".

07:45 Update the code to avoid deprecated API calls

The call to *Application.LoadLevel()* is not longer used and has been deprecated from the Unity API.

To load a new "level" we will need to load a new "scene" from a scene file asset.

At the time of writing, we must use the SceneManager to load and unload scenes. This requires a new namespace and a different function call to the SceneManager.

First, we need to add the new namespace to our *GameController* script. At the top of our script, under the existing namespaces (eg: using UnityEngine;) add a new line that reads:

```
using UnityEngine.SceneManagement;
```

Then, returning to our *Update* function, we need to replace the line *Application.LoadLevel(Application.loadedLevel)* with:

```
SceneManager.LoadScene(SceneManager.GetActiveScene()
.buildIndex);
```

This call asks the SceneManager to load the currently "active" scene. As we have only one scene and this is the only scene loaded in the editor, the SceneManager will find it automatically. For more complex scene management, please see the documentation.

04. BUILDING THE GAME

No known issues.