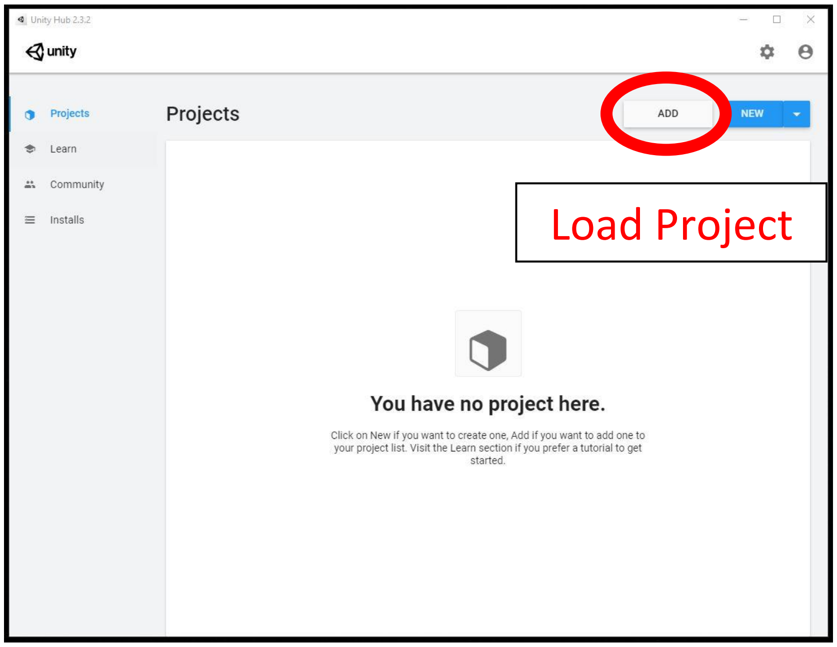
Section 1: Unity Basics

To begin, you will need to download Unity from their website. Next, step-by-step instructions will be provided for creating a basic environment in which a keyboard is used for moving an object. Upon collision with another object, both the User controlled object and the designated Target object change color. Additionally, information will be provided for understanding the Unity interface including the basics about materials, scripts, and scenes. A tutorial environment can be downloaded that has all of the following information included.

1. Download Unity
    a. Link: www.unity.com

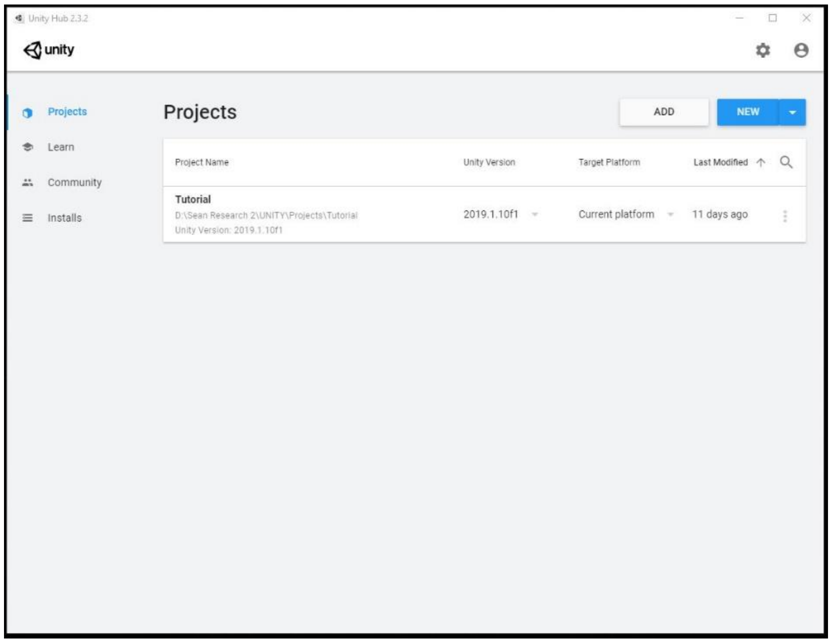
2. Unity Hub
   a. To load a project, select the Add button or press New to create a new project. This is where you will find the Tutorial project if downloaded from the MOCORE website.
   b. **If creating a new project, press NEW and create a 3D Template.**



You have no project here.

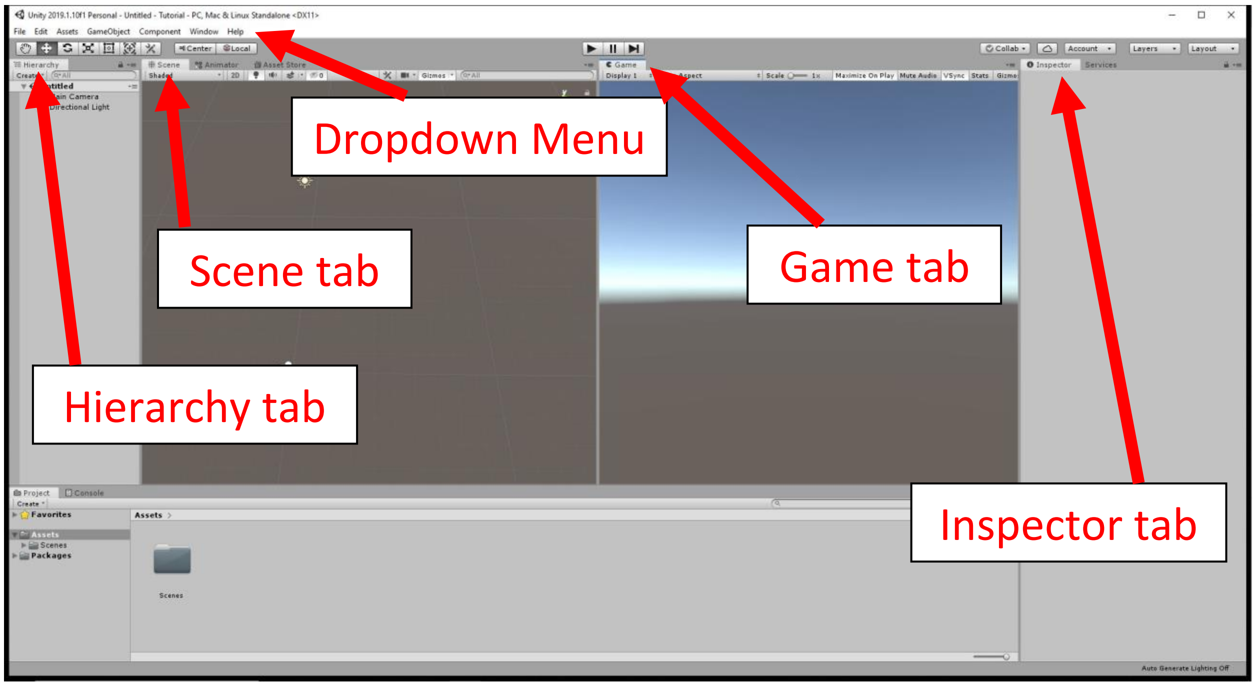
Click on New if you want to create one, Add if you want to add one to your project list. Visit the Learn section if you prefer a tutorial to get started.

   c. Find the Unity Project file folder saved on your local computer. Press Add File on the load screen when the Project file folder is selected.

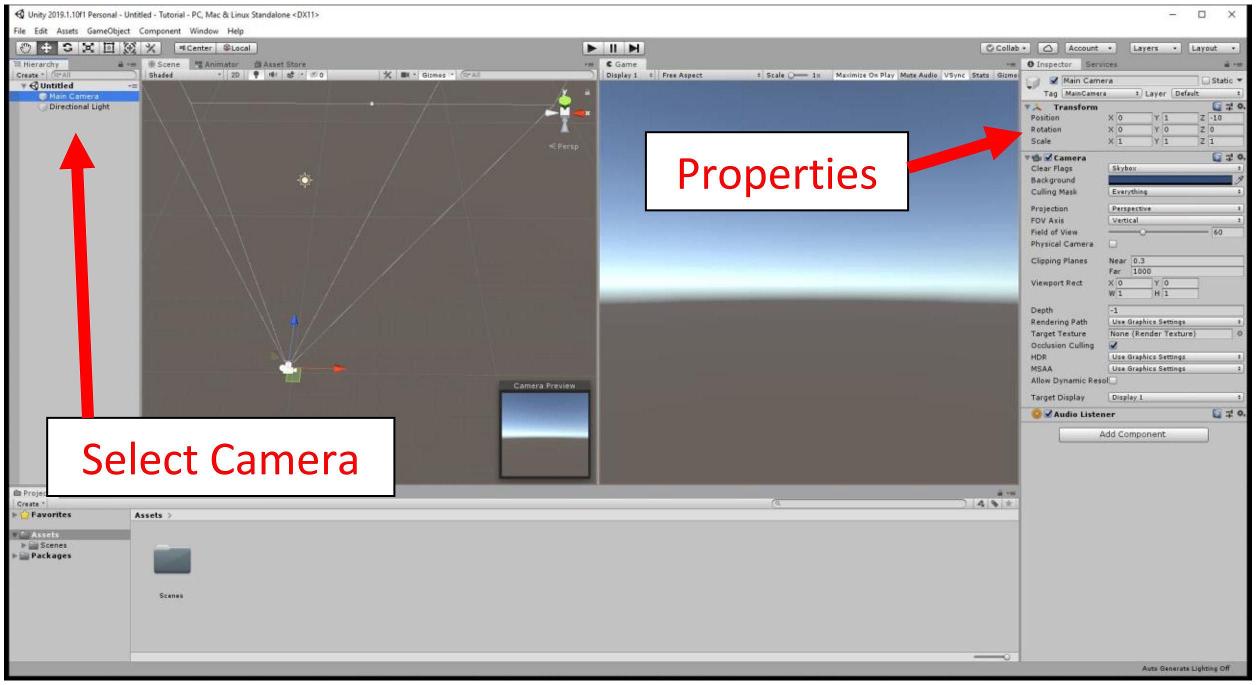


Project Name	Unity Version	Target Platform	Last Modified
Tutorial D:\Sean Research 2\UNITY\Projects\Tutorial Unity Version: 2019.1.10f1	2019.1.10f1	Current platform	11 days ago

3. Main User Interface
    a. On the Main Screen you will see the basic dropdown menus at the Top, a Hierarchy tab on the left side where each object is identified, the Scene and Game windows split in the middle (which can also be overlaid if the Game window is dragged on top the Scene window, noted below), and the Inspector/Services tab on the right side for specifics about each object highlighted in the Hierarchy tab.



b. When the Main Camera is selected on the Hierarchy tab on the left side, object properties such as position, rotation, and scaling are listed under the Inspector tab.



Main Camera · Tag: MainCamera · Layer: Default

Transform

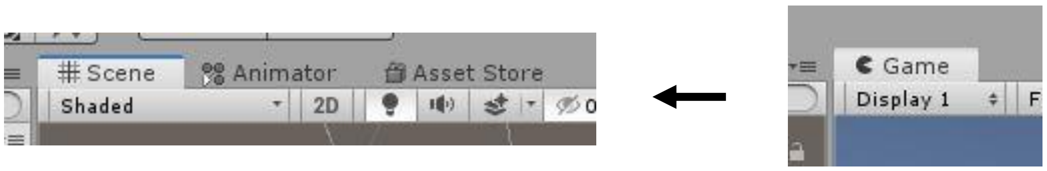
	X	Y	Z
Position	0	1	-10
Rotation	0	0	0
Scale	1	1	1

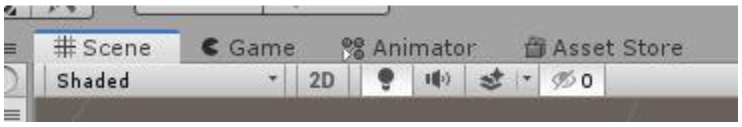
Camera

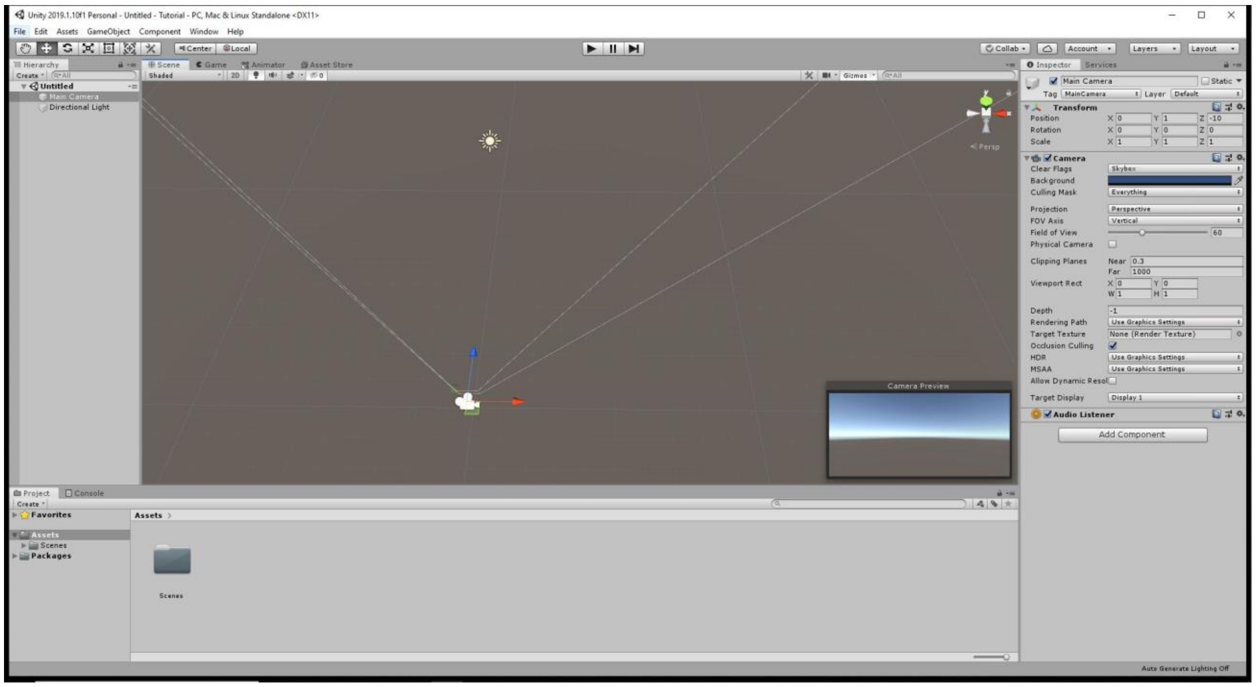
Clear Flags	Skybox
Culling Mask	Everything
Projection	Perspective
FOV Axis	Vertical
Field of View	60
Clipping Planes	Near 0.3, Far 1000
Viewport Rect	X 0, Y 0, W 1, H 1
Depth	-1
Rendering Path	Use Graphics Settings
Target Texture	None (Render Texture)
Occlusion Culling	✓
HDR	Use Graphics Settings
MSAA	Use Graphics Settings
Target Display	Display 1

Audio Listener

c. *Optional:* To save monitor space while working on the environment, the Game window can be overlaid on top of the Scene window by selecting the Icon (see below) and moving the window to the same position on the Scene window. You can then navigate between these two screens when working on the project. The Camera Preview previews what is seen in the Game window by the Main Camera.







Main Camera
Static
Tag MainCamera
Layer Default

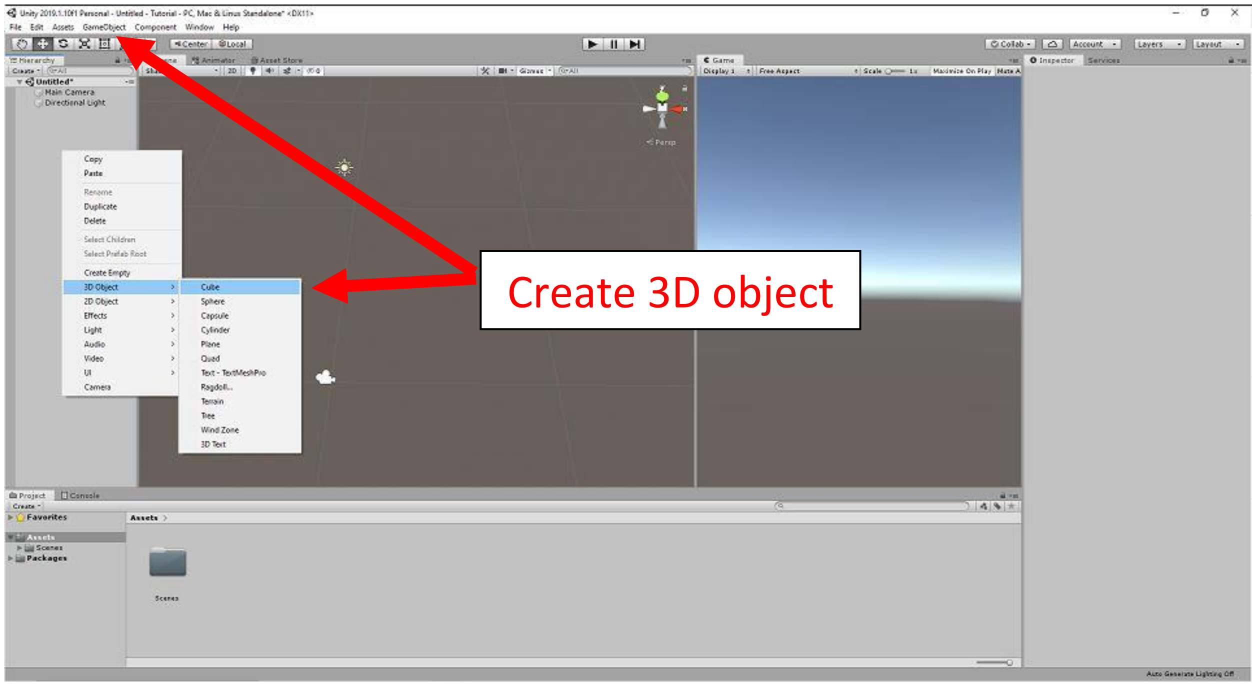
Transform
Position X 0 Y 1 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

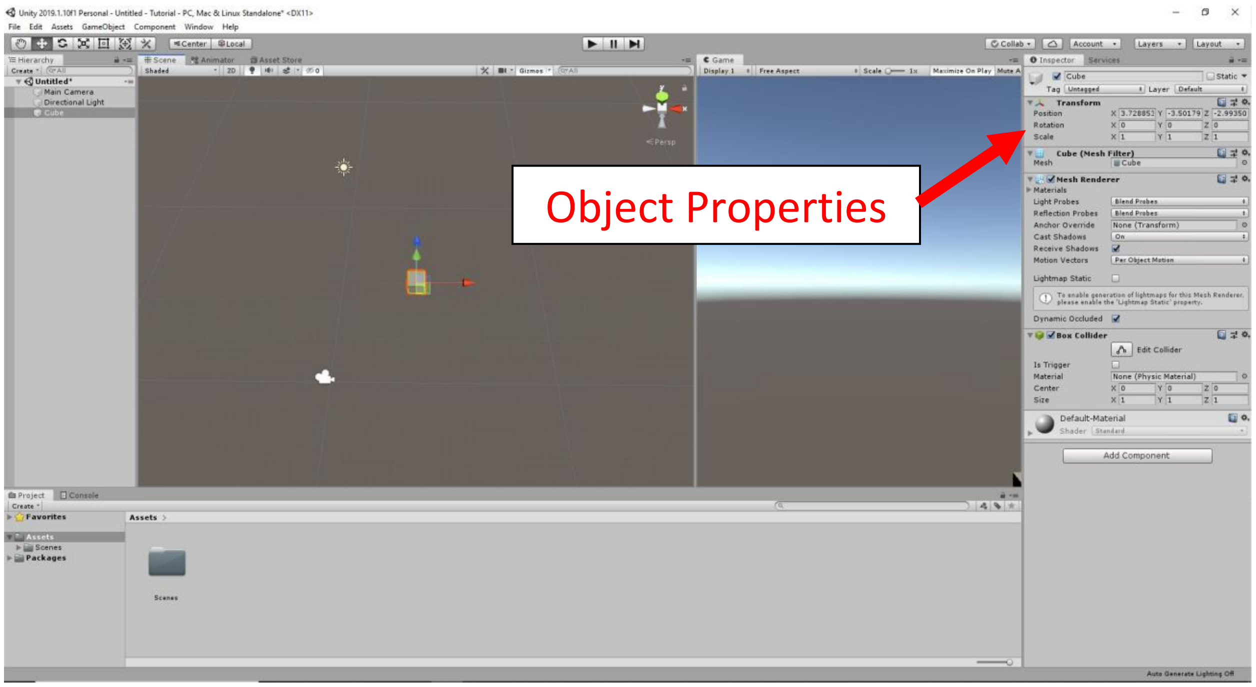
Camera
Clear Flags Skybox
Background
Culling Mask Everything
Projection Perspective
FOV Axis Vertical
Field of View 60
Physical Camera
Clipping Planes Near 0.3
Far 1000
Viewport Rect X 0 Y 0
W 1 H 1
Depth -1
Rendering Path Use Graphics Settings
Target Texture None (Render Texture)
Occlusion Culling ✓
HDR Use Graphics Settings
MSAA Use Graphics Settings
Allow Dynamic Resol
Target Display Display 1

Audio Listener

Add Component

4. How-to: Create Objects
    a. Right click on the Hierarchy tab or mouse over the GameObject tab at the top of the screen in the Dropdown menu to create a new 3D object.
    b. For this tutorial we will be creating multiple objects including a table, and two spheres. One sphere will be controlled by the User and the second will be designated as a Target sphere.





c. Additional Step: Moving Camera and Screen Orientation

   i. For better viewing and control of individual objects, move the orientation of the Scene screen by pressing the Middle Mouse or Right Mouse button to move and rotate the screen, respectively. The Scroll Wheel zooms Scene screen in and out.

   ii. To move individual objects, first select the object and use the Green, Blue, and Red arrows the change the objects Position.

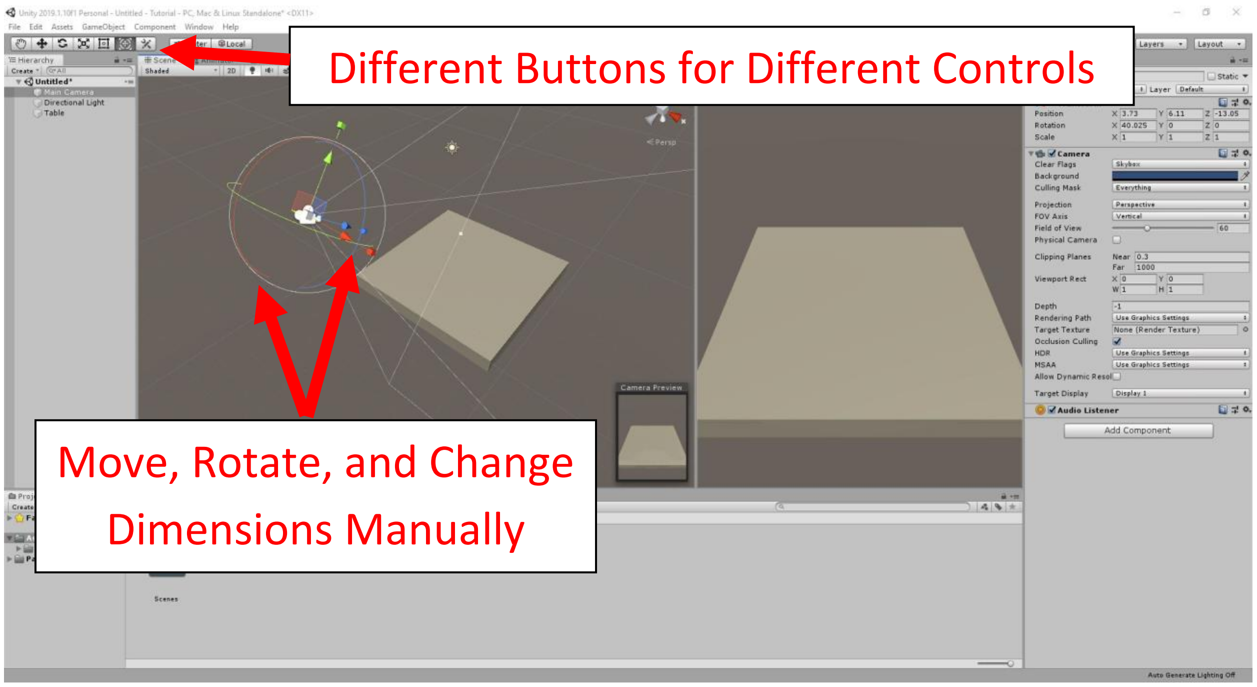
      1. You can also manually adjust the position by changing the numbers with the Inspector tab.

   iii. To rotate the object use the Green, Blue, and Red circles.

   iv. To change the dimensions of the object use the Green, Blue, and Red cubes.

   v. In the top left corner there are different buttons for different object controls related to movement, rotation, or movement+rotation.

   vi. Move the Main Camera to a position the looks down upon the Table object at a comfortable height and rotation (~45 degrees).



Position	X 3.73	Y 6.11	Z -13.05
Rotation	X 40.025	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Clear Flags	Skybox
Background	
Culling Mask	Everything
Projection	Perspective
FOV Axis	Vertical
Field of View	60
Physical Camera	☐
Clipping Planes	Near 0.3, Far 1000
Viewport Rect	X 0, Y 0, W 1, H 1
Depth	-1
Rendering Path	Use Graphics Settings
Target Texture	None (Render Texture)
Occlusion Culling	✓
HDR	Use Graphics Settings
MSAA	Use Graphics Settings
Allow Dynamic Resol	☐
Target Display	Display 1

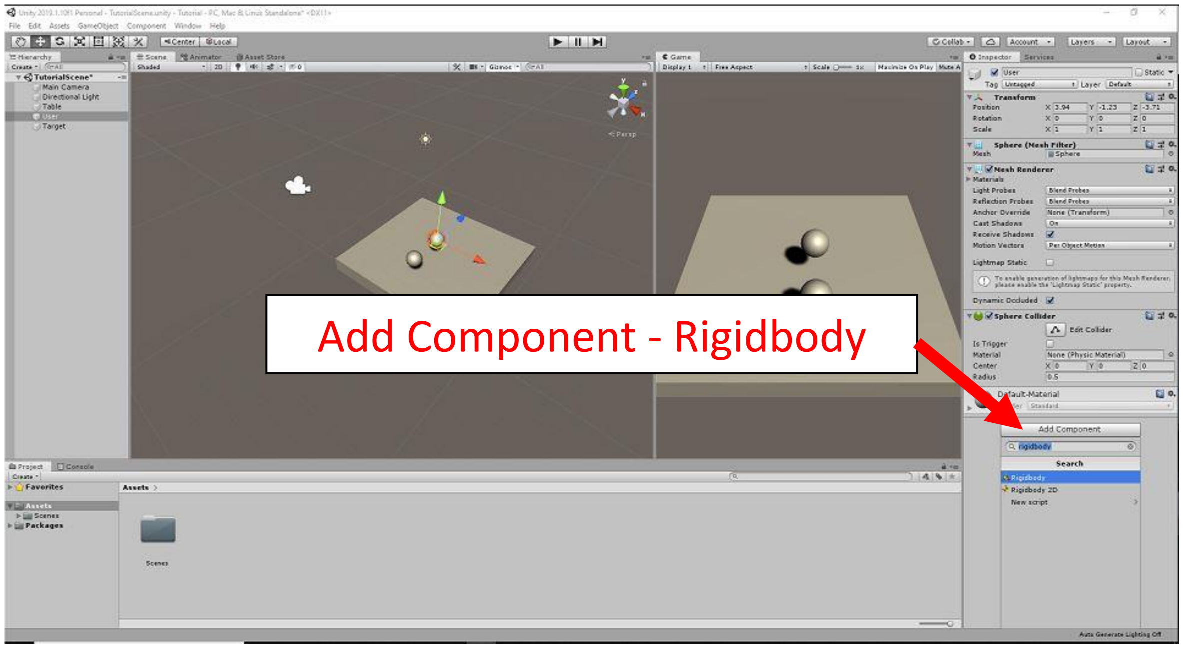
d. Next, create two spheres of equal size and label one ‘User’ and the other ‘Target’. Move both spheres to directly above the Table Top.

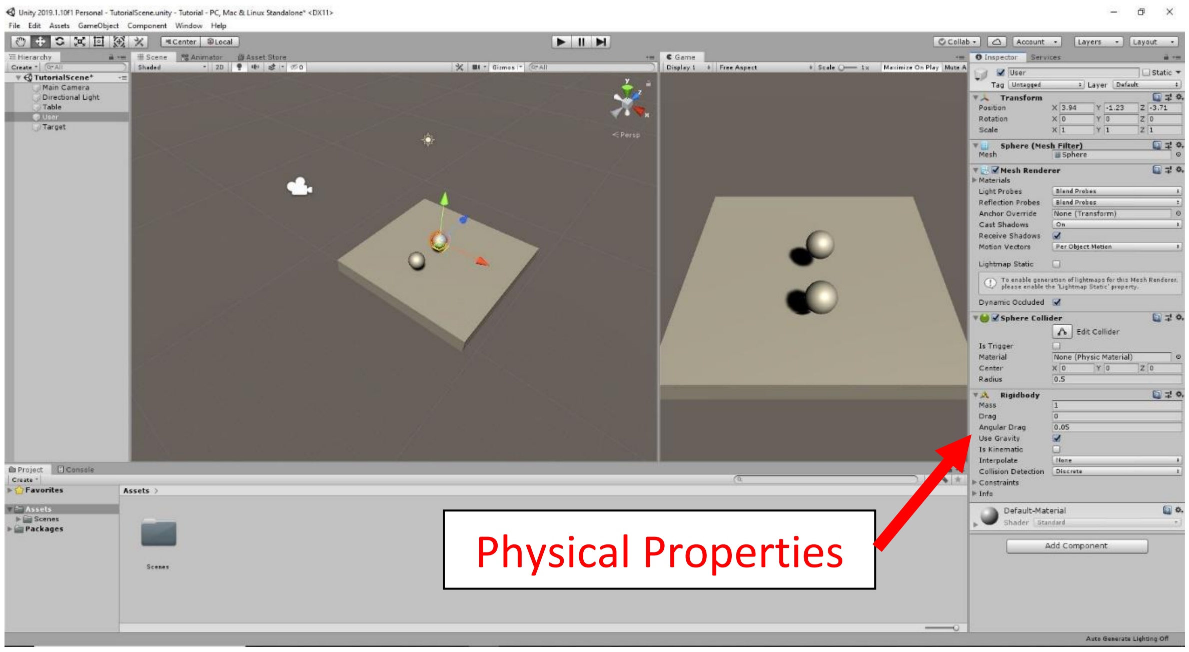
e. Select the User ball, press Add Component at the bottom of the Inspector tab, and select RigidBody.

  i. By adding the Rigidbody Component the sphere now has a specified Mass and other physical properties.

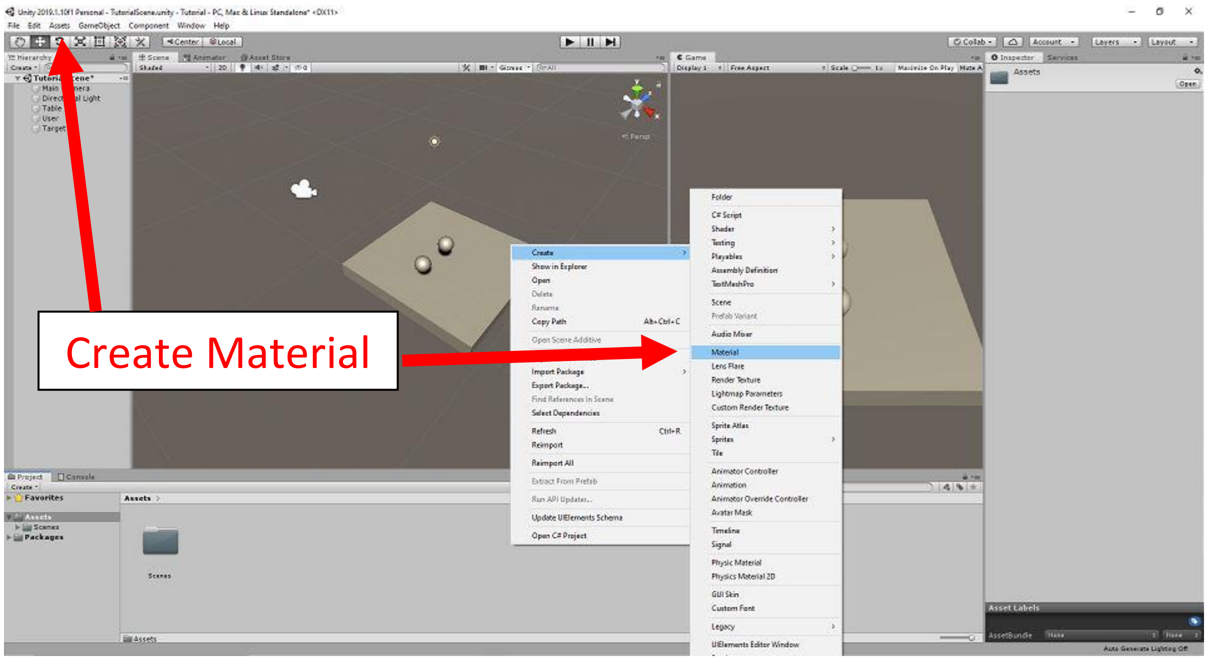
  ii. Ensure that the Use Gravity box is checked on.

f. Repeat the last step for the Target ball.

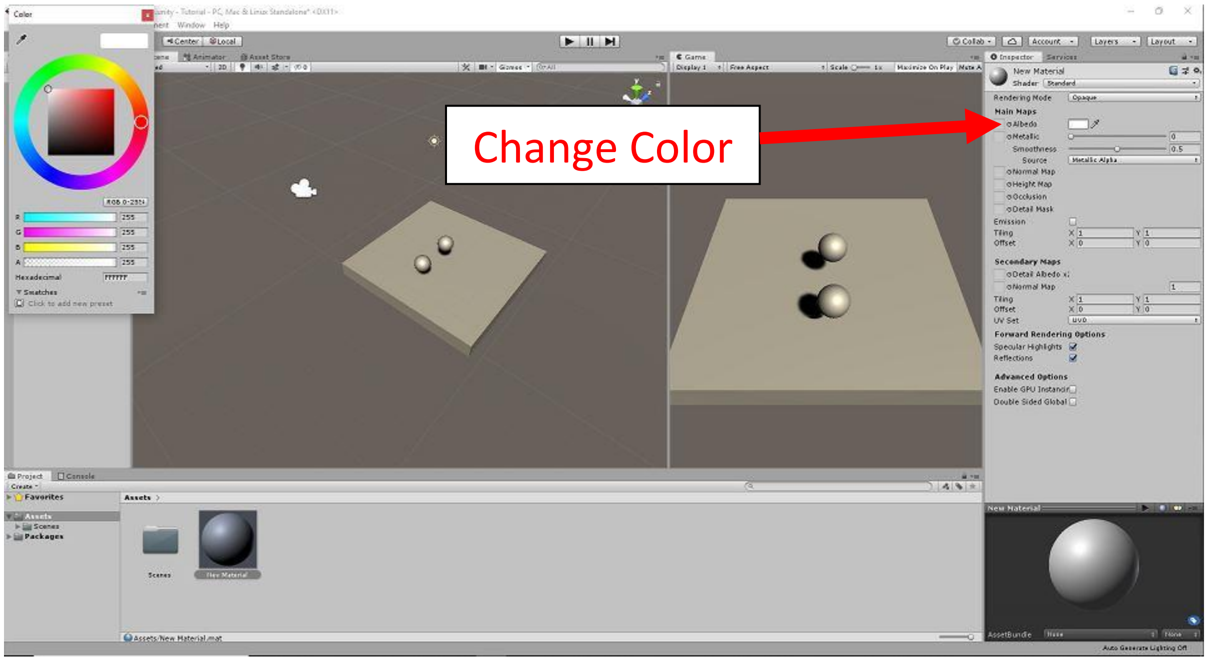




5. How-to: Create Materials
   a. Different material properties can be specified for each object such as color and transparency. For this tutorial, a basic introduction to different materials for different colors will be presented.
   b. This can also be found under the Assets tab at the Dropdown menu in the top left corner.



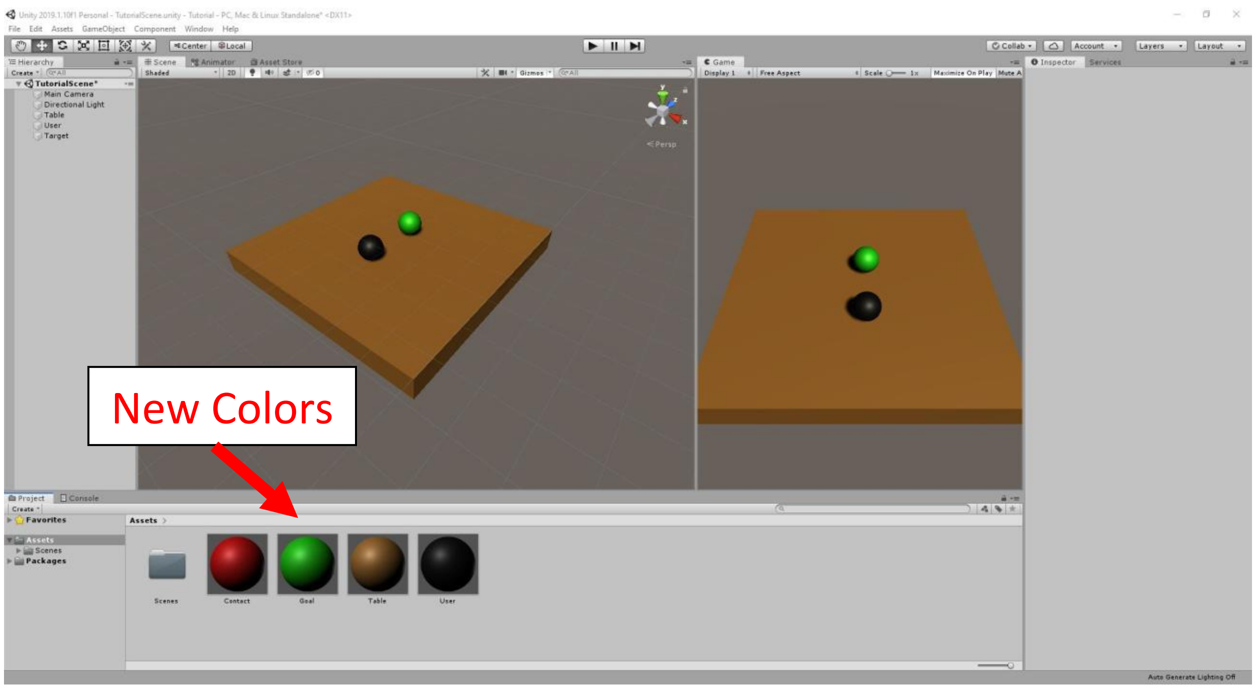
   c. After you create the material, select the white square next to the label Albedo under the Inspector tab to manually change the color.



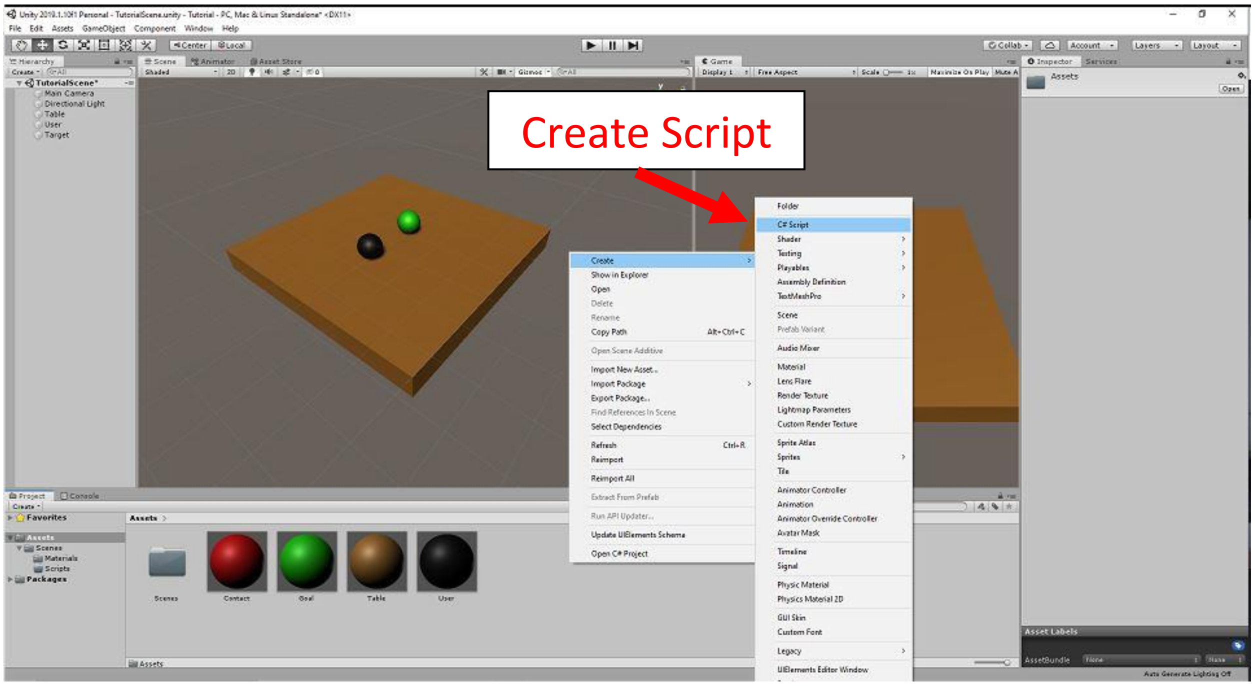
d. Create four colors. A brown one labeled Table, a black one labeled User, a green one labeled Goal, and a red one labeled Contact.

e. Drag and drop the material onto the object you want to apply the material too.

   i. Apply the Table color to the Table, the User object to the User sphere, and the Goal color to the Target object. The Contact object will be used at a later time.

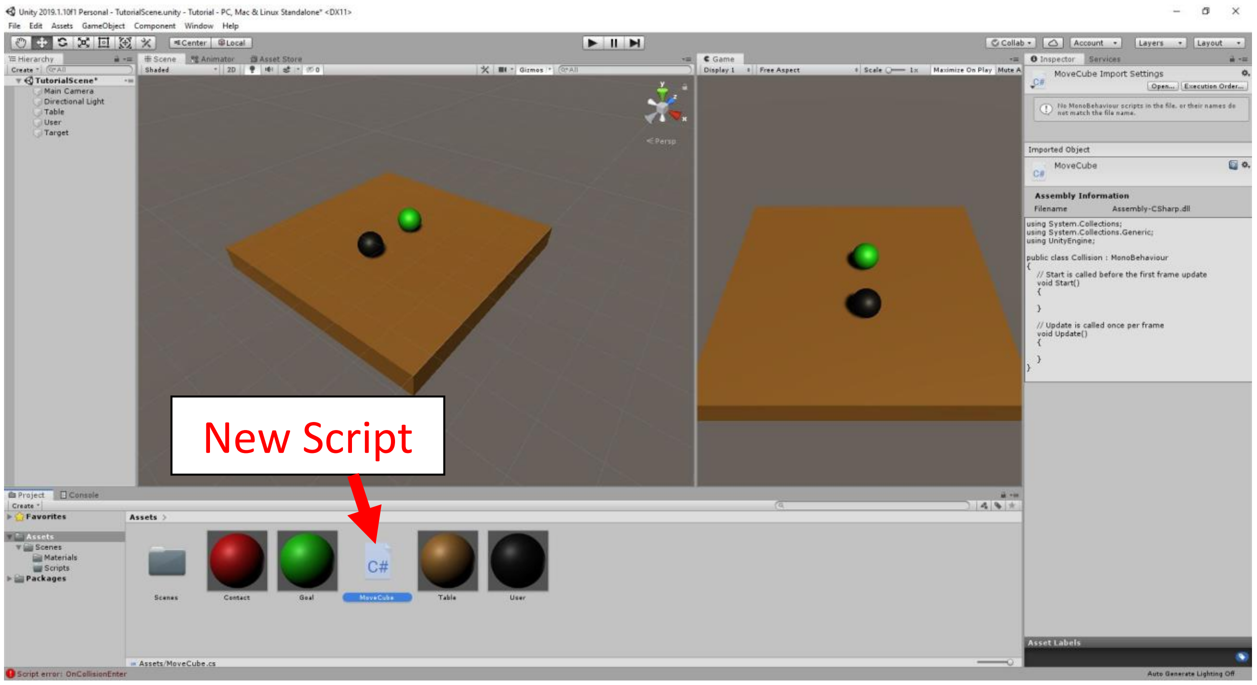


6. How-to: Create Scripts
    a. Now that the environment is created the final step is creating scripts that help run the task or for creating event triggers such as sound cues or color changes that correspond with your desired effect.
        i. We will create the following 2 scripts:
            1. One to move the User controlled sphere.
            2. One to change the material (and color) of both the User and Target spheres when they collide into each other.



b. Create a script and name it MoveCube. This Script will contain code written in C# within the Visual Studios program for controlling the User sphere with the computer keyboard.

   i. *Author note:* There are many ways to move an object. A google search for ‘moving an object in Unity’ will supply many examples. For example, you could apply a directional force that rolls the sphere across the table. In this tutorial we will present a simple code for changing the position in the X and Z directions based upon a set scaling factor.



No MonoBehaviour scripts in the file, or their names do not match the file name.

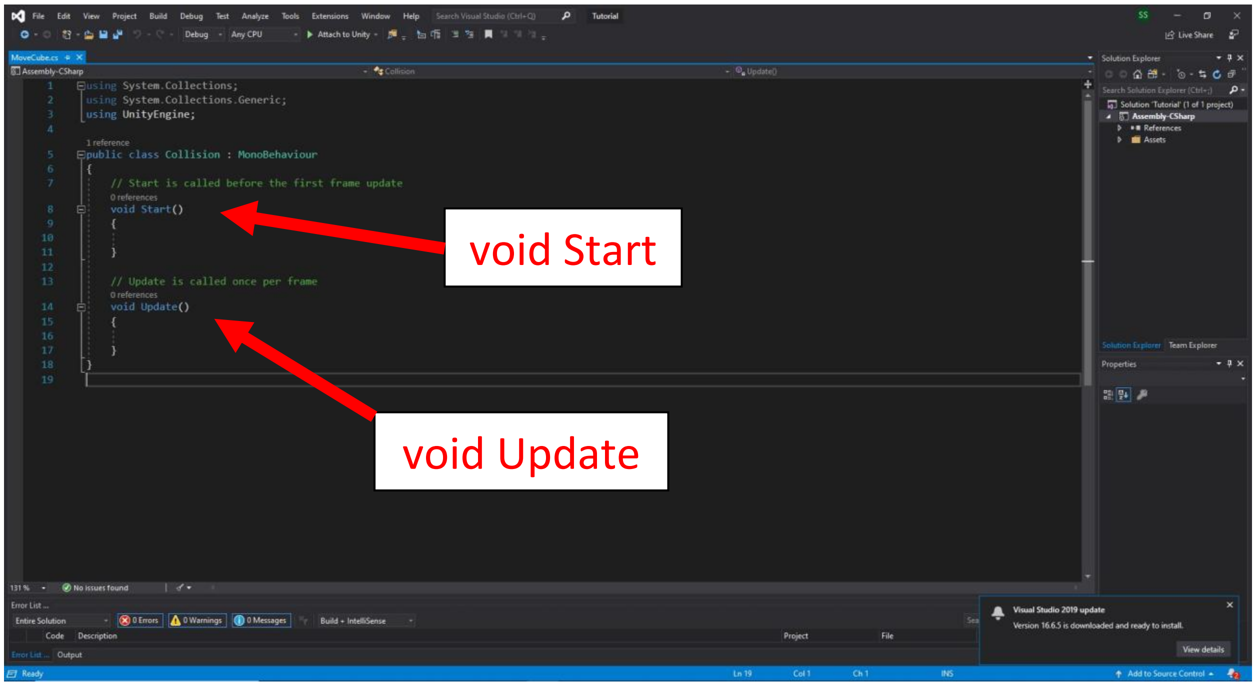
Assembly Information

Filename: Assembly-CSharp.dll

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

public class Collision : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

c. Double click on the new Script to load Visual Studio.

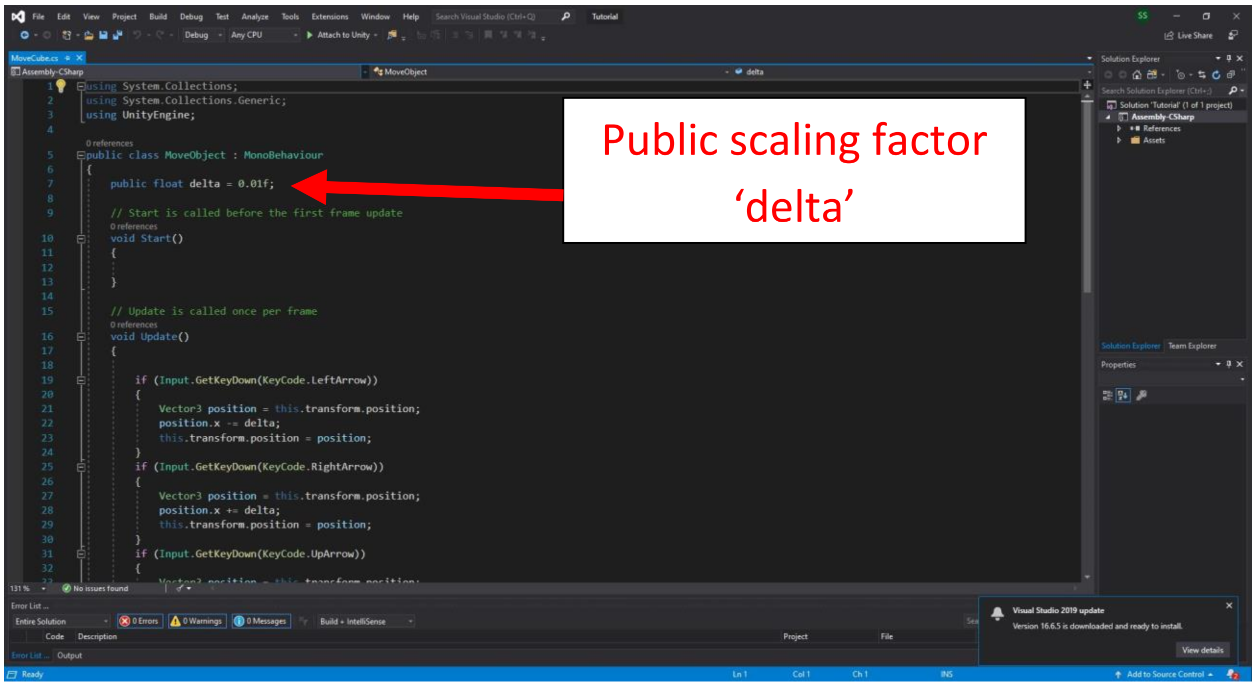
d. At the basic level, anything added following ‘void Start’ will be applied immediately upon playing the scene while ‘void Update’ will continuously run and be updated every frame of the scene (~50 Hz).



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

public class Collision : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

e. The following code has been created to change the position of the designated object based upon a scaling factor labeled as ‘delta’ according to arrow directions from a keyboard.

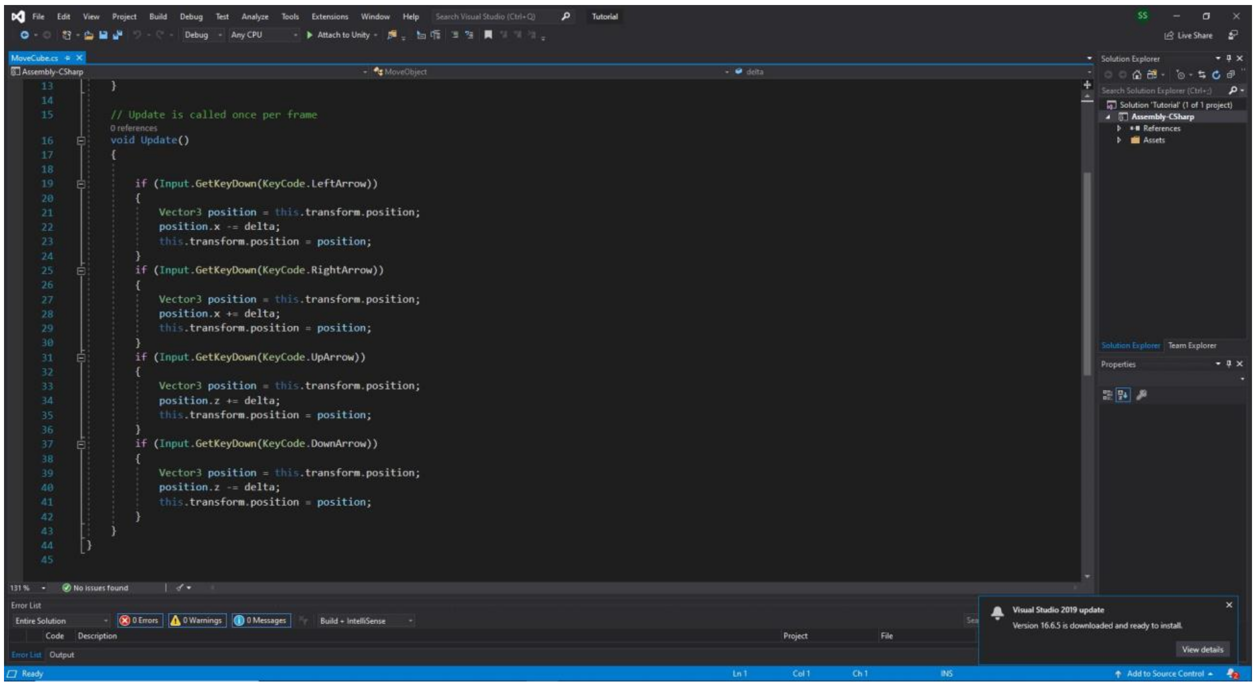
f. The variable labeled delta is identified as a ‘public’ variable. This means it is accessible from the User Main Screen within Unity and therefore can be changed. If the variable is set as ‘private’ it will be hidden from control within Unity and only accessible within Visual Studio.



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

public class MoveObject : MonoBehaviour
{
    public float delta = 0.01f;

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

    }

    // Update is called once per frame
    void Update()
    {

        if (Input.GetKeyDown(KeyCode.LeftArrow))
        {
            Vector3 position = this.transform.position;
            position.x -= delta;
            this.transform.position = position;
        }
        if (Input.GetKeyDown(KeyCode.RightArrow))
        {
            Vector3 position = this.transform.position;
            position.x += delta;
            this.transform.position = position;
        }
        if (Input.GetKeyDown(KeyCode.UpArrow))
        {
```

Visual Studio 2019 update
Version 16.6.5 is downloaded and ready to install.

g. Upon each Update,
    i. IF a Left Arrow is detected the object moves in the -X direction
        1. By a factor of delta (specified within the Unity Program).
    ii. IF a Right Arrow is detected the object moves in the +X direction.
    iii. IF an Up Arrow is detected the object moves in the +Z direction.
    iv. IF a Down Arrow is detected the object moves in the -Z direction.



```csharp
    }

    // Update is called once per frame
    0 references
    void Update()
    {

        if (Input.GetKeyDown(KeyCode.LeftArrow))
        {
            Vector3 position = this.transform.position;
            position.x -= delta;
            this.transform.position = position;
        }
        if (Input.GetKeyDown(KeyCode.RightArrow))
        {
            Vector3 position = this.transform.position;
            position.x += delta;
            this.transform.position = position;
        }
        if (Input.GetKeyDown(KeyCode.UpArrow))
        {
            Vector3 position = this.transform.position;
            position.z += delta;
            this.transform.position = position;
        }
        if (Input.GetKeyDown(KeyCode.DownArrow))
        {
            Vector3 position = this.transform.position;
            position.z -= delta;
            this.transform.position = position;
        }
    }
}
```

Author note: When creating this tutorial I accidentally swapped the User and Target sphere. Which sphere is which is completely up to you, but note here that the Target and User names are in different order from the previous steps.

> *Previously the upper sphere was renamed as User, in this case the black sphere which is positioned below, is labeled as 'User'. This will be corrected at a future date for consistency.*

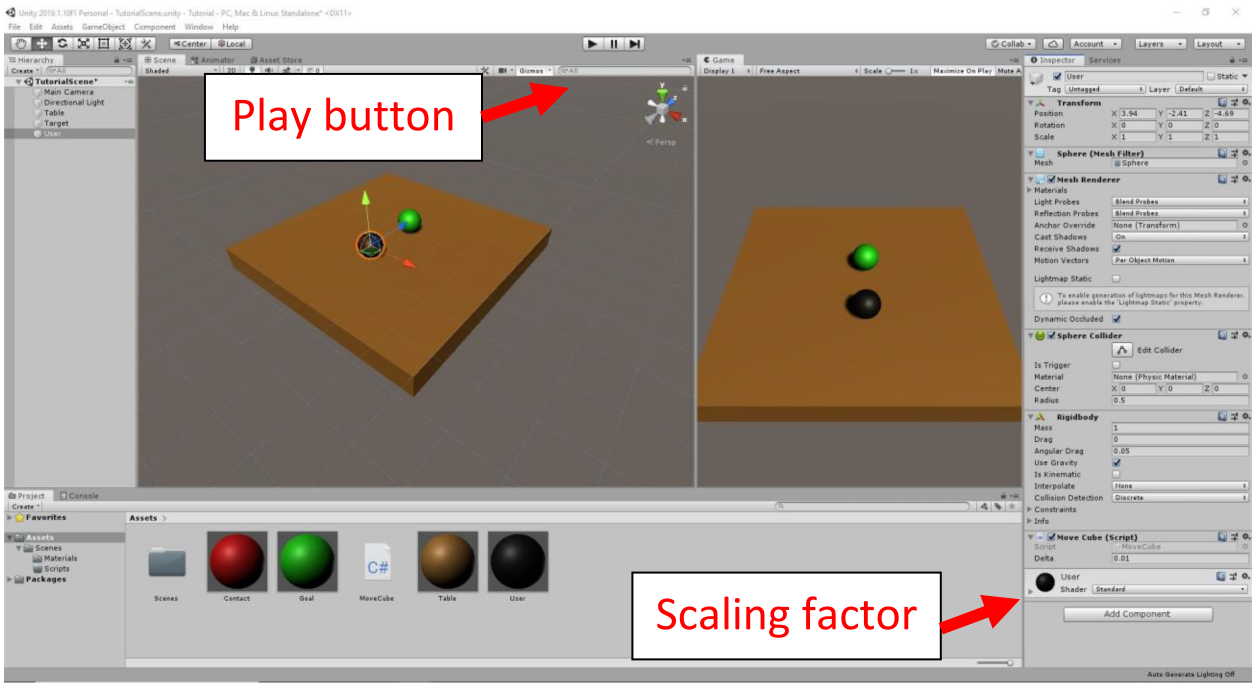
h. Apply the MoveCube script to the User sphere

i. There are multiple ways to apply a script to an object.

   i. You can press Add Component and find the correct Script you created.

   ii. You can drag and drop the script onto the component at the open area at bottom of the object information in the Inspector tab.

j. The 'delta' label can now be seen within the MoveCube script within the Inspector tab and can be changed between trials for modifying the scaling factor of the User sphere position change.



k. Now is a good time to test your Project if you have not already. Press the Play button at the top of the screen and wait for Unity to load. Once the program has loaded you should be able to control the position of the black User sphere by pressing the arrow keys on your keyboard. Each keypress is a movement command.

   i. *Note:* A 'delta' of 0.01 will be extremely low and movement will be tough to see at first. A recommended delta is between .1 and 1.

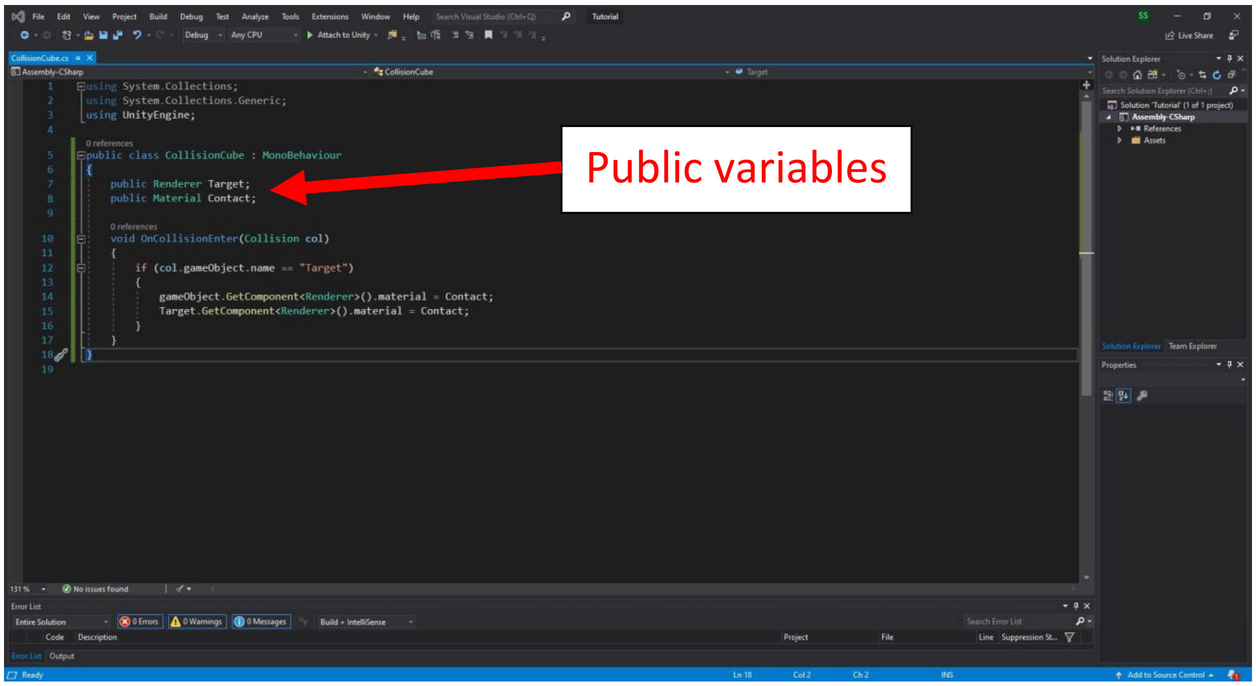
l. Next, create a second script and label it CollisionCube.
    i. This script will contain event triggers that will cause a color (material) change when the User sphere contacts the Target sphere.

m. The following image is an example script but as noted before there are multiple ways this can be done.

n. First, we created two public variables, on is a Renderer labeled Target and one is a Material labeled Contact.
    i. The Renderer label specifies which gameObject in the virtual environment is the 'Target' and will change color upon collision.
    ii. The Material label specifies which Material property will be used as the new material upon object collision.
        1. Because both objects are set as public they must both be set within the Unity environment. See below for further instructions.

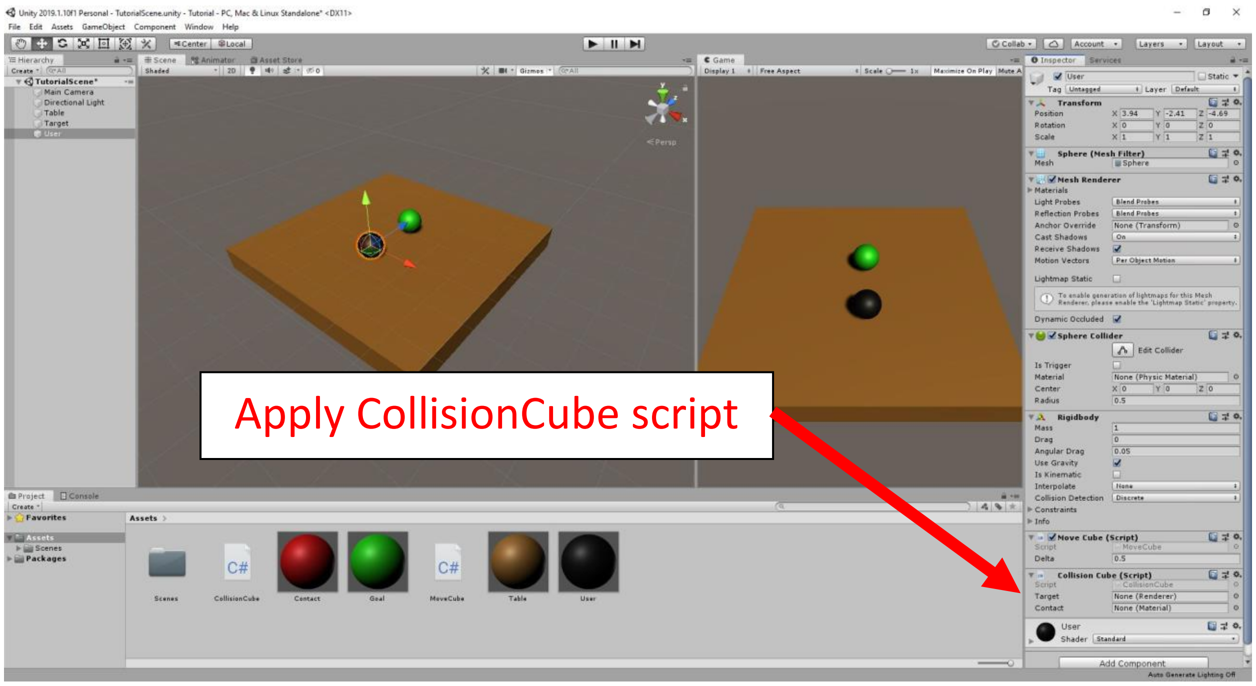
o. The 'void Start' and 'void Update' were removed and a 'void OnCollisionEnter' was used.
    i. This is a code that exists within the program and must be called upon and used in a specific way. *Google 'Unity oncillisionenter' for further documentation.*

p. Finally, the code below states that IF the gameObject (which is the object this script is attached to) makes contact with an additional gameObject named "Target" that both the gameObject and Target will change materials to the specified material, in this case named 'Contact'.

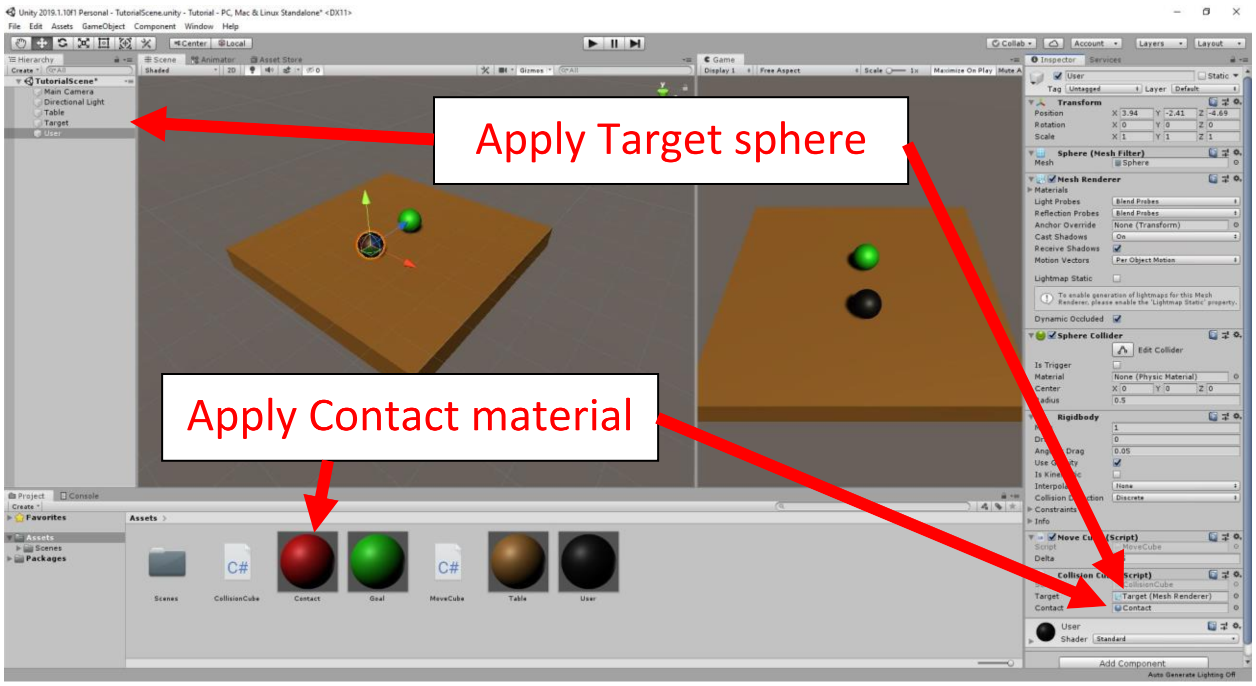


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 references
public class CollisionCube : MonoBehaviour
{
    public Renderer Target;
    public Material Contact;

    0 references
    void OnCollisionEnter(Collision col)
    {
        if (col.gameObject.name == "Target")
        {
            gameObject.GetComponent<Renderer>().material = Contact;
            Target.GetComponent<Renderer>().material = Contact;
        }
    }
}
```

q. Apply the CollisionCube script to the User sphere.

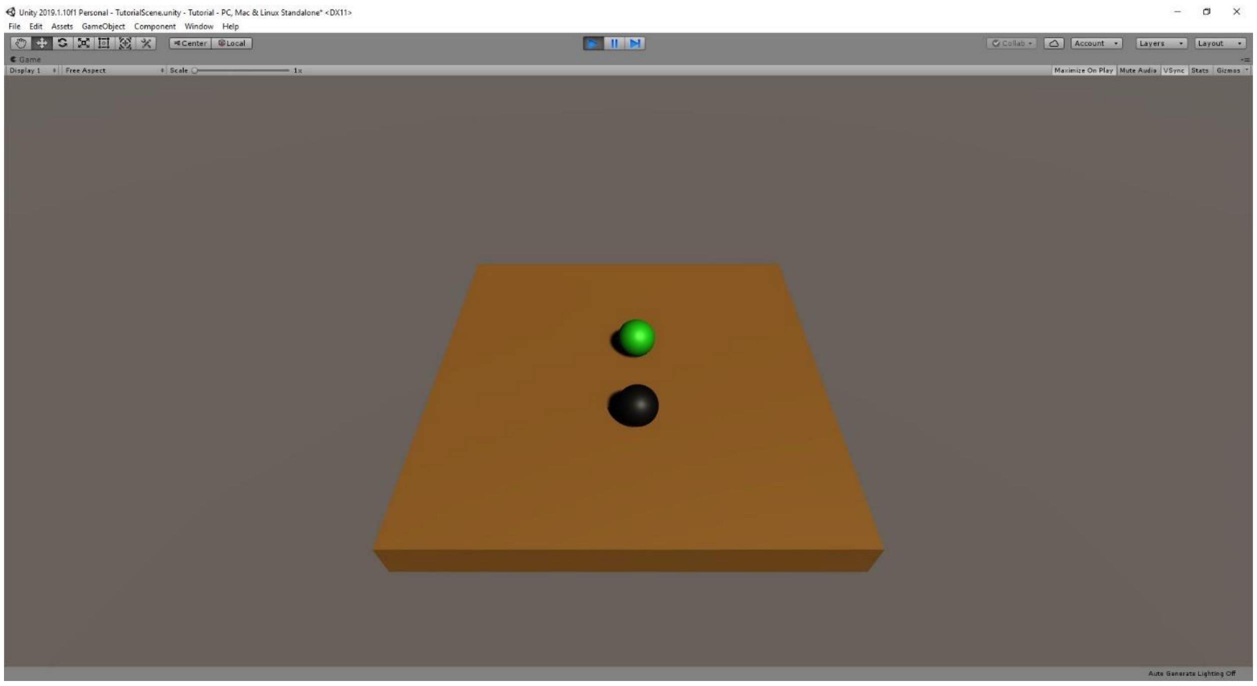


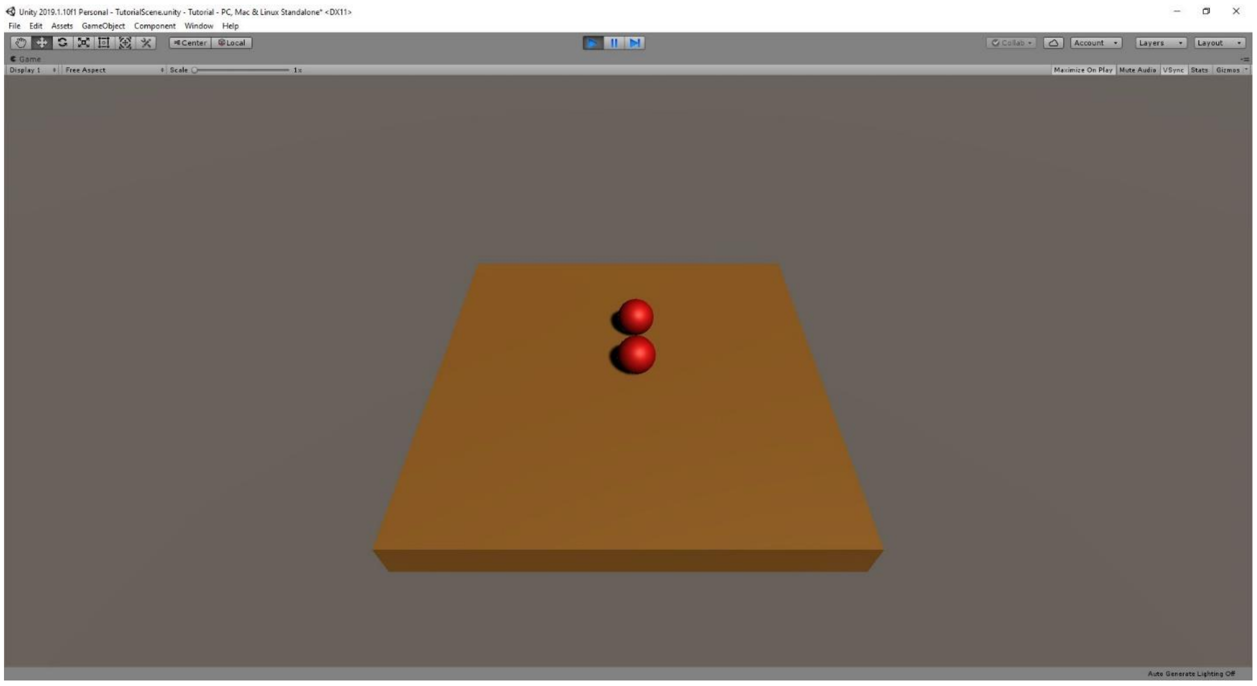
r. Click on the User sphere within the Hierarchy tab. Drag and drop the Target sphere from the Hierarchy tab to the Target option within the CollisionCube script within the User sphere. Additionally, drag and drop the Contact material to the Contact option within the same script.



s. Press Play to test the environment.

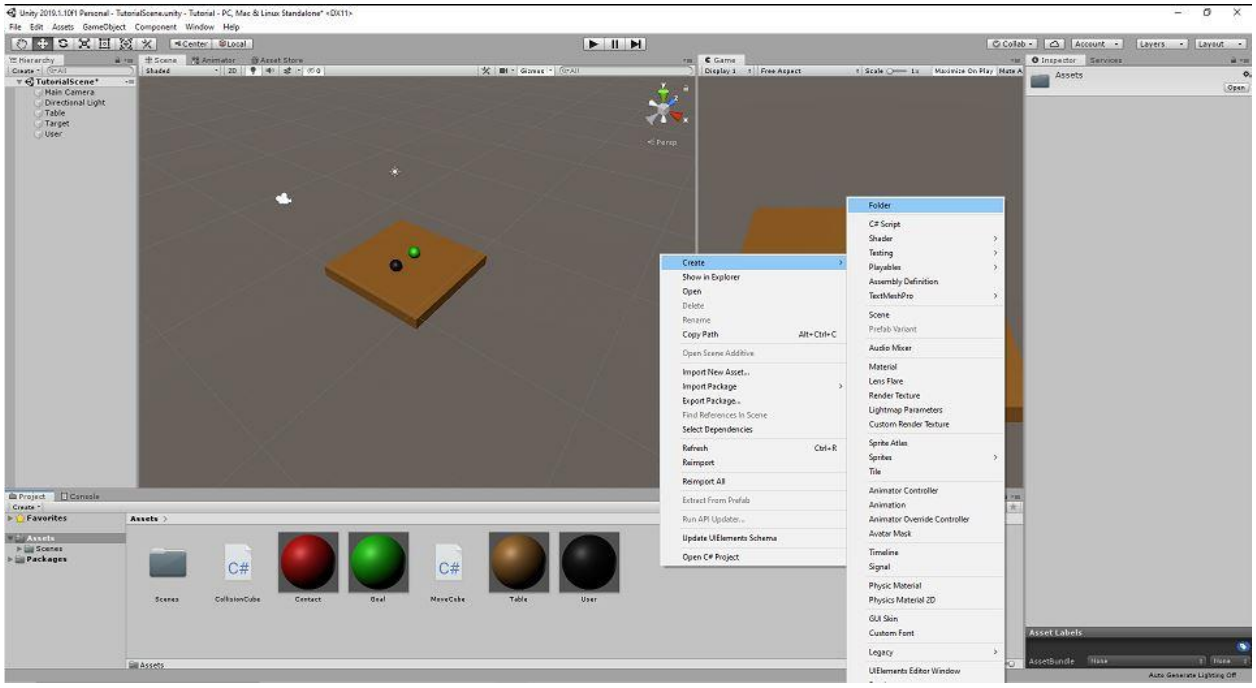
   i. When the User sphere contacts the Target sphere both objects should immediately change to red.

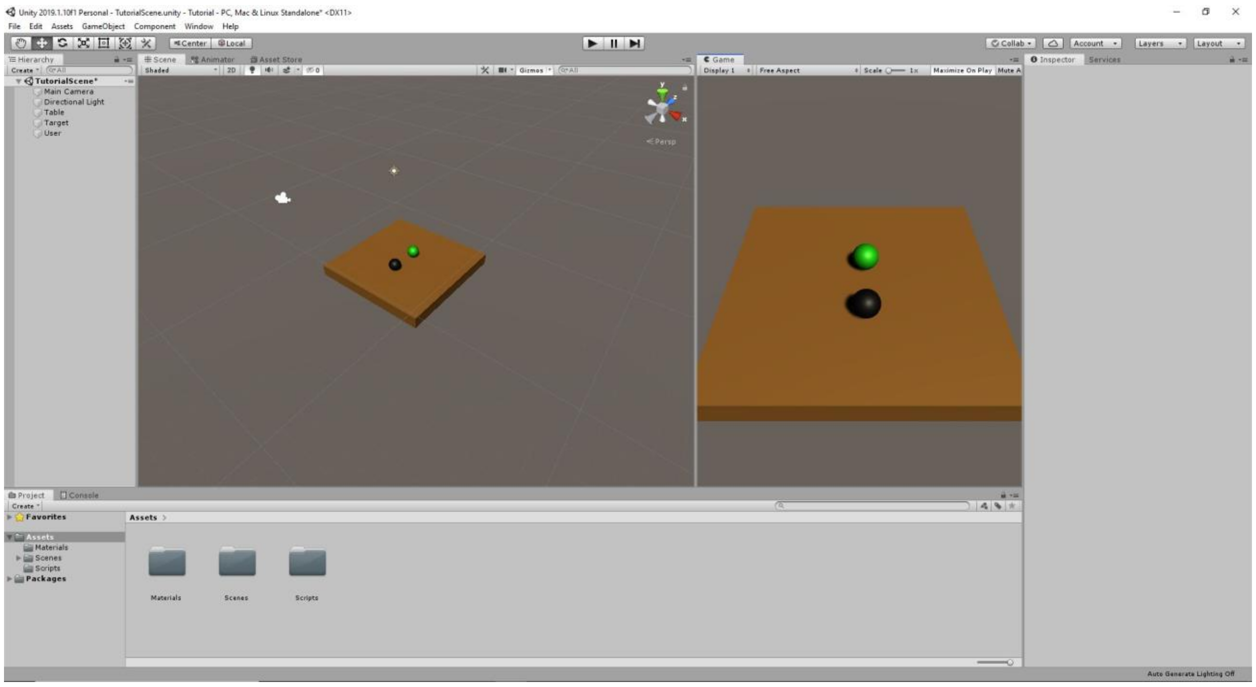

Unity 2019.1.10f1 Personal - TutorialScene.unity - Tutorial - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Game
Display 1 Free Aspect Scale 1x
Maximize On Play Mute Audio VSync Stats Gizmos
Auto Generate Lighting Off


Unity 2019.1.10f1 Personal - TutorialScene.unity - Tutorial - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Game
Display 1 Free Aspect Scale 1x
Maximize On Play Mute Audio VSync Stats Gizmos
Auto Generate Lighting Off

Congratulations! You have completed the MOCORE tutorial for creating a Unity project. Additional projects can be found in the following sections and downloaded from the MOCORE website. The basics will be removed, and the instructions will detail the specific differences with each project and the additional information not covered in this tutorial.

Optional: You can create folders to better organize your materials, scripts, and scenes.


Create
Show in Explorer
Open
Delete
Rename
Copy Path
Open Scene Additive
Import New Asset...
Import Package
Export Package...
Find References In Scene
Select Dependencies
Refresh
Reimport
Reimport All
Extract From Prefab
Run API Updater...
Update UIElements Schema
Open C# Project
Folder
C# Script
Shader
Testing
Playables
Assembly Definition
TextMeshPro
Scene
Prefab Variant
Audio Mixer
Material
Lens Flare
Render Texture
Lightmap Parameters
Custom Render Texture
Sprite Atlas
Sprites
Tile
Animator Controller
Animation
Animator Override Controller
Avatar Mask
Timeline
Signal
Physic Material
Physics Material 2D
GUI Skin
Custom Font
Legacy
UIElements Editor Window
Scenes
CollisionCube
Contact
Goal
MoveCube
Table
User


TutorialScene*
Main Camera
Directional Light
Table
Target
User
Assets
Materials
Scenes
Scripts
Packages