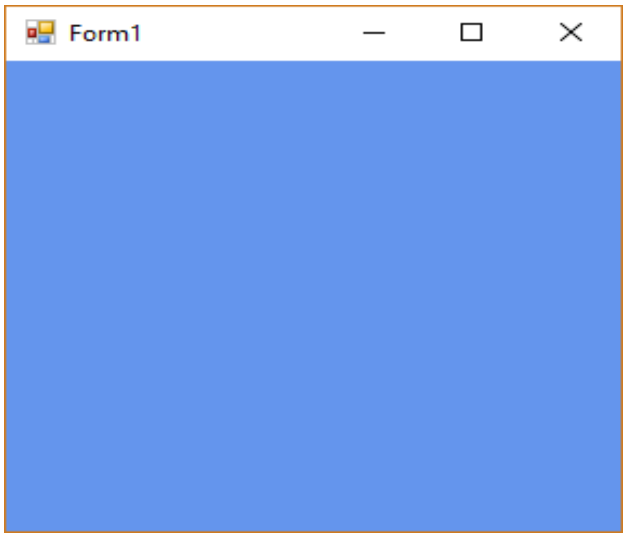
PRACTICAL NO:-01

AIM:-Setup DirectX 11, Window Framework and Initialize Direct3D Device.

INPUT:-

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX.Direct3D;
namespace directx                    {
public partial class Form1 : Form           {
Microsoft.DirectX.Direct3D.Device devices;
public Form1()  {
InitializeComponent();
InitDevices();    }
private void InitDevices()                    {
PresentParameters pp = new PresentParameters();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
devices = new Device(0,DeviceType.Hardware,this,CreateFlags.
HardwareVertexProcessing,pp); }
private void Render()    {
devices.Clear(ClearFlags.Target,Color.CornflowerBlue,0,1);
devices.Present();         }
private void Form1_Paint(object sender, PaintEventArgs e)        {
Render();         }}}
```

OUTPUT:-



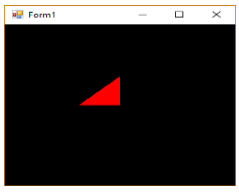
PRACTICAL NO:-02

AIM:-Buffers, Shaders and HLSL (Draw a triangle using Direct3D 11).

INPUT:-

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;
namespace Pratical_02  {
public partial class Form1 : Form            {
private Device device;
private CustomVertex.PositionColored[] vertex = new CustomVertex.PositionColored[3];
public Form1()  {
InitializeComponent();  }
private void Form1_Load(object sender, EventArgs e)     {
PresentParameters pp = new PresentParameters();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
device = new Device(0,DeviceType.Hardware,this,CreateFlags.
HardwareVertexProcessing,pp);
device.Transform.Projection = Matrix.PerspectiveFovLH(3.14f / 4, device.Viewport.Width /
device.Viewport.Height, 1f, 1000f);
device.Transform.View = Matrix.LookAtLH(new Vector3(0, 0, 20), new Vector3(), new Vector3(0, 1,
0));
device.RenderState.Lighting = false;
vertex[0] = new CustomVertex.PositionColored(new Vector3(),Color.Red.ToArgb());
vertex[1] = new CustomVertex.PositionColored(new Vector3(3,0,0), Color.Red.ToArgb());
vertex[2] = new CustomVertex.PositionColored(new Vector3(0,3,0), Color.Red.ToArgb()); }
private void Form1_Paint(object sender, PaintEventArgs e)        {
device.Clear(ClearFlags.Target, Color.Black, 1, 0);
device.BeginScene();
device.VertexFormat = CustomVertex.PositionColored.Format;
device.DrawUserPrimitives(PrimitiveType.TriangleList, vertex.Length / 3, vertex);
device.EndScene();
device.Present();          }}}
```

OUTPUT:-



PRACTICAL NO:-03

AIM:-Texturing (Texture the Triangle using Direct 3D 11).

INPUT:-

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;
namespace Pratical_04_Adding_Texture            {
public partial class Form1 : Form           {
private Device device;
private CustomVertex.PositionTextured[] vertex = new CustomVertex.PositionTextured[3];
private Texture texture;
public Form1()              {
InitializeComponent();   }
private void Form1_Load(object sender, EventArgs e)     {
PresentParameters pp = new PresentParameters();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
device = new Device(0, DeviceType.Hardware, this, CreateFlags.HardwareVertexProcessing, pp);
device.Transform.Projection = Matrix.PerspectiveFovLH(3.14f / 4, device.Viewport.Width /
device.Viewport.Height, 1f, 1000f);
device.Transform.View = Matrix.LookAtLH(new Vector3(0, 0, 20), new Vector3(), new Vector3(0, 1,
0));
device.RenderState.Lighting = false;
vertex[0] = new CustomVertex.PositionTextured(new Vector3(0, 1, 1), 0, 0);
vertex[1] = new CustomVertex.PositionTextured(new Vector3(-1, -1, 1), -1, 0);
vertex[2] = new CustomVertex.PositionTextured(new Vector3(1, -1, 1), 0, -1);
// vertex[0]=new CustomVertex.PositionTextured(new Vector3(0,1,1),0,0);
texture = new Texture(device, new Bitmap("c:\\users\\nimesh\\documents\\visual studio
2010\\Projects\\Pratical_04_Adding_Texture\\Pratical_04_Adding_Texture\\listing_img3.jpg"), 0,
Pool.Managed);}
private void Form1_Paint(object sender, PaintEventArgs e)                   {
device.Clear(ClearFlags.Target, Color.Black, 1, 0);
device.BeginScene();
device.SetTexture(0,texture);
device.VertexFormat = CustomVertex.PositionTextured.Format;
device.DrawUserPrimitives(PrimitiveType.TriangleList, vertex.Length / 3, vertex);
device.EndScene();
device.Present();           }}}
```

OUTPUT:-

PRACTICAL NO:-04

AIM:-Lightning (Programmable Diffuse Lightning using Direct3D 11).

INPUT:-

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;
namespace Pratical_05  {
public partial class Form1 : Form           {
private Microsoft.DirectX.Direct3D.Device device;
private CustomVertex.PositionNormalColored[] vertex = new
CustomVertex.PositionNormalColored[3];
public Form1()  {
InitializeComponent();   }
private void Form1_Load(object sender, EventArgs e)     {
PresentParameters pp = new PresentParameters();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
device = new Device(0, DeviceType.Hardware, this, CreateFlags.HardwareVertexProcessing, pp);
device.Transform.Projection = Matrix.PerspectiveFovLH(3.14f / 4, device.Viewport.Width /
device.Viewport.Height, 1f, 1000f);
device.Transform.View = Matrix.LookAtLH(new Vector3(0, 0, 10), new Vector3(), new Vector3(0, 1,
0));
device.RenderState.Lighting = false;
vertex[0] = new CustomVertex.PositionNormalColored(new Vector3(0, 1, 1), new Vector3(1, 0, 1),
Color.Red.ToArgb());
vertex[1] = new CustomVertex.PositionNormalColored(new Vector3(-1, -1, 1), new Vector3(1, 0, 1),
Color.Red.ToArgb());
vertex[2] = new CustomVertex.PositionNormalColored(new Vector3(1, -1, 1), new Vector3(-1, 0, 1),
Color.Red.ToArgb());
device.RenderState.Lighting = true;
device.Lights[0].Type = LightType.Directional;
device.Lights[0].Diffuse = Color.Plum;
device.Lights[0].Direction = new Vector3(0.8f, 0, -1);
device.Lights[0].Enabled = true;            }
private void Form1_Paint(object sender, PaintEventArgs e)        {
device.Clear(ClearFlags.Target, Color.CornflowerBlue, 1, 0);
device.BeginScene();
device.VertexFormat = CustomVertex.PositionNormalColored.Format;
device.DrawUserPrimitives(PrimitiveType.TriangleList, vertex.Length / 3, vertex);
device.EndScene();
device.Present();          }}}
```

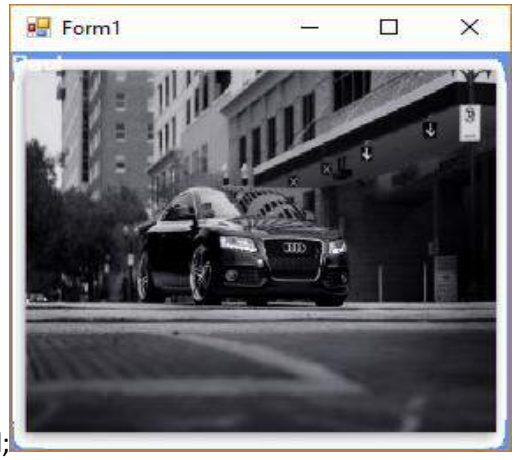
OUTPUT:-

PRACTICAL NO:-06

AIM:-Loading models into DirectX 11 and rendering.

INPUT:-

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;
namespace Pratical_06            {
public partial class Form1 : Form           {
Microsoft.DirectX.Direct3D.Device device;
Microsoft.DirectX.Direct3D.Texture texture;
Microsoft.DirectX.Direct3D.Font font;
public Form1()              {
InitializeComponent();
InitDevice();
InitFont();
LoadTexture();              }
private void Form1_Load(object sender, EventArgs e)     {}
private void InitFont()    {
System.Drawing.Font f = new System.Drawing.Font("Arial", 16f,FontStyle.Regular);
font = new Microsoft.DirectX.Direct3D.Font(device, f);    }
private void LoadTexture()        {
texture = TextureLoader.FromFile(device,"C:\\Users\\NIMESH\\Documents\\Visual Studio2010\\
Projects\\Pratical_06\\Pratical_06\\a5.jpg",400, 400, 1, 0, Format.A8B8G8R8, Pool.Managed,
Filter.Point, Filter.Point,Color.Transparent.ToArgb());      }
private void InitDevice()           {
PresentParameters pp = new PresentParameters();
pp.Windowed = true;
```




```
pp.SwapEffect = SwapEffect.Discard;
device = new Device(0, DeviceType.Hardware, this,
```

```
CreateFlags.HardwareVertexProcessing, pp);      }
private void Render()     {
device.Clear(ClearFlags.Target, Color.CornflowerBlue, 0, 1);
device.BeginScene(); using (Sprite s = new Sprite(device)) { s.Begin(SpriteFlags.AlphaBlend);
s.Draw2D(texture, new Rectangle(0, 0, 0, 0), new Rectangle(0, 0, device.Viewport.Width,
device.Viewport.Height), new Point(0, 0), 0f, new Point(0, 0), Color.White); font.DrawText(s, "Paul ",
new Point(0, 0), Color.White); s.End(); } device.EndScene(); device.Present();       }
private void Form1_Paint(object sender, PaintEventArgs e)         {
Render();          }}}
```

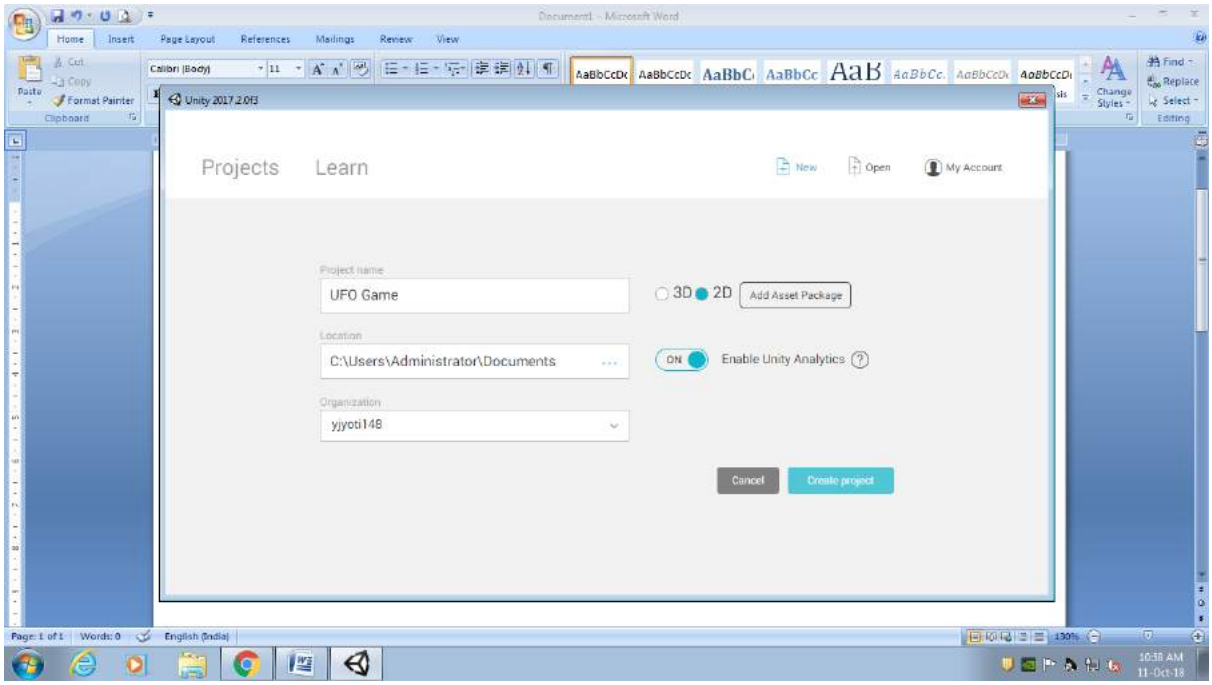
OUTPUT:-

PRACTICAL NO:-07

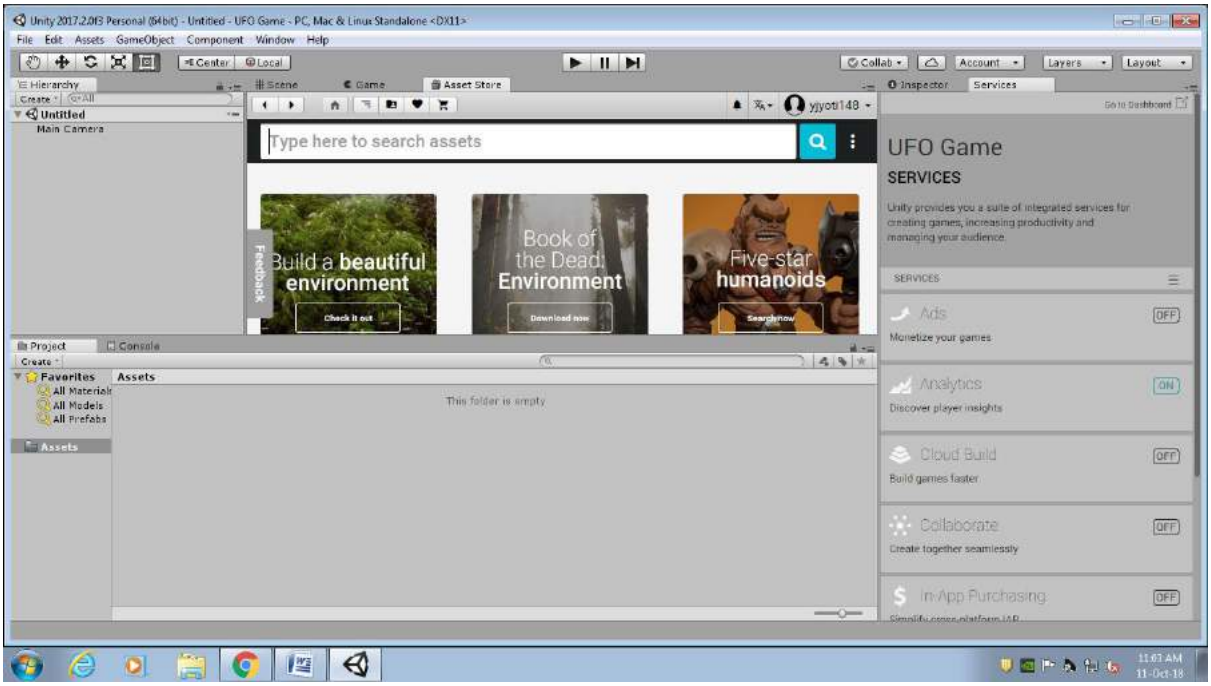
AIM:- Game development using unity 2D UFO.

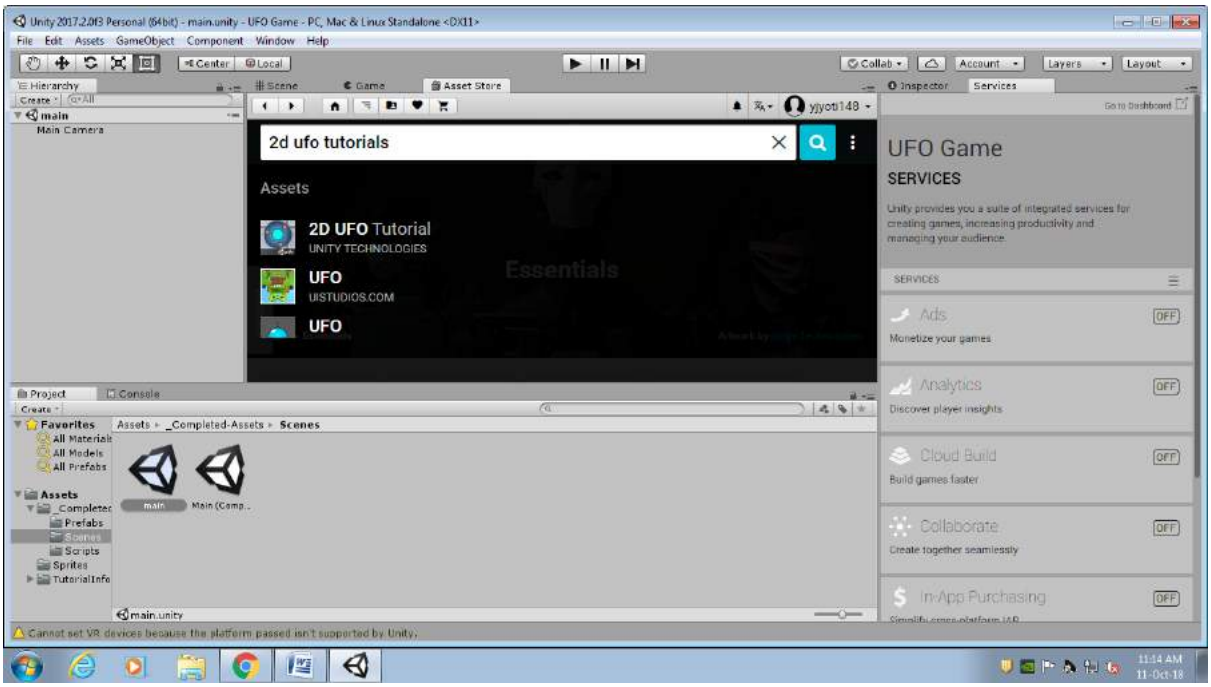
STEPS

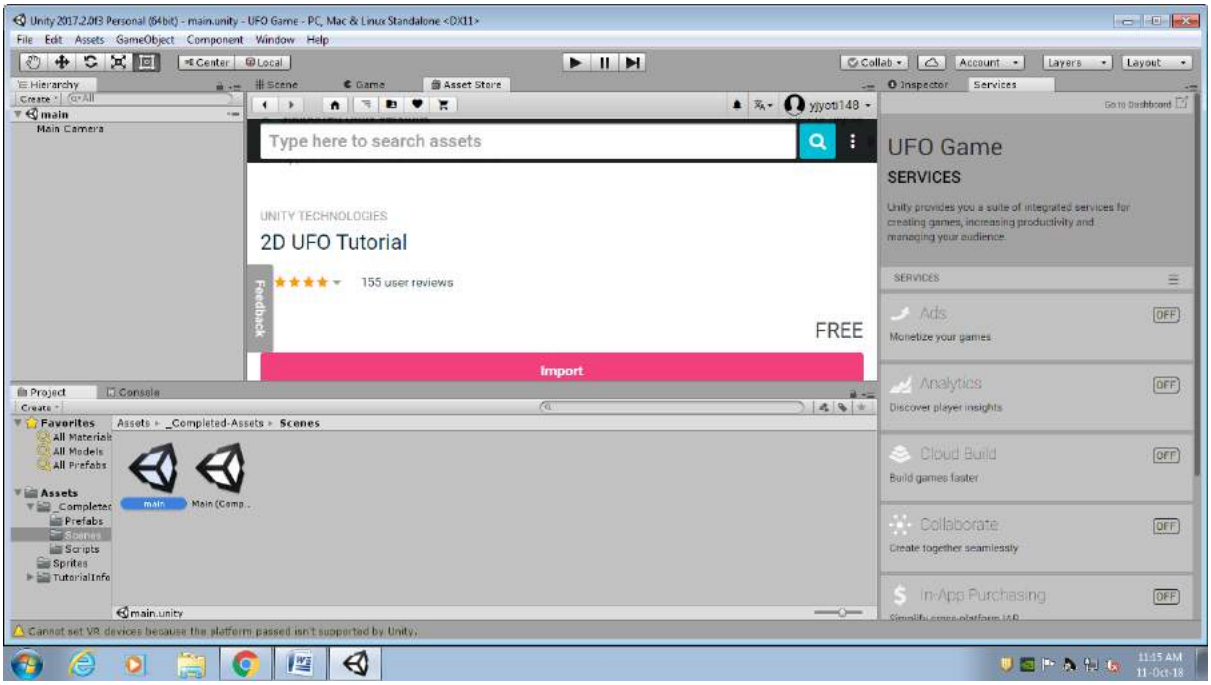
1} First give the project name and set the preferences to 2D and then click on Create Project button. File->New->Project Name



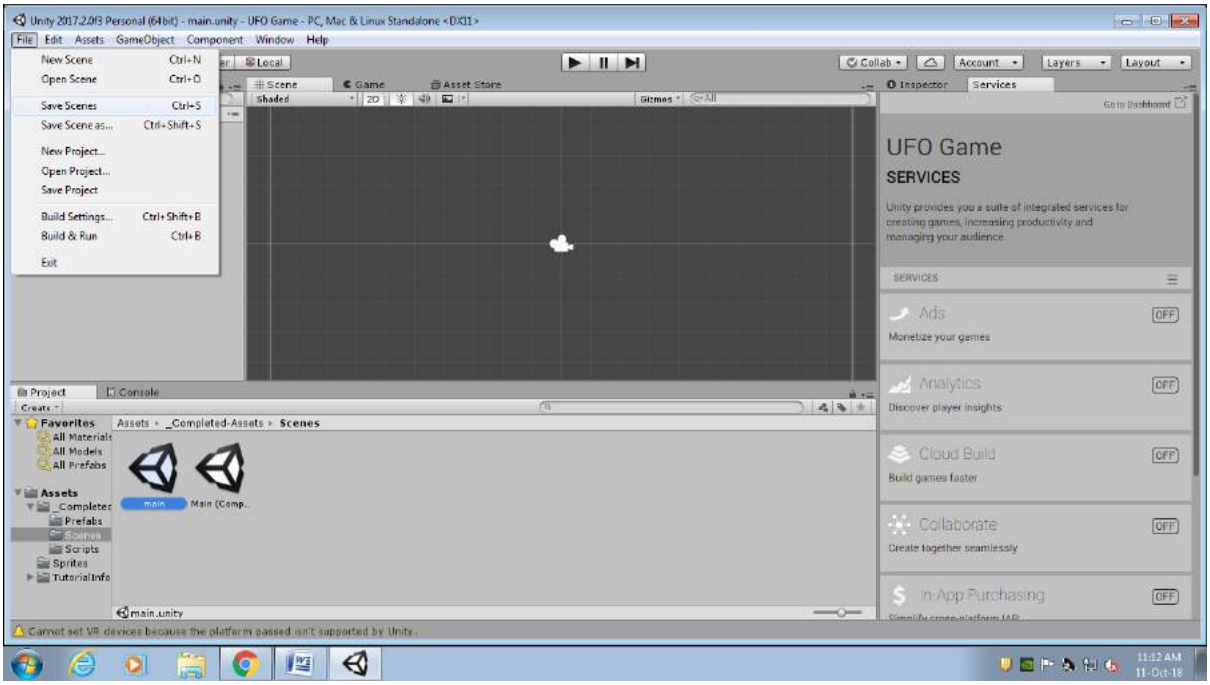
2} Now Download and Import our Assets from Assets Store. Windows -> Assets Store -> Search (2D UFO Tutorials) -> download and import .


Unity 2017.2.0f3 Personal (64bit) - Untitled - UFO Game - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create
Untitled
Main Camera
Scene Game Asset Store
yjyoti148
Type here to search assets
Feedback
Build a beautiful environment
Check it out
Book of the Dead: Environment
Download now
Five-star humanoids
Search now
Inspector Services
Go to Dashboard
UFO Game
SERVICES
Unity provides you a suite of integrated services for creating games, increasing productivity and managing your audience.
SERVICES
Ads
OFF
Monetize your games
Analytics
ON
Discover player insights
Cloud Build
OFF
Build games faster
Collaborate
OFF
Create together seamlessly
In-App Purchasing
OFF
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
Assets
Assets
This folder is empty
11:07 AM
11-Oct-18

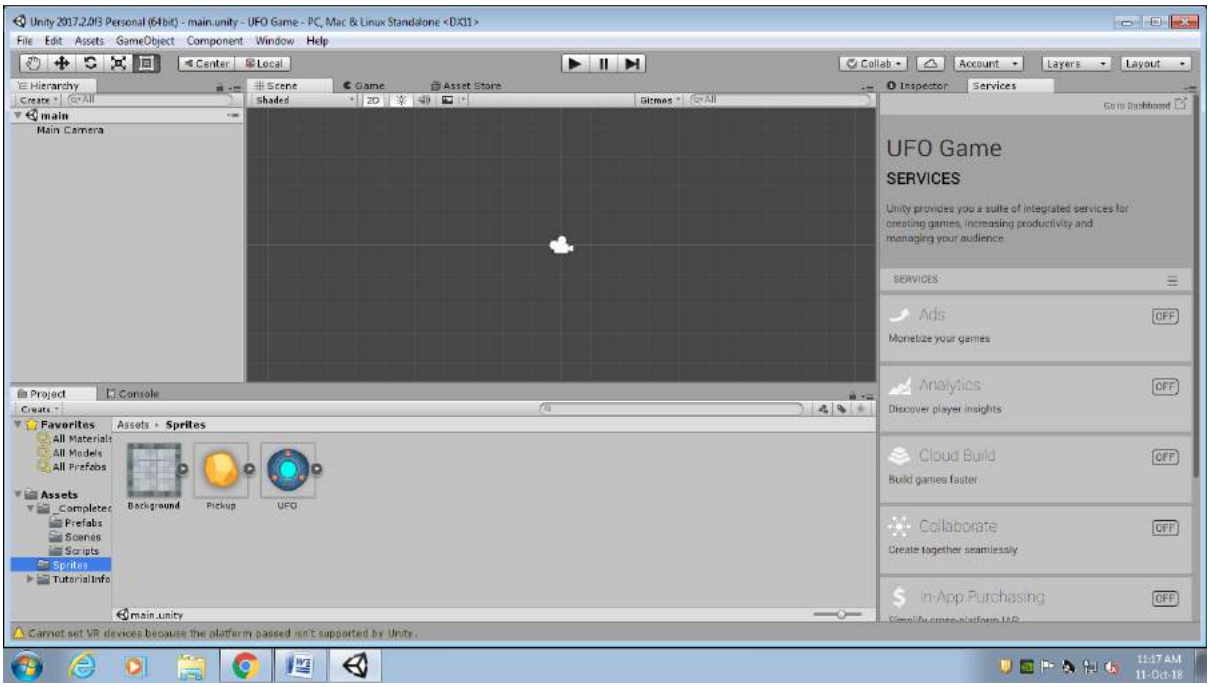


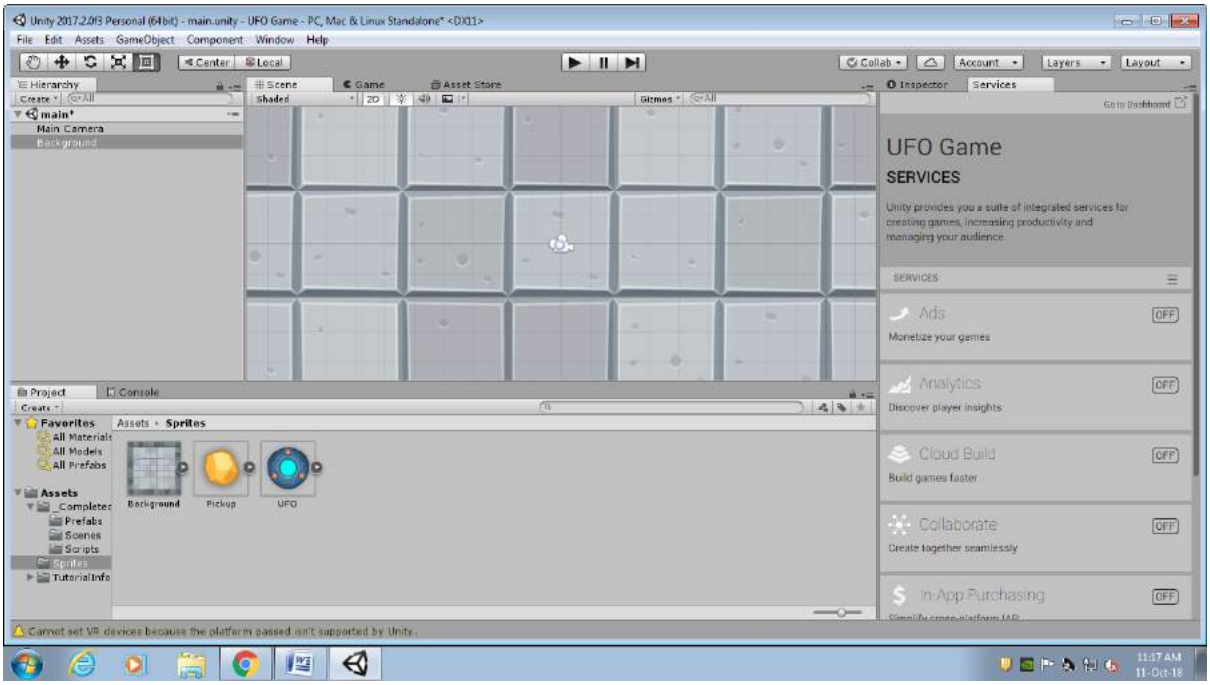


3} After import save the Scene in the folder(Scenes) and it name as Main. File -> Save Scene ->Scenes ->Save.

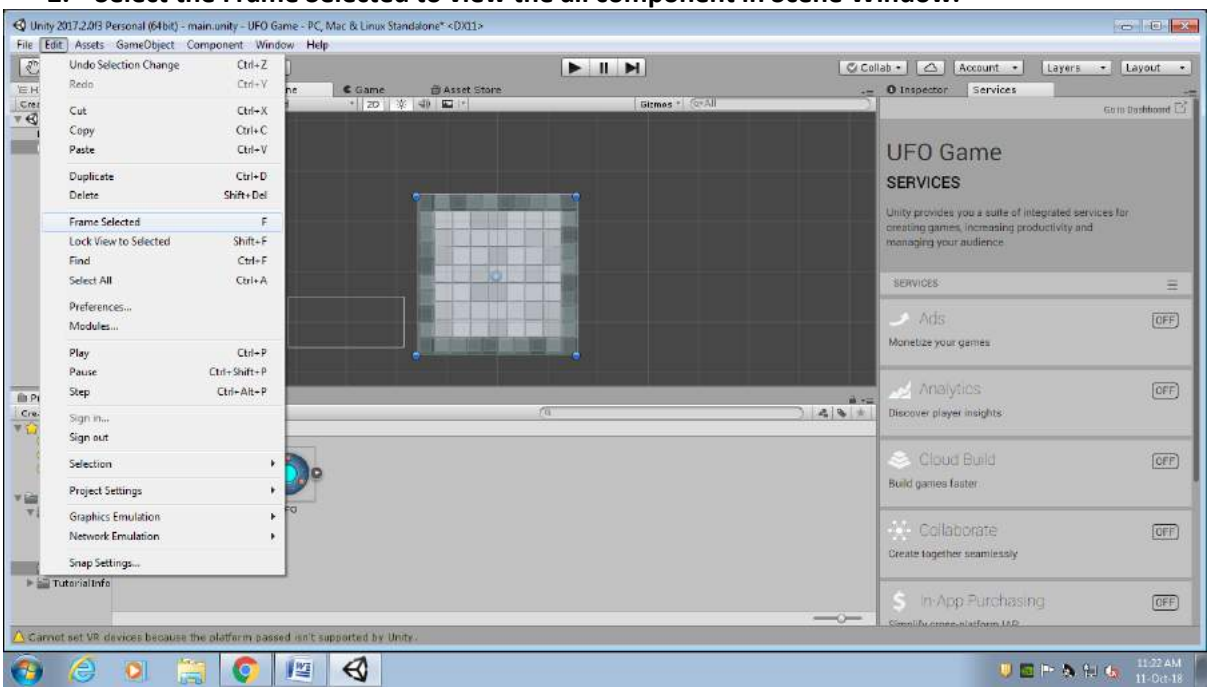


1. **Now Drag the background from the Project View into the Hirerarchy to create the game object called background.**

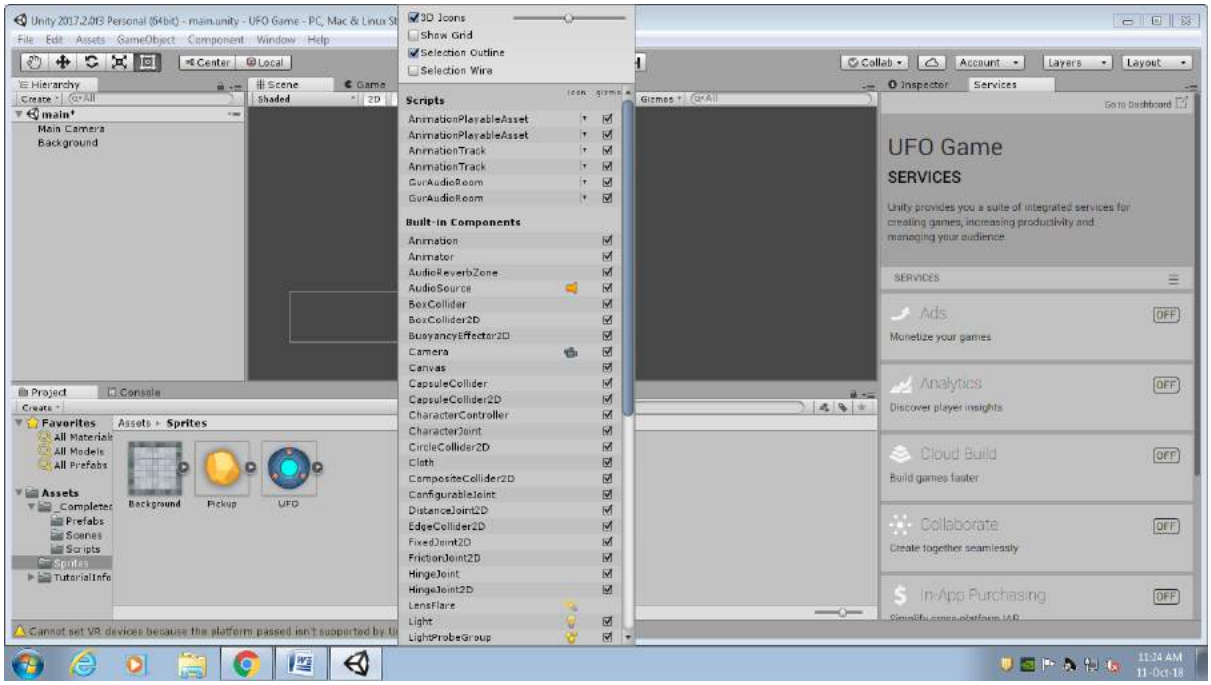




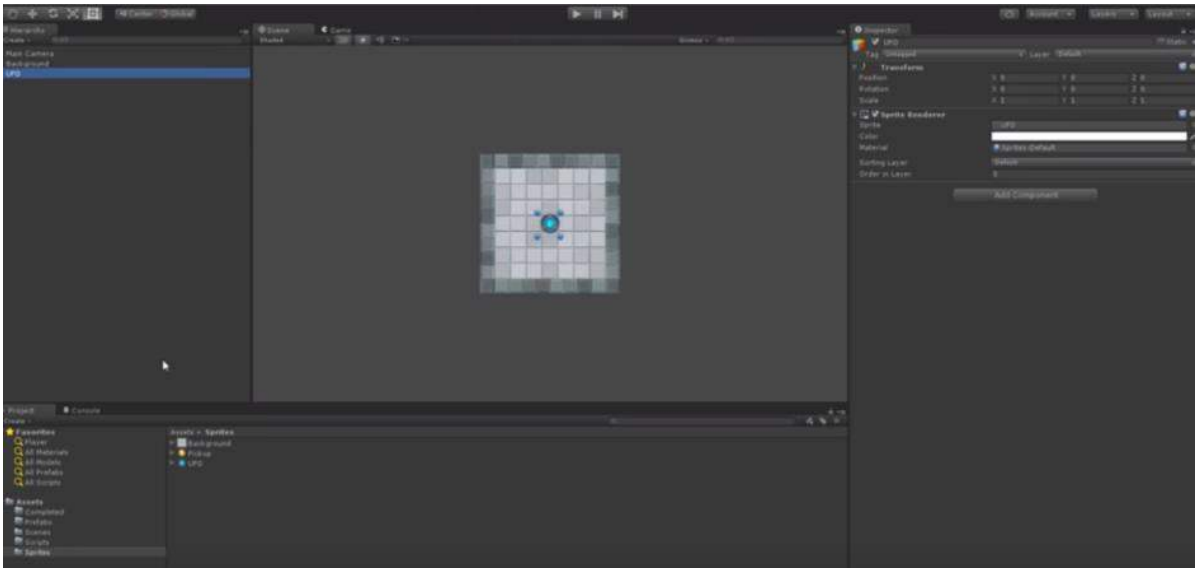
2. **Select the Frame Selected to view the all component in Scene Window.**

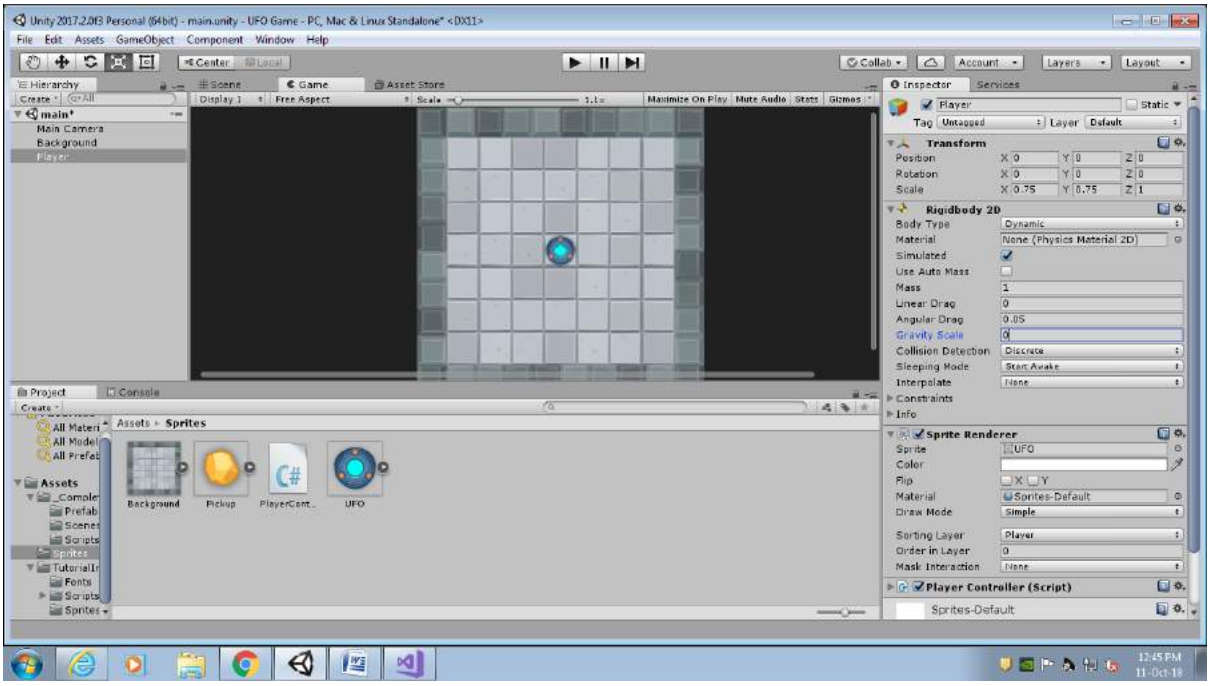


3. **Select the Gizmes menu and unchecked the “Show Grid”.**

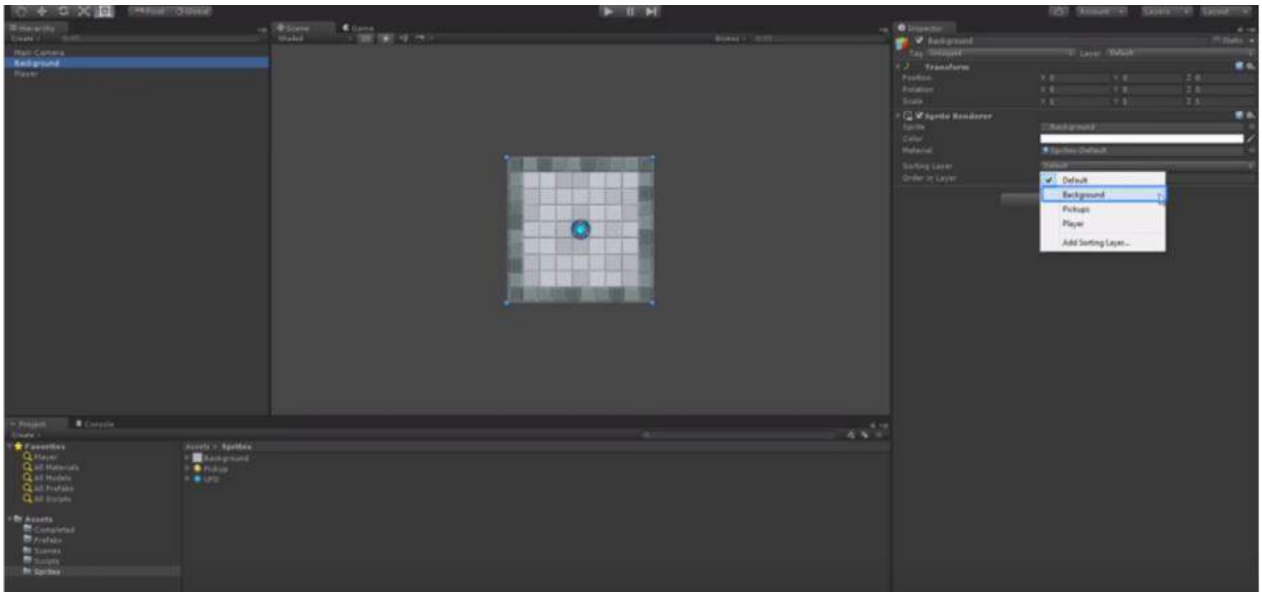


4. **Drag the UFO from the Sprite in the Hirerarchy and rename it to UFO to Player.**

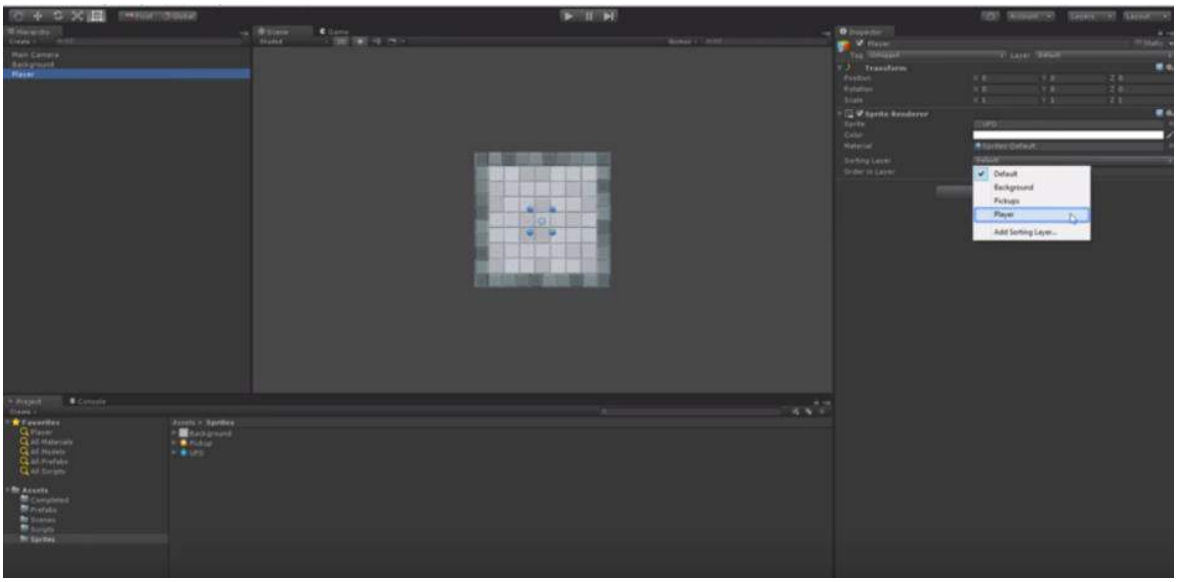




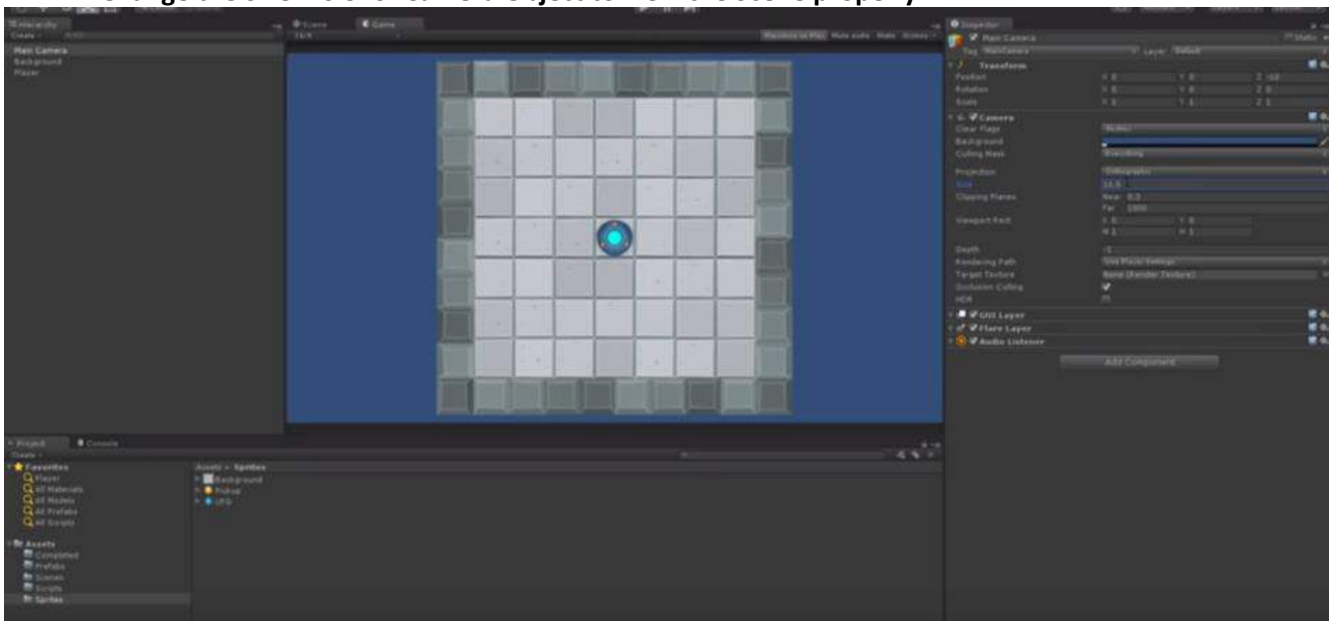
5. **Highlight Background object and Set the Sorting layer to the background from default.**



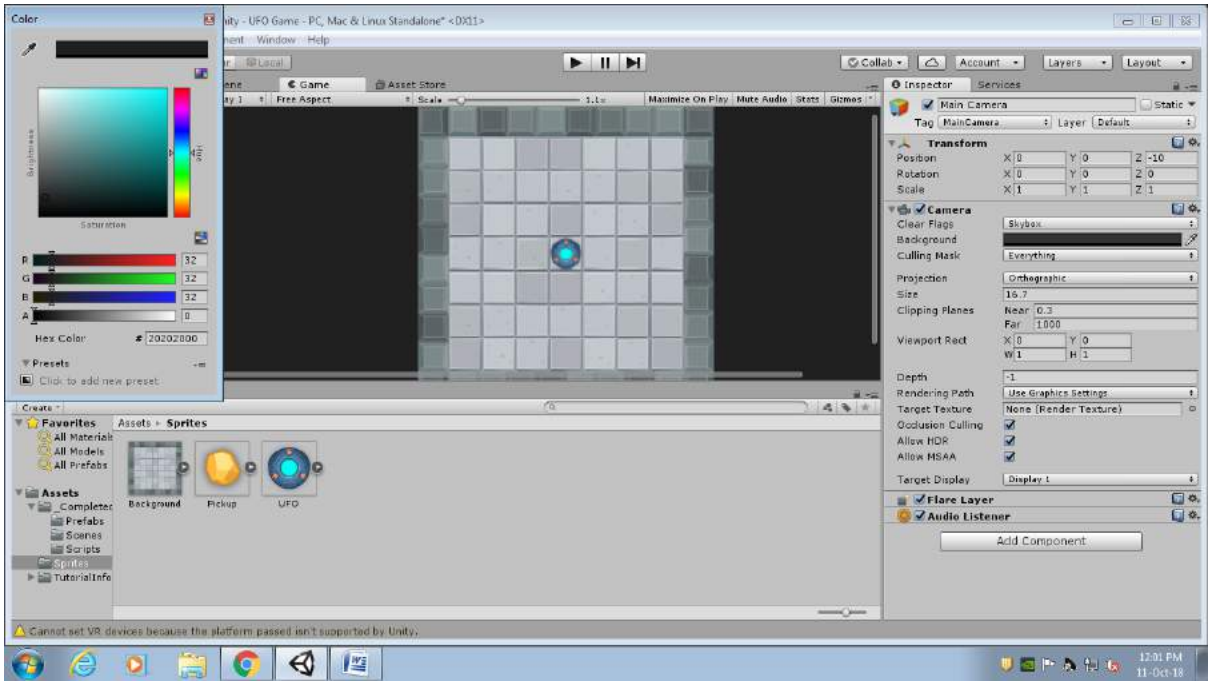
6. **Highlight Player object and Set the Sorting layer to the Player from default and Set the players scale as x=0.75 and y=o.75.**



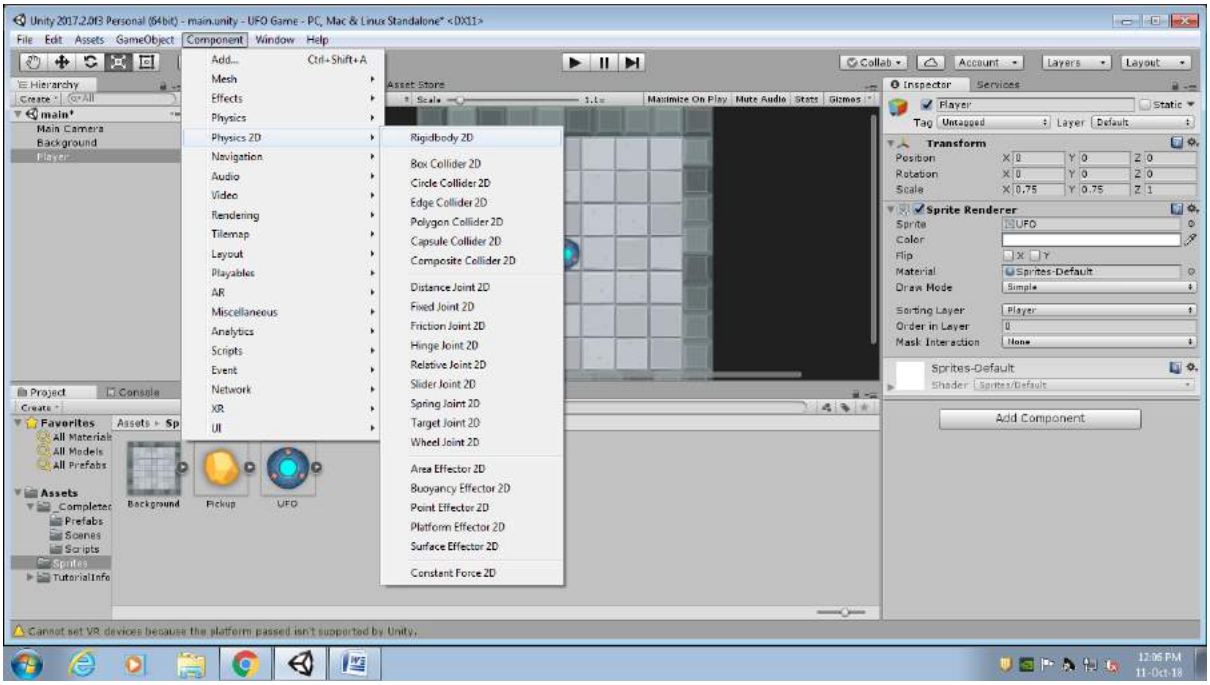
7. **Change the Size=16.5 of Camera Object to view the Scene properly.**



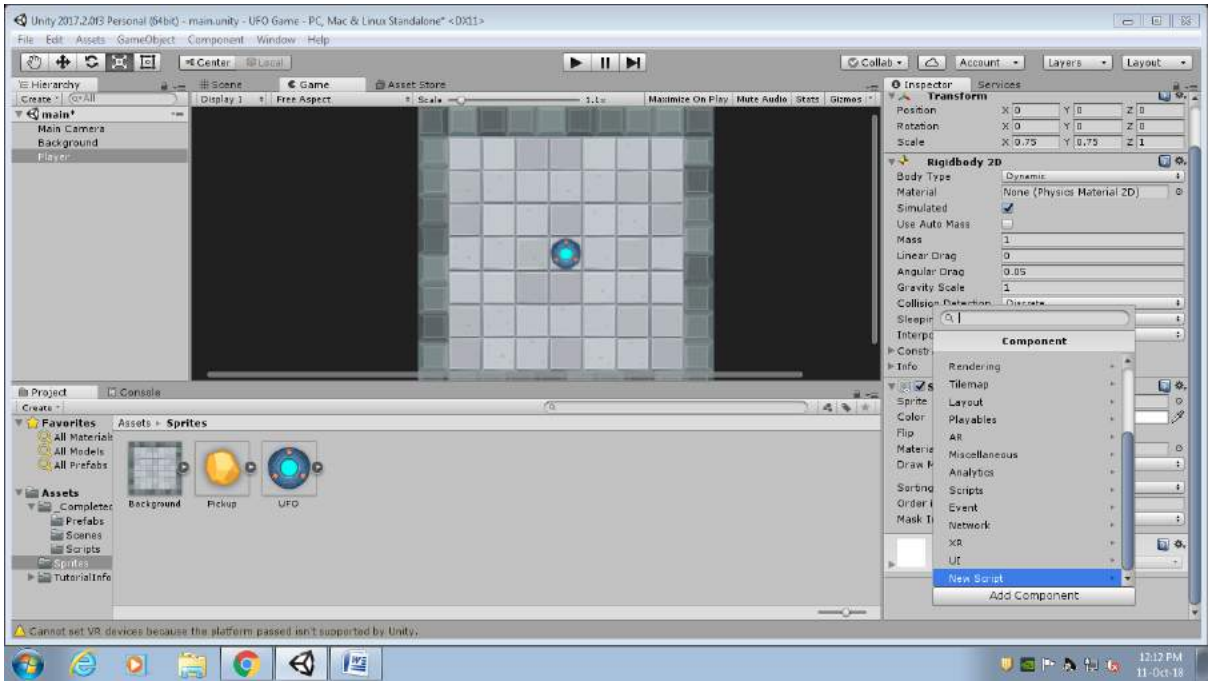
8. **Now the Background color in Camera Object to choose the “Color Picker”.**



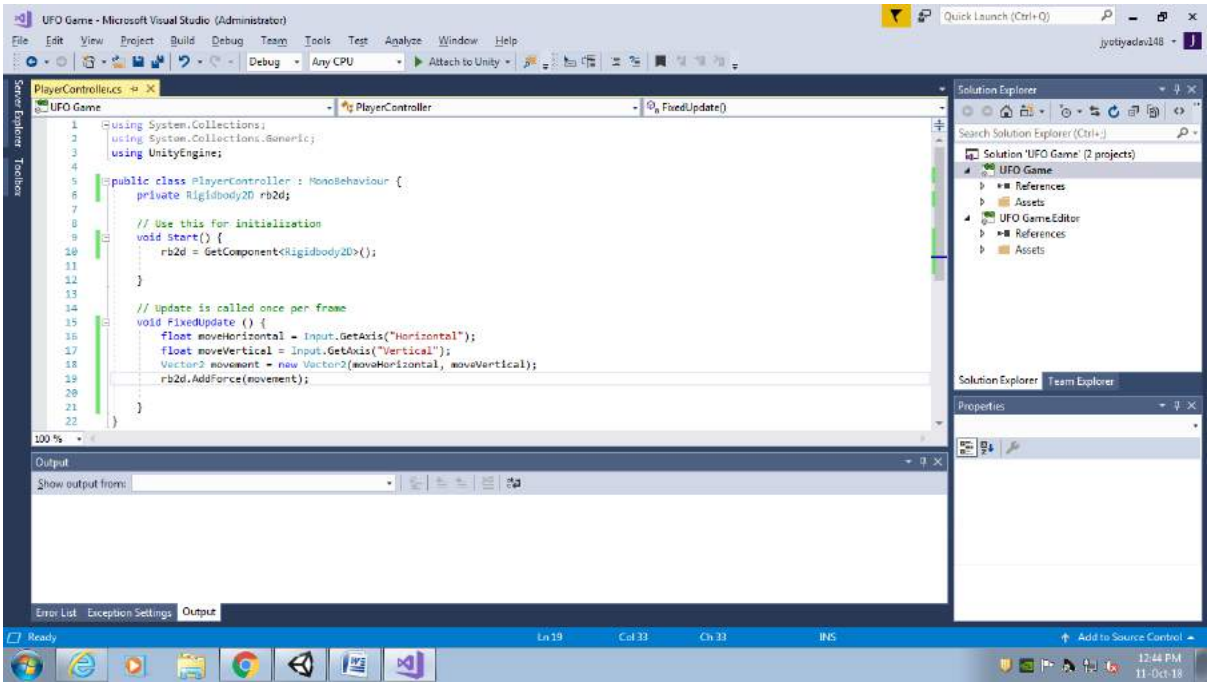
9. **Attach or Add a new component "RigidBody2D" on Player Object.**
   Component -> Physics2D -> RigidBody2D



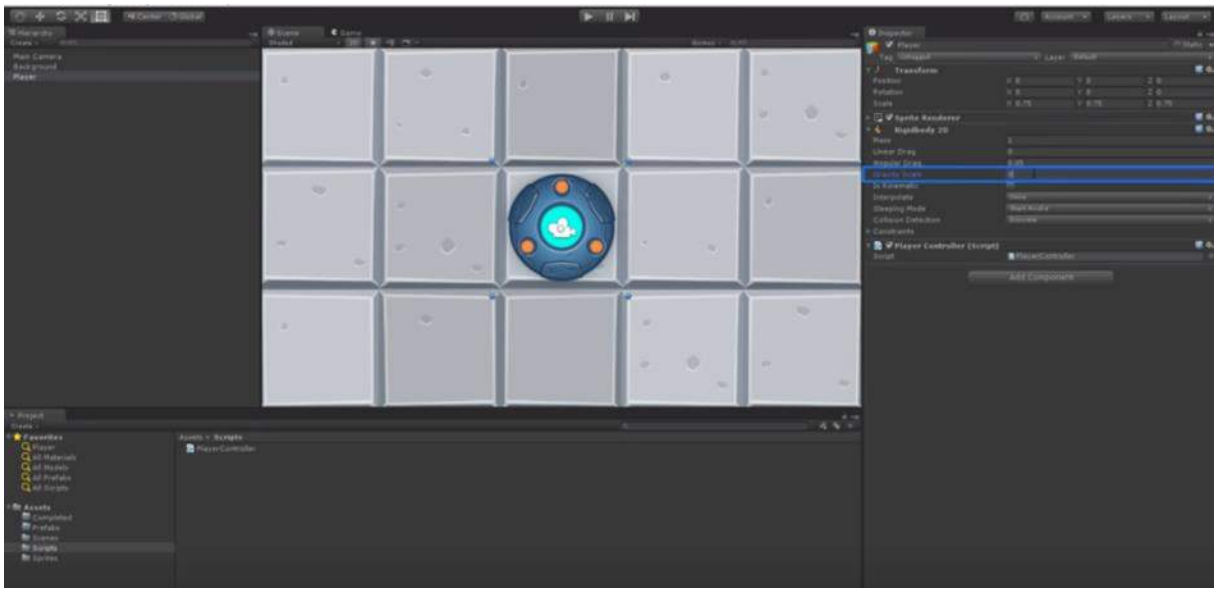
10. **For moving the player object under control Create a new C# Script as PlayerController and move code file of assets into the Script directory to maintain the project view.**
    Add Component -> New Script ->PlayerController -> Create and Add.



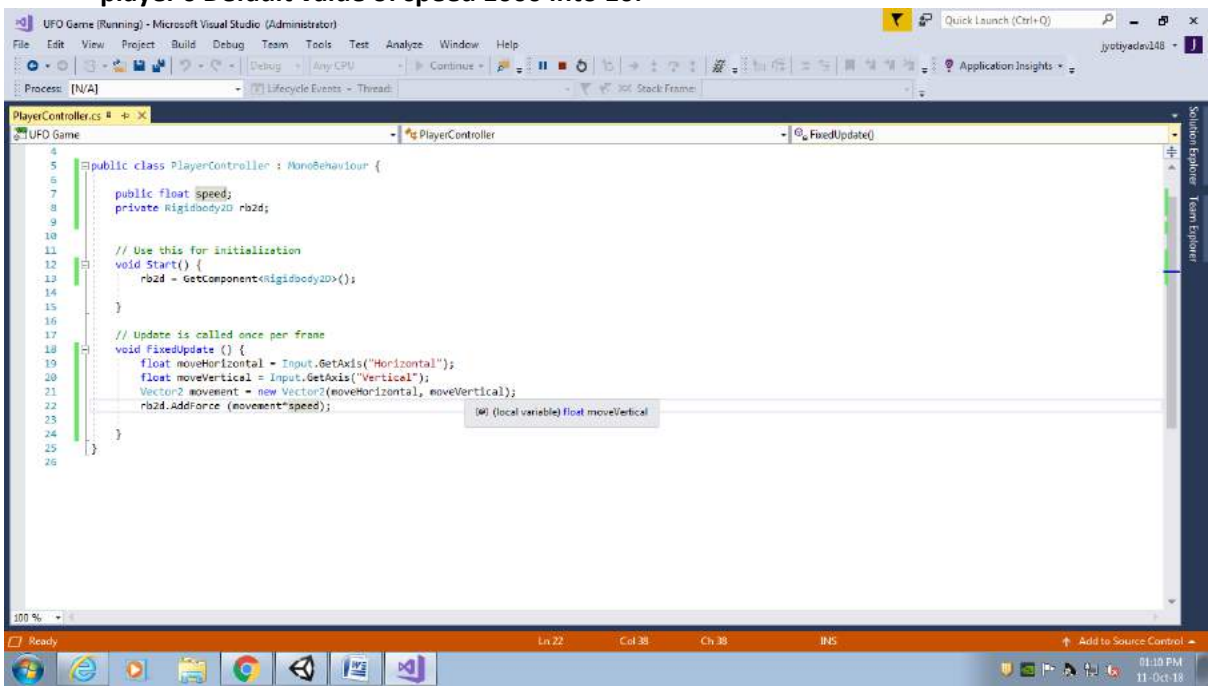
11. **Dobule Click on the PlayerController for editing or can select the player object and click on PlayerController and then choose the edit script from the Inspector.**
12. **Now Start the Coding in PlayerControllerSript and turn back to unity.**

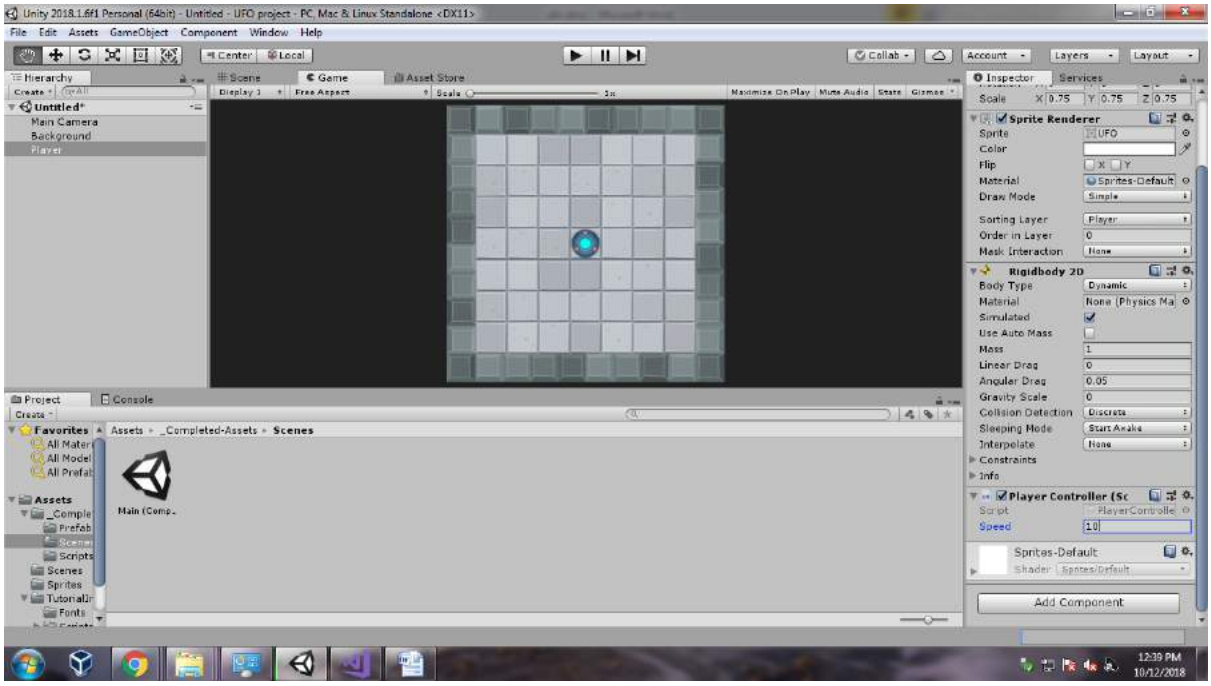


13. **Now set the Gravity Scale property (gravity scale=0) on RigidBody(Player).**

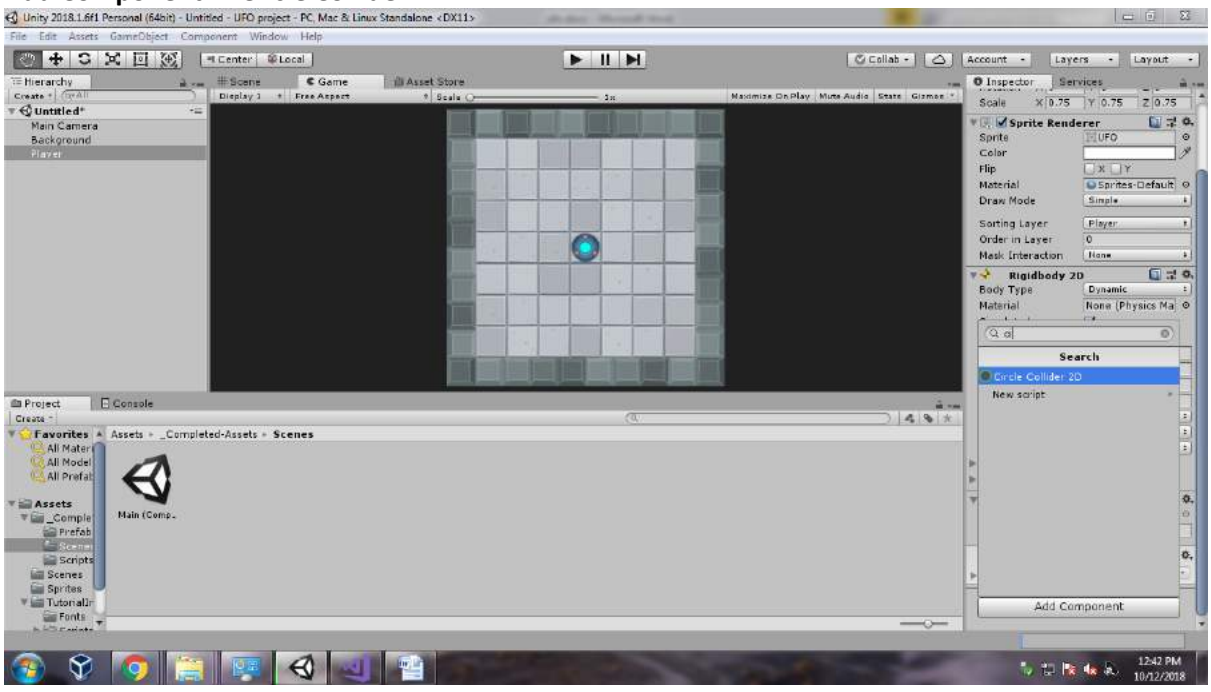


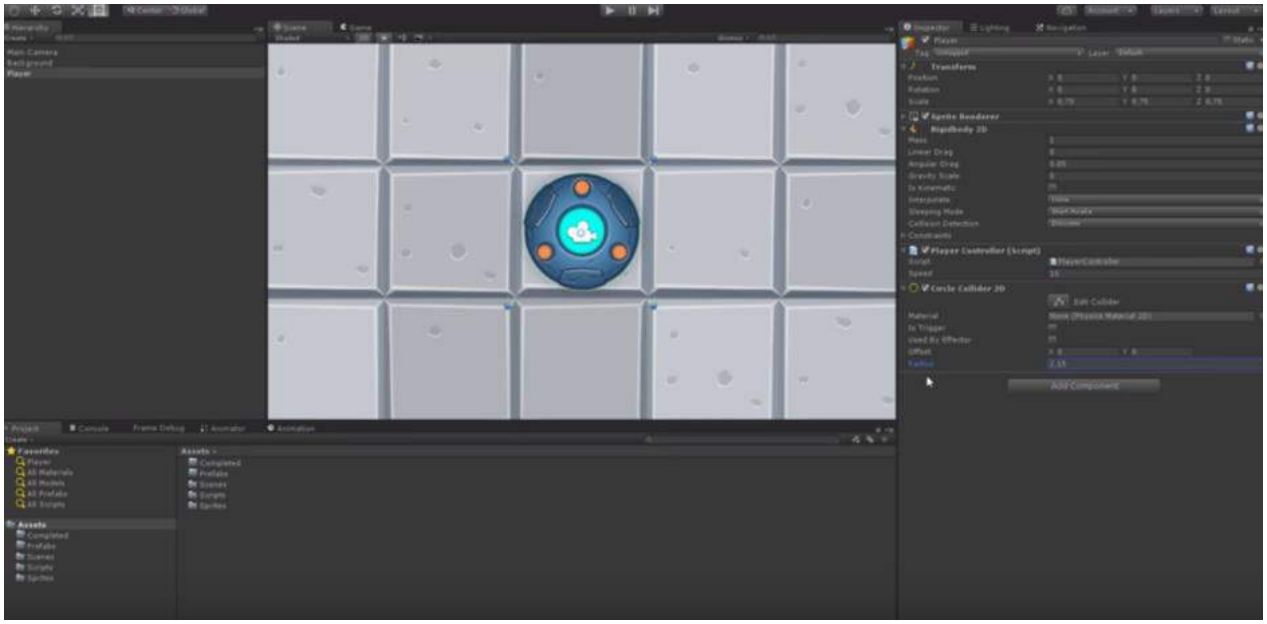
14. **Now for Speed Control includes some lines in PlayerController Script and Change the player's Default value of speed 1000 into 10.**





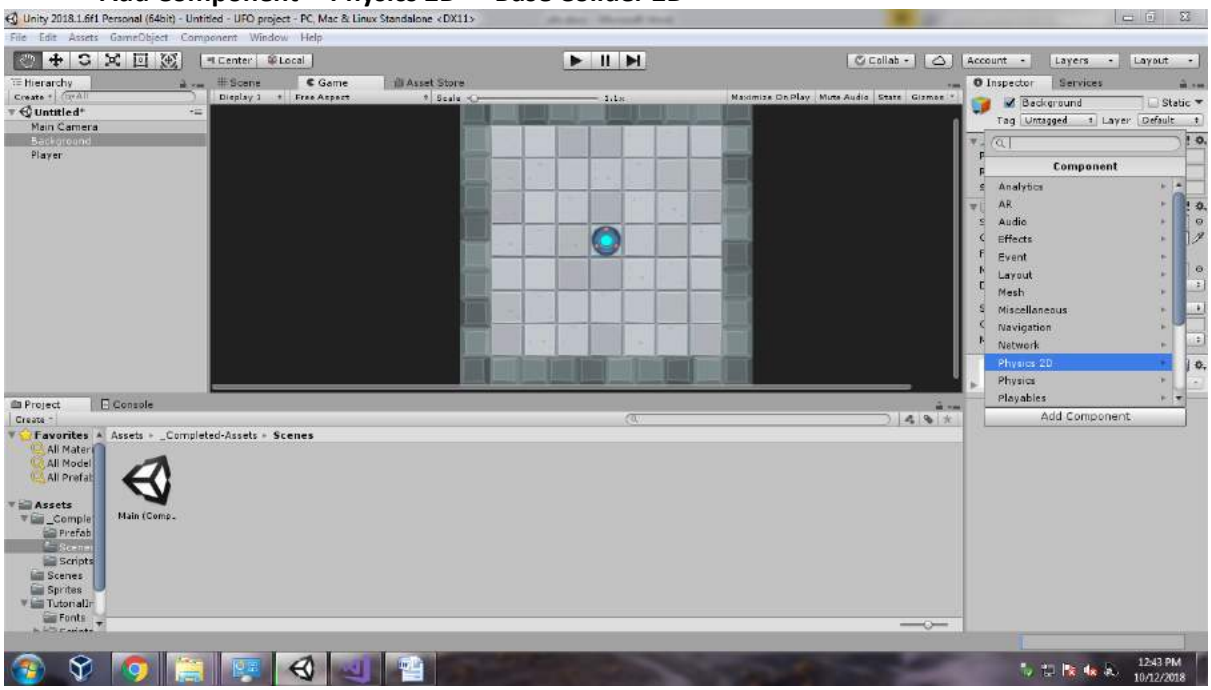
15. Add 2Dcollider on the Player for Collision detection and set the Radios of player as 2.15. Add Component -> Circle Collider 2D

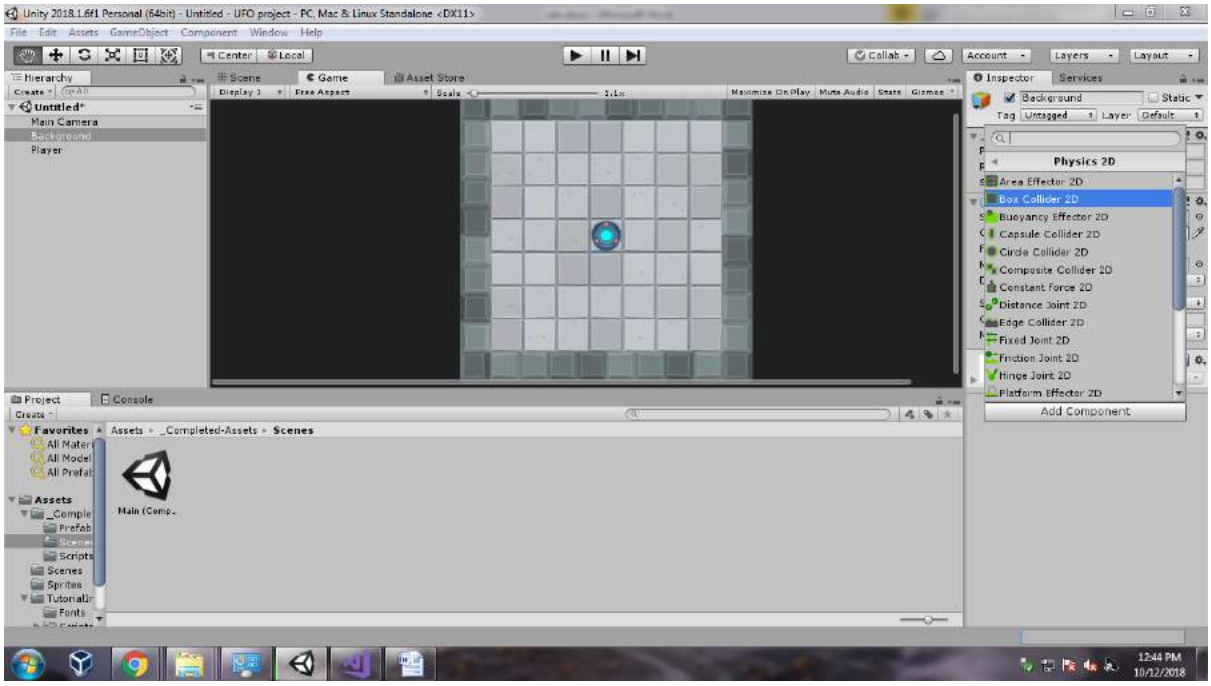




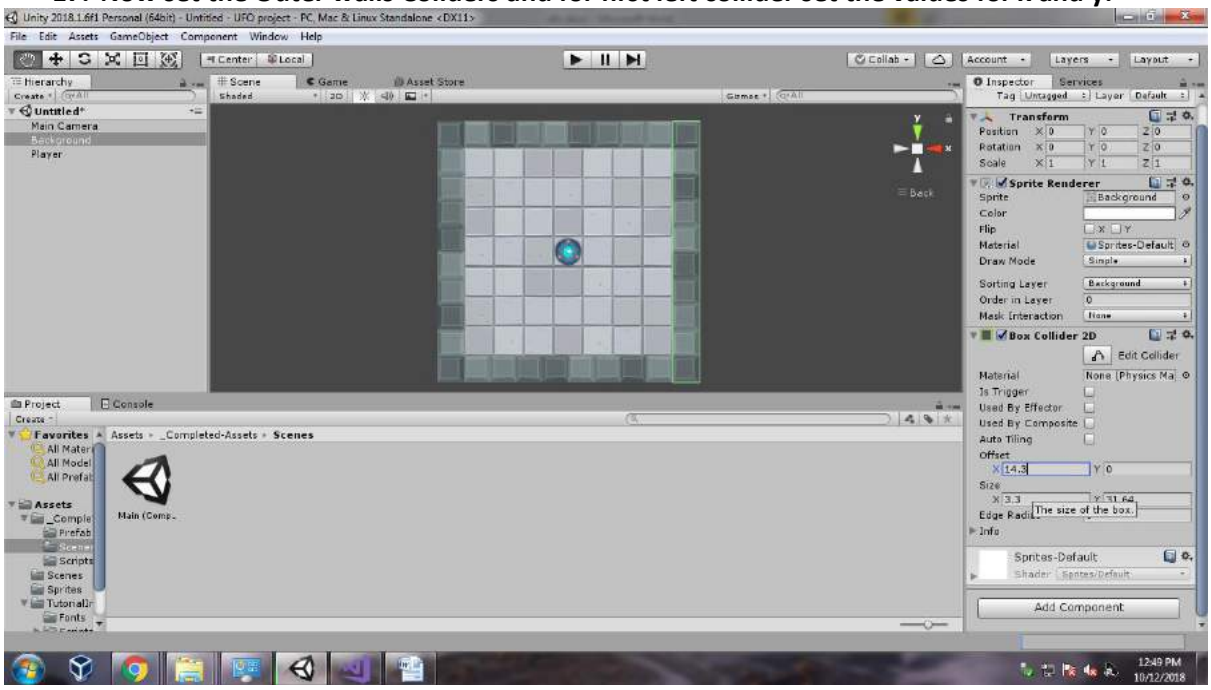
16. **Now set Collider on the Background walls for Collision detection and set the Radios of player as 2.15.**

    Add Component ->Physics 2D -> Base Colider 2D

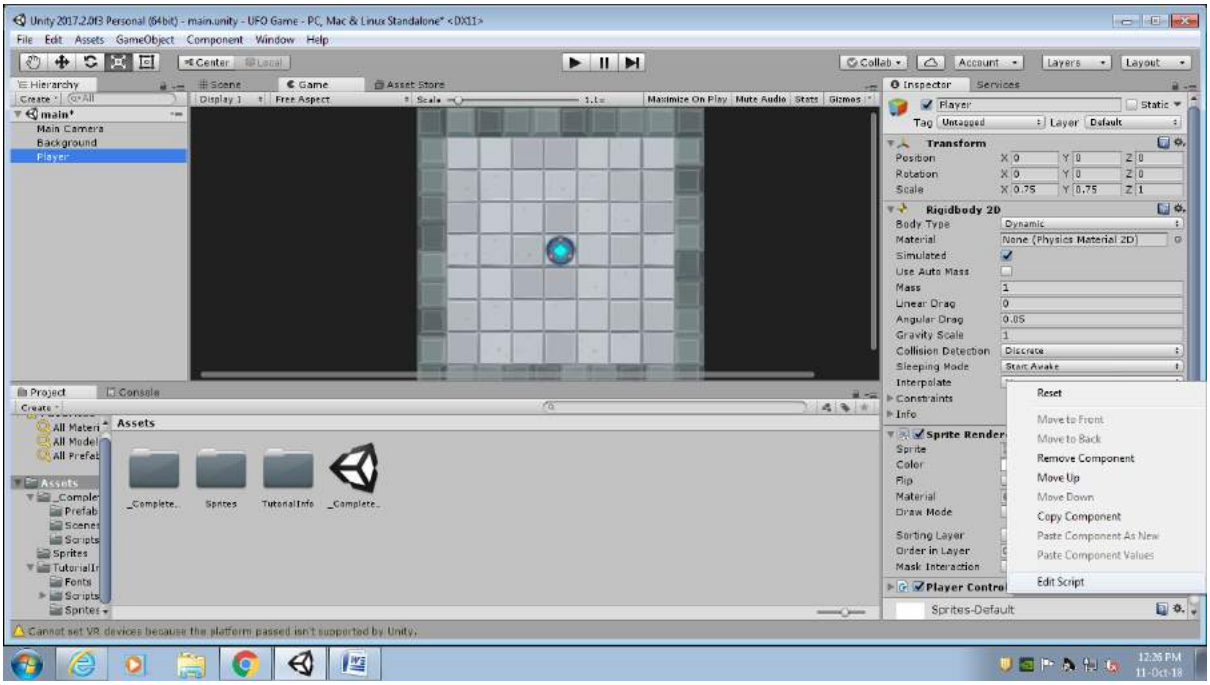


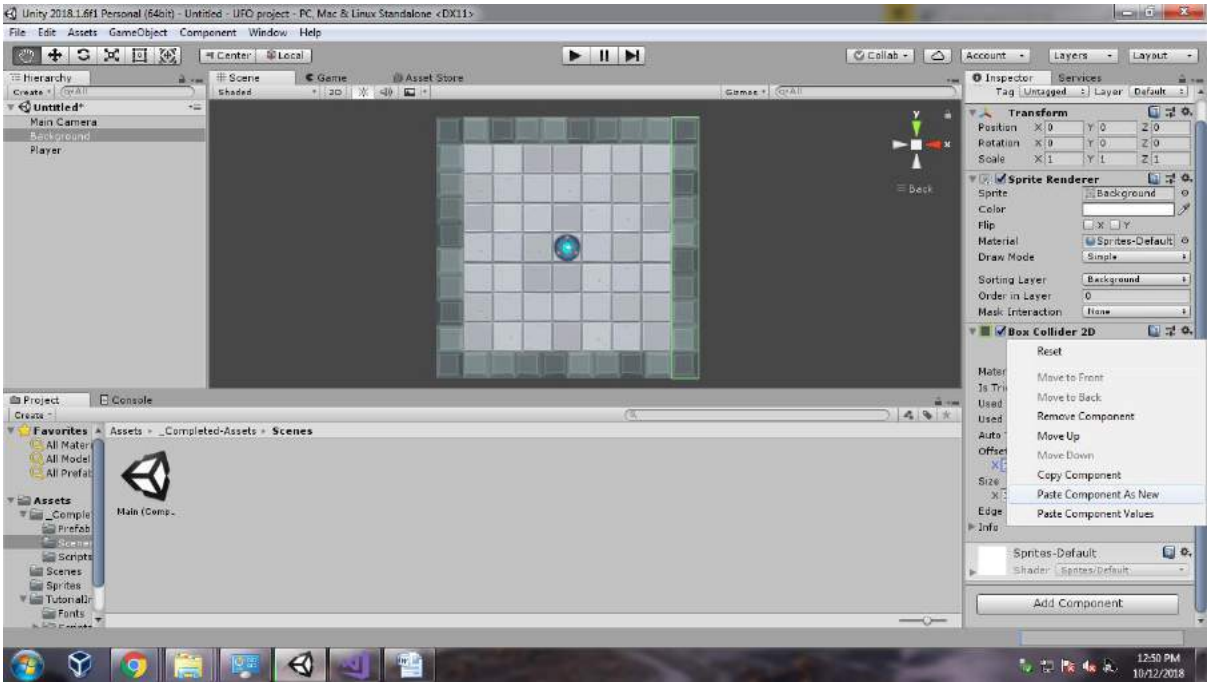


17. Now set the Outer walls Coliders and for first left collider set the values for x and y.

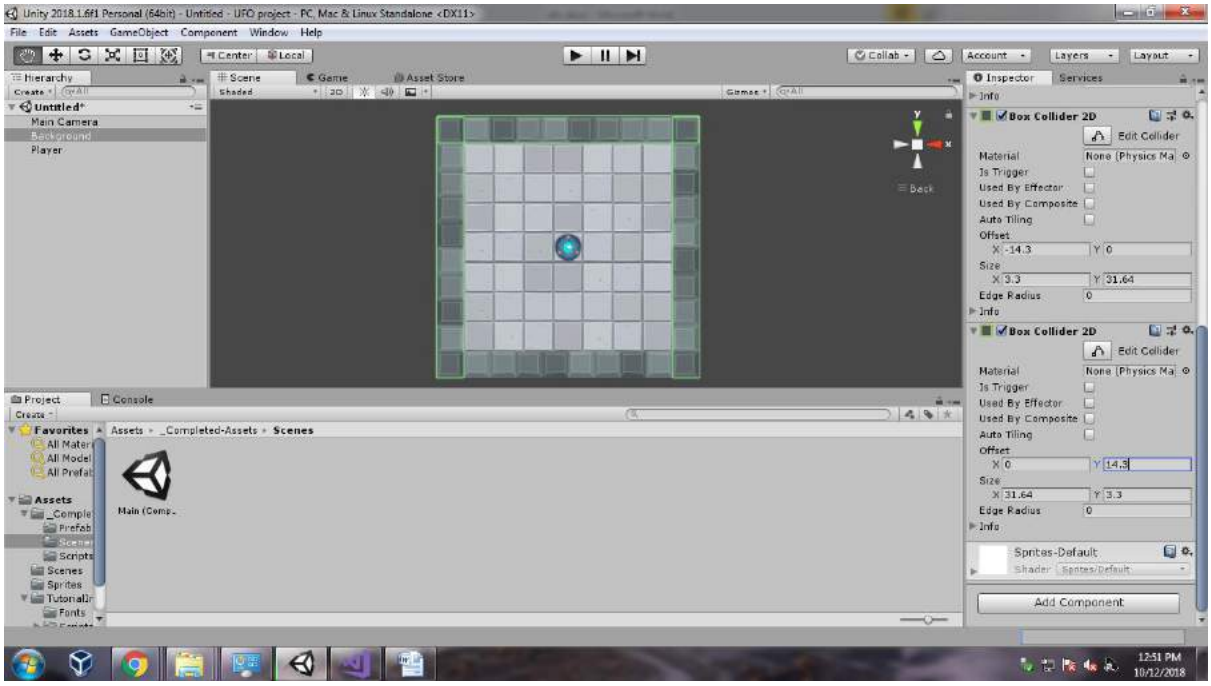


18. For Second collider copy the Box Collider 2D and Paste from the setting .

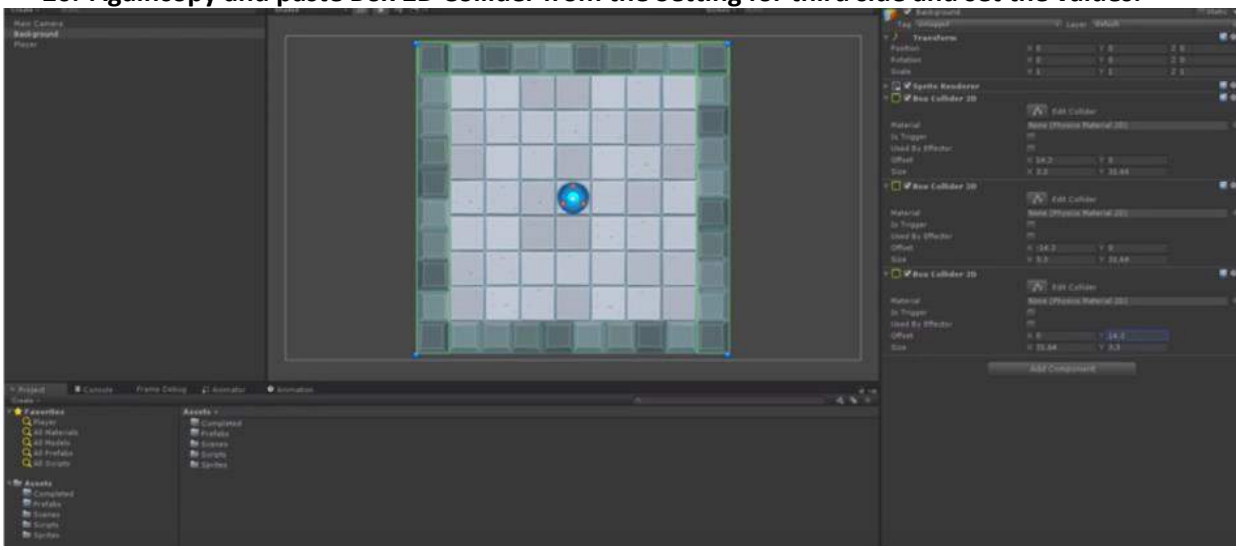




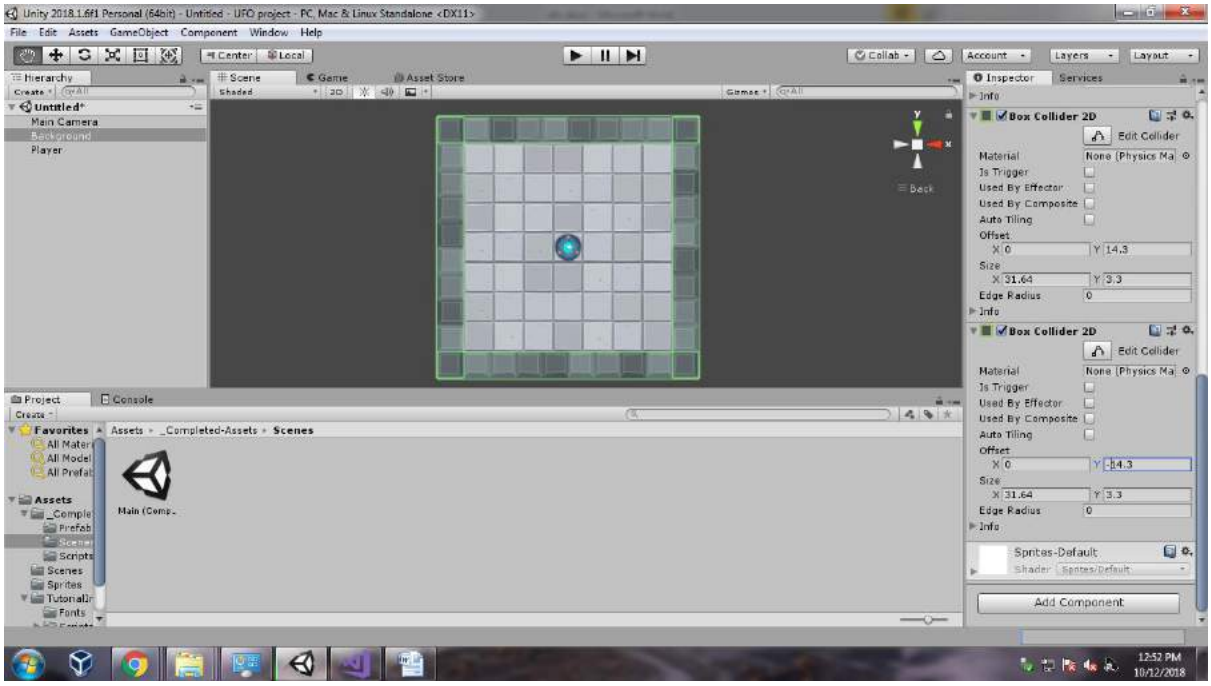
19. Now set the values of x and y for Second.



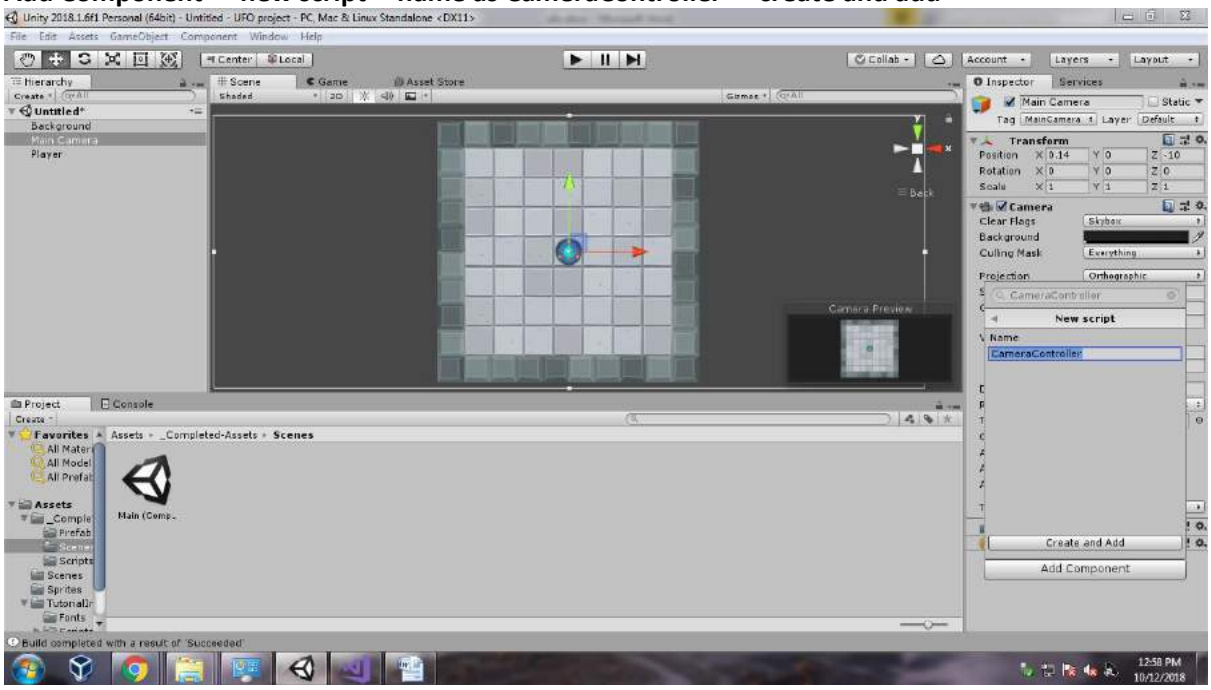
20. Againcopy and paste Box 2D Collider from the Setting for third side and set the values.



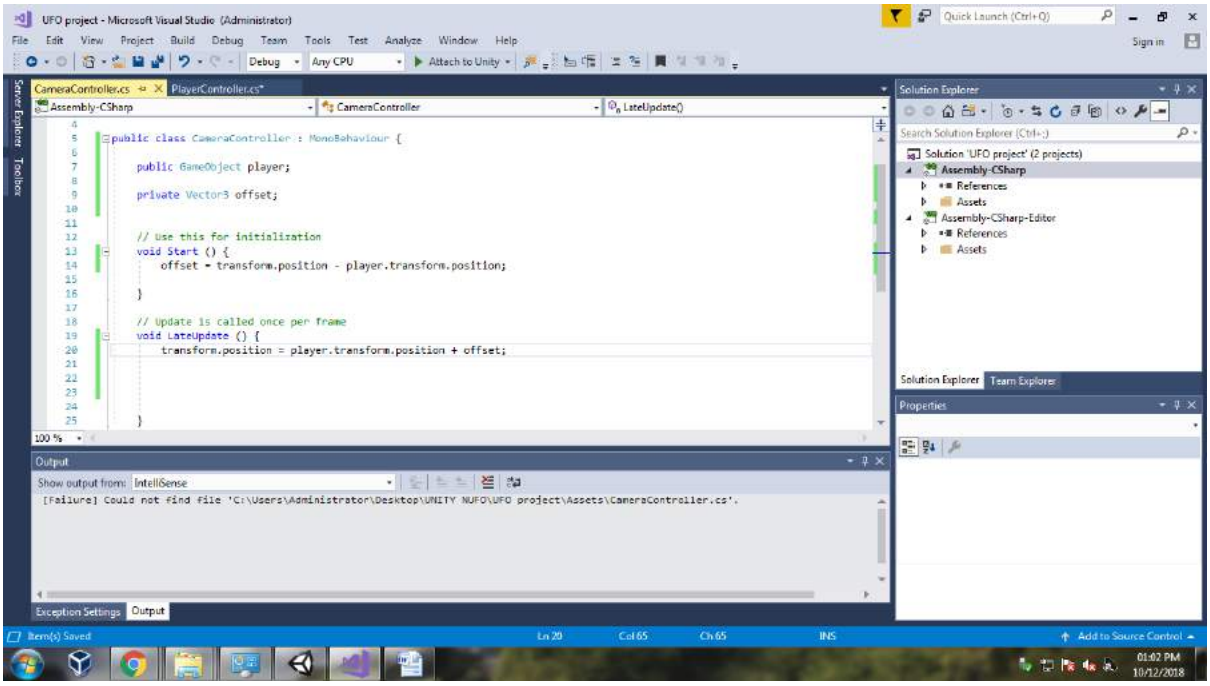
21. Again copy and paste Box 2D Collider from the Setting for last side and set the values.



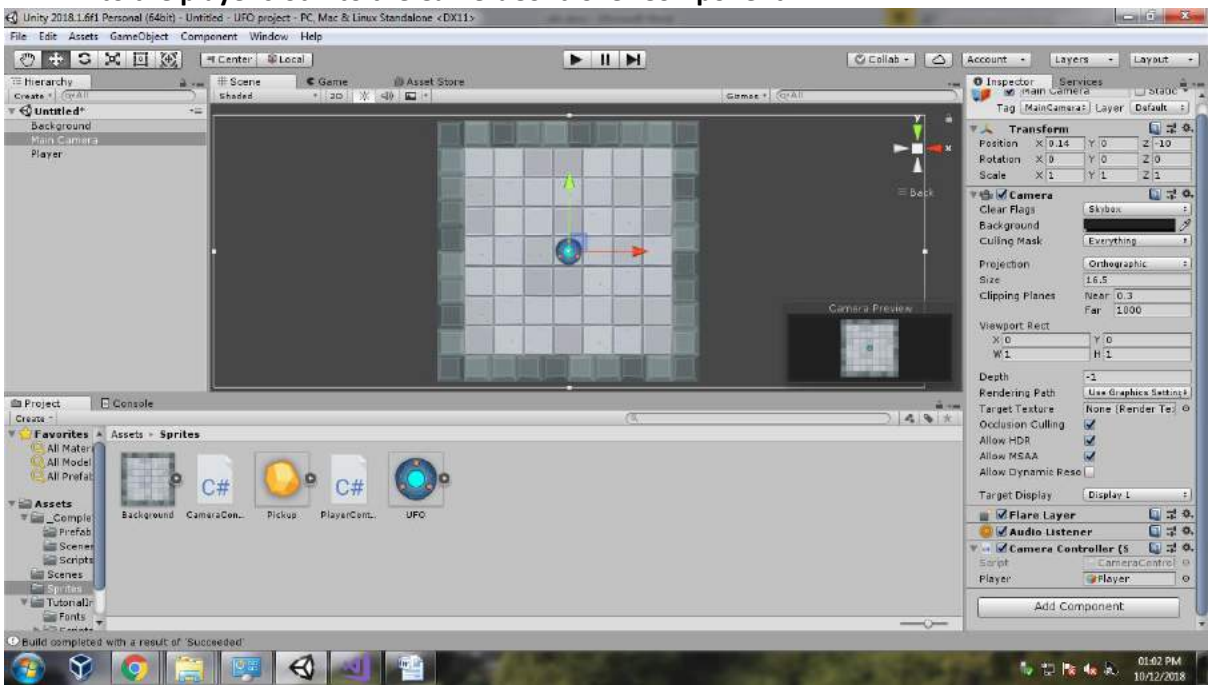
22. **To set the Camera's idle position for the project make a Script and drag it into the script. Add Component -> new script ->name as CameraController -> create and add**



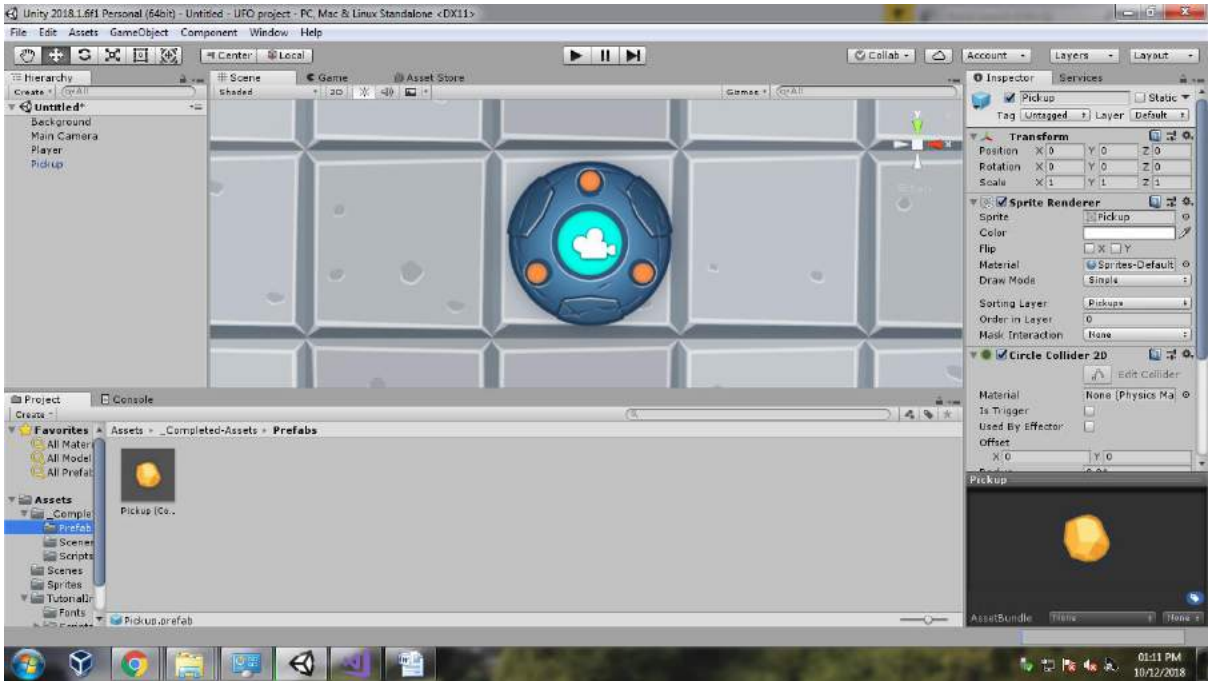
23. **Now Double click on CameraController Script and Start coding.**



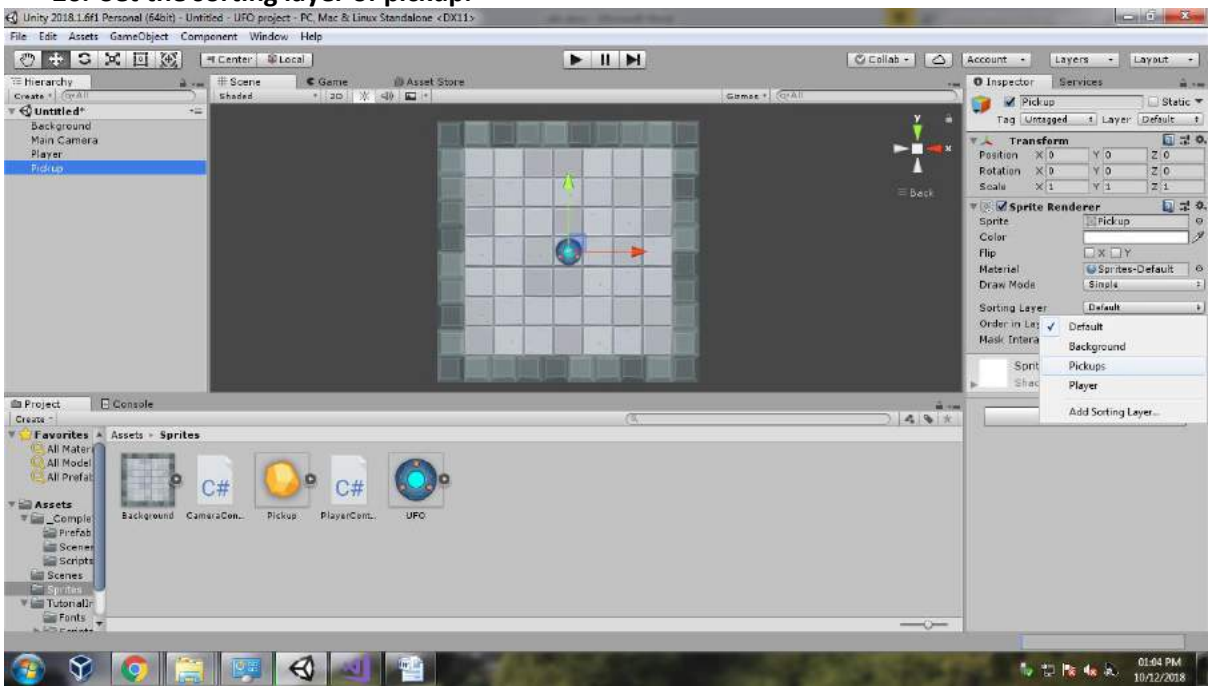
24. Now need to create the Reference of player object and for that drag the player object on to the player slot into the CameraController Component.



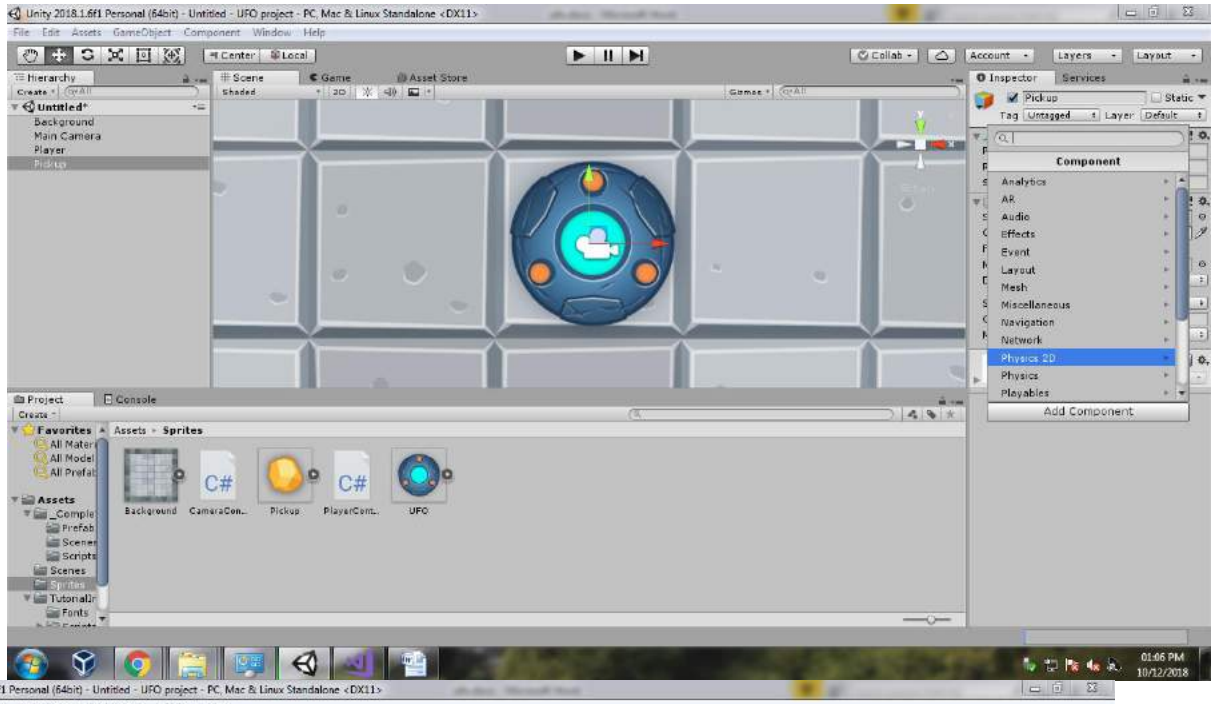
25. **Now drag the pickup from the sprites into the hierarchy**.

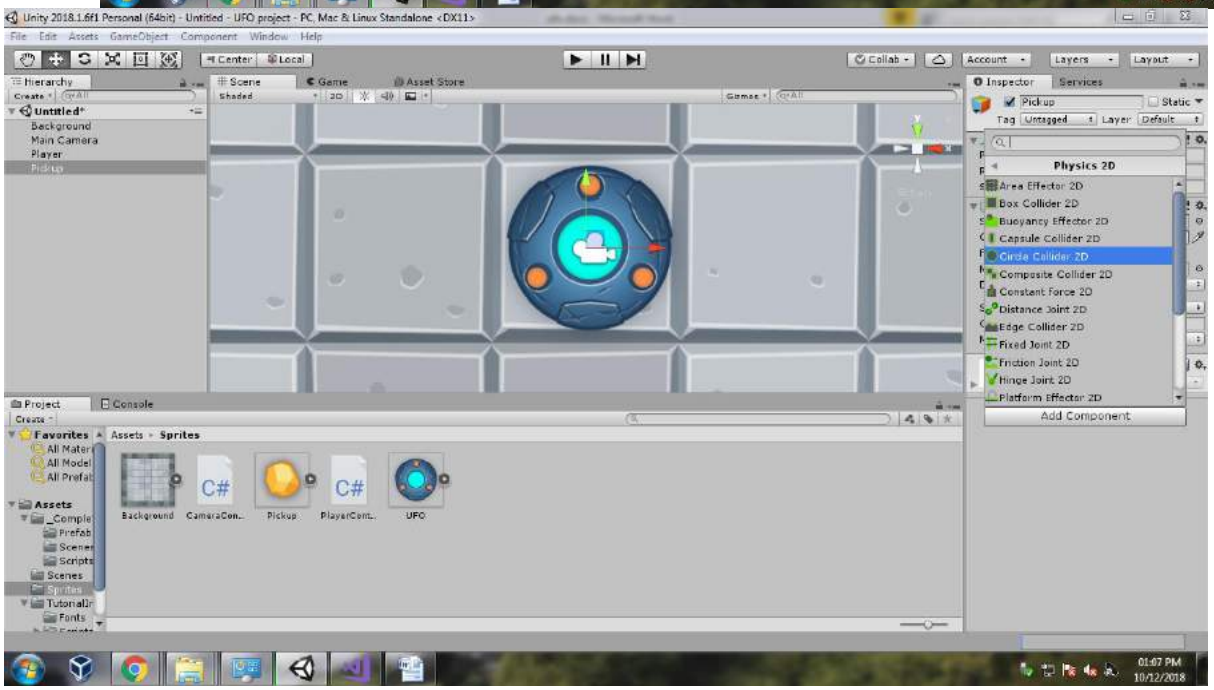


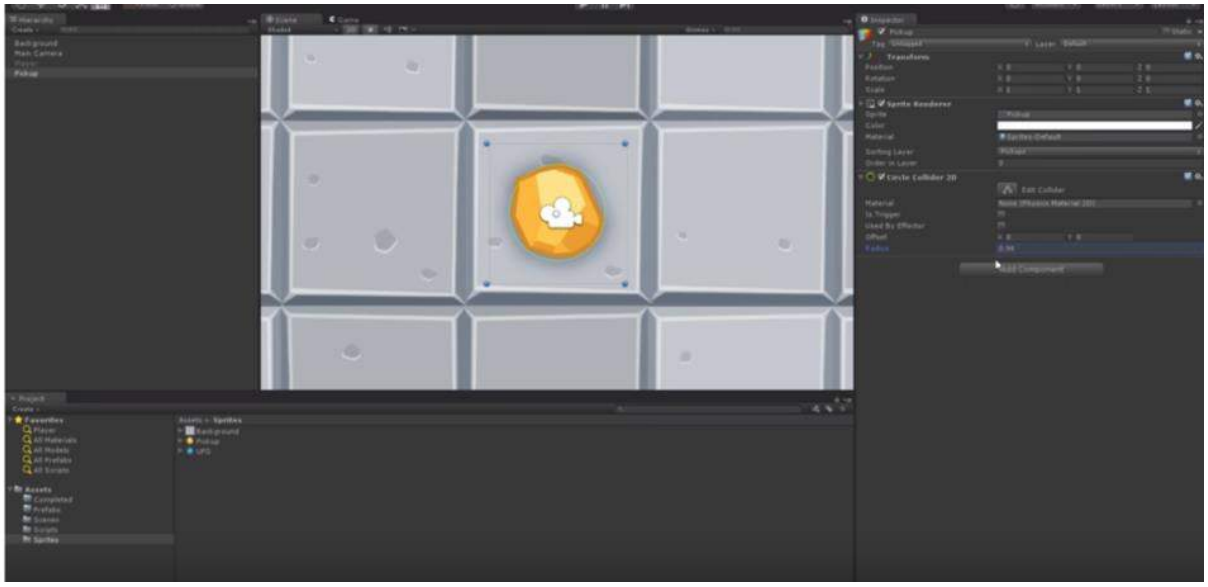
26. Set the sorting layer of pickup.



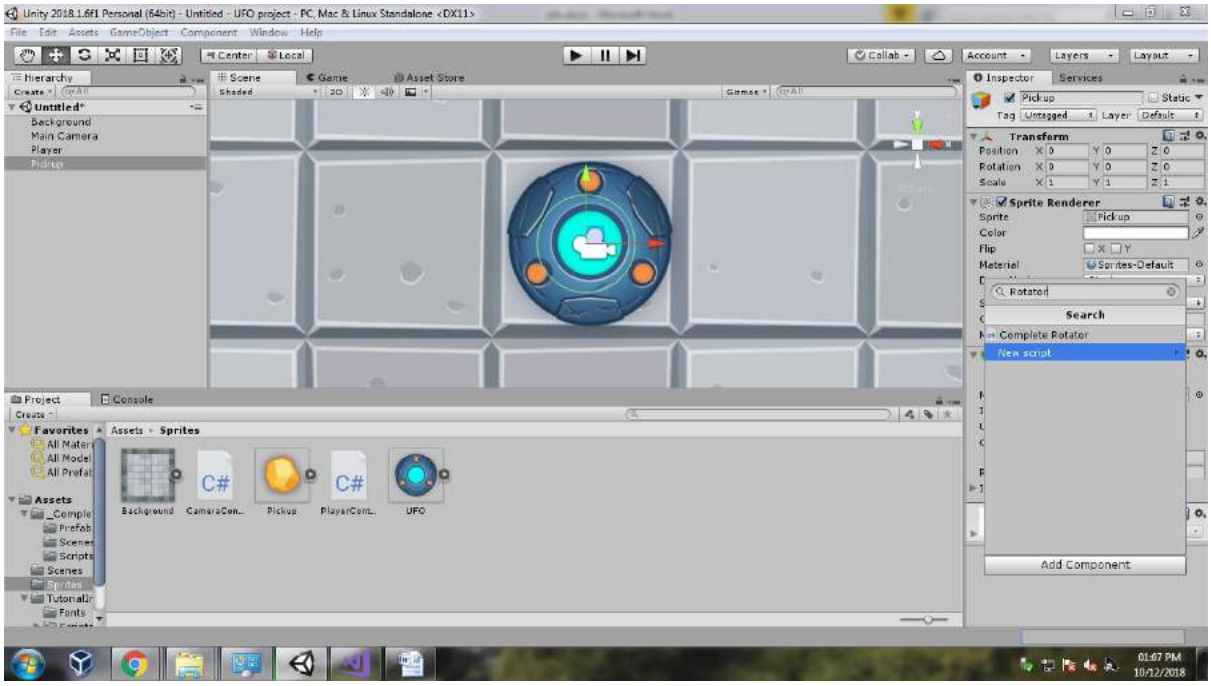
27. Add a circle collider 2D on pickup and set Radius=0.94 of the pickup for movement.
Add Component -> Physics 2D -> Circle Collider 2D -> Create and Add.


Unity 2018.1.6f1 Personal (64bit) - Untitled - UFO project - PC, Mac & Linux Standalone <DX11>
Hierarchy
Untitled*
Background
Main Camera
Player
Pickup
Scene
Game
Asset Store
Inspector
Services
Component
Analytics
AR
Audio
Effects
Event
Layout
Mesh
Miscellaneous
Navigation
Network
Physics 2D
Physics
Playables
Add Component
Project
Console
Favorites
Assets
Sprites
Background
CameraCon..
Pickup
PlayerCont..
UFO
01:06 PM
10/12/2018


Unity 2018.1.6f1 Personal (64bit) - Untitled - UFO project - PC, Mac & Linux Standalone <DX11>
Hierarchy
Untitled*
Background
Main Camera
Player
Pickup
Scene
Game
Asset Store
Inspector
Services
Physics 2D
Area Effector 2D
Box Collider 2D
Buoyancy Effector 2D
Capsule Collider 2D
Circle Collider 2D
Composite Collider 2D
Constant Force 2D
Distance Joint 2D
Edge Collider 2D
Fixed Joint 2D
Friction Joint 2D
Hinge Joint 2D
Platform Effector 2D
Add Component
Project
Console
Favorites
Assets
Sprites
Background
CameraCon..
Pickup
PlayerCont..
UFO
01:07 PM
10/12/2018

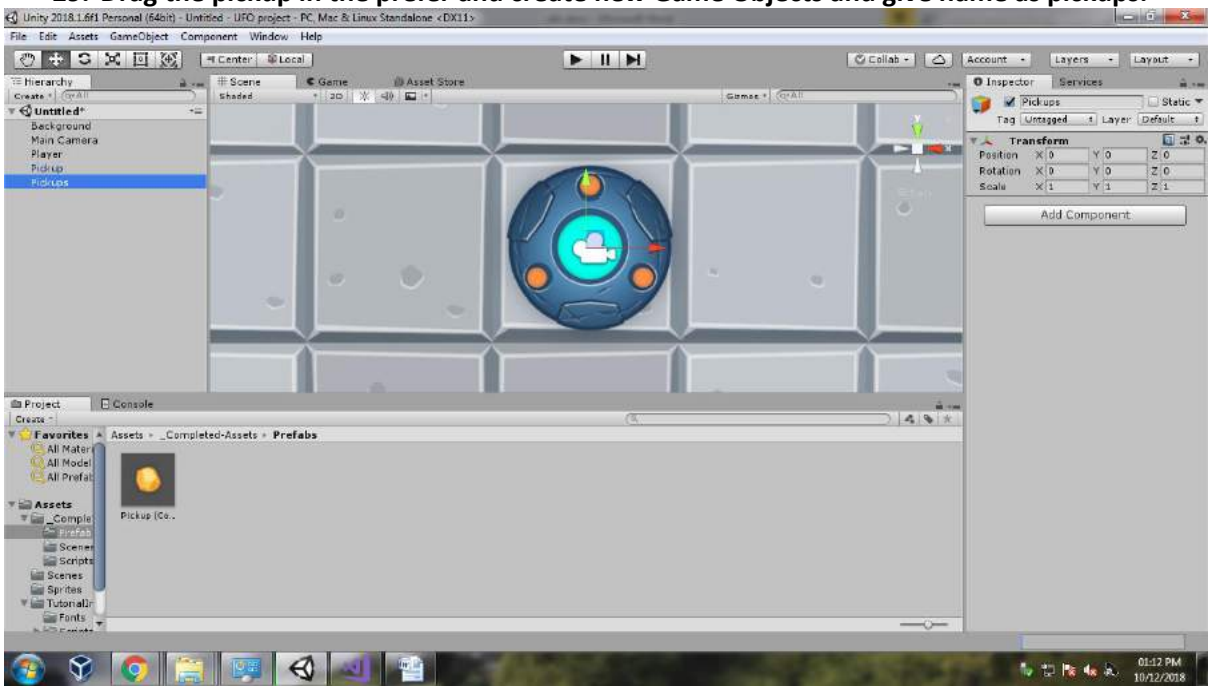


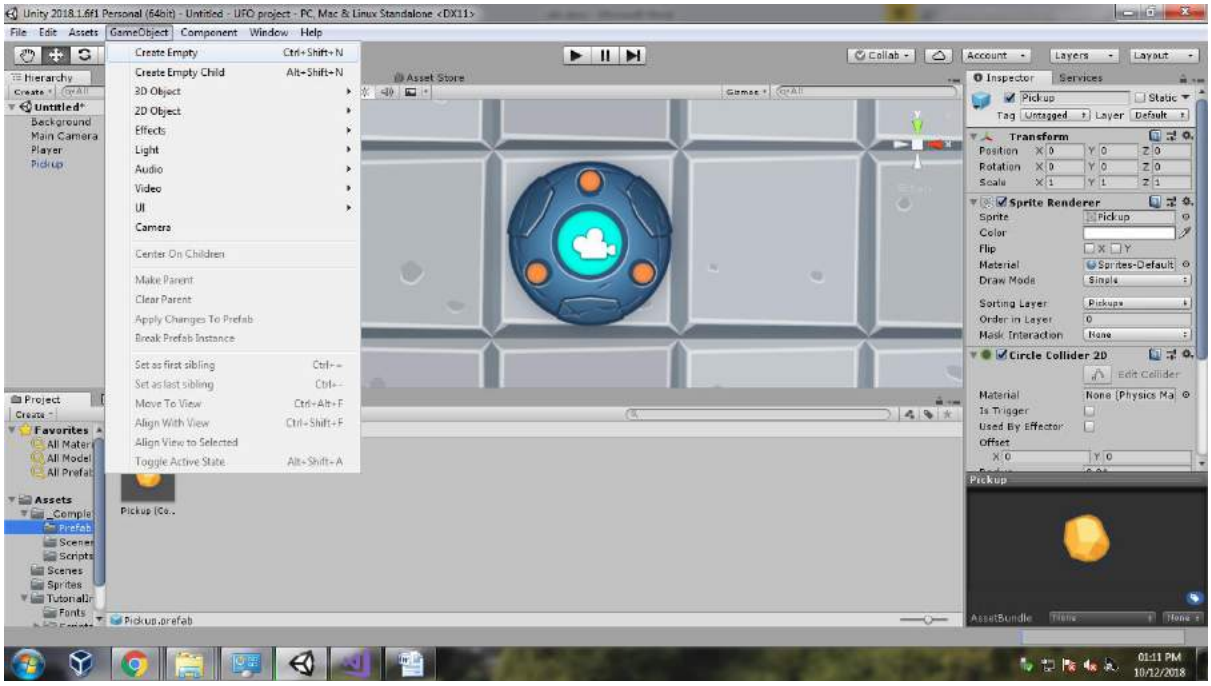
28. **Next Create new Script called Rotator and start coding.**



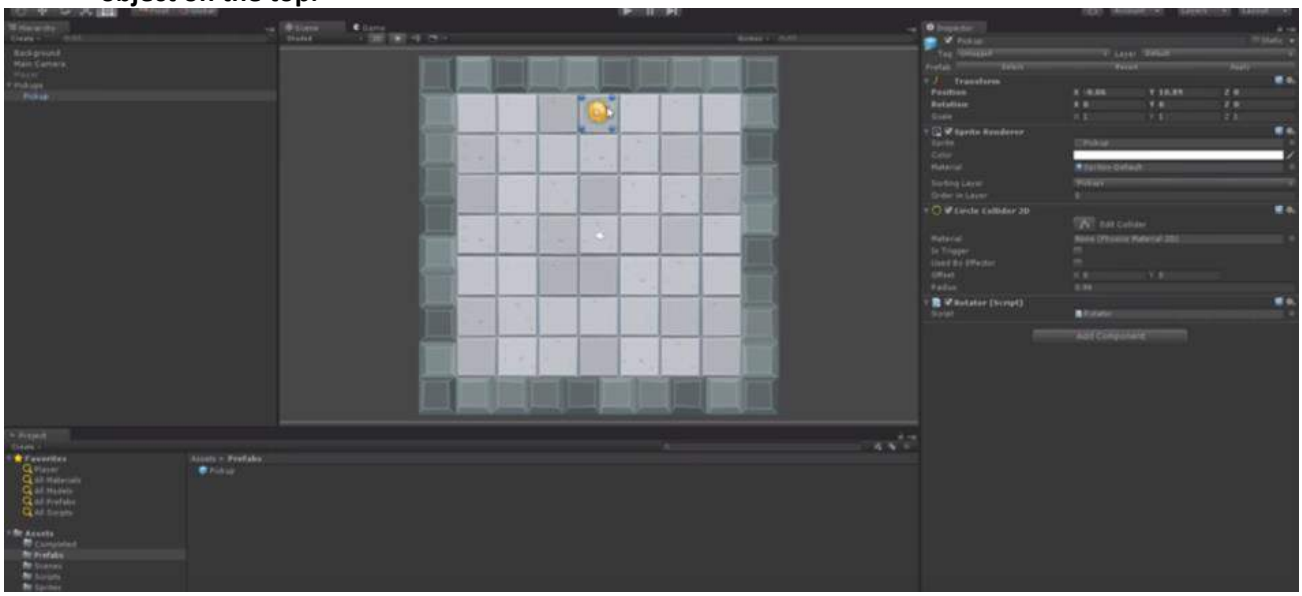
```
Rotator ▸ Update ()
1 using UnityEngine;
2 using System.Collections;
3
4 public class Rotator : MonoBehaviour {
5
6
7     void Update ()
8     {
9         transform.Rotate (new Vector3 (0, 0, 45) * Time.deltaTime);
10    }
11 }
12
```

29. Drag the pickup in the prefer and create new Game Objects and give name as pickups.

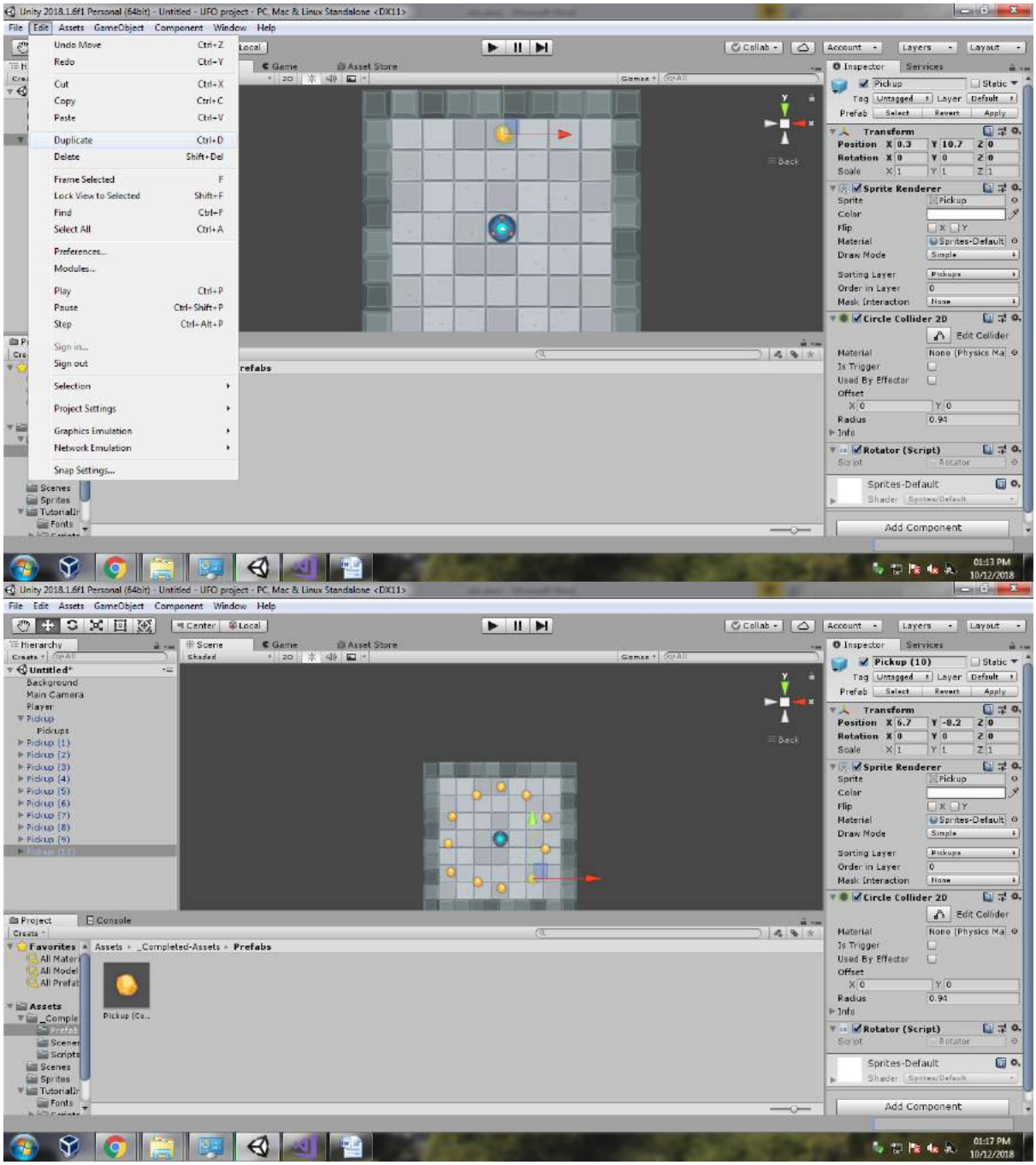




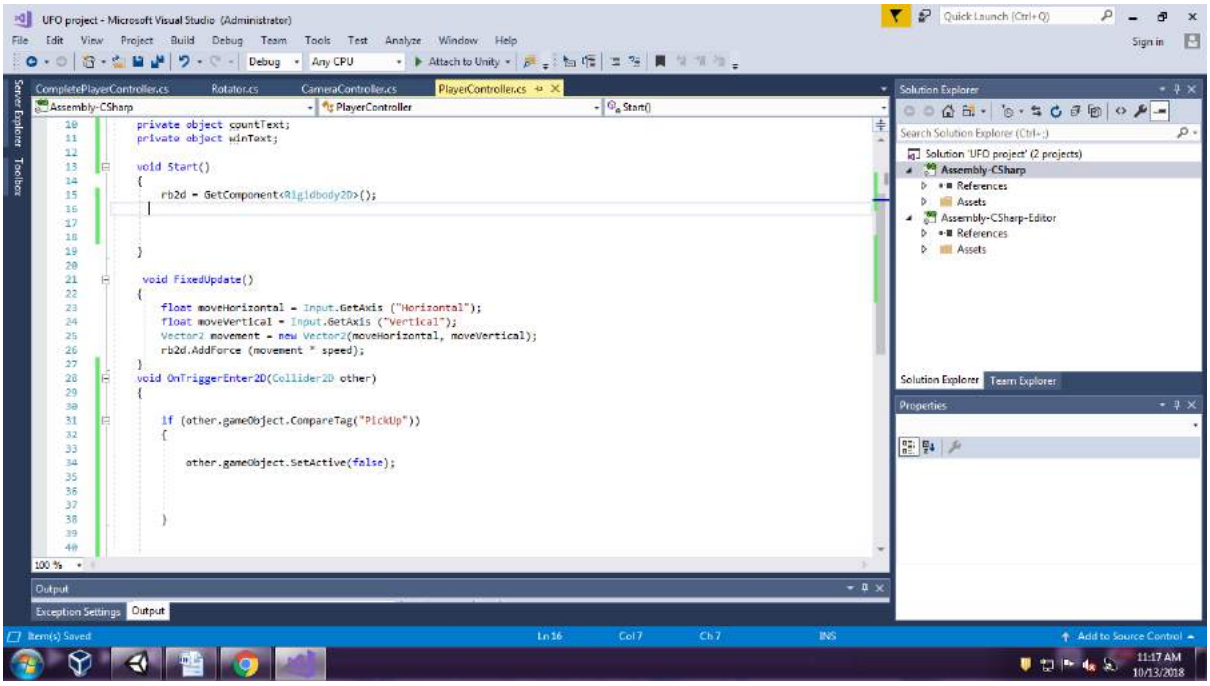
30. **Now drag the pickup object onto pickups object and then give the position of first pickup object on the top.**



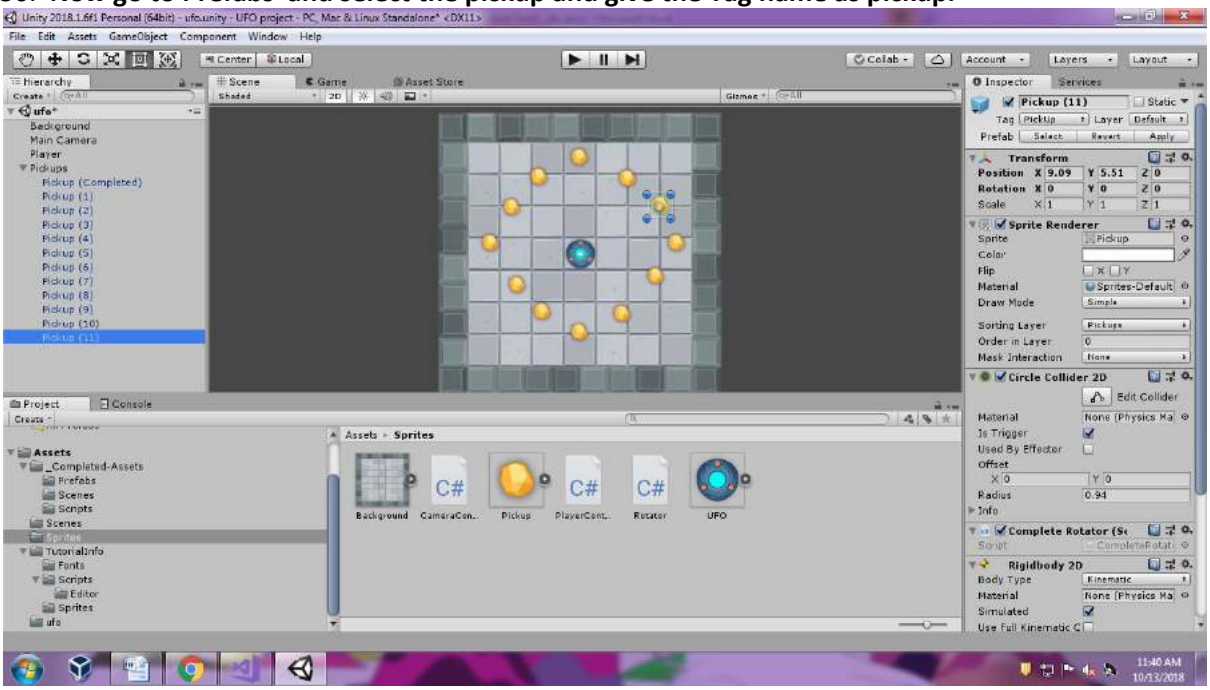
31. **Create many pickup and set the position accordingly.**



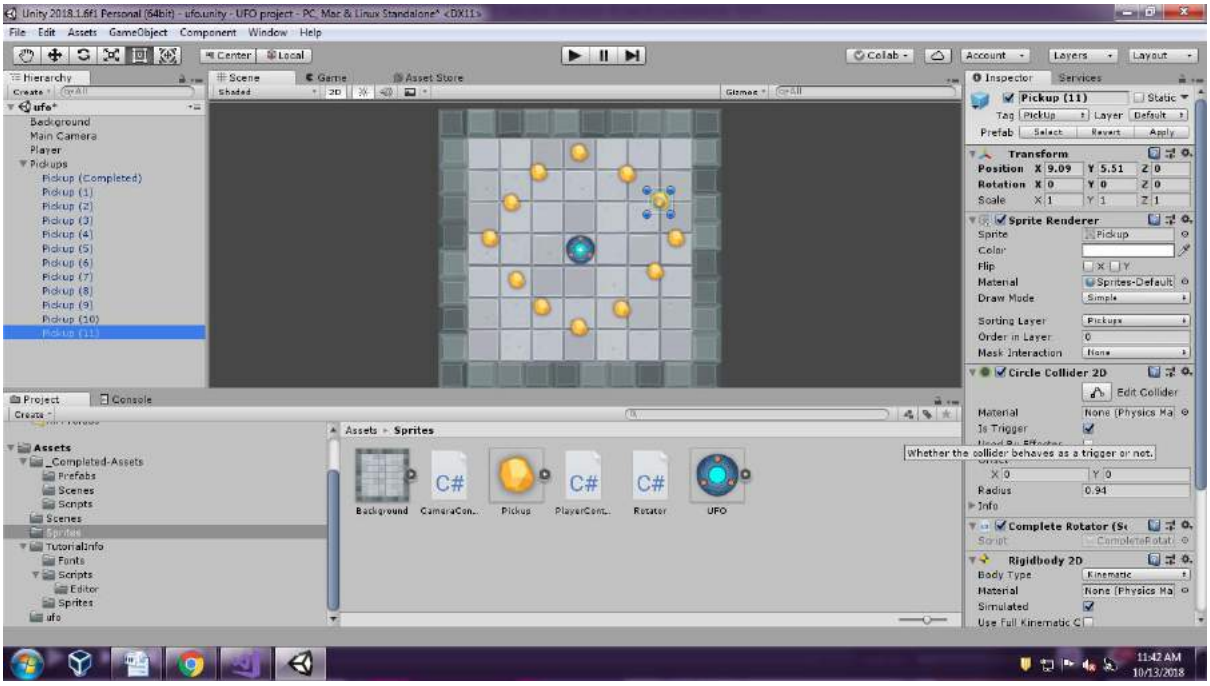
32. To detect collision between the Player and Pickups ,double click on script PlayerController and include some lines of code in PlayerController .



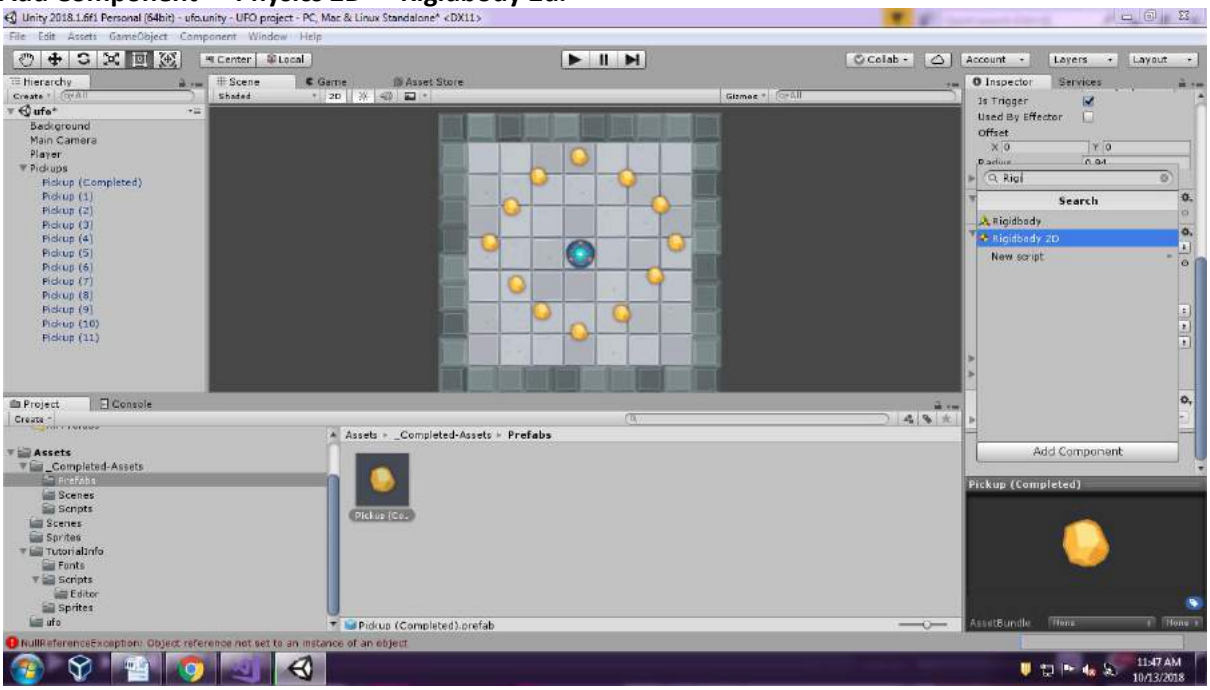
36. **Now go to Prefabs and select the pickup and give the Tag name as pickup.**



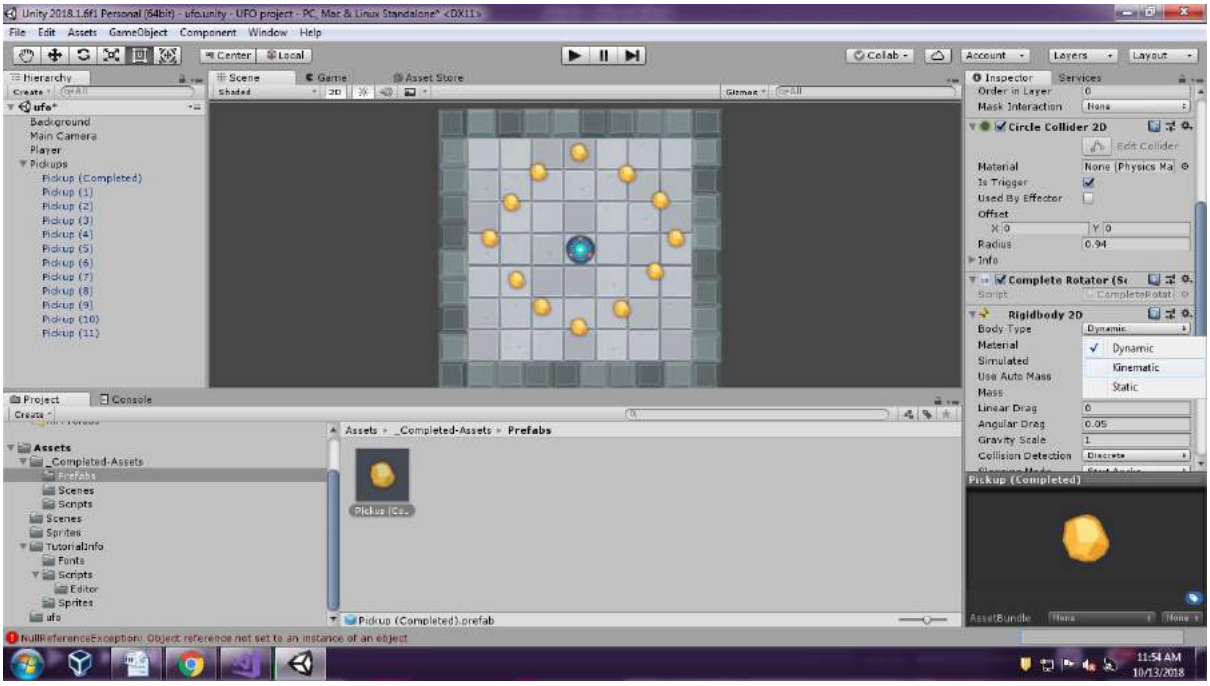
37. Now highlight the pickup11 and Check the isTrigger in CircleCollider2D .



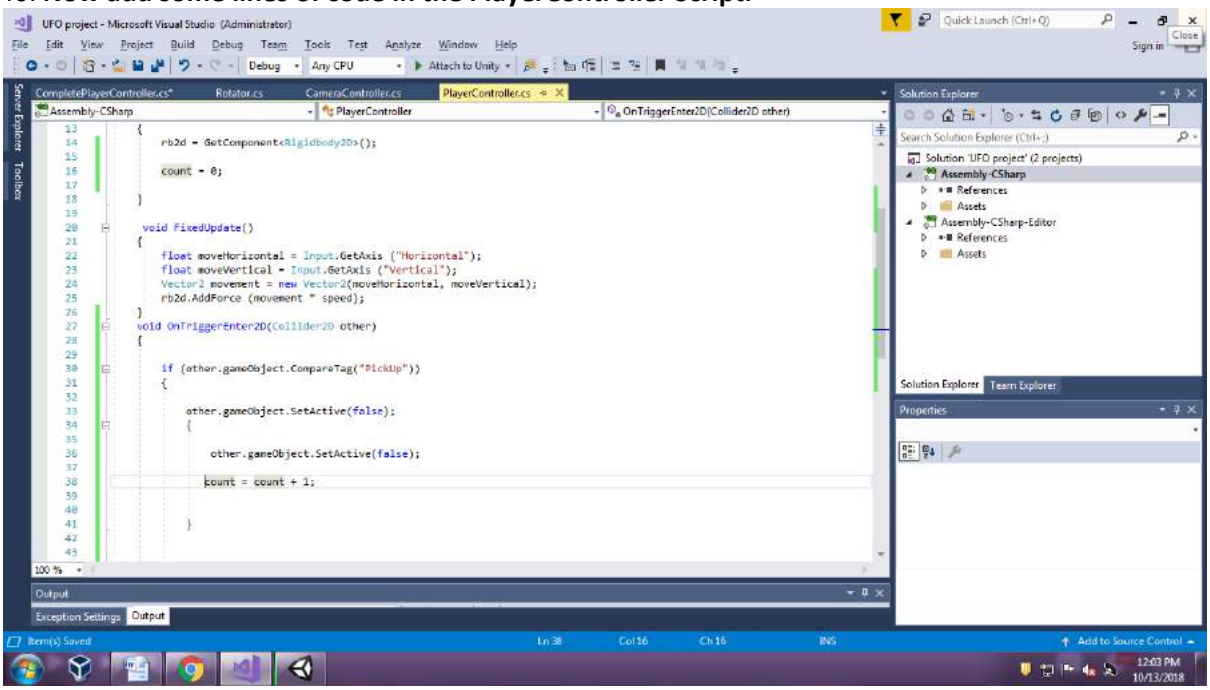
38. **Now highlight the prefabs pickup and add a component Rigidbody2D.**
Add Component -> Physics 2D -> Rigidbody 2d.



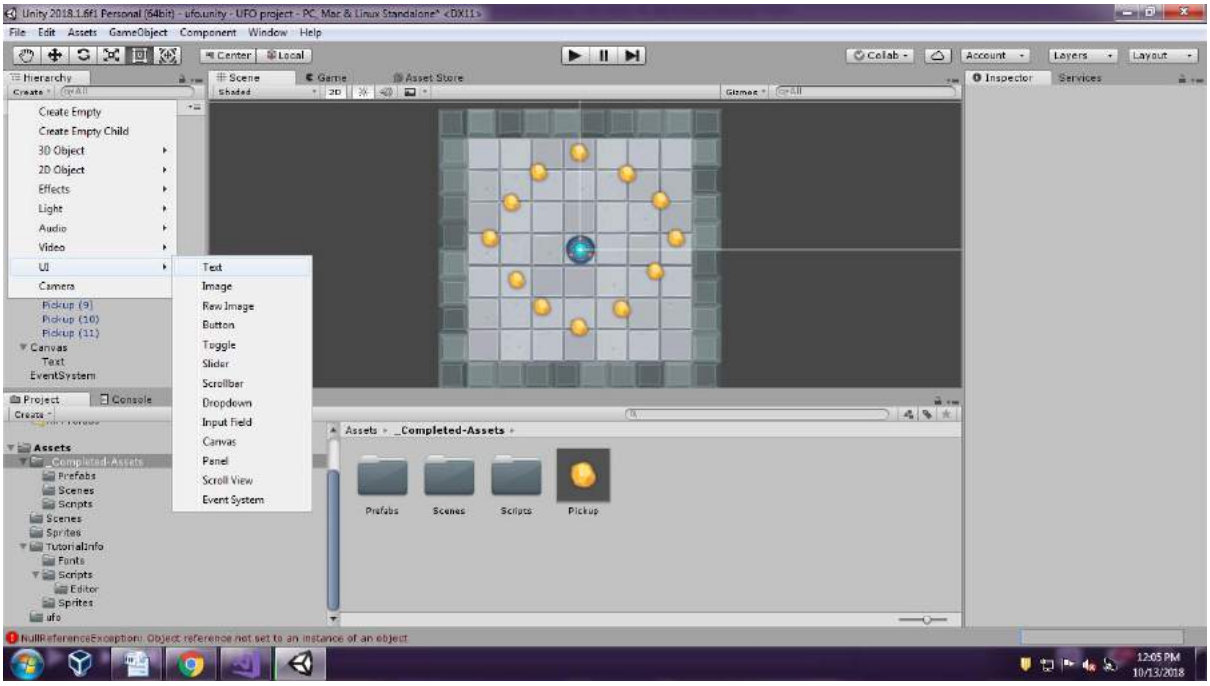
39. **Change the BodyType as Kinetic in the RigidBody 2D of pickups.**



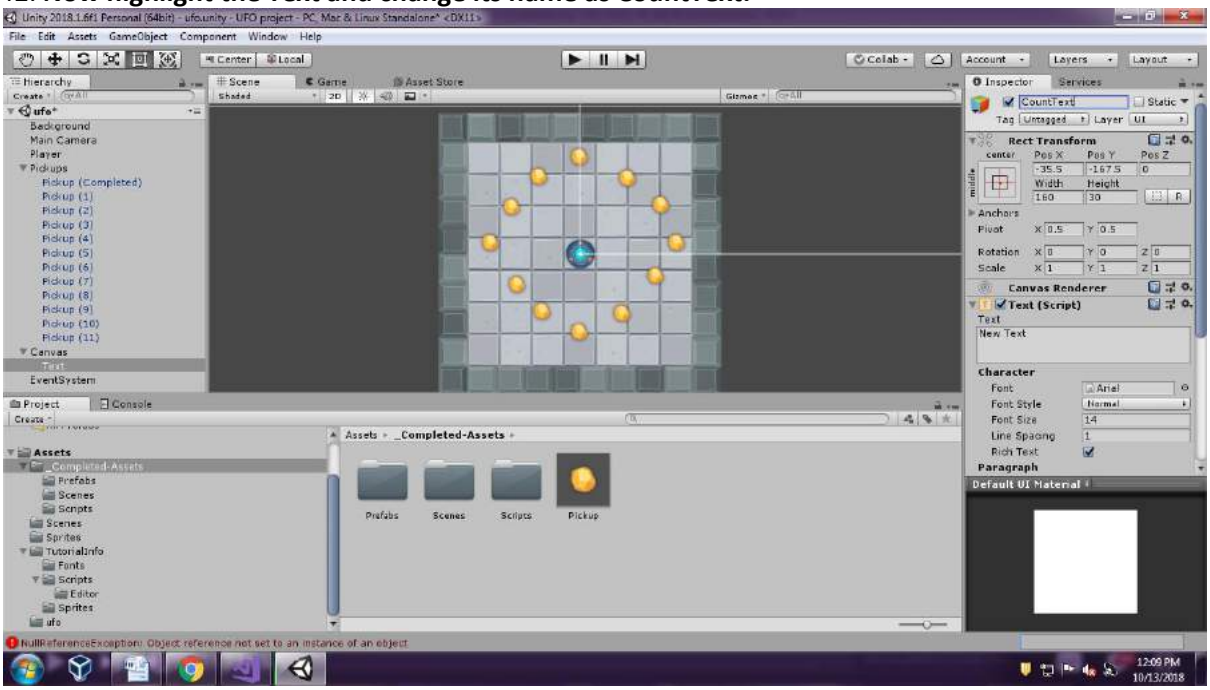
40. **Now add some lines of code in the PlayerController Script.**



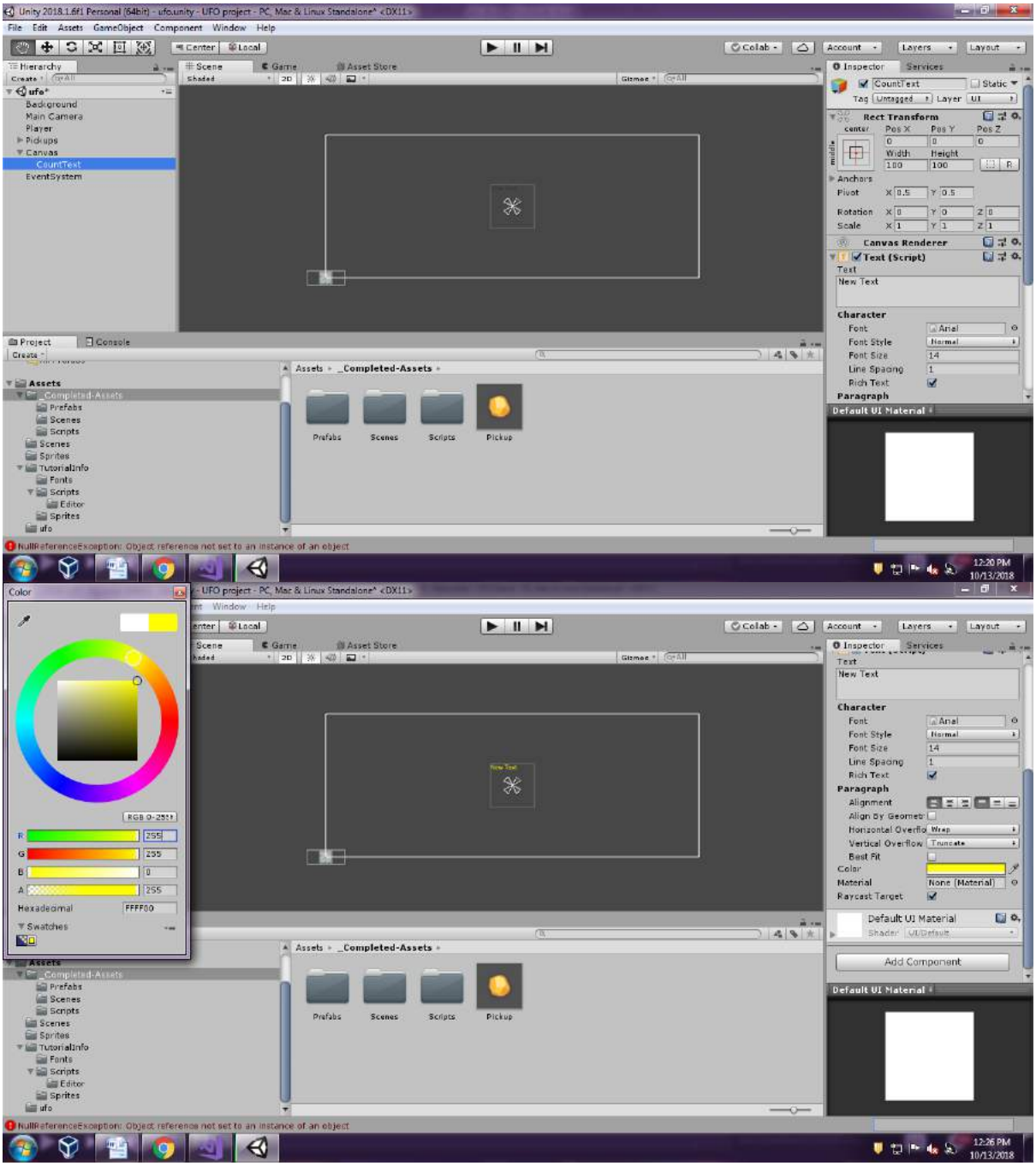
41. **Now create an UserInterface text for canvas.**
Create -> UI -> Text.



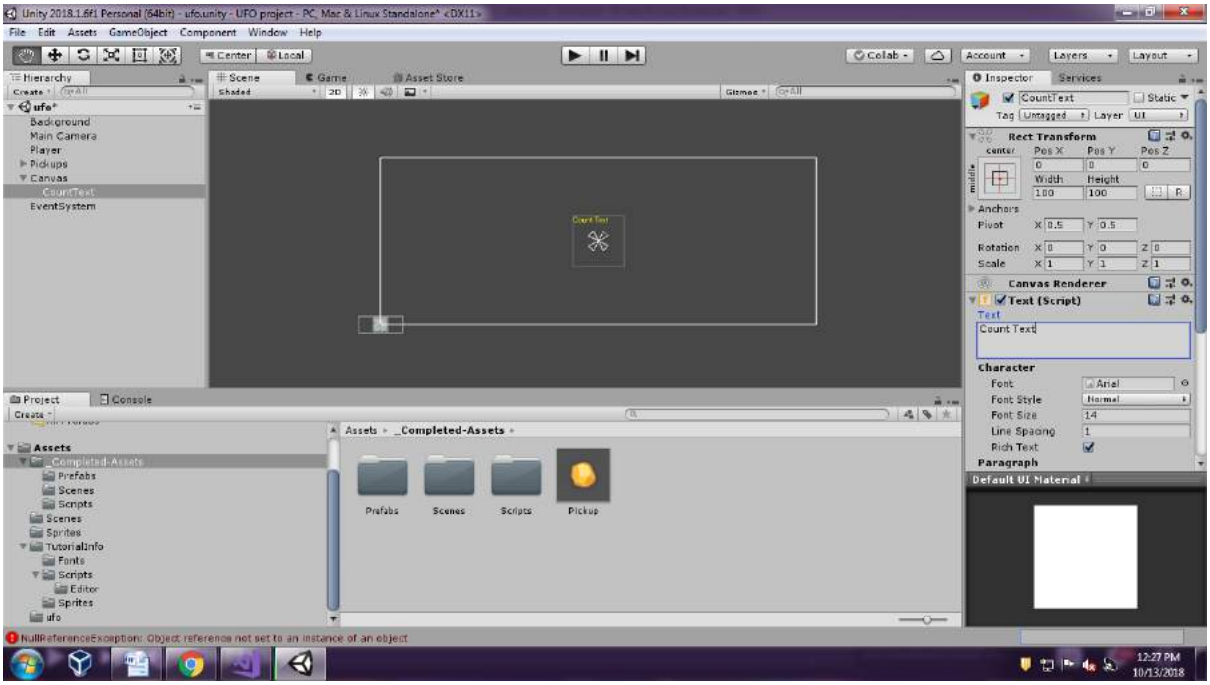
42. **Now highlight the Text and change its name as CountText.**



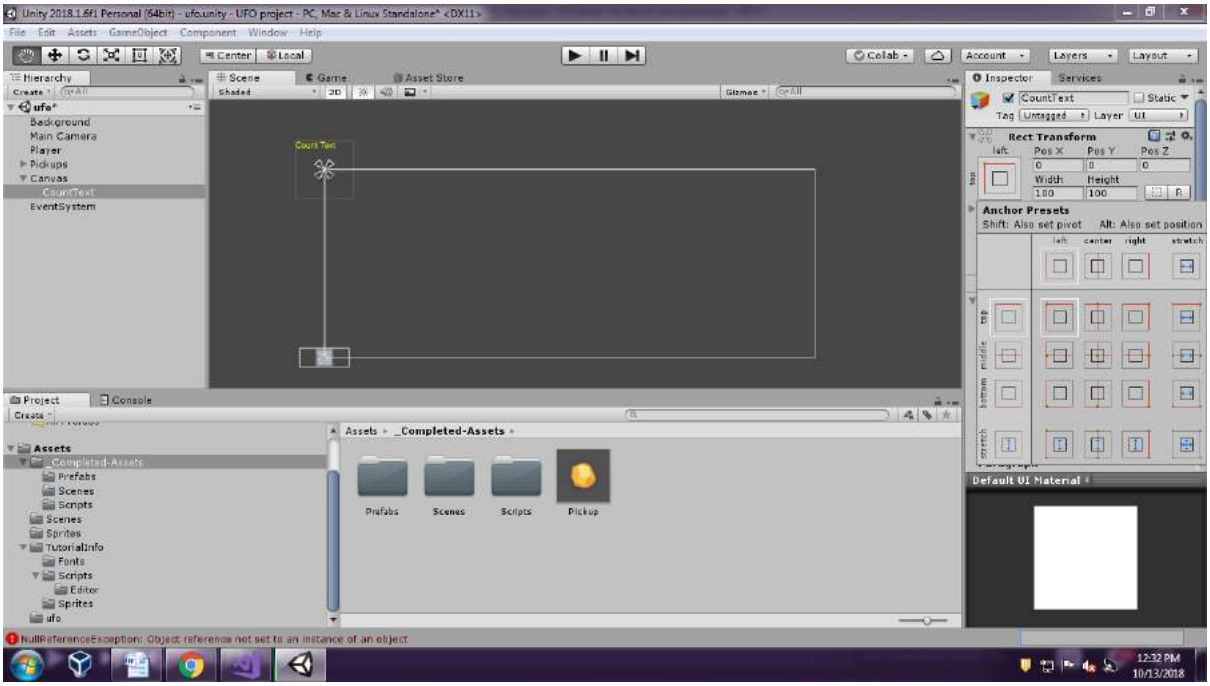
43. **Now highlight the CountText from the Hirerachy and reset is from the settings and Change the Color.**

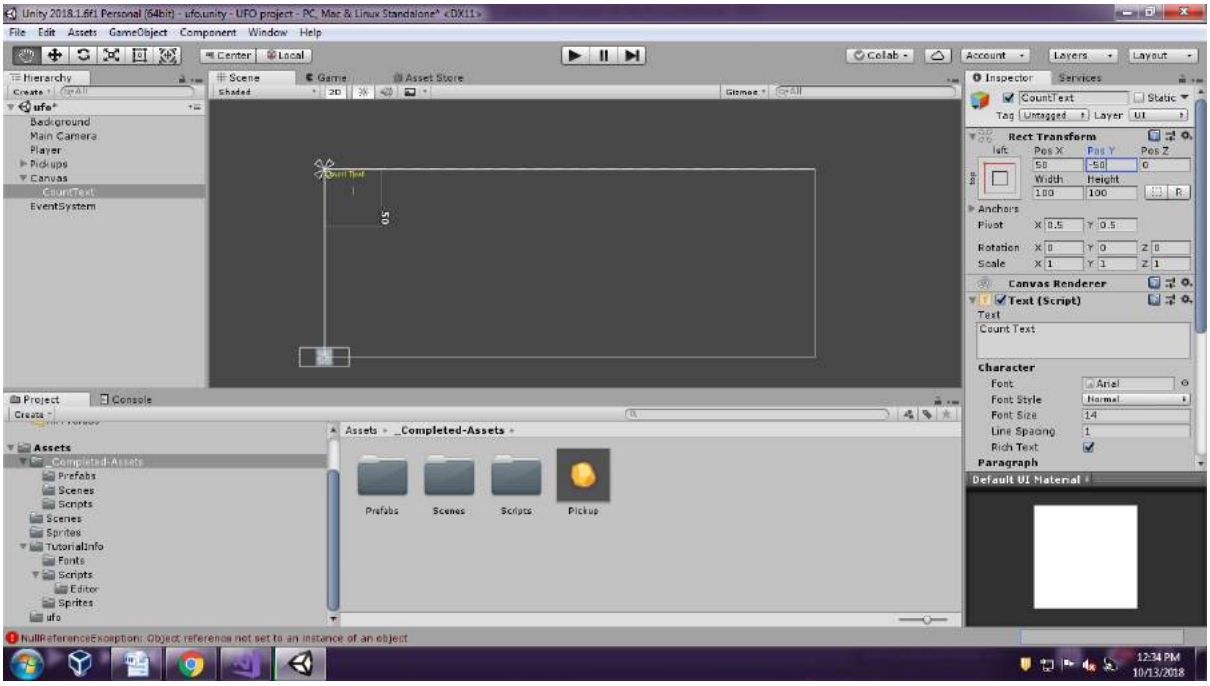


44. Now change the text as Count Text

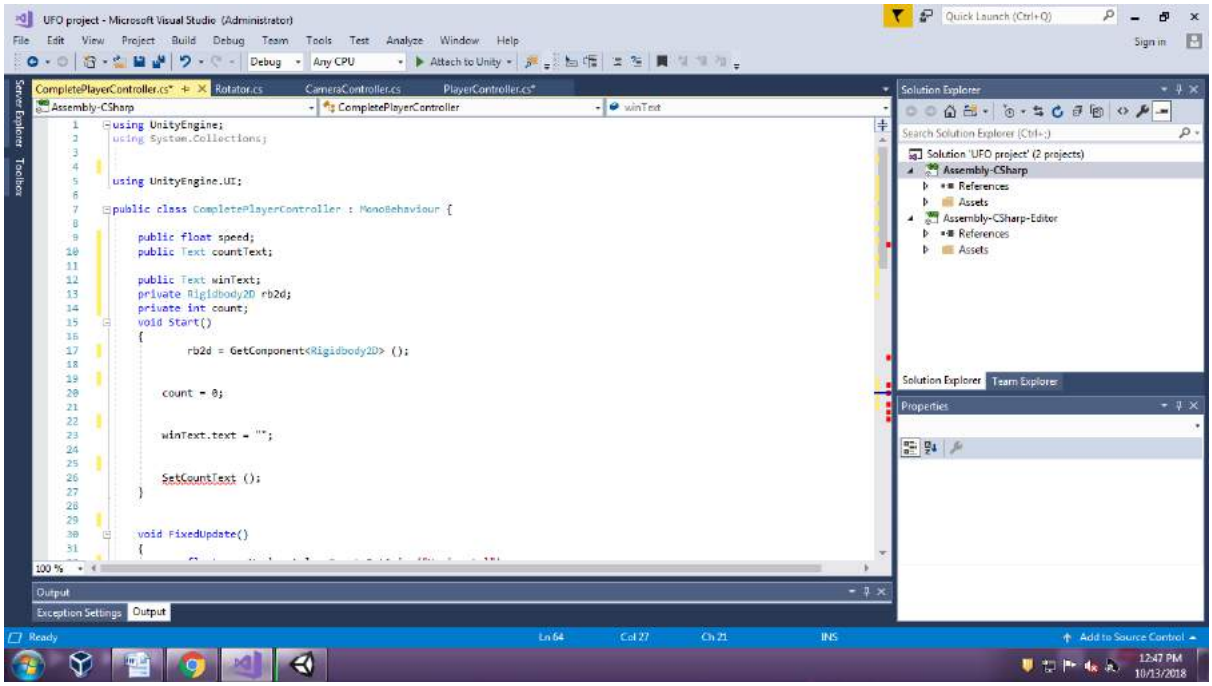


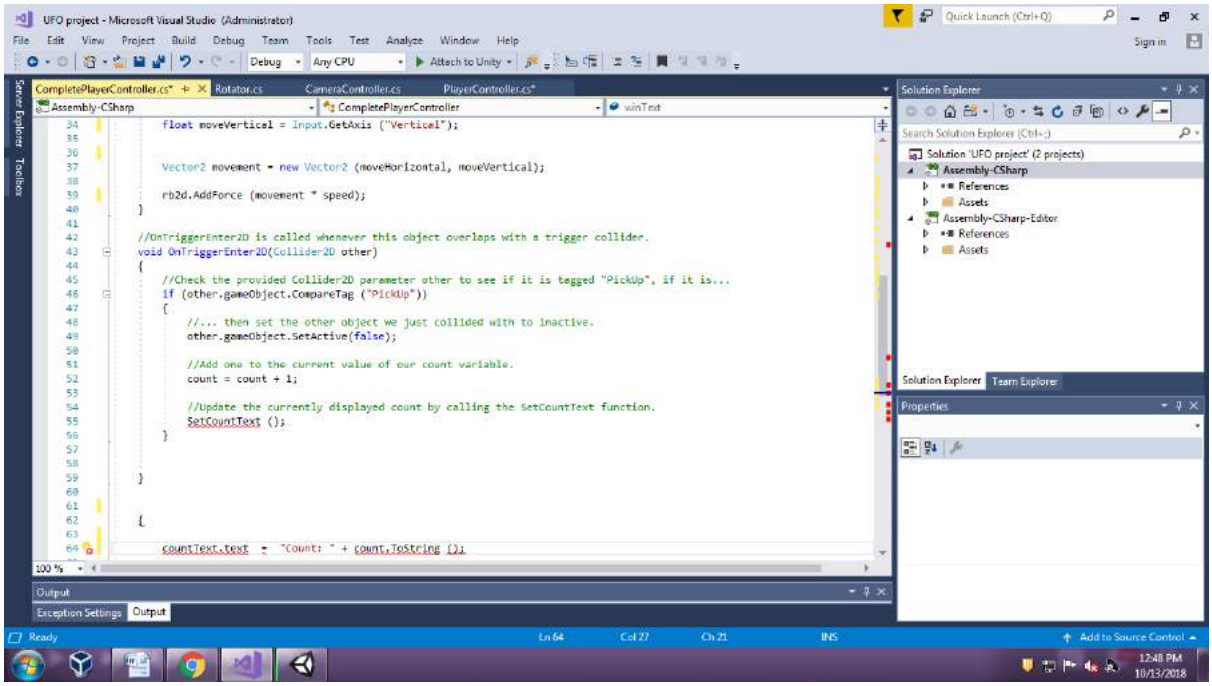
45. Now Set the Size of Count text on left corner of screen.

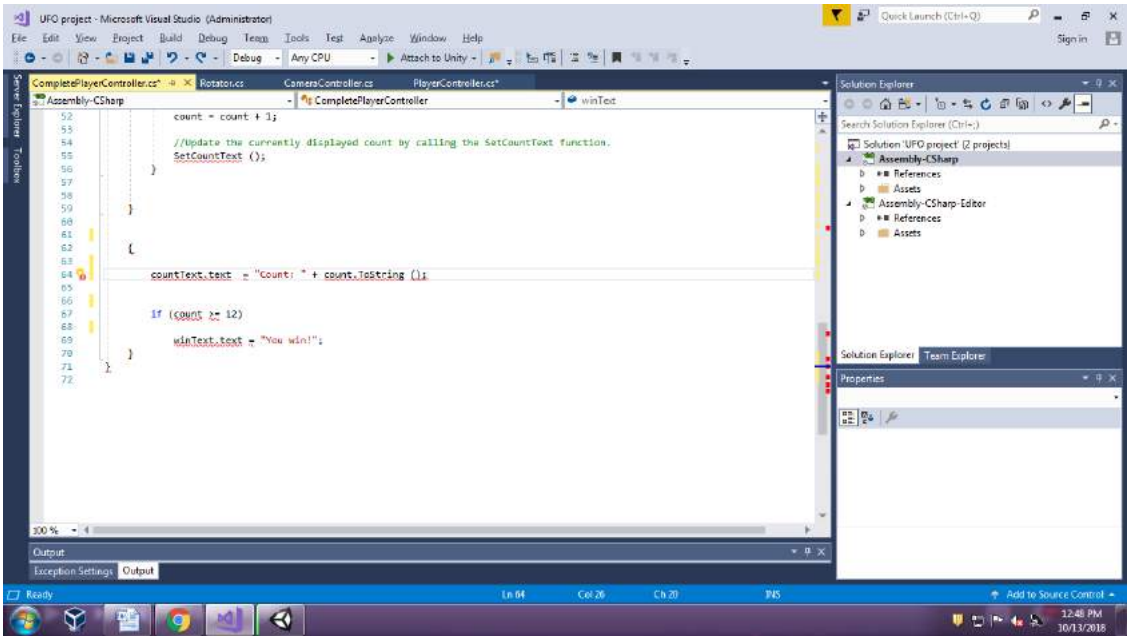


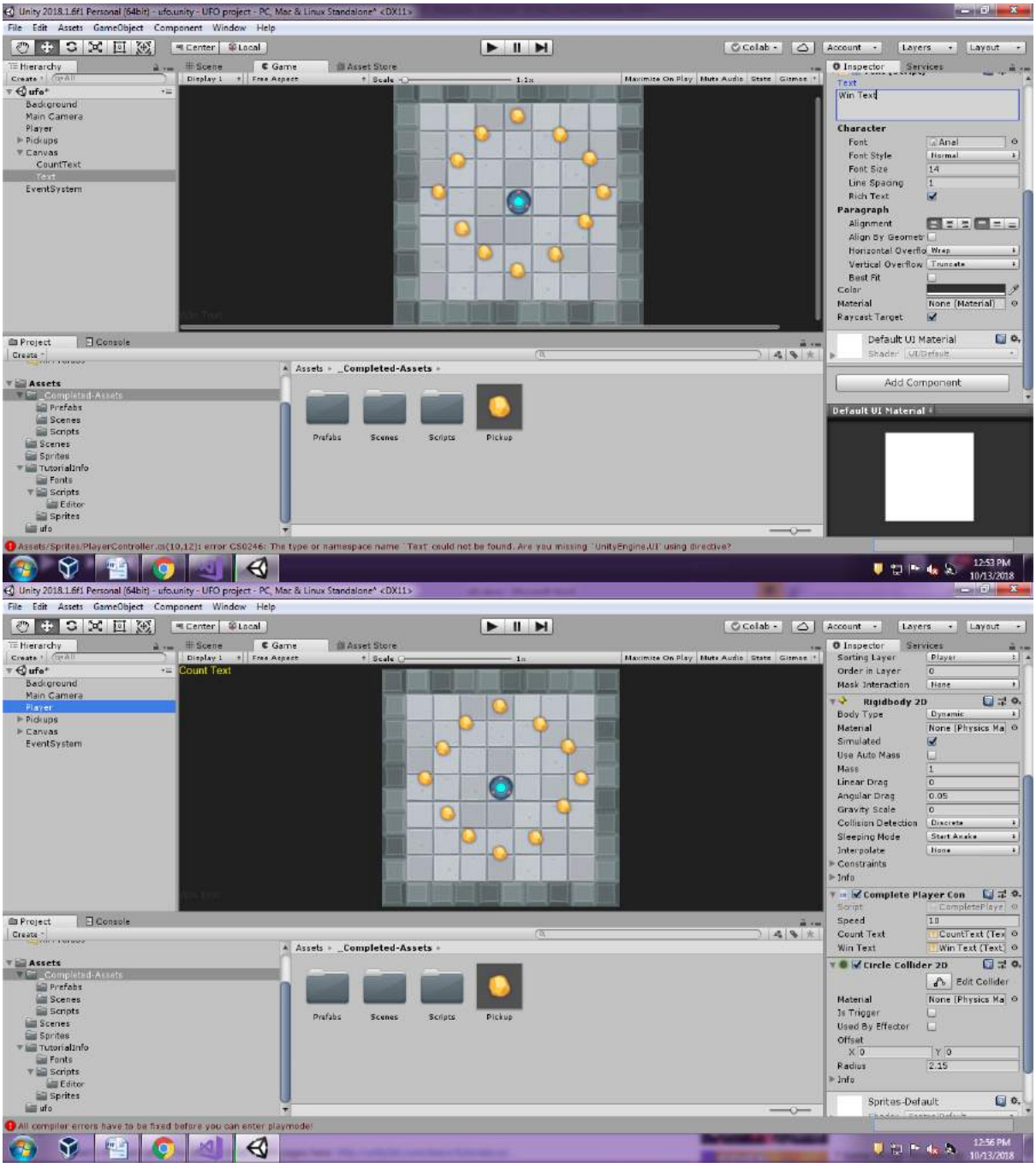


Now Again Click on PlayerController Script and some lines of code in it.


UFO project - Microsoft Visual Studio (Administrator)
CompletePlayerController.cs*
Rotator.cs
CameraController.cs
PlayerController.cs*
Assembly-CSharp
CompletePlayerController
winText
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
public class CompletePlayerController : MonoBehaviour {
public float speed;
public Text countText;
public Text winText;
private Rigidbody2D rb2d;
private int count;
void Start()
{
rb2d = GetComponent<Rigidbody2D> ();
count = 0;
winText.text = "";
SetCountText ();
}
void FixedUpdate()
{
Solution Explorer
Solution 'UFO project' (2 projects)
Assembly-CSharp
References
Assets
Assembly-CSharp-Editor
References
Assets
Properties
Output
Exception Settings
Output
Ready
Ln 64
Col 27
Ch 21
INS
Add to Source Control
12:47 PM
10/13/2018


UFO project - Microsoft Visual Studio (Administrator)
CompletePlayerController.cs*
Rotator.cs
CameraController.cs
PlayerController.cs*
Assembly-CSharp
CompletePlayerController
winText
float moveVertical = Input.GetAxis ("Vertical");
Vector2 movement = new Vector2 (moveHorizontal, moveVertical);
rb2d.AddForce (movement * speed);
}
//OnTriggerEnter2D is called whenever this object overlaps with a trigger collider.
void OnTriggerEnter2D(Collider2D other)
{
//Check the provided Collider2D parameter other to see if it is tagged "PickUp", if it is...
if (other.gameObject.CompareTag ("PickUp"))
{
//... then set the other object we just collided with to inactive.
other.gameObject.SetActive(false);
//Add one to the current value of our count variable.
count = count + 1;
//Update the currently displayed count by calling the SetCountText function.
SetCountText ();
}
}
{
countText.text = "Count: " + count.ToString ();
Solution Explorer
Solution 'UFO project' (2 projects)
Assembly-CSharp
References
Assets
Assembly-CSharp-Editor
References
Assets
Properties
Output
Exception Settings
Output
Ready
Ln 64
Col 27
Ch 21
INS
Add to Source Control
12:48 PM
10/13/2018


UFO project - Microsoft Visual Studio (Administrator)
CompletePlayerController.cs*
Rotator.cs
CameraController.cs
PlayerController.cs*
Assembly-CSharp
CompletePlayerController
winText
count = count + 1;
//Update the currently displayed count by calling the SetCountText function.
SetCountText ();
}
}
{
countText.text = "Count: " + count.ToString ();
if (count >= 12)
winText.text = "You win!";
}
}
Solution Explorer
Solution 'UFO project' (2 projects)
Assembly-CSharp
References
Assets
Assembly-CSharp-Editor
References
Assets
Properties
Output
Exception Settings
Output
Ready
Ln 64
Col 26
Ch 20
INS
Add to Source Control
12:48 PM
10/13/2018



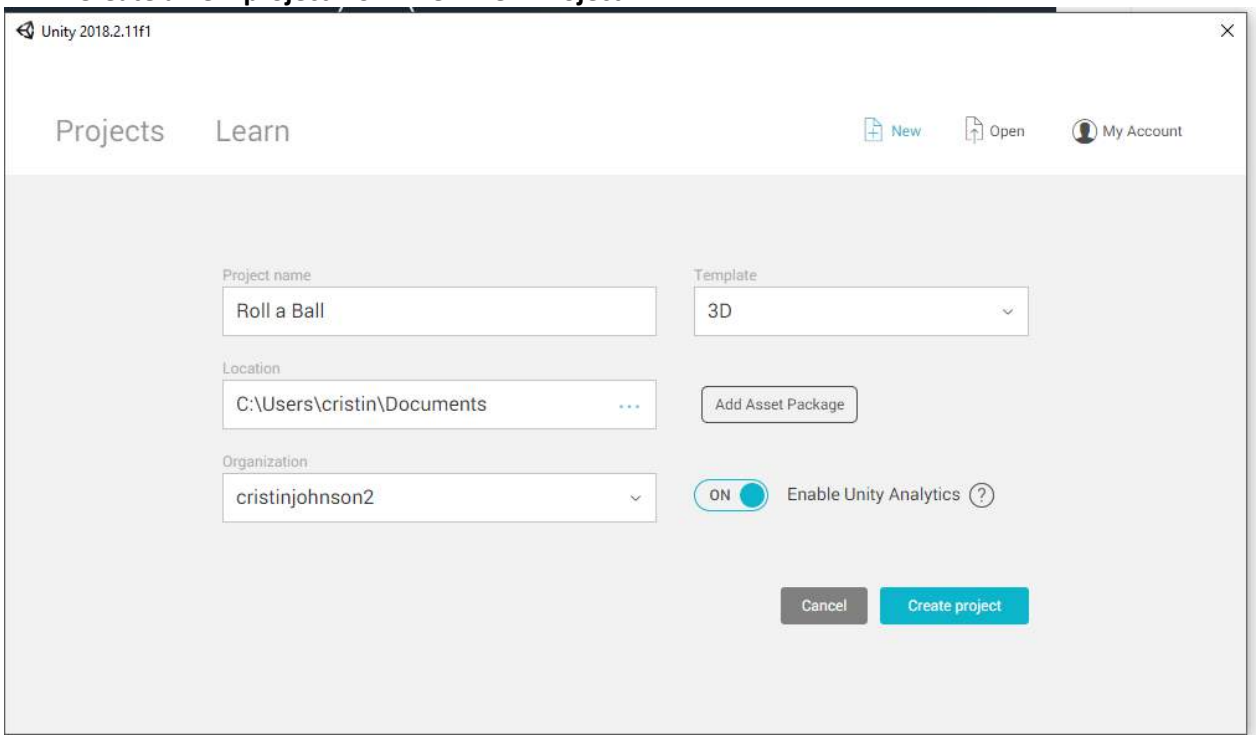
46. Now Finally Save Your Project And BUILD and RUN.

PRACTICAL NO:-09

AIM:- Game development using unity 2D ROLL BALL.

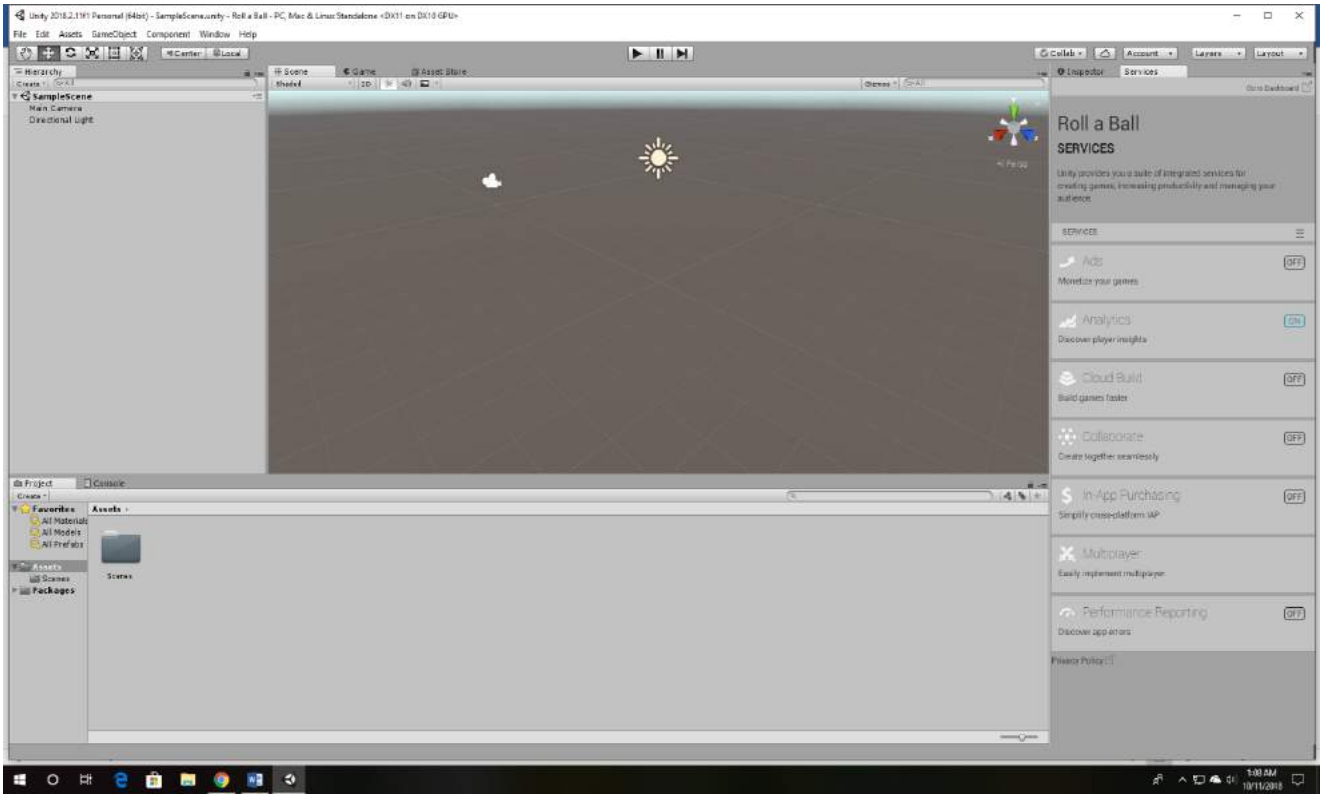
STEPS

1. **Create a new project from File > New Project.**

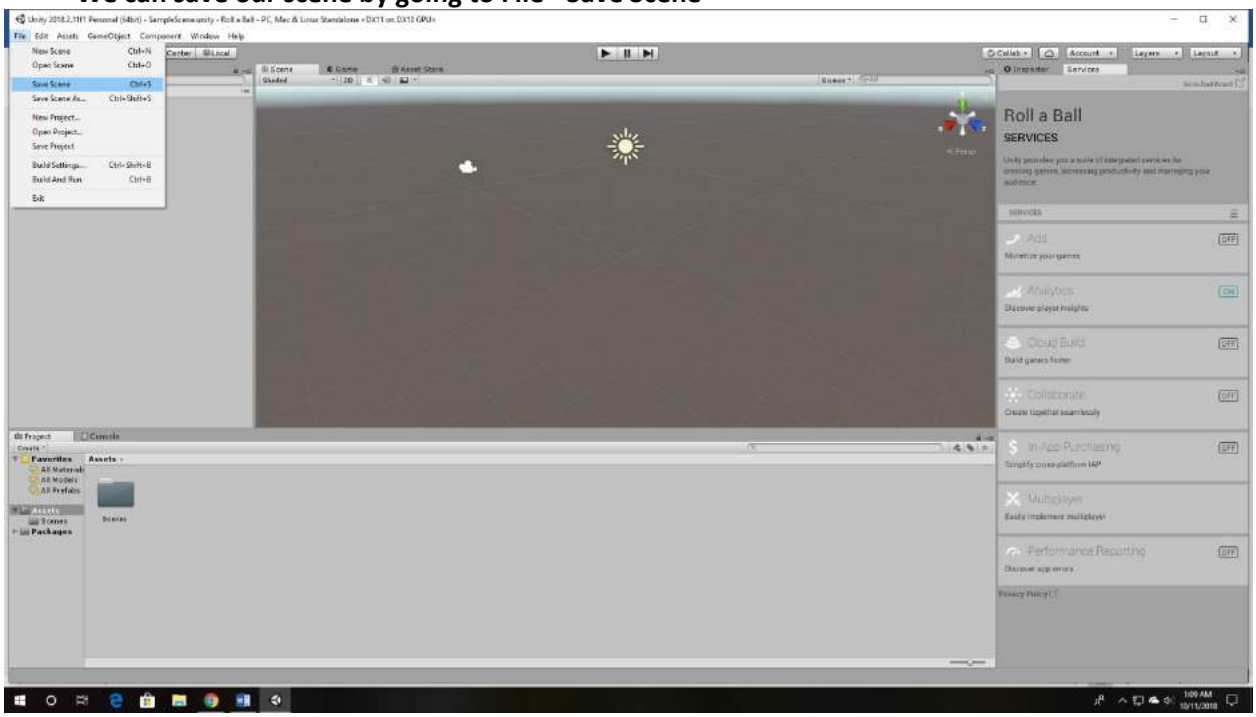


2. **Click on Create Project to make a new project. We now have our new project with a new empty scene.**

3. **Before creating anything in the new scene, we need to save our scene.**

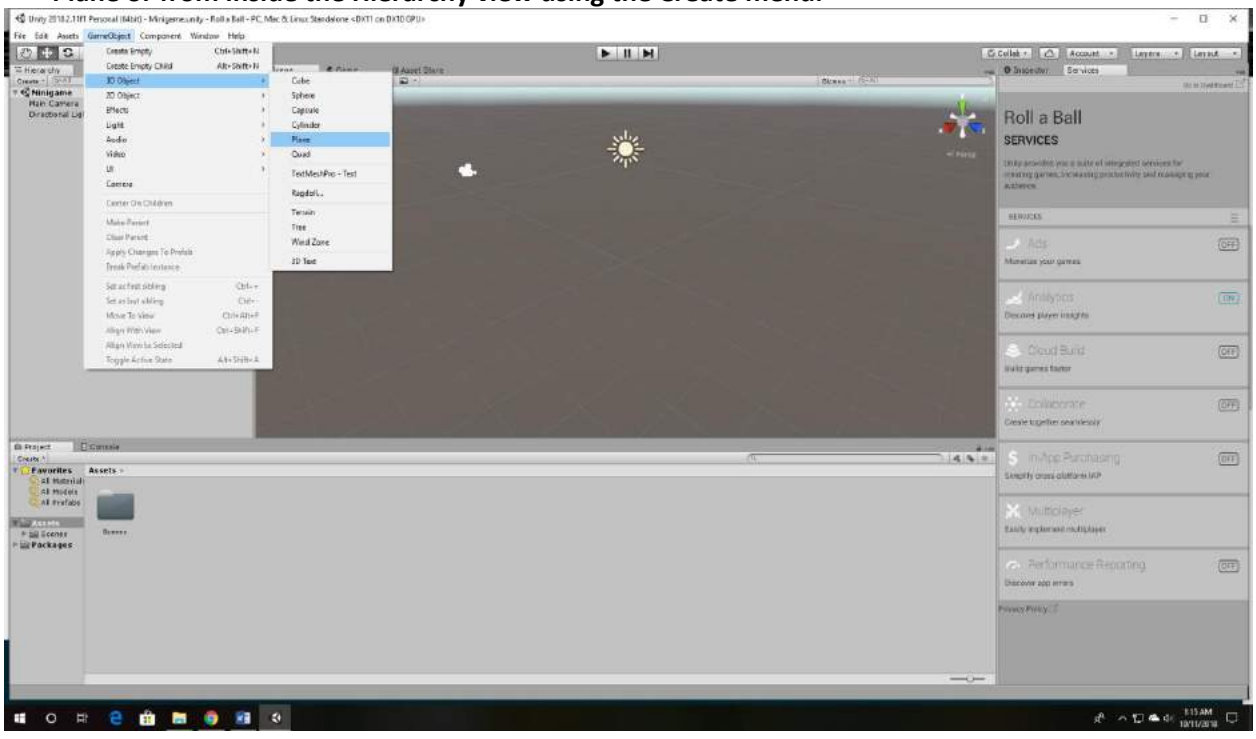


We can save our scene by going to File - Save Scene

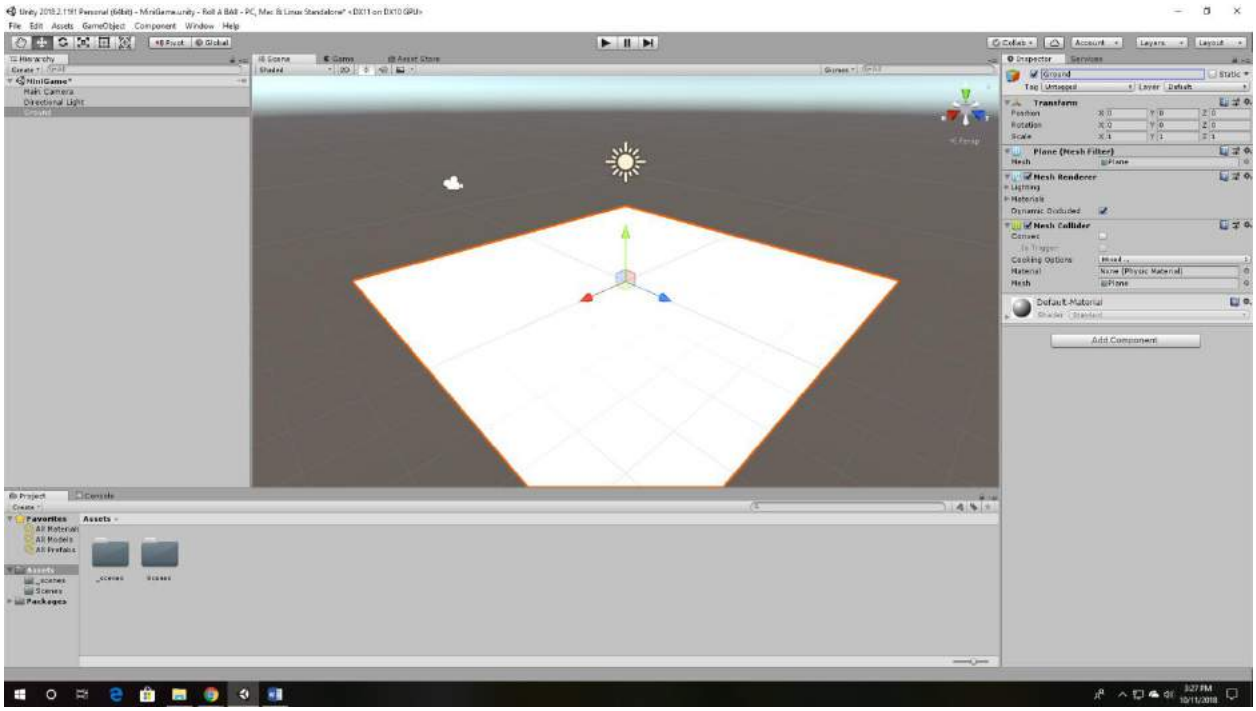


Save this scene in the Assets directory in a new folder called _scenes. Call the scene Minigame.

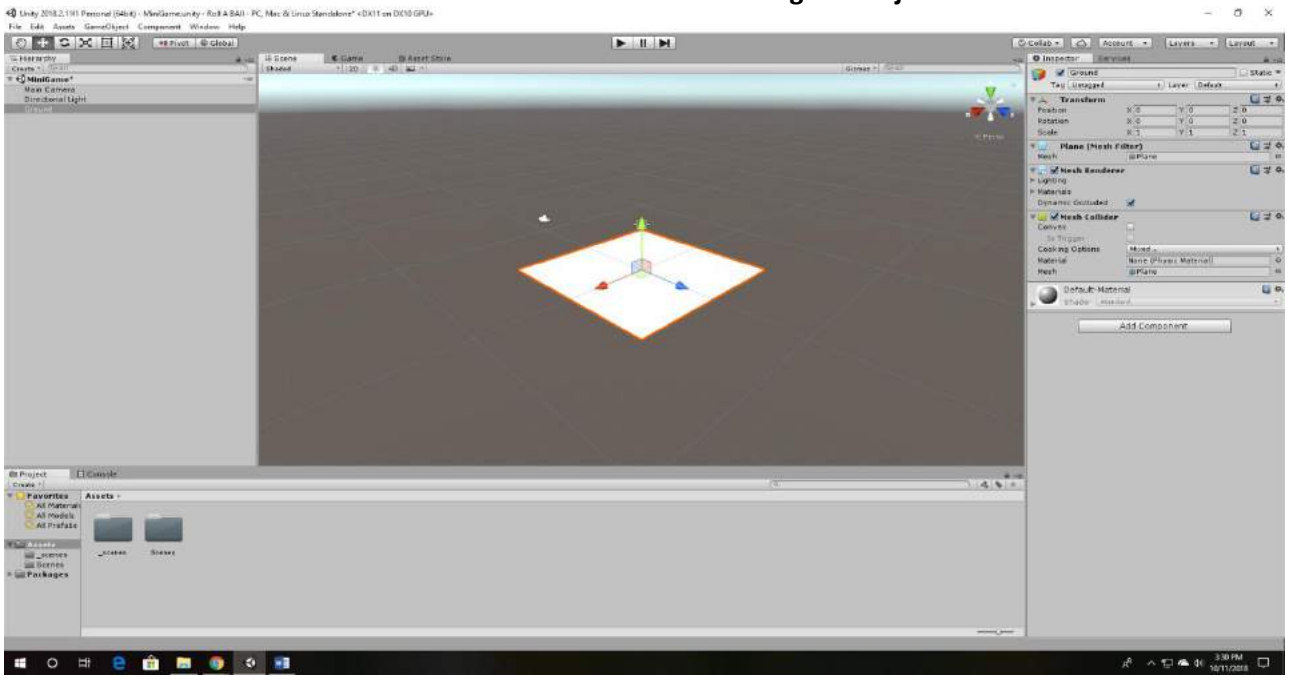
4. **Create the game board or play field. Create this plane from either Game Object - 3D Object – Plane or from inside the Hierarchy view using the Create menu.**



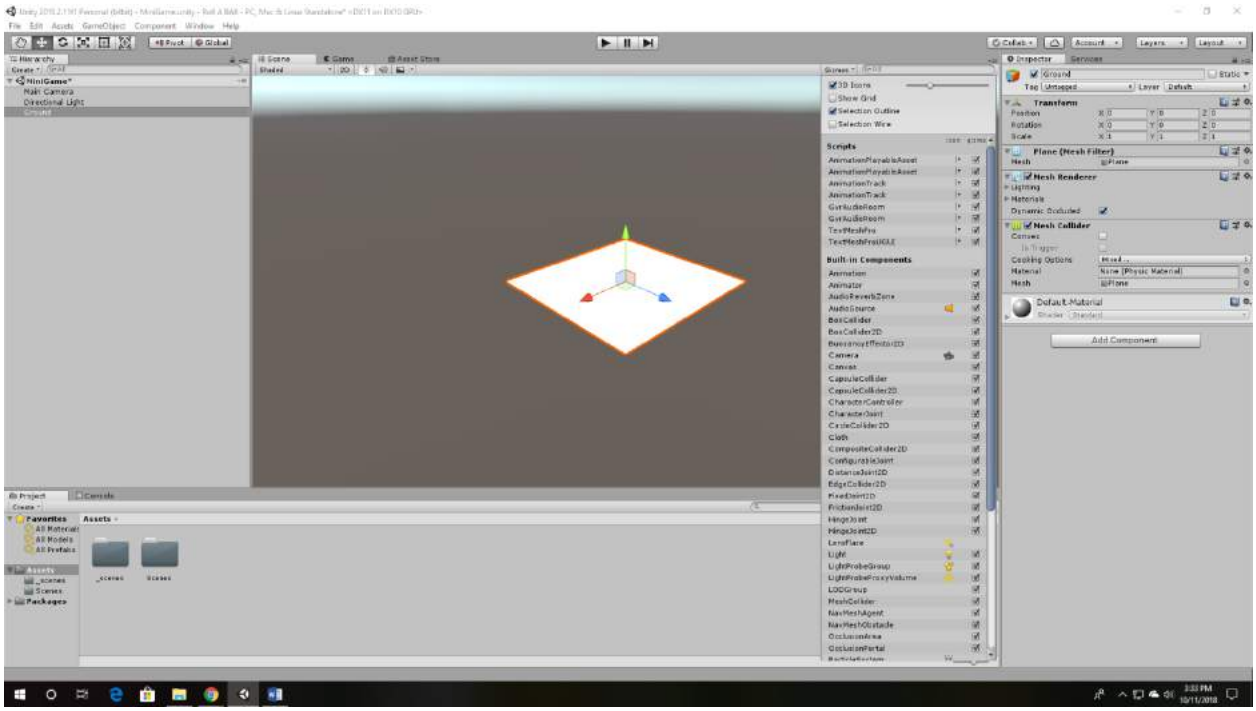
5. **Rename this game object Ground.**
6. **Reset the transform component using the context sensitive gear menu in the upper right. This will place the game object at the location of (0, 0, 0) in the scene.**



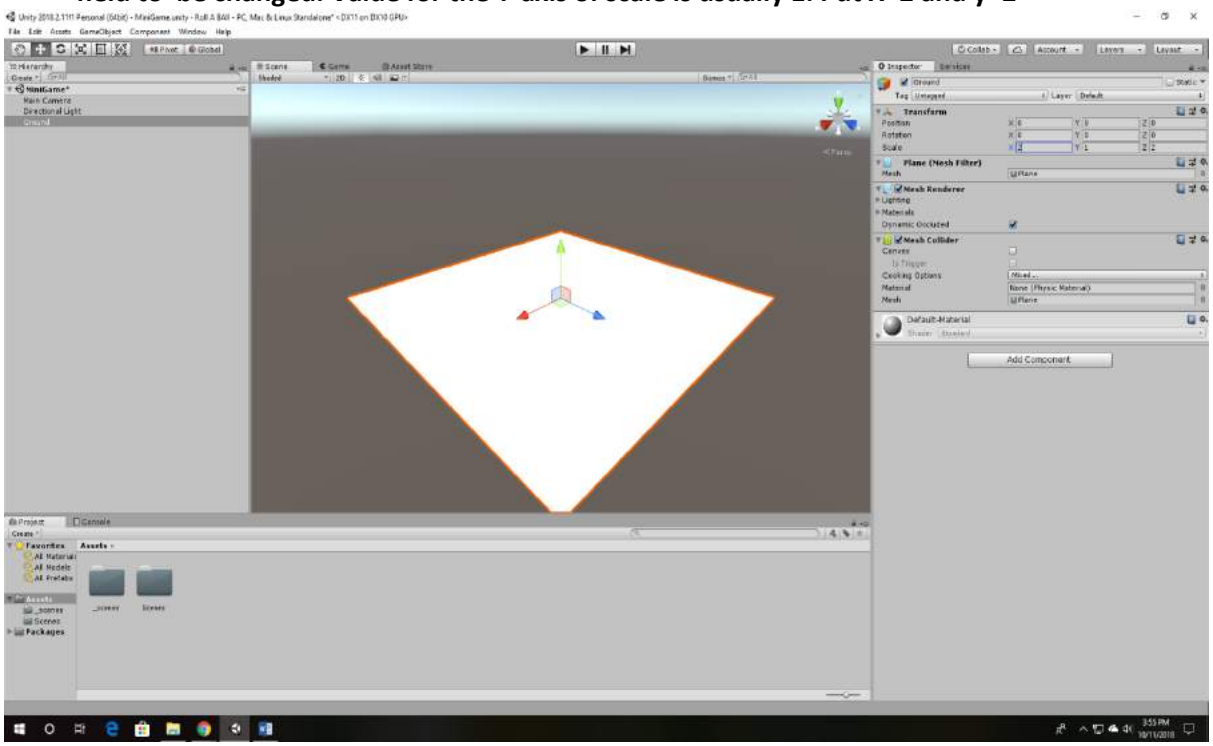
7. **Now with the game object selected and the cursor over the Scene view type the F key, or choose Frame Selected from the Edit menu to see the entire game object in the Scene view.**



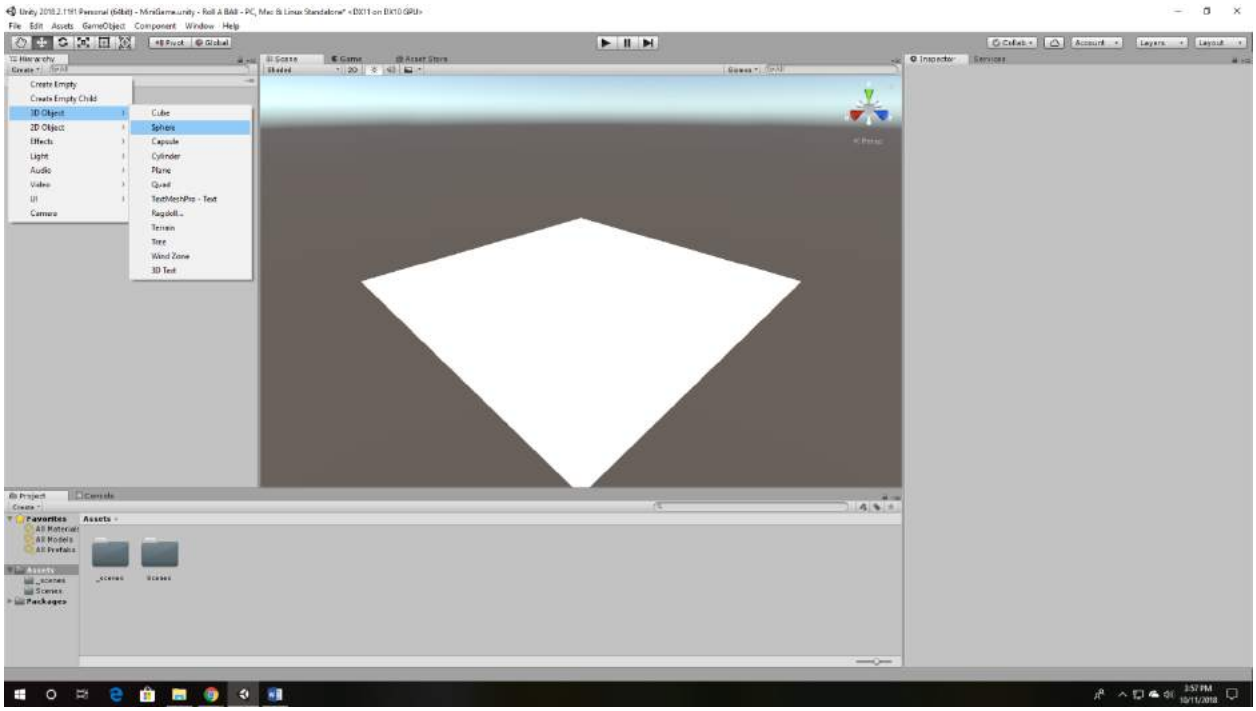
8. **Select the Gizmos menu in the Scene view and deselect Show Grid.**



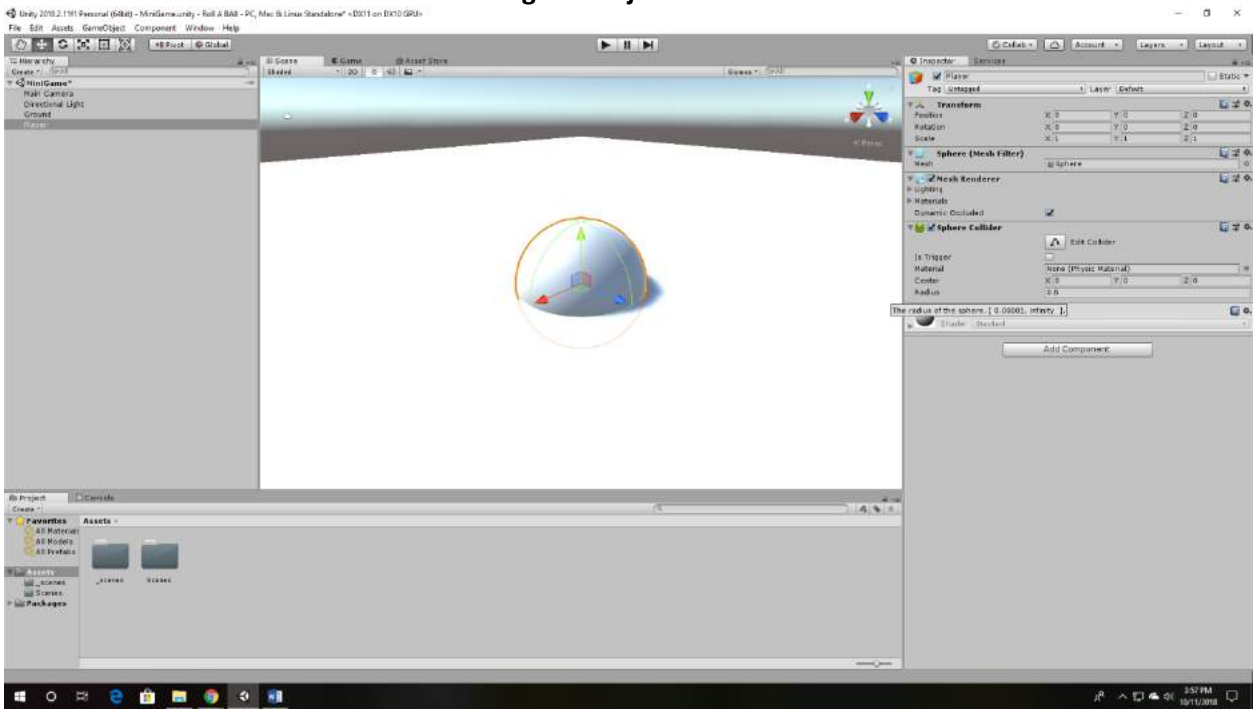
9. **Change the scale of the ground plane. Use the Scale tool, simply grab the axis handle you want to change and drag the handle rescaling the plane or type a number directly in to the field to be changed. Value for the Y-axis of scale is usually 1. Put X=2 and y=2**



10. **Create the player object. From the Hierarchy - Create menu select Sphere. Rename the sphere Player. Reset the transform to make sure it is an origin.**

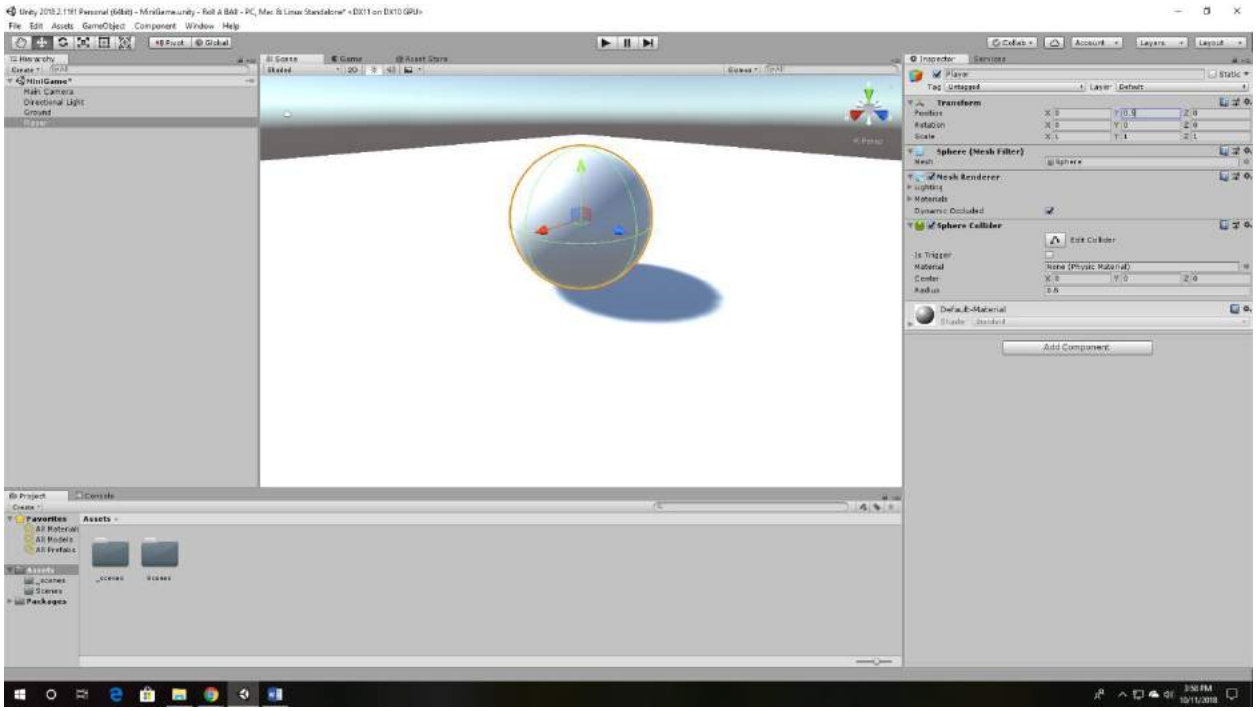


11. **Select Edit - Frame Selected or use the F key while the cursor is over the Scene view to focus our Scene view camera on our game object.**

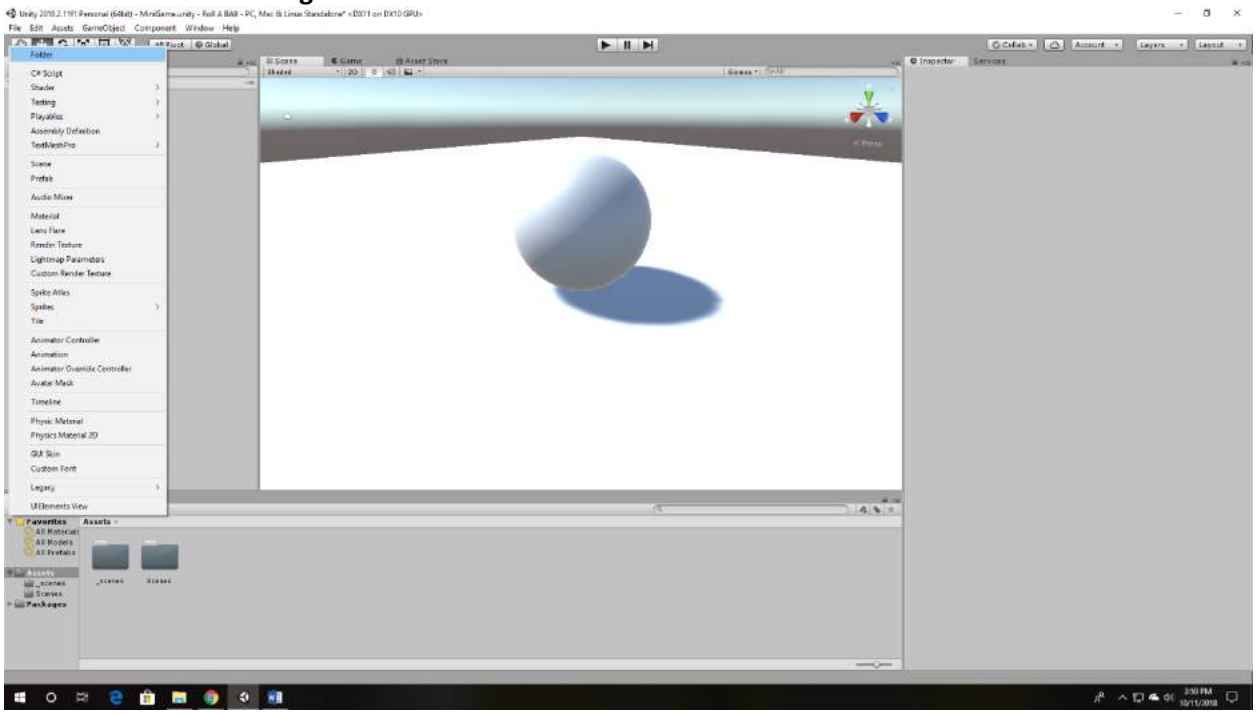


12. **The sphere is buried in the plane. This is because both game objects are in the exact same location in the scene, the origin point, or (0, 0, 0) on the X, Y and Z axis.**

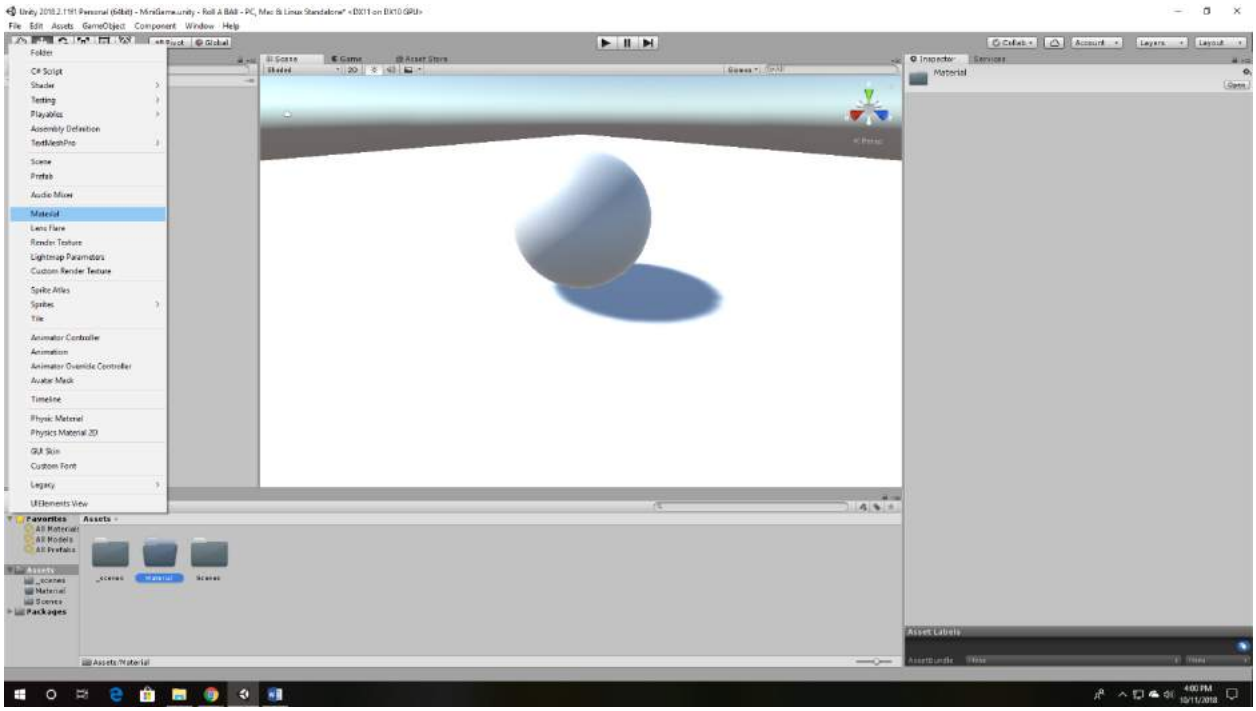
13. **Move the player's sphere up until it rests on the plane. Simply lift the player object up by half a unit in the Y axis. i.e. y= 0.5**



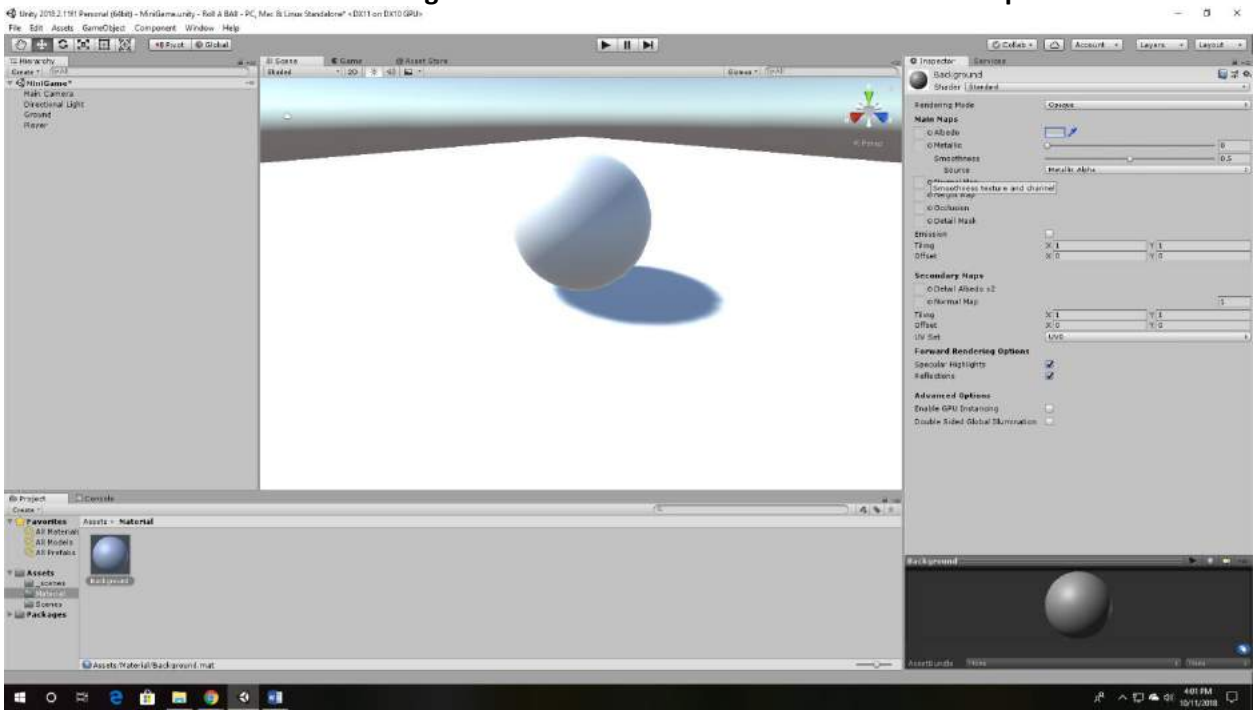
14. **Add some color to the background. To add color or texture to a model use a material.**
15. **Create a new folder in project to hold the materials. Do this by using the project's Create menu and selecting Folder. Rename this folder Material.**



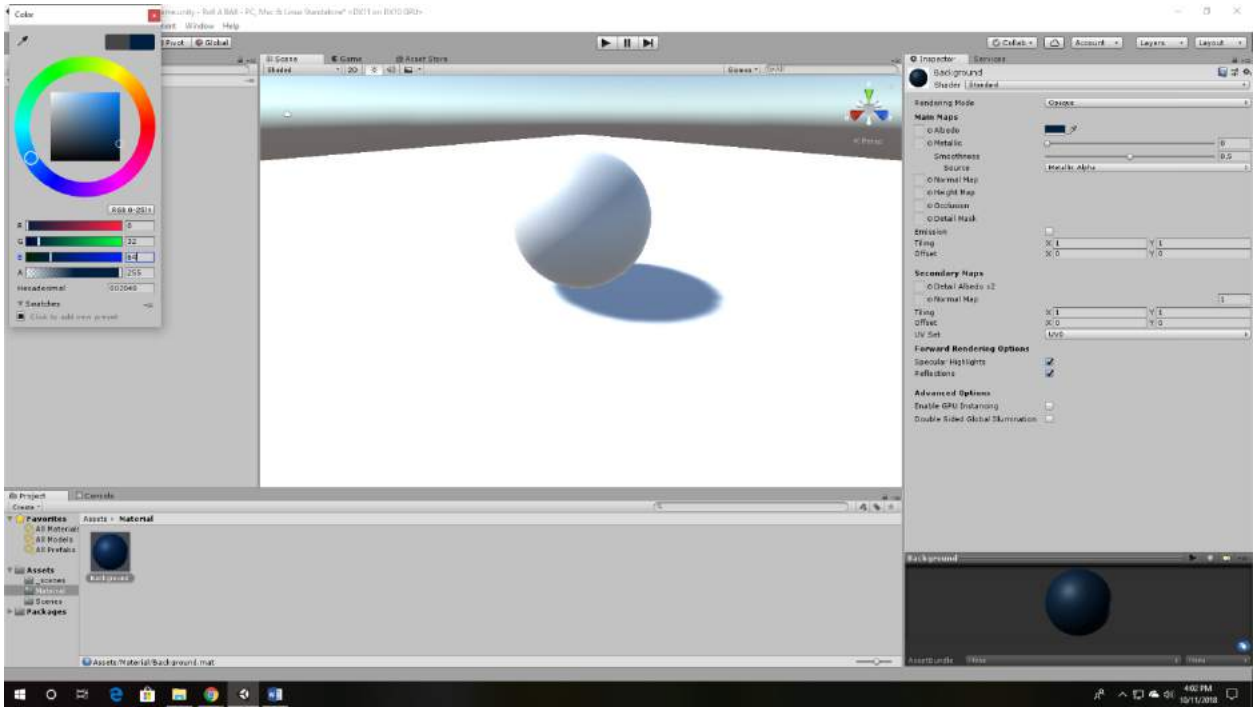
16. **With the Materials folder selected use the project's Create menu again and this time select Material**.



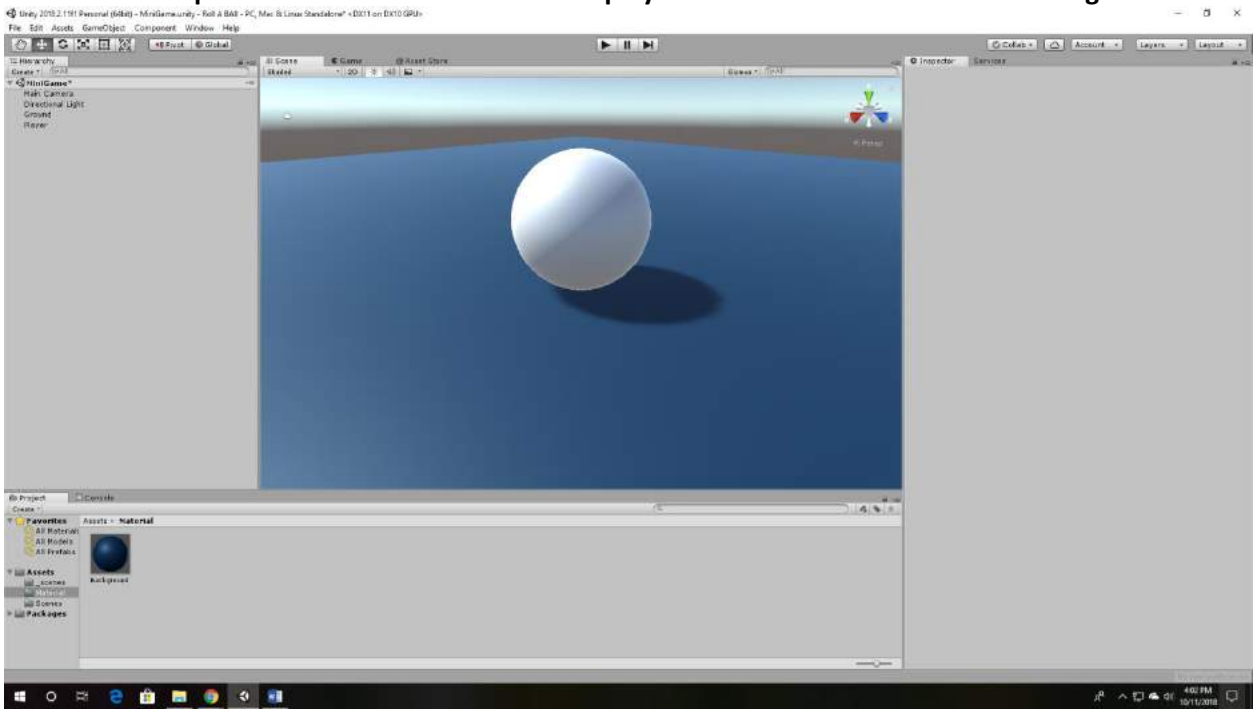
17. **Rename this material Background. Select the material and under Main Maps**



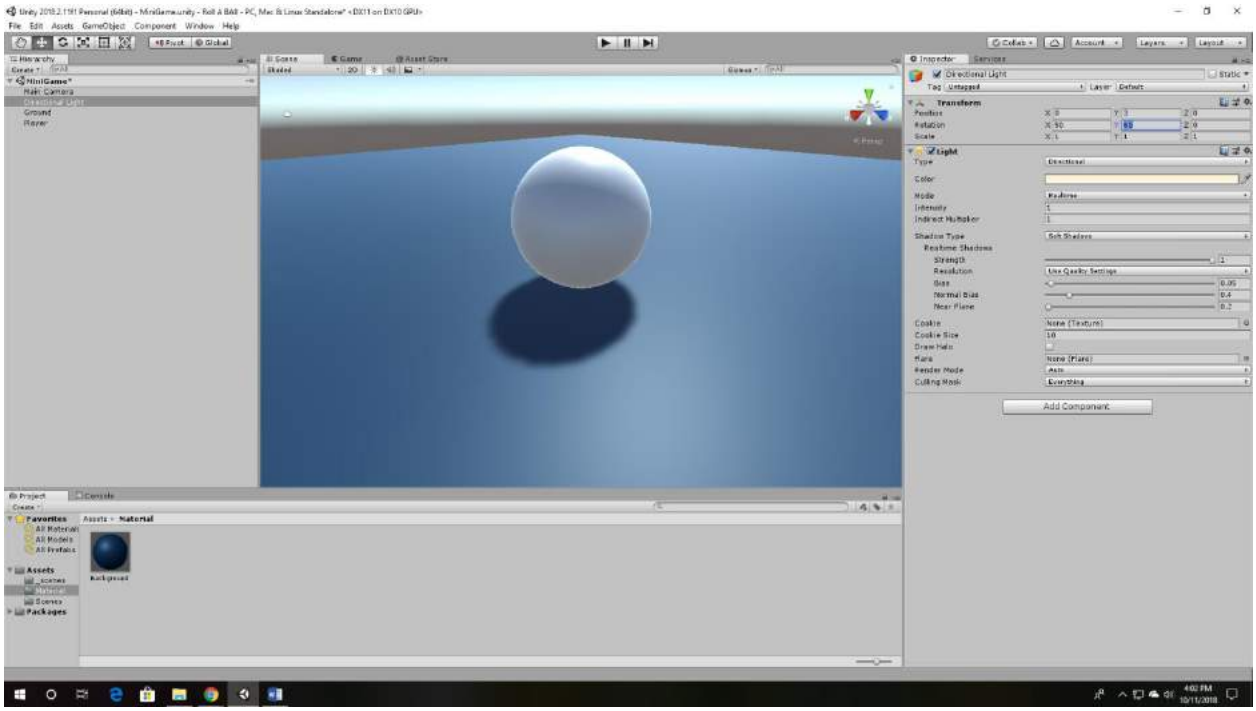
18. **Click on the Albedo's color field to open a color picker.**
19. **Change the color to a nice dark shade of blue. In this case, use the RGB values of 0, 32 and 64.**



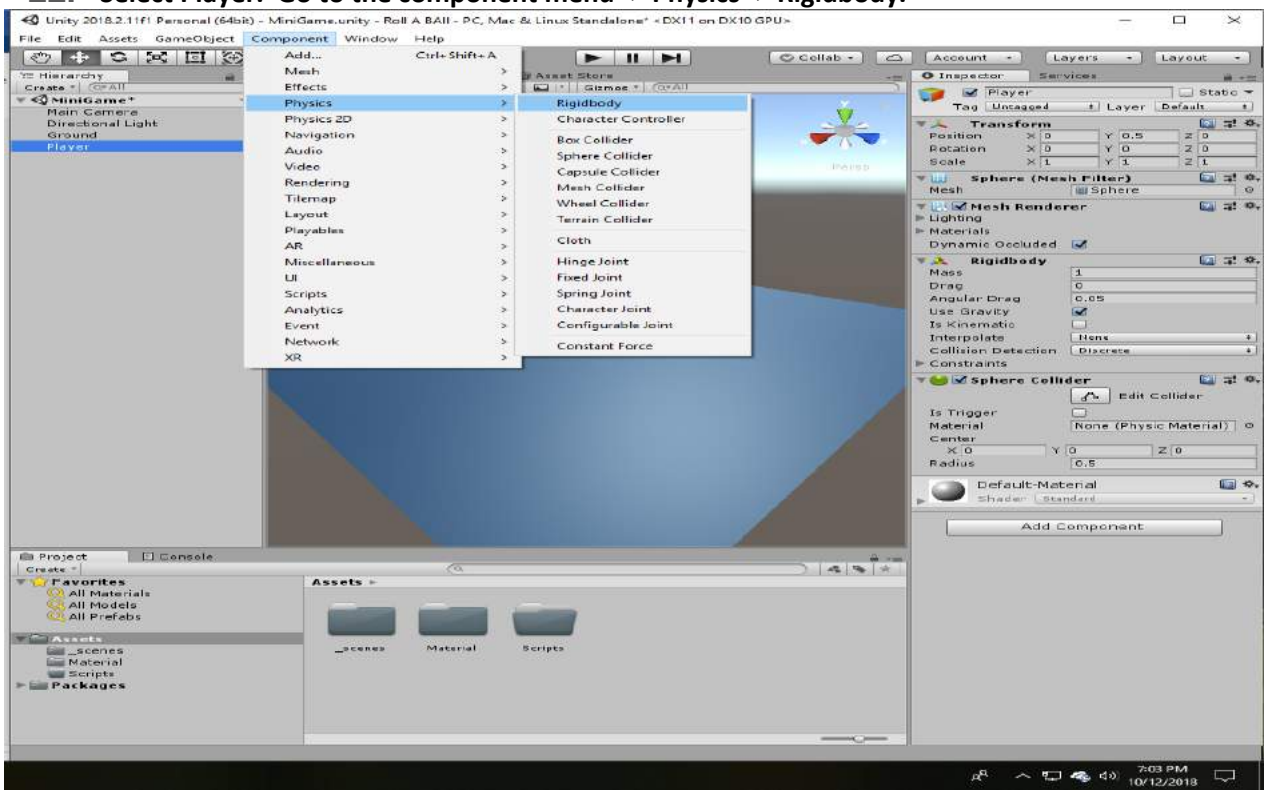
20. **To apply the texture to the plane, simply select the material in the project view and drag it on to the plane in the scene view. Now the player stands out on the dark blue background.**



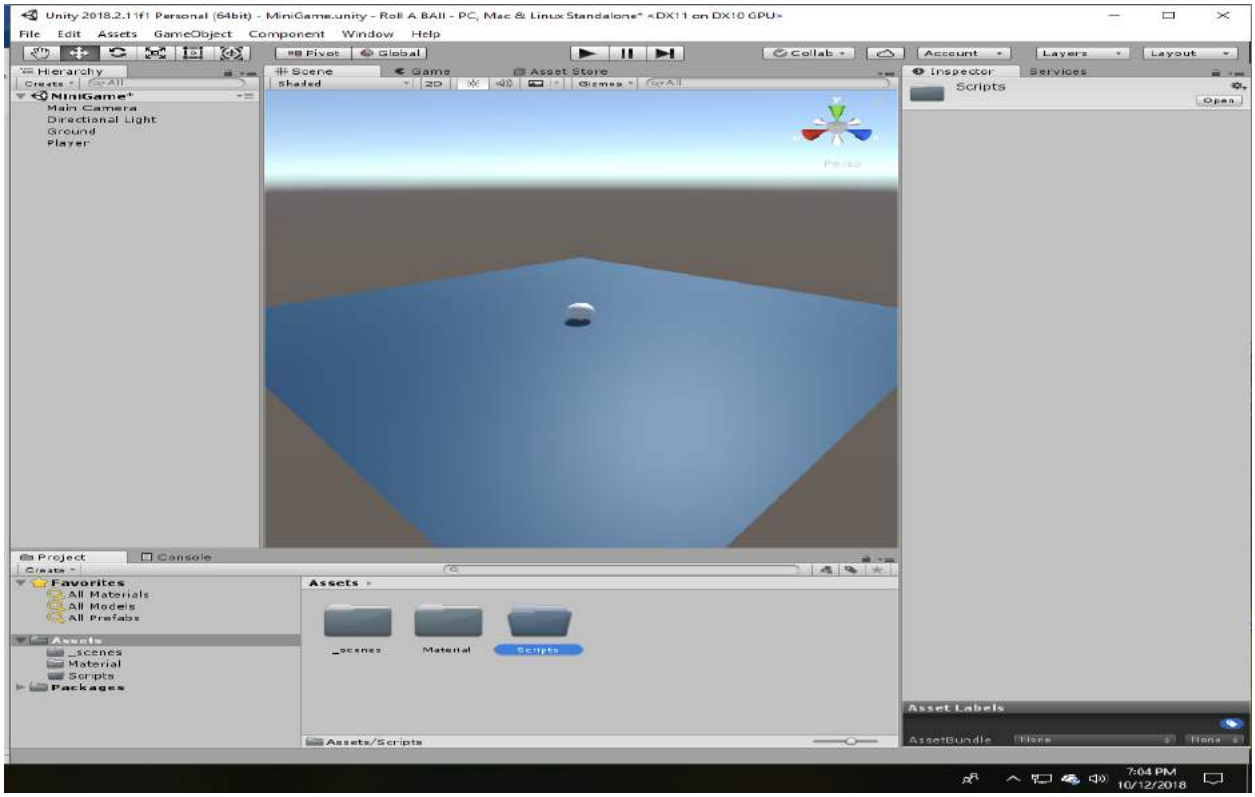
21. **Rotate the main directional light so that there is better lighting on the player. Select the directional light and in the transform component change the Transform Rotation on the Y-axis to 60**.



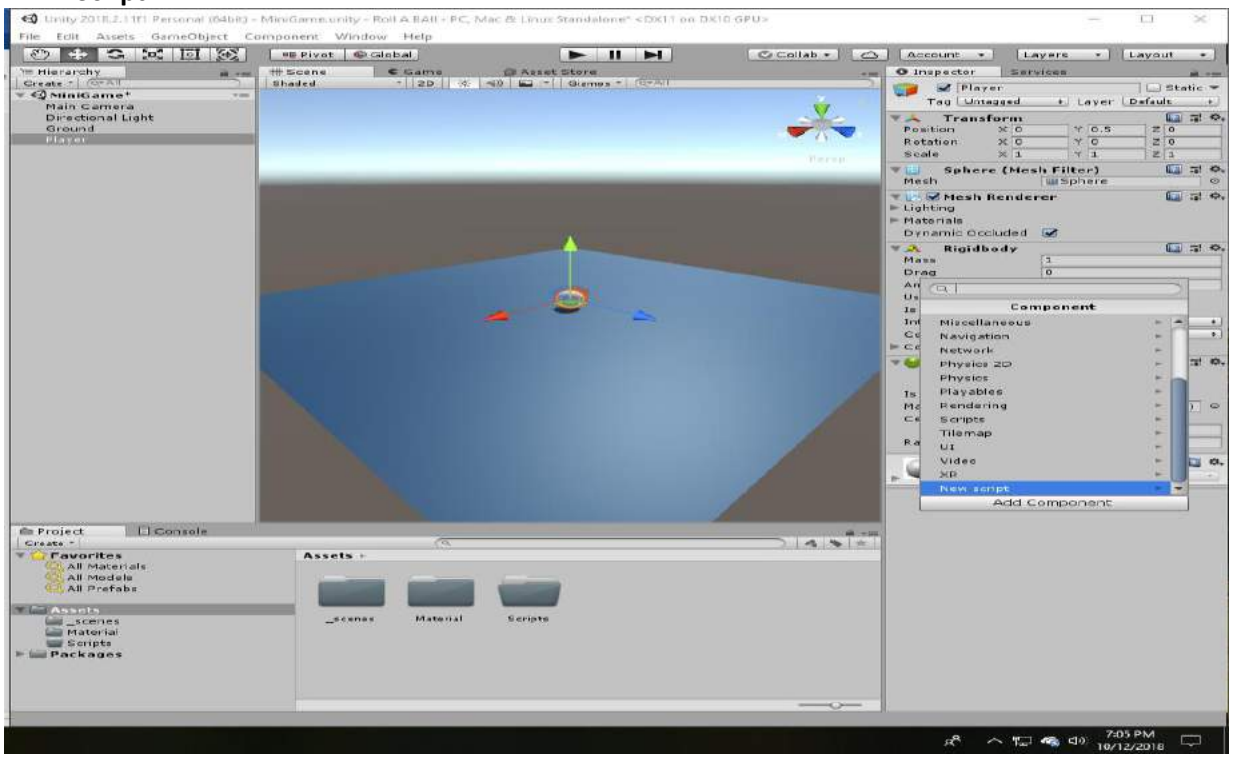
22. Select Player. Go to the component menu -> Physics -> Rigidbody.



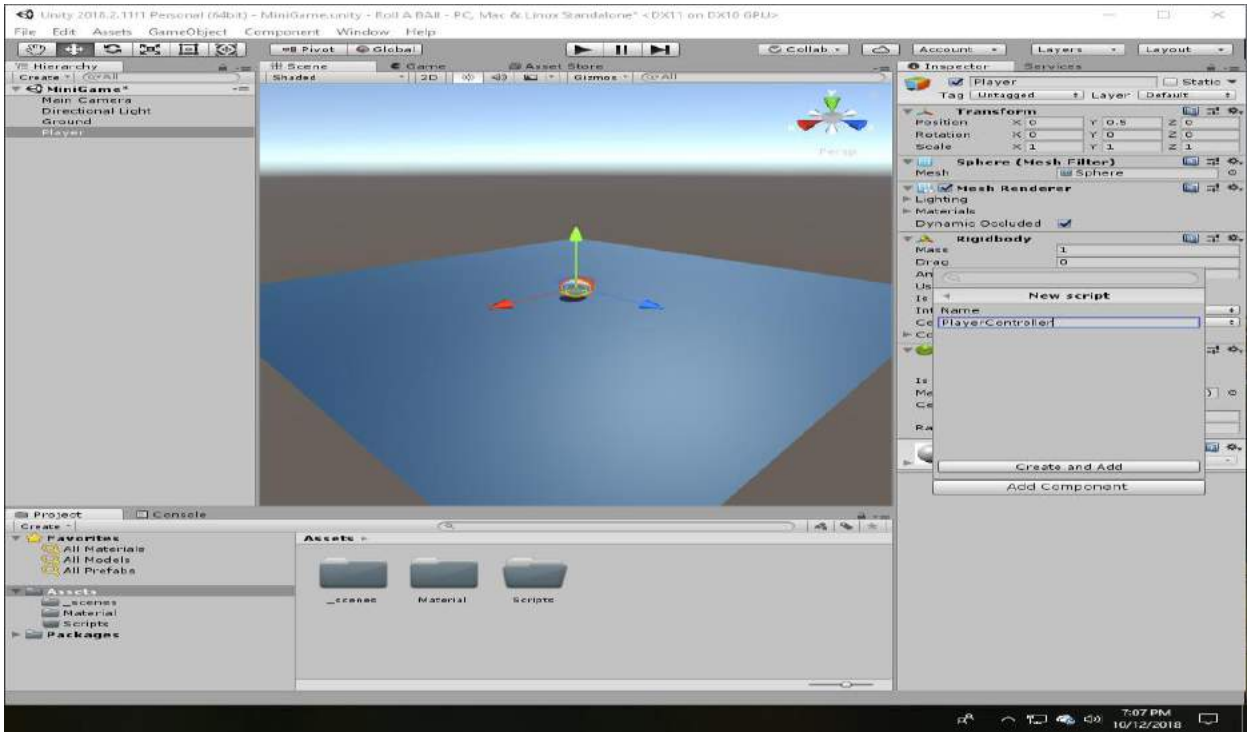
23. Create a folder by clicking on Project View -> folder. Name it Scripts.



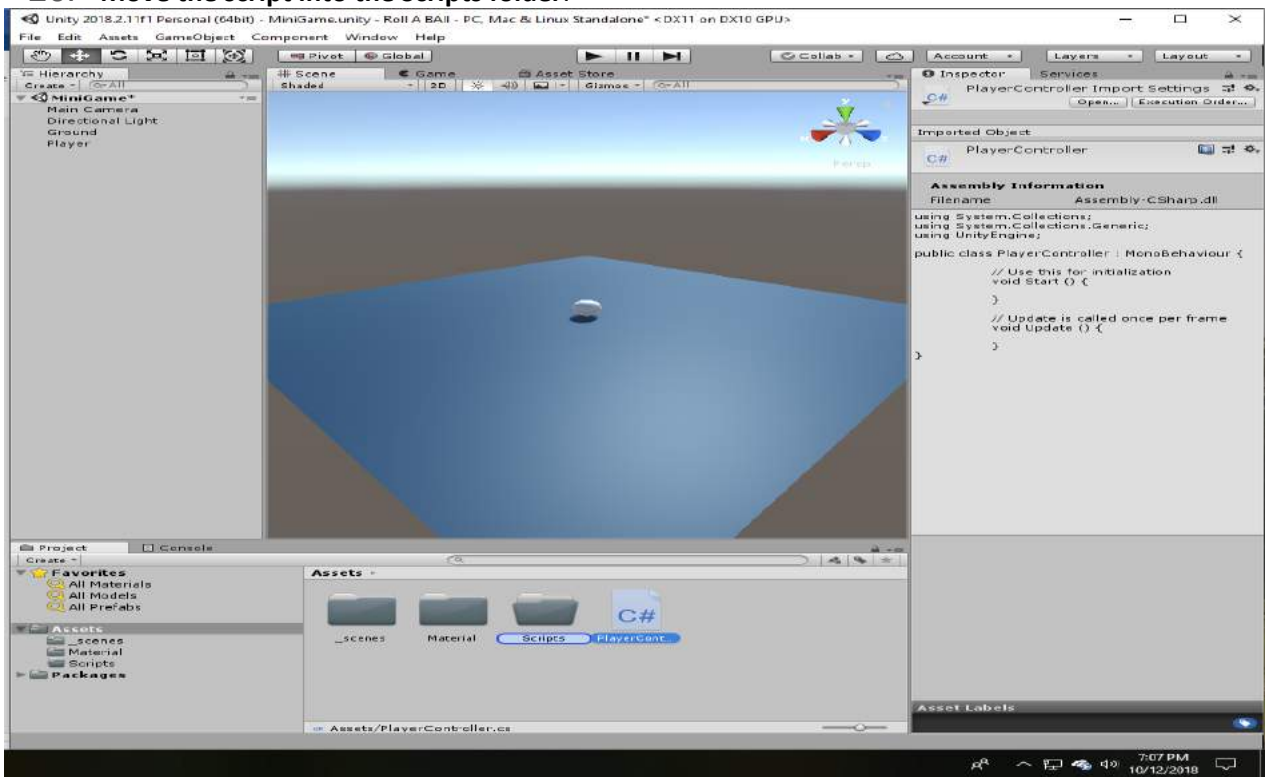
24. **Create the script. Select Player -> Click on Add component in the Inspector section -> New Script**.



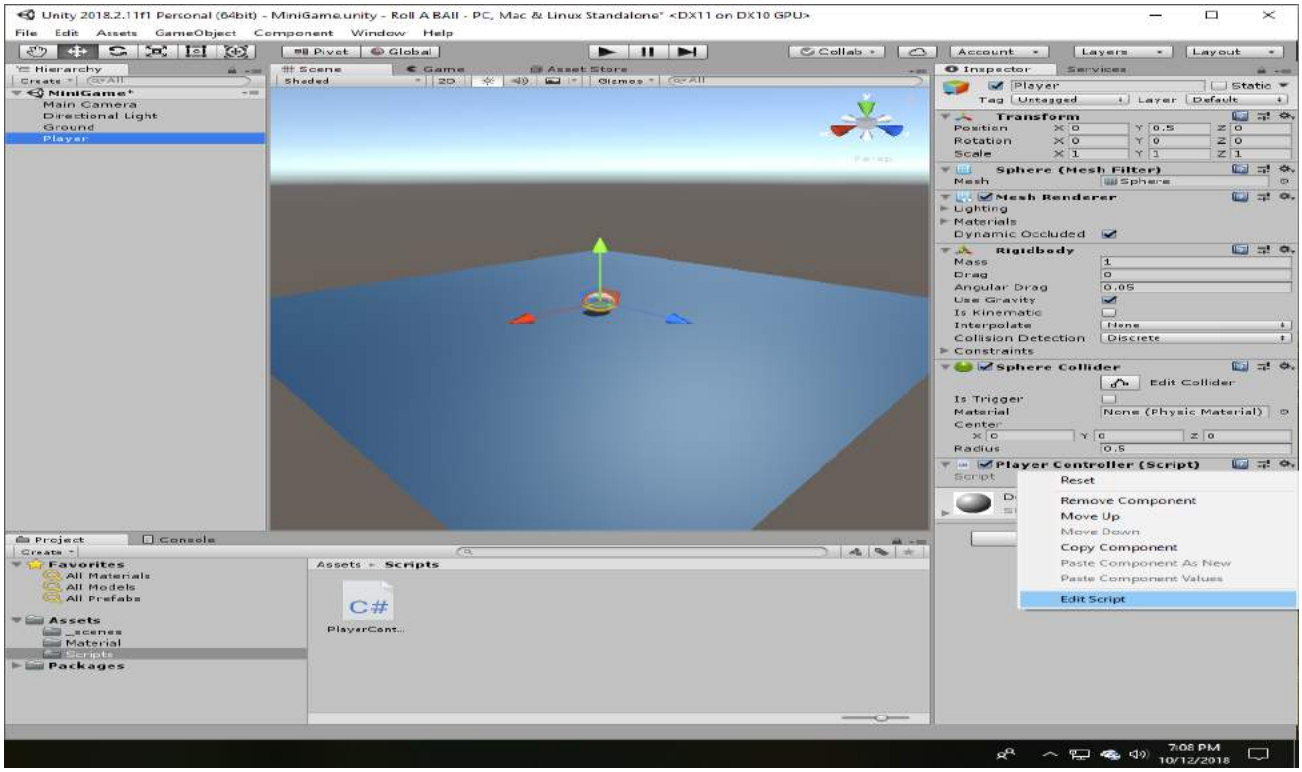
25. **Name the script, Player Controller. Choose C# as the language of the script. Click Create or hit the enter key.**



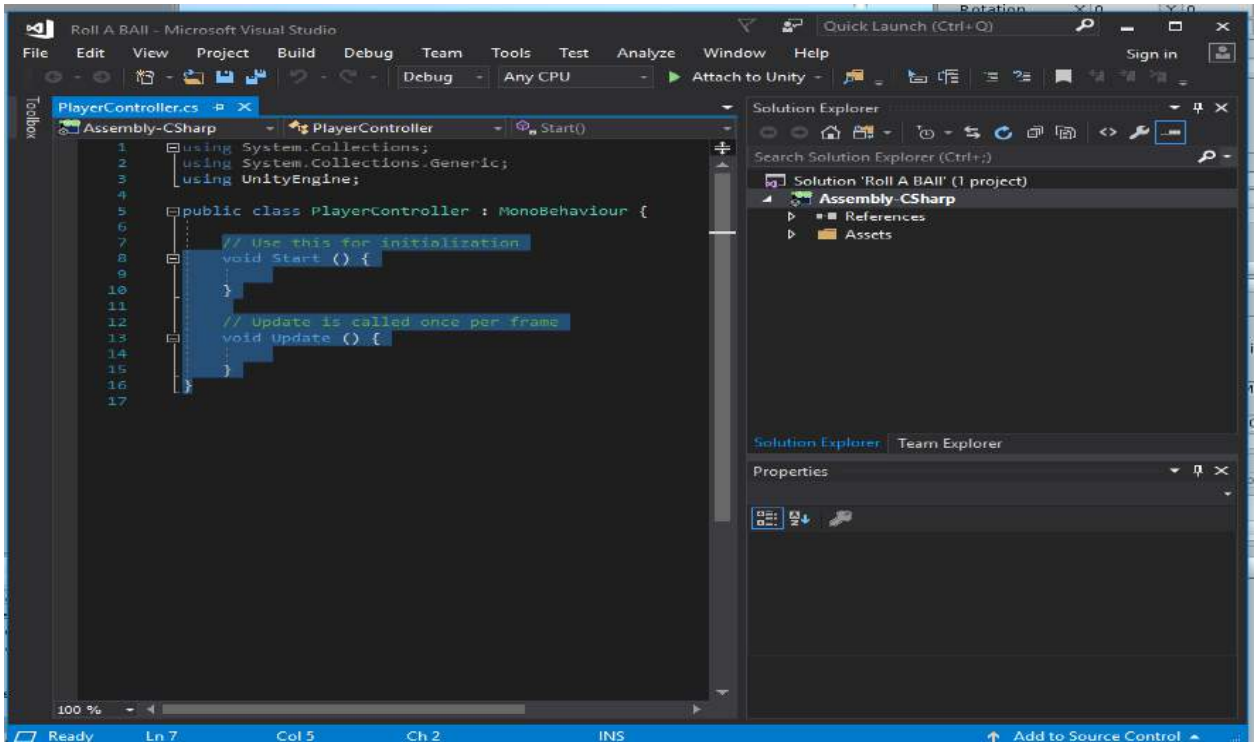
26. **Move the script into the Scripts folder**.



27. **Open the script. Go to Player -> Context sensitive Gear menu -> double click or edit script**.



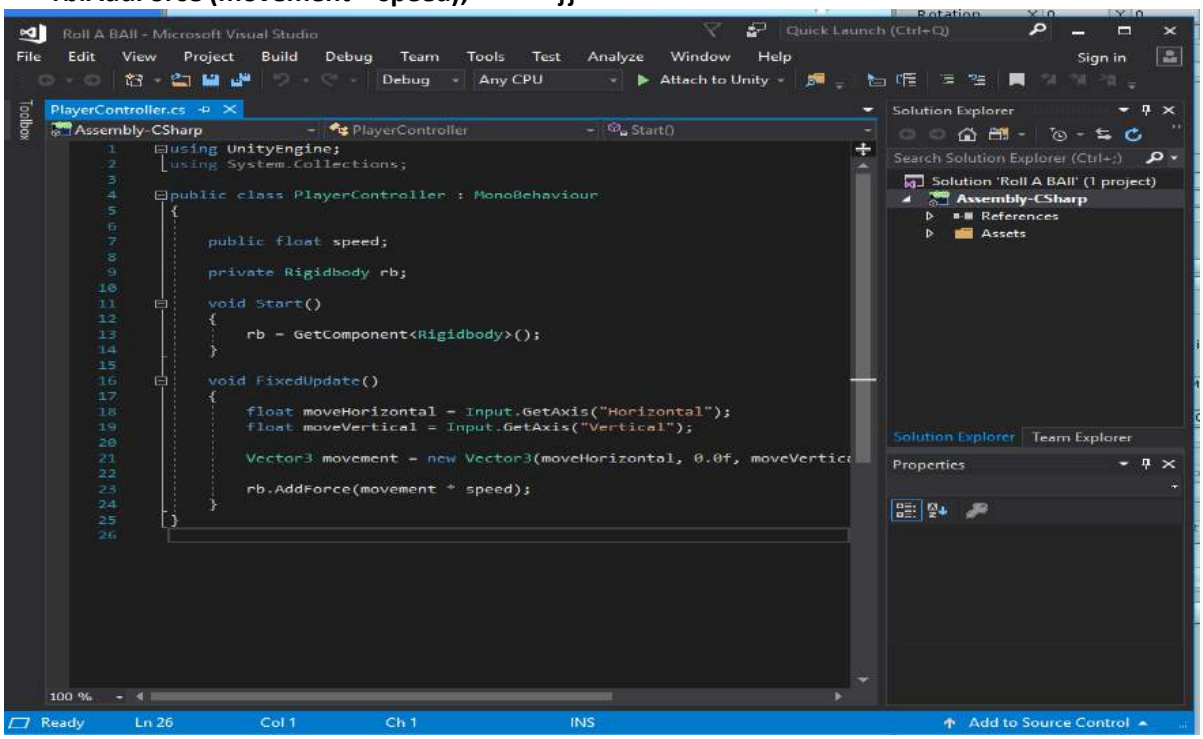
28. Remove the selected area.



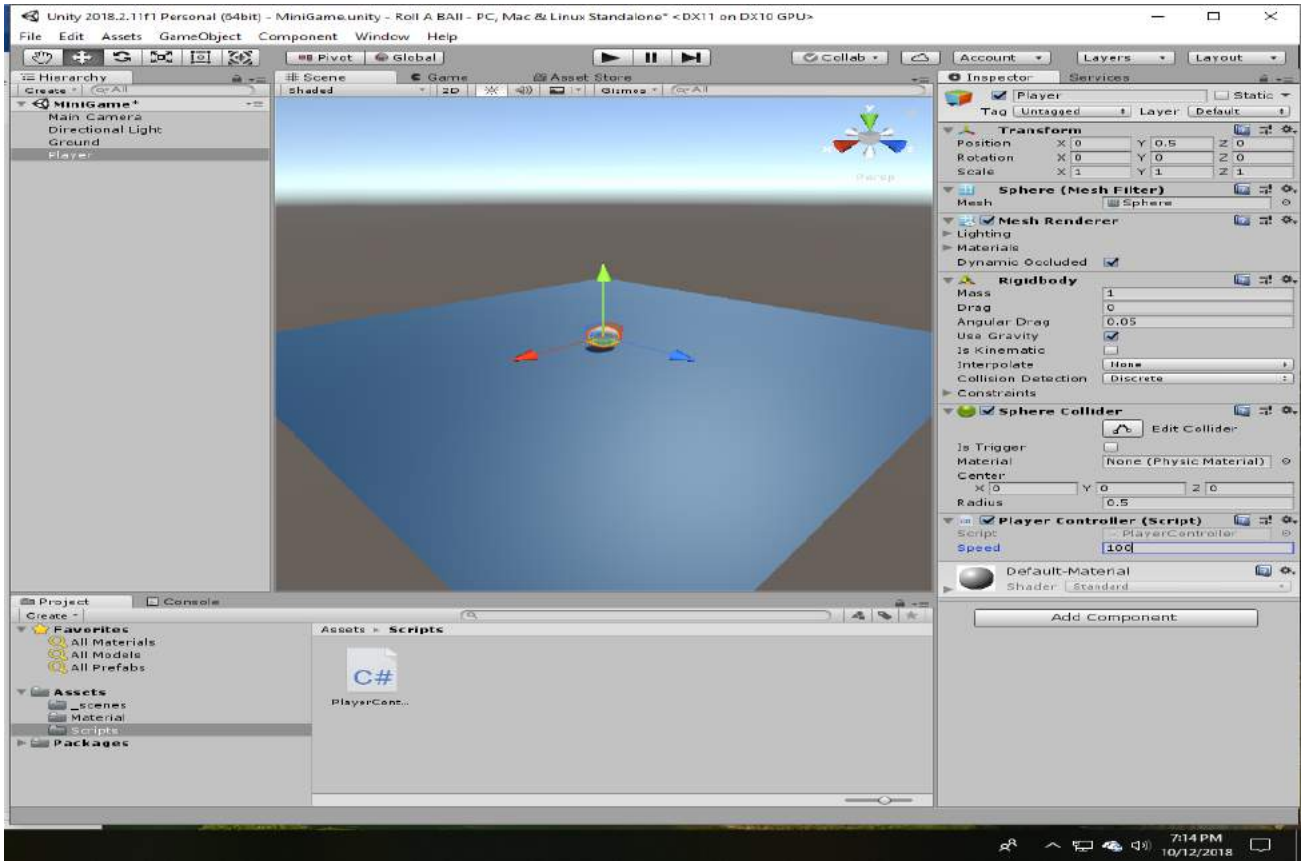
29. Enter the below code

```
using UnityEngine;
using System.Collections;
public class PlayerController : MonoBehaviour {
public float speed;
private Rigidbody rb;
void Start ()        {
```

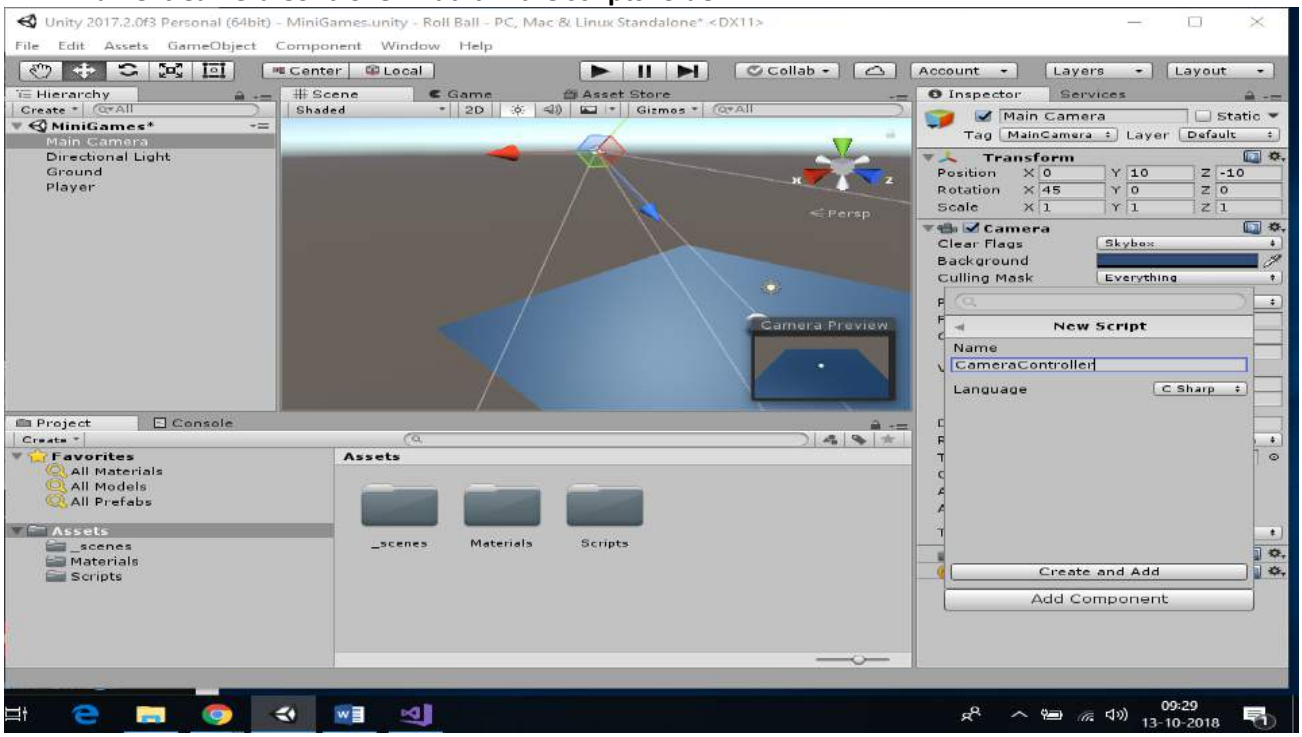
rb = GetComponent<Rigidbody>(); }
void FixedUpdate () {
float moveHorizontal = Input.GetAxis ("Horizontal");
float moveVertical = Input.GetAxis ("Vertical");
Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical);
rb.AddForce (movement * speed); }}



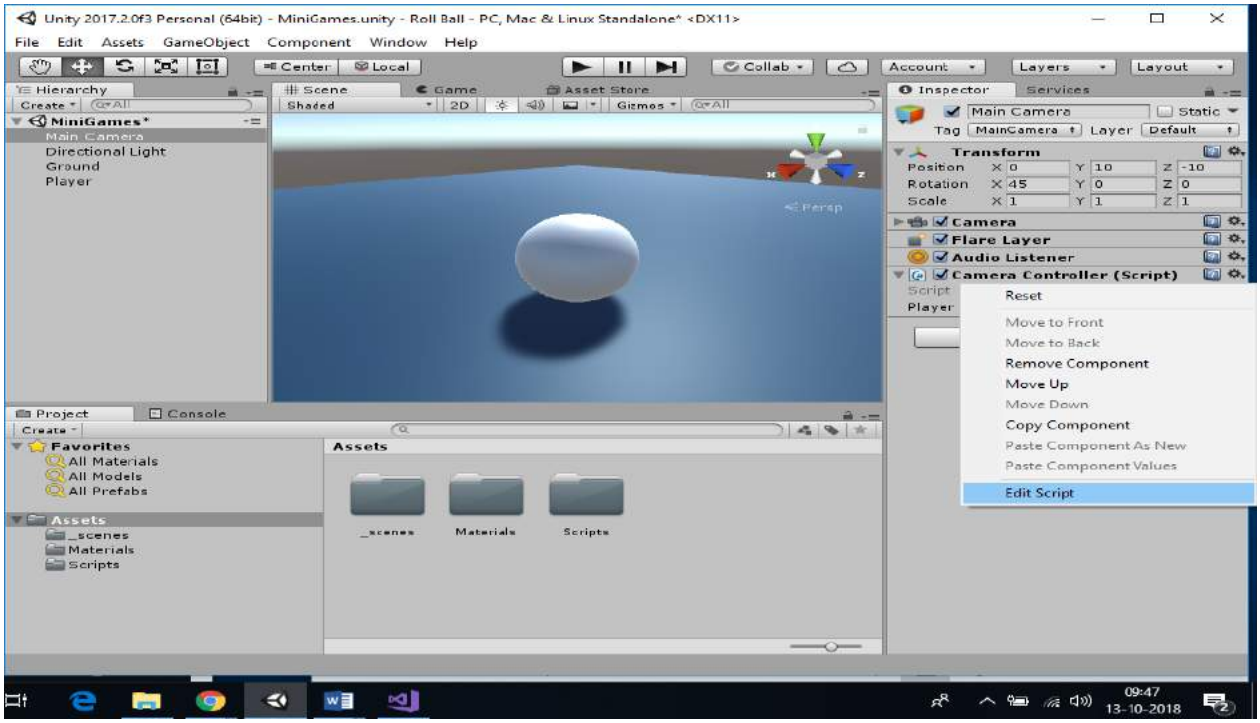
30. **Save and Return to Unity and change speed under Script component in the inspection section to 100.**



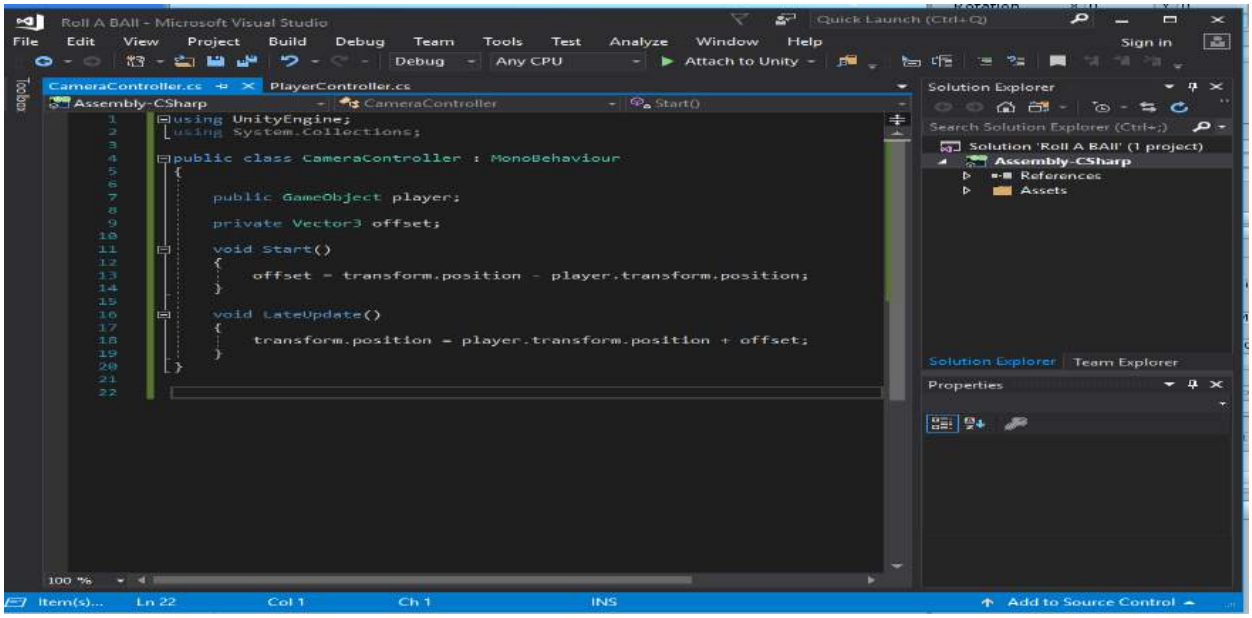
31. **Click on Main Camera and lift the camera by 10 units i.e. transform position y = 10 and tilt it by 45 i.e. transform rotate x = 45**
32. **Go to Add component button in inspection section -> Script, which is to be written in C#. Name it Camera Controller. Put it in the scripts folder**



33. **Open the script and put the following code**

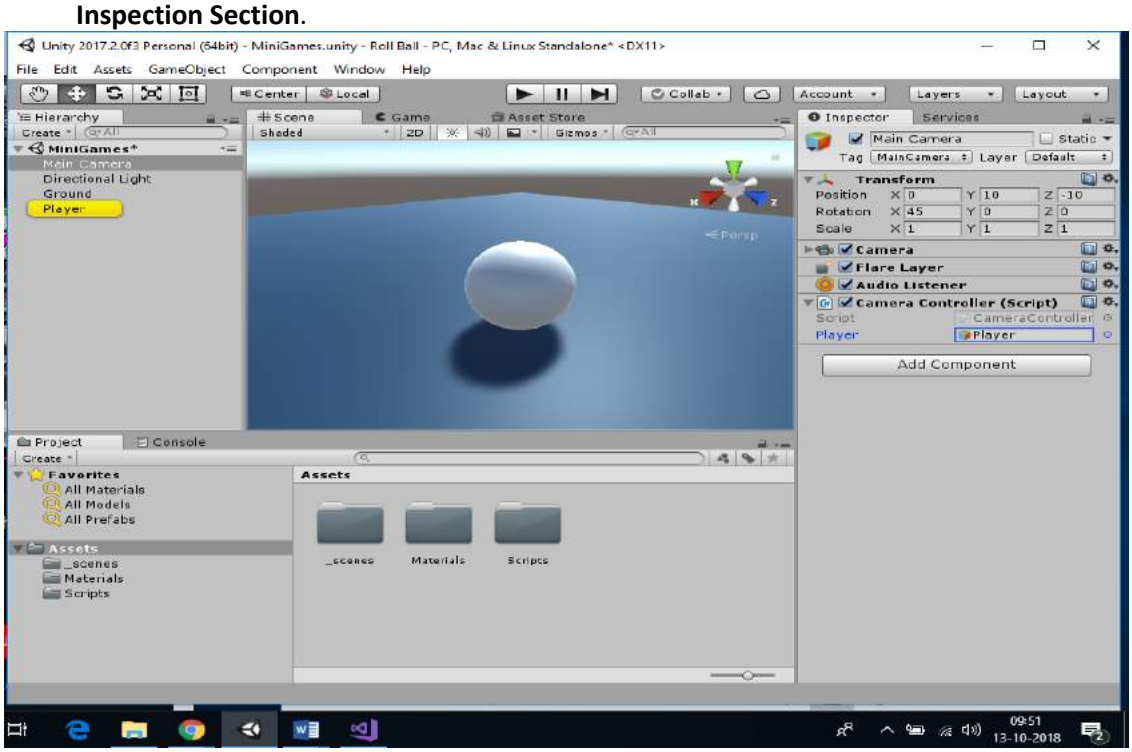



```
using UnityEngine;
using System.Collections;

public class CameraController : MonoBehaviour {

public GameObject player;

private Vector3 offset;

void Start ()
{
offset = transform.position - player.transform.position;
}

void LateUpdate ()
{
transform.position = player.transform.position + offset;
}
}
```

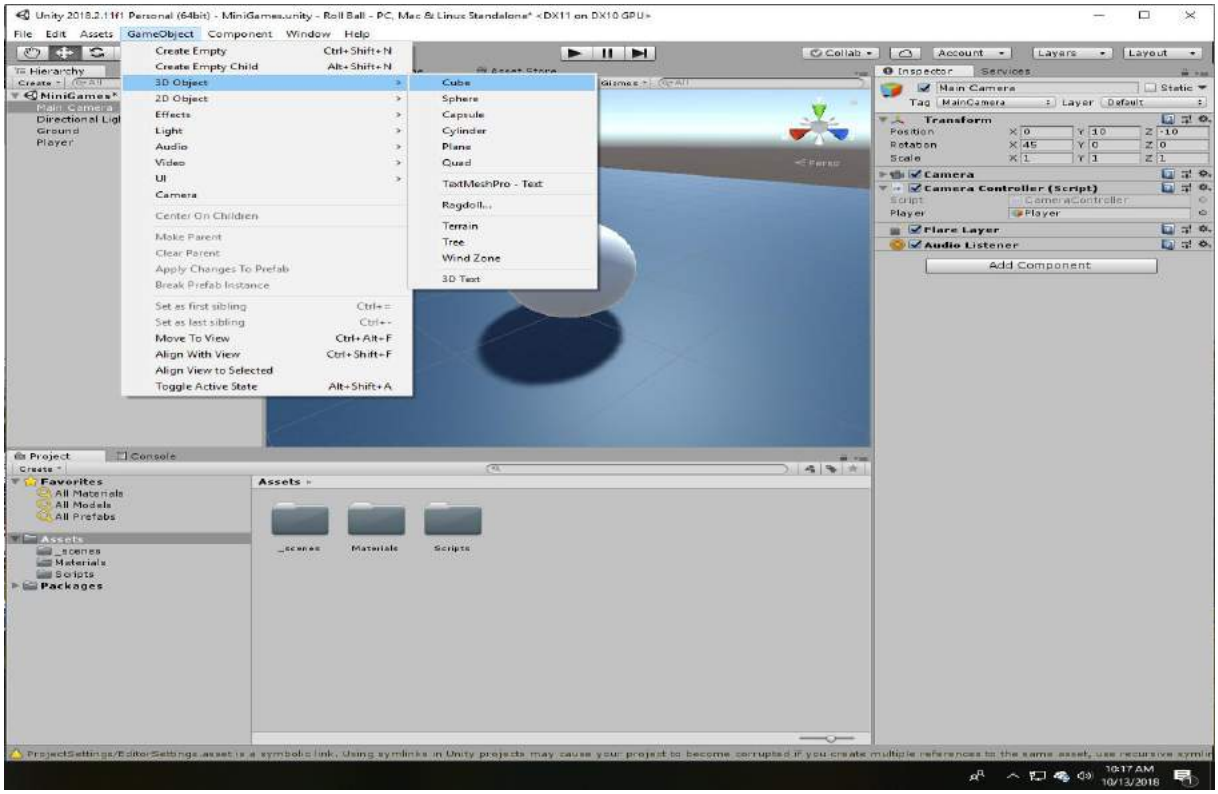


Save and return to unity.

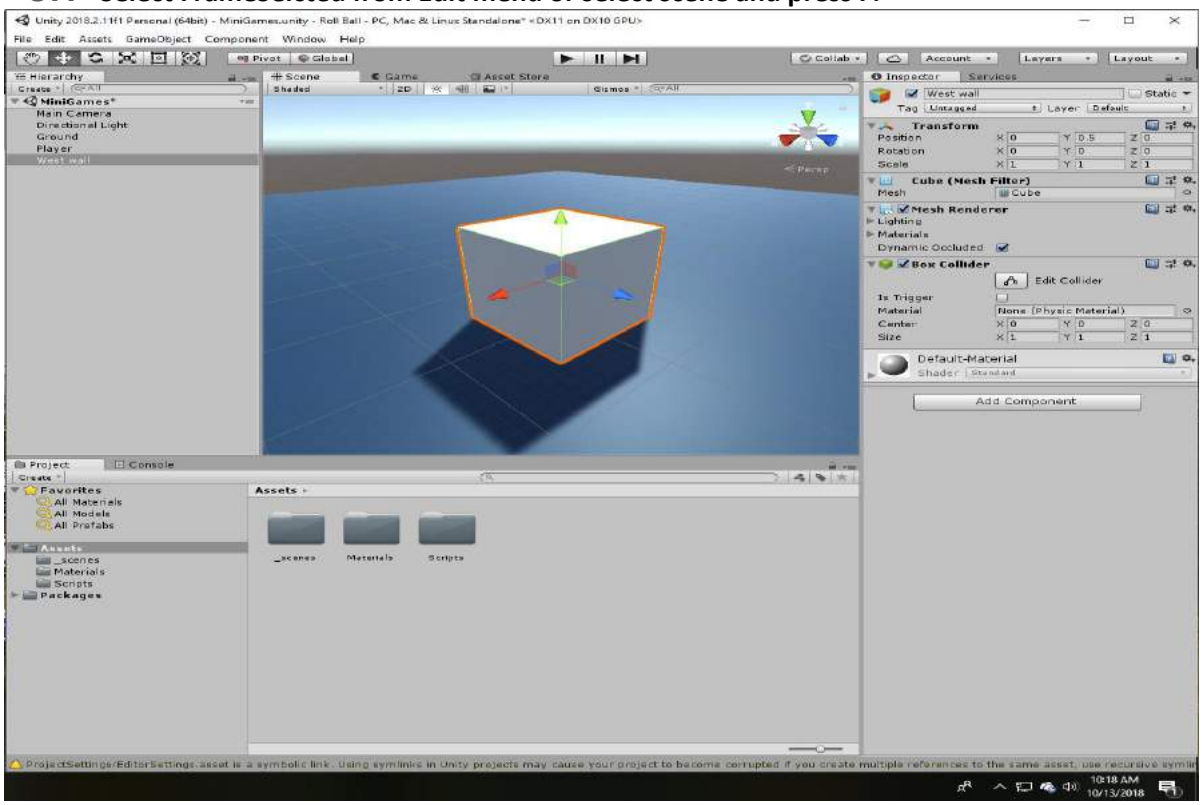
34. **Drag the payer to the Player section in the camera controller Script Component in Inspection Section.**



35. **Create a new game object and rename it Walls. Reset the values to origin.**
36. **Create GameObject -> 3D Object -> cube. Name it West Wall.**

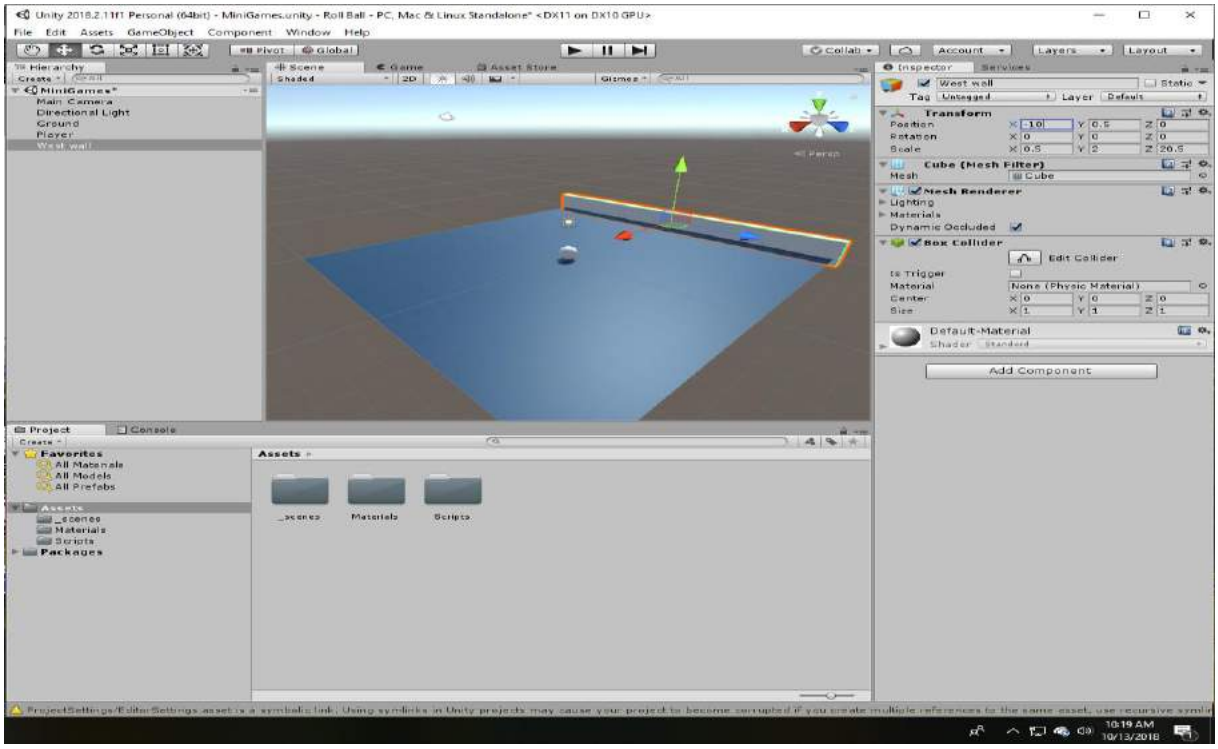


37. Select FrameSelcted from Edit menu or select scene and press F.

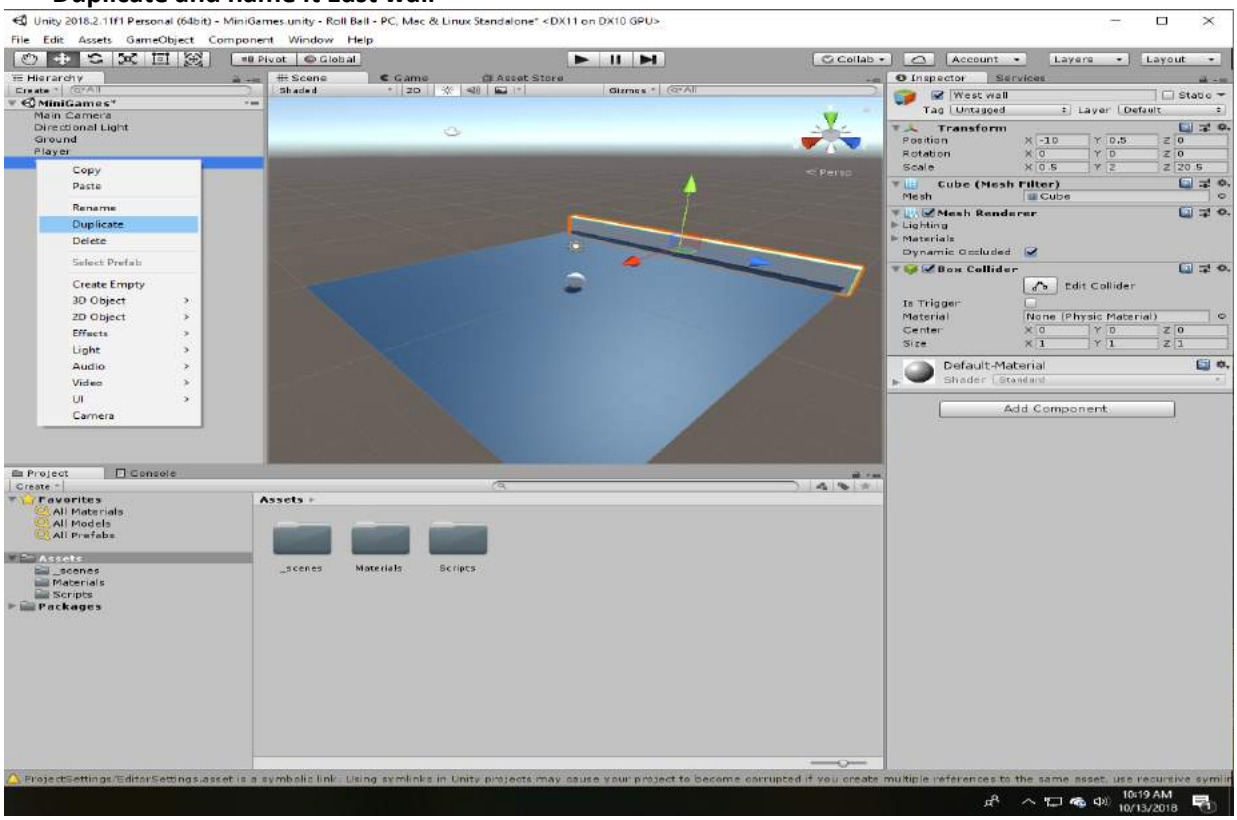


38. Change the cube's transform scale of X, Y and Z to 0.5 for thin, 2 for tall and 20.5 for long.

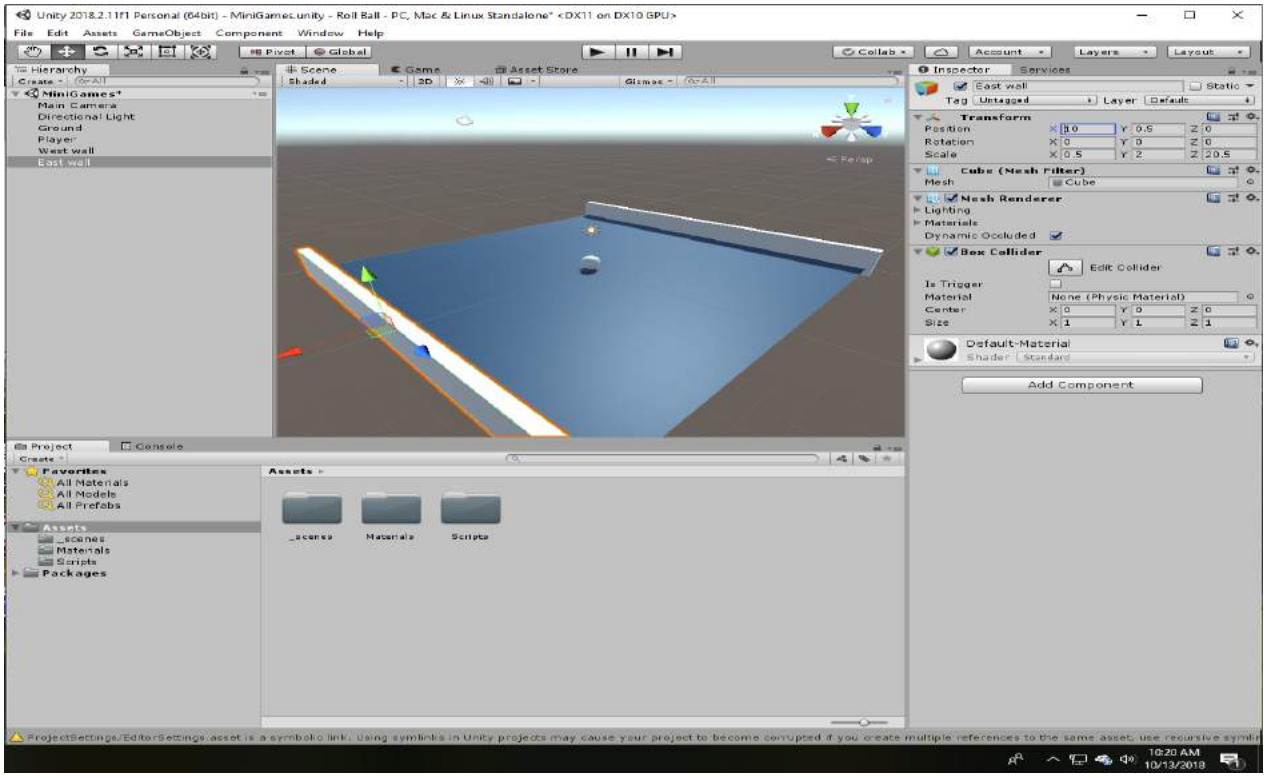
39. Set the transform's position X value to -10.



40. **Duplicate the West wall Game object. EDIT -> Duplicate or right click on West wall -> Duplicate and name it East wall**

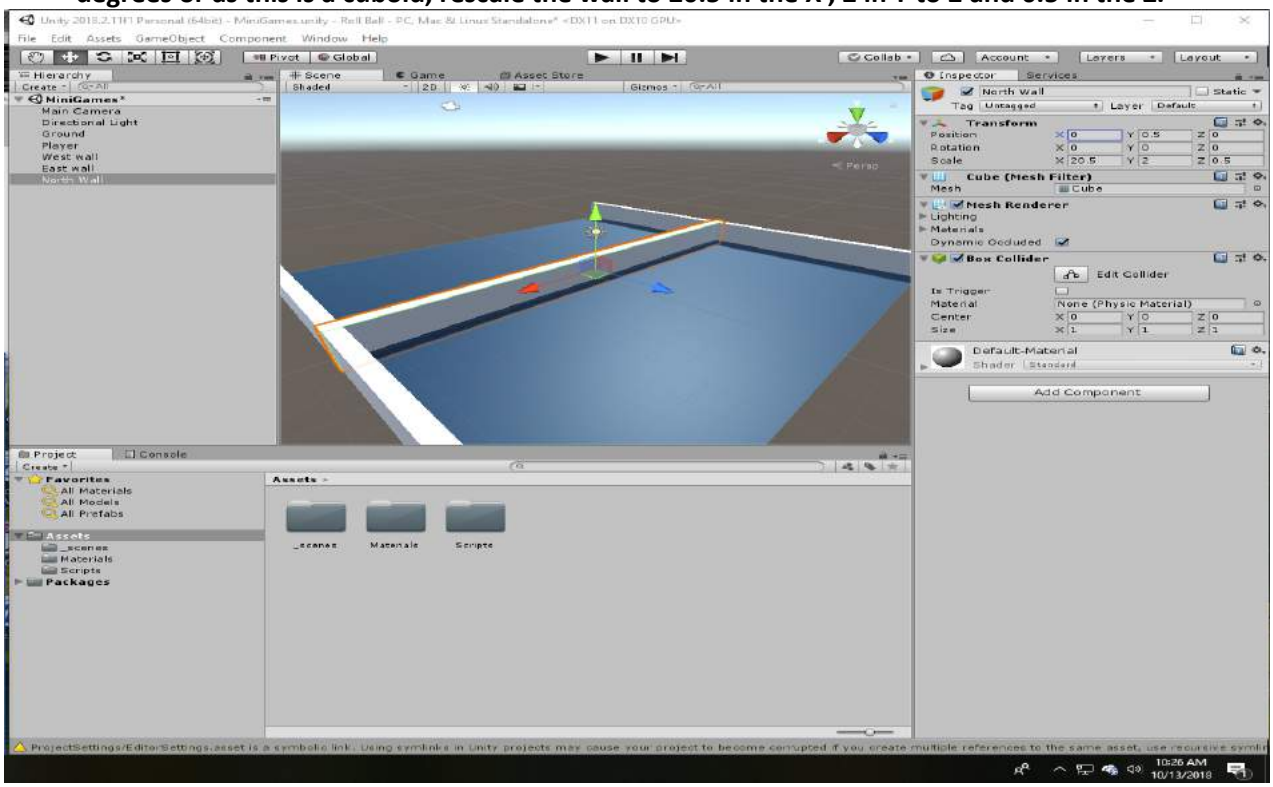


41. **To place the wall remove the negative sign in the transform's position X value.**

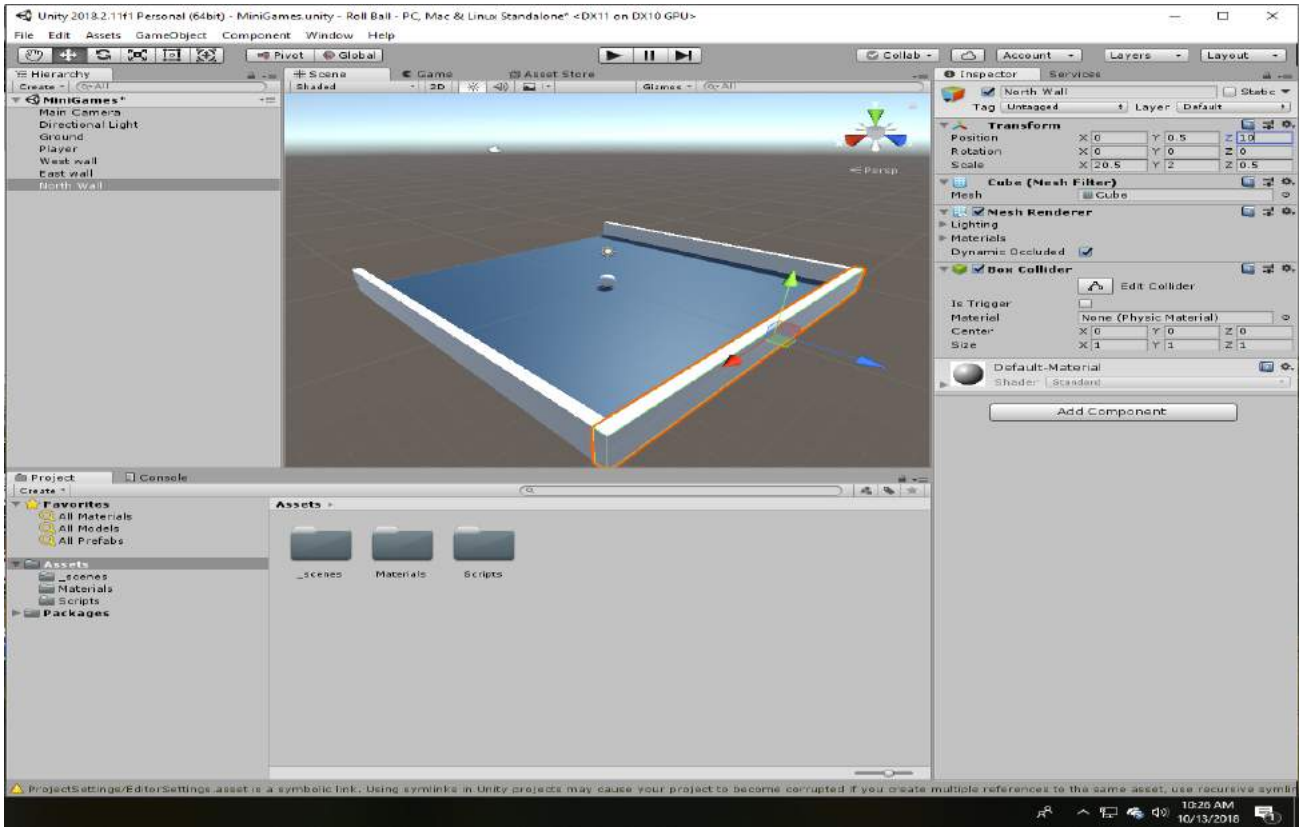


42. Duplicate the East wall Game object. EDIT -> Duplicate and name it North wall

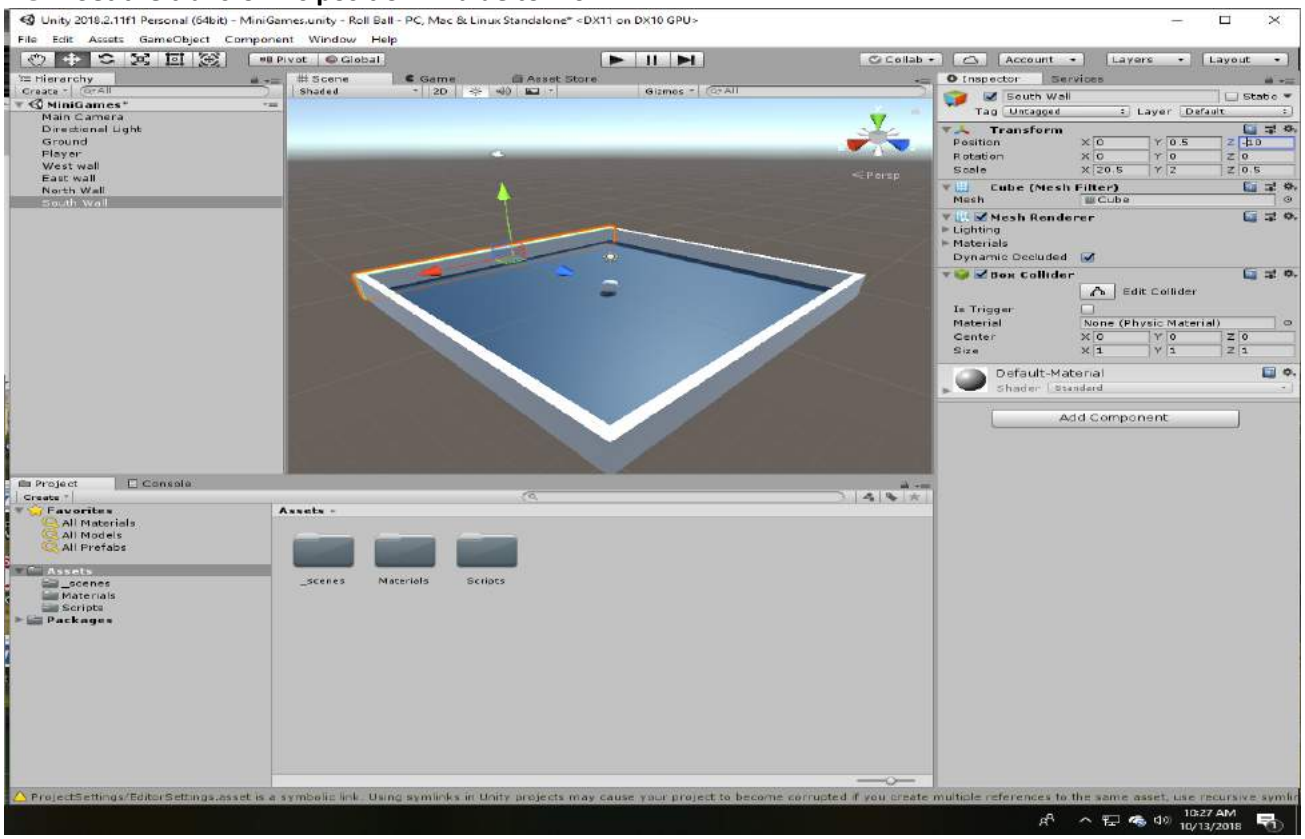
43. To place the wall reset the transform Component of the wall. Rotate the wall by 90 degrees or as this is a cuboid, rescale the wall to 20.5 in the X , 2 in Y to 2 and 0.5 in the Z.



44. Drag the wall in to place, or we can simply use the value of 10 in the transform's position Z field to place it.

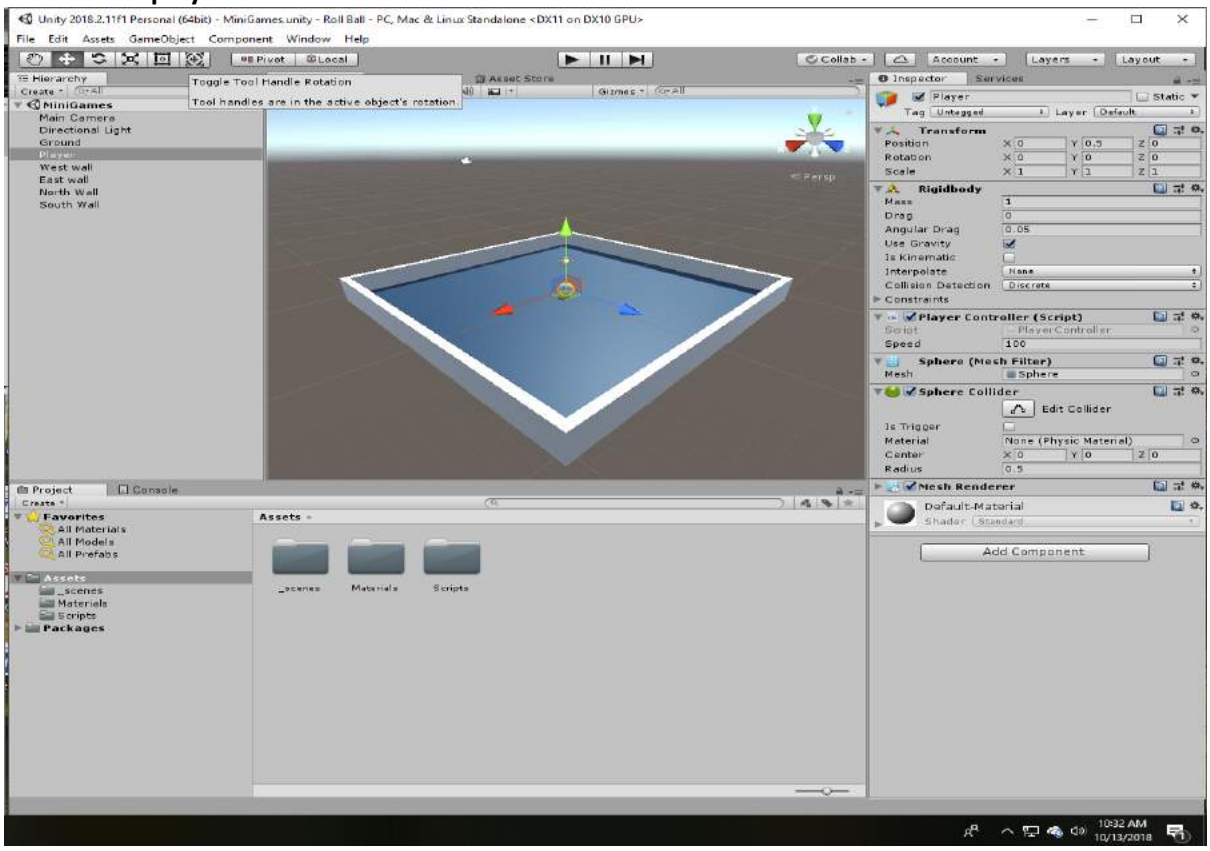


45. **Duplicate the North walGame object. EDIT -> Duplicate and name it South wall.**
46. **Set the transform's position Z value to -10.**

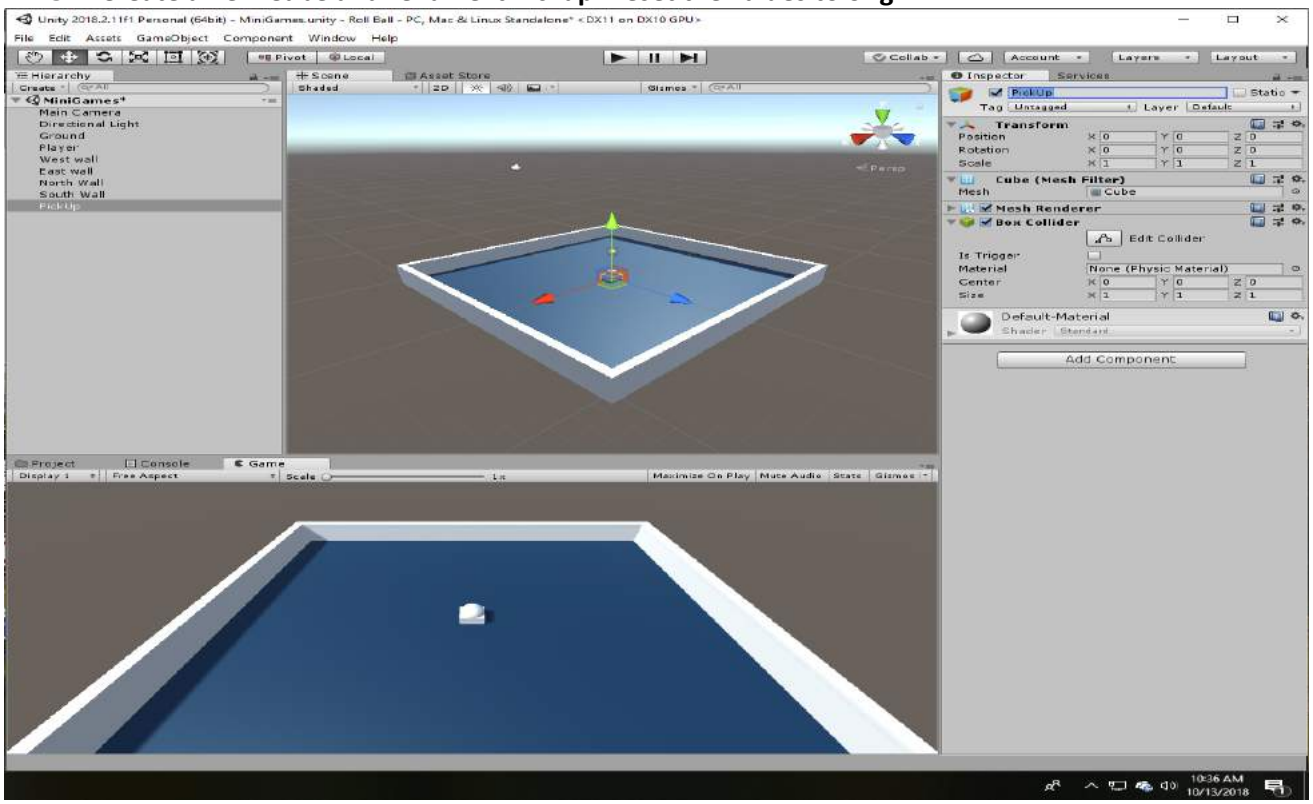


47. **Test the Program. Move the object by pressing the up-down keys on the keyboard. Then exit play mode.**

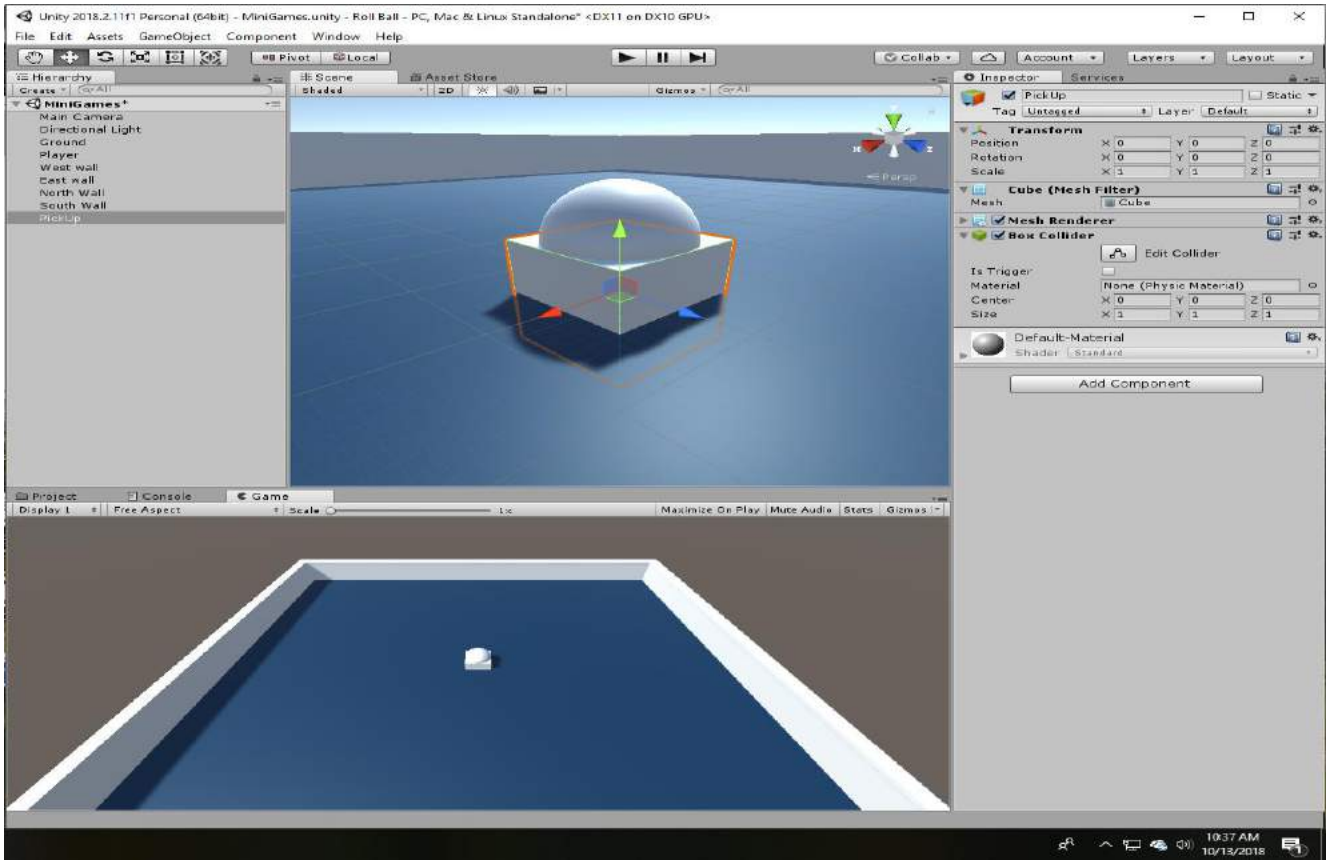
48. **Highlight the Player game object and set the editor to Local mode, this rotates the ball when play mode is entered.**



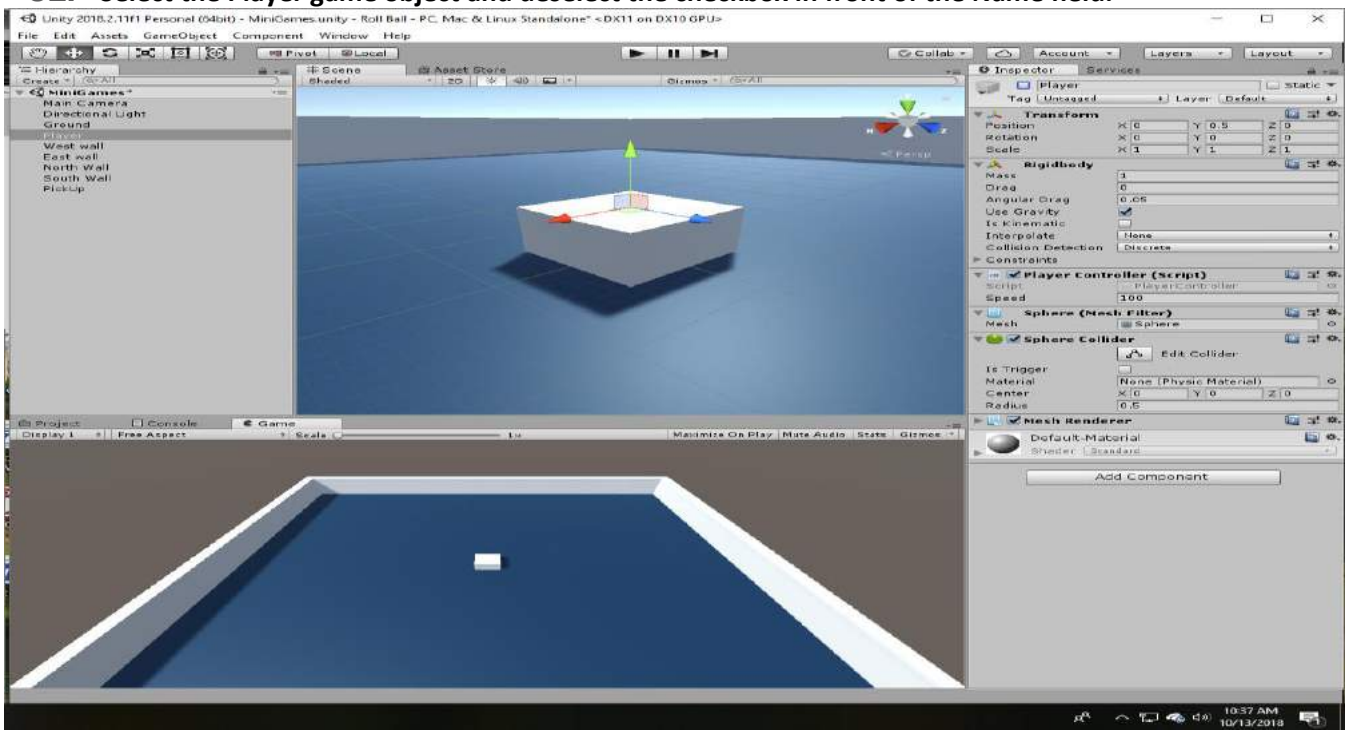
49. **Create a new Cube and rename it Pickup. Reset the values to origin**.



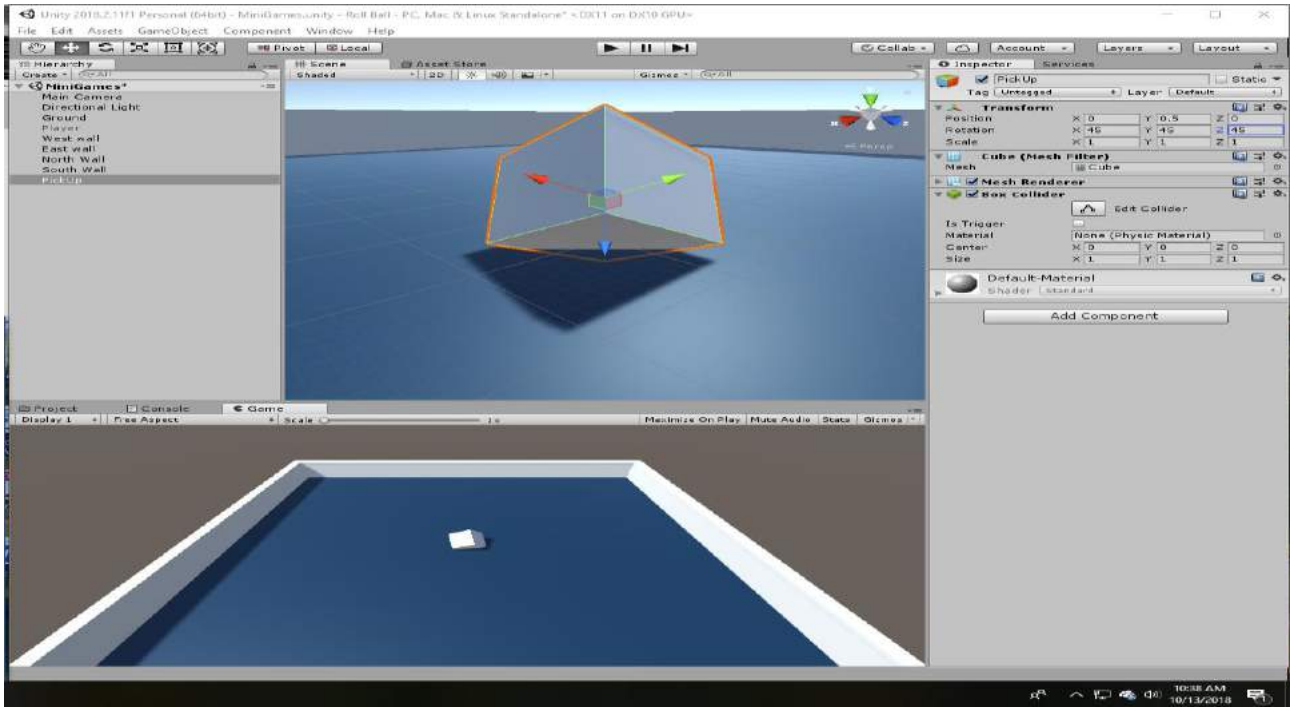
50. **Focus the scene view camera on the Pickup object.**



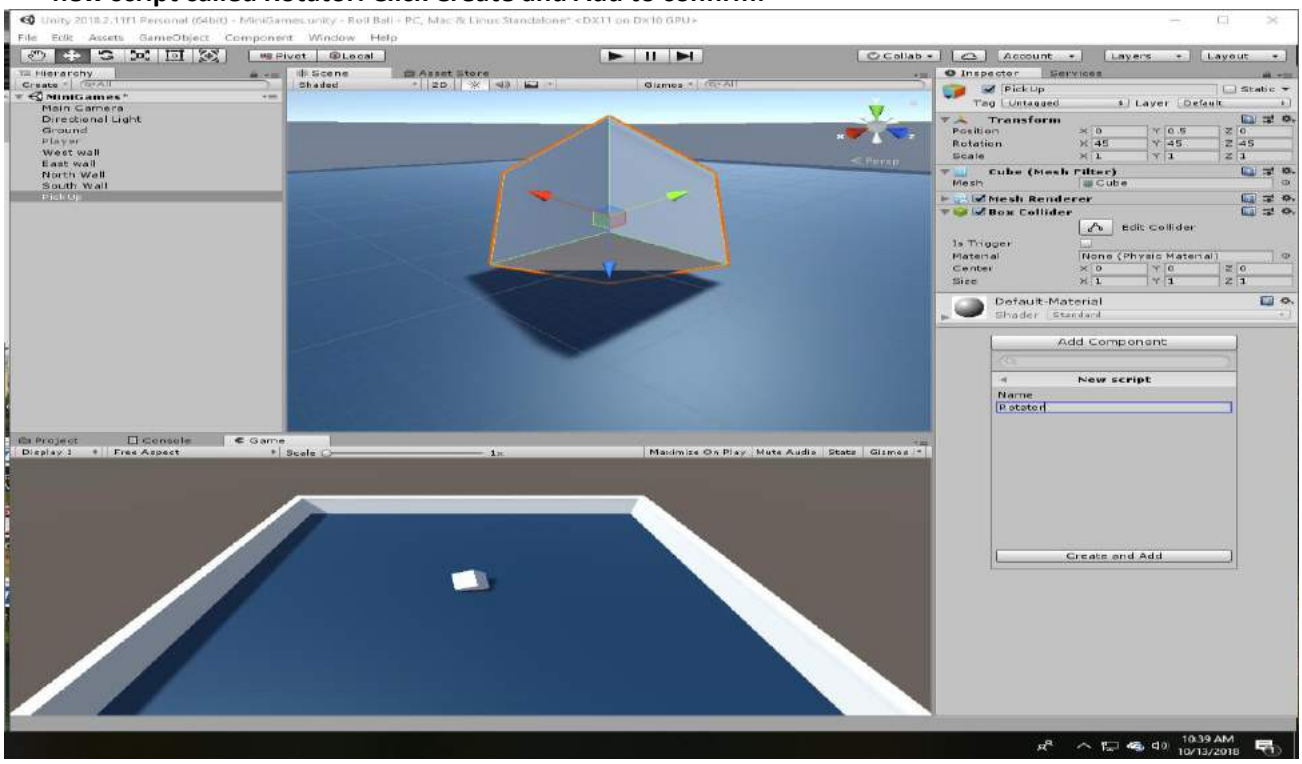
51. **Select the Player game object and deselect the checkbox in front of the Name field.**



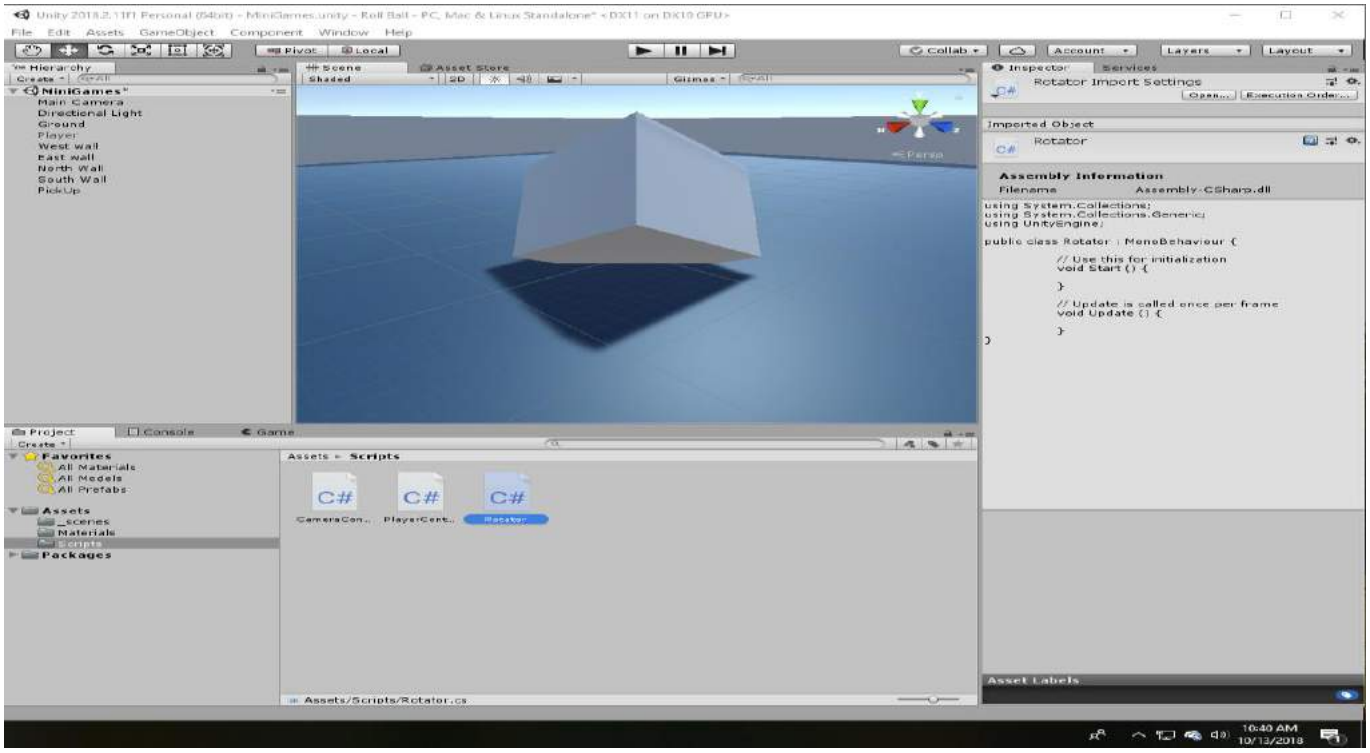
52. **The cube is also a regular shape, 1 by 1 by 1. So let's lift it up by half a unit**
53. **Tilt it over 45, 45, 45 in each of the axis of the transform's rotation**.



54. With the Pickup object selected use the Add Component button in the Inspector. Create a new script called Rotator. Click Create and Add to confirm.



55. Organize the script by placing it in the Scripts folder, and open it for editing.

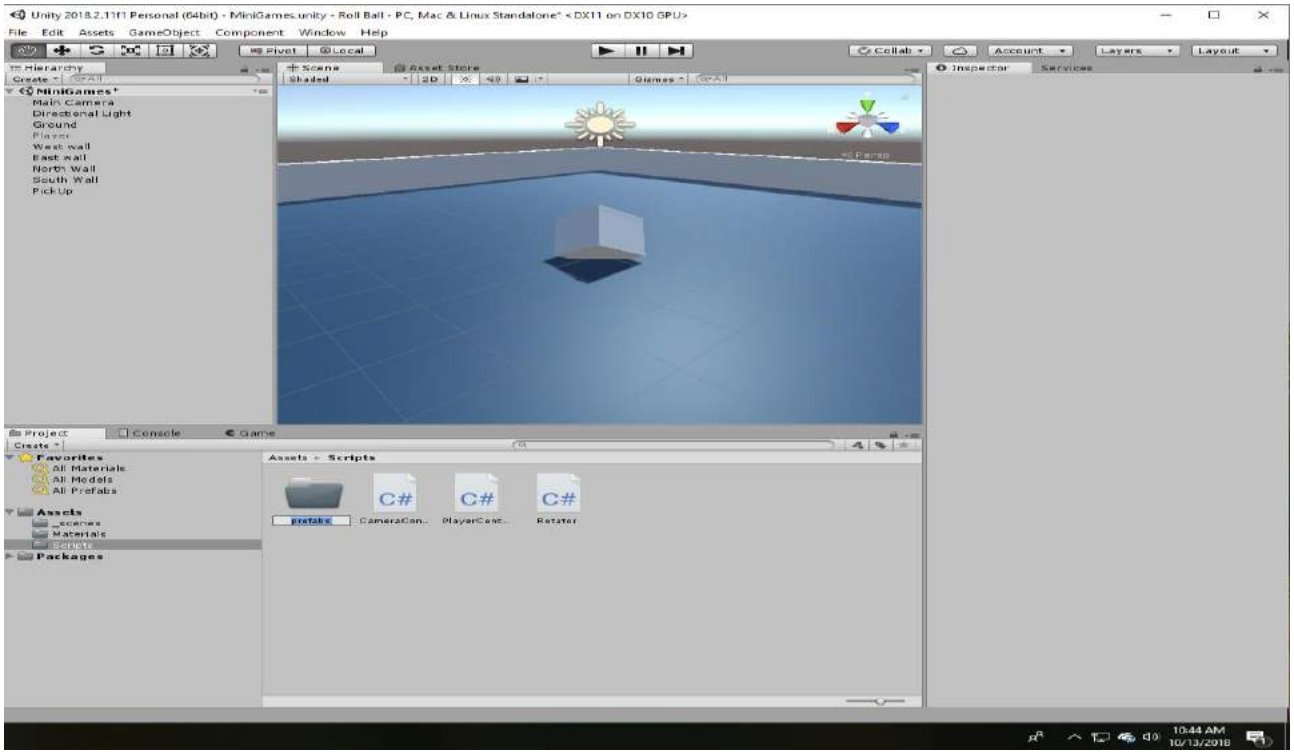


56. **Copy the code below**

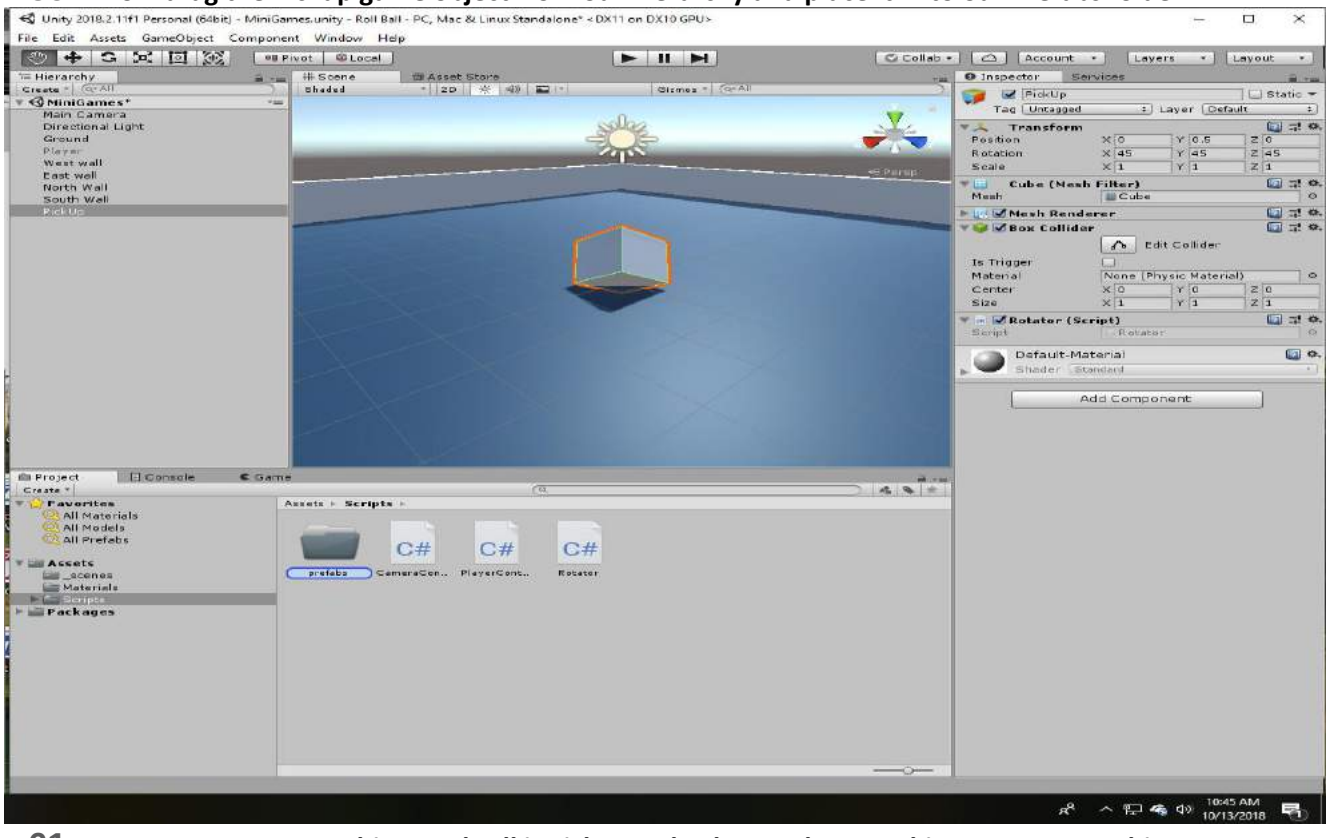
```
using UnityEngine;
using System.Collections;

public class Rotator : MonoBehaviour {

void Update ()
{
transform.Rotate (new Vector3 (15, 30, 45) * Time.deltaTime);
}
}
```

Save and return to Unity

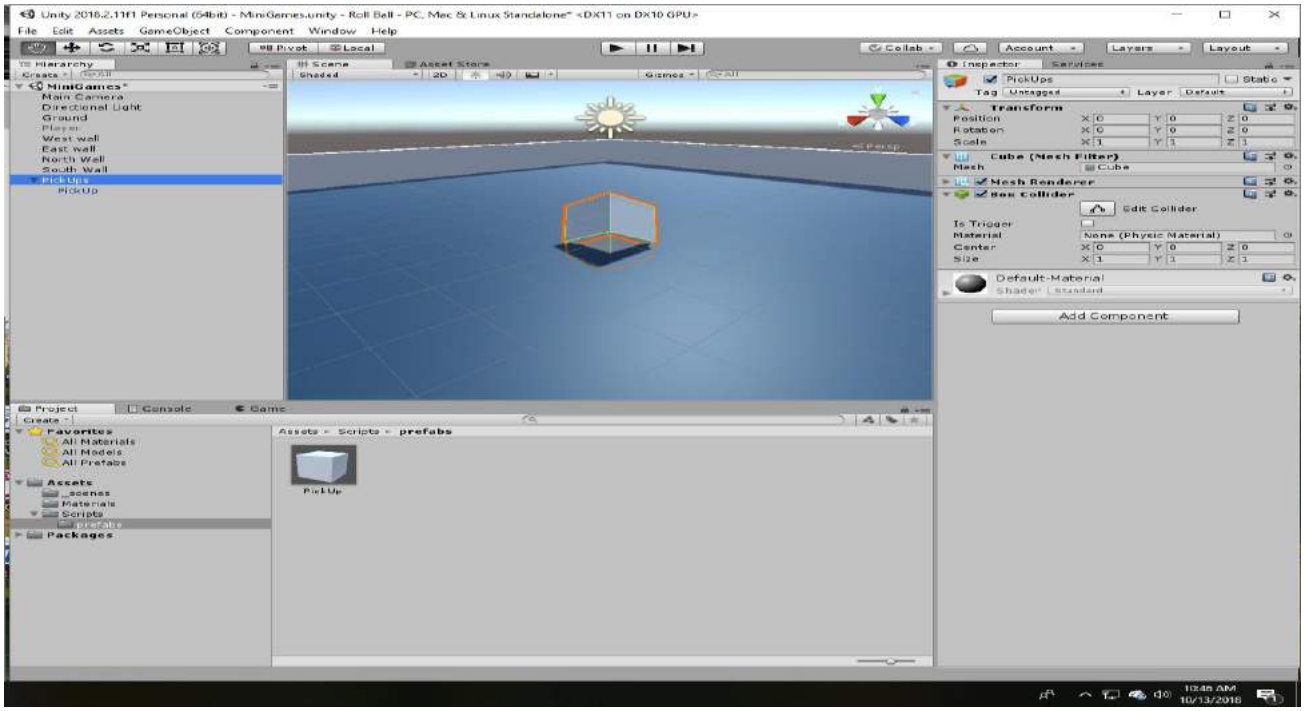
57. **Create a folder to hold prefabs.**
58. **This folder should be the root folder, or top level of our project. So select the project view and make sure that no other item or directory is highlighted.**
59. **Create – Folder Rename this folder Prefabs.**



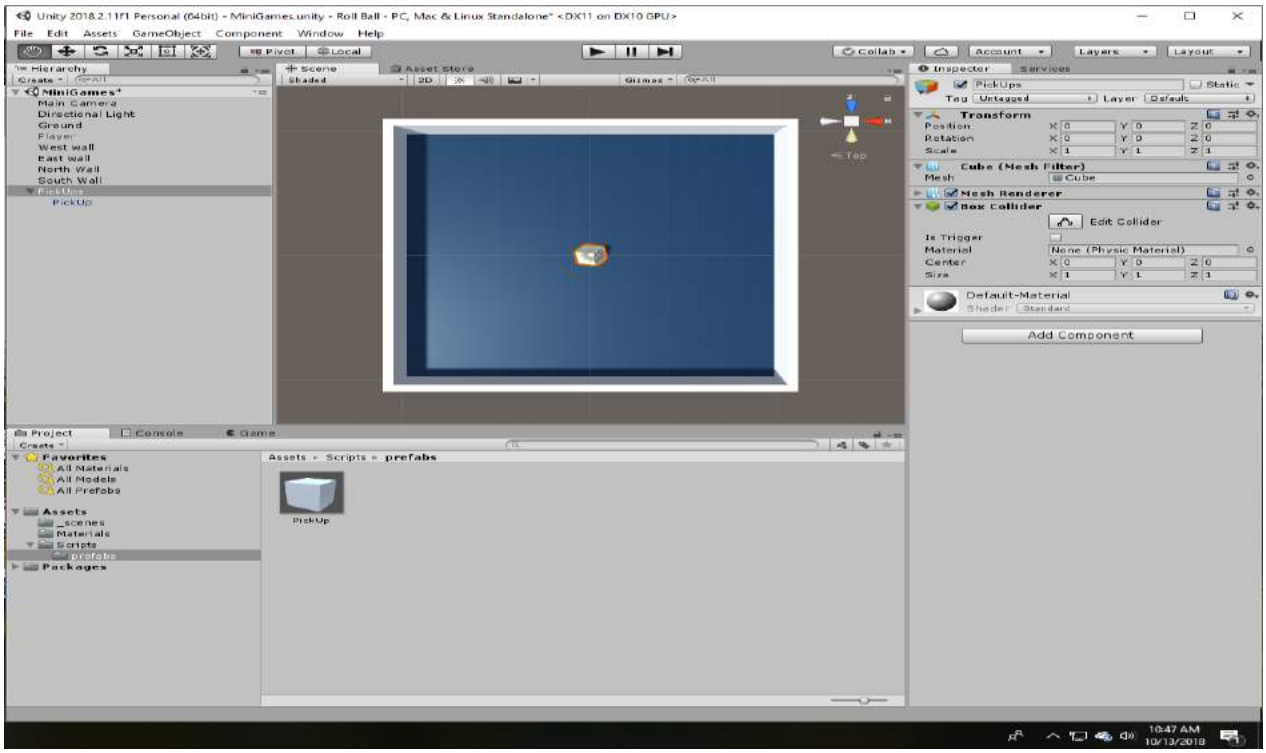
60. **Now drag the Pickup game object from our hierarchy and place it in to our Prefabs folder.**



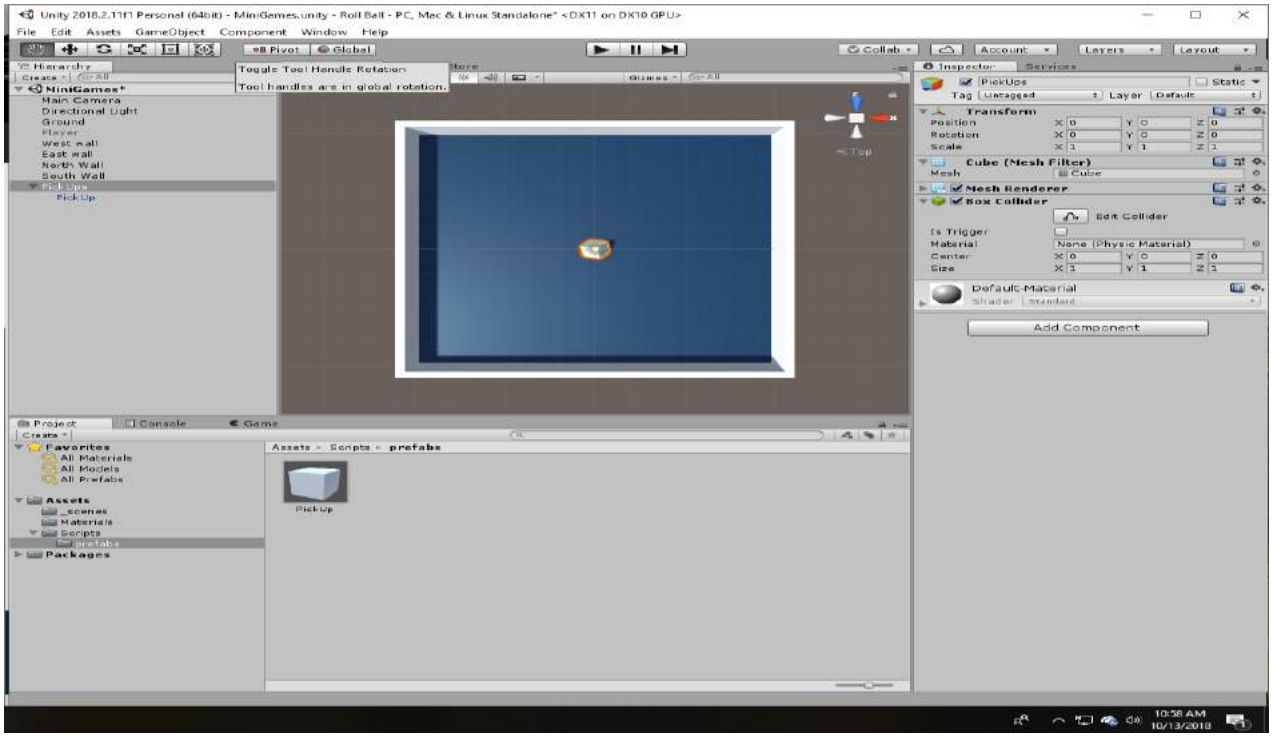
61. **Create a new game object and call it Pickups. Check to make sure this parent game object is at origin and drag our Pickup game object in to it.**



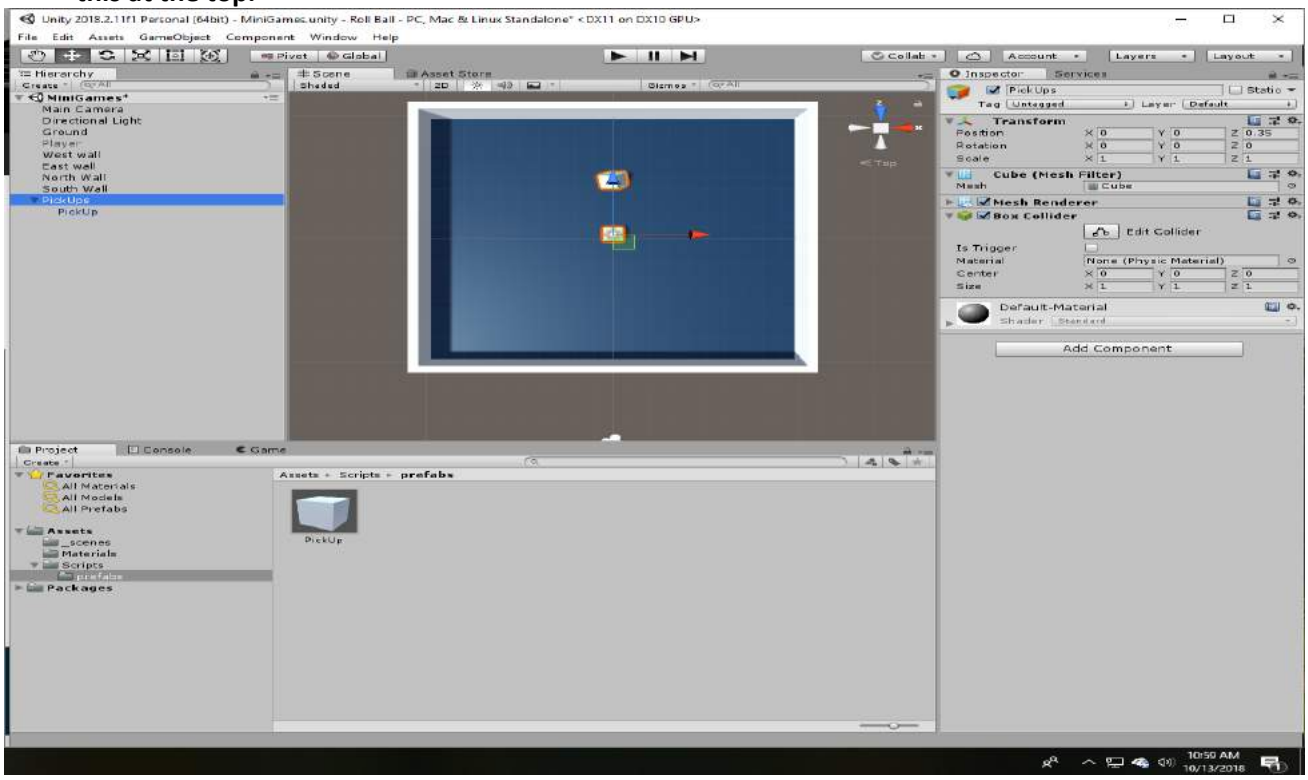
62. **Let move in to a top-down view by clicking on the gizmo in to upper-right of our scene view**



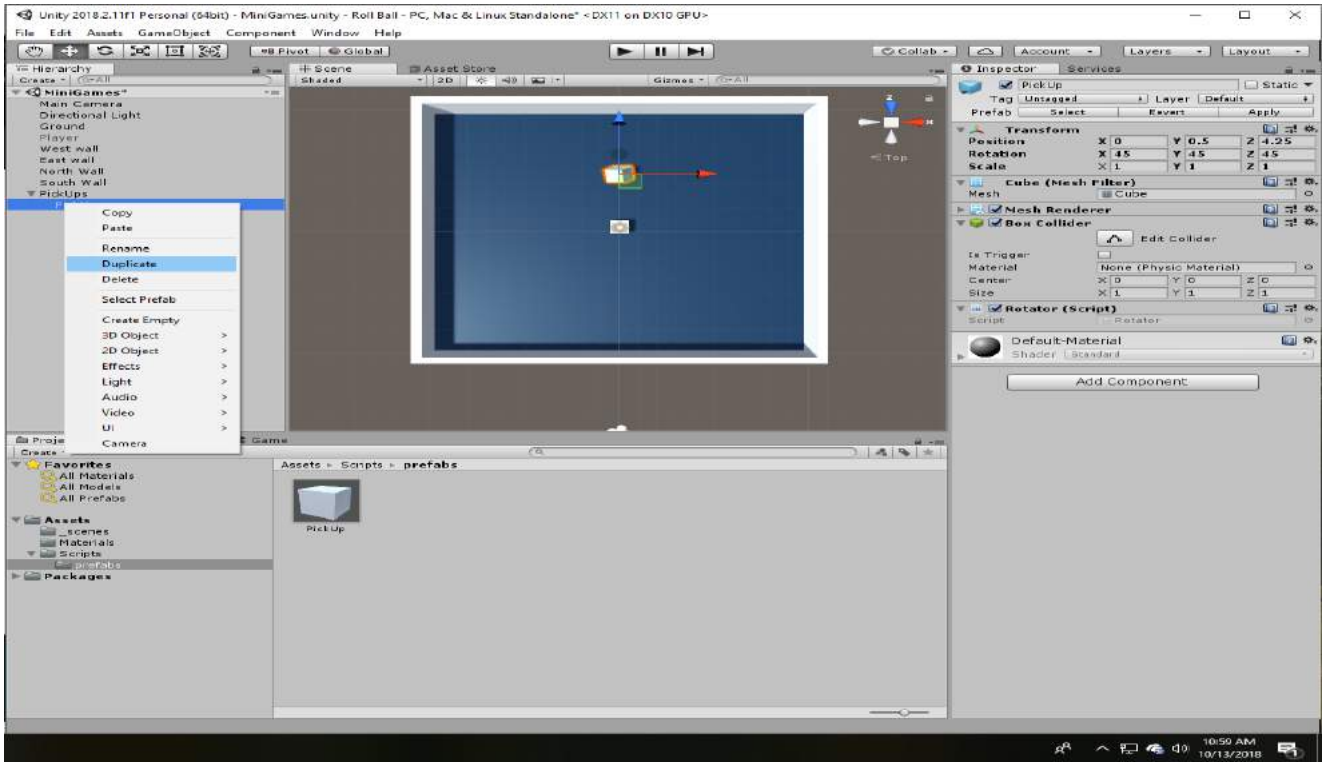
63. **Change the editor to Global mode.**



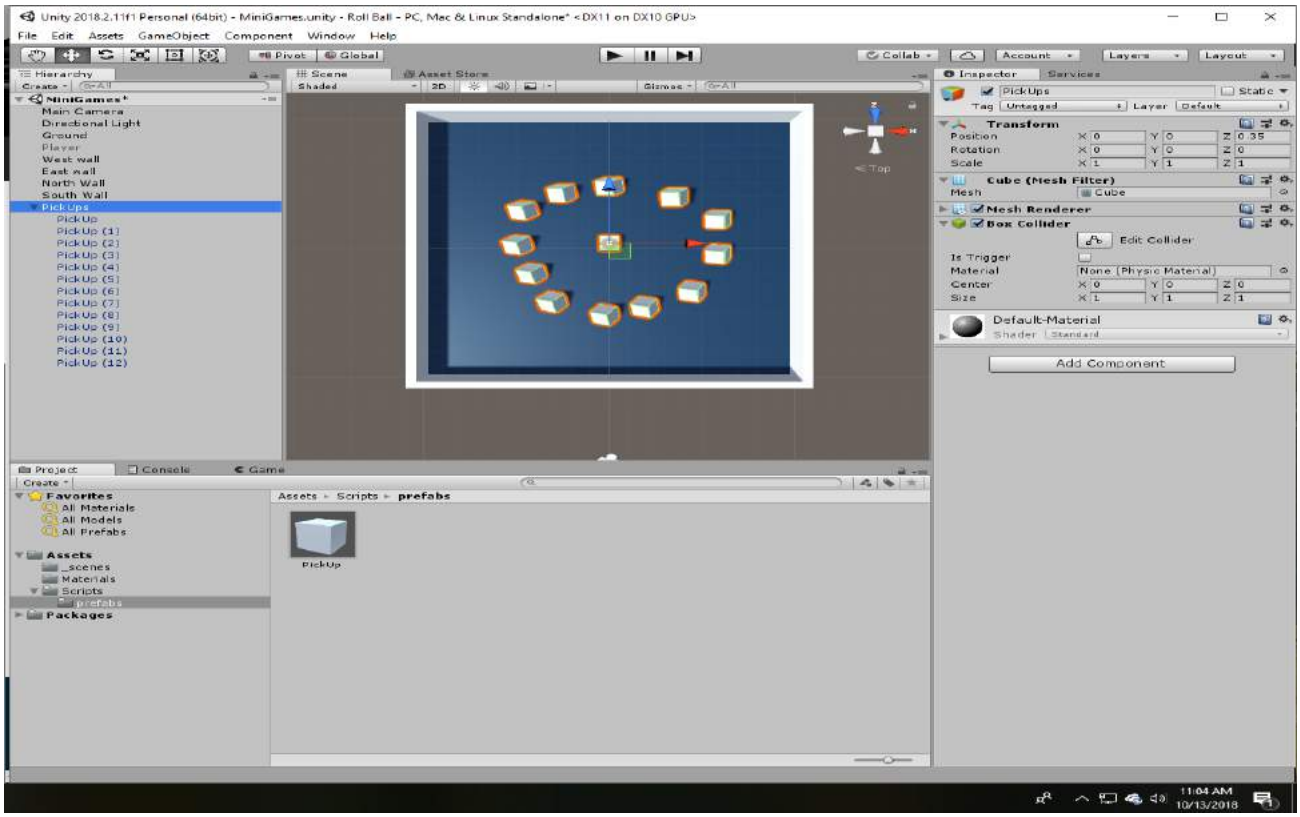
64. **Take the first Pickup object and place it in to the game area, some place convenient. Place this at the top.**



65. **With the game object selected, duplicate it.**

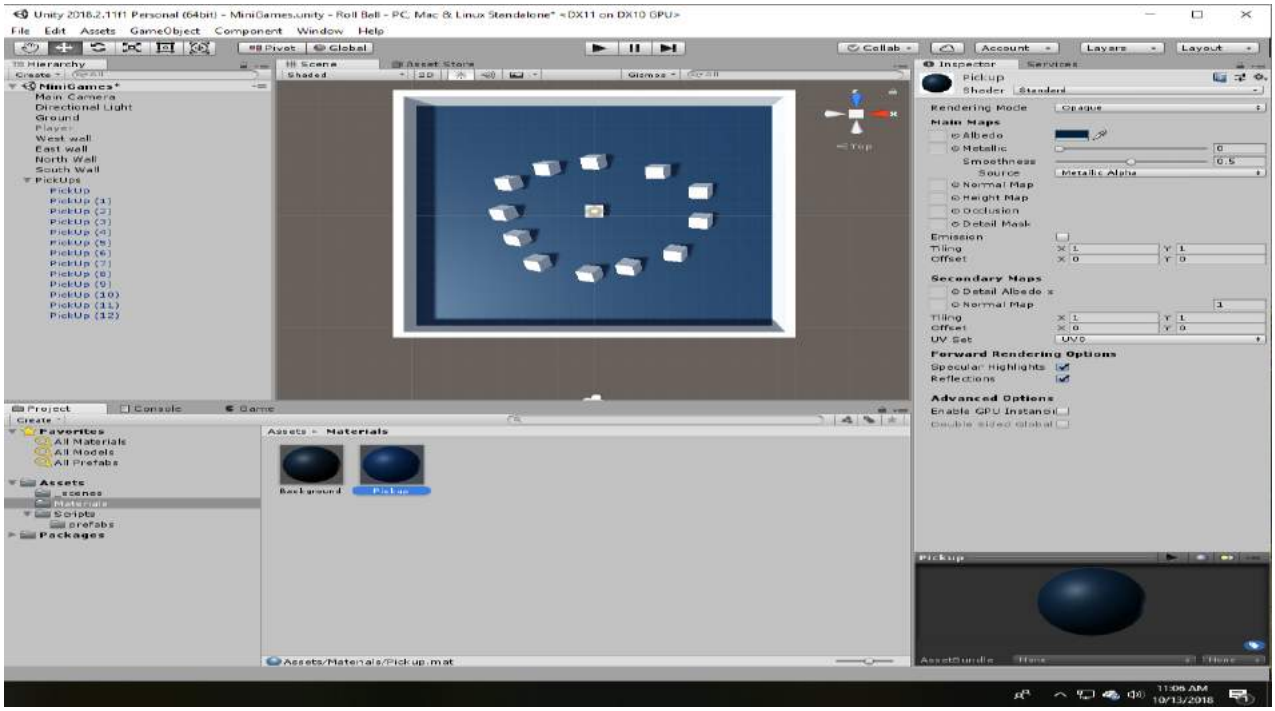


66. **Using the hot keys we will create a few more, placing them around the play area. About 12 in a circle.**

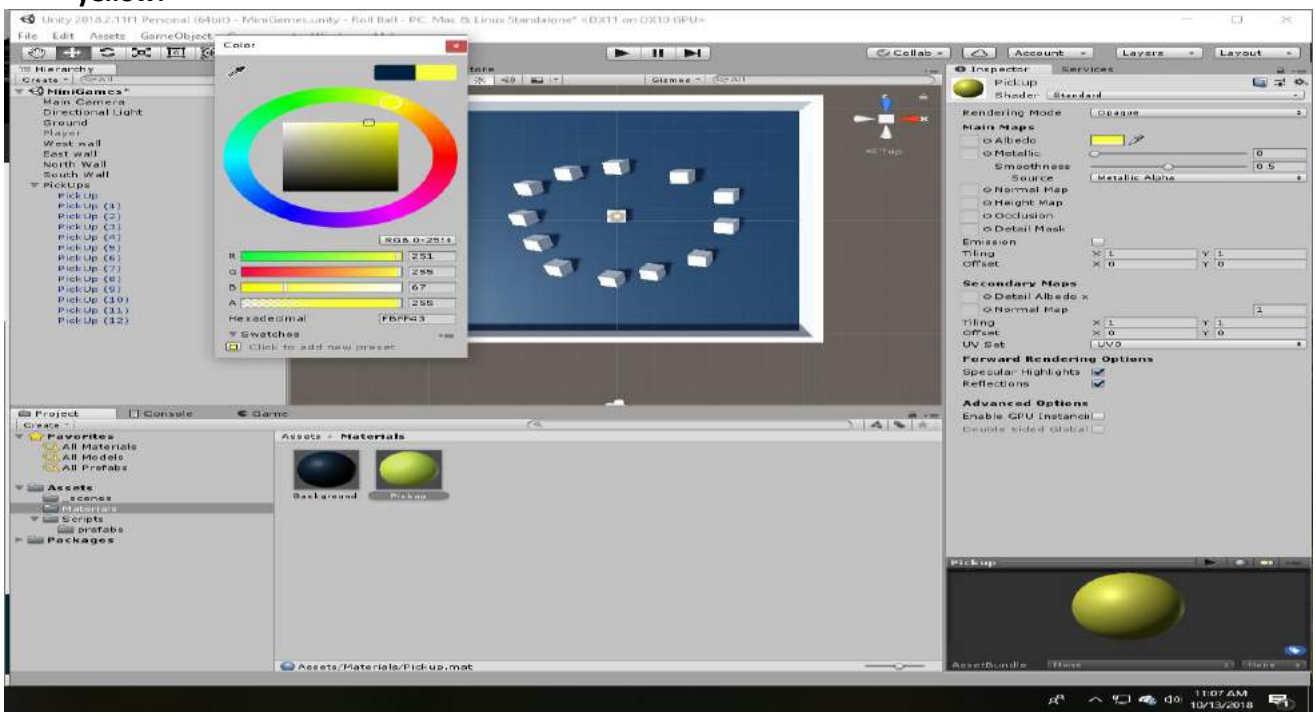


67. **Test and the pickup objects should rotate.**

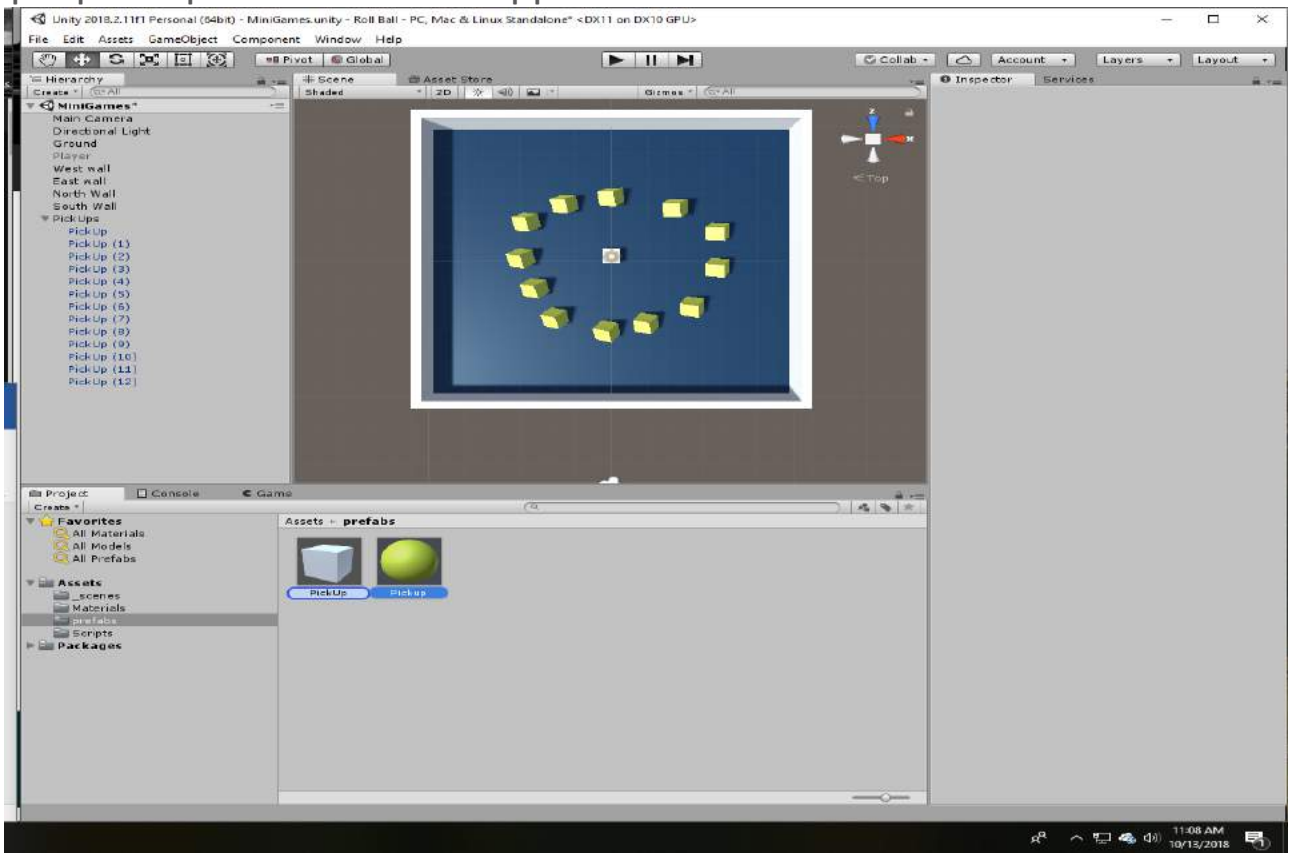
68. **Change their color. To do this we need another material. To make things easy we can simply select our existing material and duplicate it. Let's rename this new material Pickup.**



69. **With the material selected in the project view let's change the albedo colour property to yellow.**



70. **Change the color of the prefab by changing the prefab's material. Drag the prefab material 'pickup' to the prefab folder on the 'Pickup prefab'**

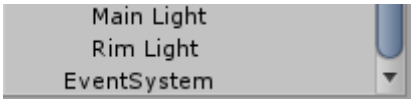


PRACTICAL NO:-08

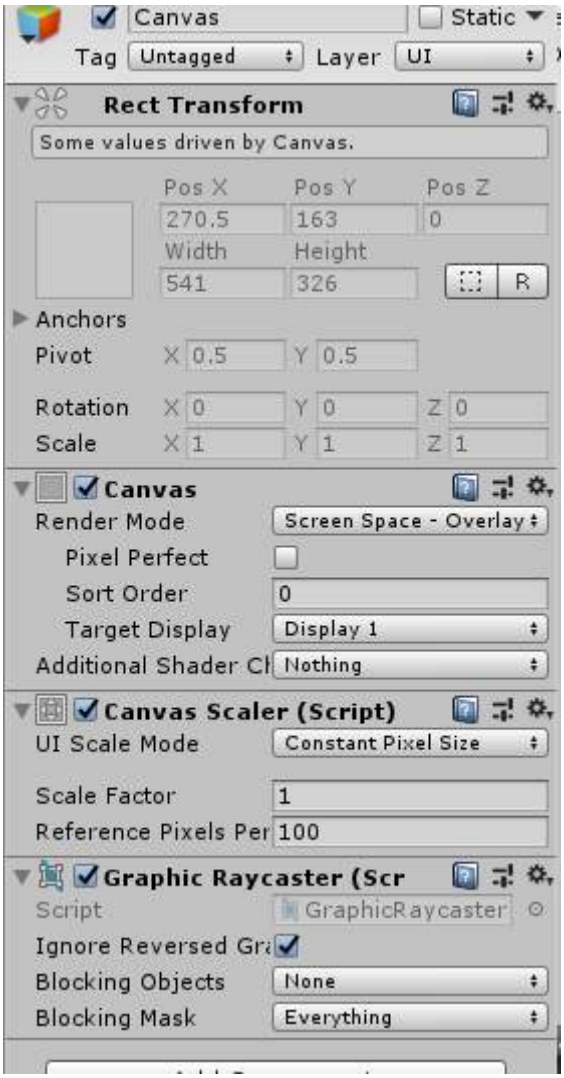
AIM:- Game development using unity SPACE SHOOTER.

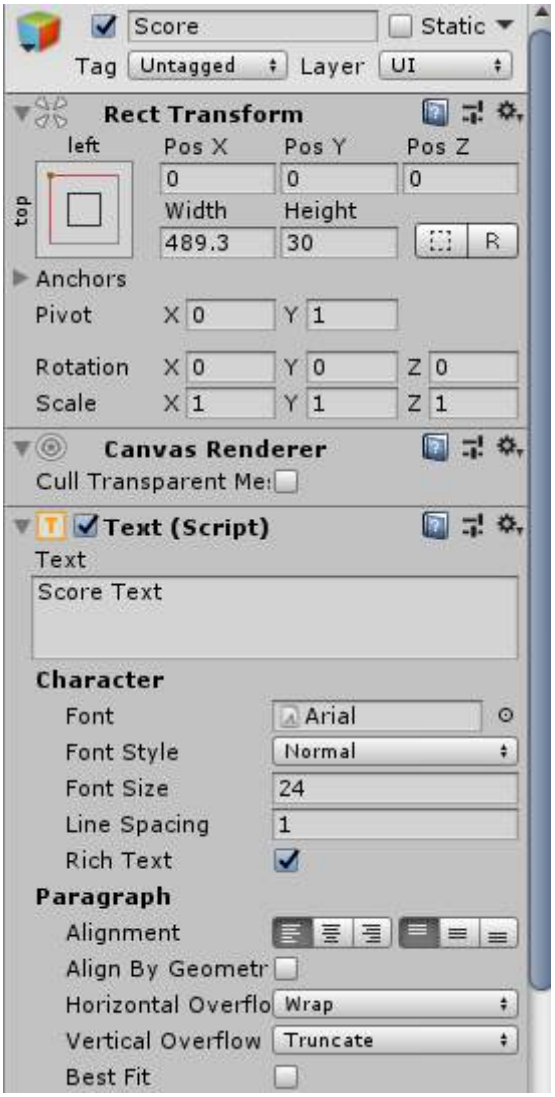
STEPS

1} GAMEOBJECT TO BE CREATED ARE AS FOLLOWS-



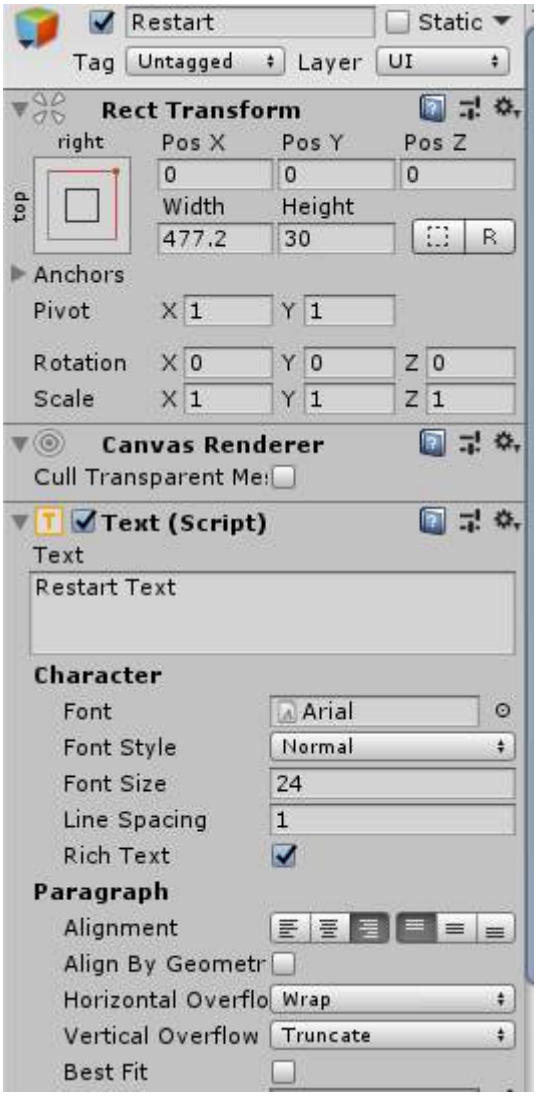
2} CREATE Canvas Object --

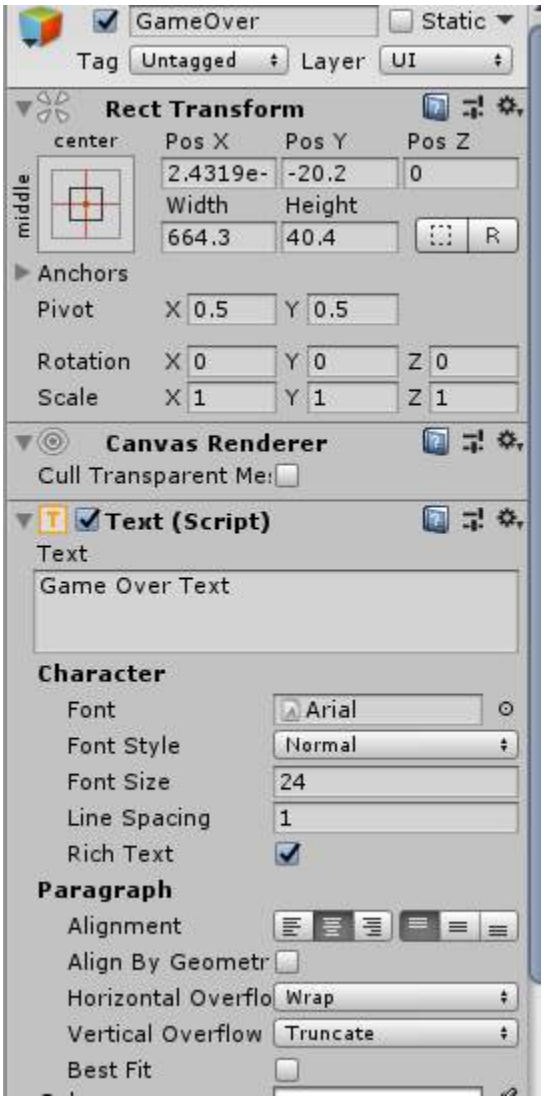




<- CREATE Score OBJECT

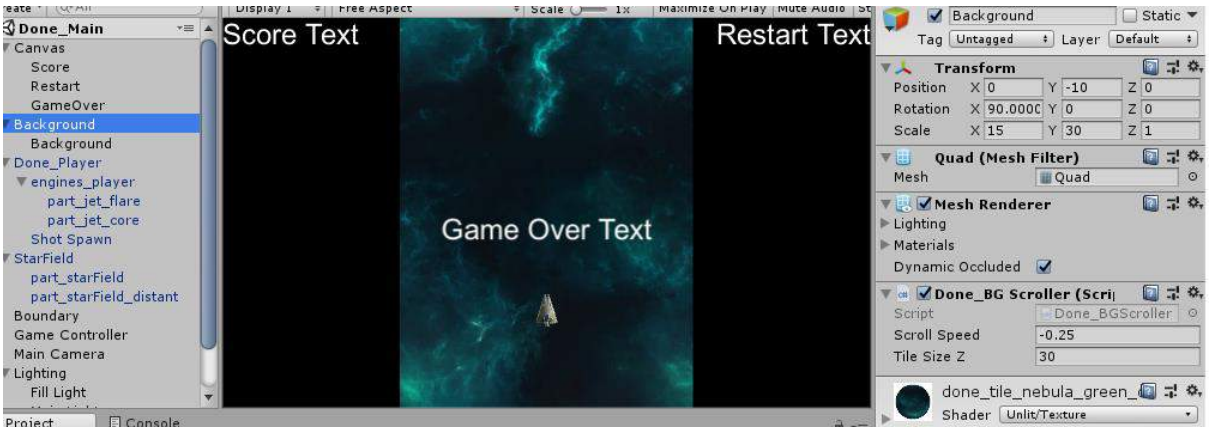
CREATE Restart OBJECT ->



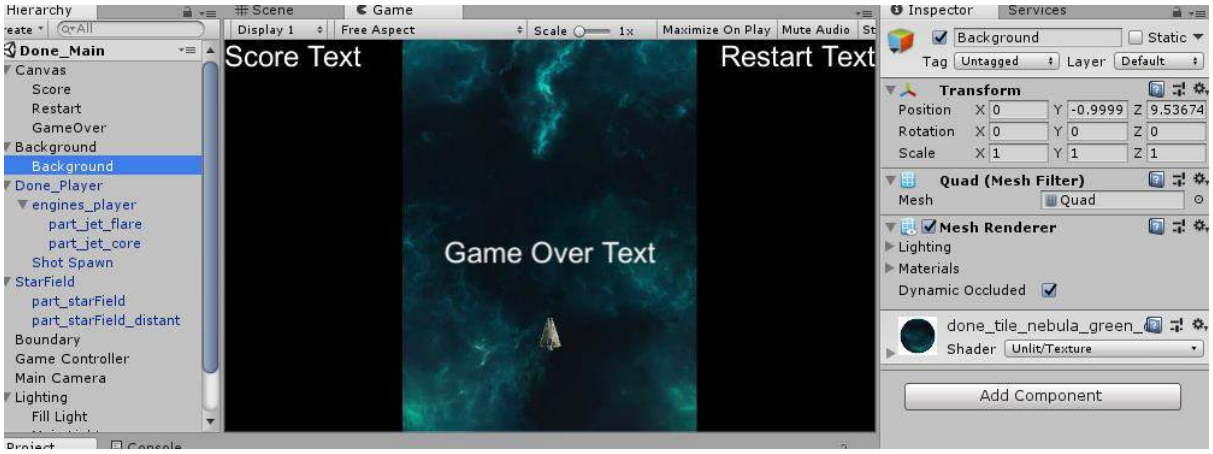


<- CREATE GameOver OBJECT

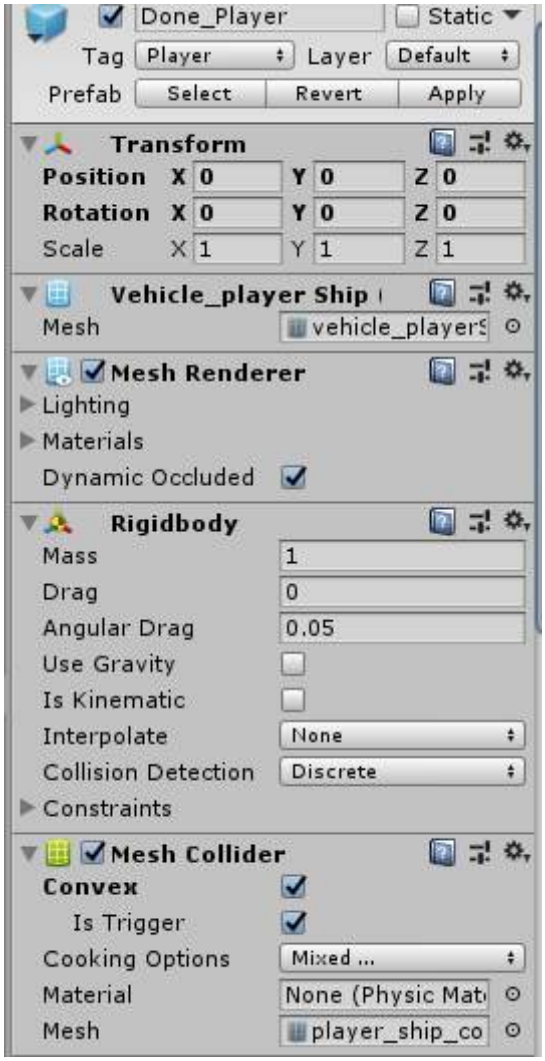
CREATE Background Object ->



CREATE INNER Background OBJECT ->

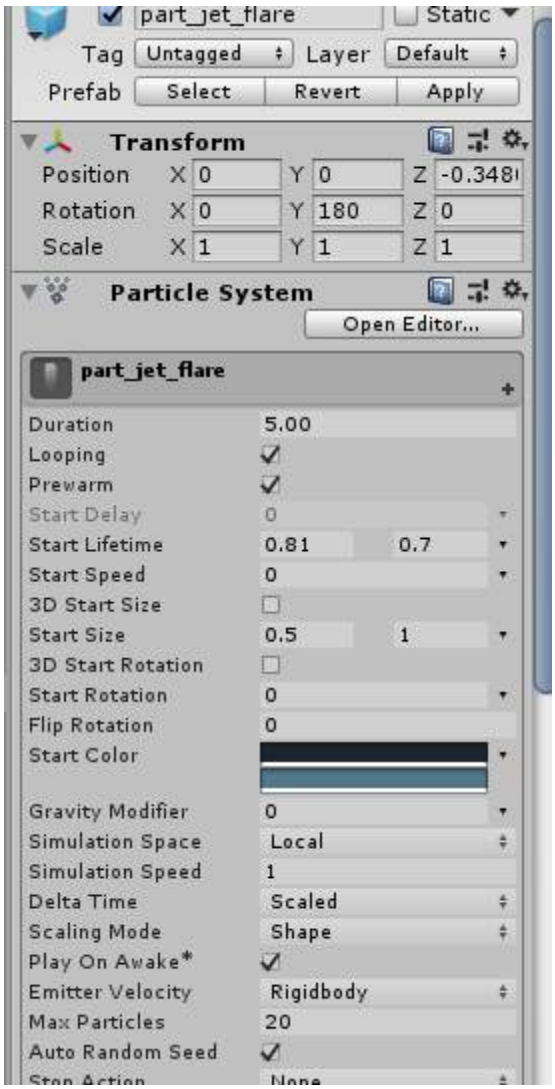


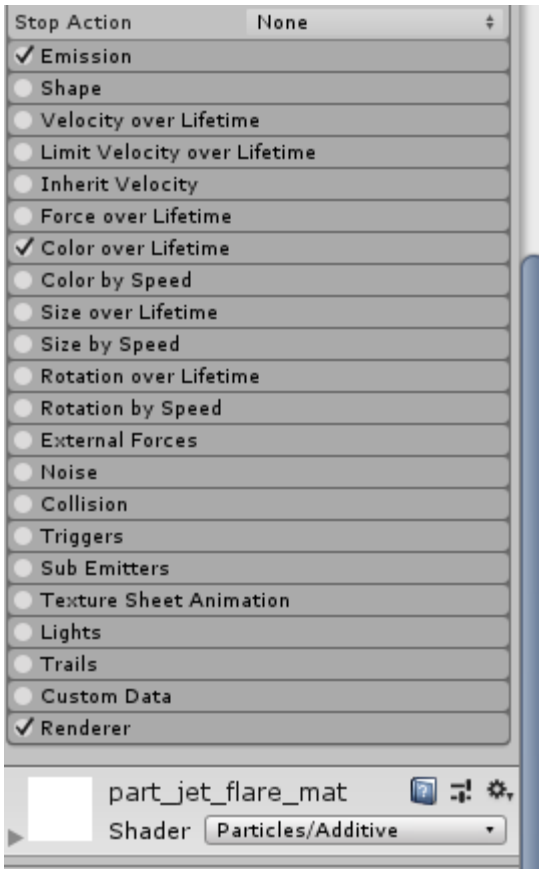
CREATE Done_Player OBJECT ->



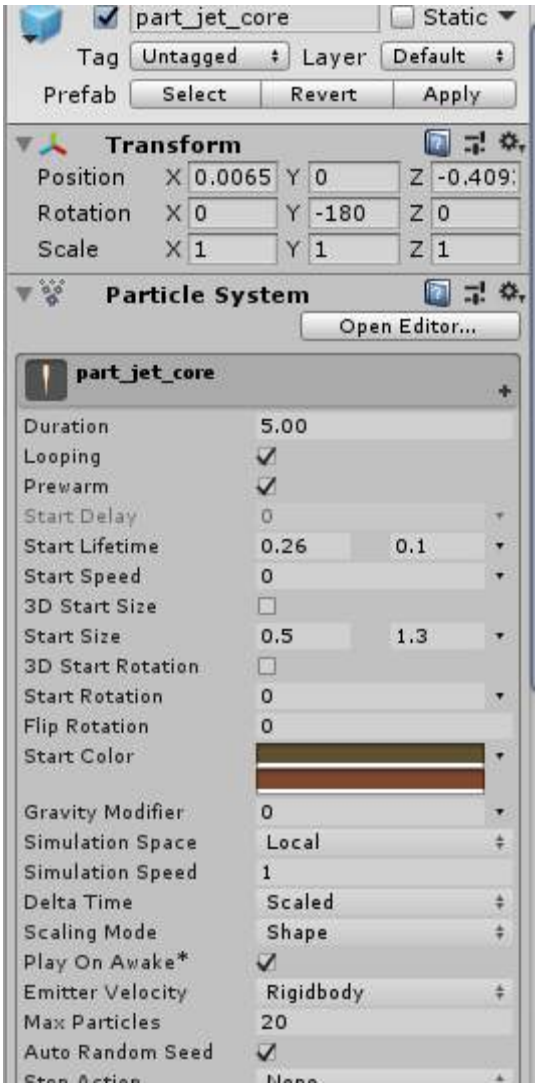
<-CREATE engines_player OBJECT

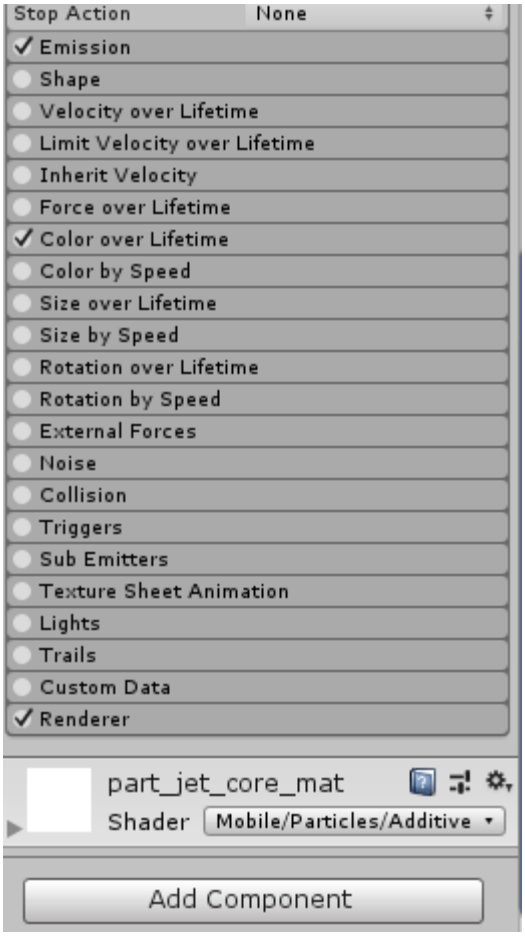
CREATE part_jet_flare OBJECT->



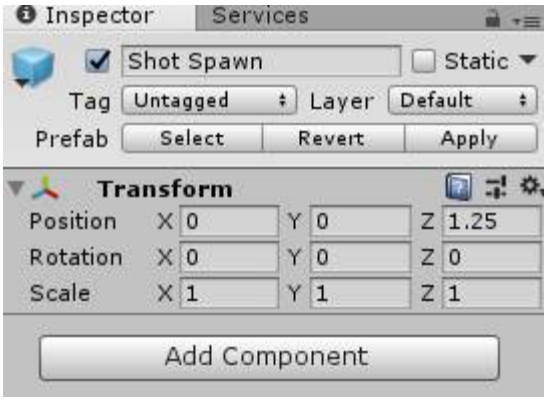


CREATE part_jet_core OBJECT—

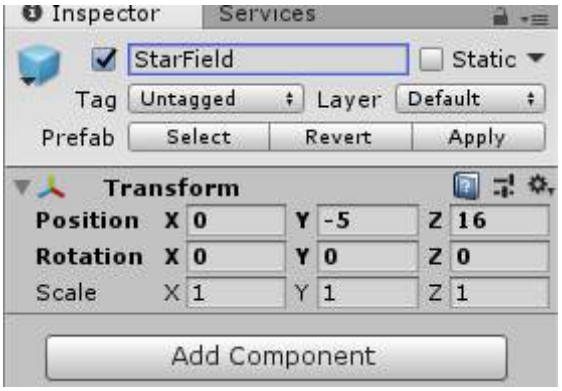




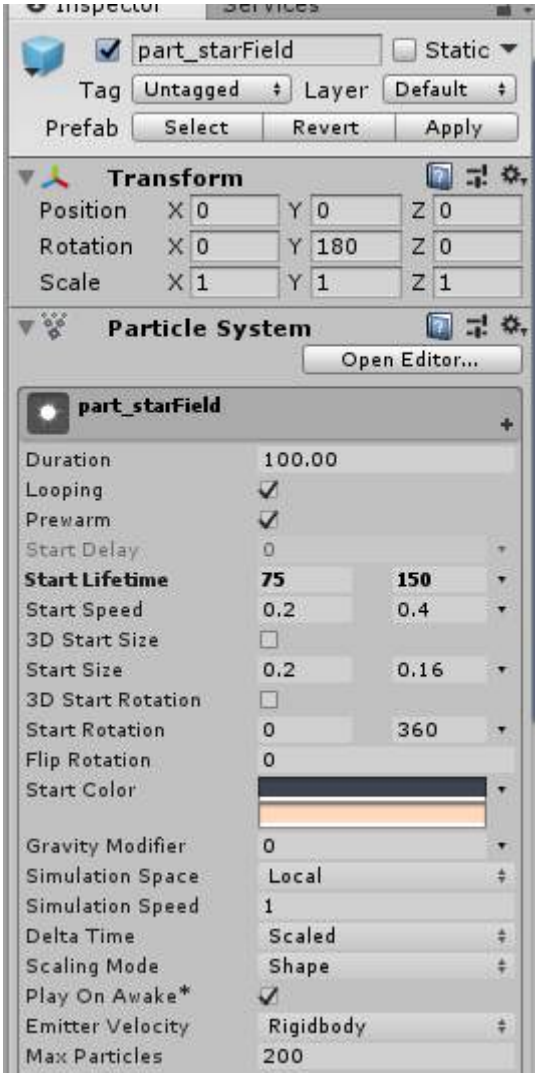
CREATE Shot Spawn OBJECT—



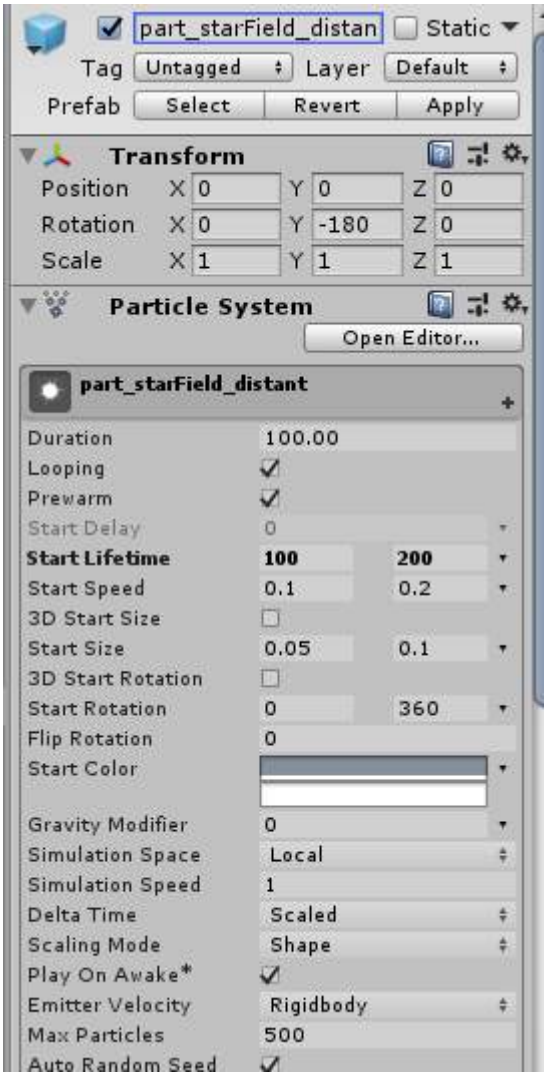
CREATE StarField OBJECT—

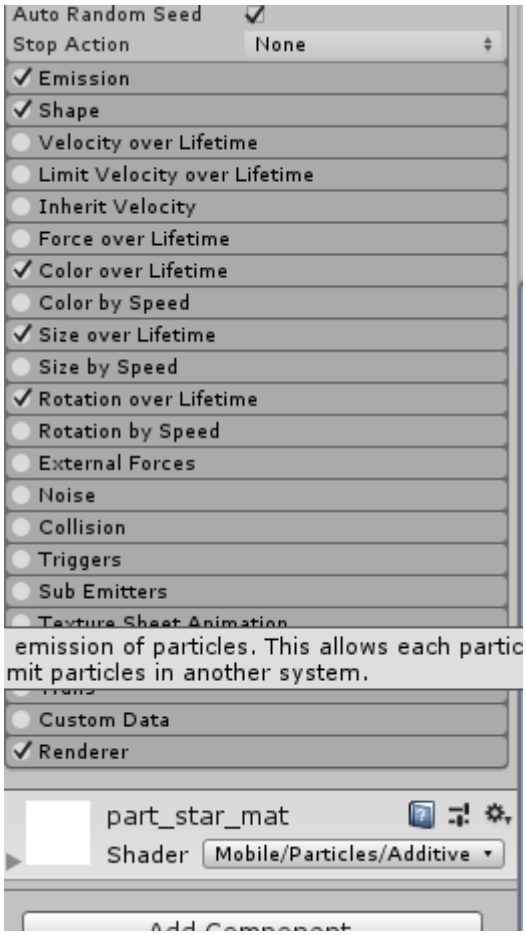


CREATE part_starField OBJCT—

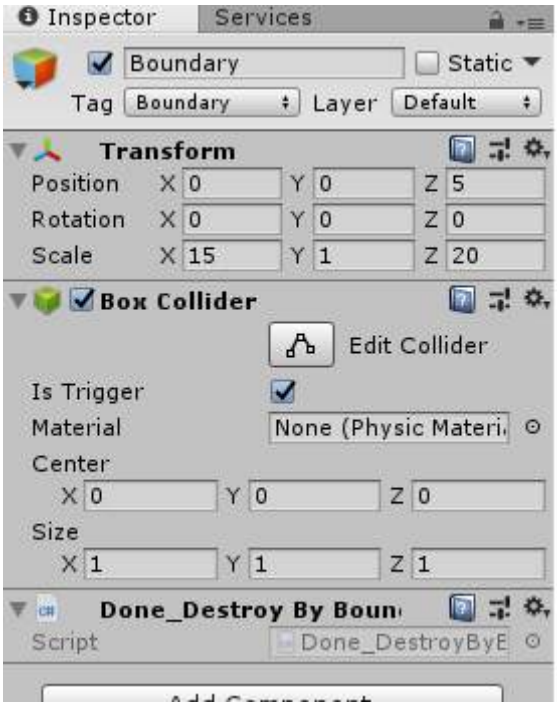


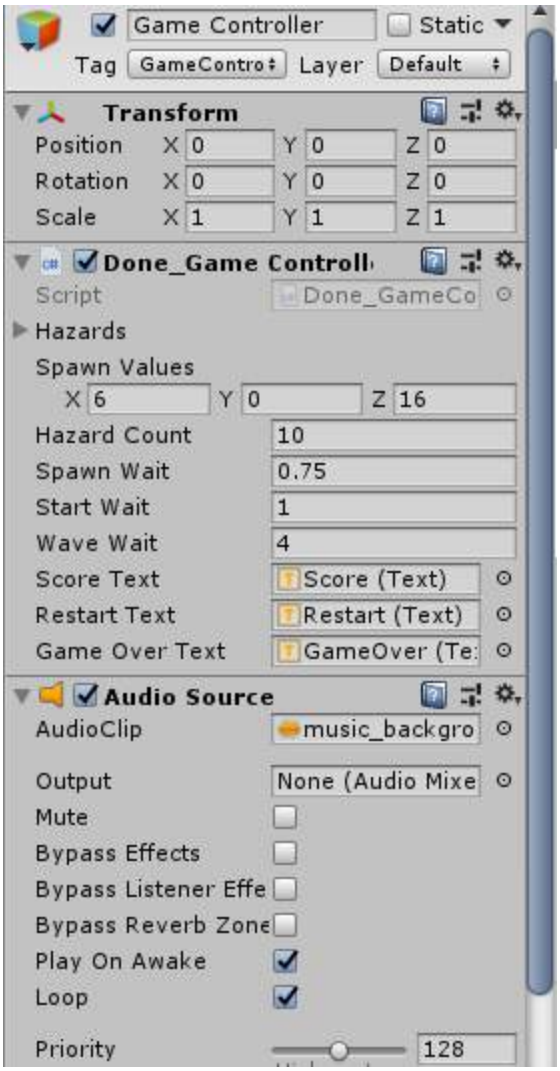
CREATE part_starField_distant OBJECT—



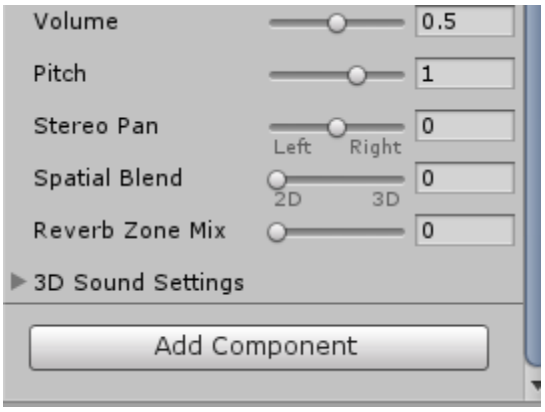


CREATE Boundary OBJECT—

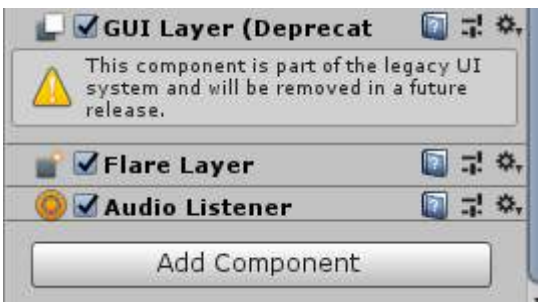


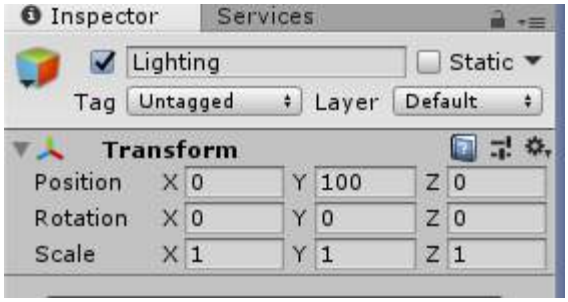


<- CREATE Game Controller OBJECT



CREATE Main Camera OBJECT—



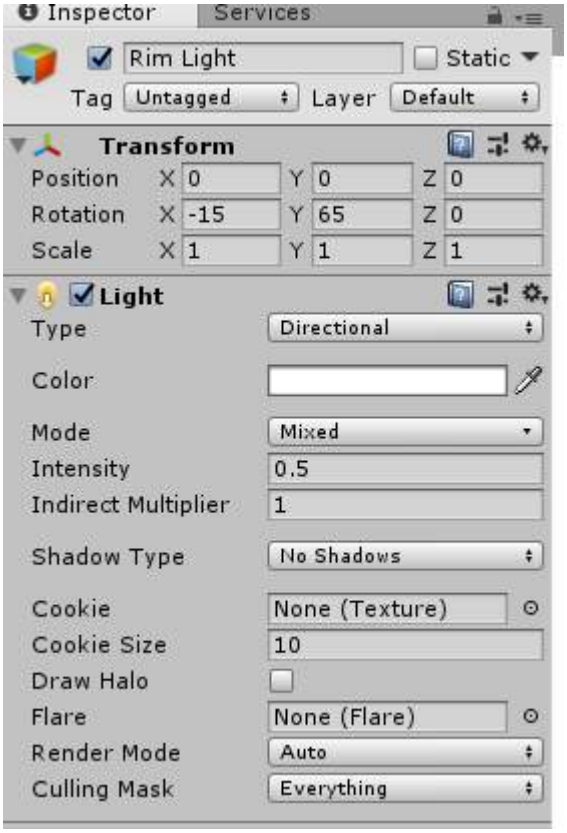


<- CREATE Lighting OBJECT

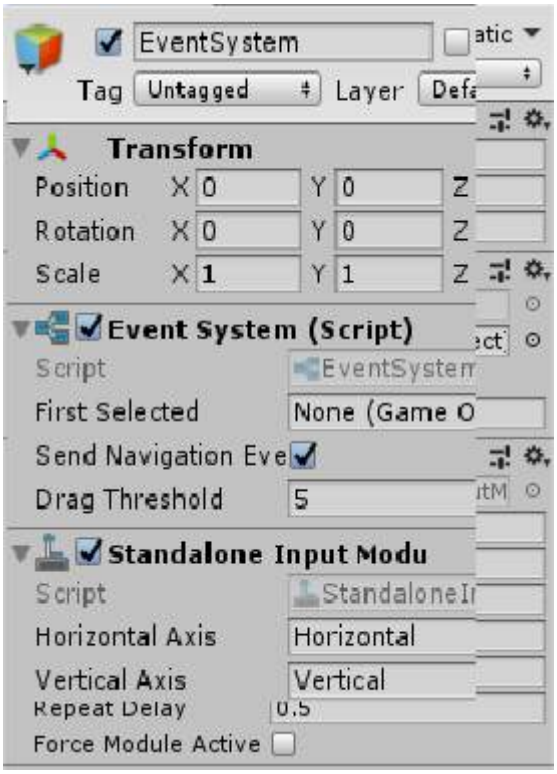
CREATE Fill Light OBJECT ->

<- CREATE Main Light OBJECT

CREATE Rim Light OBJECT—



CREATE EventSystem OBJECT—



Done_BGScroller.CS

```
using UnityEngine;
using System.Collections;
public class Done_BGScroller : MonoBehaviour  {
public float scrollSpeed;
public float tileSizeZ;
private Vector3 startPosition;
void Start ()      {
startPosition = transform.position;        }
void Update ()  {
float newPosition = Mathf.Repeat(Time.time * scrollSpeed, tileSizeZ);
transform.position = startPosition + Vector3.forward * newPosition;      }}
```

Done_DestroyByBoundary.CS

```
using UnityEngine;
using System.Collections;
public class Done_DestroyByBoundary : MonoBehaviour{
void OnTriggerExit (Collider other)        {
Destroy(other.gameObject);      }}
```

Done_DestroyByContact.CS

```
using UnityEngine;
using System.Collections;
public class Done_DestroyByContact : MonoBehaviour   {
public GameObject explosion;
public GameObject playerExplosion;
public int scoreValue;
private Done_GameController gameController;
void Start ()      {
GameObject gameControllerObject = GameObject.FindGameObjectWithTag ("GameController");
if (gameControllerObject != null)          {
gameController = gameControllerObject.GetComponent <Done_GameController>();      }
if (gameController == null)        {
Debug.Log ("Cannot find 'GameController' script");        }}
void OnTriggerEnter (Collider other)      {
if (other.tag == "Boundary" || other.tag == "Enemy")     {
return;  }
if (explosion != null)      {
Instantiate(explosion, transform.position, transform.rotation);   }
if (other.tag == "Player")          {
Instantiate(playerExplosion, other.transform.position, other.transform.rotation);
gameController.GameOver();     }
gameController.AddScore(scoreValue);
Destroy (other.gameObject);
Destroy (gameObject);  }}
```

Done_DestroyByTime.CS

```
using UnityEngine;
using System.Collections;
public class Done_DestroyByTime : MonoBehaviour       {
```

```
public float lifetime;
void Start ()       {
Destroy (gameObject, lifetime); }}
```

Done_EvasiveManeuver.CS

```
using UnityEngine;
using System.Collections;
public class Done_EvasiveManeuver : MonoBehaviour    {
public Done_Boundary boundary;
public float tilt;
public float dodge;
public float smoothing;
public Vector2 startWait;
public Vector2 maneuverTime;
public Vector2 maneuverWait;
private float currentSpeed;
private float targetManeuver;
void Start ()       {
currentSpeed = GetComponent<Rigidbody>().velocity.z;
StartCoroutine(Evade());            }
IEnumerator Evade ()    {
yield return new WaitForSeconds (Random.Range (startWait.x, startWait.y));
while (true)       {
targetManeuver = Random.Range (1, dodge) * -Mathf.Sign (transform.position.x);
yield return new WaitForSeconds (Random.Range (maneuverTime.x, maneuverTime.y));
targetManeuver = 0;
yield return new WaitForSeconds (Random.Range (maneuverWait.x, maneuverWait.y)); }}
void FixedUpdate ()       {
float newManeuver = Mathf.MoveTowards (GetComponent<Rigidbody>().velocity.x,
targetManeuver, smoothing * Time.deltaTime);
GetComponent<Rigidbody>().velocity = new Vector3 (newManeuver, 0.0f, currentSpeed);
GetComponent<Rigidbody>().position = new Vector3     (
Mathf.Clamp(GetComponent<Rigidbody>().position.x, boundary.xMin, boundary.xMax),
0.0f,
Mathf.Clamp(GetComponent<Rigidbody>().position.z, boundary.zMin, boundary.zMax));
GetComponent<Rigidbody>().rotation = Quaternion.Euler (0, 0,
GetComponent<Rigidbody>().velocity.x * -tilt);   }}
```

Done_GameController.CS

```
using UnityEngine;
using UnityEngine.SceneManagement;
using System.Collections;
using UnityEngine.UI;
public class Done_GameController : MonoBehaviour      {
public GameObject[] hazards;
public Vector3 spawnValues;
public int hazardCount;
public float spawnWait;
public float startWait;
public float waveWait;
public Text scoreText;
```

```
public Text restartText;
public Text gameOverText;
private bool gameOver;
private bool restart;
private int score;
void Start()        {
gameOver = false;
restart = false;
restartText.text = "";
gameOverText.text = "";
score = 0;
UpdateScore();
StartCoroutine(SpawnWaves()); }
void Update()    {
if (restart)         {
if (Input.GetKeyDown(KeyCode.R))         {
SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);}}}
IEnumerator SpawnWaves()       {
yield return new WaitForSeconds(startWait);
while (true)       {
for (int i = 0; i < hazardCount; i++)          {
GameObject hazard = hazards[Random.Range(0, hazards.Length)];
Vector3 spawnPosition = new Vector3(Random.Range(-spawnValues.x, spawnValues.x),
spawnValues.y, spawnValues.z);
Quaternion spawnRotation = Quaternion.identity;
Instantiate(hazard, spawnPosition, spawnRotation);
yield return new WaitForSeconds(spawnWait);   }
yield return new WaitForSeconds(waveWait);
if (gameOver)    {
restartText.text = "Press 'R' for Restart";
restart = true;
break;   }}}
public void AddScore(int newScoreValue)          {
score += newScoreValue;
UpdateScore();  }
void UpdateScore()        {
scoreText.text = "Score: " + score;          }
public void GameOver() {
gameOverText.text = "Game Over!";
gameOver = true;          }}
```

Done_Mover.CS

```
using UnityEngine;
using System.Collections;
public class Done_Mover : MonoBehaviour        {
public float speed;
void Start ()       {
GetComponent<Rigidbody>().velocity = transform.forward * speed;        }}
```

Done_Boundary.CS

```
using UnityEngine;
```

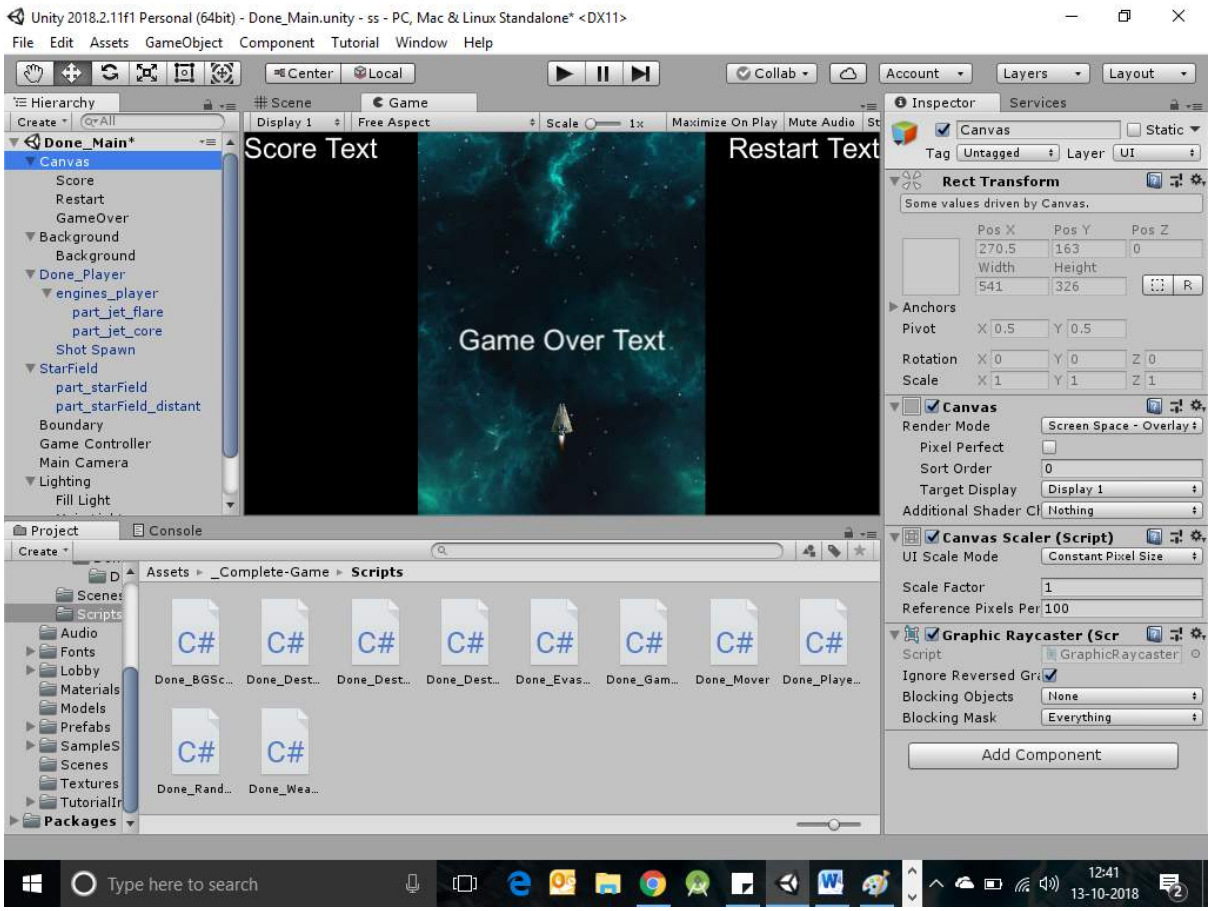
```
using System.Collections;
[System.Serializable]
public class Done_Boundary {
public float xMin, xMax, zMin, zMax; }
public class Done_PlayerController : MonoBehaviour      {
public float speed;
public float tilt;
public Done_Boundary boundary;
public GameObject shot;
public Transform shotSpawn;
public float fireRate;
private float nextFire;
void Update ()   {
if (Input.GetButton("Fire1") && Time.time > nextFire)     {
nextFire = Time.time + fireRate;
Instantiate(shot, shotSpawn.position, shotSpawn.rotation);
GetComponent<AudioSource>().Play (); }}
void FixedUpdate ()       {
float moveHorizontal = Input.GetAxis ("Horizontal");
float moveVertical = Input.GetAxis ("Vertical");
Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical);
GetComponent<Rigidbody>().velocity = movement * speed;
GetComponent<Rigidbody>().position = new Vector3 (
Mathf.Clamp (GetComponent<Rigidbody>().position.x, boundary.xMin, boundary.xMax),
0.0f,
Mathf.Clamp (GetComponent<Rigidbody>().position.z, boundary.zMin, boundary.zMax));
GetComponent<Rigidbody>().rotation = Quaternion.Euler (0.0f, 0.0f,
GetComponent<Rigidbody>().velocity.x * -tilt);   }}
```

Done_RandomRotator.CS

```
using UnityEngine;
using System.Collections;
public class Done_RandomRotator : MonoBehaviour      {
public float tumble;
void Start ()      {
GetComponent<Rigidbody>().angularVelocity = Random.insideUnitSphere * tumble;       }}
```

Done_WeaponController.CS

```
using UnityEngine;
using System.Collections;
public class Done_WeaponController : MonoBehaviour  {
public GameObject shot;
public Transform shotSpawn;
public float fireRate;
public float delay;
void Start ()      {
InvokeRepeating ("Fire", delay, fireRate);           }
void Fire ()       {
Instantiate(shot, shotSpawn.position, shotSpawn.rotation);
GetComponent<AudioSource>().Play();  }}
```


Unity 2018.2.11f1 Personal (64bit) - Done_Main.unity - ss - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Tutorial Window Help
Hierarchy
Done_Main*
Canvas
Score
Restart
GameOver
Background
Background
Done_Player
engines_player
part_jet_flare
part_jet_core
Shot Spawn
StarField
part_starField
part_starField_distant
Boundary
Game Controller
Main Camera
Lighting
Fill Light
Scene
Game
Score Text
Restart Text
Game Over Text
Inspector
Services
Canvas
Rect Transform
Anchors
Pivot
Rotation
Scale
Canvas
Render Mode
Screen Space - Overlay
Pixel Perfect
Sort Order
Target Display
Display 1
Canvas Scaler (Script)
UI Scale Mode
Constant Pixel Size
Scale Factor
Reference Pixels Per
Graphic Raycaster (Scr
Ignore Reversed Gri
Blocking Objects
None
Blocking Mask
Everything
Add Component
Project
Console
Assets ▸ _Complete-Game ▸ Scripts
Audio
Fonts
Lobby
Materials
Models
Prefabs
Scenes
Textures
Packages
Done_BGSc...
Done_Dest...
Done_Dest...
Done_Dest...
Done_Evas...
Done_Gam...
Done_Mover
Done_Playe...
Done_Rand...
Done_Wea...
Type here to search
12:41
13-10-2018

OUTPUT—