Unity

Roll-a-ball Tutorial

Contents

Introduction

Introduction to Roll-a-Ball

In this tutorial, we are going to make a very simple but playable game to use many of the basic concepts. We will see how to create new game objects, add components to these game objects, set the values on their properties and position these game objects in the scene to create a game.

In our game, the player will control a ball rolling around the game board. We will move the ball using physics and forces. We will get the input from the player through the keyboard and we will use those inputs to apply forces to the ball. We will see how to detect contact between the player's ball and the pick-up game objects. And then, we will display the current count and ends the game when all of the game objects have been picked up. We will simply use the primitive shapes like cubes, spheres and planes provided by Unity.

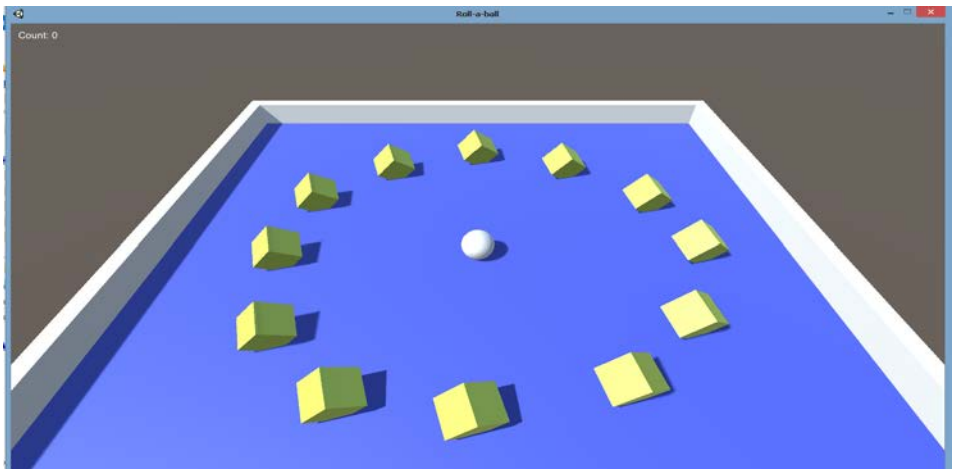
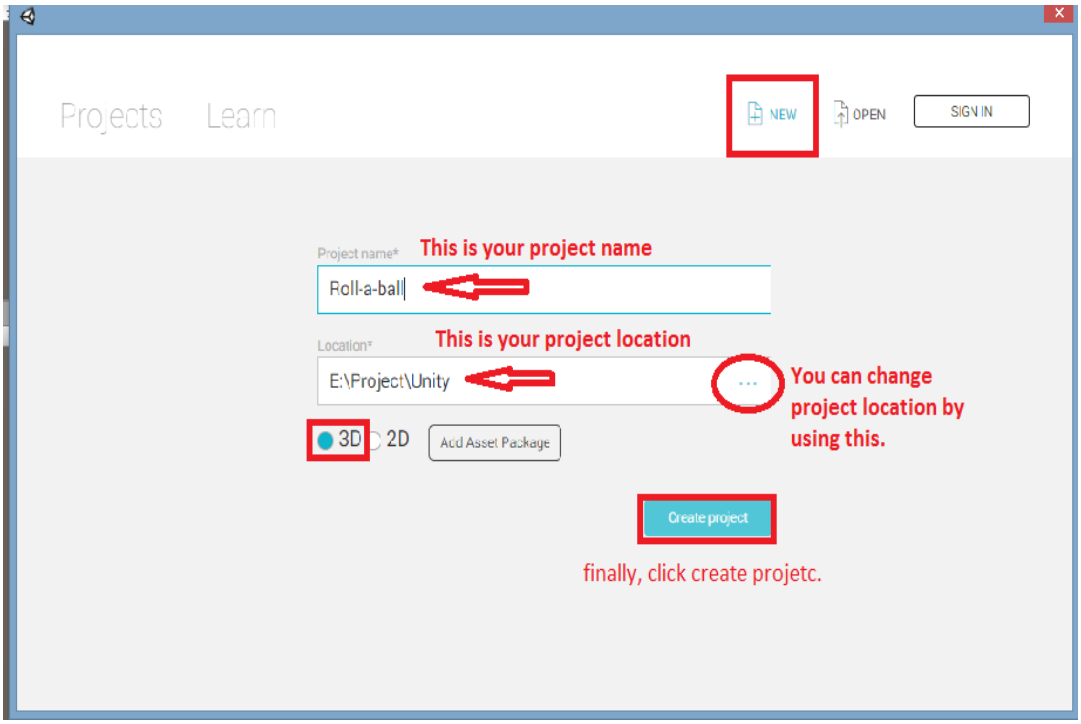


Figure: Sample of Roll-a-ball Game

1. Environment and Player

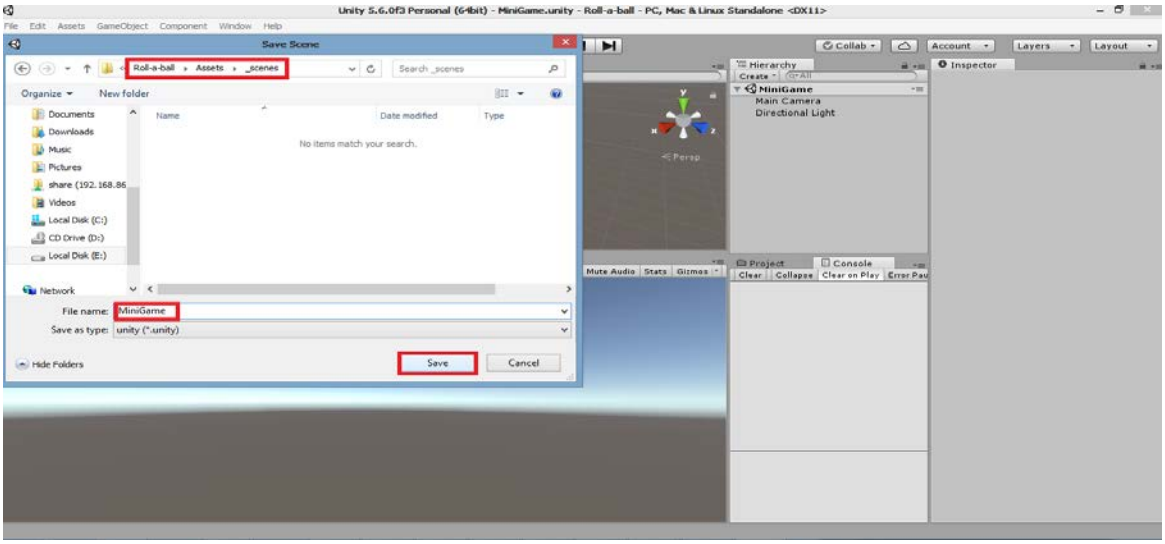
1.1. Setting up the Game

- Open the Unity Icon with double click.
- Click NEW, define project name, project location choose 3D and then click Create project.

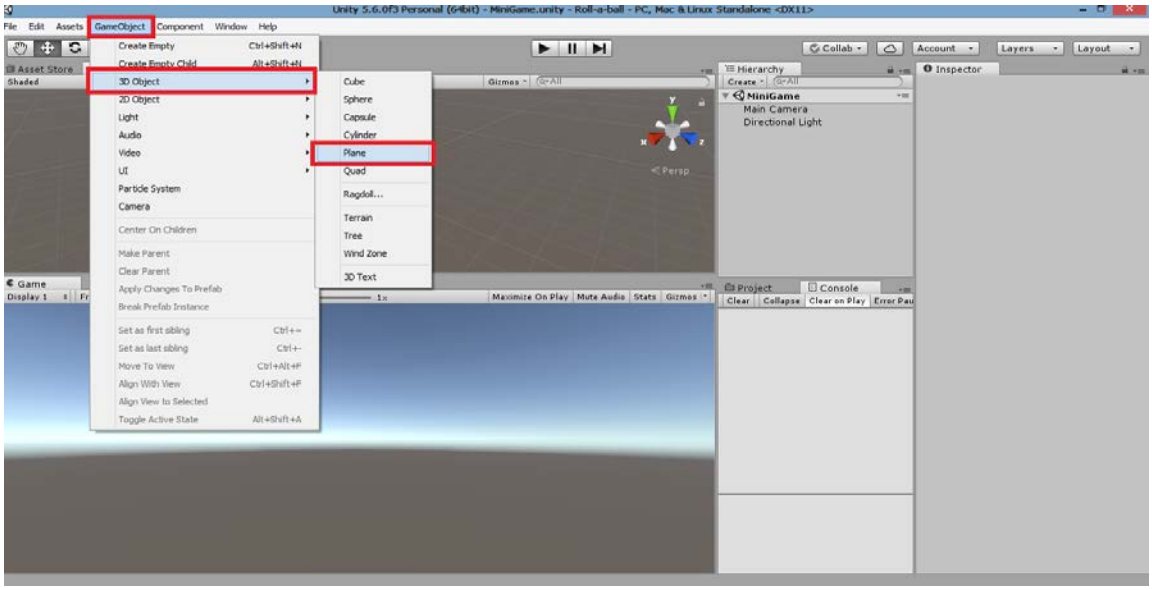


Before creating anything in the new scene, we need to save our scene.

- Save our scene by going to File - Save Scene.
- Save this scene in the Assets directory, and create a new folder called _scenes.
- Now save our scene as MiniGame in our _scenes folder.

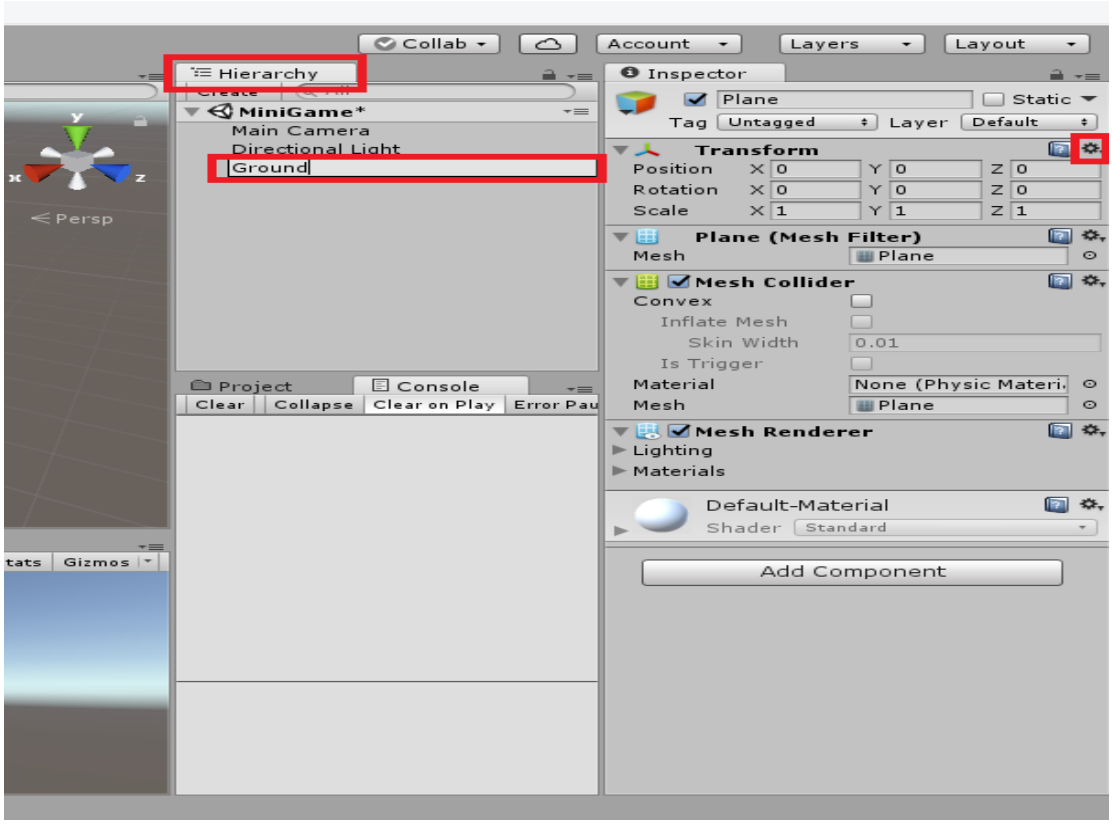


- Create a plane from GameObject - 3D Object – Plane.

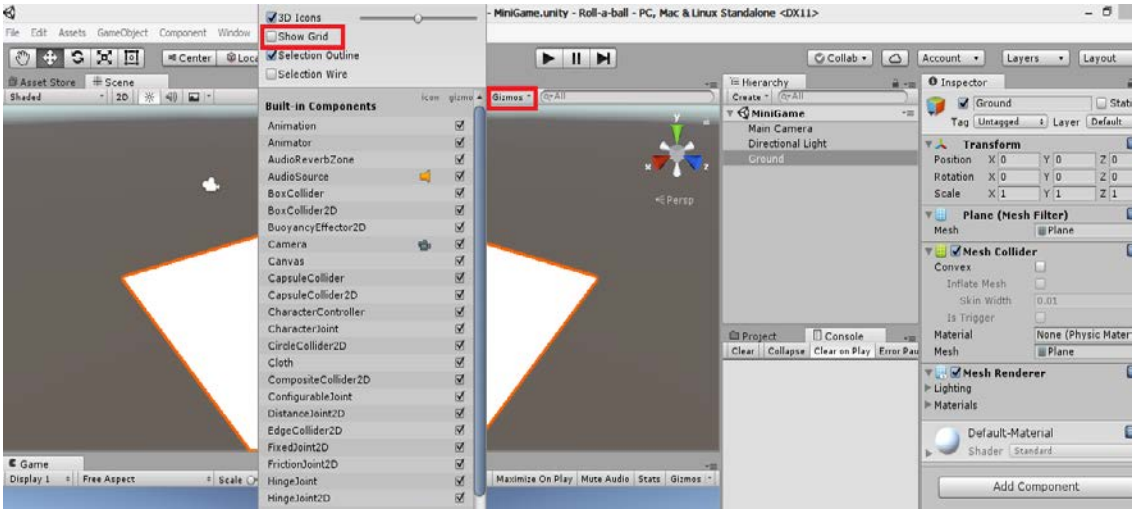


- Select the Plane and then rename the Plane as a Ground from Hierarchy View by using the selected object slowly twice or right click on selected object and rename the object.
- Reset the transform component using the context sensitive gear menu in the upper right.

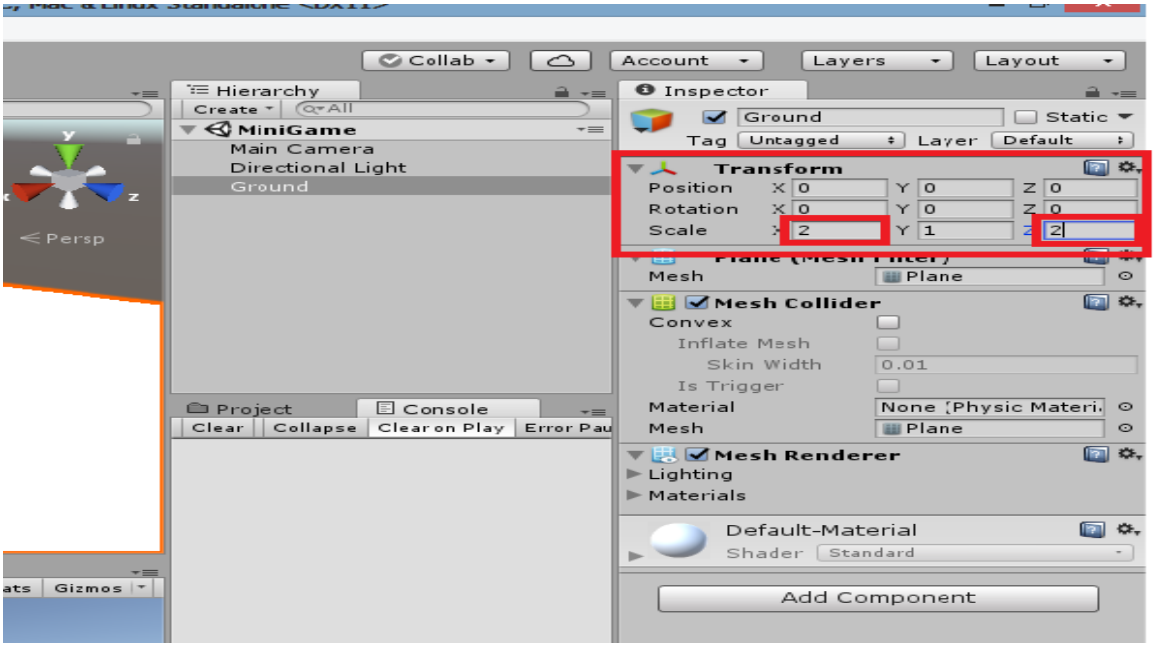
This will place the game object at the location of (0, 0, 0) in our scene.



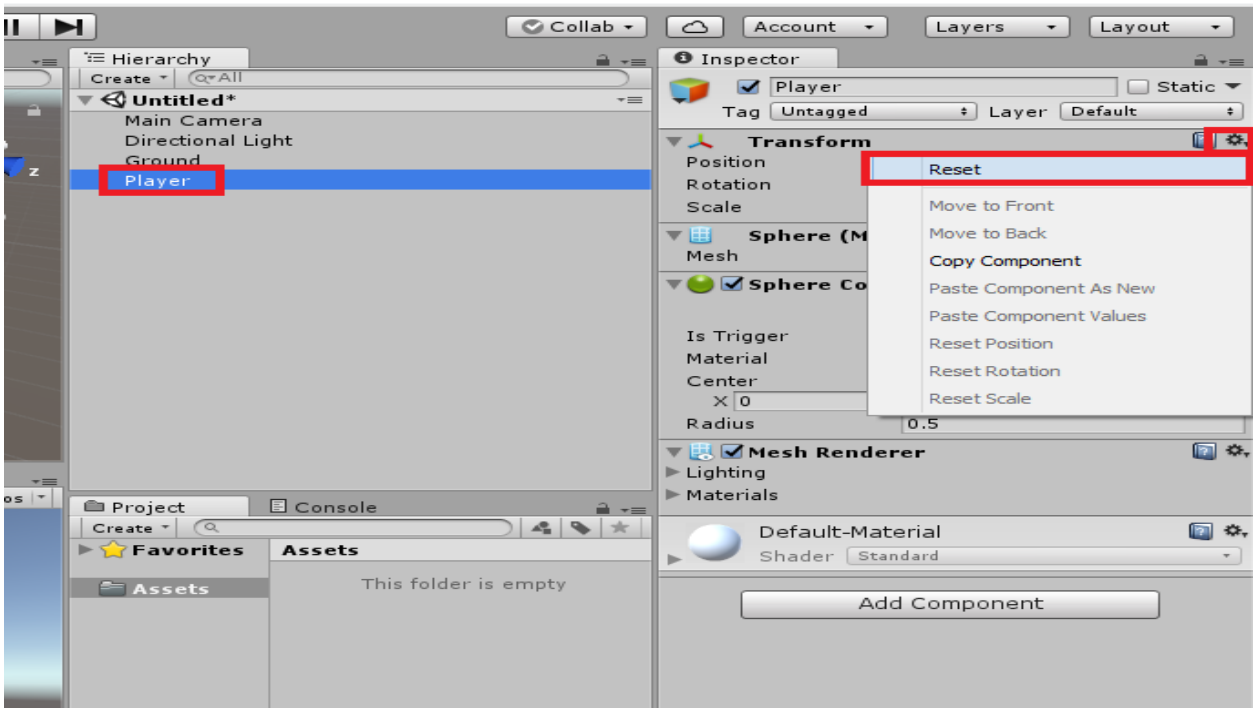
- Remove grid line from Gizmos menu in the Scene View and deselect Show Grid.



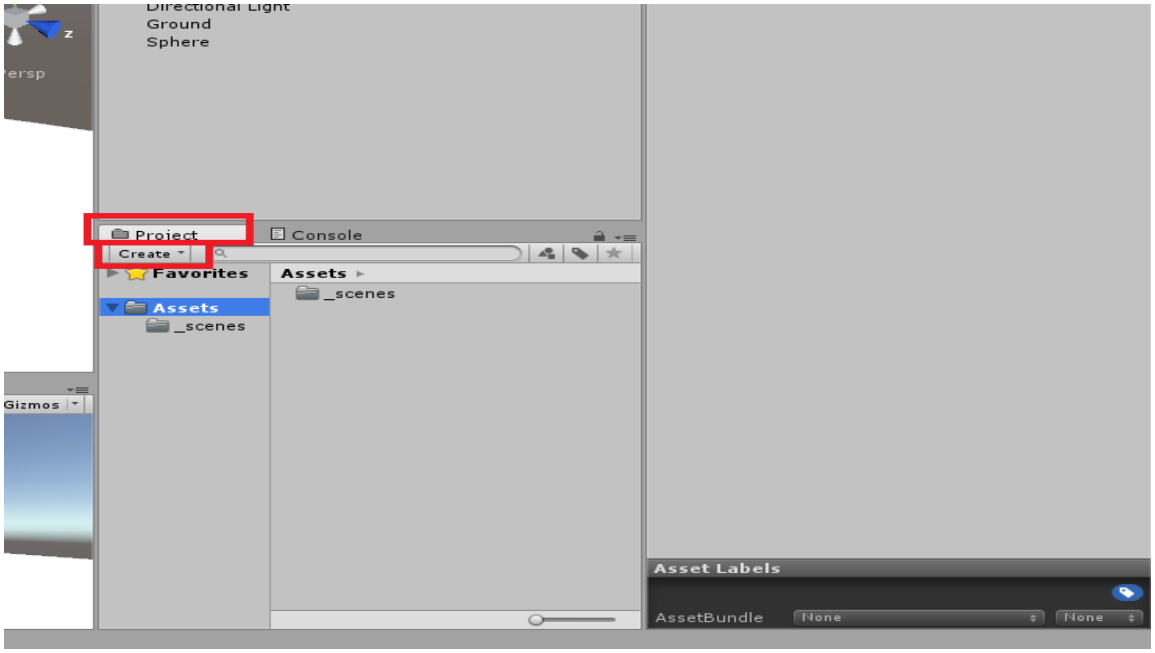
- Change the scale of the ground play to (2, 1, 2).

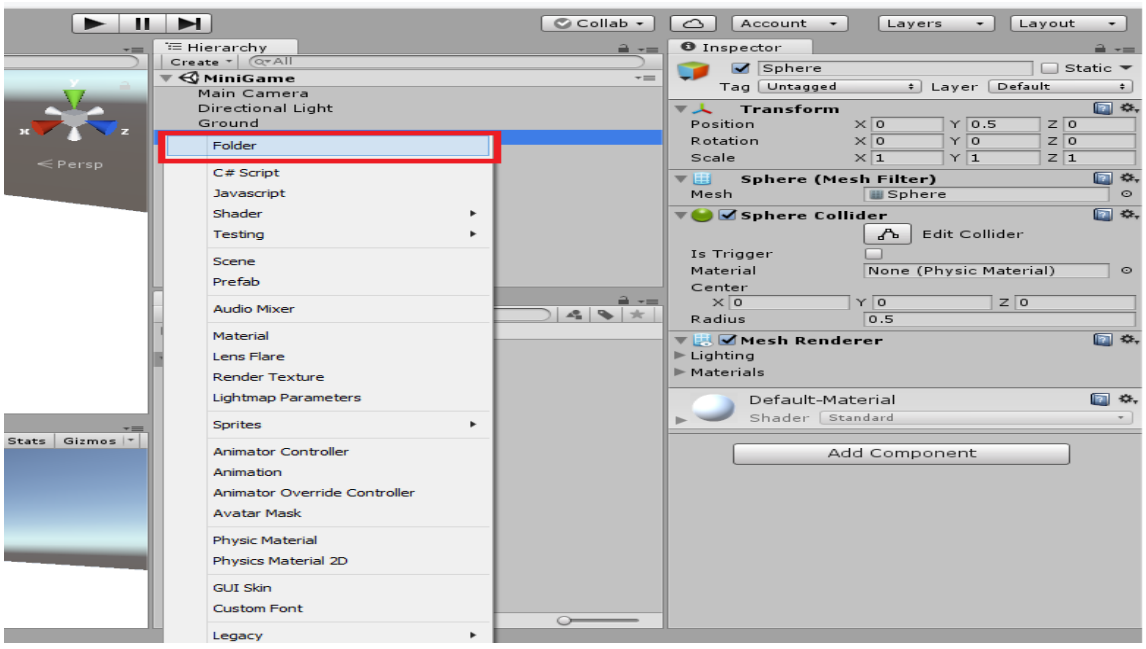


- Create a sphere from Hierarchy - 3D object – Sphere.
- Rename the sphere to Player.
- Reset the transform component using the context sensitive gear menu in the upper right.

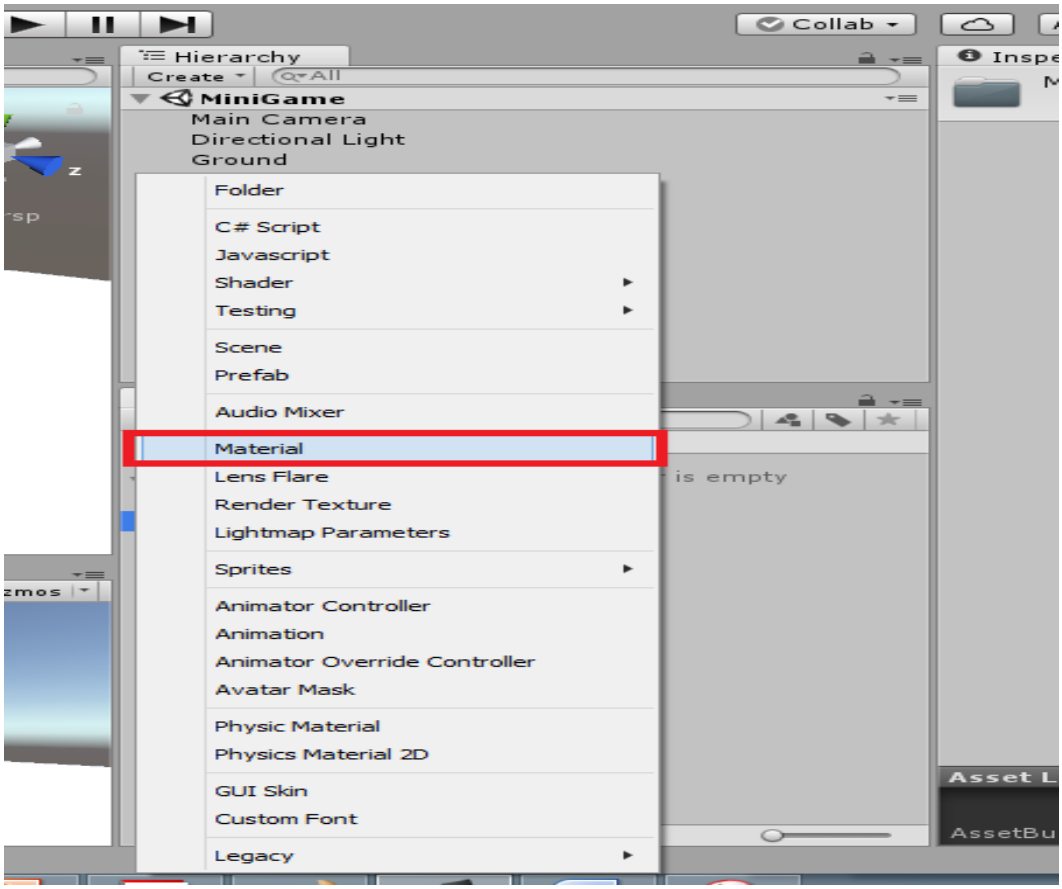


- Change transform's position of Player to (0, 0.5, 0).
  (All Unity primitive objects, cubes, spheres, capsules have a standard size, they are either (1, 1, 1) or (1, 2, 1) on X, Y, Z axis. So, we simply lift the player object up by half a unit in the Y axis.)
  To add color or texture to a model, we need to use a material.
- Create folder from Project view – create – Folder.

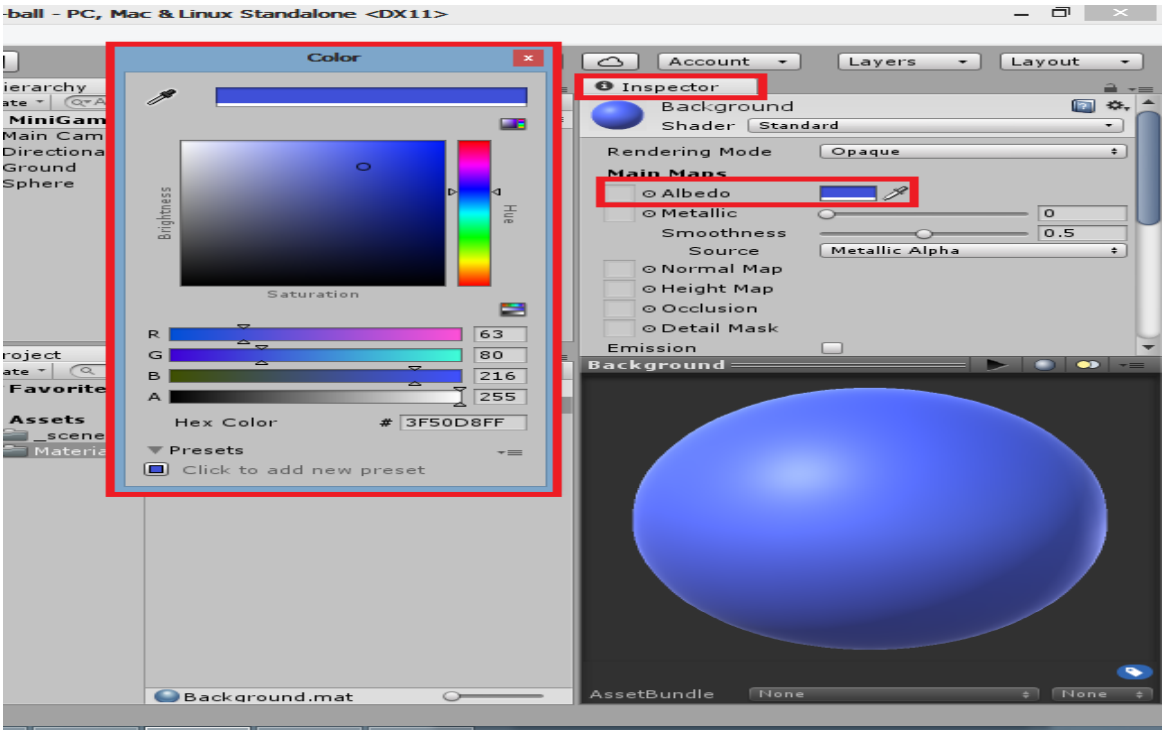




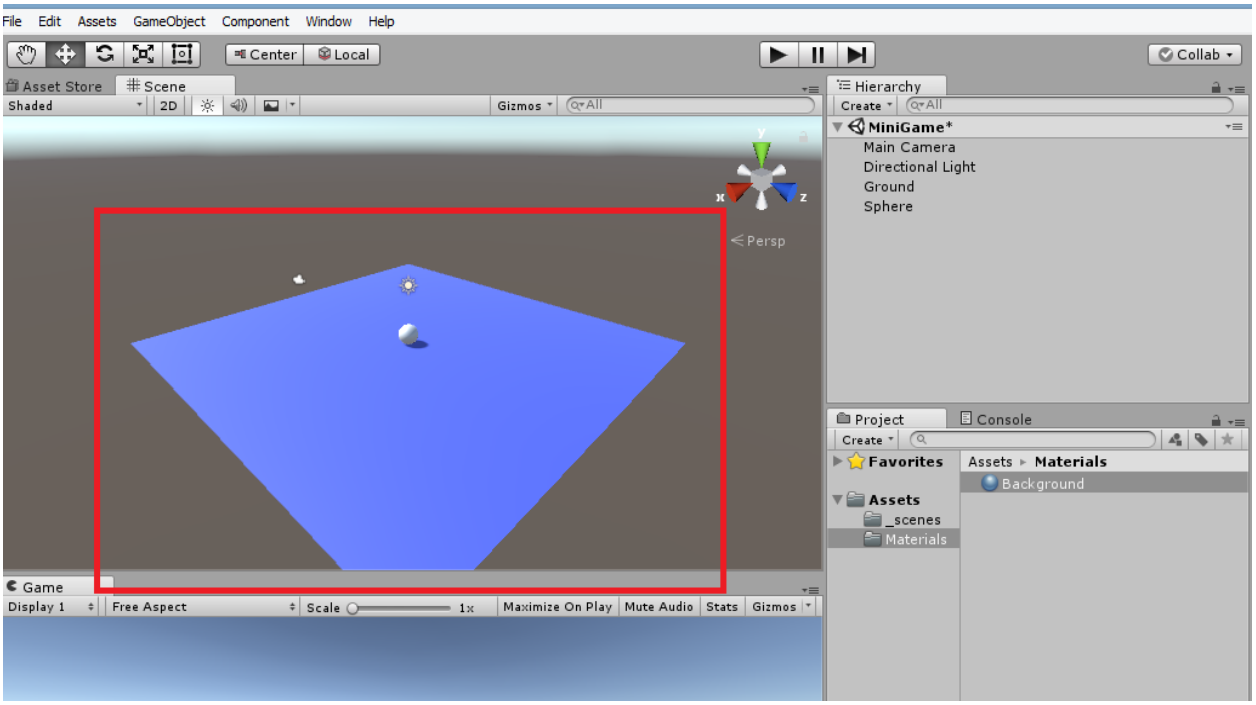
- Rename folder to Materials.
- Select Materials folder and use project's create menu again and this time choose Material.



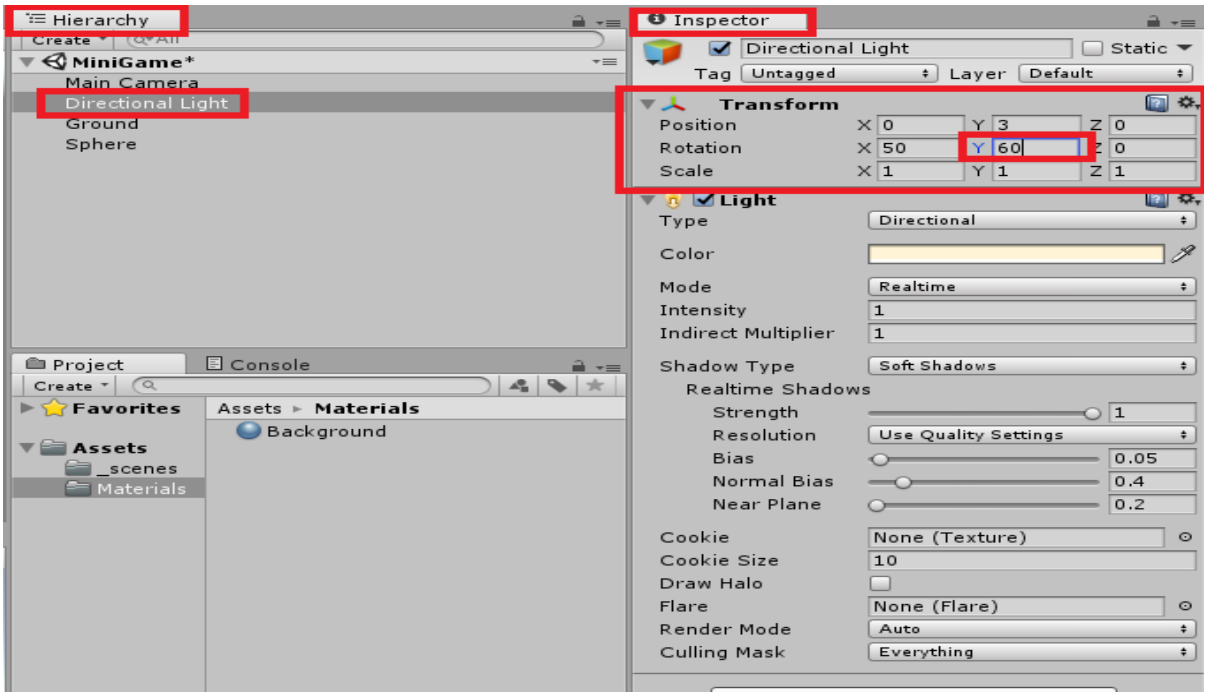
- Rename this material to Background.
- Select Background material and the first property is Albedo.
- Click on the Albedo's color field to open a color picker.
- Select the color as you want.



- 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.



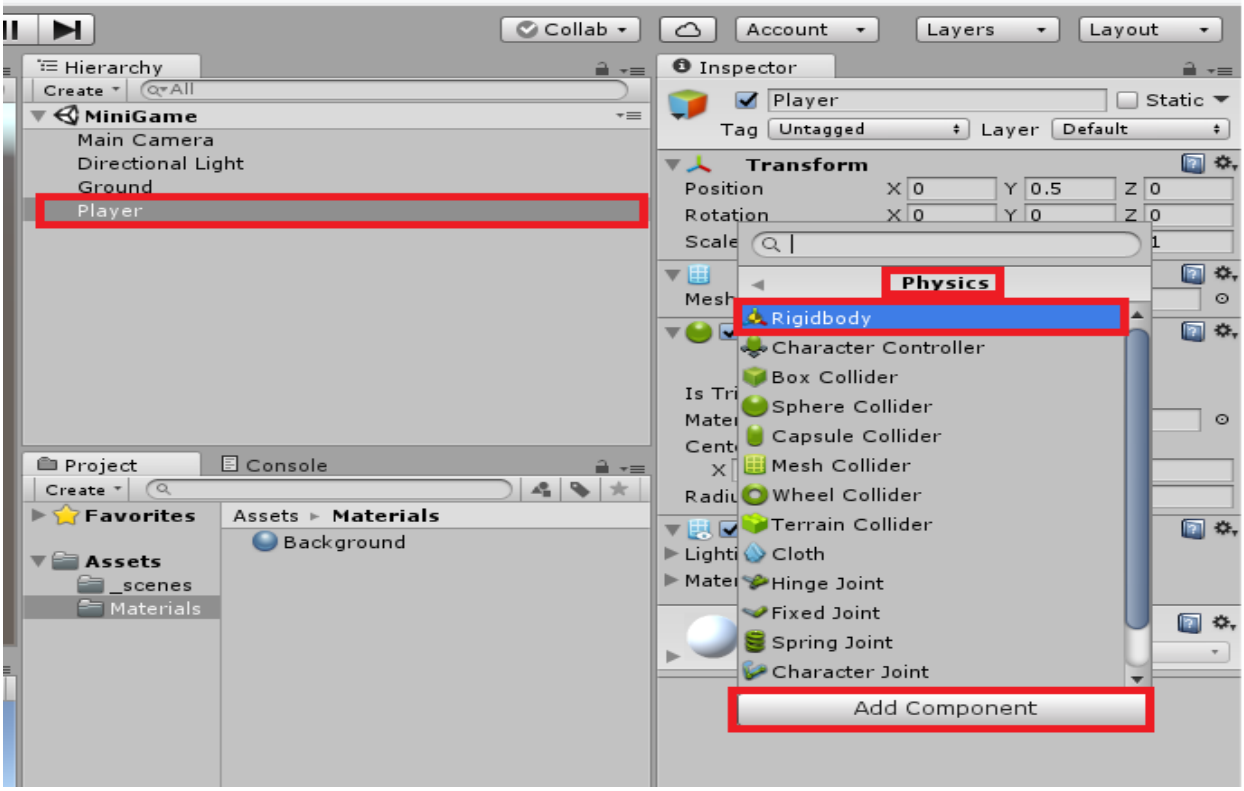
- Select the Directional Light from Hierarchy View and change the Transform's Rotation on the Y axis to 60 at the Inspector in the Transform component.



1.2. Moving the Player

Now we are going to move the player game object.We want to have the sphere roll around on the game area, bump in to walls, stay on the ground and not fly off into space. We want to be able to collide with our collectible game objects and pick them up when we do. This requires **physics**. To use physis the game object needs a rigidbody component attached.

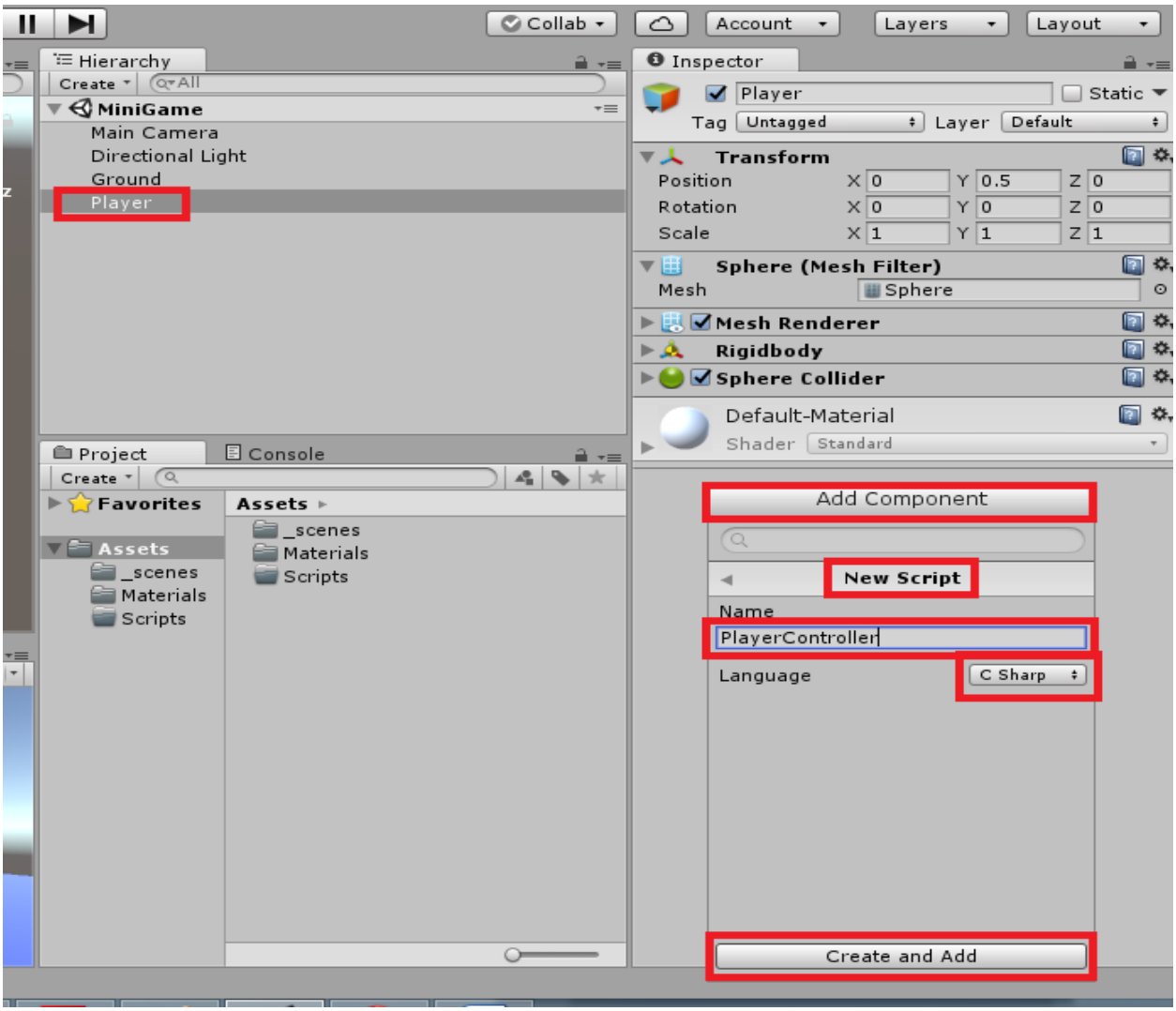
- Select Player game object and click Add Component, choose physics and then select Rigidbody.



We need to rearrange the order of the components on the game object using the context sensitive gear menu in the upper right of the component. Doing this has no effect on the performance of our game.

However, having a consistent order to the components on our game object may help us speed up our development by keeping or maintaining an organised project and hierarchy.

- In the project view click on the Create menu and choose Create Folder.
- Rename this folder Scripts.
- To create the C# script, select the Player game object and click Add Component and select New Script. And let's name this script PlayerController and click on Create and Add.



We need to move the PlayerController Script into the Scripts folder in the Project View.

- Open the PlayerController with double click on the script.
- Remove the sample code provided in the base script.

Next let's think, what do we want to do with this script?

We want to check every frame for player input and then we want to apply that input to the player game object every frame as movement.

Where will we check for and apply this input?

We have two choices. Update and Fixed Update.

-Update is called before rendering a frame and this is where most of our game code will go.

-Fixed update on the other hand is called just before performing any physics calculations and this is where our physics code will go.

We will be moving our ball by applying forces to the rigidbody, this is physics. So we will put our code in Fixed Update.

- Add this code for Player movement with speed.

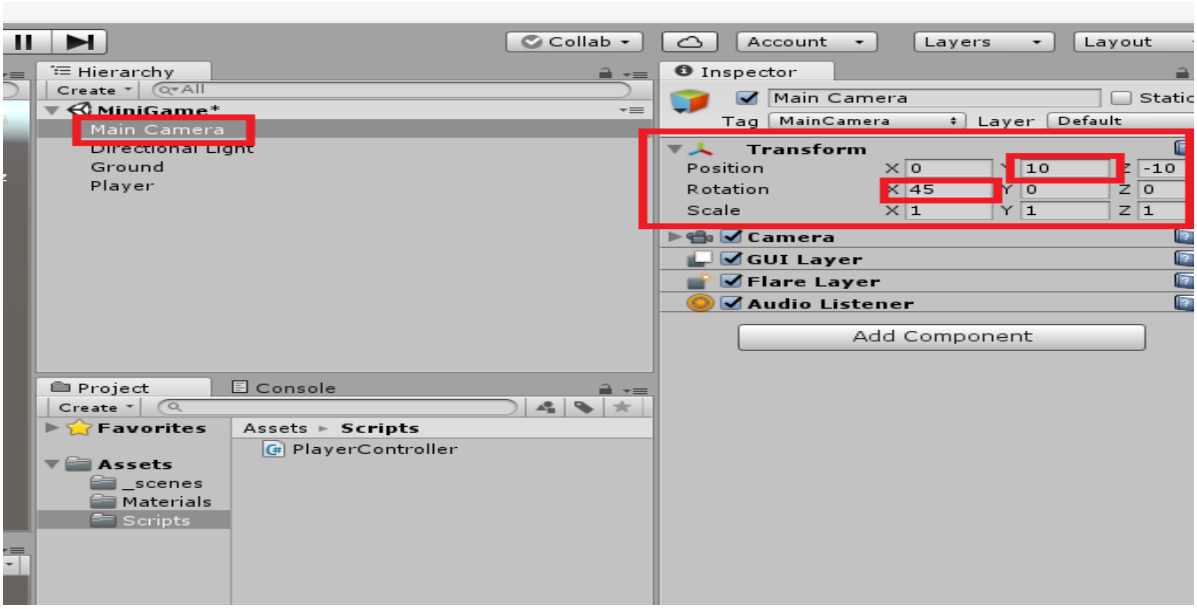
```
using UnityEngine;
using System.Collections;

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);
    }
}
```

- In Unity, select Player game object and set Speed value to 10 at Inspector in PlayerController Script Component.

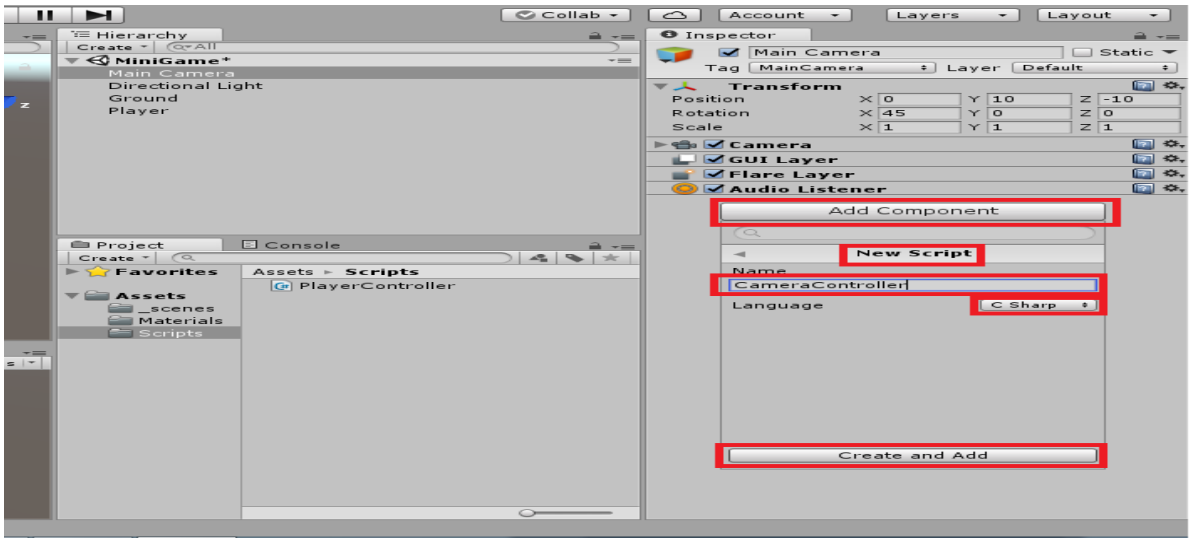
2. Camera and Play Area

2.1 Moving the Camera

- Select Main Camera and change Transform position value to (0, 10, -10) and rotation value to (45, 0, 0)



- To create the CameraController Script, click Add Component and choose New Script and give name as CameraController and finally click Create and Add.



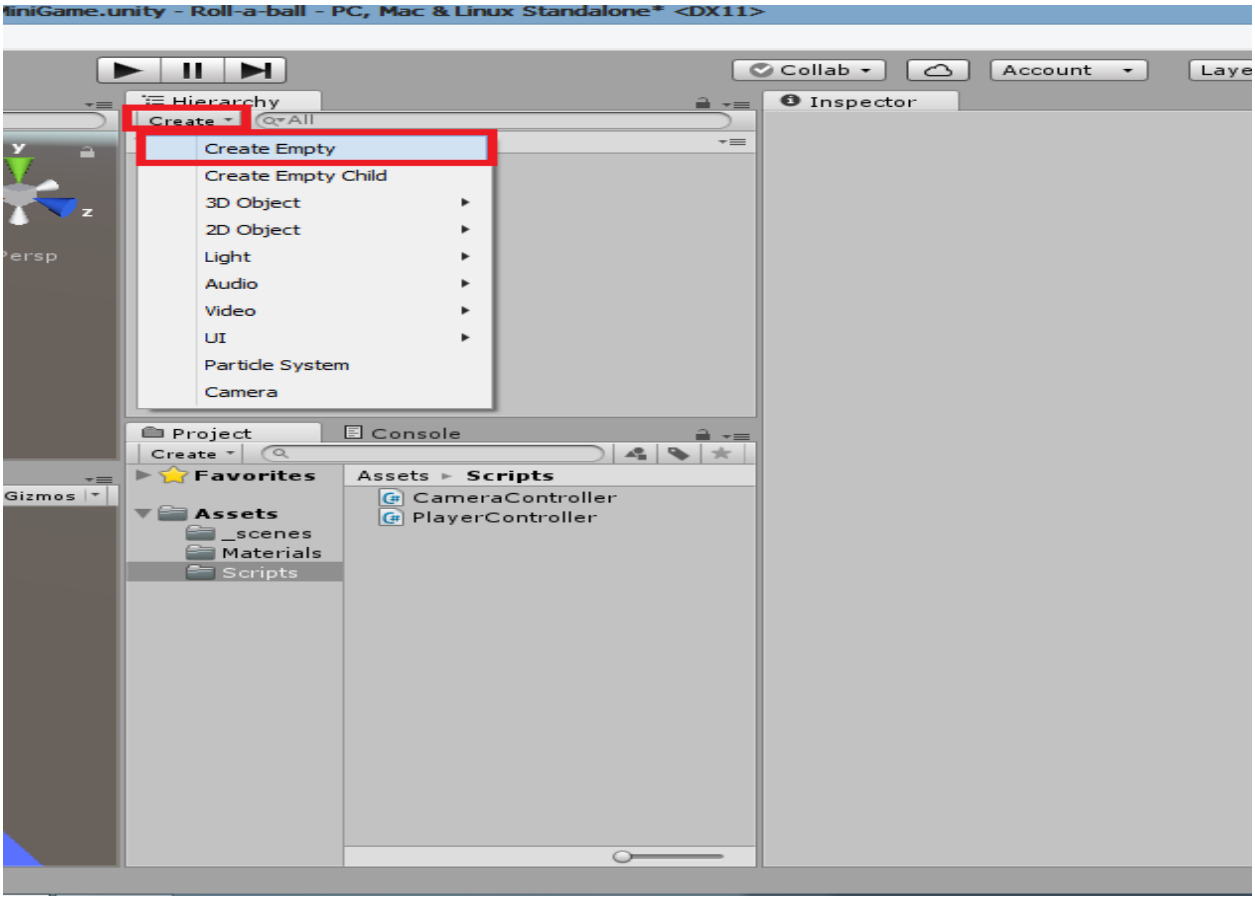
- Remove the CameraController script in the Project View to the Scripts folder.
- Open the CameraController script with double click.
- Set the code as shown in bellow.

```
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;
    }
}
```

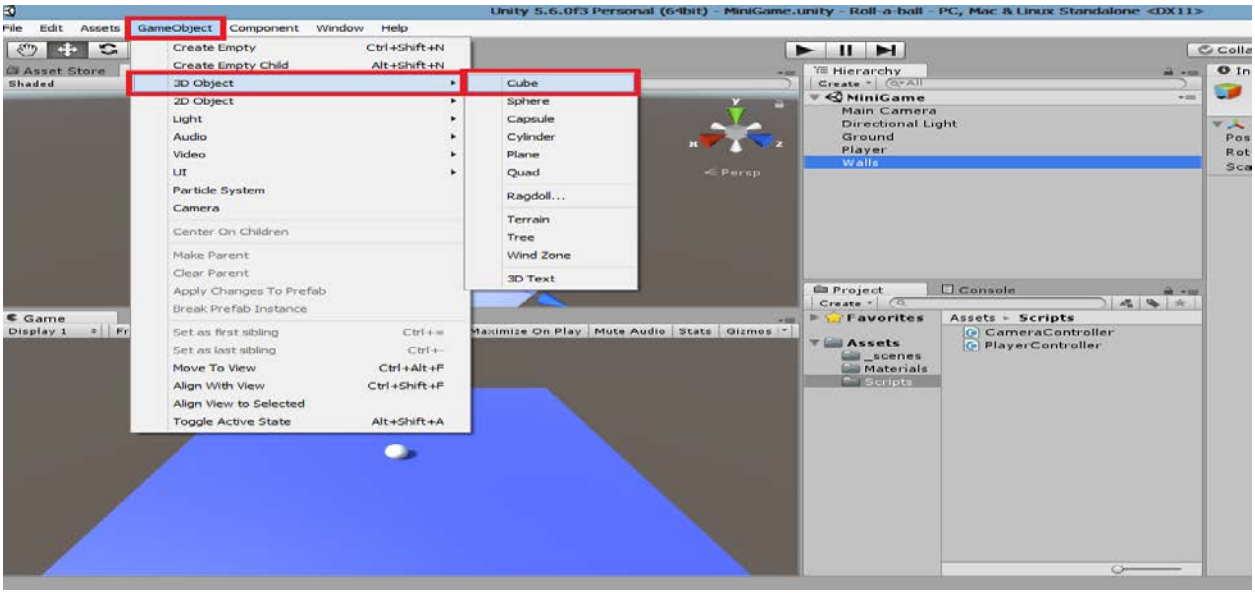
2.2 Setting up the Play Area

We will place walls around the edges to keep our player game object from falling off and we will create and place a set of collectable objects for our player to pick up.

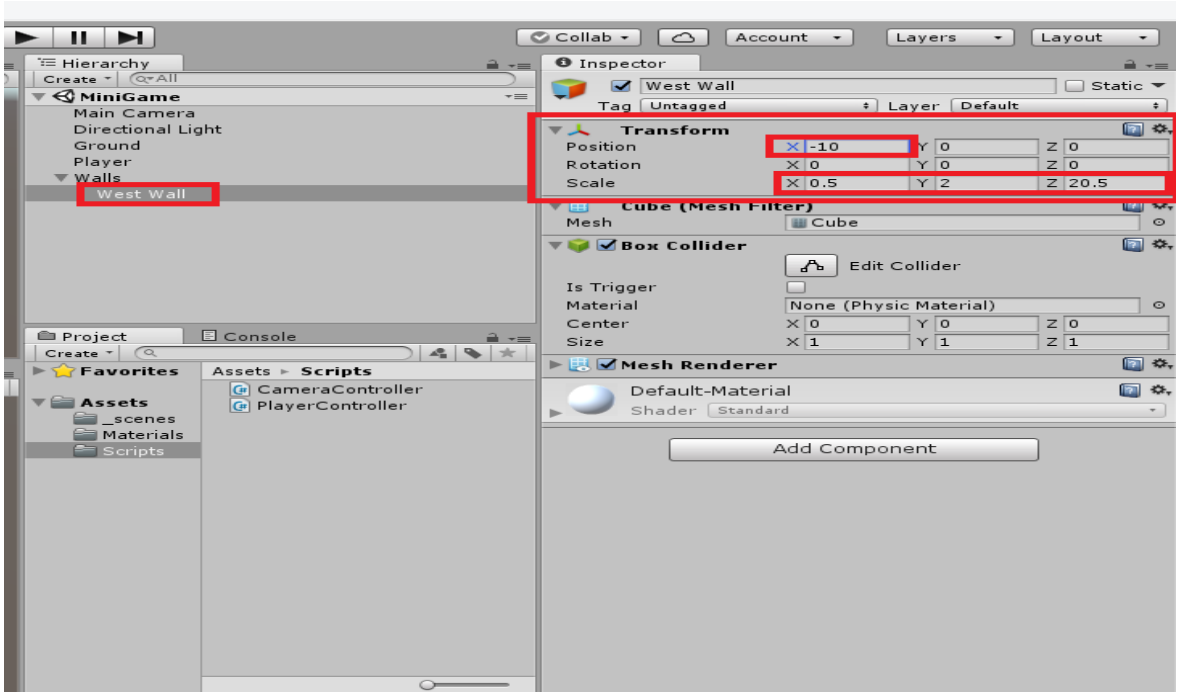
- First let's create a new game object from Hierarchy – create menu – Create Empty.



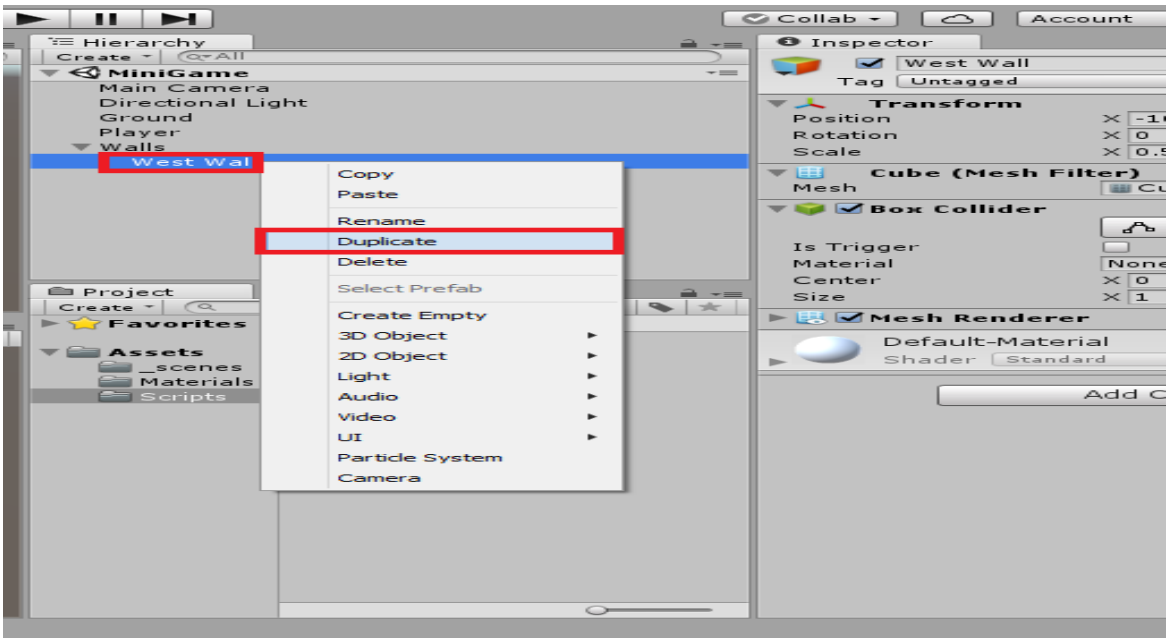
- Rename it as Walls.
- Reset this game object to origin.
- Create a new cube from GameObject – 3D Object - Cube.



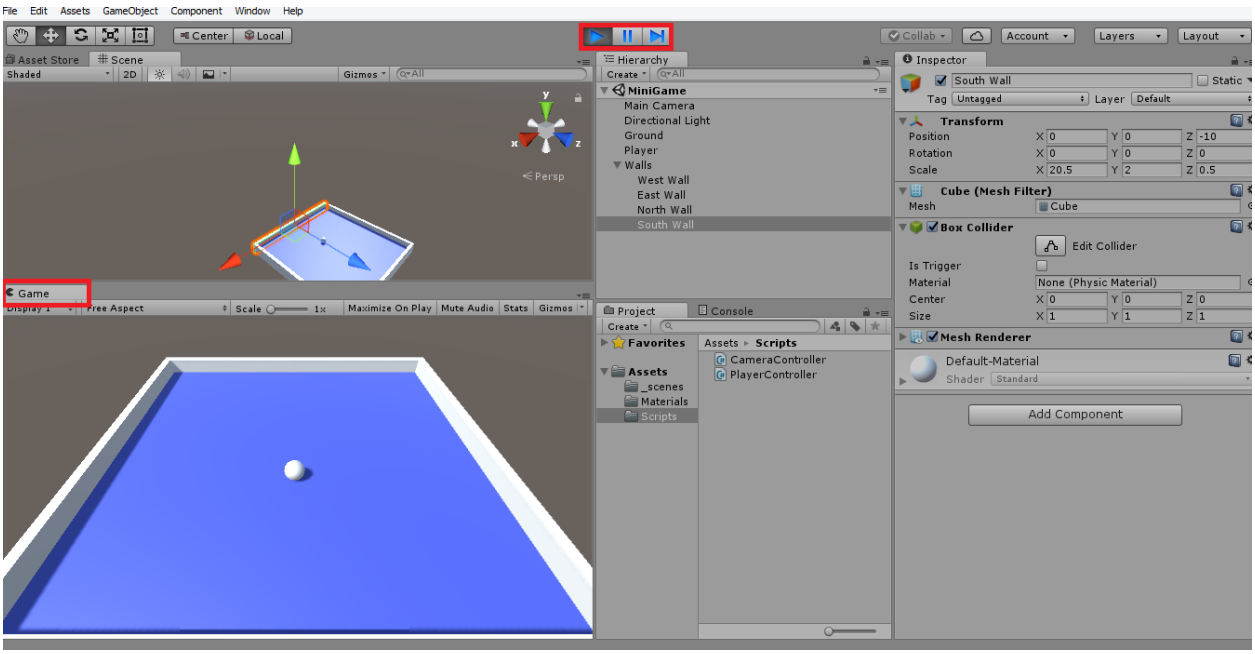
- Rename this West Wall and reset this game object to origin.
- Drag West Wall to Walls game object.
- Change transform's scale of X, Y and Z to (0.5, 2, 20.5).
- Set the transform's position X value to -10.



- Right click on the West Wall and choose duplicate.



- Rename it as East Wall.
- Change the transform’s position X value to 10.
- Duplicate the East Wall game object and then rename North Wall.
- Change the transform’s scale to (20.5, 2, 0.5).
- Set the value 10 to the transform’s position Z.
- Duplicate the North Wall game object and call it South Wall.
- Change the transform’s position Z value to -10.
- Now enter play mode and test and then exit play mode.

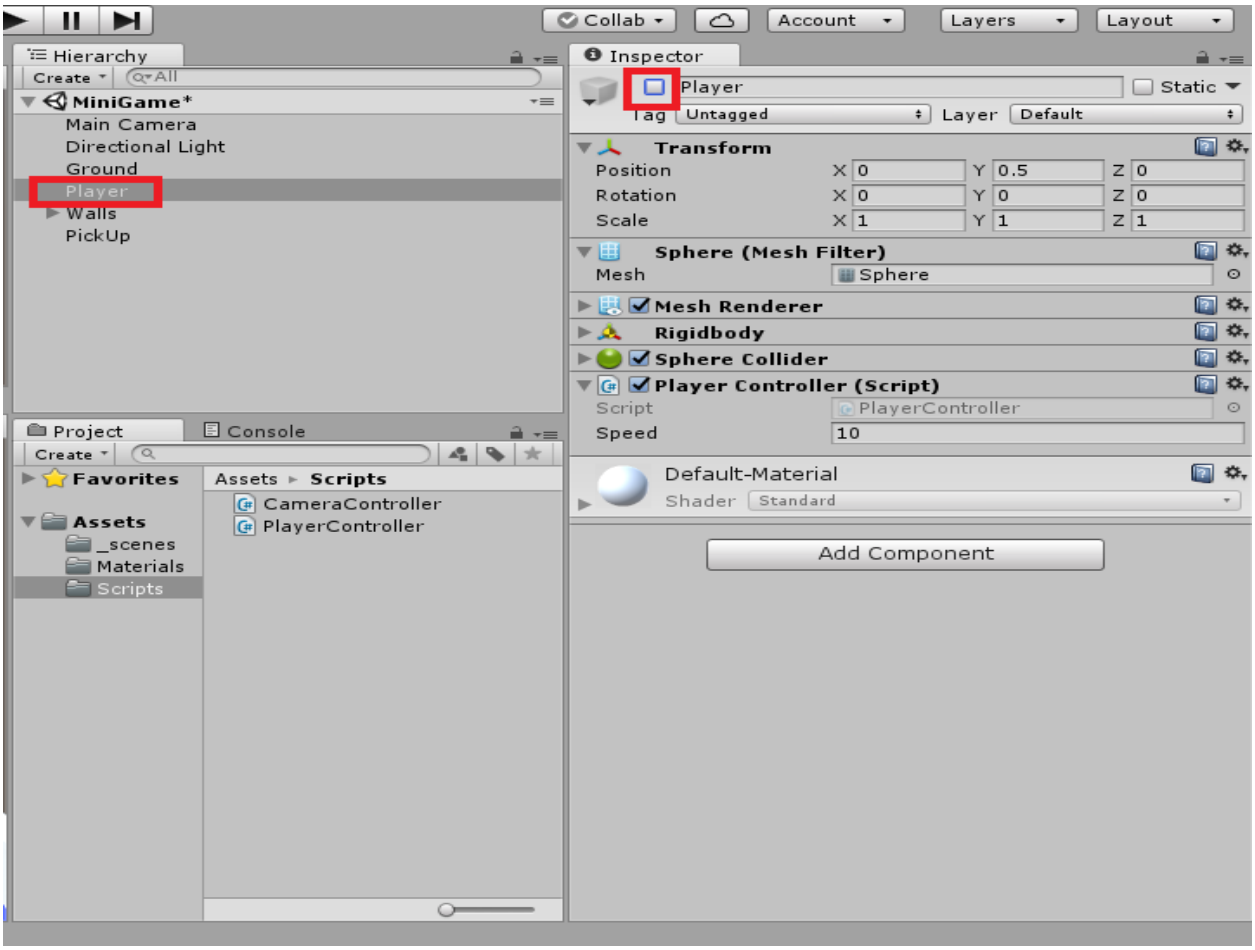


3. Collecting, Scoring and Building the game

3.1 Creating Collectable Objects

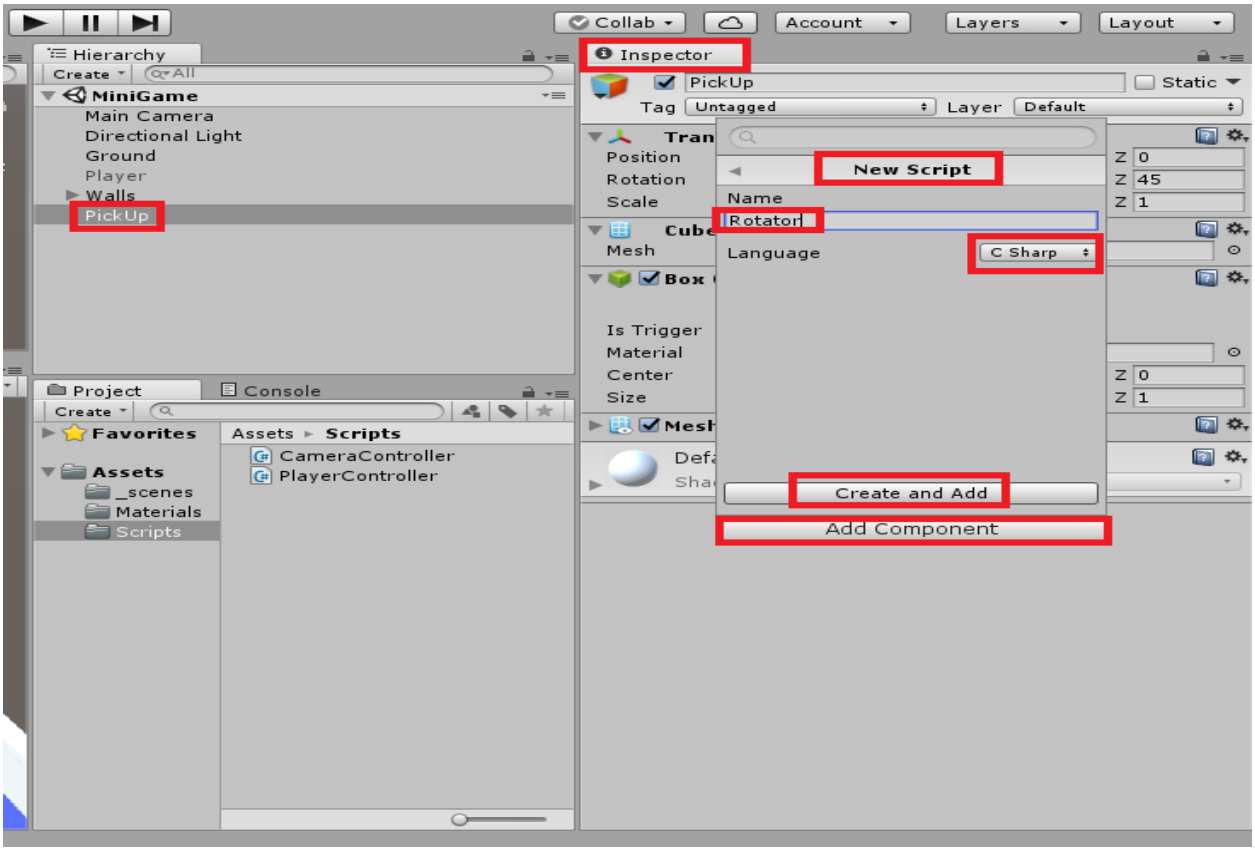
Next let's create our collectable objects.

- Create a new cube and rename it PickUp.
- Reset the PickUp's transform to origin.
- Now, Player and PickUp game objects places are the same so select the Player game object and deselect the checkbox in front of the Name field. This is the game object's Active checkbox.



- Now the cube scale is 1 by 1by 1 so set the transform's position Y value to 0.5.

- Set the transform's rotation value to (45, 45, 45).
- To rotate the PickUp object, create the script on the PickUp game object by using Add Component button in the Inspector. Create a new script called Rotator.

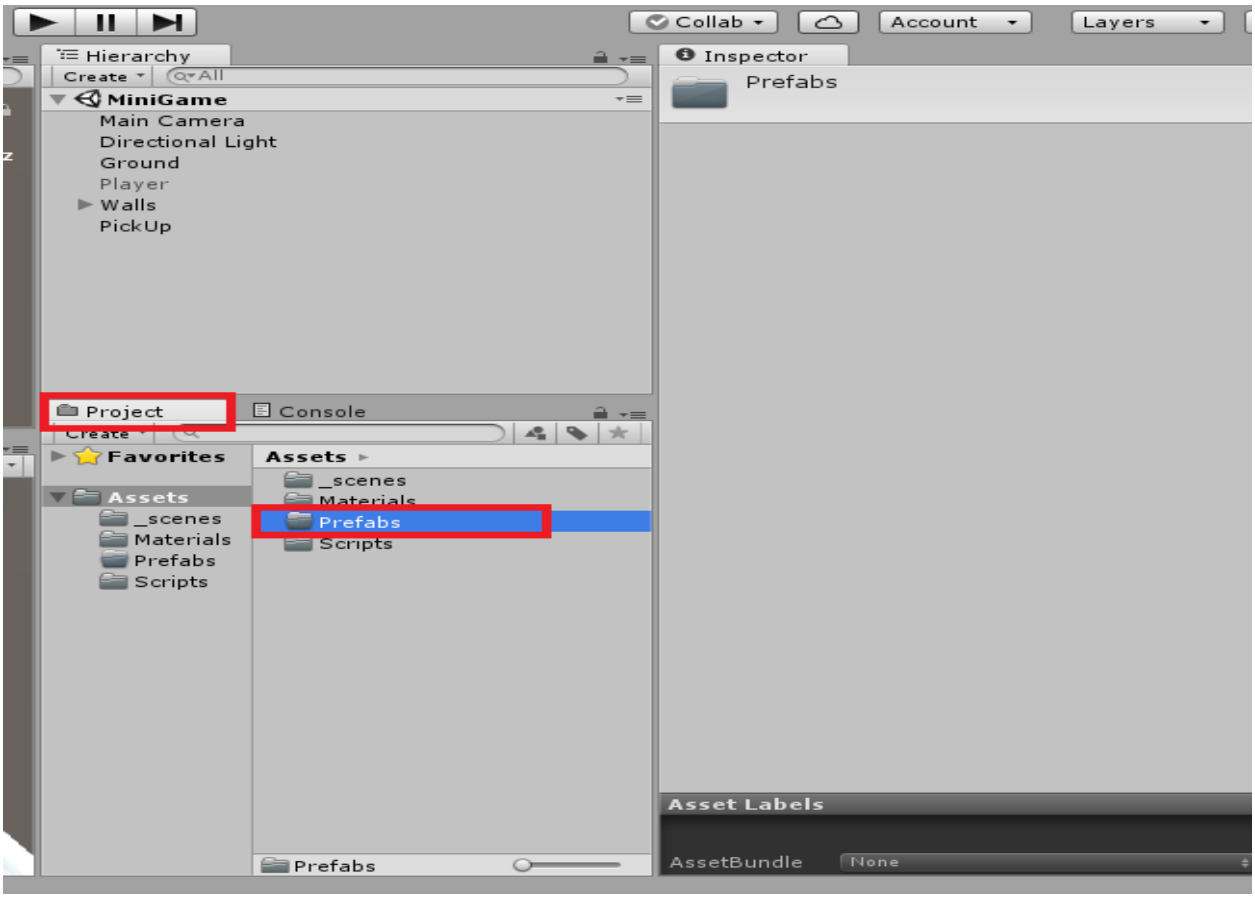


- Drag the Rotator script to the Scripts folder.
- Open the script with double click on it.
- The script code is shown as figure.

```
using UnityEngine;
using System.Collections;

public class Rotator : MonoBehaviour {

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

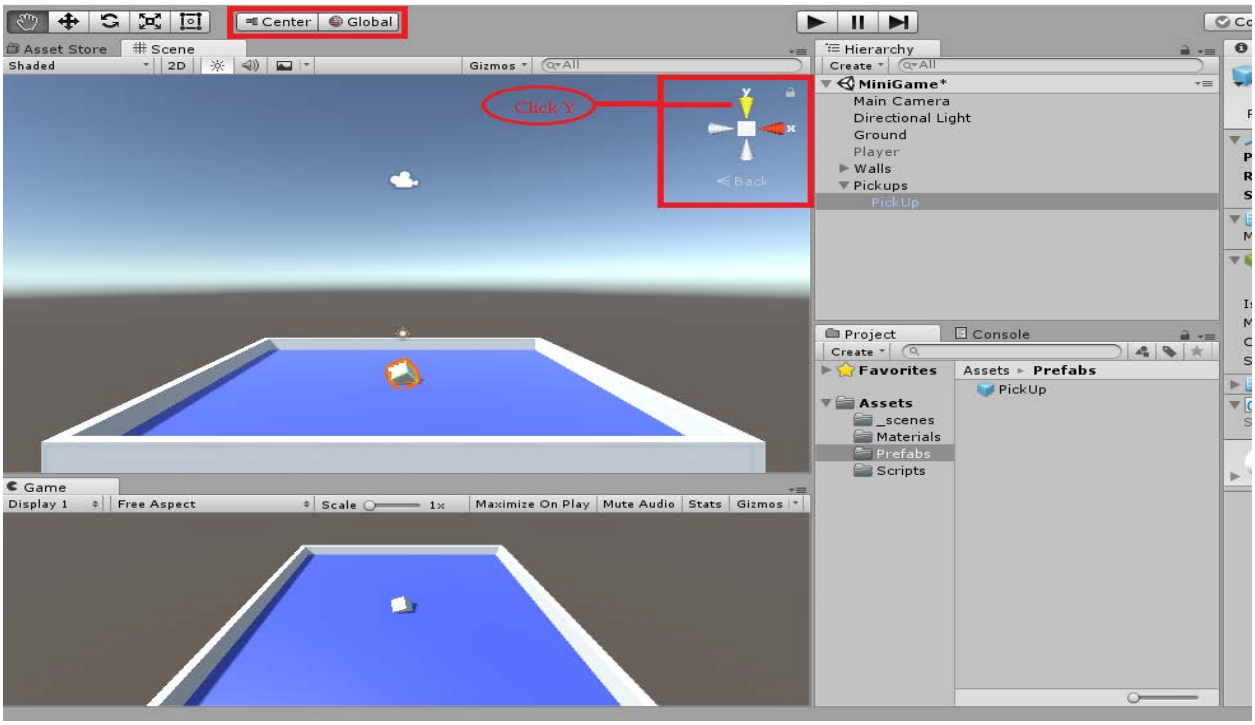
Next we want to place these around the game area. But before we do this we need to do one important step. We need to make our PickUp object into a prefab. A **prefab** is an asset that contains a template, or blueprint of a game object or game object family.

- Select the project view and then choose Create – Folder.
- Rename this folder Prefabs.



- Drag the PickUp game object from our hierarchy and place it into our Prefabs folder.
- Create a new empty game object at Hierarchy view and call it Pickups.
- Reset this empty game object.
- Drag PickUp game object into Pickups empty game object.

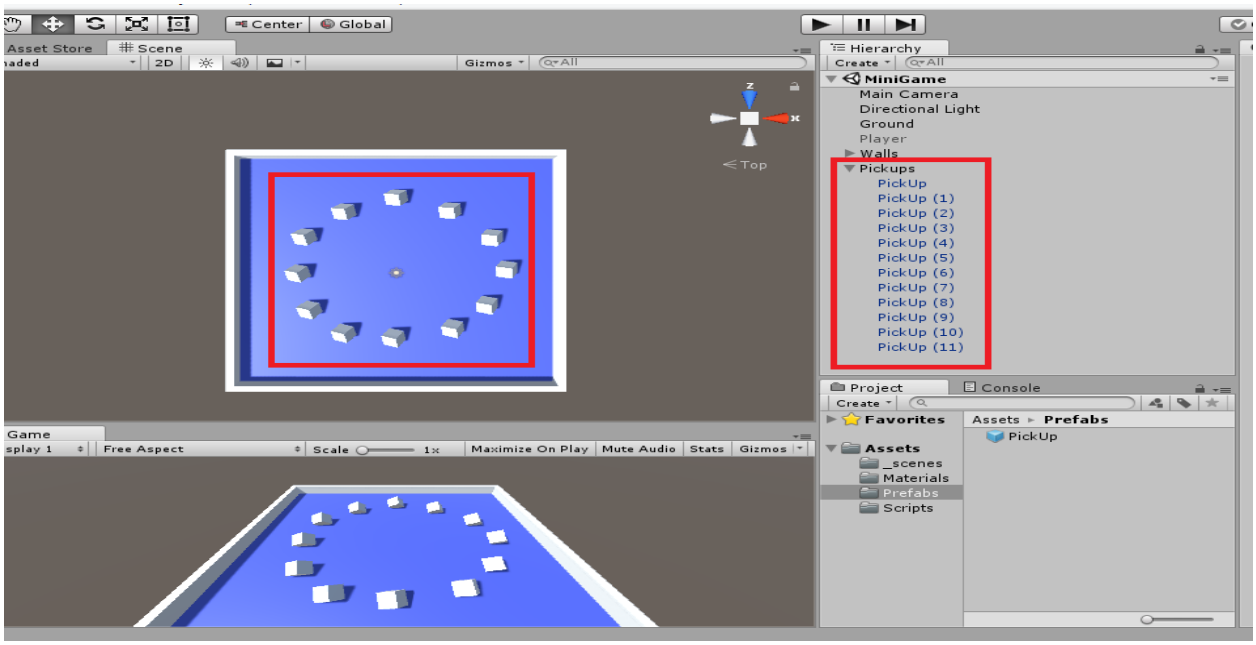
- Select PickUp game object and change view into a top-down view by clicking on the gixmo at the upper-right of our scene view.
- Change the editor to Global mode.



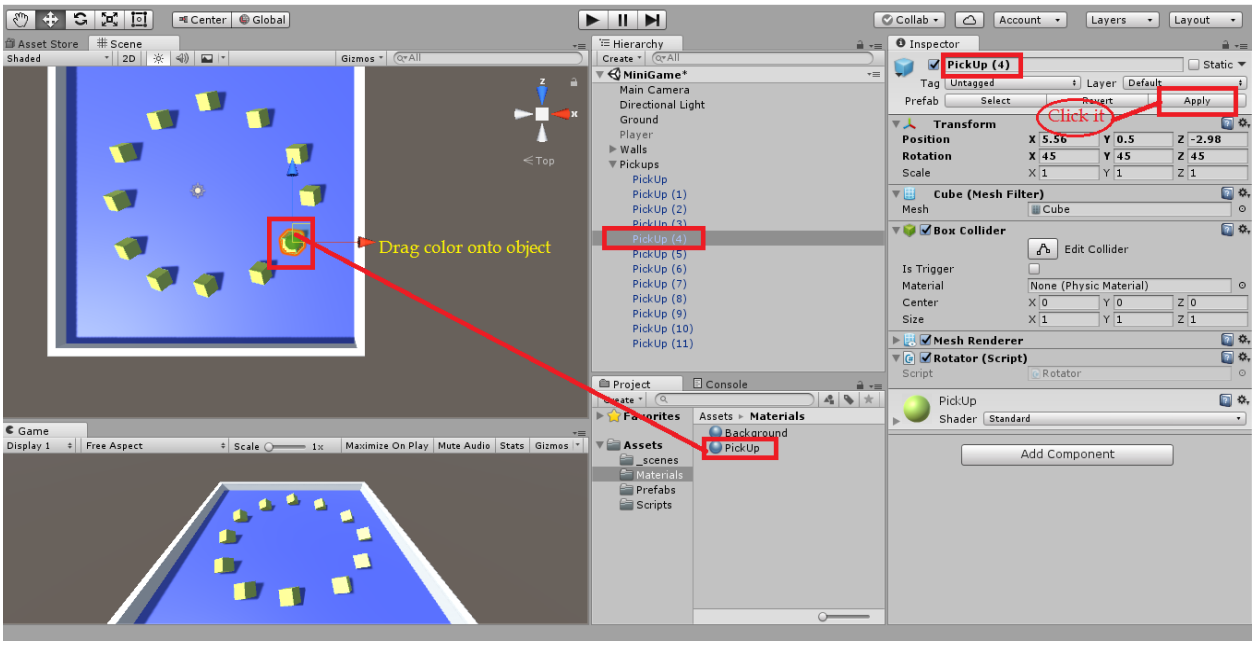
- Take our first PickUp object and place it on the game area, some place convenient.
- Select game object and duplicate it and place the second instance of the prefab and then create a few more until 12 and placing them around the play area.

(Note: I'm moving parallel to the ground or X/Z plane by selecting the X/Z plane at the center of the gizmo.)

- Let's hit play and test.



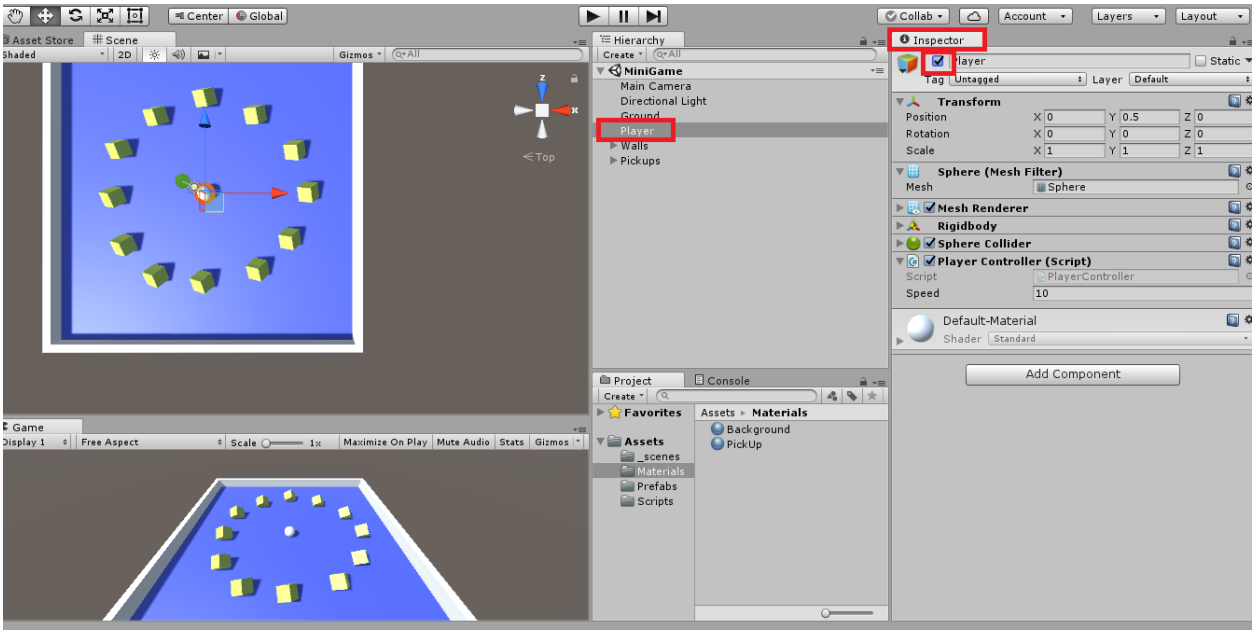
- Let's change their color, select our existing material at the project view in Materials folder and duplicate it.
- Rename this new material PickUp.
- Change the albedo color property to yellow.
- Drag color on just one instantiated prefab and use Apply button to apply those changes to the prefab asset.



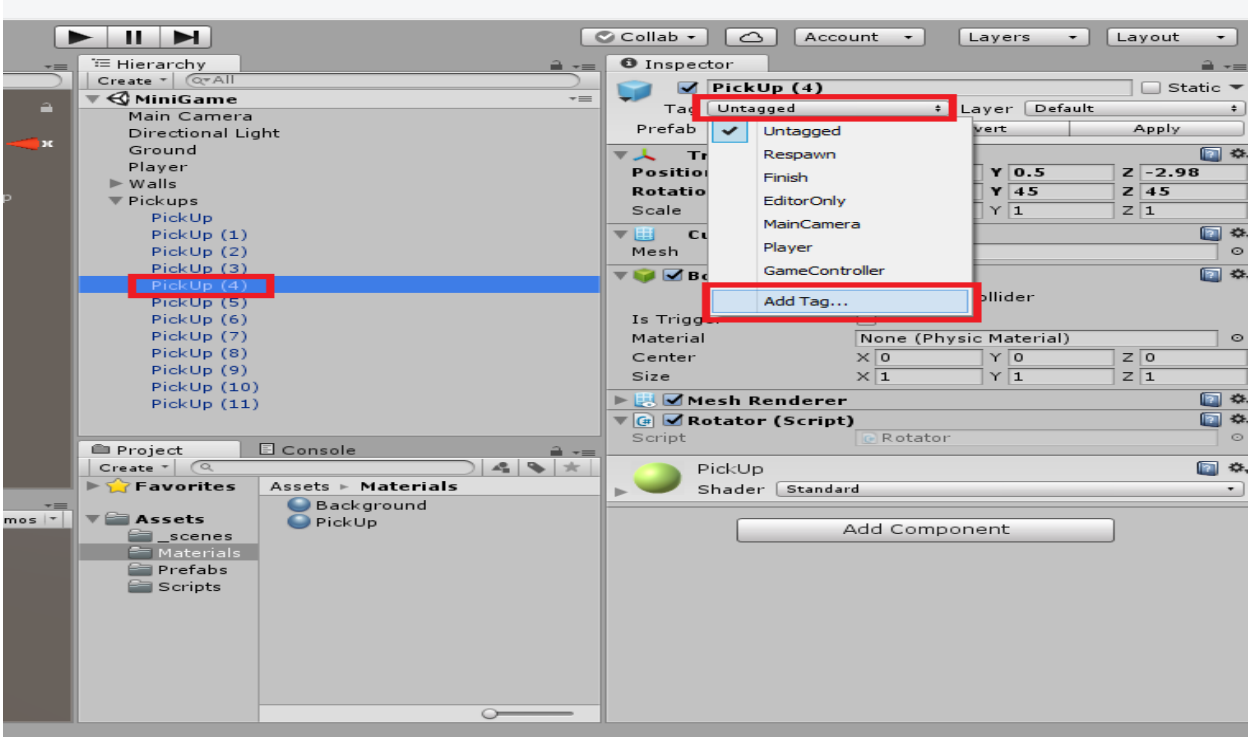
3.2 Collecting the Pick Up Objects

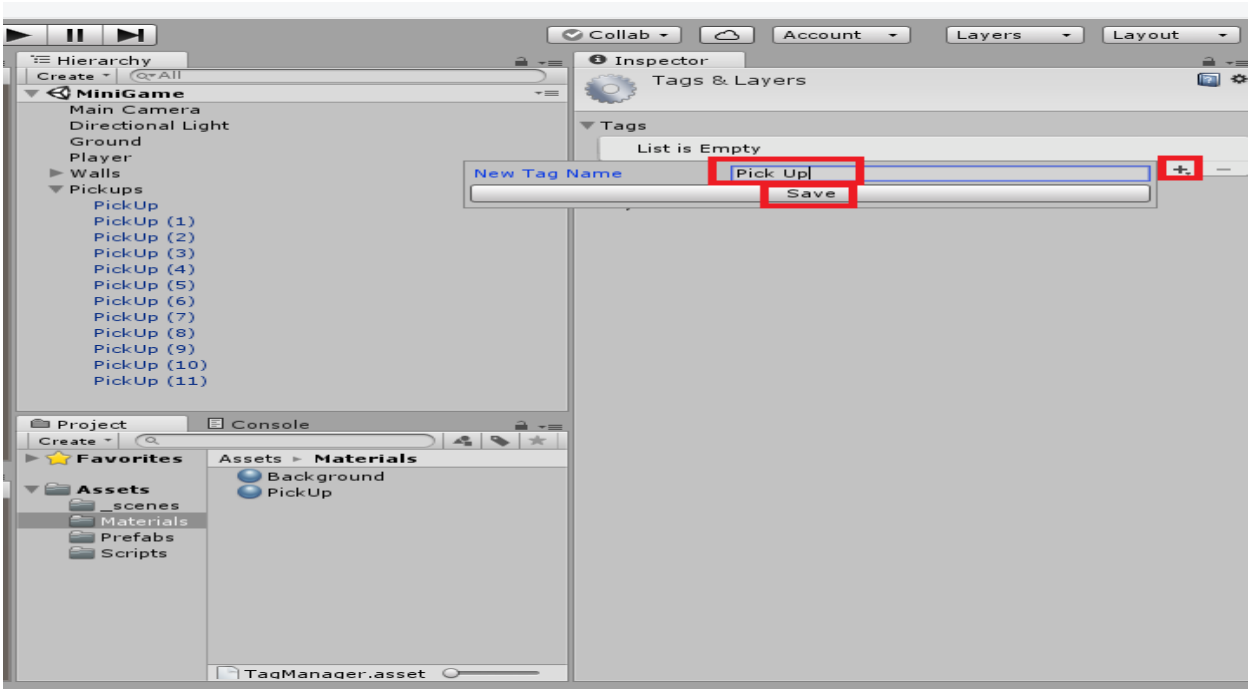
We want to be able to pick up our collectable game objects when our player game object collides with them. To do this we need to detect our collisions between the player game object and PickUp game objects.

- First let's tick the Active checkbox at Inspector and bring back our player.

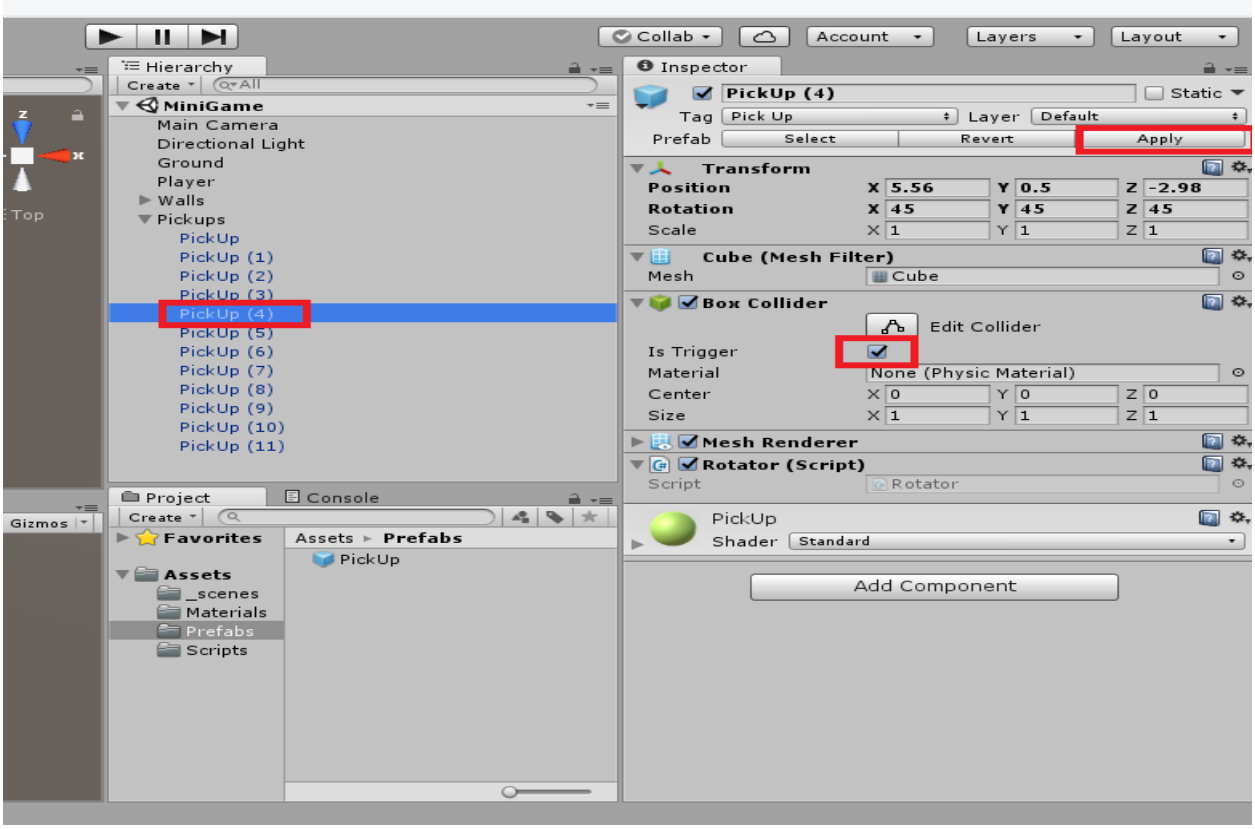


- Select the PlayerController script and open it for editing. (The code is below in this section.)
- Set the function onTriggerEnter( ) that check the collider and we need the tag for CompareTag so let's add the tag.
- Select one of the PickUp game object, click Untagged and select Add Tag and click + button and then set the name as Pick Up and save.

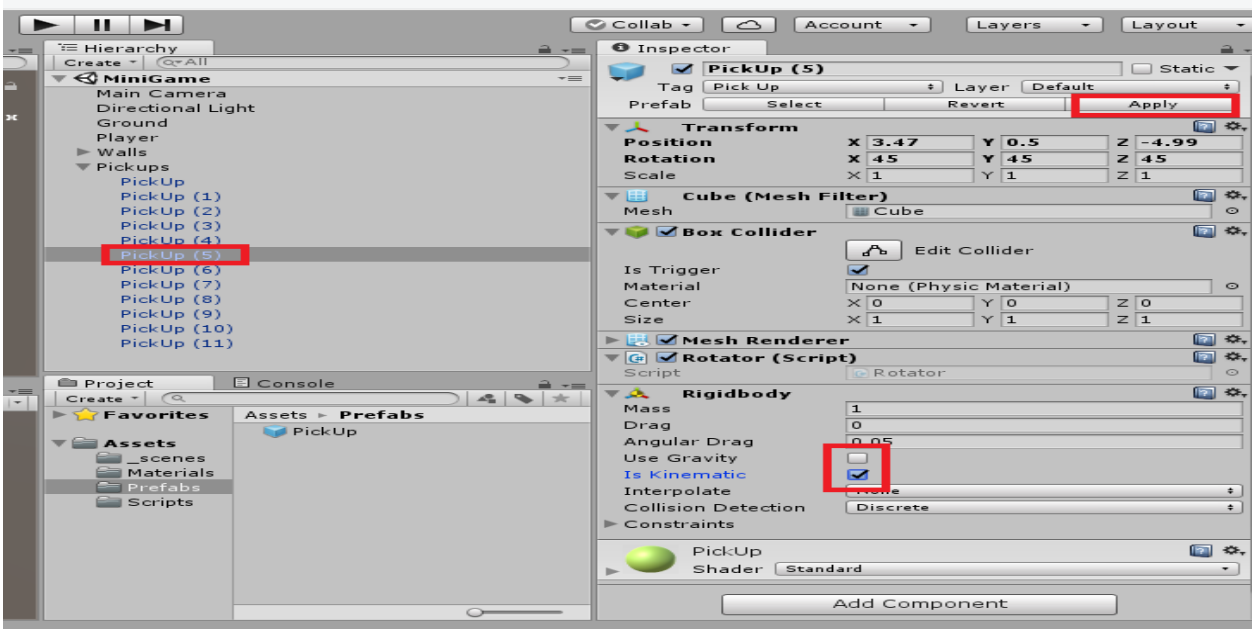




- Select tag again and choose PickUp from the list and then click Apply button to change all PickUp game objects.
- Select Is Trigger in the box collider component of the PickUp game object and then click Apply.



- Let's save our scene, enter play mode and test.
- Add Rigidbody on the PickUp game object and deselect Use Gravity and select is Kinematic and then click Apply button.



```
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);
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.CompareTag ("Pick Up"))
        {
            other.gameObject.SetActive (false);
        }
    }
}
```

3.3 Displaying the Score and Text

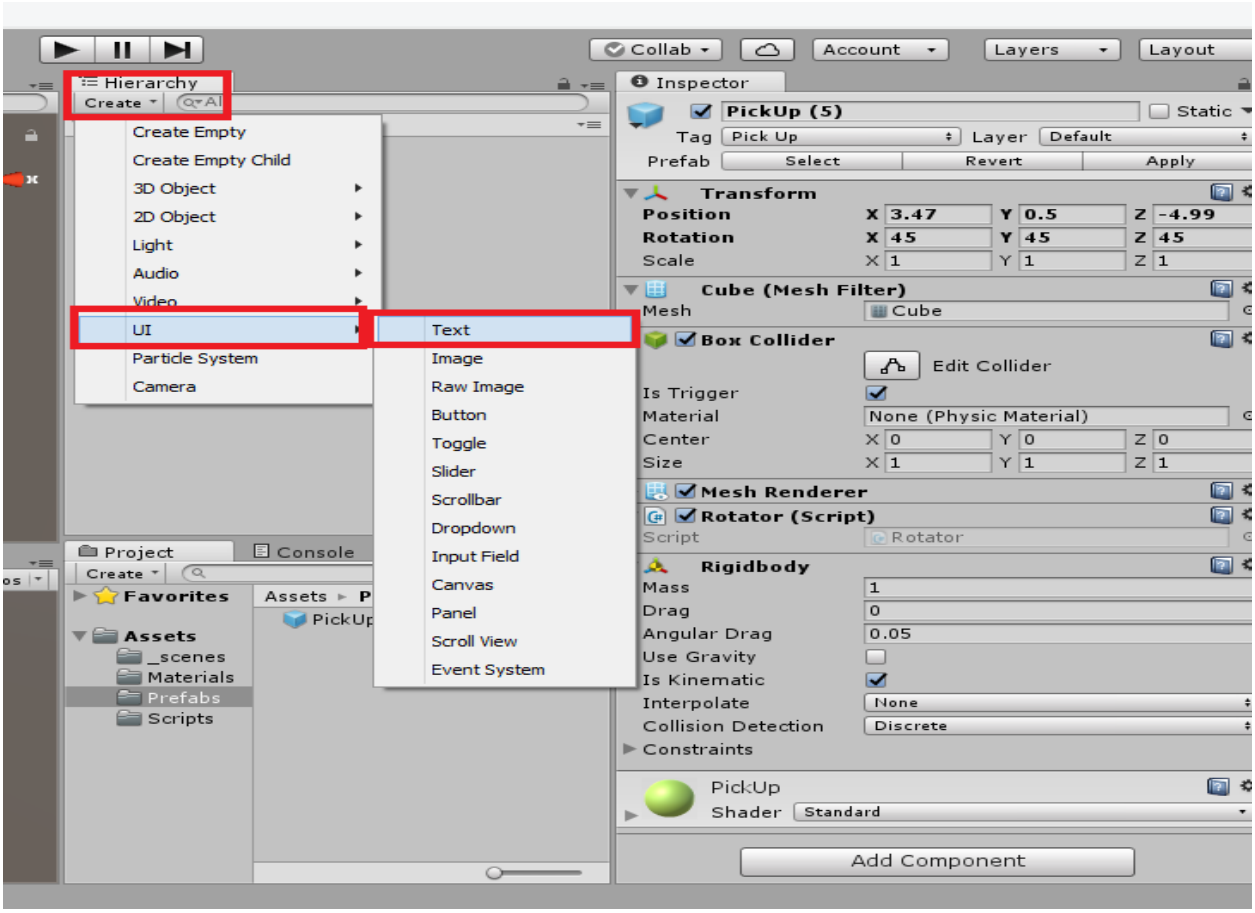
This section is counting, displaying text and ending the game.

- Select the Player game object and open the PlayerController script for editing.
- Add a private variable to hold our count.

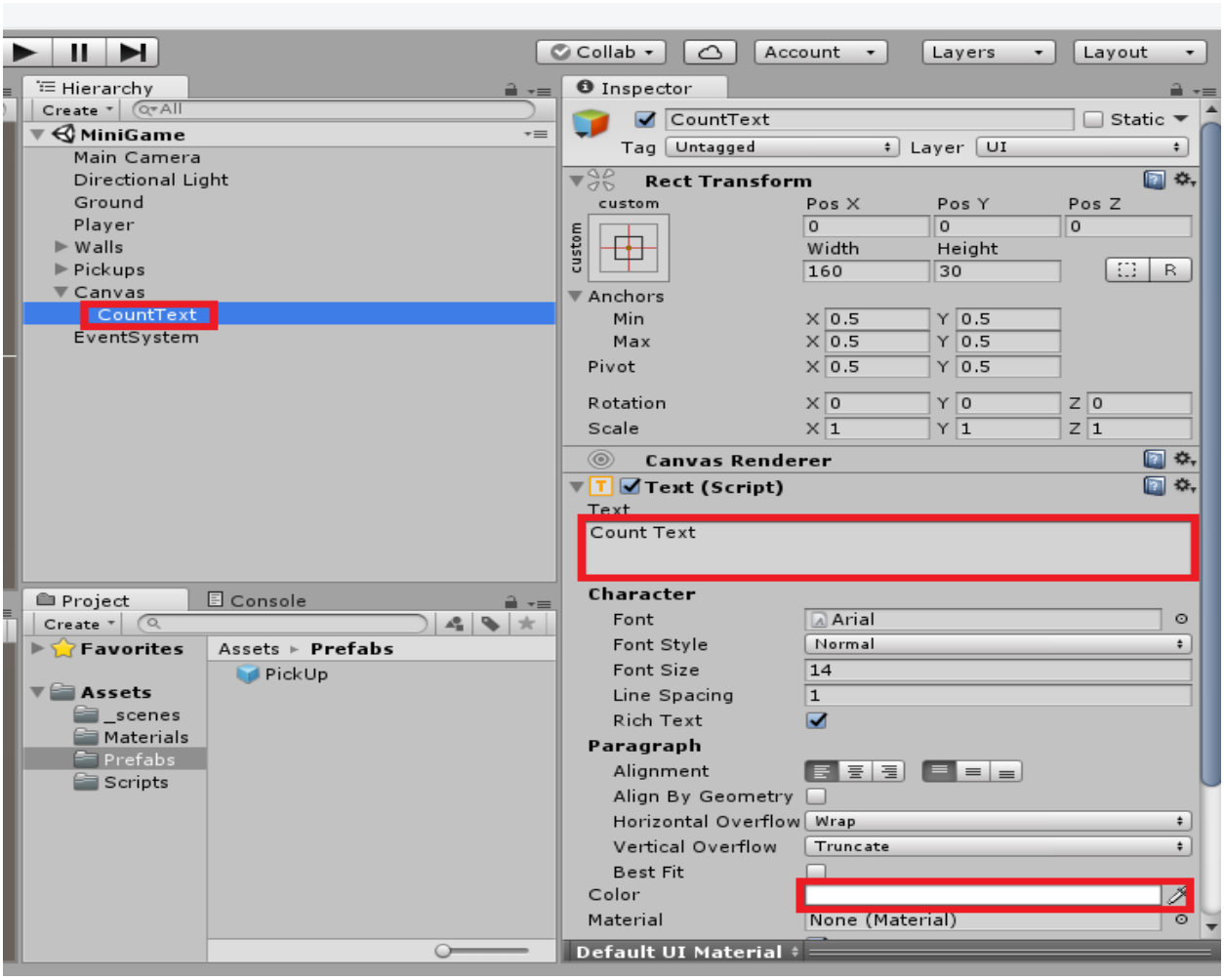
(Note: private variable don't have any acces to it in the inspector.)

To display text in this tutorial we will be using Unity's UI Toolset to display our text and values.

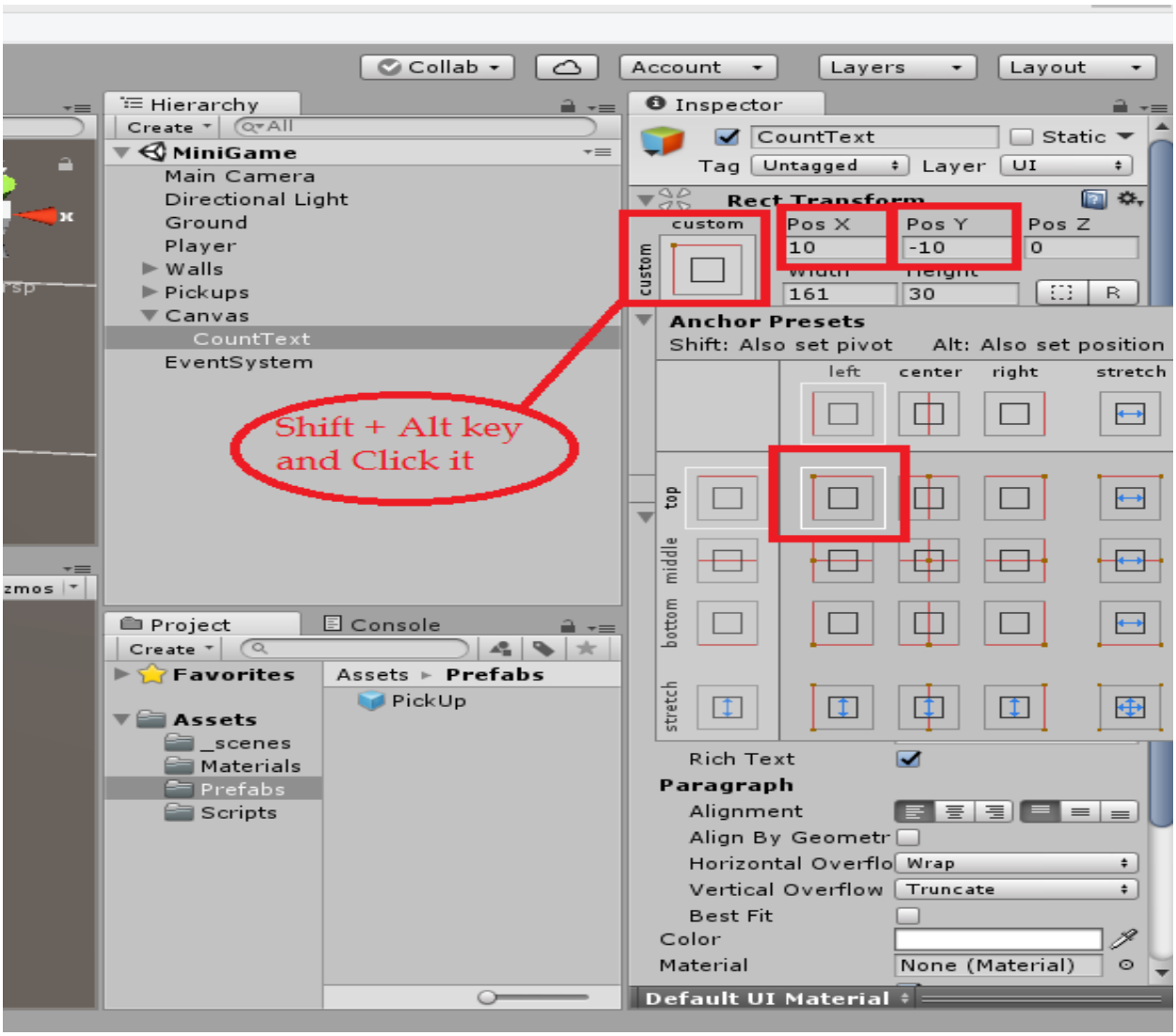
- First let's create a new UI test element from the hierarchy's Create menu.



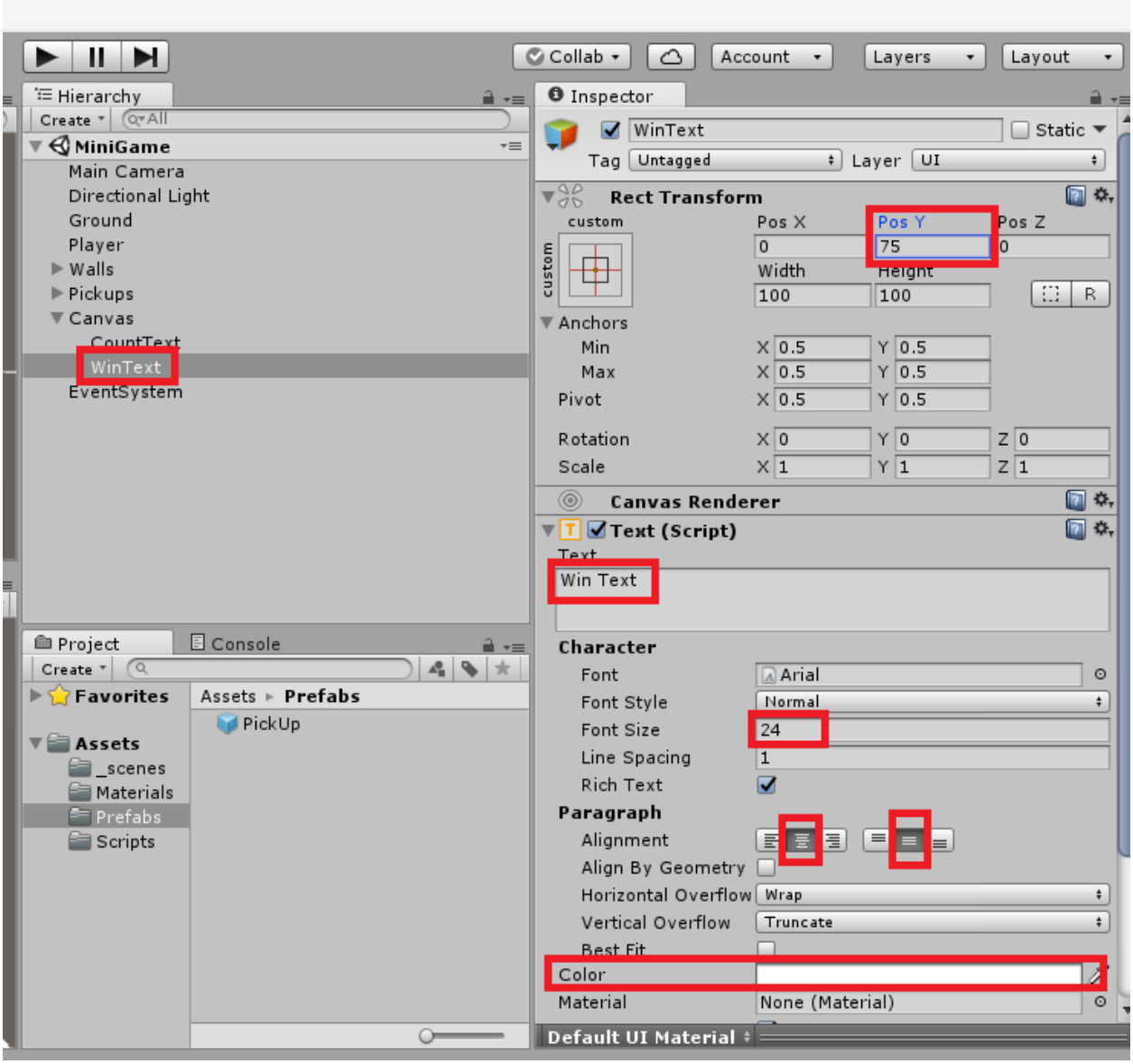
- Rename the text element CountText.
- Change Text color into white.
- Add Count Text inside Text.



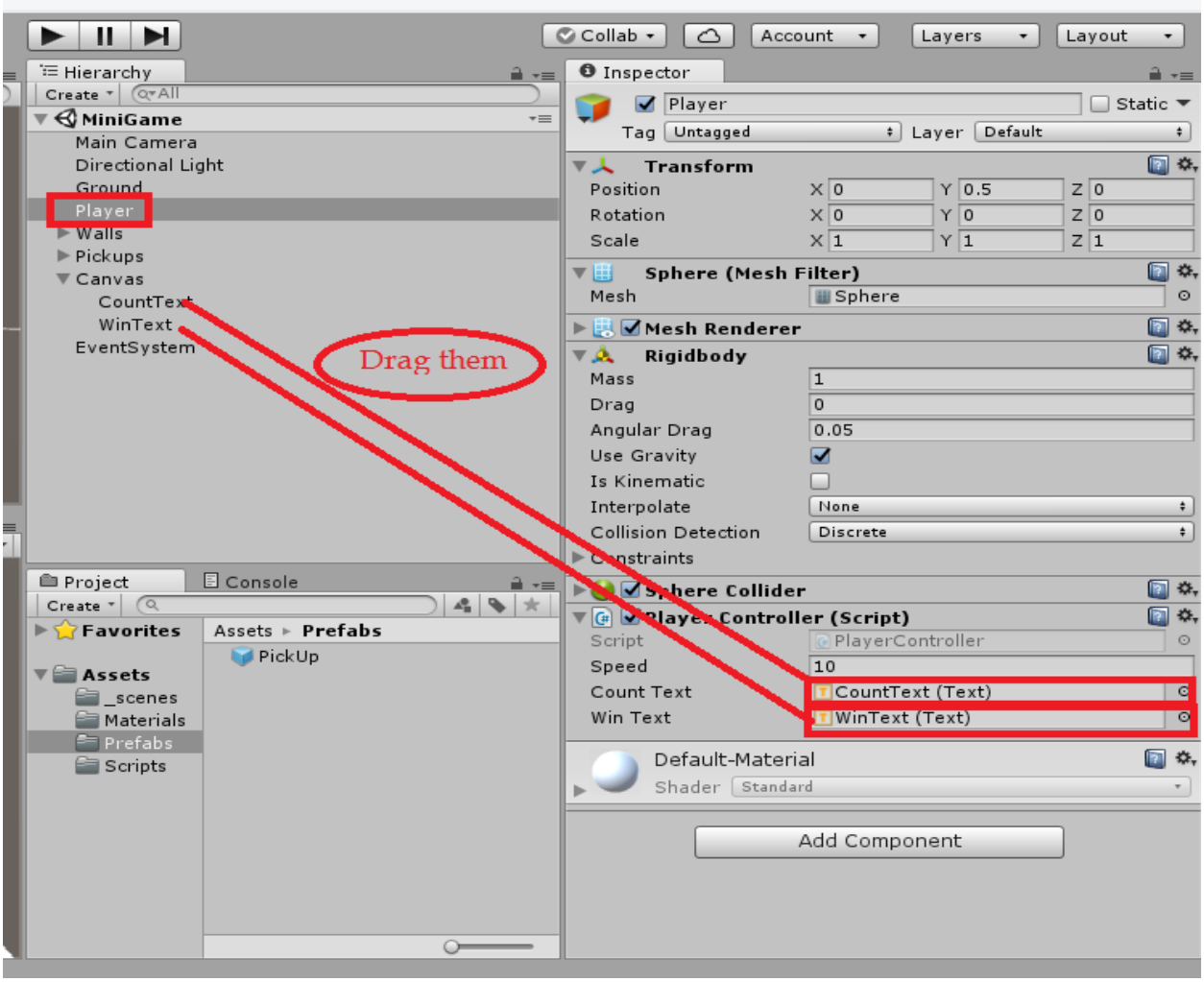
- Move the count text element into the upper left corner of the convas so we will hold down both the Shift and the Alt keys and then select the upper left corner button.
- Change the rect transfoem's Pos X and Pos Y values to 10 and -10.



- Create a new UI text game object and rename it Win Text.
- Reset it and change Pos Y to 75.
- Change Text color into white and font size to 24.
- Add some placeholder text into Win Text.
- Adjust the alignment to center and middle.



- In the script, first create two new public text variables called countText and winText to hold references to the UI text component on our text game object.
  (The code is shown as below in this section.)
- In unity, drag Count Text and Win Text into the slot at the Player game object.
- Save the scene and play.

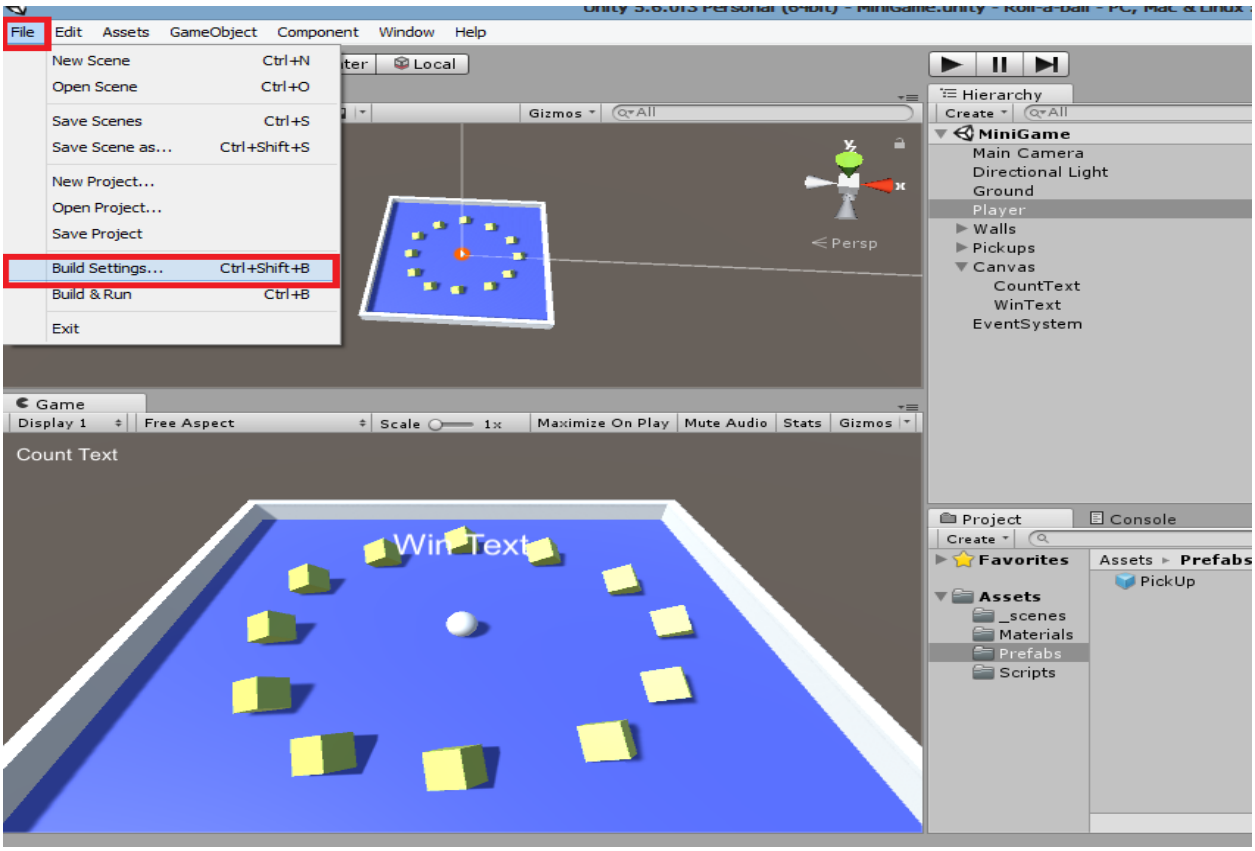

Collab
Account
Layers
Layout
Hierarchy
Create
All
MiniGame
Main Camera
Directional Light
Ground
Player
Walls
Pickups
Canvas
CountText
WinText
EventSystem
Drag them
Project
Console
Create
Favorites
Assets
_scenes
Materials
Prefabs
Scripts
Assets ▸ Prefabs
PickUp
Inspector
Player
Static
Tag
Untagged
Layer
Default
Transform
Position
Rotation
Scale
Sphere (Mesh Filter)
Mesh
Sphere
Mesh Renderer
Rigidbody
Mass
Drag
Angular Drag
Use Gravity
Is Kinematic
Interpolate
None
Collision Detection
Discrete
Constraints
Sphere Collider
Player Controller (Script)
Script
PlayerController
Speed
10
Count Text
CountText (Text)
Win Text
WinText (Text)
Default-Material
Shader
Standard
Add Component

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;

public class PlayerController : MonoBehaviour {

    public float speed;
    public Text countText;
    public Text winText;

    private Rigidbody rb;
    private int count;

    void Start ()
    {
        rb = GetComponent<Rigidbody>();
        count = 0;
        SetCountText ();
        winText.text = "";
    }

    void FixedUpdate ()
    {
        float moveHorizontal = Input.GetAxis ("Horizontal");
        float moveVertical = Input.GetAxis ("Vertical");

        Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical);

        rb.AddForce (movement * speed);
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.CompareTag ( "Pick Up"))
        {
            other.gameObject.SetActive (false);
            count = count + 1;
            SetCountText ();
        }
    }

    void SetCountText ()
    {
        countText.text = "Count: " + count.ToString ();
        if (count >= 12)
        {
            winText.text = "You Win!";
        }
    }
}
```

3.4 Building the Game

Before we build our game we should save our scene.

- To build our game we must first open the Build setting window File – Build Settings.



- We want to build a standalone application that is PC, Mac, Linux Standalone so select it.

(If we did want to change our build target we can select the desired platform from the list and click the Switch Platform button at the bottom of the window.)

- Need to add the scenes we want to build to the Build Setting window by clicking Add Open Scenes button.
- When we click Build button, this will appear a dialogue box asking us to choose a build location.

- Create a new folder inside our project called Builds and give the build name inside the Builds folder and then click Save.

When building for Windows, Unity builds .exe file and a data folder which contains all of the necessary resources.

To run the game, simply run the executable application.

Finally, we have learned how to create new game objects, how to position them in the scene, add new components to them, write our own custom behaviours for them using simple scripting in these tutorial. We've seen how to use lights, cameras, colliders, triggers, rigid bodies. We can collect and count objects. It covers a large set of basic subjects important to understanding how to use Unity.