Prerequisites:

Install visual studio 2019 or 2017 community

Unity 3d

Steps:

Download DirectX SDK from the link given in the description.

https://www.microsoft.com/en-in/download/details.aspx?id=6812

Open control panel:

If you have Windows 64 bit then uninstall these 2 files first.(Since mine is 64bit..)

- **Microsoft Visual C++ 2010 x64 Redistributable**
- **Microsoft Visual C++ 2010 x86 Redistributable**

If you have Windows 32 bit then uninstall this file only.

- **Microsoft Visual C++ 2010 x86 Redistributable**

If you don't Follow the above steps you will get an "ERROR NO:S103"

Install DirectX SDK wait till it gets installed.

How to check whether it is installed or not ?

Go to Local Disk C:

Windows folder.

Microsoft.Net

DirectX Mangaged code....

If the above folder is present then you have successfully installed DirectX SDK :)

Practical No. 1

Aim: Setup DirectX 11, Window Framework and Initialize Direct3D Device

In this practical we are just learning the window framework and initializing a Direct3D device.

Step 1:

i) Create new project, and select "Windows Forms Application", select .NET Framework as 2.0 in Visuals C#.
ii) Right Click on properties Click on open click on build Select Platform Target and Select x86.

Step 2: Click on View Code of Form 1.

Step 3:

Go to Solution Explorer, right click on project name, and select Add Reference. Click on Browse and select the given .dll files which are "Microsoft.DirectX", "Microsoft.DirectX.Direct3D", and "Microsoft.DirectX.DirectX3DX".

Right click on "References" in Solution explorer. ►Add references ►Browse to the "DirectX Managed Code" directory. ►Select these three files. ► Microsoft.DirectX.Direct3DX ► Microsoft.DirectX.Direct3D.dll ► Microsoft.DirectX.dll and add...

Step 4:

Go to Properties Section of Form, select Paint in the Event List and enter as Form1_Paint.

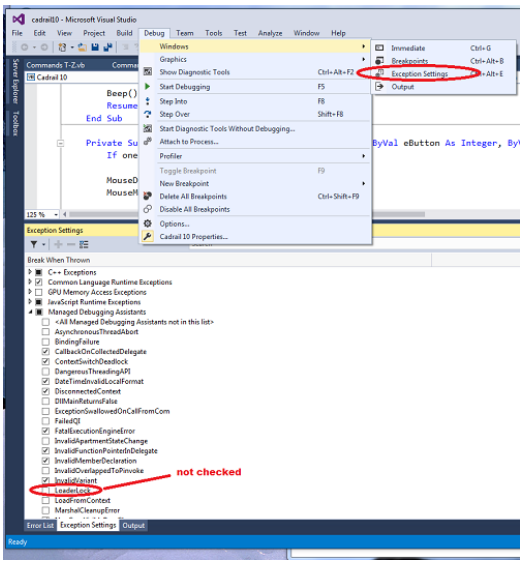
Step 5:

Edit the Form's C# code file. Namespace must be as same as your project name.

```
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 GP_P1
{
public partial class Form1 : Form
   {
      Microsoft.DirectX.Direct3D.Device device;
public Form1()
      {
InitializeComponent();
InitDevice();
```

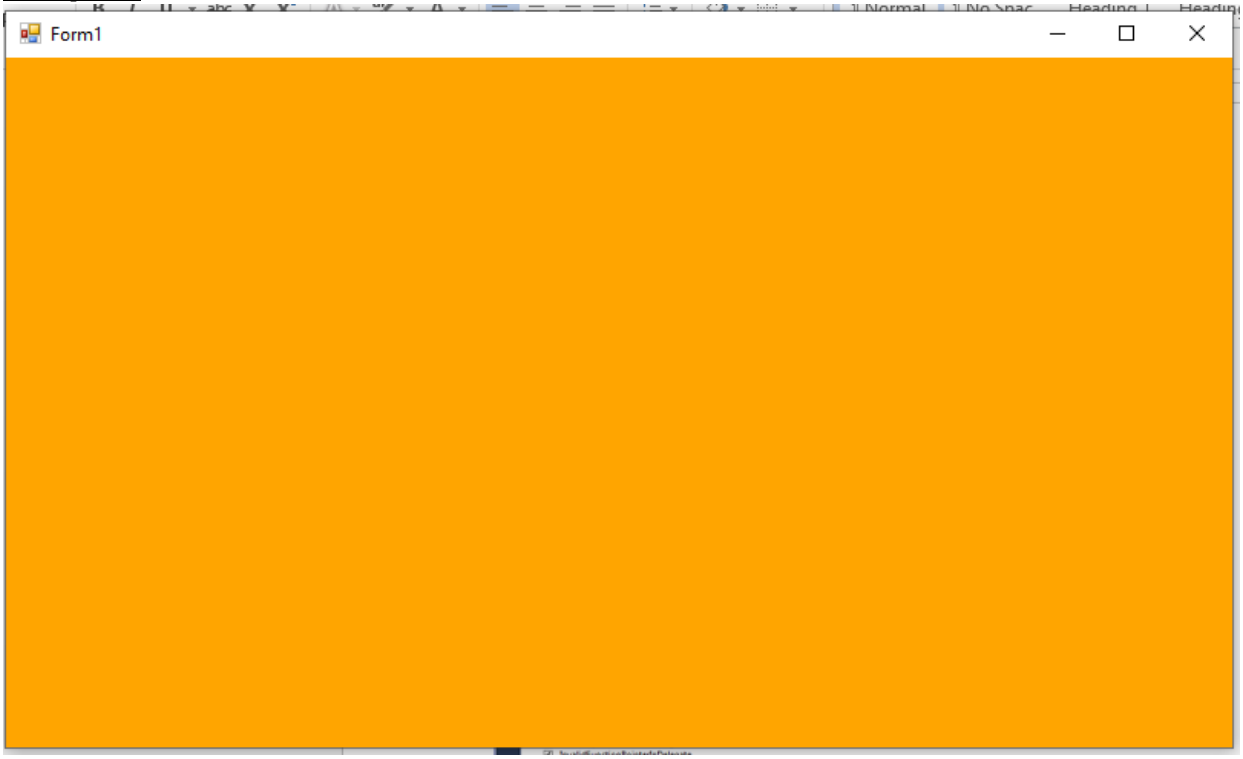
```
        }

public void InitDevice()
        {
PresentParameterspp = 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.Orange, 0, 1);
device.Present();
        }

private void Form1_Paint(object sender, PaintEventArgs e)
        {
Render();
        }
    }
}
```

Step 6: Click on Start. And here is the output. We have initialized 3D Device.



NOTE : uncheck loaderlock or else it generates exception

Output:



Practical No. 2:

Aim: Draw a triangle using Direct3D 11

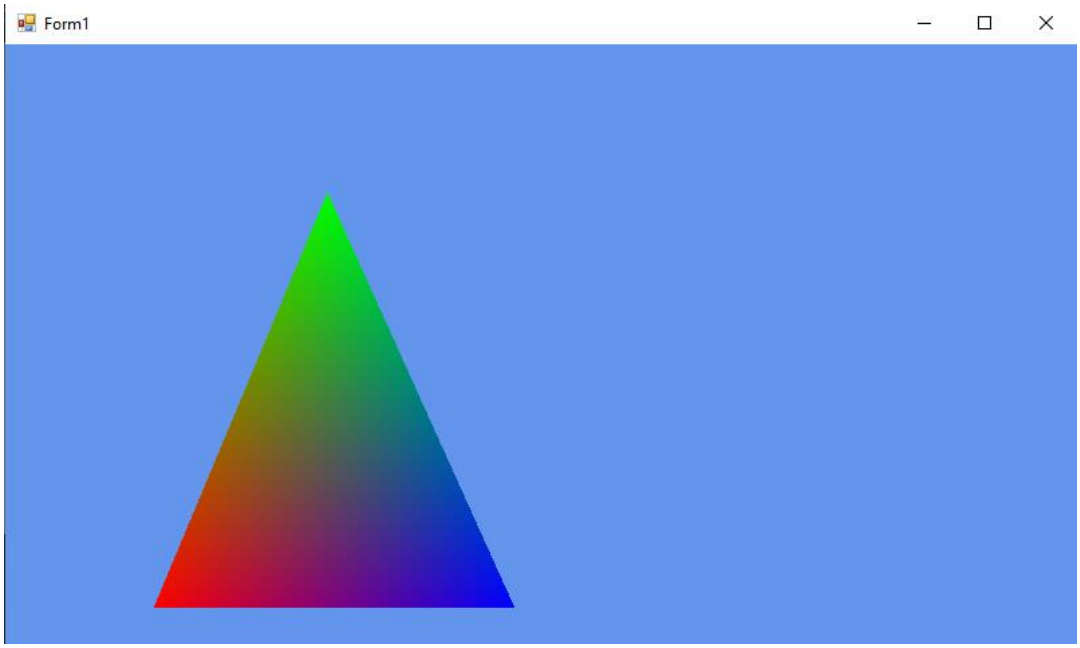
Solution:

```
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 GPRACT2
{
public partial class Form1 : Form
   {
      Microsoft.DirectX.Direct3D.Device device;
public Form1()
      {
InitializeComponent();
InitDevice();
      }
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()
      {
CustomVertex.TransformedColored[] vertexes = new
CustomVertex.TransformedColored[3];

vertexes[0].Position = new Vector4(240, 110, 0, 1.0f);//first point
vertexes[0].Color = System.Drawing.Color.FromArgb(0, 255, 0).ToArgb();

vertexes[1].Position = new Vector4(380, 420, 0, 1.0f);//second point
vertexes[1].Color = System.Drawing.Color.FromArgb(0, 0, 255).ToArgb();

vertexes[2].Position = new Vector4(110, 420, 0, 1.0f);//third point
vertexes[2].Color = System.Drawing.Color.FromArgb(255, 0, 0).ToArgb();

device.Clear(ClearFlags.Target, Color.CornflowerBlue, 1.0f, 0);
```

```
device.BeginScene();
device.VertexFormat = CustomVertex.TransformedColored.Format;
device.DrawUserPrimitives(PrimitiveType.TriangleList, 1, vertexes);
device.EndScene();
device.Present();
      }
private void Form1_Load(object sender, EventArgs e) { }

private void Form1_Paint(object sender, PaintEventArgs e)
      {
Render();
      }
   }
}
```

Note: Go to Properties Section of Form, select Paint in the Event List and enter as Form1_Paint.

Output:



Practical No. 3

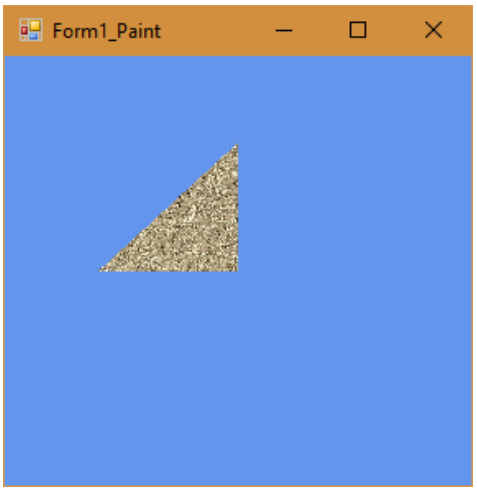
Aim: Texture the triangle using Direct3D 11

Solution:

```
using System;
usingSystem.Collections.Generic;
usingSystem.ComponentModel;
usingSystem.Data;
usingSystem.Drawing;
usingSystem.Text;
usingSystem.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace GPPRACT3
{
public partial class Form1 : Form
   {
private Microsoft.DirectX.Direct3D.Device device;
privateCustomVertex.PositionTextured[] vertex = new
CustomVertex.PositionTextured[3];
private Texture texture;
public Form1()
      {
InitializeComponent();
InitDevice();
      }
private void InitDevice()
      {
PresentParameterspp = 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, 0, 0), 0,
0);
vertex[1] = new CustomVertex.PositionTextured(new Vector3(5, 0, 0), 0,
1);
vertex[2] = new CustomVertex.PositionTextured(new Vector3(0, 5, 0),-1,
1);
texture=new Texture (device, new Bitmap
("E:\\TYCS\\images\\img1.jpg"), 0,Pool.Managed );
      }

private void Form1_Load(Object sender, EventArgs e)
      { }

private void Form1_Paint(Object sender, PaintEventArgs e)
      {
device.Clear(ClearFlags.Target, Color.CornflowerBlue, 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:



NOTE: don’t forget to add image into your solution with proper path specified

Practical No. 4

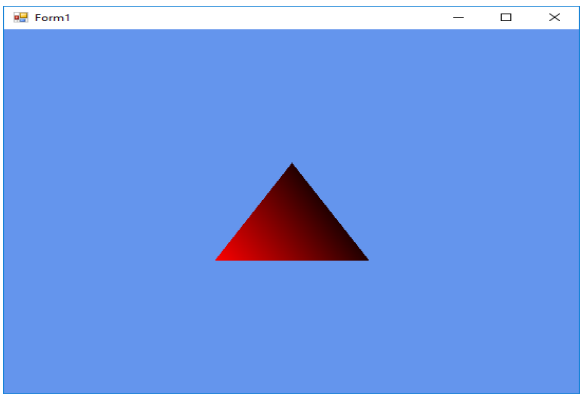
Aim: Programmable Diffuse Lightning using Direct3D 11

Solution:

```
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 GPPRACT4
{
public partial class Form1 : Form
   {

private Microsoft.DirectX.Direct3D.Device device;
privateCustomVertex.PositionNormalColored[] vertex = new
CustomVertex.PositionNormalColored[3];
public Form1()
      {
InitializeComponent();
InitDevice();
      }

public void InitDevice()
      {
PresentParameterspp = 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;
      }

public void Render()
      {
      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();
      }
private void Form1_Load(object sender, EventArgs e)

private void Form1_Paint(object sender, PaintEventArgs e)
      {
Render();
      }
   }
}
```

Output:



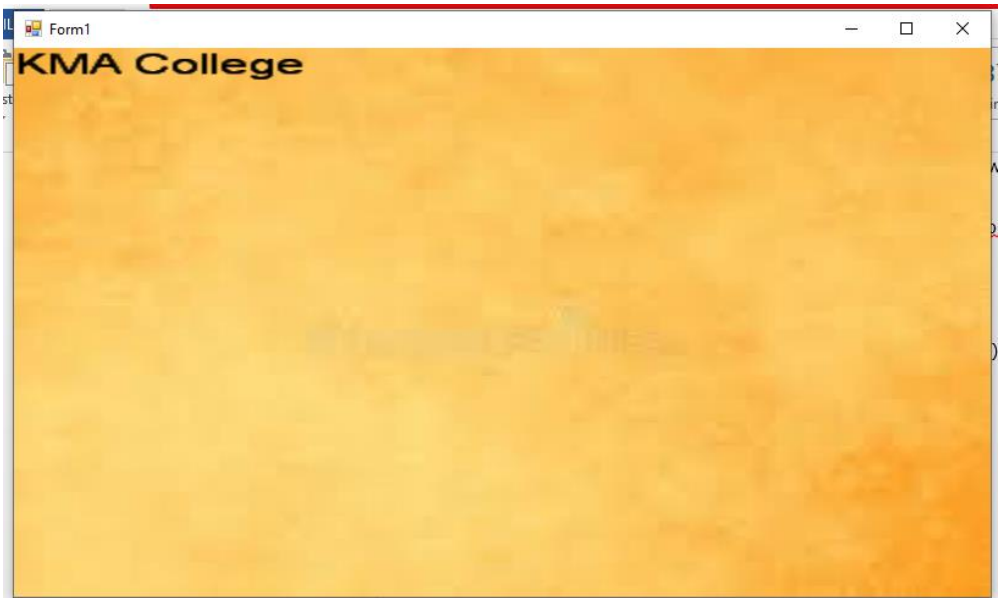
Practical No. 5

Aim: Loading models into DirectX 11 and rendering

```
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 GPPRACT5
{
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 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,"E:\\TYCS\\images\\img1.jpg",400,
      400, 1, 0, Format.A8B8G8R8, Pool.Managed, Filter.Point, Filter.Point,
      Color.Transparent.ToArgb());
 }

private void InitDevice()
 {
      PresentParameterspp = 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, "KMA College", new Point(0, 0), Color.Black);
        s.End();
 }
device.EndScene();
device.Present();
 }
private void Form1_Paint(object sender, PaintEventArgs e)
 {
Render();
 }
 }
}
```

Output:



Practical no.6

Aim :- Loading models into DirectX 11 and rendering.

ModelLoading.vcxproj :-

```
<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="15.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <VCProjectVersion>15.0</VCProjectVersion>
    <ProjectGuid>{8E309F6A-53B8-418E-AA59-F38D18459F8A}</ProjectGuid>
    <Keyword>Win32Proj</Keyword>
    <RootNamespace>My6ModelLoading</RootNamespace>
    <WindowsTargetPlatformVersion>10.0.17134.0</WindowsTargetPlatformVersion>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
Label="Configuration">
    <ConfigurationType>Application</ConfigurationType>
    <UseDebugLibraries>true</UseDebugLibraries>
    <PlatformToolset>v141</PlatformToolset>
    <CharacterSet>Unicode</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
Label="Configuration">
    <ConfigurationType>Application</ConfigurationType>
    <UseDebugLibraries>false</UseDebugLibraries>
    <PlatformToolset>v141</PlatformToolset>
    <WholeProgramOptimization>true</WholeProgramOptimization>
    <CharacterSet>Unicode</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
Label="Configuration">
    <ConfigurationType>Application</ConfigurationType>
    <UseDebugLibraries>true</UseDebugLibraries>
```

```
    <PlatformToolset>v141</PlatformToolset>
    <CharacterSet>Unicode</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
Label="Configuration">
    <ConfigurationType>Application</ConfigurationType>
    <UseDebugLibraries>false</UseDebugLibraries>
    <PlatformToolset>v141</PlatformToolset>
    <WholeProgramOptimization>true</WholeProgramOptimization>
    <CharacterSet>Unicode</CharacterSet>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
  <ImportGroup Label="Shared">
  </ImportGroup>
  <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')"
Label="LocalAppDataPlatform" />
  </ImportGroup>
  <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')"
Label="LocalAppDataPlatform" />
  </ImportGroup>
  <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')"
Label="LocalAppDataPlatform" />
  </ImportGroup>
  <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')"
Label="LocalAppDataPlatform" />
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <LinkIncremental>true</LinkIncremental>

<IntDir>$(SolutionDir)\temp\$(ProjectName)\$(Configuration)\$(Platform)\</IntDir>
    <OutDir>$(SolutionDir)\bin\$(Configuration)\$(Platform)\</OutDir>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <LinkIncremental>true</LinkIncremental>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">
    <LinkIncremental>false</LinkIncremental>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <LinkIncremental>false</LinkIncremental>

<IntDir>$(SolutionDir)\temp\$(ProjectName)\$(Configuration)\$(Platform)\</IntDir>
    <OutDir>$(SolutionDir)\bin\$(Configuration)\$(Platform)\</OutDir>
  </PropertyGroup>
```

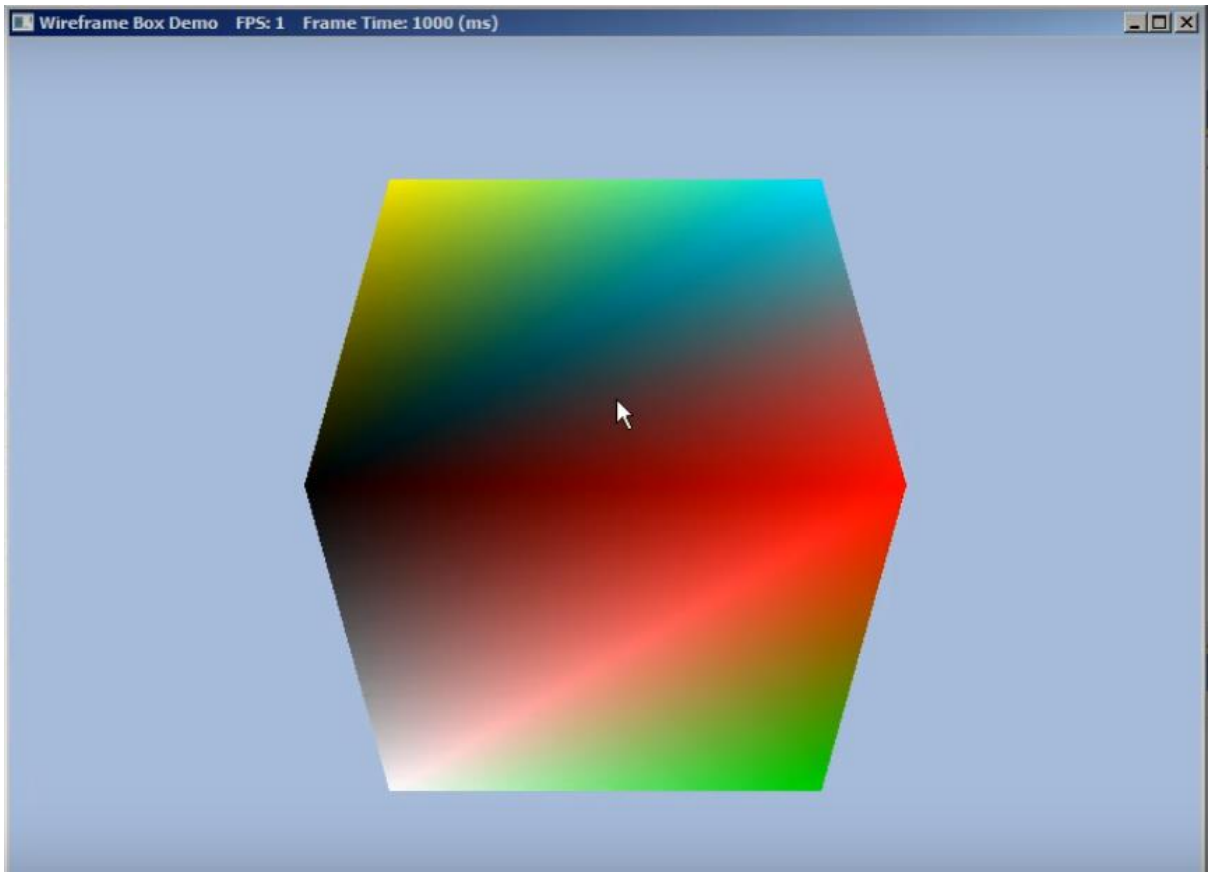
```
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <ClCompile>
      <WarningLevel>Level3</WarningLevel>
      <Optimization>Disabled</Optimization>
      <SDLCheck>true</SDLCheck>

<PreprocessorDefinitions>_DEBUG;_WINDOWS;%(PreprocessorDefinitions)</PreprocessorD
efinitions>
      <ConformanceMode>true</ConformanceMode>
    </ClCompile>
    <Link>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <SubSystem>Windows</SubSystem>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <ClCompile>
      <WarningLevel>Level3</WarningLevel>
      <Optimization>Disabled</Optimization>
      <SDLCheck>true</SDLCheck>

<PreprocessorDefinitions>WIN32;_DEBUG;_WINDOWS;%(PreprocessorDefinitions)</Preproc
essorDefinitions>
      <ConformanceMode>true</ConformanceMode>
    </ClCompile>
    <Link>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <SubSystem>Windows</SubSystem>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup
Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">
    <ClCompile>
      <WarningLevel>Level3</WarningLevel>
      <Optimization>MaxSpeed</Optimization>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <IntrinsicFunctions>true</IntrinsicFunctions>
      <SDLCheck>true</SDLCheck>

<PreprocessorDefinitions>WIN32;NDEBUG;_WINDOWS;%(PreprocessorDefinitions)</Preproc
essorDefinitions>
      <ConformanceMode>true</ConformanceMode>
    </ClCompile>
    <Link>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <OptimizeReferences>true</OptimizeReferences>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <SubSystem>Windows</SubSystem>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <ClCompile>
      <WarningLevel>Level3</WarningLevel>
      <Optimization>MaxSpeed</Optimization>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <IntrinsicFunctions>true</IntrinsicFunctions>
      <SDLCheck>true</SDLCheck>
```

```
<PreprocessorDefinitions>NDEBUG;_WINDOWS;%(PreprocessorDefinitions)</PreprocessorD
efinitions>
      <ConformanceMode>true</ConformanceMode>
    </ClCompile>
    <Link>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <OptimizeReferences>true</OptimizeReferences>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <SubSystem>Windows</SubSystem>
    </Link>
  </ItemDefinitionGroup>
  <ItemGroup>
  </ItemGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
  <ImportGroup Label="ExtensionTargets">
  </ImportGroup>
</Project>
```

ModelLoading.vcxproj.filters :-

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <ItemGroup>
    <Filter Include="Source Files">
      <UniqueIdentifier>{4FC737F1-C7A5-4376-A066-2A32D752A2FF}</UniqueIdentifier>
      <Extensions>cpp;c;cc;cxx;def;odl;idl;hpj;bat;asm;asmx</Extensions>
    </Filter>
    <Filter Include="Header Files">
      <UniqueIdentifier>{93995380-89BD-4b04-88EB-625FBE52EBFB}</UniqueIdentifier>
      <Extensions>h;hh;hpp;hxx;hm;inl;inc;xsd</Extensions>
    </Filter>
    <Filter Include="Resource Files">
      <UniqueIdentifier>{67DA6AB6-F800-4c08-8B7A-83BB121AAD01}</UniqueIdentifier>

<Extensions>rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe;resx;tiff;tif;png;
wav;mfcribbon-ms</Extensions>
    </Filter>
  </ItemGroup>
</Project>
```

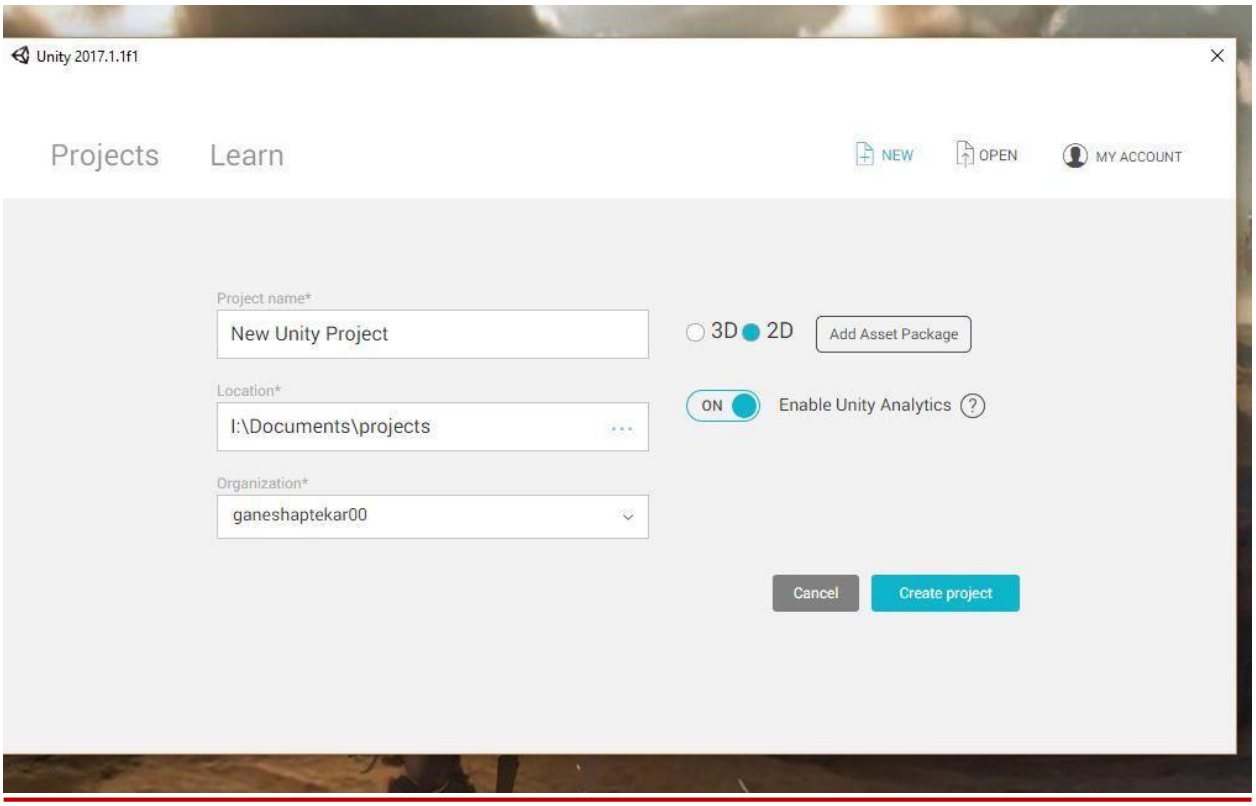
OUTPUT :-



Practical no . 7

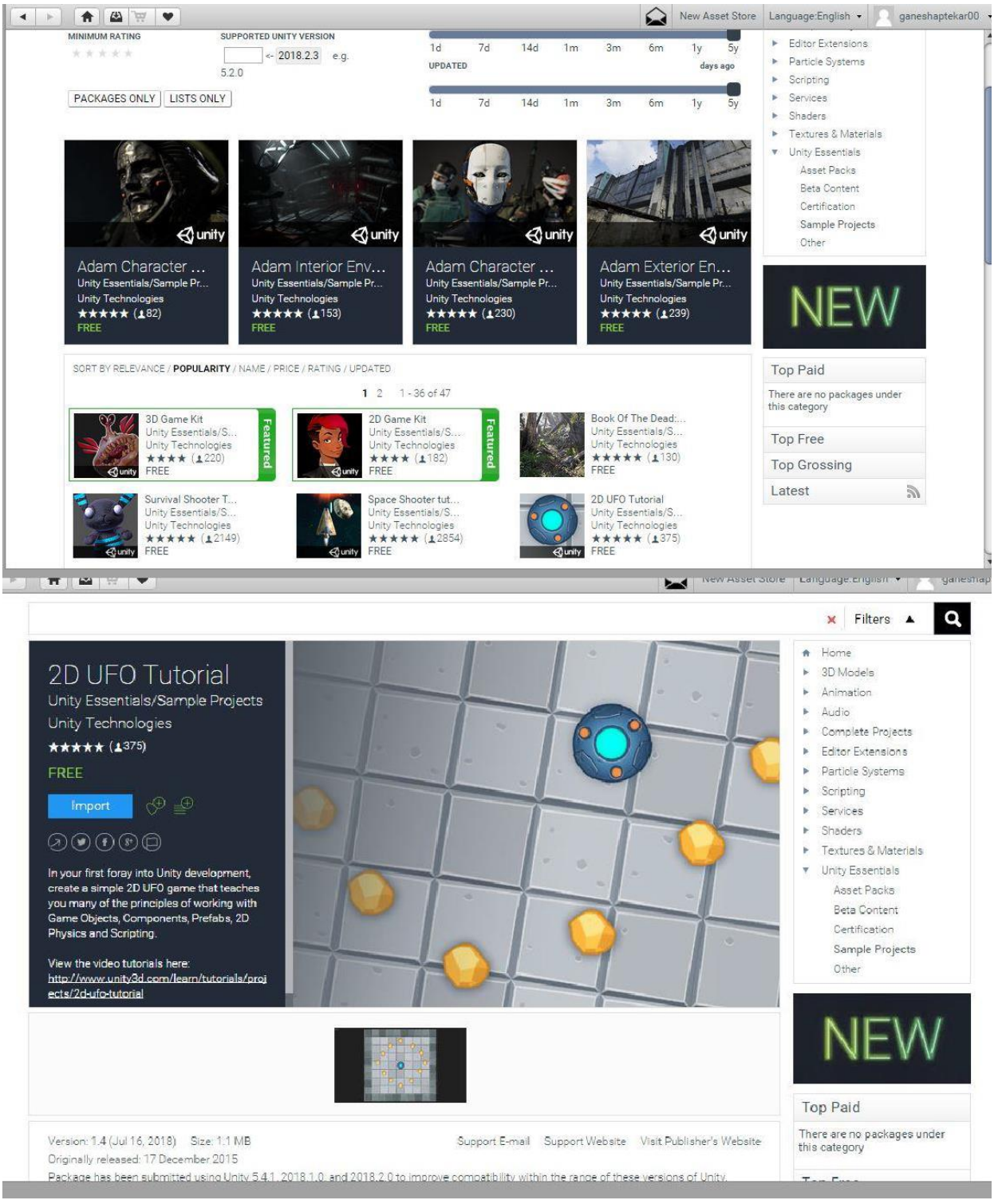
Aim: Using a unity3d software and making a 2d ufo game.

Step 1: **Open unity software and create a new project.**



Choose the 2d option, create the project.

- Go to window button and open asset store.
- Click on Unity essential, then go to sample projects
- You see 2d ufo tutorial package then open it.
- Import the package in software
- Save your scene. (Crtl+s)

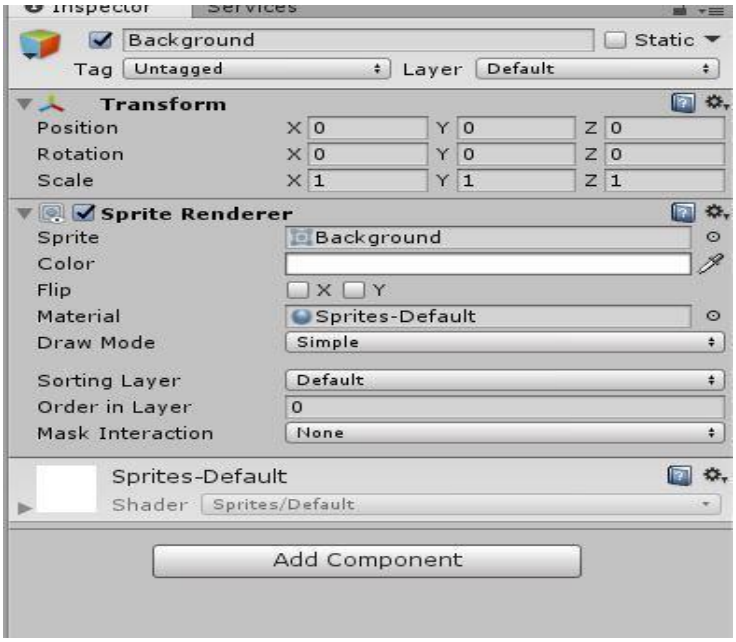


Step 2: **Setting up the field**

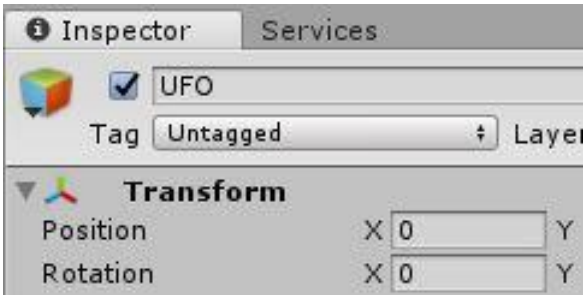
- Then click on sprites.
- This is your 2d ufo sprites.
- Drag background to hierarchy.

Then you see right corners inspector button click on it.

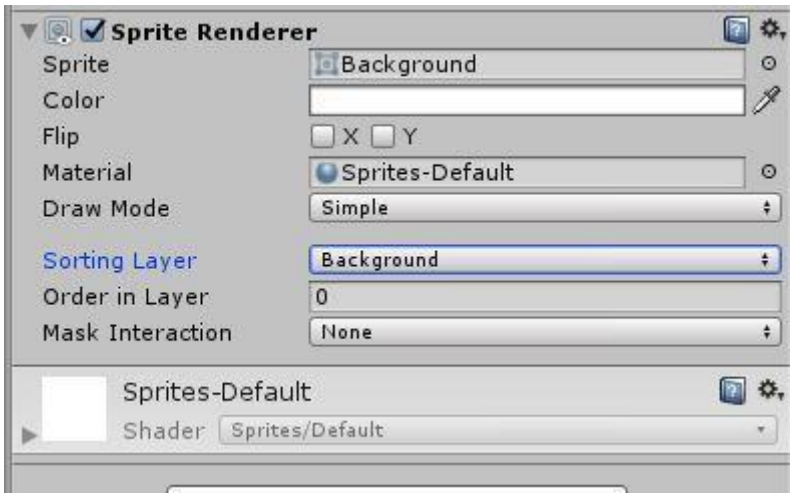
- Sprite Renderer box is created.



- Same as it is ufo. Drag to hierarchy.
- You can change the name.



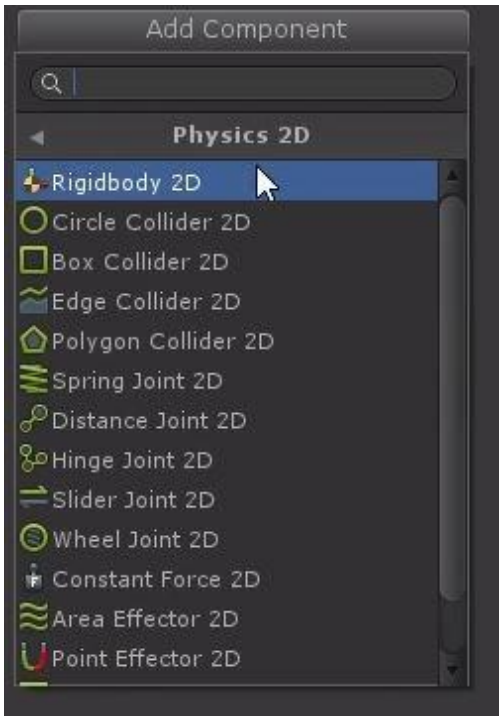
- Then you see sorting layer in sprite renderer, then set that layer background to background and ufo to player.



Step 3: **Controlling the player.**

- Clcick on ufo and add component

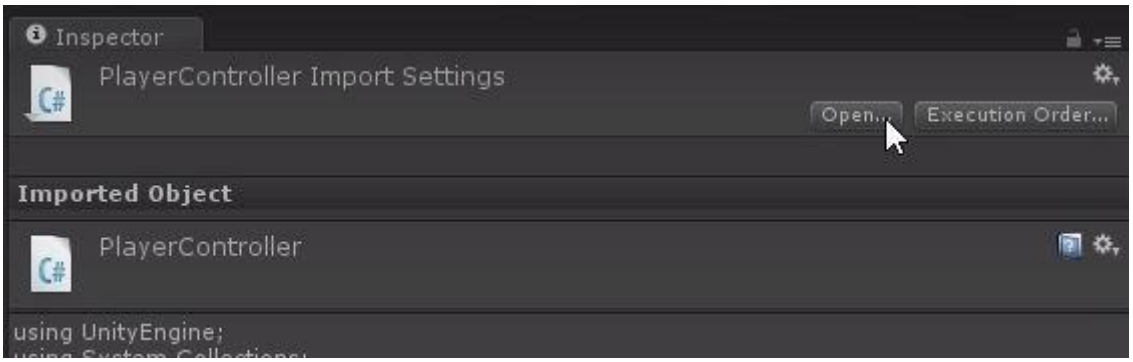
- Click on 2d physics
- Click on rigidbody



- Then you see in inspector rigidbody box created
- We need to create a script for moving our ufo

- Click on add component and create

- Name the script.
- That script darg into your asset scripts
- Open the



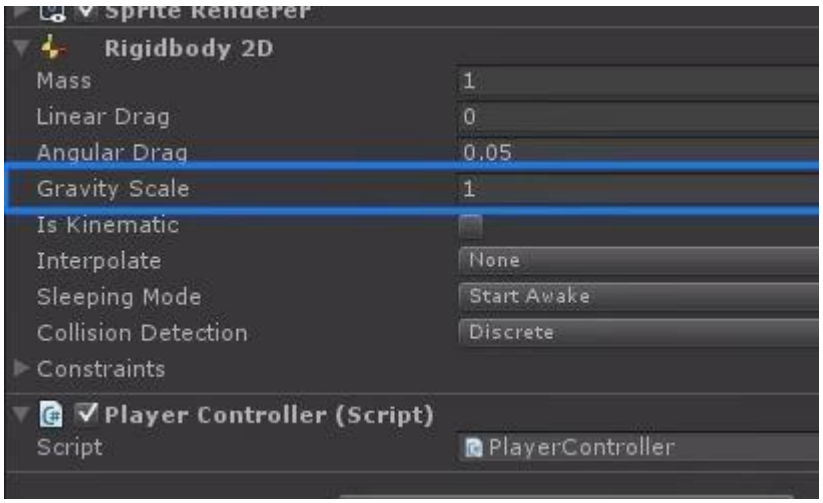
script

- Write that script into it.

```
using UnityEngine;
using System.Collections;

public class PlayerController : MonoBehaviour {

    private Rigidbody2D rb2d;

    void Start()
    {
        rb2d = GetComponent<Rigidbody2D> ();
    }

    void FixedUpdate()
    {
        float moveHorizontal = Input.GetAxis ("Horizontal");
        float moveVertical = Input.GetAxis ("Vertical");
        Vector2 movement = new Vector2 (moveHorizontal, moveVertical);
        rb2d.AddForce (movement);
    }
}
```

Then test your game..

- Then your ufo fall down because of gravity scale
- Go to rigidbody2d and gravity scale 1 to change 0.



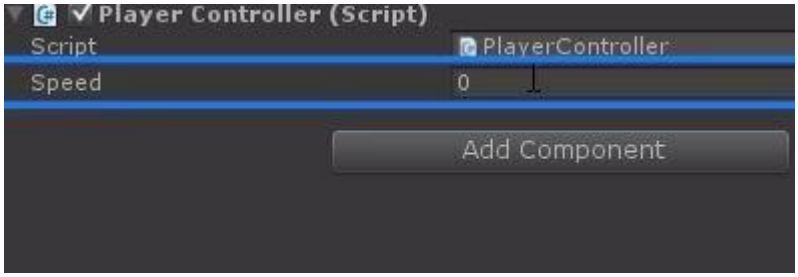
- Then your ufo speed is so slow then go to your script and add that 2 lines
- “ Public float speed ; ”

```
3
4 public class PlayerController : MonoBehaviour {
5
6     public float speed;
7
8     private Rigidbody2D rb2d;
9
10    void Start()
11    {
```

- " (movement *speed) ;"

```
Vector2 movement = new Vector2 (mo
rb2d.AddForce (movement * speed);
```

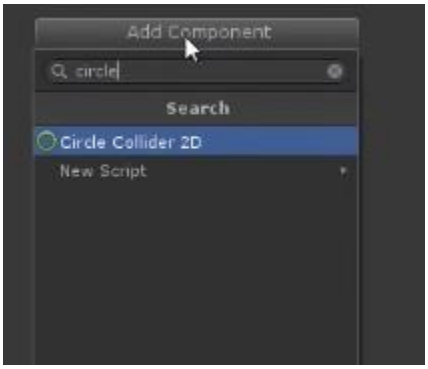
- Go to inspector and see your rigidbody 2d is updated with speed.



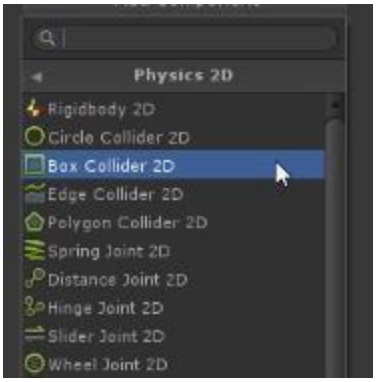
- Set your speed

Step 4 : **Adding Collision.**

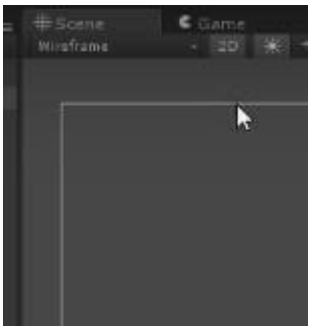
- Go to add component and type circle collider this is for ufo.



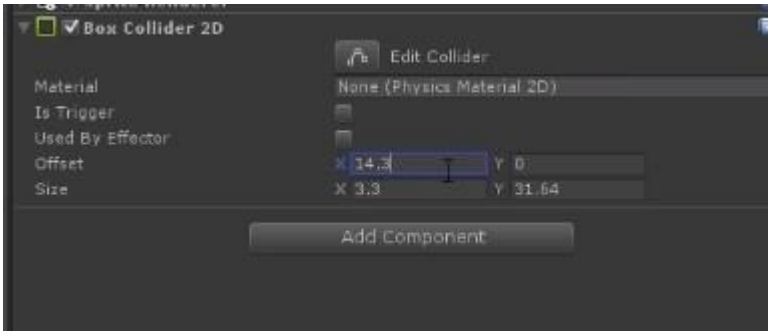
- This is the circle collider.
- This the Radius 2.15 for ufo collider.
  Then we use as same for our background.
- Go to add componet and go to 2d physics and choose box ollider.



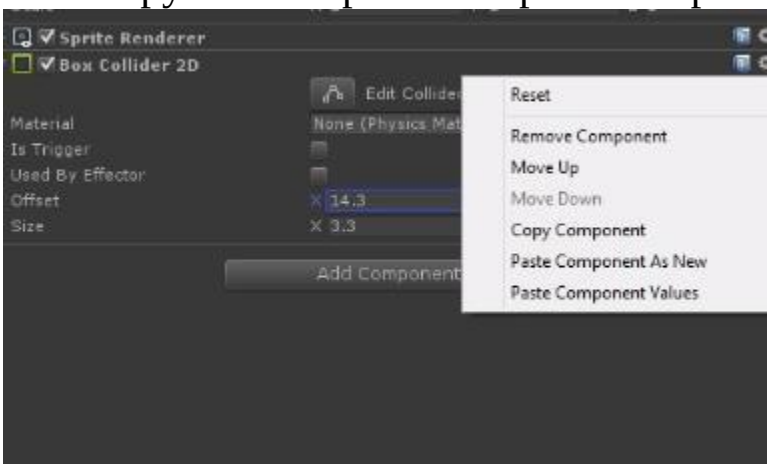
cThen go to your scene and 1inch down you see shaded button click on this and change to wireframe.



Then set you x axis offset size 14.3 and y axis is zero.
Box collider size for x axis 3.3 and y axis is 31.64.



- Then copy that componet and paste componet.



- See diagram above…..
- And same as it y axis -14.3 for offset and x is zero
- Same as size collider y axis is 3.3 and x axis is 31.64……..

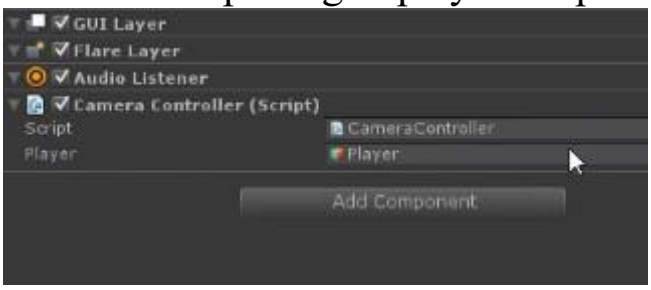
Step 5: **Follwing the player with camera…**

- That is a simple for camera
- We need to create script for camera control…

```
using UnityEngine;
using System.Collections;

public class CameraController : MonoBehaviour {

    public GameObject player;


    private Vector3 offset;

    // Use this for initialization
    void Start ()
    {
        offset = transform.position - player.transform.position;
    }

    // Update is called once per frame
    void LateUpdate ()
    {
        transform.position = player.transform.position + offset;
    }
}
```

That is our script..

And that script drag to player script…



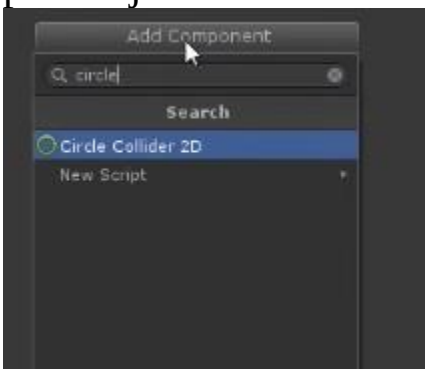
Step 6: **Creating a collectables objects.**

// Then you see.. you drag a ufo to hierarchy same as it go to sprites an take pick ups object and drag it….

Click on pick up and see inspector click on sprite render set sorting layer to pick ups

Then deactivate player object….

- Then go to add component and add again circle collider box for our pick objects.



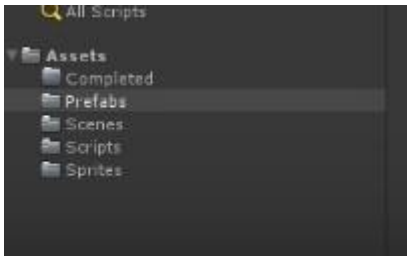
Set radious to 0.94.

- We need to raotate our object or animate, then create a new scripts

```
using UnityEngine;
using System.Collections;

public class Rotator : MonoBehaviour {


    void Update ()
    {
        transform.Rotate (new Vector3 (0, 0, 45) * Time.deltaTime);
    }
}
```

This is our script

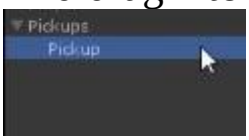
Then pickup object in hierarchy to drag



into see this..

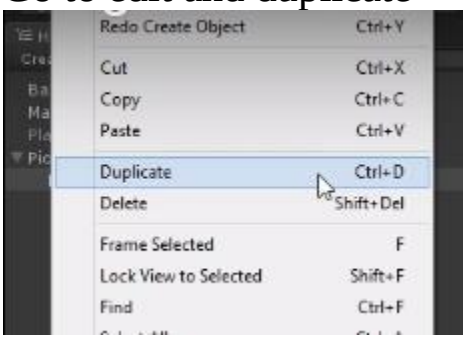
- Then we need to create a game object
- Go to game object and create empty click on it..
- Then our new object is created ..

And drag into 1st pickup in new game object..



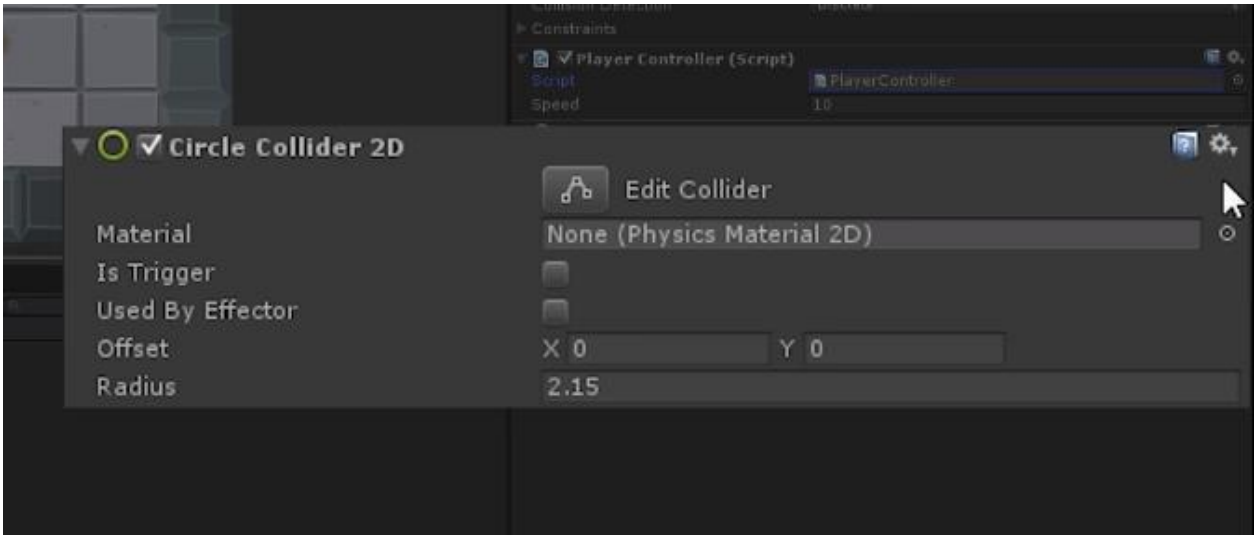
see this…

- Then create duplicate our object
- Go to edit and duplicate

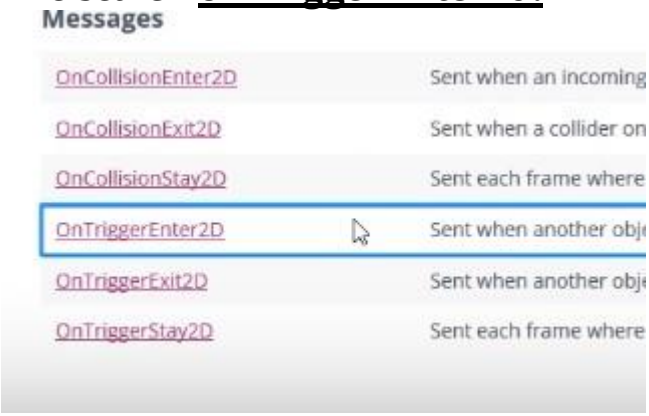


Like this….
Set your objects in game scene your own mind……

Step 7: **picking a collectables object..**
See we create pick ups object then we need change our script in player module.
Then go inspector see circle collider box the right corner you see setting button and book manual. Click on book manua



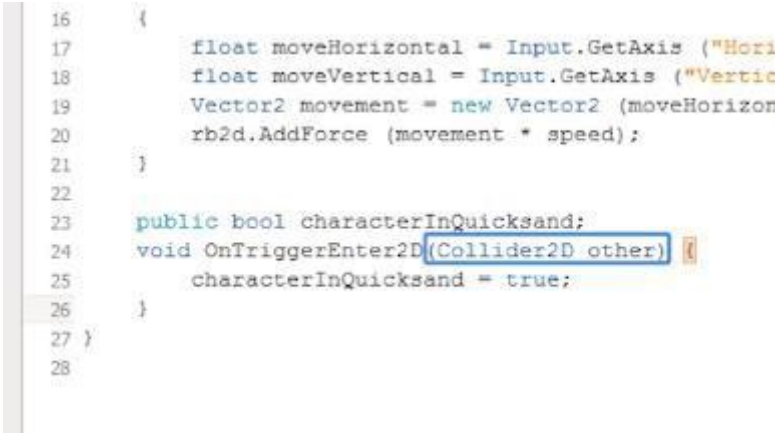
The search **onTriggerEnter2d.**



And click on it..
You see description of these..
Copy the code and paste our ufo script….

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    public bool characterInQuicksand;
    void OnTriggerEnter2D(Collider2D other) {
        characterInQuicksand = true;
    }
}
```

Below the rb2d.AddForce



```
16      {
17          float moveHorizontal = Input.GetAxis ("Hori
18          float moveVertical = Input.GetAxis ("Vertic
19          Vector2 movement = new Vector2 (moveHorizon
20          rb2d.AddForce (movement * speed);
21      }
22
23      public bool characterInQuicksand;
24      void OnTriggerEnter2D(Collider2D other) {
25          characterInQuicksand = true;
26      }
27  }
28
```

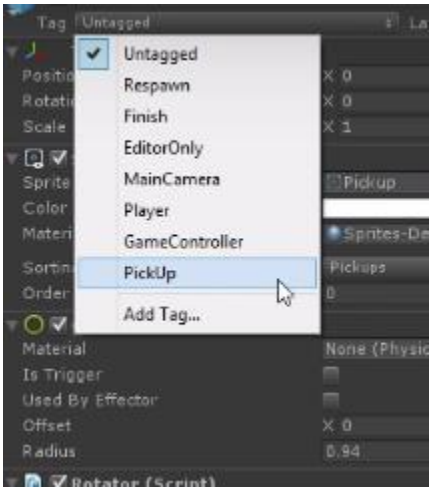
Then delete this 2 lines
1st public bool
2nd characterInQuicksand.

```
}

void OnTriggerEnter2D(Collider2D other) {
    |
}
```

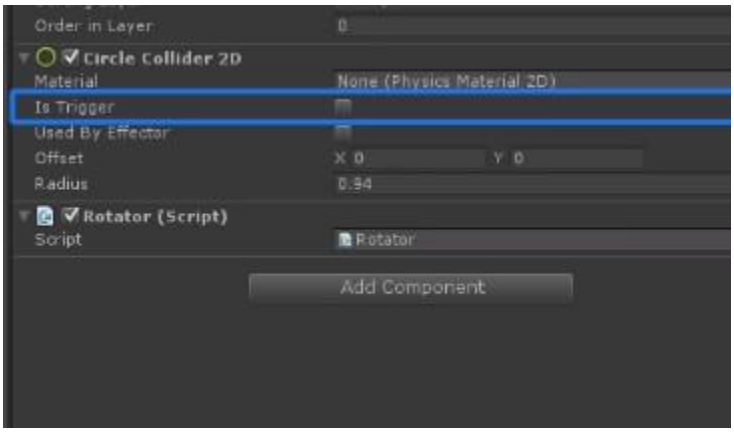
The type that code in script..

```
void OnTriggerEnter2D(Collider2D other)
{
    if (other.gameObject.CompareTag ("PickUp"))
    {
        other.gameObject.SetActive (false);
    }
}
```

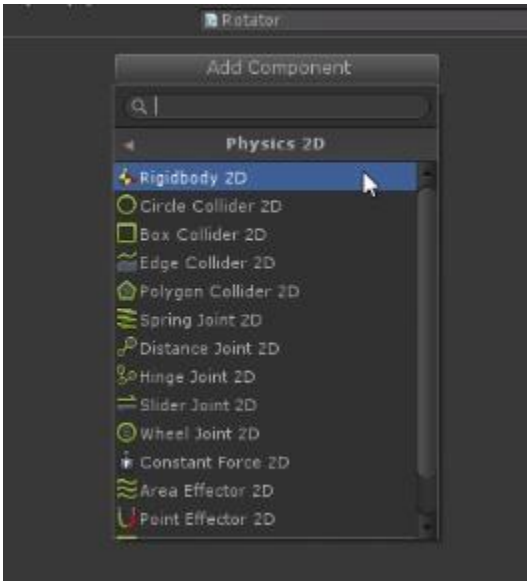
Then selcet untaggeg in inspector and change into pickups.



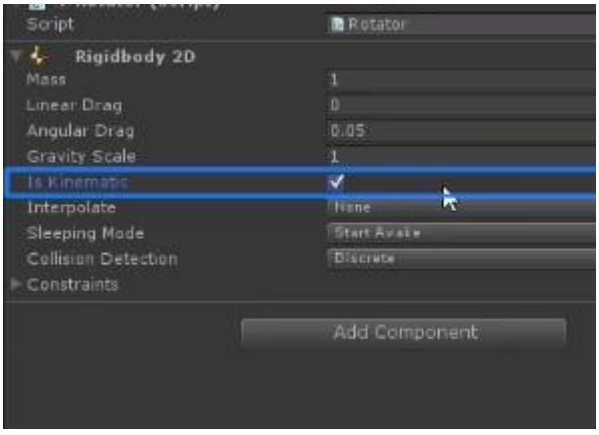
Then go to circle collider body and activated is trigger.



Add compone rigiddbody 2d..



And activated kinematic field on



Step 8: **Collecting object count and Displying score….**

- Go to player script again and add the code.
- “ Private in count ;”

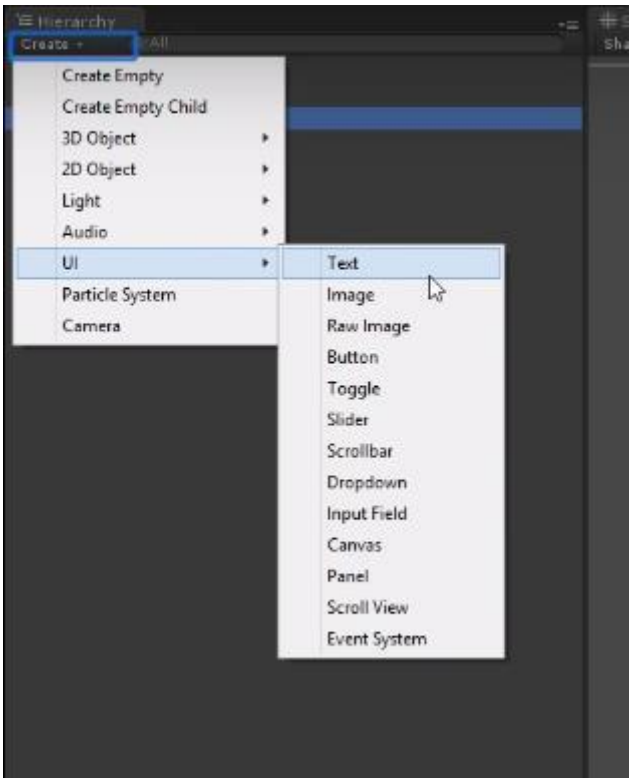
```
5
6       public float speed;
7
8       private Rigidbody2D rb2d;
9       private int count;
10
11      void Start()
12      {
{
        rb2d = GetComp
        count = 0;
    }
```

Add “ count =0 ; ”

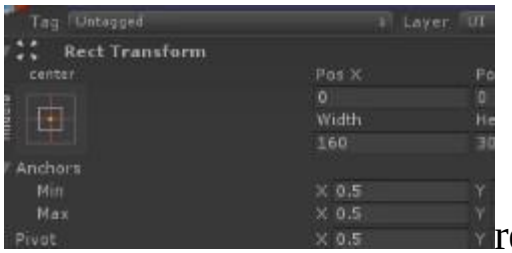
```
29          {
30              other.gameObject.SetActive (false);
31              count = count + 1;
32          }
33      }
34  }
```

Count = count+1;

- Create gaming ui object



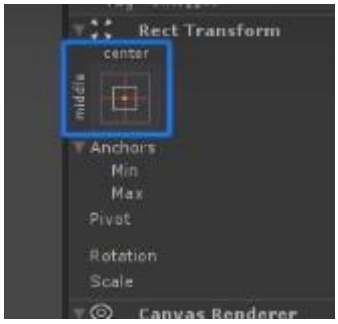
Like this…



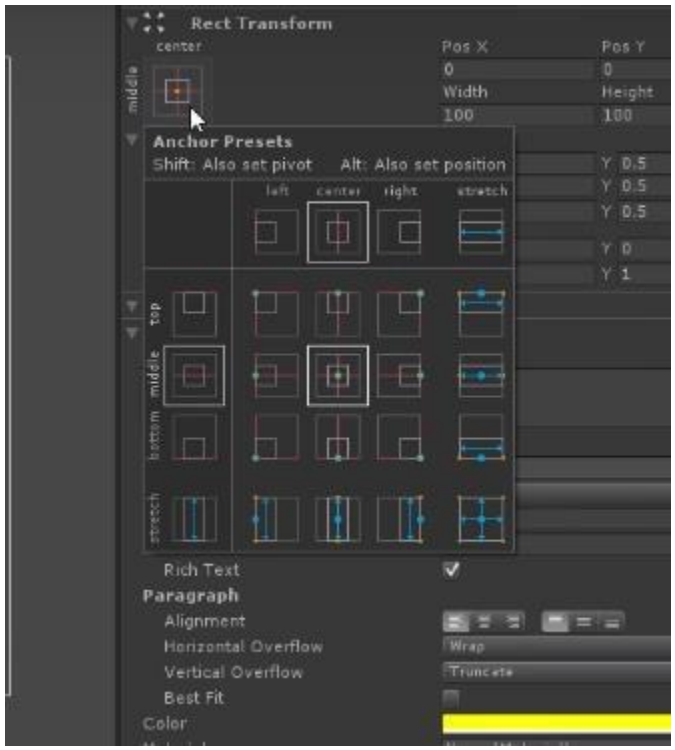
rename th ui object….

Press the f button then you see new text…then go to setting and reset coordinate

Then click on anchor

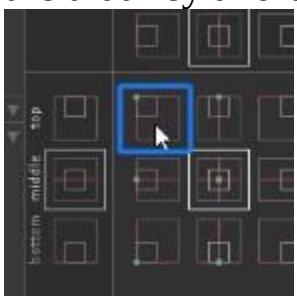


See this below.

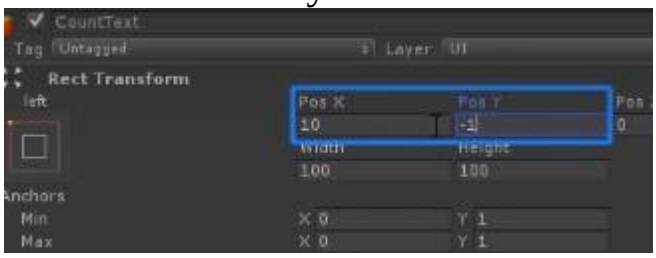


And shift+ alt click..

Hold the that key and click to that squre



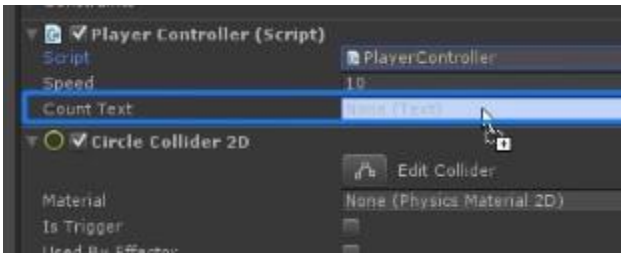
Set x axis 10 and y axis -10..



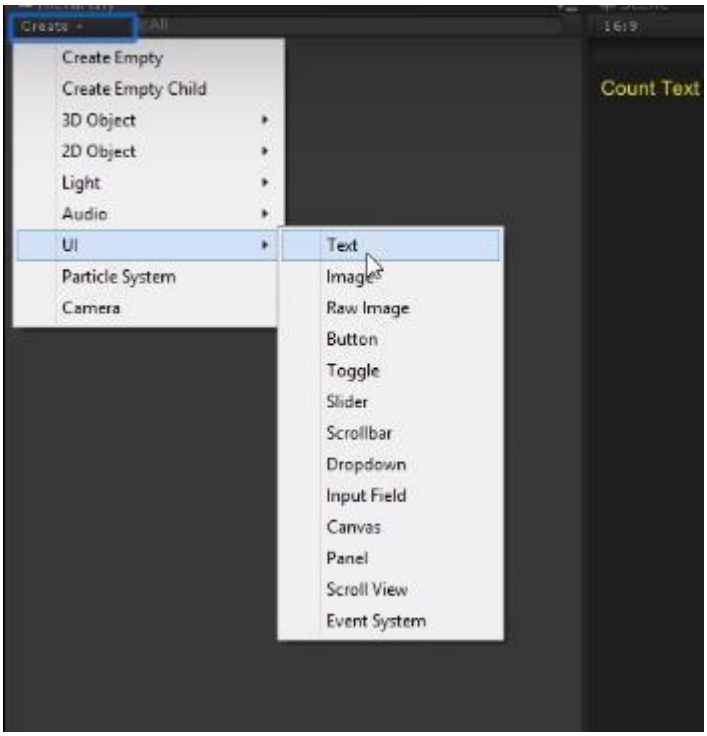
Add this code for ufo… for count text….

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

    public float speed;
    public Text countText;

        rb2d = GetComponent<Rigidbody2
        count = 0;
        SetCountText ();
    }

            count = count + 1;
            SetCountText ();
        }
    }

    void SetCountText()
    {
```

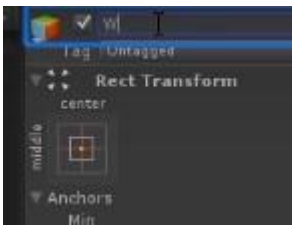
Last statement is void setcountext() close there {} parameters
And add "" **countText.text = "Count"+ count.Tostring();**"
In void setcount stament…
And go to hierchay ui object count text drag to inspectorcontroller script darg into count text



see this I will show…Then add text for win!!!! Text..



Rename it…



And set that text you want and change y axis also x axis your own mind….

Change the color and font in your choice

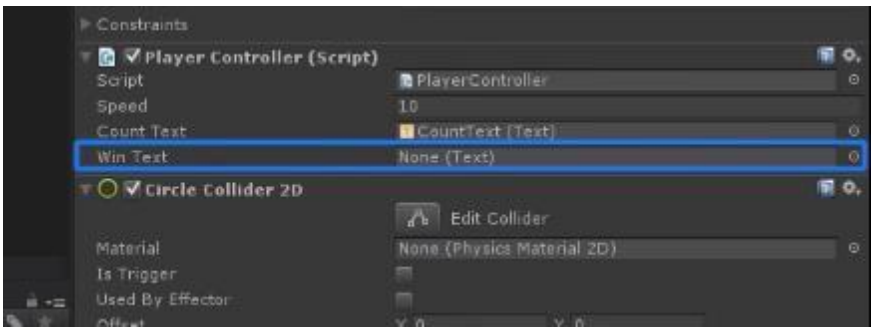
```
public Text countText;
public Text winText;

        rb2d = GetComponent<Rigidbody2D> ();
        count = 0;
        winText.text = "";
        SetCountText ();
    }

    if (count >= 12)
    {
        winText.text = "You win!";
```

Add this code in ufo script

1st create a “public text winText;”

2nd “ winText.text=””; ”

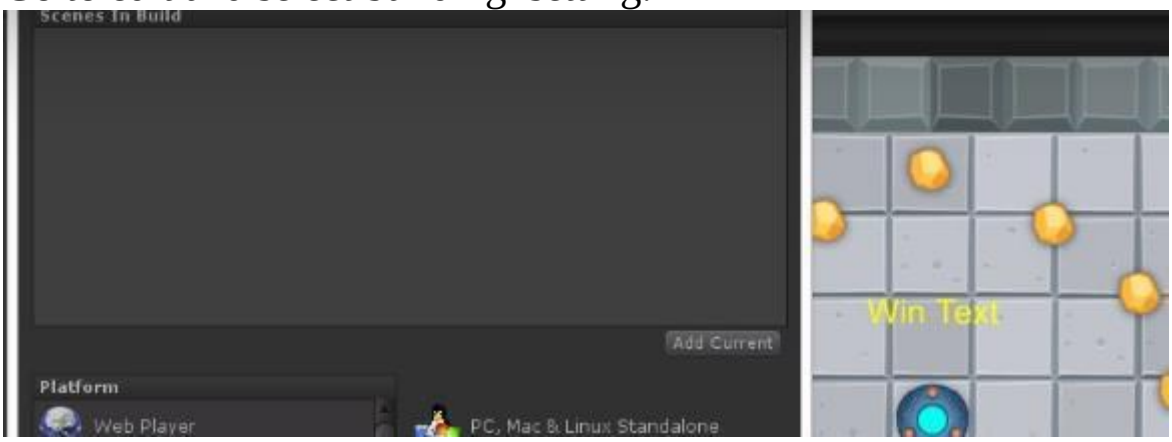
3rd “winText.text=”you win!!!!!”; “



That hierchy wint text drag into inseptor win text on controller

Step 9: **Building a game**

Go to edit and select building setting.



And click add current

And build this game

Save your game in my computer

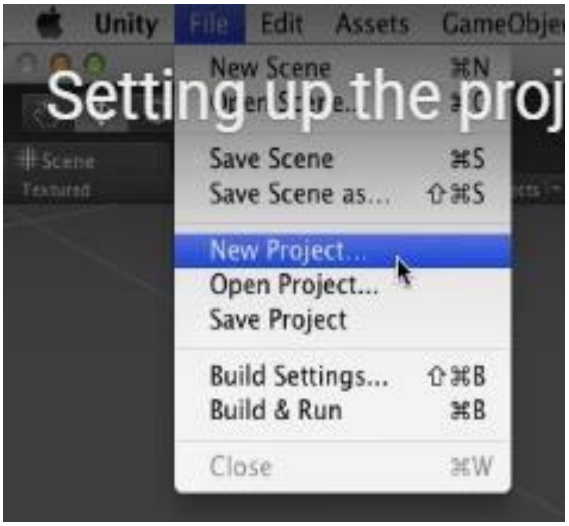
Enjoy i

Practical no. 8

Aim: Using a unity3d software and creating space shooter.

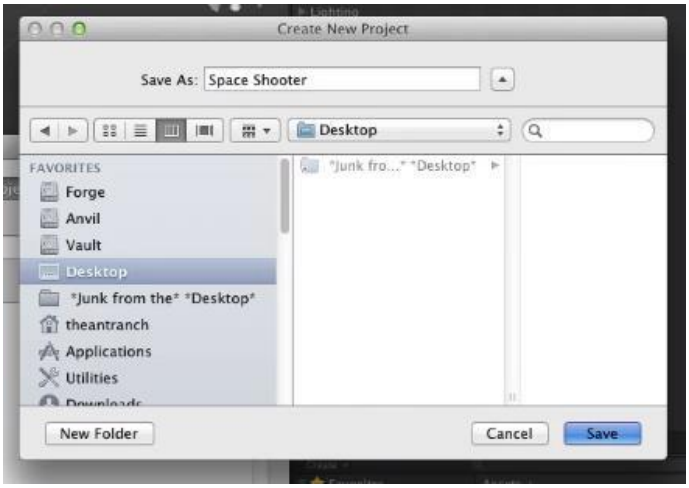
Step 1: **Open unity software and create a new project.**

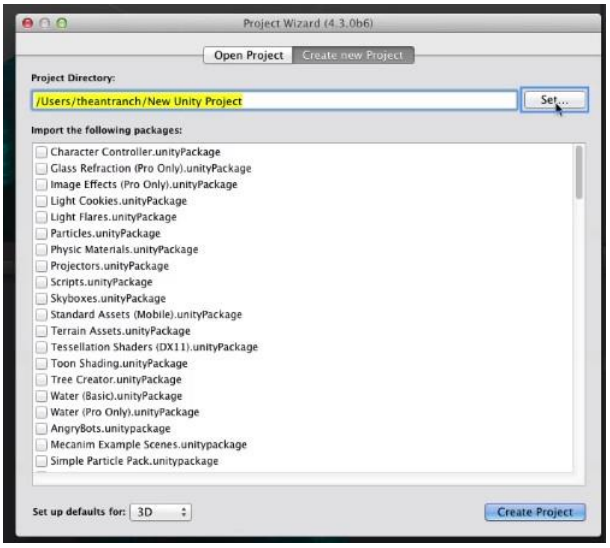
- Go to file-> New Project->Click on create new project ->set the location (By clicking on Set Button)-> Give the name to your project As Space Shooter->Click on save Button->Click on Create Button.

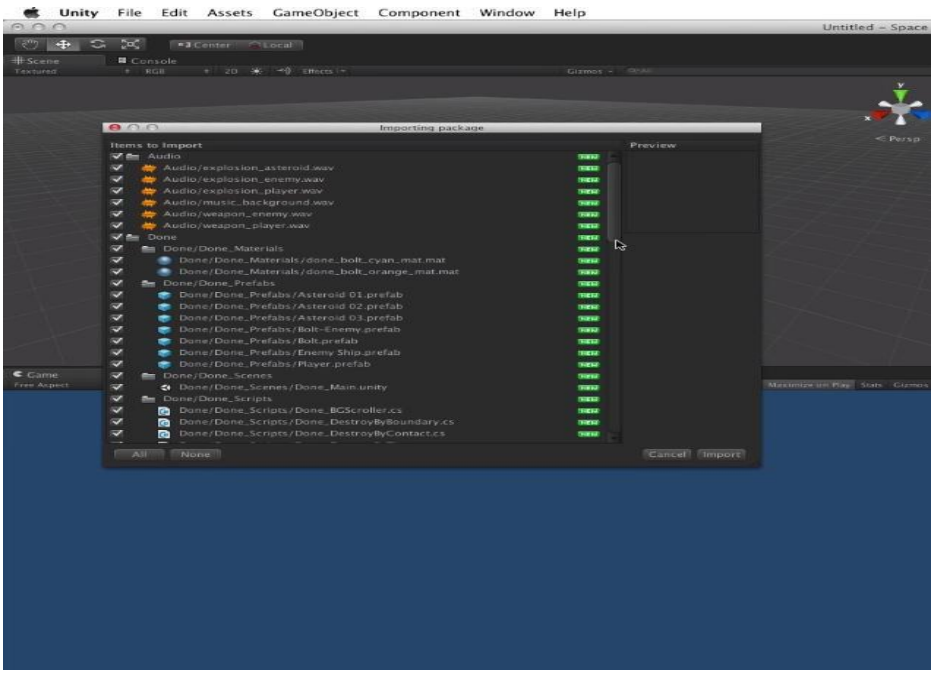


.

- Go to window button and open asset store
- Click on Unity essential, then go to sample projects
- Import the package in software



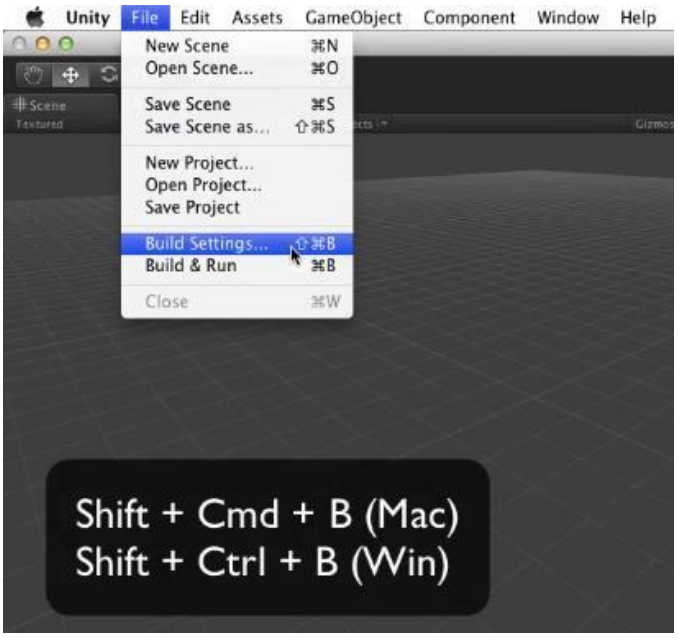


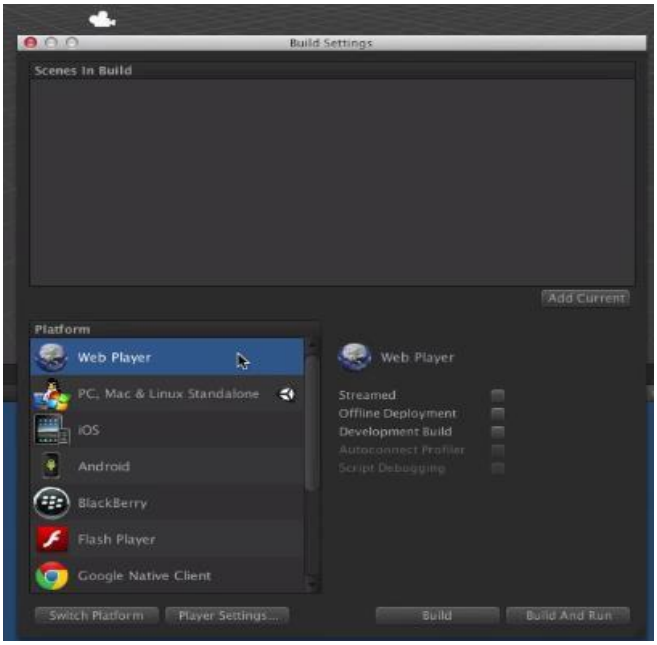


The Scene will look like after Importing the Asset -> Select All Package->Click on Import.

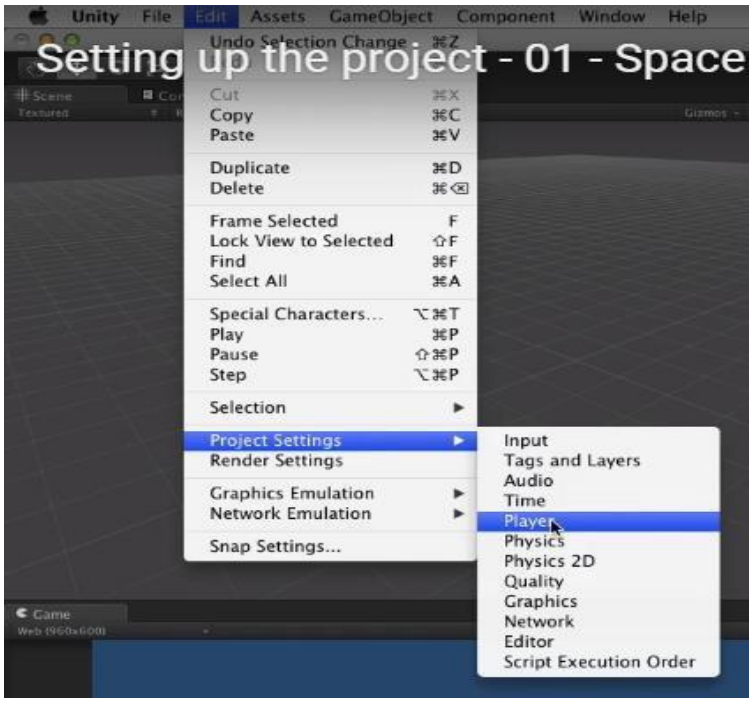
The Asset is created.

- Save your scene. (Crtl+S)
- Save the scene inside the Asset directory make new folder as _Scene. Give the name to your scene as main the save it.
  Now we will set the build target to our scene
  Go to file->Build Setting->Select Web Player->Click on switch platform

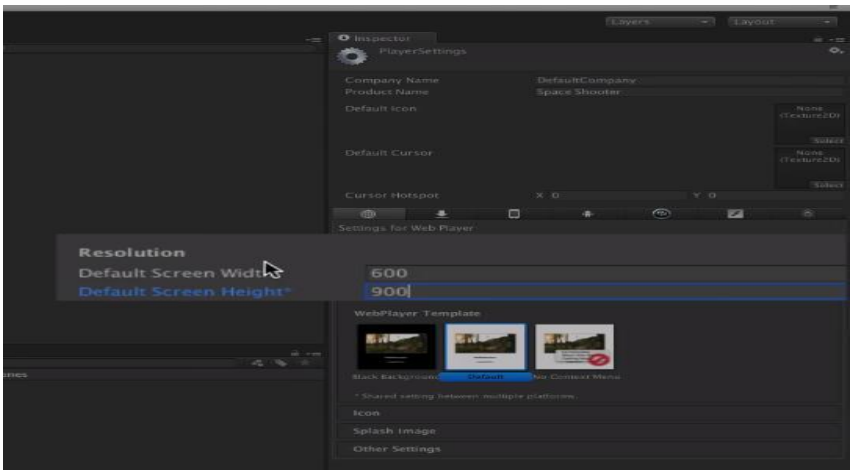




Now we need to fill the Build Detail.
Go To File-> Project Setting->Player->

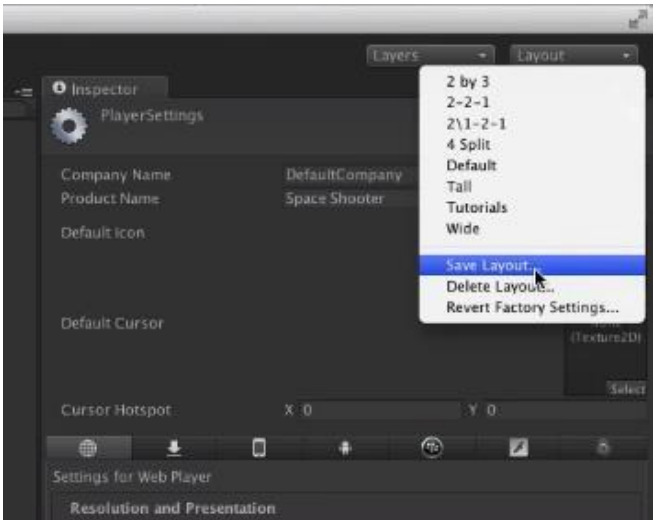


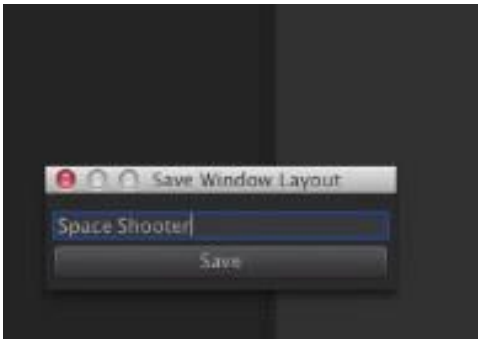
Set the resolution:
Go to Inspector window(Right Middle Corner Of the scene)->Resolution->Change the resolution width as 600 AND height as 900.



Drag the game view to the top .

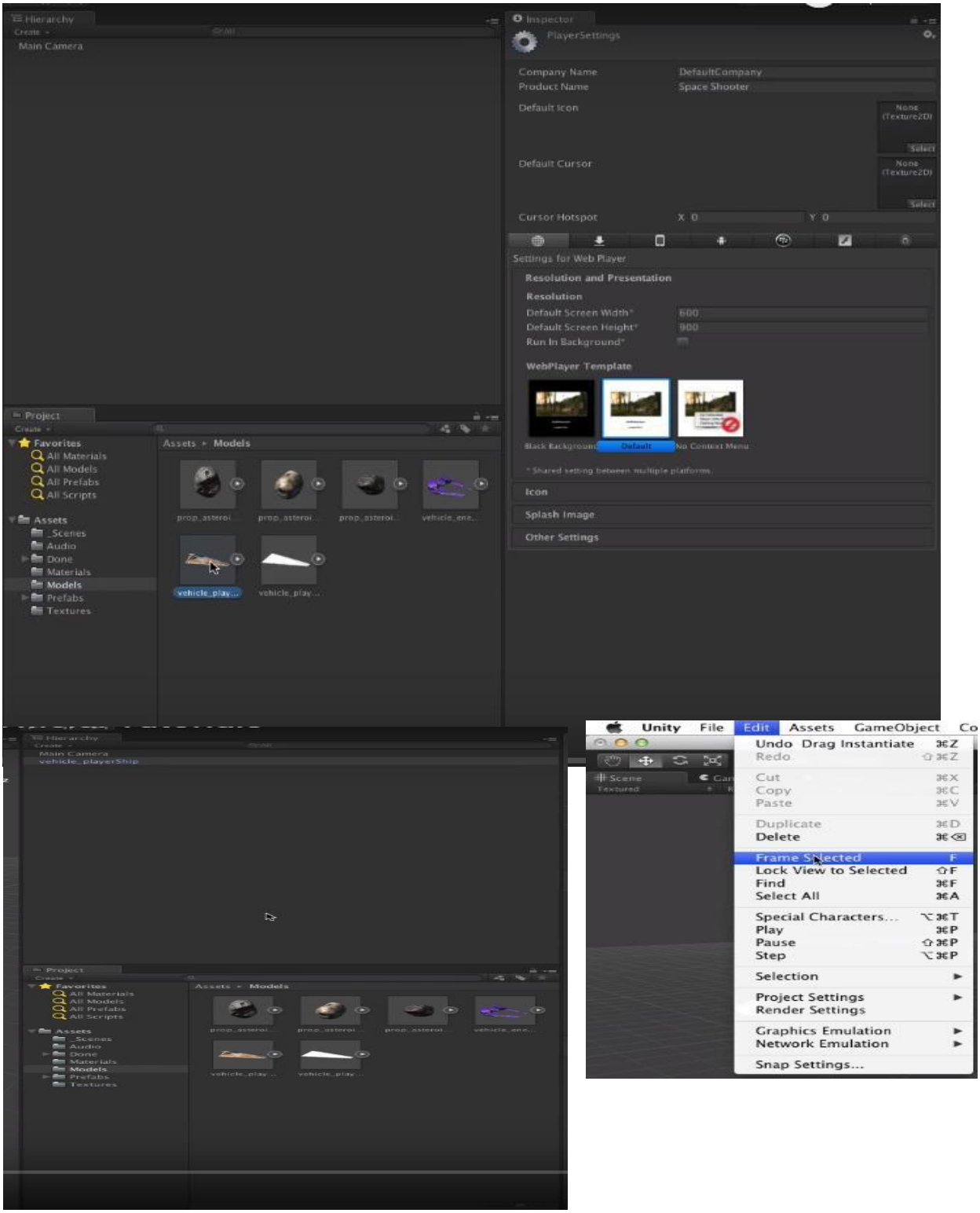
Now save the layout: choose layout->click on save layout->Give the name as Space Shooter->click on save button.





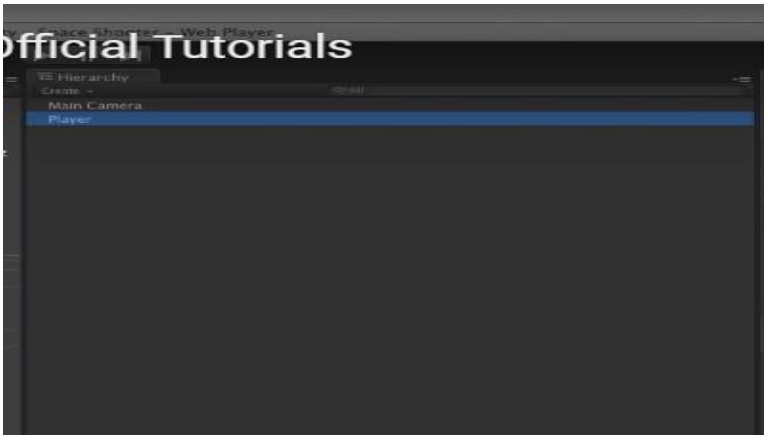
Step 2: **Setting Up the player Game Object**

Go to scene view->Add the player Ship Model from Mode Directory->Drag the vehicle player Ship from model directory to hierarchy

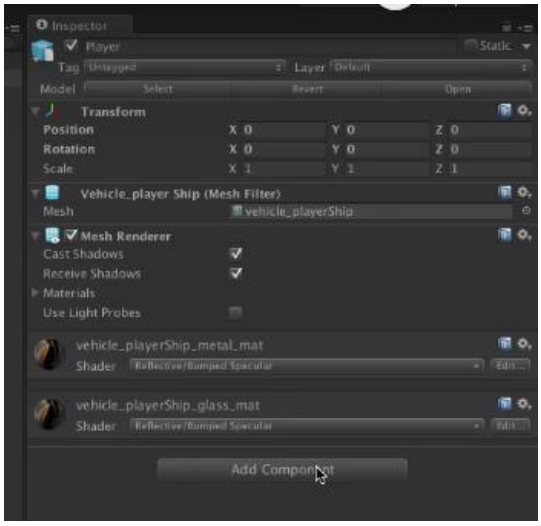


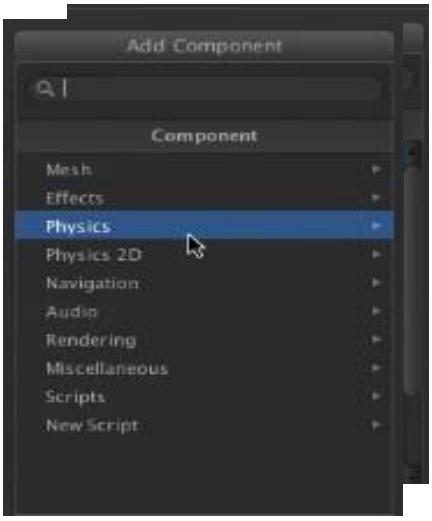
Go to edit->from selected

Now Ho To Hierarchy->double click on Vehicle ship->Rename it as Player->press Enter Key

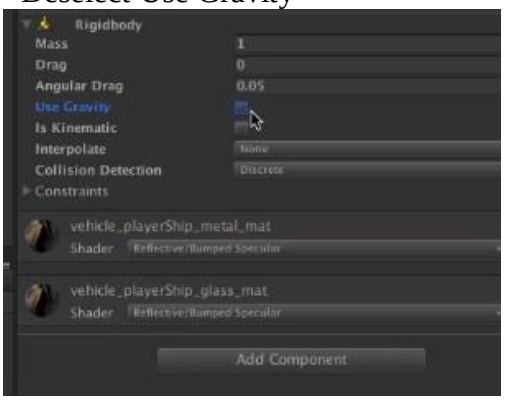


Add New Component into the Inspector Window
Click on Add Component Tab->Select Physics->Rigid Body

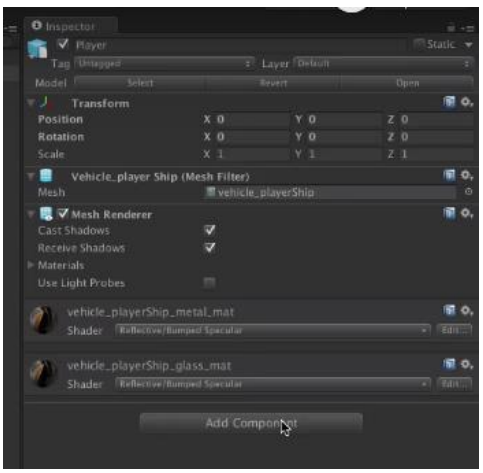


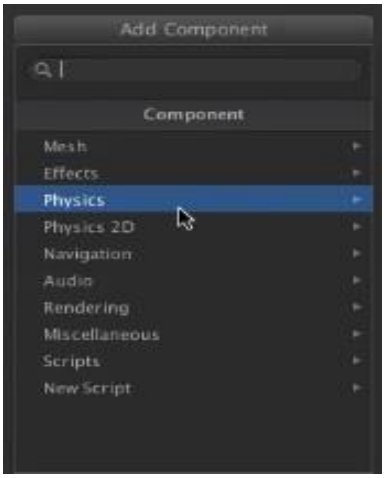


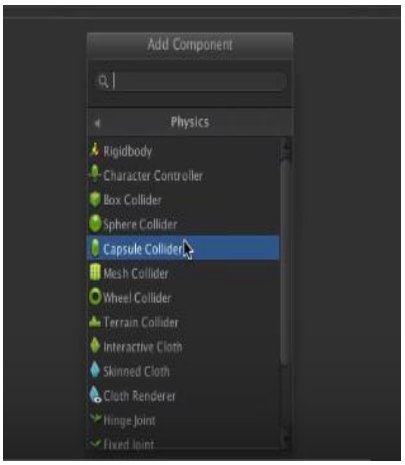
GotoRigit Body Component ->Deselect Use Gravity ->



Add New Component into the Inspector Window

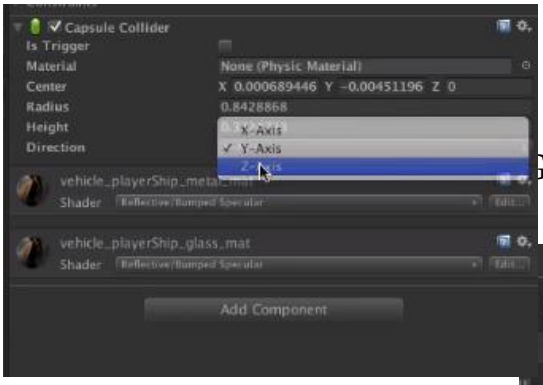




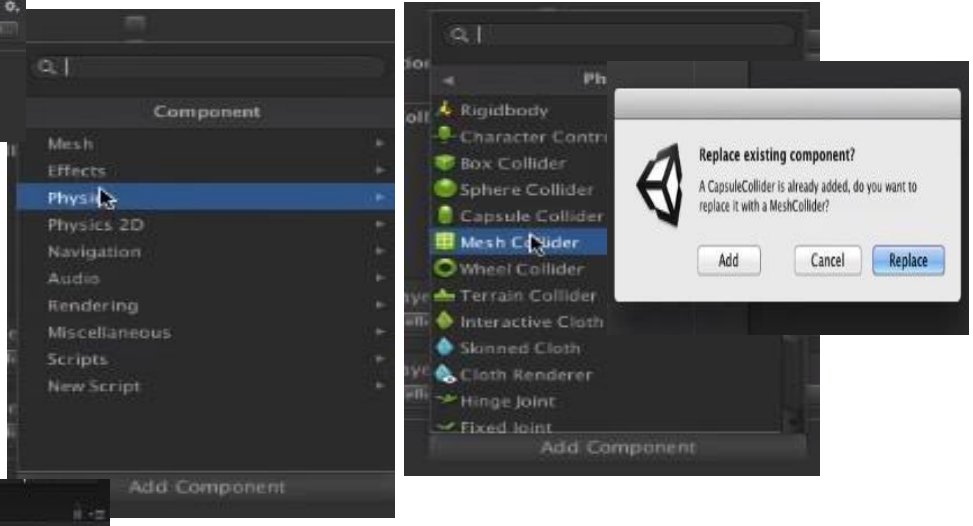


Click on Add Component Tab->Select Physics->Select Capsule Collider

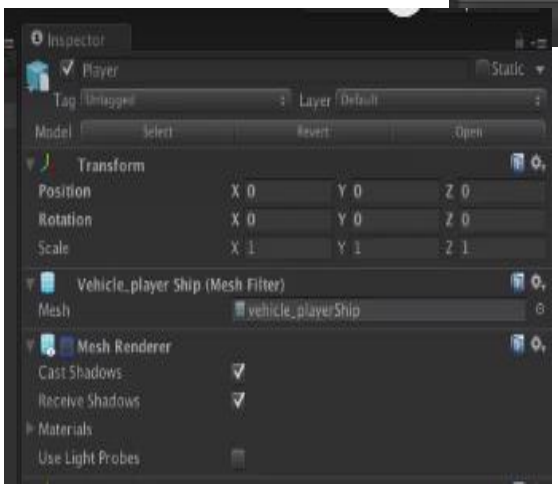
Now Change the Capsule Collider Direction to Z-axis->change the radius and Height also ->



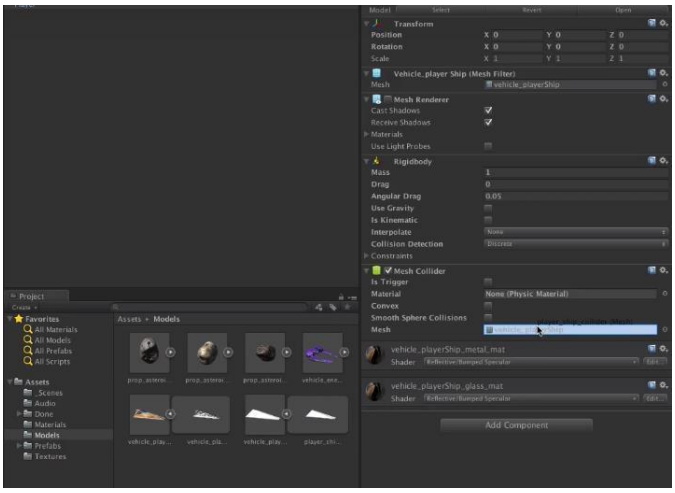
Go to Add Component button again->Physics->Mesh Collider->Replace



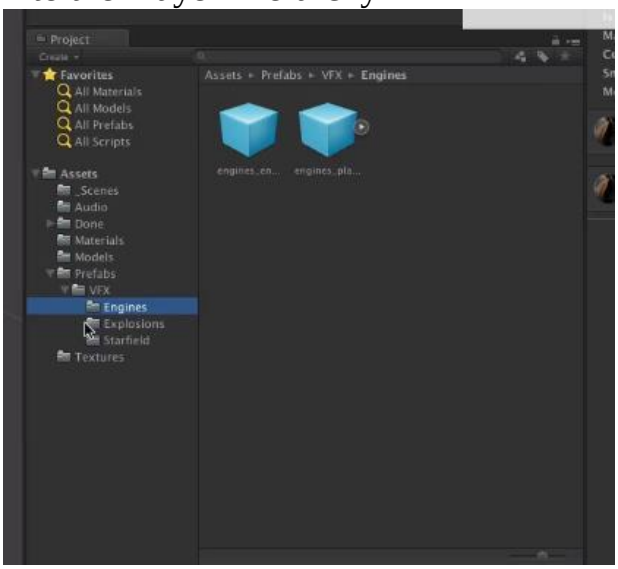
Go to mesh collider Component inside the Inspector window->turn off the Mesh renderer tab

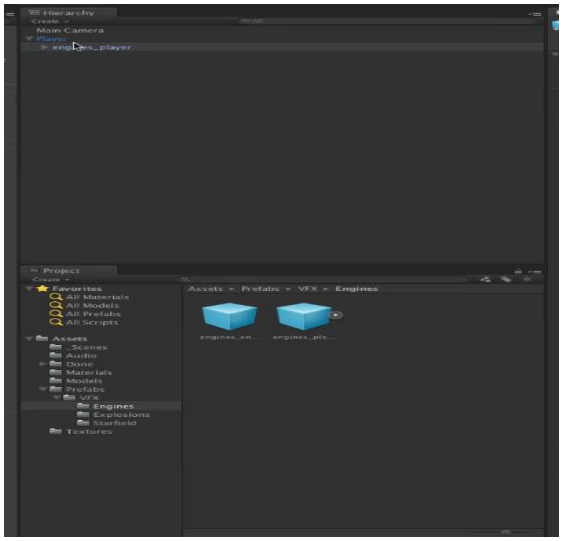


Open the Model-> Select the Mesh Asset->Drag it in to the Mesh Slot on the Mesh Collider->Turn On the Mesh Renderer tab-> Select Is Trigger tab

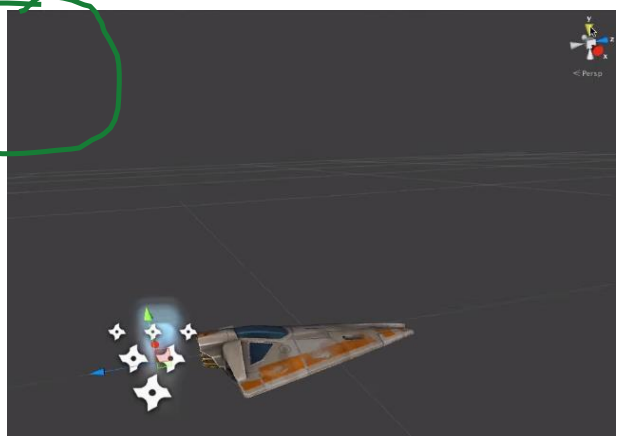


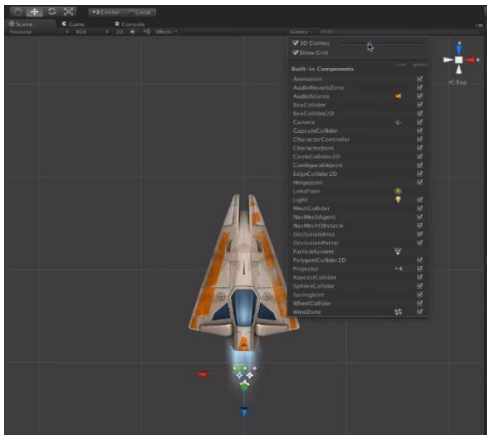
Go to->Asset->Prefab->VFX->Engines->Drag it into the Player Hierarchy





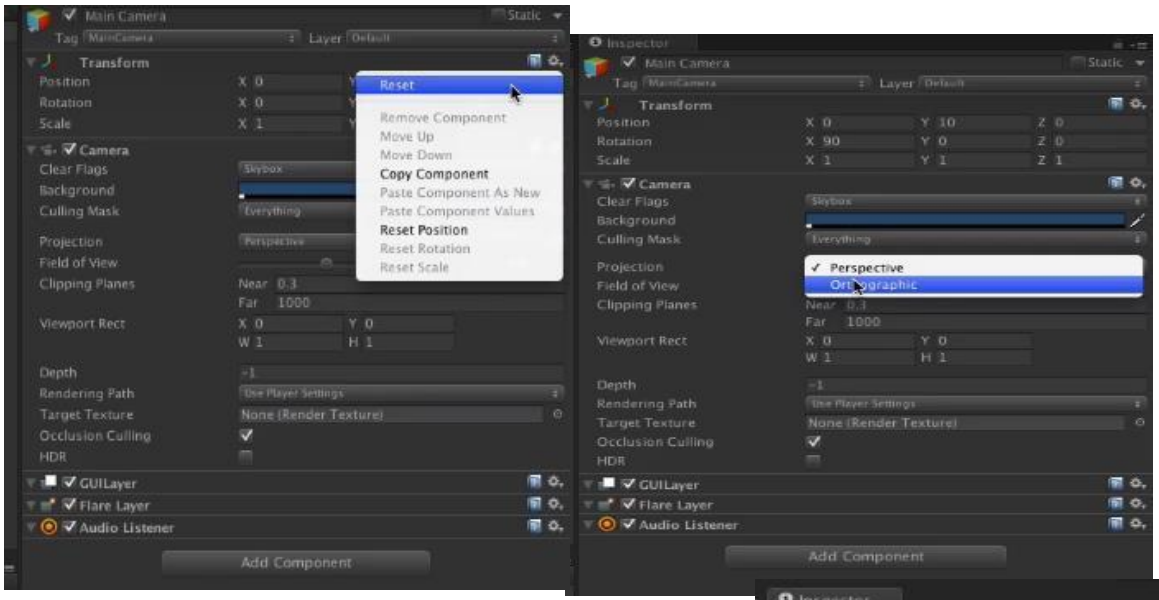
You Can Change the Position of Ship By Changing the Tap View





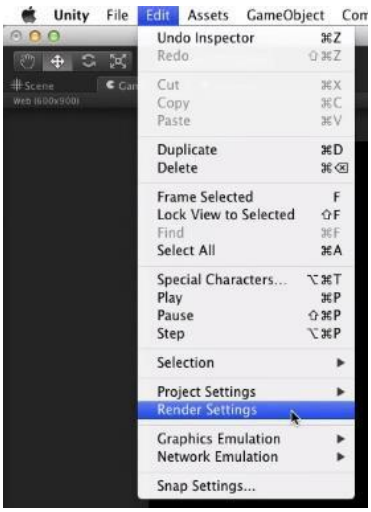
Step 3 : **Setting up the main camera and lighting for the scene.**

Click on the main camera from hierarchy->Go To Inspector window->go to transform component ->click on Reset Tab->Now Change the Rotation of x-axis as 90->go to camera view ->click on the projection->set it as orthographic->Se the size to 10

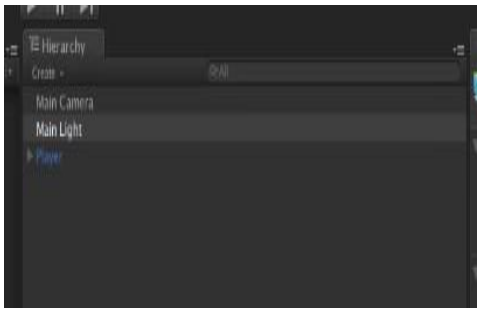


Go to camera -> change the clear flags to Solid Color-> Change the backgro und color as Black

Setting up the Light
Go to edit->Render Setting->Change the Ambient Light to 0.0.0.0(Black in color)

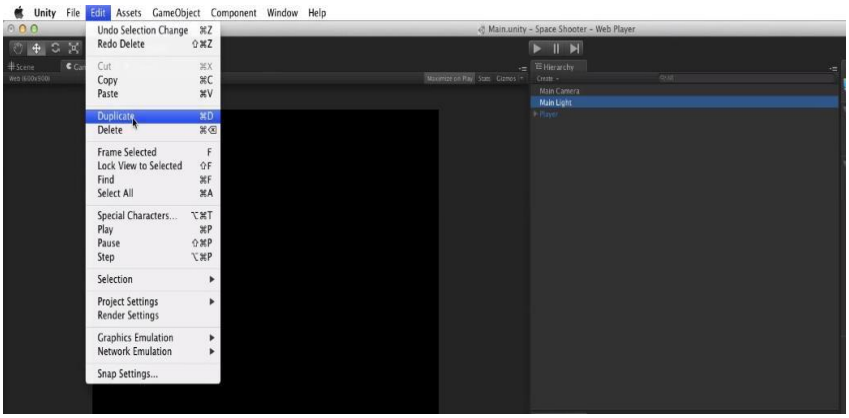


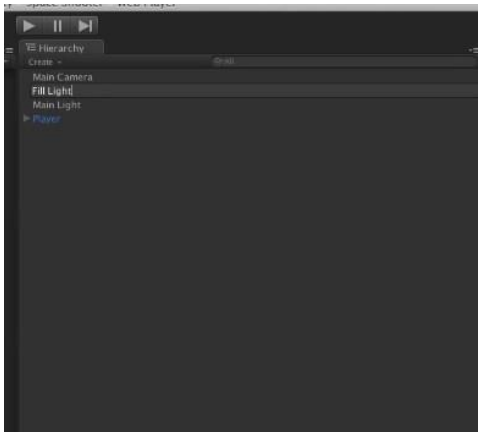
Go to hierarchy->click on create->Directional Light->Rename it with Main Light->Reset the light position->set the rotation x axis as 20->y axis as -115->



s

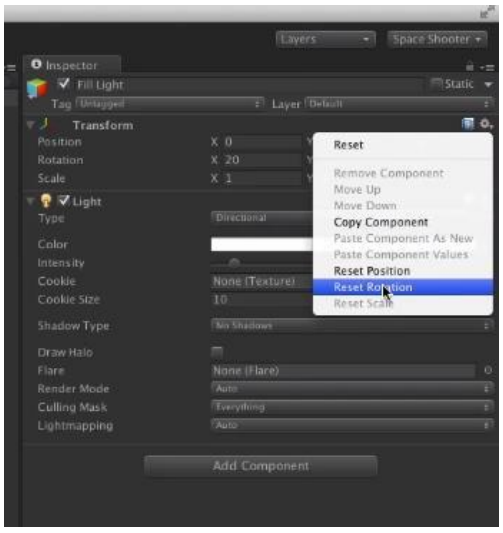
Select the Main Light from Hierarchy->go to Edit Menu->Select Duplicate ->Rename the Duplicate as Fill Light



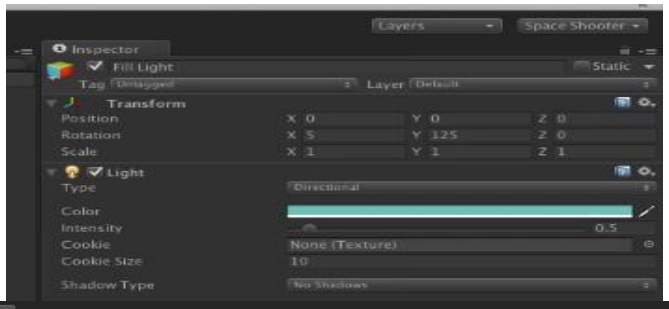


Rotation of fill Light

Go to Inspector window->reset the

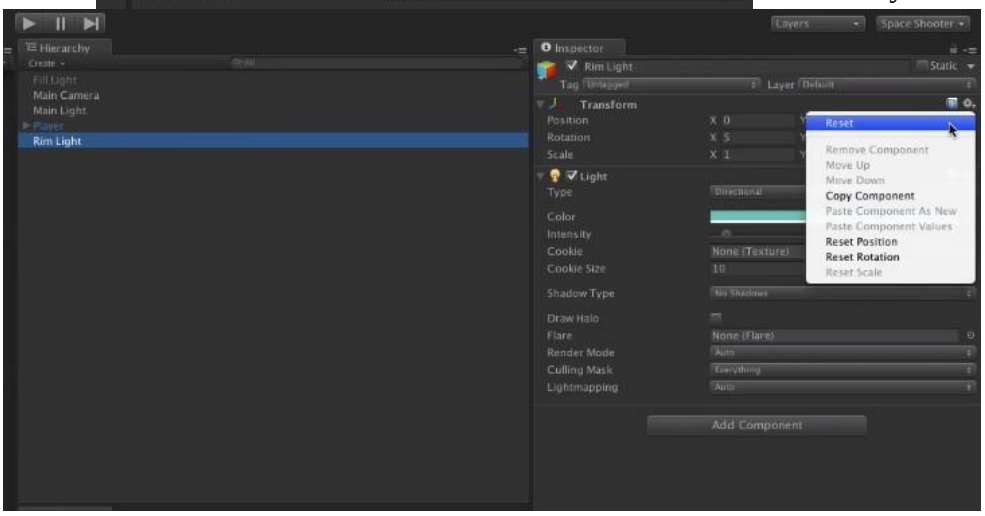


Change the light Intensity as 0.05->Change the Colors->Change the rotations x-axis as 5 and y-axis as 125->



>Rename it Light From Light From

Duplicate the fill light- with Rim Light->Deselect the Rim Inspector Window->select Rim Hierarchy->Reset the transform

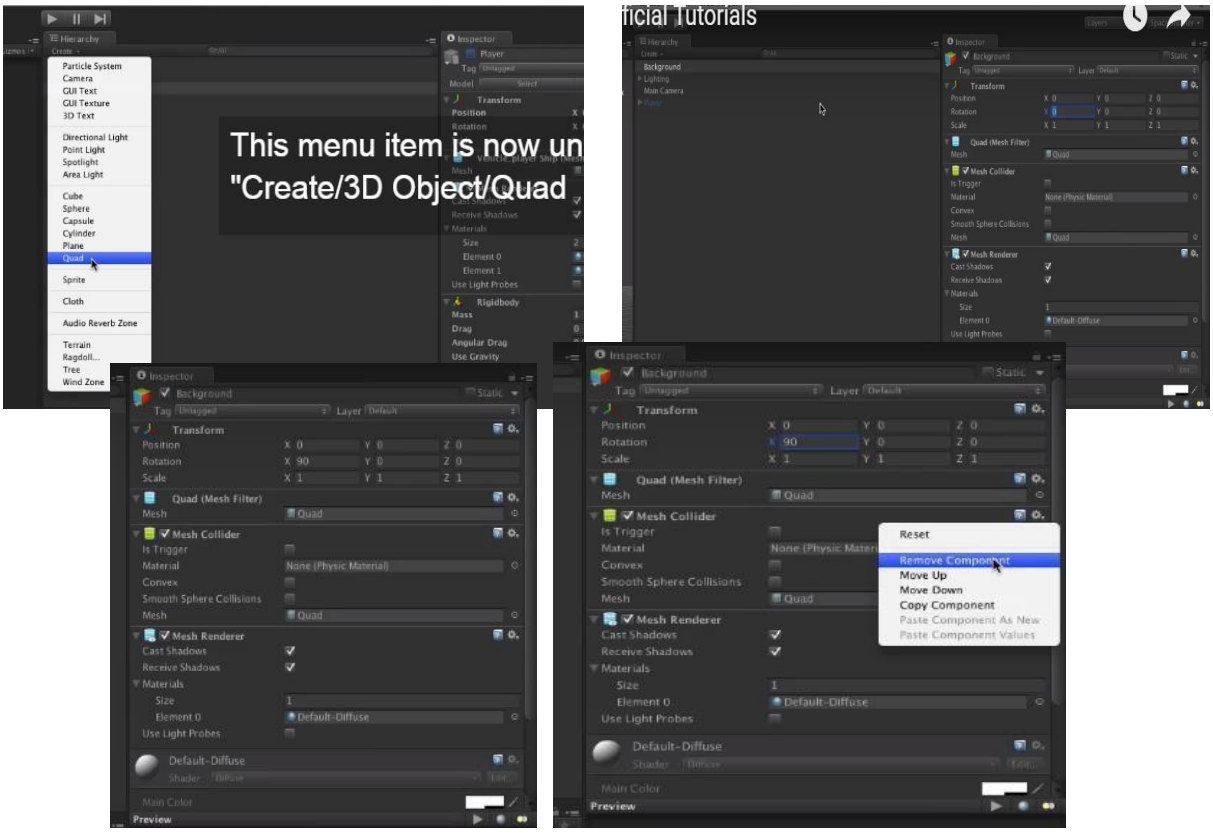


Add Empty game object to the scene->press Shift+Ctrl+N-> Rename it with Lighting->Reset the transform Component

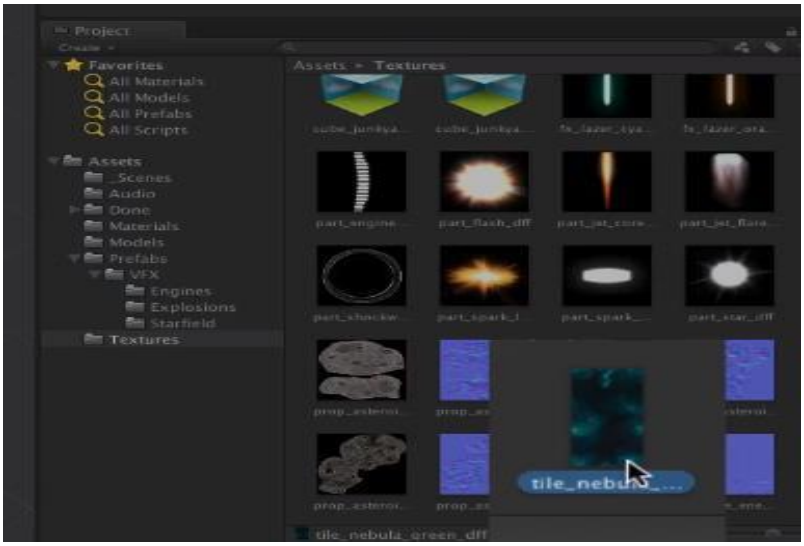
Step 4:**Adding the backgroung**

Click on player in the hierarchy->go to transform component ->deselect Player tab
Create the quad to hold the background Image
Go to Hierarchy->click on Create->Select Quad->Rename it as Background->reset the game object

Cha nge the rotat ion of back grou nd x- axis as 90 Go to Mesh Collider- >Remove Component

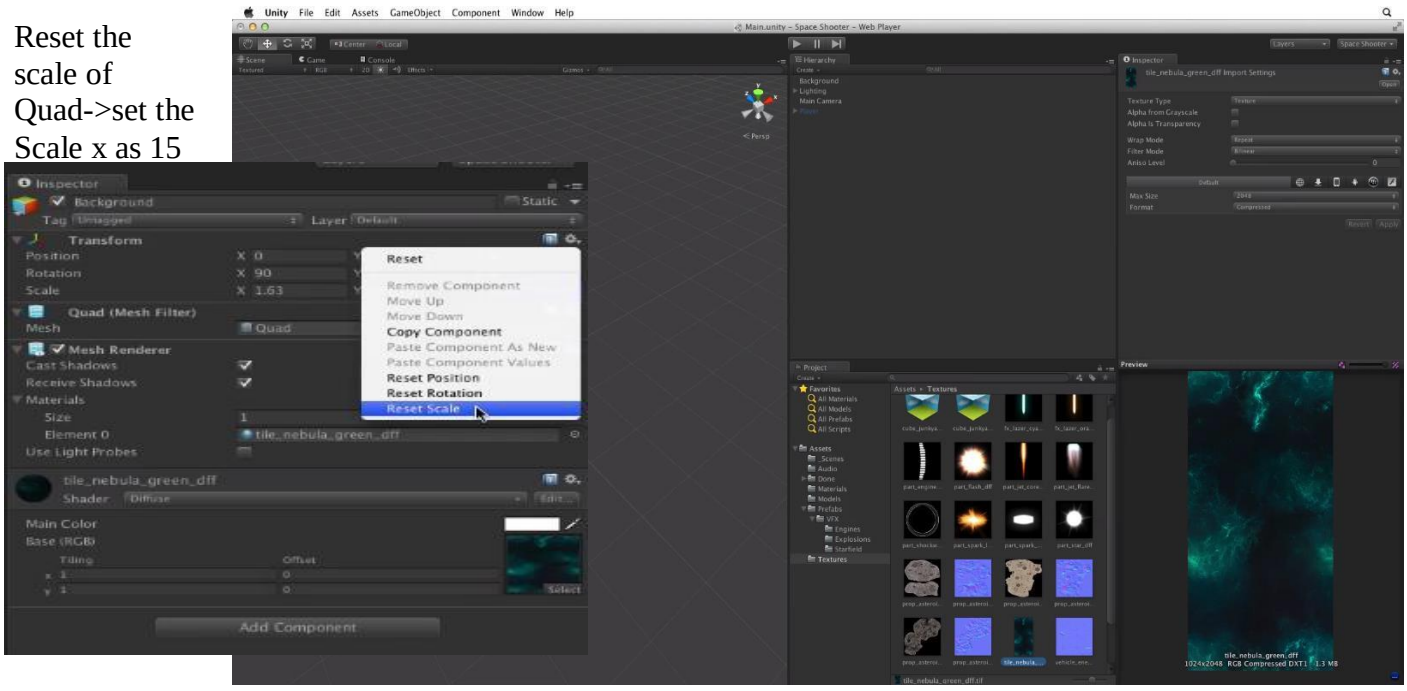


Now Add The Texture To Our Background:
Go To Project ->Assets->Texture->Select Nebula

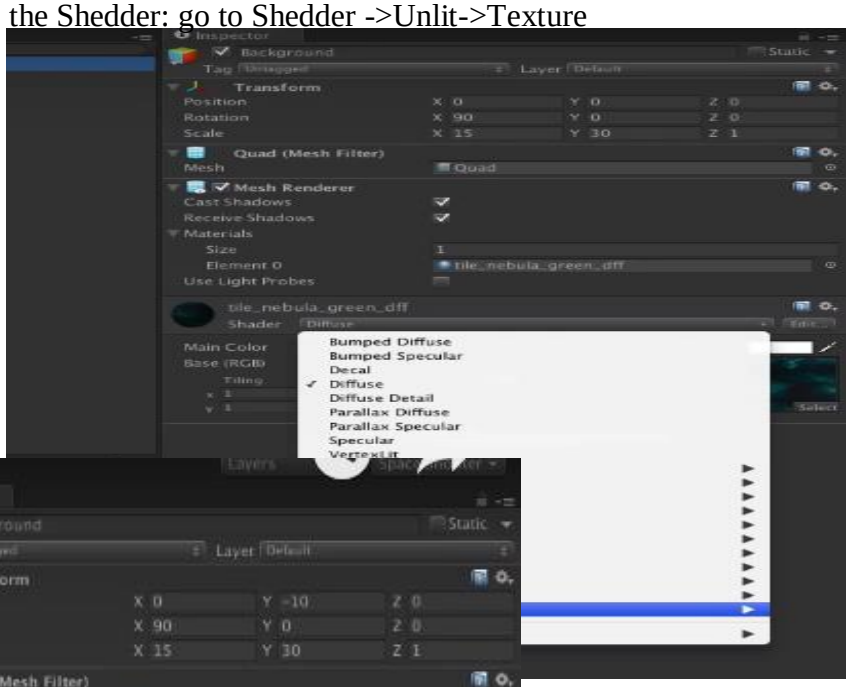


Drag the Image and drop it on the scene view
Drag the Image and drop it on the scene view
Drag the Image and drop it on the scene view
Drag the Image and Drop it to the Scene View

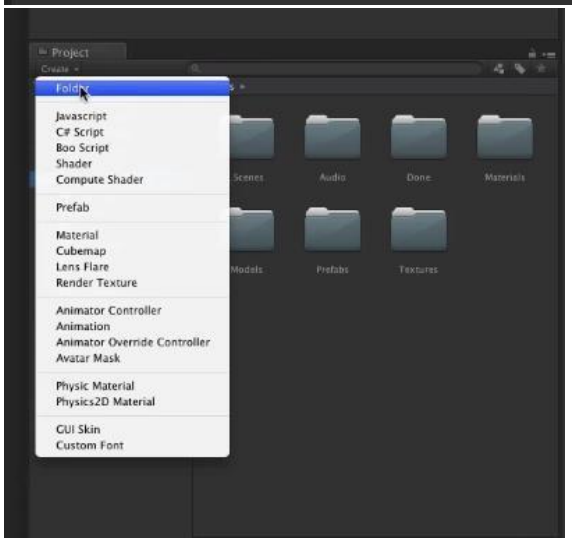
Reset the scale of Quad->set the Scale x as 15



Change the Shedder: go to Shedder ->Unlit->Texture



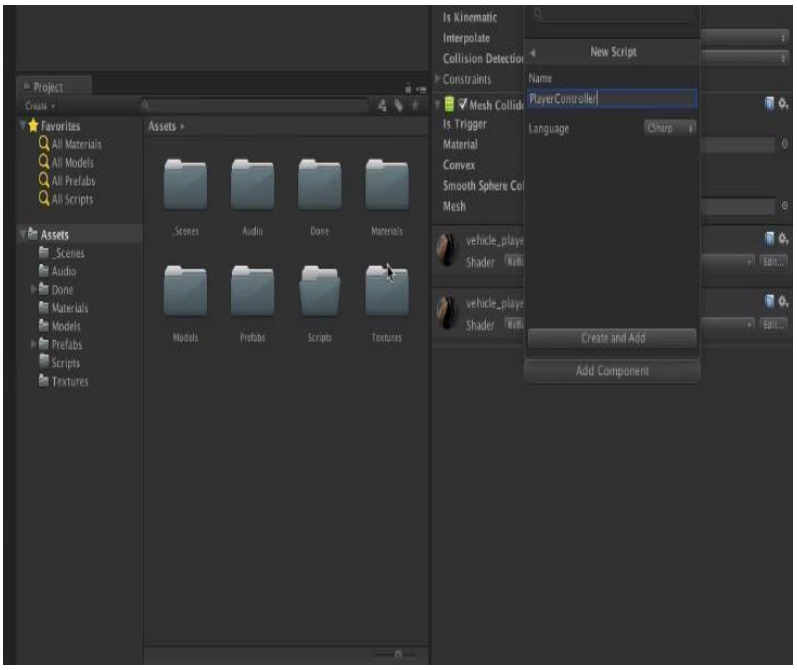
Click on background in the hierarchy->go to Inspector window->change the y position to -10

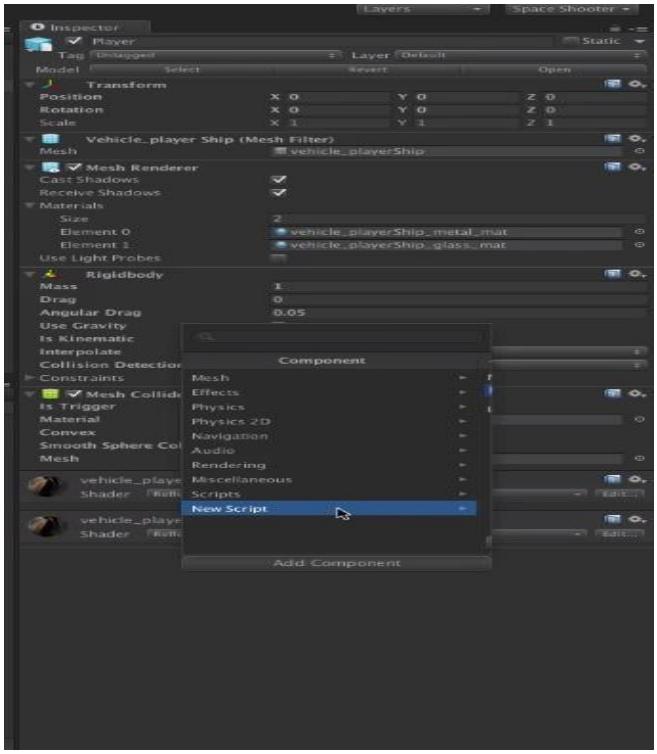


Step 5:**Moving the Player**

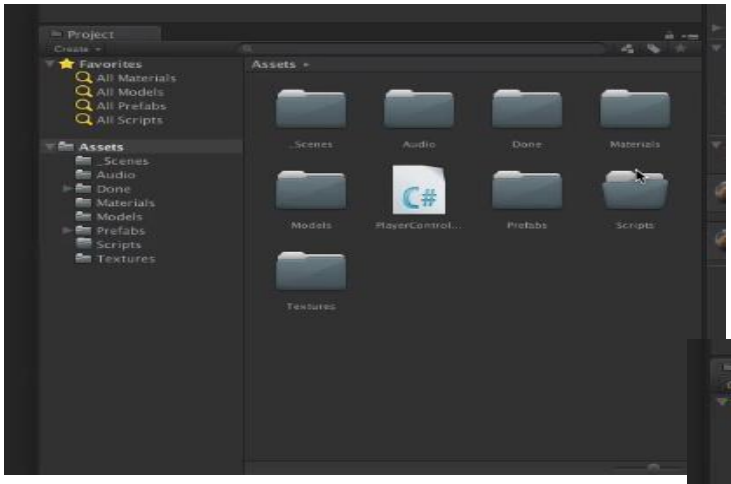
Go to Asset->click on create-> folder->give the name as Script >Press Enter->Now We

Will Create a new Script to our Player Ship->Click on Player In the Hierarchy ->Go to Inspector Window->Click on Add Component Tab->select New Script->Give the Name to script as Player Controller->Click on Create and Add Button.

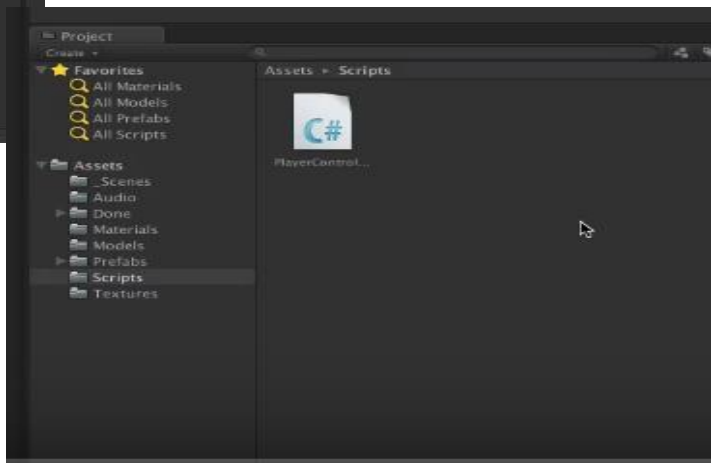




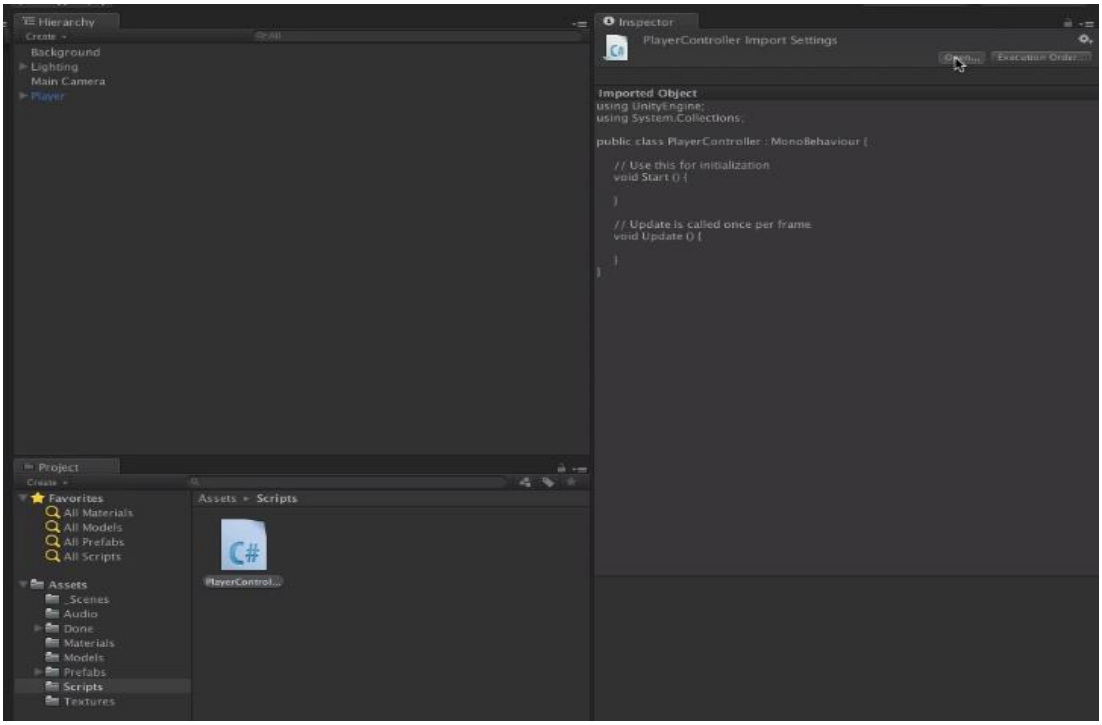
The Script is created in c-sharp .
Now Drag the Player Controller Script Into the Script Folder



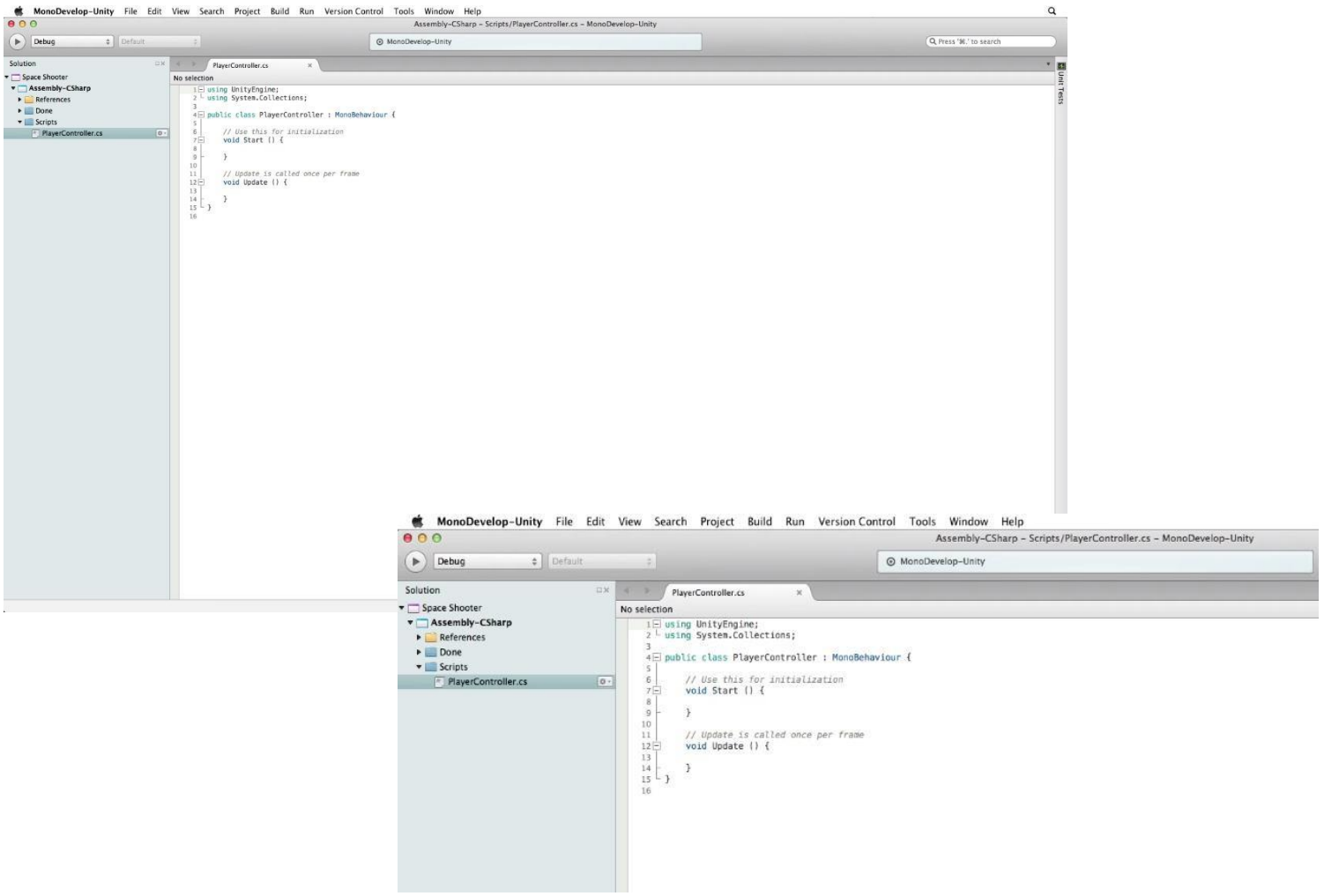
Open the script folder to view the



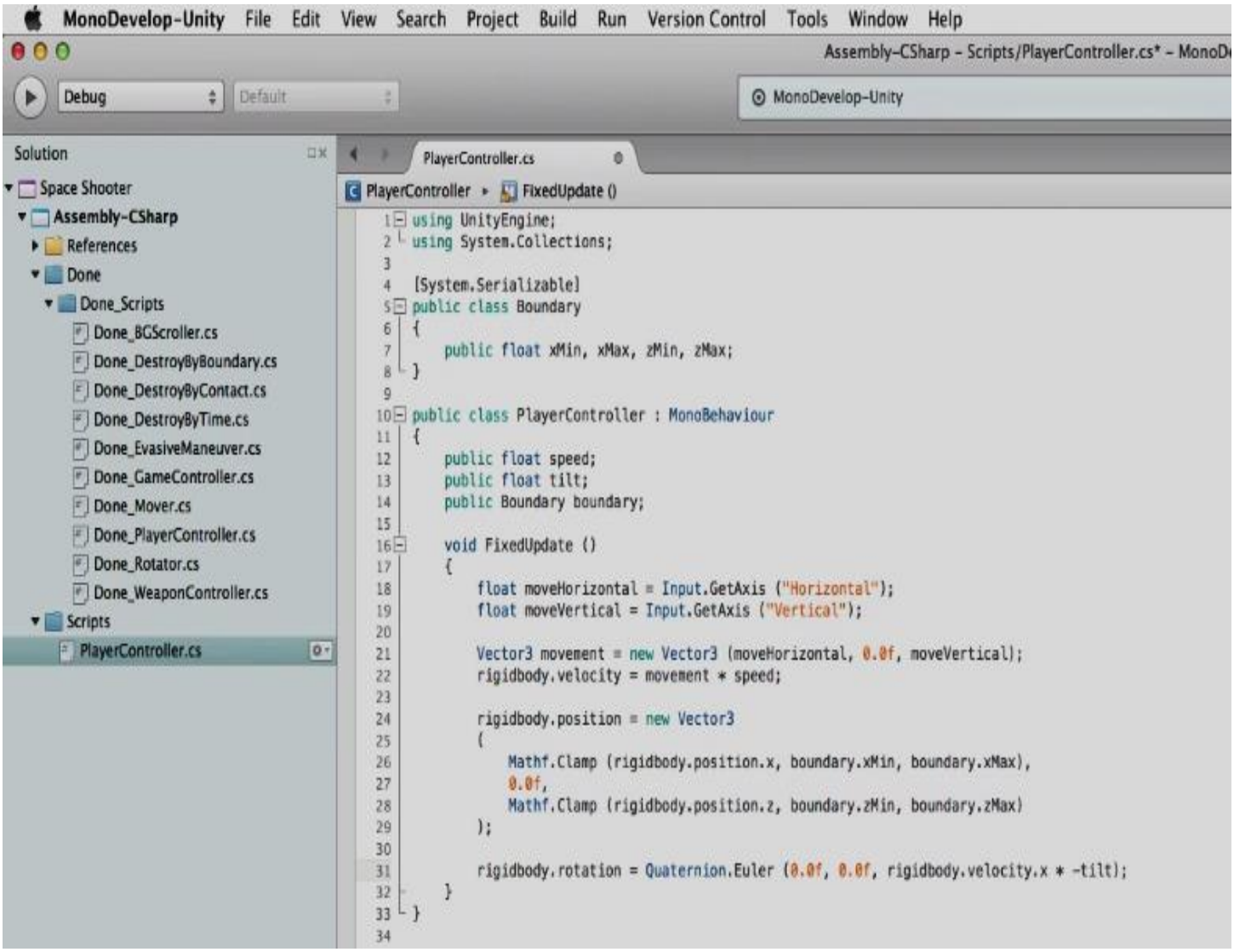
Select the script->Click on Open



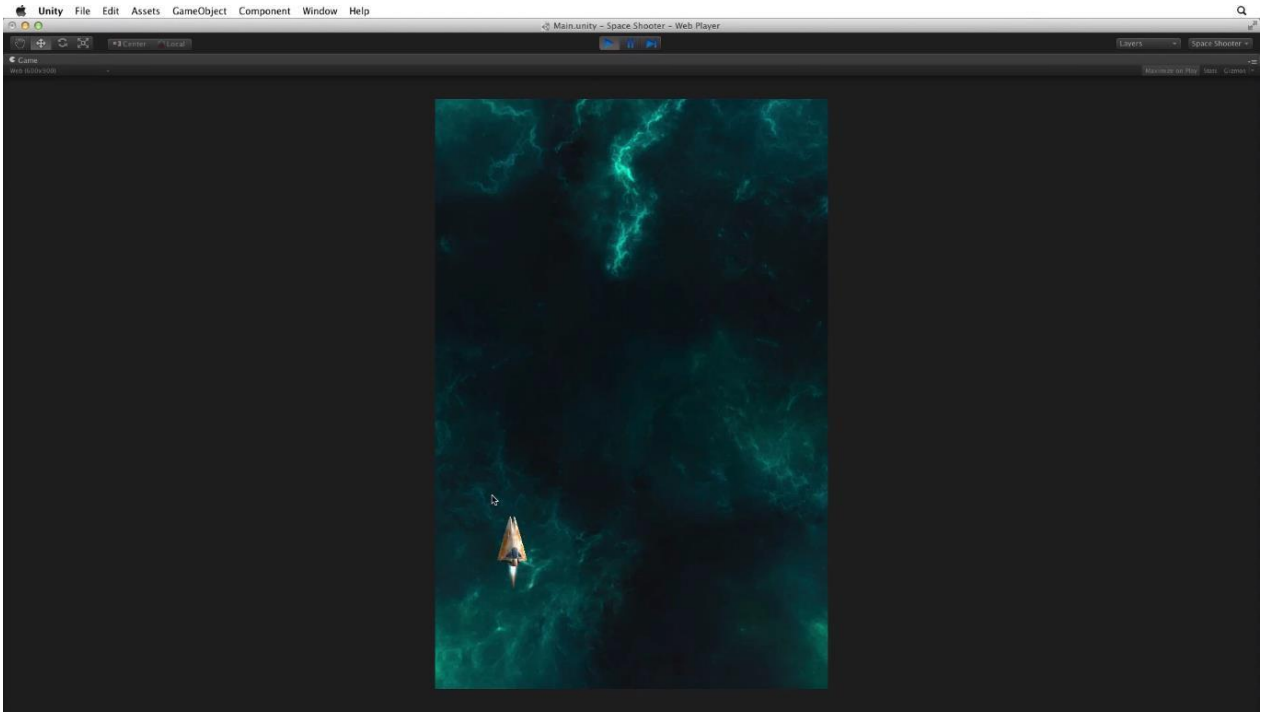
The Script will Open in the monoDe veloper code window Remove all the sample code from the script->



Type the following Source Code into the script
Set x-min Value as 6->x-max value as -6->z-min value as 4->z-max value as -4 From the Inspector Set x- In the Inspector Window do the following settings



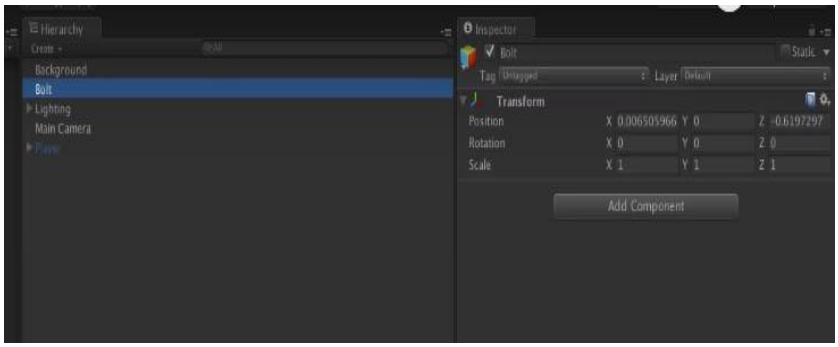
->Set x-min Value as 6
->x-max value as -6
->z-min value as 4
->z-max value as -4
->Adjust the tilt value
->Set the speed to the ship
Run The Script From Edit Menu Button the output will be



Step 5:**Creating Shots**

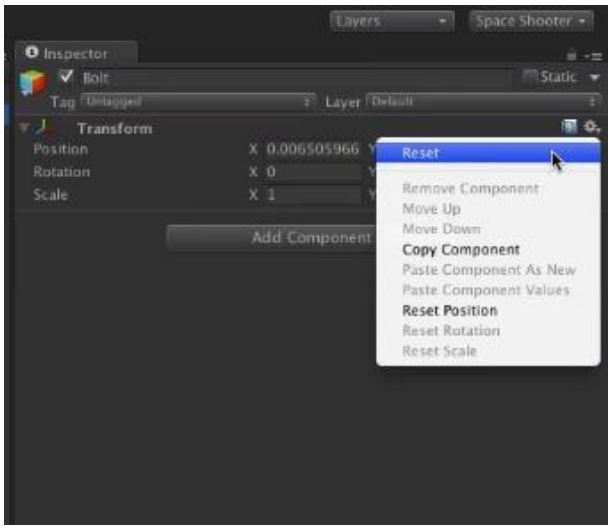
Now Will Create Shots to Our Player

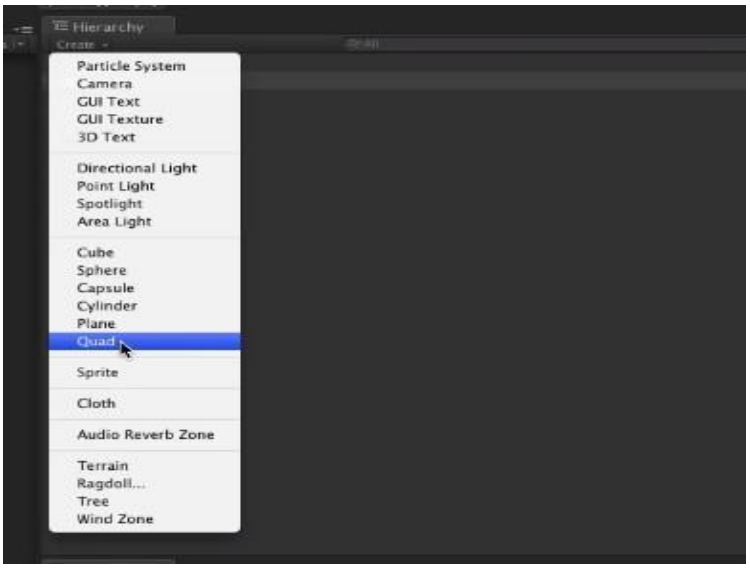
Now Create new Empty Game Object In the Hierarchy->Press Shift+Cntrl->Give the name to the Game Object (Bolt)

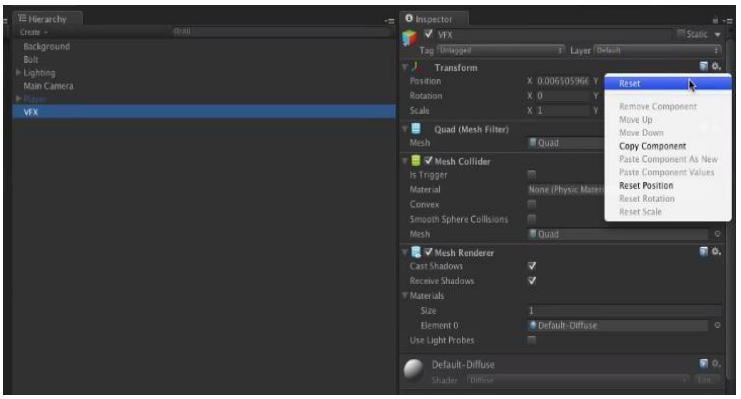


Click on Bolt From Hierarchy->Reset the game object of Transform to Origin

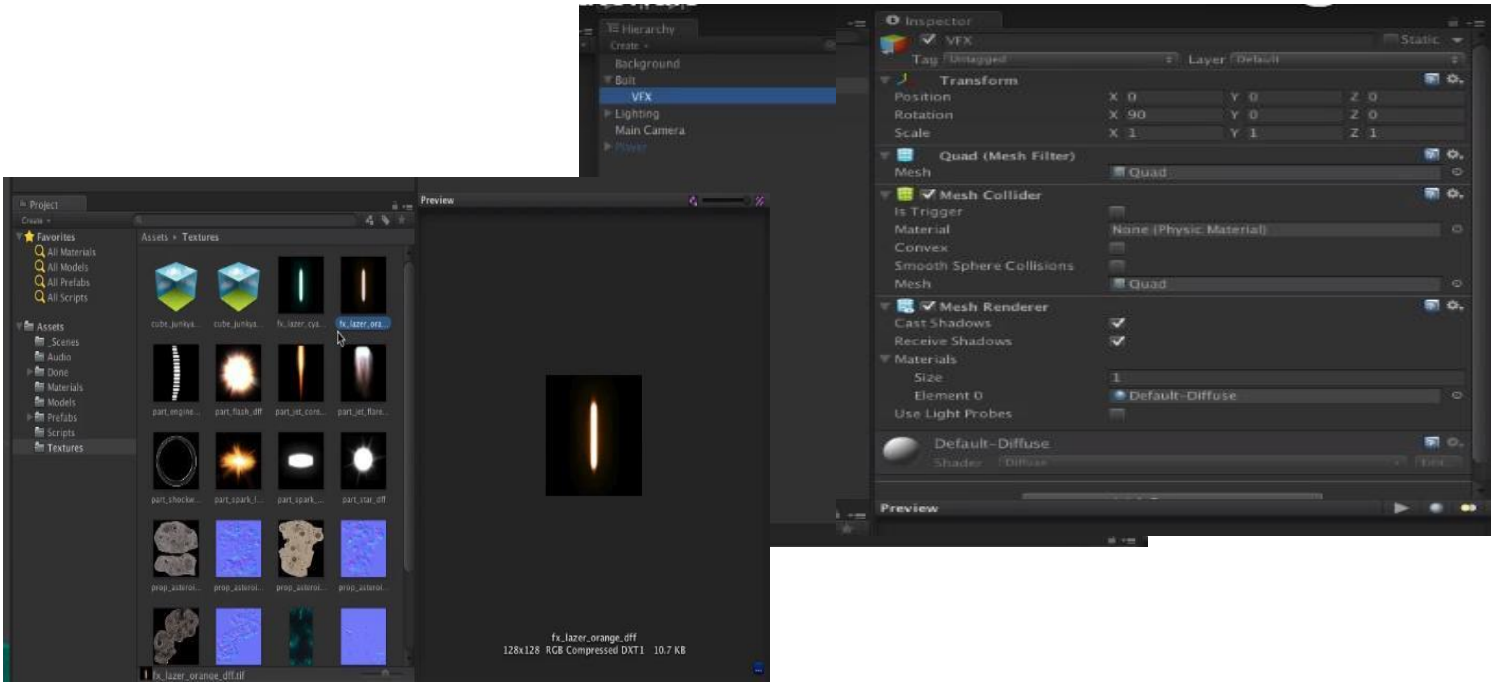
Create Quad From Hierarchy->Rename it as VFX- >Reset The Transform Position to Origin



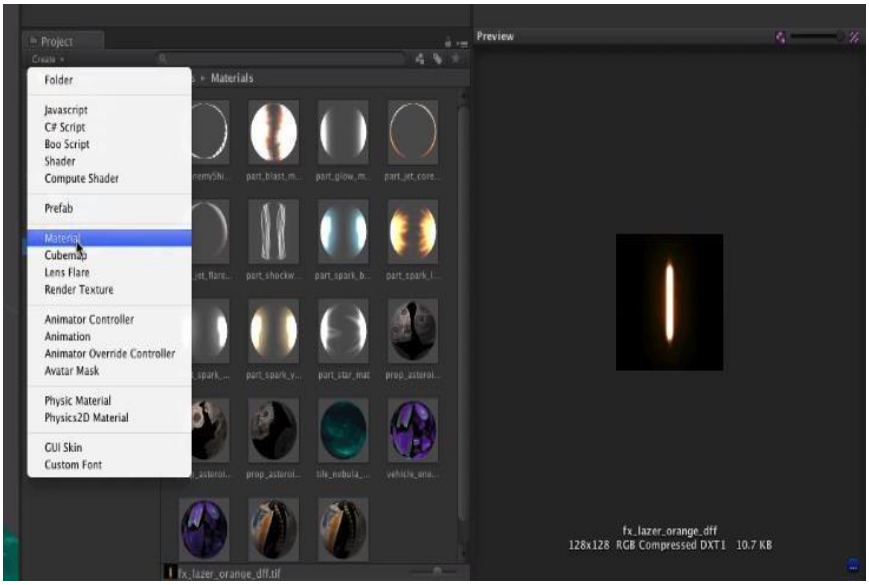




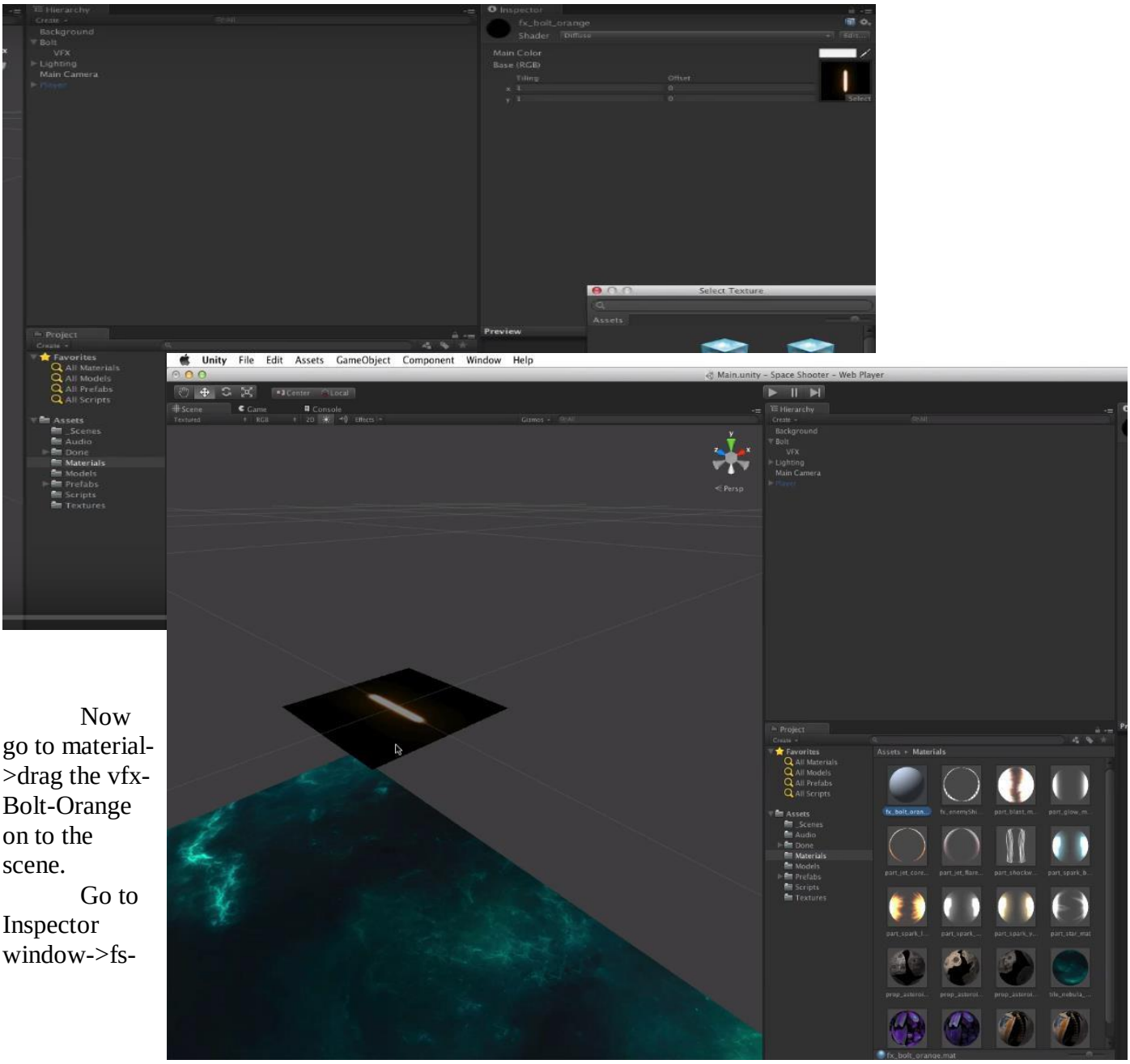
Now Drag The VFX Game Object Into the Bolt- >Change x-axis position as 90->Go to Assets->Texture->Select FX- Lazer->



Go to material folder in the asset->Click on Create tab->Choose Material->give the name as Fx-bolt-orange



Now will add the Texture Into the Material
Go to Inspector Window ->Click on fx-bolt-orange->click on Select->click on the Texture U want to add into material

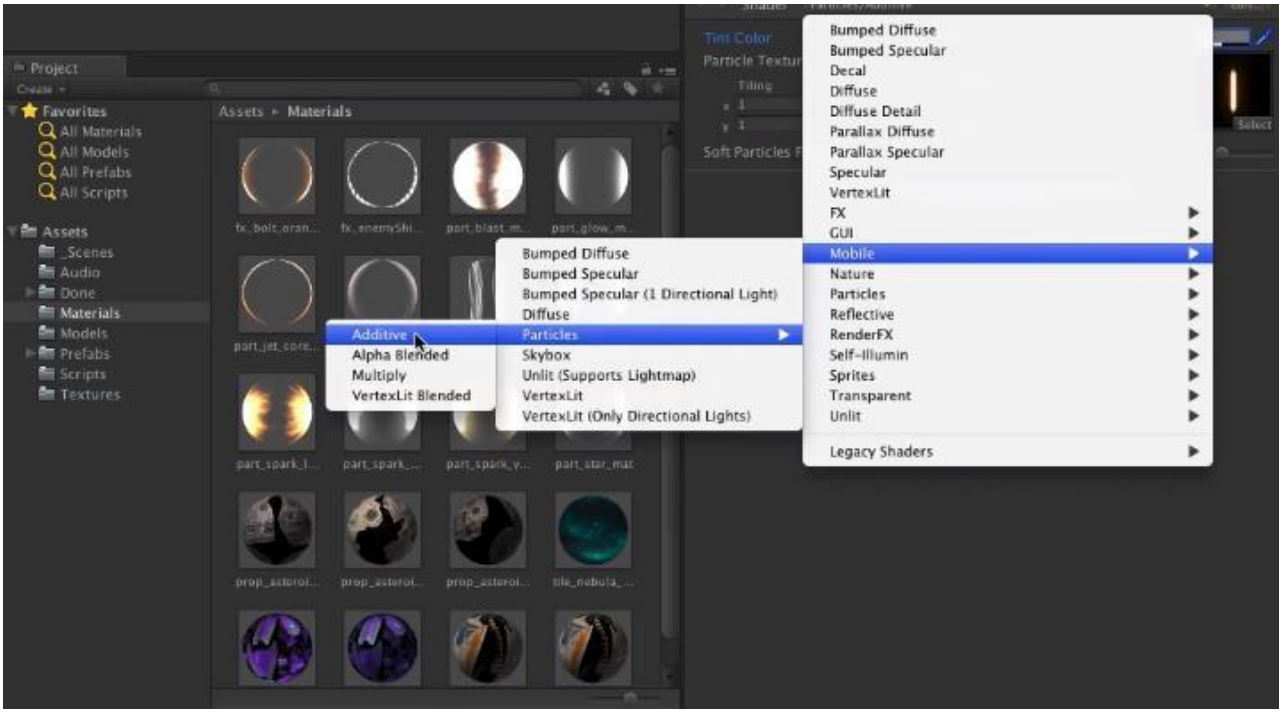


Now go to material->drag the vfx-Bolt-Orange on to the scene.

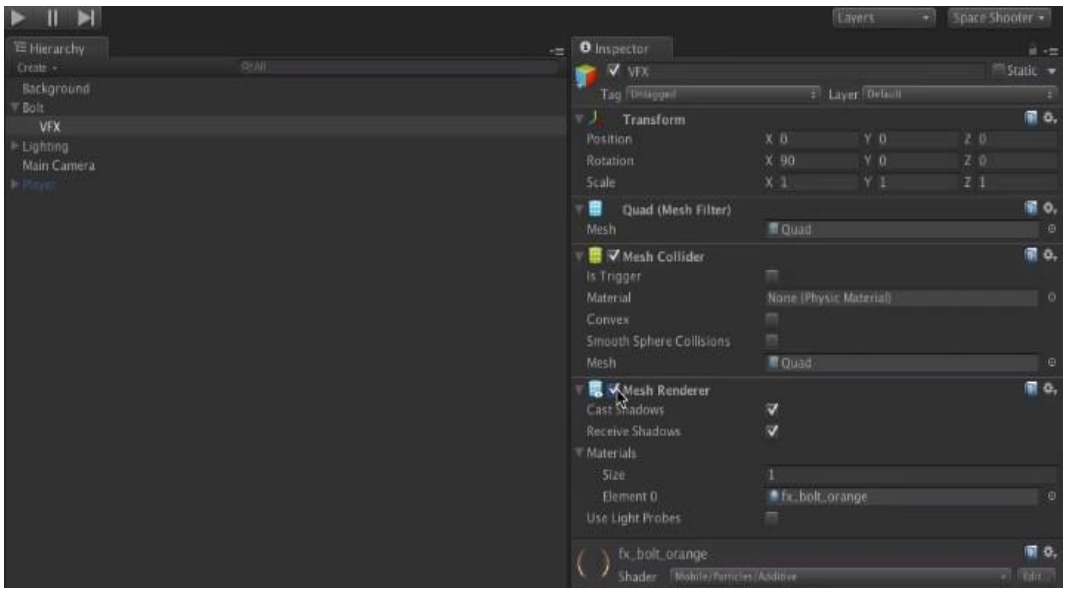
Go to Inspector window->fs-

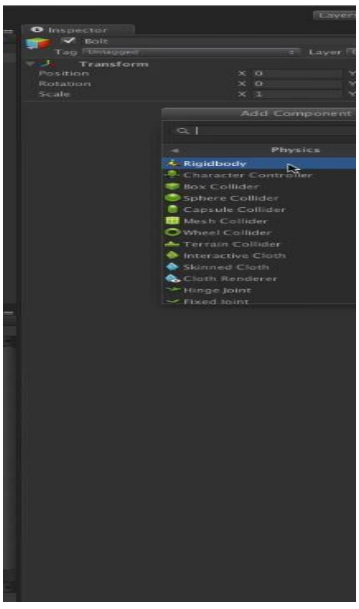
bolt-orange->Shader->Mobile->Particles->Additiv>Rigid Body->Deselect Use Gravity Tab.

Selec t Bolt from Hier arch y->Go to Inspe ctor->Cli ck on Add Com pone nt>Physics-

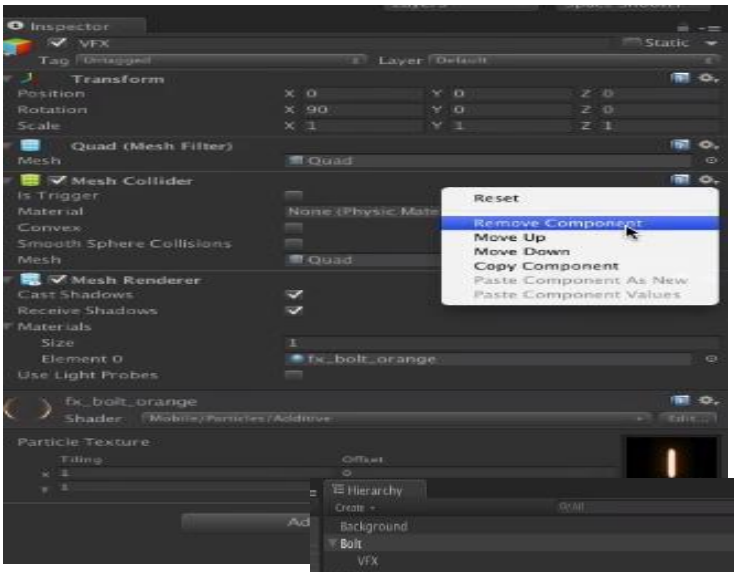


Now Go to Hierarchy->Click on VFX->Go to Inspector->Select- >Mesh Renderer tab



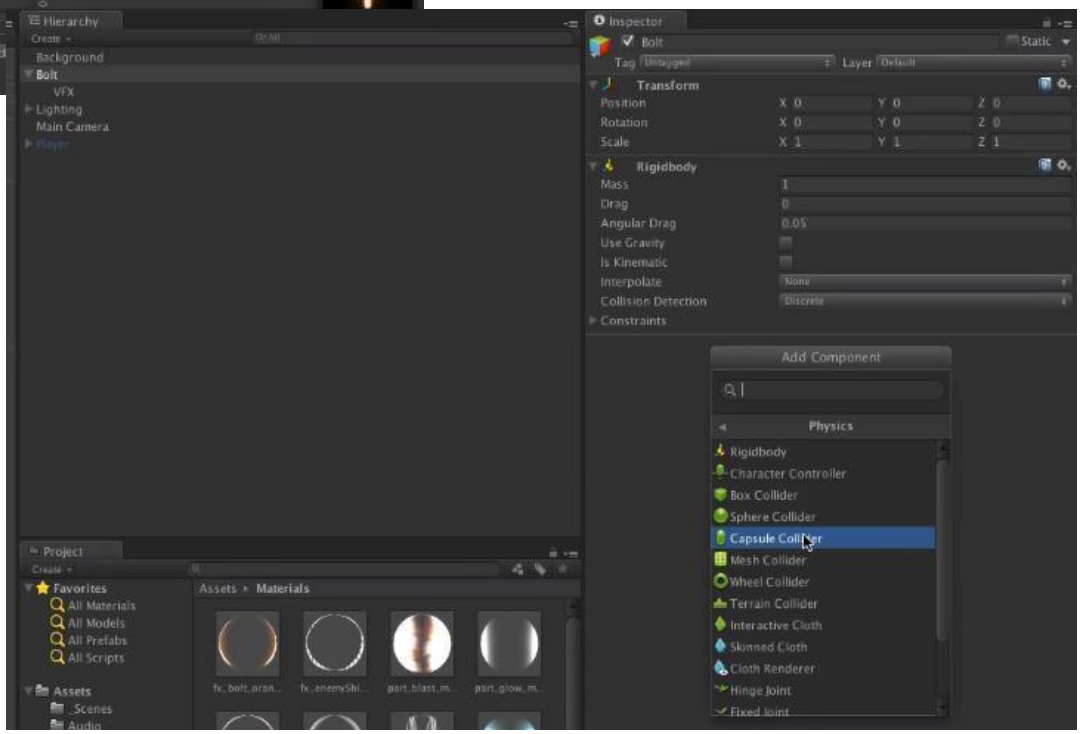


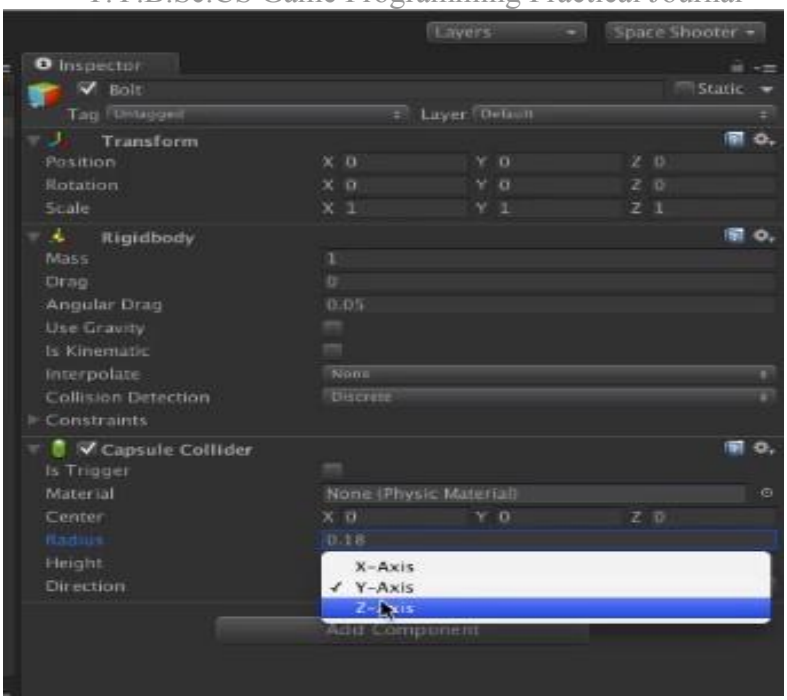
In Inspector Window->Go to mesh Collider->Click on Setting Button->Click on Remove Component



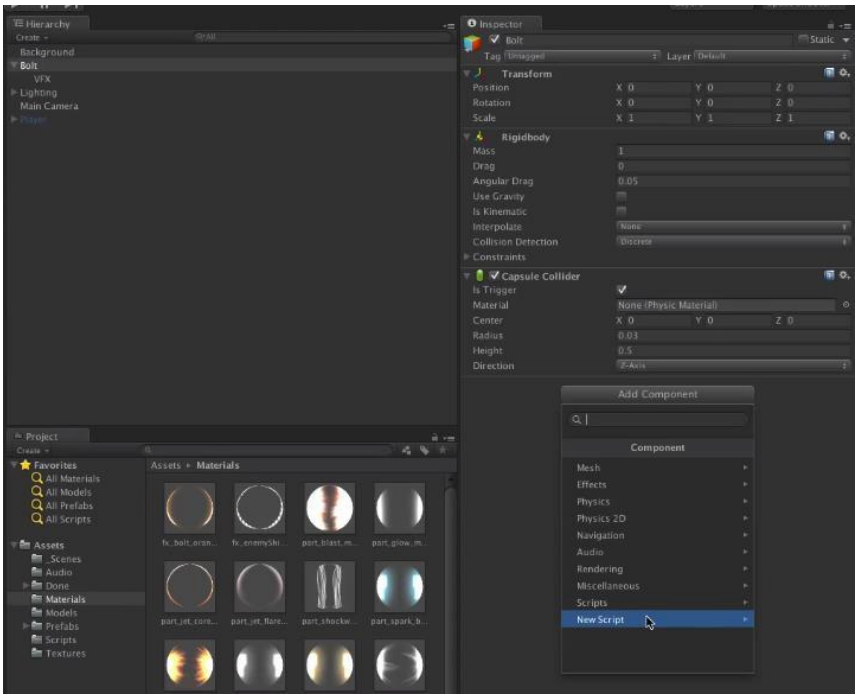
In the hierarchy Window->click on Bolt->Go to inspector Window->click on Add Component Button->Select Physics->Capsule Collider

Go to Capsule Collider Tab in to Inspector Window->Change the radius ad 0.03->Height 0.58->Direction asZ- axis.

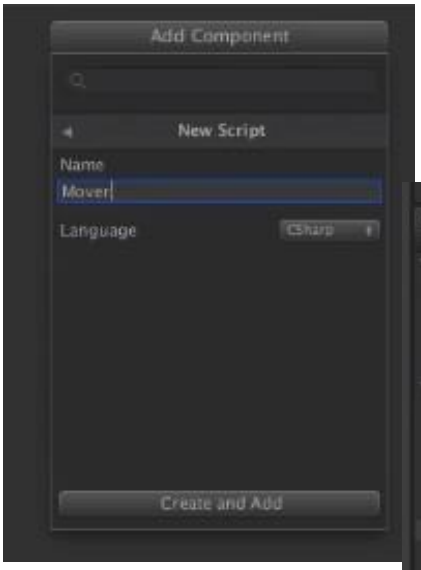




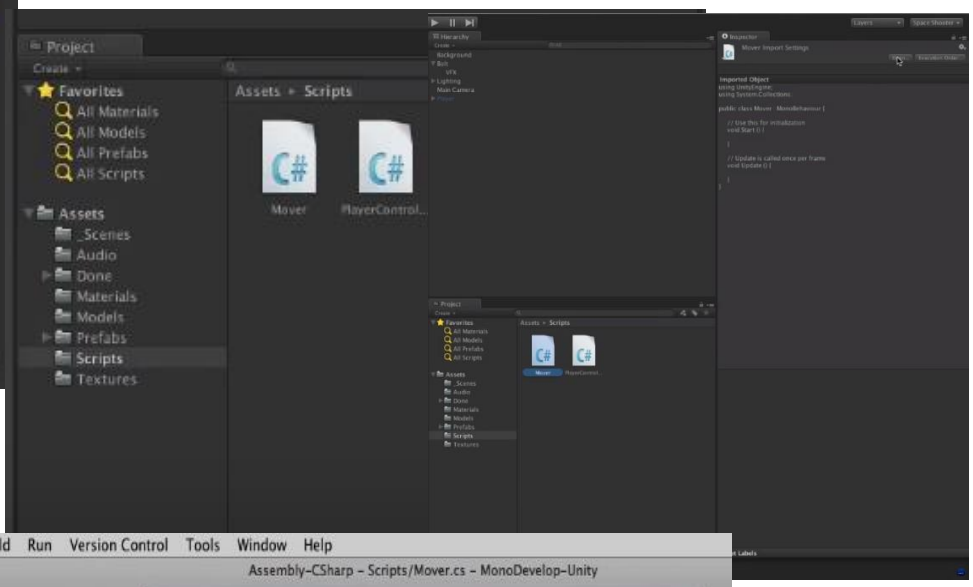
Click On Is Trigger tab in Collide

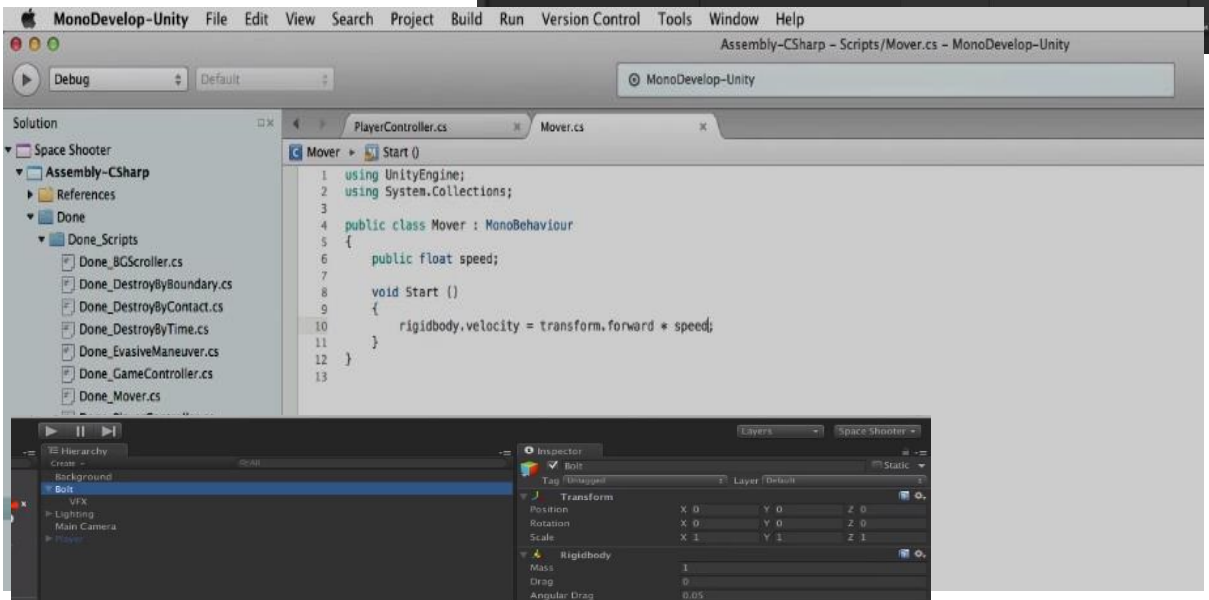


Click on Bolt In the hierarchy->Go to Inspector Window->Click On Add Component->Click on New Script->Give The Name to the Script As Mover->Press Enter.



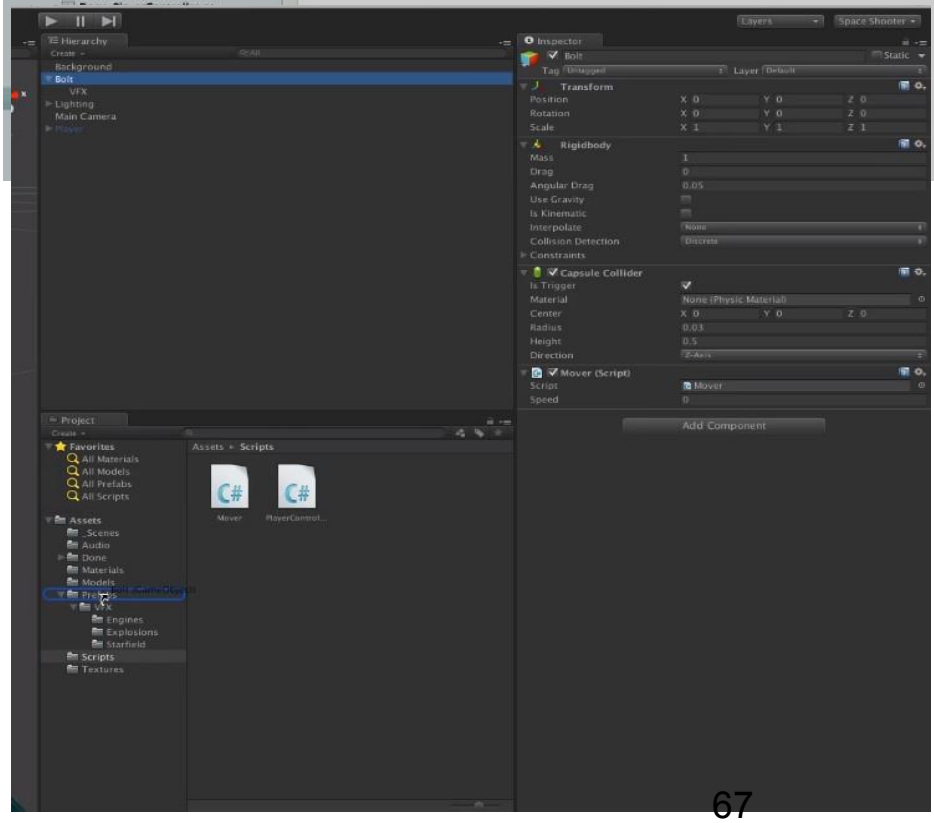
Go toAssets-> Move the Script File into The Script Folder->open the Mover Script->





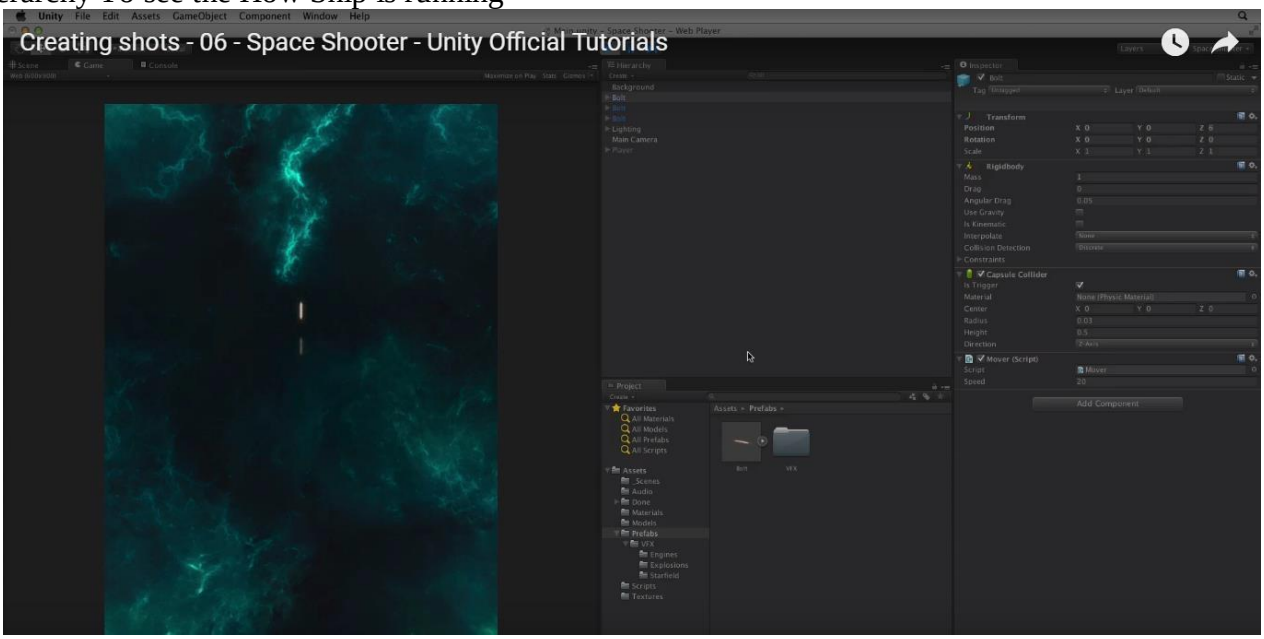
Write the following Lines of Code

into the Mov er Scrip t->Sav e the Scrip t And Com e to The unity Wind



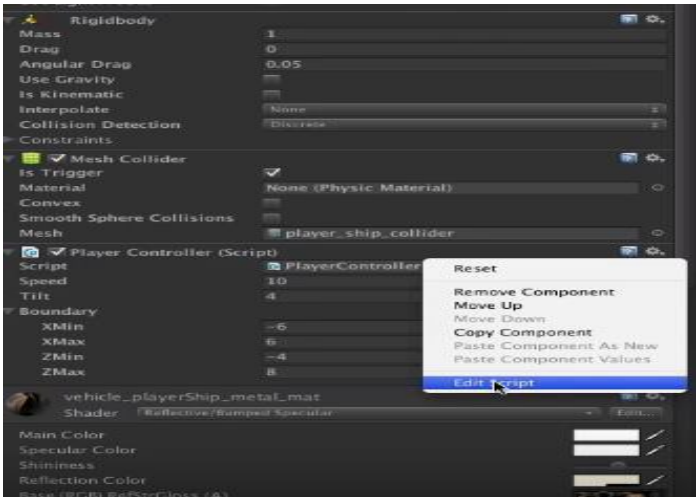
Drag the Bolt Game Object From Hierarchy to the Asset Prefab->Set the Script Speed As 20 Inspector Window.

Turn Of the Maximize on play Button on the Scene->Click on the Play Button->Drag the Bold Into the Hierarchy To see the How Ship is running

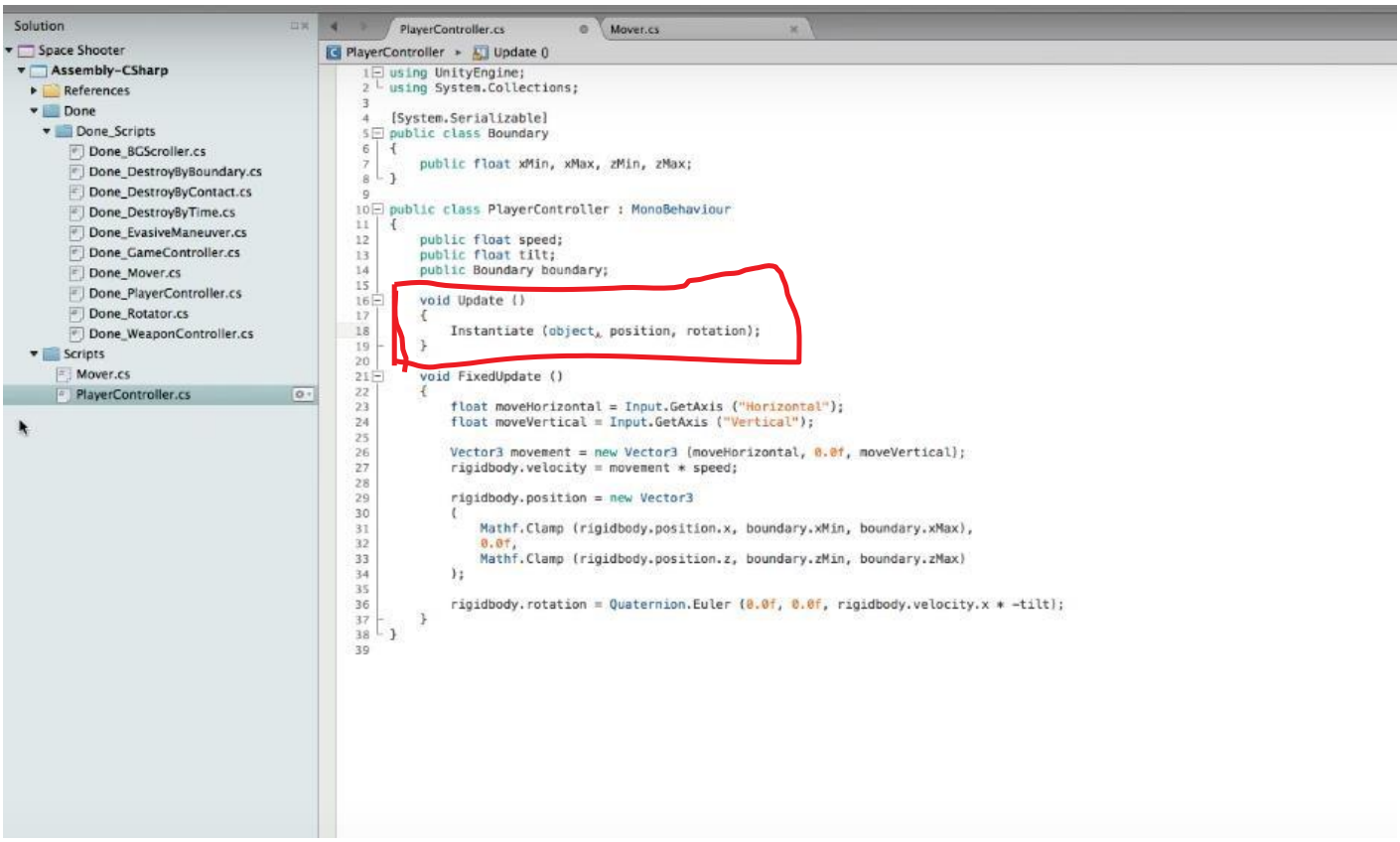


Step 6:**Creating Shots**

Select Player From Hierarchy->go to Inspector Window->Go to Player Controller Script->Click on Setting Tab->Select Edit Script Option

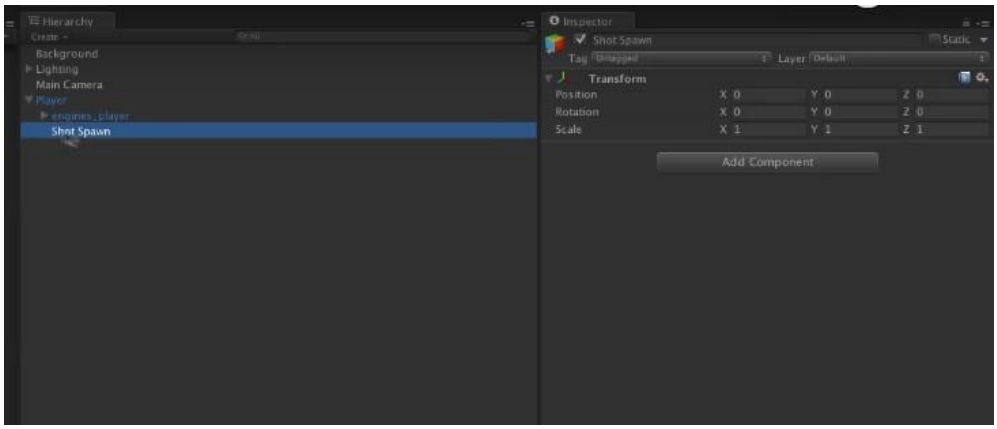


Add The Following Lines Of Code Into the Script

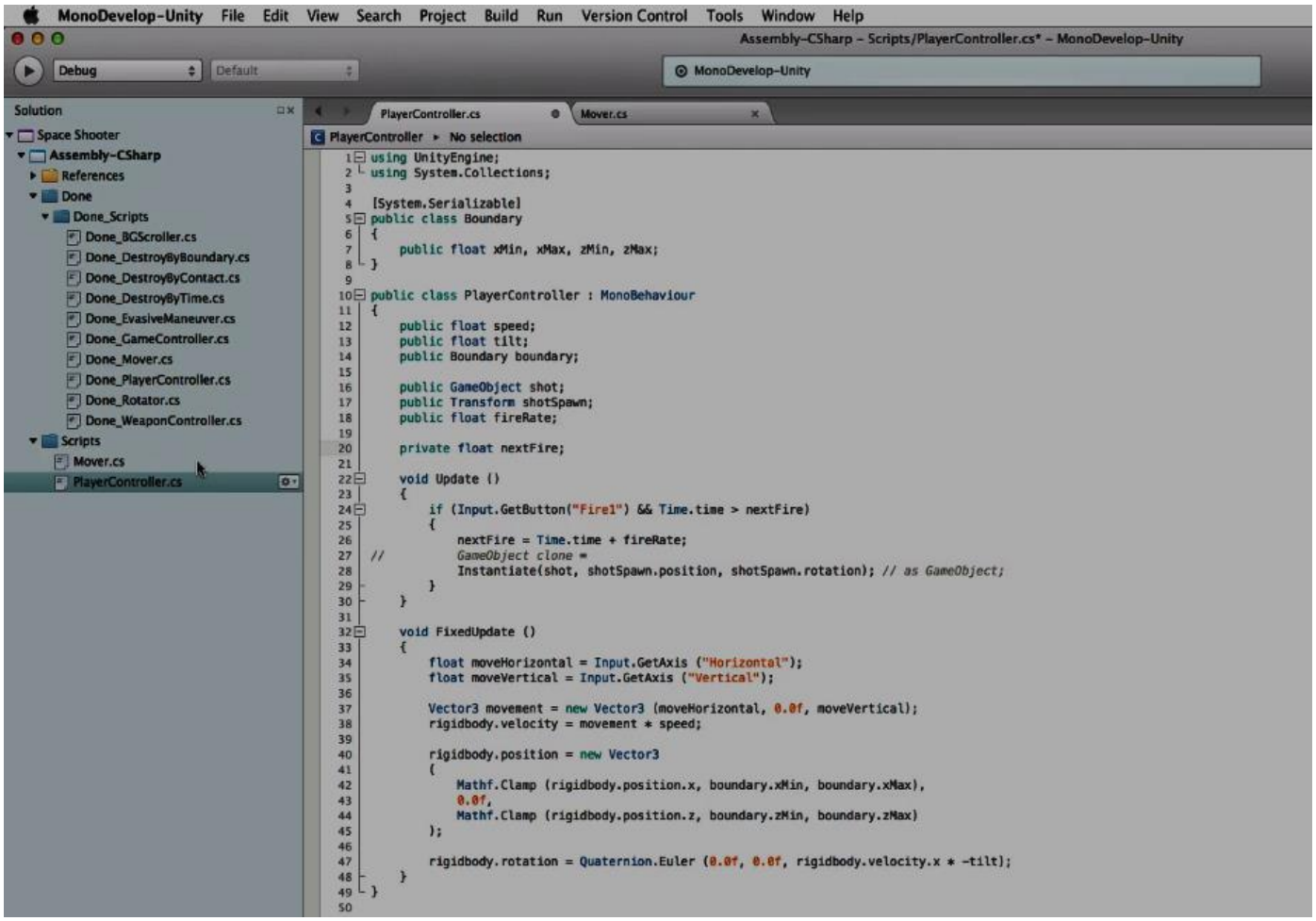


Create New Empty Game Object->Name it as Shot Spawn

Drag the Shot Spawn Game Object Into The Player



Now go to Player Controller Script And add some lines Of Code.




```
using UnityEngine;
using System.Collections;

[System.Serializable]
public class Boundary
{
    public float xMin, xMax, zMin, zMax;
}

public class PlayerController : MonoBehaviour
{
    public float speed;
    public float tilt;
    public Boundary boundary;
```

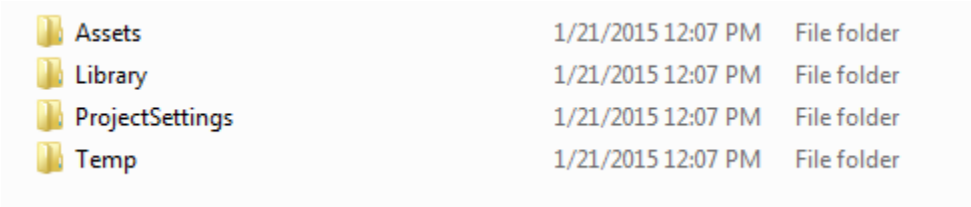
```
public GameObject shot; public
Transform shotSpawn; public float
fireRate;

private float nextFire;

void Update ()
{
    if (Input.GetButton("Fire1") && Time.time > nextFire)
    {
        nextFire = Time.time + fireRate;
        Instantiate(shot, shotSpawn.position, shotSpawn.rotation);
    }
}

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

    Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical); rigidbody.velocity
    = movement * speed;

    rigidbody.position = new Vector3 (
        Mathf.Clamp (rigidbody.position.x, boundary.xMin, boundary.xMax), 0.0f,
        Mathf.Clamp (rigidbody.position.z, boundary.zMin, boundary.zMax)
    );

    rigidbody.rotation = Quaternion.Euler (0.0f, 0.0f, rigidbody.velocity.x * -
tilt);
}
```

Practical no .9

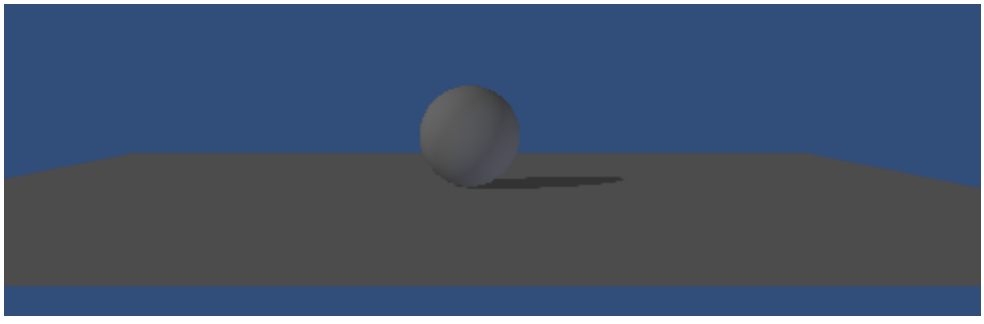
Aim :- Create a simple rolling ball game that teaches you many of the principles of working with Unity.

Setting up the Game

1. File-> New Project (Set the destination wherever you want, I recommend creating a folder called lab1)
2. Unity 3D will setup the following folders for you in your project.



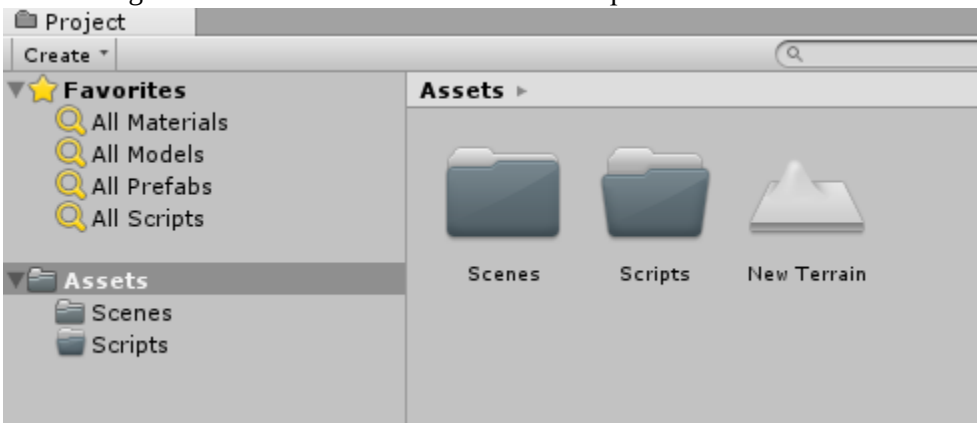
3. Next do a File->Save Scene, and create a new folder in the Assets called "Scenes", and save the file as "scene1"
4. Now lets create a plane, as our surface to work on. We can either use a terrain, or a plane. I am going to choose to do a terrain.
    a. We then will want to rename the terrain. We can double click the object, and then in the hierarchy anel on the left, rename the object from 'terrain' t 'GameTerrain'
    b. You will also notice, there are handles on the terrain. We can move it around, and transform I I t his way, or otherwise use the inspector panel, by changing the scale.
5. Now lets create our game object, which will be a 'sphere'. Once again, do GameObject->Sphere
    a. Rename the object 'Player'
    b. If you cannot find your object, highlight it in the hierarchy and press 'f' to focus on that object.
    c. Set the objects position to 0,1,0
6. Add a directional light now, so our scene can have some light. GameObject->Light->Directional light.
    a. Set the intensity to 0.1.
    b. Set the position to 0,5,0
    c. Set angle to 30,60,0
    d. Shadow Type -> Soft Shadows
    e. Resolution -> Very High Resolution
7. Add one more light by duplicating(Ctrl+D) our current Directional Light
    a. Rename it to fill light
    b. Shadowtype-> No Shadows
    c. Adjust the color to a blue-ish tint.
    d. Reverse the angles, so it should be -30,-60,0, so that the light shines up.



Save your scene.

Move the player

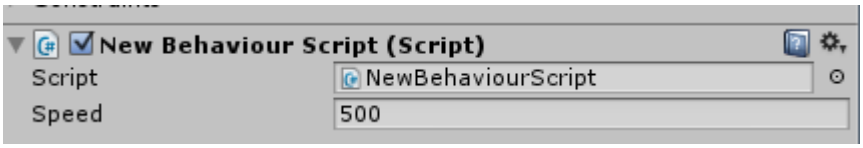
1. Select the Player. You will see in the inspector a way to "Add Component" Select Physics->Rigid Body
2. Now we are going to move the player by manipulating the physics forces. We will write a script to do this. Lets get organized though, and create a scripts folder. Go down to the Project Panel, and then right-click create new folder. Name it 'scripts'.



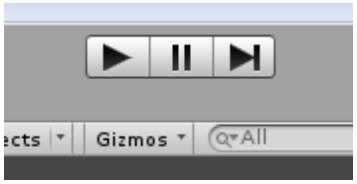
3. We then will 'add component' on our object again, which will be a C# Script. This will attach it to the object.
4. Double-click on the script, and then MonoDevelop will be loaded. MonoDevelop is Unity's primary scripting engine.
    a. You will notice in your script 2 methods.
        i. Start() – Which is what happens during initialization (like a constructor)
        ii. Update() – What happens every frame of the game on this object.
    b. If we want we can delete Start, as we will not need it, or otherwise just leave it for later.
5. Start by typing the word 'input' in the 'Update()' method.
    a. If you then hold "Ctrl" and press the single quote key ', then help will be brought up for you. (Note if you are on Mac, Ctrl is replaced with Cmd)
    b. Here is the final script

```
public class NewBehaviourScript : MonoBehaviour {

    public float speed;

    // Update is called once per frame
    void Update () {
        // Record input from our player.
        float moveHorizontal = Input.GetAxis ("Horizontal");
        float moveVertical = Input.GetAxis ("Vertical");

        // Use the input we get to move the rigid body.
        Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical);
        // Make a smooth movement by multiplying by Time.deltaTime.
        // Adjust our 'movement' vector by 'speed' to make game playable.
        rigidbody.AddForce (movement * speed * Time.deltaTime);
    }
}
```

c. Save your script in MonoDevelop. Unity is constantly compiling, so it will be ready to go.
d. Return t Unity3D. Now is a good time to save your Unity Scene. (In fact, you should constantly be saving!)
e. Notice if we select our 'Player' game object, it will have a property for speed. Lets set it to 500.



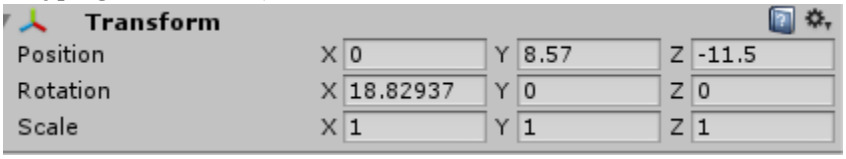
f. Test your game out by pressing the play button.



The Camera

Let us create a new camera, so that it is tied to the player.

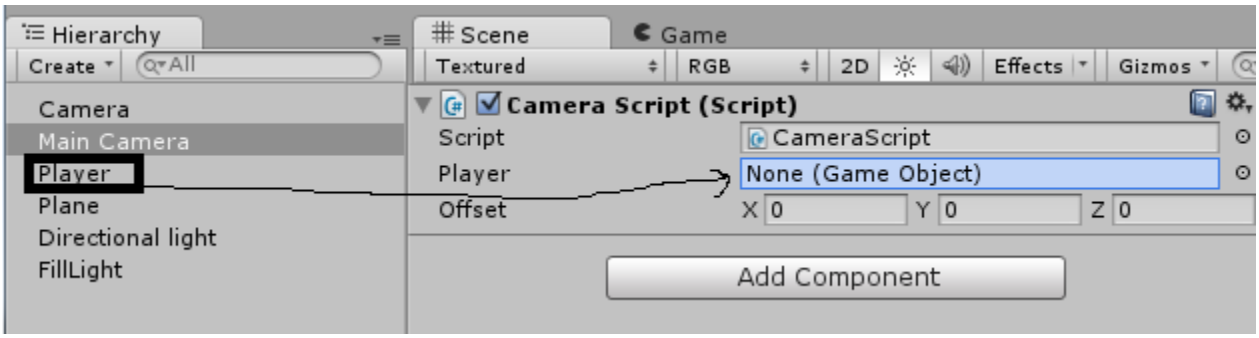
1. Game Object -> Camera (You can replicate my settings roughly by dragging around the camera, or typing them in here)



2. We wil then write a script (Add Component->Script->CSharp, and be sure to move the script into the Scripts folder)

```
public class CameraScript : MonoBehaviour {

    // Reference the player
    public GameObject player;
    // Position of our camera
    public Vector3 offset;

    void Start(){
        // Where we moved the camera in our scene.
        offset = transform.position;
    }

    // LateUpdate is called once per frame
    // But at the end of a frame.
    void LateUpdate () {
        // Our initial positin, plus the updated player position.
        transform.position = player.transform.position + offset;
    }
}
```

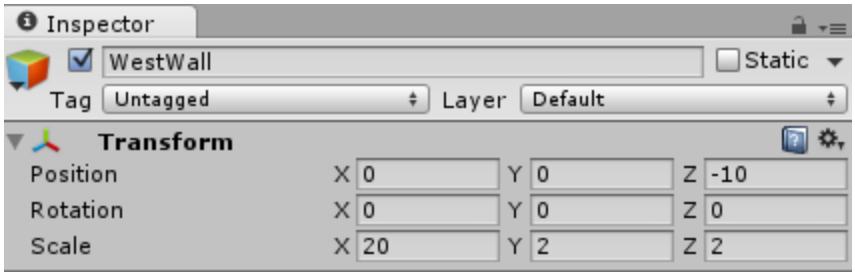
3. Drag the Player into the Camera Scripts's property for player. This creates a reference to that GameObject(in this case, our Sphere) GameObject in our script.



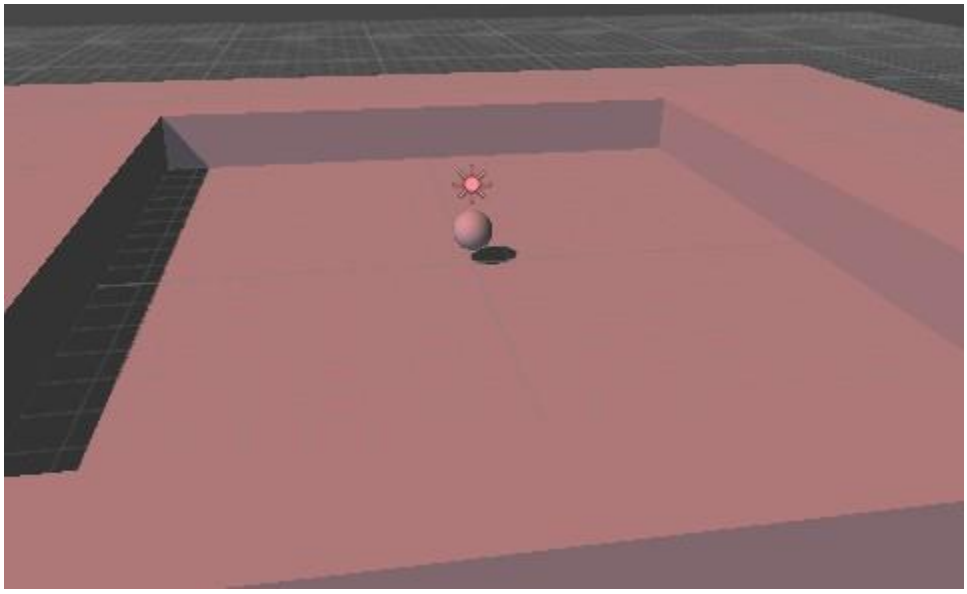
Create the Level

1. Create a new GameObject called Walls in our hierarchy. We will then create four cubes underneath it, and scale them to build walls.

2. The scale of the West Wall for example is this:

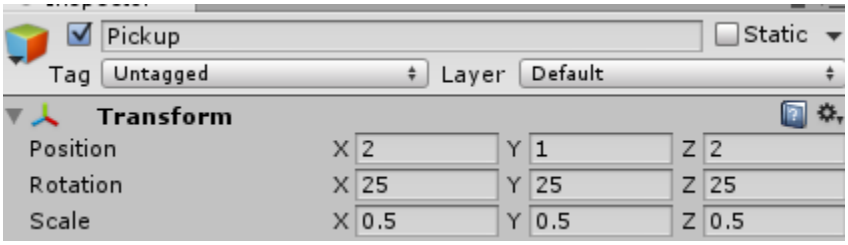


3. Your scene should roughly look like this



Create Items

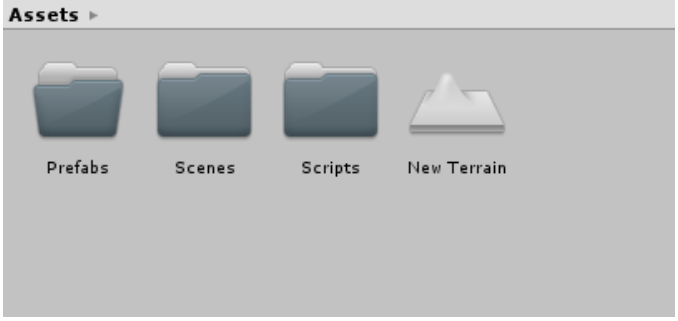
1. Create a new GameObject that is a cube called 'pickup'
2. Scale it appropriately



3. Lets now create a script called 'RotatingCube' for the pickup object to make it spin. Lets now open it up in MonoDevelop
4. Here is the final script.

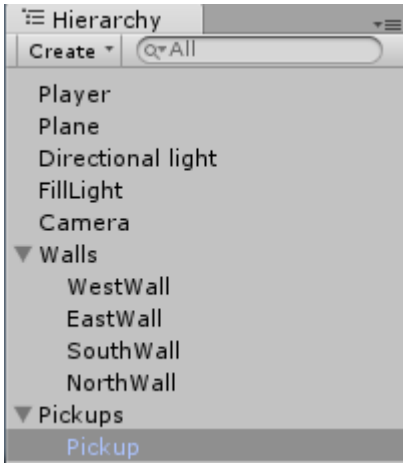
```
public class RotatingCube : MonoBehaviour {

    // Update is called once per frame
    void Update () {
        transform.Rotate (new Vector3 (15, 30, 45) * Time.deltaTime);
    }
}
```

5.
6. Now that we have created an object, lets create some more of them. We are going to make these into 'prefabs'. A 'prefab' is a reusable asset that acts as a blueprint for an object. We can use it in any scene in our project once it is created!
7. Lets create a new folder in our hierarchy called 'prefabs' to do this.



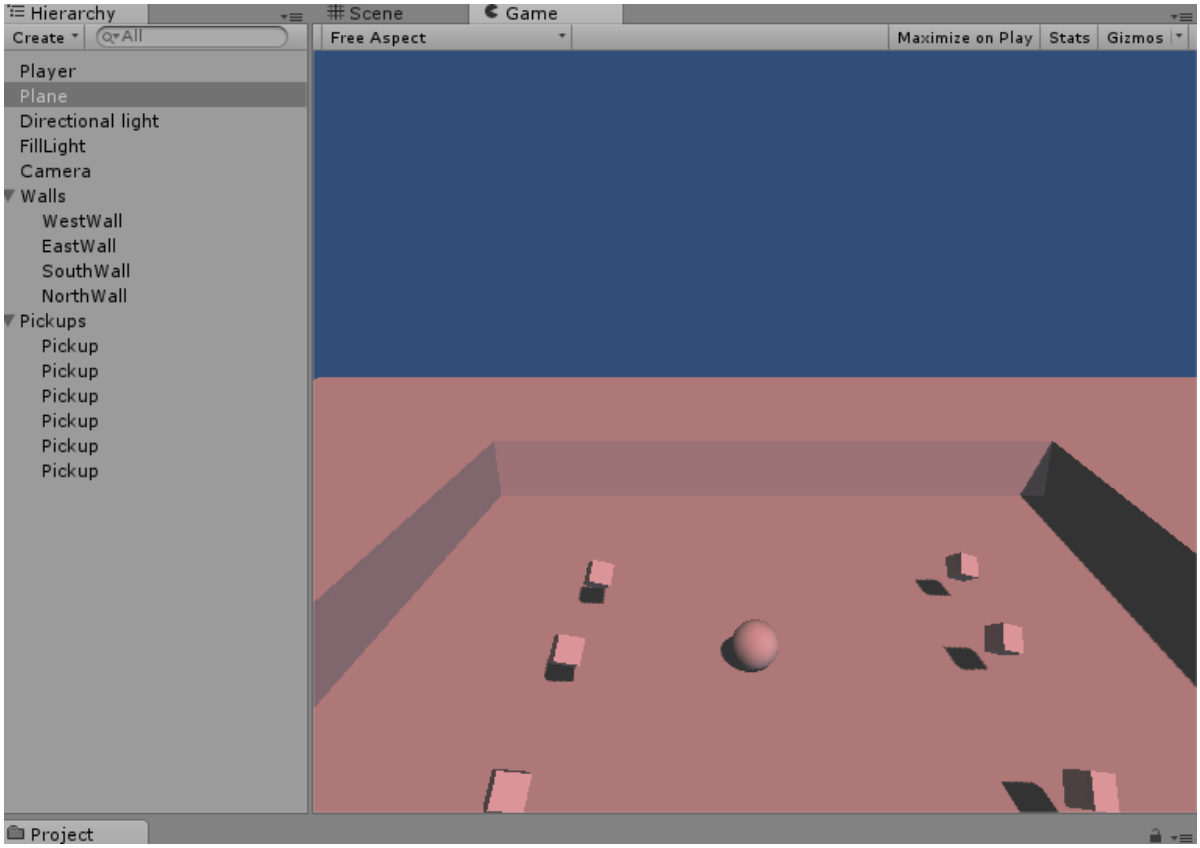
8.
9. Then drag our object into the folder from our object hierarchy. And that' it! You will notice our object is now highlighted in 'blue' in the hierarchy to indicate that it is a prefab.

9.
10. Now that it is a prefab, lets create an empty GameObject called 'pickups'. Position this game object to the origin. Now move our pickup into the object, so that it is a child.



11. Now we want to duplicate this object, and move it around. To move it around in a global coordinate space, toggle the coordinate space to global.

12. Now we manipulate the object, in regards to the world, instead of its own rotation, scale, and position.
13. Create several of them by duplicating them (note, since they are prefabs, they will already have scripts attached to them and be ready to go!)
14. You scene should look something like this:



15.

Collecting Objects

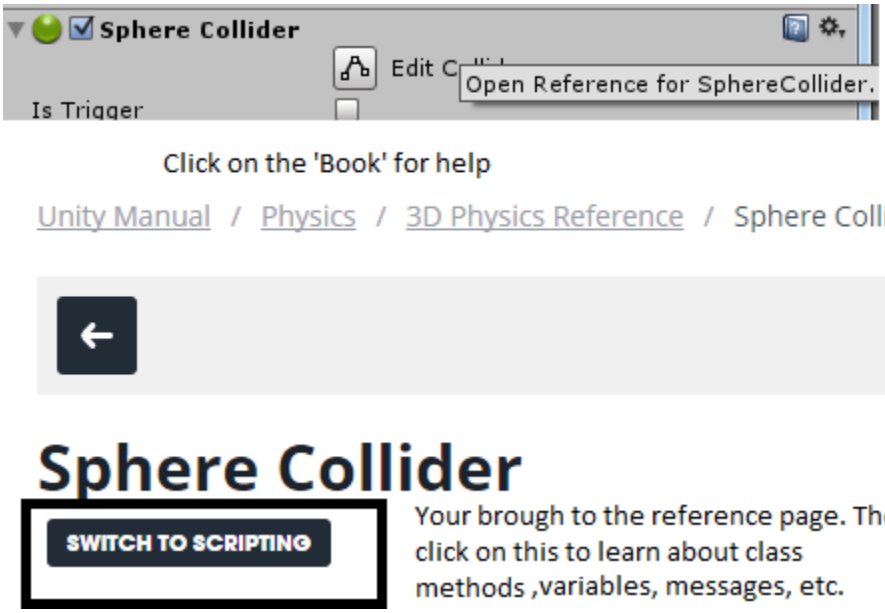
In order to win the game, the player will collide with the objects and pick them up. When they have picked up all the objects, the game is over. In order to do this, we will learn about colliding with the correct object, and picking it up.

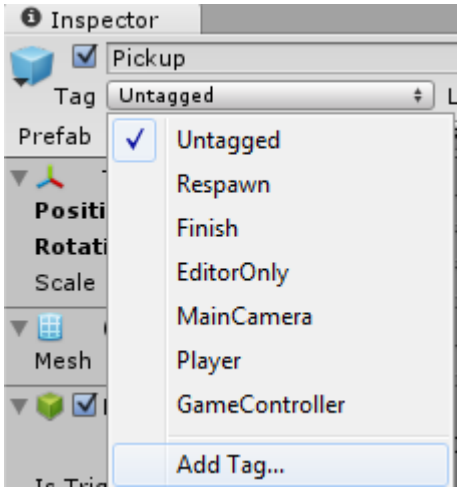
1. Open up our original player script that we wrote.
2. (Stepping back, this is how to learn about what methods are available)
3.
4. In our script, add the following method.

```
void OnTriggerEnter(Collider other){
    // Check if our object collides with a pickup object.
    if (other.gameObject.tag == "PickUp") {
        other.gameObject.SetActive(false);
    }
}
```

5.
6. Now we must 'tag' our object so that Unity knows about it. To do this, select one of our

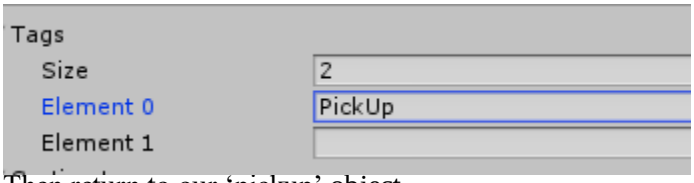
'pickup'objects and then find the 'Tag' item, and 'Add Tag



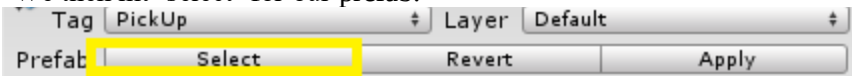


7.

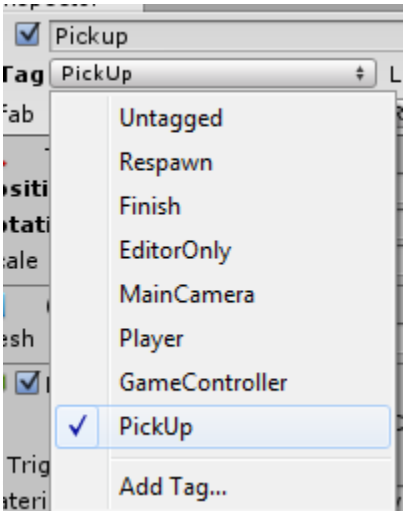
8. We then add the name of our tag (NOTE: This is case sensitive, and must exactly match what is in our code, because C# is a case sensitive language).



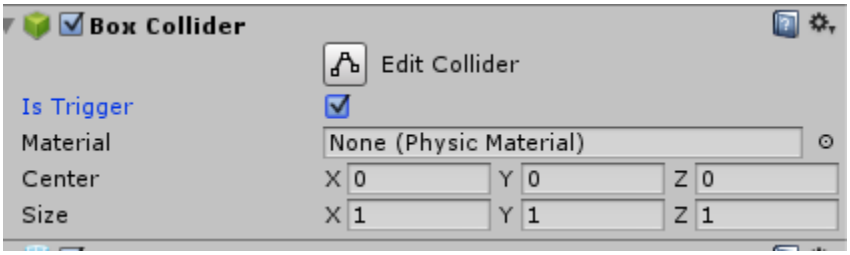
9.
10. Then return to our 'pickup' object
11. We then hit 'select' for our prefab.



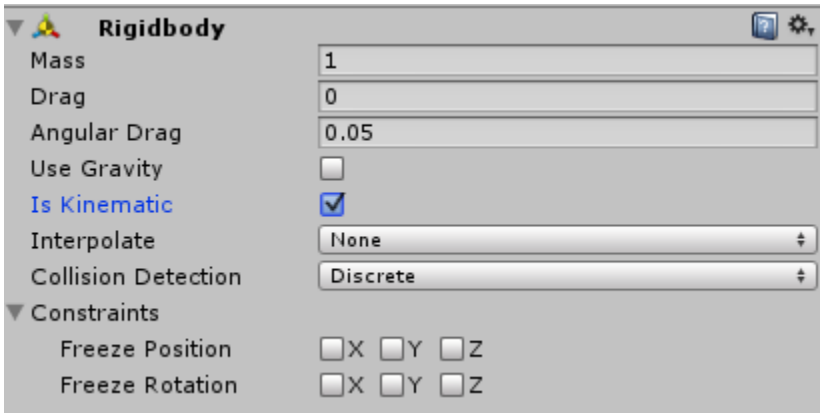
12.
13. This ensures we make changes to every game object that is of this type.
14. Finally we tag our object.



15.
16. then make sure that it is a trigger. We want our objects geometry to act as a trigger, as opposed to physical collision



17. Side note: Small performance optimization to add a rigid body for our pickup object.
    a. Is Kinematic makes objects ignore physics simulations.
    b. Rigid body avoids excess computation for every frame on dynamic objects.



Winning the Game

1. When we have picked up all of the objects, we will have won the game. Lets make a private variable in our player script that gets updated every time we collide with an object.
2. Final code:

```
public float speed;
private int count;

void Start(){
    count = 0;
}

// Update is called once per frame
void Update () {
    // Record input from our player.
    float moveHorizontal = Input.GetAxis ("Horizontal");
    float moveVertical = Input.GetAxis ("Vertical");

    // Use the input we get to move the rigid body.
    Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical);
    // Make a smooth movement by multiplying by Time.deltaTime.
    // Adjust our 'movement' vector by 'speed' to make game playable.
    rigidbody.AddForce (movement * speed * Time.deltaTime);
}

void OnTriggerEnter(Collider other){
    // Check if our object collides with a pickup object.
    if (other.gameObject.tag == "PickUp") {
        other.gameObject.SetActive(false);
        count += 1;
    }
}
```

3.Lets now display the code for our user to see how well they are doing.

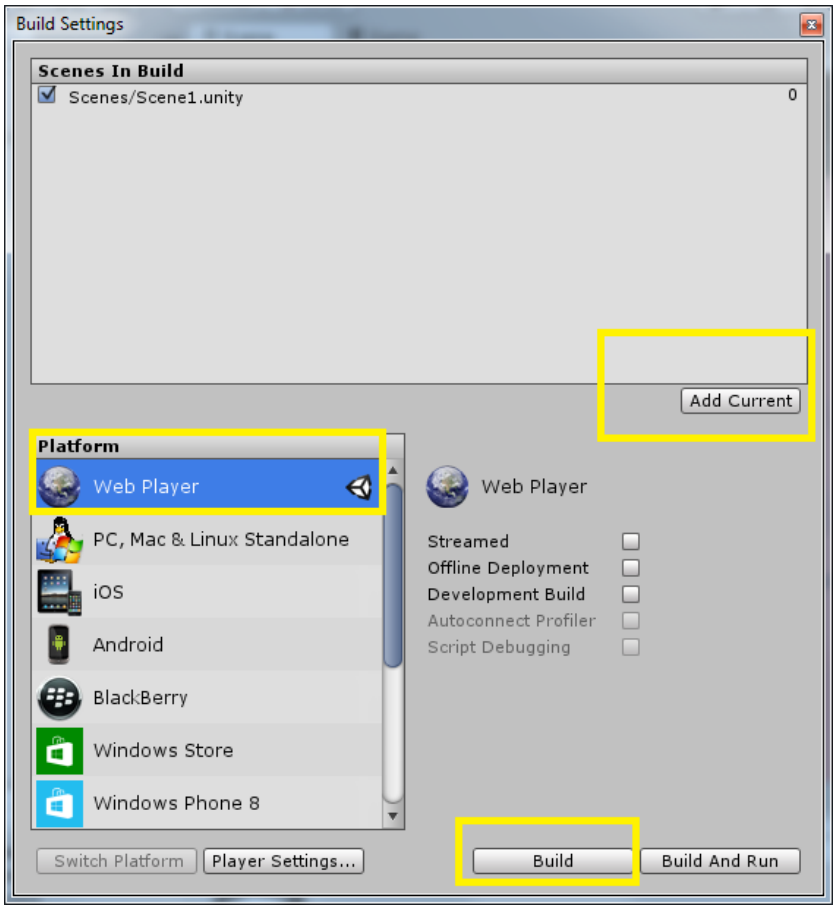
```
public float speed;
private int count;
public GUIText countText;

void Start(){
    count = 0;
    SetCountText ();
}

// Update is called once per frame
void Update () {
    // Record input from our player.
    float moveHorizontal = Input.GetAxis ("Horizontal");
    float moveVertical = Input.GetAxis ("Vertical");

    // Use the input we get to move the rigid body.
    Vector3 movement = new Vector3 (moveHorizontal, 0.0f, m
    // Make a smooth movement by multiplying by Time.deltaT
    // Adjust our 'movement' vector by 'speed' to make game
    rigidbody.AddForce (movement * speed * Time.deltaTime);
}

void OnTriggerEnter(Collider other){
    // Check if our object collides with a pickup object.
    if (other.gameObject.tag == "PickUp") {
        other.gameObject.SetActive(false);
        count += 1;
        SetCountText();
    }
}
```

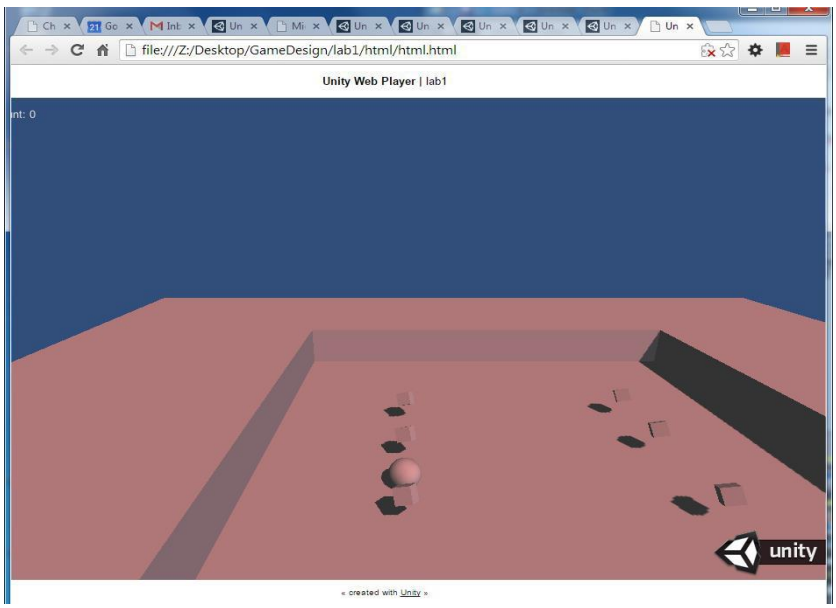
```
void SetCountText(){
    countText.text = "Count: " + count.ToString ();
}
```

```
void OnGUI() {
    if (count == 6) {
        GUI.Label ((new Rect (10, 10, 100, 20)), "WAY TO GO! YOU WIN!")
    } else {
        GUI.Label ((new Rect (10, 10, 100, 20)), countText);
    }

}
```

4.
5. SAVE all of your scripts and your Unity3D scene.
6. Play your game!!

For an Extra Cool Factor (Show your friends, export to the web!





Do go home and install the android SDK or if you're on a Mac, install the iOS XCode Tools and deploy

Practical no .10

Aim :- Creating AR Content with Vuforia.

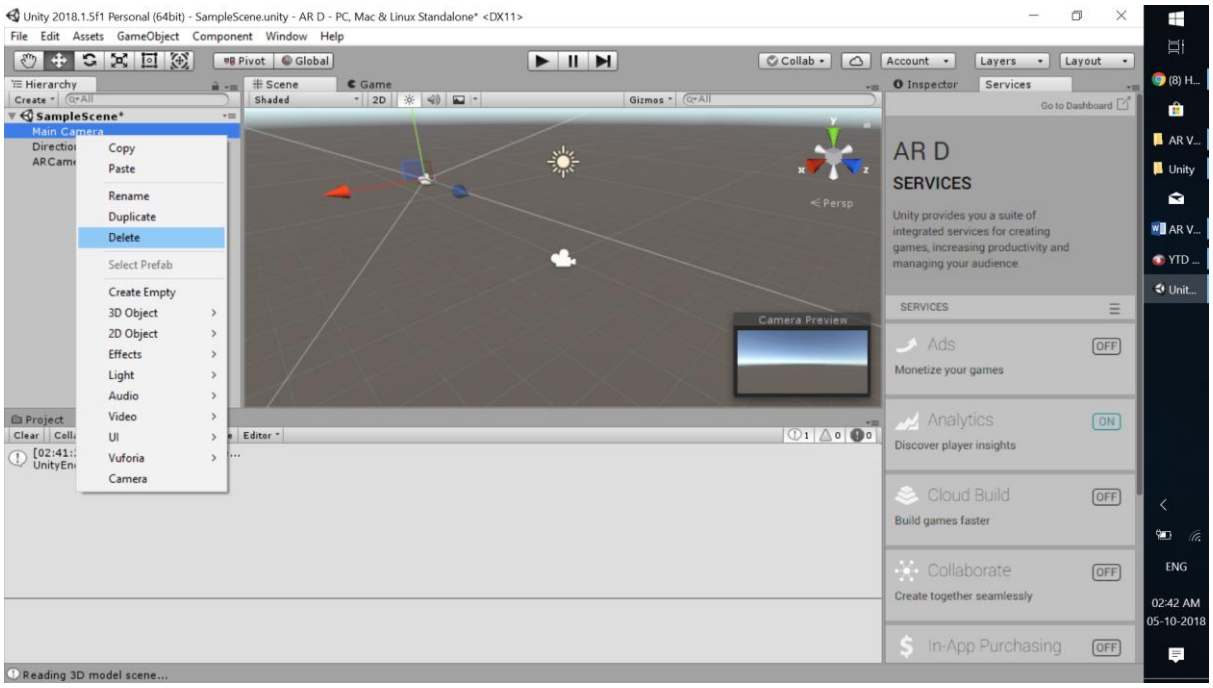
AR Last try

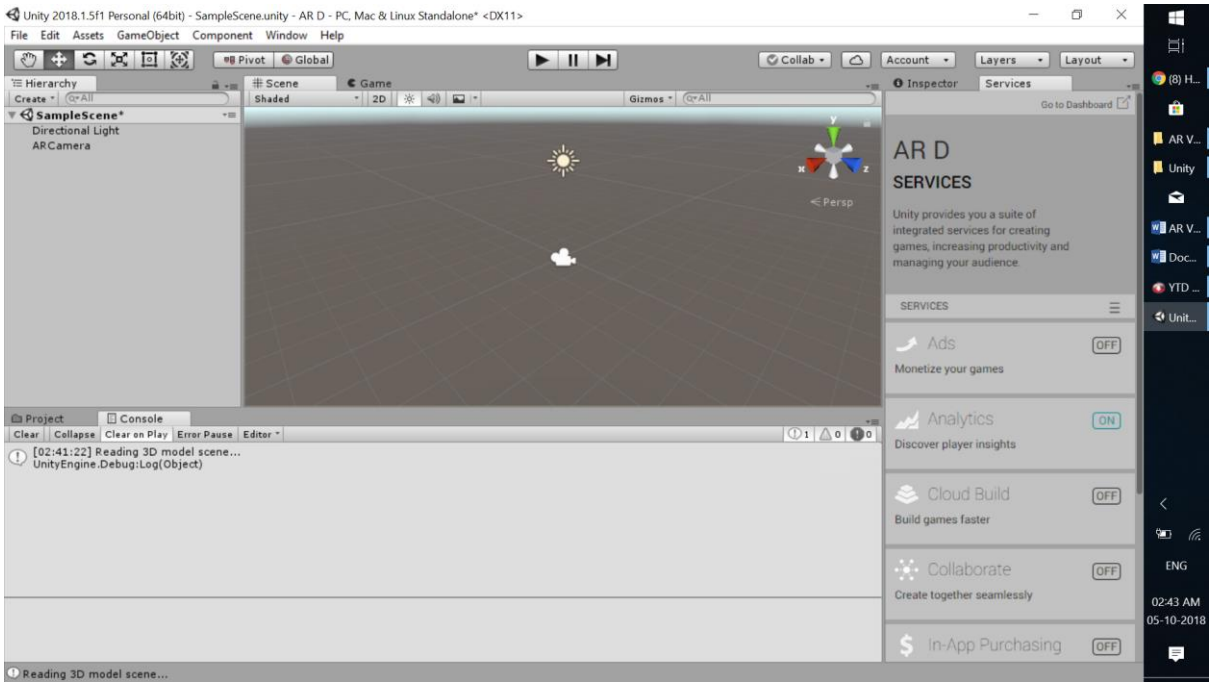
New project-3D

Create-Vuforia-AR Camera

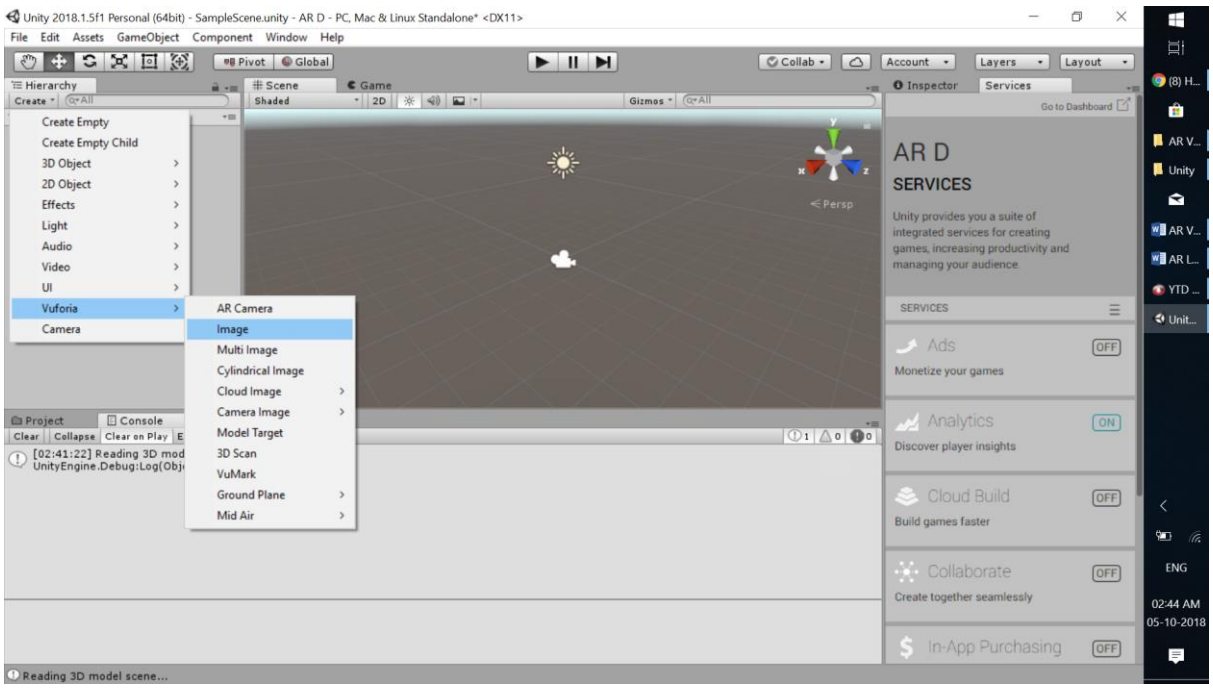
Needs some import- click on Yes

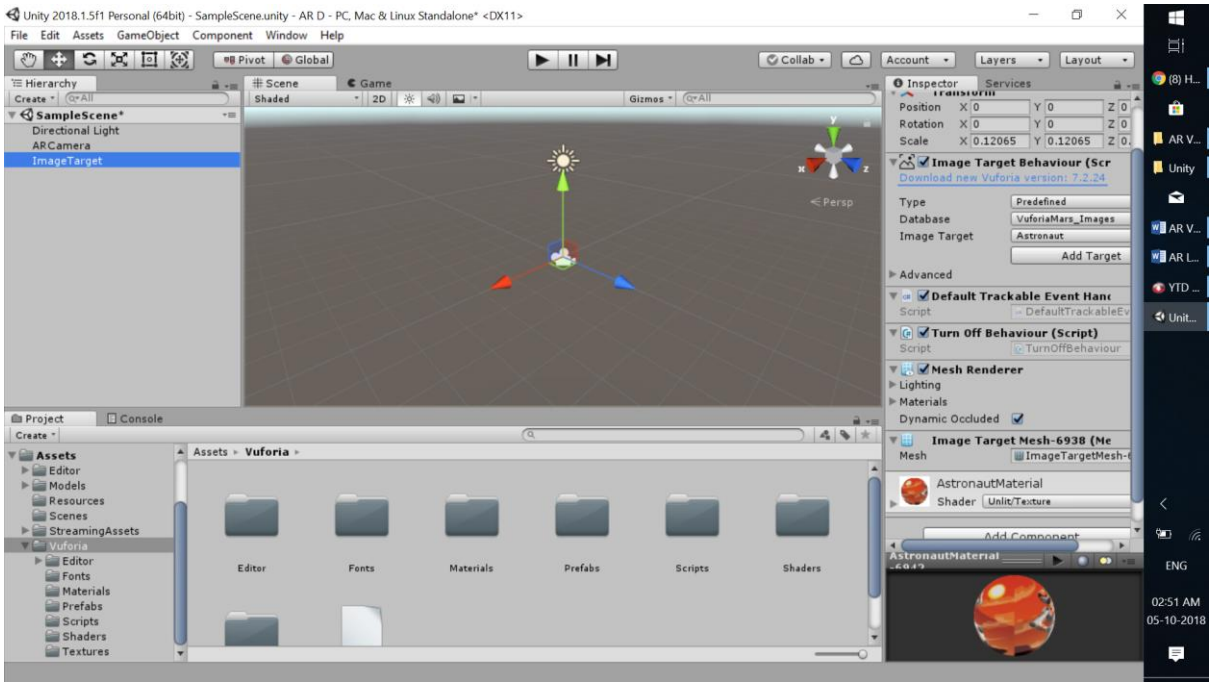
Delete the Main Camera





Now we create image

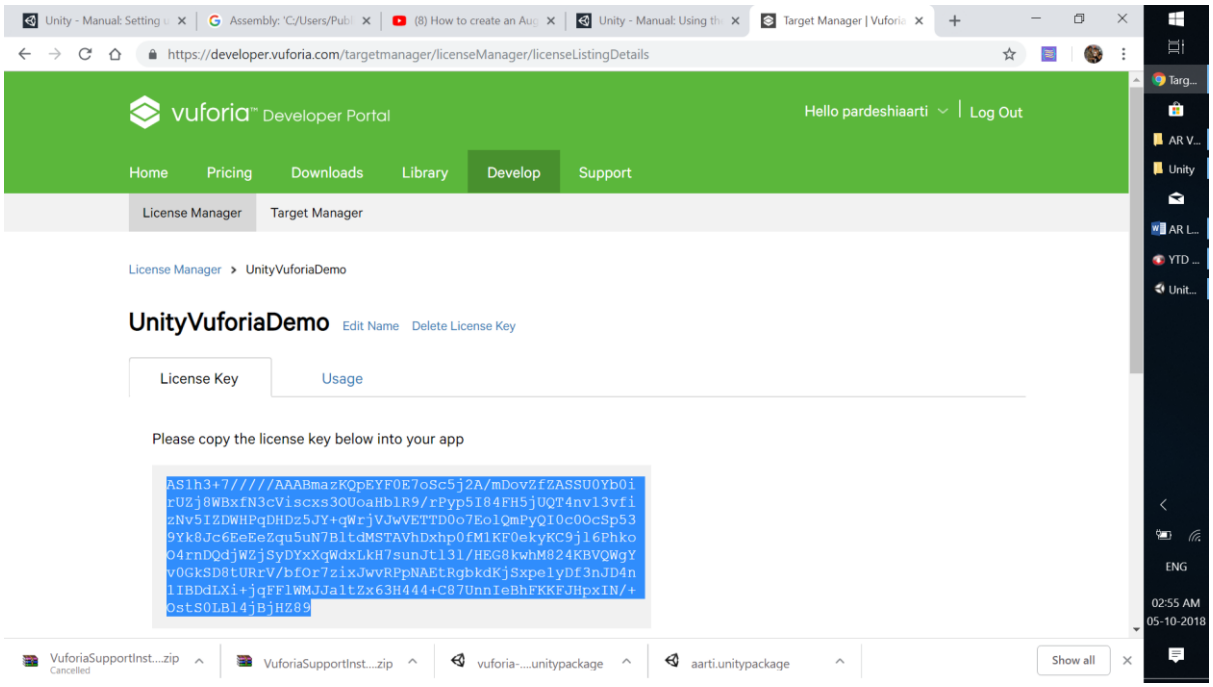




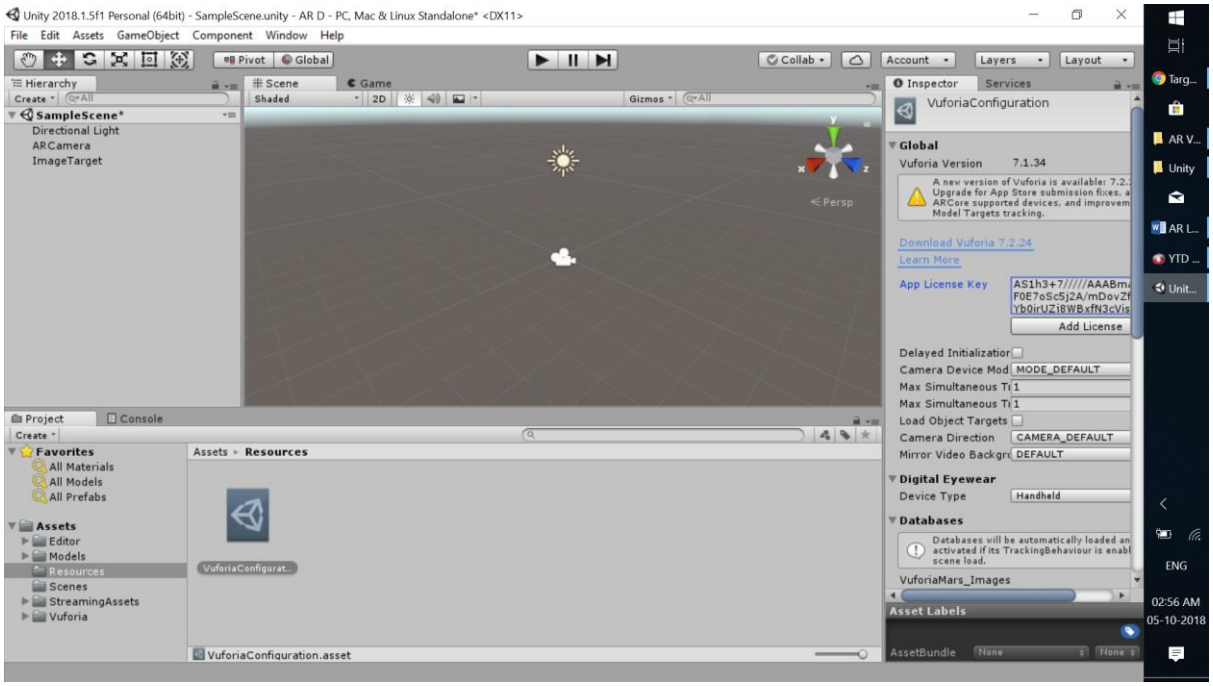
We will add our own target

Click on Add Target

Create develop liscen in Vuforia

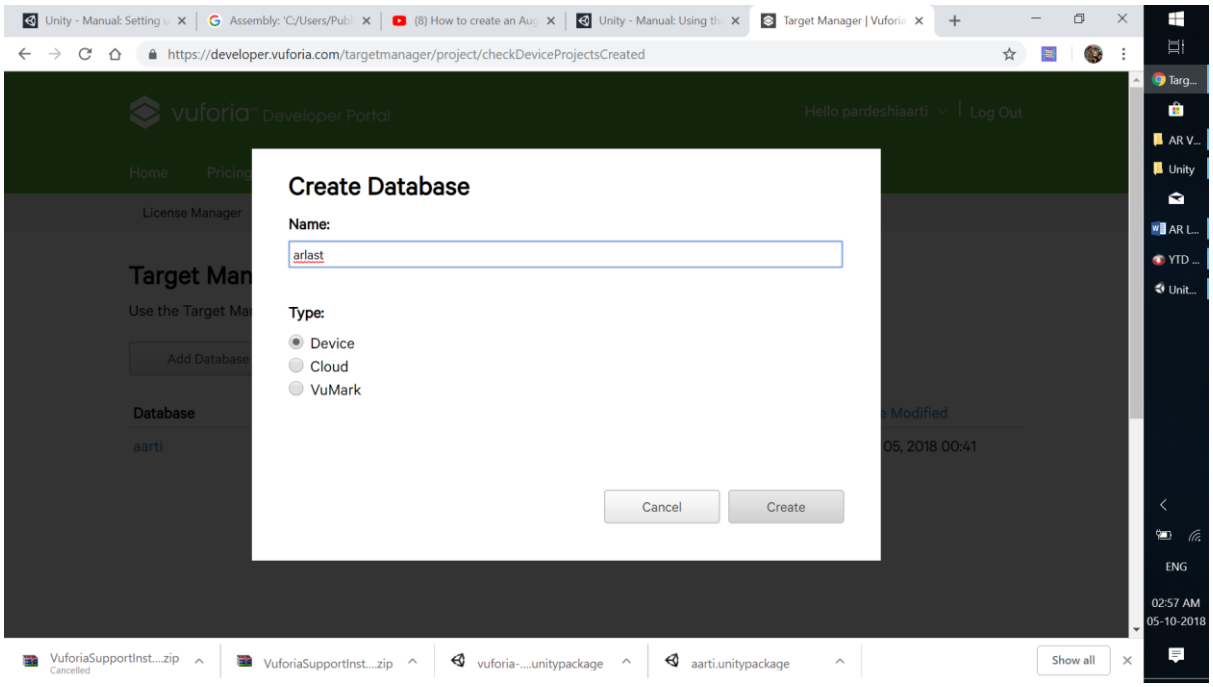


Copy and paste the key from here to Vuforia configuration



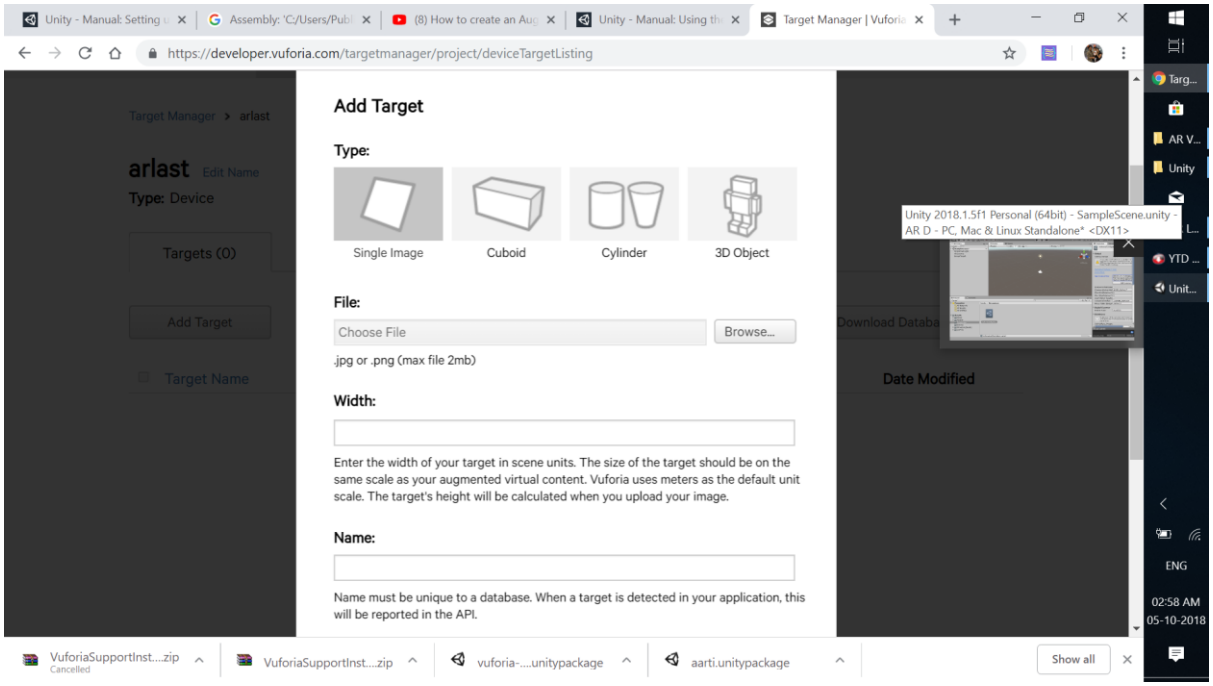
Now go back to Vuforia site

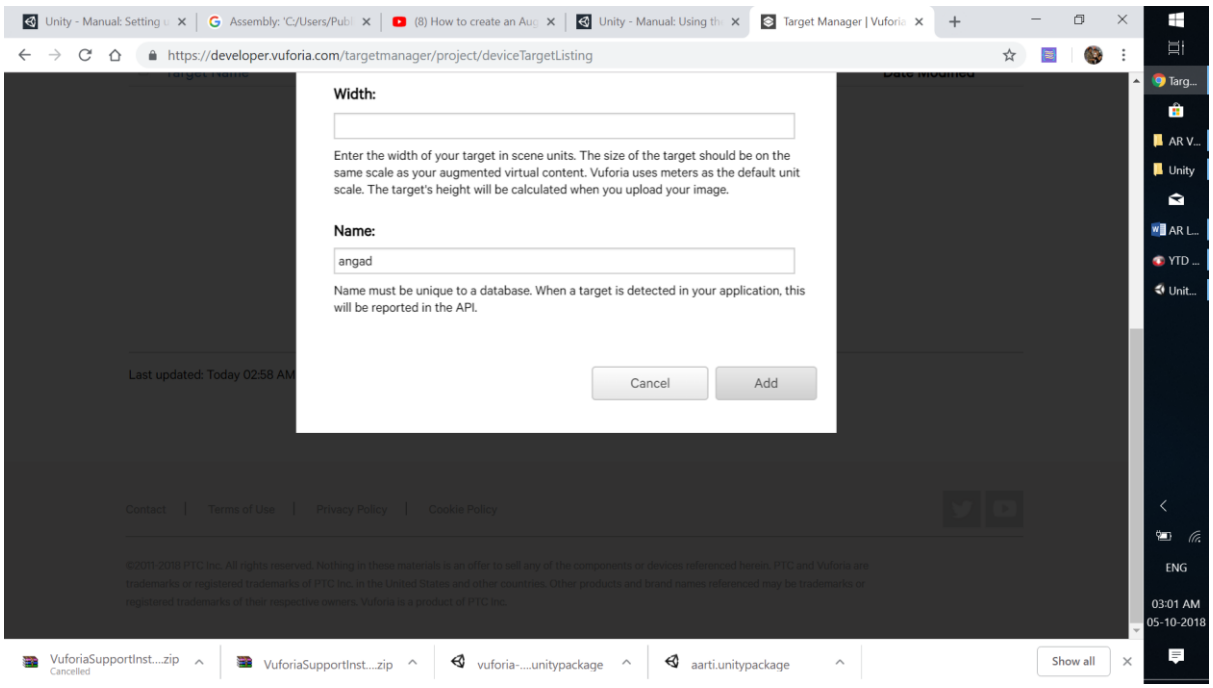
Click on target



Create

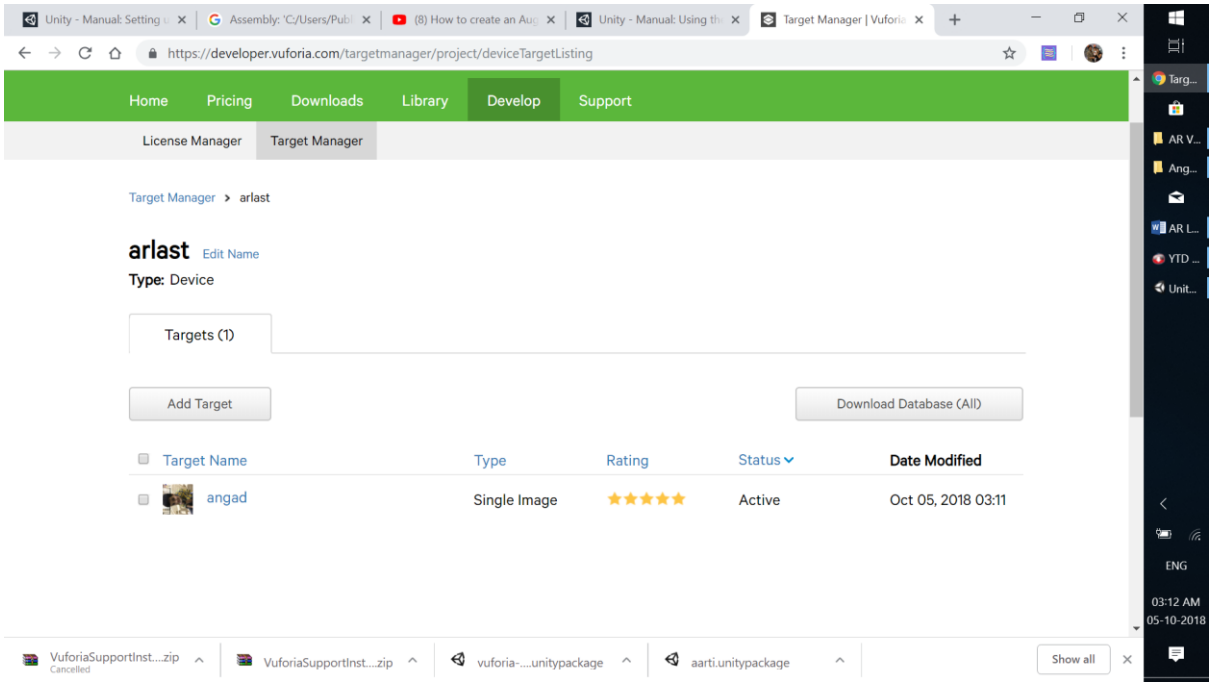
Click on arlast db- add target



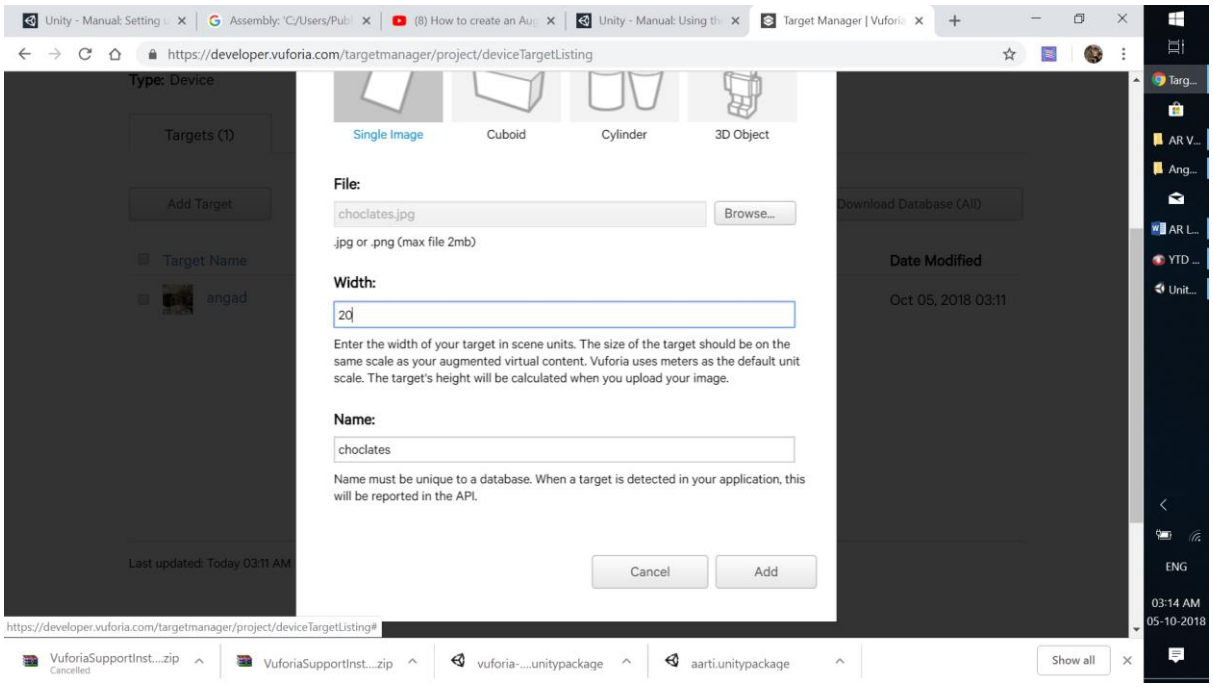


Width set to 10

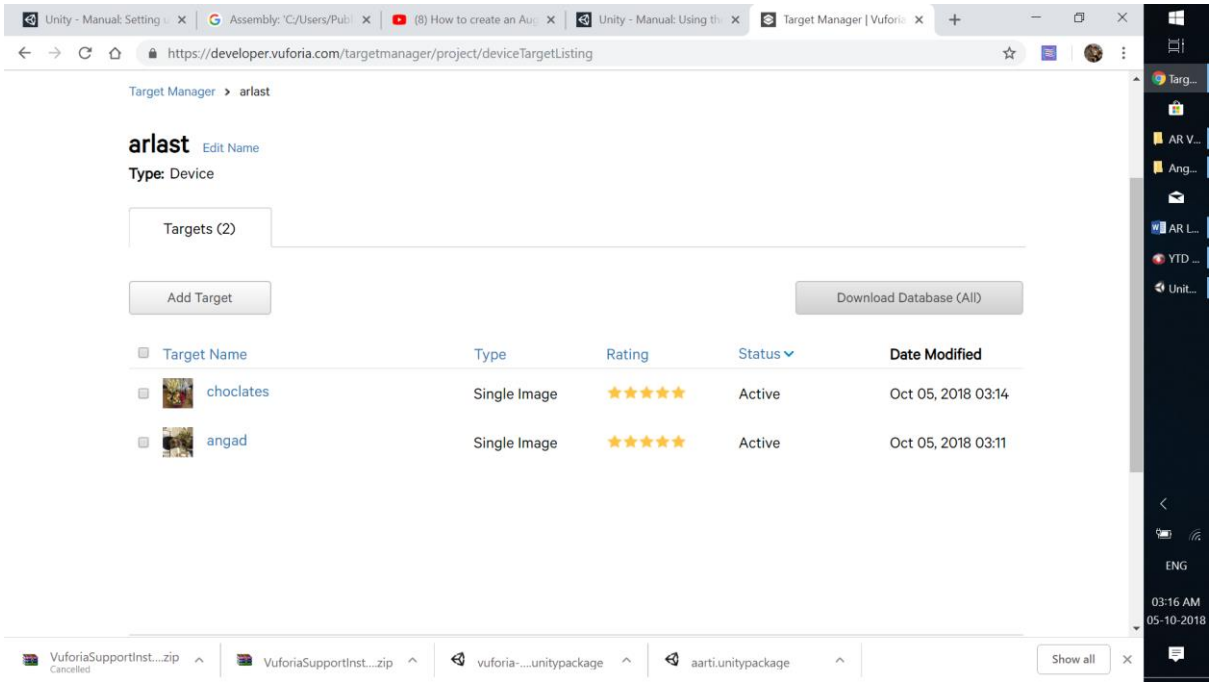
Click on add

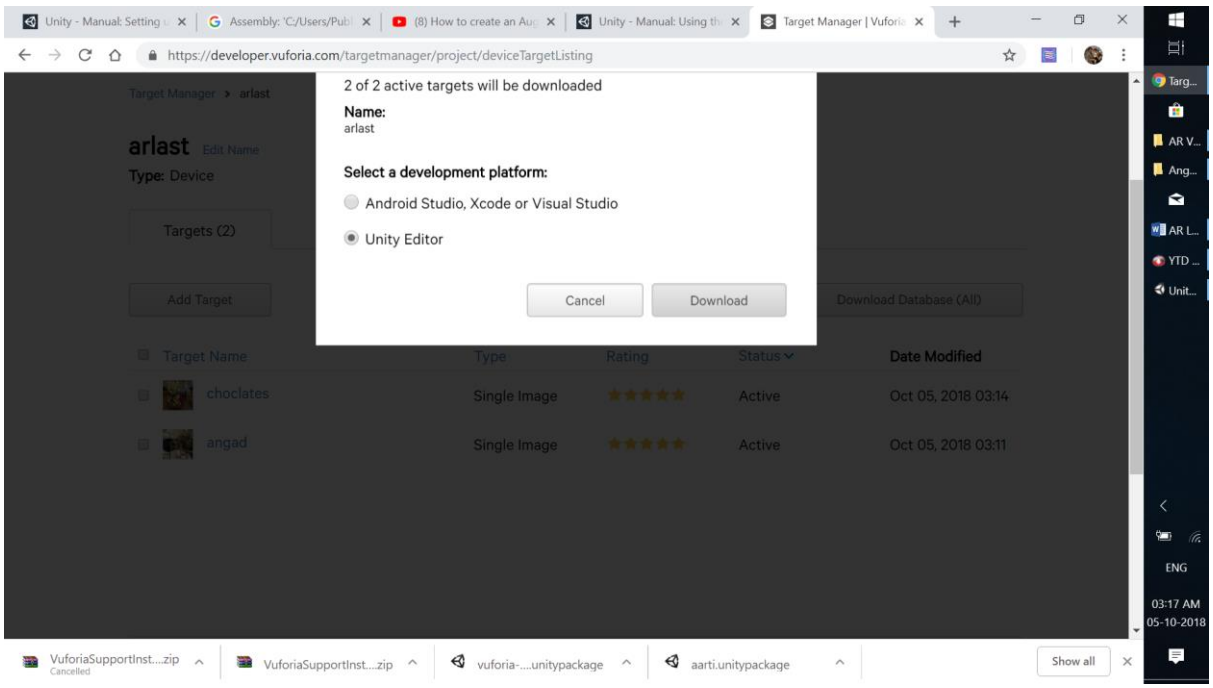


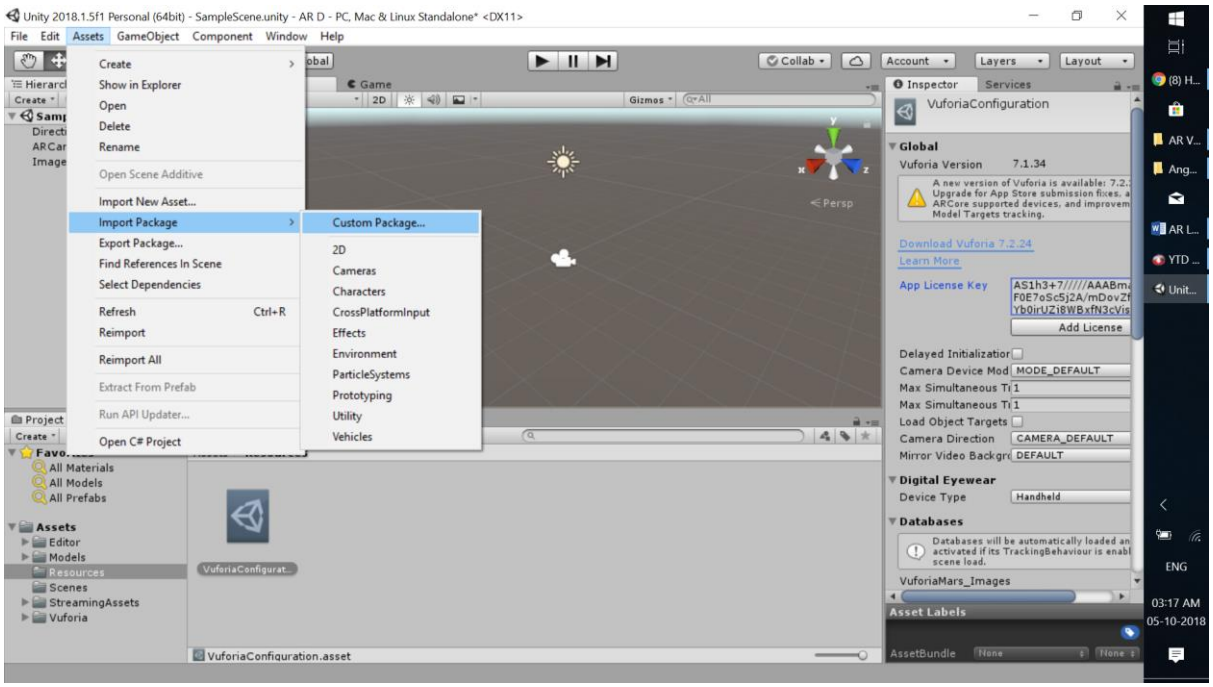
Check rating.

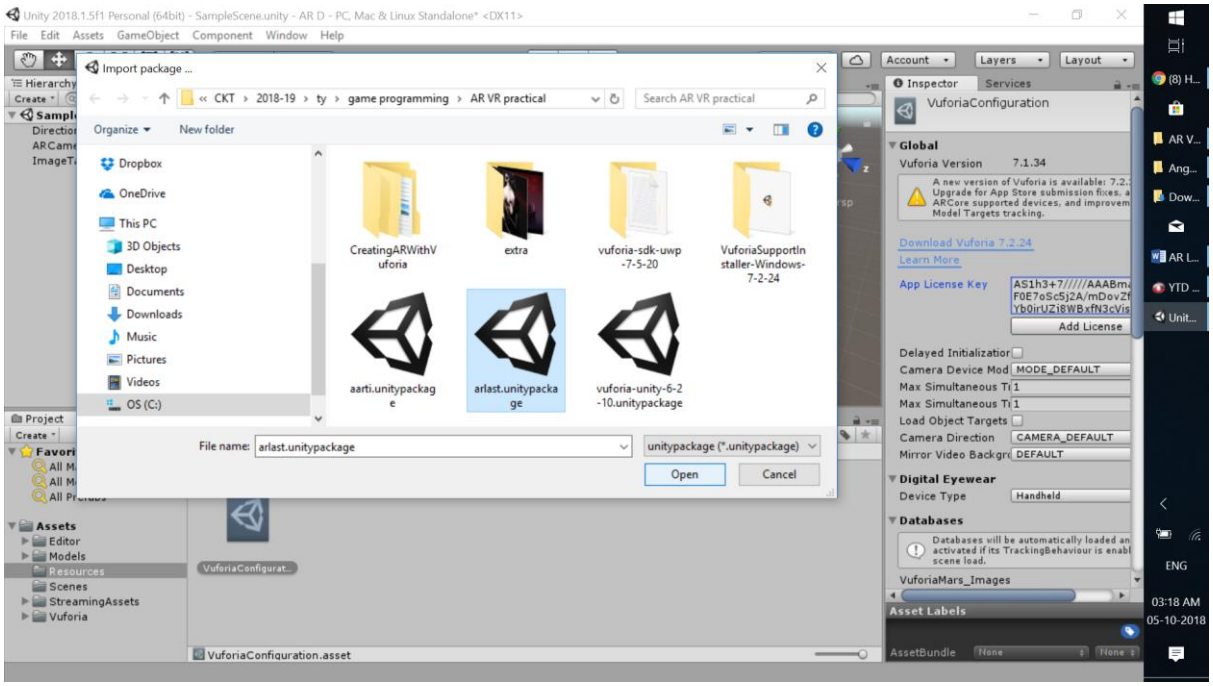


Click on Downloads all



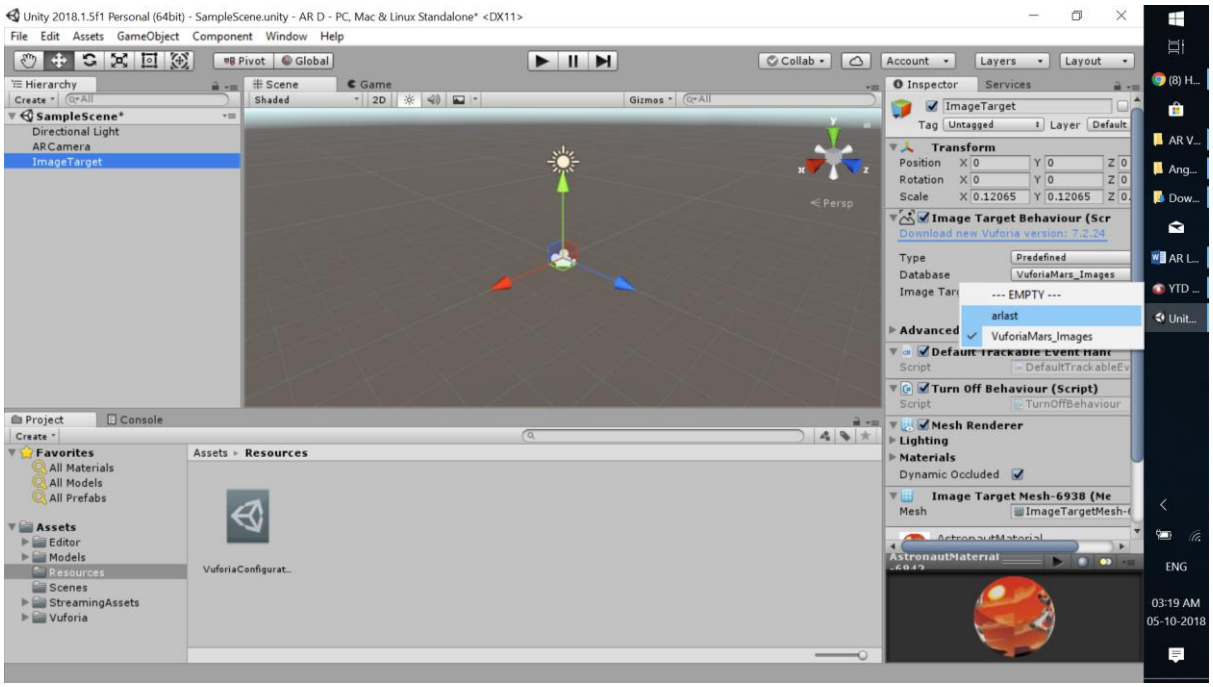

https://developer.vuforia.com/targetmanager/project/deviceTargetListing
2 of 2 active targets will be downloaded
Name:
arlast
Select a development platform:
Android Studio, Xcode or Visual Studio
Unity Editor
Cancel
Download
arlast
Type: Device
Targets (2)
Add Target
Download Database (All)
Target Name
Type
Rating
Status
Date Modified
choclates
Single Image
Active
Oct 05, 2018 03:14
angad
Single Image
Active
Oct 05, 2018 03:11
VuforiaSupportInst....zip
Cancelled
VuforiaSupportInst....zip
vuforia-....unitypackage
aarti.unitypackage
Show all
ENG
03:17 AM
05-10-2018

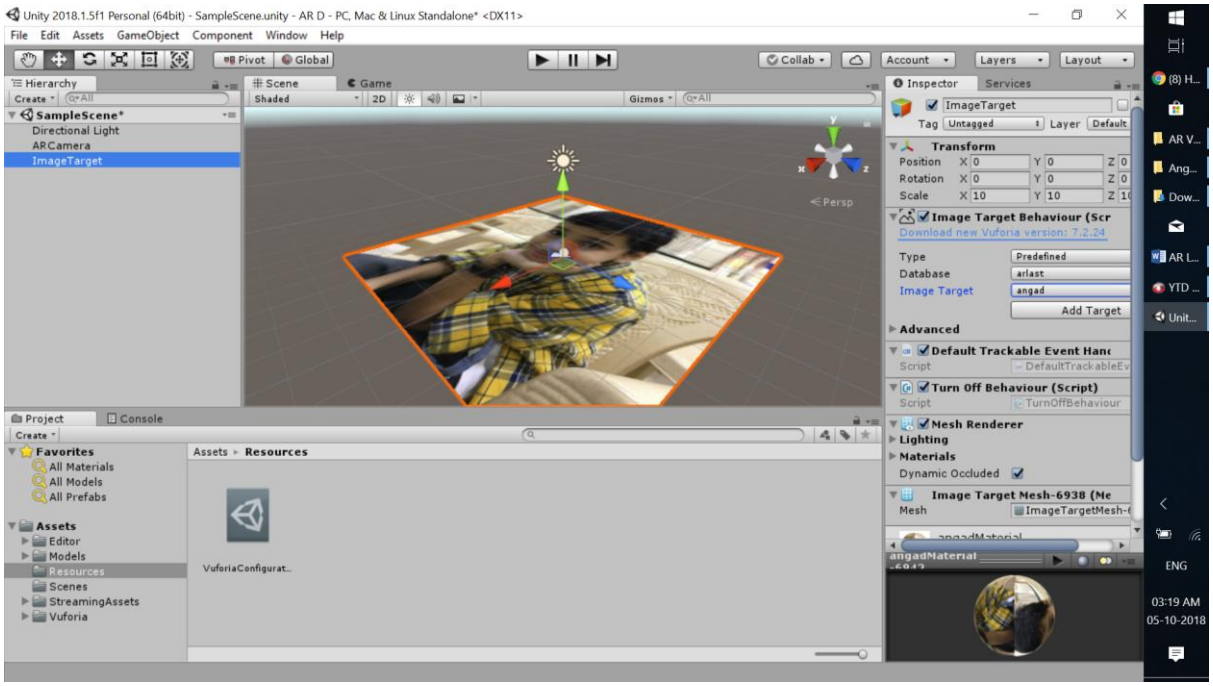

Unity 2018.1.5f1 Personal (64bit) - SampleScene.unity - AR D - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Create
Show in Explorer
Open
Delete
Rename
Open Scene Additive
Import New Asset...
Import Package
Export Package...
Find References In Scene
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Extract From Prefab
Run API Updater...
Open C# Project
Custom Package...
2D
Cameras
Characters
CrossPlatformInput
Effects
Environment
ParticleSystems
Prototyping
Utility
Vehicles
Inspector
Services
VuforiaConfiguration
Global
Vuforia Version 7.1.34
Download Vuforia 7.2.24
Learn More
App License Key
Add License
Delayed Initialization
Camera Device Mod MODE_DEFAULT
Camera Direction CAMERA_DEFAULT
Digital Eyewear
Device Type Handheld
Databases
VuforiaMars_Images
Asset Labels
AssetBundle
Assets
Editor
Models
Resources
Scenes
StreamingAssets
Vuforia
VuforiaConfiguration.asset
ENG
03:17 AM
05-10-2018


Import package ...
Organize
New folder
Dropbox
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
CreatingARWithVuforia
extra
vuforia-sdk-uwp-7-5-20
VuforiaSupportInstaller-Windows-7-2-24
aarti.unitypackage
arlast.unitypackage
vuforia-unity-6-2-10.unitypackage
File name: arlast.unitypackage
unitypackage (*.unitypackage)
Open
Cancel
VuforiaConfiguration
Global
Vuforia Version 7.1.34
Download Vuforia 7.2.24
Learn More
App License Key
Add License
Digital Eyewear
Device Type
Handheld
Databases
VuforiaMars_Images
Asset Labels
AssetBundle
VuforiaConfiguration.asset

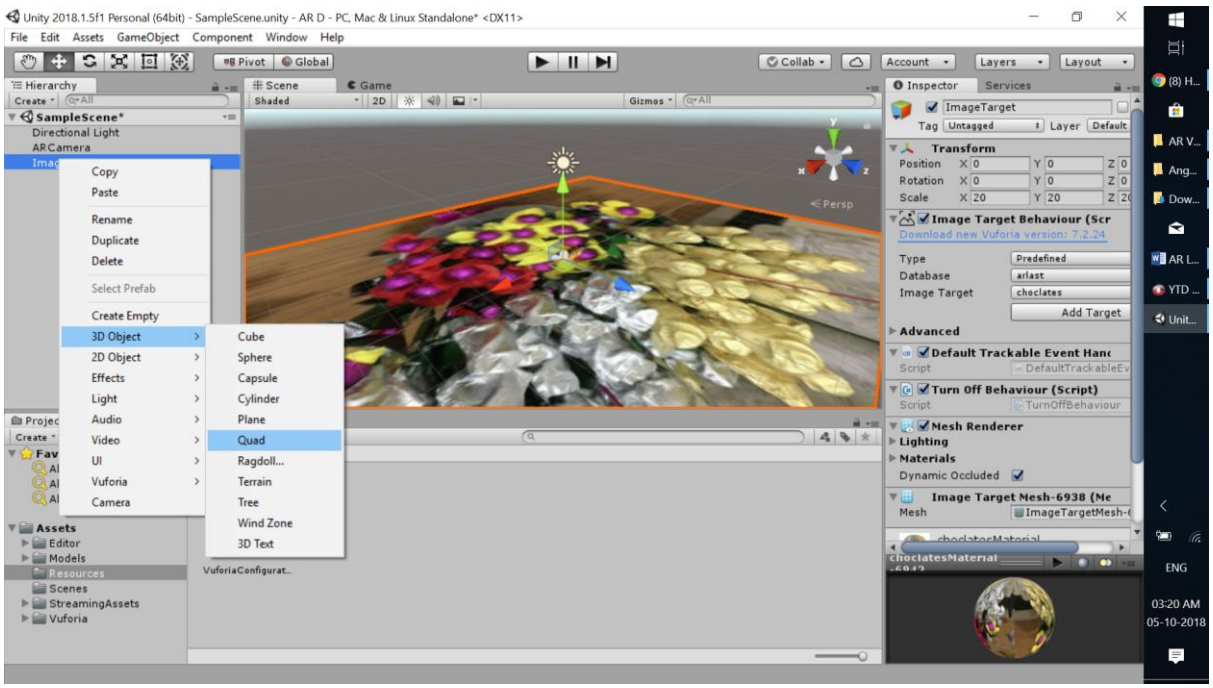
Open

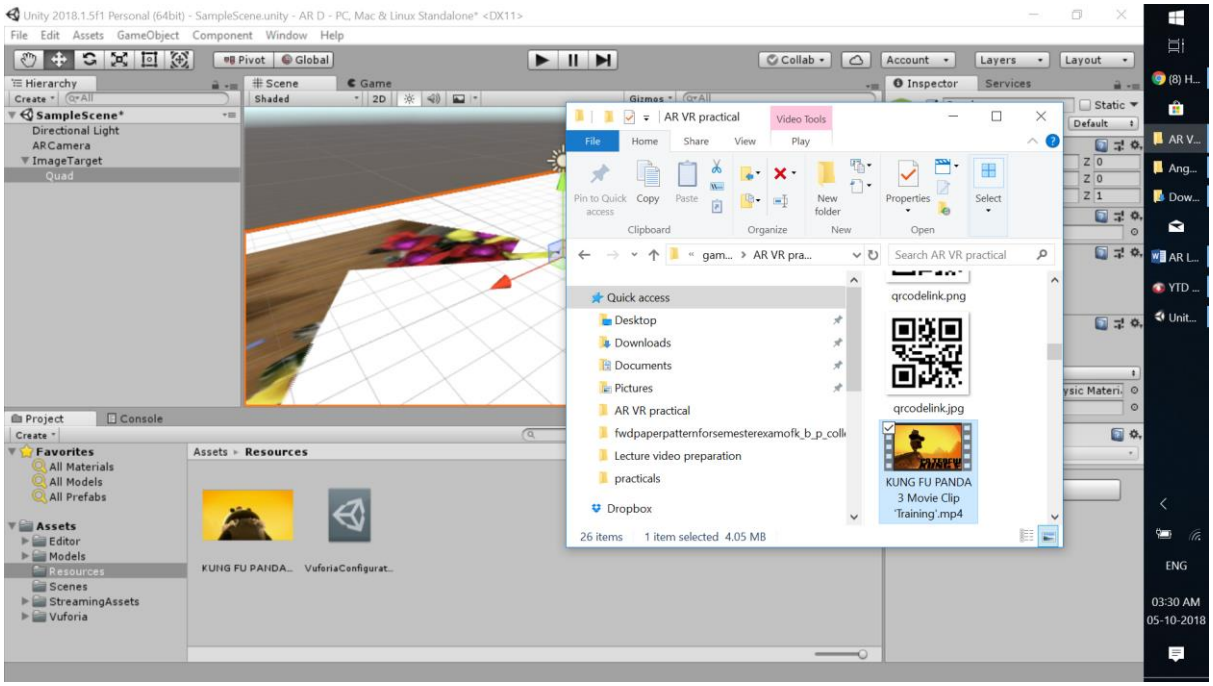
Import

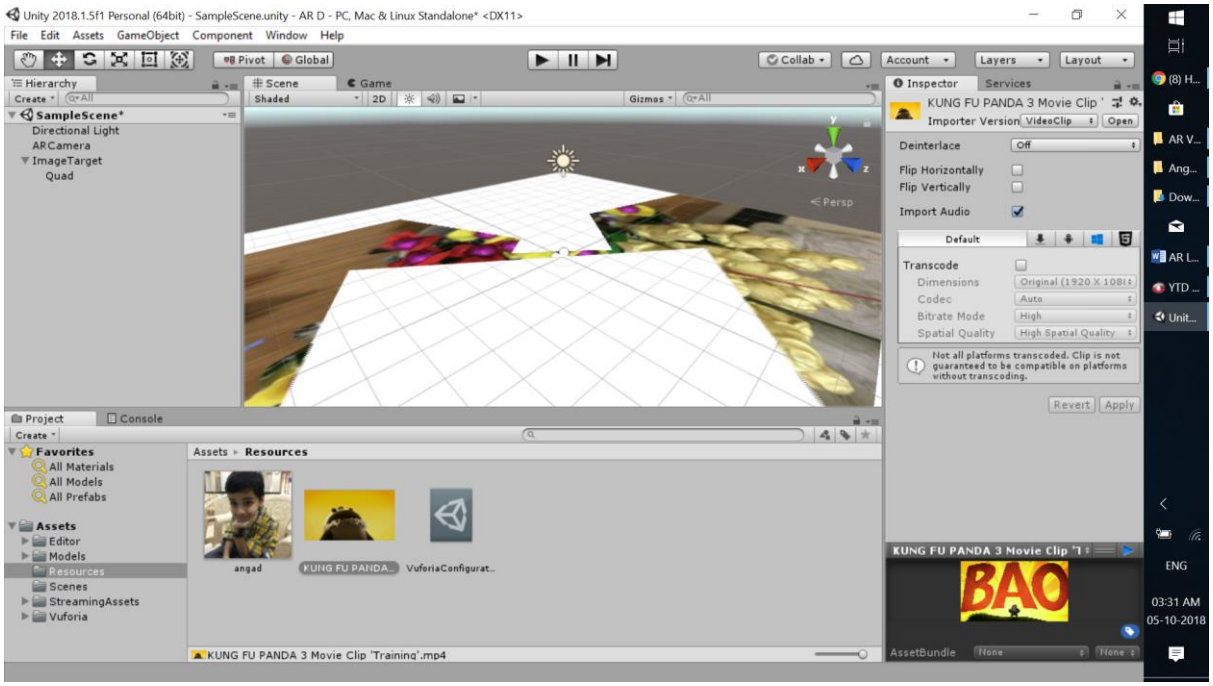

Hierarchy
SampleScene*
Directional Light
ARCamera
ImageTarget
Scene
Game
Inspector
ImageTarget
Transform
Position
Rotation
Scale
Image Target Behaviour (Scr
Download new Vuforia version: 7.2.24
Type
Predefined
Database
VuforiaMars_Images
--- EMPTY ---
arlast
VuforiaMars_Images
Advanced
Turn Off Behaviour (Script)
Mesh Renderer
Lighting
Materials
Dynamic Occluded
Image Target Mesh-6938 (Me
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Editor
Models
Resources
Scenes
StreamingAssets
Vuforia
Assets ▸ Resources
VuforiaConfigurat...
AstronautMaterial



Onclick to choclates we want to play a video







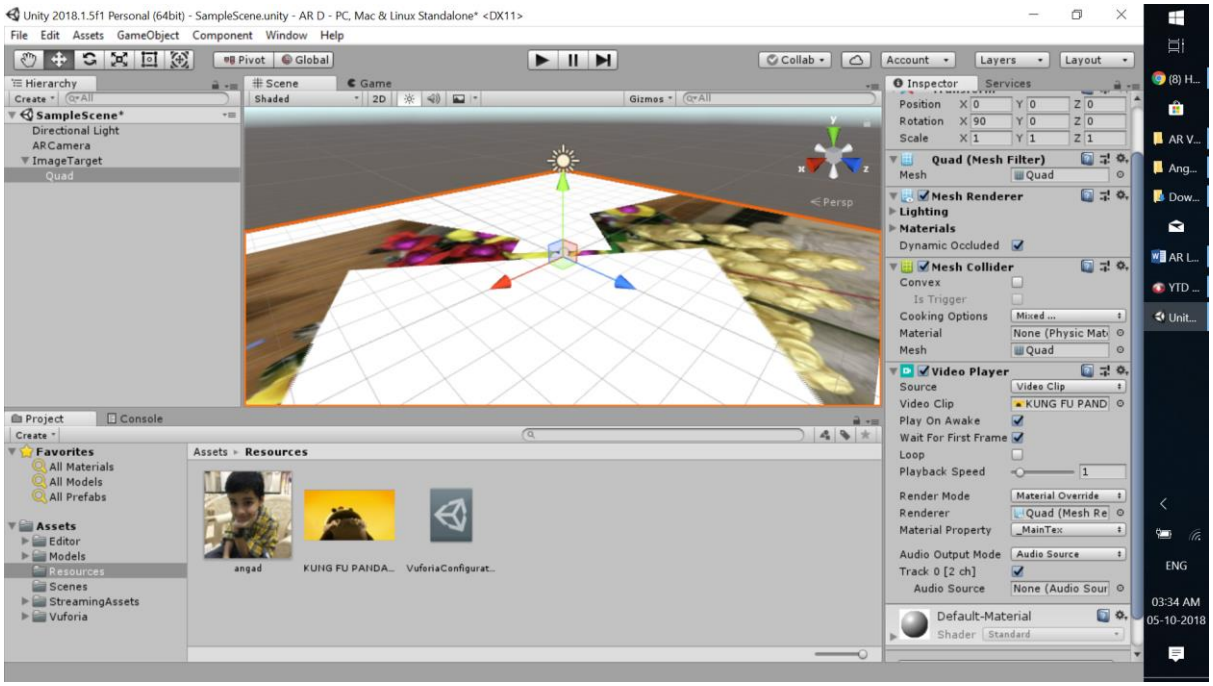
Simply drag and drop images and video in resource folder

If u click on video and play

The video will run

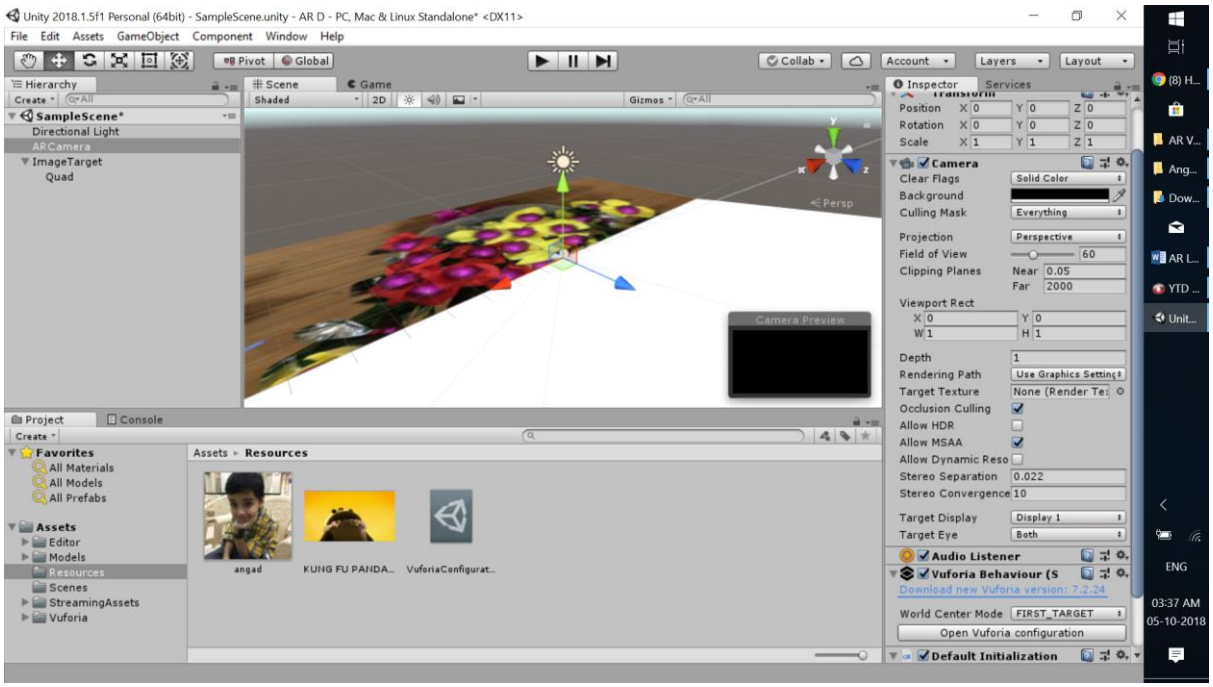
Click on Quad

Add component-video-video player



Drag and drop the video to video clip

Click AR Camera -open Vuforia Configuration



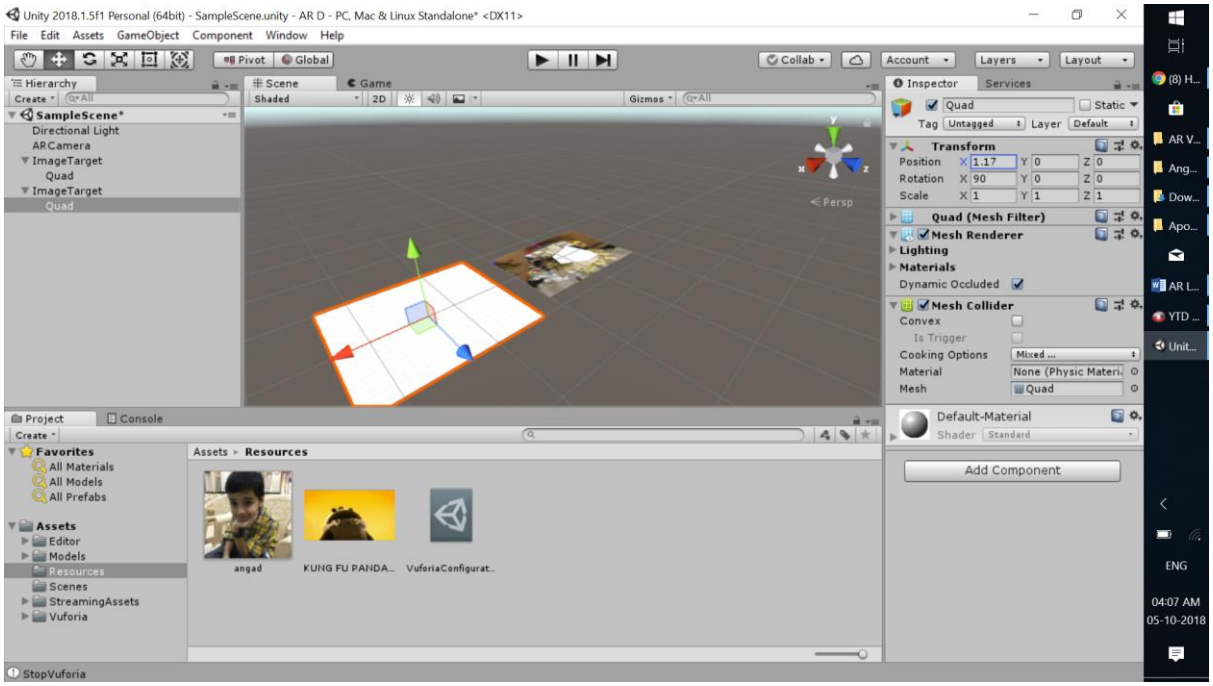
Load and activate arlast db

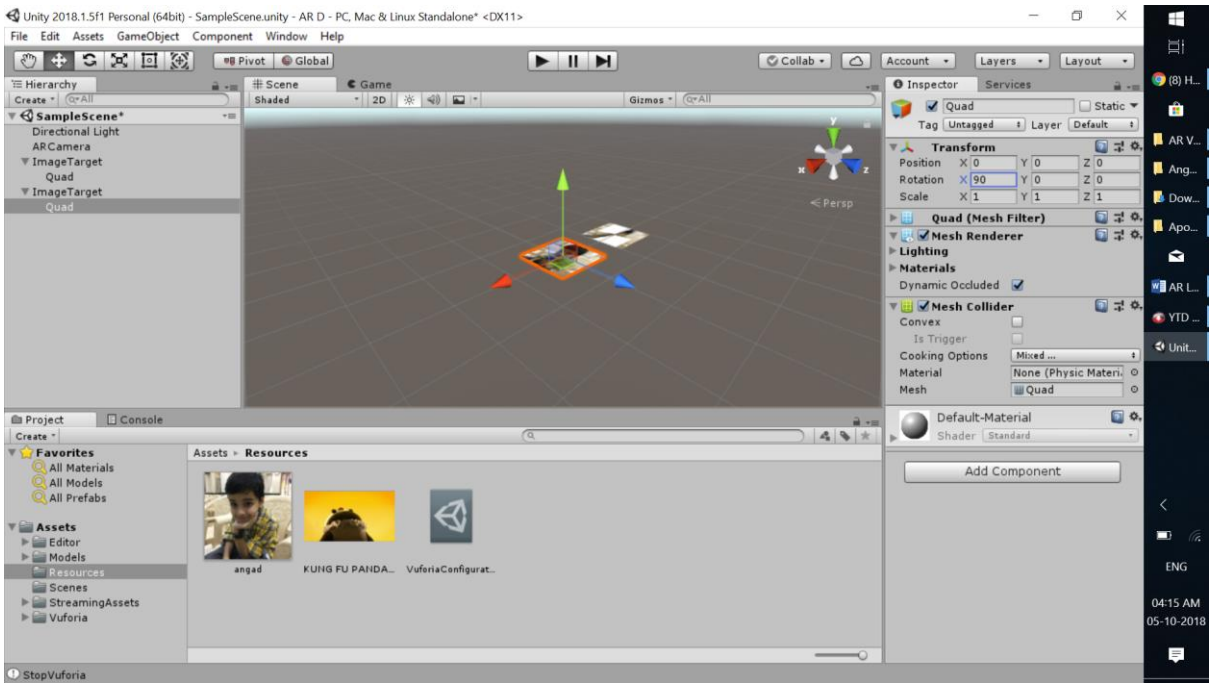
Done

Play

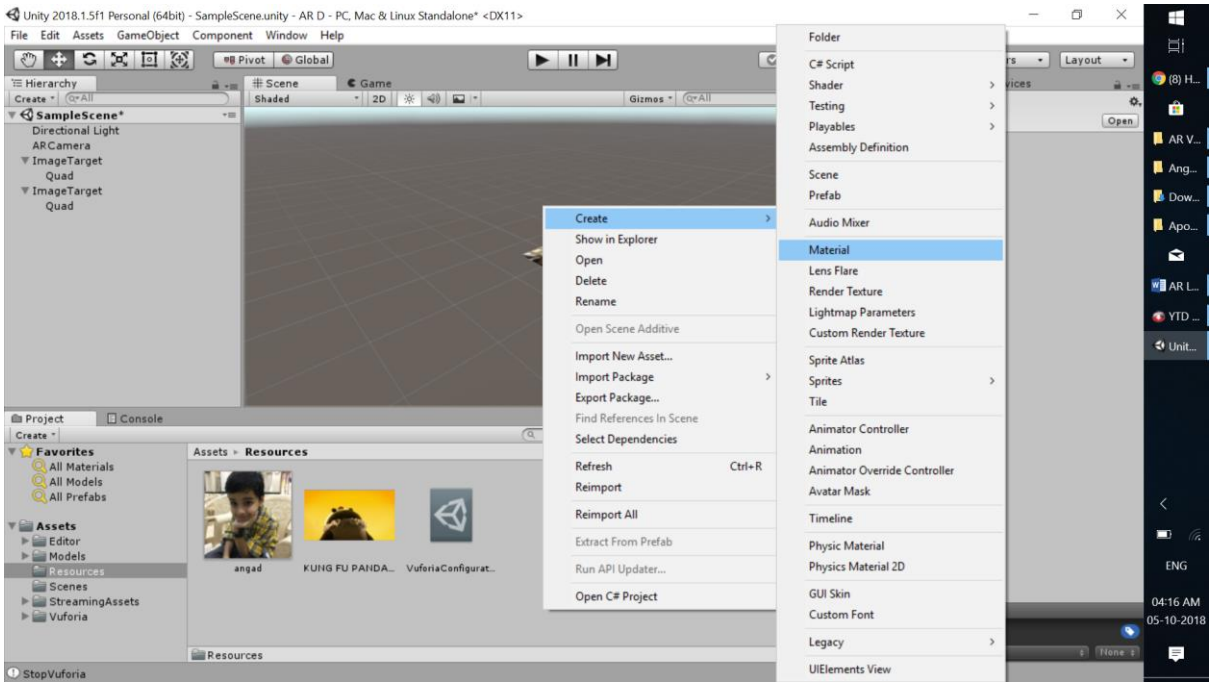
Video of output stored at C:\aarti\CKT\2018-19\ty\game programming\practicals\Final Practical Print\AR VR practical\AR Demo.mp4

We will add one more quad



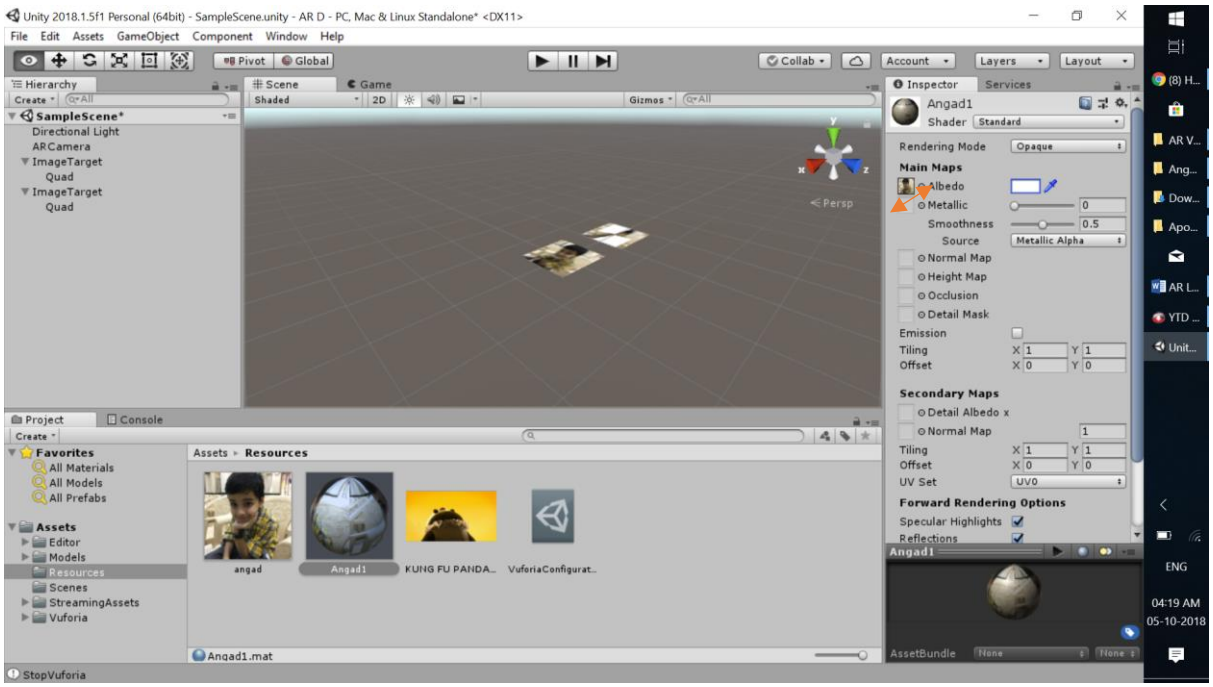


We will add texture to img itself



Name new material as “Angad1”

Now drag the angad image to albedio of that material



Now change value 2 and 2 of max simultaneous tracked images and objects respectively.

Selct the second quad and drag and drop Angad1 material on to element 0 inside material