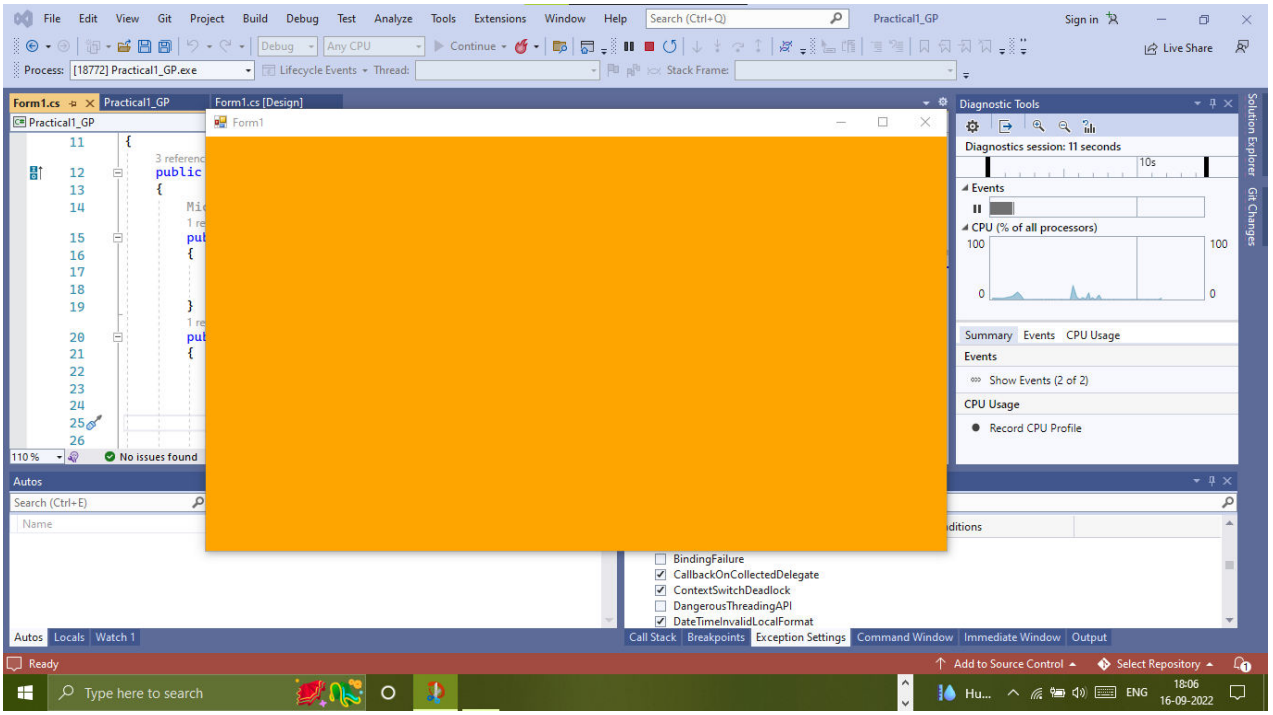
Practical 1

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

CODE:

```
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 Practical1_GP
{
   public partial class Form1 : Form
   {
      Microsoft.DirectX.Direct3D.Device device;
      public Form1()
      {
         InitializeComponent();
         InitDevice();
      }
      public void InitDevice()
      {
         PresentParameters pp = new PresentParameters();
         pp.Windowed = true;
         pp.SwapEffect = SwapEffect.Discard;
         device = new Device(0, DeviceType.Hardware, this,
CreateFlags.HardwareVertexProcessing, pp);
      }
      private void Render()
      {
         device.Clear(ClearFlags.Target, Color.CornflowerBlue, 0, 1);
         device.Present();
      }

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

OUTPUT:


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Practical1_GP
Sign in
Live Share
Form1.cs
Practical1_GP
Form1.cs [Design]
Form1
Diagnostic Tools
Diagnostics session: 11 seconds
Events
CPU (% of all processors)
Summary Events CPU Usage
Show Events (2 of 2)
Record CPU Profile
No issues found
Autos
Search (Ctrl+E)
Name
BindingFailure
CallbackOnCollectedDelegate
ContextSwitchDeadlock
DangerousThreadingAPI
DateTimeInvalidLocalFormat
Autos Locals Watch 1
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output
Ready
Add to Source Control
Select Repository
Type here to search
ENG
18:06
16-09-2022

Practical 2

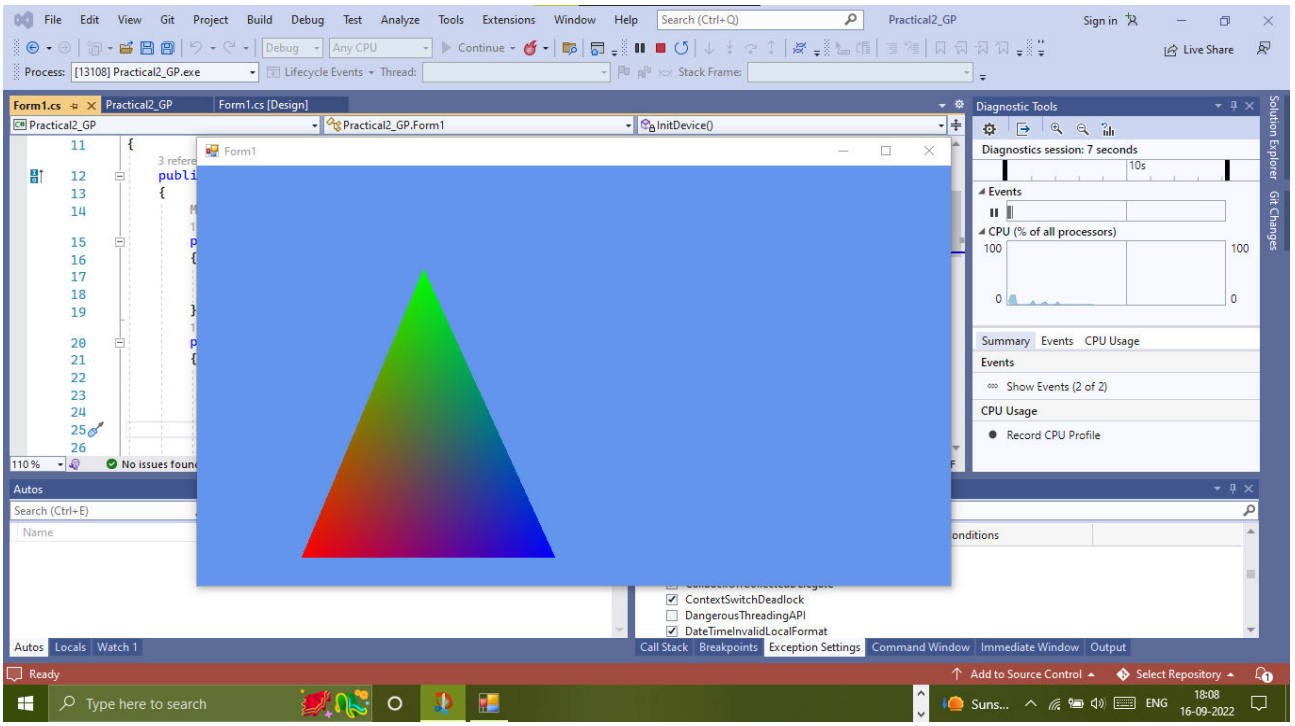
AIM: Draw a triangle using Direct3D 11.

CODE:

```
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 Practical2_GP
{
  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();
        }
    }
}
```

OUTPUT:



Practical 3

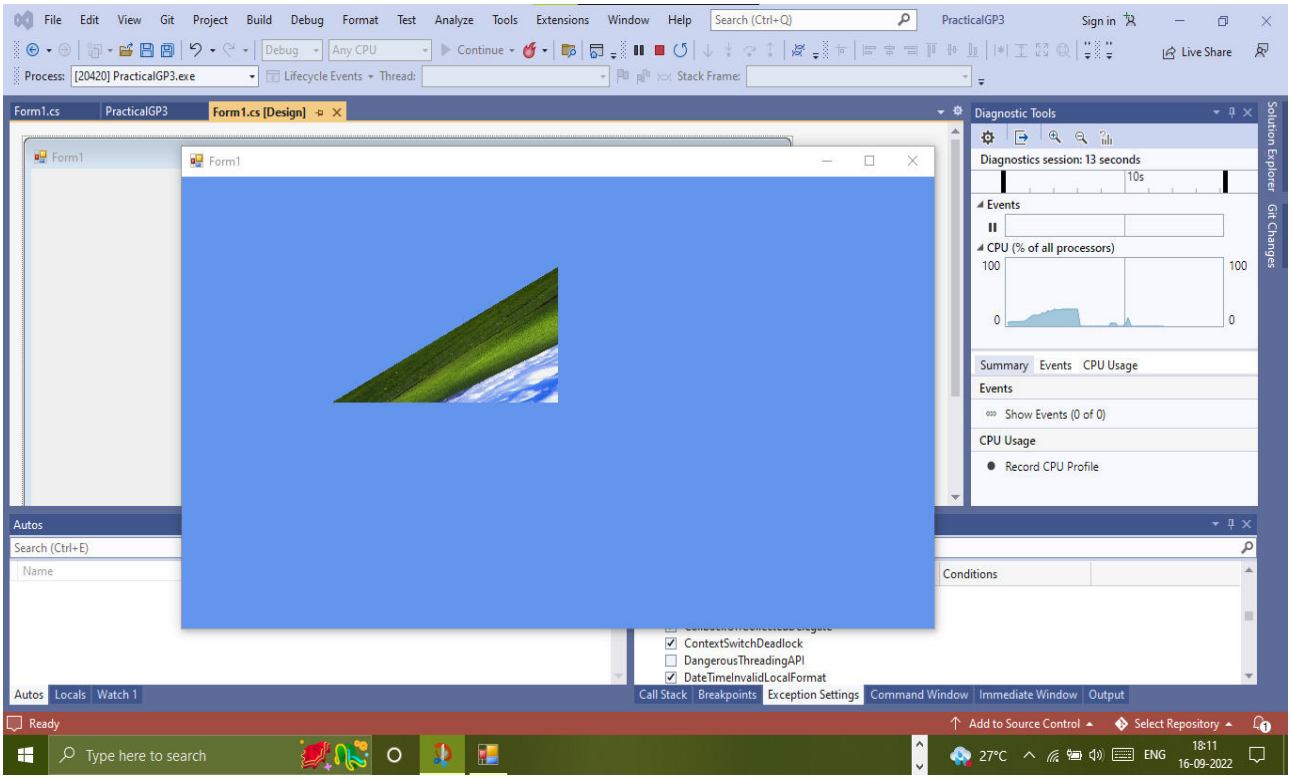
AIM: Texture The Triangle using Direct3D 11.

CODE:

```
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 PracticalGP3
{
   public partial class Form1 : Form
   {
      private Microsoft.DirectX.Direct3D.Device device;
      private CustomVertex.PositionTextured[] vertex = new
CustomVertex.PositionTextured[3];
      private Texture texture;
      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);
         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("D:\\Trial.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:



Practical 4

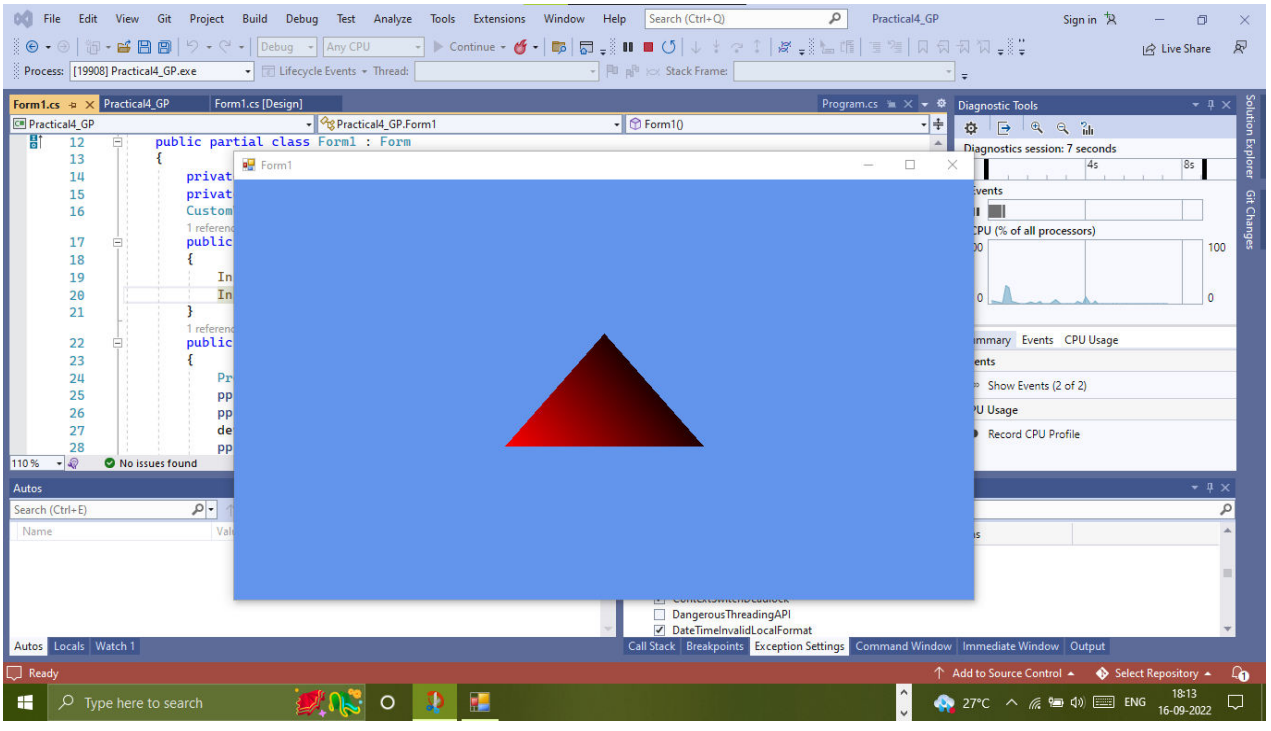
AIM: Programmable Diffuse Lightning using Direct3D 11.

CODE:

```
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 Practical4_GP
{
   public partial class Form1 : Form
   {
      private Microsoft.DirectX.Direct3D.Device device;
      private CustomVertex.PositionNormalColored[] vertex = new
      CustomVertex.PositionNormalColored[3];
      public Form1()
      {
         InitializeComponent();
         InitDevice();
      }
      public void InitDevice()
      {
         PresentParameters pp = new PresentParameters();
         pp.Windowed = true;
         pp.SwapEffect = SwapEffect.Discard;
         device = new Device(0, DeviceType.Hardware, this,
CreateFlags.HardwareVertexProcessing,
         pp);
         device.Transform.Projection = Matrix.PerspectiveFovLH(3.14f / 4,
device.Viewport.Width /
         device.Viewport.Height, 1f, 1000f);
         device.Transform.View = Matrix.LookAtLH(new Vector3(0, 0, 10), new Vector3(),
new
         Vector3(0, 1, 0));
         device.RenderState.Lighting = false;
         vertex[0] = new CustomVertex.PositionNormalColored(new Vector3(0, 1, 1), new
Vector3(1, 0,
         1), Color.Red.ToArgb());
         vertex[1] = new CustomVertex.PositionNormalColored(new Vector3(-1, -1, 1), new
Vector3(1,
         0, 1), Color.Red.ToArgb());
         vertex[2] = new CustomVertex.PositionNormalColored(new Vector3(1, -1, 1), new
Vector3(-1,
0, 1), Color.Red.ToArgb());
```

```
            device.RenderState.Lighting = true;
            device.Lights[0].Type = LightType.Directional;
            device.Lights[0].Diffuse = Color.Plum;
            device.Lights[0].Direction = new Vector3(0.8f, 0, -1);
            device.Lights[0].Enabled = true;
        }
        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 5

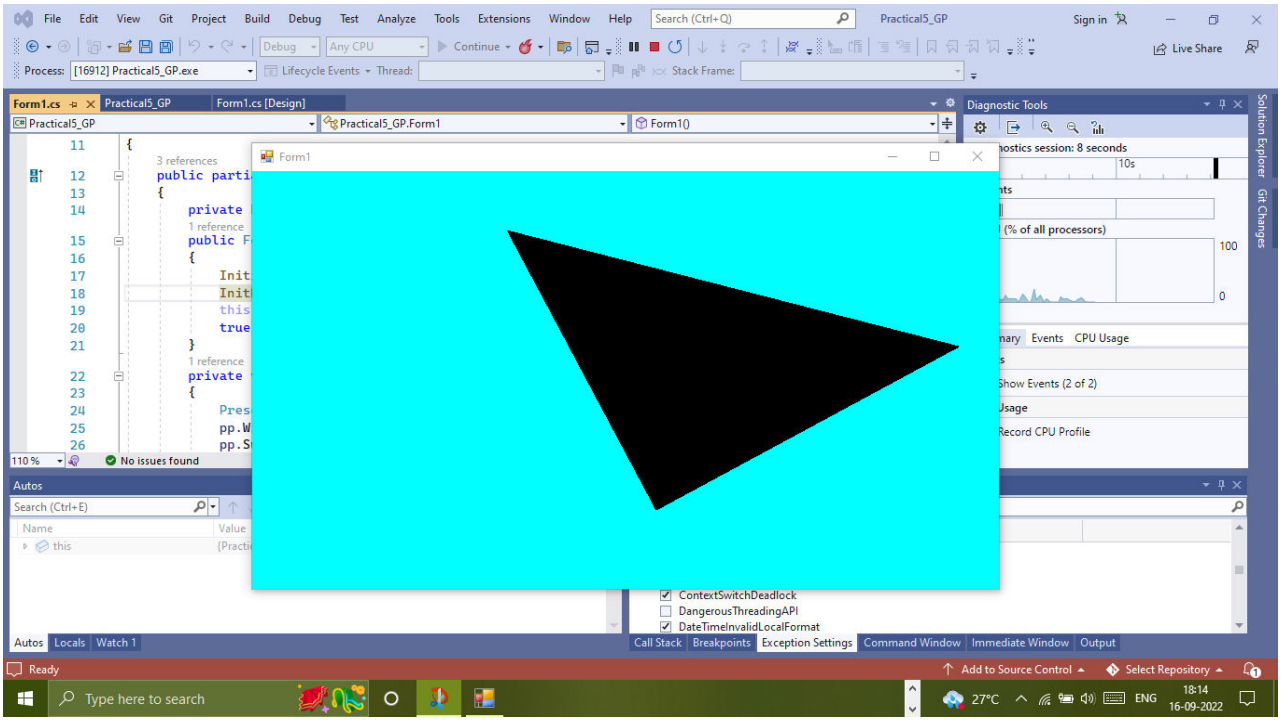
AIM: Programmable Spot Lightning using Direct3D 11.

CODE:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX.Direct3D;
using Microsoft.DirectX;
namespace Practical5_GP
{
   public partial class Form1 : Form
   {
      private Device device;
      public Form1()
      {
         InitializeComponent();
         InitDevice();
         this.SetStyle(ControlStyles.AllPaintingInWmPaint | ControlStyles.Opaque,
         true);
      }
      private void InitDevice()
      {
         PresentParameters pp = new PresentParameters();
         pp.Windowed = true;
         pp.SwapEffect = SwapEffect.Discard;
         device = new Device(0, DeviceType.Hardware, this,
         CreateFlags.SoftwareVertexProcessing, pp);
         device.RenderState.CullMode = Cull.None;
         device.RenderState.Lighting = true;
         device.Lights[0].Type = LightType.Spot;
         device.Lights[0].Range = 4;
         device.Lights[0].Position = new Vector3(0, -1, 0f);
         device.Lights[0].Enabled = true;
      }
      private void Render()
      {
         device.Transform.Projection = Matrix.PerspectiveFovLH((float)Math.PI / 4,
         this.Width / this.Height, 1f, 50f);
         device.Transform.View = Matrix.LookAtLH(new Vector3(0, 0, 30), new
         Vector3(1, 0, 0), new Vector3(0, 5, 0));
         CustomVertex.PositionNormalColored[] vertices = new
         CustomVertex.PositionNormalColored[6];
         vertices[0].Position = new Vector3(10f, 12f, 0f);
         vertices[0].Normal = new Vector3(0, 2, 0.5f);
```

```
            vertices[0].Color = Color.Yellow.ToArgb();
            vertices[1].Position = new Vector3(-5f, 5f, 0f);
            vertices[1].Normal = new Vector3(0, 2, 0.5f);
            vertices[1].Color = Color.Blue.ToArgb();
            vertices[2].Position = new Vector3(5f, 5f, -1f);
            vertices[2].Normal = new Vector3(0, 0, 0.5f);
            vertices[2].Color = Color.Pink.ToArgb();
            vertices[3].Position = new Vector3(5f, -5f, -1f);
            vertices[3].Normal = new Vector3(0, 0, 0.5f);
            vertices[3].Color = Color.Green.ToArgb();
            vertices[4].Position = new Vector3(10f, 12f, 0f);
            vertices[4].Normal = new Vector3(0, 0, 0.5f);
            vertices[4].Color = Color.Green.ToArgb();
            device.Clear(ClearFlags.Target, Color.Cyan, 1.0f, 0);
            device.BeginScene();
            device.VertexFormat = CustomVertex.PositionNormalColored.Format;
            device.Transform.World = Matrix.Translation(-5, -10 * 1 / 3, 0) *
            Matrix.RotationAxis(new Vector3(), 0);
            Console.WriteLine(device.Transform.World.ToString());
            device.DrawUserPrimitives(PrimitiveType.TriangleStrip, 3, vertices);
            device.EndScene();
            device.Present();
            this.Invalidate();
        }
        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Render();
        }
    }
}
```

OUTPUT:


Form1
Practical5_GP
Type here to search
27°C
ENG
18:14
16-09-2022

Practical 6

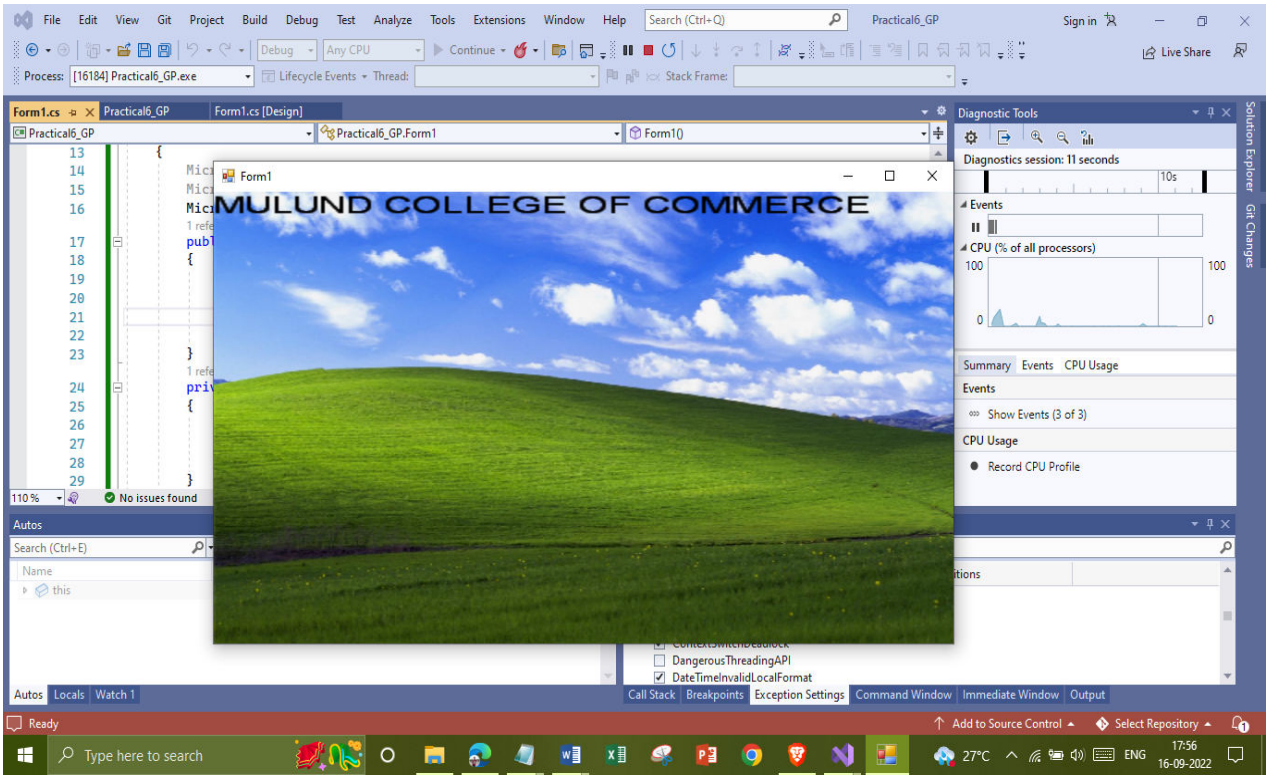
AIM: Loading models into DirectX 11 and rendering.

CODE:

```
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 Practical6_GP
{
  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, "D:\\PracImage.jpg", 400, 400, 1, 0,
      Format.A8B8G8R8, Pool.Managed, Filter.Point, Filter.Point,
      Color.Transparent.ToArgb());
    }
    private void InitDevice()
    {
      PresentParameters pp = new PresentParameters();
      pp.Windowed = true;
      pp.SwapEffect = SwapEffect.Discard;
      device = new Device(0, DeviceType.Hardware, this,
      CreateFlags.HardwareVertexProcessing, pp);
    }
    private void Render()
    {
```

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

OUTPUT:



Practical 7

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

CODE:

1. PlayerController.cs

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

public class PlayerController : MonoBehaviour
{
   private Rigidbody2D rb2d;
   public float speed;
   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 * speed);
   }
}
```

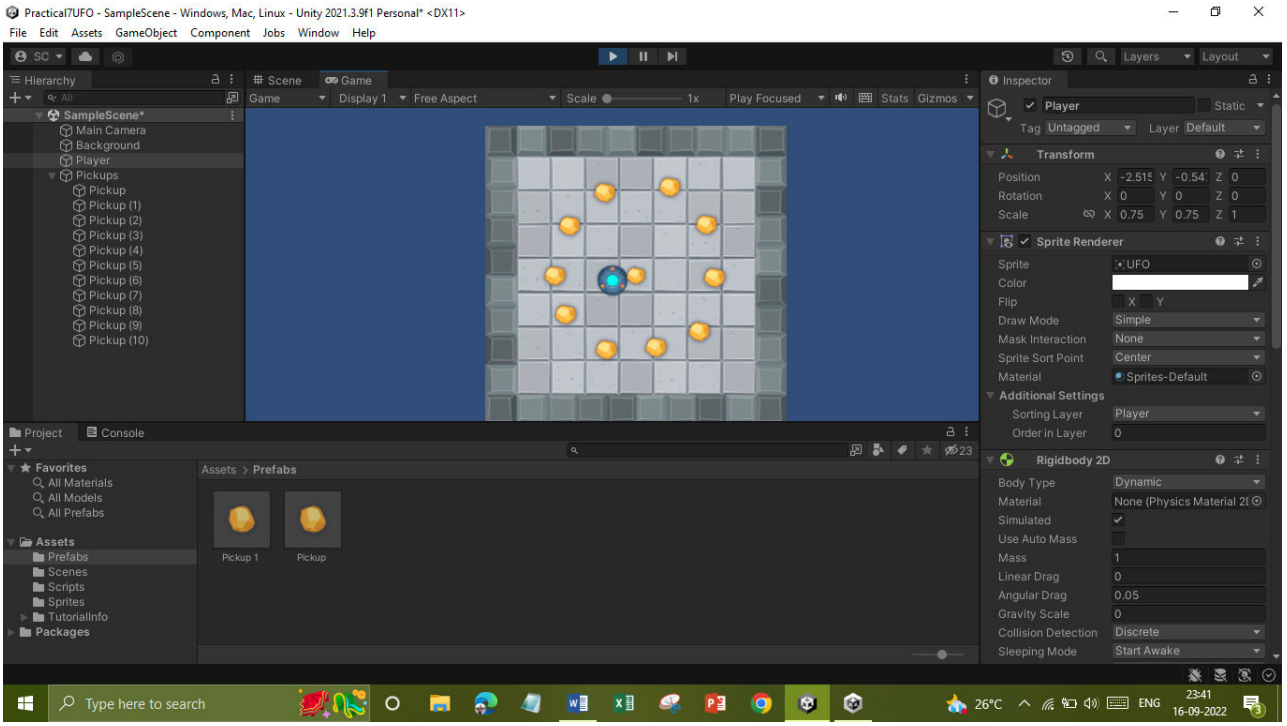
2. CamerController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CameraController : MonoBehaviour
{
        public Transform player;
        private Vector3 offset;
        void Start()
        {
                offset = transform.position - player.position;
        }
        void LateUpdate()
        {
                transform.position = player.position + offset;
        }
}
```

3. Rotator.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Rotator : MonoBehaviour
{
        void Update()
        {
                transform.Rotate(new Vector3(0, 0, 45) * Time.deltaTime);
        }
}
```

OUTPUT:



Steps to perform Practical Number 7:

1. Open Unity. Select 2d, give Project Name and click on create a new project.
2. Click on “Window Menu” at the top, then select asset store or else Press “Ctrl + 9”.
3. Search for Sample project or 2D UFO Tutorial.
4. Search for 2D UFO Tutorial and click on it.
5. Click on down load and the click on import.

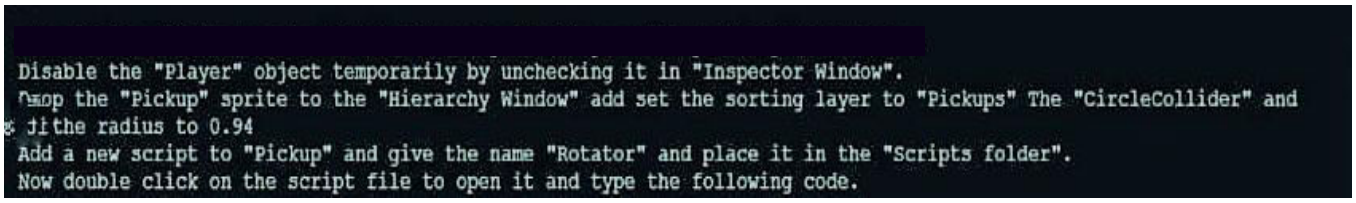
```
Click on download and after downloading then click on Import.
Then save your scene by  "Ctrl+S" or from file menu and give the name to the scene as "Main" and save it into "Scenes"
folder
Drop the sprite named "Background" to "Hierarchy Window" from "sprites" folder in project window.
Move your cursor over the image and then click "F " key to see full game object.
Drag and drop the "UFO" sprite to the "Hierarchy Window" and the go to the Inspector window to the right
and type "Player" on the top.
Click the "Background" in "Hierarchy Window" and then select "Sorting layer" in the "inspector window" at right and
Then select "Background" from the dropdown
Similarly select "Player" and select its "sorting layer" as Player.
Change the scale value to X=0.75,Y=0.75,Z=1 in "inspector window".
Click on "MainCamera" in "Hierarchy Window" and then go to "Inspector Window" and edit the value of "size" to 16.5
Click on "Player" object and then go to "Component Menu" and the select "Physics2D" nad then select "Rigidbody2D".
Now select the "Player" object and then go to "Inspector Window" and then click on "Add Component" select "New Script"
provide the name to the script as "PlayerController" then click "Create and add".
Now move the script file from "Project Window" to the "Scripts" folder
Now double click on the script file to open it for editing.
Remove the start() and update() method.
And type this code in it
```

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

public class PlayerController : MonoBehaviour
{
    private Rigidbody2D rb2d;
    public float speed;

    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 * speed);
    }

}
```

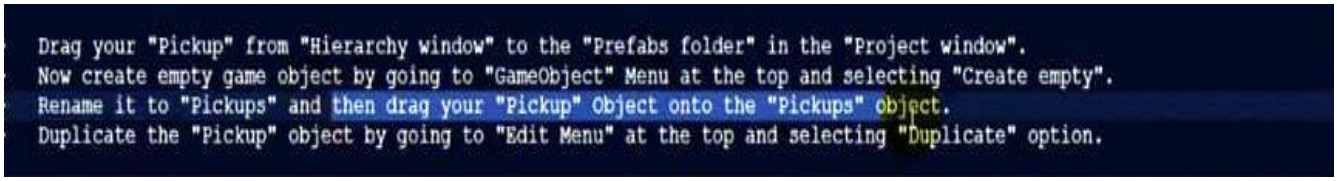
```
Click on your "Player" object and then go to "Inspector Window" and then set "Gravity Scale" to 0 and "Speed" to 10.
and also click on "Add component" then type circle.. select then "Circle Collider 2d" and then edit the "Radius"
value to 2.15.
Click on "Background" then "Add component" then type Box.. select "BoxCollider2"
Then edit the Offset X = 14.3 Size X = 3.3
              Offset Y = 0    Size Y = 31.64
```

```
Similarly copy and paste this BoxCollider 3 times like this and adjust it to fit..
Click on "MainCamera" the click on "Add Component" then select "New Script" type the name as "CameraController".
then click "Create and add" Again move the script to "Scripts folder"
Now double click on that script and type the following code.
```

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

public class CameraController : MonoBehaviour
{
    public GameObject player;
    private Vector3 offset;

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

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




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

public class Rotator : MonoBehaviour
{
    void Update()
    {
        transform.Rotate(new Vector3(0,0,45)*Time.deltaTime);
    }
}
```



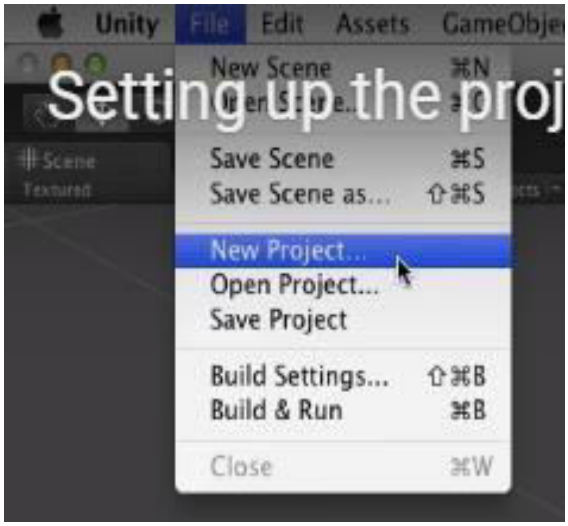
Practical 8

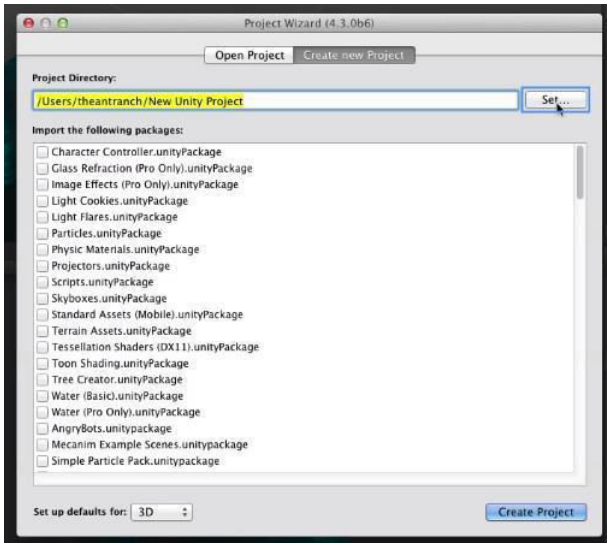
AIM: Using a unity3d software and creating space shooter.

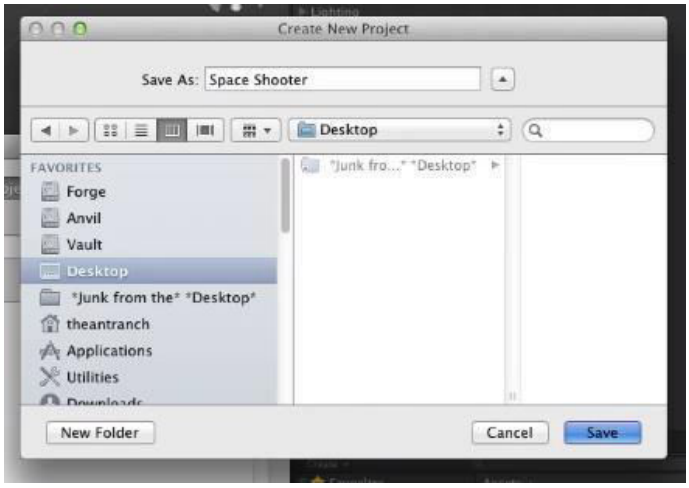
CODE:

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

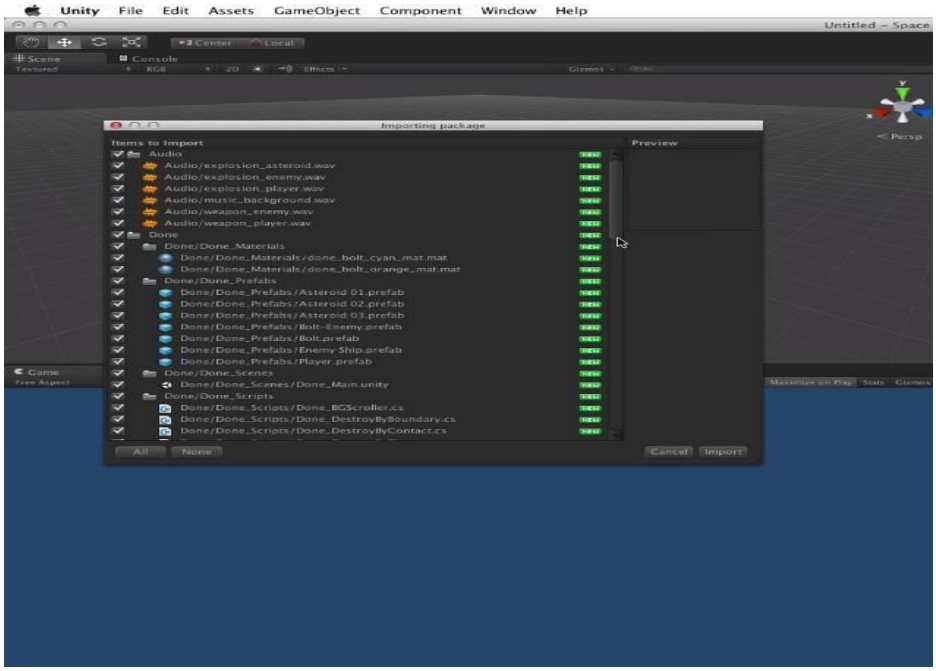
❖ Go to file-> New Project->Click on create new project ->set the location (By clicking on Set Button)-> Give the name to your project As Space Shooter->Click on save Button->Click on Create Button.







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



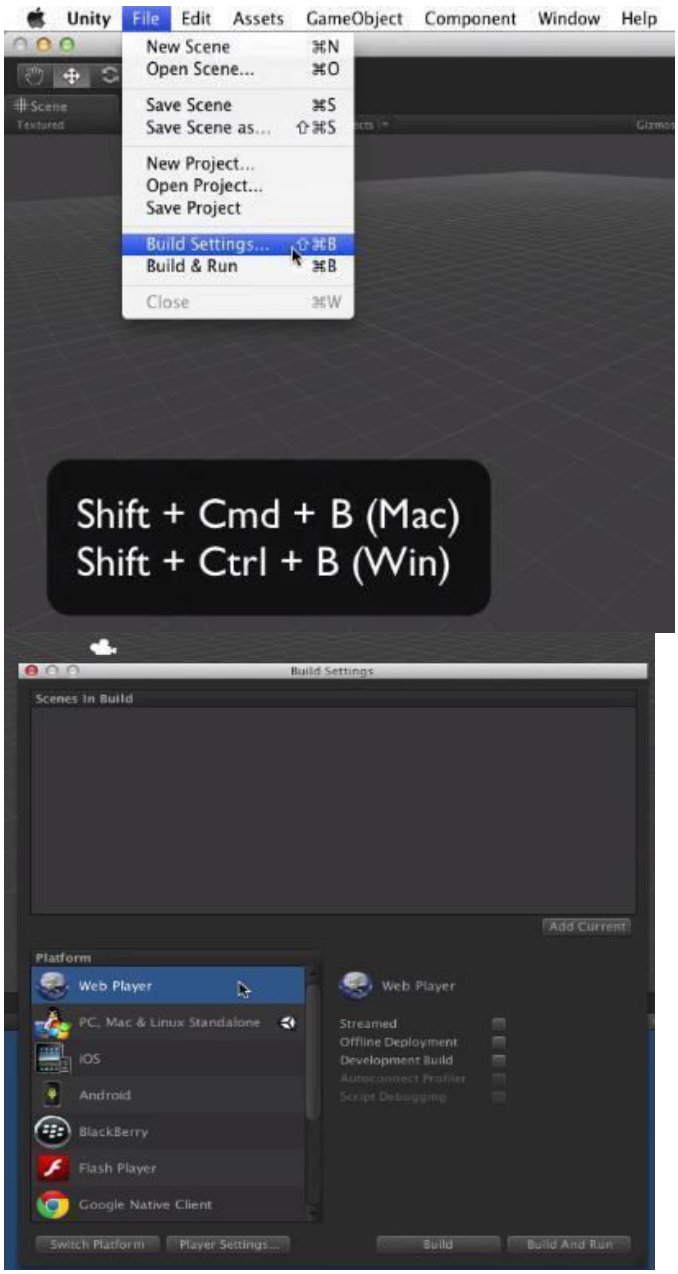
The Scene will look like after Importing the Asset -> Select All Package->Click on Import.

●The Asset is created. Save your scene. (Crtl+s)

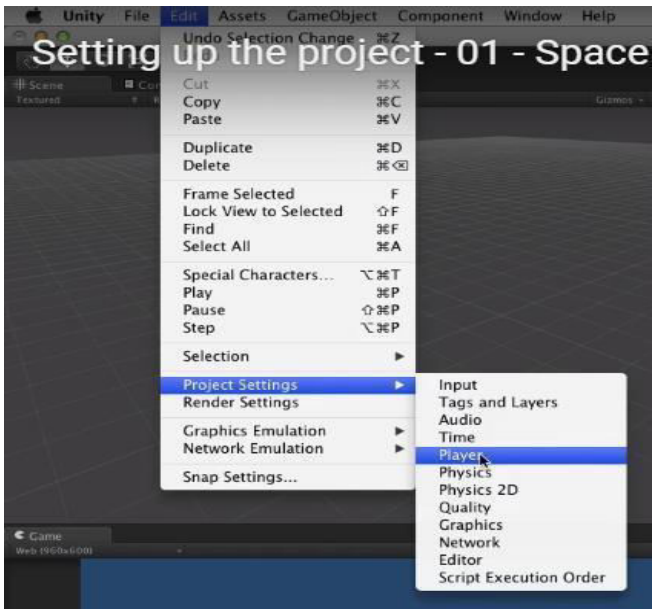
●Save the scene inside the Asset directory make new folder as _Scene. Give the name to your scene as main the save it.

Now we will set the build target to our scene.

Go to file->Build Setting->Select Web Player->Click on switch platform.



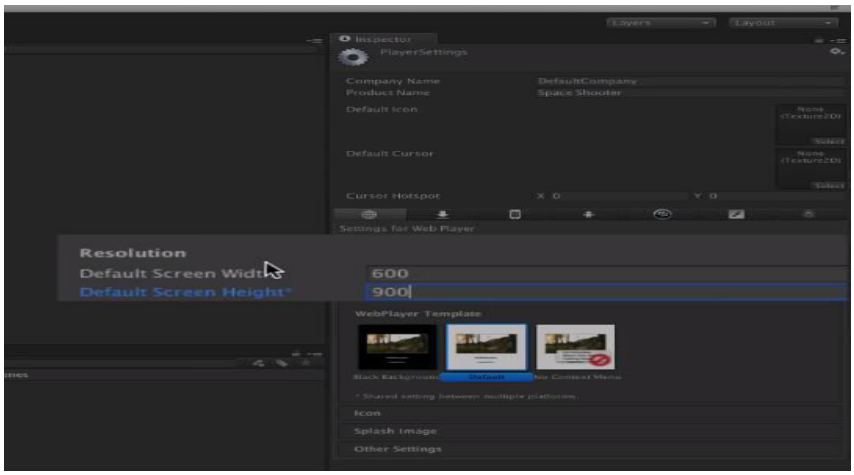
Now we need to fill the Build Detail. Go To File-> Project Setting->Player->



Set the resolution:

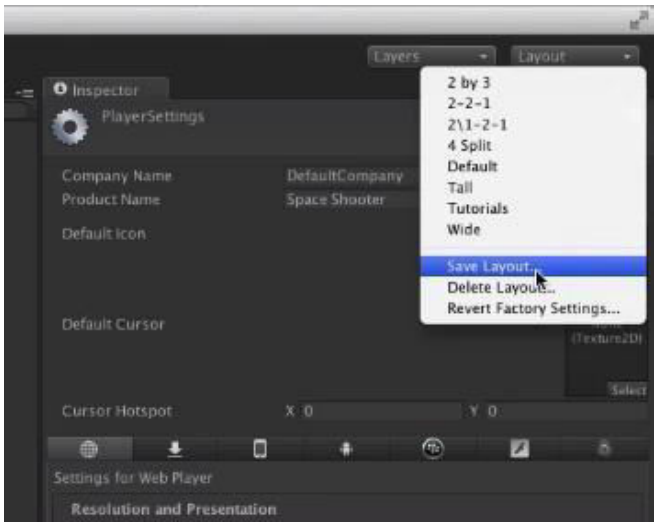
Go to Inspector window(Right Middle Corner Of the scene)->Resolution-

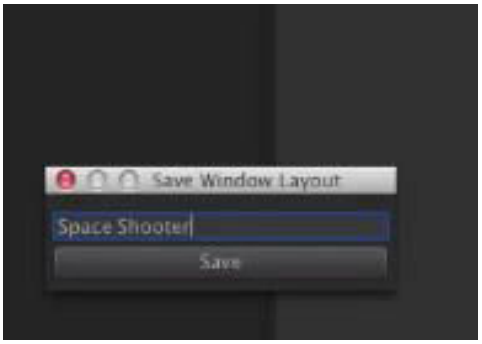
>Change the resolution width as 600 AND height as 900.



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.



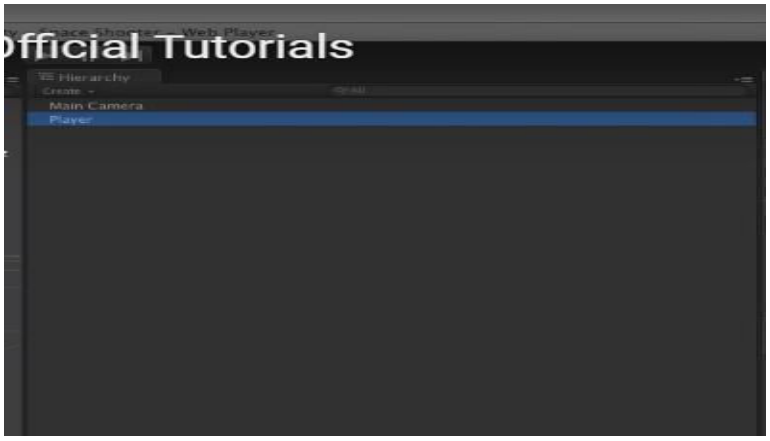


Step 2: Setting Up the player Game Object

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

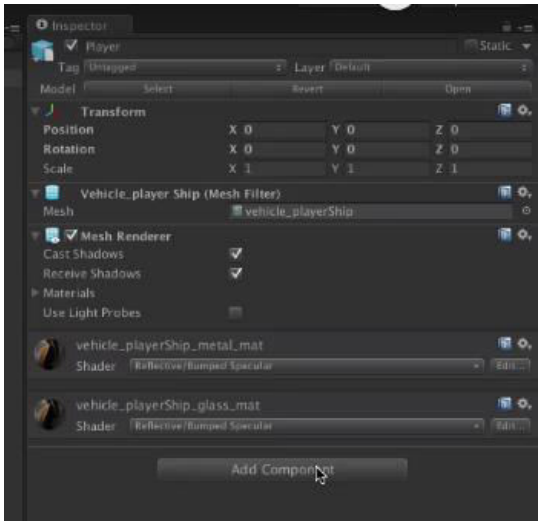
Go to edit->choose frame Selected

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

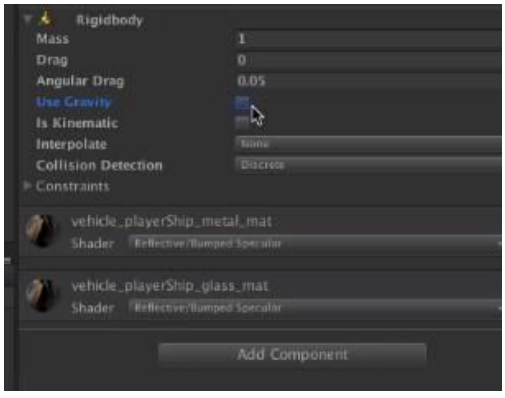


Add New Component into the Inspector Window

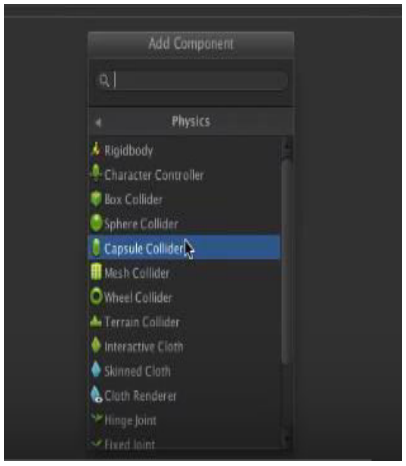
Click on Add Component Tab->Select Physics->Rigid Body



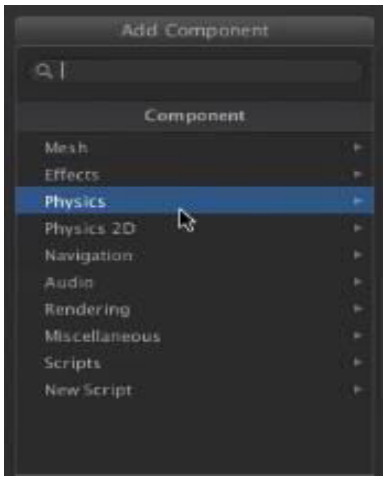
GotoRigit Body Component ->Deselect Use Gravity ->



Add New Component into the Inspector Window

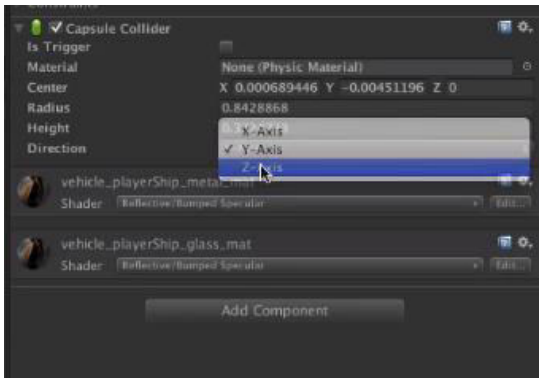


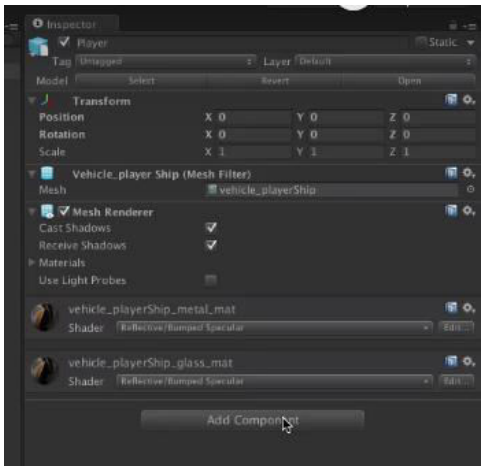
Click on Add Component Tab->Select Physics->Select Capsule Collider



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

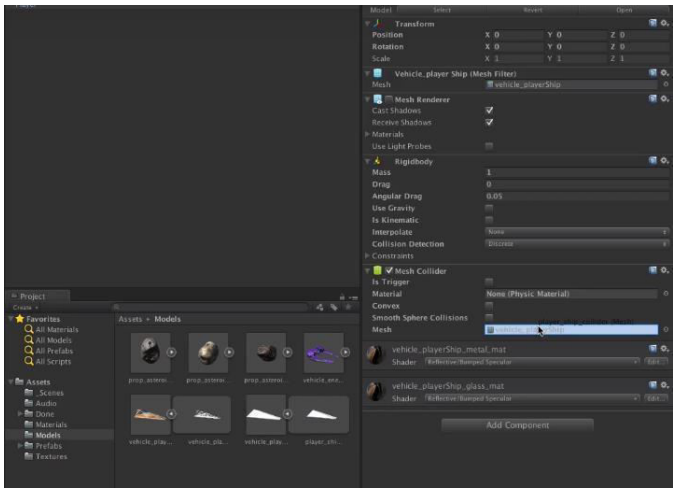
Go to Add Component button again->Physics->Mesh Collider->Replace



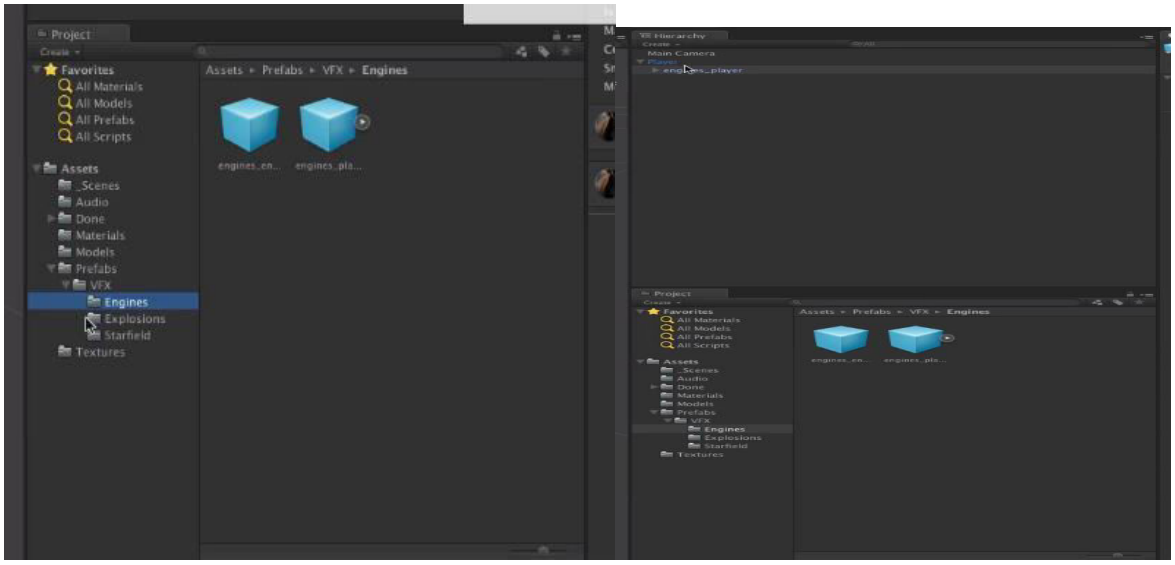


Go to mesh collider Component inside the Inspector window->turn off the Mesh renderer tab

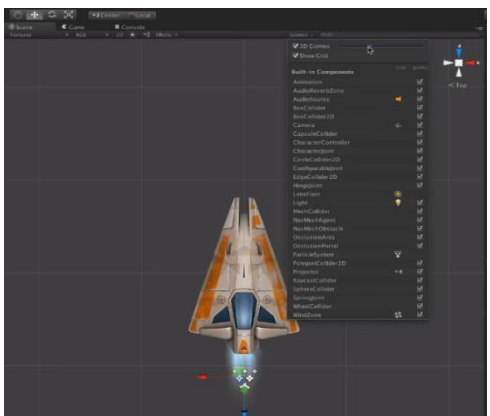
Open the Model-> Select the Mesh Asset->Drag it in to the Mesh Slot on the Mesh Collider->Turn On the Mesh Renderer tab-> Select Is Trigger
tab



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

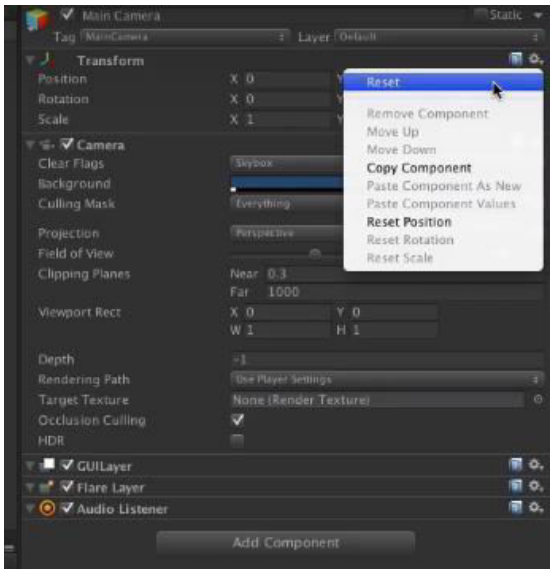


You Can Change the Position of Ship By Changing the Tap View the scene.



Step 3 : Setting up the main camera and lighting for

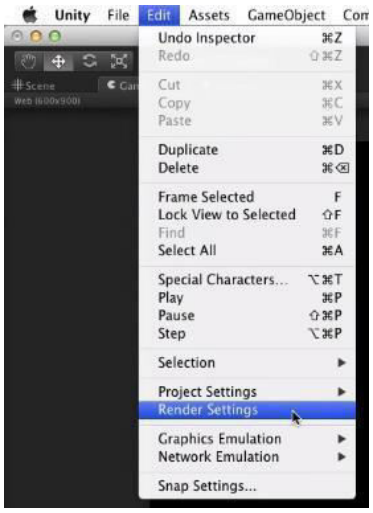
Click on the main camera from hierarchy->Go To Inspector window->go to transform component ->click on Reset Tab->Now Change the Rotation of x-axis as 90->go to camera view ->click on the projection->set it as orthographic->Se the size to 10



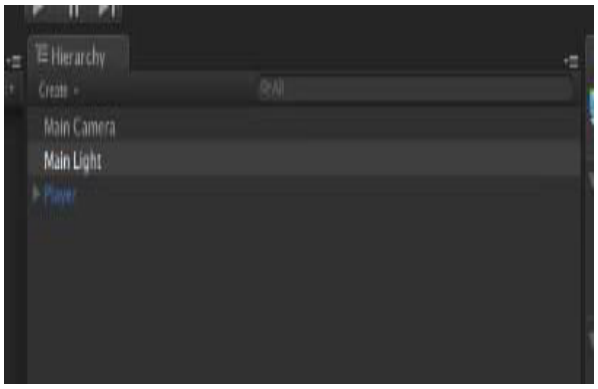
Go to camera->change the clear flags to Solid Color-> Change the background color as Black.

Setting up the Light

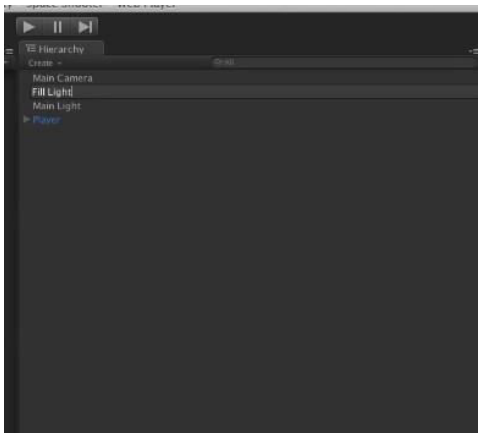
Go to edit->Render Setting->Change the Ambient Light to 0.0.0.0(Black in color)

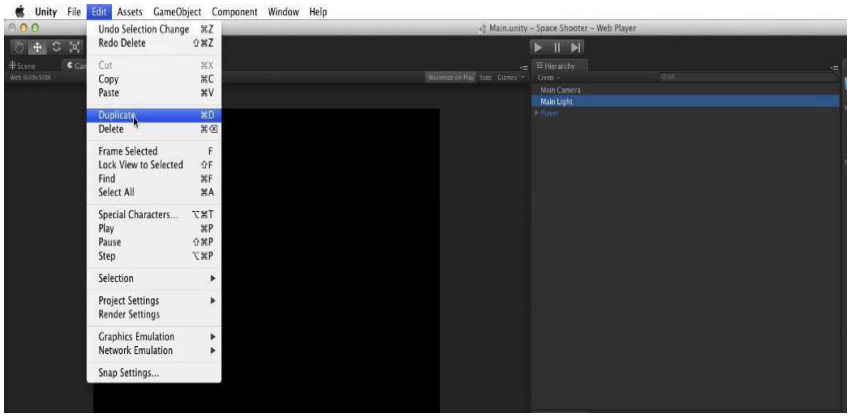


Go to hierarchy->click on create->Directional Light->Rename it with Main Light->Reset the light position->set the rotation x axis as 20->y axis as -115->

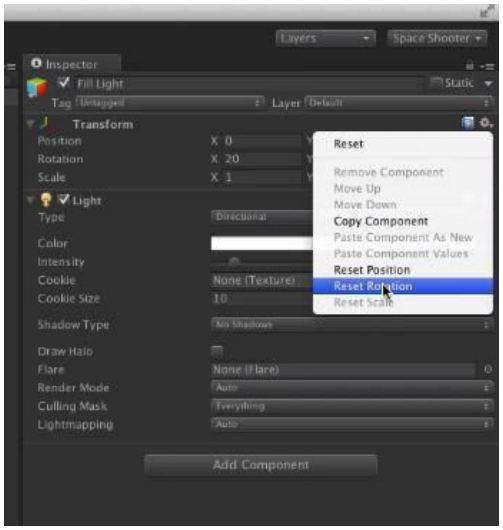


Select the Main Light from Hierarchy->go to Edit Menu->Select Duplicate ->Rename the Duplicate as Fill





Go to Inspector window->reset the Rotation of fill Light



Change the light Intensity as 0.05->Change the Colors->Change the rotations x-axis as 5 and y- axis as 125->Duplicate the fill light->Rename it with Rim Light->Deselect the Rim Light From Inspector Window->select Rim Light From 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 background.

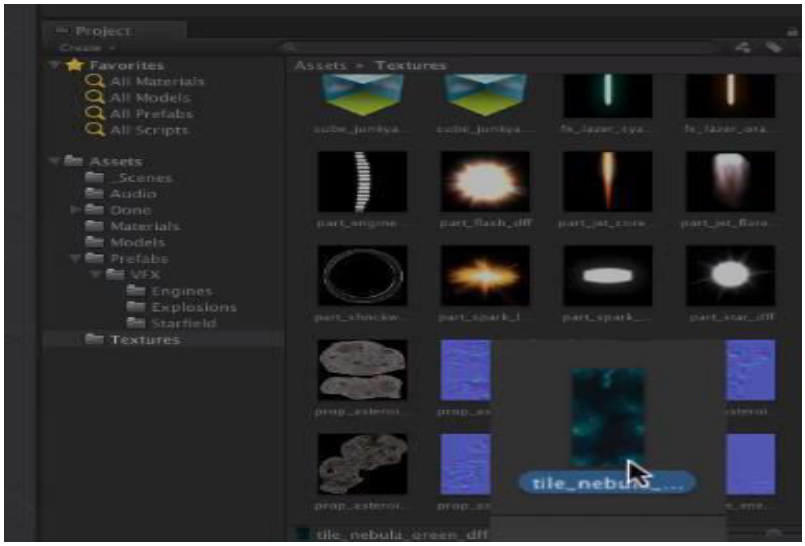
Click on player in the hierarchy->go to transform component ->deselect Player tab Create the quad to hold the background Image.

Go to Hierarchy->click on Create->Select Quad->Rename it as Background->reset the game object.

Change the rotation of background x-

Now Add The Texture To Our Background:

Go To Project ->Assets->Texture->Select Nebula-> Drag the Image and drop it on the scene view

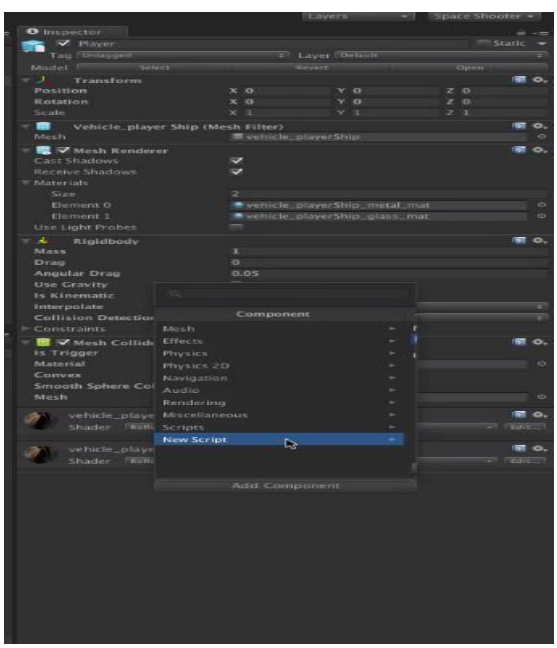


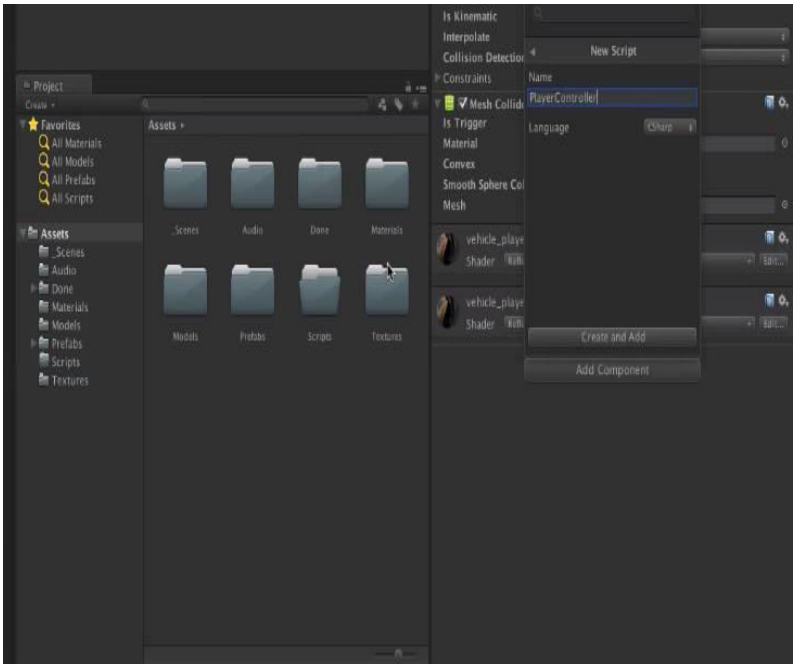
Change the Shedder: go to Shedder ->Unlit->Texture

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

Step 5:Moving the Player

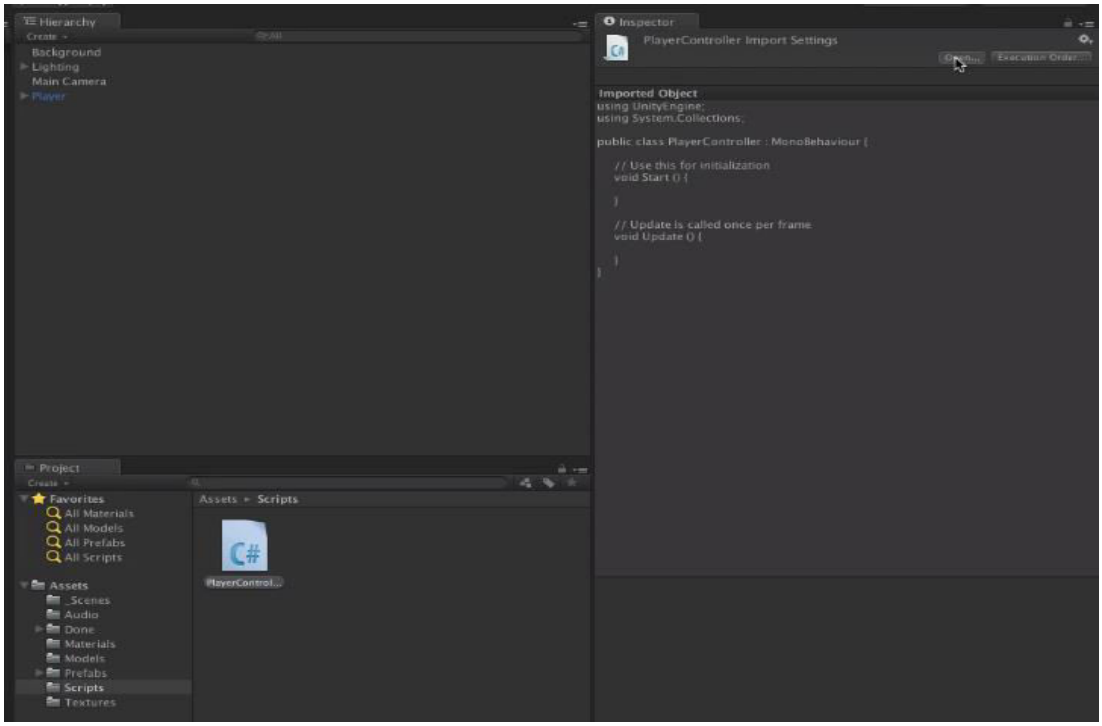
Go to Asset->click on create-> folder->give the name as Script->Press Enter->Now We Will Create a new Script to our Player Ship->Click on Player In the Hierarchy ->Go to Inspector Window->Click on Add Component Tab->select New Script->Give the Name to script as Player Controller->Click on Create and Add Button





The Script is created in c-sharp

Now Drag the Player Controller Script into the Script Folder

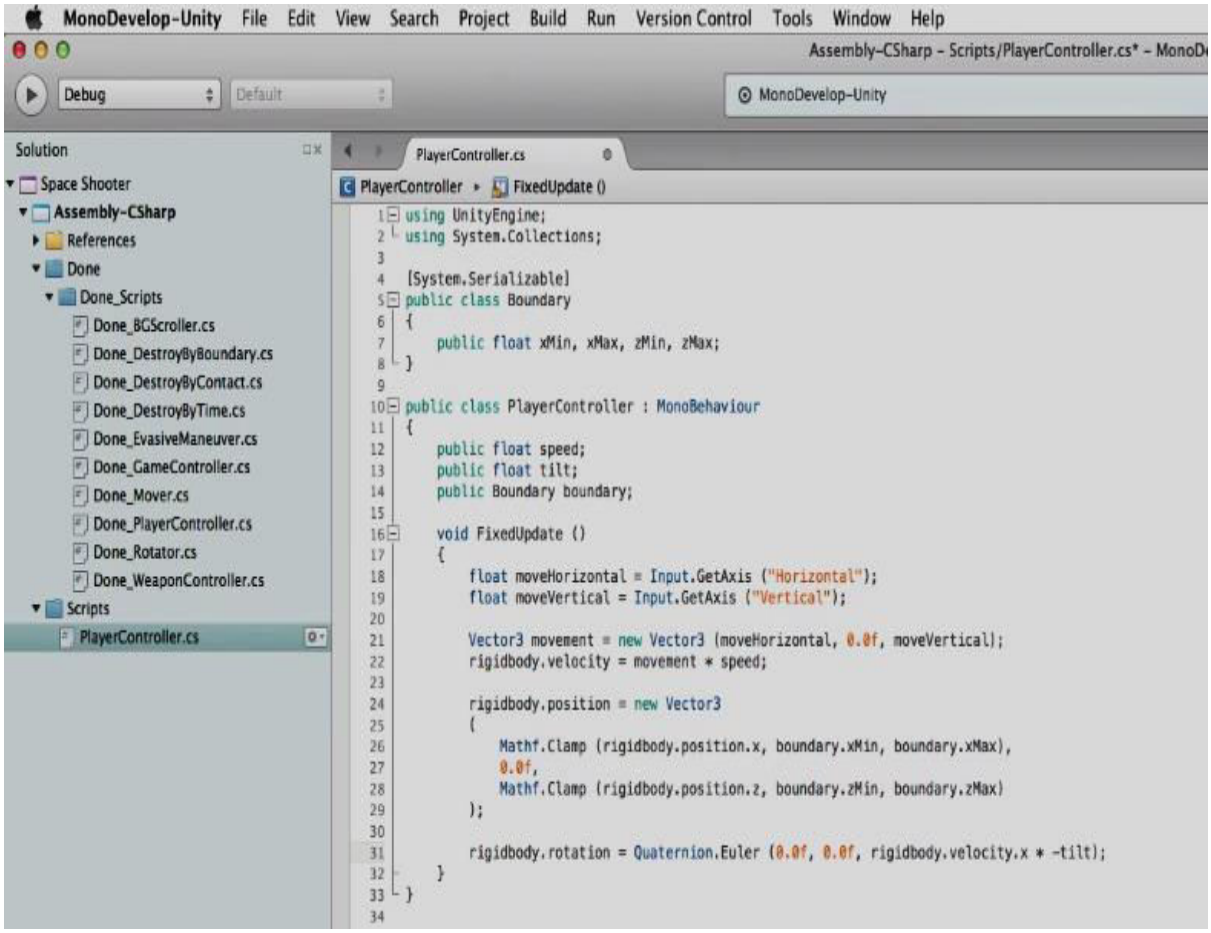


The Script will open in the mono Developer code.

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

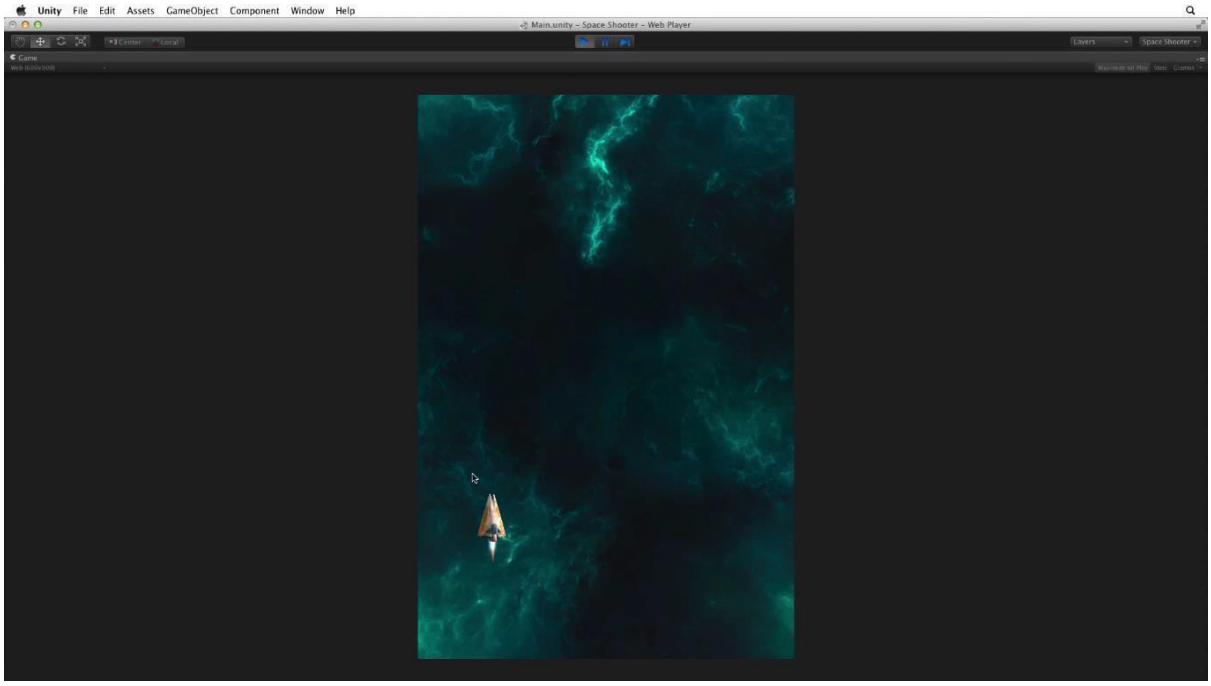
->z-min value as 4

->z-max value as -4

->Adjust the tilt value

->Set the speed to the ship

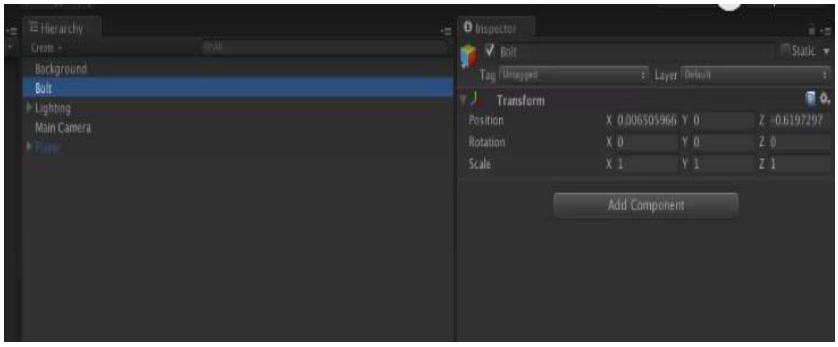
Run The Script From Edit Menu Button the output will be



Step 5:Creating Shots

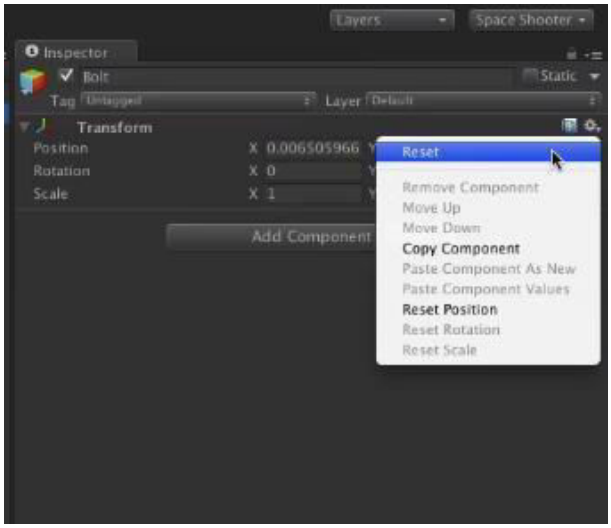
Now Will Create Shots to Our Player

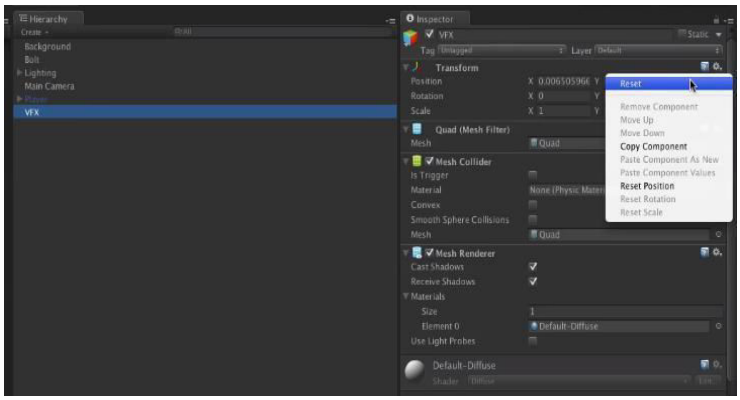
Now Create new Empty Game Object In the Hierarchy->Press Shift+Cntrl->Give the name to the Game Object (Bolt)



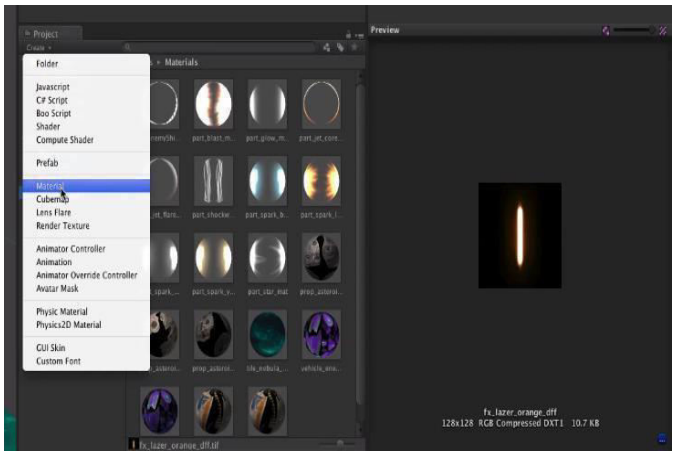
Click on Bolt From Hierarchy->Reset the game object of Transform to Origin

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



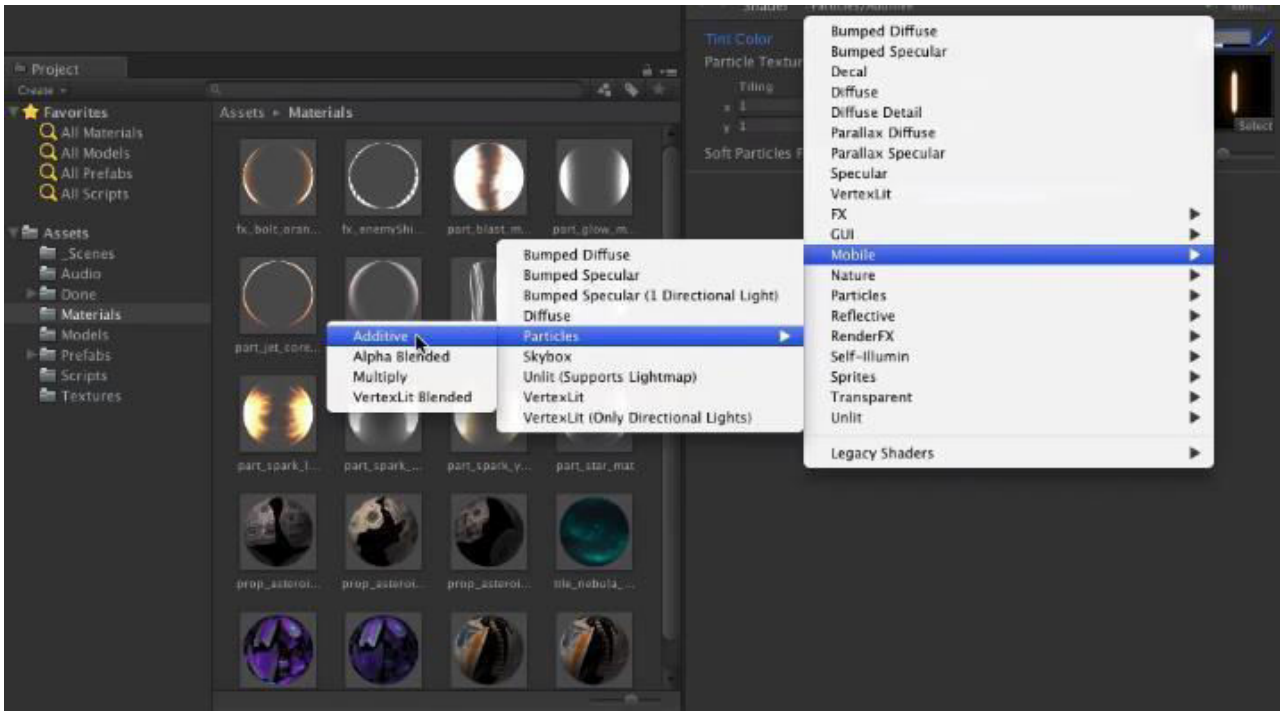
Now will add the Texture Into the Material

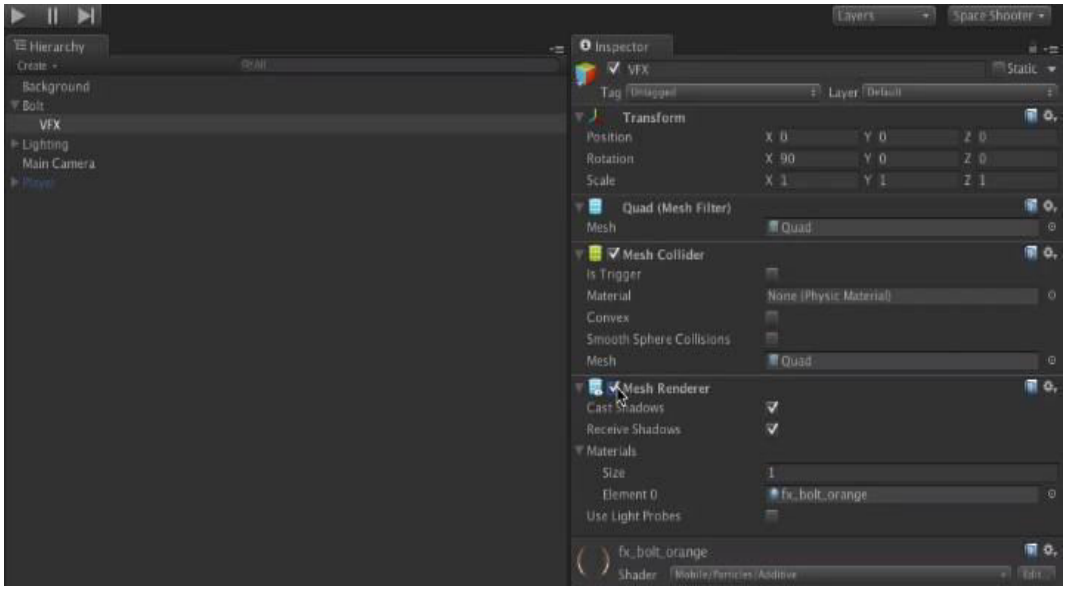
Go to Inspector Window ->Click on fx-bolt-orange->click on Select->click on the Texture you want to add into material.

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

Go to Inspector window->fs- bolt-orange->Shader->Mobile->Particles->Additive

Select Bolt from Hierarchy->Go to Inspector->Click on Add Component->Physics-> Now Go to Hierarchy->Click on VFX->Go to Inspector->Select->Mesh Renderer tab



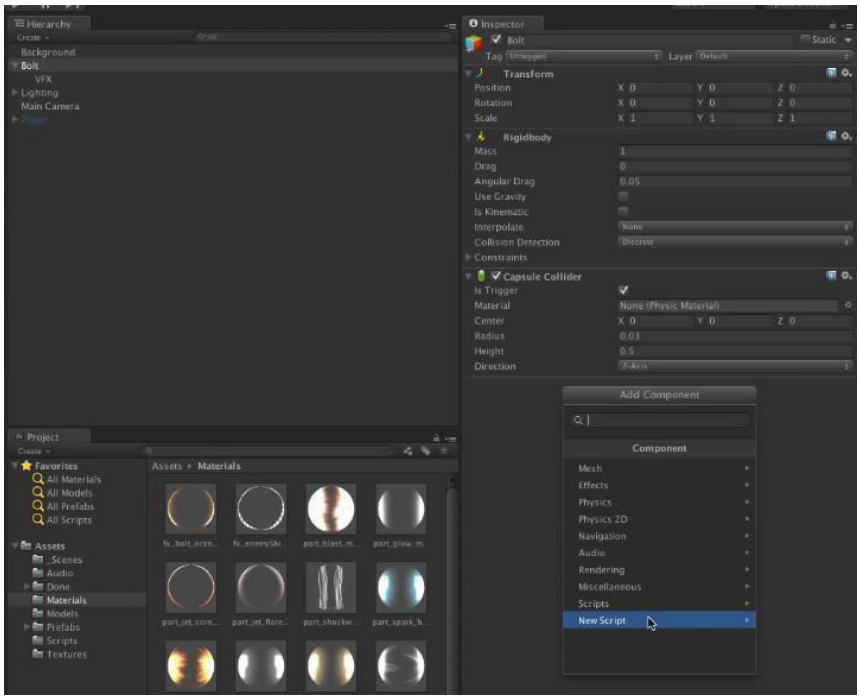


In Inspector Window->Go to mesh Collider->Click on Setting Button->Click on Remove Component

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.

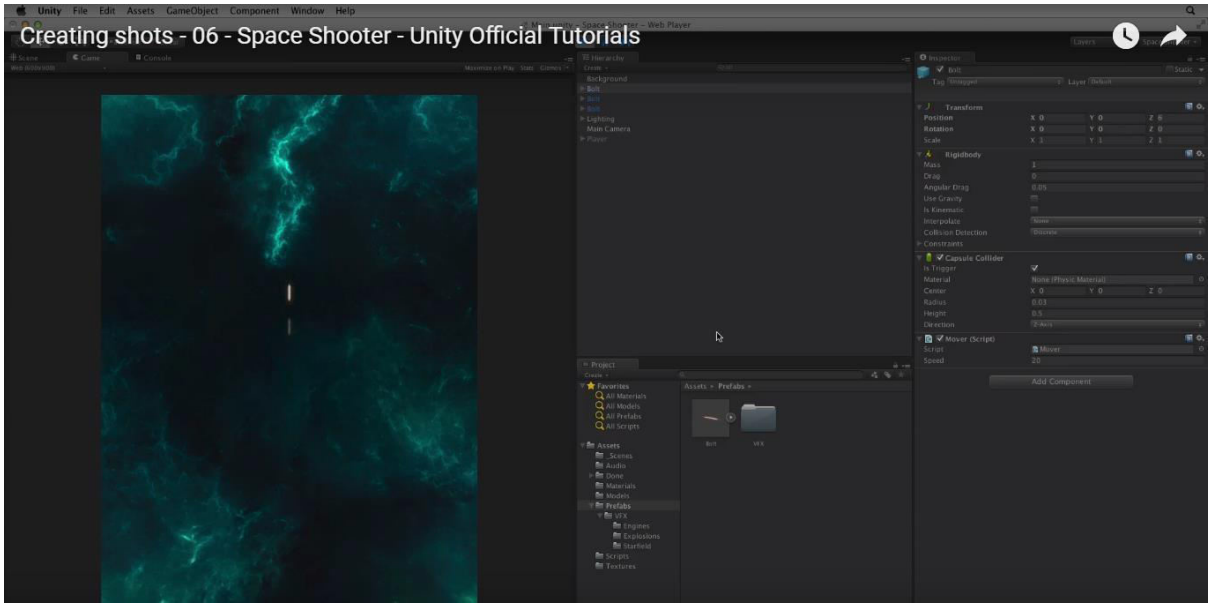
Click on Bolt In the hierarchy->Go to Inspector Window->Click On Add Component->Click on New Script->Give The Name to the Script As Mover->Press Enter.



Go toAssets-> Move the Script File into The Script Folder->open the Mover Script-> Write the following Lines of Code into the Mover Script->Save the Scrip t And Come to The unity Window

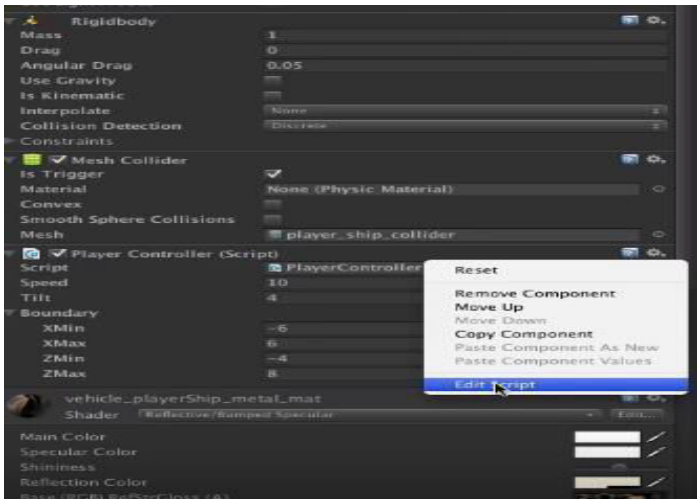
Drag the Bolt Game Object From Hierarchy to the Asset Prefab->Set the Script Speed As 20 Inspector Window.

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

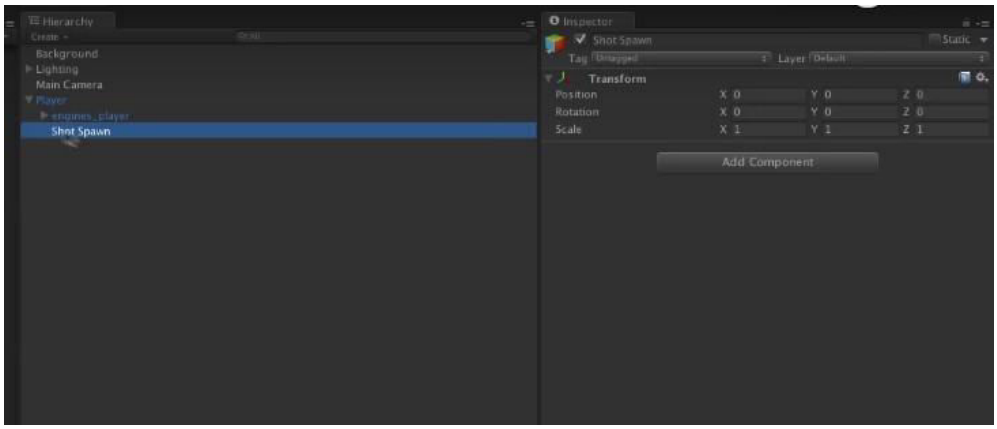
Select Player From Hierarchy->go to Inspector Window->Go to Player Controller Script->Click on Setting Tab->Select Edit Script Option



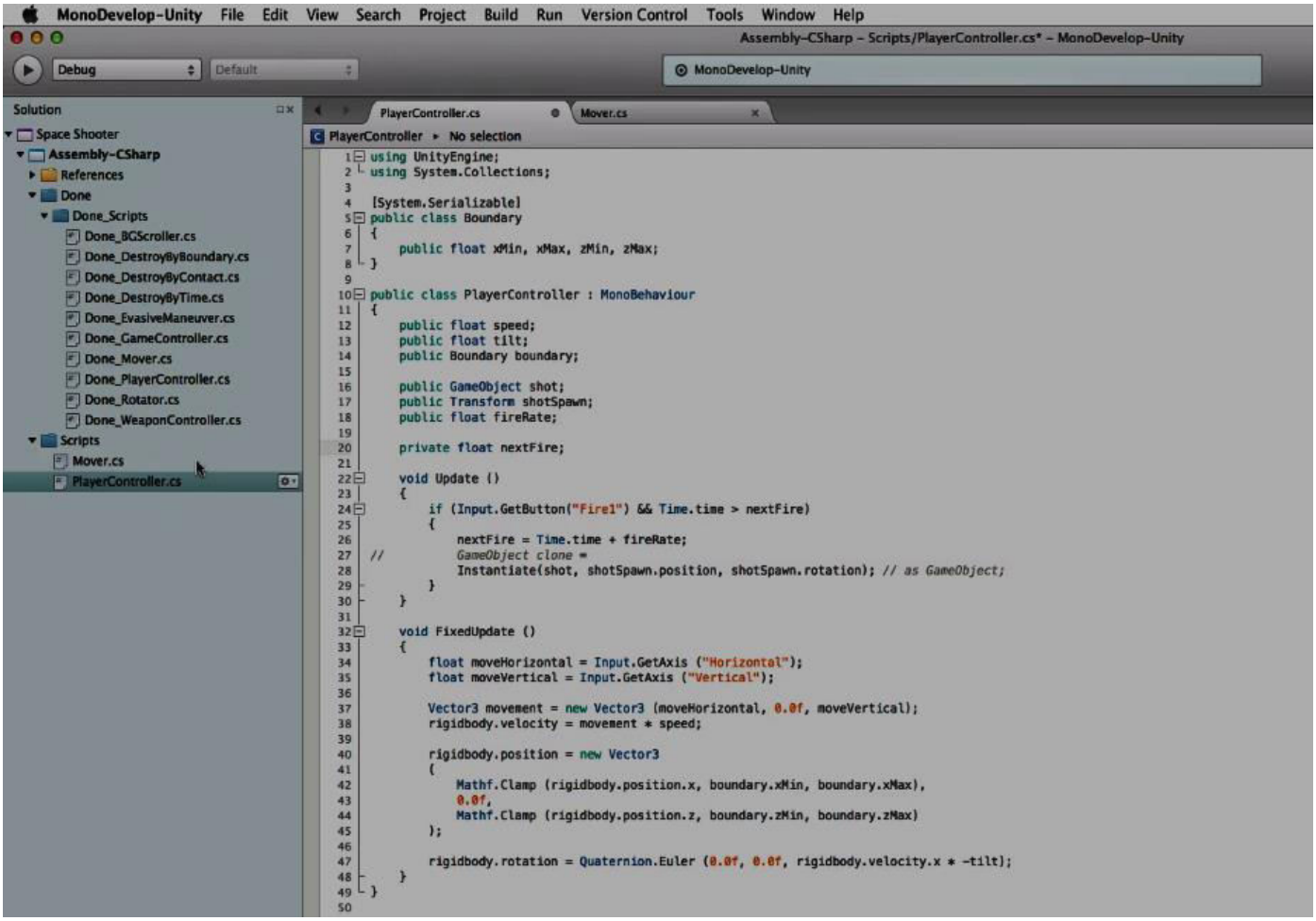
Add The Following Lines Of Code Into the Script.

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.



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 9

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

CODE:

1. PlayerController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class playerController : MonoBehaviour
{
   public float speed;
   private Rigidbody rb;
   public Text CountText;
   public Text WinText;
   private int count;
   private void Start()

   {
      rb = GetComponent<Rigidbody>();
      count=0;
      setCountText();
      WinText.text=" ";
   }

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

      Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);
      rb.AddForce(movement*speed);
   }
   private void OnTriggerEnter(Collider other)
   {
      if(other.gameObject.CompareTag("pickup"))
      {
         other.gameObject.SetActive(false);
         count=count+1;
         setCountText();
      }
   }
   void setCountText()
```

```
    {
        CountText.text= "Count: " +count.ToString();
        if(count>=5)
        {
            WinText.text="You win";
        }
    }
}
```

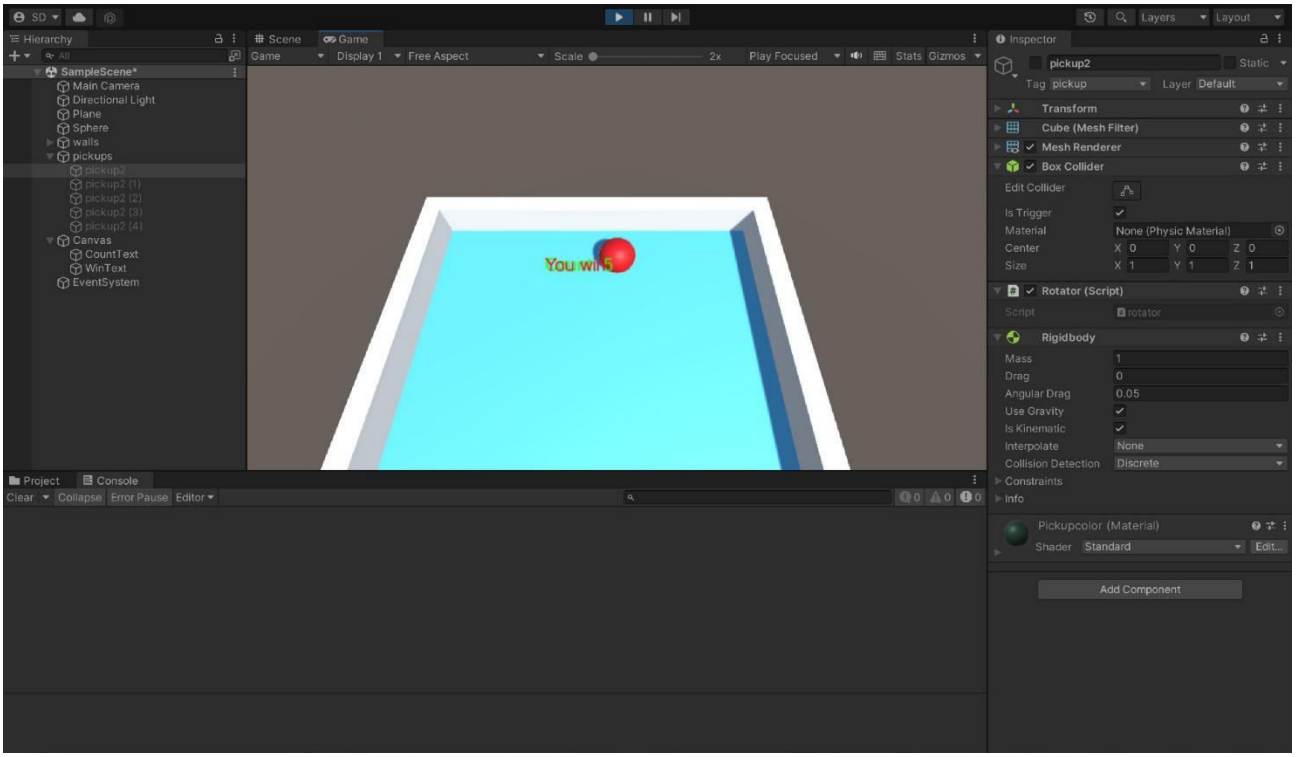
2. CamerController.cs

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

public class cameraController : MonoBehaviour
{
   public GameObject player;
    private Vector3 offset;
    private void Start()
    {
        offset = transform.position - player.transform.position;

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

            }
}
```

3. Rotator.cs

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

public class rotator : MonoBehaviour
{
    void Update()
    {

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

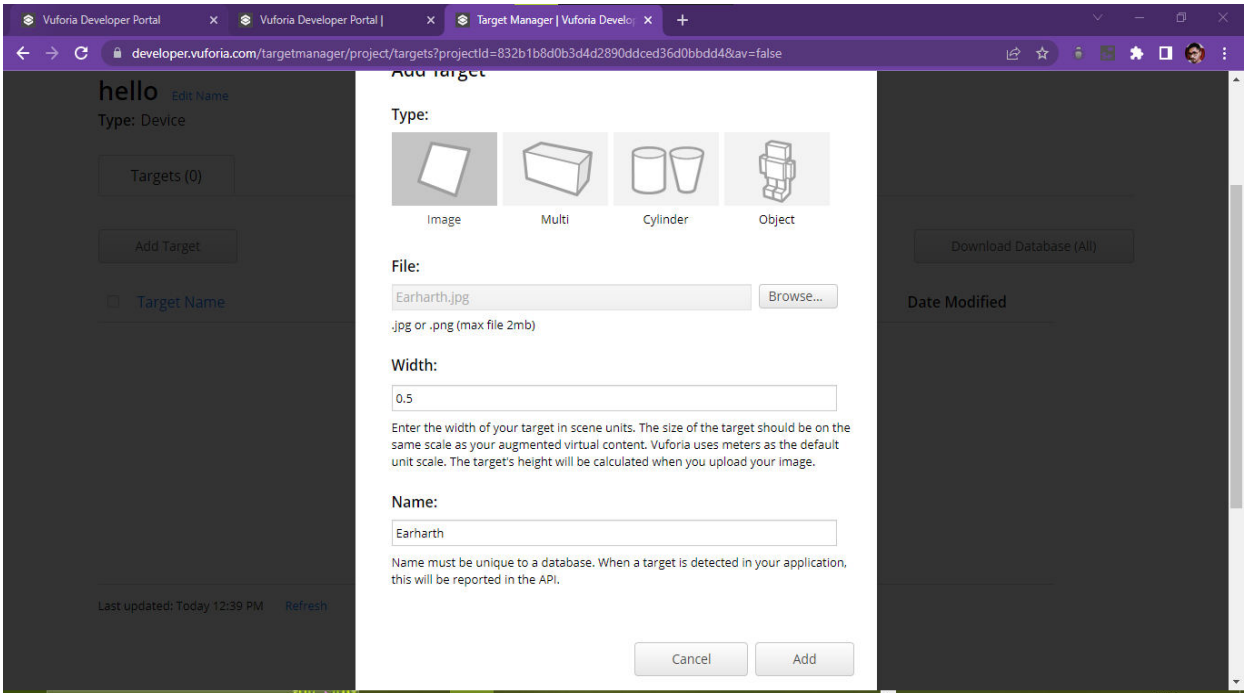
OUTPUT:


Hierarchy
SampleScene*
Main Camera
Directional Light
Plane
Sphere
walls
pickups
pickup2
pickup2 (1)
pickup2 (2)
pickup2 (3)
pickup2 (4)
Canvas
CountText
WinText
EventSystem
Scene
Game
Display 1
Free Aspect
Scale
Play Focused
Stats
Gizmos
You win
Project
Console
Clear
Collapse
Error Pause
Editor
Inspector
pickup2
Static
Tag pickup
Layer Default
Transform
Cube (Mesh Filter)
Mesh Renderer
Box Collider
Edit Collider
Is Trigger
Material
None (Physic Material)
Center
Size
Rotator (Script)
Script
rotator
Rigidbody
Mass
Drag
Angular Drag
Use Gravity
Is Kinematic
Interpolate
Collision Detection
Discrete
Constraints
Info
Pickupcolor (Material)
Shader
Standard
Edit...
Add Component

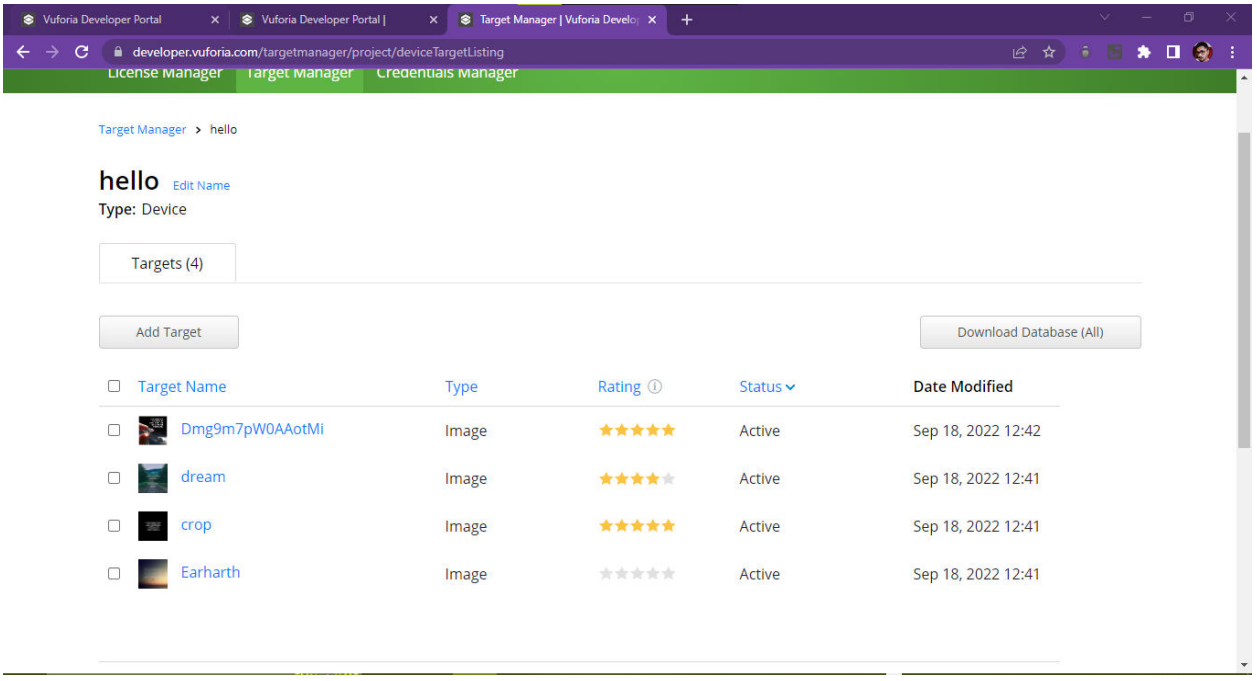
Practical 10

AIM: Creating AR Content with Vuforia.

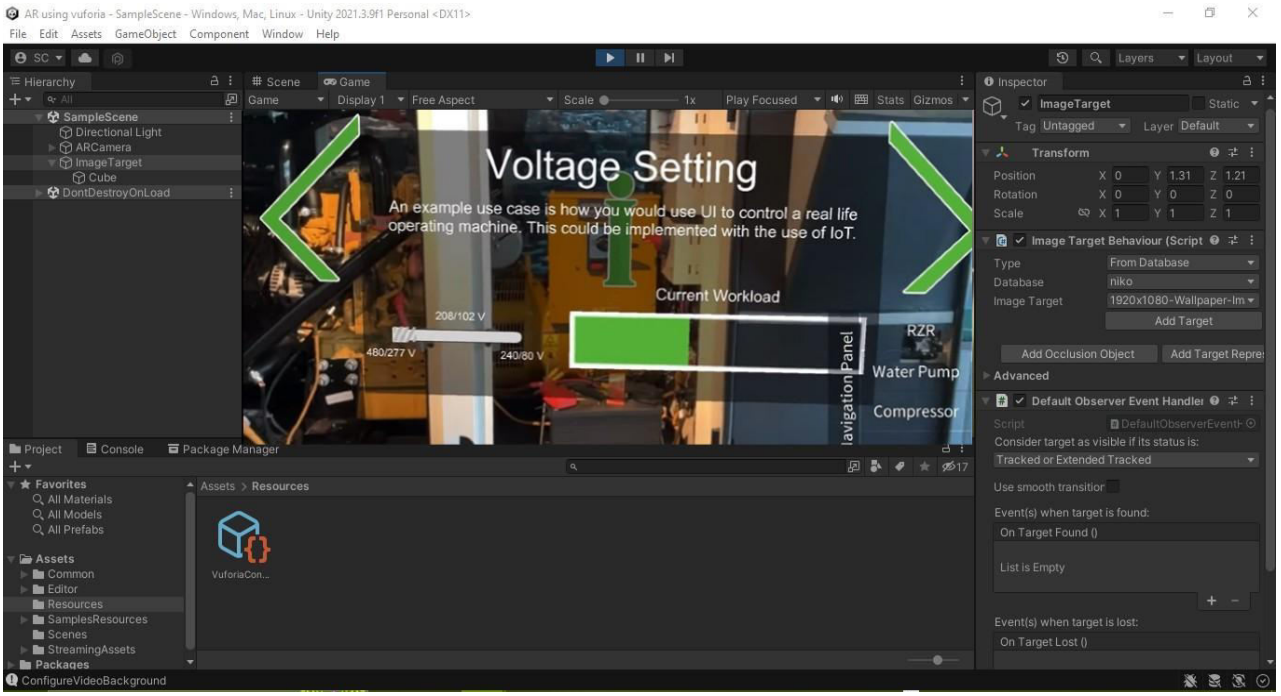
Adding Image to the Database:



hello Database:



OUTPUT:


AR using vuforia - SampleScene - Windows, Mac, Linux - Unity 2021.3.9f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
SampleScene
Directional Light
ARCamera
ImageTarget
Cube
DontDestroyOnLoad
Scene
Game
Display 1
Free Aspect
Play Focused
Stats
Gizmos
Voltage Setting
An example use case is how you would use UI to control a real life operating machine. This could be implemented with the use of IoT.
Current Workload
208/102 V
480/277 V
240/80 V
RZR
Water Pump
Compressor
Inspector
ImageTarget
Static
Tag Untagged
Layer Default
Transform
Position
Rotation
Scale
Image Target Behaviour (Script
Type
From Database
Database
niko
Image Target
1920x1080-Wallpaper-Im
Add Target
Add Occlusion Object
Add Target Repre
Advanced
Default Observer Event Handle
Consider target as visible if its status is:
Tracked or Extended Tracked
Use smooth transitior
Event(s) when target is found:
On Target Found ()
List is Empty
Event(s) when target is lost:
On Target Lost ()
Project
Console
Package Manager
Favorites
All Materials
All Models
All Prefabs
Assets
Common
Editor
Resources
SamplesResources
Scenes
StreamingAssets
Packages
Assets > Resources
VuforiaCon...
ConfigureVideoBackground