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".

Step 4:

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

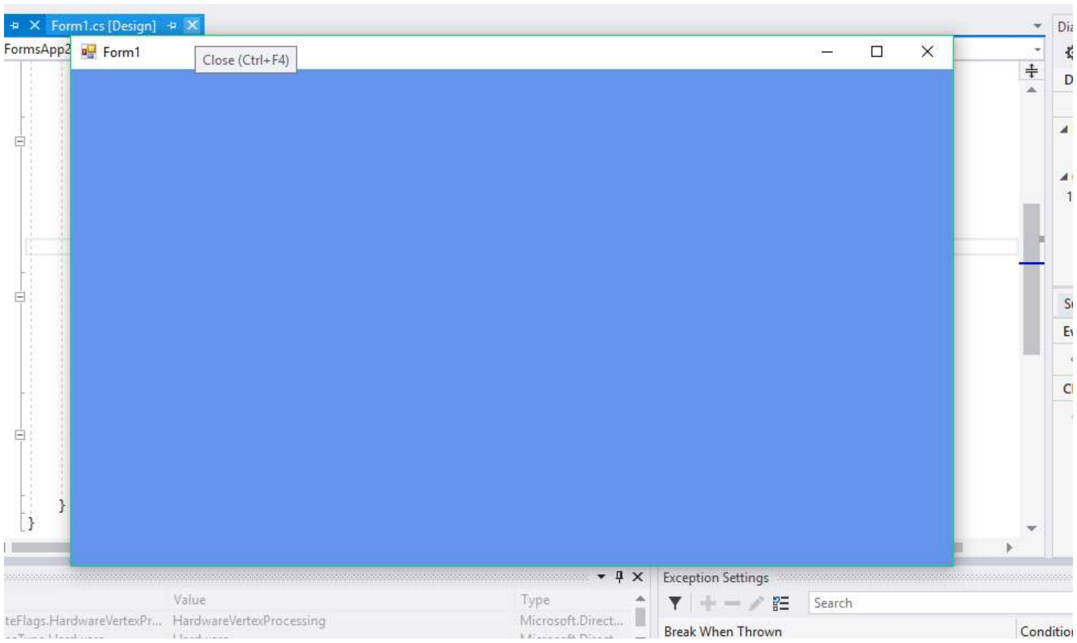
Step 5:

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

```
using System;
usingSystem.Collections.Generic;
usingSystem.ComponentModel;
usingSystem.Data;
usingSystem.Drawing;
usingSystem.Text;
usingSystem.Windows.Forms;
usingMicrosoft.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.

Output:


Form1.cs [Design]
FormsApp2
Form1
Close (Ctrl+F4)
Value
Type
teFlags.HardwareVertexPr...
HardwareVertexProcessing
Microsoft.Direct...
Exception Settings
Search
Break When Thrown
Conditio

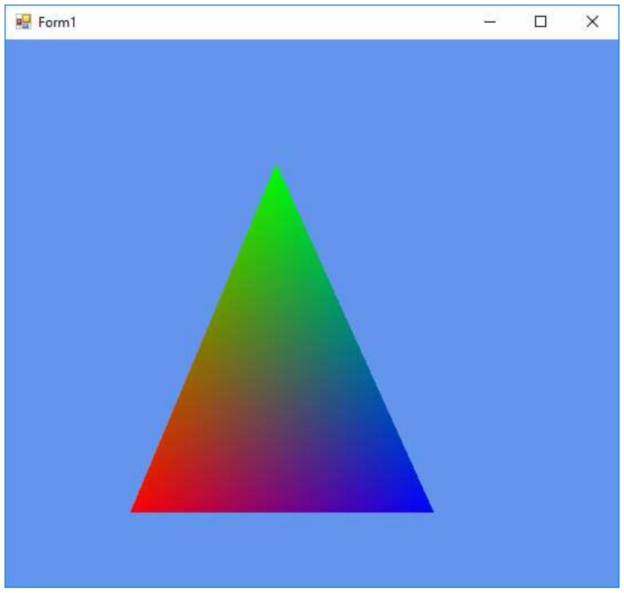
Practical No. 2

Aim: **Draw a triangle using Direct3D 11**

Solution:

```
using System;
usingSystem.Collections.Generic;
usingSystem.ComponentModel;
usingSystem.Data;
usingSystem.Drawing;
usingSystem.Text;
usingSystem.Windows.Forms;
usingMicrosoft.DirectX;
using Microsoft.DirectX.Direct3D;
namespace GP_P2
{
public partial class Form1 : Form
  {
     Microsoft.DirectX.Direct3D.Device device;
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);
     }
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:



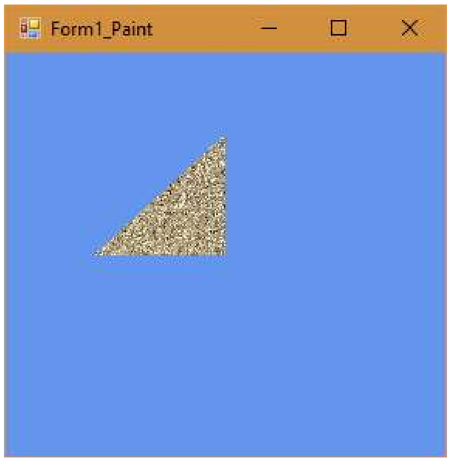
Practical No. 3

Aim: Texture the triangle using Direct3D 11

Solution:

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

namespace Gp_prac3
{
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:



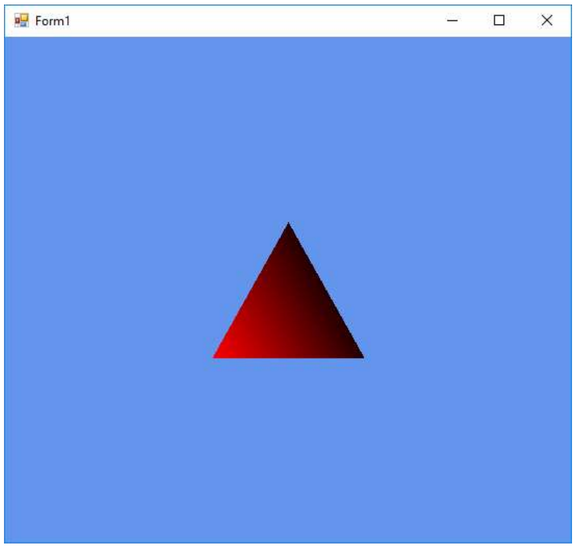
Practical No. 4

Aim: Programmable Diffuse Lightning using Direct3D 11

Solution:

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

namespace GP_P2
{
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:

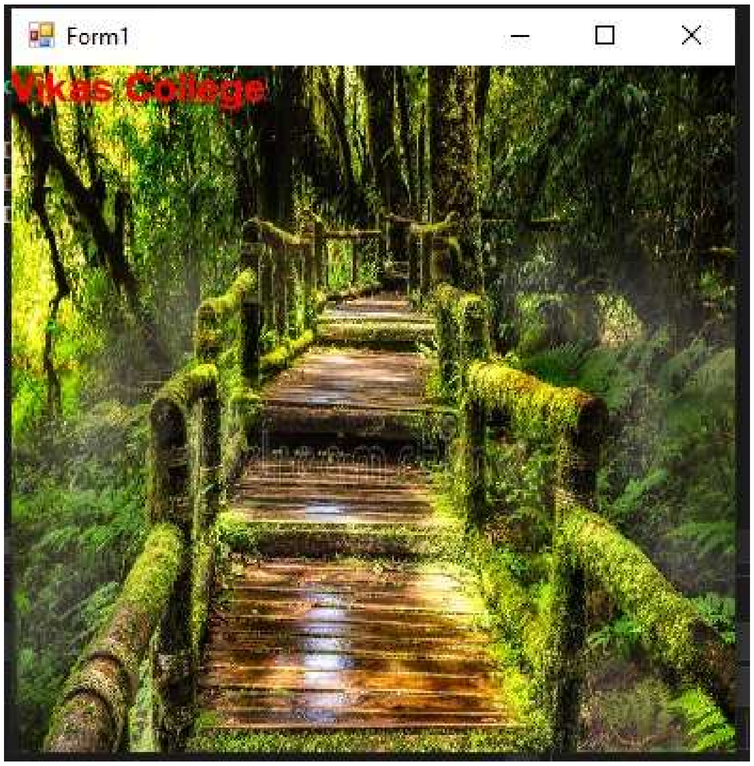


Practical No. 5

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 Practical_5
{
   public partial class Form1 : Form
   {
      private Microsoft.DirectX.Direct3D.Device device;
      private Microsoft.DirectX.Direct3D.Texture texture;
      private Microsoft.DirectX.Direct3D.Font font;
      public Form1()
      {
         InitializeComponent();
         InitDevice();
         InitFont();
         LoadTexture();
      }

      public void LoadTexture()
      {
         texture = TextureLoader.FromFile(device, "C:\\Users\\Sujith\\Documents\\Visual Studio
2022\\Practical 5\\forest.jpg", 400, 400, 1, 0, Format.A8B8G8R8, Pool.Managed, Filter.Point,
Filter.Point,Color.Transparent.ToArgb());
      }
      public void InitFont()
      {
         System.Drawing.Font f = new System.Drawing.Font("Arial", 16f,FontStyle.Bold);
         font = new Microsoft.DirectX.Direct3D.Font(device, f);
      }
      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.BeginScene();
         using (Sprite s = new Sprite(device))
         {
             s.Begin(SpriteFlags.AlphaBlend);
             s.Draw2D(texture, new Point(0, 0), 0f, new Point(0, 0), Color.White);
             font.DrawText(s, "Vikas College", new Point(0, 0), Color.Red);
             s.End();
         }
         device.EndScene();
         device.Present();
      }

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

Output:



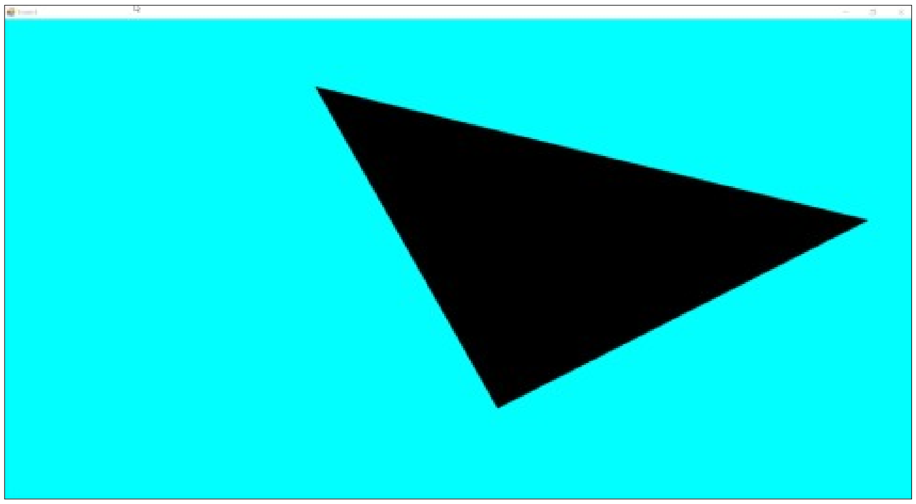
Practical No. 6

Aim: **Specular Lightning (Programmable Spot Lightning using Direct3D 11)**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX.Direct3D; //OUR CODE
using Microsoft.DirectX; //OUR CODE

namespace p11
{
        public partial class Form1 : Form
        {
                 private Device device; //OUR CODE
                private float angle = 0f; //OUR CODE
                public Form1()
                {
                        InitializeComponent();
                        InitDevice(); //OUR CODE

                this.SetStyle(ControlStyles.AllPaintingInWmPaint | ControlStyles.Opaque,
true); //OUR CODE
                }
                private void InitDevice() //OUR CODE
                {
                        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() //OUR CODE
        {
            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();
                    Vector3 v;
                    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(); //OUR CODE
                }
        }
}
```

Output:

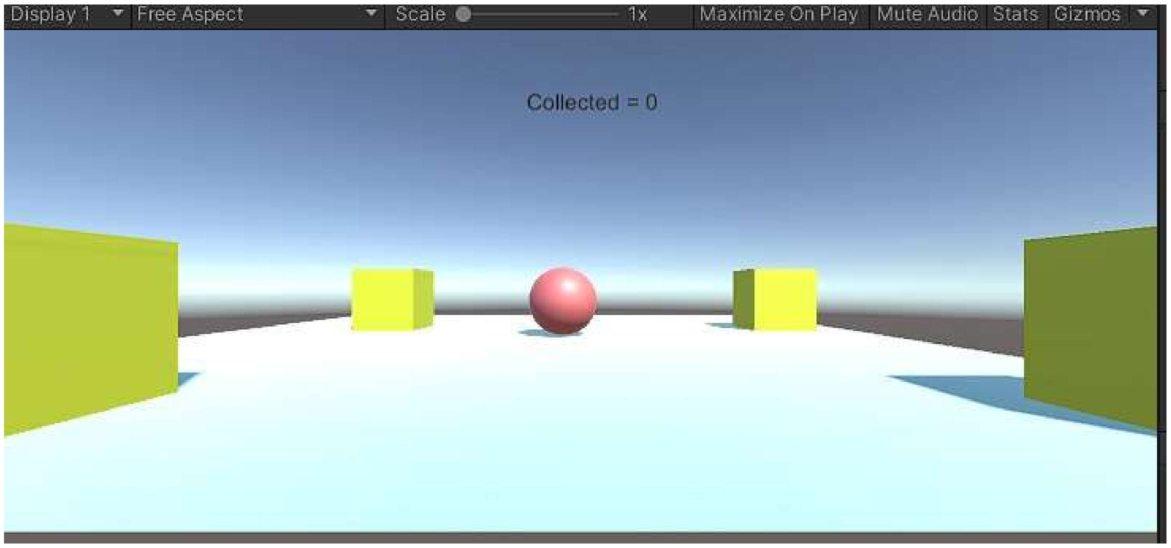


Practical No. 7

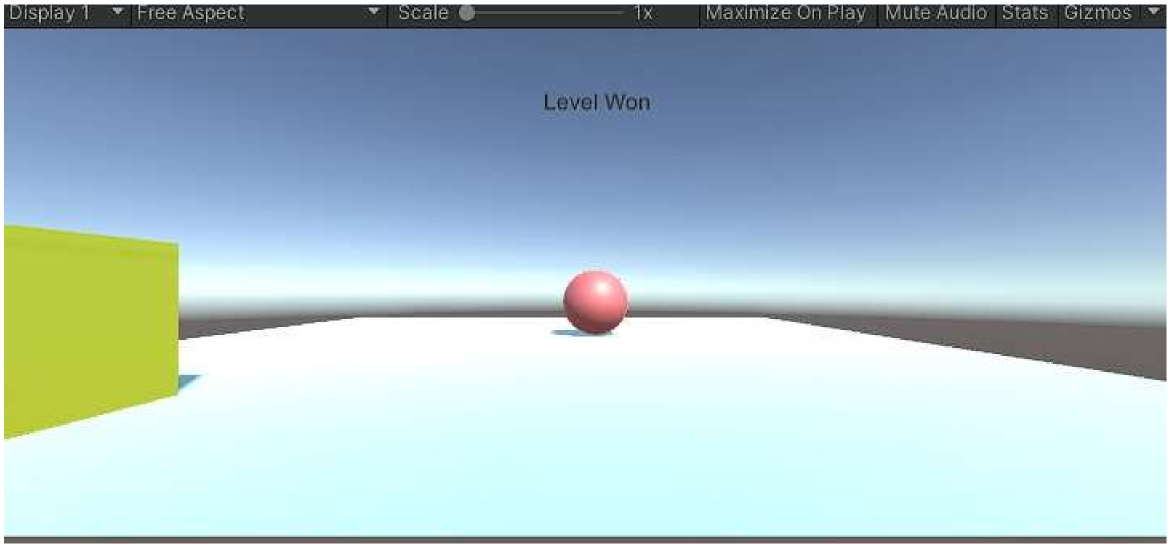
Aim: Roll ball Tutorial

```
using UnityEngine;
using UnityEngine.UI;

public class Move : MonoBehaviour
{
   public Rigidbody rb;
   public float h, v, speed = 5.0f;
   public int count;
   public Text ct;
   // Start is called before the first frame update
   void Start()
   {
      rb = GetComponent<Rigidbody>();
      count = 0;
      ct.text = "Collected = " + count;

   }

   // Update is called once per frame
   void Update()
   {
      h = Input.GetAxis("Horizontal");
      v = Input.GetAxis("Vertical");
   }

   private void FixedUpdate()
   {
      rb.AddForce(new Vector3(h,0.0f,v)*speed);
   }

   private void OnTriggerEnter(Collider other)
   {
      if (other.gameObject.CompareTag("Coins"))
      {
         other.gameObject.SetActive(false);
         count++;
         ct.text = "Collected = " + count;
      }

      if (count == 3)
      {
         ct.text = "Level Won";

      }
   }
}
```

Output:



Collected = 0



Level Won

Practical No. 8

Aim: **UFO Tutorial**

Step 1:

```
using UnityEngine;
using UnityEngine.UI;

public class Player : MonoBehaviour
{
   public Rigidbody2D rb2d;
   float moveH, moveV;
   public float speed;
   public string pickUpTag = "PickUp";
   int pickUpTotal, count;
   public Text ShowText;

   void Start()
   {
      rb2d = GetComponent<Rigidbody2D>();
      count = 0;
      pickUpTotal = 5;
      coinCount();


   }

   void FixedUpdate()
   {
      moveH = Input.GetAxis("Horizontal");
      moveV = Input.GetAxis("Vertical");
      var movement = new Vector2(moveH, moveV) * speed * Time.deltaTime;
      rb2d.AddForce(movement);
```

```
    }

    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("PickUp"))
        {
            other.gameObject.SetActive(false);
            count++;
            coinCount();
        }
    }
    private void coinCount()
    {
        ShowText.text = "Count : " + count;
        if (count >= pickUpTotal)
        {
            ShowText.text = "Level Won";
        }
    }
}
```

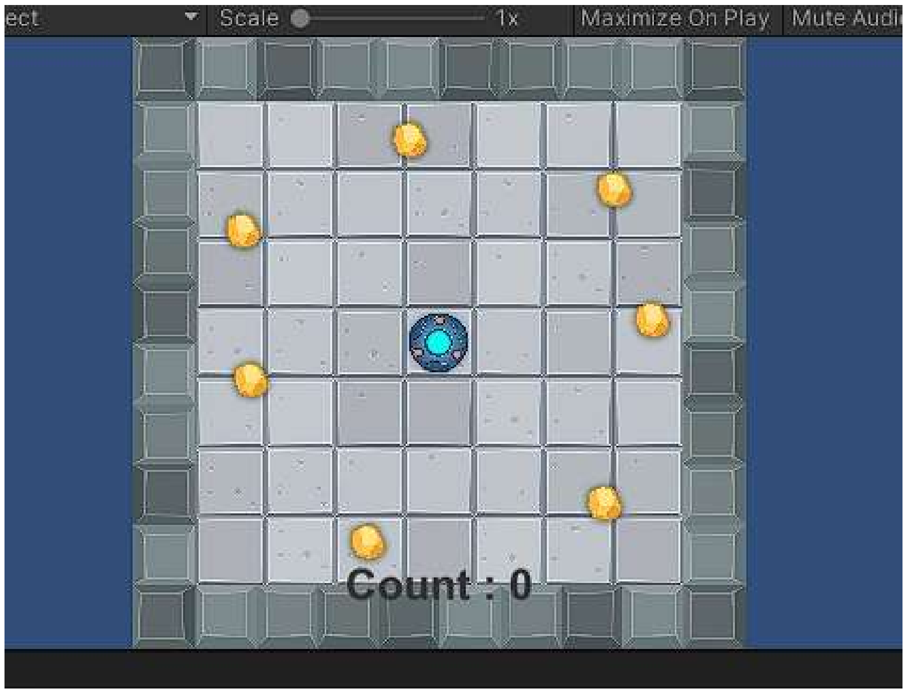
Step 2: Following the player with camera..

```
using UnityEngine;

public class CamerController : MonoBehaviour
{
    public Transform player;
    private Vector3 offset;
    void Start()
    {
        offset = transform.position - player.position;
    }

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

<u>Step 3:</u> We need to rotate our object or animate, then create a new scripts

```
using UnityEngine;

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

Output:


ect
Scale
1x
Maximize On Play
Mute Audi
Count : 0