Game Programming

krunal dhavle
TYCS-713

Name : Krunal Dhavle

Class : TYCS

Roll No : 713

Subject : Game Programming Practicals

Teacher : Karishma Jain Maam

INDEX

Date:20/08/2020

Practical no 1

AIM: Create a 2D UFO Game using the Unity Engine.

Code:

playerController.cs

```
using System.Collections;
using UnityEngine;
using System.Collections.Generic;
using UnityEngine.UI;
public class PlayerController : MonoBehaviour {
public float speed;
public Text countText;
public Text winText;
private Rigidbody2D rb2d;
private int count;
void Start()
{
rb2d = GetComponent<Rigidbody2D>();
count = 0;
winText.text = "";
SetCountText ();
}
void FixedUpdate()
{
float moveHorizontal = Input.GetAxis("Horizontal");
float moveVertical = Input.GetAxis("Vertical");
Vector2 movement = new Vector2(moveHorizontal, moveVertical);
rb2d.AddForce(movement * speed);
}
void OnTriggerEnter2D(Collider2D other)
{
if (other.gameObject.CompareTag ("PickUp"))
{
other.gameObject.SetActive (false);
count = count + 1;
SetCountText ();
}
}
void SetCountText()
{
countText.text="Count: " + count.ToString();
if (count >= 12) {
winText.text = "You Win!!!" ;
}
}}
```

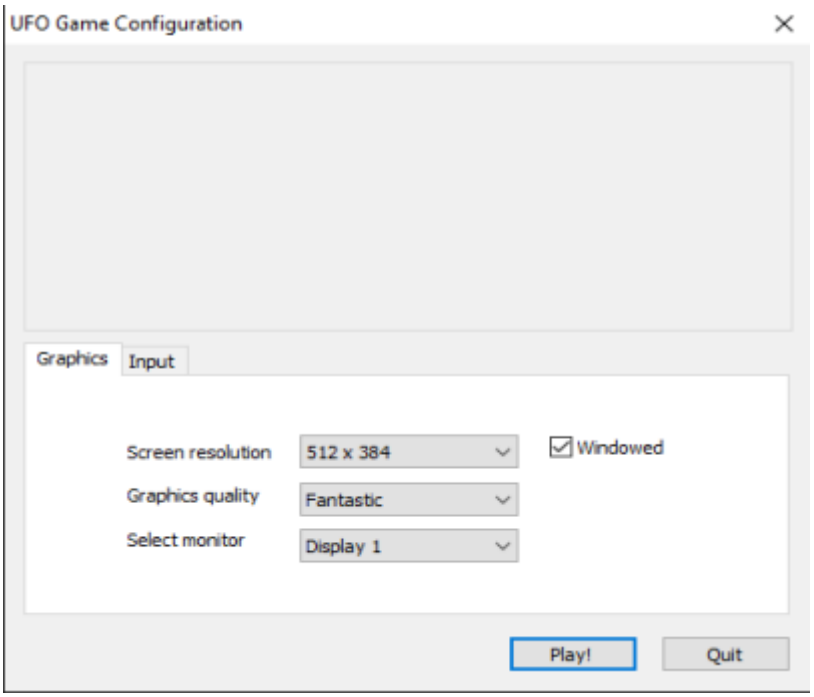
CameraController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CameraController : MonoBehaviour {
public GameObject player ;
private Vector3 offset;
//us 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;
}
}
```

Rotator.cs

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

output


UFO Game Configuration
Graphics
Input
Screen resolution
512 x 384
Windowed
Graphics quality
Fantastic
Select monitor
Display 1
Play!
Quit


UFO Game
Count: 0


UFO Game
Count: 12
You Win!!!

Date:03/09/2020

Practical no 2

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

Steps:-

1. Create a new project and select a windows form application(.Net Framework 2.0-3.5).
2. Right click on the properties → click on open → click build → select platform target → x86 or add new
3. Click on view code on form 1(design) or press F7.
4. Go to the solution explorer → right click on project name → select add reference .
5. Click on browse and add the required dll files.
6. Code the required files.
7. Add the paint method for changing the appearance .
8. Change the window name and icon if possible.
9. Disable the Exception Settings option such as LoaderLock.
10. Run the app.

Code:-

Program.cs file

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace WindowsFormsApp5
{
   static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
      }
   }}
```

Form1.cs file

```
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;

namespace WindowsFormsApp5
{
   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);


      }

      public void Render()
      {
         device.Clear(ClearFlags.Target, Color.RoyalBlue, 0, 1);
         device.Present();
      }

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

Output

Date:10/09/2020

Practical no 3

AIM: Buffers, Shaders and HLSL (Draw a triangle/rectangle using Direct3D 11)

Steps:-

11. Create a new project and select a windows form application(.Net Framework 2.0-3.5).
12. Right click on the properties → click on open → click build → select platform target → x86 or add new
13. Click on view code on form 1(design) or press F7.
14. Go to the solution explorer → right click on project name → select add reference .
15. Click on browse and add the required dll files.
16. Code the required files.
17. Add the paint method for changing the appearance .
18. Change the window name and icon if possible.
19. Disable the Exception Settings option such as LoaderLock.
20. Run the app.

Code:-

Program.cs file

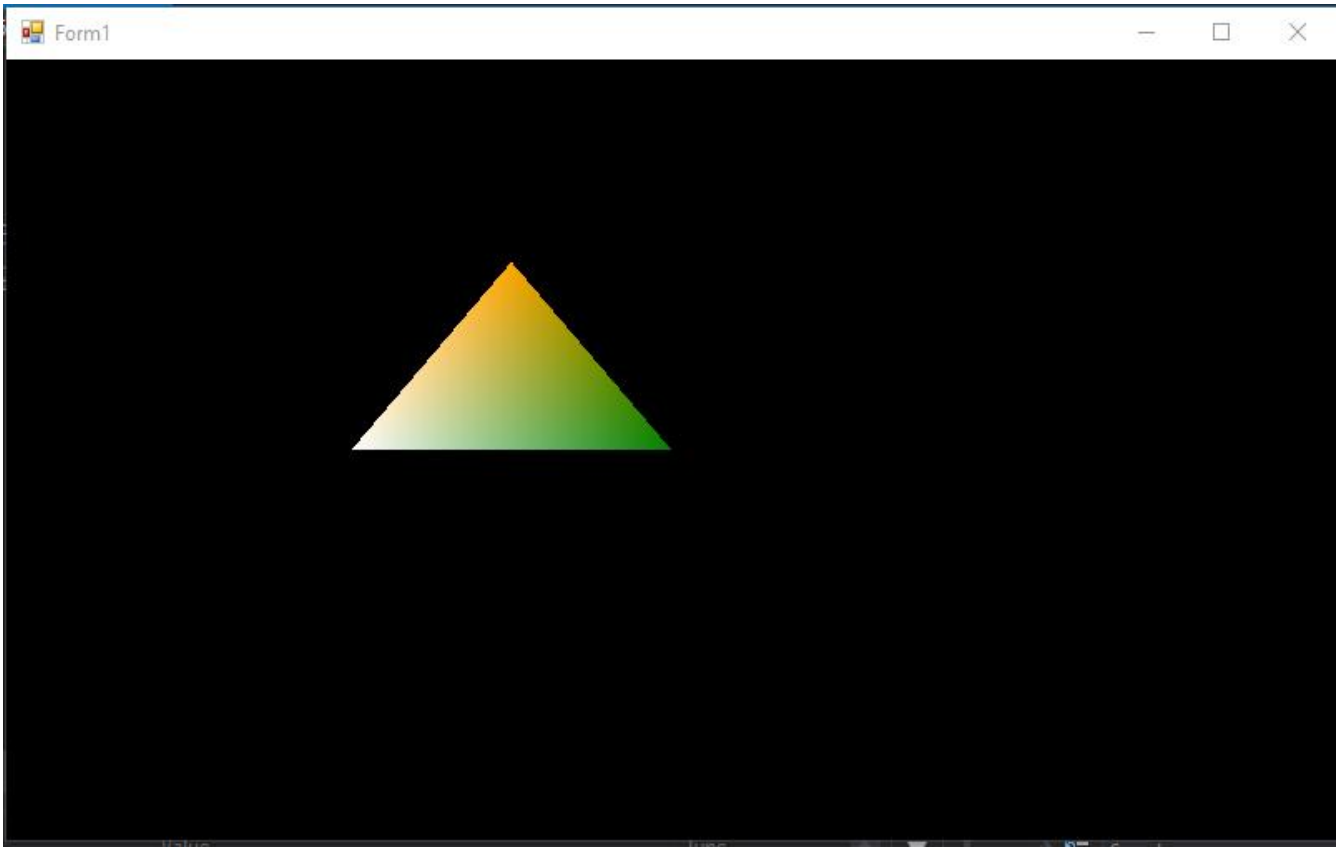
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using Microsoft.DirectX.Direct3D;

namespace WindowsFormsApp6
{
   static class Program
   {

      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
      }
   }
}
```

Form1.cs file

```
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;
using Microsoft.DirectX.Direct3D;

namespace WindowsFormsApp6
{
   public partial class Form1 : Form
   {
      private Device device;
     // Microsoft.DirectX.Direct3D.Device device;
     private CustomVertex.PositionColored[] vertex = new  CustomVertex.PositionColored [3];
       public Form1()
      {
         InitializeComponent();
      }
      private void Form1_Paint(object sender, PaintEventArgs e)
      {
         device.Clear(ClearFlags.Target, Color.Black, 1, 0);
         device.BeginScene();
         device.VertexFormat = CustomVertex.PositionColored.Format;
         device.DrawUserPrimitives(PrimitiveType.TriangleList, vertex.Length / 3, vertex);
         device.EndScene();
         device.Present();

      }

      private void Form1_Load(object sender, EventArgs e)
      {
         PresentParameters pp = new PresentParameters();
         pp.Windowed = true;
         pp.SwapEffect = SwapEffect.Discard;
         device = new Device(0, DeviceType.Hardware, this, CreateFlags.HardwareVertexProcessing,
pp);
         device.Transform.Projection = Matrix.PerspectiveFovLH(3.14f / 4 , device.Viewport.Width /
device.Viewport.Height, 1f , 1000f);
         device.Transform.View = Matrix.LookAtLH(new Vector3(0, 0, 20), new Vector3(), new Vec-
tor3(0, 1, 0));
         device.RenderState.Lighting = false;

         vertex[0] = new CustomVertex.PositionColored(new Vector3(0, 0, 0), Color.Green.ToArgb());
```

```
            vertex[1] = new CustomVertex.PositionColored(new Vector3(4, 0, 0), Color.White.ToArgb());
            vertex[2] = new CustomVertex.PositionColored(new Vector3(2, 4, 0), Color.Or-
ange.ToArgb());
        }
    }
}
```

Output:



Date:01/10/2020

Practical no 4

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

Steps:-

21. Create a new project and select a windows form application(.Net Framework 2.0-3.5).
22. Right click on the properties → click on open → click build → select platform target → x86 or add new
23. Click on view code on form 1(design) or press F7.
24. Go to the solution explorer → right click on project name → select add reference .
25. Click on browse and add the required dll files.
26. Code the required files.
27. Add the paint method for changing the appearance .
28. Change the window name and icon if possible.
29. Disable the Exception Settings option such as LoaderLock.
30. Run the app.

Code:-

Program.cs file

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using Microsoft.DirectX.Direct3D;

namespace WindowsFormsApp6
{
   static class Program
   {

      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
      }
   }
}
```

Form1.cs file

```
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 WindowsFormsApp11
{
   public partial class Form1 : Form
   {
      private Microsoft.DirectX.Direct3D.Device device;
      private CustomVertex.PositionTextured[] vertex =
          new CustomVertex.PositionTextured[3];
      private Texture tex;
      public Form1()
      {
         InitializeComponent();
      }

      private void Form1_Load(object sender, EventArgs e)
      {
         PresentParameters pp = new PresentParameters();
         pp.Windowed = true;
         pp.SwapEffect = SwapEffect.Discard;
         device = new Device(0, DeviceType.Hardware, this,
            CreateFlags.SoftwareVertexProcessing, 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, 3), new Vector3(), new Vector3(0, 1, 0));
         device.RenderState.Lighting = false;
         vertex[0] = new CustomVertex.PositionTextured(new Vector3(0, 1, 1), 0, 0);
         vertex[1] = new CustomVertex.PositionTextured(new Vector3(-1, -1, 1), -1, 0);
         vertex[2] = new CustomVertex.PositionTextured(new Vector3(1, -1, 1), 0, -1);
         tex = new Texture(device, new Bitmap("C:\\Users\\BlackBot\\source\\repos\\Win-
dowsFormsApp10\\shape1.jpg"), 0, Pool.Managed);
      }
       //performed by krunal 713
      private void Form1_Paint(object sender, PaintEventArgs e)
      {
         device.Clear(ClearFlags.Target, Color.White, 1, 0);
         device.BeginScene();
         device.SetTexture(0, tex);
```

```
            device.VertexFormat = CustomVertex.PositionTextured.Format;
            device.DrawUserPrimitives(PrimitiveType.TriangleList,
                vertex.Length / 3, vertex);
            device.EndScene();
            device.Present();
        }
    }
}
```

Output:

Date:01/10/2020

Practical no 5

AIM: Lightning (Programmable Diffuse Lightning using Direct3D 11)

Steps:-

31. Create a new project and select a windows form application(.Net Framework 2.0-3.5).
32. Right click on the properties → click on open → click build → select platform target → x86 or add new
33. Click on view code on form 1(design) or press F7.
34. Go to the solution explorer → right click on project name → select add reference .
35. Click on browse and add the required dll files.
36. Code the required files.
37. Add the paint method for changing the appearance .
38. Change the window name and icon if possible.
39. Disable the Exception Settings option such as LoaderLock.
40. Run the app.

Code:-

Program.cs file

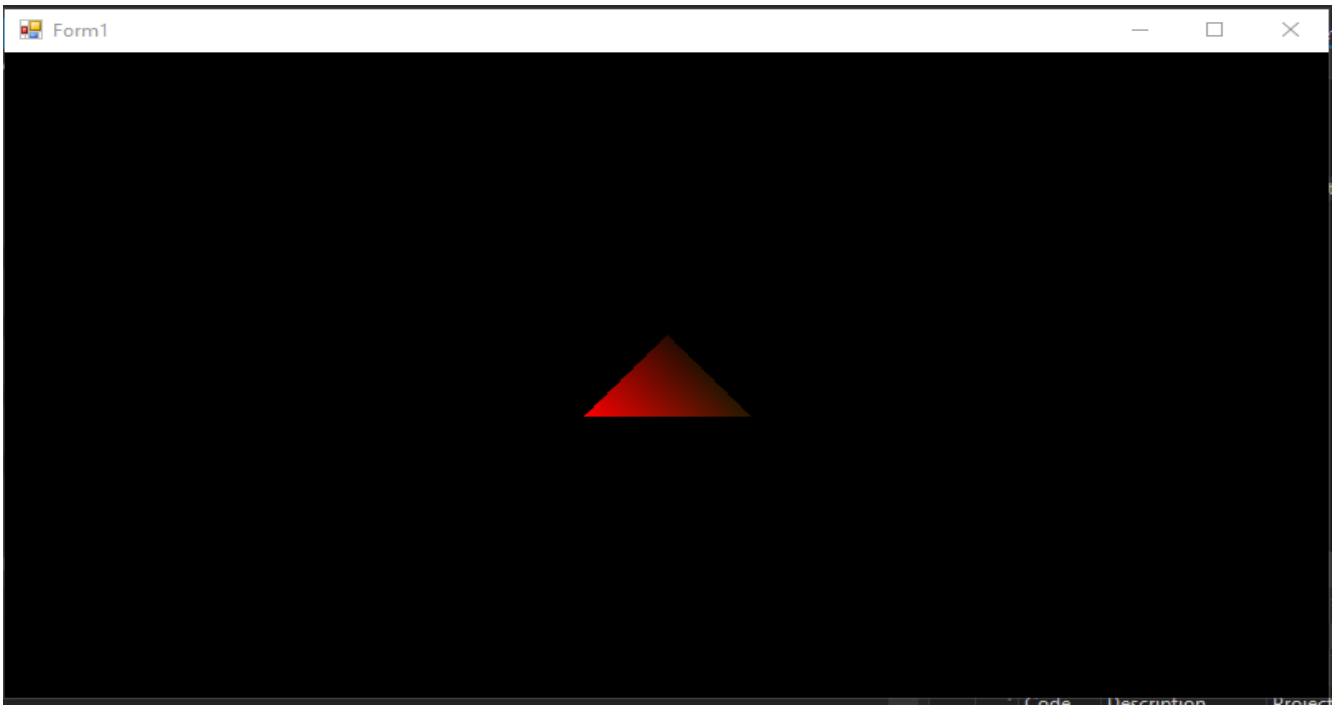
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using Microsoft.DirectX.Direct3D;

namespace WindowsFormsApp6
{
   static class Program
   {

      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
      }
   }
}
```

Form1.cs file

```
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;
using Microsoft.DirectX.Direct3D;

namespace WindowsFormsApp6
{
   public partial class Form1 : Form
   {
      private Microsoft.DirectX.Direct3D.Device device;
      // Microsoft.DirectX.Direct3D.Device device;
      private CustomVertex.PositionNormalColored[] vertex
          = new CustomVertex.PositionNormalColored[3];
      public Form1()
      {
         InitializeComponent();
      }
      private void Form1_Paint(object sender, PaintEventArgs e)
      {
         device.Clear(ClearFlags.Target, Color.Black, 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)
      {
         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 Vec-
tor3(0, 1, 0));
         device.RenderState.Lighting = false;
```

```
            vertex[0] = new CustomVertex.PositionNormalColored(new Vector3(0, 1, 1), new Vector3(1,
0, 1),
             Color.OrangeRed.ToArgb());

            vertex[1] = new CustomVertex.PositionNormalColored(new Vector3(-1, -1, 1), new Vector3(1,
0, 1),
             Color.Orange.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.White;
            device.Lights[0].Direction = new Vector3(0.8f, 0, -1);
            device.Lights[0].Enabled = true;


        }
    }
}
```

Output:



Date:08/10/2020

Practical no 6

AIM: Specular Lightning (Programmable Spot Lightning using Direct3D 11)

Steps:-

41. Create a new project and select a windows form application(.Net Framework 2.0-3.5).
42. Right click on the properties → click on open → click build → select platform target → x86 or add new
43. Click on view code on form 1(design) or press F7.
44. Go to the solution explorer → right click on project name → select add reference .
45. Click on browse and add the required dll files.
46. Code the required files.
47. Add the paint method for changing the appearance .
48. Change the window name and icon if possible.
49. Disable the Exception Settings option such as LoaderLock.
50. Run the app.

Code:-

Program.cs file

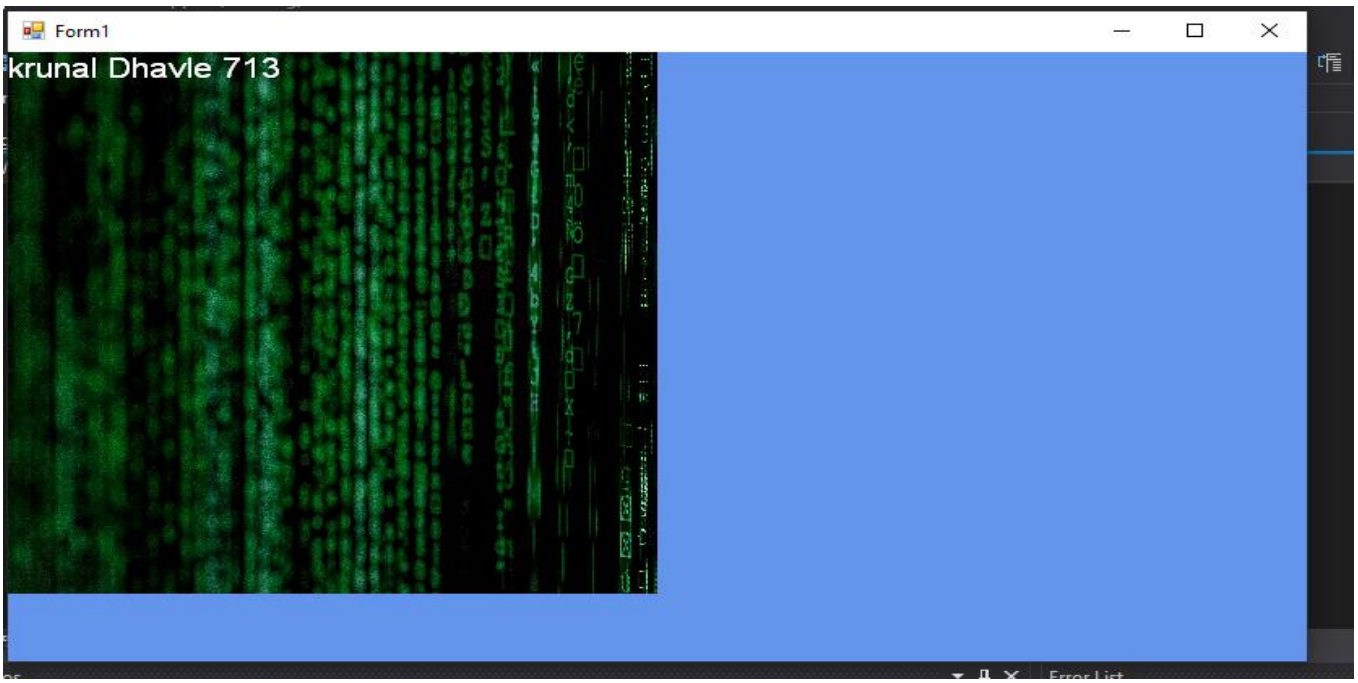
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using Microsoft.DirectX.Direct3D;

namespace WindowsFormsApp6
{
   static class Program
   {

      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
      }
   }
}
```

Form1.cs file

```
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 WindowsFormsApp12
{
   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();
         InitTexture();
      }
      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 InitTexture()
      {
         texture = TextureLoader.FromFile(device, "E:\\tycs\\gp prac\\prac6 vscode\\pic.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, 0, 0), new Point(0, 0), 0f,
new Point(0, 0), 1);

                s.Draw2D(texture, new Point(0, 0), 0.0f, new Point(0, 0), Color.White);

                font.DrawText(s, "krunal Dhavle 713", new Point(0, 0), Color.White);
                s.End();
            }
            device.EndScene();
            device.Present();
        }
        private void Form1_Load(object sender, EventArgs e)
        {

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

Output:



Date: 15/10/2020

Practical no 7

AIM: Loading models into DirectX 11 and rendering.

Steps:-

51. Create a new project and select a windows form application(.Net Framework 2.0-3.5).
52. Right click on the properties → click on open → click build → select platform target → x86 or add new
53. Click on view code on form 1(design) or press F7.
54. Go to the solution explorer → right click on project name → select add reference .
55. Click on browse and add the required dll files.
56. Code the required files.
57. Add the Load method for changing the appearance .
58. Change the window name and icon if possible.
59. Disable the Exception Settings option such as LoaderLock.
60. Add three file of airplane model in bin/Debug or bin/x86/Debug
61. Run the code.

Program Code:-

Program.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
namespace WindowsFormsApp17
{
   static class Program
   {
      [STAThread]
      static void Main()
      {
         Form1 app = new Form1();
         app.Width = 800;
         app.Height = 600;
         app.InitializeGraphics();
         app.Show();
         while (app.Created)
         {
            app.Render();
```

```
            Application.DoEvents();
        }
        app.DisposeGraphics();
    }
  }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;
namespace WindowsFormsApp17
{
  public partial class Form1 : Form
  {
    private Device device;
    private PresentParameters pres;
    private Mesh mesh;
    private Material[] materials;
    private Texture[] textures;

    public Form1()
    {
      InitializeComponent();
    }
```

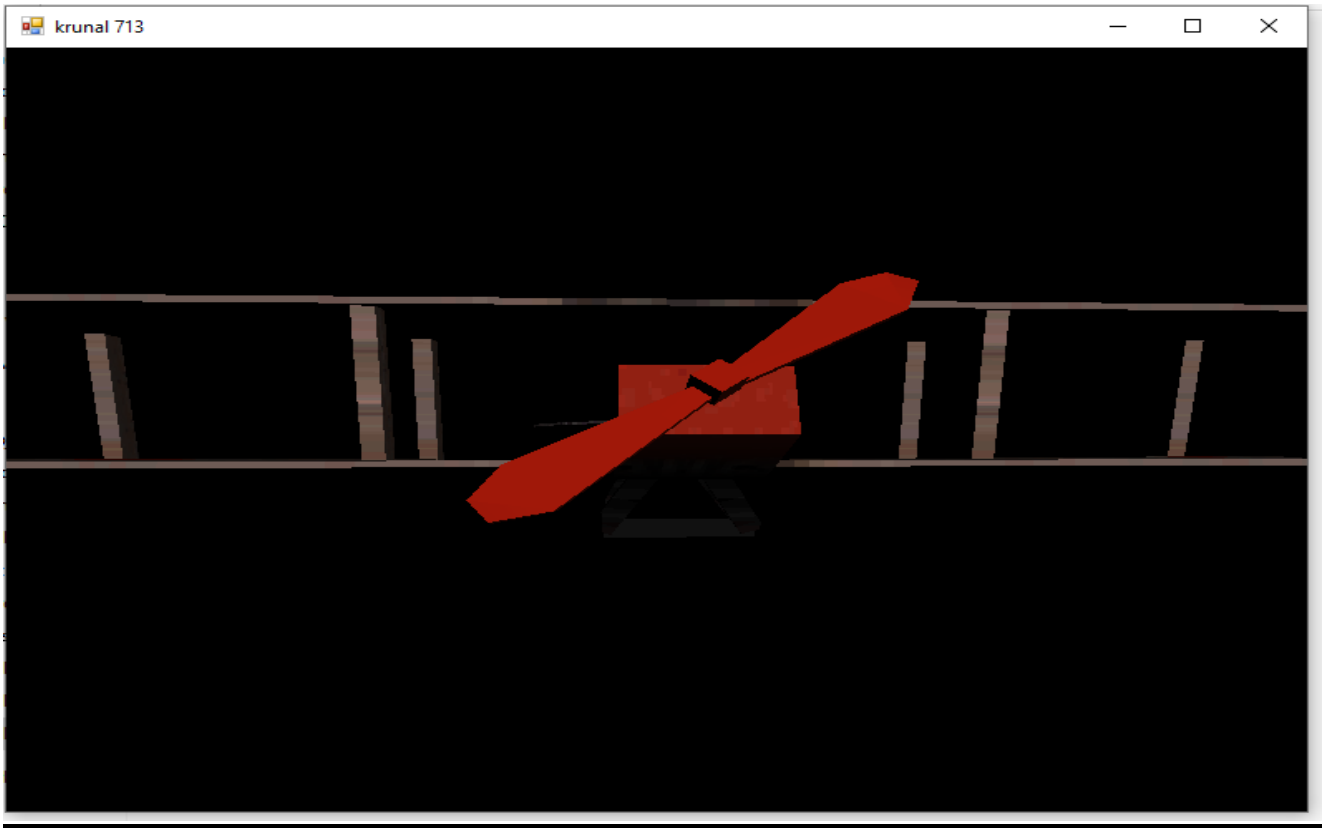
```
private void Form1_Load(object sender, EventArgs e)
{


}
public bool InitializeGraphics()
{
   pres = new PresentParameters();
   pres.Windowed = true;
   pres.SwapEffect = SwapEffect.Discard;
   pres.EnableAutoDepthStencil = true;
   pres.AutoDepthStencilFormat = DepthFormat.D16;
   device = new Device(0, DeviceType.Hardware, this,
  CreateFlags.SoftwareVertexProcessing,
   pres);
   device.RenderState.CullMode = Cull.None;
   CreateMesh(@"airplane 2.x");
   return true;
}
public void CreateMesh(string path)
{
   ExtendedMaterial[] exMaterials;
   mesh = Mesh.FromFile(path, MeshFlags.SystemMemory, device, out
  exMaterials);
   if (textures != null)

   {
      DisposeTextures();
   }
   textures = new Texture[exMaterials.Length];
   materials = new Material[exMaterials.Length];
```

```
        for (int i = 0; i < exMaterials.Length; ++i)
        {
            if (exMaterials[i].TextureFilename != null)
            {
                string texturePath = Path.Combine(Path.GetDirectoryName(path),
                exMaterials[i].TextureFilename);
                textures[i] = TextureLoader.FromFile(device, texturePath);
            }
            materials[i] = exMaterials[i].Material3D;
            materials[i].Ambient = materials[i].Diffuse;
        }
    }
    public void DisposeTextures()
    {
        if (textures == null)
        {
            return;
        }
        foreach (Texture t in textures)
        {
            if (t != null)
            {
                t.Dispose();
            }
        }
    }
    public void SetupMatrices()
    {
        float yaw = Environment.TickCount / 500.0F;
        float pitch = Environment.TickCount / 500.0F;
        device.Transform.World = Matrix.RotationYawPitchRoll(yaw, pitch, 0);
```

```
        device.Transform.View = Matrix.LookAtLH(new Vector3(0, 0, -6), new
       Vector3(0, 0, 0), new Vector3(0, 1, 0));
        device.Transform.Projection = Matrix.PerspectiveFovLH((float)Math.PI /
       2.0F, 1.0F, 1.0F, 10.0F);
    }
    public void SetupLights()
    {
        device.RenderState.Lighting = true;
        device.Lights[0].Diffuse = Color.White;
        device.Lights[0].Specular = Color.White;
        device.Lights[0].Type = LightType.Directional;
        device.Lights[0].Direction = new Vector3(-1, -1, 3);
        device.Lights[0].Enabled = true;
        device.RenderState.Ambient = Color.FromArgb(0x00, 0x00, 0x00);
    }
    public void RenderMesh()
    {
        for (int i = 0; i < materials.Length; ++i)
        {
            if (textures[i] != null)
            {

                device.SetTexture(0, textures[i]);
            }
            device.Material = materials[i];
            mesh.DrawSubset(i);
        }
    }
    public void Render()
    {
        device.Clear(ClearFlags.Target | ClearFlags.ZBuffer, Color.Black, 1.0F,
```

```
            0);
            device.BeginScene();
            SetupMatrices();
            SetupLights();
            RenderMesh();
            device.EndScene();
            device.Present();
        }
        public void DisposeGraphics()
        {
            DisposeTextures();
            device.Dispose();
        }
    }
}
```

Output



Date:29/10/2020

Practical No 8

AIM : Develop a 3D-Space-Shooter Game in Unity Game Engine
https://unity3d.com/learn/tutorials/s/space-shooter-tutorial

CODE:

1. Player-Controller.cs File

```
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 float firerate = 0.5f;
         public GameObject shot;
        public Transform shotspawn;
        private float mytime = 0.0f;
        private float nextfire = 0.5f;
        private Rigidbody rb;
        public Boundary boundary;
         private Touch theTouch;
        private Vector2 touchStartPosition, touchEndPosition;
        private string direction;
         private AudioSource audio;
 void Start()
 {
        rb = GetComponent<Rigidbody>();
```

```
        audio = GetComponent<AudioSource>();
 }
 void Update()
 {
  mytime = mytime + Time.deltaTime;
 Gametouchfire();
}
void Gametouchfire()
 {
 if (Input.touchCount > 0)
 {
        theTouch = Input.GetTouch(0);

        if (theTouch.phase == TouchPhase.Began)
         {
         touchStartPosition = theTouch.position;
        }

         else if (theTouch.phase == TouchPhase.Moved || theTouch.phase== TouchPhase.Ended)
        {
        touchEndPosition = theTouch.position;

         float x = touchEndPosition.x - touchStartPosition.x;
         float y = touchEndPosition.y - touchStartPosition.y;

         if ((Mathf.Abs(x) == 0 && Mathf.Abs(y) == 0) && mytime >nextfire)
        {
        nextfire = mytime + firerate;
         Instantiate(shot, shotspawn.position,
        shotspawn.rotation);
        audio.Play();
```

```
                    nextfire = nextfire - mytime;
                    mytime = 0.0f;
                }
            }
        }
    }
 void FixedUpdate()
 {
        //float moveHorizontal = Input.GetAxis ("Horizontal");
        //float moveVertical = Input.GetAxis ("Vertical");
        float moveHorizontal = Input.acceleration.x;
        float moveVertical = 0.0f;
        Vector3 movement = new Vector3(moveHorizontal, 0.0f,moveVertical);
         rb.velocity = movement * speed;
         rb.position = new Vector3
         (
         Mathf.Clamp(rb.position.x, boundary.xMin, boundary.xMax),
         0.0f,
         Mathf.Clamp(rb.position.z, boundary.zMin, boundary.zMax)
         );
        rb.rotation = Quaternion.Euler(0.0f, 0.0f, rb.velocity.x *-tilt);
         }
}
```

2. Destroy-By-Boundary.cs file

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

```
 void OnTriggerExit(Collider other)
 {
        Destroy(other.gameObject);
 }
}
```

3. Destroy-By-Contact.cs file

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

public class DestroyByContact : MonoBehaviour {

        public GameObject Explosion;
        public GameObject PExplosion;
        private GameController gamecontroller;
        public int scoreval;
        void Start(){
                GameObject gc = GameObject.FindWithTag("GameController");
                if (gc != null) {
                gamecontroller = gc.GetComponent<GameController> ();
                 }
                if (gamecontroller == null) {
                Debug.Log ("Game Controller not found");
                 }
        }
        void OnTriggerEnter(Collider other)
        {
        if (other.tag == "Boundary")
        {
```

```
        return;
        }
        Instantiate (Explosion,transform.position,transform.rotation);
        if (other.tag == "Player") {
        Instantiate (PExplosion,other.transform.position,other.transform.rotation);
        gamecontroller.GameOvers ();
         }
        gamecontroller.addScore (scoreval);
        Destroy(other.gameObject);
        Destroy(gameObject);
}
}
```

4. Destroy-By-Time.cs file

```
using UnityEngine;
using System.Collections;

public class DestroyByTime : MonoBehaviour
{
        public float lifetime;
        void Start(){
                Destroy (gameObject, lifetime);
        }
}
```

5. Mover.cs file

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

public class Mover : MonoBehaviour {
        public float speed;
```

```
        private Rigidbody rb;
        void Start () {
                rb = GetComponent<Rigidbody> ();
                rb.velocity = transform.forward * speed;
        }
}
```

6. Game-Controller.cs file

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class GameController : MonoBehaviour {
        public GameObject hazard;
        public Vector3 spawnvalue;
        public int hazardcount;
        public float spawnwait;
        public float startwait;
        public float wavewait;
        public Text score;
        public Text restart;
         public Text GameOver;
        private bool go;
        private bool rs;
        private int scr;
        public Button rst;
        private Touch theTouch;
        private Vector2 touchStartPosition, touchEndPosition;
        private string direction;
```

```
        void Start(){
                go = false;
                rs = false;
        restart.text = "";
        GameOver.text = "";
        scr = 0;
        UpdateScore ();
        StartCoroutine(SpawnWaves ());
        }

        IEnumerator SpawnWaves(){
                yield return new WaitForSeconds (startwait);
                while(!go){
                 for (int i=0;i<hazardcount;i++) {
                Vector3 spawnPosition = new Vector3 (Random.Range(-spawnvalue.x,
spawnvalue.x), spawnvalue.y,spawnvalue.z);
        Quaternion srotation = Quaternion.identity;
        Instantiate (hazard, spawnPosition, srotation);
        yield return new WaitForSeconds (spawnwait);
        }
        yield return new WaitForSeconds (wavewait);
        if (go) {
        restart.text = "Restart";
        rs = true;
        break;
        }
        }
}
        void Update(){
        if (rs) {
        if (Input.touchCount > 0){
```

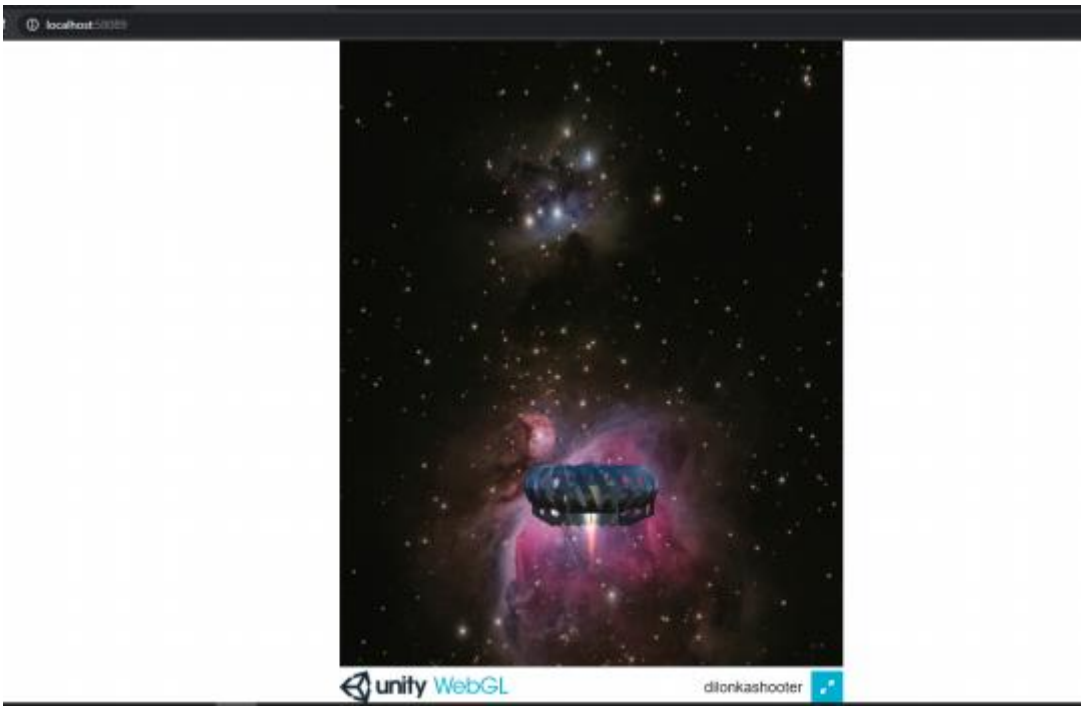
```
        theTouch = Input.GetTouch(0);
                if (theTouch.phase == TouchPhase.Began){
                touchStartPosition = theTouch.position;
                }
                else if (theTouch.phase == TouchPhase.Moved || theTouch.phase==
TouchPhase.Ended){
                touchEndPosition = theTouch.position;
                float x = touchEndPosition.x - touchStartPosition.x;
        float y = touchEndPosition.y - touchStartPosition.y;
        if (Mathf.Abs(x) == 0 && Mathf.Abs(y) == 0){
        Application.LoadLevel(Application.loadedLevel);
        }
        }
        }
  }
        public void addScore(int newscr){
        scr += newscr;
        UpdateScore ();
        }
         void UpdateScore(){
        score.text = "Score : " + scr.ToString ();
        }
        public void GameOvers(){
        GameOver.text = "Game Over";
        go = true;
         }
}
```

7. Random-Rotator.cs file

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

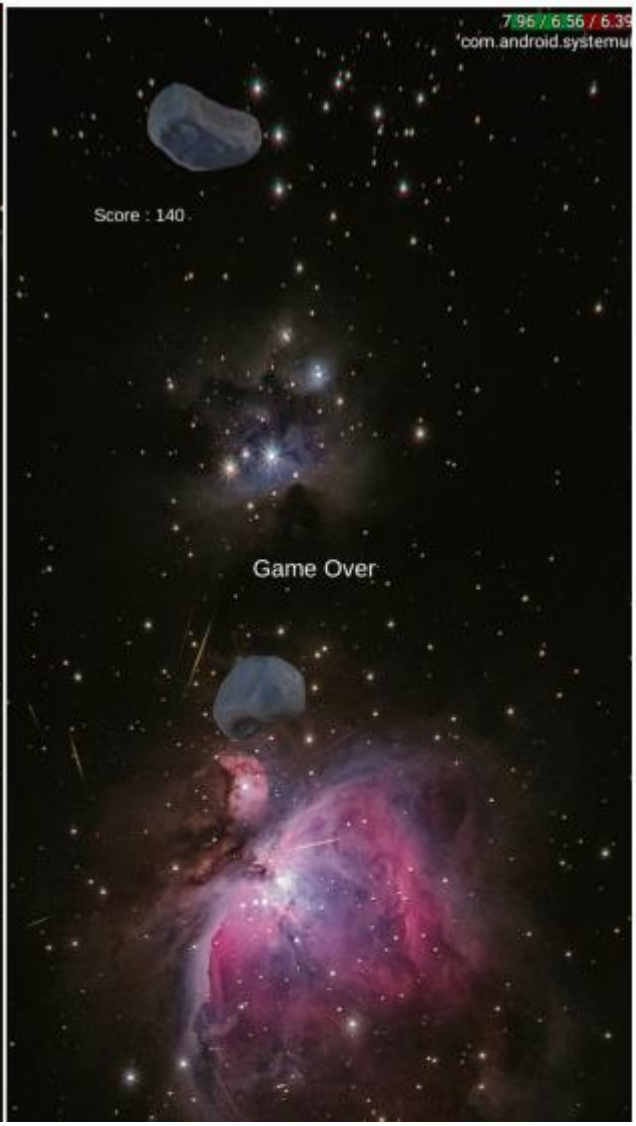
```
using UnityEngine;


public class RandomRotator : MonoBehaviour {
        public float tumble;
        private Rigidbody rb;
        void Start ()
        {
                rb = GetComponent<Rigidbody> ();
                rb.angularVelocity = Random.insideUnitSphere * tumble;
        }
 }
```

Output




Score : 10
Restart
Game Over


6.44 / 6.81 / 6.91
com.android.systemui
Score : 0


com.android.systemui
Score : 140
Game Over

Date: 12/112020

Practical No 9

Aim : Develop a 3D Roll a Ball Game in Unity Game Engine https://unity3d.com/learn/tutorials/s/roll-ball-tutorial

Code

1. **Player-Controller.cs file**

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

public class controller : MonoBehaviour {
        public float speed;
        private Rigidbody rb;
        public Text counttext;
        private int count;
        public Text wint;

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

        void FixedUpdate () {
        //For Phone Motion Sensor Controls
        float moveHorizontal=Input.acceleration.x;
         float moveVertical = Input.acceleration.y;
        // For PC Keyboard controls
        //float moveHorizontal=Input.GetAxis("Horizontal");
        //float moveVertical=Input.GetAxis("Vertical");
        Vector3 Movement = new Vector3
        (moveHorizontal,0.0f,moveVertical);
        rb.AddForce (Movement*speed);
         }
        void OnTriggerEnter(Collider other)
        {
        if (other.gameObject.CompareTag ("pickup"))
        {
        other.gameObject.SetActive (false);
        count += 1;
        settext ();
        }
         }
         void settext(){
         counttext.text = "Count :" + count.ToString();
```

```
        if (count >= 20)
        wint.text = "You Win";
         }
}
```

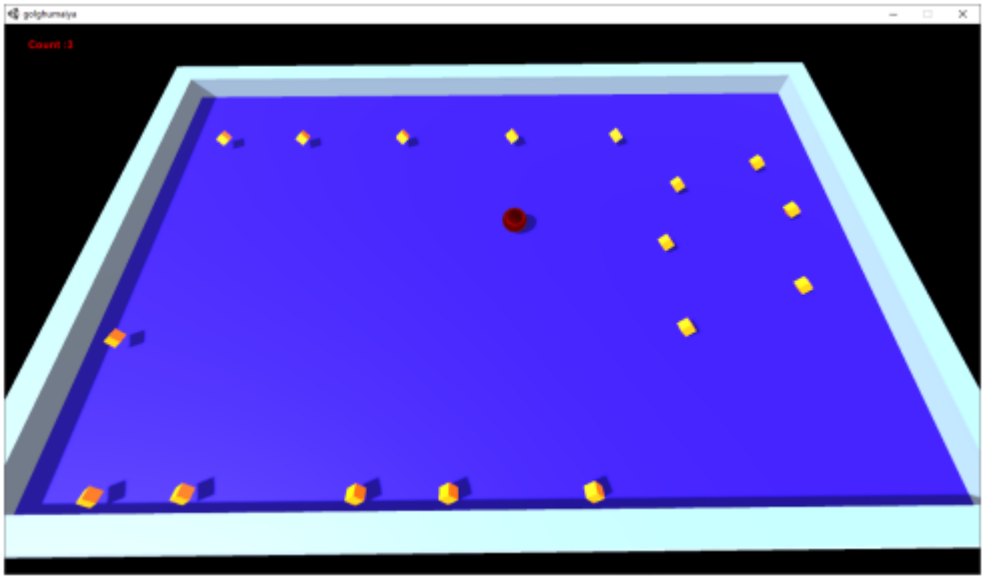
2. Camera-Controller.cs file

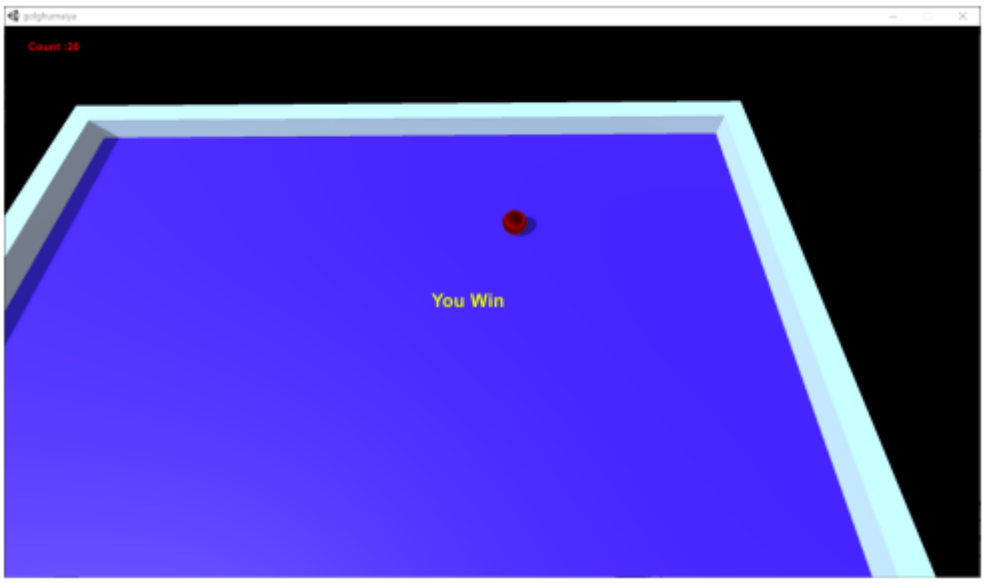
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class cameracontroller : MonoBehaviour {
                public GameObject player;
                private Vector3 offset;
                void Start () {
                offset = transform.position - player.transform.position;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

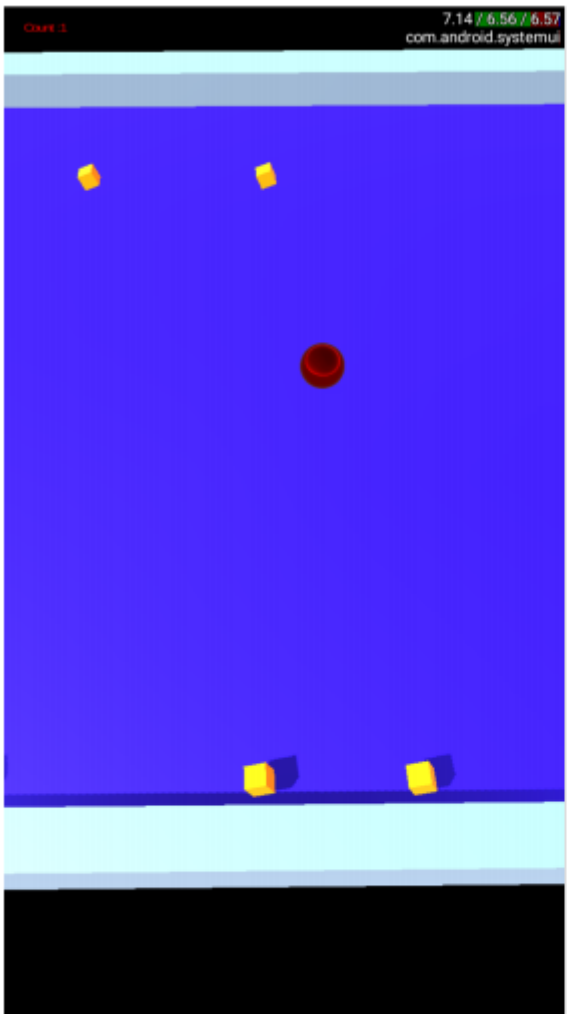
3. Rotator.cs file

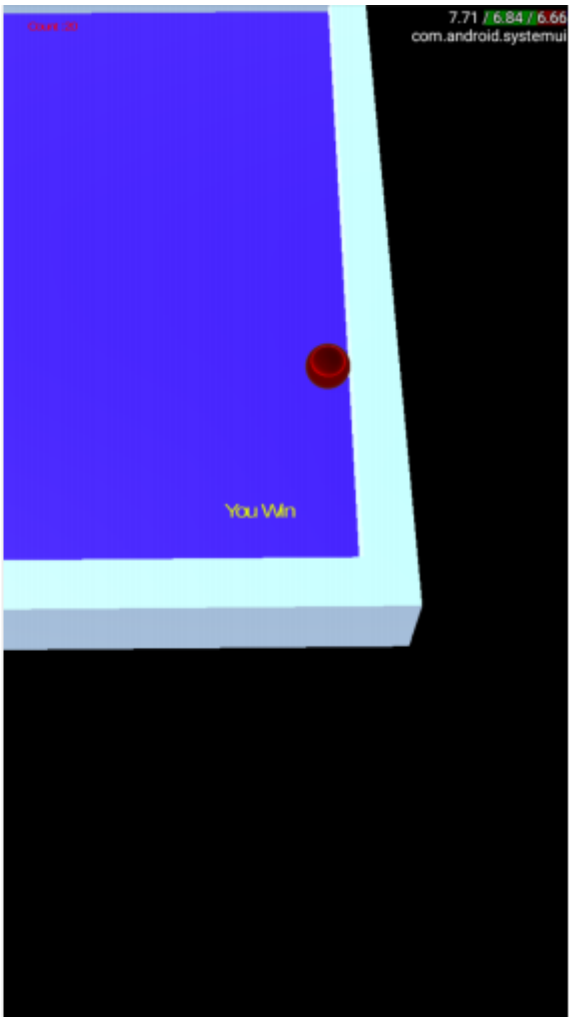
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

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

Output




You Win


7.14 / 6.56 / 6.57
com.android.systemui


7.71 / 6.84 / 6.66
com.android.systemui
You Win