Date: 15/10/2020

Practical no 7

AIM: Loading models into DirectX 11 and rendering.

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 Load method for changing the appearance .
8. Change the window name and icon if possible.
9. Disable the Exception Settings option such as LoaderLock.
10. Add three file of airplane model in bin/Debug or bin/x86/Debug
11. 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