**Practical 1**

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

**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;

namespace practical1

{

public partial class Form1 : Form

{

Microsoft.DirectX.Direct3D.Device devices;

public Form1()

{

InitializeComponent();

InitDevice();

}

private void InitDevice()

{

PresentParameters pp = new PresentParameters();

pp.Windowed = true;

pp.SwapEffect = SwapEffect.Discard;

devices = new Device(0, DeviceType.Hardware, this,CreateFlags.HardwareVertexProcessing, pp);

}

private void Render() {

devices.Clear(ClearFlags.Target, Color.CornflowerBlue, 0, 1);

devices.Present();

}

private void Form1\_Load(object sender, EventArgs e)

{

private void Form1\_Paint(object sender, PaintEventAr

{

Render();

}

}

}

**Practical 2**

**Aim:** Buffers, shaders and HLSL (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 prac2

{

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.SoftwareVertexProcessing, pp);

}

private void Render()

{

CustomVertex.TransformedColored[] v = new CustomVertex.TransformedColored[3];

v[0].Position = new Vector4(100, 100, 0, 1.4f);

v[1].Position = new Vector4(150, 300, 0, 1.0f);

v[2].Position = new Vector4(80, 300, 0, 0);

v[0].Color = System.Drawing.Color.ForestGreen.ToArgb();

v[1].Color = System.Drawing.Color.FromArgb(255, 0, 0).ToArgb();

device.Clear(ClearFlags.Target, Color.Blue, 0, 1);

device.BeginScene();

device.VertexFormat = CustomVertex.TransformedColored.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, 1, v);

device.EndScene();

device.Present();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

Render();

}

}

}

**Practical 3**

**Aim:** Texturing (Texture the Triangle using Direct 3D 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 prac3

{

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.SoftwareVertexProcessing, pp);

}

public void Render()

{

CustomVertex.PositionTextured[] v = new CustomVertex.PositionTextured[3];

Texture t;

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;

v[0] = new CustomVertex.PositionTextured(new Vector3(0, 4, 4), 0, 0);

v[1] = new CustomVertex.PositionTextured(new Vector3(-1, -4, 4), -1, 0);

v[2] = new CustomVertex.PositionTextured(new Vector3(1, -4, 4), 0, -1);

t = new Texture(device, new Bitmap("C:\\Users\\Pradeep\\Pictures\\IMG\_20220729\_193656\_517.png"), 0, Pool.Managed);

device.Clear(ClearFlags.Target, Color.Black, 1, 0);

device.BeginScene();

device.SetTexture(0, t);

device.VertexFormat = CustomVertex.PositionTextured.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, v.Length / 3, v);

device.EndScene();

device.Present();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

Render();

}

}

}

**Practical 4**

**Aim:** Lightning (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 prac4

{

public partial class Form1 : Form

{

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.SoftwareVertexProcessing, pp);

}

public void Render()

{

CustomVertex.PositionNormalColored[] v = new CustomVertex.PositionNormalColored[3];

device.Transform.Projection = Matrix.PerspectiveFovLH(3.14f / 4, device.Viewport.Width / device.Viewport.Height, 1f, 1000f);

device.Transform.View = Matrix.LookAtLH(new Vector3(2, 0, 10), new Vector3(), new Vector3(9, 50, 50));

device.RenderState.Lighting = false;

v[0] = new CustomVertex.PositionNormalColored(new Vector3(0, 1, 1), new Vector3(1, 0, 1), Color.Red.ToArgb());

v[1] = new CustomVertex.PositionNormalColored(new Vector3(-1, -1, 1), new Vector3(1, 0, 1), Color.Red.ToArgb());

v[2] = new CustomVertex.PositionNormalColored(new Vector3(0, -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.Red;

device.Lights[0].Direction = new Vector3(0.8f, 0, -1);

device.Lights[0].Enabled = true;

device.Clear(ClearFlags.Target, Color.RoyalBlue, 1, 0);

device.BeginScene();

device.VertexFormat = CustomVertex.PositionNormalColored.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, v.Length / 3, v);

device.EndScene();

device.Present();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

Render();

}

}

}

**Practical 5**

**Aim:** Specular Lightning (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;

using Microsoft.DirectX.Direct3D;

namespace Prac5

{

public partial class Form1 : Form

{

Microsoft.DirectX.Direct3D.Device device;

private CustomVertex.PositionNormalColored[] v = 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;

v[0] = new CustomVertex.PositionNormalColored(new Vector3(0, 1, 1), new Vector3(1, 0, 1), Color.Red.ToArgb());

v[1] = new CustomVertex.PositionNormalColored(new Vector3(-1, -1, 1), new Vector3(1, 0, 1), Color.Blue.ToArgb());

v[2] = new CustomVertex.PositionNormalColored(new Vector3(1, -1, 1), new Vector3(-1, 0, 1), Color.Green.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;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

device.Clear(ClearFlags.Target, Color.BlueViolet, 1, 0);

device.BeginScene();

device.VertexFormat = CustomVertex.PositionNormalColored.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, v.Length / 3, v);

device.EndScene();

device.Present();

}

}

}

**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.Direct3D;

using Microsoft.DirectX;

namespace prac6

{

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, "C:\\Users\\Pradeep\\Pictures\\salman.png",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);

}

public 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, "Salman Khan", 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();

}

}

}