|  |  |  |
| --- | --- | --- |
| **K.M.S.P. Mandal’s**  **Sant Rawool Maharaj Mahavidyalay, Kudal**  **Department Of Computer Science** | Date :-  Roll No :-05  Expt No :-2 | **Signature** |
| **Title** :- Buffers, Shaders and HLSL (Draw a triangle using Direct3D 11) | | |

**To draw a triangle used:**

**A. Vertices:**

Mathematically, the vertices of a triangle are where two edges meet; the vertices of a line are the endpoints; for a single point, the point itself is the vertex.r, in Direct3D, vertices are much more general than that. Essentially, a vertex in Direct3D can consist of additional data besides spatial location, which allows us to perform more sophisticated rendering effects.Direct3D gives us the flexibility to define our own vertex formats (i.e., it allows us to define the components of a vertex).

**B. PrimitiveType:**

PrimitiveType is nothing but Primitive Topology.Vertices are bound to the rendering pipeline in a special Direct3D data structure called a vertex buffer. A vertex buffer just stores a list of vertices in contiguous memory. However, it does not say how these vertices should be put together to form geometric primitives. For example, should every two vertices in the vertex buffer be interpreted as a line or should every three vertices in the vertex buffer be interpreted as a triangle? We tell Direct3D how to form geometric primitives from the vertex data by specifying the primitive topology.

**1. Point List:**

A point list is specified by D3D11\_PRIMITIVE\_TOPOLOGY\_POINTLIST. With a point list, every vertex in the draw call is drawn as an individual point.

**2. Line Strip:**

A line strip is specified by D3D11\_PRIMITIVE\_TOPOLOGY\_LINESTRIP. With a line strip, the vertices in the draw call are connected to form lines so n + 1 vertices induces n lines.

**3. Line List:**

A line list is specified by D3D11\_PRIMITIVE\_TOPOLOGY\_LINELIST. With a line list, every two vertices in the draw call forms an individual line (see Figure 5.13c); so 2n vertices induces n lines.

**4. Triangle Strip:**

A triangle strip is specified by D3D11\_PRIMITIVE\_TOPOLOGY\_TRIANGLESTRIP. With a triangle strip, it is assumed the triangles are connected as shown in Figure 5.13d to form a strip.

**5. Triangle List:**

A triangle list is specified by D3D11\_PRIMITIVE\_TOPOLOGY\_TRIANGLELIST. With a triangle list, every three vertices in the draw call forms an individual triangle so 3n vertices induces n triangles.

**C. Indices:**

We create a vertex list and an index list. The vertex list consists of all the unique vertices and the index list contains values that index into the vertex list to define how the vertices are to be put together to form triangles.

**Program:-**

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 newTriangle

{

public partial class Form1 : Form

{

Device d;

CustomVertex.TransformedColored[] v = new CustomVertex.TransformedColored[4];

public Form1()

{

InitializeComponent();

initGraphics();

}

public void initGraphics()

{

PresentParameters p = new PresentParameters();

p.Windowed = true;

p.SwapEffect = SwapEffect.Discard;

d = new Device(0, DeviceType.Hardware, this, CreateFlags.HardwareVertexProcessing, p);

d.Present();

}

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

{

}

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

{

v[0].Position = new Vector4(20,50, 0, 1.0f);

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

v[1].Position = new Vector4(140, 100, 0, 1.0f);

v[1].Color = System.Drawing.Color.Red.ToArgb();

v[2].Position = new Vector4(50, 100, 0, 1.0f);

v[2].Color = System.Drawing.Color.Blue.ToArgb();

v[3].Position = new Vector4(100,50, 0, 1.0f);

v[3].Color = System.Drawing.Color.Blue.ToArgb();

d.BeginScene();

d.VertexFormat = CustomVertex.TransformedColored.Format;

// d.DrawUserPrimitives(PrimitiveType.TriangleList, 1, v);

d.DrawUserPrimitives(PrimitiveType.TriangleStrip, 2, v);

d.EndScene();

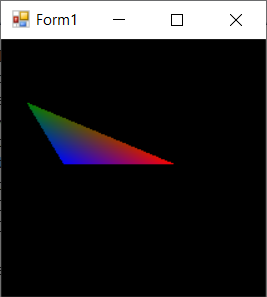
d.Present();

}

}

}

**Output:-**

****