**6B**

**Code:**

using System;

usingSystem.Collections.Generic;

usingSystem.ComponentModel;

usingSystem.Data;

usingSystem.Drawing;

usingSystem.Text;

usingSystem.Windows.Forms;

using System.IO;

usingMicrosoft.DirectX.Direct3D;

usingMicrosoft.DirectX;

namespace GP\_E

{

publicpartialclassForm1 : Form

{

private Device device;

privatePresentParameterspres;

private Mesh mesh;

privateMaterial[] materials;

privateTexture[] textures;

public Form1()

{

InitializeComponent();

}

publicboolInitializeGraphics()

{

pres = newPresentParameters();

pres.Windowed = true;

pres.SwapEffect = SwapEffect.Discard;

pres.EnableAutoDepthStencil = true;

pres.AutoDepthStencilFormat = DepthFormat.D16;

device = newDevice(0, DeviceType.Hardware, this, CreateFlags.SoftwareVertexProcessing,

pres);

device.RenderState.CullMode = Cull.None;

CreateMesh(@"airplane 2.x");

returntrue;

}

publicvoidCreateMesh(string path)

{

ExtendedMaterial[] exMaterials;

mesh = Mesh.FromFile(path, MeshFlags.SystemMemory, device, outexMaterials);

if (textures != null)

{

DisposeTextures();

}

textures = newTexture[exMaterials.Length];

materials = newMaterial[exMaterials.Length];

for (inti = 0; i<exMaterials.Length; ++i)

{

if (exMaterials[i].TextureFilename != null)

{

stringtexturePath = 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;

}

}

publicvoidDisposeTextures()

{

if (textures == null)

{

return;

}

foreach (Texture t in textures)

{

if (t != null)

{

t.Dispose();

}

}

}

publicvoidSetupMatrices()

{

float yaw = Environment.TickCount / 500.0F;

float pitch = Environment.TickCount / 500.0F;

float roll = Environment.TickCount / 500.0F;

device.Transform.World = Matrix.RotationYawPitchRoll(yaw, pitch, roll);

device.Transform.View = Matrix.LookAtLH(new Vector3(0, 0, -8), new Vector3(0, 0, 0), new

Vector3(0, 1, 0));

device.Transform.Projection = Matrix.PerspectiveFovLH((float)Math.PI / 2.0F, 1.0F, 1.0F, 12.0F);

}

publicvoidSetupLights()

{

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);

}

publicvoidRenderMesh()

{

for (inti = 0; i<materials.Length; ++i)

{

if (textures[i] != null)

{

device.SetTexture(0, textures[i]);

}

device.Material = materials[i];

mesh.DrawSubset(i);

}

}

publicvoidRender()

{

device.Clear(ClearFlags.Target | ClearFlags.ZBuffer, Color.SkyBlue, 1.0F, 0);

device.BeginScene();

SetupMatrices();

SetupLights();

RenderMesh();

device.EndScene();

device.Present();

}

publicvoidDisposeGraphics()

{

DisposeTextures();

device.Dispose();

}

privatevoid Form1\_Paint(object sender, PaintEventArgs e)

{

Render();

}

}

}

**Output:**

