**武汉纺织大学**

**《Direct3D图形编程》上机实验报告**

**题目:** **网格（二）**

**成 绩：**

**学 号： 1604240705**

**姓 名： 胡 莲**

**班 级： 计算机11603**

**指导教师： 李 敏**

**报告日期：2019 年5月20 日**

1. **实验1**
2. 题目

使用网格，在线框绘图模式下绘制飞行器，飞行器绕Y轴连续旋转，并回答飞行器由有多少顶点，多少三角形，多少个子集？

1. 实现代码

在d3dInit框架的基础上添加代码：

1. 全局变量

const int Width = 640;

const int Height = 480;

IDirect3DDevice9\* Device = 0;

ID3DXMesh\* mesh = 0;

std::vector<D3DMATERIAL9> Mtrls(0);

std::vector<IDirect3DTexture9\*> Tex(0);

1. Setup()函数

HRESULT hr = 0;

ID3DXBuffer\* adjBuffer = 0;

ID3DXBuffer\* mtrlBuffer = 0;

DWORD numMtrls = 0;

hr = D3DXLoadMeshFromX(

"bigship1.x",

D3DXMESH\_MANAGED,

Device,

&adjBuffer,

&mtrlBuffer,

0,

&numMtrls,

&mesh);

if(FAILED(hr))

{

::MessageBox(0,"D3DXLoadMeshFromX() - FAILED",0,0);

return false;

}

if(mtrlBuffer != 0 && numMtrls != 0)

{

D3DXMATERIAL\* mtrls = (D3DXMATERIAL\*)mtrlBuffer->GetBufferPointer();

for (int i = 0; i < numMtrls; i++)

{

D3DMATERIAL9 Mtrl = d3d::BLACK\_MTRL;

//mtrls[i].MatD3D.Ambient = mtrls[i].MatD3D.Diffuse;

//Mtrls.push\_back (mtrls[i].MatD3D);

Mtrls.push\_back (Mtrl);

if(mtrls[i].pTextureFilename != 0)

{

IDirect3DTexture9\* tex = 0;

D3DXCreateTextureFromFile(

Device,

mtrls[i].pTextureFilename,

&tex);

Tex.push\_back(tex);

}

else

{

Tex.push\_back(0);

}

}

}

d3d::Release<ID3DXBuffer\*>(mtrlBuffer);

DWORD\* attributeBuffer = 0;

mesh->LockAttributeBuffer(0, &attributeBuffer);

for(int a = 0; a < 4; a++)

attributeBuffer[a] = 0;

for(int b = 4; b < 8; b++)

attributeBuffer[b] = 1;

for(int c = 8; c < 12; c++)

attributeBuffer[c] = 2;

mesh->UnlockAttributeBuffer();

std::vector<DWORD> adjacencyBuffer(mesh->GetNumFaces() \* 3);

mesh->GenerateAdjacency(0.0f, &adjacencyBuffer[0]);

hr = mesh->OptimizeInplace(

D3DXMESHOPT\_ATTRSORT |

D3DXMESHOPT\_COMPACT |

D3DXMESHOPT\_VERTEXCACHE,

&adjacencyBuffer[0],

0, 0, 0);

d3d::Release<ID3DXBuffer\*>(adjBuffer);

DWORD numEntries = 0;

mesh ->GetAttributeTable(0,&numEntries);

std::vector<D3DXATTRIBUTERANGE> table(numEntries);

mesh->GetAttributeTable(&table[0],&numEntries);

&table[0].AttribId;

D3DXVECTOR3 dir(1.0f,0.0f,0.0f);

D3DXCOLOR c = d3d::WHITE;

D3DLIGHT9 light = d3d::InitDirectionalLight(&dir, &c);

Device->SetLight(0,&light);

Device->LightEnable(0,true);

Device->SetRenderState(D3DRS\_NORMALIZENORMALS,true);

D3DXVECTOR3 position(0.0f, 2.0f, -20.0f);

D3DXVECTOR3 target(0.0f, 0.0f, 0.0f);

D3DXVECTOR3 up(0.0f, 1.0f, 0.0f);

D3DXMATRIX V;

D3DXMatrixLookAtLH(&V, &position, &target, &up);

Device -> SetTransform(D3DTS\_VIEW, &V);

D3DXMATRIX proj;

D3DXMatrixPerspectiveFovLH(

&proj,

D3DX\_PI \* 0.5f,

(float)Width / (float)Height,

1.0f,

1000.0f);

Device -> SetTransform(D3DTS\_PROJECTION, &proj);

Device->SetRenderState(D3DRS\_FILLMODE,D3DFILL\_WIREFRAME);

Device -> SetRenderState(D3DRS\_LIGHTING,true);

1. Cleanup()函数

d3d::Release<ID3DXMesh\*>(mesh);

for (int i = 0; i < Tex.size(); i++)

d3d::Release<IDirect3DTexture9\*>(Tex[i]);

1. Display()函数

Device->BeginScene();

D3DXMATRIX Ry;

static float y = 0.0f;

D3DXMatrixRotationY(&Ry,y);

y += timeDelta;

Device->SetTransform(D3DTS\_WORLD,&Ry);

for(int i = 0; i < Mtrls.size(); i++)

{

Device->SetMaterial(&Mtrls[i]);

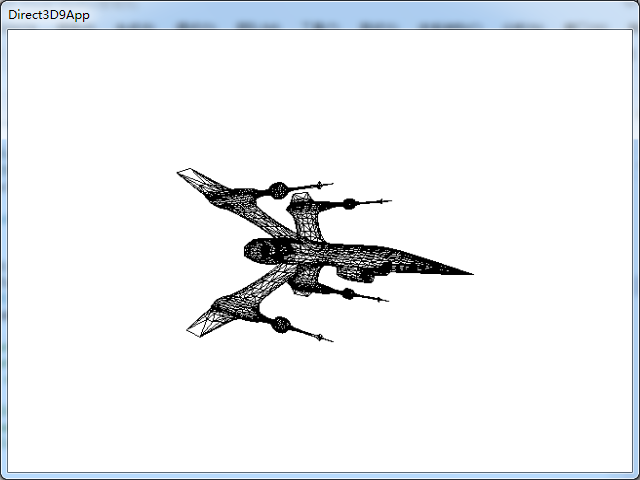
Device->SetTexture(0,Tex[i]);

mesh->DrawSubset( i );

}

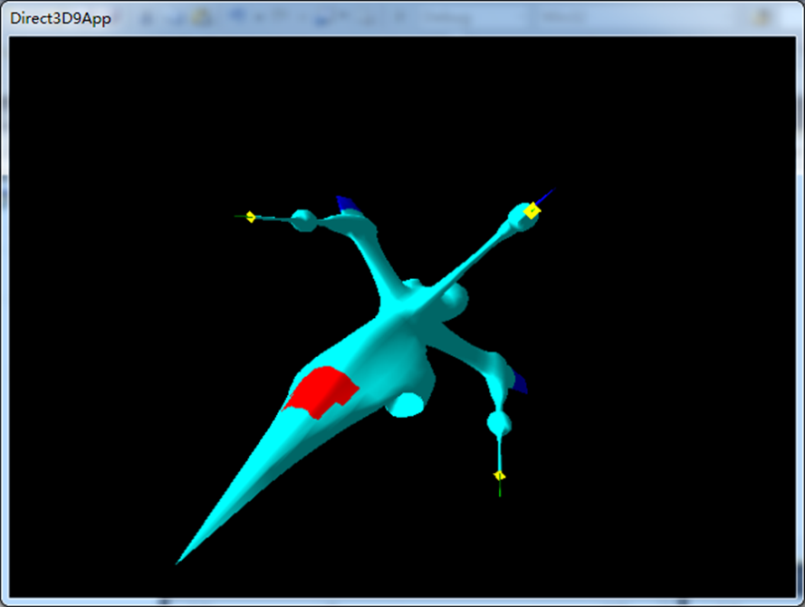
Device->EndScene();

1. 程序运行结果



1. **总结**
2. 飞行器由4415个顶点、8438个三角面、总共5个子集构成。
3. 要在Setup()函数里设置线框绘图模式Device->SetRenderState(D3DRS\_FILLMODE,D3DFILL\_WIREFRAME);
4. 按理说，设置为线框绘图模式后，模型出来应该就是默认的黑色，但是，最后还是有一些别的颜色，故在材质那里，我将所有材质颜色都设为了黑色。
5. **实验2**
6. 题目

使用网格绘制飞行器，飞行器绕Y轴连续旋转，效果如下：



1. 实现代码

在d3dInit框架的基础上添加代码：

1. 全局变量

#include <vector>

const int Width = 640;

const int Height = 480;

IDirect3DDevice9\* Device = 0;

ID3DXMesh\* mesh = 0;

std::vector<D3DMATERIAL9> Mtrls(0);

std::vector<IDirect3DTexture9\*> Tex(0);

1. Setup()函数

HRESULT hr = 0;

ID3DXBuffer\* adjBuffer = 0;

ID3DXBuffer\* mtrlBuffer = 0;

DWORD numMtrls = 0;

hr = D3DXLoadMeshFromX(

"bigship1.x",

D3DXMESH\_MANAGED,

Device,

&adjBuffer,

&mtrlBuffer,

0,

&numMtrls,

&mesh);

if(FAILED(hr))

{

::MessageBox(0,"D3DXLoadMeshFromX() - FAILED",0,0);

return false;

}

if(mtrlBuffer != 0 && numMtrls != 0)

{

D3DXMATERIAL\* mtrls = (D3DXMATERIAL\*)mtrlBuffer->GetBufferPointer();

std::vector<D3DMATERIAL9> mtrls2(0);

D3DMATERIAL9 cyanMtrl = d3d::CYAN\_MTRL;

D3DMATERIAL9 blueMtrl = d3d::BLUE\_MTRL;

D3DMATERIAL9 redMtrl = d3d::RED\_MTRL;

D3DMATERIAL9 greenMtrl = d3d::GREEN\_MTRL;

D3DMATERIAL9 yellowMtrl = d3d::YELLOW\_MTRL;

mtrls2.push\_back(cyanMtrl);

mtrls2.push\_back(blueMtrl);

mtrls2.push\_back(redMtrl);

mtrls2.push\_back(greenMtrl);

mtrls2.push\_back(yellowMtrl);

for (int i = 0; i < numMtrls; i++)

{

//mtrls[i].MatD3D.Ambient = mtrls[i].MatD3D.Diffuse;

//Mtrls.push\_back (mtrls[i].MatD3D);

Mtrls=mtrls2;

if(mtrls[i].pTextureFilename != 0)

{

IDirect3DTexture9\* tex = 0;

D3DXCreateTextureFromFile(

Device,

mtrls[i].pTextureFilename,

&tex);

Tex.push\_back(tex);

}

else

{

Tex.push\_back(0);

}

}

}

d3d::Release<ID3DXBuffer\*>(mtrlBuffer);

DWORD\* attributeBuffer = 0;

mesh->LockAttributeBuffer(0, &attributeBuffer);

for(int a = 0; a < 4; a++)

attributeBuffer[a] = 0;

for(int b = 4; b < 8; b++)

attributeBuffer[b] = 1;

for(int c = 8; c < 12; c++)

attributeBuffer[c] = 2;

mesh->UnlockAttributeBuffer();

std::vector<DWORD> adjacencyBuffer(mesh->GetNumFaces() \* 3);

mesh->GenerateAdjacency(0.0f, &adjacencyBuffer[0]);

hr = mesh->OptimizeInplace(

D3DXMESHOPT\_ATTRSORT |

D3DXMESHOPT\_COMPACT |

D3DXMESHOPT\_VERTEXCACHE,

&adjacencyBuffer[0],

0, 0, 0);

d3d::Release<ID3DXBuffer\*>(adjBuffer);

D3DXVECTOR3 dir(1.0f,0.0f,0.0f);

D3DXCOLOR c = d3d::WHITE;

D3DLIGHT9 light = d3d::InitDirectionalLight(&dir, &c);

Device->SetLight(0,&light);

Device->LightEnable(0,true);

Device->SetRenderState(D3DRS\_NORMALIZENORMALS,true);

D3DXVECTOR3 position(0.0f, 2.0f, -20.0f);

D3DXVECTOR3 target(0.0f, 0.0f, 0.0f);

D3DXVECTOR3 up(0.0f, 1.0f, 0.0f);

D3DXMATRIX V;

D3DXMatrixLookAtLH(&V, &position, &target, &up);

Device -> SetTransform(D3DTS\_VIEW, &V);

D3DXMATRIX proj;

D3DXMatrixPerspectiveFovLH(

&proj,

D3DX\_PI \* 0.5f,

(float)Width / (float)Height,

1.0f,

1000.0f);

Device -> SetTransform(D3DTS\_PROJECTION, &proj);

Device -> SetRenderState(D3DRS\_LIGHTING,true);

1. Cleanup()函数

d3d::Release<ID3DXMesh\*>(mesh);

for (int i = 0; i < Tex.size(); i++)

d3d::Release<IDirect3DTexture9\*>(Tex[i]);

1. Display()函数

Device->BeginScene();

D3DXMATRIX Ry;

static float y = 0.0f;

D3DXMatrixRotationY(&Ry,y);

y += timeDelta;

Device->SetTransform(D3DTS\_WORLD,&Ry);

for(int i = 0; i < Mtrls.size(); i++)

{

Device->SetMaterial(&Mtrls[i]);

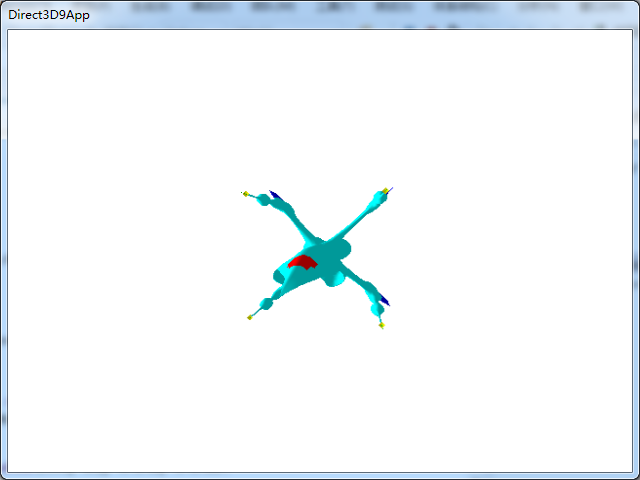
Device->SetTexture(0,Tex[i]);

mesh->DrawSubset( i );

}

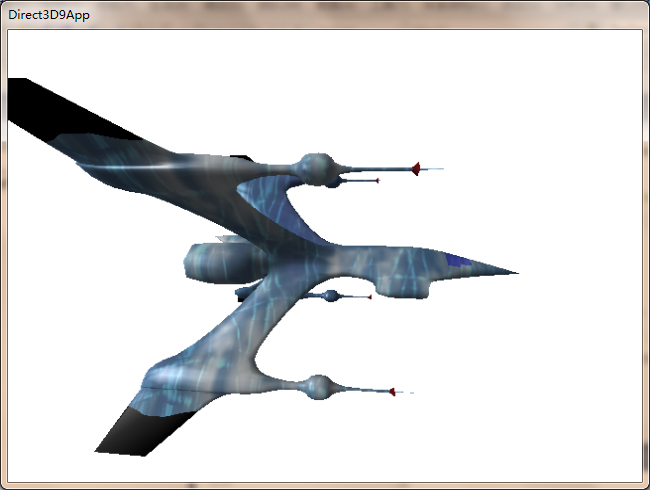
Device->EndScene();

1. 程序运行结果



1. **总结**
2. 在读取材质数据那里创建一个动态材质数组，然后分别压入不同的材质，与老师讲的每个子集赋予不同的材质有相似之处，也有不同。
3. **实验3**
4. 题目

使用网格绘制飞行器，飞行器绕Y轴连续旋转，效果如下：



1. 实现代码

在d3dInit框架的基础上添加代码：

1. 全局变量

#include <vector>

const int Width = 640;

const int Height = 480;

IDirect3DDevice9\* Device = 0;

ID3DXMesh\* mesh = 0;

std::vector<D3DMATERIAL9> Mtrls(0);

std::vector<IDirect3DTexture9\*> Tex(0);

1. Setup()函数

IDirect3DTexture9\* texture = 0;

D3DXCreateTextureFromFile(

Device,

"ice.bmp",

&texture);

HRESULT hr = 0;

ID3DXBuffer\* adjBuffer = 0;

ID3DXBuffer\* mtrlBuffer = 0;

DWORD numMtrls = 0;

hr = D3DXLoadMeshFromX(

"bigship1.x",

D3DXMESH\_MANAGED,

Device,

&adjBuffer,

&mtrlBuffer,

0,

&numMtrls,

&mesh);

if(FAILED(hr))

{

::MessageBox(0,"D3DXLoadMeshFromX() - FAILED",0,0);

return false;

}

if(mtrlBuffer != 0 && numMtrls != 0)

{

D3DXMATERIAL\* mtrls = (D3DXMATERIAL\*)mtrlBuffer->GetBufferPointer();

for (int i = 0; i < numMtrls; i++)

{

mtrls[i].MatD3D.Ambient = mtrls[i].MatD3D.Diffuse;

Mtrls.push\_back (mtrls[i].MatD3D);

if(mtrls[i].pTextureFilename != 0)

{

IDirect3DTexture9\* tex = 0;

D3DXCreateTextureFromFile(

Device,

mtrls[i].pTextureFilename,

&tex);

//Tex.push\_back(texture);

Tex.push\_back(tex);

}

else

{

//Tex.push\_back(0);

Tex.push\_back(texture);

}

}

}

Device->SetSamplerState(0, D3DSAMP\_MAGFILTER, D3DTEXF\_LINEAR);

Device->SetSamplerState(0, D3DSAMP\_MINFILTER, D3DTEXF\_LINEAR);

Device->SetSamplerState(0, D3DSAMP\_MIPFILTER, D3DTEXF\_POINT);

d3d::Release<ID3DXBuffer\*>(mtrlBuffer);

DWORD\* attributeBuffer = 0;

mesh->LockAttributeBuffer(0, &attributeBuffer);

for(int a = 0; a < 4; a++)

attributeBuffer[a] = 0;

for(int b = 4; b < 8; b++)

attributeBuffer[b] = 1;

for(int c = 8; c < 12; c++)

attributeBuffer[c] = 2;

mesh->UnlockAttributeBuffer();

std::vector<DWORD> adjacencyBuffer(mesh->GetNumFaces() \* 3);

mesh->GenerateAdjacency(0.0f, &adjacencyBuffer[0]);

hr = mesh->OptimizeInplace(

D3DXMESHOPT\_ATTRSORT |

D3DXMESHOPT\_COMPACT |

D3DXMESHOPT\_VERTEXCACHE,

&adjacencyBuffer[0],

0, 0, 0);

d3d::Release<ID3DXBuffer\*>(adjBuffer);

D3DXVECTOR3 dir(1.0f,0.0f,0.0f);

D3DXCOLOR c = d3d::WHITE;

D3DLIGHT9 light = d3d::InitDirectionalLight(&dir, &c);

Device->SetLight(0,&light);

Device->LightEnable(0,true);

Device->SetRenderState(D3DRS\_NORMALIZENORMALS,true);

D3DXVECTOR3 position(0.0f, 0.0f, -20.0f);

D3DXVECTOR3 target(0.0f, 0.0f, 0.0f);

D3DXVECTOR3 up(0.0f, 1.0f, 0.0f);

D3DXMATRIX V;

D3DXMatrixLookAtLH(&V, &position, &target, &up);

Device -> SetTransform(D3DTS\_VIEW, &V);

D3DXMATRIX proj;

D3DXMatrixPerspectiveFovLH(

&proj,

D3DX\_PI \* 0.5f,

(float)Width / (float)Height,

1.0f,

1000.0f);

Device -> SetTransform(D3DTS\_PROJECTION, &proj);

Device -> SetRenderState(D3DRS\_LIGHTING,true);

1. Cleanup()函数

d3d::Release<ID3DXMesh\*>(mesh);

for (int i = 0; i < Tex.size(); i++)

d3d::Release<IDirect3DTexture9\*>(Tex[i]);

1. Display()函数

Device->BeginScene();

D3DXMATRIX Ry;

static float y = 0.0f;

D3DXMatrixRotationY(&Ry,y);

y += timeDelta;

Device->SetTransform(D3DTS\_WORLD,&Ry);

for(int i = 0; i < Mtrls.size(); i++)

{

Device->SetMaterial(&Mtrls[i]);

Device->SetTexture(0,Tex[i]);

mesh->DrawSubset( i );

}

Device->EndScene();

1. 程序运行结果



1. **总结**
2. 文件“bigship1.x”来自DirectX SDK根目录下的Sample\Media\misc文件夹。