Họ và tên: Hoàng Minh Hải Đăng

MSSV: 2110120

**ĐỒ HỌA MÁY TÍNH (CO3059)**

**LAB MESH**

Hàm *CreateShape1(…)* tạo lưới đa giác mô phỏng hình nón, với mặt đáy nằm trên mặt phẳng *xOz*, trục hình nón trùng với trục *Oy*. Tham số gồm bán kính đáy, chiều cao hình nón và số Slice (trong hình vẽ phía dưới, Slice = 10). Hàm được hiện thực như sau:

void Mesh::CreateShape1(float radius, float height, int nSlice)

{

numVerts = nSlice + 1;

pt = new Point3[numVerts];

pt[0].set(0, height, 0);

for(int i = 0; i < nSlice; i++){

float angle = 2\*PI\* i/nSlice;

float x = radius \* sin(angle);

float z = radius \* cos(angle);

pt[i+1].set(x, 0, z);

}

numFaces = nSlice;

face = new Face[numFaces];

for(int i = 0; i < nSlice; i++){

face[i].nVerts = 3;

face[i].vert = new VertexID[3];

face[i].vert[0].vertIndex = 0;

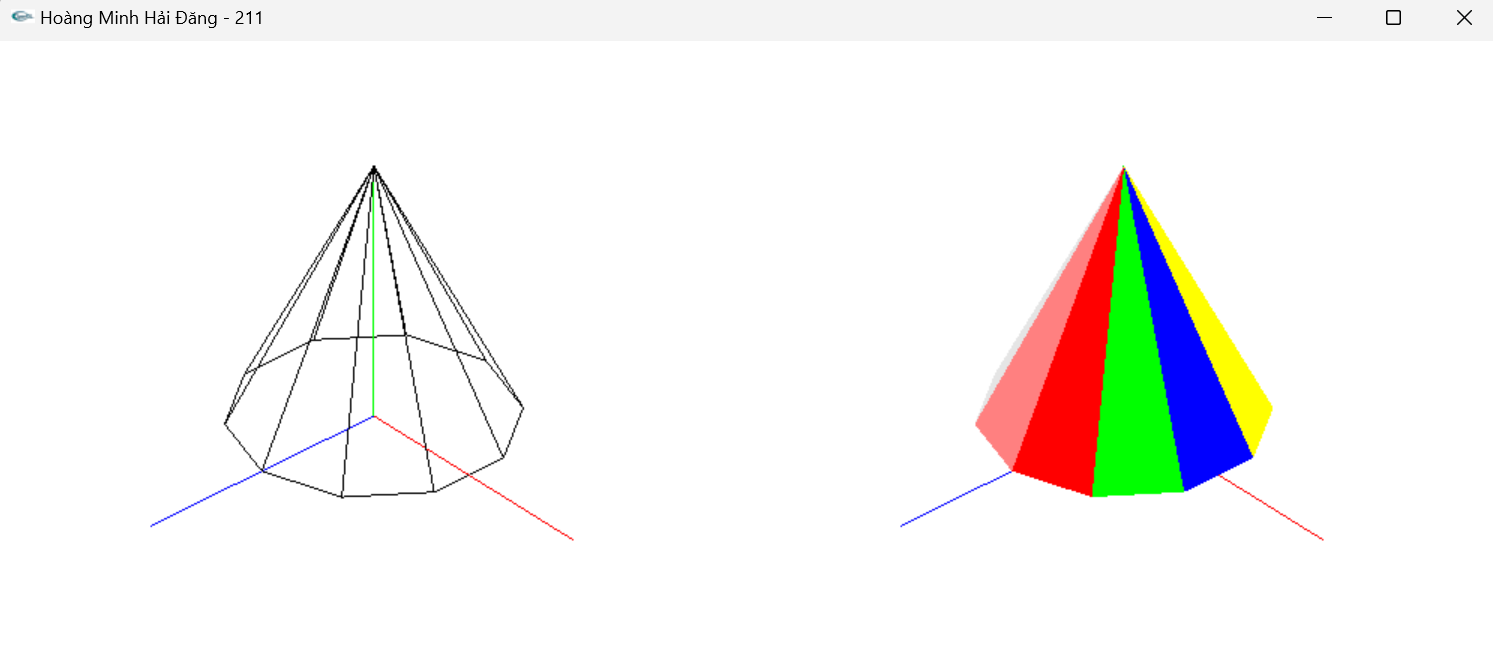
face[i].vert[1].vertIndex = i % nSlice + 1;

face[i].vert[2].vertIndex = (i + 1) % nSlice + 1;

}

}

Tham số thực cho lời gọi hàm: shape1.CreateShape1(2, 4, 10)



Hàm *CreateShape2(…)* tạo lưới đa giác mô phỏng hình lập phương bị đục từ trên xuống dưới bởi 1 hình trụ. Tâm của hình lập phương trùng với gốc tọa độ, các trục tọa độ vuông góc với các mặt của hình lập phương. Tham số gồm chiều dài cạnh hình lập phương, bán kính hình trụ và số Slice (lưu ý Slice phải chia hết cho 8). Hàm được hiện thực như sau:

void Mesh::CreateShape2(float fSize, float radius, int nSlice)

{

if(nSlice % 8 != 0)

throw("So slice phai chia het cho 8");

float d = 360 / nSlice;

float dL = fSize/ (nSlice / 4);

numVerts = 2 \* 2 \* nSlice;

numFaces = 3 \* nSlice + 4;

pt = new Point3[numVerts];

for (int i = 0; i < nSlice / 4; i++) {

pt[i].set(fSize, fSize, (nSlice/4-i)\*dL);

pt[i+ 2\*nSlice].set(fSize, 0, (nSlice/4-i)\*dL);

pt[i + nSlice/4].set((nSlice/4-i)\*dL, fSize, 0);

pt[i + nSlice/4 + 2\*nSlice].set((nSlice/4-i)\*dL, 0, 0);

pt[i + 2\*nSlice/4].set(0, fSize, i\*dL);

pt[i + 2\*nSlice/4 + 2\*nSlice].set(0, 0, i\*dL);

pt[i + 3\*nSlice/4].set(i\*dL, fSize, fSize);

pt[i + 3\*nSlice/4 + 2\*nSlice].set(i\*dL, 0, fSize);

}

for (int i = 0; i < nSlice; i++) {

float alpha = (45 + i\*d)\*PI/180;

float x = fSize/2.0 + radius \* sin(alpha);

float z = fSize/2.0 + radius \* cos(alpha);

pt[i + nSlice].set(x, fSize, z);

pt[i + 3\*nSlice].set(x, 0, z);

}

face = new Face[numFaces];

for (int i = 0; i < nSlice; i++) {

int next = (i + 1) % nSlice;

face[i].nVerts = 4;

face[i].vert = new VertexID[4];

face[i].vert[0].vertIndex = i + nSlice;

face[i].vert[1].vertIndex = next + nSlice;

face[i].vert[2].vertIndex = next + 3\*nSlice;

face[i].vert[3].vertIndex = i + 3\*nSlice;

face[i + nSlice].nVerts = 4;

face[i + nSlice].vert = new VertexID[4];

face[i + nSlice].vert[0].vertIndex = i;

face[i + nSlice].vert[1].vertIndex = next;

face[i + nSlice].vert[2].vertIndex = next + nSlice;

face[i + nSlice].vert[3].vertIndex = i + nSlice;

face[i + 2\*nSlice].nVerts = 4;

face[i + 2\*nSlice].vert = new VertexID[4];

face[i + 2\*nSlice].vert[0].vertIndex = i + 2\*nSlice ;

face[i + 2\*nSlice].vert[1].vertIndex = next + 2\*nSlice ;

face[i + 2\*nSlice].vert[2].vertIndex = next + 3\*nSlice;

face[i + 2\*nSlice].vert[3].vertIndex = i + 3\*nSlice;

}

face[0 + 3\*nSlice].nVerts = 4;

face[0 + 3\*nSlice].vert = new VertexID[4];

face[0 + 3\*nSlice].vert[0].vertIndex = 0\*nSlice/4;

face[0 + 3\*nSlice].vert[1].vertIndex = 0\*nSlice/4 + nSlice / 4;

face[0 + 3\*nSlice].vert[2].vertIndex = 0\*nSlice/4 + nSlice / 4 + 2\*nSlice;

face[0 + 3\*nSlice].vert[3].vertIndex = 0\*nSlice/4 + 2\*nSlice;

face[1 + 3\*nSlice].nVerts = 4;

face[1 + 3\*nSlice].vert = new VertexID[4];

face[1 + 3\*nSlice].vert[0].vertIndex = 1\*nSlice/4;

face[1 + 3\*nSlice].vert[1].vertIndex = 1\*nSlice/4 + nSlice / 4;

face[1 + 3\*nSlice].vert[2].vertIndex = 1\*nSlice/4 + nSlice / 4 + 2\*nSlice;

face[1 + 3\*nSlice].vert[3].vertIndex = 1\*nSlice/4 + 2\*nSlice;

face[2 + 3\*nSlice].nVerts = 4;

face[2 + 3\*nSlice].vert = new VertexID[4];

face[2 + 3\*nSlice].vert[0].vertIndex = 2\*nSlice/4;

face[2 + 3\*nSlice].vert[1].vertIndex = 2\*nSlice/4 + nSlice / 4;

face[2 + 3\*nSlice].vert[2].vertIndex = 2\*nSlice/4 + nSlice / 4 + 2\*nSlice;

face[2 + 3\*nSlice].vert[3].vertIndex = 2\*nSlice/4 + 2\*nSlice;

face[3 + 3\*nSlice].nVerts = 4;

face[3 + 3\*nSlice].vert = new VertexID[4];

face[3 + 3\*nSlice].vert[0].vertIndex = 3\*nSlice/4;

face[3 + 3\*nSlice].vert[1].vertIndex = 0\*nSlice/4;

face[3 + 3\*nSlice].vert[2].vertIndex = 0\*nSlice/4 + 2\*nSlice;

face[3 + 3\*nSlice].vert[3].vertIndex = 3\*nSlice/4 + 2\*nSlice;

}

Tham số thực cho lời gọi hàm: shape2.CreateShape2(4, 2, 16)

