**案例39-Menger海绵的画家算法**

文档编写：霍波魏

校稿/修订：孔令德

时间2019~2020

联系方式：QQ997796978

**说明：**本套案例由孔令德开发，原版本为Visual C++6.0，配套于孔令德的著作《计算机图形学-基于MFC三维图形开发》一书。孔令德计算机工程研究所的学生霍波魏在学习计算机图形学期间，对本套案例进行了升级并编写了学习文档。现在程序的编写和程序的解释都是基于Windows 10操作系统，使用Microsoft visual studio 2017平台的MFC（英文版）开发。

1. **案例描述**

本案例通过画家算法对Menger海绵进行消隐。

1. **知识点**

本案例主要用画家算法对Menger海绵进行消隐，该算法的思路如下：

先把将物体的各个表面按z 坐标排序形成深度优先级表，z 大者位于表头，z 小者位于表尾。然后按照从表头到表尾的顺序，逐个取出三角形绘制到帧缓冲中，后绘制的表面覆盖先绘制的表面，相当于消除了隐藏面。

（1）按z 从大（远）到小（近）的顺序对所有三角形排序。

（2）解决z 方向上出现的三角形深度二义性问题，必要时对三角形进行分割，使每个三角形获得一个确定的深度优先级。

（3）按z 由大到小的顺序，依次扫描转换每一个三角形。

1. **实现步骤**
2. 添加基础类与添加绘制立方体的CCube类。
3. 在CProjection类中对透视变换参数初始化、设置视点位置以及透视变换。
4. 在CCube类中计算顶点坐标、读入面表、绘制图形。
5. 在CPainter类中定义了画家算法，通过冒泡排序对表面最大深度进行排序来消除隐藏面。
6. 在CTestView类中定义了一个八叉树，用来绘制递归深度为2时的Menger海绵。
7. 在CTestView类中调用绘制立方体函数以及画家算法，绘制Menger海绵中的每一个立方体。
8. 在CTestView类中添加消息响应函数，在OnDraw中调用DoubleBuffer函数。
9. **主要算法**

1. CCube类

public:

CCube(void);

virtual ~CCube(void);

void SetParameter(double nHalfEdge, CP3d ptCenter, CString NodeSign);//设置半边长与中心点和节点标志

void ReadVertex(void);//读入点表

void ReadFacet(void);//读入面表

void AppendQueue(CPainter\* pPainter);//绘制图形

public:

double nHalfEdge;//立方体半边长

CP3d ptCenter;//中心点

CP3d V[8];//点表

CFace F[6];//面表

CString NodeSign;//节点标记

CProjection projection;//投影

CCube::CCube(void)

{

ptCenter.x = 0;

ptCenter.y = 0;

ptCenter.z = 0;

}

CCube::~CCube(void)

{

}

void CCube::SetParameter(double nHalfEdge, CP3d ptCenter, CString NodeSign)

{

this->nHalfEdge = nHalfEdge;

this->ptCenter = ptCenter;

this->NodeSign = NodeSign;

}

void CCube::ReadVertex(void)//点表

{

V[0].x = ptCenter.x - nHalfEdge, V[0].y = ptCenter.y - nHalfEdge, V[0].z = ptCenter.z - nHalfEdge;

V[1].x = ptCenter.x + nHalfEdge, V[1].y = ptCenter.y - nHalfEdge, V[1].z = ptCenter.z - nHalfEdge;

V[2].x = ptCenter.x + nHalfEdge, V[2].y = ptCenter.y + nHalfEdge, V[2].z = ptCenter.z - nHalfEdge;

V[3].x = ptCenter.x - nHalfEdge, V[3].y = ptCenter.y + nHalfEdge, V[3].z = ptCenter.z - nHalfEdge;

V[4].x = ptCenter.x - nHalfEdge, V[4].y = ptCenter.y - nHalfEdge, V[4].z = ptCenter.z + nHalfEdge;

V[5].x = ptCenter.x + nHalfEdge, V[5].y = ptCenter.y - nHalfEdge, V[5].z = ptCenter.z + nHalfEdge;

V[6].x = ptCenter.x + nHalfEdge, V[6].y = ptCenter.y + nHalfEdge, V[6].z = ptCenter.z + nHalfEdge;

V[7].x = ptCenter.x - nHalfEdge, V[7].y = ptCenter.y + nHalfEdge, V[7].z = ptCenter.z + nHalfEdge;

}

void CCube::ReadFacet(void)//面表

{

F[0].vNumber = 4; F[0].vIndex[0] = 4; F[0].vIndex[1] = 5; F[0].vIndex[2] = 6; F[0].vIndex[3] = 7; F[0].c = CRGB(0.8, 0.5, 0.2);//前面

F[1].vNumber = 4; F[1].vIndex[0] = 0; F[1].vIndex[1] = 3; F[1].vIndex[2] = 2; F[1].vIndex[3] = 1; F[1].c = CRGB(0.8, 0.4, 0.3);//后面

F[2].vNumber = 4; F[2].vIndex[0] = 0; F[2].vIndex[1] = 4; F[2].vIndex[2] = 7; F[2].vIndex[3] = 3; F[2].c = CRGB(0.9, 0.5, 0.3);//左面

F[3].vNumber = 4; F[3].vIndex[0] = 1; F[3].vIndex[1] = 2; F[3].vIndex[2] = 6; F[3].vIndex[3] = 5; F[3].c = CRGB(0.9, 0.4, 0.2);//右面

F[4].vNumber = 4; F[4].vIndex[0] = 2; F[4].vIndex[1] = 3; F[4].vIndex[2] = 7; F[4].vIndex[3] = 6; F[4].c = CRGB(1.0, 0.7, 0.6);//顶面

F[5].vNumber = 4; F[5].vIndex[0] = 0; F[5].vIndex[1] = 1; F[5].vIndex[2] = 5; F[5].vIndex[3] = 4; F[5].c = CRGB(1.0, 0.5, 0.5);//底面

}

void CCube::AppendQueue( CPainter\* pPainter)

{

CP3d\* tempV = NULL;//临时点

if (pPainter->GV != NULL)

{

tempV = new CP3d[pPainter->nV];

for (int nVertex = 0; nVertex < pPainter->nV; nVertex++)

tempV[nVertex] = pPainter->GV[nVertex];

delete[]pPainter->GV;

pPainter->GV = NULL;

}

CFace\* tempF = NULL;//临时面

if (pPainter->GF != NULL)

{

tempF = new CFace[pPainter->nF];

for (int nFacet = 0; nFacet < pPainter->nF; nFacet++)

{

tempF[nFacet].vNumber = pPainter->GF[nFacet].vNumber;

for (int i = 0; i < tempF[nFacet].vNumber; i++)

{

tempF[nFacet].vIndex[i] = pPainter->GF[nFacet].vIndex[i];

tempF[nFacet].c = pPainter->GF[nFacet].c;

}

}

delete[]pPainter->GF;

pPainter->GF = NULL;

}

int NumberofVextex = 8;//物体的顶点数

int NumberofFacet = 6;//物体的面片数

pPainter->nV = pPainter->nV + NumberofVextex;

pPainter->nF = pPainter->nF + NumberofFacet;

pPainter->GV = new CP3d[pPainter->nV];

pPainter->GF = new CFace[pPainter->nF];

//复制顶点

if (tempV != NULL)

{

for (int nVertex = 0; nVertex < pPainter->nV - NumberofVextex; nVertex++)

pPainter->GV[nVertex] = tempV[nVertex];

}

for (int nVertex = 0; nVertex < NumberofVextex; nVertex++)

pPainter->GV[pPainter->nV - NumberofVextex + nVertex] = V[nVertex];

//复制面片

if (tempF != NULL)

{

for (int nFacet = 0; nFacet < pPainter->nF - NumberofFacet; nFacet++)

{

pPainter->GF[nFacet].vNumber = tempF[nFacet].vNumber;

for (int i = 0; i < pPainter->GF[nFacet].vNumber; i++)

{

pPainter->GF[nFacet].vIndex[i] = tempF[nFacet].vIndex[i];

pPainter->GF[nFacet].c = tempF[nFacet].c;

}

}

}

for (int nFacet = 0; nFacet < NumberofFacet; nFacet++)

{

pPainter->GF[pPainter->nF - NumberofFacet + nFacet].vNumber = F[nFacet].vNumber;

for (int n = 0; n < pPainter->GF[pPainter->nF - NumberofFacet + nFacet].vNumber; n++)

{

pPainter->GF[pPainter->nF - NumberofFacet + nFacet].vIndex[n] = F[nFacet].vIndex[n];

pPainter->GF[pPainter->nF - NumberofFacet + nFacet].c = F[nFacet].c;

}

}

if (tempV != NULL)

{

delete[]tempV;

tempV = NULL;

}

if (tempF != NULL)

{

delete[]tempF;

tempF = NULL;

}

}

2.CPainter类

public:

CPainter(void);

virtual~CPainter(void);

void BubbleSort(void);//冒泡排序

double MaxDepth(double\* Depth, int nNum);//面的最大深度

void DrawObject(CDC\* pDC);

public:

CP3d\* GV;//场景中总顶点数组

CFace\* GF;//场景中总表面数组

int nV;//点的总数

int nF;//面的总数

CProjection projection;//投影

CPainter::CPainter(void)

{

GV = NULL;

GF = NULL;

nV = 0;

nF = 0;

}

CPainter::~CPainter(void)

{

if (NULL != GV)

{

delete[]GV;

GV = NULL;

}

if (NULL != GF)

{

delete[]GF;

GF = NULL;

}

}

void CPainter::DrawObject(CDC\* pDC)

{

for (int nFacet = 0; nFacet < nF; nFacet++)//更新面表索引号

for (int nVertex = 0; nVertex < GF[nFacet].vNumber; nVertex++)

GF[nFacet].vIndex[nVertex] = 8 \* (nFacet / 6) + GF[nFacet].vIndex[nVertex];

double DepthVertex[4];//面4个顶点的深度

for (int nFacet = 0; nFacet < nF; nFacet++)

{

for (int nVertex = 0; nVertex < GF[nFacet].vNumber; nVertex++)

{

DepthVertex[nVertex] = projection.PerspectiveProjection(GV[GF[nFacet].vIndex[nVertex]]).z;

}

GF[nFacet].fDepth = MaxDepth(DepthVertex, GF[nFacet].vNumber);//面的最大深度

}

BubbleSort();

for (int nFacet = 0; nFacet < nF; nFacet++)

{

CBrush NewBrush;

CP2d ScreenPoint, temp;

NewBrush.CreateSolidBrush(RGB(GF[nFacet].c.red \* 255, GF[nFacet].c.green \* 255, GF[nFacet].c.blue \* 255));

pDC->SelectObject(&NewBrush);

pDC->BeginPath();

for (int nVertex = 0; nVertex < GF[nFacet].vNumber; nVertex++)//顶点循环

{

ScreenPoint = projection.PerspectiveProjection(GV[GF[nFacet].vIndex[nVertex]]);//投影

if (0 == nVertex)

{

pDC->MoveTo(Round(ScreenPoint.x), Round(ScreenPoint.y));

temp = ScreenPoint;

}

else

pDC->LineTo(Round(ScreenPoint.x), Round(ScreenPoint.y));

}

pDC->LineTo(Round(temp.x), Round(temp.y));//闭合四边形

pDC->EndPath();

pDC->StrokeAndFillPath();//填充四边形

}

}

void CPainter::BubbleSort(void)

{

for (int i = 0; i < nF - 1; i++)

for (int j = 0; j < nF - 1 - i; j++)

if (GF[j].fDepth < GF[j + 1].fDepth)

{

CFace Temp;

Temp = GF[j];

GF[j] = GF[j + 1];

GF[j + 1] = Temp;

}

}

double CPainter::MaxDepth(double\* Depth, int nNum)//面的最大深度

{

double maxDepth = Depth[0];

for (int i = 1; i < nNum; i++)

if (maxDepth < Depth[i])

maxDepth = Depth[i];

return maxDepth;

}

3.CTestView类

public:

void DoubleBuffer(CDC\* pDC);//双缓冲绘图

void Tstree(int nLevel, double nEdge, CP3d VCenter, CString NodeSign);//27叉树

void DrawObject(CDC\* pDC);//绘制对象

protected:

int NumofObject;//小立方体的个数

CCube\* pCube;//立方体对象

CTransform3\* pTransform;//变换

double Alpha, Beta;//x方向旋转α角,y方向旋转β角

BOOL bPlay;//动画开关

void CTestView::DoubleBuffer(CDC\* pDC)//双缓冲绘图

{

CRect rect;

GetClientRect(&rect);

pDC->SetMapMode(MM\_ANISOTROPIC);

pDC->SetWindowExt(rect.Width(), rect.Height());

pDC->SetViewportExt(rect.Width(), -rect.Height());

pDC->SetViewportOrg(rect.Width() / 2, rect.Height() / 2);

CDC memDC;//声明内存DC

memDC.CreateCompatibleDC(pDC);//创建一个与显示DC兼容的内存DC

CBitmap NewBitmap, \*pOldBitmap;

NewBitmap.CreateCompatibleBitmap(pDC, rect.Width(), rect.Height());//创建兼容内存位图

pOldBitmap = memDC.SelectObject(&NewBitmap);//将兼容位图选入内存DC

//memDC.FillSolidRect(rect,pDC->GetBkColor());//设置客户区背景色

rect.OffsetRect(-rect.Width() / 2, -rect.Height() / 2);

memDC.SetMapMode(MM\_ANISOTROPIC);//内存DC自定义坐标系

memDC.SetWindowExt(rect.Width(), rect.Height());

memDC.SetViewportExt(rect.Width(), -rect.Height());

memDC.SetViewportOrg(rect.Width() / 2, rect.Height() / 2);

DrawObject(&memDC);//绘制图形

pDC->BitBlt(rect.left, rect.top, rect.Width(), rect.Height(), &memDC, -rect.Width() / 2, -rect.Height() / 2, SRCCOPY); //将内存DC中的位图拷贝到显示DC

memDC.SelectObject(pOldBitmap);

NewBitmap.DeleteObject();

memDC.DeleteDC();

}

void CTestView::DrawObject(CDC\* pDC)//绘制图形

{

CPainter\* pPainter = new CPainter;

for (int nNode = 0; nNode < NumofObject; nNode++)

if (pCube[nNode].NodeSign == CString("F"))

pCube[nNode].AppendQueue(pPainter);

pPainter->DrawObject(pDC);

delete pPainter;

pPainter = NULL;

}

void CTestView::Tstree(int nLevel, double nEdge, CP3d VCenter, CString NodeSign = \_T("F"))

{

if (0 == nLevel)

{

pCube[NumofObject].SetParameter(nEdge / 2, VCenter, NodeSign);

pCube[NumofObject].ReadVertex();

pCube[NumofObject].ReadFacet();

pTransform[NumofObject].SetMatrix(pCube[NumofObject].V, 8);

NumofObject++;

}

else

{

if (\_T("F") == NodeSign)

{

CP3d ptCenter;//立方体中心

double TempEdge = nEdge / 3.0;//立方体临时边长

//000

ptCenter = VCenter + CP3d(-TempEdge, -TempEdge, +TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//100

ptCenter = VCenter + CP3d(0, -TempEdge, +TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//200

ptCenter = VCenter + CP3d(+TempEdge, -TempEdge, +TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//010

ptCenter = VCenter + CP3d(-TempEdge, 0, +TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//110

ptCenter = VCenter + CP3d(0, 0, +TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter, \_T("E"));

//210

ptCenter = VCenter + CP3d(+TempEdge, 0, +TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//020

ptCenter = VCenter + CP3d(-TempEdge, +TempEdge, +TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//120

ptCenter = VCenter + CP3d(0, +TempEdge, +TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//220

ptCenter = VCenter + CP3d(+TempEdge, +TempEdge, +TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//001

ptCenter = VCenter + CP3d(-TempEdge, -TempEdge, 0);

Tstree(nLevel - 1, TempEdge, ptCenter);

//101

ptCenter = VCenter + CP3d(0, -TempEdge, 0);

Tstree(nLevel - 1, TempEdge, ptCenter, \_T("E"));

//201

ptCenter = VCenter + CP3d(+TempEdge, -TempEdge, 0);

Tstree(nLevel - 1, TempEdge, ptCenter);

//011

ptCenter = VCenter + CP3d(-TempEdge, 0, 0);

Tstree(nLevel - 1, TempEdge, ptCenter, \_T("E"));

//111

ptCenter = VCenter + CP3d(0, 0, 0);

Tstree(nLevel - 1, TempEdge, ptCenter, \_T("E"));

//211

ptCenter = VCenter + CP3d(+TempEdge, 0, 0);

Tstree(nLevel - 1, TempEdge, ptCenter, \_T("E"));

//021

ptCenter = VCenter + CP3d(-TempEdge, +TempEdge, 0);

Tstree(nLevel - 1, TempEdge, ptCenter);

//121

ptCenter = VCenter + CP3d(0, +TempEdge, 0);

Tstree(nLevel - 1, TempEdge, ptCenter, \_T("E"));

//221

ptCenter = VCenter + CP3d(+TempEdge, +TempEdge, 0);

Tstree(nLevel - 1, TempEdge, ptCenter);

//002

ptCenter = VCenter + CP3d(-TempEdge, -TempEdge, -TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//102

ptCenter = VCenter + CP3d(0, -TempEdge, -TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//202

ptCenter = VCenter + CP3d(+TempEdge, -TempEdge, -TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//012

ptCenter = VCenter + CP3d(-TempEdge, 0, -TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//112

ptCenter = VCenter + CP3d(0, 0, -TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter, \_T("E"));

//212

ptCenter = VCenter + CP3d(+TempEdge, 0, -TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//022

ptCenter = VCenter + CP3d(-TempEdge, +TempEdge, -TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//122

ptCenter = VCenter + CP3d(0, +TempEdge, -TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

//222

ptCenter = VCenter + CP3d(+TempEdge, +TempEdge, -TempEdge);

Tstree(nLevel - 1, TempEdge, ptCenter);

}

}

}

1. **实现效果**

Menger海绵的画家算法效果如图39-1。

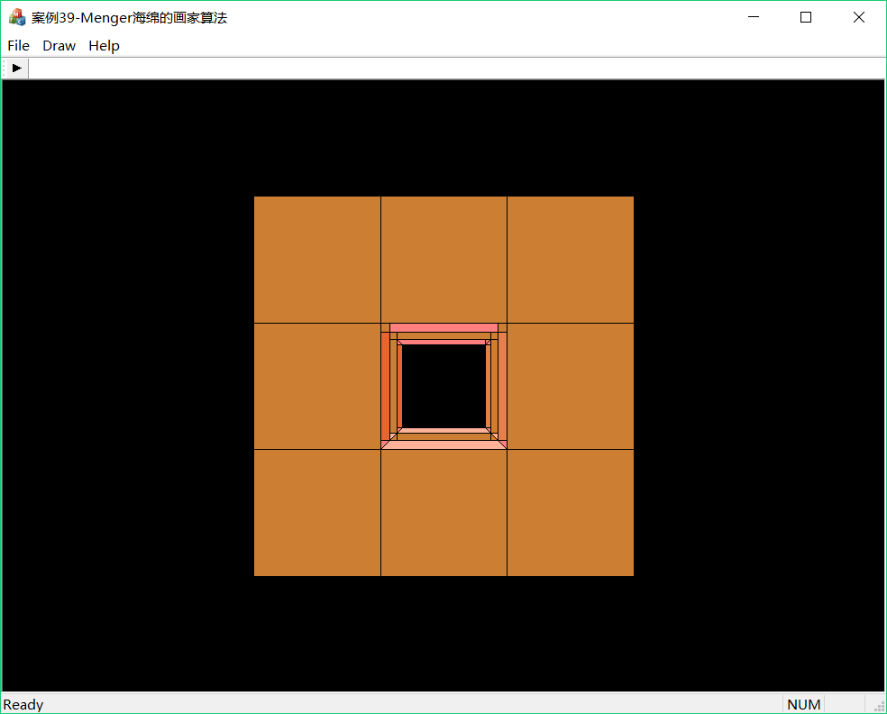


图39-1 Menger海绵的画家算法效果图