**计算机图形学大作业报告**

姓名：田宇

学号：14231006

**任务一：**

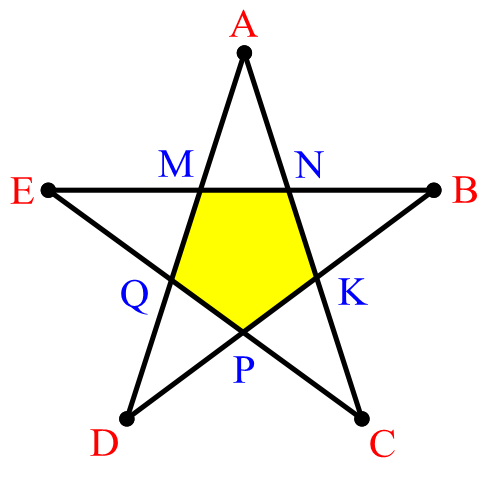
要求：创建窗口，以线框绘制一个五角星（点、线框可见），填充五角星中间的五边形

代码：FPointedStar.c

可执行文件：Star.exe

图解：

以五角星对称中心为原点坐标，给定C点的横坐标（作为参考尺寸）通过数学方法（三角函数等）计算出五角星各顶点的坐标



实现关键：

1. 填充五边形：

glColor3f(1, 1, 0);

glBegin(GL\_POLYGON);

glVertex2fv(PointM);

glVertex2fv(PointN);

glVertex2fv(PointK);

glVertex2fv(PointP);

glVertex2fv(PointQ);

glEnd();

1. 描点：点的大小默认为1个像素，可使用glPointSize改变

glColor3f(0, 0, 0);

glPointSize(5.0f);

glBegin(GL\_POINTS);

glVertex2fv(PointA);

glVertex2fv(PointD);

glVertex2fv(PointB);

glVertex2fv(PointE);

glVertex2fv(PointC);

glEnd();

1. 线框：（直线可以使用glLineWidth指定宽度）

glBegin(GL\_LINE\_LOOP);

glVertex2fv(PointA);

glVertex2fv(PointD);

glVertex2fv(PointB);

glVertex2fv(PointE);

glVertex2fv(PointC);

glEnd();

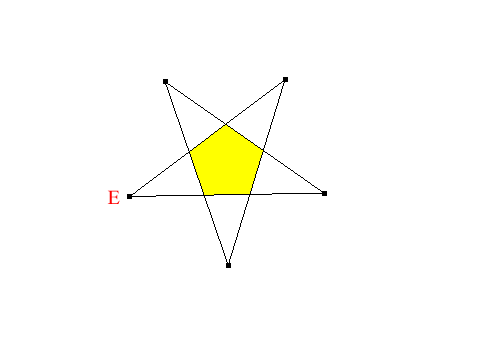
**任务二：**

要求：将绘制的五角星绕某个顶点进行旋转

代码文件：rotatedStar.c

可执行文件：rotStar.exe

图解：E点为旋转中心



实现关键：

1. 确定旋转中心：

使用glRotatef设置旋转轴相关参数，必要时借助glTranslatef平移坐标系

glTranslatef(PointE[0], PointE[1], 0.0f);

glRotatef(rot\_angle, 0.0f, 0.0f, 1.0f);

glTranslatef(-PointE[0], -PointE[1], 0.0f);

1. 二维自动旋转：

void rotIdle()

{

rot\_angle += 0.01f;

if (rot\_angle > 360.0f) rot\_angle -= 360.0f;

glutPostRedisplay();

}

//glutIdleFunc(rotIdle);

**任务三：**

要求：

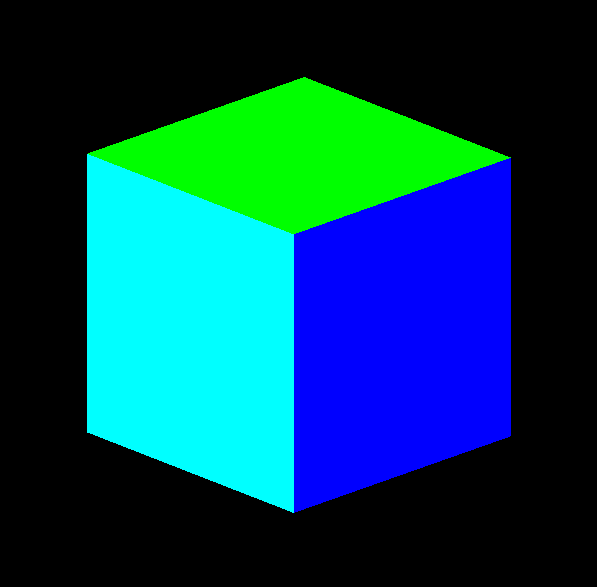
以透视投影方式绘制一个彩色立方体，该立方体可以在三维空间中自动旋转（旋转轴随意）

代码文件：rotatedCube.c

可执行文件：rotCube.exe

图解：

以立方体的旋转中心为坐标原点，给定立方体的边长信息，计算得到其他顶点的坐标，然后进行填充绘制，注意消隐



实现关键：

1. 设置消隐：使用glEnable(GL\_CULL\_FACE)启动剔除功能，GL\_BACK表示背面

#if 0

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE);

#endif

1. 各面设置不同颜色填充绘制：使用glColor3f函数调整RGB模型参数

glColor3f(1, 0, 0);

glBegin(GL\_QUADS);

glVertex3fv(VertexA);

glVertex3fv(VertexB);

glVertex3fv(VertexC);

glVertex3fv(VertexD);

glEnd();

1. 投影变换：

使用glEnable(GL\_DEPTH\_TEST)开启深度测试，当窗口大小改变时需使用Reshape函数做处理，主要包括使用glViewPort指定视口位置及大小，使用gluPerspective设置投影矩阵

void reShape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, 1.0 \* w / h, 1, 1000);

glEnable(GL\_DEPTH\_TEST);

gluLookAt(0, 0, 5, 0, 0, 0, 0, 1, 0);

}

//glutReshapeFunc(&reShape);

1. 确定旋转轴：

glRotatef(rotX, 1.0f, 0.0f, 0.0f);

glRotatef(rotY, 0.0f, 1.0f, 0.0f);

1. 三维自动旋转：使用glutIdleFunc调整旋转角度参数实现连续动画

void chgIdle()

{

rotX += 0.01f;

rotY += 0.02f;

if (rotX > 360.0f) rotX -= 360.0f;

if (rotY > 360.0f) rotY -= 360.0f;

glutPostRedisplay();

}

//glutIdleFunc(chgIdle);

**任务四：**

要求：

绘制一个三维旋转物体（茶壶），在场景中设置光源，一个白色光源和一个红色光源，对称设置，光源不随物体的旋转而移动

代码文件：rotatedTeapot.c

可执行文件：rotTeapot.exe

图解：



实现关键：

1. 使用glut库函数绘制茶壶：（glutSolidTeapot）

使用双缓冲技术，调用glutSwapBuffers交换两个缓冲区（指针），把绘制好的信息用于屏幕显示

void drawTPot(void)

{

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(0.0,0.0,5.0,0.0,0.0,0.0,0.0,1.0,0.0);

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

glPushMatrix();

glRotatef(angle,0.0,1.0,0.0);

glutSolidTeapot(1.0);

glPopMatrix();

glutSwapBuffers();

}

2）设置光源：

光照模型中的反射光分为ambient、diffuse和specular三个分量，使用glLightfv创建光源，GL\_POSITION用于定位光源

void setLight()

{

//light0: no ambient + white diffuse&specular

const GLfloat light\_ambient0[]={0.0f,0.0f,0.0f,1.0f};

const GLfloat light\_diffuse0[]={1.0f,1.0f,1.0f,1.0f};

const GLfloat light\_specular0[]={1,1.0f,1.0f,1.0f};

const GLfloat light\_position0[]={-10,10,10,0};

//light1: no ambient + red diffuse&specular...

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHT1);

glEnable(GL\_NORMALIZE);

glEnable(GL\_LIGHTING);

//light\_white

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient0);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse0);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular0);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position0);

//light\_red...

}

**任务五：**

要求：

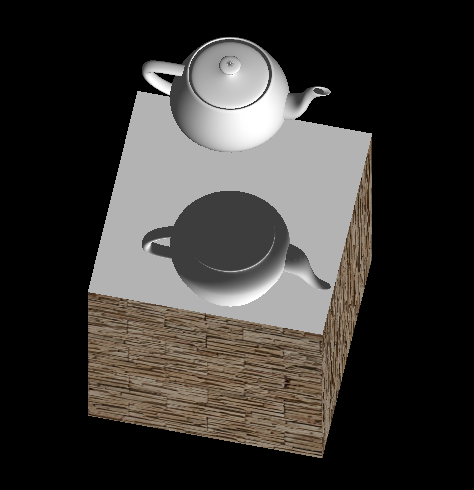
绘制两个三维物体（一个立方体和一个茶壶），在立方体上进行纹理映射（一个立方体面上贴纹理，一个立方体面上反射茶壶）

代码文件：texMapping.c

可执行文件：texCube.exe

纹理图片：TT.bmp

图解：为美观考虑，在一个立方体面上反射茶壶，其他所有立方体面上均进行纹理贴图



实现关键：

1. 加载纹理：

不使用OpenGL辅助库（GLAUX.H相关），读取bmp文件结构，获取数据信息用于定义纹理，使用glGenTextures标记纹理，glBindTexture用于创建一个新的纹理对象并绑定前述标记，调用gluBuild2DMipmaps为纹理对象填充位图资源，glTexParameteri用于设置一些纹理相关的参数

void loadTex()

{

FILE\* img = fopen(imgFile, "rb");

fseek(img, 18L, SEEK\_SET);

fread(&tWidth, 4, 1, img);

fread(&tHeight, 4, 1, img);

fseek(img, 0, SEEK\_END);

pLen = ftell(img) - 54;

pData = (GLubyte\*)malloc(pLen);

fseek(img, 54, SEEK\_SET);

fread(pData, pLen, 1, img);

fclose(img);

glGenTextures(1, &texId);

glBindTexture(GL\_TEXTURE\_2D, texId);

gluBuild2DMipmaps...

glTexParameteri...

free(pData);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

}

1. 设置环境光：

void initLight()

{

//light: ambient + white diffuse(10,10,10)

const GLfloat light\_ambient[] = {1.0f,1.0f,1.0f,1.0f};

const GLfloat light\_diffuse[] = {1.0f,1.0f,1.0f,1.0f};

const GLfloat light\_position[] = { 10, 10, 10, 0 };

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glEnable(GL\_DEPTH\_TEST);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHTING);

}

1. 纹理贴图：

使用glEnable(GL\_TEXTURE\_2D)开启纹理映射功能，glBindTexture用于使标记对应的纹理对象称为当前的活动纹理，glTexCoord2f用于指定纹理的坐标

glEnable(GL\_TEXTURE\_2D);

glBindTexture(GL\_TEXTURE\_2D, texId);

glBegin(GL\_QUADS);

glTexCoord2f(0.0f, 0.0f); glVertex3fv(VertexA);

glTexCoord2f(1.0f, 0.0f); glVertex3fv(VertexB);

glTexCoord2f(1.0f, 1.0f); glVertex3fv(VertexC);

glTexCoord2f(0.0f, 1.0f); glVertex3fv(VertexD);

glEnd();

1. 绘制镜面：

使用glEnable(GL\_STENCIL\_TEST)开启模板测试功能，glStencilFunc用于指定模板值1&0xff，绘制镜面时使用glDepthMask(GL\_FALSE)设置深度值为只读，注意镜面的透明性

glEnable(GL\_STENCIL\_TEST);

glStencilFunc(GL\_ALWAYS, 1, 0xff);

glStencilOp(GL\_KEEP, GL\_KEEP, GL\_REPLACE);

glDepthMask(GL\_FALSE);

glColor4f(1.0f, 1.0f, 1.0f, 0.5f);

glBegin(GL\_QUADS);

glNormal3f(0.0f,1.0f,0.0f);

glVertex3f(-hfEdge, -hfEdge, -hfEdge);

glVertex3f(-hfEdge, -hfEdge, hfEdge);

glVertex3f(hfEdge, -hfEdge, hfEdge);

glVertex3f(hfEdge, -hfEdge, -hfEdge);

glEnd();

glDepthMask(GL\_TRUE);

glStencilFunc(GL\_EQUAL, 1, 0xff);

glStencilFunc(GL\_KEEP, GL\_KEEP, GL\_KEEP);

1. 反射茶壶：使用glScalef将反射的茶壶像倒置

glPushMatrix();

glTranslatef(0.0f, -0.5f, 0.0f);

glScalef(1.0, -1.0, 1.0);

glutSolidTeapot(0.15);

glTranslatef(0.0f, 0.5f, 0.0f);

glPopMatrix();

glDisable(GL\_STENCIL\_TEST);