南京工程学院

实　验　报　告

课 程 名 称　虚拟现实 2020

实验项目名称　OpenGL 基本图形元素和模型变换

实验学生班级　数嵌 172

实验学生姓名　朱广锋

学　　号　202170638

同组学生姓名　无

实 验 时 间　2020.5.21

实 验 地 点

一、 实验主题

利用 VC 集成开发环境，实现基本图形元素绘制和坐标变换。

二、 实验准备

1.打开 Visual Studio 并且设置好工作目录；

2.下载 OpenGL 安装包所需文件

（http://d.download.csdn.net/down/2560229/ssagnn23），主要包括 GL.H GLAUX.H GLU.H glut.h

GLAUX.LIB GLU32.LIB glut32.lib glut.lib OPENGL32.LIB

glaux.dll glu32.dll glut32.dll glut.dll opengl32.dll

3.复制并配置 OpenGL 库函数到指定的目录（.h、.lib、.dll 分别放到 MSVC include、lib 和系统 Path 路径如 System32），检查复制后是否文件已经存在于指定目录下。

在 VSStudio 中建立一个空类型的项目，项目名为 Excer2_Graphic。其下有四个子任务：

---Excer2_Graphic

--Excer2_drawing.cpp

--Excer2_ polygon.cpp

--Excer2_cube.cpp

--Excer2_solar.cpp

三、 主要数据源、库函数、变量、涉及函数及其解释

```
// Excer2_drawing.cpp
#include <GL/glut.h>
#include <stdlib.h>

#define drawOneLine(x1,y1,x2,y2)  glBegin(GL_LINES);  \
   glVertex2f ((x1),(y1)); glVertex2f ((x2),(y2)); glEnd();
void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
}
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0, 0);
    glEnable(GL_LINE_STIPPLE);
    glLineStipple(1, 0B0000000100000001);  /*  dotted  */
    drawOneLine(50.0, 125.0, 150.0, 125.0);
    glLineStipple(1, 0B0000000011111111);  /*  dashed  */
    drawOneLine(150.0, 125.0, 250.0, 125.0);
    glLineStipple(1, 0B0001110001000111);  /*  dash/dot/dash  */
    drawOneLine(250.0, 125.0, 350.0, 125.0);
    glDisable(GL_LINE_STIPPLE);
    glFlush();
}
void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    /* 规定二维视景区域，参数分别为left,right,bottom,top  */
    gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(400, 150);
    glutInitWindowPosition(100, 100);
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
```

```
    glutMainLoop();
    return 0;
}
```

添加代码段含义：

设置绘制颜色红色， 启用线段，依次设置并绘制 点、线段、点-线 类型的线条，最后关闭线形绘制模式

函数解释：

glColor3f 设置顶点颜色（0.0f-1.0f）

glLineStipple 设置当前线形，参数为重复因子和点位模式（1画点0为空）

glEnable/glDisable 开启关闭各种参数（功能）GL_LINE_STIPPLE参数表示开启线形

```
// Excer2_ polygon.cpp
#include <GL/glut.h>
#include <stdlib.h>

void display(void)
{
    GLubyte fly[] = {
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x03, 0x80, 0x01, 0xC0, 0x06, 0xC0, 0x03, 0x60,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0x04, 0x06, 0x60, 0x20, 0x44, 0x03, 0xC0, 0x22,
        0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0x66, 0x01, 0x80, 0x66, 0x33, 0x01, 0x80, 0xCC,
        0x19, 0x81, 0x81, 0x98, 0x0C, 0xC1, 0x83, 0x30,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0x06, 0x64, 0x26, 0x60, 0x0c, 0xcc, 0x33, 0x30,
        0x18, 0xcc, 0x33, 0x18, 0x10, 0xc4, 0x23, 0x08,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55};

    GLubyte halftone[] = {
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0x04, 0x60, 0x06, 0x20, 0x04, 0x30, 0x0C, 0x20,
        0x04, 0x18, 0x18, 0x20, 0x04, 0x0C, 0x30, 0x20,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
```

```
        0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
        0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0x07, 0xe1, 0x87, 0xe0, 0x03, 0x3f, 0xfc, 0xc0,
        0x03, 0x31, 0x8c, 0xc0, 0x03, 0x33, 0xcc, 0xc0,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
        0x10, 0x63, 0xC6, 0x08, 0x10, 0x30, 0x0c, 0x08,
        0x10, 0x18, 0x18, 0x08, 0x10, 0x00, 0x00, 0x08 };
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);
    /*  先绘制一个纯色填充的矩形,  */
    /*  然后绘制两个点画矩形       */
    glRectf(25.0, 25.0, 125.0, 125.0);
    glEnable(GL_POLYGON_STIPPLE);
    glPolygonStipple(fly);
    glRectf(125.0, 25.0, 225.0, 125.0);
    glPolygonStipple(halftone);
    glRectf(225.0, 25.0, 325.0, 125.0);
    glDisable(GL_POLYGON_STIPPLE);
    glFlush();
}
void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glShadeModel(GL_FLAT);
}
void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(350, 150);
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
```

```
    glutMainLoop();
    return 0;
}
```

添加代码段含义：

先绘制填充矩形，然后开启多边形绘制，依次设置两种填充图案并绘制填充矩形

函数解释：

glRectf 绘制矩形，参数为矩阵左上和右下坐标

glPolygonStipple 设置填充形状（图案）

glEnable/glDisable 开启关闭各种参数（功能）GL_POLYGON_STIPPLE参数表示开启图案填充

图像填充是基于参数mask的二进制位来决定是否绘制

```
// Excer2_cube.cpp
#include <GL/glut.h>
#include <stdlib.h>

void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glShadeModel(GL_FLAT);
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);
    glLoadIdentity();              /* 设置单位阵 */
    gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); /* 设置视点  */
    glScalef(3, 0.5, 0.5);       /* 设置模型变换矩阵 */
    glutWireCube(1.0);          /* 绘制线框立方体 */
    glFlush();
}

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h); /* 设置视口信息 */
    glMatrixMode(GL_PROJECTION); /* 选择投影矩阵 */
    glLoadIdentity();
    glFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 20.0); /* 设置透视投影变换矩阵 */
```

```
    glMatrixMode(GL_MODELVIEW); /* 选择模型视点矩阵 */
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}
```

添加代码段含义:

display中，设置视点和模型变换矩阵，然后绘制线框立方体

reshape中，glFrustum设置透视投影空间

函数解释:

glViewport 指定视口的大小和位置

glMatrixMode 指定矩阵操作目标，参数GL_PROJECTION选择投影矩阵

glLoadIdentity 重置当前矩阵为单位矩阵

glFrustum 将当前的可视空间设置为透视投影空间

```
// Excer2_solar.cpp
#include <GL/glut.h>
#include <stdlib.h>

int year = 0, day = 0;

void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glShadeModel(GL_FLAT);
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);

    glPushMatrix();
    glutWireSphere(1.0, 20, 16);   /* 绘制太阳 */
```

```
    glRotatef((GLfloat)year, 0.0, 1.0, 0.0);
    glTranslatef(2.0, 0.0, 0.0);
    glRotatef((GLfloat)day, 0.0, 1.0, 0.0);
    glutWireSphere(0.2, 10, 8);    /* 绘制行星 */
    glPopMatrix();
    glutSwapBuffers();
}

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (GLfloat)w / (GLfloat)h, 1.0, 20.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
}

void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 'd':
        day = (day + 10) % 360;
        glutPostRedisplay();
        break;
    case 'D':
        day = (day - 10) % 360;
        glutPostRedisplay();
        break;
    case 'y':
        year = (year + 5) % 360;
        glutPostRedisplay();
        break;
    case 'Y':
        year = (year - 5) % 360;
        glutPostRedisplay();
        break;
    case 27:
        exit(0);
        break;
    default:
        break;
    }
```

```
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
```

添加代码段含义：

display中，将矩阵压栈，然后绘制中心原点的太阳，然后进行旋转（公转角度year）、平移（轨道半径2.0）、旋转（自转角度day），然后绘制地球，最后将矩阵出栈，恢复绘制之前的矩阵

keyboard中，检测按键y/Y/d/D/Esc来对公转自转进行控制以及退出

函数解释：

glutPostRedisplay 标记窗口内容需要重绘，触发display来绘制内容

glRotatef 进行旋转变换

glTranslatef 进行平移变换

glPushMatrix glPopMatrix的作用是将当前矩阵压栈/出栈，出栈后，栈顶矩阵被还原到绘制之前，可以避免当前的绘制操作对后续绘制的影响。

四、 实验任务

任务 1：建立一个绘制点和绘制线的应用源程序，源程序名为 Excer2_drawing.cpp

注意：主要应用的函数为：

glEnable () glDisable () glColor3f(,,)

glLineStippl(,)

在 Excer2_Graphic 中添加 C++ File(.cpp)文件，文件名为 Excer2_drawing.cpp

在 Excer2_drawing.cpp 中添加代码，实现绘制点和绘制线的功能。

任务 2：建立一个绘制点画模式多边形的应用源程序，源程序名为 Excer2_ polygon.cpp

主要应用的函数为：

glEnable () glDisable () glRectf()

glPolygonStipple()

在 Excer2_Graphic 中添加 C++ File(.cpp)文件，文件名为 Excer2_ polygon.cpp

在 Excer2_ polygon.cpp 中添加代码，实现绘制绘制点画模式多边形的功能。

任务 3：建立一个视角变换的立方体的应用源程序，源程序名为 Excer2_cube.cpp

主要应用的函数为：

gluLookAt() ， glScalef()， glutWireCube() glFrustum() glMatrixMode()

在 Excer2_Graphic 中添加 C++ File(.cpp)文件，文件名为 Excer2_ cube.cpp

在 Excer2_ cube.cpp 中添加代码，实现绘制一个立方体，并能实行视角变换功能。

任务 4：建立一个地球自转并绕太阳公转的应用源程序，源程序名为 Excer2_ solar.cpp

主要应用的函数为：

glPushMatrix(); glPopMatrix(); glutPostRedisplay(); glRotatef(); glTranslatef(); glutWireSphere()

在 Excer2_Graphic 中添加 C++ File(.cpp)文件，文件名为 Excer2_ solar.cpp

在 Excer2_ solar.cpp 中添加代码，绘制两个球体分别为太阳和地球，并能实行地球的自转和公转，地球和太阳大小不一致，位置不一致。

五、 主要总结

运行结果：

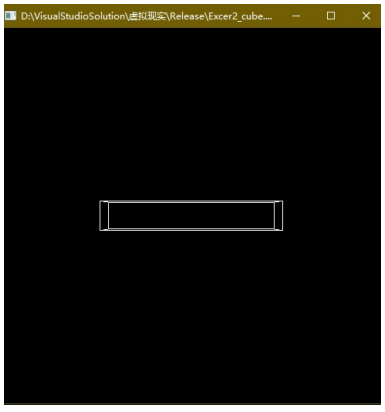
任务 1 如图为绘制 点、 线段、点-线 样式线条的结果；

glLineStipple 可以通过设置参数 pattern 来实现点线样式，如 0B0000000100000001 即为点样式。

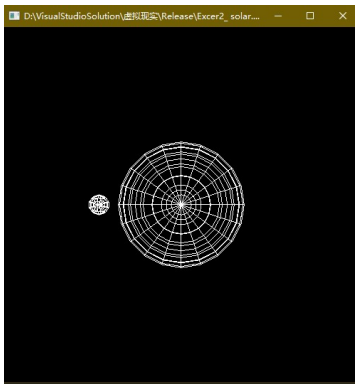
任务 2 如图为图案绘制的结果；

glPolygonStipple 的 mask 参数决定了填充多边形使用的图案；改变图案效果绘制结果。

任务 3 如图为立方体在视口中被拉伸的效果。



任务 4 如图为太阳地球绘制的效果



总结：

此次实验，涉及了图形的绘制以及视口的投影转换。主要通过 glLineStipple、glPolygonStipple 来修改线面填充样式、glFrustum 设置投影变换等，太阳地球旋转的例子，通过变换坐标系平移或旋转来绘制地球的公转/自转，可以熟知 glRotatef、glTranslatef 的用法含义。

教师评阅：

评阅项目及内容	得分
1．考勤（10 分）	
2．实验完成情况（50 分）	
3．报告撰写内容（40 分）	
合　计	
成绩评定	