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 基于OpenGL的动画编程

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 2020/6/4

实验地点

1. 实验主题

利用VC集成开发环境，实现利用OpenGL编写利用Glut工具库中的函数进行鼠标和键盘的交互。

1. 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。

1. 主要数据源、库函数、变量、涉及函数及其解释

|  |
| --- |
| // mouse.cpp  #include <GL/glut.h>  #include <stdlib.h>  float x1 = -0.25, y1 = -0.25;//  float width = 0.5, height = 0.5;  float xMin = -1, xMax = 1;  float yMin = -1, yMax = 1;  float widthWindows = 400, heightWindows = 400;  float colorR = 0.5, colorG = 0.5, colorB = 0.5;  void display(void)  {  glClearColor(0.0f, 0.0f, 0.0f, 0.0f);  glClearDepth(1.0);  glClear(GL\_COLOR\_BUFFER\_BIT);  //绘制一个矩形，z轴位置为-1.0f  glColor3f(colorR, colorG, colorB);  glRectf(x1, y1, x1 + width, y1 + height);  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 \*/  glOrtho(xMin, xMax, yMin, yMax, 0.0, 10.0);  }  void MyMouse(int button, int state, int x, int y)  {  if (state == GLUT\_DOWN)  {  switch (button)  {  case GLUT\_LEFT\_BUTTON:  colorR += 0.1;  if (colorR > 1.0)  colorR = 0.0;  glutPostRedisplay();  break;  case GLUT\_MIDDLE\_BUTTON:  colorG += 0.1;  if (colorG > 1.0)  colorG = 0.0;  glutPostRedisplay();  break;  case GLUT\_RIGHT\_BUTTON:  colorB += 0.1;  if (colorB > 1.0)  colorB = 0.0;  glutPostRedisplay();  break;  }  }  }  void MyMotion(int x, int y)  {  x1 = (float)x / widthWindows \* (xMax - xMin) + xMin - width / 2;  y1 = (float)(heightWindows - y) / heightWindows \* (yMax - yMin) + yMin - width / 2;  glutPostRedisplay();  }  void MyKeyboard(unsigned char key, int x, int y)  {  switch (key)  {  case 'W':  case 'w':// 矩形坐标变量修改使得矩形上移  y1 += 0.1;  if (y1 >= yMax - height)  y1 = yMax - height;  glutPostRedisplay();  break;  case 'S':  case 's': // 矩形坐标变量修改使得矩形下移  y1 -= 0.1;  if (y1 <= yMin)  y1 = yMin;  glutPostRedisplay();  break;  case 'A':  case 'a': // 矩形坐标变量修改使得矩形左移  x1 -= 0.1;  if (x1 <= xMin)  x1 = xMin;  glutPostRedisplay();  break;  case 'D':  case 'd': // 矩形坐标变量修改使得矩形右移  x1 += 0.1;  if (x1 >= xMax - width)  x1 = xMax - width;  glutPostRedisplay();  break;  }  }  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_SINGLE);  glutInitWindowSize(400, 400);  glutInitWindowPosition(100, 100);  glutCreateWindow(argv[0]);  glutDisplayFunc(display);  glutReshapeFunc(reshape);  glutKeyboardFunc(MyKeyboard);  glutMouseFunc(MyMouse);  glutMotionFunc(MyMotion);  glutMainLoop();  return 0;  } |
| 函数解释：  glutKeyboardFunc 设置键盘普通案件响应函数；  glutMouseFunc 设置鼠标按下响应函数；  glutMotionFunc 设置鼠标按下拖拽响应函数。 |
| // double.cpp  #include <GL/glut.h>  #include <stdlib.h>  static GLfloat spin = 0.0;  void display(void)  {  glClear(GL\_COLOR\_BUFFER\_BIT);  glPushMatrix();  glRotatef(spin, 0.0, 0.0, 1.0);  glColor3f(1.0, 1.0, 1.0);  glRectf(-25.0, -25.0, 25.0, 25.0);  glPopMatrix();  glutSwapBuffers();  }  void spinDisplay(void)  {  spin = spin + 2.0;  if (spin > 360.0)  spin = spin - 360.0;  glutPostRedisplay();  }  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();  glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);  glMatrixMode(GL\_MODELVIEW);  glLoadIdentity();  }  void mouse(int button, int state, int x, int y)  {  switch (button) {  case GLUT\_LEFT\_BUTTON:  if (state == GLUT\_DOWN)  glutIdleFunc(spinDisplay);  break;  case GLUT\_MIDDLE\_BUTTON:  case GLUT\_RIGHT\_BUTTON:  if (state == GLUT\_DOWN)  glutIdleFunc(NULL);  break;  default:  break;  }  }  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(250, 250);  glutInitWindowPosition(100, 100);  glutCreateWindow(argv[0]);  init();  glutDisplayFunc(display);  glutReshapeFunc(reshape);  glutMouseFunc(mouse);  glutMainLoop();  return 0;  } |
| 函数解释：  glutMouseFunc 设置鼠标按下响应函数； |
| // bounce.cpp  #include <GL/glut.h>  // Initial square position and size  GLfloat x = 0.0f;  GLfloat y = 0.0f;  GLfloat rsize = 25;  // Step size in x and y directions  // (number of pixels to move each time)  GLfloat xstep = 1.0f;  GLfloat ystep = 1.0f;  // Keep track of windows changing width and height  GLfloat windowWidth;  GLfloat windowHeight;  ///////////////////////////////////////////////////////////  // Called to draw scene  void RenderScene(void)  {  // Clear the window with current clearing color  glClear(GL\_COLOR\_BUFFER\_BIT);  // Set current drawing color to red  // R G B  glColor3f(1.0f, 0.0f, 0.0f);  // Draw a filled rectangle with current color  glRectf(x, y, x + rsize, y - rsize);  // Flush drawing commands and swap  glutSwapBuffers();  }  ///////////////////////////////////////////////////////////  // Called by GLUT library when idle (window not being  // resized or moved)  void TimerFunction(int value)  {  // Reverse direction when you reach left or right edge  if (x < - windowWidth || x > windowWidth - rsize)  xstep \*= -1;  if (y < rsize - windowHeight || y > windowHeight)  ystep \*= -1;  // Actually move the square  x += xstep;  y += ystep;  // Check bounds. This is in case the window is made  // smaller while the rectangle is bouncing and the  // rectangle suddenly finds itself outside the new  // clipping volume  if (x > (windowWidth - rsize + xstep))  x = windowWidth - rsize - 1;  else if (x < -(windowWidth + xstep))  x = -windowWidth - 1;  if (y > (windowHeight + ystep))  y = windowHeight - 1;  else if (y < -(windowHeight - rsize + ystep))  y = -windowHeight + rsize - 1;  // Redraw the scene with new coordinates  glutPostRedisplay();  glutTimerFunc(33, TimerFunction, 1);  }  ///////////////////////////////////////////////////////////  // Setup the rendering state  void SetupRC(void)  {  // Set clear color to blue  glClearColor(0.0f, 0.0f, 1.0f, 1.0f);  }  ///////////////////////////////////////////////////////////  // Called by GLUT library when the window has chanaged size  void ChangeSize(int w, int h)  {  GLfloat aspectRatio;  // Prevent a divide by zero  if (h == 0)  h = 1;  // Set Viewport to window dimensions  glViewport(0, 0, w, h);  // Reset coordinate system  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  // Establish clipping volume (left, right, bottom, top, near, far)  aspectRatio = (GLfloat)w / (GLfloat)h;  if (w <= h)  {  windowWidth = 100;  windowHeight = 100 / aspectRatio;  glOrtho(-100.0, 100.0, -windowHeight, windowHeight, 1.0, -1.0);  }  else  {  windowWidth = 100 \* aspectRatio;  windowHeight = 100;  glOrtho(-windowWidth, windowWidth, -100.0, 100.0, 1.0, -1.0);  }  glMatrixMode(GL\_MODELVIEW);  glLoadIdentity();  }  ///////////////////////////////////////////////////////////  // Main program entry point  int main(int argc, char\* argv[])  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 600);  glutCreateWindow("Bounce");  glutDisplayFunc(RenderScene);  glutReshapeFunc(ChangeSize);  glutTimerFunc(33, TimerFunction, 1);  SetupRC();  glutMainLoop();  return 0;  } |
| 添加代码：  if (x < - windowWidth || x > windowWidth - rsize)  xstep \*= -1;  if (y < rsize - windowHeight || y > windowHeight)  ystep \*= -1;  可以让矩形在触碰到边界时将速度反向。  函数解释：  glutTimerFunc 设置定时器回调函数，参数为 定时时间（毫秒）、回调函数、定时器的值（编号）。 |

1. 实验任务

任务1：使用鼠标键盘交互的应用源程序，源程序名Excer5\_mouse.cpp

 在Excer5\_Cartoon中添加C++ File(.cpp)文件，文件名为Excer5\_mouse.cpp

 在Excer5\_mouse.cpp中添加代码，实现鼠标键盘操作实现对绘制矩形框交互的功能。

任务2：绘制一个正方形，使用双缓存绘制旋转正方形，源程序名为Excer5\_ double.cpp

 在Excer5\_Cartoon中添加C++ File(.cpp)文件，文件名为Excer5\_ double.cpp

 在Excer5\_ double.cpp中添加代码。

任务3：建立一个反弹方块动画的应用源程序，源程序名为Excer5\_Bounce.cpp

主要应用的函数为：

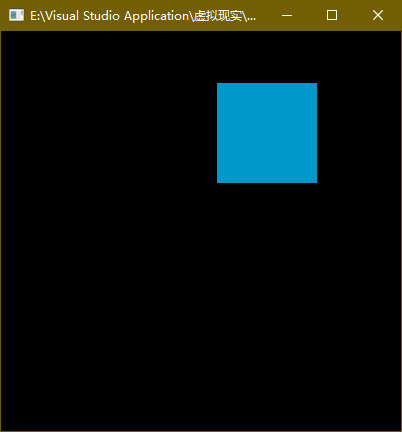
 在Excer5\_Cartoon中添加C++ File(.cpp)文件，文件名为Excer5\_Bounce.cpp

 在Excer5\_Bounce.cpp中添加代码， 实现方块遇到窗口边界反弹的效果。

1. 主要分析和解释

运行结果：

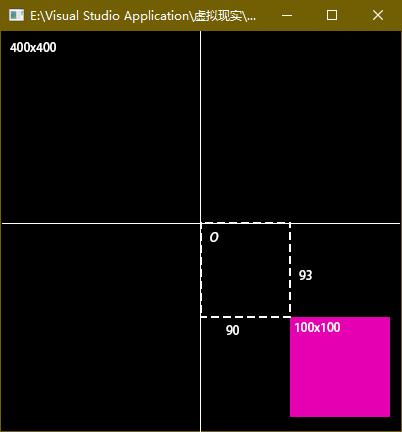
任务1 如图为鼠标拖动动画；



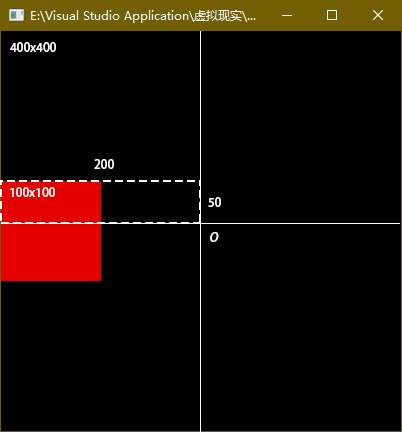
 这段程序体使用了什么交互函数？达到的功能是什么？

使用MyMouse来响应鼠标按下操作，实现方块颜色的改变，MyMotion响应按下操作，实现方块的拖动，MyKeyboard响应一般按键消息，实现键盘移动方块。

 运行程序后，利用图的形式画出MyMotion()和MyKeyboard()的作用效果（画出原点，移动距离，变换关系，画出坐标系，标出窗口大小，矩形大小，移动距离等等参数），分析轨迹的形式记录所有细节结果，并基于它们做一个简单的解释。

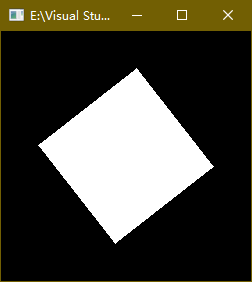


如图为鼠标拖动方块示意图，可见其坐标不是50的倍数。



如图为键盘移动方块示意图，可见其坐标刚好是50的倍数。

任务2 如图为旋转方块动画；



 glutIdleFunc ()作用是什么;在什么情况下经常使用？

设定程序空闲时的回调函数，一般用于执行后台处理任务或者连续动画

 SpinDisplay()的作用是什么，如果正方形换一个方向旋转该如何修改代码？

变换正方形角度，并指示界面重绘。

spin = spin - 2.0;

if (spin < 0)

spin = spin + 360.0;

 运行程序，程序在mouse（）函数里的 case GLUT\_MIDDLE\_BUTTON: case GLUT\_RIGHT\_BUTTON:情况下，执行时会使正方形的运动发生什么变化，归纳下glutIdleFunc(NULL)中参数NULL的含义和作用？

按下鼠标中键或右键会让方块停止旋转。

向glutIdleFunc传入NULL会取消设定的空闲执行函数。

 修改mouse（）函数，使得按下鼠标右键时反向旋转，能够加快旋转频率。

case GLUT\_RIGHT\_BUTTON:

if (state == GLUT\_DOWN)

{

speed = -3;

glutIdleFunc(spinDisplay);

}

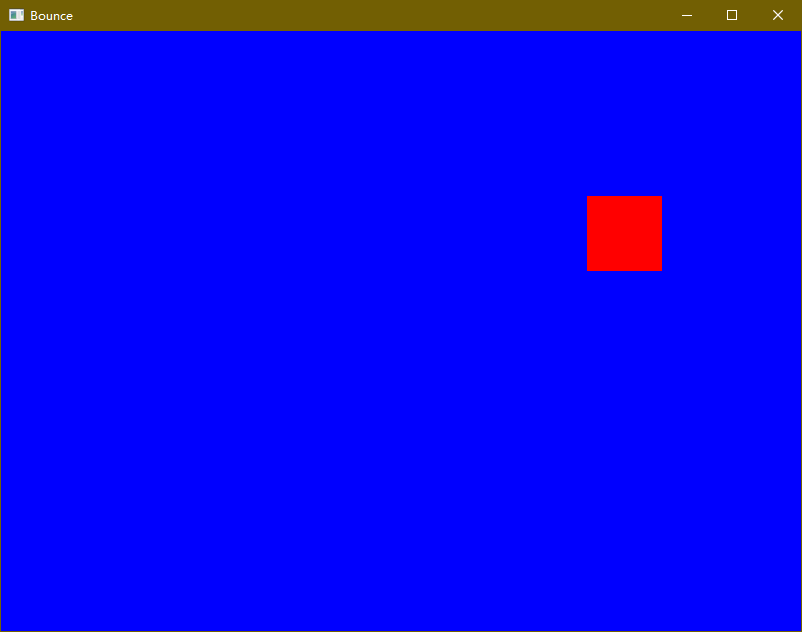
break;

 启动双缓存是使用哪个函数？观察并设定启动单缓存，程序会有什么变化？

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);参数GLUT\_DOUBLE

单缓存下，屏幕会一片白

任务3 如图为方块反弹动画。



 在黄色高亮处添加相应的代码，请解释添加代码段的含义，为什么如此添加？

if (x < -windowWidth || x > windowWidth - rsize)

xstep \*= -1;

if (y < rsize - windowHeight || y > windowHeight)

ystep \*= -1;

判断出界则将速度反向

 请解释glutTimerFunc(33, TimerFunction, 1)此函数的作用和参数的作用是什么？如果使33的值变大或变小，对动画效果有啥影响?为什么会发生这样的影响？

开启定时器；

会影响动画速度；

33的值是定时器的延时。

1. 展望

本次实验， 主要实践了OpenGL的鼠标和键盘交互，以及简单动画的编写，让我认识到OpenGL在用户交互上的流程，以及各种回调函数的设定方法与使用方法。

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 |  |