**计算机图形学**

**实验报告**

**exp1**

**Task2**

学习红宝书思考：

（1）回调函数是什么意思？GLUT中有哪些回调函数？

回调函数就是一个通过函数指针调用的函数。回调函数不是由该函数的实现方直接调用，而是在特定的事件或条件发生时由另外的一方调用的，用于对该事件或条件进行响应。

GLUT中的回调函数

void glutReshapeFunc(void (\*func)(int width, int height));

指定当窗口的大小改变时调用的函数

void glutKeyboardFunc(void (\*func)(unsigned char key, int x, int y));

注册当前窗口的键盘回调函数

void glutMouseFunc(void (\*func)(int button, int state, int x, int y));

注册当前窗口的鼠标回调函数

void glutMotionFunc(void (\*func)(int x, int y));

当鼠标在窗口中按下并移动时调用glutMotionFunc注册的回调函数

void glutPassiveMotionFunc(void (\*func)(int x, int y));

当鼠标在窗口中移动时调用glutPassiveMotionFunc注册的回调函数

void glutEntryFunc(void (\*func)(int state));

设置鼠标的进出窗口的回调函数

void glutVisibilityFunc(void (\*func)(int state));

设置当前窗口的可视回调函数

void glutIdleFunc(void (\*func)(void));

设置空闲回调函数

void glutTimerFunc(unsigned int millis, void (\*func)(int value), int value);

注册一个回调函数,当指定时间值到达后,由GLUT调用注册的函数一次

void glutMenuStateFunc(void (\*func)(int state));

注册菜单状态回调函数

void glutMenuStatusFunc(void (\*func)(int status, int x, int y));

设置菜单状态回调函数

void glutSpecialFunc(void (\*func)(int key, int x, int y));

设置当前窗口的特定键的回调函数

void glutSpaceballMotionFunc(void (\*func)(int x, int y, int z));

注册一个当前窗口的spaceball平移的回调函数

void glutSpaceballRotateFunc(void (\*func)(int x, int y, int z));

注册一个当前窗口的spaceball转动的回调函数

void glutSpaceballButtonFunc(void (\*func)(int button, int state));

注册当前窗口的spaceball的按键回调函数

void glutButtonBoxFunc(void (\*func)(int button, int state));

注册当前窗口的拨号按键盒按键回调函数

void glutDialsFunc(void (\*func)(int dial, int value));

注册当前窗口的拨号按键盒拨号回调函数

void glutTabletMotionFunc(void (\*func)(int x, int y));

注册图形板移动回调函数

void glutTabletButtonFunc(void (\*func)(int button, int state, int x, int y));

注册当前窗口的图形板按键回调函数

void glutOverlayDisplayFunc(void (\*func)(void));

注册当前窗口的重叠层的显示回调函数

void glutWindowStatusFunc(void (\*func)(int state));

注册当前窗口状态的回调函数

void glutKeyboardUpFunc(void (\*func)(unsigned char key, int x, int y));

注册释放普通按键的回调函数

void glutSpecialUpFunc(void (\*func)(int key, int x, int y));

注册释放特殊按键的回调函数

void glutJoystickFunc(void (\*func)(unsigned int buttonMask, int x, int y, int z), int pollInterval);

注册操纵杆的回调函数

（2）用鼠标改变窗口大小会发生什么？哪个函数在影响整个过程？

改变窗口大小后，界面上的图形大小会随之改变

由glutReshapeFunc(reshape);调用的void reshape (int w, int h);在影响整个过程

（3）试着画画其他图形？

Task中实现了画圆与椭圆

**Task3**

源代码：

/\*

\* 小球落地

\* 使用二维圆形实现

\* 实现了上升与下降过程中速度的变化以及落地时的形变（不明显）

\*\*\*\*\*\*\*

\* @author：宋灵冰

\*/

#include <math.h>

#include <gl/glut.h>

// 指定圆边缘精细程度

const int n = 100;

// 指定圆的半径

const GLfloat R = 0.15;

// 指定PI值

const GLfloat Pi = 3.1415926536;

// 指定运算精度

const GLfloat accuracy = 1e-7;

// 指定地面位置

const GLfloat minHeight = R - 1.0;

// 指定上升最大高度

GLfloat maxHeight = 1.0 - R;

// 指定圆心高度

GLfloat yPos = 1.0 - R;

// 指定下降速度

GLfloat gap = 0;

// 指定运动的加速度

const GLfloat acceleration = 1e-6;

// 标记运动状态（下降/上升）

bool isDown = true;

// 显示原始小球

void display() {

// 清除颜色缓冲

glClear(GL\_COLOR\_BUFFER\_BIT);

// 画小球

glBegin(GL\_POLYGON);

for (int i = 0; i<n; ++i)

glVertex2f(R\*cos(2 \* Pi / n \* i), R\*sin(2 \* Pi / n \* i) + yPos);

glEnd();

glFlush();

// 交换两个缓冲区指针

glutSwapBuffers();

}

// 指定小球落地时的形变

// a、b分别为椭圆的参数

void display(float a, float b) {

// 清除颜色缓冲

glClear(GL\_COLOR\_BUFFER\_BIT);

// 画椭圆

glBegin(GL\_POLYGON);

for (int i = 0; i<n; ++i)

glVertex2f(a \* cos(2 \* Pi / n \* i), b \* sin(2 \* Pi / n \* i) + yPos);

glEnd();

glFlush();

// 交换两个缓冲区指针

glutSwapBuffers();

}

// 指定小球运动的函数

void idle() {

// 下降状态，速度改变

if (yPos > minHeight && isDown) {

yPos -= gap;

gap += acceleration;

display();

}

// 上升状态，速度改变

else if (yPos < maxHeight && !isDown) {

yPos += gap;

if (gap > accuracy)

gap -= acceleration;

// 若速度降为0，

// 则将当前高度设置为最高点

else {

// 改变运动状态

isDown = true;

maxHeight = yPos;

}

display();

}

// 接触地面

else {

// 根据最高位置占总高度的比例计算形变程度

float scale = (maxHeight + 1) / 2 \* R;

// 若满足静止条件，则将高度置0

if (maxHeight < minHeight + accuracy) {

maxHeight = -1;

}

// 开始挤压形变

else if (yPos > minHeight - scale && isDown) {

yPos -= gap;

gap += acceleration;

display(R, 1 + yPos);

}

// 开始恢复原状

else if (yPos < minHeight && !isDown) {

yPos += gap;

gap -= acceleration;

display(R, 1 + yPos);

}

else {

// 指定撞击地面后反弹回来速度改变

gap \*= 0.85;

// 改变运动状态

isDown = false;

}

}

}

// 设置键盘退出指令函数

void keyboard(unsigned char key, int x, int y)

{

switch (key) {

case 27:

exit(0);

break;

}

}

int main(int argc, char\*\* argv) {

glutInit(&argc, argv);

// 设置双缓冲、RGB模式

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);

glutInitWindowSize(800, 800);

glutInitWindowPosition(0, 0);

glutCreateWindow("Ball");

// 注册一个绘图函数

glutDisplayFunc(display);

// 为键盘消息注册回调

glutKeyboardFunc(keyboard);

// 设置全局的回调函数

glutIdleFunc(idle);

// 进入GLUT事件处理循环

glutMainLoop();

return 0;

}