**《计算机图形学》课程报告**

计算机学院

**班 级： 194201**

**学 号： 20201003943**

**姓 名： 戴子权**

**指导教师： 武云**

**完成日期： 2022年5月22日**

**实验题目：（二维时钟建立）**

1. **实验目的与要求（二维时钟的建立）**

该实验通过绘制点函数glVertex2i通过数学的方法实现时钟基本图形框架的绘制，包括绘制时钟的外框（圆形）以及时钟的指针（时针，分针，秒针）。通过变量degree控制时钟的移动角度。该时钟运行方式有两种，一种是通过openGL自带鼠标事件函数GL\_LEFT\_BUTTON,GLUT\_DOWN来监控鼠标的移动以及按键操作，以实现通过鼠标对时钟指针进行操控，另外一种是通过openGL自带的函数glutTimerFunc实现时钟从零点开始进行自转。要求是可以在任意时刻实现时钟在自转和通过鼠标控制的旋转之间进行转换。

**二、实验内容**

该实验代码共分为四个部分，第一部分为时钟结构各部分变量的定义部分，第二部分为时钟图形显示部分（display函数），第三部分为main函数，第四部分为控制时钟指针动作部分（定义指针运动的函数），下面将依次介绍完成以上四个部分的方法以及流程。

一-时钟结构各部分变量的定义

通过对该部分的介绍将了解到时钟各部分的组成成分，是了解该实验工作的基础。

首先，定义时钟的各个指针，共有三个指针（时针，分针，秒针），将这三个指针通过define定义为handtype类型的变量，并为其赋值，时针赋值为0，分针赋值为1，秒针赋值为2，以方便后期对这三个指针动作的操作，然后为这三个指针的长度以及宽度进行赋值，最后创建三个变量，用于记录指针移动的角度。至此时钟指针部分外形定义完成。

其次定义时钟运动部分变量，由于该时钟具有两种运转模式（手动/自动）所以需要一个bool类型的变量来判断时钟是否开始运转，在自动模式中时钟开始运转，在手动模式中时钟停止运转。

然后定义时钟的边框，由于在该实验中并未直接使用画圆函数（考虑到时钟的细节部分，直接使用画圆函数不方便进行处理）所以需要定义pi=3.1415926535，然后定义时钟的半径CLOCKRADIUS。

最后使用两个变量记录秒针当前位置（由于在手动模式中只会转动秒针，所以只需要记录秒针的位置），共分为四个象限，若在一，二象限秒针的下一位置的y坐标应该小于当前位置的y坐标，若在三，四象限，秒针下一位置的y坐标应大于当前位置的y坐标。

二-时钟图形显示部分（display函数）

该部分用于将时钟的动作以及时钟图形glMatrixMode(GL\_PROJECTION输出到用户界面。

首先为了保证在用户界面放大/缩小时时钟图形不会出现拉伸等形变，需要使用openGL自带函数glMatrixMode(GL\_PROJECTION)，glLoadIdentity();进行投影操作，定义窗口函数gluOrtho2D(0, winwidth, 0, winheight);以及视区函数glViewport(0, 0, winwidth, winheight);，使用以上函数后将使视区投影到用户界面窗口中，因此视区不会因窗口的改变而改变其在窗口的相对位置。

然后定义绘图所使用的颜色，最后将自己定义的画园函数，画指针函数依次加入Display函数中实现将时钟的图形以及时钟的动作输出到窗口界面。

三-main函数部分

该部分将定义窗口的位置以及大小，以及使用鼠标动作，使用图形显示（display函数）等功能。本报告将略去定义窗口部分。

首先使用glutDisplayFunc（）函数用于将第二部分的输出内容输出到用户界面中，

然后使用glutMouseFunc()函数监控鼠标事件，其次使用glutMotionFunc()函数完成鼠标动作，最后使用glutReshapeFunc()函数提供当窗口大小改变时回调函数的接口。

四-控制时钟指针动作部分（定义指针运动的函数）

该部分为本项目的重点，分为void Timer（int value）函数，void Mouse（int button，void int state，int x，int y）函数，void Motion（int x，int y）函数，void DrawCircle(int n, int x, int y)函数以及void DrawHand(handtype type, int x, int y)函数。

void Timer（int value）函数是一个定时器函数，当时钟切换到自动运行模式时将调用该函数，该函数是一个递归函数，首先定义了每次递归各个指针需要进行的操作（定义角度变量的值），每一次递归都将使用glutPostRedisplay()函数用于时钟的重新绘制。

void Mouse（int button，void int state，int x，int y）函数用于监控鼠标的动作，当鼠标点击右键时时钟切换到自动模式，调用timer函数，时钟将自动运行。当鼠标点击左键时时钟停止自动模式（停止运行clockloop=false）,并置leftmousedown的值为true。变量leftmousedown记录当时钟为手动模式时鼠标进行的操作（上移或下移），若上移将置变量leftmousedown的值为true，否则置为false，该部分服务于motion函数。

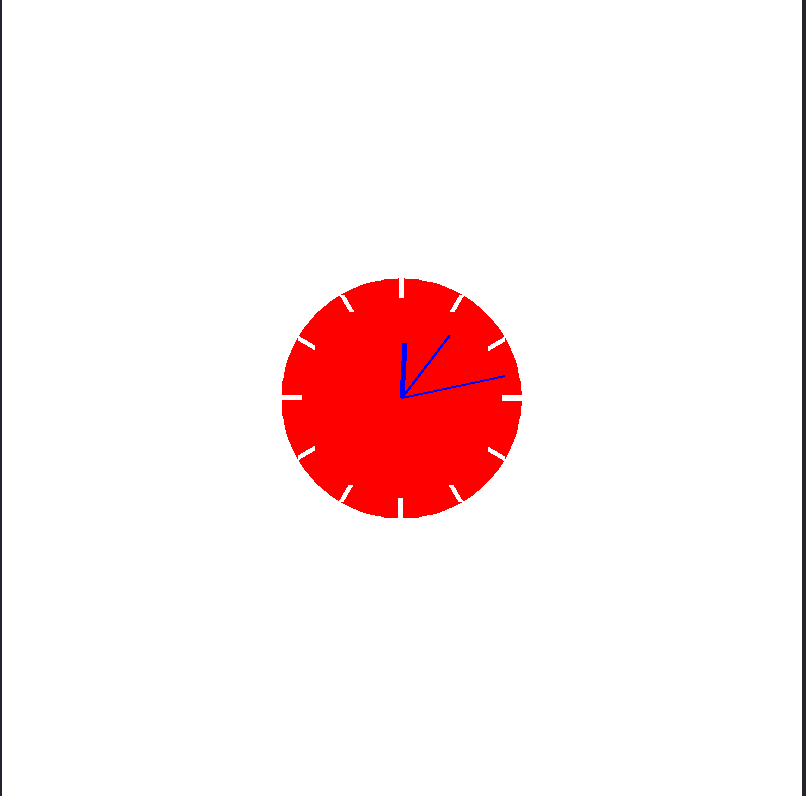
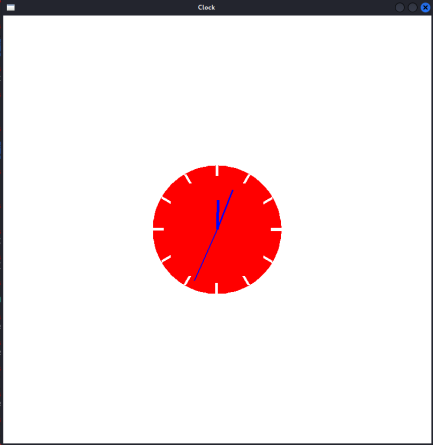
void Motion（int x，int y）函数用于实现当点击鼠标左键（时钟手动运行时的操作）。

当鼠标向上移动时时钟将进行顺时针运动，反之则进行逆时针运动，每当鼠标移动一个像素，时钟图形会进行相应的刷新（指针的角度会实时变化）。

void DrawCircle(int n, int x, int y)函数用于绘制时钟边框的图形，由于时钟边框具有刻度，因此将时钟边框分为三种类型，第一种类型为时钟的最外圈，该圈圆的半径最大，且有空白刻度。第二种类型为时钟的刻度圈，该圈半径第二大，且有空白刻度。第三种类型为时钟的内圈，该圈半径最小且无刻度。

void DrawHand(handtype type, int x, int y)函数用于实现根据时钟各个指针对应的角度对时钟指针进行移动，由于三个指针的旋转操作是相同的，在此只解说时针的旋转操作。首先使用glRotatef函数将时针置于指定角度位置，然后使用glLineWidth函数根据指定的宽度以及glBegin(GL\_LINES)函数以及glVertex2i函数根据指定的指针长度绘制指针。

**三、实验结果**



**由于实验结果是一个时钟，是由鼠标进行时事操作的所以无法直观的显示操作**

**四、体会**

通过本实验我有以下体会

1.对openGL的main函数有了更加深刻的理解。在以前的认识中我一直将main函数作为一个固定的模板使用，每次都会直接复制粘贴进行使用，通过本次实验我对main函数的结构有了更加清晰的认识，并且可以更加灵活的使用main函数。

2.处理问题的方式更加灵活。在处理时钟指针的移动时，我开始希望通过记录鼠标移动的距离来判断指针旋转的角度，但是我发现这种方式过于繁琐，且不能实时根据鼠标的移动改变指针的位置，于是我想到了一个更好的办法，可以让鼠标每移动一个像素，指针都会移动一个固定的角度，通过这样的方式不仅大大减小了操作难度，且能够实时根据鼠标的移动旋转指针。

**五、源程序**

源程序，为了减少篇幅，请删除源程序中你认为不重要的语句以及空行，但不要把所有的注释都删除了。源程序采用**小五号Times New Romar字体**，**行间距调为单倍行距**。

#include <gl/glut.h>

#include <cmath>

#define HOURHAND (handtype)0

#define MINUTEHAND (handtype)1

#define SECONDHAND (handtype)2

typedef unsigned handtype;

const double Pi = 3.1415926535;

bool leftmousedown = false;

bool clockloop = false;

int winwidth = 800;

int winheight = 800;

const int CLOCKRADIUS = 120;

const int HOURWIDTH = 5;

const int HOURLENGTH = 55;

const int MINUTEWIDTH = 3;

const int MINUTELENGTH = 80;

const int SECONDWIDTH = 2;

const int SECONDLENGTH = 105;

double hourdegree = 0.0;

double minutedegree = 0.0;

double seconddegree = 0.0;

int xlast = 0, ylast = 0;

void Timer(int);

void Mouse(int, int, int, int);

void Motion(int, int y);

void DrawCircle(int, int x = round(winwidth / 2), int y = round(winheight / 2));

void DrawHand(handtype, int x = round(winwidth / 2), int y = round(winheight / 2));

void Display();

void ChangeSize(int, int);

void Display() {

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, winwidth, 0, winheight);

glViewport(0, 0, winwidth, winheight);

glClearColor(1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

DrawCircle(100);

DrawHand(HOURHAND);

DrawHand(MINUTEHAND);

DrawHand(SECONDHAND);

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(100, 100);

glutInitWindowSize(winwidth, winheight);

glutCreateWindow("Clock");

glutDisplayFunc(Display);

glutMouseFunc(Mouse);

glutMotionFunc(Motion);

glutReshapeFunc(ChangeSize);

glutMainLoop();

return 0;

}

void Timer(int value) {

if (value == 0) {

minutedegree += 0.1;

seconddegree += 6;

hourdegree += (double)1 / 120;

glutPostRedisplay();

}

if (clockloop) {

glutTimerFunc(1000, Timer, 0);

}

}

void Mouse(int button, int state, int x, int y) {

if (button == GLUT\_LEFT\_BUTTON) {

if (state == GLUT\_DOWN) {

leftmousedown = true;

clockloop = false;

}

else

leftmousedown = false;

}

if (button == GLUT\_RIGHT\_BUTTON) {

if (state == GLUT\_DOWN) {

if (!clockloop) {

clockloop = true;

glutTimerFunc(1000, Timer, 0);

}

}

else

rightmousedown = false;

}

}

void Motion(int x, int y) {

if (leftmousedown) {

if (y > ylast) {

minutedegree += 0.1;

seconddegree += 6;

hourdegree += (double)1 / 120;

glutPostRedisplay();

xlast = x;

ylast = y;

}

else {

minutedegree -= 0.1;

seconddegree -= 6;

hourdegree -= (double)1 / 120;

glutPostRedisplay();

xlast = x;

ylast = y;

}

}

}

void DrawCircle(int n, int x, int y) {

glColor4f(1.0, 0.0, 0.0, 1.0);

glBegin(GL\_POLYGON);

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

glVertex2i(CLOCKRADIUS \* cos(2 \* Pi / n \* i) + x, CLOCKRADIUS \* sin(2 \* Pi / n \* i) + y);

glEnd();

glColor4f(1.0, 1.0, 1.0, 1.0);

glLineWidth(3);

glBegin(GL\_LINES);

int m = 60;

for (int i = 0; i < 60; i++) {

if (m % 12 == 0)

continue;

glVertex2i((CLOCKRADIUS - 10) \* cos(2 \* Pi / m \* i) + x, (CLOCKRADIUS - 10) \* sin(2 \* Pi / m \* i) + y);

glVertex2i(CLOCKRADIUS \* cos(2 \* Pi / m \* i) + x, CLOCKRADIUS \* sin(2 \* Pi / m \* i) + y);

}

glEnd();

glLineWidth(5);

m = 12;

glBegin(GL\_LINES);

for (int i = 0; i < 60; i++) {

glVertex2i((CLOCKRADIUS - 20) \* cos(2 \* Pi / m \* i) + x, (CLOCKRADIUS - 20) \* sin(2 \* Pi / m \* i) + y);

glVertex2i(CLOCKRADIUS \* cos(2 \* Pi / m \* i) + x, CLOCKRADIUS \* sin(2 \* Pi / m \* i) + y);

}

glEnd();

}

void DrawHand(handtype type, int x, int y) {

glLoadIdentity();

gluOrtho2D(0, winwidth, 0, winheight);

glViewport(0, 0, winwidth, winheight);

glPushMatrix();

glTranslatef(x, y, 0);

glPushMatrix();

switch (type)

{

case HOURHAND:

glRotatef(hourdegree, 0, 0, -1);

glColor4f(0.0, 0.0, 1.0, 1.0);

glLineWidth(HOURWIDTH);

glBegin(GL\_LINES);

glVertex2i(0, 0);

glVertex2i(0, 0 + HOURLENGTH);

glEnd();

break;

case MINUTEHAND:

glRotatef(minutedegree, 0, 0, -1);

glColor4f(0.0, 0.0, 1.0, 1.0);

glLineWidth(MINUTEWIDTH);

glBegin(GL\_LINES);

glVertex2i(0, 0);

glVertex2i(0, 0 + MINUTELENGTH);

glEnd();

break;

case SECONDHAND:

glRotatef(seconddegree, 0, 0, -1);

glColor4f(0.0, 0.0, 1.0, 1.0);

glLineWidth(SECONDWIDTH);

glBegin(GL\_LINES);

glVertex2i(0, 0);

glVertex2i(0, 0 + SECONDLENGTH);

glEnd();

break;

default:

break;

}

glPopMatrix();

glPopMatrix();

}

void ChangeSize(int w, int h) {

winwidth = w;

winheight = h;

glViewport(0, 0, w, h);

}

课程报告需要上交电子版和打印版本，格式和内容见上面，上交电子版需实验源码及课程报告电子版。电子版由每班班干部统一收齐，一个同学一个文件夹，发到我的qq邮箱。