Steven Kong

CSC 470 Computer Graphics

Lab 1

//Steven Kong

//CSC 470 Computer Graphics

//Lab 1 Problem 1

#include <math.h>

#include <GL/glut.h>

#include <fstream>

using namespace std;

const int screenWidth = 640; // width of screen window in pixels

const int screenHeight = 480; // height of screen window in pixels

GLdouble A, B, C, D; // values used for scaling and shifting

void setWindow(float left, float right, float bottom, float top) {

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(left, right, bottom, top);

}

void setViewport(float left, float right, float bottom, float top) {

glViewport(left, bottom, right - left, top - bottom);

}

void drawPolyLineFile(char \* fileName) {

fstream inStream;

inStream.open(fileName, ios::in); // open the "Brontoi.dat"

if (inStream.fail())

return;

//glClear(GL\_COLOR\_BUFFER\_BIT); // clear the screen

GLint numpolys, numLines, x, y;

inStream >> numpolys; // read the number of polylines

for (int j = 0; j < numpolys; j++) { // read each polyline

inStream >> numLines;

glBegin(GL\_LINE\_STRIP); // draw the next polyline

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

inStream >> x >> y; // read the next x, y pair

glVertex2i(x, y);

}

glEnd();

}

glFlush();

inStream.close();

}

void draw(GLfloat x, GLfloat y, int tilt, bool tilted, bool flip, int radius) {

const float Pi = 4 \* atan(1.0);

for (GLfloat angle = Pi / 180; angle <= 2 \* Pi; angle += Pi / 6) {

glPushMatrix();

glTranslatef(radius \* cos(angle) + x, radius \* sin(angle) + y, 0);

glRotatef(tilt, 0, 0, 1);

if (flip)

glRotatef(180, 1, 0, 0);

if (tilted)

tilt += 30;

glScaled(0.1, 0.1, 0.1);

glTranslatef(-(radius \* cos(angle) + x), -(radius \* sin(angle) + y), 0);

drawPolyLineFile("Brontoi.dat");

glPopMatrix();

}

}

//<<<<<<<<<<<<<<<<<<<<<<< myInit >>>>>>>>>>>>>>>>>>>>

void myInit(void) {

glClearColor(1.0, 1.0, 1.0, 0.0); // background color is white

glColor3f(0.0f, 0.0f, 0.0f); // drawing color is black

glPointSize(10.0); // a 'dot' is 2 by 2 pixels

glMatrixMode(GL\_PROJECTION); // set "camera shape"

glLoadIdentity();

gluOrtho2D(0.0, (GLdouble)screenWidth, 0.0, (GLdouble)screenHeight);

A = screenWidth / 4.0; // set values used for scaling and shifting

B = 0.0;

C = D = screenHeight / 2.0;

}

//<<<<<<<<<<<<<<<<<<<<<<<< myDisplay >>>>>>>>>>>>>>>>>

void myDisplay(void) { // plot the sinc function, using world coordinates

glClear(GL\_COLOR\_BUFFER\_BIT);

setWindow(0, 640.0, 0, 480.0); // set a fixed window

draw(screenWidth / 4, 3 \* screenHeight / 4, -90, 1, 0, 90); //top left

draw(3 \* screenWidth / 4, 3 \* screenHeight / 4 + 30, 0, 0, 0, 90); //top right

draw(screenWidth / 4, screenHeight / 4, 270, 1, 1, 90); //bottom left

draw(3 \* screenWidth / 4, screenHeight / 4 + 20, 0, 0, 1, 90); //bottom right

}

//<<<<<<<<<<<<<<<<<<<<<<<< main >>>>>>>>>>>>>>>>>>>>>>

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

glutInit(&argc, argv); // initialize the toolkit

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB); // set display mode

glutInitWindowSize(screenWidth, screenHeight); // set window size

glutInitWindowPosition(100, 150); // set window position on screen

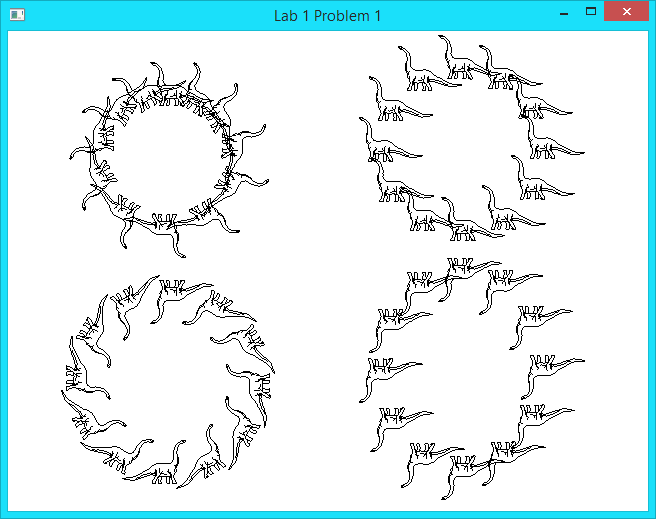
glutCreateWindow("Lab 1 Problem 1"); // open the screen window

glutDisplayFunc(myDisplay); // register redraw function

myInit();

glutMainLoop(); // go into a perpetual loop

}



//Steven Kong

//CSC 470 Computer Graphics

//Lab 1 Problem 2

#include<math.h>

#include<GL/glut.h>

#define SCREENWIDTH 640

#define SCREENHEIGHT 480

struct Point { GLfloat x, y; };

void myInit(void) {

glClearColor(0.5, 0.5, 0.5, 0.0);

glColor3f(1.0, 0.0f, 1.0f);

glPointSize(2.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, SCREENWIDTH, 0.0, SCREENHEIGHT);

}

void drawArc(Point center, GLfloat radius,

GLfloat startingAngle, GLfloat endingAngle, GLfloat r,

GLfloat g, GLfloat b) {

const GLfloat Pi = 4.0 \* atan(1.0);

Point prev, current;

glColor3f(r, g, b);

glBegin(GL\_LINES);

glVertex2f(center.x, center.y);

glVertex2f(center.x + radius \* cos((startingAngle \* Pi) / 180), center.y + radius \*

sin((startingAngle \* Pi) / 180));

glEnd();

for (GLfloat angle = (startingAngle \* Pi) / 180; angle <= (endingAngle \* Pi) / 180;

angle += Pi / 180) {

current.x = center.x + radius \* cos(angle);

current.y = center.y + radius \* sin(angle);

if (angle != (startingAngle \* Pi) / 180) {

glBegin(GL\_LINES);

glVertex2f(prev.x, prev.y);

glVertex2f(current.x, current.y);

glEnd();

glBegin(GL\_POLYGON);

glVertex2f(center.x, center.y);

glVertex2f(prev.x, prev.y);

glVertex2f(current.x, current.y);

glEnd();

glFlush();

}

prev.x = current.x;

prev.y = current.y;

}

glBegin(GL\_LINES);

glVertex2f(prev.x, prev.y);

glVertex2f(center.x, center.y);

glEnd();

}

void YingYang() {

Point center, leftCenter, rightCenter;

center.x = 320;

center.y = 240;

leftCenter.x = 240;

leftCenter.y = 240;

rightCenter.x = 400;

rightCenter.y = 240;

GLfloat radius = 161;

GLfloat radius2 = 159;

GLfloat subRadius1 = 80;

GLfloat subRadius2 = 20;

//Draw whole circle with black color

drawArc(center, radius, 0, 360, 0, 0, 0);

//Draw bottom half circle with white color

drawArc(center, radius2, 180, 360, 1, 1, 1);

//Draw left half of circle with subRadius1 with black color

drawArc(leftCenter, subRadius1, 0, 361, 0, 0, 0);

//Draw right half of circle with subRadius1 with white color

drawArc(rightCenter, subRadius1, 0, 361, 1, 1, 1);

//Draw a circle to left with subRadius2 and white color

drawArc(leftCenter, subRadius2, 0, 361, 1, 1, 1);

//Draw a circle to right with subRadius2 and black color

drawArc(rightCenter, subRadius2, 0, 361, 0, 0, 0);

}

void myDisplay() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1, 1, 1);

glRecti(100, 20, 540, 460);

YingYang();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(SCREENWIDTH, SCREENHEIGHT);

glutInitWindowPosition(100, 150);

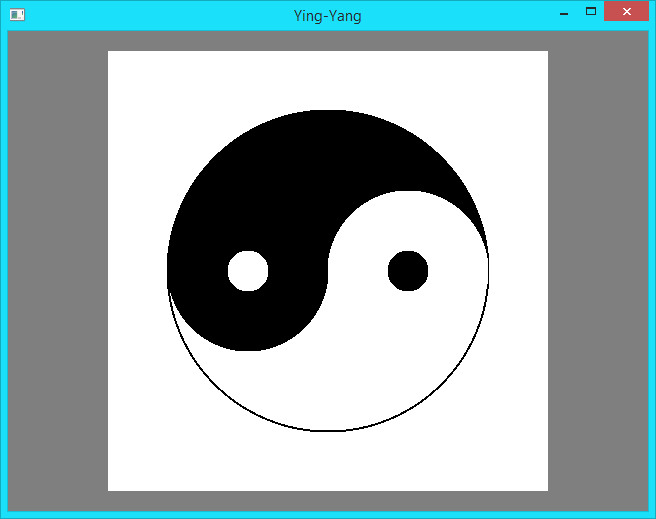
glutCreateWindow("Lab 1 Problem 2");

glutDisplayFunc(myDisplay);

myInit();

glutMainLoop();

}



//Steven Kong

//CSC 470 Computer Graphics

//Lab 1 Problem 3

#include<stdio.h>

#include <GL/glut.h>

#include<math.h>

using namespace std;

const int screenWidth = 640; // width of screen window in pixels

const int screenHeight = 480; // height of screen window in pixels

GLfloat r1 = 1, g1 = 0, b1 = 1;

GLfloat r2 = 0, g2 = 0, b2 = 1;

GLfloat r3 = 0, g3 = 1, b3 = 0;

int shape = 0;

void drawTriangle(GLfloat, GLfloat, GLfloat, GLfloat, GLfloat, GLfloat);

void drawRectangle(GLfloat, GLfloat, GLfloat, GLfloat, GLfloat, GLfloat, GLfloat, GLfloat);

void menu();

//<<<<<<<<<<<<<<<<<<<<<<< myInit >>>>>>>>>>>>>>>>>>>>

void myInit(void) {

glPointSize(10.0); // a 'dot' is 2 by 2 pixels

glMatrixMode(GL\_PROJECTION); // set "camera shape"

glLoadIdentity();

gluOrtho2D(0.0, 1, 0.0, 1);

}

//<<<<<<<<<<<<<<<<<<<<<<<< myDisplay >>>>>>>>>>>>>>>>>

void myDisplay(void) {

glClearColor(0, 0, 0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

if (shape == 1) { //Triangle

drawTriangle(0, 0, 1, 0, 0.5, 1);

drawTriangle(0, 0, .5, .325, 1, 0);

drawTriangle(0, 0, .5, 1, .45, .45);

drawTriangle(1, 0, .5, 1, .55, .45);

}

if (shape == 2) { //Pentagon

glBegin(GL\_POLYGON);

glColor3f(r3, g2, b1);

glVertex2f(0.5, 1.0);

glVertex2f(1.0, 0.575);

glVertex2f(0.8, 0.0);

glVertex2f(0.2, 0.0);

glVertex2f(0.0, 0.575);

glEnd();

drawTriangle(0.8, 0, .2, 0, .5, .4);

drawTriangle(0.2, 0, 0, .575, .4, .45);

drawTriangle(.8, 0, 1, .575, .6, .45);

drawTriangle(0.5, 1, 0, .575, .45, .55);

drawTriangle(0.5, 1, 1, .575, .55, .55);

}

if (shape == 3) { //Tetradecagon

drawTriangle(.3, 0, .5, .33, 0.7, 0);

drawRectangle(.7, 0, 1, 0, 1, .33, .6, .4);

drawRectangle(.3, 0, 0, 0, 0, .33, .4, .4);

drawTriangle(1, .33, 1, .8, .65, .55);

drawTriangle(0, .33, 0, .8, .35, .55);

drawTriangle(.5, 1, 0, 1, .45, .65);

drawTriangle(.5, 1, 1, 1, .55, .65);

drawTriangle(.55, .65, 1, .8, 1, 1);

drawTriangle(.45, .65, 0, .8, 0, 1);

}

if (shape == 4) { //Hexagon

glBegin(GL\_POLYGON);

glColor3f(r3, g2, b1);

glVertex2f(0.0, 0.25);

glVertex2f(0.5, 0.0);

glVertex2f(1.0, 0.25);

glVertex2f(1.0, 0.75);

glVertex2f(0.5, 1.0);

glVertex2f(0.0, 0.75);

glEnd();

drawTriangle(0, .25, .4, .4, 0.5, 0);

drawTriangle(0.5, 0, .575, .4, 1.0, .25);

drawTriangle(1.0, 0.25, .65, .5125, 1, .75);

drawTriangle(0, .25, .35, .5125, 0, .75);

drawTriangle(0, .75, .425, .625, 0.5, 1.0);

drawTriangle(0.5, 1.0, .575, .625, 1.0, 0.75);

const float Pi = 4 \* atan(1.0);

glBegin(GL\_POLYGON);

glColor3f(r2, g3, b1);

for (GLfloat angle = Pi / 180; angle <= 2 \* Pi; angle += Pi / 12)

{

glVertex2f(0.175\*cos(angle) + 0.5, 0.175\*sin(angle) + 0.5);

}

glEnd();

}

glFlush();

}

void drawTriangle(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2, GLfloat x3, GLfloat y3) {

glBegin(GL\_POLYGON);

glColor3f(r1, g1, b1);

glVertex2f(x1, y1);

glColor3f(r2, g2, b2);

glVertex2f(x2, y2);

glColor3f(r3, g3, b3);

glVertex2f(x3, y3);

glEnd();

}

void drawRectangle(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2, GLfloat x3, GLfloat y3, GLfloat x4, GLfloat y4) {

glBegin(GL\_POLYGON);

glColor3f(r1, g1, b1);

glVertex2f(x1, y1);

glColor3f(r2, g2, b2);

glVertex2f(x2, y2);

glColor3f(r3, g3, b3);

glVertex2f(x3, y3);

glColor3f(r1, g2, b3);

glVertex2f(x4, y4);

glEnd();

}

void colorMenu1(int id) {

if (id == 1) { r1 = 1.0; g1 = 0.0; b1 = 0.0; }

else if (id == 2) { r1 = 0.0; g1 = 1.0; b1 = 0.0; }

else if (id == 3) { r1 = 0.0; g1 = 0.0; b1 = 1.0; }

else if (id == 4) { r1 = 1.0; g1 = 1.0; b1 = 1.0; }

myDisplay();

}

void colorMenu2(int id) {

if (id == 1) { r2 = 1.0; g2 = 0.0; b2 = 0.0; }

else if (id == 2) { r2 = 0.0; g2 = 1.0; b2 = 0.0; }

else if (id == 3) { r2 = 0.0; g2 = 0.0; b2 = 1.0; }

else if (id == 4) { r2 = 1.0; g2 = 1.0; b2 = 1.0; }

myDisplay();

}

void colorMenu3(int id) {

if (id == 1) { r3 = 1.0; g3 = 0.0; b3 = 0.0; }

else if (id == 2) { r3 = 0.0; g3 = 1.0; b3 = 0.0; }

else if (id == 3) { r3 = 0.0; g3 = 0.0; b3 = 1.0; }

else if (id == 4) { r3 = 1.0; g3 = 1.0; b3 = 1.0; }

myDisplay();

}

void shapeMenu(int id) {

shape = id;

myDisplay();

}

//<<<<<<<<<<<<<<<<<<<<<<<< main >>>>>>>>>>>>>>>>>>>>>>

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

glutInit(&argc, argv); // initialize the toolkit

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB); // set display mode

glutInitWindowSize(screenWidth, screenHeight); // set window size

glutInitWindowPosition(100, 150); // set window position on screen

glutCreateWindow("Lab 1 Problem 3"); // open the screen window

glutDisplayFunc(myDisplay); // register redraw function

menu();

myInit();

glutMainLoop(); // go into a perpetual loop

}

void colors(int id) {};

void menu() {

int colorr, color1, color2, color3, shapes;

color1 = glutCreateMenu(colorMenu1);

glutAddMenuEntry("Red", 1);

glutAddMenuEntry("Green", 2);

glutAddMenuEntry("Blue", 3);

glutAddMenuEntry("White", 4);

color2 = glutCreateMenu(colorMenu2);

glutAddMenuEntry("Red", 1);

glutAddMenuEntry("Green", 2);

glutAddMenuEntry("Blue", 3);

glutAddMenuEntry("White", 4);

color3 = glutCreateMenu(colorMenu3);

glutAddMenuEntry("Red", 1);

glutAddMenuEntry("Green", 2);

glutAddMenuEntry("Blue", 3);

glutAddMenuEntry("White", 4);

colorr = glutCreateMenu(colors);

glutAddSubMenu("Color 1", color1);

glutAddSubMenu("Color 2", color2);

glutAddSubMenu("Color 3", color3);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

shapes = glutCreateMenu(shapeMenu);

glutAddMenuEntry("Triangle", 1);

glutAddMenuEntry("Pentagon", 2);

glutAddMenuEntry("Tetradecagon", 3);

glutAddMenuEntry("Hexagon", 4);

glutAttachMenu(GLUT\_LEFT\_BUTTON);

}

