//Andy Huang

//Homework 1, Problem 1

//CSC 470 Computer Graphics

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <GL/glut.h>

#include <fstream>

#include <string>

using namespace std;

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

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

char \* file = "Brontoi.dat";

const float PI = 4\*atan(1);

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 file

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 drawDino(GLfloat startX, GLfloat startY, int tilt, bool tilted, bool flip,int radius){

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

{

glPushMatrix();

glTranslatef(radius \* cos(angle) + startX, radius \* sin(angle) + startY, 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) + startX), -(radius \* sin(angle) + startY), 0);

drawPolyLineFile(file);

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

/\*

glViewport(120, 440, 50, 30); // set the next viewport

glRotatef(0, 120, 440, 0);

drawPolyLineFile("Brontoi.dat"); // draw it again

\*/

drawDino((screenWidth / 4) \* 1, (screenHeight / 4) \* 3, -90, 1, 0, 90); //Shows dinos in circle with heads at angles

drawDino((screenWidth / 4) \* 3, (screenHeight / 4) \* 3 + 30, 0, 0, 0, 90);

drawDino((screenWidth / 4) \* 1, (screenHeight / 4) \* 1, 270, 1, 1, 90);

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

}

//<<<<<<<<<<<<<<<<<<<<<<<< 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((glutGet(GLUT\_SCREEN\_WIDTH) - screenWidth) / 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - screenHeight) / 2);

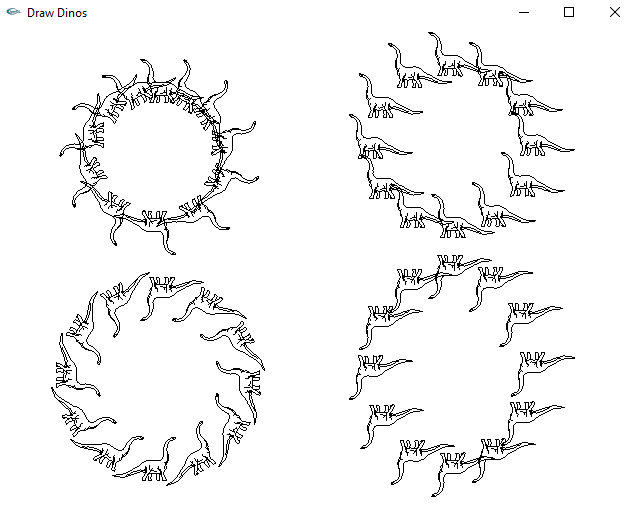
glutCreateWindow("Draw Dinos"); // open the screen window

glutDisplayFunc(myDisplay); // register redraw function

myInit();

glutMainLoop(); // go into a perpetual loop

}



//Andy Huang

//Homework 1, Problem 2

//CSC 470 Computer Graphics

#include <math.h>

#include <iostream>

#include <GL\glut.h>

#define SCREENWIDTH 640

#define SCREENHEIGHT 480

struct Point

{

int 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 float Pi = 4 \* 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;

//Calculating the Center

center.x = 640 / 2;

center.y = 480 / 2;

//Calculating the Left Center

leftCenter.x = 640 / 2 - 80;

leftCenter.y = 480 / 2;

//Calculating the Right Center

rightCenter.x = 640 / 2 + 80;

rightCenter.y = 480 / 2;

drawArc(center, 160, 0, 360, 0, 0, 0); //This is to draw the black part

drawArc(center, 162, 0, 360, 0, 0, 0); //This is to draw an outline for the circle

drawArc(center, 160, 180, 360, 1, 1, 1); //This is to draw the white circle

drawArc(leftCenter, 80, 180, 361, 0, 0, 0); //This is to draw the black semi circle

drawArc(rightCenter, 80, 0, 181, 1, 1, 1); //This is to draw the white semi circle

drawArc(leftCenter, 20, 0, 360, 1, 1, 1); //This is to draw the small white circle

drawArc(rightCenter, 20, 0, 360, 0, 0, 0); //This is to draw the small black circle

}

void myDisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(.5, .5, .5);

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);

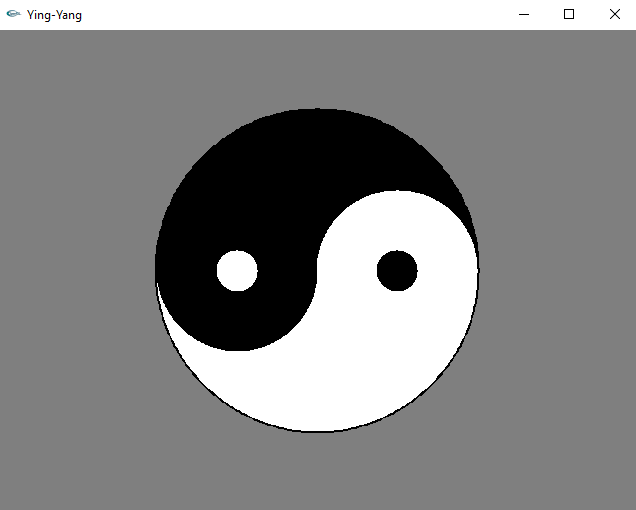
glutCreateWindow("Ying-Yang");

glutDisplayFunc(myDisplay);

myInit();

glutMainLoop();

}



//Andy Huang

//Homework 1, Problem 3

//CSC 470 Computer Graphics

#include <windows.h> // use proper includes for your system

#include <math.h>

#include <string>

#include <gl/Gl.h>

#include <gl/glut.h>

using namespace std;

const int screenWidth = 500; // width of screen window in pixels

const int screenHeight = 400; // height of screen window in pixels

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

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

int shape = -1;

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

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(left, right, bottom, top);

glClearColor(0, 0, 0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

}

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

glBegin(GL\_POLYGON);

glVertex2f(x1, y1);

glVertex2f(x2, y2);

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);

glVertex2f(x1, y1);

glVertex2f(x2, y2);

glVertex2f(x3, y3);

glVertex2f(x4, y4);

glEnd();

}

void drawPentagon(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2, GLfloat x3, GLfloat y3, GLfloat x4, GLfloat y4, GLfloat x5, GLfloat y5){

glBegin(GL\_POLYGON);

glVertex2f(x1, y1);

glVertex2f(x2, y2);

glVertex2f(x3, y3);

glVertex2f(x4, y4);

glVertex2f(x5, y5);

glEnd();

}

void drawHexagon(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2, GLfloat x3, GLfloat y3, GLfloat x4, GLfloat y4, GLfloat x5, GLfloat y5, GLfloat x6, GLfloat y6){

glBegin(GL\_POLYGON);

glVertex2f(x1, y1);

glVertex2f(x2, y2);

glVertex2f(x3, y3);

glVertex2f(x4, y4);

glVertex2f(x5, y5);

glVertex2f(x6, y6);

glEnd();

}

void myDisplay(void){

setWindow(0, 10, 0, 10);

if (shape == 2) glClearColor(r1, g1, b1, 0);

else glClearColor(0, 0, 0, 0);

glClear(GL\_COLOR\_BUFFER\_BIT);

if (shape == 0){

glColor3f(r1, g1, b1);

drawTriangle(0, 0, 10, 0, 5, 10);

glColor3f(r2, g2, b2);

drawTriangle(4.5, 4, 5.5, 4, 5, 10);

drawTriangle(0, 0, 5, 5, 5, 3.5);

drawTriangle(10, 0, 5, 5, 5, 3.5);

}

else if (shape == 1){

glColor3f(r1, g1, b1);

drawPentagon(5, 10, 0, 6, 1, 0, 9, 0, 10, 6);

glColor3f(r2, g2, b2);

drawTriangle(4.5, 4, 5.5, 4, 5, 10);

drawTriangle(1, 0, 5, 5, 5, 3.5);

drawTriangle(9, 0, 5, 5, 5, 3.5);

drawTriangle(0, 6, 5, 5, 5, 3.5);

drawTriangle(10, 6, 5, 5, 5, 3.5);

}

else if (shape == 2){

glColor3f(0, 0, 0);

drawTriangle(3, 0, 5, 3.3, 7, 0);

drawTriangle(0, 3.3, 0, 8, 3.5, 5.5);

drawTriangle(10, 3.3, 10, 8, 6.5, 5.5);

drawTriangle(5, 10, 0, 10, 4.5, 6.5);

drawTriangle(5, 10, 10, 10, 5.5, 6.5);

drawTriangle(5.5, 6.5, 10, 8, 10, 10);

drawTriangle(4.5, 6.5, 0, 8, 0, 10);

drawRectangle(7, 0, 10, 0, 10, 3.3, 6, 4);

drawRectangle(3, 0, 0, 0, 0, 3.3, 4, 4);

glColor3f(r1, g1, b1);

glRectf(10, 10, 0.0, 10);

}

else if (shape == 3){

glColor3f(r1, g1, b1);

drawHexagon(2, 0, 8, 0, 10, 5, 8, 10, 2, 10, 0, 5);

glColor3f(r2, g2, b2);

drawTriangle(0, 5, 5, 4, 5, 6);

drawTriangle(10, 5, 5, 4, 5, 6);

drawTriangle(2, 10, 4, 5, 6, 5);

drawTriangle(8, 10, 6, 5, 4, 5);

drawTriangle(2, 0, 4, 5, 6, 5);

drawTriangle(8, 0, 6, 5, 4, 5);

}

glFlush();

}

void color\_menu(int id){

if (id == 0) { r1 = 1.0; g1 = 0.0; b1 = 0.0; }

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

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

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

else if (id == 4) { r2 = 1.0; g2 = 0.0; b2 = 0.0; }

else if (id == 5) { r2 = 0.0; g2 = 1.0; b2 = 0.0; }

else if (id == 6) { r2 = 0.0; g2 = 0.0; b2 = 1.0; }

else if (id == 7) { r2 = 1.0; g2 = 1.0; b2 = 1.0; }

myDisplay();

}

void shape\_menu(int id){

if (id == 4) exit(0);

else shape = id;

myDisplay();

}

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

{

int colorMenu1, colorMenu2, shapeMenu;

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

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

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

glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - screenWidth) / 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - screenHeight) / 2); // set window position on screen

glutCreateWindow("Draw Shapes"); // open the screen window

glutDisplayFunc(myDisplay); // register redraw function

colorMenu1 = glutCreateMenu(color\_menu);

glutAddMenuEntry("RED", 0);

glutAddMenuEntry("GREEN", 1);

glutAddMenuEntry("BLUE", 2);

glutAddMenuEntry("WHITE", 3);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

colorMenu2 = glutCreateMenu(color\_menu);

glutAddMenuEntry("RED", 4);

glutAddMenuEntry("GREEN", 5);

glutAddMenuEntry("BLUE", 6);

glutAddMenuEntry("WHITE", 7);

glutCreateMenu(color\_menu);

glutAddSubMenu("OUTSIDE COLOR", colorMenu1);

glutAddSubMenu("INSIDE COLOR", colorMenu2);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

shapeMenu = glutCreateMenu(shape\_menu);

glutAddMenuEntry("TRIANGLE", 0);

glutAddMenuEntry("PENTAGON", 1);

glutAddMenuEntry("7 POINTED STAR", 2);

glutAddMenuEntry("HEXAGON", 3);

glutAddMenuEntry("QUIT", 4);

glutAttachMenu(GLUT\_LEFT\_BUTTON);

glutMainLoop(); // go into a perpetual loop

}

