In this assignment you are to create specific animations of a polygon.

The polygon should be created using the following technique:

* Each vertex is defined by clicking on the window. The final number of sides is determined by clicking on the first vertex a second time.
* Once the polygon has been defined, it should be filled in and begin to rotate.

Polygon behavior is determined as follows:

* clicking and dragging the polygon translates the polygon within the window
* clicking and dragging the polygon while holding the shift key scales the polygon
* Clicking the polygon while holding the Alt key reverses the direction of rotation.
* As usual you get bonus points for doing interesting things above and beyond the program requirements such as:
* allowing multiple polygons
* including additional transformations such as shears or reflections, but those are just suggestions. I'm sure you can come up with more.

Here's how you win at completing Cooper assignments

* Meeting the requirements of the assignment. 70% of the grade
* Internal program documentation. 10% of the grade
* Appropriately professional code structure and organization. 10% of the grade
* doing 'something' above and beyond. 10% of the grade

Here's how you lose at completing Cooper assignments

* Failing to use appropriate features of your programming language of choice. -5%
* Submitting code embedded in a .docx, .pdf, .rar, or any other file format unrelated to programming assignments. -5%
* Failing to submit evidence that the code compiles, executes, and meets the assignment requirements. -5%

#include <windows.h>

#include <GL/glut.h>

#include <iostream>

#include "Shape.h"

#include "Location.h"

#include "RGBColor.h"

using namespace std;

static bool spinning = true; //animated or not

static const int FPS = 60; //frames to render

Location origin(25, 25, 0); // point coordinate

RGBColor currentColor(0.9, 0.1, 0.1); // color

Location\* vertices; // array of vertices for the shape

Shape myShape(6, currentColor, origin, GL\_POLYGON); // create a new shape

static GLfloat currentAngleOfRotation = 0.0; // rotation angle

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: reshape

Purpose: maintain aspect ratio

Author: Cooper

Date: 2/21/2020

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void reshape(GLint width, GLint height) {

glViewport(0, 0, width, height);

GLfloat aspect = (GLfloat)width / (GLfloat)height;

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (width <= height) glOrtho(-50.0, 50.0, -50.0 / aspect, 50.0 / aspect, -1.0, 1.0);

else glOrtho(-50.0 \* aspect, 50.0 \* aspect, -50.0, 50.0, -1.0, 1.0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function: init()

purpose: initalize the hape to be drawn and calcuates the vertices of the shape

Author: Cooper

Date: 2/21/2020

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void init() {

myShape.setVertices(25);

vertices = myShape.getVertices();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function: display()

purpose: displays a rotating shape

Author: Cooper

Date: 2/21/2020

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT); //setup

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glRotatef(currentAngleOfRotation, 0.0, 0.0, 1.0); // rotate the shape

myShape.setVertices(25); // set the vertices of the shape

myShape.setOrigin(origin); // set the origin of the shape

vertices = myShape.getVertices(); // put the vertices in an array

glBegin(GL\_POLYGON); // and draw it

glColor3f(currentColor.getRed(), currentColor.getGreen(), currentColor.getBlue());

for (int i = 0; i < myShape.getSides(); i++) {

glVertex2f(vertices[i].getX(), vertices[i].getY());

}

glEnd();

glFlush();

glutSwapBuffers();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function: timer()

purpose: controls the refresh rate

Author: Cooper

Date: 2/21/2020

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void timer(int v) {

if (spinning) {

currentAngleOfRotation += 1.0;

if (currentAngleOfRotation > 360.0) {

currentAngleOfRotation -= 360.0; // reset to zero every revolution

}

glutPostRedisplay();

}

glutTimerFunc(1000 / FPS, timer, v);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function: mouse()

purpose: controls mouse behavior

left click starts the image rotating

right click pauses the rotation

Author: Cooper

Date: 2/21/2020

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) {

spinning = true;

glutSetCursor(GLUT\_CURSOR\_DESTROY);

}

else if (button == GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN) {

spinning = false;

glutSetCursor(GLUT\_CURSOR\_WAIT);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function: keyboard()

purpose: controls keyboard behavior

number key 3-8 control number of sides

keys r,g,b control shape color

Author: Cooper

Date: 2/21/2020

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void keyboard(unsigned char Key, int x, int y) {

switch (Key) {

case '3': myShape.setSides(3); break;

case '4': myShape.setSides(4); break;

case '5': myShape.setSides(5); break;

case '6': myShape.setSides(6); break;

case '7': myShape.setSides(7); break;

case '8': myShape.setSides(8); break;

case 'r': currentColor.setColors(1.0, 0.0, 0.0); break;

case 'b': currentColor.setColors(0.0, 0.0, 1.0); break;

case 'g': currentColor.setColors(0.0, 1.0, 0.0); break;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function: SpecialKeys()

purpose: controls non alphanumeric keys

arrow keys up/down/left/right change windo size

Author: Cooper

Date: 2/21/2020

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SpecialKeys(int key, int x, int y) {

switch (key) {

case GLUT\_KEY\_LEFT: glutReshapeWindow(glutGet(GLUT\_WINDOW\_WIDTH) - 1, glutGet(GLUT\_WINDOW\_HEIGHT)); break;

case GLUT\_KEY\_RIGHT: glutReshapeWindow(glutGet(GLUT\_WINDOW\_WIDTH) + 1, glutGet(GLUT\_WINDOW\_HEIGHT)); break;

case GLUT\_KEY\_UP: glutReshapeWindow(glutGet(GLUT\_WINDOW\_WIDTH), glutGet(GLUT\_WINDOW\_HEIGHT) + 1); break;

case GLUT\_KEY\_DOWN:glutReshapeWindow(glutGet(GLUT\_WINDOW\_WIDTH), glutGet(GLUT\_WINDOW\_HEIGHT) - 1); break;

case GLUT\_KEY\_F1: glutFullScreen(); break;

case GLUT\_KEY\_F2: glutReshapeWindow(800, 800); break;

case GLUT\_KEY\_F3: glutIconifyWindow(); break;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function: main()

purpose: sets up the window and assigns keyboard and

mouse listeners, sets the frame rate for

snimation, and shape aspect ratio

Author: Cooper

Date: 2/21/2020

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

init();

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowPosition(80, 80);

glutInitWindowSize(800, 500);

glutCreateWindow("Spinning Shape");

glutReshapeFunc(reshape);

glutDisplayFunc(display);

glutTimerFunc(100, timer, 0);

glutKeyboardFunc(keyboard);

glutSpecialFunc(SpecialKeys);

glutMouseFunc(mouse);

glutMainLoop();

}