#include<math.h>

#include<conio.h>

#include<stdio.h>

#include<iostream>

#include<iomanip>

#include<gl/glut.h>

using namespace std;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GLOBAL VALUES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

float theta = 0.0; //global angular value for rotation

float scalea = 1.0; //global scaling value

float scaleb = 1.0; //global scaling value

float dx = 0.0, dy = 0.0;

int frame = 1;

float Scalex = 2.0;

float Scaley = 2.0;

float scale = 1.0;

float Transx = 0.0;

float Transy = 0.0;

void init(void); //This is a function to initialize the window clear color

void RenderScene(void); //This a function to draw polyman in an opened window

void loadicon(float x[], float y[], int ndraws, int pointsperdraw[], int drawtype[], float colorr[], float colorg[], float colorb[], float rotate, float scalex, float scaley, float transx, float transy);

void settrans(void);

void DrawsAllIcons(float x[], float y[], int ndraws, int pointsperdraw[], int drawtype[], float colorr[], float colorg[], float colorb[], float rotate, float scalex, float scaley, float transx, float transy);//Generic Shape Function

void SetupRC(void);//Sets up the clear color

void TimerFunction(int); //This call back function is called each 30ms and changes the location, scale and rotation

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAIN PROGRAM\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

{

//Set up window title

char header[] = "Test One By Chris Stewart";

glutInit(&argc, argv);

//Set up the display mode with two buffers and RGB colors

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

//Initialize window size and position

glutInitWindowSize(560, 440);

glutInitWindowPosition(140, 20);

//Initialize background color of the window

SetupRC();

//Open and label window

glutCreateWindow(header);

glutDisplayFunc(RenderScene);

glutTimerFunc(30, TimerFunction, 1);

//Now draw the scene

glutMainLoop();

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RenderScene Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void RenderScene(void)

{

//load Square icon

float sx[12], sy[12], squarecolorr[3], squarecolorg[3], squarecolorb[3];

int squarendraws;

int squarepointsperdraw[3], squaredrawtype[3];

sx[0] = 1.0; sy[0] = 1.0;

sx[1] = 1.0; sy[1] = -1.0;

sx[2] = -1.0; sy[2] = -1.0;

sx[3] = -1.0; sy[3] = 1.0;

sx[4] = 1.0; sy[4] = 1.0;

sx[5] = 0.0; sy[5] = 2.0;

sx[6] = 0.0; sy[6] = -2.0;

sx[7] = 0.0; sy[7] = -1.0;

sx[8] = 0.0; sy[8] = 1.0;

sx[9] = 1.0; sy[9] = 1.0;

sx[10] = 1.0; sy[10] = -1.0;

sx[11] = 0.0; sy[11] = -1.0;

squarecolorr[0] = 1.0; squarecolorb[0] = 1.0; squarecolorg[0] = 1.0;

squarecolorr[1] = 1.0; squarecolorb[1]= 0.0; squarecolorg[1]= 0.0;

squarecolorr[2] = 1.0; squarecolorb[2] = 0.0; squarecolorg[2] = 0.0;

squarendraws = 3;

squarepointsperdraw[0] = 5;

squaredrawtype[0] = 2;

squarepointsperdraw[1] = 2;

squaredrawtype[1] = 1;

squarepointsperdraw[2] = 5;

squaredrawtype[2] = 3;

//load Trapazoid icon

float tx[12], ty[12], trapcolorr[3], trapcolorg[3], trapcolorb[3];

int trapndraws;

int trappointsperdraw[3], trapdrawtype[3];

tx[0] = 1.0; ty[0] = 1.0;

tx[1] = 1.5; ty[1] = -1.0;

tx[2] = -1.5; ty[2] = -1.0;

tx[3] = -1.0; ty[3] = 1.0;

tx[4] = 1.0; ty[4] = 1.0;

tx[5] = 0.0; ty[5] = 2.0;

tx[6] = 0.0; ty[6] = -2.0;

tx[7] = 0.0; ty[7] = 1.0;

tx[8] = 1.0; ty[8] = 1.0;

tx[9] = 1.5; ty[9] = -1.0;

tx[10] = 0.0; ty[10] = -1.0;

tx[11] = 0.0; ty[11] = 1.0;

trapcolorr[0] = 1.0; trapcolorb[0] = 1.0; trapcolorg[0] = 1.0;

trapcolorr[1] = 1.0; trapcolorb[1] = 0.0; trapcolorg[1] = 0.0;

trapcolorr[2] = 0.0; trapcolorb[2] = 0.0; trapcolorg[2] = 1.0;

trapndraws = 3;

trappointsperdraw[0] = 5;

trapdrawtype[0] = 2;

trappointsperdraw[1] = 2;

trapdrawtype[1] = 1;

trappointsperdraw[2] = 5;

trapdrawtype[2] = 3;

//load combined icon

float stx[22], sty[22], strapcolorr[5], strapcolorg[5], strapcolorb[5];

int strapndraws;

int strappointsperdraw[5], strapdrawtype[5];

stx[0] = 1.0; sty[0] = 1.0;

stx[1] = 1.0; sty[1] = 0.0;

stx[2] = -1.0; sty[2] = 0.0;

stx[3] = -1.0; sty[3] = 1.0;

stx[4] = 1.0; sty[4] = 1.0;

stx[5] = 1.5; sty[5] = 0.0;

stx[6] = 2.0; sty[6] = -1.0;

stx[7] = -2.0; sty[7] = -1.0;

stx[8] = -1.5; sty[8] = 0.0;

stx[9] = 1.5; sty[9] = 0.0;

stx[10] = 0.0; sty[10] = 3.0;

stx[11] = 0.0; sty[11] = -3.0;

stx[12] = 0.0; sty[12] = 0.0;

stx[13] = 0.0; sty[13] = 1.0;

stx[14] = 1.0; sty[14] = 1.0;

stx[15] = 1.0; sty[15] = 0.0;

stx[16] = 0.0; sty[16] = 0.0;

stx[17] = 1.5; sty[17] = 0.0;

stx[18] = 2.0; sty[18] = -1.0;

stx[19] = 0.0; sty[19] = -1.0;

stx[20] = 0.0; sty[20] = 0.0;

stx[21] = 1.5; sty[21]=-0.0;

strapcolorr[0] = 1.0; strapcolorb[0] = 1.0; strapcolorg[0] = 1.0;

strapcolorr[1] = 1.0; strapcolorb[1] = 1.0; strapcolorg[1] = 1.0;

strapcolorr[2] = 1.0; strapcolorb[2] = 0.0; strapcolorg[2] = 0.0;

strapcolorr[3] = 1.0; strapcolorb[3] = 0.0; strapcolorg[3] = 0.0;

strapcolorr[4] = 0.0; strapcolorb[4] = 0.0; strapcolorg[4] = 1.0;

strapndraws = 5;

strappointsperdraw[0] = 5;

strapdrawtype[0] = 2;

strappointsperdraw[1] = 5;

strapdrawtype[1] = 2;

strappointsperdraw[2] = 2;

strapdrawtype[2] = 1;

strappointsperdraw[3] = 5;

strapdrawtype[3] = 3;

strappointsperdraw[4] = 5;

strapdrawtype[4] = 3;

settrans();

cout << "in RenderScene" << endl;

//set the current drawing color to white

glColor3f(1.0, 1.0, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//set the viewport to the window dimensions

glViewport(0, 0, 560, 440);

//Establish the clipping volumn in user units, first clear all the translation matrices

glOrtho(-7.0, 7.0, -7.0, 7.0, 1.0, -1.0);

//clear the window with the background color

glClear(GL\_COLOR\_BUFFER\_BIT);

//set the current drawing color to white

glColor3f(1.0, 1.0, 1.0);

//now call each method to draw the figure

//DrawsAllIcons(sx, sy, squarendraws, squarepointsperdraw, squaredrawtype, squarecolorr, squarecolorg, squarecolorb, -5.0, Scalex, Scaley, 5.0, 5.0);

//DrawsAllIcons(tx, ty, trapndraws, trappointsperdraw, trapdrawtype, trapcolorr, trapcolorg, trapcolorb, -5.0, Scalex, Scaley, -5.0, -5.0);

DrawsAllIcons(stx, sty, strapndraws, strappointsperdraw, strapdrawtype, strapcolorr, strapcolorg, strapcolorb, -5.0, scale, scale, 0.0, 0.0);

glEnd();

glutSwapBuffers();

return;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Draw Icon\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void DrawsAllIcons(float x[], float y[], int ndraws, int pointsperdraw[], int drawtype[], float

colorr[], float colorg[], float colorb[], float rotate, float scalex, float scaley, float transx, float

transy) {

int count = 0;

Scalex = scalex;

Scaley = scaley;

Transx = transx;

Transy = transy;

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

if (drawtype[i] == 2) {

glColor3f(colorr[i], colorg[i], colorb[i]);

glBegin(GL\_LINE\_STRIP);

for (int j = 0; j < pointsperdraw[i]; j++) {

glVertex2f(x[count], y[count]);

count++;

}

glEnd();

}

else if (drawtype[i] == 1) {

glColor3f(colorr[i], colorg[i], colorb[i]);

glBegin(GL\_LINES);

for (int j = 0; j < pointsperdraw[i]; j++) {

glVertex2f(x[count], y[count]);

count++;

}

glEnd();

}

else if (drawtype[i] == 3) {

glColor3f(colorr[i], colorg[i], colorb[i]);

glShadeModel(GL\_FLAT);

//redraw the polygon

glBegin(GL\_POLYGON);

for (int j = 0; j < pointsperdraw[i]; j++) {

glVertex2f(x[count], y[count]);

count++;

}

if (i == ndraws - 1) {

glFlush();

}

else {

glEnd();

}

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Function SetupRC\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Setup the rendering state

void SetupRC(void)

{// this function sets the clear color of an open window and clears the open window

// Set clear color to blue

glClearColor(0.0, 0.0, 1.0, 1.0);

return;

}//end of SetupRC

void settrans(void)

/\*Sets the MODELVIEW MATRIX for the square. Note that the calls are done backqards

that is if we want to rotate and move the pattern, call glTranslate first and then glRotate \*/

{

cout << "in settrans" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(dx, dy, 0.0);

glScalef(scalea, scaleb, 0.0);

glRotatef(theta, 0.0, 0.0, 1.0);// note that the angle theta is in degrees, not radians

return;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Functioner Timer\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TimerFunction(int value)

//this call back function is call each 30 ms and changes the location,scale and rotation

// of the square.

{

switch (frame)

{

case 1: //frame 1 scale shape

scalea = scalea \* Scalex;

scaleb = scaleb \* Scaley;

if (scalea >= 4) {

frame = 2;

}

else {

frame = 2;

}

break;

case 2: //frame 2 translate and return to normal size

dx = dx + Transx;

dy = dy + Transy;

scalea = 1.0;

scaleb = 1.0;

if (dx != 0.0) {

frame = 3;

}

else {

frame = 3;

}

//we key on Polyman's position to change the frame

break;

case 3:// frame 3 spin

theta = theta - 90.0;

if (theta > -360.0)

{

frame = 3;

}

break;

}

// Redraw the scene with new coordinates

glutPostRedisplay();

glutTimerFunc(1000, TimerFunction, 1);

}





