#include<stdlib.h>

#include<GL/glut.h>

#include<math.h> //Defining libraries

#include <windows.h>

#include <time.h>

#include <math.h>

#define GLUT\_DISABLE\_ATEXIT\_HACK // In case of errors like undefined reference to `\_\_glutInitWithExit@12’, should be before including glut.h

#define PI 3.14159265

int screenWidth = 600;

int screenHeight = 600;

int delay = 10; //Defining variables (for mouse click, rotating, screen size..)

double A[3] = { 0,0,0 };

double B[3] = { 0,0,0 };

double alfa = 0;

int fx = 0, fy = 0, fz = 0;

float sphi = 0.0, stheta = 0.0;

float sside = 0, sdepth = -5;

float sx = 0, sy = 0;

bool mouse\_left\_click, mouse\_middle\_click, mouse\_right\_click;

int mouseX, mouseY;

double rotate\_y = 0;

double rotate\_x = 0;

void myIdle(int frame)

{

// Animation routine which calls itself after “delay” miliseconds.

alfa += 10;

if (alfa > 360) alfa -= 360;

A[1] = sin(alfa \* PI / 180);

A[2] = cos(alfa \* PI / 180);

B[1] = sin((alfa + 180) \* PI / 180);

B[2] = cos((alfa + 180) \* PI / 180);

// Calling Itself

glutTimerFunc(delay, myIdle, 0);

glutPostRedisplay();

}

void init()

{

glClearColor(0, 0, 0, 1);

glMatrixMode(GL\_PROJECTION); // Use the Projection Matrix

glLoadIdentity();// Reset Matrix

glOrtho(-500, 500, -500, 500, -500, 500);

glMatrixMode(GL\_MODELVIEW);// Get Back to the Modelview

myIdle(0);// Start animation

}

void front()

{//front

glPushMatrix();

glTranslatef(-220, -250, 0);

glScalef(0.01, 0.3, 0.85);

glutWireCube(300);

glPopMatrix();

// axle for steering wheel

glPushMatrix();

glTranslatef(-220, -300, 0);

glScalef(-5, 1, 0.85);

glBegin(GL\_LINES);

glVertex3f(0, 0, 0);

glVertex3f(-10, 400, 0);

glEnd();

glPopMatrix();

}

void back()

{ //back

glPushMatrix();

glTranslatef(220, -250, 0);

glScalef(0.01, 0.3, 0.85);

glutWireCube(300);

glPopMatrix();

}

void bottom()

{

// bottom

glPushMatrix();

glTranslatef(0, -300, 0);

glScalef(1.5, 0.01, 0.85);

glutWireCube(300);

glPopMatrix();

// back axle

glPushMatrix();

glTranslatef(0, -300, 0);

glScalef(1.5, 0.01, 0.85);

glBegin(GL\_LINES);

glVertex3f(150, 0, -300);

glVertex3f(150, 0, 300);

glEnd();

glPopMatrix();

// front axle

glPushMatrix();

glTranslatef(0, -300, 0);

glScalef(1.5, 0.01, 0.85);

glBegin(GL\_LINES);

glVertex3f(-150, 0, -250);

glVertex3f(-150, 0, 250);

glEnd();

glPopMatrix();

}

void top()

{//top

glPushMatrix();

glTranslatef(0, -200, 0);

glScalef(1.5, 0.01, 0.85);

glutWireCube(300);

glPopMatrix();

}

void left()

{//left

glPushMatrix();

glTranslatef(0, -250, 130);

glScalef(1.5, 0.3, 0.01);

glutWireCube(300);

glPopMatrix();

}

void right()

{//right

glPushMatrix();

glTranslatef(0, -250, -130);

glScalef(1.5, 0.3, 0.01);

glutWireCube(300);

glPopMatrix();

}

void wheel(int a, int b, int c, int d, int e)

{

float th;

glBegin(GL\_LINE\_STRIP);

for (int i = 0; i < 360; i++)

{

th = i \* 3.142 / 180;

glVertex3f(a + d \* cos(th), b + e \* sin(th), c);

}

glEnd();

}

void rotateLine() // line inside of wheels

{

// back left

glPushMatrix();

glTranslatef(220, -300, -250);

glScalef(15, 60, 10);

glBegin(GL\_LINES);

glVertex3dv(A);

glVertex3dv(B);

glEnd();

glPopMatrix();

// back right

glPushMatrix();

glTranslatef(220, -300, 250);

glScalef(15, 60, 10);

glBegin(GL\_LINES);

glVertex3dv(A);

glVertex3dv(B);

glEnd();

glPopMatrix();

// front left

glPushMatrix();

glTranslatef(-220, -300, -200);

glScalef(15, 60, 10);

glBegin(GL\_LINES);

glVertex3dv(A);

glVertex3dv(B);

glEnd();

glPopMatrix();

// front right

glPushMatrix();

glTranslatef(-220, -300, 200);

glScalef(15, 60, 10);

glBegin(GL\_LINES);

glVertex3dv(A);

glVertex3dv(B);

glEnd();

glPopMatrix();

}

void car()

{

gluLookAt(4.0, 4.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

front();

back();

top();

bottom();

left();

right();

rotateLine(); // line inside of the wheels

wheel(220, -300, 250, 60, 60); //back right

wheel(220, -300, -250, 60, 60); // back left

wheel(-220, -300, 200, 60, 60); // front right

wheel(-220, -300, -200, 60, 60); // front left

wheel(-150, 100, 0, 80, 60); // steering wheel

}

float theta;

void display()

{

theta += 0.01;

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

//glRotatef(theta, 1, 1, 0);

glTranslatef(sside, 0, -sdepth);

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

glRotatef(sphi, 0, 1, 0);

glTranslatef(sx, 0, -sy);

// Rotate when user changes rotate\_x and rotate\_y

glRotatef(rotate\_x, 1.0, 0.0, 0.0);

glRotatef(rotate\_y, 0.0, 1.0, 0.0);

car();

glutSwapBuffers();

glutPostRedisplay();

}

void myReshape(int width, int height)

{ // adjust the camera aspect ratio to match that of the viewport

glViewport(0, 0, width, height); // update viewport

//glOrtho(-width,width,-height,height,-1000,1000);

glOrtho(-1, 1, -1, 1, -1, 1);

}

void GoMenu(int value)

{ //creates menu options for colors

switch (value)

{

case 1:

glColor3f(1.0, 0.0, 0.0);

break;

case 2:

glColor3f(0.0, 1.0, 0.0);

break;

case 3:

glColor3f(0.0, 0.0, 1.0);

break;

case 4:

glColor3f(1, 1, 1);

break;

case 5:

exit(0);

break;

}

glutPostRedisplay();

}

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

//gives the ability to rotate the up, down, right, left arrow keys

// Right arrow - increase rotation by 5 degree

if (key == GLUT\_KEY\_RIGHT)

rotate\_y += 5;

// Left arrow - decrease rotation by 5 degree

else if (key == GLUT\_KEY\_LEFT)

rotate\_y -= 5;

else if (key == GLUT\_KEY\_UP)

rotate\_x += 5;

else if (key == GLUT\_KEY\_DOWN)

rotate\_x -= 5;

// Request display update

glutPostRedisplay();

}

void myKeyboard(unsigned char key, int x, int y)

{//gives the ability to exit .exe file the ESC key

switch (key) {

case 27: // Escape

exit(-1);

}

glutPostRedisplay();

}

void myMouse(int button, int state, int x, int y)

{//gives the ability to rotate the mouse

mouseX = x; mouseY = y;

mouse\_left\_click = ((button == GLUT\_LEFT\_BUTTON) && (state == GLUT\_DOWN));

mouse\_middle\_click = ((button == GLUT\_MIDDLE\_BUTTON) &&

(state == GLUT\_DOWN));

mouse\_right\_click = ((button == GLUT\_RIGHT\_BUTTON) &&

(state == GLUT\_DOWN));

glutPostRedisplay();

}

void myMouseMove(int x, int y) {

// This even callback is executed whenver the mouse is moved

// rotate

if (mouse\_left\_click)

{

sphi += (float)(x - mouseX) / 4.0;

stheta += (float)(mouseY - y) / 4.0;

// if (stheta<0) stheta=0;

}

// scale

if (mouse\_middle\_click)

{

sx += (float)(x - mouseX) \* 50;

sy += (float)(y - mouseY) \* 50;

}

// scale

if (mouse\_right\_click)

{

sside += (float)(x - mouseX) \* 50;

sdepth += (float)(y - mouseY) \* 50;

}

mouseX = x;

mouseY = y;

glutPostRedisplay();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowPosition(0, 0); //creating console window

glutCreateWindow("Wire Car"); //window name

init();

//creating pop up menu

int sub1 = glutCreateMenu(GoMenu);

glutAddMenuEntry("Red", 1);

glutAddMenuEntry("Green", 2);

glutAddMenuEntry("Blue", 3);

glutAddMenuEntry("Back to White", 4);

glutCreateMenu(GoMenu);

glutAddSubMenu("Colors", sub1);

glutAddMenuEntry("Exit", 5);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

// Save callback

glutKeyboardFunc(myKeyboard);

glutSpecialFunc(specialKeys);

glutMouseFunc(myMouse);

glutMotionFunc(myMouseMove);

glutReshapeFunc(myReshape);

glutDisplayFunc(display);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

}