Computer Graphics Project

MUHARREM CAN DEMİR

Code block;

#include <iostream>

#include<stdio.h>

#include <gl/glut.h>

#include <cmath>

#include <math.h>

#define PI 3.14159265358979324

using namespace std;

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

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

double alfa = 0;

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

float sphi = 90.0, stheta = 0;

float sside = 1.0, sdepth = 0.5;

float sx = 0, sy = 0;

int delay = 1000;

int mouseDown = 0;

int tdönüs = 0.0f;

float xrot = 50.0f;

float yrot = 130.0f;

float knm\_x = 0.0f;

float knm\_y = 0.0f;

float knm\_z = 0.0f;

float xdiff = 100.0f;

float ydiff = 100.0f;

float grow\_shrink = 70.0f;

float resize\_f = 1.0f;

void Gövde()

{

glTranslatef(knm\_x, knm\_y, knm\_z);

glBegin(GL\_LINES);

//drawing of the body of the vehicle

glColor3f(255.0f, 255.0f, 255.0f);

glVertex3f(1.0f, 0.2f, -0.5f);

glVertex3f(-1.0f, 0.2f, -0.5f);

glVertex3f(-1.0f, 0.2f, -0.5f);

glVertex3f(-1.0f, -0.2f, -0.5f);

glVertex3f(-1.0f, -0.2f, -0.5f);

glVertex3f(1.0f, -0.2f, -0.5f);

glVertex3f(1.0f, -0.2f, -0.5f);

glVertex3f(1.0f, 0.2f, -0.5f);

glVertex3f(1.0f, 0.2f, 0.5f);

glVertex3f(-1.0f, 0.2f, 0.5f);

glVertex3f(-1.0f, 0.2f, 0.5f);

glVertex3f(-1.0f, -0.2f, 0.5f);

glVertex3f(-1.0f, -0.2f, 0.5f);

glVertex3f(1.0f, -0.2f, 0.5f);

glVertex3f(1.0f, -0.2f, 0.5f);

glVertex3f(1.0f, 0.2f, 0.5f);

glVertex3f(1.0f, 0.2f, -0.5f);

glVertex3f(1.0f, 0.2f, 0.5f);

glVertex3f(-1.0f, 0.2f, -0.5f);

glVertex3f(-1.0f, 0.2f, 0.5f);

glVertex3f(1.0f, -0.2f, -0.5f);

glVertex3f(1.0f, -0.2f, 0.5f);

glVertex3f(-1.0f, -0.2f, 0.5f);

glVertex3f(-1.0f, -0.2f, -0.5f);

glVertex3f(1.0f, -0.2f, -0.3f);

glVertex3f(0.45f, 0.78f, -0.3f);

glEnd();

}

void tekerdönüs(int frame)

{

//spinning wheel

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);

glutTimerFunc(delay, tekerdönüs, 0);

glutPostRedisplay();

}

void OnTeker()

{

// left front wheel

float R = 0.2; // Radius of circle.

float X = 1.0; // X-coordinate of center of circle.

float Y = -0.2; // Y-coordinate of center of circle

int numVertices = 25; // Number of vertices on circle.

float t = 0; // Angle parameter.

int i;

glPushMatrix();

glTranslatef(1.0, -0.3, -0.5);

//spinning wheel

glRotatef(tdönüs, 0.0f, 1.0f, 0.0f);

glTranslatef(-1.0, 0.3, 0.5);

//left wheel

glBegin(GL\_LINE\_LOOP);

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

{

glVertex3f(X + R \* cos(t), Y + R \* sin(t), -0.5);

t += 2 \* PI / numVertices;

}

glPopMatrix();

glEnd();

//left front wheel turn

tekerdönüs(0);

glPushMatrix();

glTranslatef(1.0, -0.2, -0.5);

glRotatef(stheta, 0, 0, 0);

glRotatef(sphi, 0, 1, 0);

glViewport(0, 0, 1024, 768);

glOrtho(-5, 5, -5, 5, -5, 5);

glColor3f(1, 1, 1);

glBegin(GL\_LINES);

glVertex3dv(A);

glVertex3dv(B);

glEnd();

glPopMatrix();

//right front wheel

glPushMatrix();

glTranslatef(1.0, -0.3, 0.5);

glRotatef(tdönüs, 0.0f, 1.0f, 0.0f);

glTranslatef(-1.0, 0.3, -0.5);

glBegin(GL\_LINE\_LOOP);

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

{

glVertex3f(X + R \* cos(t), Y + R \* sin(t), 0.5);

t += 2 \* PI / numVertices;

}

glPopMatrix();

glEnd();

//right front wheel turn

tekerdönüs(0);

glPushMatrix();

glTranslatef(1.0, -0.2, 0.5);

glRotatef(stheta, 0, 0, 0);

glRotatef(sphi, 0, 1, 0);

glViewport(0, 0, 1024, 768);

glOrtho(-5, 5, -5, 5, -5, 5);

glColor3f(1, 1, 1);

glBegin(GL\_LINES);

glVertex3dv(A);

glVertex3dv(B);

glEnd();

glPopMatrix();

}

void ArkaTeker()

{

//right rear wheel

float R = 0.2; // Radius of circle.

float X = -1.0; // X-coordinate of center of circle.

float Y = -0.2; // Y-coordinate of center of circle.

int numVertices = 25; // Number of vertices on circle

float t = 0;

int i;

glBegin(GL\_LINE\_LOOP);

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

{

glVertex3f(X + R \* cos(t), Y + R \* sin(t), -0.5);

t += 2 \* PI / numVertices;

}

glEnd();

//right rear wheel turn

tekerdönüs(0);

glPushMatrix();

glTranslatef(-1.0, -0.2, -0.5);

glRotatef(stheta, 0, 0, 0);

glRotatef(sphi, 0, 1, 0);

glViewport(0, 0, 1024, 768);

glOrtho(-5, 5, -5, 5, -5, 5);

glColor3f(1, 1, 1);

glBegin(GL\_LINES);

glVertex3dv(A);

glVertex3dv(B);

glEnd();

glPopMatrix();

//left rear wheel

glBegin(GL\_LINE\_LOOP);

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

{

glVertex3f(X + R \* cos(t), Y + R \* sin(t), 0.5);

t += 2 \* PI / numVertices;

}

glEnd();

//left rear wheel turn

tekerdönüs(0);

glPushMatrix();

glTranslatef(-1.0, -0.2, 0.5);

glRotatef(stheta, 0, 0, 0);

glRotatef(sphi, 0, 1, 0);

glViewport(0, 0, 1024, 768);

glOrtho(-5, 5, -5, 5, -5, 5);

glColor3f(1, 1, 1);

glBegin(GL\_LINES);

glVertex3dv(A);

glVertex3dv(B);

glEnd();

glPopMatrix();

}

void Direksiyon()

{

//steering drawing

float R = 0.15; // Radius of circle.

float X = 0.1; // X-coordinate of center of circle.

float Y = 0.1; // Y-coordinate of center of circle.

int numVertices = 25; // Number of vertices on circle.

float t = 0;

int i;

glPushMatrix();

//steering angle

glRotatef(sphi, 1.0f, 1.0f, 0.0f);

glRotatef(tdönüs, 0.0f, 0.0f, -1.0f);

glTranslatef(-0.1, -0.1, -0.3);

glBegin(GL\_LINE\_LOOP);

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

{

glVertex3f(X + R \* cos(t), Y + R \* sin(t), 1.0);

t += 2 \* PI / numVertices;

}

glPopMatrix();

glEnd();

}

int init(void)

{

glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LEQUAL);

glClearDepth(1.0f);

return 1;

}

void display(void)

{

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

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(

0.0f, 0.0f, 3.0f,

0.0f, 0.0f, 0.0f,

0.0f, 1.0f, 0.0f);

glRotatef(xrot, 1.0f, 0.0f, 0.0f);

glRotatef(yrot, 0.0f, 1.0f, 0.0f);

//calling functions

Gövde();

OnTeker();

ArkaTeker();

glPushMatrix();

glTranslatef(0.0, 1.3, -0.3);

Direksiyon();

glPopMatrix();

glFlush();

glutSwapBuffers();

}

void resize(int w, int h)

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glViewport(0, 0, w, h);

gluPerspective(grow\_shrink, resize\_f \* w / h, resize\_f, 100 \* resize\_f);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void idle(void)

{

if (!mouseDown)

{

xrot += 0.3f;

yrot += 0.4f;

}

glutPostRedisplay();

}

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

{

switch (key)

{

case 'a': //turn left

tdönüs += 1.0f;

break;

case 's'://turn right

tdönüs -= 1.0f;

break;

}

glutPostRedisplay();

}

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

{

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

{

mouseDown = 1;

xdiff = x - yrot;

ydiff = -y + xrot;

}

else

mouseDown = 0;

}

void mouseMotion(int x, int y)

{

if (mouseDown)

{

yrot = x - xdiff;

xrot = y + ydiff;

glutPostRedisplay();

}

}

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

{

glutInit(&argc, argv);

glutInitWindowPosition(75, 75); // the size and position (upper left corner) of the window on the screen.

glutInitWindowSize(1024, 768);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE); //initial display mode

glutCreateWindow("WireCar");

glutDisplayFunc(display); //display callback for the current window

glutKeyboardFunc(keyboard); //keyboard callback

glutMouseFunc(mouse); //mouse callback for the current window

glutMotionFunc(mouseMotion); //motion and passive motion callback

glutReshapeFunc(resize);

if (!init())

return 1;

glutMainLoop();

return 0;

}