AYDIN ADNAN MENDERES UNIVERSITY

COMPUTER GRAPHICS PROJECT REPORT

Name : Kağan Çiloğlu

(161805016@stu.adu.edu.tr)

Student no : 161805016

Course : CSE411 – Computer Graphics

Instructor : Samsun Başarıcı

(sbasarici@adu.edu.tr)

10 January 2021

Contents

Introduction

The main goal of the project is creating an animated wire car with using predefined and self-created functions. The vehicle is expecting to react to key presses such as ‘A’ and ‘S’. While one of these keys is pressed, front wheels and steering should turn to the proper direction. Also, the wheels should also be turning to give the impression of moving.

Terms and Functions

Used functions and related terms are given below.

Built-in Functions

glutInit() : glutInit initializes the GLUT library and negotiate a session with the window system.

glutInitDisplayMode() : glutInitDisplayMode sets the *initial display mode*.
glutInitWindowSize() : glutInitWindowSize sets the *initial window position* and *size* respectively.

glutCreateWindow() : glutCreateWindow creates a top-level window.

glutKeyboardFunc() : glutKeyboardFunc sets the keyboard callback for the *current window.*

glutReshapeFunc() : glutReshapeFunc sets the reshape callback for the *current window.*

glutDisplayFunc() : glutDisplayFunc sets the display callback for the *current window.*

glutMouseFunc() : glutMouseFunc sets the mouse callback for the *current window.*

glutMotionFunc() : glutMotionFunc sets the motion and passive motion callback respectively for the *current window.*

glutMainLoop() : glutMainLoop enters the GLUT event processing loop. Once called, this routine will never return. It calls as necessary any callbacks that have been registered.

glutPostRedisplay() : glutPostRedisplay marks the *current window* as needing to be redisplayed.

glutSwapBuffers() : glutSwapBuffers swaps the buffers of the *current window* if double buffered.

glTranslatef() : multiply the current matrix by a translation matrix.

glRotatef() : multiply the current matrix by a rotation matrix.

glColor3f() : Sets the current color.

glScalef() / glScaled() : multiply the current matrix by a general scaling matrix (d for double, f for float).

glutWireCube() : glutWireCube render a solid or wireframe cube respectively.

glPopMatrix() : pops the current matrix stack.

glPushMatrix() : pushes the current matrix stack.

glBegin() : delimit the vertices of a primitive or a group of like primitives.

glVertex2f() : specifies a vertex.

Self-created Functions

myIdle() : It provides the rotational movement of the stick that makes the wheels appear to be turning.

Idle() : Makes the steering wheel turn when button A is pressed.

IdleB() : Makes the steering wheel turn when button S is pressed.

myInit() : Function that provides perspective projections and initiates animations.

myKeyboard() : Allows us to assign tasks to the keys.

myReshape() : Adjusts the camera aspect ratio to match that of the viewport.

wheels() : It contains functions that enable wheels to be created. There are 4 loops, one loop for each wheel. Each loop draws a circle. When we position these circles; we get wheels.

spinLine() : It is used for forming and positioning rotating sticks on wheels. There are 5 functions in total, 4 for wheels and 1 for steering.

steeringWheel() : The function that enables the creation and positioning of the steering wheel. The steering wheel was created like a wheel, the only difference is that the wheel is perpendicular to the surface, while the steering wheel is horizontal.

wireCar() : Creates the axles, the steering rod and each surface of the car. The surfaces of the car made via glutWireCube. Axles and steering rod of the car made via glVertex2f.

myMouse() : Makes the program react to the mouse.

myMouseMove() : Allows the user to look at the car 360 degrees via mouse movements.

Source Codes

```
#include <windows.h>
#include <time.h>
#include <math.h>
#define GLUT_DISABLE_ATEXIT_HACK
#include <gl\gl.h>
#include <gl\glu.h>
#include <gl\glut.h>

#define PI 3.14159265
int screenWidth = 600;
int screenHeight = 600;
int delay = 10;
double A[3] = { 0,0,0 };
double B[3] = { 0,0,0 };
double Q[3] = { 0,0,0 };
double W[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;

//   Animation routine which calls itself after “delay” miliseconds.
void myIdle(int frame)
{
        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 Idle(void)
{
	alfa += 10;
	if (alfa > 360) alfa -= 360;
	Q[1] = sin(alfa*PI / 180);
	Q[2] = cos(alfa*PI / 180);
	W[1] = sin((alfa + 180)*PI / 180);
	W[2] = cos((alfa + 180)*PI / 180);

	// Calling Itself
	glutTimerFunc(600, 0, 0);
	glutPostRedisplay();
}
void IdleB(void)
{
	alfa -= 10;
	if (alfa > 360) alfa -= 360;
	Q[1] = sin(alfa*PI / 180);
	Q[2] = cos(alfa*PI / 180);
	W[1] = sin((alfa + 180)*PI / 180);
```

```
	W[2] = cos((alfa + 180)*PI / 180);

	// Calling Itself
	glutTimerFunc(600, 0, 0);
	glutPostRedisplay();
}

void myInit()
{
	glColor3f(0.0f, 0.0f, 0.0f);  // set color of stuff
	glShadeModel(GL_FLAT);   // or can be GL_SMOOTH

	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();

	// Produce the perspective projection
	gluPerspective(45.0f, 1.0, 1.0, 100.0);
	gluLookAt(0, 0, -10, 0, 0, 0, 0, 1, 0);

	glMatrixMode(GL_MODELVIEW);

	// Start animation
	myIdle(0);
	Idle();
	IdleB();
}


void myKeyboard(unsigned char Key, int x, int y)
{
```

```
	switch (Key) {
	case 'a': // start left animation
		Idle();
		break;
	case 's': // start right animation
		IdleB();
		break;
	case 27:  // Escape
		exit(-1);
	}
	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 wheels() {
	float th;
	glColor3f(1, 1, 1);
		glBegin(GL_LINE_LOOP);
		for (int i = 0; i < 360; i++)
		{
			th = i * (PI / 180);
			glVertex3f(1.62 + (0.5 * cos(th)),-1.4 + (0.5* sin(th)), 1.6);
		}
	glEnd();
	glBegin(GL_LINE_LOOP);
```

```
            for (int i = 0; i < 360; i++)
            {
                th = i * (PI / 180);
                glVertex3f(-1.62 + (0.5 * cos(th)),-1.4 + (0.5* sin(th)) , 1.6);
            }
            glEnd();
        glBegin(GL_LINE_LOOP);
            for (int i = 0; i < 360; i++)
            {
                th = i * (PI / 180);
                glVertex3f(1.62 + (0.5 * cos(th)), -1.4 + (0.5* sin(th)), -1.6);
            }
        glEnd();
        glBegin(GL_LINE_LOOP);
        for (int i = 0; i < 360; i++)
        {
            th = i * (PI / 180);
            glVertex3f(-1.62 + (0.5 * cos(th)),-1.4 + (0.5* sin(th)), -1.6);
        }
        glEnd();

}
void spinLine() {
        glColor3f(1, 1, 1);
        //the line in the wheel
        glPushMatrix();
        glTranslatef(1.62, -1.4, -1.60);
        glRotatef(180, 1, 0, 1);
        glScalef(0.5, 0.5, 0.5);
        glBegin(GL_LINES);
```

```
glVertex3dv(A);
glVertex3dv(B);
glEnd();
glPopMatrix();
//the line in the wheel
glPushMatrix();
glTranslatef(-1.62, -1.4, -1.60);
glRotatef(180, 1, 0, 1);
glScalef(0.5, 0.5, 0.5);
glBegin(GL_LINES);
glVertex3dv(A);
glVertex3dv(B);
glEnd();
glPopMatrix();
//the line in the wheel
glPushMatrix();
glTranslatef(1.62, -1.4, 1.60);
glRotatef(180, 1, 0, 1);
glScalef(0.5, 0.5, 0.5);
glBegin(GL_LINES);
glVertex3dv(A);
glVertex3dv(B);
glEnd();
glPopMatrix();
//the line in the wheel
glPushMatrix();
glTranslatef(-1.62, -1.4, 1.60);
glRotatef(180, 1, 0, 1);
glScalef(0.5, 0.5, 0.5);
glBegin(GL_LINES);
```

```
        glVertex3dv(A);
        glVertex3dv(B);
        glEnd();
        glPopMatrix();

        //line in steering wheel
        glPushMatrix();
        glTranslatef(1.8, -0.01, 0);
        glScalef(0.7, 0.7, 0.7);
        glRotatef(90, 0, 0, 1);
        glBegin(GL_LINES);
        glVertex3dv(Q);
        glVertex3dv(W);
        glEnd();

        glPopMatrix();

}
void steeringWheel() {

        glPushMatrix();
        glRotatef(-90, 1, 0.0, 0);
        glBegin(GL_LINE_LOOP | GL_LINES);

        glColor3f(1, 1, 1);

        //the shape and position of the steering wheel
        for (int i = 0; i < 360; i++)
        {
```

```
            float th = i * (PI / 180);
            glVertex3f((cos(th)*0.7)+1.8, (sin(th)*0.7), 0);
        }
        glEnd();
        glPopMatrix();
}

void wireCar() {

        glColor3f(1, 1, 1);
        //left
        glPushMatrix();
        glTranslatef(0, -1, 1);
        glScalef(1.1, 0.2, 0.01);
        glutWireCube(4);
        glPopMatrix();
        //right
        glPushMatrix();
        glTranslatef(0, -1, -1);
        glScaled(1.1, 0.2, 0.01);
        glutWireCube(4);
        glPopMatrix();
        //back
        glPushMatrix();
        glTranslatef(-2.2, -1, 0);
        glScaled(0.01, 0.2, 0.50);
        glutWireCube(4);
        glPopMatrix();
        //front
        glPushMatrix();
```

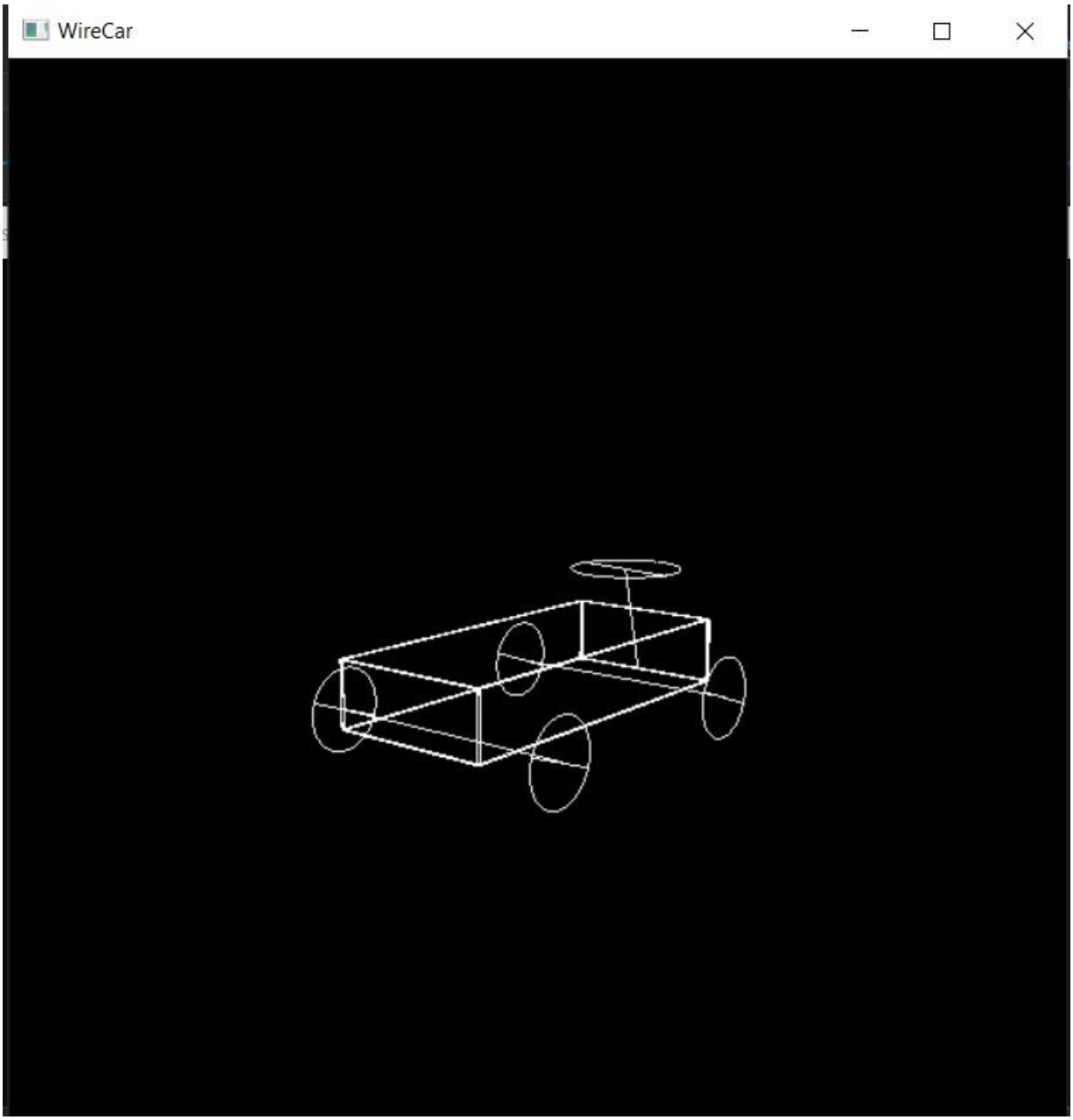
```
glTranslatef(2.2, -1, 0);
glScalef(0.01, 0.2, 0.50);
glutWireCube(4);
glPopMatrix();
// the line for wheels
glPushMatrix();
glTranslatef(-1.62, -1.40, -0.50);
glRotatef(90, 1, 0, 0);
glScaled(0.01, 0.7, -0.01);
glBegin(GL_LINES);
glVertex2f(3, 3);
glVertex2f(-1.62, -1.62);
glEnd();
glPopMatrix();
// the line between the wheels
glPushMatrix();
glTranslatef(1.62, -1.40, -0.50);
glRotatef(90, 1, 0, 0);
glScalef(0.01, 0.7, -0.01);
glBegin(GL_LINES);
glVertex2f(3, 3);
glVertex2f(-1.62, -1.62);
glEnd();
glPopMatrix();
//the line
glPushMatrix();
glTranslatef(2, -0.9, 0);
glRotatef(15, 0, 0, 1);
glScaled(0.01, 0.3, -0.01);
glBegin(GL_LINES);
```

```
	glVertex2f(3, 3);
	glVertex2f(-1.62, -1.62);
	glEnd();
	glPopMatrix();

	wheels();
	spinLine();
	steeringWheel();
}

void myDisplay(void)
{
	glClearColor(0.0, 0.0, 0.0, 0.0);
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);	// clear
screen

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();

	glTranslatef(sside, 0, -sdepth);
	glRotatef(-stheta, 1, 0, 0);
	glRotatef(sphi, 0, 1, 0);
	glTranslatef(sx, 0, -sy);
	wireCar();
	glutSwapBuffers();
}

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

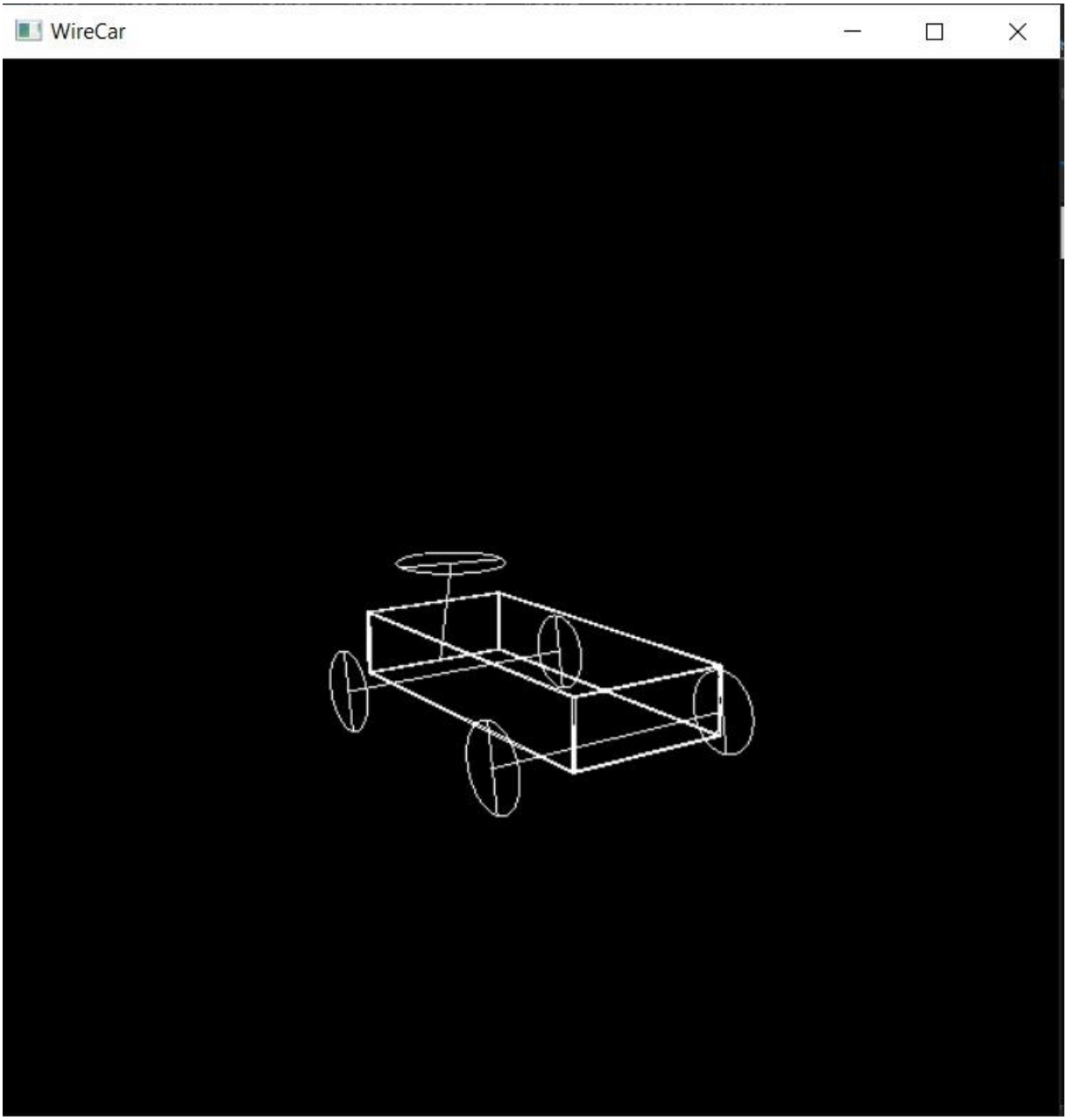
```
	mouseX = x; mouseY = y;
	mouse_left_click = ((button == GLUT_LEFT_BUTTON) && (state == 
GLUT_DOWN));
	mouse_middle_click = ((button == GLUT_MIDDLE_BUTTON) &&
		(state == GLUT_DOWN));
	mouse_right_click = ((button == GLUT_RIGHT_BUTTON) &&
		(state == GLUT_DOWN));
	glutPostRedisplay();
}

void myMouseMove(int x, int y) {
	// 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_RGB | GLUT_DOUBLE);
        glutInitWindowSize(screenWidth, screenHeight);
        glutInitWindowPosition(0, 0);
        glutCreateWindow("WireCar");
        glutKeyboardFunc(myKeyboard);
        glutReshapeFunc(myReshape);
        glutDisplayFunc(myDisplay);
        glutMouseFunc(myMouse);
        glutMotionFunc(myMouseMove);

        myInit();

        glutMainLoop();
}
```

Screenshots

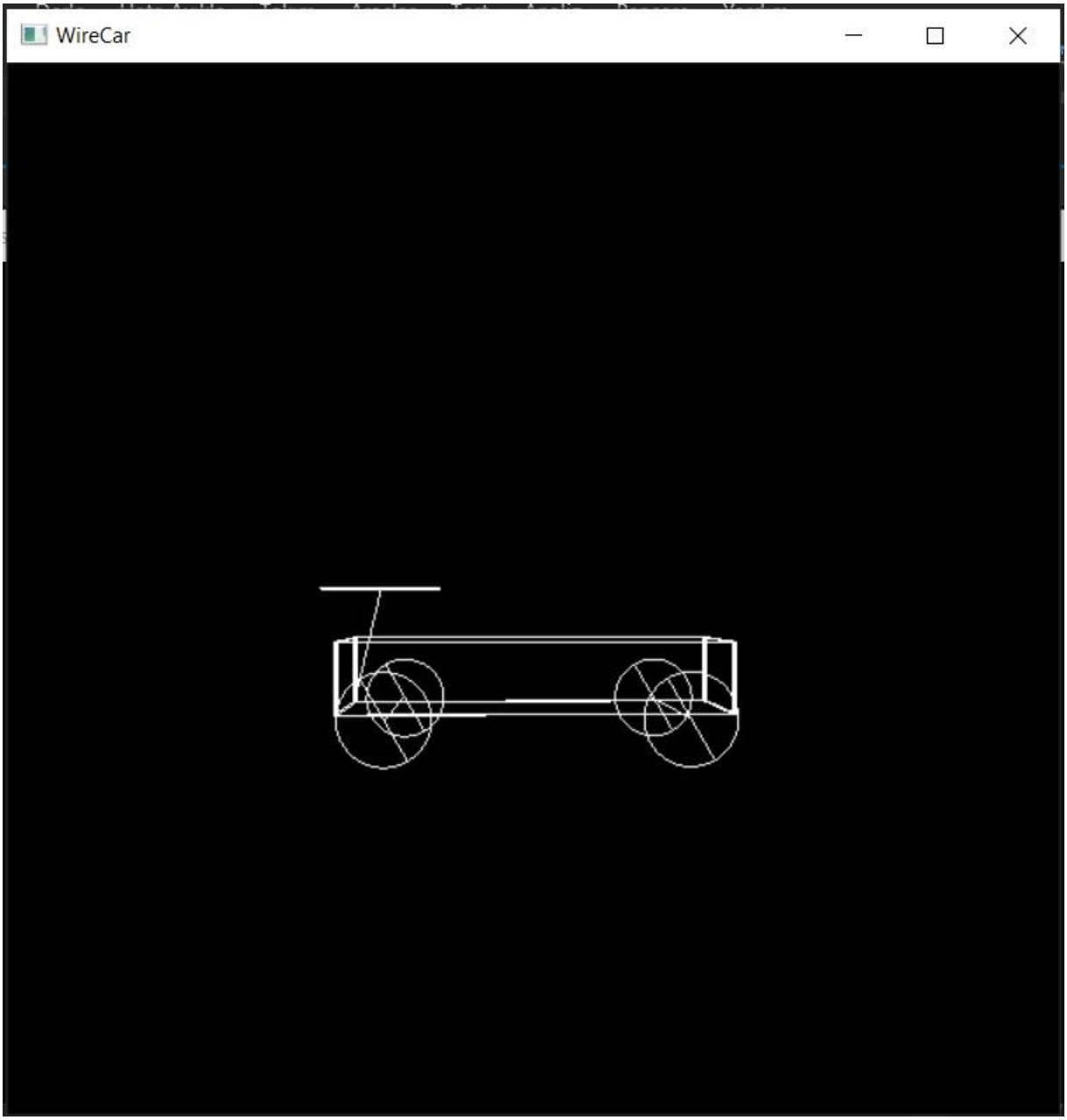


Picture 1.0: Right Angled View



Picture 1.1: Left Angled View

Picture 1.2: Top View



Picture 1.3: Side View

Picture 1.4: Backside View

Conclusion

The body, axles, and wheels of the car are formed and positioned to look like a whole car. The rotating stick has been successfully implemented to give the wheels the feeling of moving, but the rotation of the steering wheel and front wheels via key buttons couldn't be achieved.

References

Internet Sites

https://www.khronos.org/

https://www.opengl.org/

https://community.khronos.org/

https://stackoverflow.com/

Book PDFs

Computer Graphics with OpenGL (4th ed.) [Hearn, Baker & Carithers 2013]

Edward angel: Interactive computer graphics A TOP-DOWN Approach with OpenGL, 2nd edition, Addison-Wesley, 2000.

F.S. Hill, Jr.: computer graphics using OpenGL, 2nd edition, Pearson education, 2001.