T.C.

AYDIN ADNAN MENDERES UNIVERSITY

CSE 411 - COMPUTER GRAPHICS PROJECT REPORT

An Animated Wire Car

Ceylan Şentürk
161805015

161805015@stu.adu.edu.tr

Instructor: Samsun M. Başarıcı

January 2021

CONTENTS

INTRODUCTION

What is OpenGl?

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications.

OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, you must work through whatever windowing system controls the particular hardware you're using. Similarly, OpenGL does not provide high-level commands for describing models of three-dimensional objects. Such commands might allow you to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules. With OpenGL, you must build up your desired model from a small set of geometric primitives - points, lines, and polygons.

A sophisticated library that provides these features could certainly be built on top of OpenGL. The OpenGL Utility Library (GLU) provides many of the modeling features, such as quadric surfaces and NURBS (Non-Uniform Rational B-Splines) curves and surfaces. GLU is a standard part of every OpenGL implementation. Also, there is a higher-level, object-oriented toolkit, Open Inventor, which is built atop OpenGL, and is available separately for many implementations of OpenGL.

OPENGL FUNCTIONS

glutTimerFunc

Usage

```
void glutTimerFunc(unsigned int msecs,
              void (*func)(int value), value);
```

Description

glutTimerFunc registers the timer callback func to be triggered in at least msecs milliseconds. The value parameter to the timer callback will be the value of the value parameter to glutTimerFunc. Multiple timer callbacks at same or differing times may be registered simultaneously.

The number of milliseconds is a lower bound on the time before the callback is generated. GLUT attempts to deliver the timer callback as soon as possible after the expiration of the callback's time interval.

There is no support for canceling a registered callback. Instead, ignore a callback based on its value parameter when it is triggered.

Explanation

This function is used for animation in skeleton code. I used it when adjusting the speed of the steering wheel's rotation feature.

glutPostRedisplay

Usage

```
void glutPostRedisplay(void);
```

Description

Mark the normal plane of *current window* as needing to be redisplayed. The next iteration through glutMainLoop, the window's display callback will be called to redisplay the window's normal plane. Multiple calls to glutPostRedisplay before the next display callback opportunity generates only a single redisplay callback. glutPostRedisplay may be called within a window's display or overlay display callback to re-mark that window for redisplay.

Logically, normal plane damage notification for a window is treated as a glutPostRedisplay on the damaged window. Unlike damage reported by the window system, glutPostRedisplay will *not* set to true the normal plane's damaged status (returned by glutLayerGet(GLUT_NORMAL_DAMAGED).

Explanation

I used it to show animations.

gluPerspective

Usage

void **gluPerspective**(GLdouble *fovy*,

GLdouble *aspect*,

GLdouble *zNear*,

GLdouble *zFar*);

Description

gluPerspective specifies a viewing frustum into the world coordinate system. In general, the aspect ratio in gluPerspective should match the aspect ratio of the associated viewport. For example, aspect = 2.0 means the viewer's angle of view is twice as wide in *x* as it is in *y*. If the viewport is twice as wide as it is tall, it displays the image without distortion.

The matrix generated by gluPerspective is multiplied by the current matrix, just as if glMultMatrix were called with the generated matrix. To load the perspective matrix onto the current matrix stack instead, precede the call to gluPerspective with a call to glLoadIdentity.

gluLookAt

Usage

void **gluLookAt**(GLdouble *eyeX*,

GLdouble *eyeY*,

GLdouble *eyeZ*,

GLdouble *centerX*,

GLdouble *centerY*,

GLdouble *centerZ*,

GLdouble *upX*,

GLdouble *upY*,

GLdouble *upZ*);

Description

gluLookAt creates a viewing matrix derived from an eye point, a reference point indicating the center of the scene, and an *UP* vector.

The matrix maps the reference point to the negative *z* axis and the eye point to the origin. When a typical projection matrix is used, the center of the scene therefore maps to the center of the viewport. Similarly, the direction described by the *UP* vector projected onto the viewing plane is mapped to the positive *y* axis so that it points upward in the viewport. The *UP* vector must not be parallel to the line of sight from the eye point to the reference point.

glutKeyboardFunc

Usage

```
void glutKeyboardFunc(void (*func)(unsigned char key,
                                int x, int y));
```

func

The new keyboard callback function.

Description

glutKeyboardFunc sets the keyboard callback for the *current window*. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character. The state of modifier keys such as Shift cannot be determined directly; their only effect will be on the returned ASCII data. The x and y callback parameters indicate the mouse location in window relative coordinates when the key was pressed. When a new window is created, no keyboard callback is initially registered, and ASCII key strokes in the window are ignored. Passing NULL to glutKeyboardFunc disables the generation of keyboard callbacks.

During a keyboard callback, glutGetModifiers may be called to determine the state of modifier keys when the keystroke generating the callback occurred.

Explanation

I took advantage of this function to control the steering wheel rotation.

glutWireCube

Usage

```
void glutWireCube(GLdouble size);
```

Description

glutSolidCube and glutWireCube render a solid or wireframe cube respectively. The cube is centered at the modeling coordinates origin with sides of length size.

Explanation

I used the glutWireCube function to build the body of the car.

glRotate

Usage

```
void glRotated( GLdouble angle,
                GLdouble x,
                GLdouble y,
                GLdouble z);
```

Description

glRotate produces a rotation of *angle* degrees around the vector x y z . The current matrix (see glMatrixMode) is multiplied by a rotation matrix with the product replacing the current matrix.

If the matrix mode is either GL_MODELVIEW or GL_PROJECTION, all objects drawn after glRotate is called are rotated. Use glPushMatrix and glPopMatrix to save and restore the unrotated coordinate system.

Explanation

I used it when determining the slope of the shapes in the coordinate plane.

ALGORITHMS / TECHNIQUES

```
void myKeyboard(unsigned char Key, int x, int y)
{
       switch (Key) {
       case 'a': // start left animation
              animationA();
              break;
       case 's': // start right animation
              animationB();
              break;
       case 27:  // Escape
              exit(-1);
              break;
       }

}
```

I assigned the 'a' and 'b' keys with this function in order to direct the steering movements. The steering wheel turns left when the 'a' button is pressed, and turns right when the 'b' button is pressed. I created two functions "animationA" and "animationB" to provide these motions. Thanks to these functions, I solved the problem of intersecting rods inside the steering wheel turning in different directions. This shape, formed by the intersection of two different rods, was rotating in different directions due to the angle direction.

```
void left() {
       //left surface of the car
       glColor3f(1, 1, 1);
       glPushMatrix();
       glTranslatef(0, -1, 1);
       glScalef(1.1, 0.2, 0.01);
       glutWireCube(3);
       glPopMatrix();
}
```

I created four functions to create all the surfaces of the car, as seen in the example above. This example tool left surface glutWireCube (); creates with the help of function. The size of the created surface is glScaleF () and its position is glTranslatef (); It is determined by its function.

```
void wheel1()
{
       //shape of the wheel
       glBegin(GL_LINE_LOOP);
       glColor3f(1, 1, 1);
```

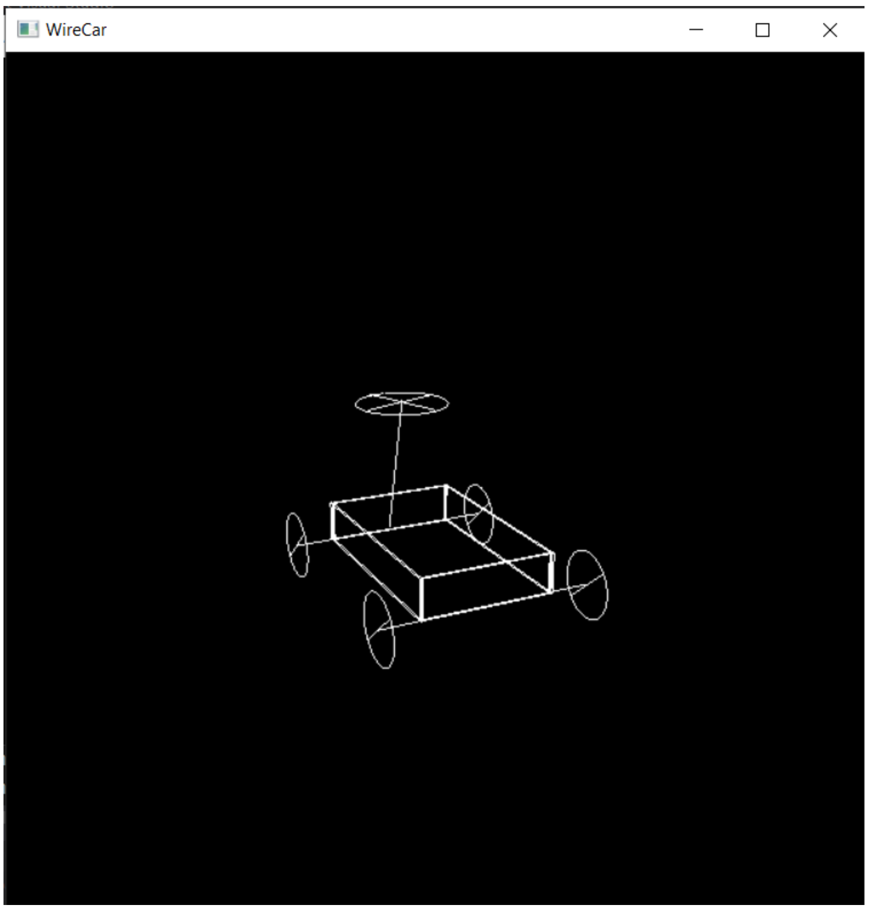
```
        for (int i = 0; i<360; i++)
        {
                float degInRad = i * (PI/180);
                glVertex3f((cos(degInRad)*0.5)-1.62, (sin(degInRad)*0.5)-1.3,-1.6);
        }
        glEnd();

}
```

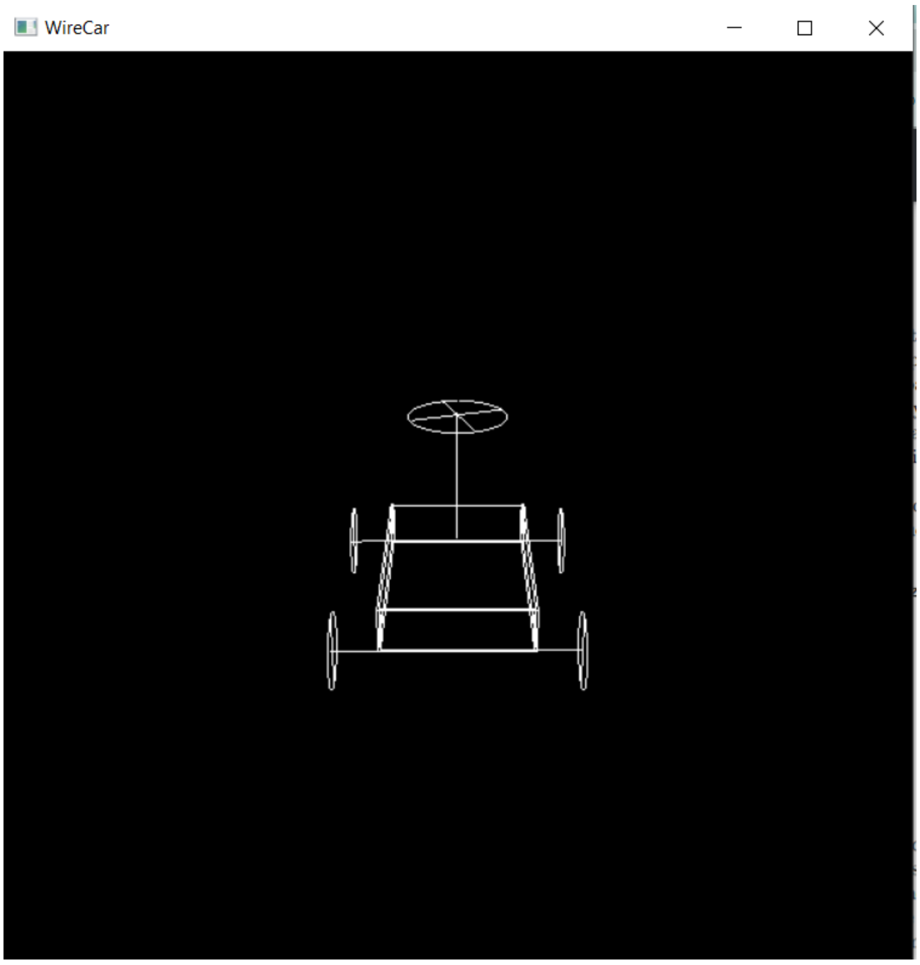
The shape of all wheels is created by the above function. Each wheel is a circle, and this circle is created with a for loop.

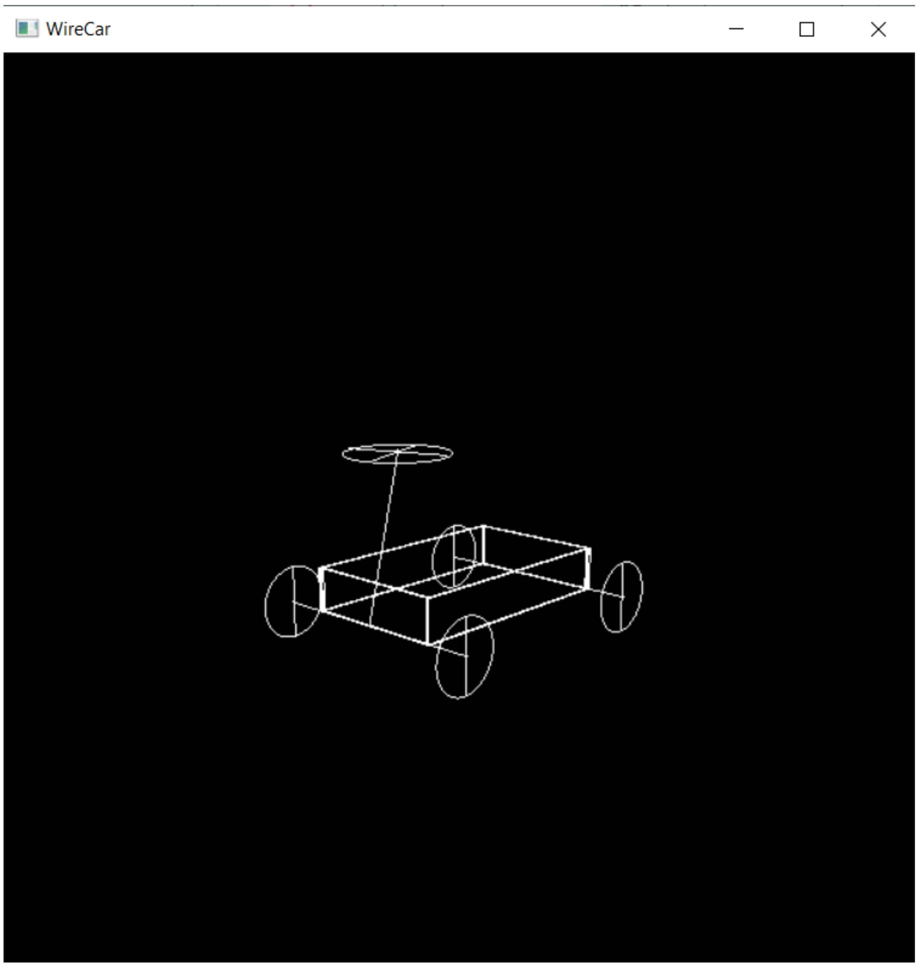
```
void steering()
{
        glPushMatrix();
        glRotatef(-rotAngle, 1, 0.0, 0);
        glBegin(GL_LINE_LOOP | GL_LINES);

        glColor3f(1, 1, 1);

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

                float degInRad = i * (PI / 180);
                glVertex3f( (cos(degInRad)*0.7) + 1.15, (sin(degInRad)*0.7)+0, 0.90);
        }
        glEnd();

        glPopMatrix();
```

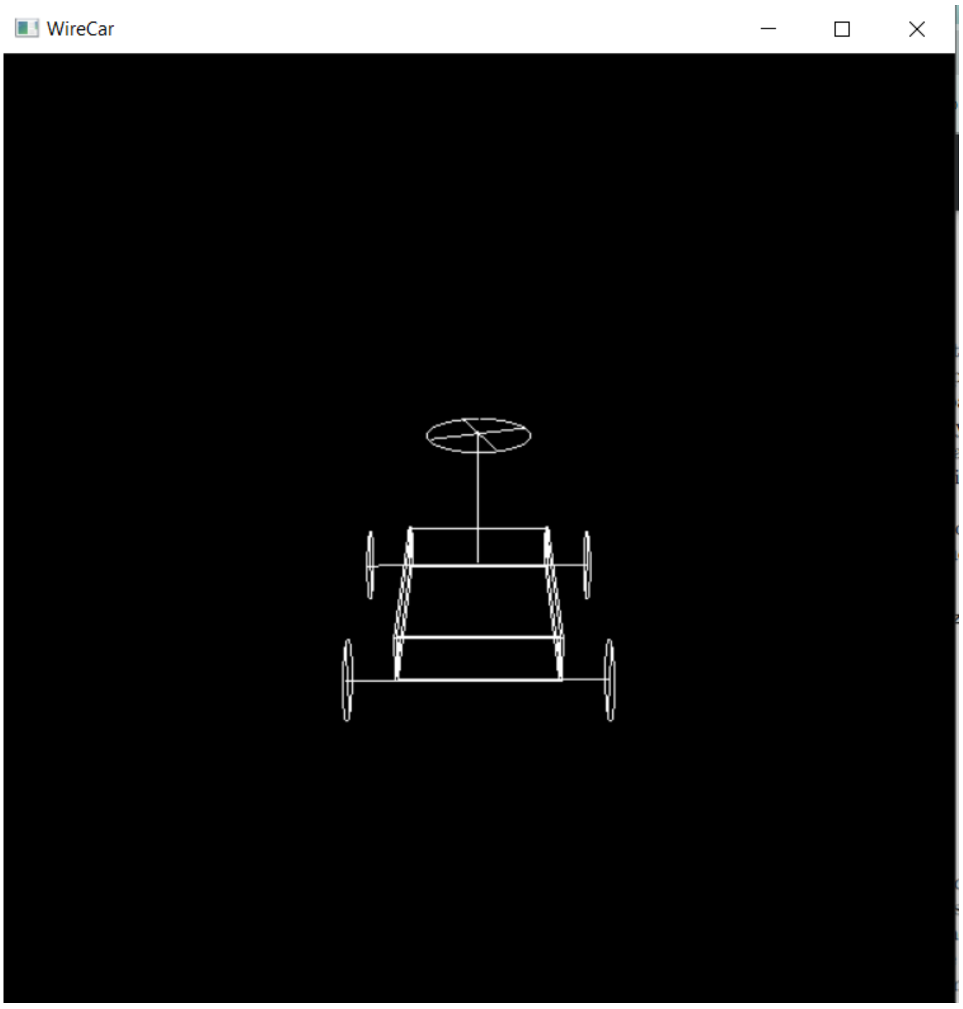
The way the steering wheel is created is the same as the wheel. The only difference is that the steering is horizontal while the wheels are perpendicular to the plane. We ensure that the steering wheel is horizontal with the glRotatef () function.

SNAPSHOTS




WireCar


WireCar


WireCar

CONCLUSION

I tried to pass this project process in the most efficient way by trying to use the information provided in the most efficient way, researching and experimenting. Making the wire cart requires both using ready-made functions and writing functions. The body, steering wheels and wheels of the car are in place. When we run the program, it gives the image that the vehicle's wheels are rotating and moving. The steering wheel can be controlled by the keys.

The car can be viewed 360 degrees. Everything seems to be in place when viewed from these angles. The shortcomings in this project are as follows:

1. The steering wheel has trouble turning left and right.
2. Wheels do not turn left and right.
3. The axle is not turning.

REFERENCES

Official OPENGL Documentation at https://www.opengl.org/documentation/

OpenGL Overview https://www.khronos.org/opengl/

OpenGL Programming Guide: The Official Guide to Learning OpenGLDave Shreiner.

Donald Hearn, M. Pauline Baker, Warren R. Carithers, “Computer Graphics with OpenGL, 4th Edition”; Pearson, 2011, ISBN: 978-0132484572 (HB)