```
#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 X[3] = \{ 0,0,0 \};
double Y[3] = \{ 0,0,0 \};
double C[3] = \{ 0,0,0 \};
double D[3] = \{ 0,0,0 \};
double alfa = 0;
      fx = 0, fy = 0, fz = 0;
float sphi = 0.0, stheta = 0.1;
float sside = 0, sdepth = -5;
float sx = 0, sy = 0;
bool mouse_left_click, mouse_middle_click, mouse_right_click;
int mouseX, mouseY;
float rotAngle = 90;
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 animationA(void) {
      //Animation for steering wheel
      alfa += 10;
      if (alfa > 360) alfa -= 360;
      X[1] = \sin(alfa*PI / 180);
      X[2] = cos(alfa*PI / 180);
      Y[1] = sin((alfa + 180)*PI / 180);
      Y[2] = cos((alfa + 180)*PI / 180);
      C[1] = cos(alfa*PI / 180);
      C[2] = \sin(alfa*PI / 180);
      D[1] = cos((alfa + 180)*PI / 180);
      D[2] = sin((alfa + 180)*PI / 180);
      glutTimerFunc(500, 0, 0);
      glutPostRedisplay();
```

}

```
void animationB(void) {
      //Animation for steering wheel
     alfa += 90;
      if (alfa < 360) alfa += 360;</pre>
     X[1] = cos(alfa*PI / 180);
     X[2] = \sin(alfa*PI / 180);
     Y[1] = cos((alfa + 180)*PI / 180);
     Y[2] = sin((alfa + 180)*PI / 180);
     C[1] = \sin(alfa*PI / 180);
     C[2] = cos(alfa*PI / 180);
     D[1] = \sin((alfa + 180)*PI / 180);
     D[2] = cos((alfa + 180)*PI / 180);
     glutTimerFunc(250, 0, 0);
      glutPostRedisplay();
void myInit()
{
      glColor3f(0.0f, 0.0f, 0.0f); // set color of stuff
                           // or can be GL_SMOOTH
     glShadeModel(GL_FLAT);
     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);
      animationA();
      animationB();
}
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;
      }
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 front() {
       //front surface of the car
       glColor3f(1, 1, 1);
       glPushMatrix();
       glTranslatef(-1.62, -1, 0);
       glScaled(0.01, 0.2, 0.65);
       glutWireCube(3);
       glPopMatrix();
       // the line between the wheels
       glColor3f(1, 1, 1);
       glPushMatrix();
       glTranslatef(-1.62, -1.30, -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 connecting the steering wheel to the car
       glColor3f(1, 1, 1);
       glPushMatrix();
       glTranslatef(1.50, -0.5, 0);
       glRotatef(15, 0, 0, 1);
       glScaled(0.01, 0.5, -0.01);
       glBegin(GL_LINES);
       glVertex2f(3, 3);
       glVertex2f(-1.62, -1.62);
       glEnd();
       glPopMatrix();
}
void back() {
       //back surface of the car
       glColor3f(1, 1, 1);
       glPushMatrix();
       glTranslatef(1.62, -1, 0);
       glScalef(0.01, 0.2, 0.65);
       glutWireCube(3);
       glPopMatrix();
       // the line between the wheels
       glColor3f(1, 1, 1);
       glPushMatrix();
       glTranslatef(1.62, -1.30, -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();
}
```

```
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();
}
void right()
       //right surface of the car
       glColor3f(1, 1, 1);
       glPushMatrix();
       glTranslatef(0, -1, -1);
       glScaled(1.1, 0.2, 0.01);
       glutWireCube(3);
       glPopMatrix();
}
void wheel1()
       //shape of the wheel
       glBegin(GL_LINE_LOOP);
       glColor3f(1, 1, 1);
       for (int i = 0; i<360; i++)</pre>
       {
              float degInRad = i * (PI/180);
              glVertex3f((cos(degInRad)*0.5)-1.62, (sin(degInRad)*0.5)-1.3,-1.6);
       }
       glEnd();
       //the line in the wheel
       glPushMatrix();
       glTranslatef(1.62, -1.30, -1.60);
       glRotatef(180, 1, 0, 1);
       glScalef(0.5, 0.5, 0.5);
       glColor3f(1, 1, 1);
       glBegin(GL LINES);
       glVertex3dv(A);
       glVertex3dv(B);
       glEnd();
       glPopMatrix();
}
void wheel2()
       //shape of the wheel
       glBegin(GL_LINE_LOOP);
       glColor3f(1, 1, 1);
       for (int i = 0; i < 360; i++)</pre>
              float degInRad = i * (PI / 180);
              glVertex3f((cos(degInRad)*0.5) + 1.62, (sin(degInRad)*0.5) - 1.3, -1.6);
       glEnd();
```

```
//the line in the wheel
      glPushMatrix();
      glTranslatef(-1.62, -1.30, -1.60);
      glRotatef(180, 1, 0, 1);
      glScalef(0.5, 0.5, 0.5);
      glColor3f(1, 1, 1);
      glBegin(GL_LINES);
      glVertex3dv(A);
      glVertex3dv(B);
      glEnd();
      glPopMatrix();
}
void wheel3()
      //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 line in the wheel
      glPushMatrix();
      glTranslatef(1.62, -1.30, 1.60);
      glRotatef(180, 1, 0, 1);
      glScalef(0.5, 0.5, 0.5);
      glColor3f(1, 1, 1);
      glBegin(GL_LINES);
      glVertex3dv(A);
      glVertex3dv(B);
      glEnd();
      glPopMatrix();
}
void wheel4()
       //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 line in the wheel
      glPushMatrix();
      glTranslatef(-1.62, -1.30, 1.60);
      glRotatef(180, 1, 0, 1);
      glScalef(0.5, 0.5, 0.5);
      glColor3f(1, 1, 1);
      glBegin(GL_LINES);
      glVertex3dv(A);
```

```
glVertex3dv(B);
      glEnd();
      glPopMatrix();
}
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();
      glColor3f(1, 1, 1);
      glPushMatrix();
      glTranslatef(1.15, 0.92, 0);
      glScalef(0.7, 0.7, 0.7);
      glRotatef(rotAngle, 0, 0, 1);
      glBegin(GL_LINES);
      glVertex3dv(X);
      glVertex3dv(Y);
      glEnd();
      glPopMatrix();
      //the lines of the steering wheel
      glColor3f(1, 1, 1);
      glPushMatrix();
      glTranslatef(1.15, 0.92, 0);
      glScalef(0.7, 0.7, 0.7);
      glRotatef(-rotAngle, 0, 0, 1);
      glBegin(GL LINES);
      glVertex3dv(C);
      glVertex3dv(D);
      glEnd();
      glPopMatrix();
void car()
      front();
      back();
       left();
      right();
      wheel1();
      wheel2();
      wheel3();
      wheel4();
      steering();
}
```

```
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);
      car();
      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;</pre>
      }
      // 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 | GLUT_DEPTH);
     glutInitWindowSize(screenWidth, screenHeight);
     glutInitWindowPosition(0, 0);
     glutCreateWindow("WireCar");
     glutKeyboardFunc(myKeyboard);
     glutReshapeFunc(myReshape);
     glutDisplayFunc(myDisplay);
     glutMouseFunc(myMouse);
      glutMotionFunc(myMouseMove);
     myInit();
      glutMainLoop();
}
```