#include"stdafx.h"

#include<GL/glut.h>

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#define PI 3.14159

#define WIN\_WIDTH 600

#define WIN\_HEIGHT 600

#define CYCLE\_LENGTH 3.3f

#define ROD\_RADIUS 0.05f

#define NUM\_SPOKES 20

#define SPOKE\_ANGLE 18

#define RADIUS\_WHEEL 1.0f

#define TUBE\_WIDTH 0.08f

#define RIGHT\_ROD 1.6f

#define RIGHT\_ANGLE 48.0f

#define MIDDLE\_ROD 1.7f

#define MIDDLE\_ANGLE 106.0f

#define BACK\_CONNECTOR 0.5f

#define LEFT\_ANGLE 50.0f

#define WHEEL\_OFFSET 0.11f

#define WHEEL\_LEN 1.1f

#define TOP\_LEN 1.5f

#define CRANK\_ROD 0.7f

#define CRANK\_RODS 1.12f

#define CRANK\_ANGLE 8.0f

#define HANDLE\_ROD 1.2f

#define FRONT\_INCLINE 70.0f

#define HANDLE\_LIMIT 70.0f

#define INC\_STEERING 2.0f

#define INC\_SPEED 0.01f

GLfloat pedalAngle, speed, steering;

GLfloat camx, camy, camz;

GLfloat anglex, angley, anglez;

int prevx, prevy;

GLenum Mouse;

GLfloat xpos, zpos, direction;

void ZCylinder(GLfloat radius, GLfloat length);

void XCylinder(GLfloat radius, GLfloat length);

void drawFrame(void);

void gear(GLfloat inner\_radius, GLfloat outer\_radius,GLfloat width, GLint teeth, GLfloat tooth\_depth);

void drawChain(void);

void drawPedals(void);

void drawTyre(void);

void drawSeat(void);

void help(void);

void init(void);

void reset(void);

void display(void);

void idle(void);

void updateScene(void);

void landmarks(void);

void special(int key, int x, int y);

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

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

void motion(int x, int y);

void reshape(int w, int h);

void glSetupFuncs(void);

GLfloat Abs(GLfloat);

GLfloat degrees(GLfloat);

GLfloat radians(GLfloat);

GLfloat angleSum(GLfloat, GLfloat);

void ZCylinder(GLfloat radius, GLfloat length)

{

GLUquadricObj \*cylinder;

cylinder = gluNewQuadric();

glPushMatrix();

glTranslatef(0.0f, 0.0f, 0.0f);

gluCylinder(cylinder, radius, radius, length, 15, 5);

glPopMatrix();

}

void XCylinder(GLfloat radius, GLfloat length)

{

glPushMatrix();

glRotatef(90.0f, 0.0f, 1.0f, 0.0f);

ZCylinder(radius, length);

glPopMatrix();

}

void updateScene()

{

GLfloat xDelta, zDelta;

GLfloat rotation;

GLfloat sin\_steering, cos\_steering;

if (-INC\_SPEED < speed && speed < INC\_SPEED) return;

if (speed < 0.0f)

pedalAngle = speed = 0.0f;

xDelta = speed\*cos(radians(direction + steering));

zDelta = speed\*sin(radians(direction + steering));

xpos += xDelta;

zpos -= zDelta;

pedalAngle = degrees(angleSum(radians(pedalAngle), speed / RADIUS\_WHEEL));

sin\_steering = sin(radians(steering));

cos\_steering = cos(radians(steering));

rotation = atan2(speed \* sin\_steering, CYCLE\_LENGTH + speed \* cos\_steering);

direction = degrees(angleSum(radians(direction), rotation));

}

GLfloat angleSum(GLfloat a, GLfloat b)

{

a += b;

if (a < 0) return a + 2 \* PI;

else if (a > 2 \* PI) return a - 2 \* PI;

else return a;

}

void drawFrame()

{

glColor3f(0.0f, 0.0f, 1.0f);

glPushMatrix();

glPushMatrix();

glColor3f(1.0f, 0.0f, 0.0f);

glPushMatrix();

glTranslatef(0.0f, 0.0f, 0.06f);

glRotatef(-2 \* pedalAngle, 0.0f, 0.0f, 1.0f);

gear(0.08f, 0.3f, 0.03f, 30, 0.03f);

glPopMatrix();

glColor3f(0.0f, 0.0f, 1.0f);

glTranslatef(0.0f, 0.0f, -0.2f);

ZCylinder(0.08f, 0.32f);

glPopMatrix();

glRotatef(RIGHT\_ANGLE, 0.0f, 0.0f, 1.0f);

XCylinder(ROD\_RADIUS, RIGHT\_ROD);

glRotatef(MIDDLE\_ANGLE - RIGHT\_ANGLE, 0.0f, 0.0f, 1.0f);

XCylinder(ROD\_RADIUS, MIDDLE\_ROD);

glColor3f(0.0f, 1.0f, 0.0f);

glTranslatef(MIDDLE\_ROD, 0.0f, 0.0f);

glRotatef(-MIDDLE\_ANGLE, 0.0f, 0.0f, 1.0f);

glScalef(0.3f, ROD\_RADIUS, 0.25f);

drawSeat();

glColor3f(0.0f, 0.0f, 1.0f);

glPopMatrix();

glPushMatrix();

glRotatef(-180.0f, 0.0f, 1.0f, 0.0f);

XCylinder(ROD\_RADIUS, BACK\_CONNECTOR);

glPushMatrix();

glTranslatef(0.5f, 0.0f, WHEEL\_OFFSET);

XCylinder(ROD\_RADIUS, RADIUS\_WHEEL + TUBE\_WIDTH);

glPopMatrix();

glPushMatrix();

glTranslatef(0.5f, 0.0f, -WHEEL\_OFFSET);

XCylinder(ROD\_RADIUS, RADIUS\_WHEEL + TUBE\_WIDTH);

glPopMatrix();

glPopMatrix();

glPushMatrix();

glTranslatef(-(BACK\_CONNECTOR + RADIUS\_WHEEL + TUBE\_WIDTH), 0.0f, 0.0f);

glPushMatrix();

glRotatef(-2 \* pedalAngle, 0.0f, 0.0f, 1.0f);

drawTyre();

glColor3f(1.0f, 0.0f, 0.0f);

gear(0.03f, 0.15f, 0.03f, 20, 0.03f);

glColor3f(0.0f, 0.0f, 1.0f);

glPopMatrix();

glRotatef(LEFT\_ANGLE, 0.0f, 0.0f, 1.0f);

glPushMatrix();

glTranslatef(0.0f, 0.0f, -WHEEL\_OFFSET);

XCylinder(ROD\_RADIUS, WHEEL\_LEN);

glPopMatrix();

glPushMatrix();

glTranslatef(0.0f, 0.0f, WHEEL\_OFFSET);

XCylinder(ROD\_RADIUS, WHEEL\_LEN);

glPopMatrix();

glTranslatef(WHEEL\_LEN, 0.0f, 0.0f);

XCylinder(ROD\_RADIUS, CRANK\_ROD);

glTranslatef(CRANK\_ROD, 0.0f, 0.0f);

glRotatef(-LEFT\_ANGLE, 0.0f, 0.0f, 1.0f);

XCylinder(ROD\_RADIUS, TOP\_LEN);

glTranslatef(TOP\_LEN, 0.0f, 0.0f);

glRotatef(-FRONT\_INCLINE, 0.0f, 0.0f, 1.0f);

glPushMatrix();

glTranslatef(-0.1f, 0.0f, 0.0f);

XCylinder(ROD\_RADIUS, 0.45f);

glPopMatrix();

glPushMatrix();

glRotatef(-steering, 1.0f, 0.0f, 0.0f);

glTranslatef(-0.3f, 0.0f, 0.0f);

glPushMatrix();

glRotatef(FRONT\_INCLINE, 0.0f, 0.0f, 1.0f);

glPushMatrix();

glTranslatef(0.0f, 0.0f, -HANDLE\_ROD / 2);

ZCylinder(ROD\_RADIUS, HANDLE\_ROD);

glPopMatrix();

glPushMatrix();

glColor3f(0.0f, 1.0f, 0.0f);

glTranslatef(0.0f, 0.0f, -HANDLE\_ROD / 2);

ZCylinder(0.07f, HANDLE\_ROD / 4);

glTranslatef(0.0f, 0.0f, HANDLE\_ROD \* 3 / 4);

ZCylinder(0.07f, HANDLE\_ROD / 4);

glColor3f(0.0f, 0.0f, 1.0f);

glPopMatrix();

glPopMatrix();

glPushMatrix();

XCylinder(ROD\_RADIUS, CRANK\_ROD);

glTranslatef(CRANK\_ROD, 0.0f, 0.0f);

glRotatef(CRANK\_ANGLE, 0.0f, 0.0f, 1.0f);

glPushMatrix();

glTranslatef(0.0f, 0.0f, WHEEL\_OFFSET);

XCylinder(ROD\_RADIUS, CRANK\_RODS);

glPopMatrix();

glPushMatrix();

glTranslatef(0.0f, 0.0f, -WHEEL\_OFFSET);

XCylinder(ROD\_RADIUS, CRANK\_RODS);

glPopMatrix();

glTranslatef(CRANK\_RODS, 0.0f, 0.0f);

glRotatef(-2 \* pedalAngle, 0.0f, 0.0f, 1.0f);

drawTyre();

glPopMatrix();

glPopMatrix();

glPopMatrix();

}

void gear(GLfloat inner\_radius, GLfloat outer\_radius, GLfloat width,

GLint teeth, GLfloat tooth\_depth)

{

GLint i;

GLfloat r0, r1, r2;

GLfloat angle, da;

GLfloat u, v, len;

const double pi = 3.14159264;

r0 = inner\_radius;

r1 = outer\_radius - tooth\_depth / 2.0;

r2 = outer\_radius + tooth\_depth / 2.0;

da = 2.0\*pi / teeth / 4.0;

glShadeModel(GL\_FLAT);

glNormal3f(0.0, 0.0, 1.0);

glBegin(GL\_QUAD\_STRIP);

for (i = 0; i <= teeth; i++) {

angle = i \* 2.0\*pi / teeth;

glVertex3f(r0\*cos(angle), r0\*sin(angle), width\*0.5);

glVertex3f(r1\*cos(angle), r1\*sin(angle), width\*0.5);

glVertex3f(r0\*cos(angle), r0\*sin(angle), width\*0.5);

glVertex3f(r1\*cos(angle + 3 \* da), r1\*sin(angle + 3 \* da), width\*0.5);

}

glEnd();

glBegin(GL\_QUADS);

da = 2.0\*pi / teeth / 4.0;

for (i = 0; i<teeth; i++) {

angle = i \* 2.0\*pi / teeth;

glVertex3f(r1\*cos(angle), r1\*sin(angle), width\*0.5);

glVertex3f(r2\*cos(angle + da), r2\*sin(angle + da), width\*0.5);

glVertex3f(r2\*cos(angle + 2 \* da), r2\*sin(angle + 2 \* da), width\*0.5);

glVertex3f(r1\*cos(angle + 3 \* da), r1\*sin(angle + 3 \* da), width\*0.5);

}

glEnd();

glNormal3f(0.0, 0.0, -1.0);

glBegin(GL\_QUAD\_STRIP);

for (i = 0; i <= teeth; i++) {

angle = i \* 2.0\*pi / teeth;

glVertex3f(r1\*cos(angle), r1\*sin(angle), -width\*0.5);

glVertex3f(r0\*cos(angle), r0\*sin(angle), -width\*0.5);

glVertex3f(r1\*cos(angle + 3 \* da), r1\*sin(angle + 3 \* da), -width\*0.5);

glVertex3f(r0\*cos(angle), r0\*sin(angle), -width\*0.5);

}

glEnd();

glBegin(GL\_QUADS);

da = 2.0\*pi / teeth / 4.0;

for (i = 0; i<teeth; i++) {

angle = i \* 2.0\*pi / teeth;

glVertex3f(r1\*cos(angle + 3 \* da), r1\*sin(angle + 3 \* da), -width\*0.5);

glVertex3f(r2\*cos(angle + 2 \* da), r2\*sin(angle + 2 \* da), -width\*0.5);

glVertex3f(r2\*cos(angle + da), r2\*sin(angle + da), -width\*0.5);

glVertex3f(r1\*cos(angle), r1\*sin(angle), -width\*0.5);

}

glEnd();

glBegin(GL\_QUAD\_STRIP);

for (i = 0; i<teeth; i++) {

angle = i \* 2.0\*pi / teeth;

glVertex3f(r1\*cos(angle), r1\*sin(angle), width\*0.5);

glVertex3f(r1\*cos(angle), r1\*sin(angle), -width\*0.5);

u = r2\*cos(angle + da) - r1\*cos(angle);

v = r2\*sin(angle + da) - r1\*sin(angle);

len = sqrt(u\*u + v\*v);

u /= len;

v /= len;

glNormal3f(v, -u, 0.0);

glVertex3f(r2\*cos(angle + da), r2\*sin(angle + da), width\*0.5);

glVertex3f(r2\*cos(angle + da), r2\*sin(angle + da), -width\*0.5);

glNormal3f(cos(angle), sin(angle), 0.0);

glVertex3f(r2\*cos(angle + 2 \* da), r2\*sin(angle + 2 \* da), width\*0.5);

glVertex3f(r2\*cos(angle + 2 \* da), r2\*sin(angle + 2 \* da), -width\*0.5);

u = r1\*cos(angle + 3 \* da) - r2\*cos(angle + 2 \* da);

v = r1\*sin(angle + 3 \* da) - r2\*sin(angle + 2 \* da);

glNormal3f(v, -u, 0.0);

glVertex3f(r1\*cos(angle + 3 \* da), r1\*sin(angle + 3 \* da), width\*0.5);

glVertex3f(r1\*cos(angle + 3 \* da), r1\*sin(angle + 3 \* da), -width\*0.5);

glNormal3f(cos(angle), sin(angle), 0.0);

}

glVertex3f(r1\*cos(0.0), r1\*sin(0.0), width\*0.5);

glVertex3f(r1\*cos(0.0), r1\*sin(0.0), -width\*0.5);

glEnd();

glShadeModel(GL\_SMOOTH);

glBegin(GL\_QUAD\_STRIP);

for (i = 0; i <= teeth; i++) {

angle = i \* 2.0\*pi / teeth;

glNormal3f(-cos(angle), -sin(angle), 0.0);

glVertex3f(r0\*cos(angle), r0\*sin(angle), -width\*0.5);

glVertex3f(r0\*cos(angle), r0\*sin(angle), width\*0.5);

}

glEnd();

}

void drawChain()

{

GLfloat depth;

static int mode = 0;

glColor3f(0.0f, 1.0f, 0.0f);

glEnable(GL\_LINE\_STIPPLE);

mode = (mode + 1) % 2;

if (mode == 0 && speed>0)

glLineStipple(1, 0x1c47);

else if (mode == 1 && speed>0)

glLineStipple(1, 0x00FF);

glBegin(GL\_LINES);

for (depth = 0.06f; depth <= 0.12f; depth += 0.01f)

{

glVertex3f(-1.6f, 0.15f, ROD\_RADIUS);

glVertex3f(0.0f, 0.3f, depth);

glVertex3f(-1.6f, -0.15f, ROD\_RADIUS);

glVertex3f(0.0f, -0.3f, depth);

}

glEnd();

glDisable(GL\_LINE\_STIPPLE);

}

void drawSeat()

{

glBegin(GL\_POLYGON);

glVertex3f(-0.1f, 1.0f, -0.5f);

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

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

glVertex3f(-0.1f, 1.0f, 0.5f);

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

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

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

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

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(-0.1f, -1.0f, -0.5f);

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

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

glVertex3f(-0.1f, -1.0f, 0.5f);

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

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

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

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

glEnd();

glBegin(GL\_QUADS);

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

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

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

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

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

glVertex3f(-0.1f, 1.0f, 0.5f);

glVertex3f(-0.1f, -1.0f, 0.5f);

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

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

glVertex3f(-0.1f, 1.0f, -0.5f);

glVertex3f(-0.1f, -1.0f, -0.5f);

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

glVertex3f(-0.1f, 1.0f, 0.5f);

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

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

glVertex3f(-0.1f, -1.0f, 0.5f);

glVertex3f(-0.1f, 1.0f, -0.5f);

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

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

glVertex3f(-0.1f, -1.0f, -0.5f);

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

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

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

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

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

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

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

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

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

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

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

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

glEnd();

}

void drawPedals()

{

glColor3f(0.0f, 1.0f, 0.0f);

glPushMatrix();

glTranslatef(0.0f, 0.0f, 0.105f);

glRotatef(-pedalAngle, 0.0f, 0.0f, 1.0f);

glTranslatef(0.25f, 0.0f, 0.0f);

glPushMatrix();

glScalef(0.5f, 0.1f, 0.1f);

glutSolidCube(1.0f);

glPopMatrix();

glPushMatrix();

glTranslatef(0.25f, 0.0f, 0.15f);

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

glScalef(0.2f, 0.02f, 0.3f);

glutSolidCube(1.0f);

glPopMatrix();

glPopMatrix();

glPushMatrix();

glTranslatef(0.0f, 0.0f, -0.105f);

glRotatef(180.0f - pedalAngle, 0.0f, 0.0f, 1.0f);

glTranslatef(0.25f, 0.0f, 0.0f);

glPushMatrix();

glScalef(0.5f, 0.1f, 0.1f);

glutSolidCube(1.0f);

glPopMatrix();

glPushMatrix();

glTranslatef(0.25f, 0.0f, -0.15f);

glRotatef(pedalAngle - 180.0f, 0.0f, 0.0f, 1.0f);

glScalef(0.2f, 0.02f, 0.3f);

glutSolidCube(1.0f);

glPopMatrix();

glPopMatrix();

glColor3f(1.0f, 0.0f, 0.0f);

}

void drawTyre(void)

{

int i;

glColor3f(1.0f, 1.0f, 1.0f);

glutSolidTorus(0.06f, 0.92f, 4, 30);

glColor3f(1.0f, 1.0f, 0.5f);

glPushMatrix();

glTranslatef(0.0f, 0.0f, -0.06f);

ZCylinder(0.02f, 0.12f);

glPopMatrix();

glutSolidTorus(0.02f, 0.02f, 3, 20);

glColor3f(1.0f, 1.0f, 1.0f);

for (i = 0; i<NUM\_SPOKES; ++i)

{

glPushMatrix();

glRotatef(i\*SPOKE\_ANGLE, 0.0f, 0.0f, 1.0f);

glBegin(GL\_LINES);

glVertex3f(0.0f, 0.02f, 0.0f);

glVertex3f(0.0f, 0.86f, 0.0f);

glEnd();

glPopMatrix();

}

glColor3f(0.0f, 0.0f, 0.0f);

glutSolidTorus(TUBE\_WIDTH, RADIUS\_WHEEL, 10, 30);

glColor3f(1.0f, 0.0f, 0.0f);

}

void init()

{

GLfloat mat\_specular[] = { 1.0,1.0,1.0,1.0 };

GLfloat mat\_shininess[] = { 100.0 };

GLfloat light\_directional[] = { 1.0,1.0,1.0,1.0 };

GLfloat light\_positional[] = { 1.0,1.0,1.0,0.0 };

GLfloat light\_diffuse[] = { 1.0,1.0,1.0 };

reset();

glShadeModel(GL\_SMOOTH);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_directional);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_diffuse);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);

glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glColorMaterial(GL\_FRONT, GL\_DIFFUSE);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_DEPTH\_TEST);

}

void landmarks(void)

{

GLfloat i;

glColor3f(1.0f, 0.0f, 1.0f);

glBegin(GL\_LINES);

for (i = -100.0f; i<100.0f; i += 1.0f)

{

glVertex3f(-100.0f, -RADIUS\_WHEEL, i);

glVertex3f(100.0f, -RADIUS\_WHEEL, i);

glVertex3f(i, -RADIUS\_WHEEL, -100.0f);

glVertex3f(i, -RADIUS\_WHEEL, 100.0f);

}

glEnd();

}

void display(void)

{

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

glEnable(GL\_NORMALIZE);

glPushMatrix();

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

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

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

landmarks();

glPushMatrix();

glTranslatef(xpos, 0.0f, zpos);

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

drawFrame();

drawChain();

drawPedals();

glPopMatrix();

glPopMatrix();

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(camx, camy, camz, camx, 0.0, 0.0, 0.0, 1.0, 0.0);

glutSwapBuffers();

}

GLfloat Abs(GLfloat a)

{

if (a < 0.0f)

return -a;

else

return a;

}

GLfloat degrees(GLfloat a)

{

return a\*180.0f / PI;

}

GLfloat radians(GLfloat a)

{

return a\*PI / 180.0f;

}

void idle(void)

{

updateScene();

glutPostRedisplay();

}

void special(int key, int x, int y)

{

switch (key)

{

case GLUT\_KEY\_UP:

camz -= 0.1f;

break;

case GLUT\_KEY\_DOWN:

camz += 0.1f;

break;

case GLUT\_KEY\_LEFT:

camx -= 0.1f;

break;

case GLUT\_KEY\_RIGHT:

camx += 0.1f;

break;

}

glutPostRedisplay();

}

void reset()

{

anglex = angley = anglez = 0.0f;

pedalAngle = steering = 0.0f;

Mouse = GLUT\_UP;

pedalAngle = speed = steering = 0.0f;

camx = camy = 0.0f;

camz = 5.0f;

xpos = zpos = 0.0f;

direction = 0.0f;

}

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

{

GLfloat r = 0.0f;

switch (key)

{

case 'r':

case 'R':

reset();

break;

case 'a':

if (steering < HANDLE\_LIMIT)

steering += INC\_STEERING;

break;

case 'd':

if (steering > -HANDLE\_LIMIT)

steering -= INC\_STEERING;

break;

case 'w':

speed += INC\_SPEED;

break;

case 's':

speed -= INC\_SPEED;

break;

case 27:

exit(1);

}

pedalAngle += speed;

if (speed < 0.0f)

speed = 0.0f;

if (pedalAngle < 0.0f)

pedalAngle = 0.0f;

if (pedalAngle >= 360.0f)

pedalAngle -= 360.0f;

glutPostRedisplay();

}

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

{

switch (button)

{

case GLUT\_LEFT\_BUTTON:

if (state == GLUT\_DOWN)

{

Mouse = GLUT\_DOWN;

prevx = x;

prevy = y;

}

if (state == GLUT\_UP)

{

Mouse = GLUT\_UP;

}

break;

case GLUT\_RIGHT\_BUTTON:

break;

}

glutPostRedisplay();

}

void motion(int x, int y)

{

if (Mouse == GLUT\_DOWN)

{

int deltax, deltay;

deltax = prevx - x;

deltay = prevy - y;

anglex += 0.5\*deltax;

angley += 0.5\*deltay;

if (deltax != 0 && deltay != 0)

anglez += 0.5\*sqrt(deltax\*deltax + deltay\*deltay);

if (anglex < 0)

anglex += 360.0;

if (angley < 0)

angley += 360.0;

if (anglez < 0)

anglez += 360.0;

if (anglex > 360.0)

anglex -= 360.0;

if (angley > 360.0)

angley -= 360.0;

if (anglez > 360.0)

anglez -= 360.0;

}

else

{

Mouse = GLUT\_UP;

}

prevx = x;

prevy = y;

glutPostRedisplay();

}

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60.0, (GLfloat)w / (GLfloat)h, 0.1, 100.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(camx, camy, camz, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

}

void glSetupFuncs(void)

{

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutIdleFunc(idle);

glutSpecialFunc(special);

glutKeyboardFunc(keyboard);

glutMouseFunc(mouse);

glutMotionFunc(motion);

glutSetCursor(GLUT\_CURSOR\_CROSSHAIR);

}

void help(void)

{

printf("Hierarchical 3D Model of a Bicycle\n");

printf("TCS2111- Computer Graphics\n");

printf("Group Project\n\n");

printf("'w' to increase the speed\n");

printf("'s' to decrease the speed\n");

printf("'d' to rotate the handle in clockwise direction\n");

printf("'a' to rotate the handle in anti-clockwise direction\n");

printf("'r' or 'R' to reset the scene\n");

printf("Arrow keys to move the camera\n");

printf("Mouse to move the scene\n");

}

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

{

help();

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowPosition(100, 100);

glutInitWindowSize(WIN\_WIDTH, WIN\_HEIGHT);

glutCreateWindow("BiCycle");

init();

glSetupFuncs();

glutMainLoop();

}