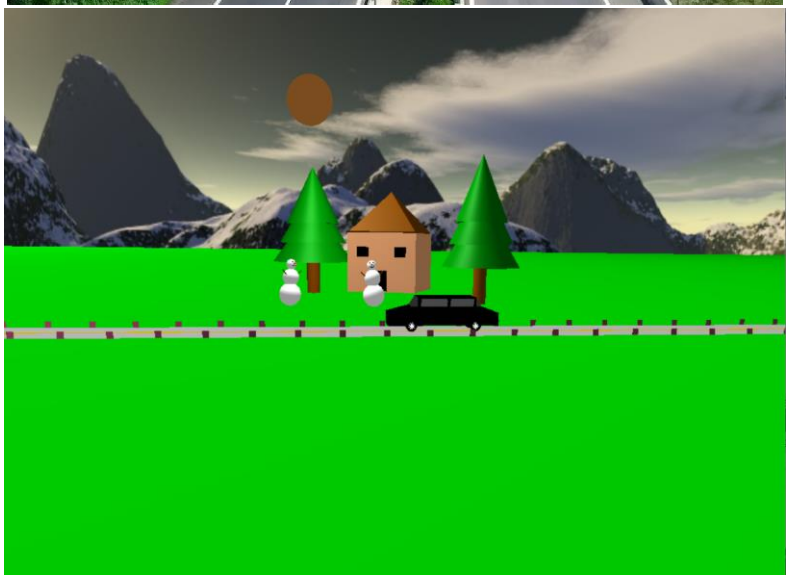
Computer Graphics Project

Full Program Report

Qichen An 20032255

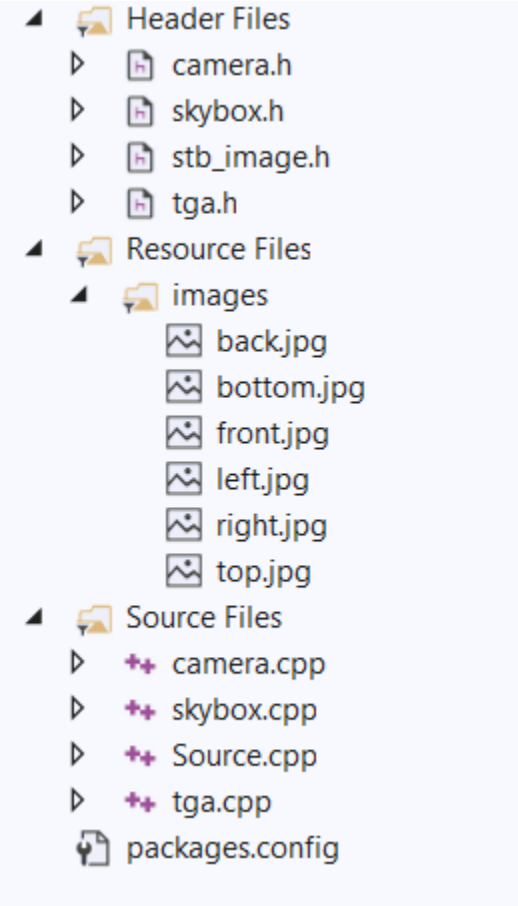
Since the beginning of the project, some creative ideas have been contributed called ‘A highway knifes through the jungle’. Based on the inspiration of the following picture, there is a scenario about a road and a car, which are simulated as the idea of this project. Roadside trees and houses are also be added to the scene. Moreover, enlightened by some given example 3D scene screenshots, snow scenes are added, snowflakes floating in the air and snow on the ground included. At the end of the project, all these ideas are basically completed as shown below.



How to use the program:

The program should be run with a package named “nupengl”.

After installing this package, let me introduce the role of following files.



Main function is in the file “source.cpp” and most of other architectures are in the same file, for example, 3D object creation, animation and keyboard functions. “camera.h” and “camera.cpp” are from projects of labs and are used for controlling the change of different perspectives. “stb_image.h”, “skybox.h” and “skybox.cpp” are utilized for design of skybox of the full scene. Moreover, “tga.h” and “tga.cpp” are designed for textures of sun and moon.

All the images which need to be imported, such as the front side image of skybox, will be put in a folder named “images”.

Requirements:

The following is a detailed explanation of requirements with the code and screenshots.

3D objects creation

First of all, the 3D objects of car, road, house, tree, moon and snowman and so on are created.


```
//method to create a house
void house(void) { ... }

//method to create a tree
void tree(void) { ... }

//method to create moon/ sun
void moonOrSun(void) { ... }

//method to create snowman
void snowman(void) { ... }

//method to create car
GLvoid DrawCar() { ... }

//method to create road
GLvoid DrawRoad() { ... }

//method to create overall scene
GLvoid DrawGLScene() { ... }
```

a. Car creation:

The main work of the car creation is to drawing cube.

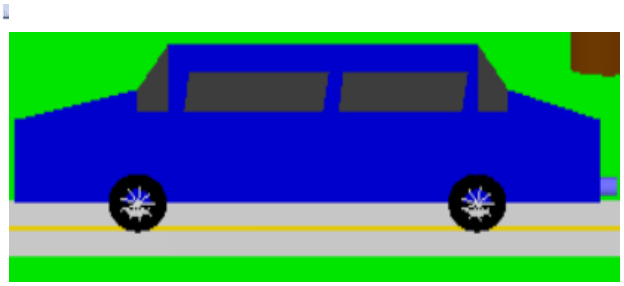
```
GLvoid DrawCar() {
    glPushMatrix();

    glTranslatef(0, -2.0, 0);
    glBegin(GL_QUADS);

/* top of cube*/
//************************FRONT BODY****
    glColor3f(r, g, b);
    glVertex3f(0.2, 0.4, 0.6);
    glVertex3f(0.6, 0.5, 0.6);
    glVertex3f(0.6, 0.5, 0.2);
    glVertex3f(0.2, 0.4, 0.2);

    /* bottom of cube*/
    glVertex3f(0.2, 0.4, 0.6);
    glVertex3f(0.6, 0.2, 0.6);
    glVertex3f(0.6, 0.2, 0.2);
    glVertex3f(0.2, 0.2, 0.2);

    /* front of cube*/
    glVertex3f(0.2, 0.2, 0.6);
    glVertex3f(0.2, 0.4, 0.6);
    glVertex3f(0.2, 0.4, 0.2);
    glVertex3f(0.2, 0.2, 0.2);
```

```
//*************************back g
glColor3f(r, g, b);            /* Se
glVertex3f(1.8, 0.5, 0.6);
glVertex3f(1.8, 0.5, 0.2);
glVertex3f(2.1, 0.4, 0.2);
glVertex3f(2.1, 0.4, 0.6);

/* bottom of cube*/
glVertex3f(2.1, 0.2, 0.6);
glVertex3f(2.1, 0.2, 0.2);
glVertex3f(1.8, 0.2, 0.6);
glVertex3f(1.8, 0.2, 0.6);

/* back of cube.*/
glVertex3f(2.1, 0.4, 0.6);
glVertex3f(2.1, 0.4, 0.2);
glVertex3f(2.1, 0.2, 0.2);
glVertex3f(2.1, 0.2, 0.6);

/* left of cube*/
glVertex3f(1.8, 0.2, 0.2);
glVertex3f(1.8, 0.5, 0.2);
glVertex3f(2.1, 0.4, 0.2);
glVertex3f(2.1, 0.2, 0.2);
```

```
    glBegin(GL_QUADS);

    /* top of cube*/
    glColor3f(0.3, 0.3, 0.3);
    glVertex3f(0.6, 0.5, 0.6);
    glVertex3f(0.6, 0.5, 0.2);
    glVertex3f(0.7, 0.65, 0.2);
    glVertex3f(0.7, 0.65, 0.6);

    glVertex3f(1.7, 0.65, .6);
    glVertex3f(1.7, 0.65, 0.2);
    glVertex3f(1.8, 0.5, 0.2);
    glVertex3f(1.8, 0.5, 0.6);

    glEnd();

    //
    glBegin(GL_TRIANGLES);

      /* top of cube*/
    glColor3f(0.3, 0.3, 0.3);
    glVertex3f(0.6, 0.5, 0.6);
    glVertex3f(0.7, 0.65, 0.6);
    glVertex3f(0.7, 0.5, 0.6);
```

However, the creation of wheel is remarkable. In the code, the variation ‘angle’ is related to the interaction to move the car. As a result, when moving the car, it is obvious that the wheels are moving and rotating.

```
//*******************WHEEL

glColor3f(0.7, 0.7, 0.7);
glPushMatrix();
glBegin(GL_LINE_STRIP);
for (theta = 0; theta < 360; theta = theta + 40)
{
    glVertex3f(0.6, 0.2, 0.62);
    glVertex3f(0.6 + (0.08 * (cos(((theta + angle) * 3.14) / 180))), 0.2 + (0.08 * (sin(((theta + angle) * 3.14) / 180))), 0.62);
}
glEnd();

glBegin(GL_LINE_STRIP);
for (theta = 0; theta < 360; theta = theta + 40)
{
    glVertex3f(0.6, 0.2, 0.18);
    glVertex3f(0.6 + (0.08 * (cos(((theta + angle) * 3.14) / 180))), 0.2 + (0.08 * (sin(((theta + angle) * 3.14) / 180))), 0.18);
}
glEnd();

glBegin(GL_LINE_STRIP);
for (theta = 0; theta < 360; theta = theta + 40)
{
    glVertex3f(1.7, 0.2, 0.18);
    glVertex3f(1.7 + (0.08 * (cos(((theta + angle) * 3.14) / 180))), 0.2 + (0.08 * (sin(((theta + angle) * 3.14) / 180))), 0.18);
}
glEnd();
```

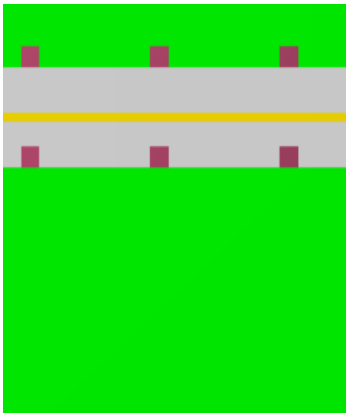


b. Road and background creation:

The idea of the creation of the ground and the road is to create a small cuboid with small height. Along the road, wheel flags are set up at fixed distances.

```
GLvoid DrawRoad() {
    //*****************************road
    glPushMatrix();
    glTranslatef(0, -2.0, 0);

    glBegin(GL_QUADS);

    glPushMatrix();
    glTranslatef(xw, 0, 0);
    glColor3f(0, 1, 0);
    glVertex3f(-100, 0.1, -100);
    glVertex3f(-100, 0.1, 0);          //
    glVertex3f(100, 0.1, 0);
    glVertex3f(100, 0.1, -100);

    glColor3f(0.7, 0.7, 0.7);
    glVertex3f(-100, 0.1, 0);
    glVertex3f(-100, 0.1, 0.45);
    glVertex3f(100, 0.1, 0.45);
    glVertex3f(100, 0.1, 0);

    glColor3f(1.0, 0.75, 0.0);
    glVertex3f(-100, 0.1, 0.45);       /
    glVertex3f(-100, 0.1, 0.55);
    glVertex3f(100, 0.1, 0.55);
    glVertex3f(100, 0.1, 0.45);

    glColor3f(0.7, 0.7, 0.7);
    glVertex3f(-100, 0.1, 0.55);
    glVertex3f(-100, 0.1, 1);          //
```

```
if (wheelflag)
{
    glPushMatrix();
    glTranslatef(xw, 0, 0);
    glColor3f(0.5, .2, 0.3);
    glBegin(GL_QUADS);
    for (i = 0; i < 200; i += 0.2)
    {
        glVertex3f(-100 + i, 0, 1);
        glVertex3f(-99.9 + i, 0, 1);
        glVertex3f(-99.9 + i, 0.2, 1);
        glVertex3f(-100 + i, 0.2, 1);
        i += 0.5;
    }
    for (i = 0; i < 200; i += 0.2)
    {
        glVertex3f(-100 + i, 0, 0);
        glVertex3f(-99.9 + i, 0, 0);
        glVertex3f(-99.9 + i, 0.2, 0);
        glVertex3f(-100 + i, 0.2, 0);
        i += 0.5;
    }
    glEnd();
    glPopMatrix();
}

glPopMatrix();
}
```



c. House creation:

```
//method to create a house
void house(void) {
    glRotated(-20, 0, 1, 0);
    //roof
    glPushMatrix();
    glColor3f(.388, .2, .0039);
    glScaled(.5, .5, .5);
    glRotated(45, 0, 1, 0);
    glutSolidOctahedron();
    glPopMatrix();
    //house
    glColor3f(.871, .612, .416);
    glTranslated(0, -.38, 0);
    glutSolidCube(.73);
    //windows
    glColor3f(0, 0, 0);
    glTranslated(-.2, .13, .32);
    glutSolidCube(.12);
    glTranslated(.4, 0, 0);
    glutSolidCube(.12);
    //door
    glTranslated(-.2, -.355, .046);
    glScaled(.5, 1.1, 0);
    glutSolidCube(.23);
}
```

d. Tree creation:

```
143   //method to create a tree
144   void tree(void) {
145       //trunk
146       glPushMatrix();
147       glColor3f(.388, .2, .0039);
148       GLUquadric* qobj = gluNewQuadric(); //cyli
149       glRotated(90, 1, 0, 0);
150       gluCylinder(qobj, .05, .05, .4, 30, 30);
151       glPopMatrix();
152       //tree leaves
153       glColor3f(0, .415, .0156);
154       glTranslated(0, -.23, 0);
155       glRotated(-90, 1, 0, 0);
156       glutSolidCone(.3, .3, 40, 40);
157       glTranslated(0, 0, .1);
158       glutSolidCone(.25, .3, 40, 40);
159       glTranslated(0, 0, .1);
160       glutSolidCone(.2, .3, 40, 40);
161   }
162
```

e. Snowman (same as the result of the first coursework):

```
181   void snowman(void) {
182       //body
183       glPushMatrix();
184       glColor3f(1, 1, 1);
185       glTranslated(0, 1, 0);
186       glutSolidSphere(.9, 100, 100);
187       glTranslated(0.0, -2.25, 0);
188       glutSolidSphere(1.5, 100, 100);
189       glTranslated(0.0, -3, 0);
190       glutSolidSphere(2, 100, 100);
191       glPopMatrix();
192       //face
193       glPushMatrix();
194       //eyes
195       glTranslated(-.35, 1.5, .75);
196       glColor3f(0, 0, 0);
197       glutSolidSphere(.1, 100, 100);
198       glTranslated(.65, 0, 0);
199       glutSolidSphere(.1, 100, 100);
200       glColor3f(.8, .3, 0);
201       //nose
202       glTranslated(-.325, -.3, .14);
203       glutSolidCone(.15, 1, 100, 100);
204       //mouth
205       glColor3f(0, 0, 0);
206       glTranslated(-.3, -.4, 0);
207       glutSolidSphere(.07, 100, 100);
208       glTranslated(.15, -.075, 0);
209       glutSolidSphere(.07, 100, 100);
210       glTranslated(.15, -.005, 0);
```

f. Sun and moon creation:

```
165   void moonOrSun(void) {
166       GLfloat pos1[] = { 0, 0, 0, 1 }, //light position
167           emission1[] = { 0, 0, 0, 1 },
168           emission_default[] = { 0, 0, 0, 1 },
169           amb1[] = { .4, .4, .4, 1.0 }; //ambient intensity
170
171       glTranslated(.05, 0, 0);
172       glLightfv(GL_LIGHT1, GL_POSITION, pos1);
173       glMaterialfv(GL_FRONT, GL_EMISSION, emission1);
174       glutSolidSphere(.4, 40, 40);
175       glMaterialfv(GL_FRONT, GL_EMISSION, emission_default);
176       glLightfv(GL_LIGHT1, GL_AMBIENT, amb1);
177       glEnable(GL_LIGHTING);
178       glEnable(GL_LIGHT1);
179   }
180
```

Keyboard functions

Then, some keyboard functions are considered. For example, moving the car, different viewpoint of the scene environment, change the car color and so on.

```
void idle(void) { ... }

void NormalKey(GLubyte key, GLint x, GLint y) { ... }

static void SpecialKeyFunc(int Key, int x, int y) { ... }

void myMenu(int id) { ... }

void colorMenu(int id) { ... }
```

a. Change the color of the car:

Car color change is a submenu implemented in the function “colorMenu”

```
int submenu = glutCreateMenu(colorMenu);
glutAddMenuEntry("blue", 6);
glutAddMenuEntry("red", 7);
glutAddMenuEntry("green", 8);
glutAddMenuEntry("black", 9);
glutAddMenuEntry("yellow", 10);
glutAddMenuEntry("grey", 11);
glutCreateMenu(myMenu);
glutAddSubMenu("car colors", submenu);
```

```
void colorMenu(int id)
{
    if (id == 6)
    {
        r = g = 0;
        b = 1;
        glutPostRedisplay();
    }
    if (id == 7)
    {
        r = 0.8;
        b = g = 0;
        glutPostRedisplay();
    }
    if (id == 8)
    {
        g = 1;
        r = b = 0;
        glutPostRedisplay();
    }
    if (id == 9)
    {
        r = b = g = 0;
        glutPostRedisplay();
    }
    if (id == 10)
    {
        b = 0;
        r = g = 1;
        glutPostRedisplay();
    }
```

b. Move the car:

Car move is implemented in the function “SpecialKeyFunc”.

Use Left arrow (<-) and right arrow (->) on the keyboard to control the move of the car.

```
static void SpecialKeyFunc(int Key, int x, int y)
{
    switch (Key) {
    case GLUT_KEY_RIGHT:
        xt += 0.02;
        angle += 5;

        glutPostRedisplay();
        break;

    case GLUT_KEY_LEFT:
        xt -= 0.02;
        angle += 5;

        glutPostRedisplay();
        break;
    }
}
```

c. Change the mode of day and night:

Altering day and night is implemented in the function “myMenu”.

```
835     if (id == 3)
836     {
837         dayNight = 1;
838         moon = false;
839         glClearColor(1, 1, 1, 1);
840         glDisable(GL_FOG);
841         glutPostRedisplay();
842     }
843
844     if (id == 4)
845     {
846         dayNight = 0;
847         moon = true;
848         fog = 1;
849         glClearColor(0.1, 0.1, 0.1, 0);
850         GLfloat fogcolour[4] = { 0.0,0.0,0.0,1.0 };
851
852         glFogfv(GL_FOG_COLOR, fogcolour);
853         glFogf(GL_FOG_DENSITY, 0.5);
854         glFogi(GL_FOG_MODE, GL_EXP);
855         glHint(GL_FOG_HINT, GL_FASTEST);
856         glEnable(GL_FOG);
857
858         glutPostRedisplay();
859     }
860 }
```

d. Change the viewpoint and perspective (e.g. X, Y and Z keys on the keyboard):

X (x): Rotate all 3D objects around x-axis

Y (y): Rotate all 3D objects around y-axis

Z (z): Rotate perspective around z-axis

U (u): Translate all 3D objects up

1(2): Alter between FILL and LINE

d(a): Scale up and scale down of perspective

s(w): Translate to left and translate to right

i(k): Pitch

q(e): Yaw

j(l): Roll

```
void NormalKey(GLubyte key, GLint x, GLint y)
{
    switch (key) {
    case 'x': xangle += 5.0;
        glutPostRedisplay();
        break;

    case 'X':xangle -= 5.0;
        glutPostRedisplay();
        break;

    case 'y':
        yangle += 5.0;
        glutPostRedisplay();
        break;

    case 'Y':
        yangle -= 5.0;
        glutPostRedisplay();
        break;

    case 'z':
        zangle += 5.0;
        glutPostRedisplay();
        break;

    case 'Z':
        zangle -= 5.0;
        glutPostRedisplay();
        break;

    case 'u':                           /* Move up */
        yt += 0.2;
```

```
case '1':
    glPolygonMode(GL_FRONT, GL_LINE);
    glutPostRedisplay();
    break;

case '2':
    glPolygonMode(GL_FRONT, GL_FILL);
    glutPostRedisplay();
    break;

case 'd':
    cam.slide(0.1, 0, 0);
    glutPostRedisplay();
    break;

case 'a':
    cam.slide(-0.1, 0, 0);
    glutPostRedisplay();
    break;

case 's':
    cam.slide(0, 0, 0.5);
    glutPostRedisplay();
    break;

case 'w':
    cam.slide(0, 0, -0.5);
    glutPostRedisplay();
    break;
```

```
case 'i':
    cam.pitch(-0.1);
    glutPostRedisplay();
    break;

case 'k':
    cam.pitch(0.1);
    glutPostRedisplay();
    break;

case 'q':
    cam.yaw(-0.1);
    glutPostRedisplay();
    break;

case 'e':
    cam.yaw(0.1);
    glutPostRedisplay();
    break;

case 'j':
    cam.roll(-0.1);
    glutPostRedisplay();
    break;

case 'l':
    cam.roll(0.1);
    glutPostRedisplay();
    break;
```

Animation

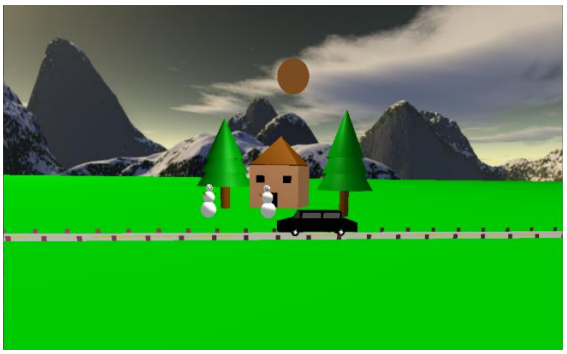
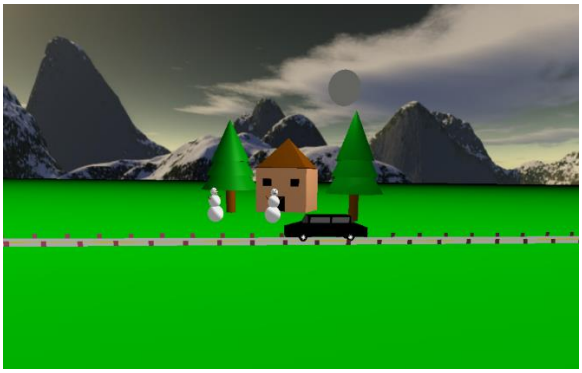
According to the animation, the sun and moon move from left to right alternately and the snowman will keep spinning.

```
void idle(void) {
    //movement of moon/sun
    moonHorizontal = moonHorizontal + .002;
    if (goDown == false) moonVertical = moonVertical + .0004;
    if (moonHorizontal > 5.5) {
        moonHorizontal = 0;
        moonVertical = 0;
        goDown = false;
    }
    if (moonVertical > .45 || goDown == true) {
        goDown = true;
        moonVertical = moonVertical - .0004;
    }

    //movement for snowmen
    if (sLeft) {
        snowmanMove = snowmanMove + 1.5;
        if (snowmanMove >= 20) {
            sLeft = false;
            sRight = true;
        }
    }
    if (sRight) {
        snowmanMove = snowmanMove - 1.5;
        if (snowmanMove <= -20) {
            sLeft = true;
            sRight = false;
        }
    }
    glutPostRedisplay();
}
```

Fog effect

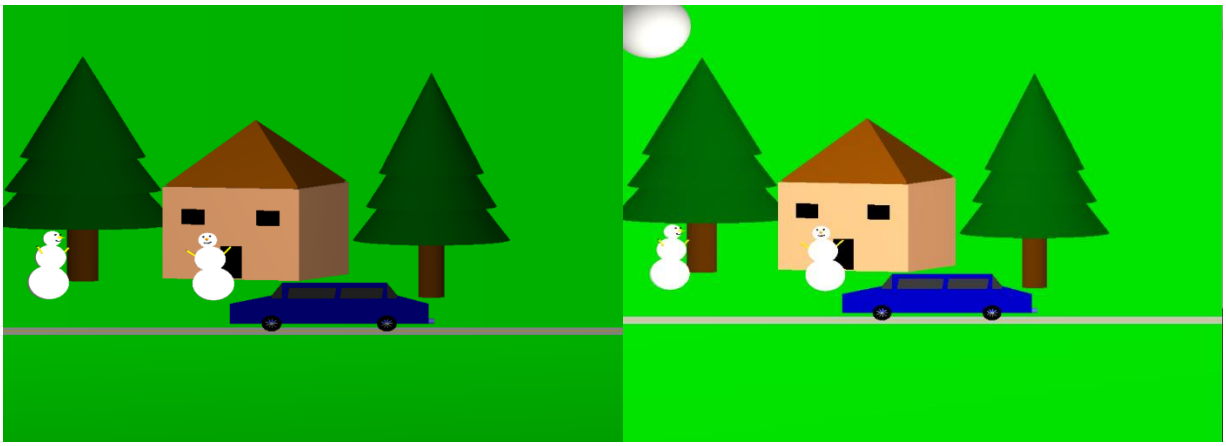
Fog effect are used for day and night alternation.

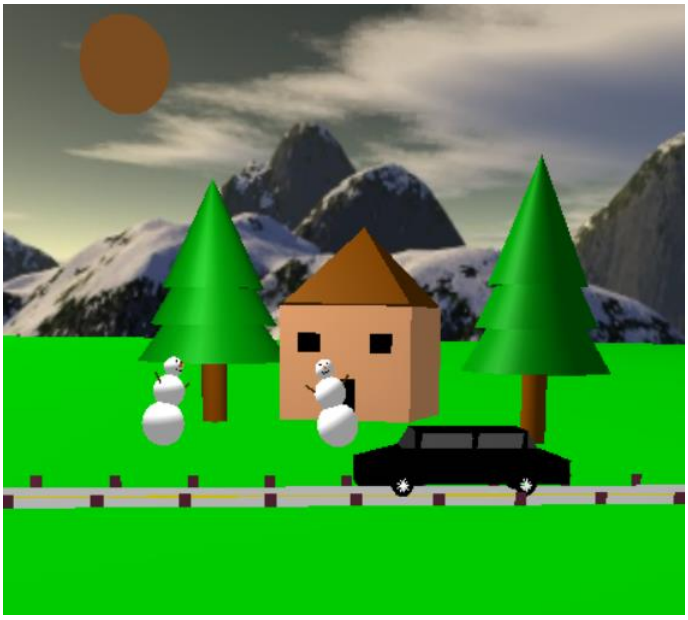


Light effect

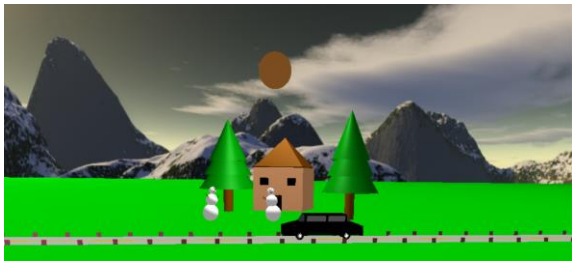
The sun or moon is set as the light source. So with the animation of sun or moon, the light effect can be overserved obviously.

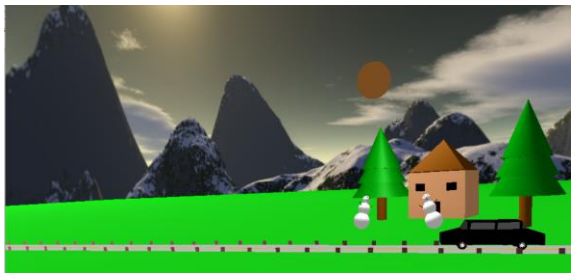


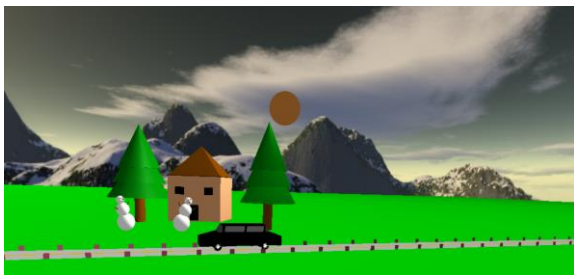


Texture and skybox

Texture are mainly set on the sun or moon and skybox. However, the effect of sun texture is not very appropriate, so that add the texture of sun or moon as comment.







Initialization and main function

```
GLvoid InitGL(GLfloat Width, GLfloat Height)
{
        /* Add line width,   ditto */
    glClearColor(1, 1, 1, 1);
    glViewport(0, 0, Width, Height);
    gluPerspective(45.0, Width / Height, 0.1, 100.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-3, 3.0, -4, 2.0, -50.0, 50.0);


    t = gluNewQuadric();
    gluQuadricDrawStyle(t, GLU_FILL);

    glEnable(GL_LIGHTING);

    glEnable(GL_LIGHT0);

    GLfloat amb0[] = { 0.2f, 0.2f, 0.2f, 1.0f };
    GLfloat diffuse0[] = { 0.8f, 0.8f, 0.8, 1.0f };
    GLfloat specular0[] = { 0.5f, 0.5f, 0.5f, 1.0f };
    GLfloat pos0[] = { 1.5f, 1.0f, 4.0f, 1.0f };

    glLightfv(GL_LIGHT0, GL_AMBIENT, amb0);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse0);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specular0);
    glLightfv(GL_LIGHT0, GL_POSITION, pos0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
```

Outcome