Computer Graphics (UCS505)

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

(Deemed to be University), Patiala – 147004

SCENERY ANIMATION

Submitted By:

Arijeet Mishra - 102117065

Tino Abraham Reji – 102117066

Raman Preet Singh - 102117070

3CS3

Submitted To:

Dr. Kuntal Chowdhury

In this project, we have made an animation using the power of OpenGL to animate a scenery containing a straight road with driving cars. In the background, we have implemented random scenery elements like buildings and mountains, while at the same time also implemented day and night cycles.

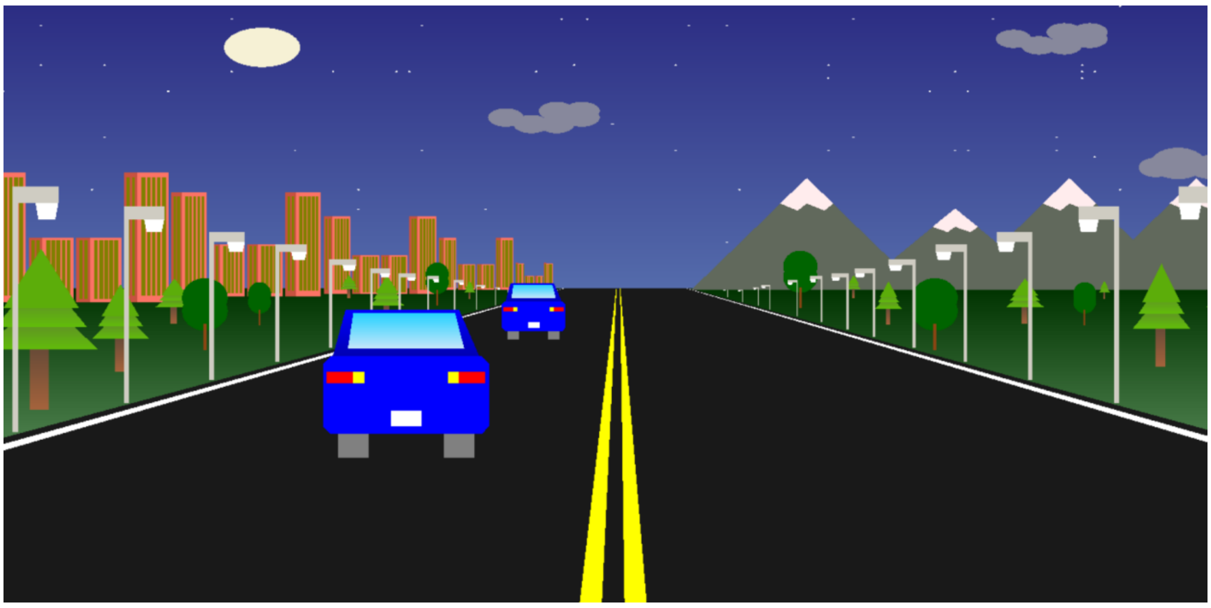

Fig 1 An excerpt from the animation

Explanation of the whole code is as follows:

Step 1: Importing necessary header files.

```
#include<windows.h>
#include <GL/glut.h>
#include <math.h>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
```

These are the necessary header files for the project. The **windows.h** is a special header file only to be used when implementing the project in Windows, and is not needed explicitly when implemented on Linux platforms.

GL/glut.h is a part of the OpenGL Utility Toolkit (GLUT), which is a library for OpenGL programs. GLUT provides functions for creating windows, handling user input, and managing events like keyboard and mouse interactions.

Rest of the header files used are normal, like they are used in normal C/C++ programs. **stdio.h** defines our project to be done in C, and **time.h** was used to write frame/time related tasks. (i.e: Day and night cycles)

Step 2: Defining some definitions

```
const double PI = 3.141592654;
float colorroadTop[]={0.300,0.300,0.300};
float colorroadBot[]={0.098,0.098,0.098};
float colorBuildWindow[]={0.133, 0.243, 0.259};
float colorgrass[]={0.000, 0.500, 0.000};
float colorlamp[]={135,136,156};
int speed = 0;
int dn=1;
```

These definitions are of important use in the code we are going to look through in the following steps.

Step 3: Defining a “frame”

```
void frame(int x)
{
   speewd=speed+1;
   if(speed>335){
   speed=0;
   dn++;
   }
   glutPostRedisplay();
   printf("FRAME: %d",speed);
   glutTimerFunc(30,frame,0);
}
```

In this function here, we have defined the duration of a frame. The frame restarts once the *speed* becomes greater than 335. Using the ***glutTimerFunc*** function, we have ensured that the frame repeats after 30 milliseconds.

Step 4: Defining the “clouds”

```
float cloudX1=0;
float cloudX1C=0.2;
void cloudAnimation(){
if(cloudX1>101){
cloudX1=0;
}
cloudX1+=cloudX1C;
}
//One code snippet for a cloud, the values can be modified for other clouds too
```

We have defined an initial x-coordinate and a speed of the cloud here. Once the cloud reaches a certain point(the right most edge of the screen), it warps to the starting position. This has been implemented on all the six clouds of the animation, and their codes are similar. They are demarcated as *cloudX1*, *cloudX2*, … , *cloudX5* and *cloudX6*.

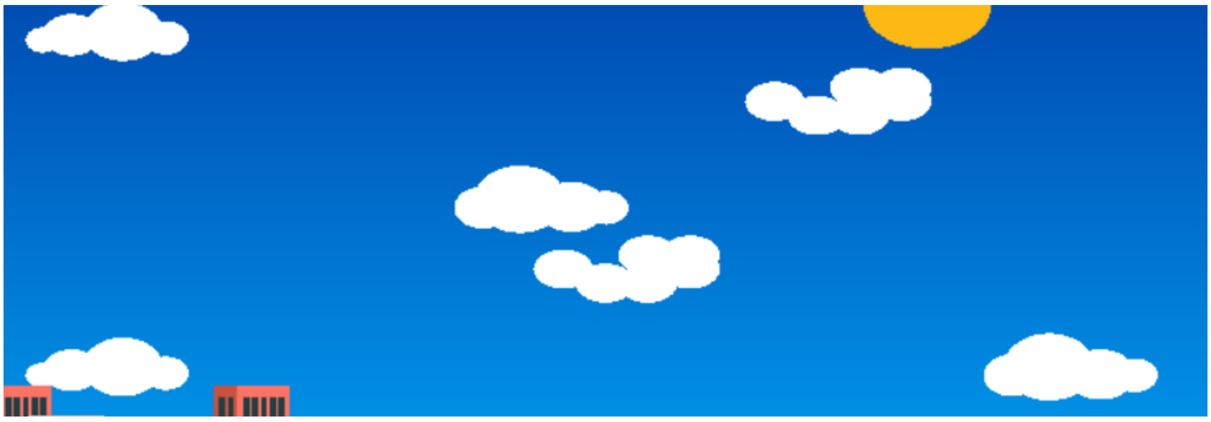

Fig 2 All six clouds from *cloudX1* to *cloudX6* in action

Step 5: Defining the "sun" and "moon" 's path

```
void circle(double r)
{
   float theta;
   glBegin(GL_LINE_LOOP);
   for(int i=0;i<360;i++)
   {
      theta=i*PI/180;// changing degree to radian
      glVertex2f(r*cos(theta),r*sin(theta));
   }
   glEnd();
}
void circle1()
{
   glColor3d(0,0,0);
   circle(50);
}
```

This function explicitly defines the sun and the moon. For the sake of convenience, both are taken to be same size. (Although in real life, sun and moon appear to be equal in sizes due to their distances and sizes balancing out from Earth). Both have been coded to move around in a circle from 0 to 359 degrees. The "Sun" has been coded to rotate in clockwise direction, or from East to West and the "Moon" has been coded to rotate in counter-clockwise direction from West to East. (Like it happens in real life).

In our animation, once the sun moves 180 degrees counter-clockwise while at the same time moon moves vice-versa(behind the scenery) during the day , the roles get reversed. Night falls and the sun goes behind the scenery while the moon dominates.

Please do note that the sizes of the sun and moon may not appear spherical due to screen resolution of the user.

Step 6: Defining a circle

```
void drawCircle(float a, float b, float r)
{
   int i;
   int triangleAmount = 100;
   GLfloat twicePi = 2.0f * PI;
   GLfloat x=a;
   GLfloat y=b;
   GLfloat radius =r;
   glBegin(GL_TRIANGLE_FAN);
   glVertex2f(x, y);
   for(i = 0; i <= triangleAmount; i++)
   {
      glVertex2f(
         x + (radius * cos(i *  twicePi / triangleAmount)),
         y + (radius * sin(i * twicePi / triangleAmount))
      );
   }
   glEnd();
}
```

This function is used to explicitly define the circles for the sun , moon and stars as a proper function does not exist in OpenGL. This happens by approximating a circle as a combination of triangles(approximately). In our example, we have taken the number of triangles rendered to be 100. We iterate the triangle points evenly over the circumference of the circle to define the circle.

Step 7: Defining the elements (or buildings)

```
void buid(){
   //3rd
   glPushMatrix();
   glTranslatef(-5.5,0,0);
   glBegin(GL_QUADS);

   glColor3f(0.78, 0.322, 0.251); // building left side shadow
   glVertex3f(12.5333f, 13.606f, 0.0f);
   glVertex3f(12.5333f, 33.4328f, 0.0f);
   glVertex3f(14.6667f, 33.4328f, 0.0f);
   glVertex3f(14.6667f, 13.606f, 0.0f);


   glColor3f(colorBuildWindow[0],colorBuildWindow[1],colorBuildWindow[2]); // left side long apartment
   glVertex3f(13.0f, 14.7015f, 0.0f);
   glVertex3f(13.0f, 32.6866f, 0.0f);
   glVertex3f(13.5f, 32.6866f, 0.0f);
   glVertex3f(13.5f, 14.7015f, 0.0f);

   glVertex3f(13.8f, 14.7015f, 0.0f);
   glVertex3f(13.8f, 32.6866f, 0.0f);
   glVertex3f(14.3f, 32.6866f, 0.0f);
   glVertex3f(14.3f, 14.7015f, 0.0f);

   glColor3f(0.984, 0.451, 0.396);  // building right side shadow
   glVertex3f(14.6667f, 13.606f, 0.0f);
   glVertex3f(14.6667f, 33.4075f, 0.0f);
```

```
    glVertex3f(19.6f, 33.4075f, 0.0f);
    glVertex3f(19.6f, 13.606f, 0.0f);

    glColor3f(colorBuildWindow[0],colorBuildWindow[1],colorBuildWindow[2]); // right side long apartment
    glVertex3f(15.3333f, 14.7015f, 0.0f);
    glVertex3f(15.3333f, 32.6866f, 0.0f);
    glVertex3f(15.9333f, 32.6866f, 0.0f);
    glVertex3f(15.9333f, 14.7015f, 0.0f);

    glVertex3f(16.1333f, 14.7015f, 0.0f);
    glVertex3f(16.1333f, 32.6866f, 0.0f);
    glVertex3f(16.7333f, 32.6866f, 0.0f);
    glVertex3f(16.7333f, 14.7015f, 0.0f);

    glVertex3f(16.9333f, 14.7015f, 0.0f);
    glVertex3f(16.9333f, 32.6866f, 0.0f);
    glVertex3f(17.5333f, 32.6866f, 0.0f);
    glVertex3f(17.5333f, 14.7015f, 0.0f);

    glVertex3f(17.7333f, 14.7015f, 0.0f);
    glVertex3f(17.7333f, 32.6866f, 0.0f);
    glVertex3f(18.3333f, 32.6866f, 0.0f);
    glVertex3f(18.3333f, 14.7015f, 0.0f);

    glVertex3f(18.5333f, 14.7015f, 0.0f);
    glVertex3f(18.5333f, 32.6866f, 0.0f);
    glVertex3f(19.1333f, 32.6866f, 0.0f);
    glVertex3f(19.1333f, 14.7015f, 0.0f);
  glEnd();
  glPopMatrix();

    //3rd
    glPushMatrix();
    glTranslatef(2,0,0);
    glBegin(GL_QUADS);

    glColor3f(0.78, 0.322, 0.251); // building left side shadow
    glVertex3f(12.5333f, 13.606f, 0.0f);
    glVertex3f(12.5333f, 33.4328f, 0.0f);
    glVertex3f(14.6667f, 33.4328f, 0.0f);
    glVertex3f(14.6667f, 13.606f, 0.0f);


    glColor3f(colorBuildWindow[0],colorBuildWindow[1],colorBuildWindow[2]); // left side long apartment
    glVertex3f(13.0f, 14.7015f, 0.0f);
    glVertex3f(13.0f, 32.6866f, 0.0f);
    glVertex3f(13.5f, 32.6866f, 0.0f);
    glVertex3f(13.5f, 14.7015f, 0.0f);

    glVertex3f(13.8f, 14.7015f, 0.0f);
    glVertex3f(13.8f, 32.6866f, 0.0f);
    glVertex3f(14.3f, 32.6866f, 0.0f);
    glVertex3f(14.3f, 14.7015f, 0.0f);

    glColor3f(0.984, 0.451, 0.396);  // building right side shadow
    glVertex3f(14.6667f, 13.606f, 0.0f);
    glVertex3f(14.6667f, 33.4075f, 0.0f);
    glVertex3f(19.6f, 33.4075f, 0.0f);
    glVertex3f(19.6f, 13.606f, 0.0f);
```

```
        glColor3f(colorBuildWindow[0],colorBuildWindow[1],colorBuildWindow[2]); // right side long apartment
        glVertex3f(15.3333f, 14.7015f, 0.0f);
        glVertex3f(15.3333f, 32.6866f, 0.0f);
        glVertex3f(15.9333f, 32.6866f, 0.0f);
        glVertex3f(15.9333f, 14.7015f, 0.0f);

        glVertex3f(16.1333f, 14.7015f, 0.0f);
        glVertex3f(16.1333f, 32.6866f, 0.0f);
        glVertex3f(16.7333f, 32.6866f, 0.0f);
        glVertex3f(16.7333f, 14.7015f, 0.0f);

        glVertex3f(16.9333f, 14.7015f, 0.0f);
        glVertex3f(16.9333f, 32.6866f, 0.0f);
        glVertex3f(17.5333f, 32.6866f, 0.0f);
        glVertex3f(17.5333f, 14.7015f, 0.0f);

        glVertex3f(17.7333f, 14.7015f, 0.0f);
        glVertex3f(17.7333f, 32.6866f, 0.0f);
        glVertex3f(18.3333f, 32.6866f, 0.0f);
        glVertex3f(18.3333f, 14.7015f, 0.0f);

        glVertex3f(18.5333f, 14.7015f, 0.0f);
        glVertex3f(18.5333f, 32.6866f, 0.0f);
        glVertex3f(19.1333f, 32.6866f, 0.0f);
        glVertex3f(19.1333f, 14.7015f, 0.0f);
    glEnd();
    glPopMatrix();
    //3rd
        glPushMatrix();
        glTranslatef(9.5,-13.59,0);
        glScalef(1,2,0);
        glBegin(GL_QUADS);

        glColor3f(0.78, 0.322, 0.251); // building left side shadow
        glVertex3f(12.5333f, 13.606f, 0.0f);
        glVertex3f(12.5333f, 33.4328f, 0.0f);
        glVertex3f(14.6667f, 33.4328f, 0.0f);
        glVertex3f(14.6667f, 13.606f, 0.0f);


        glColor3f(colorBuildWindow[0],colorBuildWindow[1],colorBuildWindow[2]); // left side long apartment
        glVertex3f(13.0f, 14.7015f, 0.0f);
        glVertex3f(13.0f, 32.6866f, 0.0f);
        glVertex3f(13.5f, 32.6866f, 0.0f);
        glVertex3f(13.5f, 14.7015f, 0.0f);

        glVertex3f(13.8f, 14.7015f, 0.0f);
        glVertex3f(13.8f, 32.6866f, 0.0f);
        glVertex3f(14.3f, 32.6866f, 0.0f);
        glVertex3f(14.3f, 14.7015f, 0.0f);

        glColor3f(0.984, 0.451, 0.396);  // building right side shadow
        glVertex3f(14.6667f, 13.606f, 0.0f);
        glVertex3f(14.6667f, 33.4075f, 0.0f);
        glVertex3f(19.6f, 33.4075f, 0.0f);
        glVertex3f(19.6f, 13.606f, 0.0f);

        glColor3f(colorBuildWindow[0],colorBuildWindow[1],colorBuildWindow[2]); // right side long apartment
        glVertex3f(15.3333f, 14.7015f, 0.0f);
        glVertex3f(15.3333f, 32.6866f, 0.0f);
        glVertex3f(15.9333f, 32.6866f, 0.0f);
        glVertex3f(15.9333f, 14.7015f, 0.0f);

        glVertex3f(16.1333f, 14.7015f, 0.0f);
        glVertex3f(16.1333f, 32.6866f, 0.0f);
        glVertex3f(16.7333f, 32.6866f, 0.0f);
        glVertex3f(16.7333f, 14.7015f, 0.0f);

        glVertex3f(16.9333f, 14.7015f, 0.0f);
        glVertex3f(16.9333f, 32.6866f, 0.0f);
        glVertex3f(17.5333f, 32.6866f, 0.0f);
        glVertex3f(17.5333f, 14.7015f, 0.0f);

        glVertex3f(17.7333f, 14.7015f, 0.0f);
        glVertex3f(17.7333f, 32.6866f, 0.0f);
        glVertex3f(18.3333f, 32.6866f, 0.0f);
        glVertex3f(18.3333f, 14.7015f, 0.0f);

        glVertex3f(18.5333f, 14.7015f, 0.0f);
        glVertex3f(18.5333f, 32.6866f, 0.0f);
        glVertex3f(19.1333f, 32.6866f, 0.0f);
        glVertex3f(19.1333f, 14.7015f, 0.0f);
    glEnd();
    glPopMatrix();

    //3rd
        glPushMatrix();
        glTranslatef(-13,-13.59,0);
        glScalef(1,2,0);
        glBegin(GL_QUADS);

        glColor3f(0.78, 0.322, 0.251); // building left side shadow
        glVertex3f(12.5333f, 13.606f, 0.0f);
        glVertex3f(12.5333f, 33.4328f, 0.0f);
        glVertex3f(14.6667f, 33.4328f, 0.0f);
        glVertex3f(14.6667f, 13.606f, 0.0f);


        glColor3f(colorBuildWindow[0],colorBuildWindow[1],colorBuildWindow[2]); // left side long apartment
        glVertex3f(13.0f, 14.7015f, 0.0f);
        glVertex3f(13.0f, 32.6866f, 0.0f);
        glVertex3f(13.5f, 32.6866f, 0.0f);
        glVertex3f(13.5f, 14.7015f, 0.0f);

        glVertex3f(13.8f, 14.7015f, 0.0f);
        glVertex3f(13.8f, 32.6866f, 0.0f);
        glVertex3f(14.3f, 32.6866f, 0.0f);
        glVertex3f(14.3f, 14.7015f, 0.0f);

        glColor3f(0.984, 0.451, 0.396);  // building right side shadow
        glVertex3f(14.6667f, 13.606f, 0.0f);
        glVertex3f(14.6667f, 33.4075f, 0.0f);
        glVertex3f(19.6f, 33.4075f, 0.0f);
        glVertex3f(19.6f, 13.606f, 0.0f);

        glColor3f(colorBuildWindow[0],colorBuildWindow[1],colorBuildWindow[2]); // right side long win
        glVertex3f(15.3333f, 14.7015f, 0.0f);
        glVertex3f(15.3333f, 32.6866f, 0.0f);
        glVertex3f(15.9333f, 32.6866f, 0.0f);
        glVertex3f(15.9333f, 14.7015f, 0.0f);
        glVertex3f(16.1333f, 14.7015f, 0.0f);
        glVertex3f(16.1333f, 32.6866f, 0.0f);
        glVertex3f(16.7333f, 32.6866f, 0.0f);
        glVertex3f(16.7333f, 14.7015f, 0.0f);

        glVertex3f(16.9333f, 14.7015f, 0.0f);
        glVertex3f(16.9333f, 32.6866f, 0.0f);
        glVertex3f(17.5333f, 32.6866f, 0.0f);
        glVertex3f(17.5333f, 14.7015f, 0.0f);

        glVertex3f(17.7333f, 14.7015f, 0.0f);
        glVertex3f(17.7333f, 32.6866f, 0.0f);
        glVertex3f(18.3333f, 32.6866f, 0.0f);
        glVertex3f(18.3333f, 14.7015f, 0.0f);

        glVertex3f(18.5333f, 14.7015f, 0.0f);
        glVertex3f(18.5333f, 32.6866f, 0.0f);
        glVertex3f(19.1333f, 32.6866f, 0.0f);
        glVertex3f(19.1333f, 14.7015f, 0.0f);
    glEnd();
    glPopMatrix();

    }
```

We have defined the scenery elements inside the function ***buid.*** This includes the buildings on the left hand side of the animation and their corresponding shadows. This has been done by defining them as simple polygons using glVertex3f(). Also, a template of four buildings has been made and then scaled down accordingly. Same has been done for night time, as shown in the illustrations below.

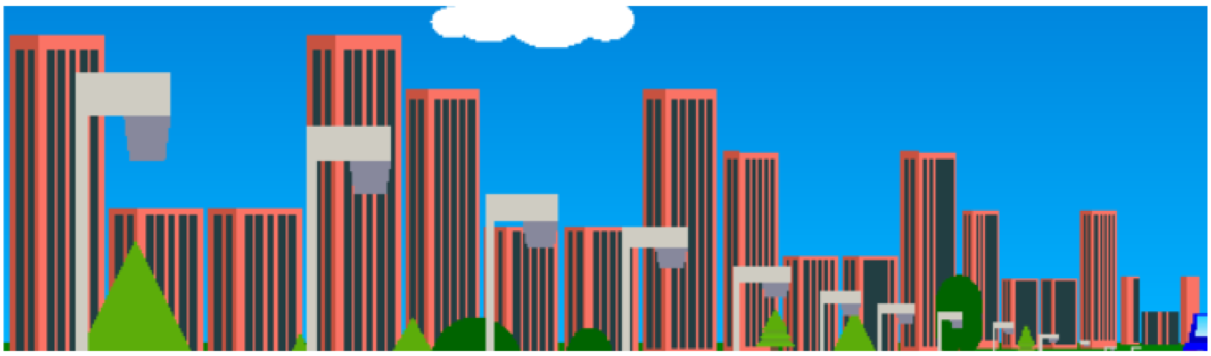

Fig 3 Buildings rendered for the daytime

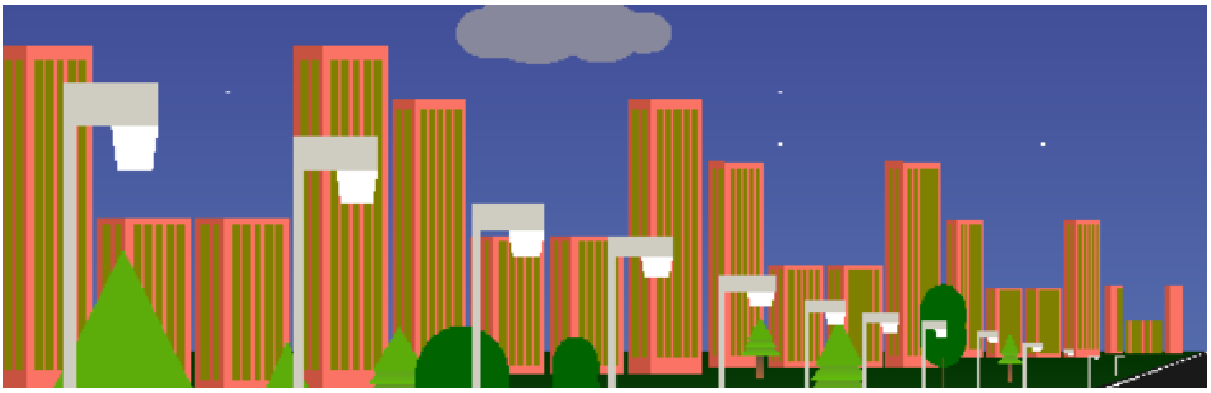

Fig 4 Same buildings in nighttime

Step 7: Defining the mountains

```
void mount()
{
   ///Hill
   glBegin(GL_POLYGON);
   glColor3f(0.38, 0.41, 0.36);
   glVertex2i(70, 70);
   glVertex2i(200, 225);
   glVertex2i(330, 70);
   glEnd();

   ///Hill_Snow
   glBegin(GL_POLYGON);
   glColor3f(1.25, 0.924, 0.930);

   glVertex2i(200, 225);
   glVertex2i(230, 190);
   glVertex2i(220, 180);
   glVertex2i(200, 190);
   glVertex2i(190, 180);
   glVertex2i(170, 190);
   glEnd();
}
```

This has also been done by defining polygons on the right side of the screen. The snow has been rendered over the mountain cap.

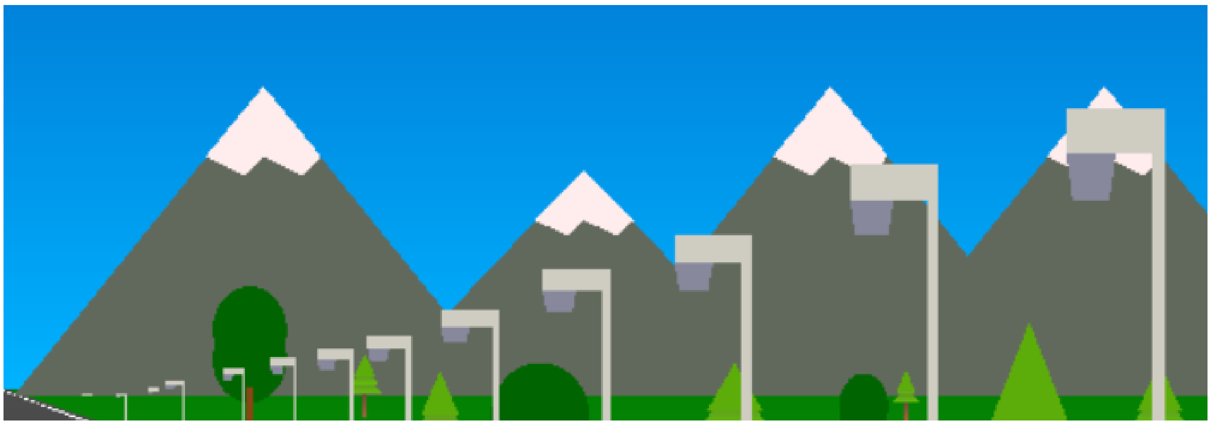

Fig 5 The Mountain

Step 8: Defining the roads

```
void road(){
    //Road
   glColor3f(colorroadTop[0],colorroadTop[1],colorroadTop[2]);
   glBegin(GL_POLYGON);
   glVertex2f(51 -5, 55);
   glVertex2f(51 + 5, 55);
   glColor3f(colorroadBot[0],colorroadBot[1],colorroadBot[2]);
   glVertex2f(160,0);
   glVertex2f(-60,0);
   glEnd();
   //Road Left Line
   glColor3f(1.0, 1.0, 1.0);
   glBegin(GL_POLYGON);
   glVertex2f(-56,0);
   glVertex2f(-52,0);
   glVertex2f(51 -4.7, 55);
   glVertex2f(51 -4.8, 55);
   glEnd();
    //Road Right Line
   glColor3f(1.0, 1.0, 1.0);
   glBegin(GL_POLYGON);
   glVertex2f(156,0);
   glVertex2f(152,0);
   glVertex2f(51 +4.7, 55);
   glVertex2f(51 +4.8, 55);
   glEnd();
   //Road Left 2nd Line
   glColor3f(1.0, 1.0, 0.0);
   glBegin(GL_POLYGON);
   glVertex2f(47,0);
   glVertex2f(49,0);
   glVertex2f(50.33, 55);
   glVertex2f(50.3, 55);
   glEnd();
    //Road Right 2nd Line
   glColor3f(1.0, 1.0, 0.0);
   glBegin(GL_POLYGON);
   glVertex2f(51,0);
   glVertex2f(53,0);
   glVertex2f(50.63, 55);
   glVertex2f(50.6 ,55);
   glEnd();
}
```

The road has been made to be of two lanes and made possible using *glVertex2f* command or both of them.

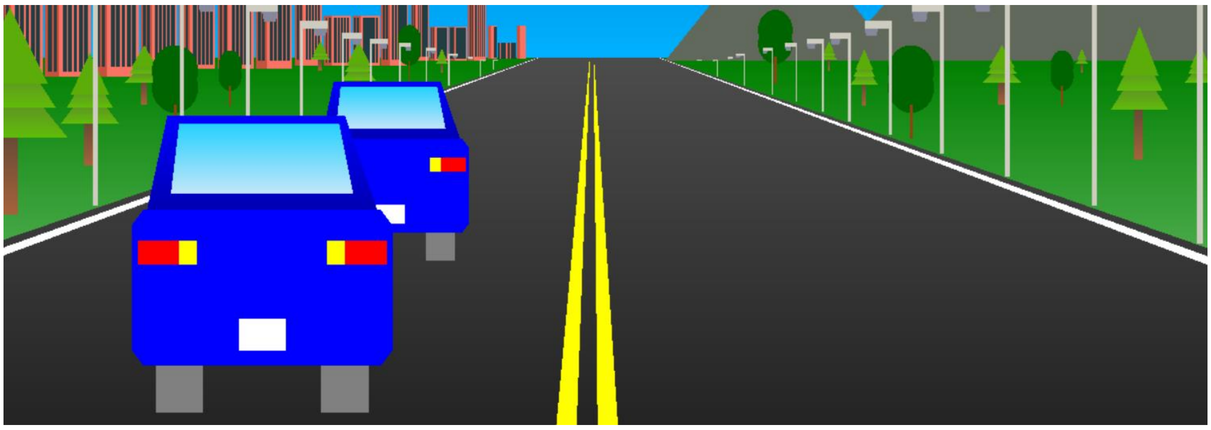

Fig 6 The roads

Step 9: Defining the roadside

```
void roadside(){
   //Road BackGround
   glColor3f(colorgrass[0],colorgrass[1],colorgrass[2]);
   glBegin(GL_POLYGON);
   glVertex2f(0, 55);
   glVertex2f(100, 55);
   glColor3f(0.7, 0.9, 0.7);
   glVertex2f(100, 50 - 50);
   glVertex2f(0, 50 - 50);
   glEnd();
}
```

Roadside function is just simple grass that has been defined. Colors has also been defined accordingly.

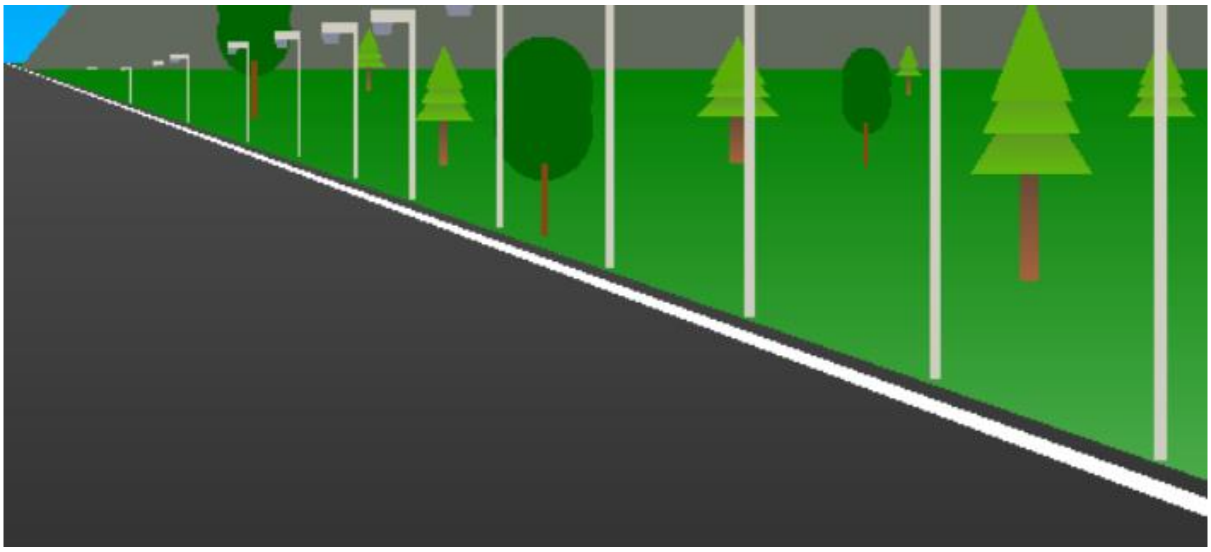

Fig 7 Roadside, complete with trees and lampposts

Step 9: Defining the trees

```
void tree(){

   glPushMatrix();
   glBegin(GL_TRIANGLES);
   glColor3f(0.38, 0.74, 0.04);
   glVertex2f(1,2.3);
   glVertex2f(4,2.3);
   glColor3f(0.372,0.388,0.266);
   glVertex2f(2.5,3.5);
   glEnd();

   glPopMatrix();
   glPushMatrix();
   glBegin(GL_TRIANGLES);
   glColor3f(0.38, 0.74, 0.04);
   glVertex2f(1.3,2.8);
   glVertex2f(3.7,2.8);
   glColor3f(0.372,0.388,0.266);
   glVertex2f(2.5,4);
   glEnd();

   glPopMatrix();
   glPushMatrix();
   glBegin(GL_TRIANGLES);
   glColor3f(0.361,0.678,0.04);
   glVertex2f(1.5,3.2);
   glVertex2f(3.5,3.2);
   glVertex2f(2.5,4.5);
   glEnd();
   glPopMatrix();

   glPushMatrix();
   glBegin(GL_QUADS);
   glColor3f(0.64705,0.388,0.2352);
   glVertex2f(2.2,1);
   glVertex2f(2.7,1);
   glColor3f(0.411,0.2941,0.2156);
   glVertex2f(2.7,2.3);
   glVertex2f(2.2,2.3);
   glEnd();
   glPopMatrix();

}
void Tree_Model_One()
{

   glPushMatrix();
   drawCircle(110,110,15);
   glPopMatrix();

   glPushMatrix();
   drawCircle(110,100,15);
   glPopMatrix();

   glBegin(GL_POLYGON);
   glColor3ub(139,69,19);
   glVertex2f(109, 70);
```

```
    glVertex2f(109, 90);
    glVertex2f(111, 90);
    glVertex2f(111, 70);
    glEnd();
}



void Tree_One()
{
    glColor3ub(0,100,0);
    glPushMatrix();
    glScalef(0.3,0.4,1);
    glTranslatef(-90,-55,0);
    Tree_Model_One();
    glPopMatrix();
}

void treebackground(){
//LEFT SIDE TREE
    glPushMatrix();
    glTranslatef(-2.5,29.5,0);
    glScalef(3,7,0);
    tree();
    glPopMatrix();
              glPushMatrix();
    glTranslatef(-2+23,27.5+20,0);
    glScalef(0.2,0.3,0);
    Tree_One();
    glPopMatrix();
              glPushMatrix();
    glTranslatef(-2+15,27.5+20,0);
    glScalef(1,2,0);
    tree();
    glPopMatrix();
        glPushMatrix();
    glTranslatef(-2+30,27.5+25,0);
    glScalef(0.5,1,0);
    tree();
    glPopMatrix();
              glPushMatrix();
    glTranslatef(-2+17.5,27.5+15,0);
    glScalef(0.4,0.5,0);
    Tree_One();
    glPopMatrix();
        glPushMatrix();
    glTranslatef(-2+40,27.5+25,0);
    glScalef(0.3,0.8,0);
    tree();
    glPopMatrix();
            glPushMatrix();
    glTranslatef(-2+37,27.5+23,0);
    glScalef(0.2,0.3,0);
    Tree_One();
    glPopMatrix();
              glPushMatrix();
    glTranslatef(-2+32,27.5+20,0);
    glScalef(0.9,2.1,0);
    tree();
    glPopMatrix();
```

```
        glPushMatrix();
glTranslatef(-2.5+10,29.5+9,0);
glScalef(1.5,3.8,0);
tree();
glPopMatrix();
//RIGHT SIDE TREE
    glPushMatrix();
glTranslatef(-2.5+92,29.5+9,0);
glScalef(1.5,4.5,0);
tree();
glPopMatrix();
          glPushMatrix();
glTranslatef(-2+88,27.5+20,0);
glScalef(0.2,0.3,0);
Tree_One();
glPopMatrix();
          glPushMatrix();
glTranslatef(-2+82,27.5+20,0);
glScalef(1,2,0);
tree();
glPopMatrix();
        glPushMatrix();
glTranslatef(-2+90,27.5+25,0);
glScalef(0.3,0.8,0);
tree();
glPopMatrix();
        glPushMatrix();
glTranslatef(-2+70,27.5+25,0);
glScalef(0.4,1,0);
tree();
glPopMatrix();
          glPushMatrix();
glTranslatef(-2+75,27.5+15,0);
glScalef(0.4,0.5,0);
Tree_One();
glPopMatrix();
        glPushMatrix();
glTranslatef(-2+72,27.5+20,0);
glScalef(0.7,1.9,0);
tree();
glPopMatrix();
        glPushMatrix();
glTranslatef(-2+65,27.5+22,0);
glScalef(0.3,0.4,0);
Tree_One();
glPopMatrix();
       glPushMatrix();
glTranslatef(-2+98,27.5+20,0);
glScalef(0.8,2,0);
tree();
glPopMatrix();
}
```

All the functions defined above are for the trees used in the project. They have been created using the *glVertex2f* function and then the *glPushMatrix()* and *glPopMatrix()* functions have been used accordingly. They have been already illustrated on Figure 7.

Step 10: Defining the lampposts

```
void lamppost()
{
   glBegin(GL_QUADS);
   glColor3ub(207, 204, 194);
   glVertex2f(-0.5f,0.6f);///TOP LEFT
   glVertex2f(-0.4f,0.6f); ///TOP RIGHT
   glVertex2f(-0.4f,-0.9f); /// BOTTOM RIGHT
   glVertex2f(-0.5f,-0.9f); /// BOTTOM LEFT
   glEnd();
   glBegin(GL_QUADS);
   glColor3ub(207, 204, 194);
   glVertex2f(-0.4f,0.6f);///TOP LEFT
   glVertex2f(0.3f,0.6f); ///TOP RIGHT
   glVertex2f(0.3f,0.5f); /// BOTTOM RIGHT
   glVertex2f(-0.4f,0.5f); /// BOTTOM LEFT
   glEnd();

   glBegin(GL_QUADS);
   glColor3ub(colorlamp[0],colorlamp[1],colorlamp[2]);
   glVertex2f(-0.1f,0.5f);///TOP LEFT
   glVertex2f(0.3f,0.5f); ///TOP RIGHT
   glVertex2f(0.25f,0.4f); /// BOTTOM RIGHT
   glVertex2f(-0.05f,0.4f); /// BOTTOM LEFT
   glEnd();
}

void leftlamp(){
glPushMatrix();
glTranslatef(5,55.5,0);
glScalef(4.5,25,0);
lamppost();
glPopMatrix();

glPushMatrix();
glTranslatef(15-1.5,55.5,0);
glScalef(4,20,0);
lamppost();
glPopMatrix();

glPushMatrix();
glTranslatef(13.5+8.5-2,55.5-1,0);
glScalef(3.5,15,0);
lamppost();
glPopMatrix();

glPushMatrix();
glTranslatef(13.5+8.5+5-2,55.5-1,0);
glScalef(3.1,12,0);
lamppost();
glPopMatrix();

glPushMatrix();
glTranslatef(13.5+8.5+5+4-2,55-1,0);
glScalef(2.7,9,0);
lamppost();
glPopMatrix();
```

```
glPushMatrix();
glTranslatef(13.5+8.5+5+4+0.9,55-1.2,0);
glScalef(2,7,0);
lamppost();
glPopMatrix();

glPushMatrix();
glTranslatef(13.5+8.5+5+4+0.9+2.1,55-1.2,0);
glScalef(1.7,5.7,0);
lamppost();
glPopMatrix();

glPushMatrix();
glTranslatef(13.5+8.5+5+4+0.9+2.1+2,54,0);
glScalef(1.2,4.7,0);
lamppost();
glPopMatrix();

glPushMatrix();
glTranslatef(13.5+8.5+5+4+0.9+2.1+2+2,54,0);
glScalef(1,3.7,0);
lamppost();
glPopMatrix();

glPushMatrix();
glTranslatef(13.5+8.5+5+4+0.9+2.1+2+2+1.7,54,0);
glScalef(0.9,2.5,0);
lamppost();
glPopMatrix();

glPushMatrix();
glTranslatef(13.5+8.5+5+4+0.9+2.1+2+2+3,54,0);
glScalef(0.5,1.9,0);
lamppost();
glPopMatrix();

glPushMatrix();
glTranslatef(13.5+8.5+5+4+0.9+2.1+2+2+3+1,54,0);
glScalef(0.5,1.3,0);
lamppost();
glPopMatrix();

glPushMatrix();
glTranslatef(13.5+8.5+5+4+0.9+2.1+2+2+3+1+1,54.3,0);
glScalef(0.5,0.8,0);
lamppost();
glPopMatrix();
}

void lamppostr()
{
   glBegin(GL_QUADS);
   glColor3ub(207, 204, 194);
   glVertex2f(-0.5f,0.6f);///TOP LEFT
   glVertex2f(-0.4f,0.6f); ///TOP RIGHT
   glVertex2f(-0.4f,-0.9f); /// BOTTOM RIGHT
   glVertex2f(-0.5f,-0.9f); /// BOTTOM LEFT
   glEnd();
   glBegin(GL_QUADS);
   glColor3ub(207, 204, 194);
```

```
    glVertex2f(-0.4f-0.8,0.6f);///TOP LEFT
    glVertex2f(0.3f-0.8,0.6f); ///TOP RIGHT
    glVertex2f(0.3f-0.8,0.5f); /// BOTTOM RIGHT
    glVertex2f(-0.4f-0.8,0.5f); /// BOTTOM LEFT
    glEnd();

    glBegin(GL_QUADS);
    glColor3ub(colorlamp[0],colorlamp[1],colorlamp[2]);
    glVertex2f(-0.1f-0.8-0.3,0.5f);///TOP LEFT
    glVertex2f(0.3f-0.8-0.3,0.5f); ///TOP RIGHT
    glVertex2f(0.25f-0.8-0.3,0.4f); /// BOTTOM RIGHT
    glVertex2f(-0.05f-0.8-0.3,0.4f); /// BOTTOM LEFT
    glEnd();
}

void rightlamp(){
glPushMatrix();
glTranslatef(105-(5),55.5,0);
glScalef(4.5,25,0);
lamppostr();
glPopMatrix();

glPushMatrix();
glTranslatef(105-(15-1.5),55.5,0);
glScalef(4,20,0);
lamppostr();
glPopMatrix();

glPushMatrix();
glTranslatef(105-(13.5+8.5-1.5),55.5-1,0);
glScalef(3.5,15,0);
lamppostr();
glPopMatrix();

glPushMatrix();
glTranslatef(105-(13.5+8.5+5-1.2),55.5-1,0);
glScalef(3.1,12,0);
lamppostr();
glPopMatrix();

glPushMatrix();
glTranslatef(105-(13.5+8.5+5+4-1),55-1,0);
glScalef(2.7,9,0);
lamppostr();
glPopMatrix();

glPushMatrix();
glTranslatef(105-(13.5+8.5+5+4+2.5),55-1.2,0);
glScalef(2,7,0);
lamppostr();
glPopMatrix();

glPushMatrix();
glTranslatef(105-(13.5+8.5+5+4+0.9+3.8),55-1.2,0);
glScalef(1.7,5.7,0);
lamppostr();
glPopMatrix();

glPushMatrix();
glTranslatef(105-(13.5+8.5+5+4+0.9+2.1+2+2),54,0);
```

```
glScalef(1.2,4.7,0);
lamppostr();
glPopMatrix();

glPushMatrix();
glTranslatef(105-(13.5+8.5+5+4+0.9+2.1+2+2+2),54,0);
glScalef(1,3.7,0);
lamppostr();
glPopMatrix();

glPushMatrix();
glTranslatef(105-(13.5+8.5+5+4+0.9+2.1+2+2+1.7+2.5),54,0);
glScalef(0.9,2.5,0);
lamppostr();
glPopMatrix();

glPushMatrix();
glTranslatef(105-(13.5+8.5+5+4+0.9+2.1+2+2+3+2.3),54,0);
glScalef(0.5,1.9,0);
lamppostr();
glPopMatrix();

glPushMatrix();
glTranslatef(105-(13.5+8.5+5+4+0.9+2.1+2+2+3+1+2.5),54,0);
glScalef(0.5,1.3,0);
lamppostr();
glPopMatrix();

glPushMatrix();
glTranslatef(105-(13.5+8.5+5+4+0.9+2.1+2+2+3+1+1+2.7),54.3,0);
glScalef(0.5,0.8,0);
lamppostr();
glPopMatrix();
}
 void buildingsAll(){
 glPushMatrix();
glTranslatef(0.5,46,0);
glScalef(0.5,0.5,0);
buid();
glPopMatrix();
        glPushMatrix();
glTranslatef(15.4,48.3,0);
glScalef(0.4,0.4,0);
buid();
glPopMatrix();
        glPushMatrix();
glTranslatef(27.4,50,0);
glScalef(0.3,0.3,0);
buid();
glPopMatrix();
        glPushMatrix();
glTranslatef(36.4,52,0);
glScalef(0.2,0.2,0);
buid();
glPopMatrix();
        glPushMatrix();
glTranslatef(42.4,53.5,0);
glScalef(0.1,0.1,0);
buid();
glPopMatrix();}
```

We have created two different kinds of lampposts, one for daytime and another for night. Each one has been made similarly as the trees. The orientation of the lampposts on the right has also been flipped for scenery purposes.

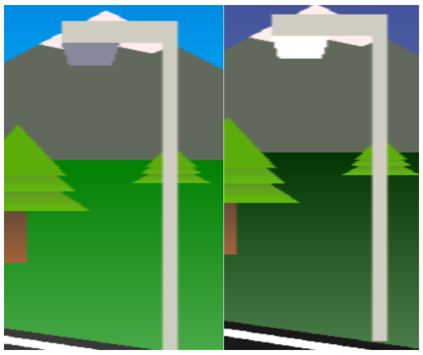

Fig 8 Lamppost for day and night
(Right and Left respectively)

Step 11: Defining the stars for night sky

```
void Stars()
{
   glPushMatrix();
   glBegin(GL_QUADS);
   glColor3ub(255,255,249);
   drawCircle(2.0,78.0,0.1);
   drawCircle(5.0,89.0,0.1);
   drawCircle(9.0,70.0,0.1);
   drawCircle(15.0,85.0,0.1);
   drawCircle(20.0,96.0,0.1);
   drawCircle(30.0,70.0,0.1);
   drawCircle(2.0,85.0,0.1);
   drawCircle(40.0,67.0,0.1);
   drawCircle(50.0,80.0,0.1);
    drawCircle(10.0,78.0,0.1);
   drawCircle(55.0,89.0,0.1);
   drawCircle(30.0,70.0,0.1);
   drawCircle(75.0,85.0,0.1);
   drawCircle(80.0,96.0,0.1);
   drawCircle(60.0,70.0,0.1);
   drawCircle(78.0,85.0,0.1);
   drawCircle(30.0,67.0,0.1);
   drawCircle(98.0,80.0,0.1);
      drawCircle(67.0,87.0,0.1);
   drawCircle(67.0,76.0,0.1);
   drawCircle(90.0,70.0,0.1);
   drawCircle(85.0,76.0,0.1);
   drawCircle(87.0,89.0,0.1);
   drawCircle(59.0,95.0,0.1);
   drawCircle(88.0,88.0,0.1);
   drawCircle(69.0,96.0,0.1);
   drawCircle(77.0,88.0,0.1);
       drawCircle(67.0,76.0,0.1);
   drawCircle(60.0,70.0,0.1);
   drawCircle(55.0,76.0,0.1);
```

```
    drawCircle(47.0,89.0,0.1);
    drawCircle(35.0,95.0,0.1);
    drawCircle(28.0,88.0,0.1);
    drawCircle(48.0,96.0,0.1);
    drawCircle(44.0,88.0,0.1);
           drawCircle(67.0,87.0,0.1);
    drawCircle(67.0,89.0,0.1);
    drawCircle(90.0,98.0,0.1);
    drawCircle(85.0,78.0,0.1);
    drawCircle(87.0,87.0,0.1);
    drawCircle(59.0,62.0,0.1);
    drawCircle(88.0,75.0,0.1);
    drawCircle(69.0,78.0,0.1);
    drawCircle(99.0,88.0,0.1);
        drawCircle(77.0,76.0,0.1);
    drawCircle(65.0,60.0,0.1);
    drawCircle(78.0,76.0,0.1);
    drawCircle(98.0,89.0,0.1);
    drawCircle(56.0,95.0,0.1);
    drawCircle(87.0,88.0,0.1);
    drawCircle(97.0,96.0,0.1);
    drawCircle(34.0,88.0,0.1);
            drawCircle(25.0,76.0,0.1);
    drawCircle(12.0,60.0,0.1);
    drawCircle(35.0,76.0,0.1);
    drawCircle(10.0,89.0,0.1);
    drawCircle(5.0,95.0,0.1);
    drawCircle(20.0,88.0,0.1);
    drawCircle(46.0,96.0,0.1);
    drawCircle(33.0,88.0,0.1);
    glPopMatrix();
    glEnd();
}
```

These have been visualized using the drawCircle command we had discussed earlier in Step 6. The radius of these stars has been taken as 0.1 pixels.

Step 12: Defining the shape for the clouds

```
void cloud1(){

    glPushMatrix();

    glTranslatef(0,0,0);
    glTranslatef(1.0, 6.0, 0);
    glutSolidSphere(0.7, 250, 250);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(1.0+1, 7.0, 0);
    glutSolidSphere(0.7, 250, 250);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(2.0+1, 7.0, 0);
    glutSolidSphere(0.7, 250, 250);
    glPopMatrix();
    glPushMatrix();
    glTranslatef(2.0, 6.0, 0);
    glutSolidSphere(0.7, 250, 250);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(0.0, 6.5, 0);
    glutSolidSphere(0.7, 250, 250);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(3.0, 6.5, 0);
    glutSolidSphere(0.7, 250, 250);
    glPopMatrix();

    glPopMatrix();
}

void cloud2()
{
    drawCircle(.415,.58,.05);
    drawCircle(.48,.6,.08);
    drawCircle(.57,.582,.06);
    drawCircle(.63,.58,.04);
}

void cloud3(){

    drawCircle(-.325,.585,.04);
    drawCircle(-.4,.6,.07);
    drawCircle(-.49,.592,.05);
    drawCircle(-.54,.58,.03);
}
```

The shapes for the clouds has been done by using circles that intersect with each other, drawn with the *drawCircle* and *glutSolidSphere* function.

Step 12: Defining the night and day sky’s color

```
void daySky(){
    //Day Sky
    glColor3f(0.000, 0.30, 0.700);
    glBegin(GL_POLYGON);
    glVertex2f(100, 100);
    glVertex2f(0, 100);
    glColor3f(0.000, 0.70, 1.000);
    glVertex2f(0, 55);
    glVertex2f(100, 55);
    glEnd();
}
void nightSky(){
    //Night Sky
    glColor3f(0.164, 0.164, 0.507);
    glBegin(GL_POLYGON);
    glVertex2f(100, 100);
    glVertex2f(0, 100);
    glColor3ub(84,107,171);
    glVertex2f(0, 55);
```

```
   glVertex2f(100, 55);
   glEnd();
}
```

Day sky and night sky has been defined using user-defined functions as shown above. They have been given a well-defined colour for day and night.

Step 13: Defining the motion of the clouds

```
void clouds(){
	//CLOUDS
	     glPushMatrix();
	     glTranslatef(cloudX1,80,0);
	     glScalef(2,2,0);
	     cloud1();
	     glPopMatrix();
	     glPushMatrix();
	     glTranslatef(cloudX2,56,0);
	     glScalef(27,30,0);
	     cloud2();
	     glPopMatrix();
	     glPushMatrix();
	     glTranslatef(cloudX3,68,0);
	     glScalef(2,2,0);
	     cloud1();
	     glPopMatrix();
	     glPushMatrix();
	     glTranslatef(cloudX4,80,0);
	     glScalef(27,30,0);
	     cloud3();
	     glPopMatrix();
	     glPushMatrix();
	     glTranslatef(cloudX5,56,0);
	     glScalef(27,30,0);
	     cloud3();
	     glPopMatrix();
	     glPushMatrix();
	     glTranslatef(cloudX6,68,0);
	     glScalef(27,30,0);
	     cloud2();
	     glPopMatrix();

}
```

In this function, we have defined the motion of the clouds accordingly using predefined functions for the clouds created earlier.

Step 14: Defining the motion of the clouds

```
void dayView(){
	   daySky();
   /*glPushMatrix();
   glScaled(1,1,0);
   glTranslatef(50.0f, 50.0f, 0);
   circle1();
   glPopMatrix();*/
```

```
        //SUN
        glPushMatrix();
        glScaled(1,1,0);
        glTranslatef(50.0f, 50.0f, 0);
        glRotatef(speed*0.5,0,0,1);
        glTranslatef(50, 0, 0);
        glColor3ub(253, 184, 19);
        drawCircle(0,0,3);
        glPopMatrix();
        glColor3ub(255,255,255);
        clouds();

    }
void nightView(){
            nightSky();
            Stars();
            /*glPushMatrix();
               glScaled(1,1,0);
               glTranslatef(50.0f, 50.0f, 0);
               circle1();
               glPopMatrix();*/
        glPushMatrix();
               glScaled(1,1,0);
               glTranslatef(50.0f, 50.0f, 0);
               glRotatef(speed*0.5,0,0,1);
               glTranslatef(50, 0, 0);
               glColor3ub(246, 241, 213);
               drawCircle(0,0,3);
               glPopMatrix();
               glColor3ub(135,136,156);
               clouds();
}
```

Now in these user-defined functions, we call the day and night view accordingly with all the scenery elements as discussed in the previous steps.

Step 15: Defining the car

```
void car1(){
        glPushMatrix();
             glTranslatef(7.0,-10.0,0.0);
             glScalef(1.5,1.5,1);
             //Center Of Car
    glColor3f(0.000, 0.000, 1.000);
    glBegin(GL_POLYGON);
    glVertex2f(30, 50);
    glVertex2f(40, 50);
    glVertex2f(40, 40);
    glVertex2f(30,40);
    glEnd();
    glColor3f(0.000, 0.000, 1.000);
    glBegin(GL_POLYGON);
    glVertex2f(30, 50);
    glVertex2f(29.5, 49);
    glVertex2f(29.5, 41);
    glVertex2f(30,40);
    glEnd();
    glColor3f(0.000, 0.000, 1.000);
```

```
glBegin(GL_POLYGON);
glVertex2f(40, 50);
glVertex2f(40.5, 50-1);
glVertex2f(40.5, 40+1);
glVertex2f(40,40);
glEnd();
//Top Center
glColor3f(0.000, 0.000, 0.7);
glBegin(GL_POLYGON);
glVertex2f(30.15, 50);
glVertex2f(39.85, 50);
glColor3f(0.000, 0.000, 1.000);
glVertex2f(37.5+1,56);
glVertex2f(32-1,56);
glEnd();
//Top Center Window
glColor3f(0.773, 0.902, 0.98);
glBegin(GL_POLYGON);
glVertex2f(31.15, 51);
glVertex2f(38.85, 51);
glColor3f(0.141, 0.82, 1.0);
glVertex2f(36.5+0.75+1,56-0.5);
glVertex2f(33-0.5-1,56-0.5);
glEnd();
//Left Lights
glColor3f(1.000, 0.000, 0.000);
glBegin(GL_POLYGON);
glVertex2f(29.75, 48);
glVertex2f(31.5,48);
glVertex2f(31.5,46.5);
glVertex2f(29.75,46.5);
glEnd();
glColor3f(1.000, 1.000, 0.000);
glBegin(GL_POLYGON);
glVertex2f(32.25, 48);
glVertex2f(31.5,48);
glVertex2f(31.5,46.5);
glVertex2f(32.25,46.5);
glEnd();
//Right Lights
glColor3f(1.000, 0.000, 0.000);
glBegin(GL_POLYGON);
glVertex2f(40.25, 48);
glVertex2f(38.5,48);
glVertex2f(38.5,46.5);
glVertex2f(40.25,46.5);
glEnd();
glColor3f(1.000, 1.000, 0.000);
glBegin(GL_POLYGON);
glVertex2f(38.5, 48);
glVertex2f(37.75,48);
glVertex2f(37.75,46.5);
glVertex2f(38.5,46.5);
glEnd();
//Center Plate
glColor3f(1.000, 1.000, 1.000);
glBegin(GL_POLYGON);
glVertex2f(34, 41);
glVertex2f(36,41);
glVertex2f(36,43);
```

```
    glVertex2f(34,43);
    glEnd();

    //Left Wheel
    glColor3f(0.500, 0.500, 0.500);
    glBegin(GL_POLYGON);
    glVertex2f(31-0.5, 40);
    glVertex2f(33-0.5,40);
    glVertex2f(33-0.5,37);
    glVertex2f(31-0.5,37);
    glEnd();

    //Right Wheel
    glColor3f(0.500, 0.500, 0.500);
    glBegin(GL_POLYGON);
    glVertex2f(40-0.5, 40);
    glVertex2f(38-0.5,40);
    glVertex2f(38-0.5,37);
    glVertex2f(40-0.5,37);
    glEnd();
    glPopMatrix();
}
```

We have defined the front and back view of the car replete with the elements. In the front view, we have implemented the hood, front window, front headlights, wipers, front grille and the wheels. For the back view, it contains the back glass, back lights, and the wheels.

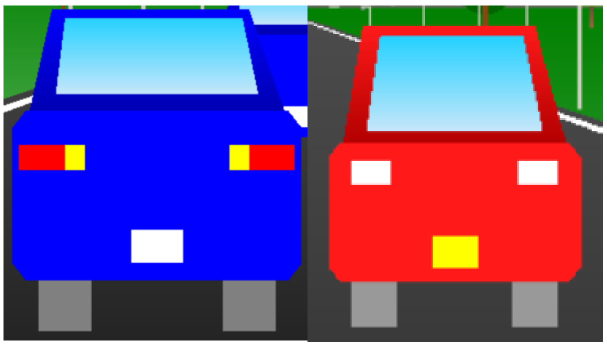

Fig 9 Back view and front view of the car

Step 15: Defining the motion of the car

```
float cartranslateX2=0.56,x2=-70+70,cartranslateY2=0.575,y2=-65+71.875,carScaleX2=0.007,sx2=1.5-
0.875,carScaleY2=0.007,sy2=1.5-0.875;

void carAnimation2(){
if(x2>39 || y2>49){
x2=-70,y2=-65,sx2=1.5,sy2=1.5;
}
x2+=cartranslateX2;
y2+=cartranslateY2;
sx2-=carScaleX2;
sy2-=carScaleY2;
}
```

In this, the car translates through the road and the size reduces at the same time because of the object going far from the viewpoint.

Step 16: The final functions

```
void display()
{
   glClear(GL_COLOR_BUFFER_BIT);
   if(dn%2==0){
      colorroadTop[0]=0.100,colorroadTop[1]=0.100,colorroadTop[2]=0.100;
           colorroadBot[0]=0.098,colorroadBot[1]=0.098,colorroadBot[2]=0.098;
           colorBuildWindow[0]=0.5,colorBuildWindow[1]= 0.5,colorBuildWindow[2]= 0.0;
           colorgrass[0]=0.000,colorgrass[1]= 0.200,colorgrass[2]= 0.000;
           colorlamp[0]=255,colorlamp[1]=255,colorlamp[2]=255;
           nightView();
           }
   else{
      colorroadTop[0]=0.300,colorroadTop[1]=0.300,colorroadTop[2]=0.300;
           colorroadBot[0]=0.098,colorroadBot[1]=0.098,colorroadBot[2]=0.098;;
           colorBuildWindow[0]=0.133,colorBuildWindow[1]= 0.243,colorBuildWindow[2]= 0.259;
           colorgrass[0]=0.000, colorgrass[1]=0.500,colorgrass[2]= 0.000;
           colorlamp[0]=135,colorlamp[1]=136,colorlamp[2]=156;
           dayView();
           }
   cloudAnimation();
   cloudAnimation2();
   cloudAnimation3();
   cloudAnimation4();
   cloudAnimation5();
   cloudAnimation6();

   roadside();
   road();
   buildingsAll();
           glPushMatrix();
glTranslatef(51.3,47,0);
glScalef(0.07,0.11,0);
mount();
glPopMatrix();
           glPushMatrix();
glTranslatef(72,47,0);
glScalef(0.07,0.11,0);
mount();
glPopMatrix();
           glPushMatrix();
glTranslatef(82,47,0);
glScalef(0.07,0.11,0);
mount();
glPopMatrix();
           glPushMatrix();
glTranslatef(65,49.13,0);
glScalef(0.06,0.08,0);
mount();
glPopMatrix();

treebackground();
leftlamp();
rightlamp();
```

```
    glPushMatrix();
    glTranslatef(x,y,0);
    glScalef(sx,sy,0);
    car1();
    glPopMatrix();
    carAnimation();

    glPushMatrix();
    glTranslatef(x2,y2,0);
    glScalef(sx2,sy2,0);
    car1();
    glPopMatrix();
    carAnimation2();

glutSwapBuffers();
}
void init()
{
    glClearColor(1,1, 1, 0.2);
   glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0,100,0,100,-1,1);
    glMatrixMode(GL_MODELVIEW);
}
int main(int argc, char** argv)
{
    int mode=GLUT_RGB|GLUT_DOUBLE;
    glutInitDisplayMode(mode);
    glutInit(&argc, argv);
    glutInitWindowSize(1280,1080);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Car Game By Tino Raman & Arijeet");
    init();
    glutDisplayFunc(display);
    glutTimerFunc(10,frame,0);
    printf("MAIN: %d",speed);
    glutMainLoop();
    return 0;
}
```

Now it is time to implement all the above and get them rendered, the final output should be the same as in Figure 1.