ITE 1011 - COMPUTER GRAPHICS

A project on

DAY NIGHT LIGHTING USING COMPUTER GRAPHICS

Submitted By

RISHIK REDDY - 18BCE0249
PRAMOD M - 18BCE0339
ASHWIN SAI K - 18BCE0375

To

Prof. PRABHUKUMAR M

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

INTRODUCTION:

Computer graphics is the use of computers to create and manipulate pictures on a display device. It comprises of software techniques to create, store, modify and represent pictures. The process transforms and presents information in a visual form.

OpenGL (Open Graphics Library) is a cross-platform, hardware-accelerated, language-independent, industrial standard API for producing 3D (including 2D) graphics. In our project we use OpenGl to design a basic day night scenario using freeGlut libraries.

There are three sets of libraries used in OpenGl programs:

1. **Core OpenGL (GL):**

Consists of hundreds of commands, which begin with a prefix "gl" (e.g., glColor, glVertex, glTranslate, glRotate). The Core OpenGL models an object via a set of geometric primitives such as point, line and polygon.

2. **OpenGL Utility Library (GLU):**

Built on-top of the core OpenGL to provide important utilities (such as setting camera view and projection) and more building models (such as qradric surfaces and polygon tessellation). GLU commands start with a prefix "glu" (e.g., gluLookAt, gluPerspective).

3. **OpenGL Utilities Toolkit (GLUT)**:

OpenGL is designed to be independent of the windowing system or operating system. GLUT is needed to interact with the Operating System (such as creating a window, handling key and mouse inputs); it also provides more building models (such as sphere and torus). GLUT commands start with a prefix of "glut" (e.g., glutCreatewindow, glutMouseFunc). GLUT is platform independent, which is built on top of platform-specific OpenGL extension such as GLX for X Window System, WGL for Microsoft Window, and AGL, CGL or Cocoa for Mac OS.
Quoting from the opengl.org: "GLUT is designed for constructing small to medium sized OpenGL programs. While GLUT is well-suited to learning OpenGL and developing simple OpenGL applications, GLUT is not a full-featured toolkit so large applications requiring sophisticated user interfaces are better off using native window system toolkits. GLUT is simple, easy, and small." Alternative of GLUT includes SDL.

4. **OpenGL Extension Wrangler Library (GLEW):**

"GLEW is a cross-platform open-source C/C++ extension loading library.

GLEW provides efficient run-time mechanisms for determining which OpenGL extensions are supported on the target platform." Source and pre-build binary available at http://glew.sourceforge.net/. A standalone utility called "glewinfo.exe" (under the "bin" directory) can be used to produce the list of OpenGL functions supported by your graphics system.

MOTIVATION:

Why OpenGL?

The main reason to use OpenGL is:

1. It Encapsulates many basic functions of 2D/3D graphics
2. Think of it as high-level language (C++) for graphics
3. Precursor for DirectX, WebGL, Java3D etc.

OpenGL has three favorites among its several drawing primitives: points, segments (straight line segments) and triangles. This is not owing to some idiosyncrasy of its specification as an API, but for a deeper reason. The material values such as color, specified at a primitive's vertices, are apparently interpolated throughout its interior. So here, briefly, is why points, segments and triangles are favoured: they have the property that every point inside each can be unambiguously (or uniquely) represented in terms of its vertices. This makes it possible for values like color defined at the vertices to be unambiguously - therefore, automatically, by means of a program - interpolated throughout the primitive.

The main motivation behind the project is because we wanted to develop something that would create the spark to learn more. And that spark could only be generated if we made the early stages of learning interactive. This is just a basic one from which many other models could be built to teach the kids.

HIGHLIGHTS:

In our project we have designed a Day Night Lighting scenario. When we first run the code we just get the basic layout of the canvas. Then on pressing the "D" key or "N" key it changes to a day or night view respectively.

In the day view as we press the "S" key at first the color of the house is changed and simultaneously the position of the sun too. On continuing to press the "S" key the sun keeps changing its position. Furthermore on pressing the "p" key we have a person on the roof of the house and on pressing the "F" key we have two persons playing football. There is even rain generated on pressing "R".

In the night view on pressing the "M" key the moon first waxes from the crescent to a full moon an on further pressing the "M" key the moon then wanes to a crescent and finally disappears. Further more on pressing the "r" key we have rain.

On pressing the "B" key the view again changes to the default view where only the layout of the canvas is shown.

EXPERIMENTAL RESULTS:

CODE:

```
#include<stdio.h>
#include<GL/glut.h>
#include<math.h>
#include <stdlib.h>
# define M_PI 3.142
void border();
void tree1();
void color();
void tree2();
void circle1(GLfloat x,GLfloat y,GLfloat radius);
void moon2();
void sun1();
void sun2();
void rain();
void sun3();
void sun4();
void person2();
void person3();
void ball(GLfloat bx, GLfloat by);
void moveball();

void daycolorchange(void);
void modifynight();

void menu(int id) //creating the menu for the day , night and to quit the result window
{
        int n=0;
        while(n<1)
        {
        switch(id)
        {
        case 1:exit(0);// to exit from the result screen
                break;
        case 2:moon2();// to show the night time
           break;
        case 3:sun2();// to show the day time
                break;

        }
```

```
        n++;
        }
}


void border()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0,1.0,1.0);
    glBegin(GL_LINE_STRIP);
         glVertex2f(1000,400);//mountain lines
         glVertex2f(800,400);
         glEnd();

        glBegin(GL_LINE_STRIP);//mountain lines
         glVertex2f(400,400);
         glVertex2f(200,400);
         glEnd();


        glBegin(GL_LINE_STRIP);//mountain lines
         glVertex2f(75,400);
         glVertex2f(0,400);
         glEnd();
        glBegin(GL_LINE_STRIP);//mountain lines
        glVertex2f(0,400);
        glVertex2f(250,700);
        glVertex2f(420,500);
        glEnd();
        glBegin(GL_LINE_STRIP);
        glVertex2f(700,650);
        glVertex2f(750,700);
        glVertex2f(1000,400);
        glEnd();
        glBegin(GL_LINE_LOOP);
        glVertex2f(350,225);
        glVertex2f(350,50);
        glVertex2f(600,50);
        glVertex2f(600,225);
        glEnd();
         glBegin(GL_LINE_LOOP);//front roof
         glVertex2f(400,300);
```

```
 glVertex2f(325,225);
 glVertex2f(625,225);
 glVertex2f(550,300);
glEnd();
 glBegin(GL_LINE_LOOP);//front wall containing window
 glVertex2f(400,300);
 glVertex2f(550,300);
 glVertex2f(550,425);
 glVertex2f(400,425);
 glEnd();
  glBegin(GL_LINE_STRIP);//roof railing
  glVertex2f(390,475);
  glVertex2f(585,650);
  glVertex2f(585,600);
  glVertex2f(450,475);
  glEnd();
   glBegin(GL_LINE_STRIP);
         glVertex2f(550,425);
         glVertex2f(550,300);
         glVertex2f(725,475);//top side wall
         glVertex2f(725,600);
         glEnd();
          glBegin(GL_LINE_STRIP);
          glVertex2f(600,225);//side wall
          glVertex2f(600,50);
          glVertex2f(800,250);
          glVertex2f(800,420);
          glEnd();
           glBegin(GL_LINE_LOOP);
           glVertex2f(550,300);//side roof
           glVertex2f(725,475);
           glVertex2f(800,420);
           glVertex2f(625,225);
           glEnd();

            glBegin(GL_LINE_LOOP);
            glVertex2f(390,425);
            glVertex2f(390,475);
            glVertex2f(560,475);//top roof
            glVertex2f(560,425);
            glEnd();
            glBegin(GL_LINE_LOOP);
```

```
                glVertex2f(560,475);//top roof
                glVertex2f(730,650);
                glVertex2f(730,600);
                glVertex2f(560,425);
                glEnd();
                glBegin(GL_LINE_STRIP);//top roof
                glVertex2f(585,650);
                glVertex2f(730,650);
                                glEnd();
                                glBegin(GL_LINE_STRIP);
                                glVertex2f(585,600);//top roof
                                glVertex2f(685,600);
                                glEnd();
                                glBegin(GL_LINE_LOOP);
                                glVertex2f(425,350);//top window
                                glVertex2f(425,400);
                                glVertex2f(510,400);
                                glVertex2f(510,350);
                                glEnd();
                                glBegin(GL_LINE_STRIP);
                                glVertex2f(425,50);//door
                                glVertex2f(425,150);
                                glVertex2f(525,150);
                                glVertex2f(525,50);
                                glEnd();
                                tree1();
                                glFlush();
                                glutSwapBuffers();


}



void circle1(GLfloat x, GLfloat y, GLfloat radius)
 {
   float angle;
   glBegin(GL_POLYGON);
   for(int i=0;i<200;i++)
         {
      angle = i*2*(M_PI/100);
      glVertex2f(x+(cos(angle)*radius),y+(sin(angle)*radius));
```

```
   }
   glEnd();
 }

void tree1()
{
        glColor3f(1.0,1.0,1.0);
        glBegin(GL_LINE_LOOP);
        glVertex2f(100,250);
        glVertex2f(175,250);
        glVertex2f(175,75);
        glVertex2f(100,75);
        glEnd();
   glBegin(GL_LINE_STRIP);
        glVertex2f(100,250);
        glVertex2f(0,250);
        glVertex2f(75,350);
   glVertex2f(25,350);
   glVertex2f(100,425);
glVertex2f(50,425);
glVertex2f(140,500);
glVertex2f(225,425);
glVertex2f(175,425);
glVertex2f(250,350);
glVertex2f(200,350);
glVertex2f(275,250);
glVertex2f(175,250);
glEnd();
}

void rain()
{
        glColor3f(1.0,1.0,1.0);
        int i;
        int rx,ry;
        for(i=0;i<500;i++)
        {
                        rx=rand()%1000;
                      ry=rand()%1000;
                        if(rx>=(50)&&rx<=(160))
                        if(ry>=990&&ry<=1000)
                                continue;
```

```
                        glBegin(GL_LINE_STRIP);
                                glVertex2f(rx-10,ry+10);
                                glVertex2f(rx,ry);
                        glEnd();

        }
        glFlush();
        glutSwapBuffers();

}
void tree2()
{
        glColor3f(0.3,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(100,250);
        glVertex2f(175,250);
        glVertex2f(175,75);
        glVertex2f(100,75);
        glEnd();
        glColor3f(0.0,0.3,0.01);

        glBegin(GL_POLYGON);
        glVertex2f(100,250);
        glVertex2f(0,250);
        glVertex2f(75,350);
glVertex2f(25,350);
glVertex2f(100,425);
glVertex2f(50,425);
glVertex2f(140,500);
glVertex2f(225,425);
glVertex2f(175,425);
glVertex2f(250,350);
glVertex2f(200,350);
glVertex2f(275,250);
glVertex2f(175,250);
glEnd();
}
void moon1(void);
void nightmode()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0,1.0,1.0);
```

```
glColor3f(0.0,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(0,400);

        glVertex2f(1000,400);
        glVertex2f(1000,1000);
        glVertex2f(0,1000);
        glEnd();//blue backround
        glColor3f(0.0,0.4,0.0);
        glBegin(GL_POLYGON);//ground color
        glVertex2f(0,400);
        glVertex2f(1000,400);
        glVertex2f(1000,0);
        glVertex2f(0,0);
        glEnd();
        glColor3f(1.0,1.0,1.0);
   circle1(50.0,700.0,2.0);
        circle1(150.0,750.0,1.0);
        circle1(550.0,800.0,1.0);
        circle1(600.0,750.0,1.0);
        circle1(450.0,600.0,1.0);
        //circle1(800.0,900.0,2.0);
        circle1(400.0,850.0,2.0);//exxtra
        //circle1(950.0,750.0,2.0);
        circle1(350.0,850.0,1.0);
        circle1(55.0,850.0,2.0);
circle1(65.0,900.0,2.0);
circle1(400.0,650.0,1.0);
circle1(200.0,800.0,2.0);


        glColor3f(0.2,0.1,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(0,400);//mountain
        glVertex2f(250,700);
        glVertex2f(500,400);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2f(500,400);
        glVertex2f(750,700);
        glVertex2f(1000,400);
        glEnd();
```

```
glColor3f(0.7,0.6,0.6);
        glBegin(GL_POLYGON);
        glVertex2f(350,225);//*front wall*//
        glVertex2f(350,50);
        glVertex2f(600,50);
        glVertex2f(600,225);
        glEnd();
        glColor3f(0.3,0.0,0.0);
         glBegin(GL_POLYGON);//front roof
         glVertex2f(400,300);
         glVertex2f(325,225);
         glVertex2f(625,225);
         glVertex2f(550,300);
         glEnd();
         glColor3f(0.7,0.6,0.6);
          glBegin(GL_POLYGON);
          glVertex2f(400,300);
          glVertex2f(550,300);
          glVertex2f(550,425);//top wall
          glVertex2f(400,425);
          glEnd();
          glColor3f(0.3,0.0,0.0);
           glBegin(GL_POLYGON);
           glVertex2f(390,475);//roof
           glVertex2f(585,650);
           glVertex2f(585,600);
           glVertex2f(450,475);
           glEnd();
           glColor3f(0.2,0.3,0.3);//top of roof
           glBegin(GL_POLYGON);
           glVertex2f(450,475);
           glVertex2f(585,600);
           glVertex2f(690,600);
           glVertex2f(560,475);
           glEnd();
           glColor3f(0.7,0.5,0.5);
            glBegin(GL_POLYGON);
                glVertex2f(550,425);
                glVertex2f(550,300);
                glVertex2f(725,475);//top side wall
                glVertex2f(725,600);
                glEnd();
```

```
glColor3f(0.7,0.5,0.5);
 glBegin(GL_POLYGON);
 glVertex2f(600,225);//side wall
 glVertex2f(600,50);
 glVertex2f(800,250);
 glVertex2f(800,420);
 glEnd();
 glColor3f(0.3,0.0,0.0);
  glBegin(GL_POLYGON);
  glVertex2f(550,300);//side roof
  glVertex2f(725,475);
  glVertex2f(810,420);
  glEnd();
  glColor3f(0.3,0.0,0.0);
  glBegin(GL_POLYGON);//part of side roof
  glVertex2f(600,223);
  glVertex2f(550,300);
  glVertex2f(810,420);
  glEnd();
  glColor3f(0.3,0.0,0.0);
  glBegin(GL_POLYGON);
  glVertex2f(585,650);
  glVertex2f(585,600);
  glVertex2f(700,600);
  glVertex2f(730,650);
  glEnd();
glColor3f(0.3,0.0,0.0);
   glBegin(GL_POLYGON);
   glVertex2f(390,425);
   glVertex2f(390,475);
   glVertex2f(560,475);//top roof
   glVertex2f(560,425);
   glEnd();
   glColor3f(0.3,0.0,0.0);
   glBegin(GL_POLYGON);
   glVertex2f(560,475);//top roof
   glVertex2f(730,650);
   glVertex2f(730,600);
   glVertex2f(560,425);
   glEnd();
   glColor3f(0.3,0.0,0.0);
   glBegin(GL_POLYGON);//top roof
```

```
                glVertex2f(585,650);
                glVertex2f(730,650);
                        glEnd();
                        glColor3f(0.3,0.0,0.0);
                        glBegin(GL_POLYGON);
                        glVertex2f(585,600);//top roof
                        glVertex2f(685,600);
                        glEnd();

                        glBegin(GL_POLYGON);
                        glVertex2f(425,350);//top window
                        glVertex2f(425,400);
                        glVertex2f(510,400);
                        glVertex2f(510,350);
                        glEnd();
                        glBegin(GL_POLYGON);
                        glVertex2f(425,50);//door
                        glVertex2f(425,150);
                        glVertex2f(525,150);
                        glVertex2f(525,50);
                        glEnd();


                        tree2();


                        }


void daycolorchange()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0,1.0,1.0);
        /*sky*/
   glColor3f(0.4,0.7,1.0);
        glBegin(GL_POLYGON);
        glVertex2f(0,400);
        glVertex2f(1000,400);
        glVertex2f(1000,1000);
        glVertex2f(0,1000);
        glEnd();
        /*ground*/
```

```
        glColor3f(0.0,0.6,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(0,400);
        glVertex2f(1000,400);
        glVertex2f(1000,0);
        glVertex2f(0,0);
        glEnd();
        /*hills*/
        glColor3f(0.3,0.1,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(0,400);
        glVertex2f(250,700);
        glVertex2f(500,400);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2f(500,400);
        glVertex2f(750,700);
        glVertex2f(1000,400);
        glEnd();
        /*1floor front wall*/
    glColor3f(0.8,0.8,0.4);
        glBegin(GL_POLYGON);
        glVertex2f(350,225);
        glVertex2f(350,50);
        glVertex2f(600,50);
        glVertex2f(600,225);
        glEnd();
        /*1floor roof*/
        glColor3f(0.5,0.0,0.0);
        glBegin(GL_POLYGON);
    glVertex2f(400,300);
        glVertex2f(325,225);
        glVertex2f(625,225);
        glVertex2f(550,300);
        glEnd();
        /*2floor front wall*/
        glColor3f(0.8,0.8,0.4);
    glBegin(GL_POLYGON);
        glVertex2f(400,300);
    glVertex2f(550,300);
        glVertex2f(550,425);
        glVertex2f(400,425);
```

```
glEnd();
/*roof*/
glColor3f(0.4,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(390,475);
glVertex2f(585,650);
glVertex2f(585,600);
glVertex2f(450,475);
glEnd();
/*top of roof*/
glColor3f(0.4,0.4,0.4);
glBegin(GL_POLYGON);
glVertex2f(450,475);
glVertex2f(585,600);
glVertex2f(690,600);
glVertex2f(560,475);
glEnd();

/*top side wall*/
glColor3f(0.7,0.7,0.32);
glBegin(GL_POLYGON);
glVertex2f(550,425);
glVertex2f(550,300);
glVertex2f(725,475);
glVertex2f(725,600);
glEnd();
/*bottom side wall*/
glColor3f(0.7,0.7,0.32);
glBegin(GL_POLYGON);
glVertex2f(600,225);
glVertex2f(600,50);
glVertex2f(800,250);
glVertex2f(800,420);
glEnd();
glColor3f(0.4,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(550,300);//side roof
glVertex2f(725,475);
glVertex2f(810,420);
glEnd();
glColor3f(0.4,0.0,0.0);
glBegin(GL_POLYGON);//part of side roof
```

```
        glVertex2f(600,223);
        glVertex2f(550,300);
        glVertex2f(810,420);
        glEnd();
        glColor3f(0.4,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(585,650);
        glVertex2f(585,600);
        glVertex2f(700,600);
        glVertex2f(730,650);
        glEnd();
glColor3f(0.5,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(390,425);
        glVertex2f(390,475);
        glVertex2f(560,475);
        glVertex2f(560,425);
        glEnd()
        /*top roof*/;
        glColor3f(0.4,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(560,475);
        glVertex2f(730,650);
        glVertex2f(730,600);
        glVertex2f(560,425);
        glEnd();
        glColor3f(0.4,0.0,0.0);
        glBegin(GL_POLYGON);//top roof
        glVertex2f(585,650);
        glVertex2f(730,650);
        glEnd();
        glColor3f(0.4,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(585,600);//top roof
glVertex2f(685,600);
        glEnd();
        /*window*/

        glBegin(GL_POLYGON);
        glVertex2f(425,350);
        glVertex2f(425,400);
        glVertex2f(510,400);
```

```
        glVertex2f(510,350);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2f(425,50);//door
        glVertex2f(425,150);
        glVertex2f(525,150);
        glVertex2f(525,50);
        glEnd();
        tree2();
        //glColor3f(1.0,1.0,0.0);
        //circle1(500.0,800.0,35.0);//sun
        glColor3f(0.5,0.5,0.5);
        circle1(850.0,800.0,20.0);
        circle1(875.0,790.0,30.0);
        circle1(910.0,793.0,40.0);
   circle1(950.0,790.0,30.0);
        glColor3f(1.0,1.0,0.0);




}

 void nightcolorchange()
{

 glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0,1.0,1.0);
glColor3f(0.0,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(0,400);

        glVertex2f(1000,400);
        glVertex2f(1000,1000);
        glVertex2f(0,1000);
        glEnd();//blue backround
        glColor3f(0.0,0.4,0.0);
        glBegin(GL_POLYGON);//ground color
        glVertex2f(0,400);
        glVertex2f(1000,400);
        glVertex2f(1000,0);
        glVertex2f(0,0);
```

```
        glEnd();
        glColor3f(1.0,1.0,1.0);
   circle1(50.0,700.0,2.0);
        circle1(150.0,750.0,1.0);
        circle1(550.0,800.0,1.0);
        circle1(600.0,750.0,1.0);
        circle1(450.0,600.0,1.0);
        //circle1(800.0,900.0,2.0);
        circle1(400.0,850.0,2.0);//exxtra
        //circle1(950.0,750.0,2.0);
        circle1(350.0,850.0,1.0);
        circle1(55.0,850.0,2.0);
circle1(65.0,900.0,2.0);
circle1(400.0,650.0,1.0);
circle1(200.0,800.0,2.0);


        glColor3f(0.2,0.1,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(0,400);//mountain
        glVertex2f(250,700);
        glVertex2f(500,400);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2f(500,400);
        glVertex2f(750,700);
        glVertex2f(1000,400);
        glEnd();
glColor3f(0.8,0.8,0.4);
        glBegin(GL_POLYGON);
        glVertex2f(350,225);//*front wall*//
        glVertex2f(350,50);
        glVertex2f(600,50);
        glVertex2f(600,225);
        glEnd();
        glColor3f(0.3,0.0,0.0);
         glBegin(GL_POLYGON);//front roof
         glVertex2f(400,300);
         glVertex2f(325,225);
         glVertex2f(625,225);
         glVertex2f(550,300);
         glEnd();
```

```
glColor3f(0.8,0.8,0.4);
 glBegin(GL_POLYGON);
 glVertex2f(400,300);
 glVertex2f(550,300);
 glVertex2f(550,425);//top wall
 glVertex2f(400,425);
 glEnd();
 glColor3f(0.3,0.0,0.0);
  glBegin(GL_POLYGON);
  glVertex2f(390,475);//roof
  glVertex2f(585,650);
  glVertex2f(585,600);
  glVertex2f(450,475);
  glEnd();
  glColor3f(0.2,0.3,0.3);//top of roof
  glBegin(GL_POLYGON);
  glVertex2f(450,475);
  glVertex2f(585,600);
  glVertex2f(690,600);
  glVertex2f(560,475);
  glEnd();
  glColor3f(0.7,0.7,0.32);
   glBegin(GL_POLYGON);
         glVertex2f(550,425);
         glVertex2f(550,300);
         glVertex2f(725,475);//top side wall
         glVertex2f(725,600);
         glEnd();
         glColor3f(0.7,0.7,0.32);
          glBegin(GL_POLYGON);
          glVertex2f(600,225);//side wall
          glVertex2f(600,50);
          glVertex2f(800,250);
          glVertex2f(800,420);
          glEnd();
          glColor3f(0.3,0.0,0.0);
           glBegin(GL_POLYGON);
           glVertex2f(550,300);//side roof
           glVertex2f(725,475);
           glVertex2f(810,420);
           glEnd();
           glColor3f(0.3,0.0,0.0);
```

```
            glBegin(GL_POLYGON);//part of side roof
            glVertex2f(600,223);
            glVertex2f(550,300);
            glVertex2f(810,420);
            glEnd();
            glColor3f(0.3,0.0,0.0);
            glBegin(GL_POLYGON);
            glVertex2f(585,650);
            glVertex2f(585,600);
            glVertex2f(700,600);
            glVertex2f(730,650);
            glEnd();
glColor3f(0.3,0.0,0.0);
             glBegin(GL_POLYGON);
             glVertex2f(390,425);
             glVertex2f(390,475);
             glVertex2f(560,475);//top roof
             glVertex2f(560,425);
             glEnd();
             glColor3f(0.3,0.0,0.0);
             glBegin(GL_POLYGON);
             glVertex2f(560,475);//top roof
             glVertex2f(730,650);
             glVertex2f(730,600);
             glVertex2f(560,425);
             glEnd();
             glColor3f(0.3,0.0,0.0);
             glBegin(GL_POLYGON);//top roof
             glVertex2f(585,650);
             glVertex2f(730,650);
                          glEnd();
                          glColor3f(0.3,0.0,0.0);
                          glBegin(GL_POLYGON);
                          glVertex2f(585,600);//top roof
                          glVertex2f(685,600);
                          glEnd();

                          glBegin(GL_POLYGON);
                          glVertex2f(425,350);//top window
                          glVertex2f(425,400);
                          glVertex2f(510,400);
                          glVertex2f(510,350);
```

```
                glEnd();
                glBegin(GL_POLYGON);
                glVertex2f(425,50);//door
                glVertex2f(425,150);
                glVertex2f(525,150);
                glVertex2f(525,50);
                glEnd();


                tree2();

                }

GLfloat bx=900.0,by=280.0;
void moveball()
{
        ball(bx,by);
        if((bx==900.0 && by==280.0)||(bx==930.0 && by==260.0))
        {
                bx=870.0;
                by=280.0;
        }
        else if(bx==870.0 && by==280.0)
        {
                bx=930.0;
                by=260.0;
        }
}


GLfloat p=120.0,q=700.0;
void movesun()
{
        //glColor3f(1.0,1.0,0.0);
        //circle1(120.0,700.0,30.0);
    if(p<500.0)
         {
                p=p+20;
                        q=q+10;

        circle1(p,q,30.0);
        glFlush();
```

```
                glutSwapBuffers();
        }
        else if(p<900)
        {
                p=p+20;
                q=q-0.2;

      circle1(p,q,30.0);
      glFlush();
    glutSwapBuffers();
        }

      else if(p>800.0||q>900)
      {
                p=120.0;
                q=700.0;

      circle1(p,q,30.0);
      glFlush();
    glutSwapBuffers();

      }

      }



void person1()
{
                        glColor3f(0.8,0.8,0.4);
                        glBegin(GL_POLYGON);
                                circle1(550, 580.5, 15);
                        glEnd;
                        glColor3f(1.0,1.0,1.0);//man
                        glBegin(GL_LINE_STRIP);
                        glVertex2f(550,565.5);
                        glVertex2f(550,535.5);
                        glVertex2f(550,535.5);
                        glVertex2f(530,560.5);
                        glVertex2f(550,535.5);
                        glVertex2f(570,560.5);
                        glVertex2f(550,535.5);
```

```
                glVertex2f(550,505);
                glVertex2f(550,505);
                glVertex2f(530,480);
                glVertex2f(550,505);
                glVertex2f(570,480);
                glEnd();

}

void person2()
{
                glColor3f(0.8,0.8,0.4);
                glBegin(GL_POLYGON);
                        circle1(850, 380.5, 15);
                glEnd();
                glColor3f(1.0,1.0,1.0);
                glBegin(GL_LINE_STRIP);
                glVertex2f(850,365.5);
                glVertex2f(850,335.5);
                glVertex2f(850,335.5);
                glVertex2f(830,360.5);
                glVertex2f(850,335.5);
                glVertex2f(870,360.5);
                glVertex2f(850,335.5);
                glVertex2f(850,305);
                glVertex2f(850,305);
                glVertex2f(830,280);
                glVertex2f(850,305);
                glVertex2f(870,280);
                glEnd();
}

void person3()
{
                glColor3f(0.8,0.8,0.4);
                glBegin(GL_POLYGON);
                        circle1(950, 360.5, 15);
                glEnd();
                glColor3f(1.0,1.0,1.0);
                glBegin(GL_LINE_STRIP);
                glVertex2f(950,345.5);
                glVertex2f(950,315.5);
```

```
                glVertex2f(950,315.5);
                glVertex2f(930,340.5);
                glVertex2f(950,315.5);
                glVertex2f(970,340.5);
                glVertex2f(950,315.5);
                glVertex2f(950,285);
                glVertex2f(950,285);
                glVertex2f(930,260);
                glVertex2f(950,285);
                glVertex2f(970,260);
                glEnd();
}

void ball(GLfloat bx, GLfloat by)
{
        glColor3f(1.0,0,0);
        circle1(bx,by,10);
}



void keys(unsigned char key,int x,int y)
{
        if(key=='b'||key=='B')
                border();
        else if(key=='d'||key=='D')
                sun1();
        else if(key=='n'||key=='N')
                moon1();
        else if(key=='m')
        {   nightmode();
                modifynight();
        }
        else if(key=='M')
        {
          nightcolorchange();
          modifynight();
        }
        else if(key=='S')
        {
                daycolorchange();
                movesun();
```

```
	}
	else if(key=='r')
	{
		nightmode();
		modifynight();
		rain();
	}
	else if(key=='p')
	{
		sun3();
		movesun();
	}
	else if(key=='f')
	{
		sun4();
	}
	else if(key=='R')
	{
		color();
		rain();
	}
}
void sun2()
{
	daycolorchange();
	glColor3f(1.0,1.0,0.0);
	circle1(120.0,700.0,30.0);
	glFlush();
	glutSwapBuffers();
}



void moon2()
{
	nightcolorchange();
	glColor3f(1.0,1.0,1.0);
				circle1(800.0,850.0,30.0);
				glColor3f(0.0,0.0,0.0);
				circle1(790.0,860.0,30.0);
				glFlush();
				glutSwapBuffers();
```

```
}

void moon1()
{
        nightmode();
        glColor3f(1.0,1.0,1.0);
                                circle1(800.0,850.0,30.0);
                                glColor3f(0.0,0.0,0.0);
                                circle1(790.0,860.0,30.0);
                                glFlush();
                                glutSwapBuffers();
}
void sun1()
{
   color();
        glColor3f(1.0,1.0,0.0);
        circle1(120.0,700.0,30.0);
        glFlush();
        glutSwapBuffers();
}

void sun3()
{
        color();
        person1();
        glColor3f(1.0,1.0,0.0);
        glFlush();
        glutSwapBuffers();

}

void sun4()
{
        color();
        person2();
        person3();
        moveball();
        glColor3f(1.0,1.0,0.0);
        circle1(120.0,700.0,30.0);
        glFlush();
        glutSwapBuffers();
```

```
}

void color()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0,1.0,1.0);
        /*sky*/
   glColor3f(0.4,0.7,1.0);
        glBegin(GL_POLYGON);
        glVertex2f(0,400);
        glVertex2f(1000,400);
        glVertex2f(1000,1000);
        glVertex2f(0,1000);
        glEnd();
        /*ground*/
        glColor3f(0.0,0.6,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(0,400);
        glVertex2f(1000,400);
        glVertex2f(1000,0);
        glVertex2f(0,0);
        glEnd();
        /*hills*/
        glColor3f(0.3,0.1,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(0,400);
        glVertex2f(250,700);
        glVertex2f(500,400);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2f(500,400);
        glVertex2f(750,700);
        glVertex2f(1000,400);
        glEnd();
        /*1floor front wall*/
   glColor3f(0.7,0.6,0.6);
        glBegin(GL_POLYGON);
        glVertex2f(350,225);
        glVertex2f(350,50);
        glVertex2f(600,50);
        glVertex2f(600,225);
        glEnd();
```

```
        /*1floor roof*/
        glColor3f(0.5,0.0,0.0);
        glBegin(GL_POLYGON);
glVertex2f(400,300);
        glVertex2f(325,225);
        glVertex2f(625,225);
        glVertex2f(550,300);
        glEnd();
        /*2floor front wall*/
        glColor3f(0.7,0.6,0.6);
glBegin(GL_POLYGON);
        glVertex2f(400,300);
glVertex2f(550,300);
        glVertex2f(550,425);
        glVertex2f(400,425);
        glEnd();
        /*roof*/
        glColor3f(0.4,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(390,475);
        glVertex2f(585,650);
        glVertex2f(585,600);
        glVertex2f(450,475);
        glEnd();
        /*top of roof*/
        glColor3f(0.6,0.5,0.6);
        glBegin(GL_POLYGON);
        glVertex2f(450,475);
        glVertex2f(585,600);
        glVertex2f(690,600);
        glVertex2f(560,475);
        glEnd();
        /*top side wall*/
        glColor3f(0.7,0.5,0.5);
        glBegin(GL_POLYGON);
        glVertex2f(550,425);
        glVertex2f(550,300);
        glVertex2f(725,475);
        glVertex2f(725,600);
        glEnd();
        /*bottom side wall*/
        glColor3f(0.7,0.5,0.5);
```

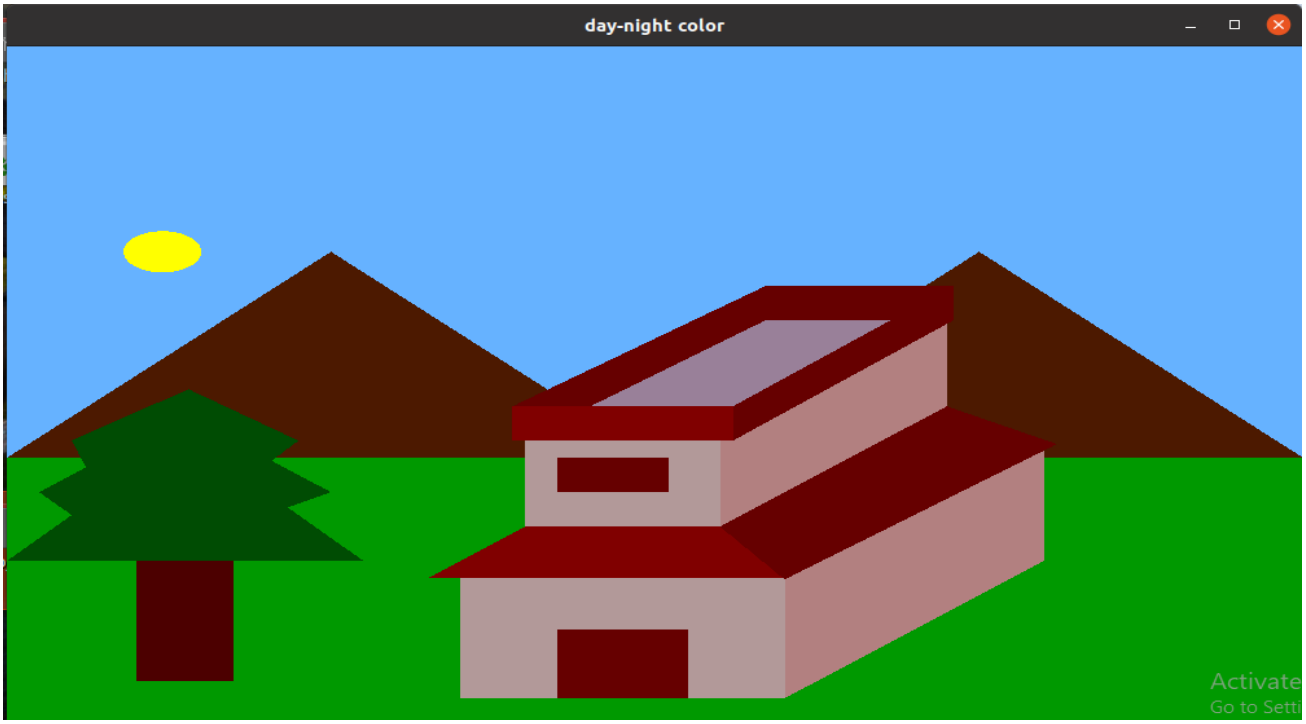
```
        glBegin(GL_POLYGON);
        glVertex2f(600,225);
        glVertex2f(600,50);
        glVertex2f(800,250);
        glVertex2f(800,420);
        glEnd();
        glColor3f(0.4,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(550,300);//side roof
        glVertex2f(725,475);
        glVertex2f(810,420);
        glEnd();
        glColor3f(0.4,0.0,0.0);
        glBegin(GL_POLYGON);//part of side roof
        glVertex2f(600,223);
        glVertex2f(550,300);
        glVertex2f(810,420);
        glEnd();
        glColor3f(0.4,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(585,650);
        glVertex2f(585,600);
        glVertex2f(700,600);
        glVertex2f(730,650);
        glEnd();
    glColor3f(0.5,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(390,425);
        glVertex2f(390,475);
        glVertex2f(560,475);
        glVertex2f(560,425);
        glEnd()
        /*top roof*/;
        glColor3f(0.4,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(560,475);
        glVertex2f(730,650);
        glVertex2f(730,600);
        glVertex2f(560,425);
        glEnd();
        glColor3f(0.4,0.0,0.0);
        glBegin(GL_POLYGON);//top roof
```

```
        glVertex2f(585,650);
        glVertex2f(730,650);
        glEnd();
        glColor3f(0.4,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(585,600);//top roof

   glVertex2f(685,600);
        glEnd();
        /*window*/

        glBegin(GL_POLYGON);
        glVertex2f(425,350);
        glVertex2f(425,400);
        glVertex2f(510,400);
        glVertex2f(510,350);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2f(425,50);//door
        glVertex2f(425,150);
        glVertex2f(525,150);
        glVertex2f(525,50);
        glEnd();


        tree2();
        glColor3f(1.0,1.0,0.0);

        glColor3f(0.5,0.5,0.5);


        glColor3f(1.0,1.0,0.0);


}
GLint i=0;
GLfloat a=800.0,b=850.0,d=800.0,e=850.0;

GLint m=0,n=0,f=30.0;

void modifynight()
{
```

```
        glColor3f(1.0,1.0,1.0);
        circle1(a,b,30.0);
        glColor3f(0.0,0.0,0.0);
        circle1(d,e,f);
        //glTranslated(80.0,90.0,0.0);
         e=e+10.0;
         d=d-10.0;

   if(d<700||e>900)
        {
                f=f+15.0;
                d=820.0;
                e=900.0;//d=800;
                //e=850.0;
                if(f>85.0)
                {
                        f=30.0;
                        d=800.0;
                        e=850.0;
                }

        }
        glFlush();
        glutSwapBuffers();

}

void init()
{
        glClearColor(0.0,0.0,0.0,1.0);
        glColor3f(1.0,1.0,1.0);
        glPointSize(1.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0.0,999.0,0.0,999.0);
}

int main( int argc, char **argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
        glutInitWindowSize(1000,1000);
```

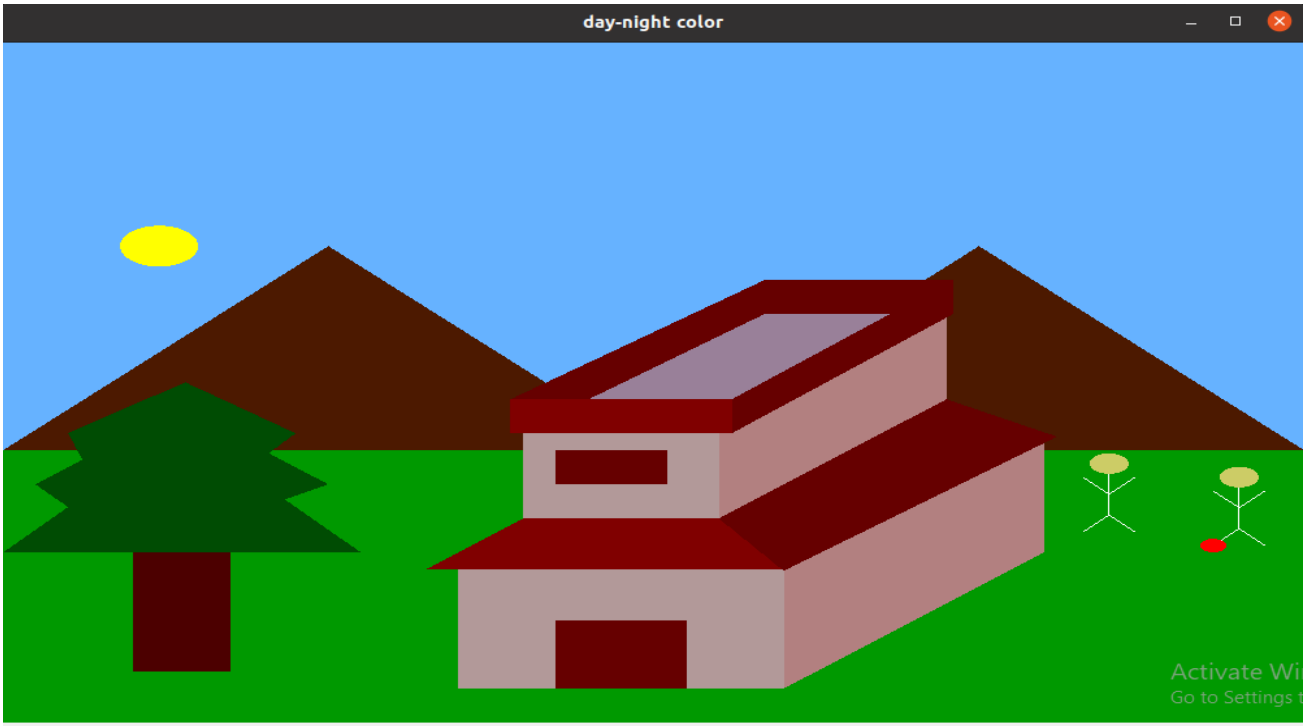
```
        glutInitWindowPosition(0,0);
        glutCreateWindow("day-night color");
        glutDisplayFunc(border);
        glutKeyboardFunc(keys);


        glutCreateMenu(menu);
        glutAddMenuEntry("quit",1);
        glutAddMenuEntry("night color change",2);
        glutAddMenuEntry("day color change",3);


        glutAttachMenu(GLUT_RIGHT_BUTTON);
        menu(GLUT_LEFT_BUTTON);
        init();
        glutMainLoop();
}
```

OUTPUT:

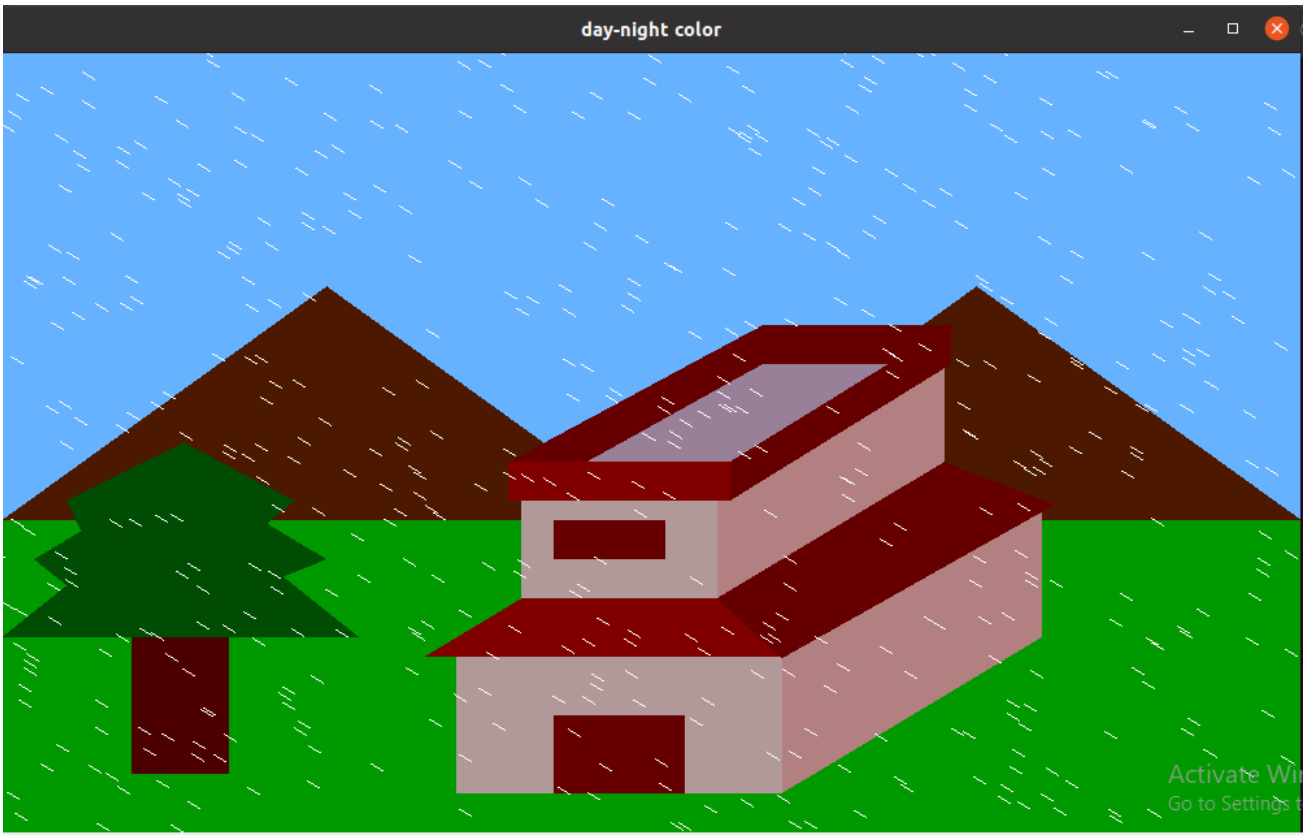


Day View

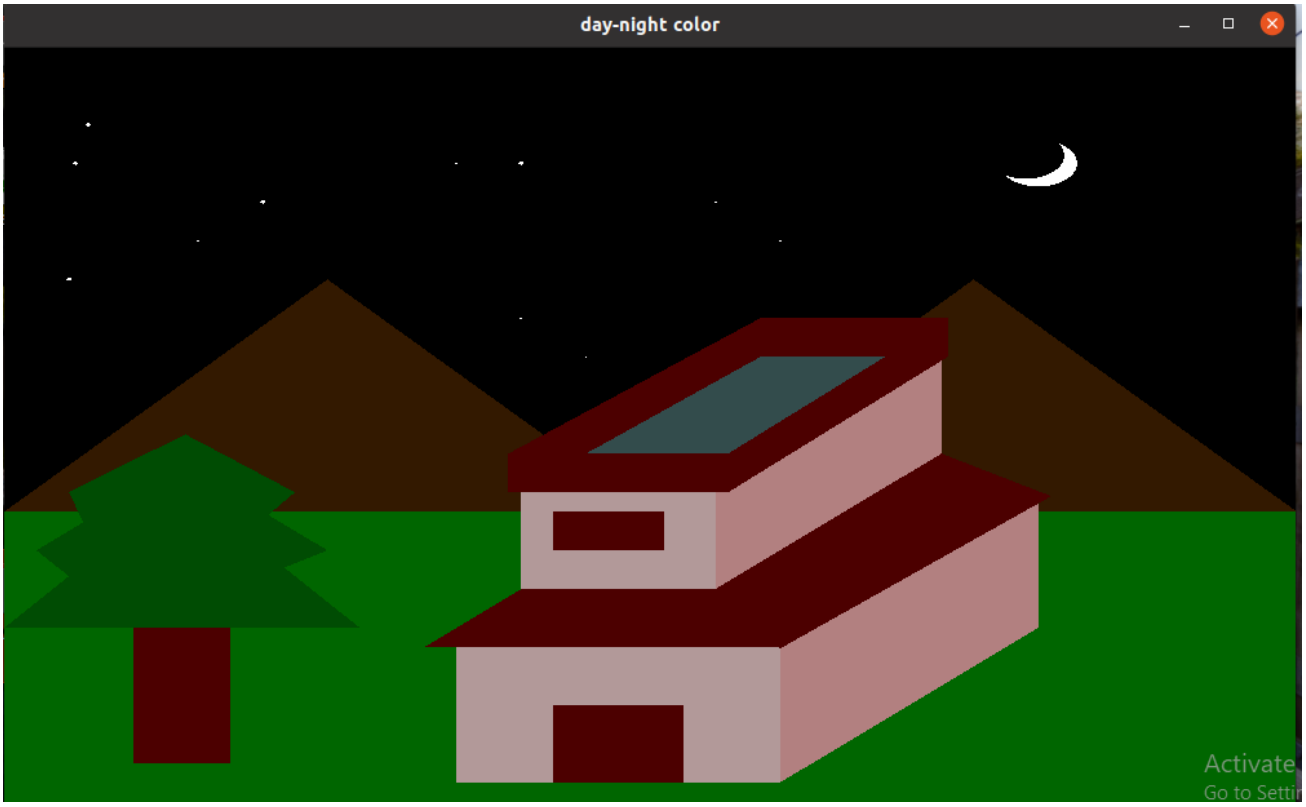
Day View with Person



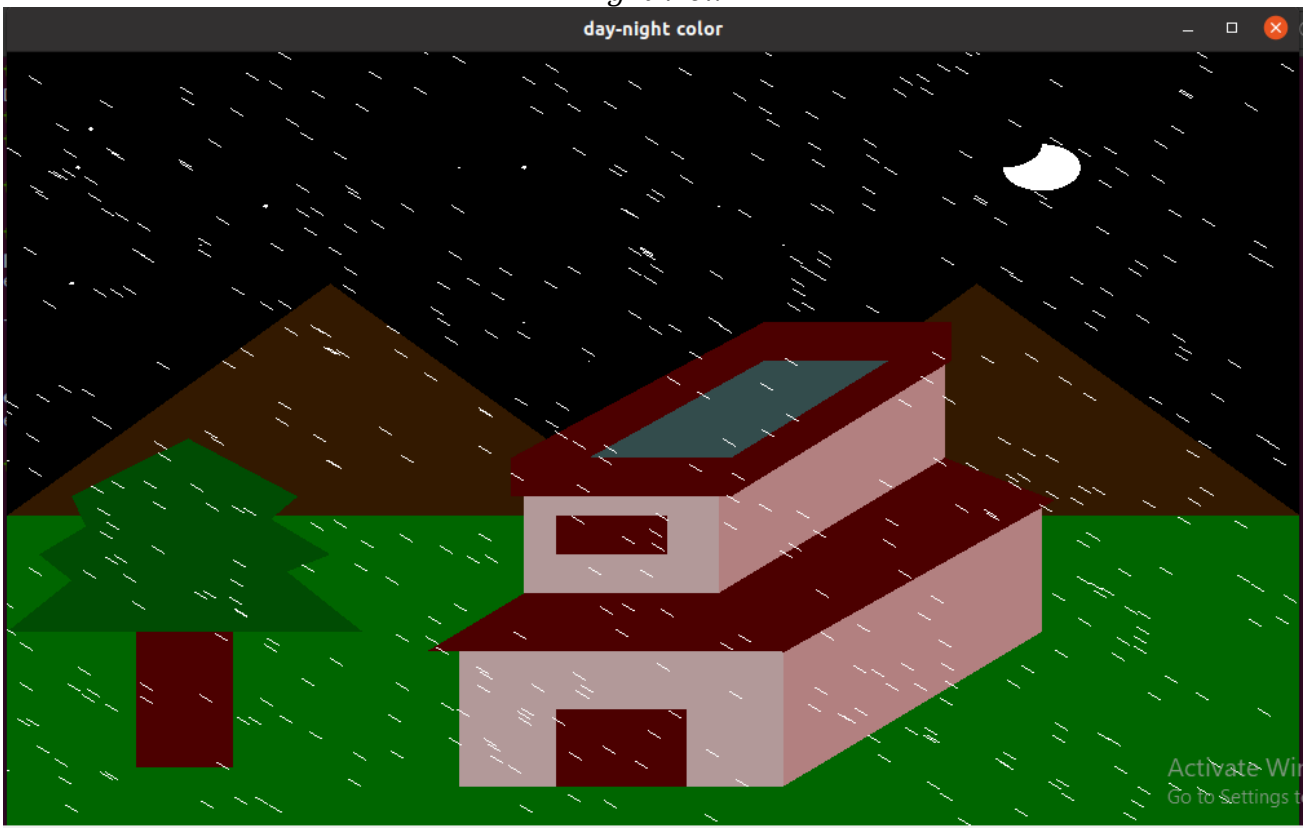
Day View with people playing Football



Day View with Rain



Night View



Night View with Rain

CONCLUSION:

In this way we were able to implement and learn few basic commands in OpenGL and draw basic figures and combine them to form a simple layout. We were also able to define functions to provide animations that make the sun move and moon wax and wane, etc.

We were also able to make it interactive by assigning functions to be called by pressing various keys which was the major goal of this project.

REFERENCES:

https://www.opengl.org/resources/libraries/

https://www.opengl.org/archives/resources/code/samples/glut_examples/examples/examples.html