**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**BELAGAVI-590014**



MINI PROJECT ENTITLED

**“Day-Night Color”**

For the academic year 2016-2017

Submitted by:

**ABHISHEK (1MV14CS006)**

Mini project carried out at

**Sir M. Visvesvaraya Institute of Technology**

**Bangalore-562157.**

Under the Guidance of

**Mrs. Monika Rani H G**

Asst. Professor, Dept. of CSE

Sir. MVIT, Bangalore.



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY**

**Hunasamaranahalli, Bangalore-562157.**

**SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY**

**BANGALORE-562157.**

(Affiliated to Visvesvaraya Technological University, Belagavi)

Bangalore-562157

**DEPARTMENT OF COMPUTER SCENCE AND ENGINEERING**



**CERTIFICATE**

Certified that the mini project work entitled **“DAY-NIGHT COLOR”** is a bonafide work carried out by **ABHISHEK (1MV14CS006),** in partial fulfillment for the award of Degree of **Bachelor of Engineering** in **Computer Science** **Engineering** of the **Visvesvaraya Technological University, Belagavi** during the year 2016-2017 in **Computer Graphics and Visualization Laboratory**. The mini project report has been approved as it satisfies the academic requirements in respect of the mini project work prescribed for the course of Bachelor of Engineering Degree.

Signature of the Guide Signature of the HOD

**Mrs. Monika Rani H G** **Prof. Dilip K Sen** Asst. Prof, Dept of CSE (HOD, Dept. of CSE)

Sir MVIT Sir MVIT

External Examiner Internal Examiner

**ACKNOWLEDGEMENT**

We would like to express our immense gratitude to all the people who have helped us and supported us in our zeal for the completion of this mini project. The success of this mini project would not have been certain if it weren’t for their help, whose constant guidance and support encourages us in our stride towards a better future..

We would like to thank **Prof. Dilip K. Sen, H.O.D.** of Computer Science and Engineering Department, Sir MVIT, Bangalore, for supporting us and helping us out in every way. To VTU for having given us this wonderful opportunity, to explore for ourselves the various possibilities in the field of Computer Graphics, and showcase our dedication and talent.

We express our sincere gratitude to our internal guide **Mrs. Monika Rani H G**, Asst. Professor, Computer Science Department, Sir MVIT. We are thankful to her for clearing all our doubts and clearing all our queries and helping us in our understanding of the subject and thus, helping us in the success of our mini project.

We would also like to thank the entire faculty of the **Computer Science & Engineering Department** for their co-operation and suggestions without which the mini project wouldn’t have been complete.

And also not to forget mentioning our friends who have helped us in uplifting our zeal and giving a hand in the completion of our mini project.

We are truly grateful to the above mentioned people who have contributed in the completion of this Mini project.

-  ***Abhishek (1MV14CS006)***

**ABSTRACT**

A beautiful day consists of two important phases which is DAY Time & NIGHT Time. There are other important phases such as the Noon and Evening time which are all included under the Day and Night respectively since we tend to look in a broader way.

This mini project has tried to simulate the two different phases of the day and night including different objects in the environment which can be normally considered while drawing a particular scenery consisting of a Tree, a house and Mountains in the background.

**INDEX**

**SL. No.** **Title** **Page No.**

1. **Introduction 7**

**2. Literature Survey 8-9**

**3. System Requirements Specifications 10**

**4. Design 11**

**5. Implementation 12-31**

**6. Important Functions 32-34**

**7. Screenshots 35-39**

**8. Conclusion 40**

**9. Future Enhancement 41**

**Bibliography 42**

**List of Figures**

**Figure no Name Page No.**

Fig 4.1 Initial Output Window 35

Fig 4.2 Initial Output Window with Menu Option 35

Fig 4.3 Options in the Menu 36

Fig 4.4 Initial Day Time 36

Fig 4.5 Movement of Sun during Day time 37

Fig 4.6 Sun on the other side of the Window 37

Fig 4.7 Initial Night Time 38

Fig 4.8 A phase of the Moon 38

Fig 4.9 Instance of Night with Full Moon 39

Fig 4.10 Instance of Night with New Moon 39

**CHAPTER 1**

**INTRODUCTION**

The ancient Chinese proverb, “a picture is worth ten thousand words” became a cliché in our society. Graphics provides one of the most natural way of communication with the computer, since our highly developed 2D and 3D pattern recognition ability allows us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics thus permits extensive, high bandwidth user computer interaction. This significantly enhances our ability to understand data, to perceive trends and to visualize real and imaginary objects, indeed to create a “virtual world” that we can explore from arbitrary points and views. It makes communication more efficient, graphics makes possible higher quality and more precise results or products, greater productivity, and lower analysis and design cost.

OpenGL is a premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry’s most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 EARLY GRAPHICS SYSTEM**

Computer graphics started with pen plotter model. We had Cathode Ray Tube Display showing the graphics. Each line drawn was a result of intense calculation which was a huge overhead a few years back.

**2.2 OPENGL**

OpenGL is a standard specification defining cross platform API for writing applications that produce 2D and 3D graphics. It contains multiple different function calls that help develop complex graphics with help of simple primitives. Developed by Silicon Graphics Inc. in 1992. Now its managed by the nonprofit technology consortium, the Khronos Group.

OpenGL has a set of library that help in various functions. They are GL, GLU and GLUT. OpenGL Library or GL provides a powerful yet primitive set of commands. OpenGL Utility Library or GLU contains several routines that help setting up matrices for specific viewing orientation, mini projections and surface rendering. OpenGL Utility Toolkit Library or GLUT contains routines that help in windowing functions and is system independent.

**2.3 EXISTING SYSTEM**

The existing graphics systems were the graphics header in C/C++. These graphics system are not system independent. Moreover, the underlying hardware knowledge is important for proper working of the code. Moreover, only 2D graphics were supported. Complex graphics concepts like camera position, shading, 3D graphics, material properties were absent.

**2.4 PROPOSED SYSTEM**

To achieve 3D graphics effects, OpenGL software was made. Moreover, system hardware independence and cross platform support OpenGL became famous. OpenGL is more streamlined than other graphics system APIs. The concept of building from primitives made it widely accepted by developers. It even supports animations, function driven events, callback functions. The transformation functions provide a more powerful ability to graphic coders to design their dreams digitally.

**CHAPTER 3**

**SYSTEM REQUIREMENTS SPECIFICATIONS**

**3.1 HARDWARE REQUIREMENTS**

Minimum hardware requirement specifications are:

* Processor : Intel i3 4th Gen or better
* GPU : NVidia GT750M or better or Intel Iris Pro or better
* RAM : 1 GB or more
* HDD : 60 GB or more at 5400 RPM or better
* Keyboard : US English QWERTY keyboard
* Mouse : Normal working mouse
* Monitor : 800 x 600 or better

**3.2 SOFTWARE REQUIREMENTS**

Minimum software specifications are:

* OS : Ubuntu 16.06 ( Linux 4.6 ) or MacOS/OSX 10.7 or Windows 7 or better
* Latest NVidia and/or Intel Drivers for GPU
* Tools, IDE, Compilers:
* freeglut3-dev, freeglut3, g++, C++11 for Linux
* XCode 8.3, C++11 for MacOS/OSX
* Visual Studio 2017, glut files, Visual C++ latest one for Windows

**CHAPTER 4**

**DESIGN**

**4.1 STATEMENT OF PROBLEM**

A Normal simple Day is one that everyone craves for. In this problem we get to see the instance of a day. A day comprising of both the day time and night time along with different objects present in the environment and the effect of color on them considering day time and night time.

**4.2 OBJECTIVE OF THE PROBLEM**

In this program, we get to see the day time and night time through the options available in the menu. The initial output window contains the outlines of the different objects present in the created environment such as Hills, a house and a tree.

The menu consists of three options which are Quit- to quit the program, night color change- to change the mode to night mode and day color change- to change the mode to day mode. Also using the keyboard keys we can show the different positons of the Sun during day time. Also we can use keyboard operation to see the different phases of the moon during different nights. The menu is controlled with the help of the Right click whereas the user can control the movement of both sun and moon through keyboard keys.

**CHAPTER 5**

**5. IMPLEMENTATION**

**/\*void menu():** This function is used to create the menu consisting of different options such as day color change to switch to day time scenery as well as night color change to switch to night mode. It also has a quit option in order to close the output

window.\*/

void menu(int id)

{

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():** This function is used to create the outlines of the objects such as the

mountains and house in the scenery. This outlines are seen when the program

is executed.\*/

void border()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,1.0,1.0);

glBegin(GL\_LINE\_STRIP);

glVertex2f(1000,400);

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 tree1():** This function is used to create a tree outline with white color and is seen on

the starting output screen.\*/

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 tree2():** This function is used to show the tree in the scenery with colors in both day

and night mode.\*/

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 circle1**(): This function is used to create the Sun in the day mode as well as the Moon

in the night mode and also in portraying stars in the night sky. By using

different values as the argument we can create circles of different radius.\*/

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 dayColorChange():** This function is used to show the day time. When the option day

color change is clicked, this function gets executed. This function

shows the color of the objects during day time including Sun and

clouds.\*/

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(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():** This function is used to change the mode to night when the

option night color change is clicked. It shows the color of the

objects during night and also some other detailing regarding

night. \*/

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 background**

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(400.0,850.0,2.0); **//extra**

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();

}

**/\*void movesun():** This function is used to change the position of the sun thus portraying a

day time mode where sun rises from one side and sets on the other

side.\*/

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 color():** This function is used to provide the color to the day time mode. It comes in

handy when the screen is switched from one mode to another. It is used

inside other function in order to provide an error free look to the daytime.\*/

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);

}

**void modifynight():** This function is used to change the phases of the moon. This function is

called when a certain keyboard key is pressed.

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 nightmode():** This function is same as the nightColorChange() difference being the

change in the color of the house. The color of the house changes with

all the objects in the background being the same.

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); **//stars**

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(400.0,850.0,2.0);

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 keys():** This is used to change the mode to night mode or to day mode depending on the

user requirement.

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')

{

color();

movesun();

}

else if(key=='S')

{

dayColorChange();

movesun();

}

}

**CHAPTER 6**

**IMPORTANT FUNCTIONS**

**6.1 Headers defined:**

**#include<GL/glut.h> :** To include glut library files.

**#include<stdlib.h> :** To include standard library function.

**#include<stdio.h> :** To include standard input and output files.

**#include<math.h> :** To include files for mathematical operations.

**6.2 OpenGL functions:**

**glClearColor( ) :** Sets the RGBA Clear color used when clearing the color

buffer.

**glClear( ) :** Clears buffer to preset values. Specifies BITWISE OR

of mask that indicate the buffers to be cleared. The mask

used in the program is GL\_COLOR\_BUFFER\_BIT.

**glMatrixMode() :**  Specifies which matrix will be affected by subsequent

transformation. Mode can be GL\_MODELVIEW,

GL\_MINI PROJECTION etc.

**glLoadIdentity() :** Sets the current transformation matrix to an Identity Matrix.

**glutSwapBuffers( ) :** Swaps the buffers of the current window if double buffered.

**glFlush( ) :** Used for force execution of GL commands in finite time.

**glBegin( ) :** Used to delimit the vertices of a primitive or a group of like

primitives.

**glutKeyboardFunc( ) :** Used for calling the functions which are assigned for

Keyboard operation.

**glutMainLoop( ) :** Cause the program to enter an event-processing loop.

**gluOrtho2D( ) :** Defines a 2D orthographic mini projection matrix.

**glutCreateWindow( ) :**  Creates a top-level window. The name will be provided to the window system as the Window’s name.

**6.3 User Defined Functions :**

**void border( ) :** This function is used to draw the outlines of some

important parts of the scenery. The Lines seen in the

output window at first are drawn through this function.

**void color( ) :** This function is used along the movesun( ) function in

order give the perfect viewing for the viewer. If this part is

removed from the program then there will be case of

formation of number of suns as long as we use move sun.

**void movesun( ) :** This function is used to show the movement of the sun at

day time. It ends when the sun has reached the other end of

the window while using the keyboard in order to move the

sun.

**void tree1( ) :** This function is used to see the tree when the output screen

is opened. It only consists of the outline of the tree.

**void tree2( ) :** This function is used so that we can see the tree in the day

time mode.

**void nightmode( ) :** This function is used to depict the night time of the

program.

**void dayColorChange( ) :** This function is used to depict the day time of the

program.

**void modifynight( ) :** In order to show the different phases of the moon. This

function can be accessed through keyboard keys.

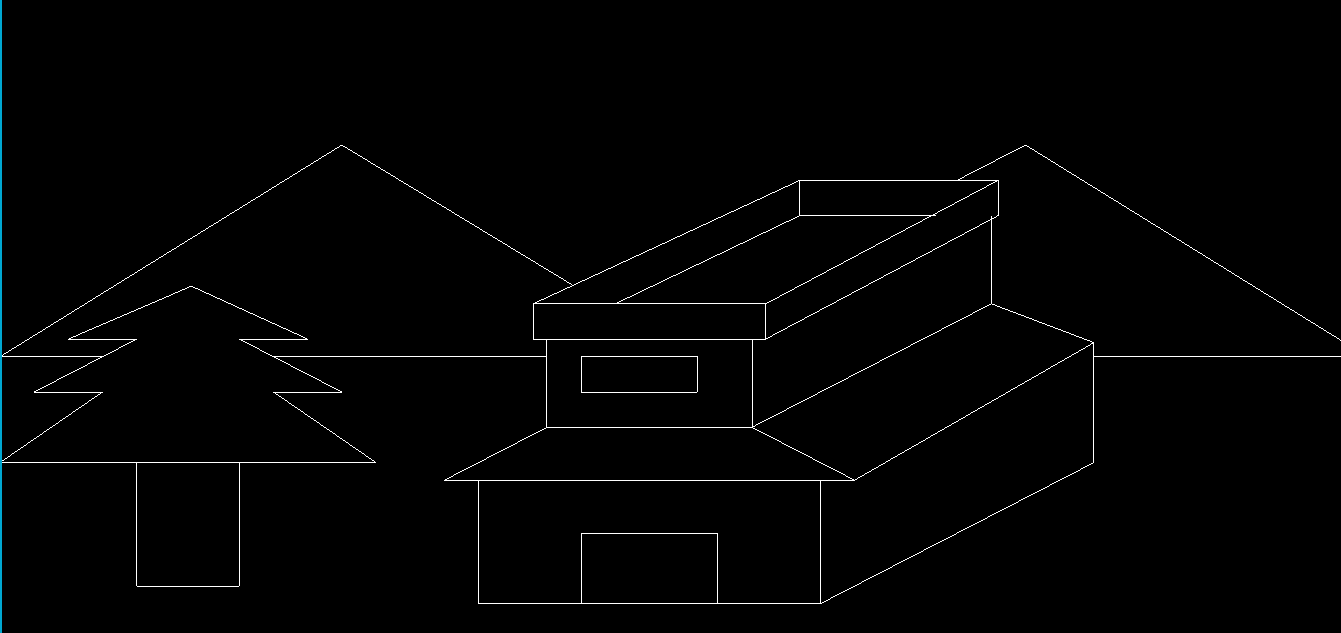
**void circle1( ) :** This function is used to create the circular objects

throughout the program.

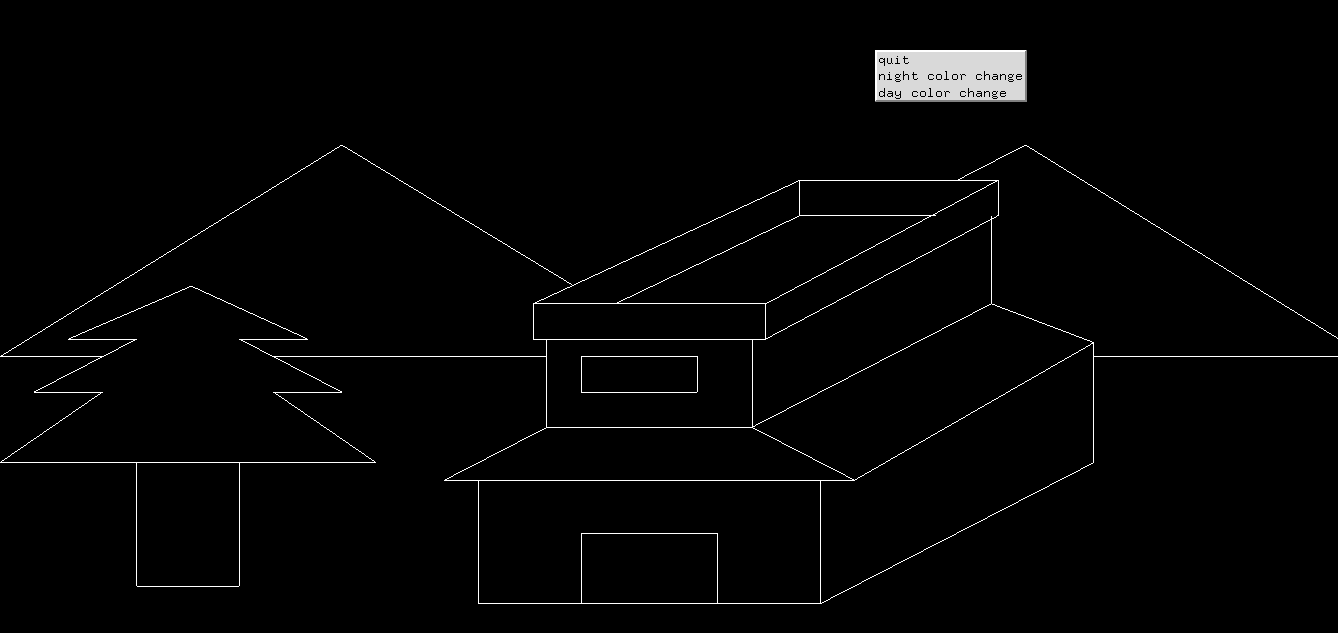
**CHAPTER 7**

**7. SCREENSHOTS**

**7.1 WELCOME SCREEN**

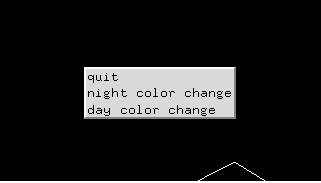
 *Figure 4.1 Initial Output Window*

**7.2 WELCOME SCREEN WITH MENU**



*Figure 4.2 Initial Output Window with Menu Option*

**7.3 OPTIONS IN MENU**

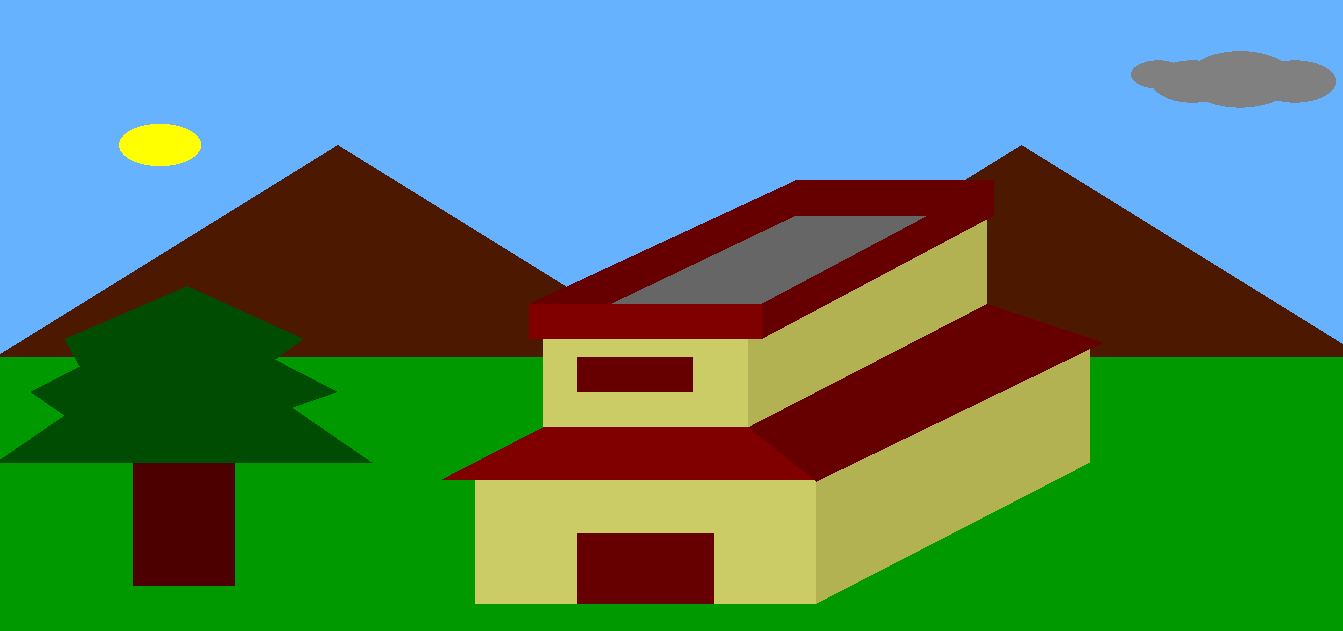
**

*Figure 4.3 Options in the Menu*

Figure 4.2 shows the Initial Output Window along with the menu when we **Right Click** on the Screen. Figure 4.3 shows the different options available in the menu which are

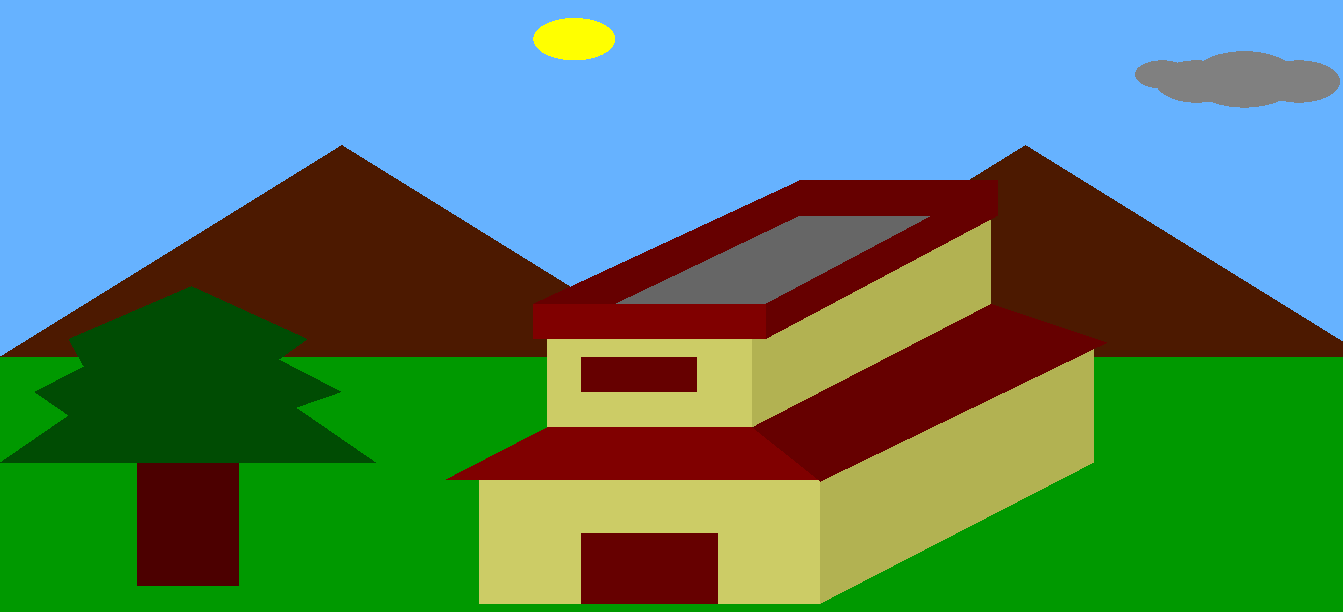
Quit, Night color change and Day color change. The options available can be used to view the different instances of a normal day.

**7.4 DAY TIME – CHANGING SUN’S POSITION**



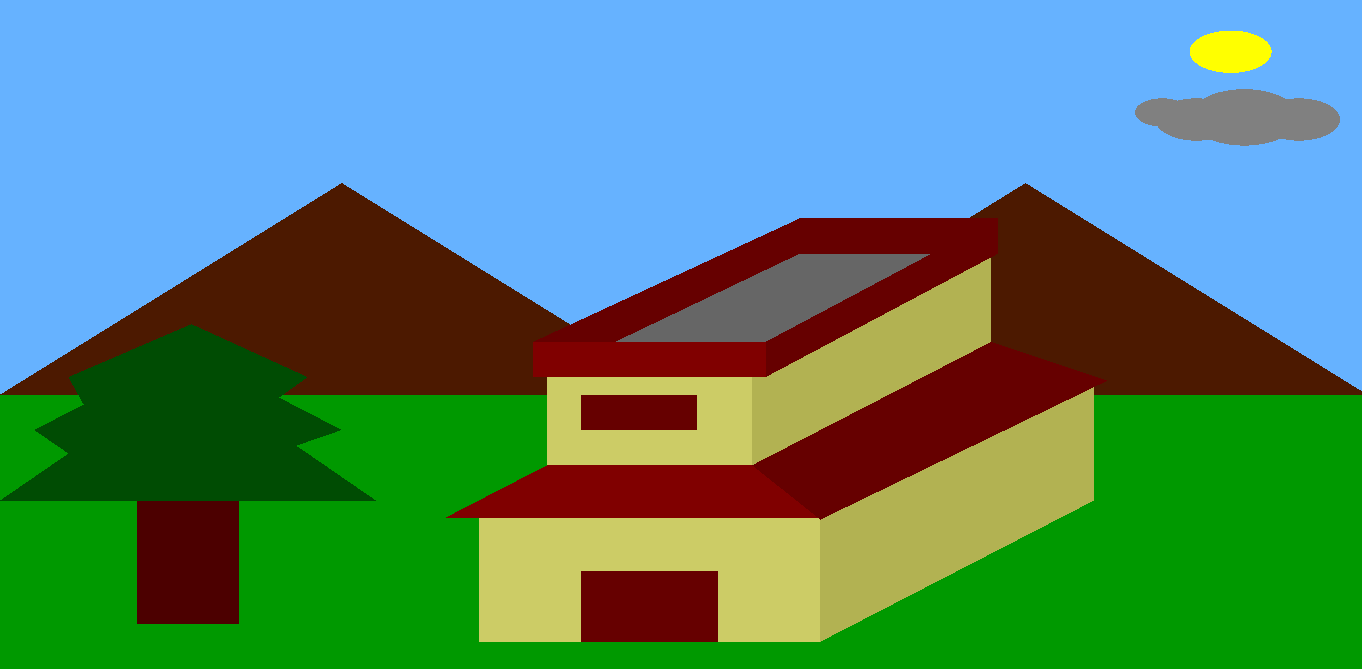
*Figure 4.4 Initial Day Time*

Figure 4.4 shows the initial output screen when the day color change option is selected from the menu. It shows the colors of the various objects with respect to day time perspective. Also there are two added objects , Sun and Cloud, which were not present in the initial black and white output screen.



*Figure 4.5 Movement of Sun during Day Time*

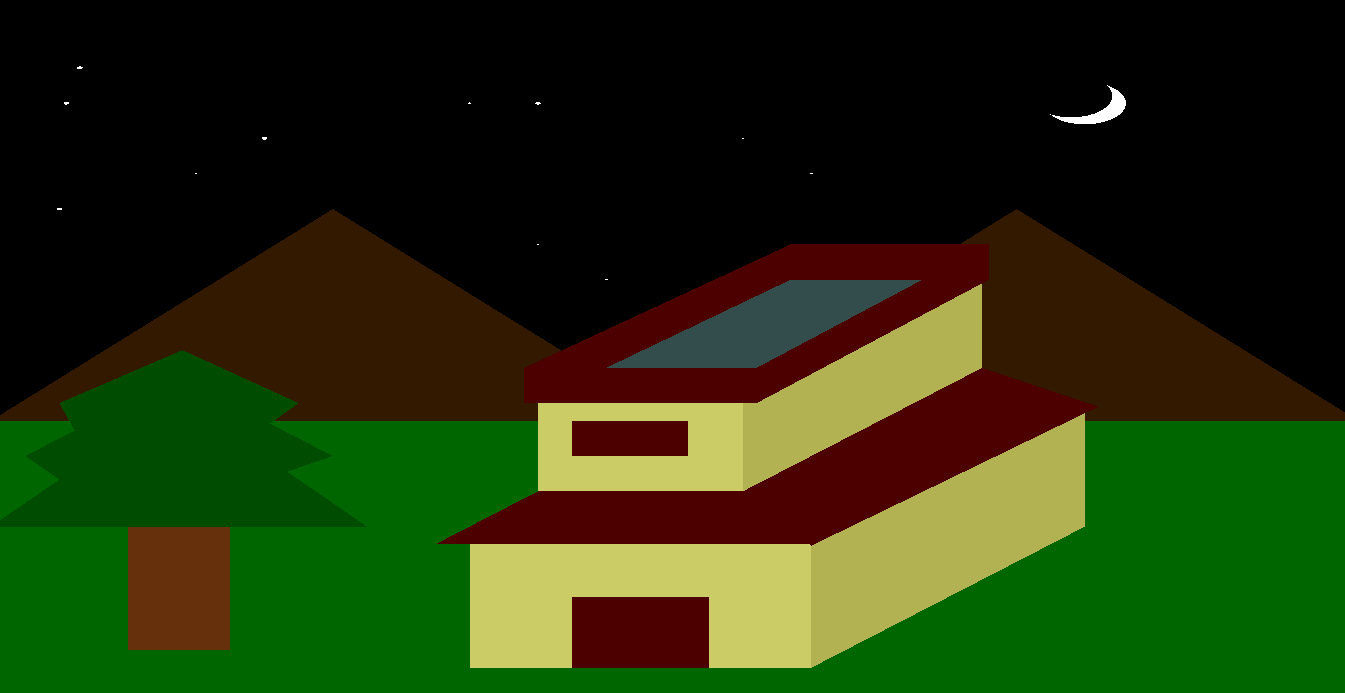
In order to show the different positions of the Sun throughout a Day, we use ‘**S**’ button or **Shift + S**. However this movement is in the presence of cloud. Figure 4.5 shows a particular instance of the day near to Noon.



*Figure 4.6 Sun on the other side of the Window*

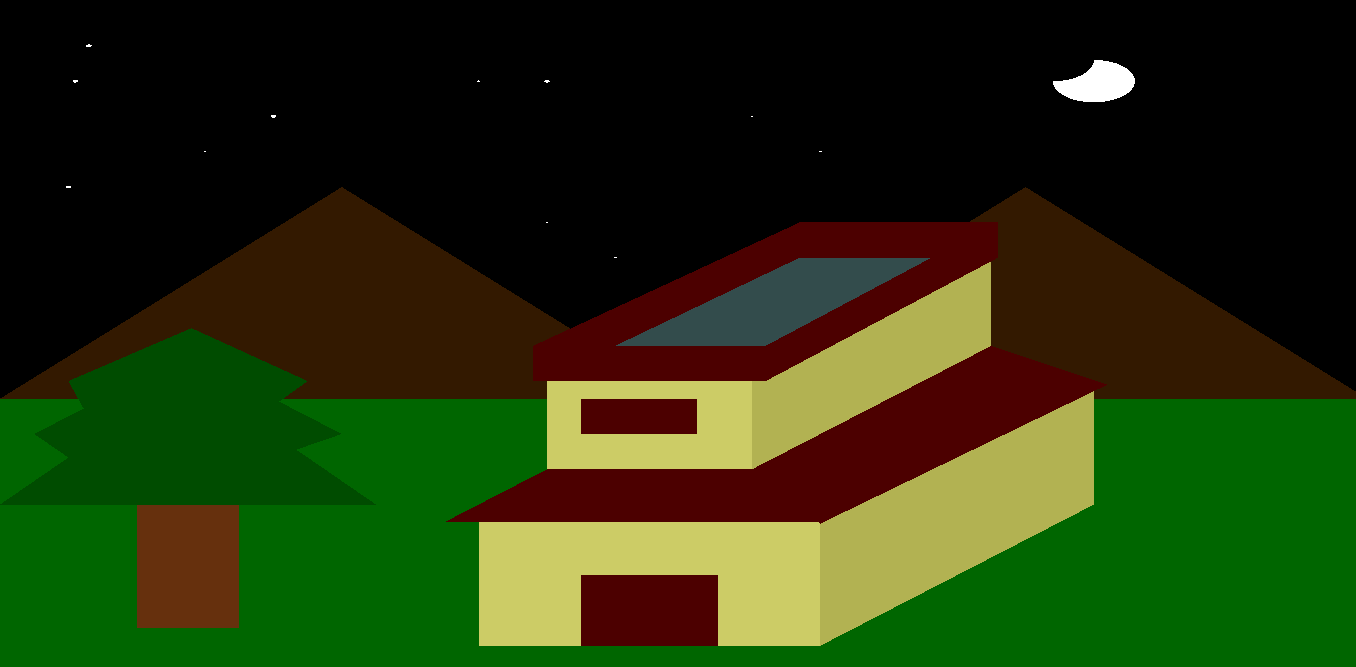
Figure 4.6 shows the range till which the sun can travel in the Window. The Sun rose from the Left Hand side of the window and sets at the other side of the window. Thus here it is depicting near Evening time.

**7.5 NIGHT TIME – PHASES OF THE MOON**

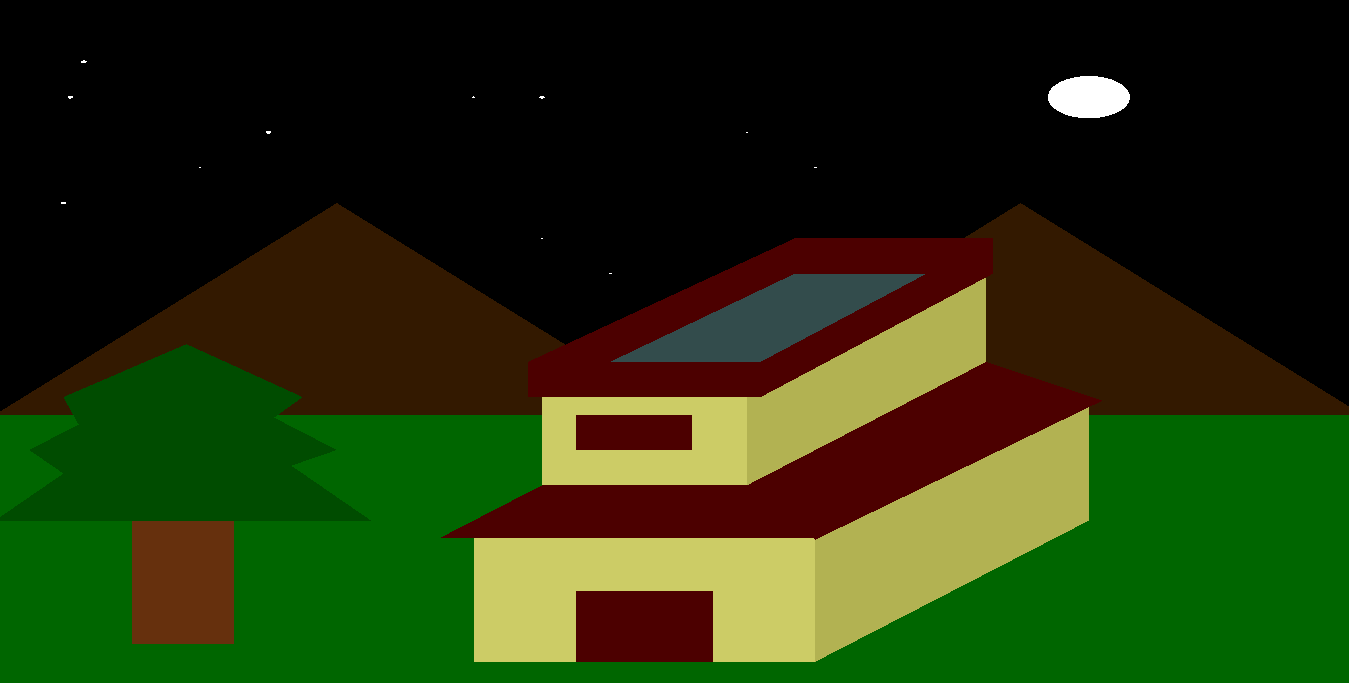


*Figure 4.7 Initial Night Time*

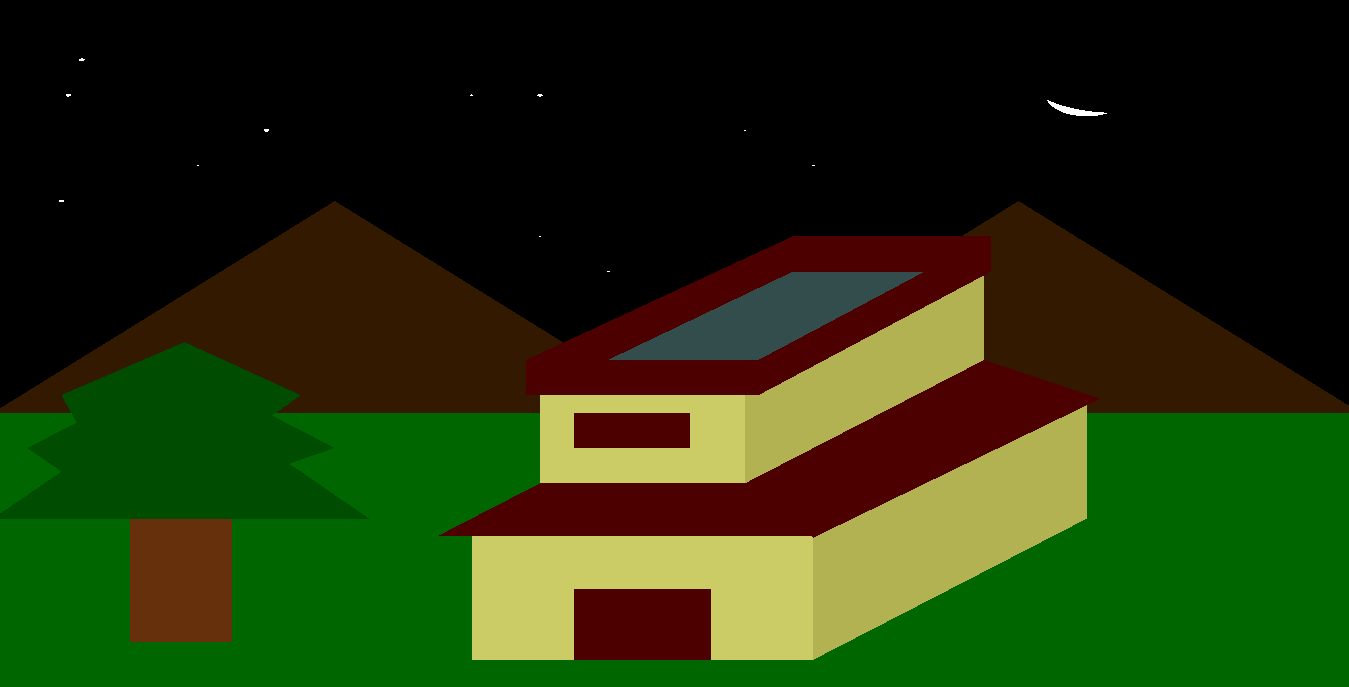
Figure 4.6 shows the initial output screen of Night Time with the stars and the Moon. It appears when the night color change option is selected from the Menu.



*Figure 4.8 A phase of the Moon*

**

*Figure 4.9 Instance of a Night with Full Moon*

**

*Figure 4.10 Instance of a Night with New Moon*

Figure 4.8 to 4.10 shows different phases of the Moon on different Nights. The phases of the Moon can be generated using **Shift + M** or by just pressing **M** key on the keyboard. Using ‘m’ will change the color of the house. Rest all the function is same as that of ‘M’.

**CHAPTER 8**

**CONCLUSION**

The mini mini project designed here provides the complete view of a normal day and night scene. Here we have designed a house, mountains, trees, the sun, the moon and the stars. We have implemented different functions in order to show the different instances of the Sun’s position during daytime as well as the phases of the Moon in different nights. In all the mini mini project provides a good view of the different instances of day time and night time.

This mini project helps me in learning how to use simple codes to build simple graphic content. These simple graphic contents can help render a complex set of graphical objects easily. It also helps me understand the mechanics of OpenGL programming, handling various functionalities to give the user a different experience.

**CHAPTER 9**

**FUTURE ENHANCEMENTS**

The mini mini project designed can be implemented in the future with more effects in lighting as well as functions. The Scene can be viewed in a 3D display. Further we can implement more user interaction as well as more functions which works according to System timing. The mini mini project can be implemented with the actual timing of the surrounding so that the color of the scene can be changed according to time.

It can be also implemented as a Wallpaper which changes color to inform the user about the time without looking for an actual clock. This mini project can also be implemented with the change of surroundings with every 4 seasons i.e. Spring, Summer, Rainy and Winter i.e. introduction of snow, rain etc. Thus in future this mini project has more scope in order to fulfill the viewing needs of the user.

**BIBLIOGRAPHY**

**1. Books:**

[1] Edward Angel, 2009, *Interactive computer graphics A top-down approach with OpenGL,*

5th edition, Addison-Wesley

[2] Donald Hearn and Pauline Baker : *Computer Graphics – OpenGL version,* 3rd edition,

Pearson Education , 2004

[3] Kilgard, M. J. , 1994, *An OpenGL Toolkit, SIGS* Publications.

[4] F.S. Hill Jr.: *Computer Graphics using OpenGL,* 3rd edition, PHI, 2009

**2. Websites :**

[1] [www.opengl.org/resources/](http://www.opengl.org/resources/)

[2] <http://pyopengl.sourceforge.net/documentation/manual-3.0/>

[3] [www.stackoverflow.com/search?q=opengl](http://www.stackoverflow.com/search?q=opengl)