VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590 018, Karnataka, India.

A MINI PROJECT ON

“DIURNAL CYCLE”

A Mini Project submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF ENGINEERING
In
COMPUTER SCIENCE & ENGINEERING

Submitted by

BILQUEES AYESHA SIDDIQUA	**1RG18CS009**
FAIQUA RAHMAN	**1RG18CS017**

Under the Guidance of

Mrs. Geetha Pawar

Asst. Prof. Dept. of CSE
RGIT, Bangalore – 36

Department of Computer Science & Engineering

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

Cholanagar, R.T. Nagar Post, Bangalore – 560 036

2020-2021

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(Affliliated to Visveshwaraya Technological University)

Cholanagar, R.T. Nagar Post, Bangalore – 560 032

Department of Computer Science Engineering

This is to certify the Mini Project entitled **"DIURNAL CYCLE"** is a bonafide work carried out by **BILQUEES AYESHA SIDDIQUA (1RG18CS009)** and **FAIQUA RAHMAN (1RG18CS017)** in partial fulfillment for the award of **Bachelor of Engineering in computer Science Engineering,** During the year **2020-2021**. It is certified that all corrections/suggestions given for the internal assessment have been incorporated in the report. This Mini Project has been approved as it satisfies the academic requirements in respect of the mini-project work.

Signature of Guide	Signature of HOD
Mrs. Geetha Pawar	**Mrs. Arudra A**
Asst. Professor,	Associate Prof. & HOD
Dept. of CSE	Dept. of CSE
RGIT, Bangalore	RGIT, Bangalore

EXTERNAL VIVA

Name of the Examiners	**Signature with Date**
1.	
2.	

VISVESVARAYA TECHNOLOGY UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DECLARATION

We hereby declare that the Mini Project work entitled **"DIURNAL CYCLE"** submitted to the **Visvesvaraya Technological University, Belagavi** during the academic year **2020-2021,** is a record of an original work done by us under the guidance of **Mrs. Geetha Pawar, Assistant Professor, Department of Compute Science and Engineering, Rajiv Gandhi Institute of Technology, Bangalore** and this project work is submitted in the partial fulfillment of requirements for the award of the degree of **Bachelor of Engineering in Computer Science & Engineering.** The results embodied in this thesis have not been submitted to any other University or Institute for award if any degree or diploma.

BILQUEES AYESHA SIDDIQUA 1RG18CS009

FAIQUA RAHMAN 1RG18CS017

ACKNOWLEDGEMENT

We take this opportunity to express our sincere gratitude and respect to the **Rajiv Gandhi Institute of Technology, Bangalore** for providing us an opportunity to carry out our project work.

We express our sincere regards and thanks to **Dr. NAGARAJ A M, Principal, RGIT, Bangalore** and **Mrs. Arudra A, Associate Prof. & HOD of Department of Computer Science & Engineering, RGIT, Bangalore,** for their encouragement and support throughout the project.

With profound sense of gratitude, we acknowledge the guidance and support extended by **Mrs. Geetha Pawar, Asst. Prof** and **Miss. Rajini Kodagali, Asst. Prof, Department of Computer Science & Engineering, RGIT, Bangalore.** Her incessant encouragement and valuable technical support have been of immense help in realizing this project. Herr guidance gave us the environment to enhance our knowledge, skills and to reach the pinnacle with sheer determination, dedication and hard work.

We also extend our thanks to the entire faculty of the Department of CSE, RGIT, Bangalore, who have encouraged us throughout the course of Bachelor Degree

BILQUEES AYESHA SIDDIQUA 1RG18CS009

FAIQUA RAHMAN 1RG18CS017

ABSTRACT

This project is about the creation of moving primitive 2D & 3D Objects. We are implementing it using different OpenGL libraries combining them together in a required manner to achieve the objective. Our objective is to create a serene, good-looking and colorful visualization of a place. By this we will understand the depictions of realistic things using basic functions of computer graphics

Table of Contents

Certificate
Declaration
Acknowledgement
Abstract

CHAPTERS	TITLE	PAGE NO
1.	**INTRODUCTION**	
	1.1 Computer Graphics	**1**
	1.2 Application of computer graphics	**1**
	1.2.1 Display of information	**1**
	1.2.2 Design	**2**
	1.2.3 Simulation & Animation	**2**
	1.2.4 User Interface	**2**
	1.3 Introduction to OpenGL	**3**
	1.4 Introduction to visual studio2019	**3**
	1.5 Block Diagram of OpenGL	**4**
2.	**SYSTEM ANALYSIS**	
	2.1 Existing System	**5**
	2.2 Proposed System	**5**
	2.3 Scope & Aim of Project	**5**
3.	**REQUIREMENT SPECIFICATION**	
	3.1 User Requirement	**6**
	3.2 Software Requirement	**6**
	3.3 Hardware Requirement	**6**
4.	**IMPLEMENTATION**	
	4.1 Built-in Function	**7**
	4.2 Header Files/imports	**9**
	4.3 Variable Declaration	**9**
	4.4 Function Declaration	**9**
5.	**SCREENSHOTS**	**27**
6.	**CONCLUSION**	
7.	**BIBLIOGRAPHY**	

CHAPTER 1

INTRODUCTION

1.1 Computer Graphics

Computer graphics is concerned with all aspects of producing pictures or images using a computer. The field began humbly 50 years ago, with the display of few lines on a cathode-ray tube (CRT). Now, we generate images with the computer that indistinguishable from the photographs of real objects. We routinely train pilots with simulated airplanes, generating graphical displays of a virtual environment in real time. Features length movies made entirely by computers have been successful, both critically and financially.

1.2 Applications of Computer Graphics

The development of Computer Graphics had been driven both by needs of the user community and by advances in hardware and software. The application of computer graphics are many and varied, we can however divide them into of four major areas:

1. Display of information
2. Design
3. Simulation
4. User interface

Although many application span two or more of these areas, the development of the field was based on separate work in each.

1.2.1 Display of information

Medical Imaging possesses interesting and important data analysis problem. Modern imaging technologies such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Ultrasound and Position Emission Tomography (PET), generate 3D data that must be subjected to algorithmic manipulation provide useful information. The field of scientific visualization provides graphical tools that help the researchers interpret the fast quantity of data that generate.

1.2.2 Design

Professions such as engineering and architecture are concerned with design. Starting with a set of specifications, engineers and architects seek a cast effective and aesthetic solution that satisfies the specifications.

Design is an interactive process. Design problems are either over determined such that they possess no solution that satisfies all criteria, much less an optimal solution, or undetermined, such that they have multiple solutions that satisfies the design criteria.

1.2.3 Simulation and Animation

Graphics system evolved to be capable of generating sophisticated images in real time, engineers and researchers began to use them as simulators. One of the most important use had been training of pilots. The use of special PLSI chips has led to a generation of arcade games as sophisticated as flight simulators.

The simulators can be used for designing the robot, planning its path, and simulating its behavior in complex environment. The success of flight simulators led to the use of Computer Graphics for animation in TV, motion pictures and advertising industries. Entire animated movies can now be made by computer at a cost less than that of movies with traditional ways.

1.2.4 User Interface

Our interaction with computers has become dominated by visual paradigm that includes icons, menus and pointing devices such as mouse. From user's perspective, winding system such as X and window system, Microsoft windows.

1.3 Introduction to OpenGL

Most of the applications will be designed to access OpenGL directly through functions in three libraries. Function in the main GL (or OpenGL in windows) library have names that begin with the letters gl and are stored in a library usually referred to as GL. The second is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects and simplifying programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations; functions in the GLU library begins with letters glu.

To interface with the window system and to get input from external devices into programs, need at least one more system-specific library that provides the "glue" between the window system and OpenGL. For the X window system, this library is functionality that should be expected in any modern windowing system. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus be recompiled with the GLUT library for other window systems. We use GLU libraries.

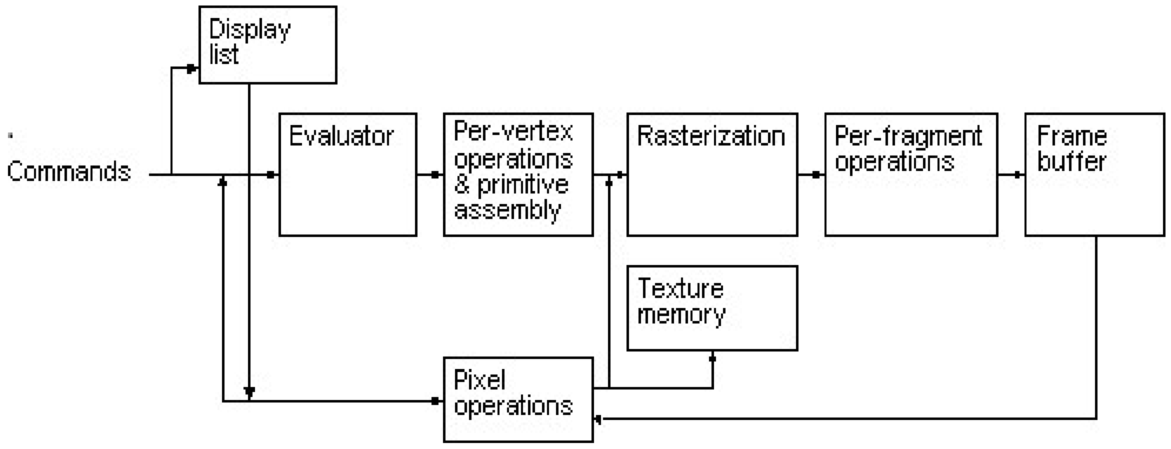
1.4 Introduction to Visual Studio 2019

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring.

Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML, and CSS. Support for other languages such as Python, Ruby, Node.js, and among others is available via plug-ins. Java (and J#) were supported in the past.

1.5 Block Diagram of OpenGL



CHAPTER 2

SYSTEM ANALYSIS

2.1 Existing System

The graphics project is designed using the graphics Utility library, which contains graphics manipulation and creation APIs, which are implemented as part of project. The object created during the course of development of this software consists of many artifacts like home, sun, clouds, trees, and birds which are seen on day-today life. System analysis and design is a very important process in any software development process. This process is called as requirement analysis. Once all the requirements are collected, we can then proceed with the actualdesigning of the system. During design process, we are involved in actual implementation of the system.

2.2 Proposed System

The proposed system is developed by using C language with openGL libraries. It is implemented on Windows. The 2D graphics packages designed here provides an interface for the user handling the display. The proposed system provides the visualization of a place which depicts realistic things using basic computer graphics functions. On Command with Keyboard clicks we can change the modes to day or night, also we can see birds flying and can turn the lights ON or OFF.

2.3 Aim

The main aim of the project is to create a serene, Good-looking and colorful visualization of a place. Various artifacts like home, sun, clouds, trees, and birds which are seen on day-today life can be implemented using basic functions of computer graphics.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 User Requirements

- Easy to understand and should be simple.
- The built-in functions should be utilized to maximum extent.
- OpenGL library facilities should be used.

3.2 Software Requirements

- Any Windows OS
- Editor: Visual Studio 2019
- Language: C language with OpenGL Library.

3.3 Hardware Requirements

- Processor: Pentium or Higher. (Preferred x64 bit – Intel i3 processor 1.6GHz)
- Ram: 4GB(At least)
- Hard Disk: 500GB (At least 80GB)
- Display: Intel HD graphics (128MB)
- Keyboard, Mouse

CHAPTER 4

IMPLEMENTATION

4.1 Built-In Functions

4.1.1 glMatrixMode(mode)

mode: Specifies which matrix stack is the target for subsequent matrix operations. Three values are accepted: GL_MODELVIEW, GL_PROJECTION and GL_TEXTURE. The initial value is GL_MODELVIEW. Additionally, if the ARB_imaging extension is supported, GL_COLOR is also accepted.

4.1.2 GlColor (* args)

Arguments can be red, green, blue Specify new red, green, and blue values for the current color. Alpha Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

4.1.3 glPopMatrix() & glPushMatrix()

glPopMatrix((void)) -> void,glPopMatrix( )

glPushMatrix((void)) -> void glPushMatrix( )

push and pop the current matrix stack

4.1.4 glTranslate ()

Multiply the current matrix by a translation matrix

glTranslate () -> glTranslate (x ,y , z)

glTranslated (x , y , z)

glTranslatef (x ,y , z)

Specify the x ,y , and z coordinates of a translation vector.

4.1.5 glRotate()

multiply the current matrix by a rotation matrix

glRotate( )-> glRotate(angle , x , y , z)

glRotated(angle , x , y , z)

glRotatef (angle , x , y , z)

Angle: Specifies the angle of rotation, in degrees.

x, y, z : Specify the x , y , and z coordinates of a vector, respectively.

4.1.6 glVertex2f (GLfloat x, GLfloat y)

The glVertex function commands are used within glBegin /glEnd pairs to specify point, line, and polygon vertices. The current color, normal, and texture coordinates are associated with the vertex when glVertex is called. When only *x* and *y* are specified, *z* defaults to 0.0 and *w* defaults to 1.0

4.1.7 glVertex3f (GLfloat x, GLfloat y, GLfloat z)

The glVertex function commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices. The current color, normal, and texture coordinates are associated with the vertex when glVertex is called. When *x*, *y*, and *z* are specified, *w* defaults to 1.0. Invoking glVertex outside of a glBegin/glEnd pair results in undefined behavior**.**

4.1.8 glPointSize (GLfloat size)

glPointSize specifies the rasterized diameter of both aliased and ant aliased points. Using a point size other than 1 has different effects, depending on whether point antialiasing is enabled. To enable and disable point antialiasing, call glEnable and glDisable with argument GL_POINT_SMOOTH. Point antialiasing is initially disabled.

4.1.9 glutMouseFunc (void (*func) (int button, int state, int x, int y))

glutMouseFunc sets the mouse callback for the *current window*. When a user presses and releases mouse buttons in the window, each press and each release generates a mouse callback. The button parameter is one of GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, or GLUT_RIGHT_BUTTON.

4.1.10 glutKeyboardFunc (void (*func) (unsigned char key, int x, int y))

glutKeyboardFunc sets the keyboard callback for the *current window*. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The state of modifier keys such as Shift cannot be determined directly; their only effect will be on the returned ASCII data.
Passing NULL to glutKeyboardFunc disables the generation of keyboard callbacks.

4.1.11 glClearColor (r, g, b, a)

glClearColor function specifies the red, green, blue, and alpha values used by glClear to clear the color buffers. Values specified by glClearColor are clamped to the range [0, 1]

4.1.12 glFlush(*void*)

The glFlush function empties all the buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in a finite amount of time.

4.1.13 glBegin（GLenum mode） AND glEnd(void)

Specifies the primitive or primitives that will be created from vertices presented between glBegin and the subsequent glEnd. Ten symbolic constants are accepted: GL_POINTS, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS, GL_QUAD_STRIP, and GL_POLYGON

4.2 Header files/imports

```
#include<stdio.h>
#include<GL/glut.h>
#include <GL/gl.h>
#include <stdlib.h>
#define SPEED 30.0
```

4.3 Variable Declaration.

```
double i = 0.0, m = 0.0, n = 0.0, o = 0.0, c = 0.0, b = 0.0;
double p = 0.75, q = 0.47, r = 0.14;
double e = 0.90, f = 0.91, g = 0.98;
int count = 0;

int light = 1, day = 1, plane = 0, comet = 0, xm = 900, bird = 0;
char ch;
```

4.4 Function Used

4.4.1 Declare Function

```
void declare (char* string)
{
while (*string)
glutBitmapCharacter (GLUT_BITMAP_TIMES_ROMAN_24, *string++);
}
```

4.4.2 Draw Pixel

```
void draw_pixel (GLint cx, GLint cy)
{

        glBegin (GL_POINTS);
        glVertex2i (cx, cy);
        glEnd ();
}
```

4.4.3 Plot Pixel

```
void plotpixels (GLint h, GLint k, GLint x, GLint y)
{
 draw_pixel(x + h, y + k);
 draw_pixel (-x + h, y + k);
 draw_pixel (x + h, -y + k);
 draw_pixel (-x + h, -y + k);
 draw_pixel(y + h, x + k);
 draw_pixel (-y + h, x + k);
 draw_pixel(y + h, -x + k);
 draw_pixel (-y + h, -x + k);
}
```

4.4.4 Draw Circle

```
 void draw_circle (GLint h, GLint k, GLint r)
{
 GLint d = 1 - r, x = 0, y = r;
 while (y > x)
 {
   plotpixels(h, k, x, y);
   if (d < 0) d += 2 * x + 3;
   else
    {
     d += 2 * (x - y) + 5;
     --y;
    }
    ++x;
    }
    Plotpixels (h, k, x, y);
  }
```

4.4.5 Draw Object

```
 void draw_object ()
 {
        int l;
```

```
if (day == 1)
{
//sky
glColor3f (0.0, 0.9, 0.9);
glBegin (GL_POLYGON);
glVertex2f (0, 380);
glVertex2f (0, 700);
glVertex2f (1100, 700);
glVertex2f (1100, 380);
glEnd ();


//sun


for (l = 0; l <= 35; l++)
{
        glColor3f (1.0, 0.9, 0.0);
        draw_circle (100, 625, l);
}


//plane
if (plane == 1)
{
        glColor3f (1.0, 1.0, 1.0);
        glBegin (GL_POLYGON);
        glVertex2f (925 + n, 625 + o);
        glVertex2f (950 + n, 640 + o);
        glVertex2f (1015 + n, 640 + o);
        glVertex2f (1030 + n, 650 + o);
        glVertex2f (1050 + n, 650 + o);
        glVertex2f (1010 + n, 625 + o);
        glEnd ();

        glColor3f (0.8, 0.8, 0.8);
        glBegin (GL_LINE_LOOP);
        glVertex2f (925 + n, 625 + o);
        glVertex2f (950 + n, 640 + o);
        glVertex2f (1015 + n, 640 + o);
        glVertex2f (1030 + n, 650 + o);
        glVertex2f (1050 + n, 650 + o);
        glVertex2f (1010 + n, 625 + o);
        glEnd ();

}

//cloud1


for (l = 0; l <= 20; l++)
{
        glColor3f (1.0, 1.0, 1.0);
```

```
            draw_circle (160 + m, 625, l);

}


for (l = 0; l <= 35; l++)
{
            glColor3f (1.0, 1.0, 1.0);
            draw_circle (200 + m, 625, l);
            draw_circle (225 + m, 625, l);
}

for (l = 0; l <= 20; l++)
{
            glColor3f (1.0, 1.0, 1.0);
            draw_circle (265 + m, 625, l);
}

//cloud2


for (l = 0; l <= 20; l++)
{
            glColor3f (1.0, 1.0, 1.0);
            draw_circle (370 + m, 615, l);
}




for (l = 0; l <= 35; l++)
{

            glColor3f (1.0, 1.0, 1.0);
            draw_circle (410 + m, 615, l);
            draw_circle (435 + m, 615, l);
            draw_circle (470 + m, 615, l);
}

for (l = 0; l <= 20; l++)
{
            glColor3f (1.0, 1.0, 1.0);
            draw_circle (500 + m, 615, l);
}





//grass
glColor3f (0.6, 0.8, 0.196078);
glBegin (GL_POLYGON);
```

```
            glVertex2f (0, 160);
            glVertex2f (0, 380);
            glVertex2f (1100, 380);
            glVertex2f (1100, 160);
            glEnd ();

    }


    else
    {

            //sky
            glColor3f (0.0, 0.0, 0.0);
            glBegin (GL_POLYGON);
            glVertex2f (0, 380);
            glVertex2f (0, 700);
            glVertex2f (1100, 700);
            glVertex2f (1100, 380);
            glEnd ();

            //moon
            int l;

            for (l = 0; l <= 35; l++)
            {
                    glColor3f (1.0, 1.0, 1.0);
                    draw_circle (100, 625, l);
            }

            //star1

            glColor3f (1.0, 1.0, 1.0);
            glBegin (GL_TRIANGLES);
            glVertex2f (575, 653);
            glVertex2f (570, 645);
            glVertex2f (580, 645);
            glVertex2f (575, 642);
            glVertex2f (570, 650);
            glVertex2f (580, 650);
            glEnd ();

            //star2
            glColor3f (1.0, 1.0, 1.0);
            glBegin (GL_TRIANGLES);
            glVertex2f (975, 643);
            glVertex2f (970, 635);
            glVertex2f (980, 635);
            glVertex2f (975, 632);
            glVertex2f (970, 640);
            glVertex2f (980, 640);
            glEnd ();
```

```
//star3
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(875, 543);
glVertex2f(870, 535);
glVertex2f(880, 535);
glVertex2f(875, 532);
glVertex2f(870, 540);
glVertex2f(880, 540);
glEnd();

//star4
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(375, 598);
glVertex2f(370, 590);
glVertex2f(380, 590);
glVertex2f(375, 587);
glVertex2f(370, 595);
glVertex2f(380, 595);
glEnd();

//star5
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(750, 628);
glVertex2f(745, 620);
glVertex2f(755, 620);
glVertex2f(750, 618);
glVertex2f(745, 625);
glVertex2f(755, 625);
glEnd();

//star6
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(200, 628);
glVertex2f(195, 620);
glVertex2f(205, 620);
glVertex2f(200, 618);
glVertex2f(195, 625);
glVertex2f(205, 625);
glEnd();

//star7
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(100, 528);
glVertex2f(95, 520);
glVertex2f(105, 520);
glVertex2f(100, 518);
glVertex2f(95, 525);
glVertex2f(105, 525);
```

```
glEnd();

//star8
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(300, 468);
glVertex2f(295, 460);
glVertex2f(305, 460);
glVertex2f(300, 458);
glVertex2f(295, 465);
glVertex2f(305, 465);
glEnd();

//star9
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(500, 543);
glVertex2f(495, 535);
glVertex2f(505, 535);
glVertex2f(500, 532);
glVertex2f(495, 540);
glVertex2f(505, 540);
glEnd();


//comet
if (comet == 1)
{
        for (l = 0; l <= 7; l++)
        {
                glColor3f(1.0, 1.0, 1.0);
                draw_circle(300 + c, 675, l);
        }

        glColor3f(1.0, 1.0, 1.0);
        glBegin(GL_TRIANGLES);
        glVertex2f(200 + c, 675);
        glVertex2f(300 + c, 682);
        glVertex2f(300 + c, 668);
        glEnd();
}

//Plane
if (plane == 1)
{

        for (l = 0; l <= 1; l++)
        {
                glColor3f(1.0, 0.0, 0.0);
                draw_circle(950 + n, 625 + o, l);
                glColor3f(1.0, 1.0, 0.0);
                draw_circle(954 + n, 623 + o, l);
```

```
                    }



            }

            //grass
            glColor3f(0.0, 0.3, 0.0);
            glBegin(GL_POLYGON);
            glVertex2f(0, 160);
            glVertex2f(0, 380);
            glVertex2f(1100, 380);
            glVertex2f(1100, 160);
            glEnd();

    }

    //Ground
    glColor3f(0.0, 0.3, 0.0);
    glBegin(GL_POLYGON);
    glVertex2f(-600, 0);
    glVertex2f(-600, 185);
    glVertex2f(1100, 185);
    glVertex2f(1100, 0);
    glEnd();

    //tree
    glColor3f(0.9, 0.2, 0.0);
    glBegin(GL_POLYGON);
    glVertex2f(280, 185);
    glVertex2f(280, 255);
    glVertex2f(295, 255);
    glVertex2f(295, 185);
    glEnd();



    for (l = 0; l <= 30; l++)
    {
            glColor3f(0.0, 0.5, 0.0);
            draw_circle(270, 250, l);
            draw_circle(310, 250, l);
    }

    for (l = 0; l <= 25; l++)
    {
            glColor3f(0.0, 0.5, 0.0);
            draw_circle(280, 290, l);
            draw_circle(300, 290, l);
    }

    for (l = 0; l <= 20; l++)
    {
```

```
		glColor3f(0.0, 0.5, 0.0);
		draw_circle(290, 315, l);
	}


	//tree 1
	glColor3f(0.9, 0.2, 0.0);
	glBegin(GL_POLYGON);
	glVertex2f(100, 135);
	glVertex2f(100, 285);
	glVertex2f(140, 285);
	glVertex2f(140, 135);
	glEnd();


	for (l = 0; l <= 40; l++)
	{
		glColor3f(0.0, 0.5, 0.0);
		draw_circle(40, 280, l);
		draw_circle(90, 280, l);
		draw_circle(150, 280, l);
		draw_circle(210, 280, l);
		draw_circle(65, 340, l);
		draw_circle(115, 340, l);
		draw_circle(175, 340, l);

	}

	for (l = 0; l <= 55; l++)
	{
		glColor3f(0.0, 0.5, 0.0);
		draw_circle(115, 360, l);


	}

	//chim

	glColor3f(0.35, 0.0, 0.0);
	glBegin(GL_POLYGON);

	glVertex2f(540, 330);
	glVertex2f(540, 430);
	glVertex2f(960, 430);
	glVertex2f(960, 330);

	glEnd();

	//home

	glColor3f(p, q, r);
	glBegin(GL_POLYGON);
```

```
glVertex2f(550, 100);
glVertex2f(550, 330);
glVertex2f(950, 330);
glVertex2f(950, 100);
glVertex2f(850, 100);
glVertex2f(850, 250);
glVertex2f(650, 250);
glVertex2f(650, 100);

glEnd();

//window border

glColor3f(0.35, 0.0, 0.0);
glBegin(GL_POLYGON);

glVertex2f(595, 205);
glVertex2f(595, 285);
glVertex2f(675, 285);
glVertex2f(675, 205);

glEnd();

glBegin(GL_POLYGON);

glVertex2f(825, 205);
glVertex2f(825, 285);
glVertex2f(905, 285);
glVertex2f(905, 205);

glEnd();

glBegin(GL_POLYGON);

glVertex2f(845, 205);
glVertex2f(845, 285);
glVertex2f(850, 285);
glVertex2f(850, 205);

glEnd();



//door
glColor3f(e, f, g);
glBegin(GL_POLYGON);

glVertex2f(800, 100);
glVertex2f(800, 220);
glVertex2f(700, 220);
glVertex2f(700, 100);

glEnd();
```

```
glColor3f(0.35, 0.0, 0.0);
glBegin(GL_POLYGON);

glVertex2f(760, 120);
glVertex2f(760, 200);
glVertex2f(700, 220);
glVertex2f(700, 100);

glEnd();



//window
glColor3f(e, f, g);
glBegin(GL_POLYGON);

glVertex2f(600, 210);
glVertex2f(600, 280);
glVertex2f(670, 280);
glVertex2f(670, 210);

glEnd();

glBegin(GL_POLYGON);

glVertex2f(830, 210);
glVertex2f(830, 280);
glVertex2f(900, 280);
glVertex2f(900, 210);

glEnd();

glColor3f(0.35, 0.0, 0.0);
glBegin(GL_POLYGON);

glVertex2f(620, 210);
glVertex2f(620, 280);
glVertex2f(625, 280);
glVertex2f(625, 210);

glEnd();

glBegin(GL_POLYGON);

glVertex2f(650, 210);
glVertex2f(650, 280);
glVertex2f(655, 280);
glVertex2f(655, 210);

glEnd();
```

```
glColor3f(0.35, 0.0, 0.0);
glBegin(GL_POLYGON);

glVertex2f(850, 205);
glVertex2f(850, 285);
glVertex2f(855, 285);
glVertex2f(855, 205);

glEnd();

glBegin(GL_POLYGON);

glVertex2f(880, 205);
glVertex2f(880, 285);
glVertex2f(885, 285);
glVertex2f(885, 205);

glEnd();


if (bird == 1)
{
        /*glColor3f(0.0,0.0,0.0);
        glBegin(GL_POLYGON);

                glVertex2f(300+i-xm,250+b);
                glVertex2f(330+i-xm,250+b);
                glVertex2f(330+i-xm,280+b);


        glEnd();*/

        glColor3f(0.73, 0.16, 0.96);
        glBegin(GL_POLYGON);

        glVertex2f(300 + i - xm, 265 + b);
        glVertex2f(330 + i - xm, 265 + b);
        glVertex2f(330 + i - xm, 250 + b);


        glEnd();

        glBegin(GL_POLYGON);

        glVertex2f(330 + i - xm, 275 + b);
        glVertex2f(340 + i - xm, 275 + b);
        glVertex2f(330 + i - xm, 265 + b);


        glEnd();

        //
```

```
                glBegin(GL_POLYGON);

                glVertex2f(200 + i - xm, 285 + b);
                glVertex2f(230 + i - xm, 285 + b);
                glVertex2f(230 + i - xm, 270 + b);

                glEnd();

                glBegin(GL_POLYGON);

                glVertex2f(230 + i - xm, 295 + b);
                glVertex2f(240 + i - xm, 295 + b);
                glVertex2f(230 + i - xm, 285 + b);


                glEnd();


                //

                glBegin(GL_POLYGON);

                glVertex2f(150 + i - xm, 285 + b);
                glVertex2f(180 + i - xm, 285 + b);
                glVertex2f(180 + i - xm, 270 + b);

                glEnd();

                glBegin(GL_POLYGON);

                glVertex2f(180 + i - xm, 295 + b);
                glVertex2f(190 + i - xm, 295 + b);
                glVertex2f(180 + i - xm, 285 + b);


                glEnd();


        }


        glFlush();
}
```

4.4.6 IDLE function

```
    void idle()
    {
         if (light == 0 && (i >= 0 && i <= 1150))
         {

         i += SPEED / 10;
         m += SPEED / 150;
         n -= 2;
         o += 0.2;
         c += 2;

 }

 if (light == 0 && (i >= 2600 && i <= 3000))
 {

         i += SPEED / 10;
         m += SPEED / 150;
         n -= 2;
         o += 0.2;
         c += 2;

 }

 if (light == 0)
 {
         i = i;
         m += SPEED / 150;
         n -= 2;
         o += 0.2;
         c += 2;

 }
 if (count <= 3)
 {

         glClearColor(1.0, 1.0, 1.0, 1.0);

         i += SPEED / 10;
         b += SPEED / 10;
         m += SPEED / 150;
         n -= 2;
         o += 0.2;
         c += 2;
 }
 if (i > 1900)
         i = 800.0;
 if (m > 1100)
         m = 0.0;
 if (o > 75)
 {
         plane = 0;
 }
 if (c > 500)
 {
```

```
                comet = 0;
        }
        if (b > 500)
        {
                b = 0.0;
                i = 800.0;
                count = count + 1;

        }

        glutPostRedisplay();

}
```

4.4.7 Mouse

```
void mouse(int btn, int state, int x, int y)
{
        if (btn == GLUT_LEFT_BUTTON && state == GLUT_UP)
          exit(0);
}
```

4.4.8 Keyboard Function

```
void keyboardFunc(unsigned char key, int x, int y)
{
        switch (key)
        {

        case 'd':
        case 'D':
                day = 1;
                p = 0.75;
                q = 0.47;
                r = 0.14;
                break;

        case 'n':
        case 'N':
                day = 0;
                p = 0.52;
                q = 0.37;
                r = 0.26;
                break;

        case 'b':
        case 'B':
                bird = 1;
                i = 800;
                b = 0.0;
                count = 0;
                break;


          case 'l':
          case 'L':
                e = 0.90;
                f = 0.91;
                g = 0.98;
                break;

          case 'f':
          case 'F':
                e = 0.0;
                f = 0.0;
                g = 0.0;
                break;

          };

          }
```

4.4.9 Main Menu

```
void main_menu(int index)
{
switch (index)
{
case 1:
      if (index == 1)
      {
            plane = 3;
            o = n = 0.0;
      }
      break;

case 2:
      if (index == 2)
      {
            comet = 3;
            c = 0.0;
      }
      break;
}
}
```

4.4.10 My Init Function

```
void myinit ()
{
        glClearColor (1.0, 1.0, 1.0, 1.0);
        glColor3f (0.0, 0.0, 1.0);
        glPointSize (2.0);
        glMatrixMode (GL_PROJECTION);
        glLoadIdentity ();
        gluOrtho2D (0.0, 1100.0, 0.0, 700.0);
}
```

4.4.11 Display Function

```
void display ()
{

  glClear (GL_COLOR_BUFFER_BIT);
  draw_object ();
  glFlush ();
}
```

4.4.12 Main Function

```
 int main(int  argc, char** argv)
{
     int c_menu;
     printf ("Project by AYESHA AND FAIQUA\n");
     printf ("--------------------------------------------------------------------------------");
     printf ("                    DIURNAL CYCLE                              ");
     printf ("--------------------------------------------------------------------------------\n\n");
     printf ("Press 'd' or 'D' to make it day. \n\n");
     printf ("Press 'n' or 'N' to make it night. \n\n");
     printf ("Press 'b' or 'B' to fly Birds. \n\n");
     printf ("Press 'l' or 'L' to turn On the lights. \n\n");
     printf ("Press 'f' or 'F' to turn Off the lights. \n\n");
     printf ("Press RIGHT MOUSE BUTTON to display menu. \n\n");
     printf ("Press LEFT MOUSE BUTTON to quit the program. \n\n\n");
     printf ("Press any key and Hit ENTER.\n");
     scanf_s ("%s", &ch,10);

     glutInit (&argc, argv);
     glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
     glutInitWindowSize (1100.0, 700.0);
     glutInitWindowPosition (0, 0);
```

```
        glutCreateWindow ("Simple Village");
        glutDisplayFunc(display);
        glutIdleFunc(idle);
        glutKeyboardFunc(keyboardFunc);
        glutMouseFunc(mouse);
        myinit ();
        c_menu = glutCreateMenu(main_menu);
        glutAddMenuEntry ("Aeroplane", 1);
        glutAddMenuEntry ("Comet", 2);
        glutAttachMenu (GLUT_RIGHT_BUTTON);
        glutMainLoop ();
        return 0;
    }
```

CHAPTER 5

SCREENSHOTS

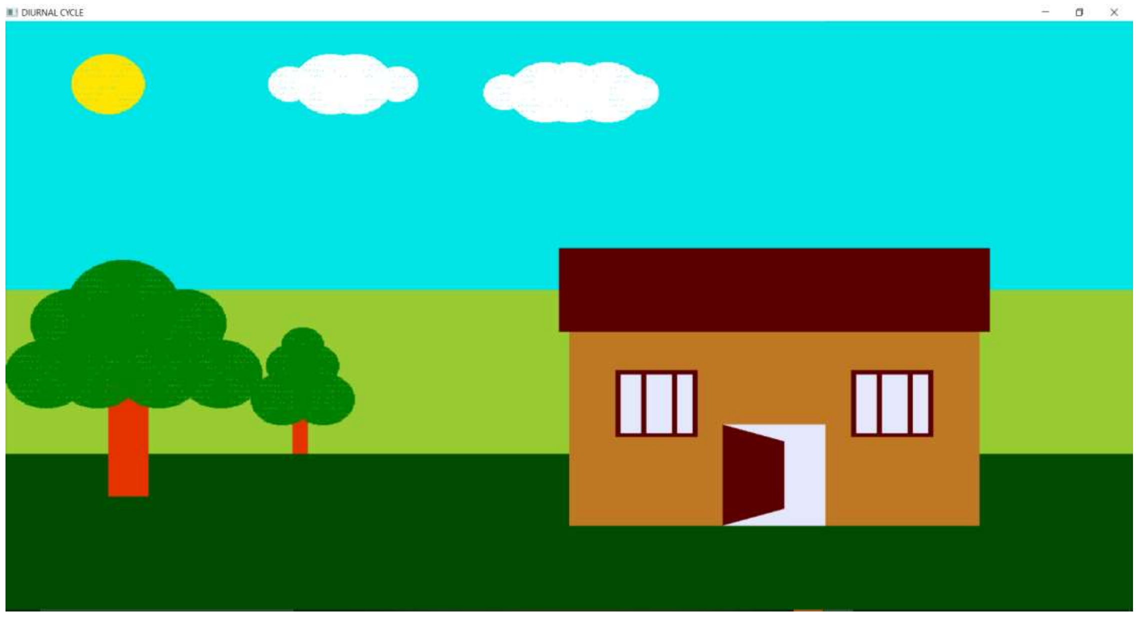


Fig1: Day Mode (Press'd' or 'D' to make it day)

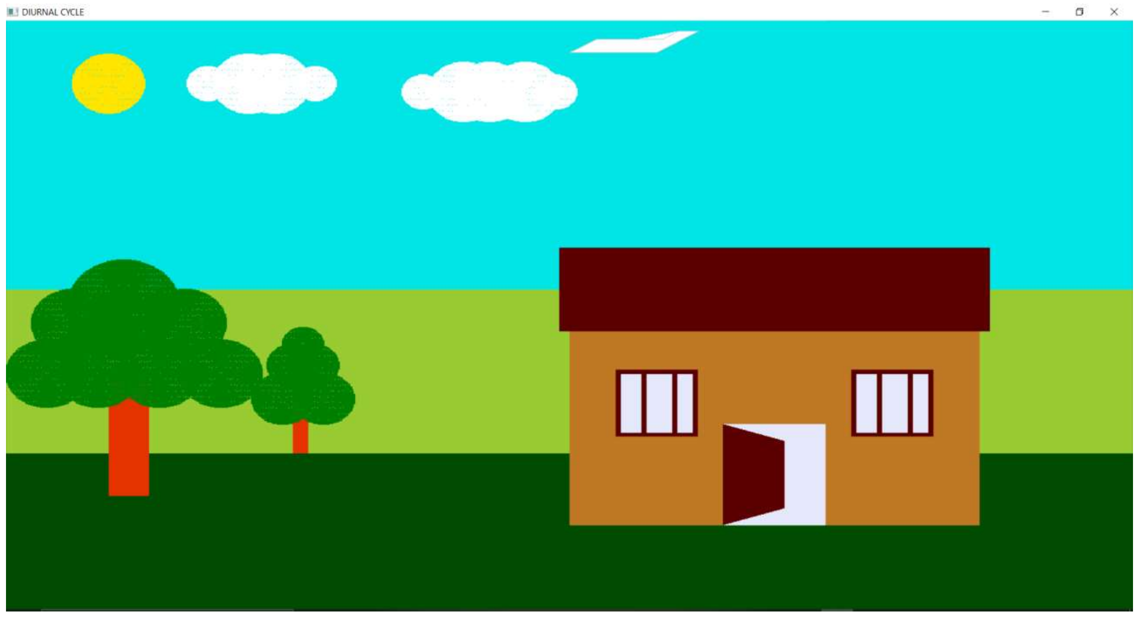


Fig 2: View of a plane in the sky

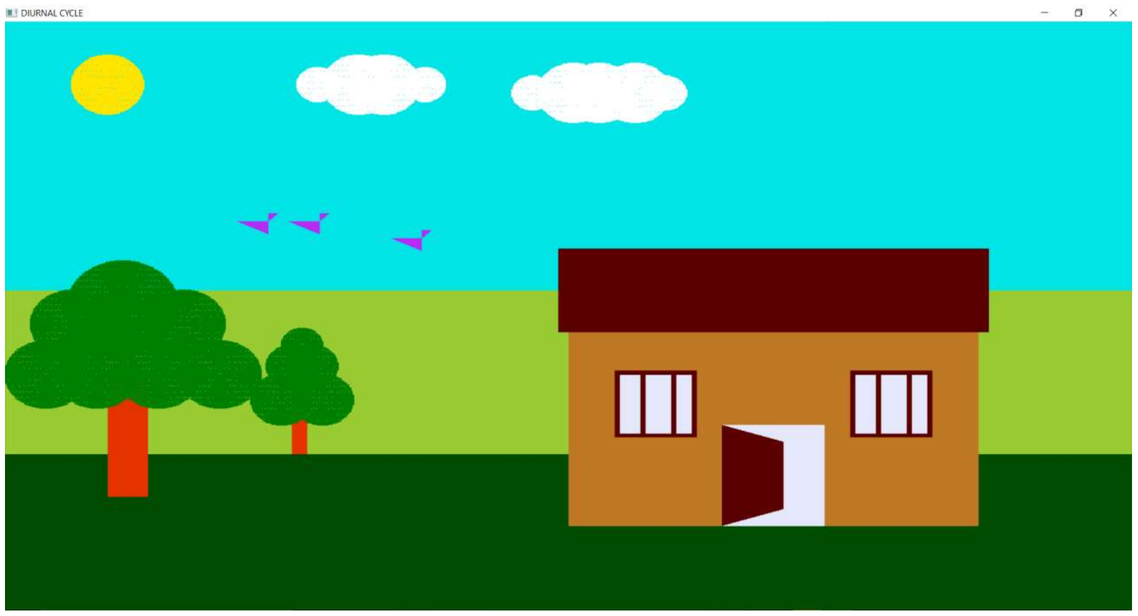


Fig 3: Flock of Birds Flying (Press 'b' or 'B' to fly Birds)

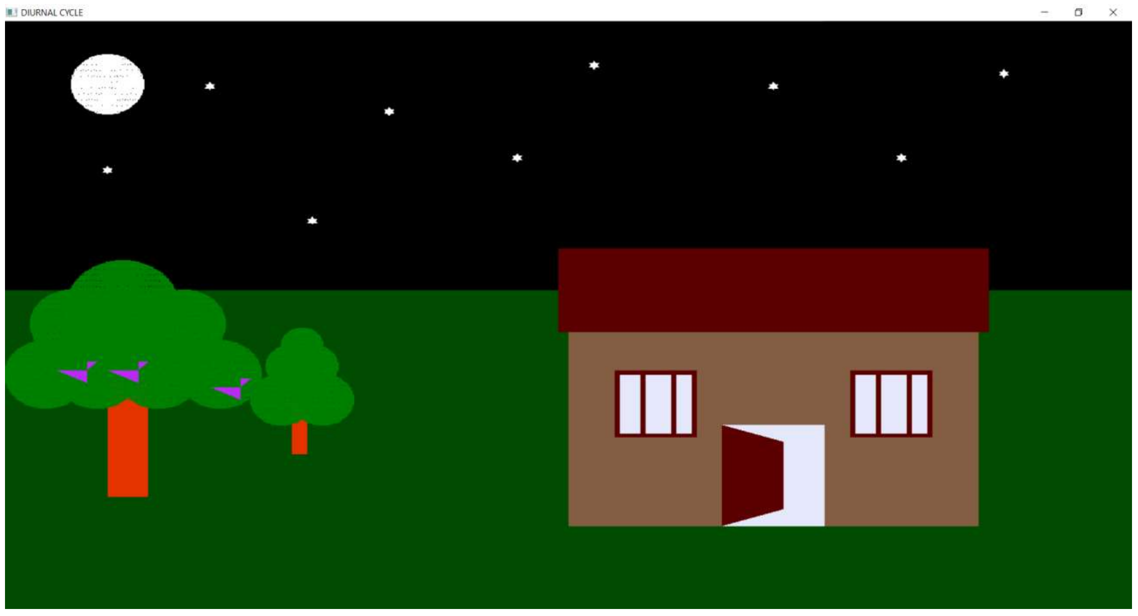


Fig 4: Night Mode (Press 'n' or 'N' to make it night)

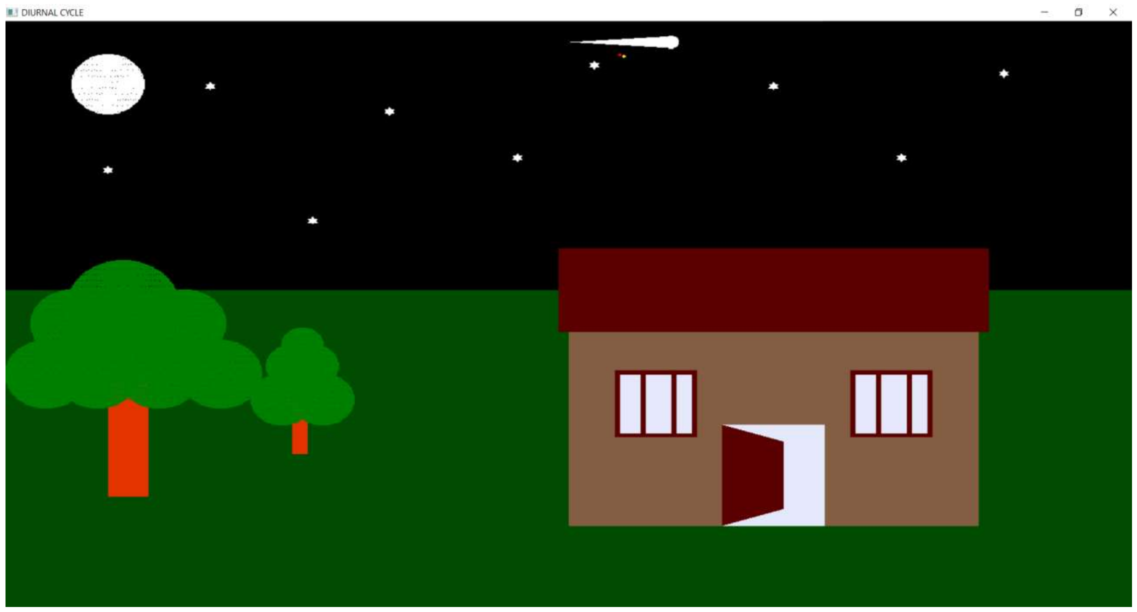


Fig 5: View of a comet passing in the night sky

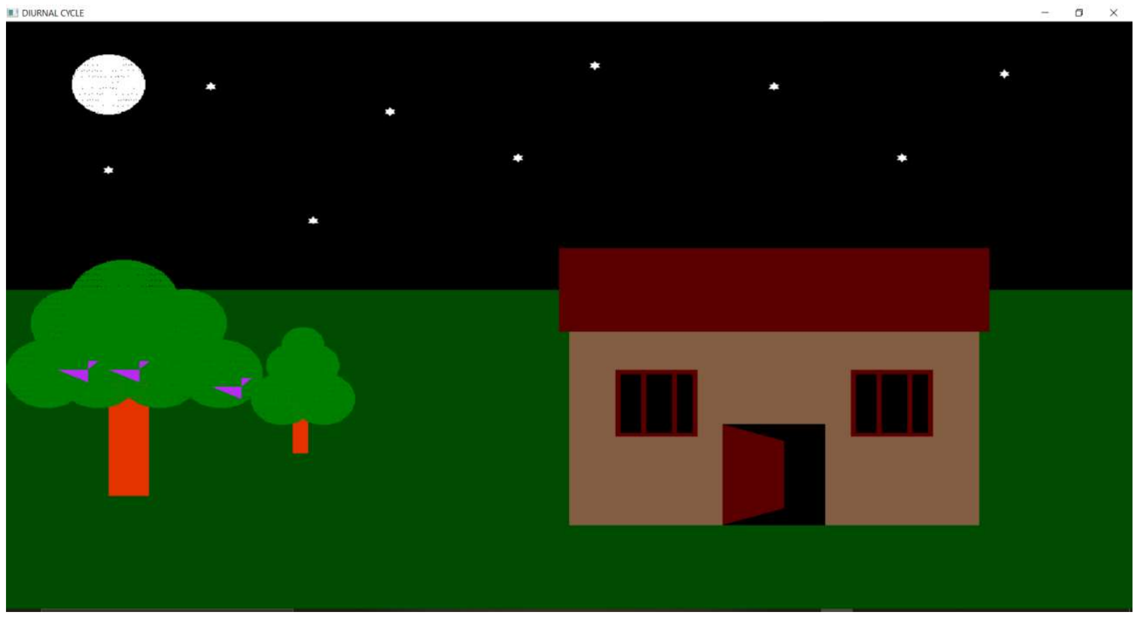


Fig 6: Lights OFF (Press 'f' or 'F' to turn off the lights)

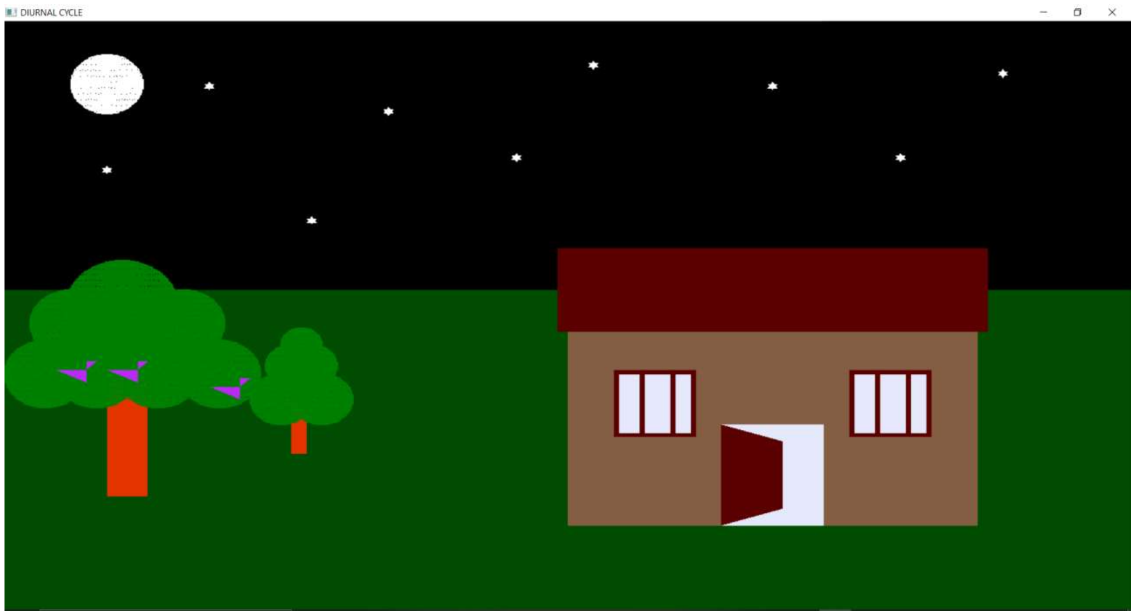


Fig 7: Lights ON (Press 'l' or 'L' to turn On the lights)

CHAPTER 6

CONCLUSION

This Project has helped us to gain knowledge about various OpenGL functions and how they can be implemented to create a full –fledged graphics project.

UsingC language &OpenGL graphics library we were able to demonstrate the Diurnal Scene.

In the end ,we can conclude saying that through this project we had an opportunity to learn computer graphics subject in detail.

During the development of this project we also learnt the various applications where this subject has been widely implemented.

Lastly, we conclude saying it was a thorough knowledge gaining experience.

FUTURE ENHANCEMENT:

The project consists of a variety of day- to- day objects being implemented in a simple manner. Further enhancement can also be done by including others objects like(a building, a road ,some vehicular movement happening in and around the place) ,etc.

BIBLIOGRAPHY

Textbook Referred:

- Interactive Computer Graphics, A Top-Down Approach with OpenGL – Edward Angel, 5th Edition, Addison- Wesley- 2008
- The offical Guide to Learning OpenGL, by Jackie Nedier, Tom Davis, Mason Woo (THE RED BOOK)
- Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd or 4th Edition, Pearson Education, 2011
- Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5th edition. Pearson Education, 2008
- James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer graphics with OpenGL: pearson education

Website Referred:

- https://github.com
- https://learnopengl.com
- https://www.udemy.com/topic/opengl/
- www.stackoverflow.com
- www.youtube.com