VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI

Mini Project Report on

"SAILING NAVY WARSHIP"

Submitted in the partial fulfillment for the requirements of Computer Graphics & Visualization Laboratory of 6th semester CSE requirement in the form of the Mini Project work

Submitted By

SRIPRIYA R — **USN: 1BY19CS158**

GAJEELEE ROHAN PRANAV — **USN: 1BY19CS194**

Under the guidance of

Prof. Muneshwara M S
Assistant Professor
Dept. of CSE

Prof. Chethana C
Assistant Professor
Dept. of CSE

Prof. Shankar R
Assistant Professor
Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

YELAHANKA, BENGALURU - 560064.

2021-2022

BMS INSTITUTE OF TECHNOLOGY &MANAGEMENT

YELAHANKA, BENGALURU – 560064

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project work entitled **"SAILING NAVY WARSHIP"** is a bonafide work carried out by **SRIPRIYA R (1BY19CS158) and GAJEELEE ROHAN PRANAV (1BY19CS194)** in partial fulfillment for *Mini Project* during the year 2021-2022. It is hereby certified that this project covers the concepts of *Computer Graphics & Visualization*. It is also certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in this report.

Signature of the Guide
Prof. Muneshwara M S/
Prof. Chethana C /
Prof. Shankar R
Assistant Professor
CSE, BMSIT&M

Signature of the HOD
Dr. Thippeswamy G
Professor & HOD
CSE, BMSIT&M

Name and Signature of the Examiners

Internal Examiner

External Examiner

INSTITUTE VISION

To emerge as one of the finest technical institutions of higher learning, to develop engineering professionals who are technically competent, ethical and environment friendly for betterment of the society.

INSTITUTE MISSION

Accomplish stimulating learning environment through high quality academic instruction, innovation and industry-institute interface.

DEPARTMENT VISION

To develop technical professionals acquainted with recent trends and technologies of computer science to serve as valuable resource for the nation/society.

DEPARTMENT MISSION

Facilitating and exposing the students to various learning opportunities through dedicated academic teaching, guidance and monitoring.

PROGRAM EDUCATIONAL OBJECTIVES

1. Lead a successful career by designing, analyzing and solving various problems in the field of Computer Science & Engineering.
2. Pursue higher studies for enduring edification.
3. Exhibit professional and team building attitude along with effective communication.
4. Identify and provide solutions for sustainable environmental development.

PROGRAM SPECIFIC OUTCOMES

1. Analyze the problem and identify computing requirements appropriate to its solution.
2. Apply design and development principles in the construction of software systems of varying complexity.

ACKNOWLEDGEMENT

We are happy to present this project after completing it successfully. This project would not have been possible without the guidance, assistance and suggestions of many individuals.

We would like to express our deep sense of gratitude and indebtedness to each and every one who has helped us make this project a success.

We heartily thank our Principal, **Dr. MOHAN BABU G N,** BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We heartily thank our Professor and Head of the Department, **Dr. Thippeswamy G**, Department of Computer Science and Engineering, BMS Institute of Technology and Management, for his constant encouragement and inspiration in taking up this project.

We gracefully thank our Project Guide, **Prof. Muneshwara M S, Prof. Chethana C** and **Prof. Shankar R** Assistant Professors, Department of Computer Science and Engineering for their intangible support and for being constant backbone for our project.

Special thanks to all the staff members of Computer Science Department for their help and kind co-operation.

Lastly, we thank our parents and friends for the support and encouragement given throughout in completing this precious work successfully.

SRIPRIYA R (1BY19CS158)

GAJEELEE ROHAN PRANAV (1BY19CS194)

ABSTRACT

The computer graphics (CG) mini project titled “Sailing Navy Warship” is project that is using a set of OpenGL functions to demonstrate how to build 2D model of objects and their movement and to demonstrate the day/night cycle of an environment along with a shooting mechanism for the ship.

The project consists of 2 scenes.

- The first scene shows the project information such as project title, name of student, USN and so on.
- The second scene will be of the ship and its background sailing on the ocean with three planes flying across the screen.

User interaction is provided by enabling the user to move the ship in forward and backward direction using the alphabetical keys. User can stop the ship and even choose the day and night cycle.

During the day cycle the sun will be visible starting from the left position. As we slowly change to the night cycle the sun will move to the right until it disappears and the moon will appear in its place. All the 2D models in the program such as the ship and the mountains also changes color based on the day/night cycle

Additionally a shooting mechanism is implemented for the ship with the user being able to fire one shot at a time using the alphabetical keys.

The planes continuously travel across the screen automatically.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Brief introduction to the project including the idea behind it is explained in this chapter.

1.1 Brief Introduction

COMPUTER GRAPHICS:

Computer graphics are graphics created using computers and, more generally, the representation and manipulation of image data by a computer hardware and software. The development of computer graphics, or simply referred to as CG, has made computers easier to interact with, and better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized the animation and video game industry. 2D computer graphics are digital images—mostly from two-dimensional models, such as 2D geometric models, text (vector array), and 2D data. 3D computer graphics in contrast to 2D computer graphics are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering images.

The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touchscreen. Due to close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics. The advantages of the interactive graphics are many in number. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D patter-recognition abilities allow us to perceive and process data rapidly and efficiently. In many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the 1980s when the scientists and engineers realized that they could not interpret the prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

OPEN GL:

OpenGL is the most extensively documented 3D graphics API (Application Program Interface) to date. Information regarding OpenGL is all over the Web and in print. It is impossible to exhaustively list all sources of OpenGL information. OpenGL programs are typically written in C and C++. One can also program OpenGL from Delphi (a Pascal-like language), Basic, Fortran, Ada, and other languages. To compile and link OpenGL programs, one will need OpenGL header files. To run OpenGL programs one may need shared or dynamically loaded OpenGL libraries, or a vendor-specific OpenGL Installable Client Driver (ICD). OpenGL is a low-level graphics library specification. It makes available to the programmer a small set of geometric primitives - points, lines, polygons, images, and bitmaps. OpenGL provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primitives, together with commands that control how these objects are rendered (drawn).

GLUT:

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres, and cylinders. GLUT even has some limited support for creating pop-up menus. The two aims of GLUT are to allow the creation of rather portable code between operating systems (GLUT is crossplatform) and to make learning OpenGL easier. All GLUT functions start with the glut prefix (for example, glutPostRedisplay marks the current window as needing to be redrawn).

1.2 Objectives

- Implementing certain technical concepts like translation, lighting, polygons drawing.
- This project implements day and night cycle option with keyboard interface.

1.3 Scope and Motivation

We found designing and developing the 2D ship along with its day and night cycle to be very interesting and a learning experience. It helped us to learn computer graphics, design of graphical user interfaces, interface to the user, user interaction handling and screen management.

“ Accomplish your tasks by one step at each time. Don't cut your coat according to your elder brother's size. Dream big, but start small; do what you can do at a time.”

— Israelmore Ayivor

1.4 Problem Statement

Computer graphics is no longer a rarity. It is an integral part of all computer user interfaces, and is indispensable for visualizing 2D.OpenGL provides more features for developing 2D objects with few lines by built in functions .So in this project the main problem statement is to design and model a ship which is capable of moving from left to right with shooting capabilities. We will be adding some additional features such as day and night cycle as well

1.5 Proposed System

To achieve our goal, open GL software is proposed. It is software which provides a graphical interface. It is an interface between application program and graphics hardware. The advantages are:

1. Open GL is designed as a streamlined.
2. It's a hardware independent interface that is it can be implemented on many different hardware platforms.
3. With OpenGL we can draw a small set of geometric primitives such as points, lines and polygons etc.
4. It provides double buffering which is vital in providing transformations.
5. It is event driven software.
6. It provides call back function.

1.6 Limitations

Our project as of now is very limited and many of the mechanism we wish to implement are still in development such as a real lighting effect for the environment, fluid motion of the water, day/night cycle automatic transformation. Also the ship is only able to move in the horizontal direction as of now

CHAPTER 2

LITERATURE SURVEY

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube (CRT) screens soon after the introduction of computers.

Computer graphics today largely interactive, the user controls the contents, structure, and appearance of objects and of displayed images by using input devices, such as keyboard, mouse, or touch-sensitive panel on the screen. Graphics based user interfaces allow millions of new users to control simple, low-cost application programs, such as spreadsheets, word processors, and drawing programs.

OpenGL (Open Graphics Library) is a standard specification defining a cross language, cross platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives.

OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation. It is also used in video games, where it competes with Direct3D on Microsoft Windows platforms (see Direct3D vs. OpenGL). OpenGL is managed by the non-profit technology consortium, the Khronos Group.

In the 1980s, developing software that could function with a wide range of graphics hardware was a real challenge. By the early 1990s, Silicon Graphics (SGI) was a leader in 3D graphics for workstations. SGI's competitors (including Sun Microsystems, Hewlett-Packard and IBM) were also able.

In addition, SGI had a large number of software customers, by changing to the OpenGL. API they planned to keep their customers locked onto SGI (and IBM) hardware for a few years while market support for OpenGL matured to bring to market 3D hardware, supported by

extensions made to the PHIGS standard. In 1992, SGI led the creation of the OpenGL architectural review board (OpenGL ARB), the group of companies that would maintain and expand the OpenGL specification took for years to come. On 17 December 1997, Microsoft and SGI initiated the Fahrenheit project, which was a joint effort with the goal of unifying the OpenGL and Direct3D interfaces (and adding a scene-graph API too). In 1998 Hewlett Packard joined the project.

It initially showed some promise of bringing order to the world of interactive 3D computer graphics APIs, but on account of financial constraints at SGI, strategic reasons at Microsoft, and general lack of industry support, it was abandoned in 1999 Many opengl functions are used for rendering and transformation purposes. Transformations functions like glRotate (), glTranslate (), glScale () can be used .

OpenGL provides a powerful but primitive set of rendering command, and all higher-level drawing must be done in terms of these commands. There are several libraries that allow you to simplify your programming tasks, including the following:

OpenGL Utility Library (GLU) contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections and rendering surfaces .

OpenGL Utility Toolkit (GLUT) is a window-system-independent toolkit, written by Mark Kill Guard, to hide the complexities of differing window APIs.

To achieve the objective of the project, information related to the light sources is required with OpenGL. we can manipulate the lighting and objects in a scene to create many different kinds of effects. It explains how to control the lighting in a scene, discusses the

OpenGL conceptual model of lighting, and describes in detail how to set the numerous illumination parameters to achieve certain effects.

To demonstrate the transformation and lightening, effects, different polygons have to be used. Polygons are typically drawn by filling in all the pixels enclosed within the boundary, but we can also draw them as outlined polygons or simply as points at the vertices.

The properties of a light source like its material, diffuse, emissive, has to be mention in the project. So to design the light source and the objects, programming guide of an OpenGL is used.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

- Processor: INTEL / AMD
- Main memory: 2 GB RAM (Min.)
- Hard Disk: Built-in is sufficient
- Keyboard: QWERTY Keyboard
- Mouse: 2 or 3 Button mouse
- Monitor: 1024 x 768 display resolution

3.2 SOFTWARE REQUIREMENTS

- Programming language – C/C++ using OpenGL
- Operating system – Windows/Linux
- Compiler – C/C++ Compiler (GCC compiler)
- IDE – Code blocks
- Functional Requirement – <GL/glut.h>

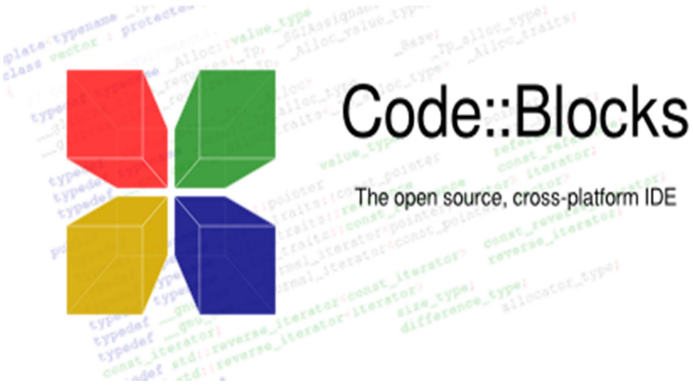

FIGURE 3.1 CODE BLOCKS

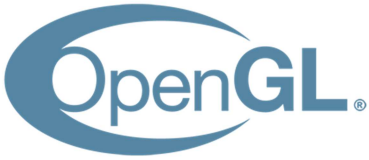

FIGURE 3.2 OpenGL API

3.3 User Interface

The legend for user interaction is as follows:

"**ESC**" to quit.
"ENTER" to go to the 2nd screen.
"f" to shoot bullet.
"a" to move backward.
"d" to move forward.
"z" Early morning cycle
"x" Morning cycle
"c" Afternoon cycle
"v" Evening cycle
"b" Night cycle
"n" Mid night cycle

CHAPTER 4

SYSTEM ANALYSIS

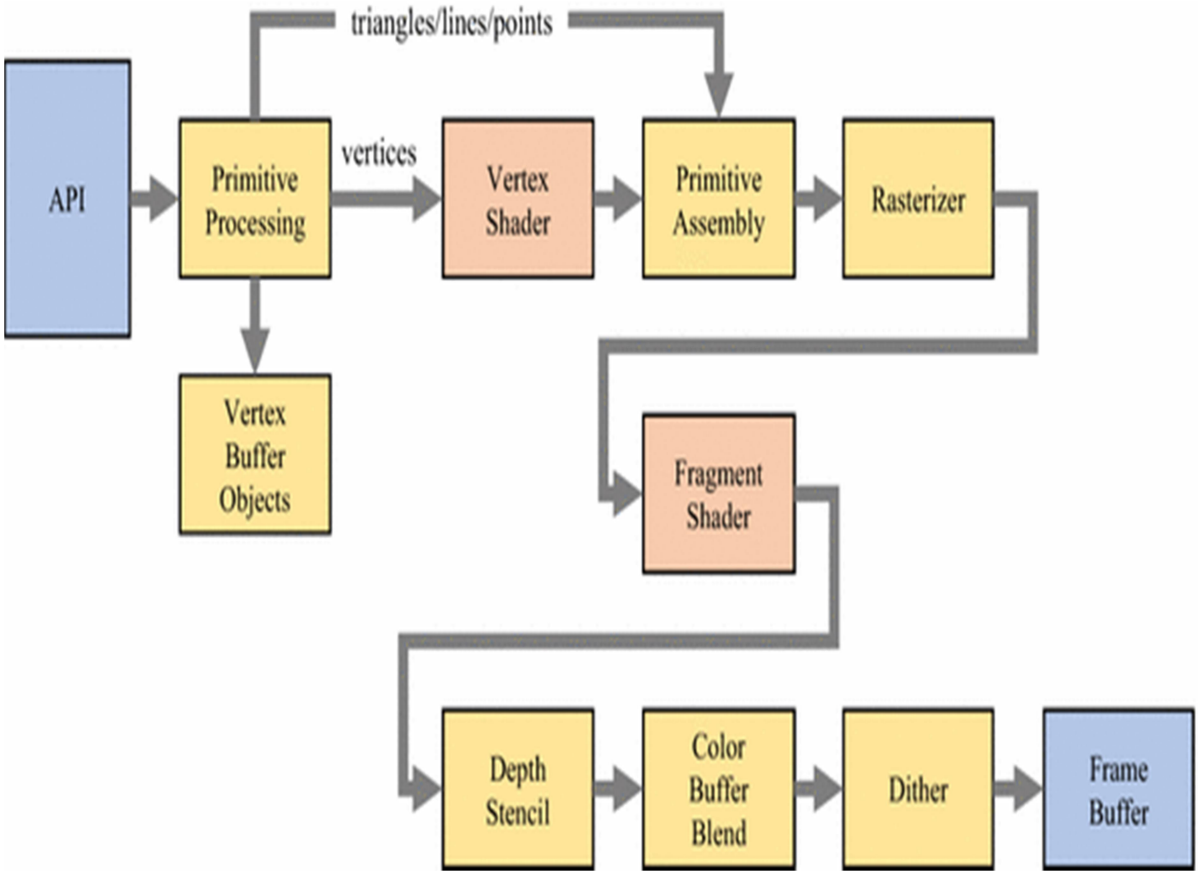

FIGURE 4.1 OPENGL PIPELINE

The OpenGL pipeline performs conditional or selection operations at a number of stages in the pipeline. One or more of these stages can be used together to implement simple conditional operations in a multipass algorithm. Examples include the depth and alpha tests.

The OpenGL rendering pipeline is initiated when you perform a rendering operation. Rendering operations require the presence of a properly-defined vertex array object and a linked Program Object or Program Pipeline Object which provides the shaders for the programmable pipeline stages.

The OpenGL rendering pipeline is initiated when you perform a rendering operation. Rendering operations require the presence of a properly-defined vertex array object and a linked Program Object or Program Pipeline Object which provides the shaders for the programmable pipeline stages.

Once initiated, the pipeline operates in the following order:

Vertex Processing: Each vertex retrieved from the vertex arrays (as defined by the VAO) is acted upon by a Vertex Shader. Each vertex in the stream is processed in turn into an output vertex.

- Optional primitive tessellation stages.
- Optional Geometry Shader primitive processing. The output is a sequence of primitives.

Vertex Post-Processing, the outputs of the last stage are adjusted or shipped to different locations.

- Transform Feedback happens here.
- Primitive Assembly
- Primitive Clipping, the perspective divide, and the viewport transform to window space.

Scan conversion and primitive parameter interpolation, which generates a number of Fragments.

A **Fragment** Shader processes each fragment. Each fragment generates a number of outputs.

Per-Sample_Processing, including but not limited to:

- Scissor Test
- Stencil Test
- Depth Test
- Blending
- Logical Operation
- Write Mask

CHAPTER 5

SYSTEM IMPLEMENTATION

OpenGL Functions:

The different OpenGL functions that are used in the project is described below:

glClearColor(0.0,0.0,0.0,1.0) :- glClearColor() 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

- glMatrixMode(GL_MODELVIEW) :- Specify which matrix is the current matrix. 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.

- gluOrtho2D(0,1024,0,768) :- Define a 2D orthographic projection matrix. Left, right- Specify the coordinates for the left and right vertical clipping planes. Bottom, top - Specify the coordinates for the bottom and top horizontal clipping planes.

- glRasterPos2i(x, y) :- The glRasterPos2 function uses the argument values for x and y while implicitly setting z and w to zero and one. The object coordinates presented by glRasterPos are treated just like those of a glVertex command. They are transformed by the current modelview and projection matrices and passed to the clipping stage.

- GlutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,string[i]) :- glutBitmapCharacter renders a bitmap character using OpenGL. Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font.

- glFlush() :- Force execution of GL commands in finite time. glFlush empties all of these 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 finite time.

- glVertex2f(0.0,30.0/3) :- Specify a vertex. glVertex commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices. The current color, normal, texture coordinates, and fog coordinate are associated with the vertex when glVertex is called.

- glEnd() :- Delimit the vertices of a primitive or a group of like primitives. glBegin and glEnd delimit the vertices that define a primitive or a group of like primitives.

- glutSwapBuffers() :- glutSwapBuffers swaps the buffers of the current window if double buffered. Performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers promotes the contents of the back buffer of the layer in use of the current window to become the contents of the front buffer. An implicit glFlush is done by glutSwapBuffers before it returns. If the layer in use is not double buffered, glutSwapBuffers has no effect.

- glPushMatrix() Push and pop the current matrix stack. There is a stack of matrices for each of the matrix modes. The current matrix in any mode is the matrix on the top of the stack for that mode.glPushMatrix pushes the current matrix stack down by one, duplicating the current matrix. That is, after a glPushMatrix call, the matrix on top of the stack is identical to the one below it.

- glPopMatrix() :- Push and pop the current matrix stack. glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack.

- glTranslatef(50,100,0) :- Multiply the current matrix by a translation matrix. glTranslate produces a translation by x y z . The current matrix is multiplied by this translation matrix, with the product replacing the current matrix.

- glutInit(&argc,argv) :- glutInit is used to initialize the GLUT library. glutInit will initialize the GLUT library and negotiate a session with the window system. During this

process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized. glutInit also processes command line options, but the specific options parse are window system dependent.

- glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB) :- Sets the initial display mode. The initial display mode is used when creating top-level windows, sub windows, and overlays to determine the OpenGL display mode for the to-be-created window or overlay.

- glutInitWindowSize(1024,768) :- Set the initial window size . Windows created by glutCreateWindow will be requested to be created with the current initial window position and size. The initial value of the initial window size GLUT state is 300 by 300. The initial window size components must be greater than zero. The intent of the initial window position and size values is to provide a suggestion to the window system for a window's initial size and position. The window system is not obligated to use this information. A GLUT program should use the window's reshape callback to determine the true size of the window.

- glutCreateWindow("Moving Ship") :- Creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name. Implicitly, the current window is set to the newly created window.

- glutPositionWindow(0,0) :- Requests a change to the position of the current window. . For top-level windows, the x and y parameters are pixel offsets from the screen origin. For sub windows, the x and y parameters are pixel offsets from the window's parent window origin.

- glutDisplayFunc(display) :-Registers the callback function (or event handler) for handling window-paint event. The OpenGL graphic system calls back this handler when it receives a window re-paint request. In the example, we register the function display() as the handler.

- glutKeyboardFunc(keyboard) :- 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 key callback parameter is the generated ASCII character. The state of modifier keys such as Shift cannot be determined directly; their only effect will be on the returned ASCII data.

- glutMainLoop() :- glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

User Defined Functions:

The whole project is divided into many small parts known as functions and these functions would take care of implementing particular parts of this project which makes the programming easy and effective. Each function takes the responsibility of designing the assigned parts hence we combined all the functions at a particular stage to get the final required design. The functions that are used to implement the “Sailing Navy Warship” are:

- void display() : This function is used to display the start of the ship scene . It calls other function like mountain3(), water() and plane(). It is also used to define and set the colour. In the end there is a glflush() to display everything in the screen.

- void frontscreen(void) :This function is used to display the first page i.e college name, project name and students name

- void Drawarc(float sa,float ea,float cx,float cy,float rd) :This function is used to draw the sun and moon and their color variant

- void cloud(int m, int n) :This function is used to draw the clouds

- void ship(float x) :This function is used to draw the ship

- void water() :This function is used to draw the water

- void mountain2(),void mountain3(),void mountain2() :These functions are used to draw all the mountains
- void flag(float x) :This function is used to draw the flag on the ship
- void crackers() :This function is used to draw the bullet fired by the ship
- void plane() :This function is used to draw the plane
- void keyboard(unsigned char key, int x, int y) :This function is used to assign the keyboard keys for user interaction.

CHAPTER 6

INTERPRETATION OF RESULTS

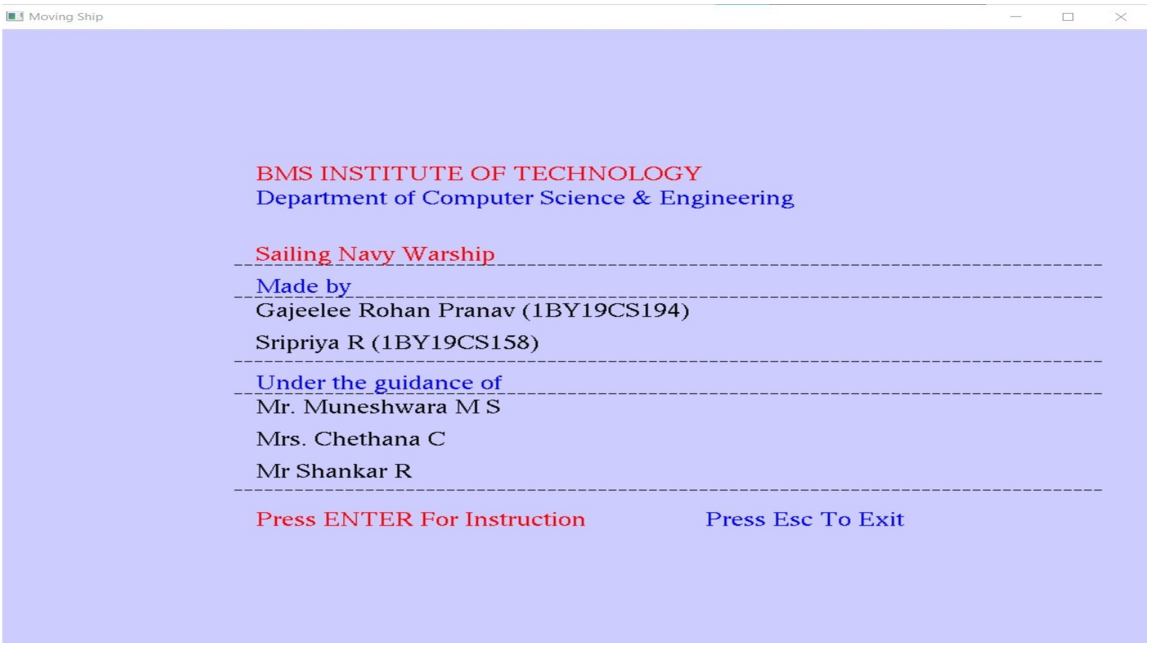

FIGURE 6.1 FIRST SCREEN

FIGURE 6.2 SECOND SCREEN

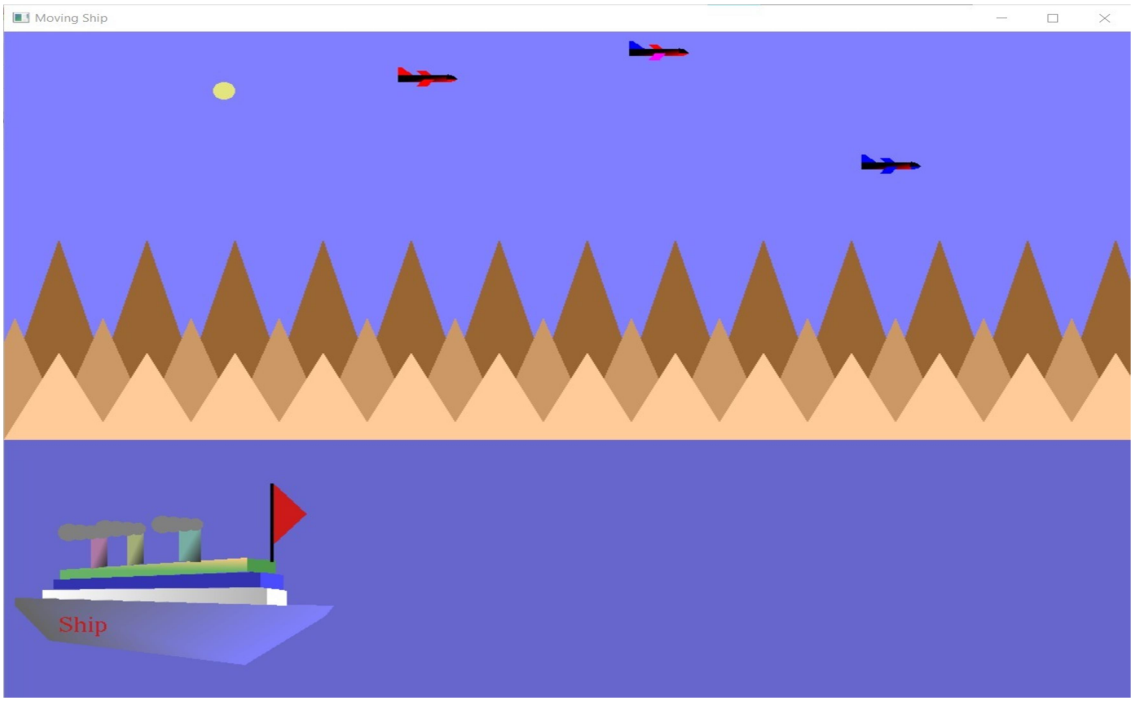

FIGURE 6.3 MORNING

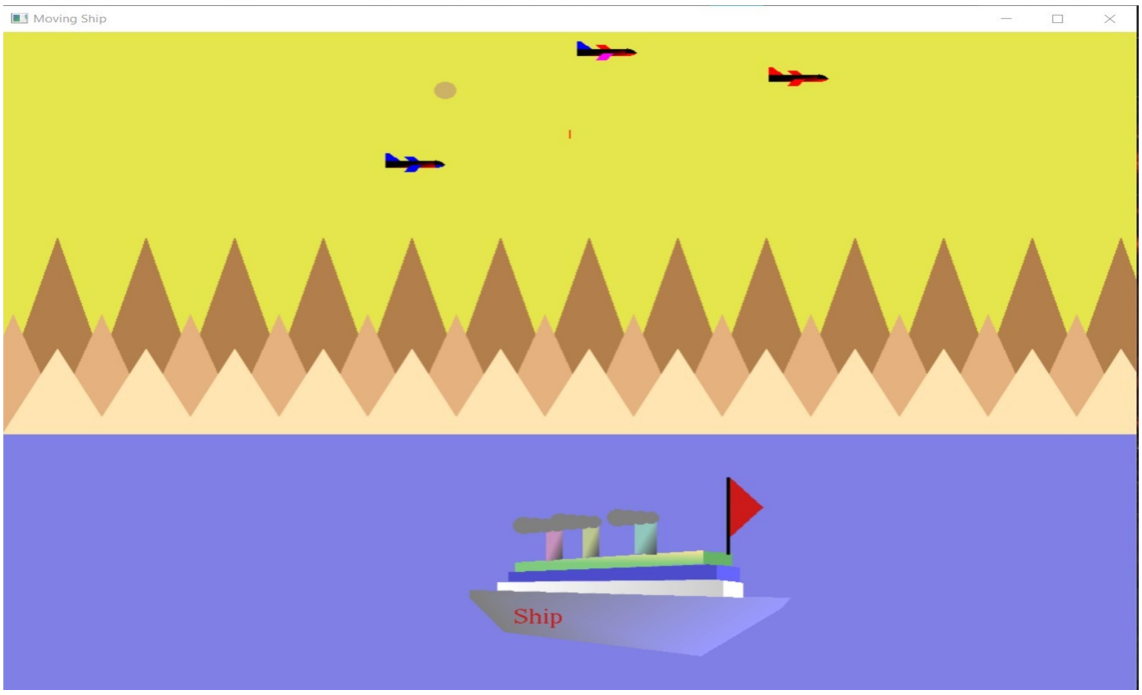

FIGURE 6.4 AFTERNOON

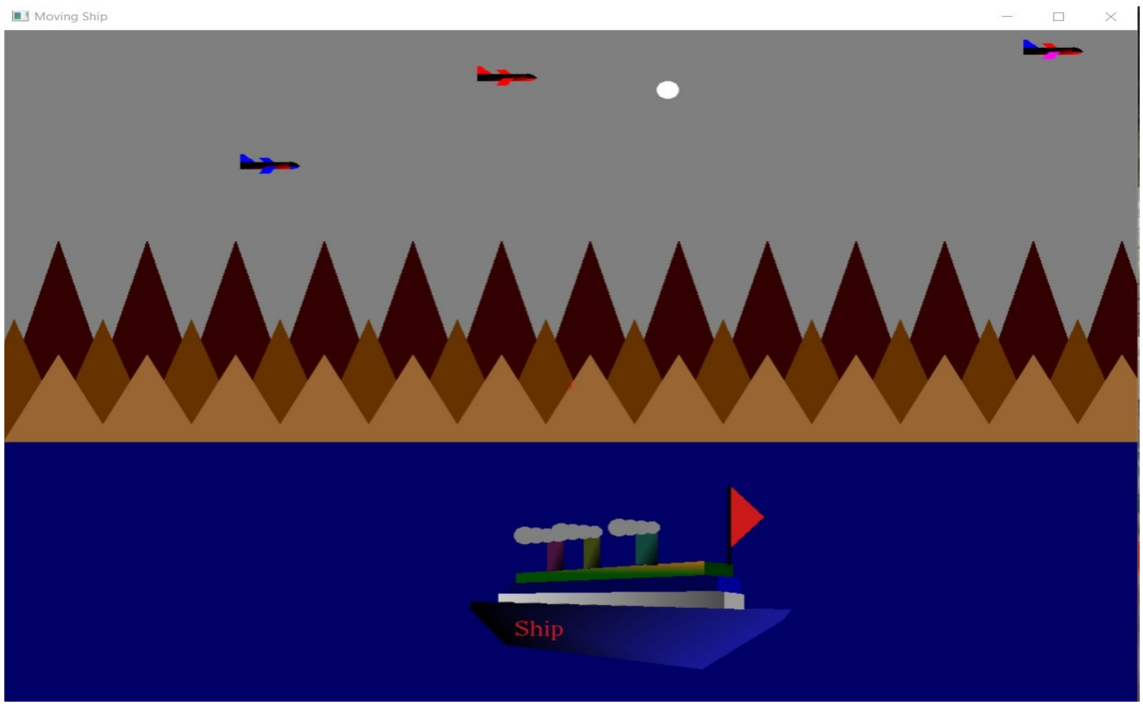

FIGURE 6.5 NIGHT

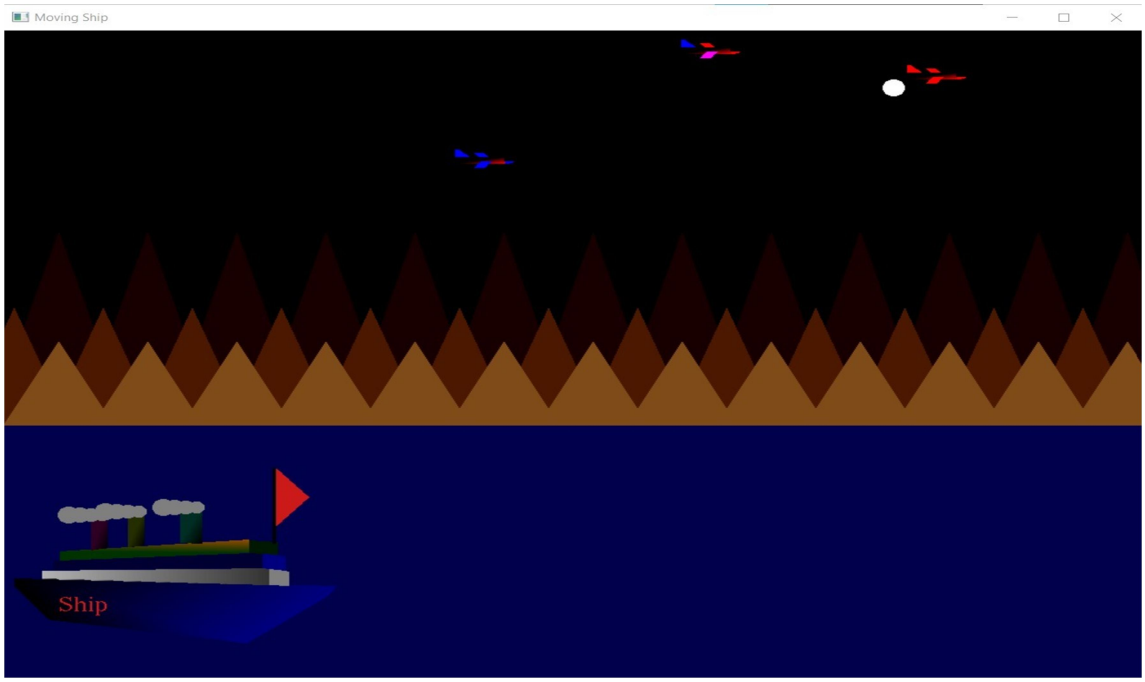

FIGURE 6.6 MID NIGHT

CHAPTER 7

CONCLUSION

We have attempted to design and implement "Sailing Navy Warship". OpenGL supports enormous flexibility in the design and the use of OpenGL graphics programs. The presence of many built in classes method take care of much functionality and reduce the job of coding as well as makes the implementation simpler. We have implemented the project making it user-friendly and error free as possible.

CHAPTER 8

FUTURE ENHANCEMENTS

These are the features that are planned to be supported in the future

- Support for movement of the sea when ship moves
- Support for 3D modeling of the whole program
- Support for automatic transformation of the day/night cycle
- Support for both horizontal and vertical movement of the ship

BIBLIOGRAPHY

1] Edward Angel's Interactive Computer Graphics Pearson Education 5th Edition.

[2] Interactive computer Graphics --A top down approach using open GL--by Edward Angle.

[3] Jackie L. Neider, Mark Warhol, Tom.R.Davis, "OpenGL Red Book", Second Revised Edition, 2005.

[4] Donald D Hearn and M.Pauline Baker, "Computer Graphics with OpenGL", 3rd Edition.