

RV COLLEGE OF ENGINEERING®
BENGALURU – 560059

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



“SINKING SHIP”

COMPUTER GRAPHICS LAB (16CS73)

OPEN ENDED EXPERIMENT REPORT

VII SEMESTER

2020-2021

Submitted by

Priya Darshini R (1RV18CS424)

Ramyashree V (1RV18CS427)

Under the Guidance of

MAMATHA T

Department of CSE, R.V.C.E.,

Bengaluru - 560059

RV COLLEGE OF ENGINEERING[®], BENGALURU - 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the **Open-Ended Experiment** titled “**SINKING SHIP**” has been carried out by **Priya Darshini R (1RV18CS424)**, **Ramyashree V (1RV18CS427)**, bonafide students of RV College of Engineering, Bengaluru, have submitted in partial fulfillment for the **Internal Assessment of Course: COMPUTER GRAPHICS LAB (16CS73)** during the year 2020-2021. It is certified that all corrections/suggestions indicated for the internal Assessment have been incorporated in the report.

MAMATHA T

Faculty Incharge,
Department of CSE,
R.V.C.E., Bengaluru –59

Dr. Ramakanth Kumar P

Head of Department,
Department of CSE,
R.V.C.E., Bengaluru–59

RV COLLEGE OF ENGINEERING® , BENGALURU - 560059
(Autonomous Institution Affiliated to VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **Priya Darshini R (1RV18CS424), Ramyashree V (1RV18CS427)** the students of Seventh Semester B.E., Computer Science and Engineering, R.V. College of Engineering, Bengaluru hereby declare that the mini-project titled **“SINKING SHIP”** has been carried out by us and submitted in partial fulfillment for the **Internal Assessment of Course: COMPUTER GRAPHICS LAB (16CS73) - Open-Ended Experiment** during the year 2020-2021. We do declare that matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Place: Bengaluru
Date: 06-01-2021

Priya Darshini R
Ramyashree V

Contents

1. Introduction

1.1 Computer Graphics

1.2 OpenGL

2. OpenGL Syntax

3. Requirement Specifications

3.1 Hardware Requirements

3.2 Software Requirements

4. User Interaction

5. Snapshots

6. Conclusion

7. Bibliography

1. INTRODUCTION

This synopsis contains implementation of ‘**SINKING SHIP**’ using a set of OpenGL functions. The project consists of three scenes. The first scene just shows the ship, which is stable and not moving. The second scene will be of the moving ship. In addition, the third scene will be of the ship that is being hit onto an iceberg and sink in the sea. User interaction is provided by enabling the user to move the ship in forward and backward direction using alphabetical keys or the special keys. User can stop the ship and even change the mode to day or night. Additionally, the user can access these functions from the menu by right clicking the mouse. The objects are drawn using GLUT functions.

1.1 Computer Graphics

Computer graphics is one of the most exciting and rapidly growing computer fields. It is also an extremely effective medium for communication between man and computer; a human being can understand the information content of a displayed diagram or perspective view much faster than he can understand a table of numbers or text containing the same information. Thus computer graphics is being used more extensively. There is a lot of development in hardware and software required to generate images, and nowadays the Cost of hardware and software is dropping rapidly. Due to this, interactive computer graphics is becoming available to more and more people.

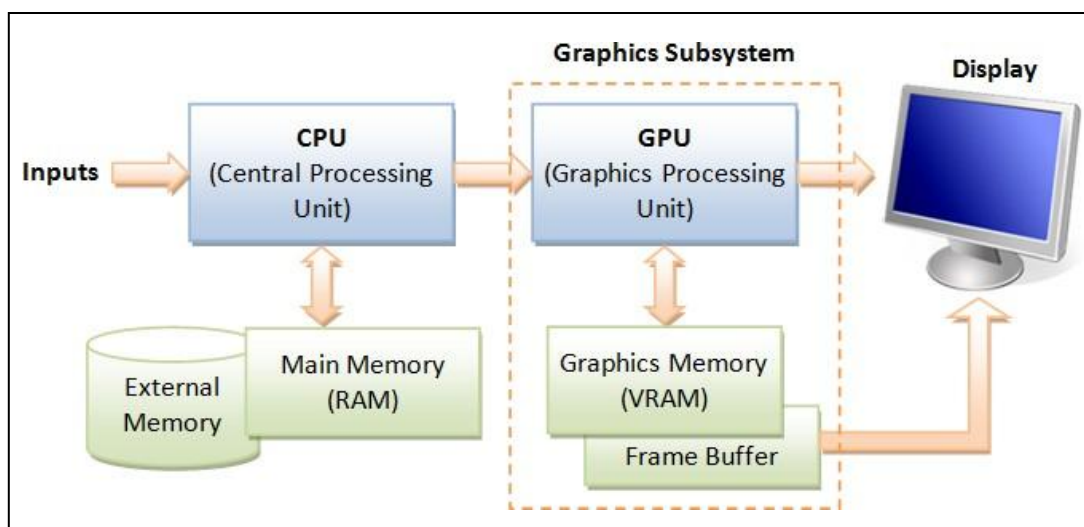


Figure 1.1 Computer Graphics System

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube (CRT) screens soon after the introduction of computers themselves. It has grown to include the creation, storage and manipulation of models and manipulation of models and images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural, and even conceptual structures, natural phenomena, and soon.

Computer graphics today is largely interactive. The user controls the contents, structure and appearance of objects and their displayed images by using input devices, such as a keyboard, mouse, or touch sensitive panel on the screen. The handling of such devices is included in the study of computer graphics, because of the close relationship between the input devices and the display.

1.2 OpenGL

OpenGL (open graphics library) is a standard specification defining a cross language across platform API for writing applications that produce 2D and 3D computer graphics. 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.

OpenGL serves two main purpose:

1. To hide the complexities of interfacing with different 3D accelerators, by presenting programmer with a single, uniform API
2. To hide the differing capabilities of hardware platforms, by requiring that all Implementations support the full OpenGL, featureset.

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects.

Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are:

- **Main GL:** Library has names that begin with the letter **gl** and are stored in a library usually referred to as GL.
- **OpenGL Utility Library (GLU):** This library uses only GL functions but contains code for creating common objects and simplifying viewing. All functions in GLU can be created from the core GL library but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations. Functions in the GLU library begins with the letters **glu**.
- **OpenGL Utility Toolkit(GLUT):** This provides the minimum functionality that should be accepted in any modern windowing system. For the X window system, this library is called GLX, for windows, it is WGL or Wiggle. And for Macintosh, it is AGL. Rather than using different library for each system, we use a readily available library called **GLUT**.

OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL. Instead, you must work through whatever windowing system controls the particular hardware you're using. Similarly, OpenGL doesn't provide high-level commands for describing models of three-dimensional objects. Such commands might allow you to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules.

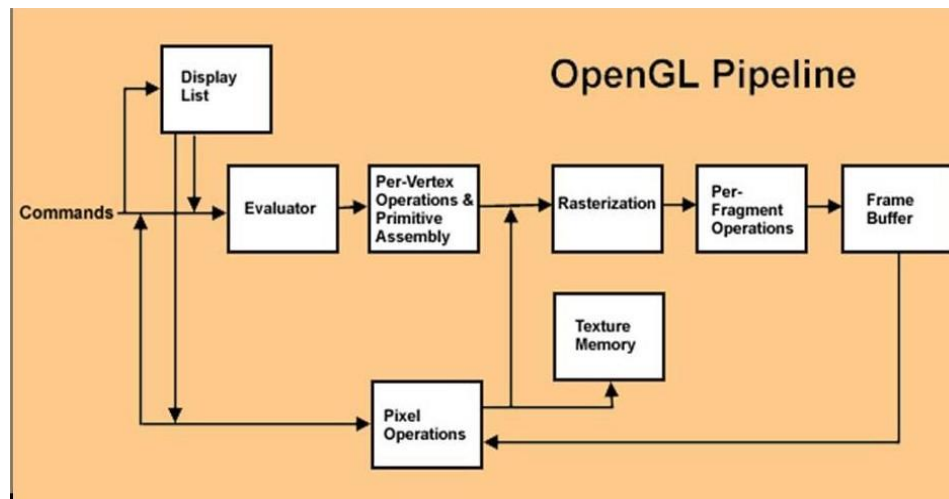


Figure 1.2 OpenGL Graphics Architecture

2. OpenGL SYNTAX

- **glVertex*():**-The glVertex function commands are used with **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. When only x and y are specified, z defaults to 0 and w defaults to 1.
- **glColor*():**-It sets a few four-valued RGBA color. glColor has two major glColor3 and glColor4. glColor3 variants specify new red, green and blue values explicitly and set the current alpha value to 1.0 implicitly. glColor4 variants specify all four color components explicitly.
- **gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top)** which defines a two-dimensional orthographic viewing region. This is equivalent to calling glOrtho.

- ▲ **glClear();**- glClear sets the bitplane area of the window to values previously selected by glClearColor, glClearIndex, glClearDepth, glClearStencil and glClearAccum. Multiple color buffers can be cleared simultaneously by selecting more than one buffer at a time.
- ▲ **glClearColor();**- Specifies the red, green, blue, and alpha values used by **glClear** color buffers. Values specified by glClearColor are clamped to the range 0.0 to 1.0.
- ▲ **glLoadIdentity();**-The current matrix with the identity matrix. It is used to replace the current matrix with the identity matrix.
- ▲ **glMatrixMode(mode);**-Sets the current matrix mode, *mode* can be **GL_MODELVIEW**, **GL_PROJECTION** or **GL_TEXTURE**.
- ▲ **void glutInit(int *argc, char**argv);**-Initializes GLUT, the arguments from main are passed in and can be used by the application.
- ▲ **void glutInitDisplayMode(unsigned int mode);**-Requests a display with properties in mode. The value of mode is determined by the logical OR of options including the color model and buffering.
- ▲ **void glutInitWindowSize(int width, int height);**- Specifies the initial position of left corner of the window in pixels.
- ▲ **int glutCreateWindow(char *title);**-A window on the display. The string title can be used to label the window. The return value provides references to the window that can be used when there are multiple windows.
- ▲ **void glutMouseFunc(void *f(int button, int state, int x, int y));**-Register the callback function f. The callback function returns the button, the state of button after the event and the position of the mouse relative to the top-left corner of the window.
- ▲ **void glutKeyboardFunc(void(*func) (void));**-This function is called every time a key is pressed. The callback function returns the key. The key can be used to resume the game or when you press 'b' or 'B' key to go back to the initial screen or when you press Esc key to exit from the application.

- ▲ **void glutDisplayFunc(void (*func) (void)):-** Register the display function function that executed when the window needs to be redrawn.
- ▲ **void glutSpecialFunc(void (*func)(void)):-** This function is called when you press special keys in the keyboard like arrow keys, function keys etc. In our program, the function is invoked when the up arrow or down arrow key is pressed for selecting the options in the main menu and when the left or right arrow key is pressed for moving the object(car) accordingly.
- ▲ **glutPostRedisplay():-** Which requests that the display callback be executed after the current callback returns.
- ▲ **void glutMouseFunc(void (*func) void):-** This function is invoked when mouse keys are pressed. This function is used as an alternative to the previous function i.e, it is used to move the object(car) to right or left in our program by clicking left and right button respectively.
- ▲ **void glutMainLoop():-** Cause the program to enter an event processing loop. It is the last statement in main function.

3. SYSTEM REQUIREMENTS SPECIFICATION

3.1 HARDWARE REQUIREMENTS

- RAM: 2GB and higher
- Hard Disk: 40GB and higher
- 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: Linux Operating System
- Compiler – GCC compiler
- Graphics library: GL/glut.h
- OpenGL 2.0

4. USER INTERACTION

Using Alphabetical Keys:

‘F’ or ‘f’: Move Ship Forward ‘R’
or ‘r’: Move Ship Backward ‘S’ or
‘s’: Stop theShip
‘N’ or ‘n’: Shift to Night Mode
‘D’ or ‘d’: Shift to Day Mode ‘Q’
or ‘q’: Quit

Using Special Keys:

LEFT BUTTON: Move Ship Backward

RIGHT BUTTON: Move Ship Forward

DOWN BUTTON: Stop the Ship

5. SNAPSHOTS



Figure 4.1 - Ship is Stable

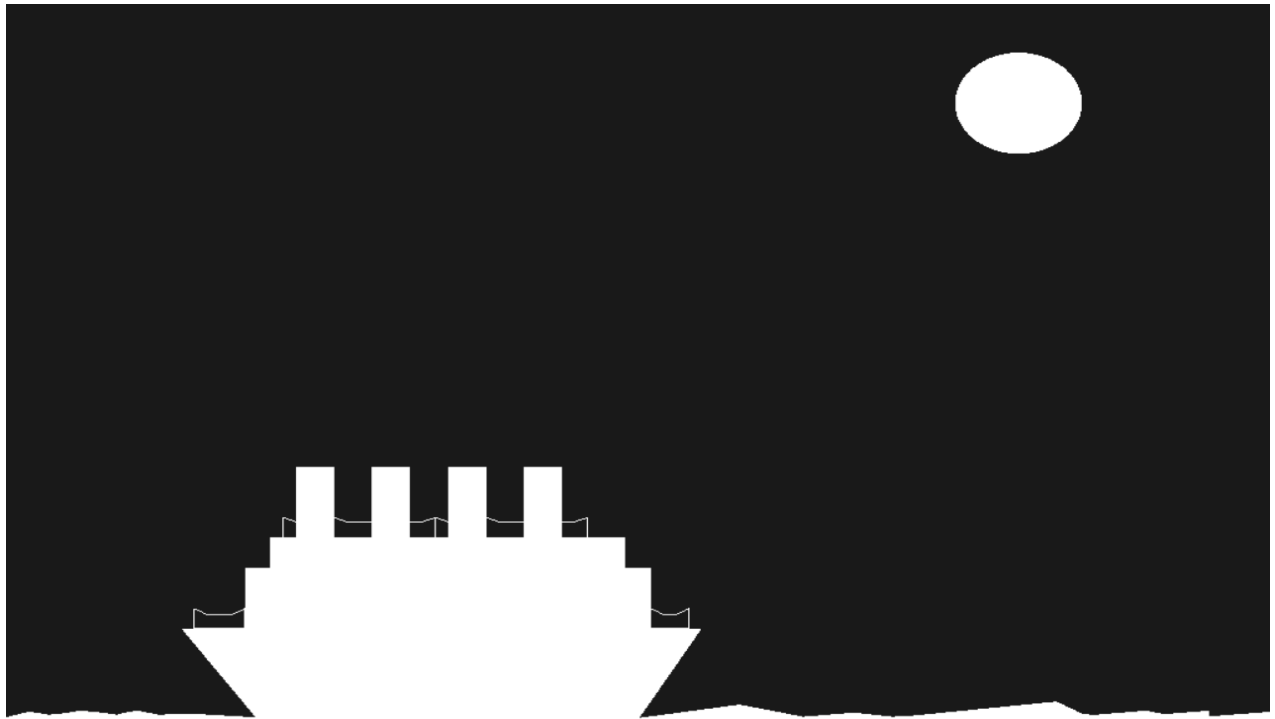


Figure 4.2 – Ship is moving

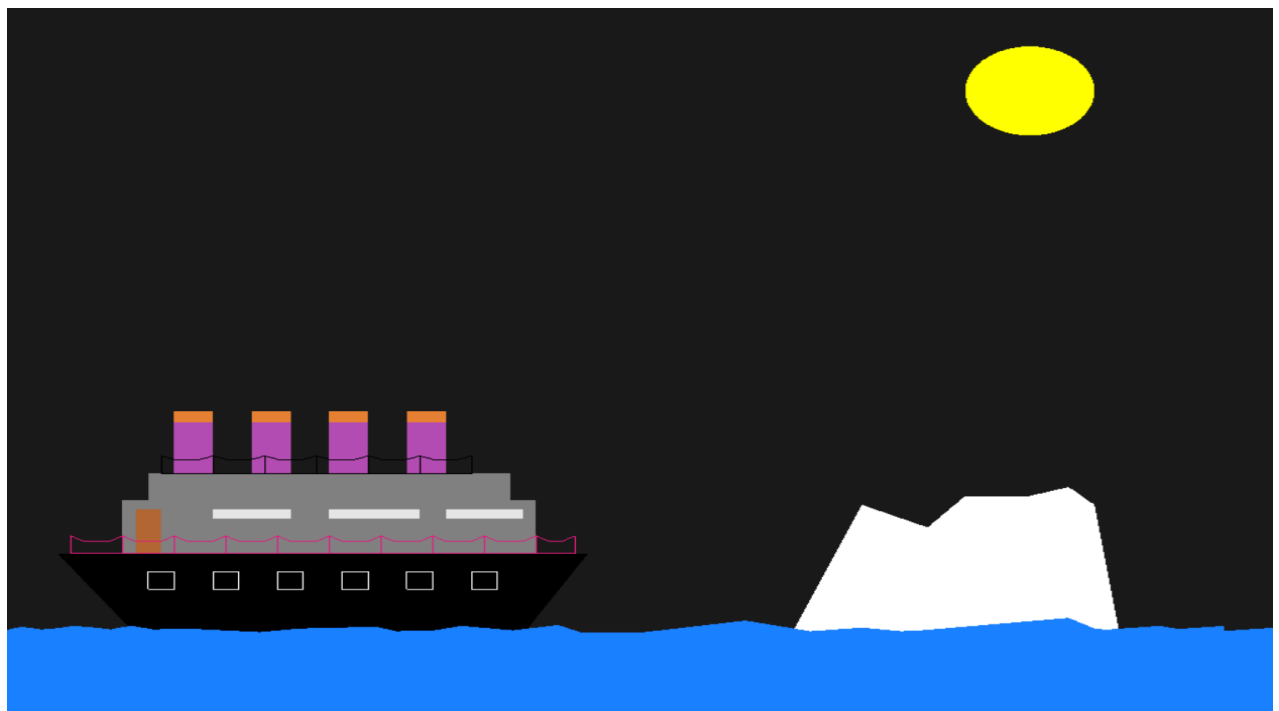


Figure 4.3 – Ship and the Iceberg



Figure 4.4 – Ship is collided with the Iceberg

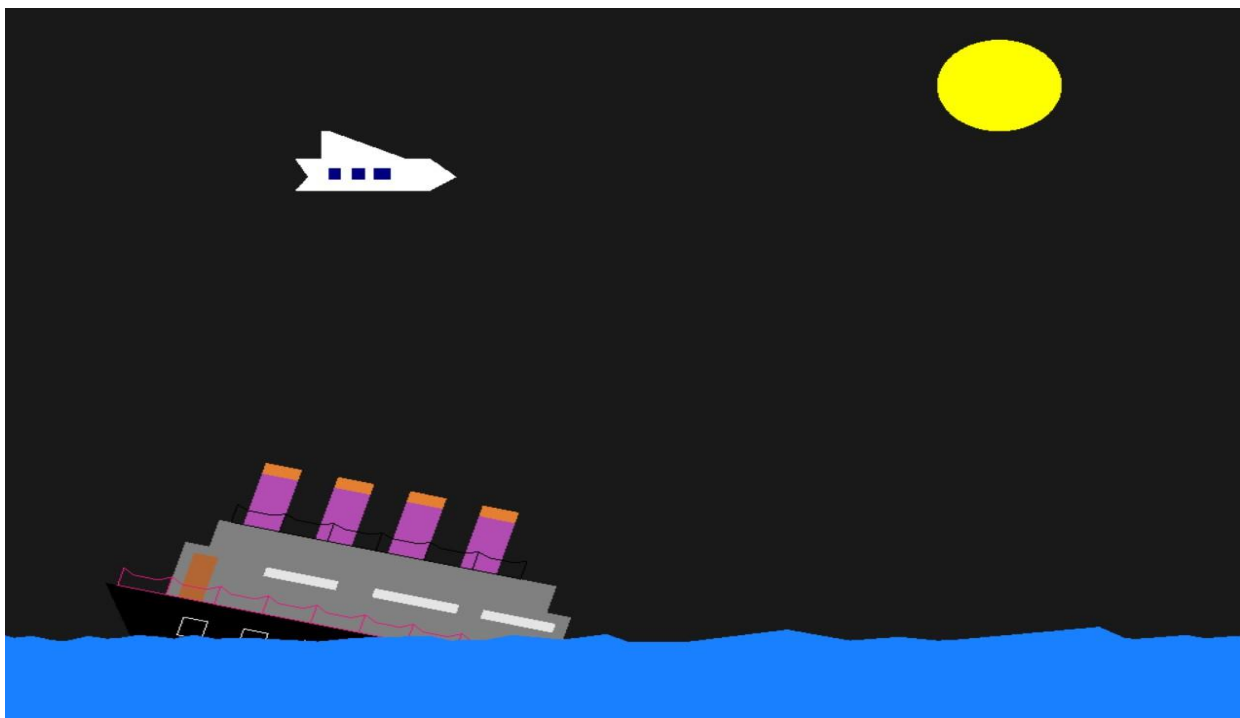


Figure 4.5 – Ship has sunk and the rocket is flying

6. CONCLUSION

We have attempted to design and implement “**Sinking Ship**”. OpenGL supports enormous flexibility in the design and the use of OpenGL graphics programs. The presence of many built-in classes methods 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. We regret any errors that may have inadvertently crept in.

7. REFERENCES

http://lazyfoo.net/tutorials/OpenGL/02_matrices_and_coloring_polygons/index.php<http://csciwww.etsu.edu/barrettm/4157/OpenGL%20Functions.htm> <https://www.glprogramming.com/red/chapter01.html>
https://www2.cs.duke.edu/courses/cps124/fall01/resources/ogl_ref.pdf