**CHAPTER 1**

**INTRODUCTION**

Computer Graphics is an art of drawing pictures, lines,charts,etc using computers with the help of programming. Computer graphics is made up of number of pixels. Pixel is the smallest graphical picture or unit represented on the computer screen. It is often abbreviated as CG, though sometimes erroneously referred to as computer-generated imagery (CGI).Today, we find computer graphics used routinely in such diverse areas as science engineering, medicine, business, industry, government, art and many more areas.

The development of computer graphics has made computer easier to interact with, and better for understanding and interpreting many types of data. A major use of computer graphics is in design processes, particularly for engineering and architectural systems. But, today almost all products are now computer designed. Generally referred as CAD (Computer Aided Design) . This is used mainly in the development or design of buildings, aircrafts, computers, and many other products.

Today, computer graphics is widespread. Such imagery is found in and on television, newspapers, weather reports, and in a variety of medical investigations and surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. In the media "such graphs are used to illustrate papers, reports, theses", and other presentation material. Many tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: two dimensional (2D), three dimensional (3D), and animated graphics. As technology has improved, 3D computer graphics have become more common, but 2D computer graphics are still widely used. Computer graphics has emerged as a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Over the past decade, other specialized fields have been developed like information visualization, and scientific visualization more concerned with "the visualization of three dimensional phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic (time) component".[1]

Here, we have used OpenGL as the graphics software to implement our mini project. OpenGL (Open graphics Library) is nothing but a cross-language, cross-platform application programming interface for rendering 2D and 3D vector graphics[2]. It was developed by the Silicon Graphics Inc in 1992.

The interface between the application program and the graphics system can be specified through that set of function that resides in graphics library. This specification is called as Application Program Interface. The API is basically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering. OpenGL is a window and operating system independent. Most of the applications are designed to access OpenGL directly to the function using three libraries. Function in the main GL library have name that begin with the letter gl and stored in the library. The second is the OpenGL utility library (GLU). This library uses only GL function but contains codes for creating common object and viewing. Rather than using a different library for each system we used available library called as the OpenGL utility toolkit (GLUT). It is used as #include<glut.h>[2].

**Summary**

This chapter gives an introduction to computer graphics and an overview of openGL.

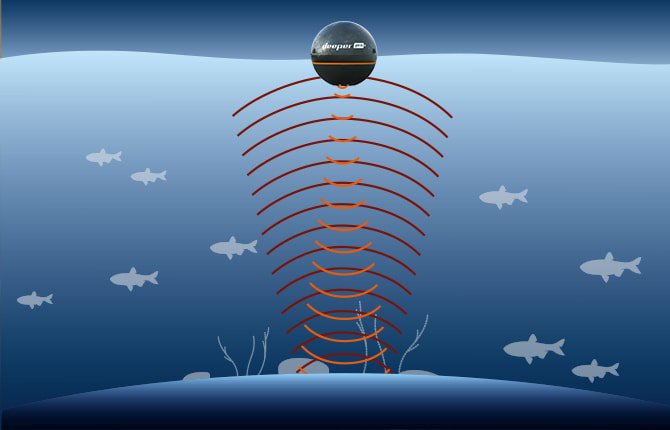
**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 EXISTING SYSTEM**

A sonar device sends pulses of sound waves down through the water. When these pulses hit objects like fish, vegetation or the bottom, they are reflected back to the surface. The sonar device measures how long it takes for the sound wave to travel down, hit an object and then bounce back up. It’s the same echo-location system bats and dolphins use. This information enables the device to judge the depth of the object it reflected off. It also measures the strength of the returning pulse – the harder the objects, the stronger the return pulse.

Once a returning pulse is received, another one is sent out. Because sound waves travel at roughly one mile a second in water, sonars can send multiple pulses per second. The Deeper PRO and Deeper PRO+send 15 pulses per second. The returning sound pulses are converted into electrical signals and then displayed, showing anglers the depth and hardness of the bottom and any objects in between**.**



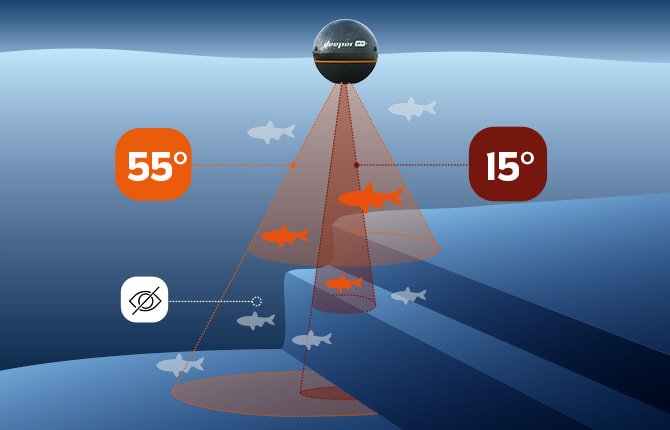
***Fig2.1.Formation of Waves***

**Here’s how it works**

Sonars send out pulse of sound to locate objects. Sound travels in waves, not straight lines, and these waves expand in cones, getting wider and wider.Most sonars can control the range of the sound wave cone by changing the scanning beam frequency. This is important because in different fishing situations different scanning beams will be more or less effective.

Wide beam scanning (usually 40° to 60° angle) is good for quickly scanning large areas and geting overall information on depth and bottom structure, but the accuracy and detail will be lower. Wide beam scanning is best suited for shallower waters because the cone covers a wider area, the deeper it scans. This means if you are scanning at a depth of 45ft / 13,7 m you will see objects in an area that has a 47ft / 14,3 m diameter.

Narrow beam scanning (around 10° to 20°) gives a more precise picture but covers a smaller area. It is better for finding the exact location of fish. Narrow beam scanning is also better suited for deeper water, as the cone does not spread as wide[3].



***Fig2.2.Working of SONAR***

[4] Models a fish like structure using polygons, points and lines. The project shows fish like structures with movement.This code can be reused in the graphical simulation of SONAR.The following code snippet demonstrates its usage.

glVertex2f(-.9+x,.05+sin(8\*x));  
glVertex2f(-.8+x,.15+sin(8\*x));  
glVertex2f(-.7+x,.12+sin(8\*x));  
glVertex2f(-.6+x,0+sin(8\*x));  
glVertex2f(-.7+x,-.12+sin(8\*x));  
glVertex2f(-.8+x,-.15+sin(8\*x));  
glVertex2f(-.9+x,-.05+sin(8\*x));  
glEnd();  
glColor3f(0,0,0);  
glBegin(GL\_LINE\_STRIP);  
glVertex2f(-.75+x,.08+sin(8\*x));  
glVertex2f(-.8+x,.04+sin(8\*x));  
glVertex2f(-.75+x,0+sin(8\*x));  
glEnd();

for(t=45;t<=-45;t+10)  
{  
tr=c\*t;  
glVertex2f(r\*cos(tr)+x,r\*sin(tr)+sin(8\*x));  
}

[5]Models a sphere like structure using polygons, cylinders and circles. The project shows sphere like structure without any movement in it. This code can be reused in the graphical simulation of SONAR. The following code snippet demonstrates its usage.

glTranslatef(0.0,0.0,-5.0);

glColor3f(0.9, 0.3, 0.2);

glRotatef(xRotated,1.0,0.0,0.0);

glRotatef(zRotated,0.0,0.0,1.0);

glScalef(1.0,1.0,1.0);

glutSolidSphere(radius,20,20);

[6]Models a curves like structure using line strips. The project shows curve like structure with movements in it. This code can be reused in the graphical simulation of SONAR. The following code snippet demonstrates its usage.

for(t = -10.0; t < 10.0; t += 0.1){

x = sin(t) + ((1/2) \* sin(5 \* t)) + ((1/4) \* cos(2.3 \* t));

y = cos(t) + ((1/2) \* cos(5 \* t)) + ((1/4) \* sin(2.3 \* t));

glVertex2f(x, y);

}

**Summary**

This chapter examines the related work based on the existing system.

**CHAPTER 3**

**PROPOSED SYSTEM**

In this project we model a simulation of sonar with known OpenGL functions available to us. The main objective of this project is to design a simulation which helps to detect the objects in the sea. This system can increase faster navigation and less risk for the sailors.

Here Sonar is simply making use of an echo. When an animal or machine makes a noise, it sends sound waves into the environment around it. Those waves bounce off nearby objects, and some of them reflect back to the object that made the noise. It's those reflected sound waves that you hear when your voice echoes back to you from a canyon. Whales and specialized machines can use reflected waves to locate distant objects and sense their shape and movement.

The modules used in this system are:

* Submarine
* Fish
* Display
* Menu
* Transformation method

In this simulation project we make use of the following scenarios.

The first scenario shows the graphical representation of sea and fishes in it.

The second scenario shows the movement of submarine in the sea.

The last scenario consists of a pop up menu which activates the SONAR and the procedures related to it.

**Summary**

This chapter demonstrates the simulation of detection of objects in the sea using computer graphics.

**CHAPTER 4**

**SOFTWARE REQURIMENT SPECIFICATION**

**4.1 Definition**

A software requirements specification (SRS) is a description of a software system to be developed.Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. The SRS is divided into functional and non-functional requirement.[7]

**4.2 Non-Functional Requirements**

In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture, because they are usually architecturally significant requirements.[8]

**4.3 Functional Requirement**

The functional requirement are the requirements which are needed to develop the software application. The requirement are broadly classified into 2 categories, they are,

1. Hardware Requirement
2. Software Requirement

**4.3.1 Hardware Requirements**

* Computer components with Graphic enabled
* Mouse, Keyboard and computer accessories
* 500 Mb space in HDD
* LCD Screen

**4.3.2 Software Requirement**

* Operating System: Windows 10 and higher, ubuntu or linux
* Programming language: c/ c++
* Graphics library used: OpenGL
* IDE: Code blocks, visual studio
* Compiler used: mingw, gcc, G++

**Summary**

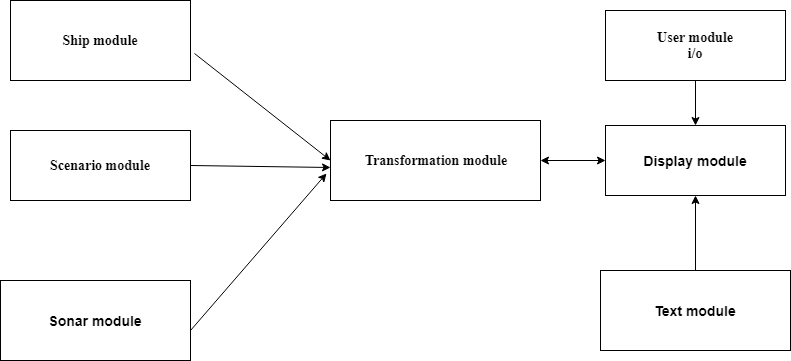
This chapter gives view on the requirement list needed to develop the graphics application.

**CHAPTER 5**

**DESIGN**

**5.1 Block Diagram**

The modular design will describe the various components used in this project .Referring to the fig.5.1 it shows different modules present in the proposed graphic project.

****

***Fig5.1Modular Design***

**5.1.1 Text Module**

* Provides the descriptive overview of all the working modules.
* It is implemented using the gltext().
* Allows user to have an interaction with the working modules.

**5.1.2 User Module**

* Allows user to have an interaction with the working modules.
* Implemented using menus and buttons.

**5.1.3 Display Module**

* Gives the view of different modules in the process of simulation of sonar.
* All the transformation could be viewed.

**5.1.4 Transformation Module**

* Depicts different operations such as movement of ship and fishes

**5.1.5 Ship Module**

* Draw the ship using polygons and other openGL function.
* It is repeatedly called to render the simulation of sonar.

**5.1.6 Scenario Module**

* In the scenario module it explains the basic scenario need for simulation of sonar.

**5.1.7 Sonar Module**

* In sonar module it tells us how the sonar wave is detected .
* When the object is found it displays the message.
* It calculates the distance of the object.

**5.2 Hardware Design**

****

***Fig5.2 Hardware Design***

**5.2.1 Keyboard, Mouse**

* Computer keyboards is a typewriter-style device which uses an arrangement of buttons

or keys to act as mechanical levers or electronic switches.

* A computer mouse is a hand-held pointing device that detects two-dimensionalmotion

relative to a surface.

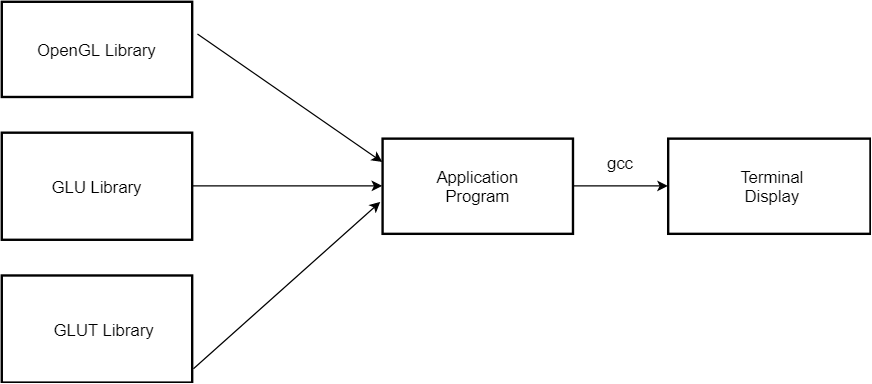
**5.2.2 GPU/Host Processor**

* A graphics processing unit (GPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to display device.

**5.2.3 CRT/CCD Screen**

* A charge coupled device(CCD) is a device for the movement of electrical charge ,usually from within the device to an area where the charge can be manipulated, for example into a digital value.

**5.3 Software Design**

****

***Fig5.3 Software Design***

**5.3.1 Application Program**

* A computer program used for specific kind of task like word processing.
* An Application program is also known as application software.

**5.3.2 Terminal Display**

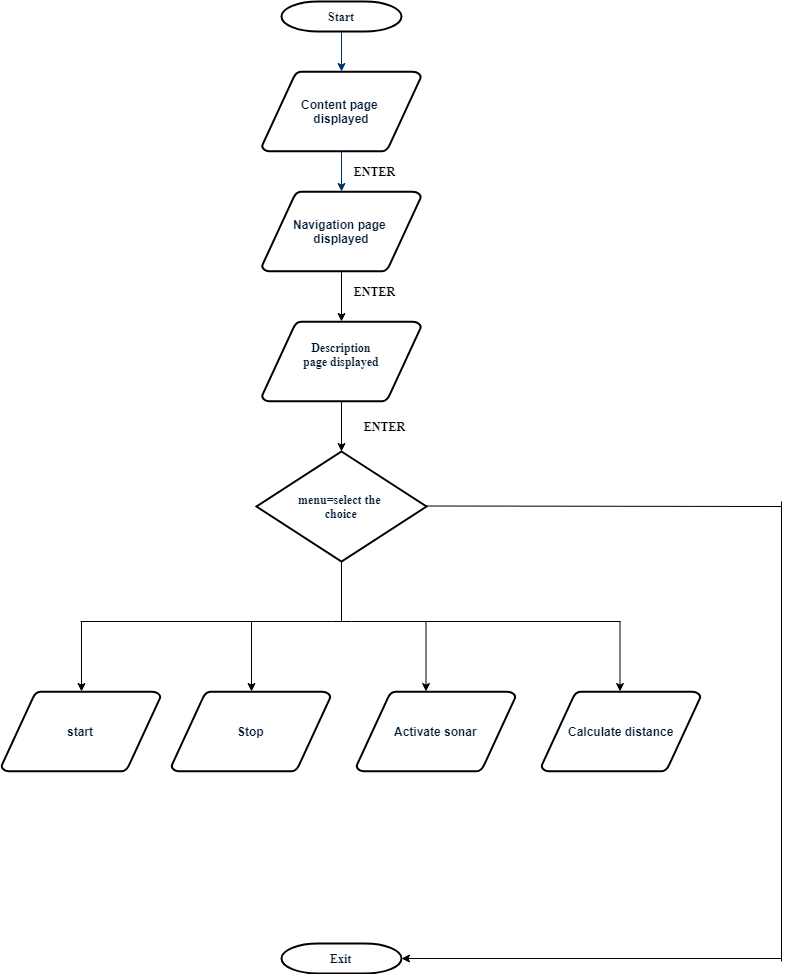
* A computer terminal is an electronic hardware device that is used for entering data into, and display or painting data from , a computer or a computing system.

**5.4 Flowchart**

A flowchartis a type of diagram that represents an algorithm, workflow or process. Flowchart can also be defined as a diagrammatic representation of an algorithm. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields. Flowcharts depict certain aspects of processes and are usually complemented by other types of diagram.Any drawing program can be used to create flowchart diagrams, but these will have no underlying data model to share data with databases or other programs such as project management systems or spreadsheet. Some tools such as Inscape and Microsoft Visio offer special support for flowchart drawing. Many software packages exist that can create flowcharts automatically, either directly from a programming language source code, or from a flowchart description language.

Types of flowchart:

* Documentflowcharts, showing controls over a document-flow through a system.
* Data flowcharts, showing controls over a data-flow in a system.
* System flowcharts, showing controls at a physical or resource level.
* Program flowchart, showing the controls in a program within a system.[9]



***Fig5.4 Flowchart***

**Summary**

This chapter gives the overall pictorial review of Simulation of Sonar with the detailed explanation of different modules.

**CHAPTER 6**

**IMPLEMENTATION**

Implementation is the core step in software development life cycle .Implementation gives the detailed view of the project and describes the pseudo code and various functions in the project. It will also give insight about various inbuilt modules and functions on openGL.

**6.1 Algorithm**

An algorithm is a step-by-step instruction to execute the program. Figure 6.1 shows the algorithm of the project.

Step 1: Initialize the graphics library.

Step 2: Display the starting page.

Step 3: Take user input to navigate to the next page.

Step 4: Display the description page.

Step 5: Ask user to select menu

While True

If choice==1

Start()

If choice==2

Stop()

If choice==3

Activate\_sonar()

If choice==4

Calculate\_distance()

If choice==5

Exit(0)

**6.2 Code Snippet**

**6.2.1 Code snippet-1**

The following code is used to draw the ship.

**6.2.2 Code Snippet-2**

void ship(float a)

{

glColor3f(0.3,0.3,0.3);

glBegin(GL\_POLYGON);

glVertex2f(w1=200+a,x1=300);

glVertex2f(y11=400+a,z1=300);

glVertex2f(430+a,400);

glVertex2f(170+a,400);

glEnd();

}

glColor3f(0.8,0.8,0.8);

glBegin(GL\_POLYGON);

glVertex2f(200+a,400);

glVertex2f(400+a,400);

glVertex2f(400+a,430);

glVertex2f(200+a,430);

glEnd();

The following code is used to draw fish.

void fish(float a)

{

glColor3f(1,0,0);

glBegin(GL\_TRIANGLES);

glVertex2f(100+800+a,100+100);

glVertex2f(100+800+a,120+100);

glVertex2f(120+800+a,110+100);

glEnd();

}

glBegin(GL\_POLYGON);

glVertex2f(110+800+a,110+100);

glVertex2f(130+800+a,130+100);

glVertex2f(150+800+a,110+100);

glVertex2f(130+800+a,90+100);

glEnd();

}

glVertex2f(780-600-a,210);

glEnd();

**6.2.3 Code Snippet-3**

The following code is used to draw a submarine

void submarine()

{

glColor3f(0,0,0);

glBegin(GL\_POLYGON);

glVertex2f(p1=1000+30-300,q1=150);

glVertex2f(1000+30-300,100);

glVertex2f(1050-300,100);

glVertex2f(1080-300,120);

glVertex2f(1080-300,130);

glVertex2f(1050-300,150);

//glVertex2f(1000,150);

glEnd();

}

glBegin(GL\_POLYGON);

glVertex2f(1080-300,130);

glVertex2f(1080-300,120);

glVertex2f(1100-300,110);

glVertex2f(1250-300,110);

glVertex2f(r1=1250-300,s111=140);

glVertex2f(1100-300,140);

//glVertex2f(1080,130);

glEnd();

**6.2.4 Code Snippet-4**

The following code is used to draw the water.

**6.2.5 Code Snippet-5**

void water()

{

glColor3f(0,0,0.5);

glBegin(GL\_POLYGON);

glVertex2f(0,0);

glVertex2f(0,300);

glVertex2f(1386,300);

glVertex2f(1386,0);

glEnd();

}

The following code is used to draw the curves.

void curves(float x,float y,float r) {

float i;

glColor3f(1,1,1);

glLineWidth(5);

glBegin(GL\_LINES);

for(i=180.0f;i<360;i++)

glVertex2f(r\*cos(M\_PI\*i/180.0)+x,r\*sin(M\_PI\*i/180.0)+y);

glEnd();

usleep(10000);

glFlush();

}

**6.3 List of inbuilt OpenGL functions used**

* The following are list of OpenGL functions used in the project.
* void glColor3f(GLfloat red,GLfloat green,GLfloat blue): This function sets the presentRGB colors. Different color is given to the object using the color parameters such asred,green,blue.
* void gluOrtho2D(GLdouble left,GLdouble right,GLdouble bottom,GLdouble top): Setsup a two-dimensional orthographic viewing region.
* void glLoadIdentity(void): glLoadIdentity replaces the current matrix with the identitymatrix.
* void glClear(GL\_COLOR\_BUFFER\_BIT):Clears all buffer whose bits are set in mask.
* The mask is formed by logical OR of values defined in gl.h GL\_COLOR\_BUFFER-BIT refers to color buffers.
* void glBegin() and void glEnd():The glBegin and glEnd functions delimit the vertices of a primitive group of like primitives.
* void glFlush(void): The glFlush function forces execution of OpenGL functions in finite time. This function has no parameters. This function does not return a value.
* void glMatrixMode(GL\_PROJECTION): Applies subsequent matrix operations to the projection matrix stack.
* void glutMainLoop(void): glutMainLoop enters the GLUT event processing loop. Thisroutine should be called at most once in a GLUT program. Once called, this routine willnever return. It will call as necessary any callbacks that have been registered.
* void glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE): Bit mask to select a singlebuffered window. Bit mask to select an RGB mode window.
* void glutKeyboardFunc(void (\*func)(unsigned char key,int x, int y)): glutKeyboardFuncsets 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.
* Void glutInitWindow size(int width int height): This function specifies the width and height of the window in pixels.
* Void glutInitWindow Postion(int x,int y):This function specifies the initial position of the top-left corner of the window in pixel.
* Void glutCreateWindow(char \*title):The glutCreateWindow()creates the window on the display.The string title can be used to label the window .The return value provides reference to the window that can be used when there are multiple windows.
* Void myinit(void):Sets the background color and viewing attribute.

**Summary**

This chapter contains the Implementation part of the project.It include the algorithm of the project and the list of OpenGL functions used.

**CHAPTER 7**

**RESULTS AND SNAPSHOTS**

**7.1 Snapshot-1**

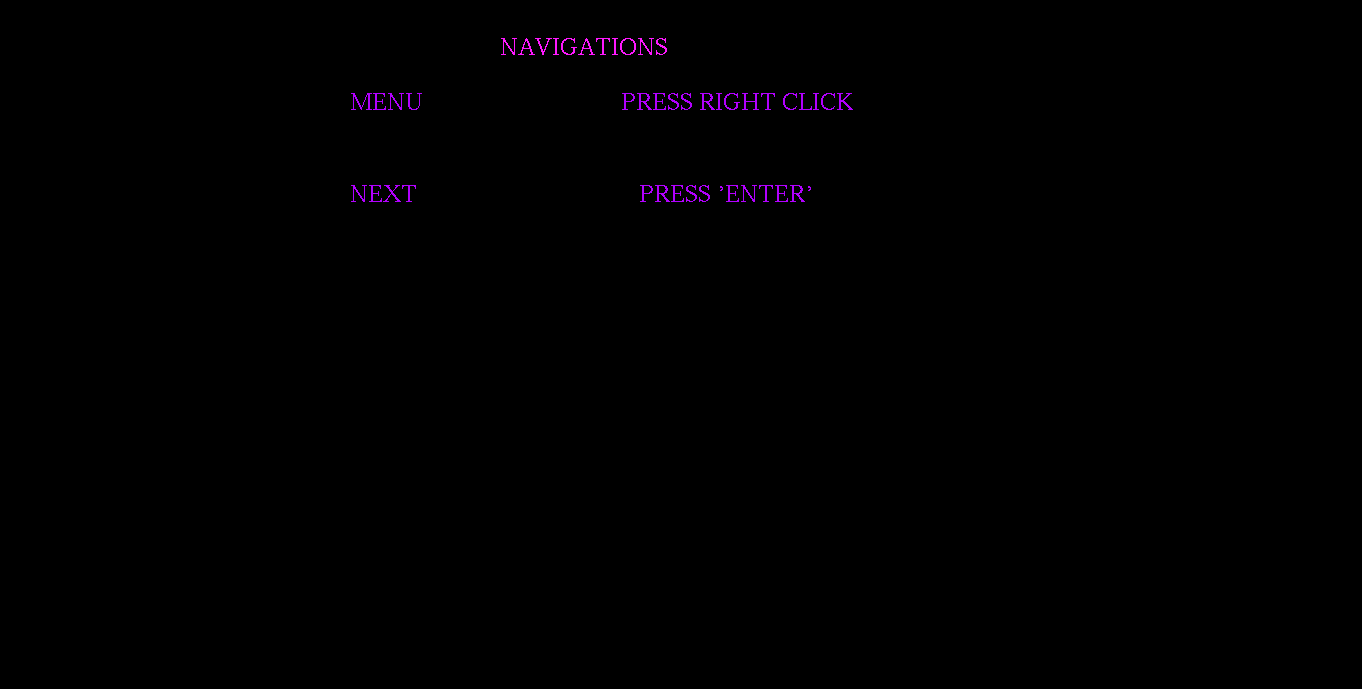
This figure shows shows the initial view of the project.



***Fig 7.1 Front Page***

**7.2 Snapshot-2**

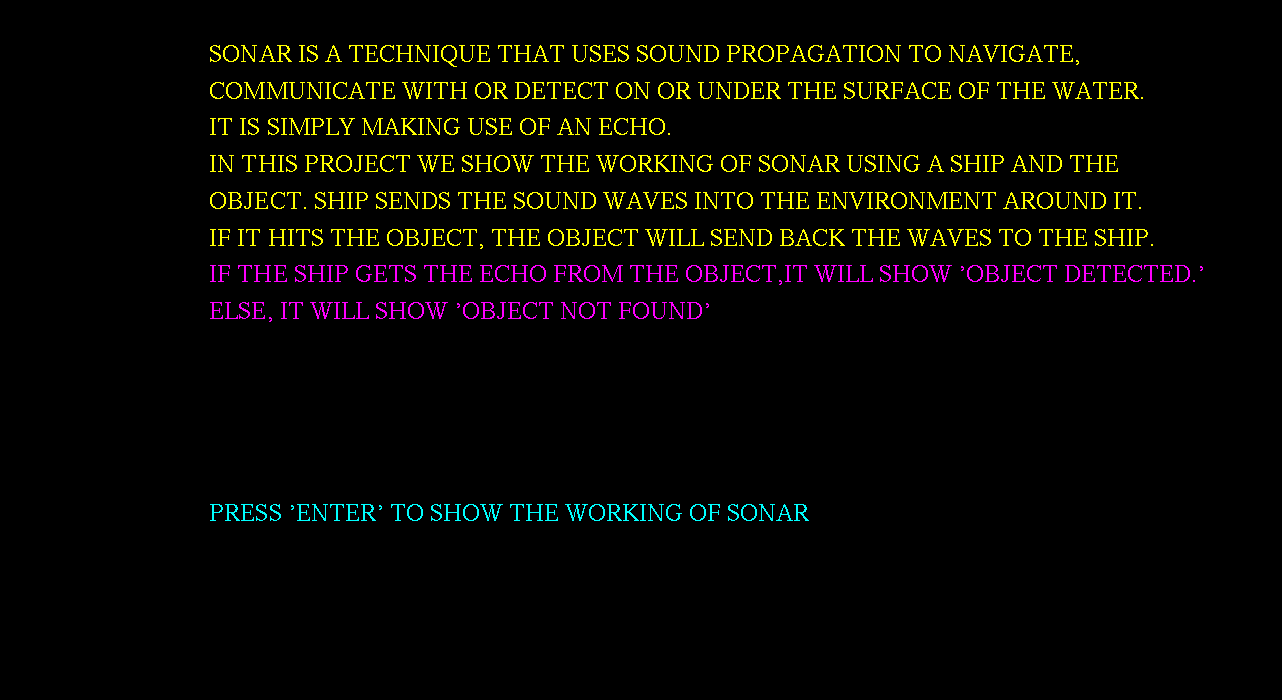
This figure shows shows the navigation page of the project.



***Fig 7.2 Navigation page***

**7.3 Snapshot-3**

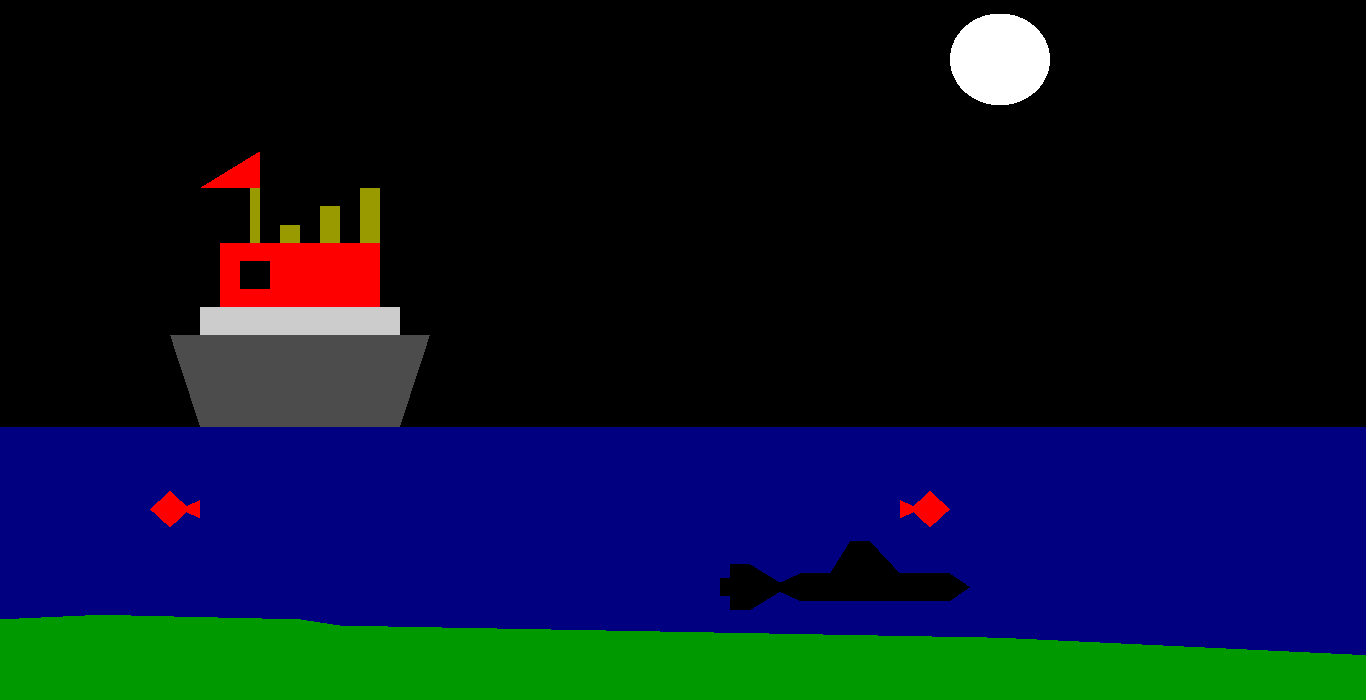
This figure shows shows the description of the project.



***Fig 7.3 Description Page***

**7.4 Snapshot-4**

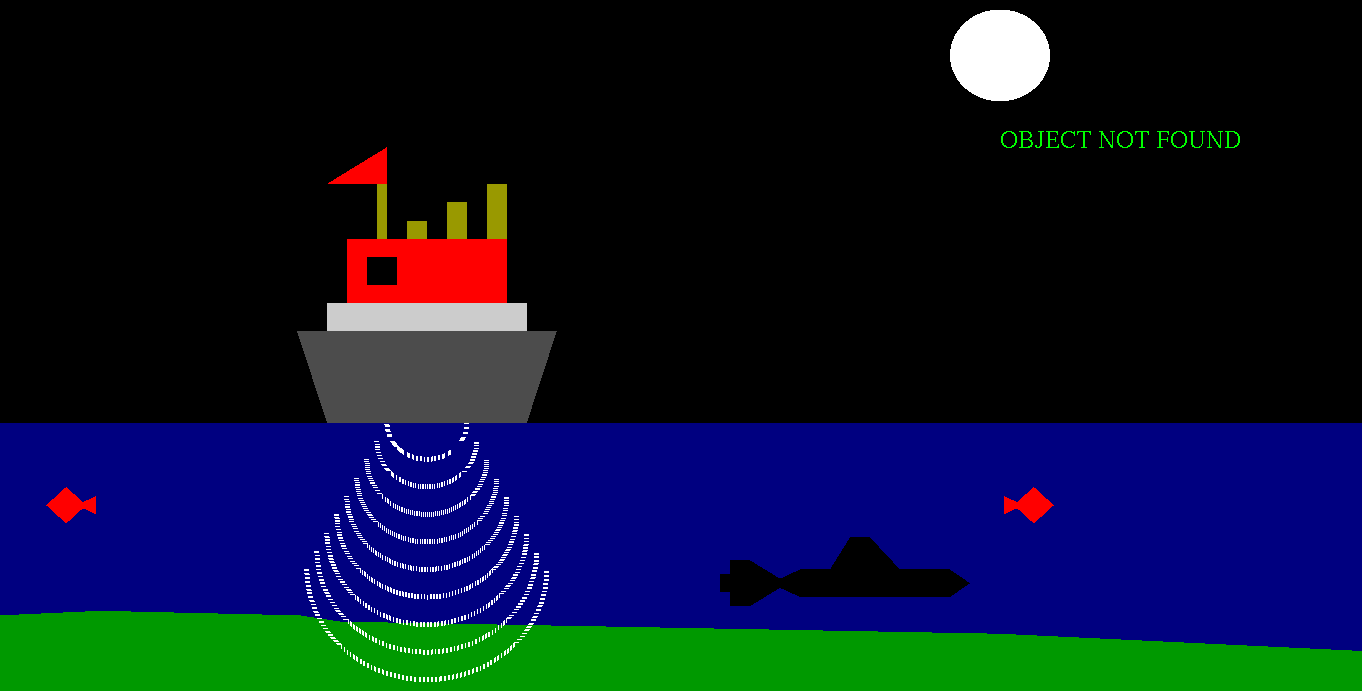
This figure shows shows the initial scenario of the project.



***Fig 7.4 Initial scenario***

**7.5 Snapshot-5**

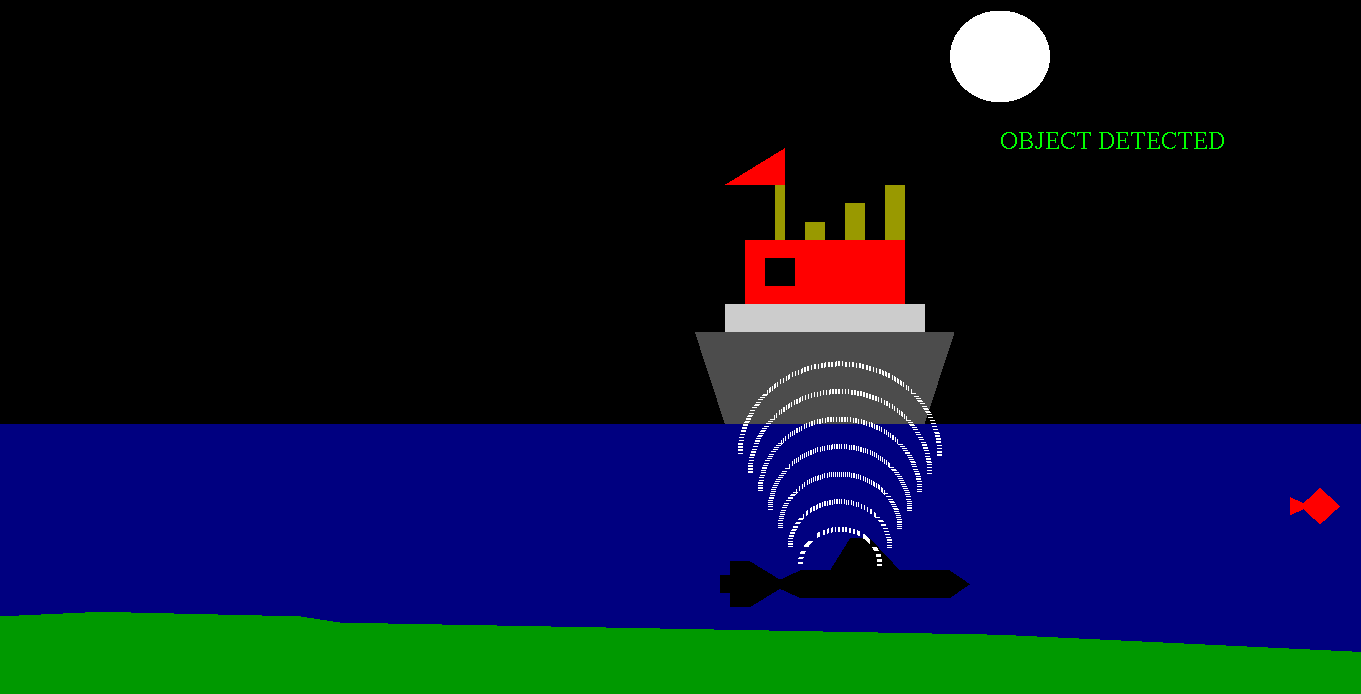
This figure shows shows the activation of sonar.



***Fig 7.5 Activation of sonar***

**7.6 Snapshot-6**

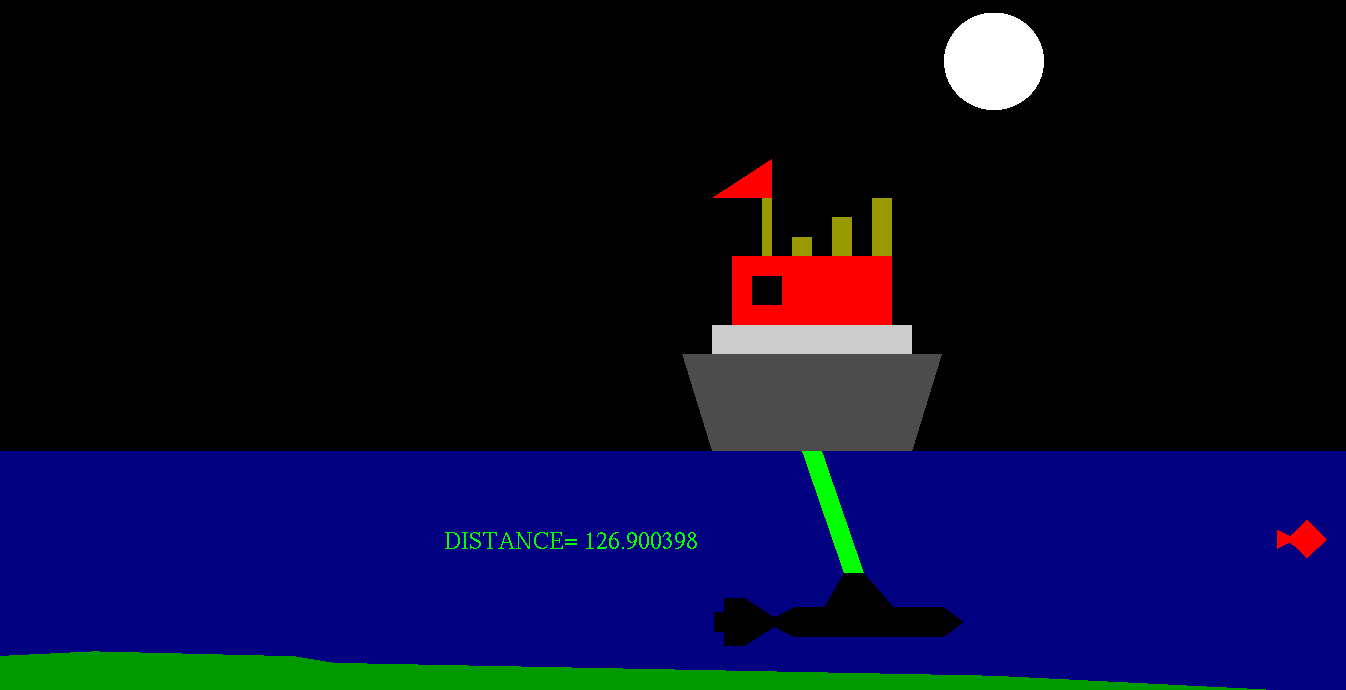
This figure shows the detection of sonar waves.



***Fig 7.6 Detection of sonar***

**7.7 Snapshot-7**

This figure shows calculates the distance between the object.



***Fig 7.7 Calculation of distance***

**Summary**

This chapter contains implementation part of the project. It includes the snapshot of the project.

**Chapter 8**

**CONCLUSION AND FUTURE ENHANCEMENT**

**8.1CONCLUSION**

In the whole project we have tried to give a clear understanding of graphical simulation of Sonar using computer graphics using different functions of OpenGL.This project mainly tries to give idea of certain scenarios that takes place in simulation of Sonar waves when the object is present under sea. The overall implementation of the code is done in CodeBlocks and the programming language used is OpenGL which is mainly used to design the computer graphics project.

**8.2 FUTURE ENHANCEMENT**

This project shows certain simulation process based on the working of SONAR. As a future work one can demonstrate all the possible simulation process or different techniques of simulation of Sonar. In this project,one can get the better detection of objects if there was continuous emission of waves from the ship in the state of motion. One can give a better working visualization using OpenGL’s graphic library. The proposed project uses OpenGL’s graphic function in 2D.As an improvement one can make use of 3D graphics for the demonstration.