**Chapter 1**

**INTRODUCTION**

**1.1 Computer Graphics**

Early Graphics System Used for general purpose computers with the standard Von Neumann architecture. Such computers are characterized by the single processing unit that processes a single instruction at a time.

There are different types of Graphics architectures which are:

* Display processors: The earliest attempts to build special purpose graphics system were concerned primarily with relieving the general purpose computer from the task of refreshing the display continuously. These **display processors** had conventional architecture but included instructions to display primitives on the CRT.
* Pipeline Architectures: The availability of inexpensive solid state memory led to the universality of raster display. For computer graphics application, the most important use of custom VLSI circuits has been in creating **pipeline Architectures***.*
* The graphics Pipeline: Each object comprises a set of graphical primitives. Each primitive comprises set of vertices.
* Vertex processing: The assignment of vertex colors can as simple as program specifying a color or as complex as computation of a color from a physically realistic lightning model that incorporates the surface properties of the object and the characteristic light sources in the scene.
* Clipping and primitive assembly: We must do clipping because of the limitations of that no imaging system can see the whole world at once. The human retina has a limited size corresponding to an approximately 90-degree field of view.

**1.2 OpenGL:**

OpenGL is strictly defined as “a software interface to graphics hardware” In essence; it is a 3D graphics and modelling library that is highly portable and very fast. Using OpenGL, you can create elegant and beautiful 3D and 2D graphics with exceptional visual quality. The Greatest advantage to using OpenGL is that it is orders of magnitude faster than a ray tracer or software rendering engine. OpenGL is intended for use with computer hardware that is designed and optimized for the display and manipulation of 3D graphics.

The interface between an application program and a graphics system can be specified through a set of functions that resides in a graphics library. These specifications are called the application programmer’s interface (API). The application program sees only the API and is thus shielded from the details of the both softer and hardware implementation of the graphics library. The software drivers are responsible for interpreting the output of API and converting these data to a form that is understood by the particular hardware. From the perspective of the application program, the functions available through the API should match the conceptual model that the user wishes to employ to specify images

## **1.3 About Project**

The mini project is about “Fishing Occupation”. A fisherman is supposed to search for the fishes and catch them with the net. After catching the fishes he reaches the market only with the fishes. The boat could be initiated, stopped and made to move at a faster pace while the boat is in motion using the keyboard keys and the mouse button to quit. Thereby the fisherman reaches the bank of the river in display 4 function. In addition to this an option is provided to go back in the reverse direction in the third display. We have used the inbuilt as well as the user defined functions to make the project work.

The scenario of travelling from one bank of the river to the other side of the river is depicted using Microsoft visual studio application. The project has been designed using visual C++ as language with OpenGL API.

**1.4 Problem Definition**

To develop an animation demonstrating the working of Fishing Occupation.

**Chapter 2**

**LITERATURE SURVEY**

People use the term “computer graphics” to mean different things in different context. Computer graphics are pictures that are generated by a computer. Everywhere you look today, you can find examples, especially in magazines and on television. Some images look so natural you can’t distinguish them from photographs of a real scene. Others have an artificial look, intended to achieve some visual effects.

There are several ways in which the graphics generated by the program can be delivered.

* Frame- by- frame: A single frame can be drawn while the user waits.
* Frame-by-frame under control of the user: A sequence of frames can be drawn, as in a corporate power point presentation; the user presses to move on to the next slide, but otherwise has no way of interacting with the slides
* Animation: A sequence of frames proceeds at a particular rate while the user watches with delight.
* Interactive program: An interactive graphics presentation is watched, where the user controls the flow from one frame to another using an input device such as a mouse or keyboard, in a manner that was unpredictable at the time the program was written. This can delight the eye.

**2.1 History**

OpenGL was developed by **‘Silicon Graphics Inc’** (SGI) on 1992 and is popular in the gaming industry where it competes with the Direct3D in the Microsoft Windows platform. OpenGL is broadly used in CAD (Computer Aided Design), virtual reality, scientific visualization, information visualization, flight simulation and video games development.

OpenGL is a standard specification that defines an API that is multi-language and multi-platform and that enables the codification of applications that output computerized graphics in 2D and 3D.

The interface consists in more than 250 different functions, which can be used to draw complex tridimensional scenes with simple primitives. It consists of many functions that help to create a real world object and a particular existence for an object can be given.

**2.2 Characteristics**

* OpenGL is a better documented API.
* OpenGL is also a cleaner API and much easier to learn and program.
* OpenGL has the best demonstrated 3D performance for any API.
* Microsoft's Direct3D group is already planning a major API change called Direct Primitive that will leave any existing investment in learning Direct3D immediate mode largely obsolete.

**2.3 Computer Graphics Library Organization**

OpenGL stands for Open Source Graphics Library. Graphics Library is a collection of APIs (Application Programming Interfaces).

Graphics Library functions are divided in three libraries. They are as follows-

1. GL Library (OpenGL in Windows)
2. GLU (OpenGL Utility Library)
3. GLUT ( OpenGL Utility Toolkit)

Functions in main GL library name function names that begin with the letter ‘gl’.

* GLU library uses only GL functions but contains code for creating objects and simplify viewing.
* To interface with the window system and to get input from external devices GLUT library is used, which is a combination of three libraries GLX for X windows, ‘wgl’ for Windows and ‘agl’ for Macintosh.
* These libraries are included in the application program using preprocessor directives. E.g.: #include<GL/glut.h>
* The following figure shows the library organization in OpenGL.

Open GL application

program

GLU

GL

GLX

GLUT

Xlib, Xtk

Frame

Buffer

**Fig 2.1 Library Organization**

**2.4 Graphics System and Functions**

* Graphics system and functions can be considered as a black box, a term used to denote a system whose properties are only described by its inputs and output without knowing the internal working.
* Inputs to graphics system are functions calls from application program, measurements from input devices such as mouse and keyboard.
* Outputs are primarily the graphics sent to output devices.

Application

Program

Graphics

System

Input Output Devices

Function calls

Data

Output

Input

**Fig 2.2 Graphics System as a Black Box**

API’s are described through functions in its library. These functions are divided into seven major groups.

1. Primitive Functions:

Primitive functions define the low level objects or atomic entities that a system can display, the primitives include line segments, polygons, pixels, points, text and various types of curves and surfaces.

1. Attribute Functions:

Attribute Functions allow us to perform operations ranging from choosing the color to display a line segment, to packing a pattern to fill inside any solid figure.

1. Viewing Functions:

Viewing functions allow us to specify various views.

1. Transformation Functions:

Transformation functions allow us to carry out transformation of objects such as rotation, translation and scaling.

1. Input Functions:

Input functions allow us to deal with the diverse forms of input that characterize modern graphics system. It deals with devices such as keyboard, mouse and data tablets.

1. Control Functions:

Control Functions enable us to communicate with the window system, to initialize the programs, and to deal with any errors that occur during the execution of the program.

1. Query Functions:

Query Functions provides information about the API.

**Chapter 3**

**SYSTEM REQUIREMENTS**

Requirements analysis is critical for project development. [Requirements](http://en.wikipedia.org/wiki/Requirement) must be documented, actionable, measurable, testable and defined to a level of detail sufficient for system design. Requirements can be  [architectural](http://en.wikipedia.org/wiki/System_architecture),  [structural](http://en.wikipedia.org/wiki/Structure),  [behavioural](http://en.wikipedia.org/wiki/Behavior),  [functional](http://en.wikipedia.org/wiki/Functional_requirements), and  [non-functional](http://en.wikipedia.org/wiki/Non-functional_requirements). A software requirements specification (SRS) is a comprehensive description of the intended purpose and the environment for software under development.

* 1. **Hardware Requirement**
* Minimum of 2GB of main memory
* Minimum of 3GB of storage
* Keyboard
* Mouse
* Display Unit
* Dual-Core or AMD with minimum of 1.5GHz speed

**3.2 Software Requirement**

* Windows – XP/7/8
* Microsoft Visual Studio C/C++ 7.0 and above versions
* OpenGL Files
* DirectX 8.0 and above versions

**Header Files**

* glut.h

**Object File Libraries**

* glut32.lib

**DLL files**

* glut32.dll

**Chapter 4**

**ANALYSIS AND DESIGN**

**4.1 Description**

Design is the planning that lays the basics for the making of every object or system. The chapter involves the designing of the various aspects and different stages of the project. When the program is made to execute, the output window is displayed first.

In the first display, initially the fisherman will be seated on the boat and will be moving in the running water. When he reaches display11 he doesn’t find any fishes, display2 starts where he finds the fishes, the fisherman then stops and applies the net to catch the fishes. In the display3 the fisherman is aheading near the market along with the net and the fishes. He reaches the market along with the fishes in display4.The boat could be initiated, stopped and made to move at a faster pace while the boat is in motion using the keyboard keys and the mouse button. In addition to this an option is provided to go back in the reverse direction in the third display. Thereby the fisherman reaches his destination as per the intention of our project. In order to provide user interaction, which will enable the user to interact with the scenario, is done using the standard keyboard.

**4.1.1 User Input:**

**Control**

* Press ‘f’ for moving at a faster pace.
* Press ‘q’ to quit.
* Press‘s’ to stop the boat.
* Press‘t’ to start the boat.
* Press ‘n’ to apply the net.
* Press ‘r’ to go in reverse direction.
* Press ‘c’ to move the fishes.

**Mouse Input**

* Left mouse button to quit.

**4.2 Flow Diagram**

THE FISHER MAN

USER INTERACTION

KEYBOARD FUNCTION

MOUSE FUNCTION

LEFT CLICK

ALPHABETS

QUIT

KEY ‘q’ TO QUIT

KEY ‘r’ TO GO IN REVERSE BOAT

KEY‘t’ TO START THE BOAT

KEY‘s’ TO STOP THE BOAT

KEY ‘n’ TO APPLY THE NET

KEY ‘f’ TO SPEED UP THE BOAT

**Fig 4.1 Design of the project**

The above figure shows the design of the project, where the user interaction, i.e., keyboard and the mouse functions has been provided. Under the keyboard function, keys ‘f’,’n’,’s’,’t’ , ’r’ and ’q’ are used to increase the speed of the boat, to apply the net, to stop the boat, to start the boat, to go in reverse direction and to quit respectively. Under mouse function, left click would do the job of quitting from the program again.

**Chapter 5**

**IMPLEMENTATION**

This chapter includes a flowchart that depicts the flow of control of the project. The purpose of various built-in functions and user-defined functis are also described.

**5.1 source code**

The execution of the program starts from the main function. It calls various inbuilt and user defined functions, Pseudo code is as follows:

void main(int argc , char \*\*argv)

{

Initialize GLUT using glutInit();

Initialize the display mode to DOUBLE or RGB;

Set the window position;

Set the window size;

Create a window;

Call myinit();

Call display();

Call keyboard();

Call mouse();

Call timerfunc();

Start execution from glutMainLoop();

}

Void myinit()

{

Clear the display to white color using glClearColor();

Set the matrix mode;

Load the identity matrix;

}

Void display()

{

Clear the color buffers;

Call the drawm() user defined function;

Call swapbuffers;

}

Void drawm()

{

Based on different conditions, we call the various display functions;

}

Void mouse()

{

Upon the click of the left mouse button, the program terminates.

}

Void drawr()

{

This function makes the fisherman move in reverse direction;

}

Void market()

{

POLYGON functions is called accordingly to construct the market;

}

Void fish1()

{

POLYGON function is called to draw the shape of purple fish;

LINE functions is called accordingly to draw the body of purple fish;

}

Void fish2()

{

POLYGON function is called to draw the shape of golden fish;

LINE functions is called accordingly to draw the body of golden fish;

}

**5.2 Flowchart**

Flowchart is a pictorial representation of the program. The figure shows the different functions used in the project.

Start

# Initialization

# MainLoop

Display()

If(flag==**1)**

Drawm()

T

F

KeyboardFunc()

MouseFunc()

Myinit()

Stop

# Display1()

Y

# Display11()

# If(a>950)

N Y

# If(k>960)

# Display2()

N

Y

# If(b>970)

# Display3()

N

# If(c>=290)

Y

Display4()

**N**

# If(flag1==1)

# Drawr()

Exit (0)

Key ‘f’

# display()

# drawr()

Key ‘r’

# snet()

Key ‘n’

Flag=0

Key‘s’

Key‘t’

Flag=1

Exit (0)

Key ‘q’

Exit (0)

Left Click

**Fig 5.1: flowchart**

The above flowchart depicts the flow control of the program, as to how it works. Starting with the main, it enters the display function wherein if the flag is set to 1, it calls the drawm() which in turn calls the display1 where depending on various conditions, the functions display11, display 2, display3, display4 else keyboard keys ‘f’ used to increase the speed of the boat calls the display(), ‘n’ calls snet(). Key ‘s’ sets flag to zero which is used to stop the boat , key ‘r’ is used to move the boat in reverse direction ,key ‘t’ sets flag to one and is used to start the boat and key ‘q’ to quit. Under the mouse function the left of the mouse is used to quit from the program.

**5.3 Functions**

The functions that are used in the program are discussed below. There are two types of functions. They are in-built and user defined functions. These functions are as follows:

**5.3.1 In-built functions**

The in –built functions are defined in OpenGL library. Some of the in-built functions which are used in the project are explained below:

* **glPushMatrix( ) ;**

Push the current matrix onto the stack. PushMatrix 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 the current matrix on to the stack. PushMatrix pushes the current matrix stack down by one, duplicating the current matrix. This is, after a glPushMatrix call, the matrix, on top of the stack is identical to the one below it.

* **glTranslatef ( x, y, z);**

Multiplies the current matrix by the translation matrix. glTranslate produces a translation by (x,y,z), where x ,y and z are the parameters. Translation is an operation that displaces points by a fixed distance in a given direction.

* **glMatrixMode(GL\_PROJECTION);**

**glMatrixMode(GL\_MODELVIEW);**

The two important matrices are modelview and projection.

* **glLoadIdentity( );**

Used to load the identity matrix.

* **glFlush();**

By this call, it ensures that the points are rendered to the screen as soon as possible.

* **glClear(COLOR\_BUFFER\_BIT);**

It is used to clear the window on the screen to white or black color.

* **glutInit( )**

Initialize the GLUT library and the graphics system.

* **glutInitDisplayMode( )**

Set the initial display mode.

* **glutMainLoop( )**

Enter the event processing loop.

* **glutCreateWindow ()**

Creates a top level window ().

* **glutInitWindowPosition( )**

Set the window position to the given specification.

**5.3.2 User-defined functions:**

* **scalefish()**

This function is used to create a fish and to manipulate it.

* **scaleman()**

This function is used to create a man and manipulate it.

* **water()**

This function is used to create water.

* **boat()**

This function is used draw the boat on water.

* **bank()**

This function is used to draw the bank on the river side.

* **snet()**

This function is used to apply net on the fishes**.**

* **sky()**

This function is used to create sky.

* **display()**

This function is used to display the various functions present within it.

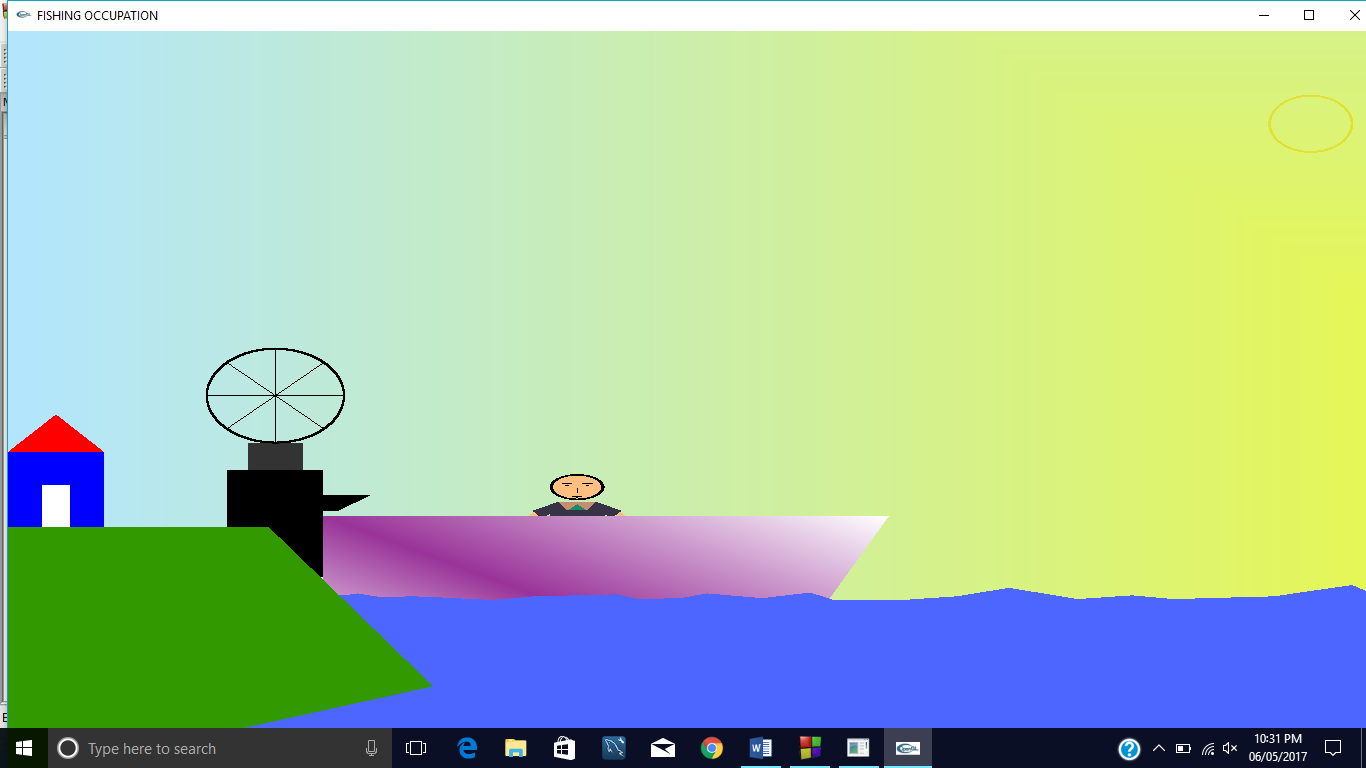
* **drawmarket()**

This function is used to draw market on the bank of the river.

**Chapter 6**

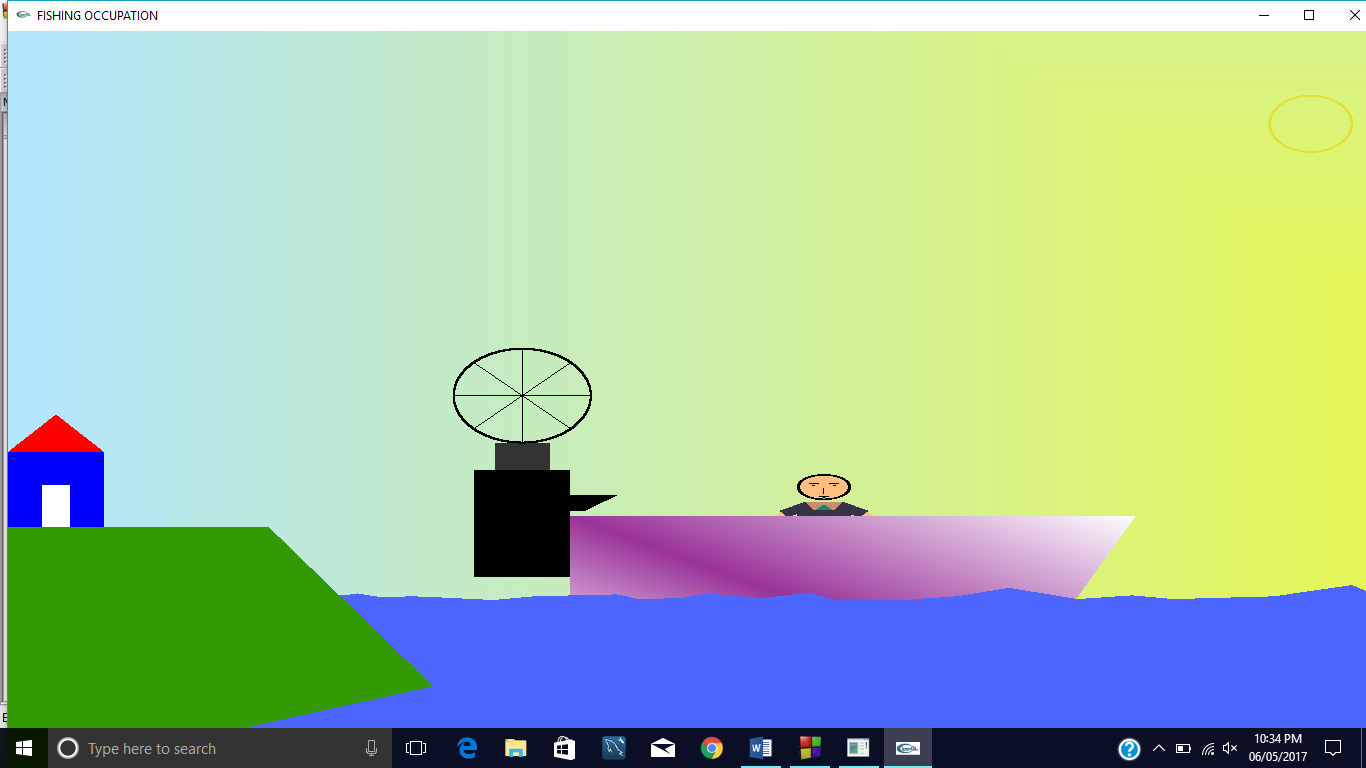
**SNAPSHOT**

**Initial Display screen**



**Fig 6.1: The above output shows the initial display screen wherein the fisherman seated on the boat is yet to start moving from the bank.**

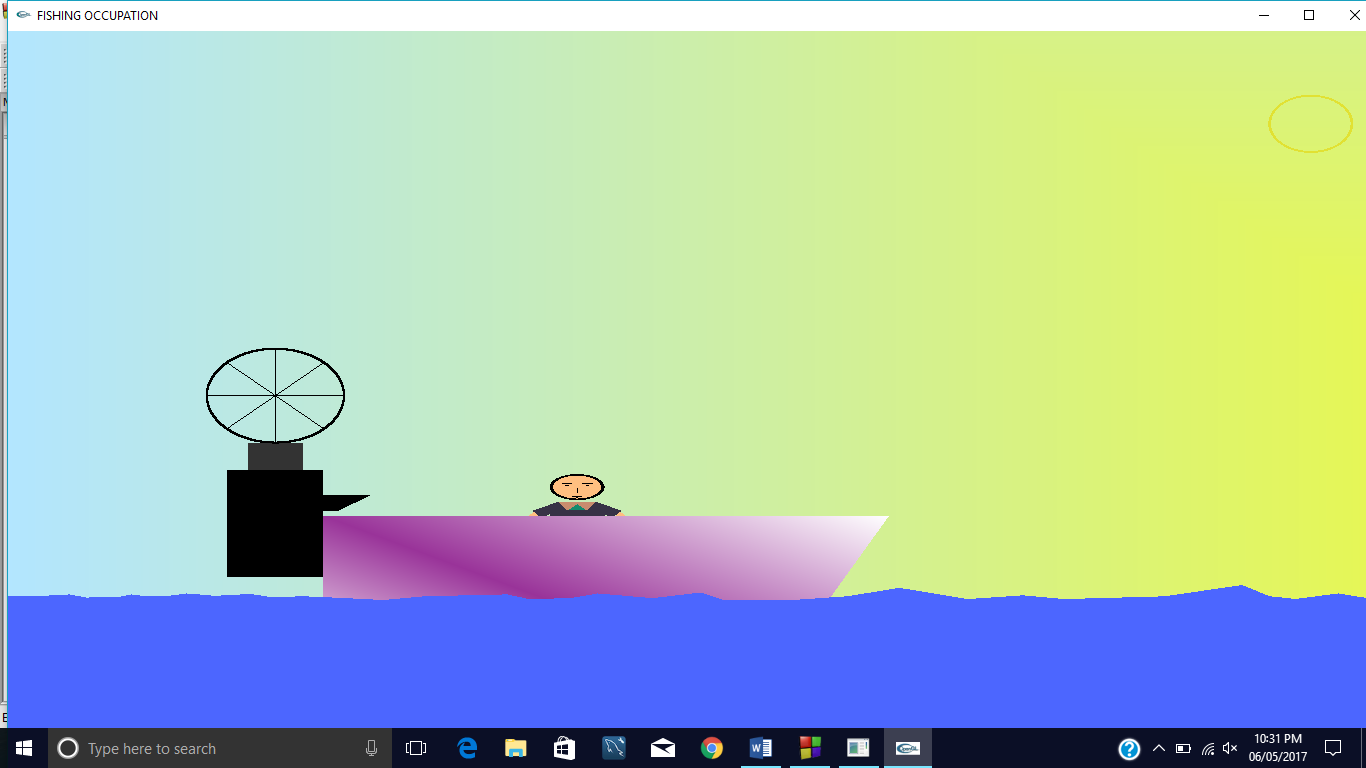
**Snapshot of the man moving forward**



**Fig 6.2:** **The above output shows the boat moving with the fisherman in the display1.**

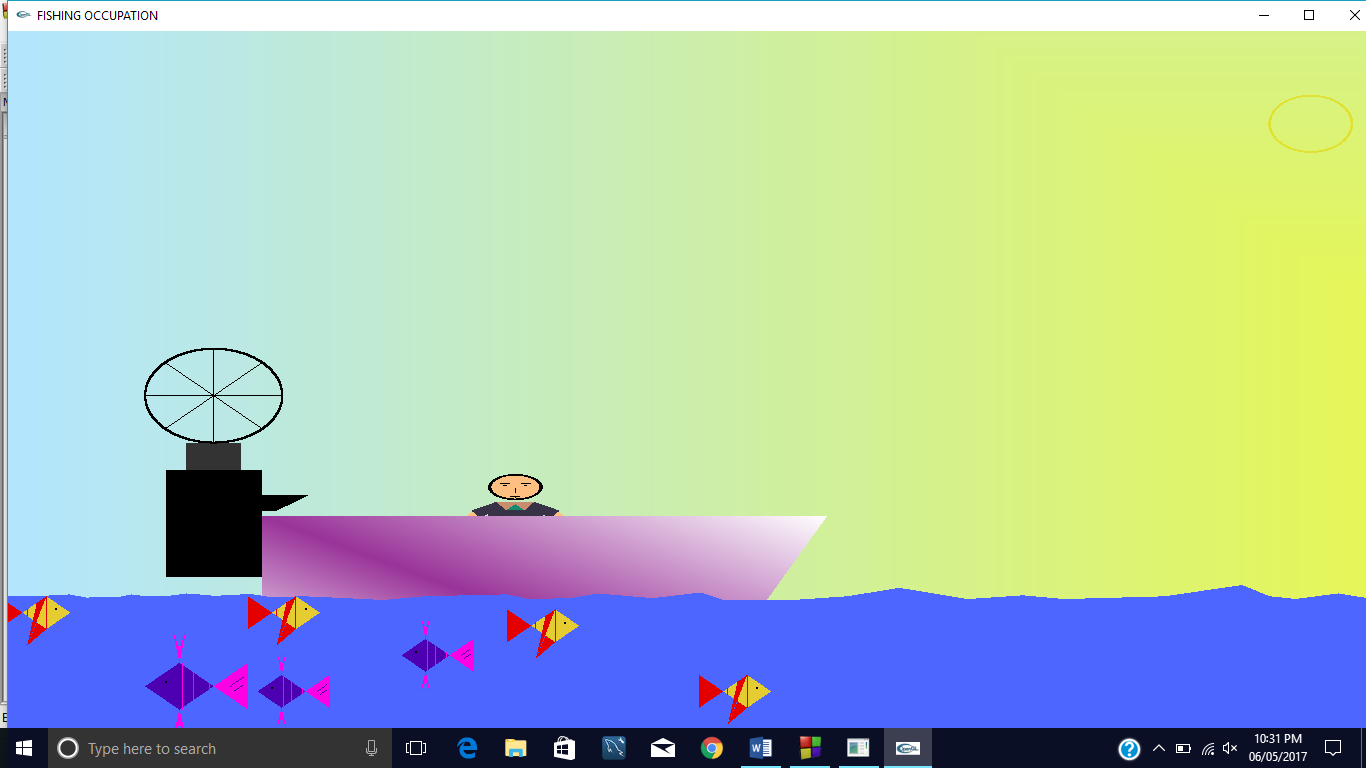
**Key ‘t’ is used for starting the boat and key‘s’ for stopping the boat.**

**Snapshot when he doesn’t find any fishes**



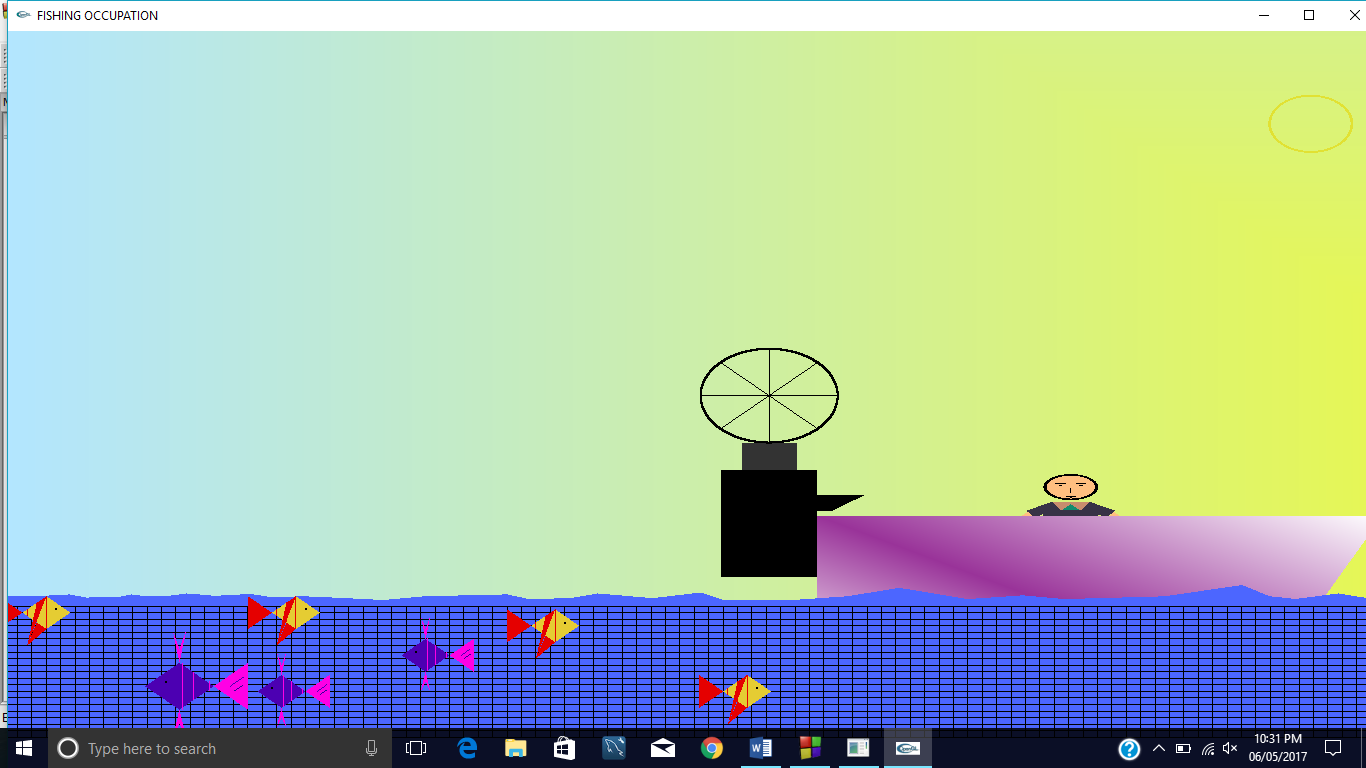
**Fig 6.3: The above output shows that the fisherman doesn’t find any fishes. Key ’s’ is used to stop the boat .Key ‘f’ is used to increase the speed of the boat.**

**Snapshot in which the man finds the fishes**



**Fig 6.4: The output shows the boat moving in the forward direction and the presence of fishes.**

**Snapshot in which the man is moving along with the net and fishes**

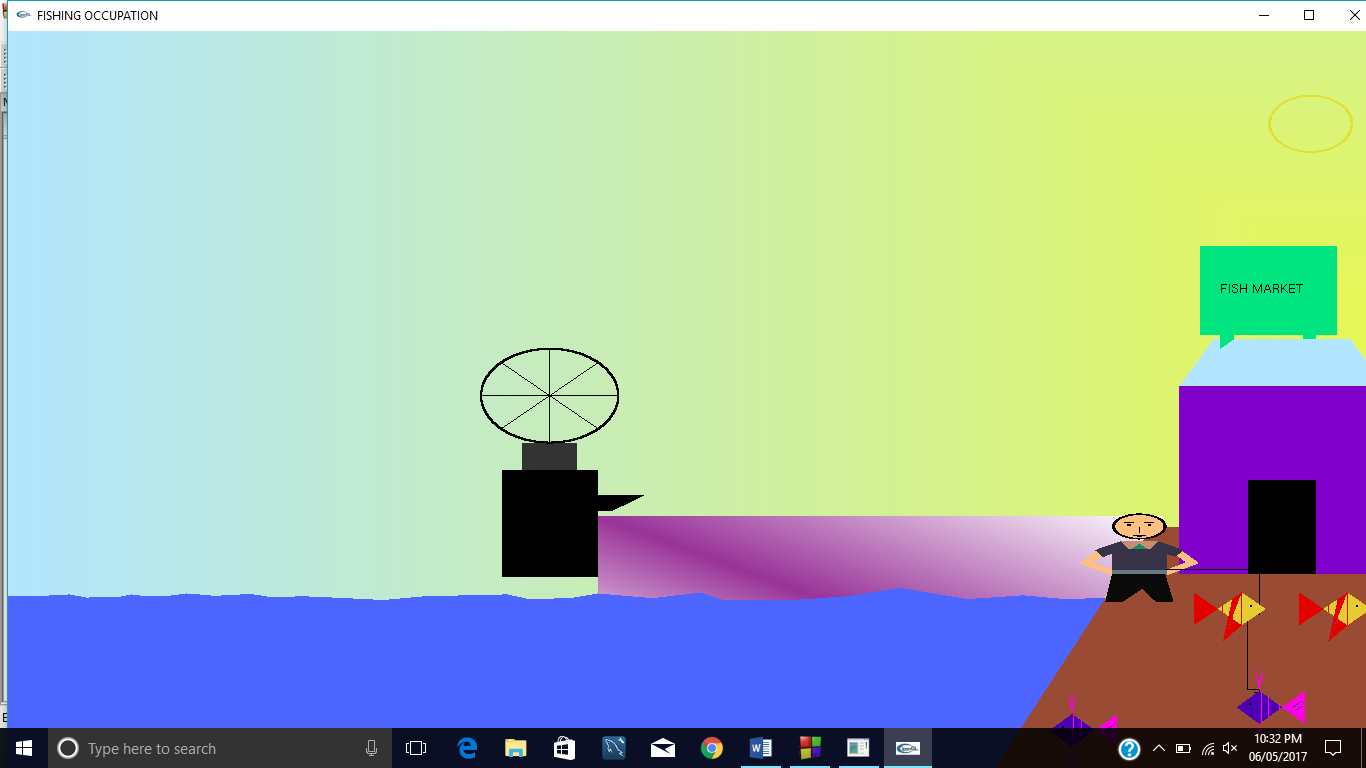


**Fig 6.5: The above output shows the fisherman is moving along with the net to the market .Key ‘t’ is used to start the boat and move forward. Snapshot in which the man applies the net** 

**Fig 6.6: The above output shows the fisherman applying net on the fishes after finding them .**

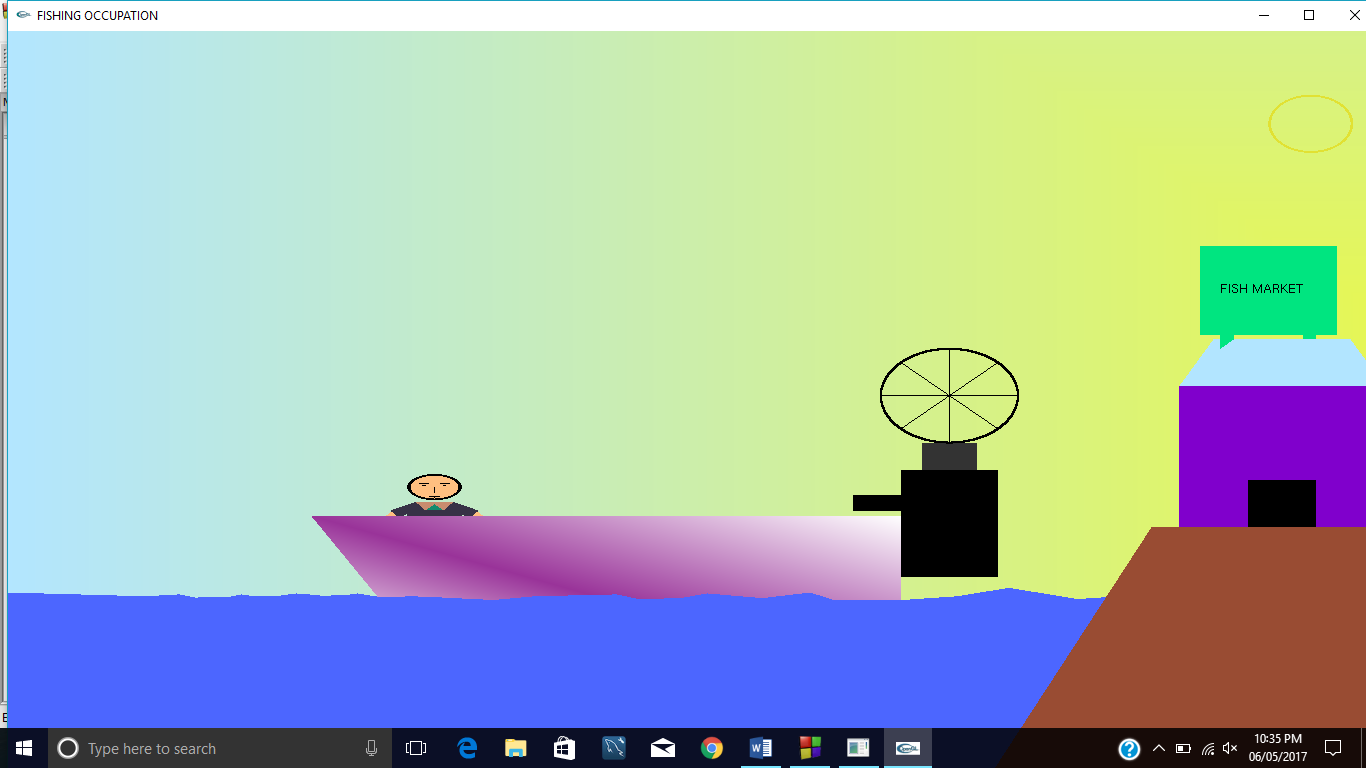
**Key ‘s’ is used to stop the boat and then apply the net.**

**Snapshot when the man reaches bank with fishes in front of the market**



**Fig 6.7: The above output shows that the man has reached the destination along with the fishes .He has reached the market in display4.**

**Snapshot of the man moving in reverse direction**



**Fig 6.8: The above output shows that the boat can move in reverse direction. Key ‘r’ is used to move the boat in reverse direction.**

**Chapter 7**

**CONCLUSION**

The mini project “**FISHING OCCUPATION**” has been efficiently developed with OpenGL. This is indeed an interactive project which has efficient interactions given through the keyboard as well as the mouse. The project shows the fisherman seated on the boat moving forward in search of fishes. He applies the net on finding them and proceeds forward to another bank reaching the market which is present on the other side of the bank. The boat could be started, stopped and made to move at a faster pace while the boat is in motion using the keyboard keys and the mouse button to quit along with an option to move in reverse direction.

**REFERENCE**

[1] Computer Graphics principles &practice - Foley, VanDam, Feiner, Huges

[2] Fundamentals of Computer Graphics -Peter Shirley

[3] Intet6rractive Computer Graphics-Edward Angel

**Sites Referred**

[1] [www.sgtackoverflow.com](http://www.sgtackoverflow.com)

[2] [en.wikipedia.org/wiki/OpenGL](file:///F:\Downloads\en.wikipedia.org\wiki\OpenGL)

[3] [www.opengl.com](http://www.opengl.com)