**CHAPTER – 1**

**INTRODUCTION**

The term computer graphics has been used in a broad sense to describe almost everything on computers that is not text or sound. Typically, the term computer graphics refers to several different things: the representation and manipulation of image data by a computer, the various technologies used to create and manipulate images, the images so produced, the sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content see study of computer graphics.

Today, computers and computer-generated images touch many aspects of daily life. Computer imagery is found on television, in newspapers, for example in weather reports, or in all kinds of medical investigation 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, thesis, and other presentation material.3D computer graphics uses a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering 2D images. Such images may be for later display or for real time viewing.

Despite these differences, 3D computer graphics rely on many of the same algorithms as 2D computer vector graphics in the wire frame model and 2D computer raster graphics in the final rendered display.

In computer graphics software, the distinction between 2D and 3D is occasionally blurred; 2D applications may use 3D techniques to achieve effects such as lighting, and primarily 3D may use 2D rendering techniques.

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry’s most widely used and supported 2D and 3D graphics Application Programming Interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions.

**1.1 INTRODUCTION TO OPENGL**

The field of computer graphics continues rapidly growing with an ever-increasing number of applications of diverse areas, such as entertainment, business, art, education, medicine, engineering, and industry. A Number of software packages have emerged to help generate and manipulate two-dimensional (2D)/three-dimensional (3D) graphics.

OpenGL is a graphic application programming interface (API) for the C/C++ programming language. The primary motivation for developing OpenGL is to create an operating system, window system, and hardware platform independent API for the development of 2D /3D graphics. Since OpenGL API was introduced in 1992, many applications, such as CAD, CAM and game development, have benefited from its cross-platform accessibility. OpenGL has become a premier environment for developing portable 2D/3D graphics applications.

It is also widely used for teaching and learning computer graphics. The feature of device independence and portability make OpenGL a strategic interface for courses on computer graphics .Computer platform vary from instructor to student and from school to student home.

By using OpenGL, programs developed on a machine can be debugged and graded on other machines with different platforms , and the resulting graphics are the same .

OpenGL is the most widely used application programming interface(API) for developing portable 2D and 3D graphics applications. It was originally developed in 1992 by Silicon Graphics and is currently controlled and maintained by the OpenGL ARB (Architecture Review Board).

According to OpenGL.org, OpenGL is the only truly open, vendor-neutral, multi platform graphics standard. OpenGL 2.1 is the latest version of OpenGL and was released on August 2 2006. OpenGL 3.0 is currently under development and will run specifically on hardware born after November 8th, 2006. GLUT supports the following functionality:

* Window management
* Even handling
* Right click menu
* Rendering fonts

**1.2 OBJECTIVE**

The objective of the Aqua Mania is to stimulate aquarium view which consists of fishes, see weeds and many more objects with computer graphics. The graphic package is based on OpenGL libraries. In this project we are demonstrating the movement of fish, whise direction can be controlled using various keys.

**CHAPTER – 2**

**S/W AND H/W REQUIREMENTS**

The Prerequisites for developing a system is known are known as System requirements and they are used as a guideline for further development of the system. This chapter explains the requirements of Software and Hardware components used in the project.

**2.1 S/W REQUIREMENT:**

Operating systems : Windows version 10

Package : Dev C++

Languages : C

**2.2 H/W REQUIRMENTS:**

Visual Display Unit : 1024x768 resolution or higher

Graphics Driver : onboard graphics

Processor : Intel Pentium Processor or higher

Processor Speed : 500 MHz or above

RAM : 128 MB or higher

Storage Space : Approximate 2 MB

**CHAPTER – 3**

**DESIGN**

Project design is a crucial stage in a project’s life cycle because it identifies key elements and sets the overall tone. For your project to be successful, you must first understand the steps involved in project design, as well as how to document them.

Creating a project design can help you avoid pitfalls down the road and also set a reasonable budget from the outset. Software development projects can be highly technical and complex. These project designs have a broad range: from simple diagrams or descriptions of requirements and functions to long, detailed documents outlining every technical specification of the application or system.

In this project our main goal is to take number of disks from the user to show how the concept of tower of Hanoi works. In this project we show how the disks replace from the first tower to the 3rd tower without placing the larger disks over the smaller disks.

Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data.

**CHAPTER 4**

**IMPLEMENTATION**

System implementation is the process of defining how the information system should be built, ensuring that the information system is operational and used.

In this section, the various OpenGL and user defined functions regarding the “AQUA MANIA” is explained.

The most commonly used functions of the graphics library in our project are:

* Call back functions
* GL functions
* GLUT functions

**4.1 CALL BACK FUNCTIONS:**

A callback function is a function which the library (GLUT) calls when it needs to know how to process something.

* **void ():**

This function is passed to glutDisplayFunc() whenever the contents of the window need to redrawn.

* **void ():**

This function is passed to the glutVisibilityFunc() and sets the visibility callback for the current window. The visibility callback for a window is called when the visibility of a window changes.

**4.2 GL FUNCTIONS:**

GL(open graphics Library) functions that are used for creating 2D/3D objects and screens.

* **glPushMatrix():**

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():**

glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack.

* **glBegin(), glEnd():**

This function delimits the vertices or a primitive or a group of liking primitives.

* **glRotatef():**

This produces a rotation of angles, degrees around the vector(x, y, z).The current matrix is multiplied by a rotation with the product replacing the current matrix.

* **glTranslatef():**

This displaces a point to new positions defined by a displacement vector.

**4.3 GLUT FUNCTIONS:**

GLUT(openGL Utility Toolkit) is a Library of utilities for OpenGL, which primarily focuses on window definition, window control and monitoring of keyboard and mouse input.

* **glutCreateWindow():**

This 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 able the window with the name.

* **glutDisplayFunc():**

Graphics are sent to the screen through a function called the display call back and register with window system. Here the function name will be called whenever the windowing system determines that the OpenGL window to be redisplayed.

* **glutMainLoop():**

This enters the GLUT event processing loop. This routine should be called most once in a GLUT program. Once called, this routine will never return.

* **glutSwapBuffers():**

This performs a buffer swap on the layer in use for the current window. This promotes the contents of the back buffer of the layer in use of the current window to become the contents of the front buffer.

* **glutInitWindowSize(int width,int size):**

Create the size, in pixels of your window.

* **glutDisplayFunc():**

It is used to invoke all the call back display.

* **glutIdleFunc():**

It sets the global idle call back to be function so a GLUT program can perform background processing tasks or continuous animation when window system events are not being received.

* **glutVisibilityFunc():**

It sets the visibility call back for the current window. The visibility call back for a window is called when the visibility of a window changes.

* **glutInitDisplayMode():**

It sets the initial display mode. It can be GLUT\_RGB, GLUT\_SINGLE, GLUT\_DOUBLE etc.

* **glutPostRedisplay():**

It is used to mark the current window as needing to be redisplayed.

* 1. **USER DEFINED FUNCTIONS USED IN THE PROJECT:**

**init main(int argc, char \*\*argv) :**

The main fu**ar**nction is called at program start up after initialization of the non-local objects with static storage duration which is the designated start of the program.

**Argc:** non-negative value representing the number of arguments passed to the program.

**Argv:** pointer to the first element of the array of pointers.

* **glutInit(&gc, argv) :**

Initializes GLUT and processes any command line arguments.

* **glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE | GLUT\_DEPTH)**
* **User Defined Functions to Create Objects**

Fish()

Grass()

Stones()

Sfish()

Bgcolor()

**4.5 CODE OF AQUA MANIA:**

**Fish:**

void fish()

{

glPushMatrix();

glPushMatrix();

glRotatef(angle,0.0,0.0,0.0);

glTranslatef(Autorun,0.0,0.0);

glTranslatef(xt,yt,0.0); //For Move

glBegin(GL\_POLYGON); // draw body

glColor3f(1.000,0.0,0.498);

glVertex2i(20-5,200-5);

glColor3f(0.298,0.000,0.600);

glVertex2i(70-5,180-5);

glColor3f(1.000,0.0,0.498);

glVertex2i(80-5,205-5);

glColor3f(0.298,0.000,0.600);

glVertex2i(65-5,230-5);

glEnd();

glPushMatrix();

glRotatef(angle,0.0,0.0,0.0);

glColor3f(1,1,1);

glBegin(GL\_POLYGON);

for(i=0;i<2\*3.14;i+=0.5)

glVertex2f(62+2\*cos(i),205+2\*sin(i));

glEnd();

glColor3f(0,0,0);

glBegin(GL\_POLYGON);

for(i=0;i<2\*3.14;i+=0.5)

glVertex2f(62+0.8\*cos(i),205+0.8\*sin(i));

glEnd();

glBegin(GL\_POLYGON); //draw tail

glColor3f(0.298,0.000,0.600);

glVertex2i(15-5,200-5);

glColor3f(1.000,0.0,0.498);

glVertex2i(10-5,180-5);

glColor3f(0.298,0.000,0.600);

glVertex2i(20-5,200-5);

glColor3f(0.298,0.000,0.600);

glVertex2i(10-5,220-5);

glEnd();

glBegin(GL\_POLYGON); //draw Top Key

glColor3f(1.0,0.0,0.0);

glVertex2i(27-5,204-5);

glVertex2i(65-5,230-5);

glVertex2i(60-5,240-5);

glEnd();

glBegin(GL\_POLYGON); //draw Buttom Key

glVertex2i(65-5,182-5);

glVertex2i(70-5,180-5);

glVertex2i(60-5,200-5);

glEnd();

glPopMatrix();

glPopMatrix();

glPopMatrix(); }

**Stones:**

glBegin(GL\_POLYGON); //stone

glColor3f(0.278,0.125,0.003);

glVertex3f(210.0, 0.0, 0);

glColor3f(0.717,0.352,0.054);

glVertex3f(215.0, 10.0, 0);

glColor3f(0.717,0.352,0.054);

glVertex3f(230.0, 15.0, 0);

glColor3f(0.278,0.125,0.003);

glVertex3f(245.0, 8.0, 0.0);

glColor3f(0.200,0.094,0.000);

glVertex3f(255.0, 0.0, 0);

glEnd();

**Grass:**

glColor3f(0,0.8,0); //grass1

glBegin(GL\_POLYGON);

glColor3f(0.074,0.349,0.011);

glVertex2f(10+10,0);

glColor3f(0.074,0.349,0.011);

glVertex2f(0+10,20);

glColor3f(0.074,0.349,0.011);

glVertex2f(20+10,10);

glColor3f(0.380,0.878,0.094);

glVertex2f(20+10,50);

glColor3f(0.380,0.878,0.094);

glVertex2f(30+10,10);

glColor3f(0.380,0.878,0.094);

glVertex2f(40+10,20);

glVertex2f(40,0);

glEnd();

**CHAPTER 5**

**CONCLUSION**

We have provided two approaches to resolve the problem i.e. the Manual approach and the automated approach. An Automated approach guides the user in a right direction to solve it. In manual approach, User interaction was provided in all the levels that user wants to play. In automated approach, the solution for all levels was provided thereby increasing interactivity and performance of this project. This project includes graphical user interface for better interactive learning and understanding of the problem. Thus it increases the usability of the project.

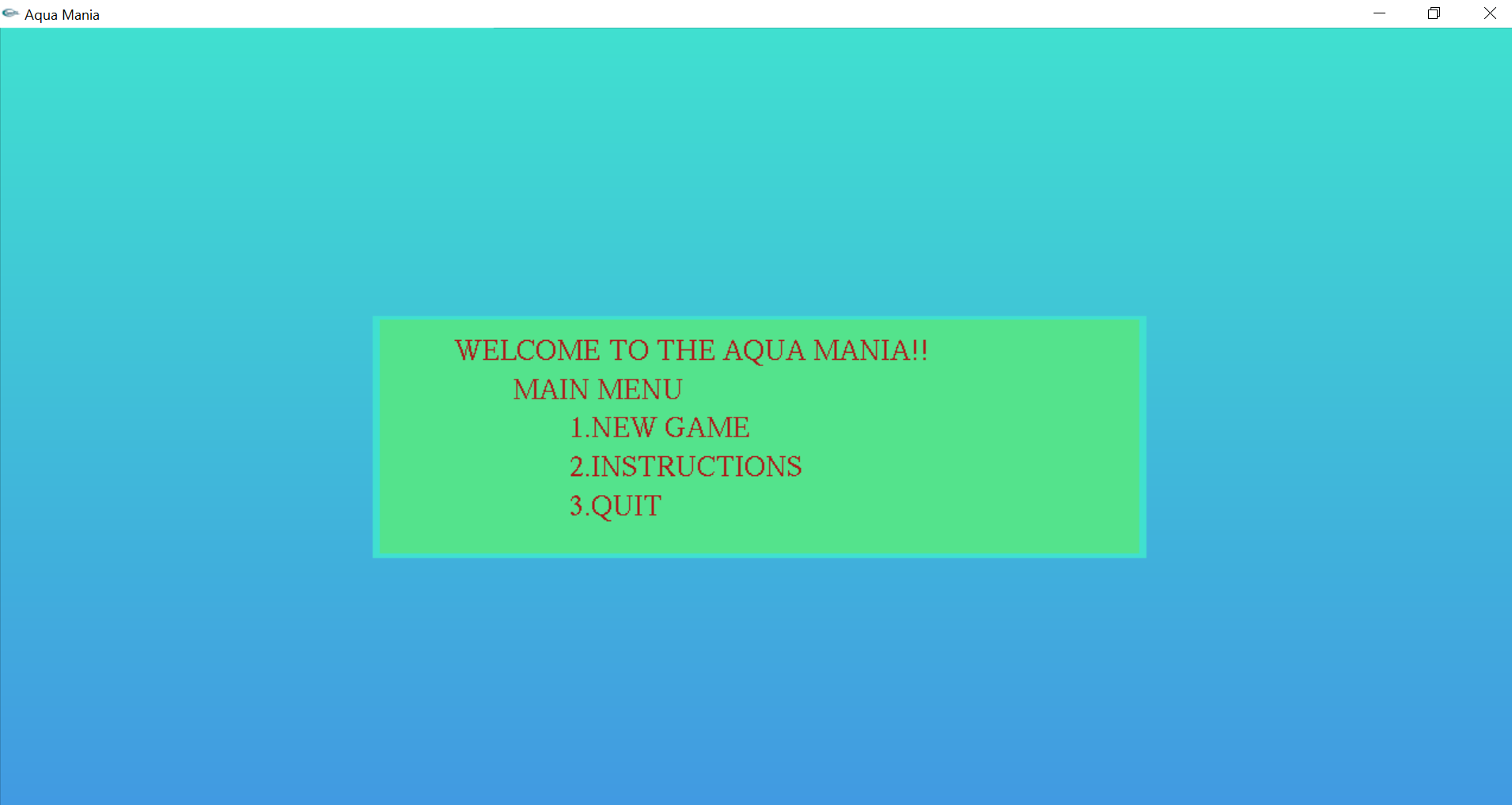
**APPENDIX-A**

**SNAPSHOTS**

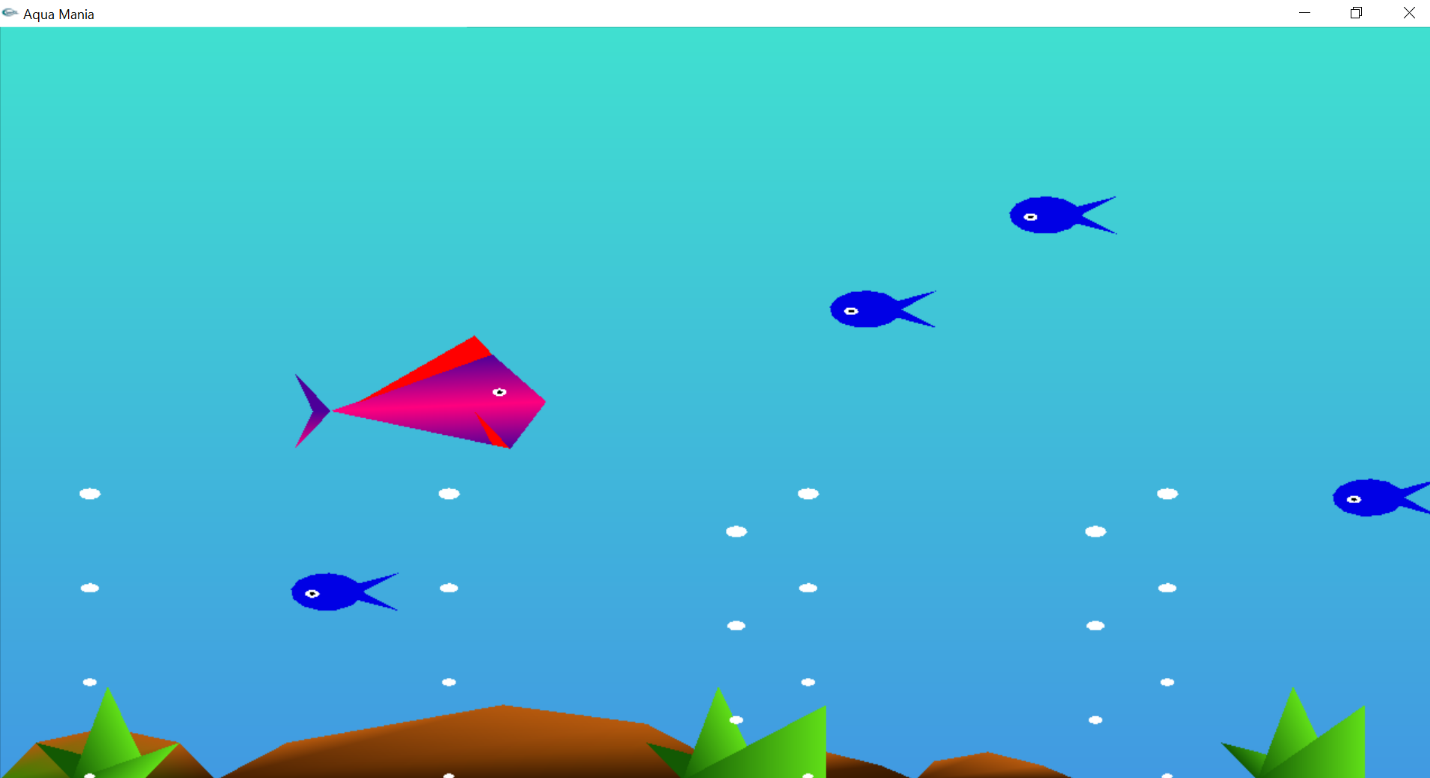
**Front page**



**Main menu**

****

**Game page**

****

**APPENDIX-B**

**REFERENCES**

[1] Edward Angel, “Interactive Computer Graphics A TOP-DOWN Approach with OpenGL”, 2nd edition, Addition-Wesley, 2000

[2] <http://www.opengl.org>

[3] James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer graphics with OpenGL: pearson education