# CHAPTER 1

# INTRODUCTION

## 1.1 Introduction to project

Our mini project is aimed towards developing a system on **"SAILING BOAT"**. Hence, we have made an attempt in developing it. Our mini project code is developed in C/C++ language by using OpenGL.

The basic design of this mini project contains a stimulation of boat. This project is an attempt to show the different graphical objects. And it mainly consists of a boat, tree and a small house.

## 1.2 OpenGL concepts

Computer graphics is one of the most effective and most commonly used means of communication with the user. It displays the information in the form of graphics objects such as pictures, charts, graphs and diagrams instead of simple text. The pictures or graphics objects may vary from engineering drawings, business graphs and architectural structures to animated movies. All the functionalities required for the development and presentation of such an environment or interface to the user is provided by the graphics package.

There is numerous number of ways in which computer graphics has made user interaction fast, effective and fun. Graphics has enabled the designers to introduce the concept of windows that act as virtual graphics terminals, each of which is capable of running an independent application. The introduction of the mouse has made the selection of objects on the interface easy by the "Point and Click" facility and a lot more.

## 1.3 Scope and application

We took over this project as this will help us to understand various transformations which could be done in computer graphics. This also gives us an idea about glut libraries used in this project.

Some of the applications are:

- Virtual reality.
- To show that implementation of Translation is easier with OpenGL.
- Implementing certain technical concept like Translation and use of Idle Function.
- How to use Animation effects used to produce computer animation.

# CHAPTER-2

## LITERATURE SURVEY

- Literature survey is the most important step in the software development process. Before developing tools, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, the next step is to determine which operating system and language can be used for developing the tool. Once the programmer start building the tool programmer needs lot of external support.

- We got to know about the various graphic functions which can be implemented using open GL like the functions GLUT(Graphics Library Utility Toolkit), GL(Graphics Library) and GLU(Graphics Library Utility),GLUT is a function which provide us with toolkit in using the open GL functions in different operating systems preferably windows(windows XP and windows 7). Basic functions that are required like clearing the screen, vertices, forming points and constructing lines is facilitated by GL functions. It helps us in creating generation of texture bit maps, drawing of quadratic surfaces or polygon primitives.

- Preferring another book to get a clear and detailed explanation of various open GL function was important in order to get clear understanding of the project. Moreover we got a clear view of how to implement 2D views and they work as our project mostly based on 2Dviews. And of course we implemented lot of texture, modeling and transformation about which we got more details.

- In order to use advanced buffers and buffers usage, we had to refer this book and also we have to take its help to implement the geometrical graphics which we have used in our project. Here we got to know the uses of various API references and importantly how to use them. There were various codes used in daily graphics programs which helped us a lot.

# CHAPTER-3

## REQUIREMENTS AND SPECIFICATIONS

The hardware and software requirements are very minimal and the software can run on most of the machine even of the past. Here we have used the system of the below specification of develop.

## 3.1 Hardware Requirements

- PROCESSOR : Pentium processor or AMD Athlon or higher CPU
- RAM : 256MB
- HARD DISK SPACE : 20GB
- MOUSE AND KEYBOARD

### Software Requirements

SOFTWARE : Microsoft Visual Studio 2010/15/17

OPERATING SYSTEM : Windows XP /Windows Vista / Windows 7/Windows 10

## 3.2 FUNCTIONAL REQUIREMENTS

The developed project also displays the statements of services the animation should provide, how the system should react to particular inputs and how the system should behave in particular situations.

## 3.3 NON-FUNCTIONAL REQUIREMENTS

The developed project also displays the constraints on the services or functions offered by the system. They include constraints on the development process, standards, etc.

# CHAPTER 4

## SYSTEM ANALYSIS AND DESIGN

### 4.1 Architecture

## HIGH LEVEL DESIGN: -

- MAKE GEOMETRY: In this we are creating required geometry for proper output, which includes geometry for triangle, circle, rings and squares.

- CREATE ENVIRONMENT: This is where global settings are made, i.e.- things that will not change in time, also we are setting a background colour of a shade of grey representing the walls of the room of our project output platform, and main role of this function is to provide a supporting environment.

- HANDLE KEYBOARD: Here we are dealing with plain key strokes to include keyboard interface.

- MAIN FUNCTION: This function includes setting up of keyboard interactions and also includes function calls of all function definitions.

- DISPLAY FUNCTION: This is the basic display callback routine. It creates the geometry, lighting, and viewing position. In this case it changes the camera around the scene.

## LOW LEVEL DESIGN

- MAKE GEOMETRY: In this function we make use of co-ordinates for all surfaces of the four different graphical objects.

- HANDLE KEYBOARD: Here we are passing three arguments for key board interaction as ASCII value of input characters as **'up-arrow'** for moving the ball upwards, **'downarrow'** for moving the ball downwards, '**left-arrow'** for moving the ball towards the left and '**right-arrow'** for moving the ball towards the right.

# CHAPTER 5:

## IMPLEMENTATION:

### 5.1 MODULE DESCRIPTION

## void glBegin (glEnum mode) ;

Initiates a new primitive of type mode and starts the collection of vertices. Values of mode include GL_POINTS and GL_LINE.

### void glEnd ( ) ;

Terminates a list of vertices.

## void glColor3f [ i f d ] (TYPE r, TYPE g, TYPE b) ;

Sets the present RGB colors. Valid types are int ( I ), float ( f ) and double (d ). The maximum and minimum values of the floating-point types are 1.0 and 0.0, respectively.

## void glClearColor (GLclampf r, GLclampf g, GLclampf b, GLclampf a);

Sets the present RGBA clear color used when clearing the color buffer. Variables of GLclampf are floating-point numbers between 0.0 and 1.0.

## int glutCreateWindow (char *title) ;

Creates a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used where there are multiple windows.

### void glutInitWindowSize (int width, int height) ;

Specifies the initial height and width of the window in pixels.

### void glutInitWindowPosition (int x, int y) ;

Specifies the initial position of the top-left corner of the window in pixels.

## void glutInitDisplayMode (unsigned int mode) ;

Request a display with the properties in mode. The value of mode is determined by the logical OR of operation including the color model (GLUT_RGB, GLUT_INDEX) and buffering (GLUT_SINGLE, GLUT_DOUBLE).

## void glFlush ( ) ;

Forces and buffers any OpenGL commands to execute.

**void glutInit (int argc, char **argv) ;**

Initializes GLUT. The arguments from main are passed in and can be used by the application.

**void glutMainLoop ( ) ;**

Cause the program to enter an event processing loop. It Appears at the end of main.

**void glutDisplayFunc (void (*func) (void)) ;**

Registers the display function func that is executed when the window needs to be redrawn.

**void glOrtho( GLdouble left, GLdouble right, Gldouble bottom,**

**GLdouble top, GLdouble zNear, GLdouble zFar);**

The glOrtho function multiplies the current matrix by an orthographic matrix.

**glutSwapBuffers( );**

The SwapBuffers function exchanges the front and back buffers if the current pixel format for the window referenced by the specified device context includes a back buffer. **19.**

x, y, z, w

Specify the x, y, z, and w object coordinates (if present) for the raster position.

**glMatrixMode (GLenum mode );**

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. Additionally, if the ARB_imaging extension is supported, GL_COLOR is also accepted.

**GL_MODELVIEW**

Applies subsequent matrix operations to the modelview matrix stack.

**GL_PROJECTION**

Applies subsequent matrix operations to the projection matrix stack.

**glPushMatrix( ); , glPopMatrix( );**

Push and pop the current matrix stack.

**glLoadIdentity( );**

Replace the current matrix with the identity matrix.

**void glPolygonMode ( GLenum face, GLenum mode );**

Select a polygon rasterization mode.

**void display( ) ;**

To display the output.

## 5.2 PSEUDO CODE :

```
#include <windows.h>
#include <stdio.h>
#include <iostream>
#include <GL/glut.h>
#include <math.h>
int boatStatus = 1;
float boatX = 0;
float boatY = 0;
void drawBoat(int);
void tree();
void scene();
float move, angle;
void DrawCircle(float cx, float cy, float r, int num_segments)
{
	glBegin(GL_TRIANGLE_FAN);
	for (int ii = 0; ii < num_segments; ii++)
	{
		float theta = 2.0f * 3.1415926f * float(ii) /
		float(num_segments);
		float x = r * cosf(theta);//calculate the component
		float y = r * sinf(theta);//calculate the component
		glVertex2f(x + cx, y + cy);//output vertex

	}
	glEnd();
}

void tree()
```

```
{
	glPushMatrix();

	glBegin(GL_POLYGON); //Tree
	glColor3f(1.2, 0.5, 0.4);
	glVertex2i(435, 344);
	glVertex2i(450, 345);
	glVertex2i(450, 550);
	glVertex2i(435, 550);

	glEnd();
	glColor3f(0.0, 0.7, 0.1);
	DrawCircle(415, 545, 50, 1000); // 4
	glColor3f(0.0, 0.7, 0.1);
	DrawCircle(430, 600, 50, 1000);
	glColor3f(0.0, 0.7, 0.1);
	DrawCircle(460, 550, 50, 1000);

	glPopMatrix();

}
void scene()
{
	glPushMatrix();

	glBegin(GL_POLYGON); //Sky
	glColor3f(0.4, 0.5, 1.0);
	glVertex2i(0, 800);
	glVertex2i(1200, 800);
	glColor3f(0.7, 0.7, 1.0);
	glVertex2i(1200, 0);
	glVertex2i(0, 0);
	glEnd();

	glBegin(GL_POLYGON); //Middle ground
	glColor3f(0.0, 0.7, 0.0);
	glVertex2i(0, 100);
```

```
glVertex2i(0, 280);
glVertex2i(200, 330);
glVertex2i(400, 350);
glVertex2i(600, 330);
glVertex2i(800, 320);
glVertex2i(1000, 300);
glVertex2i(1200, 270);
glVertex2i(1200, 100);

glEnd();


glBegin(GL_POLYGON); // River
glColor3f(0.2, 0.3, 1.1);
glVertex2i(0, 150);
glVertex2i(200, 150);
glVertex2i(400, 150);
glVertex2i(600, 150);
//glColor3f(0.2, 0.3, 1.1);
glVertex2i(800, 150);
glVertex2i(1000, 150);
glVertex2i(1200, 150);
glVertex2i(1200, 0);
glVertex2i(0, 0);
glEnd();
//tree();


glBegin(GL_POLYGON); //House
glColor3f(0.9, 0.7, 0.1);
glVertex2i(350, 344);
glVertex2i(350, 460);
glVertex2i(430, 460);
glVertex2i(430, 344);
glEnd();
```

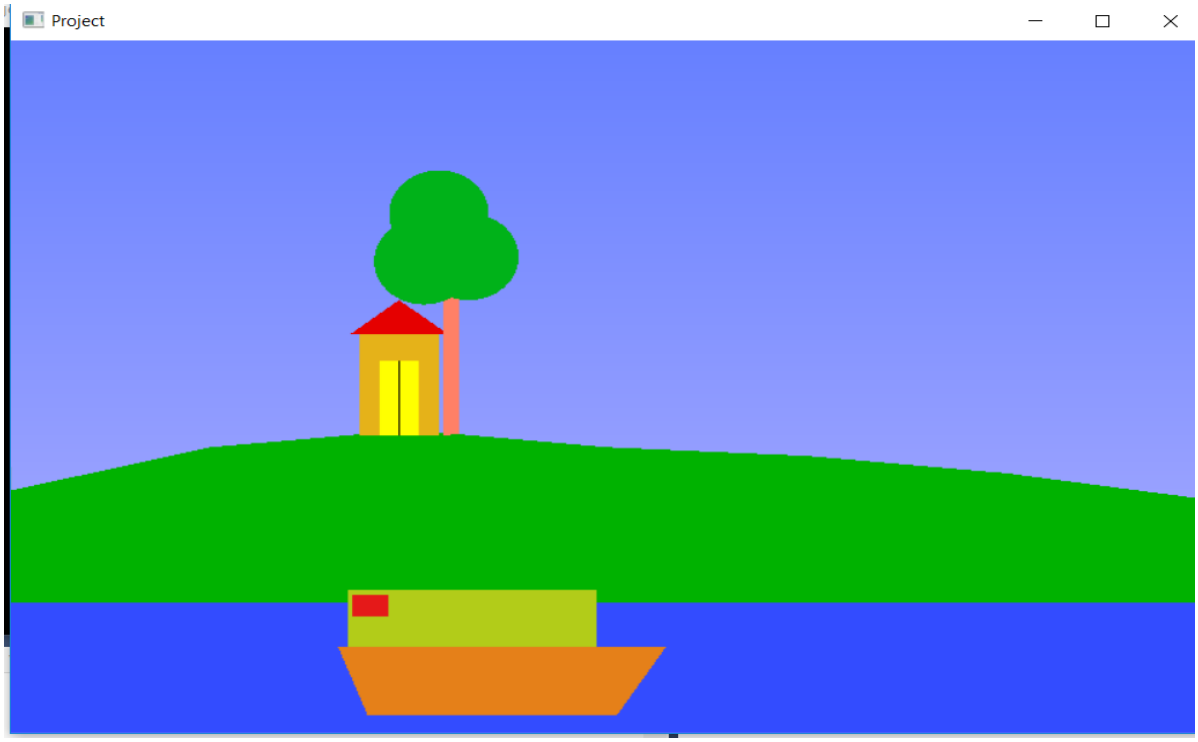
```
        glBegin(GL_POLYGON); //House
        glColor3f(0.9, 0.0, 0.0);
        glVertex2i(340, 460);
        glVertex2i(390, 500);
        glVertex2i(440, 460);
        glEnd();

        glBegin(GL_POLYGON); //Door
        glColor3f(1.0, 1.0, 0.0);
        glVertex2i(370, 344);
        glVertex2i(370, 430);
        glVertex2i(410, 430);
        glVertex2i(410, 344);
        glEnd();
        glBegin(GL_LINE_LOOP);
        glColor3f(0.0, 0.0, 0.0);
        glVertex2i(390, 344);
        glVertex2i(390, 430);
        glEnd();
       glPopMatrix();
}
void drawBoat(int i)
{
        glPushMatrix();

        glBegin(GL_POLYGON); // Boat Starts
        glColor3f(0.7, 0.8, 0.1);
        glVertex2i(230, 100);
        glVertex2i(230, 165);
        glVertex2i(480, 165);
        glVertex2i(480, 100);

        glEnd();

        glBegin(GL_POLYGON);
        glColor3f(0.9, 0.5, 0.1);
        glVertex2i(250, 20);
```

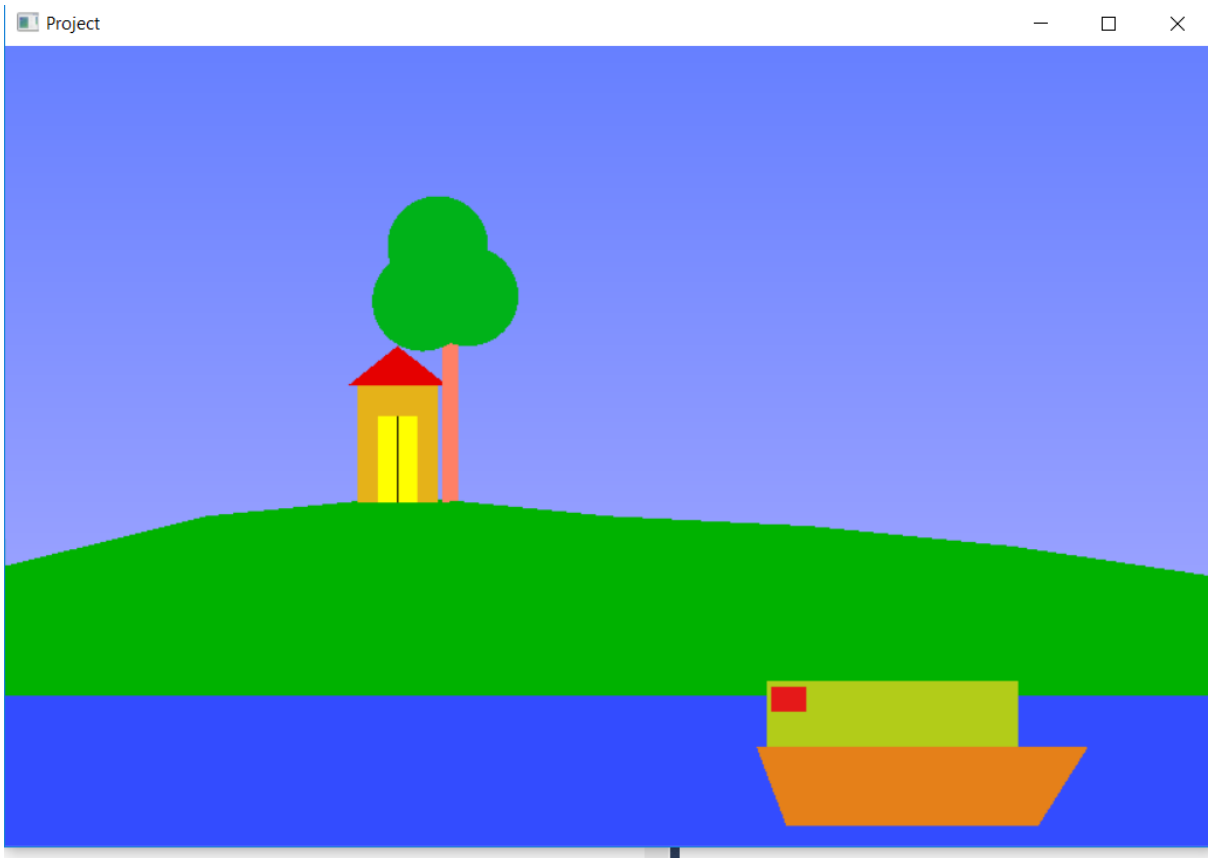
```
        glVertex2i(220, 100);
        glVertex2i(550, 100);
        glVertex2i(500, 20);
        glEnd();

        glBegin(GL_POLYGON);
        glColor3f(0.9, 0.1, 0.1);
        glVertex2i(235, 135);
        glVertex2i(235, 160);
        glVertex2i(270, 160);
        glVertex2i(270, 135);
        glEnd();
        glFlush();
        glPopMatrix();
}
void myInit(void)
{
        glClearColor(0.0, 0.0, 1.0, 0.0);
        glColor3f(1.0f, 1.0f, 1.0f);
        glPointSize(0.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0.0, 1200.0, 0.0, 800.0);
}
void keyboard(unsigned char key, int x, int y)
{

        switch (key) {

        case 'b':       //boat start
        case 'B':
                boatStatus = 1;
                break;
        case 'n':       //boat stop
        case 'N':
                boatStatus = 0;
                break;
```

```
	default:
		break;
	}
}

void boat()
{
	if (boatStatus == 1)
	{
		boatX += .5;
	}
	if (boatX > 1000)
	{
		boatX = -600;
	}
	glPushMatrix();
	glTranslatef(boatX, boatY, 0);
	drawBoat(1);
	glPopMatrix();
}

void myDisplay(void)
{
	scene();
	tree();
	boat();

	glFlush();

	glutPostRedisplay();
	glutSwapBuffers();
}

void main(int argc, char** argv)
{
	glutInit(&argc, argv);
```

```
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(1250, 600);
        glutInitWindowPosition(0, 0);
        glutCreateWindow("Project");
        glutKeyboardFunc(keyboard);
        glutDisplayFunc(myDisplay);
        myInit();
        glutMainLoop();
}
```

# Chapter 6: Snapshots



# CHAPTER 7:

# CONCLUSION AND FUTURE ENHANCEMENT

## CONCLUSION

An attempt has been made to develop an OpenGL package which meets necessary requirements of the user successfully. Since it is user friendly, it enables the user to interact efficiently and easily. The development of the mini project has given us a good exposure to OpenGL by which we have learnt some of the technique which help in development of animated pictures, gaming. Hence it is helpful for us even to take up this field as our career too and develop some other features in OpenGL and provide as a token of contribution to the graphics world.

# CHAPTER 8:

## BIBLIOGRAPHY

[1] Interactive Computer Graphics A Top-Down Approach with OpenGL, Edward Angel, 5th Edition, Addison- Wesley, 2008.

[2] Introduction to the Design and Analysis of Algorithms, Anany Levitin , 2nd Edition, Pearson Education, 2009.

[3] http://www.cprogramming.com/tutorial/opengl_introduction.html.

[4] www.**opengl**.org/resources/code/samples/redbook/.

[5] Computer Graphics Using OpenGL-F.S Hill Jr. 2nd Edition, Pearson Education 2001.

[6] Computer Graphics-OpenGL version – Donald Hearn and Pauline Baker,2nd Edition, Pearson Education 2003.

[7] www.NeHe.com.

[8] www.Wikipedia.org.

[9] www.projectparadise.com.

[10] www.openglprojects.com.