**CHAPTER 4**

**IMPLEMENTATION**

System implementation is the important stage of project when the theoretical design is tuned into practical system.

**4.1 Functions used in The Lighthouse**

The Algorithm is constructed in a set of functions. Several user defined functions and built\_in functions are used for implementing this project. The following are the functions used in this project and their purpose:

* **glutInitDisplayMode (GLUT\_DOUBLE|GLUT\_RGB):** specifies whether to use an *RGB* or color-index color model. You can also specify whether you want a single- or double-buffered window. (If you're working in color-index mode, you'll want to load certain colors into the color map; use glutSetColor () to do this.) Finally, you can use this routine to indicate that you want the window to have an associated depth, stencil, and/or accumulation buffer. For example, if you want a window with double buffering, the RGBA color model, and a depth buffer, you might call glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB|GLUT\_DEPTH).
* **The Display Callback: glutDisplayFunc (void (\*func)(void))** is the first and most important event callback in the code. Whenever GLUT determines the contents of the window need to be redisplayed, the callback function registered by glutDisplayFunc () is executed. Therefore it registers display callback function
* **Running the program:** The very last thing to be done is call glutMainLoop (void). All windows that have been created are now shown, and rendering to those windows is now effective. Event processing begins, and the registered display callback is triggered. Once this loop is entered, it is never exited!

The various other functions used are described in more detail below:

* **glutBitmapCharacter (font,string[size]);** it renders the character in the named bitmap font and advances the current raster position.
* **glutInit (&argc, char \*\*argv);** this command initializes GLUT. The argument from main are passed in and be used by the application.
* **glutInitWindowSize (600, 500);** this command specifies the initial height width of the window in pixels.
* **glutCreateWindow (“name”);** creates a window on the display. The string title can be used to label the window.
* **glutMainLoop ();** causes the program to enter an event processing loop.
* **void glBegin(glEnum mode);**

Initiates a new primitive of type mode and starts the collection of vertices. Values of mode include GL\_POINTS, GL\_LINES and GL\_POLYGON.

* **void glEnd( );**

It 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.

* **4.3.5 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.

* **4.3.6 void glutInitWindowSize(int width, int height);**

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

* **4.3.7 void glutInitWindowPosition(int x, int y);**

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

* **4.3.8 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);

* **4.3.9 void glFlush( );**

Forces any buffered any OpenGL commands to execute.

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

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

* **4.3.11 void glutMainLoop( );**

Cause the program to enter an event processing loop. It should be the last statement in main.

* **4.3.12 void glutDisplayFunc(void (\*func) (void));**

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

* **4.3.13 gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);**

Defines a two-dimensional viewing rectangle in the plane Z=0;

* **4.3.14 void glutBitmapCharacter(void \*font, int char);**

Renders the character with ASCII code char at the current raster position using the raster font given by font. Fonts include GLUT\_BITMAP\_TIMES\_ROMAN\_10 and GLUT\_BITMAP\_TIMES\_ROMAN\_8\_Y\_13. The raster position is incremented by the width of the character.

* **4.3.15 void glClear(GL\_COLOR\_BUFFER\_BIT);**

To make the screen solid and white.

* **4.3.16 void MouseFunc(myMouse);**

It is used for the implementation of mouse interface.

Passing the control to void myMouse(int button,int state,int x,int y);

* **4.3.17 void KeyboardFunc(key);**

It is used for the implementation of keyboard interface.

Passing control to void key(unsigned char key,int x,int y);

* **4.3.18 void glLoadMatrix[fd](TYPE \*m);**

Loads the 16 element array of TYPE GLfloat or GLdouble as a current matrix.

**4.2 Algorithm**

STEP 1: Start

STEP 2: if key N/n is pressed then

goto step 3

STEP 3: Display the Lighthouse scenario

STEP 4: If key A/a is pressed then

goto step 5

STEP 5: Move the boat to the left

STEP 6: if key D/d is pressed then

goto step 7

STEP 7: Move the boat to the right

STEP 8: If the key Q/q is pressed then

goto step 9

STEP 9: exit from the scene.

STEP 10: Stop