CHAPTER 1

INTRODUCTION

Computer graphics is one the most effective and commonly used methods to communicate the processed information to the user. It is widespread today. Computer imagery is found on television, in newspaper, in weather reports, or for example, in all kinds of medical investigation and surgical procedures, displaying the information in the form of graphics instead of simple text.

Computer animation is the art of creating moving images via the use of computers. It is a subfield of computer graphics and animation. Sometimes the target of the animation is the computer itself, but sometimes the target is another medium, such as film. It is also referred to as CGI (computer generated imagery or computer generated imaging), especially when used in films.

To create the illusion of movement, an image is displayed on the computer screen then quickly replaced by a new image that is similar to previous image, but shifted slightly.This technique is identical to the illusion of movement in television and motion pictures.

AIM: To implement the 3D animation of implementation of "**Ocean bed Scene 3D**" using OPENGL as graphics tool.

1.1 BRIEF DESCRIPTION

The graphics implemented in the mini project named "**Ocean bed Scene 3D**" is to demonstrate the different features of computer graphics such as modelling and rendering, lighting and shading, transformation, and user interface.

This mini project shows the simple scene of the Ocean bed, using different objects like fishes, octopus, starfish, crab and plants. These objects are created using simple 3d geometric figures like cubes and spheres.

The objective of the project is all about creating a colorful environment between the user and the program through suitable interaction which would be provided by necessary event based driven function included in code.

1.2 OBJECTIVE OF THE MINI PROJECT

The objective of the mini project is all about creating a colorful environment between the user and the program through suitable interaction which would be provided by necessary event based driven function included in code. This project mainly deals with vertices and basic primitives.

Scope of the problem: The scope of the problem in this mini project is that we can use mouse function to display menus and can change the scene properties.

1.3 CONCEPTS AND PRINCIPLES

Image

In common usage, an image or picture is an artifact, usually 2-diminsional, that has a similar appearance to some subject---usually a physical object or a person. Images may be 2-dimensional such as a photograph, screen display, and as well as a 3- dimensional such as a statue. They may be captured by optical devices - such as cameras, mirrors, lenses, telescopes, microscopes etc. and natural objects and phenomena, such as the human eye or water surfaces. A digital images is a representation of a 2-dimensional image using ones and zeros (binary). Depending on whether or not the image resolution is fixed, it may be of vector or raster type. Without qualification, the "digital image" using refers to raster images.

Pixel

In the enlarged portion of the image individual pixels are rendered as squares and can be easily seen.

In digital imaging, a pixel is the smallest piece of information in an image. Pixels are normally arranged in a regular 2-dimensional grid, and are often represented using dots or squares. Each pixel is a sample of an original image, where more samples typically provide a more accurate representation of the original. The intensity of each pixel is variable; in color systems, each pixel has typically three or four components such as red, green, and blue, or cyan, magenta, yellow, and black.

Rasterization

Fragments are potential pixel. Each fragment has a location in screen coordinates that corresponds to a pixel location in the color buffer.it means mapping of vertex coordinates into pixel in frame buffer

Graphics

Graphics are visual presentations on some surface, such as a wall, canvas, computer screen, paper, or stone to brand, inform, illustrate or entertain. Examples are photograph, drawings, line art, graphs, diagrams, topography, numbers, symbols, geometric designs, maps, engineering, or other images. Graphic often combine texts , illustrations and color. Graphic design may consist of the deliberate selection, creations or arrangement of typography alone, as in a broacher, flier, poster, website, or book without any other element. Clarity or effective communication may be the objective, association with other cultural elements may be sought, or merely, the creation of a distinctive style.

Rendering

Rendering is the process of generating an image from a model, by means of computer programs. The model is a description of three dimensional objects in a strictly defined language or data structure. It would contain geometry, viewpoint, texture, lighting, and shading information. The image is a digital image or raster graphics image. The term may be by analogy with an "artist's rendering" of a scene. 'Rendering' is also used to describe the process of calculating effects in a video editing file to produce final video output.

1.4 ORGANIZATION OF THE PROJECT

- The first chapter of the report is the *introduction.* It contains the information about the introduction to the project, objective of the project, scope of the project and the method of implementation.
- The second chapter is the *Literature survey* which includes matter from (and refer) papers/books and other sources.
- The third chapter is the *Requirements* which include the hardware and software requirements.
- The fourth chapter is the *Design* which includes the flow of control of the program and the modules used in the program.
- The fifth chapter is the *Implementation* which shows how the various modules in the program are implemented.
- The sixth chapter is the *Snapshots* which includes the test cases and the snapshots.
- The seventh chapter is the *Conclusion and the future scope* which concludes the project
- And gives the future enhancements that can be made to the project.
- Next chapter is *Bibliography* which includes the names of referred books and websites.
- Next comes the *Appendix A* and *Appendix B* which contains the information's about functions used in the program and libraries respectively.

CHAPTER 2

SOFTWARE REQUIREMENT SPECIFICATION

Software requirement specification is a complete description of the behavior of a system to be developed and may include a set of users cases that describe interactions the user will have with the software.in addition to description of the software functions, it also contains nonfunctional requirements.

2.1 FUNCTIONAL REQUIREMENTS

For the successful implementation of the project, the system must allow the user to control the movements of the engine for the respective keys on the keyboard as well as the mouse.

2.2 NON-FUNCTIONAL REQUIREMENTS

Non-Functional Requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions.

The minimum hardware requirements for the project are listed below:

- CPU frequency : 1.6 GHz
- RAM capacity : 384 MB
- Display Dimensions: 1024x786
- A Keyboard (Real/Virtual)
- A Mouse (Pointing device)

The following operating systems are supported

- Windows XP Service pack 2 or above
- Ubuntu

The software requirements are

- Any one of the above mentioned OS
- C and C++ Libraries
- OpenGL libraries

CHAPTER 3

LITERATURE SURVEY

This chapter shows the Literature Survey we have made in order to get an idea about this project, and to complete this project in time. The chapter discuss about the existing systems, overview of the proposed system and feasibility study of the project.

3.1 EXISTING TECHNOLOGY

The Ocean bed Scene 3D is structed animation usually used as a screensavers and sometimes used as an educational tool.

Many videos are taken underwater for study purpose and also during vaction. Hence National geographic channel and other similar organization have take many footage for study, explanation and also for research.

The National geographic channel and other similar organization have created and developed the softwares for the rendering of aquatic animals, which is almost very reliastic, used for explanation and also for research.

3.2 PROPOSED SYSTEM

The proposed system is an attempt of illustrating Ocean Bed Scene through a simple and yet powerful tool,OpenGL . This system concentrated on modeling and rendering of 3D objects and also transformation such as rotaton, zooming, lighting and user interface for the mentioned transformation.

This system is design with the help of an API called OpenGL implementing the different features od computer graphics.

3.3 FEASIBILITY STUDY

A feasibility study is required to be done for every project in order to assess if it can be implemented successfully by making use of the available resources.

Since OPENGL provides its users with a variety of functions for various purposes, implementation of the project in hand, is feasible, both technically and economically.

CHAPTER 4

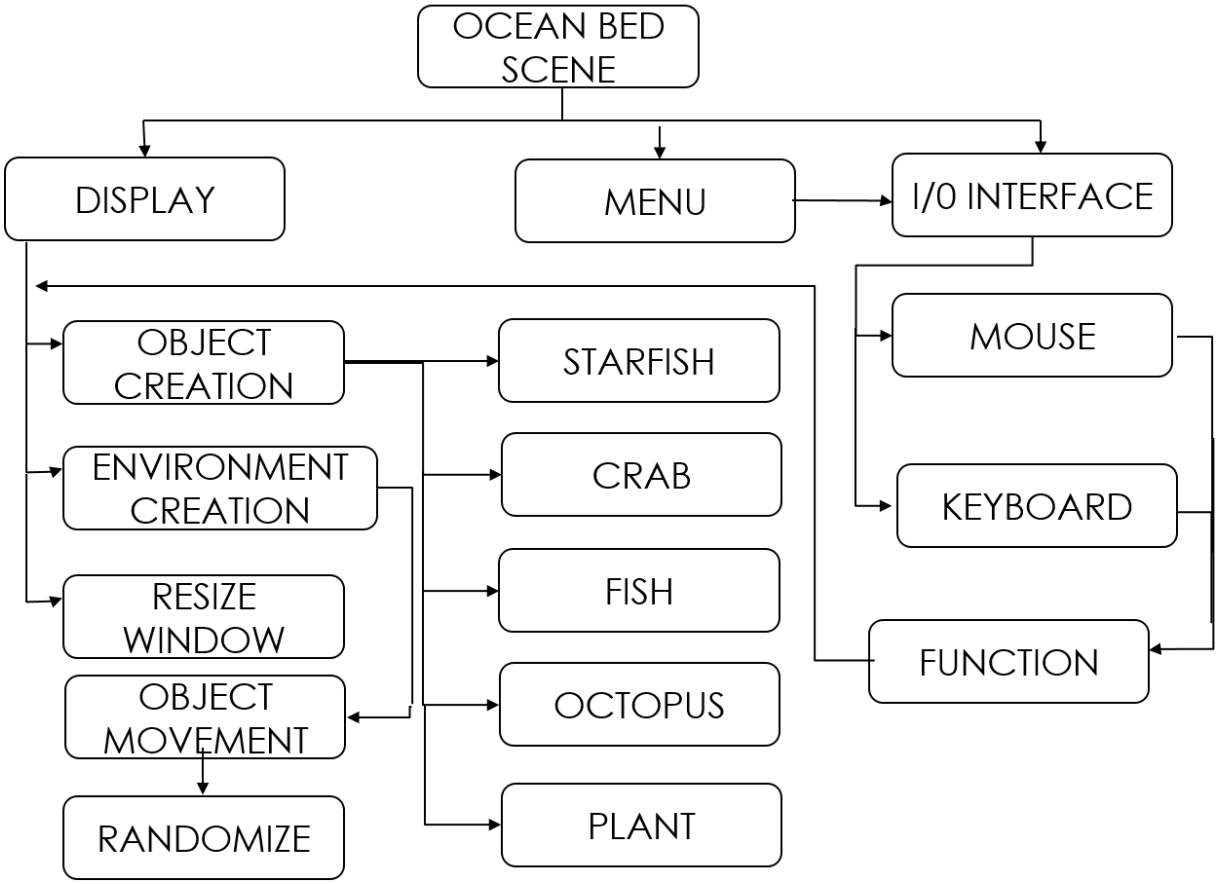
DESIGN

The complete overview of the project is shown in the block diagram; it describes the actions that will be performed on the interfaces generated by the keyboard and the mouse.

4.1 SYSTEM DESIGN

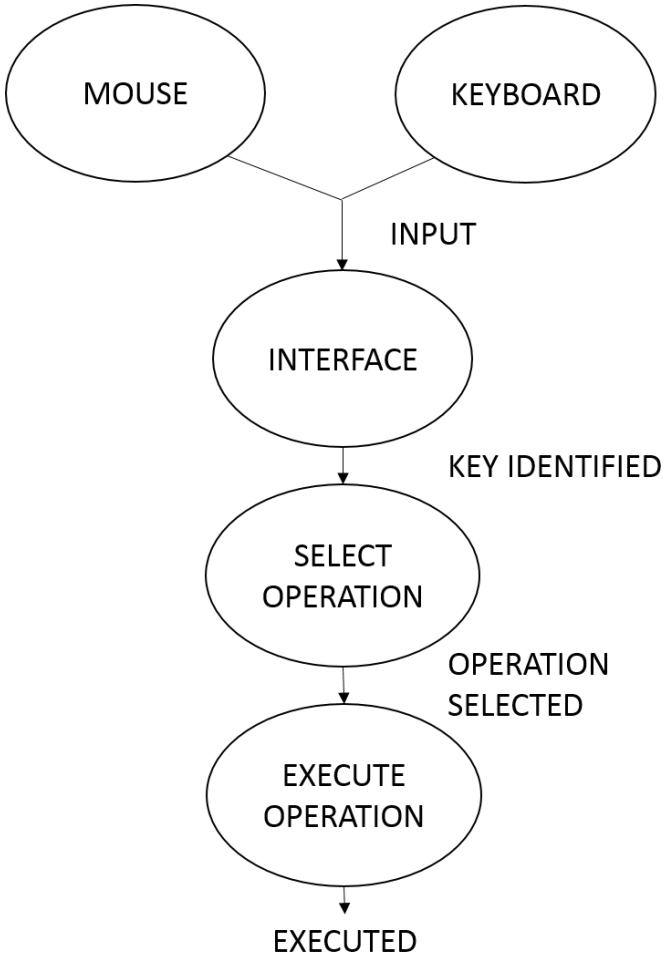
This section presents the flow associated with this project which shows the flow of the program and gives the various actions taken on the different inputs.

The overall system design is shown in fig 4.3



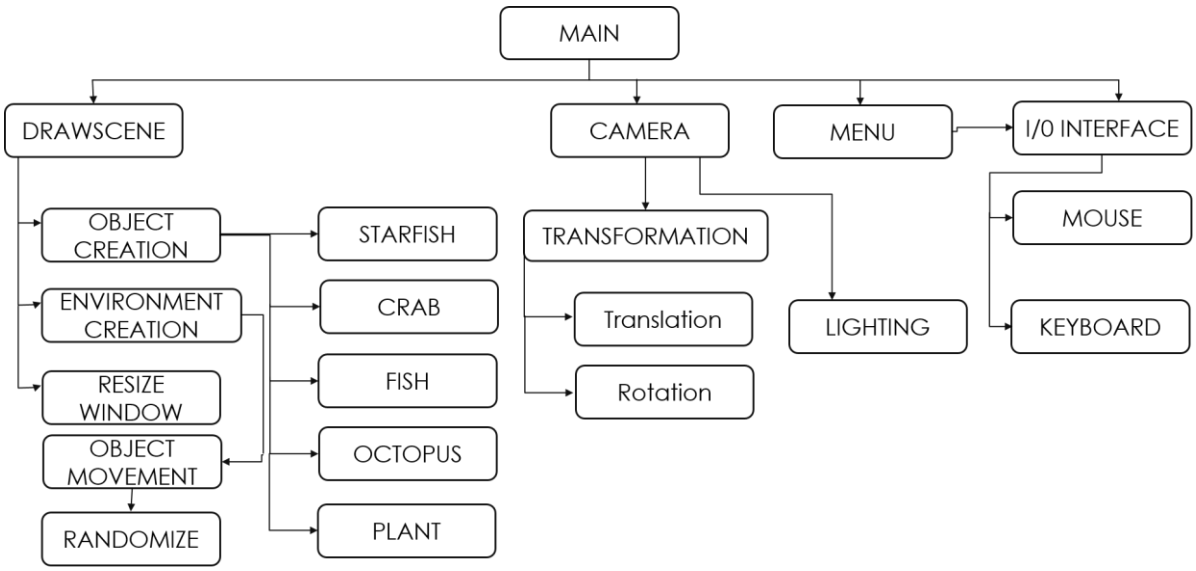
4.1 Overview

4.1.1 BLOCK DIAGRAM



4.1.1 Design of the project

4.2 DETAILED DESIGN



4.2 Detailed Design of the project

4.3 GRAPHICS FEATURES

OpenGL (**Open G**raphics **L**ibrary) is a standard specification defining a cross-language. Cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists various function calls which can be used to draw complex three dimensional scenes from simple primitives. It is widely used CAD, virtual reality, scientific visualization, information visualization, flight simulation and video games.OpenGl API uses seven major functions primitives,attribute, viewing, transformation, input, control function.

We make use of most of the above functions in our program.

To draw the different objects of our project like Gumbaz, minars etc we use primitive functions like cubes and spheres.

Transformation is the backbone of computer graphics,enabling us to manipulate the shape,size and location of the object. It is used to effect the following changes in a geometric object, change the location, change the size, change the shape, rotate, animate the object.

There are four types of transformations- translation, scaling, rotation, shearing. To apply transformation like scaling, rotation we use transformation functions provided by openGL.

Rotation of planar body is the movement when points of the travel in circular projector around a fixed point called centre of rotation. Mathematically, rotation is a rigid body movement which keeps a point fixed; unlike a translation.

Coloring is the operation in which we assign colors to the objects, openGL uses RGB coloring scheme to color the primitives. Each color component is stored separately. Usually 8 bits per component.

Lighting and shading also plays as an important feature of graphics to produce 3D images. In our project, we use 3 different light source namely Ambient, Diffuse and Specular which helps to gain perfect lighting effect on obects. The use of Smooth shading for the image during rendering gives the image more realistic look.

4.4 USER DEFINED FUNCTIONS USED IN OpenGL

The functions used in the programs are as follows:

- ***main ( ) :***

The control will first enter into this function and in this function it will create a window and it will initialize the window and then this function calls display function.

- ***static void animator(int type);****:*

This function just renders the scene every 25 milliseconds. A timer is used to give smooth animation at the same rate on differnt computers.

- ***void resizeWindow(int w, int h) :***

This function will resizes the current viewport to the full window size.

- ***void keyboardInput(unsigned char key, int x, int y) :***

This function handles keyboard input for normal keys.

- ***void keyboardInput(int key, int x, int y) :***

This function is used to process special keyboard keys like F1, F2, etc.

- ***void menu(int val)*** *:*

This function is called when a menu option has been selected. Translates the menu item identifier into a keystroke, then call's the keyboard function.

- ***void create_menu(void)****:*

This function is used to create the menu upon clicking the left button of the mouse and the menu list appears.

- ***void mouse(int button, int state, int x, int y)*** **:**

This function handles mouse input buttons and translates the menu item identifier into a keystroke, then call’s the keyboard function.

- ***void drawScene() :***

This function calls the render method of the current scene to render the contents of the scene onto the window created.

- ***void addObject(int type) :***

This function is used to add the object to the scene during the execution of the project.

4.5 OPENGL API'S USED :

- **glutKeyboardFunc(key)**: This Function is used when keyboard event occurs, the ASCI code for the key that generated the event and the location of the mouse are returned.

- **glutMainLoop():** Function that starts the event processing engine. If there are no events to process, the program will sit in a wait state until we terminate the program through some external mean hitting a special key or a combination of keys.

- **glutPostRedisplay():** Using this function, rather than invoking the display callback directly, avoids extra or unnecessary screen drawings by setting a flag inside glut main loop indicating that the display needs to be redrawn.

- **glLoadIdentity():** Function to select the matrix to which the operations apply by first setting the matrix mode.

- **glEnable(GLenum feature):** enables an OpenGL feature.

- **glDisable():** disables an OpenGL feature.

- **glutInitWindowSize(Xsize,Ysize)** and **GlutInitWindowPosition(h,w) :** To set initial window size and position of the window.

- **glutDisplayFunc(display)** : Registers display callback with the window system. Here the function named display will be called whenever the window system determines that the opengl window needs to be redisplayed.

- **glViewport(int x,int y,GLsizei width,GLsizei height):** specifies a width * height viewport in pixels whose lower left corner is at (x,y) measured from the origin of the window.

- **glClearColor():** Function used to clear the window.

- **glFlush():** Empties all of the buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.

- **glutReshapeFunc(void *f(int width,int height):** registers the reshape callback function returns the height and width of the new window. The reshape callback invokes a display callback.

- **glutTimerFunc(unsigned int msecs, void (*func)(int value), value)) :**This finction registers the timer callback func to be triggered in at least msecs milliseconds. The value parameter to the timer callback will be the value of the value parameter to glutTimerFunc. Multiple timer callbacks at same or differing times may be registered simultaneously.

- **glShadeModel(GLenum mode):** GL primitives can have either flat or smooth shading. Smooth shading, the default, causes the computed colors of vertices to be interpolated as the primitive is rasterized, typically assigning different colors to each resulting pixel fragment. Flat shading selects the computed color of just one vertex and assigns it to all the pixel fragments generated by rasterizing a single primitive. In either case, the computed color of a vertex is the result of lighting if lighting is enabled, or it is the current color at the time the vertex was specified if lighting is disabled.

- **glPolygonMode(GLenum face, GLenum mode):** This function controls the interpretation of polygons for rasterization. face describes which polygons mode applies to: both front and back-facing polygons (GL_FRONT_AND_BACK). The polygon mode affects only the final rasterization of polygons. In particular, a polygon's vertices are lit and the polygon is clipped and possibly culled before these modes are applied.

- **glutAttachMenu(int button):** attaches a mouse button for the current window to the identifier of the current menu.

- **glutAddMenuEntry(char *name, int value):** This Function adds a menu entry to the bottom of the current menu. The string name will be displayed for the newly added menu entry. If the menu entry is selected by the user, the menu's callback will be called passing value as the callback's parameter.

- **glPixelStorei(GLenum pname, GLint param):** This function can be used to set any pixel store parameter. Boolean parameters are set to false if param is 0 and true otherwise. Where pname specifies the symbolic name of the parameter to be set and param specifies the value that pname is set to.

CHAPTER 5

IMPLEMENTATION

5.1 OPENGL

OpenGl provides a set of commands to render a 3D scene, that means you provide the data in an OpenGL- usuable form and OpenGL will show this data on the screen. It is developed by many companies and it it free to use. You can develop opengl- application without licensing.
OpenGL hardware- and system- independent interface. An openGL- application will work on every platform as long as there is an installed implementation.
Bexouse it is system independent, there are no functions to create windows etc, but there are helper functiond for each platform. A very useful thing is GLUT

5.1.1 GLUT

GLUT is a complete API written by Markilgard which lets you create windows and handle the messages. It exists for several platforms, that means that a program which uses glut can be completed on many platforms without (or atleast with ver few) changes in the code.

5.2 OpenGL OVERVIEW

- OpenGL (Open Graphics Library) is the interface between graphics program and graphics hardware. *It is streamlined*.
- In other words, it provides low-level functionality. For example, all objects are built from points, lines and convex polygons.
- Higher level objects like cubes are implemented as six four-sided polygons.
- OpenGL supports features like 3-dimensions, lighting, anti-aliasing, shadows, textures, depth effects, etc.
- It is system-independent. It does not assume anything about hardware or operating system and is only concerned with efficiently rendering mathematically described scenes. As a result, it does not provide any windowing capabilities.
- It is a state machine. At any moment during the execution of a program there is a current model transformation.
- It is a rendering pipeline. The rendering pipeline consists of the following steps:

    * Defines objects mathematically.
    * Arranges objects in space relative to a viewpoint.
    * Calculates the color of the objects.
    * Rasterizes the objects.

5.2.1 OpenGL Pipeline Architecture

Most implementations of OpenGL have a similar order of operations, a series of processing stages called the OpenGL rendering pipeline. Although this is not a strict rule of how OpenGL is implemented, it provides a reliable guide for predicting what OpenGL will do. Geometric data (vertices, line, and polygons) follow a path through the row of boxes that includes evaluators and per-vertex operations, while pixel data (pixels, images and bitmaps) are treated differently for part of the process. Both types of data undergo the same final step (rasterization) before the final pixel data is written to the frame buffer.

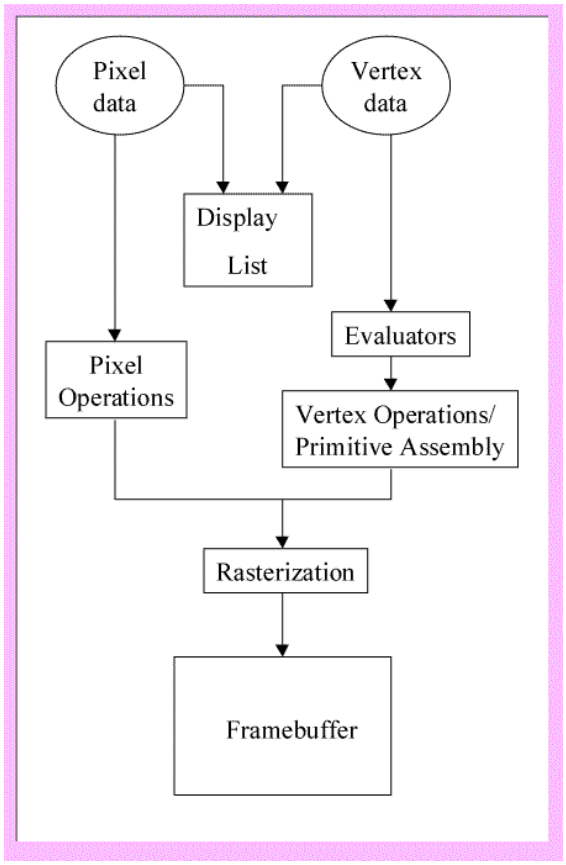


Fig 1.1 Block diagram of OpenGL pipeline architecture

5.3 Pseudocode

Include the necessary header files.

declare and initialize the variables
/*Declare function Prototypes*/

void animator(int type)
{
- requests that the display callback be executed after the current callback returns
- a timer callback to be triggered in a specified number of milliseconds.

}

void resizeWindow(int w, int h)
{
- set viewport to (0,0,w,h)
- set the size of the window to the size of the display screen
- change the size of the display window.

}

void menu(int val)
{
Switch the cases for the menu list for options like adding objects, toggle fog, wireframe, lighting and, increase and decrease of camera distance from the center, based on the keystroke from mouse or keyboard interface.
}

void create_menu(void)
{
- create the menu.
- Add the keystroke to display the menu
- The list of options are coded to display with the value to return to the menu's callback function if the menu entry is selected.

}

void mouse(int button, int state, int x, int y)
{
- Switch the cases for the mouse button
- if left mouse button clicked then if it is pressed then rotate the scene clockwise.
- if scroll button is rotated up then the distance of the camera from the center is decreased.
- if scroll button is rotated down then the distance of the camera from the center is increased.

}

void addObject(int type)
{

- first pick the x and z locations
- create an object from the class Renderable for the scene for the objects to be displayed.
- Switch the cases for the type of object to be added, hence calling new function creates an object for the scene.
- set the created objects to new position.
- add the object to rendering queue.
- increment the count of this type of object.

}

*int main(int argc, char *argv[])*
{

- initialize the variable n1 to a random value < 50
- call init and display function
- call reshape function for fullscreen output.
- call keyboard and mouse function.
- create the menu list
- setup and initialize the state of lighting and fog using keyboard input function.
- add the objects to the screen using for loop to generate the ‘n1’ objects.

}

CHAPTER 6

TESTING

System Testing of software or hardware is testing conducted on a complete, integrated system to evaluate the systems complains with its specifies requirements. System testing is actually a series of different tests whose primary purpose is to fully exercise the computed based system although each text has a different purpose.

Verification and validation is the generic name given to checking processes, which ensures that the software confirms to its specification and meets the demands of users.

There are several rules that serve as testing objectives:

- Testing is a process of executing program with the intent of finding an error.
- A good test case is one that has a high probability of finding an undiscovered error.

The testing has been conducted successfully according to the objectives stated and hence it uncovers error.

Test case	Expected output
Press F1	Displays/Hides the menu
Press F2	Add a Crab
Press F3	Add an Octopus
Press F4	Add a Starfish
Press F5	Add a fish
Press F6	Add a plant
Press F,f	Toggle fog ON/OFF
Press W,w	Toggle wire frame
Press L,l	Toggle lighting ON/OFF
Press 1,0	Toggle Focus lighting ON/OFF
Press 5	Increase elevation angle of the scene
Press 0	Decrease elevation angle of the scene
Press UP,8	Decrease distance from the center
Press DOWN,2	Increase distance from the center
Press LEFT,4	Rotate Clockwise
Press RIGHT,6	Rotate Anticlockwise
Press ESC	Exit
Mouse RIGHT BUTTON	Display menu
Mouse LEFT BUTTON	Rotate Clockwise
Mouse scroll button UP	Decrease distance from the center
Mouse scroll button DOWN	Increase distance from the center

Table 6.1 Testing result

CHAPTER 7

CONCLUSION

The images defined by our OpenGL program will be formed automatically by the hardware and software implementation of the image formation process. In this project, the images have been created using simple geometric objects.

The program also has a user friendly interface that is menu oriented.

This project “OCEAN BED SCENE 3D” includes the rendering of animated scene of many aquatic animals which shows the lighting and shading effect.

Our mini project is only an attempt to understand some of the concepts of comuter graphics using openGL .

.

CHAPTER 8

FUTURE ENHANCEMENTS

This mini project implements 'Ocean bed Scene 3D' animation using Opengl interface. The development of this project would improve the user's knowledge about computer graphics and Opengl and this project provides good understanding of Opengl prominent functions like transformation, rotation and 3D Modelling.As mentioned before, we did assume certain details during the process of making the code. Hence there is a lot of scope for improvisations on this project.

Designing and developing the 'Ocean Bed Scene 3D' was an interesting project, it helped in learning more about managing projects, deadlines and a host of other things. This was the first of its kind in terms of serious programming.

In future, the objects in the scene can be upgraded to be more reliastic and life like. Efforts have been put to see that the project is robust one, with user friendly interface. Everything in the universe is not perfect; similarly even this project can have short comings. It was an unforgettable experience working on this project and we have learnt a lot in this process. This mini project is efficient as of now. The project has many advantages and disadvantages. The mini project can be upgraded as per future requirements.

Many improvements can be thought of to this project, such as

1. Three dimensional effects of the scene can be enhanced.
2. We can improve project by using advanced functions and also interactive functions which enhances the creditability of the Ocean Bed Scene.
3. Different lighting effects can be used to enhance the visuals of the Scene.
4. Different primitives can be used.
5. More complex levels can be implemented.
6. Choice of objects to choose from the user can be made.
7. More realistic simulation of objects.
8. Including a background image.

LIMITATIONS

This mini project implements Ocean Bed Scene in 3D however there are some limitations too.
Some of the limitations are:

- If Double buffering is not used flickering will be experienced.
- It is used in virtual implementation only.
- It requires powerful processor and memory.
- The code is written in C++, hence the code must be compiled in every system everytime to execute.
- The part of the code must be changed while changing the OS platform(ie. Linux or Windows or MacOS)
- The code is not portable.

CHAPTER 9

BIBLIOGRAPHY

BOOKS :

[1] Edward angel: Interactive computer graphics A TOP-DOWN Approach with OpenGL, 2nd edition, Addison-Wesley, 2000.

[2] Angel, E., OpenGL, A Primer, Third Edition, Addison-Wesley, Reading, MA, 2008.

WEBSITES :

[3] http://www.opengl.org

[4] http://www.wikipedia.com

[5]http://www.opengl.org/about/arb

[6] www.opengl.org/wiki/Primitives

[7]www.2dix.com

[8]www.gameinfo.com

APPENDIX A - SNAPSHOTS:

Screenshots:

FIGURE 1: NORMAL EXECUTION

FIGURE 2: SCENE WITHOUT FOG

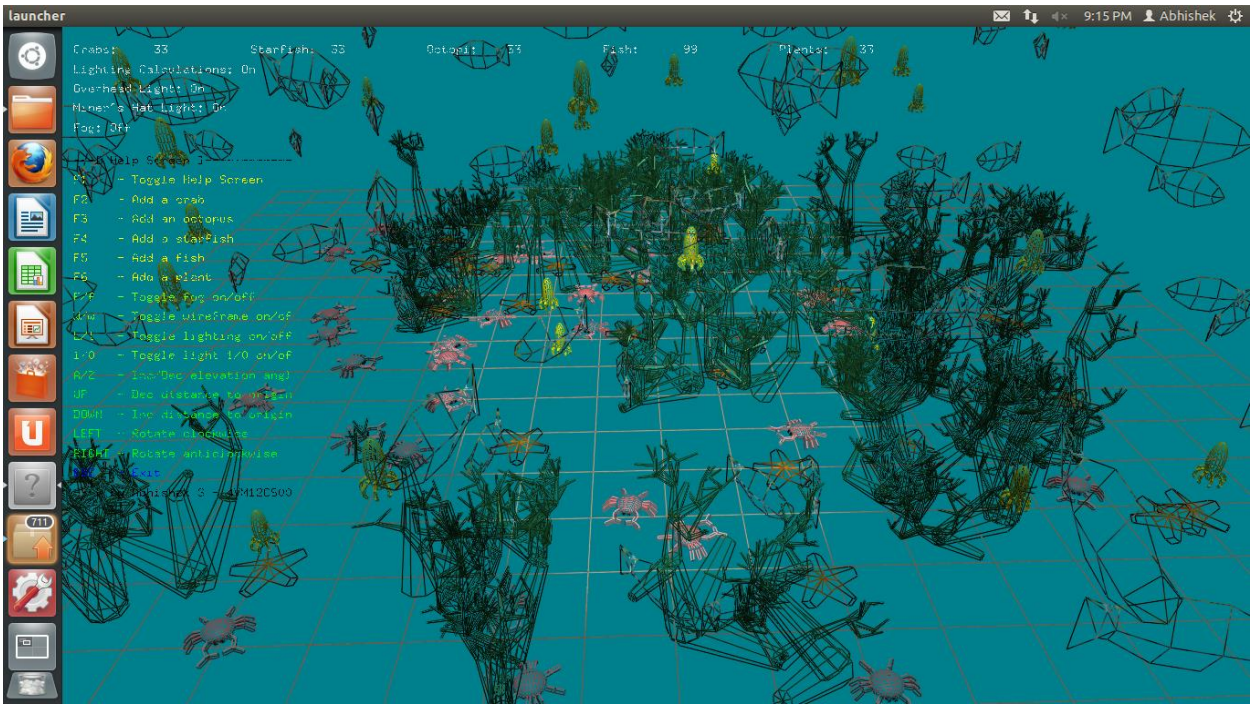


FIGURE 3: WIREFRAME

FIGURE 4: SCENE WITH LIGHTING OFF

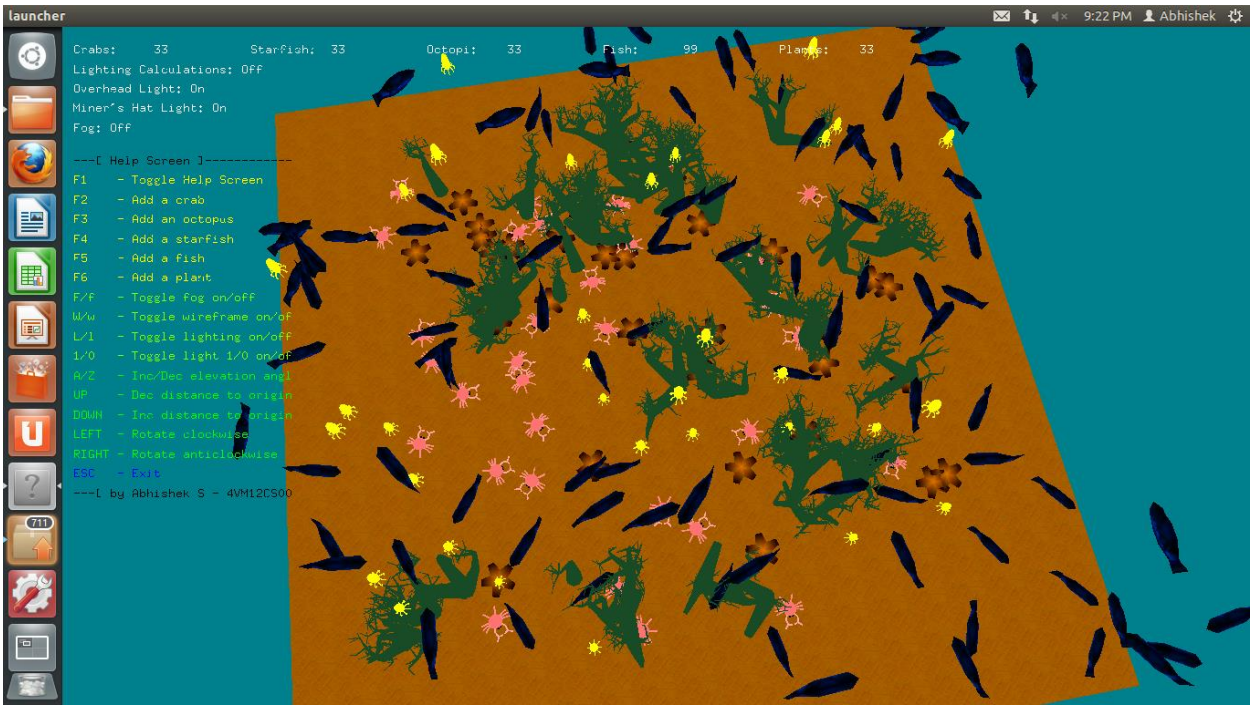


FIGURE 5: SCENE WITH TOP VIEW

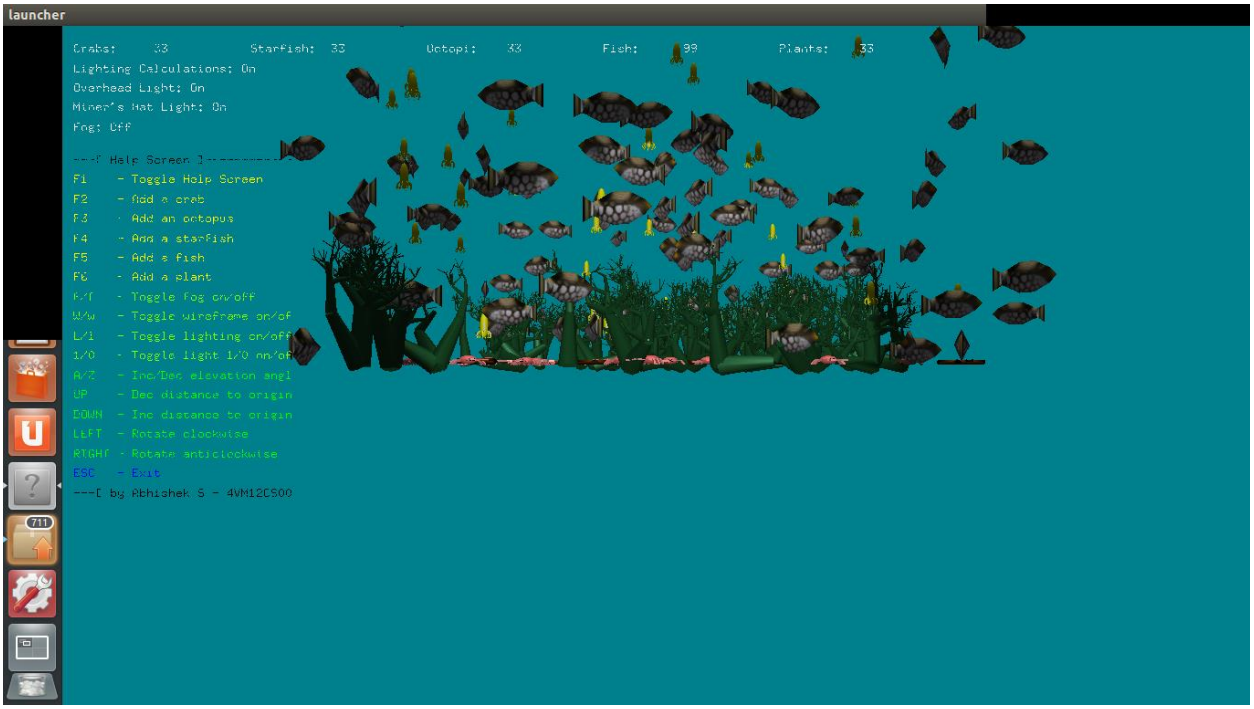


FIGURE 6: SCENE WITH HORIZONTAL VIEW

FIGURE 7: SCENE WITH LIGHTS OFF

FIGURE 8: SCENE ROTATING ANTICLOCKWISE

FIGURE 8: SCENE ROTATING CLOCKWISE

APPENDIX B - OPENGL API'S

- **glutKeyboardFunc(key)**: This Function is used when keyboard event occurs, the ASCI code for the key that generated the event and the location of the mouse are returned.

- **glutMainLoop():** Function that starts the event processing engine. If there are no events to process, the program will sit in a wait state until we terminate the program through some external mean hitting a special key or a combination of keys.

- **glutPostRedisplay():** Using this function, rather than invoking the display callback directly, avoids extra or unnecessary screen drawings by setting a flag inside glut main loop indicating that the display needs to be redrawn.

- **glLoadIdentity():** Function to select the matrix to which the operations apply by first setting the matrix mode.

- **glEnable(GLenum feature):** enables an OpenGL feature.

- **glDisable():** disables an OpenGL feature.

- **glutInitWindowSize(Xsize,Ysize)** and **GlutInitWindowPosition(h,w) :** To set initial window size and position of the window.

- **glutDisplayFunc(display)** : Registers display callback with the window system. Here the function named display will be called whenever the window system determines that the opengl window needs to be redisplayed.

- **glViewport(int x,int y,GLsizei width,GLsizei height):** specifies a width * height viewport in pixels whose lower left corner is at (x,y) measured from the origin of the window.

- **glClearColor():** Function used to clear the window.

- **glFlush():** Empties all of the buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.

- **glutReshapeFunc(void *f(int width,int height):** registers the reshape callback function returns the height and width of the new window. The reshape callback invokes a display callback.

- **glutTimerFunc(unsigned int msecs, void (*func)(int value), value)) :**This finction registers the timer callback func to be triggered in at least msecs milliseconds. The value parameter to the timer callback will be the value of the value parameter to glutTimerFunc. Multiple timer callbacks at same or differing times may be registered simultaneously.

- **glShadeModel(GLenum mode):** GL primitives can have either flat or smooth shading. Smooth shading, the default, causes the computed colors of vertices to be interpolated as the primitive is rasterized, typically assigning different colors to each resulting pixel fragment. Flat shading selects the computed color of just one vertex and assigns it to all the pixel fragments generated by rasterizing a single primitive. In either case, the computed color of a vertex is the result of lighting if lighting is enabled, or it is the current color at the time the vertex was specified if lighting is disabled.

- **glPolygonMode(GLenum face, GLenum mode):** This function controls the interpretation of polygons for rasterization. face describes which polygons mode applies to: both front and back-facing polygons (GL_FRONT_AND_BACK). The polygon mode affects only the final rasterization of polygons. In particular, a polygon's vertices are lit and the polygon is clipped and possibly culled before these modes are applied.

- **glutAttachMenu(int button):** attaches a mouse button for the current window to the identifier of the current menu.

- **glutAddMenuEntry(char *name, int value):** This Function adds a menu entry to the bottom of the current menu. The string name will be displayed for the newly added menu entry. If the menu entry is selected by the user, the menu's callback will be called passing value as the callback's parameter.

- **glPixelStorei(GLenum pname, GLint param):** This function can be used to set any pixel store parameter. Boolean parameters are set to false if param is 0 and true otherwise. Where pname specifies the symbolic name of the parameter to be set and param specifies the value that pname is set to.