**CHAPTER 1**

**PREAMBLE**

**1.1 Introduction**

There were many interesting events in past, one of them was tragedy of titanic. The ship and iceberg collision led to the shrinkage of the huge ship along with many crew members and passengers. This ship sinked on 15th April 1912 due to the collision with iceberg. The ship weighing 52k tonnes on its maiden voyage scheduled to go from Southampton to New York city. The ship sank and around 70% of the people were lost. The opengl project illustrates the ship and iceberg collision.

**1.2 Statement of problem**

The tragedy of titanic is represented via graphics using opengl commands. The ship and iceberg collision that lead to the shrinkage of the ship. This shrinkage of the ship is depicted using the graphics.

**1.3 Introduction to Computer Graphics**

Computer graphics are graphics created using computers and, more generally, the representation and manipulation of pictorial data by a computer. The development of computer graphics has made computers easier to interact with and better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized the animation and video game industry.

The term computer graphics includes almost everything on computers that is not text or sound. Today nearly all computers use some graphics and users except to control their computer through icons and pictures rather than just by typing.

The term Computer Graphics has several meanings:

* The representation and manipulation of pictorial data by a computer.
* The various technologies used to create and manipulate such pictorial data.

* The images so produced, and the subfield of the computer science which studies methods for digitally synthesizing and manipulating visual content, see study of computer graphics.

Today computers and computer-generate images touch many aspects of our daily life. Computer imagery is found on television, newspaper, weather reports, and surgical procedures. A well constructed graph can present complex statistics in a form that is easier to understand and interpret. Such graphics are used to illustrate papers, reports, theses, and other presentation materials. A large range of tools and facilities are available to enable users to visualize their data, and computer graphics are used in many disciplines.

**1.3.1 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 photographs, drawings, graphs, diagrams, typography, numbers, symbols, geometric maps, engineering drawing, or other images. Graphics often combine text, illustrations, and colour. Graphic design may consist of the deliberate selection, creation or arrangement of typography alone, as in a brochure, flier, poster, web site, or book without any other element. Clarity or effective communication maybe the objective, association with other cultural elements may be sought or merely, the creation of the distinctive style.

**1.3.2 Study of Computer Graphics**

The study of computer graphics is a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Although the term often refers to three-dimension computer graphics, it also encompasses two-dimensional graphics and image processing.

As an academic discipline, computer graphics studies the manipulation of visual and geometric information using computational techniques. It focuses on the mathematical and computational foundations of image generation and processing rather than purely aesthetic issues. Computer graphics is often differentiated from the field of visualization, although the two fields have many similarities.

**1.3.3 Applications**

* Computational biology
* Computational physics
* Computer-aided graphics
* Digital art
* Education
* Graphic design
* Info graphics
* Information visualization
* Scientific visualization
* Video games
* Web design

**1.4 Overview of OpenGL:**

OpenGL is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of function calls which can be used to draw complex two-dimensional and three-dimensional scenes from simple primitives. The header file <GL/glut.h> is included in order to utilize the inbuilt functions needed in the implementation; C++ programming language is used for the project coding.OpenGL serves two main purposes:

* To hide the complexities of interfacing with different 3D accelerators, by presenting the programmer with a single, uniform API.
* To hide the differing capabilities of hardware platforms, by requiring that all implementations support the full OpenGL features set

OpenGL’s basic operation is to accept primitives such as points, lines and polygons, and convert them into pixels. This is done by a graphics pipeline known as the OpenGl state machine. Most OpenGL commands either issue primitives to the graphics pipeline, or configure how the pipeline processes these primitives. Basic functionalities to OpenGL include:

* Rasterized points, lines and polygons as basic primitives.
* A transform and lightening pipeline
* Z-buffering
* Texture mapping
* Alpha blending

**1.5 About OPENGL**

As a software interface for graphics hardware, OpenGl’s main purpose is to render two and three-dimension objects into frame buffer. These objects are described as sequences of vertices or pixels. OpenGL performs several processing steps on this data to convert into pixels to form the final described image in the frame buffer.

**1.5.1 Primitives and Commands**

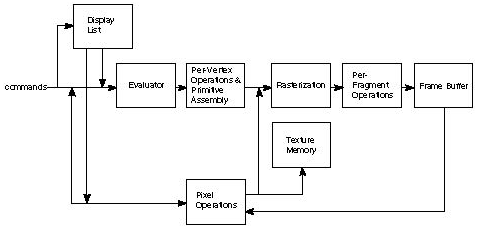
OpenGL draws primitives-points, lines segments, or polygons-subject to several selectable modes. You can control modes independently of each other; that is, setting one mode doesn’t affect weather other modes are set. Primitives are specified, modes are set, and other OpenGL operations are described by issuing commands in the form of function calls. Primitives are defined by a group of one or more vertices. A vertex defines a point, an endpoint of a line, or a corner of a polygon where two edges meet.

Data is associated with a vertex, and each vertex and its associated data are processed independently in order and in the same way. The only exception to this rule is if the group of vertices must be clipped so that a particular primitive fits in a specified region; in this case, vertex data may be modified and new vertices created. The type of clipping depends on which primitive the group of vertices represents.

Commands are always processed in the order in which they are received, although there may be an in terminate delay before a command takes effect. This means that each primitive is drawn completely before any subsequent commands take effect. It also means that state-querying commands return data that’s consistent with complete execution of all previously issued OpenGL commands.

**1.5.2 Basic Block Diagram**

The figure shown below gives an abstract, high-level block diagram of how OpenGL processes data. In the diagram, commands enter from left and proceed through what can be thought of as a processing pipeline. Some commands specify geometric objects to be drawn, and others control how the objects are handled during the various processing stages.



**Figure 1.2: OpenGL Block Diagram**

As shown by the first block in the diagram, rather than having all commands proceeds immediately through the pipeline, you can choose to accumulate some of them in a display list for processing at a later time. The evaluator stage of processing provides an efficient means for approximating curve and surface geometry by evaluating polynomial commands of input values. During the next stage, per-vertex operations and primitive assembly, OpenGL processes geometric primitives-points, line segments, and polygons, all of which are described by vertices. Vertices are transformed and lit, and primitives are clipped to the viewport in preparation for the next stage.

Rasterization produces a series of the frame buffer addresses and associated values using a two-dimensional description of a point, line segment, or polygon. Each fragment so produced is fed into the last stage, per-fragment operations, which perform the final operations on the data before it’s stored as pixels in the frame buffer. These operations include conditional updates to the frame buffer based on incoming and previously stored z-values and blending of incoming pixel colours with stored result of this stage is either stored as texture memory, for use in the rasterization stage, or rasterized and the resulting fragments merged into the frame buffer just as if they were generated from geometric data. All elements of OpenGL state, including the contents of the texture memory and even of the frame buffer, can be obtained by an OpenGL application.

**1.5.3 OpenGL Contributions**

* It is very popular in the video games development industry where it competes with Direct3D (on Microsoft Windows).
* OpenGL is also used in CAD, virtual reality, and scientific visualization programs.
* OpenGL is very portable. It will run for nearly every platform in existence, and it will run well. It even runs on windows NT 4.0 etc. The reason OpenGL runs for so many platforms is because of its Open Standard.
* OpenGL has a wide range of features, both in its core and through extensions. Its extension feature allows it to stay immediately current with new hardware features, despite the mess it can cause.
* Because the ARB is made up of a diverse group of companies, the features available in OpenGL represent a wide range of interests, and thus make it useful in many different applications.

**1.5.4 Weakness**

Though they are very powerful, they do make code messy, very much so at times. They also make it confusing with any compiler that doesn’t often reference tracking. The worst part is, many newer extensions are completely card or vendor specific. Although useful for testing a graphics card’s abilities, vendor-specific extensions are not frequently used by commercial applications.

**CHAPTER 2**

**REQUIREMENT SPECIFICATION**

Requirements Specification is a complete description of the behaviour of the system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the document also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation.

**2.1 Minimum Hardware Requirements**

The Hardware requirements are very minimal and the program can be run on most of the machines.

Processor: Intel Core i3

Processor Speed: 2.93GHz

RAM: 4GB

Storage Space: 1TB

Monitor Resolution: 1024\*768 or 1336\*768 or 1280\*1024

**2.2 Minimum Software Requirements**

Operating System: Windows/LINUX

IDE Microsoft Visual C++

OpenGL Libraries

* **Header Files**
  + - glut.h
    - windows.h
* **Object File Libraries**
  + - glu
    - glut
    - gl
* **DLL files(Dynamic Link Libraries)**
  + - glu32.dll
    - glut32.dll
    - opengl32.dll

**CHAPTER 3**

**SYSTEM DESIGN DESCRIPTION**

System design phase follows system analysis phase. Design is maintaining record proof design divisions and providing a blueprint for the implementation phase. Design is the bridge between system analysis and system implementation. System design is transition from user oriented, document oriented to programmers or database personnel. The design is a solution, a “how to” approach to creation of a new system.

This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the requirement analysis. Design goes through logical and physical stages of development; logical design reviews the present physical system, prepares input and output specifications, details the implementation plan, and prepares a logical design walkthrough.

**3.1** **Objectives of Design**

System design is like a blueprint for a building, it specifies all the features that are to be finished product. Design states how to accomplish objectives determined in the analysis phase.

**3.1.1 Logical Design**

The design of an information system produces the details that state how a system will meet the requirements identified during system analysis. This is Logical Design.

**3.1.2 Physical Design**

The process of developing program software is referred to as Physical Design. In this state the logical design elements are specified which support the business activities. The physical design ensures the system features to meet the user requirements. According to the physical design the software are implemented.

**CHAPTER 4**

**IMPLEMENTATION**

**4.1** **Header files**

* **glut.h**

GL is a fundamental OpenGL library. It provides functions that are permanent part of OpenGL. The functions start with character ‘gl’.

GLUT, the GL utility tool kit supports developing and managing menus, and managing events. The functions start with characters ‘glut’.

GLU, the GL utility library provides high-level routines to handle certain matrix operations, drawing of quadric surfaces such as sphere and cylinders. The functions start with characters ‘glu’.

* **Windows.h**

Windows.h is a windows-specific header file for a c and c++ programming language which contains declaration for all functions in the windows API, all common macros used by windows programmers, and all the datatypes used by various functions and subsystems. It defines a very large number of windows specific functions that can be used in c.

The Win32 API can be added to a C programming project by including the <windows.h> header file and linking to the appropriate libraries. To use functions in xxxx.dll, the program must be linked to xxxx.lib. some headers are not associated with a dll but with a static library.

**4.2 Description of the Functions Used**

**glutDisplayFunc ():** glutDisplay sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window in call.

**4.3 Pseudocode of Implementation**

**4.3.1 MODULE 1**

* void main ():
* This function is used to initialize the glut with c and call the keyboard, mouse, reshape and display function and to keep looping the main function with the help of glutMainLoop().
* void Init ();
* This is a initialization function that initializes the 3D buffer and sets up the initial window size and position including the name of the window.
* void display ():
* This function is used to display the called function in the display screen and to make the ship move and collide with the iceberg and make it shrink.

**4.3.2 MODULE 2**

* void water ():
* This function is used to represent the water that is behind the ship.
* void water1 ():
* This function is used to represent the water in front of the ship.
* void iceberg ():
* This function is used to draw the iceberg on the sea to which the ship will collide.
* void ship ():
* This function is used to draw the ship on the sea. This ship is a 3D ship, it moves towards the iceberg when the display function is called, by incrementing the values of the x and y coordinates.
* void moon ():
* This function is used to draw the moon in the sky.
* void cloud ():
* this function is used to draw multiple clouds in the sky.

**CHAPTER 5**

**TESTING**

**5.1 Testing Process**

Testing is an integral part of software development. Testing process decides whether the product compiles the standard that it was designed to. It includes building of test cases against which the test cases have to be tested. Testing helps to archive quality in the product that is being designed. In some cases, testing is done based on the system requirements for the product which is to be developed.

**5.2 Objective of Testing**

The main objective of testing are as follows,

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

**5.3 Unit Testing**

The product built is a collection of modules. Unit testing focuses efforts on each of these individual modules. A unit is a smallest testable part of any software. It usually has a few inputs but only one output. In procedural programming, a unit may be an individual program, function, procedure, etc.

**5.4 Integration Testing**

This is the second level in testing. Here the modules are combined together and are tested as a sub-system. The goal is to check is all modules can be integrated together and run. It occurs after unit testing and before validation testing.

* **Negative Test case of water ():**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function name | Input | Expected output | Error | Resolved |
| water () | Absence of void water (); in the display function. | Display screen except the water. | Absence of the function call in the display function. | Include void water () in the display function. |

Table 5.1(a)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function name | Input | Expected output | Error | Resolved |
| iceberg () | Coordinates of the iceberg to be drawn. | Displays the iceberg on the screen. | Missing ‘}’ in the function. | Add the missing ’}’ and hence the iceberg is displayed. |

Table 5.1(b)

* **Positive Test case of water ():**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function name | Input | Expected output | Error | Resolved |
| ship () | Coordinates of the ship to be drawn. | Displays the ship and increments the coordinates to make it move towards the iceberg. | No error. | -- |

Table 5.2(a)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function name | Input | Expected output | Error | Resolved |
| Water () | Coordinates of the iceberg to be drawn. | Water that is behind the ship is displayed when the ship moves front towards colliding the iceberg. | No error. | **--** |

Table 5.2 (b)

**CHAPTER 6**

**CONCLUSION**

It was great challenge for us to use OpenGL for the development of our project. With this newly developed software. This project is implemented using OpenGl, and its library functions. The further enhancements can be made to this project by giving sound effects to this project and also adding some sailors in the ship.

**BIBLIOGRAPHY**

[1] Computer Graphics with OpenGL Fourth Edition, Donald D. Hern, M. Pauline Baker

[2] [www.opengl.org/resources/code/samples/redbook](http://www.opengl.org/resources/code/samples/redbook)

[3] www.opengl/overview/readerview

**APPENDIX A**

**SNAPSHOTS OF THE PROJECT**