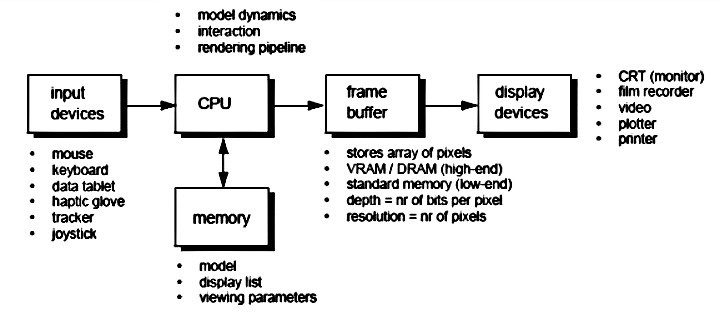
**CHAPTER 1**

**INTRODUCTION**

## HISTORY OF COMPUTER GRAPHICS,VISUALIZATION

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. We can create images that are in distinguishable from photographs of real objects. In other terms, Computer Graphics are the graphics created by the computers and more generally, the representation and manipulation of image data by a computer. The development of computer graphics has been driven both by the needs of the user community and by advances in hardware and software. Typically, the term Computer Graphics refers to several different things.

* The representation and manipulation of image data by acomputer.
* The various technologies used to create and manipulate images.
* The images so produced, and manipulating visual content

#### Figure1.1: Computer Graphics System

**Visualization**

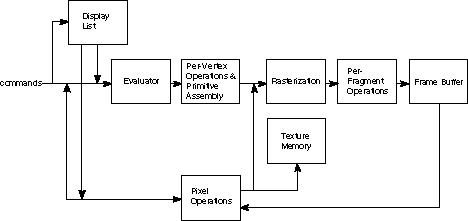
Visualization is any technique for creating images, diagrams, or animations to communicate a message. Visualization through visual imagery has been an effective way to communicate both abstract and concrete ideas since the dawn of a man. Examples from history include Egyptian hieroglyphs, Greek geometry for engineering and scientific purposes. Visualization today has ever-expanding applications in science, education, engineering (Ex: product visualization), interactive multimedia, medicine, etc. Typical of a visualization application is the field of computer graphics.

* The invention of computer graphics may be the most important development in visualization since the invention of central perspective in the Renaissance period. The development of animational so helped advance visualization. The use of visualization to present information is not a new phenomenon. It has been used in maps, scientific drawings, and data plots for over a thousand years. Computer graphics has from its beginning been used to study scientific problems.
* The recent emphasis on visualization started in 1987 with the special issue of Computer Graphics on Visualization in Scientific Computing. Since then there have been several conferences and workshops, co-sponsored by the IEEE Computer Society and ACM SIGGRAPH, devoted to the general topic, and special areas in the field, for example volume visualization. Scientific visualizations are computer generated images that’s how real spacecraft in action, out in the void far beyond Earth, or on other planets.

## INTRODUCTION TO OPENGL

OpenGL(Open Graphics Library) 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 over 250 different function calls which can be used to draw complex three- dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. 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.

**Features of OpenGL:**

* Geometric Primitives allow you to construct mathematical descriptions of objects.
* Viewing and Modeling permits arranging objects in a 3=dimensional scene, move our camera around space and select the desired vantage point for viewing the scene to be rendered.
* Materials lighting OpenGL provides commands to compute the color of any point given the properties of the materials and the sources of the light in the room
* Transformations: rotation, scaling, perspectives in 3D,etc.

#### Fig1.2: OpenGL block diagram

As shown by the first block in the diagram, rather than having all commands proceed immediately through the pipeline, you can choose to accumulate some of them in a *display list* for processing at a later time.

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 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*.

Input data can be in the form of pixels rather than vertices. Such data, which might describe an image for use in texture mapping, skips the first stage of processing described above and instead is processed as pixels, in the *pixel operations* stage. The 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.

**1.3 SCOPE OF THE PROJECT**

This project is designed and implemented using OpenGL interactive application that basically deals with providing the graphical interfaces between user and system. The mini project “OpenGL Utility Toolkit” is the implementation of the geometrical built-in functions. The geometrical objects are drawn with different colors and are oriented as solid as well as wired objects. They are subjected to different transformations, to demonstrate Scaling, Translating and Rotation operations in all directions. We use mouse interface to perform select operation among the menu (list of choices). We will introduce keyboard interaction to rotate, zoom in, zoom out, reset, full screen, exit and move the objects in all direction from program output in various snapshots.

**1.4 DEFINITION**

**1.4.1 Existing problem statement**

The model for interpretation of OpenGL commands is client-server. An application (the client) issues commands, which are interpreted and processed by OpenGL (the server). The server may or may not operate on the same computer as the client. In this sense, OpenGL is network-transparent. A server can maintain several GL *contexts*, each of which is an encapsulated GL state. A client can connect to any one of these contexts. The required network protocol can be implemented by augmenting an already existing protocol or by using an independent protocol. No OpenGL commands are provided for obtaining user input.

The effects of OpenGL commands on the frame buffer are ultimately controlled by the window system that allocates frame buffer resources. The window system determines which portions of the frame buffer OpenGL may access at any given time and communicates to OpenGL how those portions are structured. Therefore, there are no OpenGL commands to configure the frame buffer or initialize OpenGL. Frame buffer configuration is done outside of OpenGL in conjunction with the window system; OpenGL initialization takes place when the window system allocates a window for OpenGL rendering.

**1.4.2 Proposed problem statement**

Placing an object into your scene, in a 3D OpenGL environment can often be quite the achievement when you have everything in place. But just placing an object in your scene can be a little boring, so to get some character movement, or object interaction, it is necessary to both translate and rotate objects, which OpenGL makes quite simple for us to do.

Behind the extremely simple OpenGL calls that we are going to use, lays some pretty intense mathematics which I am not going to go into during these tutorials. In short, it all revolves around matrices and there is a matrix known as a transformation matrix. By multiplying together different transformation matrices, you get your final transformation. Now don’t get transformation confused with translation though, a translation moves the position of an object while a transformation is a combination of translate and rotate.

**1.4.3 Design Plan of Proposed problem statement**

This project “**Interactive object module using openGL”** is developed using OpenGL. It shows Creation, Transformation and Shadowing of Objects.

When you run this program

* Left mouse button shows the Menu.
* Keyboard buttons control the Movement, size and rotation of objects.

The main features provided are:

* Creation of Objects
  + - * Wired form
      * Solid form
* Transformation of Objects
  + - * Translation
      * Rotation
      * Scaling
* Shadows of Objects.

The program has modes for resetting the scaled objects and supporting full screen.

**1.5 Flow of report**

* In the introduction chapter a brief summary and an introduction to the project is mentioned.
* The body of the report is carefully structured in a way that leads the reader through the issue. The body is split into sections such as literature survey where it contains the reason of the project relating to themes or areas for consideration.
* The functional and non-functional requirements and software and hardware specifications are mentioned in the requirements and specifications chapter.
* The project architecture, design and implementation is briefly mentioned.
* The various test results such as unit testing,system testing, and other test cases are mentioned under the chapter testing.
* The graphical representation of the project is represented in the Snapshots chapter
* The conclusion sets out what inferences you draw from the information, including any experimental results.

**1.6 Chapter summary**

The Introduction: introduces the topic of the report in context and explains the problem and/or motivation for the projectalso states the aim/s of the project which indicates the purpose of the report briefly outlines the report structure.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 WHY INTERACTIVE OBJECT MODULE ?**

This project is designed and implemented using OpenGL interactive application that basically deals with providing the graphical interfaces between user and system. The mini project “OpenGL Utility Toolkit” is the implementation of the geometrical built-in functions. The geometrical objects are drawn with different colors and are oriented as solid as well as wired objects. They are subjected to different transformations, to demonstrate Scaling, Translating and Rotation operations in all directions. We use mouse interface to perform select operation among the menu (list of choices). We will introduce keyboard interaction to rotate, zoom in, zoom out, reset, full screen, exit and move the objects in all direction from program output in various snapshots.

**2.2 INTRODUCTION TO INTERACTIVE OBJECT MODULE**

This project “**Interactive object module using openGL”** is developed using OpenGL. It shows Creation, Transformation and Shadowing of Objects. When you run this program,

* Left mouse button shows the Menu.
* Keyboard buttons control the Movement, size and rotation of objects.

The main features provided are:

* Creation of Objects

1. Wired form

2. Solid form

* Transformation of Objects

1. Translation

2. Rotation

3. Scaling

* Shadows of Objects.

The program has modes for resetting the scaled objects and supporting full screen.

**OPENGL PROGRAMMING LANGUAGE:**

As a software interface for graphics hardware, OpenGL's main purpose is to render two- and three-dimensional objects into a frame buffer. These objects are described as sequences of vertices (which define geometric objects) or pixels (which define images). OpenGL performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer.

### Execution Model

The model for interpretation of OpenGL commands is client-server. An application (the client) issues commands, which are interpreted and processed by OpenGL (the server). The server may or may not operate on the same computer as the client. In this sense, OpenGL is network-transparent. A server can maintain several GL *contexts*, each of which is an encapsulated GL state. A client can connect to any one of these contexts. The required network protocol can be implemented by augmenting an already existing protocol or by using an independent protocol. No OpenGL commands are provided for obtaining user input.

The effects of OpenGL commands on the frame buffer are ultimately controlled by the window system that allocates frame buffer resources. The window system determines which portions of the frame buffer OpenGL may access at any given time and communicates to OpenGL how those portions are structured. Therefore, there are no OpenGL commands to configure the frame buffer or initialize OpenGL. Frame buffer configuration is done outside of OpenGL in conjunction with the window system; OpenGL initialization takes place when the window system allocates a window for OpenGL rendering.

**2.3 CHAPTER SUMMARY**

A literature survey or a literature review in this project report is the section which shows the various analyses and research made in the field of your interest and the results already published, taking into account the various parameters of the project and the extent of the project.

**CHAPTER 3**

**REQUIREMENT AND SPECIFICATIONS**

* The system requirement and specification of our project is as follows:

**3.1 FUNCTION REQUIREMENT**

**3.1.1 Purpose of the requirements document**

A requirements document is a [document](https://en.wikipedia.org/wiki/Document) containing all the requirements to a certain product. It is written to allow people to understand *what* a product should do.

Purpose and [scope](https://en.wikipedia.org/wiki/Scope_(project_management)), from both a technical and business perspective.

* Product overview and [use cases](https://en.wikipedia.org/wiki/Use_case)
* [Requirements](https://en.wikipedia.org/wiki/Requirement), including
* [functional requirements](https://en.wikipedia.org/wiki/Functional_requirement) (e.g. what a product should do)
* [usability](https://en.wikipedia.org/wiki/Usability) requirements
* technical requirements (e.g. security, network, platform, integration, client)
* environmental requirements
* support requirements
* interaction requirements (e.g. how the product should work with other systems)
* Assumptions
* Constraints
* Dependencies

**3.1.2 Hardware Requirements**

* Processor : Intel corei3 or AMD processor
* RAM : 2GB Recommended
* Hard Disk : upto 500GB
* Keyboard : Standard 101 key Keyboard
* Mouse(ps/2),Printer (optional)

**3.1.3 Software Requirements**

* Operating system : 64bit Windows 10
* Language Tool : OpenGL
* Compiler : CC Compiler /C++ Compiler.
* Libraries : Supporting OpenGL glut.h, gl.h & glu.h
* Code blocks

**3.2 Non-functional requirement**

**3.2.1 Practicality**

Project practicality is where you use only what is viewed as least needed to meet your goals.A communication system like tis needs hundreds of thousands of users to survive and thrive. Therefore, it should be designed to support large numbers of users, e.g., a substantial percent of a town or a campus.

**3.2.2 Efficiency**

Efficiency measures how well and productively a manger uses his resources to achieve goals. Project management places heavy focus on how to acquire the right project team to perform project tasks and to close project successfully within the agreed constraints.

**3.2.3 Cost**

Cost management is concerned with the process of planning and controlling the budget of a project or business. It includes activities such as planning, estimating, budgeting, financing, funding, managing, and controlling costs so that the project can be completed within the approved budget. Cost management covers the full life cycle of a project from the initial planning phase towards measuring the actual cost performance and project completion.

**3.2.4 Flexibility**

The need for flexibility is to deal with changed circumstances. Flexibility is used to scale back activities needing less effort while diverting resources to areas with unexpected problems.

**3.2.5 Modularity**

Modularity refers to the concept of making multiple [modules](http://www.defit.org/?p=74) first and then linking and combining them to form a complete system. Modularity enables re-usability and minimizes duplication. In addition to re-usability, modularity also makes it easierto fix problems as bugs can be traced to specific system modules, thus limiting the scope of detailed error searching

**3.2.6 Extensibility**

Extensibility is a [software engineering](https://en.wikipedia.org/wiki/Software_engineering) and [systems design](https://en.wikipedia.org/wiki/Systems_design) principle where the implementation takes future growth into consideration. The term extensibility can also be seen as a systemic measure of the ability to extend a [system](https://en.wikipedia.org/wiki/System)and the level of effort required to implement the extension. Extensions can be through the addition of new functionality or through modification of existing functionality.

**3.2.7 Reliability**

Reliability refers to the probability and or the likelihood that a given product will perform in the way and or manner it was intended to perform in the efforts that have been deemed required of that given product within or under a specific period of time required.

**3.2.8 Maintainability**

It is the probability that a system or system element can be repaired in a defined environment within a specified period of time. Increased maintainability implies shorter repair times

**3.2.9 Portability**

It is a measure of how easily an application can be transferred from one computer environment to another. A computer software application is considered portable to a new environment if the effort required to adapt it to the new environment is within reasonable limits. The phrase "to port" means to modify software and make it adaptable to work on a different computer system.

### 3.3 Chapter Summary

A software requirements specification is a description of a [software system](https://en.wikipedia.org/wiki/Software_system) to be developed. The software requirements specification lays out [functional](https://en.wikipedia.org/wiki/Functional_requirement) and [non-functional](https://en.wikipedia.org/wiki/Non-functional_requirements) [requirements,](https://en.wikipedia.org/wiki/Non-functional_requirements) It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.