**VISVESVARAYA TECHNOLOGICAL UNIVERSITY BELGAUM-590014, KARNATAKA**



**A MINI PROJECT REPORT**

ON

**“A FCFS Algorithm”**

Submitted in partial fulfillment of requirements for the award of 6th semester,

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

Submitted By:

**Atul Das**

**(USN: 1MJ13CS034)**

Under the Guidance of

**Mrs. Deepti Singhal**

**Assistant Professor,**

**Department of Computer Science & Engineering**

****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**MVJ COLLEGE OF ENGINEERING**

**BENGALURU-67**

**MVJ COLLEGE OF ENGINEERING**

**Channasandra, Near ITPB, Bangalore-67**

**Department Of Computer Science and Engineering**

****

***Certificate***

This is to certify that the mini project entitled **"FCFS Algorithm”** is a bonafide work carried out by **Atul Das (1MJ13CS034)** a bonafide student of **MVJ College of Engineering** in partial fulfillment for the award of degree of Bachelor of Engineering in Computer Science & Engineering of the Visvesvaraya Technological University, Belgaum during the year 2015-16. It is certified that all the corrections/suggestions indicated for Internal Assessment have been incorporated in the Report. The mini Project Report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said degree.

**Signature of Guides**

Signature of the Guide Signature of HOD

(Mrs. Deepti Singhal) (Dr. M. Pauline)

**Signature of Examiners**

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our effort with success.

I express my sincere gratitude to the **MVJ College of Engineering** for providing facilities.

I wish to place on record my grateful thanks to  **Dr. M. Pauline**, Head of the Department, Computer Science and Engg, MVJ College of Engineering, Bangalore for providing encouragement and guidance.

I consider it a privilege and honour to express my sincere gratitude to my guide **Mrs. Shoba K.M.** and **Mr. Srikanth S.** CSE Department of Computer Science & Engineering for their valuable guidance throughout the tenure of this seminar work , and whose support and encouragement made this work possible.

I wish to thank the faculty of CS&E department whose suggestions have enabled me to surpass many of the seemingly impossible hurdles.

Thank you.

**ABSTRACT**

The project aims at demonstrating and illustrating the FCFS or the First Come First Served Algorithm for the process scheduling. The objective is to draw the Gantt Chart given the Arrival Time and the Burst Time of the process in the queue.

A Gantt Chart is the graphical depiction of the process scheduling according to their arrival in the process queue and their burst time or time of completion.

The visual will consist of a Gantt Chart which will show the process bars scale and translate to the time of their completion. It will also calculate and display the average wait and average turnaround time.

This will be implemented using Visual C++ 6.0 and OpenGL software. Various features of OpenGL like coloring, spinning, rotation and transformations will be used.

CONTENTS

CHAPTERS PAGE NO'S.

Chapter 1 Introduction 1

1.1 Problem statement 1

1.2 Objective of the project 3

1.3 Scope of the project 3

1.4 Summary 3

Chapter 2 Literature Survey 4

2.1 Main features of the project 4

2.2 Technical overview 4

2.3 Summary 5

Chapter 3 Requirement Specification 6

3.1 Functional requirements 7

3.2 Non-functional requirements 8

3.3 Details of the software 8

3.4 Software Requirements 9

3.5 Hardware Requirements 9

Chapter 4 Design 10

Chapter 5 Implementation 14

Chapter 6 Testing and debugging 20

6.1 Test plans 20

Chapter 7 Screen shots 24

Conclusion 26

BIBLIOGRAPIIY 27

**LIST OF FIGURES**

**Fig Number Fig Name Chapter No Page No**

1.1 Application Programmers Model of 1 2

Graphics System

1.2 Library Organization 4 5

4.1 Henry Ford Assembly Line Approach 4 10

6.1.1 Executed Command Prompt 7 24

6.1.2 Gantt Chart 7 23

**LIST OF TABLES**

**Table Number Table Name Chapter No Page No**

6.1 Test of source code 6 - 21

6.2 Compilation of Source Code 6 - 22

6.3 Mouse Function 6 -22

6.4 Creation of application 6 -23

**CHAPTER 1**

**INTRODUCTION**

Computer Graphics is concerned with all aspects of producing pictures or images using a computer- A particular graphics software system called OpenGL, which has become a widely accepted standard for developing graphics applications. The applications of computer graphics in some of the major areas are as follows

l. Display of information.

2. Design.

3. Simulation and Animation.

4. User interfaces.

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that are used to specify the objects and operations needed to produce interactive applications.

This project named "FCFS ALGORITHM" uses OpenGL software interface and develops a Gantt chart. Simple techniques like rotation, transformation, motion, lighting, etc have been used.

**1.1 PROBLEM STATEMENT**

Computer graphics is no longer a rarity. It is an integral part of all computer user

Interfaces and is indispensable for visualizing 2D, 3D and higher dimensional objects.

In the proposed system, the OpenGL is an graphic software system designed as a

streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, you must work through whatever windowing system controls the particular hardware you're using.

OpenGL doesn't provide high-level commands for describing models of three dimensional objects. Such commands might allow you to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules. With OpenGL, you must build up your desired model from a small set of geometric primitives - points, lines, and polygons.

Key board

Application Program

Graphics library (API)

Drivers

Mouse

Display

**Fig1.1 Application programmers model of graphics system**

The interface between an application program and a graphics system can be specified through a set of functions the resides in a graphics library .These specification are called the application programmer's interface (API).The application programmer see only the API and is thus shielded from the details of both the hardware and the software implementation of the graphics library. The software drivers are responsible for interpreting the output of the API and converting this data to a form that is understood by the particular hardware.

**1.2 OBJECTIVES OF THE PROJECT**

The objective of this project is to demonstrate the user how this program interacts with the system and renders the graphic design of rotating objects with a trail following them.

The other objectives of this project are:

* It makes use of interactive programming.
* It provides features of transformations such as translation, rotation, and scaling.
* It makes use of light source and uses concepts of orthogonal and perspective views.

**1.3 SCOPE OF THE PROJECT**

The application program developed can be used in various fields as follows:

* In the field of computer science engineering to demonstrate the working of fcfs algorithm.
* A number of real time process scheduling explanation tool can be designed based on this concept.
* Educational purpose as a teaching aid.

**1.4 SUMMARY**

This chapter deals with the first phase of development of the project by knowing the drawbacks of the existing computer graphics and proposing a new system .To determine the objectives and scope of the project.

**CHAPTER -2**

**LITERATURE SURVEY**

Fcfs algorithm is the first come first served scheduling algorithm. With this scheme the process that requests the CPU first is allocated the CPU or memory space first. The implementation of the fcfs policy is easily managed with the fifo queue. When a process enters the ready queue its PCB is linkefd on to the tail of the queue. When the CPU is free it is allocated to the process at the head of the queue. The running process is then removed from the queue. The code for fcfs scheduling is simple to write and understand.

**2.1 MAIN FEATURES OF THE PROJECT**

* The application is user friendly.
* It is represented in 3D.
* It serves as an educational aid
* Enhanced by the inclusion of user interaction functions.

**2.2. TECHNICAL OVERVIEW**

Most of our applications will be designed to access OpenGL directly through functions in three libraries. Firstly functions in the main GL library begin with the letters ***gl*** and are stored in the library usually referred to as ***GL .***The second is the OpenGL utility library ***(GLU)*** and the Third is the OpenGL utility toolkit (***GLUT)***, which provides the minimum functionality that should be expected in any modern windowing system.

In GLUT, our application structures use event handling to use callback functions. (This method is similar to using the Xt Toolkit, also known as the X Intrinsics, with a widget set.) For example, first you open a window and register callback routines for specific events. Then, you create a main loop without an exit. In that loop, if an event occurs, its registered callback functions are executed. Upon completion of the callback functions, flow of control is returned to the main loop.



Geometric data (vertices, lines, and polygons) follows the path through the row of boxes that include evaluators and per-vertex operations, while pixel data (pixels, images, and bitmaps) is treated differently for part of the process. Both types of data undergo the rasterization and per-fragment operations before the final pixel data is written into the frame buffer.

All the data, whether geometry or pixels, can be saved in a display list or processed immediately. When a display list is executed, the data is sent from the display list just as if it were sent by the application.

All geometric primitives are eventually described by vertices. If evaluators are used, that data is converted to vertices and treated as vertices from then on. Vertex data may also be stored in and used from specialized vertex arrays. Per-vertex calculations are performed on each vertex, followed by rasterization to fragments. For pixel data, pixel operations are performed, and the results are either stored in the texture memory, used for polygon stippling, or rasterized to fragments. Finally, the fragments are subjected to a series of per-fragment operations, after which the final pixel values are drawn into the frame buffer.

**CHAPTER 3**

**REQUIREMENT SPECIFICATION**

A software requirement definition is an abstract description of the services which the system should provide, and the constraints under which the system must operate. It should only specify the external behavior of the system. The requirements are specified as below:

**3.1 FUNCTIONAL REQUIREMENTS:**

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases.

The various methods used in this project are as follows:-

**Init light-** The module sets the OpenGL light variables.

**Display-** The module draws the output on the screen and the functions in it

**Menu-** This module specifies the action corresponding to menu entry.

**Mouse Movement-** The module specifies the rotation with the mouse and able to see the menu.

**Idle-** This module is used to display the object more times using some delay t.

**Visible-** This module is used to make object visible to the viewer.

**Motion-**This module helps in placing the process tiles in Gantt Chart.

**3.2 NON-FUNCTIONAL REQUIREMENTS:**

These are constraints on the services or functions offered by the system.

They include timing constraints, constraints on the development process and standards. Nonfunctional requirements often apply to the system as a whole. The Non-Functional Requirements are as follows:-

**3.2.1 Dependability**

The dependability of a computer system is a property of the system that equates to its trustworthiness. Trustworthiness essentially means the degree of user confidence that the system will operate as they expect and that the system will not 'fail' in normal use.

**3.2.2 Availability**

It is the ability of the system to deliver services when requested. There is no error in the program while executing the program.

**3.2.3 Reliability**

The ability of the system to deliver services as specified. The program is compatible with all types of operating system without any failure.

**3.2.4 Safety**

It is the ability of the system to operate without catastrophic failure. This program is user friendly and it will never effects the system

**3.2.5 Security**

It is the ability of the system to protect itself against accidental or deliberate intrusion.

**3.3 DETAILS OF THE SOFTWARE**

Here, the coding of the project is done in Microsoft Visual Studio which is a commercial integrated development environment (IDE) with OpenGL (Open Graphics Library) which is a standard specification to produce 2D and 3D computer graphics.

The OpenGL Utility Toolkit called GLUT which is a library of utilities for OpenGL programs is also used.

**3.3.1 Microsoft Visual C++**

Microsoft Visual g++ is a commercial integrated development environment (IDE) product engineered by Microsoft for the C, C++ and C#/CLI programming languages. It has tools for developing and debugging C++ code, especially code written for the Microsoft Windows API, OpenGL API, the DirectX API and the Microsoft .NET Framework.

**3.3.2 OpenGL and GLUT**

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, describing a set of functions and the precise behaviors that they must perform. From this specification, hardware vendors create implementations - libraries of functions created to match the functions stated in the OpenGL specification, making use of hardware acceleration where possible. Hardware vendors have to meet specific tests to be able to qualify their implementation as an OpenGL implementation.

GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing

Programs in OpenGL. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT provides a portable API so you can write a single OpenGL program that works across all PC and workstation OS platforms.

**3.4 SOFTWARE REQUIREMENTS**

* OpenGL(Graphics Library Utility Toolkit)
* Microsoft Visual C++ 6.0
* Operating System- Windows XP and above

**3.5 HARDWARE REQUIREMENTS:**

* Hard Disk - 30 GB and Above
* Memory - 256 MB and Above
* Mouse - 3 button mouse
* Keyboard
* Visual Display Unit

**CHAPTER 4**

**DESIGN**

Graphics systems used general - purpose computers with the standard von Neumann architecture. Such computers are characterized by a single processing unit that processes a single instruction at a time. Information had to be sent to the display at a rate high enough to avoid flicker on the display. In the early days of computer graphics, computers were so slow that refreshing even simple images, containing a few hundred line segments, would burden an expensive computer.

The following diagram shows the Henry ford assembly line approach which OpenGL takes to process data.



**4.1 Display Processors:**

Built special-purpose graphics systems were concerned primarily with relieving the general purpose computer from the task of refreshing the display continuously. These display processes had conventional architectures but included instructions to display primitives on the CRT. The main advantage of the display processor was that the instructions to generate the image could be assembled once in the host and sent to the display processor, where they were stored in the display processor's own memory as a display list, or display file. The display processor would then repetitively execute the

program in the display list, at a rate sufficient to avoid flickering independently of the host, thus freeing the host for other tasks. This architecture has become closely associated with the client-server architectures.

**4.2 Pipeline Architectures:**

The major advances in graphics architectures closely parallel the advances in work stations. In both cases, the ability to create special-purpose VLSI chops was the key enabling technology development. In addition, the availability of inexpensive solid state memory led to the universality of raster displays. For computer-graphics applications, the most important use of customs VLSI circuits has been in creating pipeline architectures.

**4.3 The graphics pipeline:**

3 major steps in the imaging process:

* **Vertex processing**
* **Clipping and primitive assembly**
* **Fragment processing**

**4.3.1 Vertex Processing:**

In the first block of our pipeline, each vertex is processed independently. The two major functions of the block are to carry out coordinate transformations and to compute a color for each vertex. The assignment of vertex colors can be as simple as the program specifying a color or as complex as the computation of a color from a physically realistic lighting model that incorporates the surface properties of the object and the characteristic light sources in the scent.

**4.3.2 Clipping and primitive assembly:**

The second fundamental block in the implementation of the standard graphics pipeline is for clipping and primitive assembly. We must do clipping because of the limitation that no imaging system can see the whole world at once. The human retina has a limited size corresponding to an approximately 9O-degree field of view. Cameras have a film of limited size, and we can adjust their fields of view by selecting different lenses.

**4.3.3 Rasterization**.

The primitives that emerge from the clipper are still represented in terms of their vertices and must be further processed to generate pixels in the frame buffer. For example, if three vertices specify a triangle filled with a solid color, the raster must determine which pixels in the frame buffer are inside the polygon. The output of rasterization is a set fragment for each primitive. A fragment can be thought of as a potential pixel that carries with it information, including its color and location, that is used to update the corresponding pixel in the frame buffer. Fragment can also carry along depth information that allows later stages to determine if a particular fragment lies behind other previously rasterized fragments for a given pixel.

**4.3.4 Fragment Processing:**

The final block in our pipeline takes in the fragments generated by the rasterizer and updates the pixels in the frame buffer. If the application generated three-dimensional data, some fragments may not be visible because the surfaces that they define are behind other surfaces. The color of the pixel that corresponds to a fragment can also be read from buffer and blended with the fragment's color to create translucent effects.

**4.4 CONTROLS**

The controls provided in the project are given below:

**MOUSE CONTROLS**

* RIGHT MOUSE: Restarts the whole program..
* LEFT MOUSE: Exits from the tutorial.

**KEYBOARD CONTROLS**

* Number keys to key in the burst time, arrival time & also the number of processes.

**CHAPTER 5**

**IMPLEMENTATION**

* **glPushMatrix**

**Syntax:** glPushMatrix( );

**Purpose**: It saves the present values of attributes and matrices by placing or pushing them on top of the appropriate stack.

* **glPopMatrix**

**Syntax:** glPopMatrix( );

**Purpose:** It recovers the attributes and matrices by removing or popping them from the stack.

* **glloadldentity**

**Syntax**: glloadldentityQ;

**Purpose:** It loads the identity matrix.

* **glMatrixMode**

**Syntax:** a) glMatrixMode(GL\_PROJECTION);

b) glMatrixMode(GL\_MODEL\_VIEW);

**Purpose:** It helps in selecting the matrix(model view or projection matrix) to which the operations apply by setting the matrix mode a variable that is set to one tlpe of matrix and also part of the state.

* **glOrtho**

**Syntax:** glorthd Gldouble left, Gldouble right, Gldouble bottom, Glxlouble top, Gldouble near, Gldouble far);

**Purpose**: It gives an orthogmphic projection with a right parallelepiped viewing volume. All the parameters me distances measured from the camera.

* **glBegin**

**Syntax:** glBegin(glEnum mode);

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

* **glEnd**

**Syntax:** glEnd0;

**Purpose:** Terminates a list of vertices.

* **glutlnit**

**Syntax**: glutlnit( int \*argc, char \*\* argv);

**Purpose:** Initializes GLUT. The arguments from main are passed

In and can be used by the application.

* **glVertex**

**Syntax:**a)glVertex[23a][sifd( TYPE xcoordinate, TYPE ycoordinate, .);

b) glVertex[B4][sifd]v(TYPE \*coordinate);

**Purpose:** Specifies the position of a vertex inZ,3,or 4 dimensions. The coordinates can be specifie as short s, int i, float f or double d. If the v is presant, the argument is a pointer to an array containing the coordinates.

* **glutCreateWindow**

**Syntax:** glutCreateWindow(char \* title);

**Purpose:** It creates a window on the display. The string title can be used to label the window.

* **glutlnitDisplayMode**

**Syntax**: glutlnitDisplayMode( unsigned int mode);

**Purpose:** It requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model(GLUT\_RGB, GLUT\_INDEX) and buffering(GLUT\_SINGLE, GLUT\_DOUBLE).

* **glutMainLoop**

**Syntax:** glutMainlLoop0;

**Purpose:** It causes the program to enter an event processing loop. It should be the last statement in main.

* **glutKeyboardFunc**

**Syntax:** glutKepoardFunc(void \*(char key, int width, int height)

**Purpose:** Registers the keyboard callback function f. The callback function returns the ASCtr code of the key pressed and the position of the mouse.

* **glutSpecialFunc**

**Syntax:** glutSpecialFunc( void(\*f)(unsigrred char key,int x, int y))

**Purpose:** Registers the mottion keys such as (->,<-)callback function The position of the mouse is retumed by the callback when the mouse is moved at least one of thesekeys pressed.

* **glEnable**

**Syntax :**glEnable(Glenum feature)

**Purpose:** Enables an OpenGL feature. Features that can be enabled include GL DEPTHJEST, GL-LIGHTING, GI LIGHTi, GL\_LINE\_SMOOTH, GL\_BLEND, GL\_NORMALVE etc.

* **glutMotionFunc**

**Syntax:** void glutMotionFunc(void(\*func)(int x,int y));

**Purpose**: The motion callback for a window is called when the mouse

moves within the window while one or more mouse buttons

are used.

* **glColor**

**Syntax**: glColor3[b i f d ub us ui](TYPE r, TYPE g, TYPE b);

**Purpose**:Sets the present RGB colors. Valid types are byte(b), int(i), float(f), double(d), unsigrred byte(ub), unsigrred short(us), and unsigned int(ui).

* **glRasterPos**

**Syntax:** glRasterPos[234]sifd(TYPE xcoord, TYPE ycoord, - . ..);

**Purpose:** Specifi es a raster position.

* **glNewList**

**Syntax:** void glNewList(GLuint list,GLenum mode);

**Purpose**: Specifies the display list name, the compilation mode to be

used.

* **glTranslate**

**Synatx:** glTranslate[fd( TYPE x, TYPE y, TYPE z);

**Purpose:** Alters the current matrix by a displacement of (x,y,z).

* **glRotate**

**Syntax:** glRotatef[fd](TYPE angle, TYPE dx, TYPE dy, TYPE dz);

**Purpose:** Alters the current matrix by a rotation angle of degrees about the axis(dx,dy,dz).

* **glLight**

**Syntax:** gllight[if]v(Glrnum light, Glenum param, Type \*v);

**Purpose:** Sets scalar and vector parameter param for light source 'light'.

* **glPolygonMode**

Syntax: glPolygonMode(glEnum faces, glEnum mode);

Purpose: Sets the desired mode for polygon rendering the faces (GL\_FRONT, GL\_BACK GL\_ERONT\_AND\_BACK), mode can be GL\_POINTS, GL\_LINES, or GL\_FILL.

* **GlutSwapBuffers**

**Syntax :** glut SwapBuffers0 ;

**Purpose:** Swaps the front and back buffers.

* **glFlush**

**Syntax:** glFlush0;

**Purpose:** Forces any buffered OpenGL commands to execute.

* **glViewport**

**Syntax:** glViewPort(int x, int y, Glsizei width, Glsizei height);

**Purpose:** Specifies a width\*height viewport in pixels whose lowerJeft corner is at (x,y) measured from the origin of the window.

* **glutPostRedisplay**

**Syntax :** glutPostRedisplayQ

**Purpose:** Requests that the display callback be executed after the current callback returns.

* **gluOrtho2D**

**Syntax:** gluOrtho2D(Gldouble left, Glxlouble right, Gldouble bottom, Glclouble top);

**Purpose:** Defines a two dimensional viewing rectangle in the plane Z=0

* **glutldleFunc**

**Syntax**: glutldleFunc(void (\*f[void));

**Purpose:** Registers the display callback function f that is executed whenever there are no other events to be handled.

* **glutKeyboardFunc**

**Syntax**: glutKeyboardFunc(void \*f(char key, int width, int height);

**Purpose:** Registers the keyboard callback function f. The callback function returns the ASCII code of the key pressed and the position of the, mouse.

**CHAPTER 6**

**TESTING**

Once source code has been generated, software must be tested to uncover as many errors as possible before delivery to the customer. Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and code generation.

**6.1 TEST PLANS**

In this test plan all major activities are Unit test, Integration test, Validation test, System test.

**6.1.1 Unit Testing**

Unit testing focuses verification effort on the unit of software design or module. using the unit test plans, prepared in the design phase of the system development as a guide, important control paths are tested to uncover efforts within the boundary of the modules. The inter phase of each of the module were tested to ensure proper flow of the information into and out of the modules under the consideration.

**Table 6.1.1 Test of source code**

|  |  |
| --- | --- |
| **S1# Test Case:** | **1** |
| **Name of Test:** | **Test of Source code** |
| **Item being tested:** | **Source code** |
| **Expected Output:** | **Interface Page** |
| **Actual Output:** | **Interface Page** |
| **Remarks:** | **Pass** |

**6.1.2 Integration testing**

Data can be lost across an interface: a module and sub-functions, when combined may not produce the desired Can present problems. have an adverse effect on another's major function; global data structures can present problems

**Table 6.1.2 Source code compilation**

|  |  |
| --- | --- |
| **S1# Test Case:** | **2** |
| **Name of Test:** | **Compilation of Source Code** |
| **Item being tested:** | **Source code** |
| **Expected Output:** | **Executed Command Prompt** |
| **Actual Output:** | **Executed Command Prompt** |
| **Remarks:** | **Pass** |

**6.1.3 Validation testing**

The whole of validation testing is to expose latent defects in a software system before the system is delivered. This contrasts with validation testing which is intended to demonstrate that a system meets its specification. Validation testing requires the system to perform correctly using given acceptance test cases.

**Table 6.1.3 Mouse function**

|  |  |
| --- | --- |
| **S1# Test Case:** | **3** |
| **Name of Test:** | **Mouse Function** |
| **Item being tested:** | **Mouse Button** |
| **Expected Output:** | **Corresponding action to be executed** |
| **Actual Output:** | **Corresponding action to be executed** |
| **Remarks:** | **Pass** |

**6.1.4 System Testing**

Creation of application .Expected output fcfs.exe with 0 error, 0 warning. Actual output fcfs.exe with 0 error,4 warning.

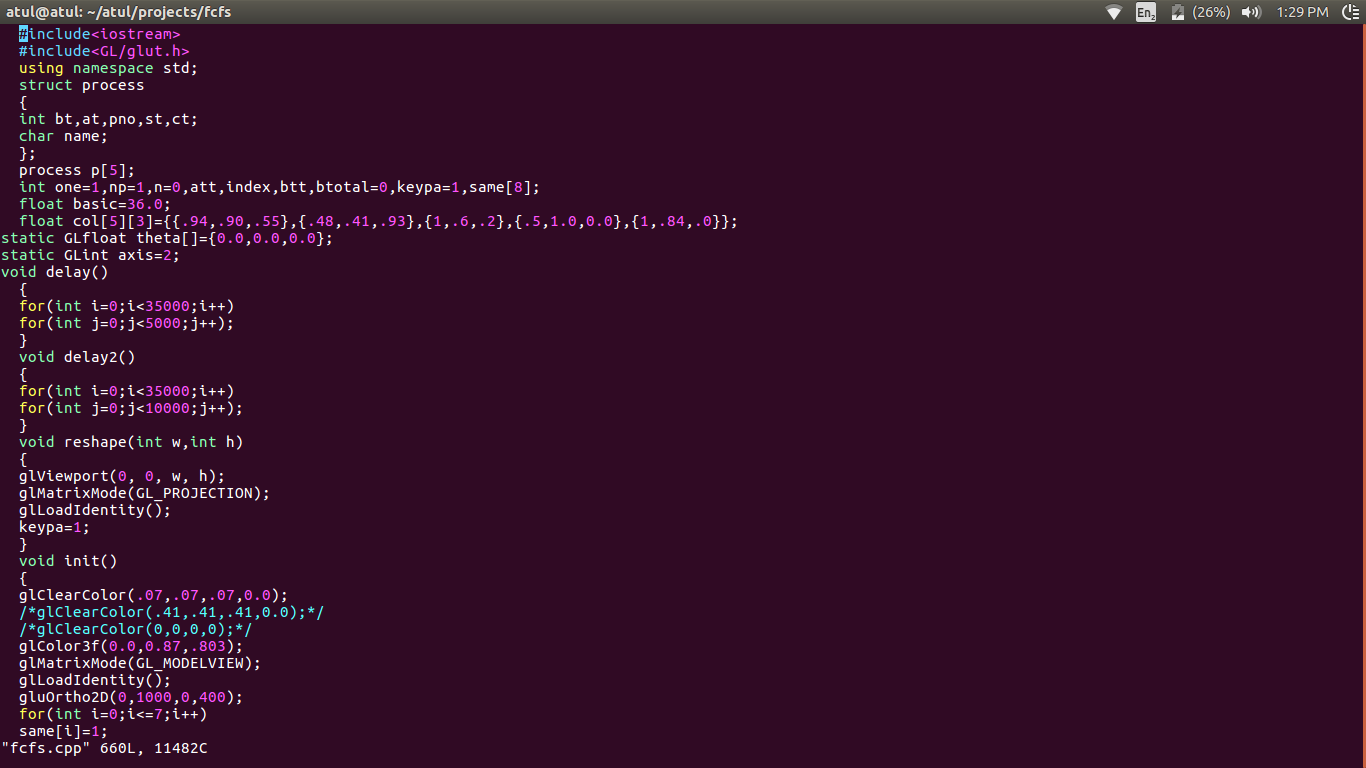
**Table 6.1.4 Creation of application**

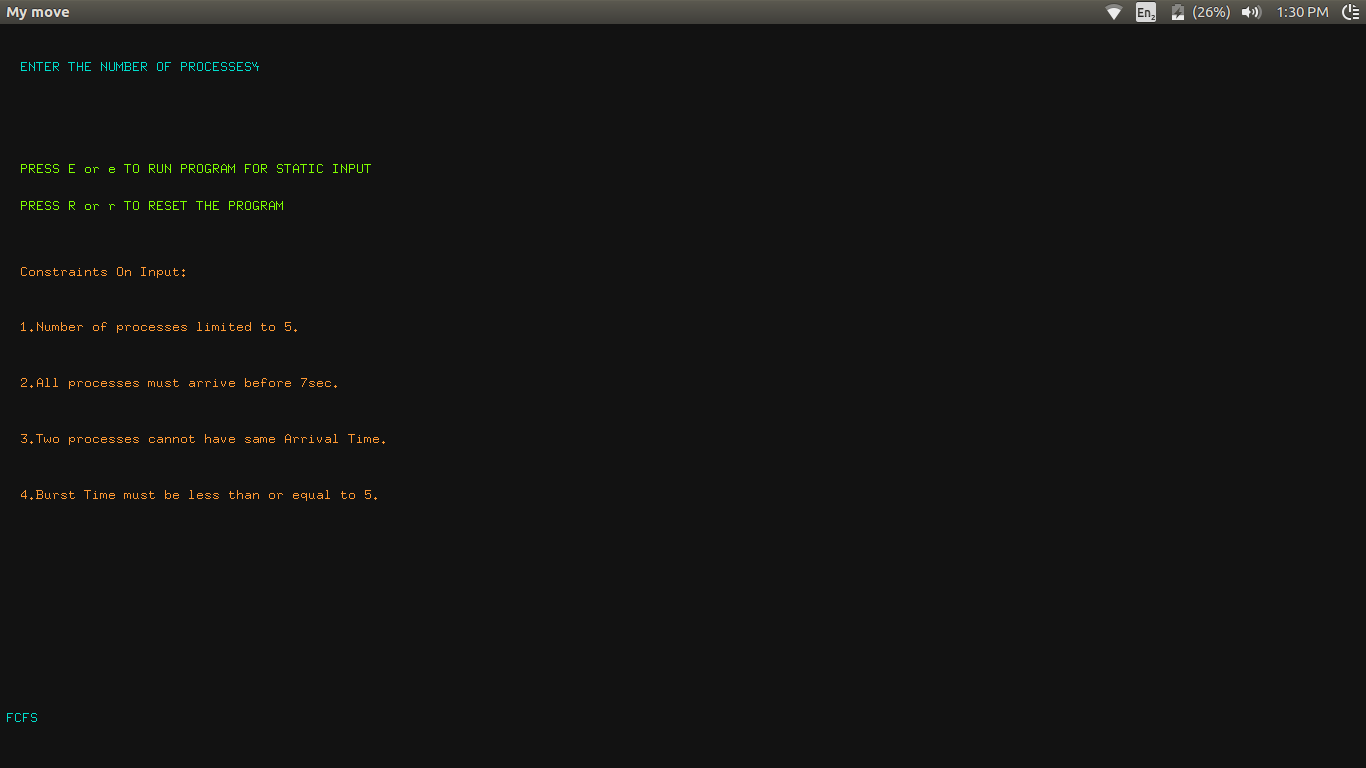
|  |  |
| --- | --- |
| **S1# Test Case:** | **4** |
| **Name of Test:** | **Creation of Application** |
| **Item being tested:** | **Source Code** |
| **Expected Output:** | **Gantt Chart with 0error and 0 warning** |
| **Actual Output:** | **Gantt Chart with 0error and 4 warning** |
| **Remarks:** | **Pass** |

**CHAPTER 7**

**SCREEN SHOTS**

Fig. 1.1 The vim editor used for writing and building the code

Fig. 1.2 The code snippet of the project

Fig. 1.3 The data entry window asking “the number of processes”

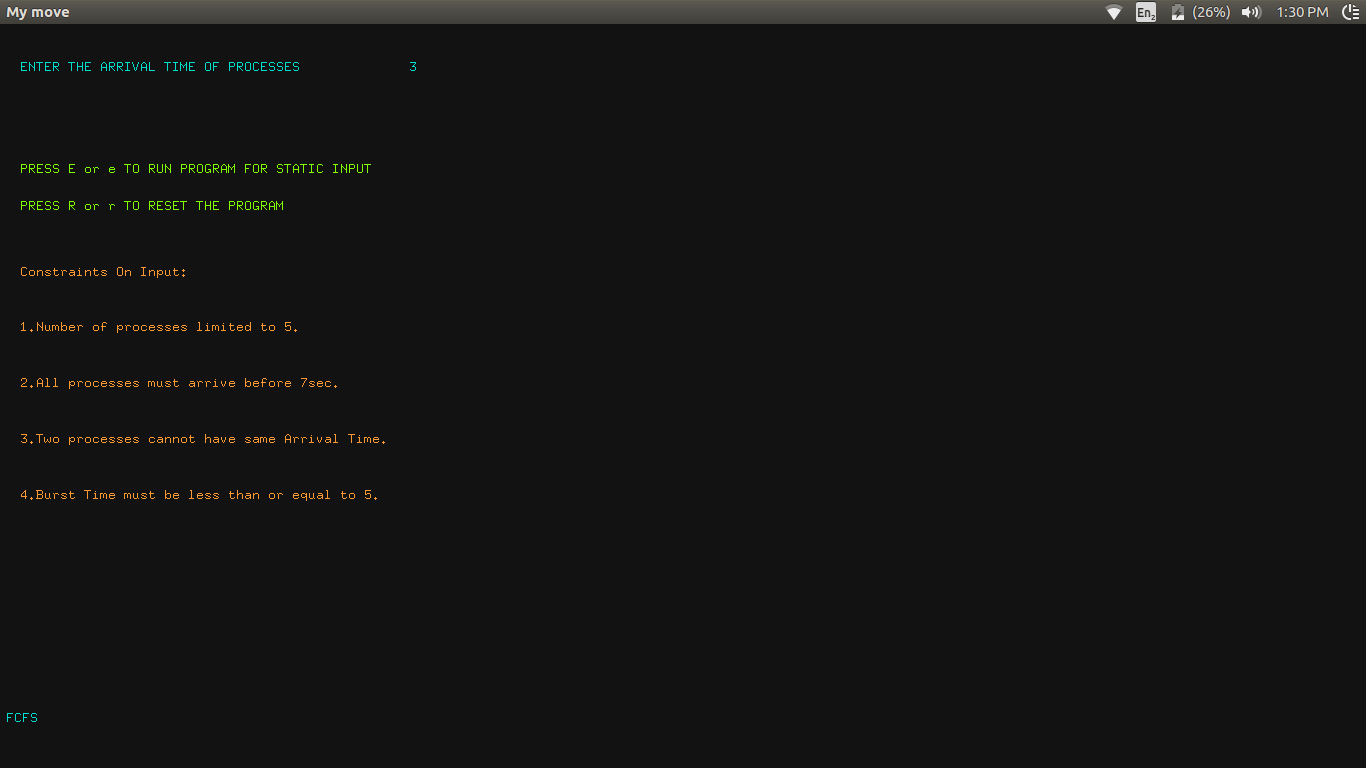
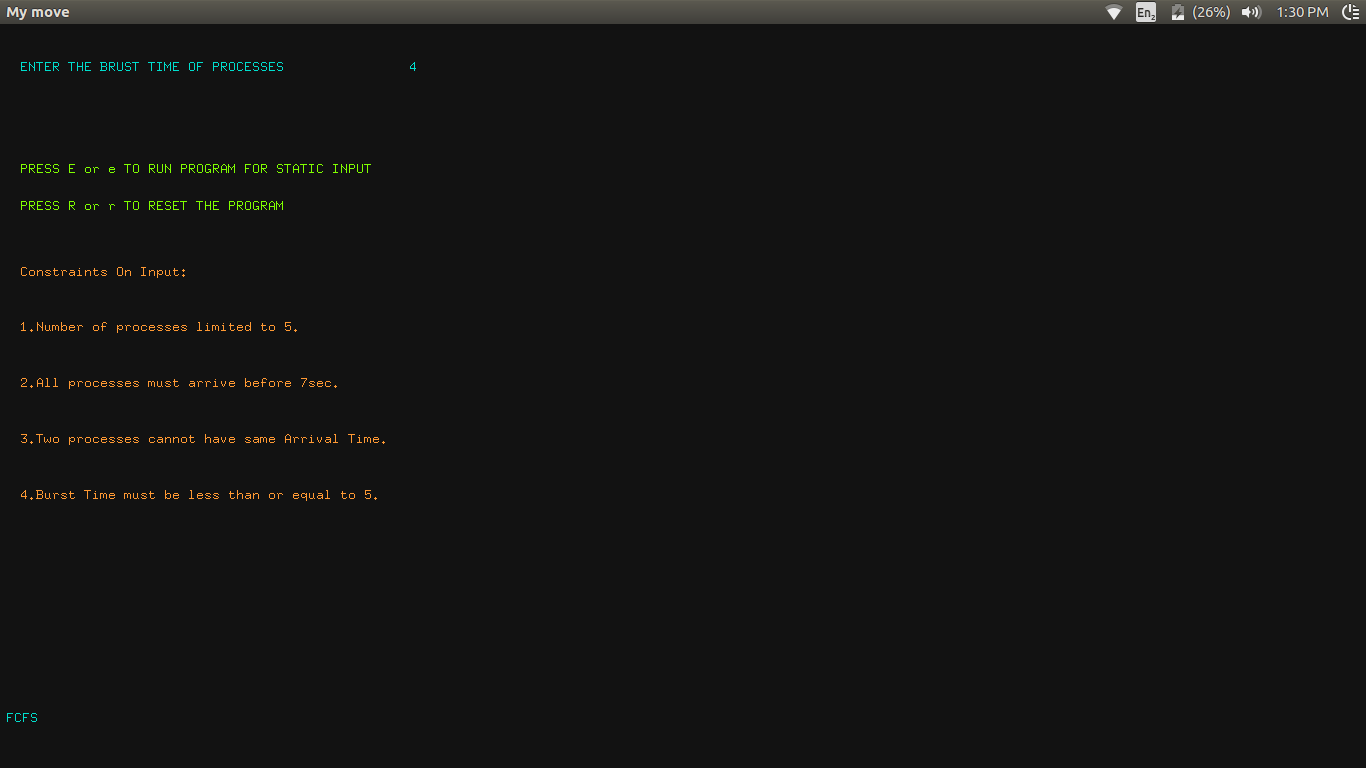
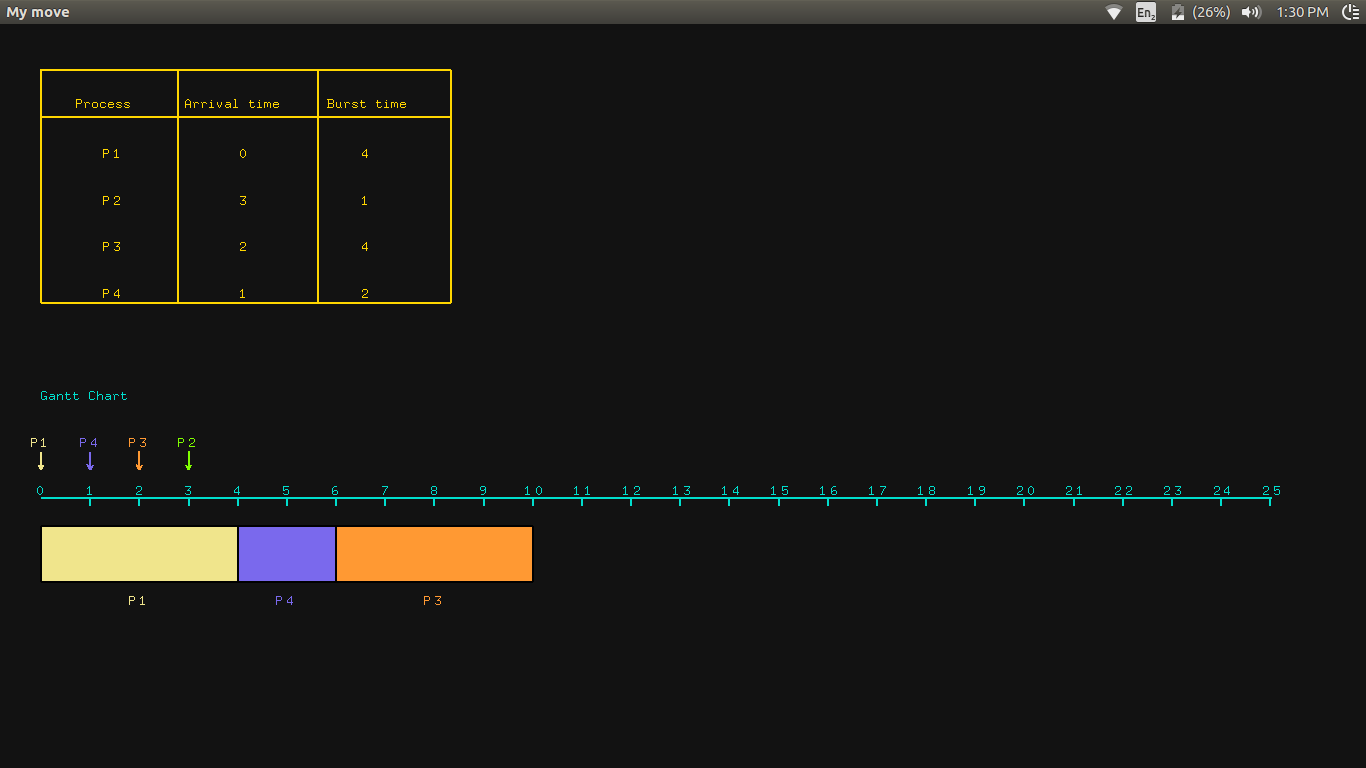
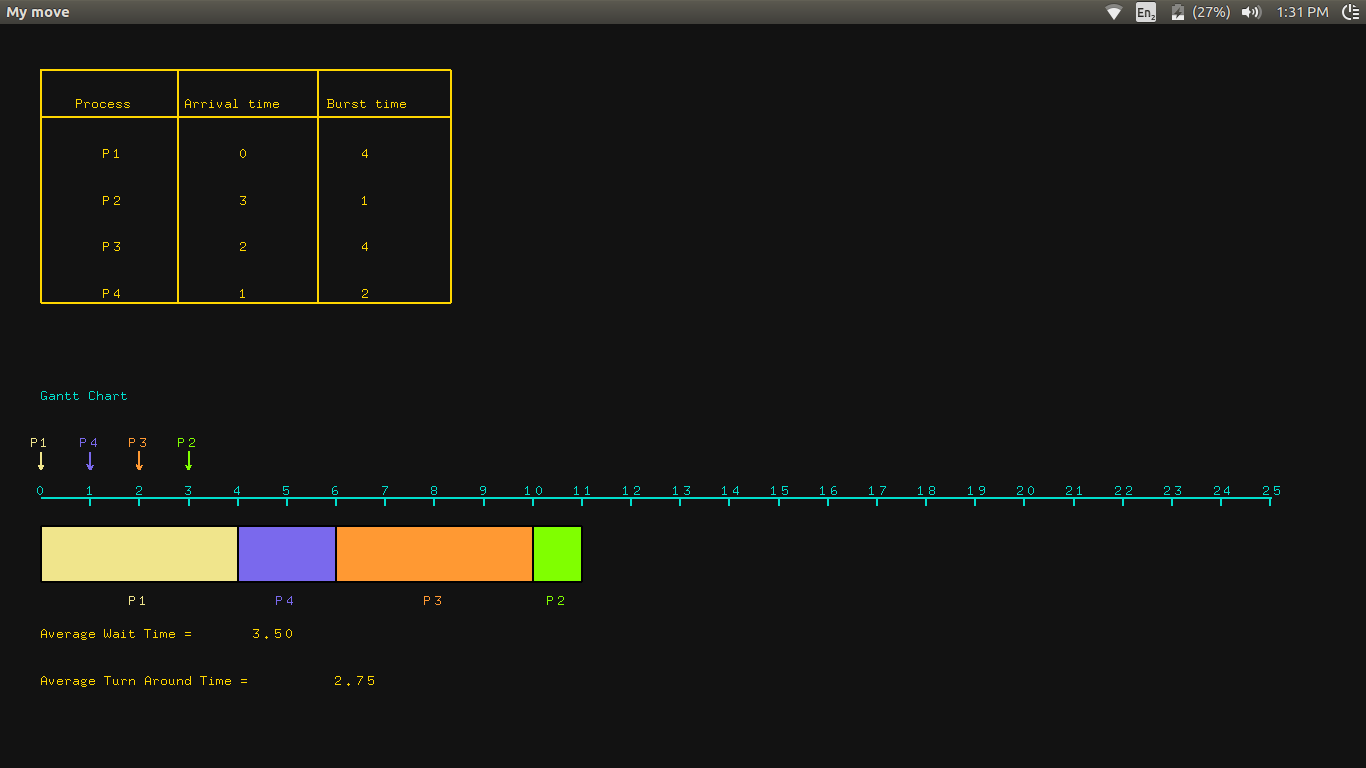
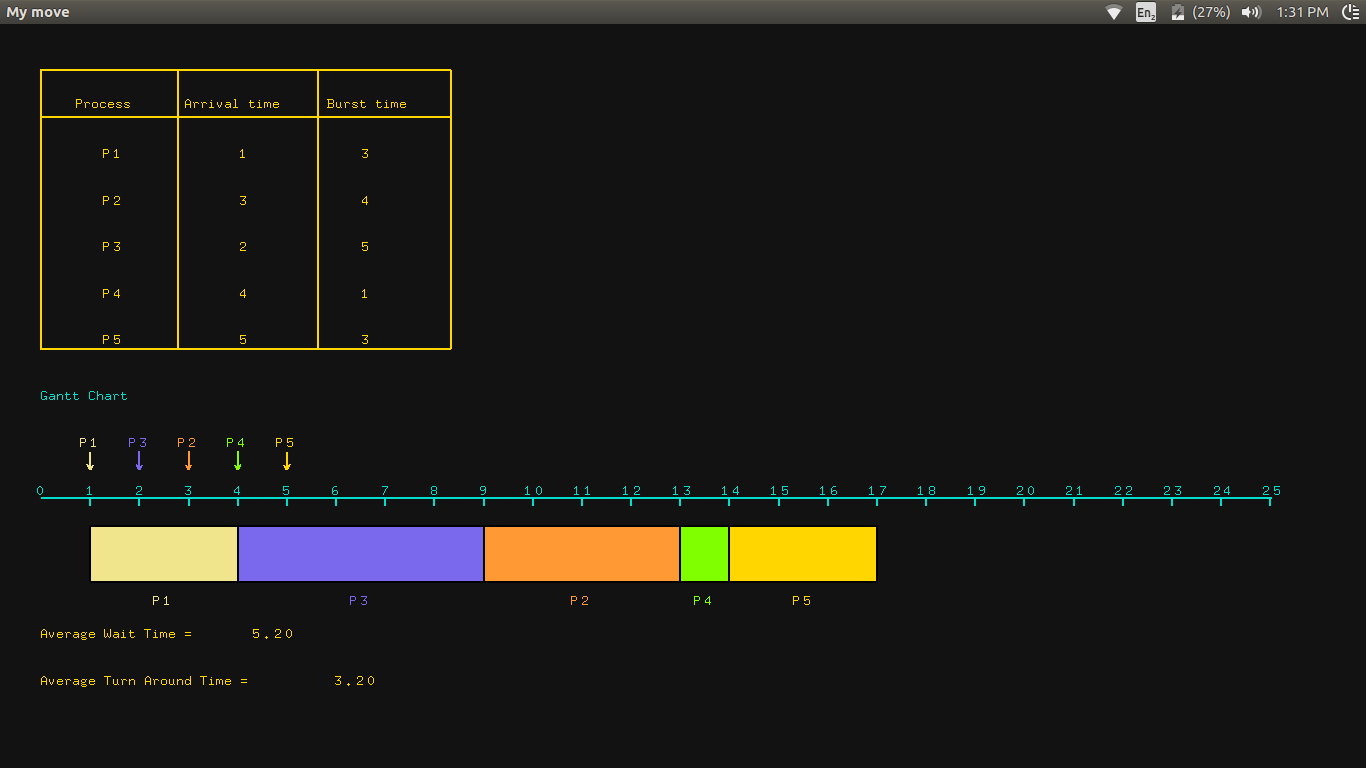
****

Fig. 1.4 The interface to key in the arrival time of processes

Fig. 1.5 Interface to enter the “burst time of the process”

Fig. 1.6 The window displaying the data entered and also the Gantt Chart depiction.

Fig. 1.7 The window displaying the “avaerage wait time” and average turnaround time.

Fig 1.8. Depicts the FCFS Algorithm of the sample static data

**CHAPTER 8**

**CONCLUSION**

In this project, the design and implementation of 3-dimensional graphics Interfacing program has been attempted. We have seen the functions performed by this application and code for the same. In OpenGL, C/C++ provide enormous flexibility in the design and the use of C/C++ graphics programs.

The presence of many in-built OpenGL functions and libraries to take care of many of the functionalities reduces the burden of coding and makes the implementation simpler.

The project started with the designing phase in which we figured the requirements needed the system design, Data flow diagram etc. Then comes the details of the implementation phase where in we have included various functionalities. And now after the testing phase, the project comes to an end.

.

**BIBLIOGRAPHY**

* During the course of this project reference to the following books and materials were made:

[1] Edward Angel, *“Interactive Computer Graphics”*, Pearson Publication, 5th edition

[2] F.S. Hill Jr,”Computer Graphics Using OpenGL”,Pearson Publication,2nd edition

[3] James D. Foley,”Computer Graphics”,Addision Wesley,1997

[4] Abraham Silberchatz,“Operating System Concepts”,Wiley India Publication, 8th Edition

* Internet was extensively browsed for various materials related to this project

[1] [http://www.opengl.org](http://www.opengl.org/)

[2] [http://www.cs.usr.edu](http://www.cs.usr.eduAbraham/)

[3] <http://math.ucsd.edu/~sbuss/MathCG/OpenGLsoft>