CHAPTER 1
INTRODUCTION

1.1 Overview of computer graphics

Suppose you want to draw a picture say, Fish moving inside the water. Suddenly you will get an idea to use paint, but the degree of accuracy, quality of image is not satisfies you and become very sad. So this problem of creating fish moving inside the water can be solved using COMPUTER GRAPHICS without any difficulties.

Computer Graphics became a powerful tool for the rapid and economical production of pictures. There is virtually no area in which Graphical displays cannot be used to some advantage so it is not surprising to find the use of CG so widespread.

Although early application in engineering & science had to relay on expensive & troublesome equipment, advances in computer technology have made interactive computer graphics a practical tool.

Today Computer Graphics is found in diverse area such as science, engineering, medicine, business, industry, government, art, entertainment, education and training. Thus Computer graphics is generalized tool for drawing and creating pictures and simulate the real world situations within a small computer window.

1.1.1 History

William fetter was credited with coining the term Computer Graphics in 1960, to describe his work at Boeing. One of the first displays of computer animation was future world (1976), which included an animation of a human face and hand-produced by Carmull and Fred Parkle at the University of Utah.

There are several international conferences and journals where the most significant results in computer-graphics are published. Among them are the SIGGRAPH and Eurographics conferences and the association for computing machinery (ACM) transaction on Graphics journals.

1.2 Applications of computer graphics

Nowadays Computer Graphics used in almost all the areas ranges from computer science engineering, medicine, business, industry, government, art, entertainment, education and training.

- **CG in the field of CAD**

Computer Aided Design methods are routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft computers, textiles and many other applications.

- **CG in presentation Graphics**

Another major application area presentation graphics used to produce illustrations for reports or generate slides. Presentation graphics is commonly used to summarize financial, statistical, mathematical, scientific data for research reports and other types of reports.2D and 3D bar chart to illustrate some mathematical or statistical report.

- **CG in computer Art**

CG methods are widely used in both fine art and commercial art applications. Artists use a variety of computer methods including special purpose hardware, artist's paintbrush program (lumena), other pain packages, desktop packages, maths packages, animation packages that provide facility for designing object motion. Ex: cartoons decision is an example of computer art which uses CG.

- **Entertainment**

Computer graphics methods are now commonly used in making motion pictures, music, videos, games and sounds. Sometimes graphics objects are combined with the actors and live scenes.

- **Education and Training**

Computer generated models of physical financial, economic system is often used as education aids. For some training application special systems are designed.

Ex: specialized system is simulator for practice sessions or training of ship captain, aircraft pilots and traffic control.

- **Image Processing**

Although the methods used in CG image processing overlap, the 2 areas are concerned with fundamentally different operations.

In CG a computer is used to create picture. Image processing on the other hand applies techniques to modify existing pictures such as photo scans, TV scans.

- **User Interface**

It is common for software packages to provide a graphical interface. A major component of a graphical interface is a window manager that allows a user to display multiple window area. Interface also displays menus, icons for fast selection and processing.

CHAPTER 2

SYSTEM REQUIREMENTS

The following list specifies the hardware and software requirement for the development of our project.

2.1 Hardware Requirement

Processor	:	intel core i5 processor
RAM	:	2GB
Secondary memory	:	250GB

2.2 Software requirement

Programing Language	:	C++
Compiler	:	g++
Operating System	:	Ubuntu 16.04 LTS

CHAPTER 3

INTRODUCTION TO OPENGL

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications.

OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms.

Most of the application will be designed to access OpenGL directly through functions in three libraries. Functions in the main GL (or OpenGL in windows) library have names that begin with the letters gl and are stored in a library usually referred to as GL (or OpenGL in windows). The second is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects and simplifying viewing. All functions in GLU can be created from the core GL library but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations functions in the GLU library begin with letters glu.

To interface with the window system and to get input from external devices into the programs, at least one more system-specific library is needed that provides the “glue” between the window system and OpenGL. For the X window system, this library is functionality that should b expected is any modern windowing system.

Fig 3.1 shows the organization of the libraries for an X Window System environment. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus can be recompiled with the GLUT library for other window systems.

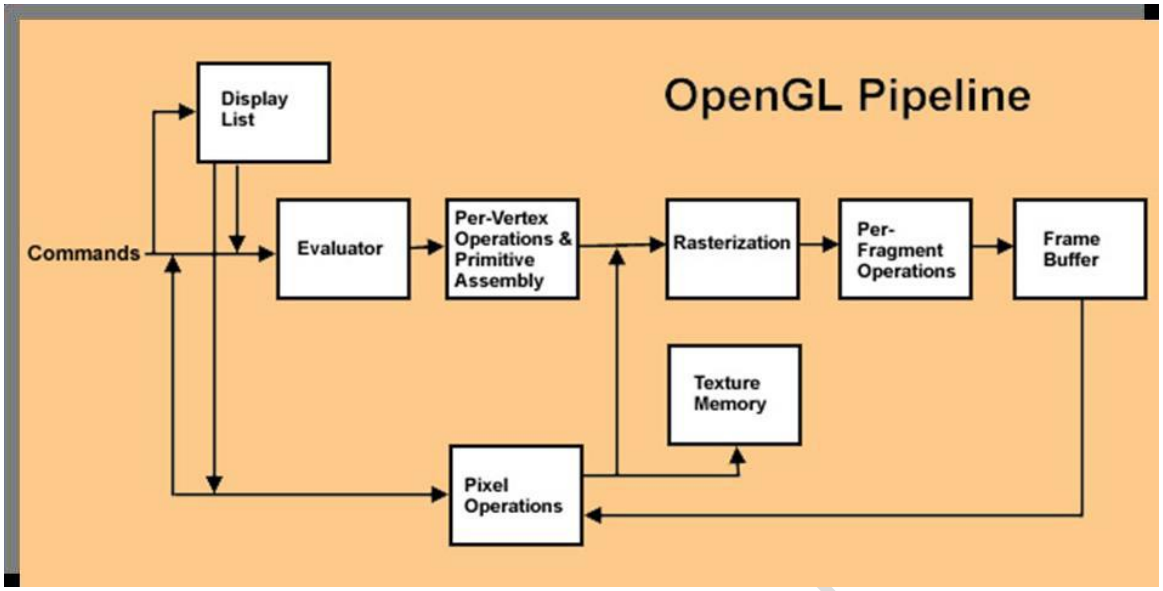


Fig 3.1 Library organization

3.1 OpenGL Command Syntax

OpenGL commands use the prefix **gl** and initial capital letters for each word making up the command name. Similarly, OpenGL defined constant begin with GL_, use all capital letters, and use underscores to separate words (GL_COLOR_BUFFER_BIT).

Some extraneous letters are appended to some command name (for example, the **3f** in **glColor3f ( )** and **glVertex3f ( )**). It's true that the **Color** part of the command name **glColor3f ( )** is enough to define the command as one that sets the current color. However, more than one such command has been defined so as to use different types of arguments. In particular, the **3** part of the suffix indicates that three arguments are given; another version of the **Color** command takes four arguments. The **f** part of the suffix indicates that the arguments are floating point numbers. Having different formats allows OpenGL to accept the user's data.

Some OpenGL commands accept as many as 8 different data types for their arguments. The letters used as suffixes to specify these data types for ISO C implementations of OpenGL are shown in table 2.1 along with the corresponding OpenGL type definitions.

Table 3.1: Command suffixes and argument data types

SUFFIX	DATA TYPE	TYPICAL CORRESPONDING C-LANGUAGE TYPE	OPENGL TYPE DEFINITION
B	8-bit integer	Signed char	GLbyte
S	16-bit integer	Short	GLshort
I	32-bit integer	Int or Long	GLint,GLsizei
F	32-bit floating-point	Float	GLfloat,Glclampf
D	64-bit floating-point	Double	GLdouble,GLclampd
Ub	8-bit unsigned Integer	Unsigned char	GLubyte,Glboolean
Us	16-bit unsigned integer	Unsigned short	GLushort
Ui	32-bit unsigned integer	Unsigned int or Unsigned long	GLuint,GLenum,GLbitfield

3.2 OpenGL as a State Machine

OpenGL is a state machine. It is put into various states (or modes) that then remain in effect until it is changed. The current color is state variable. The current color can be set to white, red or any other color, and thereafter every object is drawn with that color until current color is set to something else.

Many state variable refer to modes that are enabled or disabled with the command **glEnable ( )** and **glDisable ( )**.

3.3 Display Lists

All data, whether it describes geometry or pixel, can be saved in a display lists for current or later use. When a display list is executed, the retained data is sent from the display list just as if it were sent by the application in immediate mode.

3.4 Pixel Operations

Pixels from an array in system memory are first unpacked from one of a variety of formats into the proper number of components. Next the data is scaled, biased, and processed by a pixel map. The results are clamped and then either written into texture memory or sent to the rasterization step. If the pixel data is read from the frame buffer, pixel transfer operations (scale, bias, mapping, and clamping) are performed. Then these results are packed into an appropriate format and returned to an array in system memory.

3.5 Texture Assembly

An OpenGL application may wish to apply texture images onto geometric objects to make them look more realistic. Some OpenGL implementations may have special resources to accelerate texture performance. There may be specialized, high-performance texture memory.

3.6 Rasterization

Rasterization is the conversion of both geometric and pixel data into fragments. Each fragment square corresponds to a pixel in the frame buffer. Line and polygon stipples, line width, point size, shading model, and coverage calculations to support antialiasing are taken into consideration as vertices are connected into lines or interior pixels are calculated for a filled polygon. Color and depth values are assigned for each fragment square.

3.7 Fragment Operations

Before values are actually stored into frame buffer, a series of operations are performed that may alter or even throw out fragments. All these operations can be enabled or disabled.

The operation which may be encountered texturing, where a Texel (texture element) is generated from texture memory for each fragment and applied to the fragment. Then fog calculations may be applied, followed by scissor test, the alpha test, the stencil test, and the depth-buffer test.

3.8 Need of OpenGL

The intension of this chapter is to give you basic concepts in OpenGL. OpenGL is a device and operating system independent library for 3D-graphics and rendering. OpenGL was originally developed by Silicon Graphics Inc. (SGI) for use on their high end graphics workstations. Since then, OpenGL has become a widely accepted standard with implementations on many operating system and hardware platforms including

Windows NT and Windows X operating systems. The purpose of OpenGL library is to render two or Three-dimensional objects into frame buffer. OpenGL is a library of high-quality three-dimensional graphics and rendering functions. The library's device and platform independence make it a library of choice for developing portable graphical applications.

OpenGL drawings are constructed from primitives are simple items such as lines or polygons, which in turn are composed of vertices.

The OpenGL library assembles primitives from vertices while taking into account a variety of settings, such as color, lighting, and texture. Primitives are then processed in accordance with transformations, clipping settings, and other parameters at the end of the rasterization process is pixel data deposited into a frame buffer.

Because of high visual quality and performance any visual computing application requiring maximum performance from 3D animation to CAD to visual simulation-can exploit high-quality, high-performance OpenGL capabilities. These Capabilities allow developers in diverse markets such as broadcasting, CAD/CAM/CAE, entertainment, medical imaging, and virtual reality to produce and display incredibly compelling 2D and 3D graphics.

3.9 Advantage of Using OpenGL

- **Industry standard:** An independent consortium, the OpenGL Architectural Review Board, guides the OpenGL specification. With broad industry support, OpenGL is the only true open, vendor-neutral, multiplatform graphics standard.

- **Stable:** OpenGL implementations have been available for more than seven years on a wide variety of platforms. Additions to the specification are well controlled, and proposed updates are announced in time for developers to adopt changes.

Backward compatibility requirement ensure that existing applications do not become obsolete.

- **Reliable and Portable:** all OpenGL applications produce consistent visual display results on any OpenGL API-compliant hardware, regardless of operating system or windowing system.

- **Evolving:** Because of its thorough and forward-looking design, OpenGL allows new hardware innovations to be accessible through API via the OpenGL extension mechanism. In this way, innovations appear in the API in timely fashion, letting application developers and hardware vendors incorporate new features into their normal product release cycles.

CHAPTER 4

DESIGN AND IMPLEMENTATION

The design and Implementation part is consider with coding of the algorithms and analyzed portion of project.

First Come First Serve Algorithm is OS based Computer graphics project. This Project as name suggest demonstrate the working of First Come First Serve Algorithm or FCFS Algorithm. The First Come First Serve algorithm is one of the scheduling algorithms for process in Operating System. It has the following algorithm. In First Come First Serve the process is executed on first come, first serve basis. This is easy to learn, understand and implement. The performance of this scheduling algorithm is poor as average wait time is high. To execute this program-run the program with CTR+5 (in windows). First page will have introduction written on it. Press E or e to run the computer graphics project with default values. To run with your own values, give the inputs. Like in our case-press 4 first to give no of process. Then input 2,3,6,7 for arrival time. You easily understand and demonstrate the concept in OS-First Come First Serve Algorithm.

4.1 Proposed System

To achieve three dimensional effects, OpenGL software is proposed. It is software which provides a graphical interface. It is an interface between application program and graphics hardware. The advantages are:

- OpenGL is designed as a streamlined**.**
- It is a hardware independent interface, it can be implemented on many different hardware platforms.
- With OpenGL, we can draw a small set of geometric primitives such as points, lines and polygons etc.
- It provides double buffering which is vital in providing transformations.
- It is event driven software.
- It provides call back function.

4.2 Transformation Functions

Scaling:

The scaling operation on an object can be carried out for an object by multiplying each of the points (x,y,z) by the scaling factors sx, sy and sz.

glScalef(sx,sy,sz);

4.3 Graphic Functions and Requirements

The Header files used in OpenGL are,

•**GL** for which the commands begin with GL;

•**GLUT** the GL Utility Toolkit, opens windows, develops menus, and manages events.

4.4 Functions

Following are the different functions that are used,

1. Void glBegin ( ):

Initiates a new primitive of type mode and starts the collection of vertices. Values of mode include GL_POINTS, GL_LINES and GL_POLYGON.

2. Void glEnd ( ):

Terminates a list of vertices.

3. Void glColor3f ():

Sets the present RGB colors. Valid types are int (I), float (f) and double (d). The maximum and minimum values of the floating-point types are 1.0 and 0.0, respectively.

4. Void glClearColor ():

Sets the present RGBA clear color used when clearing the color buffer. Variables of GLclampf are floating-point numbers between 0.0 and 1.0.

5. Int glutCreateWindow ( ):

Creates a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used where there are multiple windows.

6. Void glutInitWindowSize (int width, int height):

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

7. void glutInitWindowPosition (int x, int y):

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

8. void glutInitDisplayMode ( ):

Request a display with the properties in mode. The value of mode is determined by the logical OR of operation including the color model (GLUT_RGB, GLUT_INDEX) and buffering (GLUT_SINGLE, GLUT_DOUBLE).

9. void glFlush ( ):

Forces and buffers any OpenGL commands to execute.

10. void glutInit (intargc, char **argv):

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

11. void glutMainLoop ( ):

Cause the program to enter an event processing loop. It appears at the end of main.

12. void glutDisplayFunc ( ):

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

13. void glClear(GL_COLOR_BUFFER_BIT):

To make the screen solid and white.

14.void glLoadIdentity() sets the current transformation matrix to an identity matrix.

15.void glViewport(intx,inty,GLsizeiwidth,GLsizei height)

Specifies a width * height viewport in pixels whose lower-left corner is at (x,y) measured from he origin of the window.

16.glutSwapBuffers() swaps the front and back buffers.

User defined functions are used to color the curves in a standard cycle rainbow manner which becomes very easy for the user to identify the levels of recursion for the curves.

17.void gluPerscpective(GLdoublefov, GLdouble aspect, GLdouble near, GLdouble far)

Defines a perspective viewing volume using the y direction field of view fov measured in degree,the aspect ratio of the front clipping plane, and the near and far distance.

18.void glutKeyboardFunc(myKey) refers to the keyboard callback function. The function to callback is defined as

voidmyKey(unsigned char key, int x, int y)

19.void glutMouseFunc(myMouse) refers to the mouse callback function. The function to callback is defined as

voidmyMouse(intbutton,intstate,int x)

4.5 Implementation

The whole First Come First Serve is mainly divided into 4 windows including the terminal. The first window shows the introduction page. This window consists of guide name, title of project .

The second window enters into game. It displays the 35 cars parked in a parking area keeping some space for parking a car and one car placed away from the parking area. If right button is clicked in the mouse it shows two option that is house and move car. If we click on house, it displays few houses surrounding the parking area. If we click on move car we can move that single car placed away from the parking area and park it in parking area.If we click on left button it shows camera options and quit option.

For quit the window we have to press q button the window terminates.

CHAPTER 5

SNAPSHOTS

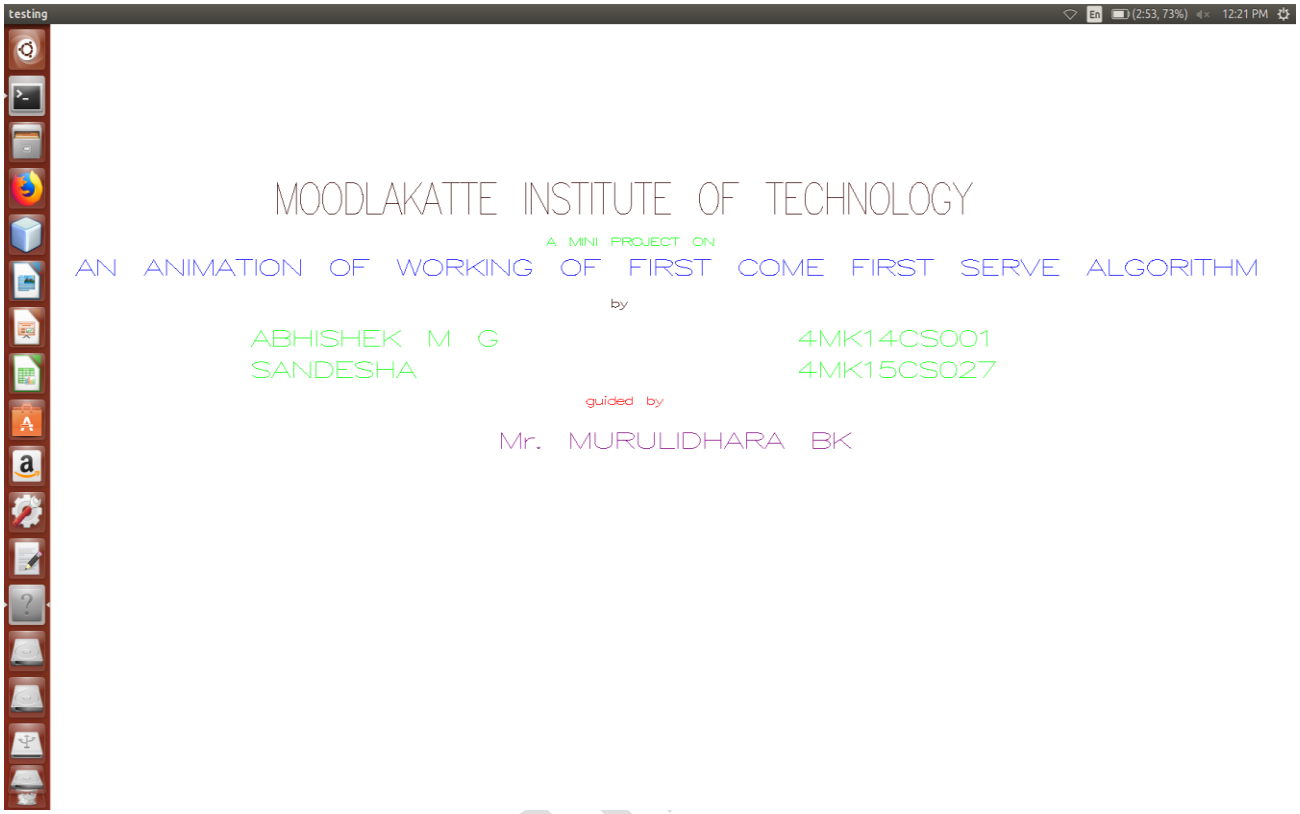


fig 5.1 Introduction about project

Description: starting page with name of project, name of the developer and guide name

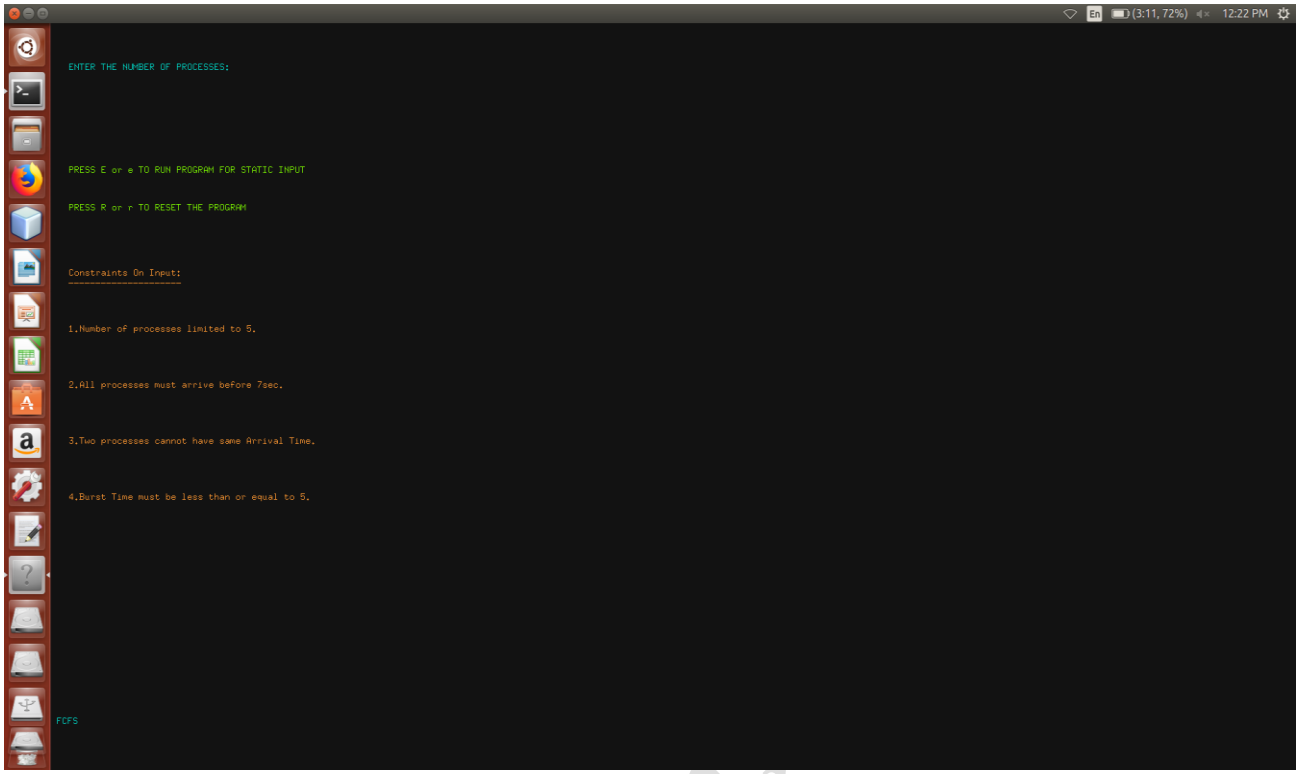


fig 5.2: processes and reset

Description: Enter number of process with reset the program page and to run process

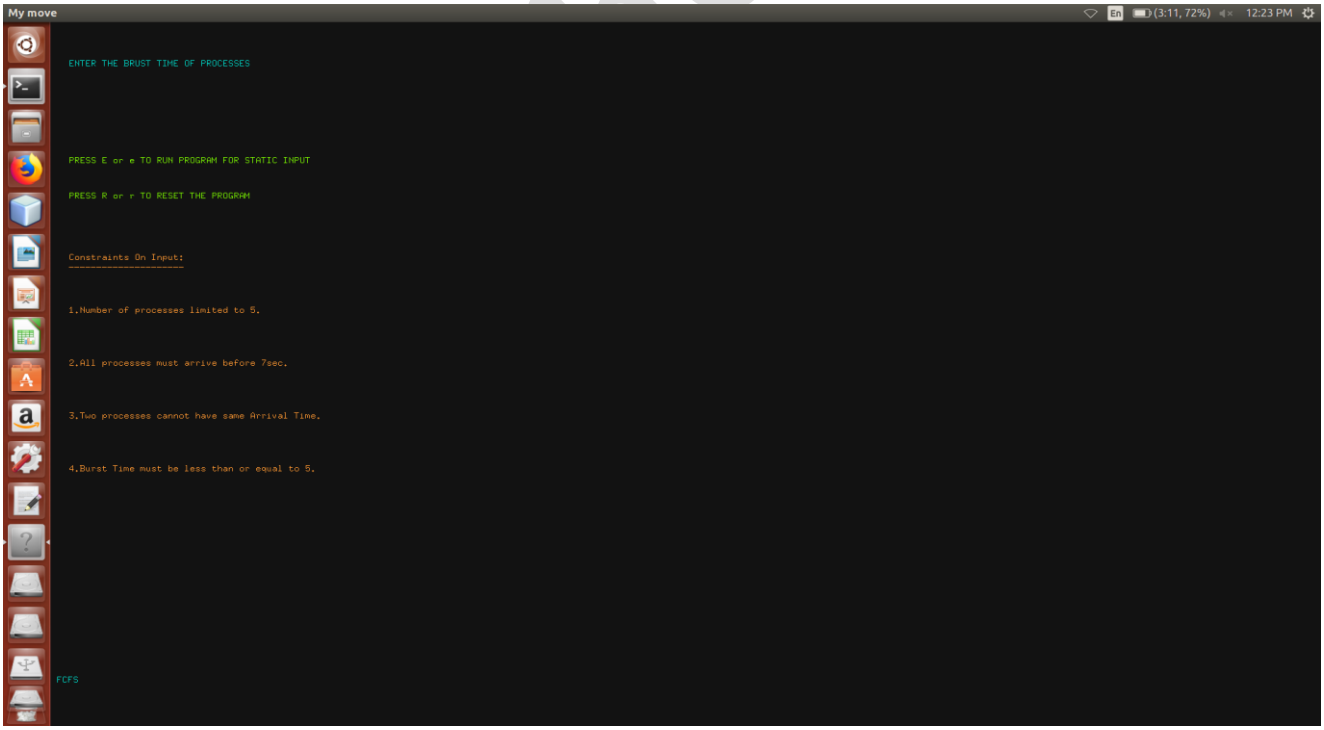
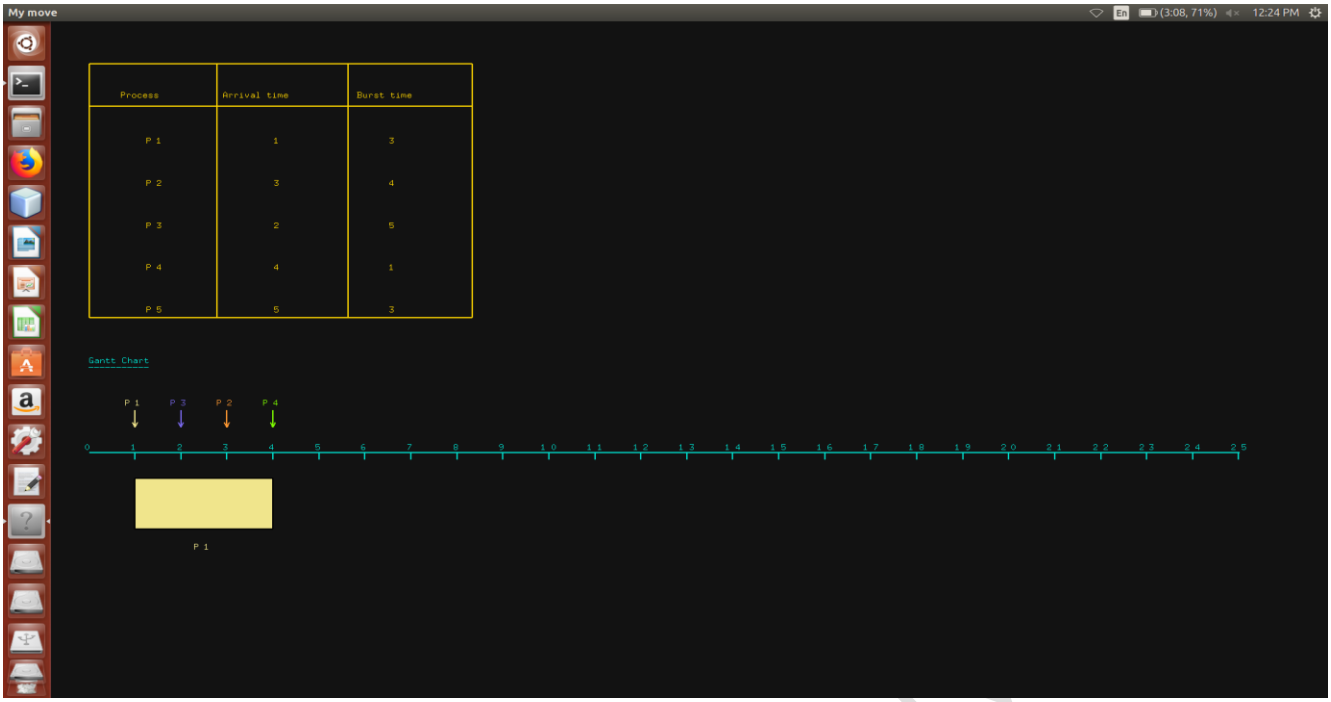


fig 5.3: Calculation process

Description: Entering of process and burst time of process



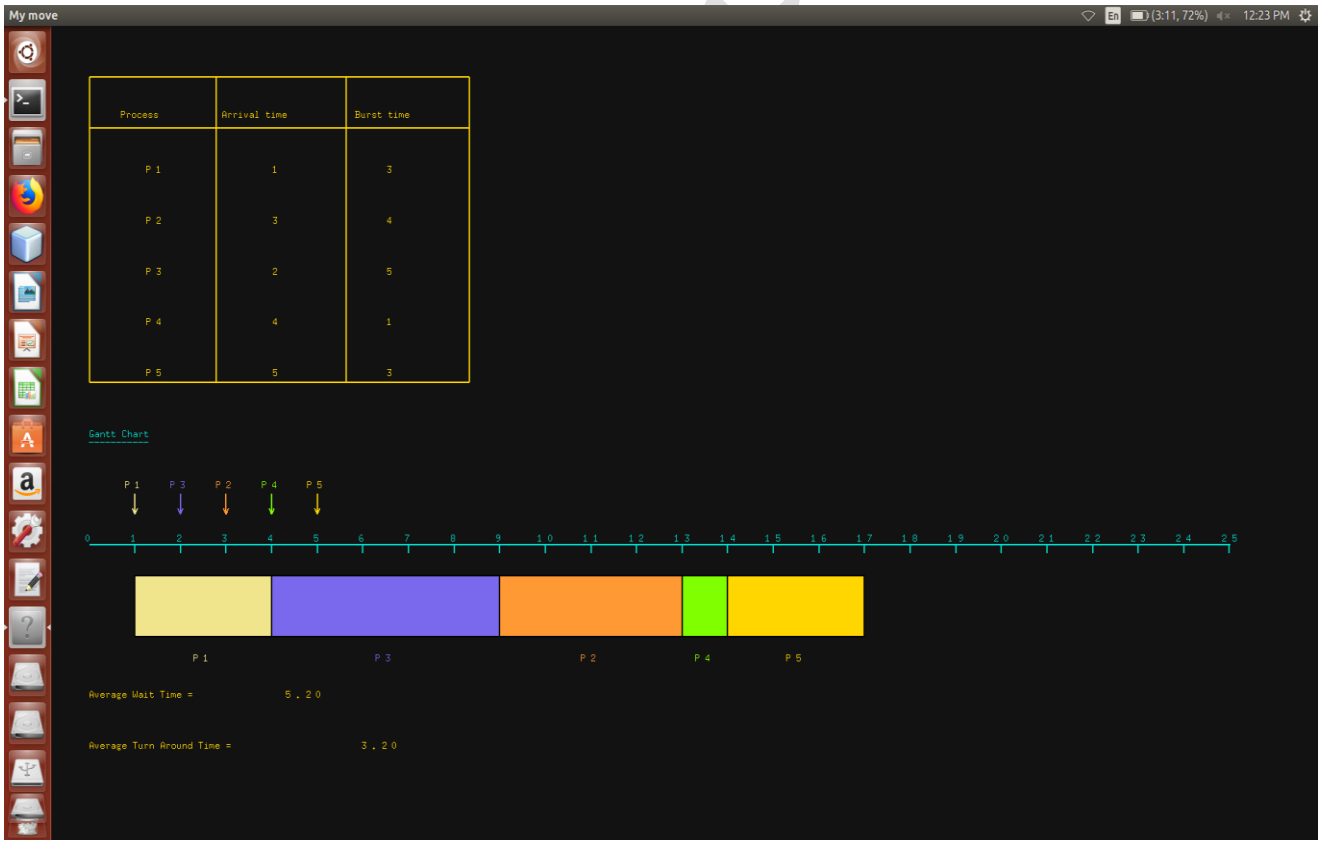


fig 5.4: Output with gantt chart

Description: Final output with table, average wait time, avg turn around time and chart

CHAPTER 6

CONCLUSION

This project will show the processes are dispatched according to their arrival time at the ready queue. When new process enters into the system, its process control block is linked to the end of the ready queue and it is removed from the front of the queue.

After execution of simulation codes of FCFS and SJF come to the conclusion that SJF algorithm is more competitive than FCFS If we talk about performance because It has minimal average waiting that is very important. SJF main flaw is : a very long process to be not executed at the time required, thing that does not happen to FCFS algorithm.

The entire project has been developed and deployed as per the requirements stated by the user, it is found to be bug free as per the testing standards that are implemented and by specification-untraced errors concentrated in the coming versions, which are planned to be developed in near future.

Finally, we like to conclude that we put all our efforts throughout the development of our project and tired to fulfill most of the requirements of the user.

REFERENCES

1. Interactive Computer Graphics A Top-Down Approach with OpenGL - Edward Angel, 5th Edition, Addison-Wesley, 2008

2. The Official Guide to Learning OpenGL, by Jackie Neider, Tom Davis, Mason Woo (THE RED BOOK)

3. OpenGL Super Bible by Richard S. Wright, Jr. and Michael Sweet

4. http://www.cs.rutgers.edu/~decarlo/428/glman.html online man pages.

5. http://www.opengl.org/sdk/docs/man online man pages.