CONTENT

CHAPTER 1

INTRODUCTION

The computer graphics is one of the most effective and commonly used ways to communicate the processed information to the user. It displays the information in the form of graphics object such as picture, charts, graph, and diagram instead of simple text. It concerns the pictorial synthesis of real or imaginary objects from their computer-based models, whereas the related field of image processing treats the converse process.

1.1 INTRODUCION TO COMPUTER GRAPHICS

Interactive computer graphics is the most important means of producing pictures since the invention of photography and television; it has added advantages that, with computer, we can make pictures not only of concrete, "real-world " object but also abstract, synthetic objects, such as mathematical surfaces. Interactive computer graphics thus permits extensive, high-bandwidth user-computer interaction. Today it is used in many different areas of industry, business, government, education, entertainment, and most recently the home.

1.1.1 About Computer Graphics

CG (Computer graphics) started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computer themselves. It includes the creation, storage, and manipulation of models and images of objects. These models include physical, mathematical, engineering, architectural, and even conceptual or abstract structures, natural phenomena, and so on. Computer Graphics today is largely interactive- the user controls the contents, structure, and appearance of objects and their displayed images by using input devices, such as keyboard, mouse or touch sensitive panel on the screen. Bitmap graphics is used for user-computer interaction.

A Bitmap is an ones and zeros representation of points (pixels or, short for 'picture elements') on the screen. Bitmap graphics provide easy-to-use and inexpensive graphics based applications. The concept of 'desktop' is a popular metaphor for organizing screen space. By means of a *window manager*, the user can create, position, and resize rectangular screen areas, called *windows,* that acted as virtual graphics terminals, each running an application. This allowed users to switch among multiple activities just by pointing at the desired window, typically with the mouse. Graphics provides one of the most natural means of communicating with the computer, since our highly developed 2D and 3D pattern –

recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. In many design, implementation, and construction processes, the information pictures can give is virtually indispensable.

1.1.2 What is Computer Graphics Technology?

Computer graphics are graphics created using computers and the representation of image data by a computer specifically with help from specialized graphic hardware and software. The interaction and understanding of computers and interpretation of data has been made easier because of computer graphics.

Computer graphics is widespread today. Many powerful tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: two dimensional (2D), three dimensional (3D), and animated graphics. As technology has improved, 3D computer graphics have become more common, but 2D computer graphics are still widely used. Computer graphics has emerged as a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content.

The Department of Computer Graphics Technology prepares visually oriented students who are interested in creating and managing the production of computer graphics for a wide range of industry. Students work in computer labs developing their graphics skills, techniques, concepts, and management ability through individual and team-based projects. After successful completion of the pre-technical graphics curriculum, students can select to specialize in one of four signature areas in interactive multimedia, technical animation, manufacturing graphics, or construction graphics.

1.1.3 Advantages of computer graphics

Computer graphics is used today in many different areas of industry, business, government, education, entertainment, and, most recently, the home. The list of applications is enormous and is growing rapidly as computers with graphics capabilities become commodity products. Let us look at some of these applications:

- User interface
- Interactive plotting in business, science, and technology
- Office automation and electronic publishing
- Computer-aided drafting and design
- Simulation and animation for scientific visualization and entertainment
- Art and Commerce
- Process control
- Cartography

1.1.4 Applications of Computer Graphics

The development of computer graphics has been driven both by the needsof the user community and by advances in hardware and software. The applications of computer graphics are many and varied. We can however divide them into four major areas.

- Display of information: More than 4000 years ago, the Babylonians developed floor plans of buildings on stones. Today, the same type of information is generated by architects using computers. Over the past 150 years, workers in the field of statistics have explored techniques for generating plots. Now, we have computer plotting packages. Supercomputers now allow researchers in many areas to solve previously intractable problems. Thus, Computer Graphics has innumerable applications.
- Design: Professions such as engineering and architecture are concerned with design. Today, the use of interactive graphical tools in CAD, in VLSI circuits, characters for animation have developed in a great way.
- Simulation and animation: One of the most important uses has been in pilots' training. Graphical flight simulators have proved to increase safety and reduce expenses. Simulators can be used for designing robots, plan it's path, etc. Video games and animated movies can now be made with low expenses.
- User interfaces: Our interaction with computers has become dominated by a visual paradigm. The users' access to internet is through graphical network browsers. Thus Computer Graphics plays a major role in all fields.

1.2 ABOUT 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. Similarly, 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. Some of openGL main features:

- ➢ It provides 3D geometric objects, such as lines, polygons, triangle meshes, spheres, cubes, quadric surfaces, NURBS curves and surfaces.

- It provides 3D modeling transformations, and viewing functions to create views of 3D scenes using the idea of a virtual camera.
- It supports high-quality rendering of scenes, including hidden-surface removal, multiple light sources, material types, transparency, textures, blending, and fog.
- It provides display lists for creating graphics caches and hierarchical models. It also supports the interactive "picking" of objects.
- "OpenGL" is actually a set of three libraries: OpenGL itself, and the supporting libraries GLU and GLUT.

1.2.1 GLUT

Glut provides the facilities for interaction that OpenGL lacks. It provides functions for managing windows on the display screen, and handling input events from the mouse and keyboard. It provides some rudimentary tools for creating Graphical User Interfaces (GUIs). It also includes functions for conveniently drawing 3D objects like the platonic solids, and a teapot. All GLUT function names start with **"glut"**.

1.2.2 OpenGL as a State Machine

OpenGL is a state machine. You put it into various states (or modes) that then remain in effect until you change them. As you've already seen, the current color is a state variable. You can set the current color to white, red, or any other color, and thereafter every object is drawn with that color until you set the current color to something else. The current color is only one of many state variables that OpenGL maintains.

Others control such things as the current viewing and projection transformations; line and polygon stipple patterns, polygon drawing modes, pixel-packing conventions, positions and characteristics of lights, and material properties of the objects being drawn. Many state variables refer tools that are enabled or disabled with the command glEnable() or glDisable().

Most implementations of OpenGL have a similar order of operations, a series of processing stages called OpenGL Rendering Pipeline.

1.2.3 OpenGL Rendering Pipeline

OpenGL can be imagined as a pipeline, with processing stations along the way that takes in the vertices of your model (i.e. points in 3D object space) and transforms them into screen coordinates. The basic pipeline is organized as shown below

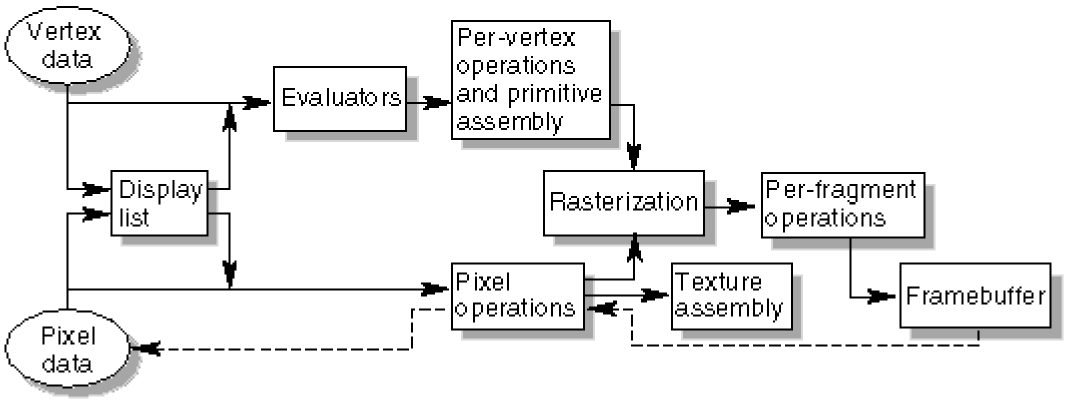


Fig 1.1: Pipeline Architecture of OpenGL

- **Display Lists**

All data, whether it describes geometry or pixels, can be saved in a *display list* 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.(The alternative to retaining data in a display list is processing the data immediately - also known as *immediate mode*.)

- **Evaluators**

All geometric primitives are eventually described by vertices. Parametric curves and surfaces may be initially described by control points and polynomial functions called basis functions. Evaluators provide a method to derive the vertices used to represent the surface from the control points. The method is a polynomial mapping, which can produce surface normal, texture coordinates, colors, and spatial coordinate values from the control points.

- **Per-Vertex Operations**

For vertex data, next is the "per-vertex operations" stage, which converts the vertices into primitives. Some vertex data (for example, spatial coordinates) are transformed by 4 x 4 floating-point matrices. Spatial coordinates are projected from a position in the 3D world to a position on your screen.

If advanced features are enabled, this stage is even busier. If texturing is used, texture coordinates may be generated and transformed here. If lighting is enabled, the lighting calculations are performed using the transformed vertex, surface normal, light source position, material properties, and other lighting information to produce a color value.

- **Primitive Assembly**

Clipping, a major part of primitive assembly, is the elimination of portions of geometry which fall outside a half-space, defined by a plane. Point clipping simply passes or rejects vertices; line or polygon clipping can add additional vertices depending upon how the line or polygon is clipped. In some cases, this is followed by perspective division, which makes distant geometric objects appear smaller than closer objects. Then viewport and depth (z coordinate) operations are applied.

If culling is enabled and the primitive is a polygon, it then may be rejected by a culling test. Depending upon the polygon mode, a polygon may be drawn as points or lines. The results of this stage are complete geometric primitives, which are the transformed and clipped vertices with related color, depth, and sometimes texture-coordinate values and guidelines for the rasterization step.

- **Pixel Operations**

While geometric data takes one path through the OpenGL rendering pipeline, pixel data takes a different route. 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.

There are special pixel copy operations to copy data in the framebuffer to other parts of the framebuffer or to the texture memory. A single pass is made through the pixel transfer operations before the data is written to the texture memory or back to the framebuffer.

If 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.

- **Rasterization**

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

- **Ant-aliasing**

Rendering resolution-independent entities (such as 3D models) for viewing on a raster (pixel-based) device such as a LCD display or television inevitably causes aliasing artifacts mostly along geometric edges and the boundaries of texture details; these artifacts are informally called "jaggies". A pre-anti-aliased bitmap texture being displayed on a screen (or screen location) at a resolution different than the resolution of the texture itself (such as a textured model in the distance from the virtual camera) will exhibit aliasing artifacts, while any procedurally defined texture will always show aliasing artifacts as they are resolution-independent; techniques such as mipmapping and texture filtering help to solve texture-related aliasing problems.

- **Fragment Operations**

It is the last process to convert fragments to pixels onto frame buffer. A texture element is generated from texture memory and it is applied to the each fragment. Then fog calculations are applied. After that, there are several fragment tests follow in order; Scissor Test ⇒ Alpha Test ⇒ Stencil Test ⇒ Depth Test.

Then fog calculations may be applied, followed by the scissor test, the alpha test, the stencil test, and the depth-buffer test (the depth buffer is for hidden-surface removal). Failing an enabled test may end the continued processing of a fragment's square. Then, blending, dithering, logical operation, and masking by a bitmask may be performed.

Finally, the thoroughly processed fragment is drawn into the appropriate buffer, where it has finally advanced to be a pixel and achieved its final resting place.

CHAPTER 2

INTRODUCTION TO PROJECT

2.1 Aim of the Project

The main aim is to understand the key ideas and Implementation of Computer Graphics using OpenGL for representing “HUMAN ANATOMY”. It is necessary to display the object in the user’s point of view.

- The aim is to draw attention of users toward computer graphics.
- The aim is to create dominant project which is simple in use.

2.1.1 Purpose

- It can be used to show how efficient this particular software is.
- To display the motion view of the objective.
- To learn the concepts of OpenGL.

2.2 The benefits of the Project

- **Simplicity:** The project is easy and simple to use.
- **Usability:** It is easy to use and implement.
- **Flexibility:** It is very flexible since it is easy to add new features to it.

2.3 Constraints on the project

- Project executes only if the system is installed with ‘glut’ package.
- Output will differ if there is fault in graphic system.
- May produce flickering image if colored buffer is not cleared properly.
- We need to initialize glut library in order to interact with window system.

2.4 Applications

- The project has the strength to create images in 2D/3D format through specific software Applications for teaching in Science Subjects.
- Display an animation or series of animations.
- Play a video sequence.
- Use vector graphics to draw various designs.
- Gaming and Cartoon Applications.

2.5 About the project

This project aims to demonstrate human anatomy such as digestive system, respiratory system and excretory system using openGL. Digestive system shows the movement of food in the body. Respiratory system shows the effect of inhalation and exhalation on lungs (i.e. breathing). Excretory system shows the main organs involved in excretion.

These different anatomies are put in a menu and one can select the menu option for look into a particular anatomy.

There are 4 menu options. For selecting Digestive System, Respiratory System, Excretory System and descriptions options. The description takes keys from keyboard to show brief description on these anatomies.

CHAPTER 3

REQUIREMENTS ANALYSIS

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

3.1 Hardware Requirements

- **Processor:** Intel Pentium 3 onwards compatible hardware.
- **RAM:** 256 Mb RAM.
- **Monitor:** EGVGA Compatible.
- **Motherboard:** 845c Intel Motherboard.
- **Keyboard:** Standard 101 key keyboard.
- **Backup Media:** Floppy/pen drive/Hard disk.
- **Hard disk:** 40 GB.
- **I/O Device:** Standard input and output devices.

3.2 Software Requirements

- **Operating System:** LINUX/ Windows 7 and above
- **Language Tool:** C with glut libraries.
- **Compiler:** GCC compiler or Code: Blocks or Visual Studio 2015
- **Libraries:** Supporting glut.h, stdio.h, stdlib.h, math.h,unistd.h etc & 16bit color resolution.

CHAPTER 4

SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could see it as the application of systems theory to product development. There is some overlap with the disciplines of analysis, system and systems engineering.

The system design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. It relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified/authenticated, how it is processed, and how it is displayed as output.

Systems design can generally be broken down into three sub-tasks:

- User Interface Design
- Data Design
- Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

This project aims to demonstrate the user interface by using keyboard and mouse I/O devices. When a predefined key is pressed, the corresponding action takes place. In an OpenGL project we would like to create a scene through which we are demonstration of human Anatomy.

4.1.1 User Input

User Interaction Keys:

Key d or D: To select Brief description on Digestive System.
Key r or R: To select Brief description on Respiratory System.

Key e or E: To select Brief description on Excretory.

Q or ESC key: To exit.

Mouse Button Interaction:

Right: The menu for selecting the Digestive System, Respiratory System, Excretory System and descriptions options.

Left: Selecting the various options present in the menu.

4.2 Block Diagram

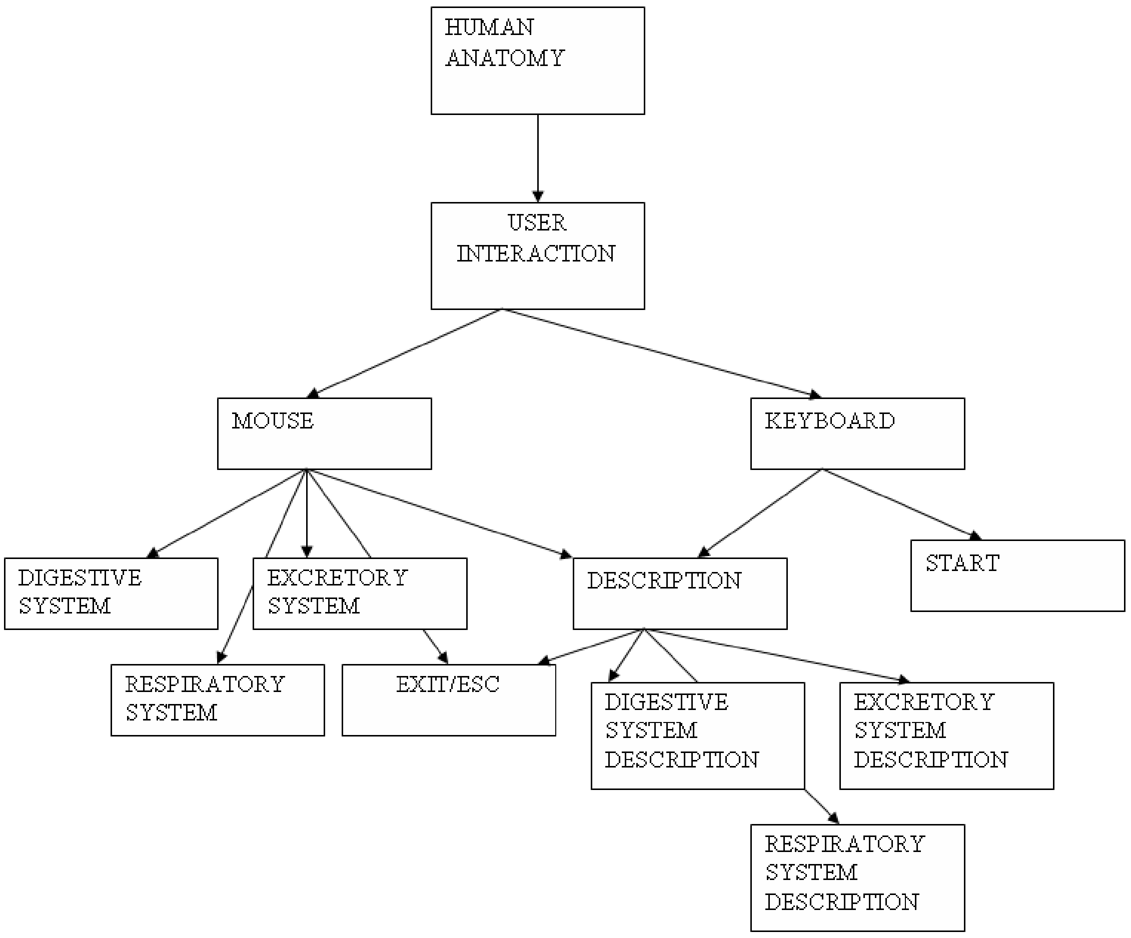


Figure 3.1 Design of the execution of project

The design of the system shows that when the program is made to run it first enters the output window. User interacts with the system using mouse

CHAPTER 5

IMPLEMENTATION

5.1 Algorithm

Step 1: Initialize the window.

Step 2: Set the viewport.

Step 3: Display the front screen. Type enter key.

Step 4: Click right button of mouse to display the menu.

Step 5: if digestive system is chosen, digestion of food is displayed along with labeling.

Step 6: if respiratory system is chosen, inhale/exhale displayed along with labeling.

Step 7: if excretory system is chosen, the same displayed along with labeling.

Step 8: if description is chosen instruction window is displayed for necessary key events.

Step 9: if Quit option is chosen, the display window closes.

5.2 Functions

Headers Defined

The in-built are defined in the OpenGL library. Some of the headers that are used as follows:

- #include<stdio.h> : to take input from standard input and write to standard output.
- #include<stdlib.h> : to include standard library functions.
- #include<GL/glut.h> : to include glut library files.
- #include<string.h> : to include string library files

5.2.1 Built-in Functions

- **glutInit ():-** It is used to initialize the GLUT library.
- **glutInitDisplayMode() :-** It sets the initial display mode.
- **glutInitWindowPosition():-** These functions are used to set the initial window position and sizerespectively.
- **glutKeyboardFunc() :-** It sets the keyboard callback for the current window.
- **glutIdleFunc**() :- It is called when no event is being processed.
- **glClearColor() :-** This function specifies clear values for the color buffers.
- **glMainLoop() :-** This function carries on the loop.
- **glMatrixMode() :-**It specifies which matrix is the current matrix.
- **glOrtho() :-** This function multiplies the current matrix by an orthographic matrix.
- **glPointSize() :-** This function specifies the diameter of rasterized points.
- **glLineWidth():-** This functions line width
- **glTranslatef() :-** This function multiplies the current matrix by a translation matrix.

- **glViewport() :-** It sets the viewport.
- **glutCreateMenu() :-** This function creates a new pop up menu.
- **glutPostRedisplay() :-** It requests the display callback be executed after the callback returns..
- **glutSwapBuffers() :-** It swaps the buffers of the current window if double buffered.
- **glFlush() :-** This function force any buffer openGL commands to execute.
- **glutBitMapCharacter():-**This function renders bit map character.
- **glClear():-**Clears buffer to preset values. Specifies BITWISE OR of masks that indicate the buffers to be cleared.
- **glBegin():-**Specifies the primitive or primitives that will be created from vertices presented between glBegin() and glEnd().
- **glVertex():-**Function commands are used within the glBegin/glend to specify point line and polygon vertices.
- **glColor3f():-**Set the color by using three values.

5.2.2 User Defined Function

frontscreen(): used to display the strings of the project title and our names in the flickers screen as well as the other strings.

Init(): used for shading purposes.

drawstring():used to print text strings.

drawstring1():used to print text strings.

body():used to draw human body.

digesdraw():used to draw various organs in digestive system with labeling.

liver():used to draw liver in digestive system.

largeintes():used to draw large intestine of digestive system.

food():used to draw food particle for digestive system.

foodp():used to draw small food particle for digestive system.

digest():used to draw digestive system.

animate():used to movement to food particle.

descDigest():used to display description about digestive system.

resdraw():used to draw various organs in respiratory system with labeling.

leftlung():used to draw left lung of respiratory system.

rightlung():used to draw right lung of respiratory system

respiration():used to draw respiratory system.

animate1():used for expansion and contraction of lungs.

descResp():used to display description about respiratory system.

excredraw():used to draw various organs in excretory system with labeling.

excretion():used to draw excretory system.

mydisplay(): used for displaying the entire scene.

myKeyboardFunc(): used for giving the keyboard inputs.

desc():instruction window is displayed for necessary key events.

mymenu(): used for setting up the menu to be displayed.

Reshape(): used for setting up the viewport.

CHAPTER 6

TESTING & RESULTS

6.1 INTRODUCTION TO TESTING

Testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results.Although crucial to software quality and widely deployed by programmers and testers, testing still remains an art, due to limited understanding of the principles of software. The difficulty in testing stems from the complexity of software, we cannot completely test a program with moderate complexity. Testing is more than just debugging. The purpose of testing can be quality assurance, verification and validation, or reliability estimation. Testing can be used as a generic metric as well. Correctness testing and reliability testing are two major areas of testing. Software testing is a trade-off between budget, time and quality.

6.1.1 Introduction to testing

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

Validation: - Are we building the right product? Validation involves checking that the program has implanted meets the requirement of the users.

Verification: - Are we building the product right?Verification involves checking that the program confirms to its specification.

6.1.2 The following states have been implemented in the testing

- **Unit Testing: -** Each individual unit is tested for correctness. These individual components are tested to ensure that they operate correctly.
- **Module Testing: -** A module is a collection of dependent components such as a function. A module encapsulates related components so can test without other system modules.
- **Sub-system Testing: -** This phase involves testing collection of modules, which have been integrated into sub-systems. Sub-systems may be independently designed and implemented.
- **System Testing: -** The Sub-systems are integrated to make up the entire system. The errors that result from unanticipated interaction between sub-systems and system components are removed.
- **Acceptance Testing: -**This is the final stage in the testing process before the system is tested for operational use. Any requirement problem o requirement definition problem revealed from acceptance testing are considered and made error free.
- **Test Plan: -**System testing is very expensive for some large systems with complex non-functional requirements; half of the system development budget may be spent on testing. Careful planning is needed to the most of testing and controlled testing cost.

Test case	Test-Case	Input	Actual	Expected	Remark

ID	description		output	output	
1	Display of digestive system with labeling	Right mouse button	digestive system displayed	digestive system should be displayed with labeling	Pass
2	Food particle movement in digestive system	Right mouse button	Food particle moves	Food particle movement	Pass
3	Display of respiratory system with labeling	Right mouse button	respiratory system displayed	respiratory system should be displayed with labeling	Pass
4	expansion and contraction of lungs	Right mouse button	Lungs size expands and then contracts accordingly "inhale" and "exhale" texts are displayed	expansion and contraction of lungs	Pass
5	Display of excretory system with labelling	Right mouse button	Excretory system displayed	excretory system should be displayed with labelling	Pass

Table 6.1. Table of contents

6.2 RESULTS

Reporting test execution results is very important part of testing, whenever test execution cycle is complete, tester should make a complete test results report which includes the Test Pass/Fail status of the test cycle. If manual testing is done then the test pass/fail result should be captured in an excel sheet and if automation testing is done using automation tool then the HTML or XML reports should be provided to stakeholders as test deliverable.

Several errors were detected and rectified and the whole project is working as it should have to work with proper output and high efficiency.

SNAPSHOTS

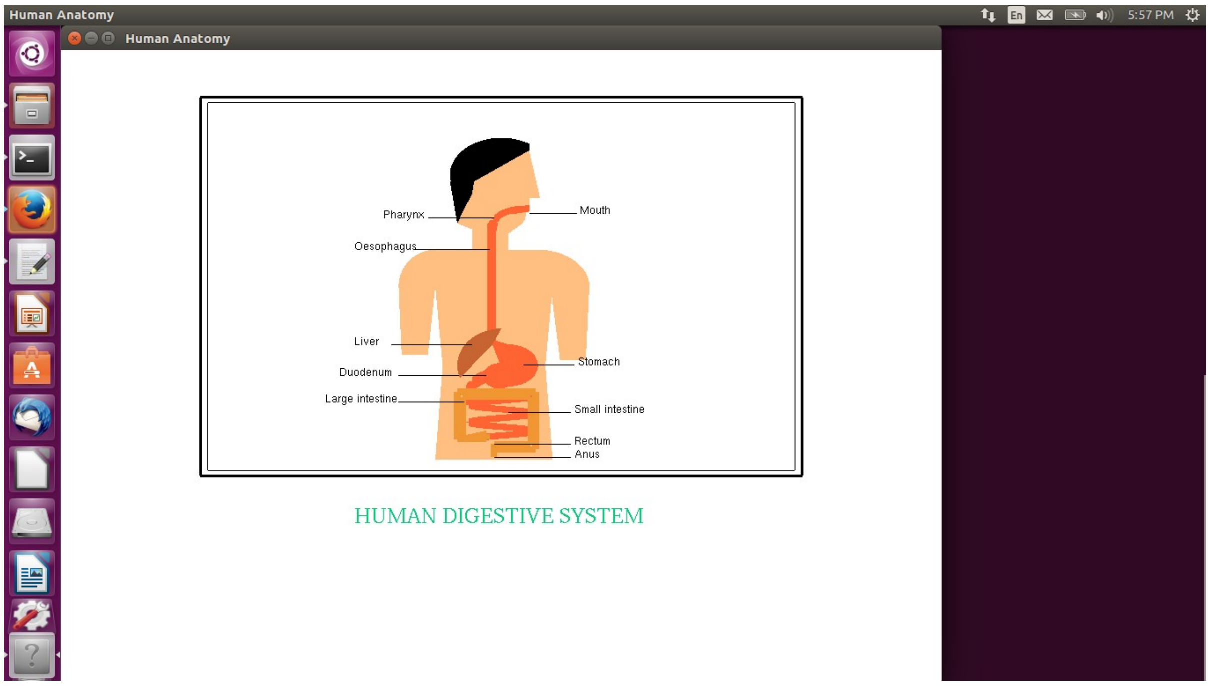


Fig 7.1: Display on selecting the option “Digestive system”.

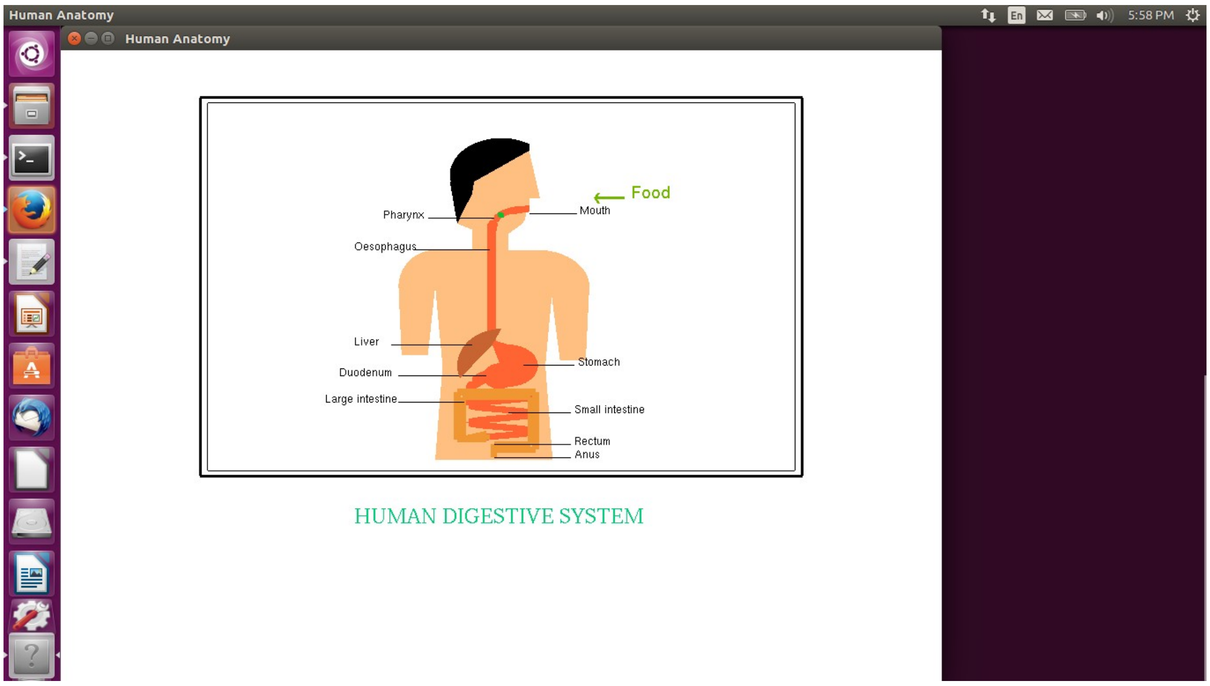


Fig 7.2(a): Display of movement of food in digestive system

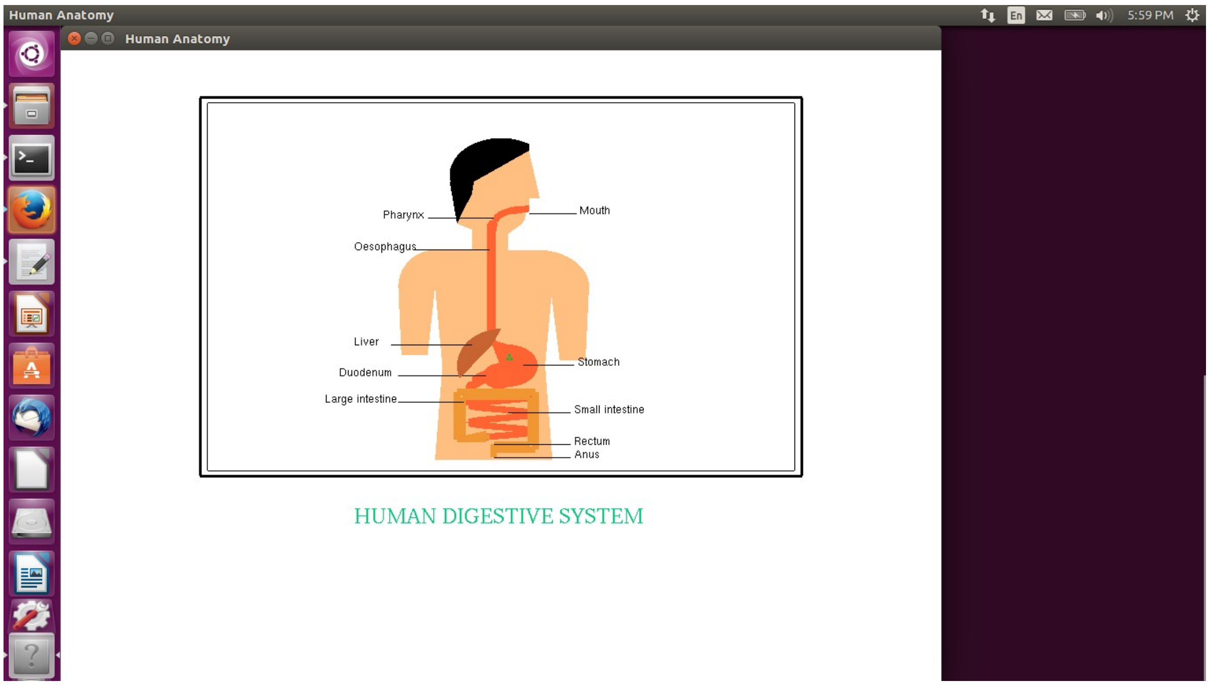


Fig 7.2(b): Display of movement of food in digestive system

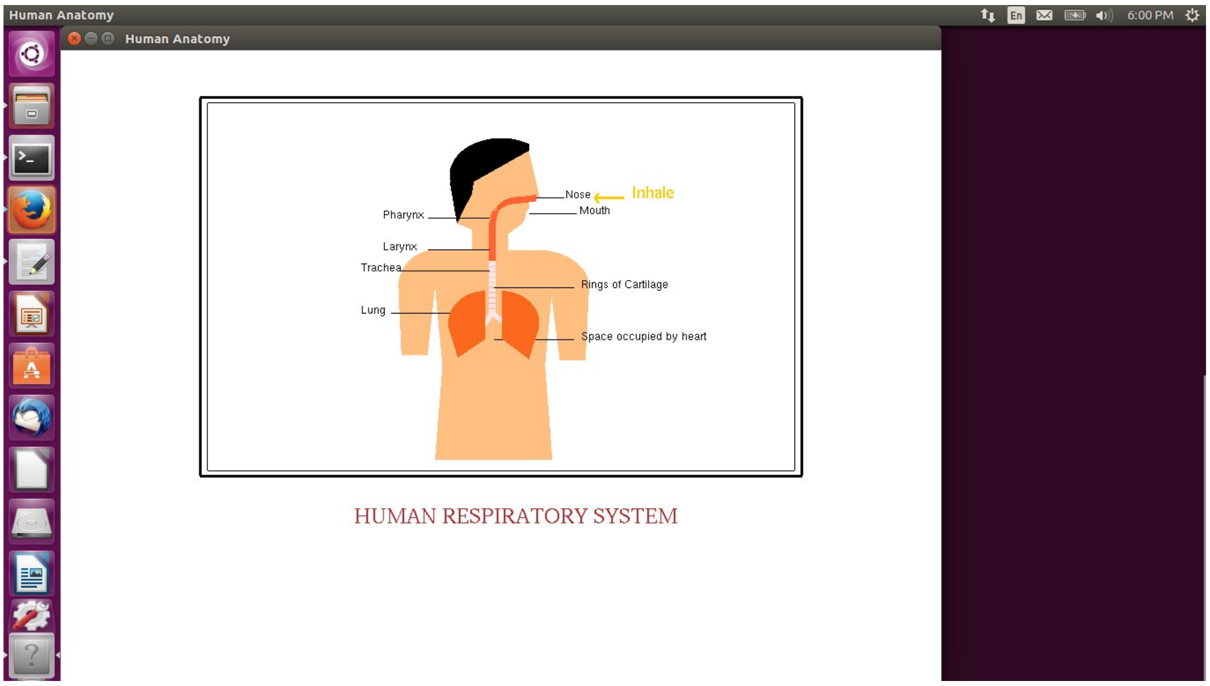


Fig 7.3: Display on selecting the option “Respiratory system” and display of inhalation

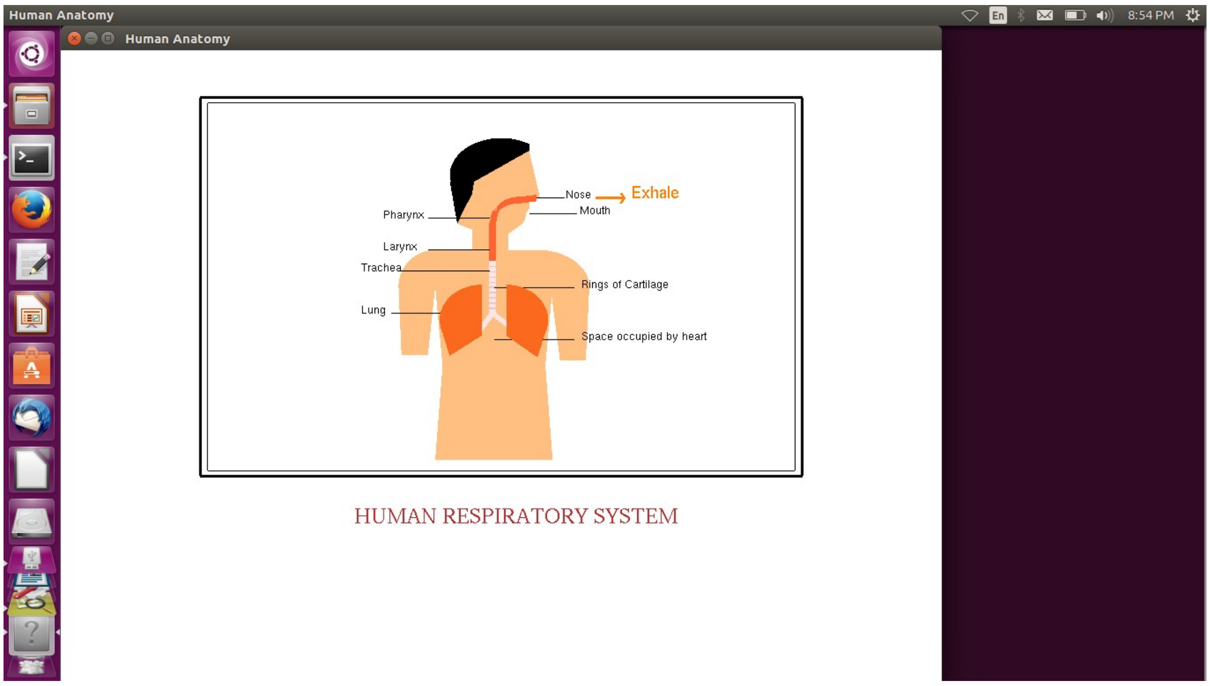
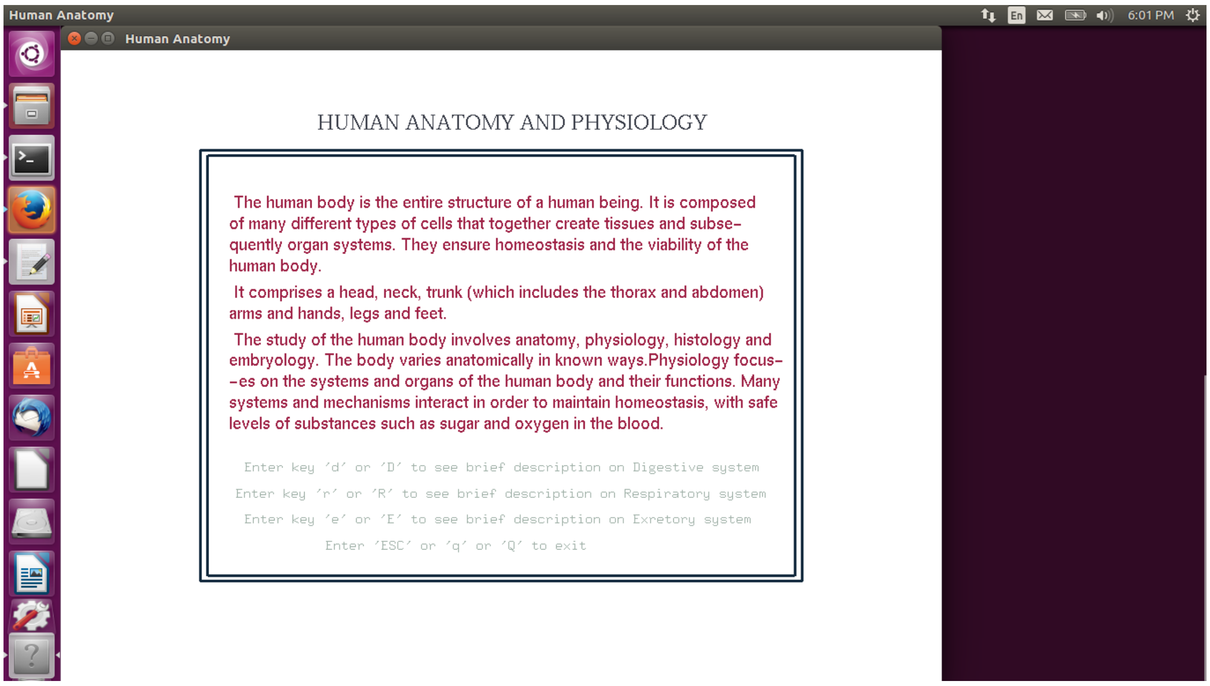


Fig 7.3: Display on selecting the option "Respiratory system" and display of exhalation



HUMAN ANATOMY AND PHYSIOLOGY

The human body is the entire structure of a human being. It is composed of many different types of cells that together create tissues and subse–quently organ systems. They ensure homeostasis and the viability of the human body.

It comprises a head, neck, trunk (which includes the thorax and abdomen) arms and hands, legs and feet.

The study of the human body involves anatomy, physiology, histology and embryology. The body varies anatomically in known ways.Physiology focus––es on the systems and organs of the human body and their functions. Many systems and mechanisms interact in order to maintain homeostasis, with safe levels of substances such as sugar and oxygen in the blood.

```
Enter key 'd' or 'D' to see brief description on Digestive system
Enter key 'r' or 'R' to see brief description on Respiratory system
Enter key 'e' or 'E' to see brief description on Exretory system
Enter 'ESC' or 'q' or 'Q' to exit
```

Fig 7.4: Display on selecting the option "Description".

CONCLUSION

This project is an effort in the development of a Graphical Software package which is the building block of graphical application. During the development of this package effort lead to understanding the display of geometric primitives like rectangles, lines, line loops, polygon, modes of display, features like translation were also designed. Various functions and operations of the graphical library provide the learning platform to get the maximum performance of the OpenGL functions.

The project "HUMAN ANATOMY" has been successfully implemented using OpenGL. The illustration of graphical principles and OpenGL features are included and application program is efficiently developed.

This project enlightens the basic idea of scaling, rotation and translation of the objects. Since it uses interaction with both keyboard and mouse it is sufficiently easy for any kind of end user to run it.

However user requirement changes and provision for changes in design has been allowed. If no major requirements are demanded from the user the current design holds. Thus this project meets the basic requirements successfully and is flexible in all respects to one and all. On conclusion this mini project is implemented with the standard OpenGL functions.

FUTURE ENHANCEMENT

- ✓ An attempt has been made to develop a Graphical Software package, which meets the necessary requirement of the user successfully. The objects in the project can move with respect to x, y and z axis, it can be enhanced to 3D-movement more precisely. We can also change the color and provide more effects.
- ✓ Implementation of the other graphical applications will defiantly enhance the representation and view of the project. The mini project can further be enhanced to demonstrate the use of other OpenGL functions of higher degree.
- ✓ Since it is user friendly it enables the user to interact efficiently and easily. Thus the package designed demonstrates the Basic Structures Orientation of some objects to the user.
- ✓ Support for advanced 3D representation of the entire scenario.
  - ✓ Support for transparency of layers and originality.

BIBLIOGRAPHY

Books

- Edward Angel: Interactive Computer Science A Top-Down Approachwith OpenGL, 5th Edition, Pearson Education 2008.
- James D Foley, Andries Van Dam, Steven K Feiner, John F Hughes, Computer Graphics, Pearson Education 1997.
- Donald Hearn, Pauline Baker: Computer Graphics - OpenGL Version, 3rd Edition, Pearson Education 2004.
- Programming Guide, 5thEdition.

Websites & Links

⇒ http://www.wikipedia.org/
⇒ http://www.opengl.org/
⇒ http://www.khronos.org/
⇒ http://www.opengl-tutorial.org/