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 Computer Graphics

Computer graphics are visual representations of data displayed on a monitor made on a computer. Computer graphics can be a series of images or single image. Computer graphics is a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Computer graphics is made up of number of pixels. Pixel is the smallest graphical unit or picture represented on the computer screen. Basically there are two types' namely interactive computer graphics and non-interactive computer graphics. Some topics in computer graphics include user interface design, sprite graphics, vector graphics, 3D modelling, shades, GPU design implicit surface visualization with ray tracing, and computer vision, among others. [1]

Computer generated imagery is used for movie making, video game and computer program development. Computer graphics are pictures and films created using computers. Usually, the term refers to computer-generated image data created with help from specialized graphical hardware and software. It is a vast and recent area in computer science. Computer graphics is responsible for displaying art and image data effectively and meaningfully to the user. It is also used for processing image data received from the physical world.[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.[1]

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.[1]

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.[2]

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.[2]

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

1.4 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.[2]

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 its 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.

1.5 About OpenGL

Open Graphics Library (OpenGL) is a cross-language, cross –platform application programming interface (API) for rendering 2D and 3D vector graphics. OpenGL provides a set of commands to render a three dimensional scene. That means you provide the data in an OpenGL useable form and OpenGL will show this data on screen. An OpenGL will work on every platform, as long as there is an installed implementation. Because it is system independent, there are no functions to create windows etc. The API is typically used to interact with a graphics processing unit(GPU), to achieve hardware-accelerated rendering. 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. [2]

Some main features are:

- 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.

1.6 OpenGL Architecture

The architecture of OpenGL is based on a client-server model. An application program written to use the OpenGL API is the "client" and runs on the CPU. The implementation of the OpenGL graphics engine (including the GLSL shader programs you will write) is the "server" and runs on the GPU. Geometry and many other types of attributes are stored in buffers called Vertx Buffer Objects (or VBOs). These buffers are allocated on the GPU and filled by your CPU program.

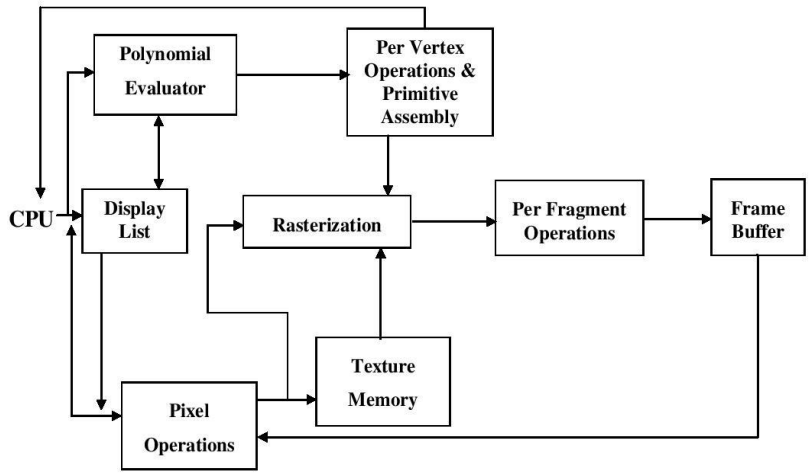

Fig. 1.6 OpenGL Architecture

Fig. 1.6 is representing the flow of graphical information, as it is processed from CPU to the frame buffer.

There are two pipelines of data flow. The upper pipeline is for geometric, vertex-based primitives. The lower pipeline is for pixel-based, image primitives. Texturing combines the two types of primitives together. [2]

1.7 OpenGL as a Renderer

OpenGL Pipeline has a serie sof processing stages in order.Those stages are –vertex operation, primitive assembly, pixel transfer, rasterization and fragment operation.

1.7.1 Geometric primitives

- Points, lines and polygons.

1.7.2 Image primitives

- Images and bitmaps.
- Separate pipeline for images and geometry.
- Linked through texture mapping.

1.7.3 Rendering depends on state

- Colors, materials, light sources, etc.

As mentioned, OpenGL is a library for rendering computer graphics. Generally, there are two operations that we do with OpenGL:

- Draw objects.
- Change the state of how OpenGL plots.

OpenGL has two types of things that it can render: geometric primitives and image primitives. *Geometric primitives* are points, lines and polygons. *Image primitives* are bitmaps and graphics images (i.e. the pixels that you might extract from a JPEG image after you've read into your program.)

The other common operation that we do with OpenGL is *setting state.* " Setting state" is the process of initializing the internal data that OpenGL uses to render our primitives. It can be as simple as setting up the size of points and color that we want a vertex to be, to initializing multiple bitmap levels for texture mapping [3].

1.8 OpenGL Contributions

It is very popular in the video games development industry where it competes with Direct3D (on Microsoft Windows). OpenGL is also used in CAD, virtual reality, and scientific visualization programs. OpenGL is very portable. It will run for nearly every platform in existence, and it will run well. It even runs on Windows NT 4.0 etc. The reason OpenGL runs for so many platforms is because of its Open Standard.

OpenGL has a wide range of features, both in its core and through extensions. Its extension feature allows it to stay immediately current with new hardware features, despite the mess it can cause [3].

1.9 Limitations of OpenGL

- OpenGL is case sensitive.
- Line Color, Filled Faces and Fill Color not supported.
- Bump mapping is not supported.
- Shadow plane is not supported.

Summary

OpenGL is the industry is most widely used environment for the development of portable, interactive 2D and 3D graphics applications. The OpenGL API is open source and is platform independent. It is the specification of an application programming interface (API) for computer graphics programming. The interface consists of different function calls, which may be used to draw complex 3D scenes. It is widely used in CAD, virtual reality, scientific and informational visualization, flight simulation and video games.

CHAPTER 2

LITERATURE SURVEY

The objects are defined in their local spaces (model spaces). We need to transform them to the common world space, known as model transform. To perform model transform, we need to operate on the so-called model-view matrix (OpenGL has a few transformation matrices), by setting the current matrix mode to model-view matrix.[3]

OpenGL's object is made up of primitives (such as triangle, quad, polygon, point and line). A primitive is defined via one or more vertices. The colour-cube is made up of 6 quads. Each quad is made up of 4 vertices, defined in counter-clockwise (CCW) order, such as the normal vector is pointing out, indicating the front face. All the 4 vertices have the same colour. The colour-cube is defined in its local space (called model space) with origin at the centre of the cube with sides of 2 units. Similarly, the pyramid is made up of 4 triangles (without the base). Each triangle is made up of 3 vertices, defined in CCW order.[3]

The 5 vertices of the pyramid are assigned different colours. The colour of the triangles are interpolated (and blend smoothly) from its 3 vertices. The basic functions like glColor3f (), glTotalef, glTranslate etc. that are most commonly used in the code are taken from the prescribed VTU text book "Donald Hearn & Pauline Baker: Computer Graphics-OpenGL version, 3rd Edition, Pearson Education,2011. The lab programs in the syllabus also serve as a basic template for creating a project. The usage of colours and specifications are taken from the various programs that were taught in the lab. OpenGL graphics rendering pipeline performs so-called view transform to bring the world space to camera's view space. In the case of the default camera position, no transform is needed.[3]

Lasers are available in a variety of form including solid, liquid and gas. Each basic type of laser is usually identified by the material within it, which provides the active light amplifying medium. In case of argon lasers, the amplifying material is argon.[4] The performance characteristic of a laser includes output, wavelength, beam diameter, beam divergence, mode shape, optical noise, oscillation, bandwidth, input power, ruggedness, reliability and lifetime of some performance specification. Important financial characteristics include acquisition and life cycle cost. All of these characteristics are interrelated in a complex manner with the design and operating variable of a laser. [5]

Summary

This chapter will brief about the literature survey carried out about the related graphics object that we are going to demonstrate in OpenGL using linux GCC compiler to simulate “Laser simulation”.

CHAPTER 3

PROJECT DESCRIPTION

A laser is a device that emits light through a process of optical amplification based on the stimulated emission of electromagnetic radiation. The term "laser" originated as an acronym for "light amplification by stimulated emission of radiation".

This project describes mainly about the Simulation of the Laser.It involves the Principle and Working of the Laser in order to produce the Laser Beam which is coherent.The light from a laser is said to be coherent, which means the wavelengths of the laser light are in phase in space and time.

To explain the process of light amplification in a laser requires an understanding of the energy transition phenomena in the atoms of its active medium. They include-Absorption,spontaneous emission,pumping and population inversion,& induced and stimulated emission.The theory of quantum mechanics states that the electrons of atoms can take different energy states, E1, E2, E3, for example, with E1<E2<E3.

If the atom is initially in the ground level E1, the atom will remain in this level until it gets excited. When an EM wave of frequency n0 is incident on the material, there is a finite probability that the atom will absorb the incident energy and jump to energy level E2. This process is called Absorption.

By quantum mechanics the lower energy level is more stable than higher energy levels, so electrons tend to occupy the lower level. Those electrons in higher energy levels decay into lower levels, with the emission of EM radiation. This process is called spontaneous emission. Pumping is the act of energy transfer from external source to the gain medium of a laser.

The energy is absorbed in the medium, producing the excited states in its atoms when the no. of particles in one excited state exceeds the no. of particles in the ground state or less excited state, population inversion is achieved.Amplification of incident wave is only possible when the population of the upper level is greater than that of the lower level. This case is called Population Inversion.Suppose the atoms of the active medium are initially in E2.

If external EM waves with frequency n0 that is near the transition frequency between E2 and E1 is incident on the medium, then there is a finite probability that the incident waves will force the atoms to undergo a transition E2 to E1. Every E2-E1 transition gives out an EM wave in the form of a photon. We call this stimulated emission .These Principles makes the laser work by producing high intensity,unidirectional,monochromatic and coherent beam of light.

Summary

The proposed project uses OpenGL to simulate the “Laser Simulation” and to analyze its principles and working using computer graphics.

CHAPTER 4

SOFTWARE REQUIREMENT SPECIFICATION

A requirements specification is a description of a software system to be developed. The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide. Use cases are also known as functional requirements. In addition to use cases the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation. For the hardware requirements the SRS specifies the logical characteristics of each interface between the software product and the hardware components. It specifies the hardware requirements like memory restrictions, cache size, the processor, RAM size etc. those are required for the software to run. Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign.

4.1 Hardware Requirements

The following are the minimum hardware requirements of the project -

- **Processor:** Intel Pentium 3 onwards compatible hardware.
- **RAM:** 512 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.

4.2 Software Requirements

The following are the minimum software requirements of the project -

- **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 & 16-bit color resolution.

4.3 Functional Requirements

The following are the minimum functional requirements of the project -

- The system must be able to interact with the user input like mouse, keyboard and menu.
- The system must allow the user to traverse from one page to another using menu.
- The electrons of atoms can take different energy states, E1, E2, E3.
- The atom is initially in the ground level E1, the atom will remain in this level until it gets excited.
- When an EM wave of frequency n0 is incident on the material, there is a finite probability that the atom will absorb the incident energy and jump to energy level E2. This process is called Absorption.
- Electrons tend to occupy the lower level. Those electrons in higher energy levels decay into lower levels, with the emission of EM radiation. This process is called spontaneous emission.
- Pumping is the act of energy transfer from external source to the gain medium of a laser.
- The energy is absorbed in the medium, producing the excited states in its atoms when the no. of particles in one excited state exceeds the no. of particles in the ground state or less excited state, population inversion is achieved.Amplification of incident wave is only possible when the population of the upper level is greater than that of the lower level. This case is called Population Inversion.

4.4 Non-functional Requirements

The following are the minimum Non-functional requirements of the project -

- Reliability:- It must be make sure that the system is reliable in its operation and for securing the sensitive details.
- Safety requirement:- Software shall not cause any harm to the human user.
- Security requirements:- System shall not behave abruptly or corrupt the project file on receiving wrong input from the user.
- Performance requirement:- The program reacts to the input quickly as specified.

Summary

This chapter includes hardware and software specifications of the project.

CHAPTER 5

DESIGN

Modular design is a design approach that subdivides a system into smaller parts called modules or skids, that can be independently created and then used in different systems.A modular system can be characterized by functional partitioning into discrete scalable, reusable modules; rigorous use of well-defined modular interfaces; and making use of industry standards for interfaces.

The modular design will describe the various components used in this project. Fig.5.1 shows the different modules present in the project.

5.1 Modular Diagram

Modular diagrams are used to show the allocation of classes and objects to modules in the physical design of a system, that is modular diagrams indicate the partitioning of the system architecture.Through these diagrams it is possible to understand the general physical architecture of a system.

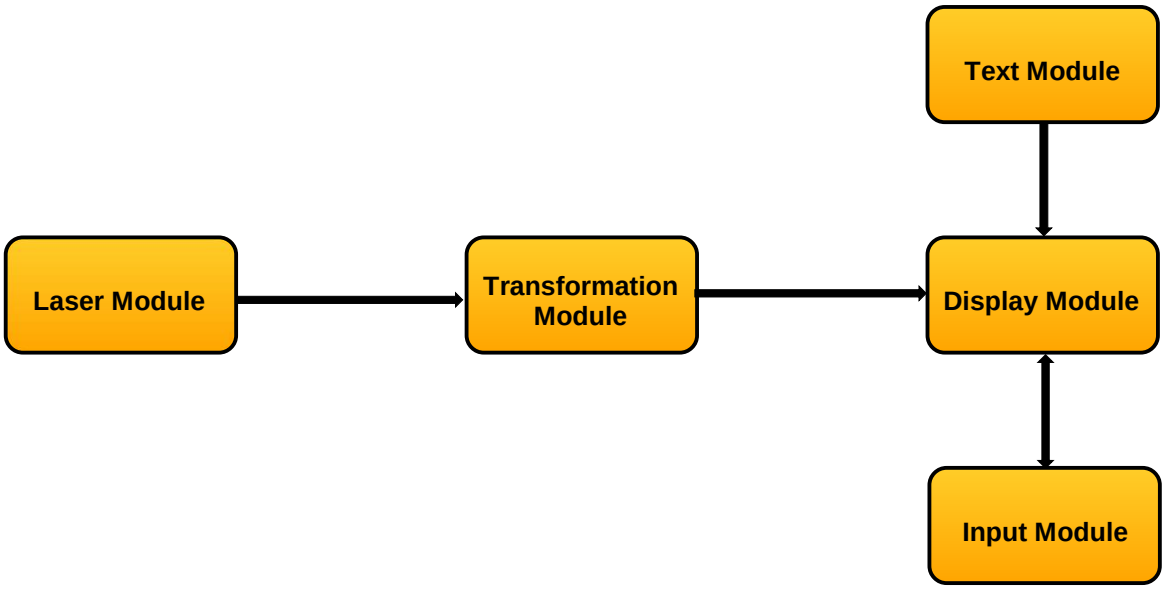

Fig.5.1 Modular diagram

Laser Module:

The Laser module simulates the principles and working that are involved in the emission of Laser Beam light.

Transformation Module:

The Transformation module describes the simulation of the Laser i.e., animation of laser and visualization of laser beam.

Text Module:

The Text module gives the text information of the four principles involved in the production of Laser light.

Display Module:

Display Module displays the objects involved in the visualization of Laser and also takes the text information from text module and is displayed

Input Module:

Input Module takes the input from Keyboard and Mouse for displaying the entities in display module based on user specification.

5.2 Flowchart

Flowchart is a graphical representation of a computer program in relation to its sequence of functions. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

Fig.5.2 gives a visual representation of the sequence of steps and decisions needed to perform a process in the proposed project.

In the Fig.5.2 each rectangular box represents a scene. The first scene is the introduction page. On right clicking the mouse a menu will pops up. The main menu in the proposed project has got three options. On selecting the first menu option introduction page will display on the screen. On selecting the second menu option i.e. "Laser Simulation", a scene will be rendered on the screen which shows a graphical representation of the proposed project. The third menu option is EXIT, which causes immediate termination of the entire program.

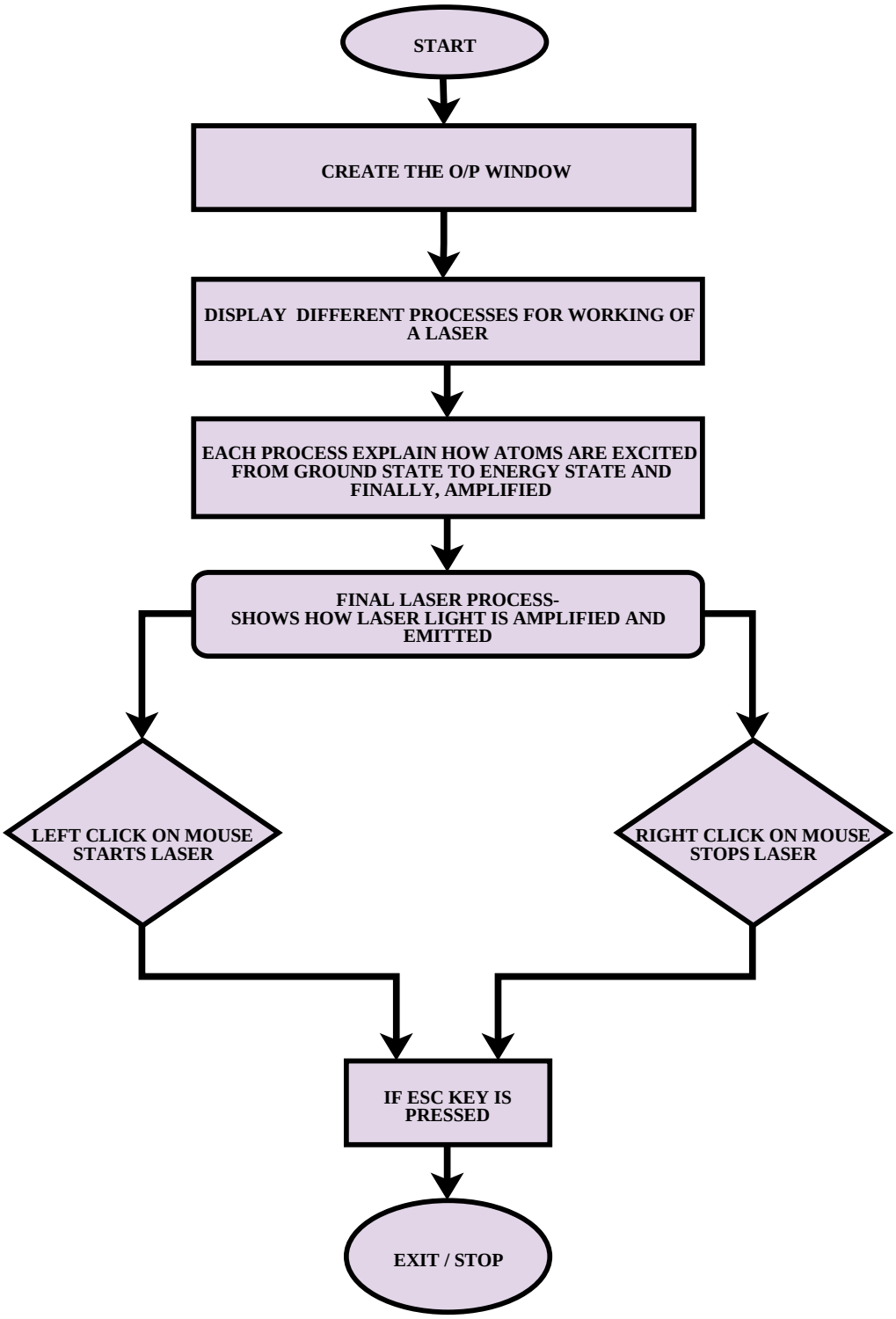

Fig 5.2 Flowchart

Summary

This chapter describes the modular design and flowchart of our project.

CHAPTER 6

IMPLEMENTATION

Implementation is the core step in software development life cycle. Implementation gives the detailed view of the project and describes the pseudo code and various important functions in the project. It will also give insight about various inbuilt modules and functions in OpenGL.

6.1 Algorithm

An algorithm is a step-by-step instruction to execute the program. The following figure 6.1 shows the algorithm of the proposed project.

Algorithm 6.1- High level view of the project

Step 1*: Initialize the window.*

Step 2*: Set the viewport.*

Step 3*: Display the front screen.*

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

Step 5*: if welcome screen is chosen, the welcome page with textual info will be displayed.*

Step 6*: if Simulation is chosen, the principles and working of laser is displayed.*

Step 7*: if 'r' keyboard button pressed laser simulation starts.*

Step 8*: if 's' keyboard button pressed laser simulation stops.*

Step 9*: if Exit is chosen, the display window closes.*

6.2 Code Snippet and Module Implementation

Implementation is the carrying out, execution, or practice of a plan, a method, or any design for doing something. As such, implementation is the action that must follow any preliminary thinking in order for something to actually happen. In an information technology context, implementation encompasses all the processes involved in getting new software or hardware operating properly in its environment, including installation, configuration, running, testing,

and making necessary changes. The word deployment is sometimes used to mean the same thing.

6.2.1 Code Snippet 1: It is for displaying the text.

```
void display1(void)     //front page
{
        glClear(GL_COLOR_BUFFER_BIT);
        glClearColor(0.0,0.0,0.0,0.0);
        glColor3f(1,0.1,1);
        output(180,440,"Mangalore Institute of Technology and Engineering",font0);
        glColor3f(1,0,0.5);
        output(205,375,"Computer Graphics And Visualization Lab",font);//normal font
        output(710,520,"",font);
        glColor3f(1,0,0);
        output(230,330,"A Mini Project on",font);
        glColor3f(1,0.5,0);
        output(218,300,"LASER SIMULATION",font0);
        glColor3f(1.0,0.0,1.0);
        output(50,250,"By: ",font);
        glColor3f(1.0,1.0,0.0);
        output(50,230,"Dhavan Rao B V              ",font);
        output(50,210,"Krishnamoorthy K Bhat           ",font);
        output(110,230,"      4MT15CS031",font);
        output(110,210,"      4MT15CS051",font);

        glColor3f(0.0,1.0,1.0);
        output(205,170,"Press ENTER key to view simulations",font);
        glColor3f(0.0,1.0,1.0);
        output(218,140,"Right click for the MENU",font);

         glColor3f(1.0,0.0,1.0);
        output(335,250,"Submitted to: ",font);
         glColor3f(1.0,1.0,0.0);
        output(335,230,"Mr.John Prakash",font);
        output(335,210,"Mr.Prashanth B.S",font);
        output(400,230,"      Sr.Asst.Professor",font);
        output(400,210,"      Asst.Professor",font);
        glColor3f(0.0,1.0,1.0);
        glutSwapBuffers();
        glFlush();
}
```

6.2.2 Code Snippet 2: It is for displaying the certain entities in the Display Window.

```
void icon()
{
    glColor3f(0.0,0.0,0.3);
    glBegin(GL_POLYGON);
    glVertex2d(20,303);
    glVertex2d(120,303);
    glColor3f(1.5,0.2,0.5);
    glVertex2d(120,310);
    glVertex2d(20,310);
    glEnd();

}
```

6.2.3 Code Snippet 3: It is to take input from input devices such as keyboard & mouse.

```
void keys(unsigned char key,int x,int y)
{
   if(flag==0 && key==13) //Ascii of 'enter' key is 13
     glutIdleFunc(display);


    if(key=='Q' || key=='q' || key==27)
    exit(1);

    if(key=='L'|'A'|'S'|'E'|'R')
    {
            a=1;
            glutPostRedisplay();
    }

}
```

6.2.4 Code Snippet 4: It is to display the laser module.

```
void quad(int x1,int y,int x2,int y2,int x3,int y3,int x4,int y4)
{
        glClearColor(0.0,0.0,0.0,0.0);
        glColor3f(1.0,5.0,0.0);
        glBegin(GL_LINE_LOOP);
        glVertex2i(x1,y);
        glVertex2i(x2,y2);
```

```
        glVertex2i(x3,y3);
        glVertex2i(x4,y4);
        glEnd();
        glFlush();

}


void laser()
{
   GLint i;
   for(i=0;i<30;i+=3)
   {
      quad(x1+i,y+i,x2+i,y2,x3+i,y3,x4+i,y4+i);

    }

     glBegin(GL_POLYGON);
     glColor3f(1.0,0.0,0.0);
     glVertex2f(350,170);
     glVertex2f(590,170);
     glVertex2f(590,175);
     glVertex2f(350,175);
     glEnd();
}


void sphere()
{

     glPushMatrix();
     glColor3f(0.0,0.0,5.0);
     glTranslated(345,195,10);
     glScaled(2,4,6);
     glutSolidSphere(2,10,10);
     glPopMatrix();
}
```

6.2.5 Code Snippet 5: It is used to animate the laser module.

```
void spin()
{
theta[axis]+=0.2;
if(theta[axis]>360.0)
```

```
theta[axis]-=360.0;
glutPostRedisplay();
}
```

6.3 Functions

Headers used

The in-built header files 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.

6.3.1 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.

6.3.2 User Defined Functions

- **icon():-** displays the absorption and spontaneous emission principles of Laser.
- **quad(int x1,int y,int x2,int y2,int x3,int y3,int x4,int y4):-**To draw the rectangular shape.
- **laser():-**This function is used to display the laser beam.
- **sphere():-**this function is used to draw the atoms inside the laser model.
- **spin():-**this function is used to animate the Laser model.
- **write(char *string):-**this function is used to display the text characters in the specified manner.
- **display():-**this function displays the required textual information for the particular entity in the display window.
- **drawstring(float x,float y,float z,const char *str):-**this function is used to display the string.

- **keys(unsigned char key, int x,int y):-**by clicking some keyboard buttons certain actions are performed.
- **mymenu(int id):-**this function is used to display the menu with options.

Summary

This chapter contains the Implementation part of the project. It includes the algorithm, different inbuilt OpenGL functions and user defined functions used in the proposed project.

CHAPTER 7

TESTING

Testing is an important phase in the development life cycle of the product. During the testing, the program to be tested was executed with a set of test cases and the output of the program for the test cases was evaluated to determine whether the program is performing as expected. Errors were found and corrected by using the following testing steps and correction was recorded for future references. Thus, a series of testing was performed on the system before it was ready for implementation. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

7.1 Testing Levels

Testing is part of Verification and Validation. Testing plays a very critical role for quality assurance and for ensuring the reliability of the software.

The objective of testing can be stated in the following ways.

- A successful test is one that uncovers as-yet-undiscovered bugs.
- A better test case has high probability of finding un-noticed bugs.
- A pessimistic approach of running the software with the intent of finding errors.

Testing can be performed in various levels like unit test, integration test and system test.

7.1.1 Unit Testing

Unit testing tests the individual components to ensure that they operate correctly. Each component is tested independently, without other system component. This system was tested with the set of proper test data for each module and the results were checked with the expected output. Unit testing focuses on verification effort on the smallest unit of the software design module.

In our project after every module is prepared, we debug it and ensure its proper functioning.

7.1.2 Integration Testing

Integration testing is another aspect of testing that is generally done in order to uncover errors associated with the flow of data across interfaces. The unit-tested modules are grouped together and tested in small segment, which makes it easier to isolate and correct errors. This approach is continued until we have integrated all modules to form the system as a whole.

In our project it is applied as this. Once we have done with each module we combine it with the previous module and ensure that the project is working properly.

7.1.3 System Testing

System testing tests a completely integrated system to verify that it meets its requirements. Once we have prepared all the module we have combined them and tested that the entire project is working properly.

7.1.4 Acceptance Testing

We have tested the system at different levels. At each level we found that it was working properly and was meeting the requirements which are specified in the requirement analysis.

7.2 Test Cases

A test case is a software testing document, which consists of events, action, input, output, expected result and actual result. Technically a test case includes test description, procedure, expected result and remarks. Test cases should be based primarily on the software requirements and developed to verify correct functionality and to establish conditions that reveal potential errors.

Individual PASS/FAIL criteria are written for each test case. All the tests need to get a PASS result for proper working of an application.

Test Case 1: Display the Laser Principle and Working screen

Objective: To display the Laser working and principle screen

Steps: The following steps have to be followed to carry out test.

1. Click the mouse right button to display the menu.
2. Select the Simulation option to display the principles and working of Laser.
3. The specified screen will be displayed.

Expected Result: The screen displaying successfully.

Result: Successful.

Test Case 2: Start Laser Simulation

Objective: To display the animation of Laser Simulation.

Steps:

1. Press the keyboard button ‘r’.

2. The simulation of Laser starts.

Expected Result: On pressing 'r' successfully the Laser simulation starts.

Result: Successful.

Test Case 2: Stop Laser Simulation

Objective: To stop the Laser Simulation

Steps:

1. Press the keyboard button 's'
2. The simulation of Laser stops.

Expected Result: On pressing 's' successfully the Laser simulation stops.

Result: Successful.

Test Case 4: Exit from the Display Window

Objective: To Exit from the Display window

Steps:

1. Click the mouse right button to display the menu.
2. Select the Exit option to come out of the Display window.

Expected Result: Exit from Display window.

Result: Successful.

The above mentioned test Cases are summarized as shown in Table 7.2.

Table 7.2 Test Cases

Test Case Number	Test Case	Expected Results	Status
1	Display the Laser Principle and Working screen	The screen displaying successfully.	Displayed
2	Start Laser Simulation	On pressing 'r' the Laser simulation starts successfully.	Displayed

3	Stop Laser Simulation	On pressing ‘s’ the Laser simulation stops successfully.	Displayed
4	Exit from the Display Window	Exit from Display window	Exited

Summary

This chapter contains the Testing phase of the project. It includes the details of different test levels and test cases.

CHAPTER 8

RESULT AND SNAPSHOT

This section includes the results and snapshots of the project. Also it contains the brief description of each snapshot.

Fig8.1-Welcome screen

Figure 8.1 shows the welcome screen of the project which describes the project title, team members, faculty guides .

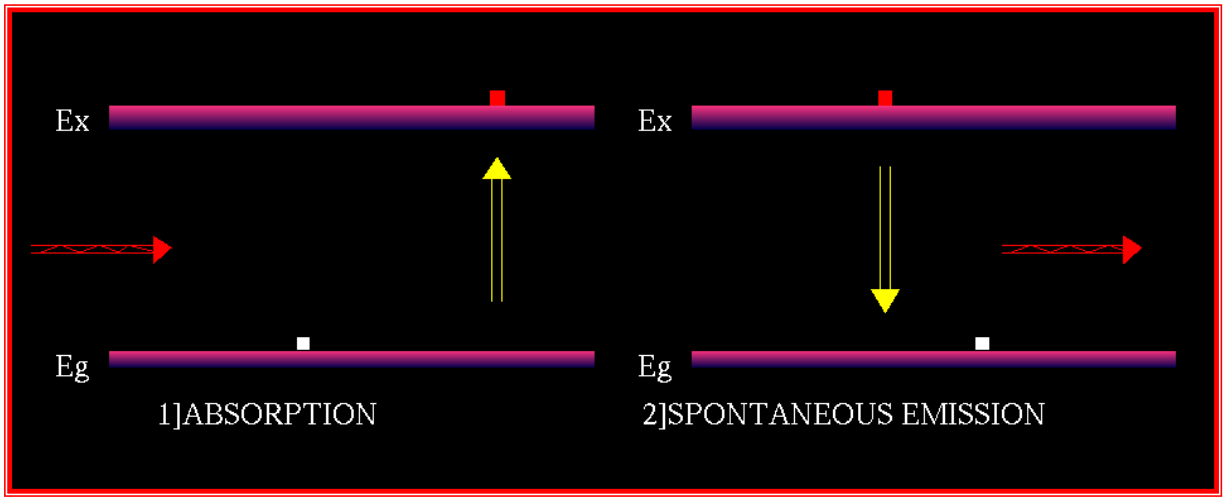

Fig8.2-Absorption and spontaneous emission

Figure 8.2 Illustrates the Absorption & Spontaneous Emission i.e,Electrons in the ground state absorbs the energy from photons to jump into higher energy level.In Spontaneous emission,electrons in the excited state return to the ground state by emitting the photon.

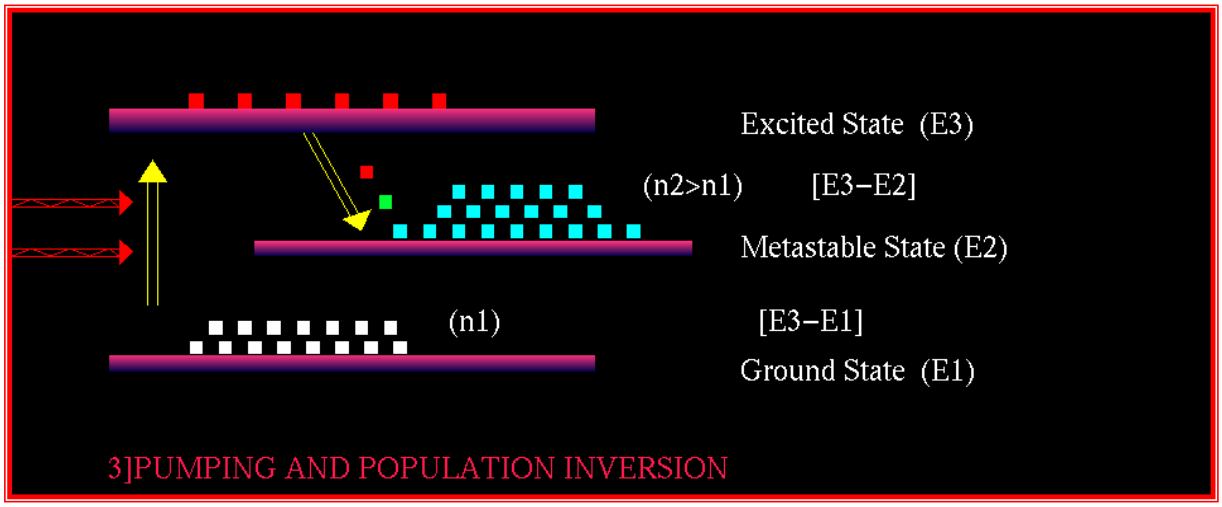

Fig8.3-Pumping and Population Inversion

Figure8.3 Illustrates the laser pumping i.e,act of energy transfer from external source to the gain medium of a laser.The energy is absorbed in the medium,producing the excited states in its atoms when the no. of particles in one excited state exceeds the no. of particles in the ground state or less excited state, population inversion is achieved.

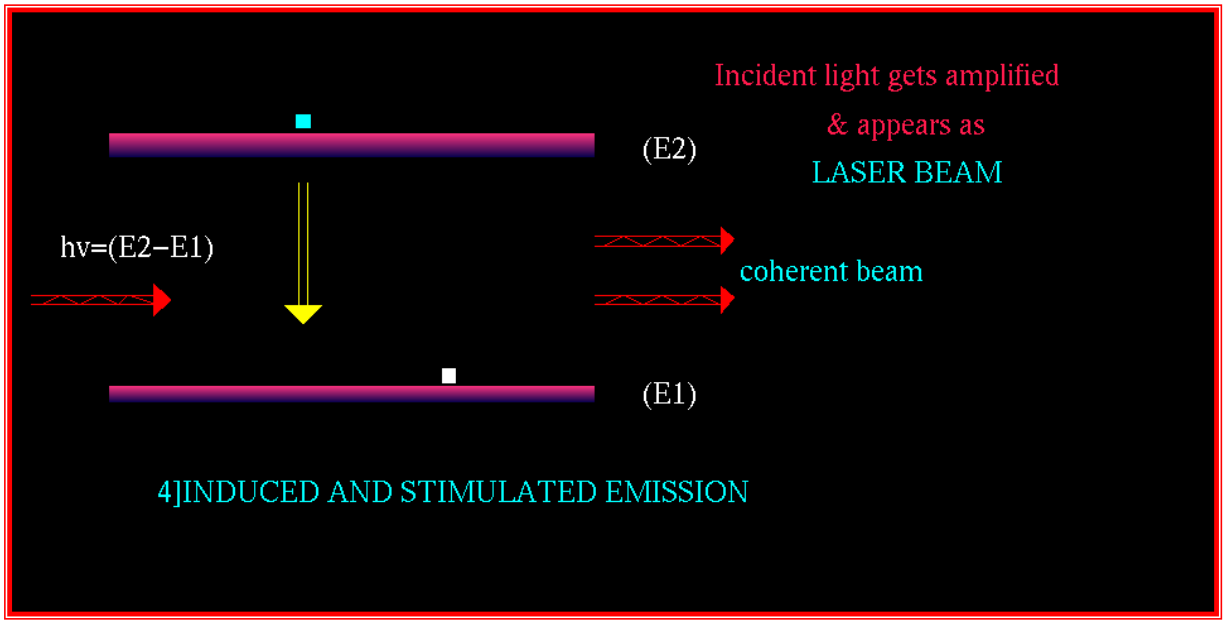

Fig8.4-Induced and stimulated emission

Figure 8.4 Illustrates the stimulated emission i.e, the process by which incoming photon of a specific frequency can interact with the excited atomic electron, causing it to drop to a lower energy level(E1).

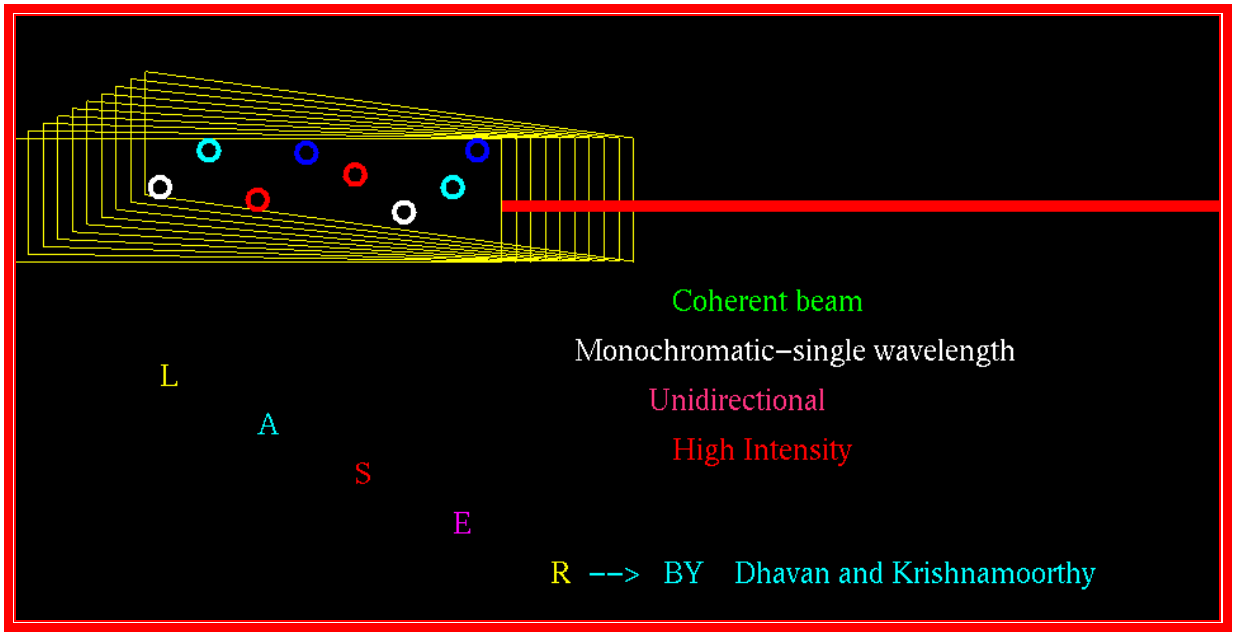

Fig8.5-Working of laser

Figure 8.5 Illustrates the simulation & production of the laser beam.

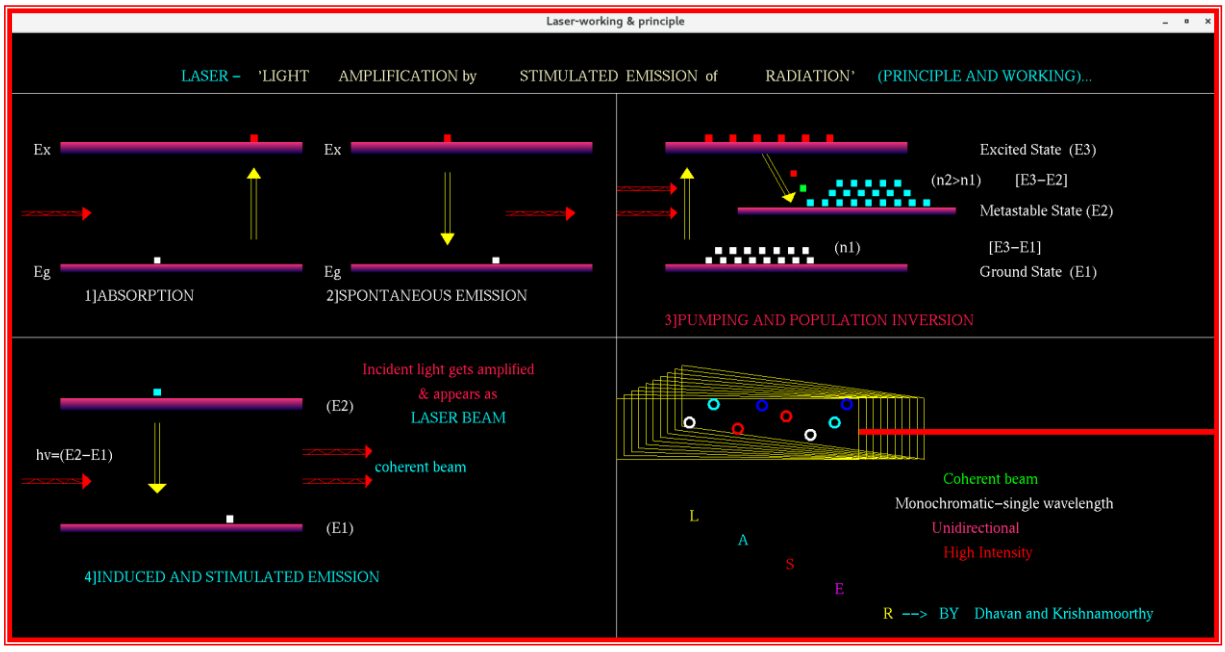

Fig8.6-Principles and working

Fig8.6-Illustrates all the principles & working of Laser together.

Summary

This section includes the results and snapshots of the project. It gives point description of each snapshot.

CHAPTER 9

USER MANUAL

These are the steps to use this project:

1) Debug the project as usual.
2) A screen will appear which the introduction page of our project.
3) Press Enter to go or click right button of the mouse and select the simulation option to go to the next module.
4) Now our first module will appear. In that as we can see the principle and working of Laser.
5) Now press ‘r’ keyboard button to start the laser simulation.
6) Press ‘s’ keyboard button to stop the laser simulation.
7) Press Esc or click right button of the mouse and select the EXIT option to close the Display window.

Summary

This Chapter is about the User Manual of the project. It specifies the steps that guides the user to look through the project without any difficulty.

CHAPTER 10

CONCLUSION AND FUTURE ENHANCEMENT

This section describes the conclusion and future work of the project.

10.1 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 "LASER SIMULATION" 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. On conclusion this mini project is implemented with the standard OpenGL functions.

10.2 Future Enhancement

This project is prepared using 2D technique. As a continuation we can develop this project using 3D technique with advanced functions of openGL. We can also develop and demonstrate the 3D Laser model in high level.

Summary

This section includes the conclusion and future work of the project.