## VISVESVARAYA TECHNOLOGICAL UNIVERSITY BELGAUM - 590018, KARNATAKA



**COMPUTER GRAPHICS LABORATORY**

**MINI PROJECT REPORT ON**

**“INTERFERENCE OF LIGHT”**

Submitted in fulfillment of requirement for the Computer Graphics Laboratory of

**BACHELOR OF ENGINEERING IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by**

**Prajwal DP Prajwal V**

**(1ST19CS096) (1ST19CS097)**

Under the guidance

of

#### Prof. POOJA M V

Assistant Professor, Dept. of CSE.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Sambhram Institute of Technology**

**BENGALURU-56009**

SAMBHRAM INSTITUTE OF TECHNOLOGYM.S.PALYA ,BANGALORE-97

 (Affiliated to Visvesvaraya Technological University, Belgaum, Karnataka) DEPARTMENT OF COMPUTER SCIENCE & ENGINEE RING

**CERTIFICATE**

This is to certify that **PRAJWAL DP(1ST19CS096)** and **PRAJWAL V(1ST19CS097)** has satisfactorily submitted **Computer Graphics Laboratory** Mini Project Report titled **“INTERFERENCE OF LIGHT”** in fulfillment of the requirements as prescribed by the **V.T.U** for **6th semester, Bachelor of Engineering in “COMPUTER SCIENCE & ENGINEERING”**, during the academic year 2020-21.

The project report has been approved as it satisfies the academic requirements in respect to the project word prescribed for the Bachelor of Engineering degree.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Mrs. POOJA M V Dr. T John Peter**

Assistant Professor, **Head, Dept. of CSE, SaIT**

**Dept. of CSE, SaIT**

**External Viva:**

**Name of the Examiners Signature with Date**

**1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ACKNOWLEDGEMENT**

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

We express my sincere gratitude to our principal **Dr.H.G Chandrakanth Principal**, **Sambhram Institute of Technology, Bangalore** for his inspiration.

We would also like to thank our guide **Prof.** **SHEELA RANI C M and Prof. POOJA M V Assistant prof, Dept of Computer Science and** E**ngineering**, for the timely inspection and guiding us throughout the process.

Also we thank the members of the faculty of **Computer science and engineering** department whose suggestions enabled us to surpass many of the seemingly impossible hurdles.

#### PRAJWAL DP(1ST19CS096) PRAJWAL V(1ST19CS096)

**ABSTRACT**

Computer Graphics has a very important role in the field of computer science. This is an effective and rapid communication link that provides between the man and machine. The human eye can observe the information content of displayed diagram or perspective view much faster than it can scan a page of words of contents. Open Graphics Library is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. It is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation.

**“INTERFERECE OF LIGHT”** is created using OpenGL. It demonstrates features such Light Source falls on the Double slit and Frindges are formed on the screen due to Interference.

# LIST OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **ACKNOWLEDGEMENT** |  | **i** |
| **ABSTRACT** |  | **ii** |
| **LIST OF FIGURES** |  | **v** |
| **CHAPTER NO.** | **CHAPTER NAME** | **PAGE NO.** |
| **1** | **INTRODUCTION** | **1-6** |
| 1.1 | Definitions | 1 |
| 1.2 | Uses of Computer Graphics | 1 |
| 1.3 | Advantages | 2 |
| 1.4 | Open GL | 2 |
| 1.5 | Basic Graphic System | 4 |
| **2** | **REQUIREMENT SPECIFICATION** | **6** |
| 2.1 | Hardware Requirement | 6 |
| 2.2 | Software Requirement | 6 |
| **3** | **PROJECT DESIGN** | **7-9** |
| 3.1 | Description | 7 |
| 3.2 | Flow Chart | 9 |

1. FUNCTION DEFINITIONS AND THEIR SYNTAX 10-15
   1. Attributes 10
   2. Working with the Window System 10
   3. Interaction 12
   4. Transformation 12
   5. [Viewing 15](#_TOC_250000)

#### IMPLEMENTATION 16-20

#### SNAPSHOTS 21-23

#### CONCLUSION 24

#### FUTURE ENHANCEMENT 25

#### BIBLIOGRAPHY 26

**LIST OF FIGURES**

**FIGURE NO: TITLE PAGE NO.**

* + 1. OpenGL Architecture 3
    2. Library Organization 4

1.5 Basic Graphics System 4

3.3 Flow Chart 9

* + 1. Viewing Transformations 14

|  |  |  |
| --- | --- | --- |
| **SNAPSHOT NO.** | **TITLE** | **PAGE NO.** |
| 6.1 | Introduction Page | 21 |
| 6.2 | Initial Scene | 21 |
| 6.3 | Second scene | 22 |
| 6.4 | Interference Pattern | 22 |
| 6.5 | Fringes | 23 |

**CHAPTER 1**

### DEFINITION:

**INTRODUCTION**

Computer graphics is concerned with all aspects of providing pictures or images using a computer. The field began humbly almost 50 years ago, with the display of a few lines on a cathode ray tube (CRT).The term “computer graphics” includes almost everything on computers that is not text or sound.

Pixel art is a form of digital art, created through the use of raster graphics software, where images are edited on the pixel level.

### USES OF COMPUTER GRAPHICS:

Applications of computer graphics are many, however some of them are as explained below:

#### User Interfaces:

Word-processing, spreadsheet and desktop-publishing programs are typical applications of such user-interface techniques.

#### Interactive Plotting in Business, Science and Technology:

The common use of graphics is to create 2D and 3D graphs of mathematical, physical and economic functions, histograms, and bar and pie charts.

#### Computer Aided Drafting and Design (CAD):

In CAD, interactive graphics is used to design components and systems of mechanical, electrical and electronic devices including structures such as buildings, automobile bodies, aero planes, ship hulls etc.

#### Simulation & Animation for Scientific Visualization & Entertainment:

Computer-produced animated movies are becoming increasing popular for scientific and engineering visualization. Cartoon characters will increasingly be modelled in the computer as 3D shape descriptions. whose movements are controlled by computer commands.

#### 2D Graphics:

These editors are used to draw 2D pictures (line, rectangle, circle and ellipse) alter those with operations like cut, copy and paste. These may also support features like translation, rotation etc.

#### 3D Graphics:

These editors are used to draw 3D pictures (line, rectangle, circle and ellipse). These may also support features like translation, rotation etc.

### ADVANTAGES:

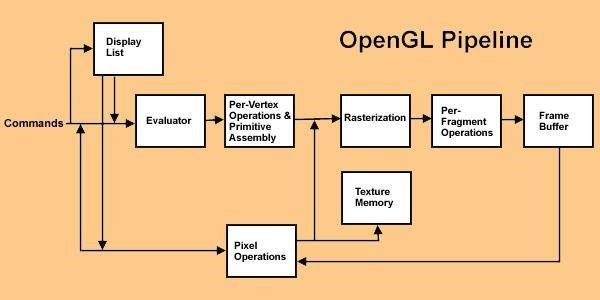
* **Scientific visualization**-it refer to the process of representing raw scientific data as images.
* **Information visualization** –it is the study of interactive visual.
* **Computer vision** –it is the field of study of artificial intelligence that trains computer to interpret and understand the visual world.
* **image processing**- it is analyses and manipulation of digitized image.
* **Computational geometry** –it is branch of computer science devoted to the study of algorithm which can be stated in terms of geometry.
* **Computational topology**- is is a subfield of topology with an overlap with areas of computer science.
* **Applied mathematics** –it is the application of mathematical methods by different fields such as physics, engineering ,business etc.

### OPENGL:

* + - Open Graphics Library is a standard specification defining a cross-language, crossplatform API for writing applications that produce 2D and 3D computer graphics.
    - OpenGL is a low-level, procedural API, requiring the programmer to dictate the exact steps required to render a scene.
    - OpenGL's basic operation is to accept primitives such as points, lines and polygons and convert them into pixels.

#### OPENGL ARCHITECTURE

The OpenGL architecture is structured as a state-based pipeline. Below is a simplified diagram of this pipeline. Commands enter the pipeline from the left.



#### Figure 1.4.1: OpenGL Architecture

Commands may either be accumulated in display lists, or processed immediately through the pipeline. Display lists allow for greater optimization and command reuse, but not all commands can be put in display lists.

The first stage in the pipeline is the evaluator. This stage effectively takes any polynomial evaluator commands and evaluates them into their corresponding vertex and attribute commands.

The second stage is the per-vertex operations, including transformations, lighting, primitive assembly, clipping, projection, and viewport mapping.

The third stage is rasterization. This stage produces fragments, which are series of frame buffer addresses and values, from the viewport-mapped primitives as well as bitmaps and pixel rectangles.

The fourth stage is the per-fragment operations. Before fragments go to the frame buffer, they may be subjected to a series of conditional tests and modifications, such as blending or z-buffering.

Parts of the framebuffer may be fed back into the pipeline as pixel rectangles. Texture memory may be used in the rasterization process when texture mapping is enabled.

#### LIBRARY ORGANISATION IN OPENGL:

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

Xlib,tk

GL

OpenGL application Program

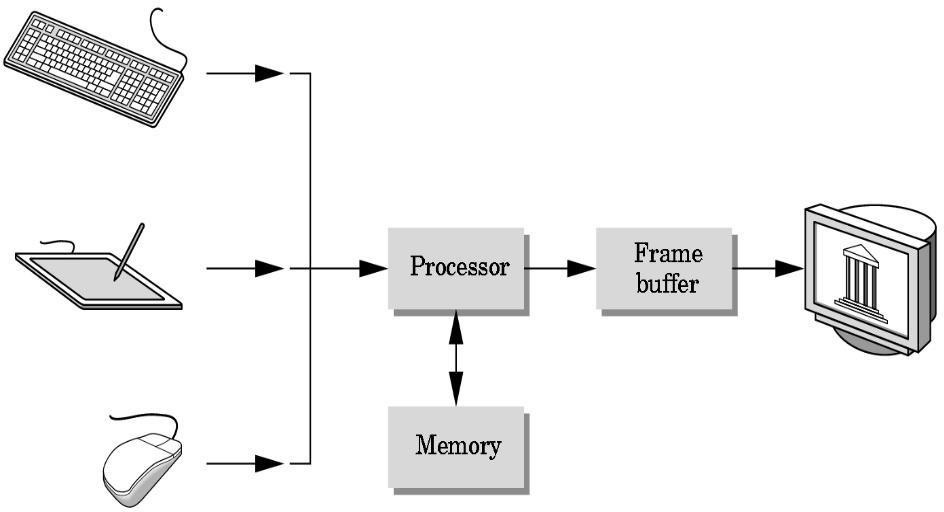
GLU

Frame Buffer

#### Figure 1.4.2.1:Library Organization

GLX

GLUT

* 1. **BASIC GRAPHICS SYSTEM:**

**Figure 1.5.1: Basic Graphics System**

**Major elements in basic graphic system:**

* + - **Input devices :**

Graphics workstations make use of various devices for data input .Most systems have keyboard and mouse ,while some other systems have trackball, joystick, button boxes, touch panels, image scanners and voice systems.

#### Processor:

Graphics processing unit, is used primarily for 3D applications. It is a singlechip processor that creates lighting effects and transforms objects every time a 3D scene is redrawn.

#### Memory:

Memory refers to a design where the graphics chip does not have its own dedicated memory, and instead shares the main system RAM with the CPU and other components.

#### Frame buffer :

A framebuffer is a portion of RAM containing a bitmap that drives a video display. It is a memory buffer containing a complete frame of data. The information in the buffer typically consists of colour values for values every pixel to be shown on the display.

#### Output devices:

Graphics hardware output devices are those hardware that generates computer Graphics and allows them to be shown on a display. Some of the output devices are

Monitors, Printers, Plotters, projectors, LCD Projection Panels ,Speakers etc.

## CHAPTER 2

**REQUIREMENT SPECIFICATION**

### SOFTWARE SPECIFICATION

* + - **Operating System :** Windows 10
    - **Cross Platform API:** Open GL
    - **Programming Language :** C Language
    - **Compiler :** Code Blocks

### HARDWARE SPECIFICATION

* + - **Processor**: intel
    - **RAM:** 4GB
    - **System type:** 64-bit processor
    - **Monitor:** VGA compatible (CRT OR LCD-TFT)

## CHAPTER 3

### DESCRIPTION

**PROJECT DESIGN**

Interference of light waves similar to interference of water waves–two different waves arrive at the observation point–the total influence is the sum of the two wave amplitudes at each time and at each point in space. High frequency of light has important consequences–Cannot follow the fast cycling of the field–Detectors measure the effect of many oscillations–Only interference that persists over many periods is observable. Only two-beam interference is discussed in this module–Multiple beam interference will be treated in next module.

**Detailed description of the project**:

The flow of the project will be like this:

When we debug the program an introduction page will appear. On clicking right button we will go to first module.

First Module:

* This module depicts a simple scenario where there is a light source, double slits screen.
* There is a text referring to the objects.
* Press ‘M’ to go to the next module.
* Press ‘I’ to go to the introduction page

Second Module:

* It is a scenario where the coherent light source will pass through the double slits screen.
* Once we click mouse left button, one more man will come.
* There is a text referring to the objects.
* The instruction to move to the next scene is shown.
* Press ‘K’ to go to the next module.
* Press ‘M’ to go to the previous module.

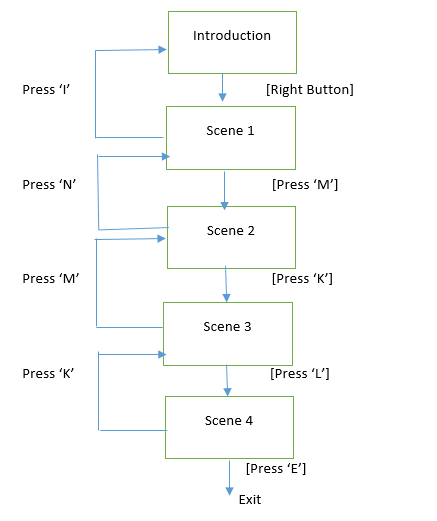
Third Module:

* It is a scenario where the formation of interference pattern is shown.
* The interference pattern formed will fall on the screen immediately the next scene will appear.
* Press ‘M’ to go to the previous module.
* Press ‘I’ to go to the introduction page

Fourth Module:

* This is our final module.
* In this module we show the formation of dark and bright fringes on the screen due to interference.
* Press ‘K’ to go to previous module.
* Press ‘I’ to go to the introduction page

### FLOW CHART



#### fig3.2 flow chart

**The brief description regarding the flow chart drawn above is follow:**

## CHAPTER 4

**FUNCTION DEFINITIONS AND THEIR SYNTAX**

### DEFINITION:

OpenGL (Open Graphics Library) is a cross-platform, hardware accelerated, language independent, industrial standard API for producing 3D (including 2D graphics). The OpenGl graphics rendering commands issued by the applications could be directed to the graphics hardware and accelerated.

### WORKING WITH THE WINDOWS SYSTEM:

In OpenGL, an object is made up of geometric primitives such as triangle, quad, line segment and a point. A primitive is made up of two or more vertices. OpenGL supports the following primitives: A geometric primitive is defined by specifying its vertices using the glVertex( ) function, enclosed within a pair glBegin( ) and glEnd( ).

#### WINDOWS MANAGER SETUP USING GLUT:

* + - * **glutInit(&argc,argv):**

Initializes GLUT, processes any command-line arguments for X functions.

#### glutInitDisplayMode(unsigned int mode):

Sets the basic display modes. Usual mode constants are GLUT\_SINGLE, GLUT\_DOUBLE, GLUT\_RGB, GLUT\_INDEX and GLUT\_DEPTH.

#### glutInitWindowSize(int width, int height):

Sets the size of screen in pixels.

#### glutInitWindowPosition(int x,int y):

Sets the initial window position on the screen for upper left corner.

#### glutCreateWindow(char \*title):

Creates a top-level window labelled with title.

#### glutDisplayFunc(void(\*func)(void)):

Sets the callback function, provided by the program, that GLUT will call when a window must be redisplayed.

#### glutMainLoop( ):

Starts the event loop: never return.

#### WINDOW MANAGER INPUT HANDLING FUNCTIONS:

* **glutReshapeFunc(void(\*func)(int width, int height)):**

Registers the callback function that GLUT calls on window reshape events.

#### glutKeyboardFunc(void(\*func)(unsigned char key, int x, int y):

Registers the callback function that GLUT calls on a keyboard event. The parameter is the ASCII value of the key pressed.

#### glutMouseFunc(void(#func)(int button, int state, int x, int y)):

Registers the callback function that GLUT calls on a mouse event. The parameter button may be GLUT\_LEFT\_BUTTON, GLUT\_RIGHT\_BUTTON or GLUT\_MIDDLE\_BUTTON.

#### glutPostRedisplay(void):

Nudges the main loop to call display( ); sometimes useful after keyboard or mouse input changes some display parameters.

#### glFlush( ):

Flush the graphics output buffer.

#### VIEWING AND MODELLING TRANSFORMATIONS:

* **glMatrixMode(GLenum mode):**

Sets the matrix to be works on. The values for mode are GL\_MODELVIEW, GL\_PROJECTION, GL\_TEXTURE which make use of the modelview, projection, texture matrices respectively.

#### glLoadIdentity(void):

Set the current matrix to identity. .

#### COLOR AND CLEARING THE SCREEN:

* **glClearColor(float red, float green, float blue, float alpha):**

Sets the clear color to red, blue or green with value alpha.

#### glClear(GLbitfield mask):

Used to clear the buffer indicated by mask using clear color. The Values for mask are GL\_COLOR\_BUFFER\_BIT and GL\_DEPTH\_BUFFER\_BIT.

#### glColor3f(float red, float green, float blue):

Sets the current drawing color to red, blue or green.

### INTERACTIONS:

GLUT supports interaction with the computer mouse that is triggered when one of the three typical buttons is pressed. A mouse callback function can be initiated when a given mouse button is pressed or released. The command glutMouseFunc( ) is used to specify the callback function to use when a specified button is in a given state at a certain location.

GLUT interaction using keyboard inputs is handled very similarly to those with the mouse. The command glutKeyboardFunc( ) is used to run a callback function and pass as parameters, the ASCII code of the pressed key.

### TRANSFORMATIONS:

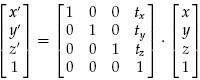
Vertex transformations (such as rotations, translations and scaling) and projections (such as perspective and orthographic) can all be represented by applying an appropriate 4 ×4 matrix to the coordinates representing the vertex. If v represents a homogeneous vertex and M is a 4 × 4transformation matrix, then M v is the image of v under the transformation by M. After transformation, all transformed vertices are clipped so that x, y, and z are within the screen co-ordinates.

Although any non-singular matrix M represents a valid projective transformation, a few special matrices are particularly useful.

#### THREE-DIMENSIONAL TRANSLATION:

A position P = (x, y, z) in three-dimensional space is translated to a location P’= (x’, y’, z’) by adding translation distances tx, ty, and tz to the Cartesian coordinates of P:

We can express these three-dimensional translation operations in matrix form



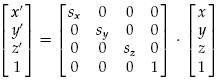
P`=T.P glTranslate (x, y, z) function is used.

Moving a coordinate position with translation vector T = (tx , ty , tz ) .

#### THREE-DIMENSIONAL SCALING:

The matrix expression for the three-dimensional scaling transformation of a position

**P** = (x, y, z) is given by



The three-dimensional scaling transformation for a point position can be represented as P`=S.P where scaling parameters sx, sy, and sz are assigned any positive values. Explicit

expressions for the scaling transformation relative to the origin are

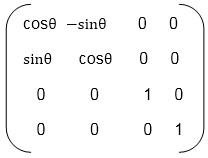
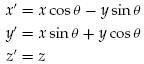
x`= x. sx y`= y. sy z`=z. sz

glScale(x, y, z) function is used

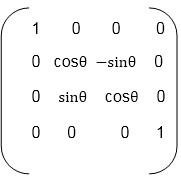
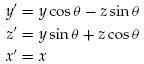
#### THREE-DIMENSIONAL ROTATION:

Rotation is moving of an object about an angle. Movement can be anticlockwise or clockwise.3D rotation is complex as compared to the 2D rotation. For 2D we describe the angle of rotation, but for a 3D angle of rotation and axis of rotation are required. This axis can be either x or y or z.

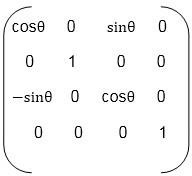
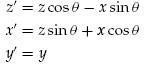
Matrix for representing three-dimensional rotations about the Z axis



Matrix for representing three-dimensional rotations about the X axis



Matrix for representing three-dimensional rotations about the Y axis



### VIEWING:

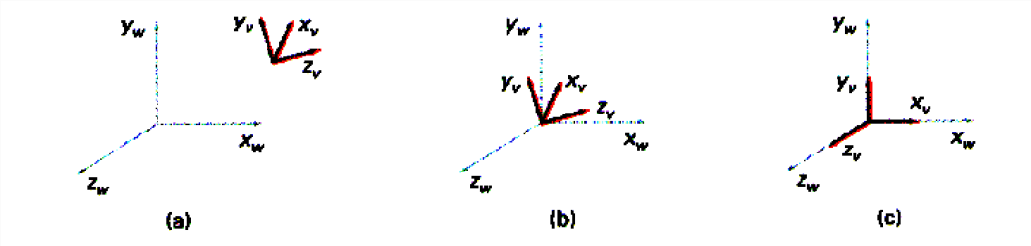
Viewing in 3D involves the following considerations:

* We can view the object in any spatial positions.
* 3D description of objects must be projected onto the flat viewing surface of the output device.
* The clipping boundaries enclose a volume of space.

#### VIEWING TRANSFORMATION:

Conversion of objection descriptions from world to viewing coordinates is equivalent to a transformation that superimposes the viewing reference frame onto the world frame using the basic geometric translate-rotate operations:

* Translate the view reference point to the origin of the world-coordinate system.
* Apply rotations to align the xv, yv, and zv axes (viewing coordinate system) with the world xw, yw, zw axes, respectively.



#### Fig 4.5.1: Viewing Transformation

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 Module Implementation**

The basic idea underlying modular design is to organize a complex system as a set of distinct components that can be developed independently and then plugged together.

**5.1.1 Module 1**

Module 1 is the code for generating initial setup.

**}**

*void scene1()*

*{*

*drawstring(50.0,30.0,0.0,"Light Source");*

*drawstring(900.0,30.0,0.0,"Double slits");*

drawstring(50,670,0.0,"Press 'M' for next scene");

screen1();

light1();

glutPostRedisplay();

glFlush();

}

**}**

**}**

**5.1.2 Module 2**

Module 2 is the code which generates the scene of coherent light source passing through the slits.

*void scene2()*

*{*

*drawstring(50,670,0.0,"Coherent light source passing …");*

*drawstring(50,640,0.0,"Press 'K' for next scene");*

*drawstring(50.0,30.0,0.0,"Light Source");*

*drawstring(900.0,30.0,0.0,"Double slits");*

*light2();*

*screen2();*

*glutPostRedisplay();*

*glFlush();*

**5.1.3 Module 3**

Module 3 is the code for generating interference pattern in the scene.

*void scene3()*

*{*

*drawstring(50.0,20.0,0.0,"Double slits");*

*screen5();*

screen3();

*screen3();*

*glColor3f(0.999988897,0.99999888,0.6568797967);*

*for(i=10;i<800;i=i+20)*

*{*

*concentric\_circles(90,420,i);*

*concentric\_circles(90,300,i);*

*for(k=0;k<10000000;k++);*

}

*keys('l',0,0);*

*glFlush();*

}

**5.1.4 Module 4**

Module 4 is the code for generating dark and bright fringes on the screen.

*void scene4()*

*{*

*int i,k,l;*

*drawstring(50,670,0.0,"Fringes formed on the screen due to INTERFERENCE"); drawstring(1000,30,0.0,"Screen");*

*drawstring(50.0,20.0,0.0,"Double slits");*

drawstring(1050.0,660.0,0.0,"Dark & Bright fringes");

screen5();

screen4();

for(i=10;i<800;i=i+20)

{

concentric\_circles(90,420,i);

concentric\_circles(90,300,i);

for(k=0;k<10000;k++)

for(l=0;l<1000;l++);

}

glFlush();

glutPostRedisplay();

glFlush();

}

**5.1.5 Module 5**

Module 5 is the code for generating the concentric circles.

void concentric\_circles(float x,float y,int i)

{

float angle,PI = 3.14;

float step=0.1,rad1,rad2;

glBegin(GL\_POINTS);

for(angle=0.0;angle<=60; angle+=step)

{

rad1= PI\*angle/180;rad2=PI\*(360-angle)/180;

glVertex2f(x+i\*cos(rad1),y+i\*sin(rad1));

glVertex2f( x+i\*cos(rad2),y+i\*sin(rad2));

}

glEnd();

glFlush();

}

**5.1.6 Module 6**

Module 6 is the code for generating the light.

void light1()

{

polygon(100,80,80,110,180,160,200,130);

polygon(180,160,200,130,260,130,215,210);

stand2();

}

.

.

**5.1.7 Module 7**

Module 7 is the code for generating the stand.

*void stand()*

*{*

*glLineWidth(10.0);*

*glColor3f(0.545098939,0.137254902,0.137254902);*

*glBegin(GL\_LINES);*

*glVertex2f(980,490);*

*glVertex2f(980,70);*

*glVertex2f(1170,360);*

*glVertex2f(1170,40);*

*glVertex2f(950,70);*

*glVertex2f(1010,70);*

glVertex2f(1140,40);

glVertex2f(1200,40);

glEnd();

}

**5.1.8 Module 8**

Module 8 is the code for generating the polygons.

*void polygon(float x1,float y1,float x2,float y2,float x3,float y3,float x4,float y4)*

*{*

*glBegin(GL\_POLYGON);*

*glVertex2f(x1,y1);*

*glVertex2f(x2,y2);*

*glVertex2f(x3,y3);*

glVertex2f(x4,y4);

glEnd();

}

.

.

**5.1.9 Module 9**

Module 9 is the code for generating fringes on the screen.

*void screen4()*

*{*

*stand();*

*glColor3f(0.376470588235,0.376470588235,0.376470588235);*

*polygon(900,350,900,650,935,640,935,340);*

*glColor3f(1.0,843137254902,0.0);*

*polygon(935,640,935,340,970,330,970,630);*

*glColor3f(0.376470588235,0.376470588235,0.376470588235);*

*polygon(970,330,970,630,1005,620,1005,320);*

*. . . . .*

}

**5.1.10 Module 10**

Module 10 is the code for generating the screen.

*void screen1()*

*{*

*stand();*

*polygon(900,650,1250,550,1250,250,900,350);*

*polygon(1000,575,1030,565,1030,370,1000,380);*

*polygon(1150,530,1120,540,1120,345,1150,335);*

……..

}

## CHAPTER 6

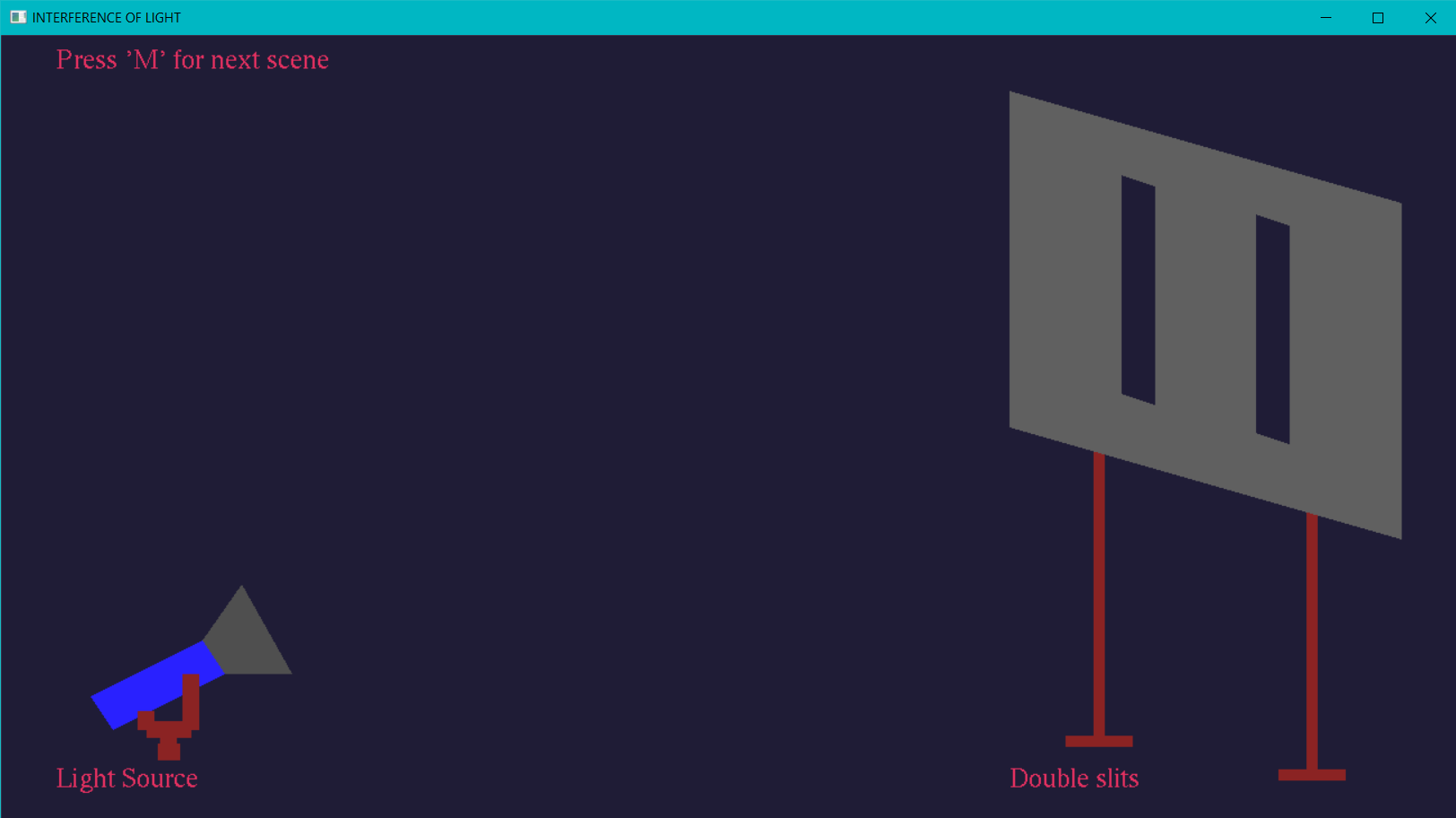
**SNAPSHOTS**

### INTRODUCTION

**Fig 6.1: Introduction page**

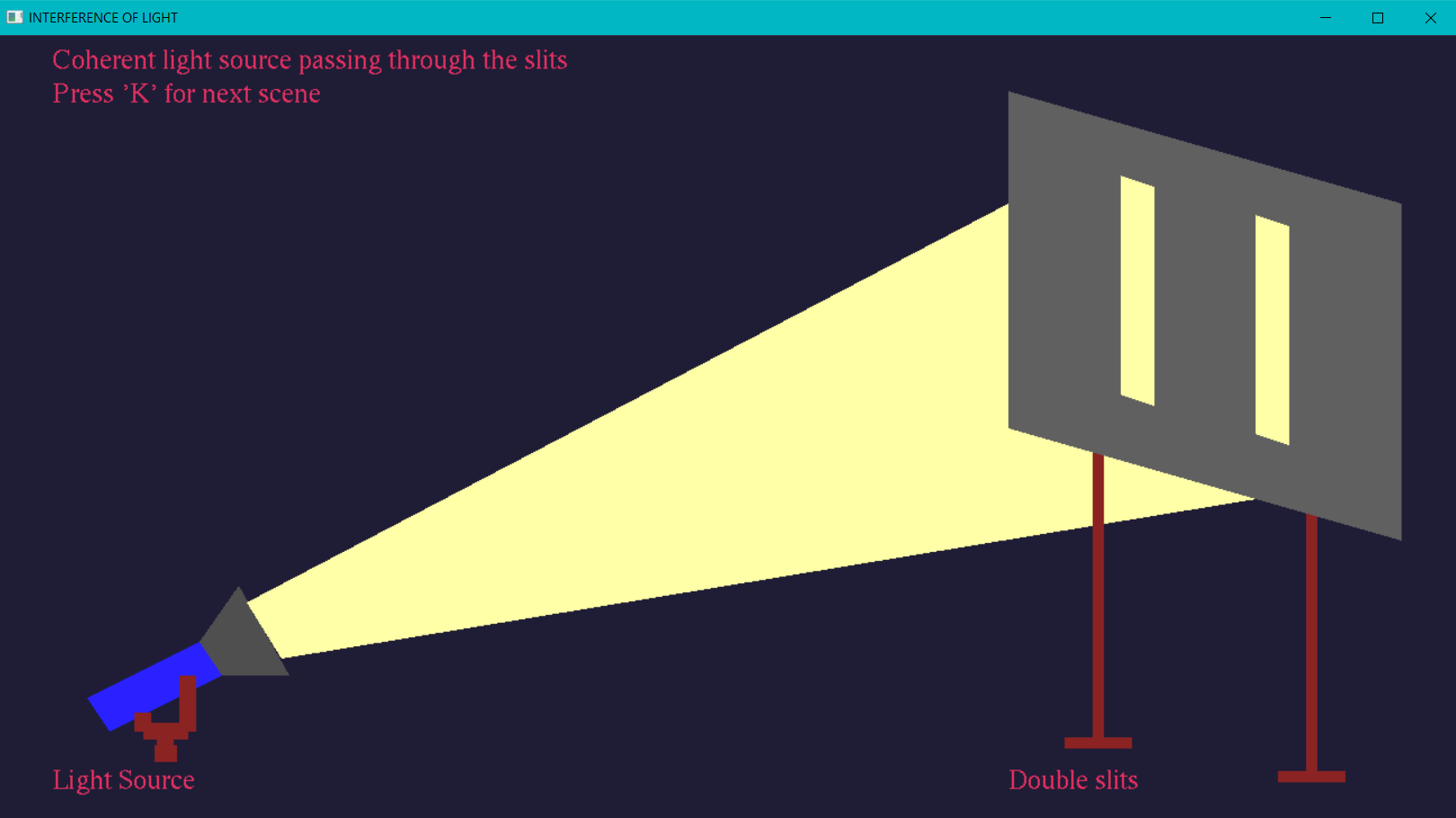
#### 

* 1. **INITIAL SCENE**



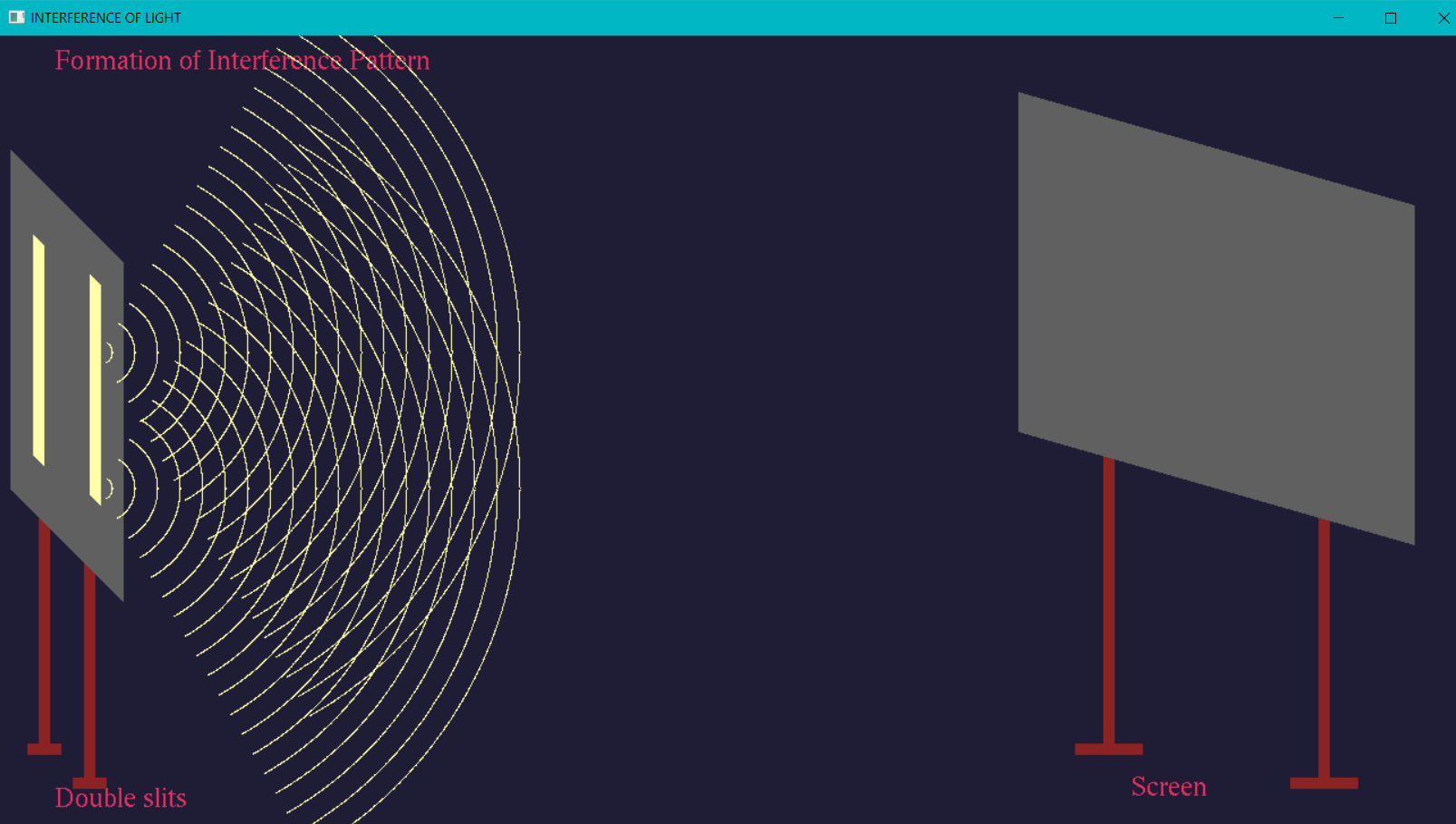
**Fig6.2: Intitial scene of interference**

* 1. **SECOND SCENE**

****

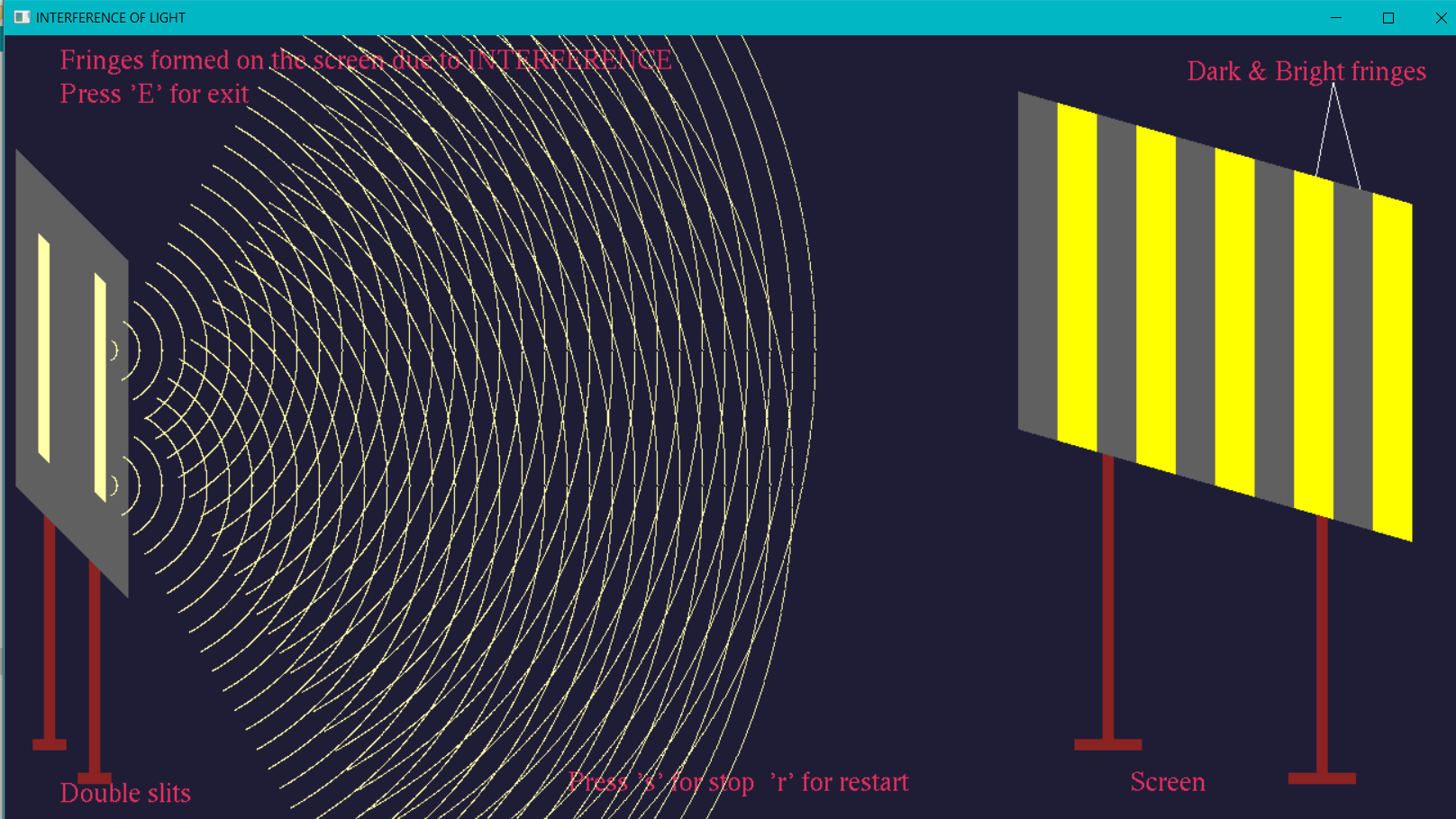
**Fig 6.3: Light source falls on Double Slits**

* 1. **INTERFERENCE PATTERN**

****

**Fig 6.4: Formation of interference**

* 1. **FRINGES**

****

**Fig 6.5: Fringes formation**

**CHAPTER 7**

**CONCLUSION**

The development of this project would improve the user’s knowledge about computer graphics and OpenGL. In this project, we have tried to show phenomenon of Interference of light using computer graphics. We have made use of different functions of OpenGL for the generation and transformation of different entities in the project.

# CHAPTER 8

**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 implement lighting and shading effect in this project.

# BIBLIOGRAPHY

### WEBSITES:

* + - [https://www.opengl.org](https://www.opengl.org/)
    - [https://docs.microsoft.com](https://docs.microsoft.com/)
    - [https://learnopengl.com](https://learnopengl.com/)
    - [https://www.javatpoint.com](https://www.javatpoint.com/)

### BOOKS:

* + - Edward Angel: “Interactive Computer Graphics- A Top down Approach with OpenGL”, 5th edition, Pearson Education, 2008.
    - Donald D Hearn and M Pauline Baker: “Computer Graphics with OpenGL”, 4th edition, Pearson Education, 2011.
    - Kelvin Sung, Peter Shirley, Steven Baer: “Interactive Computer Graphics, concepts and applications”, Cengage Learning.