**Chapter 1**

**INTRODUCTION**

**1.1 Introduction to Computer Graphics**

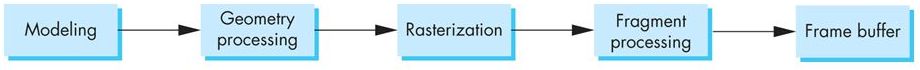
Computer Graphics is concerned with all aspects of producing pictures or images using a computer. The field began humbly almost 50 years ago, with the display of few lines on a cathode-ray tube (CRT); now it can create images by computer that are indistinguishable from photographs of real objects. Computer Graphics became a powerful tool for the rapid and economical production of pictures. There is virtually no area in which Graphical displays cannot be used to some advantage, so it is not surprising to find the use of CG so widespread.

Although early application in engineering & science had to rely on expensive & cumbersome equipment, advances in computer technology have made interactive computer graphics a practical tool. Today, find Computer Graphics in a diverse area such as science, engineering, medicine, business, industry, government, art, entertainment, education and training. So computer graphics can be considered as a generalized tool for drawing and creating pictures which simulates the real world situations within a small computer window.

**1.1.1 OpenGL**

OpenGL (Open Graphics Library) is a graphics software system and a platform-independent graphics API, which has become a widely accepted standard for developing graphics applications. It is a standard specification defining a cross-language, cross-platform API for interfacing with programmable GPUs for the purpose of writing applications that produce 2D and 3D real-time computer graphics.

The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. It enables a generic approach to creating and handling windows and other functions in different platforms. It is widely used for games, animation, CAD/CAM, medical imaging, and other applications that need a framework for visualizing and manipulating complex, three-dimensional shapes. OpenGL programs are highly portable and produce consistent results on any supported platform.

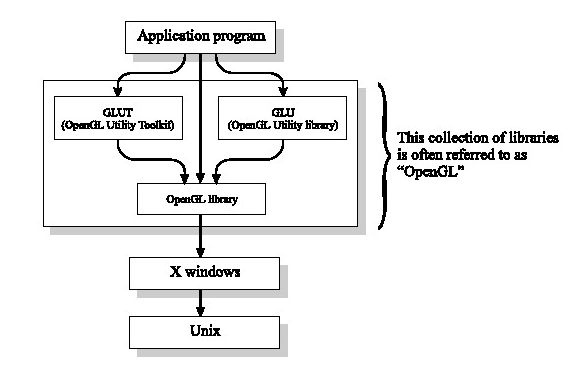
****

**Figure 1.1** Graphics pipeline in OpenGL

The Graphics pipeline is the sequence of steps that OpenGL takes when rendering objects as shown in the Figure 1.1.

OpenGL serves two main purposes:

* To hide the complexities of interfacing with different 3D accelerators, by presenting the programmer with a single, uniform API.
* To hide the differing capabilities of hardware platforms, by requiring that all implementations support the full OpenGL feature set (using software emulation if necessary).



**Figure 1.2** OpenGL three libraries

“OpenGL” is a set of three libraries: OpenGL itself, and the supporting libraries GLU and GLUT.

GLUTprovides 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”.

Figure 1.2 shows the relationships between OpenGL, GLU, and GLUT. As you can see, it’s helpful to think of “layers” of software, where each layer calls upon the facilities of software in a lower layer. However, somewhat confusingly, when most people say “OpenGL”, what they really mean is “OpenGLplus GLUplus GLUT”. It’s a slightly lazy terminology, but we’ll use it too.

To interface with the window system and to get input from external devices into our programs, we need at least one more library. For each major window system there is a system-specific library that provides the “glue” between the window system and OpenGL. For the X window system, this library is called GLX, for windows, it is WGL, and for the Macintosh, it is a gl. Rather than using a different library for each system, we use a readily available library called the OpenGL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system.

**1.1.2 Computer Graphics Technology**

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.

**1.2 OpenGL APIs**

A graphics system performs multiple tasks to produce output and handle user input. An API for interfacing with this system can contain hundreds of individual functions. It will be helpful to divide these functions into seven major groups:

1. Primitive functions
2. Attribute functions
3. Viewing functions
4. Transformation functions
5. Input functions
6. Control functions
7. Query functions

**1.3 Header Files and Libraries**

The OpenGL glut is an introduction to the OpenGL Utility Toolkit.

The OpenGL Utility Toolkit (GLUT) is a programming interface with ANSI C and FORTRAN bindings for writing window system independent OpenGL programs. It is a library of utilities for OpenGL programs.

**1.4 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

**Chapter 2**

**SYSTEM ANALYSIS**

**2.1 Scope of the Project**

The project can be implemented easily in websites that are required to give information about lights, spotlights, or activities going on in a place; in a manner to give a creative homepage to the website where the spotlight sways around in the background. It can also be used in games to give it a creative and interactive game of spotlights.

**2.2 Aim of the Project**

The aim of this project is to create an oscillation of a number of spotlights on a plane object. The spotlights shall start from a single point source of light.

**2.3 Overview of the Project**

When we look at a point on an object, the colour that we see is determined by multiple interactions among light sources and reflective surfaces. These interactions can be viewed as a recursive process. The Light-emitting (or self-luminous) surfaces from which the rays of light are emitted are called light sources.

Spotlights are characterized by a narrow range of angles through which light is emitted. It is defines as a lamp projecting a narrow, intense beam of light directly on to an object.

A simple spotlight as shown in Figure 2.1 can be constructed from a point source by limiting the angles at which light from the source can be seen. More realistic spotlights are characterized by the distribution of light within the cone, usually with most of the light concentrated in the centre of the cone.



**Figure 2.1** Spotlight

**2.3.1** **Benefits of the Project**

* Simplicity: Our 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.
* Portable: The code can be run on any machine.

**2.3.2 Constraints of the Project**

* Project gets executed only when glut is installed.
* Output will differ if there is fault in graphic system.
* May produce flickering images if colored buffer is not cleared properly.
* We need to initialize glut library in order to interact with window system.

**2.3.3 Applications of the Project**

* Can be used as demonstration package for the function like translation and rotation.
  + - * Same idea of this project can also be implemented to animated movies.

**Chapter 3**

**SYSTEM REQUIREMENTS AND SPECIFICATIONS**

The system requirement is the total configuration of the system that system should satisfy to run the designed project as desired. This condition tells about how portable our project is.

# 3.1 Functional Requirements

# The project shall display the swinging spotlights on a plane surface.

# The displayed spotlights must be of the basic Red, Green, Blue lights.

# There shall be provisions to add more spotlights of different colors.

# The facilities of pausing of the spotlight swing and resuming of the swinging of the spotlights shall be provided through menus.

# 

# 3.2 Non-Functional Requirements

# User-friendly graphical environment should be provided.

# The output at each stage should be clear, precise and accurate.

# Error handling shall be taken place appropriately.

# The user interface is to be provided to control the display of the model.

# The graphical display must ensure that it provides a perspective of 3D viewing.

# 3.3 System requirements

# The system requirements for this project are as mentioned below.

**3.3.1 Hardware Requirements**

* Architecture :x86, ARM, PowerPC, SPARC
* Processor : Intel 386 processor onwards
* Main memory : 512 MB RAM
* Hard disk : 100 MB of free hard disk space
* Monitor : VGA compatible(CRT or LCD-TFT)
* Keyboard : Standard 101 keys Keyboard
* Mouse

**3.3.2 Software Requirements**

* Operating system : Unix/Linux Ubuntu 14.04 LTS
* Language tool : OpenGL
* Graphics library : GL/glut.h
* Editor used : vi editor or gedit
* Compiler : GNU GCC compiler

**Chapter 4**

**SYSTEM DESIGN**

**4.1 Design of the system**

Thedevelopment of this project would improve the user’s knowledge about computer graphics and OpenGL. This project provides good understanding of OpenGL prominent functions like Translation and Rotation.

Keeping the factors of usability in mind we have developed this project to provide ease of use. This project will allow user to interact with it through the use of devices like mouse and keyboard.

This project will demonstrate the swinging of a Spotlight in OpenGL.

The flow of execution of the OpenGL program is as follows:

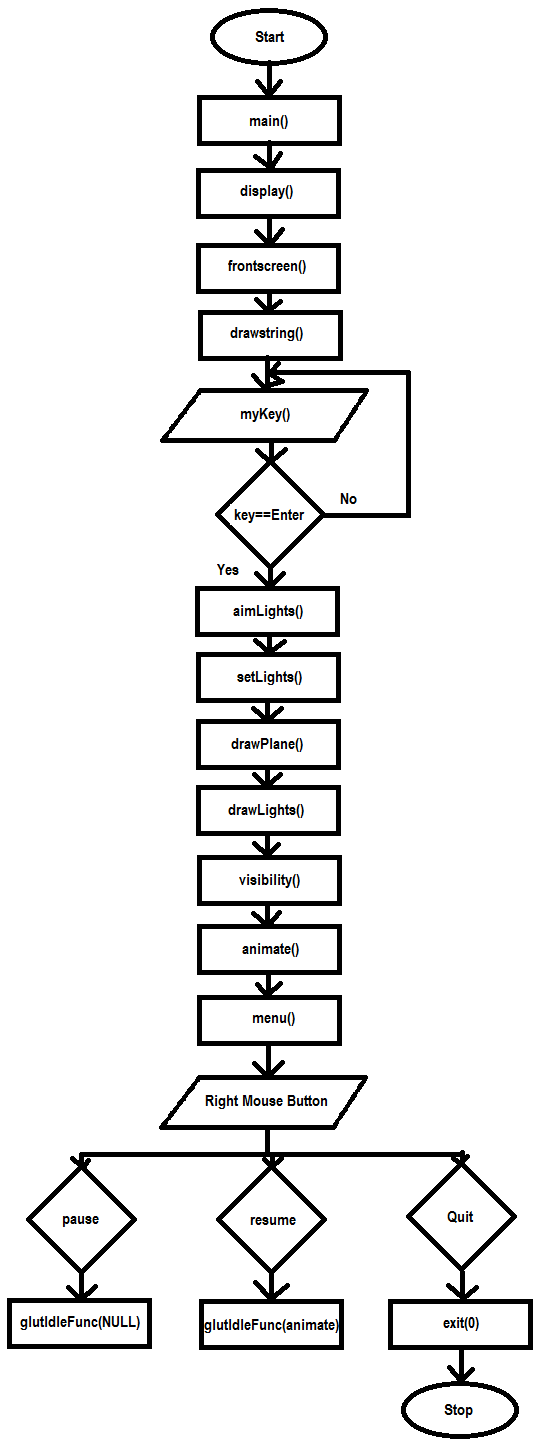
1. Execution of the program starts from main().
2. The display callback displays the spotlights on the screen
   * 1. Initially the front screen containing the introduction to the project is displayed, which in turn calls a user defined function drawstring() which is used to display array of characters on the output screen.
     2. Then the function aimLights() is called which sets the rotation angle and the arc increments of the spotlights.
     3. The function setLights() is called which sets the position of the spotlight and the direction of rotation of the spotlights.
     4. Then the drawPlane() function is called which draws the plane object.
     5. The drawLights() function draws a light ray from the point source of light to the point on the plane where the spotlight is intended to fall on.

3. The keyboard function callback is then called so that the user can interact with the program and start the oscillation of the spotlight.

4. The visibility callback and the animate function together render the image thus formed on the screen so that the output is visible to the user.

5. Then is the menu function which is required to make the program more interactive by adding menus that provide the user with options such as:

* + 1. Pausing of the Spotlight Swing.
    2. Resuming of the swinging of the Spotlight.
    3. Exit out of the program.

**4.2 Flowchart**

**Figure 4.1** Flow of control

**4.3 Pseudo code**

The execution of the program starts from the main function. It calls various inbuilt and user defined functions, the pseudo code is as follows:

static void initLights(void)

{

Function where various light material-properties are initialized to the values given.

}

static void aimLights(void)

{

Function that causes the rotation angle and arc values are increased each time it is called.

}

static void setLights(void)

{

Function to translate and rotate the position and direction of the spotlight.

}

static void drawLights(void)

{

Function to draw rays of light from the point source to the plane.

}

static void drawPlane(int w, int h)

{

Function to draw an opaque plane.

}

void display(void)

{

Function that calls other functions in a particular sequence that causes the proper display of the spotlights.

}

void animate(void)

{

Function that causes the actual movement of the spotlights on the screen.

}

void visibility(int state)

{

Function is used when the visibility of the window changes

}

void myKey( unsigned char key, int x, int y )

{

Function that provides keyboard interface.

}

void menu(int id)

{

functions that is used to implement menus for a graphical user interface.

}

int main(int argc, char \*\*argv)

{

The program execution begins from the first statement of this function. It also calls the various functioned defined in the program.

}

**4.4 Functions**

The headers, the user-defined functions and the built-in functions used in this project are as follows:

**4.4.1 Headers**

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

* **#include<stdio.h>:** It is used for performing input and output.
* **#include<stdlib.h>:** It is used for performing general functions
* **#include<string.h>:** It is used for manipulating arrays of characters.
* **#include<math.h>:** It is used for performing mathematical operations.
* **#include<GL/glut.h>:** It is used to include glut library files.

**4.4.2 User defined functions**

* **static void initLights(void)**

The various light material-properties are initialized. The various light material properties that are initialized here are: GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, GL\_QUADRATIC\_ATTENUATION.

* **static void aimLights(void)**

The rotation angle and arc values are increased each time it is called.

The rotation angle is increased using the arc value. It is done using the formula:

rot = swing \* sin(arc)

The arc value is increased using the arcIncr value. It is done using the formula:

arc = arc + arcIncr

* **static void setLights(void)**

Translation and rotation of the position and direction of each spotlight is done.

Translation and rotation is done using the built-in functions: glTranslatef, glRotatef

* **static void drawLights(void)**

Drawing of rays of light from the point source to the plane where the spotlight falls is done in this function.

This is done by using the built-in function: glVertex3f

* **static void drawPlane(int w, int h)**

This function draws an opaque plane on the screen. We used GL\_TRIANGLE\_STRIP to create the plane.

* **void display(void)**

This function calls other functions in a particular sequence that causes the proper display of the spotlights.

This function calls aimLights(), setLights(), drawPlane(w,h), drawLights() in the same order and causes the spotlights to be properly displayed.

* **void animate(void)**

This function causes the actual movement of the spotlights on the screen. This function modifies the spin value used as rotation angle in display() function.

* **void visibility(int state)**

This function is called when the visibility of the window changes. This function calls the animate function as the idle function.

* **void myKey( unsigned char key, int x, int y )**

This is the function that provides keyboard interface. The key used for interaction using keyboard is the Enter Key.

* **void menu(int id)**

This function is used to implement menus for a graphical user interface, where we have used menus like pause, resume and quit.

* **int main(int argc, char \*\*argv)**

The program execution begins from the first statement of this function. It also calls the various functioned defined in the program.

The various functions called in this function are: display(), keyboard(), visibility(), init(), menu() and this loops.

**4.4.3 Built in functions**

OpenGL functions used in the code are as follows:

* **glutInit() :** This function call initializes glut library.

The arguments from main are passed in and can be used by an application.

* **glutInitDisplayMode() :** This function call request a display with the properties that are specified in m mode. The value is determined by the logical OR operation of options including the color model.
* **glutInitWindowSize() :** This function call specifies the initial height and width of the window in pixels.
* **glutInitWindowPosition() :** This function call specifies the initial position of top left corner of the window in pixels.
* **glutCreateWindow() :** This function call creates a window on the display, the string title can be used to label the window.
* **glutDisplayFunc() :** This function call registers the display function \*func i.e executed when the window needs to be redrawn.
* **glClear() :**Clears buffer to preset values. Specifies BITWISE OR of masks that indicate the buffers to be cleared.
* **glClearColor() :** This function call sets the present RGBA clear color used when clearing the color buffer.
* **glFlush() :** This function call forces any buffered OpenGL command to execute.
* **glutPostRedisplay() :** This function call requests the display call back after the current call back returns.
* **glVertex() :** Function commands are used within the glBegin/glEnd to specify points, line and polygon vertices.
* **glColor3f() :** Set the color.
* **glutSwapBuffers() :** Swaps the buffers of the current window if double buffered.

## glutVisibilityFunc() : Sets the visibility callback for the current window.

## glutPostRedisplay() : Marks the current window as needing to be redisplayed.

* **glLightfv() :** Set the light source parameters.
* **glLightModel() :** Set the lighting model parameters.
* **glMaterial() :** Specify material parameters for the lighting model.
* **glMatrixMode() :** Specify which matrix is the current matrix
* **glFrustum() :** Multiply the current matrix by a perspective matrix
* **glLoadIdentity() :** Replaces the current matrix with identity matrix.
* **glPushMatrix() and glPopMatrix() :** Push and pop the current matrix stack.
* **glTranslate() :** It produces a translation by (x,y,z). The current matrix is multiplied by this translation matrix, with the product replacing the current matrix.
* **glRotatef() :** Multiply the current matrix by a rotation matrix.
* **glScale()** **:** Multiply the current matrix by a general scaling matrix
* **glEnable() :** Enable or disable the server side GL capabilities.
* **glutKeyboardFunc() :** It is a user interactive function which displays snapshots on hitting the appropriate keyboard keys onto the display screen.
* **glutMainLoop():** This function call causes the program to enter an event processing loop. It should be the last statement in main.

**Chapter 5**

**TESTING**

**5.1 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:** Validation involves checking that the program has implanted meets the requirement of the users.
* **Verification:** Verification involves checking that the program confirms to its specification.

**5.2 Stages in the implementation of testing**

**Unit testing**

Each individual unit is tested for corrections. These individual components will be tested to ensure that they operate correctly.

**Integration testing**

This phase involves testing collection of modules, which have been integrated into sub-systems may be independently designed and implemented. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

**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.

**User acceptance testing**

This is the final stage in the testing process before the system is tested for operational use. Any requirement problem or requirement definition problem revealed from acceptance testing are considered and made error free.

**Test plan**

Careful planning is needed to the most of testing and controlled testing cost.

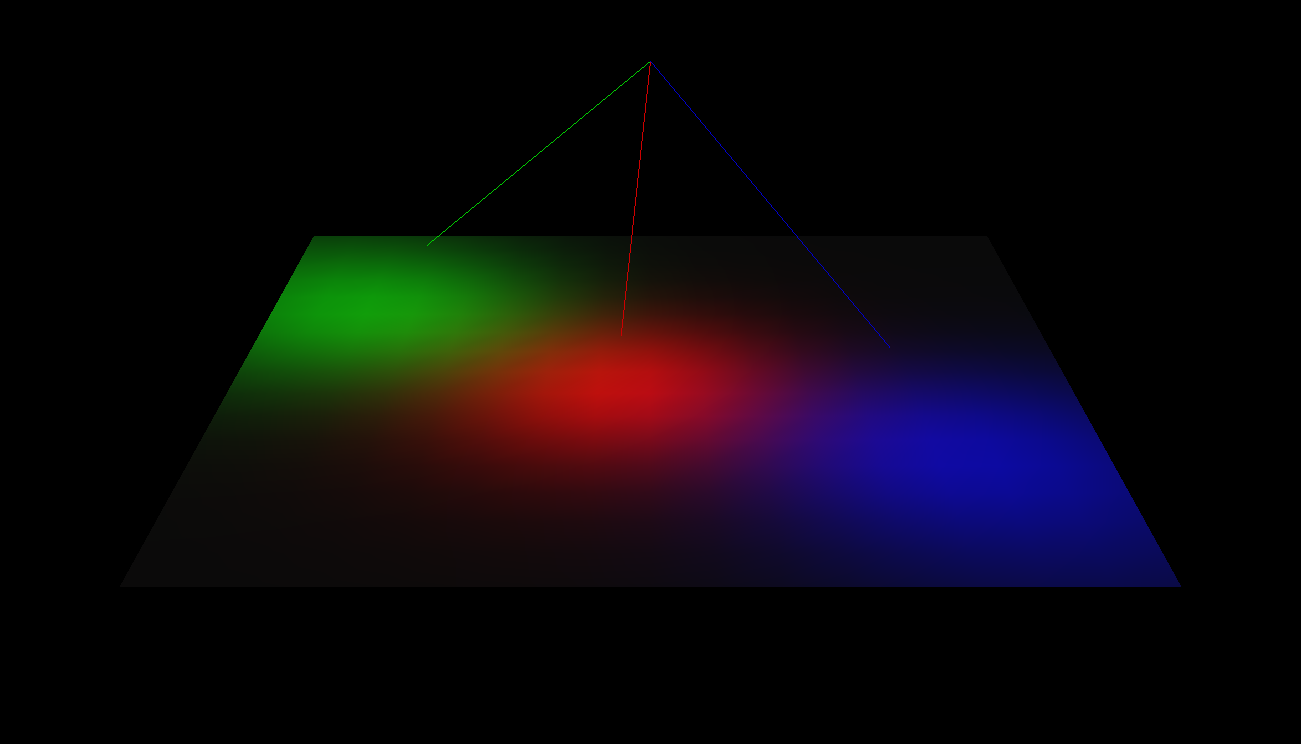
**5.3 Test cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TC ID** | **Test Case Description** | **Input** | **Actual Output** | **Expected Output** | **Remarks** |
| 1. | Spotlight swing movement | Press Enter Key | Spotlight swinging starts.  Refer Figure 6.1 | Spotlight swinging has to start | Pass |
| 2. | Display of Menu Pop -up | Click Right Mouse button | Menu pops with 3 menu entries.  Refer Figure 6.2 | Menu with 3 menu entries should pop | Pass |
| 3. | Selection of Pause Menu entry | Click Pause button | Swinging of spotlight halts.  Refer Figure 6.3 | Swinging of the spotlight should halt | Pass |
| 4. | Selection of Resume Menu entry | Click Resume button | Swinging of spotlight starts.  Refer Figure 6.4 | The spotlight should start swinging | Pass |
| 5. | Selection of Quit Menu entry | Click Quit button | The program Exits after closing the Window  Refer Figure 6.5 | The program should Exit after Closing the Window | Pass |

**Table 5.1** Test Case Table

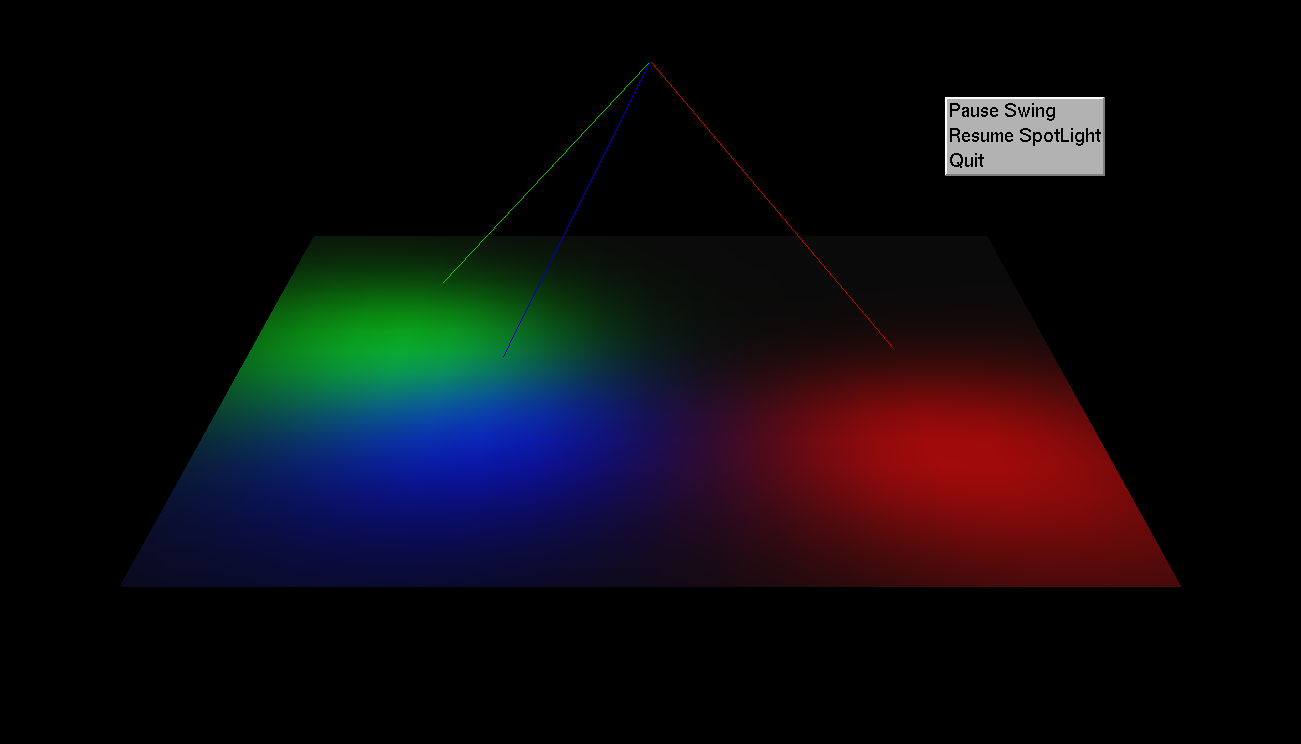
**Chapter 6**

**RESULT AND SNAPSHOTS**



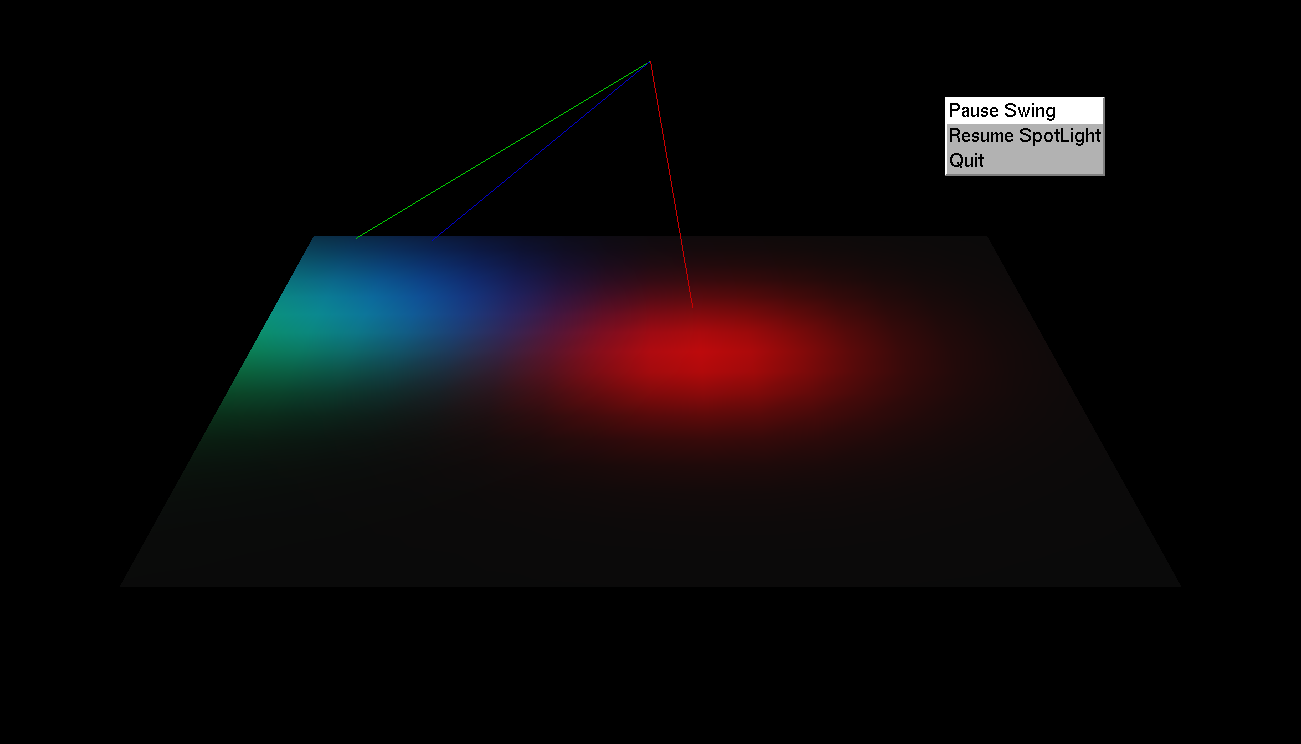
**Figure 6.1** Spotlight swing movement

The above figure illustrates the swinging of the three spotlights in a random fashion on the plane.



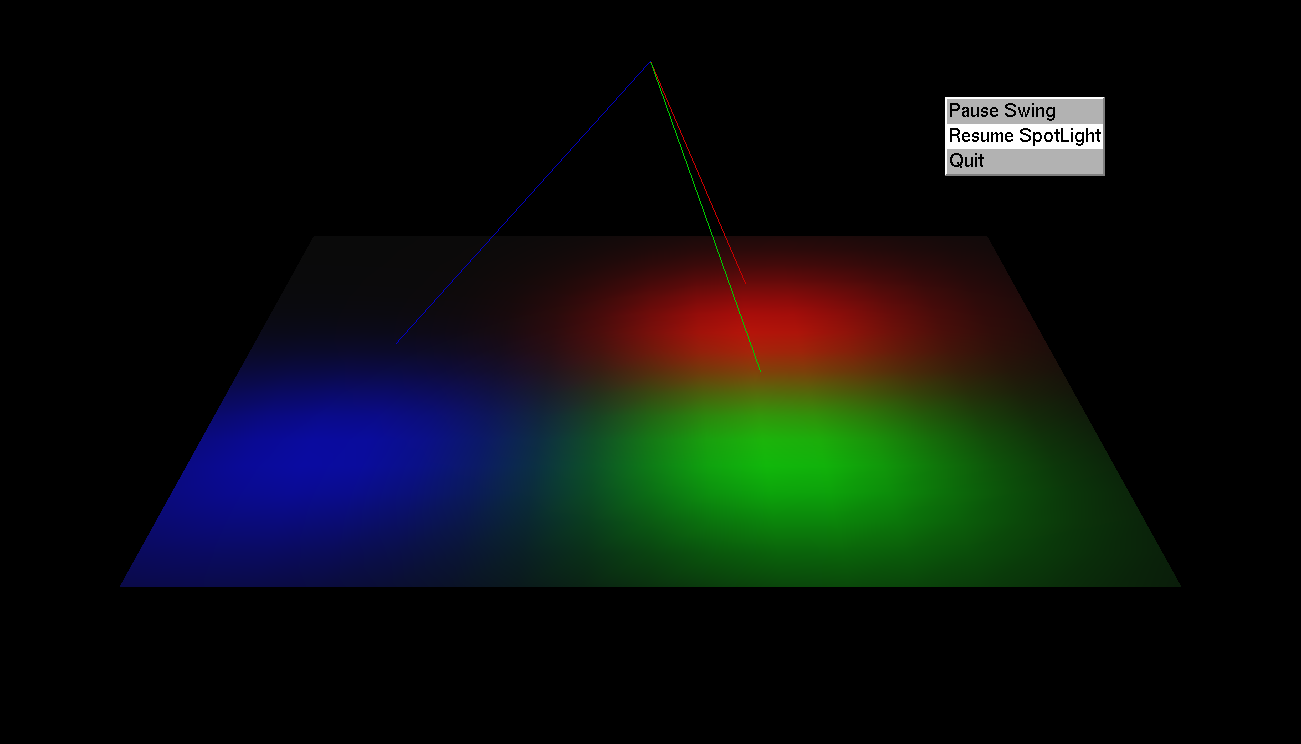
**Figure 6.2** Display of Menu Pop-up

The above figure illustrates the display of the three menus after clicking the Right Mouse Button by the user.



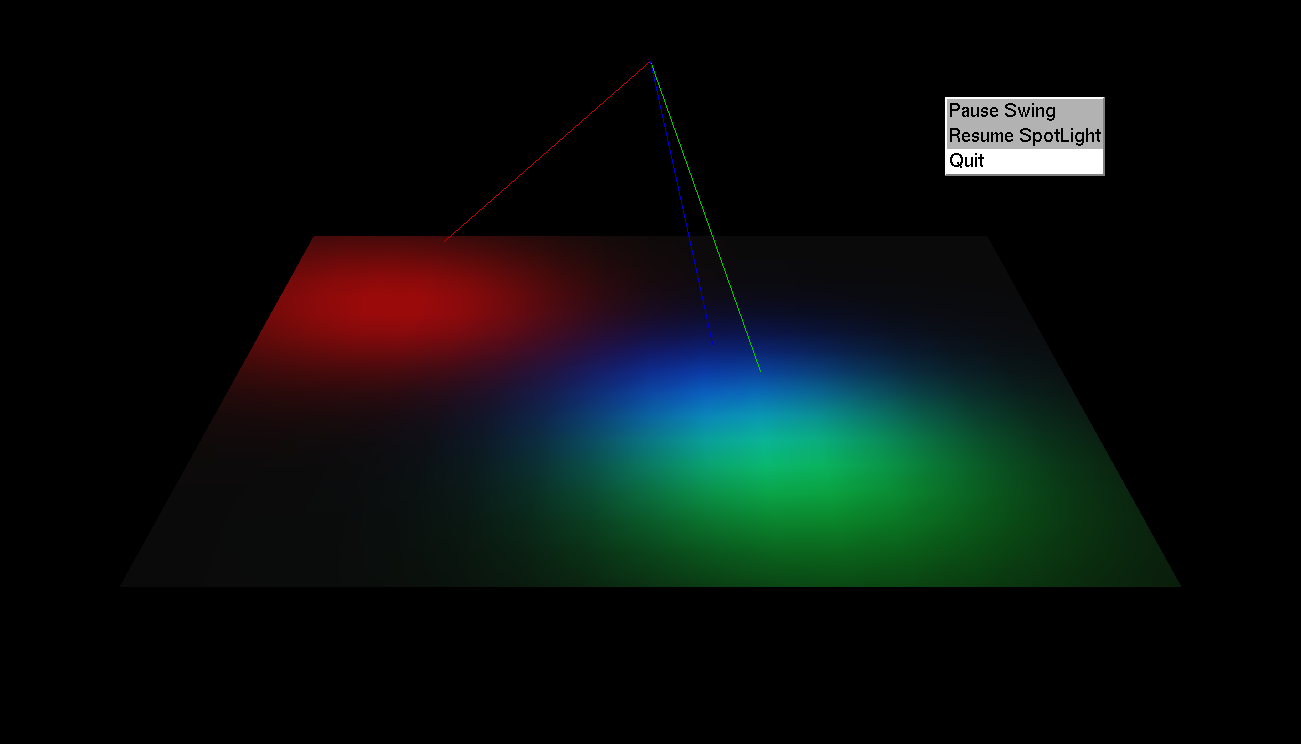
**Figure 6.3** Selection of Pause Menu entry

The above figure illustrates the usage of the menu “Pause Swing” to pause spotlight swing by selecting the appropriate menu button by the user.



**Figure 6.4** Selection of Resume Menu entry

The above figure illustrates the usage of the menu “Resume Spotlight” to resume the swinging of the spotlight by selecting the Resume menu button by the user.



**Figure 6.5** Selection of quit menu entry.

The above figure illustrates the usage of the menu “Quit” to exit the project Spotlight Swing and closing the window.

**CONCLUSION**

The Graphics Package “Spotlight swing” developed using the OpenGL graphics software system is demonstrated in this project. It is implemented using the built-in functions that are provided in standard graphics package and the user-defined functions.

We exhibited the oscillation of the spotlights of various colors on a plane object. The project consists of the spotlight which swings on the plane with the rotation angle set initially. The light from the point source illuminates the dark plane. Initially spotlights of three colors are implemented. The spotlights move in a random fashion on the plane. We have successfully implemented the usage of menu buttons that is used to pause, resume and exit the display window.

This project is an effort to the development of graphical software package which is a building block of the current high level graphical applications

**FUTURE ENHANCEMENT**

We have implemented graphical functions like movement of the spotlights on a plane surface, in a to and fro motion. We have enhanced it by making a provision to increase the number of spotlights.

In the future, this project can be further enhanced by adding some more features like increasing the number of spotlights, modifying the angle of swing and rotation of the plane surface along with the spotlight, and varying the speed of rotation of both the spotlight and the plane surface.

**BIBLIOGRAPHY**

**TEXT BOOKS:**

1. Edward Angel, 2009, Interactive computer graphics: A top-down Approach With OpenGL, 5th edition, Addison-Wesley

# OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1 (7th Edition) - [Dave Shreiner](http://www.amazon.com/Dave-Shreiner/e/B001IGNQ0G/ref=dp_byline_cont_book_1)

# OpenGL Super bible: Comprehensive Tutorial and by [Graham Sellers](http://www.amazon.in/s/ref=dp_byline_sr_book_1?ie=UTF8&field-author=Graham+Sellers&search-alias=stripbooks), [Richard S Wright Jr.](http://www.amazon.in/s/ref=dp_byline_sr_book_2?ie=UTF8&field-author=Richard+S+Wright+Jr.&search-alias=stripbooks), [Nicholas Haemel](http://www.amazon.in/s/ref=dp_byline_sr_book_3?ie=UTF8&field-author=Nicholas+Haemel&search-alias=stripbooks)

**WEBSITES:**

1. <http://www.opengl.org/>
2. <http://www.gamedev.net/page/resources/_/technical/opengl/lighting-and-normals-r1682>
3. <http://learnopengl.com/>
4. <http://falloutsoftware.com/tutorials/gl/gl8.htm>