VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belagavi-590018, Karnataka

A Mini Project Report on

“FESTIVALS OF INDIA”

Submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Engineering

In

Computer Science and Engineering

Submitted by

RAVI CHIKKARADDI(4NN22CS404)

Under the Guidance of

Mr. Deepak P.

Assistant Professor

Dept. of CS & Engineering

ESTD-2008

Department of Computer Science and Engineering

NIE Institute of Technology

Mysuru -570018

2023-24

Department of Computer Science and Engineering

NIE Institute of Technology

ESTD-2008

CERTIFICATE

This is to certify that the mini project entitled ***"FESTIVALS OF INDIA"*** is carried out by ***RAVI CHIKKARADDI*** bearing **USN:*4NN22CS404*** respectively in the partial fulfillment for the Fifth semester of **Bachelor of Engineering degree** in **Computer Science and Engineering** of the **Visvesvaraya Technological University, Belagavi** during the academic year **2023-24**. The project report has been approved as it satisfies the academic requirements with respect to project work prescribed for the Bachelor of Engineering.

Signature of the Guide	**Signature of the HOD**
Mr. Deepak P.	**Dr. Usha M S.**
Assistant Professor,	Associate Professor & Head,
Dept. of CS & Engineering	Dept. of CS & Engineering
NIEIT, Mysuru	NIEIT, Mysuru

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 consistent guidance and encouragement crowned our efforts with success.

We consider ourselves proud to be a part of **NIE Institute of Technology Mysuru** family, the institution which stood by our side in all our endeavors.

We wish to express our gratitude to **Dr. Rohini Nagapadma,** Principal, **NIE Institute of Technology Mysuru,** for providing a congenial working environment.

We express our sincere thanks to **Dr. Usha M S**, Associate Professor and Head, Department of Computer Science and Engineering, for her support and encouragement.

We would like to thank our guide**, Mr. Deepak P.** Assistant Professor, Department of Computer Science and Engineering, for her inspiration, guidance, constant supervision, direction and discussions in the successful completion of this project work.

We are thankful to the Department of Computer Science and Engineering staff members and non-staff members for their co-operation extended towards this work.

Finally, our heartfelt gratitude to our **family members, relatives** and **friends** for their constant support, motivation and encouragement throughout this project. Above all we would like to thank God Almighty, for having showered his blessings on us.

RAVI CHIKKARADDI(4NN22CS404)

ABSTRACT

Festivals of India is a 2D graphics based educating project. The project has scope to learnComputer Graphics from fundamentals. We have used OpenGL and utility tool kit to implement this project. We have used C++ language to write the required program. The project also depicts the danger that invariably has to be faced when rules are not followed, safety precautions not taken and warning signals are ignored. First slide of the project depicts the Ramzan initially. Second slide depicts the Diwali and the third gives the Christmas Fourth slide depicts the Dasara .

LIST OF CONTENTS.

Chapter-1

INTRODUCTION

1.1 Computer Graphics

Computer graphics are graphics created using computers and the representation of image data by a computer specifically with help from specialized graphic hardware and software.

The interaction and understanding of computers and interpretation of data has been made easier because of computer graphics. Computer graphic development has had a significant impact on many types of media and have revolutionized animation, movies and the video game industry.

Computer graphics is widespread today. Computer imagery is found on television, in newspapers, for example in weather reports, or for example in all kinds of medical investigation and surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. In the media "such graphs are used to illustrate papers, reports, thesis", and other presentation material.

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. Over the past decade, other specialized fields have been developed like information visualization, and scientific visualization more concerned with "the visualization of three dimensional phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic (time) component".

OpenGL Utility Toolkit (GLUT) was developed by Mark Kilgard, it Hides the complexities of differing window system APIs, Default user interface for class projects, Glut routines have prefix glut, Eg- glutCreateWindow().

1.2 OpenGL Technology

OpenGL is a graphics application programming interface (API) which was originally developed by Silicon Graphics. OpenGL is not in itself a programming language, like C++, but functions as an API which can be used as a software development tool for graphics applications. The term Open is significant in that OpenGL is operating system independent. GL refers to graphics language. OpenGL also contains a standard library referred to as the OpenGL Utilities (GLU). GLU contains routines for setting up viewing projection matrices and describing complex objects with line and polygon approximations.

OpenGL gives the programmer an interface with the graphics hardware. OpenGL is a low-level, widely supported modelling and rendering software package, available on all platforms. It can be used in a range of graphics applications, such as games, CAD design, modelling.

OpenGL is the core graphics rendering option for many 3D games, such as Quake 3. The providing of only low-level rendering routines is fully intentional because this gives the programmer a great control and flexibility in his applications. These routines can easily be used to build high-level rendering and modelling libraries. The OpenGL Utility Library (GLU) does exactly this, and is included in most OpenGL distributions! OpenGL was originally developed in 1992 by Silicon Graphics, Inc, (SGI) as a multi-purpose, platform independent graphics API. Since 1992 all of the development of OpenGL.

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

OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, you must work through whatever windowing system controls the particular hardware you're using. The OpenGL

Visualization Programming Pipeline:

OpenGL operates on image data as well as geometric primitives. Simplifies Software Development, Speeds Time-to-Market.

Routines simplify the development of graphics software—from rendering a simple geometric point, line, or filled polygon to the creation of the most complex lighted and texture mapped NURBS curved surface. OpenGL gives software developers access to geometric and image primitives, display lists, modeling transformations, lighting and texturing, anti-aliasing, blending, and many other features. Every conforming OpenGL implementation includes the full complement of OpenGL functions. The well-specified OpenGL standard has language bindings for C, C++, Fortran, Ada, and Java. All licensed OpenGL implementations come from a single specification and language binding document and are required to pass a set of conformance tests. Applications utilizing OpenGL functions are easily portable across a wide array of platforms for maximized programmer productivity and shorter time-to-market.

All elements of the OpenGL state—even the contents of the texture memory and the frame buffer—can be obtained by an OpenGL application. OpenGL also supports visualization applications with 2D images treated as types of primitives that can be manipulated just like 3D geometric objects. As shown in the OpenGL visualization programming pipeline diagram above.

1.3 Project description

In this project, we strive to obtain a slide of festivals of India. Basically, we promote to show the different festivals of India we celebrate. This project describes the festival Ramzan, Diwali and Christmas.

Ramadan is the name of the one of the twelve months of the Muslim calendar year.

Diwali is celebrated by Hindus Jains Buddhists to mark different historical events.

Christmas is celebrated to remember the birth of the Jesus Christ, who Christians believe is the son of god

We make use of C with OpenGL for entire coding purpose along with some features of Windows. The OpenGL Utility is a Programming Interface. We use light and material functions to add luster, shade and shininess to graphical objects. The toolkit supports much functionalities like multiple window rendering, callback event driven processing using sophisticated input devices etc.

1.4 OpenGL Functions Used:

This project is developed using Code Blocks and this project is implemented by making extensive use of library functions offered by graphics package of OpenGL, a summary ofthose functions follows:

1.4.1 glBegin() :

Specifies the primitives that will be created from vertices presented between glBegin and subsequent glEnd. GL_POLYGON, GL_LINE_LOOP etc.

1.4.2 glEnd(void) :

It ends the list of vertices.

1.4.3 glPushMatrix() :

void ***glPushMatrix****(void)*

glPushMatrix pushes the current matrix stack down by one level, duplicating the current matrix.

1.4.4 glPopMatrix() :

void ***glPopMatrix(****void)*

glPopMatrix pops the top matrix off the stack, destroying the contents of the popped matrix. Initially, each of the stacks contains one matrix, an identity matrix.

1.4.5 glTranslate() :

void ***glTranslate****(GLdouble x, GLdouble y, GLdouble z)*

Translation is an operation that displaces points by a fixed distance in a given direction. *Parameters x, y, z* specify the *x, y,* and *z* coordinates of a translation vector. Multiplies current matrix by a matrix that translates an object by the given x, y and z-values.

1.4.6 glClear() :

*void **glClear**(GLbitfield mask)*

glClear takes a single argument that is the bitwise *or* of several values indicating which buffer is to be cleared. GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT, GL_ACCUM_ BUFFER_BIT, and GL_STENCIL_BUFFER_BIT. Clears the specified buffers to their current clearing values.

1.4.7 glClearColor() :

*void **glClearColor**(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha)*

Sets the current clearing color for use in clearing color buffers in RGBA mode. The red, green, blue, and alpha values are clamped if necessary to the range [0,1]. The default clearing color is (0, 0, 0, 0), which is black.

1.4.8 glMatrixMode() :

*void **glMatrixMode**(GLenum mode)*

It accepts three values GL_MODELVIEW, GL_PROJECTION and GL_TEXTURE. It specifies which matrix is the current matrix. Subsequent transformation commands affect the specified matrix.

1.4.9 glutInitWindowPosition() :

*void **glutInitWindowPosition**(int x, int y);*

This API will request the windows created to have an initial position. The arguments x, y indicate the location of a corner of the window, relative to the entire display.

1.4.10 glLoadIdentity() :

*void **glLoadIdentity**(void);*

It replaces the current matrix with the identity matrix.

1.4.11 glutInitWindowSize() :

*void **glutInitWindowSize**(int width, int height);*

The API requests windows created to have an initial size. The arguments width and height indicate the window's size (in pixels). The initial window size and position are hints and may be overridden by other requests.

1.4.12 glutInitDisplayMode

*void **glutInitDisplayMode**(unsigned int mode);*

Specifies the display mode, normally the bitwise OR-ing of GLUT display mode bit *masks*. This API specifies a display mode (such as RGBA or color-index, or single or double-buffered)for windows.

1.4.13 glFlush() :

*void **glFlush**(void);*

The glFlush function forces execution of OpenGL functions in finite time.

1.4.14 glutCreateWindow() :

*int **glutCreateWindow**(char *name);*

The parameter *name* specifies any name for window and is enclosed in double quotes. This opens a window with the set characteristics like display mode, width, height, and so on. The string name will appear in the title bar of the window system. The value returned is a unique integer identifier for the window. This identifier can be used for controlling and rendering to multiple windows from the same application.

1.4.15 glutDisplayFunc() :

*void **glutDisplayFunc**(void (*func)(void))*

Specifies the new display callback function. The API specifies the function that's called whenever the contents of the window need to be redrawn. All the routines need to be redraw the scene are put in display callback function.

1.4.16 glVertex2f

*void **glVertex2f**(GLfloatx,GLfloat y);*

x	Specifies the x-coordinate of a vertex.
y	Specifies the y-coordinate of a vertex.

The glVertex function commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices. The current color, normal, and texture coordinates are associated with the vertex when glVertex is called. When only x and y are specified, z defaults to 0.0 and w defaults to 1.0. When x, y, and z are specified, w defaults to 1.0.

1.4.17 glColor3f

void glColor3f(GLfloat red, GLfloat green, GLfloat blue);

PARAMETERS:

1. Red: The new red value for the current color.
2. Green: The new green value for the current color.
3. Blue: The new blue value for the current color.

Sets the current color.

1.4.18 **glRotate**():

void glRotate(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);

PARAMETERS:

angle: The angle of rotation, in degrees.

x: The x coordinate of a vector.

y: The y coordinate of a vector.

z: The z coordinate of a vector.

The glRotated and glRotatef functions multiply the current matrix by a rotation matrix.

1.4.19 gluPerspective():

void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar);

PARAMETERS:

fovy : Specifies the field of view angle, in degrees, in the y direction.

aspect: Specifies the aspect ratio that determines the field of view in the x direction. The aspect ratio is the ratio of x (width) to y (height).

zNear: Specifies the distance from the viewer to the near clipping plane (always positive).

zFar : Specifies the distance from the viewer to the far clipping plane (always positive).

Sets up a perspective projection matrix.

1.4.20 glMaterialfv():

void glMaterialfv(GLenum face, GLenum pname, const GLfloat params);

PARAMETERS:

face : The face or faces that are being updated. Must be one of the following: GL_FRONT, GL_BACK, or GL_FRONT and GL_BACK.

Pname: The material parameter of the face or faces being updated.
The parameters that can be specified using glMaterialfv, and their interpretations by the lighting equation, are as follows.
GL_SPECULAR: The params parameter contains four integer or floating-point values that specify the seculars RGBA reflectance of the material. Integer values are mapped linearly such that the most positive represent able value maps to 1.0, and the most negative represent able value maps to -1.0. Floating-point values are mapped directly. Neither integer nor floating-point values are

clamped. The default specular reflectance for both front-facing and back-facing materials is (0.0, 0.0, 0.0, 1.0).
The glMaterialfv function specifies material parameters for the lighting model.

1.4.21 glutInit():

*glutInit(int *argcp, char **argv);*

PARAMETERS:

argcp : A pointer to the program's unmodified argc variable from main. Upon return, the value pointed to by argcp will be updated, because glutInit extracts any command line options intended for the GLUT library.

argv : The program's unmodified argv variable from main. Like argcp, the data for argv will be updated because glutInit extracts any command line options understood by the GLUT library.

glutInit(&argc,argv);

glutInit is used to initialize the GLUT library.

1.4.22 glutMainLoop ():

void *glutMainLoop(void);*
glutMainLoop();
glutMainLoop enters the GLUT event processing loop.

1.4.24 **glEnable():**

void glEnable(GLenum cap);

glEnable(GL_CULL_FACE);
PARAMETERS:

cap: A symbolic constant indicating an OpenGL capability.

The glEnable enable OpenGL capabilities.

Chapter-2

REQUIREMENTS SPECIFICATION

2.1 Hardware requirements:

- Pentium or higher processor.
- Basic Requirements
- VGA monitor.
- Hard Disk Space: 4.0 GB

2.2 Software requirements:

- The graphics package has been designed for OpenGL; hence the machine must have Dev C++.
- Software installed preferably 6.0 or later versions with mouse driver installed.
- GLUT libraries, Glut utility toolkit must be available.
- Operating System**:** Windows
- Version of Operating System**:** Windows XP, Windows NT and Higher
- The package is implemented using Microsoft visual C++, under the windows platform. OpenGL and associated toolkits are used for the package development.
- OpenGL , a software interface for graphics hardware with built in graphics libraries like glut and glut32, and header files like glut.h.

Chapter-3

INTERFACE AND ARCHITECTURE

3.1 Flow Diagram:

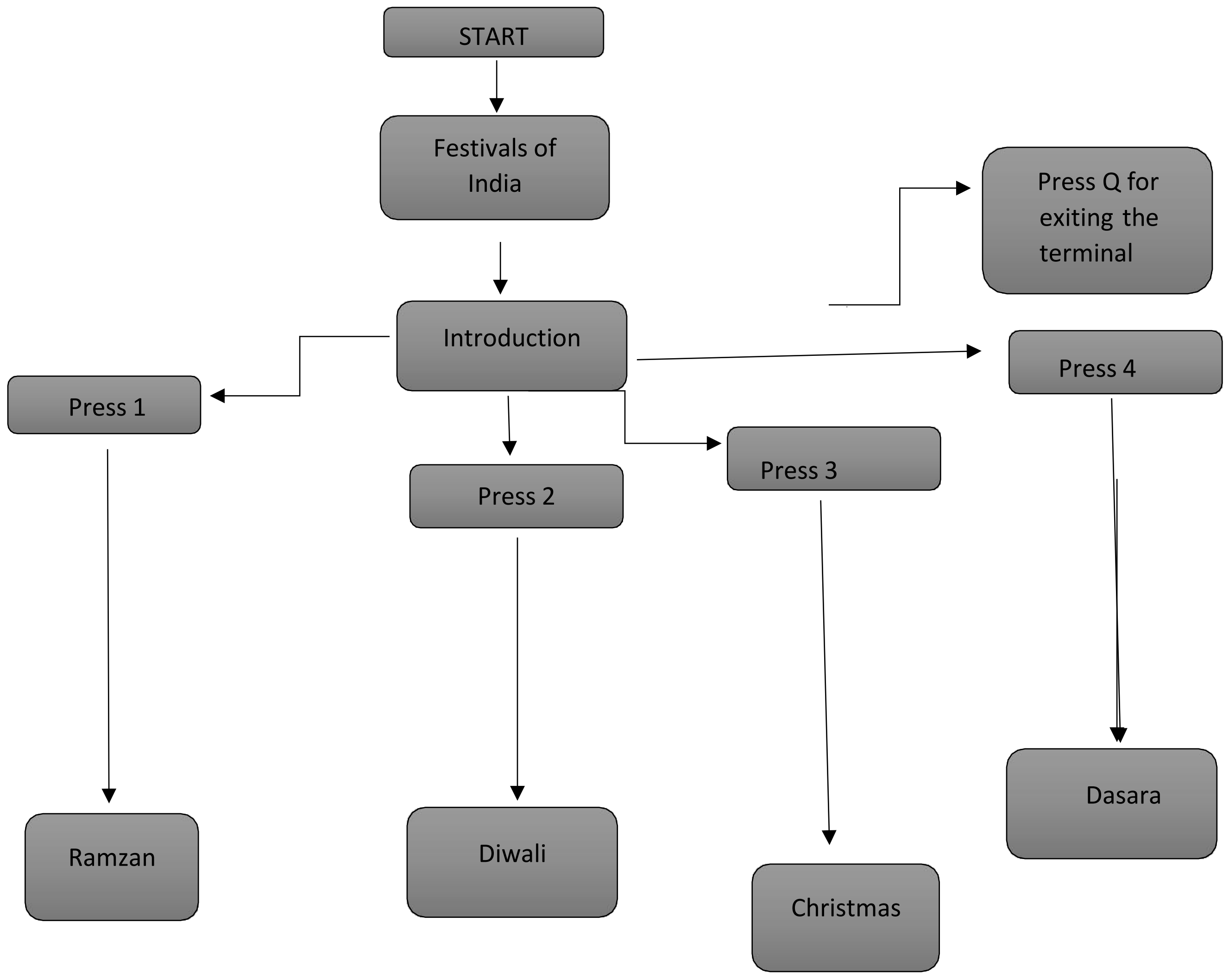


Fig 3.1 :- The flow diagram of the festivals of India

OpenGL is a software tool for developing the graphics objects. OpenGL library called GLUT i.e. Graphics Library Utility toolkit supports graphics system with the necessary modelling and rendering techniques. The Lighting system is a technique for displaying graphic objects on the monitor and displaying the light effects. It provides the following functionalities.

3.2 Initialization

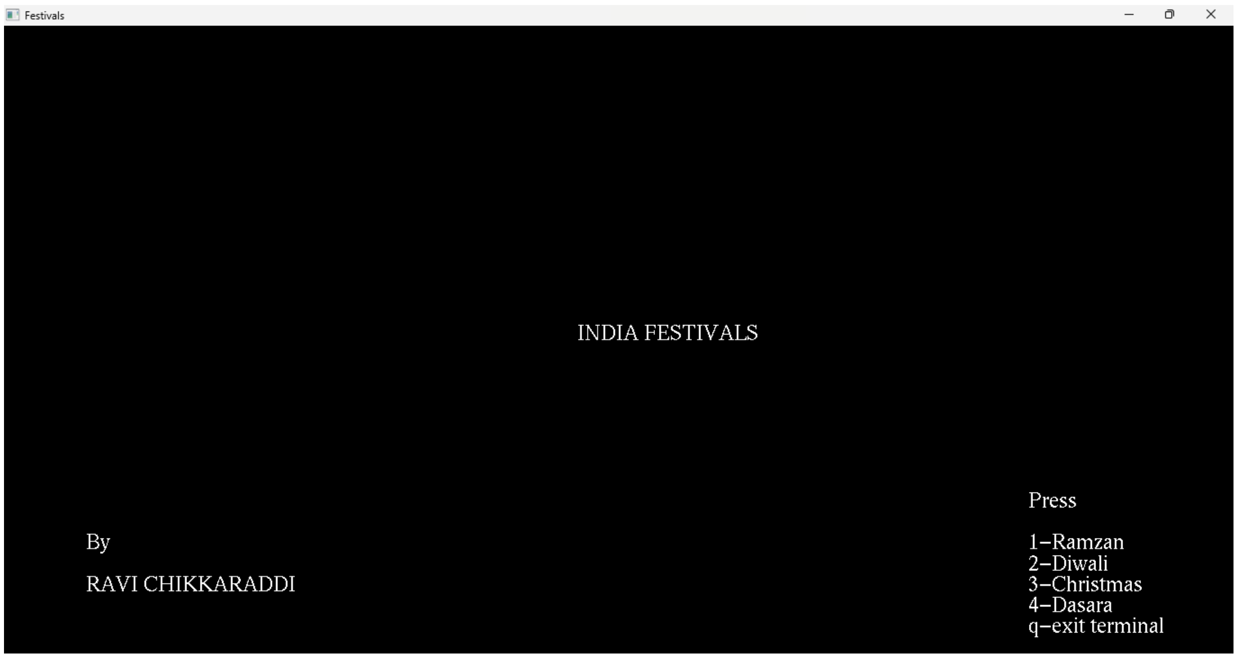
This function is the initial stage of the system where the system initializes the various aspects of the graphics system based on the user requirements, which include Command line processing, window system initialization and also the initial window creation state is controlled by these routines.

3.3 Event Processing

This routine enters GLUT's event processing loop. This routine never returns, and it continuously calls GLUT callback as and when necessary. This can be achieved with the help of the callback registration functions. These routines register callbacks to be called by the GLUT event processing loop.

Chapter-5

SNAPSHOTS



INDIA FESTIVALS

By

RAVI CHIKKARADDI

Press

1–Ramzan
2–Diwali
3–Christmas
4–Dasara
q–exit terminal

Fig 5.1 :- Introduction page

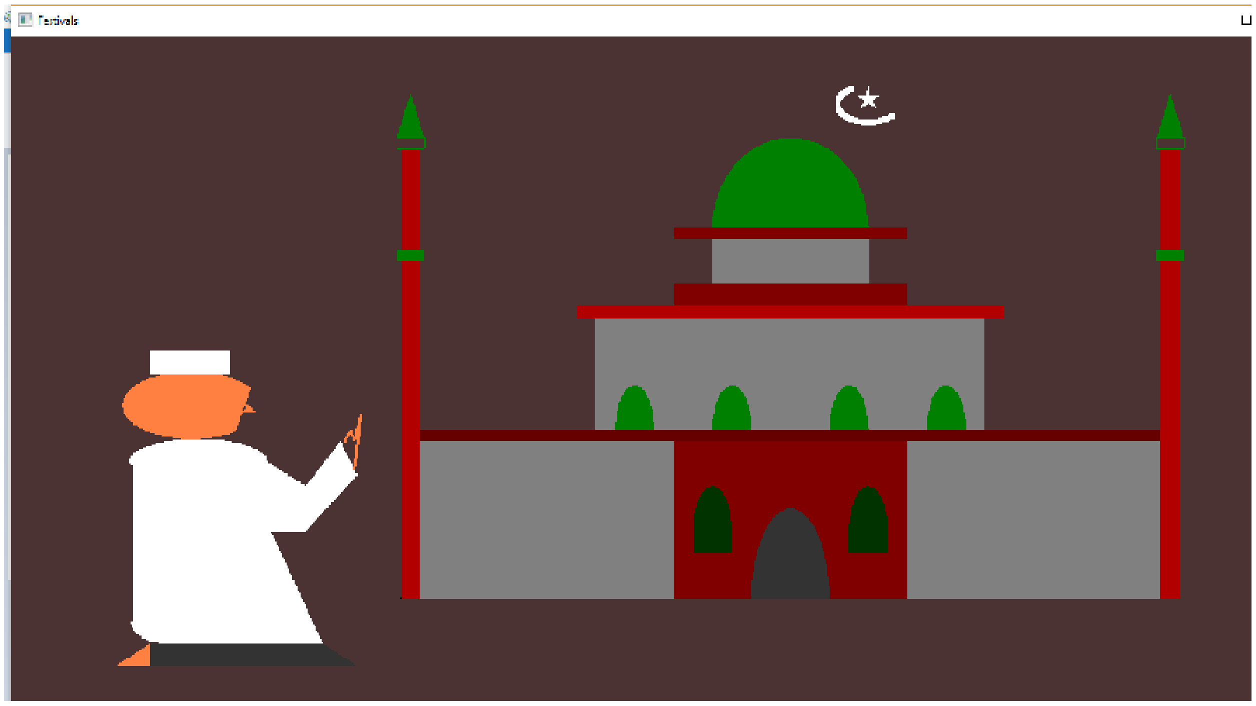


Fig 5.2:- Ramzan

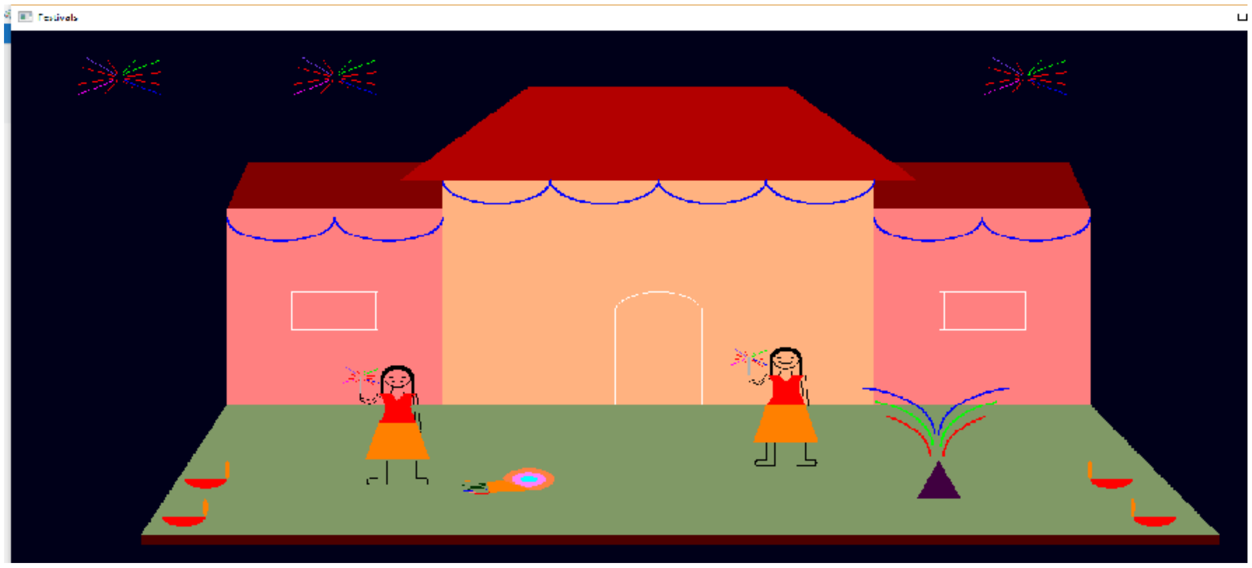


Fig 5.3:- diwali

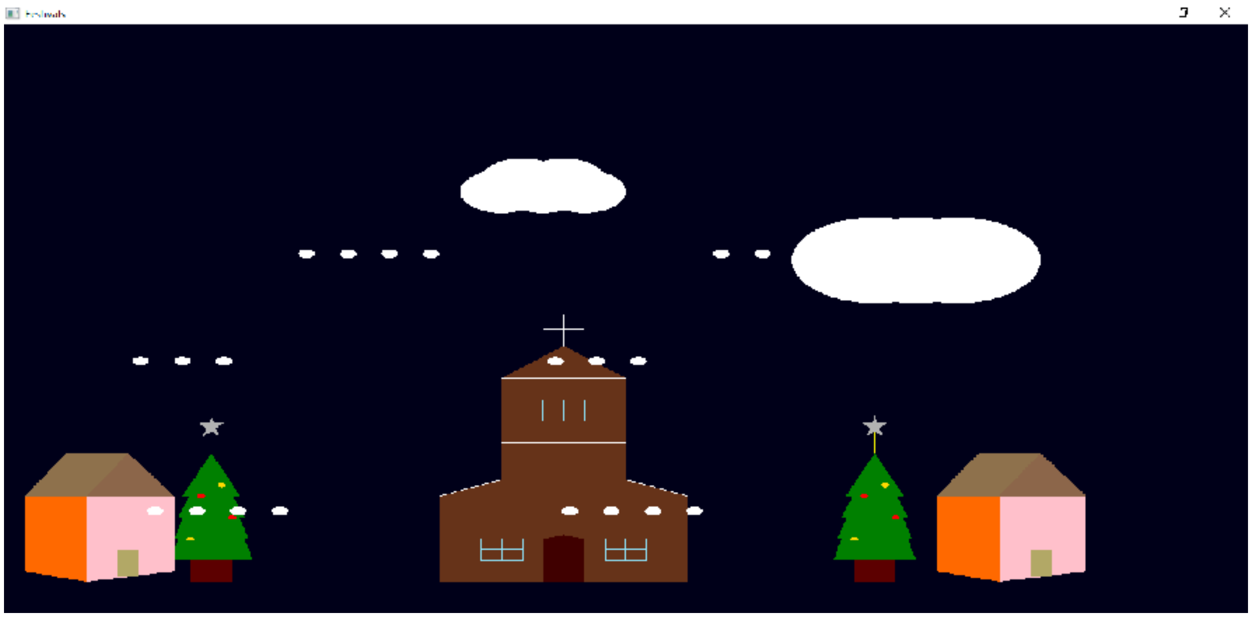


Fig 5.4 :- Christmas

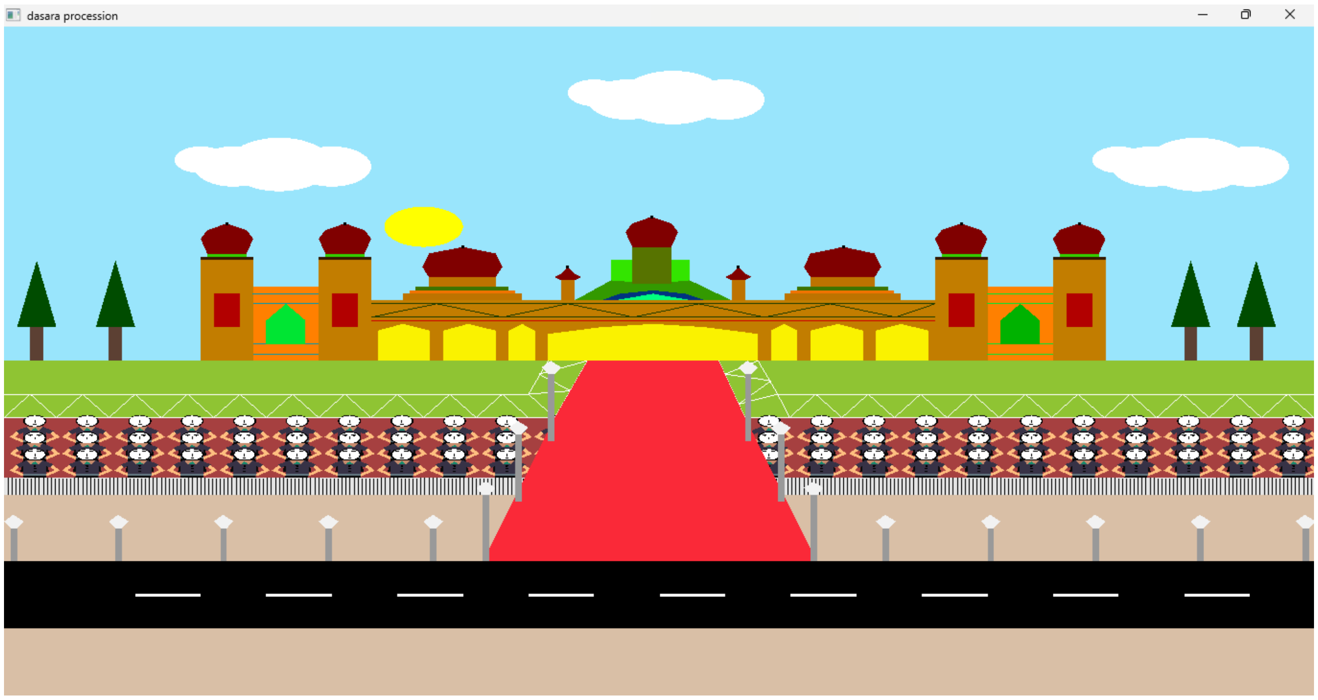


Fig 5.5 :- **Day mode of Mysore palace**

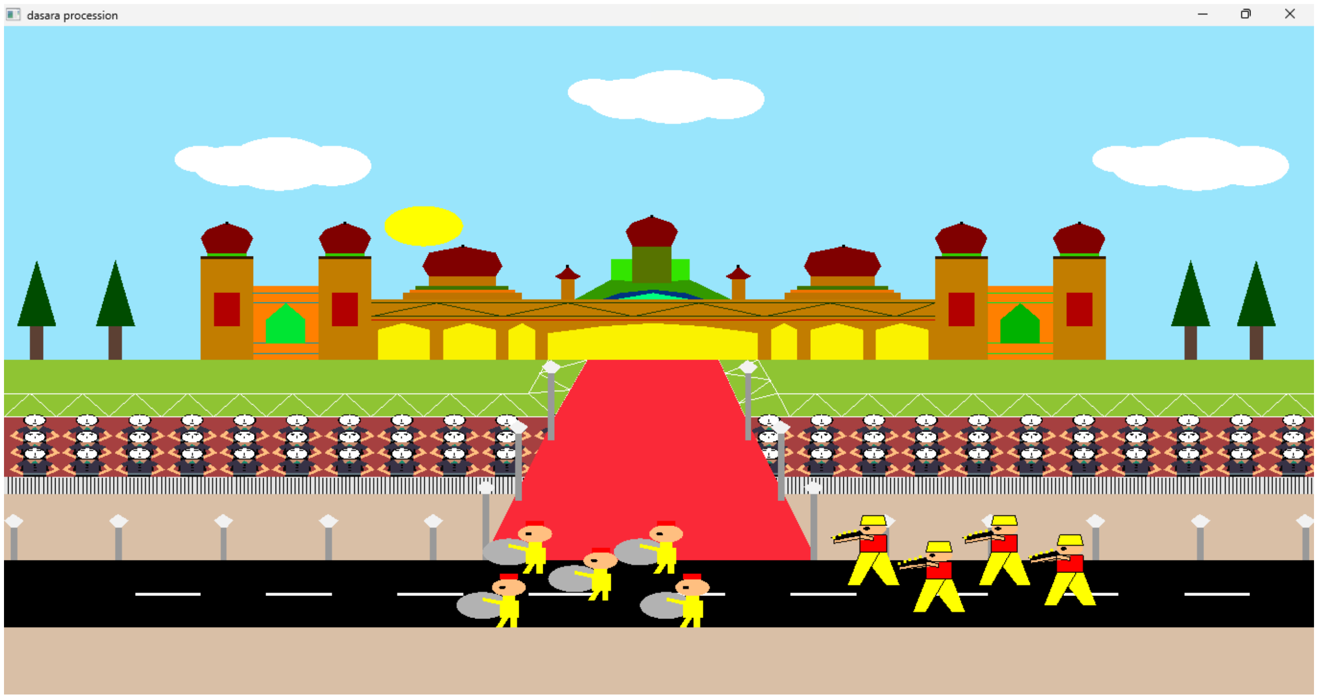


Fig 5.6: Movement of musicians

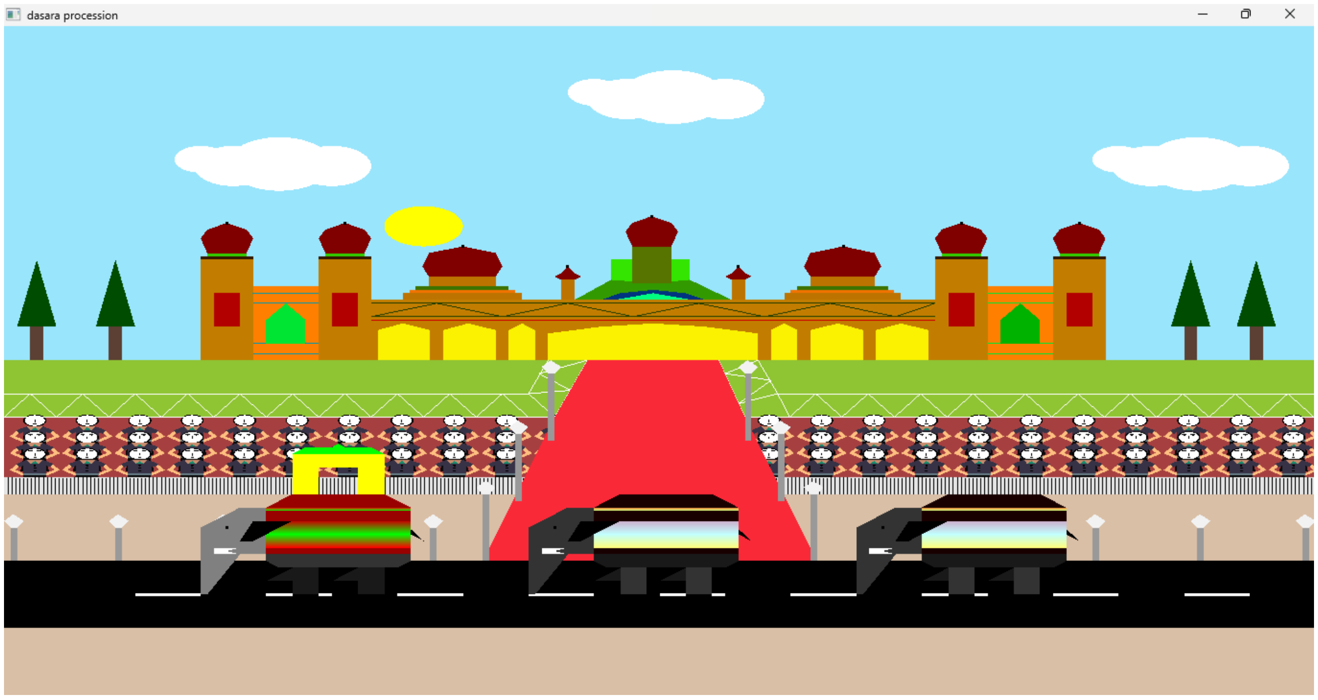


Fig 5.7: Movement of ambari along with other elephants

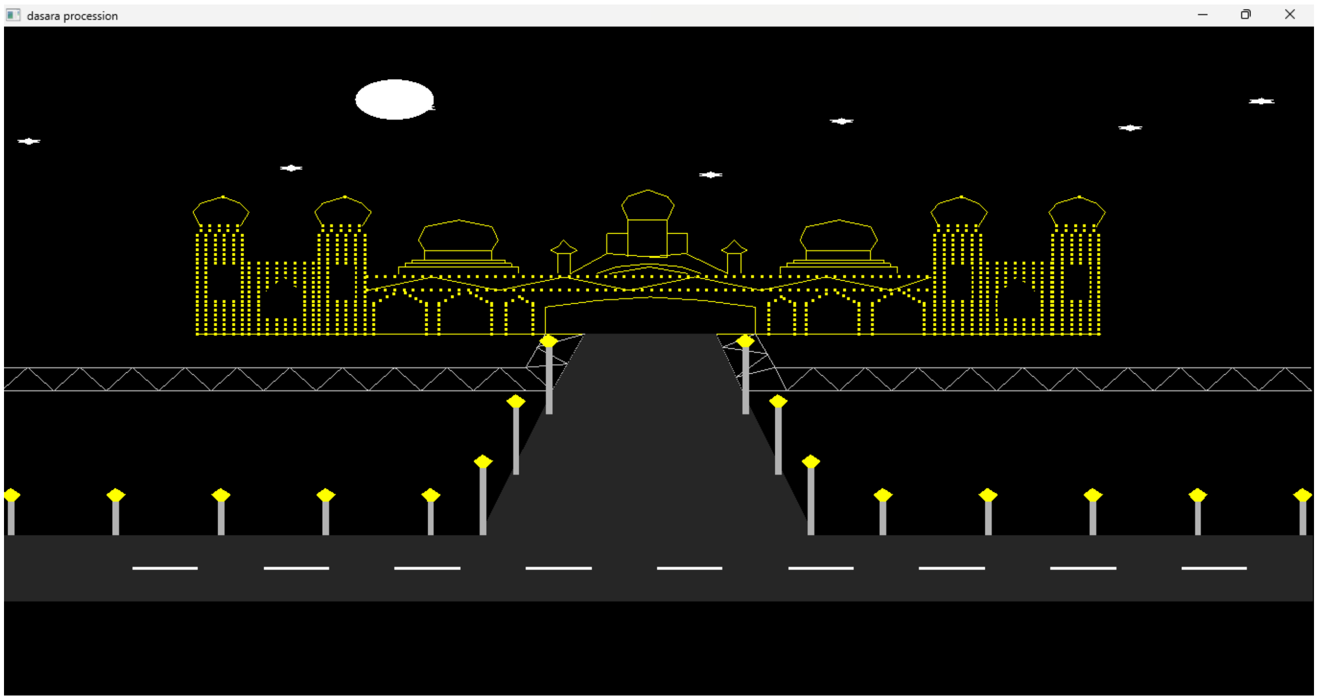


Fig 5.8: Night mode of Mysore palace

Chapter-6

CONCLUSION

Our project aims at showing the **Festivals of India** in OpenGL. Using built in functions provided by graphics library and integrating it with the C and C++.The lighting and material functions of OpenGl library add effect to the objects in animation.Showing the different Festivals of India in OpenGL. Using built in functions provided by graphicslibrary and integrating it with the C implementation, it was possible to visually representthe vertices and faces of the 3D objects. The program is user friendly as the only skill required in executing this program is the basic knowledge of Opengl.

The aim in developing this program was to design a simple program using Open GL application software by applying the skills we learnt in class, and in doing so, tounderstand the algorithms and the techniques underlying interactive graphics better.

The designed program will incorporate all the basic properties that a simple program must possess.The program is user friendly as the only skill required in executing this program is the knowledge of graphics.

Chapter-7

FUTURE ENHANCEMENT

- We can try to add more regional festivals of India
- We can try to improve the quality of the object drawn.
- Converting the entire 2D model to 3D model.
- We can decorate the objects to our prospective.
- Can add more functions like timer and mouse events

REFERENCES

Books:

[1] The Red Book –OpenGL Programming Guide,6th edition.

[2] Rost , Randi J. : OpenGL Shading Language, Addison-Wesley

[3] Interactive Computer Graphics-A Top Down Approach Using OpenGL, Edward Angel,Pearson-5th edition.

[4] www.opengl.org

[5] www.sourcecode.com

[6] www.wikipedia.org

[7] www.pearsoned.co.in

[8] www.google.com