BANGALORE UNIVERSITY

UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING

K. R. CIRCLE, BENGALURU - 560001

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A MINI PROJECT REPORT

ON

"MUSIC CONCERT"

Submitted By

AVINASH UPADHYAYA K R

(17GAEC9013) 5th SEM CSE

Under the guidance of

SAMPATH KUMAR Y R

Department of Computer Science and Engineering

DECEMBER 2019

BANGALORE UNIVERSITY

UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING

K. R. CIRCLE, BENGALURU - 560001

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the mini project entitled **"Music Concert"** is a bona fide work carried out by **AVINASH UPADHYAYA K R (17GAEC9013)** a bona fide student of **University Visvesvaraya College of Engineering** in partial fulfillment for the mini project in Computer Graphics Laboratory, 5th semester, Computer Science & Engineering, Bangalore University, Bengaluru during the year 2019-20. It is certified that all the corrections/suggestions indicated for Internal Assessment have been incorporated in the Report. The mini Project Report has been approved as it satisfies the academic requirements in respect of mini Project work prescribed for the said degree.

-----------------------	----------------------
Sampath Kumar Y R	**Dr. CHAMPA H N**
Research Scholar	Chairperson and Professor
Dept. of CSE UVCE	Dept. of CSE UVCE

Examiner 1:____________ Examiner 2:____________

ACKNOWLEDGEMENT

I take this opportunity to thank our institution **University Visvesvaraya College of Engineering** for having given us an opportunity to carry out this project.

I would like to thank **Dr. Ramesh H N, Principal, UVCE**, for providing us all the facilities to work on this project. I am indebted to him for being our pillar of strength and inspiration.

I wish to place my grateful thanks to **Dr. Champa H N, Chairperson, Department of Computer Science and Engineering, UVCE,** who helped me to make my project a great success.

I am grateful to **Sampath Kumar Y R, Department of Computer Science and Engineering**, **UVCE,** for his valuable suggestions and support, which has sustained me throughout the course of the project.

- AVINASH UPADHYAYA K R

17GAEC9013

Contents

1. INTRODUCTION

OpenGL is a library for doing computer graphics. By using it, we can create interactive applications which render high-quality color images composed of 3D geometric objects and images. OpenGL is window and operating system independent. As such, the part of our application which does rendering is platform independent.

However, in order for OpenGL to be able to render, it needs a window to draw into. Generally, this is controlled by the windowing system on whatever platform we are working on. As OpenGL is platform independent, we need some way to integrate OpenGL into each windowing system.

Overview of the project

In this Mini-project 'Music Concert', the computer graphics concepts along with the OpenGL functions will be used. The main objective of this graphics mini project is to recreate a music concert using the OpenGL API and the GLUT library.

This project will have a view of a music concert in which various animated characters will be performing music on a stage.

There will be a continuous scrolling of text across the screen from left to right at the top of the stage which reads "WELCOME". The various OpenGL fonts will be used in displaying the texts in the program. There will also be speakers in the side which can be animated in a way that resembles music coming out of it.

The graphics will also contain a lighting for the performer on the stage which can be toggled between on and off using the keyboard functions. This project will have keyboard interaction. This program will be designed in such a way that when a specific key is pressed the singer will be changed.

When a specific key is pressed the confetti will be showered from above. The confetti will be programmed using random colors and sizes. By default, whenever a performer is changed confetti will be showered.

Fireworks will go on in the background periodically and automatically. These fireworks can also be controlled using the keyboard.

The rendering will also contain several animated characters as audience with different colored clothing and different hair and skin colors.

The concepts of computer graphics and OpenGL stand a backbone to achieve the aforementioned idea. Primitive drawing, event driven interactions and basic animation have been the important concepts brought out by this application.

The report is chalked out into sections describing the basic requirements superseded by the briefing on functions used. Following this, the detailed description of how the implementation is done effectively using these functions and C language is presented. The source code is provided along with necessary comments to enhance readability of code. The screenshots have been provided for amelioration of our little effort. The conclusion and the future enhancements proposed conclude the report. The maximum efforts are been made to ensure that the view is aesthetically pleasing and eye-catching.

General Constraints

1. As software is being built to run on a WINDOWS platform, efficient use of the memory is very important.

2. The code should be efficient and optimal with the minimal redundancies.

3. Needless to say, the package should also be robust and fast.

Assumptions and Dependencies

1. It is assumed that the standard output device, namely the monitor, supports colors and users system is required to have the C compiler for the appropriate version.

2. The system is also expected to have a mouse connected since most of the drawing and other graphical operations implemented assume the presence of a mouse.

2. REQUIREMENTS SPECIFICATION

The requirement specifications of this project is not perfectly optimized. However the following hardware and software specifications were done to be of my best effort. Here are the specifications:

1. Hardware Requirements:

The hardware requirements given here is minimal requirements for the project to run even though the project can smoothly run on almost all i3h86 machines.

• Processor Speed	-300 MHz and above
• RAM Size	-64 Mb or above
• Storage Space	-2MB or above

2. Software Requirements:

• Operating System	-Windows Family
• Compiler	- CODEBLOCKS
• Graphics Library	- GL/glut.h, GL/glu.h, GL/gl.h
• Programming Language	-C using OpenGL

3. ABOUT OPENGL

The OpenGL specification describes an abstract for drawing 2D and 3D graphics. Although it is possible for the API to be implemented entirely in software, it is designed to be implemented mostly or entirely in hardware.

The API is defined as a number of functions which may be called by the client program, alongside a number of named integer constants. the function definition are superficially similar to those of the C programming language, they are independent. As such, OpenGl has many language bindings, some of the most noteworthy being the JavaScript binding WebGL(API based on OpenGL ES2.0);the C bindings WGL ,GLX, CGL; the C binding provided by iOS and the Java and C binding provided by the Android.

In addition to being language-independent, OpenGL is also platform independent. The specification says nothing on the subject of obtaining, managing an OpenGL context, leaving this as a detail of the underlying windowing system. For the same reason, OpenGL is purely concerned with the rendering, providing no APIs related to input, audio, or windowing.

4. DESIGN AND IMPLEMENTATION

DESIGN

4.1 Header files and their description

```
#include <GL/glut.h>

#include <math.h>

#include <stdio.h>

#include<string>
```

• GL/glut.h: This header files provides an interface with application programs that are installed on the system.

• stdio.h: Input & Output operations can also be performed in C using C standard input, output library.

• math.h: Contains constant definitions and external subordinate declarations for the math subroutine library.

• string: This header introduces string types, character traits and a set of converting functions.

4.2 Description of OpenGL functions

Glut library functions used are:

int main(int argc, char argv):** Execution of any program always starts with main function irrespective of where it is written in a program. It also contains the call to display, reshape, creating window and keyboard function.

glutSwapBuffers(void): Performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers promotes the contents of the back buffer of the layer in use of the current window to become the contents of the front buffer. The contents of the back buffer then become undefined. The update typically takes place during the vertical retrace of the monitor, rather than immediately after glutSwapBuffers is called.

glutPostRedisplay(): Marks the plane of current window as needing to be redisplayed. The next iteration through glutMainLoop, the window's display callback will be called to redisplay the window's normal plane. Multiple calls to glutPostRedisplay before the next display callback opportunity generates only a single redisplay callback.

glutInit(&argc,char ** argv): glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized.

glutInitDisplayMode(unsigned int mode): The initial display mode is used when creating top-level windows, sub windows, and overlays to determine the OpenGL display mode for the to-be created window or overlay. GLUT_RGB specifies the structure of each pixel. GLUT_DEPTH is a buffer to hold the depth of each pixel.

glutCreateWindow(char *title): The glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name.

glutReshapeFunc(void(* callback)(int, int)): glutReshapeFunc sets the reshape callback for the current window. The reshape callback is triggered when a window is reshaped. A reshape callback is also triggered immediately before a window's first display callback after a window is created or whenever an overlay for the window is established. The width and height parameters of the callback specify the new window size in pixels. Before the callback, the current window is set to the window that has been reshaped. glutMainLoop():glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

glutBitmapCharacter(void *font, int c): Without using any display lists, glutBitmapCharacter renders the character ‘c’ in the named bitmap font.

glPointSize(GLfloat size): glPointSize specifies the rasterized diameter of points.

glFlush(): Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerator itself. glFlush empties all of these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.

glRasterPos3f(GLfloat x , GLfloat y, GLfloat z): This function is used to set the cursor at the x, y, z location.

glBegin(GLenum) and glEnd(): glBegin and glEnd delimit the vertices that define a primitive or a group of like primitives. glBegin accepts a single argument that specifies in which often ways the vertices are interpreted.

glColor3f(GLfloat red, GLfloat green, GLfloat blue): This function is used to pick a color of specified R,G,B value.

glClear(GLbitfield): glClear sets the bit plane area of the window to values previously selected by glClearColor, glClearIndex, glClearDepth, glClearStencil, and glClearAccum. Multiple color buffers can be cleared simultaneously by selecting more than one buffer at a time using glDrawBuffer.

glVertex3f(GLfloat x, GLfloat y, GLfloat z): This function is used to draw a vertex at location x, y,z.

glEnable(GLenum): This function is used to enable the various functionalities like enabling the light, enabling the Z buffer.

glDisable(GLenum): This function is used to disable the various functionalities like enabling the light, enabling the Z buffer.

glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha):glClearColor specifies the red, green, blue, and alpha values used by glClear to clear the color buffers. Values specified by glClearColor are clamped to the range 0 to1.

glutDisplayFunc(void (*func)()): glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the current window is set to the window needing to be redisplayed and (if no overlay display callback is registered) the layer in use is set to the normal plane. The display callback is called with no parameters.

glutKeyboardFunc(void (*func)(unsigned char key, int x, int y)): glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character.

glMatrixMode(GLenum) mode: Specifies which matrix stack is the target for subsequent matrix operations. Values accepted are: GL_MODELVIEW, GL_PROJECTION. The initial value is GL_MODELVIEW. Additionally, if the ARB tension extension is supported, GL_COLOR is also accepted.

gluPerspective(GLDouble fovy, GLDouble aspect, GLDouble near, GLDouble far): gluPerspective specifies a viewing frustum into the world coordinate system. In general, the aspect ratio in gluPerspective should match the aspect ratio of the associated viewport. For example, aspect = 2.0 means the viewer's angle of view is twice as wide in x as it is in y.

4.3 Description of User-defined functions

Various user-defined functions are:

float RandomBetween(float smallNumber, float bigNumber): Generates random number between smallNumber and bigNumber.

void initialize(void): Initializes firework particles for the firework particle system. The fire particles have a random x co-ordinates each time the function is called.

void enablefunc(void): This function enables the various functionalities used in the program.

void InitGL(void): Initializes confetti particles for the confetti particle system. The confetti particles are programmed using random colors.

void drawText(const char *text, int length, float x, float y, float z): The drawText function draws the text in the position specified by x, y and z. This function uses the glRasterPos3f() function to set the cursor at the position passed as arguments.

void drawFilledCircle(GLfloat x, GLfloat y, GLfloat z): This function is used to draw a filled circle using Triangle Fans with the center specified by (x, y, z).

void DrawSticks(float cx, float cz, float r, int num_segments): DrawSticks function draws vertical sticks around a circle in the x-z plane of radius r and center (cx, cz). The smoothness of the circle is specified by num_segments.

void DrawCirclexy(float cx, float cy, float r, int num_segments): It is used to draw an hollow circle in the x-y plane with radius r and center (cx, cy). The smoothness of the circle is specified by num_segments.

void timer(int): This timer function is used to periodically change the values of x co-ordinates of the text on the display board and the y co-ordinates of the drum sticks and the height of the cylinder on the speaker.

void ctimer(int): It is used to automatically turn off the confetti shower after a singer is changed. The ctimer function is called 5 seconds after the change in singer.

void draw_confetti(void): This function draws the confetti particles when called. This function uses particle system to draw the confetti particles. The confetti particles are programmed using random colors.

void draw_human(GLfloat sc[], GLfloat pc[], GLfloat hc[], GLfloat s[], int c): The draw_human function is used to draw the audience members of the concert. It accepts the shirt color, pant color, hair color and the skin color of the audience. It also accepts whether the audience member has a cap on or not.

void drum(void): This function draws a drum using gluCylinder() and drawFilledCircle() and DrawSticks().

void draw_speaker(void): This function draws a speaker using polygons, gluCylinder and DrawCirclexy().

void draw_blast(void): The draw_blast function draws the firework particles of the firework particle system along with the various objects of the scene such as stage, singers, audience, red carpet, etc. It also has the function calls for draw_speaker(), drum(), draw_human(), draw_confetti(), etc

IMPLEMENTATION

To achieve three dimensional effects, OpenGL software is proposed. It is software which provides a graphical interface. It is an interface between application program and graphics hardware. The advantages are:

- OpenGL is designed as a streamlined**.**
- It is a hardware independent interface, it can be implemented on many different hardware platforms.
- With OpenGL, we can draw a small set of geometric primitives such as points, lines and polygons etc.
- It provides double buffering which is vital in providing transformations.
- It is event driven software.
- It provides call back function.

Transformation Functions

- **Translation:**

  Translation is done by adding the required amount of translation quantities to each of the points of the objects in the selected area. If P(x,y) be the a point and (tx, ty) translation quantities then the translated point is given by

  glTranslatef(dx,dy,dz) ;

- **Rotation:**

  The rotation of an object by an angle 'a' is accomplished by rotating each of the points of the object. The rotated points can be obtained using the OpenGL functions

  glRotatef(angle, vx,vy,vz);

- **Scaling:**

  The scaling operation on an object can be carried out for an object by multiplying each of the points (x,y,z) by the scaling factors sx, sy and sz.

  glScalef(sx,sy,sz);

5. SNAPSHOTS

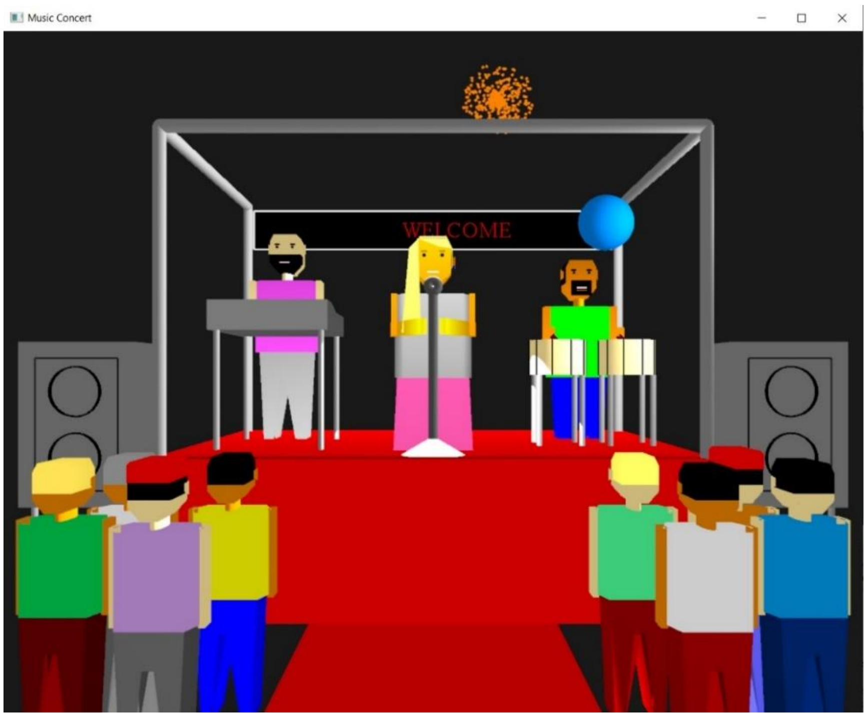

FIG 5.1 SNAPSHOT WITH FEMALE SINGER IN FRONT VIEW ALONG WITH FIREWORKS IN THE BACKGROUND. SPOTLIGHTS ARE ON.

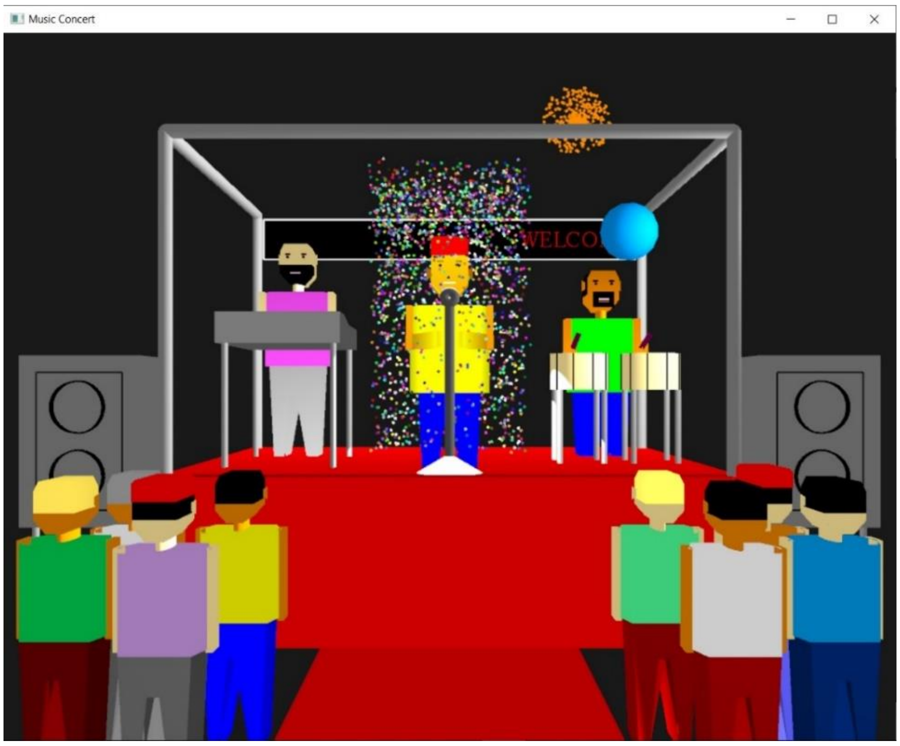

FIG 5.2 SNAPSHOT WITH MALE SINGER WITH CONFETTI SHOWER IN FRONT VIEW ALONG WITH FIREWORKS IN THE BACKGROUND. SPOTLIGHTS ARE ON.

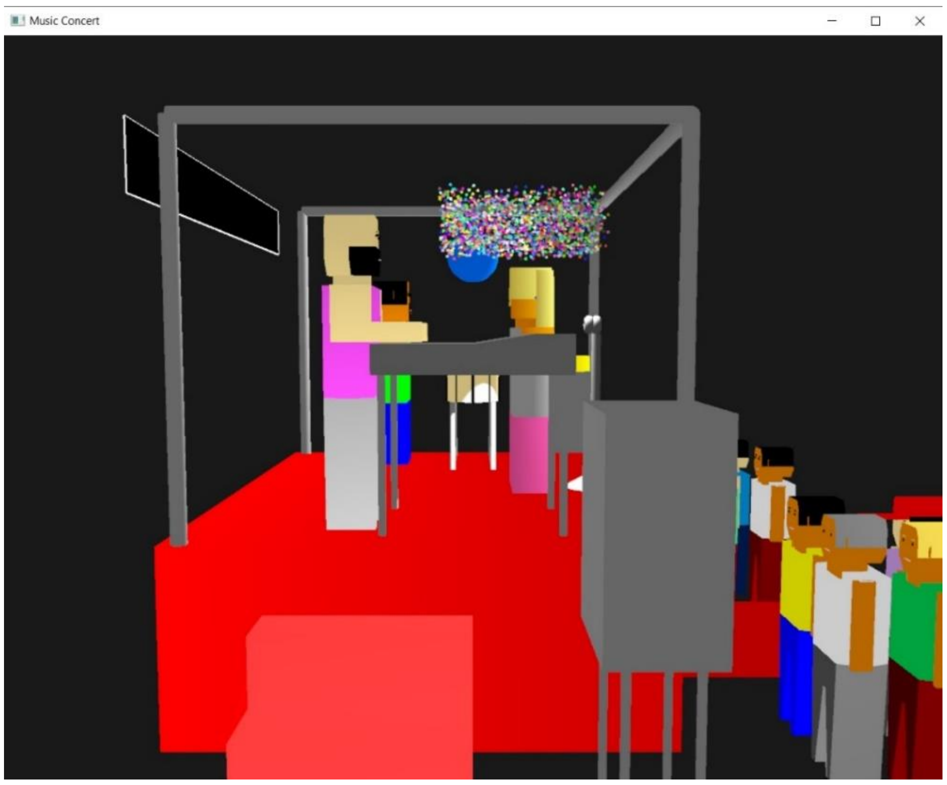

FIG 5.3 SNAPSHOT WITH FEMALE SINGER IN LEFT SIDE VIEW AND CONFETTI SHOWER HAS JUST BEGUN. SPOTLIGHTS ARE ON.

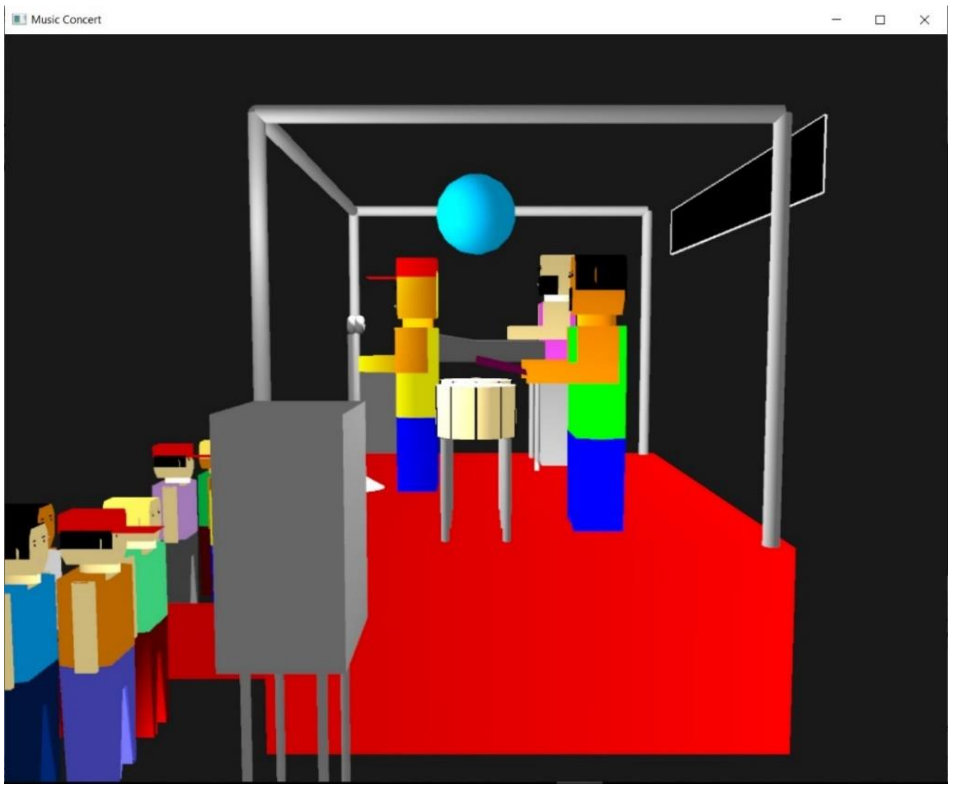

FIG 5.4 SNAPSHOT WITH MALE SINGER IN RIGHT SIDE VIEW. SPOTLIGHTS ARE ON.

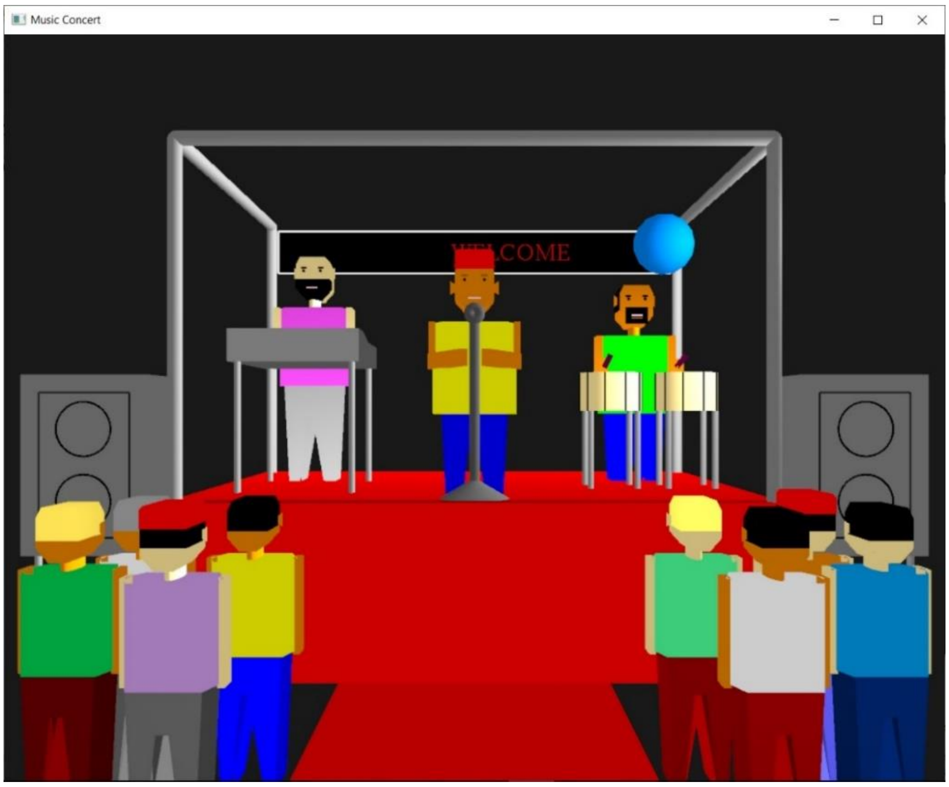

FIG 5.5 SNAPSHOT WITH MALE SINGER IN FRONT VIEW. SPOTLIGHTS ARE OFF.

6. CONCLUSION

Using OpenGL any form of images, designs and animations can be produces and also, we can give realistic effects to it like in our project.

In OpenGL, we get smooth surface to draw the objects and we have provided texture and lighting effect to the objects.

We have used many built-in functions which are user friendly. We have tried to use many of the graphic concepts using OpenGL such as translation, rotation, scaling, timer functions, particle systems, lighting, bitmap fonts, etc.

The project enabled to work with mid-level. OpenGL complexity and the project demonstrates the scope of OpenGL platform as a premier game developing launch pad. Hence, it has indeed been useful in developing many games. OpenGL in its own right is good for low cost and simple game development. It serves as an important stepping stone for venturing into other fields of Computer Graphics design and applications.

7. REFERENCE

- Interactive Computer graphics by Edward Angel
- Opengl.org
- Stackoverflow.com
- www.glprogramming.com
- www.videotutorialsrock.com