VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELGAVI, KARNATAKA -590 018

A Mini-Project Report on

"ROTATING STARS"

Submitted in partial fulfillment for the Computer Graphics Laboratory with Mini-Project (18CSL67) course of Sixth Semester of Bachelor of Engineering in Computer Science & Engineering during the academic year 2021-22.

By

Team Code: **CGP2022-A05**

PRAJWAL N S	**4MH19CS070**
GURU RATHNAKAR	**4MH19CS033**

: Under the Guidance of:

Prof. Santhosh E

Assistant Professor

Department of CS&E

MIT Mysore

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belawadi, S.R. Patna Taluk, Mandya Dist-571477.

Accredited By: NATIONAL BOARD OF ACCREDITATION

2021-22

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

~~ CERTIFICATE ~~

*Certified that the mini-project work entitled "**ROTATING STARS**" is a bonafide work carried out by **PRAJWAL N S** (4MH19CS070) & **GURURATHNAKAR** (4MH19CS033) for the Computer Graphics Laboratory with Mini-Project (18CSL67) of Sixth Semester in Computer Science & Engineering under Visvesvaraya Technological University, Belgavi during academic year 2021-22.*

It is certified that all corrections/suggestions indicated for Internal Assignment have been incorporated in the report. The report has been approved as it satisfies the course requirements.

__________________	__________________
Signature of Guide	Signature of the HOD
Prof. Santosh E	**Dr. Shivmurthy R C**
Assistant Professor	Professor & HOD
Dept. of CS&E	Dept. of CS&E
MIT Mysore	MIT Mysore

External viva

Name of the Examiners **Signature with date**

1)…………………………………………………………………………………

2) …………………………………………………………………………………

~~~~ ACKNOWLEDGEMENT ~~~~

It is the time to acknowledge all those who have extended their guidance, inspiration, and their wholehearted co-operation all along our project work.

We are also grateful to **Dr. B G Naresh Kumar**, principal, MIT Mysore and **Dr. Shivmurthy R C**, HOD, CS&E, MIT Mysore for having provided us academic environment which nurtured our practical skills contributing to the success of our project.

We wish to place a deep sense of gratitude to all Teaching and Non-Teaching staffs of Computer Science and Engineering Department for whole-hearted guidance and constant support without which Endeavour would not have been possible.

Our gratitude will not be complete without thanking our parents and our friends, who have been a constant source of support and aspirations

Prajwal N S
Guru Rathnakar

~~~ ABSTRACT ~~~

The Rotating star project is designed and implemented using openGL interactive application that basically deals with providing the graphical interfaces between user and system.

This project creates stars which rotate in the form of a circle.
The project demonstrates how the stars rotate when the right button of the mouse is clicked. It has options like start, stop and quit.

Similar cases are shown through a graphical representation with proper mouse interactions and responses. The project has been developed in Visual C++ using Open GL functions. It is simple representation of images using pictures and simple animation.

~~~~~ CONTENTS ~~~~~

CHAPTER-1

INTRODUCTION

Aim of the Project: Program to display the rotation Effect on the stars by using the Keyboard and Mouse

Overview of the Project:

Computer Graphics is a complex and diversified technology. To understand the technology it is necessary to subdivide it into manageable Parts. This can be accomplished by considering that the end product of computer graphics is a picture. The picture may, of course, be used for a large variety of purposes; e.g., it may be an engineering drawing, an exploded parts illustration for a service manual, a business graph, an architectural rendering for a proposed construction or design project, an advertising illustration, or a single frame from an animated movie. The picture is the fundamental cohesive concept in computer graphics.

Consider how: Pictures are represented in computer graphics.

- Pictures are prepared for presentation.
- Previously prepared pictures are presented.
- Interaction with the picture is accomplished.

Here “picture” is used in its broadest sense to mean any collection of lines, points, text, etc. displayed on a graphics device

A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.

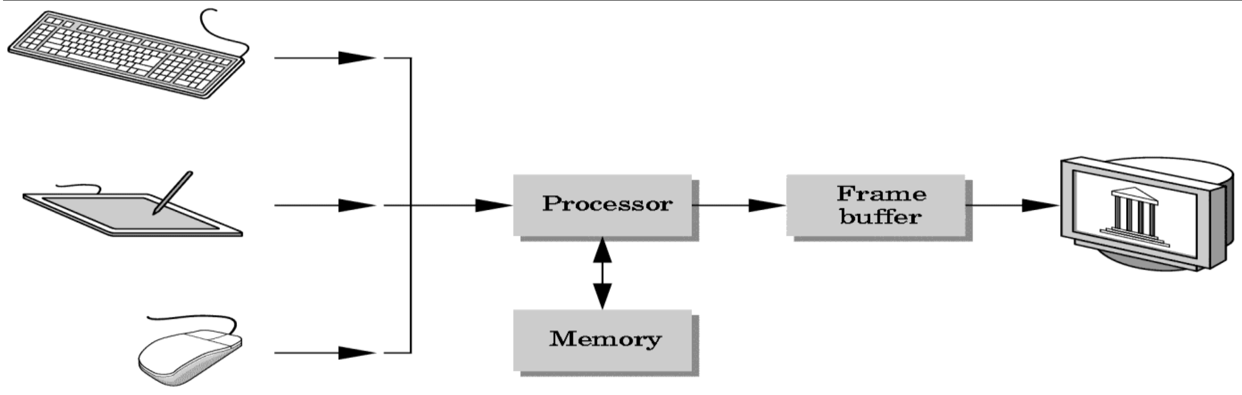


Fig. 1.1: A graphics system

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

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.

1.3Outcome of the Project

PROJECT DESCRIPTION:

This project is about rotating the stars . As the key is pressed the stars are displayed. When the start button is pressed the stars start rotating it will be continue until the stop button is pressed.

We have used functions like push and pop matrix. And other simple functions like GL polygon to create the stars.

The stars are given different colours like red,pink,blue ,purple etc.

The keyboard interfaces change the background color,rotate in clockwise and anticlockwise directions.
The program starts and after clicking the right button we get options like 'start animation', 'stop animation' and 'quit'.

In the keyboard function the keys used are:

- m: rotates in anticlockwise direction.
- s: changes the background color.
- r: rotates in clockwise direction.

In the menu section you will have to choice between the 3 options:

- Start Animation will start the moment of the stars.
- Stop Animation will stop the moment of all the stars.
- Quit will terminates the program.

REQUIREMENTS SPECIFICATION

Hardware Requirements:

- Intel® Pentium 4 CPU and higher versions
- 128 MB or more RAM.
- A standard keyboard, and Microsoft compatible mouse
- VGA monitor.

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
- Language: C
- Code ::Blocks: cross-platform Integrated Development Environment (IDE)

2 . Design And Implementation

2.1Algorithm

This project has been developed using visual C++, which is a function oriented language. Function oriented design conceals the details of an algorithm in a function but system state information is not hidden. This can cause problems because a function can change the state away, which other functions don't expect. Changes to a function and the way in which it uses a system state may cause unanticipated changes in the behavior of other functions. Functional approach to design is therefore most likely to be successful when the amount of system state information is minimized and information sharing is explicit, so the implementation technique used is procedural is a general purpose structured programming language that is powerful, efficient and compact. Compiler combines the capability of an assembly language with the features of a high level languages and therefore is well suited for writing both software and business packages. Programs written in C are fast and efficient. This is due to variety of data types and powerful operators. C is highly portable.

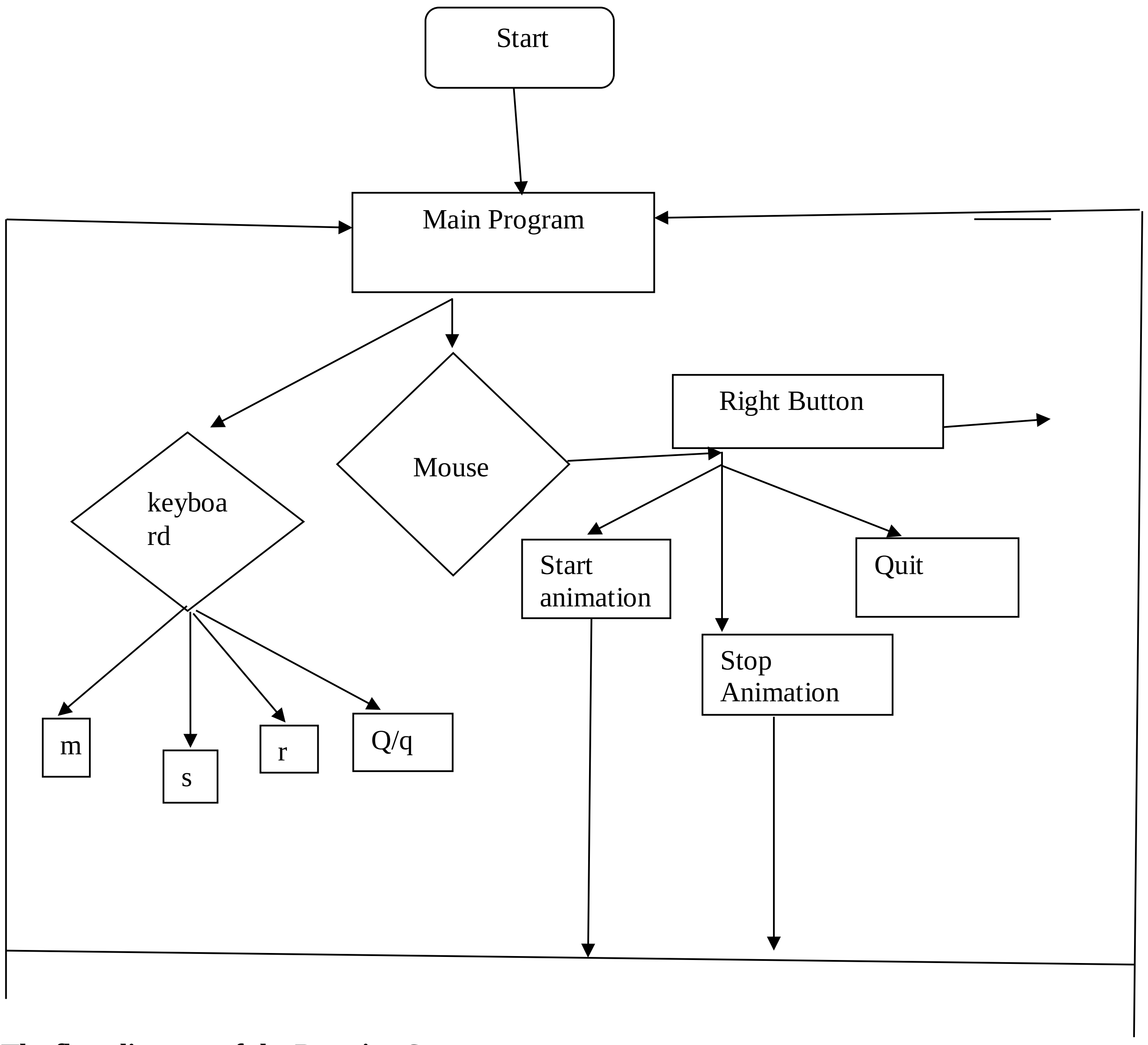
In this Visual C++ we have used Open GL APIs to build our project like glVertex2f(), glBegin(), etc. with different color functions that helped to build the graphical design a lot. The Open GL APIs provides a class of libraries for the user with the set of facilities to handle the graphics operations efficiently. So the implementation written in C++ programming language also includes many header file.

Design is the planning that lays the basis for the making of every object or system. This chapter involves the designing the various aspects and different stages of the
Project. When the program is made to execute, the output window is displayed first.

The flow of the operation from the output window is shown in the figure below.

2.2 Flow Diagram:

Flowchart is the diagrammatic representation of algorithm showing the steps as boxes and their order by connecting with arrows. It shows the flow of execution when different options are specified.



The flow diagram of the Rotating Stars

Flowchart is the diagrammatic representation of algorithm showing the steps as boxes and their order by connecting with arrows. It shows the flow of execution when different options are specified.

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.

2.3OpenGL API’S Used With Description

OpenGl Functions Used:

This project is developed using CodeBlocks and this project is implemented by making extensive use of library functions offered by graphics package of OpenGl, a summary of those functions follows:

glBegin() :

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

glPushMatrix() :

*‘ void **glPushMatrix**(void)’*

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

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.

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.

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.

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.

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.

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.

glLoadIdentity() :

*‘void **glLoadIdentity**(void);’*

It replaces the current matrix with the identity matrix.

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.

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.

glFlush() :

*‘void **glFlush**(void);’*

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

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.

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.

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.

glColor3f

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

PARAMETERS:

- Red: The new red value for the current color.
- Green: The new green value for the current color.
- Blue: The new blue value for the current color.

Sets the current color.

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.

glutKeyboardFunc():

```
void mykey(unsigned char key,int x,int y)

{

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

}
```

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.

glutMainLoop ()

‘void *glutMainLoop(void); glutMainLoop();’*

glutMainLoop enters the GLUT event processing loop.

glutMouseFun():

```
void mouse(int button,int state,int x,int y)
{
if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
glutIdleFunc(idle);
if(button==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
glutIdleFunc(NULL);
}
```

glEnable():

‘void glEnable(GLenum cap);’

‘glEnable(GL_CULL_FACE);’

PARAMETERS:

cap: A symbolic constant indicating an OpenGL capability.

The glEnable enable OpenGL capabilities.

IMPLEMENTATION

Implementation is the realization of an application, or execution of a plan, idea, or model.
This chapter provides an insight of how the code has been developed.

The pseudocode & the flow of execution from one function to another is shown in this section.

Functions:

```
Void mouse( )
{
      Provides mouse interface;
}
Void myIdleFunc()
{
      Rotates the stars;
}
Void Menu( )
{
     Shows the menu with start, stop and quit options;
}
myKey( )
{
      Shows the keyboard interfaces;
}
Mouse( )
{
     Shows the mouse interfaces;
}
Void display( )
```

```
{
        Enable lightning;
        Enable cull face;
        Materials have ambient, diffuse and specular lightning;
        Initialize positional light and ambient light;
        Define display list to find angle of views, draw floor, control lights, change color
        and size of the object;
}
```

2.4 Source Code:

```
/*
*   ROTATING STARS
*
*
*/
#include<stdio.h>
#include <windows.h>
#include<stdlib.h>
#include<math.h>
#include<GL/glut.h>
#define INNER_RADIUS 0.10
#define OUTER_RADIUS 1.0
#define NUM_STAR_POINTS 9
#define NUM_STARS 9
#define ROT_INC 0.9
#define STAR_IDX 2
#define M_PI 22/7
static GLfloat g_rotate=0;
int scene;
static GLfloat g_rotateInc=ROT_INC;
```

```
GLfloat xstep = 1.0f;
GLfloat ystep = 1.0f;
GLsizei rsize = 50;
 int al=1;
// Keep track of windows changing width and height
GLfloat windowWidth;
GLfloat windowHeight;
int flag=0;
int a=0;
void Draw_Stars(GLfloat inner,GLfloat outer,int numpoints)
{
GLfloat step=M_PI*2.0/(GLfloat)(2*numpoints);
register int i;
GLfloat angle,r;
glBegin(GL_POLYGON);
for(i=0;i<numpoints*2;++i)
{
r=(i%2==0?inner:outer);
angle=i*step;
glVertex2f(r*cos(angle),r*sin(angle));
}
glEnd();
}
void Background1(void)
{
        glColor3f(1.0,0.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(-600,-600);
glVertex2f(-600,600);
glVertex2f(600,600);
glVertex2f(600,-600);
```

```
glEnd();
glFlush();
}
void Background2(void)
{          if(al!=0)
{
glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(-600,-600);
glVertex2f(-600,600);
glVertex2f(600,600);
glVertex2f(600,-600);
glEnd();
glFlush();
glutSwapBuffers();
}          }

void Display(void)
{
register int i;
register GLfloat c;
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glClear(GL_COLOR_BUFFER_BIT);
switch(scene)
{
  case 1:


for(i=0;i<NUM_STARS;++i)
{
glPushMatrix();
glRotatef((360.0/NUM_STARS*i),0,0,1);
glTranslatef(OUTER_RADIUS,0,0);
if(a==1)
        glRotatef(g_rotate+(3*i),0,1,1);
else
        glRotatef(g_rotate-(3*i),1,0,1);
c=1.0/NUM_STARS*(GLfloat)i;
glColor3f(1.0-c,0.0,c);
glCallList(STAR_IDX);
glPopMatrix();
}
break;
}
glFlush();
glutSwapBuffers();
}
void myReshape(int w,int h)
{
glViewport(0,0,w,h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
if(w<=h)
glOrtho(-2.0,2.0,-2.0*(GLfloat)h/(GLfloat)w,2.0*(GLfloat)h/(GLfloat)w,-2.0,2.0);
else
glOrtho(-2.0*(GLfloat)w/(GLfloat)h,2.0*(GLfloat)w/(GLfloat)h,-2.0,2.0,-2.0,2.0);
glMatrixMode(GL_MODELVIEW);
}
void menu(int value){
switch(value){
case 1:Display();
                break;
```

```
         case 2: glutIdleFunc(NULL);
        break;
case 3: exit(0);
                  break;
                 }
}
void myKey(unsigned char k,int x ,int y)
{
switch(k){
case'm':
          {
                   a=1;
                   glClear(GL_COLOR_BUFFER_BIT);
                   //Background2();
                    glFlush();
                             break;
          }
case's':
          {
                   glClearColor(1.0,1.0,1.0,0);
                   glClear(GL_COLOR_BUFFER_BIT);
                   glMatrixMode(GL_MODELVIEW);
                   glLoadIdentity();
                            glFlush();
                   break;
          }
case'r':
          {
                   a=0;
                   Background1();
                            break;
```

```
            }
case'q':
case'Q': exit(0);
            break;
        case 'n':
            scene=scene+1;
            break;
default:
            printf("unknown printable command\'%c\'.\n",k);
                        break;

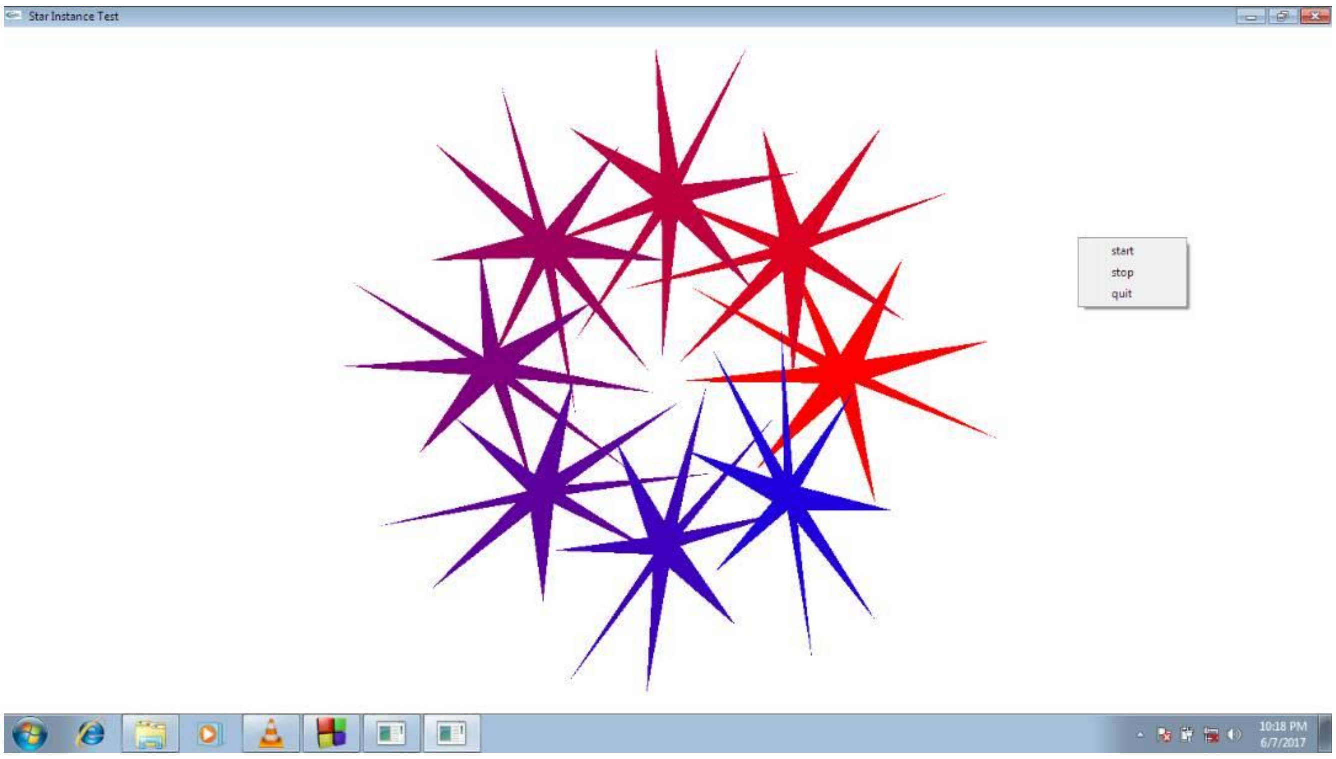


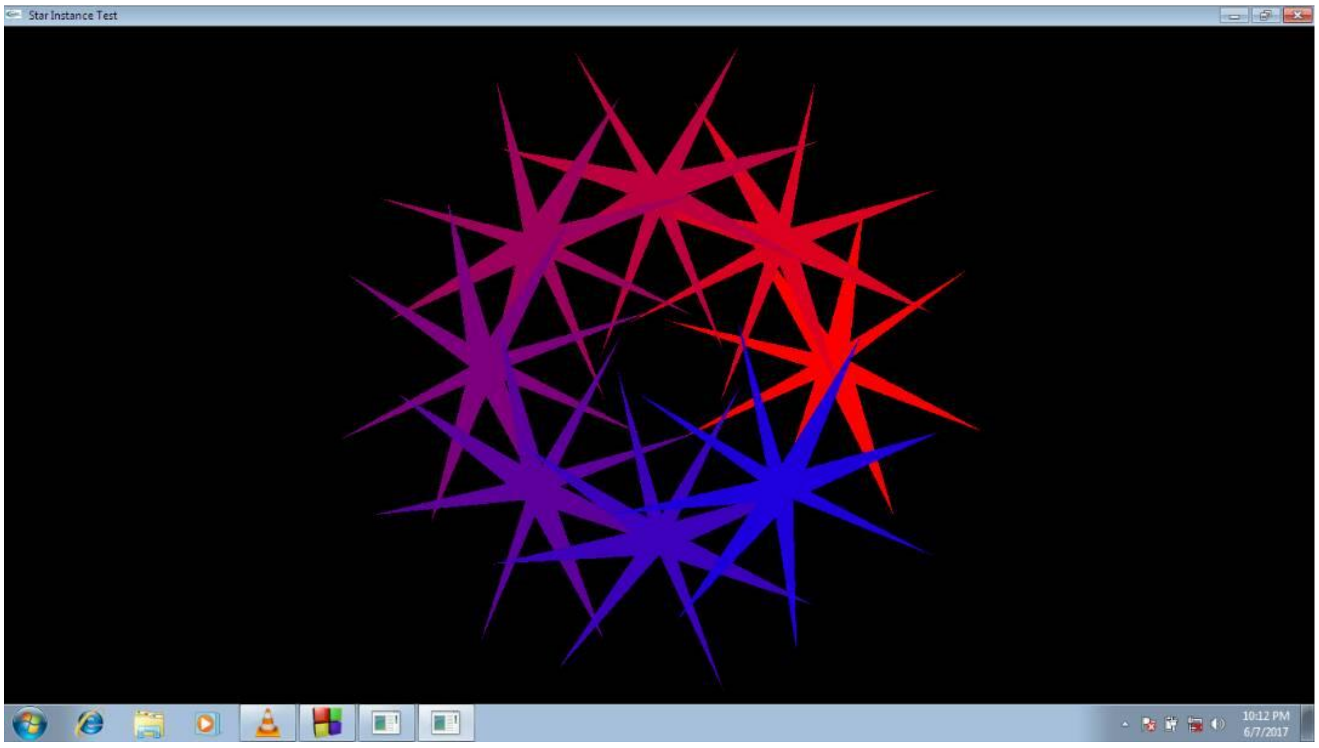
             }
}
void myMouse(int btn,int state,int x,int y)
{
if(btn==GLUT_LEFT_BUTTON&&state==GLUT_DOWN)g_rotateInc=ROT_INC;
if(btn==GLUT_MIDDLE_BUTTON&&state==GLUT_DOWN)g_rotateInc=ROT_INC;
if(btn==GLUT_RIGHT_BUTTON&&state==GLUT_DOWN)g_rotateInc=ROT_INC;
}
void myIdleFunc(void)
{
if(a==0)
{
g_rotate-=g_rotateInc;
glutSwapBuffers();
glutPostRedisplay();
}
else{
g_rotate+=g_rotateInc;
glutSwapBuffers();
```

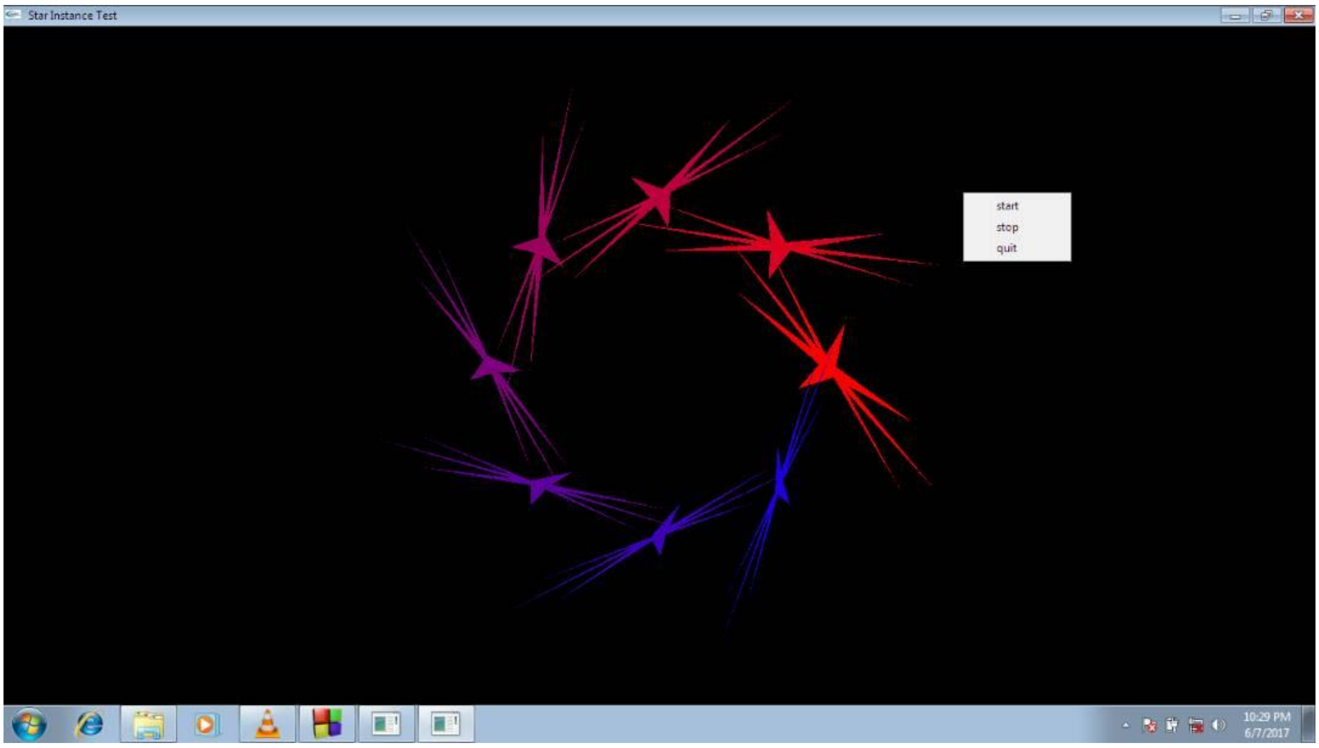
```
glutPostRedisplay();
}
}int main(int argc,char **argv){
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
glutInitWindowSize(800,800);
glutInitWindowPosition(10,10);
glutCreateWindow("Star Instance Test");
glutCreateMenu(menu);
glutAddMenuEntry("start",1);
glutAddMenuEntry("stop",2);
glutAddMenuEntry("quit",3);
glutAttachMenu(GLUT_RIGHT_BUTTON);
glutReshapeFunc(myReshape);
glutDisplayFunc(Display);
glutIdleFunc(myIdleFunc);
glutKeyboardFunc(myKey);
glutMouseFunc(myMouse);
glNewList(STAR_IDX,GL_COMPILE);
Draw_Stars(INNER_RADIUS,OUTER_RADIUS,NUM_STAR_POINTS);
glEndList():
glutMainLoop();}
```

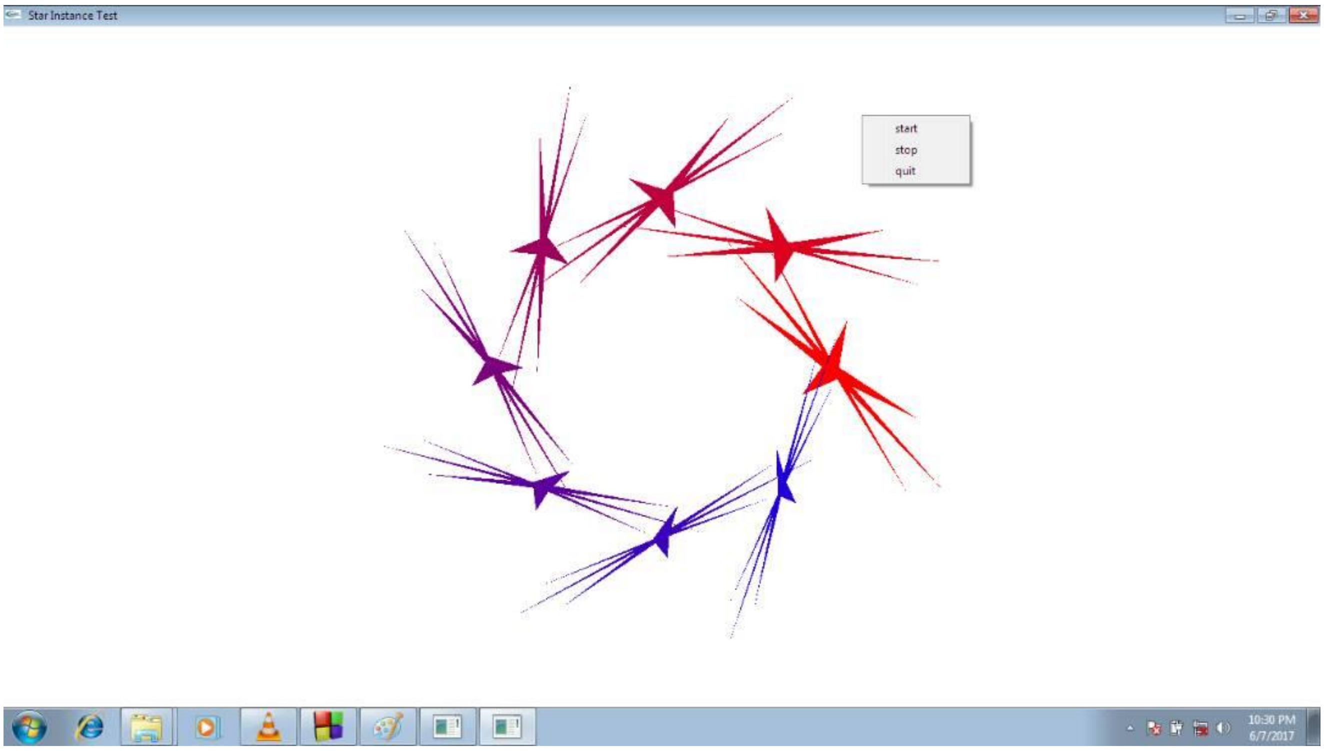
3. RESULT ANALYSIS:

SNAPSHOTS:






Star Instance Test
start
stop
quit
10:29 PM
6/7/2017


Star Instance Test
start
stop
quit
10:30 PM
6/7/2017

3.2 Discussion:

The Above Snapshots shows that how the project would be while execution
In the figure1 the stars rotate in the white background & in figure2 the background changes
To black When the S is pressed, as in figure3 when the right button is pressed the Menu will be opened
With the options START, STOP, QUIT each option will works in their own Manner, the figure4
Is also same but with Black background

4. CONCLUSION AND FUTURE WORK:

4.1 CONCLUSION:

This graphics package has been successfully implemented in Visual C++ using Open GL APIs in windows-based platform. In this animation we have used various library functions available in Visual C++ and Open GL.

It is a simple representation of images using 3D pictures and simple 3D animation. This can be well implemented using any image support processing language.

This project could be still implemented in a better way with good interface and animations. I welcome constructive comments from the critics so that our project could be made still better

4.2FUTURE ENHANCEMENT:

- We can try to add actual color to the object drawn in this program.
- We can try to improve the quality of the object drawn.

5. REFERENCES

[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.