# JCEI’s JAIHIND COLLEGE OF ENGINEERING, KURANpage



**DEPARTMENT OF AI & DS ENGINEERING**

## CERTIFICATE

This is to certify that the Manual under the subject

**“Computer Graphics (217523)”**

## SUBMITTED BY

**Exam Seat No. S190842102**

Is a Bonafede work carried out by student under the supervision of  **Prof.** **Kale A. S.** and it is

Submitted towards the partial fulfillment of the requirement of Second Year of AI & DS Engineering.

#### Prof. Kale A. S. Prof. Said S. K.

**Subject Teacher Head of Department**

#### Dept. of AI & DS Engineering. Dept. of AI & DS Engineering.

**Dr. D. J. Garkal Principal**

#### JCEI’s JAIHIND COLLEGE OF ENGINEERING, KURAN

|  |  |
| --- | --- |
| **Subject** | **Computer Graphics (217523)** |
| **Name** | **Thorve Avishkar Shrikrushna** |
| **Class** | **Second Year. (AI& DS Engineering.)** |
| **Roll No.** | **65** |
| **Exam Seat No.** | S190842102 |
| **Date** |  |
| **Signature** |  |

**INDEX**

Abstract …………………………... 4

Acknowledgement …………………………… 5

Introduction …………………………… 6

Project Description …………………………… 9

Diagram …………………………… 11

Program Code …………………………… 11

Conclusion …………………………… 26

References ……………………………. 27

**ABSTRACT**

This project is about the creation of Olympics logo. We are implementing it using using different primitives available in OpenGL library and combining them together in a required manner.

It highlights the key features of the data structures and its high quality efficiency that is obtained on its usage in the application program. This project consists of Olympic logo which is constructed by using different primitives available in OpenGL library. It illustrates the role of different callback functions that provides easier way to accomplish our project in an effective manner.

The project has been implemented by efficiently using the data structures to obtain the optimized results and also various functions and features that are made available by the OpenGL software package have been utilized effectively.

**ACKNOWLEDGEMENT**

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

We are also grateful to Dr. G. L. Shekar, principal of N. I. E., Mysore and also Dr. K. Raghuveer, HOD of ISE Department 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 our beloved lab staff incharge smt. C. K. Vanamala and sri Darshan A. H., for their whole-hearted guidance and constant support without which Endeavour would not have been possible.

Our gratitude will not be complete without thanking the Almighty God and our beloved parents and also our friends, who have been a constant source of blessings and aspirations.

**INTRODUCTION**

Computers have become 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, and so it is not surprising to find the use of computer graphics so widespread. Although early applications in engineering and science had to rely on expensive and cumbersome equipment, advances in computer technology have made interactive computer graphics a practical tool. Today, we find computer graphics used routinely in such diverse areas as science, engineering, medicine, business, industry, government, art, entertainment, advertising, education, and training.

Computer graphics are [graphics](http://en.wikipedia.org/wiki/Graphics) created using [computers](http://en.wikipedia.org/wiki/Computer) and, more generally, the [representation](http://en.wikipedia.org/wiki/Representation) and [manipulation](http://en.wikipedia.org/wiki/Manipulation) of [image](http://en.wikipedia.org/wiki/Image) [data](http://en.wikipedia.org/wiki/Data) by a [computer](http://en.wikipedia.org/wiki/Computer). The development of computer graphics has made computers easier to interact with, and better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized [animation](http://en.wikipedia.org/wiki/Animation), [movies](http://en.wikipedia.org/wiki/Movies) and the [video game](http://en.wikipedia.org/wiki/Video_game) industry.

A major use of computer graphics is in design processes, particularly for engineering and architectural systems, but almost all products are now computer designed. Generally referred to as CAD, computer-aided design methods are now routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles, and many, many other products.

Here we have used “OpenGL” as the graphics software system to implement our mini project, “Locomotive Olympics logo. Now let us have a quick look at OpenGL.



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.

Every windowing system where OpenGL is supported has additional API calls for managing OpenGL windows, color maps and other features. These additional APIs are platform dependent. For the sake of simplicity, we are using an additional freeware library for simplifying interacting with windowing systems, GLUT.

GLUT, the OpenGL Utility Toolkit is a library to make writing openGL programs regardless of windowing systems much easier.

**Block diagram of OpenGL**

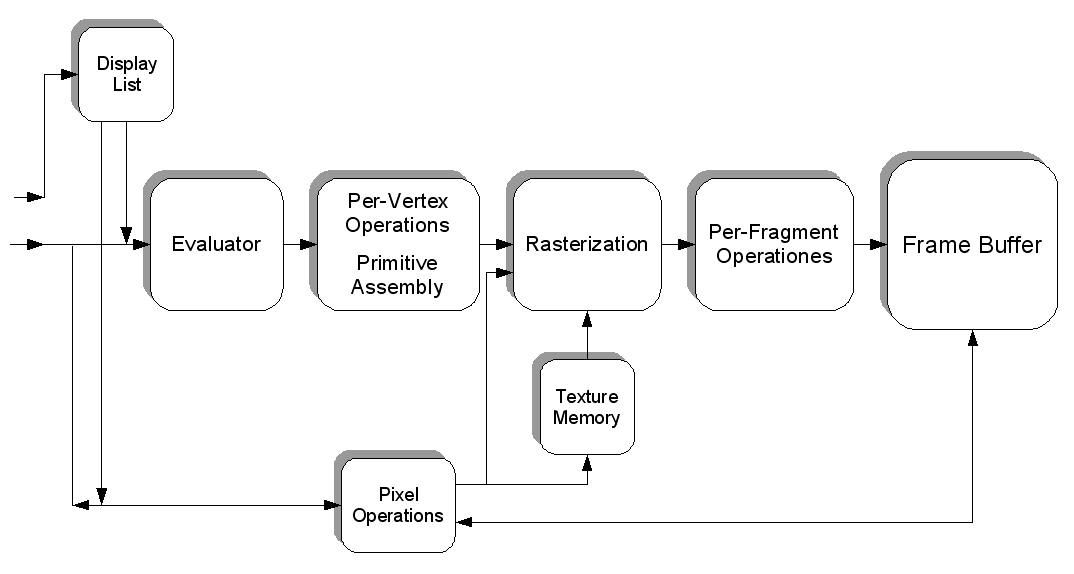


Figure 1: Graphics pipeline

To obtain an image, first we need to process the geometry of our object, for this we can employ the block diagram in figure1. This above figure is known as the graphics pipeline architecture.

**PROBLEM STATEMENT**

**Aim**

To display the Olympic logo in a fanciful way.

**Olympic logo overview**

The Olympic symbols are icons, [flags](http://en.wikipedia.org/wiki/Flag) and symbols used by the [International Olympic Committee](http://en.wikipedia.org/wiki/International_Olympic_Committee) to promote the [Olympic Games](http://en.wikipedia.org/wiki/Olympic_Games). Some—such as the flame, fanfare, and theme—are more common during Olympic competition, but others, such as the flag, can be seen throughout the year.

The symbol/logo of the Olympic Games is composed of five interlocking rings, colored blue, yellow, black, green, and red on a white field. This was originally designed in 1912 by Baron [Pierre de Coubertin](http://en.wikipedia.org/wiki/Pierre_de_Coubertin), the founder of the modern Olympic Games. These five rings represent the five continents of the world- Asia, Africa, America, Australia, and Europe. Furthermore, each color ring represents one continent like black ring represents Africa and so on.

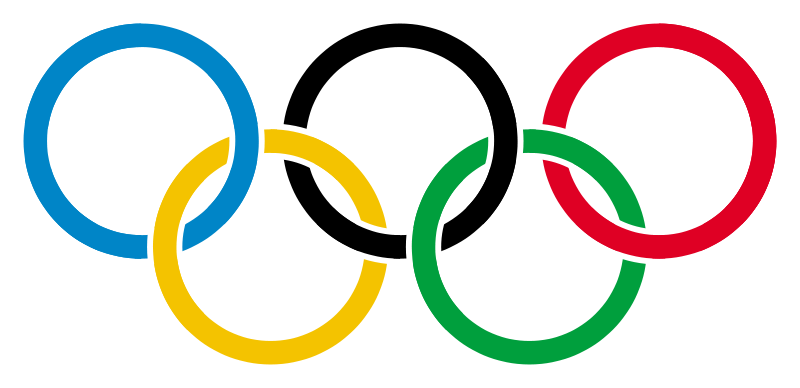


Figure 2: Olympic Logo

**Project Description:**

The Mini Project **locomotive Olympic logo** demonstrates the use of various OpenGL functions and its applications to model the requirements of an application programmer. It also illustrates the use of mathematical functions to control the execution of the project that is presented.

It highlights the key features of the data structures and its high quality efficiency that is obtained on its usage in the application program. This project consists of Olympic logo which is constructed by using different primitives available in OpenGL library and combining them together in a required manner. It illustrates the role of different callback functions that provides easier way to accomplish our project in an effective manner.

**Scope**

This project would be helpful in websites which hosts information related to Olympics; in a manner to give a creative home page for that website. Or it can also be used as a part of the game which wants to display the Olympic symbol in a creative manner.

**DESIGN AND IMPLEMENTATION**

**SOURCE CODE**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <GL/glut.h>

/\* Some <math.h> files do not define M\_PI... \*/

#ifndef M\_PI

#define M\_PI 3.141592654

#endif

#ifdef WIN32

#define drand48() (((float) rand())/((float) RAND\_MAX))

#define srand48(x) (srand((x)))

#else

extern double drand48(void);

extern void srand48(long seedval);

#endif

#define XSIZE 100

#define YSIZE 75

#define RINGS 5

#define BLUERING 0

#define BLACKRING 1

#define REDRING 2

#define YELLOWRING 3

#define GREENRING 4

#define BACKGROUND 8

enum

{

BLACK = 0,

RED,

GREEN,

YELLOW,

BLUE,

MAGENTA,

CYAN,

WHITE

};

typedef short Point[2];

GLenum rgb, doubleBuffer, directRender;

int SPEEDLIMIT=4000;

unsigned char rgb\_colors[RINGS][3];

int mapped\_colors[RINGS];

float dests[RINGS][3];

float offsets[RINGS][3];

float angs[RINGS];

float rotAxis[RINGS][3];

int iters[RINGS];

GLuint theTorus;

void FillTorus(float rc, int numc, float rt, int numt)

{

int i, j, k;

double s, t;

double x, y, z;

double pi, twopi;

pi = M\_PI;

twopi = 2 \* pi;

for (i = 0; i < numc; i++)

{

glBegin(GL\_QUAD\_STRIP);

for (j = 0; j <= numt; j++)

{

for (k = 1; k >= 0; k--)

{

s = (i + k) % numc + 0.5;

t = j % numt;

x = cos(t \* twopi / numt) \* cos(s \* twopi / numc);

y = sin(t \* twopi / numt) \* cos(s \* twopi / numc);

z = sin(s \* twopi / numc);

glNormal3f(x, y, z);

x = (rt + rc \* cos(s \* twopi / numc)) \* cos(t \* twopi / numt);

y = (rt + rc \* cos(s \* twopi / numc)) \* sin(t \* twopi / numt);

z = rc \* sin(s \* twopi / numc);

glVertex3f(x, y, z);

}

}

glEnd();

}

}

float Clamp(int iters\_left, float t)

{

if (iters\_left < 3)

return 0.0;

return (iters\_left - 2) \* t / iters\_left;

}

void Idle(void)

{

int i, j;

int more = GL\_FALSE;

for (i = 0; i < RINGS; i++)

{

if (iters[i])

{

for (j = 0; j < 3; j++)

offsets[i][j] = Clamp(iters[i], offsets[i][j]);

angs[i] = Clamp(iters[i], angs[i]);

iters[i]--;

more = GL\_TRUE;

}

}

if (more)

glutPostRedisplay();

else

glutIdleFunc(NULL);

}

void DrawScene(void)

{

int i;

glPushMatrix();

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

gluLookAt(0, 0, 10, 0, 0, 0, 0, 1, 0);

for (i = 0; i < RINGS; i++)

{

if (rgb)

glColor3ubv(rgb\_colors[i]);

else

glIndexi(mapped\_colors[i]);

glPushMatrix();

glTranslatef(dests[i][0] + offsets[i][0], dests[i][1] + offsets[i][1],

dests[i][2] + offsets[i][2]);

glRotatef(angs[i], rotAxis[i][0], rotAxis[i][1], rotAxis[i][2]);

glCallList(theTorus);

glPopMatrix();

}

glPopMatrix();

if (doubleBuffer)

glutSwapBuffers();

else

glFlush();

}

float MyRand(void)

{

return 10.0 \* (drand48() - 0.5);

}

void ReInit(void)

{

int i;

float deviation;

deviation = MyRand() / 2;

deviation = deviation \* deviation;

for (i = 0; i < RINGS; i++)

{

offsets[i][0] = MyRand();

offsets[i][1] = MyRand();

offsets[i][2] = MyRand();

angs[i] = 260.0 \* MyRand();

rotAxis[i][0] = MyRand();

rotAxis[i][1] = MyRand();

rotAxis[i][2] = MyRand();

iters[i] = (deviation \* MyRand() + 60.0);

}

}

void Init(void)

{

int i;

float top\_y = 1.0;

float bottom\_y = 0.0;

float top\_z = 0.15;

float bottom\_z = 0.69;

float spacing = 2.5;

static float lmodel\_ambient[] = {0.0, 0.0, 0.0, 0.0};

static float lmodel\_twoside[] = {GL\_FALSE};

static float lmodel\_local[] = {GL\_FALSE};

static float light0\_ambient[] = {0.1, 0.1, 0.1, 1.0};

static float light0\_diffuse[] = {1.0, 1.0, 1.0, 0.0};

static float light0\_position[] = {0.8660254, 0.5, 1, 0};

static float light0\_specular[] = {1.0, 1.0, 1.0, 0.0};

static float bevel\_mat\_ambient[] = {0.0, 0.0, 0.0, 1.0};

static float bevel\_mat\_shininess[] = {40.0};

static float bevel\_mat\_specular[] = {1.0, 1.0, 1.0, 0.0};

static float bevel\_mat\_diffuse[] = {1.0, 0.0, 0.0, 0.0};

srand48(0x102342);

ReInit();

for (i = 0; i < RINGS; i++)

rgb\_colors[i][0] = rgb\_colors[i][1] = rgb\_colors[i][2] = 0;

rgb\_colors[BLUERING][2] = 255;

rgb\_colors[REDRING][0] = 255;

rgb\_colors[GREENRING][1] = 255;

rgb\_colors[YELLOWRING][0] = 255;

rgb\_colors[YELLOWRING][1] = 255;

mapped\_colors[BLUERING] = BLUE;

mapped\_colors[REDRING] = RED;

mapped\_colors[GREENRING] = GREEN;

mapped\_colors[YELLOWRING] = YELLOW;

mapped\_colors[BLACKRING] = BLACK;

dests[BLUERING][0] = -spacing;

dests[BLUERING][1] = top\_y;

dests[BLUERING][2] = top\_z;

dests[BLACKRING][0] = 0.0;

dests[BLACKRING][1] = top\_y;

dests[BLACKRING][2] = top\_z;

dests[REDRING][0] = spacing;

dests[REDRING][1] = top\_y;

dests[REDRING][2] = top\_z;

dests[YELLOWRING][0] = -spacing / 2.0;

dests[YELLOWRING][1] = bottom\_y;

dests[YELLOWRING][2] = bottom\_z;

dests[GREENRING][0] = spacing / 2.0;

dests[GREENRING][1] = bottom\_y;

dests[GREENRING][2] = bottom\_z;

theTorus = glGenLists(1);

glNewList(theTorus, GL\_COMPILE);

FillTorus(0.1, 8, 1.0, 25);

glEndList();

glEnable(GL\_CULL\_FACE);

glCullFace(GL\_BACK);

glEnable(GL\_DEPTH\_TEST);

glClearDepth(1.0);

if (rgb)

{

glClearColor(0.5, 0.5, 0.5, 0.0);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light0\_ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light0\_diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light0\_specular);

glLightfv(GL\_LIGHT0, GL\_POSITION, light0\_position);

glEnable(GL\_LIGHT0);

glLightModelfv(GL\_LIGHT\_MODEL\_LOCAL\_VIEWER, lmodel\_local);

glLightModelfv(GL\_LIGHT\_MODEL\_TWO\_SIDE, lmodel\_twoside);

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, lmodel\_ambient);

glEnable(GL\_LIGHTING);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, bevel\_mat\_ambient);

glMaterialfv(GL\_FRONT, GL\_SHININESS, bevel\_mat\_shininess);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, bevel\_mat\_specular);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, bevel\_mat\_diffuse);

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE);

glEnable(GL\_COLOR\_MATERIAL);

glShadeModel(GL\_SMOOTH);

}

else

{

glClearIndex(BACKGROUND);

glShadeModel(GL\_FLAT);

}

glMatrixMode(GL\_PROJECTION);

gluPerspective(45, 1.33, 0.1, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

void Reshape(int width, int height)

{

glViewport(0, 0, width, height);

}

/\* ARGSUSED1 \*/

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

{

switch (key)

{

case 27 : exit(0);

break;

case ' ' : ReInit();

glutIdleFunc(Idle);

break;

}

}

GLenum Args(int argc, char \*\*argv)

{

GLint i;

rgb = GL\_TRUE;

doubleBuffer = GL\_TRUE;

for (i = 1; i < argc; i++)

{

if (strcmp(argv[i], "-ci") == 0)

{

rgb = GL\_FALSE;

}

else if (strcmp(argv[i], "-rgb") == 0)

{

rgb = GL\_TRUE;

}

else if (strcmp(argv[i], "-sb") == 0)

{

doubleBuffer = GL\_FALSE;

}

else if (strcmp(argv[i], "-db") == 0)

{

doubleBuffer = GL\_TRUE;

}

else

{

printf ("%s (Bad option).\n", argv[i]);

return GL\_FALSE;

}

}

return GL\_TRUE;

}

void visible(int vis)

{

if (vis == GLUT\_VISIBLE)

{

glutIdleFunc(Idle);

}

else

{

glutIdleFunc(NULL);

}

}

void speed\_menu(int id)

{

SPEEDLIMIT=4000;

switch(id)

{

case 1: SPEEDLIMIT =SPEEDLIMIT\*2;

ReInit();

glutIdleFunc(Idle);

break;

case 2: SPEEDLIMIT =SPEEDLIMIT\*1.5;

ReInit();

glutIdleFunc(Idle);

break;

case 3: SPEEDLIMIT =SPEEDLIMIT;

ReInit();

glutIdleFunc(Idle);

break;

case 4: SPEEDLIMIT =SPEEDLIMIT/2;

ReInit();

glutIdleFunc(Idle);

break;

case 5: SPEEDLIMIT =SPEEDLIMIT/1000;

ReInit();

glutIdleFunc(Idle);

break;

}

glutPostRedisplay();

}

void olympic\_menu(int id)

{

switch(id)

{

case 1: ReInit();

break;

case 2: exit(0);

break;

}

}

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

{

GLenum type;

glutInitWindowSize(400, 300);

glutInit(&argc, argv);

if (Args(argc, argv) == GL\_FALSE)

{

exit(1);

}

type = (rgb) ? GLUT\_RGB : GLUT\_INDEX;

type |= (doubleBuffer) ? GLUT\_DOUBLE : GLUT\_SINGLE;

glutInitDisplayMode(type);

glutCreateWindow("Olympic Logo");

Init();

glutReshapeFunc(Reshape);

glutKeyboardFunc(Key);

glutDisplayFunc(DrawScene);

sub=glutCreateMenu(speed\_menu);

glutAddMenuEntry("Very slow",1);

glutAddMenuEntry("Slow",2);

glutAddMenuEntry("Normal",3);

glutAddMenuEntry("Fast",4);

glutAddMenuEntry("Very Fast",5);

glutCreateMenu(olympic\_menu);

glutAddSubMenu("Re-initialise",sub);

glutAddMenuEntry("Random Position",1);

glutAddMenuEntry("Quit",2);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutVisibilityFunc(visible);

glutMainLoop();

return 0; /\* ANSI C requires main to return int. \*/

}

**Description:**

The above function controls the entire execution of the program. The execution starts from the main function. The following steps show the flow of execution. Initialization of OpenGL and Window System.

* Initialization of Display window.
* Display callback function is executed
* Keyboard callback function is executed.
* Visibility callback function is executed.
* Menus are created and the entries are attached to it.
* View volume is adjusted.
* Initially many stars are created.
* The Event Loop continues to execute infinitely until the exit button is pressed.

**RESULT ANALYSIS**

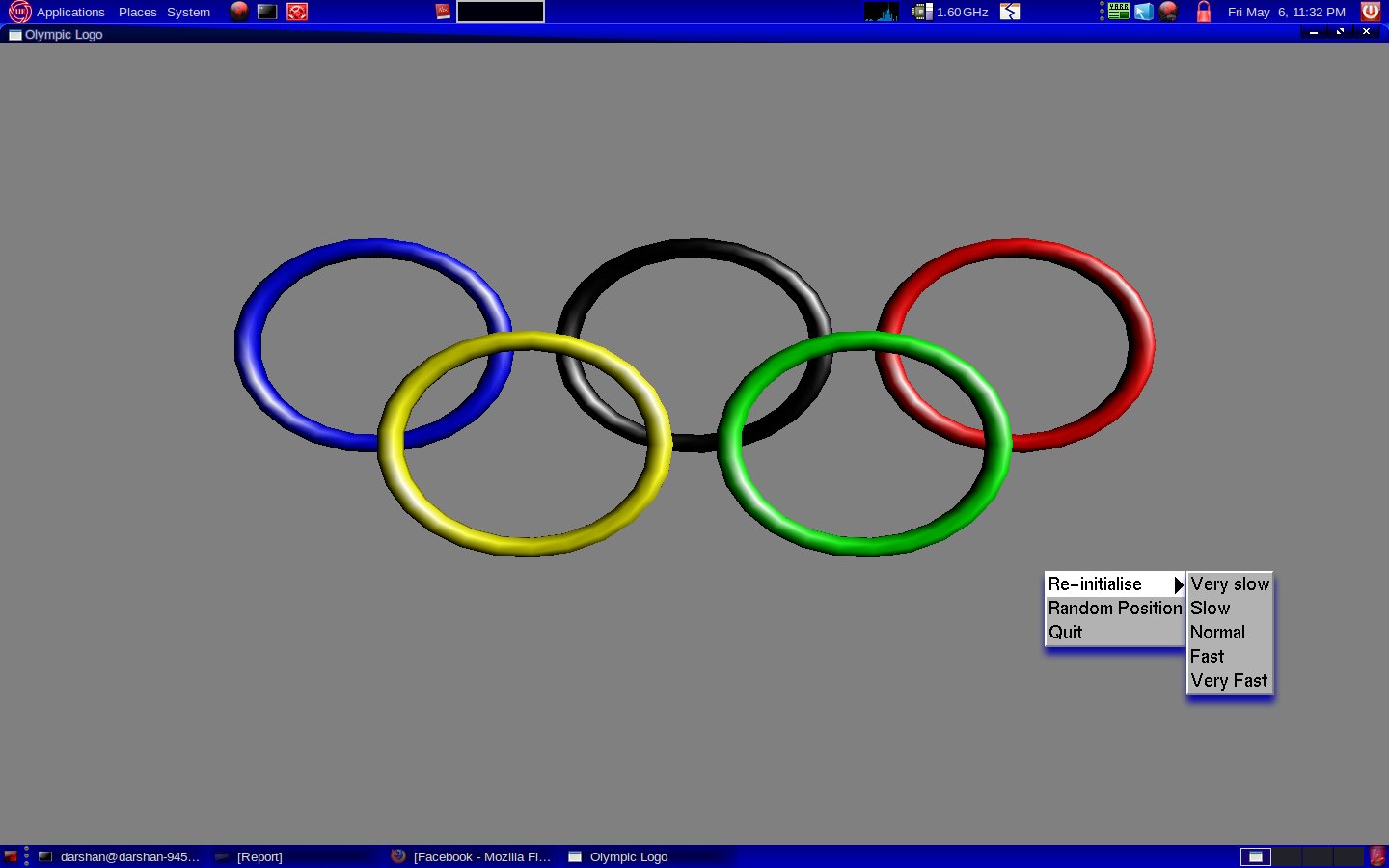


Figure 5.1: Olympic Logo along with the menu list

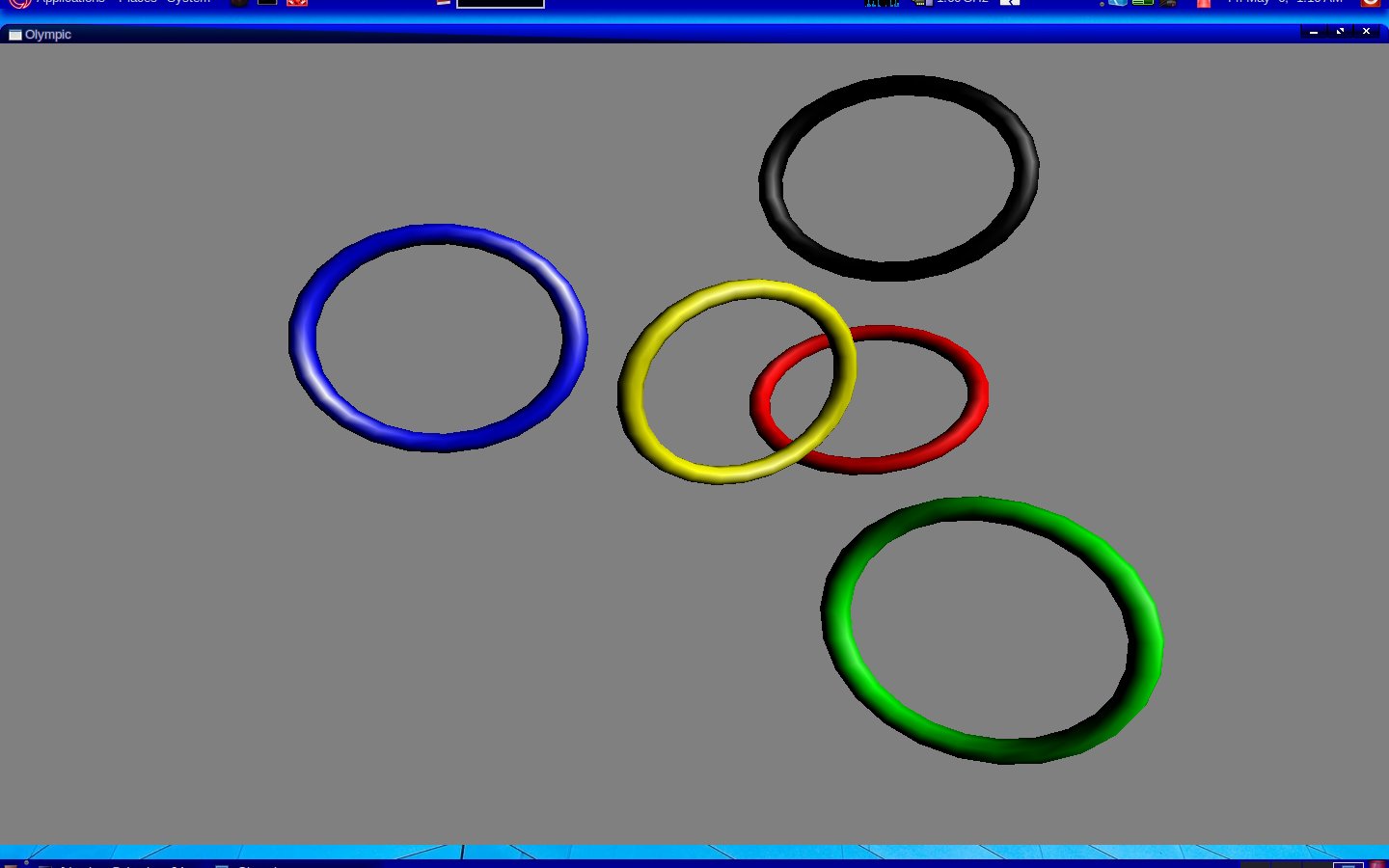


Figure 5.2: The rings in the Olympic logo at random positions

The above two figures are the snapshots of our program’s output. Figure 5.1, shows the output when the program is executed and a right mouse click is made. When a right mouse click is made a menu is popped up at the place where the mouse click happened; as shown in the fig 5.1. This menu list is a hierarchical menu, because the first menu entry contains sub-menu; which in turn contains 5 entries-

* Re-initialize
* Very Slow
* Slow
* Normal
* Very Fast
* Fast
* Random position
* Quit

When the cursor is placed on the “re-initialize” entry, sub-menu will be popped-up which contains further entries as mentioned above.

When “Very Slow” entry is clicked, the Olympic logo is redisplayed such that the motion of the rings from different positions will be very slow.

When “Slow” entry is clicked, the Olympic logo is redisplayed such that the motion of the rings from different positions will be slow.

When “Normal” entry is clicked, the Olympic logo is redisplayed such that the motion of the rings from different positions will be at normal speed.

When “Fast” entry is clicked, the Olympic logo is redisplayed such that the motion of the rings from different positions will be fast.

When “Very Fast” entry is clicked, the Olympic logo is redisplayed such that the motion of the rings from different positions will be very fast.

When the “random position” entry is clicked, the rings of the Olympic logo will be displayed at random positions on the screen as shown in fig 5.2.

When the “Quit” entry is clicked, the display window will get closed and program terminates.

**CONCLUSION**

**Conclusion**

The conclusion here - is that the application of this project. I.e., where all this project’s result can be used. This project would be helpful in websites which hosts information related to Olympics; in a manner to give a graphical home page for that website. Or it can also be used as a part of the game which wants to display the Olympic symbol with graphics effect.

**REFERENCES**

* Interactive Computer Graphics A Top-Down Approach with OpenGL, Edward Angel, Addison-Wesley, 5th Edition, 2008 - **BOOK**
* <http://en.wikipedia.org/wiki/Computer_graphics> - **LINK**
* <http://en.wikipedia.org/wiki/Opengl> - **LINK**
* <http://en.wikipedia.org/wiki/Olympic_symbols> - **LINK**