Chapter 1

Introduction

1.1 Computer Graphics

• Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly.

• Computers have become a powerful medium for the rapid and economical production of pictures.

• Graphics provide a so natural means of communicating with the computer that they have become widespread.

• Interactive graphics is the most important means of producing pictures since the invention of photography and television.

• We can make pictures of not only the real world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry. 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.

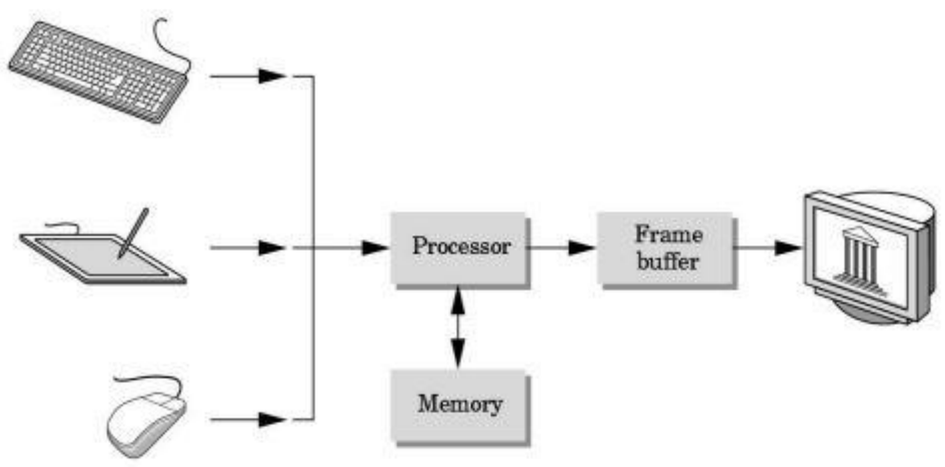


Fig 1.1: Graphic System

1.2 Areas of Application of Computer Graphics

- ➢ User interfaces and Process control
- ➢ Cartography
- ➢ Office automation and Desktop publishing
- ➢ Plotting of graphs and charts
- ➢ Computer aided Drafting and designs
- ➢ Simulation and Animation

1.3 Introduction to OpenGL

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms.

OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

OpenGL available Everywhere: Supported on all UNIX® workstations, and shipped standard with every Windows 95/98/2000/NTand MacOS PC, no other graphics API operates on a wider range of hardware platforms and software environments.

OpenGL runs on every major operating system including Mac OS, OS/2, UNIX, Windows 95/98, Windows 2000, Windows NT, Linux, Open Step, and BeOS; it also works with every major windowing system, including Win32, MacOS, Presentation Manager, and X-Window System. OpenGL is callable from Ada, C, C++, Fortran, Python, Perl and Java and offers complete independence from network protocols and topologies.

1.3.1 The OpenGL interface

Our application will be designed to access OpenGL directly through functions in three libraries namely: gl, glu, and glut.

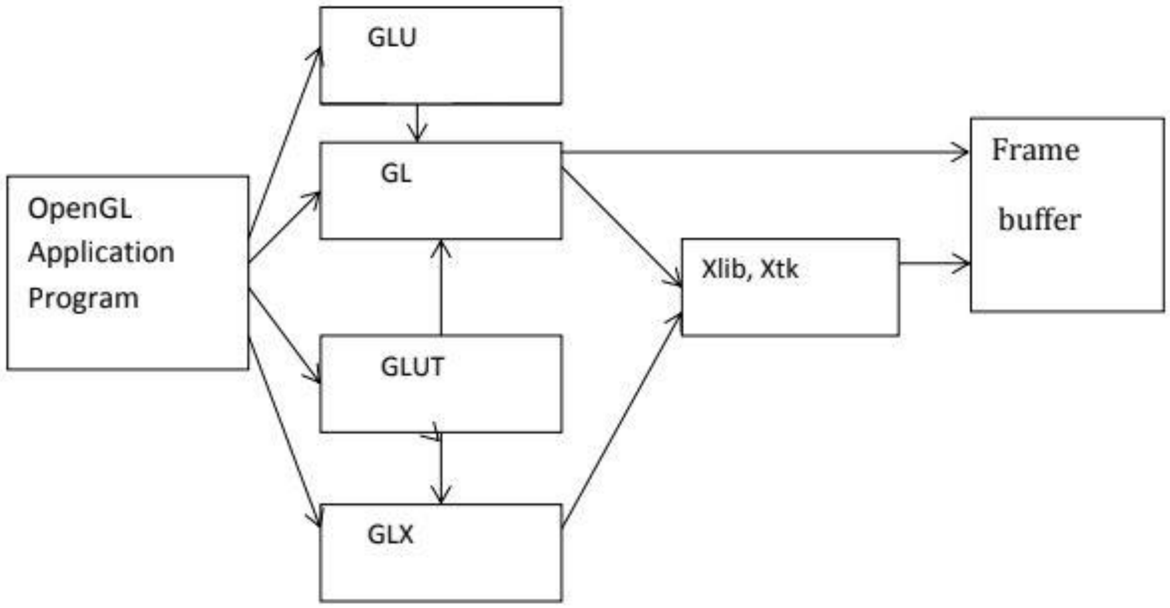


Fig 1.2: Library organization of OpenGL

1.3.2 Graphics Functions

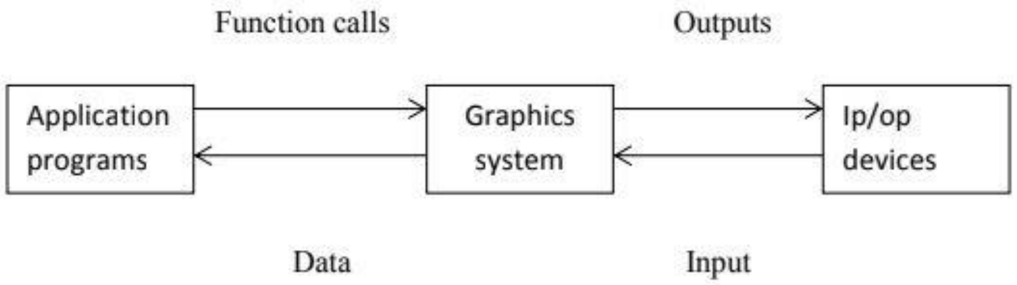


Fig 1.3: Graphics system as a black box.

Our basic model of a graphics package is a black box, a term that engineers use to denote a system whose properties are described only by its inputs and outputs. We describe an API through the functions in its library. Some of the functions are:

➢ The primitive functions define the low-level objects or atomic entities that our system can display.

➢ Attribute functions allow us to perform operations ranging from choosing the color with which we display a line segment, to picking a pattern with which to fill the inside of a polygon, to selecting a typeface for the titles of a graph.

➢ Transformation function allows carrying out transformations of objects, such as rotation, translation, and scaling.

➢ A set of input functions allow us to deal with the diverse forms of input that characterize modern graphics systems.

➢ The control functions enable us to communicate with the window systems, to initialize our programs, and to deal with any errors that take place during the execution of programs.

User mode –

When the computer system run user application like creating a text document or using any application program, then the system is in user mode. When the user application requests for a service from the operating system or an interrupt occurs or system call, then there will be a transition from user to kernel mode to fulfill the requests.

Note: To switch from kernel mode to user mode, mode bit should be 1.

Kernel Mode –

When system boots then hardware starts in kernel mode and when operating system is loaded then it starts user application in user mode. To provide protection to the hardware, we have privileged instructions which execute only in kernel mode. If user attempt to run privileged instruction in user mode, then it will treat instruction as illegal and traps to OS.

Note: To switch from user mode to kernel mode, mode bit should be 0.

Chapter 2

System Requirement

2.2.1 User Requirement:

• Easy to understand and should be simple.

• The built-in functions should be utilized to maximum extent.

• OpenGL library facilities should be used.

2.2.2 Software Requirements:

• Platform used: UBUNTU

• Technology used: OpenGL Libraries such has OpenGL Utility library, OpenGL

Utility toolkit

• Language: C

2.2.3 Hardware Requirements:

• Processor-Intel or AMD (Advanced Micro Devices)

• RAM-512MB (minimum)

• Hard Disk-1MB (minimum)

• Mouse

• Keyboard

• Monitor

Chapter 3

Design

3.1 How it works ?

- Ganesh keeps moving towards the right.
- There are two points for which if the frames are matched the game is ended and the player loose.
- The points are :

```
//x->367 y->186
        //i->292 j->26
        if(j<=26&&i==292)
                loose=1;

        //x->758 y->190
        //i->683 j->30
        if(j<=30&&i==683)
                loose=1;
```

- If jumped at the point then he can proceed
- At the end there is a point where if he reaches it is the end of the game and the player wins.
- The point is :
    - if(i>1100)
    - win=1;

Chapter 4

Discussion & Screenshots

Program Code:

```
#include<stdio.h>

#include<GL/glut.h>

#include <GL/gl.h>

#include <stdlib.h>

#include<string.h>


float i=0.0;    //movement of ganesh

float j=10.0;    //movement of ganesh

float m=0.0;    //movement of clouds

int jump=0,flag=0,count=0;

int win=0,loose=0;


void draw_pixel(GLint cx, GLint cy)

{

        glBegin(GL_POINTS);

                glVertex2i(cx,cy);

        glEnd();
```

```
}


void plotpixels(GLint h,GLint k, GLint x,GLint y)
{
        draw_pixel(x+h,y+k);
        draw_pixel(-x+h,y+k);
        draw_pixel(x+h,-y+k);
        draw_pixel(-x+h,-y+k);
        draw_pixel(y+h,x+k);
        draw_pixel(-y+h,x+k);
        draw_pixel(y+h,-x+k);
        draw_pixel(-y+h,-x+k);
}
void draw_object()
{
        int l;


        //sky
        glColor3f(0.0,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(0,160);
        glVertex2f(0,700);
```

```
glVertex2f(1100,700);

glVertex2f(1100,160);

glEnd();


//moon


        for(l=0;l<=35;l++)

        {

                glColor3f(1.0,1.0,1.0);

                draw_circle(100,625,l);

        }




//cloud1


        for(l=0;l<=20;l++)

        {

                glColor3f(1.0,1.0,1.0);

                draw_circle(160+m,625,l);


        }

for(l=0;l<=20;l++)
```

```
                {
                        glColor3f(1.0,1.0,1.0);
                        draw_circle(500+m,615,l);
                }

        //star1

        glColor3f(1.0,1.0,1.0);
        glBegin(GL_TRIANGLES);
        glVertex2f(575,653);
        glVertex2f(570,645);
        glVertex2f(580,645);
        glVertex2f(575,642);
        glVertex2f(570,650);
        glVertex2f(580,650);
        glEnd();
   //ganesh
        for(l=0;l<20;l++)
          {
                glColor3f(1.0,0.5,0.0);
                draw_circle(82+i,260+j,l);
          }
```

```
glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(65+i,180+j);
glVertex2f(65+i,240+j);
glVertex2f(100+i,240+j);
glVertex2f(100+i,180+j);
glEnd();
//right arm
glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(100+i,230+j);
glVertex2f(110+i,250+j);
glVertex2f(115+i,250+j);
glVertex2f(100+i,230+j);
glEnd();
//left arm
glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(65+i,230+j);
glVertex2f(55+i,250+j);
glVertex2f(60+i,250+j);
glVertex2f(65+i,230+j);
```

```
        glEnd();
        //left leg
        glColor3f(0.0,0.5,1.0);
        glBegin(GL_POLYGON);
        glVertex2f(50+i,160+j);
        glVertex2f(60+i,180+j);
        glVertex2f(65+i,180+j);
        glVertex2f(55+i,160+j);
        glEnd();
        //right leg
        glColor3f(0.0,0.5,1.0);
        glBegin(GL_POLYGON);
        glVertex2f(110+i,160+j);
        glVertex2f(100+i,180+j);
        glVertex2f(105+i,180+j);
        glVertex2f(115+i,160+j);
        glEnd();
        glFlush();
}
void idle()
{
if(jump==1&&j<200&&flag==0)
```

```
{
        j=j+0.5;
        if(j==200)
        {
                flag=1;
                count=50;
        }
}
if(flag==1)
{
        count--;
        if(count<=0)
        {
                if(j==200)
                        jump=0;
                if(j>0)
                        j--;
                if(j==0)
                        flag=0;
        }
}
```

```
        i+=0.25;
        ++m;

        if(i>1100)
                win=1;
        if(m>1100)
                m=0.0;

        glutPostRedisplay();

}
}
void display()
{

        glClear(GL_COLOR_BUFFER_BIT);
        draw_object();
        glFlush();
}
int main(int argc,char** argv)
{
    int c_menu;
        printf("Press 'r' or 'R' to jump\n");
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
        glutInitWindowSize(1100.0,700.0);
        glutInitWindowPosition(0,0);
        glutCreateWindow("Super ganesh");
        glutDisplayFunc(display);
        glutIdleFunc(idle);
        glutKeyboardFunc(keyboardFunc);
        myinit();
        glutMainLoop();
        return 0;
}
```

Screen Shots:

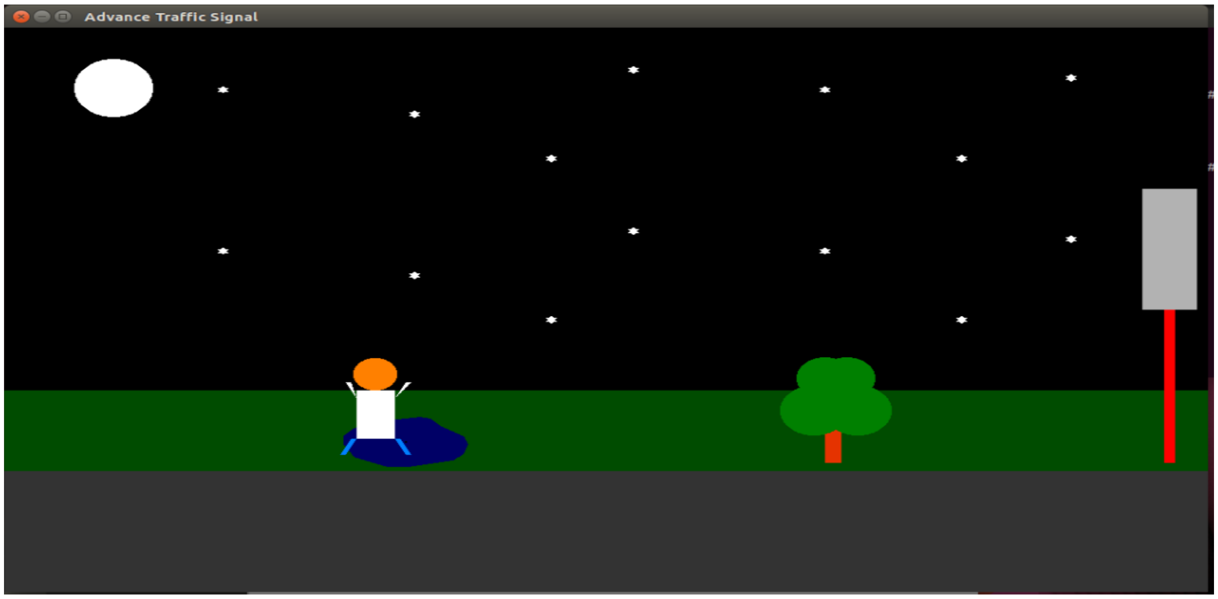
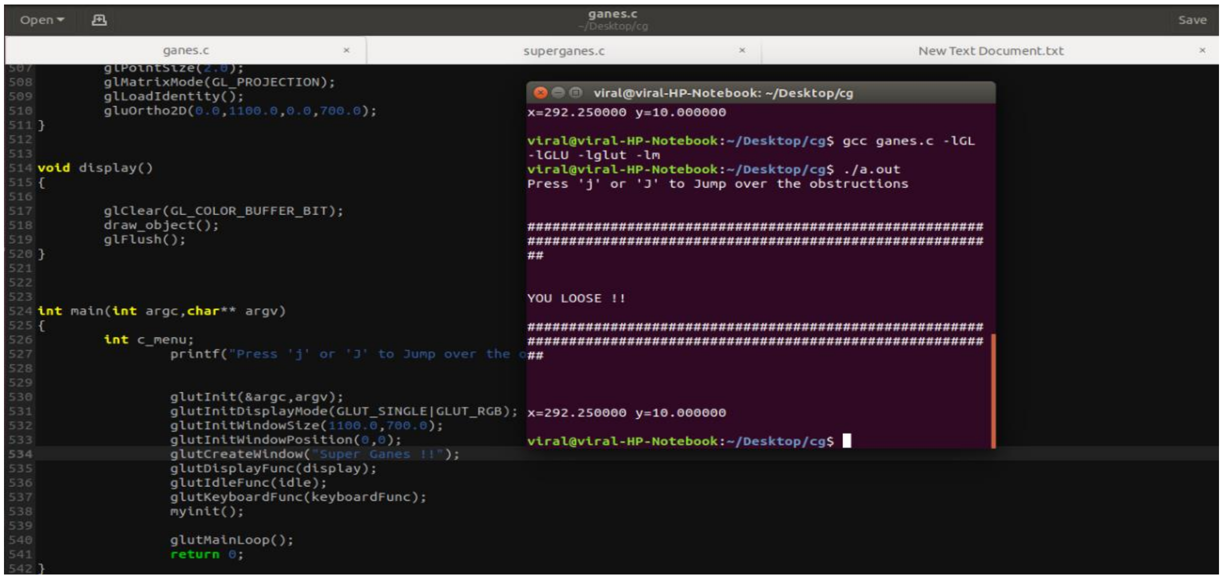


Fig 4.1



```
        glPointSize(2.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0.0,1100.0,0.0,700.0);
}


void display()
{

        glClear(GL_COLOR_BUFFER_BIT);
        draw_object();
        glFlush();
}



int main(int argc,char** argv)
{
        int c_menu;
                printf("Press 'j' or 'J' to Jump over the o


                glutInit(&argc,argv);
                glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
                glutInitWindowSize(1100.0,700.0);
                glutInitWindowPosition(0,0);
                glutCreateWindow("Super Ganes !!");
                glutDisplayFunc(display);
                glutIdleFunc(idle);
                glutKeyboardFunc(keyboardFunc);
                myinit();

                glutMainLoop();
                return 0;
}
```

```
x=292.250000 y=10.000000

viral@viral-HP-Notebook:~/Desktop/cg$ gcc ganes.c -lGL
-lGLU -lglut -lm
viral@viral-HP-Notebook:~/Desktop/cg$ ./a.out
Press 'j' or 'J' to Jump over the obstructions


#######################################################
#######################################################
##


YOU LOOSE !!

#######################################################
#######################################################
##



x=292.250000 y=10.000000

viral@viral-HP-Notebook:~/Desktop/cg$ 
```

Fig 4.2

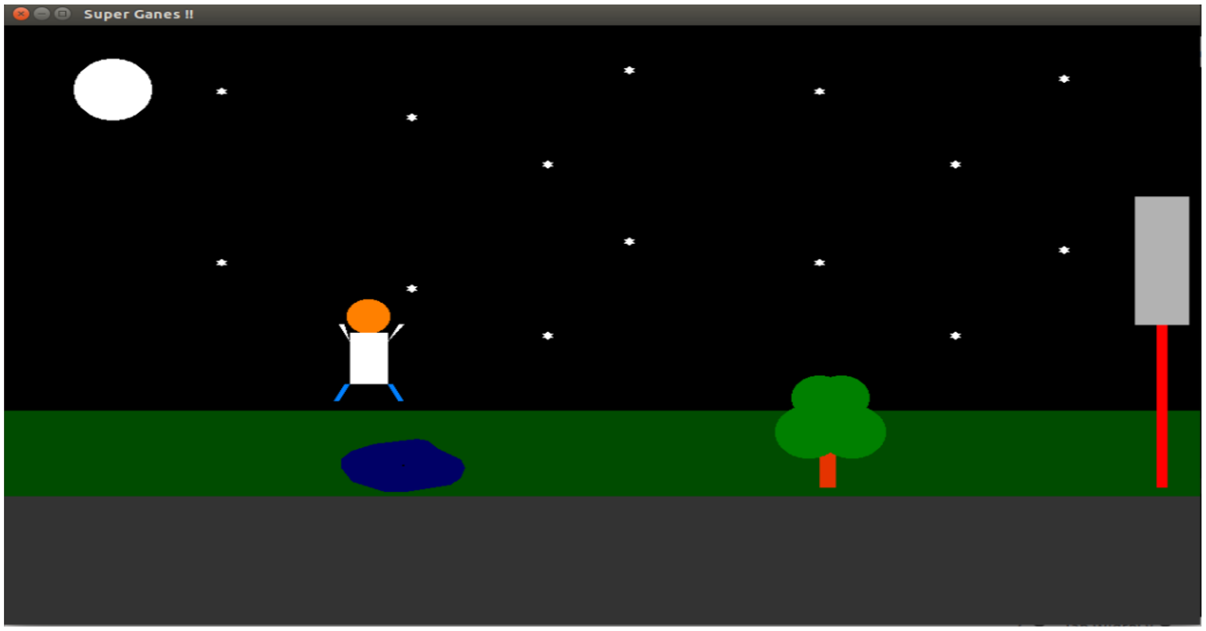


Fig 4.3

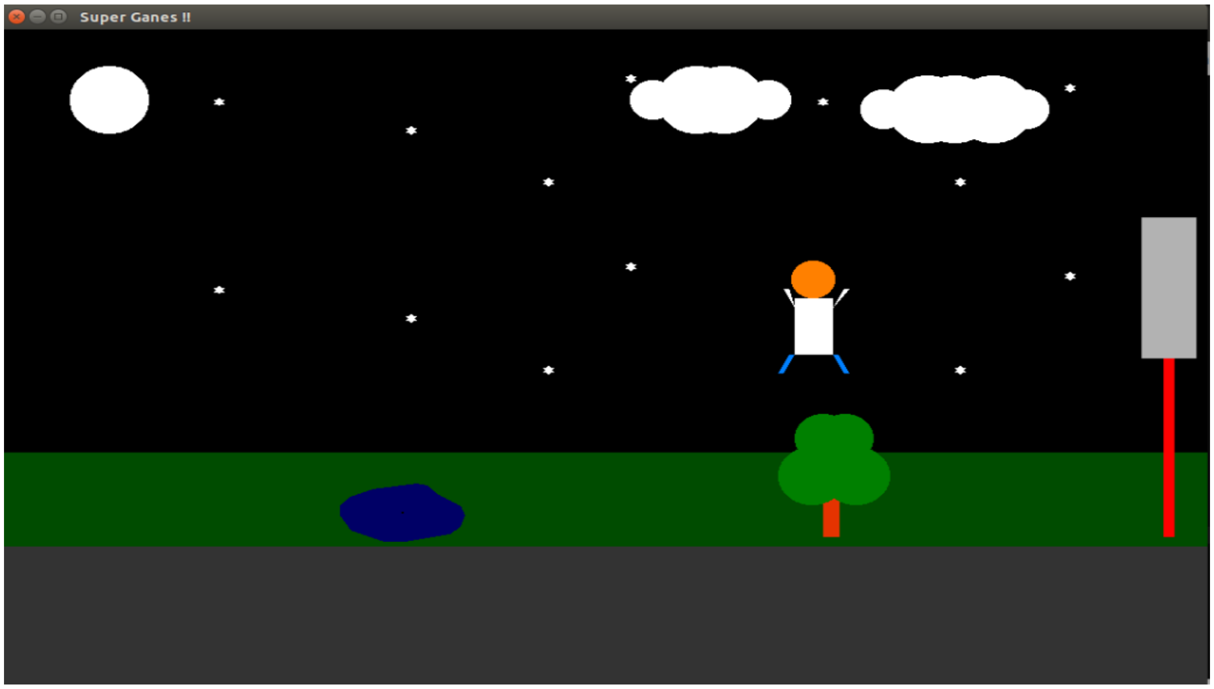


Fig 4.4

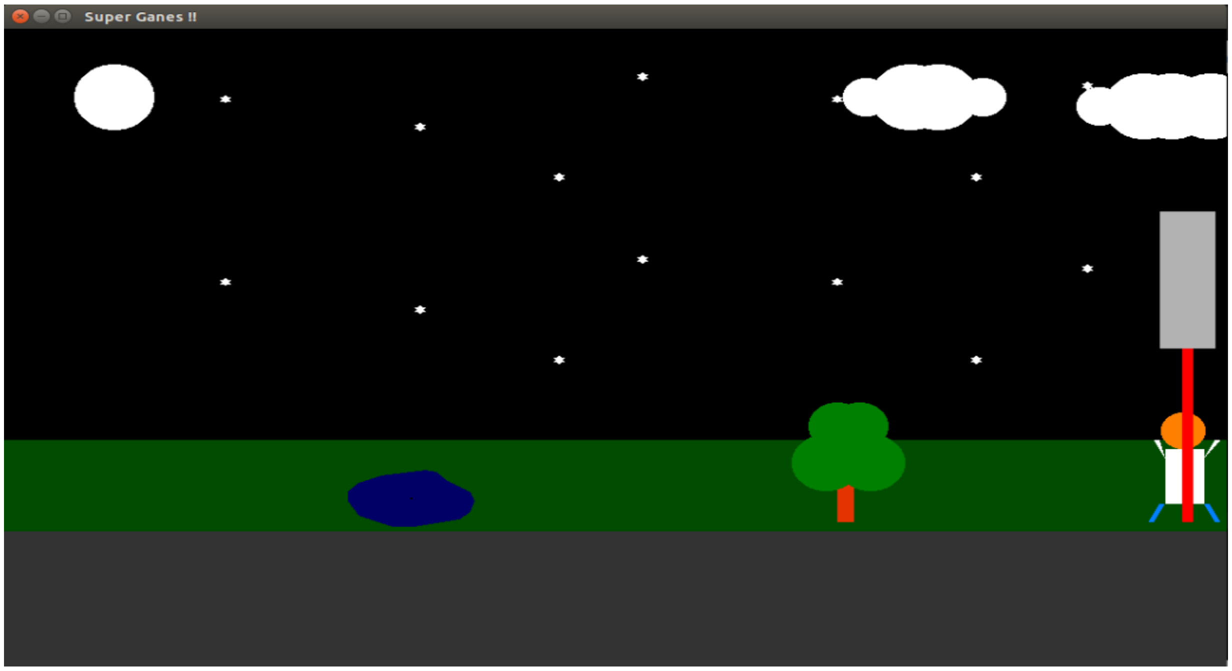
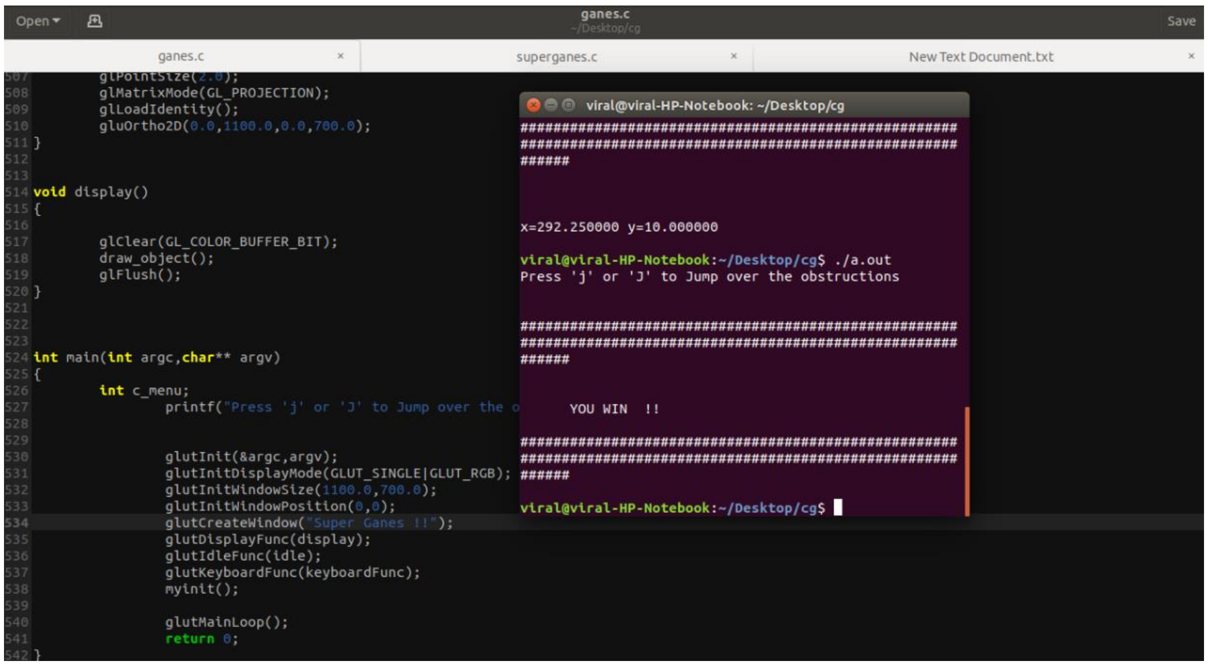


Fig 4.5



```
        glPointSize(2.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0.0,1100.0,0.0,700.0);
}


void display()
{

        glClear(GL_COLOR_BUFFER_BIT);
        draw_object();
        glFlush();
}



int main(int argc,char** argv)
{
        int c_menu;
                printf("Press 'j' or 'J' to Jump over the o


                glutInit(&argc,argv);
                glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
                glutInitWindowSize(1100.0,700.0);
                glutInitWindowPosition(0,0);
                glutCreateWindow("Super Ganes !!");
                glutDisplayFunc(display);
                glutIdleFunc(idle);
                glutKeyboardFunc(keyboardFunc);
                myinit();

                glutMainLoop();
                return 0;
}
```

```
#####################################################
#####################################################
######



x=292.250000 y=10.000000

viral@viral-HP-Notebook:~/Desktop/cg$ ./a.out
Press 'j' or 'J' to Jump over the obstructions


#####################################################
#####################################################
######


   YOU WIN  !!

#####################################################
#####################################################
######

viral@viral-HP-Notebook:~/Desktop/cg$
```

Fig 4.6

Chapter 5

Conclusion and Future Scope

Hence, with this the project report for “Super Ganesh” is completed.

Many more features will be added in future.

We want to make Super Ganesh better by using SDL library and using BMP images.

Bibliography

[1] https://www.geeksforgeeks.org/mario_game_jump/

[2] https://www.computerpoint.com/glut-tutorial/

[3] https://www.computerpoint.com/glu-tutorial/

[4] https://www.computerpoint.com/gl-tutorial/