Chapter 1

INTRODUCTION

1.1. Introduction to Project

My project is a simple screen saver in which a square bounces whenever it strikes the boundary of the screen. It gives a chance to user to be interactive by choosing options in menu by right clicking the mouse button on output screen. User can select background colors and color for square object. The size of object and speed of bouncing can be altered by options provided. We have put extra options to maximize the output screen and quitting by right mouse button only. It takes the starting position from where the square should start .If the user minimizes the screen when the output is executing the program automatically minimizes the square according to the screen size .Various inbuilt functions are used from glut library which reduces the code.

1.2. Introduction to OpenGL

Computer graphics is concerned with all aspects of producing pictures or images using computer. To interact with computer graphics we use a particular graphics software system, **OpenGL,** which has become a widely accepted standard for developing graphics applications. Fortunately, OpenGL is easy to learn, it possess most of the characteristics of other popular graphics systems.

The interface between an application program and a graphics system can be specified through a set of functions that resides in graphics library. These specifications are called the application **programmer's interface (API).** The application programmer sees only the API and is thus shielded from details of both the hardware and the software implementation of graphics library. The software **drivers** are responsible for interpreting the output of the API and converting these data to a form that is understood by the particular hardware.

Although OpenGL is easy to learn, it supports the simple two- and three-dimensional programs. Our basic model of graphics package is a **black box** described only by its inputs and outputs; we may know nothing about its internal working s. We can think of graphics system as a box inputs are function calls from an application program; measurements from input devices, such as mouse and keyboard and possibly other input, such as messages from operating system .The outputs are primarily the graphics sent to our output devices.

A good API contains hundreds of functions , so it is helpful to divide them into seven major groups : Primitive Function ,Attribute functions , Viewing functions ,Transformation functions ,Input functions , Control functions , Query functions .

Most of our applications are designed to access OpenGL directly through functions in three libraries. Functions in the main GL library have names that begin with letters gl and are stored in library usually referred to as GL. The second is the **OpenGL Utility Library(GLU)**. All function in GLU can be created from the core GL library but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations, functions in the GLU library begins with the letters glu.

Chapter 2

SYSTEM REQUIREMENTS

2.1. FUNCTIONAL REQUIREMENTS

Basic functionality of our project is a screen saver with different primitives used. Graphics of the background as well as the square can be changed which gives a very good visual effect.

2.2. HARDWARE REQUIREMENTS

- Intel® Pentium® M processor
- Speed of 2.60 GHz
- 1GB MB of RAM
- 1280 x 768 is the screen resolution

2.3. SOFTWARE REQUIREMENTS

- we use LINUX in UBUNTU as the operating system
- library function used are OpenGL Utility library,OpenGL Utility toolkit
- compiler used is cc

Chapter 3

DESIGN

3.1. DETAILED DESIGN

Design process involves design of algorithms, modules, components and subsystems. The resolution of 1300 x 800 using three-color (RGB) model can initialized using the GL/glut.h library. The implementation can be divided into 3 parts.

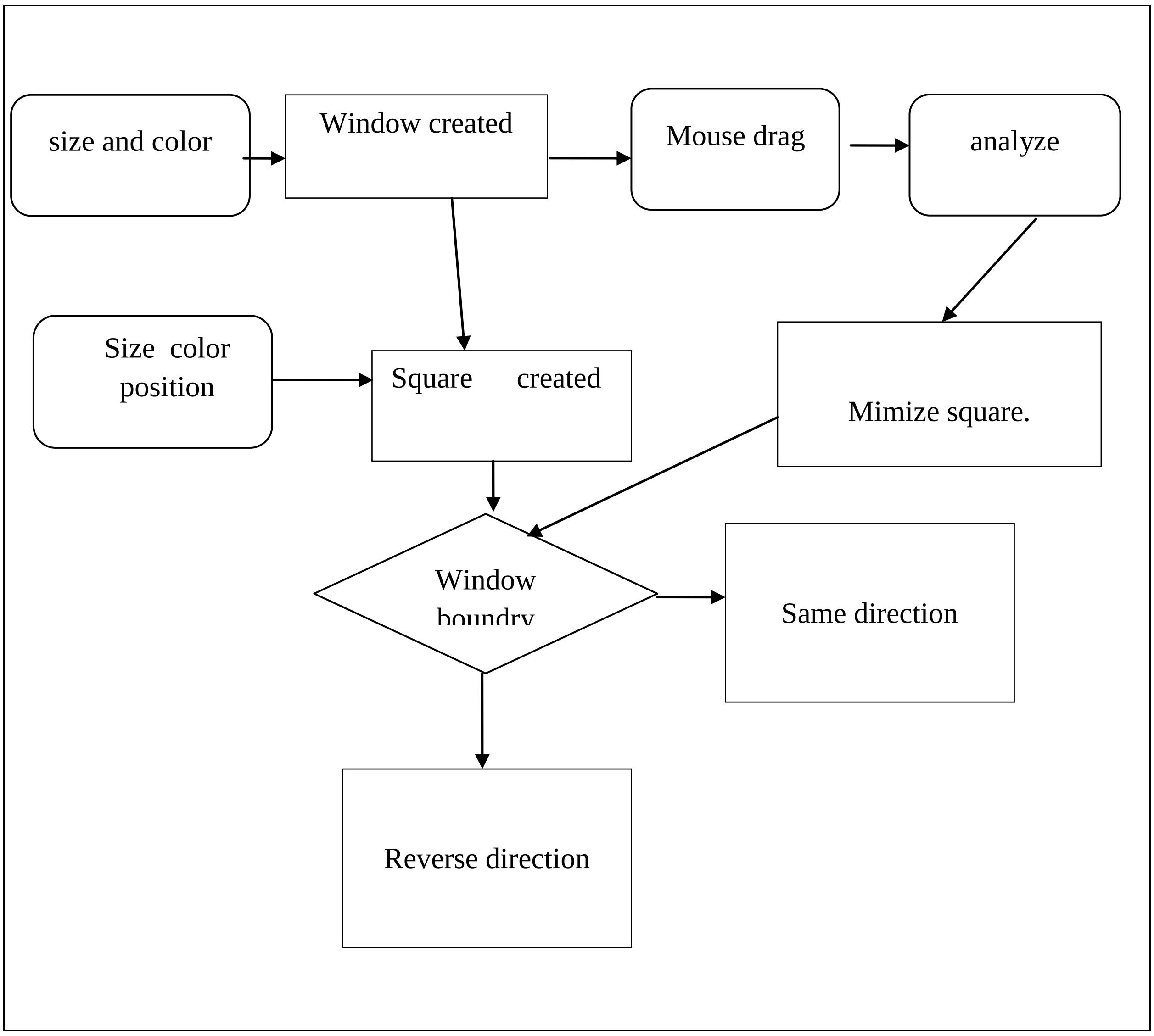
Mouse interface: There is set of actions for mouse interface written using void changesize().

3.2. SYSTEM MODEL

This design stage involves the actual building of the core components of the different modules, which are involved in the graphics editor. The design can be sub-divided as follows

- Design of User Interface
- Design of Square formation
- Design of algorithm for the movement of Square.

3.3. DATA FLOW DIAGRAM

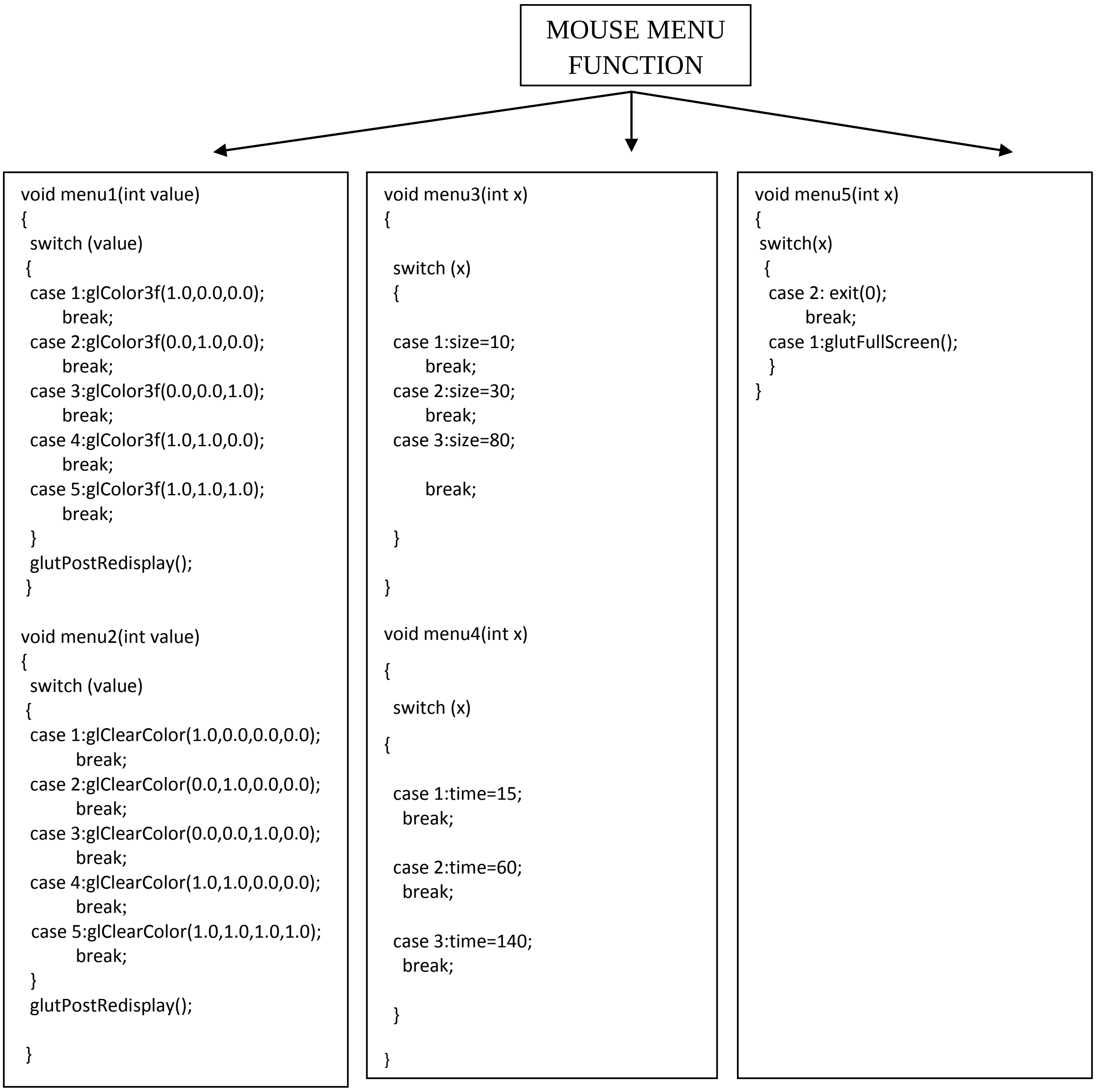


Chapter 4

IMPLEMENTATION

4.1. Description of Function

4.1.1. MOUSE Menu function:

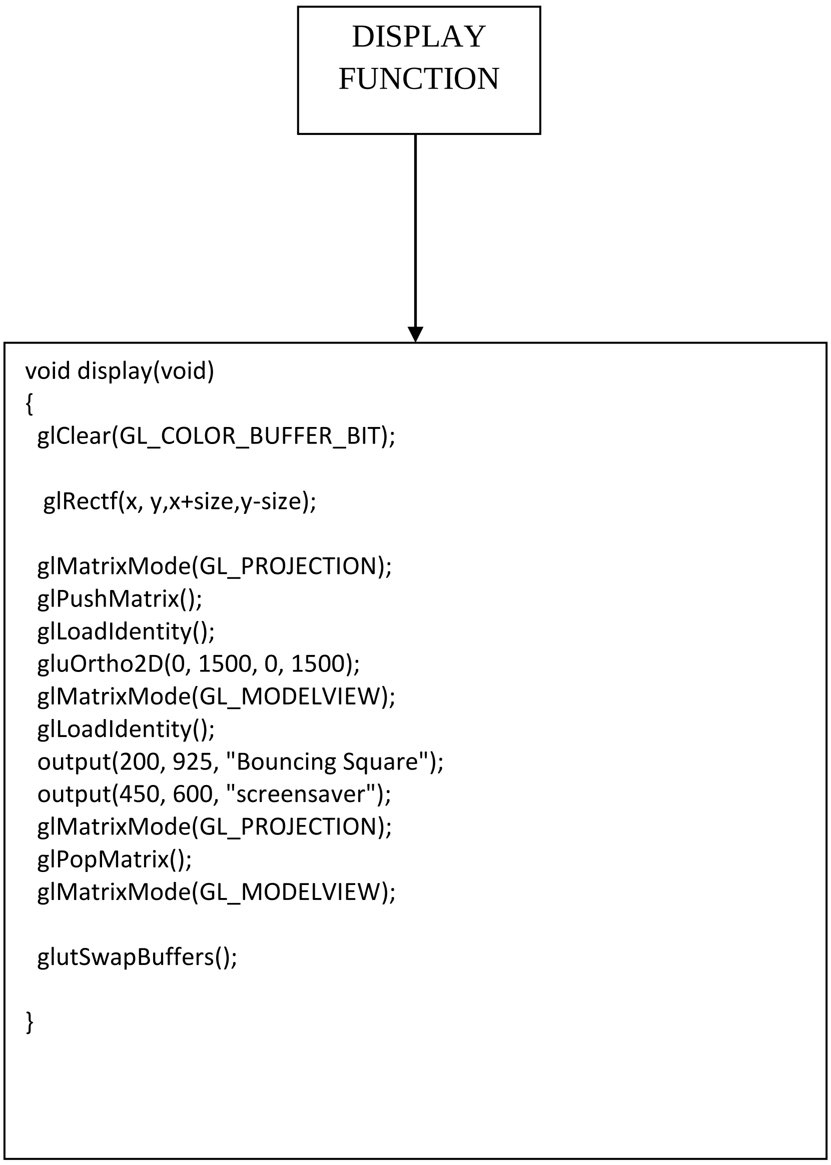


```
void menu1(int value)
{
  switch (value)
 {
  case 1:glColor3f(1.0,0.0,0.0);
        break;
  case 2:glColor3f(0.0,1.0,0.0);
        break;
  case 3:glColor3f(0.0,0.0,1.0);
        break;
  case 4:glColor3f(1.0,1.0,0.0);
        break;
  case 5:glColor3f(1.0,1.0,1.0);
        break;
  }
  glutPostRedisplay();
 }

void menu2(int value)
{
  switch (value)
 {
  case 1:glClearColor(1.0,0.0,0.0,0.0);
          break;
  case 2:glClearColor(0.0,1.0,0.0,0.0);
          break;
  case 3:glClearColor(0.0,0.0,1.0,0.0);
          break;
  case 4:glClearColor(1.0,1.0,0.0,0.0);
          break;
  case 5:glClearColor(1.0,1.0,1.0,1.0);
          break;
  }
  glutPostRedisplay();

 }
```

```
void menu3(int x)
{

  switch (x)
  {

  case 1:size=10;
        break;
  case 2:size=30;
        break;
  case 3:size=80;

        break;

  }

}

void menu4(int x)

{

  switch (x)

{

  case 1:time=15;
    break;

  case 2:time=60;
    break;

  case 3:time=140;
    break;

  }

}
```

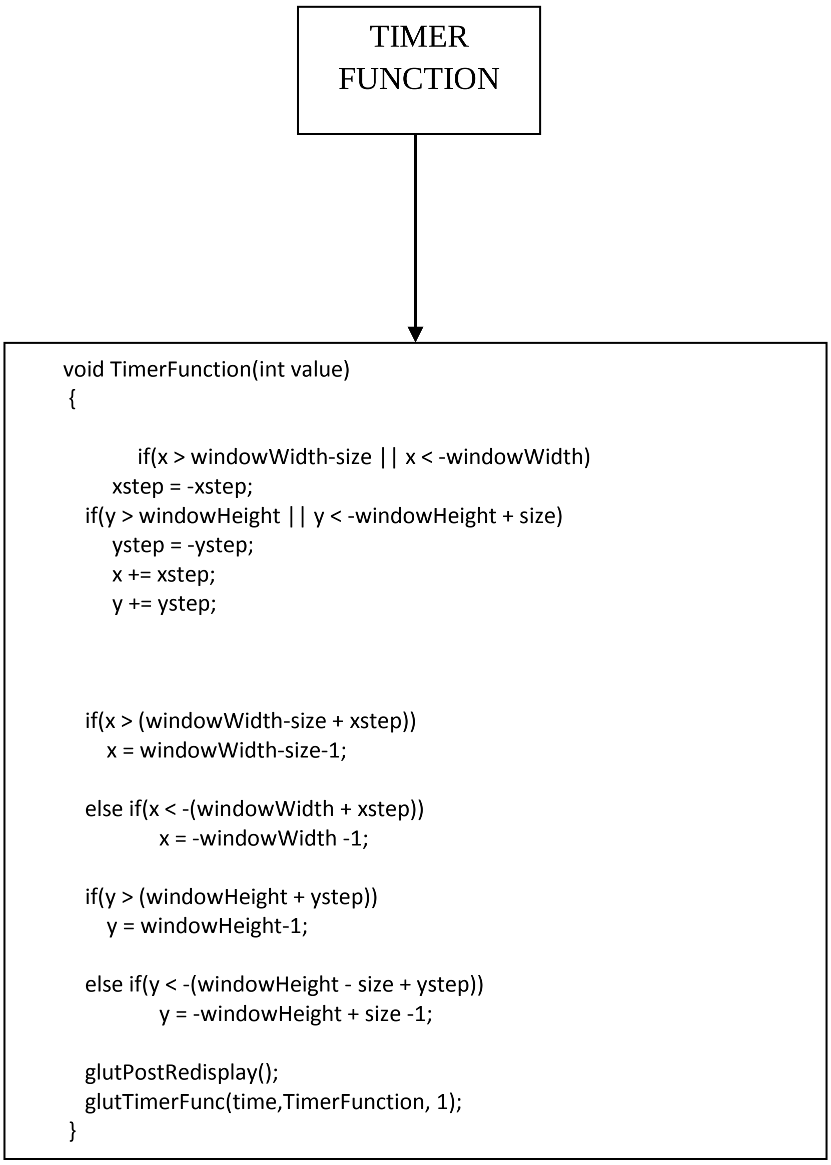
```
void menu5(int x)
{
 switch(x)
  {
   case 2: exit(0);
           break;
   case 1:glutFullScreen();
   }
}
```

4.1.2. Display function:



```
void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT);

   glRectf(x, y,x+size,y-size);

  glMatrixMode(GL_PROJECTION);
  glPushMatrix();
  glLoadIdentity();
  gluOrtho2D(0, 1500, 0, 1500);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  output(200, 925, "Bouncing Square");
  output(450, 600, "screensaver");
  glMatrixMode(GL_PROJECTION);
  glPopMatrix();
  glMatrixMode(GL_MODELVIEW);

  glutSwapBuffers();

}
```

4.1.3. TIMER function:



```
void TimerFunction(int value)
 {

          if(x > windowWidth-size || x < -windowWidth)
      xstep = -xstep;
   if(y > windowHeight || y < -windowHeight + size)
      ystep = -ystep;
      x += xstep;
      y += ystep;



   if(x > (windowWidth-size + xstep))
      x = windowWidth-size-1;

   else if(x < -(windowWidth + xstep))
             x = -windowWidth -1;

   if(y > (windowHeight + ystep))
      y = windowHeight-1;

   else if(y < -(windowHeight - size + ystep))
             y = -windowHeight + size -1;

   glutPostRedisplay();
   glutTimerFunc(time,TimerFunction, 1);
 }
```

.

Chapter 5

SOURCE CODE

```
#include<stdio.h>
#include<GL/glut.h>
//#include<stdlib.h>
#include<math.h>

GLfloat x=0.0;
GLfloat y=0.0;
GLfloat size;

GLfloat xstep=4.0;
GLfloat ystep=5.0;

GLfloat windowWidth;
GLfloat windowHeight;


int time;


void output(GLfloat x, GLfloat y, char *text)
{
  char *p;
  glPushMatrix();


 glTranslatef(x, y, 0);
  for (p = text; *p; p++)
    glutStrokeCharacter(GLUT_STROKE_ROMAN, *p);
 glPopMatrix();
}
void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT);

   glRectf(x, y,x+size,y-size);

  glMatrixMode(GL_PROJECTION);
  glPushMatrix();
  glLoadIdentity();
  gluOrtho2D(0, 1500, 0, 1500);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  output(200, 925, "Bouncing Square");
  output(450, 600, "screensaver");
  glMatrixMode(GL_PROJECTION);
```

```
glPopMatrix();
glMatrixMode(GL_MODELVIEW);

glutSwapBuffers();

}
 void TimerFunction(int value)
      {

            if(x > windowWidth-size || x < -windowWidth)
             xstep = -xstep;
         if(y > windowHeight || y < -windowHeight + size)
             ystep = -ystep;
             x += xstep;
             y += ystep;



         if(x > (windowWidth-size + xstep))
            x = windowWidth-size-1;

         else if(x < -(windowWidth + xstep))
               x = -windowWidth -1;

         if(y > (windowHeight + ystep))
            y = windowHeight-1;

         else if(y < -(windowHeight - size + ystep))
               y = -windowHeight + size -1;

         glutPostRedisplay();
         glutTimerFunc(time,TimerFunction, 1);
       }


   void ChangeSize(int w, int h)
      {
        GLfloat aspectRatio;
        if(h == 0)                        h = 1;
         glViewport(0, 0, w, h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        aspectRatio = (GLfloat)w / (GLfloat)h;


        if (w <= h)
          {
   windowWidth = 100;
   windowHeight = 100 / aspectRatio;
   glOrtho (-100.0, 100.0,-windowHeight,windowHeight, 1.0, -1.0);
```

```
            }
          else
            {
      windowWidth = 100 * aspectRatio;
      windowHeight = 100;
      glOrtho (-windowWidth,windowWidth, -100.0,100.0, 1.0, -1.0);
            }
          glMatrixMode(GL_MODELVIEW);
          glLoadIdentity();
        }
void menu1(int value)
{
  switch (value)
 {
  case 1:glColor3f(1.0,0.0,0.0);
        break;
  case 2:glColor3f(0.0,1.0,0.0);
        break;
  case 3:glColor3f(0.0,0.0,1.0);
        break;
  case 4:glColor3f(1.0,1.0,0.0);
        break;
  case 5:glColor3f(1.0,1.0,1.0);
        break;
  }
  glutPostRedisplay();

 }
void menu2(int value)
{
  switch (value)
 {
  case 1:glClearColor(1.0,0.0,0.0,0.0);
          break;
  case 2:glClearColor(0.0,1.0,0.0,0.0);
          break;
  case 3:glClearColor(0.0,0.0,1.0,0.0);
          break;
  case 4:glClearColor(1.0,1.0,0.0,0.0);
          break;
  case 5:glClearColor(1.0,1.0,1.0,1.0);
          break;
  }
  glutPostRedisplay();

 }



void menu3(int x)
{
  switch (x)
  {
  case 1:size=10;
         break;
  case 2:size=30;
         break;
  case 3:size=80;
         break;
  }
}
void menu4(int x)
{
 switch (x)
{
  case 1:time=15;
    break;
  case 2:time=60;
    break;
  case 3:time=140;
    break;
  }
}


void menu5(int x)
{
 switch(x)
  {
   case 2: exit(0);
           break;
   case 1:glutFullScreen();
   }
}



  int main(int argc,char*argv[])

  {
             int submenu1,submenu2,submenu3,submenu4;
             size=25.0;
              time=25.0;
             glutInit(&argc,argv);
             glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH );
             glutInitWindowSize(500,500);
             glutCreateWindow("BOUNCING SQUARE SCREENSAVER");
             glutDisplayFunc(display);
            glutReshapeFunc(ChangeSize);
               glutTimerFunc(time, TimerFunction, 1);
```

```
        submenu1 = glutCreateMenu(menu1);
        glutAddMenuEntry("RED", 1);
        glutAddMenuEntry("GREEN",2);
        glutAddMenuEntry("BLUE",3);
        glutAddMenuEntry("YELLOW",4);
        glutAddMenuEntry("WHITE", 5);

        submenu2 = glutCreateMenu(menu2);
        glutAddMenuEntry("RED", 1);
        glutAddMenuEntry("GREEN", 2);
        glutAddMenuEntry("BLUE", 3);
        glutAddMenuEntry("YELLOW", 4);
        glutAddMenuEntry("WHITE", 5);

        submenu3 = glutCreateMenu(menu3);
        glutAddMenuEntry("small", 1);
        glutAddMenuEntry("medium", 2);
        glutAddMenuEntry("large", 3);

        submenu4 = glutCreateMenu(menu4);
        glutAddMenuEntry("very slow",3);
        glutAddMenuEntry("medium", 2);
        glutAddMenuEntry("fast", 1);

        glutCreateMenu(menu5);
        glutAddMenuEntry("Full screen",1);
        glutAddMenuEntry("Quit",2);
        glutAddSubMenu("Background color",submenu2);
        glutAddSubMenu("Size of square", submenu3);
        glutAddSubMenu("Bouncing Speed", submenu4);
        glutAddSubMenu("Color of square/text",submenu1);
        glutAttachMenu(GLUT_RIGHT_BUTTON);

        glutMainLoop();
        return 0;
}
```

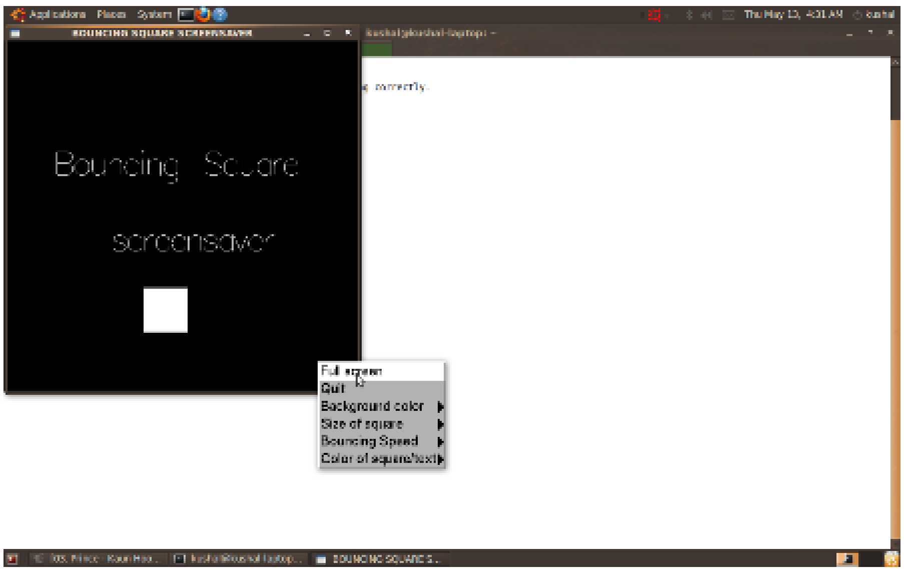
Chapter 6

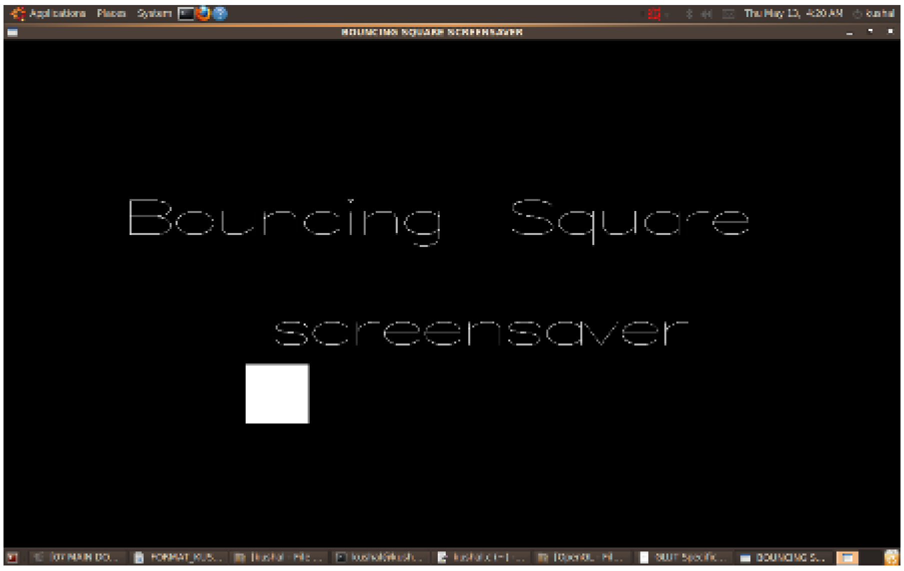
TESTING WITH SCREENSHOTS

CASE 1:

INPUT - when the user clicks on full screen.

OUTPUT- screen maximizes along with square which maximizes according to screen size.

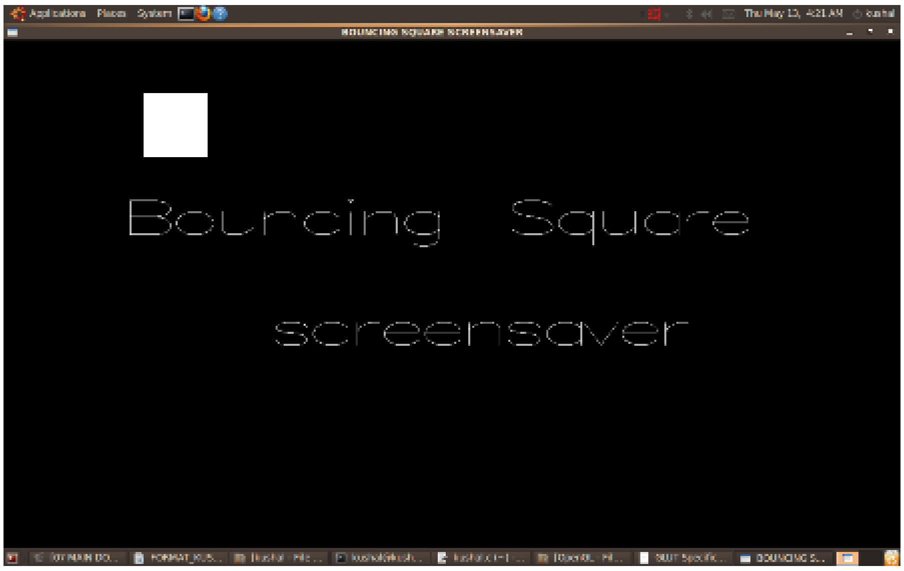




CASE 2 :

INPUT - when the square hits the top edge

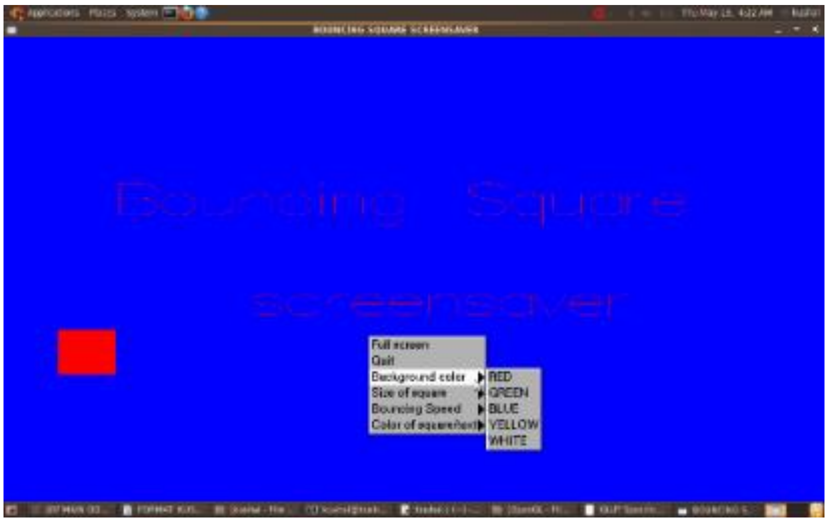
OUTPUT - bounces back as expected



CASE 3 :

Input – changing the background color to blue and square to red

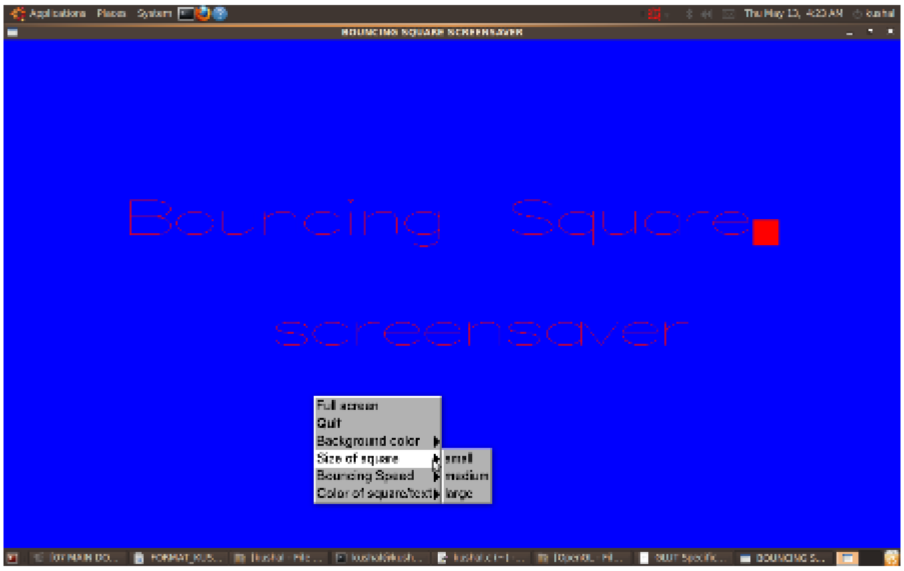
Output – background color changes to blue and square turns red



CASE 4:

Input – changing the size of square to small

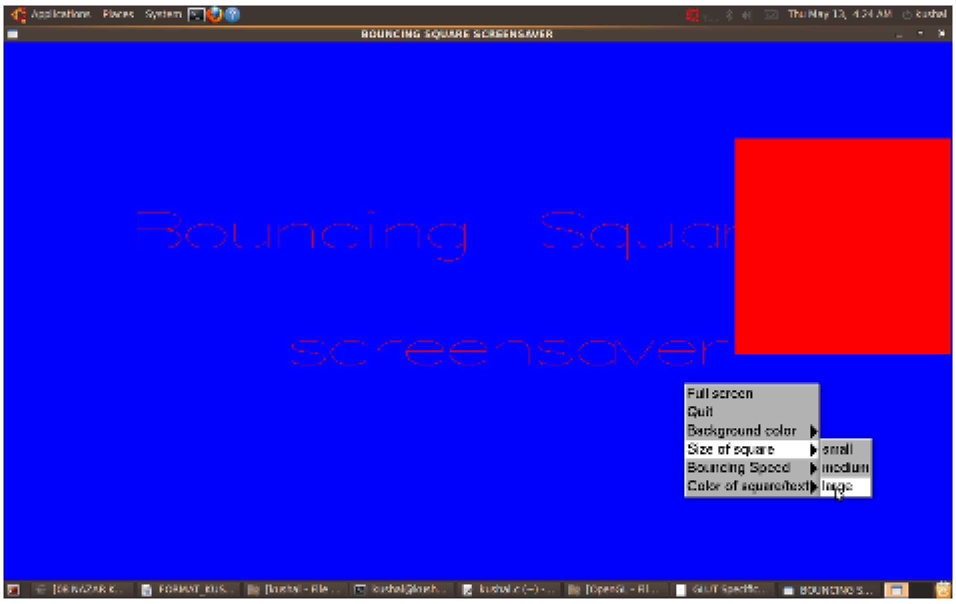
Output –square turns smaller



CASE 5 :

Input – changing the size of square to large

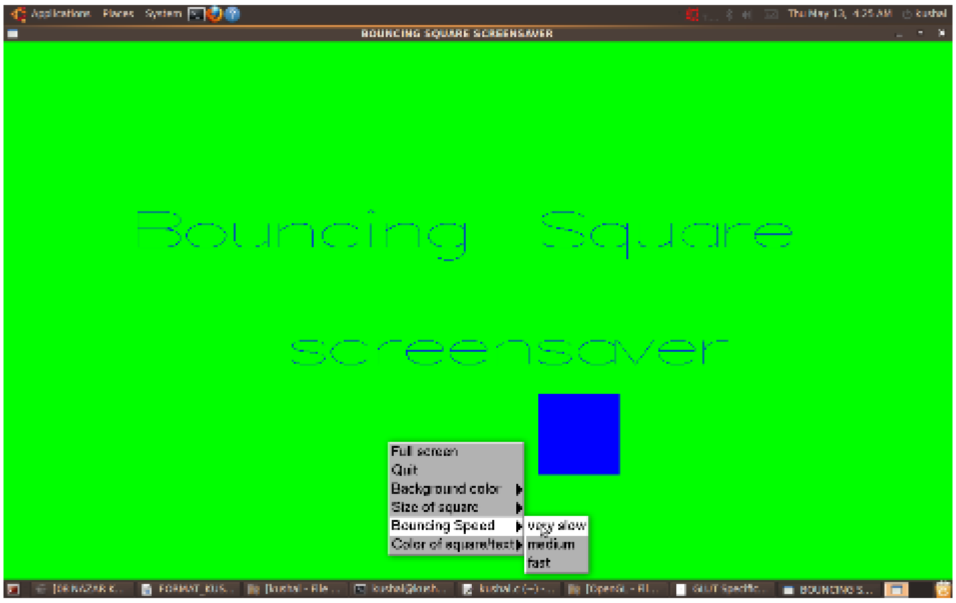
Output –square tu



CASE 6 :

Input – Once again changing the background color and changing the speed of motion

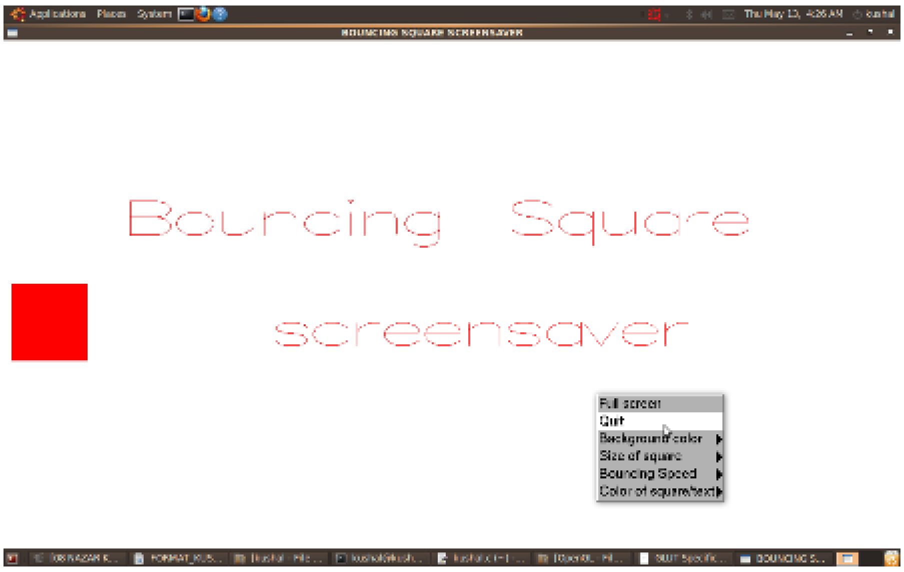
Output –background color changed and speed slowed down



CASE 7 :

Input – Quiting from display window

Output – display window quits



Chapter 7

CONCLUSION

Doing this project was a great learning experience. I got to know many OpenGL inbuilt functions. Even i dealt with many new functions which were not known to me. Taking my project as the basis i can implement various screensavers .OpenGL makes so many animations possible with the effects of scaling transformations and many more. And gave me a confidence and boost to do for more projects.

Chapter 8

FUTURE ENHANCEMENT

My project Screen Saver i.e. Bouncing of a square can be modified which i would not able to do because of lack of time. Modifications such as the color of the primitive can be changed to as and when it strikes the boundary of the screen. Shading effects ,scaling effects can be taken into consideration. This can be actually used as a screen saver.

Chapter 9

BIBLIOGRAPHY

REFERENCE BOOKS:

- Interactive Computer Graphics by Edward Angel 5th edition Pearson Education
- The OpenGL Utility Toolkit (GLUT) Programming Interface- Mark J. Kilgard

WEBSITES

- www.OpenGL.org
- www.wikipedia.com
- Last but not least www.google.com