Visvesvaraya Technological University, Belagavi – 590010

COMPUTER GRAPHICS AND VISUALIZATION
MINI PROJECT REPORT ON

FIND THE BLOCK GAME

Submitted by

TG CHANDRAKALA	4SO19CS169
VAGISH	4SO19CS170

Under the guidance of

Ms Jaishma Kumari B
(Assistant Professor, CSE Department)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ST JOSEPH ENGINEERING COLLEGE

Vamanjoor, Mangaluru -575028, Karnataka 2021-2022

ST JOSEPH ENGINEERING COLLEGE
Vamanjoor, Mangaluru- 575 028

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini project entitled ***"FIND IT BLOCKS GAME"*** *is a bonafide work carried out by*

TG CHANDRAKALA	*4SO19CS169*
VAGISHA	*4SO19CS170*

Students of Sixth semester B.E. Computer Science & Engineering, and submitted as a part of the course Computer Graphics And Application (18CSL67), during the academic year 2021-2022.

Ms. Jaishma Kumari B	**Dr Sridevi Saralaya**
Project Guide	**Head of the Department**

Name of the Examiners	**Signature with Date**
1. ----------	1. ----------
2----------	2. ----------

ABSTRACT

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. A particular graphics software system called OpenGL, which has become a widely accepted standard for developing graphics applications.

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.

Our project named “FIND IT BLOCKS GAME” uses OpenGL software interface and develops 2D images. This project uses simple techniques like OpenGl functions, mouse and keyboard functions. This project consists of 2D blocks where the user flips the blocks that match the image with other blocks. IF the user matches all the images, then the game is completed.

ACKNOWLEDGEMENT

We dedicate this page to acknowledge and thank those responsible for the shaping of the project. Without their guidance and help, the experience while constructing the dissertation would not have been so smooth and efficient.

We are extremely thankful to our Director, **Rev. Fr. Wilfred P D'Souza** our **Principal Dr Rio D'Souza** for their support and encouragement.

We owe our profound gratitude to **Dr Sridevi Saralaya, Head of the Department**, Computer Science and Engineering, whose kind consent and guidance helped us to complete this work successfully.

We sincerely thank **MS.JAISHMA KUMARI B, Assistant Professor**, Computer Science and Engineering, for his guidance and valuable suggestions which helped us to fulfill the experiments prescribed by the university.

We would like to thank all our Computer Science and Engineering staff members who have always been with us extending their support, precious suggestions, guidance and encouragement through the project.

We also like to extend thanks to our friends and family members for their continuous support.

CONTENTS

CHAPTER 1

INTRODUCTION

Computer graphics are pictures and films created using computers. Usually, the term refers to computer-generated image data created with help from specialized graphical hardware and software. It is a vast and recent area in computer science. It is concerned with all the aspects of producing pictures or images using a computer. Computer graphics is used to improve computer pictures and to simulate real world scenes.

The Project 'Find it blocks game' in which users need to remember the inside pattern of a cubic block and then match with the inside of another cubic block in continuous two steps. This program helps in learning to generate random shapes and checks your memory. "The objects are designed with simple OpenGL graphics objects"

1.1 Problem Definition

A 'find it blocks' game in which a few blocks have objects with different coloured objects hidden behind. Move the focus on to an object and press enter to uncover it. As soon as you move to another block this uncovered block gets covered. You have to uncover two blocks with same object hidden behind to clear both the blocks. This is a time-based game. Faster the board is cleared the higher the points.

1.2 Scope & Importance

In the future, we may convert this project to be a 3D project, with the necessary lighting and shading effects to give the story a more realistic appearance. Also we hope to make it more interactive by adding the functionality to allow the user to input certain parameters.

CHAPTER 2
SOFTWARE REQUIREMENT SPECIFICATION

2.1 Functional Requirement Specification

void glutInit(int *argc, char argv)

Initializes GLUT. The arguments from main are passed in and can be used by the application. argc- A pointer to the unmodified argc variable from the main function. argv- A pointer to the unmodified argv variable from the main function.

int glutCreateWindow(char *title)

Create a window on display. The string title can be used to label windows. The return value provides a reference to the window that can be used when there are multiple windows. The return value is the window handler for that particular Window. title- sets the window title.

void glutInitDisplayMode (unsigned int mode)

Requests a display with the properties in mode. The value of the mode is determined by logical OR of options including the colour model(GLUT_RGB,GLUT_INDEX) and buffering(GLUT_SINGLE,GLUT_DOUBLE). **mode-** specifies the display mode. GLUT_SINGLE- single buffer window GLUT_DOUBLE- double buffer window, required to have smooth animation.

void glutMainLoop()

This function causes the program to enter an event-processing loop. It should be the last statement in main.

void glutSwapBuffers()

Swaps the front and back buffers.

glEnable(GL_DEPTH_TEST)

If enabled, do depth comparisons and update the depth buffer.

glClearColor(GLclampf r, GLclampf g, GLclampf b, GLclampf a)

Sets the present RGBA clear color used when clearing the color buffer. Variables of GLclampf are floating point numbers between 0.0 and 1.0.

void glcolor3[ub][f](TYPE r, TYPE g, TYPE b)

Sets the present RGB color. Maximum and minimum value is 255.0 and 0.0 respectively.

void glutReshapeFunc(void (*func)(int width, int height))

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.

void glBegin(glEnum mode)

Initiates a new primitive of type mode and starts the collection of vertices. Value of mode can be

GL_POINTS, GL_LINES or GL_POLYGON.

void glEnd()

Terminates the list of vertices.

void glutSpecialFunc(void (*func)(int key, int x, int y));

`glutSpecialFunc` sets the special keyboard callback for the *current window*. The special keyboard callback is triggered when keyboard function or directional keys are pressed.

void glMatrixMode(GLenum mode)

Specifies which matrix will be affected by subsequent transformations, mode can be GL_MODELVIEW, GL_PROJECTION etc.

void glLoadIdentity()

Sets the current transformation matrix to identity matrix.

2.2 Software Requirement Specification

Operating System: Linux (Pop OS)

Language: C programing

Tools: vs code, glut.h Library, C terminal.

2.2 Hardware Requirement Specification

Installed Memory(RAM): 2GB or Higher.

Processor: 1GHz or Higher

Hard disk space: 20 GB Availability

Display: Standard output display

CHAPTER 3

IMPLEMENTATION

1) Drawing a Cube :

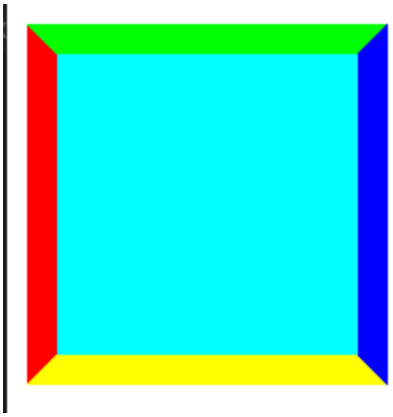
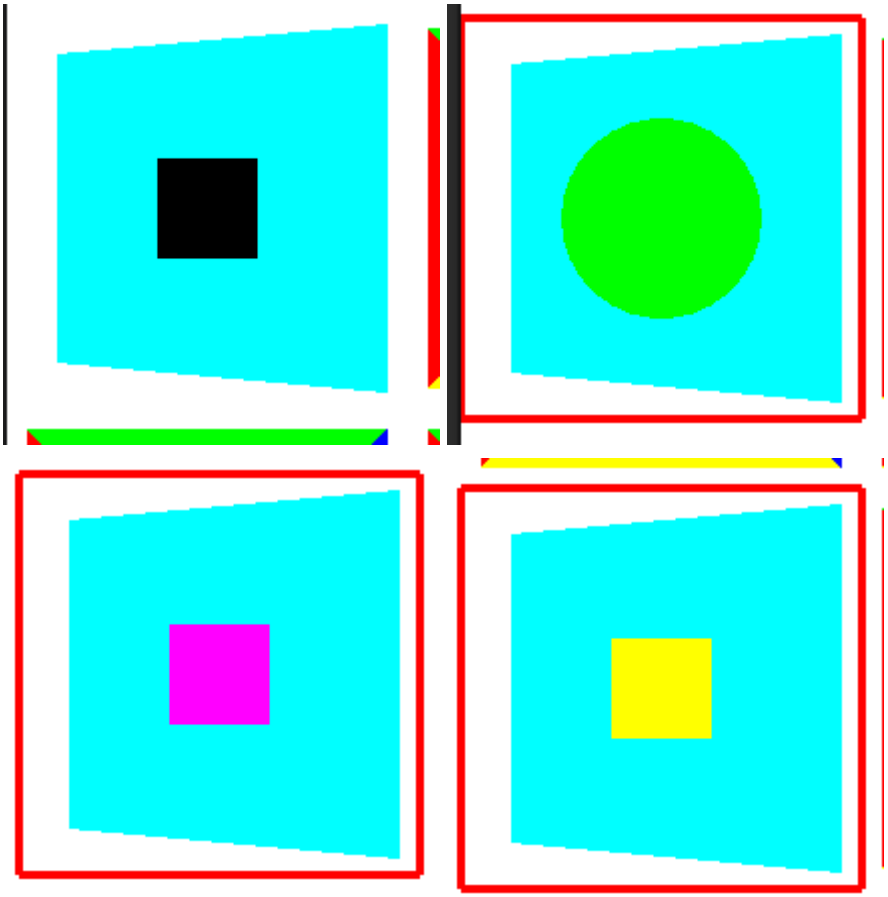


fig 1: Cube

<<Pseudo Code Snippet>>

```
glColor3f(0.0,1.0,1.0);
   glBegin(GL_POLYGON);
   glVertex3fv(cube[1]);
   glVertex3fv(cube[0]);
   glVertex3fv(cube[6]);
   glVertex3fv(cube[7]);
   glEnd();
```

2) Inner objects :



<<Pseudo Code Snippet>>

```
void selects() //to select inner pattern
  {
      switch(r)
      {
       case 1:glColor3f(1,1,1);glutSolidSphere(10,30,30);break;//white
sphere

       case 2:glColor3f(1,1,0);glutSolidCube(10);break;//yellow cube

       case 3:glColor3f(0,0,0);glutSolidCube(10);break;//black cube

       case 4:glColor3f(1,0,1);glutSolidCube(10);break;//purple cube
```

```
        case 5:glColor3f(0,0,1);glutSolidTorus(2.0,4.0,5,5);break;//blue
torus

        case 6:glColor3f(1,1,1);glutSolidSphere(10,30,30);break;//white
sphere

        case 7:glColor3f(1,1,1);glutSolidCube(10);break;//white cube

        case 8:glColor3f(1,1,0);glutSolidTorus(2.0,4.0,5,5);break;//yellow
torus

        case 9:glColor3f(0,0,0);glutSolidSphere(10,30,30);break;//black
Sphere

        case 10:glColor3f(1,1,0);glutSolidCube(10);break;//yellow cube

        case 11:glColor3f(1,0,1);glutSolidCube(10);break;//purple cube

        case 12:glColor3f(1,1,1);glutSolidTorus(2.0,4.0,5,5);// sky blue
torus

        case 13:glColor3f(0,0,1);glutSolidTorus(2.0,4.0,5,5);break;//blue
torus

        case 14:glColor3f(0,1,0);glutSolidSphere(10.0,30,30);break;//green
sphere

        case 15:glColor3f(0,0,0);glutSolidCube(10);break;//black cube

        case 16:glColor3f(1,1,1);glutSolidCube(10);break;//white cube

        case
17:glColor3f(1,1,0);glutSolidTorus(2.0,4.0,5,5);break;//yellow torus

        case 18:glColor3f(0,0,0);glutSolidSphere(10,30,30);break;//black
Sphere

        case 19:glColor3f(0,1,0);glutSolidSphere(10.0,30,30);break;//green
sphere

        case 20:glColor3f(1,1,1);glutSolidTorus(2.0,4.0,5,5);// sky blue
torus
```

```
    }
}
```

3)KeyBoard function :

<<Pseudo Code Snippet>>

```
void mykey(unsigned char key,int x,int y) //function to handle "normal"
key presses
        {
        switch(key)
         {
           case 32:   k=1;
              glutPostRedisplay();
                  break;
       case 27:  exit(0);
       case 13:
               f=1;

               if(m!=10)
               no_of_enters++;

               if(count==0)
                {
               count++;
               status_count1=status;
               stat=1;
                }

               else if(count==1)
               count++;

               glutPostRedisplay();

                  break;

       case 'x': h=1;
             glutPostRedisplay();

              break;
```

```
          case 'y':glRotatef(3,1,1,0);glutPostRedisplay();break;

              case
'r':if(m==10){initialize();randm();glutPostRedisplay();}break;
          }
       }
  void myspecial(int key,int x,int y)  //function to handle "special" key
presses
   {
     switch(key)
      {
   case GLUT_KEY_UP:if(status>5)
                          {
                     status-=5;
                     glColor3f(1.0,0.0,0.0);
                     yl=yl-40;y2=y2-40;y3=y3-40;y4=y4-40;
                     rectangle(x1,yl,x2,y2,x3,y3,x4,y4);
                 if(count==2) count=0,f=0,status_count1=status;
                     glutPostRedisplay();
                          }
                            break;
   case GLUT_KEY_DOWN:if(status<16)
                            {
                      status+=5;
                      glColor3f(1.0,0.0,0.0);
                      yl=yl+40;y2=y2+40;y3=y3+40;y4=y4+40;
                      rectangle(x1,yl,x2,y2,x3,y3,x4,y4);
                  if(count==2)count=0,f=0,status_count1=status;
                      glutPostRedisplay();
                           }
                              break;
   case GLUT_KEY_LEFT:if(status!=1&&status!=6&&status!=11&&status!=16)
                            {
                       status--;
                       glColor3f(1.0,0.0,0.0);
                       x1=x1+40;x2=x2+40;x3=x3+40;x4=x4+40;
                   rectangle(x1,yl,x2,y2,x3,y3,x4,y4);
              if(count==2)count=0,f=0,status_count1=status;
                       glutPostRedisplay();
                            }
                               break;
   case GLUT_KEY_RIGHT:if(status!=5&&status!=10&&status!=15&&status!=20)
```

```
                            {
                        status++;
                        glColor3f(1.0,0.0,0.0);
                        x1=x1-40;x2=x2-40;x3=x3-40;x4=x4-40;
                        rectangle(x1,yl,x2,y2,x3,y3,x4,y4);
                if(count==2)count=0,f=0,status_count1=status;
                        glutPostRedisplay();
                    }
                        break;
    }
  }
```

4) Score Calculation Function :

<<Pseudo Code snippet>>

```
void calculate() //find score
    {

       int score,scor[]={0,0,0,0};
     int rem=0;
     char score1[]={'0','0','0','0','\0'};
     char text8[]={"YOUR SCORE IS :"};

     score=20000/(no_of_enters);
     j=3;
     while(score!=0)
     {
       rem=score%10;
       scor[j--]=rem;
       score=score/10;
     }
      for(j=0;j<4;j++)
      score1[j]=(scor[j]+48);
          glColor3f(1,0,0);
          glRasterPos3f(30,0,0);
          for(j=0;text[j]!='\0';j++)
          glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, text[j]);
      glRasterPos3f(30,25,0);
          for(j=0;text[j]!='\0';j++)
          glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, text[j]);
```

```
    glRasterPos3f(20,10,0);
    for(j=0;text8[j]!='\0';j++)
    glutBitmapCharacter(GLUT_BITMAP_9_BY_15, text8[j]);
    for(j=0;score1[j]!='\0';j++)
    glutBitmapCharacter(GLUT_BITMAP_9_BY_15, score1[j]);
   }
```

CHAPTER 4

SCREENSHOTS

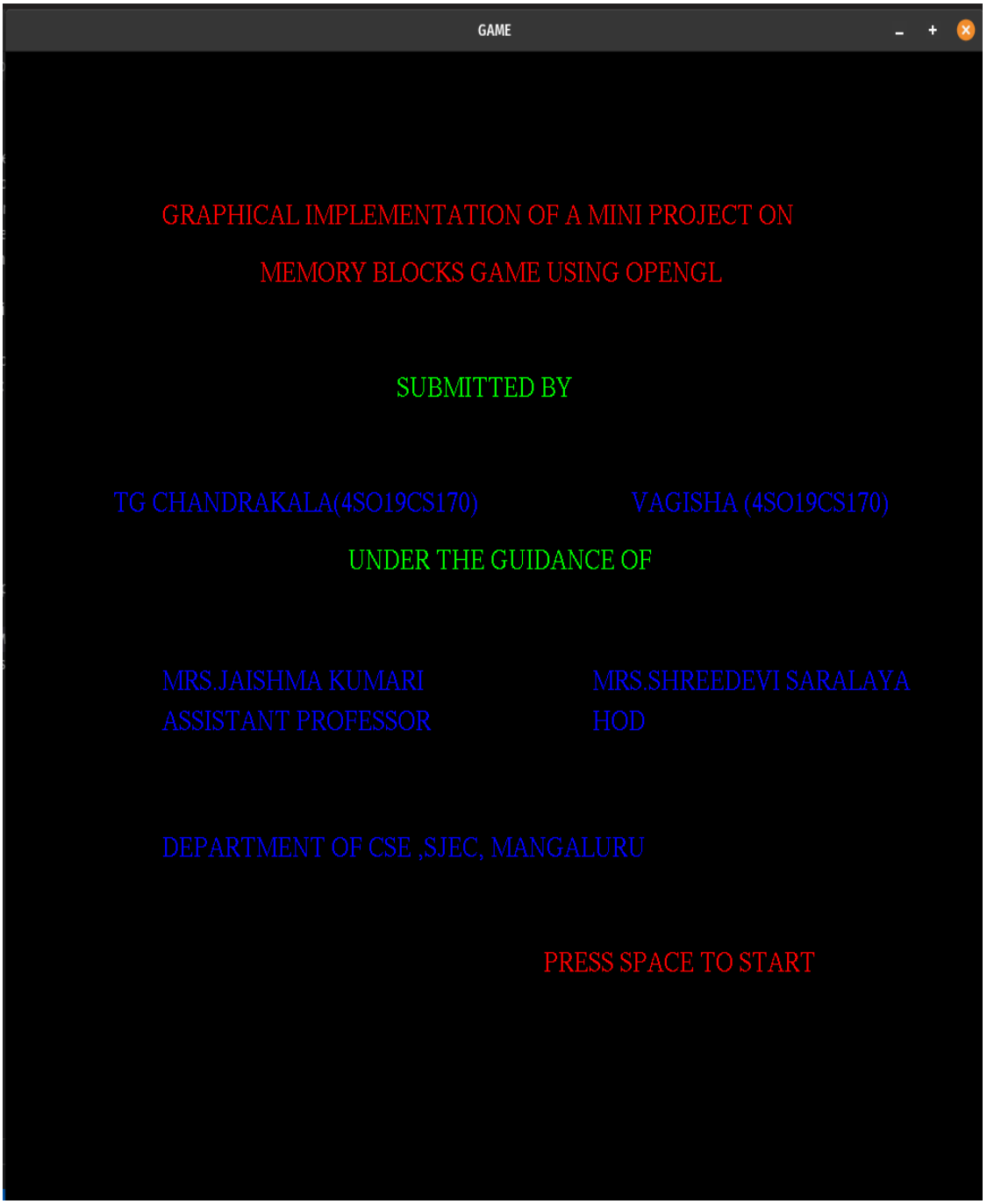


fig 1: User Interface

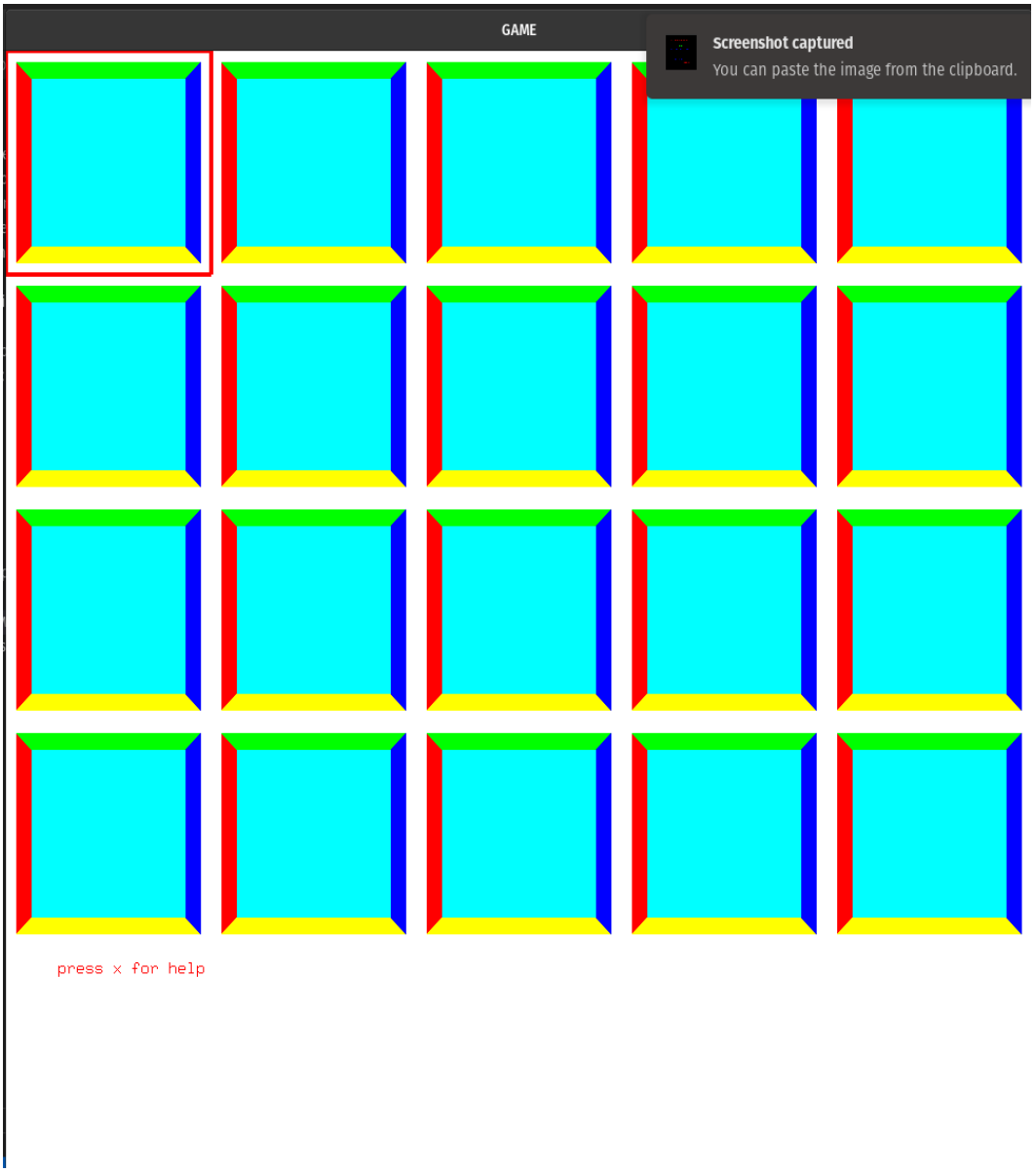


fig 2:Unfolded Cubes

HELP

1) use arrow keys for navigation

2) press enter to select the cube

3) press Esc to quit

fig3: Game Hints to user

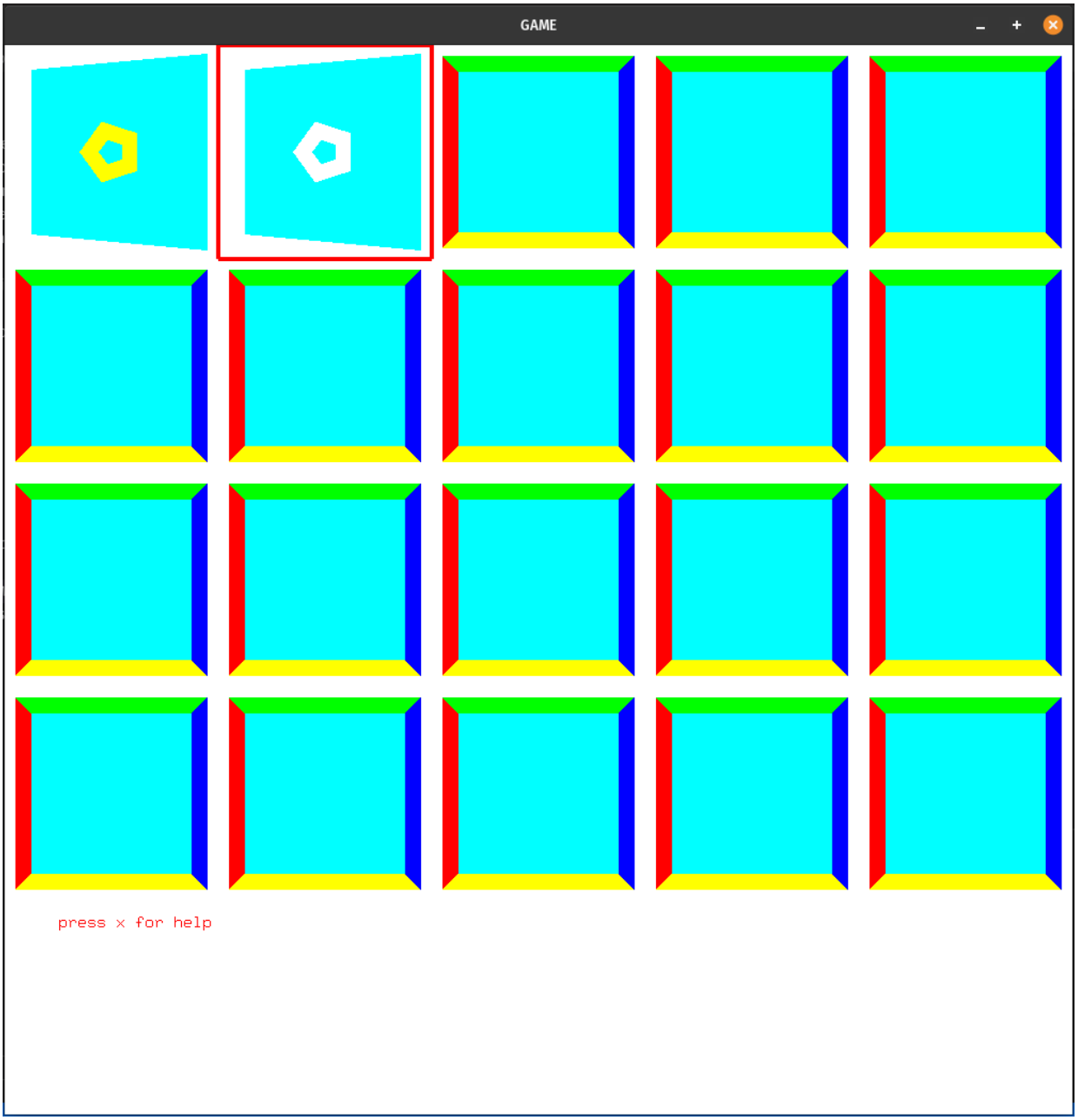


fig 4:Selection of cubes and checks for matching

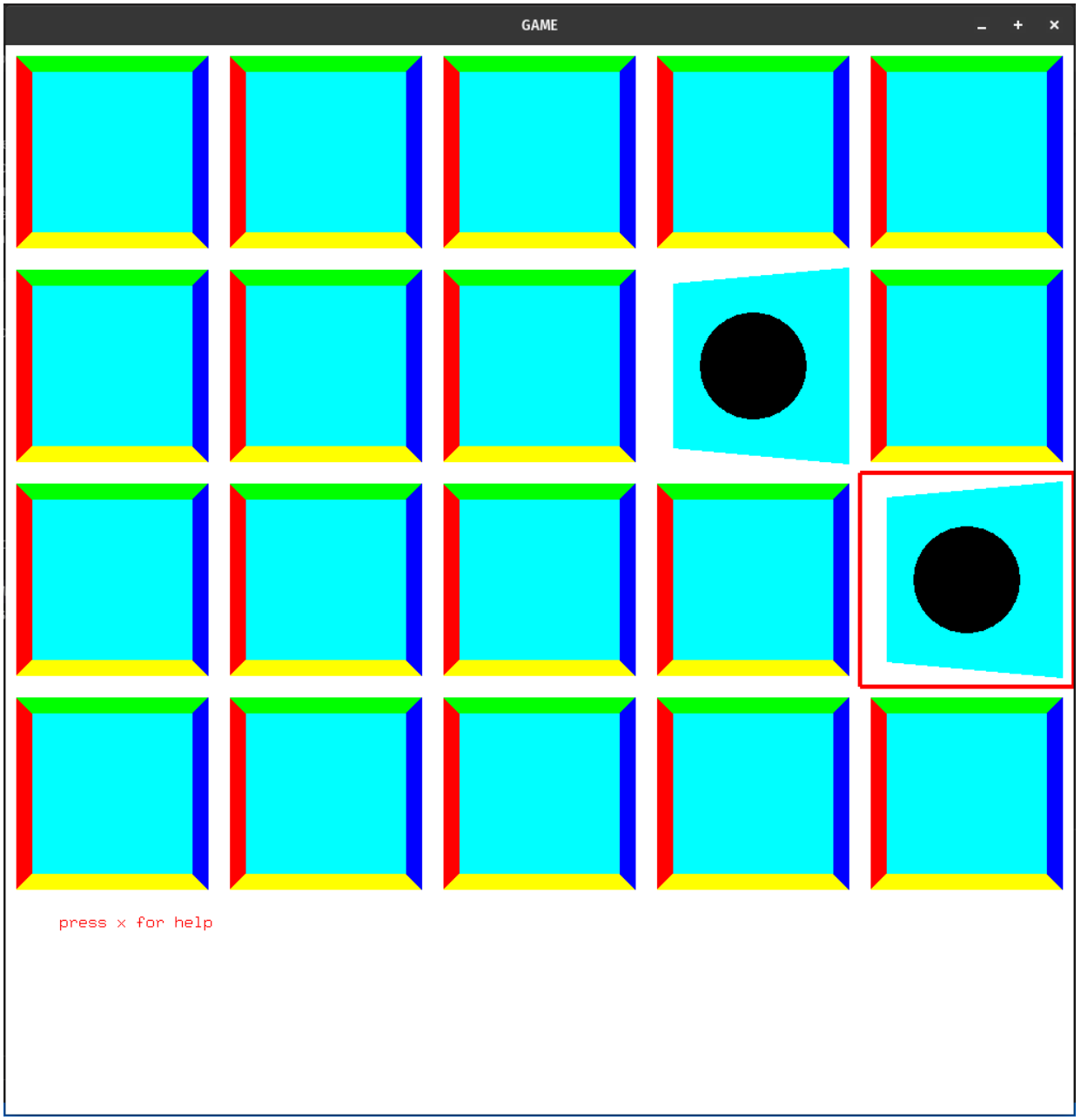


fig 5:Matched Cubes

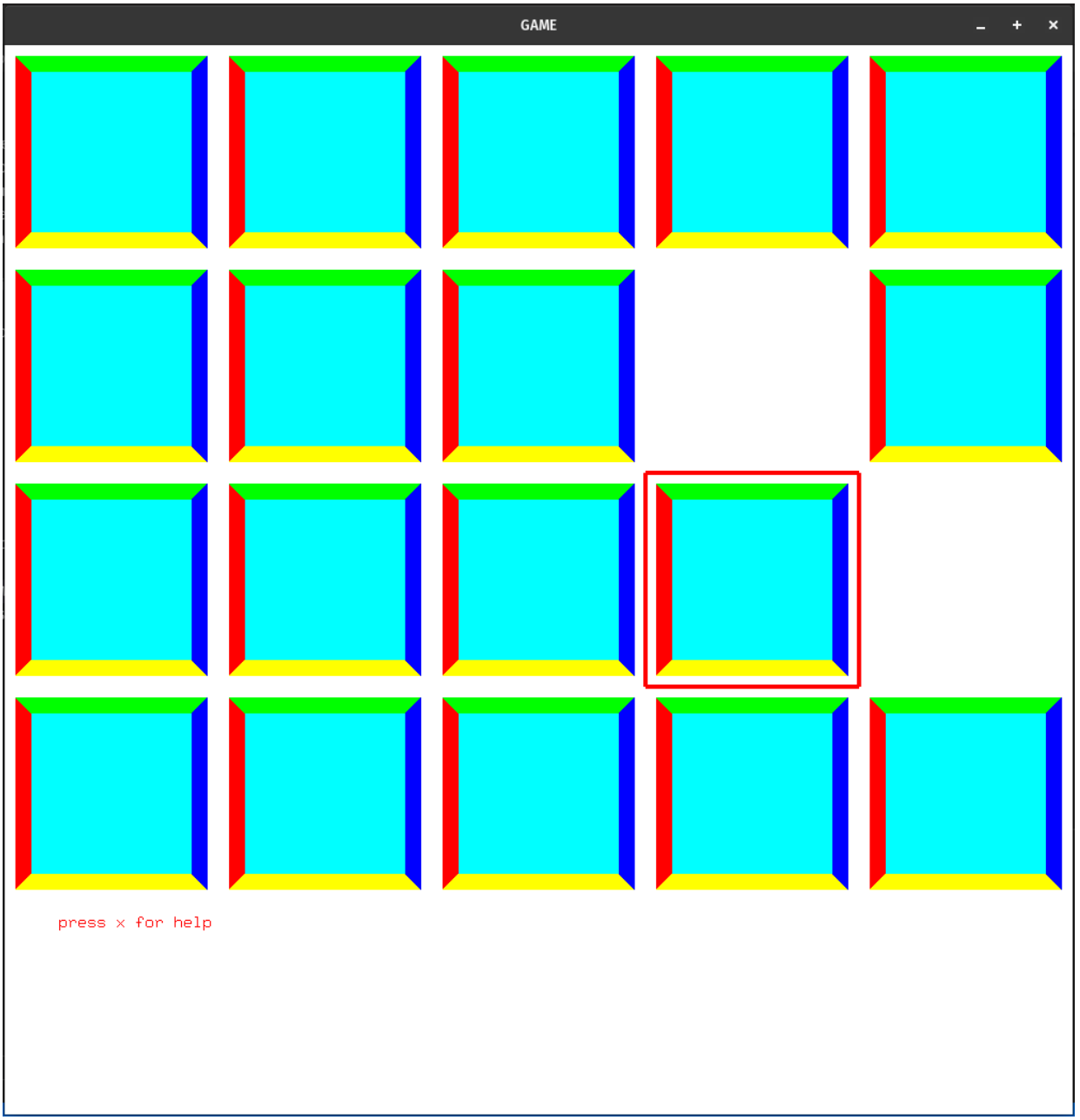


fig 6:matched cubes gets deleted

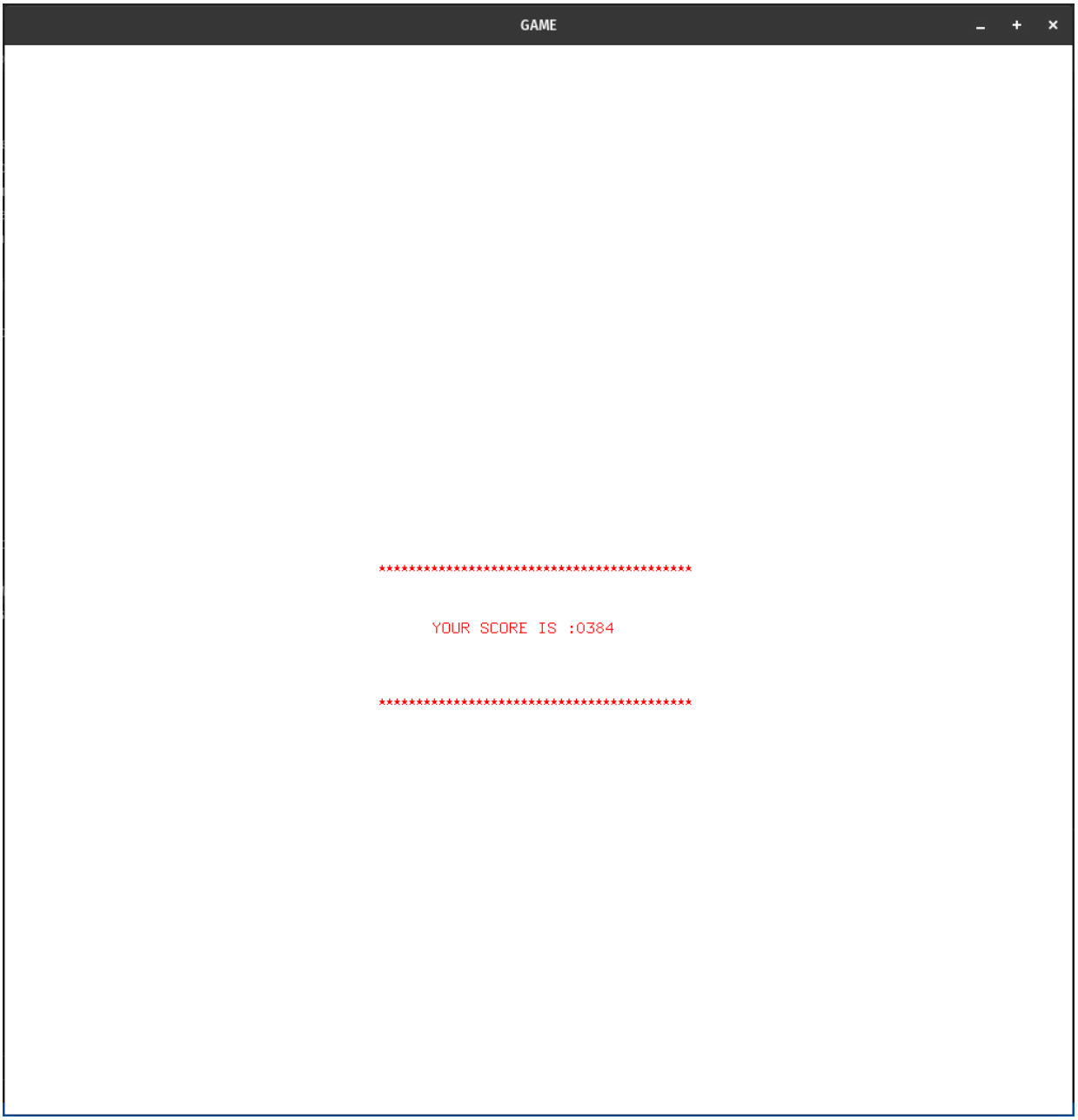


fig 8:Total Score

CHAPTER 5
CONCLUSION AND FUTURE WORK

The primary goal of this project is to showcase the concepts that we have learnt in theory and how these concepts can be used to visually demonstrate algorithms in computer science. By visually representing transformations in an animation form, the viewer gets a clear idea of it. The project has also helped in understanding the working of computer graphics using OpenGL and the various concepts, functions and methodologies required for the development of a graphics package.

REFERENCES/BIBLIOGRAPHY

1. Edward Angel: Interactive Computer Graphics, 5thEdition, Pearson Education, 2008

2. http://www.opengl.org/sdk/docs/man2/xhtml/

3. http://www.opengl.org/documentation/specs/glut/spec3/node1.html

4. http://en.m.wikipedia.org/wiki/Computer_graphics