CHAPTER 1
INTRODUCTION

This project is a simpler take on the well known SNAKE GAME using a set of OpenGL functions.

This game mainly demonstrates affine transformations of 3D objects such as translation, rotation and scaling along with lighting effects.

We are using keyboard and mouse to play the game. The various features of the game are:

- Change the theme of the game board
- Rotation of game board
- Effects of lighting seen from different angles
- High score based game with timer
- Mouse and keyboard interface
- Enable and disable lighting and shading

It has been developed on Fedora Linux with OpenGL package installed using C. The report also includes overview, implementation, analysis, design and results of the above mentioned project along with its source code.

1.1 Computer Graphics

Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention of camera and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computers themselves. It has grown to include the creation, storage, and manipulation of models and images of objects.

These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural, and even conceptual structures, natural phenomena, and so on. Computer graphics today is largely interactive. The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touch-screen. Due to close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics. The advantages of the interactive graphics are many in number. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D patter-recognition abilities allow us to perceive and process data rapidly and efficiently. In many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the 1980s when the scientists and engineers realized that they could not interpret the prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

1.2 OpenGL Interface

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects.

Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are:

- Main GL: Library has names that begin with the letter GL and are stored in a library usually referred to as GL.
- OpenGL Utility Library (GLU): This library uses only GL functions but contains code for creating common objects and simplifying viewing.
- OpenGL Utility Toolkit (GLUT): This provides the minimum functionality that should be accepted in any modern windowing system.

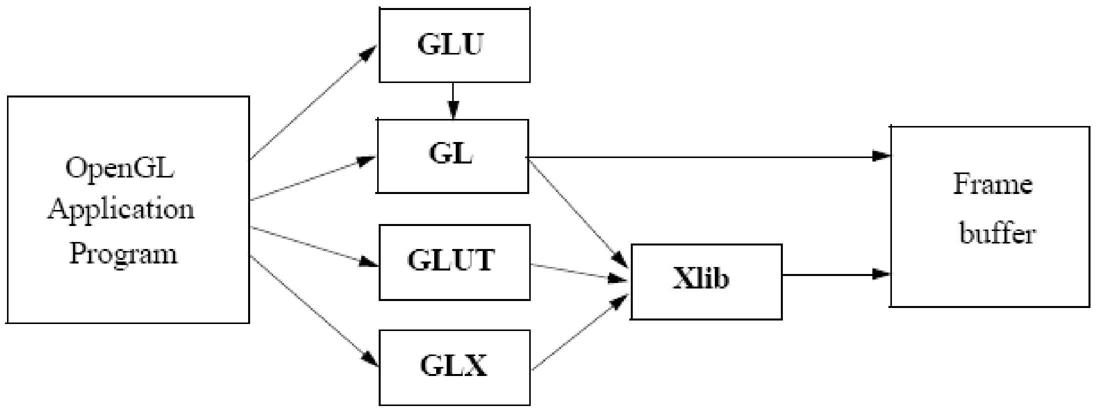


Fig1.1: Library organization

Some features of OpenGL include the following:

- Geometric and raster primitives RGBA or color index mode
- Display list or immediate mode
- Viewing and modeling transformations
- Lighting and Shading
- Hidden surface removal (Depth Buffer)
- Texture Mapping

1.3 OpenGL Overview

OpenGL (Open Graphics Library) is the interface between a graphic program and graphics hardware. It is streamlined. In other words, it provides low-level functionality. For example, all objects are built from points, lines and convex polygons. Higher level objects like cubes are implemented as six four-sided polygons.
OpenGL supports features like 3-dimensions, lighting, anti-aliasing, shadows, textures, depth effects, etc.

- It is system-independent. It does not assume anything about hardware or operating system and is only concerned with efficiently rendering mathematically described scenes. As a result, it does not provide any windowing capabilities.
- It is a state machine. At any moment during the execution of a program there is a current model transformation
- It is a rendering pipeline**.** The rendering pipeline consists of the following steps:

- Defines objects mathematically.
- Arranges objects in space relative to a viewpoint.
- Calculates the color of the objects.
- Rasterizes the objects.

1.4 Objective

- The aim of the project is to make an entertaining game.
- Adding a timer and having high scores makes the game competitive. Having elements like rotating special food which are up for more points make it fun.
- It is made user friendly with intuitive interactions and appropriate instructions are provided wherever necessary.
- Interactions happen via the keyboard. In addition to this, a menu is triggered on right click of the mouse offering various options.
- Having the option to play on different themed boards makes the game more interesting.
- Lighting effects can be noticed by rotating the board and by enabling and disabling lighting.

CHAPTER 2
DESIGN

2.1 Overview

The player has to move around the game board without hitting the boundary wall and get to the food that appears on the board as soon as possible. Every third food item rotates and is up for greater points. The board can be rotated along the x-axis to be able to see the effects of lighting. Two different themes are available for the game board, which changes the boundary design of the board. The game can be restarted without affecting the high score in a session. Lighting can be enabled and disabled.

2.2 User Interface

2.2.1 Keyboard Controls

s	Starts the game session
Arrow keys	Helps to move around the board
p	Restart the game
r	Rotate the board in forward direction
a	Rotate the board in backward direction

2.2.2 Mouse Interaction

Menu is displayed by clicking the Right Mouse Button (RMB). Left click selects the required option. The various options are discussed below:

- Theme: It provides a submenu to choose one of the two themes.
- Rotate: It provides a submenu to either choose to rotate the board in forward direction or backward direction.
- Disable Lighting: Lighting is disabled.
- Enable Lighting: On disabling lighting, it can be enabled with this option.
- Restart: Allows to restart the game. The high score remains intact.
- Quit: Used to exit from the OpenGL window.

CHAPTER 3

IMPLEMENTATION

3.1 Header files

- <GL/glut.h> : This header file provides an interface with OpenGL application programs that are installed on the system.
- <stdio.h> : This header file contains various functions for performing input and output.
- <stdlib.h> : This header file contains various functions for performing general functions.
- <time.h> : This header file is used to handle time related functions .

3.2 OpenGL functions

glutPostRedisplay( );

- Marks the current window as needing to be redisplayed.
- **Usage:**
  void glutPostRedisplay( )

glutMouseFunc(mouse);

- GLUTs mouse interface provides a lot of options for adding mouse interactivity, namely detecting clicks and mouse motion. As in the keyboard version, GLUT provides a way for you to register the function that will be responsible for processing events generated by mouse clicks.
- The name of this function is *glutMouseFunc*, and it is commonly called in the initialization phase of the application.
- **Usage:**
  void glutMouseFunc(void (*func)(int button, int state, int x, int y));

glutKeyboardFunc(key);

- This OpenGL will teach you how to add keyboard interaction to your OpenGL application through the use of glut and glut's keyboard calls.

Now glut has many different keyboard calls.

- But anyway, all you have to edit after that is the 'main' function.

glutInitWindowPosition( )andglutInitWindowSize( ) :

- Set the *initial window position* and *size* respectively.
- **Usage:**

void glutInitWindowSize(int width, int height);

void glutInitWindowPosition(int x, int y);

glutCreateWindow( ):

- Creates a top-level window.
- **Usage:**

  glutCreateWindow(char *name);

glutDisplayFunc():

- Sets the display callback for the *current window*.
- **Usage:**

  void glutDisplayFunc(void (*func)(void));
- **Parameter:**

  func : The new display callback function

glutSwapBuffers( );

- Swaps the buffers of the current window if double buffered.
- **Usage:**

  void glutSwapBuffers( )

glTranslatef( );

- Displaces points by a fixed distance in a given direction.
- **Usage:**

  void glTranslate[fd](TYPE x, TYPE y, TYPE z)
- **Parameter:**

  Displacement values.

glRotatef();

- Rotate the corresponding object by a given values.
- **Usage:**
  void glRotate[fd](TYPE angle, TYPE dx, TYPE dy, TYPE dz)
- **Parameter:**
  Angle in degrees about the axis(dx,dy,dz);
- **Description:**
  Alters the current matrix by a rotation of angle in degrees.

glScalef();

- Non-rigid body transformation by which we can make an object bigger or smaller**.**
- **Usage:**
  void glScale[fd] (TYPE sx, TYPE sy, TYPE sz);
- **Parameter:**
  Fixed point, direction of scaling, scale factor.
- **Description:**
  Alters the current matrix by a scaling of(sx,sy,sz);

glClearColor( );

- Set the background color.
- **Usage:**
- void glClearColor(GLclampf *red* , GLclampf *green* , GLclampf *blue* , GLclampf *alpha*);
- **Parameter:**
- Red, green, blue, and alpha values.

glEnable( ) and glDisable( );

- Enable or disable server-side GL capabilities
- **Usage:**
  void glEnable(GLenum cap);
  void glDisable(GLenum cap);

- **Parameter:**

  A symbolic constant indicating a GL capability.

glLightfv( ) ;

- Set light source parameters
- **Usage:**

  void glLightfv(GLenum light, GLenum pname, const GLfloat *params)
- **Parameter:**

  light - Specifies a light

  pname - Specifies a light source parameter for *light*.

glMaterialfv( ) ;

- Specify material parameters for the lighting model
- **Usage:**

  void glMaterialtfv(GLenum face, GLenum pname, const GLfloat *params)
- **Parameter:**

  face - Specifies which face or faces are being updated.

  pname - Specifies the single-valued material parameter of the face

  params - Specifies a pointer to the value or values that *pname* will be set to.

glMatrixMode( ) ;

- Specify which matrix is the current matrix
- **Usage:**

  void glMatrixMode(GLenum mode);
- **Parameter:**

  mode - Specifies which matrix stack is the target for subsequent matrix operations.

glPushMatrix( ) and glPopMatrix( ) ;

- Push and pop the current matrix stack
- **Usage:**

  void glPushMatrix( );

  void glPopMatrix( );

glShadeModel ( ) ;

- Select flat or smooth shading
- **Usage:**
  void glShadeModel(GLenum mode);
- **Parameter:**
  Mode - Specifies a symbolic value representing a shading technique.

The following are the 3D objects that are created using the API's present in the utility toolkit :

glutSolidCube

Render a solid cube.

void glutSolidTorus(GLdouble *size*);

3.3 Source code

```
#include<stdio.h>
#include<stdlib.h>
#include<GL/glut.h>
#include<time.h>
#define MAX(a,b) (((a)>(b))?(a):(b))
char t1[5],t2[5],t3[5],t4[5];
int themeflag=0, lighting=1, thememenu, foodcount=0, foodover=0, scale=0;
int, gameend=0 win=0, rot=0, antirot=0, rotcount=0, antirotcount=0, scalecount=0;
clock_t start, end;
int clockstart=0, startscreen=0, gamerestart=0 score, points, timer, highscore;
static GLfloat theta[]={0.0,0.0,0.0};
static GLint axis;
GLfloat px=51.0, py=93.0, pz=0.0;
GLfloat lightintensity[]={1.0,1.0,0.0,1.0};
GLfloat lightposition[]={100.0,70.0,50.0,0.0};
glLightfv(GL_LIGHT0,GL_POSITION,lightposition);
glLightfv(GL_LIGHT0,GL_AMBIENT,lightintensity);
glColorMaterial(GL_FRONT,GL_DIFFUSE);
```

```
GLfloat amb[]={0.7,0.7,0.7,1.0};
GLfloat diff[]={1.0,1.0,1.0,1.0};
GLfloat spec[]={0.6,0.6,0.6,1.0};
GLfloat shininess[]={80.0};
GLfloat lightintensity[]={1.0,1.0,1.0,1.0};
GLfloat lightposition[]={80.0,80.0,30.0,1.0};

GLfloat foodloc[9][3]={{129,129,0},{147,15,0},{9,81,0},{111,45,0},
                        {129,165,0},{69,69,0},{27,159,0},{87,15,0},{135,135,0}};
void display();

void poly(GLfloat x1,GLfloat y1,GLfloat z1,GLfloat x2,GLfloat y2,GLfloat
z2,GLfloat x3,GLfloat y3,GLfloat z3,GLfloat x4,GLfloat y4,GLfloat z4,GLfloat
x,GLfloat y,GLfloat z)
{
  glBegin(GL_POLYGON);
          glNormal3f(x,y,z);
          glVertex3f(x1,y1,z1);
          glVertex3f(x2,y2,z2);
          glVertex3f(x3,y3,z3);
          glVertex3f(x4,y4,z4);
  glEnd();
}

void output(int x, int y, char *string)
{
     int len, i;

          glRasterPos2f(x,y);
          for (i = 0; i < string[i]!='\0'; i++)
              {
             glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,string[i]);
              }
}

void frontscreen(void)
{
          glClear(GL_COLOR_BUFFER_BIT);
          glLoadIdentity();
          glColor3f(1.0, 0.0 1.0);
          output(230,15," Press ENTER to continue");
          output(45,15,"Maximize window for better view");
          output(100,170,"BANGALORE INSTITUTE OF TECHNOLOGY");
          output(110,160,"Dept. of Computer Science and Engineering");
          output(90,140,"COMPUTER GRAPHICS AND VISUALIZATION LAB");
          output(142,125,"Mini Project: BINGE");
          output(170,110,"By :");
          glBegin(GL_LINES);
          glVertex2f(170,108);
          glVertex2f(180,108);
          glEnd();
          output(155,100,"Rakshika Raju");
          output(150,90,"USN:1BI14CS131");
          output(153,70,"Lab In-charges:");
          glBegin(GL_LINES);
          glVertex2f(153,68);
          glVertex2f(203,68);
          glEnd();
          output(116,53,"Prof. N.Thanuja");
          output(190,53,"Prof. Kavitha K");
          output(145,43,"Assistant Professors");
          output(148,33,"Dept. of CSE, BIT");
          glFlush();
}

void cubes(GLfloat t1, GLfloat t2, GLfloat t3, GLfloat s1, GLfloat s2, GLfloat s3)
```

```
{
        glPushMatrix();
        glTranslatef(t1,t2,t3);
        glScaled(s1,s2,s3);
        glutSolidCube(1);
        glPopMatrix();
}

void boundary()
{
        glColor3f(0.000, 0.749, 1.000);
        cubes(-15,90,0,6,168,6);
        cubes(69,9,0,162,6,6);
        cubes(153,90,0,6,168,6);
        cubes(69,171,0,162,6,6);
        if(themeflag==0)
        {
                glColor3f(0.000, 0.749, 1.000);
                cubes(33,129,0,6,6,6);
                cubes(27,90,0,6,84,6);
                cubes(60,51,0,60,6,6);
                cubes(99,51,0,6,6,6);
                cubes(105,90,0,6,84,6);
                cubes(72,129,0,60,6,6);
        }
        else
        {
                glColor3f(0.000, 0.749, 1.000);
                cubes(21,132,0,6,72,6);
                cubes(117,123,0,66,6,6);
                cubes(57,48,0,6,72,6);
        }
}
```

```
void eater()
{
        glColor4f(1.000, 0.271, 0.000,0.8);
        cubes(px,py,pz,6,6,6);
}

void specialfood()
{
  if(foodcount==2)
  {
        glMatrixMode(GL_MODELVIEW);
        glPushMatrix();
        axis=0;
        glTranslatef(foodloc[foodcount][0],foodloc[foodcount][1],foodloc[foodcount]
[2]);
        glRotatef(theta[axis],0.0,1.0,1.0);
        glutSolidCube(6);
        glPopMatrix();

  }
  else if(foodcount==5)
  {
        glMatrixMode(GL_MODELVIEW);
        glPushMatrix();
        axis=1;
        glTranslatef(foodloc[foodcount][0],foodloc[foodcount][1],foodloc[foodcount]
[2]);
        glRotatef(theta[axis],1.0,0.0,1.0);
        glutSolidCube(6);
        glPopMatrix();
  }
  else
  {
        glMatrixMode(GL_MODELVIEW);
        glPushMatrix();
        axis=2;
        glTranslatef(foodloc[foodcount][0],foodloc[foodcount][1],foodloc[foodcount]
[2]);
        glRotatef(theta[axis],1.0,1.0,1.0);
        glutSolidCube(6);
        glPopMatrix();
  }
}

void food()
{
        glColor4f(1.0,1.0,0.0,0.7);
        if(foodover==0)
        {
                if((foodcount!=2)&&(foodcount!=5)&&(foodcount!=8))
                {
                        cubes(foodloc[foodcount][0], foodloc[foodcount][1],
foodloc[foodcount][2],4,4,4);
                }
                else
                        specialfood();
        }
}

void calchighscore()
{
        if(gamerestart==0)
        {
                highscore=score;
        }
        else if(gamerestart==1)
        {
                highscore=MAX(highscore,score);
        }
}

void spinfood()
{
        theta[axis]+=6.0;
        if(theta[axis]>360.0)theta[axis]-=360.0;

        if(clockstart==1)
        {
          end=clock();
          timer=(end-start)/CLOCKS_PER_SEC;
        }
        if(timer!=0)
        {
                if(foodcount!=0) score=((int)points*foodcount*1000)/((int)timer);
        }
  glutPostRedisplay();
}

void display()
{
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glClearColor(1.0,1.0,1.0,1.0);
        glMaterialfv(GL_FRONT,GL_AMBIENT,amb);
        glMaterialfv(GL_FRONT,GL_DIFFUSE,diff);
        glMaterialfv(GL_FRONT,GL_SPECULAR,spec);
        glMaterialfv(GL_FRONT,GL_SHININESS,shininess);
        glColorMaterial(GL_FRONT,GL_DIFFUSE);
        glLightfv(GL_LIGHT0,GL_POSITION,lightposition);
```

```
	glLightfv(GL_LIGHT0,GL_DIFFUSE,lightintensity);
	if(startscreen==0)
	{
		frontscreen();
	}
	else if(startscreen==1)
	{
		if(scale==0)
		{
			glScaled(0.5,0.5,0.5);
			scale=2;
		}
		if(scale==1)
		{
			glScaled(2.0,2.0,2.0);
			scale=3;
			scalecount=1;
		}
		if(scale==2)
		{
			glColor3f(1.0,0.0,1.0);
			output(220,135,"INSTRUCTIONS");
			output(220,110,"1. Use the up/down/left/right keys to move 
towards the food.");
			output(220,95,"2. You lose if you hit the boundary wall.");
			output(220,80,"3. Eating a food item will fetch you 10 
points.");
			output(220,65,"4. Every 3rd food item is a special one for 30 
points.");
			output(220,50,"5. You can press p to restart the game");

			output(-10,250,"PRESS ‘s’ TO START THE GAME");
		}
```

```
        if(rot==1&&rotcount<1)
        {
                glRotatef(-40,1,0,0);
                rot=0;
                ++rotcount;
                --antirotcount;
        }
        if(antirot==1&&antirotcount<1)
        {
                glRotatef(40,1,0,0);
                antirot=0;
                ++antirotcount;
                --rotcount;
        }
        if(scale==3)
        {
                sprintf(t1, "%d", highscore);
                output(180,150,"Highscore:");
                output(250,150,t1);
                sprintf(t2, "%d", timer);
                output(180,140,"Timer");
                output(250,140,t2);
                sprintf(t3, "%d", points);
                output(180,130,"Points");
                output(250,130,t3);
                sprintf(t4, "%d", score);
                output(180,120,"Score");
                output(250,120,t4);
        }
        if(startscreen==1)
        {
                glColor3f(1.0,1.0,0.0);
                glPointSize(10.0);
                food();
                glColor3f(0.0,0.0,1.0);
                glPointSize(18.0);
                eater();
                glColor3f(1.0,1.0,1.0);
                boundary();
                glutPostRedisplay();
        }
        if(gameend==1)
        {
                clockstart=0;
                calchighscore();
                glColor3f(1.0,0.0,0.0);
                output(180,60,"Oops, You Hit the Wall");
                if(score>=highscore)
                {
                    glColor3f(0.0,0.0,1.0);
                    output(180,50,"But You Made a Highscore!!");
                }
                output(180,40,"Press p to Restart the Game");
        }
        else if(win==1)
        {
                clockstart=0;
                calchighscore();
                glColor3f(1.0,0.0,0.0);
                output(180,60,"You Won");
                if(score>=highscore)
                {
                    glColor3f(1.0,0.0,0.0);
                    output(180,50,"Hurray!! You Made a Highscore");
                }
```

```
                        glColor3f(0.0,0.0,1.0);
                        output(180,40,"Press p to Restart the Game");
                }
                glFlush();
                  glutSwapBuffers();
        }
}

void themeboundary()
{
        if(themeflag==0)
                {
                        if((px==27 || px==105)&&(py>=51 && py<=129))
                        {gameend=1; foodcount=0;}
                        else if((py==51)&&((px>=33 && px<=87)||(px==99)))
                        {gameend=1; foodcount=0;}
                        else if((py==129)&&((px>=45 && px<=99)||(px==33)))
                        {gameend=1; foodcount=0;}
                }
                else if(themeflag==1)
                {
                        if((px==21)&&((py<=165)&&(py>=99)))
                        {gameend=1; foodcount=0;}
                        else if((py==123)&&((px>=87)&&(px<=148)))
                        {gameend=1; foodcount=0;}
                        else if((px==57)&&((py>=15)&&(py<=81)))
                        {gameend=1; foodcount=0;}
                }
}

void SpecialKey(int key, int x, int y)
{
  switch (key)
```

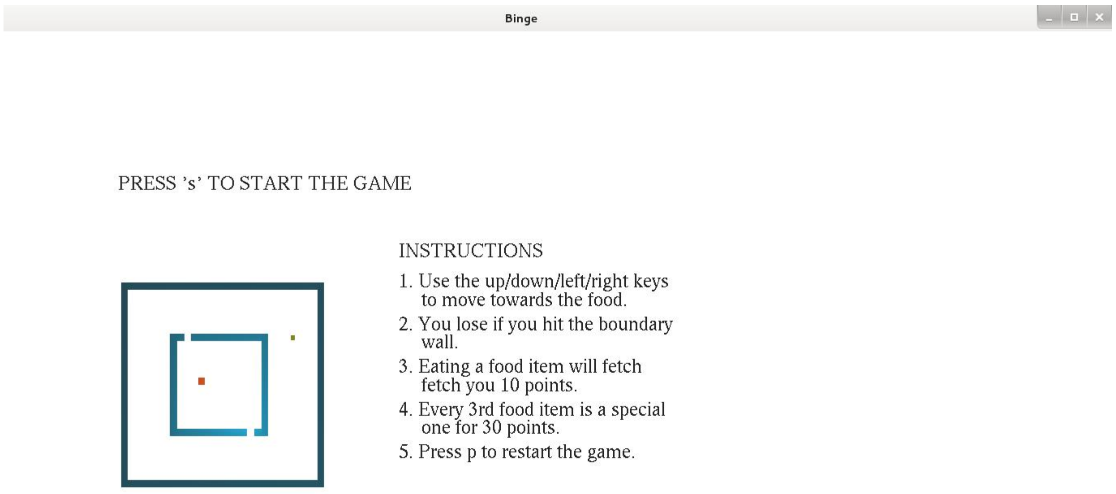
```
{
        case GLUT_KEY_UP:
                if(win==0 && gameend==0)
                py=py+6;
                glutPostRedisplay();
                if(py==171)
                {gameend=1; foodcount=0;}
                else themeboundary();
                break;

        case GLUT_KEY_DOWN:
                if(win==0 && gameend==0)
                py=py-6;
                glutPostRedisplay();
                if(py==9)
                {gameend=1; foodcount=0;}
                else themeboundary();
                break;

        case GLUT_KEY_LEFT:
                if(win==0 && gameend==0)
                px=px-6;
                glutPostRedisplay();
                if(px==-15)
                {gameend=1; foodcount=0;}
                else themeboundary();
                break;

        case GLUT_KEY_RIGHT:
                if(win==0 && gameend==0)
                px=px+6;
                glutPostRedisplay();
                if(px==153)
                {gameend=1; foodcount=0;}
                else themeboundary();
                break;
}

if((foodcount<9)&&((px==foodloc[foodcount][0])&&(py==foodloc[foodcount][1])))
   {
        foodcount++;
        if(foodcount==3||foodcount==6||foodcount==9)
        {
                points+=50;
        }
        else points+=10;
   }
   if(foodcount==9) {win=1; foodover=1;}
}

void restart()
{
        gamerestart=1;
        score=0;
        timer=0;
        points=0;
        gamestart=1;
        clockstart=1;
        start=clock();
        foodcount=0;
        foodover=0;
        gameend=0;
        win=0;
        px=51.0; py=93.0; pz=0.0;
        glutPostRedisplay();
}
```

```
void keyboard(unsigned char ch,int x, int y)
{
        if(ch=='r') rot=1;
        if(ch=='a')antirot=1;
        if(ch=='s'&& scalecount==0)
        {
                scale=1;
                gamestart=1;
                clockstart=1;
                start=clock();
                glutPostRedisplay();
        }
        if(startscreen==0 && ch==13) {startscreen=1;lighting=1;}
        if(ch=='p')
        {
                restart();
        }
}

void myreshape(int w, int h)
{
        glViewport(0,0,w,h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if(w<=h)
                glOrtho(-25.0,175.0,-2.0*(GLfloat)h/(GLfloat)w,
190.0*(GLfloat)h/(GLfloat)w,-200.0,200.0);
        else
                glOrtho(-25.0*(GLfloat)w/(GLfloat)h,  175.0*(GLfloat)w/(GLfloat)h,-
2.0,190.0,-200.0,200.0);
  glMatrixMode(GL_MODELVIEW);
  glutPostRedisplay();
```

```
}

void lightED()
{
        if(lighting==1)
        {
                glEnable(GL_LIGHTING);
                glEnable(GL_LIGHT0);
                glShadeModel(GL_SMOOTH);
                glEnable(GL_NORMALIZE);
                glEnable(GL_COLOR_MATERIAL);
        }
        if(lighting==0)
        {
                glDisable(GL_LIGHTING);
                glDisable(GL_LIGHT0);
                glShadeModel(GL_SMOOTH);
                glDisable(GL_NORMALIZE);
                glDisable(GL_COLOR_MATERIAL);
        }
}

void topmenu(int id)
{
        switch(id)
        {
        case 1:rot=1;break;
        case 2:antirot=1;break;
        case 3:themeflag=0;break;
        case 4:themeflag=1;break;
        case 5:restart();break;
        case 6:exit(0);break;
        case 7:lighting=0;lightED();break;
        case 8:lighting=1;lightED();break;
        }
}


int main(int argc,char **argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
        glutInitWindowSize(600,600);
        glutCreateWindow("Binge");
        glutReshapeFunc(myreshape);
        glutDisplayFunc(display);
        glutIdleFunc(spinfood);
        glutSpecialFunc(SpecialKey);
        glutKeyboardFunc(keyboard);
        glEnable(GL_DEPTH_TEST);
        lightED();
        thememenu=glutCreateMenu(topmenu);
        glutAddMenuEntry("Theme 1",3);
        glutAddMenuEntry("Theme 2",4);
        glutCreateMenu(topmenu);
        glutAddSubMenu("Themes",thememenu);
        glutAddMenuEntry("Rotate Forwards",1);
        glutAddMenuEntry("Rotate Backwards",2);
        glutAddMenuEntry("Disable Lighting",7);
        glutAddMenuEntry("Enable Lighting",8);
        glutAddMenuEntry("Restart",5);
        glutAddMenuEntry("Quit",6);
        glutAttachMenu(GLUT_RIGHT_BUTTON);
        glutMainLoop();
        return 0;
}
```

CHAPTER 4

RESULTS

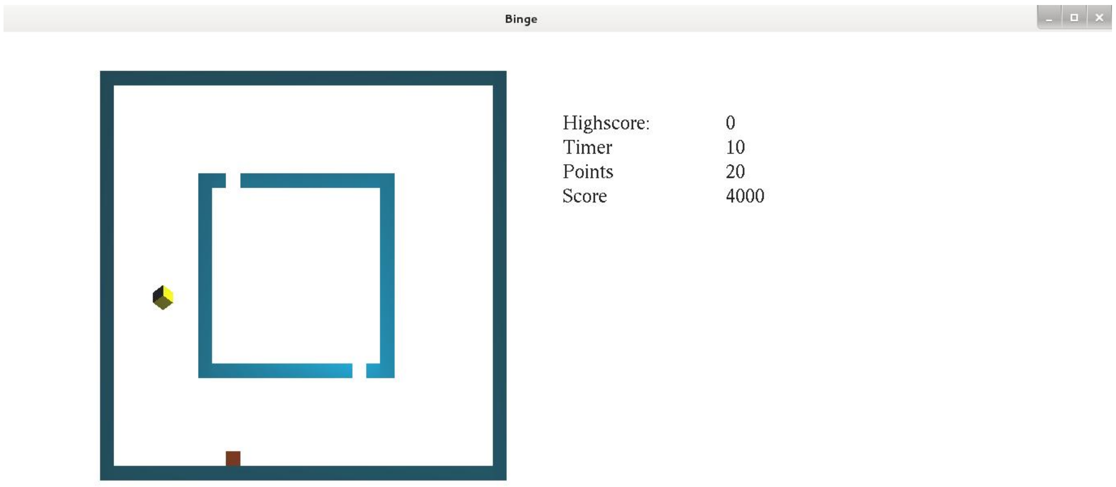


PRESS 's' TO START THE GAME

INSTRUCTIONS

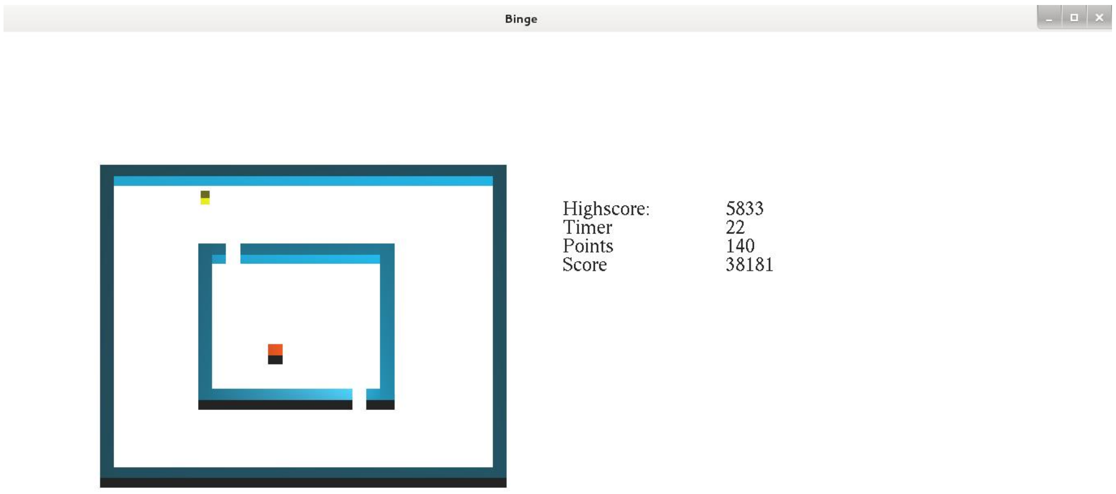
1. Use the up/down/left/right keys to move towards the food.
2. You lose if you hit the boundary wall.
3. Eating a food item will fetch fetch you 10 points.
4. Every 3rd food item is a special one for 30 points.
5. Press p to restart the game.

Snapshot 4.1: Instructions on how to play.



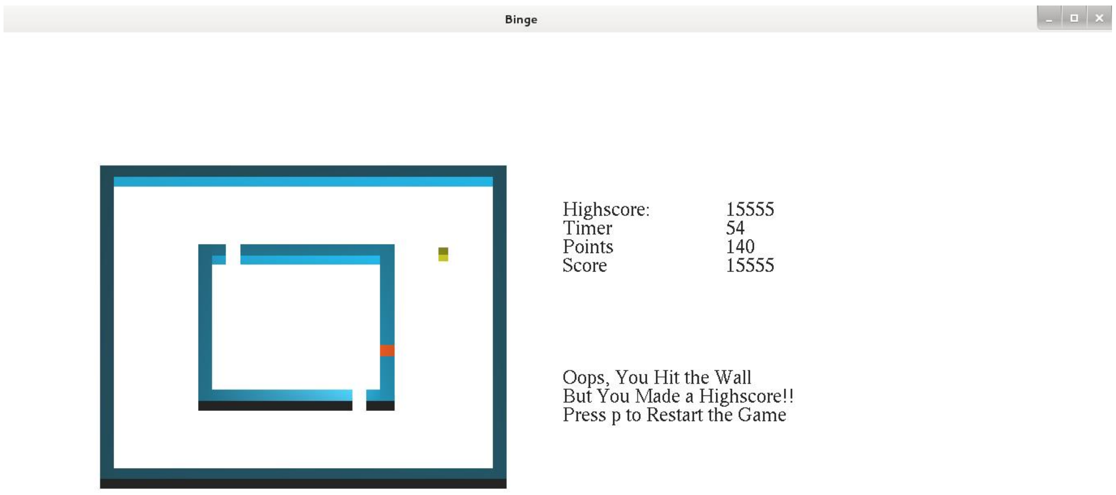
Highscore:	0
Timer	10
Points	20
Score	4000

Snapshot 4.2: Game started.



Highscore: 5833
Timer 22
Points 140
Score 38181

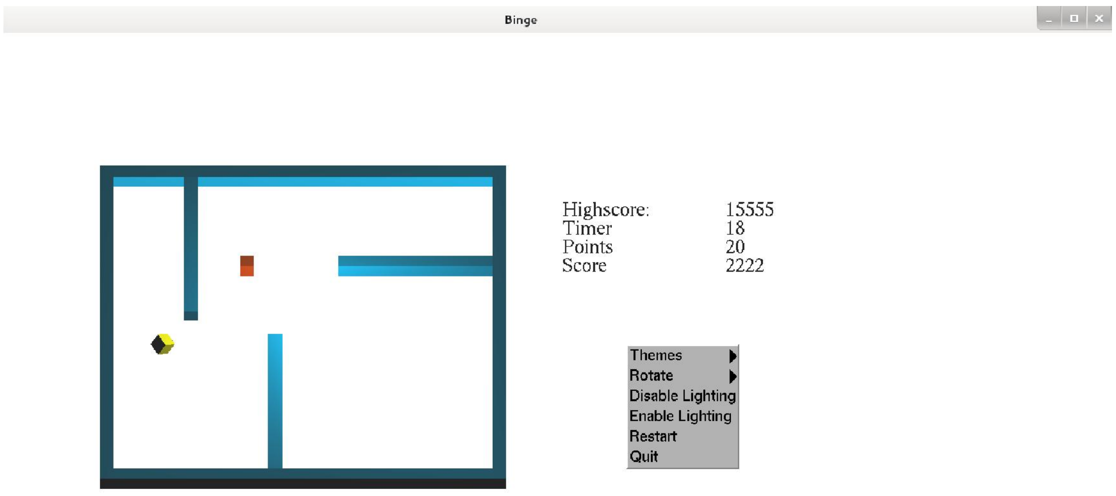
Snapshot 4.3: Rotated board to see lighting effect. Notice high score from previous game.



Highscore: 15555
Timer 54
Points 140
Score 15555

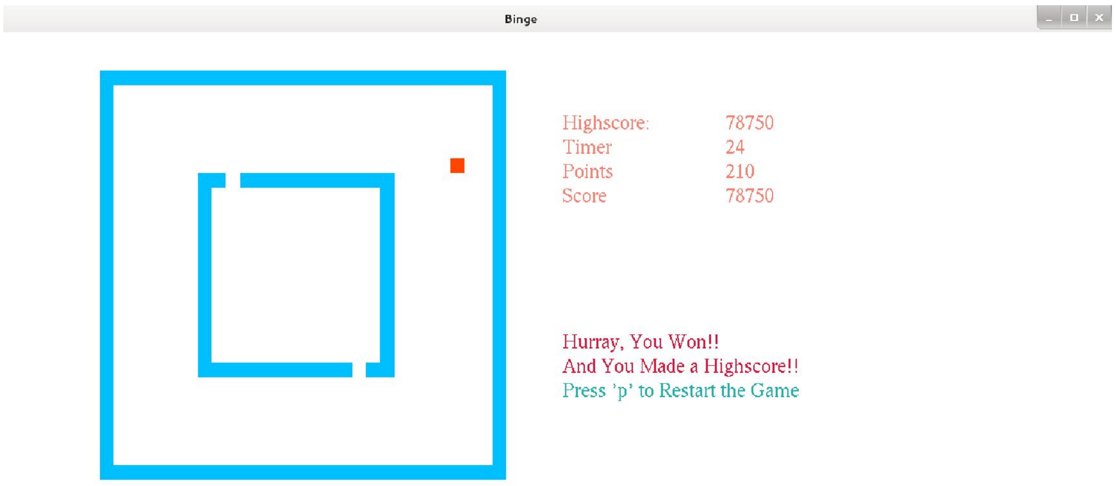
Oops, You Hit the Wall
But You Made a Highscore!!
Press p to Restart the Game

Snapshot 4.4: On hitting the boundary, but making a high score.



```
Highscore:      15555
Timer           18
Points          20
Score           2222
```

Snapshot 4.5: Theme 2 selected from menu.



```
Highscore:      78750
Timer           24
Points          210
Score           78750

Hurray, You Won!!
And You Made a Highscore!!
Press 'p' to Restart the Game
```

Snapshot 4.6: Lighting disabled and game won.

CHAPTER 5

CONCLUSION

5.1 Overview

An attempt has been made to develop an OpenGL graphics package which meets all the necessary requirements that were set out. The aim in developing this project was to depict the effect of lighting on and objects and how the change in the object material changes the way in which the lights appear on the object.

This game was designed using Open GL application software by applying the skills learnt in class, and in doing so, to understand the algorithms and techniques. It is user friendly and fun to play. The interfaces are mouse and keyboard driven and hence even a user with minimal knowledge of these devices can play the game. I finally conclude that this graphics package satisfies all requirements and provides good entertainment.

5.2 Future Enhancements

These are the features that are planned to be supported in the future

- Include more themes with color change options
- Allow rotation of the board along other axis as well
- Support for transparency of layers

CHAPTER 6

BIBLIOGRAPHY

- **Edward Angel**, “Interactive Computer Graphics”,5th edition, Pearson Education,2005.
- **Donald D Hearn** and **M. Pauline Baker,** “ Computer Graphics with OpenGL”, 3rd Edition “OpenGL Programming Guide”, Addision-Wesley Publishing Company.