Visvesvaraya Technological University
Belgaum, Karnataka-590 018

A Computer Graphics mini Project report
On

"LUDO GAME"

submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE & ENGINEERING

18CSL67- Computer Graphics Laboratory with mini project

Submitted by

MOHAMMED SHAHZADUL QUADRI	**1HK20CS093**
MOHAMMED TAMHEED SHARIFF	**1HK20CS095**

Under the Guidance of

Prof. Kusuma.K
Assistant Professor
Department of CSE, HKBKCE

Department of Computer Science & Engineering
HKBK College of Engineering
No.22/1, Opp., Manyata Tech Park Rd, Nagavara, Bengaluru, Karnataka 560045.
Approved by AICTE & Affiliated by VTU

2022-23

HKBK College of Engineering

No.22/1, Opp., Manyata Tech Park Rd, Nagavara, Bengaluru, Karnataka 560045.
Approved by AICTE & Affiliated by VTU

Department of Computer Science and Engineering

CERTIFICATE

Certified that the Computer Graphics Mini-Project (18CSL67) entitled **"Ludo Game",** carried out by **MOHAMMED SHAHZADUL QUADRI (1HK20CS093) and MOHAMMED TAMHEED SHARIFF (1HK20CS095)** is a bonafide student of **HKBK College of Engineering**, in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2022 – 23**. It is certified that all corrections/suggestions indicated for **Computer Graphics Mini-Project (18CSL67)** have been incorporated in the Report deposited in the departmental library. This report has been approved as it satisfies the academic requirements in respect of **Computer Graphics Mini-Project(18CSL67)** prescribed for the said Bachelor of Engineering.

Prof. Kusuma.K	**Dr. Smitha Kurian**	**DR. Tabassum Ara**
Guide	HOD - CSE	Principal –HKBKCE

External Viva

Name of the examiners	**Signature with date**
1._____	_____
2._____	_____

ACKNOWLEDGEMENT

I would like to express my regards and acknowledgement to all who helped me in completing this Computer Graphics Mini-Project successfully

First of all I would take this opportunity to express my heartful gratitude to the personalities of HKBK college of engineering, **Mr CM Ibrahim**, **Chairman, HKBKGI** and **Mr. Faiz Mohammed**, **Director, HKBKGI** for providing facilities throughout the course.

We would like to express our thanks to the Principal **DR. Tabassum Ara**, for her encouragement that motivated us for the successful completion of Project Work.

We wish to express our gratitude to **DR. Smitha Kurian.**, Professor and Head of Department of Computer Science & Engineering for providing such a healthy environment for the successful completion of Project work.

We express my heartfelt appreciation and gratitude to my Guide, **Prof. Kusuma.k**, Professor of Computer Science and Engineering, HKBK College of Engineering, Bangalore, for his/her intellectually- motivating support and invaluable encouragement during my Computer Graphics Mini-Project.

We would also like to thank all other teaching and technical staffs of Department of Computer Science and Engineering, who have directly or indirectly helped us in the completion of this Project Work. And lastly,we would hereby acknowledge and thank our parents who have been a source of inspiration and also instrumental in the successful completion of this project.

MOHAMMED SHAHZADUL QUADRI (1HK20CS093)
MOHAMMED TAMHEED SHARIFF (1HK20CS095)

ABSTRACT

The project aims to develop a computer-based version of the popular board game Ludo using the OpenGL graphics library. Ludo is a strategy-based board game that involves rolling dice and moving tokens across a board to reach a specific destination. By leveraging the capabilities of OpenGL, the game will provide an interactive and visually appealing user interface.

The project will involve several key components, including designing the game board, implementing the game rules, creating animated game tokens, and developing user interfaces for player interactions. The OpenGL library will be utilized to render and display the game board, tokens, and various graphical elements, resulting in an immersive gaming experience.

The implementation of Ludo in OpenGL will involve utilizing graphics primitives such as lines, circles, and textures to construct the game board and tokens. The dice rolling mechanism will be simulated using random number generation, and the movement of tokens will be controlled based on the game rules. The user interface will include features like interactive menus, player turn indicators, and visual cues for player actions.

Throughout the project, various aspects of computer graphics and game development will be explored, including 2D rendering, animation, event handling, and user interface design. By the end of the project, a fully functional Ludo game will be developed, providing players with an engaging and enjoyable digital adaptation of the classic board game.

TABLE OF CONTENTS

LIST OF FIGURES:

LIST OF SNAPSHOTS:

CHAPTER 1

INTRODUCTON

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.
- There is virtually no area in which graphical displays cannot be used to some advantage.
- 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.

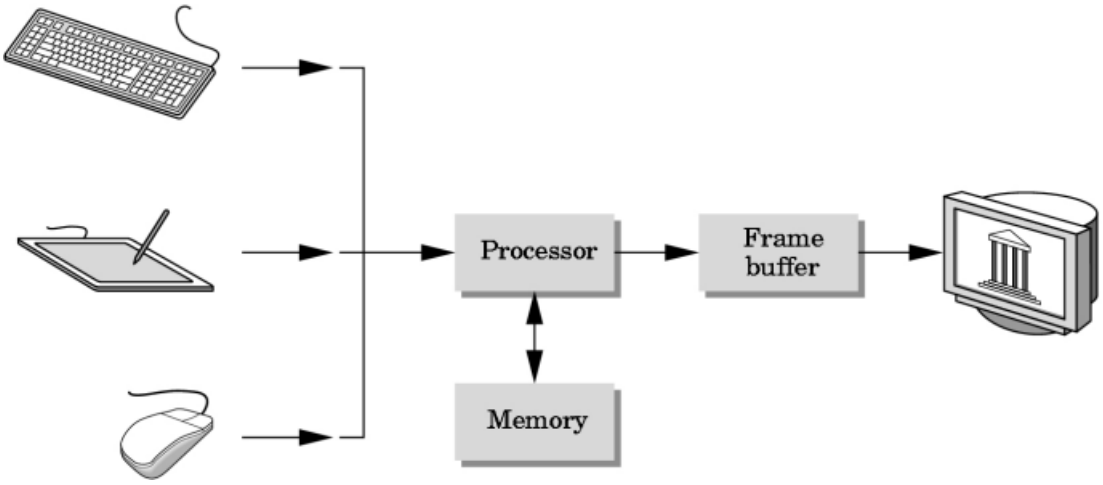

Fig 1.1 Components Of Computer Graphics System

1.2 OpenGL Technology

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/NT and 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, OPENStep, 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.

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

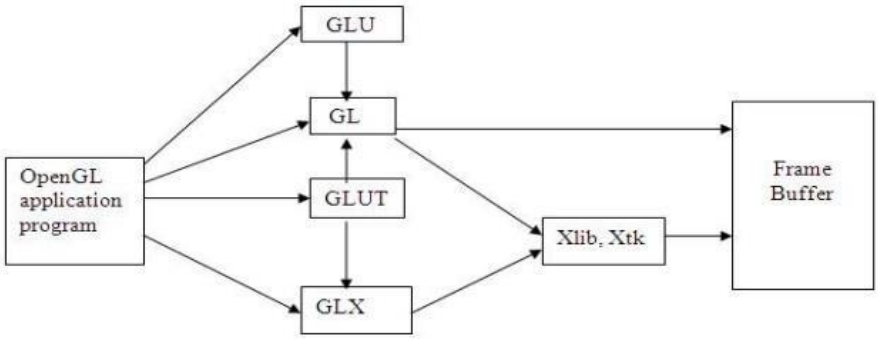

Fig 1.2 Open Gl Block Diagram

CHAPTER 2

LITERATURE SURVEY

Chapter 2 LITERATURE SURVEY

(Computer graphics) started with the display of data on hardcopy plotters and cathode ray tube screens soonafter the introduction of computer themselves. It includes the creation, storage, and manipulation of models and images of objects. These models include physical, mathematical, engineering, architectural, and even conceptualor abstract structures, natural phenomena, and so on. Computer Graphics today is largely interactive-the user controls the contents, structure, and appearance of objects and their displayed images by using input devices, suchas keyboard, mouse or touch sensitive panel on the screen. Bitmap graphics is used for user-computer interaction.A Bitmap is an ones and zeros representation of points (pixels, short for 'picture elements') on the screen. Bitmapgraphics provide easy-to-use and inexpensive graphics-based applications.

The concept of 'desktop' is a popular metaphor for organizing screen space. By means of a window manager, the user can create, position, and resize rectangular screen areas, called windows, that acted as virtual graphics terminals, each running an application. This allowed users to switch among multiple activities just by pointing at the desired window, typically with the mouse. Graphics provides one of the most natural means of communicating with the computer, since our highly developed 2D and 3D pattern – recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. In many designs, implementation, and construction processes, the information pictures can give is virtually indispensable.

2.1 Interactive and Non-Interactive graphics:

Computer graphics is the creation and manipulation of pictures with the aid of computers. It is divided into two broad classes:

- ❖ Non-Interactive Graphics.
- ❖ Interactive Graphics.

2.1.1 Non-Interactive graphics:

This is a type of graphics where observer has no control over the pictures produced on the screen. It is also called as Passive graphics

2.1.2 Interactive Graphics:

This is the type of computer graphics in which the user can control the pictures produced. It involves two-way communication between user and computer. The computer upon receiving signal from the input device can modify the displayed picture appropriately. To the user it appears that the picture changes instantaneously in response to his commands. The following fig. shows the basic graphics system:

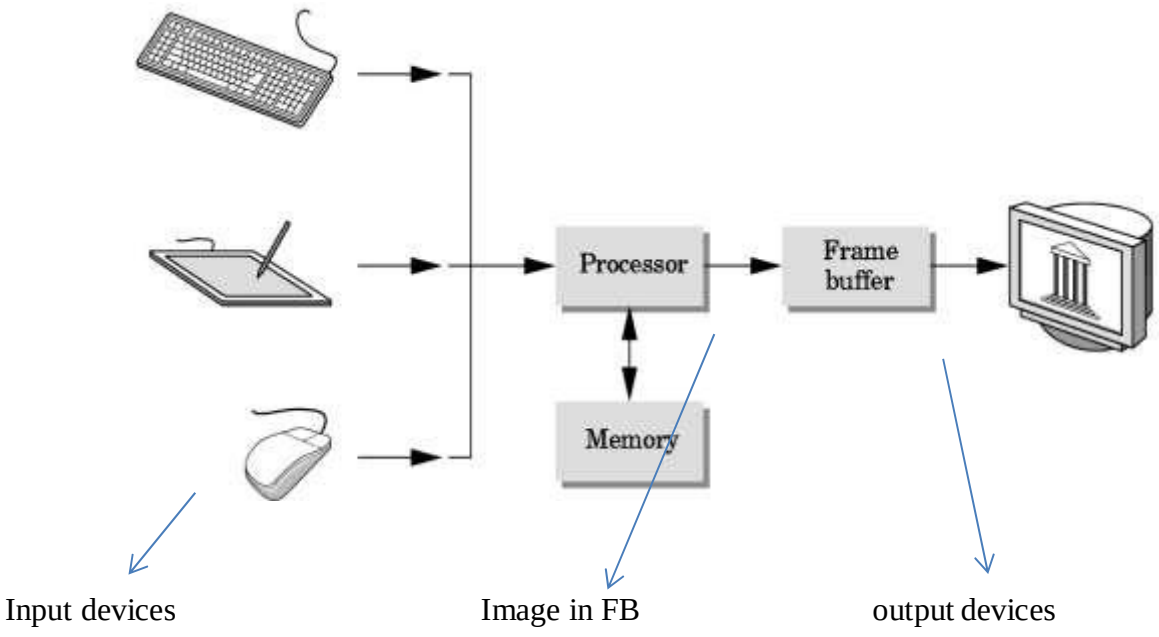

Fig 2.1: Basic Graphics System.

2.2 About OpenGL:

OpenGL is an open specification for an applications program interface for defining 2D and 3D objects. The specification is cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. It renders 3D objects to the screen, providing the same set of instructions on different computers and graphics adapters. Thus it allows us to write an application that can create the same effects in any operating system using any OpenGL-adhering graphics adapter.

In Computer graphics, a 3-dimensional primitive can be anything from a single point to an ‘n’ sided polygon. From the software standpoint, primitives utilize the basic 3-dimensional rasterization algorithms such as Bresenham's line drawing algorithm, polygon scan line fill, texture mapping and so forth.

OpenGL is a low-level, procedural API, requiring the programmer to dictate the exact steps required to render a scene. OpenGL's low-level design requires programmers to have a good knowledge of the graphics pipeline, but also gives a certain amount of freedom to implement novel rendering algorithms.

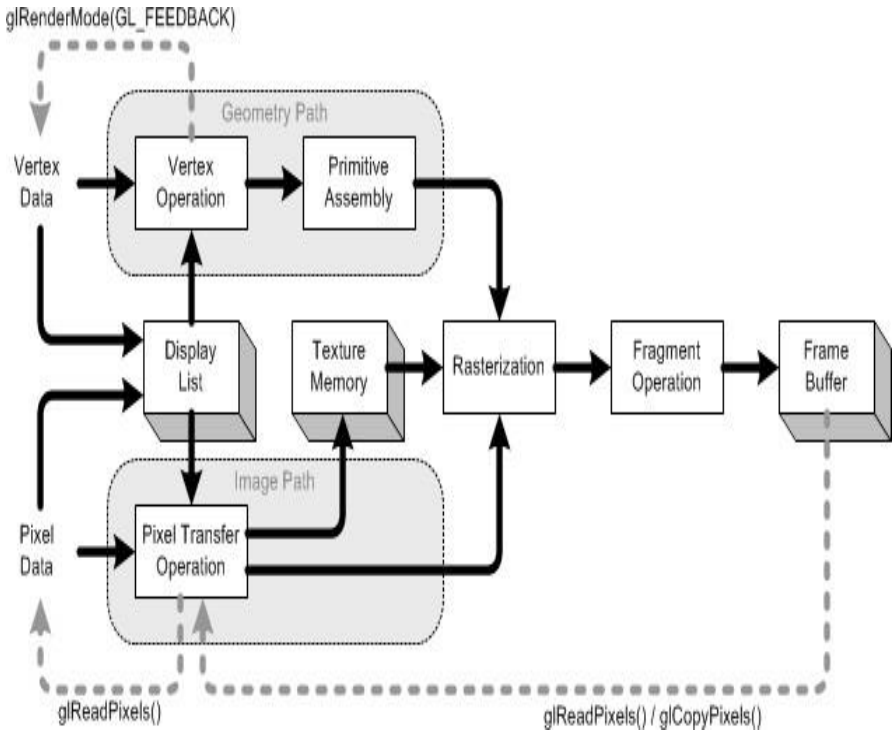

Fig 2.2: The OpenGL rendering pipeline.

2.3 Advantages of OpenGL

- OpenGL is an industry-standard guided by an independent consortium, the OpenGL Architecture Review Board oversees the OpenGL specification. It makes OpenGL the only truly open, vendor-neutral, and multi-platform graphics standard.
- OpenGL offers scalability with applications that can execute on systems running the gamut from consumer electronics to PCs, workstations, and supercomputers. Consequently, applications can scale to any class of machine that the developer chooses to target.

CHAPTER 3

SYTEM REQUIREMENTS SPECIFICATIONS

Chapter 3 SYSTEM REQUIREMENTS SPECIFICATION

A software requirement definition is an abstract description of the services which the system should provide, and the constraints under which the system must operate. It should only specify the external behaviour of the system.

3.1 Functional Requirements

Functional Requirements define the internal working of the software. The following conditions must be taken care of: The ability to perform correct operations when corresponding keys are pressed.

3.2 Non-functional Requirements

Non-functional requirements are requirements which specify criteria that can be used to judge the operation of the system, rather than specific behaviours. This should be contrasted with functional requirements that specify specific behaviour or functions. Typical non-functional requirements are reliability and scalability. Non- functional requirements are “constraints”, “quality attributes” and “quality of service requirements”.

Types of non-functional requirements

Volume: Volume of a system (the total space occupied) varies depending on how the component assemblies are arranged and connected.

Reliability: System reliability depends on component reliability but unexpected interactions can cause new types of failure and therefore affect the reliability of the system.

Security: The security of the system (it’s ability to resist attacks) is a complex property that cannot be easily measured. Attacks maybe devised that were not anticipated by the system designers and may use default built-in safeguards.

Repairability: This property reflects how easy it is to fix a problem with the system once it has been discovered. It depends on being able to diagnose the problem, access the components that are faulty and modify or replace these components.

Usability: This property reflects how easy it is to use the system. It depends on the technical system components, its operators and its operating environment.

3.3 Software Requirements:

Operating System : Windows

IDE : VS Express

Coding Language : C++

3.4 Hardware Requirements

System : Intel® Core™ i3 – 6006U CPU @ 2.00GHz

Hard Disk : 30 GB or above

Monitor : 15 VGA color

RAM : 256 MB or above

3.5 Introduction to Environment

OpenGL is a software interface to graphics hardware. This interface consists of about 120 distinct commands, which you use to specify the objects and operations needed to produce interactive three - dimensional applications.

OpenGL is designed to work efficiently even if the computer that displays the graphics you create isn't the computer that runs your graphics program. This might be the case if you work in a networked computer environment where many computers are connected to one another by wires capable of carrying digital data. In this situation, the computer on programs can work across a network even if the client and server are different kinds of computers. If an OpenGL program isn't running which your program runs and issues OpenGL drawing commands is called the client, and the computer that receives those commands and performs the drawing is called the server. The format for transmitting OpenGL commands (called the protocol) from the client to the server is always the same, so OpenGL across a network, then there's only one computer, and it is both the client and the server.

CHAPTER 4

ANALYSIS

Chapter 4 ANALYSIS

4.1Pseudo Code:

```
#include "stdafx.h"
#include "glut.h"
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int isIdle=0,ww,wh,s;
int in_about=1;
int in_guide=1;
int fontType;
void *font;
int
f1,f2,g1,g2,k1,k2,l1,l2py
1,ply2,h1,h2,h3,h4,s1,s2,
u,v,r1,op,w,dice1,enter;
int
i,ch1,ch,p1,p2,ra1,z,z1,za
1;
float a[100][100]={{-
0.4,-0.325},{-0.1375,-
0.7275},{-0.1375,-
0.61},{-0.1375,-
0.4925},{-0.1375,-
0.375},{-0.1375,-
0.2575},{-0.2575,-
0.14},{-0.375,-0.14},{-
0.495,-0.14},{-0.6110,-
0.14},{-0.7275,-0.14},{-
0.84,-0.14},{-
0.84,0.005},{-
0.84,0.1375},{-
0.7275,0.1375},{-
0.6110,0.1375},{-
.495,.1375},{-
0.375,0.1375},{-
0.2575,0.1375},{-
0.1375,0.2575},{-
0.1375,0.375},{-
0.1375,0.4925},{-
0.1375,0.61},{-
0.1375,0.7275},{-
0.1375,0.845},{0,0.845},
{0.1375,0.845},{0.1375,
0.7275},{0.1375,0.61},{0
```

```
.1375,0.4925},{0.1375,0.
375},{0.1375,0.2575},{0.
2575,0.1375},{0.375,0.1
375},{.495,0.14},{0.6110
,0.1375},{0.7275,0.1375
},{0.84,0.1375},{0.84,0.0
05},{0.84,-
0.14},{0.7275,-
0.14},{0.6110,-
0.14},{.495,-
0.14},{0.375,-
0.14},{0.2575,-
0.14},{0.1375,-
0.2575},{0.1375,-
0.375},{0.1375,-
0.4925},{0.1375,-
0.61},{0.1375,-
0.7275},{0.1375,-
0.845},{0,-0.845},{0,-
0.7275},{0,-0.61},{0,-
0.4925},{0,-0.375},{0,-
0.2575}};
float b[100][100]={{-
0.4,-
0.325},{0.1375,0.7275},{
0.1375,0.61},{0.1375,0.4
925},{0.1375,0.375},{0.1
375,0.2575},{0.2575,0.1
375},{0.375,0.1375},{0.4
95,0.14},{0.6110,0.1375
},{0.7275,0.1375},{0.84,
0.1375},{0.84,0.005},{0.
84,-0.14},{0.7275,-
0.14},{0.6110,-
0.14},{.495,-
.14},{0.375,-
0.14},{0.2575,-
0.14},{0.1375,-
0.2575},{0.1375,-
0.375},{0.1375,-
0.4925},{0.1375,-
0.61},{0.1375,-
0.7275},{0.1375,-
0.845},{0,-0.845},{-
0.1375,-0.845},{-
0.1375,-0.7275},{-
0.1375,-0.61},{-0.1375,-
0.4925},{-0.1375,-
0.375},{-0.1375,-
0.2575},{-0.2575,
```

```
0.14},{-0.375,-0.14, {-
.495,-0.14},{-0.6110,-
0.14},{-0.7275,-0.14},{-
0.84,-0.14},{-
0.84,0.005},{-
0.84,0.1375},{-
0.7275,0.1375},{-
0.6110,0.1375},{-
.495,0.1375},{-
0.375,0.1375},{-
0.2575,0.1375},{-
0.1375,0.2575},{-
0.1375,0.375},{-
0.1375,0.4925},{-
0.1375,0.61},{-
0.1375,0.7275},{-
0.1375,0.845},{0,0.845},
{0,0.7275},{0,0.61},{0,0.
4925},{0,0.375},{0,0.257
5}};
float c[10][10]={{-0.5,-
0.7},{-0.7,-0.5}};
float
d[10][10]={{0.5,0.7},{0.
7,0.5}};
int
A[100][100]={{430,527}
,{630,608},{626,567},{6
28,528},{626,485},{629,
441},{585,397},{544,398
},{502,400},{463,397},{
420,395},{381,397},{380
,350},{377,304},{421,30
3},{460,304},{501,301},
{541,302},{585,300},{62
7,260},{627,221},{627,1
77},{627,138},{627,97},
{627,58},{676,56},{723,
54},{723,96},{725,138},
{724,176},{725,216},{72
3,258},{764,300},{809,3
03},{850,304},{890,304}
,{931,303},{972,303},{9
71,352},{973,398},{932,
396},{890,396},{851,395
},{811,395},{768,396},{
724,441},{725,484},{724
,523},{725,562},{725,60
6},{728,645},{677,645},
{677,608},{677,565},{67
```

```
7,522},{677,485},{677,4
42}};
int
B[100][100]={{853,104}
,{723,96},{725,138},{72
4,176},{725,216},{723,2
58},{764,300},{809,303}
,{850,304},{890,304},{9
31,303},{972,303},{971,
352},{973,398},{932,396
},{890,396},{851,395},{
811,395},{768,396},{724
,441},{725,484},{724,52
3},{725,562},{725,606},
{728,645},{677,645},{62
9,650},{630,608},{626,5
67},{628,528},{626,485}
,{629,441},{585,397},{5
44,398},{502,400},{463,
397},{420,395},{381,397
},{380,350},{377,304},{
421,303},{460,304},{501
,301},{541,302},{585,30
0},{627,260},{627,221},
{627,177},{627,138},{62
7,97},{627,58},{676,56},
{677,97},{677,137},{677
,177},{677,219},{677,26
1}};
void
display_inter_guide();
void arrow();
void display_enter1();
void init()
{
  glClearColor(1,0.8,0.5,
0.0);
glEnable(GL_DEPTH_T
EST);
glutInitDisplayMode(GL
UT_RGBA);
glutInitWindowPosition(
0,0);
glutInitWindowSize(135
5,703);
glutCreateWindow("HK
BK - LUDO - BOARD
GAME");
}
```

```
void myReshape(int w,int
h)
{
glViewport(0,0,w,h);
glMatrixMode(GL_PROJ
ECTION);
glLoadIdentity();
if(w<=h)
glOrtho(-1.0,1.0,-
1.0*(GLfloat)h/(GLfloat)
w,
2.0*(GLfloat)h/(GLfloat)
w,-20.0,20.0);
else
glOrtho(-
1.0*(GLfloat)w/(GLfloat)
h,
1.0*(GLfloat)w/(GLfloat
)h,-1.0,1.0,-20.0,20.0);
glMatrixMode(GL_MOD
ELVIEW);
glutPostRedisplay();
}
void dice(int y)
{
  glPointSize(5);
  glColor3f(1,1,1);
  glBegin(GL_QUADS);
  glVertex2f(-0.82,0.25);
  glVertex2f(-0.82,0.38);
  glVertex2f(-0.69,0.38);
  glVertex2f(-0.69,0.25);
  glEnd();
  glBegin(GL_QUADS);
  glVertex2f(-0.82,0.38);
  glVertex2f(-0.69,0.38);
  glVertex2f(-0.67,0.4);
  glVertex2f(-0.79,0.4);
  glEnd();
  glBegin(GL_QUADS);
  glVertex2f(-0.69,0.38);
  glVertex2f(-0.67,0.4);
  glVertex2f(-0.67,0.27);
  glVertex2f(-0.69,0.25);
  glEnd();
  glColor3f(0,0,0);
  glBegin(GL_LINE_LO
OP);
  glVertex2f(-0.82,0.38);
  glVertex2f(-0.69,0.38)
```

```
  glVertex2f(-0.67,0.4);
  glVertex2f(-0.79,0.4);
  glEnd();
  glBegin(GL_LINE_LO
OP);
  glVertex2f(-0.69,0.38);
  glVertex2f(-0.67,0.4);
  glVertex2f(-0.67,0.27);
glVertex2f(-0.69,0.25);
  glEnd();
  glColor3f(0,0,0);
  switch(y)
  {
  case 0:
  case 1:
  glBegin(GL_POINTS);
            glVertex2f(-
0.75,0.31);
                        glEnd();
                        break;
    case
2:glBegin(GL_POINTS);

  glVertex2f(-0.73,0.31);
          glVertex2f(-
0.77,0.31);
                        glEnd();
          break;
    case
3:glBegin(GL_POINTS);

  glVertex2f(-0.72,0.31);
          glVertex2f(-
0.755,0.31);

  glVertex2f(-0.79,0.31);
                        glEnd();
          break;
  case
4:glBegin(GL_POINTS);
glVertex2f(-0.72,0.285);
          glVertex2f(-
0.72,0.345);
            glVertex2f(-
0.78,0.285);
            glVertex2f(-
0.78,0.345);
                        glEnd();
          break;
  case5:glBegin(GL_POI
```

```
NTS);
            glVertex2f(-
0.72,0.285);
          glVertex2f(-
0.72,0.345);
          glVertex2f(-
0.75,0.31);
            glVertex2f(-
0.78,0.285);
            glVertex2f(-
0.78,0.345);
                      glEnd();
        break;
  case
6:glBegin(GL_POINTS);
            glVertex2f(-
0.72,0.29);
          glVertex2f(-
0.755,0.29);
          glVertex2f(-
0.79,0.29);
glVertex2f(-0.72,0.34);
            glVertex2f(-
0.755,0.34);
            glVertex2f(-
0.79,0.34);
                      glEnd();
         break;
  }
  glPointSize(25);
  glFlush();
}
void display_about(void)
{
  char *about="This mini-
Project build by students
of HKBK College of
Engineering is based on a
very ancient game
LUDO,using
OpenGL.\n\nINSTRUCT
IONS :\nInput can be
provided either from
Mouse or from
Keyboard.\nFor mouse
interaction right-click on
the screen and select the
required
option.\nKeyboard and
Mouse Interfaces are
```

```
explained in Instructions
Section.\n\n\n\n\n\n\n\n\t\
t\t 1HK20CS093 -
MOHAMMED
SHAHZADUL
QUADRI\n\t\t\t
1HK20CS095 -
MOHAMMED
TAMHEED SHARIFF";
enter=0;
  int i;
  float x=-1.8,y=0.9,z=0;

  if(in_about!=0)
  {
  in_about=1;
glColor3f(1,0,0);
  glRasterPos3f(x,y,z);
for(i=0;about[i]!='\0';i++)
  {
    if(about[i]=='\n')
    {   y-=0.08;

glRasterPos3f(x,y,z);
    }
    else
glutBitmapCharacter(font
,about[i]);
  }
 glPopMatrix();
glFlush();
  }
  }
void display(void)
{
glClear(GL_COLOR_B
UFFER_BIT|GL_DEPT
H_BUFFER_BIT);
  display_about();
}
void main(void)
{
  enter=1;
glutInitDisplayMode(GL
UT_SINGLE|GLUT_RG
B|GLUT_DEPTH);
   init();
glutCreateMenu(options)
;
glutAddMenuEntry("Abo
```

```
ut LUDO Project",1);
glutAddMenuEntry("Rul
es of our Game",2);
glutAddMenuEntry("Ga
me Mode or Restart",4);
glutAddMenuEntry("Quit
",glutMouseFunc(myMo
use);
 glutKeyboardFunc(key);
glutAttachMenu(GLUT_
RIGHT_BUTTON);
glutReshapeFunc(myRes
hape);
glutDisplayFunc(display)
;
font=GLUT_BITMAP_9
_BY_15;
   glutMainLoop();
}
```

4.2 Analysis:

Analyzing a Ludo game using OpenGL involves breaking down the game into its key components, implementing the necessary graphics and interactions, and managing the game logic. Here's a step-by-step guide on how you can approach the analysis and implementation of a Ludo game using OpenGL:

- Design the game board: Start by designing the Ludo game board using OpenGL primitives such as squares circles, and lines. You can create a 2D representation of the board using orthographic projection.
- Define the game pieces: Decide on the appearance of the game pieces (tokens) for each player. You can use simple shapes or textures to represent the tokens. Each player typically has four tokens of the same color.
- Handle mouse and keyboard inputs: Implement the logic to handle mouse and keyboard inputs for interacting with the game. For example, you can use mouse clicks to select and move tokens and keyboard inputs to roll the dice.
- Implement token movement: Define the rules for token movement based on the dice roll. Tokens move along the board's colored paths in a clockwise direction. Implement logic to calculate the token's position and animate its movement accordingly.
- Handle collisions and token stacking: When multiple tokens occupy the same square, they should stack on top of each other to represent multiple tokens. Implement collision detection to check if a token can move to a specific square and handle stacking accordingly.

- Implement dice rolling: Simulate the rolling of a dice using random number generation. Use keyboard inputs or a button press to trigger the dice roll and display the result on the screen.

CHAPTER 5
IMPLEMENTATION

Chapter 5 IMPLEMENTATION

Implementation is an act or instance of implementing something that is, the process of making something active or effective. Implementation must follow any preliminary thinking in order for something to actually happen. In the context of information technology, implementation encompasses the process involved in getting new software or hardware to operate properly in its environment. This program is implemented using various OpenGL functions which are shown below.

5.1 Built Functions Used:

Various functions used in this program.

- **glutInit**() : interaction between the windowing system and OPENGL is initiated.
- **glutInitDisplayMode**() : used when double buffering is required and depth information is required
- **glutCreateWindow**() : this opens the OPENGL window and displays the title at top of the window
- **glutInitWindowSize**() : specifies the size of the window
- **glutInitWindowPosition**() : specifies the position of the window in screen co-ordinates
- **glutKeyboardFunc**() : handles normal ascii symbols
- **glutSpecialFunc**() : handles special keyboard keys
- **glutReshapeFunc**() : sets up the callback function for reshaping the window
- **glutIdleFunc**() : this handles the processing of the background
- **glutDisplayFunc**() : this handles redrawing of the window
- **glutMainLoop**() : this starts the main loop, it never returns
- **glViewport**() : used to set up the viewport
- **glVertex3fv**() : used to set up the points or vertices in three dimensions
- **glColor3fv**() : used to render color to faces
- **glFlush**() : used to flush the pipeline
- **glutPostRedisplay**() : used to trigger an automatic redrawal of the object
- **glMatrixMode**() : used to set up the required mode of the matrix
- **glLoadIdentity**() : used to load or initialize to the identity matrix
- **glTranslatef**() : used to translate or move the rotation centre from one point to another in three dimensions
- **glRotatef**() : used to rotate an object through a specified rotation angle.

5.2 User Defined Functions:

The user-defined functions that are used and can be identified as:

- drawBoard(): This function is responsible for drawing the Ludo game board on the screen. It can use OpenGL primitives such as squares, circles, and lines to create the board's visual representation.
- drawToken(player, position): This function draws a token for a given player at a specific position on the board. It can use OpenGL primitives or textures to represent the tokens.
- moveToken(token, newPosition): This function handles the movement of a token to a new position on the board. It updates the token's position and triggers the necessary animations or transformations to visually represent the movement.
- checkCollision(position): This function checks if there is a collision between tokens at a specific position on the board. It can be used to determine if a token can move to a particular square or if stacking is required.
- rollDice(): This function simulates the rolling of a dice in the Ludo game. It generates a random number between 1 and 6 to determine the dice roll result.
- updateGameState(): This function updates the game state after each move. It keeps track of the current player, the number of turns, and the positions of all tokens. It can also check for win conditions to determine the game's outcome.
- displayText(text, position): This function displays text on the screen at a specific position. It can be used to show relevant game information such as player turns, dice roll results, and game status.
- handleInput(input): This function handles user input, such as mouse clicks and keyboard inputs, to interact with the game. It can detect and process input events and trigger the appropriate actions in the game.
- initializeGame(): This function initializes the Ludo game, setting up the initial game state, board, and player positions. It can be called at the start of the game or when starting a new game.

These are just some examples of user-defined functions you can include in your Ludo game implementation using OpenGL. Depending on your specific requirements and design choices, you may need to create additional functions to handle various aspects of the game.

CHAPTER 6

RESULTS

Chapter 6 RESULTS AND DISCUSSIONS

6.1 Discussions:

Discussing the Ludo Game in OpenGL with some key points:

- Setting up the game board: The first step is to create the Ludo board using OpenGL. You can represent the board as a square or a rectangle divided into smaller squares, with four starting areas, four home areas, and a central cross area.
- Game pieces and movement: Each player will have four game pieces or tokens. Implement the logic for moving the tokens across the board based on the dice roll. You can represent the tokens as colored circles or any other suitable graphics.
- Dice rolling: Implement a dice roll mechanism to generate random numbers between 1 and 6. You can represent the dice as a cube or any other suitable shape. When the player clicks on the dice, generate a random number and use it to determine the movement of the tokens.
- Game rules and logic: Implement the rules of Ludo, such as allowing tokens to enter the board, move across the board, capture opponent tokens, and reach the home area. Handle the logic for multiple players taking turns and ensuring fair gameplay.
- User interface: Create a user interface to display the game board, dice, and other relevant information. Allow users to interact with the game using mouse clicks or other input methods.
- Animation and visual effects: Add animations and visual effects to enhance the gameplay experience. For example, you can animate the movement of tokens across the board or add particle effects when tokens are captured.
- Scorekeeping and game progression: Implement a scoring system to keep track of each player's progress. Determine the winner based on the first player to get all tokens to the home area.

Remember to break down the implementation into smaller tasks and tackle them one by one. You can use libraries like GLFW, GLEW, and GLUT to handle window creation, user input, and other OpenGL-related tasks. Also consider using a suitable programming language like C++ or Python for the implementation.

6.2 Snapshots:

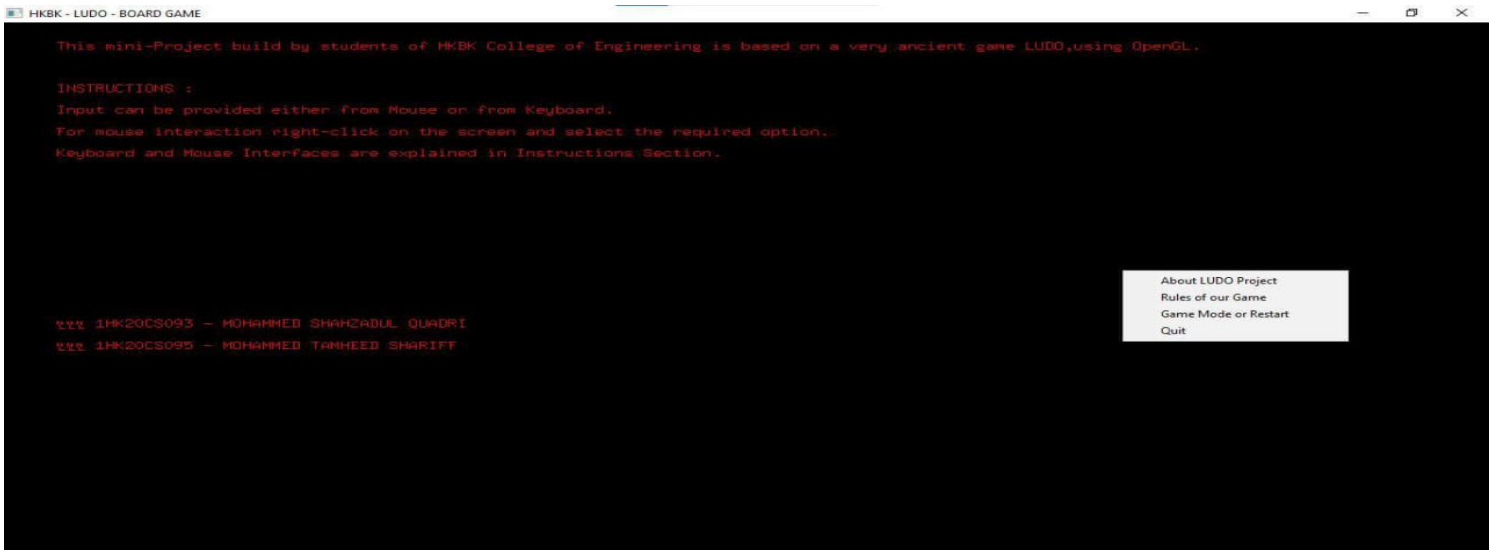

Fig 6.1– About The Game

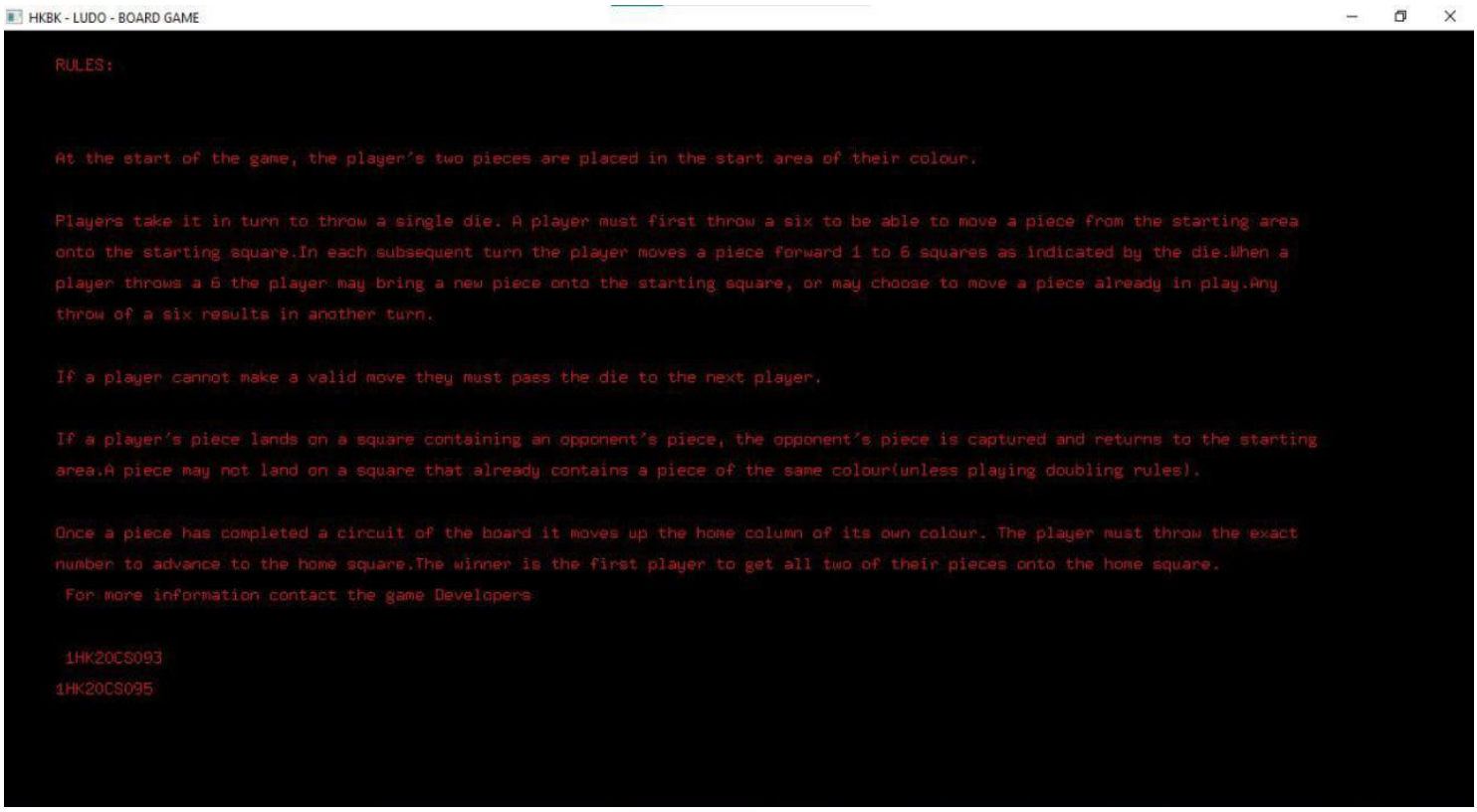

Fig 6.2 - Game Rules

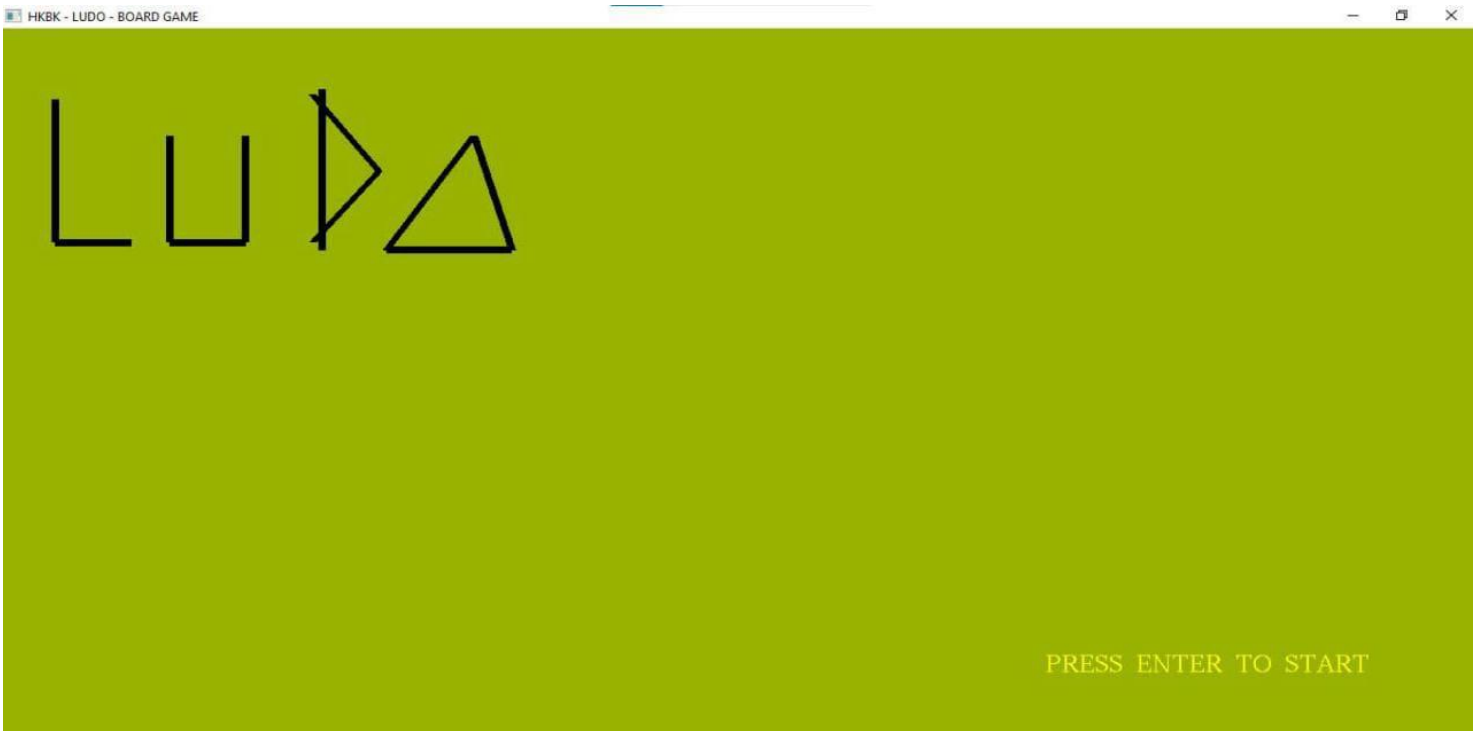

Fig 6.3 - Loading Game

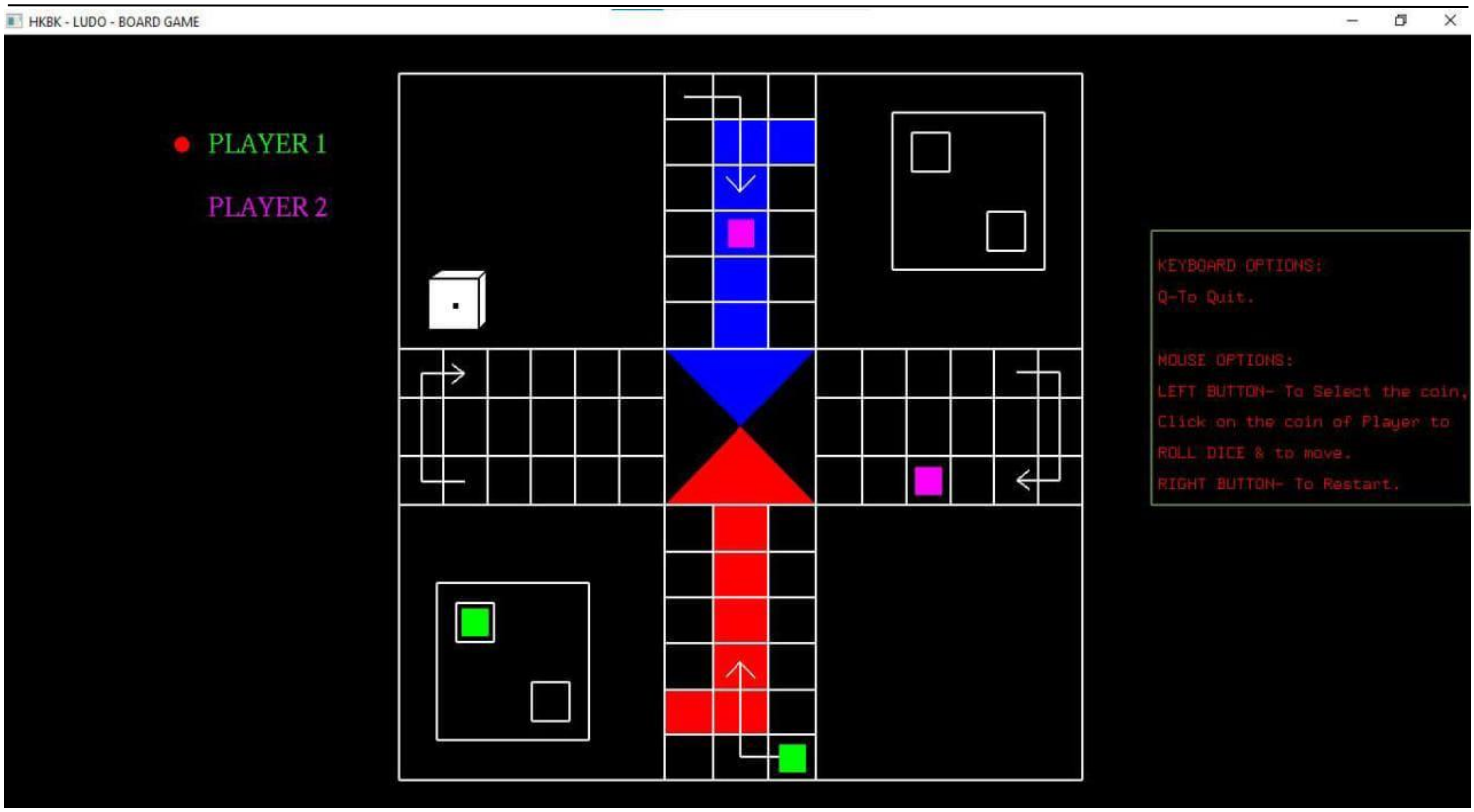

Fig 6.4 – Ludo Game

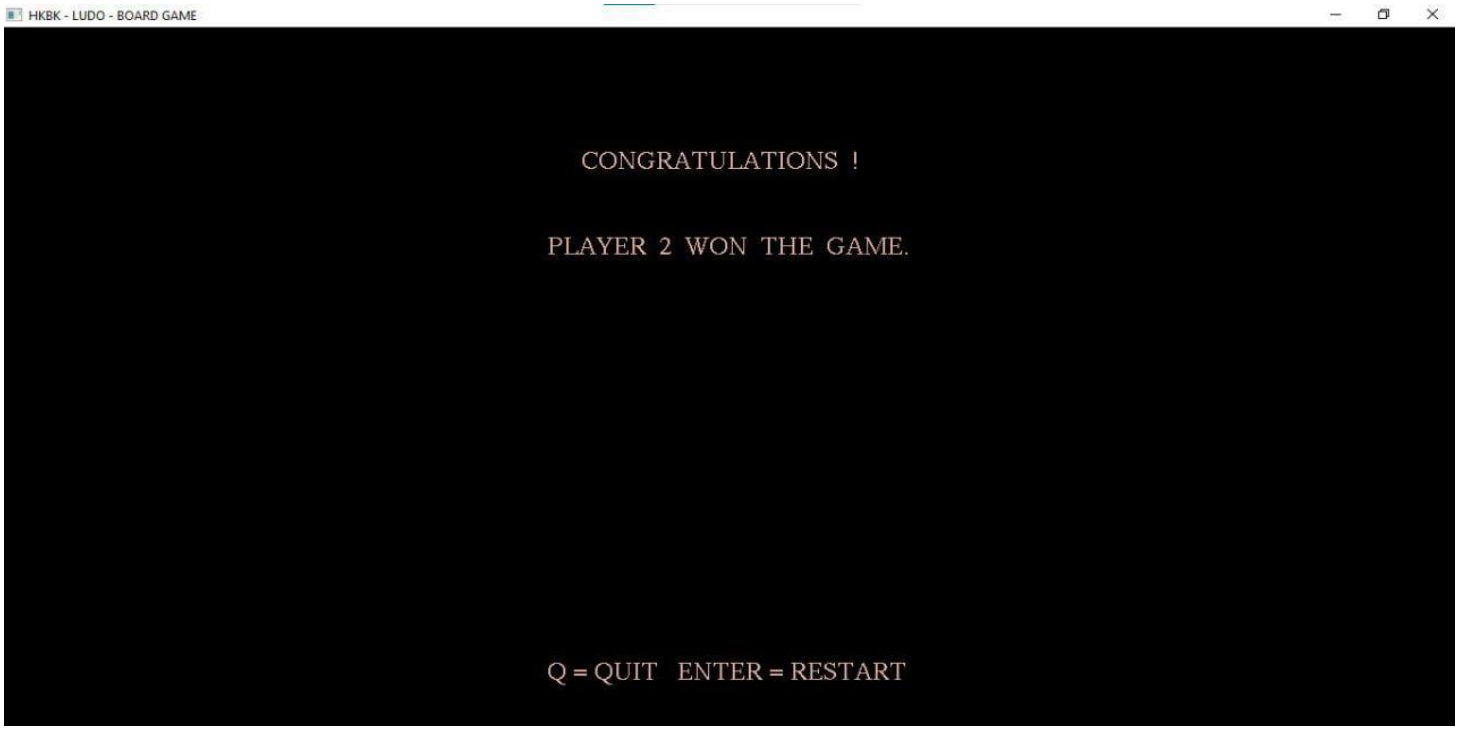

Fig 6.5 - Result Display

CHAPTER 7

CONCLUSIONS

Chapter 7 CONCLUSIONS AND FUTURE SCOPE

The project "Ludo Game " implements a basic animation of a moving s using the GLUT library. It incorporates interactive elements such as changing the background color, controlling the ship's movement, and triggering fireworks. User-defined functions are utilized to handle rendering different components of the animation. While the code snippet is incomplete and may require modifications, it serves as a foundation for creating more advanced graphical animations.

7.1 General Constraints:

- The code relies on the GLUT library, which is specific to the OpenGLenvironment.
- The code is designed for a console application, limiting its graphical capabilities.
- The code may not be portable across different operating systems due to itsreliance on Windows-specific headers.

7.2 Future Enhancements:

- Advanced Visual Effects: Enhance the graphics by incorporating advanced visual effects such as realistic water simulation, dynamic lighting and shadows, particle systems for realistic smoke or fire, and advanced shading techniques to create a more visually immersive experience.
- Interactive User Experience: Add more interactive elements to engage users, such as allowing direct control of the ship's movement using keyboard or mouse inputs, implementing interactive objects within the scene that respond to user interactions, or incorporating user-driven events or mini-games.
- Audio Integration: Include sound effects and background music to complement the animation and create a more immersive experience. Implementing audio cues for ship movements, fireworks, and other elements can greatly enhance the overall sensory experience.
- Performance Optimization: Optimize the code and rendering pipeline to improve performance, ensuring smooth animation even on lower-end systems or when rendering complex scenes. Implement techniques like level-of-detail rendering, occlusion culling, and efficient resource management to enhance performance.
- Extensibility and Modularity: Refactor the code to improve modularity and extensibility, making it easier to add new features or modify existing ones without disrupting the overall structure. This includes adopting design patterns, separating concerns, and implementing a well-structured architecture.

BIBLIOGRAPHY:

The Resources used to design and implement the Project Successively, the Following are some of the resources:-

TEXTBOOKS:-

1) INTERACTIVE COMPUTER GRAPHICS A TOP-DOWN APPROACH
   -By Edward Angel.

2) COMPUTER GRAPHICS,PRINCIPLES & PRACTICES
   - Foley van dam
   - Feiner hughes

LINKS / WEB REFERENCES:

1) http://jerome.jouvie.free.fr/OpenGl/Lessons/Lesson3.php
2) The website providing OpenGL lessons and tutorials, including Lesson 3, created by JeromeJouvie.

3) http://opengl.org
4) The official website of the OpenGL graphics API, providing documentation, resources, andsupport for OpenGL developers.

HKBK College of Engineering
No.22/1, Opp., Manyata Tech Park Rd, Nagawara, Bengaluru, Karnataka 560045.
Approved by AICTE & Affiliated by VTU

Department of Computer Science & Engineering

DEPARTMENT MISSION AND VISION

VISION

To advance the intellectual capacity of the nation and the international community by imparting knowledge to graduates who are globally recognized as innovators, entrepreneur and competent professionals.

MISSION

M - 1. To provide excellent technical knowledge and computing skills to make the graduates globally competitive with professional ethics.

M - 2. To involve in research activities and be committed to lifelong learning to make positive contributions to the society. Institute

INSTITUTE MISSION AND VISION

VISION

To empower students through wholesome education and enable the students to develop into highly qualified and trained professionals with ethics and emerge as responsible citizen with broad outlook to build a vibrant nation.

MISSION

M - 1. To achieve academic excellence through in-depth knowledge in science, engineering and technology through dedication to duty, innovation in teaching and faith in human values.

M - 2. To enable our students to develop into outstanding professionals with high ethical standards to face the challenges of the 21st century

M - 3. To provide educational opportunities to the deprived and weaker section of the society, to uplift their socio-economic status.

PROGRAM OUTCOMES(PO'S)

PO-1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO-2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO-3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO-4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO-5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO-6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO-7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO-8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO-9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO-10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO-11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO-12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO-1. Problem-Solving Skills: An ability to investigate and solve a problem by analysis, interpretation of data, design and implementation through appropriate techniques, tools and skills.

PSO-2. Professional Skills: An ability to apply algorithmic principles, computing skills and computer science theory in the modelling and design of computer-based systems.

PSO-3. Entrepreneurial Ability: An ability to apply design, development principles and management skills in the construction of software product of varying complexity to become an entrepreneur.

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

PEO-1. To provide students with a strong foundation in engineering fundamentals and in the computer science and engineering to work in the global scenario.

PEO-2. To provide sound knowledge of programming and computing techniques and good communication and interpersonal skills so that they will be capable of analyzing, designing and building innovative software systems.

PEO-3. To equip students in the chosen field of engineering and related fields to enable him to work in multidisciplinary teams.

PEO-4. To inculcate in students professional, personal and ethical attitude to relate engineering issues to broader social context and become responsible citizen.

PEO-5. To provide students with an environment for life-long learning which allow them to successfully adapt to the evolving technologies throughout their professional career and face the global challenges.