CHAPTER 1

1. INTRODUCTION

The overview of Computer Graphics is how it generates graphical display and how images are created. The aspects of OpenGL API and brief description about the project is as follows

1.1 About Computer Graphics

Computer graphics are graphics created using computers and, more generally, the representation and manipulation of image data by a computer.

The development of computer graphics has made computers easier to interact with, and better for understanding and interpreting many types of data. Developments in computer graphics had a profound impact on many types of media and have revolutionized animation, movies and the video game industry. Computer generated imagery can be categorized into several different types: 2D, 3D and animated graphics. As technology has improved, 3D computer graphics have become more commonly used graphic technology. Today, computers and computer-generated images touch many aspects of daily life.

1.2 About OpenGL

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.

OpenGL's main purpose is to render two- and three-dimensional objects into a framebuffer. These objects are described as sequences of vertices (which define geometric objects) or pixels (which define images). OpenGL draws primitives—points, line segments, or polygons—subject to several selectable modes. A sophisticated library that provides these features are OPENGL Utility Library (GLU).

OpenGL provides you with fairly direct control over the fundamental operations of two- and three-dimensional graphics. This includes specification of such parameters as transformation matrices, lighting equation, rotation matrices.

1.3 About Project

The mini project designed and implemented here is '**Snake and Ladders**'. This project involves implementation of a very old Indian Board Game i.e. Snakes and Ladders. It is played between two or more players on a gameboard having numbered, gridded squares. A number of "ladders" and "snakes" are pictured on the board, each connecting two specific board squares. The object of the game is to navigate one's game piece, according to die rolls, from the start (bottom square) to the finish (top square), helped or hindered by ladders and snakes respectively. The game is a simple race contest based on sheer luck, and is popular with young children.

Here we have developed mesh as base of our game board. We are mapping images to ensure proper graphics for enhanced gaming experience. User can exit game at any moment by pressing ESC key. We have kept multiple frames in order to change the users perspective of the game flow eventually.

CHAPTER 2

2. SYSTEM REQUIREMENTS

In this section, the various requirements that are essential for this mini project are specified. For the successful, efficient and problem free designing of any project or program using Computer Graphics, the system should meet some requirements.

2.1 Hardware Requirement

- Processor: Intel i3 Core
- Ram: 1 GB
- Hard Disk: 40 GB

2.2 Software Requirement

- Operating system: Windows XP/7/8/10, Linux(Ubuntu)
- Platform: Open GL.
- Language: C++ with OpenGL as API
- Libraries: LodePNG
- Software: Sublime Text Editor, GCC Compiler, MS Visual Studio 2008/2010 Package.

CHAPTER 3

SYSTEM DESIGN

Flow Chart is the pictorial representation of data flow of the project, which represents the data control will flow from one block to another block.

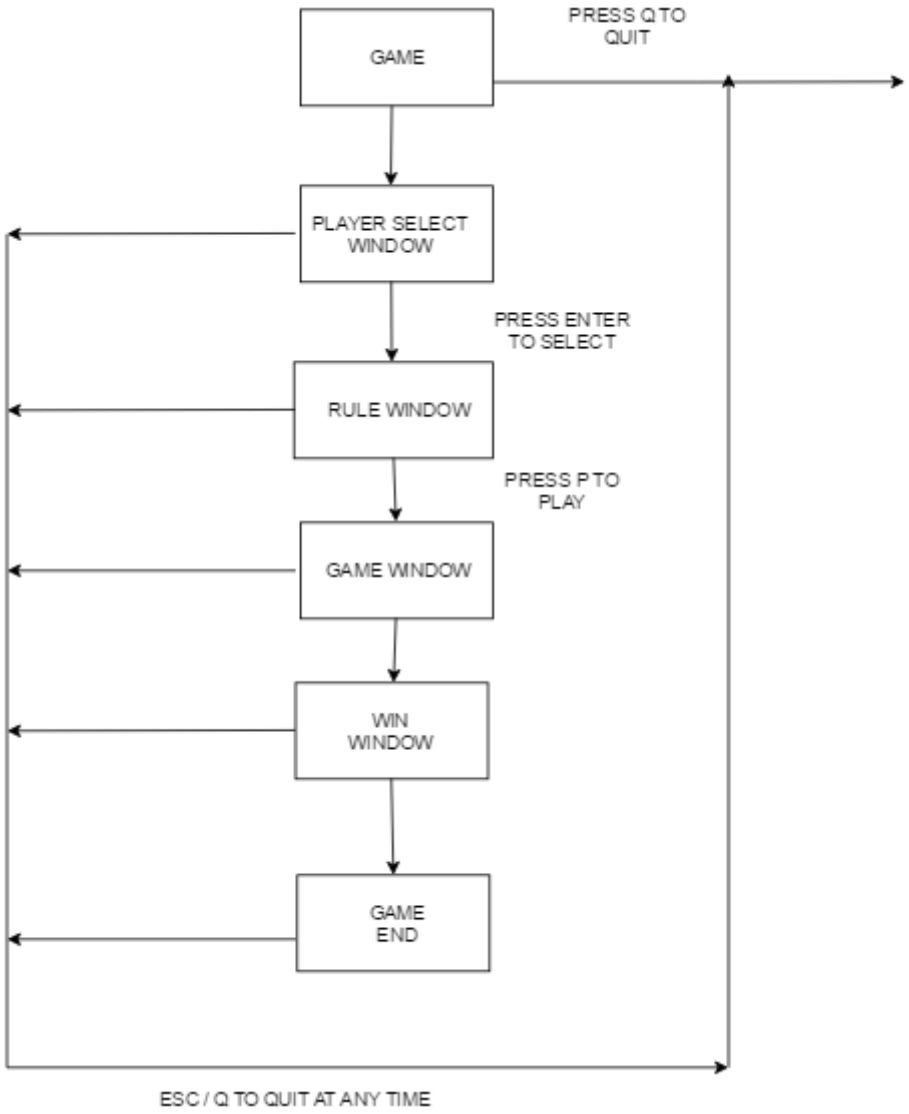

Figure 3.1 Shows the flow of control between different windows.

The more detailed outlook to the gameplay and its functioning is shown in the following flowchart

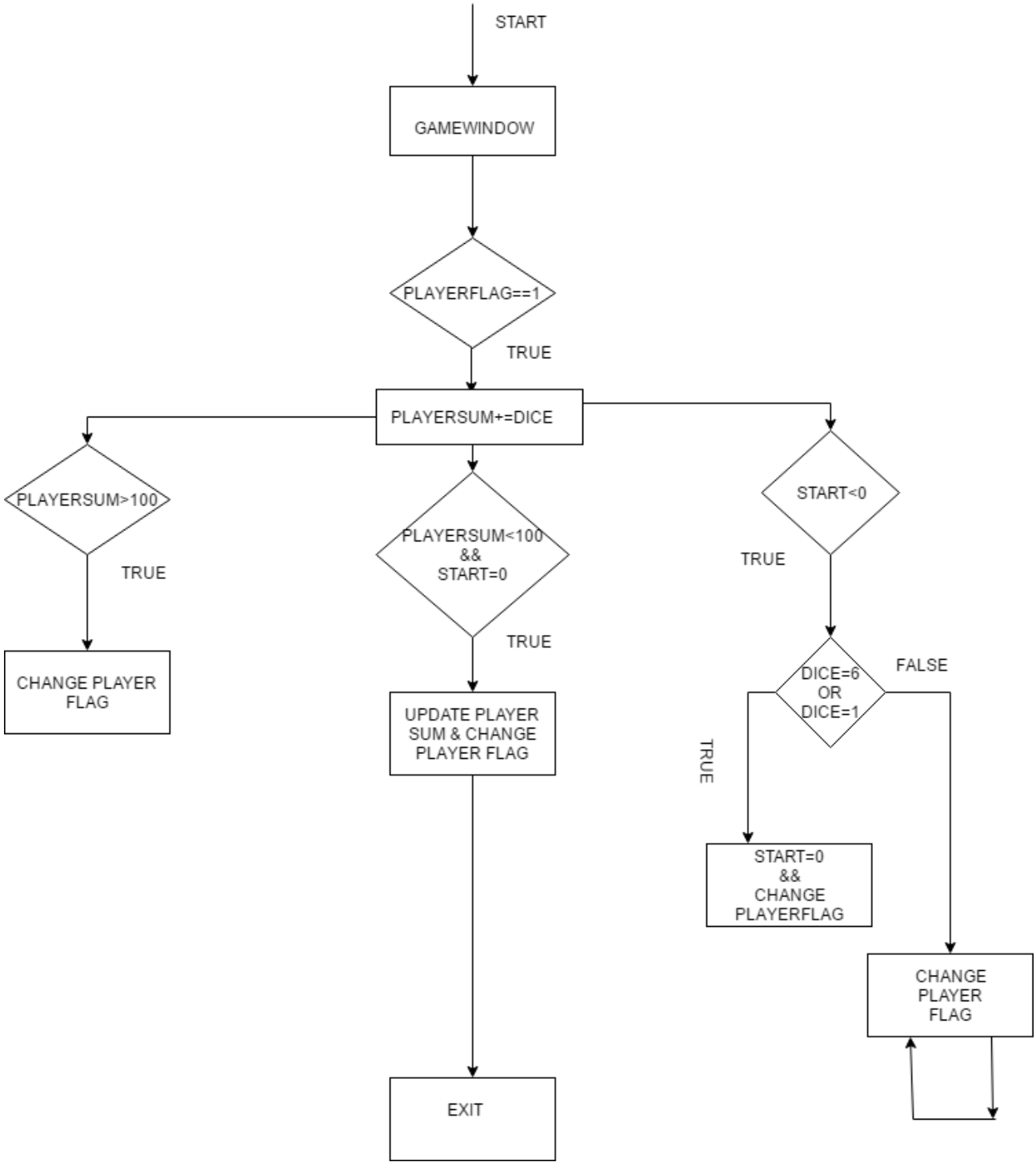

Figure 3.2 flowchart which shows the game flow.

Chapter 4

Source Code and Implementation

4.1 Source Code

- The required header files are for the standard library, for the graphics library and the used defined header in which the primitives and other built-in functions are defined.

```
//Glut required header files
#include <GL/glut.h>
#include <GL/glext.h>

//Other header files
#include <stdlib.h>
#include<math.h>
#include<stdio.h>
#include <iostream>
#include <vector>

//Library files to load image
#include "lodepng/lodepng.h"
#include "lodepng/lodepng.cpp"
```

```
/*****    Variables to be used in the program   *****/

//Variables used for Window transitions and renderring
int windowWidth;
int windowHeight;
bool window1=false;
bool window2=false;
bool window3=false;
bool window4=false;

    //Logo image and attributes
    vector <unsigned char> image_logo;
    unsigned logowidth;
    unsigned logoheight;
```

```
//Variables used for gameplay
int n=0;                                //Stores the image loading flag
float spin;                             //Stores spinning factor of the cube
int winner;                             //Stores the Winning player
int dice[4];                            //Stores dice values of players
int dicenum;                            //Stores Dice Value
int numplayers=0;                       //Stores number of players
int pc_counter=1;                       //Stores chances condition factor
void *currentfont;                      //Stores font address
int set_pointer=0;                      //Set program counter
int select_flag=0;                      //Stores user specified no. of players
int snake_pos[101];                     //Stores snake heads in the mesh
int stair_pos[101];                     //Stores ladders bottom in the mesh
int dice_position=-1;                   //Stores the dice movement
int player_sum[4]={0};                  //Stores the current
float dice_dimension=50;                //Stores the dice
int player_flag[4]={1,0,0,0};           //Stores the player which has current chance
float start[4]={-70,-70,-70,-70};       //Start positions of the players
float right_movement[4]={0};            //Monitors player position horizontally
float up_movement[4]={0};               //Monitors player position vertically
```

- **Functions used in program**

```
//For loading images
void setTexture(vector<unsigned char> img, unsigned width, unsigned height);
void invert(vector<unsigned char> &img,const unsigned width,const unsigned height);
void loadImage(const char* name,int n);

//For Stroke Drawing
void drawStrokeText(const char str[250],int x,int y,int z,float p1,float p2);
void setFont(void *font);
void drawstring(float x,float y,char *str);
```

```
/*****    Function prototypes for User defined functions    *****/

//Required for First Window
void windowOne();
void drawoptions();
void selectoptions();

//Required for Second Window
void windowTwo();

//Required for Third Window
void windowThree();
void drawMesh();
void drawplayer();
void drawdice();
void spinDice();
void gameplay();
void diceimages();
void diceposition();
void check_ladder();
void check_snake();

//Required for Fourth Window
void windowFour();


    /***** Glut functions with changed definitions    *****/

    static void init(void);
    static void idle(void);
    static void display(void);
    static void key(unsigned char key,int x,int y);
    static void specialkeys(int key,int x,int y);
    void mouse(int button, int state, int x, int y);
```

- These functions are required for successful compilation and working of the program.
- Main function handles control flow as the starting function.
- There are 4 frames that facilitate the proper game flow from player selection to winner display.

- **Main function**

The **main( )** function contains the main glut graphics engine initialization and other actions related engine.

```
int main(int argc, char *argv[])
{
    //loading image to memory
      loadImage("logo.png",n);  n=1;
      loadImage("board.png",n);

    //generating textures
      glGenTextures(1, &texname);

    //glut initialisation calls
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);

    //window attributes
      glutInitWindowSize(WIDTH,HEIGHT);
      glutInitWindowPosition(10,10);
      glutCreateWindow("Snake and Ladders");
      windowWidth=glutGet(GLUT_WINDOW_WIDTH);
      windowHeight=glutGet(GLUT_WINDOW_HEIGHT);

    //Other Initialisations
      glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
      glEnable(GL_BLEND);

    //Calls for functions
      init();
      glutFullScreen();
      glutDisplayFunc(display);
      glutKeyboardFunc(key);
      glutSpecialFunc(specialkeys);
      glutIdleFunc(idle);
      glutMouseFunc(mouse);

    //Continuous renderring the buffer
      glutMainLoop();

    return EXIT_SUCCESS;
}
```

- **init( )** is used to set viewport and camera projection of the program along with the coordinate system.

```
//Initialisation for the program
static void init(void)
{
    glClearColor(0.0,0.0,0.0,0.0);
    glViewport(0, 0,WIDTH, HEIGHT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0,1000,0,1000,0,1000);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity() ;
}
```

- **display( )** is used to output desired frame/window.

```
//Display Control of the Window and content
static void display(void)
{
    if(!window2)
        windowOne();
    else if(!window3)
        windowTwo();
    else if(!window4)
        windowThree();
    else
        windowFour();
}
```

- **key( )** is used to define keyword to perform specific functions and changing frames on pressing specific key.

```
if(key=='q' || key=='Q' || key==27)
{
    exit(1);
}
else if(key==13)
{
    window2=true;
}
else if(key=='p' || key=='P')
{
    window3=true;
}
```

- **specialkeys( )** function is used to select number of players by left right key selection.

```
if(key==GLUT_KEY_RIGHT)
{
    select_flag=(select_flag+1)%3;
}
else if(key==GLUT_KEY_LEFT)
{
    select_flag--;
    if(select_flag<0)
        select_flag=2;
}
```

- **idle( )** is used to redisplay the frame or in other words calling of concurrent display function.

- **mouse( )** enable us to take input from mouse.

  Left click -> is used here to rotate the dice.

  Right click -> is used to produce the dice output and end the player's turn.

  - **setTexture( )** is used to generate textures from images used.

```
glBindTexture(GL_TEXTURE_2D, texname);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```

  - OpenGL loads inverted image but we require the image to be upright so we invert the image using **Invert()**.

```
first = imageptr + h * width * 4;
last = imageptr + (height - h - 1) * width*4;
for( int i = 0; i < (int)width*4; ++i )
{
    temp = *first;
    *first = *last;
    *last = temp;
    ++first;
    ++last;
}
```

- **Loadimage( )** function is used to load images using lodepng library.

```
if((error=lodepng::decode(image_logo,logowidth,logoheight,name)))
{
    cout<<name<<":"<<lodepng_error_text(error)<<endl;
}
else
{
    invert(image_logo,logowidth,logoheight);
    cout<<"\n Logo Image Loaded Successfully\n";
}
```

- We have used Bitmap text from the glut for displaying the text in the window.
- For placing the cursor, for drawing bitmap character we use **glutBitmapCharacter.**

```
void *currentfont;

void setFont(void *font) //function to change the font of the text
{
    currentfont = font;
}
void drawstring(float x, float y, float z, char *str)//To render the text on the screen
{
    char *c;
    glRasterPos3f(x, y, z);

    for (c = str;*c != '\0';c++)
        glutBitmapCharacter(currentfont, *c);
}
```

- **glutBitmapCharacter** can print a single character at a time, thus a function **drawstring** is made which loops over the character array to print the whole passed string.

- **drawStrokeText( )** is used to render text on the display window.

```
for (i=0;str[i]!='\0';i++)
{
        glutStrokeCharacter(GLUT_STROKE_ROMAN , str[i]);
}
```

- **drawoptions( ) and selectoptions( )** functions are used to input no. of players from the user(s). Here we use LEFT and RIGHT key to toggle the no. of players options.

```
glColor3f(1.0,0.0,0.0);
glPushMatrix();
glTranslatef(transx,100,0);
    glRectf(cn-75,50.0,cn+75,150.0);
glPopMatrix();

glColor3f(0.0,1.0,0.0);
glPushMatrix();
    glTranslatef(transx,100,0);
glRectf(cn-350,50,cn-200,150);
glPopMatrix();

glColor3f(0.0,0.0,1.0);
glPushMatrix();
glTranslatef(transx,100,0);
    glRectf(cn+200,50,cn+350,150);
glPopMatrix();
```

- **WindowOne( )** is used to showcase frame #1 content. This window is a welcome screen to the program. Button select is done using keyboard arrow keys i.e. using left and right keys you can make the selection of what option to choose, then using the enter key you make the selection.

```
//Display function for Window One
void windowOne()
{
    glClear(GL_COLOR_BUFFER_BIT);
    float scale=0.70;
    drawoptions();
    selectoptions();
    glPushMatrix();
    //image begin
     glEnable(GL_TEXTURE_2D);
    setTexture(image_logo,logowidth,logoheight);
    glPushMatrix();
    glTranslatef(300,500,0);
    glScalef(scale,scale,1);
    glBegin(GL_POLYGON);
        glTexCoord2d(0,0);  glVertex2f(0,0);
        glTexCoord2d(0,1);  glVertex2f(0,logoheight);
        glTexCoord2d(1,1);  glVertex2f(logowidth,logoheight);
        glTexCoord2d(1,0);  glVertex2f(logowidth,0);
    glEnd();
    glDisable(GL_TEXTURE_2D);
    //image end
    glPopMatrix();
    glutSwapBuffers();
}
```

- **WindowTwo( )** is used as an information frame which tells the user the instructions and rules for playing the game. The instructions and rules are rendered using stroke functions.

```
glColor3f(0.0,1.0,0.0);
    drawStrokeText("Snake and Ladders - The Game of Chance",xpos,ypos,0,0.210,0.210);
    glBegin(GL_LINES);
        glVertex2f(xpos,ypos-15);glVertex2f(xpos+620,ypos-15);
    glEnd();
glPopMatrix();

glPushMatrix();
glTranslatef(xtrans,ytrans,0);
glColor3f(0.698, 0.133, 0.133);
glLineWidth(2.0);
    drawStrokeText("Here's a Snakes and ladders board game to play with your friends and family.
    glColor3f(1.0, 1.0, 0.0);
    drawStrokeText("RULES:",xpos*2.50+80,ypos-180,0,0.17,0.17);
```

Rules of the game are **:-**

- Objective of the game is to get to the number 100 which is the final destination.
- Each player puts their counter on the space near the arrow mark.
- Take it in turns to play the dice. Counter forwards to the number of spaces shown on the dice.
- Left Click starts the dice roll and Right Click stops the dice.
- If your counter lands at the bottom of a ladder, you can move up to the top of the ladder.
- If your counter lands on the head of a snake, you must slide down to the bottom of the snake.
- The first player to get to the space that says '100' is the winner.

After this user needs to press ‘P’ to play the game or Quit if the user wish to leave the game by pressing ECS key or ‘Q’.

The user after this frame enters the main gameplay frame which is Frame #3.

- **WindowThree( )** is the main gameplay window which shows the Board and Dice game.

We update flags based on the mouse key press events. These enable player movement

on the board .

- **drawMesh( )** is used to map the board of the game to facilitate player position and movement along the board.

```
glPointSize(4.0);
glColor3f(1.0,0.0,0.0);
glBegin(GL_LINE_LOOP);
    glVertex3f(i,j,50);
    glVertex3f(i,j+85,50);
    glVertex3f(i+70,j+85,50);
    glVertex3f(i+70,j,50);
glEnd();
```

- **drawPlayer( )** is used to draw the players circle on the snake and ladders board.

```
//Player 1
glColor3f(1.0,0.0,1.0);
glBegin(GL_POLYGON);

    for(int i=0;i<360;i++)
    {
        glVertex3f((radius*cos((pi/(float)180)*theta))+35+right_movement[0]+start[0],
        theta=theta+1;
    }
glEnd();

//Player 2
glPointSize(200.0);
glColor3f(0.0,0.0,1.0);
glBegin(GL_POLYGON);

    for(int i=0;i<360;i++)
    {
        glVertex3f((radius*cos((pi/(float)180)*theta))+35+right_movement[1]+start[1],
        theta=theta+1;
    }

glEnd();
```

- **drawDice( )** is used to render the dice cube on the frame. We are using OpenGL inbuilt functions to make the faces of the cube and connect them accordingly in a 3D plane.

```
//Bottom of the cube
glColor3f(1,0,0);
  glVertex3f(-dice_dimension,-dice_dimension,dice_dimension);
  glVertex3f(dice_dimension,-dice_dimension,dice_dimension);
  glVertex3f(dice_dimension,-dice_dimension,-dice_dimension);
  glVertex3f(-dice_dimension,-dice_dimension,-dice_dimension);
```

- **generate_num( )** is a function which outputs the dice values . It generates the dice numbers and return the same for proper gameplay. We use random function to generate numbers.

```
if(chancenum==0)
    dicenum=generate_num();
else
    dicenum=chancenum;
printf("dicenum : %d\n",dicenum);
return dicenum;
```

- **spinDice( )** is used to spin the cube in random directions in the 3D space according to programmer specified spin .

```
//Function to Spin Dice
void spinDice()
{
    spin = spin+50.0;
    if(spin > 360)
        spin-=359;
    glutPostRedisplay();
}
```

- **gameplay( )** – this is the main brain of the program. This function enables us

- to move the players

- to flag the turns of the respective players

- to monitor snakes and ladders encounter with the player

- to monitor the winner of the game.

We have put multiple conditions to ensure the movement or turns of the respective players isn't missed upon failure. This is how the game works.

This function uses array index data to monito snake and ladders to and from space positions on the board.

```
stair_pos[1]=38;stair_pos[4]=14;stair_pos[9]=31;stair_pos[21]=42;
stair_pos[28]=84;stair_pos[36]=44;stair_pos[51]=67;stair_pos[71]=91;stair_pos[80]=100;

snake_pos[16]=6;snake_pos[47]=26;snake_pos[49]=30;snake_pos[56]=53;snake_pos[62]=19;
snake_pos[63]=60;snake_pos[87]=24;snake_pos[93]=73;snake_pos[95]=75;snake_pos[98]=78;
```

The game engine ensures proper management of turns among players by using player flag and monitoring their respective sum along the whole tenure of the gameplay.

```
if(player_flag[((pc_counter)%numplayers)]==1 )
{
    printf("%d-->",player_sum[pc_counter%numplayers]);
    dice[((pc_counter)%numplayers)]=generate_num();

    if(( player_sum[((pc_counter)%numplayers)]+dice[((pc_counter)%numplayers)])>100)
    {
        player_flag[((pc_counter)%numplayers)]=0;
        player_flag[((pc_counter+1)%numplayers)]=1;
    }

    if(( player_sum[((pc_counter)%numplayers)]+dice[1])<=100 && (start[((pc_counter)%numplayers)]==0))
    {
        player_sum[((pc_counter)%numplayers)]+=dice[((pc_counter)%numplayers)];
        if(player_sum[((pc_counter)%numplayers)]==100)
        {
            printf("Winner decided\n");
            window4=true;
            winner=pc_counter%numplayers;
        }


 if(stair_pos[( player_sum[((pc_counter)%numplayers)]+1)]!=0)
 {
     //If Ladder is Found
     player_sum[((pc_counter)%numplayers)]=stair_pos[player_sum[((pc_counter)%numplayers)]+1]-1;

     if((( player_sum[((pc_counter)%numplayers)]/10)%2)!=0)
     {
         right_movement[((pc_counter)%numplayers)]=70*(9-(player_sum[((pc_counter)%numplayers)]%10));
     }
     else
     {
         right_movement[((pc_counter)%numplayers)]=70*( player_sum[((pc_counter)%numplayers)]%10);
     }

     up_movement[((pc_counter)%numplayers)]=85*( player_sum[((pc_counter)%numplayers)]/10);
     player_flag[((pc_counter)%numplayers)]=0;
     player_flag[((pc_counter+1)%numplayers)]=1;
 }
 else
 {
     if((( player_sum[((pc_counter)%numplayers)]/10)%2)!=0)
     {
         right_movement[((pc_counter)%numplayers)]=70*(9-(player_sum[((pc_counter)%numplayers)]%10));
     }
     else
     {
         right_movement[((pc_counter)%numplayers)]=70*( player_sum[((pc_counter)%numplayers)]%10);
     }

     up_movement[((pc_counter)%numplayers)]=85*( player_sum[((pc_counter)%numplayers)]/10);
     player_flag[((pc_counter)%numplayers)]=0;
     player_flag[((pc_counter+1)%numplayers)]=1;
 }
```

- **diceImages( )** is used to display dice value after the number is generated for the respective player.

```
if(dicenum==1)
{
    n=11;
    loadImage("dice1.png",n);
}

if(dicenum==2)
{
    n=12;
    loadImage("dice2.png",n);
}

if(dicenum==3)
{
    n=13;
    loadImage("dice3.png",n);
}
```

- We use LodePNG here to load these images accordingly using **diceposition( )** function;

```
if(dice[0]==1)
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glEnable(GL_TEXTURE_2D);
    setTexture(image_dice1,dice1width,dice1height);
    glPushMatrix();
    glTranslatef(850,200,0);
    glScalef(0.9,1,1);
    glBegin(GL_POLYGON);
        glTexCoord2d(0,0);  glVertex2f(60,-60);
        glTexCoord2d(0,1);  glVertex2f(60,60);
        glTexCoord2d(1,1);  glVertex2f(-60,60);
        glTexCoord2d(1,0);  glVertex2f(-60,-60);
    glEnd();
    glPopMatrix();
    glDisable(GL_TEXTURE_2D);
}
if(dice[0]==2)
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glEnable(GL_TEXTURE_2D);
    setTexture(image_dice2,dice2width,dice2height);
    glPushMatrix();
    glTranslatef(850,200,0);
    glScalef(0.9,1,1);
    glBegin(GL_POLYGON);
        glTexCoord2d(0,0);  glVertex2f(60,-60);
        glTexCoord2d(0,1);  glVertex2f(60,60);
        glTexCoord2d(1,1);  glVertex2f(-60,60);
        glTexCoord2d(1,0);  glVertex2f(-60,-60);
    glEnd();
    glPopMatrix();
    glDisable(GL_TEXTURE_2D);
}
```

- **WindowThree( )** function is used to render different images and figures whether 2D or 3D on the Frame#3. This function holds all the related function calls and other required conditions to run the game engine in an optimised manner.

```
//Display function for Window Three
void windowThree()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glClearColor(0.0,0.0,0.0,0.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glEnable(GL_TEXTURE_2D);
    setTexture(image_board,boardwidth,boardheight);
    glPushMatrix();
    glTranslatef(30,80,0);
    glScalef(0.9,1,1);
    glBegin(GL_POLYGON);
        glTexCoord2d(0,0);  glVertex2f(0,0);
        glTexCoord2d(0,1);  glVertex2f(0,pixelheight);
        glTexCoord2d(1,1);  glVertex2f(pixelwidth,pixelheight);
        glTexCoord2d(1,0);  glVertex2f(pixelwidth,0);
    glEnd();
    glPopMatrix();
    glDisable(GL_TEXTURE_2D);

    drawMesh();
    drawplayer();
    diceimages();


            glPushMatrix();
                glTranslatef(900.0,400.0,0.0);
                glRotatef(spin, 1.0, 0.5, 1.0);
                if(dice_position<0)
                    drawdice();
                if(dice_position>0)
                    diceposition();


            glPopMatrix();

            glutSwapBuffers();
        }
```

NOTE : After every frame change we use glutSwapBuffer( ) to change the buffer used for rendering the frames in the game.

- **WindowFour( )** is used to display information regarding the winner. This displays the Congratulation message and the player details who is at the destination position winning the game .

```
void windowFour()
{
    int num=0;
    num=(winner+1);
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1.0,1.0,1.0,1.0);

    setFont(GLUT_BITMAP_HELVETICA_18);
    char name[50]={"WINNER IS PLAYER --> "};
    char buffer[10]={'\0'};
    drawstring(500,500,name);

    glColor3f(1.0,1.0,1.0);
    glBegin(GL_LINE_LOOP);
        glVertex2f(480,400);
        glVertex2f(700,400);
        glVertex2f(700,600);
        glVertex2f(480,600);
    glEnd();

    glPointSize(10.0);
    glColor3f(1.0,1.0,0.0);
    sprintf(buffer,"%d",num);
    drawstring(650,500,buffer);

    glFlush();
    glutSwapBuffers();


}
```

Chapter 5

SNAPSHOTS

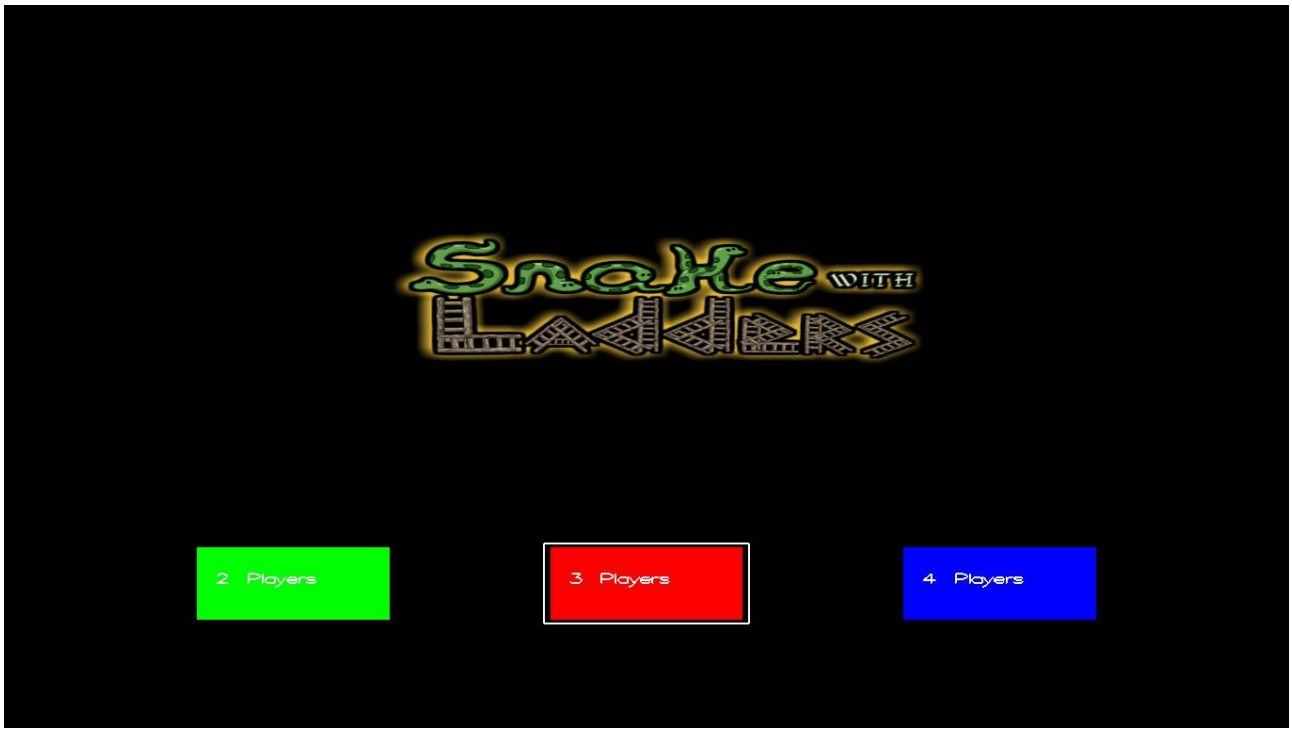

Figure 5.1 Welcome window and number of player choose option

Figure 5.2 Instruction and Rules Window

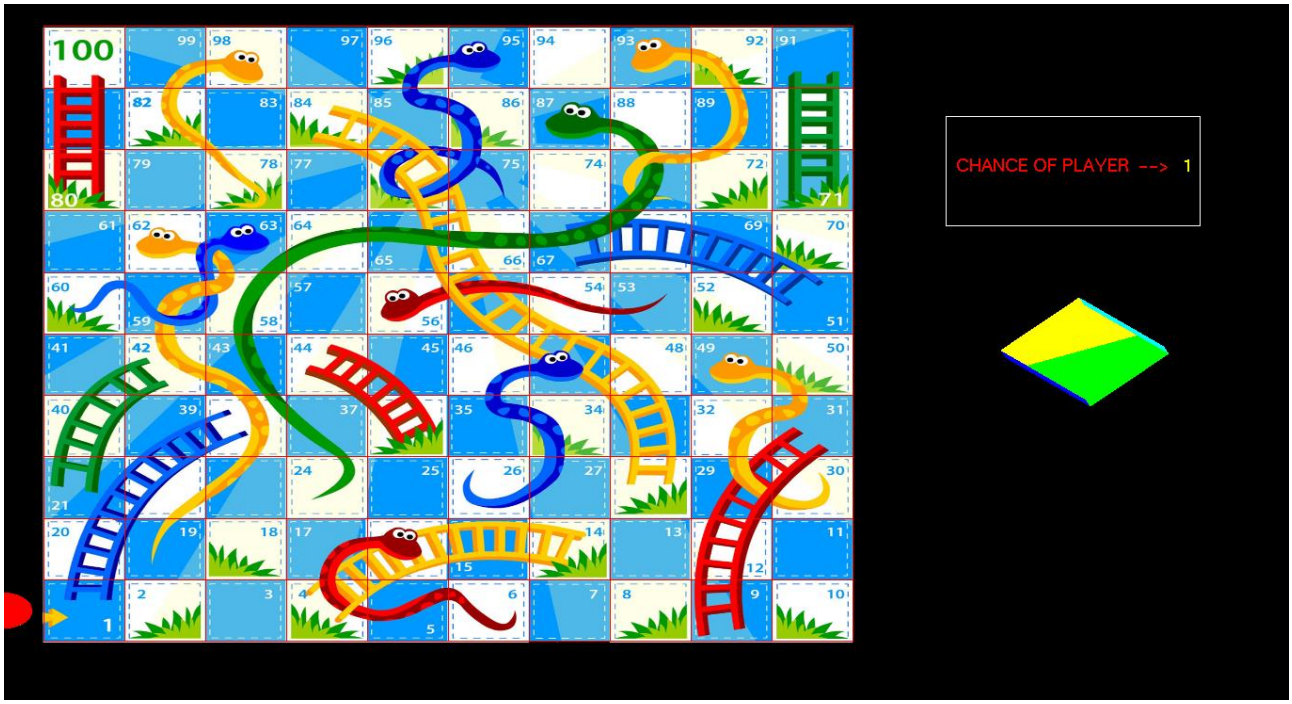

Figure 5.3 Game Window when Left Click Event occurs i.e. the player rolls the Die

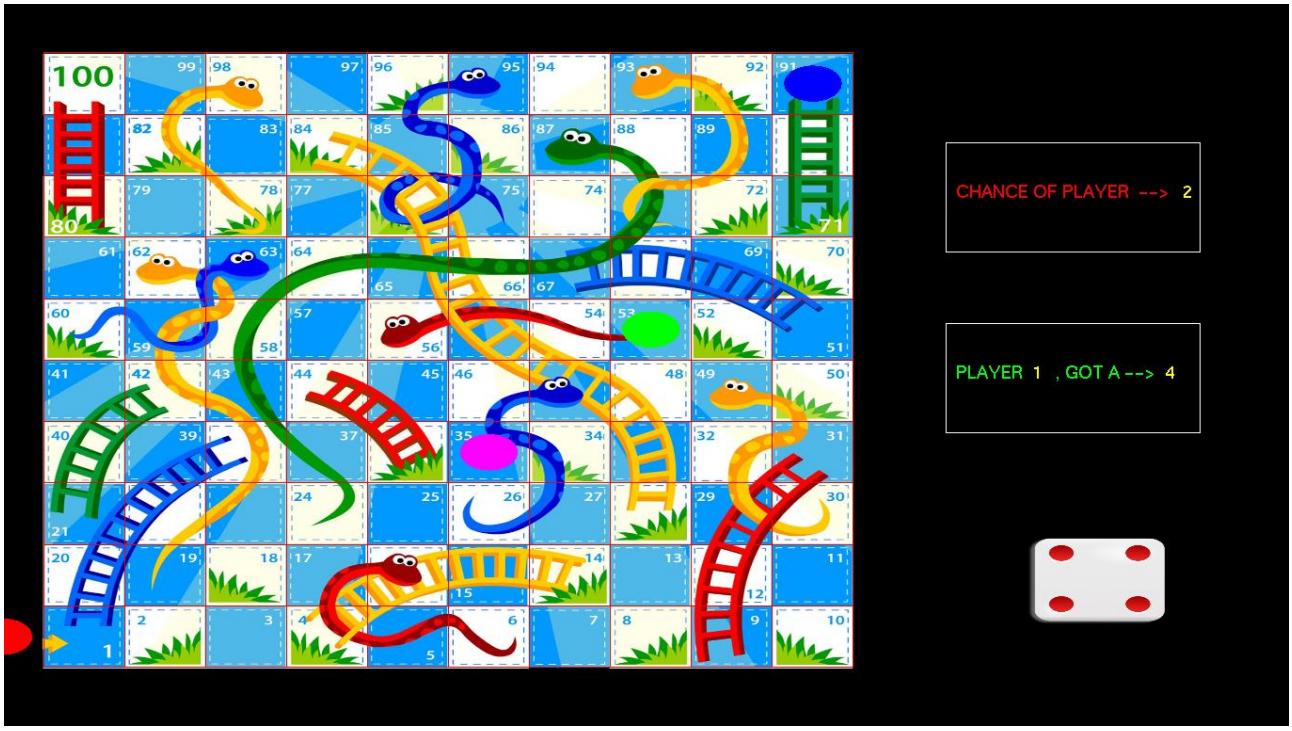

Figure 5.4 Game Window when Right Click Event occurs i.e. the player gets a number on the Die.

Figure 5.5 Game completed window

Chapter 6

Conclusion

An attempt has been made to develop an OpenGL graphics package, which meets necessary requirements of the users successfully. It enables us to learn about the basic concept in OPENGL graphics and graphics and know standard library graphics function and to explore some other function. OpenGL graphics is a huge library which consists of numerous functions.

The various shapes at lower level or to simulate any real thing animation etc. at high level. This project has given us an insight into the use of Computer graphics. As we had to use many built-in and user defined functions, we have managed to get a certain degree of familiarity with these functions and have now understood the power of these functions. We were able to comprehend the true nature of the most powerful tool graphics in OpenGL and have understood the reason why graphics is so powerful for programmers.

We can now converse with the certain degree of confidence about graphics in openGL. Finally, we have implemented this mini project "Snake and Ladders Board Game" using openGL package. Also, it is a fun to play game with siblings, friends, family members.

We would like to end by saying that doing this graphics project has been a memorable experience in which we have learned a lot, although there is a scope for further improvement. We got to know a lot of different applications of OPENGL while doing this project

FUTURE ENHANCEMENT

Scope of further improvement:

- Various lightening effects can be implemented to show shadows.
- The game can include 3D graphics and support for material properties.
- Better Dice and activity animations can be implemented.
- Various user interactivity can be enhanced by better graphics implementation.

BIBLIOGRAPGHY

Books:

The books that helped us in implementing this project are as follows:

- **Edward Angel:** Interactive Computer Graphics - A Top-Down Approach with OpenGL, 5th Edition, Pearson Education, 2008.

Reference Websites:

- **OpenGL Forums -** https://www.opengl.org/discussion_boards
- **Stackoverflow -** http://www.stackoverflow.com
- **GitHub -** https://github.com
- **OpenGL Reference -** https://www.khronos.org
- **Youtube –** https://www.youtube.com
- **Wikipedia -** https://en.wikipedia.org