INTERNATIONAL ISLAMIC UNIVERSITY CHITTAGONG

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Project Report
on
The World of Games

Course Title : Computer Graphics Lab

Course Code : CSE-4742

Submitted by

- Rehnuma Tasneem_C223288
- Saima Kawsar_C223297

Submitted to

Sultana Tasnim Jahan
Lecturer
CSE, IIUC

Date of Submission : 18-01-2026

TABLE OF CONTENTS

Introduction

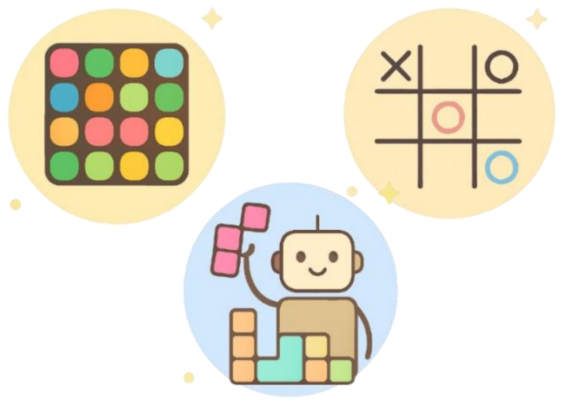
Gaming World is a graphical game-based project developed using C++ programming language and OpenGL (GLUT). This project is designed to demonstrate how computer graphics, user interaction, and basic game logic can be combined to create an interactive application. The project includes three popular and well-known games: Candy Crush, Tic Tac Toe, and Tetris, all integrated into a single program with a common main menu.

In modern software development, games are not only a source of entertainment but also an effective way to learn programming concepts. It was developed to gain practical experience in 2D graphics rendering, event handling, and real-time interaction. Instead of focusing only on theory, this project emphasizes hands-on learning through visual output and user control. The application starts with a main menu where the user can choose which game to play. Each game has its own gameplay rules, controls, and display system, but all of them run inside the same OpenGL window. The menu system makes the application user-friendly and easy to navigate. This project also helps in understanding how graphical objects are drawn on the screen, how keyboard and mouse inputs are handled, and how timers are used to control animations. Overall, Game Paradise serves as a learning platform to explore computer graphics and game development in a simple yet effective way.

Objectives

The objectives of this project are:

- To learn the basics of 2D game development using C++ and OpenGL
- To understand how graphics are drawn and updated on the screen
- To handle keyboard and mouse input for interactive gameplay
- To implement basic game logic such as scoring, turns, and win conditions
- To use timers and animations for smooth game movement
- To combine multiple games into a single application
- To improve problem-solving and programming skills through practical work



Importance of the Project

We chose this project because we wanted to learn programming in a more practical and interesting way, not just through theory. While studying computer graphics, only reading books was not enough to clearly understand how graphics, input handling, and animation actually work. So, we decided to build a game-based project where we could see real-time visual output and interact with the program directly.

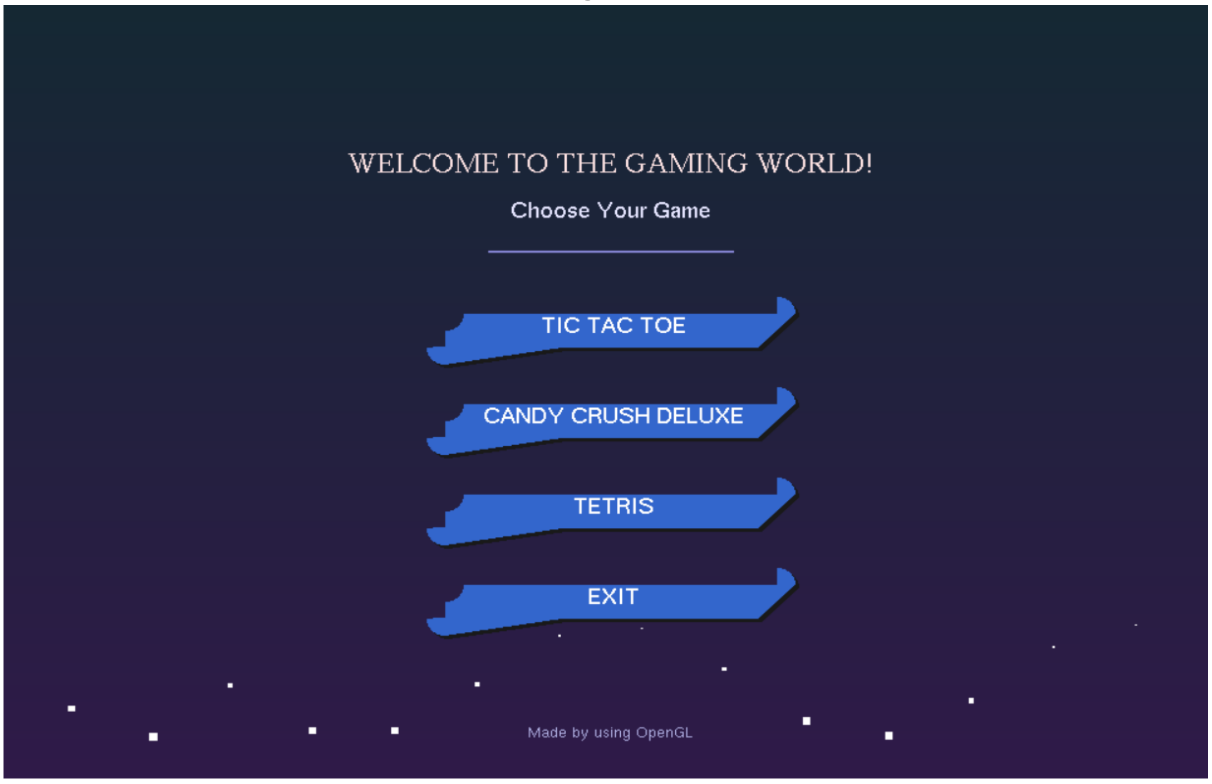
We selected this project because games like Candy Crush, Tic Tac Toe, and Tetris are simple but involve different types of logic. While working on this project, we learned how to draw objects on the screen, handle keyboard and mouse input, and control movement using timers. This project helped us understand how real applications are developed and increased our confidence in C++ programming. Overall, we chose this project because it helped us learn computer graphics in a practical, enjoyable, and meaningful way.

Key Aspects of the Project

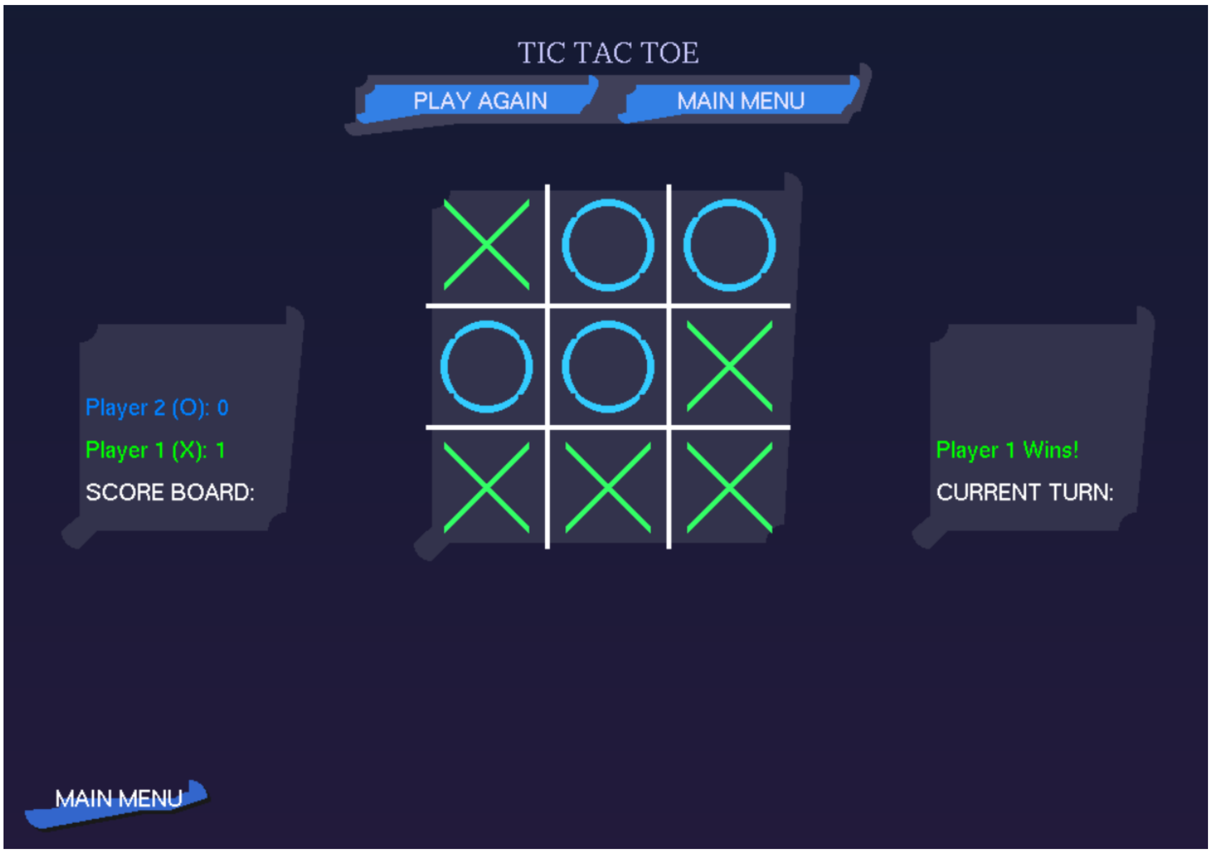
1. **Multiple Games in One Application**
   This project integrates three games Candy Crush, Tic Tac Toe, and Tetris into a single application, which makes the system organized and easy to use.
2. **Main Menu System**
   A graphical main menu is provided to select and switch between games, improving navigation and user experience.
3. **Use of OpenGL (GLUT)**
   OpenGL is used to render all 2D graphics such as grids, blocks, shapes, and text on the screen.
4. **Mouse Interaction**
   Mouse input is used for menu selection, Candy Crush gameplay, and making moves in Tic Tac Toe.
5. **Keyboard Controls**
   Keyboard input is used to control block movement and rotation in the Tetris game.
6. **Game Logic Implementation**
   Each game includes proper logic such as scoring systems, matching rules, player turns, and win or game-over conditions.
7. **Animation and Timers**
   Timers are used to control smooth animation and real-time movement, especially for the Tetris game.
8. **User-Friendly Interface**
   The interface is kept simple and clear so that users can easily understand and play the games.

Sample Input-Output

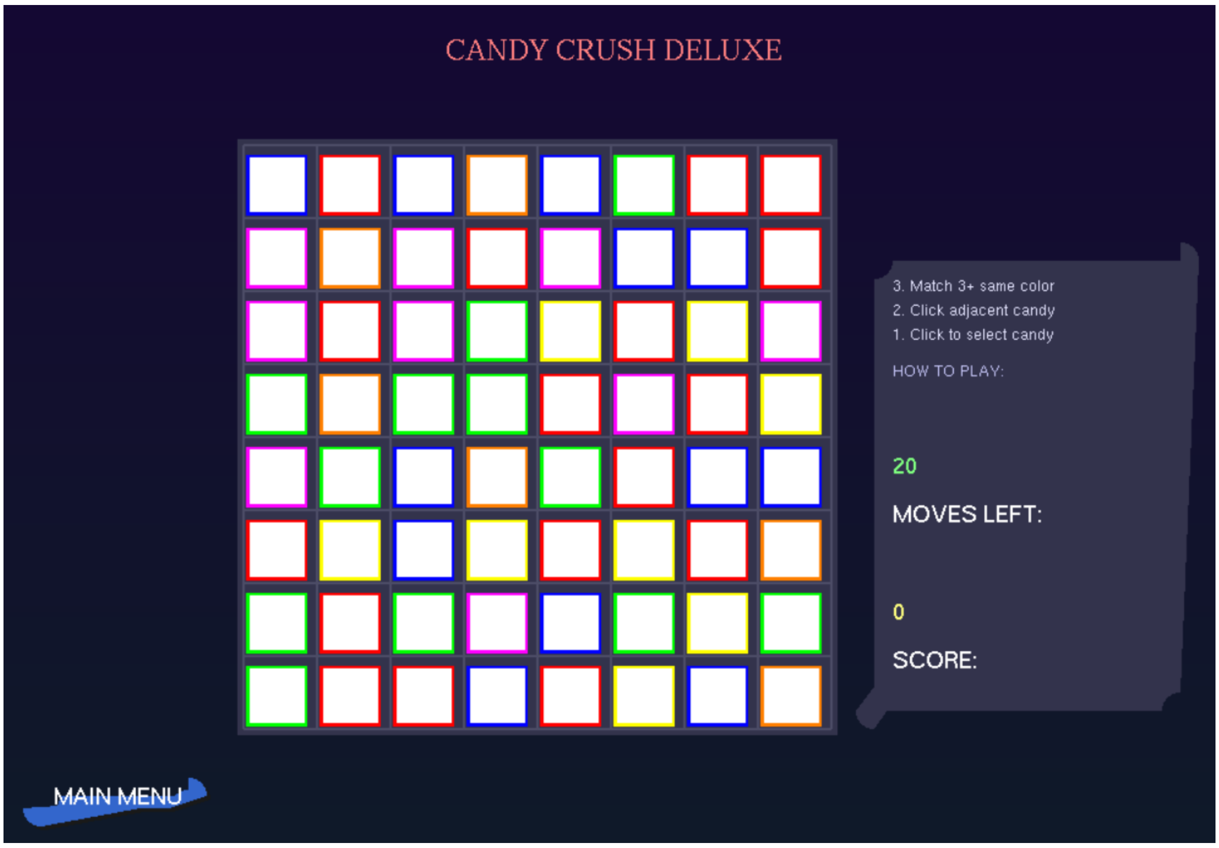
(Starting Interface)



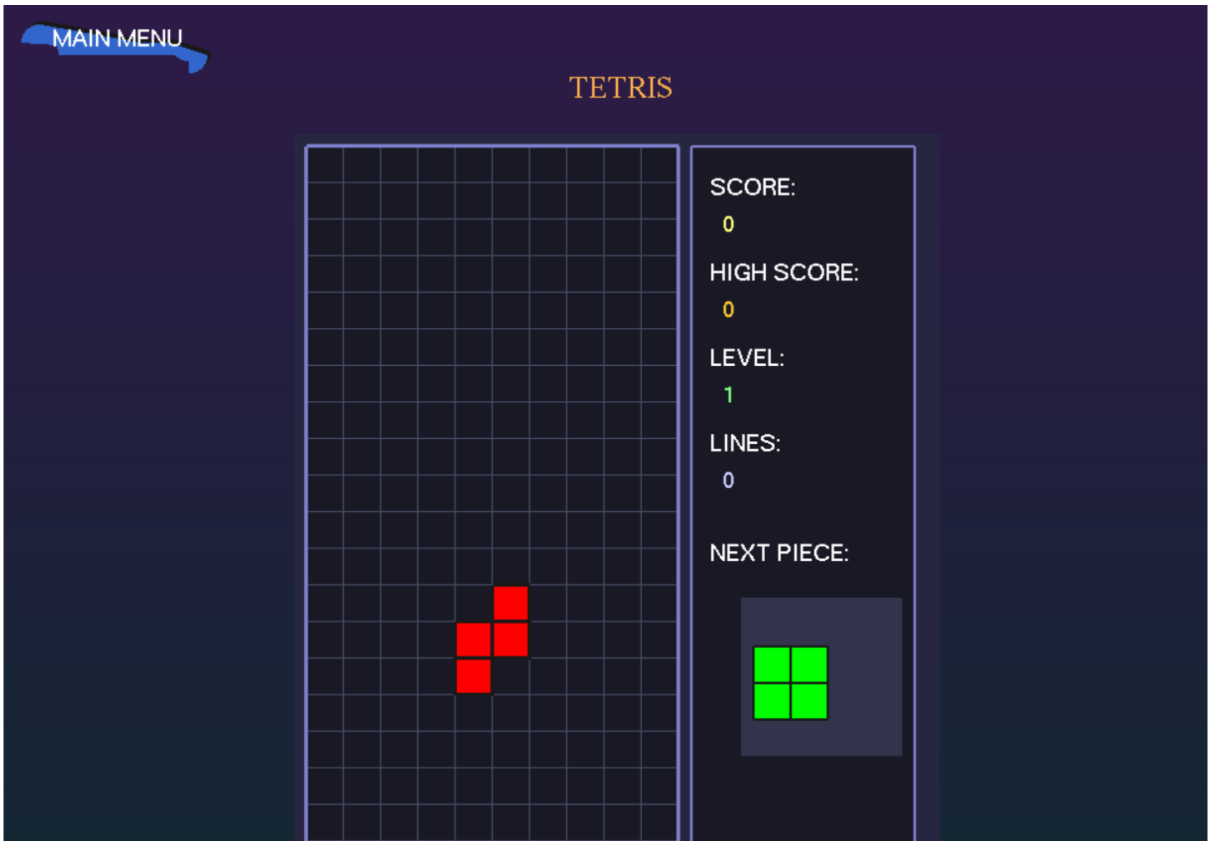
(Tic Tac Toe Game Interface)



(Candy Crush Game Interface)



(Tetris Game Interface)



Conclusion

In conclusion, Gaming World is a complete and practical implementation of a multi-game application developed using C++ and OpenGL (GLUT). Through this project, we were able to apply theoretical concepts of computer graphics into a real working system. Designing and developing three different games Candy Crush, Tic Tac Toe, and Tetris helped us understand how graphical objects are drawn, how user input is handled, and how game logic works in real-time applications. While working on this project, we gained hands-on experience in 2D graphics rendering, keyboard and mouse interaction, animation using timers, and basic game algorithms. Each game contributed differently to our learning process: Candy Crush improved our understanding of grid-based logic, Tic Tac Toe helped us learn turn-based logic and win conditions, and Tetris introduced time-based movement and collision detection. This project also improved our problem-solving skills, coding structure, and confidence in using OpenGL.

Overall, this project was a valuable learning experience that successfully met its objectives. It not only strengthened our foundation in computer graphics but also motivated us to explore more advanced topics such as sound effects, better animations, and artificial intelligence in future projects. Game Paradise proved to be an educational, practical, and enjoyable project.

Code Description:

This is an OpenGL-based Multi-Game Arcade project where users can select and play from main menu. Each game runs independently using separate states and input handling.

```
#include <GL/glut.h>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <set>
#include <fstream>
#include <string>

using namespace std;

/* ================= WINDOW
================== */
const int WINDOW_WIDTH = 1000;
const int WINDOW_HEIGHT = 700;

/* ================= STATES
================== */
enum Screen { MAIN_MENU,
CANDY_CRUSH, TIC_TAC_TOE,
TETRIS };
Screen currentScreen = MAIN_MENU;

/* ================= FONT
================== */
#define FONT_L
GLUT_BITMAP_TIMES_ROMAN_24
#define FONT_M
GLUT_BITMAP_HELVETICA_18
#define FONT_S
GLUT_BITMAP_HELVETICA_12

/* ================= BUTTON
================== */
struct Button {
    float x, y, w, h;
    string text;
    bool hover;
    int id;
};
vector<Button> menuButtons;
Button backButton;

/* ================= UTIL
================== */
void drawText(float x, float y, string s,
void* f) {
    glRasterPos2f(x, y);
    for (char c : s) glutBitmapCharacter(f,
c);
}

int textWidth(string s, void* f) {
    int w = 0;
    for (char c : s) w += glutBitmapWidth(f,
c);
    return w;
}

void drawRect(float x, float y, float w,
float h) {
    glBegin(GL_QUADS);
    glVertex2f(x, y);
    glVertex2f(x+w, y);
    glVertex2f(x+w, y+h);
    glVertex2f(x, y+h);
    glEnd();
}

void background() {
    glBegin(GL_QUADS);
    glColor3f(0.1f,0.1f,0.25f);
    glVertex2f(0,0);
    glVertex2f(WINDOW_WIDTH,0);
    glColor3f(0.2f,0.1f,0.4f);
    glVertex2f(WINDOW_WIDTH,WIND
OW_HEIGHT);
    glVertex2f(0,WINDOW_HEIGHT);
    glEnd();
}

void drawButton(Button &b) {
    glColor3f(0.1f,0.1f,0.1f);
    drawRect(b.x+3,b.y-3,b.w,b.h);

    if (b.hover) glColor3f(0.3f,0.6f,1);
    else glColor3f(0.2f,0.4f,0.8f);
    drawRect(b.x,b.y,b.w,b.h);

    glColor3f(1,1,1);
```

```
   drawText(b.x + (b.w-
textWidth(b.text,FONT_M))/2,
            b.y + b.h/2+5, b.text, FONT_M);
}

/* ================= MENU
================= */
void initMenu() {
   menuButtons.clear();
   float
w=300,h=60,x=(WINDOW_WIDTH-
w)/2,y=300;

   menuButtons.push_back({x,y+80,w,h,"
CANDY CRUSH",false,1});
   menuButtons.push_back({x,y,w,h,"TIC
TAC TOE",false,2});
   menuButtons.push_back({x,y-
80,w,h,"TETRIS",false,3});
   menuButtons.push_back({x,y-
160,w,h,"EXIT",false,4});

   backButton={20,20,140,40,"MAIN
MENU",false,0};
}

void displayMenu() {
   glClear(GL_COLOR_BUFFER_BIT);
   background();

   glColor3f(1,1,0.6f);
   drawText((WINDOW_WIDTH-
textWidth("MULTI GAME
ARCADE",FONT_L))/2,
            550,"MULTI GAME
ARCADE",FONT_L);

   for(auto &b:menuButtons)
drawButton(b);
   glutSwapBuffers();
}

/* ================= TETRIS
================= */
const int ROWS=20, COLS=10, BS=30;
int board[ROWS][COLS]={0};
int score=0;
float delay=0.5f;
bool rotateFlag=false;
int dx=0;
```

```
struct P{int x,y;} cur[4],bak[4];
int figures[7][4]={
 {1,3,5,7},{2,4,5,7},{3,5,4,6},
 {3,5,4,7},{2,3,5,7},{3,5,7,6},{2,3,4,5}
};

bool check() {
   for(int i=0;i<4;i++){
      if(cur[i].x<0||cur[i].x>=COLS||cur[i].
y>=ROWS) return false;
      if(board[cur[i].y][cur[i].x]) return
false;
   }
   return true;
}

void spawn() {
   int t=rand()%7;
   for(int i=0;i<4;i++){
      cur[i].x=figures[t][i]%2+COLS/2-1;
      cur[i].y=figures[t][i]/2;
   }
}

void rotate() {
   P p=cur[1];
   for(int i=0;i<4;i++){
      int x=cur[i].y-p.y;
      int y=cur[i].x-p.x;
      cur[i].x=p.x-x;
      cur[i].y=p.y+y;
   }
   if(!check()) for(int i=0;i<4;i++)
cur[i]=bak[i];
}

void tick() {
   for(int i=0;i<4;i++) bak[i]=cur[i];
   for(int i=0;i<4;i++) cur[i].x+=dx;
   if(!check()) for(int i=0;i<4;i++)
cur[i]=bak[i];

   if(rotateFlag) rotate();

   for(int i=0;i<4;i++) bak[i]=cur[i];
   for(int i=0;i<4;i++) cur[i].y++;

   if(!check()){
      for(int i=0;i<4;i++)
```

```
            board[bak[i].y][bak[i].x]=1;
        score+=10;
        spawn();
    }

    int k=ROWS-1;
    for(int i=ROWS-1;i>=0;i--){
        int c=0;
        for(int j=0;j<COLS;j++)
if(board[i][j]) c++;
        if(c<COLS){
            for(int j=0;j<COLS;j++)
board[k][j]=board[i][j];
            k--;
        } else score+=100;
    }
    for(int i=k;i>=0;i--)
        for(int j=0;j<COLS;j++)
board[i][j]=0;

    dx=0; rotateFlag=false;
}

void displayTetris() {
    glClear(GL_COLOR_BUFFER_BIT);
    background();
    drawButton(backButton);

    float ox=350, oy=100;
    glColor3f(1,1,1);
    drawText(430,620,"TETRIS",FONT_L)
;

    for(int i=0;i<ROWS;i++)
        for(int j=0;j<COLS;j++)
            if(board[i][j]){
                glColor3f(0,1,1);
                drawRect(ox+j*BS,oy+i*BS,BS
-1,BS-1);
            }

    for(int i=0;i<4;i++){
        glColor3f(1,0,0);
        drawRect(ox+cur[i].x*BS,oy+cur[i].y
*BS,BS-1,BS-1);
    }

    glColor3f(1,1,1);
    drawText(100,400,"SCORE:",FONT_M
);
    drawText(100,360,to_string(score),FON
T_M);

    glutSwapBuffers();
}

void timer(int){
    if(currentScreen==TETRIS){
        tick();
        glutPostRedisplay();
    }
    glutTimerFunc((int)(delay*1000),timer,
0);
}

/* ================= TIC TAC TOE
================== */
int ttt[3][3]={0}, turn=1;

void displayTTT(){
    glClear(GL_COLOR_BUFFER_BIT);
    background();
    drawButton(backButton);

    float s=300,x=(WINDOW_WIDTH-
s)/2,y=200,c=s/3;

    glColor3f(1,1,1);
    for(int i=1;i<3;i++){
        glBegin(GL_LINES);
        glVertex2f(x+i*c,y);
glVertex2f(x+i*c,y+s);
        glVertex2f(x,y+i*c);
glVertex2f(x+s,y+i*c);
        glEnd();
    }

    for(int i=0;i<3;i++)
        for(int j=0;j<3;j++){
            float cx=x+j*c+c/2, cy=y+i*c+c/2;
            if(ttt[i][j]==1){
                glColor3f(1,0,0);
                glBegin(GL_LINES);
                glVertex2f(cx-30,cy-30);
glVertex2f(cx+30,cy+30);
                glVertex2f(cx-30,cy+30);
glVertex2f(cx+30,cy-30);
                glEnd();
            }
            if(ttt[i][j]==2){
```

```
            glColor3f(0,1,1);
            glBegin(GL_LINE_LOOP);
            for(int a=0;a<360;a+=10)
               glVertex2f(cx+30*cos(a*3.14/
180),cy+30*sin(a*3.14/180));
            glEnd();
         }
      }
   glutSwapBuffers();
}

/* ================= INPUT
================= */
void mouse(int b,int x,int y){
   y=WINDOW_HEIGHT-y;

   if(b==GLUT_LEFT_BUTTON){
      if(x>=backButton.x&&x<=backButto
n.x+backButton.w &&
        y>=backButton.y&&y<=backButto
n.y+backButton.h){
         currentScreen=MAIN_MENU;
         glutPostRedisplay();
         return;
      }

      if(currentScreen==MAIN_MENU){
         for(auto &bt:menuButtons)
            if(x>=bt.x&&x<=bt.x+bt.w&&y
>=bt.y&&y<=bt.y+bt.h){
               if(bt.id==1)
currentScreen=CANDY_CRUSH;
               if(bt.id==2)
currentScreen=TIC_TAC_TOE;
               if(bt.id==3)
currentScreen=TETRIS;
               if(bt.id==4) exit(0);
            }
      }
   }
   glutPostRedisplay();
}

void motion(int x,int y){
   y=WINDOW_HEIGHT-y;
   for(auto &b:menuButtons)
      b.hover=(x>=b.x&&x<=b.x+b.w&&
y>=b.y&&y<=b.y+b.h);
   backButton.hover=(x>=backButton.x&
&x<=backButton.x+backButton.w&&
                y>=backButton.y&&y<=bac
kButton.y+backButton.h);
   glutPostRedisplay();
}

void keyboard(unsigned char k,int,int){
   if(currentScreen==TETRIS){
      if(k=='a') dx=-1;
      if(k=='d') dx=1;
      if(k=='w') rotateFlag=true;
      if(k==27)
currentScreen=MAIN_MENU;
   }
}

/* ================= MAIN
================= */
void display(){
   if(currentScreen==MAIN_MENU)
displayMenu();
   if(currentScreen==TETRIS)
displayTetris();
   if(currentScreen==TIC_TAC_TOE)
displayTTT();
}

int main(int argc,char**argv){
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_DOUBLE|
GLUT_RGB);
   glutInitWindowSize(WINDOW_WIDT
H,WINDOW_HEIGHT);
   glutCreateWindow("Multi Game
Arcade");

   gluOrtho2D(0,WINDOW_WIDTH,0,W
INDOW_HEIGHT);
   srand(time(0));
   spawn();
   initMenu();

   glutDisplayFunc(display);
   glutMouseFunc(mouse);
   glutPassiveMotionFunc(motion);
   glutKeyboardFunc(keyboard);
   glutTimerFunc(500,timer,0);

   glutMainLoop();
   return 0;
}
```