Final Project Report — Programming Fundamentals

University Name: FAST National University Karachi Campus

Department: Department of Computer Science

Course: Programming Fundamentals

Project Title: Piano Tiles

Submitted By: Rahia Sarfraz (Roll No. 25k-0546)

Rumaisa Abbasi (Roll No. 25k-0585)

Submitted To: Kinza Mushtaq

Semester: Fall 2025

Date: 21st November, 2025

Abstract

The Piano Tiles project is an interactive console game built in C. In this game, players must quickly press keys that correspond to randomly appearing falling tiles in 4 columns before they reach the bottom of the screen and disappear from the screen. It's a fun and fast-paced way to test reflexes, timing, and accuracy, keeping players engaged and on their toes. When a player hits tile within the range, a musical note plays, giving instant visual and auditory response! As the game goes on, the tiles appear faster, gradually increasing the challenge and pushing the player's focus and reflexes. The game also included element of surprise and difficulty by freeze tile which instantly ends the game when clicked and humorous GenZ-style roast and comments to enhance user engagement.

1. Introduction

The Piano Tiles game is a smooth and engaging project designed to bring a mobile-game experience into a C console application. Just like the original Piano Tiles, the main goal is simple: hit the right tiles at the right time. Tiles fall down four lanes, and the player must press A, S, D, or F before the tiles reach the bottom. The game becomes faster and more challenging as the player scores higher, making every second feel exciting. To make the game more engaging and user friendly, sound effects, colors and playful Gen-Z style comments are added giving console more lively experience.

2. Objectives

- Real Time Score tracking and remaining lives update
- User friendly Interaction to offer interactive and responsive gameplay
- To enhance responsivity and difficulty by adding freeze tiles and gradually increasing speed.
- To introduce sound integration to build element of fun.
- To design a clean, user-friendly interface for smooth gameplay.

3. System Design

System Overview

Flow of the program:

Start → Display game rules → Show grid and falling tiles → User interact and play game → Live lives and score updating → Real time Gen-Z comments → Click on freeze tiles or lives finish, the game ends

Algorithm

```
1.  Start the program
2.  Set up game variables: score = 0, lives = 5, speed = 300ms, gameOver = false.
3.  Initialize array of 'Tile' struct (Height = 15 – length of each column)
        Each tile gets a random lane (0-3), starting y-position (negative, spaced out),
        and freeze flag (10% chance of each tile to be flagged).
4.  Game starts
5.  Display game rules
6.  For each row (y from 0 to HEIGHT-1):
        For each lane (x from 0 to WIDTH-1):
            Check if any tile is at this (x, y) position.
              If yes, print a colored block (green for normal, blue for freeze); else, print empty space.
7.  If a key is pressed (non-blocking check):
        Map key to lane: A=0, S=1, D=2, F=3.
        First, scan tiles for freeze tiles at the bottom row (y == HEIGHT-1) in that lane:
            If found, set gameOver = true and display message.
        Then, scan for hittable tiles at the bottom row in that lane:
            If found, increment score, play sound, print random "roast" message, reset
            tile (new random lane/y/freeze), and potentially increase speed
            (every 5 points, decrease speed by 5ms if >60ms).
8.  For each tile:
        Increment y-position (move down).
        If y >= HEIGHT, reset tile: new random lane, y=0, new freeze chance.
    Sleep for 'speed' milliseconds to control frame rate.
    Repeat loop.
9.  When gameOver is true, display final score.
10. Prompt to play again (Y/N).
11. If yes, reset variables and restart loop; else, exit.
12. Stop background music and print farewell message.
13. End
```

Input & Output

Input: The player presses specific keys (A, S, D, F) to tap falling tiles.
The user chooses to restart or end the game using Y/N.

Output: Display grid and continuously falling tiles in each column.
Display current score and remaining lives.
Display Gen-Z comments throughout the game.
Plays a note or beep when a tile is successfully hit.

4. Implementation

Language: C

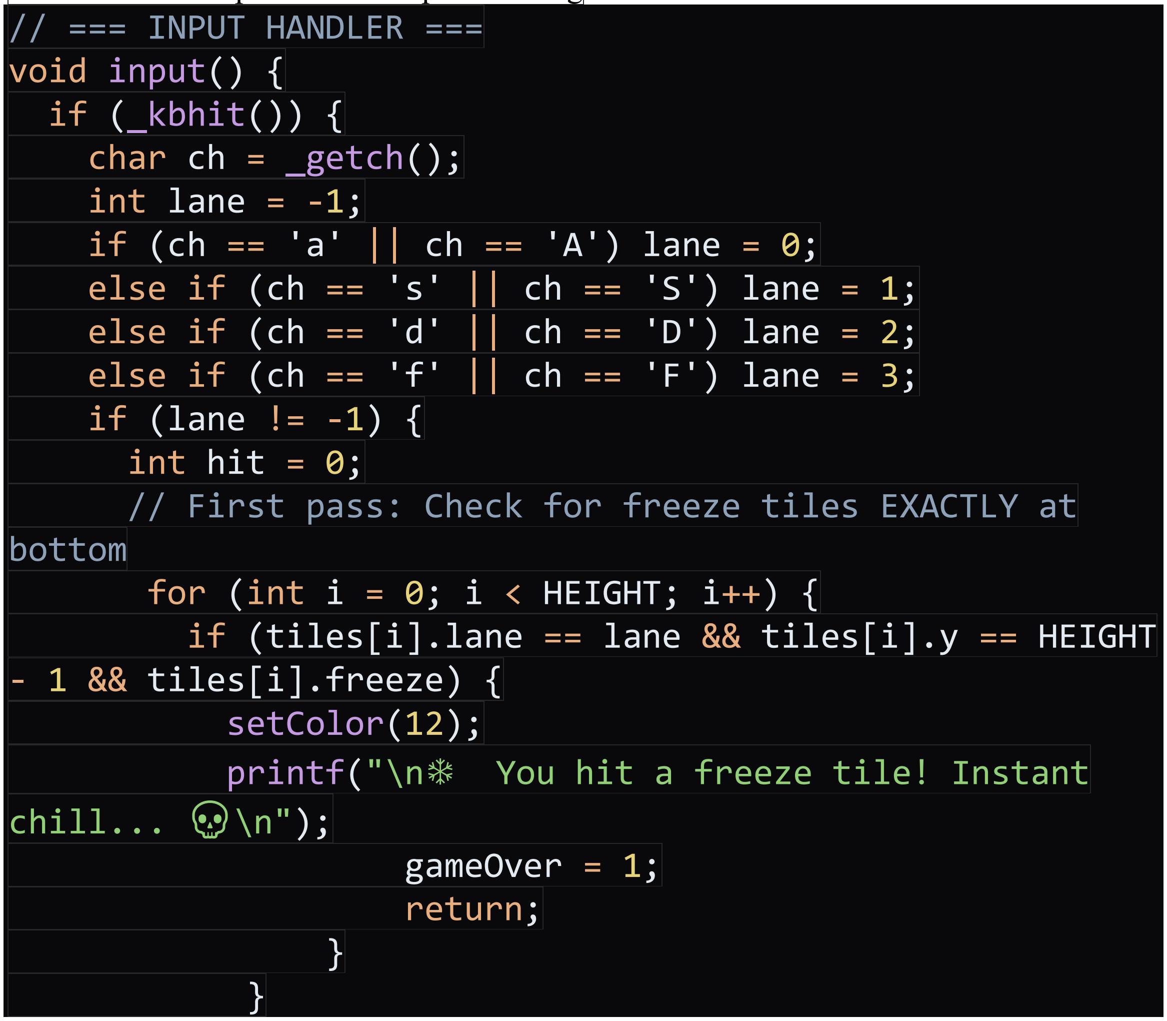
Compiler/IDE: Code::Blocks / Dev C++ / GCC - Visual Studio Code.

Key Features

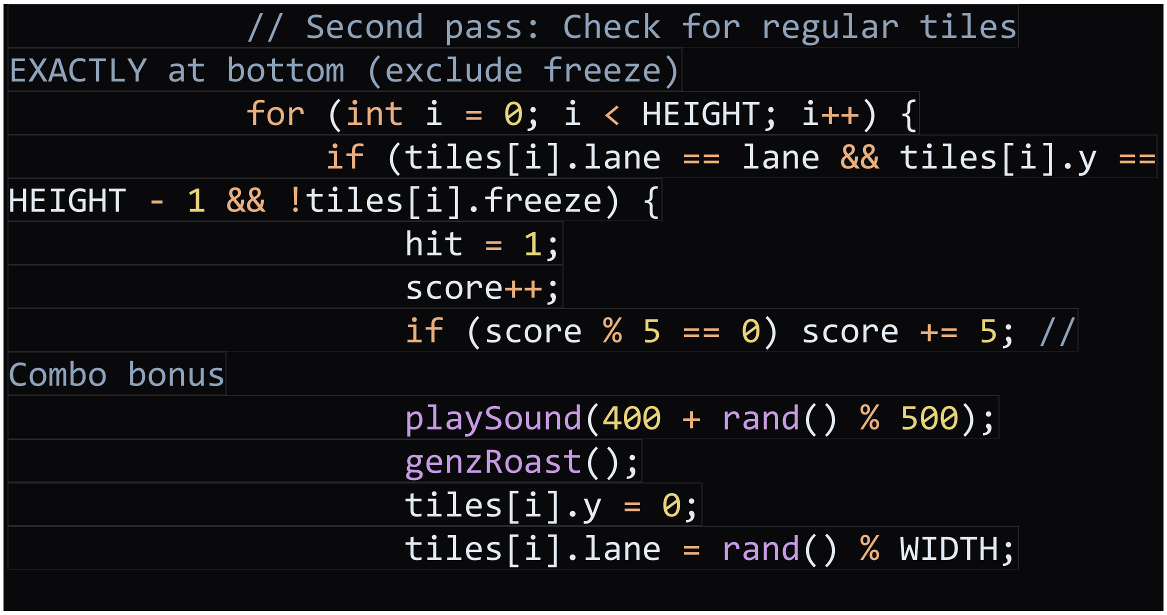
- Real-Time Tile Falling System
- Keyboard-Based Controls [A, S, D, F]
- Dynamic Score and Lives updating
- End Game when touch Freeze tiles or lives end
- Fun and Custom Gen-Z comments
- Restart or end game
- Color effects and Sound effects to bring audio and visionary element
- Runs smoothly on low-spec PC

Code Snippet

Main function responsible for input handling

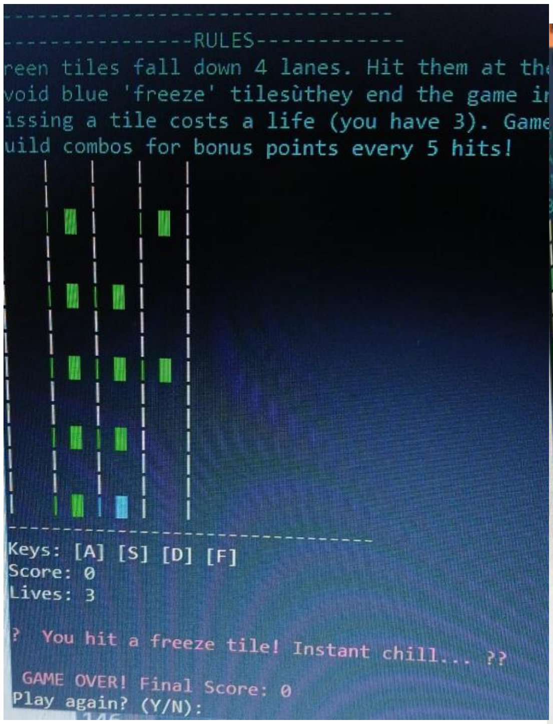


```c
// === INPUT HANDLER ===
void input() {
  if (_kbhit()) {
    char ch = _getch();
    int lane = -1;
    if (ch == 'a' || ch == 'A') lane = 0;
    else if (ch == 's' || ch == 'S') lane = 1;
    else if (ch == 'd' || ch == 'D') lane = 2;
    else if (ch == 'f' || ch == 'F') lane = 3;
    if (lane != -1) {
      int hit = 0;
      // First pass: Check for freeze tiles EXACTLY at bottom
       for (int i = 0; i < HEIGHT; i++) {
         if (tiles[i].lane == lane && tiles[i].y == HEIGHT - 1 && tiles[i].freeze) {
           setColor(12);
           printf("\n❄  You hit a freeze tile! Instant chill... 💀\n");
                    gameOver = 1;
                    return;
                }
            }
```

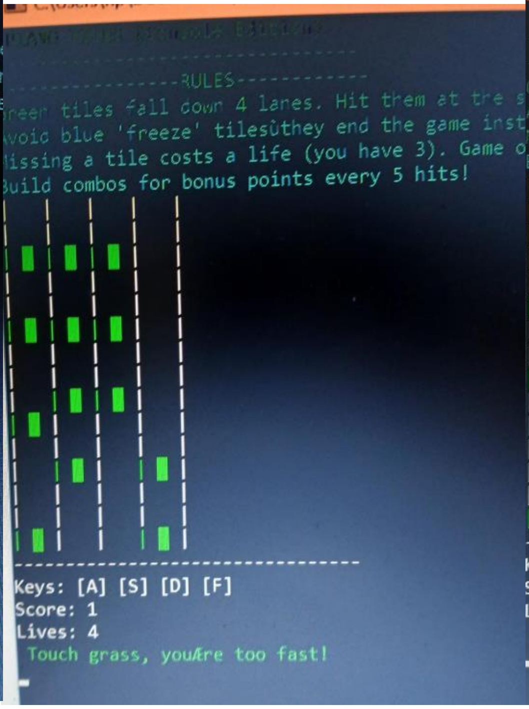


```
            // Second pass: Check for regular tiles
EXACTLY at bottom (exclude freeze)
            for (int i = 0; i < HEIGHT; i++) {
                if (tiles[i].lane == lane && tiles[i].y ==
HEIGHT - 1 && !tiles[i].freeze) {
                    hit = 1;
                    score++;
                    if (score % 5 == 0) score += 5; //
Combo bonus
                    playSound(400 + rand() % 500);
                    genzRoast();
                    tiles[i].y = 0;
                    tiles[i].lane = rand() % WIDTH;
```

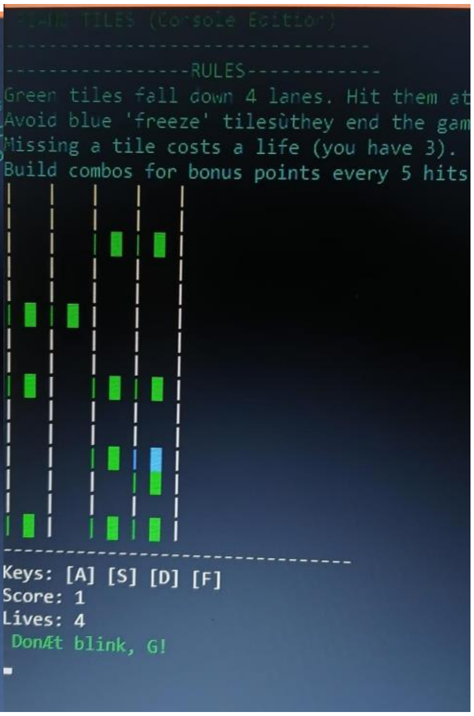
Sample Output



```
--------------------------------------
----------------RULES------------
reen tiles fall down 4 lanes. Hit them at the
void blue 'freeze' tilesùthey end the game ir
issing a tile costs a life (you have 3). Game
uild combos for bonus points every 5 hits!

--------------------------------------
Keys: [A] [S] [D] [F]
Score: 0
Lives: 3

? You hit a freeze tile! Instant chill... ??

 GAME OVER! Final Score: 0
Play again? (Y/N):
```

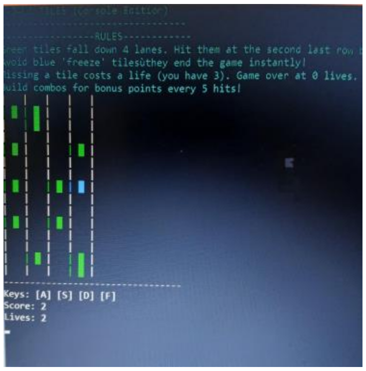
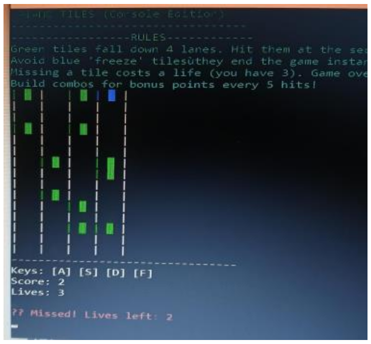
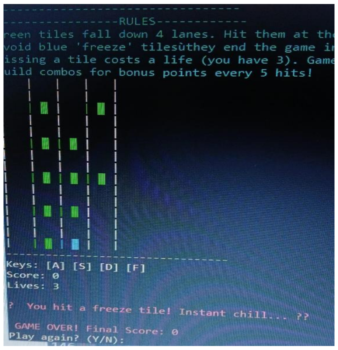


```
--------------------------------------
----------------RULES------------
reen tiles fall down 4 lanes. Hit them at the s
voic blue 'freeze' tilesûthey end the game inst
issing a tile costs a life (you have 3). Game o
Build combos for bonus points every 5 hits!

--------------------------------------
Keys: [A] [S] [D] [F]
Score: 1
Lives: 4
 Touch grass, youÆre too fast!
```



```
PIANO TILES (Console Edition)
--------------------------------------
----------------RULES------------
Green tiles fall down 4 lanes. Hit them at
Avoid blue 'freeze' tilesùthey end the gam
Missing a tile costs a life (you have 3).
Build combos for bonus points every 5 hits

--------------------------------------
Keys: [A] [S] [D] [F]
Score: 1
Lives: 4
 DonÆt blink, G!
```

5. Testing & Results

Test No	Input	Expected	Actual Output	Status
1	Click a tile using A S D F	Continue Smooth game	 TILES (Console Edition) -----------------RULES------------ een tiles fall down 4 lanes. Hit them at the second last row void blue 'freeze' tilesùthey end the game instantly! issing a tile costs a life (you have 3). Game over at 0 lives. uild combos for bonus points every 5 hits! Keys: [A] [S] [D] [F] Score: 2 Lives: 2	✓
2	Missed a tile	Show remaining lives and indicates tile missed	 TILES (Console Edition) -------------------RULES------------- Greer tiles fall down 4 lanes. Hit them at the se Avoid blue 'freeze' tilesùthey end the game instar Missing a tile costs a life (you have 3). Game ove Build combos for bonus points every 5 hits! Keys: [A] [S] [D] [F] Score: 2 Lives: 3 ?? Missed! Lives left: 2	✓
3	Click blue restricted tile	Program end instantaneously	 ----------------RULES------------ reen tiles fall down 4 lanes. Hit them at the void blue 'freeze' tilesùthey end the game ir issing a tile costs a life (you have 3). Game uild combos for bonus points every 5 hits! Keys: [A] [S] [D] [F] Score: 0 Lives: 3 ? You hit a freeze tile! Instant chill... ?? GAME OVER! Final Score: 0 Play again? (Y/N):	✓

The program performed successfully for all conditions. It handled both marked green tiles and restricted blue tiles efficiently, generated top tier comment, and allowed smooth gameplay. Execution speed was near-instant, and the program ran smoothly in all Dev C++ , visual studio and CMD.

6. Conclusion, Limitations & References

Conclusion

The Piano Tiles game successfully demonstrates how interactive, real-time applications can be built using fundamental C programming concepts such as loops, arrays, structures, condition handling, and keyboard input. The game recreates the excitement of fast-paced reaction-based gameplay while maintaining a simple console interface. Through features like falling tiles, score tracking, lives system, sound effects, and freeze-tile challenges, the project highlights both logic building and event-driven programming.

Developing this project enhanced understanding of essential programming skills, including timing control, user interaction, random generation, and screen rendering. Although the console environment limits graphics and audio capabilities, the game still delivers an engaging and enjoyable experience. With further improvements—such as enhanced visuals, background music, game modes, and online leaderboards—the project has strong potential for expansion into more advanced versions.

Overall, Piano Tiles serves as a solid foundation for learning game development concepts in C and showcases how creative ideas can be transformed into a functional and entertaining application.

.

Limitations

- Data is lost when the program closes (no file handling yet).
- No graphical interface — purely console-based.
- Limited Sound Support to only 1 sound.
- No Advanced Game Modes.
- Limited Tile Variety to only freeze tiles.
- Keyboard Dependency
- Platform Restriction. No cross-platform integrated.

Future Enhancements

- Improved Graphics and Animations (dynamic backgrounds, different types of tiles and animated score counters)
- Graphic-Based UI Framework
- Special tiles: Bomb Tiles, Bonus Tiles, Fire Tiles, Combo Tiles
- Different background and modes (Classic Mode, Arcade Mode, Zen Mode, Rush Mode) for each gameplay.
- Custom Music Mode when tiles tapped.

References

- https://github.com/example/piano-tiles-console
- Learn OpenGL or SDL for Console Games
- Wikipedia: ANSI Escape Codes