“Final Project Report — Programming Fundamentals”

University Name: FAST – National University of Computer and Emerging Sciences

Department: Department of Computer Science

Course: Programming Fundamentals

Project Title: Falling Stars

Submitted By: - Anseeba Ali Jivani (Roll No. 25K-0691)

- Esh Tu Raazia (Roll No. 25K-0668)

Submitted To: Miss. Izzah Salam

Semester: Fall 2025

Date: 25th November, 2025

Table of Contents

Abstract

"Falling Stars" is a 2D console-based game developed in the C programming language, designed to provide a simple yet engaging gaming experience. The objective of the game is to collect falling stars in a movable container while avoiding missed catches that result in loss of lives. With a maximum of three chances, the game challenges players to react quickly and coordinate their movements effectively. The project addresses common issues found in similar games; particularly the imbalance between falling speed and container movement, by ensuring smoother and more playable mechanics. This system demonstrates fundamental programming concepts such as loops, conditionals statements, functions, and smooth keyboard input processing, thus contributing to a well-structured and technically sound game application.

1. Introduction

"Falling Stars" is a console-based 2D game where the player aims to catch falling stars using a container positioned at the bottom of the screen. Each missed star deducts one life, and the player loses the game after three missed attempts. Beyond entertainment, the game encourages players to improve hand-eye coordination and quick decision-making skills. Designed for simplicity and accessibility, the game runs in a text-based environment while still offering an engaging and interactive experience. It uses fundamental C programming concepts such as timing, movement control, and input handling, all of which are necessary for forming a functional console-based game.

2. Objectives

- To design an engaging 2D console-based game where the player catches falling stars.
- To ensure balanced gameplay by aligning falling speed with container movement speed.
- To provide a smooth and frustration-free player experience without unnecessary interruptions such as ads.
- To apply core programming concepts including loops, conditional statements, and functions.
- To help improve player skills such as reaction time, hand-eye coordination, and quick decision-making.
- To create a functional game using C language and standard console libraries.

3. System Design

System Overview

The Falling Stars Game is designed as a 2D console-based application in C. The system controls the falling motion of stars, the movement of the container, life count, levels, and scoring. The program repeatedly clears and redraws the screen to create the illusion of animation. Gameplay continues until the player either completes all levels or loses all lives.

Flow of the program:

Start → Display Intro → Initialize Level → Generate Falling Star → Move Container → Check Catch or Miss → Update Score/Lives → Level Complete or Game Over → Exit

Algorithm

1. Start the program.
2. Hide the console cursor and display the game instructions.
3. Set initial values: lives, score, level, fall speed, and container width.
4. For each level:
    - Set required score and difficulty.
    - Reposition the container.
    - Continuously generate falling stars.
    - Move the container according to arrow key input.
    - Redraw the screen using clearScreen() and drawScreen().
    - Increase score if the star is caught; otherwise decrease lives.
5. If lives reach zero → display 'Game Over'.
6. If required score is achieved → move to the next level.
7. After completing all levels → display 'You Win'.
8. End the program.

Input & Output

Input:

- Arrow keys (< and >) for moving the container left and right.
- Randomly generated star position handled internally.
- Player key press to start the game (via getch()).

Output:

- Updated game screen showing falling star, container, lives, score, and level.
- Messages such as Level Complete, Game Over, or You Win.

4. Implementation

Language: C

Compiler/IDE: Dev C++

Operating System: Windows

Key Features

- Animated falling stars using repeated screen clearing and redrawing
- Container movement controlled through left and right arrow keys
- Life and scoring system with a maximum of 3 lives
- Smooth gameplay through balanced falling speed and container movement
- Level progression, where difficulty increases each level
- Real-time input handling using kbhit() and getch()
- Clear game interface with a header showing lives, score, and level

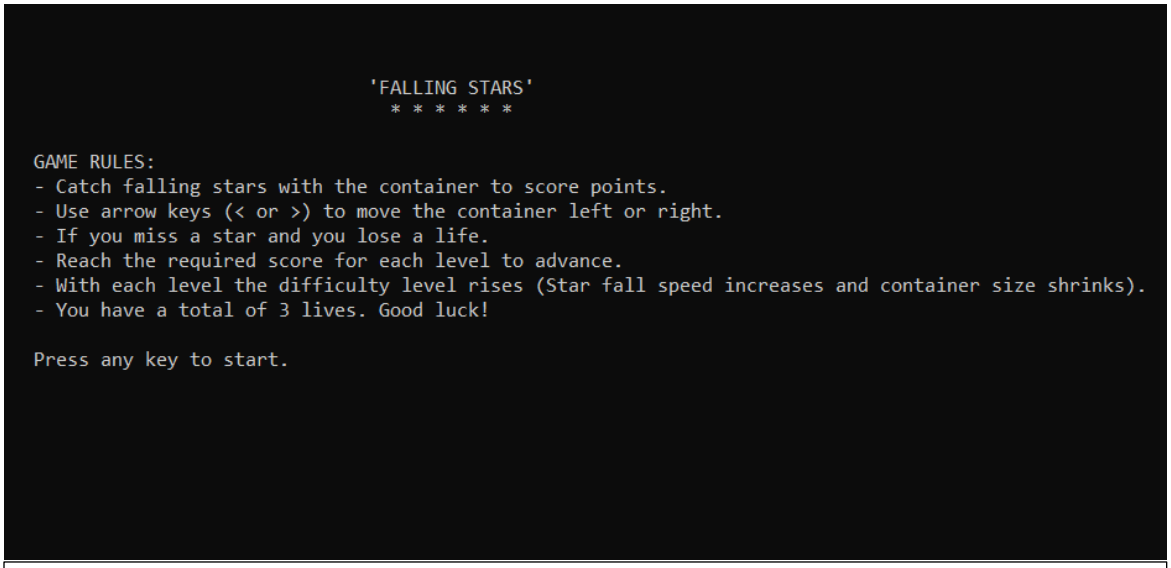
Code Snippet

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
#include <windows.h>

#define SCREEN_HEIGHT 23
#define SCREEN_WIDTH  60

    void clearScreen(){
        system("cls");
    }

    void gotoxy(int x, int y){
        COORD coord;
        coord.X = x;
        coord.Y = y;
        SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
    }

    void displayIntro(){
        clearScreen();
        printf("\n\n\n");
        printf("\t\t\t\t 'FALLING STARS' \n\t\t\t\t   * * * * * *\n");
        printf("\n");
        printf("   GAME RULES:\n");
        printf("   - Catch falling stars with the container to score points.\n");
        printf("   - Use arrow keys (< or >) to move the container left or right.\n");
        printf("   - If you miss a star and you lose a life.\n");
        printf("   - Reach the required score for each level to advance.\n");
        printf("   - With each level the difficulty level rises (Star fall speed increases and container size shrinks).\n");
        printf("   - You have a total of 3 lives. Good luck!\n");
        printf("\n");
        printf("   Press any key to start.\n");
        getch();
    }

    void drawHeader(int lives, int levelScore, int level){
        int i;
        printf("================== FALLING STAR GAME ==================\n");

        printf("Lives: ");
```

```
    printf("Lives: ");
    for (i = 0; i < 3; i++){
        if (i < lives){
            printf("%c ", 220);
        }
        else{
            printf("  ");
        }
    }
    printf("\t\tScore: %d", levelScore);
    printf("\t\tLevel: %d\n", level);
    printf("=========================================================\n\n");

}

void drawScreen(int starRow, int starCol, int containerRow, int containerLeft, int containerRight){

int r, c;
for (r = 0; r < SCREEN_HEIGHT; r++){
    for (c = 0; c < SCREEN_WIDTH; c++){
        if (r == starRow && c == starCol){
            printf("*");
        }
        else if (r == containerRow && c >= containerLeft && c <= containerRight){
            if (c == containerLeft || c == containerRight){
                printf("|");
            }
            else{
                printf("_");
            }
        }
        else{
            printf(" ");
        }
    }
    printf("\n");
}

fflush(stdout);
}

int main() {
    srand(time(NULL));

    CONSOLE_CURSOR_INFO info;
    info.dwSize = 100;
    info.bVisible = FALSE;
    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &info);

    displayIntro();

    int lives = 3;
    int totalScore = 0;
    int level = 1;
    int requiredScore;
    int fallSpeed;
    int containerWidth;
    int levelScore;

start_level:

levelScore = 0;

if (level == 1){
    requiredScore = 5;
    fallSpeed = 100;
    containerWidth = 14;
}
else if(level == 2){
    requiredScore = 7;
    fallSpeed = 80;
    containerWidth = 12;
}
else if(level == 3){
    requiredScore = 12;
    fallSpeed = 50;
    containerWidth = 10;
}

```

```
120
121     int containerRow = SCREEN_HEIGHT - 2;
122     int containerLeft = (SCREEN_WIDTH / 2);
123     int containerRight = containerLeft + containerWidth;
124
125     clearScreen();
126     drawHeader(lives, levelScore, level);
127     while (levelScore < requiredScore && lives > 0){
128
129         int starRow = 4;
130         int starCol = (rand() % (SCREEN_WIDTH-2)) + 1;
131
132         while (starRow < containerRow){
133
134             if (kbhit()){
135                 int key = getch();
136
137                 if (key == 224){
138                     key = getch();
139                     if (key == 75 && containerLeft > 1){
140                         containerLeft -= 2; containerRight -= 2;
141                     }
142                     else if(key == 77 && containerRight < SCREEN_WIDTH - 2){
143                         containerLeft += 2;
144                         containerRight += 2;
145                     }
146                 }
147
148             }
149
150             gotoxy(0, 5);
151             drawScreen(starRow, starCol, containerRow, containerLeft, containerRight);
152             Sleep(fallSpeed);
153             starRow++;
154         }
155
156         if(starCol >= containerLeft && starCol <= containerRight){
157             levelScore++;
158             totalScore++;
159             gotoxy(0, 0);
160             drawHeader(lives, levelScore, level);
161         }
162         else{
163             lives--;
164             gotoxy(0, 0);
165             drawHeader(lives, levelScore, level);
166         }
167     }
168
169     if (lives <= 0){
170         clearScreen();
171         printf("\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t  GAME OVER!\n");
172         printf("\t\t\tYour Score: %d\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n", totalScore);
173         return 0;
174     }
175
176     if (level < 3){
177         clearScreen();
178         printf("\n\n\n\n\n\n\n\n\n\n\n\n\t\t\tLEVEL %d COMPLETE!\n\n\n\n\n\n\n\n\n\n\n\n\n\n", level);
179         Sleep(1500);
180         level++;
181         goto start_level;
182     }
183
184     clearScreen();
185     printf("\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t  YOU WIN THE GAME!\n");
186     printf("\t\t\tFinal Score: %d\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n", totalScore);
187
188     return 0;
189 }
```

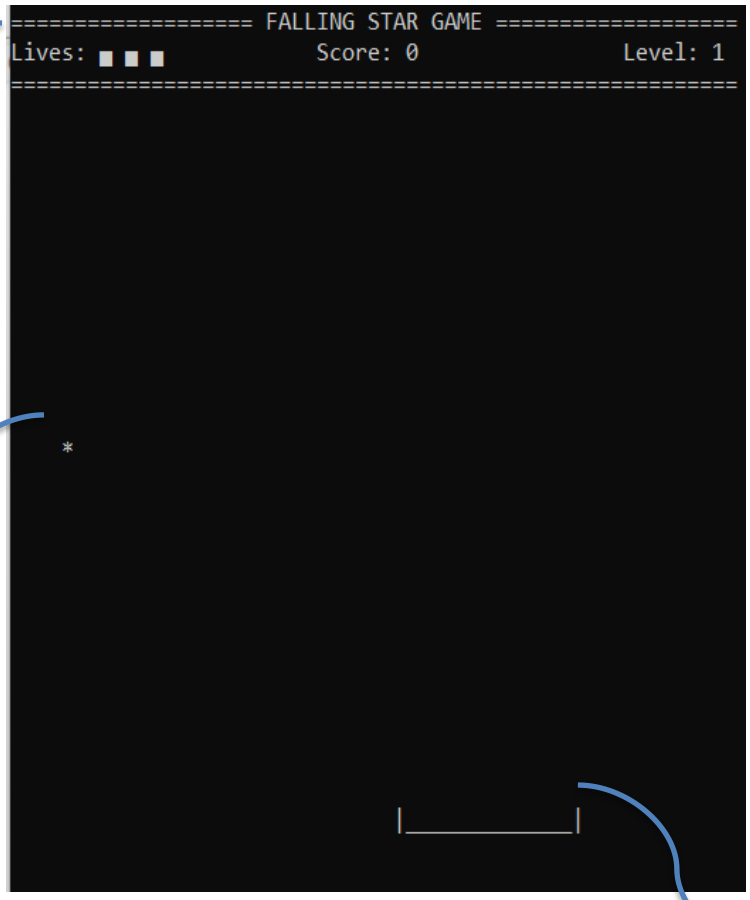
Sample Output



DISPLAY INTRO

LIVES:

Maximum of three lives (chances). It will decrease and updated in header if player miss to catch star.



HEADER:
Will show title of the game, level, score for that level and lives.

FALLING STAR

CONTAINER:

At the bottom of the game.

5. Testing & Results

TEST No.	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
1.	Player catches star	Score increases by 1	Score increased correctly	✓
2.	Player misses star	Life decreases by 1	Life decreased correctly	✓
3.	Player reaches required score	Level completes and next level starts	Level successfully advanced.	✓
4.	All lives lost	Game over message	Game ended with "GAME OVER"	✓
5.	All three levels completed	You win message	Game ended with "YOU WIN"	✓

The game performed successfully under all test scenarios. The falling stars were displayed smoothly, and the container responded instantly to the player's arrow key inputs. Scores and lives were updated accurately after each action, and the difficulty level increased progressively as intended, providing a balanced challenge. Overall, the program executed efficiently, running smoothly without noticeable delays.

6. Conclusion, Limitations & References

Conclusion

The Falling Stars game effectively demonstrates the use of essential C programming concepts such as loops, conditional statements, functions, and real-time input handling. The project successfully delivers a functional, engaging console-based game where difficulty rises gradually across levels. It strengthens understanding of animation logic, game design fundamentals, and interactive console programming.

Limitations

- The game is fully console-based and lacks graphical elements.
- Difficulty levels are fixed and cannot be customized by the player.
- The game does not save high scores or progress (no file handling).
- Certain functions used in the code, such as, Sleep(), getch(), kbhit(), and console cursor manipulation are specific to Windows and may not function correctly on other operating systems or compilers.

Future Enhancements

- Add file handling to store high scores or player progress.
- Introduce more levels, new objects, and improved difficulty settings.
- Convert the game into a GUI version using graphics libraries.
- Add sound effects for catching or missing stars.

References

- https://stackoverflow.com/questions/74446197/c-how-does-if-systemcls-systemclear-work
- https://www.geeksforgeeks.org/c/how-to-use-gotoxy-in-codeblocks/
- https://www.w3schools.com/c/ref_stdlib_srand.php
- https://learn.microsoft.com/en-us/windows/console/setconsolecursorinfo
- https://www.programiz.com/c-programming
- https://www.geeksforgeeks.org/c-programming-language/