Fundamentals of Programming

Iran university of science and technology

Capstone: Block Puzzle Game - Project Report

1. Introduction

The Block Puzzle Game is a console-based puzzle game created using C++. The goal of the game is for players to arrange blocks in predefined patterns. Upon achieving these patterns, the game levels up, providing users with new challenges. This project demonstrates core C++ concepts such as object-oriented programming, array manipulation, user input handling, and modular code design.

2. Objective

The primary objective of this project is to develop an engaging game where players use logical thinking to arrange blocks into specified patterns. The game becomes more challenging as the player advances through levels, each featuring increasingly complex block arrangements. This project aims to improve problem-solving and programming skills while offering an interactive experience.

3. Game Features

- Interactive Gameplay: Players interact with the game by arranging blocks on a grid to form specific patterns.
- Leveling System: Each level presents a new block pattern to form. Players level up when they achieve the pattern.
- Modular Design: The project is divided into multiple files for better organization, making the code easy to read, extend, and maintain.
- Increasing Difficulty: With each new level, the pattern becomes more difficult, ensuring continuous challenge for the player.

4. Detailed Game Mechanics

The game is structured around the idea of arranging blocks into specific patterns. Players can move blocks horizontally or vertically. Each block can be represented by a character or symbol, and when they align in the correct configuration, they form a pattern. The game then verifies whether the arrangement is correct. If it is, the player progresses to the next level.

Key Mechanics:

- Block Movement: Players input commands to move blocks along the x or y axis.
- Pattern Matching: Once blocks are in place, the game checks if they match the target pattern.
- Level Transition: After completing a pattern, the player automatically moves to the next level, which has a new, more complex pattern.

5. Code Structure

The code is broken down into multiple files for better modularity and maintainability.

Files:

- main.cpp: The entry point of the game, where the program is initialized and the main game loop runs. This file handles player input and displays the current game state.
- game_logic.cpp: Contains the core game logic, including block movement, pattern checking, and level progression.
- utils.cpp: Contains utility functions for handling common operations like input validation, displaying the game grid, and managing the user interface.

This modular structure makes it easier to update individual components, such as adding new game levels or modifying the block movement logic.

6. Algorithm Details

Pattern Matching Algorithm:

The core of the game is the pattern matching system. To detect whether a player has arranged the blocks correctly, an algorithm compares the current state of the blocks to a predefined pattern. This is achieved by iterating through the blocks and checking if their coordinates match the target pattern's coordinates. If they match, the level is completed.

Level Up Logic:

When the pattern is matched, the game moves to the next level by calling a function that loads a new pattern and increases the difficulty by adding more blocks or making the patterns more complex.

7. How to Run the Game

To run the Block Puzzle game, follow these steps:

1. Clone or Download: Download or clone the project from GitHub.
2. Compile the Code: Use a C++ compiler like GCC or Visual Studio:
3. g++ -o block_puzzle main.cpp game_logic.cpp utils.cpp
4. Run the Game: Execute the compiled file:
5. ./block_puzzle

8. Challenges Faced

1. Pattern Recognition:

One of the most challenging aspects of this project was implementing a reliable system to detect when the blocks form a valid pattern. I had to carefully design an algorithm that efficiently checks the block positions and compares them with the predefined pattern.

2. User Interface:

Since the game runs in a text-based terminal, designing a user-friendly interface was another challenge. I had to ensure the grid was displayed clearly and the user's input was processed smoothly. Input validation and error handling were crucial to avoid unexpected behavior.

3. Game Logic Complexity:

As the game progresses to higher levels, managing the increasing complexity of the block patterns and interactions required careful planning of the game's logic to keep it scalable.

9. Testing

Testing was done by running the game through multiple scenarios, including:

- Checking whether the blocks were correctly displayed on the screen.
- Verifying that the pattern detection algorithm worked as expected.
- Ensuring that the game transitioned smoothly between levels.

Additionally, user feedback was gathered to refine the interface and user input flow, making the game easier to play.

10. Performance Considerations

To ensure the game runs efficiently even with complex patterns:

- The game uses simple arrays and basic data structures, which allows for quick pattern matching.
- I avoided unnecessary memory allocation by using fixed-size arrays for block positions.

As the game doesn't require advanced graphics or heavy computation, performance wasn't a significant issue, but keeping the code simple and efficient was important for smooth gameplay.

11. Future Improvements

- Graphical Interface: I plan to transition the game to a graphical user interface (GUI) using libraries like SFML or SDL, which will make it more visually appealing and interactive.
- Multiplayer Mode: I'd like to add a two-player mode where players can compete or cooperate to solve puzzles.
- Save/Load Game: Implementing a system for saving and loading game progress will allow players to continue their game at a later time.
- Difficulty Adjustment: Adding options for the player to adjust the difficulty level manually, such as increasing the number of blocks or reducing the time allowed for each level.

12. Conclusion

This Block Puzzle game project has been an excellent opportunity to apply C++ programming skills in an interactive context. It helped me learn how to structure a game efficiently and manage complex game logic. The game's modular design ensures it can be extended in the future with more features, and the experience gained from handling challenges like pattern recognition and user input will be invaluable for future projects.