CSC10002 - PROJECT
THE MATCHING GAME

March 19, 2023

I Introduction

The Matching Game (commonly known as Pikachu Puzzle Game) includes a board of multiple cells, each of which presents a figure. The player finds and matches a pair of cells that contain the same figure and connect each other in some particular pattern. A legal match will make the two cells disappear. The game ends when all matching pairs are found. Figure 1 shows some snapshots from the Pikachu Puzzle Game.
In this project, we will develop a simplified version of this Matching Game by remaking the game with characters (instead of figures).

Figure 1: The Pikachu Puzzle Game [1]

[1] google.com

II Project Description

II.1 Standard Features

This mode contains the essential steps to make the game playable.

1. Game starting: Initialize a board with the given size while satisfying the following conditions
   - The total number of cells must be even.
   - The number of distinct characters is specified in advance.
   - For each character, determine an even number to define the number of occurrences for that character.
2. Any matching pair must satisfy the following conditions
   - Characters on the two cells must be identical.
   - When the cells disappear, their positions are replaced with blank spaces.
   - The matching pattern must be one of the below motifs
     - I Matching

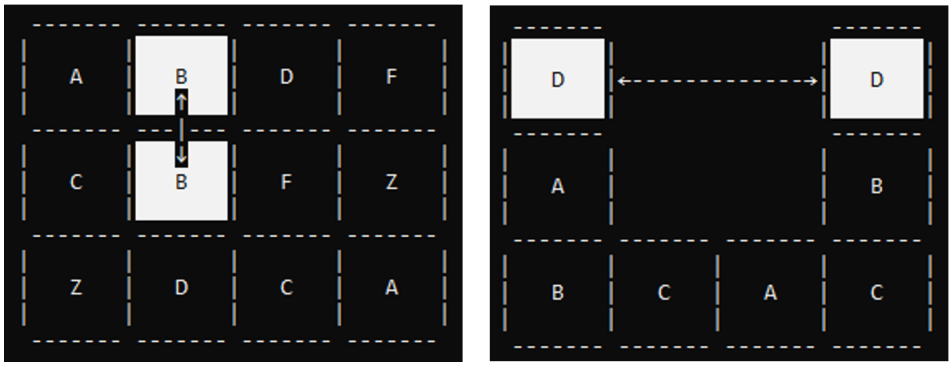

Figure 2: I Matching

     - L Matching

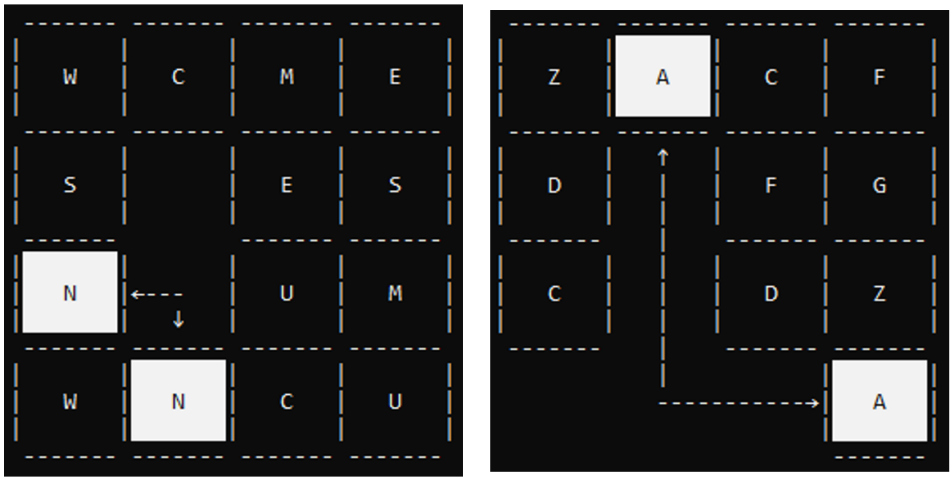

Figure 3: L Matching

– U Matching

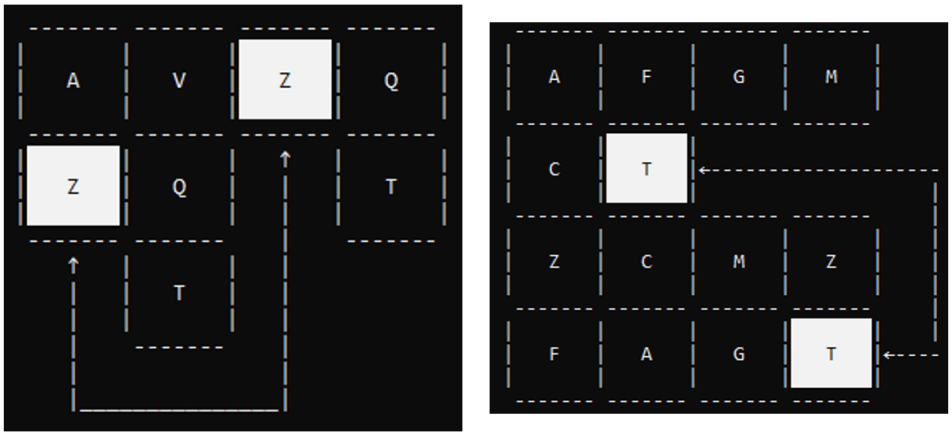

Figure 4: U Matching

– Z Matching

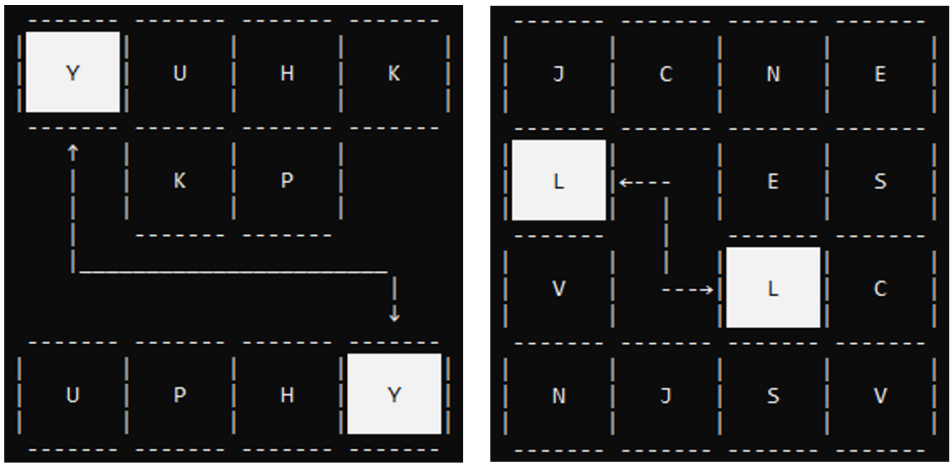

Figure 5: Z Matching

3. Game finishing: check the following conditions
   - Are there any cells left?
   - Are there any valid pairs left?

II.2 Technical requirements

- There must be a structural design for the board and player account.
- Game board content must be represented as a 2D-Pointer array.
- Save game must be stored as a binary file, which includes: player info, records, save stages, and other info.

II.3 Advanced Features

Players will experience the game better if you can add one (or more) of the following extra features to the game.

- Color and sound effects
- Visual effects (Figure 6)

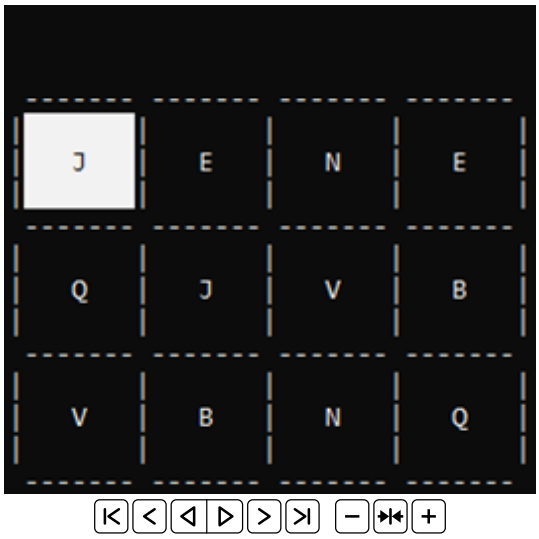

Figure 6: Visual Effect

- Background: You can design anything for a background. Then, when a matching pair disappears, the background content corresponds to those emptied cells.

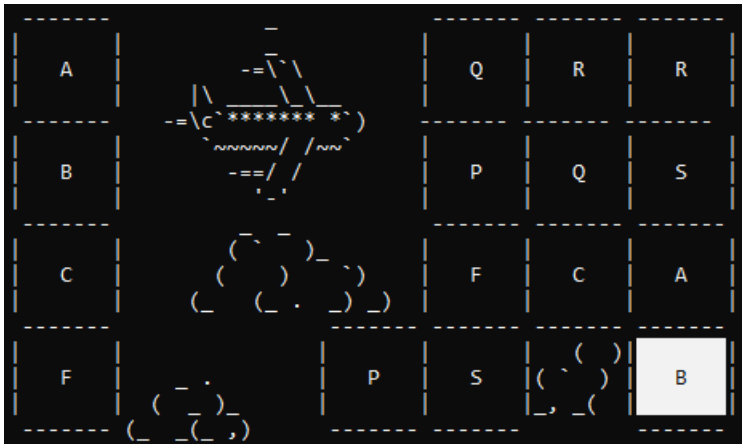

Figure 7: Game with background [2]

- Leaderboard: The top N players that finished their games in the shortest time will register their information to the Leaderboard.
- Move suggestion: Show players a valid match when they press the "Help" key.
- Game account and saved stages for each account (See II.4).

[2]Ref: `http://www.cplusplus.com/forum/general/58945/`

II.4 Extra Advance Feature

- Stages difficulty increase: You are requested to create a more difficult stage after players finish the previous one by sliding the neighboring cells (from the game stage) into the newly emptied spaces in a particular direction (left to right, up to down, etc.) after a matched (Figure 8). If you plan to implement this feature, you will have to represent your game board content in both a 2D-Pointer array and Linkedlist. A report on a comparison of running time and the complexity of each implementation is also required.

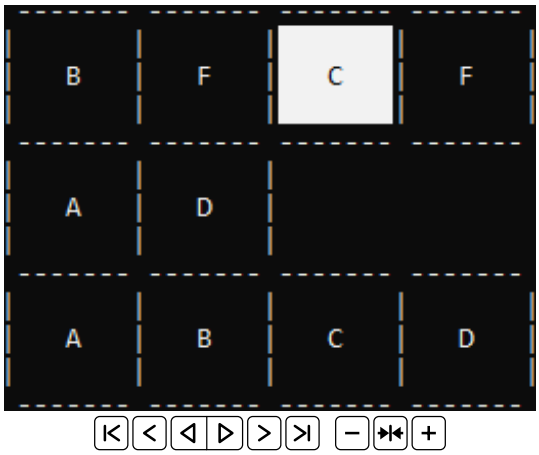

Figure 8: Example of left-sliding

- Save file "hacking": In this feature, a player is allowed to "hack" into their save file after they log in. If you plan to implement this feature, you will have to follow the predefined structures (for both programming and saving to file), which are defined below.

```
#define PADDING 500 // bytes
#define NAMESIZE 50
#define PASSSIZE 50
#define BOARDSIZE 999
#define URLSIZE 100


struct State{ //Representing a board state
     int p, q; // Size of the board game
     int p_, q_; // Current cursor position
     char board[BOARDSIZE]; // Current board state
     char file_background[URLSIZE]; // Link to background file. This variable's value is NULL if
         there is no current background
     // 500 byte NULL
};


struct Date{
    int dd, mm, yy;
};


struct Record{
     Date date; // Date of completed record
```

```
    int points; // points achieved
    // 500 byte NULL
};


struct savefile{
    char mask; // You are required to transfer all char-type variables by performing xor each with
        the mask-variable, bit-by-bit.
    char name[NAMESIZE]; // username
    char password[PASSSIZE]; // password
    // 500 byte NULL
    Record record[5]; // List of sorted best records
    State state[5]; // List of save state
};
```

  - You are allowed to customize your structures to fit the predefined ones.
  - A "Template.bin" save file is provided so you can determine if your reading and saving functions are legit. (*Username*: bhthong | *Password*: fit@hcmus)

III Submission and Grading

III.1 Submission

Your submission **ID1_ID2.zip**, which will be uploaded to Moodle, must contain the following files and folders:

- A `Source` folder contains all "*.cpp" and "*.h" source files
  - These files must be executable via the ".exe" file built from g++. Any other compiler used must be noted in the report, but not recommended.
  - There should be explication comments for your source code.
- A `report.pdf` report which includes all the below points:
  - Your name, ID, and class
  - A tutorial of how the game works (Use case diagram recommended)
  - An description of how to complete all the requirements in Standard Mode by using verbal descriptions or pseudo-code
  - A comparison of the complexity of the game's basic functions between using Pointer and LinkedList. (If students chose to implement the Stages difficulty increase Extra Advance Feature)
  - Any other remarks about your design and implementation for the Advanced Features
  - All references, e.g., weblinks and books, must be appropriately cited. If you discuss with your classmates or any high-level mentor(s), their names must be listed too
  - A demonstration video is encouraged.
  - **DO NOT** include your source code in the report. That is pointless.

- In your report, there must be an explanation for the necessity of these files and instructions on using them.
- This project description is associated with a checklist. Please make sure that you complete the checklist and include it in your submission.

III.2 Grading

- The lectures will evaluate your submission based on the implementation, report, and what is provided in the checklist.
- Your submission will get a zero score if it commits one of the following issues.
  - No checklist or report included
  - Regulations violated
  - The program is not runnable or malfunctioning.
- Plagiarism and Cheating will result in a 0 point for the entire course and will be subject to appropriate referral to the Management Board for further action.