Chapter 1. Project Overview

This project is a simplified version of the Pikachu Matching Game, runs on *amd64/x86_64 Linux Bash Terminal* and *x32/x64 Windows Terminal*. All of the source codes are contributed by Luu Nam Dat (Student ID 22127062) and Nguyen Huynh Hai Dang (Student ID 22127052). All of the source codes are integrated and maintained in the codebase by Luu Nam Dat. Any referenced code is cited in the beginning of its corresponding file, and is cited in section x.x.x.

The gameplay demonstration of this project can be found on youtube here.

Chapter 2. Project Hierarchy

2.1. Diagram explanation

The hierarchy of this project is visualized via a diagram in section *2.2*. The diagram divides the source code into modules, each module has its relations to other modules indicated by arrows. A sample diagram is shown in *figure 1*.

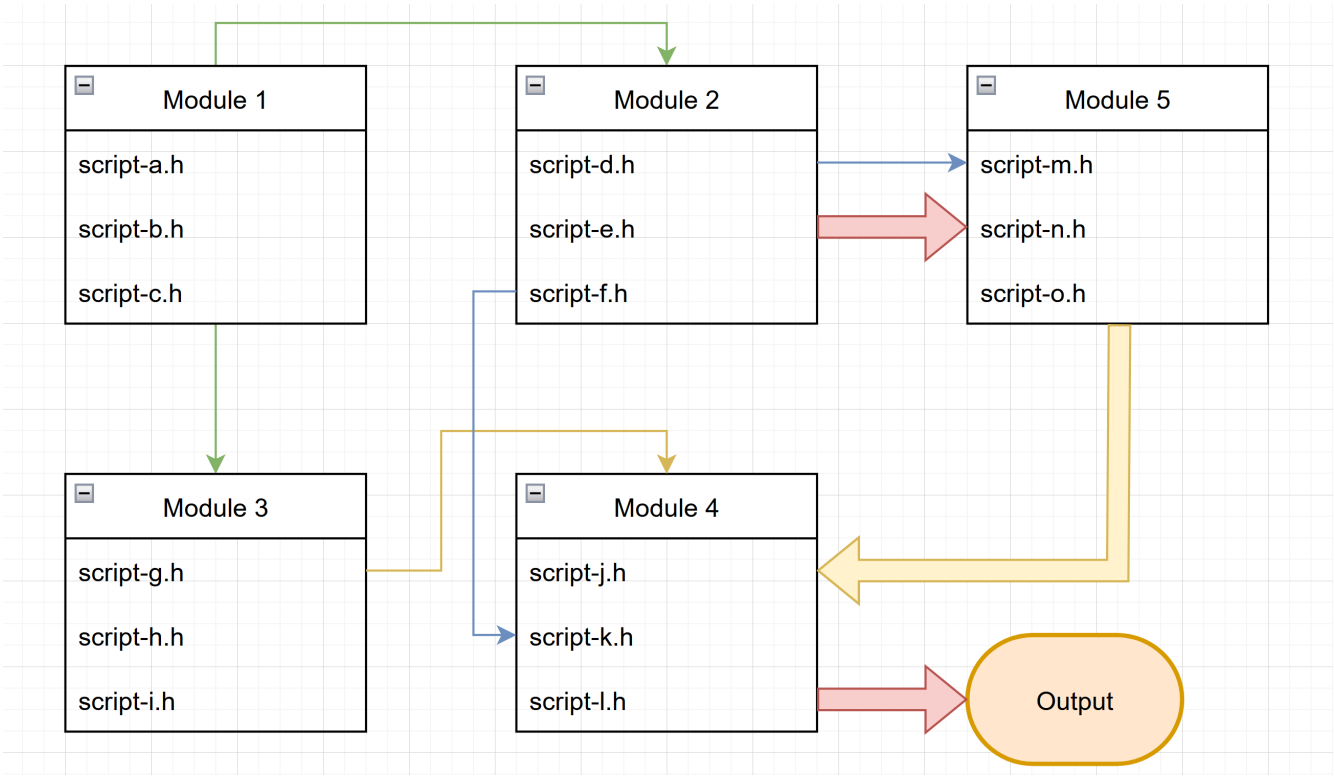

Figure 1: Sample Diagram

In *figure 1*, arrows that are connected to the bottom or the top of a box (like green and yellow arrows) indicates the relation of any item to a **module**. While arrows that are connected to the side of a box (like blue or yellow arrows) indicates the relation of any item to a **script**.

Additionally, a thin arrow from A to B shows that A exports parameters and methods (functions) to B. On the other hand, a thick arrow from C to D shows that C exports data to D.

2.2. Project Hierarchy

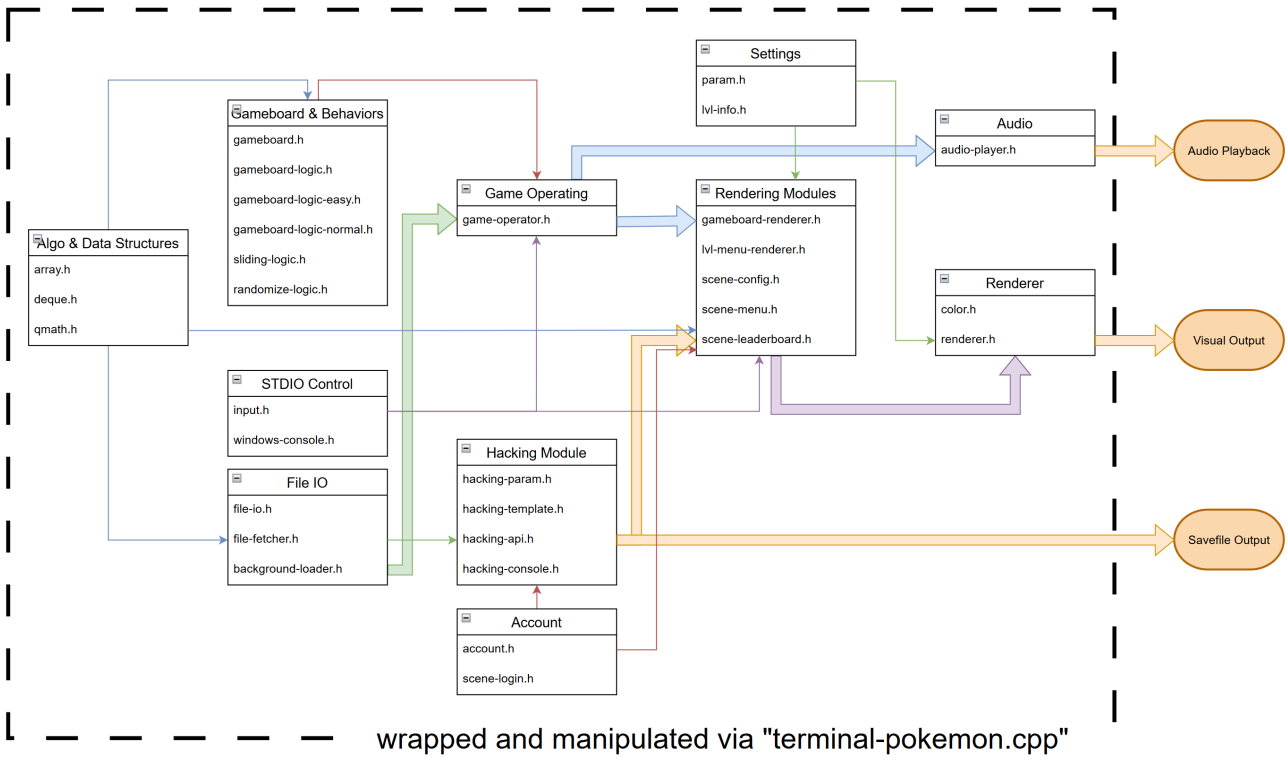

Figure 2: Hierarchy Diagram

This project consists of 55 source code scripts; 31 of them are ".h" files and 24 others, ".cpp" files. Each of them serves a specific role and is grouped into a module. This section briefly explains the usage of each module, then the detailed implementation shall be addressed in **Chapter 6 - Implementation**.

2.2.1. Algorithms & Data Structures

This module provides 2 simple data structures that are used thorough the project: Array and Deque; and a simple sorting algorithm used for ranking players: Bubble Sort.

2.2.2. STDIO Control

Offers functions to detect and extract keypresses. On Windows, this module provides additional controls to the terminal so as to optimize the poor performance of Windows' console.

2.2.3. File IO

Contains functions to list files in any directory (folder) and load some files into game objects.

2.2.4. Gameboard & Behaviors

Defines the gameboard structure & its operating logics. Matching rules, tiles sliding rules, and extra rules are defined here.

This module only implements the gameboard as a 2 dimensional array. In order to satisfy the additional requirement, a small game which implements the gameboard as a 2 dimensional linked list has also been developed and included in this project under directory `"sub/"`.

2.2.5. Audio

Emits sounds in fixed frequencies and durations.

2.2.6. Settings

Determines console resolution, quantity of difficulty levels and stages, as well as other constant values.

2.2.7. Renderer

This module provides a grid for other modules to draw on, and then it prints that grid onto the console.

Also supports printing with background color and foreground color.

2.2.8. Rendering Modules

Delivers functions to draw some data into the Renderer's grid. Such data are the gameboard, the leaderboard, the title menu, the level selector, etc…

2.2.9. Game Operating Module

Operates the game. This module wraps the gameboard, the input control, the renderer, and the audio player altogether to make the game playable.

2.2.10. Account

Account object definition & logging in interface.

2.2.11. Hacking Module

The module to fulfill the "savefile hacking" requirement.

Chapter 3. Features

This chapter shows which of the required features are implemented and where they are. How they are implemented is later explained in **Chapter 6 - Implementation**.

Feature	Completed	Implemented in
Game Starting	Yes	src/terminal-pokemon.cpp
Matching I	Yes	src/gameboard-logic-*.h
Matching L	Yes	src/gameboard-logic-*.h
Matching U	Yes	src/gameboard-logic-*.h
Matching Z	Yes	src/gameboard-logic-*.h
Game Finish Verification	Yes	src/gameboard-logic-*.h
Color Effect	Yes	src/color.h
Visual Effect	Yes	Rendering Modules
Sound Effect	Yes	src/audio-player.h
Background	Yes	src/gameboard-renderer.h
Leaderboard	Yes	src/scene-leaderboard.h
Stage Difficulty Increase	Yes	src/sliding-logic.h
Save file Hacking	Yes	Hacking Module
Board definition	Yes	src/gameboard.h
2d-Linkedlist Board	Yes	sub/gameboard.h
Account definition	Yes	src/account.h
Binary save file	Yes	src/hacking-api.h

Asides from **Source code aesthetics, Source code comments, Game performance,** and **Game aesthetics**—whose completion can not be marked as "Yes" or "No"—all other programming and technical requirements are all met.

Chapter 4. Dependencies

On Linux, this project has no dependencies other than the C++ standard library. On Windows, this project additionally requires the Windows API header file `"windows.h"`, which should already be available after MinGW installation.

All used libraries are:

- `cstdint` *(access to uint8_t, uint16_t,…)*
- `cstdio`
- `cstdlib` *(access to system)*
- `cstring`
- `ctime`
- `fstream`
- `iostream` *(to take advantage of the fact that iostream is not synced with stdio)*
- `random`
- `string`

Exclusive libraries for Linux:

- `filesystem` *(to list files in a directory)*
- `chrono` and `thread` *(to make a sleep function)*
- `termios.h`, `sys/ioctl.h`, `sys/select.h` *(to detect user keypress)*

Exclusive libraries for Windows:

- `conio.h` *(to detect user keypress)*
- `dirent.h` *(to list files in a directory)*
- `windows.h`

This project was built successfully on the following compilers:

- g++ (GCC) 12.2.1 20230201
- g++ (Debian 10.2.1-6) 10.2.1 20210110
- g++ (MinGW.org GCC-6.3.0-1) 6.3.0

The build scripts for this project are also provided. To build the game, execute `build.sh` on Linux or `build.bat` on Windows. To build the sub-version of the game which implemented the gameboard as a 2 dimensional linked list, execute `build-linkedlist-game.sh` on Linux or `build-linkedlist-game.bat` on Windows.

Chapter 5. References

Before going to the detailed implementation, all references are firstly cited, as well as why they are necessary to this project is also addressed.

Beyond this chapter, all line of codes belongs to Luu Nam Dat and Nguyen Huynh Hai Dang.

Code References

Source	Application
"Bash - adding color" http://www.andrewnoske.com/wiki/Bash_-_adding_color	Add colored text to the console output
Morgan McGuire "_kbhit() for Linux" https://www.flipcode.com/archives/ _kbhit_for_Linux.shtml	Detect user keypress on POSIX operating systems
"How can I get the list of files in a directory using C or C++" https://stackoverflow.com/questions/612097/how-can-i-get-the-list-of-files-in-a-directory-using-c-or-c	List files in a directory on different operating systems

Documents

Source	Application
"Standard C++ Library reference" https://cplusplus.com/reference/	Documentation of the C++ standard library
"Windows API Reference" https://learn.microsoft.com/en-us/previous-versions//aa383749(v=vs.85)	Documentation of the `<windows.h>` header file
neilharan "OpenAFIS - A high performance C++ fingerprint matching library" https://github.com/neilharan/openafis	This repository's source code organization is applied in our project

Chapter 6. Implementation

This chapter explains how each module was implemented and what algorithms and techniques were used in the process.

6.1. Algorithms & Data Structures

This module consists of 3 header files: "qmath.h", "deque.h", and "array.h"

"qmath.h" *(abbreviation of "quick math")* is designed to provide handy mathematic functions and simple algorithms such as a sorting algorithm. Currently, it only has a simple Bubble Sort function, but that alone is already enough for this project.

"deque.h" implements a double-ended queue via a linked list. It is a data structure that can dynamically be expanded or contracted on both ends (either front or back) in $O(1)$. It is possible to access/add/modify/remove arbitrary elements in this data structure as well, and "deque.h" also provides all necessary methods to do that. But these methods are never used in this project since accessing an arbitrary element in a linked list has $O(N)$ time complexity, where N is the number of elements in the list.

"array.h" provides a wrapper for C++ dynamic array. In C++, creating an dynamic array via a pointer (for example: `type* arr = new type[length]`) can be quite inconvenient. The created object does not keep track of its length on its own, and it needs to be deallocated manually via a `delete` keyword. Using built-in structures like `std::vector` and `std::array` are encouraged thereof; but to reduce the project's dependency, "array.h" is introduced to replicate the usage of `std::array`.

While "qmath.h" is only used in the leaderboard feature, "deque.h" and "array.h" are robust containers which are used thorough the project as a substitute for static array, pointer array, `std::array`, `std::vector`, `std::queue` and many more built-in containers.

6.2. STDIO Control

This module consists of 2 header files: "input.h" and "windows-console.h"

"input.h" is simple. It offers a function to wait for the user's keypress then return which key is pressed. Works on Windows, UNIX-like and POSIX operating systems.

"windows-console.h" provides exclusive functions for Windows so as to optimize the poor performance of the Windows' Terminal. It allows a program to re-render arbitrary cell on the console, which reserves computing resources from being wasted into re-rendering cells that have not been update after a frame. "windows-console.h" is possible thanks to the Windows' API header file `<windows.h>`

Regarding of how poor the Windows' Terminal performs without "windows-console.h", a small benchmark is proposed in order to measure the fps (frames per second) of this game with "windows-console.h" removed. The game has the resolution of 120×40 (measured in characters). The benchmark runs on only one computer with a *11th Gen Intel i5-11320H (8) @ 3.187GHz CPU*, but on different operating systems and terminal emulators. The result is shown in the table below:

Platform	FPS
Native Arch Linux x86_64 (Kernel: 6.1.11-arch1-1)	144.7
Debian GNU/Linux 11 (bullseye) on Windows 10 x86_64	91.4
Alacritty terminal emulator on Windows 11	91.2
Windows Terminal (cmd.exe/powershell.exe)	0.3

6.3. File IO

(tmp) Contains functions to list files in any directory (folder) and load some files into game objects.

6.4. Gameboard & Behaviors

(tmp) Defines the gameboard structure & its operating logics. Matching rules, tiles sliding rules, and extra rules are defined here.

(tmp) This module only implements the gameboard as a 2 dimensional array. In order to satisfy the additional requirement, a small game which implements the gameboard as a 2 dimensional linked list has also been developed and included in this project under directory `"sub/"`.

6.5. Audio

(tmp) Emits sounds in fixed frequencies and durations.

6.6. Settings

(tmp) Determines console resolution, quantity of difficulty levels and stages, as well as other constant values.

6.7. Renderer

(tmp) This module provides a grid for other modules to draw on, and then it prints that grid onto the console.

(tmp) Also supports printing with background color and foreground color.

6.8. Rendering Modules

(tmp) Delivers functions to draw some data into the Renderer's grid. Such data are the gameboard, the leaderboard, the title menu, the level selector, etc…

6.9. Game Operating Module

(tmp) Operates the game. This module wraps the gameboard, the input control, the renderer, and the audio player altogether to make the game playable.

6.10. Account

(tmp) Account object definition & logging in interface.

2.2.11. Hacking Module

(tmp) The module to fulfill the “savefile hacking” requirement.