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*.

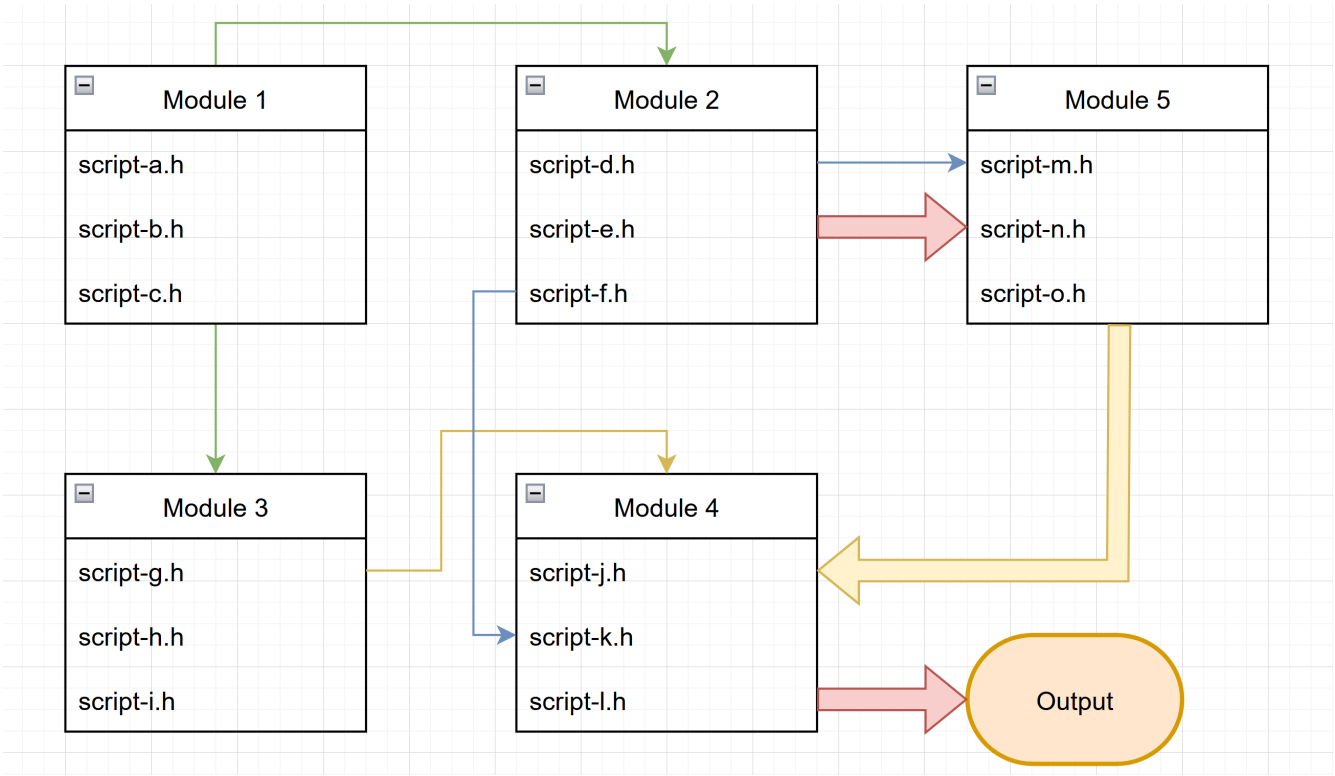

Figure 1: Sample Diagram

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

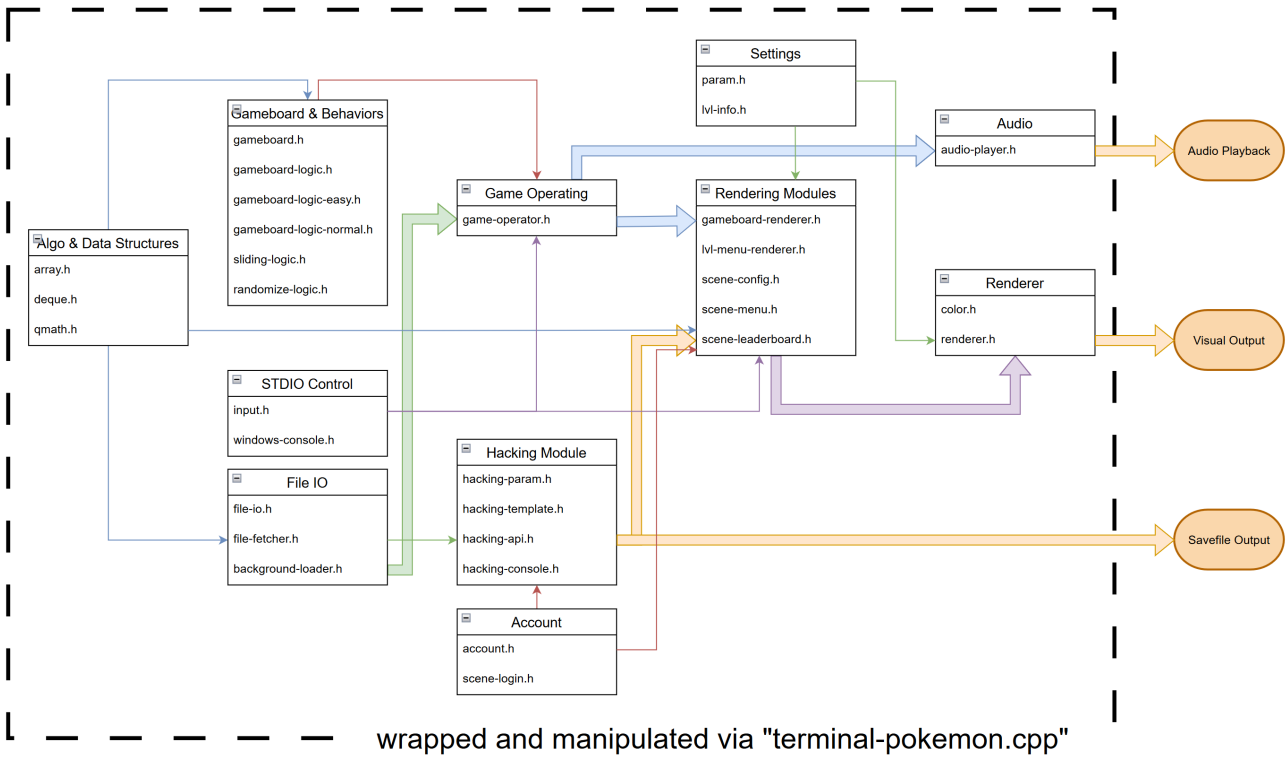

Figure 2: Hierarchy Diagram

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++ (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.