

Project Report

Giulio Salvi and Jacopo Paradisi

15th September 2025

Project Structure:

The project root folder has several files and subfolders.

File or Folder Name	Description
bin/	It contains all the compilation outputs: the object non-executable files are placed in the subfolder 'objects' while the executable files are saved in the subfolder 'exec'
docs/	It contains the Doxygen configuration file ('Doxyfile') and its output in the subfolder 'html'
h/	It contains all the header files
src/	It contains all the source files
logs/	It contains all the logs file generated by the game
cli_schema	A file describing the schema of CLI and its options
COPYING	A file containing a copy of the GNU GPLv3 license
docs-mainpage.md	A file used by Doxygen for generating the documentation's main page
icon.png	The project icon, used by Doxygen
Makefile	The make file used for setting up the project, compiling and generating the documentation
Project Specification.pdf	A file containing the specifications of the project
README.md	The README file for the repository

The code of the whole project is divided into modules as follows:

- ANSI module: it is formed by the files 'h/ansi.h', 'h/ansi_const.h' and 'src/ansi.c'; it provides implementations of the most popular ANSI escape codes.
- Utility module: it is formed by the files 'h/utility.h' and 'src/utility.c'; it provides various utility functions for strings and numbers.
- CLI module: it is formed by the files 'h/cli.h' and 'src/cli.c'; it provides a command-line interface for the program with which users can manipulate the game configuration.

- Game Configuration module: it is formed by the files 'h/config.h', 'h/config_file.h', 'h/game_configuration.h', 'src/config.c' and 'src/config_file.c'; it provides function for saving game configuration to a file, load the game configuration file, load the game configuration from the CLI options and for asking to the user the setting of the game configuration in the terminal.
- Vector module: it is formed by the files 'h/vector.h' and 'src/vector.c'; it provides a dynamically-allocated vector structure and function for its manipulation.
- TUI module: it is formed by the files 'h/tui.h' and 'src/tui.c'; it provides a graphical user interface on the terminal.
- Logs module: it is formed by the files 'h/logs_configuration.h', 'h/logs.h' and 'src/logs.c'; it provides a dynamic way for the program for printing debug informations, either to the terminal or to a log file, in a verbose behavior.
- Main module: it is formed by the files 'h/main.h' and 'src/main.c'; it is the implementation for the game logic.

The full documentation can be found at [this link](#). The project repository can be found at [this link on GitHub](#).

About the Makefile:

In the Makefile there are present several rules: the make rule 'setup' helps to set up correctly the workspace, the make rule 'gen-docs' generates the documentation using the Doxyfile and the make rule 'go' compiles and runs at-fly the project. Running the make rule 'setup' is highly recommended in order to let the game run correctly.

Organization of the team:

The team was born after Giulio was suggested by the professor to form a group.

With the already advanced state of the project he was working on, we subdivided the remaining work on modular and independent tasks such that everyone could develop their assigned features without caring about the overall integration.

When the individual development was completed, through a meeting we connected all the modular features focusing on smoothing our individual code to better integrate with the overall codebase.



As a result, we present how the modular tasks were individually subdivided:

Module	Author
Ansi Codebase (pre-existing before the group birth)	Giulio
Game Base Mechanics (pre-existing before the group birth)	Giulio
Game Command Line setup (CLI and configuration file)	Giulio
Game Graphics and Terminal Arrangement (TUI)	Jacopo

Overall, we both guided our efforts into solving the issues in each module, even if not personal, to improve the state of the project.

Main difficulties and how they have been handled:

Overall, the project's main difficulty was the graphical user interface because making it work across different platforms took a relevant amount of time.

As a result, we made it work except for legacy terminal's hosts which are not capable of such tasks in the terminal: going more in depth, the problem involved some ANSI escape codes which were well beyond the technologies of the older versions of the Windows terminal.

Hence, we simply decided to avoid implementing a solution for a problem which wouldn't affect a regular user since the absolute majority of people today have a terminal capable of interpreting ANSI escape codes.

Beyond small issues among platforms, the integration between the modules was successful even if a little bit tough to achieve, and our strategy worked well and presumably better than directly working on code not written personally.

Concluding, cooperation and teaming played a big role in our group which, starting from an individual codebase, built around it some useful features, while keeping the overall project compact.