

General Information

Allocated time: 2 labs.

Due date: Due @ 23:59 December 5, 2025

Assignment weight: 8%

OBJECTIVES:

This assignment is to practice coding using the C language.

OVERVIEW

In this assignment, you will implement a game using the Ncurses library in the C language.

Introduction

In this project, you will create a simple 3rd person maze game that can be played within a terminal environment. To accomplish this, use the Ncurses library to control terminal IO. You must plan your game as well as your milestones. The project plan and the proposed milestones are essential parts of your **Design document, which will be part of the graded submission.**

Groups

This project will be for group work, where it is a part of your task to form a group of 2 students.

All group members must be in the same lab group.

You will be assigned randomly to a group if you did not form a group within the session.

Getting started

Create a directory A2.

Develop the basic idea for your game. You need to plan the following:

- The goal for each level.
- The overall goal.
- What do the levels look like?
- What are items in the game?

Place the answers to these questions in a README.md file under a section named “GAME PLAN”.

Splash Screen/Home Screen

First, create a main program that initializes the Ncurses library.

Next, create a splash screen for your project. It should contain the Game’s name, some context for the game, as well as any instructions on how to play the game.

Add a call to your splash screen in your main program. In the splash screen, you must have a selection menu, which will give the user the option to select any level or exit the game.

Game Levels

There should be 2 levels with increasing difficulty from easy to hard. Create a function named `display_level (int i)`, which displays level `i` on the screen. Levels should be 30x80 (to fit into a standard terminal). Levels must include boundaries and objects other than the main character.

Levels' specifications

- The main character must move with the arrows. “The allowed motion will be based on your level design.”
- Set up the general algorithm such that the game loops continuously, waiting for keyboard input. Initially, set it up such that if **q** is pressed, a message is displayed asking if the player wants to quit (y/n). If they respond positively, then the game ends. If they respond negatively, the game continues.
- In addition, if **p** is pressed, then the game pauses, and a message is displayed that the game is paused. When the game is paused, no other keys will be recognized except if **p** is pressed again, continuing the game.
- Add at least 2 elements that the player needs to interact with (e.g., press “o” for opening a door; press “X” for picking some object up, or press “space” to shoot an enemy.)
- Add a non-role-playing character (NPC) that has some purpose in the game. The NPC must move around the game (the movement algorithm is up to you) and must be able to interact with the main character (ex., press “t” to talk with the NPC).
- Once all elements are in place, one should be able to play the game from start to finish. Full points will be awarded here based on how well everything fits together as a whole (gameplay).

Documentation

Finally, create documentation for your game using Doxygen. All files and functions should be well-commented to produce good documentation for the project. A dedicated target must be included in the makefile to create the Doxygen documentation (from a pre-existing configuration file) within a `doc` folder to create HTML documentation and LaTeX files.

Makefile

The makefile of the project must contain the following targets:

1. A target with the app name to generate the executable for the game, showing zero errors and zero warnings.
2. A **clean** target to remove all the intermediate files.
3. **Mem_valgrind** to check the memory utilization.
4. A **debug** target to produce an executable in the debug mode.
5. A **Doc** target to generate the project documentation.

Submission and Grading

You will submit directory A2 with the following structure.

| A2 | | | |
|--|--------------------------------------|-----------------------------|---------------------|
| 1. src a. *.c b. *.h c. Makefile | 2. doc a. LATEX b. HTML | 3. bin a. appGame | 4. README.md |

| Item | mark |
|--|------|
| Group formation | 5 |
| Makefile | 5 |
| Clean compilation (no errors, warnings, or memory leaks) | 5 |
| Documentation (README, HTML) | 20 |
| Splash Screen | 10 |
| Two levels with NPC, Quit, and Pause functionality | 30 |
| Game Play of all levels successively | 25 |
| Total Submission | 100 |

Your solution must be compressed before submission and named **GNN_A2.tar.gz**. The group number **NN** will be assigned to you by your lab instructor. The contents of your tar.gz file must expand into a directory called A2.