Source-code documentation:

Source code is documented using Doxygen and is found in file refman.pdf.

Overview: what the software does, what it doesn't do? (this can be taken/updated from the project plan)

The software runs a game, where the player controls a character on the SFML screen. The program simulates a randomly generated dungeon environment with random types of enemies, where the player character moves around in. The program simulates enemies that attack the player and can be killed by the player. The software terminates when either the SFML window is closed or when the “Quit Game” option is chosen in the game main menu screen. There is a collider in the program to handle collisions between player, enemies, walls and projectiles. The player can collect potions and coins when killed enemies drop them. In a nutshell, the game has the following:

- Simple 2D graphics.
- Moving through corridors and rooms.
- Combat between the player and several types of monsters.
- Collectibles which can be used later(potions and coins).
- Winning and losing conditions.
- Randomly generated dungeons.
- In-game sound effects and music.
- Animated enemies that drop random items on death.
- Enemies that are somewhat unpredictable, because of the randomization of their movements and interactions.
- Player HUD for keeping track of health and potions.
- Main menu screen for restarting the game after game over.
- Collider to handle collisions between different in-game objects.

The program doesn’t implement all additional features such as character levelling, Passive NPCs (NPCs that aren’t enemies), interactive inventory or map, quests or events and ability to craft/modify items. There are also some other nuanced aspects of the game that could be present in the implementation of already existing features of the game. Some examples of such features include:

- The randomly generated map does not have circular room entrances, which means the player can only progress forward or in a dead end, but the player can never go in circles when traversing rooms.
- The enemies may sometimes camp at room entrances. This may or may not be considered a desirable implementation of enemies depending on the individual player preferences.
- Although the amount for a specific type of enemy is randomized between each room, the enemies themselves are not randomized. This means that ranged

enemies are copies of each other and melee enemies are also copies of each other; This means that an enemy of a certain type never has unique attributes such as movement speed, firerate or damage etc.
- Although the enemies drop random potions, the potions themselves are not randomized.
- Enemies do not drop weapons and there is no other ways of obtaining new weapons, there is only one weapon that the player uses for the rest of the game.
- There is only one shape for a room, which is the empty rectangular room with one block wide entrances.
- There is no animation for projectiles or the player, only the enemies are animated.
- Some of the sound effects are not implemented, such as using a potion.
- The ranged enemies will fire projectiles offscreen, which may or may not cause lag depending on how the map was randomized. However, there is a timer for projectiles, which means that projectiles will get deleted after some amount of time has elapsed, which reduces lag. In that sense this may or may not be considered to be a problem.

Software structure: overall architecture, class relationships (diagram very strongly recommended), interfaces to external libraries

The software structure diagram is located in the doc folder.

Instructions for building and using the software

Instructions for building and running the program are located in the project doc folder as a pdf named: “Instructions for building and using the software”

How to use program/How to play the game:

Player movement: WASD keys

Player shooting: Arrow keys

Using potions: Number keys 1 & 2

Navigating options in the menu screen: Click with left mouse button

Objective of the game is to kill the final boss of the game. The losing condition of the game is the player dying from taking too much damage from enemies. When this happens the game will go back to the main menu screen.

How to compile the program ('make' should be sufficient), as taken from git repository. If external libraries are needed, describe the requirements here

How to use the software: a basic user guide

For more detailed information for compiling and using external libraries, check the “Instructions for building and using the software” pdf file in the project doc folder.

Here is all that information in short:

The only external library used in the program is the SFML library, which means SFML and mingw32 have to be installed.

How to compile the program: Call make command in the terminal.

Testing: how the different modules in software were tested, description of the methods and outcomes

Testing was done by running the main function of the program. After implementation of a new feature, it was tested by playing the game. This way it was quickly seen how the game behaves and making the right adjustments was easy. Debugging was done simply by looking at the output of the variables used.

After testing that the updated functionality in the game works(for ex. doesn’t crash the program), the corresponding branch, where the change was made was merged into the master branch.

Work log: This might be a simplified/restructured version of the weekly meeting notes file. Detailed description of division of work and everyone's responsibilities

For each week, description of what was done and roughly how many hours were used, for each project member.

Work log for each group member is located in the doc folder of the project.