1. Overview

Basic features:

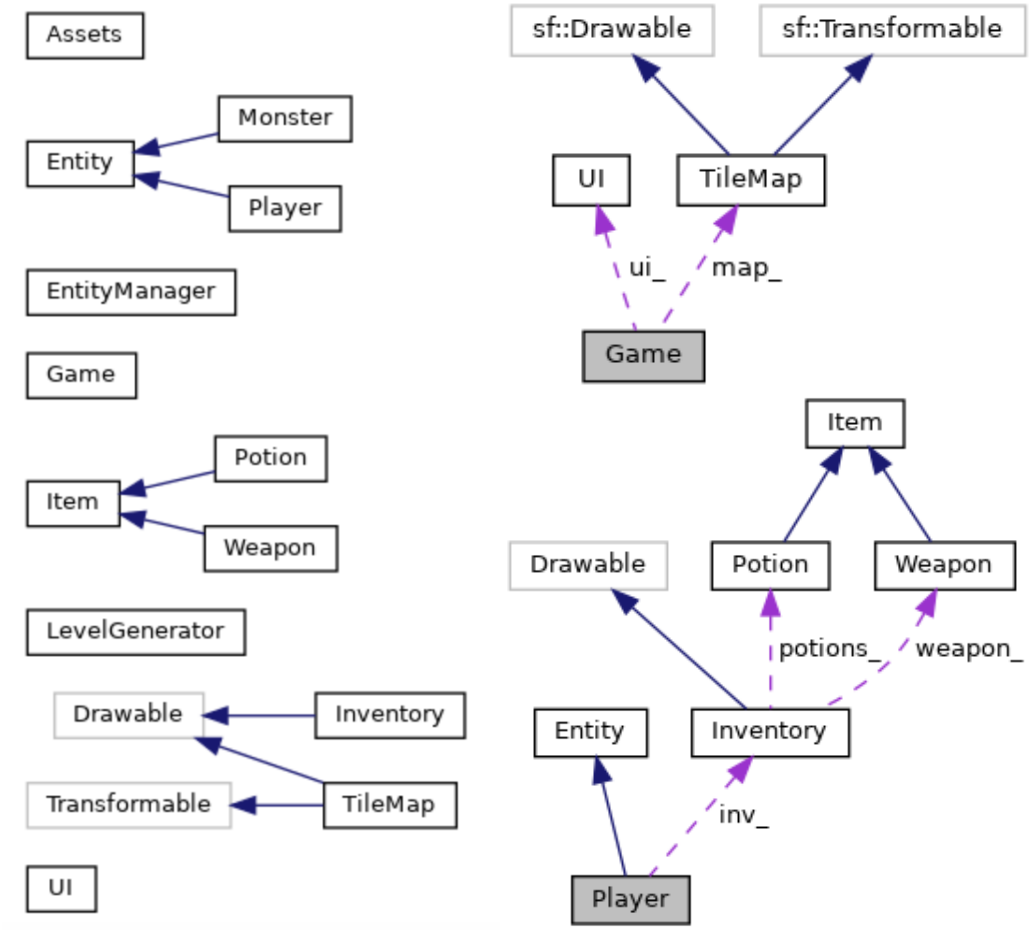
- **Simple 2D graphics** – SFML library was used for the 2D graphics. Assets were created with use of AI + photoshop.
- **Moving through corridors and rooms** – The player can move in four directions and travel from room to room by walking into doors (use WASD to move).
- **Combat between the player and different types of monsters**:
    - Basic attack - performed by walking into enemy, the player deals damage based on the weapon's stats.
    - Special ranged attack - hits everyone around the player and then has a cooldown (cannot be used for several turns, press E to use).
    - Types of monsters:
        - Skeleton – weak and fast enemy.
        - Orc – tougher version of a skeleton, has more health.
        - Orc boss – slow and tanky final challenge.
- **Collectibles** – can be used by the player, the player can pick up health potions by walking into barrels and upgrade the weapon by stepping onto chests (press Q to use health potion).
- **Inventory** – the player has two counters showing remaining health potions and the weapon's statistics.
- Some **sort of progression** (winning & losing conditions)
- To win you need to defeat the boss after several rooms with minions, if you die – the game is over.

Additional features:

- **Randomly generated monsters and items** – monsters randomly spawn in each room, they have random spawn location and number, random number of items are spawned on random tiles too.
- **Character leveling** (skills & abilities) – killing monsters gives experience points, when the player reach the needed amount, they level up. Level-up grants the player higher health points and better special ability damage, the level progress can be tracked with the XP bar.
- **Sound effects** – there are sounds for fighting, different for different monsters and the player, sound of dying, game over, level up etc.
- **Additional UI elements** – there are status bars, tracking the player's health, progress in leveling and the loading of the ability. There is also Inventory displayed with number of potions and statistics of the weapon. Each enemy has its health status displayed above their head and there are some additional UI elements informing about the game status like a pop up text for level up, or for game over or victory.

2. Software structure

Relationships between classes is shown below.



3. Instruction for building and using software.

Compile the build and run.

How to compile the program.

Compiling is done by the CMake file. SFML library is automatically pulled from the net, no need for manual installation.

How to use software.

After the game starts, you can walk using WASD, healing with potion using Q, using special attack with E. To attack an enemy, you must walk into it. To find items you must walk into chests/barrels. To go to the next room, you must go to the door tile. The goal is to find and beat the boss.

4. Testing

The software can be tested using unit tests for the classes that it was relevant. The tests executable is built in the same directory as the main game build.

5. Worklog

Division of work:

Dementev Viktor:

Responsible for creating graphical part of the game and CMake file. Made Assets, used to load all assets (different class is used to load floor tiles). Responsible for creating Tilemap class, but Szymon did it instead. Responsible for all assets for the game. Responsible for level – to – level transition mechanic and random level generation. As well was mostly responsible for design of game mechanics.

Abramczyk Szymon:

Improved CMake and created the Game class with basic player movement. Solved file path issues and implemented turn-based player movement. Created the TileMap class and expanded the Assets class. Created the Inventory class and improved memory management. Added sound effects, unit tests, and automatic asset loading. Refactored the code, created specialized classes, and improved code quality. Made various small refinements and adjustments to the project.

Risto-Pekka Siponen:

Responsible for combat gameplay. Created Entity, Monster and Player classes. Implemented combat between players and monsters. Created different types of monsters (Undead, Orc, Boss), as well as different attack types (normal and ranged attacks). Implemented player level-up/progression system. Added basic doxygen documentation.

Semenova Anastasiia:

Established an Item class incorporating the attribute "amount" to denote the quantity of the item. The class includes a method "use_item()" for potential functionalities. Further specialization is achieved through inheriting classes like Weapon, Armour, and Potions, each tailored to specific item categories with unique properties and use cases. After that, the team made their modifications.

Weekly work for each member.

- Week 1
    - **Dementev Viktor:** Created CMake file and worked in group to run at least something. Around 5 hours of work.
    - **Abramczyk Szymon:** Improved CMake to automatically use all .cpp files, made a basic implementation of the Game class with input handling and basic movement for the player. Around 4 hours of work.
    - **Risto-Pekka Siponen:** Fixed issues with Visual Studio, did research on SFML. Around 4 hours of work.
    - **Semenova Anastasiia:** Reviewed and understood the existing script provided. Discussed potential improvements. Tried to make personal small draft plan for develop this part and discussed with team. SFML researching. 4 hours of work.

- Week 2:
    - **Dementev Viktor:** Made Assets class and SpriteInfo Class, tried to solve the issue with file path being incorrect. Around 6 hours.
    - **Abramczyk Szymon:** Solved the issue with file path for the content, added turn based moving for the player. Around 4 hours of work.
    - **Risto-Pekka Siponen:** Started research of combat and implementation of entities. Around 3 hours of work.
    - **Semenova Anastasiia**: Implemented proposed changes to the Item class. Tested the modified Item class. 5 hours of work.

- Week 3:
    - **Dementev Viktor:** Made skeleton for the world class – soon abandoned. Made improvements and fixes to CMake file to load assets properly to build. Around 3.5 hours.
    - **Abramczyk Szymon:** Edited the CMake to copy all .png files to build folder, added a TileMap class responsible level loading and displaying using tile maps from a tile set. Adjusted tile loading to our own tile set. Added an Inventory class and extended Assets class to handle different types of assets than just sprites. Around 8 hours.
    - **Risto-Pekka Siponen:** Added Entity, Monster and Player classes. Implemented monster pathfinding and basic combat gameplay. Around 6 hours of work.
    - **Semenova Anastasiia:** Implemented modifications for Weapon, Armour, and Potion classes as per the proposed design. Around 8 hours

- Week 4:
    - **Dementev Viktor:** worked with Szymon to create Tilemap class and draw the level. Started drawing assets. Around 7 hours.
    - **Abramczyk Szymon:** Changed the movement to be entirely tile based. Added usage of smart pointers to improve the game's memory management. Integrated the Inventory class to the player. Made small improvements. Around 7 hours.
    - **Risto-Pekka Siponen:** Added additional monster types, and improved combat features. Added victory and defeat conditions for the game (Victory and Game Over screens). Added basic UI for player information. Around 8 hours of work.

    - **Semenova Anastasiia:** Continued with develop implementation Weapon, Armour, and Potion classes. Tested it. Around 9 hours.
- Week 5:
    - **Dementev Viktor:** made level – to – level transition, random level generation, drew all needed assets. Around 12 hours of work.
    - **Abramczyk Szymon:** Added sound effects to the game. Wrote unit tests to test basic functionalities of the classes. Improved Assets class to automate the loading of assets. Refactored Item, Weapon and Potion classes to integrate them with other parts of the program. Improved the game loop to not do redundant things. Refactored code to move some responsibility from the Game class to new smaller classes like UI or LevelGenerator. Refactored code for better encapsulation, added EntityManager class. Improved formatting of the code to be consistent. Made a lot of small improvements and adjustments. Around 25 hours.
    - **Risto-Pekka Siponen:** Implemented player progression system. Added random monster generation for each level. Improved game UI. Created basic Doxygen documentation. Around 13 hours of work.
    - **Semenova Anastasiia:** Worked on integration Item, Weapon and Potion class and debugging them, but finally Szymon refactored them and modified. Around 7 hours of work.