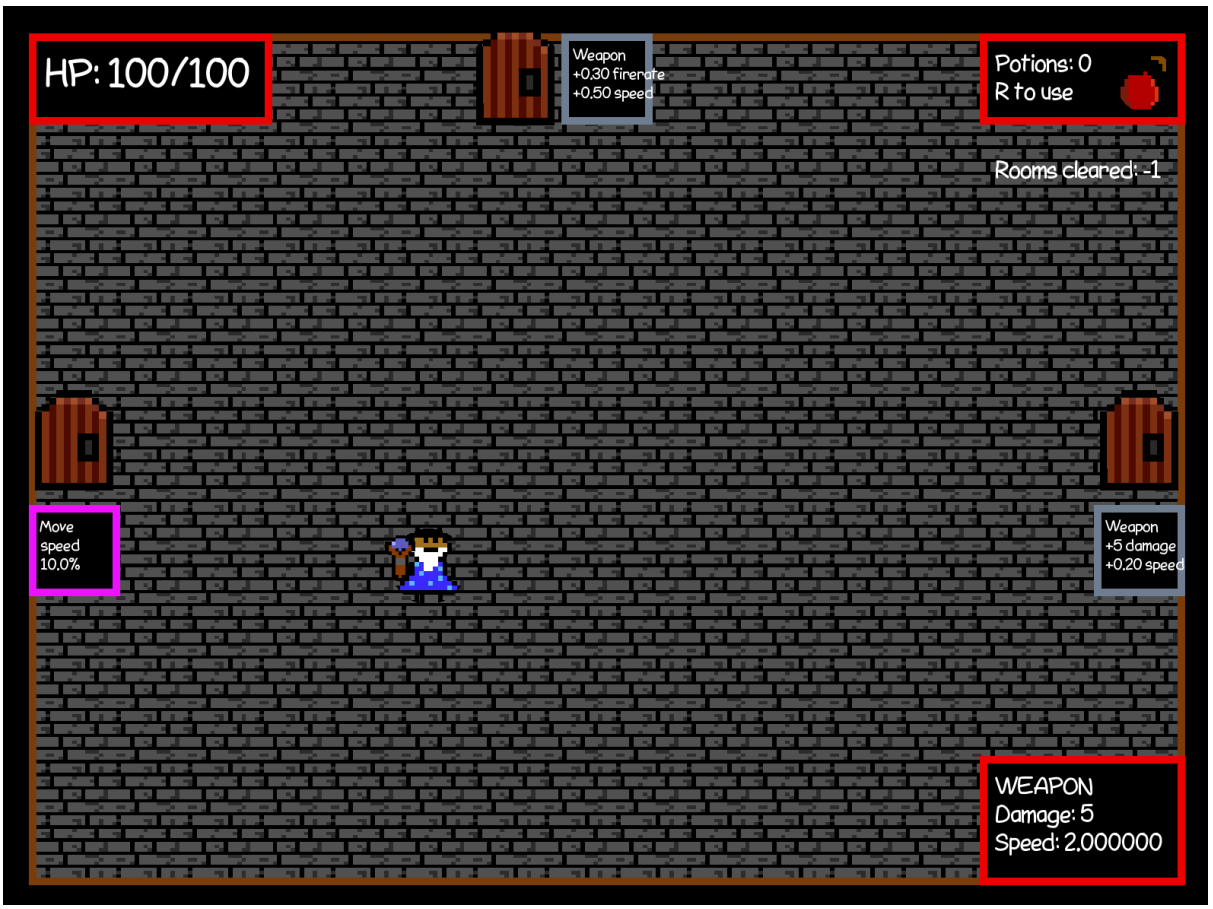
Overview

Our project is a simple top-down roguelike dungeon crawler game, where the player clears rooms of enemies and eventually wins the game by clearing enough rooms and killing the final boss.

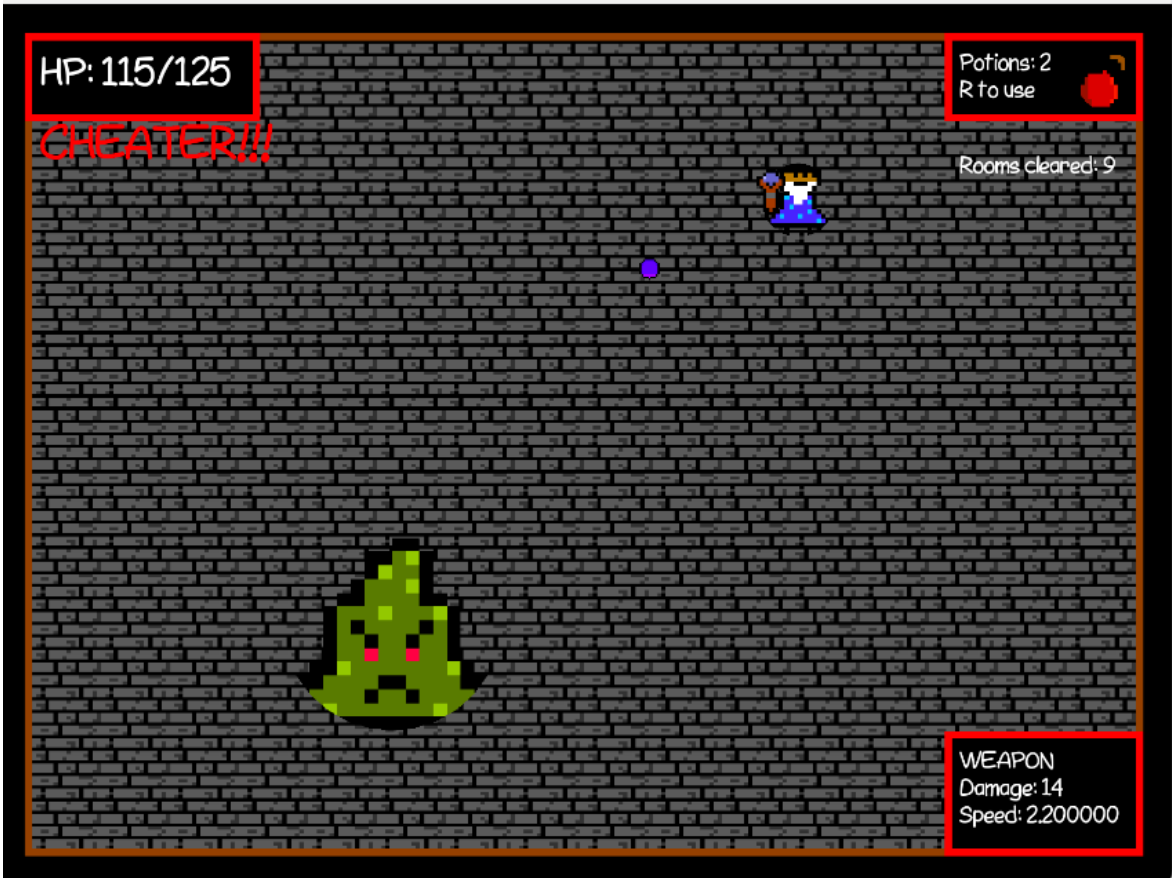
The game is themed in a traditional fantasy style. After clearing the room, the player can choose from different doors to go to. Different doors have different prizes such as weapon upgrades, health potions and a speed boost.

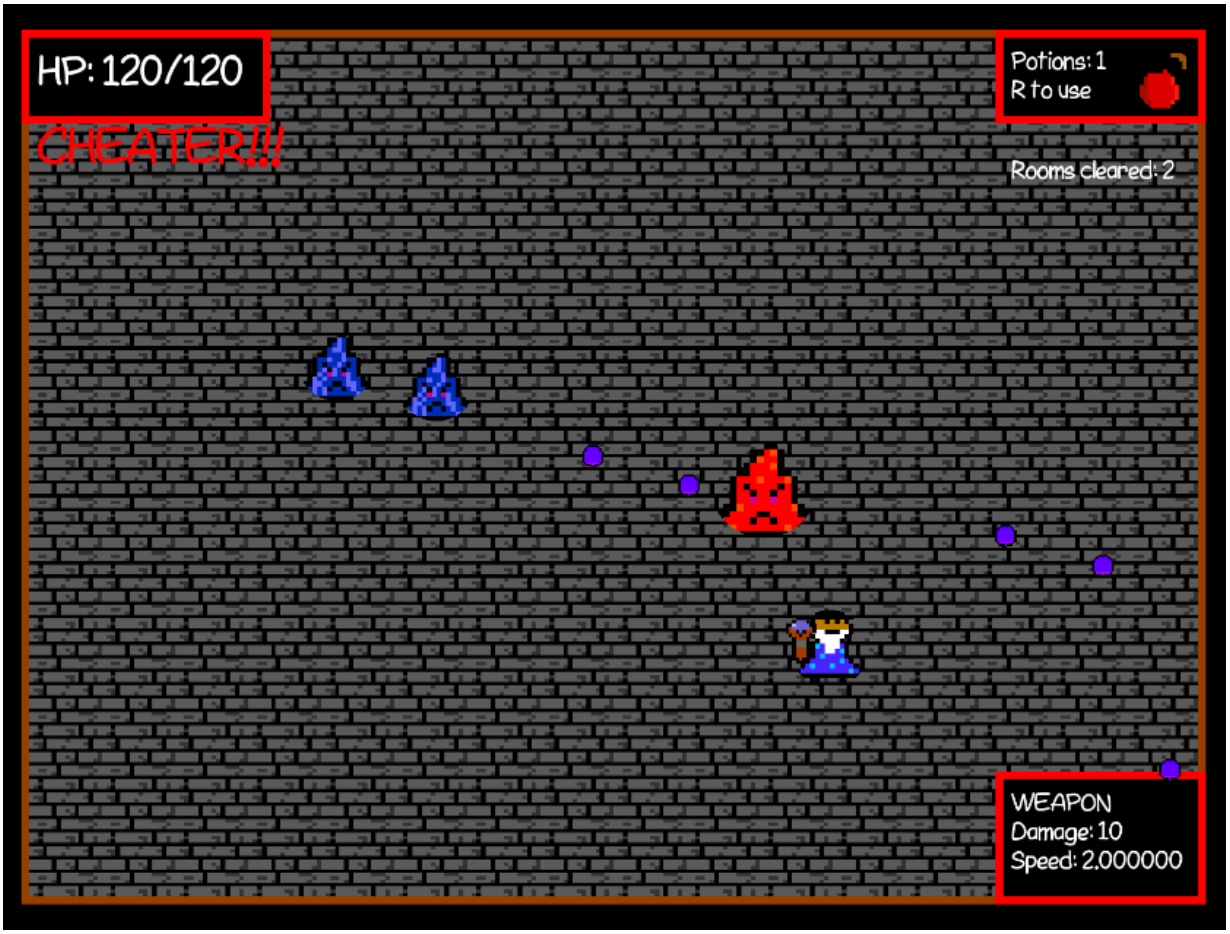
The game is realtime, and the player and monsters use magical based ranged weapons.

Screenshots:

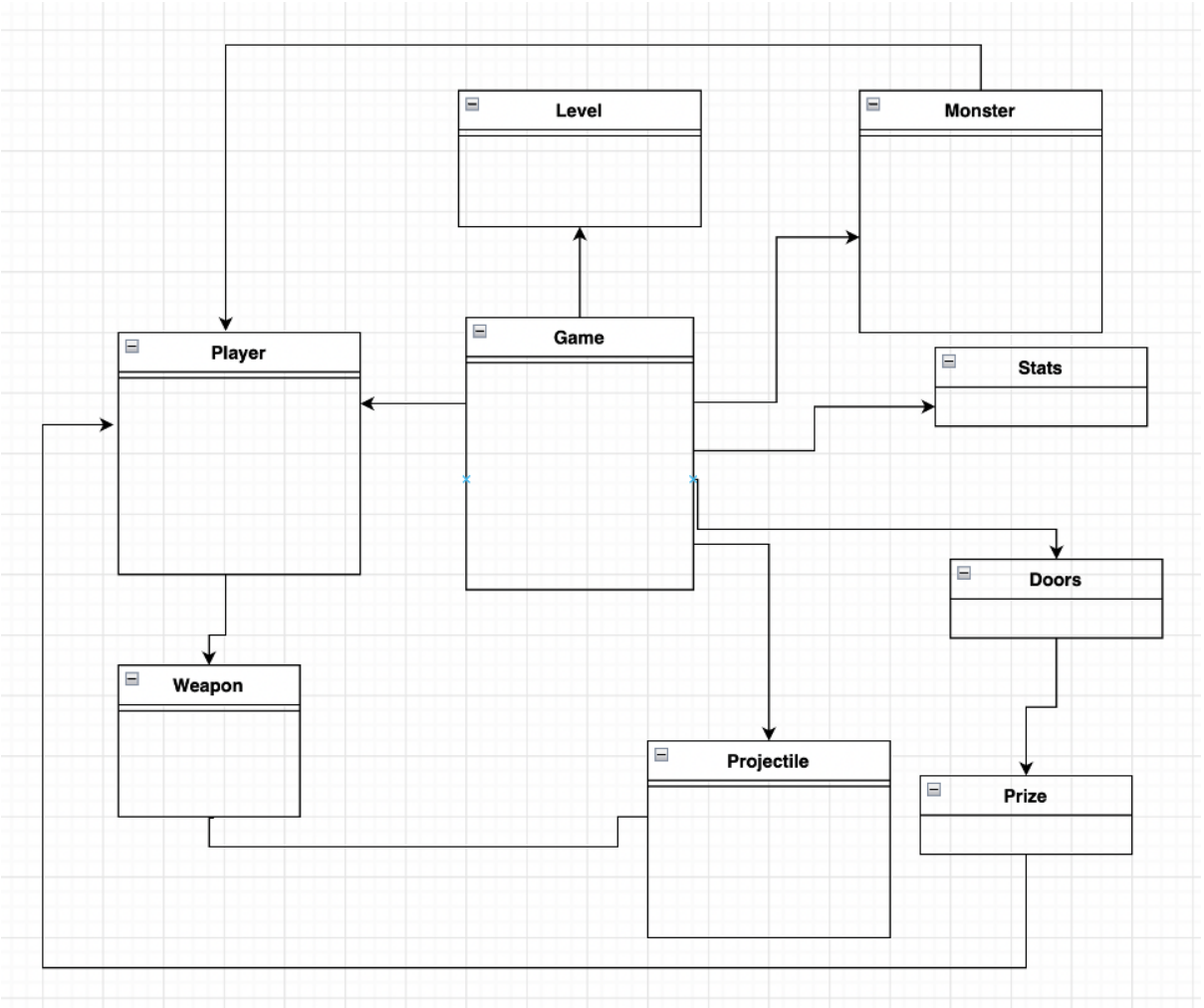

HP: 100/100
Weapon
+0,30 firerate
+0,50 speed
Potions: 0
R to use
Rooms cleared: -1
Move
speed
10.0%
Weapon
+5 damage
+0.20 speed
WEAPON
Damage: 5
Speed: 2.000000


HP: -5/130
Potions: 0
R to use
Rooms cleared: 5
You perished :(
Press R to restart
WEAPON
Damage: 11
Speed: 2.500000


HP: 115/125
CHEATER!!!
Potions: 2
R to use
Rooms cleared: 9
WEAPON
Damage: 14
Speed: 2.200000


HP: 120/120
CHEATER!!!
Potions: 1
R to use
Rooms cleared: 2
WEAPON
Damage: 10
Speed: 2.000000

Software structure



The derived classes of the Prize class (ExtraHP, Speedboost, WeaponUpgrade and Healthpotion) that represent different types of prizes are omitted from the UML graph for clarity.

Game:

Creates the game, updates game state and renders it while game is running

Level:

Creates monsters, obstacles and borders of the room and draws them into same window.

Player:

Character that can move and attack by user commands.

Monster:

Monsters try to damage the player. All the monsters need to be killed to win the game.

Weapon:

Class for weapons that the player and monsters use. Launches a projectile that causes the damage on hit.

Projectile:

Class for projectile that the weapon shoots. Weapon class determines damage of projectiles.

Prize:

Class for prizes that player gets from doors.

Doors:

Class for doors that appear to the room when all the monsters are killed. Creates and handels the prizes as well as the needed graphics for the doors.

Stats:

Class that draws the stat boxes

Instructions for building and using the software

The Linux build is very simple (especially on Aalto computers) and the recommended method. Mac is fine as well (at least on ARM). WSL is not ideal, and building on Windows natively is not supported.

Build instructions:

This is assuming everything (WSL,C++ compiler, CMake, etc.) is installed on your computer.

Two disclaimers regarding WSL:

- *The program will run very slowly on WSL due to the lack of GPU acceleration*
- *You need to install the WSL Gui support using the following guide*
    - *Due to this will only work on Windows 11*
    - *https://ubuntu.com/tutorials/install-ubuntu-on-wsl2-on-windows-11-with-gui-support#5-install-and-use-a-gui-package*

1. Make sure SFML 2.5.1 (Simple Fast Media Library) is installed on your computer. On Aalto computers this is automatically installed
    a. If it is not installed use the command **sudo apt-get install libsfml-dev** *(Linux/WSL (Debian))*
    b. On Mac install it using homebrew (https://brew.sh/) with **brew install sfml**
2. Clone the project using **git clone git@version.aalto.fi:basicj1/dungeon-crawler-1.git**
3. Go inside the cloned folder using **cd dungeon-crawler1**
4. Configure CMake with **cmake .**
    a. This part might fail if something is not right with your install
5. Build with **make**
6. Open the successfully compiled program with **./DungeonCrawler1** (Linux/WSL)**,** or **open DungeonCrawler1** (Mac)
7. Done!

Controls:

The player is controlled using the W,A,S and D keys, while using the Left Mouse Button makes the player shoot in the direction of the cursor. Once all the enemies have been vanquished, three doors will appear, which will give the player a buff and spawn them in a new room with more enemies. The player can restart the game at any point by pressing the P button. Health potions are used by pressing the R button. Doors can be used simply by running into them.

PS. (Pressing Enter kills all the enemies in the room you are currently in. This is a cheat meant to help with testing, but if you find the game too difficult, this can help with seeing the game to its conclusion)

Testing

We tested the program simply by running it and seeing how it functions. This was done due to the heavy user focused design of the software, thus automated testing would not have been justified. Additionally something that could be called an unintentional bug might occasionally be a desirable feature when talking about interactive games.

Work log

Tiia Aaltonen

Responsibilities: Doors and prizes placed on them, functions and classes needed for them. Figuring out a way to develop the game on Windows 10 OS and helping other members with it. The base structure of the game class. Stats class and needed functions in Player for displaying statistics. General helping/debugging when needed.

Hours:

- **Week 45:** 5h
    - First meeting, plans and getting to know the libraries
- **Week 46:** 5h
    - Basic structure of game class and player movement, struggling with SFML
- **Week 47:** 16h
    - Getting the game to run on Windows 10, Stat class and needed functions, started Prize class
- **Week 48:** 9h
    - Doors, possibility to pick up prizes and use healthpotions,
- **Week 49:** 14h
    - Prize subclasses and prize randomization, needed functions in Player and weapon for using the prizes

Mikael Gustafsson

Responsibilities: Level generation, including enemy spawning and the retooling of the enemy class related to this, time-step code, initial player movement, graphics (sprites) and their rendering, mouse aiming, player death and game restarting, player cheats for easier testing, additional UI features, building and testing on Mac and Linux, configuring CMake for cross-platform building. Additionally project management, meeting notes, and help with QA.

Hours:

- **Week 45:** 3h
    - First meeting, and some planning related to game concept and general functionality
- **Week 46:** 5h
    - Compilation on Mac, collaboration with player movement, and initial fixed time step, configuring Git, meeting
- **Week 47:** 7h
    - Final time step fixed, starting on level class and simple collision detection for player, changes to print functions
- **Week 48:** 22h
    - Randomly generated enemies, shooting in mouse direction, enemy dying, changes to projectiles, cheat mode, restarting of game, sprites, rendering, CMake, testing on Linux,

- **Week 49:** 13h
    - Enemy types added, bugs with ui fixed, more CMake, filesystem changes and cleanup of code, help with memory issues and testing, final merging of branches and title screen

Pyry Satama

Responsibilities: Projectile, player, monster, weapon classes. Some modifications to game class.

Hours:

- **Week 45:** 5h

  First meeting, planning

- **Week 46:** 18h

  compilation, meeting, modified player and monster classes (drawing monster on screen, movement, etc)

- **Week 47:** 18h

  player, monster, projectile classes, meeting.

- **Week 48:** 0h

  in the army

- **Week 49:** 6h

  meeting, weapon class modifications, project documentation

Jakov Basic

Responsibilities: Projectile, player, monster, weapon classes. Moving from one room to another. General ideas for the project.

Hours:

- **Week 45:** 5h

  first meeting, planning

- **Week 46:** 10h

  compilation, meeting, initialized player and monster classes

- **Week 47:** 10h

  player, monster, projectile classes, meeting.

- **Week 48:** 12h

  modified classes player, monster, projectile, weapon

- **Week 49:** 12h

  movement from one room to another. Modified previously mentioned classes