Dungeon Crawler

team 4

Project plan

Joel Sågfors (project manager)

Johan Rabb

Jon Galán

Antti Matikainen

General description

We hope to make the game include the following features:

- We are planning to have the game run in real time, rather than it being turn based.
- The graphics will be simple, rendered using boxes filled with a solid color or even sprites.
- Allows attacking an object, if you are in close proximity, and there are no obstacles blocking the path.
- Collision detection will make sure characters can't pass through walls.
- A map generation algorithm that places pre-designed rooms randomly, and makes paths between the rooms.
- Algorithms for the AI, for pathfinding and attacking.
- Player inventory and a menu for accessing the inventory.
- Player can pick up items lying on the ground.
- Weapon switching.
- Keyboard-based user interface, possible also mouse support.
- Object oriented style for the characters:
    - A general character class.
    - Virtual classes for character types, such as wizard, fighter, and monster with different attributes.
    - Virtual classes for enemy types. If we have time, the enemy classes can be customized in a human-readable text file.
- Customizable controls, either configured from a configuration file or in the game menu.
- Aiming either with mouse or with keyboard. Input detection will be handled by the SFML library.

Architecture

- The equipped weapon will be stored in a variable.
- Player inventory is stored in a vector.
- The map consists of a 2-dimensional array pointing to tile-classes.
- Items are described in a human readable text file, which makes it easy to customize the game. Possibly in JSON format.
- Rooms are designed with ASCII in a text file.
- Character positioning implemented using SFML.
- Separate loops for the main menu and game itself.
- Main game algorithm:
  1. Get user input
  2. Parse input and determine the action to take
  3. Call a function related to the chosen action
  4. Call function for non-player character’s actions
  5. Draw the screen
  6. Return to 1.
- Path-finding could use for example Dijkstra's algorithm, until the character is only a few tiles from the player, at which point they will move straight towards the player.
- Hit detection of an attack: The attack have a direction and every character have a hit radius or hitbox. The attack propagates forward and if it at some time is inside the hitbox of a character, the character takes damage.
- Game speed will depend on the framerate, which means the framerate needs to be limited to a reasonable number to keep it constant, using built in SFML function.

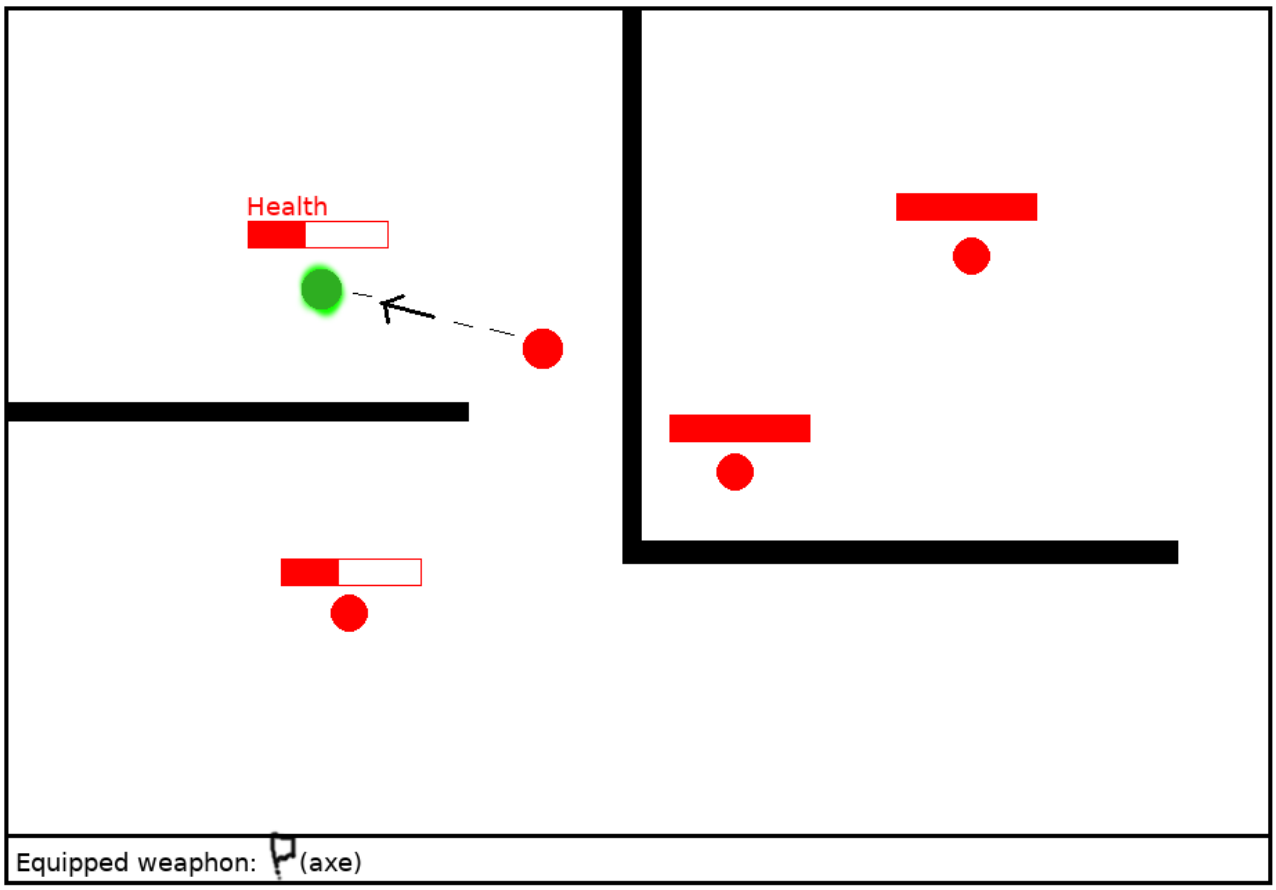

Figure 1. Mockup of user interface. All characters have a health bar. The player is indicated with a green circle. The enemy closest to the player has shot an arrow towards the player, and the current position of the arrow is shown.

Preliminary schedule

In the first and the beginning of the second week we will do basic testing of the SFML library to explore the possibilities of the library. This includes simple tests like making a circle moving towards the coordinates of the mouse cursor.

During week two, we hope to begin working on the basic classes and functions, as well as on a simple map.

During week three we'll focus on creating the algorithms for path finding, collision detection, and hit detection of attacks. We'd also have to implement functions to use various attributes of the classes, such as inventory related functions. In the end we add more features from the list above, such as sprite-based graphics.

During the development we will probably use simple draft versions of maps and chsaracter classes. If we make the properties for these easily customizable, it is easy to add different character classes and more advanced maps, such as special abilities for a character class.