Requirements Snake

Group 5

CSE2115 Software Engineering Methods

Contents

1. Functional Requirements 3
    - 1.1 Must Haves 3
    - 1.2 Should Haves 4
    - 1.3 Could Haves 5
    - 1.4 Wont Haves 6
2. Non-functional requirements 7

1 FUNCTIONAL REQUIREMENTS

In this section the requirements for the game are grouped together in clusters, each on an individual page. This is done following the MoSCoW model for prioritizing these requirements.

1.1 MUST HAVES

- Game must have a snake object that can be moved around via the WASD keys on the keyboard.
- A (Postgres) SQL database with code for the connection between it and the game.
- Authenticate using an username and a password, via database and login screen. Done with hashing and adding a random salt.
- The score of each play should be recorded in the database with a nickname (only nicknames are stored for users).
- Game must have a board by at least 10x10
- The snake should grow bigger when it ate an item. This increases the score with 10 points.
- Food is spawned randomly in the game.
- Game starts at the login screen. After that you come in a screen where you can choose to can play levels, free mode (no levels), change settings and log out.
- The Game has a simple display to keep track of a current score of a play.
- The snake must know when it hit a wall/hit, if this happens you lose the game and go back to level 1 in level mode. In free play mode (no level) you start over.
- Game should be pauseable.
- At the end of each play, the user should be able to enter his/her name together with the recorded score
- At the end of each play, the game should show the top 5 scores that have ever been recorded

- At the end of each play, the user should be able to enter his/her name together with the recorded score

1.2 SHOULD HAVES

- User should be able to click Play Again after dying
- A menu list you can navigate through and change settings. The include
- Explanation on how to play, accessed with a button on the home screen. Here they will see a single screen with text and in-game images on how to play the game.
- A user should be able to delete their account.
- A home screen with leader board/play button.
- Hashed passwords in the database
- A start screen showing the *Snake* game logo.
- Always show the username in the top right of the screen after login.

1.3 COULD HAVES

- Different difficulty levels in terms of snake speed. Speeds are measured in game-tiles travelled per second. There are four modes: 2, 3, 4 and 5 tiles per second.
- Different levels have different appearance regarding walls and ground
- Soundtrack and sound effects licensed under Creative Commons.
- A user should start with more than one life
- A user should be able to reset his password
- A user can collect powerups/extra lives. Power ups can temporarily slow your snake down (timer shown when power up expires), make your snake smaller or give extra points when picked up.
- Different skins for your movable snake (only provided by the game).
- Users can provide custom level maps in txt format made with instructions on how to make them.
- User can win snake skins if your score in free mode (not the level mode) is above certain thresholds.
- Portals where the snake can go through to teleport to other places of the level.
- A *Timed Run* game mode where you need to collect a certain amount of pellets as fast as possible. In this mode the times of each play is recorded in the database.
- Username and password are remembered when starting the game.

1.4 WON’T HAVES

- 3D version of the game
- Make the game multiplayer (client and server)
- Have customizable controls
- Changeable themes for the menu screens
- Snake that can choose to move forward from its tail in game.
- Movable obstacles other that the snake itself.
- Snake can move diagonally.
- Let the game behave like a side-scroller, i.e. the snake stays in the centre and the world moves around the snake.

2 NON-FUNCTIONAL REQUIREMENTS

In this section the requirements the non-functional requirements are listed. These involve constraints and functions offered by the system, like development and process constraints, timing constraints and constraints imposed by common standards.

- Use SQL and JDBC driver
- The game should run on Windows, MacOS and Linux systems having a JRE installed.
- Use prepared statements in Java
- Game made with the libGDX library.
- Have a branch coverage of at least 70%
- Development of the product should follow the Test Driven Development paradigm
- Scrum is used as a project management framework