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.

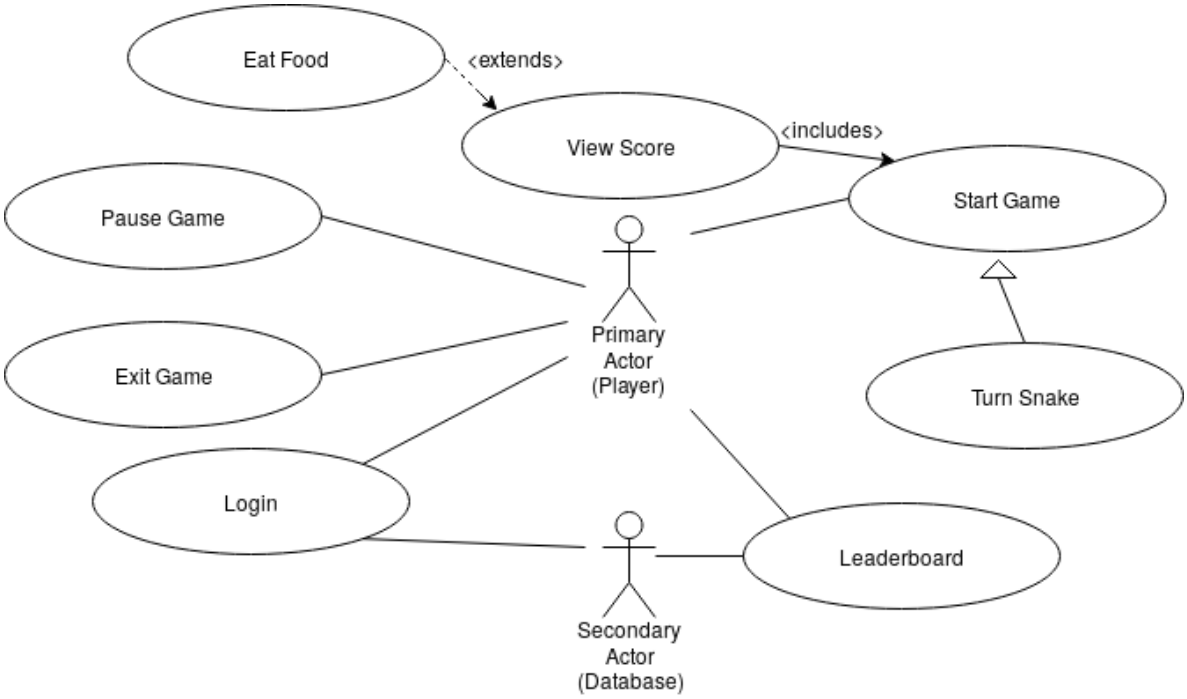
2 NON-FUNCTIONAL REQUIREMENTS

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

2019/11/26

Use cases for our project



Use case #1

- **Use case**: Start Game
- **Purpose**: To initialize a new game.
- **Overview**: The player presses the start game button in the main menu after logging in. The system then loads a new board and opens a game screen after few seconds the main game loop starts and the snake starts moving forward. *WASD* controls are enabled for player to control the snake. During gameplay food objects spawn around the board. If the board initialization is unsuccessful the player is returned to the main screen and an error message is displayed.
- **Actors**: The player.
- **Pre-Conditions**: A user is authenticated by database and logged in.
- **Post-Conditions**: After the game ends the score of the player is saved to the database and a new game can be played.

Use case #2

- **Use case**: Pause Game
- **Purpose**: To save the current state of the game during gameplay and resume it afterwards if necessary.
- **Overview**: While a player is in the middle of the game there is an option to pause the current state of the game by pressing the pause corner in the top corner of the screen. When the button is pressed the main loop of the game is stopped and the positions of the snake, food, power ups, and any other objects on the board are saved. Upon pressing the pause button again all variables are restored to their initial state before pausing and the main loop is started again, thus the game resumes.
- **Actors**: The player.

- **Pre-Conditions**: A game must be initialized in order to be able to pause a current session.
- **Post-Conditions**: The state of the game after unpausing must identical to the one before.

Use case #3

- **Use case**: View Score
- **Purpose**: For a player to have the ability of tracking how well he is doing represented by a numerical value during the runtime of a game
- **Overview**: While playing a game in a corner of the screen current score of a session is displayed. The score increments everytime a food object is consumed, indicated by the extensive "Eat Food" use case.
- **Actors**: The player.
- **Pre-Conditions**: A running session of a game must be instantiated and the initial value of the score must be set to zero.
- **Post-Conditions**: The value of the score after the game has ended must be saved to the database.

Use case #4

- **Use case**: Login
- **Purpose**: Authenticate the user of the system to gain access to his account where personal achievements are stored.
- **Overview**: Upon launching the system a user is prompted with a log in screen in which he must enter his credentials or create a new account if not done already. After entering his information a request to the database is made to retrieve that particular users information. If the credentials match up the user is granted access. In the case where a user cannot be found or credentials are incorrect a text is displayed that such user does not exist or the information provided is incorrect.
- **Actors**: The player as the primary actor and a database as the secondary actor.
- **Pre-Conditions**: None.
- **Post-Conditions**: The main menu screen is shown for the user with the ability to play a new game.

Use case #5

- **Use case**: Exit game
- **Purpose**: To close the game after the player has pressed the close button.
- **Overview**: When the player is done playing he can press the exit button which first prompts it a new window asking him to confirm his decision. If the player agrees he is logged of from the system and the system window is closed. If he disagrees the smaller prompt closes and the player is once again able to select any option from the main menu.
- **Actors**: The player.
- **Pre-Conditions**: The player must be logged in and in the main menu screen.
- **Post-Conditions**: The system is shutdown and the player cannot see his progress anymore or play the game.

Use case #6

- **Use case**: Turn snake
- **Purpose**: The ability to move snake game object during an active session of a game.
- **Overview**: When a user is in an active session of a game and the main loop of the game is running, meaning that the snake is moving forward, the player, by pressing buttons *WASD* can control the snake. If the snake is moving from right to left only buttons W (up) and S (down) can be used, because a snake does not have the ability to immediately start moving

in the opposite direction, button A and pressing a button that is the same as the current direction gives no change. If the snake is moving from left to right, only W and S can only be pressed also. If the snake is moving from the top to bottom only A (left) and D (right) buttons can be pressed, same goes if the snake is going from bottom up. Basically, the snake only turn 90 degrees at a time.
- **Actors**: The player.
- **Pre-Conditions**: A game session must be started.
- **Post-Conditions**: The direction of the snake must be changed accordingly to the direction that was pressed.

Use case #7

- **Use case**: Leaderboard
- **Purpose**: A user can check the overall leaderboards for his own and other players highscores.
- **Overview**: After authenticating to the game, in the main menu there exists a button Leaderboards. After pressing the button a request is made to the database to retrieve the top 10 highest scores of all players and the top score of the player that is logged in. The system then displays a list of best achievements with the users highest score.
- **Actors**: The player, the database
- **Pre-Conditions**: A user must be logged in to the application.
- **Post-Conditions**: The leaderboard must be displayed and the ability to go back to main menu is enabled.