Software Engineering Methods

Assignment 1

Snake-Group-1

Table of Contents

Exercise 1.1) Requirement Engineering - using MoSCoW method

i) Must Have

- Authentication phase: players must log-in to the game before being able to play with it (via a database; encrypted).
- The player must be able to start a new game.
- The player must be able to stop their current game.
- The game must initialize the player's score at 0.
- The player must be able to move a snake either to the left, right, up or down at a time.
- The game must not allow the snake to be moved outside of the board.
- The game must stop when the snake hits itself or the borders of the board.
- The game must keep track of the player's score.
- The player must be able to gain points by eating a pellet.
- The game must place pellets on random places on the board.
- The leaderboard must show the top 5 scores in the database.
- The score of each game must be recorded in a database.
- At the end of each play, the user must be able to enter his/her name together with the recorded score.

ii) Should Have

- The game should have multiple levels of varying difficulty (via obstacles, amount of points needed, etc.)
- The game should show the points needed to clear the level.The game should show a visible grid on the board.
- The game should have a snake that gets longer when it gains points.
- The game should show the player's game statistics after losing a game.
- The player should be able to play another game without restarting the application.

iii) Could Have

- The game could play music when the player is playing a game.
- The player could be able to pause the game while in progress.
- The game could play a sound when the snake gains points.
- The game could play a sound when the snake dies.
- The player could be able to choose what happens when colliding with a wall (whether the snake warps or dies) via a setting.
- The game could award bonus points for pellets that are more difficult to get.
- The player could choose different themes that change the looks of the game.
- The snake could move freely.
- The game could be playable with multiple players.
- The game could have a friend system, enabling players to befriend each other and see the scores of other players.
- The player could earn badges and unlock new themes when it achieves certain scores.
- The game could have a loading bar when the player needs to wait.

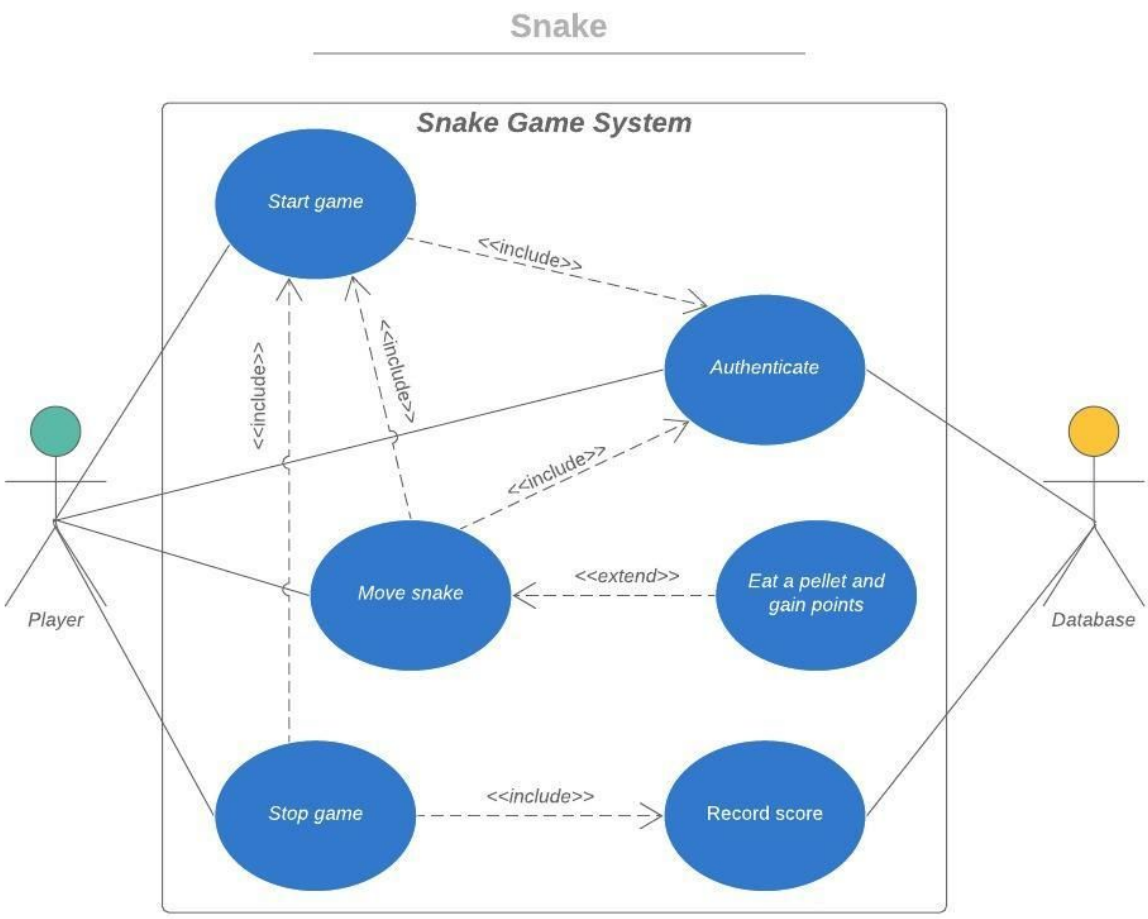
iv) Won't Have

- The player won't be able to modify the background of a theme by uploading a picture from their computer.
- The game won't offer the option of a 3D version of the game.
- The game won't have controller support.
- The game won't have realistic graphics.

Exercise 1.2) Non-Functional Requirements

- Use SQL and JDBC driver
- Use prepared statements in Java to avoid code-injection vulnerabilities
- The game must be playable on Windows (7 or higher), Mac OS X (10.8 and higher), and Linux.
- The game must be implemented in Java (Jdk 12)
- The game UI will be designed using JavaFX
- The implementation of the game will have at least 80% line test coverage
- The implementation of the game will have at least 70% branch coverage
- The Scrum methodology will be applied to the development process.

Exercise 2.1) Use Cases Diagram



Exercise 2.2) Modelling Use Cases

Record score

- Title or Reference Name
    - Record Score
- Author
    - Jeongwoo Park
- Date
    - 26/11/2019
- Purpose
    - Record the score that a player got into database after the game ends so that we can later on check the highscore.
- Overview
    - The authenticated player first plays a game. After he/she completes a game, he/she gets the score according to the number of pellets that the snake ate. Then, the score is stored in the database.
- Cross References
    - Stop game
- Actors
    - Database, Player
- Pre Conditions
    - The player is authenticated. The connection with database is set up. The player finished playing a game.
- Post Conditions
    - The score is stored in the database.
    - The player goes back to the entry page.
- Normal flow of events

Player actions	**System actions**	**Database actions**
1. Player completes a game.	3. The system shows a score.	2. The score is stored in the database.

- Alternative flow of events
    - The player closes the window right after the game. If he/she closes it before the data is stored, then the data is not saved. If he/she closes it after the data is stored, then the data is successfully stored.
- Exceptional flow of events
    - The database connection crashes due to system failure before the score is stored.

Move snake

- Title or Reference Name
    - Move snake
- Author
    - Remco den Heijer
- Date
    - 26/11/2019
- Purpose
    - Player is able to move a snake on the board.
- Overview
    - Player pushes arrow button left, right, up or down to move a snake on the board in a direction. Player can only move the snake once the game is started.
- Cross References
    - Eat a pellet and gain points, Authenticate, Start game
- Actors
    - Player
- Pre Conditions
    - Game is started. Snake is within the board. Snake is shown to the player.
- Post Conditions
    - Snake is within the board.
- Normal flow of events

Player actions	System actions
1. Player pushes an arrow button.	2. Check if coordinate the player wants to move to is within the board.
	3. If it is, move the snake to new coordinate.
	4.If player hits a wall, the game is stopped. This is equivalent to coordinates outside the board.

- Alternative flow of events
    - Step 4: Wrap the coordinate around so snake comes out of the opposite border.
- Exceptional flow of events
    - Step 1: Player is not touching any button.
    - Step 2: Snakes moves in current direction every game tick.

Gain points by eating pellet

- Title or Reference Name
    - Gain points by eating pellet
- Author
    - Rohan Deshamudre
- Date
    - 26/11/2019
- Purpose
    - Used as a quantitative indicator of success in the game. A goal is often made of attaining a better score than your previous scores or for multiplayer game, attaining a better score than the opponents in order to win.
- Overview
    - Random pellets appear on the board and the snake to moves towards the pellet and when they collide, the snake 'eats' the pellet. When a snake 'eats' a pellet, it gains points that add to the score and the aim is to eat as many pellets as possible.
- Cross References
    - Move snake
- Actors
    - Player
- Pre Conditions
    - The snake is able to move
    - There is a pellet on the board
- Post Conditions
    - The points gained by eating the pellet is added to the score
    - The pellet is no longer on the board
- Normal Flow of Events

Player actions	System actions
1. Player moves towards the pellet and eats it.	2. Pellet disappears from the screen and a new one appears on a random spot on the board.
	3. Points gained by eating the pellet is added to the score.

- Alternative Flow of Events
    - Step 3: The snake hits a wall or itself and dies.
    - Step 3: Player moves away from the pellet and does not eat it.
- Exceptional Flow of Events
    - Step 1: The game crashes due to an unforeseen bug/error in the system.
    - Step 2: There is no pellet on the screen.
    - Step 3: The points gained is not added to the score.

Stop game

- Title or Reference Name
    - Stop game
- Author
    - Remco den Heijer
- Date
    - 26/11/2019
- Purpose
    - Player should be able to stop a game.
- Overview
    - Player clicks on a button which stops the game. Stopping the game means that the player cannot move his snake anymore (the game ended) and the score is recorded.
- Cross References
    - Record score, Start game
- Actors
    - Player
- Pre Conditions
    - Game is started. User is able to move the snake.
- Post Conditions
    - Game ended. User is not able to move the snake.
- Normal flow of events

Player actions	**System actions**
1. Player clicks stop button.	2. Game is ended. Player cannot move the snake anymore.
	3. Score is shown to player. The score is recorded..

- Alternative flow of events
    - Step 2: Game is not stopped, but the game window is closed.
- Exceptional flow of events
    - Step 2: Game is not stopped, but the system crashes.
    - Step 3: Player is not able to record his score.

Start game

- Title or Reference Name
    - Start game
- Author
    - Leon de Klerk
- Date
    - 26/11/2019
- Purpose
    - Enable the player to play the game and start all processes, that make up Snake.
- Overview
    - The player is provided with a button to start playing the game. From this point on the actual game will start and the player is able to play Snake. This will start rendering the board, the snake and all other game elements on the screen. It also initializes the current score of the player to zero.
- Cross References
    - Authenticate
- Actors
    - Player
- Pre Conditions
    - The game program is started.
    - The user is authenticated.
    - The game is not started (either no game was started or the game just ended).
- Post Conditions
    - All game elements are loaded and rendered on the screen.
    - The game is started and the player can play.
- Normal flow of events

Player actions	**System actions**
1. Player presses the start button.	2. The system initializes the score to 0.
	3. The scene is loaded and rendered.
	4. The game starts.

- Exceptional flow of events
    - The only exceptional case is when there is an unforeseen bug or system/pc failure that results in the whole program crashing.

Authenticate

- Title or Reference Name
    - Authenticate
- Author
    - Andrei
- Date
    - 26/11/2019
- Purpose
    - Give the user access to the game and his/her previous game sessions.
- Overview
    - The user is prompted to enter their username and password. The system validates the username and password via a connection to the database. If validation is successful, the player can access the game and all its functionalities; otherwise (unsuccessful authentication) an error message is shown and the user is prompted to try again.
- Cross references
    - Start game, Move snake
- Actors
    - Database, Player
- Pre Conditions:
    - The program must be started and the player should be on the login screen.
- Post Conditions:
    - The player is logged into the game and can access their data and play the game.
- Normal flow of events

Player actions	**System actions**	**Database actions**
1. Player starts the program.	2. System loads the login screen.	
3. Player enters username and password.	4. System connects to the database.	5. Validates the username and password.
	6. The system displays the main menu.	

- Alternative flow of events
    - Step 5: the username or password entered are incorrect.
- Exceptional flow of events
    - Step 4: the system cannot connect to the database, an error message is displayed.
    - An unforeseen bug or system/pc failure that results in the whole program crashing.