Sana Hameed

Sh223nw@student.lnu.se

**Tobias Ohlsson**

Assignment 2

**Modeling-Software Design**

**Time Log**

**Hangman Game**

|  |  |  |
| --- | --- | --- |
| **Task** | **Estimated** | **Actual** |
| **Creating time log** | **30m** | **15m** |
| **Use case** | **1h** | **45m** |
| **State machine** | **1h** | **40m** |
| **Activity diagram** | **50m** | **38m** |
| **Class diagram** | **30m** | **20m** |

**Use Case Diagram**

**Hangman Game**

**Fully Dressed Use Case**

**Hangman game**

1. **Use case: New Game**

Primary Actors: Any one playing the game

Goal: to start a new game

**Precondition:**

1. System support the game configuration
2. The file has been run and game screen appeared.

**Scenario:**

1. Go to the new game button and click on it.
2. New game is loaded on system.
3. **Alternative Scenario:**

Press start new game button and play a game.

1. **Use case: Play a single word game**

**Precondition:**

1. Game was played before.

2. Game support to guess a single word.

**Main Scenario:**

1. Play game and try to guess word.
2. You can try for multiple time if you guess wrong

**Alternative Scenario:**

Guess word if you enter wrong word you can enter again a right word

1. **Use case: Select Level**

**Precondition:**

1. Required level has been unblocked.

2. Game supports loading level.

**Scenario:**

1. Go to the new game and click to the select level.

2. Select level and load the select level.

**Alternative Scenario:**

Click on select level and directly select the level.

**4) Use case: Exit Game**

**Precondition:**

A game level is being played.

**Scenario:**

Click on exit game.

**State Machine Diagram**

**Hangman Game**

Main menu

Splash screen

Enter

Point

Play game

Open game

Play Level

Select Level menu

Win game if guess word is right

Guess words

Restart game if word is wrong

Play next level of the game

Exit game if user want

**Implementation:**

This is a game made to simulate the classical game of Hangman; in which there will be a 'questioner' who will present the player with a word whose letters are blanked out, and the player will then have to guess what the word is correctly. A wrong guess would result in the player's lose the game, a stick figure, getting closer and closer to be 'hanged'. Hence, the Hangman name of the game.

In this version, the game takes on the role of the 'questioner', with the player allowed to choose the difficulty he (or she) wishes. The game assumes that longer words are more difficult, thus the higher the difficulty, the longer the word given towards the player. The rest of the game just follows the pen-and-paper version.

Unlike the pen-and-paper version however, this Hangman game lacks the ability to come up with new words, and thus is somewhat limited in the sense that the words 'questioned' are all hard-coded into the app itself.

The original draw of this idea was to somehow implement the Hangman game with the ability to generate words on its own; we eventually found out that such an algorithm would probably be quite costly and difficult to implement. However, this being the first game we develop, we hope that our subsequent ones would rely less on hard-coded data.

**Class Diagram**

**Hangman Game**

**Main Class**

**Play Hangman**

**Class**

**Methods**

Startpoint()

Guesswords()

Find empty position()

Wordlength()

Playhangman()

Result()

Terminatehangman()

**Attributes**

Found guessed words

Guessed words

Missing words

Main function

Objects of classes

Play game

End game

**Game Result**

WIN

LOSE

**Handed in:**

https://github.com/Sanahameed/sh223nw\_1dv600