Sabawoon Popal & Aung Khant Myo

CSC 450

Professor Chays

March 28, 2024

Design of Network Application

**OBJECTIVE: A GAME OF HANGMAN FOR ENTERTAINMENT PURPOSES AND LEARNING NEW WORDS.**

**INFORMATION STORED ON EACH HOST:**

**Server:**

* **Simple version:**
  + **Letters that the client has used to guess the word.**
  + **Name of client**
  + **Number of attempts left (Starting with 6, representing each limb).**
  + **Record (Wins / Games played)**
  + **An array of words containing playable words to start the game.**
* **Enhanced Version:**
  + **Leaderboard showing the wins each client has**
    - **With the server already having the clients name in the simple version, we can use that to label who has how many wins.**
  + **Words are split up into categories. So each category has their own array of words.**

**Client:**

* **The letters that are a part of the word if the client guessed them (IN THEIR PROPER ORDER).**
* **Attempts remaining to keep the client aware.**

**INFORMATION NEEDING TO BE COMMUNICATED:**

* **The letters that the client guesses. Client will validate and send a letter to the server.**
  + **Server will reply back whether the letter is in the word.**
* **Leaderboard stored in server to be shown to the client after an instance of the hangman game ends.**

**POTENTIAL ISSUES:**

* **Synchronization: Updating the leaderboard after each win from the various clients.**
  + **In addition, the potential of more than one client winning at the same time.**
* **Layout of the game**
  + **How to display the game of hangman? Use GUI to draw the stickman? Or just provide underscores in replacement for letters that have not been guessed?**

**Algorithm Diagram for Hangman Game**

**Server Client**

| 1. **Listen on a specific port for a connection request.** | 1. **Initiate a connection request, establishing the name of client trying to connect, and the server’s IP address and port. (String, String (localhost), int)** |
| --- | --- |
| 1. **Accept connection request** |  |
| 1. **Send introductory hangman message to client.** 2. **Simple version: Welcome message, the game immediately starts. (Option: Separate the word to guess as an array for better comparison.)** 3. **Enhanced version: Welcome message, server sends categories to user to pick from. ex. (“sports** | **5. Receive introductory hangman message from server.**   1. **Simple: Receive the array object of the word to guess. Hints will be available since category selection is not existent.** 2. **Enhanced: Hints aren't allowed since client picks the category.** |
|  | **6. Print introductory hangman message from server.** |
|  | **7. Read reply from client.**   1. **Simple: Client expected to immediately start guessing letter per turn.** 2. **Enhanced: Client chooses the category (Either typing in the name of category or number corresponding to the category.)** 3. **Both: Input one letter. Verify if the input is non repeating, singular letter. Convert the letter to small case to make it case-insensitive.** |
|  | **8. Send reply to server.**   1. **Simple: Send the character as a guess attempt.** 2. **Enhanced:** |
| **9. Receive the guessed character.** |  |
| **10. Determine if the guess is correct.** |  |
| **11. if the guess is correct, reveal the position of the letter in the word.**   1. **Enhanced: If the user completely guessed it, increment win count and display leaderboard. If the user is wrong, - 1 attempt.** 2. **Both: If there are zero attempts left, display game over.** |  |
|  | **12: Receive the position of the letter if the letter is guessed correctly** |
|  | **13: Display the position of the letter in the word.** |
| **14: Send Play again prompt to client.** |  |
|  | **15: Receive prompt from server.** |
|  | **16: Determine response:**   1. **if yes, repeat step 4 - 16.** 2. **if no, end program.** |