Musical Chairs, Design

Linnea Dahl 941014-3409
Albin Sundqvist 931010-6019
Tim Kulich 910213-5077
Markus Norström 940413-2355

November 24, 2015

Contents

1 Introduction

For this project a version of the musical chairs will be implemented. By using code and general knowledge of how distributed system works a program will be created where you can play a game of musical chairs.

2 Target audience

This program is a game suitable for anyone at any age to play. With its simplicity it is easy for people to understand, especially if you have played a game of musical chairs before.

3 Plan for the project

In our group of four people the work load will be divided equally. Although one of us will be more invested in the code and another one of us will be more invested in the report.

During the next two months the project will be carried out. The final report is to be finished by the 17th of January. During this time a prototype of the program will be done as well as a demonstration of the prototype. All of these tasks will be carried out before the scheduled deadline.

Week 48 During this week we will begin to make the prototype.

Week 49 We will continue creating the prototype and begin to prepare the presentation for the status of the project.

Week 50 Presentation of the status of the project is to be done. During this week we will also make sure that the prototype is finished.

Week 51 Demonstration of the prototype.

Week 52 Christmas break for the group.

Week 53 Christmas break for the group.

Week 1 We will start writing the final report.

Week 2 The final report will be finished.

4 Design

4.1 Implementing our design

Sockets will be used for client-server communication. The other reasonable option, RMI, is better suited for programs with many functions. This implementation of Dancing Chairs will only have the client call a few functions, making sockets the superior alternative.
The program will be written in Java. Any language is viable since it's implemented using sockets. The project group decided unanimously to use Java since it is a useful language to improve and gain experience in.

4.2 The client side

Each client will first ask the server to join a game, "Request To Join". When the game starts the client will receive a notification. The client will then sit and wait for the server to broadcast the "start the timer". Once the client receive the broadcast its timer will start. When the player presses the sit down button the timer will stop, be recorded in a variable and then immediately be sent to the server.

4.3 The server side

The server wait for RTJ from clients. Once a RTJ is received the client will be put in a playerQueue. The server then waits for the playerQueue to be full (max 5 players) or a client forcestart (must be more than one player and no current game running).
Before starting each round the server checks how many clients that are participating in the game and creates a semaphore. The semaphore count is one less than the active clients, this will be updated each round. The server will compare each of the clients response-timers and the slowest client does not get a chair. That player loses and will be put in the playerQueue and either wait for a new game or leave. Once only one player remains, he will be announced as the winner.

4.4 Timestamp

One of the biggest problems with this game is going to be how to determine which player takes the longest time to sit down on a chair. This is a problem since there will be different delays for different players based on how far away the players are from the server. Since the aim of the game is to be faster than the other players, this would make for a pretty boring game if this was not resolved.
The solution for this is to have every player start their own timer at the moment it receives the signal to "start timer". This ensures that the reaction time is measured by the time the player takes to react, regardless of delay, instead of including the propagation delay between the server to the client. The individual timers will be sent to the server where they can be compared and decide which is the slowest.
A timer at the server side will be implemented to determine when a player has taken too long to answer. This will make the rounds end when a player disconnects or takes too long to answer.

5 Problems that can occur and solutions

5.1 Server Response-time

When the server sends out the 'start timer' message to the clients it will also start a timer that determines how long the players have to deliver a respond message to the server with their timestamp, a cutoff-time. This means that the round trip time + reaction time need to be less than the cutoff time for the players timestamp to be recorded on the server.

This can be solved by increasing the cut-off time. However this would make the clients wait for a longer period of time before continuing with a game when a client disconnect or forgets to sit down.

5.2 Disconnecting

In the case of a player disconnecting or a client fail to send before the servers cutoff-time, the client will be removed from the game. When a player reconnects it will be placed in the playerQueue and wait for a new game to start.

6 Further development

6.1 Graphics and audio

Further development of this project would be to make a graphic version of this game. Other potential additions could be to have actual music instead of a message telling you to choose a chair.

6.2 Improved timer

As mentioned in the "Server-Response time" section, there is an uneven amount of time for each player to send a response. It is not a priority at this time but a possible improvement would be to get rid off or minimize this problem as much as possible.

6.3 Single player

The current design aims at playing against other people. A possible extension could be to have bots to enable a single player mode.