Documentation of the Qualitymeasurements

Max Burgert, Tom Cinbis, Paul Höft, Tim Königl

Goals

In our project we agreed on two different Quality Measurements which we divided into smaller parts to better define our goals:

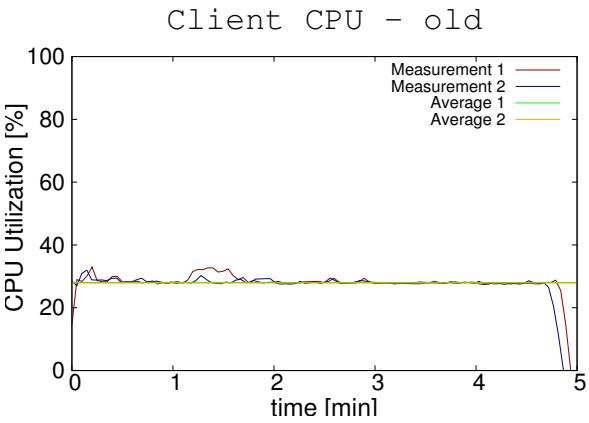
1. Functionality
   (a) **Interoperability:** We want our game to be playable for more than only windows users, so the aim of interoperability is that our game can run on the three biggest operating systems: Linux, macOSX and Windows.
   (b) **Accuracy:** To achieve correct implemented game rules and mechanics we set ourselves the task to make our collision detection as pixelcorrect as possivble.
   (c) **Suitability:** In our project we want all of our classes and methods to be useful and as few as possible redundancy.
2. Efficiency
   (a) **Resource utilization:** For this part we set some average limits for the CPU and the RAM which our program should not exceed: - Client uses in average: 55% CPU and less than 160MB used Heap. (Reference Computer: Intel Core i7 6650U 2.20 GH with air cooling, 8GB RAM, Windows 10 64bit, 2736x1824 Pixel.)
   (b) **Timebehavior**: Because we decided us to implement a real time game, our update time is a part which has too work in real short timeperiods. So our limit was 100ms for three player update operations.

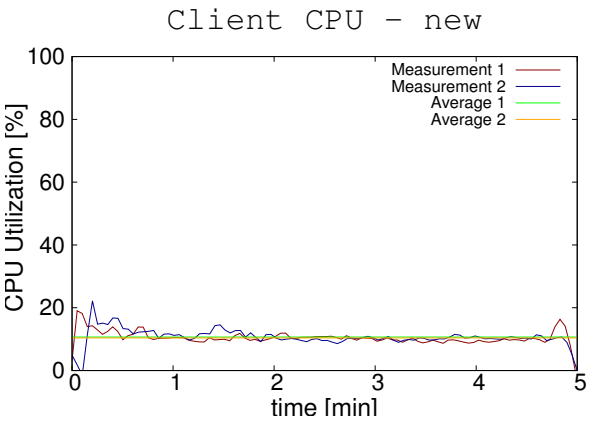
Conduction

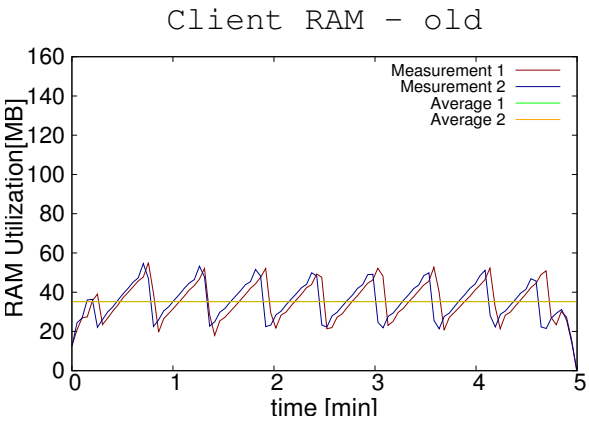
1. (a) Java is already designed for cross platform programming, but even though we had some problems with JavaFX. That was because we used some Windows only commands from JavaFX. After we figured that out and found alternatives we could remove most of these bugs. Furhtermore we used a linux bootable distribution from a USB and a Macbook Pro to test the other two big operating systems.
   (b) The difficulty was to test this, so we had to write UnitTests in JUnit. This was a bit more complicated than we expected,but after figuring out how to use assertions and mock objects the UnitTests founds some problems in our error handling wich we had not thought of.
   (c) We documented everything to ensure that methods have some kind of function and used the all options IntelliJ was providing to impove our code.
2. (a) First we tested with JMeter the used resources while playing some games. To our surprise both client and server had nearly CPU usage of 100% while the used Ram was inside the given range. To fix this we analysed the resource utilization of every single thread with JMeter and found that both our sending and reading from the network threads used most of the CPU. To fix this we slowed them down with a minimal sleep in nanoseconds to ensure they were not operating throughout the whole time. This dramatically reduced our CPU usage and had no effect on the game it self which can be seen later in our results.
   (b) This is ensured by a special unittest for our player class. Therefore we updated our player three times in different directions with a timeout of 100ms. This test never failed on our reference computer, cause we planned the player class efficient enough.

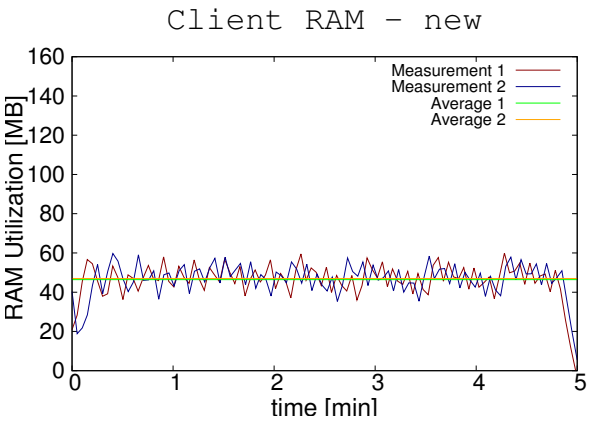
Results

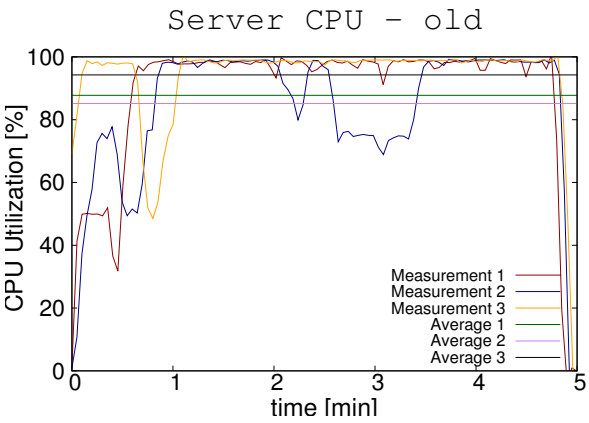
Besides the suitability which is not easy to validate, we reached all of our quality goals. A long time we had many problems with running our game on MacOSX, but we now finally achieved it that it also works on MacOSX besides Linux and Windows which is a big success.
Additionally, for our resource utilization we plotted the measured units over the time directly after milestone 3 and before milestone 4 to see exactly what we achieved. The following graphs show on the left hand side the old measurements and on the right hand side the new measurements with the improvements.

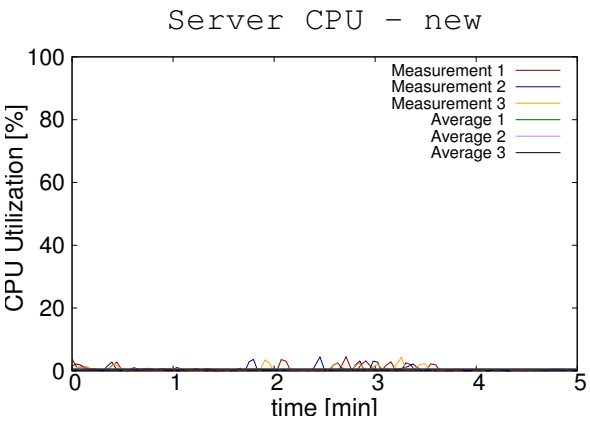


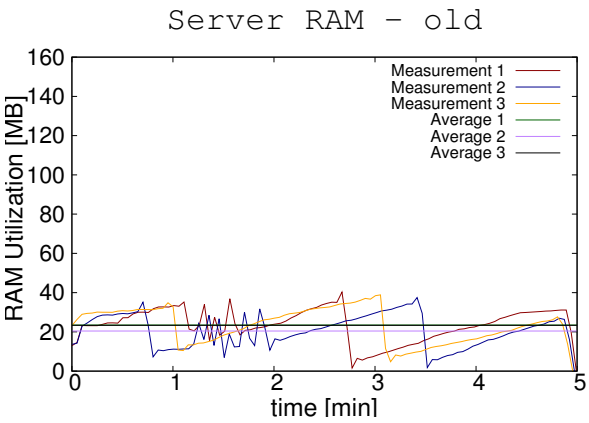


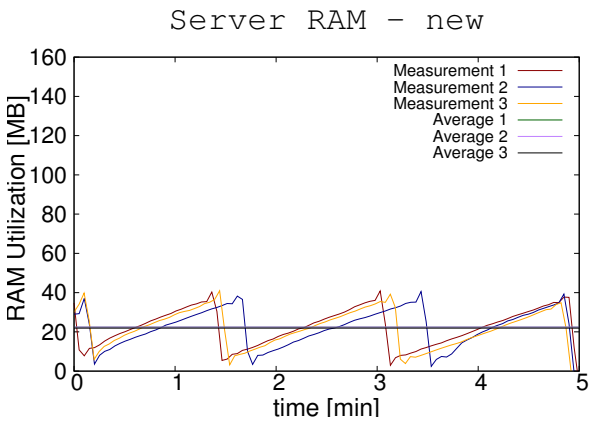












All in all did our project benefit a lot of all of these measurements just from the fact that we planned to reach them and thought about optimizing these important factors.