Uppsala University
Faculty of Science and Technology
Course: Process Oriented Programming (1DT049)
Group number 6
Authors: Markus Ebbesson, Emil Möller, Mikael Svärdström and Mattias Öst
Tutor: Karl Marklund
Version 1
Date of submission: 2012-04-25

sketch online

project proposal of a web based cooperative paint application

Contents

1 Brainstorm

To find an idea of a project that would satisfy and engage all group members a session of brainstorming was done.

1.1 Preconceptions

A few preconceptions existed in the group. There was a will to use several programming languages to be able to try to implement integration between these. The whole group agreed to that graphics are festive and the conclusion of that is that something with a GUI is preferred. There was also an urge in the group of implementing something web based.

1.2 Ideas

Lake evolution - A simulation of progress of several species living in a lake. In beginning of “time” there will only be one simple life form that with some probability factor sooner or later will transform into a more sophisticated life form one after another. Once a life form has evolved it will start to swim around in the lake looking for food and if this particular life form bumps into another life form they will fight and only the strongest will survive. In the case of a life form not finding any food they will eventually die and if it finds food it will be able to lay eggs. Every life form has several attributes that control how well they will survive.

Traffic simulation - A traffic system with multiple cars and a few crossroads with traffic lights. Random factors make the simulation different every time. This project can be done in both small scale, including a few cars at a time arriving randomly driving through a couple of crossroads, and big scale, including probability of cars crashing or police cars arriving and given priority in the traffic et cetera.

Seat reservation system - An application for seat reservation for a restaurant, cinema or such. Gives users the ability to book one of the seats that are available at the moment. If a seat is taken a user will not be able to book it. In this case the most interesting problem about this application would be to implement a solution to where two or more users want the same seat at the same time.

Package handler - An GUI-based application that gives a Windows user the ability to overview and search for available and suited applications that is not installed on the system. Then being able to automatically download and install one or more of the selected applications instead of visiting different websites and going thru the installation process of every program. Similar to package handlers like pacman or apt-get.

Online multiuser paint - A web based application that gives two or more users the ability to draw on the same workspace at the same time. This project is possible to do in both small scale and big scale by the amount of added functions such as colors, pen size and text. The discussions in the group resulted in the mutual ambition of a smooth

drawing board that updates in real time, for example even while a user is drawing a line and have not yet finished it or released the mouse button. This would be a challenge as well as something unique that similar applications online do not offer.

1.3 Summary

The five ideas that the group have discussed have some similarities with each other. They all interact directly with the user using some sort of graphics to display the progress and what is happening in the program.

For the concurrency part of the summary, the first idea “Lake evolution” would have many different creatures. All the creatures would share the same map to move on. The functionality behind the animation of creatures moving will be some kind of parent process creating creature-processes for running their movement functions. So later on when the animation is up and running each creature will get updated on the map concurrently.

The second idea “Traffic simulation” would work in similarity to “Lake evolution”. Each car’s movement function would be created as a process and they would run concurrently with the animation.

The third idea of a “Seat reservation system” on the web. Every user should see the latest view, so when a user reserves a seat another user will not be able to reserve that same seat. When two users are trying to reserve the same seat at the same time the server will choose the user who were first to avoid data races.

“Package Handler” was the fourth idea. The concurrent part of this idea is while installing and downloading a program the interface should still be responding and running. The server side of this application should also be able to handle several jobs concurrently.

The fifth idea “Online multiuser paint” would deal with some kind of animation from interpreting the user’s mouse events. The main problem would be to get the animation to run smoothly with multiple other users drawing simultaneously, possibly while the own user is also drawing.

1.4 Conclusions

The group brainstorming went pretty well; in the beginning no one really had any expectations at all. Once the discussions started all members got very much engaged, talking about different ideas and eventually why, and to some extent how, the group would implement the idea in question. The group agreed that these five ideas were better suited for our main problem, concurrency, and left out a few worse ideas.

A good thing at this moment is that all group members now know what everyone wants to achieve with this project. Also everyone has got the knowledge about the co-worker's programming experiences and what each individual should start to learn about. One thing that the group can improve for next time is that everyone should at least spend some time alone and think about various ideas before the brainstorming session.

2 Project Selections

The "Online multiuser paint" idea was chosen because it fitted the preconceptions best and after a voting attached in Attachment 1. The idea is eminently graphical, which was one of the preconceptions, and is also most suitable to implement as a web application.

After some research on the web we found that Google and CoSketch have similar solutions to this idea. Those solutions do not update in real time while a user is drawing but waits until the user releases the mouse button. This is something that this project will try to achieve to be able to offer a unique online drawing experience.

The fact that this idea is web based, graphical and can be implemented using several different languages makes it the most interesting idea. The main functionality of the project will be written in javascript but as the project progress more design and functionality will be added depending on time factor using other languages such as xHTML/HTML5, CSS/CSS3, PHP and the jQuery-library.

Most focus will be put on the main challenge in the project of making the application work smooth and with proper concurrence so the user will have a good experience while painting/drawing with friends using our application.

3 System Architecture

A user loads the application on the website and receive an own unique instance of a sketch board with an unique URL to this specific sketch board. The user will be able to manipulate the sketch board area and these changes will be sent to the server which will handle the intelligence of how the sketch board looks like for all the users connecting to this specific sketch board. The server side application is also responsible for concurrently sending and receiving changes made by the users of the application. This is illustrated and attached in Attachment 2.

4 Concurrency models and programming languages

Our way of solving the problem will involve spawning several processes and letting them concurrently work towards a certain mutual goal. Since this being the case a big effort will be spent on avoiding deadlocks, starvation and synchronization problems.

The goal is for all users of the application to experience the same smooth experience sketching at the same time on the same sketch board. In this particular application we

as a group all relate to the dining philosophers problem since the server side application has the responsibility of giving all the users the correct information of what should be displayed and at the same time the users wants to send information to the other users through the server. The problem being several processes that are ready to execute since there is no limited resource, but the applications still have to consider if it would be appropriate. Introducing a conductor solution will give the processes ability to ask if they should execute or not. An alternative solution would be using a resource hierarchy solution.

5 Developments tools

Here is a short introduction about the developments tools that will be used within the project.

5.1 Source code editor

Since all of the group members are most used to working with Sublime, Emacs and Notepad++ they will be used for editing and writing the code. All of these tools offer good support for the various syntaxes of the languages that will be used, including color highlighting, line numbers, automatic indentation et cetera.

5.2 Revision control and source code management

To manage and share code and other files a Github repository will be used. The main benefit of using this is that it is possible to give all group members an overview via the web of the project and project files with read and write access. Other useful functionality provided by Github is the todo-list, an overview of bug- and issue lists, as well as a progress function with the project. Since the project is developed by only four people all group members will have full access to editing these functionalities.

5.3 Build tool

Front end javascript and design code will be interpreted by a web server service on our server. The server side program will be implemented in Erlang and will probably take advantage of own created makefiles. Erlang has the advantage of great support for concurrency and is proven to be effective handling processes.

5.4 Unit testing

For the testing part of the project there are several frameworks for the javascript tests to choose from. Today there is no standard or ultimate framework for the javascript but some research online have shown that popular frameworks are QUnit (for jQuery), JSUnit and YUITest, therefore one of those will be chosen.

Considering testing the Erlang code on the server EUnit will be used since it is simple and it is the only unit testing framework for Erlang that is known to all undersigned.

5.5 Documentation generation

In terms of the documentation part of the project there is no standard generator for javascript. There are several options and JSDoc Toolkit will be used for this project since it seems to be the most established documentation generator used by most coders. EDoc is the documentation generator that will be used for the Erlang code on the server.

Attachment 1

	Markus	Emil	Mikael	Mattias	**Total**
Lake evolution	7	6	9	8	30
Traffic simulation	6	3	3	4	16
Seat reservation system	5	5	6	5	21
Package handler	3	4	4	5	16
Online multiuser paint	9	8	8	8	**33**

Voting of ideas, every person voted on each idea on a scale 1-10.

Attachment 2

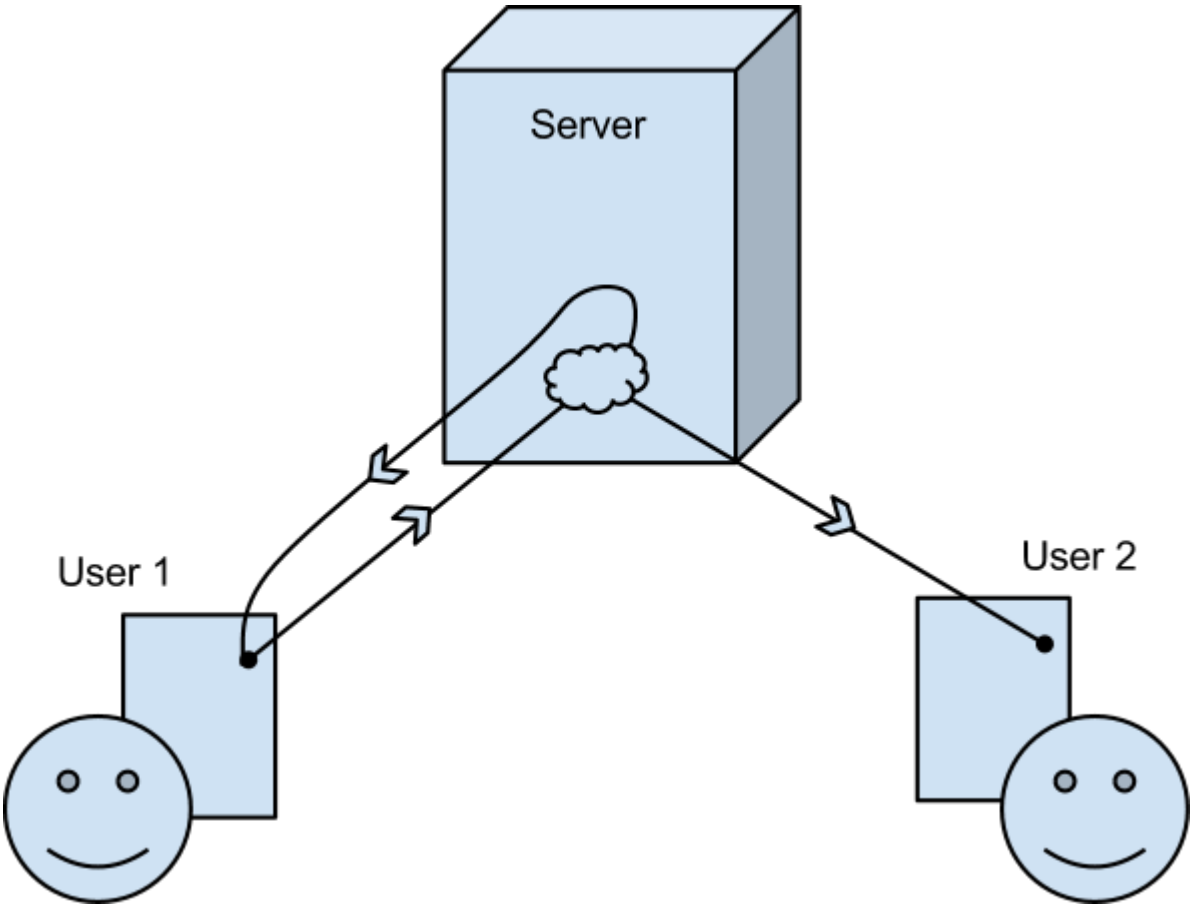

Illustration of the information that has to be sent when User 1 draws a dot in his workspace. Information is sent to the server, then interpreted and saved. The change is then confirmed and sent to all members including User 1.