Krak-vejkort

Visualiseringen

af
Jacob Stenum Czepluch (jstc@itu.dk),
Niels Liljedahl Christensen (nlch@itu.dk),
Mikkel Larsen (milar@itu.dk),
Sigurt Bladt Dinesen (sidi@itu.dk)

IT-University
Copenhagen
First year project
Rasmus Pagh
April 3, 2012

Contents

1 Introduction

This will be our introduction to this small report...

We should remember to have a quick bit about how the visualisation works at the moment. The touchpad zoom, mouse move around, and the relative to mouse pointer placement zoom.

This is a section. Use it. Love it. This is an example of a long text This is an example of a long text This is an example of a long text This is an example of a long text This is an example of a long text This is an example of a long text This is an example of a long text

Then we have a new line with indent This is an example of a long text This is an example of a long text This is an example of a long text This is an example of a long text

2 Design choices

In this section we will go through our design choices. We will go through things like design pattern, data structure, and visualisation. At the same time we will give explanations to our choices.

2.1 Design Pattern

For our overall architectural pattern we have chosen to use the Model, View, Controller(MVC) pattern. We chose to do so, to make sure that we have a very modular and easy to maintain class structure. Using the MVC pattern results in seperation of the different aspects of our application, while still providing a loose coupling between these elements. During this first visualisation part of the application it has proven very useful to us, since we have been chancing our classes and data structures quite a few times.

2.2 Data Structure

We started out making a simple data structure that put all the edges into one big ArrayList. This was not fast, but we chose to do it, to make sure that things were working before we tried implementing a more complicated, but faster data structure. We had several data structures up for discusssion, but we ended up using a quad tree. We actually started out making a kd-tree, but since we had some trouble implmenting it correctly, we decided to take a look the quad tree aswell. Luckily it turned out to perform very well, and we did not even have to sort our input for it to work well. In our discussion section we will further discuss pros and cons of the different data structures.

2.3 Visualisation

2.3.1 Platform

To start with we had to decided whether we wanted to use Java and Swing or Java, JavaScript, SVG, and HTML for the visualisation part. There we good and bad things about both. The good thing about Swing was that we all knew how to use it, and we were sure that we were able to do what we wanted in Swing. The "bad" thing about Swing is that it does not have as many possibilities as HTML with JavaScript and SVG. The good thing about HTML with JavaScript and SVG would be that it is very easy to customize to look nice and sleek. However, none of us have ever used it before, and since we would rather spend time making something that work and has a nice data structure, we chose to go with Java and Swing.

2.3.2 How everything is drawn

As mentioned above we chose to use Swing and the BasicStroke API to do the drawing of the map.

Technically the map first draws all of Denmark. It is however only the two biggest road types that are drawn. We decided to do so, because it is both unnessesary to draw all roads, when you are very far away from the map, but also to make the program run smoother. The program is designed to show more and more road types the more you zoom in towards the map. It is only the roads that are inside the given view that are drawn. We have however made a "buffer" that finds the longest road in the view, and extends the frame outside the view with the given lengt. This is done to make sure that all roads are in the view are drawn and vissible.

2.3.3 User interaction

We wanted to make a very simple yet featurefull user interaction. This resulted in a user interface with no physical buttons. All you need to navigate the map is a mouse. To zoom in and out you either scroll up or down. The mouse pointer decides what the zooming point is. It is also very easy to go up and down or from side to side. This is done my simply dragging and dropping the map with the mouse pointer. Our main reason for making the user interaction like this, is that this way is basically the most intuitive way for a lot of people to navigate and interact with maps and alike, due to the great amount of tablets and smartphones.

2.3.4 Types of the Krax-data

I am not quite sure what we mean here. Is it the content of the .xml .dtd or the .txt files? I just need to be sure what we mean before I write this part.

2.4 Outline

MVC
: Great!

Data structure
: QuadTree

Visualisation
: Platform

: How everything is drawn

: How the user interacts

: Types of the Krax-data

3 Implementation

This sections should describe our implementation.

3.1 Outline

Model

The Edge data type

XMLReader

QuadTree

FormatConverter

/KraxToXMLConverter

View

The overall class structure

The use of the Observer design pattern

Mouse and window listeners

Controller

The method calls from and to the map / data structure

4 Discussion

In this sections we will discuss what could have been done better, and/or what we think we have done right.

4.1 Outline

Limitations

Possible improvements

Data structure discussion

QuadTree vs KD-tree

Balanced vs unbalanced QuadTree

5 Conclusion

This section will make a quick summary and conclusion on out project so far.

5.1 Test 2

This is a subsection