Final Project

ETH Zurich

Hand-out: 18 December 2006
Due: 22 January 2007

Please solve this assignment either alone or in groups of two. Try to form uniform groups according to your respective programming experience, so that you can choose a task that is equally challenging for both of you. Also, both members of a project group should be in the same exercise group.

1 The project(s)

As you will spend a considerable amount of time on the final project, we want you to be very happy with it. Therefore, we give you a lot of freedom. In short, you should provide an extension to Traffic.

By now, you have done several exercises with Traffic, and probably you were annoyed about something, or you desperately missed a certain game feature. The time has come where you can extend Traffic and give it your personal touch. You can do almost anything you want, provided that your assistant agrees with what you have in mind, and that your idea is worked out clearly.

Another important point: your project must not have any offensive content, and it should not use copyrighted material.

2 Examples

To help you in choosing a project and to give you an idea of the level of difficulty we expect from your projects, here are some examples of possible projects:
Location information

Provide Traffic users with the ability to select a location on the map and:

- Enter information about it (e.g. if a monument some description of its history), and have the information stored in association with the map.
- Retrieve previously entered information about the location.
- Edit it.

Level of difficulty: low

Route planner

Relying on statically available (XML) data about the VBZ tram/bus/local train schedule, provide a functionality through which a user can select a starting place, one or more destination places, and an approximate time, and get a route description: itinerary, stops and the corresponding times, possibly an animation.

Level of difficulty: low to medium

Map-based web search

Using the Google API, add a mechanism enabling Traffic users, when they move the cursor around, to:

- See, in a tooltip-like window, information about the location, deduced from its name and obtained by displaying the first n (e.g., 10) hits on Google.
- By clicking, cause a browser window or tab to appear, with the results of the search.

Notes:

- It is possible to coordinate this project with the “Location information” project; in that case the keywords used for the search may include, in addition to the place name, text from the information that has been entered explicitly.
- Google is the most obvious example but other search engines (Flickr for images...) are possible.

Level of difficulty: medium

Google maps

Assuming precise coordinates are available, provide the ability to display the Google Earth view of a Traffic location or a tram/bus itinerary.
Construction site marking

Mark certain stations as “out-of-order” and have the route planner compute alternative routes to the destination.

Level of difficulty: medium

Avoiding tram collisions

Take care of avoiding collisions between trams during animations. When it happens that a tram is running through another with its front, it should stop and wait for the other tram to pass. As an additional task, it would be interesting to tackle possible traffic jams (deadlocks) that could arise when each tram is waiting for some other tram to move.

Level of difficulty: medium

Detecting tram collisions

The idea here would be not to avoid collisions, but to detect them and visualize them in a nice way.

Level of difficulty: medium

Traffic map editor

Write a map editor that allows to add places, roads and line sections to a map and then store it into an xml file that can be loaded by other traffic applications.

Level of difficulty: medium

Topological details

Add topological details such as mountains, rivers, or lakes to the map. Mountains could be added randomly.

Level of difficulty: medium

Traffic board game

Based on the Traffic engine (and the “Flat Hunt” application), use a special map to be your game board. Any boardgames that require the user to move in between locations “dots”, “fields”, “squares”, etc.) are suited. Examples include “Risk”, “Monopoly”, “Leiterli-spiel”, etc.

Level of difficulty: medium to high (depending on the game implemented, and its level of complexity)

Plan a Zürich trip

Automatically schedule trips across Zürich. Let the user define a set of stations he wants to visit during his trip. Have the computer calculate the shortest or at least a short route that visits all stations requested.

**Traffic flow simulations**

You know some statistic data (home, workplace, ...) of the workers in Zürich. Every morning the whole Zürich is going to work. A stupid worker always tries to take the shortest route. But a smart commuter now and then tries different routes and takes the fastest one. Can you simulate this in Traffic?

Level of difficulty: high

**Better shortest path**

Shortest path calculation at the moment is provided in two variants: shortest path by minimal distance (taking the polypoints of TRAFFIC_LINESECTION to find this distance), and shortest path by minimal number of switches between means of transportation (e.g. switches between trams). Both versions work quite slow since the number of nodes (type TRAFFIC_NODE) of the graph is high. A TRAFFIC_NODE is for example a specific stop of a TRAFFIC_LINE in a distinct direction. An undirected TRAFFIC_LINE constitutes of 2*n TRAFFIC_NODEs if n is the number of stations it stops at, and every new TRAFFIC_LINESECTION in the system adds two more nodes to the graph. This results in a graph with a huge number of nodes. While this might seem a waste of space and even worse leads to a very long computation time for shortest paths, it is necessary to be able to calculate the shortest path with minimal number of switches between transport vehicles using the classical Dijkstra algorithm.

A possible project would be to investigate improvements of the algorithm performance or to add new types of shortest paths calculations, e.g. fastest journey taking into account timetables, restricting the means of transportation that should be taken (e.g. only walking on the roads).

Level of difficulty: high

3 The Tasks and Their Deadlines

3.1 Initial Description of the Idea

You must come up with a brief description (a few sentences) of what you want to do in the project until 21 December 2006. Email this brief description to your assistant. He/she will reply within a few days if he/she does not approve the idea. If you do not get such an email, consider your idea approved and you can already start working on the next tasks.
3.2 Project Description, Analysis, and Design

You must come up with a thorough description of the requirements of the project. If you are working in a group of 2, this must also include a clear statement of how the work will be divided between the group members.

Before you start to program your application, you should think about how to best model your system. Consider such questions as:

- What should the final system look like?
- What are the advantages/drawbacks?
- How do the classes work together?

You must write a short report describing your design decisions and the overall architecture of your system. This report should include a BON diagram of your classes. Deadline for submitting this report and the requirements document: 8 January 2007.

3.3 Implementation and Documentation

Implement the system you designed. Make sure that your code is readable, well-commented and equipped with contracts. Then test your system to make sure that it works as specified in the requirements. You also have to write a (short) developer guide describing your system. You may reuse parts of the design document for this.


4 The Presentations

In the exercise session

You will have to present your project in front of your group and assistant, in the exercise sessions on 23 January and (for the Knuth group only) 25 January. Your project might be selected to be presented in front of everyone during the last lecture (30 January) - read more below. Your presentation in the exercise session should take no more than 5 minutes if you have worked alone and 10 minutes if you have worked in a group of 2. You are not only to show how your project works, but also to talk about your design decisions and your experiences during this project. All group members should participate in the presentation.

In the lecture

In the last lecture of the semester (30 January 2007) we will have a few presentations of the most interesting projects. These selected projects will be demoed in front of everybody (Prof. Meyer, the assistants, and your colleagues). These projects will be selected by the assistants based on the presentations that you do in the exercise sessions.
5 Submission: What and How to Hand in

All the deliverables of the project will be uploaded to the wiki located at \url{http://wiki.se.inf.ethz.ch/info1_06/}. To access the wiki, use the following

- user name: students
- password: info1

Your assistant will instruct you on how to use the wiki. You will have to create a page for your project according to the template that you will be given. This page will contain the following sections:

- Project name: state the name of the project here
- Student name(s): name(s) of the 1 or 2 group members
- Initial description of the idea: a few sentences with a brief description of what you want to do in the project
- Project description: link to an uploaded PDF document with a detailed description of the requirements of the project and the division of work between the group members (if applicable)
- Analysis and design: link to an uploaded PDF document for the analysis and design phase
- Implementation: link to an uploaded .zip file containing the code
- Developer guide: link to an uploaded PDF file

Note: The source code that you upload on the wiki should only contain .e files and the project ecf file. Don’t include any hard coded paths in the ecf file. In other words, you should only have relative paths in the ecf file.

To upload files to the wiki you will need to create a personal account. Create such an account with your full name as user name. Only the above-mentioned formats can be uploaded to the wiki.

6 The Assessment Criteria

- Design
  - Extendibility
  - Ease of use
- Functionality
  - Does the implementation satisfy the specification
- Quality of contracts
  - Preconditions
  - Postconditions
  - Class invariants
  - Loop invariants and variants
- Documentation
7 Final Remark

This exercise gives a lot of freedom concerning the amount of work you invest. You may improve and upgrade the functionality of your application to any extent you want. Feel free to use your imagination! There are no limits. The main goal is to program and gain experience in extending large systems.