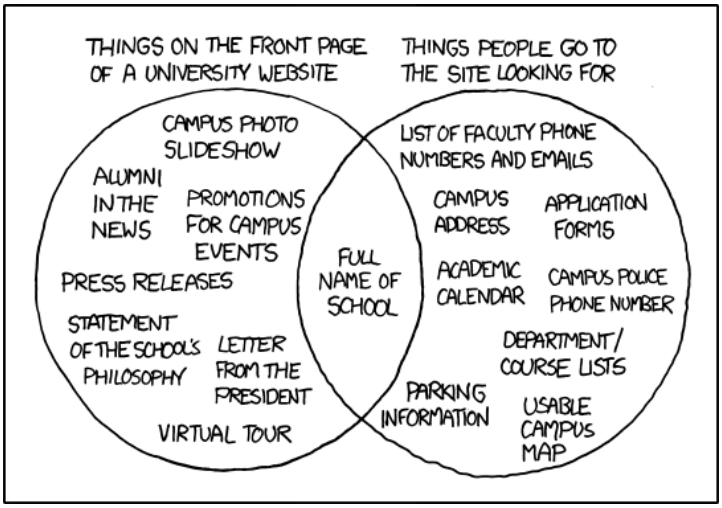
Assignment 1: Getting started

ETH Zurich

Hand-out: Tuesday, 20 September 2011



University Website © Randall Munroe (xkcd.com)

1 Welcome to ETH!

Goal

The goal of this task is to familiarize you with the infrastructure provided by ETH Zurich and to make sure that you have subscribed to this course.

To do

- Use your personal computer or log in to a computer in one of the ETH computer rooms.
- Log in to the n.ethz account administration page (http://www.passwort.ethz.ch). After logging in you will see the welcome screen. Click on "Passwort ändern"(top left of the screen), change your password[1] and log out. After this you can log on to the ETH webmail interface (https://mail.ethz.ch/exchange) with your user name and your newly set password.

[1] Your password should be simple enough for you to remember; yet it should be complex enough so that other people cannot guess it. See password recommendations at CERN: http://security.web.cern.ch/security/passwords.

- If you are using an ETH computer, try to find the Eiffel development environment - EiffelStudio - on it. If you are using your personal computer, follow the instructions in the next task to install EiffelStudio.
- Bookmark the Forum der Informatik Studierenden: http://forum.vis.ethz.ch. The forum provides a platform for questions and discussions with your peers.
- **IMPORTANT:** Make sure that you have subscribed to this course (and all the other courses you attend) on http://www.mystudies.ethz.ch. Otherwise it is difficult to give you your testat at the end of the semester.

Hint

You will find that your n.ethz login and password are very useful on many other ETH web pages, e.g. the http://www.mystudies.ethz.ch page mentioned above.

To hand in

There is nothing to hand in; the task is accomplished as soon as you have subscribed to the course.

2 EiffelStudio installation

Goal

In this task you will install the software development platform EiffelStudio. If you intend to work in a computer lab of ETH you will not need to install EiffelStudio. Proceed with task 3. We support the following operating systems:

- Windows (Microsoft C compiler only!)
- Linux
- Mac OS X

A general suggestion: if you encounter precompilation problems during installation, and you think you have fixed them, before precompiling again you may want to delete the EIFGEN directory, which may contain corrupted files at this point.

To do

For Windows

1. As a first step you need to get the Microsoft C compiler. If you already have it installed (e.g as part of Microsoft VisualStudio), skip this point. To install the Microsoft C compiler, follow the instructions at http://dev.eiffel.com/index.php/Installing_Microsoft_C_compiler.
2. Download EiffelStudio 6.8 from sourceforge[2].
3. Start the installer and follow the instructions. Choose the option *EiffelBase, WEL and EiffelVision2* in the step "Precompiled Libraries". The precompilation of the libraries takes a long time during installation, but will significantly speed up later compilations.

[2]Click on the hyperlink, or go to http://sourceforge.net/projects/eiffelstudio/files/ and click on Eiffel Studio 6.8. Then click on the top build in the list (86627) and finally on "Eiffel68_gpl_86627-windows.msi".

For Linux

Please note that we assume a basic set of preinstalled tools and libraries, including tar, gcc, pkg-config, headerfiles, and others. We recommend to install EiffelStudio with the default locale (en_US).

1. Install the *development* versions of libgtk and libxtst libraries, if you don't have them yet (`libgtk2.0-dev` and `libxtst-dev` on Ubuntu).
2. Pick a directory of your choice without any spaces in the path, for example /opt:
   `cd /opt`
3. Download EiffelStudio 6.8 from sourceforge[3]. Let's suppose you downloaded it to your desktop.
4. Extract it: `sudo tar -xvjf ${HOME}/Desktop/Eiffel68_gpl_86627-linux-x86.tar.bz2`
5. Set the following environment variables (note that how and where to change environment variables depends on which distribution you use). Make sure you set these permanently and systemwide! Note: If you download the 64 bit version of EiffelStudio make sure to change the environment variable ISE_PLATFORM accordingly (as explained in the installation instructions of EiffelStudio).

   Add the following lines to the end of `/etc/profile` with: `sudo gedit /etc/profile`

   ```
   export ISE_EIFFEL=/opt/Eiffel68
   export ISE_PLATFORM=linux-x86
   export PATH=$PATH:$ISE_EIFFEL/studio/spec/$ISE_PLATFORM/bin
   ```

6. Restart your account.
7. Prompt: `sudo chmod 777 /opt/Eiffel68/precomp/spec/linux-x86`.
8. To speed things up later, we need to precompile some libraries. Note that this may take some time.

   ```
   cd /opt/Eiffel68
   ./make_install
   ```

   Answer 'y' to both questions: this will build the precompiled libraries.
9. To start EiffelStudio type `estudio` in the console.

For Mac OSX

The installation has been tested on Mac Os X Leopard, Snow Leopard and Lion.

1. Please follow the instructions on http://dev.eiffel.com/EiffelOnMac under section "Using Macports".
2. After having installed and compiled EiffelStudio, as indicated by the message at the end of the installation process, you will have to update some configuration files. To achieve a consistent setup for `bash` and `X11`, we recommend to edit the following files located in your home directory: `.bashrc`, `.bash_profile` and `.profile`. If these files do not exist, create them. If they exist and already have content, add the following lines anywhere.

[3] Follow the instructions in footnote 2: the file you are looking for is "Eiffel68_gpl_86627-linux-x86.tar.bz2".

3. First you need to edit `.bashrc` according to the instructions you get after a successful installation. The following is an example:

```
export ISE_PLATFORM=macosx-x86-64
export ISE_EIFFEL=/Applications/MacPorts/Eiffel68
export PATH=$PATH:$ISE_EIFFEL/studio/spec/$ISE_PLATFORM/bin
```

   Copy your individual output to `.bashrc` or modify the above given example to your needs.

4. Then you should modify `.bash_profile` to link to `.bashrc`:

```
. ~/.bashrc
ENV=$HOME/.bashrc
export ENV
```

5. Finally, you should modify `.profile` to link to `.bash_profile`. This step is necessary to propagate the environmental settings to the X11 X-window system:

```
. ~/.bash_profile
```

6. To launch EiffelStudio, type `estudio` in a bash terminal window.

To hand in

There is nothing to hand in.

3 Your first Traffic program

In this task you will download Traffic and experiment with some feature calls. You need to have EiffelStudio installed (see task 2).

To do

1. Download http://se.inf.ethz.ch/courses/2011b_fall/eprog/assignments/01/assignment_1.zip and unzip it to a folder of your choice.

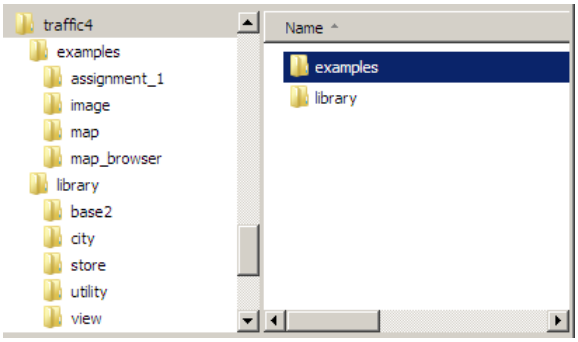


Figure 1: Directory structure of Traffic.

2. Figure 1 shows the directory structure of the archive. The top-level directory `library` contains the core of Traffic: Eiffel class files that model the transportation system in a city and support its visualization. The other top-level directory `example` contains some applications that use the Traffic library. For instance, `map_browser` allows you to browse the map of Zurich and display information about the city objects. Directory `assignment_1` contains the application that you will explore in this assignment. During the semester we will add more application examples (in particular, other assignments).

3. Start EiffelStudio: you will see the dialog of figure 2. If this does not happen automatically, go to `File` ▷ `Open project` and it should appear.

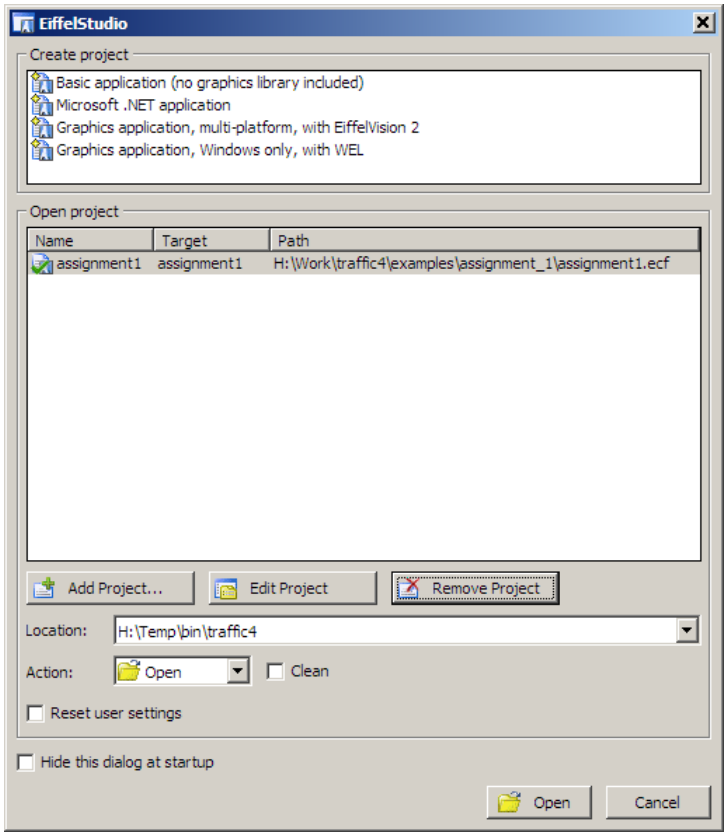


Figure 2: Open an Eiffel project

4. Click on `Add project` and choose the file found under the directory where you unpacked Traffic, at `examples/assignment_1/assignment_1.ecf`. Click `Open`. This will compile the test application.

5. Launch the program by clicking on the green Start button (see figure 3); you will see a map of Zurich center. You can move the map by dragging and zoom it with the mouse wheel: just like in Google Maps. Notice that some items on the map are highlighted (namely, the ones that depict Polybahn and its two stops) and the console at the bottom of the window displays information about Polybahn. After a couple of seconds, a cable car will appear and start moving along Polybahn. Close the application by clicking on the stop button (the red square) in EiffelStudio or just close the application window.

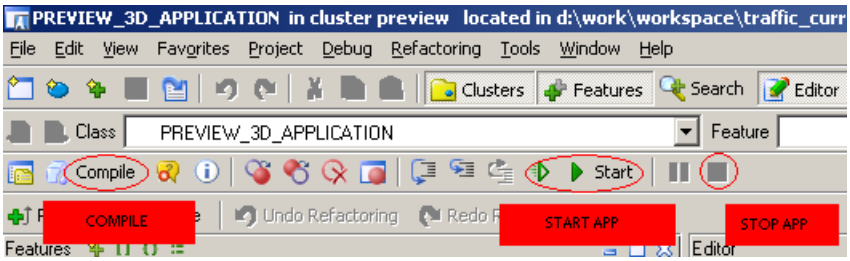


Figure 3: Compiling a project, starting and stopping an application

6. Open the class *PREVIEW*. In the feature *explore*, between the **do** and the **end** you will find the following text:

```
central_view.highlight
polyterrasse_view.highlight
polybahn_view.highlight
console.output (polybahn)

wait (3)

zurich.add_public_transport (Polybahn_line_number)
zurich_map.update
zurich_map.start_animation
```

7. Using two dashes (−−) at the start of each line, turn the first three lines starting from *central_map.highlight* into comments. By doing so, you are telling the machine not to take them into consideration. Save, recompile the project (by clicking on the compile button in EiffelStudio) and launch it again. You will see that Polybahn is not highlighted anymore. This makes sense, because as we just said, the commented instructions are not part of the code that will be executed anymore.

8. Now, without uncommenting the previously commented lines, try writing the instructions all by yourself. You may want to try writing one line at the time, then compile and execute to see the effect. Notice that when you write an object name (the first part of the instruction, before the dot) and then the dot, the "completion" feature of EiffelStudio jumps in and lets you choose between the available features that you can invoke on that object.

9. If you haven't done it yet, please read through sections 2.1 and 2.2 of chapter 2 of Touch of Class. Note that examples in the book use an older version of the Traffic library. If you want to try out these examples, you can download Traffic 3.3.1.1134 from http://download.origo.ethz.ch/traffic/2070/traffic_ev_1134.zip.

To hand in

There is nothing to hand in.