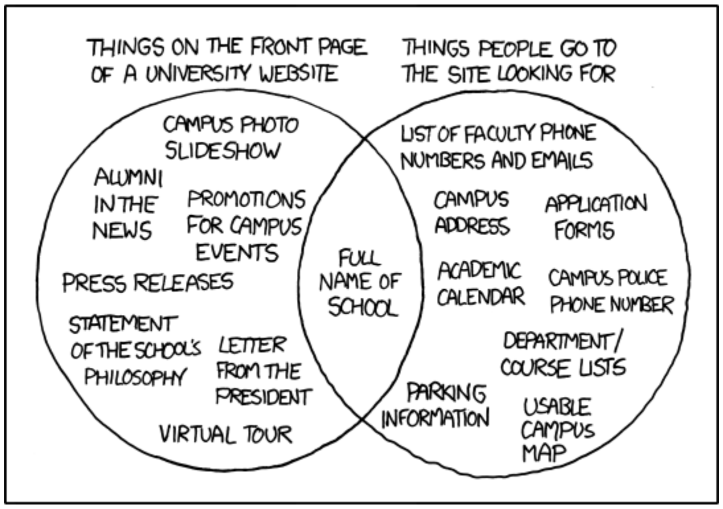
Assignment 1: Getting started

ETH Zurich

Hand-out: Monday, 17 September 2012


THINGS ON THE FRONT PAGE OF A UNIVERSITY WEBSITE
THINGS PEOPLE GO TO THE SITE LOOKING FOR
CAMPUS PHOTO SLIDESHOW
ALUMNI IN THE NEWS
PROMOTIONS FOR CAMPUS EVENTS
PRESS RELEASES
STATEMENT OF THE SCHOOL'S PHILOSOPHY
LETTER FROM THE PRESIDENT
VIRTUAL TOUR
FULL NAME OF SCHOOL
LIST OF FACULTY PHONE NUMBERS AND EMAILS
CAMPUS ADDRESS
APPLICATION FORMS
ACADEMIC CALENDAR
CAMPUS POLICE PHONE NUMBER
DEPARTMENT/ COURSE LISTS
PARKING INFORMATION
USABLE CAMPUS MAP

University Website © Randall Munroe (http://xkcd.com/773)

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 Inforum: `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 `https://www.lehrbetrieb.ethz.ch/myStudies/`. 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 `https://www.lehrbetrieb.ethz.ch/myStudies/` 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

If you encounter any problems during EiffelStudio installation please consult the troubleshooting page `https://bitbucket.org/nadiapolikarpova/traffic/wiki/Troubleshooting`, the Inforum, or ask your assistant (in case you are not assigned to an execise group yet, ask any assistant).

In case you encounter a crash of the EiffelStudio development environment, please click the "Submit bug report" button and provide the following credentials: login *ethinfo1*, password *ethinfo1*.

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 7.1 from sourceforge: go to `http://sourceforge.net/projects/eiffelstudio/files/`, click on Eiffel Studio 7.1 and then on the top build in the list

(88986). Finally choose "Eiffel71_gpl_88986-windows.msi" (32 bit) or "Eiffel71_gpl_88986-win64.msi" (64 bit), depending on whether your C compiler is 32 or 64 bit (if in doubt choose 32 bit).

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.

For Linux

Please note that we assume a basic set of preinstalled tools and libraries, including tar, gcc, pkg-config, and others. We recommend installing 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. Download EiffelStudio 7.1 from sourceforge: go to `http://sourceforge.net/projects/eiffelstudio/files/`, click on Eiffel Studio 7.1 and then on the top build in the list (88986). Finally choose "Eiffel71_gpl_88986-linux-x86.tar.bz2". Let's suppose you downloaded it to your desktop.
3. Extract it: `sudo tar -xvjf ${HOME}/Desktop/Eiffel71_gpl_88986-linux-x86.tar.bz2 -C /opt`
4. 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/Eiffel71
   export ISE_PLATFORM=linux-x86
   export PATH=$PATH:$ISE_EIFFEL/studio/spec/$ISE_PLATFORM/bin
   ```
5. Restart your account.
6. To start EiffelStudio type `estudio` in the console. When you open your first Eiffel project (see task 3), EiffelStudio will ask if the required libraries should be precompiled. Answer **yes**; precompilation will take some time, but it will speed up things later.

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". When giving the command *sudo port install eiffelstudioXX* replace *XX* by 70.
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. 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/Eiffel70
   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. First you have to download the EiffelBase2 library, which you will need in this and other assignments. Download `http://se.inf.ethz.ch/courses/2012b_fall/eprog/assignments/base2.zip` and unzip it into `$ISE_EIFFEL/library/`, where `$ISE_EIFFEL` is the directory where you installed EiffelStudio (for example `C:\Program Files (x86)\Eiffel Software\EiffelStudio 7.1 GPL\` on Windows). **Note:** it is important that you use this directory; to make sure everything is correct, check that you have the file `$ISE_EIFFEL/library/base2/base2.ecf` on your machine.

2. Download `http://se.inf.ethz.ch/courses/2012b_fall/eprog/assignments/01/traffic.zip` and unzip it to a folder of your choice.

3. 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).

4. 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.

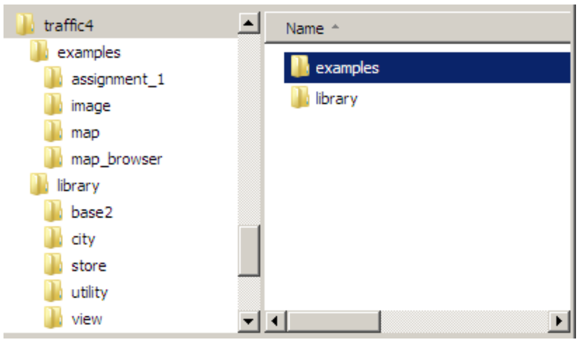


Figure 1: Directory structure of Traffic.

5. 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 application.

6. 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.

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

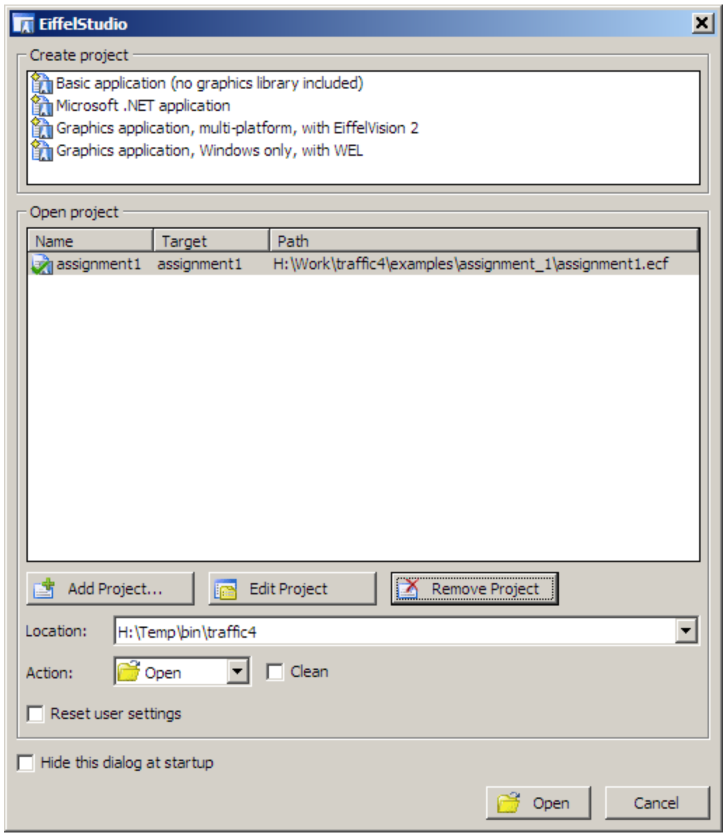
```
Central_view.highlight
Polyterrasse_view.highlight
Polybahn_view.highlight
console.output (Polybahn)

wait (3)

Zurich.add_public_transport (Polybahn_line_number)
Zurich_map.update
Zurich_map.start_animation
```

8. 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.

9. 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.


EiffelStudio
Create project
Basic application (no graphics library included)
Microsoft .NET application
Graphics application, multi-platform, with EiffelVision 2
Graphics application, Windows only, with WEL
Open project
Add Project...
Edit Project
Remove Project
Open
Cancel

Name	Target	Path
assignment1	assignment1	H:\Work\traffic4\examples\assignment_1\assignment1.ecf

Location: H:\Temp\bin\traffic4

Action: Open ☐ Clean

☐ Reset user settings

☐ Hide this dialog at startup

Figure 2: Open an Eiffel project

10. 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 from `https://bitbucket.org/nadiapolikarpova/traffic/downloads/traffic_3.zip`.

To hand in

There is nothing to hand in.

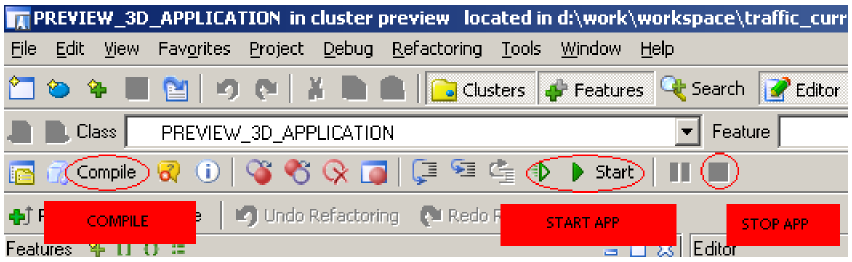

PREVIEW_3D_APPLICATION in cluster preview located in d:\work\workspace\traffic_curr
File Edit View Favorites Project Debug Refactoring Tools Window Help
Clusters
Features
Search
Editor
Class
PREVIEW_3D_APPLICATION
Feature
Compile
Start
COMPILE
Undo Refactoring
START APP
STOP APP
Features
Editor

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