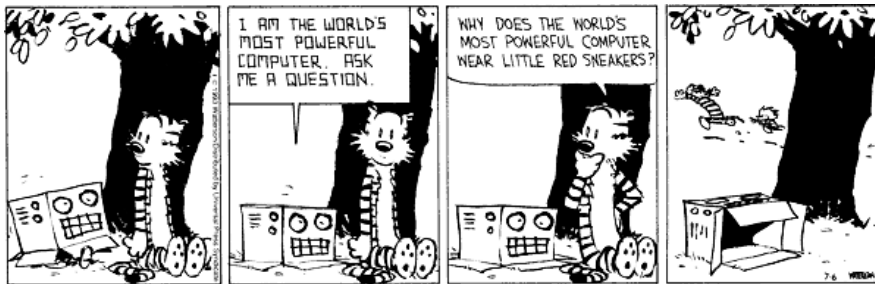


Assignment 2: Taking off

Hand-out: 31 October 2005

Due: 8 November 2005



Calvin and Hobbes© Bill Watterson

1 Get everything running

Goal

- Get EiffelStudio running.
- Turn on the logging mechanism.
- Install EiffelMedia.
- Compile Flathunt.

Todo

In this task you will install all the needed software for Introduction to Programming. If you work in a computer lab of ETH you will **not** need to install EiffelStudio and EiffelMedia, and you will **not** need to turn on the logging mechanism of EiffelStudio. Note that EiffelMedia is not supported under Mac and won't work.

1.1 EiffelStudio installation

Skip this point if you intend to work in an ETH Computer lab.

Download and install EiffelStudio 5.6 from <http://www.eiffel.com/downloads>. If you have any questions regarding this, ask your assistant.

1.2 Enable logging

Skip this point if you intend to work exclusively in an ETH Computer lab.

Enable the option in EiffelStudio that creates the ‘backup’ directory: this directory contains the learning logs. The following steps need to be followed once. You will **not** have to repeat them again if you have already enabled the creation of learning logs!

1. Locate the Eiffel home directory (where EiffelStudio is installed - usually it is Eiffel56).
2. From this directory, you will find the file `general.cfg` in either: `eifinit\studio` or `eifinit\studio\spec\windows` (or `eifinit\studio\spec\xxx` if EiffelStudio is not running on Windows - where `xxx` is the name of the operating system). (e.g.: If EiffelStudio 5.6 was installed on the C: drive in the Windows environment, the complete path to `general.cfg` would be `C:\Eiffel56\eifinit\studio` or `C:\Eiffel56\eifinit\studio\spec\windows`)
3. Open `general.cfg` with any text editor.
4. The `automatic_backup` option must be set to ‘true’(It is set to ‘false’ by default). This option is found at the bottom on the `general.cfg` file. Simply replace ‘false’ by ‘true’ on the line `automatic_backup: false`.
5. Save `general.cfg` keeping the same extension (`cfg`).
6. If EiffelStudio was open, close EiffelStudio and restart it.

1.3 Install EiffelMedia

Skip this point if you intend to work in an ETH Computer lab.

Download and install EiffelMedia from <http://eiffelmedia.origo.ethz.ch/download.html>. If you need any help ask your assistant.

1.4 Open and compile Flathunt

1. Download the Traffic software from <http://se.inf.ethz.ch/traffic>. Make sure to choose the newest release.
2. Unzip `traffic.zip` to a directory of your choice.
3. Start EiffelStudio. In most cases EiffelStudio will show the dialog of figure 1. If this does not happen automatically, go to **File** ▷ **New project** and it should appear.
4. Select the option **Open existing Ace (control file)** and click **Next**.
5. In the dialog that is appearing now, browse to where you unzipped the Traffic software and go to the directory under `example\flat_hunt`. In this directory you will find two files with the ending `.ace` for windows:

Depending on your C compiler choose `flathunt_windows_msc.ace` for Microsoft Compiler or `flathunt_windows_bcb.ace` for the Borland compiler. Click on open and a new dialog should appear (as in figure 2). Make sure that the option **Compile the generated project** is checked and click OK.

6. Now it is time to get a coffee and let Flathunt compile.

To hand in

There is nothing to hand in.

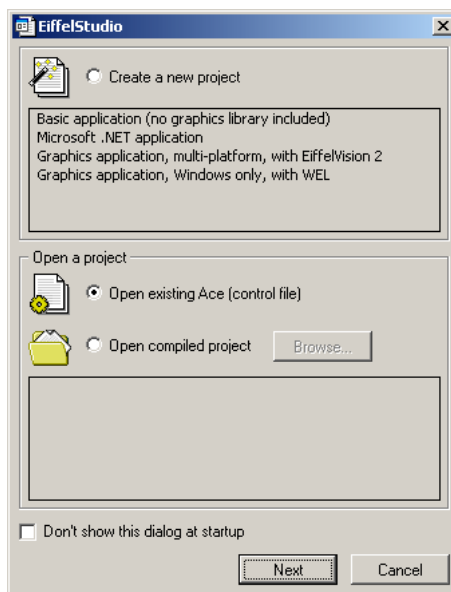


Figure 1: Eiffel Start-up Dialog

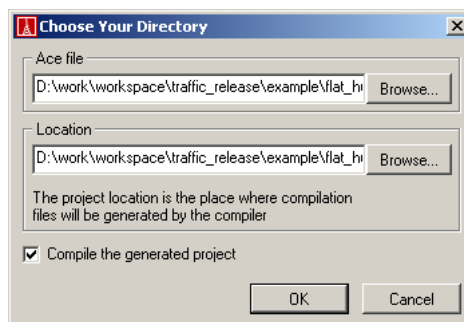


Figure 2: Just before compilation

2 Your first program

Goal

Write your first feature calls.

Todo

1. Open the class *START* in the Flathunt project that you compiled in task 1.4. Without changing anything, run the program by clicking on either the **Debug** button, or the button to its left (see figure 3). Start the game and play a bit with it, then close it.

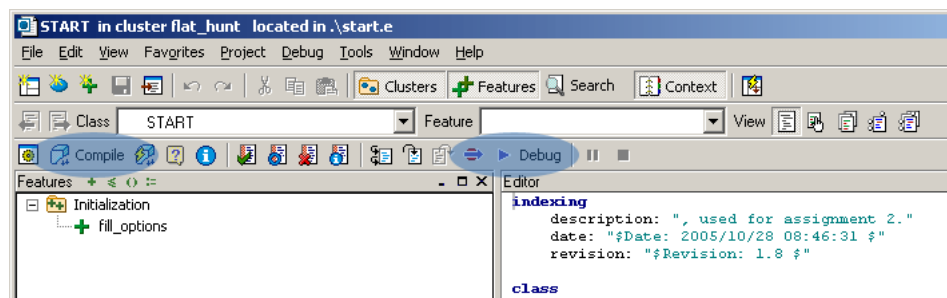


Figure 3: Just before compilation

2. In the feature *fill_options*, between the **do** and the **end**, fill in the following text:

```
1 option_panel.add_option_menu (options_list_game_mode,
option_title_game_mode)
```
3. Compile (see figure 3) the project again and launch it. Now you should see the option to choose between the different game modes. The first argument of the feature *add_option_menu* is a list of options that you can choose from. The second arguments states the title of the option menu. Close the game again.
4. There are three more option menus that you should add now:
 - (a) The option menu for choosing the number of players. The query for the list of options is *options_list_number_of_flathunters*; the query for the title for this option menu is *option_title_number_of_flathunters*.
 - (b) The option menu for choosing the map size. The query for the list of options is *options_list_map_size*; the query for the title for this option menu is *option_title_map_size*.
 - (c) The option menu for choosing the image style that is used for the flathunters and the estate agent. The query for the list of options is *options_list_characters*; the query for the title for this option menu is *option_title_characters*.
5. Change the order of the lines of code that you added. What happens?

To hand in

Submit the logs that were created by EiffelStudio. You will need to follow these steps every week once you are ready to submit your solution(s).

1. Locate the log files: They are found in the EIFGEN directory in a subdirectory called BACKUP. Supposing you saved a project called `flat_hunt` in the following directory `C:\MyProjects\flat_hunt`, you will find the logs in `C:\MyProjects\flat_hunt\EIFGEN\BACKUP`.
2. In the BACKUP directory, you will find subdirectories called COMP001, COMP002, etc. Rename the BACKUP directory as your username.
3. Zip the renamed BACKUP directory.
4. Upload the renamed and zipped BACKUP directory to <http://www.dcs.bbk.ac.uk/~gngch01/fileupload.html>. (TIP: Bookmark this website so that you don't have to remember it every time you want to upload the file.)

There is nothing to hand in to your assistant.

3 Commands vs. Queries

Goal

Understand the difference between commands and queries.

Todo

Look at the features listed below. They can be found in class *TRAFFIC_MAP*. For each of the features, try to figure out whether it is a command or a query. The class *TRAFFIC_MAP* represents the transportation system in a city with all the lines, the places and so on.

- A feature *has_place*, used under the form *a_map.has_place(a_place_name)*.
- A feature *add_place*, used under the form *a_map.add_place(a_place)*.
- A feature *places*, used under the form *a_map.places*.
- A feature *name*, used under the form *a_map.name*.
- A feature *set_name*, used under the form *a_map.set_name(a_map_name)*
- A feature *remove_line_section*, used under the form *a_map.remove_line_section(a_line_section)*.

To hand in

Submit your solution to your assistant.