Flat Hunt Redesign
and
ESDL Extensions

Ursina Caluori
ucaluori@student.ethz.ch

August 28, 2005

Contents

1 Introduction

For two years now, the department of Computer Science at ETH Zurich has been applying a new teaching approach called "Inverted Curriculum" [1] [2] [6] to the *Introduction to Programming* course. This technique is also referred to as a "outside-in" strategy of teaching. Rather than beginning with writing the infamous "Hello World" program, the students work from the start with a big software framework, which they gradually get to know better. At first, the exercises consist of merely calling a few library functions. Later on, control structures, Design by Contract, genericity and other advanced topics are introduced, some of which also using the framework.
The framework consists of a game-like application called *Flat Hunt* and several libraries upon which the game is built. Namely *TRAFFIC* [4], *EiffelBase* and *EiffelVision2*.

Due to the complexity of *EiffelVision2* and the sheer unlimited multimedia capabilities of *EiffelMedia* (which is the new name of *ESDL* [7] [8] [5]), it was decided (by people of a higher echelon, might I add) that *Flat Hunt* should be redesigned to use *EiffelMedia* for visualization rather than *EiffelVision2*. And exactly that also happens to be the very goal of this semester thesis. Well, actually almost exactly to be exact - for the goal also includes *ESDL* extensions (see section 3), suggestions for student assignments (see section 4) and last but not least also the redesign from a graphical point of view (meaning not only the application code will get a refresh but also the "look and feel" of the game).

Since *Flat Hunt* is a teaching application, a great responsibilty lay upon my shoulders in producing a very clean design and code of impeccable quality (which is generally a utopia in software engineering, if you ask me - hence I don't claim to have completely accomplished that, but nonetheless endeavored to achieve).

2 Design Decisions

There were some tricky design decisions that had to be dealt with..

2.1 Singleton Scenes vs. last_scene

2.2 Menu Design

2.3 Main Controller Necessary?

2.4 PLAYER and PLAYER_DISPLAYER

2.5 Necessity of FLAT_HUNT_SCENE

3 ESDL Extensions

Since I initially wanted to do my semester thesis on *ESDL*, but then due to several circumstances ended up doing it on *Flat Hunt*, I wanted to keep the option open to simultaneously develop *ESDL* if needed for my project. So I named my thesis "Flat Hunt Redesign and ESDL Extensions".
As I made progress with my work it turned out that *ESDL* already sufficed for *Flat Hunt* in virtually every aspect. And then *ESDL* was transformed into a new project called *EiffelMedia* (*EM*) which would be aggressively developed over summer by a number of people.
So basically what I did with *ESDL* was use it, rather than extend it (except for fixing minor bugs which I ran across, but that does not count as an extension in my opinion).
In the end, I was relieved that *ESDL*, or now *EM*, already had such good support for everything I needed, because redesigning *Flat Hunt* proved to be much more elaborate than I had imagined.

4 Assigment Suggestions

5 Thanks

Thanks to...

Michela Pedroni for having been a great assistant who gave me a lot freedom in all aspects and was very supportive.

My Predecessors for their work.

Till G. Bay for ESDL support and the like.

My Friends, Family and Cats for motivating me and creating a pleasant working atmosphere.

References

[1] Bertrand Meyer. *The Outside-In Method of Teaching Introductory Programming*. 2003.
`http://www.inf.ethz.ch/~meyer/publications/teaching/teaching-psi.pdf`

[2] Michela Pedroni. *Teaching Introductory Programming with the Inverted Curriculum Approach*. ETH Zurich, 2003.
`http://se.inf.ethz.ch/projects/michela_pedroni`

[3] Roger Küng. *Touch User Guide*. ETH Zurich, 2005.
`http://se.inf.ethz.ch/projects/roger_kueng`

[4] Sibylle Aregger. *Redesign of the TRAFFIC library*. ETH Zurich, 2005.
`http://se.inf.ethz.ch/projects/sibylle_aregger`

[5] Rolf Bruderer. *Object-Oriented Framework for Teaching Introductory Programming*. ETH Zurich, 2005.
`http://se.inf.ethz.ch/projects/rolf_bruderer`

[6] Marcel Kessler. *Exercise Design for Introductory Programming. "Learn-by-doing" basic OO-concepts using Inverted Curriculum*. ETH Zurich, 2004.
`http://se.inf.ethz.ch/projects/marcel_kessler`

[7] Benno Baumgartner. *ESDL - Eiffel Simple Direct Media Library*. ETH Zurich, 2004.
`http://se.inf.ethz.ch/projects/benno_baumgartner`

[8] Till G. Bay. *Eiffel SDL Multimedia Library (ESDL)*. ETH Zurich, 2003.
`http://se.inf.ethz.ch/projects/till_bay`

A Flat Hunt User Guide

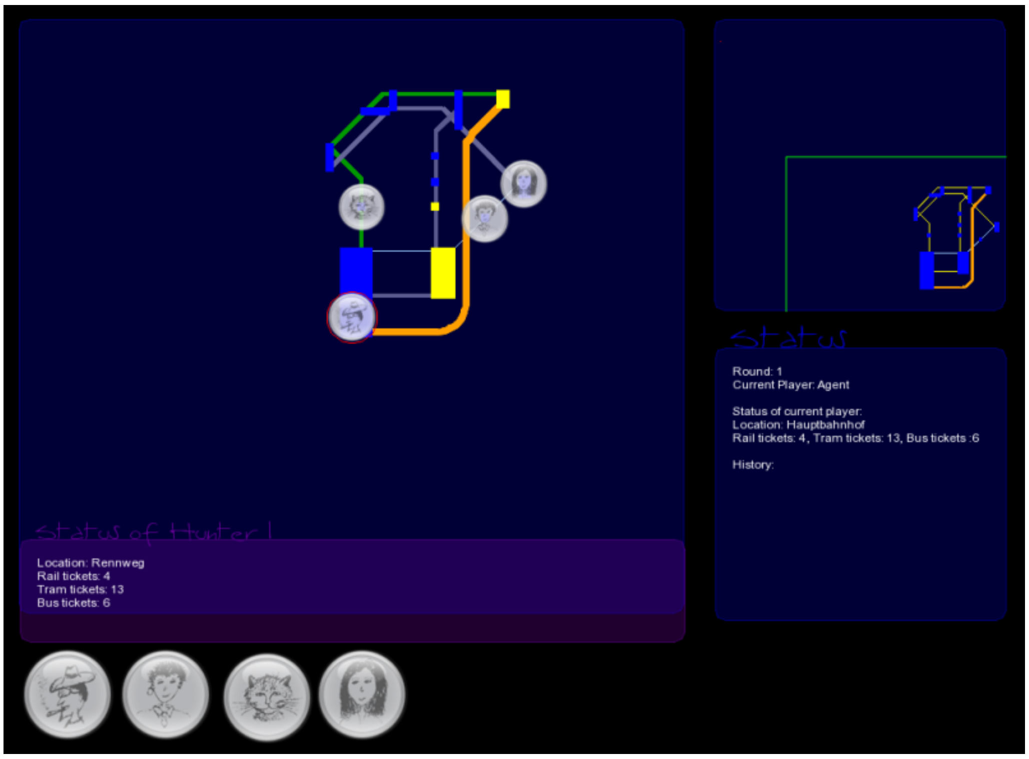
Flat Hunt is an application that is used to teach you programming, along with another application named *Touch* [1]. *Flat Hunt* is a "Scotland Yard"-like game that will mainly appear in the assignments for the Introduction to Programming course. It is based on *TRAFFIC* [2] for modeling the city where the game takes place and on *EiffelMedia* (formerly known as *ESDL* [3]) for visualization.

This document describes how to use *Flat Hunt*.

A.1 Introduction

Welcome to *Flat Hunt*!

Flat Hunt is a simple adaptation of the well-known board game “Scotland Yard” (see Figure 1). Instead of some agents hunting Mr. X all around London, it is about a group of students starting off at ETH Zurich. To make their student life a bit more pleasant, they are desperately trying to find a flat in this little big city. But to get a flat, they must first meet the estate agent, who is running all around Zurich showing his flats to other people…



```
Location: Rennweg
Rail tickets: 4
Tram tickets: 13
Bus tickets: 6
```

```
Round: 1
Current Player: Agent

Status of current player:
Location: Hauptbahnhof
Rail tickets: 4, Tram tickets: 13, Bus tickets :6

History:
```

Figure 1: Screenshot of *Flat Hunt* in action

A.2 The Story

As the title suggests (and the introduction mentions), it is all about finding a flat in Zurich…

However, this is not so easy… There is this guy, the estate agent, who is renting flats. The problem is that he is always busy showing flats to other customers, and even in his office they don't really always know where exactly he is. The only thing they know is what kind of transport he is moving around with. This is because the estate agent is taking part in a new VBZ-project called "Customer tracking".

In collaboration with ETH, they equipped some volunteers with transponders. These transponders gather information like current position and type of transport, and send it in real-time to the office. However, for privacy reasons, only the type of transport can be accessed all the time.

Once in a while, the estate agent (Figure 2) calls his office to tell the secretary which flat he is currently visiting. So sometimes, the people there in the office can tell "you" where to look for him…"You" meaning yourself and your friends (Figure 3), the guys you want to share the flat with…

Figure 2: Estate agent

Figure 3: You and friends

A.3 Gameplay

Playing Flat Hunt is not very difficult, especially for those that know the game "Scotland Yard"…

A.3.1 General Rules

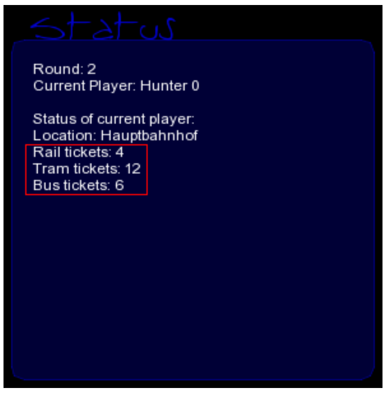
The game lasts for at most 23 rounds. In these 23 rounds, the flat hunters try to find the estate agent, while he tries to avoid them (this is because he would rather rent the flats to elderly couples, since presumably they make fewer parties in the middle of the night…).

In each round, every player can make one move on the public transport system. The estate agent is the first, then it's the hunters turn. One move is either

- one or two stops by tram (colored lines),
- one stop by train (thick orange lines),
- or one stop by bus (thin light blue lines).

A move with a certain transport can only be made if one has still enough tickets (see Figure 4), if there is a connection (obviously), and if there is no other player at that destination (and in the case of tram lines, if there is no hunter in between).

Attention: If you are at a bus-only stop, and you run out of bus tickets, you will get stuck there forever, so be careful…



Round: 2
Current Player: Hunter 0

Status of current player:
Location: Hauptbahnhof
Rail tickets: 4
Tram tickets: 12
Bus tickets: 6

Figure 4: Ticket status

The possible places you can move to are colored yellow (see Figure 5). To make a move, just click on one of those highlighted places. The red circle centers on the player whose turn it is, and in the status box at the right, the game status and information about the current player get displayed. If you want to know the status of another player just click on his picture at the bottom. Click again to close the just opened status box.

The game is over when

a) the hunters could not find the estate agent within 23 rounds,

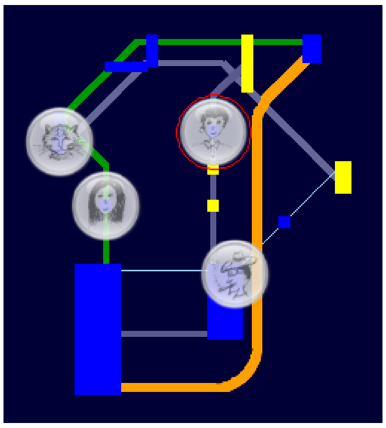


Figure 5: Highlighted places

b) one flat hunter moves onto the place where the agent currently is,

c) or the hunters encircle the estate agent so that he cannot move anymore.

In case a), the winner is the estate agent (he does not have to rent his flat to students), whereas in b) and c) it is the hunters that win, as they get to meet the estate agent on time and thus manage to find a flat.

A.3.2 Game Modes

There are four modes to play *Flat Hunt*: *Hunt*, *Escape*, *Versus* and *Demo*. Depending on the mode, zero (*Versus*), one (*Hunt/Escape*) or two (*Demo*) parts are taken over by the computer.

Hunt This is probably the most typical situation; the player tries to find the agent, which is played by the computer. Thus, the player only knows about every fifth move where the agent just was... The agent shows himself only in rounds number 1, 3, 8, 13, 18, and 23. In these rounds, the exact route of the agent is displayed under *History* in the status box at the bottom right corner and in the estate agent's own status box if opened. In all other rounds only the types of the transportation means he has taken so far are listed there.

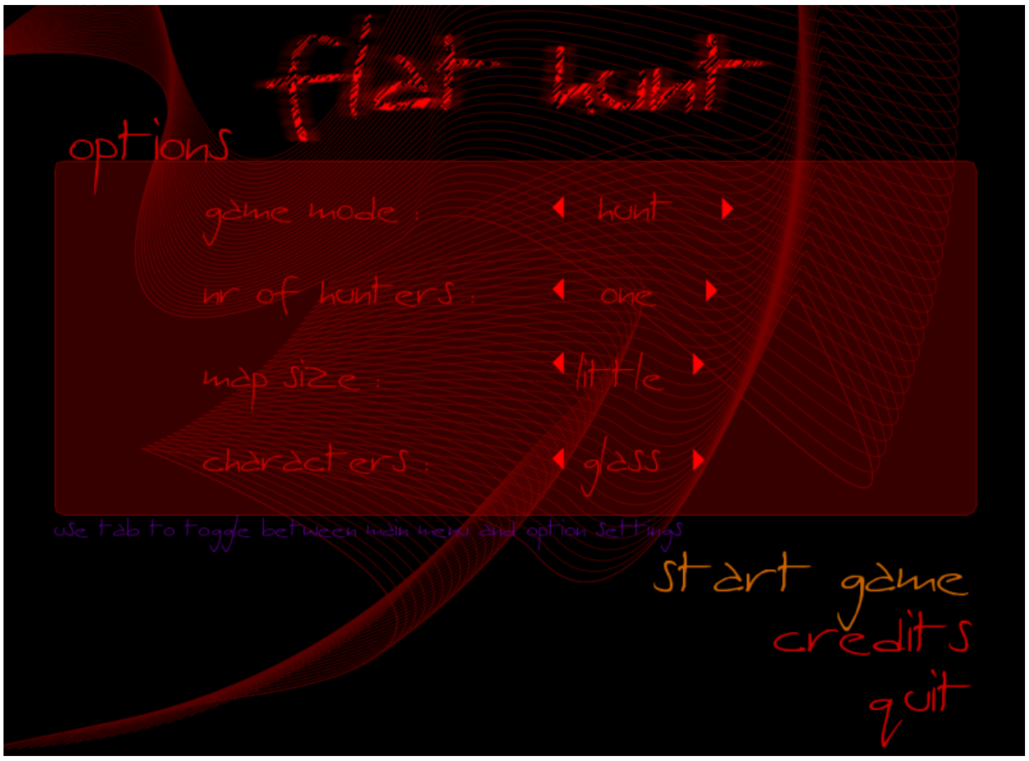
Escape This is the exact opposite of *Hunt* mode: The agent is played by you, and the hunters are played by the computer. The hunters always move as close in your direction as possible, as they somehow manage to decode your transponder signal, and thus always know your precise location (so much for privacy...). You just have to try to avoid them as long as possible...

Versus This is the multiplayer mode. One of the players is the agent; the other plays all the hunters. While the player of the agent is making a move, the player of the hunters is supposed to look away...

Demo This mode is more or less the opposite of the buzzword "interactive", but is about as entertaining as watching fish in an aquarium. The computer is playing against himself, trying to catch the agent as fast as possible.

A.3.3 Other

When you run *Flat Hunt*, the first you'll see is a menu (see Figure 6). You can either let the default options in place and just select *start game* or you can adjust the settings to your needs. *Game mode* is explained in subsubsection A.3.2, *number of hunters* and *map size* should be self-explanatory and *characters* specifies which pictures to use for the players. To toggle between the settings menu and the normal menu press *tab*.



game mode : hunt
nr of hunters : one
map size : little
characters : glass
use tab to toggle between main menu and option settings

Figure 6: Screenshot of the start menu

During the game when you press *p*, the pause menu is shown. *Continue* makes the pause menu disappear and lets you resume the game, *new game* takes you to the start menu and *quit* quits the application.

The game over menu is similar to the pause menu, only there is no *continue* in this one.

A.4 Special Features

A.4.1 Map Control

Map control is fairly simple: you got two maps in a game scene, one of which only is a smaller version of the other one. The big map is on the left and that's where the action takes place, the little map on the right is meerely a navigation tool. To control the big map, use your mouse as follows:

left click: only has an impact if clicked on a highlighted place

richt click + move mouse: moves map in the direction of your mouse movement

middle click + move mouse up: zoom in

middle click + move mouse down: zoom out

When you *left click* + *move mouse* in the little map, a red rectangle is drawn between the point where your left mouse button is pressed down and the point when its released. As soon as you release the mouse button, the map segment that is inside this rectangle gets displayed on the big map.

A.4.2 Music Player

Flat Hunt comes with an integrated music player and some default background music. Since not everyone likes the same sound, there is also the possibility to play your own.
Just put your `.ogg`-files in the directory `${FLAT_HUNT}/resources/sound` before you start the Flat Hunt application. Flat Hunt will then automatically load all the `.ogg`-files from this directory and play them in alphabetical order (unless you enable shuffle, obviously). Music player control: see subsubsection A.4.3.

A.4.3 Keyboard Shortcuts

During the game, the following shortcuts are available:

p: pause the game and show pause menu

s: music player toggle shuffle

v: music player decrease volume

shift + v: music player increase volume (at startup the volume is already at maximum)

page up: music player next song

page down: music player previous song

A.5 Legal Stuff and Thanks

This document is based upon its prior version, which was written by Michela Pedroni and Marcel Kessler (thanks!). All graphics for the game were designed by me and Photoshop.

Thanks to Michela Pedroni for her assistance, all my predecessors for their work, Till G. Bay (and others) for the *EiffelMedia* (formerly *ESDL*) Library and Bertrand Meyer for the *Eiffel* language.

References

[1] Roger Küng. *Touch User Guide*. ETH Zurich, 2005.
`http://se.inf.ethz.ch/projects/roger_kueng`

[2] Sibylle Aregger. *Redesign of the TRAFFIC library*. ETH Zurich, 2005.
`http://se.inf.ethz.ch/projects/sibylle_aregger`

[3] Till G. Bay. *Eiffel SDL Multimedia Library (ESDL)*. ETH Zurich, 2003.
`http://se.inf.ethz.ch/projects/till_bay`

B Flat Hunt Developer Guide

TRAFFIC [2], *Touch* [1] and *Flat Hunt* is software that hopefully makes learning to program more fun and more interesting for you. *TRAFFIC* is a library that supports the reading and display of public transportation systems. A library is a piece of software whose functionality can be used by other software. *Flat Hunt* is an application that uses the *TRAFFIC* library to model a city map. For the visualization the *EiffelMedia* Library (formerly known as *ESDL* [6][5]) is used. It is a strategy game, similar to Scotland Yard, but with a different background story (for more information about the story and gameplay of *Flat Hunt*, read the Flat Hunt User Guide). *Touch* is also an application that uses the *TRAFFIC* library. For more information on *TRAFFIC* and *Touch*: see [2] and [1] or visit this website.

This document describes how *Flat Hunt* is built, what classes are important and highlights some of their features.

B.1 Getting Started

What you need for running Flat Hunt...

B.1.1 Requirements

B.1.2 Installation

B.2 Design

B.2.1 Overview

When opening *Flat Hunt* in EiffelStudio, the cluster view in the bottom left corner of EiffelStudio shows many clusters. For you only the top-level clusters *Traffic* and *Flat_hunt* are important.

To remove complexity, *Flat Hunt* is structured in three clusters (see Figure 1): *Model*, *View* and *Controller*. In each cluster, there are several classes, and sometimes there are subclusters.

B.2.2 Controller cluster

Cluster Controller is the fundamental cluster in *Flat Hunt*. Here are the classes that "control" the actions. They make sure that the displayer classes in cluster View display the proper information, which they get from the Model classes. For example, feature prepare in class `GAME` controls the display update by calling `current_player.displayer.display_after_move`.

B.2.3 Model cluster

In the cluster Model, there are two important parent classes: Class `PLAYER` and class `BRAIN`. `PLAYER` is the parent of `FLAT_HUNTER` and `ESTATE_AGENT`, and `BRAIN` is the parent of `HUMAN`, `FLAT_HUNTER_BOT` and `ESTATE_AGENT_BOT`. These Model classes describe the internal representation of "real world" objects. Here is a description of some of these classes.

B.2.4 View cluster

This clusters job is to make sure that the user sees what is going on. It includes all scenes and menus, as well as displayers for the game players and status information.

B.2.5 Other

B.3 The States of the Game

B.3.1 Overview

Every game has at least two states: playing and game over. *Flat Hunt* has six states in total; three playing states and three game over states (see Figure x). These game states are defined in class `GAME_CONSTANTS`:

```
   Agent_stuck, Agent_stuck, Agent_caught, Agent_escapes,
Prepare_state, Play_state, Move_state:  INTEGER is unique
```

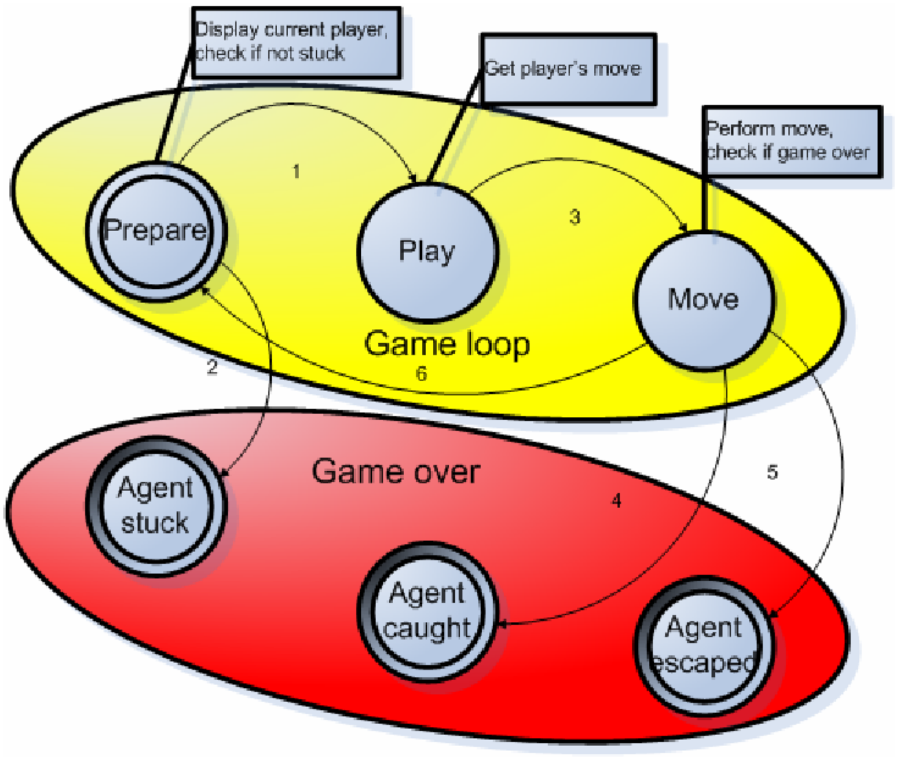


Figure 7: Game states and loop

B.3.2 Game Loop

For each player in each round in *Flat Hunt*, the game goes through the following states: `Prepare`, `Play` and `Move`. In addition, there are three game over states: `Agent_stuck`, `Agent_caught` and `Agent_escaped`.

Prepare If the game is in this state, the current player gets a red circle and the possible moves are calculated and displayed. If the current player is the estate agent, and there are no possible moves, the agent is stuck and thus the game is over (state `Agent_stuck`). If that is not the case, the game goes in state `Play`.

Play In this state, if the current player is played by a human, the game waits until the human player clicks on one of the places that are highlighted. If the player is controlled by an artificial intelligence, then the best of the possible moves is calculated. The game then goes in state `Move`.

Move In this state, the move selected in state `Play` is performed. After the move, the game checks if the player hits the place of the estate agent. If that is the case, the game goes into state `Agent_caught`. If the agent did not get caught, and the round number is greater than 23, then the estate agent is the winner and the game goes into state `Agent_escaped`. If none of the above is the case, then it's the next player's turn and the game loop starts again in state `Prepare`.

B.4 Guided "Walk-Through"

What happens when you start Flat Hunt*? In this last chapter we will go step-by-step through a typical* Flat Hunt *game. However, because there are lots of details involved, we concentrate on the more important steps...*

1. start game blabla
2. do something
3. end game blabla

B.5 Legal Stuff and Thanks

This document is based upon its prior version, which was written by Michela Pedroni and Marcel Kessler (thanks!). All graphics for the game were designed by me and Photoshop. The code of *Flat Hunt* is based on its prior version [6], which is mainly the work of Marcel Kessler. Major parts had to be rewritten by me though.

Thanks to Michela Pedroni for her assistance, all my predecessors for their work, Till G. Bay (and others) for the *EiffelMedia* Library (formerly *ESDL* [6][5]) and Bertrand Meyer for the *Eiffel* language.

References

[1] Roger Küng. *Touch User Guide*. ETH Zurich, 2005.
http://se.inf.ethz.ch/projects/roger_kueng

[2] Sibylle Aregger. *Redesign of the TRAFFIC library*. ETH Zurich, 2005.
http://se.inf.ethz.ch/projects/sibylle_aregger

[3] Rolf Bruderer. *Object-Oriented Framework for Teaching Introductory Programming*. ETH Zurich, 2005.
http://se.inf.ethz.ch/projects/rolf_bruderer

[4] Marcel Kessler. *Exercise Design for Introductory Programming. "Learn-by-doing" basic OO-concepts using Inverted Curriculum*. ETH Zurich, 2004.
http://se.inf.ethz.ch/projects/marcel_kessler

[5] Benno Baumgartner. *ESDL - Eiffel Simple Direct Media Library*. ETH Zurich, 2004.
http://se.inf.ethz.ch/projects/benno_baumgartner

[6] Till G. Bay. *Eiffel SDL Multimedia Library (ESDL)*. ETH Zurich, 2003.
http://se.inf.ethz.ch/projects/till_bay