Visualize and Open Up

Michela Pedroni and Till G. Bay

Chair of Software Engineering, ETH Zurich
8092 Zurich, Switzerland
{pedronim | bay}@inf.ethz.ch

Abstract. Motivating students of the Nintendo generation for Computer Science can only be achieved by providing them with an exiting and fresh CS1 course. We report on our experience of redesigning the introductory programming course at ETH Zurich and show how we combine state-of-the-art visualizations with open project assignments to enliven students enthusiasm for our field. We describe our setup and the libraries involved, show example applications that were built in our course, and present the data gathered in the evaluation of the open assignment.

1 Introduction

Today's students are used to graphical programs; the levels are set by the computer games they play in their spare time. Graphical and multimedia applications are what they want to produce. Together with others [1–3], we believe that to "make CS courses fun" [4] and motivate students by using state-of-the-art graphics/multimedia libraries together with freedom for creativity is critical to their success in learning.

Over the past three years, we have significantly changed the way we teach programming to our first semester CS students by taking a new approach, called the Inverted Curriculum [5]. The Inverted Curriculum is an objects-first, component-based approach relying on a large software framework with a strong visual aspect. Using this approach, students start out as consumers of library components by using their abstract interfaces, before they progressively discover the implementations. Until the end of the course, they are capable of producing similar software elements themselves. This results in a topic introduction that is outside-in, starting with the notion of object, method call, and class interface, while the internals, such as control structures, local variables, and assignment are covered later in the course. The Inverted Curriculum allows students to produce interactive graphical applications right from the start, taking advantage of the power of provided libraries.

To complement the course redesign we introduced a project assignment on which students work in small groups over a period of 4-5 weeks

toward the end of the semester. While in the first iteration of the course the project assignment specified in detail what kind of application the students should produce, we completely opened up the project in the subsequent two iterations.

The idea of open project assignments is not novel. Sindre, Line, and Valvag [6] report on their experience with letting students freely choose what kind of game they produce. We believe that we can and should go even one step further. We refrain from narrowing down the domain of projects and therefore put no limitations to students' creativity, thus motivating exceptional results.

We present the implementation of the open assignment and accompany it with the results and feedback we gained. Section 2 describes the general setting and the libraries that we use during the course, while section 3 explains the implementation of the open project assignment. Section 4 expands on the results and feedback that we got by students, and section 5 presents our conclusions.

2 Setting

2.1 Course setup

Since winter 2003 the Introduction to Programming course for first semester Computer Science majors implements the ideas of the Inverted Curriculum. The number of students that participated in our courses so far is approximately 600 (250 in 2003/2004, 180 in 2004/2005, and 170 in 2005/2006).

Introduction to Programming is the only Computer Science course for CS masters in the first semester. In the second semester, a course called Data Structures and Algorithms is held as a follow-up of Introduction to Programming. The other courses of the first semester are mostly math courses, laying the basic knowledge for advanced studies in CS.

Our course consists of seven weekly lessons, where four are plenary lectures held by the professor and three are held in groups of up to 25 students by graduate and doctoral student tutors.

The semester stretches over 14 weeks with Christmas break after week 9. We divide the semester into two parts: the first part from week 1 to week 8 where students are handed out weekly assignments that they are supposed to solve alone. In week 9 (before Christmas break) we hand out the description for the project assignment which they solve in small groups until the end of the semester. Furthermore, we provide up to three sit-in assignments spread over the semester to help students assess their

current status and get a feeling on how they are performing compared to their classmates. All the handed in weekly assignments, mock exams and the project are corrected (but not graded) by the tutors.

Instead of grading assignments in first-year courses, ETH has the policy that students need to get a certificate to be admitted to the exam. The exam determines whether students are allowed to move into their second year of study. It takes place after the second semester toward the end of the semester break. To get the admittance certificate, we require them to do about 70-80% of the weekly assignments and mock exams (not necessarily correctly, but showing a clear effort), and to do the project assignment. While having no grading during the semester is quite unusual, it allows even newcomers to programming to take a long-term approach to learning and prevents them to get obsessed about their grades during the semester.

2.2 Technical foundation

The technical building grounds of the course are the libraries we use to develop the multimedia applications. Our approach focuses on the use of libraries early in the Computer Science education. By using libraries that were written by others, students learn to read code. We believe this to be one of the most important skills an engineer of our field needs to provide. A consequence of this activity is that programming patterns are studied in practice rather than in the abstract. By exploring the libraries, students see how others have solved a problem.

The libraries we use provide a vast API which makes them suitable for the open assignment that we give to the students. Students can choose to use only the simple mini frameworks provided by the libraries, or they can go for their full power. By using the mini frameworks, complexity remains hidden and the students can concentrate on their own application design (e.g. they can use a ready made keyboard handling framework). But if they want to do more, they can use the finer grained and more complicated API that is also provided (e.g. they can devise their own keyboard handling facility).

The two libraries we heavily use in our courses are described in the following two subsections. They are both partly developed and maintained by our students. Later in their studies, the students need to write a number of semester theses and a final master thesis. They can choose at which institute or chair they want to write these workings. We are very happy that many of them choose to contribute to the libraries they used before

and thus help us to improve the quality as well as the capabilities of our foundation. Of course many of the students are motivated to contribute to the libraries because of the bugs they have encountered when using them.

EiffelMedia Recently Carter [7] has shown that the number one reason for boys to study Computer Science is their interest in computer games whereas for girls it was the possibility to use computers in other fields - in both cases one can argue that the visualization capabilities are of great importance. *EiffelMedia* [8] is the rendering library we use in our applications. By using this library, we can ensure that our visualization capabilities are state-of-the-art and impress and motivate the students. As mentioned above, the library is maintained together with the students and is open source. *EiffelMedia* has a rich set of frameworks that allow building multimedia applications such as games like Antworld shown in figure 1. The features of the library include 2D graphics, sound support, video decoding, 3D graphics, networking support, a widget toolkit and many more.

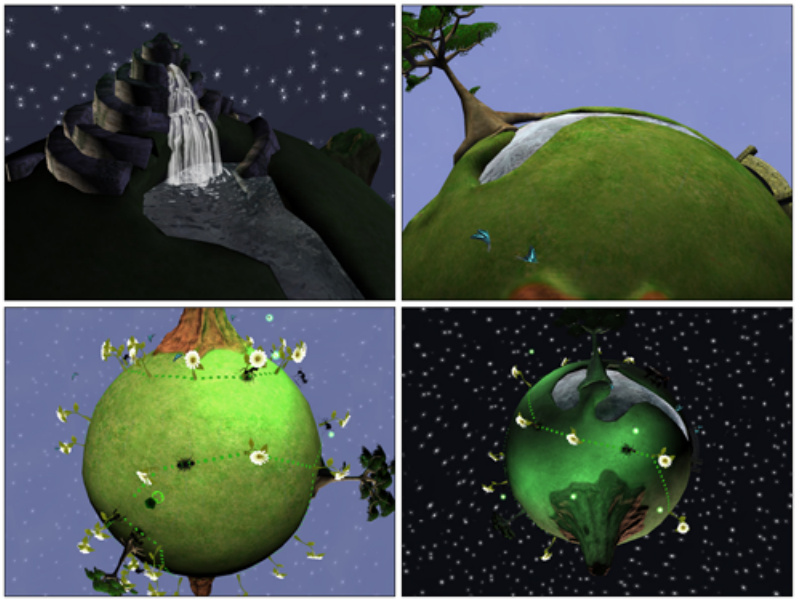

Fig. 1. Day and a night in *Antworld*.

Traffic The second library our course relies on is *Traffic* [9]. *Traffic* models the public transportation system of a city consisting of transport lines (e.g. metro lines, bus lines, light rail) and places (metro stations, landmarks). It contains city modeling classes, visualization facilities for the display of city maps and additional supporting classes for building applications. *Traffic* comes with several applications: One of them -*Flathunt*- is a strategy round-based game, others are either example applications to show certain features of the library, or they accompany the textbook *Touch of Class* [10], currently under development. Both - *Traffic* and its applications - have been specially developed for the use in classroom with the Inverted Curriculum. *Traffic* uses *EiffelMedia* to visualize the city model.

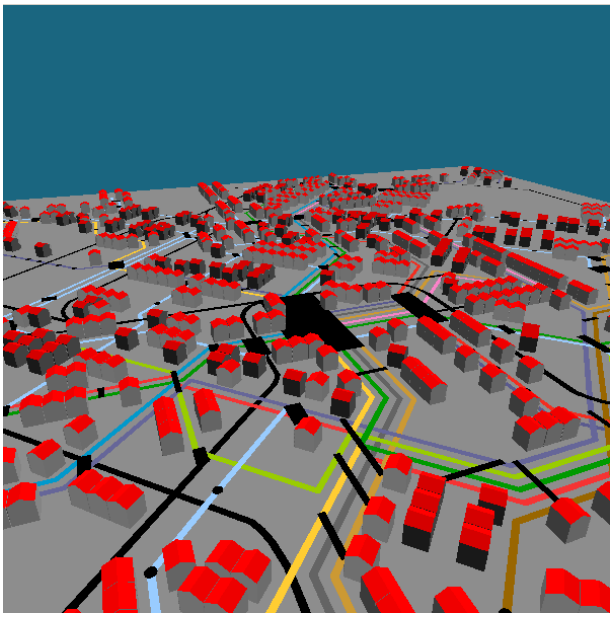

Fig. 2. 3D model of a city in *Traffic*.

Both of the libraries are particularly well suited for using in an open project because they offer a wealth of functionality and highlight visualization and multimedia. With this foundation, we feel comfortable to give students the freedom of an open project assignment.

3 Procedure

For the open project assignment students were required to choose a partner of equal strength, so that they could design their project idea to be challenging for both of them and fitting their programming skills. In previous years, the number of students per group was three, but we decreased it to prevent organizational problems such as code synchronization and task distribution among students.

To help students organize their project, we provided them with a wiki, where each group as a first task generated a new page for their project and a description of what their application should do. Using this platform, they were able to discuss issues online, upload any of the intermediary results (such as the documentation and source code), and - if interested - they could browse other groups' projects at any point.

We guided students by officially giving them two options in deciding on the project idea: Option A was to implement an extension to *Flathunt*/*Traffic* or *EiffelMedia* with examples of possible projects of reasonable size. Since the students had already been using these libraries during the course, most of them chose this path. Option B was totally open, telling them to do whatever they wanted with the one restriction that their idea had to get approved by the tutor. As a result of the openness, almost every group was asking approval from their tutor to also get feedback on the feasibility of their project.

As a consequence of timing problems that some of the students had in previous years and feedback stating that we should emphasize the importance of the software design phase, we provided much stricter guidelines than the years before by identifying four milestones that students had to meet. The first milestone was due after a very short period of four days. For this milestone, they had to hand in a first project idea and a description on the wiki. The next milestone was planned for right after Christmas break where they had to give a more precise description of the requirements and the task distribution among the partners. The third milestone followed one week later, where they should provide a document describing the OO design of their system. The last milestone was two weeks later (in the last week of the semester) and they had to hand in the code and a short developer guide.

The project was complemented by having each group give a short presentation to their tutors and fellow students during the exercise session of the last week of the semester. In each of the exercise groups one of the projects was elected to be shown in a subsequent event called the *Object-*

oriental bazaar. The bazaar was the closing lecture of the semester and was open to the public. Students were asked to invite their friends and the department was encouraged to come and see what our first-semester students had done. In a first round, the elected projects were presented to the curious public, but the second half was devoted to all the projects where each group was present with a laptop showing and explaining their projects to interested students, tutors, professors or other guests. This happening was greatly appreciated by our students since they felt that we valued their efforts.

4 Results and student feedback

The open project assignment of this year resulted in over 70 applications [11]. About 50% of these applications were games written with *EiffelMedia* such as two sudoku solvers, some Battleship implementations, the traditional pong game, spaceshooters or jump'n'run games. Another 25% were either extensions to *Flathunt* or extensions to the *Traffic* library such as a multiplayer *Flathunt*, a timetable *Traffic* extension, or a modifications of the existing game *Flathunt*. The remaining 25% were applications or libraries of any kind, such as an InstantMessenger, a math parser, a browser, collection classes, or an RSS feed reader. The games usually came with 2d graphical user interfaces, but we also had five console applications and a few games with amazing 3D visuals, such as *Antworld* shown in figure 1.

As part of our course evaluation, we asked students whether they liked doing the project stretching over several weeks. The overall answers were positive, the average grade always ranged from 3.9 - 4.0 points out of 5 for all three iterations of the course. In the evaluations for 2004/2005 and 2005/2006 we also asked them whether they appreciated being able to freely choose the project task. Here the mean grades increased significantly from 3.6 (in 2004/2005) to 4.5 (in 2005/2006) out of 5. We believe that the increase is mostly due to the fact, that much work had been done on *EiffelMedia* in between iterations and that we guided students more, thus improving their timing.

5 Conclusion

Motivating students for our field is difficult and can only succeed if we provide courses that immediately show what great things can be achieved us-

ing computers and programming. In this report, we show that by combining state-of-the-art multimedia libraries with the freedom of open project assignments we can improve the students perception of their CS curriculum. We encourage open project assignments provided they are embedded in a well structured course and project framework that allows the students to validate their progress. We will continue to strive for more elaborate visualization techniques as we realize that we constantly have to keep up with the entertainment industry in order to meet the expectations of our customers, the students.

References

1. Guzdial, M., Soloway, E.: Teaching the nintendo generation to program. Commun. ACM **45** (2002)
2. Feldman, T.J., Zelenski, J.D.: The quest for excellence in designing cs1/cs2 assignments. In: SIGCSE '96: Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education, New York, NY, USA, ACM Press (1996) 319–323
3. Becker, K.: Teaching with games: the minesweeper and asteroids experience. J. Comput. Small Coll. **17** (2001) 23–33
4. Mahmoud, Q.H.: Revitalizing computing science education. Computer **38** (2005) 100–99
5. Pedroni, M., Meyer, B.: The inverted curriculum in practice. In: SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education, New York, NY, USA, ACM Press (2006)
6. Sindre, G., Line, S., Valvag, O.V.: Positive experiences with an open project assignment in an introductory programming course. In: ICSE '03: Proceedings of the 25th International Conference on Software Engineering, Washington, DC, USA, IEEE Computer Society (2003) 608–613
7. Carter, L.: Why students with an apparent aptitude for computer science don't choose to major in computer science. In: SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education, New York, NY, USA, ACM Press (2006) 27–31
8. Bay, T.G.: Eiffelmedia - the multimedia library for eiffel (2006) available online at: http://eiffelmedia.origo.ethz.ch/.
9. Pedroni, M.: Traffic (2006) available online at: http://se.inf.ethz.ch/traffic.
10. Meyer, B.: Touch of class (2006) available online at: http://se.ethz.ch/ meyer/down/touch/.
11. Bay, T.G.: Collection of games and appication built in our courses (2006) available online at: http://games.ethz.ch/.