How to learn introductory programming over Web

Arto Haataja, Jarkko Suhonen and Erkki Sutinen

Department of Computer Science, University of Joensuu, Finland

Jarkko.Suhonen@cs.joensuu.fi
P.O. Box 111, FIN 80101, Joensuu, Finland
http://cs.joensuu.fi

Keywords: *introductory programming, virtual university, high school, distance education*

Abstract: *A part of the Virtual University of Finland, the eastern Finland universities offer high school students an opportunity to take their first 15 university credits of Computer Science over the Web. At the University of Joensuu courses consist of three parts: general introduction to computers (three credits), introduction to Computer Science (five credits), and programming (seven credits). Instruction is almost entirely delivered over the Web. The students follow a schedule given at the web site, learn the related chapters from their textbooks, and return exercises by strict deadlines.*

The students live as far as hundred kilometers away from the university and their local high schools have not been able to hire qualified programming teachers. We solved the problem by organizing on-line teachers at the university to answer students' questions and assigning a tutor - with hardly any experience in programming - to encourage the students at the local school. To intensify the learning outcomes, we are planning to use three different learning tools in the web-based course environment: Excel, Jeliot, and BlueJ. Each of these can be used to visually understand a given program. In particular, the environments serve as virtual laboratories for real problems: the students can study their own programs.

At the University of Joensuu, the course started in August 2000. Out of the 80 enrolled students, more than 65 were active after the first three months. Altogether, the course will take 16 months. Students' activity and commitment to their studies indicate that the approach chosen to teach programming has proved efficient.

Virtual Computer Science studies

In Finland, the Ministry of Education is funding a three-year project to establish the Virtual University of Finland, during years 2001-2003. One of the particular goals in the project is to develop new methods for science education. The three universities in eastern Finland, University of Joensuu, University of Kuopio, and Lappeenranta University of Technology, work jointly in the virtual university project. One of the concrete objectives is to create a web-based learning environment in introductory Computer Science, intended for high schools students. From the research perspective of educational technology and Computer Science education, this task is particularly challenging. We have aimed at designing a solid model for building the environment.

The project is called "virtual certificate", indicating that high school students can take 15 credits of Computer Science studies in one and a half year via the Internet [6]. In the Finnish system, each credit equals 40 hours of studying; 160 credits are required for the Master's degree. Thus, after passing all the 15 credits of the program a student has the first year Computer Science studies completed. Moreover, is the student passes the program with grade 2/3, she is free to enter the university as a Computer Science major. Almost all teaching is done via Internet because the students come from all around the district of North Karelia. We have minimized the need for face-to-face teaching situations so that the students do not have to waste their time and money by travelling from their home to the university.

The most important part of our project is to give students good programming skills. Hence, our emphasis is to make the learning of programming as smooth as possible. One important part of the studies are ***tutor-teachers*** in the high schools and ***on-line tutors*** in the university. For example in the programming courses these tutors can give very valuable contribution to the learning process. Most of the courses, including programming courses, have a similar structure: students have material to work with, they do weekly assignments, and their learning outcomes are evaluated by an exam. However, we have tried to add flavors to studies by employing a few other teaching methods:

- ***Group activities***. A few courses will be implemented so that students work in small groups. We hope that this encourages students to work collaboratively and brings about interaction into the learning process.
- ***Learning by writing***. Students compose an essay on a certain subject. This is hopefully done in cooperation with local newspapers: the students will be reporting on what they learn.
- ***Netbus consulting***. An Internet-connected bus will visit each school for a few hours. A Question & Answer session will be organized, as well as individual consulting.
- ***Days at campus***. To make the students experience themselves as members of the academic community they are invited to the campus on four days during their studies.
- ***Camp at campus***. The theoretical course on algorithm will be arranged at campus, as a one-week intensive period.

Description of introductory programming courses

In general the basic structure of introductory programming course is similar to the other courses. We chose Java as our teaching language because the graphical java-applets provide more interesting and exciting ways to learn programming than standard text based programming languages. Basically, our method of teaching programming is very simple. The students' have got a learning material and weekly assignments to work with and the students own contribution to the learning process is essential. The student must have 1/3 of the assignments completed if she wants to take part in the exam. Furthermore by doing extra exercises she can achieve bonus points for the exam. The weekly assignments purpose is to make students under-

stand the basic and most important aspects of introductory programming. We think that learning by doing [4] is the most efficient way to understand deeply about the programming principles. In the following chapters we will describe more deeply about our methods for teaching the introductory programming.

Learning material

One major design principle is to join together the printed learning material, i.e., textbooks, and the material on the web. The ***textbook*** gives a detailed knowledge of the domain to the students while web materials support the learning process. ***Web-material*** provides overview of the programming domain and many programming examples to support the textbook material in a meaningful way. With the web-material the student can easily figure out the different concepts and their relationships in the course. We hope that in this way the web-material will provide a specific mental map of the programming concepts. Moreover the web-material divides the domain in logical parts so the students can concentrate on certain subject at once. Hence, all together one programming course consists of smaller units and in every unit there are exercises to work with. By splitting the course into smaller units we can control the learning process in a sophisticated way so that the knowledge flow does not get too stressful. We are using the WebCT program as our learning environment but all web-materials are designed to function independently.

When the web-material is providing all the crucial parts of the programming domain we use the course books to give detailed information to the students. In this way we can link the web-material and textbooks together so that the potential of these two media is efficiently used to support the learning process. We can be quite sure that academic programming textbooks give accurate and deep knowledge about the subjects in hand. In the web-material, we can concentrate on providing examples, pictures, animations etc. to support the learning process in multiple ways. In this way we can focus our attention to enrich the learning process and we do not have to write all detailed Java programming issues ourselves.

Tutor-teachers

Another crucial aspect of our teaching strategy is a tutor-teachers. These teachers work at local high schools and their main contribution to the project is to work as mental support for the students. Because the tutors are not necessarily computer science professional they are not capable of teaching programming for the youngsters. We rely on that tutors can contribute by providing the pedagogical know-how to the students use. For this reason the tutor can choose the working methods and timetables during the courses by her self. There are few minor obligations for the tutor but mainly she can organize the activities in schools rather freely.

On-line tutors

Because the high school teachers lack computer science knowledge, on-line tutoring system had to be set up. On-line tutors work at the university and their main obligation is to go through the students' answers to the weekly assignments. After the assessment on-line tutor gives a feedback to the students. In the feedback tutor can tell all the good or bad parts of the answers. We hope that this way the on-line tutor can give crucial hints and instructions to the students about the things she should concentrate on during the next assignments. When all the assignments have been assessed and the feedback has been given the tutor at the university will release the model solutions of the assignments.

After the "correct" answers have been released we hope that students look up the solutions and compare them to their own answers. In this way (with model answers and feedback) the students can reflect about the good and bad aspects of his solution to the problem in hand [9]. We have made it clear that the model solutions are just one way to solve the given problem. All that really matters is that the answer solves the problem appropriately. In some cases we have directly used the students' answers as a model answer. We hope that this encourages the students to really concentrate on finding the solutions to the given problems.

Another important task of an on-line tutor is to give instant help for those students who need it. Students can use e-mail or bulletin board messages to contact the tutor. In the project our aim is to answer the questions as quickly as possible so that the upcoming problems do not disturb the learning process. After the course the on-line tutor corrects the exams and gives marks. We feel that she has got a pretty good idea about the student's level of understanding about the subject covered during the course.

What have we learned about teaching programming over the net

For us the development of the distance education studies has been fairly new experience. We knew that the programming part of our studies could be difficult one. In some cases we have managed well and the students' opinions have been mainly positive. The assignments and learning materials have inspired the students to work hard with their studies. This can be seen in the students' answers to the exercises, which in many cases have been high quality. In fact one of the first big surprises was the fact that the majority of students accomplished over 70 % of all assignments. Under the normal circumstances university students' first aim is to get 1/3 of the exercises done so they can participate in the exam.

We assumed that the same thing was going to happen with our students. In some cases we had to tell them to slow down with the assignments because the students were complaining about the amount of assignments per week. We pointed out that there was no need for them to accomplish all of the assignments. We think the reason for the amount of completed assignments derives from the high school's learning culture. Often all the work at high school is compulsory to the students and learning is measured for example by the amount of exercises completed. So students do not work with their assignments in order to learn. They do the assignments to prove that they are capable to keep up with the studies.

An other quite disturbing discovery was the fact that students had in some cases over 35 hours of high school studies per week. By taking our courses the students' workload became far too heavy. For some students this lead to situation where they had to quit our studies. They simply did not have any time to concentrate in a fairly complicated domain like programming. At the same time in some schools the tutor-teachers resources were too low so the students did not get the necessary support during the critical periods of the courses. Still there has been a group of students, altogether about 40, who have been capable of handling their high school studies and our studies successfully.

How to intensify the learning outcomes?

During the "virtual certificate"-project it has been clear that out method of teaching programming is not effective for all kind of learners. To make the learning process as efficient as possible we are planning

use the following methods: interactive visual tools, adaptive learning materials and intensive collaboration.

Visual tools

Visualization has long been an important pedagogical tool in CS education. The use of the web and interactive animations provide opportunities to expand the availability of visualization-based teaching and learning tools. We have been experimenting BlueJ, Jeliot and Excel, which provide activating learning tools.

BlueJ is a visual programming environment designed to teach object-oriented programming using Java as the implementation language [8]. BlueJ is based on the Blue system. Blue is an integrated teaching environment and language, developed at Sydney University and Monash University, Australia. BlueJ helps students to develop understanding of object-oriented concepts such as objects and classes, message passing, method invocation and parameter passing. The aim is to concentrate on solving programming problems without becoming distracted by the mechanics of compiling, executing and testing Java programs. The editor is language sensitive and assists with debugging. BlueJ can be downloaded at [2].

Jeliot 2000 is a program animation system intended for teaching computer science especially to high school students [1]. The emphasis is on program animation that demonstrates the execution of input-output, assignment, selection and loop statements. Jeliot 2000 is implemented in Java using version 1.2.2 of the SDK. Jeliot 2000 is complete in that it animates all the constructs of programs it accepts, but the current implementation is limited in the language constructs that it supports. Jeliot 2000 is available at [7].

We have also been studying how to prepare animations of simple algorithms with Microsoft Excel spreadsheet program. Excel offers a light, adaptive and inspiring platform for creating visualizations of various needs in CS [10]. A teacher can prepare visualizations for teaching or visualizations can be student assignments. Two standard features of Excel, i.e. data visualization and macro programming with VBA provide together an easy-to-use environment to animate algorithms. See the Excel page at [5].

In the Excel environment the user starts from a rough idea and he approaches step-by-step a satisfactory solution with the help of the feedback of the system by adjusting and improving the visualization. It has been turned out [10] that Excel provides the students with an easy entrance into the world of algorithms. The students start by simple programs or concepts and visualize them with Excel. After this they can compile more complicated algorithms and animate them with more advanced systems like Jeliot. Jeliot and Excel have been succesfully experimented in computer science courses in the University of Helsinki.

Adaptive learning material

During the two programming courses we noticed some difficulties in students' learning process. Especially during the first programming course we noticed that our learning material was not sufficient for the different learners' needs. Students complained that the learning material was focused on wrong things. For us it was important to offer students a smooth start. And that was the reason why we concentrated apparently too much in the easy part of course to get the things going on. We thought that if the beginning went well we could speed up the pace. In some cases this lead to the situation where some parts of the programming, for example while-loops, were not understood correctly. Maybe the easy start blinded some student to believe that our studies were not so demanding and complicated. The exam at the end of the course showed this very clearly. There were students who managed very well, but there were also students whose performance was fairly poor. In the next year we are going emphasize more on those aspects of the course which appeared to be difficult to the students.

One rather clear absence in our model of teaching is the fact that our web-based learning environment does not pay attention to the needs of different students. For example the learning materials and exercises are same to all students. In this situation it is difficult to come up with learning materials and exercises some are suitable for the use of all kind of students. It was very clear that certain assignments were too difficult for some students and at the same time other students found them to be too easy and they got frustrated. This leads to one clear conclusion; we have to provide different materials and exercises to different kind of students. One can say that the learning environment should be adaptive to the needs of different students' [3].

Collaboration

Modern learning theories emphasize the value of a discussion and collaboration in the learning process [11]. We have found that discussing about programming concepts in the normal bulletin board format is almost impossible. In normal face-to-face learning situation it is easy to point out the different parts of the code. But it is not a trivial task to specify the problem solely in writing format. When the code gets longer the difficulties are even more obvious. We are sure that a collaboration tool for teaching programming should provide means to present or point out the parts of the program witch are under the discussion. When we think further the next step would be environment where students can write and compile programs together. At the same time they can discuss or point out about the most important aspects of the code in hand.

Discussion

The purpose of this paper is to shortly describe our experiences on delivering the introductory programming studies over the net. Our project is going on strongly and right now we are planning the next phase, which will start at the autumn 2001. We are aware that out method of teaching the programming needs to be improved in certain issues. On the other hand we believe that our general principle of delivering the studies is working rather well. There are some crucial issues to deal with but our intention is to enchance implementation of the studies as described in previous chapter.

During the project a few rather interesting subjects for the future research has arisen. One interesting discovery is that our students are eager in giving frank feedback about all aspect of the project. Because the feedback has been mainly pertinent we have been able to change certain thing in the project. For example the time limits of the weekly assignments have been modified according to the opinions of our students. We feel that this kind of chat-like intensive feedback can work as design help for the development and implementation of the distance education projects. Another interesting subject is how to combine the face-to-face learning situations with the distance education as efficient as possible. In future, the need for maximizing the balance between face-to-face education and distance education will be at the great interest of the business and academic communities.

Bibliography

1. Ben-Ari, M., Levy,R., Uronen, P.A., An extended experiment with Jeliot 2000, Accepted for presentation at the Programming Visualization Workshop, Porvoo, Finland, July 2000.
2. BlueJ,http://www.bluej.org/.
3. Brusilovsky, P., Adaptive and intelligent technologies for web-based education. In Peylo, C. & Rollinger, C. (ed.). Künstliche intelligence, Special issue on intelligent systems and teleteaching, Vol.4, 19-25, 1999.
4. Dewey, J., Democracy and Education; An Introduction to the Philosophy of Education, MacMillan, New York, 1918.
5. Excel animations,http://www.cs.helsinki.fi/research/excel/.
6. Haataja, A., Kontkanen, S., Suhonen, J., Sutinen, E., Teaching University Level Computer Science to High School Students over the Web, Accepted for publication at the ED-MEDIA 2001, Tampere, Finland, June 2001.
7. Jeliot 2000,http://stwww.weizmann.ac.il/g-cs.benari/vis.htm.
8. Kölling, M., BlueJ - The Interactive Learning Environments,http://www.bluej.org/.
9. Polya, G., How to Solve It, Princeton University Press, 1958.
10. Rautama, E.; Sutinen, E.; Tarhio, J., Excel as an animation environment, In Proc. ItiCS '97, Integrating technology into Computer Science Education, Uppsala, Sweden, 1997.
11. Slavin, R., Cooperative Learning: Theory, Research and Practice, Allon and Bacon, 1990.