Enhance students' motivation to learn programming by using direct visual feed-back

[1]Lars Reng, [2]Lise Busk Kofoed

Department of Architecture, Design & Media technology, Aalborg University, Denmark,
lre@create.aau.dk[1]
Department of Architecture, Design & Media technology, Aalborg University, Denmark,
lk@create.aau.dk[2]

Abstract

The technical subjects chosen are within programming. Using image-processing algorithms as means to provide direct visual feedback for learning basic C/C++. The pedagogical approach is within a PBL framework and is based on dialogue and collaborative learning. At the same time the intention was to establish a community of practice among the students and the teachers. A direct visual feedback and a higher level of merging between the artistic, creative, and technical lectures have been the focus of motivation as well as a complete restructuring of the elements of the technical lectures. The paper will present the test results based on over 200 students, gathered over a period of three years.

The paper will explain different steps of the new programming courses in detail, and relate students test data to each of the initiatives causing the leaps of improvement. Furthermore the students' technical abilities and enhanced balance between the interdisciplinary disciplines of the study are analyzed. The conclusion is that the technical courses have got a higher status for the students. The students now see it as a very important basis for their further study, and their learning results have improved to a satisfactory level seen from the study board's point of view.

1. Introduction

During the last decade there has been an amazing development in new media and media-platforms. The industry has changed and so has the demand for the engineers working in the content related fields of the many fast growing areas. As a natural response to the new demands of the industry many universities are trying to establish new educations to close the void, by creating a new type of engineer that can meet the new challenges from industry [2].The Medialogy master and bachelor programs was established at Aalborg University approximately eight years ago. Medialogy is a multidisciplinary education with a goal to create a new type engineer with a strong skillset and understanding bridging from the latest media technologies in both software and hardware, to new and old artistic disciplines and to get a deep understanding of the human perceptual system 2. Traditional technical engineers seem to lack an understanding of both the human cognitive system and the techniques and problems linked to creating high value media content. Equally problematic, is the artists and media directors lack of knowledge of how technically challenging their many ideas are for the engineers to develop. The candidates from Medialogy should be able to close this gap, being a valuable asset for any company as a mediator or part of any of the three groups.

The paper investigates the problems of teaching highly technical topics to students who are neither technically skilled nor keen on improving their skills in these topics. The relatively new interdisciplinary engineering program; Medialogy within the technical faculty at Aalborg University, have in the last years been drawing a large amount of students into the void between the many creative fields of media, art, design and the technical engineering disciplines. Whereas the numerous engineering educations have a great appeal on those students, who are both inclined and passionate about highly technical topics, Medialogy seem to have a great appeal for those students who are passionate about the creative side of media, art, and design. Since the start of the new Medialogy study 6-7 years ago, it has been evident that the level, of the more artistic minded part of student's technical side was far from the desired goal of the program. Studying Medialogy implies a certain level of mathematic and programming, which has been a great

problem for many students. The more artistic minded students seem not to be interested in those technical subjects and when they start the Medialogy study they cannot see the reason for learning technical subjects with the results that many students have stopped their study. So a big challenge for the program has been to find a pedagogical approach, which could motivate an interest for the technical subjects and avoid students leaving the study. The authors of this paper have, with full support from the head of studies, taken the initiative to fundamentally change the way technical topics are being taught.

All educations under Aalborg University are following the Aalborg University pedagogical model based on Problem Based Learning (PBL) principles.[1,3] According to the Aalborg PBL model students would every semester use approximately half of the study time by working in their group with a project where they have to choose and solve a problem within a field selected from the overall theme of their semester. The other half of the time would be used on courses related to the project topic or as a prerequisite for courses for an upcoming semester. So each semester students would have to analyze the semester theme, find a relevant problem they want to solve, use or develop suited theories, find useful methods, design a product, and after thorough testing evaluate if the problem had been solved.

After the first three years it became evident for Medialogy that the type of engineer students that were attracted to Medialogy, were less technical inclined and prone to learn hard technical topics. So the consequences were that the teachers expectations to the students' technical level were lowered, and some areas where a technical skillset was the desired goal only a superficial understanding was expected. It became clear that the education needed to critically evaluate the technical parts of the program, and the result showed an urgent need for improvement of the technical courses, especially programming.

Four years ago the first author of this paper was hired to teach programming on the bachelor level of Medialogy. The paper will describe the stepwise transformation of the course in the attempt to raise the students' technical level to the original desired goal.

2. Methods

We have used the case method combined with action research. All four programming courses have been followed and analysed during four years. The courses have been developed during the process according to previous experiences and results. It has been a process with characteristics from continuous improvement (CI) processes, and the evaluation processes have been inspired from CI evaluation methods [9]. The data are course assignments, exams, interviews with students and project supervisors.

The pedagogical approach is based on the PBL principles [7] and one of the main focuses in the courses is to relate the course content to understandable problems for the students so they could see the purpose of the learning. Another teaching strategy was to emphasise and support reflection as a mean for understanding [8] and to use students previous experience either from programming courses or from related areas to support their learning process and to keep motivation [3,4]. We have used experiments with inspiration from Donald Schôn [5,6] when developing different aspects of the courses. But one of the most important factors was to develop course material and course content "just-in-time".

3. Programming at Medialogy

The bachelor program of Medialogy was originally only two year education since the students could apply after taking a two year long multimedia college education, which would give them the needed merit to skip the first year of a three year bachelor education. The programming part of the education was therefore split evenly between the third to sixth semester. The third semester would introduce the basic concepts of programming, the fourth would add the object oriented programming (OOP) concepts, the fifth adds 3D graphics programming with OpenGL, and the sixth finally introduces artificial intelligence programming (AIP).

This paper will focus on the changes made to the 3rd semester C/C++ programming course. The students are on the 3rd semester introduced the basic concepts of procedural programming. The curriculum includes elements such as data-types, variables, loops, branching, arrays, structs, functions, pointers, searching & sorting, linked lists, trees, etc. All object oriented programming is not taught until the 4th semester.

3.1. Detecting the problem (Spring 2007)

The first course to teach was the Object Oriented Programming course (OOP). The previous teacher had left rather suddenly a month before with no desire to support the successor. Therefore no exchange of knowledge and experience from previous years was given. And as a direct result here of, the course was designed and run in a very traditional engineering education style. The course, which was the students' second programming course, was split in 15 lectures of 2x45 minutes of auditorium lecturing and 2x45 minutes of assisted exercises; in the students' own group rooms.

It rather quickly became evident that only a fraction of the students mastered the basics of the first programming course ending only a month before. As a direct result, it was very hard to focus the teaching on the desired curriculum.

To investigate if this problem had occurred before, and also how well the students preformed at the end of their bachelor, the entire 6th semester Artificial Intelligence Programming course (AIP) was observed with focus on the students' abilities to apply basic programming to solve the exercises. The results were shocking. From a whole semester of 6th semester students only one student dared to attempt to solve the final free competition exercise. So even though his simple artificial intelligence did not work perfectly, he still won.

Numerous meetings with the coordinator of studies concluded that the last four years of changing teachers and choice of programming language had proven that more extensive changes would be required in order to break the reoccurring problem with a weak technical level of the majority of the students. High levels of freedom to initiate new ideas were therefore granted. Also, it was agreed that the requirements to pass the exam should be raised greatly the first year, and then by another 10% the following three years, as the new improvements to the course hopefully would gain full effect.

3.2. Raising the bar (Fall 2007)

Interviews with the students had revealed that many had been able to pass the 3rd semester programming exam using a high level of memorization of blackboard examples. In order to prepare the students for the challenging programming courses on later semesters, the severity of the 3rd semester course had to be raised. This would clearly result in a much higher number of failed exams and students dropping out, unless a better way of teaching the curriculum was found.

The students had for the last four years, since beginning of the education, had a course on the 3rd semester in image processing and they had built an interactive installation using this as part of their semester project. This had however been done with tools such as EyesWeb, where the details of the different operations were hidden and handled by the tool. In order to bring more focus on the programming part of the semester it was decided to remove the tools, and instead the students were required to program the entire project artefact in C/C++. The open computer vision library (OpenCV) was all the external software they were allowed to use.

The programming course was run in parallel with the early stages of their semester project and the image-processing course. The programming course was done in a classical style; using two hours of lecturing and another two for exercises.

The students came poorly prepared to the programming lectures and almost half of them seemed to lack any motivation to learn programming, or do the exercises required. More than half the students were unable to apply what had just been taught in the lecture minutes before the exercises. There was almost no flow in the exercises, and most students seemed to be stuck as soon as the teaching assistants had left them.

The semester project artefacts were programmed by the project groups in C/C++. (Each project group had 4 -6 students), Project exams unfortunately revealed that only the strongest programmer in the group was able to explain the code, indicating that not all students had benefitted from the extended programming practice possibility which work on the semester projects should have offered. On the other hand those that had programmed major parts of the

projects could explain in a very high level of detail how the different image-processing algorithms affect an image. The programming exam questions were increased greatly in level of difficulty, as agreed upon. This unfortunately resulted in more than 50% failing the course. (See figure 1 in the result section below).

3.3. Merging the courses (Fall 2008)

Interviews with supervisors and skilled students had the previous year indicated that programming all the algorithms from the image-processing course would greatly improve the students' understanding of how and why they worked the way they do. Another interesting observation done during the breaks of the programming courses, was that many of the students who lacked the motivation to learn programming, was using their breaks to continue work on the more artistic courses. It was therefore decided to merge the image-processing and programming courses, and to use all the filters used in the more artistic classes as programming exercises. If the more artistic minded students were asked to recreate some of the effects they liked from software such as Adobe Photoshop, it might increase their motivation to understand how these filters were programmed, and thereby learn more programming. Another important benefit was that changes in images often seem to make more sense for the students than only numbers from a program. It was therefore decided to completely merge the two courses: programming and image-processing.

Instead of starting with a two hours lecture and then two hours of exercises the students were taken out of the auditorium and into a large seminar room. This allowed the lecture- and exercise time to be interleaved in 15-20 minute intervals, and thereby allowing students to implement every new method directly after it had been presented and discussed. The teacher was during lecture time close to the students and could move around between students. It was easier for students to ask questions, but also for the teacher to ask if the students had understood what was going on.

The effect of this merger did according to interviews with students and supervisors have a positive effect not only on the students' knowledge on image-processing but it also resulted in better semester project artefacts. The level of the programming exam was raised by approximately 10%. Even under these conditions the results of the exam indicated that the changes from the previous year have had a positive effect on the level of the students' programming skills. (See figure 2 in the result section below).

3.4. The semester spirit (Fall 2009)

A phenomenon that is daily discussed among the teachers but still is a bit of a mystery is the semester spirit. Even though we know there is a great difference between the best and weakest students and a great difference between the most and least dedicated students, everybody knows that semesters can be very different. Something can make students at a whole semester accept lazy behaviour or they work harder than those a year before them. Even though only minor changes were added to the image-processing and programming course, the students in the fall semester seemed to be less motivated and more reluctant to implement all the exercises. This had an overall effect on both semester projects and the programming exam. Again the severity level of the programming exam had been raised by approximately 10%. (See figure 3 in the result section below).

3.5. Focus on the artists (Fall 2010)

In order to do everything possible to avoid another year with a collective lack of motivation, several of the master students that had passed the exam two years earlier was invited to talk to the new 3rd semester in the very beginning of the course. Not only the strong programmers were invited, a few very artistic minded master students presented how they used their programming skills to improve their work daily. The aim of this special effort was to try and motivate"the hard to reach" and more artistic minded new students. Another initiative added this year was that the last part of the image-processing course was delayed and given at the time where the students were implementing their semester project artefact. Instead of traditional course teaching, these hours were used as an open support to any image-processing or programming problem related to the semester project.

The programming exam had the desired level of severity at approximately another 10% harder than the year before. The results were better than expected, (as shown in the result section below).

4. Results

In Denmark students are graded after the 7-scale system. The grades -3 and 00 are failing grades. The grades 02, 4, 7, 10, and 12 are passing grades. (12 = A). The below four figures depicture the total sum of grades given according to the 7-scale. The x and y –axis represent the grade and frequency for the fall courses 2007 - 2010.

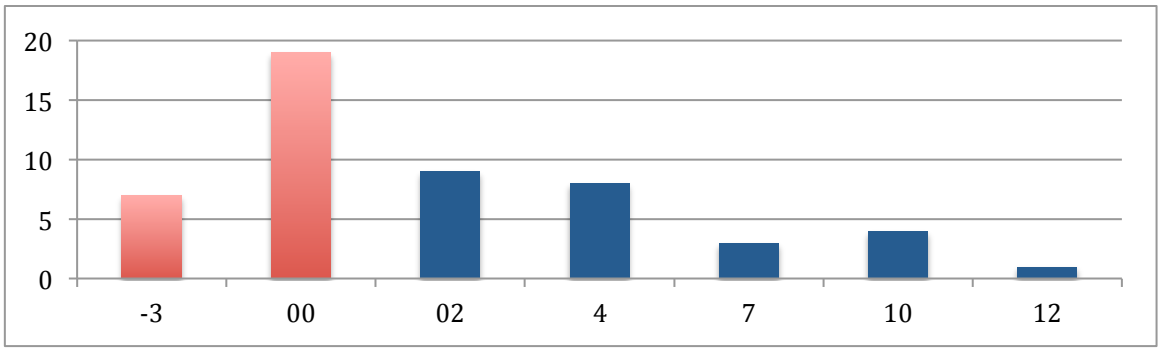

Figure 1: Programming grades 2007

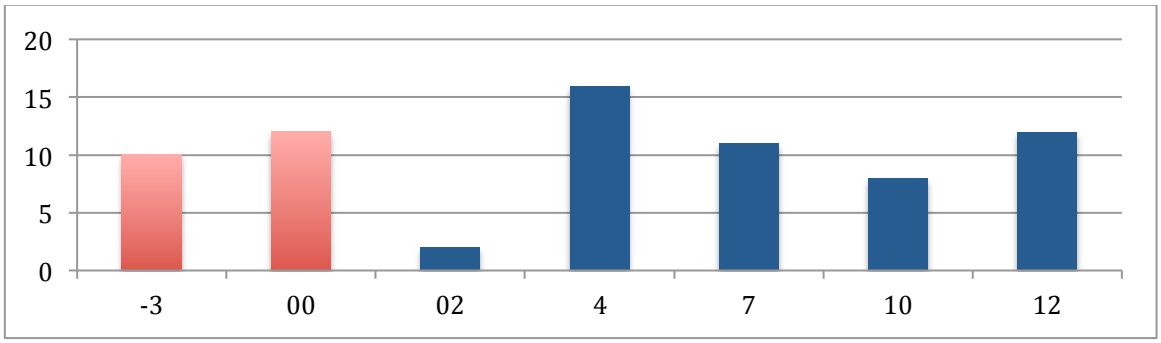

Figure 2: Programming grades 2008

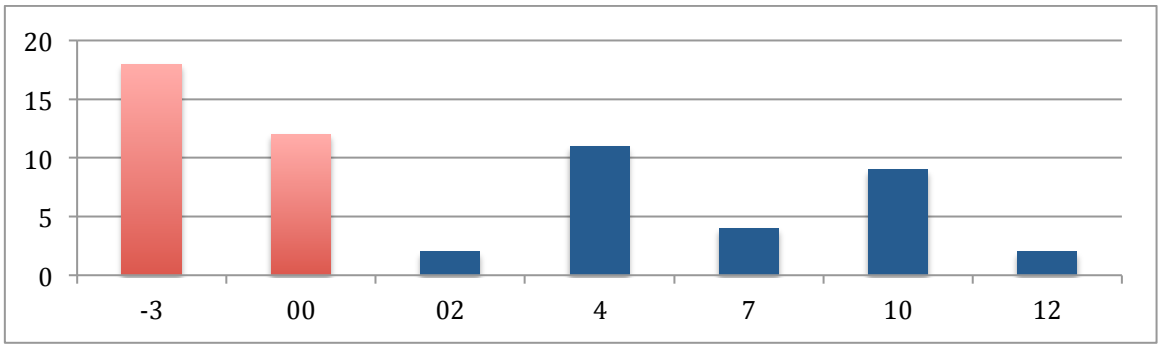

Figure 3: Programming grades 2009

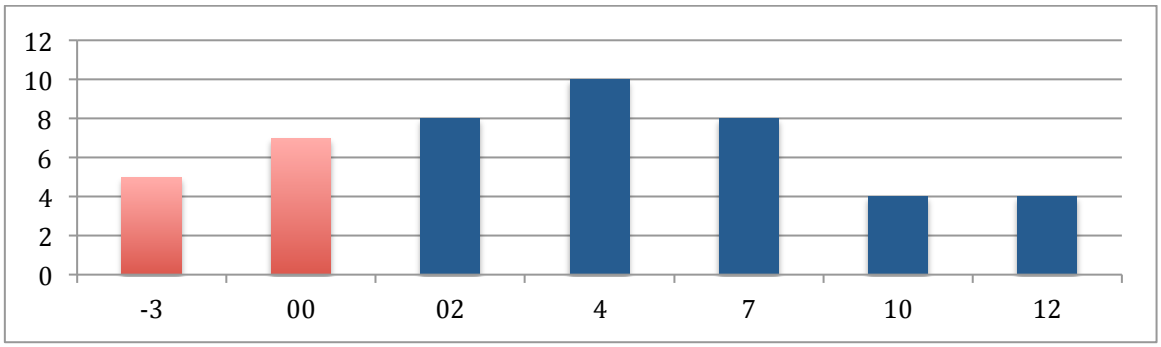

Figure 4: Programming grades 2010

5. Conclusion

The merging of the courses programming and image-processing were done as a natural support to the students new requirement of having to implement an image-processing based interactive artefact by the use of C/C++ in their semester projects. When this initiative was approved four years ago there were none or little actual programming in the projects on the bachelor semester (6th). The artificial intelligence programming course was rarely used in any semester projects. Today this has changed dramatically. Several bachelor groups are today implementing advanced algorithms for artificial intelligence in their semester projects. Numerous projects are being programmed in industry standard game engines, and students are starting to program full-scale commercial productions in their early master semesters. The ideas of improving the course have been good, and the use of programming in the students semester project had supported the motivation as well as the learning.

The direct visual feedback achieved by using images as output for the exercises in the programming course, has been observed as a strong motivator and potent debugger for the very graphical minded students at Medialogy. The direct visual feedback is not the only initiative applied to the 3rd semester programming course .For the moment it is therefore not possible to conclude the effect based on the exam results depicted in the result section.

References:

1. Busk Kofoed, L. Nordahl, R. (2007) *Learning Lab – teaching experienced students PBL.* In proceedings of the 18th Conference of the Australasian Association for Engineering Education, Melbourne; Department of Computer Science and Software Engineering. University of Melbourne.
2. Nordahl, Rolf and Kofoed, Lise B. (2008). *Medialogy – design of a transdisciplinary education using problem based learning.* In Proceedings form SEFI 36th Annual Conference., Aalborg.
3. Kofoed, Lise; Hansen, Søren ; Kolmos, Anette.(2004) Teaching Prosess competencies in a PBL curriculum.In:*The Aalborg model : progress, diversity and challenges*. (Eds. Kolmos, Anette. Fink, Flemming K., Krogh, Lone. Aalborg: Aalborg University Press.
4. Weick, K. E., Sutchcliffe, K. M., &Obstfeld, D. (2005). *Organizing and the process of sensemaking.* Organization Science, 16(4), 409-421.
5. Schön, Donald, D. (2009*) The Reflective Practitioner. How Professionals Think In Action.* Ashgate Publishing Limited, England.
6. Schön, Donald, D. (1990) *Educating the Reflective Practitioner – Toward a New Desogn for Teaching and Learning in the Professions*.Jossey – Bass Publishers, San Francisco.
7. Graaff; E. de and Kolmos, A. (2003) Characteristics *of problem-based learning:* International Journal of Engineering Educations. 5(19). 657-662
8. Cowan, John (2006). *On becoming an Innovative University Teacher – reflection in Action.* Open University Press, McGraw-Hill Education.
9. Jørgensen, Frances; Kofoed, Lise B. (2007) Integrating the development of continuous improvement and innovation capabilities into engineering education. In: European Journal of Engineering Education. 2007; Vol. 32 nr 2. Pp. 181 – 191