FDM3507 Programming as Design Practice 7.5 credits

Programmering som designpraktik

This is a translation of the Swedish, legally binding, course syllabus.

If the course is discontinued, students may request to be examined during the following two academic years

Establishment

Course syllabus for FDM3507 valid from Spring 2015

Grading scale

Education cycle

Third cycle

Specific prerequisites

All doctoral students or researchers, with either computer science/technical background or background in art and design.

Language of instruction

The language of instruction is specified in the course offering information in the course catalogue.

Intended learning outcomes

On completion of the course, the doctoral student should be able to:

Use a number of different programming paradigms in small projects and argue for when each paradigm is appropriate.

Use a number of different programmings practices in small projects and argue for when each practice is appropriate.

Understand and apply technologies for interactive interfaces, and the fundamentals of sound, video and generative computer graphics.

Discuss when it is appropriate to choose programming as procedure, and when reuse of existing solutions can be more appropriate.

Combine several systems existing and new, in order to create a new entity that reaches a set design goal.

For this, they should learn technologies for communication between programs and other involved systems, so that these can share control data, sound streams and video streams between themselves.

Course contents

The aim for software development has traditionally been to create a predictable, rigorous, transparent process, so that a, for the assignment well-working, software can be delivered in time, and within budget. Computers are used today in many other locations than clearly task oriented contexts, such as class-rooms, homes, and studios of architects, designers and artists. The traditional working method for software development has been changed as well as the tools for it. New procedures have arisen, that go beyond the formal, engineering view on programming. In particular, the aesthetic aspects have been emphasised and new forms for how to organise software development projects have arisen. Courses will particularly focus the issue: How can the programming procedure promote aesthetic creativity?

The aim of this course is to teach students skills that are required for programming within creative practice. The course will partly present programming languages that are used in creative contexts and partly how one can approach programming in various ways. The course will interleave examples of how developers have used the programming languages with concrete proficiency exercises where the student uses the programming languages.

Disposition

Outline

The course consists of two distinct blocks:

1. Theoretical Lectures

2. Project work

Lectures and workshops

During the first block, 10 two-hour lectures will be given.

1. Creative programming: examples and history (One lecture)

2. The Processing language (Three lectures)

a. The fundamentals

b. Programming with methods

c. Object orientation

3. Pure Data language (Two lectures)

a. Manipulation of control-data

b. Sound manipulation

4. Parameter mapping and the Open Sound Control protocol (One lecture)

5. Sensor platforms (One lecture)

6. Elective topic lecture 1- the contents are decided after the first meeting with the participants

7. Elective topic lecture 2- the contents are decided after the first meeting with the participants

This course is collaborative: course participants will take turns to prepare and present the lectures and to organise the practical study groups.

In the first meeting, all participants tell which parts in the course they want to participate in by presenting, and vote on what they want to be included in the course contents

During the course, participants will, individually and in pairs, work with project where they apply what they learnt.

Between the lectures, 10 two-hour lab meetings are planned, where the participants apply what they learnt on previous lecture through small project work. To which extent this is followed, depends on whether the course has enough number of participants that want to participate and lead the sessions.

Project work

The course includes three project works:

Two small, individual where the student tests a new programming language and development paradigm. At least one of them must produce results in real time.

The third project work is carried out in pairs, where two or several of earlier works will be adapted so that they can be incorporated in a new combination, where they together constitute a new whole.

In this group assignment, we try again to mix skills so much as possible so that each technical knowledgeable participant will work with an artist/designer.

In addition to lectures, much space for reflection around the practice will be given.

All these project works will be carried out during unscheduled time, as individual projects.

During that period, the participants should plan six 2-hour meetings where they present their project, and participate in crit sessions on one another's work.

Six meetings on project work:

1 One to start the project and present ideas for the first project.

2 One to show the development and obtain feedback on the first project (Crit session).

3 One to show results of the first project work and also present ideas for the other project.

4 One to show the development and obtain feedback on the other project (Crit session).

5 One to show results of the other project work, create groups for the third project and start to retrieve ideas for the third project.

6 One to show progress and obtain feedback on the third project work (Crit session).

Blog

During the whole course, each student should write on a blog about his or her activities. After each lecture, the blog should be updated with a personal reflection where the contents of the lecture and research articles are related to the own research.

The aim is to give a good ground to write a research article that is based on the experience with this course. To write an article is however not an examination requirement.

If the student has technical questions, these should be asked on relevant internet forums. A link to the issue should then be placed on the blogg, so that the responsible lecturer can reply on the issue. This will help the students to learn to formulate their technical questions well and to use these forums to find solutions to technical problems.

After each lecture, each student should write a small program along the contents of the lecture. They should also reflect on how the contents of the lecture related to the experience of writing the program.

The blogs are read by fellow students, who give feedback:

- All students read one another's blogs before meetings (if the blogs are too many, each student obtains a smaller number assigned).

- The students should comment on the content of these blogs, and bring up the comments on the following Crit-session.

- Students should reflect on the feedback they get from the crit sessions, and document their work on their blog.

Course literature

Ca. 20 forskningsartiklar. Dessa innefattar bland annat:

“Processing: Programming for Designers and Artists”, Casey Reas, Ben Fry, Design Management Review, Vol. 20 No. 1, 2009

“Crafting Code at the Demo-scene”, Hansen, Norgård, Halskov,DIS 2014

"Pure Data: another integrated computer music environment."Proceedings of the Second Intercollege Computer Music Concerts, Miller Puckette, (1996): 37-41.

“Code Bending: A New Creative Coding Practice”, Ilias Bergstrom, and Beau Lotto, Leonardo Journal, MIT Press, February 2015, Vol. 48, No. 1

“Open Sound Control: an enabling technology for musical networking”. Matthew Wright (2005), Organised Sound, 10, pp 193-200. doi:10.1017/S1355771805000932.

“OSC-Namespace and OSC-State: schemata for describing the namespace and state of OSC-enabled systems”, Ilias Bergstrom, Joan Llobera, New Interfaces for Musical Expression, NIME’14, June 30 – July 3, 2014

“Soma: live performance where congruent musical, visual, and proprioceptive stimuli fuse to form a combined aesthetic narrative”, Ilias Bergstrom, Beau Lotto, Leonardo Journal, MIT Press (Accepted for publication, posted online August 26, 2014)

“Mutable mapping: gradual re-routing of OSC control data as a form of artistic performance”, Bergström et al, ACE 2009

“The Practices of Programming”, Ilias Bergström, Kristina Höök, in preparation.

Examination

Based on recommendation from KTH’s coordinator for disabilities, the examiner will decide how to adapt an examination for students with documented disability.

The examiner may apply another examination format when re-examining individual students.

The examination consists of:

1. An active participation during the lectures and the tutorial sessions.

a. Each participant presents at least one lecture and/or organises one tutorial session.

b. You should not miss more than 2 lectures, 2 tutorial sessions and 2 crit sessions. If you miss more sessions, you need to talk to your course planner about how you can compensate for this.

2. All three projects should have been carried out and presented.

3. The whole process is documented by the student in an individual blog. The blog is seen as basis of examination, both through the process it describes, and the quality in itself.

Ethical approach

- All members of a group are responsible for the group's work.
- In any assessment, every student shall honestly disclose any help received and sources used.
- In an oral assessment, every student shall be able to present and answer questions about the entire assignment and solution.