computing requirements *creative technology*

from a *new media* perspective

Anton Eliëns, with Angelika Mader

abstract In this report we look at the requirements for the *computer science* or *computing* track in the *creative technology* curriculum. We will propose computing courses, that may be taken as a reference when developing the final curriculum.

status: 18/2/08 (discussion)

introduction

Although the *creative technology* curriculum is at this stage still in development, with contributions of a variety of tracks or sub-disciplines, an outline of the educational goals of *creative technology* will look like:

educational targets – *creative technology*

- skills – computing, mathematics, simulation, technology
- knowledge – mathematics, computer & software architecture
- theory – systems engineering, media & communication, human factors
- experience – project(s), deployment in social context

Correspondingly, the topics treated in the curriculum, or the elements of which the curriculum will consist will encompass:

learning goals – *creative technology*

- computing – architecture, networks, programming
- technology – new media, smart technology
- creative applications – creativity (mental + artistic), psychology, research/design methods, communication
- business – marketing, planning, project management
- design – sketch, prototype, realize

Taking these elements as a guideline will help us in determining what role the *computing* track will play in setting up the curriculum.

background - the role of computing

From a more general perspective, the area of computing, or in other words, the discipline of *computer science* should set as educational goals:

educational targets – *computing*

- skills – programming in various languages, able to learn new languages quickly
- knowledge – networks, web-applications, programming languages, operating systems
- theory – integration of languages, computer & software architecture, algorithmic complexity
- experience – application development, (technical) requirements analysis

More specifically, the *computer science* track in *creative technology* should cover, at a yet to be determined level of depth, the following topics and subjects:

learning goals – *computing*

- network – internet, organisations, graphs
- computer – elements, programming, algorithms
- operating system – assembly, compilers, multi-programming
- language – formal/natural, imperative, functional, logical, C++/Java
- hardware – memory, chaching, graphical programming support
- database – representation, storage, query (optimazation)
- web – client/server, web-services, data-driven application(s) – standard(s)

- media – scripting (ECMA+), event handlers/models

The level or depth at which these tpoics should be treated is determined by the requirements of the two specialisations envisioned for *creative technology*, respectively *new media* and *smart technology*.

new media – targets and learning goals
The eucational targets for the *new media* curriculum, may be summarized as fowllows:

educational targets – *new media*

- skill(s) – scripting, programming, interaction design
- knowledge – web, multimedia & game technology
- theory – understanding of media & communication theory
- experience – concept development & realization of (playful) application(s)

Elements of which the *new media* curriculum will consist, at least for the students taking *new media* as a specialisation, include:

learning goals – *new media*

- interactive video – in customizable format
- web technology – for developing information portal(s)
- animation – for simulations and (physical) systems
- virtual reality – for games and virtual environments
- game development – for entertainment and instruction
- rich internet application(s) – for multimedia (web) applications

Since *new media*, which includes the area of (serious) game development, requires a wide range of skills and knowledge, including programming as well as digital content creation, it is unlikely that all students will or need to be trained in computer science uniformly.

smart technology – additional requirements
For the specialisation of *smart technology* we may, perhaps somewhat naivelty, come up with the following list of educational goals:

educational targets – *smart technology*

- skill(s) – modeling, construction
- knowledge – mechanics, ubiquitous computing, smart systems
- theory – human perception, privacy, security
- experience – deployment of (multi) sensor systems

Minimally, the topics in *smart technology* will include:

learning goals – *smart technology*

- modeling, control systems,
- smart technology engineering
- instrumentation – software development

In particular for instrumentation the *smart technology* curriculum will very likely require more advanced, that is specialized, programming skills than the *new media* curriculum.

the computing curriculum – first proposal
In the *new media* curriculum, which is at the time of writing, still in development, the following courses will likely be included:

course(s)

course	credits	description
NM1	3	web technology (1) – html, javascript. css
NM2	6	animation in 2D
NM3	6	web technology (2) – php, sql, web services
NM4	6	3D virtual environments – x3d/vrml
NM5	6	game development – C++/DirectX

Taking the *new media* curriculum, sketched above, as a point of departure, we arrive at the following (mandatory) courses for the *computer science* curriculum:

course(s)

course	credits	description
CS1	3	computer & network architecture(s)
CS2	6	programming fundamental(s) – C++/Java
CS3	6	advanced programming – idoms, APIs

In the proposal above we have, with an eye on practical feasibility, allowed for including *acripting* in the courses for *new media*, thus alleviating the requirements for the *computing* courses.

As to the actual realization of the courses, see the references below, it might be worthwhile to look for a cooperation that allows for dividing laber between the staff alloted to the various tracks, and, more in particular, that allows for a distribution of the work according to personal style and preference for respectively an inspirational, example-based approach, and a more technical bottom-up approach.

As indicated in reference (8), we should be beware of making a choice based on *ease of education.* More explicitly, with regard to both system aspects and performance a choice for C++ as a first (real) programming language, as opposed to scripting languages, seems to be preferred over a choice for Java, despite the pitfalls in teaching a complex language like C++.

relation to other tracks

When we look at the other tracks within the *creative technology* curriculum, we may list as educational targets for *mathematics*:

educational targets – *mathematics*

- skill(s) – problem solving
- knowledge – algebra(s), graph theory
- theory – dynamic systems, logic
- experience – modeling complex systems

and, as educational targets for the track *design*, that is to be developed in collaboration with the *industrial design* department:

educational targets – *design*

- skill(s) – drawing, modelling
- knowledge – design methodology
- theory – human factors
- experience – design & prototyping

For the *mathematics* track we may remark that the track itself does not require computing skills as such. However, it is very likely that the *mathematics* track will inspire and inform the *new media* track, and to some extent even the *computing* track.

The *design* track might benefit from computing skills, but it seems most likely that these will be covered within the *sacripting* part of *new media*

conclusions

In this note we have sketched the requirements for the *computer science* or *cimputing* track within *creative technology*, and we have proposed a collection of courses to meet these requirements.

We again emphasize that in the realization of the courses, and the distribution of thw workload among the staff, we should strive for mutual contributions to profit from individual style and expertise.

reference(s)

online: create.eliens.net [resource(s)]

1. Teaching Software Engineering through Game Design
2. Toy Projects Considered Harmful

3. The Rethinking CS101 Project
4. Creating a Science of Games (CACM)
5. Software Engineering Issues in Interactive Installation Art
6. Programming by Example – A Creative Programming Environment, Remixed
7. Supporting Creative Thinking through Opportunistic Software Development
8. Computer Science Education: Where Are the Software Engineers of Tomorrow?