

The Design of Technological Tools for Thinking and Learning

General Information

Learning Sciences 426 (cross-listed with Computer Science)
Spring 2009, Wednesday, 2:00-5:00 PM
Annenberg 303, South Learning Studio

Professor

Uri Wilensky

Tel. (847) 467-3818, Annenberg 337
uri@northwestern.edu

Teaching assistants

Michelle Wilkerson

Tel. (760) 877-0121, Annenberg 224
m-wilkerson@northwestern.edu
Office hours: Friday, 2-5pm (in Ford SB 250)

Pratim Sengupta

Tel. (847) 467-7432, Annenberg 222
g-sen@northwestern.edu
Office hours: Thursday, 10am -1 pm

Course Web Site

<http://ccl.northwestern.edu/dtttl2009/>

Blackboard Web Site

<http://courses.northwestern.edu/>
(use your NetID and password to log in)

Email Lists

Course Instructors: cd-fac@ccl.northwestern.edu
Course Members (students and faculty): cd@ccl.northwestern.edu

Course Description

This course is a hands-on practicum in designing and building technology-enabled curricula and learning environments. We will use many rich software toolkits designed to enable novice computer-users to get their “hands dirty” doing iterative software design. In addition to the hands-on component, the course is also designed to introduce you to the Constructionist Learning design perspective. This perspective, first named by Seymour Papert and greatly influenced by the work of Jean Piaget, is very influential in the learning sciences today. The Constructionist approach starts with the assumption that teaching cannot successfully proceed by simply transferring knowledge to students’ heads. Skillful teaching starts with the current state of knowledge of the student. In order for students to learn effectively, they need to construct the knowledge structures for themselves. In the spirit of Constructionism, we will engage in our own construction of artifacts in this class and, through this activity, explore and evaluate the design of kits and tools intended to enable learners to construct their own motivating and powerful artifacts. We will do this by constructing both physical and virtual artifacts and by engaging in reflective discussion of both the artifacts themselves and the tools used to construct them. In the final project, students will put all of this together by designing and implementing a constructionist learning environment.

After completing this course, you should be able to:

1. Design and implement educational software at the prototype level.
2. Design technology-enabled activities that take advantage of the computational medium.
3. Exercise good judgment in such design within the target context, content domain and deployment situation.
4. Avoid common educational software design errors.
5. Assess learning technologies as to appropriateness for educational needs.
6. Evaluate and utilize educational claims of software authors and promoters.
7. Understand the Constructionist design perspective and use it to author and assess software tools and learning environments.

This class will emphasize authoring projects using Logo-like languages. Logo is a computer programming language designed explicitly for use by children and is in use in large numbers of schools, from elementary on up.

Note that no previous programming background is assumed.

In fact, the computer languages used in this course are designed to be easy to learn and many thousands of children use them. It is my belief that even if you do not intend to be an educational software designer yourself, it is the reality of today – and more so, of tomorrow – that should inform your choice to become educated about the promise of technology in

education. I am confident all of you can learn the programming aspect of the course, as have many students in the past, who had had no prior programming experience. However, programming does take time and you will be expected to devote substantial time to it. This might be frustrating to many of you, initially, but after the first few weeks, you will have the skills you need. You are strongly encouraged to get help from your fellow students through the class email list as well as from the TAs. The TAs will hold weekly office hours designed especially for technical and programming support. We will attempt to schedule these office hours flexibly, and per special requests.

In addition to projects, there will be weekly readings: typically, one paper or two short papers per week. There is a considerable literature that we will not have time to read this term. I have provided a more extensive bibliography at the end of the syllabus. You may find some of these readings to be useful to you in completing the final project.

Software packages we will use

We will use quite a number of learning software packages in this course. The 3 packages we will use the most are all based on the computer language Logo.

They are:

- **Microworlds Logo** – a multi-media version of basic Logo in common use in elementary schools worldwide. It also includes music, graphics, video and web tools.
- **NetLogo** – a multi-agent version of Logo, this language is tuned for constructing models of complex dynamic systems. It is useful for creating models of ecological systems, chemical systems, economic trade, social behavior, etc.
- **NetLogoLab** – a NetLogo extension that enables NetLogo to communicate with real world devices such as robots and sensors. We will construct devices that have sensors and motors and can interact with objects in the world (e.g., LEGO robots).

Besides these 3 basic packages, in the software review section of the class, we will also explore a number of other packages. Software we might look at includes: TableTop, Genscope, Biologica, Zoombinis, SimCalc, HubNet, ChemSense, Fathom, MediaMoo, Moose Crossing, CSILE, Hypergami, Stagecast Creator, Vehicles, RelLab, Interactive Physics, the Sims, Impromptu, Geometer's sketchpad, MatLab, Squeak, Boxer, Model-It, STELLA, MyWorld, Scratch, etc.

Summary of Requirements

This course is designed to be somewhere between a class and a working group. I'm hoping that we'll work together to make sense of readings, and, for most of the class projects, you will be working in small groups.

So the requirements for everyone are:

- Keep up with the readings and participate in class, both in person and virtually. You will be expected to post a comment on each week's reading by Monday at 5.
- Complete and present several (mostly group) programming assignments using Logo, NetLogo and NetLogoLab.
- Review one educational software package and present your review in class.
- Design and implement your final project.
- Give a presentation during the last week of the course.

In addition, due to the group project nature of the class, you are also asked to send email to cd-fac@ccl.northwestern.edu (as soon as you know) if you cannot make a particular class meeting. You are also responsible for communicating with your project-mates and letting them know in advance if there is any problem with your part of the project.

About the Final Project

The final project is to design and implement a constructionist learning environment. There are two* basic alternatives for this project:

1) Standalone Educational Software (scaffolding in software)

Design and implement some constructionist educational software. This option would involve writing a *design specification* for the software that describes what the software is for, who it serves, why it is needed, why it is best done in software, etc. Subsequent to receiving feedback on the design specification you will need to start working on a *functional specification* of the software itself and then embark on implementing it. You are free to use any authoring tools you like to implement the software as long as you make a good argument for their being well matched to the task. Suggested educational software genres are: a simulation game, a microworld, a collaborative role-play or MUD (a collaborative virtual space or Multi-User Dimension).

2) Software-embedded curriculum (scaffolding in curricular materials)

Design and implement an educational activity that has a computationally embedded component. In this option, you are asked to use one of the three main software environments used in this course: Microworlds Logo, NetLogoLab or NetLogo. As above, you would begin with a *design specification*. Depending on the design, you may or may not require a functional specification – it could be a *curriculum flow specification* instead. You would then go on to construct the software and/or Lego constructions that form the kernel of the activity, flesh out the curricular materials that accompany the software and write up a paper that describes one person's (could be yourself) path through the activity.

For some students, the final project could take a different direction, such as designing a (computational) research model of organizational change using NetLogo. If you're interested in this option, come and talk to me.

Important dates for the final project

- The final project design specification is due by **May 13th**.
- The final project functional specification (or curricular flow specification) is due by **May 27th**.
- The final project is due by **June 6th**.
- Final projects will be presented on **June 10th**. You are welcome to invite friends and/or relatives to attend.

Grading

All assignments and projects will be graded as either complete or incomplete. If a project is judged incomplete, you will have an opportunity to complete it or redo it the following week. If you cannot complete the final project by **June 6th**, you may take an incomplete for the course. No penalty will be assessed for late final projects – they can be handed in as late as the following quarter and your incomplete grade will be changed at that time, but you must make a coherent presentation on **June 10th**. You will also be assessed on your class participation both in class and virtually.

Readings

A course reader is available from Quartet Copies at 825 Clark Street, Evanston. You also need to purchase a book at Norris:

- Papert, S. (1980). *Mindstorms*. New York: Basic Books.

(See the courses website for links to book merchants.)