

# Teaching programming concretely

*MassCUE / M.A.S.S Presentation – 10/28/2009*

*Russell Zahniser – MIT Scheller Teacher Education Program*

*russell@zahniser.net*

## Agenda

- 1) Why teach programming in high school?
- 2) My experience teaching APCS
- 3) Mistakes that I made
- 4) My solution

## Why teach programming in high school?

- 1) Programming is a basic skill for science and engineering jobs
  - Prior programming experience is necessary to get the benefit of college classes
  - Compare to learning the craft of journalism or fiction writing when you struggle with the mechanics of grammar
- 2) Programming is a powerful mental discipline, like karate for your mind
  - Builds accuracy of thought and confidence in the power of your reasoning
  - Helps students to develop logical and mathematical reasoning
- 3) Programming is the art form of choice for children in a computer-saturated culture
  - Permits ideas to be expressed and experienced in an almost limitless medium

## My vision for my APCS class

- 1) A “real academic class,” not just messing around on computers
- 2) Allow students to write key parts of a well designed, complex game
  - Scaffolding to allow them to create cool programs that would normally be beyond the reach of a beginning student
- 3) Daily lectures introducing new ideas
  - Give students a precise description of key ideas without a lot of frustrating trial and error figuring it out for themselves
- 4) Homework and written tests to keep all students on track and assess learning

## My results

- 1) “Basic” ideas like variables, methods, and objects were the hardest to learn
  - Algorithms and data structures are what programmers tend to think are “hard”
  - I found myself unable to get these basic ideas across to some students, and that left them unable to understand their programs at all
- 2) Projects were too elaborate
  - Students did best on projects that gave them the freedom to implement their own simple design, even though their design was not as “good” as mine
- 3) The only students who thrived were those who could have just learned from a book
  - Yet, others were bright students who had done well in my physics class
  - I had hoped my more graphical approach would open up programming to those who were not mathematical thinkers

## **My mistakes**

- 1) Syntax is easy; the ideas expressed by that syntax are hard
  - Programming languages have idioms beyond simple nouns and verbs, expressing concepts that exist only in the strange imaginary world inside the computer
- 2) The goal in high school should be to competently write bad code
  - Good program design is part of the craft of software engineering, which is something to tackle only once the mechanics of coding are automatic
- 3) Most students cannot grasp an idea presented only in words
  - My physics class had been successful with the same students because it relied on picking the perfect concrete model from which to gradually learn an abstract idea

## **My new vision**

- 1) Present ideas through a model, with words only supporting that model
- 2) Show how an expert imagines the program running
  - Student previously had to understand an abstract idea first, so that they would be able to imagine how a program operates, so that they could write that program
  - By giving students a precise animated model of how the program runs, we allow them to experiment through trial and error as they gradually assimilate the idea
- 3) Lots of small exercises, giving opportunity to practice and master each concept
  - Provide automatic feedback from the computer to help student reach the answer
- 4) Structured, self-paced learning, allowing the student to choose what to tackle next but having the computer ensure they have mastered each topic before moving on
  - Goal is to not permit a student to get in over their head, even if that means that some students will take five times as long to learn a topic
  - By having the computer structure the learning, present ideas, and evaluate student work, the teacher is freed up to wander around helping frustrated students

## **Try out CodeMotion!**

- 1) Free beta version, tested with kids and teachers
- 2) Teaches all those hard, “basic” ideas
- 3) Covers the first few months of APCS; will eventually be developed into a full course
- 4) Try it with your class, your children, or yourself!

[www.zahniser.net/~russell/codemotion](http://www.zahniser.net/~russell/codemotion)