# LIVING IN A COMPUTING WORLD: A STEP TOWARDS MAKING KNOWLEDGE OF COMPUTING ACCESSIBLE TO EVERY STUDENT

■ Jody Paul ■

**The new CS Principles course provides an opportunity to advance the goal of making knowledge of computing and computer science accessible to every student. A pilot course used to inform the development of the CS Principles curriculum framework enabled its instructor to explore alternative pedagogical practices in pursuit of that goal. This article presents observations and reflections of the instructor with respect to these attempted practices.**

## 1 Introduction

*Living in a Computing World (LiaCW)* was an early (Fall 2010) pilot course offered at Metropolitan State College of Denver, supported by a grant from the National Science Foundation, and intended to inform the design and specification of the CS Principles course. [1] It also afforded an opportunity to try alternative pedagogical practices in pursuit of the goal of making knowledge of computing accessible to every student. What follows is a collection of observations and reflections concerning various attempted pedagogical practices in an unrestricted open-enrollment environment with the intent of improving engagement and learning.

Metropolitan State College of Denver (MSCD, http://www.mscd.edu) is a public, open enrollment, non-residential college attended by 24,000 undergraduate students, 94% of whom are from the metropolitan Denver area. It is ranked in the top 100 schools in the USA for graduating Latino students and students of color. MSCD has a statutory mission that includes *providing access for underserved and low-income students*. [2]

Enrollment for *LiaCW* was fully open, the only requirement was that the students be *college ready*; that is, students had to be *minimally eligible to enroll* in College Algebra 1 and English 1. *LiaCW* did not satisfy any degree requirements other than college-residency units. The course thus had no prerequisite courses, co-requisite courses, nor was it prerequisite to any other course.

As an instructor saddled with over 30 years of experience using primarily traditional practices, the thought of adopting a student-centered agenda and approach raised questions and insecurities.

## 2 A Student-driven Agenda and Opportunistic Approach Engages Students in Learning Fundamental CS Concepts

The basic idea of student-centered learning was articulated in the early 20th century by the likes of Dewey [3], Piaget [4], and Vygotsky [5], and the subject of renewed attention in the past couple of decades. [6, 7, 8, 9, 10], Yet it always seemed too daunting to put into actual practice when there was limited time and a fixed curriculum that must be covered. The pilot offering of *LiaCW* was a lower-risk opportunity to venture into sharing responsibility with students because *LiaCW* needed to satisfy no specific degree requirement and was itself a prerequisite to no other course at MSCD.

Still, as an instructor saddled with over 30 years of experience using primarily traditional practices, the thought of adopting a student-centered agenda and approach raised questions and insecurities. Wasn't I responsible for establishing what students should know? Wasn't I the expert on what students should do and when they should do it? Wasn't I supposed to teach the information? Would I be helping students learn if I wasn't actively teaching? Truth be told, by teach I really meant lecture.

During the first day of class and at several times in the semester, students were prompted to provide items of individual and collective interest. Typical prompts included, "things you've wondered about," "what you'd like to know," and "what you would like to be able to do." Students were initially hesitant with this unfamiliar approach. They were unclear as to the parameters, for example, asking, "Does it have to be about computers?" (Short answer, "No." See later discussion about making connections between the world and fundamentals of computer science.) They also seemed unsure whether this was "for real" or just some teaching tactic.

As with many later class activities, the first few minutes were devoted to individuals coming up with ideas on their own. Next, they shared, refined, and created more items in pairs. Pairs were combined into larger groups with more sharing and discussion. Finally, the groups shared their collections of items with everyone. These items formed the surface agenda for the course, establishing the domains and examples used to illustrate and explore the ideas and practices from the CS Principles curriculum framework.

In addition, vigilance to happenstance and current events provided numerous opportunities to leverage active interests and curiosity. Such events permeate our lives and they were never in short supply. (One particularly sad example was when a student's laptop was stolen from home. This proved to be of great interest and was a rich source for addressing a wide range of concepts in the curriculum framework.)

With this student-driven and opportunistic approach, it was obvious to students that their interests were being directly addressed. Their level of engagement was high as these were the very items in which they had expressed intrinsic interest. As the semester progressed, students became quite comfortable with adding, deleting, and tweaking the list of items. They also readily expressed feelings as to whether a topic needed additional treatment or had been adequately addressed.

Would this approach support covering the intended material? Perhaps because the CS Principles concepts are indeed fundamental, it turned out that the everyday world in which we live—students included — naturally provides vehicles for exploring and exercising those fundamental computer science concepts. In fact, there were always many CS concepts that could be associated. There was no difficulty with identifying some appropriate linked concept; rather, the chore was to determine which of the many concepts best fit the particular context.

Having identified the concept to address, my role as instructor then involved (a) helping to bring the concepts to conscious attention within the given context, (b) illuminating the connections between those concepts and other aspects of the world, and (c) providing the association of the concepts with the jargon that is the hallmark of one knowledgeable in the field of computer science.

Preparation and lack thereof contributed most to successes and failures in applying this student-driven and opportunistic approach. Having prepared for the semester by collecting and creating a number of ready-to-go activities, it was relatively straight-forward to connect them with the current topic or tailor them to the current domain as appropriate. This allowed the expressed interests and curiosities of students to drive what was addressed. Coupled with opportunistic spontaneity, this turns out to be a fine way to proceed… but only if you have a large quantity of activities prepared in advance. Alas, the amount of effort required to put together such activities was too great to apply this approach to the extent desired. Sometimes I was simply not up to that level of preparation and students found themselves sitting through a lecture presentation. I think students found those lectures to be relaxing, comfortable, and neither as engaging nor as demanding on their attention as the active-learning experiences.

## 3 Don't Interfere with the Natural Learning Process

Early on in the course, I discovered two sure ways to inhibit these students' receptiveness to learning some computer science concept. The concept might be anything at all in the curriculum framework: Boolean logic, programming, privacy, recursion, etc. The loss of receptivity typically occurred in the first few seconds of the introduction of a new topic. Phrased as lessons learned about avoiding virtually guaranteed negative outcomes:

- Do not begin with a numeric example.
- Do not begin by naming the concept and stating that's the next item to be learned.

Body language and facial expressions were clear giveaways that I had lost many, most, or all of the students. The triggers were seemingly sufficient to establish the upcoming experience as *less-accessible* (perhaps *hard to learn* or *uninteresting*).

If I began with a numeric example, a well-established anti-receptiveness trigger was activated. This trigger was so strong and deeply rooted that it had to be circumvented. Note that I had not associated this trigger with our existing computer science majors (all of whom are required to have math minors) and computer science minors (most of whom are math majors) who already have intrinsic interest in the field. The problem is not trivial to address because the available materials and textbooks generally make extensive use of numerically-based examples.

If I began by naming the concept to be learned, the disconnected setup appeared to be taken as an indication that something difficult lay ahead. This also contradicted the intent to establish a cognitive mode conducive to learning in that there was no preparation that would bring to consciousness some extant knowledge or a memory with which to relate.

I began with such poorly chosen starts several times during the semester (much to my chagrin). It appeared that once an anti-receptiveness switch had been thrown, it was all but impossible to get back on track in real time. Stoically pushing on and hoping for the best did not address the problem and was an ineffective use of the time.

Fortunately, I also discovered a simple recovery mechanism: abort the lesson as quickly and graciously as possible. For example, if I had already started writing some numbers and then observed the crumpling effect on students, briefly doing something else with the numbers would bring rapid closure to the episode (much to the relief of the students). Similarly, if I had ruined the environment by providing the concept label before activating an internal context, I could readily abort by saying, "on second thought let's do that another time" (again to the apparent relief of the students). Having closed the negative session, it was now fine to start something new, either a different topic entirely or by using an introductory example that came from a student-centered domain.

Although I wanted to be the one to teach students new concepts, it turned out that my overt attempts to do so really just got in their way. One structural aspect of this I've already covered, naming what they are expected to learn next. But there was an even more important lesson for me in this context:

➜ Students already know many "computer science" concepts.

The computer science concepts in the CS Principles framework [11] appear to be so fundamental that most people already know them, albeit not in a CS context. Instances of the concepts themselves are everywhere to be found in the real world, presumably because that is from whence they derive. There were no concepts in the curriculum that were completely disconnected from students' personal experiences and experiential knowledge.

Students generally did not realize these concepts explicitly. Nor did they know the computer science jargon. So, an effective approach was:

➜ Activate each concept prior to labeling or formalizing it.

That is, first providing students with triggers from real-world situations that would remind them of the concepts. Once the familiar concepts were thus activated, they could then be expressed and demonstrated in computer science terms.

This is but another side of the same coin of not labeling or attempting to formalize a concept before its recollection or acquisition. Concept activation refers here to engaging in an activity that results in related information being transferred from long-term memory to consciousness. Such activation appeared to be the key to the acquisition of the intended curriculum framework content.

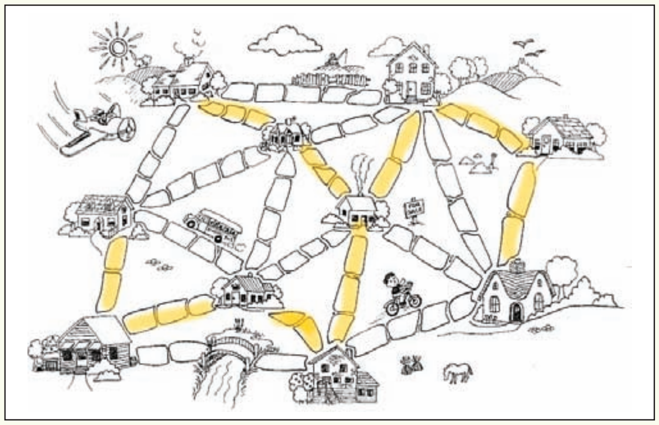
Figure 1: Students demonstrated pre-existing knowledge of advanced concepts and algorithms when activated and addressed in familiar real-world contexts, such as *minimal spanning trees and Kruskal's algorithm* situated within this "muddy town" activity from *CS Unplugged*

Although introducing concepts by thought experiments sometimes works, they were not the most reliable or effective. Rather,

➜ Tangible self-directed discovery experiences work well for introducing CS concepts.

The greatest successes resulted from activities that leveraged curiosity in tangible contexts. An example was the "learning" of programming concepts without any explicit instruction through the use of LightBot (a software game in which the player solves a series of puzzles by instructing a virtual robot). Pairs of students worked together to play the game and clearly demonstrated that they already possessed the ability to write, test, and debug programs that used logic, sequences, function calls, recursion, and more; all without any specific preparation and without intervention by the instructor. The game play served to activate the concepts. During the activity and in the associated reflective period, the appropriate computer science terminology was first casually introduced to label the associated concepts, then formalized.

Figure 2: Tangible manipulatives (such as these building blocks) supported concept activation.

## 4 Successful Group-work Arises from Modeling and Facilitation

Effective group-work was one of the six computational thinking practices in the CS Principles curriculum framework in the fall of 2010 (subsequently that thinking practice became *Collaborating*). Students cringed and exhibited other displays of distaste when I first stated that there would be a much group-work associated with the class. Indeed, students consistently report that although they had classes (K-16) in which they were required to work in groups they had received no instruction with respect to how to work in groups. Apparently, group-work techniques and skills had been treated as though they were innate or would be acquired through *non*-directed experiences. From their previous experiences, most students appeared to have learned that they intensely dislike working in groups.

As with the inappropriate use of numeric examples, this pre-existing negative trigger needed to be circumvented and hopefully attenuated. A first step was to offer an alternative interpretation to students: that they simply had never had the opportunity for positive group-work experiences.

The intent of *LiaCW* included helping students achieve the ability to engage in productive group-work themselves. Here again, rather than treating this as a subject of theoretical study or lecture presentation, we used active learning experiences. These included modeling, simple instruction, and short duration participatory activities. Modeling the intended behaviors and actively participating in a group activity with clear expectations and instruction always came prior to explanation or labeling. This is consistent with the learning approach adopted for the other aspects of the course and seemed to stimulate the natural curiosity and explanation ability of students. The result was that students appeared very engaged in the learning of teamwork practices.

As with all activities, after engaging in an experience, students were given the opportunity and requirement to reflect on what had just happened and share the observations and insights explicitly. With respect to teamwork practices in particular, they were asked to identify *what worked well* and *what didn't work so well*. These were often couched in terms of identifying for a future project or event "what was useful and we should do again", "what should we do more of", and "what should we change."

We commonly followed the following sequential progression: solo-preparation, pair-work, quad-work, and group-share. Note that including pair-share virtually ensured that every student was engaged. When working singly, a student could easily disengage from the current task. When working in groups of three or more, a student could also adopt a non-participatory role. However, when the situation involved interacting with just one other person, it was difficult and uncomfortable not to be involved. Thus including pair-work in the process ensured that the level of engagement would be high. (Of course, it was still possible for both students to be off-task.)

Well before the middle of the semester, students not only accepted the prevalent practice of group-work but also indicated that they looked forward to it. It became the norm and a comfortable expectation. Since the size of the class was small and there were three facilitators (instructor and two community assistants), there was ample opportunity for real-time assessment and cognitive coaching.

Figure 3: Most class time was devoted to small-group activities.

## 5 Reflection Helped Everyone

Students were required to reflect explicitly on each activity and to report observations, insights, and key ideas. Reflections were oral, written, or both for the same event and might occur immediately following the experience, as an assignment for that same evening, several days after the experience, or a combination of these times.

Expressing the thoughts resulting from cognitive reflection worked well to help students recognize that they had learned something that they themselves valued. The acts of reflection and explicit expression may have also added to the learning itself. In addition, such expressions helped instructors identify what students knew and believed with respect to the course content and logistics, and to gain better understanding of the students' experiences and concerns.

As part of the process, I also reflected on each activity and each day of class. These were shared with the community assistants (who likewise shared their observations and insights with me) and in an abbreviated form with the students. Such reflection and sharing was very useful in planning and conducting later activities, although especially humbling in those cases where my perception was at odds with that of everyone else who had participated in an experience.

## 6 Lessons Learned

Here are nuggets that summarize what the experience with *Living in a Computing World* taught me.

- A *student-centered* approach and agenda really does engage students.
  **Let students determine the surface agenda (domains for the examples).**
- Use examples from domains that promote students' receptiveness.
  **Learn and avoid using what alienates and hinders the receptiveness of the group of students (e.g., numeric-based first examples).**
- Employ engaging modes of self-expression.
  **Provide opportunities to create web-accessible artifacts (*e.g.*, *Xtranormal*, *Scratch*).**
- Students already know many "computer science" concepts.
  **Reacquaint them with the concepts from life in the real-world.**
- *Activate* each concept *prior* to labeling or formalizing it.
  **Provide the activation experience first; afterwards express it in computer science terms.**
- Tangible experiences and self-directed discovery work best for introducing concepts.
  **This even works for programming as evidenced by using *Light-Bot* and *Scratch*.**
- Model, mentor, and facilitate collaborative group-work.
  **There is a body of knowledge that supports successful team-work practices.**
- Reflection helps everyone: students and instructor.
  **The process of reflection, coupled with explication and sharing, benefits all learners in the environment — including students and instructors.**
- Use existing lessons & materials.
  **See the list of sources below. Especially useful for *LiaCW* was *Computer Science Unplugged*.**

### References

[1] See http://csprinciples.org

[2] Colorado Department of Higher Education. (2011) FY 2012-13 Joint Budget Committee Hearing, 19 December 2011. http://www.state.co.us/gov_dir/leg_dir/jbc/2011-12/hedhrg.pdf

[3] Dewey, J. (1916) *Democracy and Education: An Introduction to the Philosophy of Education*, New York: Macmillan.

[4] Piaget, J. (1926) T*he language and thought of the child*. London: Routledge & Kegan.

[5] Vygotsky, L. S. (1977, originally published 1926) *Educational Psychology*. Boca Raton: St. Lucie Press.

[6] Rogers, C. R. & Freiberg, H. J. (1994) *Freedom to Learn*, 3rd edition. Upper Saddle River: Prentice Hall.

[7] Barr, R. B., & Tagg, J. (1995) "From Teaching to Learning — A New Paradigm for Undergraduate Education." *Change, 27*(6), 12–25.

[8] McCombs, B., & Whistler, J. S. (1997) *The Learner-Centered Classroom and School: Strategies for Increasing Student Motivation and Achievement*. San Francisco: Jossey-Bass Publishers.

[9] Pedersen, S. & Liu, M. (2003) "Teachers' Beliefs About Issues in the Implementation of a Student-Centered Learning Environment." *Educational Technology, Research and Development*, 51(2), pp. 57-74.

[10] Estes, C. (2004) "Promoting Student-Centered Learning in Experiential Education." *Journal of Experiential Education, 27*(2), pp. 141-161.

[11] The College Board. (2011) *Computer Science: Principles; Computational Thinking Practices; Big Ideas, Key Concepts, and Supporting Concepts*. See http://www.csprinciples.org/home/about-the-project

**JODY PAUL**
Department of Mathematical and Computer Sciences
Metropolitan State College of Denver, Denver, Colorado 80204 USA
*jody@acm.org*

**Categories and Subject Descriptors:** K.3.2 [Computers and Education]: Computer and Information Science Education – Computer science education, Curriculum
**General Terms:** Experimentation, Human Factors, Design, Management, Measurement
**Keywords:** Computer science education, pedagogy, CS Principles

**DOI:** 10.1145/2189835.2189860 © 2012 ACM 2153-2184/12/06 $10.00

## Sources for Information, Materials, and Tools

The following are links to sources used for the LiaCW course pilot.

**Computer Science: Principles**
*http://csprinciples.org/*
**Computer Science Unplugged**
*http://csunplugged.org/*
**Exploring Computer Science**
*http://www.exploringcs.org/*
**Computer Science for Fun**
*http://www.cs4fn.org/*
**Computer Science Teachers Association**
*http://www.csta.acm.org/*
**Xtranormal**
*http://www.xtranormal.com/*
**Kompozer**
*http://kompozer.net/*
**Scratch**
*http://scratch.mit.edu/*
**LightBot**
*http://chat.kongregate.com/gamez/0002/2915/live/BillBotKong.swf*
*http://chat.kongregate.com/gamez/0008/3984/live/Lightbot2.0Kong.swf*
**Can Animals and Machines Be Persons?**
*http://amzn.to/yABPTw*
**The Five Dysfunctions of a Team**
*http://www.tablegroup.com/books/dysfunctions/*