Effective Pedagogical Practices in the Computer Science Classroom
PANEL PRESENTATION

Chris Healy
(moderator)
Furman University
Greenville, SC
chris.healy@furman.edu

Andy D. Digh
Mercer University
Macon, GA
digh_ad@mercer.edu

Paula Gabbert
Furman University
Greenville, SC
paula.gabbert@furman.edu

Michael Verdicchio
The Citadel
Charleston, SC
mv@citadel.edu

ABSTRACT

This panel of presenters, with over 80 years of teaching instruction between them, will share with attendees a variety of actual tools, toys, and/or visuals that have helped students understand concepts in the programming sequence, i.e. CS 1, CS 2, CS 7 (algorithms) classes. These are based on moments in the classroom when students have reported that new or difficult concepts were made easier to learn.

INTRODUCTION

Background

The book *Models of Teaching* provides evidence from educational theorist David Ausubel that any "new ideas can be usefully learned and retained if they can be related to already available concepts or propositions that provide ideational anchors." In addition, the book further illustrates that "when we help students acquire information and skills, we also teach them how to learn as well as how to think critically, compare, and apply new knowledge" [7]. This offers an important reminder that when providing additional connections or hooks for students in our course, it makes the course outcomes even stronger. As David Gooblar notes in *The Missing Course: Everything They Never Taught You About College Teaching*, "the more our brains work while acquiring knowledge, the better they are at retaining it" [5]. In addition, any time you use schemas to help a student "think about something they have previously learned or experienced," you help them transfer it from working memory into long-term memory, which is their "seat of understanding" [3].

Panel Structure

After brief introductions, each of the four panelists will have up to 15 minutes to focus on several effective pedagogical practices they have used to engage students and enhance student learning. Actual demonstrations involving visual, auditory, and tactile tools will be given. Most importantly, they will highlight having fun in the classroom while teaching computer science. In the last 30 minutes, audience members will be encouraged to comment on any similar times when they have watched "the light come on" with their own students in the classroom. They will be able to share any interesting or innovative

methods of their own “to simplify and clarify complex topics” as well as ensuring that what is said will easily be understood or remembered [1].

CHRIS HEALY, FURMAN UNIVERSITY

Chris Healy has taught computer science at Furman University since 1999, including all courses in the core CS curriculum.

- Sandwich-making pipeline simulation

  One vital topic in computer organization is pipelined instruction execution. To introduce this in a fun way, I created a make-believe sandwich shop during class. Five student volunteers are each responsible for one stage along a sandwich-making assembly line:

  Fetch: Get the next customer's sandwich order
  Bread: Obtain the desired type of bread and slice it open
  Meat: Lay slices of the desired meat along the sub
  Condiments: Top the sandwich with the desired condiments
  Pay: The customer pays for the sandwich

  Other students in the class submit sandwich orders, and observe the progress through the pipeline. Students see how a pipeline is filled, and how throughput is improved over one person doing all the work. Then, hazards are introduced, in order to foreshadow concepts of structural and data hazards. Examples of pipeline stalls include:

  - During Fetch, there could be no customers entering the delicatessen, analogous to an instruction cache miss
  - During Condiments, the order might ask for all the condiments
  - During Pay, a customer might have to write a check

- The liar puzzle skit

  This is a discrete math activity for enjoying truth tables. The liar puzzle concerns a venturing pilgrim who comes to a fork in a road. One path leads to his destination, while the other leads to disaster. Twins guard the path entrances. One twin always tells the truth and the other twin always lies. The pilgrim can only ask one question. This activity requires three student volunteers to play the part of the pilgrim and the twins. They play out each scenario to figure out the outcomes, and the instructor records the results in the form of a truth table on the board [4].

- Making flag images in Java

  This activity stresses image output and nested loops for CS 1. With very little code, it is simple to write a Java program to create a color image that depicts geometric patterns, such as one we can find on a flag. Therefore, this is an ideal in-class participation exercise, where students assist the instructor in finishing the implementation. Students figure out simple formulas to express the boundaries between the colors.

  Suitable national flag examples include France, Germany, the Czech Republic, Japan, and even the UK. Having finished the Union Jack, the instructor can mention that several other flags incorporate this design. Finally, we discuss why some other flags would be much more difficult to draw. One humorous snag is that if the code is written carelessly, the flag may become

upside-down or backwards. The instructor might want to intentionally write the nested loops or geometric formula incorrectly to see how students figure out the mistake.

- Countdown numbers game
  The TV game show "Countdown" features a numbers game that goes like this. You are shown a three digit number (called the target), plus six smaller numbers. Your job is to write an arithmetical expression using some or all of the six small numbers that equals the target. For example, the target could be `770`, and the small numbers at your disposal are `1`, `4`, `4`, `9`, `10`, and 100. A possible solution is `10 * (100 - 9 * (4 - 1) + 4)`. Students are surprised to find that this task, requiring only basic arithmetic, is sometimes easy, sometimes difficult, and sometimes impossible [6].

  This game can become the basis for various activities in CS 2, CS 7, or discrete math. For example, we can ask how many different binary trees could represent an expression. The answer turns out to be a sequence of numbers that the students are unlikely to have seen before, so I can introduce them to the useful Online Encyclopedia of Integer Sequences.

ANDY DIGH, MERCER UNIVERSITY

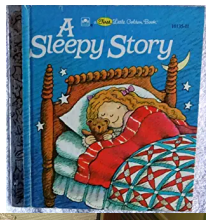
Andy has been teaching computer science at Mercer University since 1998 where he teaches CS 1, CS 2, and CS 7 every year. The following are some of his favorite toys and visuals for helping students grasp content and making learning extra special.

- Cups for the Swap Algorithm. In CS 1, to teach how to swap the values in different variables, red solo cups are filled with different beverages in cups X and Y. The third temporary cup is kept out of sight. You challenge students to think critically about how to swap the contents of the two cups. Later in the course when teaching sorting, you can connect back to the swap algorithm. You will discover that some of them may later even want to use a temporary variable named “cup.”

- Different Hats & Props. Good teaching is often a play of sorts. In CS 1, using two different hats for programming the building of a class versus programming using a class. When wearing the construction hat, you do care about your private instance variables and how your data structure is built. However, if you take off that hat and put on your derby hat as the client, you do not care how it is built and merely want to use the public methods of the class. Students are challenged to always remember, “what hat they are wearing” when working with classes in the future.

- Potato Head. In CS 2, using a Potato Head toy is a great way of teaching inheritance. The plain spud with no attachments is the base class. When you add on “enhancements” to the base class, you are setting up derived classes. Students are often confused about when and why you should set up inheritance, and this helps them see it as being built upon an existing class.

- Recursive Golden Little Book. In CS 2, the children's book *A Sleepy Story* brilliantly introduces recursion by having a mother read a bedtime story to her daughter in which six different animals hear the same bedtime story from its mother. Then, the sixth animal goes to sleep, and each of the animals goes to sleep in reverse order. At the end, the little girl falls asleep [2].



- Recursive Nesting Dolls. In CS 2, Russian nesting dolls can help explain how recursion actually works. This aids in the understanding of how parameters and local variables will be implemented within different methods.

- Real World Stacks & Queues. In CS 2, there are so many nice real world examples of stacks such as Pez candy dispensers, cans of tennis balls, and baby stacking rings to illustrate how last in, first out order works with all insertions and deletions coming from the top. For queues, toy cars lined up at a drive thru window can help students better connect to how first in, first out order works. Often, students struggle at the implementation level where you must keep track of pointers to the front for deletions and to the rear for insertions.

- Connect Four Game & Storage Cabinets. In CS 1 or CS 2, a connect four game and/or storage cabinets matrix boxes can help introduce multi-dimensional arrays, explain row/column major order, and work with coordinates. This is also a great way to help them think about the code for printing a matrix in different orders. The storage cabinet is perfect too in CS 7 when teaching the adjacency matrix representation of graphs.

- Barrel of Monkeys & Slinky. In CS 2 or CS 7, a barrel of monkeys for explaining linked lists and a classic Slinky toy to explain how array lists can easily expand to accept new elements and contract when elements are removed.

- Musical Sorts. In CS 7, YouTube provides a wealth of great sorting videos and "audiblizations" that help students visualize 15 different sorts along with their complexity and number of comparisons. There is a fantastic one on selection sort featuring Romanian dancers.

PAULA GABBERT, FURMAN UNIVERSITY

Paula Gabbert has been on the computer science faculty at Furman University since 1996. She has taught the CS 1 and CS 2 courses under a variety of approaches and programming languages. The following ideas were used to accommodate remote learning in the most recent offerings.

- Zybook - interactive textbook encourages students to actually *READ* the text
    - Overview of text philosophy: less text and more interactive activities to apply concepts
    - Example activities with integrated development environment in the text
    - Benefits for students: instant feedback, engaged reading, focused reading, additional low stakes programming practice

    - Side benefits for faculty: configurable text, built-in lab activities, presentation mode to discuss examples in class, real-time usage and completion statistics by student
- Repl.it - cooperative programming or "google docs for programming"
    - Overview of repl.it philosophy: make programming accessible to all, supports numerous programming languages and web development
    - Overview of Teams for Education functionality
    - Benefits for students: "real" pairs programming, easily sharing code with professor for feedback, code is accessible anywhere (class, lab, dorm), one location for assignment distribution, work area, and submission.
    - Benefits for faculty: some testing and grading support, very inexpensive

MICHAEL VERDICCHIO, THE CITADEL

Michael Verdicchio started "teaching" CS1 as an undergraduate lab assistant in 2002 and has been hooked ever since. He is now an Associate Professor in the Department of Cyber and Computer Sciences at The Citadel where he has taught CS 1, CS 2, CS 7, software engineering, and other courses since 2011.

- Giant, oversized playing cards along board to demo array searching/sorting algorithms in CS 1
    - Cards can be lined up in the marker tray across the board and index variables drawn above.
    - Good for slo-mo demonstrations of selection sort, insertion sort, etc.
    - Cards can be flipped facing the board to show binary search, where only the card at the midpoint is revealed to guide the next direction of the search.
    - Students can also be used as index variables for fun.
- Regular playing cards split and passed back (then forward) row by row to manually implement merge sort in CS 2 / CS 3
    - Emphasizes the "amnesia" of a recursive method call and shows the power of delegating subproblems in divide-and-conquer algorithms.
    - For merge sort, for example, all the student has to do is cut their deck and pass them back to two people and say, "Here, sort these." On the way back, all the student has to do is merge the two sorted stacks and pass them forward.
- Physical towers of Hanoi for recursion lectures where one student takes the bottom disc and just hands *(n-1)* discs to the next person.
    - Like above, this emphasizes the amnesia of recursive method calls simply solving the instance of the problem they wake up.
    - "How do I solve this tower? Easy. Give the top *n-1* discs to the person behind me; tell them it is now their problem, and then move the bottom disc.
    - Many metaphors for micromanagement and "lazy" work habits here. Especially in my military college where students "delegate" many menial tasks to freshmen.
- Interactive coding demos in CS 1 / CS 2
    - This is a good way to enact the following teaching strategy:
        - I do, you watch
        - I do, you help
        - You do, I help

        - You do, I watch
    - My students, especially in CS 2, can read slides/books on their own, and we have found that the class time is better spent with interactive coding demos at least once per week.
    - I prepare demos that parallel their programming assignments, so they have some ideas to base off. If I do not do this, they will be on YouTube looking for other people narrating their code, but usually way out of scope from what we are doing.
    - Zoom/screencasting makes it easy to record and archive these demos for posting in the LMS for later review.
    - I am excited to try the new feature, "Code With Me", from IntelliJ IDEA:
        - This will facilitate pair programming for group assignments, and I can use it in the classroom so students can see my demo right on their screens instead of squinting at the board.
- Literal stack of paper plates for illustrating call stacks/frames/local references in method calls for CS 1 / CS 2
    - Teaching call stacks is critical when learning methods/functions, and important to review when teaching recursion. A literal stack of anything you can write on helps to cement these concepts.
    - I have used giant notecards or paper plates. For each method invocation, I write the values of the local variables with a sharpie and stack it on top of previous method calls. When the return statement is reached, I dramatically throw the card out into the room, revealing the one underneath. This works especially well with a document camera overhead.
- Animations and visualizations for algorithms from websites like visualgo.net, algs4.cs.princeton.edu
    - Before I teach the pseudocode/theory for searching/sorting algorithms, I usually "preview" with a clean animation to give the initial sense of how it works. Then we get into the pseudocode/real code, and then review the animation.
    - Animations are also good for showing loop invariants.
    - It is easier to watch the same animation two or three times than it is to manually trace it on the board.

ESTIMATED PANEL TIME
90 minutes

ACKNOWLEDGEMENTS
Andy wishes to thank Michael Berry of the University of Tennessee at Knoxville who first showed him cups for swapping while he was a teaching assistant. Also, thanks to his colleague Laurie White from Mercer University who shared with him the idea of wearing a construction hat when teaching how to create a class. In addition, he wants to praise many dear colleagues through the years from Advanced Placement Computer Science readings who shared teaching strategies.

Michael wants to extend thanks to Kevin Wayne and Robert Sedgwick from Princeton University for their amazing work in *Algorithms*, 4th Edition, and its companion site, https://algs4.cs.princeton.edu. Also,

thanks to Armando Fox and David Patterson from UC Berkeley for their technical and pedagogical contributions to Engineering Software as a Service (http://www.saasbook.info).

REFERENCES

[1] Bain, Ken. *What the Best College Teachers Do*. Harvard University Press, 2004.

[2] Burrowes, Elizabeth. *A Sleepy Story*. Golden Books, 1982.

[3] Carr, Nicholas. *The Shallows: What the Internet is Doing To Our Brains*. W.W. Norton & Company Publishers, 2020.

[4] Francis, Dick. *Decider*. Putnam Publishing Group, 1993.

[5] Gooblar, David. *The Missing Course: Everything They Never Taught You About College Teaching*. Harvard University Press, 2019.

[6] Healy, Chris. "A Mental Game as a Source of CS Case Studies," *Journal of Computing Sciences in Colleges*, May 2017, pp. 11-16.

[7] Joyce, Bruce. *Models of Teaching*. Allyn and Bacon Publishers, 1992.