Tommy Gober

Direct Instruction Review

Pg 104, #4

Because quite a bit of my computer class lessons are based on Direct Instruction I think it best to self-review one of my recent lessons. The class began as usual, students file in and get to their assigned seats and will usually get right on the computer. This can generally serve as a focus activity and is beneficial both for the class and myself. After the bell rings, I will usually stand at the class podium with my wireless keyboard and the lesson’s PowerPoint on the screen behind me. At this point, the students know class is about to begin and will usually begin to close out of any games they may be playing (or simply minimize the screen, which I don’t mind.)

I will usually ask one or two of the students to “remind us what we did last time”. This helps the class recall what was done, and lets me know what has been retained. After filling in any gaps or giving them a few clues, I will announce what our daily goal is, or what kind of problem we will be dealing with. This allows students to know where we’re going and what to watch for. If they will be given a worksheet, I will let them know; if they will be writing their own program, I will announce that as well. Some days are spent building up in preparation for the next class where a more complex problem will be solved. (For example, if a program will be assigned the next day in class, I will let them know this and let them know that the lesson will be oriented toward helping them with the coming assignment. This is typically enough to let them know to take notes.)

After stating the purpose of the lesson, we will either pick up where we had left off of a previous program, or begin anew. I will generally begin programs on the screen for the class to follow along. If we are going to make a program to solve a problem, we will get the necessary prep work out of the way then take a break to think aloud how we might solve this problem. At first, the class did not share their ideas very much and I only had participation from a few. I gradually eased into “raise your hand if you agree with Adam”, “raise your hand if you agree with Greg”. From voting we moved into vocal agreement from the class and the offering of individual ideas of how something would work or not work. Of course, it took some modeling, but they soon caught on. We’ve more or less weeded out the “no that’s dumb” response or the blank “I dunno”. All hypotheses are welcomed, they just need to be explained or defended if asked.

After a brain storming session, we will write down the “best” solutions on the board to explore them further. I will generally pick the right one and a few of the most plausible solutions. (Sometimes there are more than one solutions, in which case those are all explored equally and students may decide which they feel is easiest/most efficient/most reliable. After exploring the “winning” solutions we will settle on an approach and determine what is necessary to make the program run how we intend. After making a list of requirements I will allow students to write the code necessary. At this point I will typically walk around the room helping students troubleshoot their code and answer any “I can’t remember how to…” questions. Allowing students to pursue their own solutions has backfired on me several times and required me to answer hundreds of questions over the course of the class. This is perhaps the most stressful time of class for me. Sure, each student may only need help a couple of times during class, but that number is multiplied for each student in class. If too many divergent paths are being taken and I can’t keep up, I will sometimes try to consolidate our problem-solving approaches by writing the basis of my own solution on the screen. *Somehow*, this will tend to focus the number of solutions around my own. The amount of variance in a student’s work is inversely proportional to the amount of code I write on the screen.

Of course, by allowing students to do their own work, I allow them to freely explore their capabilities and allow them to individually come up with solutions to their own problems. I try to be as hands-off as possible during this time. By allowing them individual time, I can best assess their progress as well as point out novel approaches to the rest of the class and allow them to be openly praised for such critical thinking. I think I’m most proud when they blind-side me with a great solution to a problem that works better than what I had planned. Also, if several of the students encounter similar problems or hang-ups, I will usually address the class “So it looks like a few of you are having a problem with…”, several solutions are voiced and the process begins again: *What will it take to accomplish this? Where did we go wrong? Why doesn’t \_\_\_ work as intended?* This open forum approach to learning to program both allows the class to be fluid and keeps students involved, but it also fosters a programming discipline that is encountered both in the college classroom as well as the workplace. Problems are not tackled individually, but as a team. Code might be written individually, but the group-think process helps immensely. It also creates a can-do attitude. When a student is stumped, they don’t get frustrated and give up, they will turn to one another for guidance. Peer-assistance helps both students further understand the concepts in class.

Unless the class finishes their work early, I will let them work right up until the class lets out. I generally will take the last 5 minutes before the bell to let them save their work, print off anything they need to turn in and let them know if we’ll be continuing this for the next class. By this point the problems they’ve all encountered in class have been brought out into the open and there is rarely a stone left unturned. I’ll generally close with “so, that’s how you do \_\_\_\_\_\_”. I could probably improve my summarizing at the end of class, but the exploration has always been my goal. In a class like programming, so much of the class is centered around learning the building blocks of code necessary to perform meaningful actions. (Similar to how a foreign language class may operate.) By basing their assumptions on what they already know, students are free to explore what can be or what is possible.