# Commentary: Week 2

CSE120: Computer Science: Principles

The week’s goal is to introduce The Processing Visualization Language.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lecture 4 | Lab 3 | Lecture 5 | Lab 4 | Lecture 6 |
| Assignment 4 | | Assignment 5 | | Assignment 6 |

**Lecture 4:** The goal of the lecture is to apply by analogy what was learned last week in the context of Processing (and, of course, introduce the language). I start by simply showing the snow angel code, and then running it. I point out that we just have a few instructions, and that already we can create something *from scratch* that is fun to play with. I then program it. (Note, the demo code and the code I program are different: the latter has a diagonal line from the corner … I don’t mention it, but it will come up as a question. Explain what’s happening, but don’t try to teach this yet.) Call attention to the parts of the program – apply last week’s analogy. Go over color. Go over the RF Robot, discussing what they see, emphasize the role of comments. Finally, color the robot.

**Assignment 4:** The first programming assignment is creating the Simpsons. This is a fun assignment and students can do it, but it requires persistence. The assignment is intended to be detailed, but to require a sequence of small issues to work out. Virtually all students are successful – it’s a red flag if they are not. The extra credit allows them to do another – many students do so, and this is a good time to get them to work extra. NB, I don’t give credit for the Ninja Turtles … they’re just too easy and we’ll use them in examples a lot.

**Lab 3:** The goal, besides just giving more experience programming in Processing, is to highlight the difference in Static vs Dynamic programs. RF don’t do this this early, but we move fast, and completely understanding dynamic behavior is critical for assignments in the near future. The code RF robot code is included in this directory.

**Lecture 5:** It introduces variables, declarations, initializations and value modification. These are big. With my tight time constraints, I have to rip through these(lec5.ppt), and plan to emphasize them repeatedly in lab and subsequent lectures. With more time, I would do this lecture in two parts, given here as lec5a.ppt and lec5b.ppt. (I use the first of these and an example from the second using min( ).) I recommend the two part approach. After explaining the concepts being introduced, the programming is the most of the lecture(s), and I just use this as time to patter about the concepts, what I’m doing and how it works. This is a lot of fun, and engages the students well. The main goal is to get through to min( ) so they can do Assignment 5, which is their first use of these ideas.

**Assignment 5:** The assignment builds on the Simpsons work of Assignment 4. (When students see this code run, they hum the opening music of the Simpsons.) Because some students might not have completed Assignment 4, a “basic form” file is provided; I only allow students to use basicForm.txt if they did not turn in Assignment 4, because I want students to build on their own work. The assignment gives textual explanation of material covered in lecture in order that students have something to study.

**Assignment 5** (Extra Credit): The goal of this assignment is to emphasize order of execution as being critical – it should already be understood – and to have the students read and understand code. So, all they need to do is recode the draw( ) function in order to make the figure come out right, and then add a new box to the figure.

**Lab 4:** The goal of this lab is to provide an opportunity, like Assignment 4 did, to build structures with Processing, and to provide an opportunity, like Assignment 5 does, to focus on modifying coordinates to reposition an object. The lab session can be used to explain the ideas of “changing position through addition” without the confusion of a variable.

**Lecture 6:** This lecture introduces functions, parameters and arguments. This is a very difficult subject for everyone, and so this lecture might also benefit from being broken in half. The structure of functions is reviewed. Then parameters are introduced and illustrated. Then arguments are introduced and illustrated. Then I illustrate the ideas with the Mice Demo, which parallels assignment 6. If the lecture is split, I would recommend doing the intro to functions with many more examples (and probably handle return values), and then use the demo as the next lecture; also, I would delay assigning homework 6 until the demo had been done.

**Assignment 6:** The assignment is just constructing functions and including parameters in them. The use of color as a parameter has been illustrated before, but it might be worth a reminder to the students; specifically, **color( )** constructs a color data object and **color** is the type of that data. Notice that because there is no iteration yet, realizing multiple instances requires multiple explicit calls. The students love the last, optional step in which the owls are given random colors. The benefit of having them do this is that it sets up a great discussion for (a) why are the owls no longer colored as developed throughout the whole assignment, and (b) why are they flashing rather than being some random set of stable colors? (They won’t comprehend either one completely at this point, but there are many other opportunities for them to consider it; this starts that process.)

**Heads Up:** CS Principles as I teach it has a holiday Monday – MLK. Students have the day off, but because Assignment 6 has been written to be less do-this-do-that and more about achieving specifications, they get a little more time to turn it in. We resume with the next lab.