MOTIVATING STUDENTS TAKING CS1 BY USING IMAGE MANIPULATION IN C AND C++

Paul E. Dickson
Hampshire College
School of Cognitive Science
893 West St.
Amherst, MA 01002, USA
(413) 559-5861
pdickson@hampshire.edu

ABSTRACT

Creating a CS1 course that teaches good programming techniques and computer science concepts while also motivating students is a challenge faced by all computer science educators. We have all seen that students who are excited about their assignments make a greater effort and learn the concepts embodied in those assignments better. Based on this experience we have designed a CS1 curriculum that motivates students through assignments that involve image manipulation and creation. For this course we have somewhat reordered how concepts are typically taught in order to get students quickly into image manipulation. The choice of which concept to introduce is motivated by what it enables students to do to images. All concepts are covered using C for the first three quarters of the semester and then revisited during the final quarter using C++ to reinforce material and present object-oriented concepts.

INTRODUCTION

At Hampshire College we have no set majors and a student body that tends towards the arts. We therefore strive to offer CS1 courses that will provide a solid grounding to students focusing on computer science while also attracting students who would not usually take a computer science course. Bearing this in mind and also considering that students tend to learn more when excited and motivated by the subject matter, we set out to design a CS1 curriculum that focuses on image manipulation and creation.

We chose image manipulation and creation because it gives students visual feedback that lets them check the correctness of their projects by visual inspection and because gives a tangible feel of accomplishment. The approach to image manipulation includes both image processing and computer graphics, each used when it is the more appropriate way to convey a new computer science concept.

The typical order that programming material taught is somewhat changed so that each new concept enables students to progress with their image manipulation. For example, students use libraries well before they understand specifics of how libraries work and learn arrays and for loops before if statements. Reorderings of this type did not appear to cause students any difficulty despite our concerns while designing the course.

The decision was made to teach this course using the C family of programming languages for two reasons. First, our computer science students take

classes at a number of different colleges and these colleges do not use a common programming language. We felt that background in C and C++ would put our students in a good position to quickly learn any newer language required for the various courses. Second, we believed that by having material introduced first with C and then with C++, our students would gain a better appreciation for the differences between procedural and object-oriented programming. In addition, revisiting programming concepts in the context of a new but similar language would enable students to more firmly grasp the concepts.

RELATED WORK

Using image manipulation and graphics to help motivate students to learn concepts in the computer science curriculum and in CS1 courses in particular is not new. It has been introduced at every level from 9th grade [4] up to college database courses [6]. The integration varies from a couple of image-related projects [7, 8] to full courses [3, 5]. Common to all of these courses is that the instructors found that integrating image-related projects into the courses motivated students and helped with students' understanding of material.

One approach taken was to give students access to prebuilt libraries to give them a head start on image processing, which enables them to do more complex projects. McAndrew and Venables [4] used an environment that did not require much programming and primarily enabled students to run prewritten image processing techniques that were based on computer vision. This was effective for the 9th and 10th grade students in their course but is not as applicable to a full-semester CS1 course. Hunt [3] also used prebuilt image libraries but focused on having students use and expand those libraries. He had his students use Java's built-in image classes so that they could learn how to interact with prewritten code. While this proved effective, we wanted to start from a more rudimentary level and to build the larger framework embodied by Java's image classes.

Another approach taken was not to base a full course on images but instead to just have a couple of projects that included images. At Swarthmore College two projects were added to the CS1 course: a greyscale image processing project and a color image processing project [8]. These projects enabled students to learn basic GUI development while reinforcing arrays and other programming concepts. Students became fully engaged with each project. At the College of Staten Island computer vision techniques were introduced as a way of introducing research into a CS1 course [7]. Their results focused on the effects of adding a research component to CS1. However, these projects used computer vision techniques to scan brain images and some of the positive results may have come from students' excitement over getting to use techniques on images.

Clemson University has taken the most aggressive approach to integrating images into the computer science curriculum by bringing graphics into lower-level computer science courses. Graphics have been used for both a CS1 course [5] and a database course [6] as part of their Τέχνη project. For the CS1 course the approach is similar to ours using C programs to create portable pixel map (PPM) format images. UNIX image tools are used to convert images to more popular formats like JPEG and GIF so that students can view what they created. This approach focuses on

4 major image projects and uses only C; our course has 13 assignments that relate to images and also introduce C++ and a little bit of web design. Our use of a greater number of assignments enables us to give smaller assignments that enable a more gradual progression of skills.

COURSE ORGANIZATION

The goal of this course is to teach programming and the concepts included in a standard CS1 course while keeping students engaged and excited about the material. Part of this goal is to give students a fundamental understanding of programming that can be applied to other languages and environments. With this in mind, all course material is presented using the Emacs text editor within a command line environment. All programming is demonstrated using the terminal built into OSX; students who used Windows-based machines are encouraged to install Cygwin [2] so that they can also use a command line interface. We believe that by learning this baseline style of programming, students gain a better understanding of programming language vs. programming environment and are better positioned to learn more supportive programming environments in the future.

As with all programming courses this one begins with a "hello world" equivalent. (The exact syllabus used for the course can be found at http://helios.hampshire.edu/~pedcs/classes/cs110Spring10/syllabus.html.) The course then quickly covers variables, for loops, and arrays. Immediately after this, on the sixth day of class, the students are introduced to image libraries that enable them to read and write PPM images. These libraries are given without explanation of code and students are only shown how to use them. The students are told that before the end of the semester they will understand how the libraries work. Students tended to be a little overwhelmed when they are first asked to use these libraries but quickly move past that as they become excited by being able to create their first images. The use of Makefiles eases the introduction of libraries because students do not have to focus on how to include the libraries in their code.

At the same time students are introduced to the image libraries, they are given a basic introduction to web design and shown how to create webpages that contain images. This is done so that students can display their images as part of homework assignments and more easily show the images to friends. Students have been excited by a computer science course that enabled them to show off their work in ways more common to art courses. Because students uploaded the web pages to our school's Linux-based student webspace, creating and posting webpages also reinforced the command line material covered.

Once students have these image libraries and have seen how to use for loops to access every pixel in the image array, they are introduced to conditionals and if statements. If statements are the logical next step as they enable students to select certain pixels in an image that they wish to change. Conditionals with their Boolean logic follow as students see how they can be applied to select the pixels they wish to manipulate.

From here the course proceeds to nested for loops. Students find this normally difficult concept easier to grasp since it is far more logical to traverse an

image array by *x* and *y* coordinates than as a single dimensional array as they have been doing. With this ability to traverse the image in two dimensions, students are then introduced to basic filtering concepts like blur and haze, which are typically used with photography. Students have found it exciting to better understand how filters in Photoshop actually work.

From here students are introduced to while loops that are introduced in the context of applying image compression techniques (which keep running a process to decrease the number of colors used until a certain number is reached). Students then have structures explained and the Pixel structure they have been using all semester is put into context. They are then introduced to functions in the context of how functions can simplify their code and previous pieces of code are rewritten using functions.

We next address pointers, and this goes more smoothly than would normally be expected since the images that have been used all semester have been referenced as pointers in the code. This familiarity makes pointers easier to understand and gives students a good starting point for seeing where pointers are useful.

The final areas covered in C are libraries, input, and output. The concept of libraries is introduced and students are brought to understand how the libraries they have been using all semester function. This leads to the concept of input and output, both user and from the disk, and again the image libraries work as a map, showing students examples of how to read and write from image files.

From here the course shifts to C++ and the concepts of object-oriented programming. Students spend the end of the semester reimplementing their image manipulation algorithms in C++ as part of a piece of image manipulation software. Shifting code from procedural to object oriented sheds light on the differences between the two ways of writing code. Students seem to find this section empowering since it gives them greater insight into the similarities between syntax in programming languages that they had heard so much about while also showing them just how much programming they have learned.

The final assignment given is to build a text-based interface to their image manipulation software written in C++. This program runs from the command line and enables a user to load or create images, filter or modify images, and save images. At each step the user is given a list of options that to perform on the images. All of the options and the interface are created by students. This assignment shows students that they had the knowledge to build an entire piece of stand-alone software that might have application beyond a single assignment or the class. This assignment gave students a great sense of accomplishment.

COURSE STRUCTURE

The ordering of the material presented within this course is successful only because of the structure and support built into the course. The course has 18 homework assignments given over its 26 meetings. This means that homework assignments are due twice a week about every other week (the course met twice a week). The constant assignments mean that students are up to date with material presented in a previous class and that they have worked on their own with that

material before they learn the additional material that builds on it (e.g., they have done an assignment with for loops before being exposed to images).

These homework assignments are generally open ended, which encourages students to experiment. A typical example is the assignment given after students learn if statements. The assignment specifies that they create two images with code that included if statements. We discovered that such assignments tended to bring out the best in students since they tended to play with their assignments until they get interesting images, spending a lot more time programming than they would have otherwise.

The other piece crucial to this course is the use of undergraduate TAs (UTAs). We have had three UTAs who each attended class and held evening lab hours. They graded one assignment per week and generally helped the course go more smoothly. A description of our UTA program, its benefits, and our justification of it and using UTAs to help ease grading burdens has been published [1].

CONCLUSIONS

The goal for this course was to build a CS1 course that would keep students excited by the material throughout the semester and also appeal to students whose primary focus was not computer science. The course has been a success: students learn how to program and are engaged with the material. Part of the success is that students are given a chance to build real software that creates output that they can show to their peers. Very often the "coolest" thing a student creates during a CS1 course is an ASCII art pine tree. For this course students created real images that they took pride in showing to others.

The final assignment of the semester was a great success and left students with a sense of accomplishment. The act of building software that could be used by others and included hundreds of lines of code left them feeling that they had built something concrete. By rewriting all of their code so that it became part of this software, they not only reinforced their understanding of material but also came to better understand why commenting code is so important as they tried to understand the code they had written a month before.

By the end of the semester all students had a decent grasp of Unix and enough knowledge of web design to build basic websites. Seven of the 21 students in the course took computer science courses again the next semester. Because Hampshire College does not have set majors, we do not have data on numbers of computer science majors.

Using image manipulation and creation as a basis for motivating students to learn CS1 material appears to be a valid method for teaching CS1. The reordering of material necessitated by this approach appears to have no ill effects on student understanding. We highly recommend this curriculum because the levels of student motivation make it a pleasure to teach.

REFERENCES

[1] Dickson, P., Using undergraduate teaching assistants in a small college environment, *SIGCSE '11: Proceedings of the 42nd SIGCSE Technical Symposium on Computer Science Education*, ?-?, 2011.
[2] Cygwin developers, Cygwin Information and Installation, http://www.cygwin.com/, retrieved November 15, 2010.
[3] Hunt, K., Using image processing to teach CS1 and CS2, *Inroads-The SIGCSE Bulletin*, 35 (4), 86-89, 2003.
[4] McAndrew, A., Venables, A., A "secondary" look at digital image processing, *SIGCSE '05: Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, 337-341, 2005.
[5] Matzko, S., Davis, T., Teaching CS1 with graphics and C, *ITICSE '06: Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 168-172, 2006.
[6] Matzko, S., Davis, T., A graphics-based approach to data structures, *Inroads-The SIGCSE Bulletin*, 40 (3), 109-113, 2008.
[7] Sturm, D., Zelikovitz, S., Integrating research projects in CS1, *Journal of Computing in Small Colleges,* 25 (6), 55-59, 2010.
[8] Wicentowski, R., Newhall, T., Using image processing projects to teach CS1 topics, *SIGCSE '05: Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, 287-291, 2005.