Using Image Processing Projects to Teach CS1 Topics

Richard Wicentowski and Tia Newhall
Computer Science Department
Swarthmore College
Swarthmore, PA 10981
{richardw, newhall}@cs.swarthmore.edu

ABSTRACT

As Computer Science educators, we know that students learn more from projects that are fun and challenging, that seem "real" to them, and that allow them to be creative in designing their solutions. When we have students beating down our office doors wanting to show us what they've done, we know we have designed a project that truly meets its pedagogical goals. In CS1 courses, it is often difficult to come up with large, real-world programming projects that are at an appropriate level and that really excite students. This is particularly true in the first half of the course when students are learning basic programming and problem solving skills. We found that assignments based on image processing are an effective way to teach many CS1 topics. Because students enjoy working on the projects, they come away with a solid understanding of the topics reinforced by the projects. In this paper, we discuss many ways in which image processing could be used to teach CS1 topics. As an example, we present two image processing projects that we use in our CS1 course. These large, real-world programs are designed so that students can successfully master them early in their first semester of programming. Even though our CS1 course is taught using the C programming language, these projects could easily be used by a CS1 course in C, C++, or Java. We provide starting point code for Java and C versions of the projects, and provide sample assignment write-ups on our project webpage [12].

Categories and Subject Descriptors

K.3.2 [**Computing Milieux**]: COMPUTERS AND EDUCATION—*Computer and Information Science Education*

General Terms

Algorithms

Keywords

Computer Science Education, CS1, Image Processing Projects

1. INTRODUCTION

At Swarthmore College, the computer science curriculum offers a broad exposure to the discipline through three introductory courses, which could be termed CS1, CS1.5, and CS2. CS1 takes an imperative approach using C, often with Roberts' text [10]; CS1.5 takes a functional approach using Scheme with Abelson and Sussman's text [1]; while CS2 takes an object-oriented approach using Java, often with Goodrich and Tamassia's text [6].

Our version of CS1 is called *The Imperative Paradigm: Unix and C*. It is an introduction to programming in the C programming language, and we include an introduction to Computer Science, Unix, and some data structures and algorithms. Typically, we introduce a new feature of the C language each week and give students one or more programming assignments to reinforce this feature. We like to give students several larger programming assignments during the semester so they can experience the benefits of using good modular design, using makefiles, and incremental implementation and testing.

We have found that projects based on image processing are a good way to introduce larger, real-world projects early in a CS1 course. We present as examples two image processing projects that are used in our CS1 course. One version manipulates greyscale images and can be used after teaching arrays early in the semester. Typically, we provide much of the shell of the program and students implement functions to manipulate the image in different ways. The other version manipulates color images and can be used after covering compound data types, such as structs or classes, later in the semester. For this version, students are required to design more of the program themselves.

Most of our CS1 students list the image processing project as their favorite from our course. As a result, we think it is a good vehicle for demonstrating how data structures and algorithms are used in practice, as well as for developing communication, programming, and problem solving skills in students. Based on our own observations, in semesters when we use these assignments, our CS1 students have a better understanding of arrays, and of the semantics of passing arrays to functions, then they do in semesters when we have not used these assignments.

An additional benefit of using these assignments is that students feel a real sense of accomplishment in completing them. The projects initially seem very large and difficult. Students come to realize, however, that their solution to one image manipulation feature can be used as a starting point for adding other features. Thus, what students first see as a daunting assignment turns out to be manageable. Nearly all students implement the required parts. In addition, many students add some impressive extra-credit features.

Our success with these projects made us realize other ways in which image processing could be used to teach introductory CS topics. We are not proposing that one use only image processing projects in CS1, but that there are numerous points when image processing could be used to teach CS1 topics.

There has been prior work on using image processing projects in CS1 and CS2 courses [3, 2, 5, 7]. However, previous work presents either specific image processing projects or image processing libraries. Our work focuses on how image processing projects can be used more generally to meet specific pedagogical goals. Additionally, we present survey results measuring the effectiveness of the application of some of our ideas. In Section 2 we discuss several of our ideas for how to use image processing to teach CS1 topics. In Section 3 we present our two image processing projects, we discuss how they are used, and we present student reaction to the projects, and we conclude in Section 4.

2. USING IMAGE PROCESSING IN CS1

Image processing can be used as an effective way to teach a number of CS1 topics, including both one-dimensional and multi-dimensional arrays, iteration, recursion, pointers, dynamic memory allocation, compound data types, sorting, file I/O, functions, and function call semantics. In Java, projects centered around image processing also can serve as an introduction to teaching event-driven programming, threaded programs, and GUI programming, as well as exploring issues associated with designing user interfaces. For a CS1 course using an object-oriented language such as C++ or Java, a color and greyscale image processing project can be used as a real-world example of inheritance; from Image and Pixel base classes, specialized greyscale and color classes can be derived.

For a CS1 class, an instructor would likely provide students with a library or class containing the I/O routines necessary for translating an image file to a 2-D array of pixel values. In advanced classes, reading the raw image file may itself be a challenging class project. Most modern image formats are quite complex and require that students create supporting data structures to translate the image file into a 2-D array of pixels. For example, the GIF [4] file specification requires that an 8-bit color table be included with the image, and the image data is compressed using LZW compression, a modification of the Lemper-Ziv compression algorithm. For students to extract a GIF image, they would need to create a data structure to hold the color table, then uncompress the image, mapping each pixel to its appropriate color in the color table. This task is beyond the scope of our CS1 course, but could be incorporated into a CS2 project.

Once the image has been stored into a two dimensional array of pixels, there are numerous ways in which the image array of pixels can be manipulated to produce different effects on the image. Some effects give students real-life examples of the importance of finding a generalized solution to a problem. For example, image tiling and splitting effects can be coded in a few lines if students find the general recursive or iterative pattern for how the image is split at each step. On the other hand, if students try to hard code the solution for tiling or splitting an image, they could end up writing hundreds of lines of code to implement the effect (see the infinite splitting effect in Figure 2 as an example).

Image processing projects that require dynamic memory allocation for the image array can be used to reinforce students' understanding of pointers, function call semantics, type, and scope. Effects such as doubling the size of image or rotating the image by an arbitrary angle require dynamic memory allocation. When students are required to write functions to perform such tasks, they need to have a good understanding of pointers, including the ability to return a dynamically allocated 2-dimensional array, and getting the function prototype correct.

Image processing can be used as part of a larger problem that focuses on problem solving. For example, an eight-squares puzzle game with an image manipulation component could be used as a complete assignment, or as an extension of an earlier image processing project. Here students have to solve a more difficult problem where they need to keep state about the game board, modify the image after each step to reflect the move made by the user, detect and disallow invalid moves, and detect when the puzzle has been solved. For the Java version of this assignment, it is straightforward to add mouse click event handler methods for playing the game; for the C version, a text interface may be used to read in the user's next move.

For CS1 courses in Java, there are many image processing effects that can implemented because the Java Swing library is part of the Java language and it is fairly easy to use. Students can easily implement image processing effects in response to mousing events on a displayed image. For example, zooming into an arbitrary part of the image can be triggered by a mouse click, or arbitrarily rotating an image can be triggered by a mouse press-move-release sequence. Also, many drawing effects can be implemented in response to mouse events. In fact, most parts of a simple painting program, such as Microsoft's Paint [8], could be implemented as modifications to a displayed image; in response to mouse events, an eye dropper effect to select a color, a paint brush effect with different brush widths and brush types, a copy and paste effect, and an image cropping effect can be implemented.

3. OUR IMAGE PROCESSING PROJECTS

The primary goal of our CS1 project is to reinforce arrays and compound data types through the use of a real world application. We also accomplish a number of secondary goals: to illustrate the importance of good modular design; to illustrate the importance of implementing and testing code incrementally; to show the importance of efficiency and generalization in designing algorithms; to give students practice implementing sorting algorithms; to give students practice using makefiles; and to provide an opportunity for students to work in teams.

Our image processing project takes an image file as input, and then provides a menu or GUI interface to the user to select different image manipulation features such as blurring the image or rotating the image. Students implement each image modification as a separate function. Each image manipulation function takes the two dimensional image array as input. To solve the assignment, students must know how arrays are passed to functions, how to access array elements, and how to develop algorithms for processing the pixel array to get the desired effect.

Extra credit parts are used to challenge some of the more ambitious students. Some of the image manipulation features we suggest as extensions are quite difficult (like rotating the image to an arbitrary angle). We also encourage students to come up with their own features. This is a nice way to allow for some creativity and flexibility in a course where the assignments are often completely defined.

3.1 Greyscale Image Processing Project

The greyscale image processing project is given to students during the sixth week of class after introducing arrays and simple sorting algorithms. At that point in the semester, students have written functions, they know about local variables and scope, they have done some simple I/O, and they may have learned about C libraries. They have not, however, written large programs, nor do they know pointers or dynamic memory allocation, nor have they had to explicitly link in library code to build an executable file.

As a starting point, we give students a compiled image extraction and display library (such as a .so or .class file). The library contains functions to convert between an image file and a two-dimensional array of pixel values, and functions to display the modified image. We use the Java Swing Library [11] for Java, and libtiff and Tcl/Tk [9] for C. In the C version, image manipulations are stored to a temporary file that is then displayed by the GUI image viewer written in Tcl/Tk. The Tcl/Tk parts of our code could easily be replaced by code that uses an external image viewing program, like xv or a web browser, to view the modified image file after the user selects an effect.

We additionally give students a starting point source file containing a partially complete main function that calls some initialization functions in the image display library, a main loop function that prints a menu and reads in the user's menu option, and function prototypes for some of the image processing features they will implement. In the Java version, instead of a menu, the starting point code creates GUI buttons with associated action methods that will call the student-implemented image manipulation methods. We also provide a complete makefile for students to use to compile their program.

We provide function prototypes in this assignment's starting point because function writing is still quite new to the students, and because the syntax required for passing two dimensional arrays as parameters can be tricky. We also provide them with prototypes because we want them to see how a good, modularly designed solution is structured. Our survey results (see Section 3.3) show that the majority of our students feel that providing them with function prototypes was important to their successful completion of this project.

Students are required to implement a number of different image manipulation features, and are encouraged to implement extra features. For example, in the past we have required 15 features including making a negative of the image, flipping the image vertically and horizontally, switching the top left and bottom right corners, darkening, lightening, polarizing, scrolling both horizontally and vertically, zooming in (to the center or any of the four corners), blurring, rotating 90 degrees, sorting the rows by pixel value, and reverting to the original image (the other effects are cumulative). An example of some of these is shown in Figure 1. Some extra features we propose are edge detection, an eight-squares puzzle effect, displaying a histogram of pixel values, splitting the image, tiling effects, and rotating the image by an arbitrary degree amount. We provide students some pointers to on-line documentation about some of these effects. An example of some of the effects is shown in Figure 2.

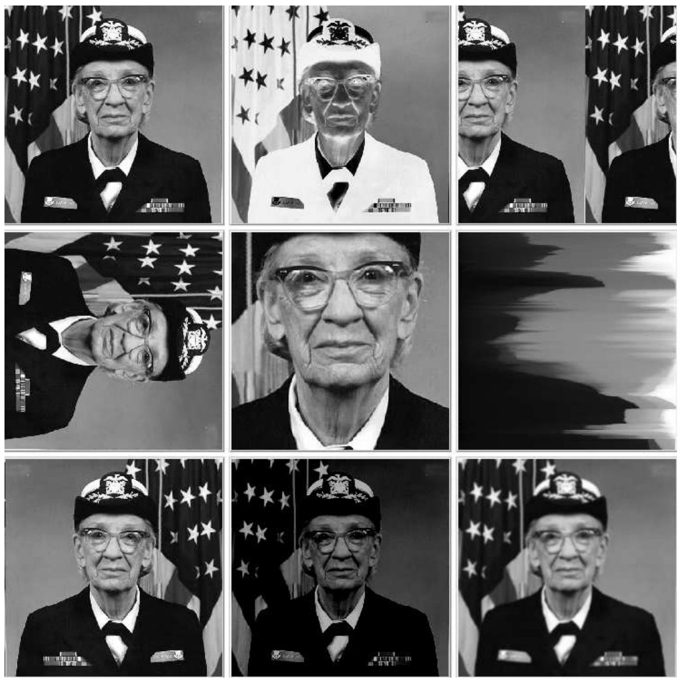

Figure 1: Example Image Processing Features. ***Top Row, from left: revert to original, make negative, scroll vertically by some number of pixels (200 in this example). Middle Row: Rotate 90 degrees, Zoom (center zoom chosen), Sort Rows by pixel value. Bottom Row: Flip Vertically, Darken Image, Blur Image. (Grace Hopper image from*** `www.arlingtoncemetery.net/ghopper.htm`***)***

Initially, fifteen required image features sounds daunting. However, students quickly see that by starting with some of the easier effects, like producing the negative image, they have a starting point for solving some of the more difficult effects. Some effects, such as blurring the image, cannot be done in-place on the image and require students to discover that they need to make a temporary copy of the image to correctly implement the effect. Students also need to be careful about stepping beyond the row or column bounds in their solutions. Often a strange looking result will help to reinforce this.

A few of the effects are a bit odd, and are included mainly for pedagogical reasons. For example, sorting the rows by pixel value may not be a useful, real-world image effect, but it is a nice way to have students implement a simple sorting algorithm as part of a larger assignment. Another example is the histogram effect where students must create a temporary array for the histogram, step through every pixel in the 2-D array to find the histogram values, and then figure out how to modify the image so that it displays the histogram. One tricky part to this effect is that the histogram may need to be scaled. If, for example, a single pixel value occurs in the

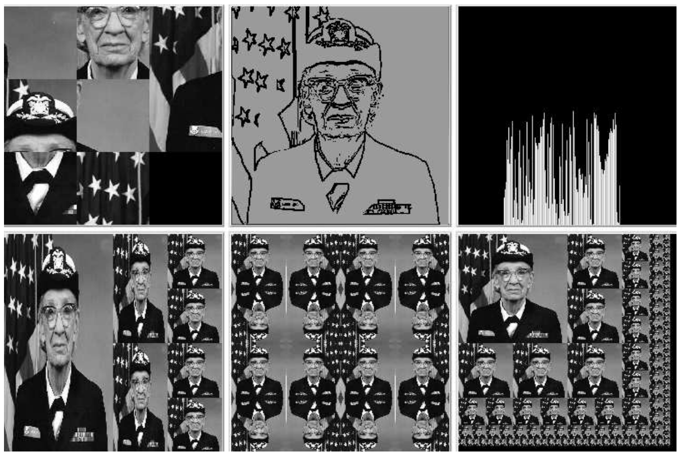

Figure 2: Example Extra Credit Image Processing Features. ***Top Row, from left: Eight Squares Puzzle, Edge Detect, Histogram of Pixel Values. Bottom Row: Split Image Right, A Tiling Effect, "Infinite Split"***

image more than the number of pixels in the height of the image, then the histogram bucket values need to be scaled so that they can be represented by a bar that fits within the height of the resulting image.

3.2 Color Image Processing Project

The color version of the project is introduced in the tenth week of classes after we cover structs. A color image is manipulated as a two dimensional array of RGB pixel structures. The pixels can be implemented as a struct in C, a class in C++ or Java, or as a three element array. Since the color version of the assignment comes later in the semester, we do not give as much starting point code as we do for the greyscale version. We still give students a complete makefile and part of a main function containing calls to our image processing library to create the initial 2-D array from an input image file. However, students are responsible for designing more of the program, including most of the function prototypes, themselves. We also require that students implement a solution that works for a color image of any size. This is a good way to reinforce pointers and dynamic memory allocation.

Many of the greyscale image processing features can be used in the color image assignment. In addition, students implement color "polarizing" features, where they shift all red values (for example) to extremes based on a threshold value. We have also taken pictures in front of a blue screen to create images that students can use for implementing background replacing features. Students can replace the blue background with different colors, different patterns, or with another image.

Some of the greyscale effects cannot be easily implemented in the full color version. For example, sorting rows by pixel value and the histogram effects may not make a lot of sense for color images. Of course, both effects could be done on a single R, G, or B pixel value, or on a function of each pixel's RGB values, but the resulting image will be strange and students can not easily tell if they have correctly implemented the effect.

Project Helpful in Reinforcing Understanding of:	
arrays in general	86%
semantics of passing arrays	83%
array data access/manipulation	90%
searching and sorting algorithms	67%

Table 1: Survey results showing percentage of students who answered either "very helpful" or "helpful" to questions about how well the assignment helped reinforce their understanding of arrays.

3.3 Student Response to the Project

One of the most satisfying results of using these projects, aside from meeting pedagogical goals, is that students really enjoy them. It is so much easier for students to learn the material if they are excited and motivated to do the work.

To better quantify how well our projects meet our pedagogical goals, we surveyed 42 students in our introductory course about their experiences with the greyscale version of our project. Overall the students enjoyed the project and felt that it helped them to understand arrays. In Table 1 we show the percentage of students answering either "very helpful" or "helpful" to questions about how well the project help to reinforce their understanding of two-dimensional arrays.

Of the surveyed students, over 80% said that they "really enjoyed" or "enjoyed" the project, and only one student didn't like the project. Almost all students felt that it was at about the right level of difficulty, and most stated that it would have been difficult to complete without being given the function prototypes for some effects. Although a few students felt that the starting point code made the assignment "slightly easy", no student thought that it made the assignment "too easy". Given that students enter our introductory course with a wide range of previous programming experience, it is not surprising that a few students would find this assignment easy. Our survey results reinforced our feeling that giving students the function prototypes is appropriate for this assignment.

This is the first project where we strongly encourage students to work in pairs, and we wanted to gauge student reaction to this experience. Most students (36 out of 42) worked with a partner on this project. Almost all found it helpful to work with a partner; most mentioning that it was nice to split up the work, and that working with a partner was particularly useful for problem solving and debugging. One student stated "I liked working with a partner because we never got stuck for extended periods of time–whenever one of us was stumped the other usually came up with the solution." Another student mentioned that working with a partner "allowed us to brainstorm together and after discussing an idea we could come up with something that we probably wouldn't have thought up on our own." The only negative comments about working with a partner had to do with managing scheduling difficulties.

On both the survey and on course evaluations from previous semesters we asked students what they liked or disliked about the project and why. When we asked in course evaluations which assignment was the favorite and why, nearly all students picked the image processing project. Most mentioned that it was fun and challenging. For example, one student said "The image lab was the highlight of the course." Several students also liked that it was a real-world problem: "I thought the picture lab was a lot of fun...[it] is nice to see how programming can be used in real applications" and "the graphics programs were awesome and very practical." Another student stated that she "thought it was a really cool assignment–it gave us the opportunity to manipulate images and actually feel like we were doing real programming." Another student stated that "I really enjoyed this assignment and felt that it allowed me a chance to see the effects of my program and develop some understanding of how my programs work."

Many students mentioned that they were surprised that they could complete what seemed like a very large assignment: "I liked this assignment because it gave us a chance to put the skills we had learned to use and create a program that produced visually impressive results. It seemed really intimidating at first, but it went pretty quickly and it was satisfying successfully implementing the effects." Some students also liked the open-ended aspect of the assignment: "Pictures were cool because [it was] open-ended." The only negative comments we heard about the assignment was that one student didn't like that we gave them function prototypes: "We were fiddling around with functions that were already given." Another student felt that the image manipulation features were a bit too repetitive and would have liked fewer features.

The survey results support what we had suspected about these projects based on our interactions with students: our students enjoy the projects, they like designing and implementing extensions, they like that the projects are real-world applications, and they feel a real sense of accomplishment in completing the projects. The results also fit our observations that in semesters when we have used these projects, students seem to have a better understanding of arrays and of the semantics of passing arrays to functions.

4. CONCLUSIONS

Image processing projects are an effective pedagogical tool for teaching a wide range of CS1 topics, including multi-dimensional arrays, function call semantics, and modular design. Although these are large programming projects, they can be structured in such a way that CS1 students can successfully complete them. Student reaction to these fun, real-world applications of CS1 material has been overwhelmingly positive, resulting in students who are more engaged in learning. Students feel a real sense of accomplishment in completing these projects and become more confident and capable programmers.

5. REFERENCES

[1] Abelson, H., and Sussman, G. J. *Structure and Interpretation of Computer Programs, Second Edition.* McGraw Hill, 2001.

[2] Astrachan, O., and Rodger, S. H. Animation, visualization, and interaction in CS1 assignments. *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education* (February 1998).

[3] Burger, K. R. Teaching two-dimensional array concepts in java with image processing examples. *Proceedings of the Thirty-Forth SIGCSE Technical Symposium on Computer Science Education* (February 2003).

[4] Compuserve Information Service. Graphic Image Format.

[5] Fell, H. J., and Proulx, V. K. Exploring Martian planetary images: C++ exercises for CS1. *Technical Symposium on Computer Science Education Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education* (February 1997).

[6] Goodrich, M. T., and Tamassia, R. *Data Structures and Algorithms in Java, Second Edition.* John Wiley and Sons, Inc., 2001.

[7] Hunt, K. Using image processing to teach CS1 and CS2. *SIGCSE Bulletin 35*, 4 (December 2003), 86–89.

[8] Microsoft Corporation. Microsoft Paint. http://www.microsoft.com/.

[9] Ousterhout, J. K. An X11 toolkit based on the Tcl language. *Proceedings of USENIX Winter Conference* (1991).

[10] Roberts, E. *The Art and Science of C.* Addison Wesley, 1995.

[11] Sun MicroSystems. Java Swing Library, part of the Java 2 Platform. http://java.sun.com/j2se/.

[12] Wicentowski, R., and Newhall, T. Two image processing projects for a CS1 course. www.cs.swarthmore.edu/ ˜newhall/imagemanip/.