Using Graphics Research to Teach Freshman Computer Science

Sarah Matzko and Timothy Davis
Department of Computer Science
Clemson University

Abstract

Although research is often a daunting task for undergraduates, early exposure to research topics and papers can be an effective means of stimulating interest and teaching students about real-world solutions to non-trivial problems. To that end, we have initiated a new educational project, termed τέχνη, to teach general concepts in computer science by infusing large-scale problems in computer graphics into the undergraduate curriculum. In this paper, we present our approach to teaching a first course in computer science (CS1) using a recently published graphics research problem. The results from our first semester offering are promising, as students, energized by the subject matter and visual output, produced excellent work and evaluated the class highly.

Categories and Subject Descriptors

K.3.2 [Computer and Education]: Computer and Information Science Education – Curriculum.

General Terms

Education, Design, Experimentation, Research

Keywords

Computer graphics, problem-based learning, computer science education, curriculum issues, undergraduate research.

1. Introduction

Providing research opportunities for undergraduates is a widely respected method of improving education for students while offering expanded options for professors. Although research is a difficult enterprise for undergraduates to embark upon and is intimidating to many students, early exposure to research topics and papers is an effective technique to prepare students for engaging in research in later years of their education. While the concept of introducing undergraduate students to research topics is readily acceptable, the application proves more difficult, as beginning students may become unenthusiastic or discouraged by theoretical topics. However, if these topics involve graphical results, students may be motivated by the prospect of creating such visuals themselves. Indeed, due to the entertainment industry, including film, television, and computer gaming, students entering college typically have had broad exposure to computer-generated images and are likely to possess an interest in their production and use. Thus, if the research topics are graphical in nature, we can afford students exposure to research that they would deem worthwhile.

Currently, we have leveraged student interest in graphics by developing a new approach, termed τέχνη, to teach computer science concepts through computer graphics problems in the undergraduate B.A. curriculum. This program is further outlined in Section 2, with related work presented in Section 3. While the τέχνη approach spans all courses in the curriculum, our focus in this paper is freshman computer science (CS1). The first part of this course, described in Section 4, led to the final project involving results from recently published graphics research, as detailed in Section 5. Images produced from students assignments are also provided, while a discussion of what we have learned is given in Section 6.

2. The τέχνη Project

The name τέχνη is the Greek word for art and shares its root with τέχνγολοία, the Greek word for technology. As such, the term reveals the close academic relationship the two fields have held historically. A major goal of the τέχνη project is to reunite these areas for more effective means of teaching.

The inspiration for τέχνη originated from our experiences in the establishment of a new cross-disciplinary digital production arts (DPA) program. This master's level degree combines elements of computer science, art, theater, and psychology, among others. Graduates who have completed the program pursue careers in the special effects industry for film, television, and gaming. Studios that have hired our students include Rhythm & Hues, Industrial Light & Magic, Pixar, Blue Sky, Electronic Arts, and Sony Imageworks.

The primary goal of the τέχνη project is to incorporate graphics projects and research from DPA and computer science in the undergraduate computer science curriculum. This material takes the form of semester-long projects in required courses leading to a B.A. in computer science. We believe this approach to be an effective pedagogical method in teaching general computer science concepts since it naturally encompasses several education-theoretical techniques, including: visual feedback, problem-based learning [Duch et al. 2001], intentional learning [Martinez 1997], constructivism [Mordecai 1998; Ben-Ari 1998], and problem-

based learning [Cunningham 2002; Boud and Feleiti 1991]. Through this approach, our goal is to improve understanding of key computer science concepts, while engaging students through projects focused on a field of current interest. This approach provides opportunities for students to explore new topics as they naturally arise in large-scale graphics projects. Accordingly, the instruction is problem-based, with projects ranging from traditional graphics problems, such as ray tracing, to cutting-edge implementations from current research, as discussed later in this paper.

Our first experimental course in the τέχνη program was a sophomore-level course in C/C++/Unix. Students were assigned a semester-long project of implementing a ray tracer, a task previously reserved for our graduate-level advanced graphics course. The ray tracer project provided an ideal pedagogical platform for problem-based learning for several reasons: it naturally covers a broad range of computer science concepts; it provides visual feedback at all stages, allowing program correctness to be determined immediately; and it naturally leads to discussion and implementation of an object-oriented paradigm. Finally, the project allowed for a certain degree of artistic expression, usually devoid in computer science undergraduate projects, in the form of scene content. We have offered the course several times with excellent results, in terms of student engagement and images produced. Additional details are found in [Davis et al. 2004].

Our next step involved bringing graphics concepts to the earliest courses in the curriculum. Initially, we were unsure of whether freshmen were capable of the task we were setting before them, even with the assistance of faculty and graduate students involved in the project. At the end of the semester, however, we were delightfully surprised with the results.

3. Related Work

Because of the positive impact undergraduate research has had on both students and faculty in the past, a growing number of universities now offer research opportunities for undergraduates [Wenderholm 2004]. Many institutions, including Clemson University, already involve undergraduates in research, and seek to continue the trend [Barker 2002]. This kind of research typically takes the form of optional work for highly competent upperclassmen. However, introducing lower-level students to research topics is unusual, since early classes are typically more focused on teaching supporting topics, such as programming logic, algorithms, and data structures.

While involving current graphics research in CS1 is uncommon, image processing in CS1 is gaining popularity, due to the interest these projects generate and the principles they reinforce. As shown by previous work, image processing projects can be used quite effectively in teaching students from elementary school [McAndrew and Venables 2005] to college [Wicentowski, and Newhall 2005; Astrachan and Rodger 1998; Burger 2003; Fell and Proulx 1997; Hunt 2003], since students typically enjoy working on real-world problems and producing visual results.

Additionally, image processing projects provide an environment in which a wide range of computer science topics, such as dynamic memory allocation and two-dimensional array access, naturally arise [Burger 2003]. Coupled with a problem-based learning model, such projects demonstrate to students the necessity of these and other complex programming techniques. Similarly, because images are composed of thousands of pixels, students cannot hard-code the solutions and are forced to design general solutions to problems [Wicentowski, and Newhall 2005].

Our method of teaching freshman computer science under τέχνη also involves image processing projects, but the nature of these projects differs from these other approaches in several ways. First, instead of including a single project, where much or all of the code is provided, we require students to develop all image input, parsing, and output functions. In fact, we submit that this approach is preferable, as it allows the introduction and presentation of a broad range of topics to arise naturally during the implementation of the projects. Second, all assigned projects build toward a large final project. In this way, students can experience the incremental nature of software development and gain further understanding of the intricacies of handling a larger system. Finally, the final project requires an implementation of a chosen research problem recently published in a computer graphics journal or conference. The next section discusses the contents of the course leading up to this final project.

4. Early Course Content

Since freshmen enter CS1 with little or no knowledge of computer science, we must start at a point that is obviously a substantial distance from current research in the computer graphics field. With proper planning and guidance, however, students can reach a point within a semester where they are capable of reproducing the results of cutting-edge graphics research. The class was structured around this final project with preliminary assignments introducing concepts necessary for later coding. These key skills included 1) parsing image files, 2) performing computation on a large amount of data multiple times, and 3) outputting resulting image files. Each project in the class sequentially progressed toward the final goal of implementing a color transfer algorithm, described in Section 5.

4.1 Image Creation Project

The first project required students to create an image in portable pixel map (PPM) format, a simple image file format containing a brief ASCII header and three bytes (red, green, and blue) per pixel. Image processing tools can display or convert PPM files to other more commonly used formats, such as JPEG or GIF.

The first project required the creation of a single-color 800x600 image in binary PPM format using C. This task naturally provided the impetus for introducing several data types and looping. Since the code necessary for this assignment was minimal — a main function, *printf*, and *for* loop — no starter code was provided. Resulting images from student programs ranged from the obvious (e.g., a monochromatic purple image) to the more ingenious (e.g., diagonal stripes with varying intensities), as shown in Figure 1.

4.2 Image Grayscale Project

Once students acquired the ability to write an image file, they were ready to move on to reading, or parsing, an image file. The

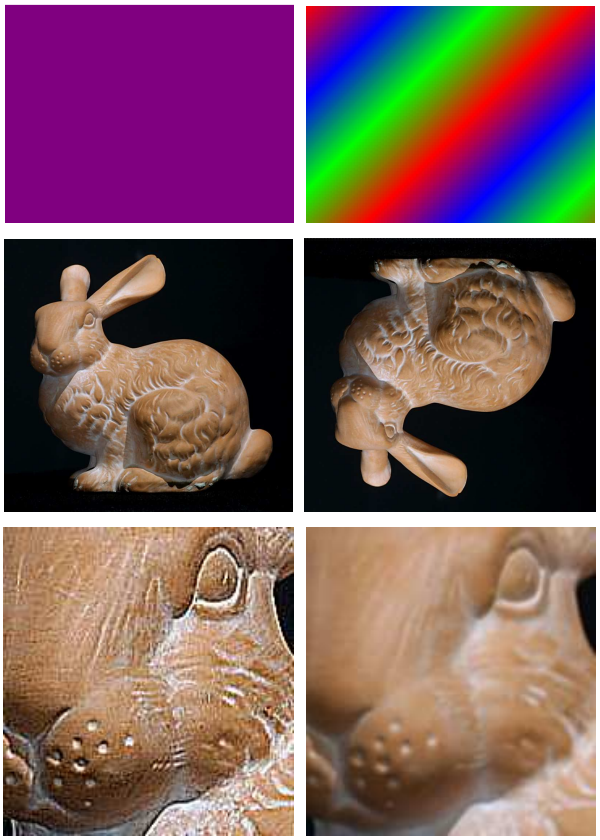

Figure 1: Image creation project results (top);
Image manipulation project results (original, flipped, sharpened, blurred) (bottom)

second project required students to read in a color image, process the pixels, and output a corresponding grayscale image. At this stage, however, students were not prepared to dynamically allocate memory for an input image, nor to read string values into arrays. However, since converting an image to grayscale can be handled one pixel at a time, students simply read three bytes (red, green, and blue) from the input image, converted the values into a grayscale equivalent, and output that single corresponding value. This project required students to learn about programming techniques, such as input/output processing and manipulating data mathematically.

4.2 Image Manipulation Project

The third project involved more difficult image processing effects, necessitating repeated or out-of-order access to the pixels in the image for processing. As a result, this third project required students to store and manipulate image data in their programs, which was accomplished through dynamic memory allocation and accessing pixel values in an array. This final preparation project allowed students to apply any one of various image effects, including color fading, image tiling, image rotating, color balancing, image resizing, half-toning, or convolution filtering (e.g., providing edge detection, or blurring or sharpening an image). Output results from student projects are shown in Figure 1.

5. Research Project

The final project was based on a color transfer algorithm presented in the 2001 paper, “Color Transfer Between Images” by Reinherd, et al. [2001] published in *IEEE Computer Graphics and Applications*. The paper describes an algorithm for transferring the color scheme of one image to another. Not only is the algorithm well described and easily understood, it produces interesting results and requires complex programming for undergraduate students. All of these characteristics make the algorithm an ideal choice for implementation in the final project in our course.

The color transfer algorithm requires two input images: a target image and a reference image. The target image’s values are modified to reflect the color scheme in the reference image (see Figure 2). However, the algorithm requires the image data to be in lαβ color space, instead of RGB, to prevent the range of colors from being skewed. The paper provides methods for converting from lαβ to RGB and from RGB to lαβ, both of which must be implemented to produce the final image.

The final project thus required the following steps, with general computer science topics indicated in parentheses:

1. Read in two images specified on the command line (command line and string processing). Students opened each file (input/output), determined the number of pixels (parsing), allocated the necessary space (dynamic memory allocation), and read in the image data (input/output and casting).

2. Convert the RGB values in both images to LMS format via the provided conversion matrices (mathematical functions). Because performing the matrix multiplication was not a key part of the assignment, we provided matrix multiplication code to the students (code compatibility); however, alternatively, this material could have been covered in a lab setting.

3. Convert the log of the LMS values into lαβ format via the provided conversion matrices (looping).

4. Compute the mean and standard deviation of the l, α, and β values in both images (mathematical functions). The difficulty with this task was that the mean and standard deviation had to be computed separately for l, α, and β, as well as for both images. Students therefore had to keep track of all six data points to correctly perform the scaling (data handling). To simplify this task, students defined a structure for storing image data that held one image’s width, height, data array, means, standard deviations, and any other data necessary (data structures).

5. Scale the lαβ values in the target image to have the same mean and standard deviation as the lαβ values in the reference image (data storage and manipulation). The steps for scaling (described in [Reinhard et al. 2001]) involved simple arithmetic.

6. Reverse the conversion to lαβ to acquire the new RGB values for the target image using the provided conversion matrices.

7. Output the target image (input/output).

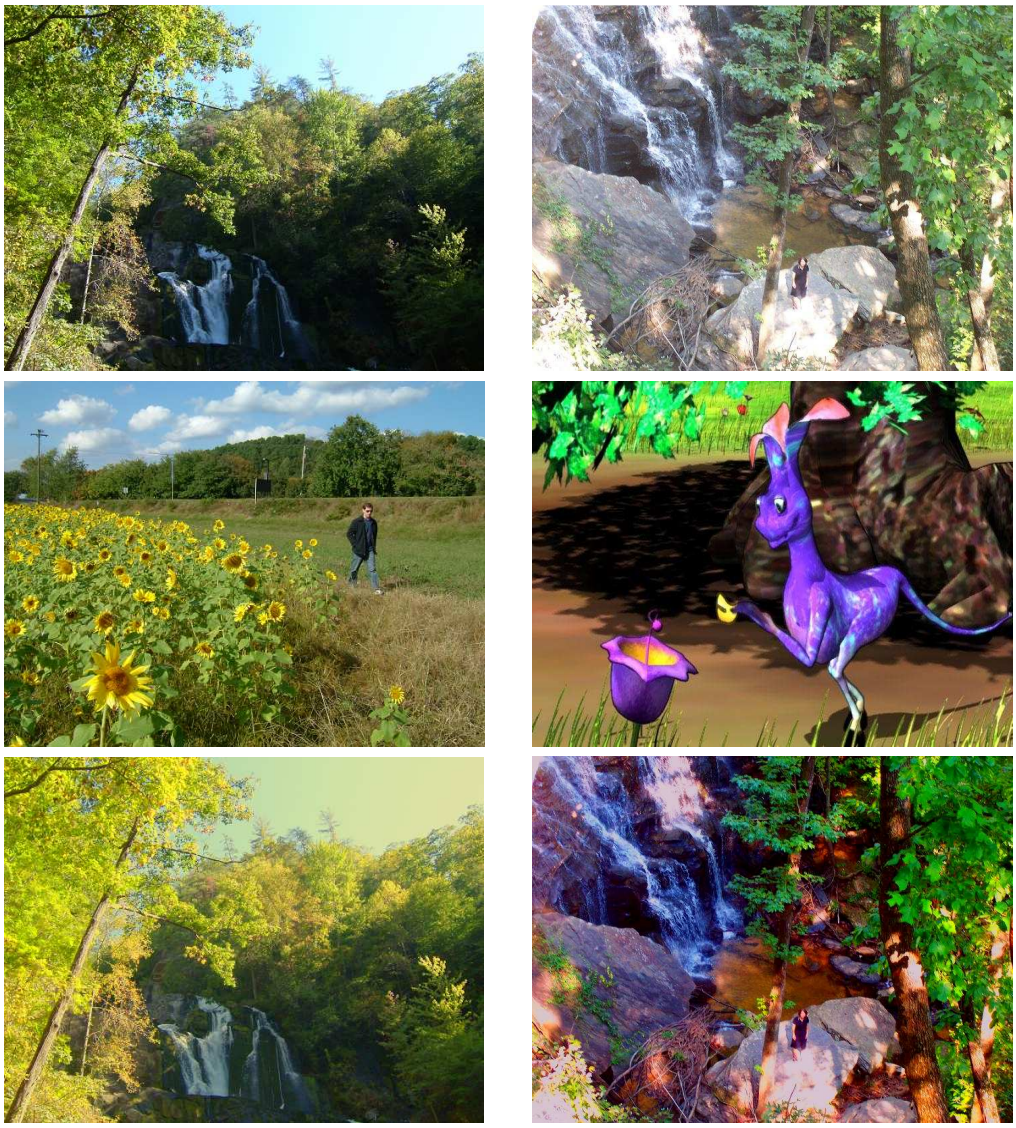

Figure 2: Results of students' color transfer programs. Top: target images. Middle: reference images. Bottom: Re-colored images.

Because this project was relatively large, the assignment was completed in pairs. Allowing the students to work in pairs provided a way for them to discuss the algorithm and work through solutions. Since research shows that students work best with someone they perceive to be of similar technical competence to their own [Katira et al. 2004], students were asked to submit names of people with whom they believed they were matched evenly and would work well. Students were then paired with requested students whose grades were similar. The project provided a link to the paper, a list of steps necessary to complete the assignment (similar to the list given above), and example input and output images. The conversion matrices were not provided directly; instead, students were required to obtain this information from the paper itself. After the projects were completed, students were required to present the logic of the program by stepping through the code individually with the instructor.

The submitted projects and resulting images indicate that the project was within the range of the students' abilities and was a good introduction to research topics. Two thirds of the students were able to implement a correct solution for the color transfer algorithm. All but a handful of students were able to explain what each line of their submitted code did (or should have done) and why. Several students referenced the research paper for explaining parts of their solutions, which indicates an understanding and synthesis of the information. Additionally, of the 19 students who responded to the course evaluations, 11 students (58%) said that after seeing the paper they had interest in doing computer science research. Example images produced from student projects are shown in Figure 2.

6. Discussion

Involving research in undergraduate courses has produced many beneficial outcomes. While entering freshmen often do not have enough background in computer science to investigate new topics, they can examine and reproduce the results of other current research. In doing so, these students observe what is involved in

performing research, how to document the results, and what problems remain as open questions. Thus, by incorporating research topics in early courses, we can introduce students to the process of research, preparing them to become more competent computer scientists [Becker 2005], or engage in research of their own at a later time.

Exposing students to research provides additional advantages. First, research topics illustrate to students what computer science is about. Some students mistakenly believe that computer science is about using software applications [Beaubouef and Mason 2005], while others believe that the field is limited to currently popular technologies, such web development or particular programming environments. One former computer science student expressed frustration at the topics in the sophomore data structures class. He asked, "Why would I need to know about different sorting algorithms and how to compare them? It's not like new algorithms will be developed, and the libraries already have sorting methods built in." His goal was to learn how to create web pages, and despite a year in computer science, he believed the field to be at a permanent point in development that could no longer benefit from further algorithm research. If students study the work of current computer scientists early on, these misconceptions can be addressed within their first year.

Introducing students to current research also accomplishes the difficult task of creating projects that are meaningful and realistic. Because of the low competency levels in freshmen classes, instructors often resort to "toy programs" that are not meaningful outside of the classroom. These toy programs tend to promote boredom in both students and instructors; however, projects based on current research tend to be more engaging since they typically meet a practical need or otherwise address a meaningful problem. Further, professors can choose research topics that are of interest to them or relevant to their research program. Given the pressure to perform research and publish papers, some faculty members would not otherwise find lower-level classes to be interesting. However, introducing these research topics to students in a palatable way is an effective means to engage both the professors and the students. Over time, these topics will inevitably change to reflect current work.

We are continuing to provide new courses with the τέχνη approach. On the heels of our CS1 experiment, we are currently offering CS2 using graphics problems to teach second-semester computing concepts.

Acknowledgments

This work was supported in part by the CISE Directorate of the U.S. National Science Foundation under award EIA-0305318.

References

ASTRACHAN, O., AND RODGER, S. H. 1998. Animation, Visualization, and Interaction in CS 1 Assignments, *SIGCSE Bulletin*, 30(1), 317-321.

BARKER, J. F. 2002. *Newsletter to Faculty and Staff (Oct 2002)*. http://www.clemson.edu/pres/newsltrs/102202.htm

BECKER, K. 2005. Cutting-edge Research by Undergraduates on a Shoestring? *Journal of Computing Sciences in Colleges*, 21(1), 160-168.

BEN-ARI, M. 1998. Constructivism in computer science education. *SIGCSE Bulletin*, 30(1), 257-261.

BEAUBOUEF, T. AND MASON, J. 2005. Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations. *SIGCSE Bulletin*, 37(2), 103-106.

BOUD, D., AND FELEITI, E. 1991. *The Challenge of Problem-Based Learning*, Kogan-Page, London.

BURGER, K. R. 2003. Teaching Two-Dimensional Array Concepts in Java with Image Processing Examples, *SIGCSE Bulletin*, 35(1), 205-209.

CUNNINGHAM, S. 2002. Graphical Problem Solving and Visual Communication in the Beginning Computer Graphics Course, *SIGCSE Bulletin*, 34(1), 181-185.

DAVIS, T. A., R. M. GEIST, R. M., MATZKO, S., AND WESTALL, J. M. 2004. τέχνη: A First Step, *SIGCSE Bulletin*, 36(1), 125-129.

DUCH, B., GRON, S., AND ALLEN, D. 2001. *The Power of Problem-Based Learning*, Stylus Publishing, LLC, Sterling, VA.

FELL, H. J., AND PROULX, V. K. 1997. Exploring Martian Planetary Images: C++ Exercises for CS1, *SIGCSE Bulletin*, 29(1), 30-34.

HUNT, K. 2003. Using Image Processing to Teach CS1 and CS2. *SIGCSE Bulletin*, 35(4), 86-89.

KATIRA, N., WILLIAMS, L., WIEBE, E., MILLER, C., BALIK, S. AND GEHRINGER, E. 2004. On Understanding Compatibility of Student Pair Programmers. *SIGCSE Bulletin*, 36(1), 7-11.

MARTINEZ, M. 1997. Designing Intentional Learning Environments, *Proceedings of the 15th Annual International Conference on Computer Documentation*, ACM Press, 177-178.

MCANDREW, A., AND VENABLES, A. 2005. A "Secondary" Look at Digital Image Processing. *SIGCSE Bulletin* 37(1), 337-341.

MORDECAI, B-A. 1998. Constructivism in Computer Science Education, *SIGCSE Bulletin*, 30(1), 257-261.

REINHARD, E., M. ASHIKHMIN, M, GOOCH, B., AND SHIRLEY, P. 2001. Color Transfer between Images. *IEEE Computer Graphic and Applications*, 21(5), 34-41.

WENDERHOLM, E. 2004. Challenges and the Elements of Success in Undergraduate Research. In *Working Group Reports From ITiCSE*, 73-75.

WICENTOWSKI, R., AND NEWHALL, T. 2005. Using Image Processing Projects to Teach CS1 Topics, *SIGCSE Bulletin*, 37(1), 287-291.