

A Graphics-Based Approach to Data Structures

Sarah Matzko and Timothy A. Davis
Clemson University
Department of Computer Science
Clemson, SC 29634
{smatzko—tadavis}@cs.clemson.edu

ABSTRACT

The underlying goal of the τέχνη project is to educate students in undergraduate computer science courses through the study of and solution to large-scale problems in computer graphics. Our ultimate aim is that this approach would be applied to all computer science courses in the B.A. curriculum. In the first years of this project, we have been working on the foundational sequence, which includes CS1, CS2, and CS3 (data structures and advanced programming). For this last course, which also includes the study of algorithms in our curriculum, we present an approach to teaching data structure concepts using advanced graphics algorithms. The results thus far have been promising, and we are continuing to evaluate and enhance the approach.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design, Experimentation.

Keywords

Computer graphics, problem-based learning, data structures, computer science education, curriculum issues.

1. INTRODUCTION

We live in a visually-oriented world. Today, that statement translates to a multi-billion dollar entertainment industry that includes film, television, and computer gaming. These markets are especially relevant to college students, who typically have a great deal of first-hand exposure to computer-generated images resulting in a natural interest in their production and use. We have attempted to take advantage of this interest by translating it to the educational domain. As a result, we have developed a new approach to computer science instruction, termed τέχνη, to teach basic concepts

through non-trivial problems in computer graphics in the undergraduate curriculum.

We believe computer graphics is a natural field to use to teach general computer science concepts due to the large amount of data inherent in graphics applications, as well as the rich set of problem types and complexities. Additionally, graphics programming projects allow students to uncover problems and evaluate solution correctness quickly through visual inspection. Finally, such problems garner student interest and allow them to choose not only how they will solve the problem in terms of algorithmic design, but how they will compose their final scenes in terms of artistic form and content.

The τέχνη approach was designed to span all courses in the curriculum; however, in this paper we focus on CS3, a data structures, for second-year students. Specifically, we describe the organization and content of the course, including the graphics projects. The programming assignments in the course build to a final project at the end of the semester. Results from these projects are shown throughout.

We start by describing the τέχνη project and our results thus far in Section 2. Section 3 presents related work describing similar approaches, while Section 4 provides the organization and rationale for the course. Section 5 discusses assigned projects and shows student results. Finally, Section 6 provides results and conclusions.

2. THE τέχνη PROJECT

Several years ago, we began work on the τέχνη project to build a curriculum where core computer science concepts would be taught through problem-based learning with graphics applications. The name τέχνη, which is the Greek word for art, shares its root with *τέχνηλογία*, the Greek word for technology. From these words, we can easily see the close academic relationship these two fields have held historically. The τέχνη project embodies the effort to reunite these areas for enhancing educational techniques.

The τέχνη project grew out of our experiences of establishing a new cross-disciplinary master's level program, termed digital production arts (DPA). The DPA program draws elements from computer science, art, theater, and psychology, among others, to educate students in computer animation and special effects production. Most of the graduates of the program pursue careers in the film or gaming industry, often acquiring positions prior to graduation. Some of the major studios employing our students include: Rhythm & Hues, Industrial Light & Magic, Dream-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ITiCSE'08, June 30–July 2, 2008, Madrid, Spain.

Copyright 2008 ACM xxxxxxxx.

Works, Pixar, Blue Sky, Tippett, Electronic Arts, and Sony Imageworks.

As mentioned above, DPA is a graduate-level program; however, undergraduates here at Clemson have shown an interest in the program, and have planned their academic curriculum and courses in preparation for applying. The pool of students from other schools, as well as from other countries, that apply to the program is also increasing.

One of the goals of the *τέχνη* project is to incorporate graphics problems of interest and research from the DPA and computer science programs in the undergraduate computer science curriculum. These problems take the form of semester-long projects in undergraduate courses leading to the B.A. in computer science. The overarching goal of *τέχνη* is to present computer science concepts in a more effective manner through graphics. We believe this approach is sound since it encompasses several education-theoretical techniques, including: visual feedback, problem-based learning [7], intentional learning [11], and constructivism [2] [14]. Through this approach, we hope to improve understanding of general computer science concepts, while engaging students through projects involving a relevant field of interest. This approach allows students to explore new concepts as they naturally arise in these large-scale graphics problems. Accordingly, instruction is problem-based, with course projects ranging from traditional problems in graphics, such as ray tracing, to cutting-edge topics from current research.

Computer graphics has been shown to be a good application area for problem-based learning [5] [6]. With constructivism as a basis, the approach starts at a basic level to provide students with experiences that facilitate the construction of knowledge. Problem-based learning "is a learner-centered approach in which learning episodes are motivated by an initial problem that bears some resemblance to 'real world' problems" [3]. The graphics projects assigned in *τέχνη* courses indeed constitute real-world graphics problems. In solving them, students construct knowledge as they encounter roadblocks to the final goal. The *τέχνη* approach extends this concept through its scope, which spans the undergraduate curriculum. Our hope is that this approach will ignite excitement in students to explore real-world problems, while providing them with the skills to do so.

At this point, we have offered several courses using the *τέχνη* approach, including a four-course sequence for incoming students. This sequence begins with CS1, in which we use a semester-long image processing project to introduce basic concepts in computer science. Stages of the project include: basic image generation, gray-scaling, convolution filtering, and image recoloring. To complete the final project, first-semester freshman actually read a published research paper [16]! We use C to teach this course for a variety of reasons [9] [15]. Additional information can be found in [11].

The second course in the sequence, CS2, requires the students to learn more advanced concepts in programming through a semester-long project to implement a ray tracer, an assignment usually reserved for a graduate-level graphics course. The ray tracer provides an ideal pedagogical platform for problem-based learning for a variety of reasons. First, ray tracer implementation covers a wide range of concepts, which must be mastered to complete the project. Second, visual feedback is available at all stages, allow-

ing program correctness to be determined quickly. Additionally, graphical objects, such as spheres, correspond directly to objects in code, thus leading naturally to an object-oriented implementation. Accordingly, students begin programming in C in this course and transition to C++ for the final project. Full details of the course are reported in [19].

In the fourth course in the sequence, Tools and Techniques for Software Development, students make the complete transition to object-oriented development with Java. Here, the semester-long project involves the implementation of a three-dimensional chess game, which can be played across a network. Programming game pieces and their movement aid in the instruction of objects, instancing, and their associated methods.

We have offered the above courses with excellent results. Students seem to be engaged in the courses, as evidenced in the impressive projects they produce, some of which rival work in our graduate-level courses. Student evaluations for these classes rate them highly, with students citing visual feedback and the ability to show impressive results to friends and family as strong positives. The *τέχνη* approach, as applied to CS3, is described in the following sections of this paper.

3. RELATED WORK

Most of the relevant work in this area involves undergraduate teaching through graphics-related topics, such as image processing. These techniques are becoming more popular since they are easily adaptable to teaching computer science, and engaging to undergraduate students. Further, such projects are interesting to students from elementary school [13] to college [18] [1] [4] [8] [10] since students enjoy visual real-world problems. Coupled with problem-based learning, the projects demonstrate the necessity of complex programming techniques. Additionally, the projects are more difficult for students to create hardcoded solutions for due to the large amount of data in graphics images; therefore, students must design and implement general solutions to problems [18].

Previous work using graphics applications for teaching has been limited in scale. Some approaches provide GUI environments [1], function prototypes [18], and pre-written functions [1] [4] [8] [10] to handle image file input and data manipulation. CS3 under *τέχνη* is unique in that all assignments are graphics-based and build toward a large final project. Typically, we also require students to write all code for these projects; however, starter code is provided in CS3 to allow students to focus on the implementation and testing of data structures. Further, the starter code provided consists of code that most students in the course programmed in CS1 and CS2.

4. COURSE CONTENT AND RATIONALE

As part of the *τέχνη* curriculum, CS3 is based on a semester-long graphics project: photon mapping. As mentioned previously, the goal is not to teach computer graphics topics, but to use interesting graphics problems to teach computer science topics, specifically, data structures and algorithms. To accomplish this goal, *τέχνη* incorporates components from several educational approaches, including constructivism, intentional learning, problem-based learning, and visual feedback. Our hope is that this approach will create a learning environment that motivates, broad-

dens, and supports students at all levels without additional load to faculty.

The project chosen for this course is photon mapping, an advanced computer graphics technique for computing global illumination in a user-defined scene [17]. Photon mapping is an extension of ray tracing, which fits nicely in our program since students write a ray tracer in the course immediately preceding this one. Further, it is a non-trivial application involving the processing of large amounts of data, and is usually reserved for upper-level courses.

The basic idea is that photons are emitted from light sources in the scene as a pre-processing step. As they travel, their paths are traced and any intersections with object surfaces are computed. At each intersection, the photon may be absorbed by the surface, bounced in a computed direction, or scattered in a random direction. During rendering, to compute the illumination of a surface point, photons within a small radius are gathered and counted. This approach simulates the way photons behave in the real world and results in images with high levels of realism. One of the complexities of the algorithm is that it requires a large number of photons to be emitted, often numbering in the hundreds of thousands, or millions.

The next section covers the photon mapping projects assigned in the course, as well as student images (interesting or artistic images were awarded extra points to encourage creativity among the students). Course projects were supplemented with short projects covering additional data structures in the lab portion of the course.

5. PROJECTS

The CS3 projects for the course build sequentially to the final project, with each project working toward the competency needed for the next.

5.1 Random Photon Lighting Project

The first project required students to modify several code classes to generate simple images with photon mapping applied to discrete areas or specific surfaces. As mentioned previously, starter code was provided to give the students a starting point for the project. The surface area for mapping photons was limited since, at this stage, the students were only beginning to learn about the photon mapping technique and how to use the code. The project required the students to place photons on the surface(s) randomly, rather than have the photons originate from a light source, bounce around the environment, and collect on surfaces in a manner closer to the real behavior of photons in the real world. Additionally, students were required to program the gathering of photons locally and produce final pixel colors based on photon placement.

Given the constraints of this first project, the submitted images are fairly simple and show only illumination from pseudo-random photon mapping. Sample images from this project are shown in Figure 1. Students were able to verify the correctness of their programs simply by observing the final image and ensuring that the selected surface was more brightly illuminated over the others.

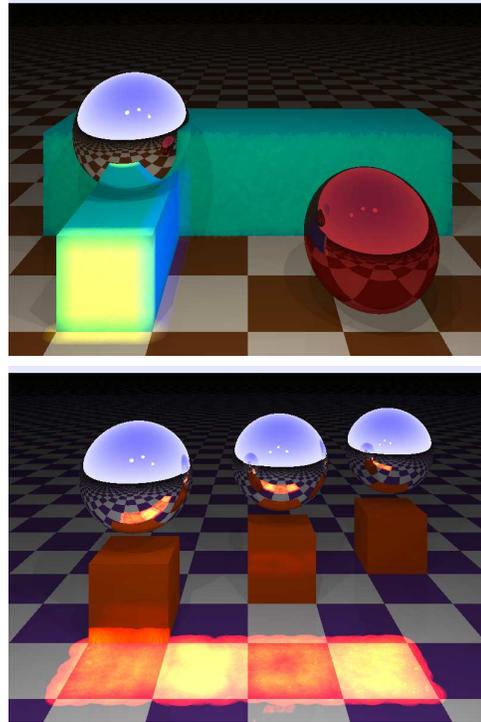


Figure 1: Student images from Random Photon Lighting Project.

5.2 Heap-Based Random Lighting Project

The second project focused on a specific data structure: the Max-Heap. For this project, students were required to modify the code such that photons would be stored in a MaxHeap data structure. Accordingly, students used an array to store the data in a heap, and wrote access functions to insert and delete photons in the heap.

In this scheme, when the photons are needed to illuminate a certain spot in the scene, the heap is queried to find the given number of photons that are closest to this spot. For large numbers of photons, the MaxHeap is not the best data structure for this task since it requires a large amount of time to generate final results. Indeed, computation for a full scene would require an enormous amount of runtime (tens of hours) if no further efficiency enhancements were undertaken. This situation, however, is exactly what we want to illuminate.

Specifically, one goal of this project is to relate to students the importance of selecting the right data structure, and the algorithms that accompany it, for a given application. In this project, if a data structure or algorithm is chosen unwisely, the consequences will become apparent in terms of execution time.

Sample images for this project are similar to those shown for project 1, since only the underlying data structure was changed, not the method for illuminating the scene.

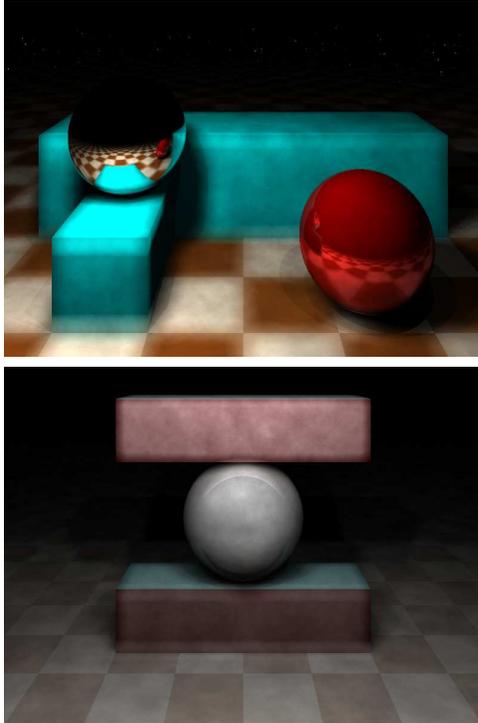


Figure 2: Student images from kd-Tree Photon Mapping Project.

5.3 kd-Tree Photon Mapping Project

The third project introduced substantial changes to both the illumination algorithm and the underlying data structure. Specifically, student programs were required to emit photons from a light source in the scene, rather than simply placing them randomly without regard to illumination sources. To reduce the complexity of photon behavior, only first-order intersections were considered, i.e., emitted photons did not bounce, but were absorbed by the first surface of intersection. This surface was determined by tracing the path of the photon from the light source along its initial direction, which was determined randomly as an approximation to the realistic behavior of photons.

In the second part of the project, the students replaced the Max-Heap data structure with a kd-tree. The kd-tree is a multi-dimensional binary search tree and represents an advanced data structure for the students. Further, the kd-tree is a natural fit for storing photon locations as it partitions three-dimensional space (similar to a DSP tree) in an efficient manner. With this modification, the execution time fell from hours to less than an hour for most scenes, which was well received by the students. At this point, students have first-hand understanding of differing data types and why one is chosen over another, a result which occurred naturally during the course of the project. This kind of discovery is extremely beneficial to learning and strongly reinforces course lecture.

Figure 2 shows images from this phase of the project. Since light does not bounce, all illumination shown is first-order and obviously lacking with regard to realism. Such images appear rather flat and poorly lit.

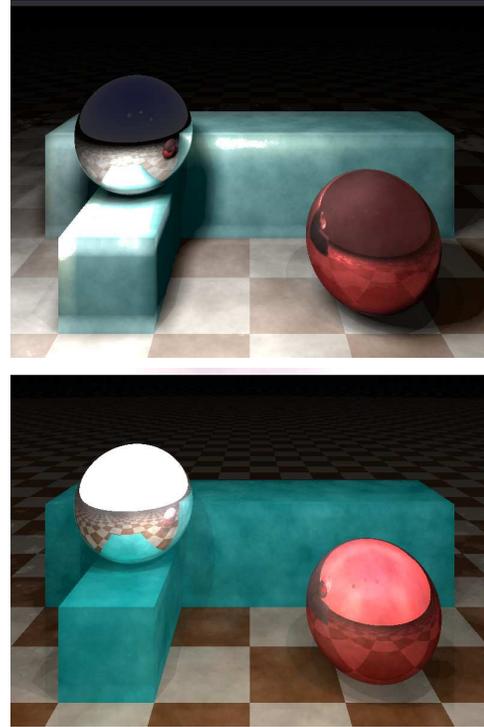


Figure 3: Student images from Complete Photon Mapping Project.

5.4 Complete Photon Mapping Project

Now that a reasonable data structure is in place, it is time to enhance the algorithm to produce more realistic lighting effects. Consequently, the fourth project added photon bounce and scatter, i.e., as photons paths are traced through the scene, a decision is made at each surface intersection as to whether the photon is absorbed, bounced in a predictable direction, or scattered in a random direction, based on a probability technique known as Russian Roulette. Once again, this procedure was adopted to mimic the way photons behave in the real world.

The result of these changes was additional light travel in the scene, which ultimately produced more illumination on the surfaces. Figure 3 shows project images from this phase.

5.5 Balanced kd-Tree with Caustics Project

This final project added enhancements both to the graphic results and the underlying data structure. First, the kd-tree required balancing to aid in reducing execution time. Students were still waiting for long periods of time to produce single image results. With the balanced kd-tree, execution time was reduced by 50% or more. When measured in terms of tens of minutes, these time reductions were once again welcomed.

Second, student programs must handle caustics, or concentrated patches of light due to reflection or refraction. Since the semester was drawing to a close at this point, few students were able to complete this portion; however, a student image with caustics (seen in the reflection) is shown in Figure 4.

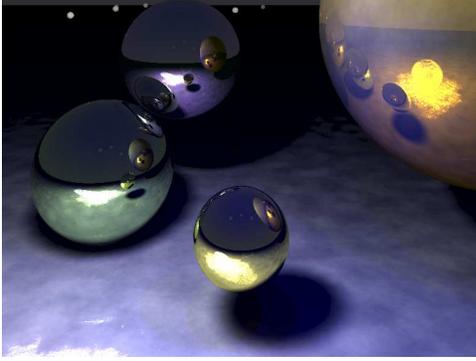


Figure 4: Student image from Balanced kd-Tree with Caustics Project.

6. RESULTS AND CONCLUSION

Preliminary results indicate that the assignments were engaging, yet challenging, for sophomores. The course consisted of two sections of 20 students each. Pre-tests and post-tests evaluating the coding ability of students in the course show marked improvement upon completion.

While the course was considered successful, at least two areas of instruction require attention. First, in an effort to prevent the starter code from being shared with students in other classes, the instructor released the code as object files. The decision caused problems for students accustomed to working from home; thus, additional object files were made available to work within Linux and MacOS environments. Unfortunately, compiler versions varied across machines, which created additional problems. Also, some students were frustrated by their inability to look "under the hood" in order to better understand how the program components worked together.

A second problem was the attempt on the part of the instructor to grade and debug each program submitted. While well-written programs could be graded quickly, students who were struggling did not implement the efficient algorithms and data structures, producing programs that ran for up to two days(!). Finding the errors in these programs required a great deal of detailed study of the code, resulting in a large latency time between submission and feedback.

Even with these issues, most students were engaged in the course as they completed assignments. Overall, students appeared to enjoy seeing the results of their work in an attractive way and considered the class a rewarding experience.

Problem-based learning with graphics applications provides an exciting way to introduce students to data structures and algorithms, while perhaps broadening their creative abilities by allowing them to produce images of their own design. Through the τέχνη project, we are continuing to provide new courses with this unique problem-based approach. Our focus now, however, is on upper-level undergraduate courses, such as networking and database design.

7. ACKNOWLEDGMENTS

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

8. REFERENCES

- [1] Astrachan, O., and Rodger, S. H. Animation, visualization, and interaction in CS1 assignments. *SIGCSE Bulletin*, 30(1), 1998, 317-321.
- [2] Ben-Ari, M. Constructivism in computer science education. *SIGCSE Bulletin*, 30(1), 1998, 257-261.
- [3] Boud, D., and Feleiti, E. *The Challenge of Problem-Based Learning*, Kogan-Page, London, 1991.
- [4] Burger, K. R. Teaching two-dimensional array concepts in Java with image processing examples. *SIGCSE Bulletin*, 35(1), 2003, 205-209.
- [5] Cunningham, S. Graphical problem solving and visual communication in the beginning computer graphics course. *SIGCSE Bulletin*, 34(1), 2002, 181-185.
- [6] Davis, T. A., Geist, R. M., Matzko, S., and Westall, J. M. τέχνη: a first step. *SIGCSE Bulletin*, 36(1), 2004, 125-129.
- [7] Duch, B., Gron, S., and Allen, D. *The power of problem-based learning*. Stylus Publishing, LLC, Sterling, VA, 2001.
- [8] Fell, H. J., and Proulx, V. K. Exploring Martian planetary images: C++ exercises for CS1. *SIGCSE Bulletin*, 29(1), 1997, 30-34.
- [9] Hu, C. Rethinking of teaching objects-first. *Education and Information Technologies*, 9(3), 2004, 209-218.
- [10] Hunt, K. Using image processing to teach CS1 and CS2. *SIGCSE Bulletin*, 35(4), 2003, 86-89.
- [11] Martinez, M. Designing intentional learning environments. *Proceedings of the 15th Annual International Conference on Computer Documentation*, ACM Press, 1997, pp. 177-178.
- [12] Matzko, S., and Davis, T. A., Teaching CS1 with graphics and C, *SIGCSE Bulletin*, 38(3), 2006.
- [13] McAndrew, A., and Venables, A. A "secondary" look at digital image processing. *SIGCSE Bulletin*, 37(1), 2005, 337-341.
- [14] Mordecai, B-A. Constructivism in computer science education. *SIGCSE Bulletin*, 30(1), 1998, pp. 257-261.
- [15] Reek, M. M. A top-down approach to teaching programming, *SIGCSE Bulletin*, 27(1), 1995, 6-9.
- [16] Reinhard, E., Ashikhmin, M., Gooch, B., and Shirley, P. Color Transfer between Images. *IEEE Computer Graphic and Applications*, 21(5), 2001, 34-41.
- [17] Wann Jensen, H., *Realistic image synthesis using photon mapping*, A.K. Peters, 2001.
- [18] Wicentowski, R., and Newhall, T. Using image processing projects to teach CS1 topics. *SIGCSE Bulletin*, 37(1), 2005, 287-291.
- [19] Davis, T. A., Graphics-based learning in first-year computer science, (Education Program), *Eurographics 2006*, 2006.