Teaching Applied Computing without Programming: A Case-Based Introductory Course for General Education

Joe Marks, William Freeman, Henry Leitner

TR2001-06   December 2001

Abstract

We introduce general-education students to key ideas in applied computing through case studies from computer graphics, computer animation, image processing, computer vision, information retrieval, and artificial intelligence. Each case study consists of two lectures: one an intuitive exposition of relevant computer-science concepts, and the other a hands-on introduction to a working system that embodies these concepts. Students use these systems to perform design and problem-solving tasks, thereby reinforcing the abstract concepts presented. Computer programming is neither required nor taught. The course has been offered for two years at the Harvard University Extension School, and has achieved high ratings in student surveys.

Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education, February 2001

Teaching Applied Computing without Programming: A Case-Based Introductory Course for General Education

Joe Marks and William Freeman
MERL — Mitsubishi Electric Research Labs
Cambridge, MA 02139
{marks, freeman}@merl.com

Henry Leitner
Harvard University Extension School
Cambridge, MA 02138
leitner@fas.harvard.edu

Abstract

We introduce general-education students to key ideas in applied computing through case studies from computer graphics, computer animation, image processing, computer vision, information retrieval, and artificial intelligence. Each case study consists of two lectures: one an intuitive exposition of relevant computer-science concepts, and the other a hands-on introduction to a working system that embodies these concepts. Students use these systems to perform design and problem-solving tasks, thereby reinforcing the abstract concepts presented. Computer programming is neither required nor taught. The course has been offered for two years at the Harvard University Extension School, and has achieved high ratings in student surveys.

1 Introduction

Computer science is currently one of the most exciting and dynamic disciplines. Yet students often perceive introductory computer-science courses as difficult and dull, especially those students who are motivated mainly by intellectual curiosity. We propose two reasons why introductory computer-science courses often fail to engage such students. One is a focus on computer systems instead of computing applications. The second reason is an emphasis on computer programming. It is as if we insist on teaching auto repair to teenagers who just want to learn how to drive.

We have developed an alternative introductory course that emphasizes computer applications over computer systems, and hands-on exploration with implemented systems over tedious programming exercises. Our course is organized around case studies chosen from the most exciting areas in the field of applied computer science, such as artificial intelligence, computer graphics, computer vision, information retrieval, and human-computer interaction. Each application is the subject of two lectures: the first provides an intuitive overview of the computer-science concepts involved and the second concentrates on specific details of the application. Each case study is tied to a particular software system. As their assignment for each unit, students are required to use the software systems for design and problem-solving tasks, thereby reinforcing the concepts presented in the first half of each case study. Programming is not required, nor is it taught.

Other educators have attempted to make introductory computer science more interesting by surveying selected topics from more advanced courses. Holmes and Smith describe a CS1 curriculum in which half of the course is devoted to a survey of computing concepts such as text compression, resource scheduling, searching, sorting, and graph algorithms; the remainder of the course is devoted to traditional programming fare [1]. Perhaps the best-known work in this area is that of Alan Biermann, who presents a number of the intellectual achievements in the field of computer science in his *Great Ideas in Computer Science* textbook [2]. Prof. Biermann's viewpoint is that students learn best by doing; they are thus asked to write relatively simple programs, to design circuits, code assembly language, hand-simulate a compiler, and even to work with programs that elucidate the problem of noncomputability. Biermann effectively takes some rather complex and technical "great ideas" and make them comprehensible to nonspecialists. Another related approach is due to Bell, Witten, and Fellows [3]. They attempt to teach advanced computer-science concepts to children by means of simple games and activities that do not require a computer.

In contrast to these previous efforts, our focus is on adult students; we eschew programming completely; our advanced concepts are selected solely from the realm of applied computing; and we present these concepts in the context of complete on-line applications with which the students can conduct in-depth exploration and experimentation.

In the rest of the paper we outline each case study in our current curriculum, and discuss our experience with this kind of course. More detailed information can be found on the course web site [4].

2 The Case Studies

We chose our case studies primarily to cover an adequate subset of the key areas in applied computing and to provide a representative sampling of specific concepts in those areas. Table 1 contains a summary of the areas and concepts associated with each case study.

Case Study	**Areas – *Concepts***
Ray Tracing	computer graphics – *geometric modeling, light transport and reflection* computational geometry – *computing intersections, geometric searching*
Animated Particle Systems	discrete-event system simulation – *random numbers* numerical methods – *numerical integration*
Interactive Optimization	computational complexity – *algorithm and problem complexity, the Traveling-Salesman Problem, NP-completeness* artificial intelligence – *heuristic search and optimization* probability and statistics – *empirical analysis of algorithms* human-computer interaction – *design of cooperative user interfaces*
Image Enhancement	electronic imaging– *image sensing and representation* image processing – *point operations, image filtering, noise removal*
Face Recognition	computer vision – *shape recognition, shape tracking, motion analysis* human-computer interaction – *camera-based interfaces*
Information Retrieval on the WWW	classical data processing – *relational databases, efficient sorting and searching* information retrieval – *vector-space model for term sets, inverse indices, link analysis, semantic nets, collaborative filtering*

Table 1: Summary of the case studies

Our second selection criterion was that each case study be accessible to students who might never have used a computer for anything other than word processing or web browsing. The key to accessibility is the availability of software systems that are suitable for hands-on exploration and experimentation by novices. Also, these systems should do something intrinsically interesting and fun. For three of our case studies we were able to find suitable commercial or freeware systems. For the other three, we developed our own software.

A final criterion was that the case studies be in areas in which we were knowledgeable beyond the elementary level. The research careers of the two lecturers (JM and WF) have involved computer graphics, computer vision, image processing, human-computer interaction, artificial intelligence, and operations research, so these are the areas from which we primarily selected our material.

In the following subsections we describe each case study in terms of its topic, the areas of applied computing that it represents, the key concepts in these areas that are used in the accompanying software systems, and a description of those systems.

2.1 Case Study #1: Ray Tracing

Ray tracing is a conceptually simple but very powerful technique for generating synthetic imagery [5]. Ray tracing embodies an *inverse-camera* model in which rays are cast from an eye point into a virtual scene. For each picture element, or *pixel* in the image, a ray is cast into the scene. The interactions of these rays with the geometry in the scene determine the color of the pixel (see Figure 1). Although computationally expensive, ray tracing can produce synthetic images of stunning realism.

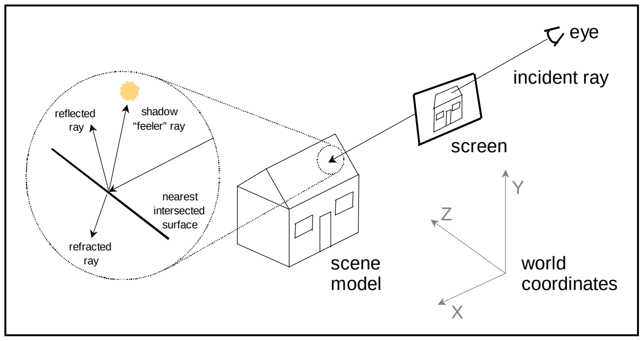

Figure 1: Ray tracing illustrated

The ray-tracing case study introduces the general area of computer graphics, and two fundamental computer-graphics concepts: the modeling of three-dimensional objects in terms of polygonal and curved surfaces; and the interaction of light with these surfaces. The case study also introduces the area of computational geometry, which provides the algorithmic basis for much of computer graphics. Within this area two important algorithmic concepts are presented: the computation of intersections (in this case, line-surface intersections), and geometric searching in three dimensions (in this case the search is for the nearest surface intersected by a ray).

Primed with an understanding of these concepts, students are ready to use a ray-tracing system to produce their own synthetic imagery. POVRAY is a free software package that uses ray tracing to generate 2D images from 3D scene descriptions [6]. As the assignment for this case study, the student must produce several images by using POVRAY's geometric-modeling language to create a 3D scene, to specify surface and material characteristics of object models, and to locate and orient lights and a camera.

2.2 Case Study #2: Animated Particle Systems

In computer animation, natural phenomena like fire, smoke, explosions, water, etc. are usually generated using particle systems [5]. A particle system comprises individual particles that are created, move, change color, disappear, and spawn other particles according to simple physical laws and user-supplied probability distributions. In the simplest systems, the particles are drawn as points of light on the screen for each frame of the animation. Figure 2 contains a frame from a particle-system animation produced with the system used in our case study.

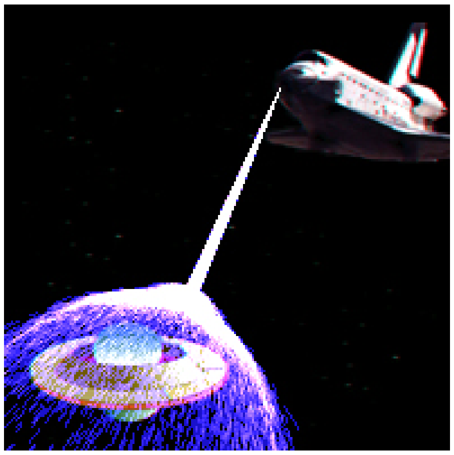

Figure 2: NASA's true mission illustrated with a particle system

The relevant concepts for this case study come from the areas of discrete-event system simulation (the generation of random numbers) and numerical methods (the use of numerical integration to simulate Newtonian physics). We developed our own simple particle-system software specifically for this course. In the assignment for this case study, students modify the probability distributions that affect the particles' motion and appearance to produce a broad range of animation effects.

2.3 Case Study #3: Interactive Optimization

This case study was the most difficult one to develop. We wanted to include concepts from computational complexity in our course, because they are at the core of all computer science. The challenge is to do that in a way that is as exciting and seductive as the other case studies. Our initial attempt was a case study that required the students to conduct an empirical investigation of various search heuristics for cartographic label placement, an NP-hard layout problem [7]. Although a very visual and familiar problem, student surveys showed this case study to be the least popular one in the first offering of the course.

Fortunately, one of the research projects underway at that time in our laboratory involved the development of interactive systems for human-guided search and optimization [8]. The goal of this research is to develop hybrid interactive systems that combine the brute-force searching ability of the computer with a human's visual perception, judgment, and experience to find better solutions to NP-hard optimization problems. One of our systems targets the problem of vehicle routing, so we chose that problem and our existing system as the basis for the third case study.

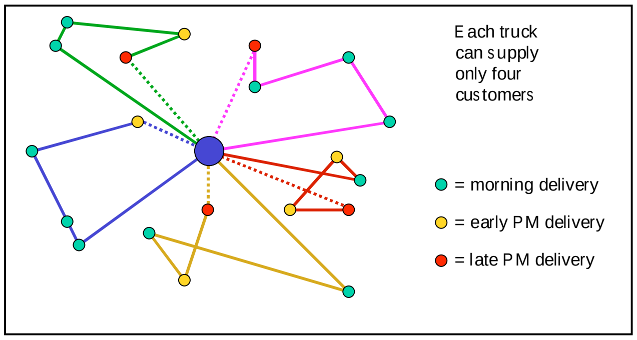

Figure 3: A solution to a simple CVRTW problem that shows the warehouse, customer locations, delivery constraints, and truck routes

The routing of delivery vehicles is one of the basic problems in supply-chain management. Capacitated vehicle routing with time windows (CVRTW) is one of the basic formalizations of the vehicle-routing problem. In CVRTW problems, trucks deliver goods from a central warehouse to customers at fixed locations. Each customer requires a certain quantity of goods, and specifies a time window within which delivery of the goods must commence. All trucks have the same capacity, and travel at the same speed. Each delivery takes the same amount of time, and each customer receives only one delivery. All trucks must return to the warehouse by a fixed time. Figure 3 shows one solution to a simple CVRTW problem. The optimization task is first to minimize the number of trucks required to service all the customers; and second is to minimize the total distance traveled. CVRTW is thus a significant generalization of the Traveling Salesman Problem, and therefore also NP-hard.

In the concepts lecture for this case study we present elements of the classical theory of computational complexity: algorithm and problem complexity, the Traveling-Salesman Problem, and NP-completeness. We also introduce the area of artificial intelligence, focusing on the concept of heuristic search and optimization. We briefly survey probability and statistics, and show how simple ideas from these fields are useful in the empirical analysis of algorithms. And finally we describe the area of computer-human interaction, and focus specifically on issues that relate to the design of cooperative user interfaces. To reinforce these concepts the associated assignment involves the use of the aforementioned interactive system to determine the minimal sizes of truck fleets and efficient delivery schedules for given CVRTW problems. Students use the system's capabilities to focus the computer's search on selected subspaces of the problem space to avoid local minima and to expend the computer's cycles where they think they can make the most difference. The quantitative nature of this task allows for some friendly competition to take place!

2.4 Case Study #4: Image Enhancement

Many of the images we see in our daily lives are digitally enhanced in some way. This case study examines many of the common techniques used to modify or enhance images using a computer. In class lectures, we describe simple methods to modify the tonescale, sharpen, and de-noise images. We use a commercially available image-processing program, Adobe Photoshop, to study these image-enhancement operations.

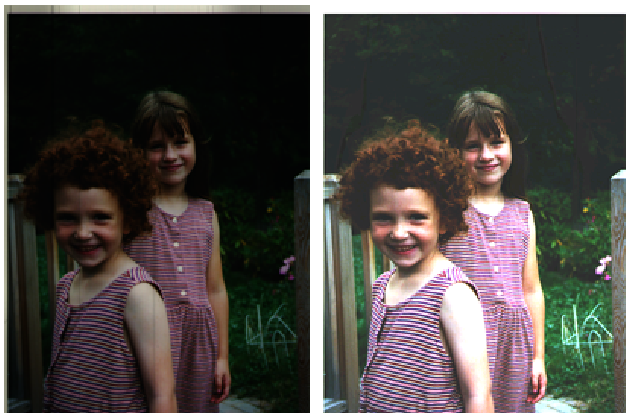

Figure 4: Image-processing example, simulating possible image-enhancement steps for a slide scanner

The students enhance several supplied images, simulating the image-processing steps that might take place in a photographic slide digitizer (see Figure 4). In addition, students modify an image to "hide" information in some reversible way of their own design. Other students then try to recover the original image from the obscured images made by their classmates.

2.5 Case Study #5: Face Recognition

Computers are typically blind to the person who is using them, but that will change soon. In the near future, computers will recognize faces, identify people from the pattern of their irises, and interpret a user's movements, gestures, and glances. The concepts lecture for this case study reviews the state of the art in computers analyzing human activity and recognizing people. Fundamental visual measurements that we cover include tracking, shape and object recognition, and motion analysis.

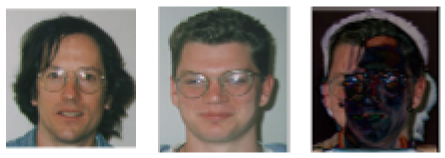

Figure 5: Two fairly dissimilar faces (left and middle) and the absolute value of their difference image (right), used in computing a similarity score

The application project is face recognition. The simplified face-recognition algorithm we use is simple enough to explain in one lecture, yet works surprisingly well on a small database of 40 or so images of faces of volunteers from the class. We also cover how to measure performance for recognition tasks. The students use the program to study the effect of different image-similarity metrics on face-recognition performance. Figure 5 shows the first step for one metric. Multiple images of the same individuals in the database demonstrate the algorithms' sensitivity to lighting, facial expression, and head pose.

2.6 Case Study #6: Information Retrieval on the WWW

One way to consider the World Wide Web is as a large text database, albeit one that is constantly changing and haphazardly organized. One way to find useful information on the Web is with a good search engine. In this case study the students perform a comparative analysis of several different search engines for a variety of information-gathering tasks. Another way to find information on the Web is through the use of *collaborative* or *social filtering* to find documents (or products) that other users with similar interests or needs found relevant. Students are also given several tasks to perform with a "recommender" system that use collaborative-filtering techniques.

Although classical database theory arguably has little to do with finding information on the Web, we begin by surveying this area and by studying some of its key concepts: relational-database theory, and searching and sorting algorithms. This initial exposure to standard relational databases sets up a useful contrast when we then introduce the area of information retrieval [9], with its focus on freeform text data. We present the two key concepts behind information retrieval (the vector-space model for term sets and inverse indices) and also a selection of more recent ideas (link analysis, the use of semantic nets, and collaborative filtering).

3 Discussion

Each case study is presented in four hours of lecture spread over two class meetings, one per week for twelve weeks. Two additional hours per week of teaching-assistant office hours bring the total number of contact hours for the semester to around 50. There are no exams, because we feel that exams do not fulfil a useful motivational or educational goal for this kind of course. Instead, the students' assignment scores determine grades.

In the first two years of the course student grades have been very high, because most students complete all of the assignments successfully. We believe that most of the assignments would be impossible to complete without a firm grasp of the underlying ideas, so we feel confident that most students leave the course with a good understanding of the concepts behind each of the case studies.

Furthermore, the students themselves report positively on their learning experience in the course and therefore rate the course very favorably. The following student comment typifies the general feeling of the participants near the end of the semester: "It was thought-provoking and inspirational … I left class every week excited about the future of computing and amazed at the possibilities being explored presently." Note that the Harvard University Extension School serves a diverse student body spanning an age range from early teens to the early nineties, with the average being a working adult of 32 years; they are a motivated and demanding group, with many students commuting after work from far away. Most of our students enroll for self-enrichment; a small percentage of the students use it as an elective in a liberal arts degree program.

4 Conclusion

The goal of our course is to impart a broad understanding of the concepts essential to several substantial computer applications in a popular-science format. We believe that these concepts represent the intellectual core of applied computer science, and are more relevant for students who want to know what computers can do and how they do it, but who do not plan on becoming computer programmers. For us, the course has been one of the most rewarding experiences in our teaching careers; student feedback has been extremely positive. Although the course is currently offered only in an adult-education context, we believe it can be modified to serve as an alternative introductory or survey course for college "non-majors."

5 Acknowledgements

Special thanks to the course teaching assistants: Denny Bromley, Emily Anderson, and Lisa Friedland. And thanks also to Brad Andalman, Emily Anderson, Neal Lesh, Brian Mirtich, David Ratajczak, and Kathy Ryall for software development. Seed funding for the development of course materials was provided by Harvard University Extension School.

References

[1] Holmes, G. and Smith, T.C. Adding some spice to CS1 curricula, in *Proceedings of SIGCSE'97*, San Jose, California, Feb. 1997, 204-208.

[2] Biermann, A. *Great Ideas in Computer Science*, The MIT Press, 1990.

[3] Computer Science Unplugged: Off-line activities and games for all ages. http://unplugged.canterbury.ac.nz/

[4] CSCI E5: An Introduction to Applied Computer Science, Harvard University Extension School. http://lab.dce.harvard.edu/extension/cscie5/E52000/E52000.html

[5] Foley, J., van Dam, A., Feiner, S., and Hughes, J. *Computer Graphics: Principles and Practice, 2nd ed.*, Addison Wesley, 1996.

[6] The Persistence of Vision Raytracer. http://www.povray.org/

[7] Christensen, J., Marks, J., and Shieber, S. An empirical study of algorithms for point feature label placement, *ACM Trans. on Graphics, 14(3)*, July 1995, 203-232.

[8] Anderson, D., Anderson, E., Lesh, N., Marks, J., Mirtich, B., Ratajczak, D., and Ryall, K. Human-guided simple search, in *Proceedings of AAAI 2000*, Austin, Texas, Aug. 2000, AAAI Press, 209-216.

[9] Salton, G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison Wesley, 1989.