

AC 2009-2188: A CREATIVELY ENGAGING INTRODUCTORY COURSE IN COMPUTER SCIENCE THAT GENTLY MOTIVATES EXPLORATION OF ADVANCED MATHEMATICAL CONCEPTS

Eric Freudenthal, University of Texas, El Paso

Eric Freudenthal is an Assistant Professor of computer science at the University of Texas at El Paso.

Mary Kay Roy, University of Texas, El Paso

Mary "Kay" Roy is on the adjunct faculty of computer science at the University of Texas at El Paso.

Alexandria Ogrey, University of Texas, El Paso

Alexandria N. Ogrey is a B.S. candidate studying computer science at the University of Texas at El Paso.

Ann Gates, University of Texas, El Paso

Ann Q. Gates is a Professor of computer science at the University of Texas at El Paso.

A Creatively Engaging Introductory Course in Computer Science that Gently Motivates Exploration of Mathematical Concepts

Abstract

We describe reforms to a highly engaging algorithm-centric introductory course in media programming offered to pre-engineering students at the University of Texas at El Paso, an urban Hispanic-serving institution (HSI) as part of a required entering students program.

In order to become eligible to attend the introductory programming course that begins the computer science degree plan at UTEP ("CS-1"), a large fraction of incoming freshmen must attend several semesters of preparatory "pre calculus" math courses. Most of these students will have limited if any prior exposure to programming or engineering. The initial implementation of our course was intended solely to provide an engaging first experience with programming, and followed Mark Guzdial's "Media Computation" curriculum. Dr. Guzdial's curriculum has successfully engaged Liberal Arts students in programming through the creation of aesthetically motivated multimedia projects. Attendees in pre-engineering and pre-professional programs reported lack of interest in these aesthetically- focused projects and requested more practical projects and assignments. The course has been modified to focus more on the design of algorithms in a manner that exposes foundational mathematical concepts.

Introduction

This paper describes the retargeting of an engaging media programming course developed by Mark Guzdial at Georgia Tech and was originally intended for liberal arts students.¹ Our course, informally titled "Multimedia Exposed," primarily targets freshmen intending to study engineering or computer science. Although the objectives and approaches of the reformed course are quite different from Guzdial's, the evolution was not expected or intended, but instead was the result of a series of incremental reforms motivated by our observations of student interests and needs.

The first variant of this course was offered in Fall 2007. It was taught by Computer Science faculty in consultation with staff of the University's career guidance center. Early results have been very promising. Many students intending to study engineering and computation find the course both enjoyable and engaging, and appear to be highly motivated towards continuing in this direction. We are conducting a longitudinal study to determine the effectiveness of this course in improving student success in CS and Engineering.

In order to engage a large number of freshmen, the course is incorporated into a required first semester "University Studies" program designed to teach skills necessary for academic success and to provide career guidance. Students attending this course are provided an accessible early exposure to simple dynamic systems simulations in a manner that includes both programming and mathematical modeling. Students' reactions to these experiences support the course's career guidance components - which provide opportunities for students to adjust their choice of major early in their academic careers.

In the original design of the course, Guzdial stated that he expected the students taking the course would become competent in modifying code, but not unlikely to become professional programmers.¹ Guzdial selected media computation because intended students of this course were more likely to use computers for communication than for computation.² Since one of the benefits the students would receive from learning programming from a programming course is the development of problem-solving skills, we feel that our adaptations contribute to this objective.

Much of Guzdial's course initially examines the manipulation of entire raster images. By relying on opaque pixel iterators that enumerate all pixels without regard to location, Guzdial's course delays exposure to Cartesian coordinates and avoids the need for nested row-column iteration. As a result, enumeration and pixel-access mechanisms taught in early labs were ill-suited for projects that generate numerically sequential values.

While teaching Guzdial's multimedia programming course in its original form for two semesters, we observed that even non-engineering students were already comfortable with Cartesian coordinates. We modified the graphical library to include a new "Raster" class which provides only random-access row-column accessor functions that conveniently represent coordinates and Red-Green-Blue pixel contents as Python tuples.

This permitted the course to begin with an immediate exposure to nested loops, a topic that is difficult for many students to master in CS-1. In contrast, when introduced to nested looping in this 'natural' context, students expressed no confusion and they were self-motivated to experiment with adjustments to ranges and strides. Furthermore, students were not intimidated by these explicit uses of tuples to represent RGB triples and more quickly designed and constructed novel algorithms using these natural and compact abstractions than the cohorts of students who first learned the object-oriented interface. This early and solid understanding of Cartesian pixel accessors enabled us to refocus the course from media manipulation to the use of a raster to plot and investigate mathematical functions and dynamic systems.

Table 1: Alternate implementations of a function examined in the first day's lecture that converts dark pixels to green..

Initial AWT interface using opaque "distance" function	Initial AWT interface with explicit range comparison	Modified Raster interface with explicit range comparison
<pre>def changeKimL(): file = r"F:\KimPossible.jpg" i = makePicture(file) green= makeColor(0, 255, 0) black = makeColor(0, 0, 0) for px in i.getPixels(): c = px.getColor() if distance(c, black) < 50.0: px.setColor (green) show (image)</pre>	<pre>def changeKimM(): file = r"F:\KimPossible.jpg" i = makePicture(file) green= makeColor(0, 255, 0) for px in i.getPixels(): r = px.getRed() g = px.getGreen() b = px.getBlue() if(r<40 and g<40 and b<40): px.setColor (green) show (image)</pre>	<pre>def changeKimR(): file = r"F:\KimPossible.jpg" p = Raster(file) green = (0, 255, 0) cols,rows = r.widthHeight() for x in range (cols): for y in range (rows): r,g,b = p.getRGB((x,y)) if r<40 and g<40 and b<40: p.setRGB((x,y), green) p.repaint()</pre>

Table 1 includes three versions of essentially the same function used in an initial programming exercise for all of our courses and approximates an early exercise in Guzdial's media programming text. These functions read a JPEG image of a familiar cartoon character who is wearing a black shirt and shoes, and then dramatically recolor all of the black pixels to green.

Object oriented programming techniques are important. It is also important that programming techniques in early courses should be chosen to minimize cognitive load while maximizing pedagogical value. We observe that object oriented (OO) features that have been widely adopted by the software development community to manage software comprehensibility can instead obscure the simple algorithm being implemented and impede student creativity. The functions in the left and middle columns reference the heavily OO “AWT” toolkit employed throughout Guzdial's book. Like the early examples in Guzdial's book, *ChangeKimL* (in the left column) is compact and expressive **only** because it uses AWT's opaque pixel iterator *getPixels()* and Guzdial's opaque color-distance function *distance()*, which computes Euclidean distance^{*}. We also observe that understanding of this function requires that students learn the relationship between the numeric representation of colors as R-G-B triples, and a *Color* object's interfaces, which are constructed using a function called *makeColor()*.

Mechanisms should be exposed only when they teach generalizable techniques or deepen understanding. The distance and color-object interfaces referenced by *ChangeKimL* draw student attention away from algorithm design but provide few, if any, pedagogical benefits. In contrast, students had little trouble understanding the code and spontaneously experimented with alternate decision criteria when working with *ChangeKimM* (in the middle column), a variant also written using AWT that exposes students to tuples, as well as Boolean and relational operators.

We subsequently developed an alternate Raster interface which is used by *ChangeKimR* (in the right column). Raster provides only random column-row addressing, and exploits Python's incorporation of tuples as a first class data type. Raster uses tuples to encode location (column, row) and color (red, green, blue). As a result, *ChangeKimR*'s doubly nested iteration is explicit, and colors are painlessly packed and unpacked. Like the other two functions, first-day students experience little trouble developing an intuitive understanding of *ChangeKimR*, and are highly motivated and able to modify it to recolor other objects within the cartoon image.

Furthermore, the techniques used for iteration and pixel access in *ChangeKimR* are directly transferrable to future projects in mathematics and dynamic systems.

As illustrated by *ChangeKimR*, programs written using the new Raster interface are compact and expressive. Our Raster class's constructor's parameters can specify either the name of a JPEG image file or the dimensions and initial color of a blank canvas. Pixels are always represented as 3-tuples and referenced using 2-tuples of column-row coordinates. This helps to reinforce the mental model that students form of the Raster image as a whole, and facilitates iterative processing of pixels through explicit numerical iteration. In order to facilitate projects that plot

^{*} The distance function (Euclidean distance) is examined in a later chapter of Guzdial's text. However, few students attending the course are engaged by an examination of the mathematical analysis of this metric in this context.

mathematical functions, raster's origin is located in the lower-left corner, and thus column-row addressing directly mimics x-y coordinates within the first quadrant of a Cartesian plane.

This use of random pixel accessors and numeric iteration is suitable both for problems that traverse all pixels and those that instead traverse a numerical series. This enables knowledge learned in early labs to be more easily transferred to subsequent projects. As a result, students can focus on conceptual exploration rather than linguistic constructs. In contrast, students attending the original course were confused when one method of iteration was used in early labs, and then another was introduced to solve slightly different problems at a later point. As illustrated above, our avoidance of opaque functions seems to elicit curiosity from students, and we find that even non-STEM students are well prepared and motivated to investigate "distance" metrics as a lab exercise which provides well-motivated practice with arithmetic, relational, and Boolean operators.

Focus on Algorithm Design

The primary mode of instruction in Guzdial's course is the presentation of detailed "recipes" provided within the text that are translated into functions. As illustrated in Table 1, the object-oriented graphical library provided to students is sufficiently complex that this translation is a non-trivial task. Furthermore, we find that the resulting code expansion can obscure the underlying algorithm. Since the modification and design of trivial algorithms that manipulate easily-understood systems appears to be well within all of our students' capabilities, and because algorithm design is one of Computer Science's primary forms of intrinsic creativity, we prefer to focus student attention on the design of algorithms rather than the application of linguistic constructs. From a teaching perspective, it seems preferable for students to (1) select a specific goal derived from a general range of possibilities, (2) design the algorithm, and (3) implement the steps for themselves.

A very wide range of students is engaged by Multimedia Exposed's short and dramatic imperative programs. Groups of students ranging in ages from middle-school to college who never previously programmed have been able to understand and modify simple imperative programs that directly examine and selectively manipulate the values stored within every pixel, even when exposure is limited to a single 45 minute session. While these students are surely not prepared to independently construct new programs, they are able to understand the goal of the program; the steps of the algorithm; the general flow of the lines of code; and the logical statements forming the criteria for changing a pixel's color. Students become deeply engaged in manipulating this concrete application of relational operators, logical operators, and color values themselves to select and apply color changes, and few require any assistance from the instructor. We suspect that the ability to directly observe the results of program execution enables students to quickly detect and correct misconceptions.

Relationship to Traditional CS-1 Courses and Extension to Mathematics

Multimedia Exposed's early use of standard programming constructs has made the integration of image manipulation with more traditional programming assignments in this course very smooth. Students learn many of the same basic programming skills as in a conventional first-semester

programming course, including the use of variables, assignment statements, function definitions, and iterative and control-flow structures. These concepts are applied consistently from project to project so that the programming skills are not tied to multimedia processing alone but are instead universally applicable.

Students gain understandings of advanced concepts surprisingly quickly. For example, during the third week of class, all students in a non-engineering section of Multimedia Exposed provided correct answers to a quiz focused on the manipulation of nested for- loop's range and stride.

As reported previously^{0,2} many students enrolled in technical and STEM programs lose interest in media manipulation solely motivated by aesthetic creativity and desire projects relevant to their anticipated careers.

Many of the students in STEM programs attending our course are freshmen, who are likely to spend their first three semesters in preparatory math courses prior to attending their first major's course. For these groups, our course focus shifts to an exploration of mathematics and the modeling of physical systems.

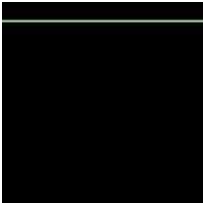
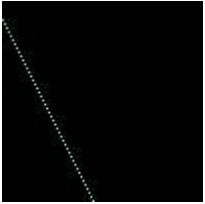
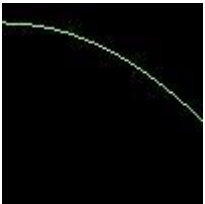

Extension to Mathematics and Physics

In response to student demands for increased technical depth and relevance to engineering curriculum, and to expand the set of graphical tools available for students to understand and apply, we added several new modules that examined algorithms that generate lines and curves. These projects were later expanded to include the examination of dynamic systems that model familiar physical phenomena. The approaches described below were motivated by our instructor's observations that non-engineering students had a strong aversion to explicit mathematical inquiry: Students were engaged by such projects only when (1) the project's connection to "mathematics" was initially obscured and (2) the distance between well-understood concepts and the project was narrow.

Numerous studies, including one conducted on students attending Multimedia Exposed and other pre-major's introductory courses at similar Hispanic-serving institutions⁵, indicate that students entering college have low confidence in their ability to understand higher math and physics. This low confidence can intimidate students choosing math-intensive academic programs such as Computer Science and other STEM disciplines, and may reduce students' ability to sustain motivation in such programs.

Fortunately, an entry-level programming course is well-positioned to increase student confidence and competence in basic mathematics through the construction and examination of simple programs. Such programs provide concrete examples of familiar concepts which can be explored first through direct modeling that illuminates and motivates mathematical abstraction. For example, Table 2 presents a sequence of assignments that incrementally examine (1) the drawing of lines, (2) the introduction of slope and (3) acceleration, and finally (4) bounce and exponential decay. Similar to the approach presented in Kalman's *Elementary Mathematical Models*,⁴ integration of linear summation to quadratics is initially presented through more familiar summation approaches.

Table 2. Successive projects drawing lines, curves (parabolae), and ballistic simulation.

<pre>def horiz(): numCols, numRows = 100,100 img = Raster((numCols, numRows)) row = 90 slope = 0 for col in range(numCols): row += slope img.setRGB((col, row), green)</pre>	
<pre>def down(): numCols, numRows = 100,100 img = Raster((numCols, numRows)) row = 90 slope = -2 for col in range(numCols): img.setRGB((col, row), green) row += slope</pre>	
<pre>def parabola(): numCols, numRows = 100,100 img = Raster((numCols, numRows)) row = 90 # initial position and slope slope = 0 rate = -.1 # acceleration for col in range(numCols): img.setRGB((col, row), green) row += slope slope += rate</pre>	
<pre>def bounce(): numCols, numRows = 100,100 img = Raster((numCols, numRows)) row = 90 # initial pos, slope, and accel slope = 0 rate = -.2 decay = .8 # decay at each bounce for col in range(numCols): img.setRGB((col, row), green) if (row <= 0 and slope < 0): slope *= -decay row += slope slope += rate</pre>	

Mathematical analyses of slope, y-intercept, linear sums, and quadratic closed-form representations occur only *after* students are familiar with the graphical subsystem, and have examined and described each dynamic property. Even then, analysis begins with discussion of observations and analysis that confirms and reinforces student understandings. Finally, each of these *now familiar dynamic systems* are reduced to closed form, first using geometric, and then algebraic representations

Thus, the simple programs that simulate familiar dynamic phenomena effectively become accessible and concrete “manipulatives” that limit the amount of abstraction exposed at each

stage of exploration. The final example provides an intuitively understandable concrete explanation of the familiar phenomenon of ballistic motion which is elusive to many students completing a course of college-level physics.³

Variants of Multimedia Exposed

The course has been presented to groups that vary widely according to major, academic experience, and mathematical background. Some sections are composed primarily of cohorts from pre-engineering and pre-science, while others are composed of mixed, primarily non-technical majors. One version of the course focuses only on programming, while another integrates programming into a freshman survival-skills course that reviews study, presentation, and analysis skills. Furthermore, a “one-shot” teaser has been presented multiple times to incoming freshmen and middle-schoolers attending a summer camp hosted by our university.

Pre-engineers

Pre-engineers and pre-science students are generally segregated into math-centric sections of Multimedia Exposed. In these sections, some students enjoy their multimedia manipulations but quickly demonstrate eagerness to move beyond basic projects. One bored student asked "is that all there is?" after constructing a complex function that produced multiple color changes in an image. As illustrated above, we are able to move quickly through a progression of mathematical functions from lines to parabolas that model familiar physical phenomena.

Non-engineers

While pre-engineering students tended to show greater interest in graphing of mathematical functions as the level of math increased and the functions became more complex (Ex: linear to quadratic), it was harder to engage non-majors. Non-majors gamely experimented with isolated linear functions to create colorful grid patterns with offsets. However, when faced with the prospect of a progression to quadratics (by name), the attitude expressed by one kinesiology major "No more math, please!" This same group of students who initially objected to quadratics was enthusiastically engaged by a subsequent project examining the same concepts framed as a kinesiology problem. That same student explained concepts to others in the class unfamiliar with the technical aspects of the problem. We now actively map problems into contexts relevant to students attending the course and defer math terminology until it is useful for explaining intuitive understandings students already possess.

One-shot “Teasers” for Middle-Schoolers

A range of middle- and high-school students (including at least one Girl Scout troop) participating in community outreach programs sponsored by UTEP have attended our forty-five minute to two- hour short courses intended to motivate interest in programming through examination and extension of the algorithm presented in Table 1. It is interesting that, while students generally participate enthusiastically, their teachers will frequently decline to work on their own project. Those who have been coaxed to participate indicated surprise about their high level of enjoyment.

Lessons learned

We observe that all students who have attended Multimedia Exposed were already familiar and comfortable with the 2-d Cartesian coordinate system used to model the Raster object. However, it was interesting to observe that previous graphing experience in four quadrants did not assist pre-engineering majors in determining how to translate or scale parabolic functions "out of the corner." They realized that the lower left-hand corner marked the origin, but it took a second day to get them to think through the problem and make the connection between other terms in the function and resulting orientation of the parabola on the image. Once they did, parabolas of all colors, sizes, and orientations appeared on various colored backgrounds, so diverse, in fact, that one student named their resulting composite image "ParabolasGoneWild." From this experience, we realized that we needed to explicitly draw the Raster image within the larger Cartesian framework more often, and provide some intermediate practice on the effect of additional function terms on the resulting graph.

Future Work

We plan to integrate simulations that model and plot the dynamic behavior of physical systems such as ballistics and oscillating springs over time. Model code has been written that does not reference any trigonometric functions. Furthermore, the programs are remarkably short – typically the loops contain four or less simple arithmetic and assignment statements. Informal demonstrations to students with no experience with calculus indicate that these systems may be readily accessible to students attending Multimedia Exposed. It is interesting to see that upper-division computer science students also are fascinated by these simple programs that provide concrete examples of familiar theory. They indicate that these programming models are more concrete and intuitively understandable than the calculus-based physics course they attended years earlier. We are establishing collaboration with physics and math educators to investigate how explicitly programmed summation-based simulations might augment courses that introduce integration and its application.

Evaluation of the course is ongoing. In coordination with the Computing Alliance of Hispanic Serving Institutions, we are investigating the effect of Multimedia Exposed on academic success in subsequent STEM coursework and changes to student attitudes toward math and physics.

As described above, there are indications that students attending this media- and increasingly math-centric course are developing strong programming skills. Competence of students

completing this course will be compared with that of students completing the traditional first semester major's course.

We are considering the creation of a CartesianRaster class whose origin is centered in order to facilitate exploration of functions with negative values. One potential approach is to use this project to motivate and introduce object-oriented programming through the definition of this derived class.

Finally, we are adapting and refining this approach as a vehicle for teaching algorithms. One of the main ideas is to present algorithms as general methods that students can use and adapt to solve related problems of interest. A key step is the stripping out complex layers of background information that are not needed to understand the underlying principles. We find that by teaching algorithmic schemas to solve “purified” problems, students can follow the reasoning far more easily and can sometimes develop the computational idea themselves. Then the layered elaborations can be introduced step-by-step to extend the basic computational method to solve the more complex problems that are studied in traditional textbooks.⁶

Conclusions

We observe benefits to limiting the imposition of strongly object oriented constructs to those which simplify the range of projects that students build. Furthermore, there is evidence that simple student programming exercises that explicitly implement and evaluate mathematical systems through summation appear to be more intuitively understandable to many students than presentations that solely reference integration.

Acknowledgement

This report is based on work supported by the National Science Foundation through grants CNS-0540592 and DUE-0717877. Any opinions, findings, and conclusions or recommendations expressed in the paper are those of the authors and do not necessarily reflect the views of the NSF.

References

- ⁰Eric Freudenthal, Mary K. Roy, Alexandria Ogrey, Sherri Terrell, Olga Kosheleva, Pilar Gonzalez, and Ann Gates, Work in Progress - Initial Evaluation of an Introductory Course in Programming that Assists in Career Choices, Proc Frontiers in Education, 2008.
- ¹Design Process for a Non-Majors Computing Course, Proc.36th ACM Technical Symposium on Computer Science Education (SIGCSE), ACM, 2005.
- ²Mark Guzdial, Narrating Data Structures: The Role of Context in CS2, The Journal of Educational Resources in Computing (JERIC), ACM, 2008.
- ³David Hestenes, Malcolm Wells, and Gregg Swackhamer, Force Concept Inventory, The Physics Teacher, Vol. 30, March 1992, 141-158.
- ⁴Dan Kalman, Elementary Mathematical Models, Mathematical Association of America (Press), 1997.
- ⁵Heather Thiry, Lecia Barker, and Sarah Hug, *CAHSI Evaluation Progress Report*, The Computing Alliance for Hispanic Serving Institutions, 2009, <http://cahsi.cs.utep.edu/Portals/0/2008InterimEvaluationReport.pdf>.
- ⁶Alan Siegel and Eric Freudenthal, *Experiments in teaching an engaging and demystifying introduction to algorithms: Installment 1: Huffman Codes*, UTEP Computer Science Technical Report UTEP-CS-09-12