MPCT – Media Propelled Computational Thinking

Eric Freudenthal, Mary K. Roy, Alexandria Ogrey, and Tanja Magoc
University of Texas at El Paso
500 W. University Ave
El Paso, Texas, 79968
915-747-6954
{efreudenthal, mkroy, tmagoc}@utep.edu, anogrey@miners.utep.edu

Alan Siegel
New York University
251 Mercer Street
New York, NY, 10012
212-998-3122
siegel@cs.nyu.edu

ABSTRACT

Media-Propelled Computational Thinking (MPCT – pronounced *impact*) is a course designed to introduce programming in the context of engaging problems in media computation, math, and physics. Programming concepts are introduced as incremental steps needed to solve pragmatic problems students already understand. The problems, graphical API, and hands-on program features are intended to expose fundamental concepts in mathematics and quantitative science.

MPCT is offered in an entering students program for freshmen who plan to specialize in a variety of STEM (science, technology, engineering and math) and non-STEM subjects. The curriculum is intended to strengthen student intuition and interest in mathematical modeling and programming by engaging students in the direct manipulation of simple mathematical systems that model and display familiar physical phenomena. MPCT uses programs as concrete and manipulatable examples of fundamental concepts to engage a diverse range of students including women and underrepresented minorities.

Variants of MPCT are being developed for high schools, and as a means to introduce computational science to upper division undergraduates studying non-computational STEM disciplines. This paper provides an overview of MPCT and representative problem studies including models of ballistics and resonant systems. The evaluation plan is described and very preliminary results are presented.

Categories and Subject Descriptors

K.3.2 Computer and Information Science Education – *curriculum, computer science education*

General Terms

Algorithms, Design.

Keywords

First year programs, computational thinking, CS-Zero

SIGCSE'10, March 10–13, 2010, Milwaukee, Wisconsin, USA.

1. Introduction

At the University of Texas at El Paso (UTEP), many entering students must spend two-to-three semesters in remedial math courses prior to becoming calculus ready and able to enroll in their first computer science course (CS-1). The attrition rate in these remedial courses is high. Even after this preparation, only half of the students remaining in the program pass CS-1 on their first attempt.

In the Fall of 2007, MPCT was introduced in an effort to reduce attrition among students who lack the prerequisites for CS-1, and to maintain student interest in the subject. The first version was based on Mark Guzdial's Media Programming [1] because it had documented success in engaging students with weak math skills [2]. However, Media Programming is designed for students in Liberal Arts programs, and focuses on aesthetic -- rather than engineering -- creativity. Our course reviews were consistent with observations by Guzdial at other institutions [3]: students enrolled in pre-engineering and other pre-professional programs (e.g. nursing) enjoyed learning how programs worked, but were generally not engaged by the aesthetically focused projects that dominate the Media Programming curriculum [4].

Consequently, we were forced to rethink our course objectives. Following Guzdial's lead, we realized that programming should not be taught for its own sake. In response, we designed a curriculum where students construct simple graphical programs in Python that simulate familiar physical phenomena. Through these hands on projects and guided class discussions, students discover the mathematical principles that relate their program's iterative execution with the continuous processes that are being modeled.

MPCT now principally uses image manipulation to strengthen mathematical intuition at the pre-calculus level, and to illustrate the modeling of physical processes. Furthermore, we discovered that it is easier for students to focus on problem-solving and algorithm development using a minimal and mostly-imperative graphical library than Java's object-oriented AWT framework.

MPCT has been incorporated into selected sections of a required first-semester "University Studies" program designed to strengthen academic skills and to provide career guidance. These sections offer an accessible introduction to programming and pre-calculus via Python-based modeling. The content includes some closed-form calculations for basic concepts such as slopes, accelerations, summations and parabolas. MPCT supports the course's career guidance components by providing experiences representative of actual work in STEM disciplines.

Programming techniques in early courses should be chosen to minimize cognitive load while maximizing pedagogical value. The redirection of MPCT to introductory computation included a significant reevaluation of the programming interfaces used to support coursework. The original programming interface used the rich object oriented (OO) Java AWT toolbox. With this approach, even the design of extremely simple algorithms requires fairly complex access code before anything can be programmed. Consequently, the *conceptual* content embedded within our introductory programming lessons were often overwhelmed by the mainly *clerical* task of managing the access and manipulation abstractions for pixels in Java.

We developed the alternative shallow Raster class described in the upper row of Table 1. All pixel accesses explicitly specify row-column addresses. Thus, a pixel location is just a (row ,column) pair of integers, and a pixel color is just a red-green-blue triple. Algorithms that visit all pixels in a rectangular region can be programmed by a pair of nested loops. As reported in [5], students have little trouble understanding nested iteration in this context.

The course begins with by presenting simple code that draws dots and lines. Students adapt it to draw rectangles, triangles, trapezoids and parallelograms by the end of the second class. The lower row of Table 1 presents a program examined within the first two weeks of class that dramatically modifies a familiar cartoon image. The example was selected to illustrate low-overhead programming – *the code does not even require a function definition.*

In order to facilitate projects that plot mathematical functions and leverage students' incoming knowledge, `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.

In short, the simplifications in the necessary code allow the class to focus on the logic and the algorithms rather than object-oriented hierarchies that provide abstractions ill suited to the problem being addressed. Later projects require the plotting of functions with negative range that are (deliberately) inconvenient to be represented with Raster's origin, which is located in the image's lower-left corner. MPCT pragmatically utilizes this inconvenience to motivate the introduction of a `PosNegGraph` class that extends `Raster`. Both `Raster` and `PosNegGraph` are referenced by examples in this paper.

The next section summarizes the curriculum, including example problems in the newly developed modules on mechanical resonance and coupling, and planned extensions of MPCT's pedagogical approach to other courses. Finally, we report on our in-progress course evaluation and adaptations to the evaluation plan in response to MPCT's evolved focus.

2. Topic Sequence

Table 3 indicates the sequencing and approximate duration of each of MPCT's major educational themes. After familiarizing students with iteration through generation of geometric shapes, MPCT proceeds to introduce if-statements and Boolean expressions in the context of modifying images. Follow-up projects draw ensembles of lines and curves. For students intending to study a STEM discipline, the course also includes the construction of simple programs that model ballistic and resonant systems. Exercises mimic the familiar phenomenon of ball bounce and spring resonance, which are frequently poorly understood, even by students who have completed a semester of college physics [6].

Table 1.
Random-access Raster class

Public interface of `Raster`	Constructors • `Raster((numCols, numRows))` • `Raster((filename))` Accessors *Origin is in lower-left corner* *Parameters & return values are tuples* • `r, g, b = getRGB((col, row))` • `setRGB((col, row), (r, g, b))` Controls • `autoRepaint` *true: redraw on every access* • `ignoreOutOfRange` *true: silently ignore out-of-range accesses* Actions • `write(filename)` *save to file* • `repaint()` *redraw now*

Program presented in early session that dramatically recolors a familiar cartoon image.	`url = "http:….jpg"`
	`p = Raster(url)`
	`green = (0, 255, 0)`
	`cols,rows = p.widthHeight()`
	`for col in range (cols):`
	`  for row in range (rows):`
	`  r,g,b = p.getRGB((x,y))`
	`  if r<40 and g<40 and b<40:`
	`  p.setRGB((col,row), green)`

Table 2.
PosNegGraph

Public interface of PosNegGraph	Constructor • `PosNegGraph((numCols, range))` The y axis extends from +range to –range
Extends Raster	Public interface *Same as Raster* *only* `setRGB` *and* `getRGB` *are overloaded*

To ensure that students are not intimidated by unfamiliar mathematical abstractions, we prefer to first introduce evolving processes using the simplest-possible generators, and then to guide students into discovering algebraic simplifications they are fully prepared to understand.

The topics sequence through increasingly complex computational themes where all of the physical modeling is based on rates of change or summations (as opposed to integration). Table 4 presents a characteristic exercise associated with each theme.

For problems in mathematics and physics, we endeavor to minimize the level of outside knowledge required, and prefer show how processes evolve at an intuitive level and where incremental changes in the process state can be explained in physical and computational terms -- much like [7].

Math-centric programming projects generally begin with an exploration of the effects of rates-of-change. Afterwards, students are guided to reduce the already familiar summation problems (implicitly a recursive formulation) to closed form.

Theme A: Introduction to (graphical) programming. In a manner analogous to an immersive language course, students begin to 'converse' in Jython using only three statements to draw a multitude of geometric objects:

- `Raster(size)` – Used to "construct" a computer image:
- `setRGB(pos, color)` – Used to draw a dot:
- `for loops` – Used for iteration.

Table 3.
Course modules, themes, outcomes, and schedule

Theme & Wk	Major Course Module	Outcomes

A **1-2**	**First week / teaser**
	• **"Conversational" introduction to programming**
	  ○ **Login, start IDE**
	  ○ **Use looping to draw lines, boxes, triangles, parallelograms, etc**
		• Can login to computer
		• Knows what a program is
		• Can run a program
		• Familiarity with variables, expressions, looping, drawing w/ Raster class
		• Ability to draw rectangles and other simple geometric forms

B **2-3**	**If-statements, relational, and Boolean operators** • **Conditional re-coloring of images**	• RGB encoded as tuples • Familiarity w/ if-statements and Boolean expressions
C **4-8**	**Generating lines w/ summation and object extension** • **Horiz, vert, slope, y-intercept** • **Relate summation to closed form (y=mx+b)** • **Examine translation, negative ranges and PosNegGraph object**	• Intuition that slope is a "rate of change." • Functions and objects – to reduce complexity • Intuition re. multiplication • Greater proficiency at programming
D **9**	**Touch of OO: (classes)** • **Define & use PosNegGraph** • **Plot lines w/ neg range**	• Familiar w/ objects • Can plot fns w/ negative range
E **10-11**	**Generating curves (parabolas)** • **Using summation** • **Relate to closed form**	• Intuition that curves have 'slope that changes' • Knowledge: parabolas' slope changes linearly. • Familiarity w/ linear sums ○ Including geometric proofs

F **12**	**Ballistic motion (in vacuum)**
	• **Acceleration as change-of-rate**
	• **Gravity as constant acceleration**
	• **Program that simulates bounce**
	• **Relationship to parabola**
		• Intuition: physics models familiar phenomena
		• Familiarity:
		  ○ Velocity as rate
		  ○ Acceleration as rate of velocity change
		  ○ Gravity as constant accel
		  ○ Relationship to parabola

G **13-14**	**Resonance**
	• **Idealized (linear) spring**
	• **Construct and run simulation**
	  ○ **Plot position & velocity**
	  ○ **Coupled resonance**
		• Knowledge:
		  ○ Generated by linear acceleration
		  ○ Rate-of-change is sinusoidal
		  ○ Frequency independent of amplitude

Together, these commands allow students to:

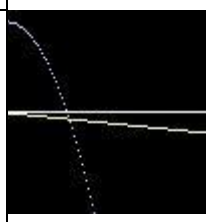
- Draw lines as a sequence of dots stacked in a row.
- Draw boxes as a sequence of lines stacked in a column
- Draw triangles, parallelograms, and trapezoids by deriving the inner loop's range from the outer loop's iteration variable.

Table 4.
Characteristic projects for each theme

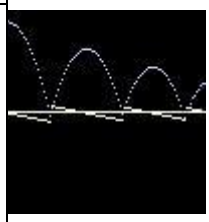
```
# Theme A – geometric shapes
nCols, nRows = 100,100
p = Raster((nCols, nRows))
for row in range(0, 30):
 for col in range (0, row):
  p.setRGB((row+20, col+20), white)
```

```
# Themes C-D
def drawLine(row=45, step=-1):
  nCols, range = 100,50
  i = PosNegRaster((nCols, range))
…
  row = 45         # y-intercept
  step = -2       # slope
  for col in range(numCols):
    i.setRGB((col, row), white)
    i.setRGB((col, slope), green)
    row += step
```

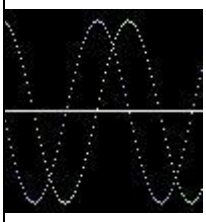
```
def parabola():          # Theme E
  …
  row=45; slope=0 # initial params
  rate = -.1       # acceleration
  for col in range(numCols):
    i.setRGB((col, row), white)
    i.setRGB((col, slope), green)
    row += slope
    slope += rate
```



```
def bounce():          # Theme  F
  …
  row=45; slope=0; rate = -0.2
  decay = .8       # bounce
  for col in range(numCols):
    i.setRGB((col, row), green)
    i.setRGB((col, slope), blue)
    if (row <= 0 and slope < 0):
      slope *= -decay # bounce!
    row += slope
    slope += rate
```



```
                          # Theme G
def harmonic(pos = 45, speed=0):
 i = PostNegGraph()
 springConst = 0.01
 interval = 0.1; mass = 0.01
 for x in range(i.width):
   i.setRGB((x, pos), green)
   i.setRGB((x, speed), blue)
   force = -pos * springConst
   accel = force / mass
   speed += accel * interval
   position += speed * interval
```



Initial programming exercises are sufficiently short to be conveniently typed directly to the interpreter. The first exercises contain no explicit arithmetic operations, which are gradually introduced to implement increasingly sophisticated computation. The difficulty of reliably typing more advanced programs motivates their storage within files.

The overall program is designed to foster experimentation and to encourage students to review fundamental concepts in algebra and geometry in a way where motivated understanding leads to successful program-generated displays.

Theme B: Control flow and Boolean expressions: The program presented at the bottom of Table 1, which is adapted from Guzdial's Media Programming course, introduces RGB

encoding, relational operators, Boolean expressions, and if-statements in the context of re-coloring images. Students are very engaged by this project and enjoy manipulating images taken by their camera or discovered on the web.

Theme C: Drawing lines. Once students are familiar with Python's syntax and semantics, follow-up projects examine various approaches to drawing lines. Early projects utilize only progressive generators such as drawLine. Even this project is intended to review (or revitalize) such basic mathematical concepts as slope and y-intercept in a hands-on visceral way that is difficult to match with pencil and paper or a graphing calculator.

Theme D Functions and Objects: Projects in this theme will plot multiple graphs with both positive and negative ranges. The equivalence of progressive and standard "closed form" equations of lines are also discussed in class and reinforced with short programming assignments that draw parallel lines and manipulate the dynamic range of grayscale images. User-defined functions and the PosNegGraph class are introduced as an approach to simplify the task of programming through modularization.

Theme E: Examination and generation of curves. The main study concerns curves whose slope changes linearly as shown in the parabola in Table 3. Projects in Theme E examine the effect of various initial conditions and rates of change including examples where the slope and rate have different and same signs. These are intended to provide students with intuitive understandings necessary to examine ballistic motion in Theme F. Finally, the relationship between parabolas and quadratic functions are made concrete through geometric proofs such as depicted in Figure 1.

Theme F: Ballistics. In ballistic problems, objects are accelerated only by gravity. Their trajectory is parabolic. The slope of their trajectory with respect to time corresponds to velocity. The slope of their velocity with respect to time corresponds to acceleration. The mapping of trajectories to parabolas is straightforward: slope corresponds to velocity, and the slope's constant rate of change maps to acceleration. Students first simulate a single "toss," and then are challenged to simulate a bounce as an inelastic collision with the ground, where at each bounce, velocity is reduced by 20%. This leads to an exponential decay in the maximum height achieved after each bounce. The `parabola` program's overlaid plot of position and velocity illustrates both continuous and discontinuous evolution of physical parameters.

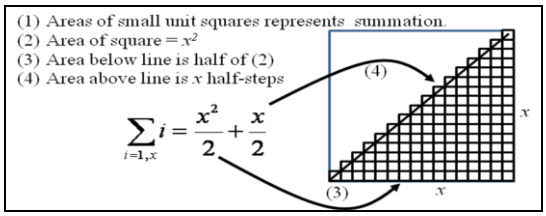


Figure 1.
Graphical depiction of linear sums in closed form as a quadratic function

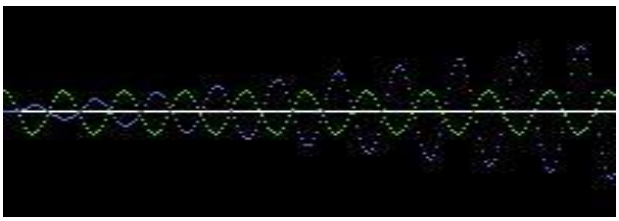


Figure 2
Simulation of Coupled Resonance

Theme G: Resonance. MPCT completes with an example of coupled resonance illustrating the principle underlying an opera singers' wine-glass shattering trick and the catastrophic failure of the Tacoma Narrows bridge (see Figure 3) in 1940 and the crashes of several Lockheed Electras around 1960.

Figure 2 illustrates the result of an advanced project that examines coupling when both the oscillator and resonator are tuned to the same frequency. Like the catastrophic failures enumerated above, this resonator quickly accumulates energy from the oscillator.

3. Extensions to other courses

A new introductory course in computational science intended for upper-division students of STEM disciplines that traditionally do not include computation such as biology, geology, finance, and math is planned for the Spring of 2010. This course will introduce programming using examples from MPCT, and then will proceed to implement simulations modeling more dynamic systems in which a sudden action could change the "normal" or expected behavior such as production-consumer markets, investment valuation, predator-prey models, and biological processes.

We are also adapting this approach of motivating math from concrete problems to the teaching of algorithms. There, the objective is to use specific problems as a vehicle for teaching algorithms as general methods that can be adapted to solve related problems of interest. 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. As in MPCT, the layered elaborations are introduced step-by-step to solve increasingly complex problems as described in [9].

Figure 3
The Tacoma Narrows Bridge [8]

4. Evaluation

More than half of the students entering UTEP intending to study CS are not "calculus ready," as required for CS1. First-attempt pass rates for CS-1 range from 50 to 70%, which surely contributes both to student time-to-graduation and attrition. Thus the need for intervention is clear. During the development of MPCT, several variants of Media Programming were offered at UTEP. In all versions, almost all students demonstrated proficiency at basic programming concepts and passed.

UTEP is a member of the Computing Alliance of Hispanic Institutions (CAHSI), which is evaluating various CS-zero courses offered at multiple institutions serving predominantly Hispanic populations of students. These evaluations include intermittent classroom observations and both pre- and post-course surveys examining preparation, social context, student engagement with the course, interest in further study of computation, and success in subsequent coursework.

In their 2008 report, the CAHSI evaluator measured high levels of both interest and confidence [10] among entering students prior to attending the course, though less than 20% had previously programmed. Post-course surveys of students attending the precursor courses indicate that 25% of the students were not motivated to continue studies related to computer science. This is likely a positive result if the course helped students correct unrealistic expectations about computer science early in their academic careers.

Students who attended a variety of CS-0s (including Media Programming at UTEP and Alice at other institutions) had similar passing rates that were similar to the general population. The mathematically-oriented revisions to MPCT have required several semesters of development and have only recently been implemented – hence no longitudinal data is available yet.

During the Spring 2009 term, the evaluation was broadened to include instruments that examine changes in interest, self-efficacy and competence related to mathematics. Several open-ended essay questions were included in order to guide the selection of relevant questions for the intended Fall evaluation.

Our longitudinal evaluation will be broadened to also compare the academic success of students in subsequent math courses with the academic success of students who do not attend MPCT.

Entering STEM students attended MPCT during the Fall semester of 2008, and non-STEM students attended distinct sections during both the Fall and Spring semesters. The mathematics included in the section was first offered Spring 2009, and therefore was only attended by non-STEM students. A post-course survey of the Spring 2009 non-STEM cohort examined changes in (1) perceptions of knowledge and understandings of key concepts, (2) perceptions of mathematics' relevance to 'real life' scenarios, and (3) attitudes toward learning math concepts in the context of programming.

As described in [11], the preliminary findings from the non-STEM section are encouraging. Although the sample size was too small to draw reliable conclusions, they reflect the instructors' observations of student motivation and engagement. Survey results indicate shifts from low level to higher levels of understanding math concepts introduced in MPCT, positive attitude toward MPCT structure and its intended objectives, and highly favorable perceptions of MPCT's relevance to real-life applications. We expect statistically reliable results from the Fall 2009 cohort, which is much larger, containing more than sixty pre-STEM and twenty non-STEM students.

5. Conclusion

Continuing evaluation of introductory programming offerings at UTEP targeting pre-STEM students have motivated evolutions in curriculum, course objectives, and evaluation strategies. Interestingly, the resulting course, MPCT, which engages students in a "computational reasoning," integrates both programming and mathematics, is engaging to both pre-STEM and non-STEM students with weak math skills. Results from early evaluation efforts are encouraging and have lead to refinements in evaluation strategy that mirror the course's evolution.

6. Acknowledgement

This report is based on work supported by the National Science Foundation through grants CNS-0540592, IIS-0829683, 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.

7. References

[1] Guzdial, *Computing and Programming with Python, a Multimedia* Approach, Prentice Hall, 2006.

[2] Guzdial, *Design Process for a Non-Majors Computing* Course, Proc.36th ACM Technical Symposium on Computer Science Education (SIGCSE), ACM, 2005.

[3] Guzdial, *Narrating Data Structures: The Role of Context in CS2*, The Journal of Educational Resources in Computing (JERIC), ACM, 2008.

[4] Freudenthal,, Roy, Ogrey, Terrell, Kosheleva, Gonzalez, and Gates, *Work in progress – Initial evaluation of an introductory course in programming that assists in career choices*, Proc. FIE, 2008.

[5] Freudenthal, Roy, Ogrey, Gates, *A Creatively Engaging Introductory Course in Computer Science that Gently Motivates Exploration of Mathematical Concepts*, Proc. ASEE, 2009

[6] Hestenes, Wells, and Swackhamer, *Force Concept Inventory*, The Physics Teacher, Vol. 30, March 1992, pp 141-158.

[7] Kalman, *Elementary Mathematical Models*, Mathematical Association of America (Press), 1997.

[8] Prelinger archive photo released to the public domain http://www.archive.org/details/prelinger

[9] Siegel and Freudenthal, *Experiments in teaching an engaging and* demystifying introduction to algorithms: Installment 1: Huffman Codes, UTEP Computer Science Technical Report UTEP-CS-09-12, April 2009.

[10] Thiry, Barker, and Hug, CAHSI Evaluation Progress Report, The Computing Alliance for Hispanic Serving Institutions, 2009, http://cahsi.cs.utep.edu/Portals/0/2008InterimEvaluationReport.pdf

[11] Suskavcevic, Kosheleva, Gates, and Freudenthal, Preliminary Assessment of Attitudes towards Mathematics for a Non-STEM Section of Computational Computer Science Zero, UTEP CS Technical Report UTEP-CS-09-13, May 2009.