AC 2010-1108: A COMPUTATIONAL INTRODUCTION TO STEM STUDIES

Eric Freudenthal, University of Texas, El Paso

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

Rebeca Gonzalez, Chapin High School

Rebeca Gonzalez is a mechanical engineer working as a teacher of computer science, pre-engineering, and math at Chapin High School in El Paso, Texas.

Sarah Hug, University of Colorado

Sarah Hug is an assessment and technology consultant. Dr. Hug also serves as the Graduate Admissions Coordinator for the Alliance for Technology, Learning, and Society at the University of Colorado in Boulder and a researcher for the National Center for Women and Information Technology. She also teaches in the Department of Educational Psychology at the University of Colorado-Boulder.

Alexandria Ogrey, University of Texas, El Paso

Alexandria Ogrey is an undergraduate studying computer science at the University of Texas at El Paso.

Mary Kay Roy, University of Texas, El Paso

Mary Kay Roy is an instructor of computer science at the University of Texas at El Paso

Alan Siegel, NYU

Alan Siegel is on computer science faculty at NYU's Courant Institute of Mathematical Sciences.
A Computational Introduction to STEM Studies

Abstract

We report on the content and early evaluation of a new introductory programming course “Media Propelled Computational Thinking,” (abbreviated MPCT and pronounced iMPaCT). MPCT is integrated into a freshman-level entering students program that aims at retaining students by responding to the academic recruitment and attrition challenges of computer science and other STEM disciplines. This course is intended to provide meaningful experiences of relevance to students choosing majors that also fortifies their qualitative understandings of foundational math and physics concepts.

MPCT’s activities are designed to provide analytical challenges typical of STEM professions and to motivate additional inquiry. Preliminary evaluation results are encouraging – students from a wide range of academic majors find MPCT engaging and report that the analytical tasks were effective at conveying insight and decreasing anxiety towards foundational mathematical concepts.

This paper extends prior reports on MPCT with evaluation results indicating that more than half of attendees indicated increased confidence in the understanding and application of quantitative analysis tasks and detected differences in that nature of students’ engagement with math in MPCT and traditional math courses. In addition, this report includes an overview of an emerging effort to investigate the integration of MPCT into secondary school curricula.

Introduction

MPCT is integrated into an Entering Students Program (ESP) [4] attended by entering freshmen at the University of Texas at El Paso, a primarily Hispanic-Serving Institution (HSI) serving an economically disadvantaged bi-national urban area on the US-Mexico border. The objective of the entering students program is to assist students in developing skills necessary for academic success in college and to assist in career selection. MPCT, which is allocated approximately half of the course’s instructional time, provides technical content to complement the entering students program’s curriculum that reviews study, note-taking, presentation, and writing skills, and career guidance.

Introductory courses for technical disciplines such as computer science can offer a limited curriculum representing a discipline’s activities. Activities and projects of introductory computing curriculum designed to attract students generally focus on the dramatic outcomes of tasks whose programming challenges are frequently more clerical than analytical. Consider the voluminous specification required to generate the detailed choreography of a robotic dance (without regards for physics). While the graphical outcomes of these projects are impressive, and the specification of these moves may provide may provide useful practice of coding skills, we have concern that the technical tasks have little similarity with analytically intense academic coursework typical of computer science and other STEM disciplines. We have encountered students who enjoy analytical work and enjoyed a high-school program based on Alice who ultimately chose to study non-computational disciplines because they found the detail-work associated with the dramatic projects mundane.

MPCT’s design reflects an understanding that effective career selection includes a matching both of practitioners’ natural inclinations (proclivities) and aptitudes with the principal activities required for a profession. MPCT’s activities are designed to provide analytical challenges typical of STEM professions and to motivate additional inquiry.

MPCT’s design is substantially motivated by observations that students’ perceptions of and deficiencies in mathematics both contribute substantially to avoidance of and attrition from study of computer science and other STEM disciplines. MPCT exploits programmed systems’ facility at manipulating computation to
provide students with a review of foundational mathematical concepts necessary for success in a variety of STEM disciplines that was engaging to most attendees, regardless of gender, ethnicity, or intended major (STEM and non-STEM). In order to permit students to focus on analytical tasks and algorithm design, typical programming projects in MPCT are surprisingly short – most are four to ten lines of Python code (though a few extend to forty) that require the development and implementation of algorithms that use numerical iteration, if-statements, function, and class definition. In order to provide visceral understandings of program behavior, output is generally graphical, involving the direct manipulation of pixels within an RGB image using explicit Cartesian coordinates.

MPCT’s exploitation of programming in teaching mathematical concepts differs from the increasingly common use of MATLAB in first year programs and engineering math courses [12] in two critical ways. First, MPCT includes a sustained focus on developing qualitative understanding of lower-level foundational mathematical topics including an extensive review of simple linear systems of one variable. Furthermore, entering students have little experience with engineered design. This motivated MPCT’s sustained focus on (2) the low-level processes of rendering graphical objects within raster images.

This paper presents MPCT’s progression of concepts and projects. Programming fundamentals are introduced and exercised within lab assignments during the first couple of weeks of class in the context of graphical manipulation such as the use of nested for-range statements to enumerate the coordinates of pixels within geometric objects. Subsequent labs first examine the plotting of simple mathematical functions, which are later extended to explore the simulation and mathematical simulation of familiar physical phenomena such as ballistics and resonance.

We also describe ongoing efforts to evaluate MPCT. The preliminary results indicate increased confidence in ability to understand and apply foundational quantitative analysis skills and interest in pursuing STEM studies. Our research plans include a longitudinal study of academic progress that examines the success of students who pursue STEM studies after attending MPCT.

Pedagogical Approach

Our orientation is that programming techniques in early courses should be chosen to minimize cognitive load while maximizing pedagogical value. The focusing 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 exposed by the programming framework of [1]. With this approach, even the design of extremely simple algorithms requires fairly intricate code before anything can be programmed. Consequently, the conceptual content embedded within our introductory programming lessons were often overwhelmed by the complexity of code required to manage the access and manipulation abstractions for pixels in Java.

This motivated our development of an alternative shallow Raster class described in Table 1 in which pixels are explicitly referenced by their coordinates. Each access of a pixel requires explicit specification of its row and column address. Thus, a pixel location is just a (column, row) pair of integers represented using a native Python tuple (vector) type, and a pixel color is just a red-green-blue tuple. 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 by presenting simple code that draws dots and lines. Students adapt them to draw rectangles, triangles, trapezoids and parallelograms by the end of the second class. Figure 1 presents a program sometimes examined within the first two weeks of class that dramatically modifies a familiar cartoon image. This example was selected to illustrate low-overhead programming – the code does not even require a function definition.

<table>
<thead>
<tr>
<th>Constructors</th>
<th>Controls</th>
</tr>
</thead>
</table>

Table 1. Public interface to Raster class
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.

```python
url = "http:...jpg"
p = Raster(url)
green = (0, 255, 0)
cols,rows = p.getWidthHeight()
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)
```

Figure 1. Program to recolor dark pixels

In short, the simplifications in the necessary code allow the class time 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 represent 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.

**Topic Sequence**

Figure 2 illustrates MPCT’s current flowchart of course modules and prerequisite dependencies. A strong–inter-module dependency is represented by a solid line; and a weaker dependency (indicated by a dotted line) indicates an alternate path that permits a course to move more quickly at the cost of removing a particularly engaging presentation of a key concept.

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.

For example, A.2 provides a preferred first exposure to nested looping that provides opportunity for students to play with both column- and row- major filling of simple polygons in a dramatic and motivated context. Students apply their newly gained knowledge of nested iteration to the filling of a rectangular region
Programming Basics

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.

As described below, 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.

**Module A.1: Create raster, random access pixels, iteration to draw horizontal lines:** This introductory lesson exploits Python’s interactive mode to permit students to issue single commands to create and manipulate a raster image (represented as a Raster object) and “draw” upon through the column-row selection of individual pixels that are conveniently represented using Python’s native tuple (vector) type. The

---

Figure 2. Linkages among course modules

---
clerical challenge of reliably typing sequences of commands first motivates the storage of a sequence of commands within a file in a manner can be easily repaired or reused using copy/paste. Finally, iteration is presented as a convenient and intuitive solution to the challenge of repeating a single command multiple times while varying a single parameter in order to draw horizontal or vertical lines as is illustrated in the top row of Figure 3.

![Figure 3. Exploration of iteration](image)

**Module A.2: Examine nested iteration to draw filled rectangles, and other simple polygons:** As illustrated in the lower two rows of Figure 3, A.2 exploits students’ familiarity with direct pixel manipulation and iteration presented in A.1 to provide students with an engaging opportunity to playfully tinker with tiny programs that generate filled geometric forms. A.2’s engaging sequence of exercises first introduce students to the nesting of loops as an intuitive and simple approach to the clerical challenge of enumerating the location of every pixel within rectangular region and then provide an opportunity to develop intuition about inter-loop interactions.

For example, the first project traverses every pixel within a rectangular region in column-major order, and its immediate successor challenges students to fill the same region in row-major order. Students’ conceptual errors are displayed in easily understood graphical results that are amenable to analysis and repair. A.2’s scope can be modulated by available time and desired learning outcome. Optional lessons examine interactions between inner and outer iteration bounds in order to draw non-rectilinear forms such as triangles, parallelograms, and trapezoids and manipulate iteration stride to enable the drawing of striped patterns. Should the learning priorities include only a superficial understanding of iteration, A.2 can be omitted entirely and the topic can instead be briefly presented in the context of A.3.

**Module A.3: If-statements, relational, and Boolean operators to selectively recolor raster images:** As illustrated above in Figure 1, this lesson uses nested iteration and exploits conditional expressions including relational, Boolean, and grouping operators to conditionally change the color of pixels within a rectangular image. Earlier modules used intuitively named pre-defined constants (e.g. “green”) to represent colors. A.3 exposes the graphical system’s underlying representation of color as red-green-blue tuples. A raster image is initialized from a jpeg explicitly obtained from “the web” using HTTP.

Each pixel location within the rectangular (raster) image is systematically visited using nested iteration. Each pixel’s red-green-blue vector is obtained and unpacked into its components using the get method. An if-statement that contains relational and Boolean operators controls whether the pixel’s color-vector is overwritten. Projects challenge students to both experiment with the creation of conditional expressions that select alternate colors and the specification of vectors representing desired alternate colors. Finally, students are exposed to the complexity of automating the detection of forms easily recognized by the human vision system. Optional projects examine simple image processing techniques such as blurring that provide insight into convolutional operations.

**Linear systems**
Module B.1: Use of summation to draw sloped lines and the computation of slope: This module examines the nature of linear systems. As illustrated by the top two images and programs from Figure 4, initial projects extend the concept of iteration introduced to draw horizontal lines in A.1 to instead draw sloped lines using summation. A row variable is initialized at column zero. A constant “step size” value is repeatedly summed into the row variable increasing its value at a linear rate that is graphically depicted. Students characterize the effect of various step sizes and initial row values. Later exercises lead students to derive the meaning of their generalization as slope and y-intercept.

As illustrated by the image and program at the bottom of Figure 4, students are subsequently challenged to draw lines that connect designated points (to draw a geometric shape). To do this, students must derive the step size from the desired change in row and column using division. We observe that most attending the class can parrot “y=mx+b” as an equation for a line, but nonetheless initially have trouble computing step size; even math-phobic students enrolled in non-STEM programs and are visibly delighted when they derive an approach to determining step size (slope) using division.

Module B.2: Linear transformations: Drawing lines with multiplication and adjusting image grayscale. This module challenges students to generalize their understanding of line generation to the linear projection from an input variable to a dependent output variable. Early projects examine the concept and utility of computing a y-intercept in order to easily compute the row corresponding to any column.

A particularly motivational project from this lesson examines the grayscale correction of pictures with limited dynamic range. Following a discovery-learning model, students are initially challenged to “improve” the picture’s contrast without guidance. Typically students will suggest applying either linear scaling or constant offsets.

```python
pic = Raster((80,80))
row = 15
for col in range(0,pic.width):
    pic.setRGB((col, row), green)

row, step = 15, 0.5
for col in range(0,pic.width):
    pic.setRGB((col, row), green)
    row = row + 1

first, last = (10, 10), (70, 50)
run = last[0]-first[0]
rise = last[1]-first[1]
step, row = float(rise)/run, first[1]
for col in range(first[0], last[0]):
    pic.setRGB((col, row), green)
    row += step
```

Figure 4. Exploration of line drawing
Figure 5, (extracted from teaching resources) illustrates a composite approach that applies a linear transform that projects the input image’s limited grayscale to a full 0…255 range. Related exercises include inverting a negative image thus permitting students to explore the computation of inverse relationships.

Managing complexity using language features

Module C.1: Defining functions to facilitate drawing multiple lines: Functions are introduced as an approach to reduce complexity through the creation of reusable code fragments. As illustrated in Figure 6, students are challenged to write programs that draw a path using line segments that connected designated start and end points and avoid colored obstacles. Rather than focusing on the path problem, this module provides opportunities to first practice the application of computing slope. Observe that, the body of drawline appearing in Figure 6 is identical to the program at the bottom of Figure 4. In this manner, reduction of the program complexity is achieved by drawing multiple lines with multiple calls to a single function rather than the insertion of repeated code fragments.
Module C.2: A user-defined class that facilitates plotting of functions with negative range: The linear systems examined previously only have non-negative range. In this module, students are encouraged to examine negative range by translating the image by the adding of a constant to the row value. The repetitive task of translating the image to examine the negative range motivates the examination of a user-defined class that centers and draws the X-axis. As illustrated by Figure 7, the easily understood PosNegGraph class conveniently translates the origin to the middle of the left edge.

![Figure 7. Illustration of PosNegGraph plot](image)

Non-linear systems

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).

Lab exercises examining non-linear systems extend the summation techniques previously examined to simulate the dynamics of familiar physical phenomena. 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].

Finite differences are computed at each time step and are summed into system state. Longitudinal studies are being conducted to determine if cohorts of students who are first exposed to these concepts through simulation have improved academic success in subsequent courses that examine the same phenomena using calculus.

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.

D.1: Examining curves as lines whose slope varies to simulate simple ballistics: Boolean expressions and conditional statements (taught in A.3) should be taught if the simulation of a ball bounce is included in the lesson. The instructor leads a discussion about the similarities and differences between lines and curves and elicits an observation that a curve’s slope must vary. Students add the rate variable, and begin to experiment with drawing curves.

The main study concerns curves whose slope changes linearly as shown in the parabola in the bottom Figure 8. Once sufficient time has been spent experimenting with drawing curves, students begin to apply what they have learned to simulating familiar phenomena such as ballistics and ball bounce.

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.

As illustrated in Figure 9, students are able to easily examine the effects of varying initial conditions including examples where the slope and rate have different and same initial signs. Finally, the relationship between parabolas and quadratic functions are made concrete through geometric proofs such as depicted in Figure 10.

D.2: Examination of systems whose dynamism depends on state such as exponential decay and resonance. It is interesting that upperclassmen who have attended multiple courses in Calculus are enthralled by the progressive computation of exponential decay illustrated in Figure 11 that could easily be understood by a freshman if presented in an accessible context. We observe that the spring resonator of Figure 12 is accessible to students attending a first course in mechanics. The electrical resonator of Figure 12 is incorporated in an introductory programming course offered to students in electrical engineering in the Spring of 2010 that incorporates the pedagogy of MPCT.
MPCT completes with three more complex examples of physical phenomena: an example of coupled resonance illustrating the principle underlying an opera singers’ wine-glass shattering trick, the catastrophic failure of the Tacoma Narrows bridge (see Figure 3) in 1940 and the crashes of several Lockheed Electras around 1960.

Figure 11. Exponential decay

Figure 12. Simple mechanical resonator

Figure 13. Electrical resonator

Figure 14. The Tacoma Narrows Bridge

Figure 15 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.
Extensions to other courses

A new introductory course titled “Computation for Science and Engineering” 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 Spring 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.

Two other pilots are beginning in Spring 2010. A section of an introductory programming course offered to students enrolled in Electrical Engineering will be taught using MPCT’s pedagogical approach and will include projects that simulate dynamism in electrical systems. A section of a statistics course attended by students of psychology will also include elements from MPCT with the expectation that the process of constructing simulators of stochastic systems will assist in students’ understanding of coupled and independent random 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].

Extensions to K-12

MPCT’s lessons provide concrete and understandable methodology for exposing foundational principles of relevance to STEM related subjects taught in K-12 in a manner that appears to facilitate and promote the development of qualitative understandings, thus facilitating reflection and higher order thinking. This observation has motivated our exploration of the potential integration of MPCT into high- (and maybe middle-) school curricula.

Students from high- and middle-schools have attended a variety of half-day and one-hour enrichment programs hosted at UTEP introducing them to early lessons in MPCT. These sessions for example, exposed the participating students to programming through the generation of lines as sequences of equally-sized steps and parabolas as sequences of linearly changing step sizes.

These short lessons effectively engaged most attendees in qualitative reasoning about the gross characteristics and input-output relationships of linear and quadratic systems such as the effect of varying the value of y-intercept, and rates of change. Most attendees quickly developed sufficient understandings of simple programs while developing other arithmetic and problem solving skills that allowed them to modify, or create their own programs in order to solve different, more complex (but related) problems. It is telling that parents (who participate with their sixth-graders at parent-child events) are frequently astounded by the simplicity and beauty of principles they remembered as opaque and confusing.

These same principles are the foundational concepts of quantitative analysis underlying a variety of K-12 subjects including all algebra levels, geometry, biology, chemistry, physics, and a variety of career pathway
courses offered through technology education. High school math, science, technology-education and pre-engineering teachers participating in the development of MPCT are investigating the effectiveness of inserting modules from MPCT into their curricular content directed to anchor students’ typically weak understandings of concepts in the context of kinematics, ballistics, projectile motion, capacitive discharge, exponential population growth, and even precursors to integration introduced as “area beneath a curve” – which directly corresponds to pixel counts.

**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 reduction is likely a positive result if we assume the course helped students’ correct misconceptions about computer science as a discipline or unrealistic expectations about their skill level with computing early in their academic careers 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.

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.

During the Spring 2009 term, the evaluation was broadened to include instruments that examine changes in interest, self-efficacy and competence related to mathematics. Approximately 60 students completed pre- and post-course questionnaires assessing attitudes towards math and computer science. These questionnaires included multiple-choice and open-ended essay questions in order both facilitate quantitative analysis and to guide the selection of relevant questions for the intended Fall 2010 evaluation.

Pre-survey results indicate that in addition to high confidence in programming skills, students have high confidence in their math skills that are inconsistent with their performance in class.

Our longitudinal evaluation will also compare the academic success of students in subsequent math courses with the academic success of students who do not attend MPCT.
Preliminary Evaluation of Math-Centric Course

The math-centric curriculum was first taught in Fall 2009. That term, approximately ninety students were enrolled in four sections of MPCT. The evaluation included pre-, mid- and post-course surveys, interview sessions that qualitatively probed student perceptions of computer science, and “thinkaloud” exercises in which the students were observed constructing simulators of physical phenomena. The following is an initial summary of results from the surveys.

The study quantitatively examined the following questions:

- Do students report a change in mathematics skills following the course?
- Do students report a change in their interests regarding computation following the course?
- Do students report a change in attitude regarding mathematics following this course?
- Do students report a change in value of computer science as a discipline?
- Do students who major in computer science perceive the major as supportive following their experience in MPCT?

In addition, the study investigated the following qualitative questions:

1. How do students perceive mathematics course activities?
2. How do students perceive MPCT course activities?

![Figure 17. Student change in interest regarding computing activities.](image)

The study also sought to understand if the changes listed above change (or fail to change) in differing patterns for certain students. For example, do benefits of MPCT differ by mathematical experience? By ethnicity? By gender? By course type? These differences were tested when applicable and feasible. Differences are reported where they exist at a 95% confidence level, meaning when the data was sufficiently
different to indicate statistical significance with 95% surety. Note most differences were not statistically different in terms of gender, race, or course type.

MPCT students believe that they gain mathematics skills through programming

According to post-survey data from 57 MPCT students (see Error! Reference source not found.), 67%-84% of students reported an increase in their ability to solve mathematics and computer programming-related problems that coincide with course objectives. In each case, there was a portion of students who felt they were completely capable of performing these tasks before the course, though these portions of students are relatively small (5%-31%). A few students report they are unable to accomplish the mathematical and programming related tasks described, and the numbers of students with this response increases as listed tasks become more complex. No differences were detected by course type, math course level, or gender, though those students enrolled in calculus did tend to indicate they already knew how to complete many of the tasks described before enrolling in MPCT.

MPCT students view increased interest in computing activities

Students enrolled in this introductory programming course were asked to describe their reported interest in engaging in computing-related activities following the MPCT course. Seventy four students answered 5 items related to computer interest following their course (e.g. solving technical problems; learning new computer applications). As depicted in Error! Reference source not found., an interest score was developed for each student, averaging all of the ratings on each of the 10-point items. As per MPCT faculty request, these scores were then grouped into the following: 1= scores of 1.0-3.75 (strong decrease in interest) 2=3.76-5.5 (slight decrease in interest) 3=5.51-7.75 (slight increase in interest) 4=7.76-10 (strong increase in interest). The distribution of students falling within each of the interest categories is depicted below. Note that over 67% of the students reported an increase in interest regarding computing activities following the MPCT course. No minority of students indicated a decrease in interest regarding computing activities following MPCT. No differences were statistically significant at this time.

![Please indicate your level of agreement with the following statements.](image)

Figure 18. Post-survey responded regarding math confidence.
Development of mathematics confidence over the course of semester

Our data allows for two ways to measure change in confidence over the course of the semester. One approach is to compare the number and percent of students with math confidence during the pre-survey and during the post-survey. Another indication is students’ reported change in confidence at the end of the semester. The first chart shows the responses of 73 students on one of two post surveys given at the end of the semester. As indicated in Error! Reference source not found., while 82% say they are confident in their math ability, 48% also describe themselves as anxious when asked to complete math problems. The percent of students who indicate they are confident in their math ability rose from 73% in the pre-survey to 82% in the post survey. However, the number of students who are anxious when solving mathematics problems also rose from 42% to 48%. This increase in anxiety may be due to experiences in college mathematics courses (likely pre-calculus) attended concurrently with MPCT – which typically have pass rates below 50%.

Another way we can measure change in mathematics confidence is through items that ask students to reflect upon their experiences in a course and describe any changes they discern in their own attitudes towards mathematics. Figure 19 presents data regarding 57 students’ self-reported CHANGE in interest and confidence due to the MPCT course. Note that 72% of students indicate they are more confident in their mathematic ability because of their experience of MPCT, and two-thirds of students are more confident they could succeed in a course that dealt with quantitative reasoning following MPCT.

![Figure 19. Self-reported change in confidence.](image)

Students view computer science as valuable discipline
A goal of the introductory computing courses throughout CAHSI is to improve students’ perceptions of computing as a discipline. All students were asked to describe their perceptions of computing following the MPCT course, and were asked to describe any change in impression experienced after the course (e.g., “In the computing field I could do work that is exciting to me”, “I could earn an attractive salary”). Just as in the case of computing activity interest, the student averages were grouped, from a ten point scale to a grouping of 4 levels, from negative changes in computing impressions to positive changes in computing impressions. As indicated in Figure 20, nearly all students (60 of 71) experienced a positive change, indicating that they were more likely to value the computing discipline as a career field following their participation in MPCT. There was a significant difference by math course level, with precalculus and algebra students indicating a greater increase in their value of computing than calculus students and “non-science” math course takers.

![Impressions of Computing](image)

**Figure 20. Change in student impression of computing.**

**CS Majors’ views of computing discipline as a positive community**

Computer Science majors answered items regarding their impressions of computer science as a positive learning community following their enrollment in MPCT, using a scale of items such as “In this discipline, you could get helpful assistance from a tutor, if you felt you needed such help”. As indicated in Figure 21, most students (25 of 32) experienced an increase in positive regard for their discipline following their MPCT course.
Figure 21. C.S. Majors' impression of discipline as supportive community.

Figure 22. Student characterization of course and related disciplines.

Please characterize the following concepts and tasks as related to "computer science class" "mathematics class" and/or "this course": You can select more than one answer for each item.

When solving problems with mathematics:

- I am creative and discover things on my own
- I memorize mathematical equations
- I mostly use facts and procedures I have learned
- I memorize syntax
- The purpose of problem solving in this class is...
- My primary goal is to compute a correct answer
- I reason about relationships
- The purpose of problem solving in this class is...
- I reason about system behavior

Figure 22. Student characterization of course and related disciplines.
What is math class like? What is MPCT like?

Surveys allowed for students to indicate which characteristics they associated with computer science classes, mathematics class, and MPCT class. Students were told they could check as many options as they would like for each discipline, and could also check multiple disciplines for each characteristic. While 57 students took this survey, sums will not add to 57. See Figure 22.

This graph indicates that nearly half of students view MPCT as promoting these “top 5” attitudes, behaviors, and concepts:

- I am creative and discover things on my own (29 of 59)
- I reason about relationships (27 of 59)
- I reason about system behavior (26 of 59)
- The purpose of problem solving in this class is to understand concepts presented (26 of 59)
- I memorize syntax (25 of 59)

At the same time, students were more likely to describe mathematics class in the following ways:

- I memorize mathematical equations (50 of 59)
- My primary goal is to compute a correct answer (45 of 59)
- The purpose of problem solving in this class is to understand concepts presented (43 of 59)
- I mostly use facts and procedures I have learned (40 of 59)
- The purpose of problem solving in this class is to get the right answers (39 of 59)

<table>
<thead>
<tr>
<th>“Describe math class activities” codes</th>
<th>Number of occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Response</td>
<td>39</td>
</tr>
<tr>
<td>Math is presentation and practice (both of above)</td>
<td>9</td>
</tr>
<tr>
<td>Positive, general statements (math class is fun, interesting, not difficult)</td>
<td>9</td>
</tr>
<tr>
<td>Math is memorization/learning and execution of equations (in 2 cases viewed as learning, in others as memorization of equations)</td>
<td>7</td>
</tr>
<tr>
<td>Math is difficult</td>
<td>5</td>
</tr>
<tr>
<td>Not Applicable</td>
<td>5</td>
</tr>
<tr>
<td>Math is presentation (indicates math class is lecture based, concepts are “covered”)</td>
<td>4</td>
</tr>
<tr>
<td>Math is problem solving</td>
<td>3</td>
</tr>
<tr>
<td>Math is practice (indicates math class involves continuous practice of strategies)</td>
<td>1</td>
</tr>
<tr>
<td>Neutral</td>
<td>1</td>
</tr>
<tr>
<td>Teacher is expert (in math class, the teacher TA or instructor holds the knowledge and offers it through presentation and examples)</td>
<td>2</td>
</tr>
</tbody>
</table>
Following these forced choice survey items, students were asked to describe the activities in which they participated in math class and in MPCT class. Responses were coded using a grounded theory approach, in which the themes emerged from the data with no preconceived coding system in place. As indicated in Table 2 and Figure 23, the responses uncovered a theme regarding mathematics as a course that focuses on “presentation and practice”, in which students received lectures and then mimicked the behaviors of the more knowledgeable teacher.

The teacher presents the topic and gives solves problems related to the topic. We ask any questions to fully understand the topic, then gives us a homework assignment which challenge us creatively and mathematically.

Figure 23. Typical quote indicating "Math is presentation and practice."

As illustrated in Table 3 and Figure 24, a few less pervasive though related themes emerged from MPCT data as well, in particular the creation/creative aspects of MPCT, the exploration involved in MPCT, and the ways in which problem solving is a main theme of the course activity. The less descriptive “programming” code was used when a response was vague as to what programs were for, or what programs did (i.e. “we programmed the computer with JES”).

**Conclusion**

Continuing evaluation of introductory programming offerings at UTEP targeting pre-STEM students have motivated evolutions in curriculum, course objectives, and evaluation strategies. The resulting course, MPCT, engages students in a “computational reasoning,” integrates both programming and mathematics, is similarly engaging to Hispanic and non-Hispanic, pre-STEM and non-STEM students of both genders with weak math skills in a culturally and gender-neutral manner. Results from early evaluation efforts are encouraging and have lead to refinements in evaluation strategy that mirror the course’s evolution.

<table>
<thead>
<tr>
<th>“Describe MPCT class activities” codes</th>
<th>Number of occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>No response</td>
<td>37</td>
</tr>
<tr>
<td>Positive (interesting, fun, awesome)</td>
<td>11</td>
</tr>
<tr>
<td>MPCT is programming</td>
<td>11</td>
</tr>
<tr>
<td>MPCT is exploration</td>
<td>6</td>
</tr>
<tr>
<td>MPCT is problem solving</td>
<td>5</td>
</tr>
<tr>
<td>MPCT is creative/involves creating</td>
<td>3</td>
</tr>
<tr>
<td>Negative (slow, boring)</td>
<td>3</td>
</tr>
<tr>
<td>Not applicable</td>
<td>2</td>
</tr>
<tr>
<td>MPCT allows for visualization</td>
<td>2</td>
</tr>
<tr>
<td>MPCT involves mapping relationships</td>
<td>2</td>
</tr>
<tr>
<td>MPCT is complicated/challenging</td>
<td>2</td>
</tr>
<tr>
<td>Ccso is memorization</td>
<td>1</td>
</tr>
<tr>
<td>MPCT is collaborative</td>
<td>1</td>
</tr>
</tbody>
</table>

Simple tasks like make a dot, then a line, and then a curved line are asked to be done and use your imagination to think of a way to get them done without any instruction which comes after the lesson.

Figure 24. Typical quote about exploratory nature of MPCT programming activities.
Acknowledgement

This report is based on work supported by the National Science Foundation through grants IIS-0829683 and DUE-0717877 and gifts from Texas Instruments Inc. and Calculex, Inc. Assistance in project planning and evaluation was provided by the NSF-BPC Computing Alliance of Hispanic Serving Institutions (CNS-0837556). 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 or other sponsors.

References