A Computer Scientist Teaches Gen Ed Math

Victor Winter
Department of Computer Science
University of Nebraska-Omaha
vwinter@unomaha.edu

Betty Love
Department of Mathematics
University of Nebraska-Omaha
blove@unomaha.edu

Michelle Friend
Department of Teacher Education
University of Nebraska-Omaha
mefriend@unomaha.edu

Michael Matthews
Department of Mathematics
University of Nebraska-Omaha
michaelmatthews@unomaha.edu

***Abstract*—A nationwide effort is underway to provide students pursuing higher education with options for satisfying general education (gen ed) math requirements. Within the context of this effort, computer science has an opportunity to introduce students to programming fundamentals and computer science principles while also satisfying gen ed math requirements. This paper is an experience report that describes initial efforts at the University of Nebraska-Omaha in piloting a course, satisfying the gen ed math requirements for non-STEM majors, whose content spans computer science, mathematics as well as the visual arts.**

***Keywords*-mathematical thinking; computational thinking; spatial reasoning; computer programming; visual arts**

I. Introduction

Historically speaking, college algebra has been used across the United States at institutions of higher learning to fulfill the general education (gen ed) requirements related to mathematics and quantitative reasoning. The genesis of the exclusionary idea that college algebra be the only option to satisfy gen ed math requirements solidified in response to the utility of calculus-based mathematics for war-related efforts during WWII and the Cold War that followed [1].

Providing college algebra as the only option for satisfying gen ed math requirements, however, is not particularly well-suited for students pursuing studies in non-STEM disciplines. For these disciplines, the mathematical formulas and procedures studied in college algebra are oftentimes seen as abstract and arcane and of little practical use [7]. In many cases, a student's time would be more effectively spent pursuing mathematical subjects other than advanced algebra (e.g., statistics or data science) [3] [2].

The mismatch between the educational goals of college algebra and the needs of various non-STEM disciplines has had a staggering negative impact on graduation rates [9] [7] [4] [6] [5]. The high failure rates typical of college algebra courses have provoked a national search for alternatives which would support students in successfully gaining the mathematical and quantitative reasoning skills and knowledge deemed necessary for their field of study. As a consequence, many universities now offer a variety of courses that can be used to fulfill gen ed math requirements. A course on statistics, for example, is now offered as an option to satisfy gen ed math requirements at many universities. In a similar vein, and in response to an executive order given in 2017, the California State University (CSU) system now offers a number of courses that satisfy gen ed math requirements. *Mathematical Ideas* and *The Power of Mathematics* are the titles of two foundation that CSU students in certain majors can now take in lieu of college algebra.

In response to gen ed math-related challenges, and with the support of the NSF, a group of faculty at the University of Nebraska-Omaha have developed a course called *Introduction to Mathematical and Computational Thinking* (MCT) which satisfies the general education math requirements, thereby serving as an alternative to the traditional college algebra pathway.

This paper reports on the findings of the pilot offering of MCT. The primary research questions related to the pilot are the following:

1) Can the novel educational content and environment used in MCT improve students' overall attitudes towards math?
2) What should be the appropriate depth of the computational and mathematical content for this student population?
3) Are assignments engaging?

A *student attitudes survey* as well as a *2-part mathography* were the instruments used to gather data relating to questions 1 and 3. Instructor and TA reflections as well as analysis of student transcripts were the instruments used to collect data for questions 2 and 3.

This work is funded in part by the National Science Foundation under grant IUSE-1712080. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

978-1-7281-5584-5/19/$31.00 ©2019 IEEE
DOI 10.1109/CSCI49370.2019.00151

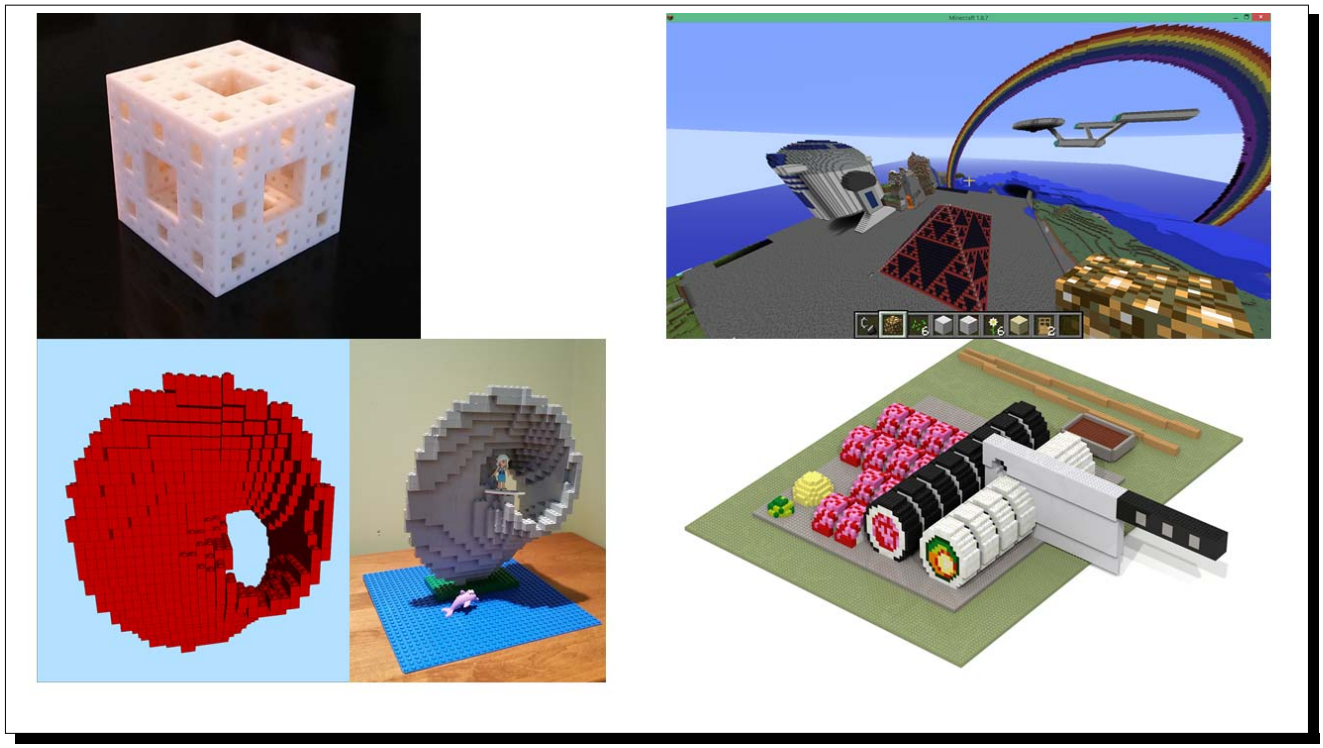

Figure 1: Artifacts created using Bricklayer.

II. Materials and Methods

Introduction to Mathematical and Computational Thinking (MCT) extends and modifies the content and ed tech infrastructure of an online ecosystem called Bricklayer.

Bricklayer [12] [13] is a collection of interactive web apps, downloadable software, YouTube videos, and reading material that facilitates a thoughtful and systematic exploration of construction techniques underlying 2- and 3-dimensional block-based visual art. Such exploration provides opportunities to exercise and develop spatial reasoning abilities as well as mathematical and computational thinking skills [14]. Bricklayer is open-source, freely-available, and can be found at:

https://bricklayer.org

Bricklayer programming is done using a (freely available) integrated development environment called the BricklayerIDE. More specificially, within the BricklayerIDE, programs are written in the general-purpose functional programming language SML, with graphical and tool integration capabilities provided by the Bricklayer library. Thus, Bricklayer provides users with an authentic programming environment[1] in which mathematical and computational skills can be developed. The output of Bricklayer programs are files which, through system-level scripts, are seamlessly integrated with third-party tools such as: LEGO® Digital Designer (LDD), LDraw, Minecraft, and STL viewers such as 3D Builder. The collage shown in Figure 1 highlights some of Bricklayer's artifact construction potential.

In addition to the Bricklayer graphics library (implemented in SML), the Bricklayer ecosystem also provides a second graphics library (implemented in javascript), with reduced functionality, called *Bricklayer-lite*. Bricklayer-lite is a visual programming language whose programs have a syntax consisting of assembled puzzle pieces which are similar in appearance to Scratch programs. Bricklayer-lite programs can be developed and executed through a web browser using a Google Blockly-based IDE, which we (also) refer to as Bricklayer-lite. A noteworthy capability of Bricklayer-lite is that, in addition to producing an artifact, the execution of a Bricklayer-lite program will produce a well-formed and well-formatted Bricklayer (ASCII) program text which can be executed (outside of the bricklayer-lite

[1] In this context, *authenticity* characterizes the degree to which a language realistically combines computational thinking with general-purpose programming.

framework) using the BricklayerIDE.

A. Participants

At the University of Nebraska-Omaha (UNO), prior to the offering of *Introduction to Mathematical and Computational Thinking*(MCT), college algebra was the only option to satisfy the gen ed math requirement. At UNO, more than $1,600$ students enroll in gen ed math every year and the DWF rate for the college algebra course is generally greater than 30%. Sadly, in gen ed math circles, the DFW rate mentioned is actually not bad. In a 2012 opinion piece in the New York Times, Hacker reported that the DWF rate for the City University of New York was 57% [9].

From the pool of 1600 UNO students that take gen ed math annually, approximately 32 students go on to pursue a STEM degree requiring calculus while the rest pursue non-STEM majors. To gain a better understanding student's backgrounds, the MCT instructor pulled the transcripts for all the students enrolled in the MCT pilot. These transcripts showed students that were very successful (e.g., mostly A's and B's) in their academic pursuits outside of math. The scholastic difficulties faced by these students, all of which were non-STEM majors, was overwhelmingly localized to college algebra and its prerequisites. In general, in this student pool it is not uncommon to find students that have taken (and failed) the traditional college algebra gen ed math course multiple times while otherwise enjoying a high GPA.

B. Educational Objectives and Topics

The educational objectives of MCT are as follows: develop an understanding of pattern and algorithm, both informally and formally using the discrete domain of visual block-based-based art. Physical construction tools (e.g., LEGO and graph paper) as well as virtual tools (e.g., web apps providing enhanced versions of graph paper) were employed to explore and develop techniques for creating 2D artifacts, especially artifacts containing patterns.

Frieze and wallpaper patterns (e.g., tessellations) represent a large and mathematically interesting set of artifacts in which the elements of pattern are size invariant (e.g., the size of tiles in a tessellation remains fixed). As such the coordinates for positioning tiles are governed by arithmetic progressions. Translational symmetries, reflection symmetries, and rotational symmetries provide attributes for understanding and classifying these artifacts.

Fractals such as the Menger sponge (3D), the Menger carpet (2D), and space-filling curves are examples of an altogether different class of artifacts. These artifacts are generally created using an inductive construction algorithm where the construction of a larger artifact involves creating one or more copies of smaller artifacts. In the case of fractals, geometric progressions provide the mathematics for determining coordinates and the size of these artifacts (measured in unit blocks) grow exponentially. Inductive construction algorithms can also be used to create artifacts whose coordinates are governed by arithmetic progressions as shown in Figure 2.

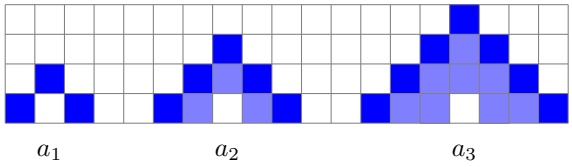

Figure 2: An artifact sequence [10].

Elementary cellular automata provide examples of construction algorithms that are rule-based which create complex artifacts of an altogether different nature. And last, but not least, is (patternless) pixel art, whose construction algorithms are labor intensive and place emphasis on the importance of reliable manual processes.

Informal (e.g., non-verbal) understanding of construction algorithms can be developed through manual processes (e.g., unplugged paper-and-pencil constructions or virtual constructions involving mouse-clicks). Such (manual) explorations are suitable for the construction of 2D artifacts. In this setting, composition (including inductive composition) and decomposition can be explored as techniques for understanding how artifacts can be constructed. This informal understanding of algorithm serves as the foundation for developing the ability to express such ideas in formal symbolic terms. The formalism in which this expression is developed is an environment consisting of the general-purpose functional programming language SML extended with a set of graphics libraries that provide suitable abstractions for creating visual artifacts of the kind described. From a very basic starting point, the sophistication of graphics abstractions and language constructs is then gradually increased thereby increasing the expressive power of the programmer. Through the use of parameterization the code creating an artifact can be generalized. The introduction of user-defined functions provides a means to a significant increase in abstraction and expressive power.

The transition from 2D to 3D introduces an additional set of challenges. Three dimensional artifacts generally contain significantly more blocks that their two dimensional cousins. For example, a 10×10 square contains 100 blocks while a $10 \times 10 \times 10$ cube contains 1000 blocks. This increase in size creates conditions that confront the limitations of third-party display tools such as LDraw and LEGO Digital Designer. To mitigate these issues, abstractions are provided and techniques developed to reduce the number of blocks contained in an artifact (e.g., hollow cubes can be created). However, number sense, coarse grained estimation, and counting skills are very beneficial in the design stage of 3D artifact construction. For example, an application of the inclusion-exclusion principle reveals that a hollow $10 \times 10 \times 10$ cube

contains $600 - 120 + 8 = 488$ blocks. The understanding of coordinates in three dimensions is also significantly more complex (and abstract) than the understanding of coordinates in two dimensions. In 2D, one can use graph paper (or its virtual equivalent) to develop an understanding of the positional relationships between the components of an artifact (e.g., the upper-right corner of a square touching the lower-left corner of a rectangle). This kind of prototyping can contribute significantly to the understanding of key aspects of an artifact. Reasoning about similar relationships in three dimensions is significantly more abstract and relies more heavily on spatial and mathematical reasoning.

The final portion of the course introduces predicates and general traversals. Predicates allow for property-based descriptions of artifacts and traversals enable the property-based construction of artifacts. For example, traversing an 8×8 square and placing a black brick in cells whose coordinates sum to an even number will yield a checkerboard pattern.

Collectively, the content of the MCT course contains elements of quantitative reasoning, symbolic reasoning, spatial reasoning, computational thinking, critical thinking, and mathematical thinking. The domains and nature of the material are such that the each of the stated educational elements can be explored to a considerable depth, thus making the general course suitable for a variety of audiences.

C. Data Collection

In the MCT pilot, a variety of data was collected, both qualitative and quantitative. Curricular successes and not-successes (not full failures, but less successful than ideal) are based on both instructor and student reflections. Student attitudes are the results of a pre- and post-course survey, as described in section III-C.

A major source of data was a reflective activity known as a "mathography" - an autobiography of their history with mathematics [8]. The post-course mathography assignment directed students to reflect specifically on their experience in MCT, and was administered by the education researcher. A total of eight students completed the post-course mathography. Students were asked to describe their peak experience in the course, their lowest experience in the course, good things about the course, and things they would suggest changing.

III. Results

A. Curricular Successes

Pixel Art – a highly engaging coding assignment in MCT involved the construction of a pixel art image where the image is selected by the student rather than being assigned by the teacher. Constructing pixel art requires a reliable construction process that includes (1) an overall construction algorithm (e.g., which brick to place next), (2) discipline in adhering to the construction algorithm during the implementation phase, (3) testing – frequent program execution, and (4) validation – manual confirmation that the image being created by the code is consistent with the original image. During construction, a variety of calculation-based problems naturally arise regarding the determination of cell coordinates. For example, a brute-force approach involves the calculation of coordinates manually in *absolute terms* as distances from the origin. A less labor intensive approach is to calculate coordinates in *relative terms* as offsets from locally known positions (e.g., the blue brick is to be placed 1 position to the right of previously placed brick). Subitizing[2] abilities can also be exercised during pixel art construction, especially for images composed of a small number of colors (e.g., black and white images).

Various algorithms can be employed in the construction of pixel art. Determining a suitable construction algorithm for a particular pixel art image is influenced by the properties of the artifact (e.g., its complexity and symmetries), the set of available brick shapes. One greedy algorithm involves partitioning the image into monochromatic regions and then "coloring in" regions in order of decreasing size. Other construction algorithms seek to leverage patterns or symmetry present in the image. However, the most general algorithm for effectively constructing pixel art is to decompose a pixel art image into a sequence of rows/column and then construct the artifact one row/column at a time. In this approach, you create a row/column, validate its correctness,and then move on to the next row/column.

> *One of my peak experiences was when we did an assignment involving pixel art. We got the freedom to choose any type of art picture that we would want to create using Bricklayer.*
>
> – Student 2

Art Shows – an art show is an engaging Bricklayer programming assignment where students are given a large degree of freedom. Art show submissions can be constrained through the use of themes. One example of an an art show theme would restrict submissions to the set of artifacts which can be constructed (primarily) using circles and rings. Another theme might focus on artifacts relating to architecture. A distinguishing feature of an art show is the presentation of student work. Specifically, student works of art (i.e., Bricklayer artifacts) are displayed for all the students to see. The (student) audience is asked to identify interesting aspects of an artifact such as use of color, structural complexity, creativity, and so on. A discussion can be initiated about the code that would be needed to recreate the artifact (e.g., is the artifact easily recreated by others?). In addition, the instructor can perform an inspection of the code used to create the artifact providing a live critique of

[2]Subitizing is the ability to "see" the number of elements in a collection without explicitly counting. Instantly recognizing the number of dots on the face of a die is an example of subitizing.

the aspects of the program such as organizational structure, choice of variable names, and even formatting.

Code-alongs – a *code-along* is an activity where the instructor and students develop code in the classroom in real-time in lock-step. Rather that taking notes in the traditional paper-and-pencil fashion, in a code-along students are expected to create executable code on their own machines. The requirement is that the code being created by the students needs to be well-formatted, executable and also correct. If needed, this expectation can be enforced by requiring students to submit their code at the end of the code-along. The benefit of code-alongs is that it lets students confront, in a supportive environment, the myriad of problems that can arise during software development. Problems range from syntax errors and type errors to system-level issues (e.g., location of files and proper file extensions).

With respect to the efficacy of code-alongs, an enabling feature of Bricklayer is that it is well suited to incremental software construction processes in which frequent execution of programs producing visually meaningful output (e.g., partially completed visual artifacts) is possible. Such incremental construction provides a robust framework for synchronization between instructor-student program construction and also demonstrates the value (e.g., reduced complexity) in establishing a tight feedback loop between the discovery and resolution of program errors.

During code-along the instructor frequently checks to make sure the programs being developed by students are up-to-date and functioning correctly. Such inspections encourage students to adopt a more proactive approach to problem resolution (e.g., they will raise their hands when they discover an error).

> *The code-alongs were also extremely helpful for understanding how the code works.*
>
> – Student 4

B. Curricular Not-Successes

Cellular automata and laces The rules governing the construction of elementary cellular automata are an example of a well-defined algorithm for constructing a set of artifacts. Another set of artifacts, having scale symmetry, are two dimensional versions of sponges, which we call *laces* (sometimes referred to by others as carpets). The construction of a lace is governed by rules that involve a composition of elements whose size is changing exponentially. The set of laces is a strict subset of a class of pattern referred to by Stephen Wolfram as nested patterns [15]. Both elementary cellular automata and laces are interesting because their construction algorithms are (conceptually) concise and the artifacts that result can (depending on scale) oftentimes be surprising, beautiful, unanticipated, and mathematically interesting.

The artifact shown in Figure 3 is a lace which was constructed through a sequence of mouse clicks.

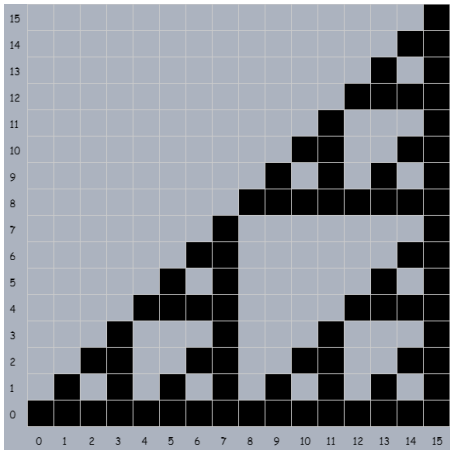

Figure 3: An example of a lace.

In spite of their mathematical and algorithmic appeal, exercises involving the construction of cellular automata and laces did not achieve their desired results. Overall, students had difficulty understanding the underlying construction algorithms and could not effectively apply these algorithms in general (i.e., to the construction of other cellular automata and laces).

C. Student Attitudes Survey

Students in the class were asked by the education researcher to take both a pre-course and post-course survey. Several questions, shown in Table I, asked about expectations and experiences in the course. The S-STEM Math measure [11] is a series of eight questions measuring confidence and interest in mathematics with questions such as "I am good at math" and "Math is hard for me." Three questions addressed students' attitudes about their experience and were asked only on the post-course survey; these are shown in Table II which shows both the mean result and the number of students who were positive (4 or 5), neutral (3), or negative (1 or 2) on each question. Nine students completed both the pre- and post-course survey. The questions were out of a five-point Likert scale, with a score of five being the high answer (e.g. "Strongly agree") and a score of one being the low answer (e.g. "Strongly disagree").

The post-survey results suggest that most students were very positive about the experience in the course, with a small number who were neutral or negative. There was a small non-significant positive change in the pre- to post-course survey results; most students were more positive after the course, with a few decreasing results. The small sample size contributes to the lack of significance, and a more robust

Table I: Pre- and Post-Survey Results (n=9)

Measure	Pre-survey) mean (sd)	Post-survey mean (sd)	t (p)
Right now how confident do you feel that you will/have succeed in MCT?	3.0 (1.12)	3.67 (1.12)	-1.33 (0.21)
How interesting to you is/was the content of MCT?	3.22 (0.97)	4.0 (1.12)	-1.49 (0.17)
How relevant is the class to your broader interests and goals? 2.89 (1.36)	2.22 (0.83)	1.20 (0.26)	
S-STEM Math [11]	3.18 (0.7)	3.25 (0.92)	-0.33 (0.75)

Table II: Post-Survey Results (n=9)

Question	Mean (sd)	Positive	Neutral	Negative
How glad are you that you took MCT?	4.0 (1.5)	n=7	n=0	n=2
Learning the math taught in MCT was fun	4.4 (1.0)	n=8	n=0	n=1
How did MCT compare to other math classes you have taken?	4.3 (0.8)	n=7	n=3	n=0

determination of findings will be possible in future iterations when more students enroll.

D. Supportive Environment

Six out of the eight student reflections (i.e., mathographies) mentioned collaboration and the ease of getting help as a major asset to the class. TAs were not only available to help students as they completed in-class activities, but also circulated and offered help proactively. Several students mentioned that they had been reticent to ask for help in previous math classes, but that receiving help from a professor or TA had made them see the value of getting assistance when they were struggling. The culture of the class, where making mistakes and needing help was normalized, seemed to really help students who would otherwise not seek out help. Time to work in class was also helpful, due to the availability of TAs and professors, but also other students: "we could talk about the assignments and brainstorm as a class" - Student 1.

IV. Discussion

In this section we discuss implications of this iteration of the course, and plans for future improvement.

A. Student creativity

Students were particularly taken with the open-ended nature of the assignments in the course, which allowed them the freedom to choose many elements of their projects. Unlike most math classes and many introductory programming classes, which offer highly constrained assignments, this course emphasized the artistic elements of output, which led students to choose personally relevant work. Student 1 felt overwhelmed by the initial pixel art assignment, reporting she thought, "I could never create something that looks that good!" But then she decided on a pixel art minion and was so proud of her work she showed her family. Indeed, unlike many introductory CS and mathematics courses, several students remarked on showing their work to others.

> *I showed everyone in my life what I did [on the pixel art assignment]. Even the kids at my job loved it.*
> – Student 2

B. Instructor Reflections

As an instructor, it is important to be aware that the students enrolled in MCT will be different from students enrolled in typical CS courses. In CS courses, it is not uncommon for students to be proactive with respect to assignments and grades. For example, a CS student may request an extension on an assignment and question an instructor as to why points were deducted on a particular homework assignment or test. In contrast, the students in MCT were more passive and it was not uncommon for students to fall behind on the submission of homework assignments which in some cases then led to attendance failures. In response, a set of interventions aimed at getting students back on track was initiated.

The physical classroom where MCT was taught was unoccupied in the hour prior to the course. TA's (and sometimes the instructor) would typically show up early to get the class setup (e.g., laptops ready). A growing number of students also began showing up early. The TA's would ask the students if they needed help with anything and the typical response was "yes". These interactions were very beneficial and resulted in the establishment of an informal tutoring center/lab that occurred right before every class.

From the perspective of instruction a couple of things were noteworthy about how students went about solving homework problems. One thing that was noticeable was that students oftentimes did not leverage in their construction process information relating to symmetry. For example, if one has calculated the coordinates of the midpoint of the bottom side of a square, then one can use this information to determine the midpoints of all the other sides of the square (in contrast to recalculating the midpoint of each side independently). It was also noticeable that the use of prototypes was not systematically employed by students in their artifact construction process. For example, a 2D artifact can be sketched on paper or created online using

the Grid. This visual prototype can greatly facilitate organization, positional calculations as well as visually expose any symmetries the artifact might possess.

In spite of a somewhat bumpy start, both the instructors and TA's observed a turning point was reached at the beginning of the second half of the semester. It was around this time that there was a sharp increase in the ability of students to correctly answer questions asked by the instructor during lectures. We have spent considerable time discussing what brought about this change. While we have no definitive answers at this time, we do believe the following.

- Throughout the semester, the MCT instructor placed importance on the use of proper terminology (e.g., Cartesian coordinate system, rectangular prism, formal parameters, actual parameters, etc.). We suspect that this "technical jargon" represented a new vocabulary for students which initially provided barriers to understanding lectures and problems statements.
- The course content was heavily spatialized. The instructor would create a visual artifact and they ask students to visualize transformations or modifications of the artifact (e.g., imagine rotating the artifact clockwise $90°$, what happens if these two artifacts are placed on top of each other, etc). We suspect that students had little prior experience with this type of engagement which initially provided barriers to understanding lectures.

C. Plans for Improvement

Future classes will be "more flipped" giving students more time to solve problems during class and ask for help. Also, the before class tutoring center will be institutionalized. A greater emphasis will be placed on prototyping (e.g., assignments will be updated to explicitly require prototyping). Furthermore, LEGO® sets have been purchased to provide a richer tactile prototyping environment.

Lastly, more emphasis will be placed on the recognition and understanding basic forms of symmetry. An instructor program has been created capable of generating random artifacts possing a desired set of symmetries (e.g., 2-fold symmetry and reflection symmetry). This program can also add desired symmetry to an existing artifact. Complementing this, an interactive web app is being developed, called *Mystique*, in which students can develop their symmetric reasoning capabilities.

V. Conclusion

Overall, the pilot version of the MCT course was considered a success. Of the ten students who completed the course, all passed. Two sections of the course will be offered in the future, and they have larger enrollments as news of this alternative spreads. This suggests that a course that combines art, programming, and mathematics can be successful in providing an alternative pathway to the traditional algebra route towards calculus, particularly for students who have negative prior experiences or for whom calculus is not a good fit. Carefully designed, computer science can fulfill a need and support solid mathematics learning.

References

[1] C. B. Allendoerfer. The Case Against Calculus. *The Mathematics Teacher*, 56(7):482–485, 1963.

[2] P. Burdman. Changing Equations: How Community Colleges are Re-thinking College Readiness in Math. *LearningWorks*, September 2013.

[3] P. Burdman. Should College Statistics Courses Substitute For Intermediate Algebra? *Stanford - The College Puzzle*, October 2013.

[4] P. Burdman. DEGREES OF FREEDOM: Diversifying math requirements for College readiness and graduation (report 1 of a 3-part series). Technical report, Policy Analysis for California Education (PACE), April 2015.

[5] P. Burdman. DEGREES OF FREEDOM: Probing Math Placement Policies at California Colleges and Universities (Report 3 of a 3-part series). Technical report, Policy Analysis for California Education (PACE), May 2015.

[6] P. Burdman. DEGREES OF FREEDOM: Varying Routes to Math Readiness and the Challenge of Intersegmental Alignment (Report 2 of a 3-part series). Technical report, Policy Analysis for California Education (PACE), May 2015.

[7] J. Christopher Edley. At Cal State, algebra is a civil rights issue. *Highlighting Strategies for Student Success*, June 2017.

[8] C. Drake. Turning points: Using teachers' mathematics life stories to understand the implementation of mathematics education reform. *Journal of Mathematics Teacher Education*, 9(6):579–608, 2006.

[9] A. Hacker. Is Algebra Necessary? *The New York Times*, July 2012.

[10] D. Sladkey. Visual Patterns Pattern #189. www.visualpatterns.org, 2018. Accessed: 2018-06-17.

[11] A. Unfried, M. Faber, D. S. Stanhope, and E. Wiebe. The development and validation of a measure of student attitudes toward science, technology, engineering, and math (s-stem). *Journal of Psychoeducational Assessment*, 33(7):622–639, 2015.

[12] V. Winter. The World Needs More Computer Science! What to do? In D. Conway, S. A. Hillen, M. Landis, M. T. Schlegelmilch, and P. Wolcott, editors, *Digital Media, Tools, and Approaches in Teaching and Their Added Value*, pages 119–141. Waxmann Verlag GmbH, Germany, 2015.

[13] V. Winter, B. Love, and C. Corritore. The Bricklayer Ecosystem - Art, Math, and Code. *Electronic Proceedings in Theoretical Computer Science (EPTCS)*, 2016.

[14] V. Winter, B. Love, and C. Corritore. The art of the Wunderlich cube and the development of spatial abilities. *International Journal of Child-Computer Interaction*, 2018.

[15] S. Wolfram. *A New Kind of Science*. Wolfram Media, Inc., 2002.