2011-06-13

Investigating the Influence of Computer Programs on Perception and Application of Mathematical Skills

Neil M. Bly
Brigham Young University - Provo

Investigating the Influence of Computer Programming on Perception and Application of Mathematical Skills

Neil Bly

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirement for the degree of

Master of Science

Peter Rich, Chair
Keith Leatham
Geoff Wright

Department of Instructional Psychology and Technology

Brigham Young University

June 2011

ABSTRACT

Investigating the Influence of Computer Programming
on Perception and Application
of Mathematical Skills

Neil Bly
Department of Instructional Psychology and Technology, BYU
Master of Science

Existing research suggests an intuitive relationship between mathematics and computer programming. These previous studies have focused primarily on the cognitive connection and have ignored the potential impact of programming on an individual's perception and application of mathematical skills. By surveying and interviewing a variety of participants, this study aims to provide a descriptive foundation for the experiential side of cognitive correlations and causalities. These phenomenological accounts, garnered from individual interviews of seven different programmers, indicate four specific areas of interest. First, learning to program provided context for many abstract concepts. Second, programming illustrated the important distinction between understanding the application of math in a specific situation and the execution of a known procedure. Third, programming habits helped participants divide complex problems into more manageable tasks. Finally, the necessity of solving a programming problem provided motivation and eliminated apprehension toward mathematics.

Keywords: mathematics, computer programming, attitude, influence, phenomenological, logo

TABLE OF CONTENTS

Introduction

Previous research suggests an intuitive relationship between mathematics and computer programming. These studies have focused primarily on the cognitive connection and have ignored the potential impact of programming on an individual's perception and application of mathematical skills. By surveying and interviewing a variety of participants, this study aims to provide a descriptive foundation for the experiential side of cognitive correlations and causalities.

Literature Review

Cognitive Benefits of Computer Programming

The educational community has seen numerous studies showing the various cognitive benefits of computer programming (Clements & Gullo, 1984; Linn, 1985; Palumbo, 1990; Subhi, 1999). Programming involves breaking down complex problems into sub-problems and requires the programmer to sequence a solution of step-by-step instructions. In addition, modern programming languages facilitate an ideal environment for testing and modifying these solutions as programmers methodically debug and re-write code. The combination of these cognitive processes, combined with the computer's ability to allow interactive and immediate feedback, suggests potential support in problem solving effectiveness (De Corte, Verschaffel, & Schrooten, 1992).

In his book *Mindstorms*, Seymour Papert (1980) stated,

> In my vision, the child programs the computer and, in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building. (p. 5)

Claims such as Papert's, that programming creates an "intimate contact" with mathematics, suggest that programming's cognitive benefits extend beyond increased problem solving abilities. Some research does in fact suggest additional benefits of computer programming on other aspects of cognitive abilities. In one example, Soloway, Lockhead & Clement (1982) found that students with experience in learning BASIC computer programming improved their word-solving ability on algebra problems. In another 12-week study, eighteen 6-year-old children were assessed in their abilities in vocabulary, impulsivity/reflectivity, and divergent thinking. The children were then randomly assigned to either a computer programming (Logo) or computer-assisted instruction (CAI) treatment group. Post-testing revealed significantly higher scores on measures of reflectivity and divergent thinking for the programming group compared to those of the CAI group. The programming group additionally outperformed the CAI group in areas of meta-cognitive ability and ability to describe directions (Clements & Gullo, 1984). A recent report by the National Mathematics Advisory Panel (Flawn, 2008) discussing the use of computers in mathematics education stated the following:

> Research indicates that learning to write computer programs improves students' performance compared to conventional instruction, with the greatest effects on understanding of concepts and applications, especially geometric concepts, and weaker effects on computation…The Panel recommends that computer programming be considered as an effective tool, especially for elementary school students, for developing specific mathematics concepts and applications, and mathematical problem solving abilities. Effects are larger if the computer programming language is designed for learning (e.g., Logo) and if students' programming is carefully guided by teachers so as to explicitly teach students to achieve specific mathematical goals. (p. 51)

Taken together, these statements indicate an important relationship between learning mathematics and computer programming. The following two sections provide possible evidence that the National Mathematics Advisory Panel may have based their claim upon when referring to the important relationship between mathematics and computer programming.

Mathematics as a Predictor of Success in Computer Programming

In efforts to predict an individual's success in computer science courses, researchers have analyzed high school math achievements scores, standardized national tests such as the SAT, and enrollment in previous math courses (Leeper & Silver, 1982). In these studies mathematics has continually surfaced as the strongest positive predictor of performance in computer programming (Bergin & Reilly, 2006; Byrne & Lyons, 2001). Furthermore, prior background in mathematics proved to be a significant variable in the retention of students in introductory computer programming courses (Konvalina, Wileman, & Stephens, 1983).

After conducting a study of 110 introductory computer science students, Byrne and Lyons (2001) confirmed findings from earlier research supporting the relationship between mathematics ability and the probable success in computer programming. In response to these findings, the authors made the following suggestion:

> There is a belief that the concepts which a student has to comprehend in order to master mathematics problems are similar to those for programming. Mathematics aptitude is thus often a pre-requisite for acceptance into computer science . . . These results support the theories and research to date. (Byrne & Lyons, 2001, p. 51)

These predictors suggest that there is a positive relationship between mathematics and programming, establishing a precedent to investigate math's influence on computer programming proficiency. Considering the previously described benefits of computer

programming, it would seem logical to suppose that there is overlap in the primary cognitive skills utilized in these two subjects (Byrne & Lyons, 2001) and that engaging in one improves both cognitive ability and motivation for the other. Stated simply, if math is a consistent predictor in computer programming performance, would it not follow that experience in programming be predictive of success in mathematics?

Logo Programming in Mainstream Education

The Logo programming environment was the clearest attempt to integrate forms of computer programming into mainstream school curriculum in the past. Logo was created in 1966 as part of a constructionist learning initiative by Seymour Papert (Agalianos, Noss, & Whitty, 2001). Papert originally designed Logo specifically to teach various concepts of mathematics to children (Feurzeig et al., 1970).

Despite mixed reception, the Logo experiments provided educators with hints of the possibilities learning to program could offer. Several studies demonstrated Logo to have a substantial impact on a student's performance in mathematics. Clements, Battista, & Sarama (2001) found programming in Logo to show an increase in students' understanding of shape, measurement, symmetry, and arithmetic processes. In research conducted in elementary algebra, experience with Logo suggests a complementary link for children in constructing algebraic meaning in a non-computational context (Noss, 1986). The study illustrated that working with Logo may allow children to develop primitive conceptions of mathematical ideas to which they may link their further understandings.

Logo has demonstrated a similar influence in cognitive ability. A 14-week training of 48 children in Logo and other computer-aided instruction resulted in the Logo group exhibiting a significantly higher performance in conflict resolution, rule determination, and self-directed

work. The Logo group's behavior similarly suggested a significantly higher increase of meta-cognitive function (Clements & Nastasi, 1988). A study of 286 gifted students compared the impact of computer-assisted-learning games and learning to program in Logo finding that students who learned to program increased their problem solving abilities more than the computer-assisted-learning group (Subhi,1999).

While Papert's intention of Logo was to initiate a fundamental rethinking of educational practices and the role of technology, the influx of computers in school classrooms in the early 1980s saw a wide variety of application (Agalianos et al., 2001). In response to this misguided usage, Papert disappointedly commented, "It started a route that made it [Logo] a school thing, whereas originally...it was instead of school" (Agalianos et al., 2001, p. 483).

Shortcomings of Computer Programming and Mathematics Studies

In many of the experimental studies in the aforementioned literature, mathematics skills were tested after only a relatively short training period in computer science. This limitation led to contradictory or inconclusive evidence among the different researchers. In a review of the effects of programming on learning, Palumbo (1990) offers insight into these shortcomings. Palumbo explained how many studies failed to account for the time it takes to learn to program as well as competency in programming in the first place. For example, Gorman and Bourne (1983) reported that third-grade children programming in Logo for 1 hour a week performed significantly higher on a test of rule learning than did children with 1/2 hour a week of programming exposure (Clements & Gullo, 1984). De Corte, Verschaffel, and Schrooten (1992) also found that when they accounted for programming competency, they found positive effects on other problem-solving skills. Thus in order to investigate any link between math and

programming, students must first be able to demonstrate a certain level of competence in either domain.

There seems to be an implicit assumption that programming will improve one's skills in mathematics in some way. Byrne and Lyons (2001) comment that "the link between mathematics ability and programming is widely accepted, although it's empirical demonstration is questionable" (p. 49). Statements like these acknowledge the assumed relationship, yet admit no immediate references to support such wide acceptance. The two subjects are treated as having an obvious relationship, almost intuitive in nature. Tooke (2001) states, "there is a trivially obvious connection between mathematics and the computer. It is, in fact, a symbiotic relationship" (p. 2). Such claims lead us to ask why mathematics and computer programming are consistently grouped together.

While research suggests debatable evidence of increased mathematical ability via Logo studies and Computer Science predictors and that success in programming and success in mathematics are linked, they have neglected other aspects that learning to program might have on one's mathematical abilities. As Palumbo (1990) noted,

> Although it does not seem possible to achieve expertise in programming after 75 or even 100 hours of programming exposure, this does not limit the possibility of some type of problem-solving transfer. Although expertise and years of practice are required to achieve any type of low-road, automatic transfer from one problem situation to another, there is a possibility that some type of high-road, mindful transfer might occur…longer, more intense exposure should be provided to determine if any generalized transfer will occur. (p. 81)

The level of transfer is therefore greatly dependent on exposure to activity (Pea & Kurland, 1987). While no doubt important, the specific focus on establishing a cognitive connection between mathematics and computer programming has ignored the effect learning to program might have on a student's attitude toward mathematics and its perceived utility. Such research tells us little about what Van Manen (1996) refers to as the *lived experience* of individuals who have learned to program. Understanding how learning to program might motivate a learner to pursue or use mathematics in ways s/he would formerly not have pursued, can be just as compelling of an argument to its importance as any cognitive benefit measured through objective assessment of one's resulting mathematical abilities. The purpose of this study was to interview a variety of professionals with a diverse application of computer programming skills in order to better understand the effect it may or may not have had on their perception and use of mathematics.

Methods

The individual participant's personal narrative is an extremely valuable component in understanding the influence programming has had on academic, professional, and daily experiences. Research supports that stories are more memorable and better support learning and understanding than nonstory narratives (Patton, 2002). Concerning storytelling, language scholar Richard Mitchell (1979) states,

> Our knowledge is made up of the stories that we can tell, stories that must be told in the language that we know. (Even mathematics is a "language" that states propositions and tells stories. It's a very elaborate form of "play" language. That's why it's such fun for those who speak it well.) Where we can tell no story, we can have no knowledge. (p. 13)

Storytelling enables participants to relate detailed instances where they perceive the benefits of programming and the effect this may or may not have had on their mathematical or other understanding. Inasmuch as people's stories reveal the unfolding narrative of their perceived life, we sought to engage participants in telling the story of when they learned to program and other stories of how it affected related events in their lives.

Participants

The research consisted of two phases: A preliminary selection assessment survey was first conducted (Appendix A) to identify potential participants, followed by a more focused and in-depth interview with the selected candidates (Appendix B).

Maximum variation purposeful sampling methods were used to select the research participants in order to gain diverse programming application and experience (Patton, 2002). The initial assessment was an online snowball survey sent to participants allowing potential candidates to self-identify as competent in different programming abilities. Participants were sought out particularly in areas where the individuals would most likely have been exposed to adequate programming and mathematics experience such as computer programming, engineering, Web-design, and other related venues and online forums. Students taking the survey were asked to recommend other individuals with programming experience.

Assessment questions focused on the following items. See Appendix A for further details.

- Are you currently involved in computer programming? If not, how long has it been since you were?
- How many hours per week do/did you spend programming?
- How many years of programming experience do you have in any of the following languages?

Individuals were not compensated for their time, and participation was conducted on a purely voluntary basis. Participants needed to be completely willing to discuss their previous academic performance, including both positive and negative experiences. Participants needed at least 2-3 years of programming in order to participate, though ability in a specific programming language was not a priority. Due to the duration and personal nature of the interviews, availability and local proximity of participants was a practical factor in the candidate selection process. Though there was no intentional gender discrimination and the project allowed for both male and female participants, only 3 individuals out of the 60 that completed surveys identified themselves as females. Of these responses, seven local male participants with diverse cultural, academic, and professional backgrounds were selected for conducting open-ended interviews regarding their experience. Participant cultural and academic diversity spanned the U.S., Canada, England, and Nepal. Programming backgrounds ranged from self-taught to formal high school and university instruction at both undergraduate and graduate levels. The professional occupations of the selected individuals included Web, mobile, and online education programmers, a geographer, a bioinformatician, and a director of development. While potentially useful in future studies, the extraneous data provided by all candidate survey responses is outside the scope of this study and results of the survey were strictly limited to the selection process.

Analysis

As this study focuses on individual perception, phenomenological methods were used to interpret findings. Phenomenological analysis seeks to expose the lived experience of an individual or group of people and understand this experience within a particular living context (Crotty, 1998). Van Manen (1996) stated,

> [The] aim of phenomenology is to transform lived experience into textual expression of its essence—in such a way that the effect of the text is at once a reflexive re-living and reflective appropriation of something meaningful: a notion by which a reader is powerfully animated in his or her own lived experience. (p. 36)

In this approach it was equally important to highlight the structure of the experience and events as they occurred as well as the individual narrative, or interpretation of events. Thus the analysis of individual interviews sought to highlight the structure and story of each individual.

Participants were individually interviewed regarding their experience prior to, during, and following their programming experience. These interviews lasted from 1-2 hours and sought to understand specific stories surrounding the influence programming has had on their learning, focusing specifically on mathematics. The interviews were digitally recorded for archival and retrieval purposes and subsequently transcribed for analysis (see Appendices D - J).

Textual and structural analyses were conducted following Moustakas' (1994) recommendations and procedures of pulling out the invariant horizons from a participant's recounting of an experience (Appendix C). Experiences were reconstructed into narratives that represent the essence of learning to program in relation to its effect on mathematics. Out of these narratives, four important underlying themes were identified: (a) programming provided concrete examples of abstract mathematics, (b) programming provided the "why" to the mathematic "how", (c) programming helped participants learn to break tasks into sub-tasks, and (d) programming provided a context wherein necessity removed obstruction.

Findings

To best understand the unique experience of each participant, the following section first presents an individual vignette of each participant. These vignettes are a combination of the

textual (i.e., direct quotes) and structural elements (i.e., identified themes) of each participant's interview.

Vignette: Asim

Originally from Nepal, Asim became interested in programming at a young age. After his family moved to the US, Asim started working as a programmer while in high school. He currently lives in Utah where he studies bioinformatics and programs for educational applications.

Growing up in Nepal, Asim's favorite subject "was always science." Regardless of the field, science seemed to come easily to him. "It was just kind of natural, I just liked it [science]." He enjoyed other classes as well, but stated, "I didn't like the Nepali classes. I hated the Nepali classes." All students in his school were required to take computer classes beginning in the eighth grade. They began with "Lotus 1, 2, 3 at the time. It was Lotus, and it was D Base, and it was . . . Word Perfect." The classes that introduced him to programming were visually based. "It was visual. And at that time my favorite thing was moving the circles around in different paths. Ok, so let's see if I can do like straight across, down, like diagonally." This type of learning gave him immediate feedback, and after his very first class, Asim "really started liking it [programming]." He began earning money for his programming work when he was in high school. "After coming to the US in my sophomore year, I started programming for my part-time job, and most of the learning actually came from doing it myself, not from classes."

In Nepal Asim had math classes concurrent with his programming courses. "Math was my second least favorite subject actually. I had absolutely no care for math. I tried getting out of homework as much as I could for math, and I was really bad at it." Not until an important national exam approached did Asim began to study math more earnestly, because "you're

expected to pass that with flying colors. And for that, then in the ninth grade I started taking math more seriously. I actually started liking math, and then I got really good. I became one of the best in my classes, after that." Asim believed that his ability to divide problems into smaller pieces assisted him in math. According to him, programming "taught me how to dissect things into smaller chunks," and that "when you're doing some kind of mathematical algorithm, that [breaking it into increments] is how you approach it."

Asim explained, "But definitely before that [learning to program] . . . when I approached math I never thought of dissecting the problems into smaller chunks. 'Ok, let's derive this first and see where we can go from there.' It was never like that. It was always 'how do I get from point A to B?' You really don't think between point A to B there would be A to C, C to D, D to E, E to B . . . It's actually usually that way when you're solving math problems. . . . I think programming taught me how to go to those intermediate points and derive stuff. . . . I think that helped me."

While in college Asim continues to study computers and is currently majoring in bioinformatics. He has taken classes in data structures, discreet structures, and enterprise level programming. One class in particular was very statistics heavy, and Asim felt he didn't have the math background to figure out the needed algorithm. "I had to go and ask my professor for it, to teach me how to do the math behind those things. He taught me and then I could do it." In statistics, he found calculus useful. "For stats, you have to do a bit of advanced calculus, if you want to optimize it. That did force me to learn some extra things, just because I tried to make the program faster … When you actually use calculus, it's computationally easier. It's harder to derive, but, when you start calculating for a machine, it would be much easier in an equation than just looping through and adding stuff to it." Another time, in a statistics class, he used his

programming skills to quickly write a program that could solve a complex math equation involving incrementing. "That was cheating, kind of. But, yeah, I did get the very correct answer." Without programming, Asim thinks people could have more difficulty grasping this process. When it comes to incrementing, "people who have not had programming will not understand it as much as someone who has had programming because that is how a computer does it."

Asim has found that there are many mathematical and programming concepts that overlap. "Definitely computational time. Calculus really comes in handy when this is converging to this, especially for something like sorting algorithms and everything; calculus comes in very handy." Programming made "math much more interesting." It gave him the chance to manipulate the rules. "It [using the rules your own way] makes you kind of less scared of math. People are scared of math. . . . When you start getting into programming, there are so few rules, and everything else you make up. Same as in math, there are few rules, that you can apply it by however you feel like." In both math and programming "you just have this set of tools that you know, and you're given this problem that you've never seen before, and you solve it." Math became not only more interesting, but a tool to help solve the problems he was confronted with. With the tools and the freedom to use them, Asim claimed, "you're not scared of something you don't know the solution to."

Asim currently programs games and uses "a lot of math" in his work. "It definitely helps in optimizations, shuffling, and making random things happen. You can kind of control it if you're good at math. You can use it better if you're good at math." According to Asim, math also helps when "optimizing arrays, sorting arrays, and shuffling arrays." Depending on the program being used, he has needed math to control positioning and drawing. "Knowing that

[algebra] always helped your equations." After college he would like to work with "tech and science, I want it to overlap." Whether in bioinformatics or other fields, "I always want to be involved in cutting-edge stuff."

Vignette: Ben

Since his youth, Ben was always interested in science. He began programming in college and received a degree in computer science. He lives in Utah and works as a web programmer.

According to Ben, of all the subjects in school "biology and computer science were my two favorites." Other classes proved more of a challenge. "Actually I did have a rough time with math … probably math and arts." These were both areas of difficulty, but the obstacles were different. "With the arts, definitely a lack of interest. With math it was more of a struggle. It was more difficult for me to do well. It [math] took more time and effort … where arts was just no interest." In college Ben took a computer science class when he was trying to decide on a major, "and I landed on that [computer science] and I decided to pursue that."

Most of Ben's programming experience came from his college courses, but "probably twenty percent or thirty percent has been through books, not college." He understood that computer programming had to be learned gradually. "I don't think you learn the entire thing [programming] at one time. It's over a period of time; you pick more things up. But most of it was learned academically, and then, other stuff, along the way." Computer science affected Ben's approach to problem solving. "I definitely take a divide and conquer approach in most everything I do. … I would say it [the divide and conquer approach] leaked over into the rest of my life, and I started doing other things that way as well."

Knowledge of math was essential in Ben's programming classes, and he thought he was "pretty average in them [math-intensive programming classes]." He presumed other

programmers would do better in math, and was surprised to find out his classmates' abilities varied. "I also assume people [in programming classes] are really good at math too. I think when I look at people around me and how they perform [in math], it's pretty even. But some people obviously are on different levels and they're either really good or really bad at it."

Programming facilitated a better understanding of logarithms for Ben. "In the past when I was dealing with logarithms, I didn't quite understand why they were necessary, and then from programming, it's like, that's a good use of them." Especially when he was "figuring out how long a function will take, you can use a logarithm for that. I think that's when I had to go back and learn exactly what a logarithm was." He understood that there was a concrete application of logarithms and that math was an instrument to improve his programs. "When you're doing an algorithm and you want to make it as fast as possible, you're using mathematical concepts to increase in speed." According to Ben, "it [logarithms] was easier to remember, because I had an actual use for it." Programming became "more like a useful tool for math." His attitude about math changed, "I think it's [math has] always been a struggle, but I think it's helped my attitude, basically." Programming allowed Ben to "see things from a new perspective."

While programming in Assembly, Ben's comprehension of another math concept was increased. "Especially with converting from whatever base you're working in, if it's a hexadecimal or a base 2. That is one thing [in programming] that's definitely helped me to understand that [math] concept better."

According to Ben, math and computer programming are inherently linked. "I think math overlaps it [programming] completely, in one way or another. . . I think pretty much everything in computer science you could link to math at some level or another." He thinks that programming relies more on math than math does on programming. "I think math would lose

something without programming, but not as much, where programming would lose almost everything without the math." Because math and programming are so closely related, Ben believes “they should have more computer science classes that are more mathematically based, as well. I feel like they should have more specific math classes for CS majors. . . Classes on modeling things mathematically that you would then program would be pretty useful. Methods for taking a real world problem and creating a mathematical solution or model of it, and then programming it from there.”

Ben has learned to program in multiple languages including C, C++, C#, PHP, and Perl. “I program now, and I plan to continue to be a programmer for the rest of my life.” At this point in his career, Ben’s work is not math intensive, but “in the future, I don’t know. It may become really intensive, I don’t know. It very well could be.”

Vignette: Brian

Brian began focusing on academics while in college where he did well in both programming and math. He graduated with a bachelor’s and master’s degree in information systems and now works as a contract programmer building Internet applications.

Brian was involved in several sports in his youth, and academics were not always his focus. In high school Brian graduated with a GPA of 2.6. “I guess I went skiing just about every day and did not pay attention in classes.” Many of his days were spent outside of school, but Brian still grew to enjoy certain subjects more than others. “I really liked math and science, and I hated English and humanities and anything that had to do with that.” The distinction between the subjects he enjoyed and those he didn’t had to do with rules. “It just felt like there were rules [in humanities-related classes], that somebody made up somewhere . . . because with math,

obviously math also [had] rules and laws that other people had set up, but I felt like they were universal things that could be applied and proved over time."

Even with a passion for math and science, Brian wasn't introduced to computer programming until college. "My parents didn't even have a computer until I graduated from high school, so that was a big part of it." In college Brian did extremely well in programming. "My programming classes were the ones I excelled the most at because I really enjoyed them." During the completion of his bachelor's and master's degrees in information systems, he only took two programming classes. "I really enjoyed the [Java] class, so I ordered about six books after the class was over and started learning programming on my own. Everything that I've learned with Flash and ActionScript and Flex, has all been on my own. There was no formal education in that." By the time he graduated from college, Brian's GPA was much improved from high school, finishing with a 3.8 for his bachelor's and a 3.93 for his master's.

Brian also did well in his college math classes. "I took calculus one through three in college, and I got an A in each of those classes. So, I just enjoyed it . . . I think it [mathematics] is more of a mindset than anything. It's a quantitative way of looking at things in the universe. But being good at mathematics mostly just means that you understand the theories and the rules and how to apply them."

While Brian was working with computers, knowledge of trigonometry was required to do his job. Because he had never taken a trigonometry class, "Anything that has to do with sine or cosine, or any of that type of thing, I've had to figure out on my own . . . I always thought that geometry and trig were kind of worthless, because I never planned on being an architect, or an engineer, that would actually need to use them. But once I jumped into programming and had to start using those all the time, I started realizing that it was actually an extremely useful practice

to know those things, and I wish that I'd learned them in the first place." There have been many situations when an understanding of trigonometry has been essential for Brian. "I was creating a clock animation in Flex . . . but I actually wanted to draw it to the stage, and to do that I had to know the exact mathematical coordinates of where the clock would be placed. And since the clock was refreshing at thirty frames per second, that meant I needed to refresh my angle, and my trajectory, and everything else thirty times per second, three hundred and sixty times a minute, and it [trigonometry] became really important." Working in a 3-D environment also requires mathematics for "drawing glass buttons, and doing things in 3-D space and trying to make them look spherical." For Brian, programming "just helps math be more realistic."

In these cases Brian had to seek out the mathematical knowledge he lacked in order to solve programming problems. However, had the same information been presented to him in math class without a practical application, Brian "probably wouldn't have been interested at all. I feel like a practical application of math always changes my attitude of it. And, to add to that, I think that when you're in a math class, the best way to learn is to add practical examples." Brian felt that these practical implementations made the trigonometry concepts more concrete. "I think that's what it [relevant application] does and when people go to school for engineering or something, I think the engineering aspect of things solidifies the mathematical concepts. It makes them actually make sense instead of just being theories."

Brian found that math shared the same problem-solving approach inherent in programming. "I think the relationship is that they [math and programming] are both quantitative. With math you're trying to come to a solution for a problem, and the route you take to get there is the puzzle or the mystery. That's exactly what you do with programming. You have an end that you're trying to get to in a problem, and a barrier in between, and how you get

there is basically the puzzle that helps you get down that path." This step-by-step, methodical way of solving of puzzles became a habit for Brian. "It basically just creates a habit of always thinking about what this one choice is going to do, and how it's going to affect every other aspect of the program. . . . It's like playing chess for your first couple times, you don't think about every possibility that could happen down the road, you just think about your current move. But as you play, you start realizing that if you make this move then this could possibly happen and this could possibly happen and all these things could happen, and your thought to make a single move increases . . . and I feel like that's what programming does. It basically just creates a habit of always thinking about what this one choice is going to do, and how it's going to affect every other aspect of the program."

Outside of his work, this way of thinking still pervades. "It makes me methodical, probably too methodical. Some things should just be simple, but the truth is that as a programmer, I think way too methodically, and that's how it's affected my life." Currently, Brian is "doing things in Flex and building rich Internet applications." In the future, "my career goals would be to be recognized by Adobe consulting and possibly become part of their team."

Vignette: Jason

Jason, originally from Washington, began programming in middle school. As an adult he became more interested in social issues and studied environmental science in college. Jason would like to use technology to help people and improve the world around him. He currently works as a programmer in Utah.

In elementary school his math teacher was very motivating. "She was the head of technology for our district so we all had brand new Mac LC II computers, and we were using computers all the time and she got me to sign up to be part of the Math Olympiad . . . We got like

fifth place out of a couple hundred schools. We thought it was really good." The team did relatively well in the competition, but "I didn't fit in very well, and so I knew that I was smart and I knew that I had this capability, but the other kids on our team thought I was a weirdo and they'd make fun of me. So, I had some confidence, but I think, socially, I didn't have a lot of confidence in myself."

During middle school Jason became interested in computers:

> I got kicked out of PE because I spit on the floor, and I think I punched a kid in the face because I didn't like jocks, and I didn't like sports. So I was banned from PE for my seventh grade year. I just sat in the office that period. And then at some point, I don't know why, I got curious, I had some cool teachers in fifth, sixth grade, and I started getting into computers.

He began bringing programming books to school. "I would just read those while I had to sit through that period. And so I started studying first Visual Basic, and then I taught myself Java, and through middle school I was learning Java completely on the side. I didn't even have enough computer equipment to really program." At the same time, Jason's math grades fell. "I was having a lot of social problems that actually really affected me academically. And so, my math stuff kind of fell out at that point. And then, I think what happened was, once I got into programming, I realized it kind of re-sparked what I had earlier with math."

In ninth grade Jason was able to take a programming class at school. "It was the first time I'd ever actually been able to program, to have the actual equipment to start doing stuff and applying it. I took that class and there was a web design class, and I started writing in ninth grade, I wrote a program that keeps track of which homework assignments each teacher had in

Perl . . . it was my idea, as far as I remember, to create this program where teachers could enter in homework assignments and then students could put in their student ID number and check them." He felt that he was struggling academically, but "I was trying to do really innovative things, it was really cool . . . I got passionate about it and I realized that with programming I could solve problems that hadn't been solved before." He started a web design company called Seattle Web Design when he was 14. "All through high school I was working on websites. I started working for a web hosting company when I was a sophomore in high school, and I basically continued in that industry." It wasn't just programming problems he was solving. "I realized that I could do cool things with programming, and actually see myself being successful, I saw a path for myself to get out of the crumby situation I was in and so, the problem, I guess, was my life that I was figuring out how to solve with computers."

There were times when Jason's programming required him to learn more math than he knew at the time. "I'd have programming problems that I didn't have the math yet in my mind to solve, and so I'd go to the next grade math teacher at my school and say, 'How do I solve this problem for this program?' I was making one of these brick games, that you bash the bricks with a paddle, like Breakout. I didn't really have the math to understand the angles everywhere, and so I'd go to these teachers and have them help me solve these math problems."

Jason struggled through his high school math classes. "I was above average in math, but when it got to complicated stuff like calculus, my memory just made me suck at it." The time it took to memorize math formulas and then to solve problems also was a difficulty in college. "I didn't have a good enough memory to memorize all the formulas. One of my biggest problems in math was that I didn't have enough time. If I had the time, I could have basically solved all the formulas and figured out what I needed to do." Jason doesn't feel he needs to rely on

memory as much with computers. "With programming, I don't need to have a good memory, I have references I can just quickly access the information that I need to solve the problems, so that is no longer a limitation for me. So I think inherently I have some good math problem solving skills, but I don't have the brain . . . I think of programming as an alternative to math. I can't solve this using normal math but I can write a program that will solve it for me."

Jason's attitude toward math was affected by his programming experience. He thinks that without programming "I'd probably hate it [math] more. Programming allows me to use math to help me do what I want to do. I think without it I'd have less patience for it, and I couldn't even see the purpose in it . . . I think without the programming, I would be lost with the math. I would be more confused about it. Whereas with the programming, it's helped me put it to use effectively."

Some areas where Jason sees math being used in programming are loops, vectors, and storing data. "Also, mathematical concepts of optimization. You have to figure out what's the quickest way of accomplishing something. How do you solve it in the least number of steps?" A science teacher once told him, "Mathematicians solve theoretical problems, but scientists actually do something cool with them." Without programming Jason thinks he would turn to some other science to see the value of math. "Physics is just completely based on math, and I would look to that to use the math in a useful way."

At work Jason's doesn't do a lot of hard math. "There's this running joke at where I work that anytime we have a project manager come in and they'll say, 'We need to do this and this,' and they'll start throwing some math at me. I'll be like, 'I don't know math, I don't do math, I'm a programmer.' Obviously programmers are supposed to be really good at math. I don't consider myself to be a math expert or a math genius." Jason knows that math is an

important part of computer programming, but likes to have the computer solve problems for him. "I use programming as a tool to help me to use, to kind of harness the power of the math . . . I'm too lazy to be a mathematician. I don't want to go that deep, and so I expect the program to solve the math problems."

In community college Jason became passionate about his sociology and social science classes. He received a degree in environmental studies and would someday like to start his own business that integrates both IT and environmental ideas. "Figuring out what kind of computer programs can help solve environmental problems or societal problems" is what he would like to pursue in the future. According to Jason, "it has to apply to people. At some point, it has to help or affect people. That's kind of my personal thing. Whatever you're doing, it doesn't matter if it doesn't help people in some way, if it doesn't benefit people."

Vignette: Mark

Mark grew up in England and came to the United States where he received a master's degree in Geographic Information Systems. While studying he began taking programming classes to enable him to construct digital maps. After graduation he has worked as a computer programmer, and frequently publishes articles concerning geography and technology.

Growing up in England, Mark considered himself a good student: "I was one of the better students in my year at high school." At age 16 most students narrow in on only two or three subjects as they continue their secondary studies and, "I did physics, math, and geography. Bad choice, because English, history, and geography would have been my easier subjects, ones that came naturally to me." He felt most successful in geography and, after retaking both math and physics classes, he completed "a physical geography degree which was very much focused on quantitative science. Most of my focus was on meteorology."

Mark contributes his struggle with the quantitative world with "the way I was wired." He generally admires excellent writers, and enjoys reading well-written books. "I pick up on that stuff, and a lot of other people may not pick up on that stuff [beautifully written works] . . . it just resonates with me. The quantitative world may have seemed less instinctive for Mark, but "I find [it] fascinating. The whole world of physics, when I suddenly looked at this world of physics and got it, I thought, I can answer all of these questions that I've always wondered about. It answers so many real questions. It's not this ephemeral, sort of wishy-washy thing. It's really solid. It gives you something."

His interest in the physical world gave him a tangible use for math. Before studying physics he approached math with the question, "What's the point? I've worked through all of these equations and I get that bloody number at the end of it all, well, what's the point?" He began to view math as a tool to discover actual concrete answers about the world. "If you say, you drop a ball off a building and you could do all of these calculations and you get an exact number from something, that's something you can visualize and apply to a real thing." To Mark "math is more the pure side of things, and physics the applied side of thing." With physics math became more real and less abstract. "I can say that car was going that fast and got there at this certain amount of time. It was real. So, that's what I liked about it, but it wasn't like going for a walk for me."

After finishing university in England, Mark had a variety of jobs, but "really never alighted on anything that was really anything more than just fluff." Moving to the US did not immediately provide a career path either, and he was about to complete his premed prerequisites when he realized, "I didn't really have passion for it [medicine]." After conversing with his advisor, Mark decided to go back to geography and was introduced to the GIS program. About

that time, the Internet was becoming more widely used, and the possibility of combining geography and technology was intriguing to Mark. "I remember I sat down thinking you could have maps on the Internet. You could actually do all of these cool things with maps. I formulated this idea in 1996, which is ten years before Google ever came out with their Google Maps." The new possibilities led Mark to computer programming.

At the time Mark was studying at the university, programming classes were only offered to students studying computer science. After disagreeing with faculty about the need of computer skills for geography students, "I said, I'm going in to the Computer Science department and I'm going to take some no-credit classes up there, and just see if I can program. I had no idea what programming was about, I had not a clue." He began by taking a two-part C++ class. "It destroyed me. It was so difficult, I had no idea what I was doing, I just didn't know what I was doing." Later he took a class in Visual Basic, and after that, "most of my [programming] work has been on my own."

In 1997 the Economic and Social Research Institute (ESRI) came out with a software program to help users create maps. Being on the pioneering edge of digital mapping got Mark noticed. "The forest service got wind of what I was doing, and they paid me to do a piece of work for them which ended up being my master's degree. So it was a really good scenario." After graduation Mark worked as a programmer, and then was hired again by the forest service to make more maps. Even though he has been working with computers ever since, "I never regarded myself as a computer programmer, ever. Ever, ever, ever, and I never will. I'm not a computer programmer. I am an upstart who is interested into the world of computer science . . . I can get by, I can do it, I love doing it, but, is it like walking down the street? It is not like

walking down the street. It's like me when I did physics; I like it. It's tangible, it's real, I get something out of the end of it. But it doesn't come as natural to me."

Mark has always viewed programming as an exercise in logic:

> I try to describe it to my wife and I said it's like playing chess. You try to look moves ahead and think like, if I do that, and I do that, and I do that, and I do this, what's that going to end up with? Huh. That. So, I, I kind of regarded it as a logic exercise and yes, I think being a programmer I think in a more logical way.

The more Mark learned about programming, the more it was like physics in that "it's very tangible."

Mark's unique experience and skills have allowed him to write for several online and printed magazines including: *Directions Magazine*, *Positions Magazine*, *GIS World*, and *Business Locations*. He has noticed that often people who are very technical have a hard time explaining computer concepts to other people. "There's a disconnect between a programmer and a business person, or someone who doesn't live in that world, and I'm always interested by the connection between the two." In the future Mark sees himself continuing "writing articles about technical things, explaining technical things."

Vignette: Andrew

Andrew received a bachelor's degree in electronic information technology. He currently programs web applications that allow other employees to improve efficiency in their work.

Andrew grew up in California and Alberta, Canada, and performed well academically. "I believe my grade point average was pretty good, so I think the general education seemed pretty easy to me." The classes he did the best in were "sciences and probably math class." He had varied hobbies and interests, and began programming in his early twenties.

For Andrew, learning to program occurred in a formal, college setting, and "the first language was Java, so it started off with object-oriented programming." While studying electronics he continued taking programming classes and gained experience building websites. What kept him programming was "the challenge. I always like to think that a person would like programming if they . . . will spend hours trying to do something only to get a single word to print on a screen, and they think they've accomplished the world. But, it's the thought of working really hard and then seeing something happen, right there, right when you complete it."

While taking computer science classes as part of his major, Andrew had concurrent math classes. "The basics in what computers and programming is, is just math, addition of *1*'s and *0*'s. You're adding, you're subtracting; the concepts are closely tied together." Andrew considered himself a proficient mathematician, "but I don't know if anybody really just assumes I am based on being a programmer." Since finishing his degree in electronic information technology he hasn't yet faced a mathematical challenge that has impeded his programming. "I've found most of the math that I do in programming is basic; it's addition, subtraction, multiplication, and division … I feel like I'm good at math but since getting out of college, I don't know if I've really had to use math other than the basics."

His present work has not yet required Andrew to solve higher-level math problems but there are instances where he thinks more complex math would be necessary. "There are things out there application wise, that have to do higher math, for nuclear reactions, for instance. I'm sure programmers for that would have to have a higher understanding of math to get their software to work correctly and be accurate." There is a chance for Andrew that more complicated math soon will play a larger role. "Actually with the one goal I'm doing, yes. Math

will probably be a little bit higher as I will actually try to get work in combining programming with higher functionality of science and data manipulation."

Andrew projects that in the future "I'll continue on programming." He also plans on going back to school in order to "further help with my company's goals."

Vignette: Scott

Scott feels two people influenced him along his career path: his father who was an engineer and his uncle who worked for a university where Scott spent time using computers. Later he attended university studying zoology, computer science, and business. He presently resides in Utah and works in the technology department of an online language education company.

Scott enjoyed learning, especially when it involved problem solving. "I'm the son of an engineer and so all growing up I was trained to solve problems. So everything that I would encounter I would look at it and I would try and understand what's happening and then try and fix it, finish it, improve it." In high school "chemistry and biology were probably my most favorite subjects." He despised busy work, and enjoyed solving problems efficiently. Scott did well in math but decided to try "sort of a social experiment my senior year of high school, where in our BC calculus class, I wouldn't even do the majority of the homework, and then I would get like higher grades in the class on the exams, and try to show my teacher that I didn't need to do the homework to be successful."

Programming was introduced to Scott through his uncle who worked at a university "which had some of the first desktop computers in the state. We would go up there to visit him and we would see the computers, and that was my first exposure to computing or personal computers." Scott's family bought a home computer shortly after. "We hooked it up to a

monochrome television, and I used it mostly for playing Atari-style games on it." Soon he wasn't just playing games on it, he was also programming them. "My uncle brought over some programming magazines, and they were BASIC. They had programs in it that you could program into it [the computer]." Unlike the math homework he refused to do, Scott thought that "programming is the useful application of the math . . . it's not busy work; it's not wasting my time."

By the time he was actually able to take a programming class in high school "it was too simple for my understanding, but it was a lot of time in front of a computer, and my first formal exposure to programming." The Internet provided Scott with even more to learn. "With the Internet, what made it that much more exciting for me was how immediate the results were. There was just this wealth of information and you could learn and make changes and see it happening right away. That's where it really started getting interesting, when I started building web pages and then the natural progression from static html to dynamic scripting languages, and then dabbling like in Perl and like the early PHP."

While programming Scott continued to take math classes. "I think I would describe being good at mathematics as understanding that things can be solved mathematically. And then having enough exposure to the different aspects of mathematics that when you're presented with a situation, you know which area to go to solve the problem." According to Scott, good mathematicians are not necessarily the ones who use math every day, "but when they are presented with a situation, they can identify it as, this is a mathematical problem, and these are the tools that I know exist to solve it." Scott has sometimes had to acquire higher math knowledge to make his programs better. This occurs "where you're trying to solve something

programmatically, and you know that it can be solved mathematically. There's often a faster way to solve this problem, if you had better math."

Understanding programming shed new light on mathematical concepts. "Exposure to binary operations, [is] a whole field of mathematics that I definitely would have never understood without being involved with programming and computer hardware." Recently he has spent considerable time with his fourteen-year-old brother-in-law studying both math and computers. "We sat down and taught him how to do binary addition, and so all of that's coming out of a programming perspective of, we're trying to learn about programming and then we're learning new things about math that go into the language to make it happen." Relating math to programming has changed his brother's-in-law attitude toward math. "It's a lot more interesting than just an exercise to like calculate areas of spheres. He's writing a program, and this is what I explained to him, I said look, you don't actually have to know and remember the math, or do the math to calculate every sphere, but you have to understand it well enough to write a program to do it for you. That's how we're leveraging the power of computers and programming is you don't have to then remember the math. You just know that it's there. You can look at it and figure out how it's working, but you don't ever have to do it." Scott thinks programming is "connecting the dots" to real life, and explained, "programming has connected the math to real life, where as before it was fairly abstract. Even when it wasn't abstract, for whatever reason, it just didn't seem as apparent or as necessary."

Scott's experience with both programming and math influence him often in his profession. "I'm sure that it [programming] made me more systematic, and possibly more creative, understanding that there's more than one way to solve a problem or to do different things than I would, I'm sure that it affected me, the way that I thought about it, the way that I

attacked it." At work as a programmer Scott is "happy to do the math if I'm writing a program that's going to save me work down the line. I'm not happy to do that same math when I have to do it over and over and over on a series of problems that don't have anything to do with anything. They're just taking up my time." In the future Scott would "like to build a business that leverages technology either just in the information or Internet space, or possibly in the medical field." He wouldn't want to do anything "that doesn't make the world a better place or allow people to do things more efficiently." He is sure that math will play a part in his future because, according to Scott, "it's all math."

Discussion

This section examines themes noted across participants, as identified in their individual vignettes. The limitations and implications of the findings are subsequently presented.

Identifying Themes

While the individual stories told by each participant provide insight into their personal experiences with mathematics and computer programming, four distinct themes emerged when looking across participants. Namely, (a) programming provided concrete examples of abstract mathematics, (b) programming provided the "why" to the mathematic "how", (c) programming helped participants learn to break tasks into sub-tasks, and (d) programming provided a context wherein necessity removed obstruction.

Programming provided concrete examples of abstract mathematics. Giménez (2004) stated, "The biggest struggle in mathematics is about meaning and significance, which is not revealed by the teaching practice. Why is mathematics to be learned and taught? Why is math applicable?" (p. 145). Participants shared the viewpoint that programming provided concrete examples of otherwise abstract concepts. Ben recounted, "In the past when I was dealing with

logarithms, I didn't quite understand why they were necessary, and then from programming, it's like, that's a good use of them." While trying to calculate the execution time of a function he discovered the implications of logarithms and consequently "had to go back and learn exactly what a logarithm was . . . When you're doing an algorithm and you want to make it as fast as possible, you're using mathematical concepts to increase in speed." It was not until Ben had a concrete application of logarithms that the value and practicality of this math concept became apparent. To Ben, programming became "more like a useful tool for math" and concepts were "easier to remember, because I had an actual use for it."

This contextualizing of seemingly difficult mathematics facilitated a new landscape of rich pedagogy for the participants. Papert (1972) explains, "Most children never see the point of the formal use of language. They certainly never have the experience of inventing original formalisms adapted to a particular task. Yet anyone who works with a computer does this all the time" (p. 246). Brian recounted an experience where he had to use trigonometry to create an accurate clock animation explaining, "I was creating a clock animation in Flex . . . but I actually wanted to draw it to the stage, and to do that I had to know the exact mathematical coordinates of where the clock would be placed. And since the clock was refreshing at thirty frames per second, that meant I needed to refresh my angle, and my trajectory, and everything else thirty times per second, three hundred and sixty times a minute, and it [trigonometry] became really important." In addition to the trigonometry, working in a 3-D environment also required mathematics for "drawing glass buttons, and doing things in 3-D space and trying to make them look spherical." The practicality of this exercise influenced Brian's attitude toward trigonometry: "I think that's what it [relevant application] does and when people go to school for engineering or something, I think the engineering aspect of things solidifies the mathematical

concepts. It makes them actually make sense instead of just being theories." Brian concluded that programming "just helps math be more realistic."

There are many specific concepts that schools have mandated as important to learn yet suffer from any sort of obvious practical application. In early math classes, examples such as buying apples and oranges at the store, train arrival times, and calculating distances between cities are easy and intuitive for students to relate back to the real world. As students enter high school, however, the math becomes increasingly difficult to create such practical analogies. Baker and O'Neil (1994) describe the "problem with traditional word problems is related to the fact that many students do not see them as real. The problems often seem contrived and arbitrary and have nonrealistic goals" (p. 202). Programming exercises such as calculating sorting algorithms provided Asim with what he felt was an appropriate vehicle to better clarify abstract calculus concepts: "Definitely computational time. Calculus really comes in handy when this is converging to this, especially for something like sorting algorithms and everything."

Certain areas of math, such as alternate base systems, were considered by participants to be otherwise unapproachable or incoherent. Scott explains that "exposure to binary operations [is] a whole field of mathematics that I definitely would have never understood without being involved with programming and computer hardware." Ben supports this sentiment stating, "Especially with converting from whatever base you're working in, if it's a hexadecimal or a base 2. That is one thing [in programming] that's definitely helped me to understand that [math] concept better."

It is not necessarily important that programming had a direct impact on the difficulty of performing the mathematical operations but rather that it provided a context for the participants to justify learning the concept. Brian said, "I always thought that geometry and trig were kind of

worthless, because I never planned on being an architect, or an engineer, that would actually need to use them. But once I jumped into programming and had to start using those all the time, I started realizing that it was actually an extremely useful practice to know those things, and I wish that I'd learned them in the first place." Scott summarized this sentence when he positioned programming as "connecting the dots" to how math applies to real life. He explained, "Programming has connected the math to real life, where as before it was fairly abstract. Even when it wasn't abstract, for whatever reason, it just didn't seem as apparent or as necessary."

Programming provided the "why" to the mathematic "how". In an earlier example it was necessary for Ben to go back and relearn the concept of logarithms despite previous study in high school math classes. Without a meaningful application of logarithms there was never an anchor of understanding for Ben to draw on years later. Once the link between concept and application had been created through this programming case, Ben stated "it [logarithms] was easier to remember, because I had an actual use for it."

The National Council of Teachers of Mathematics (1989) state, "students need to view themselves as capable of using their growing mathematical knowledge to make sense of new problem situations in the world around them" (p. ix.). It is important to distinguish the difference between one's capacity to perform a particular mathematical procedure and being able to recognize a situation where that calculation would be needed. In other words, there is a distinction made between application and execution. In many cases, simply knowing the "why" is more important than recalling the procedural "how." If one knows what math is needed in a given situation, then the specific process can always be obtained from another source or even calculated by alternative means.

Scott illustrates this idea with an experience where he and his brother-in-law were working through a problem set and needed to calculate the area of a sphere. He explained to his brother-in-law that "You don't actually have to know and remember the math, or do the math to calculate every sphere, but you have to understand it well enough to write a program to do it for you." While Scott's brother-in-law used computer programming to perform the actual mathematic calculations he had obviously been able to identify the mathematical concept needed to overcome the problem. Steffe (1991) underscores this effect, explaining the purpose of "problem solving is not simply the solution of specific problems. The primary reason is to encourage the interiorization and reorganization of the involved schemes as a result of the activity. These interiorized and reorganized schemes constitute operative mathematical concepts that are constructed by means of reflective abstraction" (p. 187).

Scott continued, "We're leveraging the power of computers and programming so you don't have to then remember the math. You just know that it's there. You can look at it and figure out how it's working, but you don't ever have to do it." The significance of this experience is not to say that programming should be thought of as a substitute for learning math, but rather that this experience helped provide an anchor for a particular concept. The programming concept facilitated an environment where Scott was able to solve a problem using a mathematical concept. Papert (1972) proposed this idea of "creating an environment in which the child will become highly involved in experiences of a kind to provide rich soil for the growth of intuitions and concepts for dealing with thinking, learning, playing, and so on" (p. 247).

Asim shares a similar story where he needed to solve a problem in a statistics class. He stated, "I did not want to sit there, I just needed the answer, I just had to fill in the answer, and the error percentage was like five percent or something, so you could be off by five percent, it

didn't matter, and, the ideal way to do it was using calculus . . . and I did not want to sit there and do the calculus . . . When I did my calculator stuff in intervals of 1 and just started adding it, but when you go down to like intervals of 0.5, the errors get smaller, 0.05, it gets smaller still. And, this is where the programming skills come in handy, you write a program to do it in intervals of 0.0000001, and then start adding, it gets very, very close to the correct answer." As a result of the programming, he actually ended up with a much more accurate answer than had he simply performed the required calculus.

Because he had substituted programming in place of a mathematical procedure, Asim felt that he had used a deceitful shortcut: "That was cheating, kind of. But, yeah, I did get the very correct answer." While it is certainly possible that the class required him to know the correct calculus procedure, in more practical applications, this sense of cheating the math could be viewed as equally valid as any other method. In everyday scenarios, the results are often more important than the method used. Furthermore, the end result of cheating the math could be compared to Olkin and Schoenfeld's (1994) research finding that "The joy of confronting a novel situation and trying to make sense of it - the joy of banging your head against a mathematical wall, and then discovering that there may be ways of either going around or over that wall" (p. 43).

Jason expressed this same sentiment toward using programming to get around a limitation of his math knowledge: "With programming, I don't need to have a good memory, I have references I can just quickly access the information that I need to solve the problems, so that is no longer a limitation for me . . . I think of programming as an alternative to math. I can't solve this using normal math but I can write a program that will solve it for me." Jason's use of programming is only an alternative for the procedure, not the understanding of the mathematical

application. Jason feels that programming actually enables him to make use of math that would otherwise not be at his disposal stating, "programming allows me to use math to help me do what I want to do. I think without it, I'd have less patience for it, and I couldn't even see the purpose in it . . . I use programming as a tool to help me to use, to kind of harness the power of the math."

Programming helped participants learn to break tasks into sub-tasks. The game of chess was used to illustrate how programming helps participants consider the effects and possibilities of individual steps in a problem. Brian explained, "It's like playing chess for your first couple times, you don't think about every possibility that could happen down the road, you just think about your current move. But as you play, you start realizing that if you make this move then this could possibly happen and this could possibly happen and all these things could happen, and your thought to make a single move increases . . . and I feel like that's what programming does. It basically just creates a habit of always thinking about what this one choice is going to do, and how it's going to affect every other aspect of the program." Mark, also supported the chess analogy, "It's like playing chess. You try to look moves ahead and think like, if I do that, and I do that, and I do that, and I do this, what's that going to end up with? Huh. That. So, I kind of regarded it as a logic exercise and yes, I think being a programmer I think in a more logical way."

Programming applications generally consists of breaking down complex problems into smaller chunks of subtasks. The programmer then sequences a series of step-by-step instructions to create solutions to these more focused steps. With each step, the programmer is forced to anticipate the consequences of every decision for the individual task and on the problem as a whole. This breaking-down approach was reported to influence general problem solving habits

as almost any task could be divided into smaller subtasks. Asim stated, "Most importantly, it [programming] taught me how to dissect things into smaller chunks."

When applied to math, this idea of breaking problems into smaller, logical increments becomes equally valuable. In mathematics, Nunokawa (2001) stated, "Identifying appropriate subgoals can be helpful for solving complicated problems" (p. 187). Participants reported taking a more systematic approach to mathematics after learning to program. Asim explained that "when you're doing some kind of mathematical algorithm, that [breaking it into increments] is how you approach it." This process of dividing the problem into subtasks reinforces the mathematical ideas: "The solver's generation of subgoals and his understanding of the problem situation was intertwined with each other. His settlement of subgoals was supported by his understanding of the situation" (Nunokawa, 2001, p. 203).

This logical strategy helped reduce the barriers of more complex problems, allowing participants to create a more approachable path between the problem and the solution. Catrambone (1998) states, "it is usually predicted that a learner possessing an appropriate subgoal, or set of subgoals, will be in a better position to solve new problems compared to learners who memorized only a set of steps" (p. 357). Asim explained, "But definitely before that [learning to program] . . . when I approached math I never thought of dissecting the problems into smaller chunks. 'Ok, let's derive this first and see where we can go from there.' It was never like that. It was always 'how do I get from point A to B?' You really don't think between point A to B there would be A to C, C to D, D to E, E to B . . . It's actually usually that way when you're solving math problems. . . . I think programming taught me how to go to those intermediate points and derive stuff. . . . I think that helped me."

Brian described a similar connection, “I think the relationship is that they [math and programming] are both quantitative. With math you’re trying to come to a solution for a problem, and the route you take to get there is the puzzle or the mystery. That’s exactly what you do with programming. You have an end that you’re trying to get to in a problem, and a barrier in between, and how you get there is basically the puzzle that helps you get down that path.”

Participants felt these problem-solving skills extended beyond mathematics. Ben reported, “I definitely take a divide and conquer approach in most everything I do . . . I would say it [the divide and conquer approach] leaked over into the rest of my life, and I started doing other things that way as well.” Scott explained what he saw as programming’s broader influence on his life saying, “I’m sure that it [programming] made me more systematic, and possibly more creative, understanding that there’s more than one way to solve a problem or to do different things than I would, I’m sure that it affected me, the way that I thought about it, the way that I attacked it.”

Programming provided a context wherein necessity removed obstruction. While programming, participants often needed to actively seek mathematical instruction to solve a programming obstacle. Asim expressed how “It [programming] does force you to learn things [in math].” Scott reported how there were times "where you’re trying to solve something programmatically, and you know that it can be solved mathematically. There’s often a faster way to solve this problem, if you had better math.”

In some cases, this resulted in the individual approaching their math teachers for direction. Jason explains, “I’d have programming problems that I didn’t have the math yet in my mind to solve, and so I’d go to the next grade math teacher at my school and say, ‘How do I

solve this problem for this program?' I was making one of these brick games, that you bash the bricks with a paddle, like Breakout. I didn't really have the math to understand the angles everywhere, and so I'd go to these teachers and have them help me solve these math problems." Asim describes a situation where he needed assistance while programming a solution for a problem in bioinformatics "because the algorithm was dependent on math, and I did not have the background for it. I had to go and ask my professor for it, to teach me how to do the math behind those things. He taught me and then I could do it."

One could imagine a scenario where two different students find themselves confronted with a new concept in mathematics. The first student is being exposed to this new concept in a typical math class and, without a practical application to relate to, may find the ideas too abstract, difficult or even unnecessary. The student is learning the math simply to fulfill a school requirement, which does little to impart a sense of importance to the ideas or minimize anxiety and may result in avoiding situations which require mathematics (Ashcraft, 2002, p. 181). A second student however, finds himself in a situation where he has been trying to solve a programmatic problem and now recognizes the need to learn this new concept in order to move the project forward. As the necessity for understanding this concept developed organically, there is already a natural sense of relevance surrounding the required mathematics and its application. The student therefore approaches the learning process without trepidation as "what seemed most abstract and distant from the real world turns into concrete instruments familiarly employed to achieve personal goals" (Papert, 1972).

For this second student, the necessity of needing to solve an existing problem provided motivation and removed any obstructions to learning seemingly difficult mathematics. Papert (1971) described this outcome when using math to achieve specific goals in a larger project:

> To achieve these goals mathematical principles are needed; conversely in this context mathematical principles become sources of power, thereby acquiring meaning for large categories of students who fail to see any point or pleasure in bookish math and who, under prevailing school conditions, simply drop out by labeling themselves "not mathematically minded." (p. 4)

Brian explained how he is frequently required to learn new math concepts saying "most of the math that I do now is all calculating things within a 3-D space and I never took trig, so anything that has to do with sine or cosine, or any of that type of thing, I've had to figure out on my own." If this same information had been presented to him in a classroom setting he "probably wouldn't have been interested at all. I feel like a practical application of math always changes my attitude of it."

Several participants echoed this sentiment that programming improved their attitudes toward mathematics. Ben said, "It [learning how to program] probably bettered my attitude towards math, and that I had something to use it for." For Ben, programming allowed him to "see things from a new perspective." Similarly, Jason stated where he "would be without programming" explaining how, "In terms of my math . . . I'd probably hate it more." Jason adds, "It's [math is] really abstract and seeing the programming, putting it to use, helps me see it. . . . I think without the programming, I would be lost with the math. I would be more confused about it. Whereas with the programming it's helped me put it to use effectively." Scott simply states, "I'm happy to do the math if I'm writing a program that's going to save me work down the line."

Asim believes his attitude changed because knowing programming "made math much more interesting." Specifically, learning to program helped Asim view math as simply another set of tools, allowing him to get past any intimidating barriers. He explains, "It [solving math

problems] is like programming. In programming you just have this set of tools that you know and you're given this problem that you've never seen before and you solve it. . . . When you start getting into programming, there are so few rules, and everything else you make up. Same as in math, there are few rules, that you can apply it by however you feel like. Those are the two areas where you can get exposed to that kind of thinking early on: 'ok I have these set of tools, how do I get to this?'" Discussing problem solving and mathematics, Suydam (1987) similarly explained, "If problem solving is treated as 'apply the procedure,' then the students try to follow the rules in subsequent problems. If you teach problem solving as an approach, where you must think and can apply anything that works, then students are likely to be less rigid" (p. 104). Asim felt the freedom to use tools "helped me to do math. It makes you kind of less scared of math. People are scared of math."

Implications

This research suggests that programming increases participants' utilitarian view of mathematics, and by extension, their attitudes. Existing research in mathematics discusses similar issues with "the bridging process between everyday practices and school mathematics" (Presmeg, 2002, p. 295). Dossey (1992) made the following suggestion:

> A philosophy [of mathematics] should call for experiences that help mathematician, teacher, and student to experience the invention of mathematics. It should call for experiences that allow for the mathematization, or modeling, of ideas and events. (p. 42)

Programming provided a rich environment where difficult and easily forgotten mathematical concepts could be explored, reinforced, and appreciated. Studies such as Moses and Cobb (2001), advocate project-based approaches to mathematics instruction, "using a version of experiential learning" where "each step is designed to help students bridge the transition from

real-life to mathematical language and operations" (p. 119). Considering the positive experiences of participants in this study, further study on utilizing programming projects to facilitate these experiences and help overcome contextual obstacles should be investigated.

Improved attitudes, increased motivation, and decreased anxiety were all reportedly beneficial effects of programming. Wilkins and Ma (2003) stated, "a person's mathematical disposition related to her or his beliefs about and attitude toward mathematics may be as important as content knowledge for making informed decisions in terms of willingness to use this knowledge in everyday life" (p. 52). While not typically considered standard instruction, future studies should explore adding computer programming to school curriculum, potentially leading to improved mathematics understanding, self-efficacy, and decreased anxiety. In contrast to the typical American experience, growing up in Nepal, Asim felt his early exposure to programming in school increased his attitude and approach to problems in concurrent math classes. In a similar vein, Hembree (1990) warned of ignoring the consequences of mathematics anxiety:

> Despite its lack of independent identity, the research of mathematics anxiety has prospered, spurred by increasing perceptions that the construct threatens both achievement and participation in mathematics. These suggestions have national import; when otherwise capable students avoid the study of mathematics, their options regarding careers are reduced, eroding the country's resource base in science and technology. (p. 34)

While the participants were able to provide many examples of mathematics' application and experience, none would describe themselves as mathematicians. In Mark's extreme case, while he identified himself as a programmer and problem-solver, he explicitly distanced himself from

mathematics. This suggests that a possible misunderstanding of what it means to be a mathematician is being perpetuated either academically or culturally. Dossey (1992) explained,

> The conception of mathematics held by the teacher may have a great deal to do with the way in which mathematics is characterized in classroom teaching. The subtle messages communicated to children about mathematics and its nature may, in turn, affect the way they grow to view mathematics and its role in their world. (p. 42)

There may be failure among students and professionals to recognize certain types of problem solving as mathematics, perhaps confusing the purely calculative side of math with the intent or application. Studies suggest children typically hold a fundamentally narrow viewpoint that mathematics is primarily computation and arithmetic (Fuson, Kalchman, & Bransford 2005; Grootenboer, 2003; Young-Loveridge, Taylor, Sharma & Hawera, 2006). Further research should be done to investigate the way mathematics is presented or defined from an academic, cultural, and practical sense. Young-Loveridge (2006) stated,

> Given that mathematics is part of the core curriculum, we think it could be important for teachers to help children engage with ideas about the nature of mathematics. In preparation for such discussions, teachers might benefit from examining their own beliefs about what mathematics is, and reflecting on the subtle messages they might convey to students about the nature of mathematics. (p. 589)

Conclusion

This study revealed the potential effects learning to program could have on one's perception and application of mathematics. The results of the research pointed to four specific areas of interest. First, learning to program provided context for many abstract concepts. Second, programming illustrated the important distinction between understanding application of

math in a specific situation and the execution of a known procedure. Third, programming habits helped participants divide complex problems into more manageable tasks. Finally, the necessity of solving a programming problem provided motivation and eliminated apprehension toward mathematics.

References

Agalianos, A., Noss, R., & Whitty, G. (2001). Logo in mainstream schools: The struggle over the soul of an educational innovation. *British Journal of Sociology of Education, 22*(4), 479-500.

Ashcraft, M. H. (2002). Math anxiety: Personal, educational, and cognitive consequences. *Current Directions in Psychological Science, 11*(5), 181-185. doi:10.1111/1467-8721.00196

Baker, E. L., & O'Neil, H. F. (1994). *Technology assessment in education and training.* Hillsdale, NJ: Psychology Press.

Bergin, S., & Reilly, R. (2006). Predicting introductory programming performance: A multi-institutional multivariate study. *Computer Science Education, 16*(4), 303-323.

Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. *SIGCSE Bulletin*, 33(3), 49-52. doi:10.1145/507758.377467

Catrambone, R. (1998). The subgoal learning model: Creating better examples so that students can solve novel problems. *Journal of Experimental Psychology, General, 127*(4), 355–376.

Clements, D. H., Battista, M. T., & Sarama, J. (2001). Logo and Geometry. *Journal for Research in Mathematics Education. Monograph*, *10*, i-177. doi:10.2307/749924

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology, 76*(6), 1051-1058.

Clements, D. H., & Nastasi, B. K. (1988). Social and cognitive interactions in educational computer environments. *American Educational Research Journal, 25*(1), 87–106.

Crotty, M. (1998). *The foundations of social research: Meaning and perspective in the*

research process. London, England: Sage Publications.

de Corte, E., Verschaffel, L., & Schrooten, H. (1992). Cognitive effects of learning to program in logo: A one-year study with sixth graders. In E. de Corte, M. C. Linn, H. Mandl, & L. Verschaffel (Eds.), *NATO ASI Series. Series F: Computer and Systems Sciences: Vol. 84. Computer-Based learning environments and problem solving.* (pp. 207-28). New York: Springer-Verlag.

Dossey, J. (1992). The nature of mathematics: Its role and its influence. In D. A. Grouws (Ed.), *Handbook of research on mathematics teaching and learning* (pp. 39-48). New York: Macmillan.

Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Solomon, C. (1970). Programming-languages as a conceptual framework for teaching mathematics. *SIGCUE Outlook*, 4(2), 13-17. doi:10.1145/965754.965757

Flawn, T. (2008). *Foundations for success: The final report of the National Mathematics Advisory Panel.* Washington, DC: U.S. Department of Education.

Fuson, K. C., Kalchman, M. & Bransford, J. D. (2005). Mathematics understanding: An introduction. In M. S. Donovan & J. D. Bransford (Eds.), *How Students Learn Mathematics in the Classroom* (pp. 217-256). Washington, DC: National Academies Press.

Gorman, H., & Bourne, L. E. (1983). Learning to think by learning logo: Rule learning in third-grade computer programmers. *Bulletin of the Psychonomic Society, 21*(3), 165-67.

Grootenboer, P. J. (2003). The affective views of primary school children. In N. A. Pateman, B. J. Doughtery, & J. Zillioz (Eds.), *Navigating between Theory and Practice* (Proceedings of the 27th conference of the International Group for the Psychology of Mathematics Education, Vol. 3, pp. 1-8). Honolulu: University of Hawai'i.

Hembree, R. (1990). The nature, effects, and relief of mathematics anxiety. *Journal for Research In Mathematics Education, 21*(1), 33-46.

Keitel, C. (2004). Students' struggle for sense making. In J. Gimenez, G. E. Fitzsimons, & C. Hahn (Eds.), *A challenge for mathematics education: To reconcile commonalities and differences.* (pp. 142-149). Barcelona, Spain: GRAO.

Konvalina, J., Wileman, S. A., & Stephens, L. J. (1983). *Math proficiency: A key to success for computer science students.* Commun. ACM, 26(5), 377-382. doi:10.1145/69586.358140

Leeper, R. R., & Silver, J. L. (1982). *Predicting success in a first programming course.* (pp. 147-150). Indianapolis, Indiana: ACM. doi:10.1145/800066.801357

Linn, M. C. (1985). The cognitive consequences of programming instruction in classrooms. *Educational Researcher, 14*(5), 14-29.

Mitchell, R. (1979). *Less than words can say: The underground grammarian* (1st ed.). Boston, MA: Little, Brown.

Moses, B., & Cobb, C. (2001). *Radical equations: Math literacy and civil rights.* Boston, MA: Beacon.

Moustakas, C. (1994). *Phenomenological research methods* (1st ed.). Thousand Oaks, CA: Sage Publications.

National Council of Teachers of Mathematics (1989). *Curriculum and evaluation standards for school mathematics*. Reston, VA: Author.

Noss, R. (1986). Constructing a conceptual framework for elementary algebra through logo programming. *Educational Studies in Mathematics, 17*(4), 335-357.

Nunokawa, K. (2001). Interactions between subgoals and understanding of problem situations in mathematical problem solving. *The Journal of Mathematical Behavior, 20*(2), 187-205.

doi:10.1016/S0732-3123(01)00069-4

Olkin, I. & Schoenfeld, A. (1994). A discussion of Bruce Reznick's chapter. In A. Schoenfeld (Ed.), *Mathematical thinking and problem solving* (pp. 39-51). Hillsdale, NJ: Lawrence Erlbaum Associates.

Palumbo, D. B. (1990). Programming language/problem-solving research: A review of relevant issues. *Review of Educational Research, 60*(1), 65-89.

Papert. S. (1971). *Teaching children to be mathematicians vs. teaching about mathematics*. Cambridge, MA: MIT Artificial Intelligence Laboratory.

Papert, S. (1972). Teaching children thinking. *Innovations in Education & Training International, 9*(5), 245. doi:10.1080/1355800720090503

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Book.

Patton, M. Q. (2002). *Qualitative research & evaluation methods* (3rd ed.). Thousand Oaks, CA: Sage Publications.

Pea, R. D., & Kurland, D. M. (1987). On the cognitive effects of learning computer programming. *Mirrors of minds: patterns of experience in educational computing* (pp. 147-177). Ablex Publishing. Retrieved from http://portal.acm.org/citation.cfm?id=30914

Presmeg, N. (2002). Beliefs about the nature of mathematics in the bridging of everyday and school mathematical practices. In G. Leder, E. Pehkonen, & G. Torner (Eds.), *Beliefs: A hidden variable in mathematics education?* (pp. 293-312). Dordrecht, Netherlands: Kluwer.

Soloway, E., Lochhead, J., & Clement, J. (1982). Does computer programming enhance problem solving ability? Some positive evidence on algebra word problems. In R. J.

Seidel, R. E. Anderson, & B. Hunter (Eds.), *Computer literacy (*pp. 171-185). New York: Academic Press.

Steffe, L. P., & Wood, T. (Eds.). (1990). *Transforming children's mathematical education.* Hillsdale, NJ: Lawrence Erlbaum.

Subhi, T. (1999). The impact of LOGO on gifted children's achievement and creativity. *Journal of Computer Assisted Learning, 15*(2), 98-108. doi:10.1046/j.1365-2729.1999.152082.x

Suydam, M. (1987). Indications from research on problem solving. In F. R. Curcio (Ed.), *Teaching and learning: A problem solving focus*. Reston, VA: National Council of Teachers of Mathematics.

Tooke, D. (2001). Introduction. *Computers in the schools*, *17*(1), 1-7. doi:10.1300/J025v17n01_01

van Manen, M. (1996). *Researching lived experience: Human science for an action sensitive pedagogy*. Albany: State University of New York Press.

Wilkins, J. L. M., & Ma, X. (2003). Modeling change in student attitude toward and beliefs about mathematics. *The Journal of Educational Research, 97*(1), 52-63.

Young-Loveridge, J., Taylor, M., Sharma, S., & Hawera, N. (2006). Students' perspectives on the nature of mathematics. In P. Grootenboer, R. Zevenbergen, & M. Chinnappan (Eds.), *Identities, cultures and learning spaces* (Proceedings of the 29th annual conference of Mathematics Education Research Group of Australasia, Vol. 2, pp. 583- 590). Sydney, Australia: MERGA.

Appendix A: Survey

Programming: Math Experience Survey

Q1 Investigating the Perceived Long-term Effects of Learning to Program on Mathematic Ability

Purpose: This survey is being conducted by researchers at Brigham Young University to identify how and in what ways learning to program computer software may affect one's mathematical abilities. The survey is intended to take roughly 15-20 minutes to complete.

Confidentiality: Your answers will remain completely anonymous, unless otherwise indicated. You are not expected to experience any discomforts by completing this survey. However, participation in this study is completely voluntary. If at any time you wish to withdraw from the study, you may click on the "withdraw" button and your data will be erased. This research has met the Institutional Review Board requirements for human research and is listed under BYU IRB, E08-044.

Benefits: We anticipate that the results of this study will shed light on implicit or implied benefits that learning to program offers to mathematical understanding. You will not be compensated for completing this survey at this time. However, if you so indicate, we will send you the results of this study electronically.

By signing the following, you agree that you have read and understood the above consent and desire of your own free will to participate in this study

- ❍ I agree to participate. (1)
- ❍ I do not want to participate. (2)

Q28 Please enter your name in the following field

Q29 email

Q1 Age

- ❍ younger than 10 yrs. old (1)
- ❍ 10-14 (2)
- ❍ 15-19 (3)
- ❍ 20-24 (4)
- ❍ 25-29 (5)
- ❍ 30-34 (6)
- ❍ 35-39 (7)
- ❍ 40+ (8)

Q2 Gender
- Male (1)
- Female (2)

Q3 Current Occupation

Q4 Highest Degree Earned
- GED (1)
- High School Diploma (2)
- Associates Degree (3)
- Bachelors Degree (4)
- Masters Degree (5)
- Doctorate Degree (6)
- None OR Not applicable (7)

Q5 Do you have any trade or specialization certificates related to computers and or math - what are they?

Q6 Are you currently involved in computer programming? If not, how long has it been since you were actively programming for at least 2 hours a day?

Q7 How many hours per week do/did you spend programming?
- 0 (1)
- 1 - 2 (2)
- 3 - 4 (3)
- 5 - 6 (4)
- 7 - 8 (5)
- 9 - 10 (6)
- 10 - 12 (7)
- 13 - 15 (8)
- 16 - 20 (9)
- 20 - 25 (10)
- 26 - 30 (11)
- 31 - 35 (12)
- 36 - 40 (13)
- 40+ (14)

Q8 How many years of programming experience do you have in any of the following languages?

	None (1)	1 (2)	2 (3)	3 (4)	4 (5)	5 (6)	6 (7)	7 (8)	8 (9)	9 (10)	10 + (11)
Java (1)	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍
C/C++ (2)	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍
Visual BASIC (3)	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍
PHP/Perl (4)	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍
Pascal/Delphi (5)	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍
Assembly Language (6)	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍
JavaScript/ASP (7)	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍
ActionScript (Flash/Flex) (8)	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍
Other (specify) (9)	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍

Q9 How old were you when you began to program? (your age)

- ❍ 5 - 10 years old (1)
- ❍ 11 (2)
- ❍ 12 (3)
- ❍ 13 (4)
- ❍ 14 (5)
- ❍ 15 (6)
- ❍ 16 (7)
- ❍ 17 (8)
- ❍ 18 (9)
- ❍ 19 (10)
- ❍ 20 (11)
- ❍ 21 (12)
- ❍ 22 (13)
- ❍ 23 (14)
- ❍ 24 (15)
- ❍ 25 (16)
- ❍ 25+ (17)

Q10 How did you learn to program? (Click all that apply)
- ❑ Formal Classes (1)
- ❑ Picked it up from a book (2)
- ❑ Online forums (3)
- ❑ Video tutorials (4)
- ❑ A friend (5)
- ❑ Other (6) _____

Q11 What was your reason for learning to program?

Q22 Which math classes have you completed? (mark all that apply)
- ❑ Pre-Algebra (1)
- ❑ Algebra 1 (2)
- ❑ Algebra 2 (3)
- ❑ Advanced Algebra (4)
- ❑ Geometry (5)
- ❑ Trigonometry (6)
- ❑ Analytic Geometry (7)
- ❑ Calculus AB (8)
- ❑ Calculus BC (9)
- ❑ Other (10) _____

Q12 What math courses did you take while in High School?

Q13 What math courses have you taken since High School?

Q14 Describe how you felt about your math skills prior to learning to program.

Q15 Can you think of a specific instance in which something you learned in programming helped you comprehend a mathematical concept better? Please describe your experience.

Q16 What specific concepts do you feel are related between math and programming, if any?

Q17 In what ways has your knowledge of programming influenced your attitude toward math?

Q18 Were there any mathematical challenges you faced that stopped your programming progress? Please explain.

Q19 Would like to receive the results of this survey? (we will not sell or send your email to anyone; doing so would violate our University IRB approval.)
- ❍ yes (1)
- ❍ no (2)

Q20 Would you be willing to be interviewed for a research project?

- ❍ Yes (1)
- ❍ No (2)

Q21 Are there others that you know that you believe would be good candidates for this study? Please forward this survey to them (click here), or provide us their email(s) below (multiple entries separated by commas). Thank you for your time and help.

Appendix B: Interview Questions

Academics

Tell me about your general academic experience.
• How did you perform academically?
• What was your favorite subject?
• Can you explain what you liked about it that made it your favorite?
• What sorts of things would you do in that class that you liked?
• What was your least favorite subject?
• Can you explain what you disliked about it that made it your least favorite?
• What sorts of things did you struggle with academically?

Other Interests

Tell me about your other interests outside of academics.
• Hobbies?

Programming

How did you learn how to program?
• What materials did you use?
• Was it formal or informal (i.e., in a classroom, or on your own)?

Describe for me your first programming experience.
• How old were you?
• What caused you to begin to program?
• What were your goals?
• What did you create?
• Can I see that program, and can you explain to me how you made it?
• If this was not in a class, please describe for me what you first formally-taught programming experience was like.
• In what ways did this affect the way you began to think about problem solving *at that time?*

Now I'd like you to talk a little bit about your other programming experience.
• First, what caused you to continue to learn about computer programming?
• What programs did you learn? Why did you choose these programs?
• In what order did you learn these skills?

Mathematics

Tell me about your experience specifically in mathematics.
• What is mathematics, what does it mean to be good at mathematics?
• As a programmer, do people assume you are good in math?
• How did you feel about your math skills prior to learning to program?

• Were there any mathematical challenges you faced that stopped your programming progress?
• What math courses did you take at the same time as you were learning programming?
• Did you independently learn new math concepts outside of class?

Tell me about any experience where either math influenced the way you thought about programming or vice versa.
• If you had any such experience, will you share it with me?
• What was the problem you were faced with?
• How did understanding math help you solve this problem?
• OR How did understanding computer programming help you approach this math problem?
• In what ways has knowing how to program affected your understanding of math?
• Can you think of a specific instance in which something you learned in programming first helped you comprehend a mathematical concept when it was introduced?
• What specific concepts do you feel are related between math and programming?
• In what ways are they related?
• Do you have a personal experience that illustrates this relationship?
• In what ways has your knowledge of programming influenced your attitude toward math?
• What, specifically, do you find useful about math and programming?

Goals

Talk to me about where you are now.
• What are your professional goals?
• How has learning to program shaped these goals?
• What is the role of math in these goals?
• To what extent do you use mathematical concepts in your current programming (and vice versa)?

Appendix C: Codes

Problem solving {ps} How did you use what you learned (skills) in programming to solve problems outside of computer programming? How did your problem solving skills change?

Attitude math {am} Is math important to you? Did/do you care about it? Did/do you like it? What does it mean to be good at it? Do people assume you're good at math because you're a programmer? Do you assume that other programmers are good at math? How did your attitude towards math change as you learned to program?

Attitude academics {aa} Did you care about it? What were your favorite and least favorite subjects? What did you enjoy doing in class? What did you find boring or useless?

Attitude programming {ap} Did/do you care about it? Did/do you like it? What does it mean to be good at it? What do you assume about it or people who do it?
Self-Efficacy {se} How do you perceive your ability? What was hard? What was easy? Did you feel out of place? What did you feel you were good at?

Math affecting programming {map} How did math skills influence your programming? Did your lack of math knowledge cause you frustration? Did you have to learn new math skills to support programming?

Programming affecting math {pam} How did programming influence your math (skills and attitude)? When did math become more useful once you were programming?
Shared concepts {sc} What principles do both math and programming share? What particular parts of math were clearer once you started to program? What seemingly abstract math concepts made more sense as a programmer?

Motivation {mot} Why did you program? When did you need to learn concepts outside of class to further your programming? What do you see yourself doing in the future with programming? Do you think that math will play a part in your programming future?

Appendix D: Coded Interview: Asim

Neil: So, tell me about your general academic experience. How was, how was school for you?

Asim: {aa} I goofed off a lot in school, and, but I did get all my stuff done. {/aa} {se} I wasn't bad at it, I did pretty good. Programming classes were very easy for me. {/se}

Neil: So you took programming in, how old were you when you started programming?

Asim: When I, I was in, let's see, the eighth grade, so that's like, well no, that's 14, right?

Neil: So, 14?

Asim: Uh huh.

Neil: In school? So you started programming in school as a class.

Asim: Yeah.

Neil: Had you done anything before, at all, with programming?

Asim: Um, a little bit of GW Basic. But, no, wait, that is when I started programming, in the eighth grade. I used to use the computer, not programming wise though, just for general stuff.

Neil: So, eighth grade, already. Um, so, for the other subjects, what was your favorite subjects?

Asim: {aa} It was always science. {/aa}

Neil: Science, what was your least favorite?

Asim: Language classes. {aa} No wait, I liked English, I didn't like the Nepali classes. I hated the Nepali classes. {/aa}

Neil: Um, so you said you goofed around, but, did you, did you, so, but you did well? Um …

Asim: I mean, I always studied during the exams and everything so.
Neil: So what made science your favorite class? The science classes …

Asim: {aa} I don't know, it was just kind of natural, I just liked it. {/aa}

Neil: Just liked it?

Asim: Yeah.

Neil: Um, the ones that you didn't like, or even if you did like them, which ones were, were there any of them that you had to struggle a little more with, that were a little harder to do?

Asim: {aa} Well, I just wasn't interested in them so I just didn't put the effort to do them.

Neil: Interest, interest equals …

Asim: Interest definitely equals effort. {/aa}

Neil: Well, that's important. Um, so what did you do outside of school? What were your hobbies and stuff?

Asim: Um, played around a lot. I played cricket, which you don't see very often here. I played soccer. Um, just hung out with friends, that kind of stuff. I used to, whenever I could, I used to like, kind of travel. Travel as in, inside the city, go to different places, like visit historical sights mostly. I love visiting historical sights, in Nepal we have so much of those, it's crazy, like …

Neil: A little more than Provo …

Asim: Like six-seven hundred years old things. You could even find those if you really looked around. Um, and that kind of stuff and just the regular stuff.

Neil: So what, not always common that people start taking, is that, is that common? Did you choose to take the programming class?

Asim: {mot} No, I was forced to take it. {/mot}

Neil: You were forced to take it! Was is like, chemistry, biology, programming? You just had to take it, or was it part of …

Asim: You know what's called computer science? But, what they mostly focused on was something like Excel. Well no wait, it was Lotus 1, 2, 3 at the time. It was Lotus, and it was D Base, yeah. And it was, what was the other one? Word Perfect.

Neil: And so, when did they actually start getting you into code?

Asim: All at the same time.

Neil: Oh, all at the same time, and then it just, did it just carry through passed eighth ninth, tenth, eleventh, like? So you just kept on going with it?

Asim: {ap} Yeah, I, I really, I mean, it was kind of like almost after my first class itself, I really started liking it. {/ap}

Neil: Oh, so what about like college courses?

Asim: I have had, uh, the Intro Programming class, Data Structures, and what else? Discreet Structures, and, like, Enterprise Level Programming classes. {mot} Well, I had to do a lot of programming in my bioinformatics classes as well. That is my major. {/mot}

Neil: Oh.

Asim: {se} I'm not a computer scientist or anything like that. {/se} {mot} But, most of my programming knowledge, mostly it came from experience, because, from my, I mean, I did programming, it was kind of, I even made programs for people when I was in high school, like that they could use. And after coming, after coming to the US, in, from my sophomore year, I started programming for my part-time job, and most of the learning actually came from doing it myself, not from classes. {/mot} And obviously, making, optimizing everything, that comes from classes, but …

Neil: Um, as you learned programming, how do you think that affected problem solving?

Asim: {ps} Most importantly, it taught me how to dissect things into smaller chunks. {/ps}

Neil: Into smaller chunks?

Asim: Yeah.

Neil: Um … so, did they just start, when you started learning in, um eighth grade, was it, did they just start you with the basics and go? Or did, what was the first concepts, how did they first originally start teaching you?

Asim: I mean, it was like, they kind of just showed you how to print out stuff and they give you this block of code. Ok, so today we are going to make a circle. And they made the circle, right? And then, ok, let's move the circle. And they kind of started doing it that way.

Neil: So, visually.

Asim: Yeah.

Neil: So, something that you could have immediate feedback.

Asim: {ap} It was visual, yeah, it was always visual. And then ok, this does that, and at that time my favorite thing was moving the circles around in different paths. Ok, so let's see if I can do like straight across, down, like diagonally. {/ap}

Neil: So they didn't just start you with that typical variables, basic math, print back to score?

Asim: No.

Neil: Interesting. Um, so with math, so, what was your experience with math? Especially since you had been taking programming classes so long, obviously you were taking concurrent math

classes. Um what, um yeah, what was your experience, how did you, what did you think about math, in like high school?

Asim: {am} Math was, math was my second least favorite subject actually. I had absolutely no care for math. {se} I never, I, I tried getting out of homeworks as much as I could for math, and I was really bad at it. I mean, it's not that, I mean, I just didn't put the effort into it, and after, like around the ninth grade, we have this, we have this nation-wide exam, where you have to take at the end of the tenth grade, and you have to pass, you're expected to pass that with like with very good, with like flying colors. And for that, then in the ninth grade I started taking math more seriously. I actually started liking math, and then I got really good and I was one of the best in my classes, after that. {/se} {/am}

Neil: What was the change? What do you think was the change? Was it just interest? Or was there something else you were working on?

Asim: {am} No, there, no interest. {mot} Like, first it was because I just had to get that national exam, I had to do well on that national exam, that was the forcing factor, but as I started getting into it, I liked it more and more. {/mot} {/am} {pam} But definitely before that, I never, I don't think I, I mean, when I approached math I never thought of dissecting the problems into like smaller chunks. 'Ok, let's derive this first and see where we can go from there.' It was never like that. It was always like oh, 'how do I get from point A to B?' And you really don't think between point A to B there would be A to C, C to D, D to E, E to B. Like that kind of stuff. {sc} It's actually usually that way, right, when you're solving math problems, but when I thought of it, I was almost like, 'how do you get from A to B?' Like I would just, and then, at first it was like that, but when, but I think programming, like, taught me how to go into those like intermediate points and derive stuff. Ok, let's see where we can go from here, started doing that kind of thing. I think that helped me … {/sc} {/pam}

Neil: So, dissect.

Asim: Yeah.

Neil: Do you, do people assume that you're good at math because they know you program?

Asim: Yeah, usually, yeah.

Neil: So my …

Asim: {am} Yeah, I, I, I've had people come up to me, "Ok, do you think you know how to solve this?" And my friends, who I've never talked, I haven't discussed math or anything with them, I help them with programming but, yeah. It is, it is kind of stereotypical that people do think if you're good at programming you're good at math. {/am}

Neil: Um, what do you think it means to be good at math? If someone says that person's good at, like what do you think it means to be a good mathematician? What does that mean to you?

Asim: {am} I mean, having something thrown at your face, which you haven't seen before, and actually solving it. I mean, not something complex like, for example, I haven't seen what an *i* is, the imaginary number, but you have to solve a problem using that, but that, I don't know what that is. But, I mean, if you have the tool, tools for it and you're given this strange problem that you can solve using those tools, having the ability to solve that. {/am} {sc} I mean, now that I think about it, it is like programming. In programming you just have this set of tools that you know and you're given this problem that you've never seen before and you kind of solve it, right? {/sc}

Neil: Yeah, or you don't know maybe even what it, you're always solving a problem and you even don't actually know where you're going to end up. Sometimes you're just given the tools with no specific destination.

Asim: Yeah, yeah. Exactly, exactly.

Neil: You don't even know what the answer is even supposed to, to remotely look like. Um, so prior, so prior to programming, I guess, yeah, you shifted so it was around ninth grade that you shifted to just more interest in math. Have you ever been programming, and what you were working on had to be stopped for a while because there was like a math block?

Asim: Yes.

Neil: Where you needed to know some math to …

Asim: Oh, yes. {pam} I said no on the questionnaire, but now, for my, for my bioinformatics class. Yes, because that was very statistics heavy. You had to do a lot of statistics on everything. Yes, because I did not know, I mean, the, ok, so the, this was more for the algorithm though. Because the algorithm dependent on math, was dependent on math, and I did not have the background for it. I had to go and ask my professor for it, to teach me how to do the math behind those things. He taught me and then I could do it. But, I mean, for programming itself, just for programming, not considering like, algorithms and everything, even, I think basic algebra and basic calculus should be enough. {/pam} I have that, but …

Neil: Um, so, so have you ever had to learn math outside of class because of programming?

Asim: Yes.

Neil: So, like you're doing something not class related, and you're like, oh for this …

Asim: Yes, I have. Yes, I have. {mot} It does force you to learn things. {/mot}

Neil: Is that primarily the algebra and the basic calculus, or that kind of stuff?

Asim: I mean usually, yeah. {pam} But for, but for stats, you have to do a bit of advanced calculus, if you want to optimize it. That did force me to learn some extra things, just because I

tried to make the program faster. I mean, you can really do things just summing up, right? Putting even a little, using that, I mean summing up one thing, one thing. But when you actually have it in a, when you actually use calculus, it's, it's, computationally it's easier. It's harder to derive, but, when you start calculating for a machine, it would much be, it would be easier in an equation than just looping through and adding stuff to it. {/pam} {mot} And, that is what forced, I guessed. That is, I have been forced to do it. {/mot}

Neil: So, that's actually increased, it's made your programs better by knowing …

Asim: {pam} {map} Yes, it's made my programs faster, and it gave me more math skills. {/map} {/pam}

Neil: So what about the other way around? What about when programming, has programming, I mean you mentioned dissecting problems, any other experiences where you think that programming has affected the way you would approach math? Or any specific, specific math experiences?

Asim: Yes. Yes.

Neil: Or when you're learning math, your programming background?

Asim: {pam} {mot} For, I don't know which class it was, I think it was one of my stats classes, um, I did not want to sit there, I just needed the answer, I just had to fill in the answer, and the error percentage was like five percent or something, so you could be off by five percent, it didn't matter, and, the ideal way to do it was using calculus, right? {/mot} Well you can, but when you, when you, let's see, when you jump in, there's there's the x and the y. As the x moves forward, you can get the value of y. But when you jump in 1's, I mean, that, you get like little blocks of errors, and it can't go beyond 5. {mot} And I did not want to sit there and do the calculus, so I tried doing that, it was, it seemed like it was off by very much, when I did my calculator stuff in intervals of 1 and just started adding it, but when you go down to like intervals of 0.5, the errors get smaller, 0.05, it gets smaller still. {/mot} {sc} And, this is where the programming skills come in handy, you write a program to do it in terms, of like, in intervals of 0.0000001, and then start adding, it gets very, very close to the correct answer. {/sc} {/pam}

Neil: So, you actually …

Asim: {pam} That, that was cheating, kind of. But, yeah, I did get the very correct answer. {/pam}

Neil: So it, do you feel that it was the programming background that even gave you, I mean obviously it was, to even come up with that?

Asim: Yeah.

Neil: Because, so, um. It seems that that was a situation where, um, had you just been another student without programming background, trying to do the statistics problem, you would have

had only one way to do it. But, because you knew another way, um, would you feel that you had a better understanding of that whole problem? The fact that you dissected it in two different ways?

Asim: {pam} Yeah. Yes. {sc} But, when, when, when professors teach that method I was saying, of like incrementing it, one at a time and everything, and taking that area, people who do not have, I don't think, who have not had programming, will not understand it as much as someone who has had programming because that is how a computer does it. {/sc} You just know that kind, because you've been programming … {/pam}

Neil: … several small tasks.

Asim: Yeah. {pam} {sc} Well, I mean, even when you do kind of like, something mathematical, even when you're doing some kind of mathematical algorithm, that is how you approach it. {/sc} {/pam}

Neil: Right, um …

Asim: You don't start, because a computer can't solve an equation for you.

Neil: Um, so what other concepts do you think overlap with math? What other concepts had you been programming that you connect with mathematics?

Asim: {pam} {map} {sc} Um, definitely computational time. Like, calculus really comes in handy when, when you say ok, this is converging to this, especially for something like, um, sort, sorting algorithms and everything, that calculus comes in very handy. {/sc} {/map} {/pam}

Neil: So, how about now, so, going from programming, so now that you've been programming forever, where do you, what, do you feel that there's other areas in math, if you were just now going to leave your programming aside, or just become a mathematician, are there any instances where you'd feel that, you're like oh, now that I have this background in programming …

Asim: {pam} {sc} It helps. It would help in, I don't know, some sort of, it would help very much in all approximations in everything, I'm sure. {/sc} {/pam}

Neil: How do you … do you feel it's influenced your attitude towards math?

Asim: Yes.

Neil: In what way?

Asim: {am} I find, for one, it made math much more interesting. {/am}

Neil: Why?

Asim: I think because, because, say for something like chemistry, right, there are rules that are just rules that you don't, you don't get much creativity to do things. In physics, until you start using calculations and everything, start deriving things, it's always still just learning about rules. Biology, it's still, I mean, it's almost always about rules, right? {sc} But when you start getting into programming, there are so few rules, and everything else you make up. Same as in math, there are few rules, that you can apply it by however you feel like. I mean, those are the two areas where you can get exposed to that kind of thinking early on. Because, I mean in physics and chemistry, you really never, I mean you really never get those rules and you're let loose saying like use it how ever you feel like. You don't get that. But that, um, that kind of thing actually helped me, because I was used to that kind of thinking, ok I have these set of tools, how do I get to this? {/sc}

Neil: So …

Asim: {pam} {sc} That, that helped me, doing, to do math. {/sc} {/pam}

Neil: So, more of a freedom in math.

Asim: Yeah.

Neil: Um …

Asim: {am} It makes you kind of less scared of math. People are scared of math. {/am}

Neil: I'm scared of math! Um, so, so just less scared, just because of the connection. Just because …

Asim: Uh huh. {am} {ps} You, your mind gets trained to think that way, I think. You're not, you're, you're not scared of something you don't know the solution to. {/ps} {/am}

Neil: I agree. Um …

Asim: {ps} It doesn't make you afraid of trial and error. {/ps}

Neil: How do you think, um so, how does math and programming, how's that shaped and play into your professional goals?

Asim: {mot} {pam} Oh, since I program games I use a lot of math, and programming, obviously. {/pam} {/mot}

Neil: So, what, what, what do you want to, what's your long-term professional goal? Where do you want to end up in ten years? What do you want to, what do you want to be doing?

Asim: Ten years, wow. {mot} Ten years … I always want to be involved in like, cutting-edge stuff. So I think, I mean, yeah, I don't know how math would play out in that, but.

Neil: Like computers? Like programming? Like tech …

Asim: Yeah, programming. Any, any, any, anything … {/mot}

Neil: Tech-based science?

Asim: Yeah, tech and science, I want it to overlap.

Neil: You're studying bioinformatics?

Asim: Yeah.

Neil: Um, that field? Are you looking to go into that field? Or maybe …

Asim: Maybe, maybe … not really …

Neil: Um, how does math play into your current programming life?

Asim: {map} {pam} {sc} It definitely helps in optimizations and, well you are efficient at something like, something that has, I don't know, right now I can think of something that has like shuffling and, making random things happen. You can kind of control it if you're good at math. You can use it better if you're good at math. And, I mean, when in games, like optimizing things, like optimizing arrays, sorting arrays, shuffling arrays for example, you really need to have that math skill, kind of. And, oh wait, positioning, like drawing things onto stages and everything. Well, right now we're doing things in Flex. But, oh like six months ago we were doing everything in Flash and everything had to be controlled. Like positioning, every, like positioning was a really big thing. And yeah, algebra, knowing that kind of stuff always helped your equations and everything. Knowing how equations behaved … {/sc} {/pam} {/map}

Neil: So, a lot of math.

Asim: {pam} {mot} Yeah, when you're programming games, you have to have a lot of math. {/mot} {/pam}

Neil: Excellent.

Appendix E: Coded Interview: Ben

Neil: Tell me about your general academic experience. How did you perform academically?

Ben: Uh, it was kind of, uh a little bit hit and miss. I worked most of the time I was in college, and while I was working, that kind of made doing school difficult. So, some of the time, um, my grades were just, ya know, average. And then, once I stopped working and went to school full time, my grades increased drastically.

Neil: What was your favorite subjects?

Ben: In computer science or in general?

Neil: In, in general.

Ben: {aa} Uh, I think biology and, uh, computer science were my two favorites. {/aa}

Neil: Um, can you explain why they maybe were your favorites? What was it about those subjects that drew you to them?

Ben: {aa} Um, I think, uh, both have a pretty hard, uh, science to them. So, there's not like a lot of like emotion in them. So I think that's probably what was appealing to me. {/aa}

Neil: So, what was your least favorite subject?

Ben: {aa} Uh, probably English. {/aa}

Neil: English. Now was this the same even like in high school and college? Were you still always interested in the sciences versus, science versus English.

Ben: Yeah, exactly.

Neil: Um, what kind of things did you struggle with academically?

Ben: {se} Um, actually I did have kind of a rough time with math, and uh … probably math and arts. {/se}

Neil: Um, was it um, a lack of interest, or just …

Ben: {se} {aa} Uh, with the arts, definitely a lack of interest. {aa/} Uh, with math it was more of a struggle I think. It was more difficult for me to, um, to do well in the subjects; it took more time and effort … where arts was just like, no interest. {/se}

Neil: Um, did you end up doing, like, well … in the end? Like, so, in math, did it take you a long time to get it, but you ended up doing well, or did, was it frustrating enough that you just did the minimum amount to get by?

Ben: {se} No, I did well, it was just more of a challenge. {/se}

Neil: More of a challenge?

Ben: Yeah.

Neil: Um, what other hobbies do you have outside of … what hobbies? What are your hobbies outside of academics?

Ben: Um, like, I like snowboarding, I like working out. Um, music, movies, uh … vacations.

Neil: Did any of the hobbies, did you ever feel the hobbies tied into what you did academically?

Ben: Uh, no, not really.

Neil: So, how did you learn to program?

Ben: Um, it was, uh, mostly through, uh, college, and books, and experience.

Neil: So, was it like a formal … was it like you got interested in it and then you went to class, or you got interested outside of class, and so you took class because you were interested?

Ben: {mot} Uh, no, I got interested in class. {/mot}

Neil: Chicken and egg …

Ben: Yeah, I got interested in it in class then and uh, most of it was like through, you know, uh, college and academics. And then, I would say, um, probably twenty percent, maybe twenty or thirty percent has been through, just through books, not college. Not in school.

Neil: So, how old were you when you had your first programming experience?

Ben: Uh, I think I was 24.

Neil: And what, what, what was the inciting instance that caused you to program?

Ben: {mot} Uh, I took a computer science class when I was trying to figure out what I wanted to do, uh, for my major, and I landed on that and I decided to pursue that. {/mot}

Neil: What did, what did you first start creating? What did you want out of the computer science? What, what did you hope, like, to be able to work on? What were the programs that got you excited?

Ben: Uh, I don't remember actually. I don't know.

Neil: Um, how do you, how do you think, do you feel that that played at all into, um, problem solving? How you approached problem solving?

Ben: Oh yeah, definitely.

Neil: In what …

Ben: {ps} I definitely take a divide and conquer approach in most everything I do. {/ps}

Neil: Did you, did you see that that, um, did that approach help with programming, or did, as you programmed, and taking this divide and conquer, did you feel that that leaked over into the rest of, did you find yourself analyzing things differently or approaching other problems outside of programming as you programmed?

Ben: {ps} Uh, I would say that I, uh, it leaked over into the rest of my life, and, uh, I started, I started doing other things that way as well. {/ps}

Neil: So, what uh, what programs did you learn? What languages?

Ben: Um, I've learned, uh, C, C++, C#, um, a little, some, Python, some PHP, some Perl, uh, like, SQL, I don't know if you want to consider that a language or not, but, um …

Neil: So, you took it as a class, so, what did you start, what did you start off with?

Ben: Oh, some Assembly too. Uh, I started out with C.

Neil: With C? Did, um, so since you took it in a class, I take it, correct me if I'm wrong, was it a … suddenly you just jumped in and just learned as a class, so obviously you just learned the basics and then just kept moving on. There wasn't like …

Ben: Yeah, it's not like, I don't think you learn the entire thing at one time. It's, over a period of time, ya know, you pick more things up. But yeah, uh, the, the basic, most of it was learned academically, and then, other stuff, ya know, along the way.

Neil: So, what about your, so we were talking about mathematics earlier, what, um, what do you think it means when someone, to be a good mathematician? To be good at mathematics? What does that …

Ben: Um …

Neil: What do you define mathematics as?

Ben: What do I define mathematics as?

Neil: Yeah, yeah, when you, when someone talks to you about mathematics, what do you, what do you think that encompasses?

Ben: Oh.

Neil: Where do you think the borders are of what becomes mathematics and what becomes …

Ben: {am} I kind of feel like, uh, math is really, really pretty broad. It could, it's basically, ya know, uh, an understanding of, of anything in the universe, really. Like, and, that could, ya know, that falls into every category, I think. I don't reckon there is boundaries in math. {/am}

Neil: Do people assume you're good at math because you program?

Ben: {am} Uh, I think so. {/am}

Neil: Have you had any experiences or even just like random little comments, or conversations, or … vibes when you're in particular classes, or when you tell somebody you're, you did computer science?

Ben: {se} Um, a lot of my computer science classes, I've, I felt like, when they're really math involved, I feel like I'm pretty, pretty average in them. {/se} {am} Uh, like usually I, I also assume people are really good at math too. Um, and I think like, uh, when I look at people around me and how they perform, it's pretty, it's pretty even. But, uh, like some people, ya know, obviously are like on different levels, and they're either really good or really bad at it. But um … {/am}

Neil: So, does it seem to make sense to you, that they have to teach a lot of math in computer science courses, or do you, do you sometime, how do you feel when they, when you have to take these math courses, or when they start talking about math? Does it feel like, like a necessary step? Does it feel that it's intuitive, that it's part of the curriculum? Does it …

Ben: Well, I kind of feel like, uh, at least when I went to school, math was like, pretty general for science majors, um, ya know, you have your standard classes that aren't like, in your major but they, kind of, you know, you know, touch on major stuff along the way. {am} {pam} Um, but I kind of feel like they should have, I don't know, more computer science classes that are more mathematically, uh, based, as well. I, I feel like they should have more, like, uh, I guess, specific math classes for, for CS majors. {/pam} {/am}

Neil: Such as … Like where would you find, so in other words, so, as you're, as you're doing programming, are there particular mathematical challenges that you face that you would have liked to have been better prepared for? Or, that you think …

Ben: {map} {pam} {sc} Yeah, I think, I kind of feel like maybe um, classes like on modeling, uh, modeling things mathematically that you would then program, I think would be, uh, pretty useful. Like methods for, ya know, taking a real world problem, and creating a mathematical, uh, solution or, like a model of it, and then, uh, programming it from there. {/sc} {/pam} {/map}

Neil: So, have you had particular instances where you got held up on your programming because the math became the, the block?

Ben: {map} Uh, nothing that I can think of in my mind right now, but, um, of course a lot of things are already like, ya know, you can research them and find, ya know, what people have done to work in that area on it, but … {/map}

Neil: So, were you taking, um, math classes, while you were taking programming classes?

Ben: Yeah, they were pretty, pretty concurrent.

Neil: Concurrent?

Ben: Yeah.

Neil: Um, did you ever find yourself learning new math concepts outside of the math classes because of the programming? Where I'm in a programming class and now I need to know something about math.

Ben: {map} {pam} Uh, yeah. Um … I'm sure I have, I can't think of anything at the moment, but … {/pam} {/map}

Neil: Um … so, any, can you think of any experiences where either math influenced the way you programmed or programming influenced your approach to math?

Ben: {map} {pam} Um, I'm sure there have been instances, but I can't think of anything off the top of my head. {/pam} {/map}

Neil: Has there ever been a time you were trying to do a math, you were working on a math problem and that something you knew from programming helped out in solving the problem, and understanding the concept behind the math? Or …

Ben: {pam} {sc} Yeah, actually I would say with logarithms. I is an instance I can think of.

Neil: In which way? Math helped programming, or programming helped math?

Ben: Uh, where programming helped math. {/sc} {/pam}

Neil: Why would you think that would be?

Ben: {pam} Well, I mean, um, I think, um, in the past when I was dealing with logarithms, like I didn't really quite understand the, why they were necessary and stuff, and then from programming, it's like, oh, that's a good use of them. {/pam}

Neil: Um, so, so, do you think it was more of that programming gave you a less abstract use for logarithms, where you could actually visualize instead of …

Ben: {pam} {sc} Yeah, exactly. Yeah. {/sc} {/pam}

Neil: … this magical thing that seems to not apply on a day to day. Um, so do you think knowing how to program has overall improved your understanding like, going along with that, the logarithms, is, or any other kind of, uh area of math, has it improved your general understanding of mathematics?

Ben: {pam} Uh, I would say yes. {am} But more so I would say it probably, uh, uh, bettered my attitude towards math, and that I had like, you know, something to use it for. {am} {/pam}

Neil: Expand on that.

Ben: {pam} {map} {sc} Well, I mean for example, like um, when you're doing like a, a, you know, like an algorithm, and you want to make it as fast as possible, um, you're using mathematical concepts to like, ya know, to increase in speed. {/sc} {/map} {/pam}

Neil: So, do you, so do you think that, are you less, cuz you said that, ya know, previously you struggled more with math, um, is math more of a friend than a foe now? Or, I mean, not that it necessarily, maybe it has or it hasn't, um, you could expand on that, whether it's made it less complex for you, or is it just …

Ben: {pam} {am} Uh, I don't think it's made it, uh, really like easier to, to, ya know, understand, I think it's always been a struggle, but I think it's been more, helped my attitude, basically. {/am} {/pam}

Neil: So, you found a reason to need to know it?

Ben: {pam} {am} Yeah, exactly, yeah. Um, yes, it's definitely, like, you know, when you, like, can see things from a new perspective. {/am} {/pam} It helps out that way too, but …

Neil: So, where do you like think the similarities lie between math and programming? How do you feel about the relationship between …

Ben: Uh, I kind of feel like, um, well, you know, I think it kind of depends on what kind of programming you're doing because like there's a lot of different kinds. {pam} {map} But I think probably the majority or at least half of it is not that math intensive, but I think the other half, uh, is. Like, I feel that a lot of programming is like, has been done to kind of like buffer, a lot of the, that's not the mathematical knowledge you would need otherwise. {/map} {/pam}

Neil: Do you think that was a good thing or a bad thing?

Ben: {am} {ap} Uh, I think it's a good thing. {/ap} {/am}

Neil: The abstraction. You said you, you have experience working with like Assembly language. Did you find that a little more mathematical? Or was it, like with the higher languages versus the lower level languages?

Ben: {pam} {map} Yeah, it's well, Assembly's a lot more, ya know, nuts and bolts. {sc} So, it's, uh, especially like with converting like, ya know, uh, like from whatever base you're working in, like if it's a hexadecimal or a base 2. Uh, that is one thing that's, ya know, definitely helped me to understand that concept better. {/sc} {/map} {/pam}

Neil: So, what are your professional goals? Right now, where are you now? What are your goals, and how does programming fit into that?

Ben: {mot} Well, I program now, and I plan to continue to be a programmer for, ya know, the rest of my life, so it's uh … {/mot}

Neil: And how does math relate into your goals? What do you, how do you feel the mathematical concepts play into your current programming experiences?

Ben: {map} Right now it's not much, uh, I don't have to do a lot of math intensive stuff right now. But, that's mainly because I'm doing a lot of web programming stuff, but um, but uh, it does apply a little bit, I mean like you have to like kind of take into consideration a lot of math stuff when, ya know, in, in my job as far as the like, simple math, but ya know, making sure you're, everything is done correctly, but um, but in the future, ya know, I don't know. It may become like really intensive, I don't know. It, it very well could be. {/map}

Neil: So one last, going back to something we talked about kind of before, um, when we were talking about the relationship between, do you, where do you feel they share, or if you, maybe you don't feel they share at all, but, it seems, what, what concepts do you think math and programming share? Where do you feel the overlap hits between the two subjects, or where are they miles apart?

Ben: {map} {am} Well, I kind of feel like you could, you could place, uh, pretty much all of programming under math. I mean, I think, math overlaps it completely, in one way or another. Um, I think, I think pretty much everything in computer science you could link to math at some, at some level or another. {/am} {/map}

Neil: So, so we kind of talked about this, so overall do you feel that, if, would your math be worse off if you lost programming, or would your programming be worse off if you lost the math part? Do you feel they're supportive, or independent, even though they share the same structures as far as, as far as you've learned them concurrently.

Ben: {map} I feel like, I feel like, um, programming would take a bigger hit if, without the math. {pam} I think it would, well, I think, I think math would lose something without programming, but not as much, where programming would lose, uh, almost everything without the math. {/pam} /map}

Neil: So, programming would lose. So, stronger towards the …

Ben: {map} Yeah, programming. {/map}

Neil: … programming relies more on math than math relies on programming.

Ben: {map} Exactly, yeah. {/map}

Neil: You talked about the logarithms and that kind of stuff. So is that kind of, yeah, so in that, that, that instance, um, that allowed you to understand the math but it didn't really create the math, I guess, I don't know.

Ben: {pam} {sc} Yeah, I mean it was more like a useful tool for the math. Like finding out begin of like a algorithm, ya know, I mean, uh, in that case, that's a specific example of like where, ya know, for example logarithim is quite useful. {/sc} {/pam}

Neil: So, had you learned logarithms, I know we're going back, but had you learned um logarithms before, and then just, like previously in your life, maybe in high school, and just like kind of not understood them until …

Ben: {pam} {sc} Right, no, it was more like, I learned logarithms before and like ok, that's interesting, but I could not think of a single use for that. And then, then, like oh, figuring out how long an function will take and you can use a logarithm for that, uh, you know, that's, I think that when I did that I had to go back and learn exactly what a logarithm was, but then, like oh. {/sc} {/pam}

Neil: Did you feel it easier to learn?

Ben: Uh …

Neil: Or was is about the same to learn, or was it easier to learn because you found use for it, or the same to learn because it wasn't that crazy of a concept?

Ben: {se} {pam} {am} I, I think it was easier to learn since I had already learned it before, but it was easier to remember, because I had an actual use for it. {/am} {/pam} {/se}

Neil: Easier to remember. I think that's it.

Appendix F: Coded Interview Brian

Neil: Tell me about your general academic experience? How did you perform academically? How was, how was school?

Brian: {aa} Uh, well, in high school I graduated with a 2.6 GPA. I guess I went skiing just about every day, and, did not pay attention in classes. {/aa} {se} But in college, I actually ended up graduating with both a bachelor's and a master's degree, and I had a 3.8 for my bachelors and a 3.93 for my masters. So not too bad. {/se}

Neil: What did you, what did you graduate in, for bachelors and masters?

Brian: Information Systems for both.

Neil: Um, so did you like, did you like school? So, setting aside that you didn't like to go, how, what, what was your favorite subject in school? What was your least favorite subject?

Brian: {aa} I really liked math and science, and I hated English and humanities and anything that had to do with that. {/aa}

Neil: Why? Why did you hate them?

Brian: {aa} Actually, I liked music, so that's not true when I say humanities. Um, I just, I just didn't like it. It just felt like there were, mostly it just felt like there were rules, that somebody made up somewhere, that didn't have anything to do with, I don't know. Does that make any sense? {am} Because with math, it was like, they, obviously math was also rules and laws that other people had set up, but I felt like they were universal things that could be applied and proved over time. {/am} {se} And, I actually never really liked writing papers, but I was always good at it. {/se} I always had a really creative mind, and a lot of times I'd have English teachers tell me that I should write for a living, but I didn't really like to do it. I didn't find it enjoyable. So… {/aa}

Neil: But, you liked the other ones? You liked the sciences. You liked the math.

Brian: {aa} I liked the sciences, yeah. I liked taking chemistry, biology. {/aa}

Neil: Um, were there any things that you struggled with academically? Any thing that were like, a little more, liking or not liking it aside, was some things just seemed a little harder?

Brian: {se} Um, English literature seemed extremely hard. And I don't know why, it just did. {/se}

Neil: Did you like the science, or did, did you like it, and it was easier, or you liked it because it was easier? Interest …

Brian: {aa} {se} Um, I liked it because it was challenging, and I didn't think, it wasn't easy, well, I guess it was a little bit, it made sense faster, and I really understood it. So, maybe it was the fact that I liked, that I understood it so I liked it better. {/se} {/aa}

Neil: Um, so skiing. What other hobbies did you have all throughout all growing up … high school, pre-college, college?

Brian: Golf, basketball, waterskiing, hiking, mountain biking.

Neil: So, very active.

Brian: And I, I did a … While I was in high school I used to be a body builder, and so I was about forty pounds heavier and, and lifted weights about two hours a day.

Neil: Woah. That's a lot. So, how did you learn to program?

Brian: That was all in college.

Neil: Um, was it, did you start formally or informally? What, what led, what led to programming?

Brian: {ap} {se} It was formally. I started formally, but it was like, basically the classes, my programming classes were the ones I excelled the most at because I really enjoyed them. {/se} {/ap}

Neil: So when you took the programming, you didn't really know you were going to like programming?

Brian: {ap} No. I had absolutely no idea.

Neil: And now you're programming.

Brian: So, and part of that comes from the fact that my family's not really oriented towards anything like that. My family's really active, and they're huge people people. My parents didn't even have a computer until I graduated from high school, and so yeah, so that was a big part of it. {/ap}

Neil: So, you just ended up being in class and thinking like, I like this kind of thing.

Brian: {ap} Exactly. {/ap}

Neil: So, so you just started from the basics, exactly as it would be presented in class. Did you start doing scripting first and then move into programming, or did they like dump you into . . .

Brian: So, it was actually Java. We jumped straight into Java. {mot} {ap} And I really enjoyed the class, so I ordered about six books after the class was over …

Neil: Really?

Brian: … and started going through books on my own, and started learning, programming on my own, so. {/ap} {/mot}

Neil: Um …

Brian: {mot} Everything that I've learned with Flash and ActionScript and Flex, has all been on my own, so that's, there was nothing, there was no formal education in that. {/mot}

Neil: Oh, past, past the first …

Brian: {mot} There's, I only took two programming classes and everything else I've learned on my own. {/mot}

Neil: Oh, just following that. Um, so what about your experience with math? How was math? You said you understood it easier, it was easier, it made sense, unlike …

Brian: {am} {se} Yeah, it made sense. Um, I took calculus 1 through 3 in college, and I got an A in each of those classes. So, I just, I just enjoyed it. {/se} {/am}

Neil: Um, so to you, what is mathematics? What does it mean, and what does it mean to be good at mathematics? What do you think that means?

Brian: {am} That's a good question. Mathematics is more of a, I think it's more of a mindset than anything. It's just, it's a quantitative way of looking at things in the universe. But being good at mathematics mostly just means that you understand the theories and the rules and how to apply them. I don't think that you can be a prodigy and be good at math, I think that there has to be somewhere … I mean, you can, you can natively understand the rules and theories really easily, but I don't think that you can just automatically be born a person that's good at math. {/am}

Neil: Um, do people, since they know you're a programmer, do they assume that you're good at math?

Brian: {am} Um, I don't think so, no. {/am}

Neil: Um, so you've always, so now that, so you said that you've always done good in the sciences and math, you didn't even program until, so you had a lot of experience with math before you ever started programming. How, how far into math, what, what level of math had you had by the time you'd started programming?

Brian: Calculus 3.

Neil: So, all the way to Calculus … um, did you take any concurrent math courses with programming?

Brian: No.

Neil: You were done? Just had finished all the math. Um, have you ever been programming and some kind of mathematical challenge stopped your programming process?

Brian: {map} No. {/map}

Neil: Um … do you ever, have you been in situations where you've needed to learn mathematical concepts outside of the class, since you've been programming … you've got to, so it didn't stop you, but you didn't know about something, you had to go figure something else out math related? Where you're like, oh, I remember this.

Brian: {mot} Yes.

Neil: So, I have to learn …

Brian: Yes. {map} {pam} {sc} So, most of the, most of the math that I do now is all calculating things within a 3-D space and I never took trig, so anything that has to do with sine or cosine, or any of that type of thing, I've had to figure out on my own. {/map} {/sc} {/pam} /mot}

Neil: Uh, trig. Um, so you never took it specifically, or in like high school?

Brian: Never took it.

Neil: Um, alright. Do you have any experiences where math influenced the way you thought about programming or visa versa? Kind of an open-ended question, but …

Brian: {am} {pam} Um, so I have had experiences with, how it, programming has made me think about math, but not the other way around. {/pam} {/am}

Neil: That's good, no. Explain that one. How has programming helped you?

Brian: {pam} {am} So, doing things like, I'm going actually specifically reference trigonometry, even though I never took it. {sc} I always thought that geometry and trig were kind of, worthless, because I never planned on being an architect, or, um an engineer, that would actually need to use them, but once I jumped into programming and had to start using those all the time, I started realizing that it was actually an extremely useful practice to know those things, and I wish that I'd learned them in the first place. {/sc} Um, and it actually made me realize that you can apply it to a ton of different concepts instead of just programming or doing whatever you're doing. {/am} {/pam} So …

Neil: So, you had learned that, I assume, like younger basics of like some kind of general, like triangles, and kind of space, and the area under a …

Brian: Yeah, exactly.

Neil: … curve.

Brian: Obviously that stuff but not, not calculating, um, the area of a triangle if it's, if it's rotated by twenty-two degrees or something like that, and trying to figure out what the width and height and everything else is inside there.

Neil: So, what, what, what were you doing when you needed those?

Brian: I wasn't.

Neil: No, in programming. What …

Brian: Oh, what did I do when I needed those?

Neil: What, what programming task were you doing when you decided, oh, I need … this math is useful now!

Brian: {mot} {pam} Well, it was basically, ok, so the first one that I did was creating, um, just a clock animation and I, I was doing it in Flex because I didn't want to use the timeline or create some sort of shape tween or something else, but I actually wanted to draw it to the stage, and um, to do that I had to know the exact mathematical coordinates of where the clock would be placed and since the clock was refreshing at thirty frames per second, that meant that I needed to refresh my angle, and my trajectory, and everything else, um, thirty times per second, three hundred and sixty times a minute, and, so yeah, it became really important when I was doing that. {/pam} {/mot}

Neil: Um, so a concept like that, um, I'm trying to think of a good way to say this. The concept … how?

Brian: {pam} The other, the other way it became like, good was um, drawing glass buttons, and doing things in 3-D space and trying to make them look spherical, and actually understanding the mathematics of how you could make that look a certain way. {/pam}

Neil: So, do you feel in that environment, because you had to do it to accomplish a goal, do you think that um if you had been presented with these ideas, without the computer programming aspect, um, how do you think the concept would have differed in your like, approach to that? All the math you learned had a teacher just giving you this in class. What do you think, what do you think the experience would have been in a math class had you just learned …

Brian: … in a class …

Neil: … had you just learned this separate from any other kind of computer programming application?

Brian: {pam} {am} I probably wouldn't have been interested at all. {/am} {/pam} So.

Neil: So, do you feel that um, do you think the knowledge of programming has changed your attitude toward math?

Brian: {am} So, I feel like a practical application of math always changes my attitude of it. And, to add to that, I think that anything, like, when you're in a math class, I think the best way to learn is to add practical examples to that which a lot of teachers have no capability or capacity to do. {/am}

Neil: Or, some of them may be hard to do.

Brian: Or, they've been trained …

Neil: So, what would be a practical, without computer programming, would, could you think of a practical example of that situation you were in? Without the, to? …

Brian: That's really hard to say. I don't know actually.

Neil: Because it seems that would be hard to come up with a little paragraph practical example in a textbook to explain, right?

Brian: Yeah. It would be.

Neil: So, it didn't seem it was useful to you until you all the sudden had to do it, and then you, so you had a, you had a personal, customized practical example of why you needed it, therefore it increased the importance of it.

Brian: {pam} {am} Yeah. {/am} {/pam}

Neil: The attitude.

Brian: {am} Well, I think that's what it does and people, when people go to school for engineering or something, I think that it just, like, the engineering aspect of things solidifies the mathematical concepts. It makes them actually make sense instead of just being theories. {/am} So.

Neil: So, now that you've gotten into programming, looking back on any math, what do you find about the relationship, um, similar to what you were just talking about, um, what do you find useful about math in programming, or their relationship? How do you view their relationship?

Brian: {sc} I think the relationship is that they're both, um, quantitative, and, basically with math you're trying to come to, you're trying to come to a solution for a problem, and the route you take to get there is the puzzle or the mystery and I kind of feel that that's, that's exactly what you do with programming. You have an end that you're trying to get to in a problem, and a barrier in

between, and how you get there is basically the puzzle, I guess, that helps you get down that path. {/sc}

Neil: So, do you feel that programming has increased, like, your problem solving approaches for puzzles?

Brian: {ps} Yes. {/ps}

Neil: In which aspects? In math? In programming? Or just in any area? What, what ways do you think it's increased you problem solving?

Brian: {ps} Just in my habits. There's always, I mean, it's like playing chess for your first couple times, like there's not, you don't think about every possibility that could happen down the road, you just think about your current move. But as you play, you start realizing that if you make this move then this could possibly happen and this could possibly happen and all these things could happen, and your thought to make a single move increases. And I feel like that's what programming does. It basically just creates a habit of always thinking about what this one choice is going to do, and how it's going to affect every other aspect of the program. {/ps}

Neil: So, methodical.

Brian: Methodical, yes.

Neil: Habits. Understanding the outcomes. Um …

Brian: {am} I don't feel like math does that same thing, because typically math, it's like, in programming, there's multiple, there's hundreds of paths you could take to accomplish a goal. But in math, there's typically only a single path. {/am}

Neil: So, do you feel, would you say math has a greater influence on your programming, or programming a greater influence on math? Understanding that you obviously program way more than you, you don't just sit around probably doing math for eight hours a day. But how do you view, um …

Brian: {pam} I would say programming has a greater influence on math. {/pam}

Neil: … on math. A specific … any specific reasons? Like what aspect of math? The ability, the attitude, the? …

Brian: Just the approach, I think. The approach of math.

Neil: So, what …

Brian: {pam} {am} It, it kind of just helps math be more realistic, I guess. {/am} {/pam}

Neil: … be more realistic. Concrete.

Brian: Yeah.

Neil: Um, so what are your professional goals? Where are you, where are you going from here?

Brian: {mot} My professional goals are to one day … be rich. {/mot}

Neil: And how has learning to program shaped these goals? And how has programming and math influenced your abilities to be rich?

Brian: I don't know. Um …

Neil: Um, do you use a lot of mathematical concepts?…

Brian: No, I mean my, ok, so my goals are, are like, I'd like to do, I mean, it's hard to say, like I used to say I wanted to go into management and I wanted to do some things in that realm, or I wanted to do more things with people. {ap} But, the problem is that programming is really exciting to me. And so, I mean, some people harp on the fact, like do you really want to be a code monkey all your life? I, I don't know. I know that right now I really like doing it, and so, um, so where I go from there? {/ap} {mot} I mean, currently I'm doing things in Flex and building rich Internet applications, and um, my career goals would be to be recognized by Adobe consulting and possibly become part of their team. I don't know if that would happen. Or to even be able to, like, to go to the Adobe conferences and like discuss, like, major topics that other people thought of, and maybe present things, and shape the way, um, things are built and run. Like that would be a major career goal of mine. {/mot} So.

Neil: So what, how do you think mathematical concepts play into that? Or into your current programming, where you are now, how does, how does math affect your daily life?

Brian: {map} It makes me methodical, probably too, too methodical. Some things you just don't, you just shouldn't think so far into the future about some things. {sc} Some things should just be simple, but yeah, but the truth is that as a programmer, I think way too methodically, and yeah, so that's kind of how it's affected my life. {/sc} {/map}

Neil: Alright. Thank you.

Appendix G: Coded Interview: Jason

Neil: Alright, um . . . to start off, tell me about your general academic experience. How did you perform academically, high school through college?

Jason: {se} {aa} Well, um, I did really poorly in high school, um … I, I thought that I was too smart, and so, I didn't really do any work. I just thought that I owned the school, and that I was so smart that it didn't matter, so, um, I just kind of did my own thing. {/aa} {mot} {ap} Like I would, ya know, make websites and do my own thing and thought that I'd be a millionaire pretty much cuz the dot com boom was about to happen. {/ap} {/mot} Like, when I realized that that wasn't really going to work out, I decided to go to college, and um, in college I did really well. {am} Ya know, community, I started at community college and just did really well in my like sociology, social science classes, and ironically I just did crap in math, uh, and my science classes, I just couldn't really … I don't know, I didn't have a good enough, like, memory to memorize all the formulas. And so, I could figure them out, and I could like, um, one of the, my biggest problem in math was that, the tests were, I didn't have enough time. Cuz if I had the time, I could have uh, basically solved all the formulas and figured out what I needed to do, but, anyway. {/am} So, I uh, any rate, I did ok, I got like a 3.0 in college. {/se}

Neil: What was your favorite subject?

Jason: {aa} Environmental studies. {/aa}

Neil: Environmental studies. What was your … why?

Jason: {mot} Well, I think, I think kinda going towards, ya know, to kinda tie it into the rest of my experience, I was really interested in technologies that were, like, benefitting the world. And so, I looked at it a kind of a philosophy that encouraged, um, ya know, innovation and interesting things. So, I'm still looking at, ya know, how can technology benefit people, and how do we tie those two things together. {/mot}

Neil: What's your least favorite subject? What was your least favorite subject, rather?

Jason: {aa} Um, thinking … definitely, probably, in college at least, uh, economics. Ya know, I didn't, just, I didn't like the way that it, basically, uh, took people out of the equation. Well, turned people into an equation, actually. {/aa}

Neil: Um, so, with, economics was your least. What was, were there any things that you struggled with academically? Was that tied to why you, was something like, you didn't prefer a subject over the other because of an academic struggle?

Jason: Interesting question. Um, I'm trying to think of where my worst grades are. {se} My worst grades are actually in Spanish. Uh, any sort of foreign language class, again, I just think, I had problems memorizing things. I had problems remembering what it was, and I, I did ok with like, ya know, speaking and, ya know, trying new things, but it was just memorizing that I

couldn't do it. So, in college, I think my worst grade was actually probably Spanish. {aa} And even in high school, um, my main problem was just, I didn't care. So, but when I got to college, when I was actually trying, it was just hard to keep up with all of the work, in uh, Spanish, and that sort of thing. {/aa} So that was difficult. Any classes with tons and tons of reading also kinda killed me. {/se}

Neil: Um, what did you like to do outside of academics? Hobbies?

Jason: Um, well, um, from high school and through college, I was really into the music scene, so I was always going to concerts and involved in like political stuff, back in Seattle. Ya know, we'd do, I don't know, we'd go to anti-war protests and crap like that. It was, exciting, and um, ya know, I would also do websites. {ap} And so, I started my first web design company when I was in ninth grade. I was like 14. It was called [company name removed]. And then, so all through high school really, I was working on websites. I started working for a web hosting company when I was in, I guess, a sophomore in high school, and I basically continued in that industry, so while I was going to study environmental studies at college, I was working the whole time doing computers on the side. {/ap}

Neil: So, is that, is that when you learned to program?

Jason: I learned to program when I was twelve.

Neil: Would you … how did that start?

Jason: Yeah, well, basically, this goes back to academics. {aa} I hated, um, basically I hated middle school, and I didn't like … this is kind of, this is really the epicenter. I got kicked out of PE because I spit on the floor and I think I punched a kid in the face cuz I didn't like jocks, and I didn't like sports. {/aa} {mot} And so, I was banned from PE for my seventh grade year. And so, I just sat in the office that period. And then at some point, I don't know why, I got curious, I had some cool teachers in like fifth, sixth grade, and I started getting into computers. {/mot} Um, my parents had a computer, like I was able to borrow one from a family member, and I started bringing in these big like three-inch thick computer programming books, in like seventh grade, and I would just read those while I had to sit through that period, because I was banned from PE. And so I started studying first Visual Basic, um, and then I taught myself Java, and then so through middle school I was learning Java, just completely on the side. I didn't even have the, enough computer equipment to really program, so by the time I got to ninth grade, um, I think it was ninth grade, I was able to actually take a Java class, and, it was the first time I'd ever actually been able to program, uh, ya know, to have the actual equipment to start doing stuff and applying it. {mot} And so, I took that class and there was like a web design class, and I started writing, in ninth grade, I wrote a program to, that keeps track of which homework assignments each teacher had in Perl. So I learned Perl and Java. {/mot} You know, by the time I was in ninth grade, I felt like I already knew I was learned programming. And then, ever since then, ya know, I've learned PHP, and kind of went on from there.

Neil: Um, how do you think that affected, as you were learning programming, how do you think that affected your problem solving?

Jason: That's a really interesting question. {ap} {mot} {ps} Um, kind of as a side note, but maybe it applies, I was really passionate about programming because I knew that it was something that I could do well, and that I could, I could use it to solve problems that no one had solved before. {/ps} I mean, like, it was my idea, as far as I remember, in the ninth grade to create this program where teachers could enter in homework assignments and then students could put in their student ID number and check them. And it was like, no one was doing that. {/mot} And this was like in, I don't know, like, '97, '98, I mean, it was, it was old school for those things, and people weren't doing a lot of that, and so, I was trying to do really innovative things, it was really cool, and um, but I was struggling academically. I just didn't care about any of my other classes, and so, one of the good mentors I had took me aside and was like, hey look, you're, you can't be in our programming club if you don't get your grades up in your classes. And so, um, and you need to go to college and all this stuff. And I was like, I'm not going to college, I'm going to be an Internet millionaire. Who cares? {ps} And so, um, but really, I got passionate about it, back to your original question. It … I got passionate about it and I realized that with programming I could solve problems, um, that hadn't been solved before, or in new ways. {/ps}

Neil: Um, so, it changed your outlook on various problems?

Jason: Um, kind of. {ap} I guess one interesting thing that it did was it, it helped me see a path for myself to be successful. {ps} So, before that, I, I was just listening to death metal in seventh grade, and pretty much, had no passions except for, I don't know, lighting things on fire and being an idiot, and so when I realized that I could do cool things with programming, and actually see myself being successful, I saw a path for myself to kind of get out of the crumby situation I was in, and, and so, the problem, I guess, was my life that I was figuring out how to solve with computers. {/ps} {/ap}

Neil: So, as they said, you needed to bring up your other grades, your other subjects. Um, what about your experiences in math? First of all, what do you think makes a good mathematician? What does it mean to be a good mathematician?

Jason: {am} Um … I had a science teacher who would always kind of make fun of mathematicians. He basically said that um, that mathematicians kind of solve theoretical problems, but scientists actually do something cool with them. And so, I don't know, I, and then I've had math teachers that kind of say, well math comes before science. And so there's kind of always this battle. I've always been really interested in science, and science, I mean obviously relies on math, but as for being a good mathematician, ya know, personally I see it as someone that does something, that, that can solve problems in innovative ways but, it has to apply to people. At some point, it has to help or affect people. {am} {mot} That's kind of my personal thing. Whatever you're doing, it doesn't matter as long as, if it help doesn't help people in some way, if it doesn't benefit people. {/mot}

Neil: Do people think that because you program that you're good at math? Do they assume you know math?

Jason: {se} {am} Uh, there's this running joke at where I work that, um, that uh, anytime that someone comes in, we have like a project manager come in and uh, they'll talk to us and they'll say, we need to do this and this, and they'll start throwing some math at me, I'll just be like, I don't know math, I don't do math, I'm a programmer, and then they'll, it's just this big joke. {/am} Obviously, programmers are supposed to be really good at math. I don't consider myself to be a math expert or a math genius. I mean, at school, I was above average in math, but when it got to complicated stuff like calculus, my memory just made me suck at it. {map} With programming, I don't need to have a good memory, I have references I can just, I can quickly access the information that I need to solve the problems, so that, that is no longer a limitation for me. {/map} So I think I'm inherently, I have some good math problem solving skills, but I don't have the um, I, I just, I don't have the brain. {/se} It's … it's the drugs.

Neil: How did you feel about your math skills before you learned to program?

Jason: Um …

Neil: And did you, you took concurrent, so you were in, you were doing programming, and then you also took concurrent math classes, right?

Jason: Sure.

Neil: So, how did you feel before and …

Jason: I'll try. Ok, fifth grade. Ok, before, I was into computers on a very basic level. I mean, I don't know, I was like eleven. {am} Um, but I had a really passionate teacher: [teacher name removed], [school name removed]. Fourth and fifth grade I had the same teacher. She was the head of technology for our district so she had, we all have brand new Mac LC II computers, and we were like using computers all the time and she got me to sign up to be part of the Math Olympiad. {/am} And so, our team, I was part of this math team, and it was a lot of mental math, ya know: 2 times 20, 2 times 48, ya know, this kinda thing, and we, I would go to classes and after school, it was almost like a club, and then, at the end of the year, at some point in the year, we went to this big math competition, and ya know, as a side, I didn't fit in well, even in elementary school, I was a poor kid in kind of a rich neighborhood. {se} Like, I was smelly, like, I was fat. I didn't fit in very well, and so like, I, I knew that I was smart and I knew that I had this capability, but the other kids, kind of, on our team, just thought I was a weirdo and they'd make fun of me. So, I didn't have a lot of, I had some confidence, but, I think, socially, I didn't have a lot of confidence in myself. And so, here we go, we're part of this big competition, and we do really well at the competition. But like, even after we kind of, we got like fifth place out of a couple hundred, you know, schools. We thought it was really good. But even after that, I was still made fun of by my friends or whatever, in that, or my peers, at that level. Well anyway, a couple of years later, um, math, I was kinda started to struggle with a little bit more. {/se} I, I believed I had this potential, but then I started doing programming. {pam} And so, I'd often go, I'd have programming problems that I didn't have the math yet in my mind to solve, and so I'd go to like the next grade math teacher at my school, and this was in high school, in like ninth grade, I went to like the tenth grade math teacher and said, how do I solve this, this problem for this program? I was making, um, one of these brick games, that you bash the bricks with a

paddle, yeah, like um, Breakout, or whichever these types of games. {sc} And so, just experimenting with Java, working on building these games, I didn't really have the math to understand the angles everywhere, and so, I'd go to these teachers, and have them help me, ya know, solve these math problems. {/sc} {/pam} Um, I don't know if I answered your question. I don't even remember your question.

Neil: Um, it was, how did you feel about math?

Jason: {am} {se} Ok, so, growing up I had a really strong confidence in my math skills, but I got made fun of, and then sort of, sixth grade, I was having a lot of social problems that actually really affected me academically. {/se} And so, I kind of, like my math stuff kind of fell out at that point. And then, I think what happened was, once I got into programming, I realized I did have, it kind of resparked what I had earlier with math. {/am} {ap} Ya know, math and science, I was really strong in like fifth, sixth grade. Well, that kind of died out in middle school, but then it switched over to programming and then I was able to make sale of that. {/ap}

Neil: You actually answered a few later questions. That's ok, shootin' 'em down.

Jason: Sorry.

Neil: Um, so you're talking about, so you had at least one situation, um, with the breaking game, Breakout type game where you felt you needed to go outside of a class, so, you obviously had a concurrent math class at that age. But, this was a chance where you were faced with a program challenge and that you had to actually seek out math on your own.

Jason: Yeah.

Neil: Um, do you feel that programming influenced your attitude towards math? Did learning to program have any kind of influence on your attitude? I mean …

Jason: Interesting question, ok. Um, I definitely, well, ok, I'll put it this way, I didn't learn programming formally. {am} {pam} So, I learned programming informally. And I know that computer science, for example, people who study computer science in college, they have to learn some pretty rigorous math to understand things like optimization, and um, well, ya know, how processes work, and ya know, computer, fancy computer hardware. I didn't learn any of that, and so I didn't get the respect for math that, I think, formal computer science education gives people. {/am} So, I kind of think as, in a weird way, I think of programming as an alternative to math. Um, so like, I can't solve this using normal math but I can write a program that will solve it for me. {/pam}

Neil: So, do you think they share some concepts? So, you're solving a problem, so you have a problem faced with you, you said you don't know how to do the math, but you can do they program. Do you feel that there's like some kind of related concepts that they share that enable you to tackle …

Jason: … the same types of problems? {am} {map} Um, yeah, well I think, I know for a fact that programming is built on complicated math, ya know, it's a really, really good and exciting crazy math just to get to where we are with computer programs. I just, like, I just can't, I can't go that low. I just, I have to use a higher level programming language to solve the problems, because like, I can't, I don't know, for whatever reason, like, the lower level math that actually is the foundation of the programming and I fully admit that math is the foundation of computers and computer science and programming. {/map} Like I just, I don't know, I can't grasp it, so I use programming as a tool to help me to use, to kind of harness the power of the math without having to actually go … I don't, I don't want to have to go there. {/am}

Neil: So, do you know, can you think of any specific concepts that they, uh, share? I guess, besides the low underlying structure that programming …

Jason: Like at the level where I do programming, what sort of math do I still use?

Neil: Yeah, math that you still use, or do you see any, um, analogies between the two?

Jason: Interesting. Um…yeah, let's see. {sc} One of the biggest and simplest things, forms of math that computers use is this like loops. So, in math, in computers you're always doing things with loops, and an important part of that is that each time you go through a loop, you, you have to, uh, like increment, sort of, by one. Ya know, you go through you're zero and you're going to one, and you're going from zero to a thousand. And so this simple counting is used all the time in programming. But also, complicated objects are stored in vectors which is sort of a math form of um, having lots of numbers in kind of a square. And so, there's, ya know, when we store data in a computer, at least in a, when you're programming data you have arrays and you have hashes and associative arrays and those are all forms of vectors in kind of a mathematical form of storing, uh, data. And that's, that's quite used. Also, ya know, mathematical concepts of optimization. I mean, when you're, you have to figure out what's the quickest way of accomplishing something. How do you solve it in the least number of steps? And so, um, in order to do that, you usually, um, you have to talk to people that are smarter that yourself usually to figure out how to do it. {/sc}

Neil: Um …

Jason: {sc} They're usually math problems. To figure out how to optimize things, make it more efficient. {/sc}

Neil: So, you're talking about vectors, you're talking about incrementing, doing, um, functions, doing one process over and over and over again on different data sets, ya know, and that kind of stuff that's common, everyday practice. Now, you started programming at a young age, so, I guess I'm asking you to imagine, just pontificate on this a little on this. How would you have felt, how do you feel being presented with those concepts had you not understood programming?

Jason: Oh, interesting question. {se} Um, you know, it's interesting, it was weird for me, kind of, like I described earlier when I was young, I knew I was good at math, and I thought I was good at math and science, and I've always been passionate about science and interested in it, but I

sucked, ya know, in college, I just couldn't do super well, I found myself doing better at social, ya know, social studies, which quite surprised me. {/se} {am} {pam} So, anyway, back to uh, back to kind of where I would be without programming, in terms of my math … um, I don't know, I'd probably hate it more. Um, pretty much, programming allows me to use math to help me do what I want to do. I think without it, I'd have less patience for it, and I couldn't even see the purpose in it. {/am} And even now, it frustrates me to have to get into the math side of things to really have to, ya know, if I know that something it's working well, I just want to get a computer to solve it for me. {/pam} {se} I mean, I guess I'm lazy. I'm too lazy to be a mathematician. I'm not, I don't want to go that deep, and so, without, with, with, I expect the program to solve the math problems. {/se}

Neil: But you understand the problem? So, right? You recognize …

Jason: Ok, so …

Neil: You obviously recognize that there's a problem that you know how to solve. But so, assuming, we're just assuming that it's ok to have something else do the work for you.

Jason: Interesting, yeah, yeah …

Neil: It's interesting that you say, because you obviously have identified the problem which is half the solution, right, is identifying the actual problem, recognizing the problem, understanding the possible solution, just not doing the work …

Jason: Yeah, I mean …

Neil: … the work in between.

Jason: {am} {aa} I love math and I love science in a way that, um, so much cool stuff has come from math and science, and I guess, I don't see it in any way as an end to itself. {/aa} {pam} And I think with programming, I can easily see the value of the underlying math. I can easily see how we can put that math to use. I think without programming, I'd have to have some other, ya know, I'd probably look to science, and ya know, different sciences, you know, physics for example. {/pam} Physics is just completely based on math, and I would look to that to use the math in a useful way, I guess. {/am}

Neil: Which sometimes is hard, right?

Jason: Which sometimes is hard and stuff …

Neil: In programming. So I asked earlier, how has it affected your attitude towards it, but in reality, and correct me if I'm wrong, programming has made you understand, or at least acknowledge why we need to know that math. Or has helped, hand in hand …

Jason: Well, ok.

Neil: Because, I mean, personally, some math stuff is really abstract …

Jason: {am} It's really abstract, of course, and so, yeah, I mean, seeing the programming …

Neil: … unlike chemistry.

Jason: … putting it to use, it helps me see it. But, the problem is, there's not always a clear link between the programming and the math. So, I know that, that the math has been solved, and I know that as I program something, that I'm, um, that I'm using math to accomplish my goal, but I don't necessarily always know what the math is. And so, I think without the programming, I would be lost with the math. I would be more confused about it. Where as with the programming it's kinda helped me, um, put it to use effectively, or something like that. {/am}

Neil: So, what are your professional goals, and how does programming and math play into that? Current? Current profession?

Jason: Yeah, no, that's cool.

Neil: Future profession?

Jason: Yeah, no. {mot} {ap} I mean, right now, I do computer programming, it's fun. Um, I have a degree in Environmental Studies, I'd really like to start my own business, and um, apply both IT and, uh, ya know, environmental ideas. So, figuring out what kind of computer programs can help solve environmental problems, or can help, ya know, societal problems. {/mot} It's a weird question, ya know. It's important, but, um, ya know, I think there's a lot of ways where those two things can overlap. {/ap} {am} And so, where math comes in actually in a weird way, uh, in college, I studied some economics, and it was kind of a weird thing, cuz like, I hated it, but on the other hand, I saw that the whole world runs on economics, and economics is literally just math. And it takes human problems and turns them into math problems. And, I have like this inherent rejection of that, as like a valid way of dealing with problems, cuz I think of them as really being important about people. {pam} And so, um, I, when I have this um, when I have my personal goals, I realize I'm going to have to use math both in the economic side and then also in the computer side to make a successful business. Even though I want to help people, I going to have to use math, both to figure out, crunch the numbers, and to do the technology. {/pam} {/am}

Neil: Excellent. That's it!

Appendix H: Coded Interview: Mark

Neil: Describe your general academic, general academic experience.

Mark: {aa} Ok, so, always interested in school, always liked, always liked learning, always looked forward to going to school, at 10, actually I got wrapped in the knuckles when I was nine by a teacher saying you're much better than this, you're much better than this. {/aa} I was the kid that would take ox's eyeballs into the, into to the, one of those kid who'd cut up the ox's eyeball. I was the one that ran into all of the people that were fighting for the eyeball saying, look what I've got in my hand. {se} So anyway, a teacher pushed me into being, to being, to achieving my potential, so from the age of 10 to the age of 16, I was a very good student, I was one of the better students in my year at high school. I did very well in my, in the first level of exams in England, which are called Levels, I think I got eight or nine on Levels, which was I think there were two or three of us that got that many in the year. So I was as good as a sixteen, fifteen-sixteen year old. {/se} Um, when, all the kids, at sixteen in England, high school equivalent, people leave school, um, majority, the better students stay on, stay for a couple more years, take fewer subjects, take fewer exams, consequently spend their time focused on two or three subjects. {se} I did physics, math, and geography. Bad choice, because, English, history, and geography would have been my, my easier subjects, ones that came naturally to me. Um, failed, and um, retook them, dropped the math, did geography, got an A in geography, which, I was the only person in that year at that time to have ever got an A in geography. Didn't succeed with the physics, did the physics again, finally passed the physics and managed to qualify for university, which, again in England is difficult to do. {/se} Not many people got there, got there at that time. Did, did a physical geography degree which was very much focused on quantitative things, science. Um, most of my focus was on meteorology. {se} Um, um spent the first, didn't do so well in my first year. The next two years, it was a three-year degree, our degrees in England are three years long, um, just focused on one subject, so we don't do things. Did well for the next three years, got one of the better degrees in the year, so again, I was, like when I was in high school, I was one of the better university students. {/se} Um, so um, and again, I got, I got in a pure calculus exam which was called meteorology, I got an A, which was an unusual thing to get in an English university, so I was pleased about that. {mot} Um, and then when I came to America, I did a GIS, Geography in GIS Master's degree at the [university name removed]. And that's where I did mostly well, programming was what I did there. {/mot}

Neil: And so, you're talking about the English, history, and geography are your favorite subjects. Math wasn't.

Mark: {se} Well, they were the ones that came most naturally to me. {/se}

Neil: And so, that's what I'd like to ask, so, naturally how, was it because do you feel that it was just easier, or that you were more interested in that way, again, um, mentioning about quantitative, maybe versus … qualitative, something a little more quantitative?

Mark: {se} Yeah, yeah, so that's, so that's, so, so my, thinking about that has always been … that I'm wired to understand those subjects more than I am, um, the qualitative world, but I'm

terribly interested in the quantitative world. And, my early teachers also were, were, contributed probably to my lack of quantitative natural abilities. But, I'm inclined to think it's more of the way I was wired. {/se}

Neil: So, you wouldn't say that just because they're easier they're your favorite.

Mark: {aa} No, I think, like writing, I love writing, I love reading things that are well-written, I love the language, and yet, if you told me, and if you asked me what an adverb was, I haven't got a clue, I couldn't tell you what an adverb was. {/aa} I mean, I have no idea at all. And my wife is like horrified that I cannot know what an adjective is or what a noun is. I mean, I know, I know, but I don't, I don't look at the language in a, um, in a um, rule based way. So, so when I write, I write and think, and, you know, like spelling, I'll look at a word, and know what's wrong just by looking at it. You're probably the same. You can look at a word and go, that doesn't look right to me. Whereas, my wife is not naturally, she's good at spelling, but she's not naturally good at spelling, and she won't see that. {se} She won't see a word when it's spelled wrong, and I'll always see a word spelled wrong. And I'm not, that's not because I'm interested in words, that's not because I have had a great teacher, that's because I'm wired to see that stuff. {/se} And, and when language, when language flows, when things are written well, I always pick up on, like I mentioned that John Pill book, beautifully written, absolutely beautifully written, and Bob Dylan's autobiography, have you ever read that?

Neil: I haven't.

Mark: It's fantastic, the way it's written. It's written in a very conversational way, it just flows beautifully. {se} But I pick up on that stuff, and a lot of people, not a lot of people, other people may not pick up on that stuff, so I think there's a wiring that means that I look at that stuff and that, that it just, it just resonates with me. {/se} {aa} But quantitative stuff, even though it doesn't come easy to me, is I find fascinating you know. The, the whole world of physics, when I suddenly looked at this world of physics and got it, I thought, god, I can answer all of these questions that I've always wondered about, ya know, it answers so many real questions, it's not this sort of, this sort of ephemeral, sort of, ya know, wishy-washy thing. It's really solid. It gives you something. {/aa} {am} I, I always thought about math, ya know, what's the point? Ya know, I've worked through all of these equations and I get that bloody number at the end of it all, well, what's the point? Whereas if you say you know, you drop a ball off a building and ya know, you could do all of these calculations and you know exactly, you get an exact number from something that's, something you can visualize and apply to a real thing. So I know physics and math so close, but math is more the pure side of things, and physics the applied side of thing. {/am} {aa} So, my interest in physics, is, and it translates into geography and you know all of that sort of a thing is that it's a real, tangible something you can get your hand around and grasp. That's what interests me about it. So, quantitative, that's what interests me is the quantitative world. It doesn't come naturally, but I'm interested in it, and it's a struggle for me at times, but I'm interested in it because it's something you can wrap your hands around. {/aa}

Neil: With uh, you mentioned um, talk about that physics is applicable, physics is applicable, you can drop something and you can measure, and that resonates, because it, it goes along with

what you see and hear and feel everyday. Um, do you feel that the lack of application, do you, would you consider that a factor in how you approach mathematics? The lack of application?

Mark: Ah, that's a good question. That is a very good question. I really do think that, it may be, that may be part of it, but I, there are certain concepts in math that I just never, I don't know what it is, but like, one of my better friends, who I was at school with, and I got a better degree than him, which I love to remind him about, this guy's a genius, this guy's a bloody genius. He's so super bright, he's just brilliant. It just, math was just natural to him. It was almost like, it was like going for a walk, just like going for a walk. {am} It was natural, he loved it, I mean mentally he had that infused, but he loved getting to that number, which I never love getting to that number. So maybe there's, obviously something in both of those statements that A. it was like going for a walk, and B. he loved getting to that number. For me, it wasn't like going for a walk, it was a struggle, and, and getting to that number was like, what's the point? What have I got out of it? I've got a bloody number. {/am} {aa} Whereas physics, he was brilliant at physics as well, but physics was something that attracted me purely because of that grasp thing. That thing, that thing I can say, that car was going that fast and got there at this certain amount of time. It was real. So, that's what I liked about it, but it wasn't like going for a walk for me. {/aa} For him it was, it was just so natural.

Neil: So, so with all the, ya know, the background where in things like English, geography came easy, what was the transition into programming? What was the transition into that world? When did you start programming, when did you say, because it seems, things were kind of split for you, at what point did you cross over into, I mean, geography has a lot of science like you said, physics and geography, share, they go hand in hand, what, where, how did programming enter the picture?

Mark: That's again, a good question. I, I, the, there's, there's an interest element to this, there's a work element to this. When I first came to America, I mean, my life at work has been very wishy-washy it's fair to say, I've had loads of jobs, and I've never really grasped onto something. {aa} I had a love geography, a real passion for geography, physical geography. And when I left university, I left this purely, this pure world of academia. I mean, really, I should have stayed in the world of academia, I should have just … I was going to take a PhD, I didn't take it, but I could have done a PhD and probably stayed in that world because I was very happy and comfortable in that world. But, um, I didn't follow that path. {/aa} Um, so when I left university, I was always looking for that passionate thing that I was really interested in, and, and work was such a bunch of BS. I was so sort of naïve to going into work and think I was going to be at a university or training to do these really interesting things, ya know, I'm a bright guy, I've got a good degree, and so I realized that's actually not how the world works. Ya know, work has all of these down sides to it. And uh, so I never really, I really never alighted on anything that really anything more than just fluff. So, it was just fluff and nothing, nothing I could put my hands around. Ya know, like I was a, I spent awhile as a, a, a market researcher in England, and, ya know, there was something, somewhat interesting, ya know, you gather information, people filled out information about things they've bought and send it in, and they got like a little bit of money back, and this company I worked for compiled it, and then we'd publish reports and so, that, there was something, somewhat interest in that. But it really wasn't enough for me to, to be interested in. And then I came over here, and, and I didn't know what I was going to do when I

got over here, and it was a totally different world, so I, new country, a new everything, so I spent a lot of time, like, messing around. I was going to go to medical school, so I thought great, medical school is, is something you train for, something that's interesting, something that's tangible. Um, as I'm talking I'm learning about myself, to be fair. I've never really had these deep thoughts about this stuff. It's fair to say, actually that's what it was. I mean, there was even a point, I mean, I've known my wife for, since 1985 we started dating, so, we'd been together for a, for a long, long time. There was a point in our marriage where after about six or seven years, she said, I just don't know if I can stay with you, you, you just don't, you cannot, I want to have children one of these days, and you're just all over the place. Can you just … So it was, it was, I, I remember having the conversation with her that, ya know, you don't like looking at it, but I hate living in this because this is not … I want to do something that was like, when I was at university, she'd say to me, but you're not at university, this is the real world, ya know, you can't go back to that little encapsulated academic world. And, um, so I, I, I was going to go to medical school, I did all of the exams, and most of the premed things, and my grades were, were pretty good. I didn't have the passion though. I didn't really have passion for it. I was really going through the motions and I, um, so I went back to university, and I thought well, maybe I'll go back to geography, go back to my roots, go to another, go back to academia, and uh, do something there. {mot} And uh, I found out, when I got over here, I found out we'd got the Internet, and I thought, that's fantastic, the Internet's a wonderful thing. And then I, I, I was talking to a, the main, he was ultimately my advisor, when, when I did my masters and he introduced me to a side of GIS. I asked, so what is this GIS thing? And I got an idea about what it was, just enough to be dangerous, just enough to think that we could take this stuff that's on a desktop I don't fully understand, put it on the Internet, now that would be cool. {/mot} {ap} And I remember I sat down thinking, blimey, that's, that's really, you could have maps on the Internet, you could actually do all of these cool things with maps. You could, and I, I formulated this idea about what it could be in 1996 was when I had these thoughts, long, which is ten years before Google ever came out with their Google Maps and stuff they've done, long time ago. So I went in with a, yeah, all of the sudden I've got like a passion here, this is actually something I'm into, and um, so I went into this master's degree knowing exactly what I wanted to do, and I told my committee, I said, this is what I want to do, ya know, don't give me any BS about doing traffic, ya know, studies about, ya know, transportation systems, and they didn't do much physical geography as well out there, so I was going to be pushed into human geography route, which I didn't want to do. So, I knew what I wanted to do, I told them what I wanted to do. They tried to tell me not to do that and I said, I don't care what you say, I'm going to do this. So it wasn't really, it was more, programming was part of the whole if you look at it like me. Programming, it was a programming master's degree. Uh, but it was, it really, I'd found something that I thought, this could be really cool, I found, I'd found something I felt quite passionate about, and now, it, it was my, it was my thinking. I hadn't read this in a book, I thought that would be a good idea, I'd thought, that would be a great idea, and, I found uh, at the time, there was some open source things out there, and there was also company, which is a big company in the GIS world, ESRI, and they'd just come out with a product, a really crude product to do that kind of work, so it was really coming circle. In the mid-1997's they'd released this product, and I was looking around for a good tool to do this work, so it really worked out beautifully. {/ap} {mot} The forest service got wind of what I was doing, and they paid me to do a piece or work for them which ended up being my master's degree. So it was all, it was a really good scenario. {/mot} {aa} It was great, I didn't particularly care for my academic

experience at the [university name removed], it was just, I was older, and it was, political, and it was very, it wasn't like my first year at, my experience when I was younger. It was a disappointing experience to be fair. {/aa} So that's how I got into programming. {mot} But, that, I, I actually worked, when I was at the university, I was working with the assistant administrator as well, so I was programming degree, I was working as the assistant to the administrator. Um, and then I went to work in California as a programmer there, and really a jack-of-all-trades. I did a bit of, I did lots of sort of PHP, I did some Java, I did, with a really bright group of people, came back, worked for the forest service, they, they, when I got back they, I told them I was back and they said, Oh, come, come and do some more map work with us. So, I worked for them for probably, actually it was a contracting company, it wasn't directly with the forest service. Um, I worked there for probably five years, just basically building maps, and learning more about Flex and Flash and … {/mot}

Neil: So, did you immediately start programming maps? Were you learning in the classroom setting, mostly by yourself, before the program? … As soon as you realized that this is what you wanted to do, how did that unfold, cuz, so how did you go from not knowing programming, to programming maps?

Mark: It went, the. I mean, in many ways. I, I, I look back on it and thought, think, now, now it's kind of brave. But, I was also, I was also flapping around wondering what to do and I suddenly found great, this is as an original idea as I've got, the best one in my life. No. It was a good idea, and I've, I've now seen what my idea's really become without me, ya know, other people have certainly thought this is a good idea. But in 1996-97, it was, it was, had these ideas about what Google Maps is now, without ever having seen Google Maps. {ap} So, um, but answering your question, the, the, the Geography Department at the [university name removed] had yet to twig that computing was going to be a huge part of the, of all experiences, not just computer science. Ya know, if you're an economist, or if you're a geographer, if you're, if you're … you're doing, computers become a part of that. Programming is a tool within that, so you're not just using software, potentially you're writing software. They hadn't really got on board with that. They had scripting classes that they did internally, but I made the point to them that you need to train people to be programmers, you can't just expect people to just sit around in the Geography Department and program, you probably need to get them into computer science. And, and, there was, you know, debate about that, and in the end I said, ya know, bullocks. You know, I'm going in to the Computer Science Department and I'm going to take some no-credit classes up there, and uh, just see if I can program. I had no idea what programming was about, I had not a clue. Then, um, the class I took was, I began computer science with not offering. Computer Science was only offering classes for computer scientists, so that world, I think that world must now have changed. Where other disciplines use programming either to teach programming either internally or go into the Computer Science Department. Like, ya know, like computer science for non-computer scientists. {/ap}

Neil: Yeah, yeah.

Mark: There really we're those classes, so I actually took a weeding-out class for guys who wanted to do computer science as a degree, and it was a C++, it was a two part C++. {ap} {se} And it did, destroyed me. It was so difficult, I had no idea what I was doing, I just didn't know

what I was doing. And uh, I took the classes, I don't, I don't even know what grade I got in the classes, probably a poor one, but it was a real head-bang class, I mean, some of the concepts were like, oh my gosh, this is like, whoa. And uh, I, I, I picked up some stuff from that, but ultimately the work I did was a Visual Basic class, which was, you know, easier, easier, for me to work in than learning all of this tough C++. {/se} But the C++ stuff was, ya know, sort of flame to the fire. So at the time I hated it, but looking back at it, it probably wasn't a terrible thing. So, it was Visual Basic. So, really most of my, my, my work has been on my own. {se} So, I never regarded myself as a computer programmer, ever. Ever, ever, ever, and I never will. I'm not a computer programmer. I am a, an upstart who is interested into the world of computer science and working with people like [name removed] and [name removed], proper people. I don't regard myself as a proper computer programmer. I can get by, I can do it, I love doing it, but, does it, is it like walking down the street? It is not like walking down the street, it's like me, when I did physics, I like it. It's tangible, it's real, I get something out of the end of it. But it doesn't come as natural to me as it does to, for example, to [name removed]. {/ap} {/se} My, my mate who was a genius at college, who I got a better degree than …

Neil: Yeah.

Mark: … said that twice now. I've got to email him again to remind him. My brother does the same thing, because my brother isn't nearly as intelligent as him. Um, he's a, he's a computer scientist, and he's brilliant, he's just brilliant.

Neil: Do you think, since you started to programming, do you think that's changed anything about the way you've approached like problem solving?

Mark: Yes, I do, yeah. {pam} And I, I think I wrote on the thing online that, I've never really thought about, I think if you're going to be a really top programmer, you have to be very mathematical; you've got to think in that mathematical way. Your brain's got to be wired that way. They're the best. People like me, are sort of pretenders to the throne, who will never be the best kind of accept that, but, um, but um, lost my train of thought. But we can kinda make do in that world. {/pam} {ap} But, I've, I've always thought of programming as an exercise in logic rather than a mathematical exercise. I've always thought of it as something you know, you, like, my wife, I try to describe it to my wife and she, I said it's like playing chess. {/ap} {ps} You try to look moves ahead and think like, if I do that, and I do that, and I do that, and I do this, what's that going to end up with? Huh. That. So, I, I kind of regarded it as a logic exercise and yes, I think being a programmer, I'm, I think in a more logical way, I think. I'm certainly, it's subtle, but I think I probably have become, my thinking's more that way inclined. {/ps}

Neil: So, considering that you do program now, but describe yourself as a non-programmer, do people assume you're good at math?

Mark: {am} Do … yes. Yes. I'm sure, I'm sure they do. {/am}

Neil: Um, how do you feel about that? How do you, how do you … if, if, if I were to say you were a computer programmer, you must be good at math. Um, do you feel, knowing what

you've told about not connecting really with math previously, has that changed at all since you've been programming?

Mark: Well, have my math skills changed? Or has my …

Neil: Or … yeah.

Mark: I don't know.

Neil: Your relationship with math.

Mark: I don't know, don't really know. I don't think so, to be honest with you. I don't think so. Um, I mean I did actually do a, well, I got over here, one of my premeds was to do a, some algebra classes and uh, actually I did more than algebra, but uh, I did a bunch of math classes which I wasn't looking forward to, but the, the, the people that taught me, this one guy in particular, I had, I had like a load of eurekas during this class. Ahh! So, that's … ahh! So, some of these fundamental things I never learned as a kid were explained to me and I went, man, this stuffs not so difficult. So, I don't know, I don't know how to answer your question in terms of do I think my math skills have improved … maybe. My, I, I, I think I think in a more logical way now. Whether that translates into math skills, I don't know. But, I think I, I think I'm more logical in the way that I approach things. But, I don't know about math.

Neil: Do, do you have to use math frequently? Do you find yourself using math?

Mark: Not very often. Sometimes in programming exercises I've used math a little bit, but not a great deal.

Neil: Um, have you ever had, encountered a problem where you've actually had to go learn a mathematical concept to fix a programming? …

Mark: No. No.

Neil: So, for your day to day, math is pretty separate from your …

Mark: Yeah, very separate.

Neil: … from programming?

Mark: Yep.

Neil: Um, so at this stage with programming, what do you see yourself, how do you see yourself playing the role of continuing to program? Are you planning on continuing to program? How does that? …

Mark: Yeah, I, I … that's a very good question.

Neil: Considering where you've been, where do you want to go with it?

Mark: Yeah. I, cuz, I, I, I … yeah. That's something that I have been thinking about every single day. I mean, I've been a programmer now for ten years, and I've always felt that I was a fish out of water, I don't feel this is a natural place to be. I feel like, ya know, like I write, I wrote a, I've written, um, just in the last three months actually, I've got, Adobe just published an article by me. Adobe developers sent, and there's um, there's a, there's a magazine called *Directions Magazine*, and uh, a geographic magazine, it's online actually. They just published an article by me and um, there's an Australian actual physical publication called *Positions Magazine* which is online and physical, touchy-feely. They're about to, that's about to be released in about three or four days, and I've got another one coming out in, it was called *GIS World*, which is another physical magazine and on the web magazine, and I've got another location, *Business Locations Magazine*, which is another physical and, and uh digital. Um, and they're all, they're, they're all explanations of things that technically you could go into, you could talk to … I've got another really good friend, a guy who's an assistant administrator up at the [university name removed], he's, he's another genius, physics, math, fantastic. He's so super bright, I tell him, I actually tell him, I hate your guts cuz you're so bright. You know, why am I so thick whereas you? … He's really, he's a lovely guy, I love him. Um, but, um, what was I going to tell you about [name removed]? There was a point to my, I lost my train of thought again. Um, physics, math, programming … oh, yeah. He, when I was at the [university name removed], I taught a basic programming class, a basic computing class, and I called [name removed] in to talk about a certain part of this course. Nobody understood a word he spoke, he said. And my, my mate who lives in Vancouver, guy, my mate from school, I'm not convinced, he probably could do a good job explaining technical things, but usually there's a disconnect between a programmer and a business person, or someone who doesn't live in that world, and I'm always interested by the connection between the two, because it, I've done a lot of jobs where by, you have business people and you have this programming team, and there's no communication between to the two, and if there is, it's garbage. No one understands what the other one is talking about, particularly from the technical perspective. And I've worked in so many situations where I've asked people questions and they can't explain it to me in understandable that I understand, and I'm reasonably technical now. And I'm like, you can't, you can't explain this to me, ya know, and they're so down in the weeds, they're so down in the weeds with all of the details, and thinking in this mathematical, well not mathematical, this, this analytical programming way. They can't step away from it, and look beyond that detail, and therefore their explanations are untranslated. They live in that world and that's their world. There's no translation. And I think that's a big problem in business actually. Um, in the world of work, cuz I've seen, I've seen projects go south, not run as well as they could be because there is that disconnection between we need this done, and then the technical people talking about how they can get it done. So, in answer to your question, I'm very interested in technology, I, I'm, I'm, I'm ever a student of things, ya know like. Like, geography, I was always fascinated with geography because it was tangible, it was things you could see, and it was like, why've you got to wake up, and why is that? I've all of the sudden it's there now. And all of those kind of crazy questions that you can see and that you can find the answers to in physics. Programming is kind of something like that for me, I think. I find it very interesting, it's very tangible. But, in terms of the world of business which, ya know, living out of the academic world now, I'm still thinking of academia. In the real world, somebody like me, I think, and I'm still arriving at these conclusions, somebody like me,

actually this job has been something that's struck me in those terms, because, um, when you sit somewhere in between, you have to be able to talk the language of both. And if you only talk one or the other, it doesn't work so well, you need people who can talk both. And I don't, I actually would be more inclined to be on the more business side, not, not, not business per se, but what those skills are there. You know, I'm very social, I'm very, you know, I can talk for America. And uh, and then there's the other side of it, which is the geeky, nerdy people.

Interruption

Mark: Um, so yeah. That's probably the place, I'm beginning to think, my wife tells me all of the time, she said, why did you sit in front of the screen for ten hours a day, and then you just spend your time talking to geeky people. These two, in particular, these two are exceptions, they're, they're not like that at all. There's a lot of ya know, there's a lot of people who live in that like, math nerds, and therefore they're computer nerds, and they only talk their own little language and they play video games all day and that kind of group. My wife said, you're not, not that type of person. You're a, you're a talker and a social person, and so I live, I live with those guys, feeling, feeling like I'm in the wrong place, but then trying to find something where is the place where I sit. I think that probably the role that I probably foresee myself taking is to translate that kind of stuff. Writing these articles, for example, I got some feedback from one of these magazines that uh, the editor of the magazine, and he said, it was challenging for us to walk through your transcript to, just to, just to, we don't usually work with technical things, and it really was not very technical. It was really not very technical and it really struck me, I thought, well my gosh, you know, if this is something that somebody who is, who's an editor of a, a, I mean, it's a GIS magazine, and they're dealing with, there's technical things within it, if this guy is saying how much he enjoyed the technical challenge of walking through my manuscript, what would it be like if he walked, spoke to one of these GIS people that are programmers. I mean, they'd be talking gibberish, complete gibberish. So again, it struck me how far distant programmers are and other people are. And the mystique about a programmer, you're a nice guy, you must be really bright, and so when you say to me, when people ask me if I must be good at math, people perceive that if you're a programmer that you'd be bright. There is this perception that you have to be really bright to be a programmer, and I, I break that exception.

Neil: Do you think that learning to program let you understand that group more? Like the way they function, like, not feeling necessarily part of them, but like, if you understand them, um, do you feel that learning to program brought an under, more of an understanding of that world, or how that world works? Cuz you've separated, and I know exactly what you mean by separating the two types of people in this scenario, um, and that now you find yourself, you can be in the middle.

Mark: Yeah.

Neil: Um, let me think of how to phrase this. Do you feel that learning to program was what was able to put you in the middle? I mean, I realize …

Mark: Oh no question. No, no question, yeah. I mean learning to program was really to understand these guys' world. And again, I feel I'm an outsider, ya know, jumping in and

pretending, really. I really do feel like that a lot. Cuz the more in depth I get, the more in detail I get, I get in the weeds. I mean, these guys love being in the weeds. I, I, I like being in the weeds as well, but deep down, trying to understand these deep things, but um, a part of me, actually somebody interviewed me the last week, I had somebody call me out of the blue, said can interview you, this is a job interview, I'm not looking for a job, and well, the world certainly. And she said, what are your strengths? And I said, my strength is, I hadn't actually planned ya know, I hadn't like planned this interview. She said, strengths and weaknesses, as they tend to do. And I've never thought about this before, I mean, I have historically like planned a good answer what my strengths are, but off the top of my head, I said my strength is, I am good at looking at the big picture. And, and I am good looking at the big picture. With a technical understanding, I realize I'm good at stepping away, out of the weeds. I can talk to these guys and say to them, you know, I don't quite know what you're saying, I think I know what you're talking about, but just explain it a little bit better. Oh, I've got it, ok. Good. How does that translate into what we're trying to do here. And if they go back into the weeds, I can pull them out and way, yeah, that's all well and good, but, but how does this translate to this? Where are we going to get from there to here? And pull them away from their world and it does mean, I do feel that now, I can do that. I wouldn't, I wouldn't, I would feel too intimidated, and I wouldn't even know where to start if I'd not worked with these guys for ten years, but now I'm able to not feel, you always feel some soft of intimidated by these people, but not too intimidated, not too scared to say, stop right there. Tell me a little bit more. Oh, ok, got it. And to actually pull them out of the weeds and say, ok, what we're trying to do here, how are we going to get to that point in three days? Are we going to be able to do this and this and this and well, so, pulling all of those pieces together and sometimes they can't see. And certainly the business people are going, what the crap? What is, ya know, that, that's the place where I probably see myself fitting going forward. But, what that job, I have no idea because I'm still programming. When I leave this job, I'll go straight into being a programmer again. I'm feeling like an imposter, but, that's where I see myself going to, so, yeah. In answer to your question, that's me.

Neil: Another, another programming … oh, to be the person who straddles …

Mark: I think, yeah. I think probably my role would be some kind of straddling thing, some kind of a, I mean, I, I, I was the architect on this as well, so I have the big picture on this stuff, and trying to pull the pieces that you guys are thinking about, the pieces that I knew about, and that was kind of an interesting job, but I don't see myself doing that. That's still lives very much, tech-land, ya know, pure tech-land. I see myself somewhere living outside of that, ya know, writing articles about technical things, explaining technical things. I don't know, somewhere …

Neil: Yeah. I totally. Yeah, yeah. I can relate on that whole aspect on any level. Um, do you, so I guess, maybe a wrapping up with a, um, bringing it back a little to math, do you, so you, so you noticed that people perceive programmers as mathematicians …

Mark: Uh huh.

Neil: … would you say, and we kind of touched upon this earlier, would you say, now that you now how to program, that that's benefitted, um, yeah, I guess we talked about this, the benefit, the relationship with math, the benefit and relationship with math. Has there ever been a new,

when you've encountered math now, where programming has helped, as opposed to math helping programming? Have you found that because you've learned to program, that something was a little easier? Um, kind of tying in perhaps about if you mentioned that physics is applicable. Programming is very applicable. You can do something to get from point A to B, as opposed to just arriving at a number. You get a result, a tangible result.

Mark: I, I can't see, as, as we sit here talking, I can't think of the direct relationship between the two. My mind always goes back to logic. I can't actually tie them together. I think, without having done the physics, probably, my programming skills would be, I probably wouldn't be a programmer. But, maybe I would be a programmer, I don't really know. I can't really answer, cuz I can't really think of the relationship between the two, but there must be, otherwise. But I, I, I'm so sure it's the way your brain is wired, I think there's there's, an actual sitting down and doing math in a classroom, and then being a programmer, they, they live in two separate worlds. There is a little bit of math that applies in, in programming. Every once and a while, like if you're a game programmer, there's loads of physics, loads of math, tons of that in there. That's, yes, definitely. In, in what we do here, in most average programming, you don't really get to touch calculations. [Name removed] has done a bit of it, I've done some of it with like panning and zooming stuff in maps, but I really don't do any straight classroom math in programming. But, I do really think that, that, the, the key to people that are programmers are, their brain is wired, they've got, they've got a mathematically wired brain. Which doesn't mean you solve problems, you just have a brain that's wired to understand whatever those core pieces that make up math is. And that's what a programmer is. But even though there's not a direct relationship, I think it's brain wiring that makes you a good programmer. I, I'm trying, I'm trying to think of a good analogy, with, with, with. I can't really think of a good one. Like I was trying to say about English, that you can see words and that you can read, you've got like an English brain so to speak, and you can actually see something, some people can read and learn, and learn all the rules, but they'll never quite see it. I can't actually think of how that translates to not writing and, ya know, in a, in, in, in, in the real world how having that skill applies to something slightly different. But it must do. But my view on programming is just that, is the brain wiring, not so much what you actually do in a classroom. So.

Neil: The logistics.

Mark: Yeah.

Neil: Ok.

Mark: Good?

Neil: Yes.

Appendix I: Coded Interview: Andrew

Neil: Tell me about your general academic experience. How did you feel you performed academically?

Andrew: {se} Well, I feel I performed, uh, rather well. Uh, I was able to, uh, get all my classes done, um, during my general classes. Uh, I believe my grade point average was, um, pretty good, so I think, I think for me the general education was … it seemed pretty easy to me. {/se}

Neil: What was your favorite subject?

Andrew: For general?

Neil: Um, for anything. Anything in school, like high school, early college. What was your favorite subjects?

Andrew: {aa} Um, I'd have to say, probably favorite subject mostly had to do with the teacher and the class was American Heritage. {/aa}

Neil: And, so why? I mean you said because of the teacher.

Andrew: Well, it, he, ya know, it was done by [name removed] and he did, um, really well with his presentation and the way he just kind of delivered the subject.

Neil: So, was …

Andrew: Entertaining. {aa} He was entertaining. {/aa}

Neil: So was history and those kinds of things normally your favorite subjects, even in like high school?

Andrew: {aa} Not too much, no. I mean I liked them, but it wasn't that they were never really my subjects that I really liked, really. {/aa}

Neil: What subjects did you do, what subjects did you do the best in, would you say?

Andrew: {se} {am} Uh, I would probably say, uh, like in high school, and probably early, it was just sciences and probably math class. {/am} {/se}

Neil: What was your least favorite, and why?

Andrew: {aa} English was probably my least favorite, um, mostly because I didn't like writing and I didn't want to have to worry about that kind of sentence structure and stuff. {/aa}

Neil: Um, what else did you like to do outside of academics? What hobbies?

Andrew: Um, I liked to, you know, I played a lot of computer games, um, hung out with friends and um, ya know, movies, and, I did a lot of reading. I liked to read a lot.

Neil: Um, so how'd you learn to program?

Andrew: {mot} Um, through various ways. Uh, took, ya know, classes when in college and a lot of the stuff that I do now is I either pick it up from a book or as I'm into an issue, I will, ya know, research it online and find various ways to do it and then kinda combine them all or just see how they did and do my own. {/mot}

Neil: So, when you first learned to program, was it a formal or informal setting? Was it like, in school?

Andrew: I believe it was formal. I believe it was a classroom was my …

Neil: So, you began with, um, your first introduction, was it like the whole basic framework of a step-by-step in-class how to program? Um …

Andrew: Yeah.

Neil: Like what was the first type, the first uh, um …

Andrew: Like language, or …

Neil: Yeah, what was the first language, what was the first concepts you began learning in programming?

Andrew: It was, you know, the first language was Java, so it started off with object-oriented programming, and that's kinda what the concepts, just to learn the different, um, functionalities of the object-oriented programming.

Neil: And how old were you?

Andrew: I believe, it would be twenty-one.

Neil: So, what was the first kinds of programs you created in the Java class?

Andrew: Um, I could remember basic ones. Uh, um, basic ones were, respond-ya know, writing questions and receiving all these responses from inputs, from, ya know, from people. You know, kinda like I guess it would be kind of a survey type where you ask question and receive a response and then you reprint their response back. That might be one of my first types of programming.

Neil: So what, what in programming got you, or whether it was by choice or just because that was along the career, what kept you learning programming? Why did you continue to learn it?

Andrew: {mot} {ap} Um, the challenge, I think. Uh, working with it, uh, I always like to, ya know, think that a person would like programming if they liked to … the type of person that will spend hours trying to do something only to get a single word to print on a screen, and they think they've accomplished the world. But, it's, it's … the thought of working really hard and then seeing, ya know, something happen, right there, right when you complete it. That is really good, and uh, the mind challenges. Ya know, the different ways to do things. {/ap} {/mot}

Neil: So, where did you go on from that? From that Java class, where did you, what was kind of a quick summary of the progression of your programming experience?

Andrew: {mot} Um, I went to, in my electronics classes that I did, I took some other programming classes, and did, kinda, ya know, a data base, um, instruction, and then, then went into actually, did some PHP programming for a class project, at the, for my senior project. Programmed, a couple other people and I programmed a website. {/mot}

Neil: And what, your senior project, and what was your degree in?

Andrew: What was that?

Neil: What was your degree in?

Andrew: It was, Electronic Information Technology. And what was your other question you said?

Neil: Oh no, that, yeah, what, the senior, what was, what degree was the senior project part of?

Andrew: Oh, ok.

Neil: So there was, I assume then there was a heavy, or somewhat involved in programming involved in that … uh …

Andrew: {mot} Yep.

Neil: degree.

Andrew: Yeah, we had uh, had to, we programmed some interactive stuff, and we also uh, wrote our, um, webspider, basically kinda like the Google search engine. {/mot} So …

Neil: And, ok. Um, math. Um, what, to you, what is mathematics, and what does it mean to be good at mathematics?

Andrew: {am} Um, well, mathematics is, what does it mean to me, uh. It … you need it, as far as I can tell, what it means to me is the, the basic operations you need to know a lot of basic things in life, uh, and also it helps through a lot of things, like, ya know, you've got to balance your checkbook, you've got to that type of things, you've got to get through uh, life … math um,

and then to be good at it, um, say, um, I mean there's a couple type. I mean, there are people that have to work really hard to get it, and then they, um can do the math, or there's people that just, I guess when I say good, I guess it'd be, it kinda comes easy to them. They don't really have to do much. Once they learn a concept, it's, it's there. {/am}

Neil: So, as a programmer, do people assume you're good at math?

Andrew: {am} Um, I haven't really found anybody like that assumes, just because I program that I'm good at math. {se} Um, I feel I am good at math, but I don't know if anybody really just assumes I am based on being a programmer. {/se} {/am}

Neil: So, how so, have you always felt you are good at math, even prior to learning programming?

Andrew: {se} {am} Yes. {/am} {/se}

Neil: Um, have you ever been programming and had some kind of mathematical challenge that stopped the programming process? Even …

Andrew: {map} {pam} I haven't as of yet. Um, like I said, like I've found most of the math that I do is … in programming is basic; it's addition, subtraction, multiplication, and division. {/pam} {/map}

Neil: And have you …

Andrew: Nothing really, nothing really fancy.

Neil: Have you taken um math courses at the same time you took programming courses?

Andrew: Yes.

Neil: So, were there ever times where you had to learn math concepts outside of class?

Andrew: Outside of programming classes?

Neil: Outside of any, outside of math classes. Like have you ever been faced with, at work with …

Andrew: {pam} Nope. No, I haven't. {/pam}

Neil: Um … so, do you have any experiences where you feel that math has influenced the way you've thought about programming? Or vice versa?

Andrew: {map} Um, I actually don't. I'm not sure if … like I said, other than everything I've done is kind of basic mathematical operations … I really haven't had any, at least that come to

mind. I mean, maybe in the background there's been influence, but not that I can distinguish. {/map}

Neil: And most of the math you've used has been pretty, pretty basic math?

Andrew: Yes.

Neil: What, what about … how do you feel that, now that you know how to program, knowledge of programming, how does that influence your attitude toward math? Understanding you've always had a good attitude towards math. How, how has your knowledge of programming influenced that attitude toward math, now?

Andrew: {pam} {am} Mm, I don't know if it's really influenced me much. {se} Um, like I said I feel like I'm good at math but I haven't, since out of getting out of college I don't know if I've really … like I said in the profession and the way my life is, don't think I've ever really had to use math other than like I said, the basics of, ya know, balancing a checkbook … or figuring our gas mileage or that kind of thing. {/se} So again, it's just the basics, so … for learning programming, I don't know if it's influenced me … uh, to learn more or anything in math. {/am} {/pam}

Neil: Um, do you feel that it could foreseeably influence one or the other, or if not, if not in your personal experience, the global application of the two subjects? How do you feel …

Andrew: Um … it could.

Neil: … other programmers …

Andrew: {pam} Um, there are … I can see, or think of some things that programming and math … you know, in higher math would be something that could influence each other. For the things that I do now, I don't, I don't see it much. Um, like for, there are things out there, application wise that have to do higher math for say, ya know, nuclear reactions, for instance … I'm sure programmers for that would have to have a higher understanding of math to get their, ya know, to get their software to work correctly and be accurate. {/pam} But …

Neil: So do you feel …

Andrew: {pam} I guess life-critical programs would, depending on what they do, might have to have influence with each other, with math and them. {/pam}

Neil: So, what kind of stuff do you program on a daily basis?

Andrew: I program web applications and applications to allow other employees in my company to get their job done, and kind of like accounting applications or purchasing, um, shopping cart kind of applications. Um, I then do a lot of reports coming from databases so they can see all their information they need to know.

Neil: Um, do you feel that there's any shared concepts between the two? Any kind of shared, yeah, shared concepts?

Andrew: {sc} Well, there's always, like I said, the basics cuz as in what computers and programming is, I mean, is just math, addition of, ya know, *1*'s and *0*'s. I mean, so the basics of, ya know, you're adding, you're subtracting, um, is, the concepts are like, usually, closely tied together. {/sc}

Neil: So, what are your professional goals? And how has learning to program shaped these goals?

Andrew: {mot} {ap} Um, well my professional goals … is I'll continue on programming, I uh, also want to look into, uh, again, furthering my education, uh, where I work so that I can, uh, further help with my company's goals. And so, with programming, um, with the side that I do for programming, I think I'd be able to help, uh, my company with their goals and what they do. {/ap} {/mot}

Neil: Do you view math as playing a role in these goals?

Andrew: {pam} {mot} {am} {sc} Uh, actually, with the one goal I'm doing, yes. I will probably … math will probably be a little bit higher, as I will actually try to get work in combining, specifically combining programming with higher functionality of science and data manipulation type thing. {/sc} {/am} {/mot} {/pam}

Neil: So, um, you feel with the data manipulation that math could probably prove more useful than it might in the web apps you are currently designing?

Andrew: {pam} {am} Yeah. {/am} {/pam}

Neil: Excellent. Um, I think that's it. Thank you.

Andrew: Thank you.

Appendix J: Coded Interview: Scott

Neil: Tell me about your general academic experience.

Scott: Overall?

Neil: Overall. How did you perform academically? What was your favorite subjects?

Scott: {se} Um, overall I performed well. Um, and, most often I was bored. {/se}

Scott: busy work or standardized tests or who knows what else. {aa} Um, areas that I particularly enjoyed though, uh, math, sciences, music, um, equal doses of natural sciences and like hard science. {am} Um, from a math perspective, I, I enjoyed learning it, um, but it in a lot of cases that's where I felt like there was the busy work. {/aa} In fact I did a, I did sort of a social experiment my senior year of high school where in our BC calculus class, I wouldn't turn in, wouldn't even do the majority of the homework, and then I would get like higher, the higher grades in the class on the exams, and, try to show my teacher that I didn't need to do the homework to, to uh, be successful. And he said that while it was true that I didn't need to do the homework to be successful at math, I did need to do the homework to be successful in his class. And so, I got a bad grade. {/am}

Neil: Um, so, but did you like doing math?

Scott: {am} I enjoyed it. {ps} It makes sense like doing it, it makes sense, and so, it's problem solving, like, I'm the son of an engineer and so, like all growing up, ya know, I was like pre-programmed, or I was trained to solve problems, right? So, everything that I would encounter, I would look at it and I would try and understand what's happening and then try and fix it, or ya know, finish it, improve it, whatever. {/ps} {/am}

Neil: So what were, is your favorite subjects and your least favorite?

Scott: Um …

Neil: Or area of subjects.

Scott: {aa} I really, really liked uh, chemistry and biology were probably my most favorite subjects. Um, my least favorite subject, I'm trying to think because I'm sure I've put some kind of mental blocks to not remember. I don't know. I mean, I probably relatively less favorite would be like English or literature, but I still enjoyed English and literature, right? There weren't subjects that I just completely abhorred. {/aa}

Neil: Hated or afraid of.

Scott: Right, no. {aa} I, I enjoyed everything. And in fact, when I got to college, that was my bigger problem, you know, you hear people who say, I just, you know, I just can't find my "voi",

ya know, like your path, like I just don't know what I want to do, and for me it was less that I couldn't find something that I really was attracted to, but that everything that I tried I enjoyed reasonably well and I said well, I could see myself doing this like forever, right? So, in general I enjoy learning for the sake of learning any subject. {/aa} So.

Neil: What hobbies, what other interests did you have outside of academics? Hobbies?

Scott: Um, going back how far, like high school, college?

Neil: High school, elementary, high school, as you got into college. So, pre-professional.

Scott: So, right. Um, I spent a lot of time, like growing up I spent a lot of time playing soccer all the way through high school. Um, I did, and increasingly from about the time I was maybe like eleven or twelve, until I got out of high school, I did a lot of music. Um, for some reason, I stopped doing music almost completely like, after my mission. Not right after, but like some time after I just stopped listened to a lot of music and doing any sort of music. Um, I'm trying to think of what other hobbies I had per se, because those were definitely the ones that took up the majority of my time. Cycling, skiing, swimming, I mean I was fairly active.

Neil: So, how'd you learn to program?

Scott: Um, my, my uncle, who worked at the [university name removed] which had some of the first like, desktop computers in the state, um, we would go up there to visit him and we would see the computers, and that was kind of my first exposure to like computing or personal computers. And then we got at our house we got a Texas Instruments, um, I'm trying to remember the model …

Neil: TA994A?

Scott: I think. I mean it was right around that same vintage. {mot} And we got one of those and we hooked it up to a monochrome television, and I used it mostly for playing, ya know, Atari-style games on it, and Hunt the Wampus, and, and . . .

Neil: Munchman?

Scott: … some other stuff like that right? Which they were great games, and I just, ya know, had a grand time doing that, and then my, my uncle brought over some programming magazines, and they were basic, they were, they had like programs in it that you could like program into it. {/mot}

Neil: How old, how old were you? Like ten, nine, eight?

Scott: Yeah, probably like eight, nine, ten, someplace in there. {ap} They had these programs in there, and they were all basic and there was one particularly, the first one that I ever tried to do. I mean we did some very simplistic programs, like he showed me how to do it, right? And I took this one and it seemed like it was really long because it was like five or six pages of code, right,

and I painstakingly went through and I transposed the whole thing, because they basically have it all typed out, for you, you just have to type it all in. I typed it all in, and it was, it was about ice caves, it was a game about ice caves, and the illustrations in the magazine make it look really, really cool. But the actual game was like a, a, like a role-playing game where there was like no graphics at all, right, it was just kind of like this choose your own adventure, like, directions, like you have to make your way through this like maze, right? And so I went to like all this work, and I thought is was going to be really cool, and I ran it, and it was like this command line game. But that was the first time that I like actually like, wrote, typed in, transposed a program, and started understanding kind of at a deeper level. From there I did some additional stuff, ya know, in that environment and all the way through like in high school, we had some, we had a computer lab with a bunch of Macs and some … {se} They had a programming class which was, by that point it was, you know, too, too simple for like, my understanding, but it was a lot of time, you know, in front of a computer, and like my first probably formal exposure to programming. {/se} You know, here's how you're supposed to do things, and the logic of it and stuff like that. {/ap}

Neil: Um …

Scott: {mot} I really learned to program on my own. {/mot} After, after like the Internet really hit, um, which happened while I was on my mission. Like before my mission I had used email with like Pine, and like IRC chat, and you know, a few people had like, pretty slow modems from home, but mostly it would be like, ya know, my uncle's office at the university. And while I was gone on my mission is when the Internet just exploded, so I came back, and everyone had access to the Internet now, and actually I guess my, my first, right before my mission was the first, or like one of the first semesters where you could register online at BYU. Before that it was all telephone and the catalogue, right? And it was right then that you didn't have to do it online, and most people still did it on the phone, but you could do it online. That was like the first or second semester at [university name removed], right? Then you come back after, like fast-forward two years later and like there's no more phone registration, it's all on the Internet.

Neil: So as you began, so basically around college age, cuz you're starting to go to college, you're getting more and more immersed into programming, how do you, um, do you think that affected the way you approached problem solving at all in those early stages?

Scott: {ps} I mean, I'm sure that it made me more systematic, and possibly more creative, I mean understanding that there's more than one way to, to ya know, solve a problem or to do different things than I would, I'm sure that it affected me, the way that I thought about it, the way that I attacked it. {/ps}

Neil: So, as you started learning programming, what was, what was the, you mentioned this on the survey that it was a mix of formal and informal. Um, what got that ball rolling? What were the first things that you started learning in programming, what were the first topics, or the reason to, to just jump into it?

Scott: {ap} I mean the thing that really sucked me in, cuz I had thought about, I mean, I had thought about like in high school I took programming classes and different things and it, it was interesting and I did stuff like that, but I wasn't just like totally hooked, and I thought about

doing something like that in college or as a career. But I didn't think it was like, not cool as in like geeky, but like I just didn't think that was a cool thing that I wanted to be like doing with most of my life, right? And, and it's like partially because it just, ya know, I had come up through like a lot of command line stuff, text editing, pretty basic, but ya know, gooey's when, I mean, I remember like with Windows 1.0 and then like 3.1, like my aunt and uncle got 3.1, and I spent the night at their house, and I stayed up like all night. Cuz Windows was really cool at Windows 3.1, right? {/ap}

Neil: 1.1 for groups.

Scott: Right. {ap} So, so I'm like sitting there just going through all of it, thinking this is amazing, right? But a lot of the programming was still, ya know, there was a lot of abstraction between like that and then like what actual programming was, right? So, I thought, oh, I don't know that I want to be like just in the code all day long. {mot} But with the Internet, what made it that much more exciting for me was how immediate the results were, right? There was just this wealth of information and you could learn and make changes and see it happening like, right away. So, ya know, that's where it really started getting interesting, was when I started building web pages and then the natural progression from static html to dynamic scripting languages, and then dabbling like in Perl and like the early PHP. Ya know, then moving into like more formal programming and getting into like object oriented architecture, and understanding, you know, the engineering behind a computer, right? So that's kind of like the natural progression. But the thing that really tipped it off was actually when the web became pervasive in like, '98. '97, '98, '99. {/mot} {/ap}

Neil: Um, going back with math, um, what do you think it means to be a good, good at mathematics? If someone's good at mathematics, what is mathematics?

Scott: {am} Um, I think I would describe being good at mathematics as … I guess, I mean in general, it's just understanding that things can be solved mathematically. And then having enough exposure to the different aspects of mathematics, that when you're presented with a situation, you know which area to go to to like solve the problem. {/am}

Neil: So, exposure.

Scott: I think part of it is exposure.

Neil: Knowing your options.

Scott: {am} Because, I mean, if you think about, you know, like calculus, even like algebra, average people, like regular, ya know, the average user of like life, probably doesn't need or even think or even realize that they can, that the algebra or the calculus is like all around them, you know? They take for granted that in all of the things they use, someone else already done the math for them. But, there are some people, the people that I think who are good at mathematics, are the ones who, ya know, they don't necessarily need it every day, but when they are presented with a situation, they know that, ya know, they can identify it as, this is a mathematical problem, and these are the tools that I know exist to solve it. {/am}

Neil: Um, so you, how to phrase this. Um, so sometimes maybe knowing the option even more so that knowing exactly how to do it at the …

Scott: I think so. {am} Because I think it's unrealistic for like regular people, meaning someone who's not like teaching or studying mathematics all day every day, I think it's unrealistic to expect them to remember everything. There is so much out there. They're not going to remember everything, but they're going to remember enough that they'll know, you know, in this situation, you know, I need to calculate the area under this curve. {/am}

Neil: Right.

Scott: {am} And they can kind of work through it from there or go get, ya know, go get a little refresher. Pull a book and figure it out, right? {/am}

Neil: Like they've been able to identify their problem.

Scott: Right. {am} Identify the problem. And I think part of it also is being able to understand, uh, if you have an example set of how to do something. I think you're good at mathematics if you can work through an example of something, and figure out or understand what's happening, and then go and apply that same principle elsewhere, right? And that's kind of actually. {/am} {ps} It's really related to like the Internet, and like programming. Like, if I get through the hiring process for like programmers, different things, um, you know, a skill that, that I think is critical now, is how well can they google, basically, right? {/ps}

Neil: I agree.

Scott: {ap} {am} And it's like, it's like, ok, you have a problem, and no one's going to come and teach you like how to solve it. So, the real skill is going out and finding the solution on your own, reading through a tutorial or something and then coming back and applying it to whatever problem you are trying to solve right? {/ap} {sc} And it's the same with math, it's like knowing the tools that are available, and how to effectively utilize those to solve your problems. {/am} {/sc}

Neil: Um, as a programmer, do people assume you are good at math?

Scott: {am} I think they probably do. I mean math and science, so synonymous. Computers are so scientific that, for like lay people, who don't understand like, computers, they think anyone who does probably understands math. {/am}

Neil: So, how do you see, now you already said you excelled and were really good at math all along, um, did you see any change in your math skills from before you learned to program?

Scott: Like how did my math skills, how did my exposure to programming change my math skills?

Neil: Or were there any time, was there any challenges in math that actually stopped you in your programming process?

Scott: {pam} {map} Um, I mean, stopped, probably not. Like I said, I think potentially you, I would run into like little like road blocks, but like I said, it's mostly uh, uh, remembering the specifics of something and having to go like figure it out, right? So if you're writing a program to calculate the area of a sphere, and you can't remember off the top of your head that it's … ya know… {/map} {/pam}

Neil: So, with that in mind … so, so if you're writing a program for that … has there been times when you were programming and you had to learn a new math concept outside of what you know because of something you were programming led you the other way around? Maybe …

Scott: {pam} {mot} Yes, I think that has definitely happened. Where you're trying to solve something programmatically, and you know that it can be solved mathematically, that that's, ya know, that there's often a faster way to solve this problem, if you had the, the better math. And so then you go and say, how can I, ya know, how can I solve this? {/mot} {/pam}

Neil: So it gave you a reason to learn a math concept outside of a formal environment.

Scott: {pam} {am} {sc} Well, and even like, I mean, we mostly think about math in like, base ten, right? But exposure to like, ya know, even like binary operations. I mean, that's, that's a whole other, like, field of, ya know, mathematics that I definitely would have never understood without, ya know, being involved with programming and computer hardware and stuff like that. {/sc} {/am} {/pam}

Neil: So under, so yeah, I'd like you to expand on that. How, yeah the understanding, so you're saying that understanding these certain computer programming aspects, increased your awareness. Like the binary. Was that a concept, um …

Scott: {sc} {pam} I mean, I understood the concept of like binary, like 1, 0, 2 state previously, but then, the whole concept of like taking two numbers represented in binary and then like adding and subtracting them, I'm sure that I never would have, ya know, even gone there without, uh, without having programming. {/sc} I mean, here's another example. Like, over this past holiday, I'm taking my brother-in-law, who's fourteen through a book about, it's uh, it's exposure to computer science, but the language that they're using is Python, okay? And so, we're going through data types and talking about it, and we have like a two-hour discussion about um, 32-bit limitations, like how to represent what's the largest integer you can represent with 32-bits. And then if you're you're using, ya know, if you're taking half of that away for the negative, and how come you can show one more integer on a negative side than you can on a positive? {sc} Ya know, and, and a few years ago, I don't even remember what we were doing, but we sat down and taught him how to do binary addition, and so, ya know, all of that's coming out of a programming perspective of, we're trying to learn about programming and then we're learning new things about math, that go into, like, the computer or into the language to make it happen. {/sc} {/pam}

Neil: So, in a situation like that, do you feel that, um, it has any influence of, um, the attitude towards math?

Scott: {am} Well, let's take my brother-in-law: super smart kid. Similarly I think he's bored at school and he doesn't enjoy doing math, but, if we're working through a problem set in this computer science book where he has to calculate the area of a sphere, then all of the sudden, it's, it's a lot more interesting than just an exercise to like calculate areas of spheres, right? {/am} {pam} He's writing a program, and this is what I explained to him, I said look, you don't actually have to know and remember the math, or do the math to calculate every sphere, but you have to understand it well enough to write a program to do it for you, right? I said that's how we're leveraging like the power of computers and programming is you don't have to then remember the math. You just know that it's there. You, you can look at it and figure out how it's working, but, you don't ever have to do it. {/pam} {am} The trick is, ya know, so many people, now, like I said, are accustomed to just having, having things work around them and do the work for them, that they're forgetting how it actually works underneath. And we're, like in danger of, ya know, of like losing that critical knowledge. {/am}

Neil: So, but, by putting your brother-in-law in that situation, he's also finding a, or would you agree that he's finding himself in a situation where he needs to apply something that may have previously been abstract and therefore not as useful in his daily life?

Scott: Yes.

Neil: Where he has. So obviously he's interested in learning the programming. Enough to learn the math to help the programming, right?

Scott: Right.

Neil: Previously he didn't want to learn the math.

Scott: {am} The math thing, it's kind of connecting the dots to how does this math apply to real life, for him, right? And it does make it that much more interesting for him. {/am}

Neil: So, as a global statement, how would you feel math and programming, I mean, you've mentioned it a few times in different ways, but, or how, how useful is programming to math?

Scott: How useful is programming to math?

Neil: You've talked a lot about the math being fundamental to programming, how has programming been useful to your mathematic life?

Scott: I mean, not to restate what we just hashed, but that is, it's connected the dots, right? {pam} {am} Programming has connected the math to real life, where as before it was fairly abstract. Even, ya know, even when it wasn't abstract, for whatever reason, it just didn't seem as apparent or as necessary. I mean it made, ya know, there may be some fundamental like, uh, link between being good at mathematics and having like a penchant for efficiency, right? I'm happy

to do the math if I'm writing a program that's gonna save me like work, or something down the line, right? I'm not happy to do that same math when I have to do it over and over and over on a series of like problems, that don't have anything to do with anything. They're just taking up my time, right? {/am} {/pam}

Neil: So replacing, it no longer turns into busy work, it turns into useful work.

Scott: Right. {pam} The programming is the useful application of the math that's yes, it's not busy work; it's not wasting my time. {/pam} It's …

Neil: It's more like a, it's more like an apprentice would learn something, seeing how something fits in the, in the long run.

Scott: Right.

Neil: How it gets you to the, to the end of your goal, as opposed to just being anything by itself. Ok, um, so what are your professional goals now? How do you, what, what are your professional goals?

Scott: Um, interestingly, I mean, I've always had extremely entrepreneurial underpinnings, right? {mot} And, along with zoology and computer science at university, I also studied business, and I really would like to combine, ya know, any or all of these things into building new business, that, similar to the explanation I gave for programming, but that, that actually does something good, right? I'm not interested in building a business that doesn't make the world a better place or allow people to do things more efficiently. But I would like to build a business that leverages technology either just in the information or Internet space, or possibly in the medical field, because I think that, uh, computers are under, they're integral, but they're, I think they're underutilized in the medical field. I think there are ways that we can leverage the technology to improve the health care and lower the cost of it, for everybody. {/mot}

Neil: You think math will be, play a role in that?

Scott: {mot} {am} Sure … it's all math. {/am} {/mot}

Neil: It's all …

Scott: {am} It's all math. {/am}

Neil: Um, alright then, thank you.