The Report Committee for Kevin Ng

Certifies that this is the approved version of the following report:

Exploring a LOGO Microworld: The first minutes

APPROVED BY

SUPERVISING COMMITTEE:

Supervisor: _____

Walter Stroup

_____

Anthony Petrosino

Exploring a LOGO Microworld: The first minutes

by

Kevin Ng, B.S.M.E.

Report

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Arts

The University of Texas at Austin

August, 2014

Dedication

This paper is dedicated to my kids. I hope the things I have learned over the last three years will make me a better father and teacher.

Acknowledgements

UTeach*Engineering* has given me a fluency in engineering that I could not have imagined as an engineering undergraduate. Thank you to everyone involved, especially everyone in MASEE Cohort 4. The last few years were extremely fun because of you. I would like to thank some people individually (in the order I met you):

Dr. Crawford. I thought you were a good professor back in 1999 (I still remember some of your stories), but I like how you have evolved to make engineering more accessible. Thanks for reminding everyone that engineers can be funny.

Theresa Dobbs. I did not know what I was getting into when I signed up for ESIT 2011. The only thing Cory told me was that you are awesome. It did not take me long to figure out what he meant. Thanks for taking such good care of us. Even Hisham appreciates you and he has trouble liking anything from Texas' flagship university.

Cheryl Farmer. I remember telling Hisham about your apology regarding state of the EYW curriculum in 2012. He shared my disbelief. We will sign-up for your "embarrassing" curriculum any day. Also, thank you for giving me the nicest introduction I have ever received.

Dr. Allen. The projects that you develop (fuel efficient car, earthquake, water heater) are absolutely incredible.

Dr. Marshall. You were the person who introduced me to educational theory during ESIT 2011. At the time, I did not fully understand the importance of it, but I certainly do now. Thank you.

Dr. Petrosino. Your curriculum history class is the only history class I have ever enjoyed. I think this says less about history than it does about how history is taught.

Dr. Stroup. Thank you for introducing us to Senge, Sterman, and Papert. My exploration of system dynamics and microworlds will not only benefit me, but my kids as well. Fight the good fight.

Exploring a LOGO Microworld: The first minutes

by

Kevin Ng, M.A.
The University of Texas at Austin, 2014
Supervisor: Walter Stroup

In his 1980 book, *Mindstorms*, Seymour Papert proposes using microworlds to help children learn mathematics like mathematicians. In a microworld like LOGO that is culturally rich in math, Papert claims that learning math can be as natural as learning French in France. Although the technology at the time was adequate, LOGO faltered due to improper implementation in the classroom. A newfound political interest in inquiry and computer literacy could breathe new life into Papert's vision. In contrast with the routinized approaches to introducing aspects of programming that, arguably, limited the trajectory for the implementation of programming in schools (Papert, 1980), this report explores what can and does happen in the first few minutes using a more open, student directed, approach to programming with high school physics students. A grounded theory approach led to connections with Vygotsky's Zone of Proximal Development.

Table of Contents

List of Tables

List of Figures

Chapter 1 - Introduction

The driving force for this study is to investigate the viability of computer microworlds to help students make sense of physics. Although mixing programming and physics may seem like a culture clash given their perceived domains in the virtual and physical world respectively, the two actually have a history of collaboration. Some of the earliest IBM punch card calculators, the precursors to the modern programmable computer, were used to model uranium implosions at Los Alamos. More recently, the demand for realism in video games has required game companies to employ physics engines to calculate trajectories, momentum, and shading. In this historical context, the lack of computer modeling in physics curriculums can be seen as a glaring omission. Instead, physics is often taught with a heavy emphasis on formulas and mathematical computation.

In his book *Mindstorms*, Papert dubs school math the "QWERTY of education". The QWERTY keyboard was originally designed with a very practical purpose: to minimize the chance of neighboring typebars clashing. Although current technology has rendered the typewriter obsolete, the QWERTY lives on due to network effects. Likewise, the methods used in school are a set of "historical accidents" that, although practical once upon a time, do not take advantage of the technology currently available (Papert, 1980). Instead of using a computer to force feed "indigestible" information at a faster rate, Papert suggests immersing kids in a culturally-rich computer environment like LOGO (Papert, 1980). By giving students agency and allowing them to externalize their intuition, students can behave like mathematicians (Lawler, 1997). Unfortunately, due to poor implementation at the classroom level, LOGO faltered and became a niche resource.

Recent changes may give microworlds like LOGO a second chance to flourish. In 2011, the National Science Foundation (NSF) proposed the Cyberlearning initiatives as a way to improve learning through the optimization of technology (National Science Foundation, 2011). The proposed activities include studying how people learn with technology, improving the use of technology to manage and share data on learning, and designing new learning technologies and environments (National Science Foundation, 2011). This proposal signals a shift in power back to the NSF, which lost influence and power to the Department of Education when the United States Congress enacted a standards-based education reform called the No Child Left Behind

Act of 2001 (NCLB). A reversal of power is significant because it also represents a reversal of educational philosophy. The NSF, as its name suggests, values science, which in its purest form is a heuristic, inefficient, and fluid form of learning. The Department of Education, on the other hand, values algorithmic teaching, success, and efficiency. While those values are not intrinsically bad, the high-stakes testing ecosystem created in the wake of NCLB unintentionally created a classroom culture that was ineffective (Ravitch, 2010). The results that were promised by NCLB have yet to materialize as of this writing (Bracey, 2003).

Legislators in seventeen states plus the District of Columbia have introduced legislation that will allow high school students to use computer science courses to satisfy their state's foreign language graduation requirement (Heitin, 2014), but LOGO itself is a reminder that effective implementation is not guaranteed. In contrast with the routinized approaches to introducing aspects of programming that, arguably, limited the trajectory for the implementation of programming in schools (Papert, 1980), this report explores what can and does happen in the first few minutes using a more open, student directed, approach to programming with high school physics students.

Chapter 2 - Literature Review

The primary design criterion of LOGO was to be "appropriable" to help reconcile student's relationship with math (Papert, 1980). Typical schools build systems of prerequisites that alienate students from powerful mathematical ideas and ultimately create mathematically illiterate graduates (Papert, 1980). Unfortunately, a mathematically-illiterate culture will not find any problems with this. "For most people, nothing is more natural than that the most advanced ideas in mathematics should be inaccessible to children" (Papert, 1980). In a microworld with a mathematically-rich culture, learning math becomes spontaneous. Students will "reconstruct knowledge in such a way that no great effort is needed to teach it" (Papert, 1980). Papert compares learning math in a LOGO microworld to a child learning French in France. A native speaker learns a word by encountering it in her environment, testing its meaning, and refining her understanding of the word until comfort with the culture is established. Likewise, a child can learn math by developing mathematical intuition and "debugging" or resolving conflicts with the microworld as she uses and tests it (Papert, 1980). This type of "abstraction" is critical to learning, but is often avoided even in problem-solving situations because it is "difficult" (Lawler, 1997).

Spontaneous learning neither eliminates the need for instruction nor minimizes the teacher. After all, even native French speakers need French instruction. Here, Vygotsky provides an additional dimension to the analogy. Like Papert, he believes children, driven by an intrinsic desire to understand their environment, develop mental models of these ideas (Vygotsky, 1987). Literacy is achieved via a systematic manipulation of objects in the environment (Vygotsky, 1987). This type of spontaneous learning has its limitations however. If asked to explain her pronunciation or sentence structure, the native French speaker (assuming she even understands the question) would probably say something to the effect of, "I don't know… I just know." How can someone show no awareness of the language she commands so effortlessly?

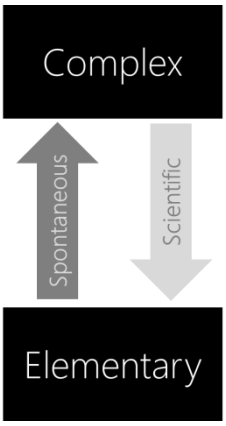


Figure 1: Two Types of Conceptual Development According to Vygotsky

Vygotsky's explanation involves a second developmental process that couples with the spontaneous. The two paths share the same endpoints but move in opposite directions as shown in Figure 1. The aforementioned spontaneous concept development starts with a child's encounter with a concrete (elementary) thing and transitions to an abstract (complex generalities) understanding after a long developmental process (Vygotsky, 1987). Scientific conceptual development, on the other hand, starts a child with an abstract understanding of the thing (complex generalities) before leading him to the actual thing itself (Vygotsky, 1987). The strength of one path is the weakness of the other (Vygotsky, 1987). Scientific conceptual development in the context of the prior French analogy would be similar to learning French as a foreign language. For example, foreign language students typically start with vocabulary, syntax, and pronunciation – ideas so abstract that they would be very difficult for students to learn spontaneously (Vygotsky, 1987) – and working "down" toward verbal fluency and pronunciation.

The interplay between the two paths forms the Zone of Proximal Development (ZPD). In utilizing the ZPD for instructional purposes, instead of focusing on the development that has already happen, the teacher focuses on development that has yet to happen (Kozulin, 2003). The bottom threshold of the ZPD is defined by problem solving that the student can do individually. The top threshold of the ZPD is defined by problem solving that the student cannot do. In the middle is the ZPD, which Vygotsky defines as the problem solving that the student can do with guidance from an adult or in collaboration with more capable peers

(Kozulin, 2003). When employed during the "sensitive period", instruction leads development just enough to unlock "a swath of developmental processes" (Vygotsky, 1987).

With their embedded complementarity framework, Stroup and Wilensky provide more insight into this interplay between the spontaneous and scientific paths, which they call agent-based and aggregate reasoning methods respectively. Students' reasoning is neither fully agent-based nor fully aggregate, but an interaction of the two working as complementary pairs (Stroup & Wilensky, 2014). Just as in the wave-particle duality theory of light, neither model alone sufficiently models the students learning. Whereas Vygotsky's ideas are framed in the context of childhood development, Stroup and Wilensky's ideas seem framed in the context of sense-making in a complex system. Although the difference may be nuanced, it makes the ideas of embedded complementarity much more relevant in areas like STEM, where making sense of a dynamic world is seen as a critical objective.

The physical world is not culturally poor in physics, so why are misconceptions in physics so prevalent and persistent (Clement, 1982)? Perhaps the dynamic complexity of the physical world makes it difficult for children to test their intuition. Dynamic complex systems are dynamic, tightly coupled, governed by feedback, nonlinear, history-dependent, self-organizing, adaptive, counterintuitive, policy resistant, and characterized by trade-offs (Sterman, 2000). Entire blocks of study in physics are devoted to objects in states of change and these blocks of study are strongly interwoven; it is almost impossible to talk about forces, energy and momentum in isolation. Calculus is often treated as a companion course to help students understand the non-linear relationships between variables. If that was not enough, several physics concepts are notoriously counter-intuitive (Clement, 1982). The current push in the physics community toward a single unifying theory of everything may reduce detail complexity (Sterman, 2000), but does nothing to reduce the dynamic complexity.

Research into bridging conceptual understanding and successful computation in physics has led to several terms to characterize the tacit nature of this activity, including blended processing (Kuo, 2012), abstraction (Chi, 1981), common sense (Sherin, 1999), transfer, "physical intuition" (Larkin, 1980) and the ability to "see" the math (Walkington, 2011). Looking through the lens of Vygotsky, perhaps the reason learning physics is so difficult is two-fold:

1. The dynamic complexity of the physical world limits students' ability to experience spontaneous conceptual development related to physics

2. The tradition of teaching physics with computation keeps the scientific conceptual development very abstract and makes it less tangible for students

In other words, the spontaneous development of the student may be too low to make the scientific instruction useful. The other possibility is that the scientific instruction is too abstract to lead spontaneous development. It is in this gap that a microworld like LOGO can serve as a "transitional object" that connects students to the powerful ideas that are not yet fully developed in their minds (Papert, 1980).

LOGO is not the first or only attempt to help students make these connections to powerful ideas in the classroom. InSight Maker (Foreman-Roe & Bellinger, 2013) made it possible for students to simulate and discuss complex system dynamics (Sterman, 2000). NetLogo adapted LOGO to help students experience agent-based computer modeling (Wilensky, 1999). HubNet made it possible to explore embedded complementarity though networked computers (Stroup & Wilensky, 2014). For this study, I decided to use an Introbuttons activity written by Dr. Walter Stroup for NetLogo (Stroup, 2008a). The NetLogo model library does include specific activities for physics, but as my introductory study, a more general approach seemed more appropriate.

Research question. As their initial engagement with programming (the first few minutes), can students in a high school physics class use the Introbuttons microworld (Stroup, 2008a) and aspects of the Introbuttons activity sequence (Stroup, 2008b) as a scaffolding environment to support an open ended introduction to programming? In this paper, we will discuss findings specific to the Introbuttons activity and then tie any conclusions back to a physics-specific classroom implementation.

Chapter 3 - Methods

The experiment was conducted in a classroom of thirteen high-school seniors in a large suburban senior high school. The Introbuttons activity served as a seamless substitute in-lieu of the programming unit that was originally planned. Because the experiment took place at the end of the school year, there was only time for one experiment, so the priority was to collect and store as much information as possible so that it would be easy to review and analyze at a later time. Although the students' source code was initially presumed to be the primary source of value, the students' dialogue and interactions with the computer ended-up being most valuable.

Step 1: Group students in pairs. Although the classroom had enough computers for most kids to work individually, the students were asked to work in pairs for two reasons. First and most importantly, it fostered student collaboration and dialogue. Without dialogue, it would have been significantly more difficult to determine what the students were thinking as they explored the Introbuttons activity. Second, it allowed the recording responsibilities to be delegated to one person in each group. Without any recording responsibilities, the researcher was free to roam, observe, and interact with the groups during the experiment. These general observations were foundational in making an initial theory during analysis.

Step 2: Distribute the software to each group. The Introbuttons activity (Stroup, 2008a) runs on NetLogo, a freely distributed version of LOGO. Acquiring the software is easy, but installing and running software on school computers can be a challenge. Because the network would not allow the NetLogo download, NetLogo was first copied onto multiple USB flash drives for later distribution. Both Windows and Mac versions were copied to accommodate students who brought their own laptops. Running NetLogo on the school computers required a teacher login. The Introbuttons activity used in conjunction with NetLogo was designed for version 4.1.3, but the only version that worked on Mac was 5.0.3 (Stroup, 2008a). We did not encounter any compatibility issues.

Step 3: Distribute a recording device to each group. The student who was not operating the computer was asked to record the dialogue audio and the computer monitor. The students were encouraged to swap responsibilities periodically. After the experiment started, some students discovered screen-casting software, which had some advantages. The video quality of screen-casting was superior but it required an external microphones and additional post-processing time to convert into the usable .mp4 format.

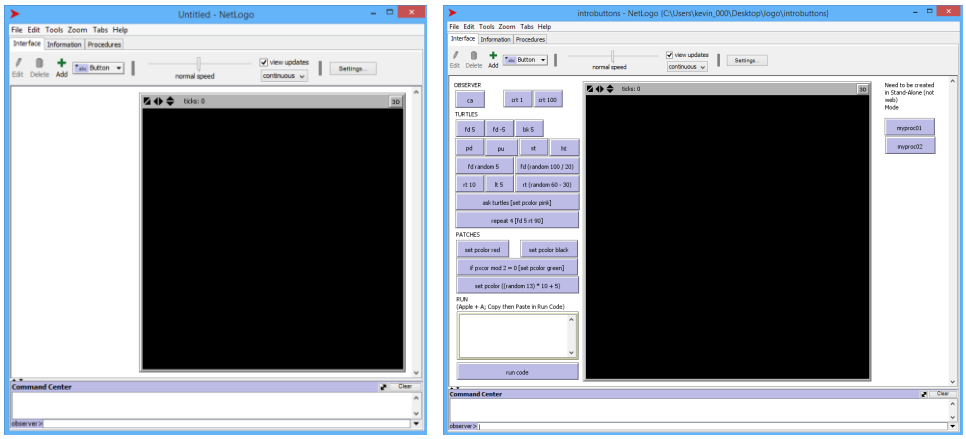


Figure 2: NetLogo (left) running the Introbuttons activity (right)

Step 4: Ask students to start the Introbuttons activity. NetLogo will not start the Introbuttons activity automatically (Stroup, 2008a). After starting NetLogo, students were required to open the Introbuttons activity files manually from the File menu. Figure 2 shows the difference between the two interfaces. The Introbuttons activity includes two rows of buttons on either size of the black "space". On the left side, the buttons are physical instances of different turtle commands and are labeled accordingly. Using these buttons, students can clear the screen, create turtles, control turtle movements, and change the color of the "space". On the bottom of this column is a caption box which displays the source code produced by the Introbuttons. Immediately below the caption box is the **run code** button, which prompts the user to enter a command. On the right side, the buttons are physical instances of procedures. The procedures are initially empty so the buttons do nothing, but once working source code is added, the buttons can be used to run those procedures. The remaining steps were adapted from Dr. Walter Stroup's Introbuttons activity (Stroup, 2008b).

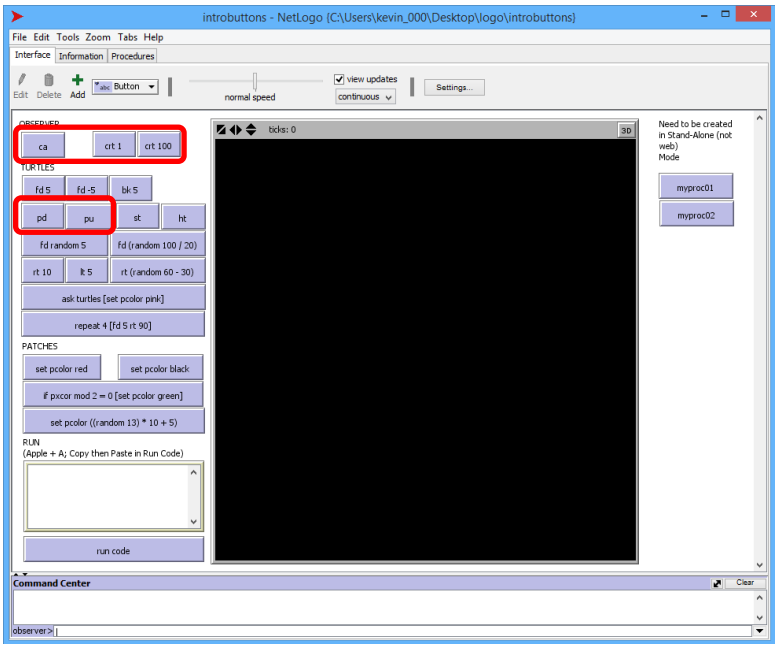


Figure 3: Basic Introbuttons

Step 5: Ask students to play with the Introbuttons. Due to concerns about time, several buttons (shown in Figure 3) were explicitly taught to jumpstart the experiment: **ca** (clear all), **crt 1** (create one turtle), **crt 100** (create one hundred turtles), **pu** (pen up), and **pd** (pen down). These buttons were chosen because they were some of the most basic buttons of the set. After the brief introduction, students were asked to start playing with the Introbuttons. Students were often asked how they did something, particularly if it was found to be interesting. Students who showed concerned about doing something wrong were given encouragement and assured that there was no right or wrong answers. Periodically, students were asked to share what they noticed.

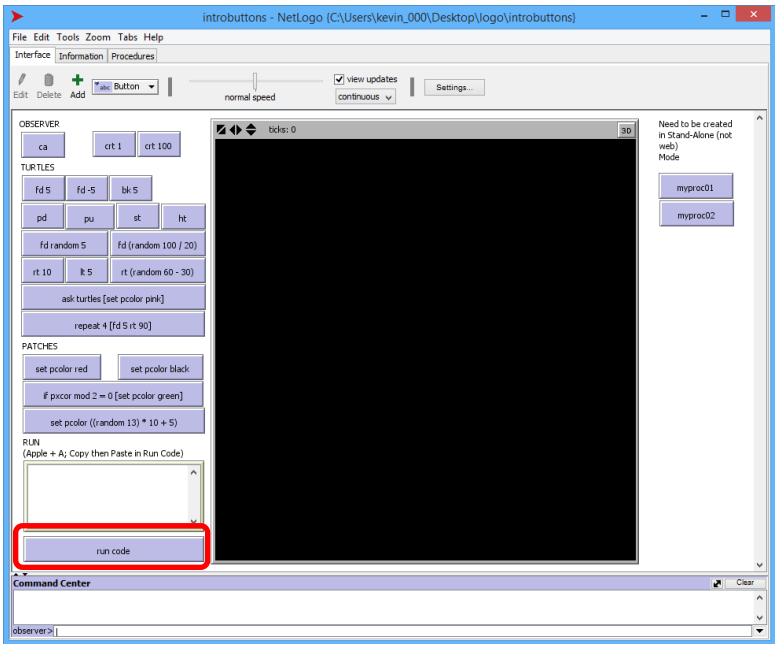


Figure 4: "Run code" Button

Step 6: Ask students to start typing source code. After some time, students were asked to click the **run code** button (see Figure 4) and enter source code. Although they were asked to look at the source code that was being generated in the caption box above the **run code** button, students were not told to type anything specifically.

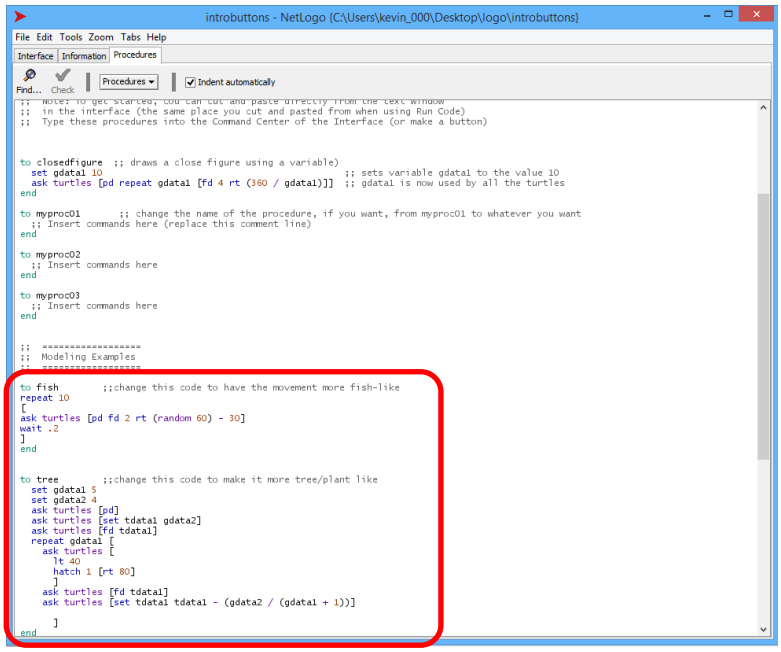


Figure 5: Procedures Tab Showing the Fish and Tree Procedures

Step 7: Ask students to explore and modify the included FISH and TREE procedures. Students were asked to navigate to the "procedures" tab and view the source code (see Figure 5). The Fish and Tree procedures are at the end of the source code. The goal is to modify the source code to make the fish look more like a fish and the tree to look more like a tree.

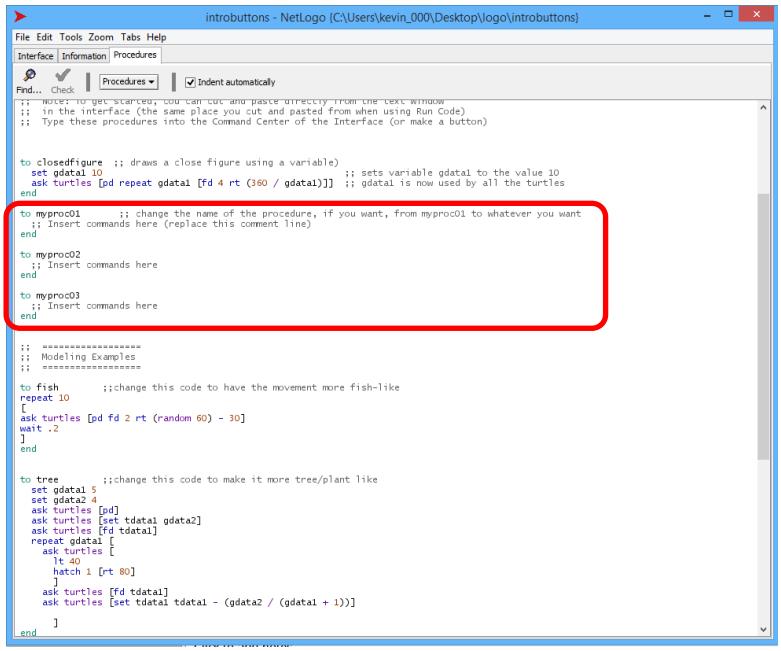


Figure 6: Source Code for myproc01 and myproc02

Step 8: Ask students to make their own procedures. After students felt comfortable modifying the fish or tree procedures, they were asked to make their own procedures using the **myproc01** and **myproc02** buttons. The **myproc01** and **myproc02** procedures are already instantiated in the source code (see Figure 6), but they are empty and therefore do nothing. No further instructions were provided except for those written in the comments.

Step 9: Save, collect, and analyze data. At the end of the period, students were asked to return their recording devices (or save their screen-casts) and save their source code. Note that screen-casts can take up to thirty minutes to process and save. After collection, all the data files (camera video, screen-cast, and source code) were immediately renamed to keep them organized. All memory cards were cleared and all batteries were charged to prepare for the following day.

The framework of this experiment (Brown, 1992; Cobb, Confrey, diSessa, Lehrer, & Schauble, 2003; diSessa & Cobb, 2004; Edelson, 2002) has been supplemented with strands of grounded theory (i.e., Glasser & Strauss, 1967; Glaser, 1992; Strauss & Corbin, 1990), which involves repeatedly drawing samples and refining the data collection until a stable theory emerges from the data (Lichtman, 2013). To harness the data, an exploratory approach to

coding was taken given the theory-driven nature of the experiment (Schoenfeld A. H., 1992). The following three coding methods were ultimately used (not predetermined):

- An adaptation of Magnitude coding was used to visualize the size of students' inputs as a function of time (Saldaña, 2009)
- An adaptation of Theme coding was used to confirm the perceived emergent progression of students' activities in the Introbuttons activity (Saldaña, 2009)
- Axial coding was used to identify areas of generality as evidence for scientific conceptual development (Strauss, 1987)

In no way do we suggest this coding scheme meets the "standards for novel methods" (Schoenfeld, 1992) or posit the data as proof. Rather, we will describe our coding methods with as much detail and transparency as possible so that readers can 1) identify the limitations of this study and 2) replicate the methods in their own experiments and add to the body of data. The overall intent is for our results and conclusions to seem reasonable given an understanding of our coding methodology and its limitations.

Chapter 4 - Results and Data Analysis

The extent of the instructions during the activity, especially in the early minutes, was to "play around with it". As a result, the six groups took different paths in their exploration. Some of the activities seen during the first day of the activity included drawing shapes, creating patterns, configuring the background colors, using the 3-D viewer, and trying to crash the computer by creating turtles. Some of the words the students used to describe their creations included "dandelion", "kaleidoscope", "rainbow spider web", and "fireworks". Groups were assured that they were not doing anything wrong.

Initial theory development. As mentioned in the previous chapter, my live observation of the experiment helped generate the first iteration of the theory. It seemed that students' interactions with LOGO spontaneously increased in complexity. Most students were able to type their own source code by the end of the first day even though they had not been taught how. To see how my observation of increasing complexity looked at a macro view, the size of the students' inputs were coded.

Figure 7: Data Coded According to Input Size

Figure 7 above shows the results of all six groups (higher resolution versions are available in Appendix A). Button presses were coded as a size of "1" because they required only one button press. Custom source code was coded as the number of characters that were typed. If the source code compiled and ran without generating an error, the input was coded as positive. If the source code generated an error message, the input was coded as negative. Button presses were always +1 because they always ran without error. Pressing the **run code** button but ignoring it was coded as 0. To manage the scope of this analysis, coding stopped when a lull occurred in the activity. In all but one group, this happened 12-15 minutes after entering the

LOGO microwold. For example, in the twelfth minute, Group 5 stopped to show what they had learned and in the thirteenth minute, Group 3 crashed their computer by creating too many turtles. Group 6 crashed theirs in the fifteenth minute.

This high-level view confirmed the original observations. The rapid, indiscriminate button pressing is easy to see in the early minutes as the data points are so densely packed that they almost overlap. In Group 3, the following exchange happened in the second minute (1:50) while the students experimented with the Introbuttons:

David: We're just clicking every single button.

Rachel: That's how you do it.

As the students spent more time in LOGO, their inputs both increased in size (complexity) and decreased in frequency as students started deciphering the buttons. Less than two minutes later, Group 3 had determined the function of three additional Introbuttons:

David: Those, ah. Those... crt creates a turtle. crt 1 creates one turtle. crt 100 creates a hundred turtles

Rachel: (clicked **crt 1** to create a turtle and clicked **fd 5**, **fd -5**, and **bk 5**). This is like movements. Forward five. Back five. (Compares **fd -5** and **bk 5**). Looks like there are two back five buttons. Whatever.

In Group 2, the students used the 3-D view to confirm their understanding of several Introbuttons.

Mark: (clicks **ask turtles [set pcolor pink]**) Does it change in 3D?

Mark: So the dots are above the pink (creates a few turtles then experiments with other buttons)

Several things happened here. First, the Introbuttons provided concrete items for the students to manipulate (Vygotsky, 1987). In the context of the LOGO microworld, the Introbuttons allowed the students to externalize their intuition of LOGO so that it could be tested (Papert, 1980). Fortunately in a microworld, there is consistent application of decision rules and complete, accurate, and immediate feedback (Sterman, 2000). The lack of dynamic complexity and inconsistency commonly found in the real world made LOGO more accessible for reflection (Sterman, 2000). Therefore, the students' ability to learn LOGO seems limited only by the speed they were able to resolve conflicts with their intuition. Once they gained literacy of the buttons, students progressed from random button pressing to deliberate button

pressing. As time progressed, the students started typing source code and the pace slowed even more.

Intermediate theory development. In attempt to understand how students taught themselves to write source code, several additional attempts were made to code the data, but many of them were unsuccessful. Some codes were too granular and resulted in noise. Other codes were too high-level and required aggressive assumptions to interpret the data. Finally, based on the conclusions of the initial analysis and watching the videos many times, the data was coded according to a progression that seemed to consistently emerge:

Step 1: Students press buttons indiscriminately.

Step 2: Students interpret button meanings and/or use them discriminately. In this step, students either showed deliberate use of the Introbuttons or explained the meaning of Introbuttons out-loud. These two activities could have been treated as two discrete steps, but were ultimately combined because they both show a conscious awareness of the buttons' meaning. Also, for groups that did not talk much, students' interpretation of the buttons was not externalized verbally and would be impossible to code.

*Step 3: Students click the **run code** button.* Clicking the **run code** button did not necessarily signal readiness to type source code. Sometimes students clicked it inadvertently and canceled out of it when they were confused by the text field. These instances were coded as "Step 3" but identified as "ignored" to show that a conscious awareness of the source code had not yet be achieved.

Step 4: Students imitate the source code from the caption box. Students accomplished this two different ways. Some copy-and-pasted it into the text field while others typed it in manually. Both were coded the same way.

Step 5: Students type custom source code. Actions were coded as "Step 5" if the source code entered was different than any of the Introbuttons. Even a change to one parameter qualified the input as custom source code.

Table 1: Data Coded According to Progression

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
Step 1: Students press buttons indiscriminately	*Insufficient data*	0:00-1:27 1:41-3:03 3:13-3:18	0:00-3:20 3:36-5:45 6:14-6:30 8:41-8:55	0:00-3:09 5:01-5:35 5:40-6:40	0:00-1:09 1:20-1:59 3:11-5:07	0:00-2:02 3:02-7:34 7:47-9:12 12:08-12:20 12:59-13:44
Step 2: Students interpret button meanings and/or use them discriminately	*Insufficient data*	1:28-1:40 3:04-3:12	3:21-3:35 5:46-6:13 6:31-6:38 6:56-8:31 9:00-9:29	3:10-5:00 5:36-5:39 6:41-7:21	1:10-1:19 2:00-3:10 5:08-5:30	2:03-3:01 draw shape
Step 3: Students click the **run code** button	*Insufficient data*	3:19	0:16 (ignored) 1:06 (ignored) 3:13 (ignored) 6:39 8:32	7:22 8:04	4:04 (ignored)	7:35 (ignored) 9:13 11:00 11:14 12:21 13:45
Step 4: Students imitate the source code from the caption box	*Insufficient data*	3:20-4:13	6:40-6:55 8:33-8:40	7:23-8:03 8:05-8:28	5:31-6:39	7:36-7:46 11:01-11:13 11:46-12:07 12:22-12:58
Step 5: Students type custom source code	*Insufficient data*	4:14-5:15	8:56-8:59 9:30-		6:40-	9:14-9:49 11:15-11:45 13:46-

The results of the coding, shown in Table 1, showed that the anticipated progression did exist, but was not linear. The time periods spent at each step were coded for all groups until Step 5 was reached. All groups bounced between Step 1 and Step 2 at the beginning, so the initial wave of rapid button pushing was not comprised entirely of indiscriminate button-pressing as first thought; it alternated between random pressing, button interpretation, and deliberate pressing. A close inspection of the raw data shows that students were using their understanding of a few buttons to determine understandings of others. Used repeatedly, this cycle allowed students to expand their understanding to additional Introbuttons. A combination commonly used by students was clicking **ca** then **crt** to reset the screen and create new turtles. Here are more examples:

- In the third minute (2:01), Group 4 used the **fd 5** button to confirm that the **pd** button drew lines.
- In the fourth minute (3:52), Group 5 used the **fd 5** button to experiment with the **rt 10** and **lt 5** buttons. By alternating between the forward and turn buttons, this group confirmed that **rt 10** and **lt 5** caused the turtle to turn right 10 degrees and left 5 degrees respectively.

Some groups also returned to Step 1 or Step 2 after progressing to Step 4 or 5. For example:

- In the tenth minute (9:00), Group 3 typed their first custom code ("crt 200") and used the **fd 5** and **fd -5** buttons to see if in fact two-hundred turtles had been created.
- In the thirteenth minute (12:59), after imitating code from the caption box, Group 6 clicked some additional buttons to generate additional code to imitate.

Vygotsky claims that children learn to act with an object before they become consciously aware of it, but once conscious awareness is achieved, it leads to voluntary action (Vygotsky, 1987). In this context, the progression outlined in Table 1 could be interpreted as a progression in awareness:

- Students are able to use the Introbuttons (Step 1) before they are consciously aware of their meaning (Step 2). Conscious awareness of the Introbuttons leads to voluntary action using the Introbuttons.
- Students are able to use the button meanings (Step 2) before they are consciously aware of the source code (Step 3 and Step 4). Conscious awareness of the **run code** button and source code leads to voluntary action using the **run code** button and source code.
- Students are able to use the **run code** button (Step 3) and imitate source code (Step 4) before they are consciously aware of the source code syntax (Step 5). Conscious awareness of the source code syntax, leads to voluntary action using the source code syntax.

If the students were able to achieve voluntary action, and thus conscious awareness, how did this happen? And how did it happen so quickly without direct instruction? Although it seems clear that the concrete nature of the Introbuttons placed students on the spontaneous path of conceptual development, the Introbuttons also appear to also provide scientific instruction by displaying source code in the caption box above the **run code** button. If that instruction lay within the students' ZPD, it might explain how the students were able to learn the source code.

Combined with the immediate and consistent nature of feedback in the microworld environment, this scaffolding resulted in very quick learning.

Final theory development. Vygotsky's own ambiguity in defining ZPD makes it hard to confirm its boundaries in this experiment. However, Vygotsky also theorized that conscious awareness of concepts (generality) could develop as a child uses a native language to mediate meaning in a foreign language (Vygotsky, 1987). If the source code is treated as a foreign language, perhaps the student dialogue could be used to confirm the effectiveness of the scientific instruction provided by the Introbuttons activity.

Table 2: Indications of Conscious Awareness

Dialogue		**LOGO terms referenced**	**Generality expressed**
David:	Those, ah. Those... "crt" creates a turtle. "crt 1" creates one turtle. "crt 100" creates a hundred turtles	"fd", "bk"	"movements"
Rachel:	This is like movements. Forward five. Back five. (Compares <fd -5> and <bk 5>) Looks like there are two back five buttons. Whatever.		
Keon:	Oh, so you can change their color?	"ask turtles [set color pink]"	Syntax, Procedure, Parameter
Shawn:	Yeah.		
Keon:	How so?		
Shawn:	You type in, ok, like here it says ask turtles bracket set color pink. You can go, ask turtles bracket set color, let's try red.		
Zuriel:	Oooh (creates a few turtles and experiments with fd, rt, and lt buttons)	"rt", "lt"	"turn"
Zuriel:	Oh okay, I see. That's right turn by 10 degrees. Left turn by 5. Random turn anywhere between 60 and 30.		
Zuriel:	I figured out the backgrounds.	"pxcor", "mod"	"background scheme"
Teacher:	Oh really?		
Zuriel:	So, ask patches. Patches is the background, like the background scheme. And then, pxcor mod equals... four. Four is divided into sections, you could say. So this section here, there's red, blue, white, green is one section of four. And then you can go in. Four equals what you are selecting. Zero is the first one. Three is the last one. So you are selecting each bar within that given section and you can set the color to whatever… So you can go in and give it this.		
Afraz:	Ok, so I guess it's not separated by commas. It's just strictly…	"crt 1", "crt 100", "crt 2", "crt 99999"	Syntax, Procedure, Parameter
Mitchell:	Ok so when it comes to the coding, all it is is like, ask turtle and then you type whatever it says on the button. Or if it's the top ones, you just type in ca or crt 1 or crt 2… or 100.		
Afraz:	Uh huh.		
Mitchell:	Which I'm kind of wondering, like, if you type in crt 2 will it create two turtles?		
Afraz:	What did you say? We can try that.		
Mitchell:	Type in crt space two.		
Mitchell:	It created two?		
Afraz:	Yeah, it did.		
Mitchell:	Oh, ok. So we don't even need to use those buttons. We can put in any information that we want.		
Afraz:	crt one thousand. Haha! (types "crt 99999" and watches computer freeze)		
Mitchell:	Oh God.		

Table 2 contains evidence of conscious awareness from the experiment. Notice that these students did more than translate the source code into English; they translated the source code into concepts and created a super-concept, or generality, that includes them. For example, in

the first example, **fd** and **bk**, were not just translated as forward and backward respectively, they were further categorized into movements, a generality that includes both forward and backward. Although it is likely these generalities were already formed in the students' minds, using these existing generalities to mediate a relationship with LOGO causes the following to happen:

1. The meaning of the concept is torn from its immediate connection in the native language (Vygotsky, 1987)
2. Spontaneous concepts acquire new relationships with scientific concepts (Vygotsky, 1987)
3. The student develops a new relationship with the concept (Vygotsky, 1987).

In other words, this mediation causes the student to gain a deeper understanding of the concept or generality. This fits particularly well with Papert's suggestion that children struggling to resolve conflicts in intuition need an intermediate to help them gain a better understanding of themselves and with the concept (Papert, 1980).

Table 3: Potential Evidence for Embedded Complementarity

Introbutton	Generality	Relationship
fd 5 **bk 5**	"movements"	Group 3 used the Introbuttons **fd 5**, **fd -5**, and **bk 5** (agent-based) to develop a generality of turtle movements (aggregate). Aggregate informed by agent-based reasoning?
ask turtles bracket set color pink	Syntax, Procedure, Parameter	Group 4 used the **ask turtles [set color pink]** Introbutton (agent-based) to develop a generality of syntax, procedure, and parameter (aggregate). Aggregate informed by agent-based reasoning?
rt 10 **lt 5**	"turn"	Group 5 used the Introbuttons **rt 10** and **lt 5** (agent-based) to develop the generality of turns in NetLogo (aggregate). Aggregate informed by agent-based reasoning?
ask patches [if pxcor mod 2 = 0 [set pcolor green]]	"background scheme"	Group 5 used the **ask patches [if pxcor mod 2 = 0 [set pcolor green]]** Introbutton (agent-based) to develop the generality of a background scheme in NetLogo (aggregate). Aggregate informed by agent-based reasoning?
crt 1 **crt 100**	Syntax, Procedure, Parameter	Group 6 used the **crt 1** and **crt 100** buttons (agent-based) to develop a generality of syntax, procedure, and parameter (aggregate). Aggregate informed by agent-based reasoning?

Are the examples of generality actually agent-informed aggregate reasoning (Stroup & Wilensky, 2014)? In order for this to be true, we also need evidence of agent-based reasoning. Table 3 shows what happens when the "LOGO terms" in Table 2 are reverted to their respective Introbuttons. By treating these "terms" as concrete buttons, the agent-based component of

embedded complementarity materializes and completes the coupling. Additionally, the tacit nature of the students' concept development observed between Steps (Table 1) could be explained as the "embedded" component of embedded complementarity. Not enough data was collected to explain why embedded complementarity did not materialize in every group. The remaining groups displayed behavior that arguably qualified (for example, Groups 1 and 2 used the **fd 5** and **bk 5** buttons to determine the boundary behavior in NetLogo), but did not employ the precise language found in the other examples. An exit interview would have been useful in extracting this data, but we did not have the foresight to ask the necessary questions. In a true implementation of grounded theory, another experiment would be modified and conducted at this point to confirm our observations.

Chapter 5 - Conclusion

Because the research question posed at the end of the literature review is multifaceted, I will first address the core of the question and add "layers" until the entire question is addressed.

Can students in a high school class use the Introbuttons microworld and aspects of the Introbuttons activity sequence as a scaffolding environment to support an open-ended introduction to programming? Although the limited scope of this study makes it hard to make conclusive claims, the explicit generality produced by four of the five groups within the first few minutes is encouraging. Although not all groups produced evidence of generality in this experiment, it is not clear whether this is due to a limitation of microworlds as a whole, the Introbuttons microworld/activity specifically, the limits of the students' ZPD (Vygotsky, 1987), or the experiment's scope and/or data resolution. A follow-up experiment that extends the experiment time and tracks keystrokes and button presses automatically would be helpful in providing more insight. Nevertheless, all groups, even those that struggled initially, were able to increase their input size by typing working source code within the first few minutes without direct instruction. Even imitation (Vygotsky, 1987) of existing source code would indicate potential for development. Spontaneous learning in a microworld can provide a compelling and productive alternative to the routinized approaches to introducing aspects of programming that arguably, have limited the trajectory for the implementation of programming in schools (Papert, 1980). As such, it might well begin to provide a credible way to achieve the objectives of the Cyberlearning initiative (National Science Foundation, 2011).

The inherent time limitation of the experiment also makes it difficult to gauge the effectiveness of the Introbuttons microworld over the course of a full school year. It is not known, for example, whether the progress shown by the students would have continued at the same pace after the allotted time. Vygotsky's framework of ZPD suggests that the pace of spontaneous learning will slow as the students face ideas that grow increasingly abstract and complex. However, if embedded complementarity is also in play (and the sequencing and generality coding show that it might), the results are too difficult to predict using this study alone. Again, a longer study would be necessary to project its effectiveness in a year-long class.

Implications for a general classroom using microworlds:

- What teacher training is required?
- How should teachers assess students?

- What are the teacher's roles and responsibilities in a managing a microworld?
- What should the prerequisites be for the class?
- What should a modern microworld look like? LOGO was built on 1970s technology. Surely a microworld built on today's technology would be very different given the technological advances in computing and networking.

As their initial engagement with programming (the first few minutes), **can students in a high school class use the Introbuttons microworld and aspects of the Introbuttons activity sequence as a scaffolding environment to support an open-ended introduction to programming?** Although none of the students had prior experience with the Introbuttons microworld or NetLogo, not enough data was collected to know each student's overall programming experience. It would be interesting to conduct the experiment again with prior knowledge of each student's course history. Although programming-as-a-foreign-language legislation is gaining momentum, neither its successful passing nor successful implementation is guaranteed. Therefore, some teachers who want to integrate computer science into their curriculum will not be able to depend on prerequisite knowledge of programming.

Implications for teachers of first-time programmers:

- Would a microworld learning environment supplement or eliminate traditional routinized methods of teaching computer science?
- What type of teacher is needed?

As their initial engagement with programming (the first few minutes), can students in a high school *physics* class use the Introbuttons microworld and aspects of the Introbuttons activity sequence as a scaffolding environment to support an open-ended introduction to programming? In addition to all the aforementioned conclusions and considerations, a physics-specific implementation of a microworld has the added component of culture clash. Although this is starting to change, standardized physics tests (e.g., Advanced Placement, district exams and finals) conform to standards that have a lineage in traditional physics teaching. This means a heavy emphasis on formulas and numerical calculation and virtually zero emphasis on computer modeling/programming. While a microworld may help students connect to the "powerful ideas" of physics, it may not prepare them as well for the tests as they are currently written. Will teachers be willing to risk this potential trade-off?

Implications for physics teachers:

- Neither Introbuttons nor NetLogo is designed specifically for physics, so it would need to be adapted. What would the physics version of the Introbuttons activity look like? Could a physics-specific item from the models library be modified to work as effectively? What are the generalities and powerful ideas in physics and could a microworld let students discover them? Given that the real world is culture-rich in physics, would a physics-specific microworld need "introbuttons"? Or could it be used to augment the student's physical interaction with the real world?
- If physics was taught using computer science, the Introbuttons activity could be used to help students achieve computer science literacy as a first step. How long would that take? And will spending time to teach computer science on the front-end pay-off in over a year-long class?
- What are the downstream effects for students taking a microworld-based physics class? Will it prepare students for a college physics class with traditional values on formulas and numerical calculation?
- What kind of physics teacher is required? Will traditional physics teachers be able to make the transition without falling into old habits?

Chapter 6: Applications to Practice

It is impossible for me to discuss how the UTeach Summer Masters (MASEE) influenced my development as a teacher without also discussing the Engineering Summer Institute for Teachers (ESIT), and Engineer Your World (EYW). I had the privilege of participating in all three and have always considered each as a different member of the same family. All were run by UTeachEngineering and all were funded by the same NSF grant.

Although I am a heavy participant of UTeachEngineering now, I never planned on becoming a teacher. After earning my Bachelors of Science in Mechanical Engineering from The University of Texas at Austin, I went into business consulting, where I experienced culture shock. My peers were much more diverse than my engineering classmates and I found it hard to communicate ideas. In engineering school, everyone seemed to speak the same form of engineering English, but that was not the case anymore. I finally learned how to communicate with non-engineers, but it was a long, hard road for me. I also found that my colleagues scoffed when I presented engineering ideas that I considered commonplace. It seemed that even my ideas were indigestible to my new peers. I probably would have made a terrible engineering teacher at that time.

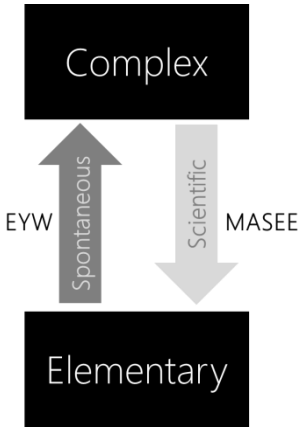


Figure 8: Vygotsky's Development Framework Applied to UTeach

My years of consulting had helped me realize that I needed to rethink how to present engineering ideas to non-engineers, but it did not show me how. Having recently become a teacher, I needed help. This is where UTeachEngineering came into the picture. Perhaps

because this research report is still fresh in my mind, I believe UTeachEngineering was so effective because it provided me with both Spontaneous and Scientific teacher development (see Figure 9).

The EYW provided the Everyday development by giving me concrete experience. The two most important tangibles were the recently minted UTeachEngineering Engineering Design Process (EDP) and project-based learning. The former became a critical anchor in my classroom because it encapsulated engineering into a friendly illustration. In my three years of teaching EYW, I found that students had broad misconceptions about engineering. Some thought it was construction. Others thought it was gadgeteering. In these situations, I could always point to the EDP. The latter provided my students with meaningful context to explore the EDP and develop engineering habits of mind organically. My high school is academically competitive with a culture that punishes failure. In this context, the clichés about failure are meaningless. Engineering projects not only gave my students safe opportunities to experience failure, it showed them how to use failure to learn. While my students were getting hands-one experience with engineering, I was getting hands-on experience with teaching engineering.

As I taught EYW, my understanding of project-based learning started to grow, but hands-on experience alone was not enough. At the time, project-based learning was a hot topic, and lots of teachers either wanted to get started or claimed they were already doing it. From my perspective, some projects that I experienced outside of EYW did not seem right, but I could not explain why. Although I had a vague ability to identify great projects from those that were meaningless, I had no conscious awareness of what made certain projects better than others. This is where MASEE came in: it provided the Scientific development.

Using theory and history, MASEE gave me a conscious awareness of EYW's design. Reading and discussing Sadler and Vygotsky have helped me understand why we encourage engineers to prototype. Reading and discussing Walkington helped me explain why we reflect forward and backwards in our engineering notebooks. Reading *How People Learn* empowered me to discuss the differences between experts and novices with my students. MASEE also helped me find answers to a lot of questions that I had collected over the years. Reading and discussing the history of our educational system helped me understand why our schools are the way they are and that some of the "new" ideas in education are not really new. Reading Papert and Hatano helped me understand why students struggle with physics and math. Many of those questions were the genesis of this Masters report.

Dr. Stroup says that literacy is empowerment. It is more than being able to understand; it is "being able to do what you want to do". As I simultaneously taught EYW and pursued my Masters, I felt more literate in engineering. It was more than just comfort or familiarity with the curriculum; it was the ability to extend the curriculum to meet my students' capabilities and keep within the spirit of the original. It started in small ways, like having unplanned discussions about patent protection. As I grew more literate, my extensions became larger (like adding subsystems). This past year, I was able to help my students send their aerial imaging high enough to see the curvature of the earth. Although the project materialized very differently than originally prescribed, I made sure all the major components of the original stayed intact: sub-systems, Pugh chart, C-Sketching, flight plan, and of course the EDP. I even pulled in some reverse engineering because it was appropriate for our situation.

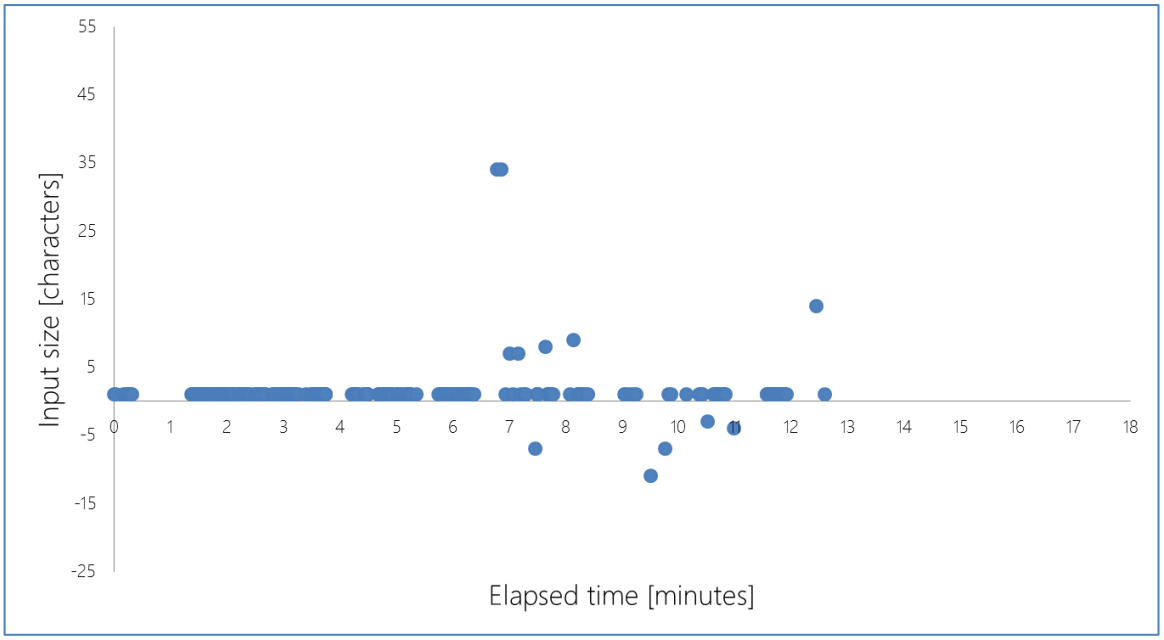
My adaptive expertise of teaching engineering also allowed me to extend engineering into my physics classroom. Although it is hard to find engineering projects that encompass the full spectrum of physics TEKS, I have found opportunities to pull small threads of engineering and weave them into my physics class. For example, when my students were building musical instruments to learn waves, I asked students to brainstorm using a C-Sketch. The activity allowed me to briefly discuss the reasons it is better than individual brainstorming. Although my individual efforts pale in comparison to the coherence the UTeachEngineering team has achieved with the EYW curriculum, my training has allowed me to add engineering components discriminately and responsibly. Although I think that some exposure to engineering is better than none, adding engineering components haphazardly does the students no favors.

Next year, I will be building a curriculum for a new STEM academy and I know nothing about my situation except that I am representing 12th grade science. It is an exciting opportunity, but I would not have felt prepared to walk into such an uncertain situation without all the training and development from UTeachEngineering. EYW and ESIT have given me concrete experience with STEM and MASEE has given me an understanding of the educational theory to make STEM effective. This means, as I plan for next year with my fellow 12th grade teachers, I can speak from both experience and from theory. I will be able to intelligently discuss the practicality and effectiveness of ideas. I will be able to distinguish powerful learning techniques from fads. I will know how to integrate engineering components to develop engineering habits of mind. I know that I will not have all the answers, but now I know where

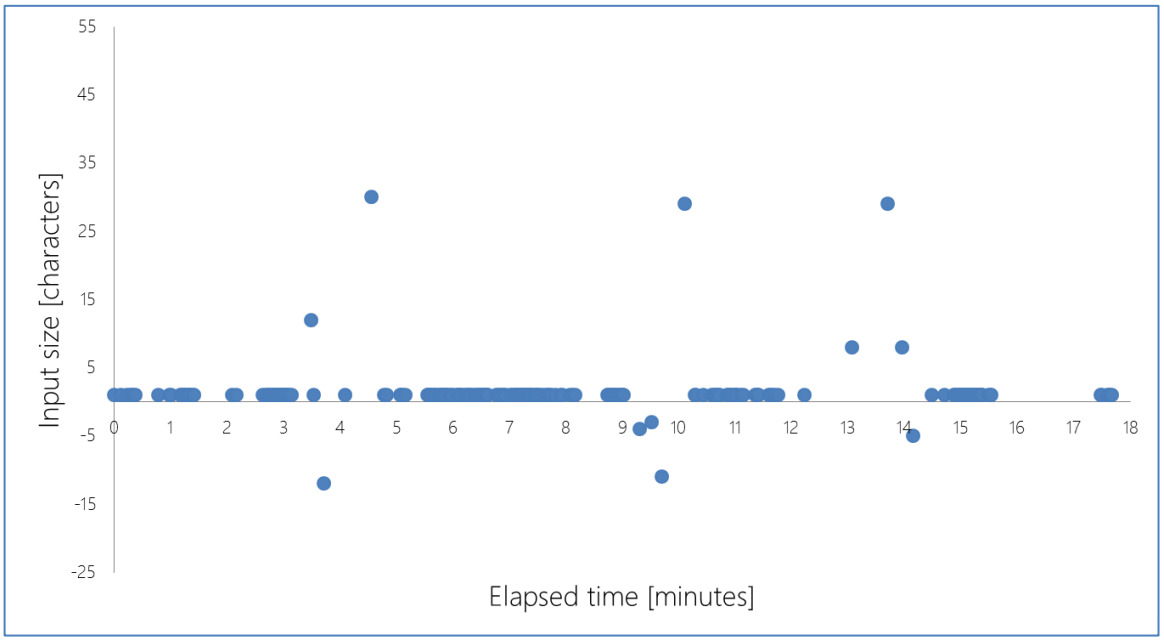
to start my research and I have a community of teachers and researchers to ask. Even four years ago, I neither had the capability nor the resources to do any of this.

Appendix 1 – Data Coded for Input Complexity

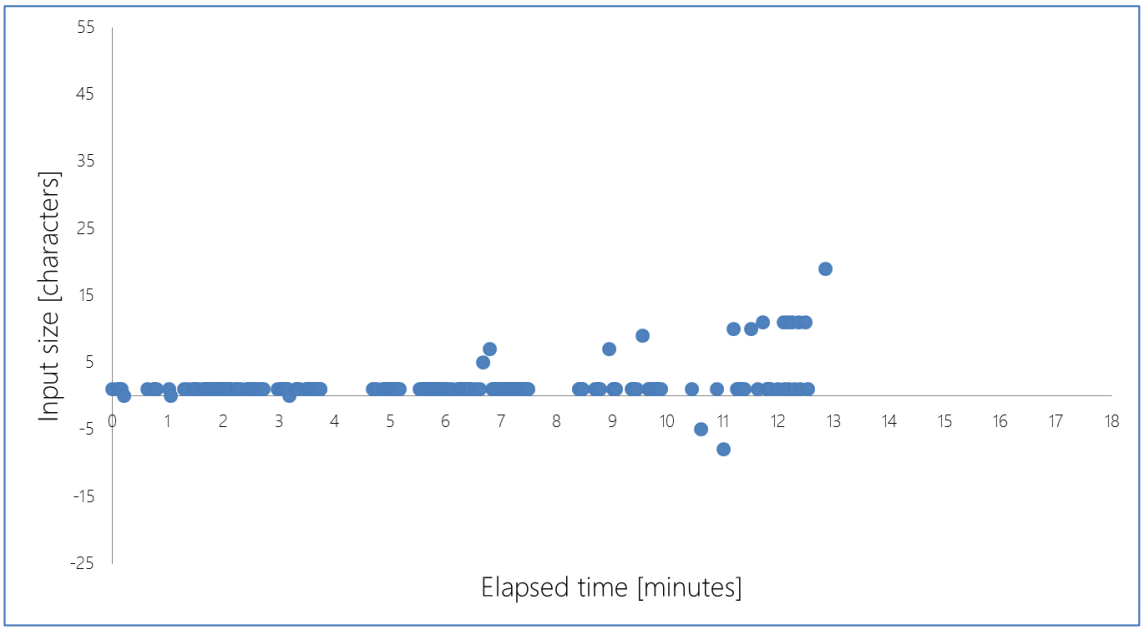
Coded data for Group 1



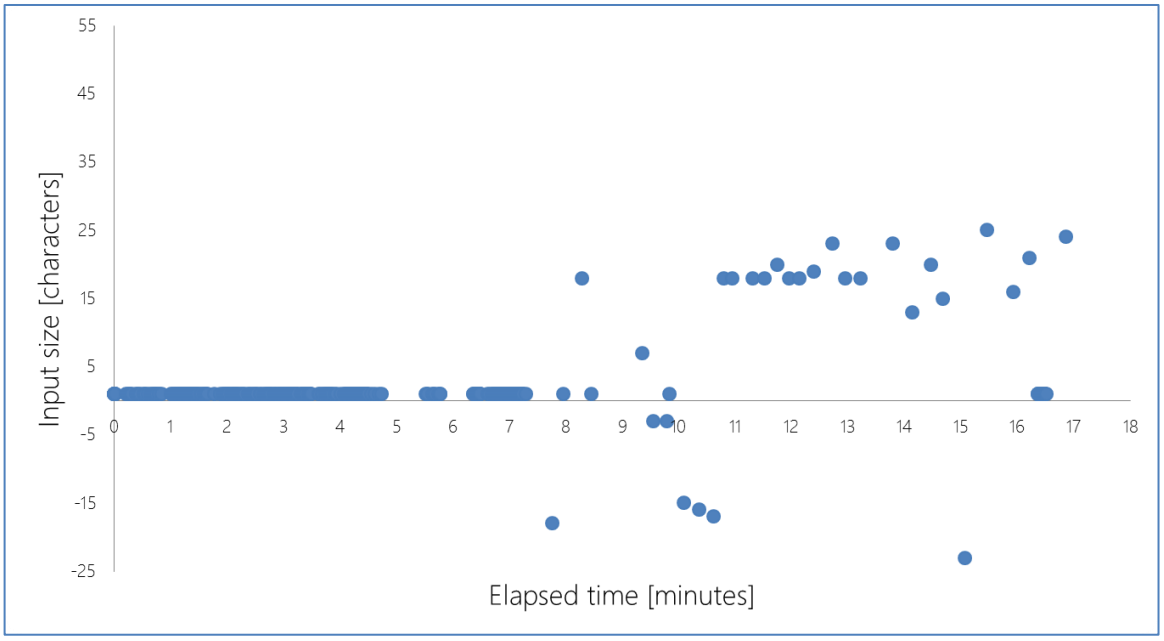
Coded data for Group 2



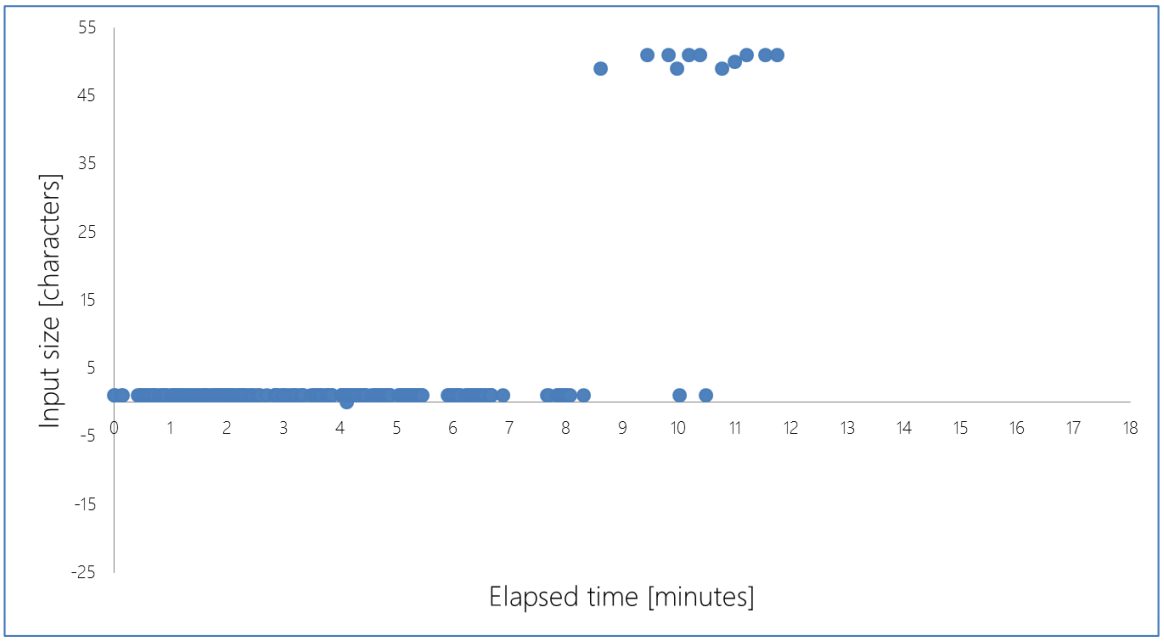
Coded data for Group 3



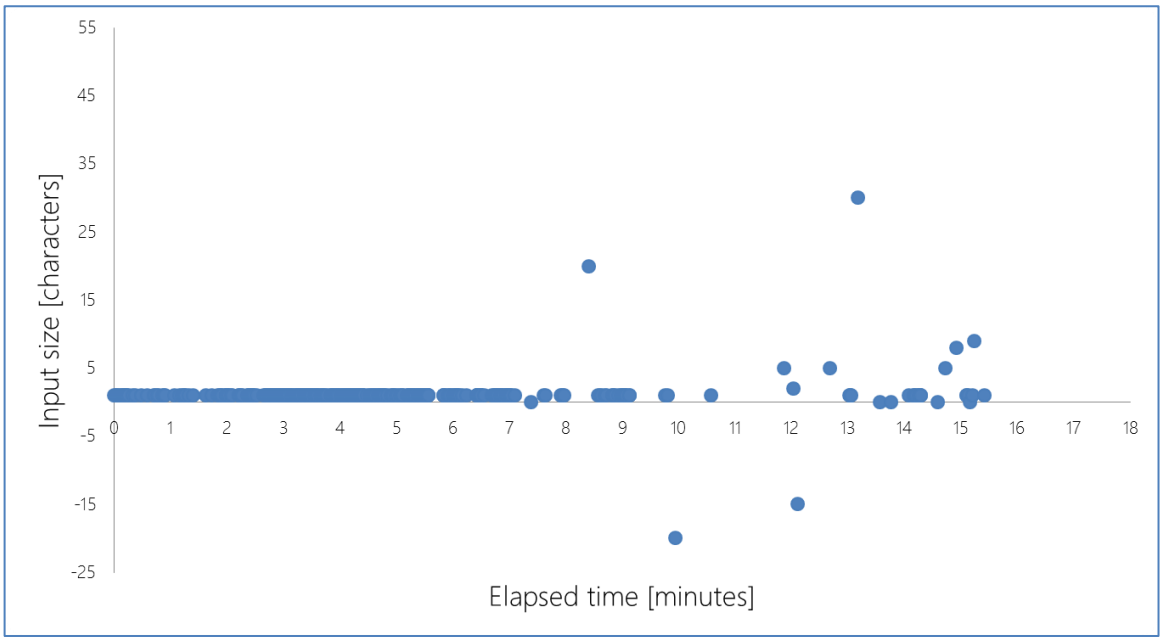
Coded data for Group 4



Coded data for Group 5



Coded data for Group 6



Appendix 2 – Raw Data

Group 1 raw data
Clicks buttons [7:00] Teacher: What are some things that you notice? [8:00] Copies text from the caption box and pastes it into command center [9:00] Steve: If I did a thousand? Cole: Inaudible Steve: I wanna do a thousand Cole: That's massive. It's like an army. Steve: I'm setting up my troops Madison: Is there a button for a thousand? Steve: No, there's only one for a hundred. You have to just… you have to go into observer and type it in yourself. Cole: What did you put in? [10:00] Steve: What? I put in a thousand. I'm going to clear it. (presses ca) Steve: C R T ten thousand… Oh man! Cole: I think its too complicated for the program to just… Steve: Hahaha Cole: 3-D? (switches to 3-D view) Cole: It's completely full [11:00]

Group 2 raw data	Code
[0:17] Mark: What do the "fd"s stand for?	Illiteracy
Katie: I don't know. Try those cr 1.	Unsure
Mark: Wait. Which one? crt 1?	What does that do?
Katie: Yeah	
[CR clicks the crt 1 button]	Meaningless clicking

Group 2 raw data	Code
Katie: Whoa!	
[0:25] Mark: Do the 100	Tries more buttons
[CR clicks the crt 100 button]	
Katie: That's a big one.	
[0:31] (CR clicks the bk 5 button several times to create moving concentric circles)	Illiteracy
Katie: Whoa. Whoa. Whoa!	Unsure
Mark: I don't know what that does though. //refering to the bk 5 button	What does that do?
[1:12] Mark: What does that button do?	
[presses ht button and turtles disappear]	
Katie: What does that do? Whoa!	Changes perspective
[pans window looking for turtles]	
Mark: There's a 3D button. Hang on.	Tests intuition
[1:19] (creates 100 new turtles. presses bk 5 and fd -5 several times to reverse the turtles, making the circle larger. Then clicks fd 5 and the turtles move forward, making the turtle smaller… until the circle collapses and the circle grows larger)	
Mark: fd 5 makes it go smaller.	Intuition is wrong
[clicks fd5]	
Mark: Ok…	
[1:31] (presses bk 5 multiple times and the same thing happens except the turtles move in opposite directions.)	Changes perspective
Katie: They all do that //intuition is wrong	Unsure
[clicks 3D View button]	
[1:45] Mark: What did that do? I don't think I did the right one.	
[rotates the 3D view]	
Katie: Whoa! Cool! That is cool!	
Mark: I don't think I did that right.	
Teacher: No… There's no right or wrong.	
Mark: There's no right or wrong? Ok...	
Teacher: At this point, you're just playing around. How did you do that?	
Mark: There was a button called 3D view…	
Teacher: Cool. That is pretty cool.	Unsure

Group 2 raw data	Code
[keeps playing with 3D view]	Changes perspective to test intuition
Mark: How do I go back? Ok	
[2:31] Mark: So now random… what does that do?	
[clicks ask turtles [set pcolor pink]	Confirms intuition
Mark: Does it change in 3D?	
Mark: So the dots are above the pink	
Mark: No, no, no. This one bad.	Verbalizes code
[3:02]	
[clicks rt 10]	
Mark: That's rotate	
// How?	
[clicks lt 5]	
[clicks rt (random 60 - 30)]	
Mark: That's random	
[clicks fd random 5 repeatedly]	
Katie: Whoa! Randomizes it.	
[3:19 clicks run code button]	
Katie: Halt?	
Mark: What user code?	
(Types "fd random 25")	
Mark: Did that do anything?	
(3:40 Tries it again)	
(4:17 Tries it again: ask turtles [set pcolor green])	
Mark: I don't know what that did, but it did something	
Katie: Yeah	
[5:24]	
Mark: Wanna try?	
Katie: Sure.	
Katie: What's that?	
Mark: I think that's the fds	
Katie: Oh, what's that? Whoa!	
Katie: I dunno	

Group 2 raw data	Code
Mark: Try and clear Katie: Oh, wait. It's within the box. It stays in the box and it continues to spread out.	

Group 3 raw data	Code
// Clicks buttons that teacher introduces	Awareness fd random
[0:18] Clicks run code button; ignores	
[1:04] David: What does "fd random" do?	How does the ZPD change?
Teacher: fd? Play around with it and see what it does.	
David: Ok.	
[1:07] [Clicked the "run code" button]	
[1:07] David: Do I need to type anything in the "user input code"?	
// Ignores run code button	
Teacher: No. We'll talk about it in a little bit. Just play around with it right now.	Illiteracy all buttons
David: Ah, ok	Mediating crt
[1:50] David: We're just hitting every single button	
Rachel: That's how you do it	
// repeatedly clicks [repeat 4 [fd 5 rt 90]	
Rachel: They're all centered around the middle.	Awareness fd and bk
David: I know.	
// clicks all buttons. Buttons allow for small delay time and consequently short cycle time	
//clicks myproc01 and myproc02	
//repeatedly clicks crt 100	
Rachel: Gonna create more turtles	Illiteracy
[2:50] David: Wanna try it?	
Rachel: Sure.	Volition fd
[3:00 switched users]	

Group 3 raw data	Code
[3:16]// clicks “run code” button and ignores	
David: I think you have to…	Literacy crt and fd
Rachel: Yeah… What did he say to start it?	
David: Those, ah. Those... “crt” creates a turtle. “crt 1” creates one turtle. “crt 100” creates a hundred turtles	
[3:31] Rachel: This is like movements. Forward five. Back five. [Compares <fd -5> and <bk 5>] Looks like there are two back five buttons. Whatever.	
// How?	
Rachel: [plays around with window size]	
Rachel: I think I broke it	
//What did BL do at 5:08? Caption box came up? Types info in caption box but seems to get an error message	Literacy
[5:50] Rachel: So it has collision along the edges	
[6:05] It just makes the area pink underneath.	
[6:00] David: See where it says “continuous” next to settings? What does that do?	
//changes settings and turtles change edge behavior	
Rachel: [Pulls down menu] On ticks. I don’t know. [creates 100 turtles and clicks the fd 5 button continuously]	
//how did he know to do that?	
David: Oh. That’s sick.	
Rachel: Yeah, after it goes off the edge they don’t go back.	
David: Yeah, yeah, yeah.	
// How?	
//I should have had students talk to each other about what they are noticing so far	
BL noticed that you can replicate the button actions by typing in the code.	
// How?	
// Clicks run code button and types test code. I think he copies something from the caption box.	
[6:44] Rachel: So pretty much you just type that and then it does it.	

Group 3 raw data	Code
[6:57] Rachel: Kinda basic but kinda cool though. [8:50] [BL tries to create 200 turtles by typing the source code rather than using buttons] // How? [9:30] Rachel: Let's try to crash it. [9:55] David: I wonder what would happen if you tried to make a million. // how would i document the progress happening here? Rachel: I think the computer would die David: Literally put as many zeros as you can Rachel: Does it do half turtles? [15:03] I can remake it [19:xx] make procedures	

Group 4 raw data	Codes
[Clicks buttons]	Clicks meaninglessly
Keon: Huh	
Shawn: And there's…	
Keon: This is underwhelming	Underwhelming
[screen turns orange and then pattered]	
Shawn: Ooh, that's an interesting color scheme. That's very interesting.	Interesting
[hides turtle]	
Shawn: Hmm.	What happened?
Keon: Move forward and backward	
// How?	
Shawn: Where did it go? Oh there it is.	Explains result
[mouses over code]	
[creates more turtles and moves them]	
Shawn: Ooh.	
[1:00]	

Group 4 raw data	Codes
Keon: They're all moving in different directions	Explains result
Shawn: That's interesting.	Fun
Keon: [inaudible]	Artwork
[repeatedly clicks crt 100 and then bk 5]	Desire
Shawn: You just made a bunch of them. That's interesting. That's a lot of turtles. Oh my gosh. This is actually kinda fun. It's like you're making digital artwork. I'm definitely doing this after you [inaudible].	Explains result
[2:00]	Explains button
Shawn: They're just making a bunch of lines. This is really cool.	Experiments with a button
Keon: Yeah. That's what pen down does. It let's you draw lines.	Confirms button
Shawn: That's cool. What does the pu do?	
Keon: pu? Let's see.	
Shawn: Yeah.	Clarifies button function
Keon: Oh yeah, it's pen up and pen down.	
Shawn: Let's see what that does.	
Keon: No pen down makes them draw and pen up…	Personifies the image
Shawn: Oh, gets rid of that?	
Keon: Yeah.	
Shawn: Wait, what about the… okay ht when you did that gets rid of one of them, but what happens when…	
[creates a burst pattern using crt, pd and bk]	Experiments with a button
Shawn: Oh cool! It's like a little fireworks show. And then some.	Explains result
[clears]	
Shawn: Ok, what does st do?	Experiments with a button
[3:00]	Explains result
[clicks the turn buttons]	
Keon: Random…	
Shawn: Oh cool!	
[clicks rt(random 60-30)]	
Keon: This makes them rotate...	
Shawn: Rotate them just a little bit.	
[clicks ask turtles set pcolor pink]	

Group 4 raw data	Codes
Shawn: Oh that makes… puts paint on them for no reason.	
[4:00]	
[creates lots of turtles]	
Shawn: That's a crap ton of them. Oh,	
that's…	
[4:20]	Personifies image
[creates lots of turtles]	
Shawn: Oh gosh...	
Teacher: You're creating a bunch of turtles?	
Shawn: Yeah, we're creating a crap ton of turtles.	
Teacher: What's happening there? Oh, it's just taking so long to create...?	
Keon: Yeah.	
Shawn: That's pretty cool. It's like a cool… 3-D art almost.	
[5:00]	Personifies image
Shawn: Oh, there's actually a 3-D!	
[Opens 3-D view]	Changes perspective and
Shawn: Oh wow, that's so cool! That's 3-D…	confirms guess
[creates a bunch of turtles and fd random]	
Keon: [inaudible]	Awesome
Shawn: That does. It looks more like an actual firework almost. It's like a	Random
colorful firework with like multiple colors going on.	Colorful
[6:00]	
Keon: These turtles are above the line.	
Shawn: Yeah. That's actually pretty cool. It's like the turtles are three-	Personifies image
dimensional.	
Teacher: What are some things you are noticing so far?	
Shawn: That you can make awesome random colorful drawings. What	
does st do?	Aware of code
[Creates lots of turtles and draws lines]	
Shawn: Lots of turtles. Turtles everywhere!	Replicates button code
[7:00]	
Shawn: Turtles are our friends. Like a rainbow spiderweb.	Debugs code
[7:20 clicks the run code button and starts typing, "bk 5]]	Types custom code by

Group 4 raw data	Codes
[gets error message]	changing parameters
Keon: Apparently that is not right.	
Shawn: It's the code that's like here [points to screen]	Code works
Keon: But there's a space [inaudible]	Teacher prompt, but this group already started coding
[types more code] //I think it was rt 90, which is not a button.	
Keon: So that makes it turn 90 degrees.	
Shawn: Oh cool!	
// How?	Aware of syntax
[8:50 switch]	
[Types crt 100]	Corrects syntax
Teacher: I see JV typing in stuff. Start typing in code.	
[Types custom code and gets an error]	
Shawn: Ok, there is extra stuff with it.	
Keon: Ask turtle	
Shawn: Oh, ask turtle, okay	
// How?	Literacy
[Types custom code and gets an error]	
Shawn: What's… what's?	
Keon: You have to…	
Shawn: I'm typing...	
[Types custom code and gets an error]	
[Types custom code and nothing seems to happen]	
[Types custom code and gets desired result]	Corrects syntax
[11:30 Types custom code and gets desired result]	
[Types custom code and gets desired result]	
Types code several more times	
Shawn: Ah yeah. That's what I'm talking about.	
Keon: Oh, so you can change their color?	
Shawn: Yeah.	
Keon: How so?	
Shawn: You type in, ok, like here it says ask turtles bracket set color	Changes parameters

Group 4 raw data	Codes
pink. You can go, ask turtles bracket set color, let's try red. [turtle color changes] Keon: Hmm. It says pcolor right here. Now, it says set pcolor. And that was the one that put it on the background. [Starts to type new code but forgets to type "ask turtles"] Keon: You have to ask turtles Shawn: You always have to ask turtles. Keon: pcolor puts it in the background. Yeah. Puts it in the background. Shawn: Let's try green. [changes background color to green] Keon: Yeah, puts it in the background. [Types more custom code] Keon: Why are you having them go backwards? Shawn: I dunno. [15:00] Shawn: I kinda wanna still keep doing this. This is fun.	Corrects syntax

Group 5 raw data	Codes
[0:00] [clicks the buttons identified by teacher] [starts to move down the panel] [1:00] Teacher: How did you do that? [2:00] [clicks crt 100 repeatedly] Zuriel: You're making a lot of turtles [2:46] day 1.2 [creates a few turtles then experiments with other buttons] [3:00] [clicks repeat 4] Zuriel: Oooh	

[creates a few turtles and experiments with fd, rt, and lt buttons]
Zuriel: Oh okay, I see. That's right turn by 10 degrees. Left turn by 5. Random turn anywhere between 60 and 30.
// How?
[4:00]
[creates lots of turtles one at a time]
[alternates between fd and lt]
[5:00]
Zuriel: Look, we're making circles today.
Zuriel: Look Mr. Teacher, we're making circles!
// How?
 	
Teacher: How did you do that?
Zuriel: Ah, you create a whole bunch. Make them go forward 5 pen down. Left turn 5 [inaudible]. Then repeat.
Teacher: So it looks like they're going off the screen it looks like?
Zuriel: What they do is they go down, and where this screen ends it starts at the top and it keeps on repeating.
//when did he first realize this?
 	
Teacher: That's pretty cool.
[4:08 clicks run code and ignores] //wait, maybe this was not the first time
[tries to type in code. where? the command center?]
[5:21]
Zuriel: Oh, ok. I got ya.
[highlights code in the run box]
[6:45 Copies code into the observer box]
 	
//when did he first do this?
Teacher: Are you typing in code?
Zuriel: Yeah, I'm still messing with the same command.
Zuriel: Give me a color
Anthony: Red
Zuriel: Give me another color
Anthony: Blue	

Zuriel: Yeah Zuriel: Give me another color Anthony: White One more Anthony: Black Zuriel: Oh. Three. We need to get rid of that one. Zuriel: I figured out the backgrounds. Teacher: Oh really? Zuriel: So, ask patches. Patches is the background, like the background scheme. And then, pcolor mod equals... four. Four is divided into sections, you could say. So this section here, there's red, blue, white, green is one section of four. And then you can go in. Four equals what you are selecting. Zero is the first one. Three is the last one. So you are selecting each bar within that given section and you can set the color to whatever… So you can go in and give it this. Teacher: How did you figure that out? I did this one right here and noticed that two gives you two options. And then it repeats. And zero… I assumed zero was the first one. So I went in and physically changed... Teacher: So you made some assumptions and then you just experimented to confirm your assumptions? Zuriel: And then I changed the colors as well. Teacher: Which colors are available? Zuriel: So far green, black, white, red, blue. That's all I've played up with. Teacher: How did you know that? Zuriel: I just plugged them in. Teacher: Ok. Wait, you just typed in the colors? Zuriel: Yeah. Down at the pcolor. Let's try yellow. Anthony: Yellow Zuriel: Give me a purple. Purple has not been defined. Anthony: Maybe we only have primaries.	

Group 6 raw data	Code
[clicks buttons - first three rows - as teacher talks] [starts to explore rest of buttons] [Makes a burst pattern] [Clears and experiments with other buttons] Mitchell: That's kinda cool Afraz: Yeah Afraz: My arm is going to literally fall off by the time this video is over [1:35 Clicks run code button and ignores] Mitchell: I don't know what we are supposed to be doing here. Wanna try? Afraz: Yeah. 1.2 [0:25 Draws a closed shape using the buttons //literacy // Uses buttons like a joystick [3:00] Do you want to switch? I don't know what else I can do. [3:03 Switch] Mitchell: Is he going to tell us what to do? Afraz: I don't think so. I think we're just messing with it right now. Mitchell: Do we have a set of instructions? [inaudible] Teacher: Right now you're just playing around with it Afraz: Ok. Teacher: I just want you to experiment. And try to make something cool looking. Teacher: What are some things you are noticing right now? [4:50 Clicks run code button and ignores] [Adjusts window size] [Opens 3-D view and closes it] Afraz: That was the 3-D view Mitchell: Yeah [Opens 3-D view again but it only had one turtle -- nothing to see] [5:35 Highlights code] [5:50 Copy and pastes custom code. It works]	

Group 6 raw data	Code
[6:45 3-D view] 1.3 [0:00 Types custom code. Gets error] [2:05 Types crt 1] [2:40 Types custom code. Gets error] [2:50 Types crt 1] Tries to copy paste big block of code and gets error Mitchell: I guess you can't type them in all at once. Afraz: Ok, so I guess it's not separated by commas. It's just strictly… (switch) Mitchell: Ok so when it comes to the coding, all it is is like, ask turtle and then you type whatever it says on the button. Or if it's the top ones, you just type in ca or crt 1 or crt 2… or 100. Afraz: Uh huh. Mitchell: Which I'm kind of wondering, like, if you type in crt 2 will it create two turtles? Afraz: What did you say? We can try that. Mitchell: Type in crt space two. Mitchell: It created two? Afraz: Yeah, it did. Mitchell: Oh, ok. So we don't even need to use those buttons. We can put in any information that we want. Afraz: crt one thousand. Haha! (types crt 99999) Mitchell: Oh God. Afraz: That would probably crash it actually. Yeah, I crashed it. Mitchell: Wait. Afraz: No. Hold on. Mitchell: Forward. Oh God. Afraz: Oh. [laughter] Afraz: Maybe I shouldn't have created 99,999. Mitchell: I think we also crashed ours. We figured we don't have to input	

Group 6 raw data	Code
the exact buttons. Teacher: So yours crashed? Oh, it's still drawing. Mitchell: We typed in crt 99999 Teacher: It looks cool though. It looks like a ball. Afraz: Explosion of colors. Teacher: It looks like a dandelion or something. Afraz: Or a kaleidoscope. Teacher: Or a Death Star? Afraz: This is awesome. What do we do now though? It's just going to draw all of them. Mitchell: We could reopen the program.	

Bibliography

Bracey, G. W. (2003, April). April Foolishness: The 20th Anniversary of "A Nation at Risk". *The Phi Delta Kappan, Vol. 84. No. 8,* pp. 616-621.

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions. *The Journal of the Learning Sciences*, 137-178.

Chi, M. T. (1981). Categorization and Representation of Physics Problems by Experts and Novices. *Cognitive Science*, 121-152.

Clement, J. (1982). Students' preconceptions in introductory mechanics. *American Journal of Physics*, 66-71.

Cobb, P., Confrey, J., diSessa, A., Lehrer, R., & Schauble, L. (2003). Design experiments in educational reserach. *Educational Resarcher*, 9-13.

diSessa, A. A., & Cobb, P. (2004). Ontological innovation and the role of theory in design experiments. *Journal of the Learning Sciences*, 77-103.

Edelson, D. C. (2002). Design research: What we learn when we engage in design. *Journal of the Learning Sciences*, 105-121.

Foreman-Roe, S., & Bellinger, G. (2013). *Insight Maker.* Retrieved from http://insightmaker.com

Glaser, B. G. (1992). *Basics of grounded theory analysis: Emergence vs. forcing.* Mill Valley, CA: Sociology Press.

Glasser, B. G., & Strauss, A. (1967). *The discovery of grounded theory: Strategies for qualitative research.* Chicago: Aldine Publishing Company.

Heitin, L. (2014, February 25). *Computer Science: Not Just an Elective Anymore*. Retrieved from Education Week: http://www.edweek.org/ew/articles/2014/02/26/22computer_ep.h33.html

Kozulin, G. A. (2003). Vygotsky's educational theory and practice in cultural context. In J. Haenen, H. Schrinemakers, & S. tufkens, *Sociocultural Theory and the Practice of Teaching Historical Concepts* (pp. 246-266). Cambridge University Press.

Kuo, E. (2012). How Students Blend Conceptual and Formal Mathematical Reasoning in Solving Physics Problems. *Science Education, Vol. 97, No. 1*, 32-57.

Larkin, J. (1980). Expert and Novice Performance in Solving Physics Problems. *Science, New Series, Vol. 208, No. 4450*, 1335-1342.

Lawler, R. W. (1997). *Learing with Computers.* Intellect Ltd.

Lichtman, M. (2013). *Qualitative Research in Education: A User's Guide.* Los Angeles: SAGE Publications.

National Science Foundation. (2011, September 16). *Cyberlearning: Transforming Education*. Retrieved from National Science Foundation: http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503581

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas.* New York: Basic Books, Inc.

Ravitch, D. (2010). *The Death and Life of the Great American School System.* New York: Basic Books.

Saldaña, J. (2009). *The Coding Manual for Qualitative Researchers.* London: SAGE Publications.

Schoenfeld, A. H. (1992). On Paradigms and Methods: What Do You Do When the Ones You Know Don't Do What You. *The Journal of the Learning Sciences, Vol. 2, No. 2*, 179-214.

Schoenfeld, A. H. (1992). On paradigms and methods: What do you do when the ones you know don't do what you want them to? Issues in the analysis of data in the form of videotapes. *Journal of the Learning Sciences*, 179-214.

Sherin, B. (1999). *Common Sense Clarified: Intuitive Knowledge and Its Role in Physics Expertise.* Boston.

Sterman, J. D. (2000). *Business Dynamics, Systems Thinking and Modeling for a Complex World.* Boston: Irwin McGGraw-Hill.

Strauss, A. (1987). *Qualitative research for social scientists.* Cambridge: Cambridge University Press.

Strauss, A., & Corbin, J. (1990). *Basics of qualitative research.* Thousand Oaks, CA: Sage.

Stroup, W. (2008a). *Introbuttons.* Retrieved from Generative Center, The University of Texas at Austin: http://generative.edb.utexas.edu/classes/knl2014sum/in_class/23NLapplet/introbuttons.zip

Stroup, W. (2008b). *NetLogo Literacy Activities.* Retrieved from Generative Center, The University of Texas at Austin:

http://generative.edb.utexas.edu/classes/knl2014sum/in_class/23NLapplet/introbuttons.zip

Stroup, W. M., & Wilensky, U. (2014). On the Embedded Complementarity of Agent-Based and Aggregate Reasoning in Students' Developing Understanding of Dynamic Systems.

Vygotsky, L. S. (1987). *The Collected Works of L. S. Vygotsky.* New York: Plenum Press.

Walkington, C. (2011). *Bridges and Barriers to Constructing Conceptual Cohesion Across Modalities and Temporalities: Challenges of STEM Integration in the Precollege Engineering Classroom.*

Wilensky, U. (1999). *NetLogo.* Retrieved from Center for Connected Learning and Compter-Based Modeling: http://ccl.northwestern.edu/netlogo