Making Meaning of Angles and Degrees: Art and Programming the LogoTurtle

Abstract:

Exploring geometry by making art with Logo programming using a physical floor turtle allows for deeper comprehension of mathematics concepts like angles and degrees. A computer screen is too precise to capture the nuances that a floor turtle immediately exposes when students program geometric shapes to create art. The physical construction of a floor turtle is also an ideal STEAM project, combining 3D printing, electronics, microcontroller programming, and debugging. Students who use this analog tool to create art easily develop personally meaningful mathematics challenges that provide for the construction of authentic mathematical knowledge.

Key Words:

art, programming, Logo, robot, geometry, STEAM, 3D printing, electronics, microcontroller, debug, debugging, mathematics, math, constructionist, floor turtle

Contact Information:

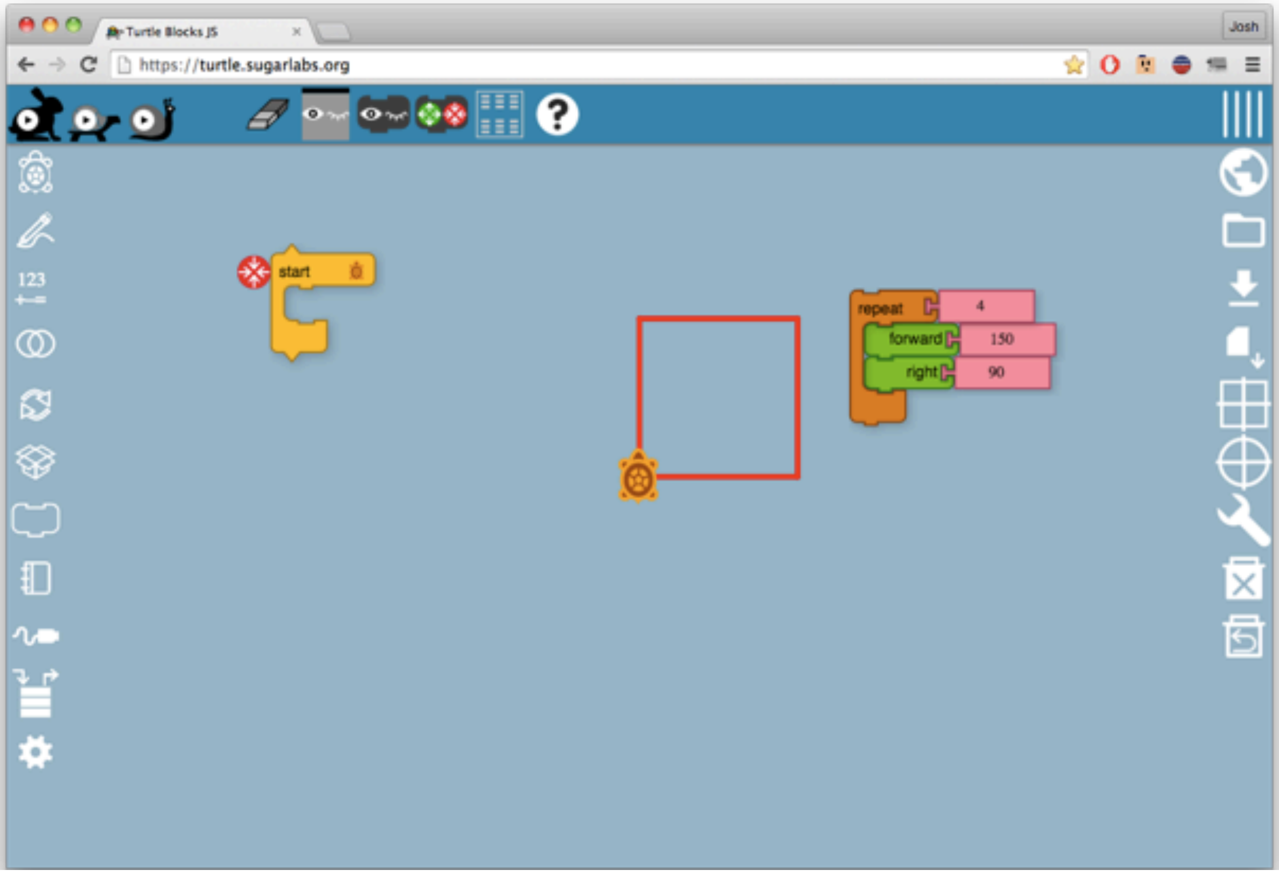
burker.josh05@gmail.com
http://joshburker.com

I. The Pedagogy of Logo Programming: Learning to Think About Thinking

Dr. Seymour Papert decried in 1980:

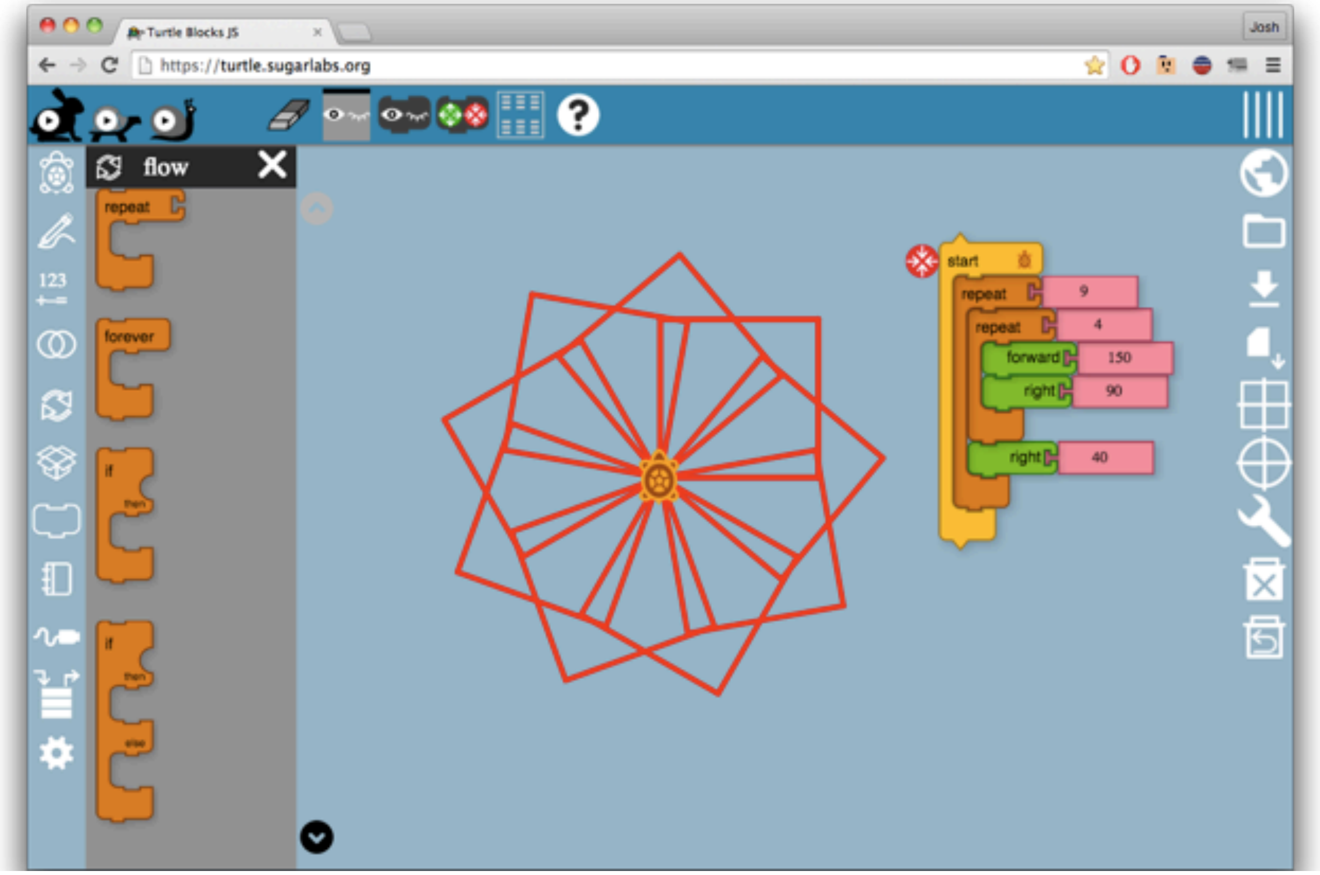
> The intellectual environments offered to children by today's cultures are poor in opportunities to bring their thinking about thinking into the open, to learn to talk about it and to test their ideas by externalizing them. Access to computers can dramatically change this situation. Even the simplest Turtle work can open new opportunities for sharpening one's thinking about thinking: Programming the Turtle starts by making one reflect on how one does oneself what one would like the Turtle to do. Thus teaching the Turtle to act or to "think" can lead one to reflect on one's own actions and thinking. And as children move on, they program the computer to make more complex decisions and find themselves engaged in reflecting on more complex aspects of their own thinking (Papert,1993).

When students learn to program in Logo, they learn to think about the process of moving the turtle around a two dimensional plane and in doing so learn about the process of thinking through a series of movements that the turtle is to execute. Learning to program the turtle to move efficiently develops in the student the ability to think algorithmically. Brian Silverman (Constructing Modern Knowledge Faculty, 2015) explained the process as expressing a design that expresses over time (Figures 1 and 2).



[Burker-CTCLogoTurtlePaper1.png]

Figure 1: A Turtle Blocks Design.



```
start
  repeat 9
    repeat 4
      forward 150
      right 90
    right 40
```

[Burker-CTCLogoTurtlePaper2.png]

Figure 2: The same Turtle Blocks design expressed (rotated 40 degrees) over time (repeated 9 times).

Learning to program Logo is a physical experience: “working with the Turtle mobilizes the child’s expertise and pleasure in motion,” (Papert, 1980) using the student’s “well-established knowledge of ‘body-geometry’ as a starting point for the development of bridges into formal geometry.” Dr. Cynthia Solomon, Papert, and others like Molly and Daniel Watt emphasized the importance of the student’s body in space. The student was instructed to relate one’s personal position and movement to that of the turtle. The act of “playing turtle” encouraged mindful movement: “While walking, observe the number of steps you take, the direction in which you are heading, the sequence of the instructions you are giving yourself to follow, and how you decide when to stop” (Watt & Watt, 1986). As Papert explained, “This method (which includes the advice summed up as ‘play Turtle’) tries to establish a firm connection between personal activity and the creation of formal knowledge” (1980). Rewarding children who purposefully move about the room as they seek to create their Logo designs is contrary to the approach of keeping students tethered to their desks working on abstract math problems.

The physical component of composing and debugging procedures, fittingly, was often executed on robotic Floor Turtles. “This ‘floor turtle’ has wheels, a dome shape, and a pen so that it can draw a line as it moves” (Papert, 1980). However, it was unwieldy, described as “a yellow robot shaped rather like R2D2 and, like him, mounted on wheels,” and “almost as big as the children who were using it, connected by wires and telephone links to a faraway computer that filled a room” (Papert, 1993). As terminal and later personal computer screens became less expensive and more ubiquitous, students moved to “‘screen Turtles,’ which they program[med] to draw moving pictures in bright colors” (Papert, 1980).

Programming Logo has always been about debugging. The simplest way to determine “‘why the Turtle did that dumb thing,’” is to “Play Turtle” (Papert, 1980). However, learning to debug “computational procedures” (Papert, 1980) is not an easy chore. “It always takes time to trap and eliminate bugs,” (Papert, 1980) and furthermore, “It always takes time to learn necessary component skills.” This trade-off of time versus “productivity” is contrary to the way computer science is taught, according to Brian Silverman.

> "The idea of mucking about in programming is, for better or for worse, not very popular in grown up computer science. Programming these days has become more of an engineering discipline with the idea being that you should do things once and do it right. Logo is and always was about debugging. Before the do it right stage there are always dozens if not hundreds of do it not-quite-right stages." (LogoTurtle, 2016).

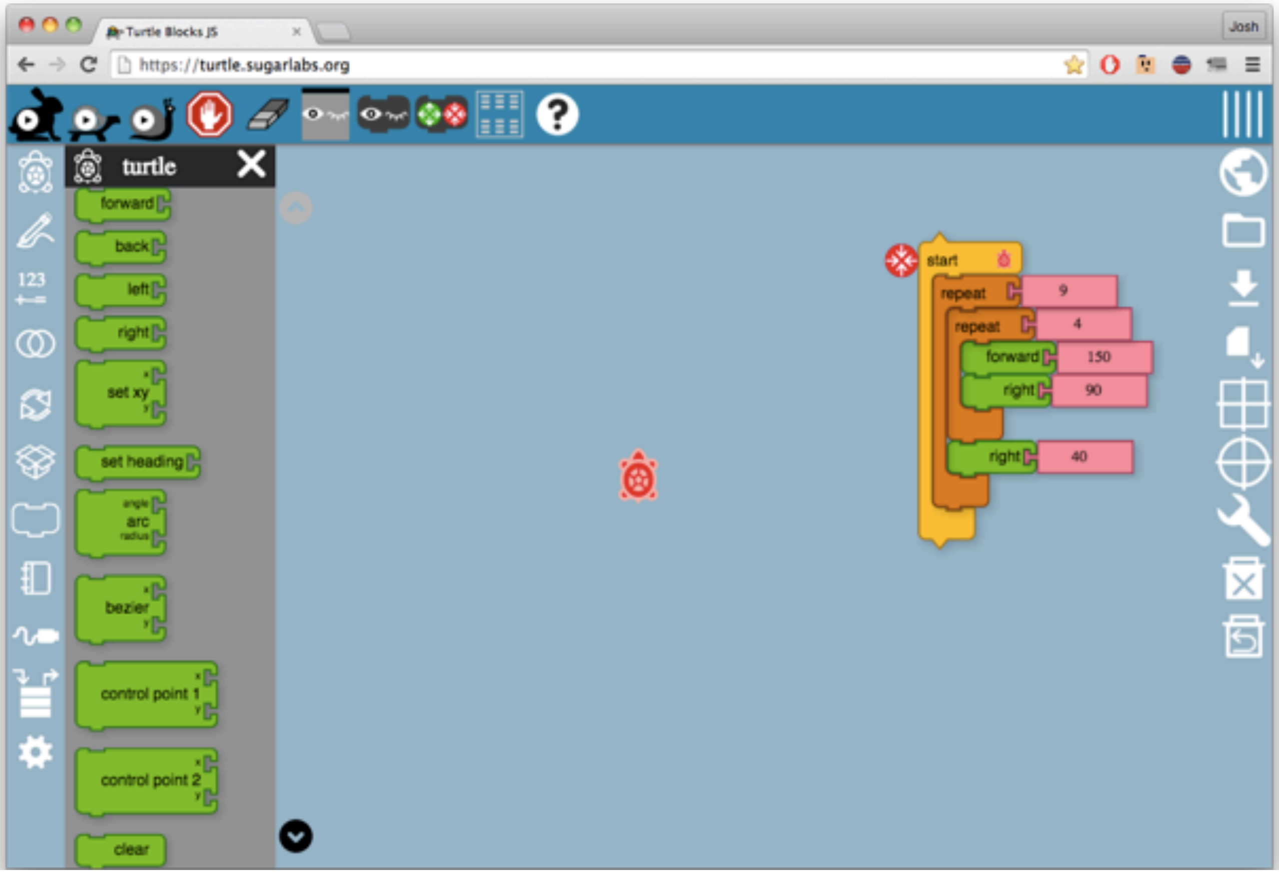
In Logo, “the child is not criticized for an error in drawing,” (Papert, 1980) because there is value in debugging. The “programmer is encouraged to study the bug rather than forget the error” (Papert, 1980). In Logo programming, “One does not expect anything to work at the first try” (Papert, 1980), but at the same time, “One does not judge by standards like ‘right—you get a good grade’ and ‘wrong—you get a bad grade.’” Instead, the challenge is how one can “fix” the mistake (Papert, 1980), knowing that “to fix it one has first to understand what happened in its own terms.” Logo programming embraces mistakes and learning to fix the mistake as the basis for learning.

Finally, Logo programming is particularly well suited to the creation and exploration of geometric art. The particular type of geometry made possible by Logo programming, “Turtle geometry” (Papert, 1980), is “a kind of geometry that is easily learnable and an effective carrier of very general mathematical ideas.” Educators can use Logo as a “microworld,” (Papert, 1980), or a ‘place,’ a ‘province of Mathland,’ where certain kinds of mathematical thinking could hatch and grow with particular ease.” Instead of abstract concepts, students programming the turtle’s movements in Logo can explore their hypotheses about angles, degrees, arcs, rays, and combinations thereof.

Learning to program in Logo changes the use of a computer from a consumption device to a creative tool where it is "used as a mathematically expressive medium, one that frees us to design personally meaningful and intellectually coherent and easily learnable mathematical topics for children" (Papert, 1980). Students are free to explore Turtle geometry and mathematical concepts free from the typical "constraints" of learning math because the solution to the problem is often physical in nature (by "playing turtle") or arrived at through the encouraged practice of debugging. Programming and debugging personally meaningful designs makes mathematics an artistic pursuit.

II. Turtle Blocks: Our Screen Turtle

Turtle Blocks JS (https://turtle.sugarlabs.org) is a browser-based implementation of a Logo programming environment that uses a block-based programming scheme (Figure 3). The ease with which students can connect and uncouple the blocks to make the turtle move and draw on the screen is a great way to get students interested in programming. The barrier to using the tool is lowered because students need not type commands. Instead, the commands are on the blocks.



```
start
  repeat 9
    repeat 4
      forward 150
      right 90
    right 40
```

[Burker-CTCLogoTurtlePaper3.png]

Figure 3: Turtle Blocks in the web browser, with the movement palette open and a constructed procedure of connected blocks.

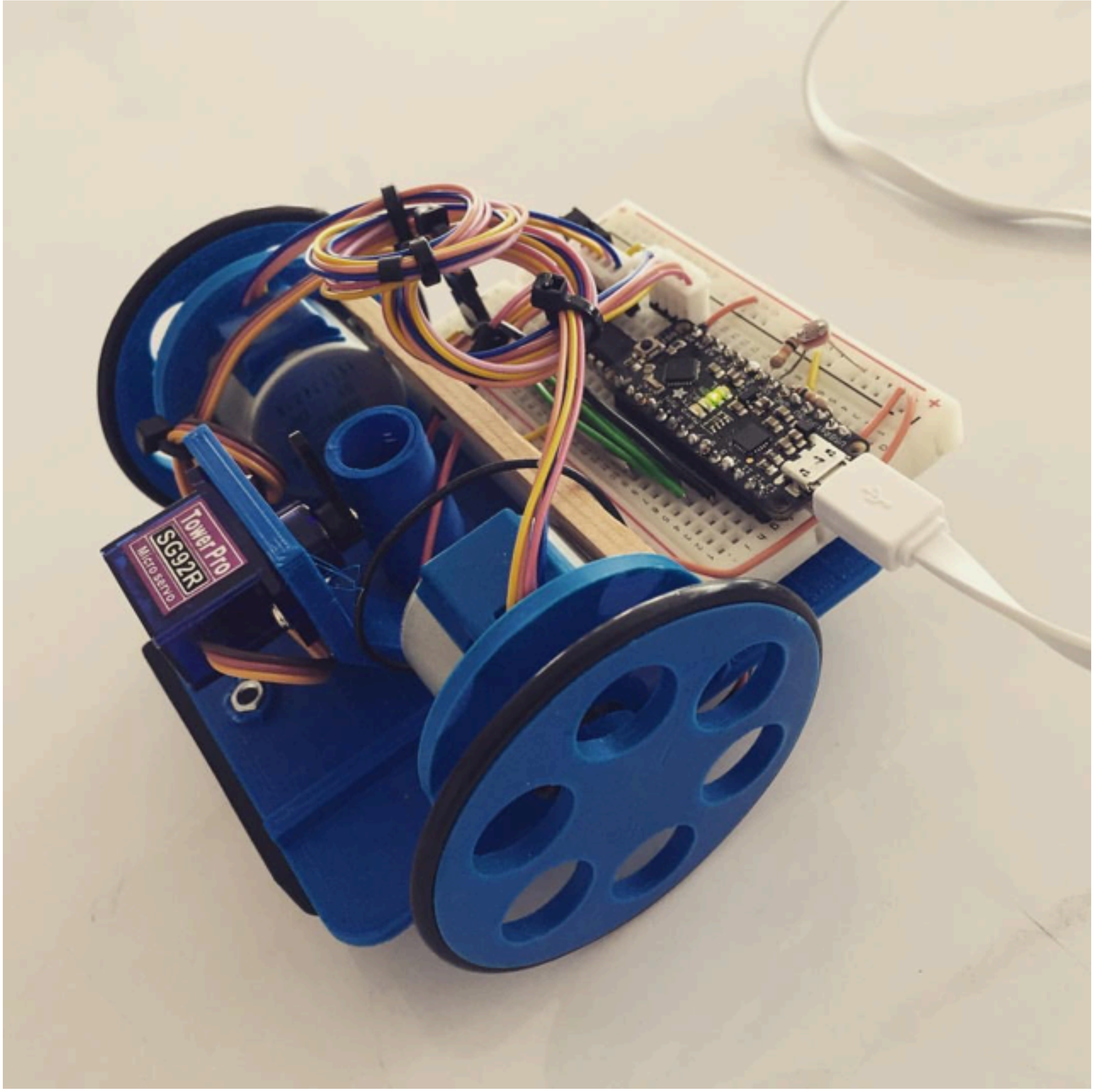
The value, like distance, is connected to a block and can be modified. For example, the “right” movement block comes with the number “90” attached when dragged from the palette. Some students might already know that the 90 refers to 90 degrees, but not knowing the “label” of the value does not prevent the student from trying and using the block as well as modifying the value to observe the results. The block-based programming paradigm strikes a reasonable balance between ease of use at the risk of shallow exploration. Some students might be tempted to enter huge or minute values into the number blocks and fail to realize the intent of changing a value yet remaining within a range of numbers. However, block-based programming is engaging because of its ease of use and its immediate feedback.

Turtle Blocks offers engagement with the possibility of deep exploration provided a few prerequisites for instruction. First, early explorations require some scaffolding. Demonstrate how to make the screen turtle move forward by using the appropriate block and an attached value. Demonstrate that the turtle can turn left or right, but it turns “on a dime” unlike a car, which also moves forward as it turns. Showing the students how blocks can be connected to cause the turtle to execute commands in sequence is an important lesson, as is how to disconnect the blocks for debugging. Students should be encouraged to get out of their seats and to walk purposefully around the room to design and debug their Turtle geometry designs, following the advice to put themselves in the “turtle’s shoes.” Students need to be provided ample time for the free exploration of the software and the individual creation of knowledge. Finally, collaboration is vital in Logo programming. Students can help one another “play turtle” or debug by reciting or recording the commands the turtle follows. Furthermore, techniques, procedures, or designs can be shared and “remixed” by other students, creating a common vocabulary within the local Logo community.

Turtle Blocks is free to use and runs in a web browser, so it can be used anywhere there is internet access. Students’ work is saved to the local machine in the browser cache and may also be downloaded to a USB drive or placed on cloud storage. The transportability and accessibility promotes learning outside of the classroom or school: students can continue to learn on their own devices and on their own time, learning beyond the “Hour of Code” as it were. Additionally, the head developer, Walter Bender, has been very supportive of my work with the software and has added features as needed. The lack of cost and portability made this implementation of Logo ideal for the particular setting where I worked with middle school students during the 2015-16 school year, an example of which I share later in the paper.

III. LogoTurtle

The LogoTurtle is a microcontroller-based 3D printed robot that runs Brian Silverman and Eric Nauman's LogoTurtle software (Figure 4) (LogoTurtle, 2016).



[Burker-CTCLogoTurtlePaper4.png]

Figure 4: An assembled LogoTurtle floor robot.

The LogoTurtle itself can be considered a perfect STEAM (Science, Technology, Engineering, Art, and Mathematics) project because of the combination of parts and skills that combine to create the robot. The robot is built around 3D printed parts that were “recycled” from another project found on the internet (LogoTurtle, 2016). The microcontroller is a simple, inexpensive computer that is not without some limitations: for example, it is unable to handle floating point math. The electronics are hand assembled by the LogoTurtle’s “owner.” An illustrated set of directions (http://joshburker.com/logoturtle/LogoTurtle.html) provides a bill of materials, assembly directions, a schematic diagram, and tips on how to calibrate and run the first procedure from the LogoTurtle (LogoTurtle, 2016). With minimal electronics skills, such as assembling a circuit on a breadboard and soldering, anyone can assemble a LogoTurtle for about US $60 in electronics parts.

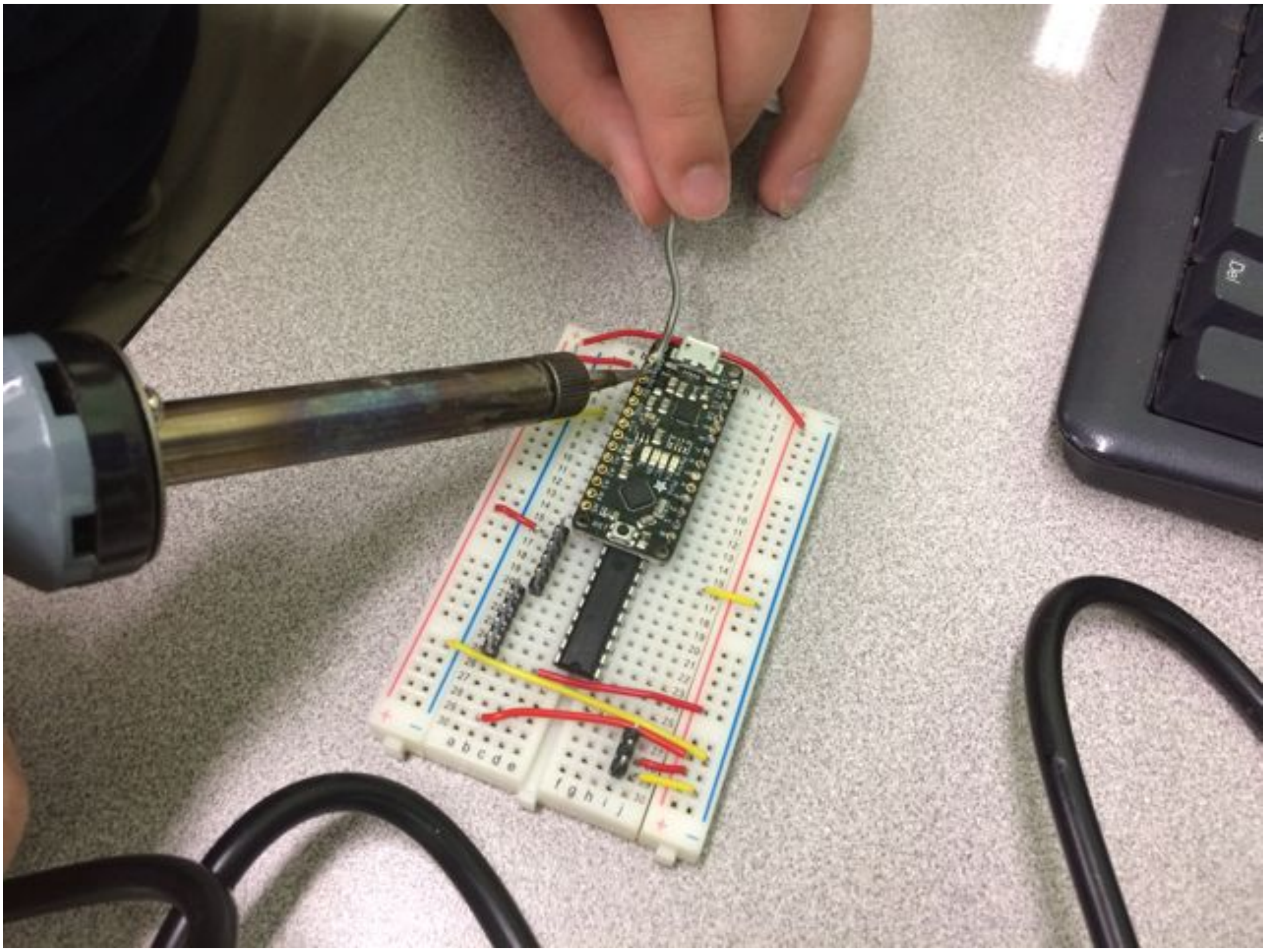
The LogoTurtle uses a text-based programming environment. LogoTurtle encourages Brian Silverman’s belief that interesting designs can emerge from fewer than ten lines of code (Constructing Modern Knowledge Faculty, 2015). Programmers of LogoTurtle aspire to create “elegant” code that is procedural. Logo teachers “describe the process of making the computer remember a procedure [...] *teaching the computer a new command*” (Watt & Watt, 1986). By combining procedures in a strategy called “structured programming,” (Papert, 1980) the programmer can create complexity from small parts that are modular and easy to debug.

The LogoTurtle is a tool that lets the user create work that he or she would otherwise be incapable of producing. The LogoTurtle, a computer controlled robot, allows for both relative precision and repetition when creating art. Because the LogoTurtle can be programmed to move forward and backward any distance and can turn left or right as well as drive in arcs with user-defined degrees and radius, the programmer can use the LogoTurtle to physically draw quite precisely according to the written program. Additionally, the ability to accurately repeat a design countless numbers of time is not something of which a human artist, except those with the highest level of training, is typically capable.

However, the LogoTurtle is also an imprecise tool. The LogoTurtle software, though in use in a number of different schools, is still a first version release. Shortly after the official release, I noticed a “drift” that occurs during repetition and turns. When I did not hear back from Brian Silverman about the “bug,” I realized instead I had discovered the need to “muck around” with the math that I was using with the LogoTurtle. Unlike the screen turtle of Turtle Blocks, the LogoTurtle hardware is not exact. Programming an “ideal” design on the LogoTurtle is a process of debugging and “mucking” with the mathematics, namely angles and degrees, needed to produce the design. One cannot simply move a screen Turtle design to the LogoTurtle and expect the designs to look identical. The lack of floating point mathematics in the microcontroller used by the LogoTurtle means that Silverman and Nauman needed to calculate pi, for turning and the arc commands, without using decimals. The result is that a 90 degree turn on the screen turtle might not translate to a 90 degree turn on the LogoTurtle. Additionally, the friction of the pen and wheels on the paper, as well as in the stepper motors themselves, require additional “mucking” with the math to create the desired design. Finally, the electronics add a layer of imprecision, with two stepper motors running independent of one another on four AA batteries that wear down over time and affect the stepper motor performance. Though imprecise, the LogoTurtle can be potentially upgraded through its software, and hardware “hacks” such as a wood shim to create a consistent 112mm wheel base across different LogoTurtles helps make a 90 degree turn on the LogoTurtle a true 90 degrees. The imprecision is part of the learning potential offered by the LogoTurtle, as will be demonstrated.

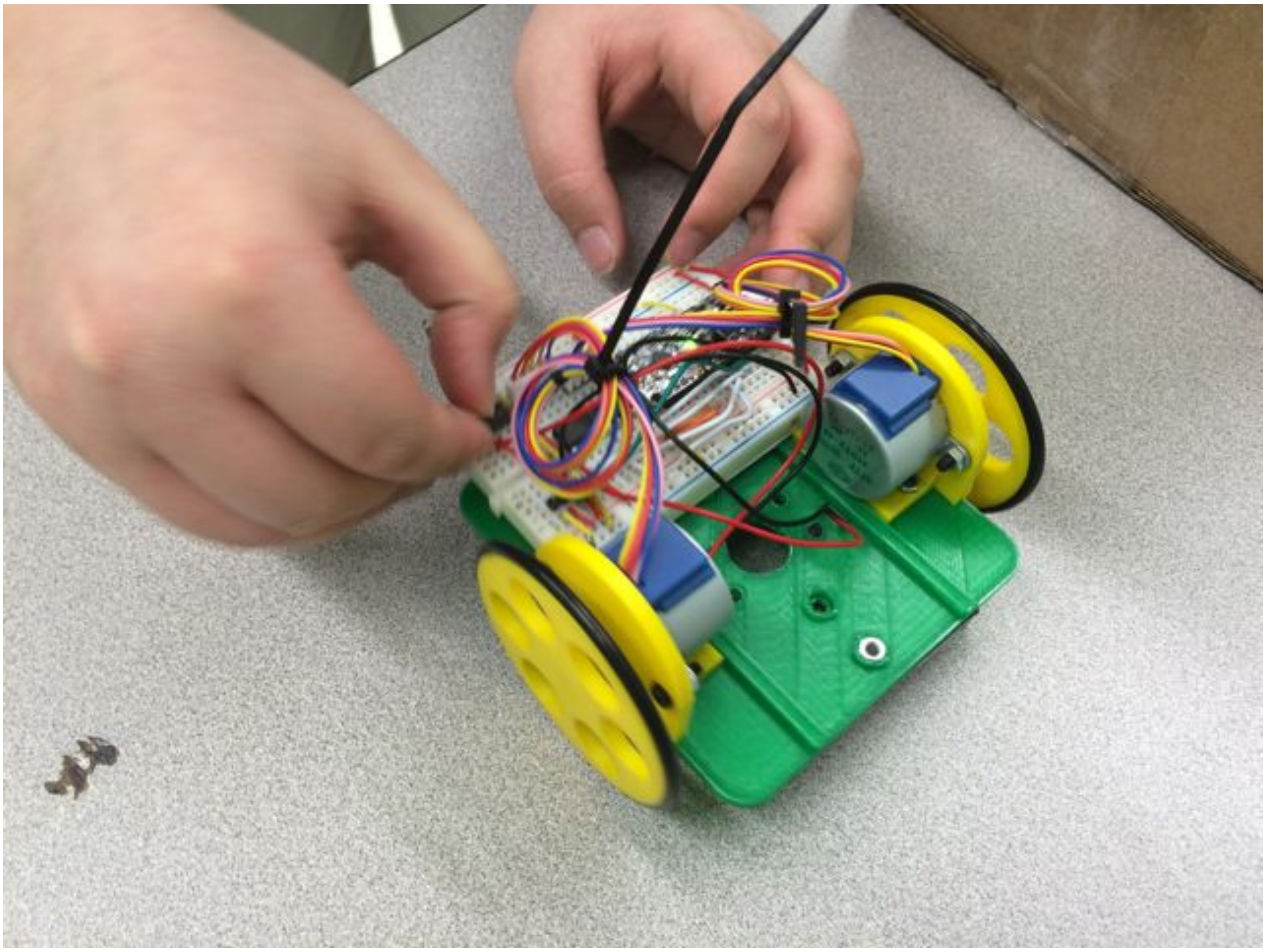
IV. Making Sense of Angles and Degrees: An Example

During the 2015-16 school year I facilitated an after school Logo programming club for middle school students at an inner city charter school in Bridgeport, Connecticut. The students met weekly for an hour to learn Logo programming, starting with Turtle Blocks. As their skills increased the students created iron-on designs from their Turtle Blocks programs. They also learned a workflow for producing 3D printed versions of their Turtle Blocks designs. During the second semester two of the students constructed a LogoTurtle. They learned how to solder, assemble the electronics components, and troubleshoot their assembly (Figures 5 & 6).



[CTC-BurkerLogoTurtlePaper5.jpg]

Figure 5: Student assembling the LogoTurtle electronics and soldering the microcontroller pins.

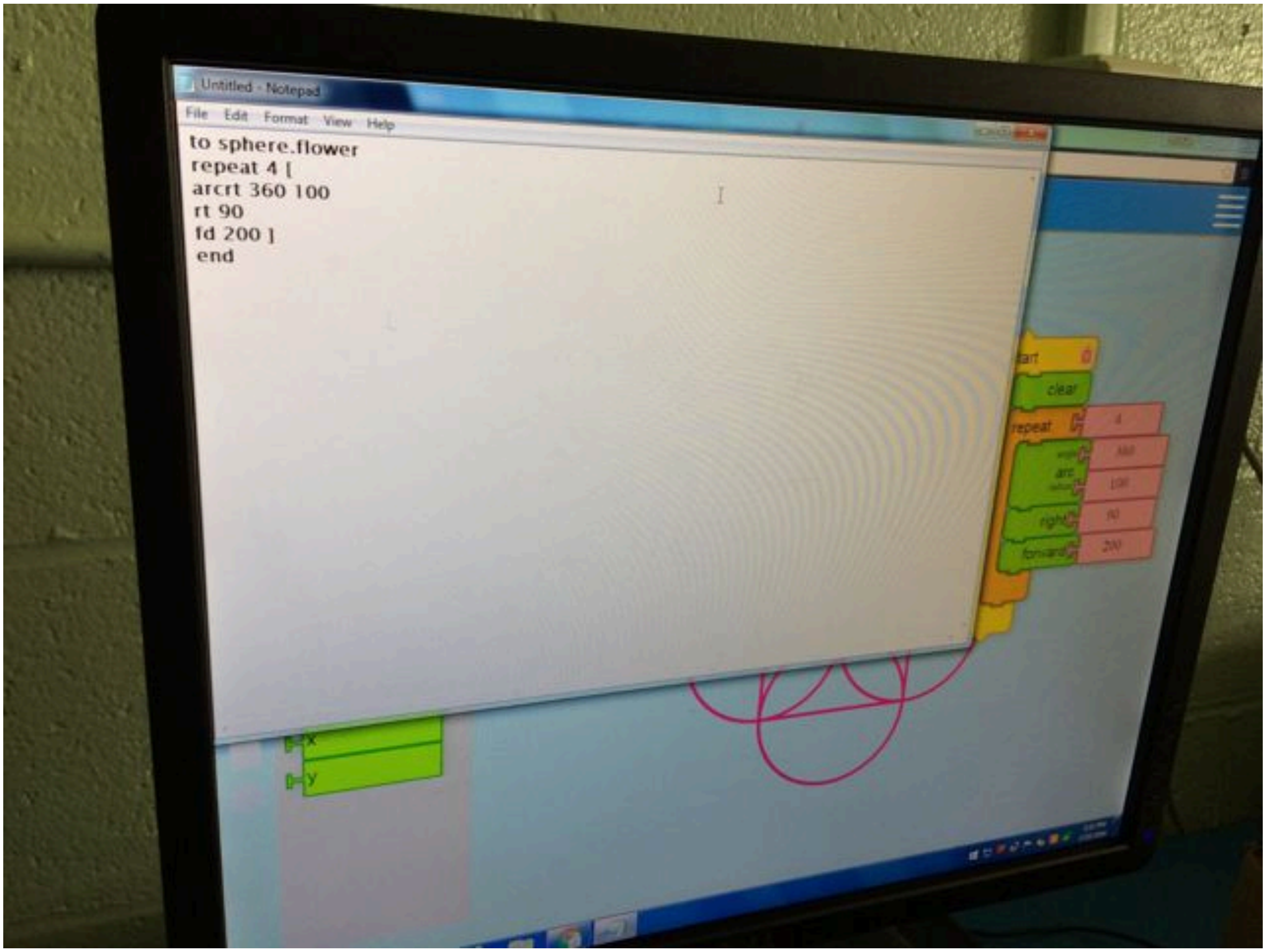


[CTC-BurkerLogoTurtlePaper6.jpg]

Figure 6: 3D printed robot parts with assembled electronics.

Three club members, two boys and one girl, all seventh grade students, worked on creating designs using the LogoTurtle. Interestingly, all three showed the same willingness to iterate and improve upon the design until it reached a level of “perfection” that pleased each of their aesthetic sensibilities.

For these three students, designing a procedure to run on the LogoTurtle started with programming a design in Turtle Blocks. The graphical programming environment and ease of iteration allowed the students to quickly develop and refine a design that could be transferred to the LogoTurtle (Figure 7). The syntax matches between Turtle Blocks and LogoTurtle, but the programmer needs to know the specific vocabulary for the LogoTurtle, as it is a text-based programming environment where spelling matters. Additionally, there is formatting to consider, especially the use of brackets when defining a repeat loop, for example.

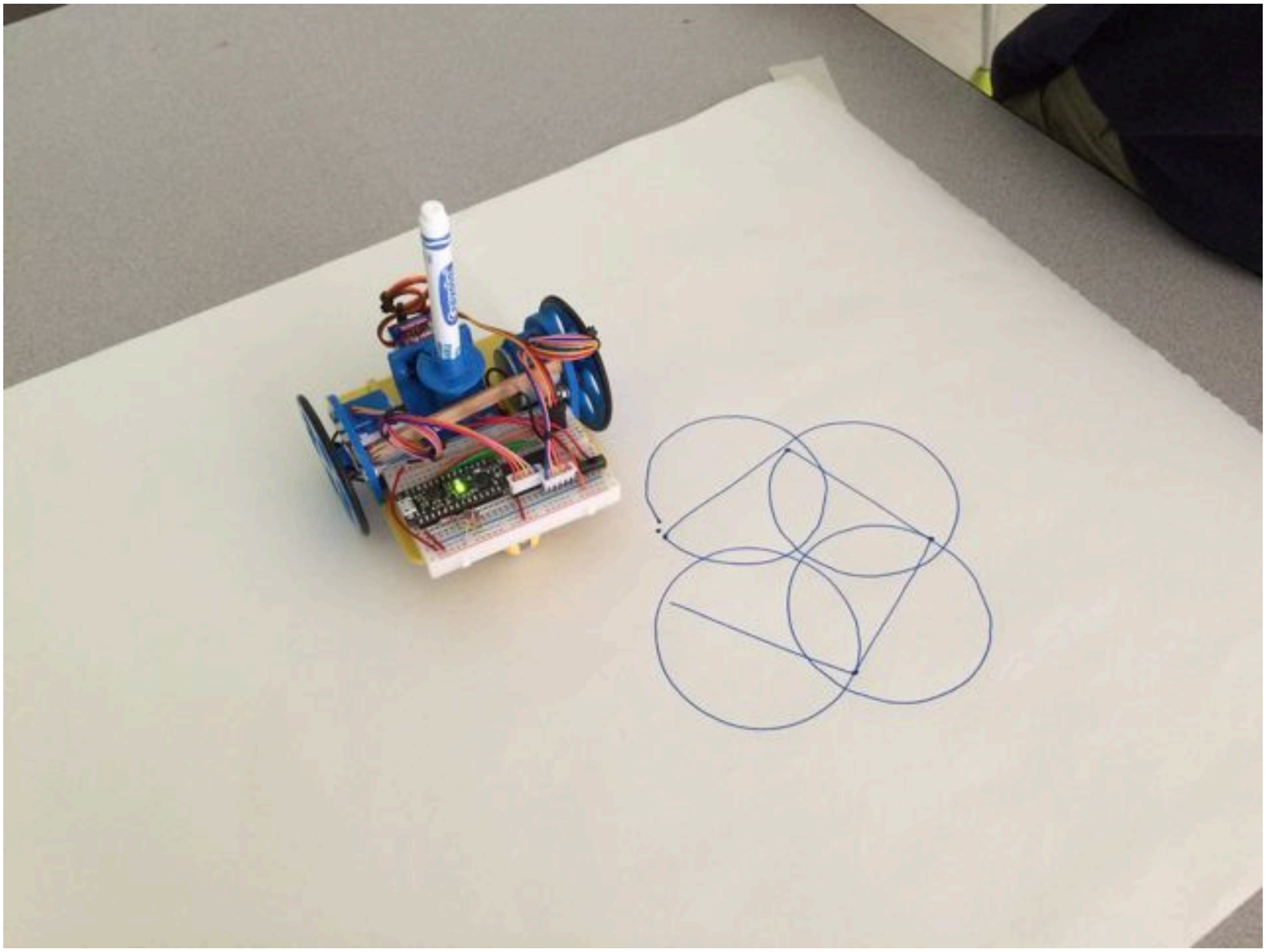


```
to sphere.flower
repeat 4 [
arcrt 360 100
rt 90
fd 200 ]
end
```

[CTC-BurkerLogoTurtlePaper7.jpg]

Figure 7: Transferring Turtle Blocks design to LogoTurtle syntax.

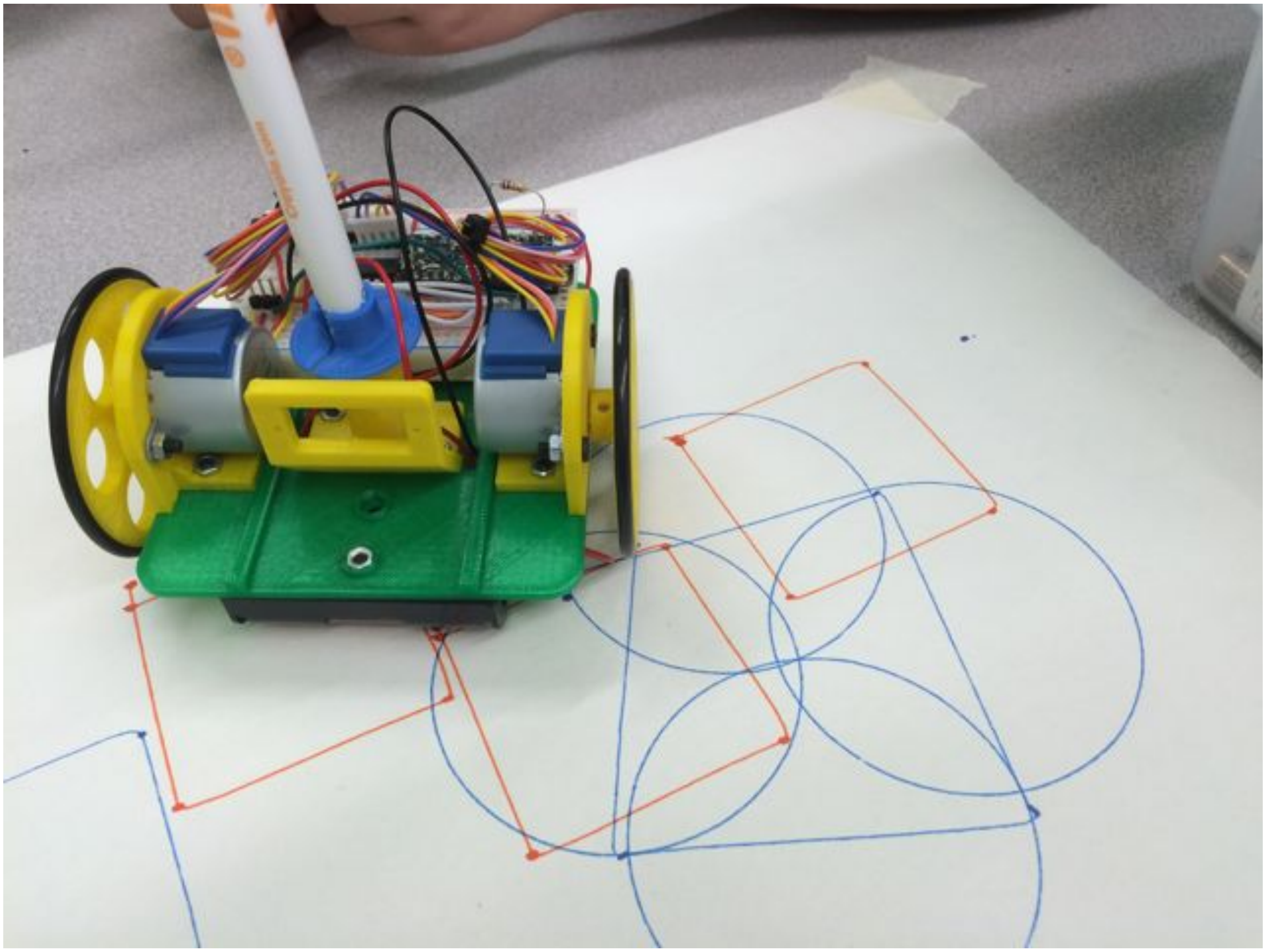
When the student ran the procedure on the LogoTurtle, the imprecision was immediately apparent because the design created by the LogoTurtle did not match that drawn by the screen Turtle in Turtle Blocks (Figure 8).



[CTC-BurkerLogoTurtlePaper8.jpg]

Figure 8: The imprecision of the LogoTurtle is immediately apparent when trying to duplicate a screen Turtle design.

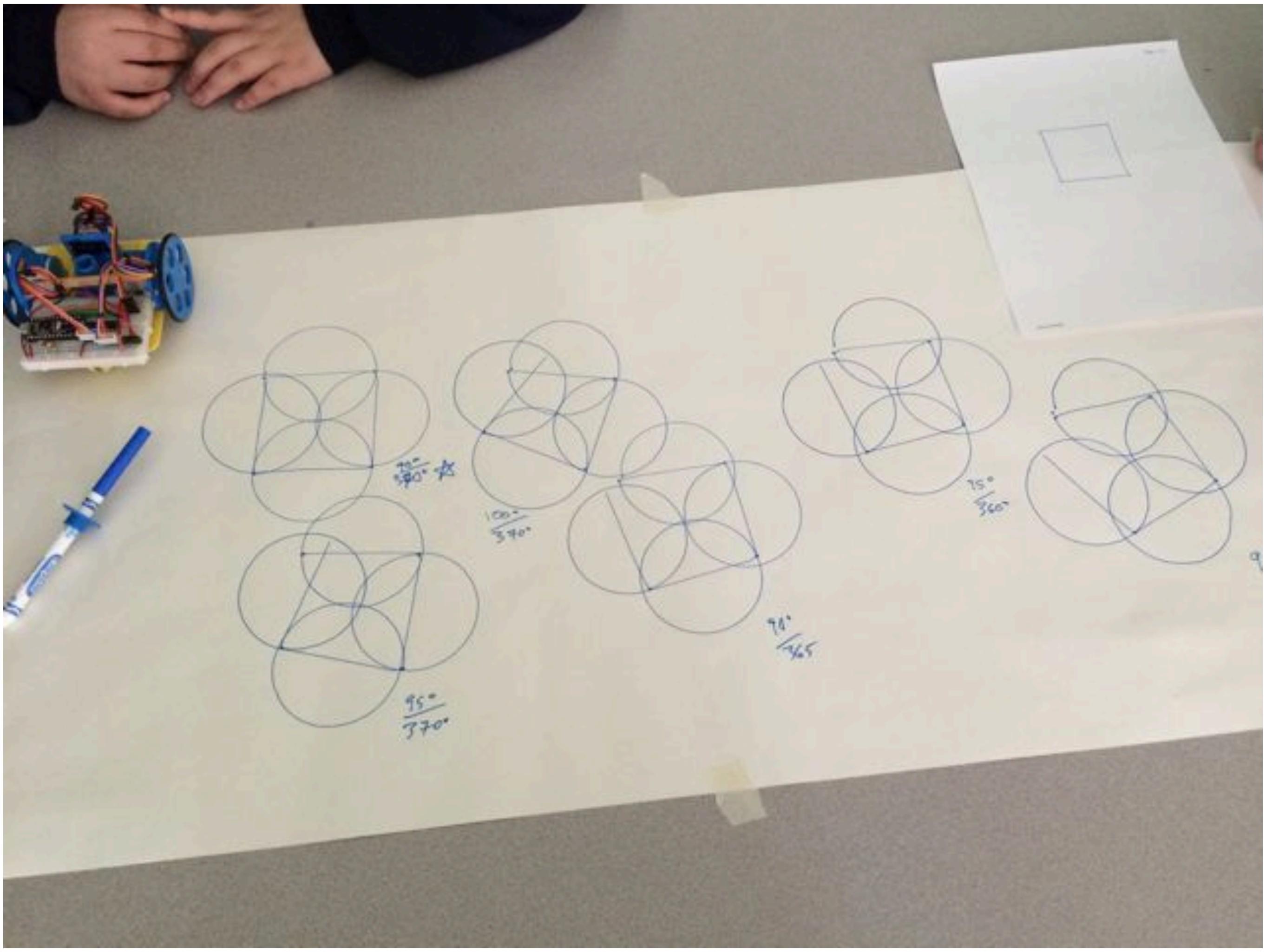
Examination revealed that the angles and degrees driven and drawn by the LogoTurtle did not work as they did on the screen. The student and I checked the calibration of the LogoTurtle, confirming that it could draw a square with 90 degree corners (Figure 9). The act of troubleshooting, of breaking the problem down into smaller parts and isolating the cause of the discrepancy, emphasized to the student the importance of debugging the hardware as well as the software.



[CTC-BurkerLogoTurtlePaper9.jpg]

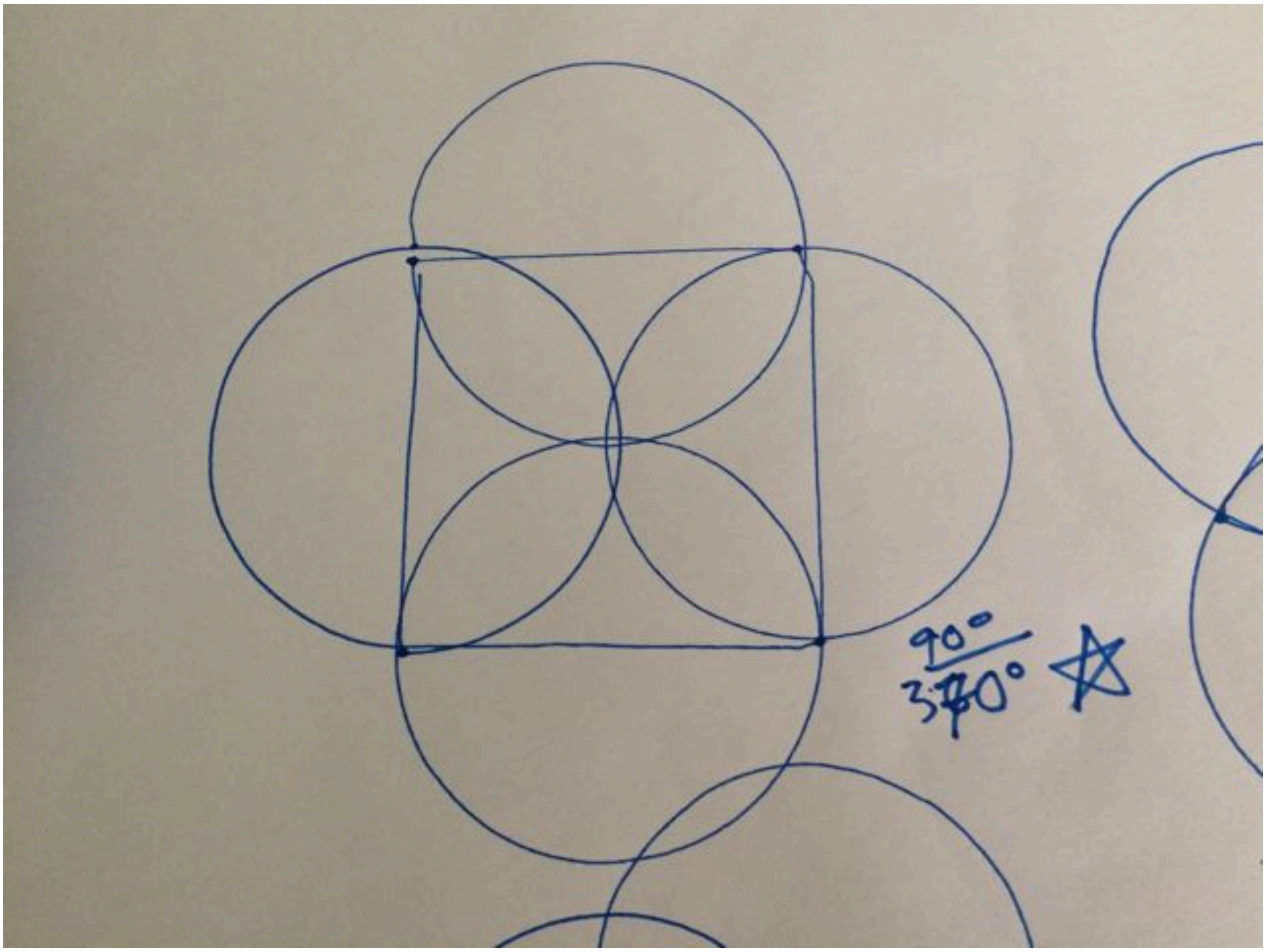
Figure 8: Calibration squares indicating the LogoTurtle is properly calculating and turning 90 degrees.

Rather than give up in frustration, the student chose to iterate through six versions of the design in order to get the math problem “right” (Figure 9). The student noticed that when programmed to draw a 360 degree arc, the LogoTurtle was a little short of 360 degrees. Likewise, the student “mucked with” the 90 degree angle that the LogoTurtle turned in the design. The student persisted until the angles and degrees programmed in the procedure caused the LogoTurtle to draw an almost exact replica of the screen Turtle design (Figure 10).



[CTC-BurkerLogoTurtlePaper10.jpg]

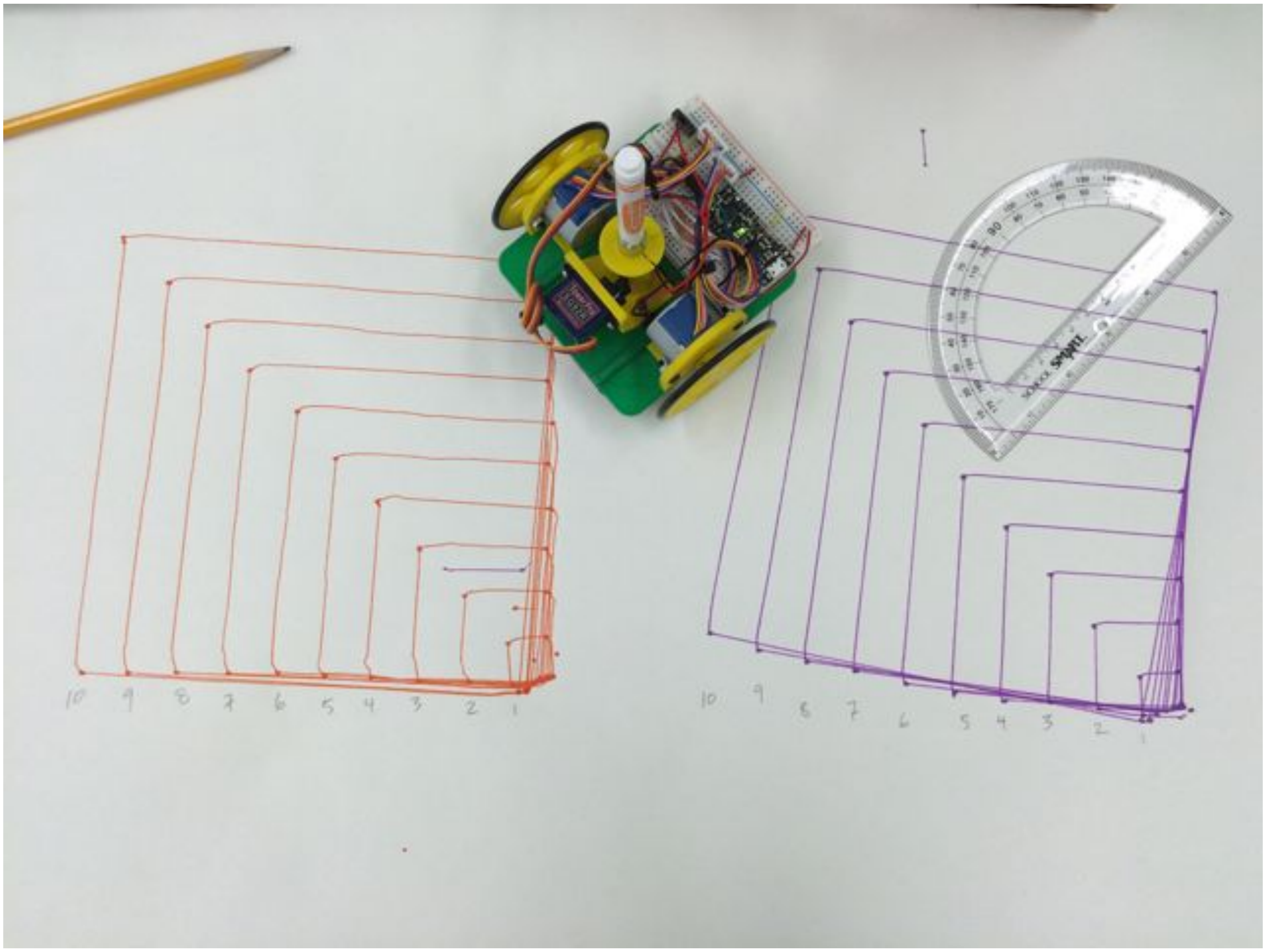
Figure 9: Six iterations of the design with different angles and degrees in each procedure, working towards the idealized screen Turtle design programmed in Turtle Blocks.



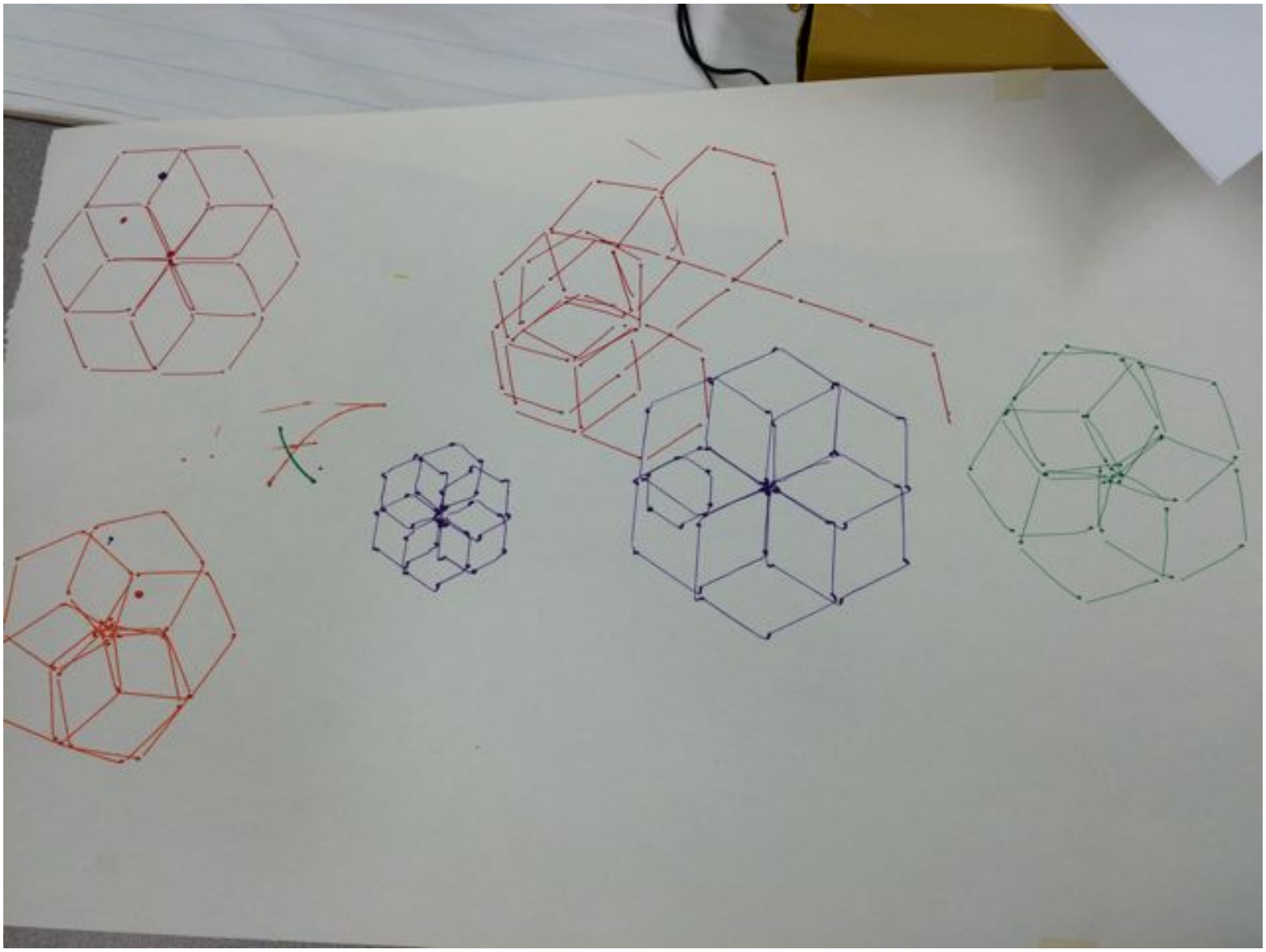
[CTC-BurkerLogoTurtlePaper11.jpg]

Figure 10: The sixth iteration of the design, and as close to the "perfect" screen Turtle design programmed in Turtle Blocks as the student felt the LogoTurtle design needed to be.

In working towards programming this idealized version of the design on the LogoTurtle, this student constructed mathematical knowledge while creating his art. He "played" with angles and degrees to work towards the "perfect" version of the drawing, learning the true meaning of an angle or degree and their role in the creation of the design. Lest we think this student particularly perseverated on creating a design, other students on different occasions demonstrated the same tenacity and willingness to iterate and experiment with angles and degrees in pursuit of their idealized design, too (Figure 11).



[CTC-BurkerLogoTurtlePaper12.jpg]



[CTC-BurkerLogoTurtlePaper13.jpg]

Figure 11: Two other examples where a boy (top) and a girl iterated on a design until it approached the "perfection" they achieved with a screen Turtle in Turtle Blocks.

The LogoTurtle creates the conditions for students to embrace iteration and construction of mathematical knowledge. How many students choose to do a math problem they get "wrong" six times until they get it "correct?" Moreover, how many students can critically assess their own geometric mathematical designs and have the tools needed to correct the problems in their math in order to "solve" the problem? Combining art with a mathematic and scientific challenge encouraged iteration and construction of mathematical knowledge about angles and degrees in these three students. The work these three students completed with the aid of the LogoTurtle sets them apart from their peers:

> Thus, they are prepared for all the many formal topics—geometry, trigonometry, drafting, and so on—in which the concept of *angle* plays a central part. But they are prepared for something else as well, an aspect of

> the use of angular measure in our society to which the school math is systematically blind.
>
> One of the most widespread representations of the idea of angle in the lives of contemporary Americans is in navigation. Many millions navigate boats or airplanes or read maps. For most there is a total dissociation between these *live* activities and the *dead* school math. We have stressed the fact that using the Turtle as metaphorical carrier for the idea of angle connects it firmly to body geometry. We have called this body syntonicity. Here we see a *cultural syntonicity:* The Turtle connects the ida of angle to navigation, activity firmly and positively rooted in the extra school culture of many children. And as computers continue to spread into the world, the cultural syntonicity of Turtle geometry will become more and more powerful (Papert, 1980).

Working with the LogoTurtle values aesthetics and encourages collaboration and interpersonal relations, two important considerations when creating technology projects that girls will enjoy and learn from (Hanor, 1998). The ability to choose color or shapes, Hanor found, encourages girls to engage in technology-based projects (1998). Using LogoTurtle to program and draw geometric shapes and patterns might produce aesthetic experiences that "are integrated experiences that incorporate perceptual and cognitive pleasures derived from repetition, playfulness, daydreaming, and fantasy" (Hanor, 1998), important considerations when designing technology curriculum that includes girls' interests. While girls are more interested in using computers to produce tangible results (Bhargava et al., 1999), unstructured time is important for girls to give them the opportunity to "mess around" with technology (Hanor, 1998). This approach is compatible with the idea of "mucking around" with math and the resulting designs created by the LogoTurtle. The communication required in debugging a design on the LogoTurtle provides girls and boys with the opportunity to collaborate and socialize as they seek their solutions; the technology acts as a go-between device that enhances the quality of conversation. The collaborative possibilities offered by students' work with the LogoTurtle is beneficial to both genders but is particularly appealing to girls who might otherwise be intimidated by technology. The value placed on aesthetics, through the creation of drawings programmed on the LogoTurtle, values girls' and boys' aesthetic choices and helps the students develop as artists.

These three students' work with the LogoTurtle provided them with important lessons about the nature of angles and degrees. By focusing the mathematical investigation through the lens of art, the students felt a personal attachment to the problem because it valued their aesthetic perspective and judgement. The willingness to "muck around" with the math, when framed in the context of programming the LogoTurtle to draw exactly what the student envisioned and designed, promoted iteration and reflection on the mathematics that created the designs. Each student emerged from their own mathematic challenge with the

LogoTurtle a better programmer, mathematician, and electronic artist.

V. Obstacles and Workarounds

This project, like any STEAM project in schools, is not without obstacles. The LogoTurtle is not an out of the box solution for many teachers and students. Unlike other programmable robots, the LogoTurtle must be assembled by its user, which requires secondary knowledge like electronics assembly, soldering, and perhaps 3D printing if the user decides to produce her own parts for the LogoTurtle. However, a complete set of documentation (http://joshburker.com/logoturtle/LogoTurtle.html) is available, so anyone can create and assemble a LogoTurtle. The 3D printed parts can be ordered directly from the repository where the digital files are hosted. The project could easily be built during a workshop where people with different skills sets or abilities could collaborate to create a single LogoTurtle. Additionally, an online community (https://groups.google.com/forum/#!forum/logo-turtle) exists where users can find assistance troubleshooting their LogoTurtles. The hurdles of LogoTurtle fabrication are not too difficult to clear.

The LogoTurtle is a “hard fun” project, a concept that Dr. Seymour Papert described as such:

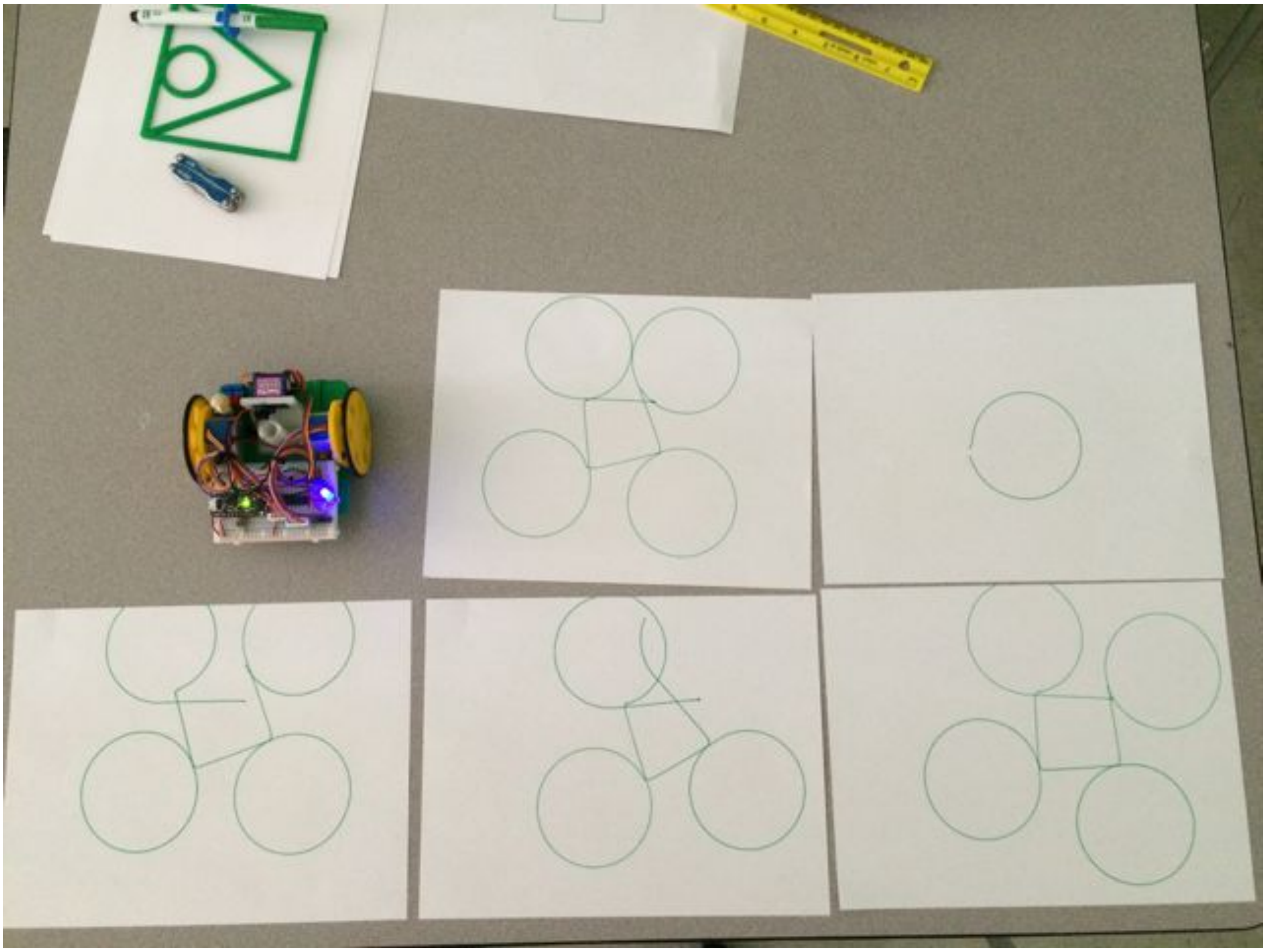
> We learn best and we work best if we enjoy what we are doing. But fun and enjoying doesn’t mean ‘easy.’ The best fun is hard fun. Our sports heroes work very hard at getting better at their sports. The most successful carpenter enjoys doing carpentry. The successful businessman enjoys working hard at making deals (Martinez & Stager, 2013).

With debugging at its core, the LogoTurtle expects its user to be willing to “muck around” with the math and iterate through the design to be rewarded with a personally satisfying solution. Some may balk at the imprecision of the LogoTurtle, but it is this very limitation that creates opportunities for deep explorations of mathematics in the guise and challenge of of creating beautiful vector art.

Finally, the LogoTurtle flies in the face of behaviorists who demand more data and “accountability” in schools, achieved through testing on abstract, memorized information that might be contextually void for the students. The LogoTurtle uses technology as more than a means to “drill” students on mathematical knowledge. Recall Papert’s warning:

> In many schools today, the phrase "computer-aided instruction" means making the computer teach the child. One might say the *computer is being used to program* the child. In my vision, *the child programs the computer* and, in doing so, both acquires a sense of master 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 (Papert, 1980).

The LogoTurtle is meant to be programmed by the child, and the wonderful art produced through the students' construction of authentic mathematical knowledge is beautiful evidence of their learning (Figure 12).



[CTC-BurkerLogoTurtlePaper14.jpg]

Figure 12: Another Turtle Blocks design translated to the LogoTurtle, the math "mucked around" with until the LogoTurtle drew the desired design.

Works Cited:

Bhargava, A., Kirova-Petrova, A., & McNair, S. (1999). Computers, gender bias, and young children. Information Technology in Childhood Education.

Constructing Modern Knowledge Faculty (2015). Retrieved from http://constructingmodernknowledge.com/cmk08/?page_id=224

Hanor, J. H. (1998).Concepts and strategies learned from girls' interactions with computers. Theory into practice.

LogoTurtle (2016). Retrieved from http://joshburker.com/logoturtle/LogoTurtle.html

Martinez, S. & Stager, G. (2013) Invent to learn: Making, tinkering, and engineering in the classroom. Torrance Park, Calif.: Constructing Modern Knowledge Press.

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

Papert, S. (1993) The children’s machine: Rethinking school in the age of the computer. New York: Basic Books.

Watt, M., & Watt, D. (1986). Teaching with Logo: Building blocks for learning. Menlo Park, Calif.: Addison-Wesley Pub.