# De Bits a Átomos: Programación en Bloques de la Tortuga JS y Fabricación Personal en Proyectos de de Jóvenes

## Going from Bits to Atoms: Programming in Turtle Blocks JS and Personal Fabrication in Youth Maker Projects

Josh Burker
Creator of Learning and Discovery Experiences. Westport Library. USA
burker.josh05@gmail.com

**Resumen**
El Software y Hardware deben ser vistos como herramientas en el aula moderna, indistinguibles en importancia y potencial creativo, como los son el lápiz y el papel. Bloques de la Tortuga JS, un ambiente de programación de bloques inspirado en Logo que se ejecuta en un navegador, proporciona un micromundo fácil de utilizar en el que los estudiantes pueden explorar geometría, diseñar a través de iteración, programar y depurar. Los diseños creados en el entorno Bloques de la Tortuga pueden ser descargados como gráficos vectoriales simples (SVG) y posteriormente procesadas para impresión 3D, transformando el diseño digital en una herramienta tangible y funcional. Los estudiantes pueden utilizar el entorno Bloques de la Tortuga en conjunto con una serie de dispositivos de fabricación, incluyendo las impresoras 3D y cortadoras láser, permitiéndoles crear artefactos cada vez más complejas. En este artículo, presento el trabajo realizado con Bloques de Tortuga JS en una academia de verano para jóvenes de secundaria, así como un taller de fabricación para niños de 10 a 13 años de edad.

**Palabras claves**
Construccionismo, Programación, Logo, Fabricación

**Abstract**
Software and hardware should be seen as tools in the modern classroom indistinguishable in importance and creative potential as the pencil and paper. Turtle Blocks JS, a Logo-inspired, block-based programming environment that runs in a web browser provides an easy to use microworld in which students may explore geometry, design through iteration, programming and debugging. The designs created in Turtle Blocks can be downloaded as simple vector graphics (SVG) and subsequently processed for 3D printing, transforming the digital design into a tangible, functional tool. Students can use Turtle Blocks in conjunction with a number of fabrication devices, including 3D printers and laser cutters, allowing them to create increasingly complex artifacts. In this paper, I present the work done with Turtle Blocks JS in a summer academy for middle school boys as well as a workshop for 10–13 year olds in a makerspace.

**Keywords**
Constructionism, Programming, Logo, Fabrication

## I. Turtle Blocks and Classroom Fabrication

In Dr. Seymour Papert's *Mindstorms* he introduces the concept of a "microworld" whose design "makes it a 'growing place' for a specific species of powerful ideas or intellectual structures" (Papert, 1980). Dr. Papert's Logo turtle is a purposefully welcoming "mascot" for the Logo programming language. Papert believed "[t]he idea is that early experience with Turtles is a good way to 'get to know' what it is like to learn a formal subject by 'getting to know' its powerful ideas" (Papert, 1980). Molly and Dan Watt characterize learning Logo in a microworld as "open-ended exploration, not by focused instruction" (Watt & Watt, 1986). In exploring rays and angles through Logo programming and the designs the turtle can be “taught” to draw, children and adults can learn the workings of geometry through hands-on experimentation, debugging, and iteration.

Turtle Blocks JS (https://turtle.sugarlabs.org) is a browser-based implementation of Walter Bender’s Turtle Blocks software project for the One Laptop Per Child XO Sugar operating system. Being browser-based, Turtle Blocks JS will run on any operating system that includes a Java runtime environment. Work can be saved to a local drive and transported to a different computer via a USB drive or other storage device, and projects are also stored to the local browser cache, so a student using the same computer from class to class can expect her work to be available for long-term projects.

Turtle Blocks is based on the Logo programming language and includes the turtle and Logo primitives such as forward, right, left, and down. When Turtle Blocks opens for the first time and subsequently through the use of the “?” help button in the toolbar at the top of the window, a short introduction to the various palettes and the functions of some individual buttons are explained. A separate, rich set of documentation for each block is being developed (Turtle Blocks documentation, 2015). Additionally, a guide that includes "recipes" for constructing specific projects is under development (Turtle Blocks Guide, 2015). Currently, there are samples included in the “Worldwide” folder, accessible from the button on the far right of the top toolbar, then by clicking on the globe icon and scrolling down. Each of these projects, which users can contribute to by sharing their own projects, is remixable simply by opening it and making modifications to the sequence or collection of blocks.

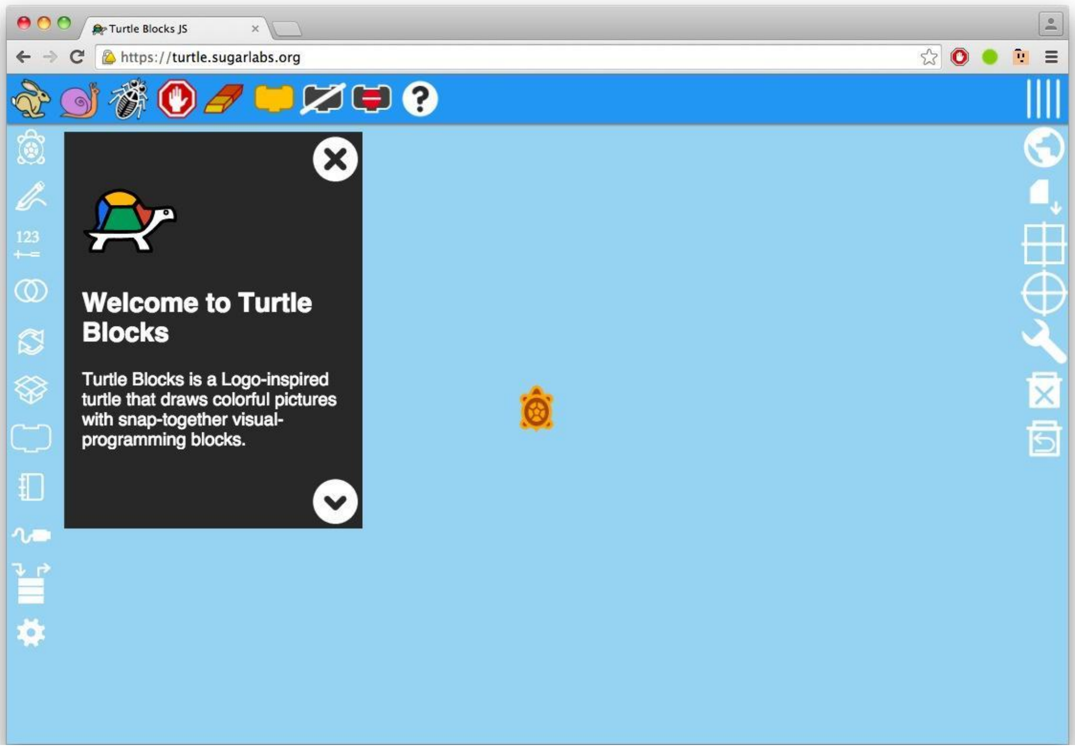

Turtle Blocks JS
https://turtle.sugarlabs.org
123
Welcome to Turtle Blocks
Turtle Blocks is a Logo-inspired turtle that draws colorful pictures with snap-together visual-programming blocks.

Figure 1: The Turtle Blocks environment

When I teach people how to use Turtle Blocks I provide minimal scaffolding, hoping to encourage the users to explore the software for solutions to the problems they encounter trying to create a design. I teach the user how to connect blocks in order for the turtle to execute a sequence of commands. I teach her how to disconnect blocks from one another, by clicking the block she wishes to remove and dragging it downwards from the stack. I also teach people how to name a procedure, by building the blocks within an “Action” stack and naming the stack, which creates a single block from the named collection of blocks. I explain how to "erase" blocks by dragging them towards the bottom of the browser window, which reveals a garbage can into which users can drop unneeded blocks. When I facilitate a project with Turtle Blocks, I do not mention the Worldwide examples, nor do I expect users to read documentation or the guide. However, if a user begins to be frustrated, these resources, as well as my own experience, are available for him or her to use to assist them in getting started.

Another lesson I always teach when introducing people to Turtle Blocks JS is the advice offered by Molly and Dan Watt, to “walk yourself through a path on the floor that corresponds to the path you want the turtle on the screen to follow” (Watt & Watt, 1986). Asking the user to consciously move through a plane, while counting “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) helps the student visualize the design she wishes to create, and provides another context for learning geometry. As Paper asserts in *Mindstorms* (Papert, 1980), “[W]orking with the Turtle mobilizes the child’s expertise and pleasure in motion. It draws on the child’s well-established knowledge of ‘body-geometry’ as a starting point for the development of bridges into formal geometry.” By contextualizing geometry in terms a child can understand, namely through motion in space, it is possible to expose young children to powerful mathematical ideas.

One context in which I have encouraged students to explore Turtle Blocks JS programming is to create a tool to help them create art. Specifically, I have worked with Walter Bender to add some blocks to the turtle’s palette, as well as to modify some blocks, to support the creation of 3D printed designs. I have created a workflow where students’ designs can be transformed into 3D printed designs. These artifacts are tools that allow the students to stamp modeling compound, like PlayDoh, or clay, to create complex geometric figures and make tiles. The same workflow can be applied to using a laser to etch or cut a design. Both projects involve using Turtle Blocks JS to create a tool. This tool then allows students to create art whose complexity would otherwise be unattainable by them.

Personal fabrication, using 3D printers or laser cutters, is increasingly entering schools and libraries, providing accessibility to tools that allow people to transform their digital designs into physical models that exist in the real world. Digital design and personal fabrication should be seen as an update to the 20th century vocational shop skills that were once taught in American schools, like woodshop, metalshop, or pottery. These tools allow for the fabrication of tangible artifacts using a variety of materials. In the case of 3D printing, the material is oftentimes a plastic filament extruded under pressure and heat to build up a design over layers. In the case of a laser cutter, a design is created through removing material, whether through cutting or etching the material with the laser.

Designing a Turtle Blocks JS procedure to be 3D printed needs to take into considerations the limitations of the device and the materials. The diameter of the filament and the extruder itself should influence how thick of a line the turtle draws: if the turtle's line is too thin, it might be physically impossible for the printer to replicate the design. Additionally, the print area of the 3D printer dictates how large of a design can be printed. Consider, too, the size of the person who will be using the design if intended to be used as a stamp: a smaller, 5cm square design fits nicely in a child's hand and is less awkward for them to make a good impression that an 11cm square stamp. A laser cutter has a higher resolution than a 3D printer, but other factors must be considered. If the stamp is too large, it is difficult to ink the stamp without special ink pads. Similarly, a large stamp is difficult to even apply even pressure to stamp a good image. When transferring a digital image to a physical tool, factor these considerations into the design choices.

## II. Two Scenarios of Turtle Art and Fabrication in Learning Environments

In one two and a half hour workshop, held during a five day workshop in association with a makerspace in a public library for eight students ages 10 to 13, students used Turtle Blocks JS to create designs specifically to 3D print as stamps for PlayDoh. Two of the students had previously used Brian Silverman's TurtleArt application, while the others had exposure to block-based programming through Scratch. The designs were to be printed on a MakerBot Replicator in ABS plastic sized approximately five by five centimeters.

Before the students started programming, I demonstrated how to define a procedure in Turtle Blocks and brought to their attention the block named after the action that Turtle Blocks creates. The students were given the prompt, "Learn how to express a sequence of things that express over time" before they started programming. Otherwise, the students were free to experiment to create their knowledge of the software by playing with the blocks instead of following step by step directions how to create designs. Students were encouraged to share with the group any discoveries they made about the behavior of blocks, which was demonstrated to the class with the aid of an LCD projector and screen. Two girls worked closely side by side, creating different designs but sharing discoveries. Students were not given a stop time, and worked on the design until they felt it was finished. The students worked primarily without my assistance, though some ideas were clarified when they questioned me about how a block worked or how a procedure to make a specific move might be constructed.

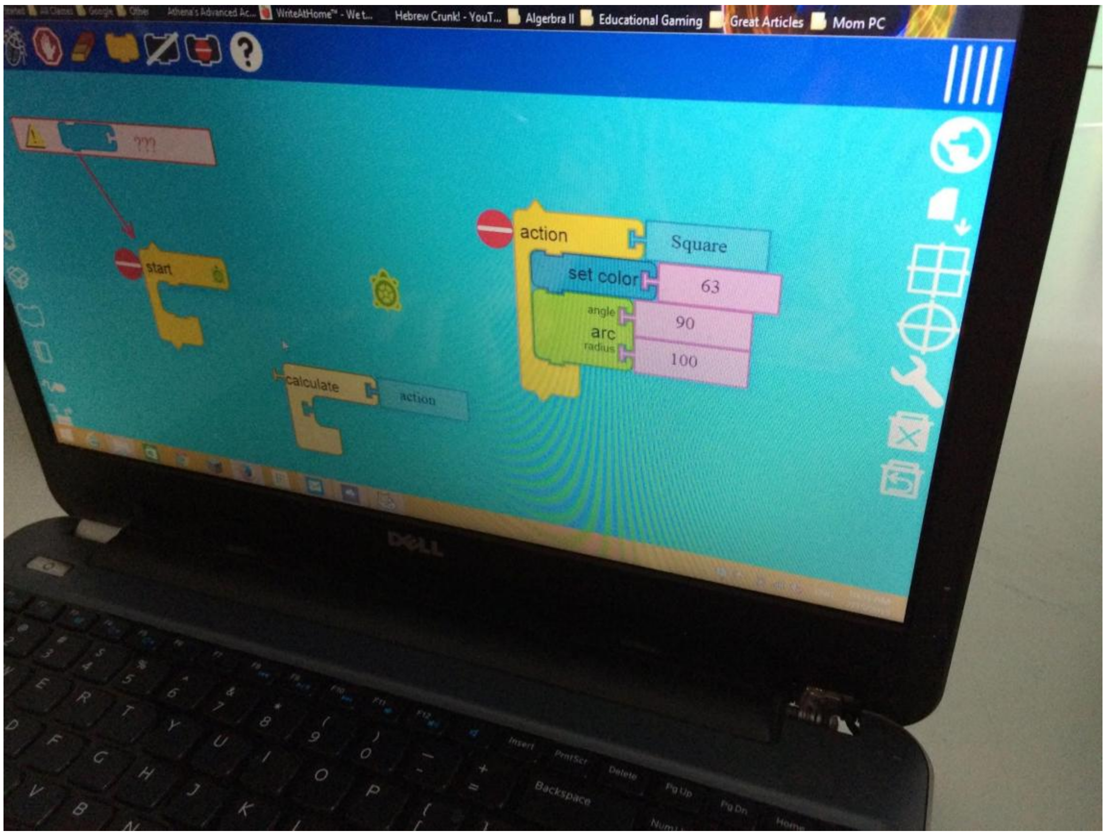
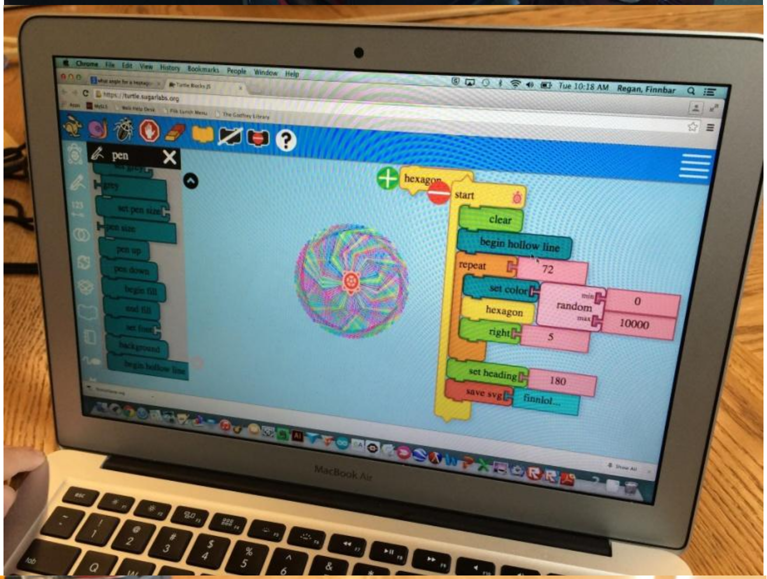
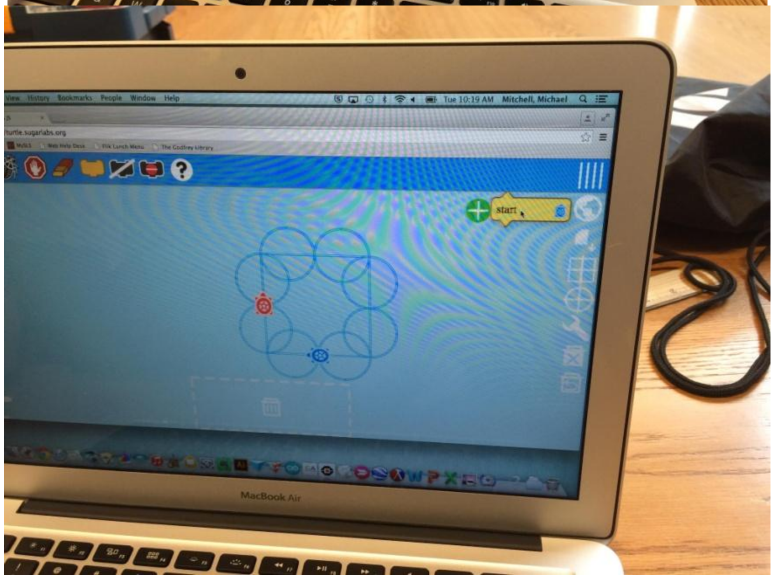


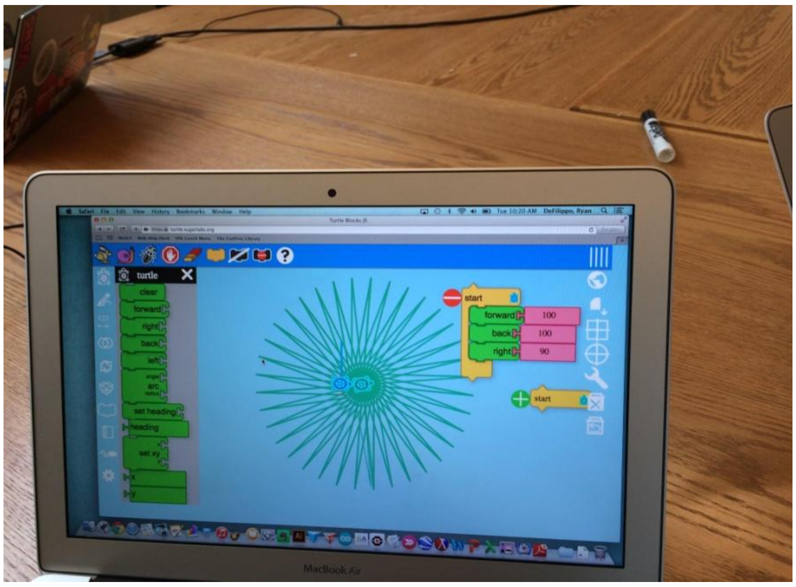
Figure 2: The Action stack

A workshop for middle school boys taught at a private school during a two week long summer technology camp explored programming Turtle Block JS designs to be etched with a laser cutter, a different fabrication technique than the other workshop. The intent was to produce rubber stamps for stamping ink on paper. These students had more extensive experience with Scratch and were programming Tic-Tac-Toe games before my arrival. None had used Turtle Art or Turtle Blocks before. These students were scaffolded with the same introduction to Turtle Blocks JS as in the 3D printing workshop, but without the prompt to express a design that expresses over time. They were encouraged to create a design that was approximately four inches by four inches. Otherwise, the path they took to create the design was entirely theirs to create.

The boys worked independently and with the exception of four of the fourteen students worked independently, instead of collaboratively. The head instructor for the technology camp participated as well and offered his assistance as needed to explain block usage or how to think about constructing a procedure.


pen
set pen size
pen size
pen up
pen down
begin fill
end fill
set font
background
begin hollow line
hexagon
start
clear
begin hollow line
repeat
72
set color
random
min
0
max
10000
hexagon
right
5
set heading
180
save svg
finnlol
MacBook Air


start
MacBook Air



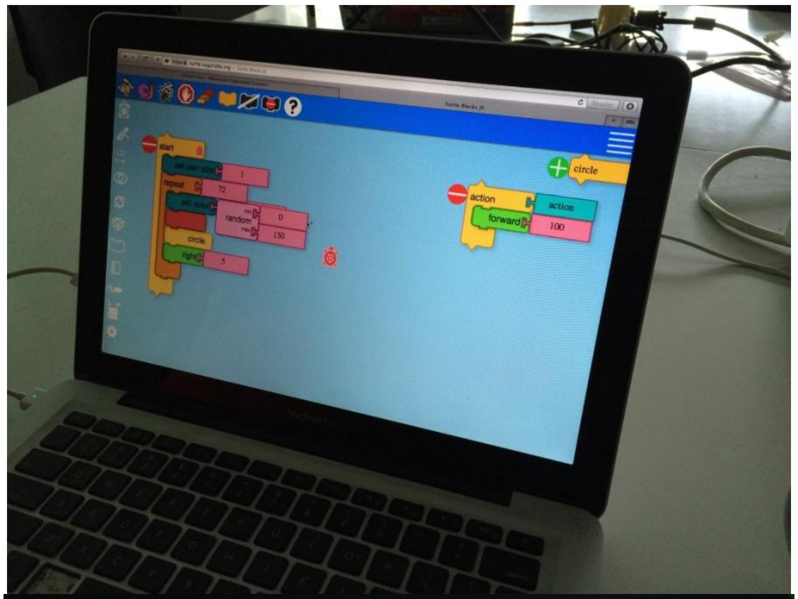
Figures 3–6: Laser Etching workshop

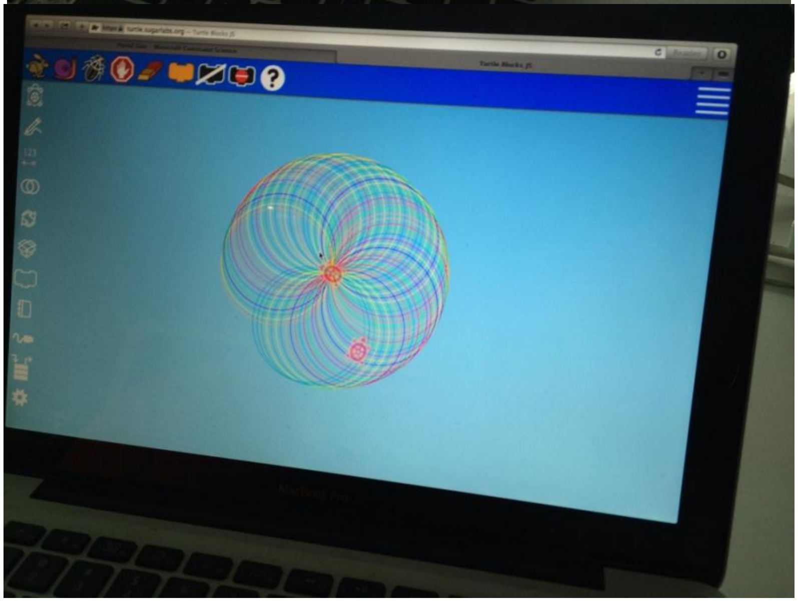
## III. Analyzing the Students' Work

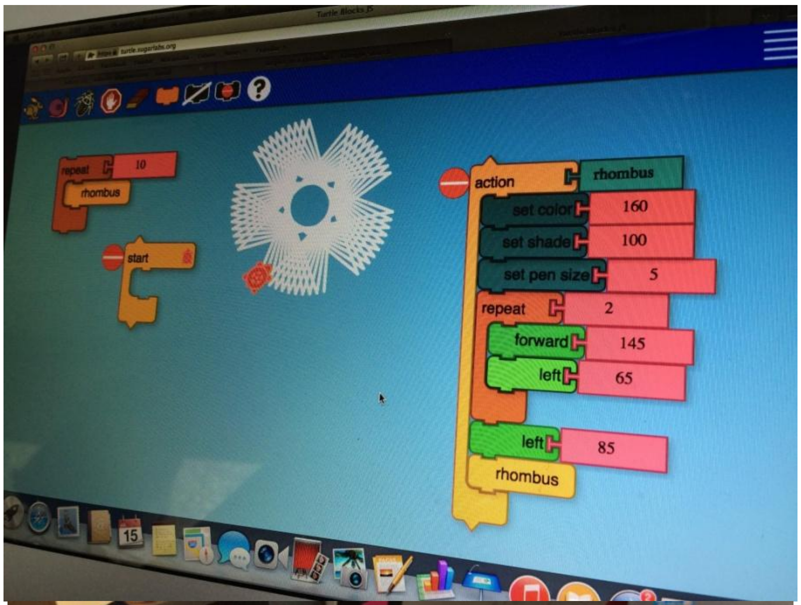
Turtle Blocks JS encourages an iterative approach to design and refining one's procedures and programming. When Turtle Blocks JS first loads, there is simply a "Start" block on the workplane. In order to start, one has to begin experimenting with what the blocks do, how they can be combined, and what kinds of designs are produced when blocks are connected. Blocks can be so easily added to an existing stack, and existing stacks can be so easily disassembled and reassembled in different sequences. Programmers can easily see the difference a change in the sequence of blocks might produce by how the design is affected by block sequence, flow, pen color, or other combinations of blocks. For people new to Turtle Blocks JS, the first designs created often emerge from serendipitous combinations of blocks that are arrived at through iteration on an initial sequence. When a combination of blocks does not produce the design the user has in her or his head, it is easy to tinker with the blocks until the desired design, or perhaps a different design, emerges. In fact, the first designs that many young programmers create emerge without an understanding of exactly what moves the turtle is making, or the particular sequence of moves and how that affects the design. Rather, the non-scaffolded approach where the user is permitted to create knowledge of how the turtle works provides the context for wanting to learn what the blocks do. After creating an initial design through chance, the user typically seeks to understand how a particular sequence of moves created the design so she or he can perhaps replicate the design on somebody else's computer. This process of consciously thinking about the sequence of blocks and moves allows the user to begin constructing new designs with a better understanding of how particular turtle moves can be combined. The realization that repeating a design while moving on the Cartesian plane between each repetition of the design makes the concept of the Cartesian plane concrete to young users. While initial designs programmed in Turtle Blocks JS might emerge entirely by a chance sequence of blocks, the ease with which blocks are manipulated and resequenced encourages design through iteration, repetition, and movement on the Cartesian plane. These are very important concepts in geometry, taught in this case through the student constructing knowledge of geometry from the workings of the turtle’s movements.

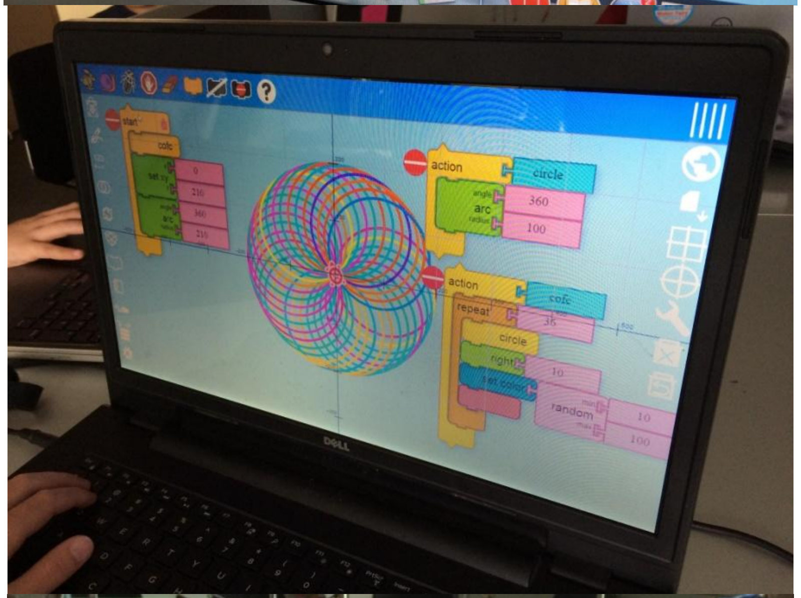
Properly taught, Turtle Blocks JS encourages an algorithmic approach to programming. Brian Silverman (Constructing Modern Knowledge Faculty, 2015), in a conversation, characterized algorithmic thinking as “learning how to express a sequence of things that express over time.” If students are prompted to explore Turtle Blocks JS as an opportunity to create a sequence of turtle moves that change over time, the concept of creating a pattern that changes as the program runs is made concrete to the programmers.

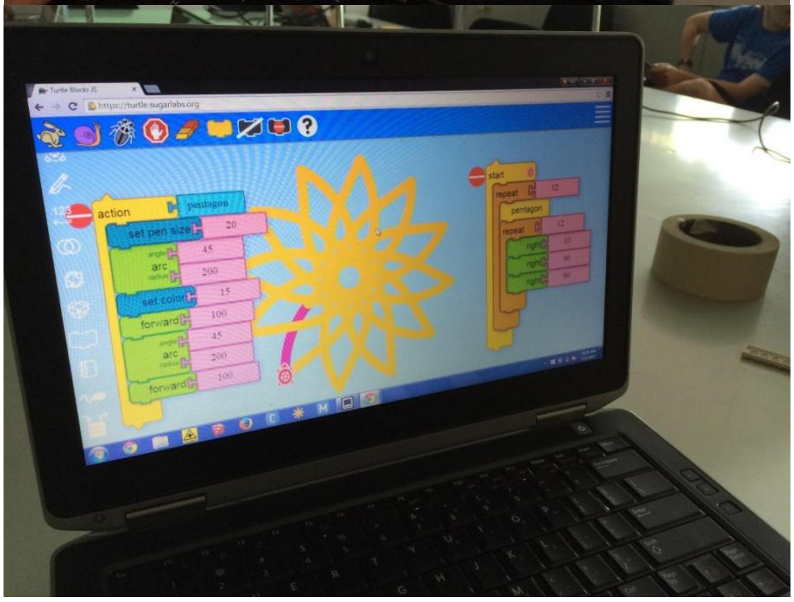
Students at the library workshop were quickly able to create procedures after prompted to “learn how to express a sequence of things that express over time.” Additionally, the concept of a design created by rotating a simple shape came easily to these students despite all of them being inexperienced with Turtle Blocks.






repeat 10
rhombus
start
action rhombus
set color 160
set shade 100
set pen size 5
repeat 2
forward 145
left 65
left 85
rhombus


start
cofc
action circle
arc 360 100
action cofc
repeat 36
circle
right 10
set color
random 10 100


action pentagon
set pen size 20
arc 45 200
set color 15
forward 100
arc 45 200
forward 100
start
repeat 12
pentagon
repeat 12

Figures 7–11: Student output from the Library workshop

The students were instructed to add a few blocks that were created for 3D printing the designs. First, they added a No Background block to the top of their main procedure. Generally, a pen size of 15 is also necessary to account for resizing the design for 3D printing and to make sure the line is wide enough for the 3D printer to produce. In order to work around Tinkercad's interpretation of an imported SVG file, a Begin Hollow Line block is added before the turtle begins drawing. Likewise, toward the end of the main procedure a End Hollow Line block is added. Finally, a Save SVG block, with a title added for clarity, is added to the end of the main procedure to save the design.

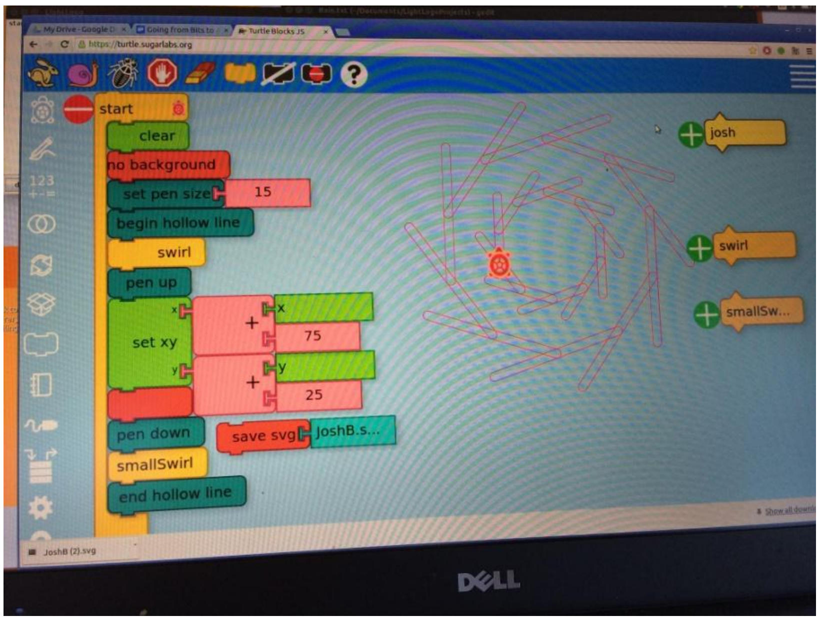


Figure 12: Using the Hollow Line blocks

Next, students were instructed how to import the SVG file into Tinkercad (https://tinkercad.com). Typically, importing the design at 25% and 10mm tall produces a design that is workably sized. The design can be further resized depending on the desired size of the 3D print. A 1mm tall base is added below the design, which is sized to be 1.75mm tall. This way, when used, there is not enough depth for the modeling compound to get stuck in the design but there is enough relief to the design that a good impression is possible. Sized at approximately 5cm x 5cm x 1.75mm, these tiles 3D printed in about eight minutes.

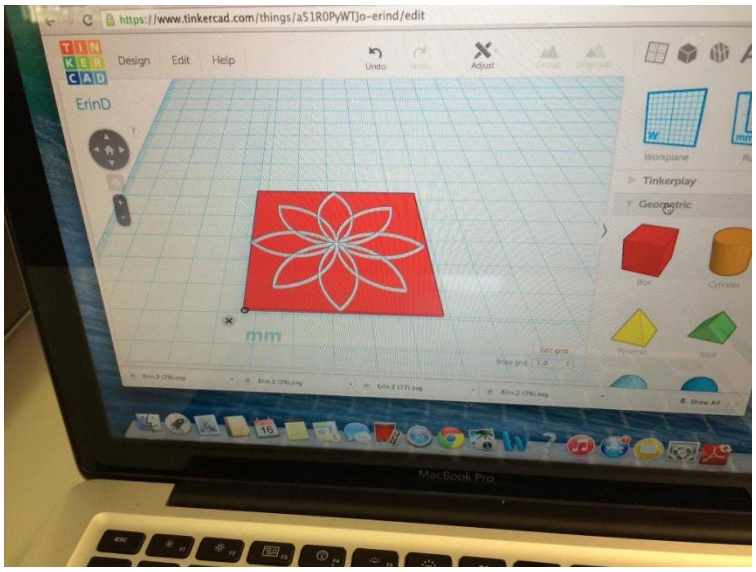
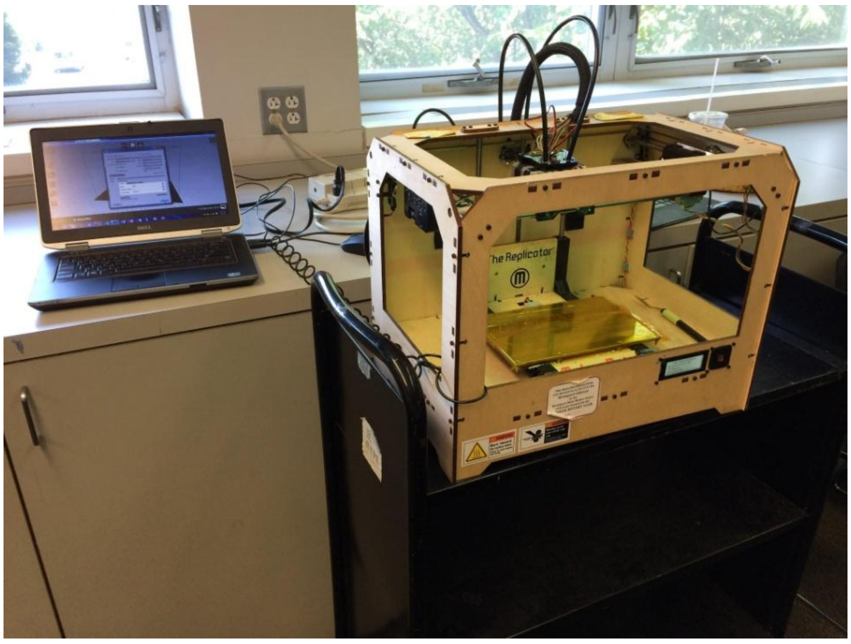
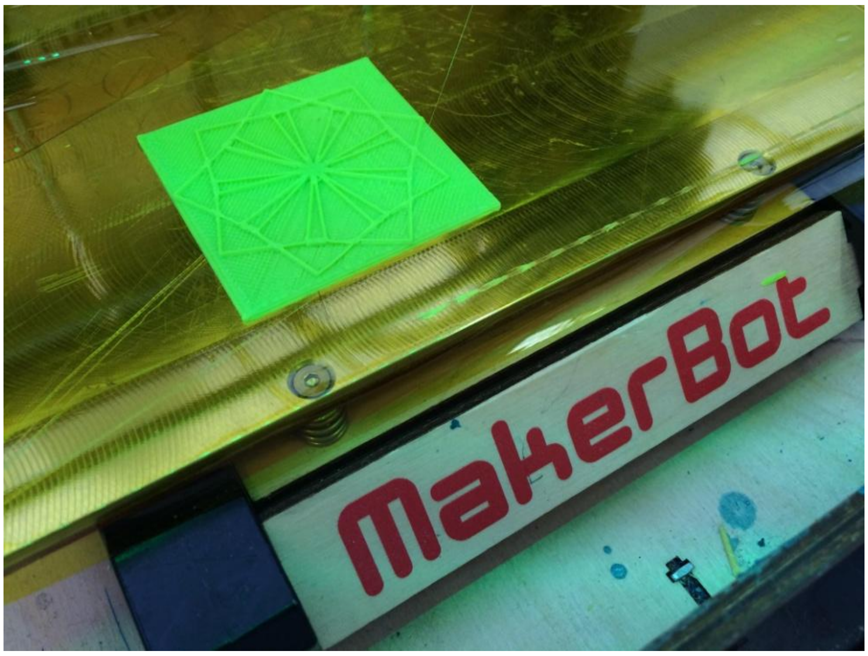
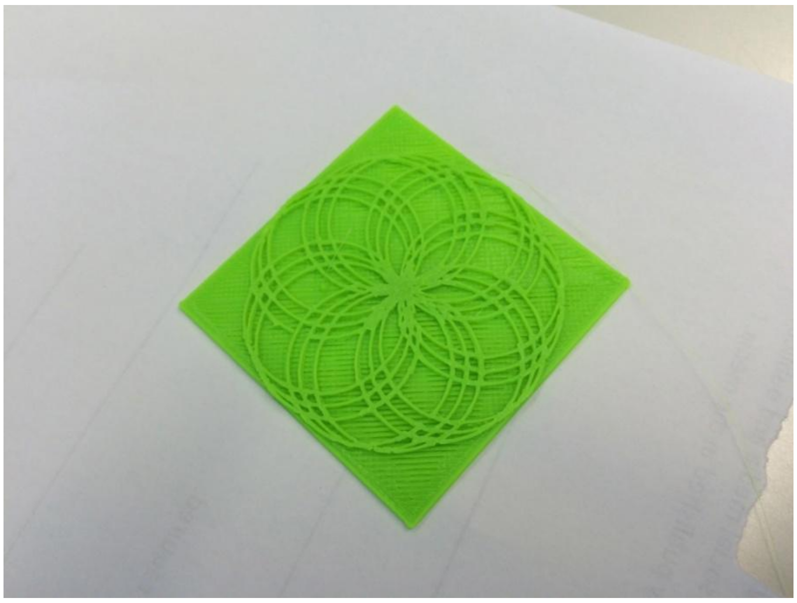


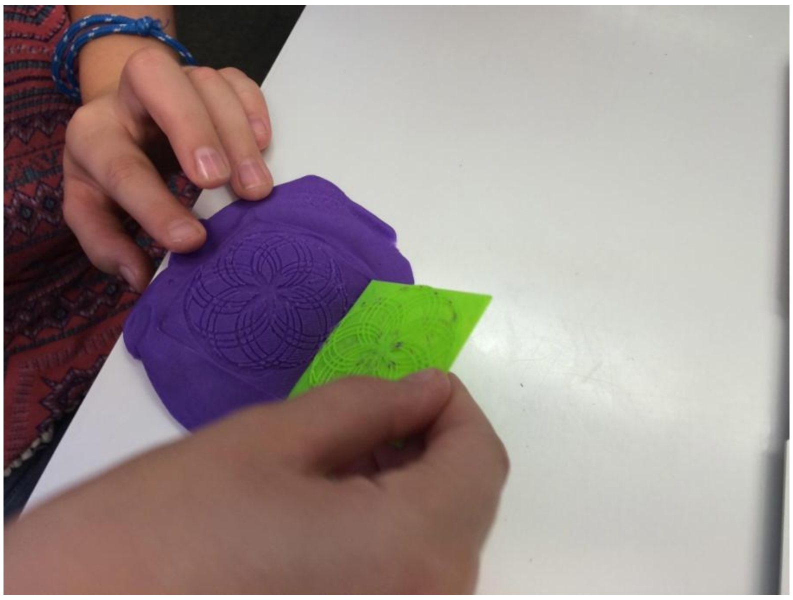
Figure 13: Importing a design into Tinkercad

Because some students had not yet been trained how to use the 3D printers and did not have the necessary waivers signed, I ran the 3D printer. The students used a shared Tinkercad account of which I was the owner, and I downloaded their models to a laptop connected to the 3D printer. Additionally, I managed the software for the 3D printer as well because of similar liability issues.




MakerBot





Figures 14–19: Creating a 3D-printed tile and stamping it into PlayDoh

By the end of a two hour session, of the eight students working in Turtle Blocks JS, three completed a design that was 3D printed and two were still actively working on a design that would be printed the following day.

Two of the girls who were still working on their tile designs put aside that project and started a new one. They wanted to create a procedure that would write a word. I suggested that they program each letter as a separate procedure.

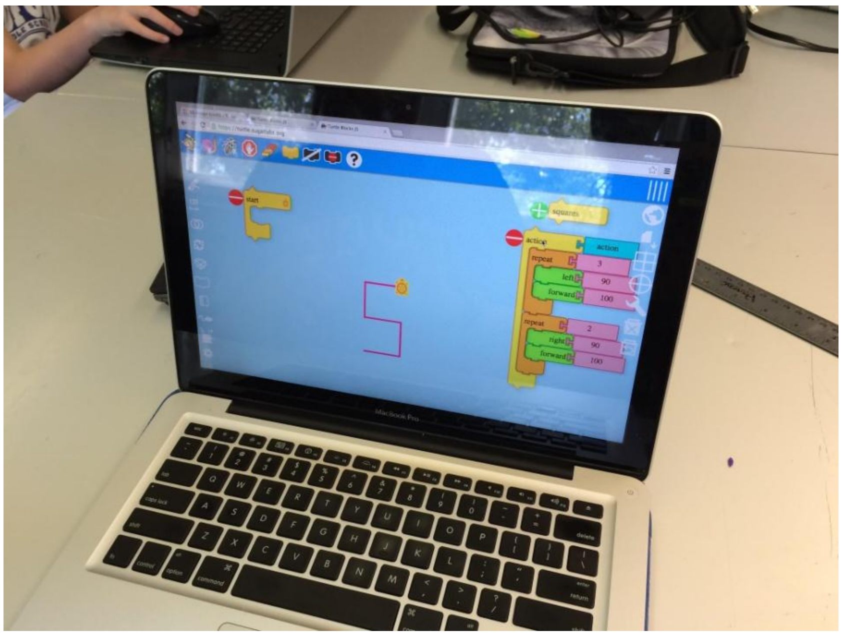


Figure 20: Programming the letter S in Turtle Blocks

One girl changed the assignment to produce a pendant and earrings. Unfortunately, the size of the line was quite small and some parts were fragile. However, the models printed and the earrings were light enough to comfortably wear.

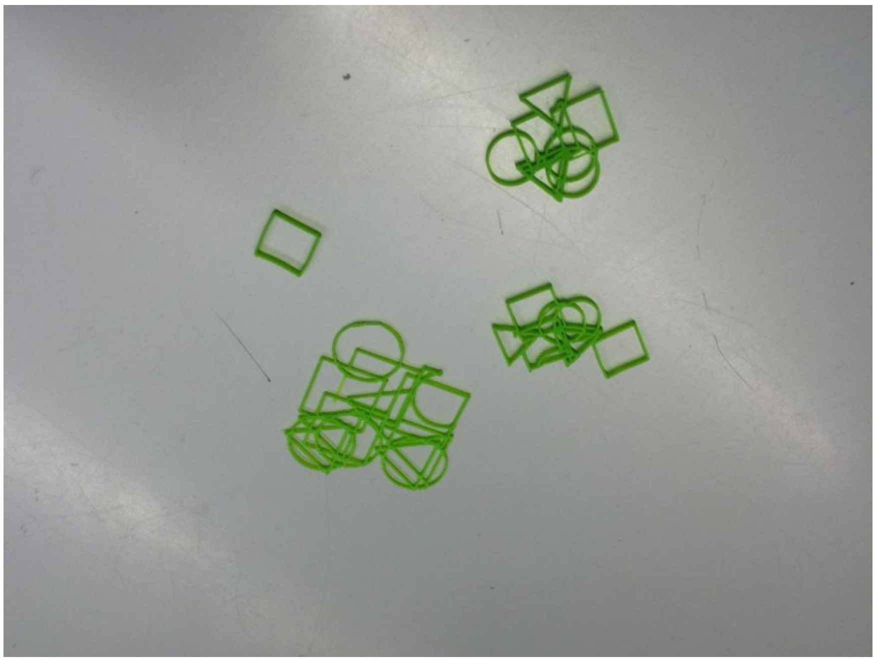
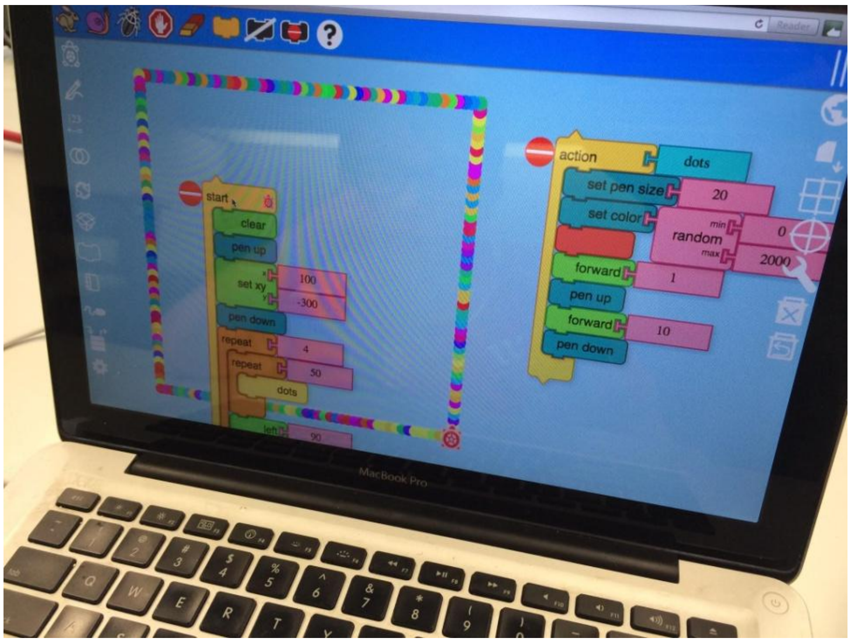
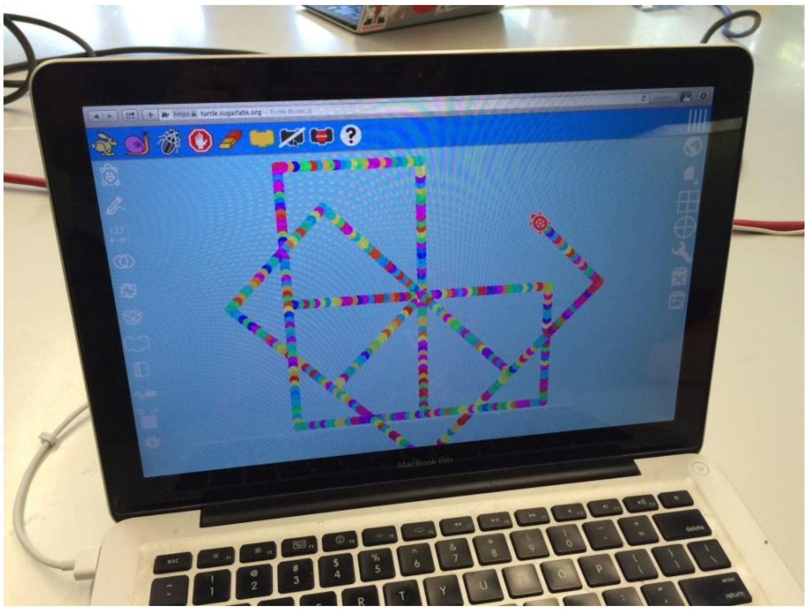


Figure 21: Designing earrings in Turtle Blocks

Another student pursued other programming challenges in Turtle Blocks JS but did not care to produce a tile. His procedures made use of the ability to change the pen color, something that would not be realized in a 3D printed version of the design. He also explored the concept of creating a design from a pattern that repeats over time.


```
start
  clear
  pen up
  set xy 100
         -300
  pen down
  repeat 4
    repeat 50
      dots
    left 90

action dots
  set pen size 20
  set color random min 0
                   max 2000
  forward 1
  pen up
  forward 10
  pen down
```


Figures 22 and 23: Changing the pen colors

One student did not complete the project. He did not pay attention during the brief demonstration. He resisted all of my attempts to assist him. He did spend time programming in Turtle Blocks and created a repeating pattern.

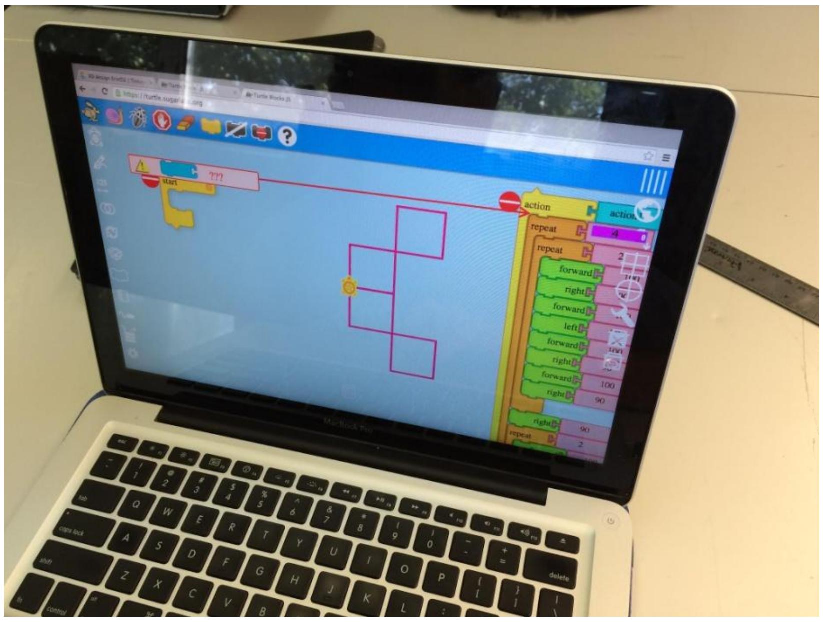
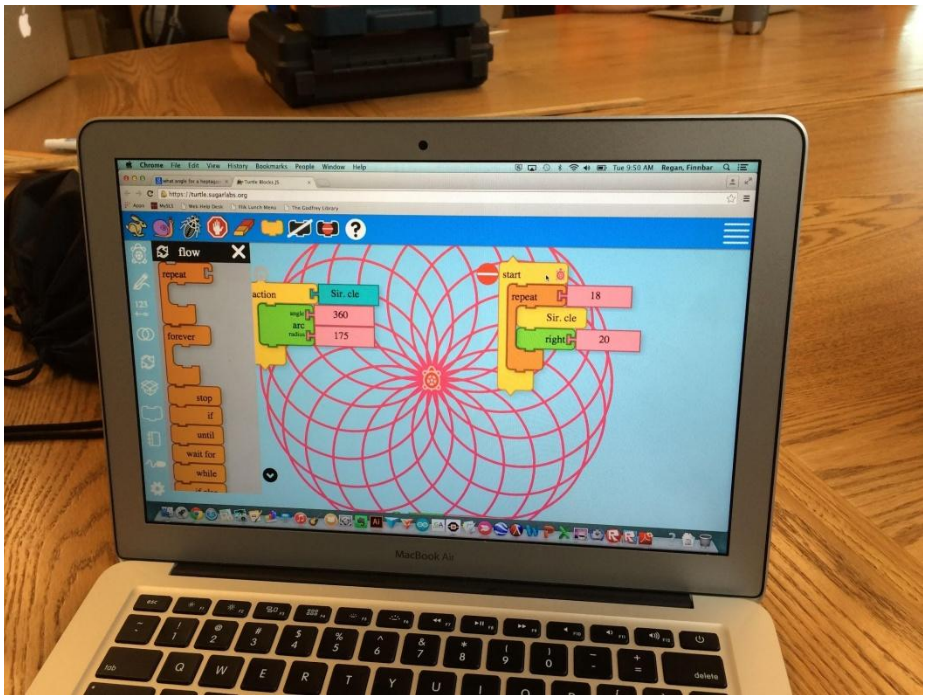
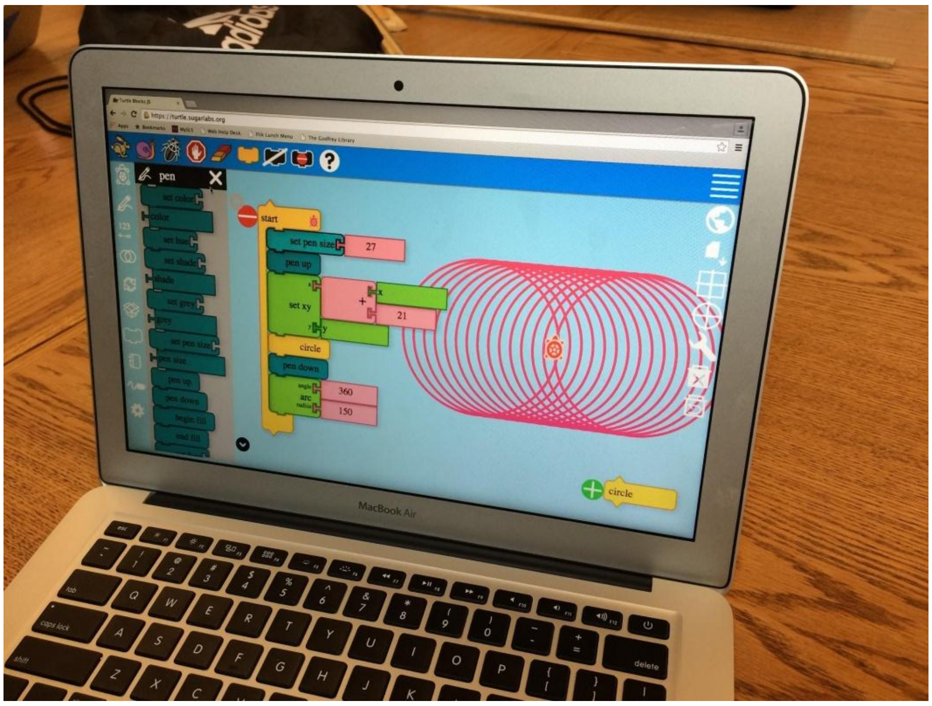
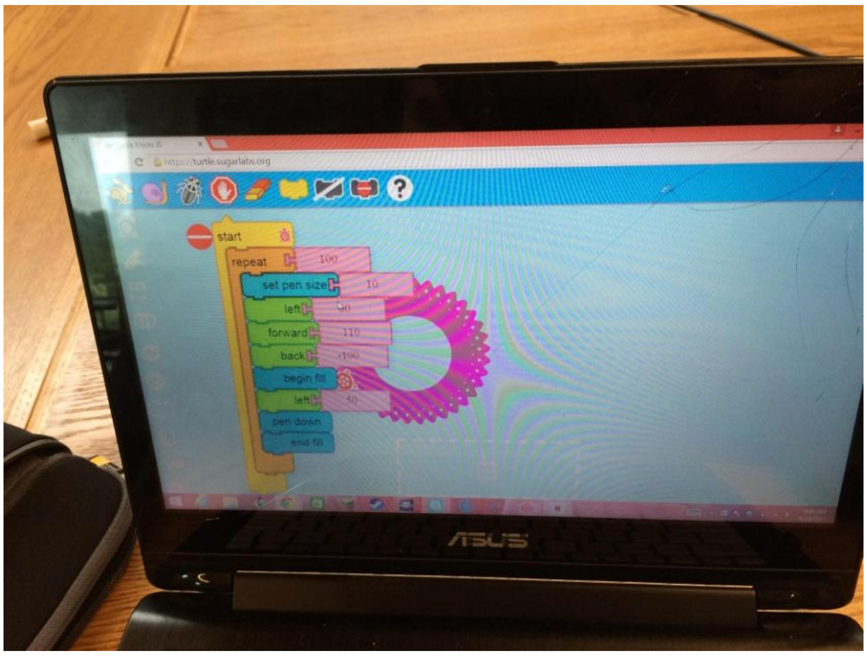


Figure 24: Repeating patterns

The fourteen students at the technology camp who were given a short tutorial on using Turtle Blocks but not provided a prompt took more time to develop designs that could be etched into rubber. Some designs were too large or too complex to be accurately etched by the laser cutter.


flow
repeat
forever
stop
if
until
wait for
while
action
Sir. cle
angle
arc
radius
360
175
start
repeat
18
Sir. cle
right
20


pen
set color
set hue
set shade
set grey
set pen size
pen up
pen down
begin fill
end fill
start
set pen size
27
pen up
set xy
x
y
21
circle
pen down
360
150
circle



Figures 25–27: Designs by the students attending a technology camp

Additionally, students not provided a prompt needed more one to one assistance from the head teacher and myself to clarify concepts or to develop a design.

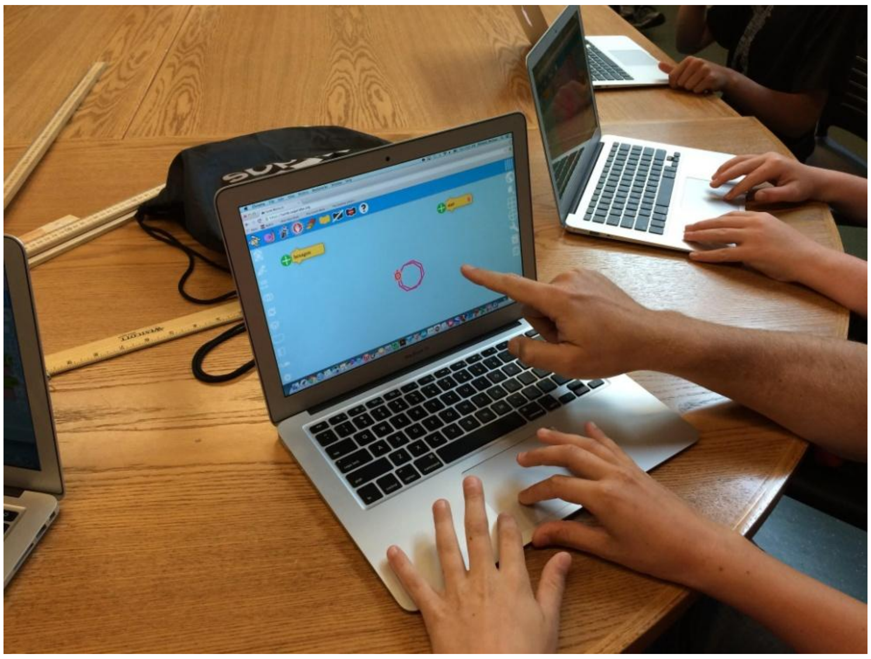


Figure 28: Helping hands

By the end of the day, six of the fourteen students had designs that were etched onto rubber with an Epilog 200w laser cutter.

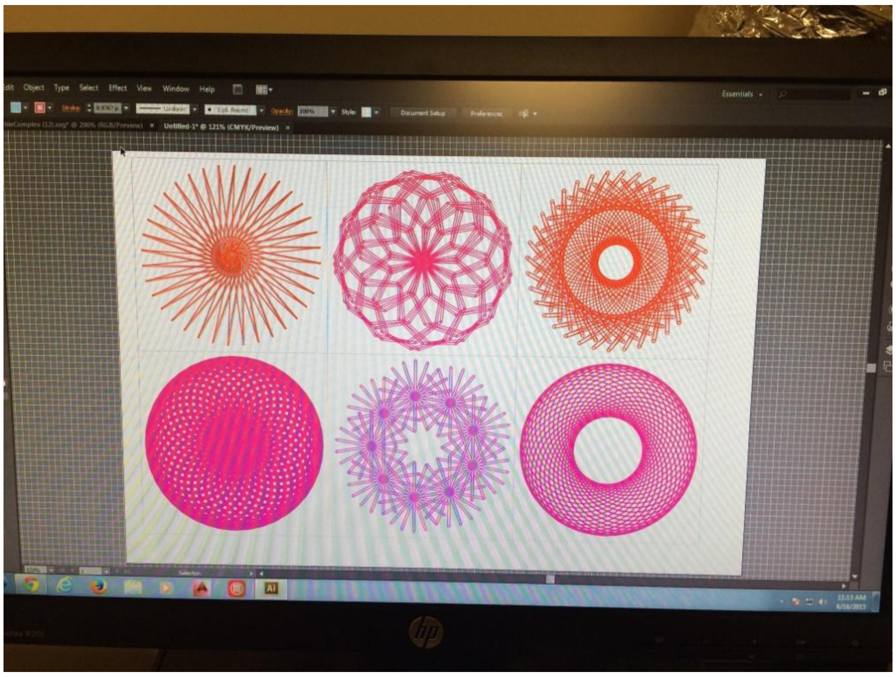


Figure 29: Designs etched onto rubber stamps

The students followed the same Turtle Blocks JS procedure of creating a design, suppressing the background and saving an SVG file, but they did not need to use the hollow lines in their procedures. The completed designs were resized by the head teacher in Adobe Illustrator to fit six designs on a sheet of 8inch x 10inch rubber. The designs took about half an hour to etch. The sheet of rubber was cut with an X-Acto knife to separate the individual stamps. Students, on their own, designed handles and bases in Tinkercad to be 3D printed. The rubber stamps were glued to the 3D printed handles with contact cement.

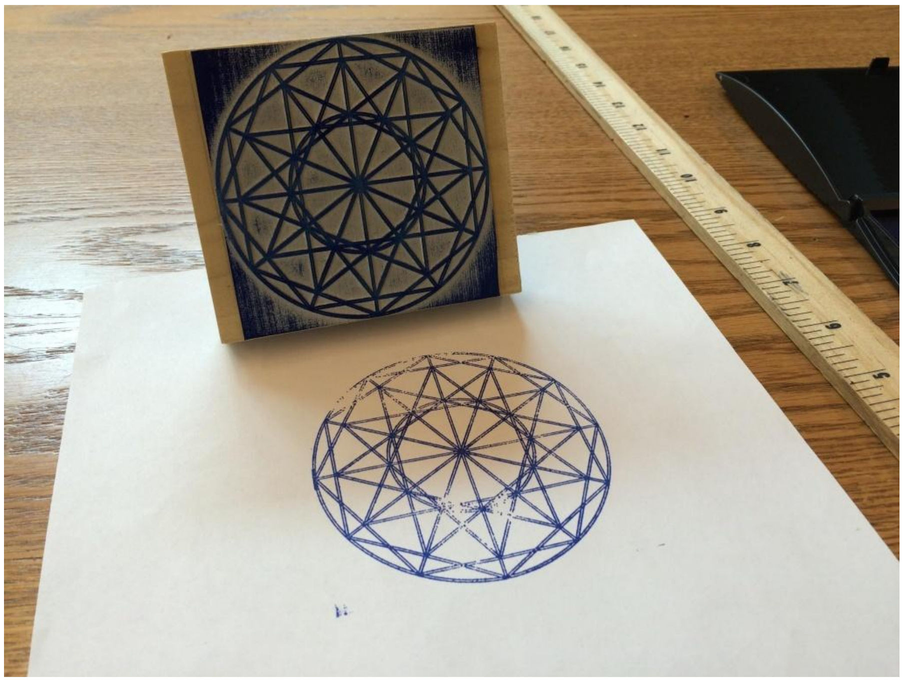


Figure 30: A rubber stamp with a 3D-printed handle designed by Josh

Teaching Turtle Blocks JS also also provides an opportunity to teach programmers about building procedures. Dr. Cynthia Solomon, at Constructing Modern Knowledge 2013 (Constructing Modern Knowledge Faculty, 2015), explained that successful Logo programs are built on procedures, which she characterized as small tools. These small tools can be combined, she explained, to create larger, more complex designs. Rather than attempting to program a design in a monolithic block of code, students can learn to create smaller procedures that can be combined to accomplish what a large, unwieldy collection of blocks can also draw. The advantage to writing procedural programs is the flexibility procedures offer, such as changing the line width or color during the course of the main procedure, since the small procedures are composed of moves, not customization features. Additionally, writing procedurally allows for easier debugging, as individual procedures can be isolated if the master procedure does not run as expected.

Learning to debug a Turtle Blocks JS procedure involves learning what individual blocks cause the turtle to do, how the sequence of blocks matters greatly, and learning to situate oneself in the turtle's "shoes." Learning what the blocks cause the turtle to do involves the student changing the number associated with the block. For example, by default the Forward block is connected to the number 100. When a student clicks the block to run it for the first time, 100 is established as a unit in Turtle Blocks. It may not correspond directly to any measurement with which the student is presently aware, but it establishes a context for what measurement or distance means in this microworld. Additionally, a student might not be aware that a circle is 360 degrees, but through experimenting with the number connected to the Left or Right block, the student creates knowledge of degrees, though she or he might not call the unit a degree. Furthermore, through continued exploration the student might discover that turning at least 360 of these units causes the turtle to make a complete rotation. Combining the design the student created through earlier experimentation and creation of procedures, she or he might try rotating the design to create a more complex design. Experimentation of the blocks and the numbers associated with the blocks allows students to naturally debug a procedure.

Even though the user might know what each block in her or his procedure means and does, when a design still is not drawn as the user intended she or he soon discovers that the sequence of the blocks in the procedure matter greatly. Most often the serendipitous discovery of a design is the result of dragging blocks to the workplane, clicking on them to run them, and adding additional blocks to the workplane and repeating the process. Eventually, a design can emerge from such experimentation. When the design is created, the student will try to recreate the exact sequence of blocks in a collection, or procedure. However, since the initial design oftentimes emerged from haphazard clicking of blocks, the student is challenged with the necessity of combining the blocks in a definite sequence in order to recreate the design as a procedure in a tidy collection of assembled blocks. Debugging involves learning to consciously combine the blocks in a sequence that produces the moves the turtle needs to accomplish in the order needed to properly draw the design.

When the turtle is not producing the design expected, suggesting that the student get out of her or his chair and put her- or himself in the turtle's shoes is an effective debugging technique. With a friend reading the movements and the programmer making the turtle's movements, the problem in sequence, choice of blocks, or flow quickly becomes apparent. Students are also unaccustomed to the invitation to get out of their seats and walk about the classroom, not aimlessly but with purpose. This is a memorable exercise that they can use to solve future Turtle Blocks JS programming problems.

## IV. Conclusions

This project, because of Turtle Blocks JS' vector art SVG format capabilities, can be adapted to other forms of fabrication, too. Its extensibility is a response to Dr. Gary Stager's prompt, "And then?", (http://stager.tv/blog/? p=3214) because the artifact is not the end product for static display, but rather a tool that allows the student to begin an exploration of an entirely different skill. In these examples, the student transitions from a programmer to a digital fabricator to a potter or a print artist. Multiple skills are developed and expanded. The tools even allow for exploration of different mediums: whether the student uses PlayDoh or clay to be fired in a kiln, for example. The workflow could easily transfer to using a vinyl or paper cutter to cut the designs, which could be used to decorate a card or to personalize one's laptop.

Using Turtle Blocks JS in conjunction with digital fabrication present students the opportunity to create tools that allows for them to produce increasingly complex work. The creation of a clay stamp with a complex Turtle Blocks JS design allows the student to create patterns that are too complex to create freehand. In this way, the tool helps students create work that would otherwise be impossible for them to produce. The complexity of the design might be beyond what the student could produce by hand, much as the math required to create the design requires the student to explore the math within the microworld, because otherwise the computations would be impossible for the child at her or his current stage of development. The computer helps the student explore complex ideas perhaps beyond the student's immediate comprehension, allowing the student to operate in Vygotsky's Zone of Proximal Development, where the work requires skills just slightly beyond their current capability. The turtle is the tool that helps the student explore this lofty mathematical work. Their explorations create the resulting beautiful designs.

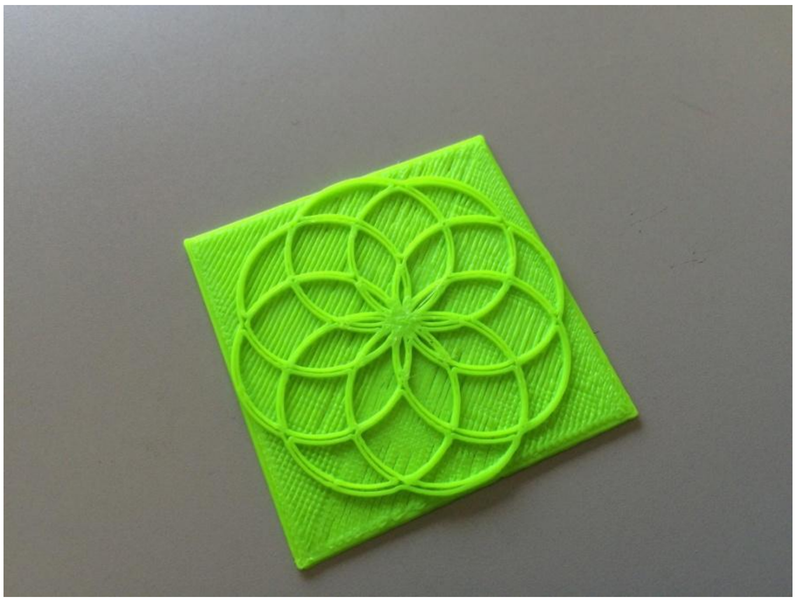


Figure 31: Student-produced artwork produced with Turtle Blocks

This Turtle Blocks JS project creates an open-ended tool that encourages further exploration of its use. Creating a simple procedure such as a single polygon and producing a clay stamp or rubber stamp from the design allows the user to explore the patterns that emerge when overlapping the design in the real world, as opposed to producing a stamp that already has a rotation of the design through 360 degrees. People can use simple polygons to produce art of their own beyond the vision of the original programmer. Likewise, different mediums, be it ink, paint, or even conductive foil, can be used with the stamps to create different works of art that are beautiful and functional. The tool can be used to create things that themselves are tools: tiles that function as a backsplash in the kitchen, decorations for the classroom, mathematical manipulatives, or even stickers to customize products that are otherwise generic in their appearance.

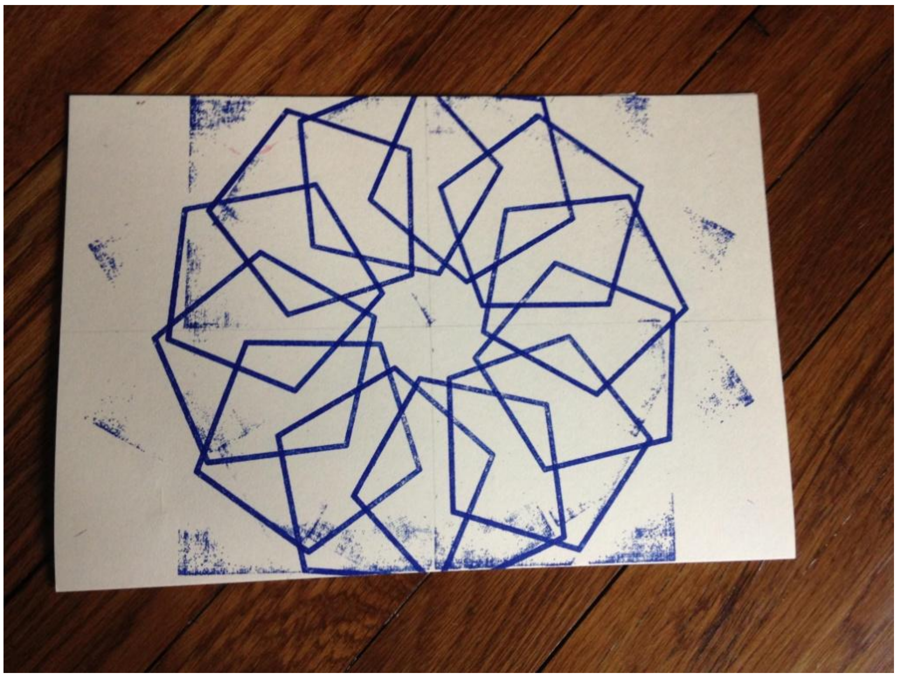


Figure 32: Applying designs generated in Turtle Blocks to real-world applications

Turtle Blocks JS' role as a microworld in which people may explore mathematics, geometry, and design through Logo programming is an important constructionist software environment. The designs produced in Turtle Blocks JS can be transformed through personal fabrication such as 3D printing or laser cutting and etching into tangible tools that allow for further exploration of mathematics, geometry, and design in the real world. An easy workflow coupled with increased access to personal fabrication tools should encourage educators and students to transform their designs produced through programming from bits to atoms.

Presentación del manuscrito: 10 de septiembre de 2015

Fecha de aprobación: 10 de septiembre de 2015

Fecha de publicación: 15 de septiembre de 2015

Burker, J. (2015). De Bits a Átomos: Programación en Bloques de la Tortuga JS y Fabricación Personal en Proyectos de de Jóvenes. *RED. Revista de Educación a Distancia. Número 46*. 15 de Septiembre de 2015. Consultado el (dd/mm/aa) en http://www.um.es/ead/red/46

## Referencias

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

Turtle Blocks Documentation (2015, July 27). Retrieved from https://github.com/walterbender/Turtle Blocksjs/tree/master/documentation

Turtle Blocks Guide (2015, August 12). Retrieved from https://github.com/walterbender/Turtle Blocksjs/tree/master/guide

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

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