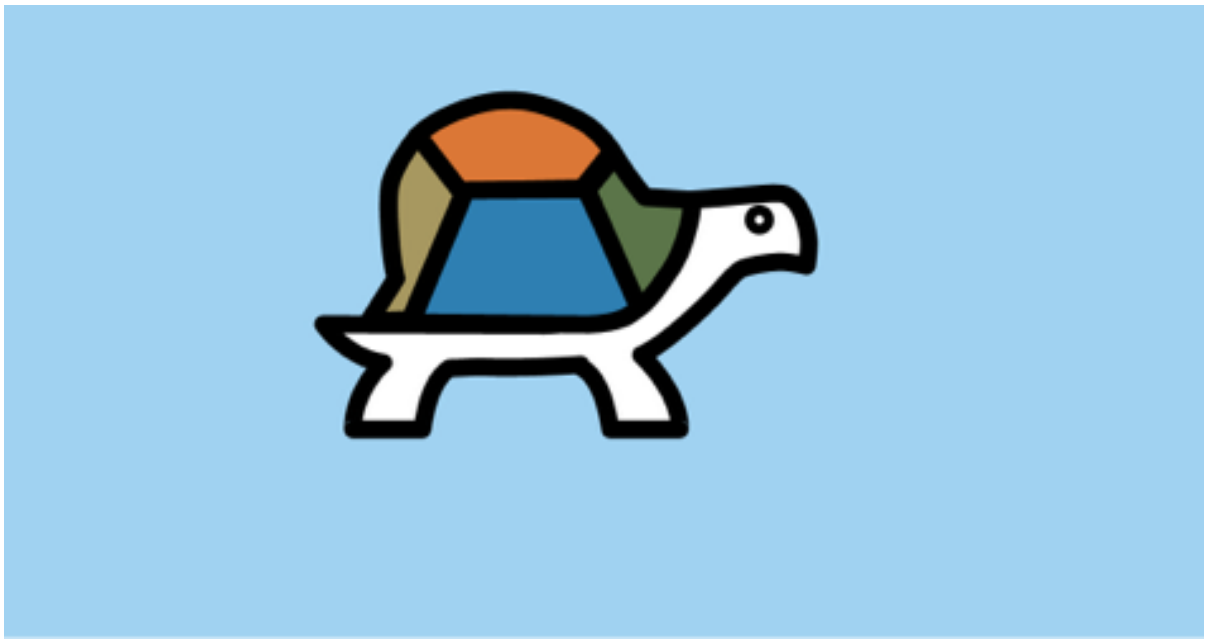


sugarlabs

Table of Contents

All About Turtle Blocks

What is Turtle Blocks?

Turtle Blocks is an activity with a Logo-inspired graphical "turtle" that draws colorful art based on snap-together visual programming elements. Its "low floor" provides an easy entry point for beginners. It also has "high ceiling" programming features, which will challenge the more adventurous student.

Where can we get Turtle Blocks?

You should put the link to the JS version here and describe that you just need to open the link in a web browser.
https://turtle.sugarlabs.org/ Turtle Blocks can be easily run on a browser through the javascript version.

Note

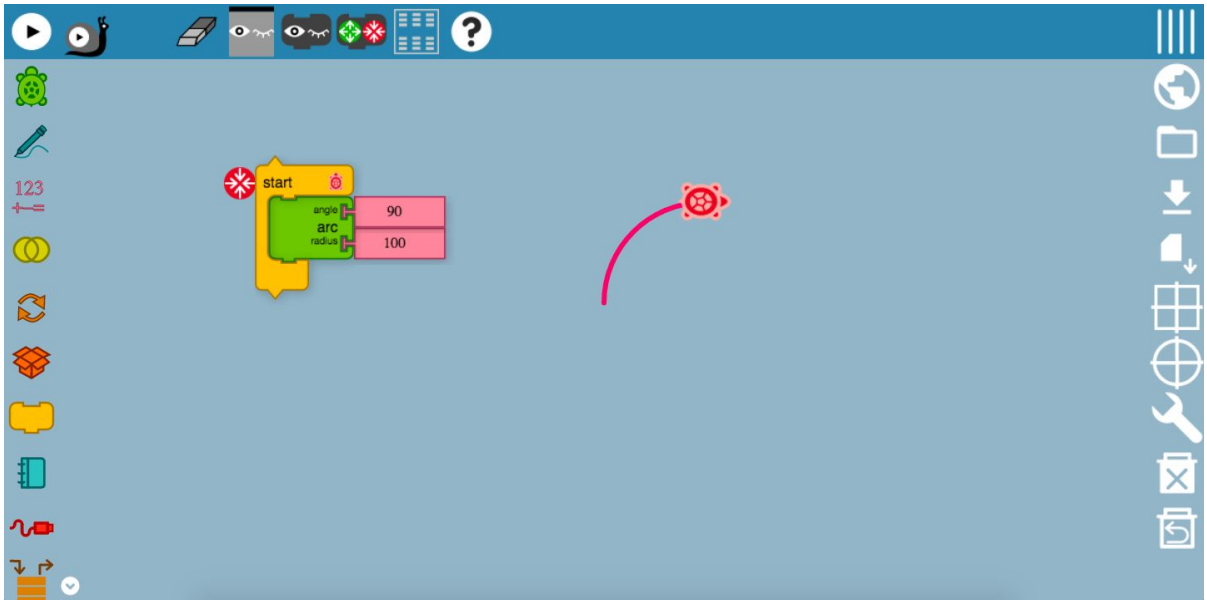
This manual is based on a JS version and there is a Python version that runs on GNU/Linux platforms and native to Sugar. It can be found at:

http://activities.sugarlabs.org/en-US/sugar/addon/4027

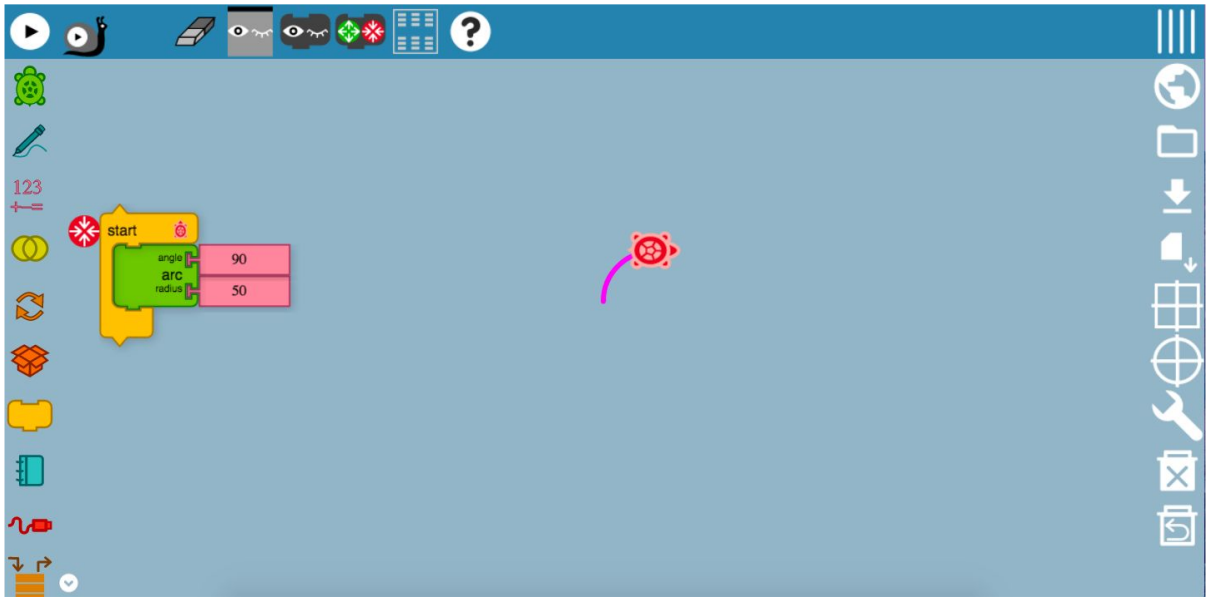
This is an advanced manual for Turtle Blocks, where users are assumed to have understood the basic program structure of Turtle Blocks, such as the basic mechanics of selecting blocks and building simple programs as well as the functionality of basic blocks such as Forward, Right, Set Color, etc.

Turtle Palette

Angle-Arc-Radius Block

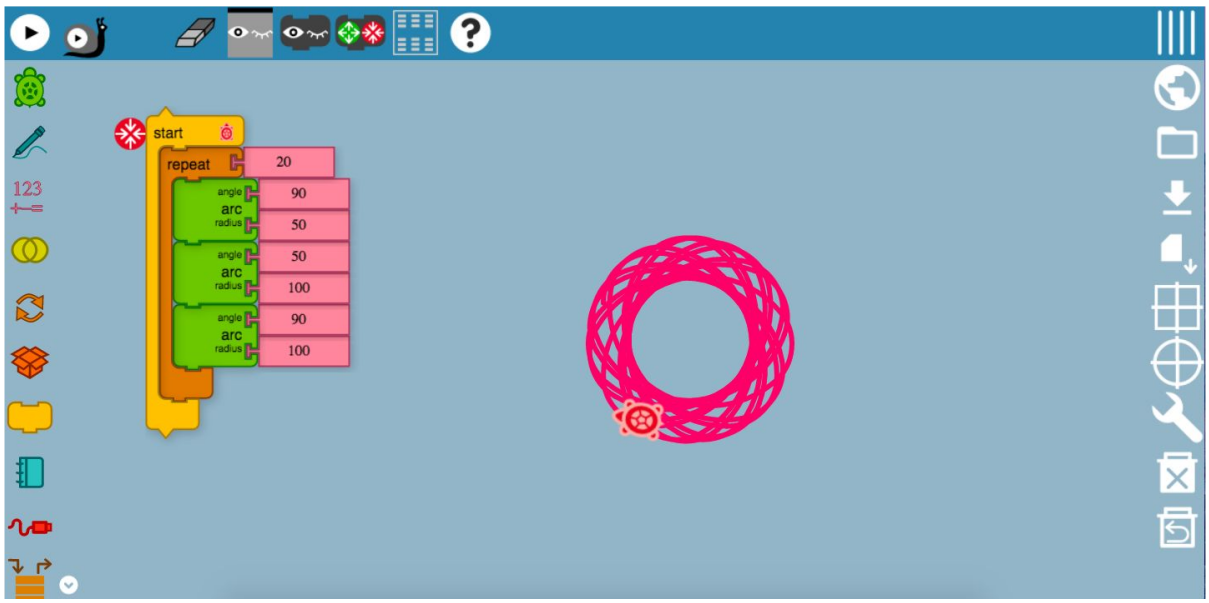


The Arc Block moves the Turtle along the path of an arc with a specified radius and across a specified angle. Have you noticed the 90 degree turning point? This is a result of the angle of the block being set to 90 degrees. By setting the angle, we have created the arc of a quarter circle! In this case, we have set the arc radius to be 100 pixels. The 100 pixels specify the radius of the circle upon which the arc is drawn.



As you can see in the example above, a smaller radius will result in a smaller circle and hence a tighter turn!

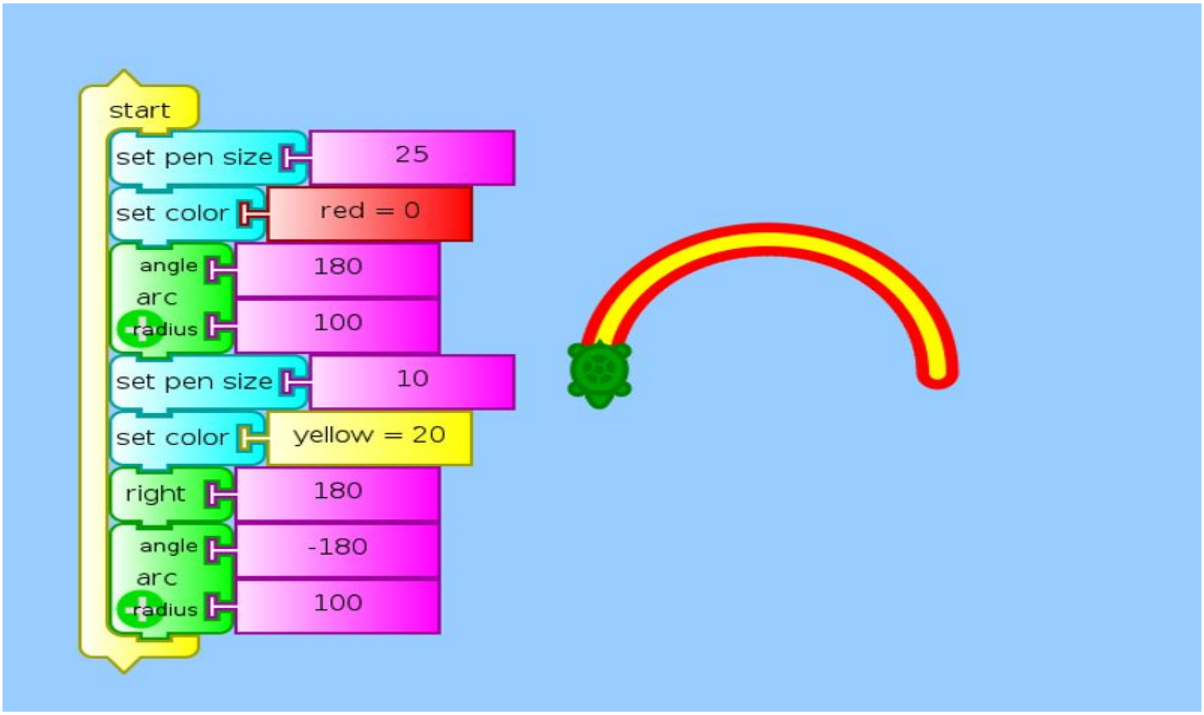
Of course, we could change the angle and arc radius to different values to experiment with different designs!



This example showcases a little weaving and knitting!

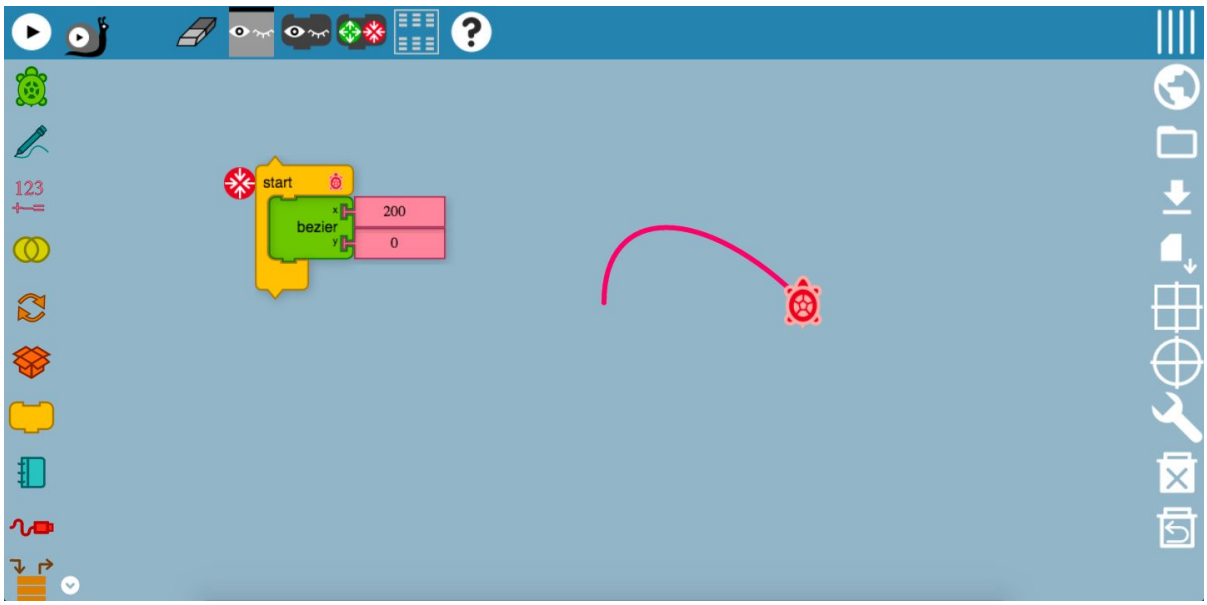
The Arc Block would be useful for visualizing geometric shapes and learning the concept of angles and radius.

Below is an example of using the Angle-Arc-Radius Block to create a little rainbow! (Taken from the Turtle Art wiki page.)

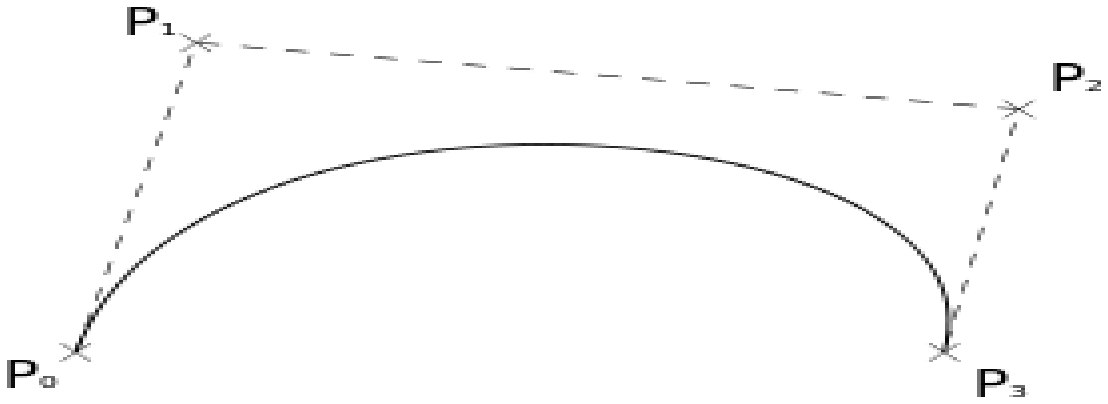


The first Angle-Arc-Radius Block creates the red outline of the semicircle, and brings the Turtle to the bottom of the right arc. After which, the right turn brings the Turtle to be facing the left direction. With a smaller pen size, the yellow arc outline lies in the middle of the thick red outline.

Bezier Block



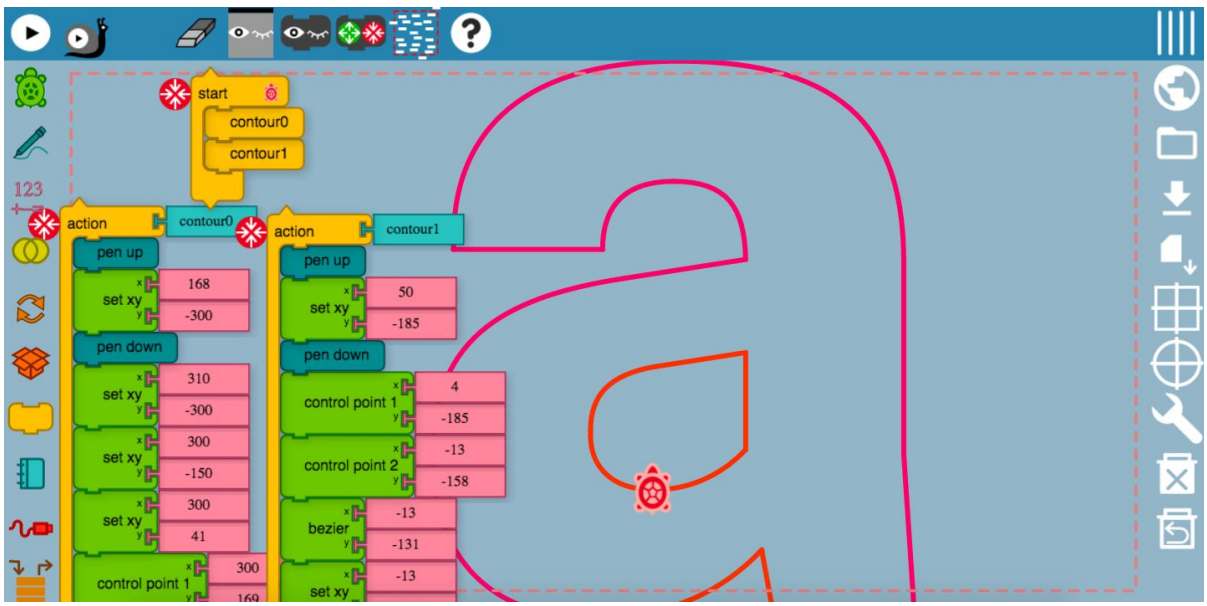
If you are unfamiliar with parametric curves, the Bezier Block may be slightly confusing. What is a Bezier? Bezier curve is discovered by the French engineer Pierre Bézier. These curves can be generated under the control of other points. Approximate tangents by using control points are used to generate curves.



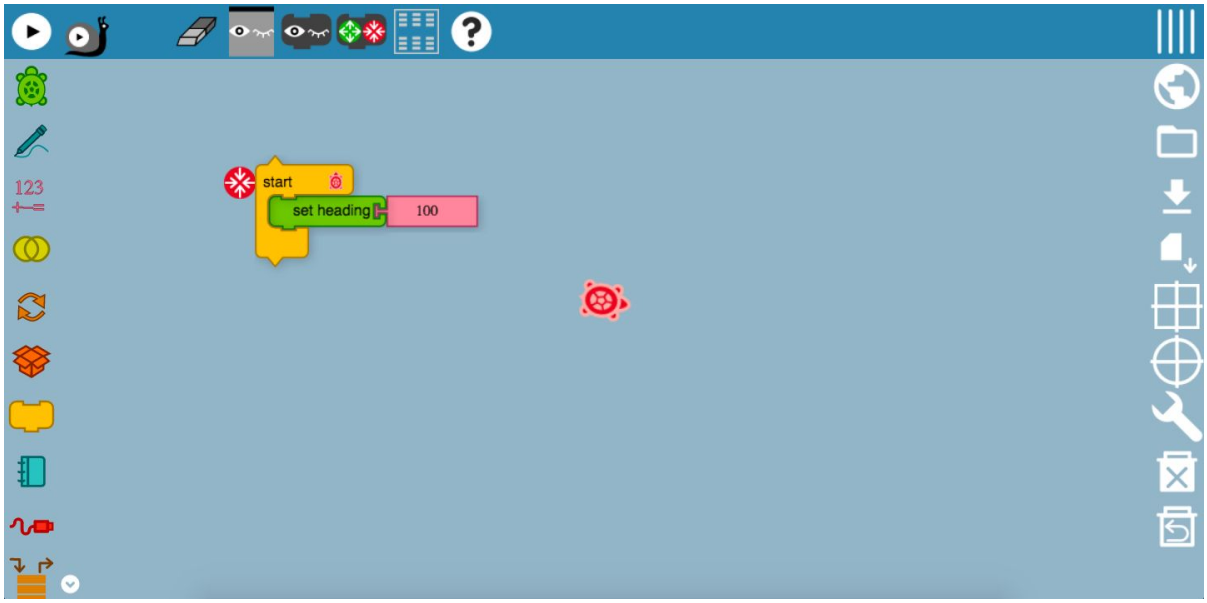
The Bezier Block causes the Turtle to move along a varying path depending on the position of the control points. The relative positions of the Control Points result in steeper and smaller curves or wider, more gentle curves. In this figure there are four points: P0, P1, P2, and P3. P0 is mapped to the current position of the Turtle. P1 is Control Point 1. P2 is Control Point 2. P3 is the point specified in the Bezier block.

The x and y coordinates of these points determine where they are in relating to each other. For example, you could swap P0 and P3. What would happen? Try it yourself!

The image below shows how we can use Bezier curves to make alphabet letters:

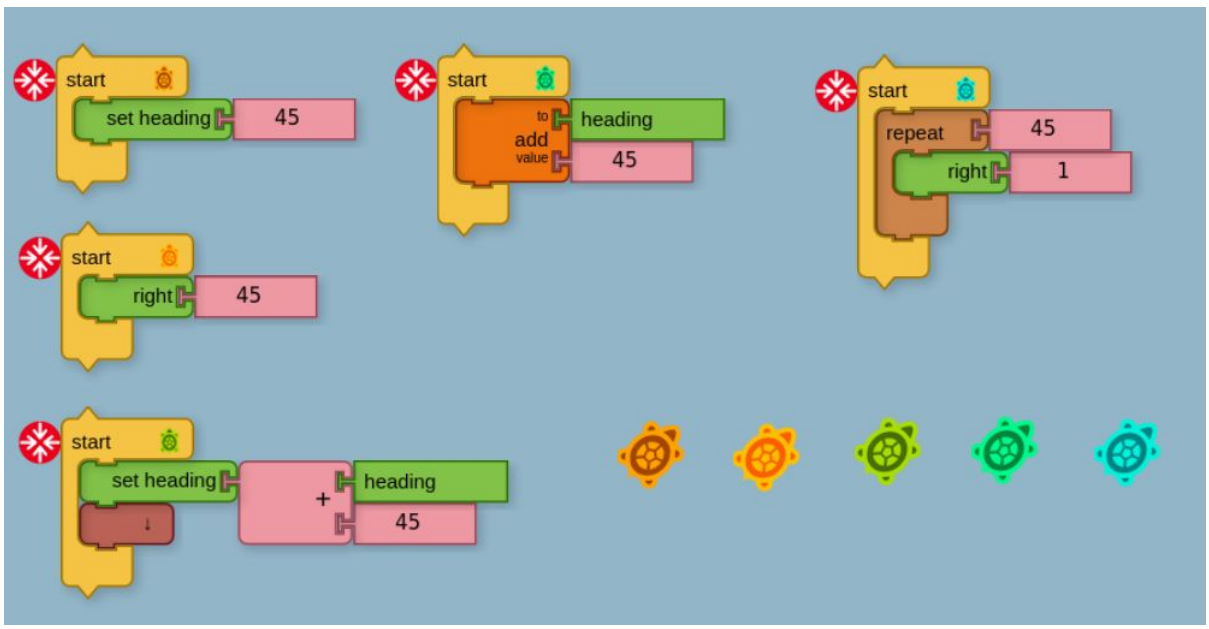


Set Heading Block



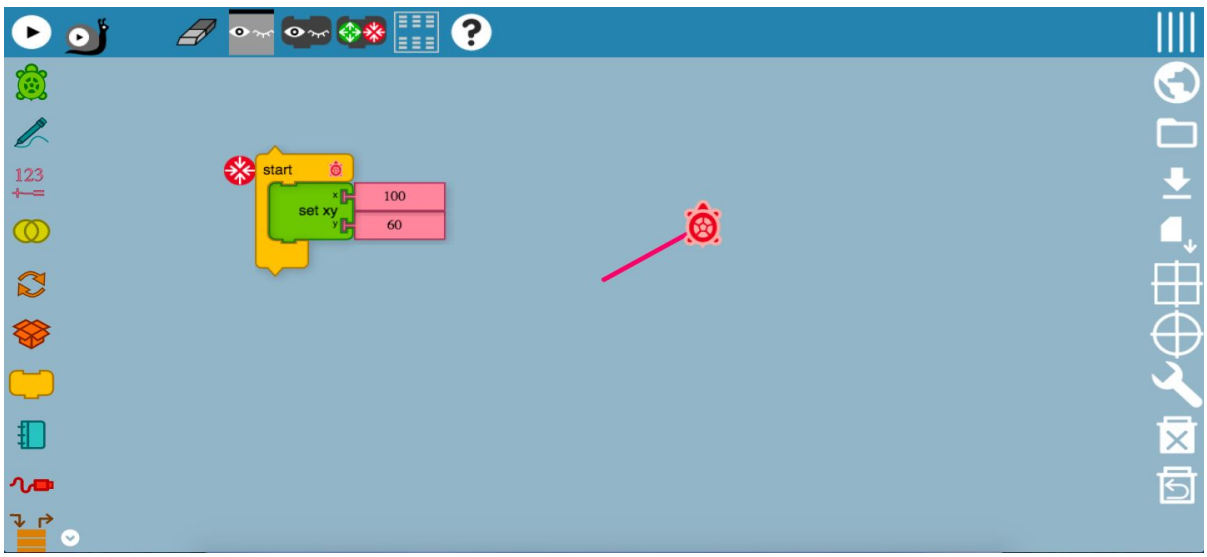
The Set-Heading Block is similar to the turn Left and turn Right blocks, but it is used to set an absolute heading rather than a heading that is relative to the current heading. As the name suggests, it specifies the direction for the Turtle to head, based on a specified angle. It is assumed that the north direction is vertically upwards on the screen and that the angle increases clockwise. In the figure, the Turtle is in between the east and south direction, but has not yet reached south-east. Try it.

Or you could use the add to block: add to heading some value. Try that too.



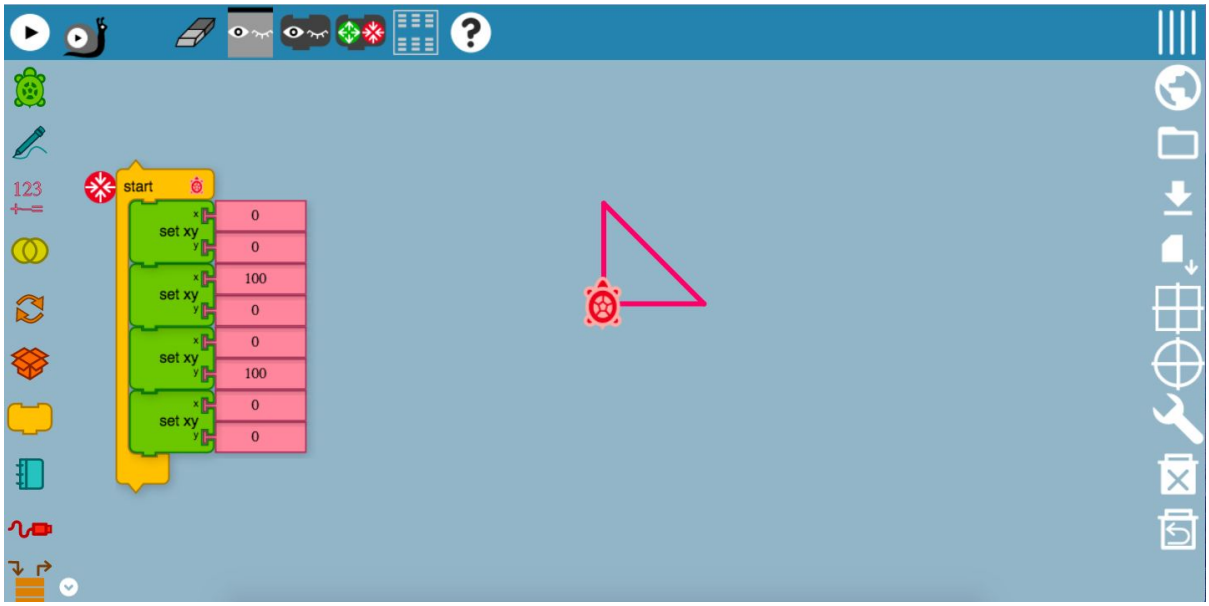
1. **Orange Turtle:** A Set-Heading Block with a positive 45 degree value allows the Turtle to turn in the clockwise direction in 45 degrees, setting the heading for the Turtle to proceed forward.
    a. Please note that the Orange Turtle is slightly different than all of the others since it will set the turtle to 45 degrees regardless of the initial heading. If for example you ran these scripts a second time, all of the turtles except the orange one would be heading 90 degrees.
2. **Yellow Turtle:** A positive 45 degree right block steers the Turtle to turn in the right direction at 45 degrees.
3. **Dark Green Turtle:** You can use Set Heading to simulate left and right by saying, Set Heading = sum of current heading and some new value. Current heading is stored in the heading block, which is 45 degrees.
4. **Light Green Turtle:** Like the Dark Green Turtle, the heading direction for this Turtle is the addition of 45 degrees to the heading block, which means that the Heading Block will now have 45 degrees and hence turn in the 45 degrees direction.
5. **Blue Turtle:** The Blue Turtle turned right by 1 degree for 45 times, which is equivalent to turning right by 45 degrees.

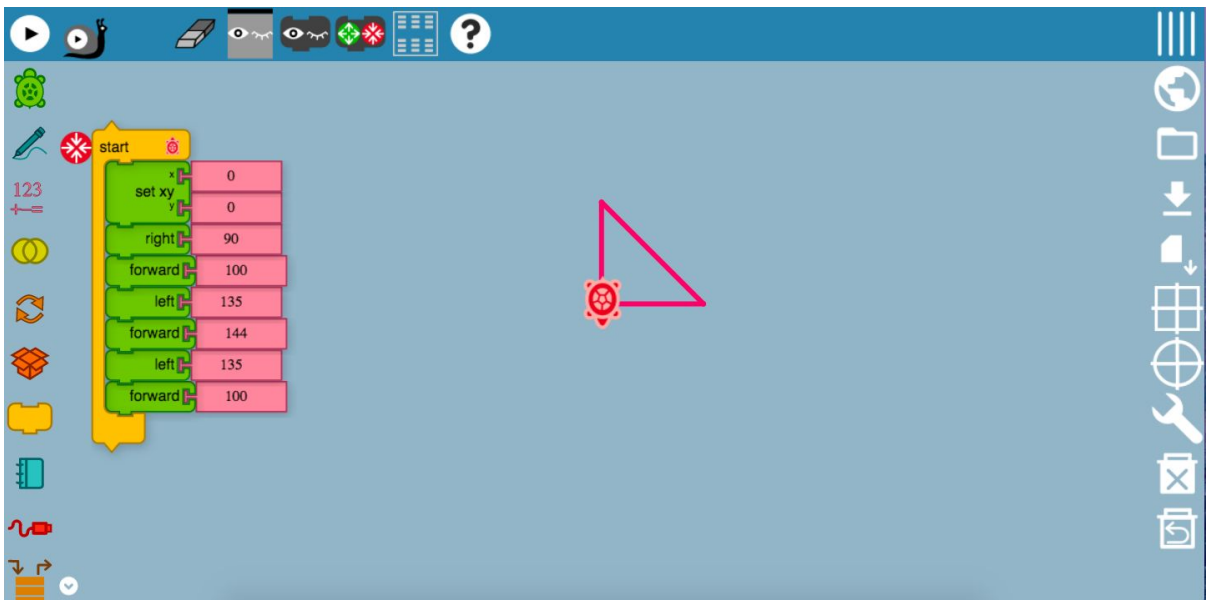
Set XY Block



Doesn't this Set XY Block reminds you of the straight-line graph? Yes, it indeed is! In this case, the Turtle has moved 100 pixels in the horizontal direction, which means that it is 100 horizontal pixels away from the starting point. Applying the same concept, it is 60 pixels vertically away from the starting point. However, have you noticed how the line is slanted and it is not exactly 60 vertical pixels away from the starting point? The y point is in fact 60 vertical pixels away from the end of the horizontal line, which forms this straight-line graph. It will be useful for you to form shapes and even applying the concept of graph. Perhaps, you may even want to use Turtle Blocks to visualize graphical equations!

Just as the Set Heading Block is tied to absolute heading, the Set XY Block is tied to absolute position. The Set XY Block is the equivalent of the Set Heading and Forward blocks in combination, but it doesn't require you to calculate the angle at which you want to go forward (or the distance) Try using it to draw a right isosceles triangle (and do the same with Forward and Set Heading.)



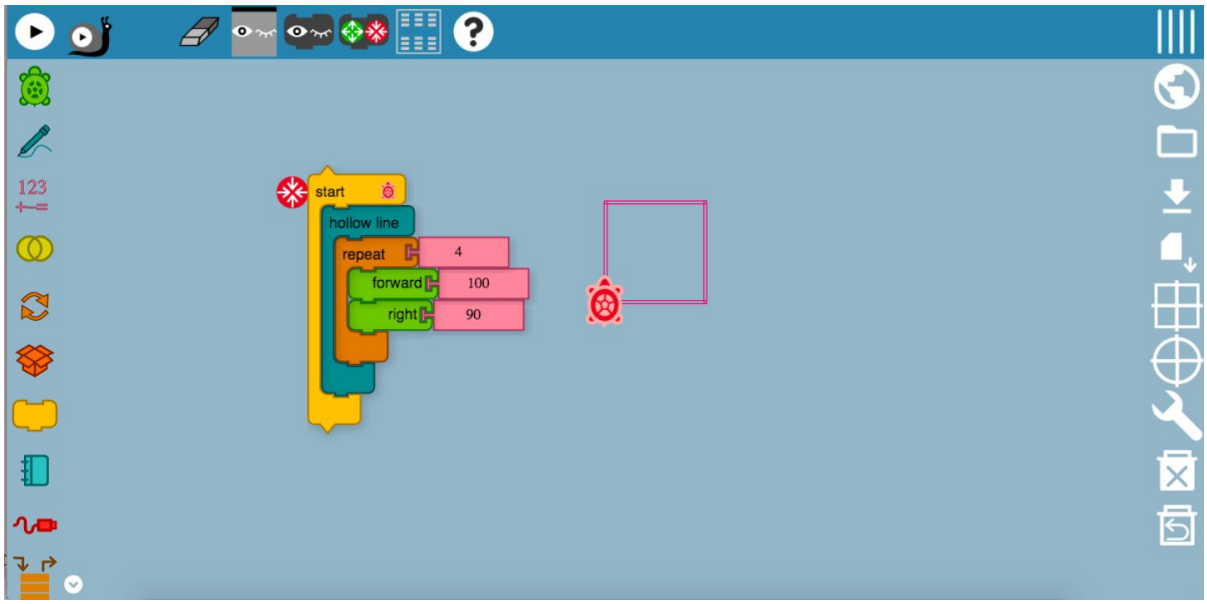


In using the Set XY Bock, it is apparent that we require no prior knowledge of the Pythagorean Theorem in order to determine the length of the hypotenuse of the triangle.

Pen Palette

The pen palette is a collection of tools and blocks that can be used to manipulate the color, thickness and many other features of the drawing. Essentially, it is the tool that draws out the lines.

Hollow Line Block

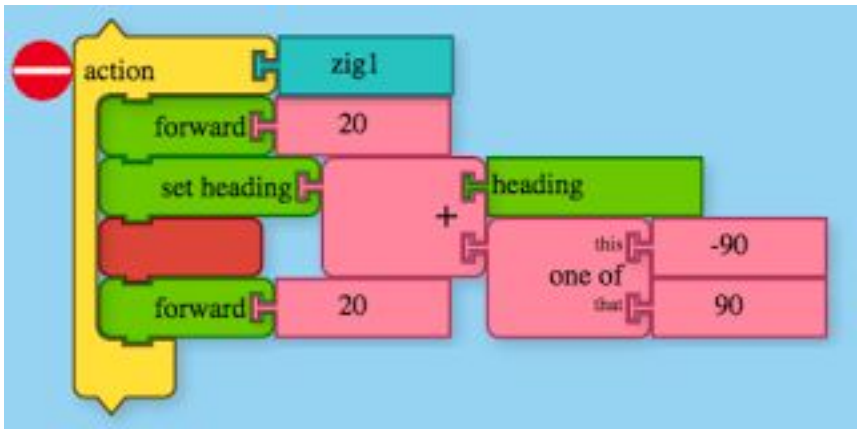


The Hollow Line Block can be used for creating hollow lines throughout the drawing, and is useful for 3D printing.

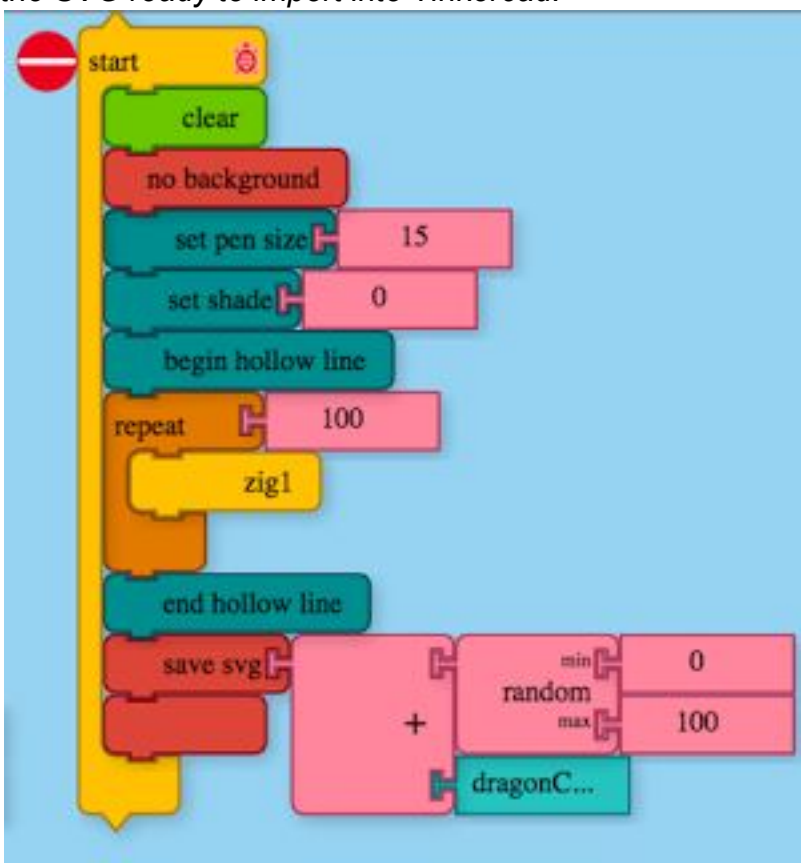
Below is a write-up from Josh Burker on Hollow lines for 3D Printing:

“

We can recreate the TurtleArt Dragon Curves procedure in Turtle Blocks. Walter also added a One of block to Turtle Blocks to make this possible. Start by creating a zig procedure.

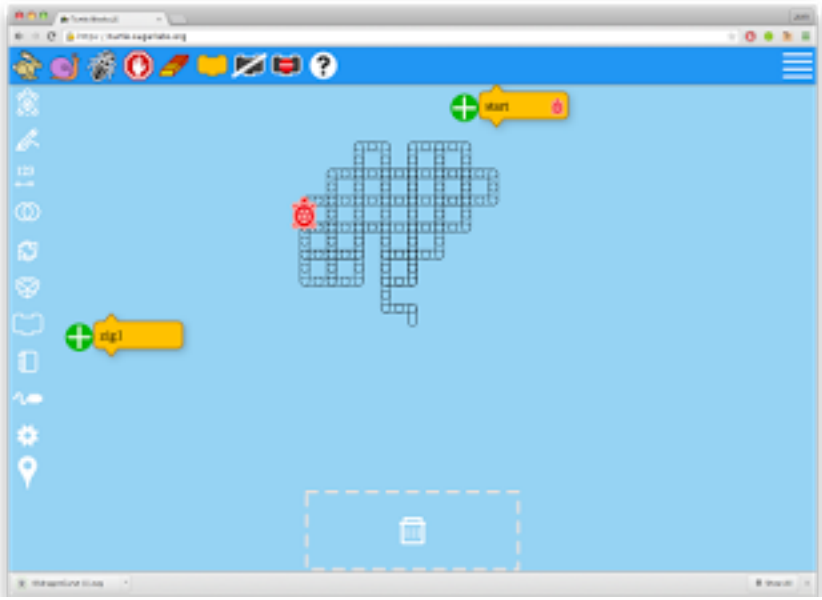


When you create your master procedure you need to add a couple of blocks to make the SVG ready to import into Tinkercad.

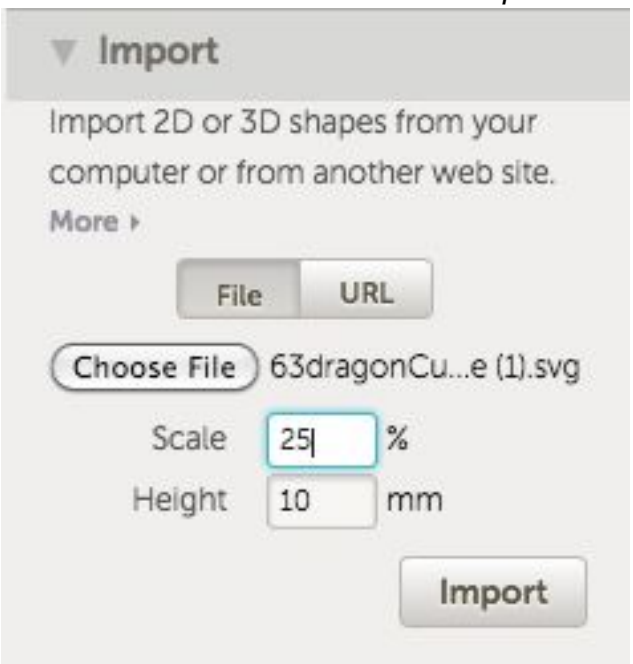


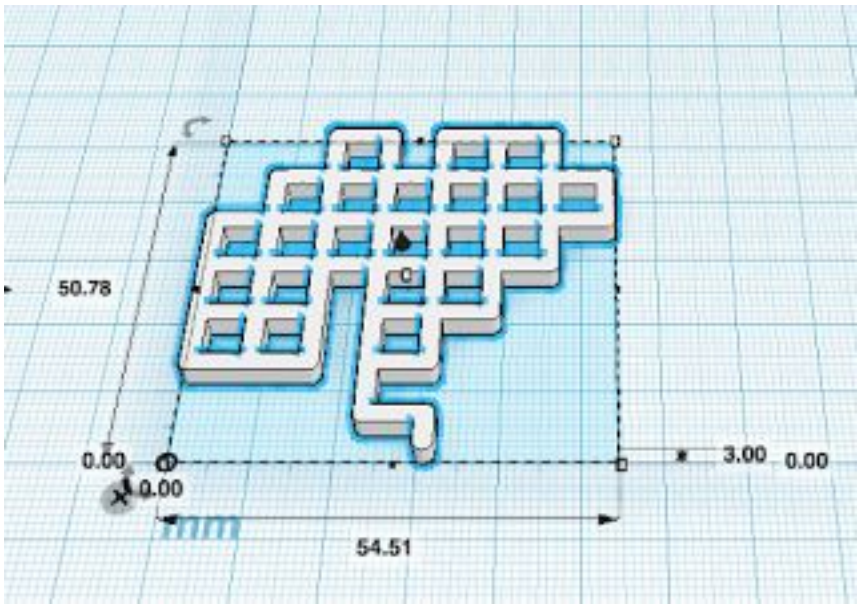
First, suppress the background by adding the red "no background" block. Next, add the begin hollow line block before the turtle starts drawing. Once the design is drawn, add the end hollow line block to the procedure. The design will look a little unusual

when you run it. An SVG file downloads at the end.

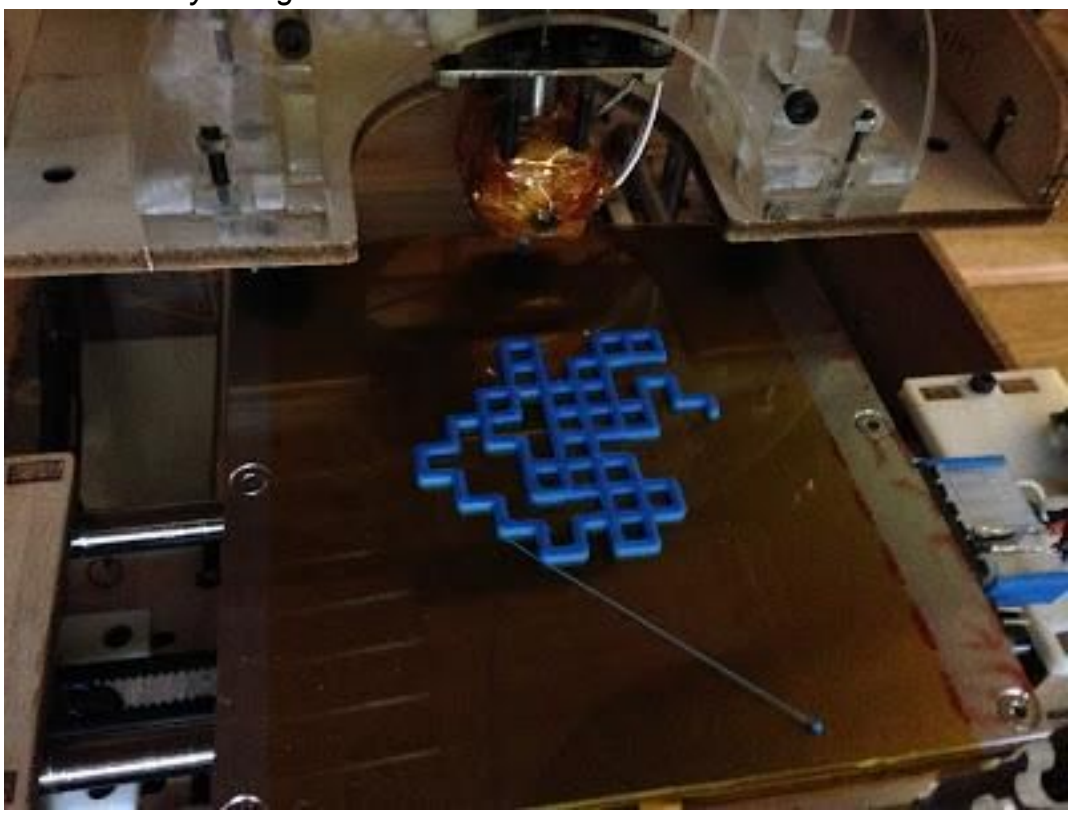


The scale at which you import your design into Tinkercad depends on the size of the build plate on your 3D printer. I sized scaled my import to 25% and further reduced the size of the model once it was imported.





The resulting model sliced fine in ReplicatorG. I printed the model with 2 shells and 5% infill on my Thing-O-Matic.



Turtle Blocks takes a few of the hurdles out of trying to get a turtle graphics project 3D printed. By streamlining the process and eliminating the intimidating step of

getting Inkscape running and converting PNG files to SVG files, Turtle Blocks should help more people experiment with programming 3D printed designs.

”

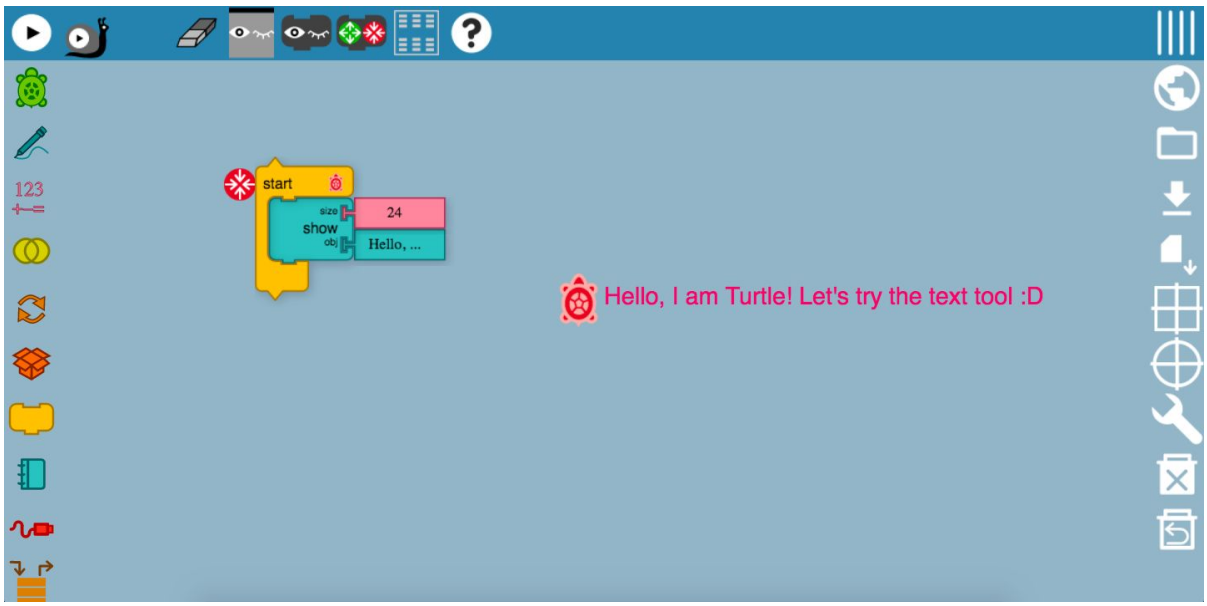
Media Palette

Turtle Blocks provides rich-media tools that enable the incorporation of sound, typography, images, and video.

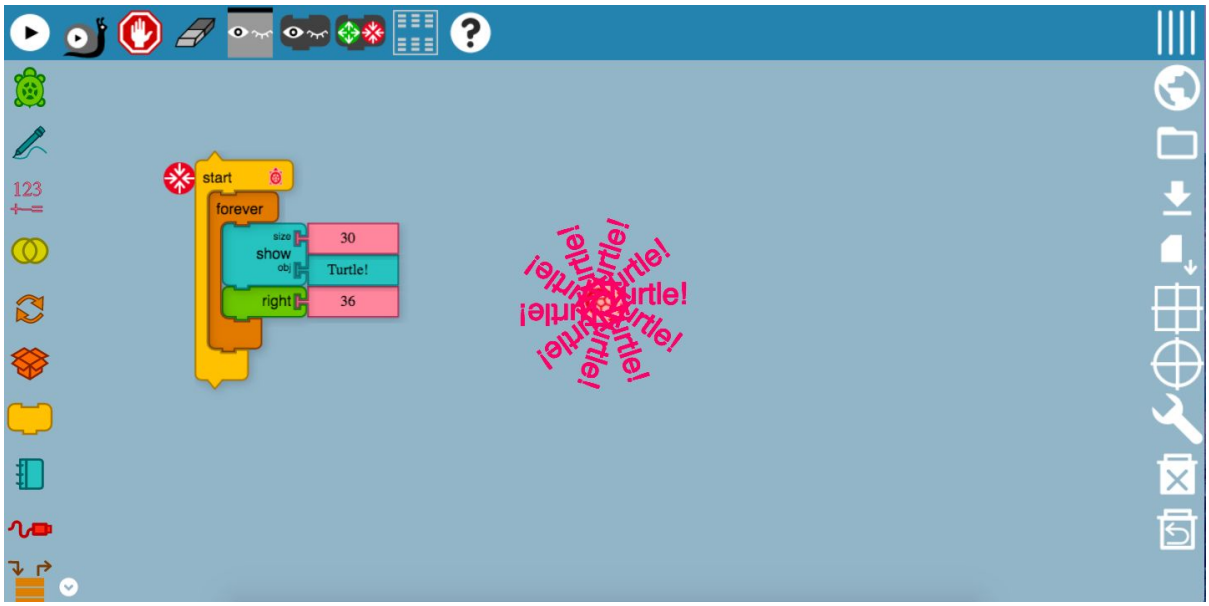
At the heart of the multimedia extensions is the *Show* Block. It can be used to show text, image data from the web or the local file system, or a web camera. Other extensions include blocks for synthetic speech, tone generation, and video playback.

Show Block

Text Tool

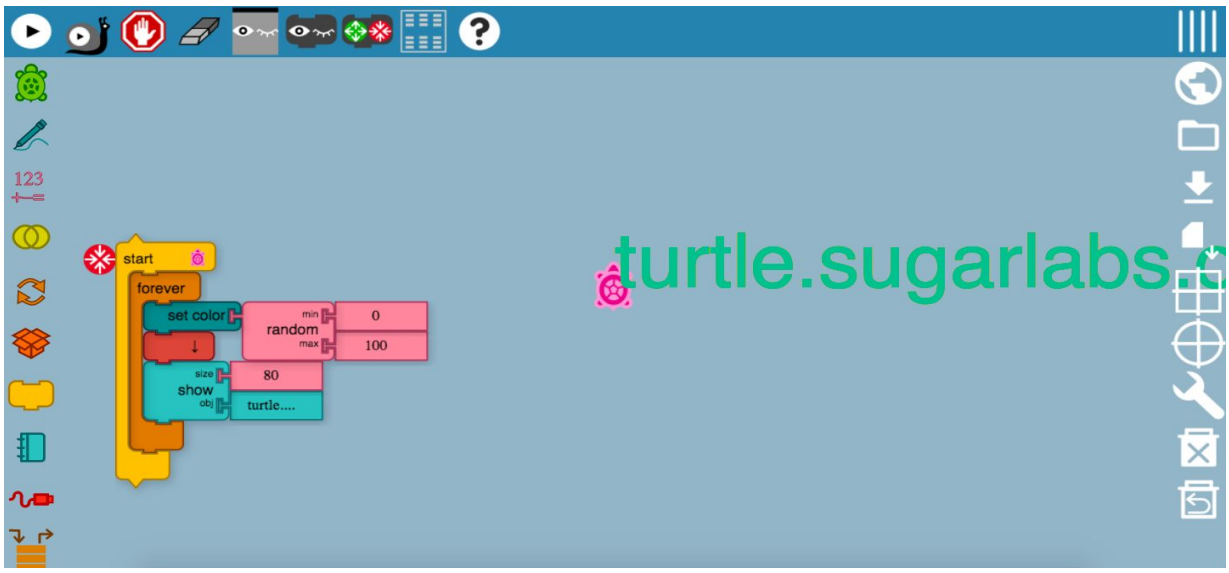


This is the Show Block! It displays the input that you have entered. For example, the size of the object here was specified to be 24, while the object, have been set to “Hello, I am Turtle! Let’s try the text tool!”. This explains what the Turtle is trying to show here. The object has been specified to be a text input. You could really try to experiment with font sizes and print out different texts on the screen!

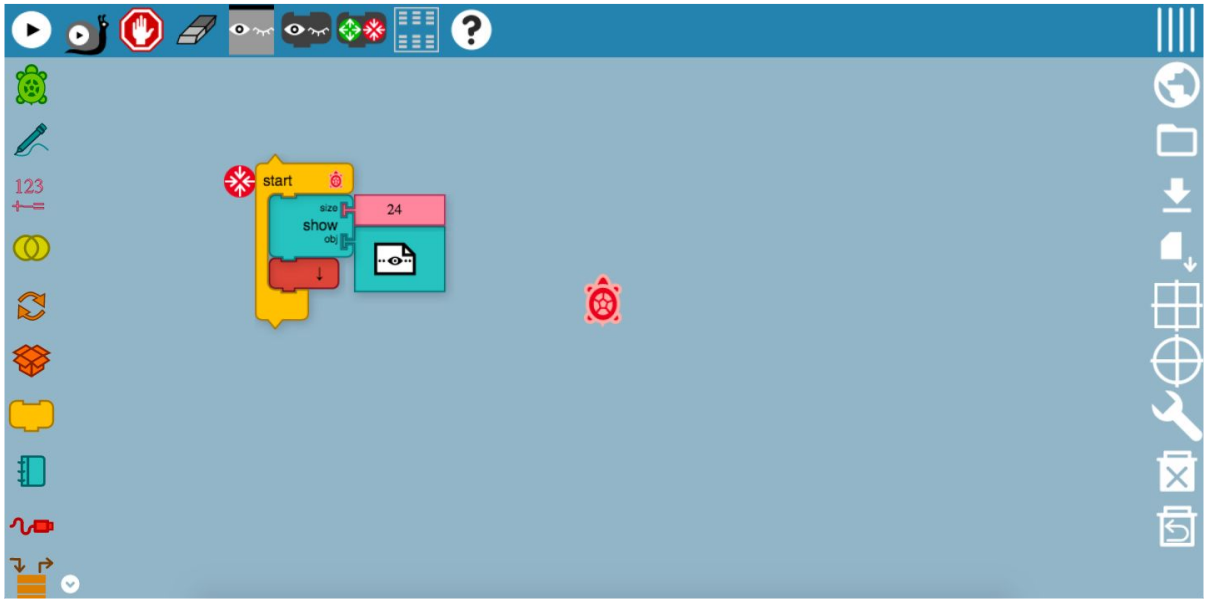


You could also generate a little animation where the text would be spinning infinitely. This can also be done for the camera tool, which is shown in the manual as well. (See Show Block, Import Media Tool)

This is an example from the Turtle Blocks planet, where the user randomize text colors:

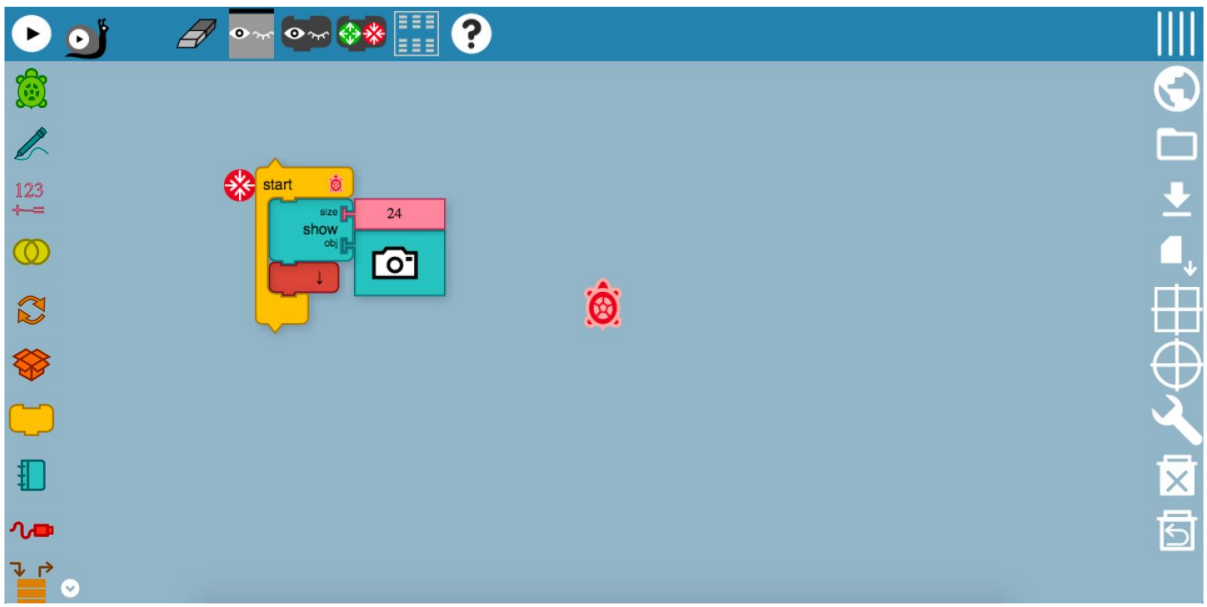


Video Tool



In the Show Block, one of the most amazing function is to display a live video camera on the canvas, which in this way, you can record anything that you want. To adjust the size of the video screen, you can also make edits to the size tool.

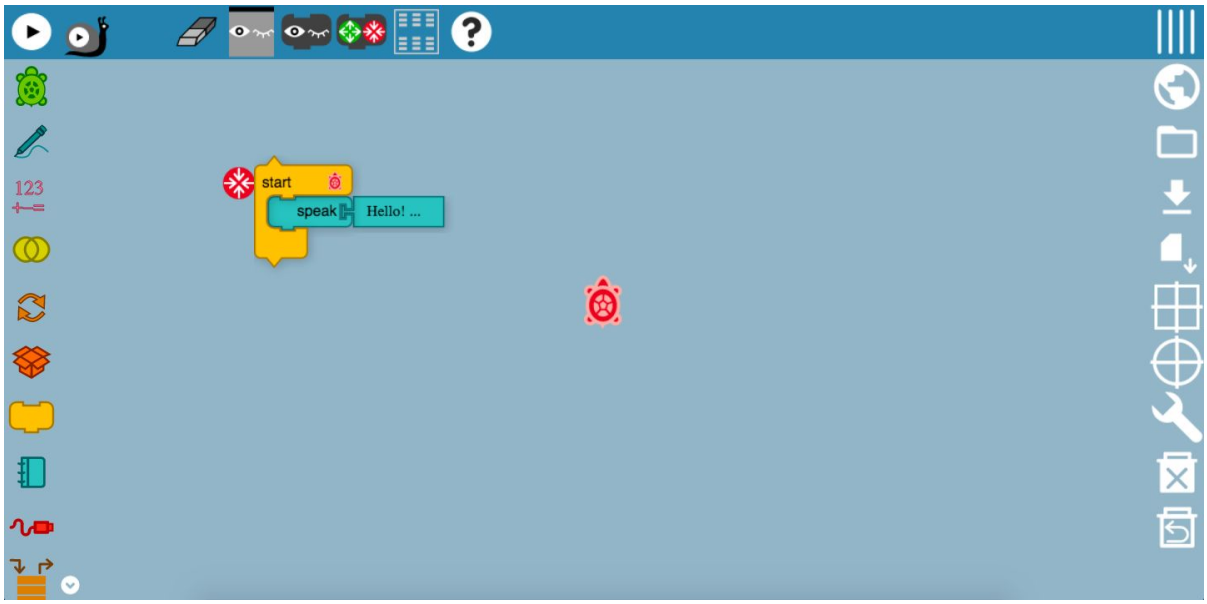
Camera Tool



Like the Video Tool, the Camera Tool displays a camera screen on the canvas. Incorporated with the Loudness Block, it can be used as an image capture after the Loudness Block starts a burglar alarm, or even a friendly “Hello” welcome. (See Sensors: Loudness Block)

Try it now!

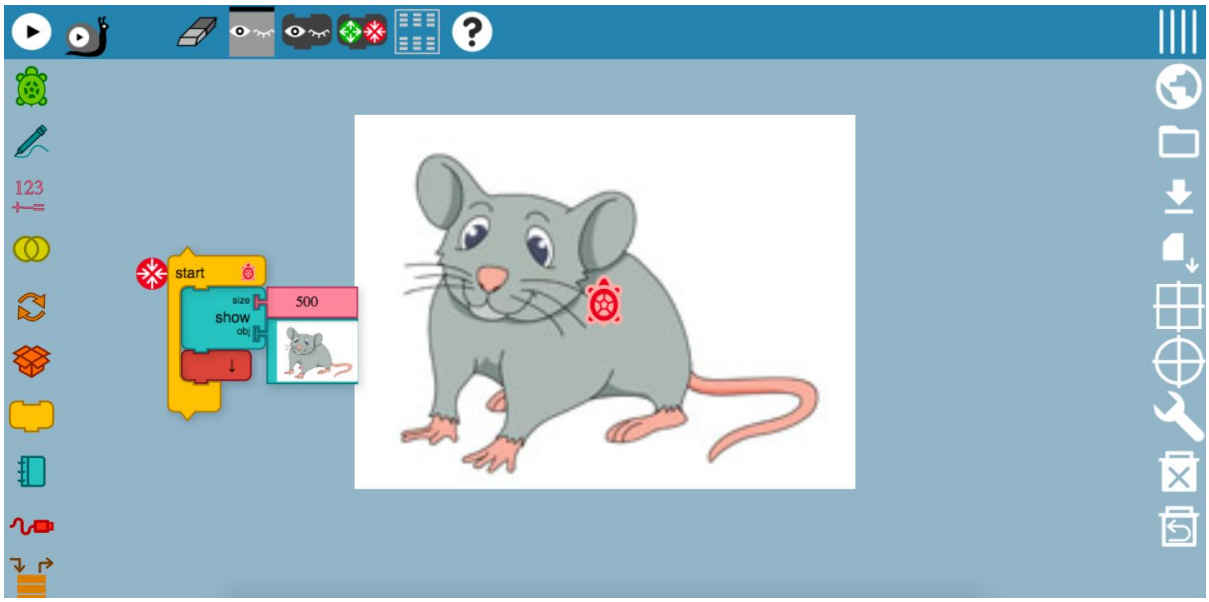
Speak Tool



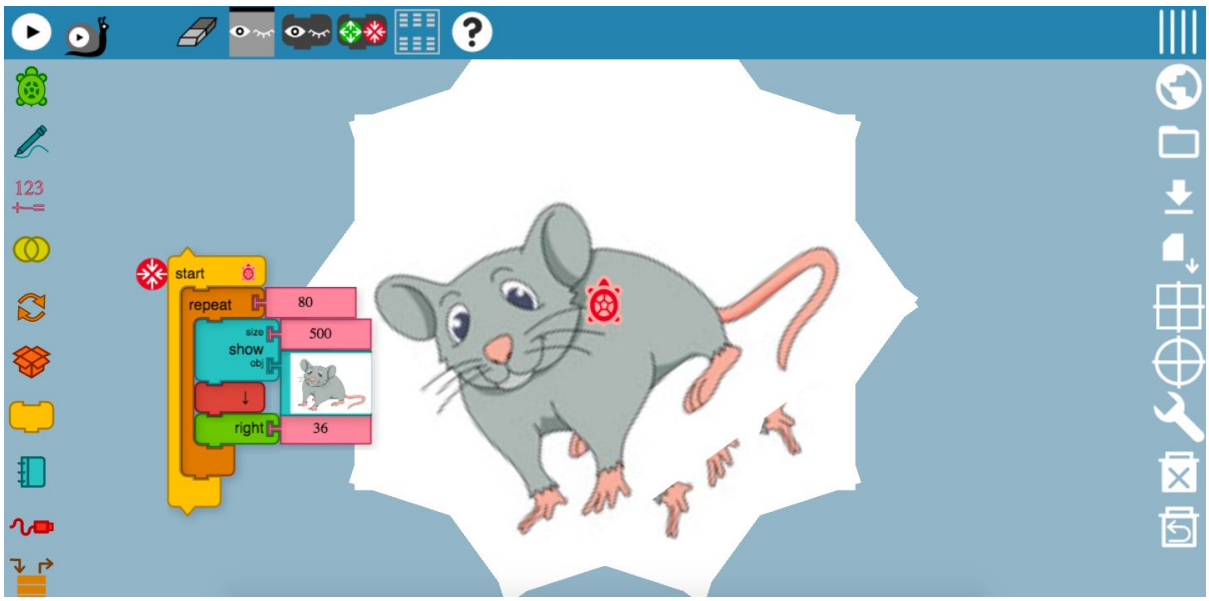
The Speak Tool is used to program the Turtle to produce sounds, of whatever that you have typed. For example, in this program, the input for the speak block is “Hello! I am Turtle.” So, turn up your volume and you can hear the sound output! Just imagine how this great function can help you to pronounce words, and even make your art even cooler as the Turtle speaks while creating shapes.

Import Media Tool

The Media Tool tool is used with the Show Block, to display images on the canvas, like this:

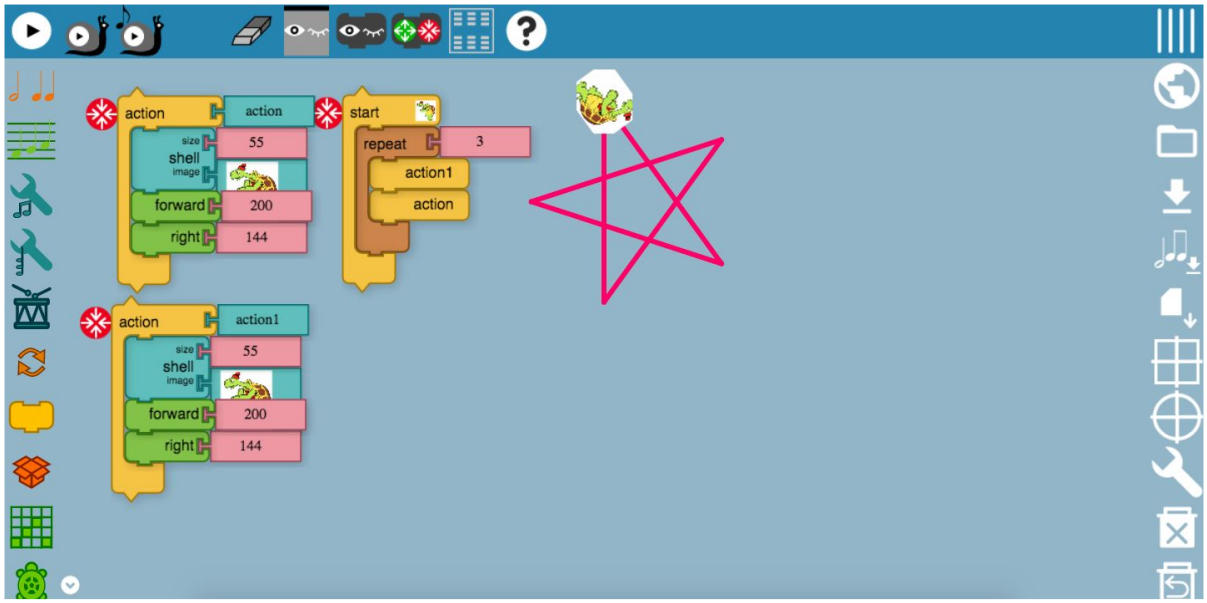


Or to show spinning images:



Avatars

Turtle Shell



Perhaps you may be thinking, why am I using a Turtle to draw? Well, did you know that you could change the image of the Turtle? We call this, changing the image of the Turtle shell. Here, the Turtle's shell was replaced with a clipart character!

Note

When you use a Show Block with an image, it puts a picture on the canvas.

When you use a Shell Block with an image, it changes the turtle to that image. The image moves as the turtle moves.

Action Blocks

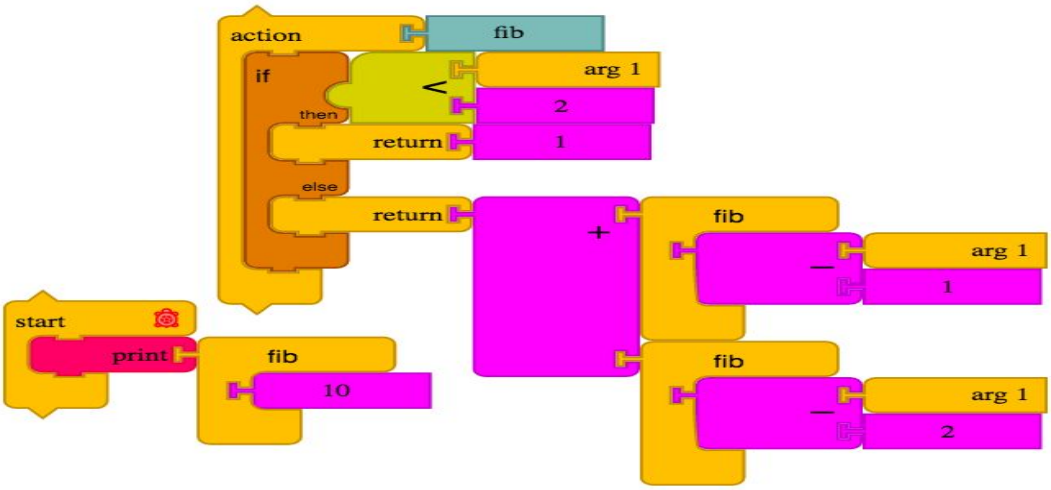
Perhaps you may have heard of Fibonacci numbers, perhaps you haven't.

Part of the Mathematics glossary: The Fibonacci sequence is a set of numbers that starts with a one or a zero, followed by a one, and proceeds based on the rule that each number (called a Fibonacci number) is equal to the sum of the preceding two numbers.

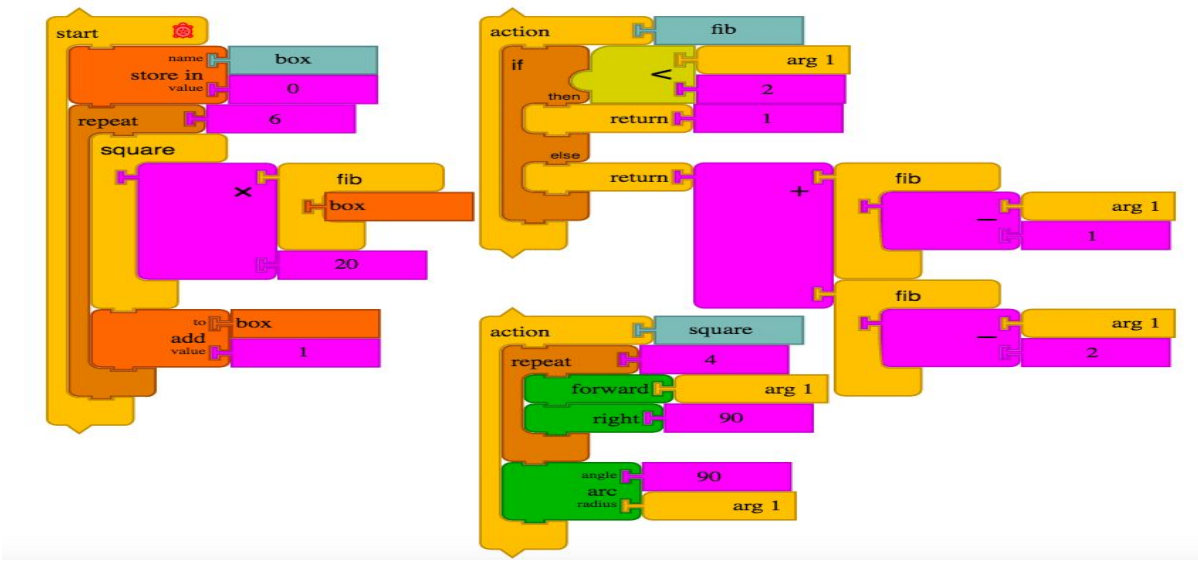
The Number Palette is often used with the Boolean Palette to create artwork based on the values and math concepts, playing around with the numbers. Well, what about creating a nautilus based on this Number Palette?

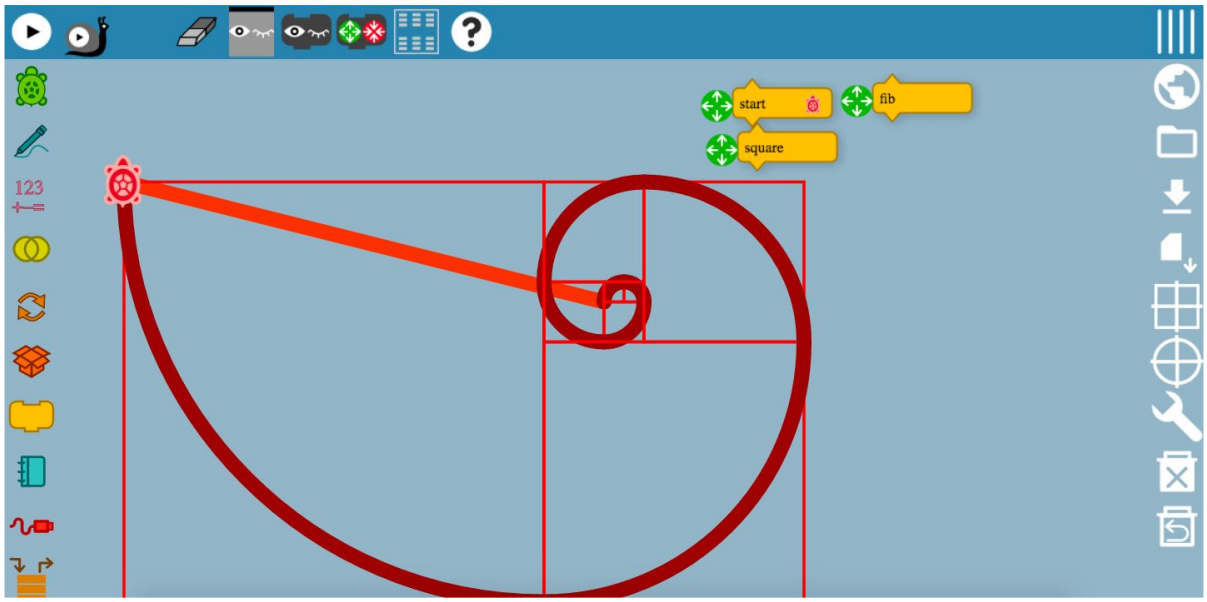
This is an example from the Turtle Blocks Guide:

Calculating the Fibonacci sequence is often done using a recursive method. In the example below, we pass an argument to the *Fib* action, which returns a value if the argument is < 2; otherwise it returns the sum of the result of calling the *Fib* action with argument - 1 and argument - 2.

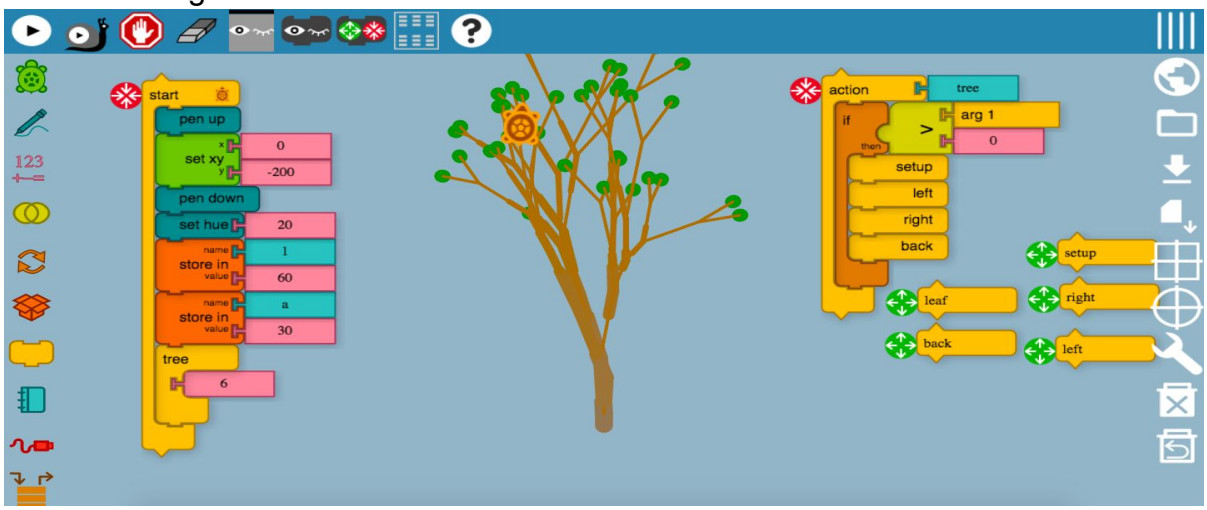


In the second example, we use a *Repeat* loop to generate the first six Fibonacci numbers and use them to draw a nautilus.





This is an example of recursion. You can pass arguments to an *Action* stack. In this example, which can be found in Turtle Blocks Planet, the user passes an argument to the If-Then Block, where if the value is greater than 0, the action will begin to draw the tree left-right-back.



Sensors

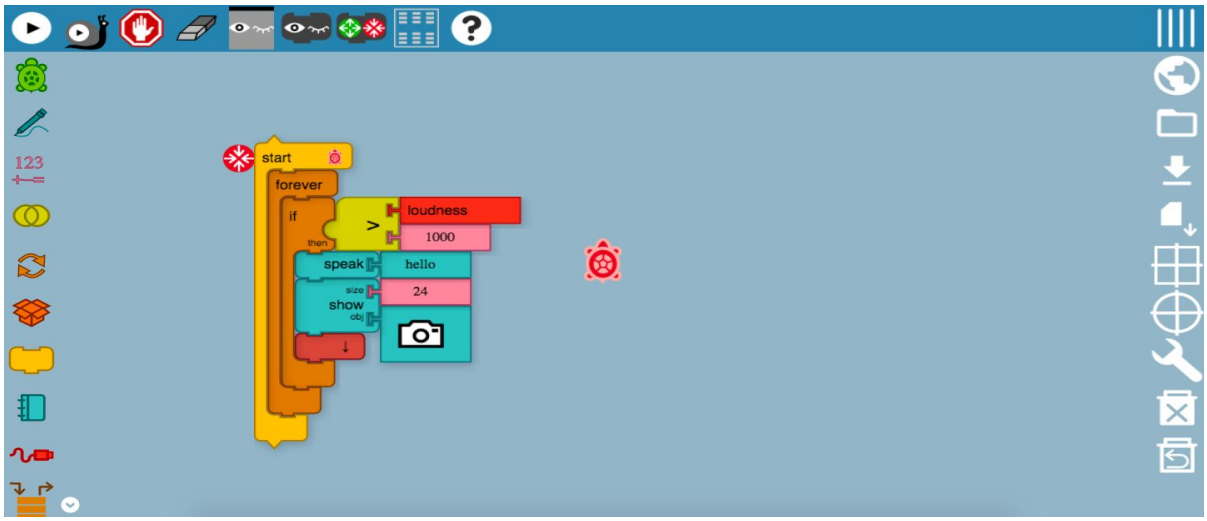
Seymour Papert's idea of learning through making is well supported in Turtle Blocks. According to Papert, "learning happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe". Research and development that supports and demonstrates the children's learning benefits as they interact with the

physical world continues to grow. In similar ways, children can communicate with the physical world using a variety of sensors in Turtle Blocks. Sensor blocks include Keyboard input, Sound, Time, Camera, Mouse location, Color that the turtle sees. For example, children may want to build a burglar alarm and save photos of the thief to a file. Turtle Blocks also makes it possible to save and restore sensor data from a file. Children may use a “URL” Block to import data from a web page.

Teachers from the Sugar community have developed extensive collection of examples using Turtle Block sensors. Guzmán Trinidad, a physics teacher from Uruguay, wrote a book, *Physics of the XO* (*https://www.gitbook.com/book/icarito/physics-with-xo/details*), which includes a wide variety of sensors and experiments. Tony Forster, an engineer from Australia, has also made remarkable contributions to the community by documenting examples using Turtle Blocks. In one example, Tony uses the series of switches to measure gravitational acceleration; a ball rolling down a ramp trips the switches in sequence. Examining the time between switch events can be used to determine the gravitational constant.

One of the typical challenges of using sensors is calibration. This is true as well in Turtle Blocks. The typical project life-cycle includes: (1) reading values; (2) plotting values as they change over time; (3) finding minimum and maximum values; and finally (4) incorporating a sensor block in a Turtle program.

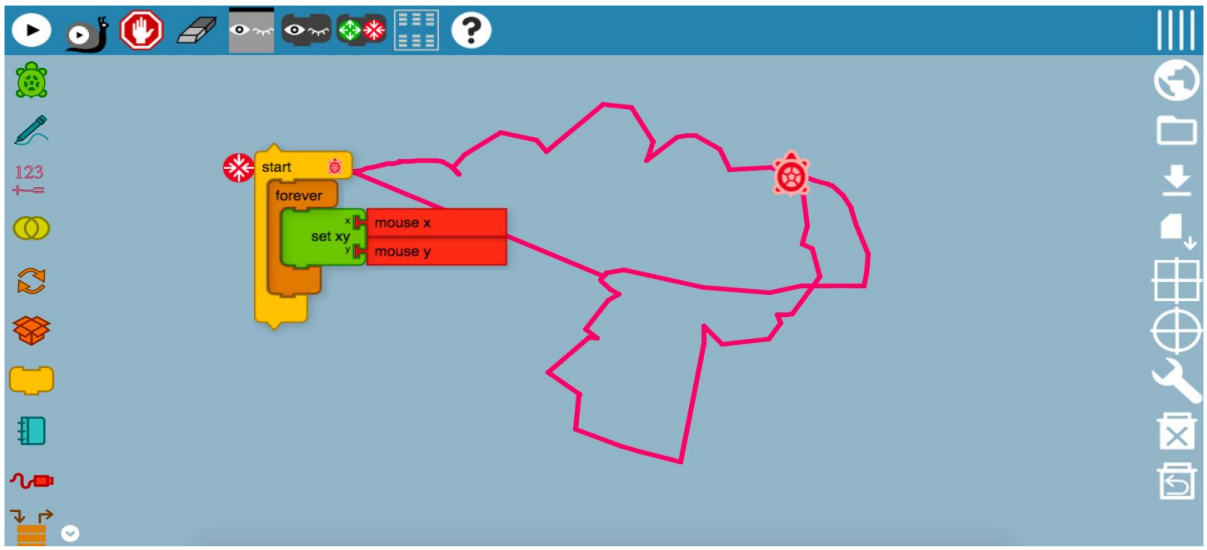
Loudness Block



The Loudness Block is now working again, thanks to the constant updates and maintenance from the developers behind-the-scene. This program may seem slightly complicated, but don’t worry! The whole program was enclosed within a Forever

Block, which means that this program would continue running on forever and ever, with no breaks and interruptions. Next, the If-Then Block would carry out the action. Firstly, if the Turtle sensed a loudness value of 1000 counts from the surroundings, it will say, “Hello”, and even take a snapshot of the surroundings!

Mouse Blocks

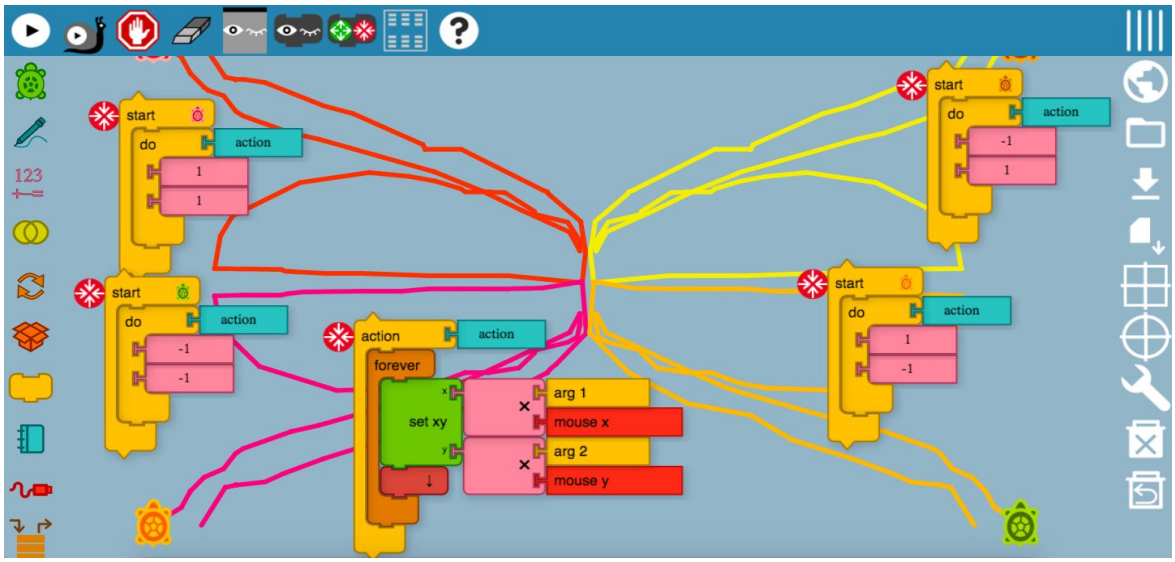


This Mouse Block is really nice to draw out different images. You can use a mouse-pad or even a mouse! Again, this program has been set to run forever without any interruptions, so one could continue drawing on and on unless the user stops the program. So, remember the Set XY Block? This block sets the x and y coordinates of the mouse and the Turtle will draw in the direction of wherever the mouse direction goes.

Writing your own paint program is empowering: it demystifies a commonly used tool. At the same time, it places the burden of responsibility on the programmer: once we write it, it belongs to us, and we are responsible for making it cool.

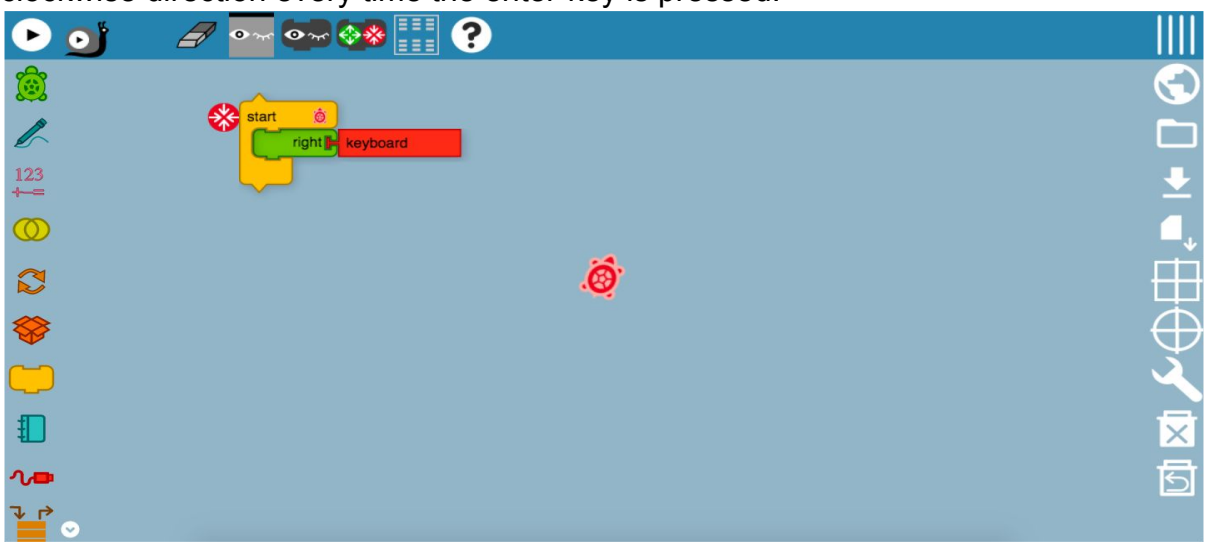
This is an example of using the mouse blocks together with math blocks to create reflection paint:

By combining multiple turtles and passing arguments to actions, we can have some more fun with paint. In the example below, the *Paint Action* uses *Arg 1* and *Arg 2* to reflect the mouse coordinates about the y and x axes. The result is that the painting is reflected into all four quadrants.



Keyboard Block

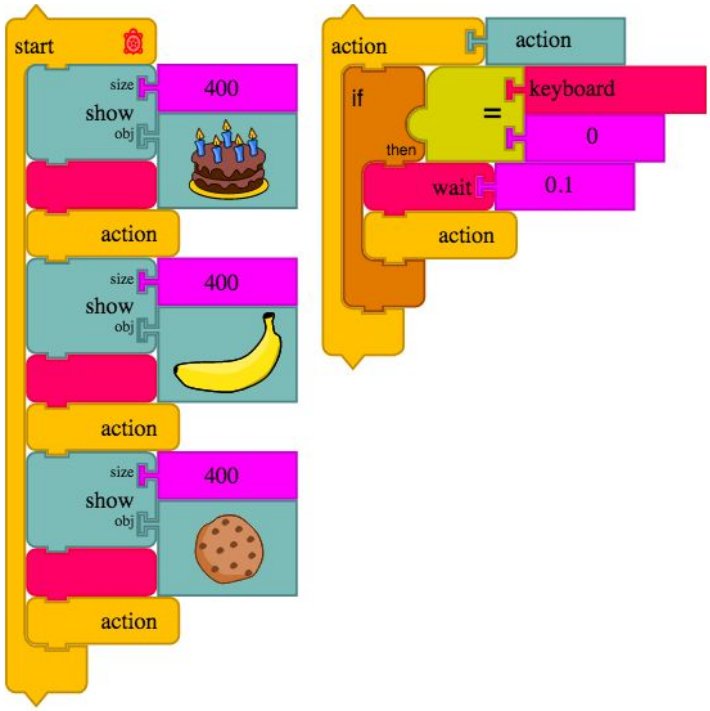
This is an amazing example of the Sensor Tool – the Keyboard Block! Just as how you would have used to control the Turtle by setting values, this will move the Turtle using the enter-key on the keyboard! In the image above, the Turtle is moving in a clockwise direction every time the enter-key is pressed.



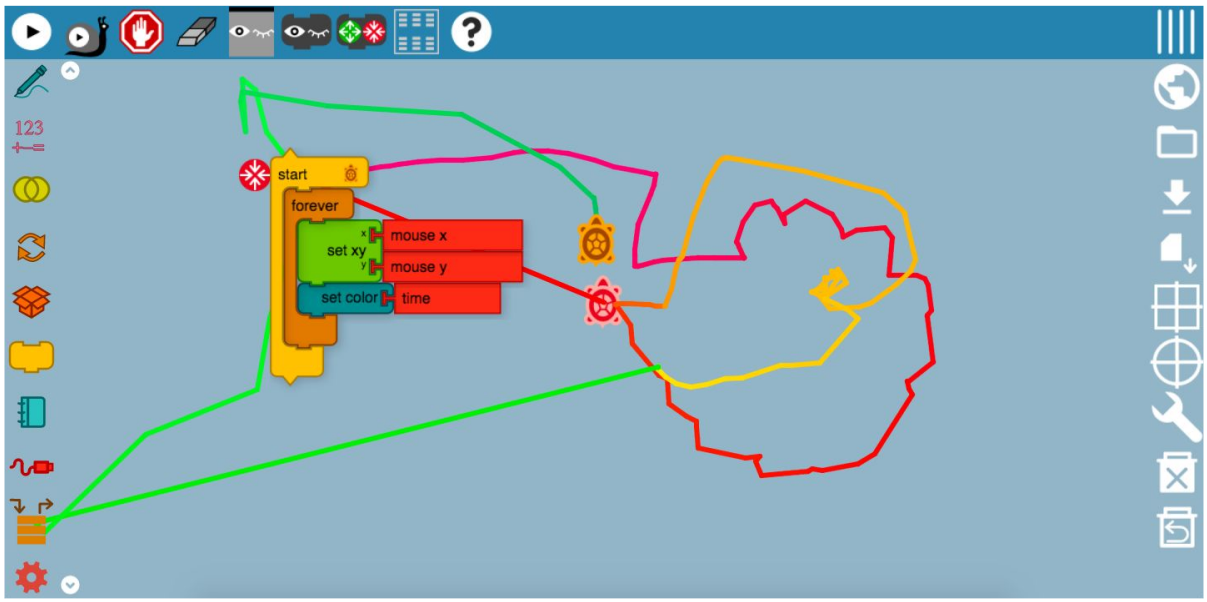
Below is an example from Turtle Block Guide on using keyboard for Powerpoint:

Why use Powerpoint when you can write Powerpoint? In this example, an Action stack is used to detect keyboard input: if the keyboard value is zero, then no key has

been pressed, so we call the action again. If a key is pressed, the keyboard value is greater than zero, so we return from the action and show the next image.



Time Block

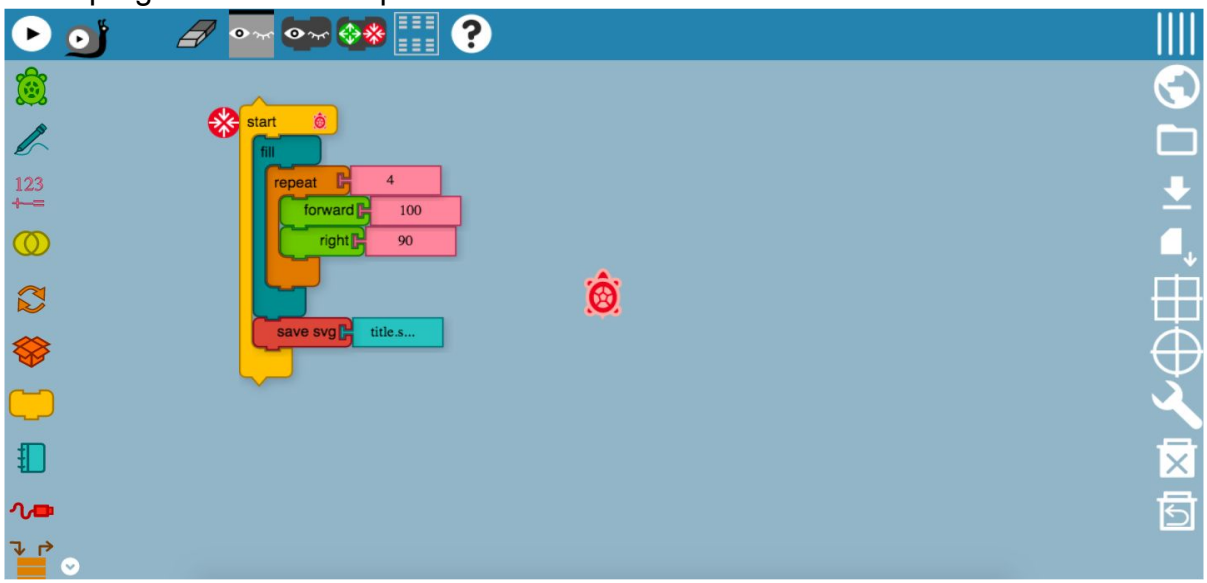


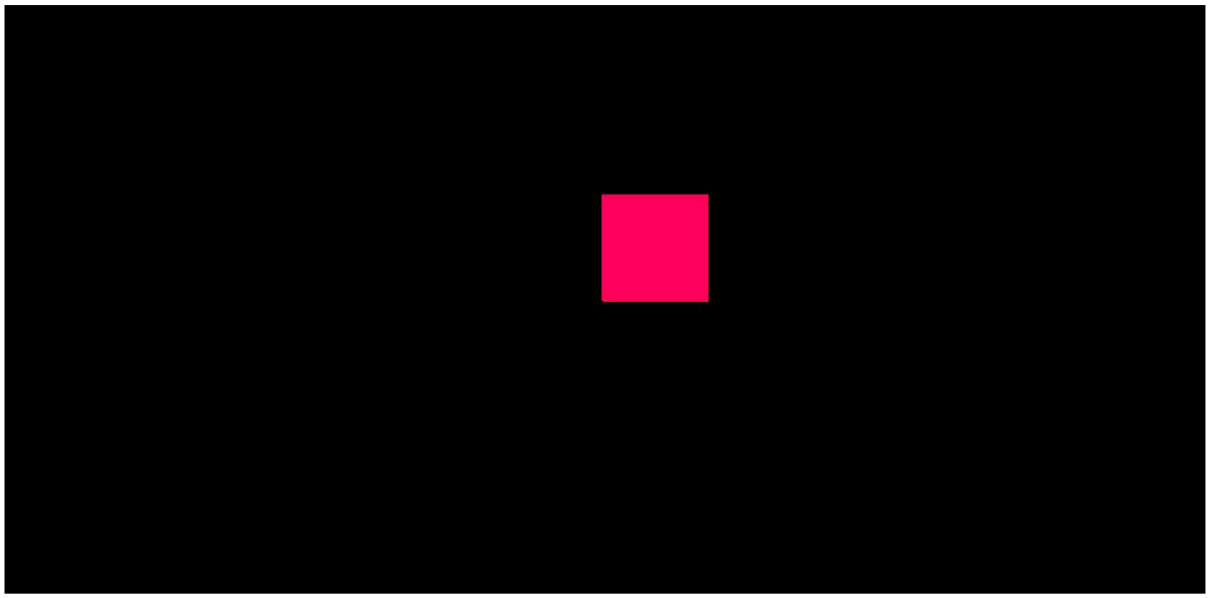
In the previous Mouse Blocks write-up, the color of the drawings remain the same throughout, and this makes it really hard to beautify your drawing. Here, a Time Block was inserted as an addition to the Mouse Blocks, so that the color of the pen tool would change after a set period of time, giving your drawings the extra touch of creativity.

Extras Palette

Save SVG Tool

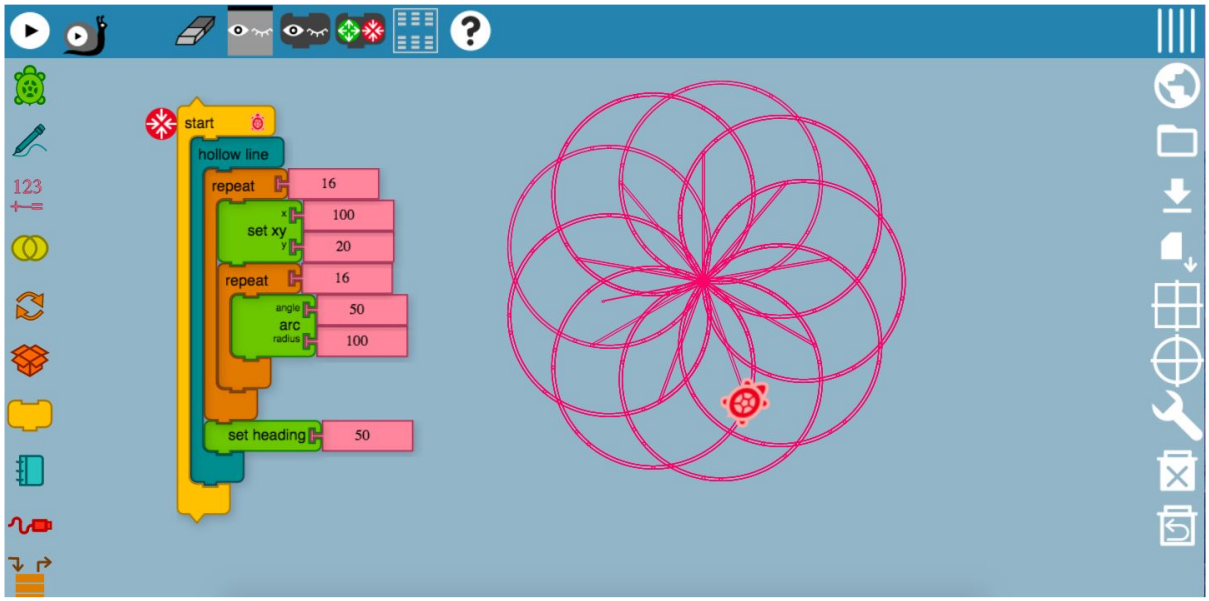
As it can be seen from the image, I have created a colored square using the fill and movement tools. After which, the SVG Tool would be able to save the output of this program, which is the colored square into my files. It is useful in showing the output of the program without the palette and blocks.



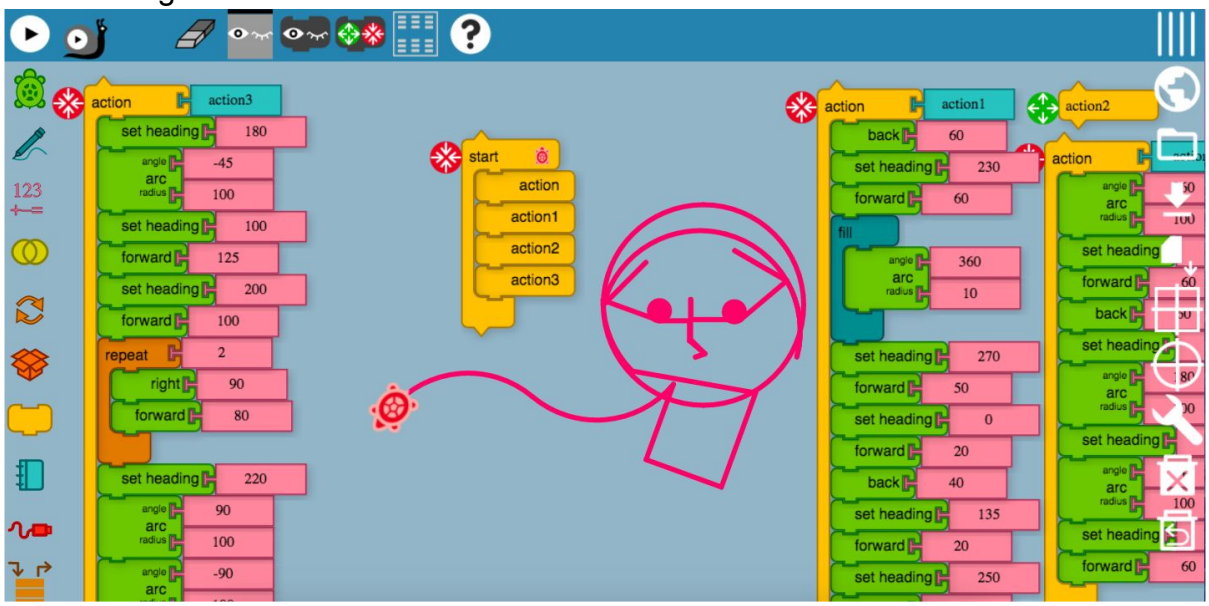


Mashing it All Up

Original Examples:

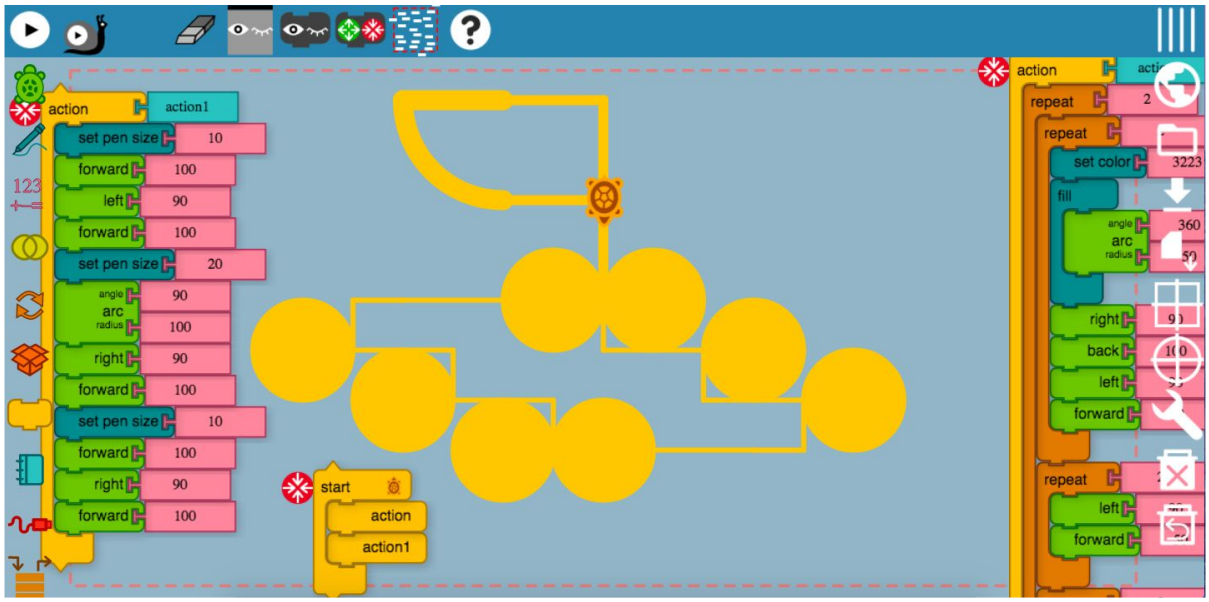


Simple Geometry Art Design: Set-XY Block, Flow Blocks, Angle-Arc-Radius Block, Set Heading Block and Hollow Line Block



Little Boy with a Party Horn: Set Heading Block, Angle-Arc-Radius-Block,

Movement Blocks, Flow Blocks, Fill Blocks



Weird Tank Machine: Movement Blocks, Angle-Arc-Radius Blocks, Pen Blocks, Fill Blocks

Sources

http://whatis.techtarget.com/definition/Fibonacci-sequence

https://en.wikipedia.org/wiki/B%C3%A9zier_curve

https://upload.wikimedia.org/wikipedia/commons/d/d0/Bezier_curve.svg

http://wiki.sugarlabs.org/go/File:TurtleCard-7.png

http://wiki.sugarlabs.org/go/Activities/Turtle_Art

https://www.tutorialspoint.com/computer_graphics/computer_graphics_curves.htm

Credits

Mr Walter Bender

https://turtle.sugarlabs.org/

https://github.com/walterbender/turtleblocksjs/blob/master/guide/README.md