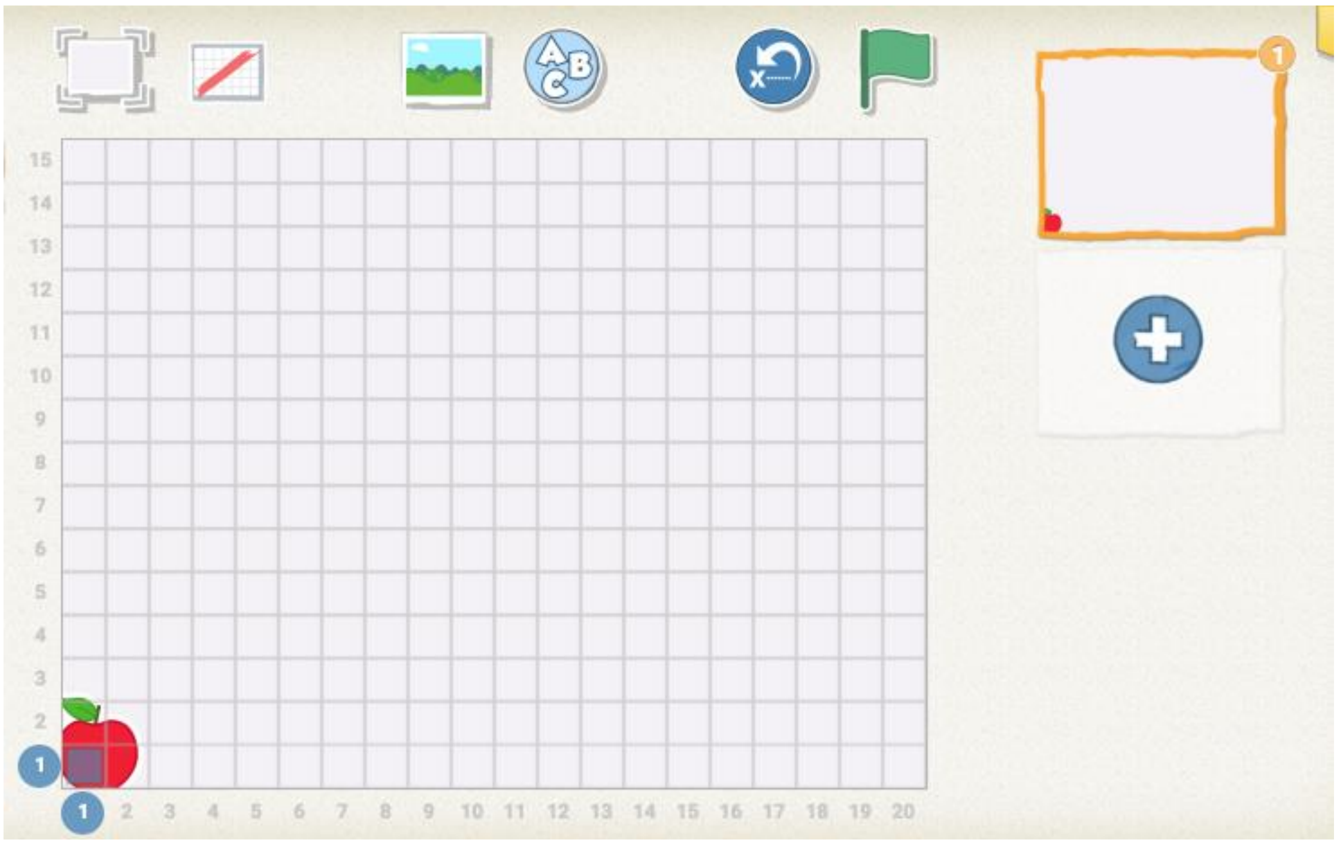
Let’s Get Moving	Grade 1 and 2
Coding Handout	

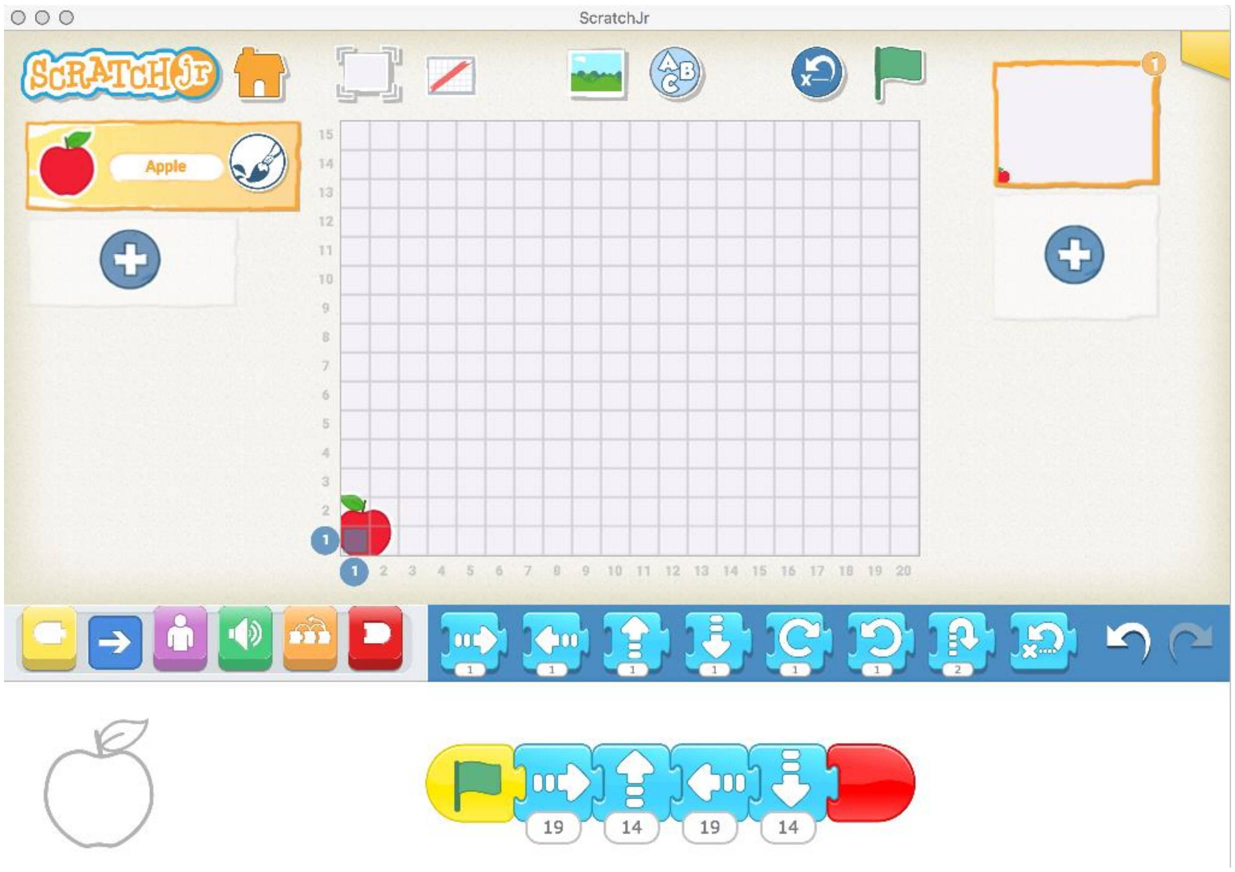
Make your sprite move through geometric shapes
Firstly, we’ll take a look at how to create a path on the screen in a geometric pattern with a sprite and the motion code blocks. We’ll have the sprite move through the largest rectangle possible.

To begin, show the grid on the project stage. Choose a sprite and place it in one of the outermost corners. We chose the apple because it is small. Some sprites have an image that is much larger than the ‘centre square’ it is following. The centre square that we are directing, is highlighted on the image when we are using the grid.



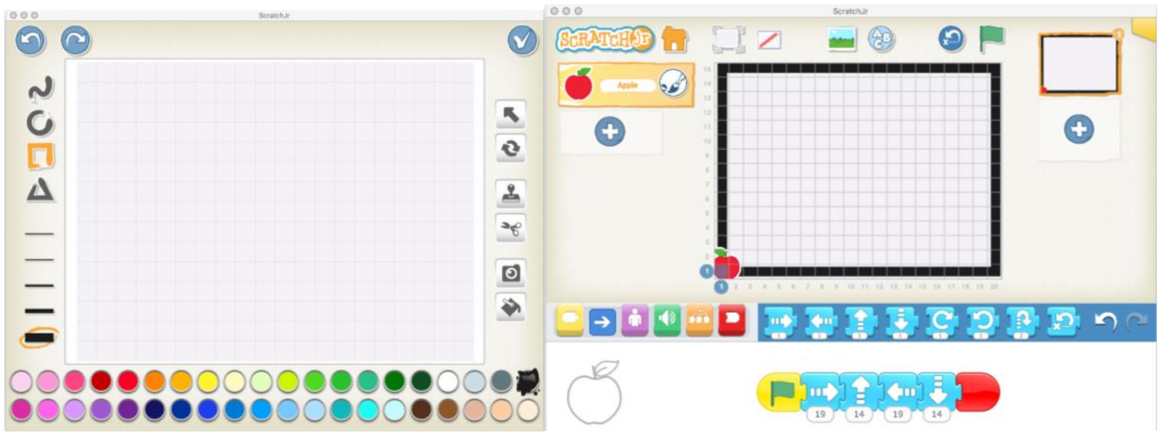
Task 1: Large Rectangle
The largest rectangle possible will be perimeter of the project stage. To go to the end of the grid, students can either count the number of squares to their destination or perform a subtraction. For example, if I want the apple to move to the bottom right corner of the stage, I can do 20 – 1 = 19. We can then do another subtraction, 15 – 1 = 14, to determine the length of the other sides of the

rectangle. Once we know the length of the sides of our rectangle, we can put them into the code. We'll need a start block, 4 motion blocks and an end block.



The code will begin when the green flag is clicked. The sprite will move 19 spaces to the right, 14 spaces up, 19 spaces to the left and 14 spaces down to complete the largest rectangle possible on the project stage.

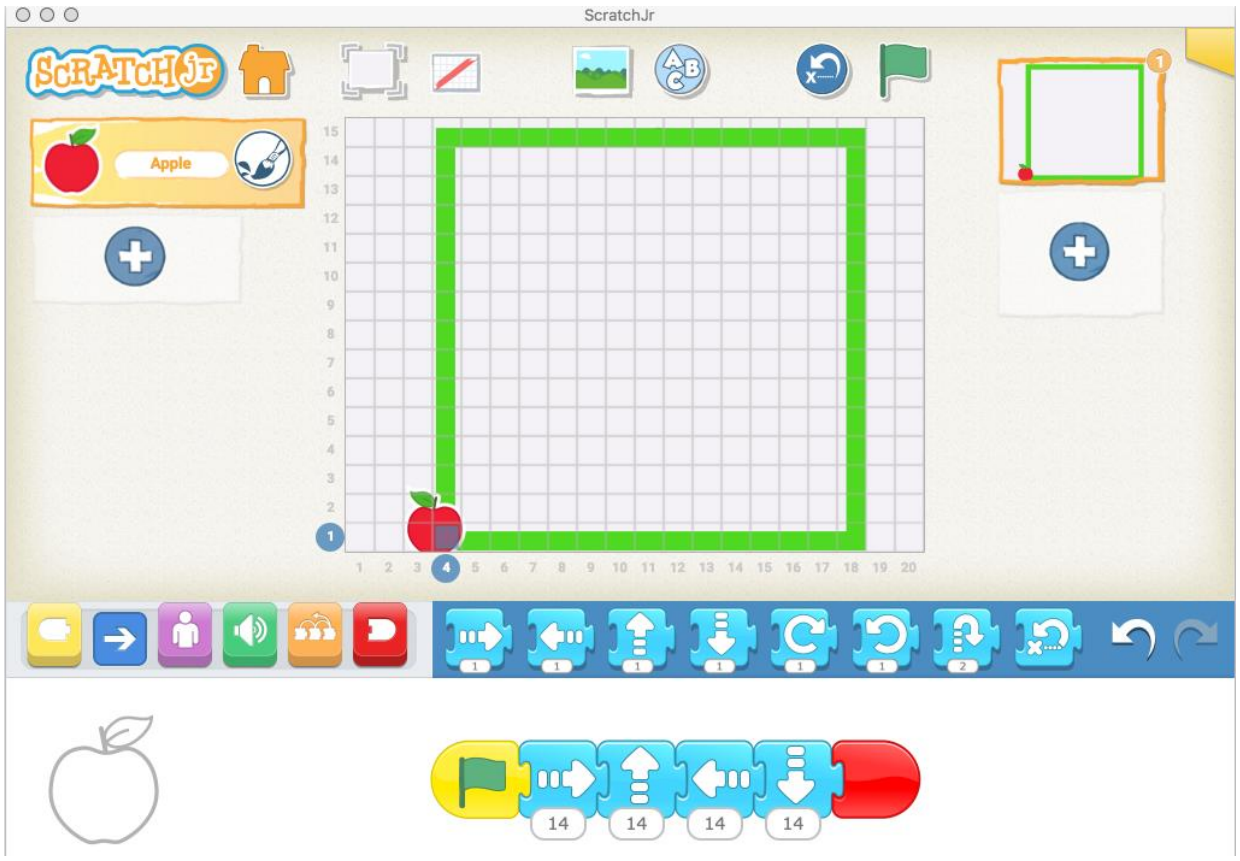
It may also be helpful to use the background paint editor to draw the shape out first and then have the sprite follow it.



Task 2: Large Square

In order to have a sprite move through the largest square, students will need to determine the size of the largest square that can be accommodated in the grid using the given information. In order to draw a square, all sides must be equal. Since the grid is shorter than it is wide, the height of the grid will be the limiting factor for the sides. The grid is 15 squares tall, so the width will need to be 15 squares as well.

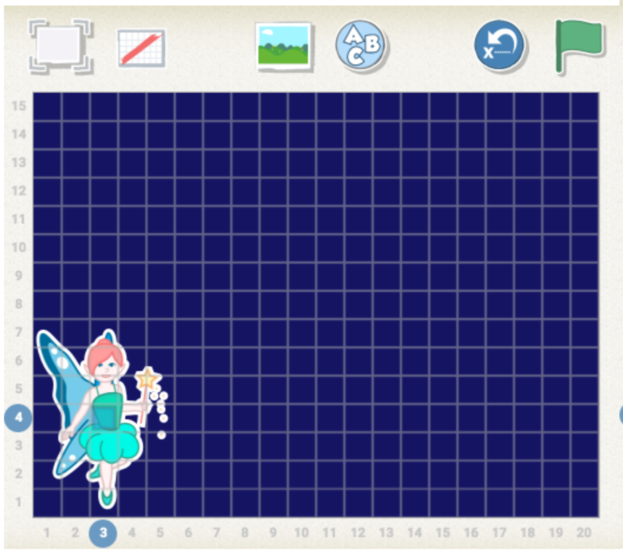
Again, it may be helpful to draw the shape in the background before adding the sprite. Since the background editor doesn't show the grid numbers students will have to count out 15 squares wide and use the rectangle drawing tool to add the square. They can then use the arrow tool to move the square within the stage. We can see the subtractions better if the lines of the square fall between the smaller squares on the grid. In the image below, we see 18 – 4 = 14.

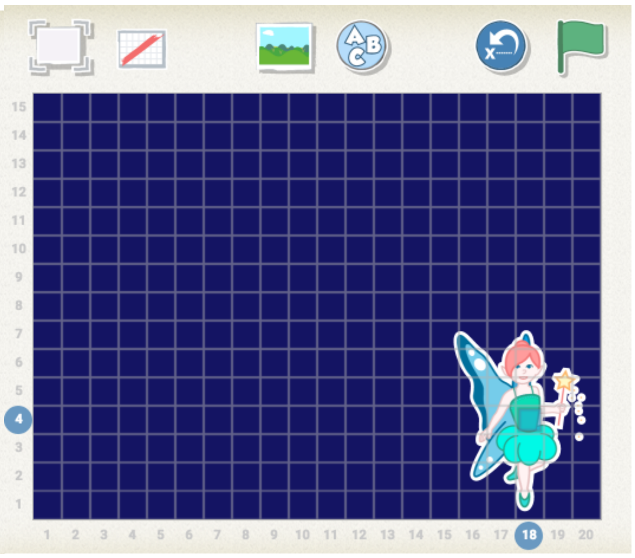


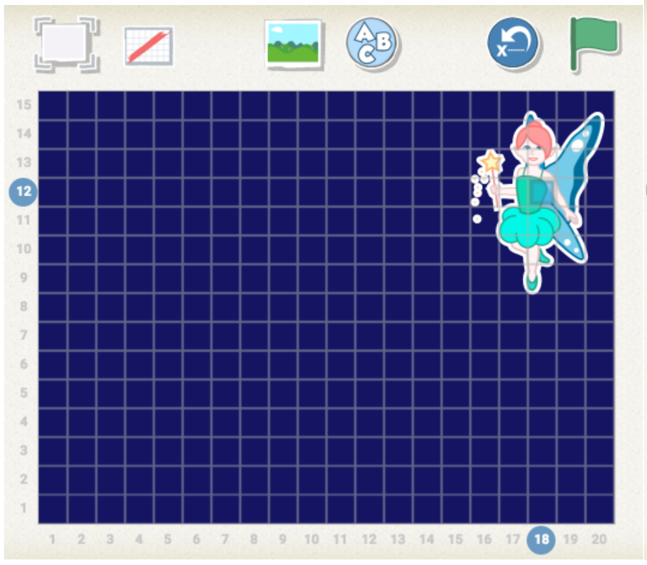
This activity can be done together on a smartboard or independently on iPads. If students are looking to be challenged, you can show how to do one of the shapes and then have them try to code for the other by themselves.

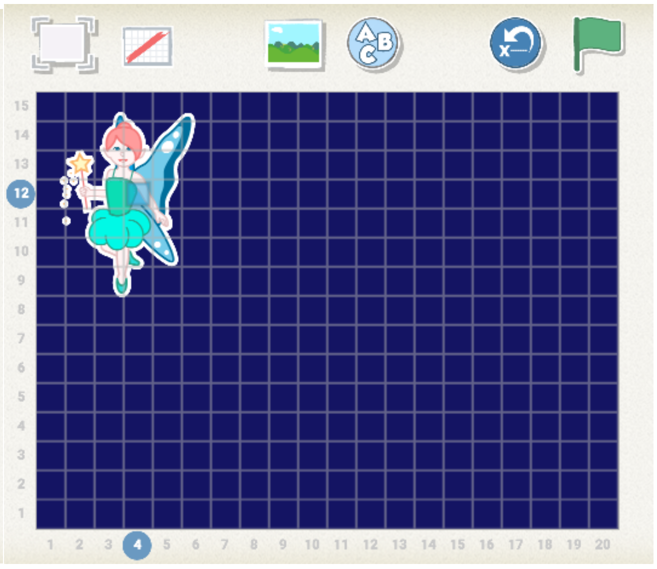
Task 3: Visible Rectangle

For the next step in the activity, the character will be replaced with something larger. Students can be challenged to have their sprites draw through rectangles around the perimeter of the project stage while keeping their sprites fully visible through the entire motion. This can be accomplished with a little trial and error by placing the sprite in each respective corner to see how it will stay visible as it moves around the edge.

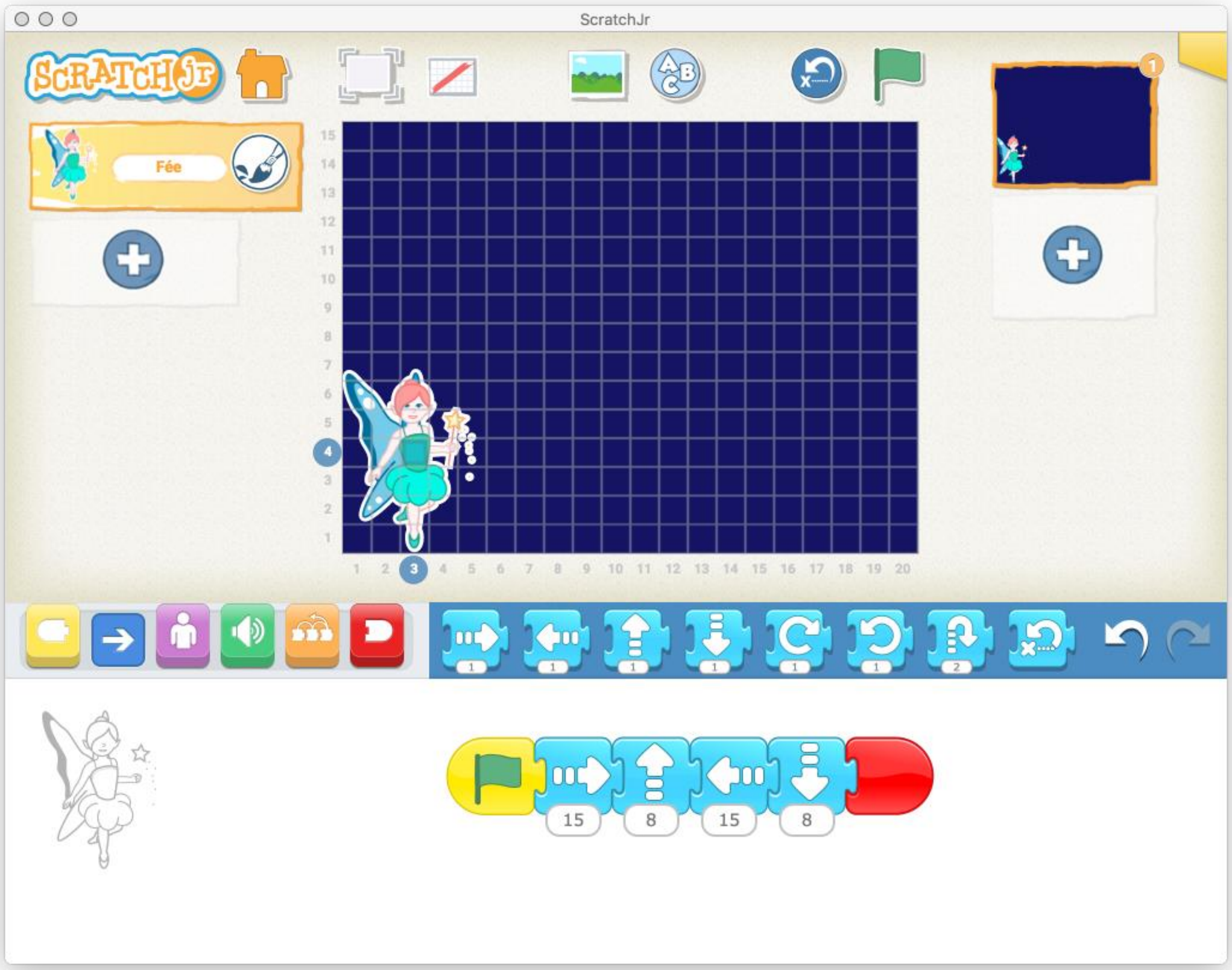




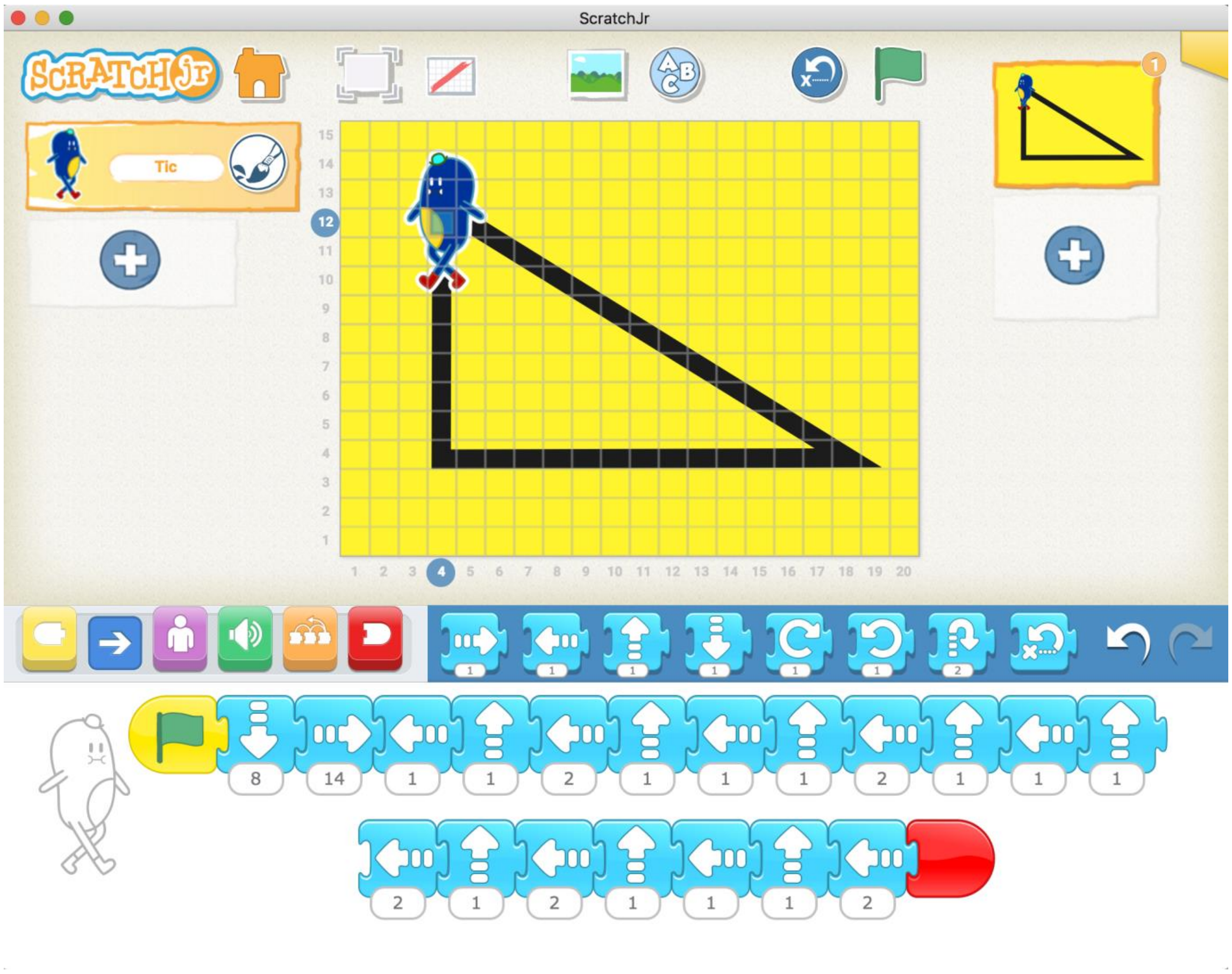




Once we know how far the sprite can go in any direction, we can see how many steps it can take along the grid. Along the height, we see the top position is 12 and the bottom position is 4. Twelve minus four gives us 8 steps. Along the width, we see the rightmost position is 18 and the leftmost 3. Eighteen minus three gives is 15 steps. The final code for this example is shown below. The code for different sprites however will differ. The code is easy to check by running the algorithm by clicking the green flag to determine if the sprite stays visible throughout.



Extension: Have students draw other geometric shapes in the background editor and discuss why it might be difficult to code a path for them. Ex: circle or triangle. The paths of these shapes don't always fall squarely on one single square. We are confined to the rules of our code blocks so we can't move in half squares or at angles. We can, however, approximate the movement along a curved or angled line. Have students draw a right-angle triangle in a background. Have the students code a path around it. They should start by completing the square corner. Those steps will be easy. They will then go through a process of trial and error to complete the last edge of the triangle. They should test their code often to make sure they aren't veering off path. Even if the sprite doesn't exactly follow the shape, keep in mind, it is an approximation. Below is an example of finished code for a right-angle triangle. Note: The code is broken up into 2 lines only to illustrate it in a single screen shot.


ScratchJr
Tic
8
14
1
1
2
1
1
1
2
1
1
1
2
1
2
1
1
1
2