Transformations	Grades 3 & 4

Coding Guide

We will be building our programs in Scratch. You can access the tool here: scratch.mit.edu

Scratch is a free, block-based coding tool. You do not need an account to access the tool; however, you may want to create accounts so that your students can save and revisit their work.

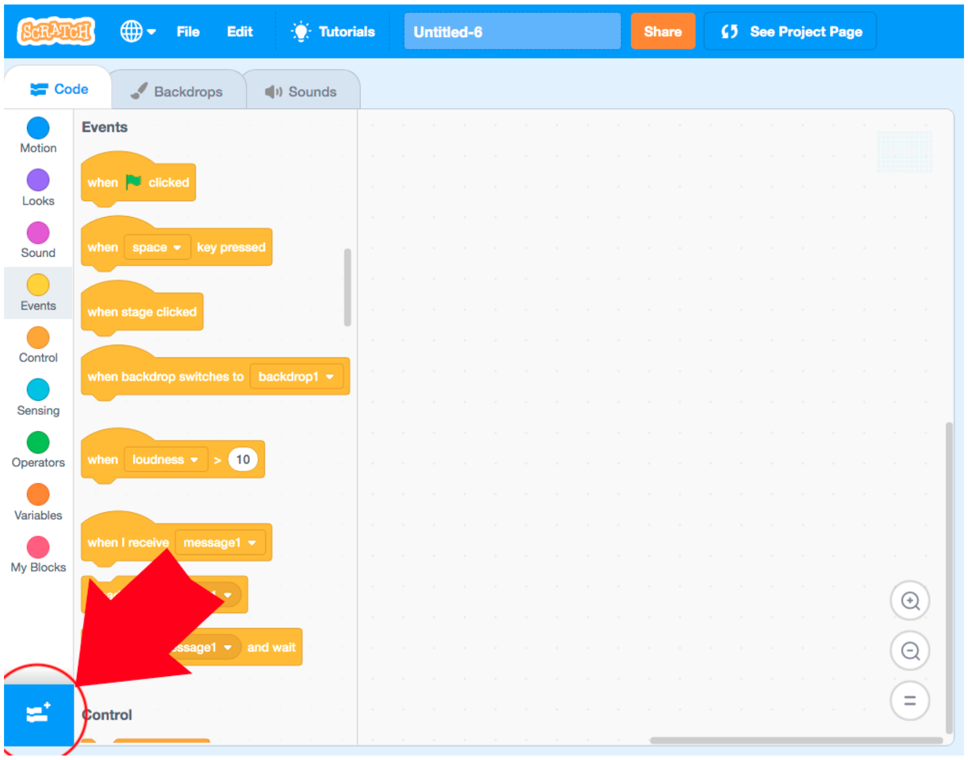
Setting Up Scratch

Before we begin drawing shapes, we are going to set up our Scratch file to optimize it for our workshop.

To open a new program file, click the CREATE button at the top of the Scratch homepage. This will open the Scratch coding application.

Add the Pen Extension

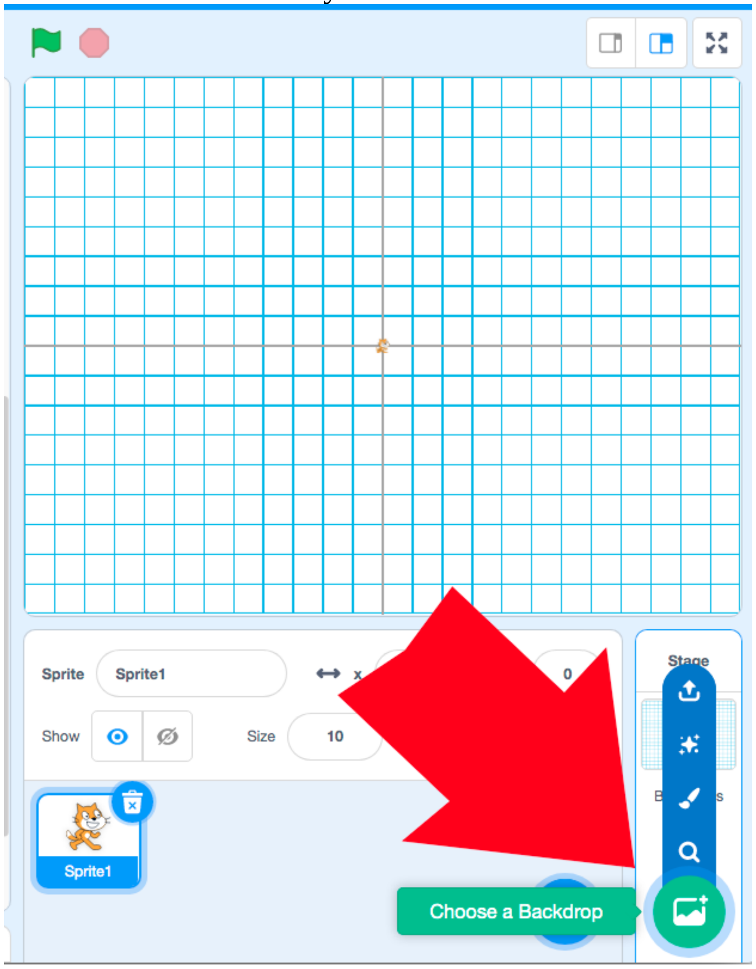
To draw using Scratch, we will need to add an extension called the **Pen Tool**. You can find this extension by clicking the **Add Extension** button in the bottom left corner of the screen.



Choose the Pen Extension. This will add a set of teal **Pen Blocks** to your **Code Tab**

Add a Grid

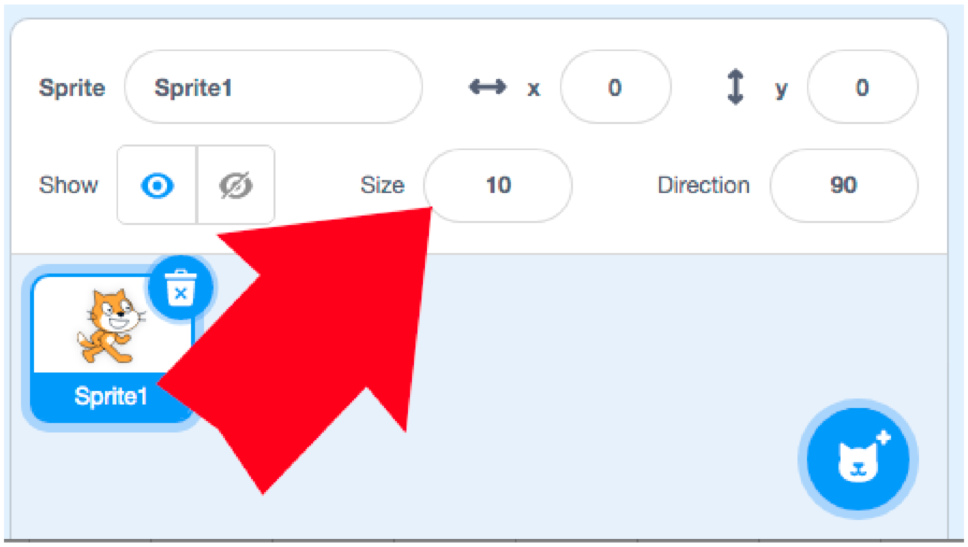
We will be adding a grid background to help us visualize the steps of our transformations. Scratch has pre-set grid backgrounds in their background library. You can find this library by clicking the **Choose a Backdrop** button in the bottom left corner of your screen.



In the Background library search bar, type “grid” to bring up background options. For this activity, I chose the background with gridlines every 20-pixels.

Resize your Sprite

Sprites are programmable elements — the default sprite that you work with in Scratch is the Scratch Cat. We will be using the Scratch Cat for this program, but we will be shrinking it down so that we can better see our pen lines as we draw. Using the Size field in the Sprite panel, resize your sprite to be 10 pixels tall instead of 100 pixels. Make sure that the Scratch Cat sprite (labeled Sprite1) is selected when you make this change.



Now we're ready to start drawing shapes.

Activity 1: Drawing Shapes with Algorithms

IMPORTANT NOTE: For all of the activities in this lesson, it's important to note that the gridlines on our background appear every 20 pixels, and 1 step for our sprite is equal to 1 pixel. So, to move one square on our gride, our sprite will have to be programmed to move 20 steps.

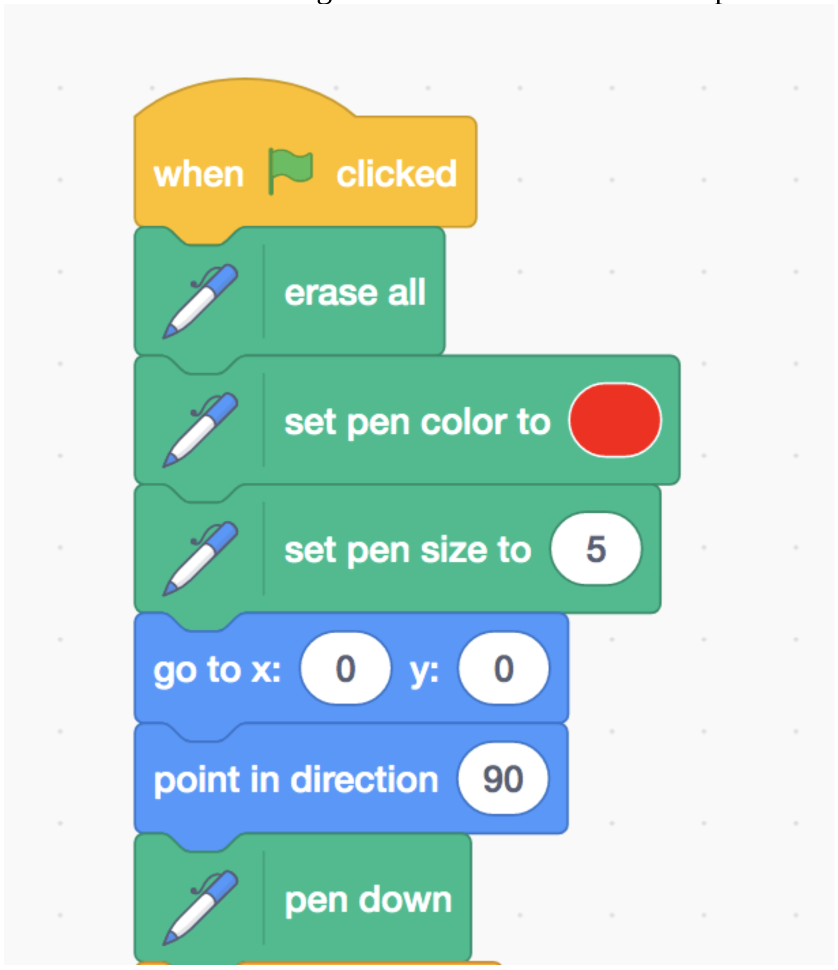
Likewise, the x-y coordinate system on this grid is measured in terms of pixels, so for the (x, y) coordinate of (0, 0) would be the centre of the screen, and the coordinate (0, 20) would place the sprite on the first gridline above (0, 0) on the y-axis — a line that we might otherwise interpret at (0, 1) on paper.

Set up the Pen Tool

- Under the light orange **Events** tab, select the **When green flag clicked** block. This is the block that we will use to trigger the start of our program. Drag the block onto the coding canvas.
- Under the teal **Pen** menu, choose the **erase all** block and connect it directly below the **When green flag clicked** block. This will ensure that we have a fresh slate every time we relaunch our program.
- Under the teal **Pen** menu, choose the **set pen color to** block and add it below the **erase all** block. This block will allow us to set the colour with which our pen draws. Click on the coloured oval on the block to select your colour. In the examples, we've used red, which is very visible against the grid background.
- Under the teal **Pen** menu, choose the **set pen size to** block and add it below the **set pen color to** block. Change the number value on the block to 5. This will make our pen draw with a heavier line that is easier to see.

- Under the blue **Motion** menu, choose the **go to x: 0 y: 0** block and add it below the **set pen size to** block. This block sets the coordinates for where the program begins drawing. (0,0) — also known as the origin — is at the very centre of our screen.
- Under the blue **Motion** menu, choose the **point in direction 90** block and add it below the **go to x: 0 y:0** block. This block controls the direction in which our sprite is facing instead of choosing a random direction (we can leave it set to 90 degrees).
- Under the teal **Pen** menu, choose the **pen down** block and add it below the **point in direction 90** block. This block tells our program to "put pen to paper" and start drawing.

Your program should now look like the image below. We will use this setup to draw our shapes.


```
when [flag] clicked
erase all
set pen color to (red)
set pen size to 5
go to x: 0 y: 0
point in direction 90
pen down
```

Draw a Square

To draw any basic geometric shape, we will need to decide three things:
1. How many sides the shape has
2. The angle of the turn our sprite will have to make at each corner in degrees.
3. The length of each side in terms of "steps" (pixels)

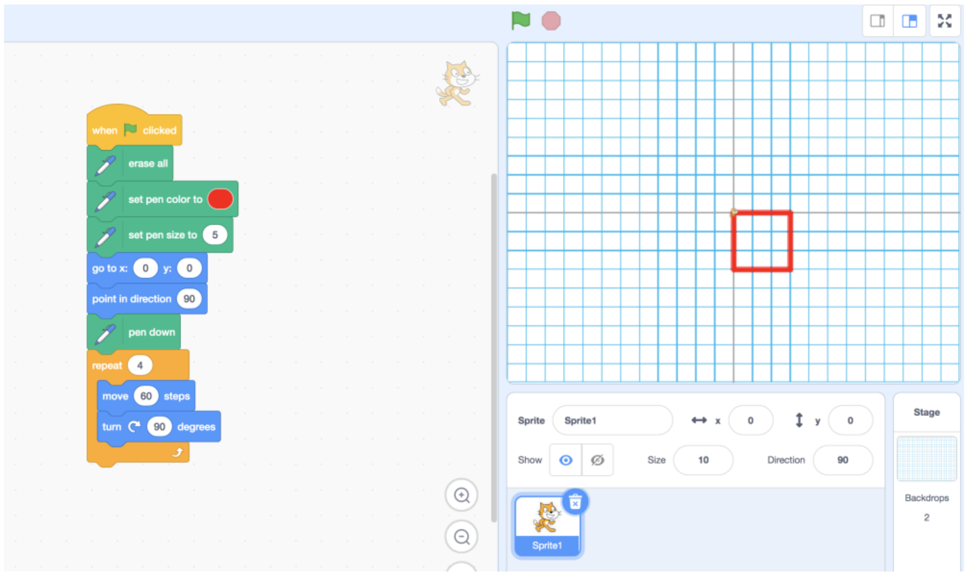
For example, to draw a square in Scratch:

1. A square has 4 sides
2. The angle of all corners is 90˚ (or a right angle)
3. Each side has the same length. Since each square on our grid background is 20 px tall and 20 px wide, let's make each side of our square 60 px, or 60 steps, long. This will make each side of the shape the length of 3 squares on our grid.

To represent this in code:

- In the orange **Control** tab, choose the **repeat 10** block. This block is used to create what is known as a **loop**. A loop makes code more efficient by given us a way to easily repeat a set of instructions. Any code that we put inside the mouth-like gap on this block will be repeated for the number of times that we indicate.
  - Since we are drawing a square, we will want to repeat our code four times, once for each side of the square. Change the value on the block from 10 to 4.
  - Under the **motion** menu, choose the **move 10 steps** block and nest it inside the **repeat 4** block. Since we decided that each side will be 60 px, or 60 steps, long, change the value on the block from 10 to 60.
  - Under the **motion** menu, choose the **turn clockwise 15 degrees** block and nest it inside the **repeat 4** block, directly under the **move 60 steps** block

Your program should look like this:



```
when 🏳 clicked
erase all
set pen color to 🟥
set pen size to 5
go to x: 0 y: 0
point in direction 90
pen down
repeat 4
    move 60 steps
    turn ↻ 90 degrees
```

Sprite: Sprite1   x: 0   y: 0
Show   Size: 10   Direction: 90

Draw a Rectangle

To draw a rectangle, establish the rules for the shape:

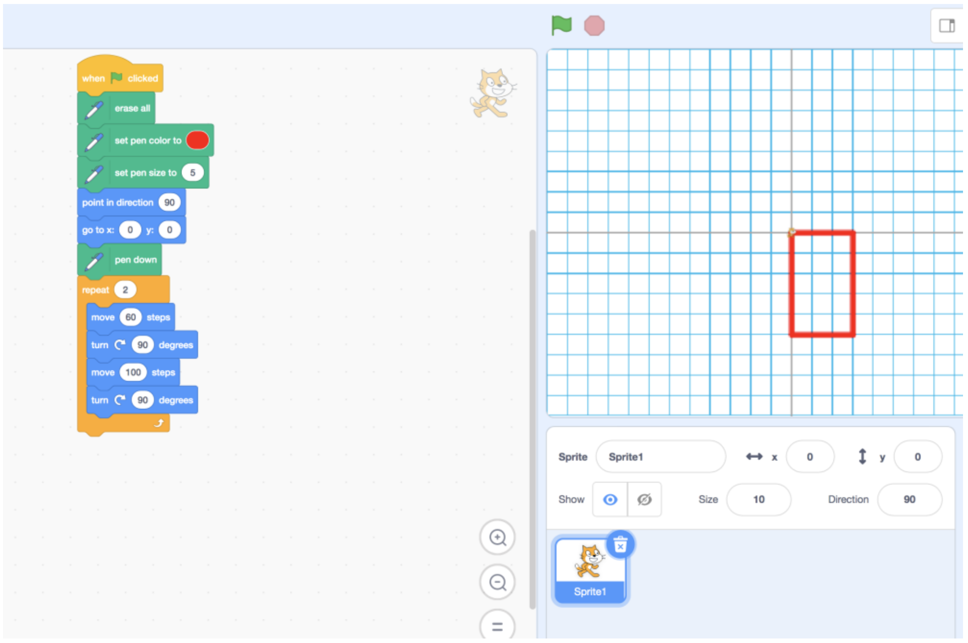
1. A rectangle has 4 sides
2. The angle of all corners is 90˚ (or a right angle)
3. A rectangle has two shorter sides (of equal length) and two longer sides (of equal length). The two shorter sides are opposite and parallel to each other, and the two longer sides are opposite

and parallel to each other. For this example, we'll make our shorter sides 60 px (or 60 steps) long, and our longer sides 100 px (or 100 steps) long.

Building off our existing code for a square:

- Under the blue **Motion** menu, choose the **move 10 steps** block and nest it inside the **repeat 4** block under the **turn clockwise 90 degrees** block. Change the value from 10 steps to 100 steps. This block will draw the longer sides of our rectangle (the existing **move 60 steps** block will draw the shorter sides).
- Under the blue **Motion** menu, choose the **turn clockwise 15 degrees** block and nest it inside the **repeat 4** block under the **move 100 steps** block.
- Since we only need to repeat the steps inside our loop twice (instead of four times), change the value of the **repeat** block to 2.

Your code should look like this:



```
when 🏴 clicked
erase all
set pen color to 🟥
set pen size to 5
point in direction 90
go to x: 0 y: 0
pen down
repeat 2
    move 60 steps
    turn ↻ 90 degrees
    move 100 steps
    turn ↻ 90 degrees
```

Sprite: Sprite1 · x: 0 · y: 0 · Size: 10 · Direction: 90

Draw a Triangle

To draw a triangle, establish the rules for the shape:

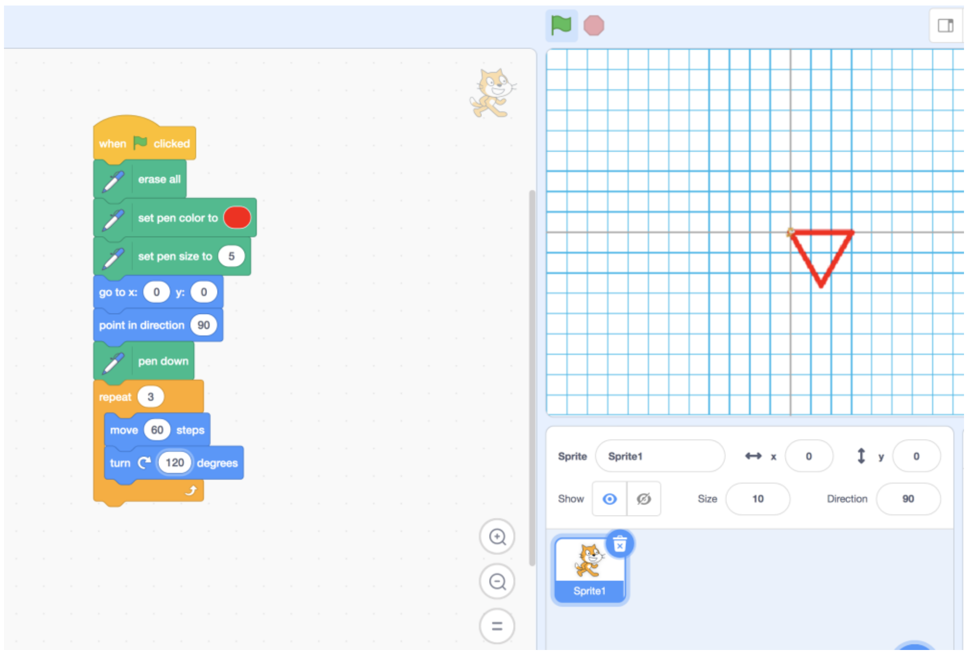
1. A triangle has 3 sides
2. The external angle of all corners is 120˚ (360˚ divided by 3)
3. We're drawing an equilateral triangle, so all three sides are the same length. For this example, we'll make all three sides 60 px, or 60 steps, long.

Because all sides and angles of our triangle are the same, we will only need two blocks inside our **repeat** block, one **repeat 10 steps** block, and one **turn clockwise 15 degrees** block.

- Change the value of the **repeat 10 steps** block to 60 steps

- Change the value of the **turn clockwise 15 degrees** block to 120 degrees
- Change the value of the repeat block to 3

Your program should look like this:



```
when [flag] clicked
erase all
set pen color to [red]
set pen size to 5
go to x: 0 y: 0
point in direction 90
pen down
repeat 3
    move 60 steps
    turn ⟳ 120 degrees
```

Sprite: Sprite1 x: 0 y: 0
Show Size: 10 Direction: 90

1. Opportunity for extension: encourage your students to spend some time trying to make different geometric polygons by altering the values for the **repeat** block, **move ? steps** block, and the **turn clockwise ? degrees** block. Examples of shapes that they can try:

- Pentagon (repeat 5 times; move 100 steps; turn 72 degrees)
- Hexagon (repeat 6 times; move 100 steps; turn 60 degrees)
- Octagon (repeat 8 times; move 100 steps; turn 45 degrees)

Activity 2: Applying Translations

Now that we are comfortable drawing shapes, we can begin experimenting with transformations.

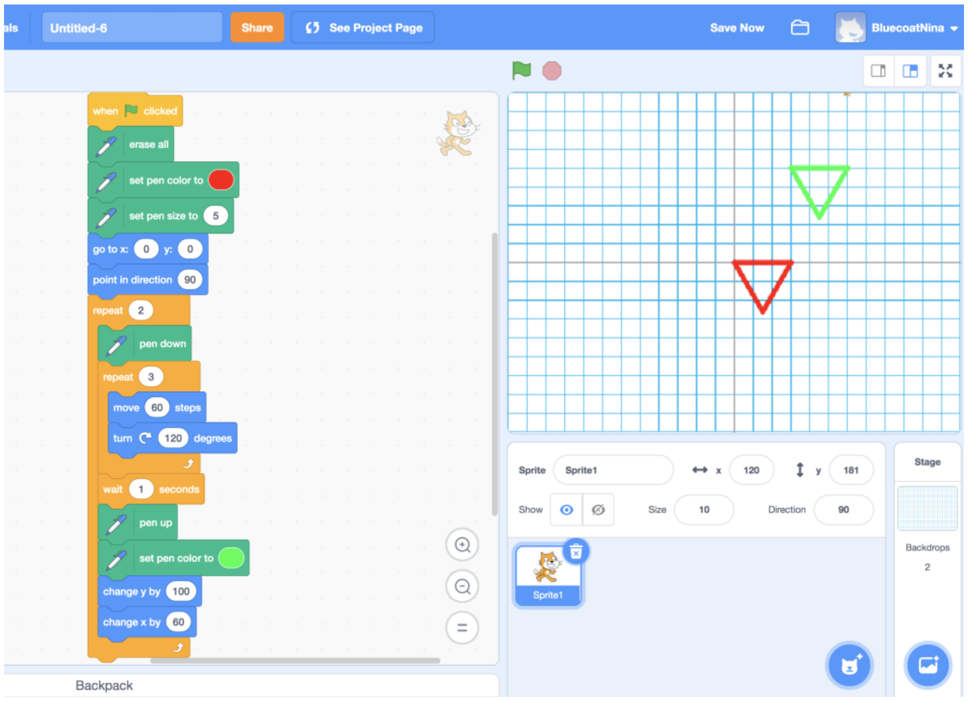
Translations are the simplest transformations to add to our code. They involve moving our shapes up and down or left and right. The shape's direction, size, or arrangement do not change as we move the shape from one location to another.

Coordinates allow us to communicate precisely about the translations that we apply to our shapes. We can change the vertical position of the starting position of our shape by changing its y-value, and we can change the horizontal position of the shape by changing its x-value (remember that our original shape was drawn with the starting position of y = 0 and x = 0 or with the coordinate (0, 0)).

We can build our translations program by adding to our base program for drawing a triangle.

- Under the yellow **Control** menu, choose the **wait 1 second** block and add it to the program.
- Under the teal **Pen** menu, choose the **pen up** block and add it below the **wait 1 second** block. This will lift the pen off the paper while we make the movements of our translation. (If you want the pen to draw lines between the first and second shapes, you can skip this step)
- Under the teal **Pen** menu, choose the **set pen color to** button. Use the coloured oval to select a new colour. This will help us visualize the differences between our pre-translation and post-translation shapes.
- Now we need to describe changes in x- and y-positions to translate the shape.
  - Under the **Motion** menu, choose the **change y by 10** block and place it below the **set pen color to** block.
  - Under the **Motion** menu, choose the **change x by 10** block and place it below the **set pen color to** block.
  - Any values that we choose for these two new blocks will dictate the new position of our shape. For this example, we changed the **change y by 10** block to 100 and the **change x by 10** block to 60.
- Finally, we need go code our program to re-draw our shape in its new position. To do this, we will add a loop.
  - In the orange **Control** menu, chose the **repeat 10** block. Disconnect the portion of the existing code that we'd like to repeat by clicking the **pen down** block and dragging to separate it. Move this piece of code into the new **repeat 10** block. Change the value of the **repeat** block to 2. Connect the **Repeat 2** block to the program directly below the **point in direction 90** block.

Your program should look like this:



```
when 🏴 clicked
erase all
set pen color to 🟥
set pen size to 5
go to x: 0 y: 0
point in direction 90
repeat 2
    pen down
    repeat 3
        move 60 steps
        turn ↻ 120 degrees
    wait 1 seconds
    pen up
    set pen color to 🟩
    change y by 100
    change x by 60
```

Encourage your students to test the program with different x- and y- values to see the resulting translations.

Activity 3: Applying Rotations

A **Rotation** is a type of transformation that takes a shape and rotates it around a given point. The shape does not change size and is not distorted in any way (e.g., an equilateral triangle that is rotated will still be an equilateral triangle of the same size) — it will just be pointed in a new direction. We describe how far a shape has rotated around a given point in terms of degrees, where 90˚ is a quarter-turn, 180˚ is a half-turn, and 360˚ is a full rotation (or a full circle).

We are going to build a new program that rotations a rectangle around the point (0, 0). We can use any shape, but the rectangle makes the result of the reflection very apparent.

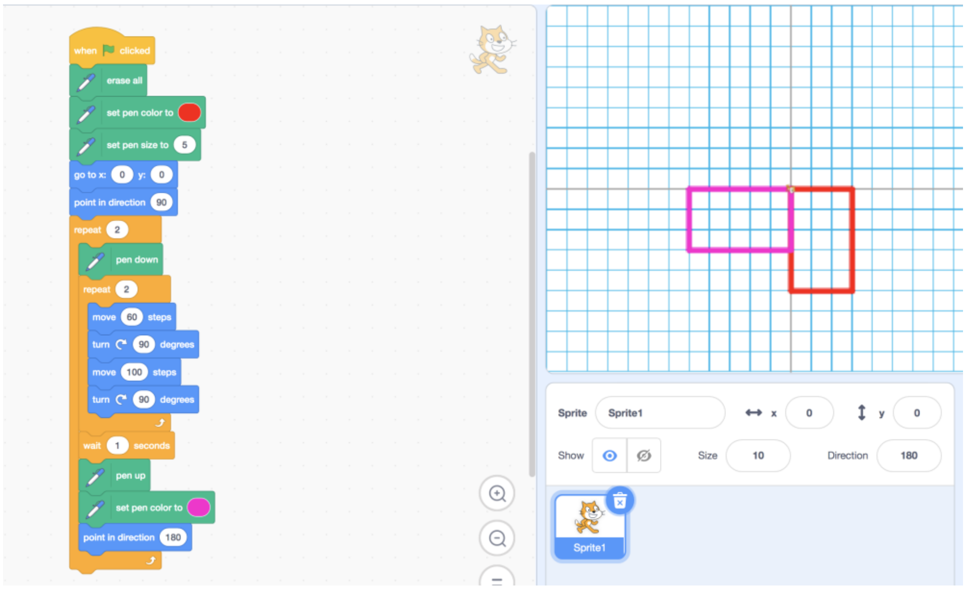
First let's alter the translations program so that we're drawing a rectangle instead of a triangle:

- Under the blue **Motion** menu, choose the **move 10 steps** block and nest it inside the **repeat 3** block under the **turn clockwise 120 degrees** block. Change the value from 10 steps to 100 steps. Change the values on the existing **turn clockwise 120 degrees** block to 90 degrees.
- Under the blue **Motion** menu, choose the **turn clockwise 15 degrees** block and nest it inside the **repeat 4** block under the **move 100 steps** block.
- Since we only need to repeat the steps inside our loop twice (instead of four times), change the value of the **repeat** block to 2.

Now, let’s alter the transformation section of our program.

- The **wait 1 second** block, the **pen up** block, and the **set pen color to** blocks do not change.
- Remove the blue **change y by** and **change x by** blocks by dragging them back into the blocks library panel.
- Under the blue **Motion** tab, find the **point to direction** block and place it under **set pen color to** block for the rotated rectangle. Our first rectangle was drawn with the sprite pointed in the direction 90˚. In this example, we set the new direction to 180˚.

The program should look like this:



```
when 🏴 clicked
erase all
set pen color to ●
set pen size to 5
go to x: 0 y: 0
point in direction 90
repeat 2
    pen down
    repeat 2
        move 60 steps
        turn ↻ 90 degrees
        move 100 steps
        turn ↻ 90 degrees
    wait 1 seconds
    pen up
    set pen color to ●
    point in direction 180
```

Encourage your students to test the program with different degrees of rotation in the **point in direction** block.

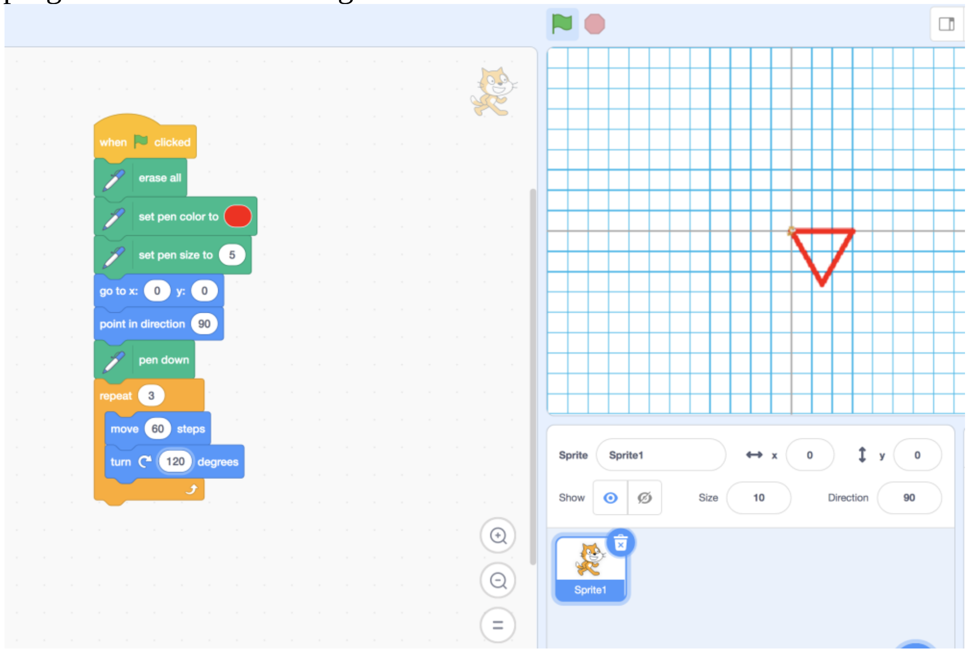
Activity 4: Applying Reflections

A **reflection** is a transformation that acts like a mirror. The shape that is being reflected does not change size and is not distorted in any way. Instead, the shape and all its points are flipped to the opposite side of a defined line of reflection — like how when you stand in front of a mirror and lift your right hand, your reflection will show a hand lifted on the same side as your lifted hand (but if it were a real person and not a reflected image, this would be the person’s left hand).

We are going to build a program that reflects a triangle vertically. The line of reflection will be across the y = 0 line, which means that one of the sides of the reflected triangle will overlap perfectly with one of the sides of the first triangle.

To create a reflection, we must redraw our triangle as if we are drawing it in a mirror (Note: because we are working with a shape with equal sides, it’s tempting to create reflections in Scratch in the same way that we performed rotations in the last activity (by changing the values in the **point in direction** block). While the end result will look like a reflection, this would only would with equilateral shapes, and not with shapes with different side lengths like rectangles or irregular triangles).

Start with our program to draw one triangle:



```
when 🏁 clicked
erase all
set pen color to ●
set pen size to 5
go to x: 0 y: 0
point in direction 90
pen down
repeat 3
    move 60 steps
    turn ⟳ 120 degrees
```

Sprite: Sprite1 x: 0 y: 0
Show Size: 10 Direction: 90

- In the orange **Control** menu, choose a **wait 1 second** block and add it to the bottom of the program (outside of the loop).

- In the teal **Pen** menu, choose the **pen up** block, and add it below the **wait 1 second** block.

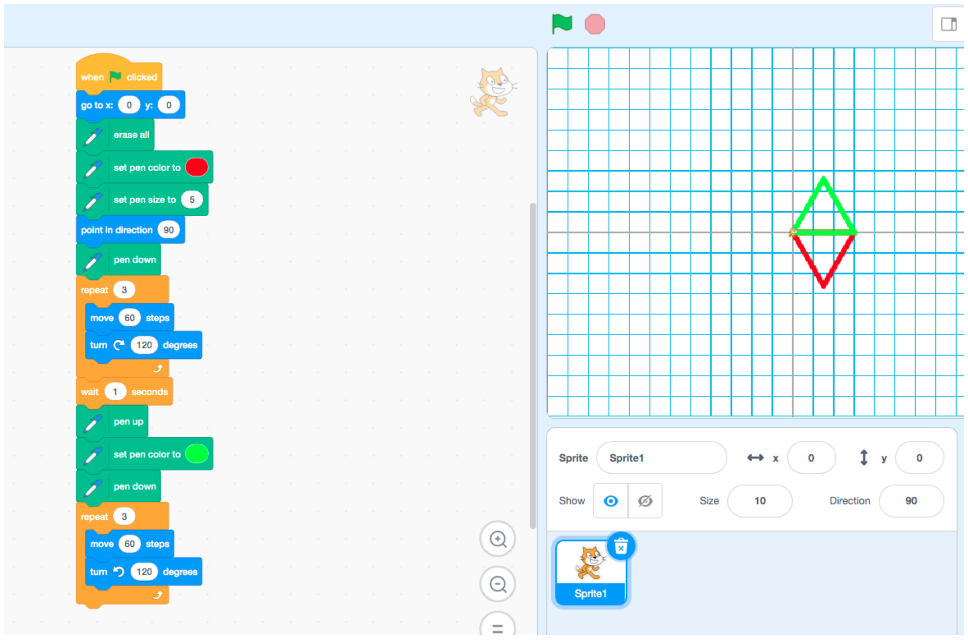
- In the teal **Pen** menu, choose the **set pen color to** block and add it below the **pen up** block. Click the coloured oval to select a new colour for your reflected triangle.
-
- In the teal **Pen** menu, choose the **pen down** block and add it below the **set pen color to** block.

Now, we need to program our drawing instructions for our reflection. It will be just like our instruction for drawing our first triangle, except we will be having our sprite turn counterclockwise instead of clockwise.

- In the orange **Control** menu, choose the **repeat 10** loop block and add it below the **pen down** block. Change the value of this block to 3.
- In the blue **Motion** menu, choose the **move 10 steps** block and nest it inside the new **repeat 3** block. Change the value of this block to 60 steps.

- In the blue **Motion** menu, choose the **turn counterclockwise 15 degrees** block and nest it inside the new **repeat 3** block, beneath the **move 60 steps** block. Change the value of this block to 120 degrees.

The program should look like this:



```
when 🏴 clicked
go to x: 0 y: 0
erase all
set pen color to 🟥
set pen size to 5
point in direction 90
pen down
repeat 3
    move 60 steps
    turn ↻ 120 degrees
wait 1 seconds
pen up
set pen color to 🟩
pen down
repeat 3
    move 60 steps
    turn ↺ 120 degrees
```

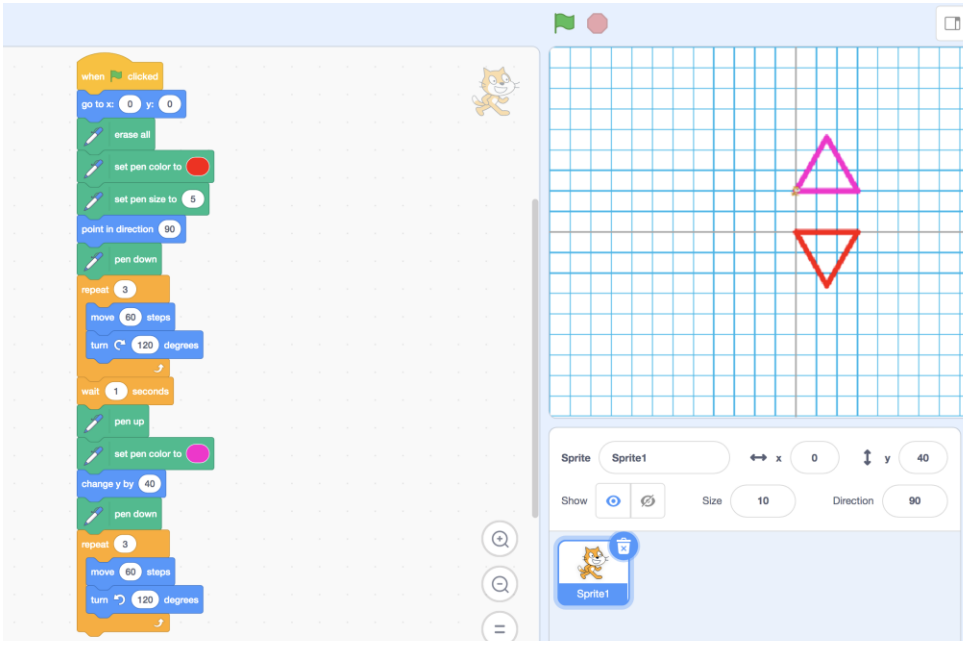
Sprite: Sprite1 x: 0 y: 0
Show Size: 10 Direction: 90

Note that the **go to x: 0 y: 0** block has moved to the top of our program, right under the start block. This is not a necessary step, but if you find that your program is drawing extra lines between your transformations, then moving this block (as pictured) might correct this issue.

If we want to change the line of reflection so that the triangles are separate, we'll have to add a block to shift where our reflected triangle is drawn.

- In the blue **Motion** tab, choose the **change y by 10** block. Place this block between the **set pen color to** block and **pen down** block for our reflected triangle. Putting the block here means that we are moving the position of our reflected triangle while the pen is "lifted" from the page and not drawing.
- Change the value of this block to 60. This value creates a line of reflection that looks like y = 1 on our grid (actually, it's y = 20, if you recall that every gridline is at a 20 px mark).

The program should look like this:



```
when 🏴 clicked
go to x: 0 y: 0
erase all
set pen color to 🟥
set pen size to 5
point in direction 90
pen down
repeat 3
    move 60 steps
    turn ↻ 120 degrees
wait 1 seconds
pen up
set pen color to ●
change y by 40
pen down
repeat 3
    move 60 steps
    turn ↺ 120 degrees
```

Sprite: Sprite1 ↔ x: 0 ↕ y: 40
Show Size: 10 Direction: 90

Opportunity for Extension: In both of our examples, we've reflected the triangle vertically. How would you reflect a triangle horizontally (across a vertical line of reflection such as x = 0)? How would you perform a reflection across a diagonal line of reflection?

Additional Opportunity for Extension: If your students are comfortable with the different transformation programs, you can challenge them to remix the different programs that we've built in this lesson to perform more than one translation on a shape. For example, students can be challenged to translate a triangle from (0, 0) to (20, 120) and then rotate it 120˚ around that point (20,120).