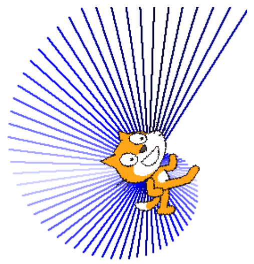
Scratch Art 02

Inspired by the works of TurtleArt - an extension of Logo and the work done by Seymour Papert - we have created a series of activities focused on the creation of art using a simple programming structure. Scratch is a simplified graphical programming environment that mimics a lot of the features of TurtleArt. We can draw lines, change colors, shades, and pen widths -- all to create great geometric art.



SCRATCH
SKILL REQUIREMENTS
DIFFICULTY 1-5

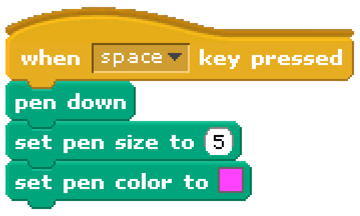
ELECTRICAL PROTOTYPING	○○○○○
ROBOTICS	○○○○○
SOLDERING	○○○○○
PROGRAMMING	●○○○○
DIY	○○○○○

MATERIALS LIST

- Computer
- Scratch
- PicoBoard (optional)

STEP 1: Setting up your palette color and pen size.

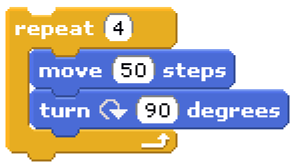
Scratch has a neat feature of a pen that follows the Sprite as it moves across the screen. We can modify the pen size, color, and position (up or down). When we start, we want to make sure that the pen is down, we set the pen size, and the color to something we like.



```
when [space] key pressed
pen down
set pen size to (5)
set pen color to [■]
```

STEP 2: Control Loops

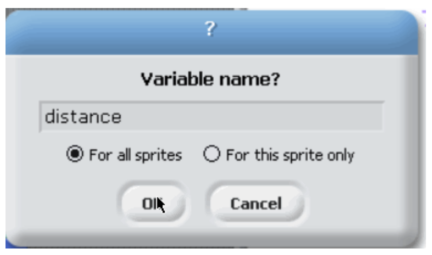
In programming we try to be as efficient as possible. If we wanted to draw a box, we could have 8 lines of code that: move forward, turn 90 degrees, move forward, turn 90 degrees, move forward, turn 90 degrees, and move forward, turn 90 degrees… That gets tedious and messy. Instead, we are going to introduce the repeat block. Can you tell what this code is going to do?



```
repeat (4)
    move (50) steps
    turn ↻ (90) degrees
```

STEP 3: Run →

In this example, we used the **when [space] key pressed** event block. So, press the space key and see what happens? Change the color, move the sprite, and press the space key again.



?

Variable name?

distance

(●) For all sprites ( ) For this sprite only

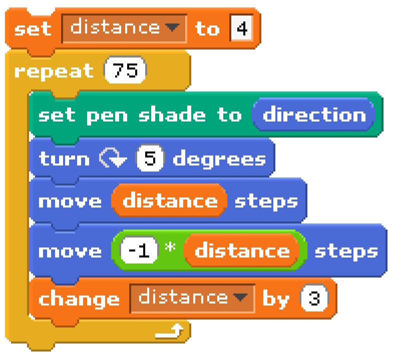
OK Cancel

STEP 3: Let's draw!

You will notice a few new blocks that show up now under the **Variables palette**. We want the block that says **set (distance) to [ ].** Find that block and the others here to create this sequence. I start my first line off with a distance of 4.

There is now a **(distance)** variable block that you can use in place of a number. We use this to move the sprite move out a distance and then return back by moving **[(-1) * distance].** You'll find that green multiplication block under the **Operators palette**.

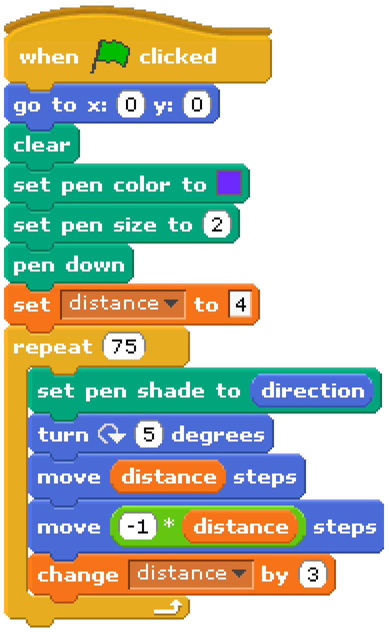
There is another value block that's called **(direction)** that you'll find under the **Motion palette**. This will have a value equal to the direction that your sprite is pointed in. So, each time we turn, the value of direction should change. We use this to set the **shade** of our pen color.



```
set [distance] to (4)
repeat (75)
    set pen shade to (direction)
    turn ↻ (5) degrees
    move (distance) steps
    move ((-1) * (distance)) steps
    change [distance] by (3)
```

STEP 4: Click, Modify, and Repeat

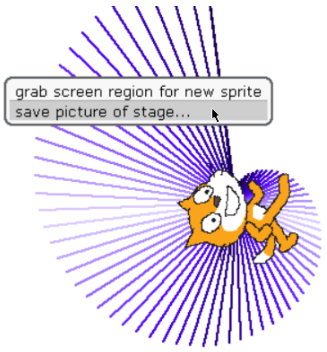
Click the Green flag to run your design. Now, play around with different colors, turning amounts, and step sizes. Remove the [clear} block at the beginning if you want to create multiple patterns on the same screen. Here is the complete program, in case you need a little help.



```
when [flag] clicked
go to x: 0 y: 0
clear
set pen color to [ ]
set pen size to 2
pen down
set distance to 4
repeat 75
    set pen shade to direction
    turn ↻ 5 degrees
    move distance steps
    move -1 * distance steps
    change distance by 3
```

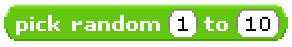
STEP 5: Save / Capture

Right click on the stage and select "save picture of stage…" This will prompt you to save a copy of the background as a GIF file.



TAKING IT FURTHER

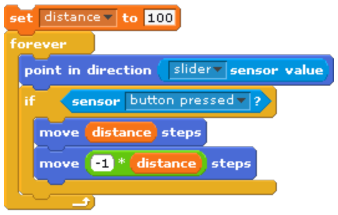
- Use random block to start the sprite at a random place on the screen or start with a random color. Colors are specified on a range of 0 - 200.



```
pick random 1 to 10
```

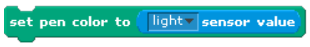
- Do you have a PicoBoard? You can use the button, slider, and the light sensor with this sketch to create some fun drawings. Try this out. Push the button on the PicoBoard to draw a line. Move the slider to change the direction. You will notice that the sprite doesn't rotate all the way around a circle. This is because the slider only goes from 0 - 100.

  Hint: You'll need a multiply operator block. What can I multiply 100 by to get a full circle (360 °)



```
set distance to 100
forever
    point in direction slider sensor value
    if sensor button pressed ?
        move distance steps
        move -1 * distance steps
```

- Other fun things? How about setting the pen color to one of the sensor values like the light? Have your partner move his/her hand above the light sensor as your push the button on the PicoBoard and move the slider around!



```
set pen color to light sensor value
```