**Pythagoras in Scratch - Observation sheet**

https://scratch.mit.edu/projects/317001702

The backdrop for this project is the Xy 20-pixel vector grid which you can get in the Backdrop library of Scratch2 and Scratch3. I can imagine this project (without the need to understand the code) being helpful for secondary SEN students in understanding Pythagoras. They could draw different right-angled triangles on squared paper and check the squares.

# See how the project code works:

Click the *'See inside'* button to view the sprites and the code. The main sprite is the black **anchor** point. It is a fixed point at x,y **3, -2**. Scratch reads the anchor's position in pixels (or steps) as 60, -40 (its position makes maximum use of the stage space available.) The variable *squareSize* represents 20 pixels on the Scratch stage. When the Green Flag is clicked, a moveable **red sprite** is set to a starting position 4 squares to the left of the anchor. Similarly a **green sprite** is set in position 3 squares directly above the anchor. The user uses **Left/Right** arrows sprites to move the red pen-point (and pen) and **Up/Down** arrows to move the green point. Moving the points changes the length of the red line in relation to the anchor and the **area of its square**. A variable *redLength* stores the length of the line and another *redArea* stores and displays its area. Likewise when the green pen-point is moved there is a change in the *greenLength* and *greenArea* variables. The black anchor point is fixed, but the two changing lines cause a **blue line** to change also. The BLUE side is the HYPOTENUSE (which is always opposite 90°). At start, the right-angled triangle has sides of lengths 3, 4 and 5 (the hypotenuse is always the longest of the three).
**NOTE 1:** $3^2 + 4^2 = 5^2$ (9+16=25). The square root of the Hypotenuse = length of blue side. Note the two whole number lengths of the hypotenuse among the examples shown. **NOTE 2:** Some squares distort at the edges of the stage, otherwise you can continue to get more examples. **NOTE 3:** Almost all the 36 examples here fit without distortion at the edges.

**You can get all of the following examples of Pythagoras by moving the points**

| **the sum of two squares** | Area of Red square | Area of Green square | Area of Blue square on (hypotenuse) | Length of (hypotenuse) |
|---|---|---|---|---|
| **$1^2 + 1^2$** | **1** | **1** | **2** | $\sqrt{2}$ |
| **$1^2 + 2^2$** | **1** | **4** | **5** | $\sqrt{5}$ |
| **$1^2 + 3^2$** | **1** | **9** | **10** | $\sqrt{10}$ |
| **$1^2 + 4^2$** | **1** | **16** | **17** | $\sqrt{17}$ |
| **$1^2 + 5^2$** | **1** | **25** | **26** | $\sqrt{26}$ |
| **$1^2 + 6^2$** | **1** | **36** | **37** | $\sqrt{37}$ |
| **$1^2 + 7^2$** | **1** | **49** | **50** | $\sqrt{50}$ ![1] |
| **$1^2 + 8^2$** | **1** | **64** | **65** | $\sqrt{65}$ ![2] |
| **$1^2 + 9^2$** | **1** | **81** | **82** | $\sqrt{82}$ |
| **$2^2 + 2^2$** | **4** | **4** | **8** | $\sqrt{8}$ |
| **$2^2 + 3^2$** | **4** | **9** | **13** | $\sqrt{13}$ |
| **$2^2 + 4^2$** | **4** | **16** | **20** | $\sqrt{20}$ |
| **$2^2 + 5^2$** | **4** | **25** | **29** | $\sqrt{29}$ |
| **$2^2 + 6^2$** | **4** | **36** | **40** | $\sqrt{40}$ |
| **$2^2 + 7^2$** | **4** | **49** | **53** | $\sqrt{53}$ |
| **$2^2 + 8^2$** | **4** | **64** | **68** | $\sqrt{68}$ |
| **$2^2 + 9^2$** | **4** | **81** | **85** | $\sqrt{85}$ ![3] |
| **$3^2 + 3^2$** | **9** | **9** | **18** | $\sqrt{18}$ |
| **$3^2 + 4^2$** | **9** | **16** | **25** | $\sqrt{25} = 5$ |
| **$3^2 + 5^2$** | **9** | **25** | **34** | $\sqrt{34}$ |
| **$3^2 + 6^2$** | **9** | **36** | **45** | $\sqrt{45}$ |
| **$3^2 + 7^2$** | **9** | **49** | **58** | $\sqrt{58}$ |
| **$3^2 + 8^2$** | **9** | **64** | **73** | $\sqrt{73}$ |
| **$3^2 + 9^2$** | **9** | **81** | **90** | $\sqrt{90}$ |
| **$4^2 + 4^2$** | **16** | **16** | **32** | $\sqrt{32}$ |
| **$4^2 + 5^2$** | **16** | **25** | **41** | $\sqrt{41}$ |
| **$4^2 + 6^2$** | **16** | **36** | **52** | $\sqrt{52}$ |
| **$4^2 + 7^2$** | **16** | **49** | **65** | $\sqrt{65}$ ![2] |
| **$4^2 + 8^2$** | **16** | **64** | **80** | $\sqrt{80}$ |
| **$5^2 + 5^2$** | **25** | **25** | **50** | $\sqrt{50}$ ![1] |
| **$5^2 + 6^2$** | **25** | **36** | **61** | $\sqrt{61}$ |
| **$5^2 + 7^2$** | **25** | **49** | **74** | $\sqrt{74}$ |
| **$6^2 + 6^2$** | **36** | **36** | **72** | $\sqrt{72}$ |
| **$6^2 + 7^2$** | **36** | **49** | **85** | $\sqrt{85}$ ![3] |
| **$6^2 + 8^2$** | **36** | **64** | **100** | $\sqrt{100} = 10$ |
| **$7^2 + 7^2$** | **49** | **49** | **98** | $\sqrt{98}$ |

## **Secondary:** Discover Pythagoras

https://scratch.mit.edu/projects/317001702

*Click 'See inside' to get the full picture of this Scratch project. It has three pen sprites, Red, Green and Blue to draw 3 right-angled triangles. There is also an 'Anchor' sprite, coloured black.*

### 1. Left & Right Arrows (Red)

```
Events   when this sprite clicked
Events   broadcast [left ▾]
```
move left

```
when this sprite clicked
broadcast [right ▾]
```
move right

*Two scripts of code are all that is needed on the arrow sprites. When clicked, the* ***redPoint*** *sprite moves either left or right to increase or decrease the size of the square on the red side of the right-angled triangle.*

### 2. Anchor point (Black)

```
when ⚑ clicked
set x to ((3) * (squareSize))
set y to ((-2) * (squareSize))
```
anchorPoint

***squareSize is 20.***

*This script sets the* ***anchorPoint*** *at 3, -2 on the grid. Scratch 'thinks in pixels' and recognises that as the position 60, -40 on the stage.*

### 3. Four scripts of code on the redPoint pen sprite

*The variable* ***squareSize*** *is fixed at 20 to correlate the grid with stage pixels. The variables* ***redLength*** *and* ***redArea*** *go with the* ***redPoint*** *sprite. Their value varies as the sprite is moved by the arrows.*

redPoint

*The Left and Right arrows send the broadcasts* ***left / right*** *to the Red point.* →

**Script 1 of 4**

```
when I receive [left ▾]    Events
change x by (-20)          Motion
```

**Script 2 of 4**

```
when I receive [right ▾]
change x by (20)
```

**Script 3 of 4 is the main script on the redPoint**

redPoint

```
when ⚑ clicked                                                      Events
point in direction (90)                                             Motion
go to x: ((-1) * (squareSize)) y: ((-2) * (squareSize))             Motion/ Operators/ Variables
set pen size to (3)                                                 Pen Extension
set pen color to (0)                                                Pen Extension
forever                                                             Control
    set [redLength ▾] to ((distance to [anchorPoint ▾]) / (squareSize))   Variables/ Operators/ Sensing/ Variables
    set [redArea ▾] to ((redLength) * (redLength))                  Variables/ Operators/ Variables
    create clone of [myself ▾]                                      Control
```

**Script 4 of 4 tells each clone what to do**

```
when I start as a clone
pen down
go to [anchorPoint ▾]
turn ↻ (90) degrees
move ((redLength) * (squareSize)) steps
turn ↻ (90) degrees
move ((redLength) * (squareSize)) steps
turn ↻ (90) degrees
go to [redShadow ▾]
pen up
wait (0.01) seconds
erase all
delete this clone
```
redPoint

### 4 The Green & Blue sprites

*The Green point follows a similar behaviour to the Red point. It is controlled by clicking Up/ Down arrows which move the drawing point Up and Down. So the code on the Green point is very similar to the code on the Red point.*

*The Blue point behaves quite differently as it is controlled by the movements of the Red and Green points. Blue connects the Red and Green to continuously create a hypotenuse. Blue is all about connecting the Red and Green, and its code differs slightly from them.*

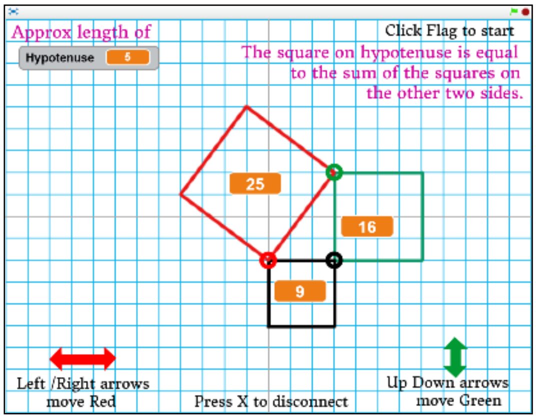
*Study the Red point's code which draws the various squares with a Red outline.*

**PTO ▷**

# Pythagoras



## Red Sprite’s code




```
when flag clicked
go to x: (0 * squareSize) y: (-2 * squareSize)
pen up
set pen size to 3
set pen color to [red]
forever
    point towards pen2
    set redLength to (distance to pen2) / squareSize
    set redArea to blackArea + greenArea
    set Hypotenuse to round (sqrt of redArea * 100) / 100
    create clone of myself
```


```
when left arrow key pressed
change x by -20
```

```
when right arrow key pressed
change x by 20
```

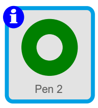
```
when I start as a clone
hide
pen down
go to x: (x position of pen2) y: (y position of pen2)
turn ↺ 90 degrees
move (redLength * squareSize) steps
turn ↺ 90 degrees
move (redLength * squareSize) steps
turn ↺ 90 degrees
go to redShadow
pen up
wait 0.01 secs
clear
delete this clone
```

```
when flag clicked
hide
forever
    set x to (x position of pen1)
    set y to (y position of pen1)
```


```
when flag clicked
hide
point in direction 180
forever
    set x to (x position of pen2)
    set y to (y position of pen2)
```


## Green Sprite’s code

```
when flag clicked
go to x: (3 * squareSize) y: (2 * squareSize)
set pen size to 3
set pen color to [ ]
forever
    set greenLength to (distance to pen3) / squareSize
    set greenArea to greenLength * greenLength
    create clone of myself
```




```
when up arrow key pressed
change y by 20
```

```
when down arrow key pressed
change y by -20
```

```
when I start as a clone
hide
pen down
go to x: (x position of pen3) y: (y position of pen3)
turn ↺ 90 degrees
move (greenLength * squareSize) steps
turn ↺ 90 degrees
move (greenLength * squareSize) steps
turn ↺ 90 degrees
go to greenShadow
pen up
wait 0.01 secs
clear
delete this clone
```

# Pythagoras

Black Sprite's code: The black sprite is stationary