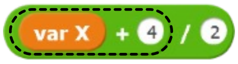
EQUATION OF A LINE

... to Simultaneous Equations

Project objective: When the SPACEBAR is pressed

$$x - 2y = -4$$

$$y = \frac{x+4}{2} \rightarrow$$



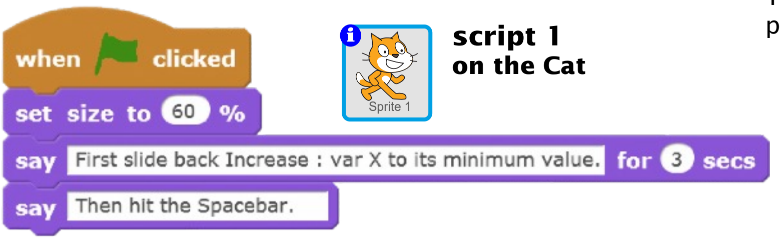
```
((var X) + (4)) / (2)
```

A Setup: Backdrop and Instructions.

1 *Choose backdrop from library* and select *Category* > *Other* > *xy-grid-20px* >**OK**

Alternative: To make Scratch resemble graph paper refer to the book *SCRATCH with Ready-Steady-Code* p.82. With the *squareSize* set to 20 pixels each square on the *10px* grid is equivalent to a half value (0.5).

2 **Code the cat to give instructions.** Drag the cat down to the lower left corner of the stage. Code the cat sprite with these two scripts.



```
when green flag clicked
set size to (60) %
say [First slide back Increase : var X to its minimum value.] for (3) secs
say [Then hit the Spacebar.]
```

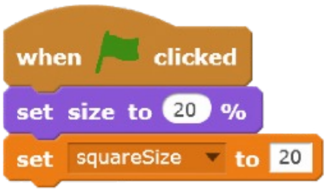
script 1 on the Cat

B Code the Value of X to Increase

1 **Get the X sprite from the Sprite library.**
Go to *Category* > *Things* > *button 5* Click OK
Re-colour green and rename the sprite *Increase*.

2 Make two variables: *squareSize* (for all sprites) and make *var X* (**for this sprite only**). Set the slider Min to -8 and Max to 8.

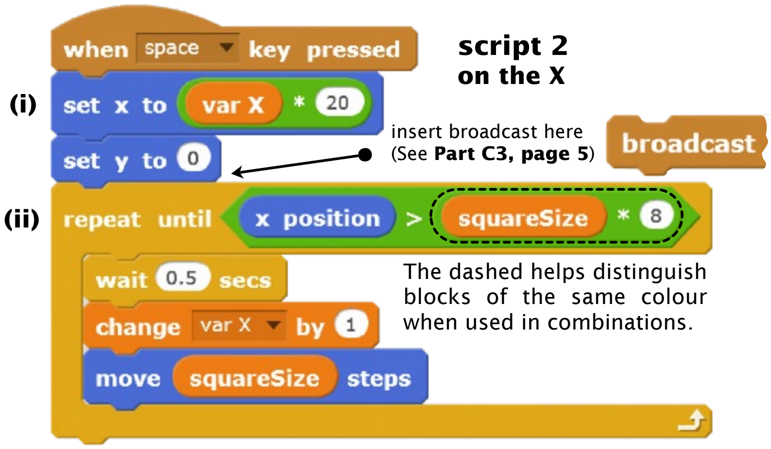
3 Code the *Increase* sprite with these two scripts.



```
when green flag clicked
set size to (20) %
set [squareSize] to (20)
```

script 1 on the X

script 2 on the X



```
when [space] key pressed
(i)  set x to ((var X) * (20))
     set y to (0)
(ii) repeat until <(x position) > ((squareSize) * (8))>
       wait (0.5) secs
       change [var X] by (1)
       move (squareSize) steps
     end
```

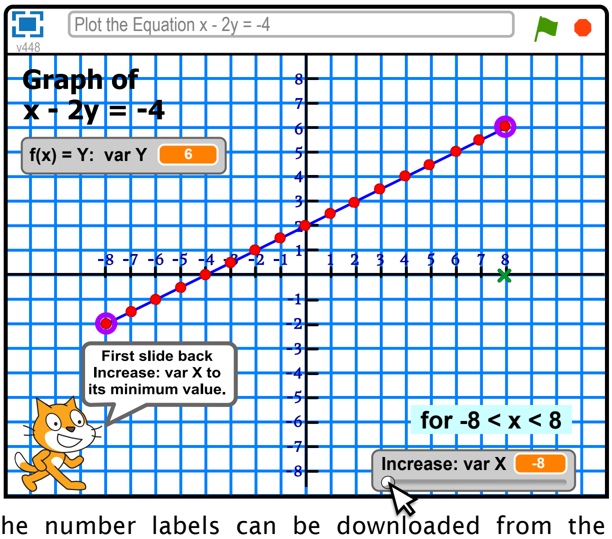
insert broadcast here (See **Part C3, page 5**) → broadcast

The dashed helps distinguish blocks of the same colour when used in combinations.

4 Code a Linear Equation to Draw Dynamically in Scratch

Picture shows the start and end of

scratch.mit.edu/projects/195696458



Plot the Equation x - 2y = -4

Graph of x - 2y = -4

f(x) = Y: var Y 6

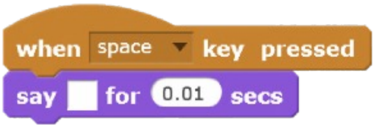
First slide back Increase: var X to its minimum value.

for -8 < x < 8

Increase: var X -8

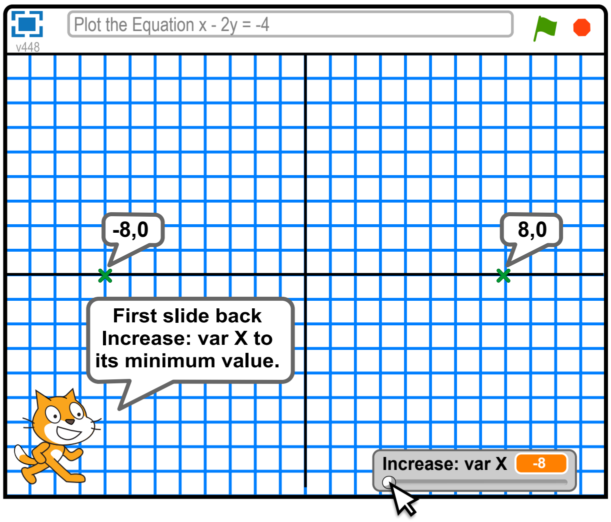
The number labels can be downloaded from the project at **scratch.mit.edu/projects/104133413**

script 2 on the Cat



```
when [space] key pressed
say [ ] for (0.01) secs
```

PART B: scratch.mit.edu/projects/195695421



Plot the Equation x - 2y = -4

-8,0

8,0

First slide back Increase: var X to its minimum value.

Increase: var X -8

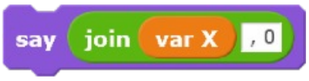
What does each line of code do (script 2)?
When the Spacebar is pressed

(i) The *Increase* sprite is set at -8, 0

(ii) The **repeat** loop runs the following algorithm **until** the x position of the X is at 8 squares.

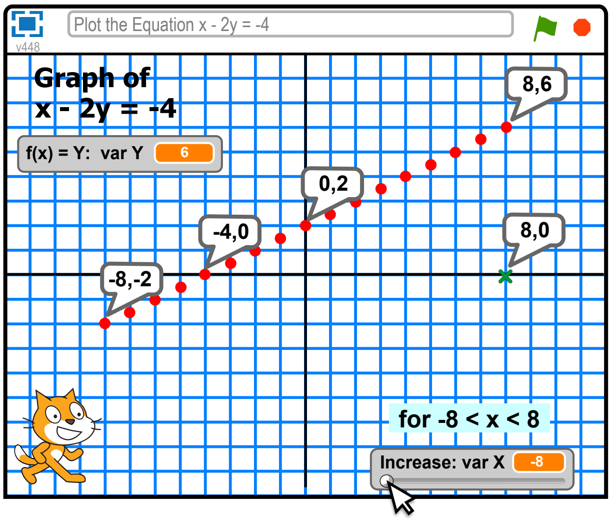
- wait half a second
- move forward one square (20 px) at a time
- change the value of the variable *var X* by 1 in each iteration

Optional: Make the sprite tell its location (as shown in the picture) with this code. Where is it best placed in the script?

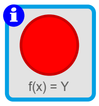


```
say (join (var X) [, 0])
```

C Code the value of Y to change as X increases



1. In the Scratch Sprite library go to *Category* > *Things* > *Ball*
   Click OK
   Rename the sprite *f(x) = Y*
   Re-colour the ball red (or Draw a red dot sprite)

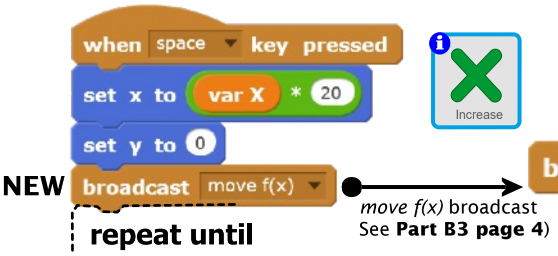


2. Make two variables (**For this sprite only**).
   Give one the name *x*, give the other the name *y*
   Hide *x* but **show** ***y***

hide ☐ x show ☑ y f(x)= Y: y -2

The monitor window tells you that y 'belongs' to **f(x)=Y**

3. Insert a broadcast block into script 2 on the Increase sprite (Part B) *where shown*

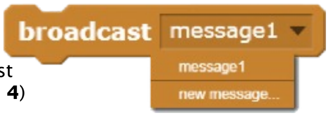


```
when [space] key pressed
set x to ((var X) * 20)
set y to (0)
NEW broadcast [move f(x)]
repeat until
```

move f(x) broadcast
See **Part B3 page 4**)

PART C: scratch.mit.edu/projects/195696075

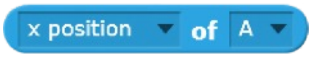
Create a broadcast 'message'



Give the new message the name: *move f(x)*. You want the message to tell the red dot sprite *f(x)=Y* to move!

▷ This message (broadcast) is received by the Event block on top of the script on the red dot sprite f(x)=Y and triggers it into action.

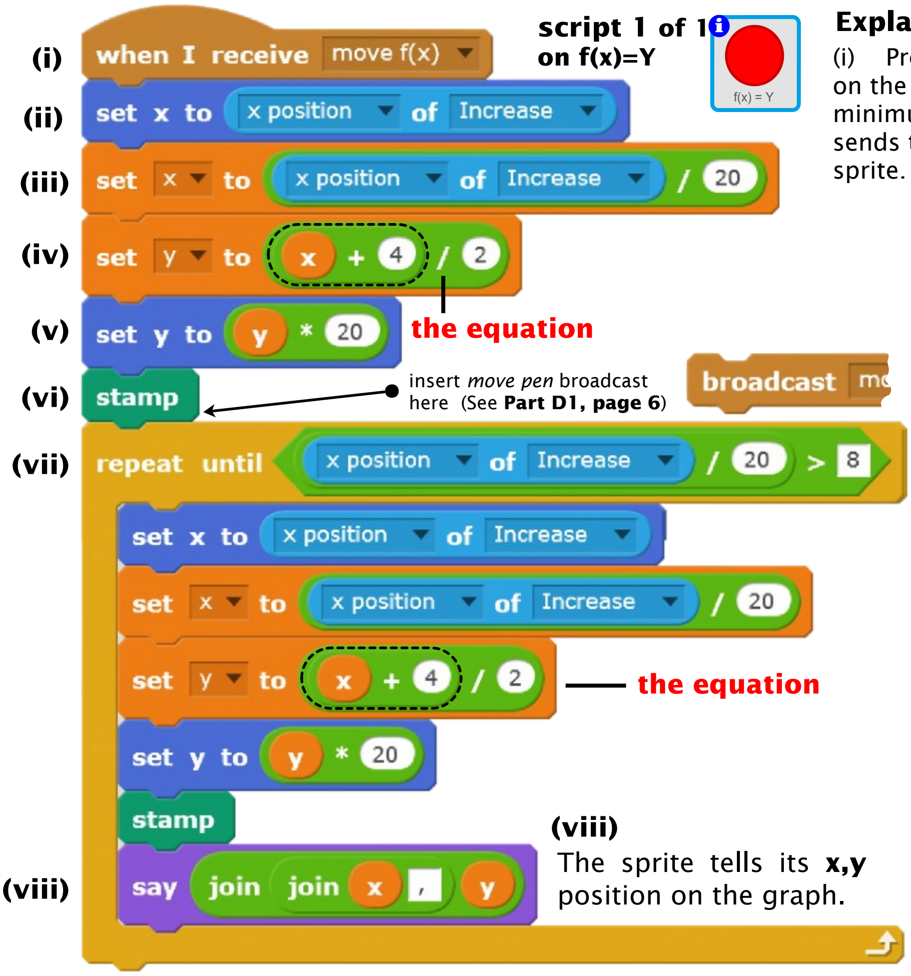
4. The *attribute reporter* (Sensing palette) is essential to understanding the single script on this sprite.



It has two drop-lists

▷ It makes the red dot *f(x)=Y* move from left to right as the x-position of *Increase* moves from left to right. At the same time it changes the value of *f(x)=Y* and we get the dynamic action of the graph.

script 1 of 1
on f(x)=Y



```
(i)     when I receive [move f(x)]
(ii)    set x to (x position of Increase)
(iii)   set [x] to ((x position of Increase) / 20)
(iv)    set [y] to ((x + 4) / 2)
(v)     set y to ((y) * 20)
(vi)    stamp
(vii)   repeat until <((x position of Increase) / 20) > 8>
            set x to (x position of Increase)
            set [x] to ((x position of Increase) / 20)
            set [y] to ((x + 4) / 2)
            set y to ((y) * 20)
            stamp
(viii)      say (join (join (x) [,]) (y))
        end
```

the equation

insert *move pen* broadcast here (See **Part D1, page 6**)

the equation

(viii) The sprite tells its **x,y** position on the graph.

Explanation for each line of code:

(i) Pressing the Spacebar triggers the action on the green *Increase* X sprite and sets its minimum position. The script on the green X sends the new *move f(x)* broadcast to this sprite.

(ii) **The x-position** of this sprite is set to the x-position of the green x. In other words, the x-position of this sprite is set from the slider variable of Increase x - almost at the same instant as the Spacebar is pressed.

(iii) The variables *x* and *var X* are matched with the start x-position value. (You don't need to see *x*).

(iv) The variable *y* gets its value from the value of *x* in the project equation. As the value of *x* changes the direction (slope) of the 'string' of dots follows its pattern.

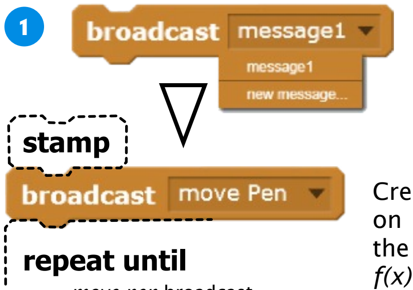
(v) **The y-position** of this sprite is set from variable *y*.

(vi) The first 'stamp' of the red-dot is made with this command before the loop begins to run.

(vii) The loop repeats all of the steps for each value of the Increase x as it moves across the number line.

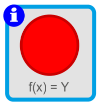
D Code the Pen to Draw the Graph of the Line

1



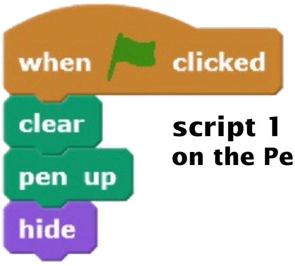
```
broadcast message1 ▾
          message1
          new message...
stamp
broadcast move Pen ▾
repeat until
```

move pen broadcast
See **Part C4, page 5**)



Create a broadcast 'message' on the red dot sprite. Give it the name: *move Pen.* You want *f(x)=Y* to communicate with the Pen sprite to make it move.

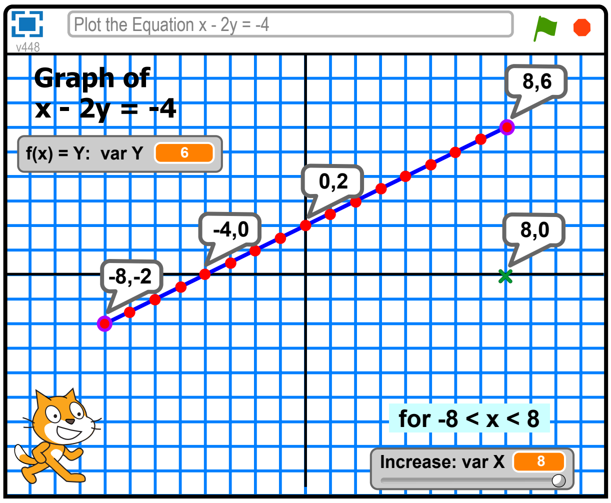
2



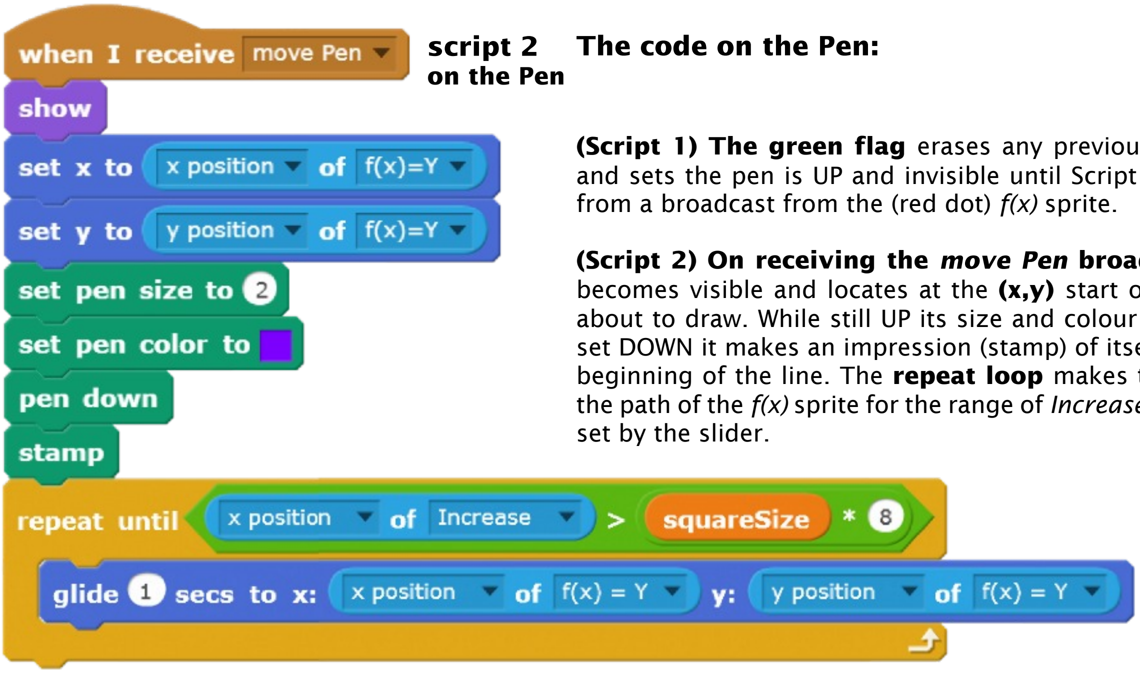
```
when ⚑ clicked
clear
pen up
hide
```

script 1 on the Pen

From the Sprite library, get a sprite to use as a pen (or draw a quick Pen sprite of your own). Here are the two scripts on the Pen sprite.



PART D: scratch.mit.edu/projects/195696458



```
when I receive move Pen ▾
show
set x to (x position ▾ of f(x)=Y ▾)
set y to (y position ▾ of f(x)=Y ▾)
set pen size to 2
set pen color to ■
pen down
stamp
repeat until <(x position ▾ of Increase ▾) > ((squareSize) * 8)>
    glide 1 secs to x: (x position ▾ of f(x) = Y ▾) y: (y position ▾ of f(x) = Y ▾)
```

script 2 on the Pen

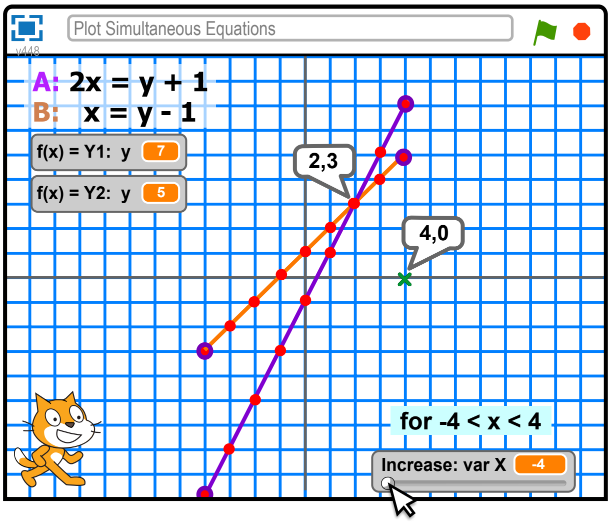
The code on the Pen:

(Script 1) The green flag erases any previously drawn line and sets the pen is UP and invisible until Script 2 is triggered from a broadcast from the (red dot) *f(x)* sprite.

(Script 2) On receiving the *move Pen* broadcast the Pen becomes visible and locates at the **(x,y)** start of the line it is about to draw. While still UP its size and colour are set. When set DOWN it makes an impression (stamp) of itself to mark the beginning of the line. The **repeat loop** makes the pen follow the path of the *f(x)* sprite for the range of *Increase* that has been set by the slider.

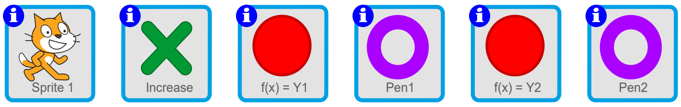
5 Easy Step to Simultaneous Equations

Go to scratch.mit.edu/projects/195706136



***Look Inside* to see the code** See inside

Only one *Increase* sprite to plot the range -4 to 4



There are two equations A (purple) and B (brown) and two solution algorithms in Scratch code

one for f(x)=Y1 **y =** ((2 * x) - 1)

and one for f(x)=Y2 **y =** (x + 1)

As the line A intersects the x-axis at 0.5, the *10px* grid would be a better background to use. See half intervals illustrated for this graph at

scratch.mit.edu/projects/193451663