Scratch Tutorial

Victor Norman

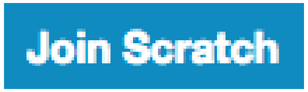
Calvin College

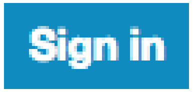
vtn2@calvin.edu

Let’s get started!

- Go to

  http://scratch.mit.edu

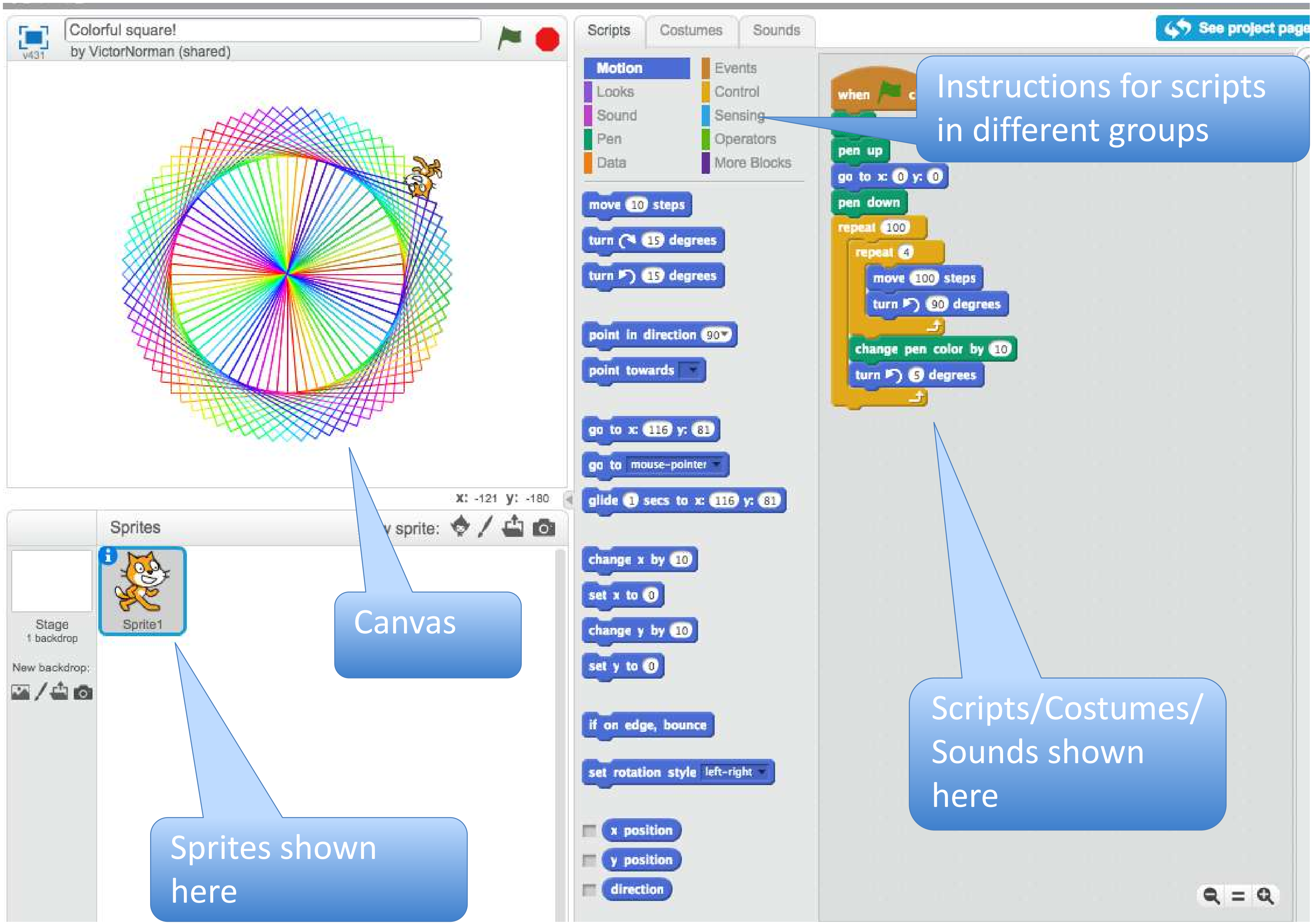
- If you are new to Scratch, click: 

- else click: 

Note: you can use Scratch without logging in, but you can’t see others’ work or save your own work.

Try some stuff

- Click on some projects in Featured Projects
- Click the green flag ⚑ to run the project.
- Click on See inside to see the code.
- Click on various Sprites in lower-left area.
  - In right-side area, you see the Scripts, or Costumes, or Sounds for that sprite.
- Click on the Stage on the left side.
  - In right-side area, you can see Scripts, Backdrops, or Sounds.



```
when 🏴 c
pen up
go to x: 0 y: 0
pen down
repeat 100
    repeat 4
        move 100 steps
        turn ↺ 90 degrees
    change pen color by 10
    turn ↺ 5 degrees
```

Instructions for scripts in different groups

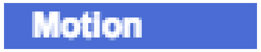
Canvas

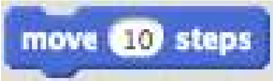
Scripts/Costumes/ Sounds shown here

Sprites shown here

Lesson 1: Basic, basic script

Goal: Make Scratch the cat walk back and forth on the screen.

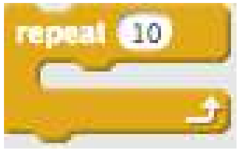
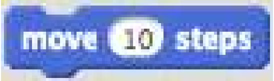
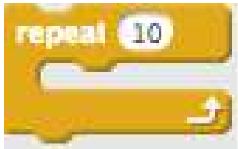
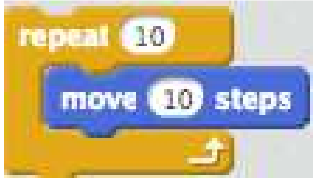
1. File → New
2. Click on 

3. Drag  over to the script area.

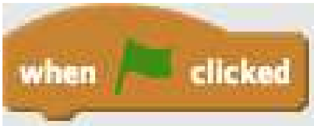
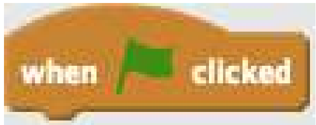
4. Double-click on it.

   - When you double-click on a script, it runs it, once.

Basic, basic script (2)

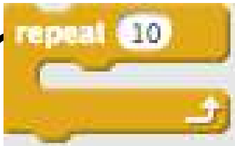
5. Click on , then drag  into script area.
6. Put  “inside” the  loop: 
7. Double-click on that. When cat gets to the right edge, drag it back to left edge with your mouse.

Basic, basic script (3)

8. Click on and drag  into script area. Attach to the top of your script. (Notice that some things can only attach at the top, or bottom.)
9. Click on the ⚑ above the canvas area.
10.  is an "event handler": the script is run when the "green flag clicked event" happens.

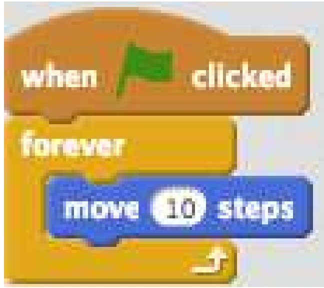
Basic, basic script (4)

11. We want the cat to walk back and forth across the screen, changing direction when it hits the edges.

12. Detach all three instructions in your script so they are separated. Dr back onto the area that shows the commands – this is how you delete something.

```
repeat 10
```

13. Make this instead ------------->

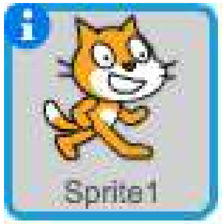


```
when [flag] clicked
forever
    move 10 steps
```

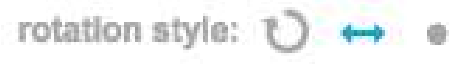
Basic, basic script (5)

13. We need to make the cat reverse direction when it hits the edge. Click on Motion and find if on edge, bounce . Add that to your script in the correct place.

14. The cat flips upside down now. To fix this, click on ℹ in here:

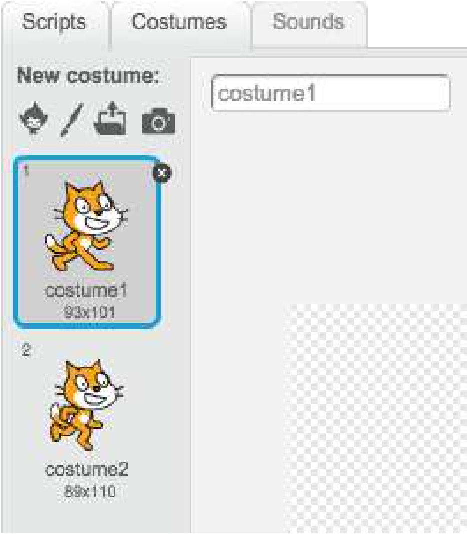


15. Change the sprite’s rotation style to this:

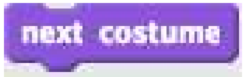

rotation style: ↻ ↔ ●

Basic, basic script (6)

16. Better… but the cat is sliding everywhere. Let’s make it walk. Click on Costumes: Notice that it already has 2 costumes defined for it ----->.

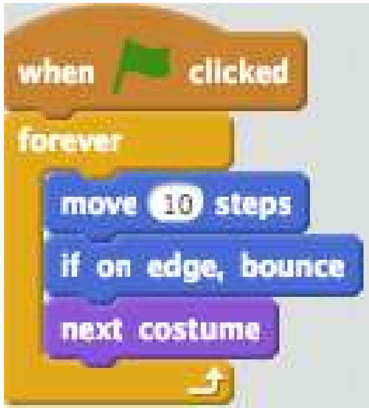


costume1

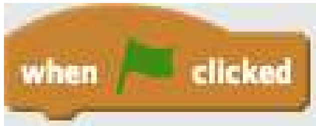
17. Click back on Scripts and click on Looks . In there, you’ll find this:  next costume . Add that to your script in the correct place. Then, try it!

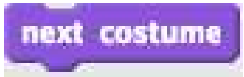
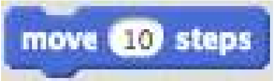
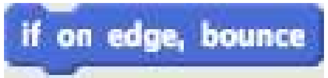
Basic, basic script (7)

18. Your script should look like this:



```
when [flag] clicked
forever
    move (10) steps
    if on edge, bounce
    next costume
```

19. To demonstrate how we can do multiple scripts simultaneously, *right-click* on  and choose “duplicate”.

20. Take the  block out of one loop and take  and  out of the other. Now, try it!

Play!

Try one or more of these:

- Click on the Stage and add multiple Backdrops, that change every 3 seconds.
- Make the cat change costumes more slowly.
- Make the cat’s color change as it moves.
- Make music play in the background.

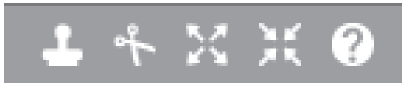
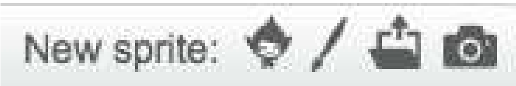
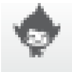
Lesson 2: More Control

In this lesson, we’ll learn about

- deleting sprites and adding sprites
- if statements
- user interaction
- drawing with the pen

Goal: make a program that draws lines when you click on the canvas.

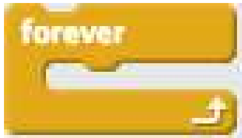
Lesson 2 (1)

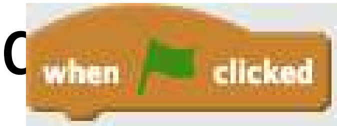
- From the main Scratch web page, click Create or, if you are already in the creation page, choose File → New.
- At the top, you'll see  . Hover over these to see what they do. Choose ✂ and click on the cat to delete it.
- Now, hover over these  and click  . You can choose anything you want, but I like the Beetle (under Animals) for this exercise. Click OK.

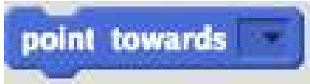
Lesson 2 (2)

- Let’s make this Beetle smaller. Choose and click on your sprite until it is a good size. Mine looks like s:
- Now, think: we want the sprite to turn to face the mouse pointer at all times. We also want it to move to where the mouse is when we click. Finally, we want it to draw a line each time. So, there are 3 steps. Can you figure out how to write this script?
- Breaking a problem up into smaller problems is called *Problem Decomposition*.

Lesson 2 (3)

- First, let’s make the sprite turn to face the mouse pointer. Get these onto the scripting area but don’t connect them to each other: , 

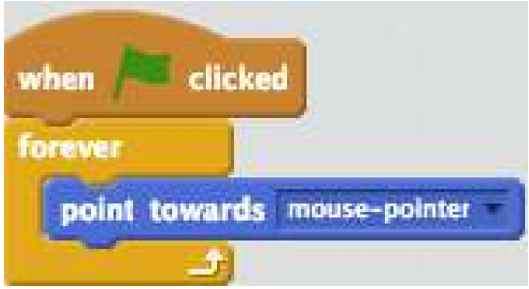
and 

. 

- Click on the down arrow in “point towards” and choose “mouse-pointer”. Double-click on this block now, and watch your sprite.
- Now, connect these blocks and see if you can get the sprite to follow your every move!

Lesson 2 (4)

- Your code should look like this:
- Click on this, next to your sprite: ℹ



```
when ⚑ clicked
forever
    point towards mouse-pointer ▾
```

- Notice that each sprite has: an (x, y) location, a direction, a rotation style, and whether it is visible or not.
  - Which direction in Scratch is northwest? Where is direction 0?

Lesson 2 (5)

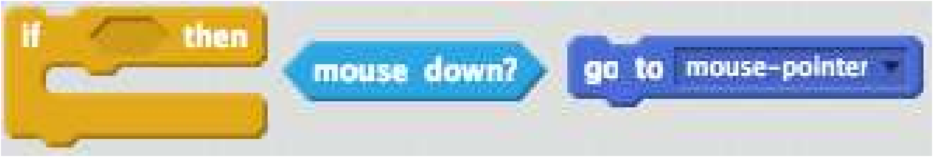
- Look above the sprite area and below the canvas and you'll see something like this:

x: -156 y: -67

- What are these numbers?
- Watching these numbers, figure out how big the canvas is:
  - What is the smallest x value? _____ Largest? _____
  - What is the smallest y value? _____ Largest? _____
- Stop 🛑 your script. Now, drag your sprite to (0, 0). What part of the sprite is exactly at (0, 0)?

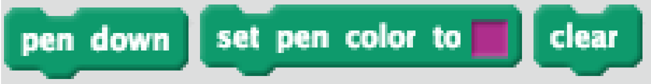
Lesson 2 (6)

- Back to the lesson: now we want the sprite to move to wherever we click. For this step, we'll need:



- (If you have problems finding these, their colors should help you figure out what group they are in.)
- Put them together and then put the whole thing into your forever loop.

Lesson 2 (7)

- Final step: make the sprite draw its path whenever it moves. For this, you'll need these: 

pen down

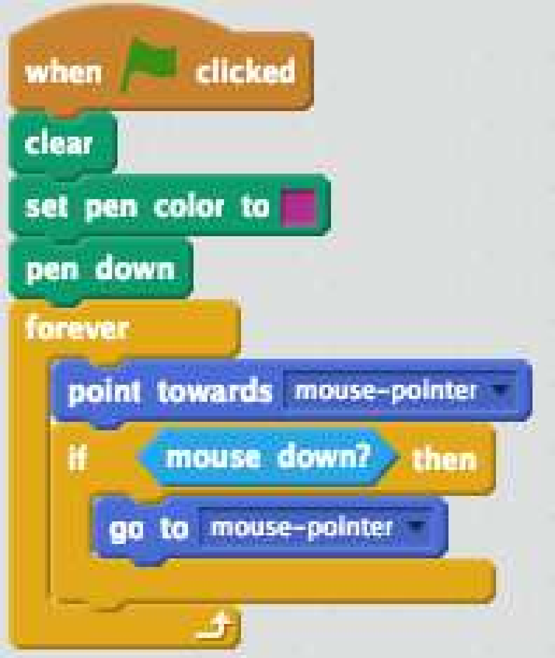
set pen color to 🟪

clear

- Figure out the order of these and where to attach them in the script. When you are done, you should be able to
  - draw whenever you click.
  - stop the script and then when you start, it should be on a clear canvas.
- What happens if you keep the mouse button down while you move it around?

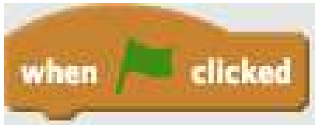
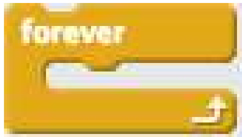
Lesson 2 (8)

- My solution looks like this:



```
when 🏁 clicked
clear
set pen color to ■
pen down
forever
    point towards mouse-pointer ▾
    if mouse down? then
        go to mouse-pointer ▾
```

Notice a few things:

- Under  , there is code to *initialize* the setup.
- Then there is the  with code in it.
- This organization is very common.

Play!

Try one or more of these:

- Make the color of the pen change (constantly, or whenever you click).
- Set or change the width of the pen.
- Make the sprite always start at (0, 0) when you start.
- Make the sprite “glide” to where you have clicked, instead of “zipping” right there.
  - Note that there is no “glide toward” command…

Lesson 3: More interaction

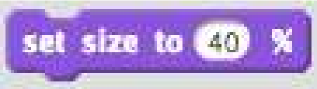
In this lesson, we’ll learn about

- creating custom backdrops
- if-else statements
- sensing colors
- reacting to keyboard “events”.

Goal: make a program that lets you navigate the cat through a maze.

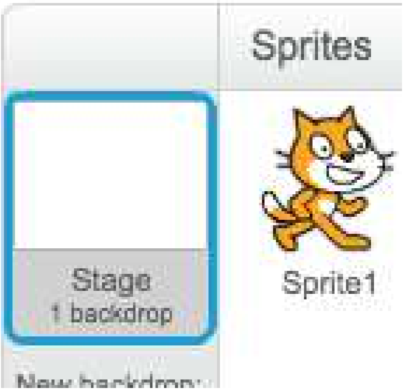
Lesson 3 (1)

- Shrink your cat down to a pretty small size. I made mine 40% of initial size. (You can do this quickly by dragging this to your script area and double-clicking on it once.)



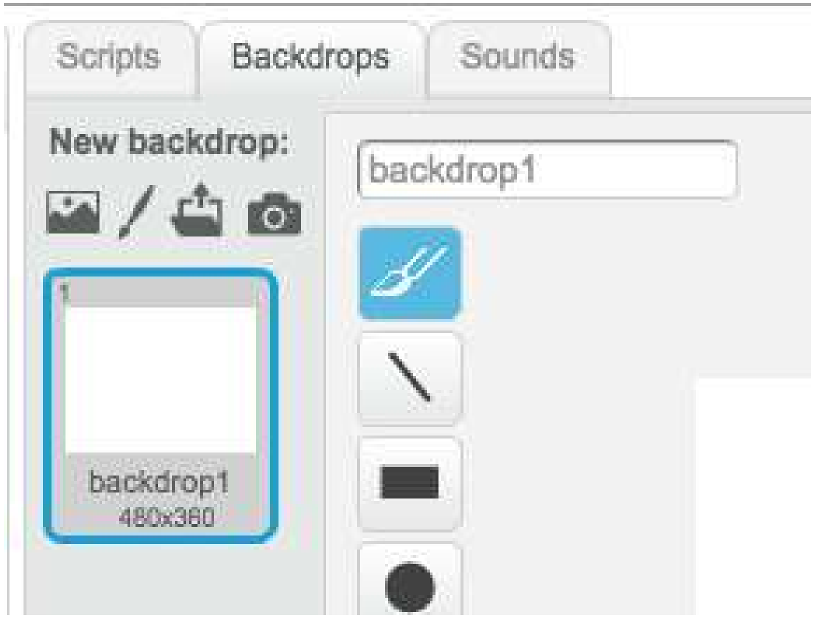
```
set size to 40 %
```

- Drag your cat to the lower left corner of the screen.
- Select the Stage to the left of your cat Sprite:



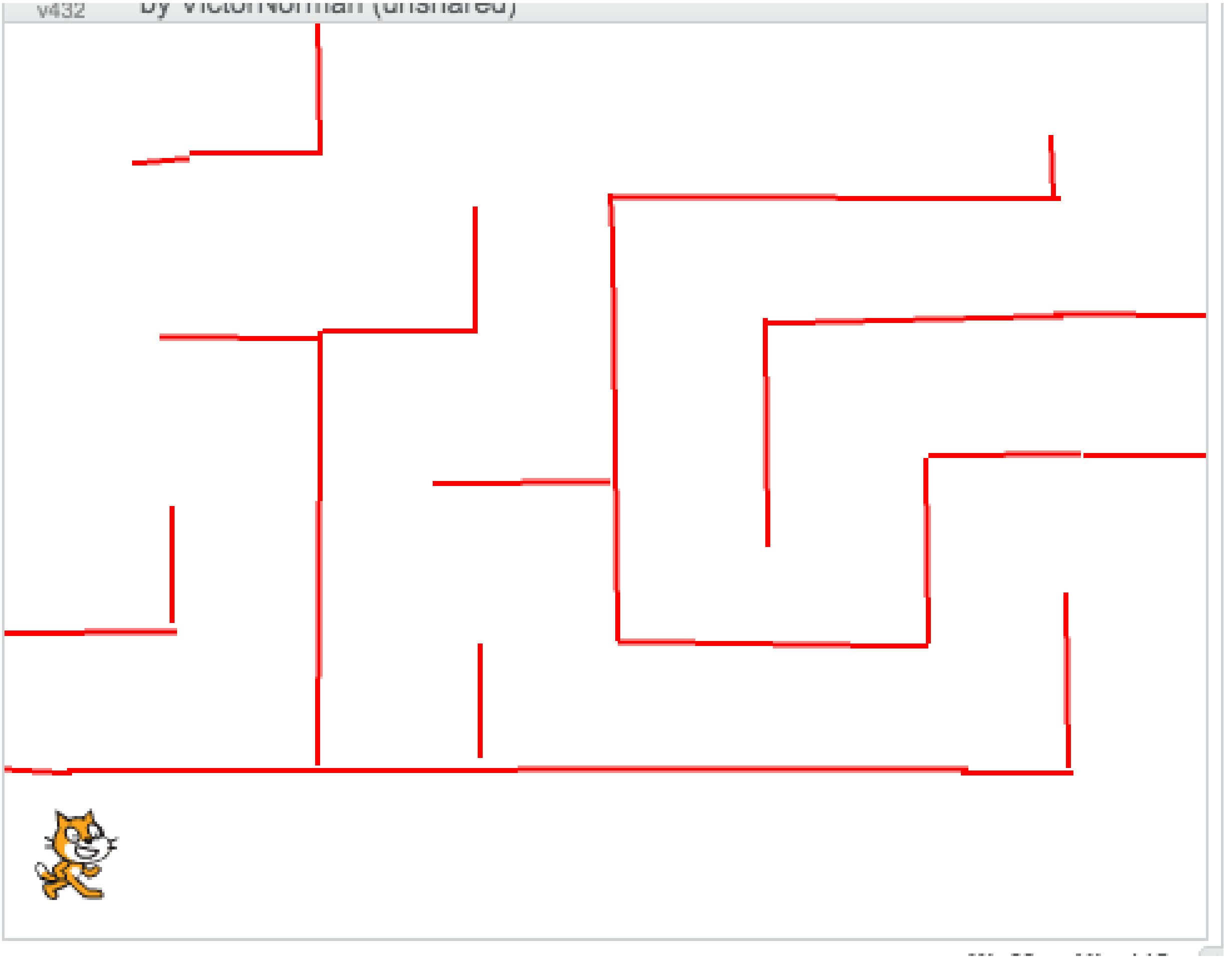
Lesson 3 (2)

- Now, click on the Backdrops tab:
- Click on the Line tool.
- Choose a color. I chose red.
- Create a maze for your cat to have to negotiate through. Make sure the hallways are wide enough for your cat to fit through.

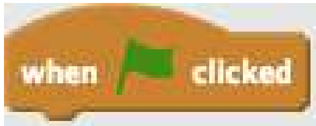
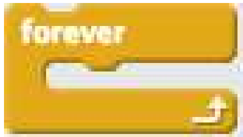


Lesson 3 (3)

- My maze ended up looking like this:

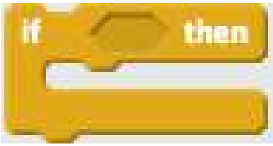
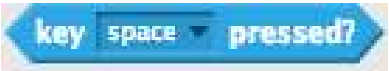


Lesson 3 (4)

- Click on the cat again, and get into the Script writing “mode”.
- Put a  block and  block in and connect them.

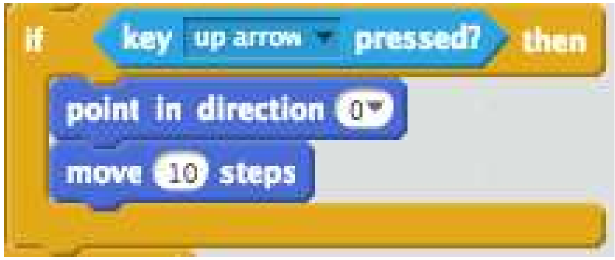
- We need to check if the user pressed the up-arrow key, and if so, the cat moves up. See if you can figure out how to do this.

Lesson 3 (5)

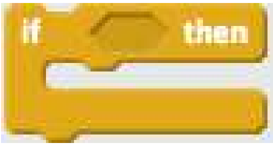
- You need the . This is a *conditional* block: if the "condition" is true, then the code inside it is run. Otherwise, it isn't.
- The condition we need to check is the up-arrow key being pressed: 
- Now, if that condition is true, we want to make the cat face up and then move 10 steps.
- See if you can figure this out.

Lesson 3 (6)

- My code now looks like this (inside the forever loop):

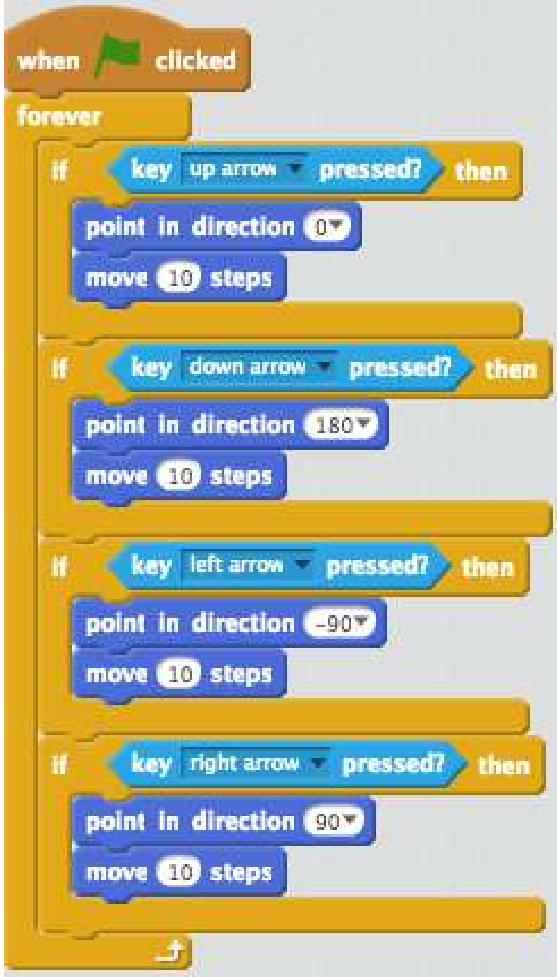


```
if key up arrow pressed? then
  point in direction 0
  move 10 steps
```

- Try it!
- Right-click on the  and duplicate the code. Fix those replacements so that they handle the down, left, and right arrow keys.

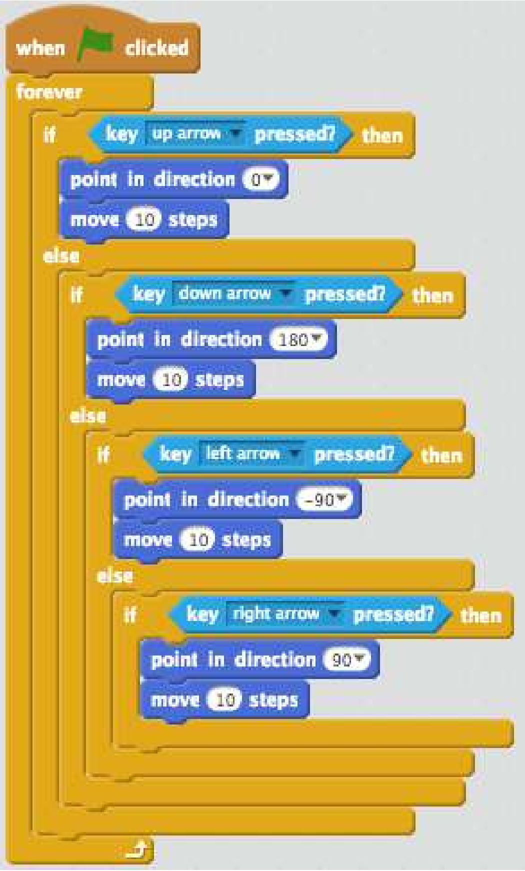
Lesson 3 (7)

My code looks like:



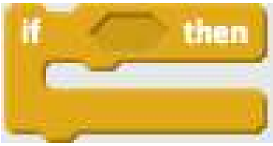
```
when ⚑ clicked
forever
    if <key [up arrow ▾] pressed?> then
        point in direction (0 ▾)
        move (10) steps
    if <key [down arrow ▾] pressed?> then
        point in direction (180 ▾)
        move (10) steps
    if <key [left arrow ▾] pressed?> then
        point in direction (-90 ▾)
        move (10) steps
    if <key [right arrow ▾] pressed?> then
        point in direction (90 ▾)
        move (10) steps
```

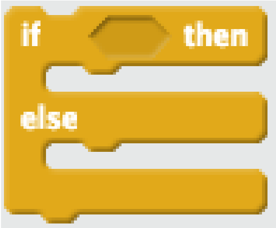
Alternatively we could use if-else blocks:



```
when ⚑ clicked
forever
    if <key [up arrow ▾] pressed?> then
        point in direction (0 ▾)
        move (10) steps
    else
        if <key [down arrow ▾] pressed?> then
            point in direction (180 ▾)
            move (10) steps
        else
            if <key [left arrow ▾] pressed?> then
                point in direction (-90 ▾)
                move (10) steps
            else
                if <key [right arrow ▾] pressed?> then
                    point in direction (90 ▾)
                    move (10) steps
```

Lesson 3 (8)

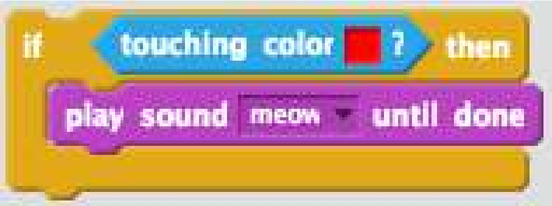
- Either the multiple 

or multiple nested 

blocks work in this case because we only have to handle one keypress at a time, and order does not matter.
- At this point, you should be able to walk your cat around the screen. But, nothing happens when you hit a wall. How to handle this?

Lesson 3 (9)

- Look in the Sensing area. How can we tell if the cat is touching a wall?
- What should happen when the cat touches a wall?
- Here is my code:



```
if touching color 🟥 ? then
    play sound meow until done
```

- Where should this code be put?
- Perhaps the cat should also say "Ouch!" Try it!

Lesson 3 (10)

- Each time you restart your game, you cat should be moved to the start position.
- Add code to do this.
- Next, add a second sprite – a mouse, perhaps – and place it in your maze.
- Add code so that when the cat touches the mouse, something happens, like, perhaps:
  - the mouse disappears and the cat says “Yum!”
  - the mouse grows much bigger and eats the cat!
  - the cat and mouse have a nice conversation.
  - the game ends.

Play!

- Try one or more of the following:
  - at the end, have a Game Over screen.
  - if the cat touches a wall, have it move back 10 steps (hard, I think).
  - have the mouse jump to new set locations periodically (and randomly).

Lesson 4: Art!

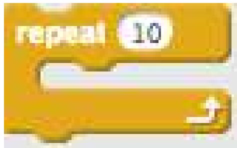
In this lesson, we'll learn about…

- creating custom blocks.
- how to create art with Scratch.
- sending messages.
- getting user input and using variables.

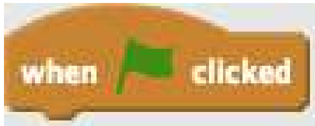
Goal of the project: make a program that draws beautiful colorful pictures.

Lesson 4 (1)

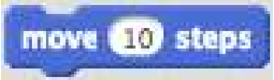
- Create a new project
- In the script area for the cat, combine the following blocks to make the cat draw a square, 100 "units" long on each side:



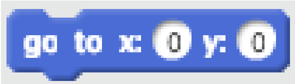
repeat 10



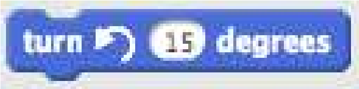
when 🟩 clicked



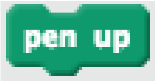
move 10 steps



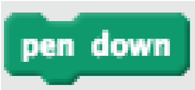
go to x: 0 y: 0



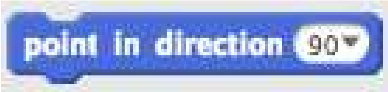
turn ↺ 15 degrees



pen up



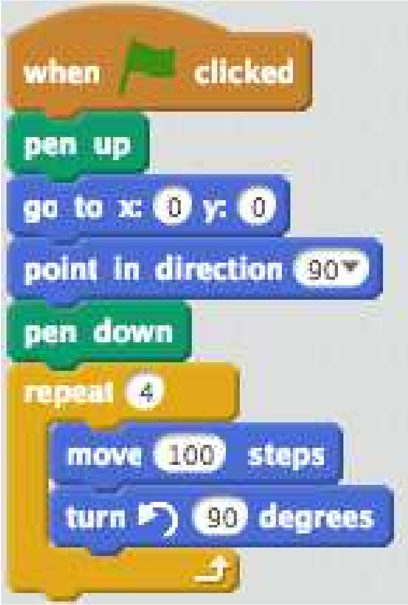
pen down



point in direction 90▾

Lesson 4 (2)

- Your code should look like this:

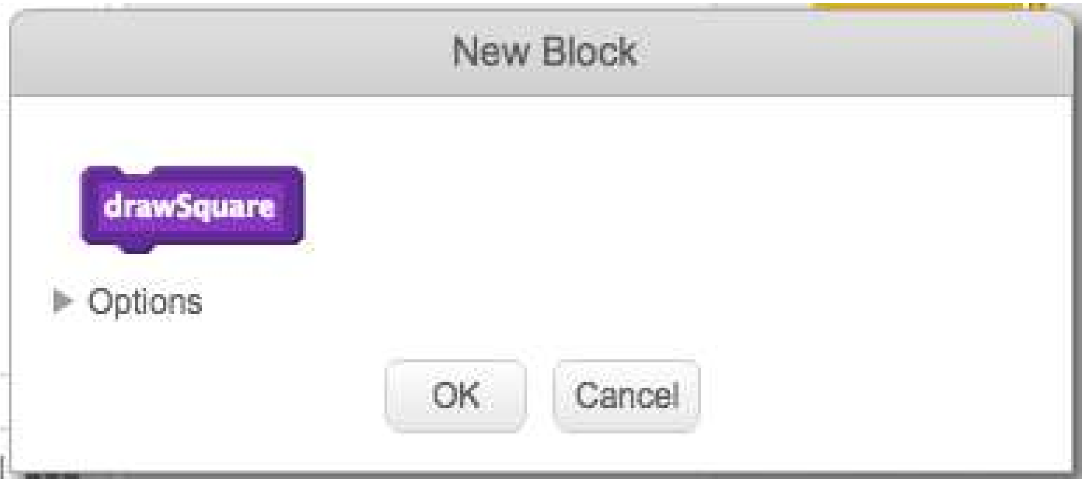


```
when ⚑ clicked
pen up
go to x: 0 y: 0
point in direction 90▾
pen down
repeat 4
    move 100 steps
    turn ↺ 90 degrees
```

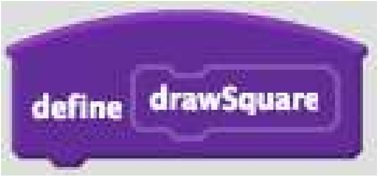
- Notice that the repeat loop draws a square. Let’s make a new block called “drawSquare” that does that code:
- First, click on More Blocks , and click Make a Block

Lesson 4 (3)

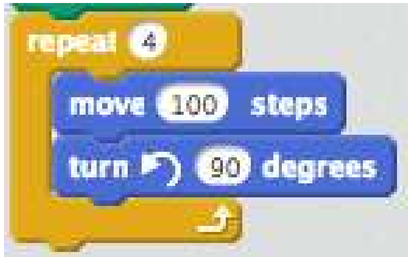
- In the resulting dialog box, call your new block drawSqu and click OK.



- You get a resulting block like this:



Lesson 4 (4)

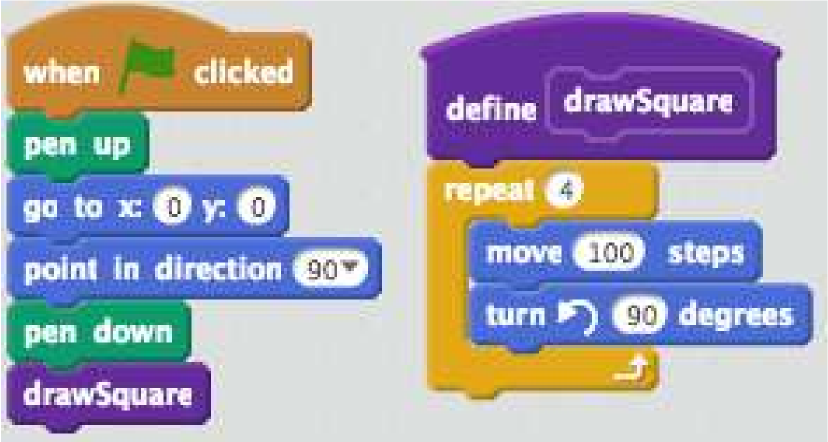
- Now, move your  into this new block.

```
repeat (4)
    move (100) steps
    turn ↺ (90) degrees
```

- And, call from your “main” code.

```
drawSquare
```

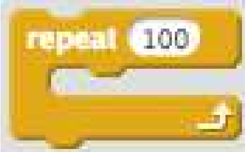
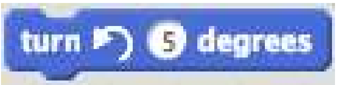
- In the end, your whole script should look like:

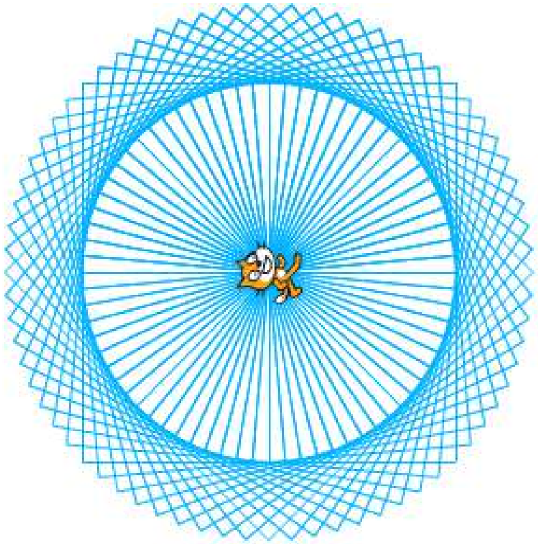


```
when 🏴 clicked
pen up
go to x: (0) y: (0)
point in direction (90 ▾)
pen down
drawSquare
```

```
define drawSquare
repeat (4)
    move (100) steps
    turn ↺ (90) degrees
```

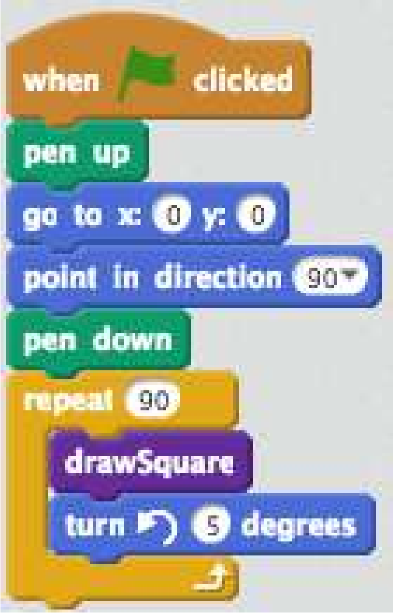
Lesson 4 (5)

- Now, let’s make the cat draw multiple squares, with a little turn between each one. To do this, you need another  repeat 100 and a  turn ↶ 5 degrees .
- If you get these in the correct places in your code, you should get a picture like this:



Lesson 4 (6)

- Your “main” code should look like this:

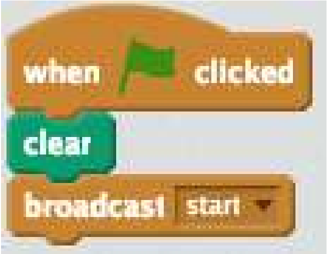


```
when ⚑ clicked
pen up
go to x: 0 y: 0
point in direction 90▾
pen down
repeat 90
    drawSquare
    turn ↺ 5 degrees
```

- Things get more interesting if you make the pen change color for each square. To do this, add this `change pen color by 10` in the correct place.
- You might want to clear the screen before each run: add `clear` to the correct place.

Lesson 4 (7)

- Now, the best way to organize Scratch code is to put all project setup code in the Stage script area, and then *broadcast* a *message* to each object that needs to run code.
- Add this code to your Stage script area:

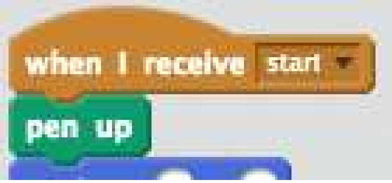


```
when ⚑ clicked
clear
broadcast start
```

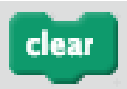
To do this, you have to create the new message “start”.

Lesson 4 (8)

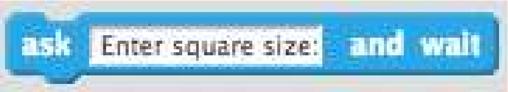
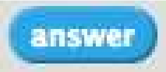
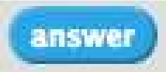
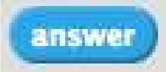
- Now, change the code for your cat sprite to run when it receives the “start” message:



```
when I receive start ▾
pen up
```

- Note that I put the  clear in the code in the Stage because that is set-up code for the whole project. The rest of the initialization code is sprite-specific so I left it in the cat sprite script area.

Lesson 4 (9)

- Now, let’s make it so that the program asks the user how big to draw the square.
- Add this block  ask Enter square size: and wait to your code for the Stage. The answer the user gives will be stored in the variable  answer .
- Now, we want the cat’s code to use the value in  answer instead of the hard-coded value 100.
- Find the place where 100 is used, and replace it with  answer .

Play

- Try one or more of these:
  - ask the user for the number of sides of the shape, and draw that regular polygon instead of a square.
  - make it so that if the user enters 0, a random value between 25 and 150 is chosen.
  - make it so that drawSquare is called with a *parameter*, and this parameter is changed by 1 each time drawSquare is called.

Lesson 5: Physics

In this project, we'll see how Scratch can be used to help teach physics concepts. We'll also learn more about

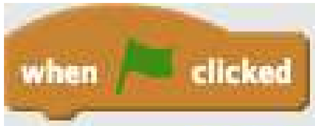
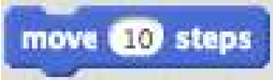
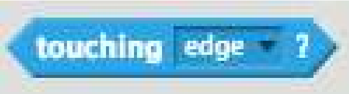
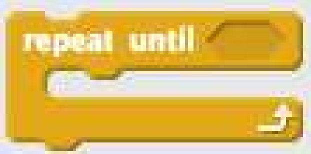
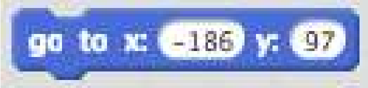
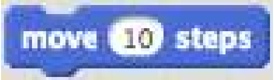
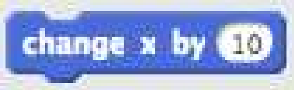
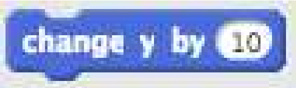
- variables
- wait block

Goal: create a simulation of a ball being tossed, and accelerating downward and bouncing.

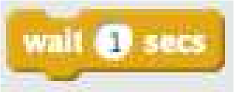
Lesson 5 (1)

- Create a new project.
- Move the Scratch cat up to the upper left corner, at about (-210, 100), and shrink it significantly.
- Create a 2nd sprite – a tennis ball – and put to the right of the cat.

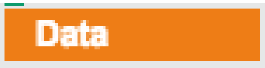
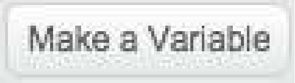
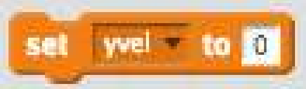
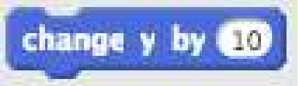
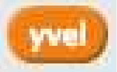
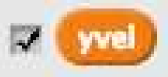
Lesson 5 (2)

- For the tennis ball, create a script to repeatedly move the tennis ball to the right. Use these blocks:  when clicked  move 10 steps  touching edge ?  repeat until  go to x: -186 y: 97
- Try it.
- Now, replace  move 10 steps with  change x by 10 and  change y by 10 but make y change by -10.
- Before running the code, predict what the ball will do.

Lesson 5 (3)

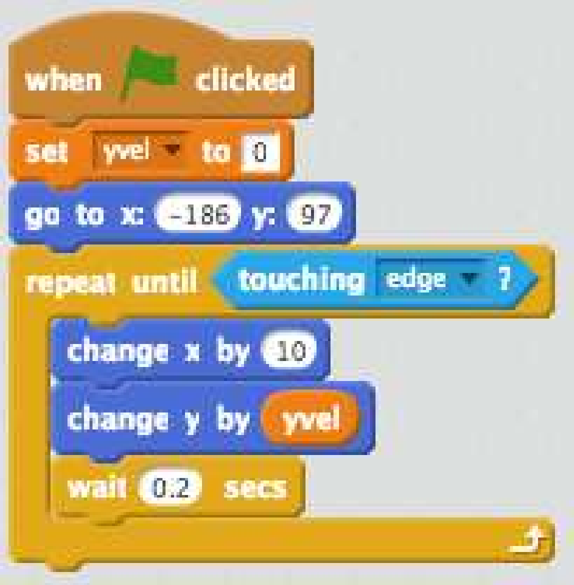
- Try changing the y value to different values and see what effect it has.
- Add a  wait 1 secs call in the loop, with value 0.2.
- Now, when you throw a ball horizontally, the horizontal velocity (speed) is constant – it does not change. Only the vertical velocity changes. Gravity changes the vertical velocity by constantly adding more to it.
- How can we simulate this?

Lesson 5 (4)

- Let’s create a variable *yvel*, initialized to 0, and use that to hold the current vertical velocity:
- Click on , and then . Call the variable *yvel*, and click OK.
- Add  to your code at the top, and change  to use the value of yvel: 
- Click on the box next to *yvel* so that you can see its value displayed on the screen: 

Lesson 5 (5)

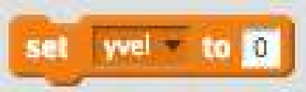
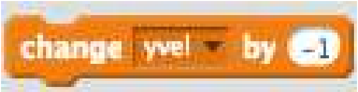
- Run your code: it should be the same as before.
If it isn't, compare your code to mine ---->



```
when [flag] clicked
set yvel to 0
go to x: -186 y: 97
repeat until touching edge ?
    change x by 10
    change y by yvel
    wait 0.2 secs
```

- Add clear , pen up , and pen down to your script. This is a little tricky. We want to raise the pen before resetting the ball's location, then lower it afterward. Test your code.

Lesson 5 (6)

- Now, change  set yvel to 0 to change yvel by -5.
- When you run the code, you see a staircase look. This demonstrates the xvel and yvel components independently.
- How do we make the ball arc downward, like would happen in real life? Gravity adds velocity to a ball, so we need to do the same.
- Add this code in the loop:  change yvel by -1

Lesson 5 (7)

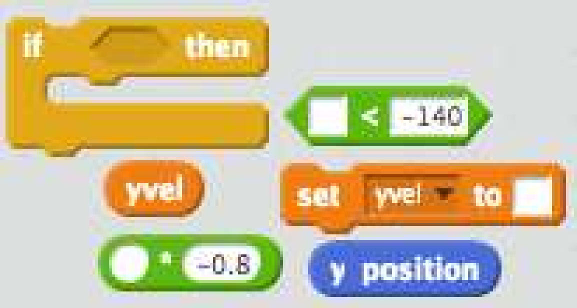
- Now, we can see how the xvel remains constant but the yvel changes a little each time. This is *acceleration.*
- Note that you can see the acceleration both in the drawing and in the *yvel* shown on the screen.
- Now, let’s make the ball bounce. What happens to the velocities when a ball bounces?

Lesson 5 (8)

- Does the x velocity change? ________
- Does the y velocity change? ________
- If so, by how much?
- We need to change the y velocity from, say, -10 to 10 when it bounces. I.e., we need to multiply it by -1. But, some energy is lost when a ball bounces, so instead we'll multiply by -0.8.
- But, first how do we write code to detect when the ball should bounce?

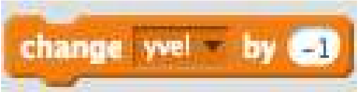
Lesson 5 (9)

- Let’s say that the bottom of the screen is at -140 (the bottom is really -180).
- Then, we will bounce the ball when the y position of the ball is < -140. Add these blocks to your code, in the correct order and correct place:



- Finaly, change your x velocity from 10 to 5 or 3, so that you can see multiple bounces.

Play!

- The staircase effect is good for learning, but unrealistic. How would you change the code to remove it?
- Add a comment to  change yvel by -1 to explain how this simulates acceleration. Do this by right-clicking on the block.
- Add code to ask the user for an x velocity and then move the ball that much each time.
- Add a platform for the cat to stand on, green grass at the bottom, and bouncing sounds.

Solutions to some challenges

- Search in Scratch for **`VictorNorman`** and look at “Bouncing ball simulation v2” (scratch.mit.edu/projects/30438566/) to see how to implement the platform, sound effects, etc.
- Search in Scratch for **`VictorNorman`** and look at “Bouncing ball simulation v3” (scratch.mit.edu/projects/30438566/) to see how to implement asking the user for a value, and using that for x vel and/or clearing the screen.

Other Ideas for Scratch

- Create a dynamic greeting card – an excellent creative activity for pre-Holiday fun.
- Tell a story (exercises creative writing skills).
- Create a music video. You can import music into scratch and play it, and then change scenes, dance moves, etc., for different parts of the song.
- Create a game from the physics demo: aim the ball and try to get it to land in a bucket.

Other Ideas for Scratch

- Frogger-like game
  http://scratch.mit.edu/projects/11424585/
- Repel the invading snowmen
  http://scratch.mit.edu/projects/16199527/
- Raindrops: demonstrates cloning
  http://scratch.mit.edu/projects/18267289/
- Math
  http://scratch.mit.edu/projects/16707152/
- Biology: predator-prey simulation
  http://scratch.mit.edu/projects/25807305/

Scratch Curriculum

- CS First (Google): http://www.cs-first.com/
- Middle Years CS: https://www.cs.hmc.edu/MyCS/index.html
- Educational material for Scratch: http://scratched.gse.harvard.edu/
  - e.g., Scratch Curriculum Guide http://scratched.gse.harvard.edu/resources/scratch-curriculum-guide

Other tools for teaching programming

- http://www.calvin.edu/~vtn2/outreach.html
- Idea: develop Scratch activities to align with Common Core instructional goals. Anyone want to help? ☺