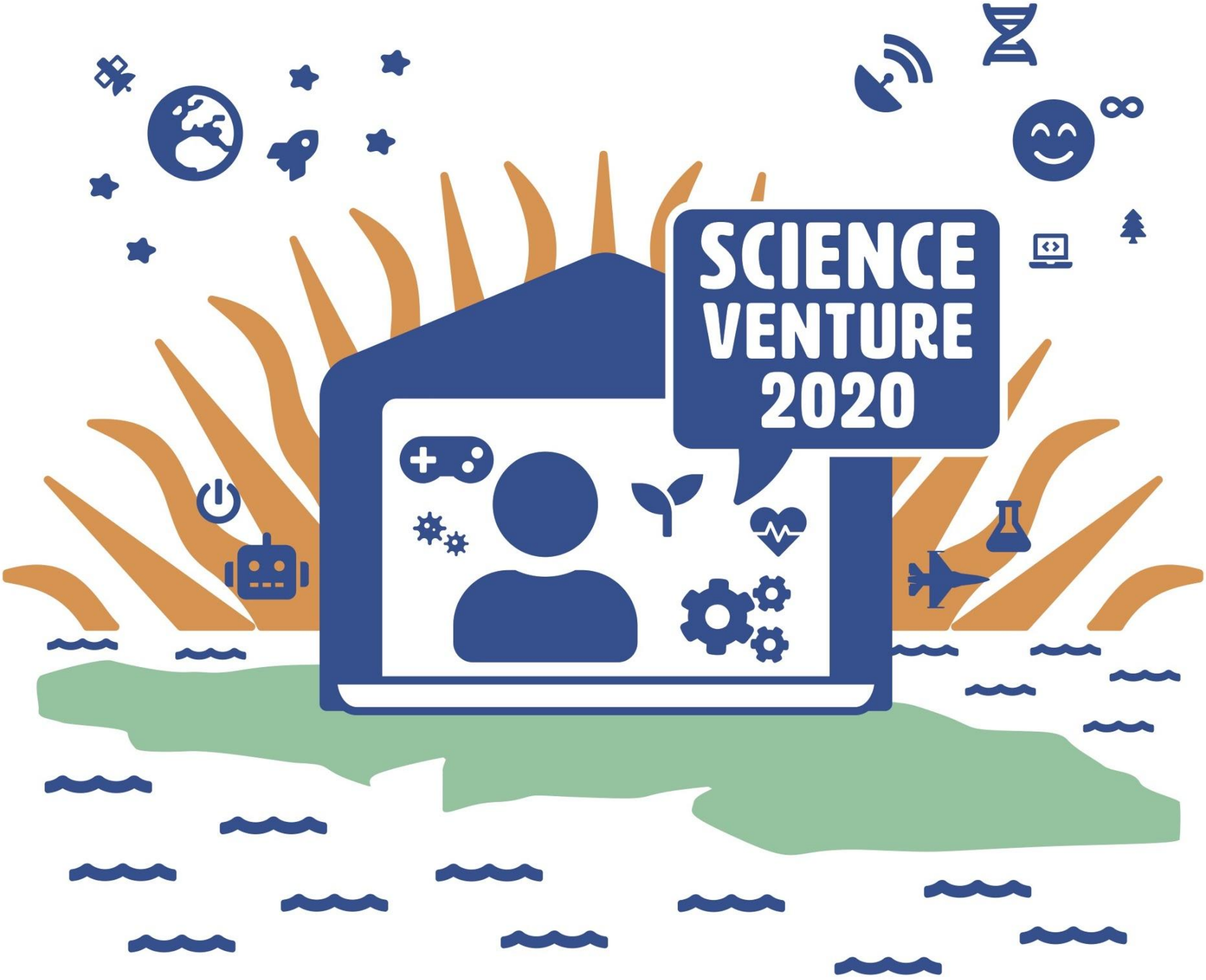


CoderGirl

Scratch Scores

In this lesson, campers will learn how to add scores to their game! Instructors will lead campers through the development of a simple chase game that keeps track of the score. Use this lesson to show some helpful code that campers may want to add to make their game more interactive.

Materials

- Device with internet access

Key Words & Concepts

Scratch: a free block-based visual computer programming language and website. Users can create online projects using the block-like interface. Here are some common features of a Scratch program:

- Sprites: objects in our scene. They can be characters or props. Each sprite has its own script.
- Blocks: they control what the sprites and the backdrop do. They are used to code the behaviour of our characters. They can be attached together to make a list of instructions for a sprite to follow.
- Script: a list of instructions written using the blocks

Some of the categories of Blocks:

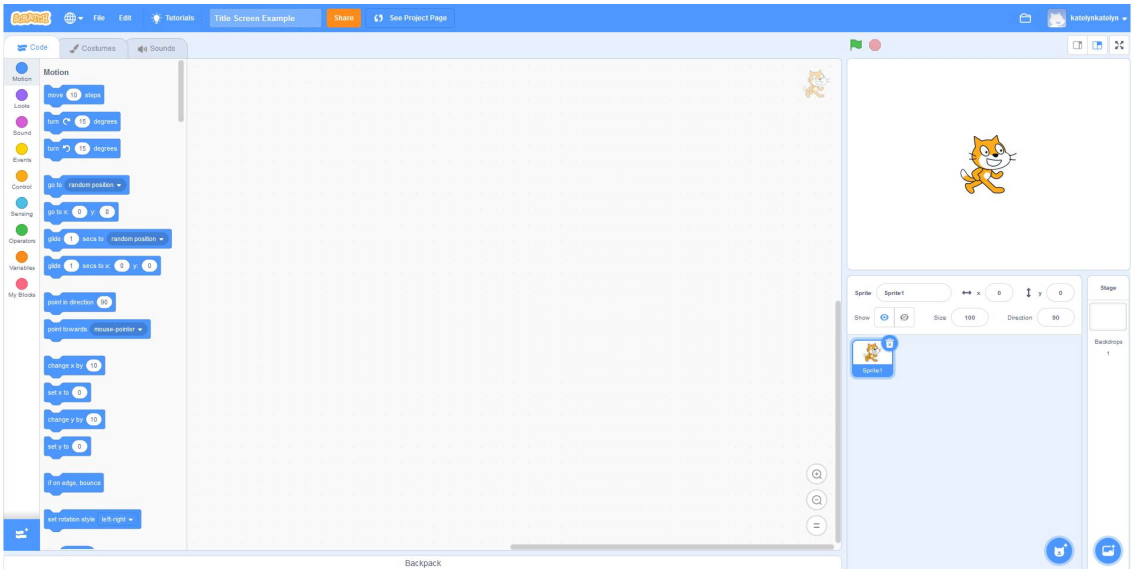
- Motion: blocks that control the movement of the sprite
- Looks: blocks that control what costume your sprite is wearing, whether the sprite is visible, and the size of your sprite
- Events: These must start every piece of code in order for it to run.
- Control: these help your sprite wait or repeat pieces of code

Scratch Scores

What is Scratch?

Scratch is software that lets us program animations, interactive stories, and games by dragging and dropping code blocks to control the characters and background.

To use the Scratch editor, go to scratch.mit.edu and click on the "**Create**" tab near the left of the site's navigation bar. Your screen should look like this:



Sprite: Sprite1 ↔ x 0 ↕ y 0
Show Size 100 Direction 90

On the very left is the **block drawer**. You can drag blocks from there into the main script area to use, and you can also drop blocks there that you want to discard. The top right corner with the orange cat is the stage. This is where all the action happens! The block scripts (everything in the main script area) control what happens on the stage.

On the right below the stage, you can see all the **sprites**, or objects, in your scene. The currently selected sprite will be outlined in blue as shown in this picture. In this scene, the orange cat is the only sprite.

On the upper left there are three tabs: **code**, **costumes**, and **sounds**. We will be focusing on code and costumes this week.

The **code tab** is where you can see and edit block scripts for the currently selected sprite.

The **costumes tab** is where you can see the different appearances your sprite has. With the use of certain blocks, sprites can easily change between costumes to show different actions and emotions

The **two blue circles** at the very bottom right of the screen let us add new sprites and backdrops, in that order.

Scratch Scores

Important note:
Each sprite has its own separate scripts, costumes, and sounds, so make sure you have selected the correct sprite before you start making changes to your project.

Helpful terminology

Sprites

- Any characters and objects we create are called sprites. The orange cat that you see when you first open a new Scratch project
- Each sprite has its own script and can be programmed with blocks to do certain things.
- Sprites can also have multiple costumes. Click on the “costume” tab to see which costumes your sprite currently has. From here, you can rename, edit, add, and delete costumes.

Stage

- The stage is the background of our scene.
- The stage can have multiple different costumes, just like sprites. On the stage, these costumes are called backdrops.
- The stage can be programmed with blocks to do things like switch backdrops.

Blocks

- Blocks are like instructions for our sprites and stage. Blocks can tell the sprites and stage to move, wait, follow the mouse, speak, and all sorts of other things.
- There are 9 categories of blocks. We will mostly be focusing on the following:
  - Motion: blocks that control the movement of the sprite. If you click on motion blocks while they are still in the block drawer, your sprite will move accordingly (this is a good way to fix your sprite if you accidentally rotate or flip it while coding).
  - Looks: These blocks control what costume your sprite is wearing, whether your sprite is visible, and the size of your sprite.
  - Events: These blocks must start every piece of code in order for it to run. Notice how they can attach to other blocks on the bottom, but not the top.
  - Control: These blocks help your sprite wait or repeat pieces of code.

Scratch Scores

Green flag

- Clicking the green flag starts our Scratch program (top left corner in the picture below)
- We can stop our Scratch program by clicking the red stop sign.
- Our program does not automatically reset each time it is restarted. We have to program in where the character starts, and what backdrop we want for the beginning.
- Under the 'control' block section, we can drag a "Start when green flag clicked". Any code we attach underneath this block will run from the top to the bottom once the green flag is clicked.

Saving your projects

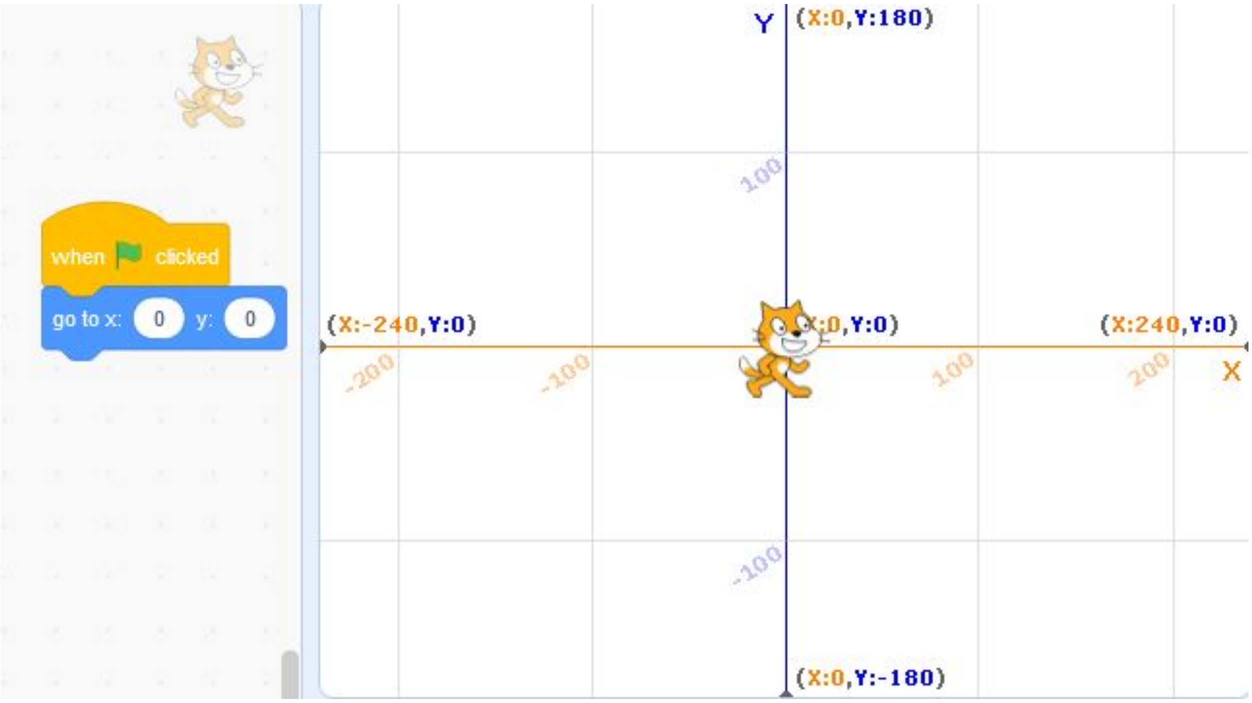
- If your parent/guardian is okay with it, you may make an account on Scratch (free, requires an email)
- Otherwise, you will save your project directly to the computer.
- Click on "File" at the top left corner of the screen and then click "Save to your computer"
- Choose a name for your project and a place on your computer to save it to (parents may need to help with this)
- We recommend saving often and naming each save with a version number at the end, for example: "Movie1", "Movie2", "Movie3", etc so that in case there is a bug in a newer version of the project that you can't fix, you can go back to the older version

X and Y

- We use something called a "coordinate system" to figure out where our sprites (characters and objects) are on the stage. We assign a number, or a coordinate, to our sprite and we can tell the computer where to move the sprite using these coordinates.
- We say that X controls the horizontal coordinate - how far left or right the character is.
  - Larger X coordinate = sprite is further right
  - Smaller X coordinate = sprite is further left
- We say that Y controls the vertical coordinate - how far up or down the character is.
  - Larger Ycoordinate = sprite is further up
  - Smaller Y coordinate = sprite is further down
- Here is what happens if we put X and Y both equal to 0: The sprite is at the center of the screen!

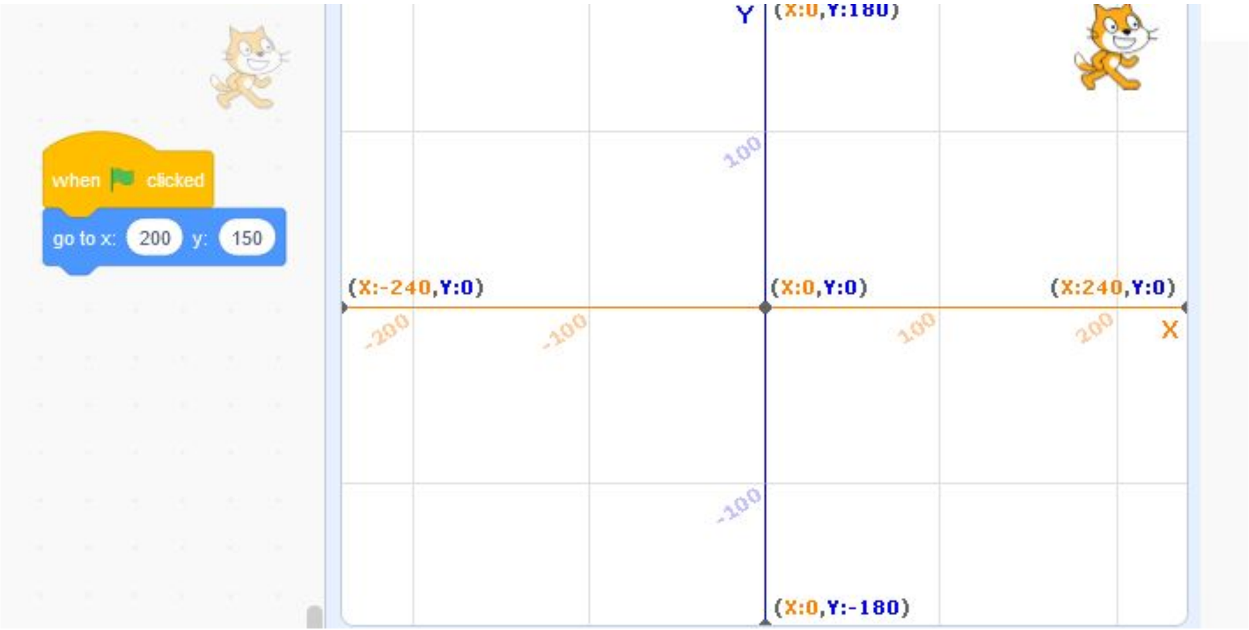
Scratch Scores

Here is what happens if we put X and Y both equal to 0: The sprite is at the center of the screen.



when 🏁 clicked
go to x: 0 y: 0

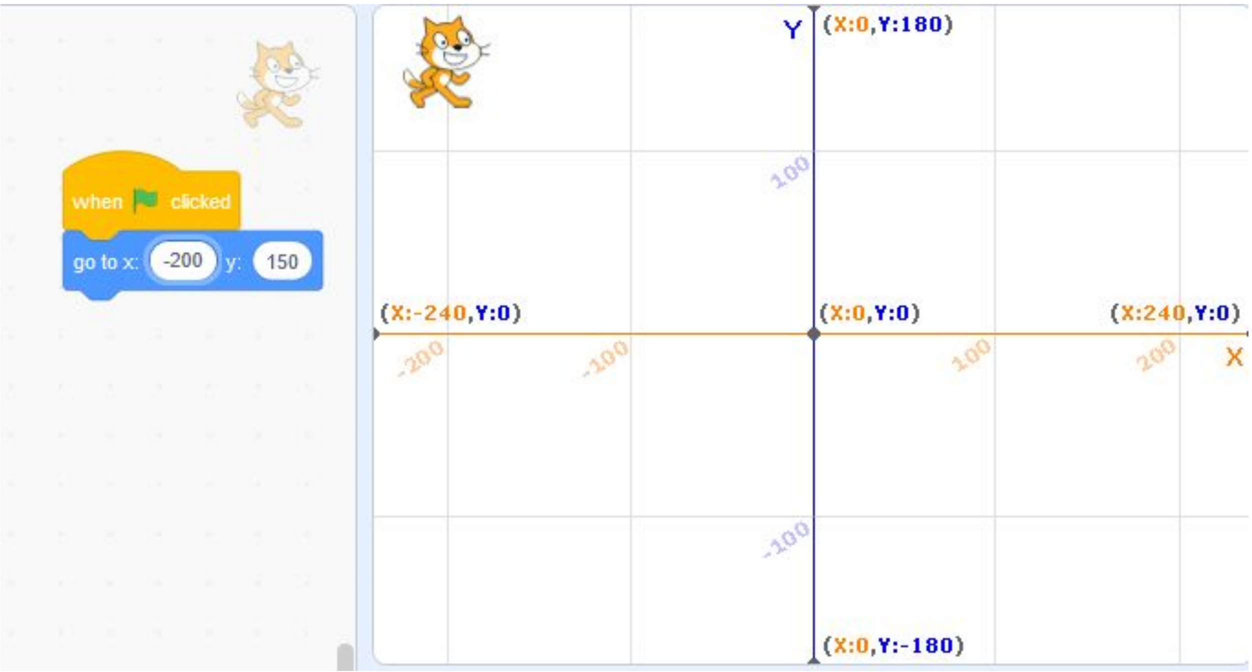
Here is what happens if we make both X and Y really big: The sprite goes very high up on the screen (large Y) and to the right of the screen (large x)



when 🏁 clicked
go to x: 200 y: 150

Scratch Scores

Here is what happens if we make X small but leave Y big: The sprite goes very high up on the screen (large Y) and to the left of the screen (small x)



```
when 🏁 clicked
go to x: -200 y: 150
```

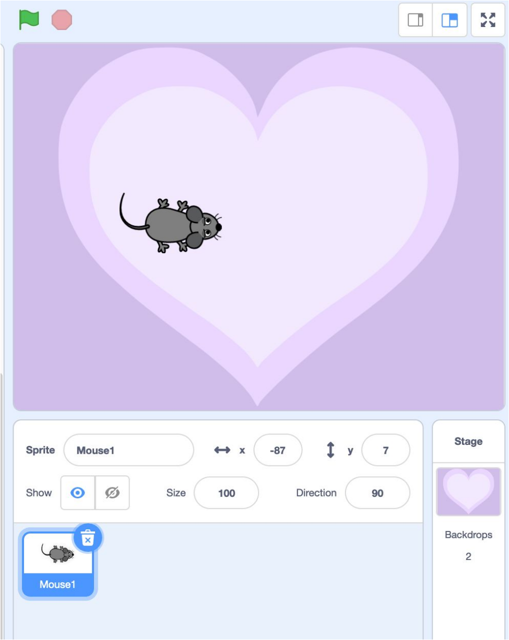
Try out some other combinations for yourself! How many combos can you come up with?

Scratch tutorials: https://scratch.mit.edu/projects/editor/?tutorial=all

Scratch Scores

Steps:

1) To start this program we need a sprite to be our character that is being chased. For my program I chose to use the mouse character. I have also chosen to change my background to make the program look more exciting!

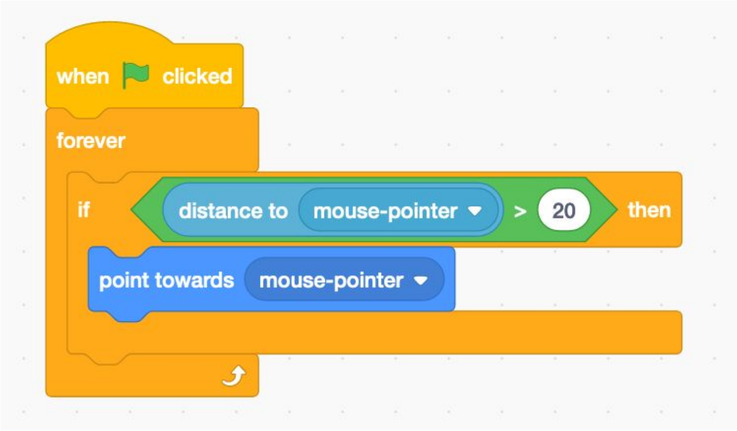


Sprite: Mouse1 ↔ x: -87 ↕ y: 7
Show Size: 100 Direction: 90
Stage
Backdrops: 2

2) To start coding the script for our mouse, we need the "when clicked block". This is so that our program will start running when we click the green flag.

3) Next, we want to connect a forever block to the first block since we want our code to continue running.

4) Now we want to add an if statement inside the forever block. For the condition, we want to check if the mouse pointer is greater than 20 units away. If yes we want the mouse to point towards the mouse-pointer.



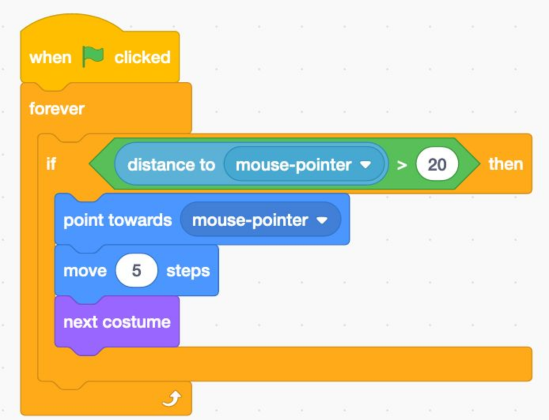
```
when 🏁 clicked
forever
    if distance to mouse-pointer > 20 then
        point towards mouse-pointer
```

Scratch Scores

5) We also want the mouse to move towards the mouse-pointer so we are going to add a move 5 steps forward block after the “point towards mouse-pointer” block.

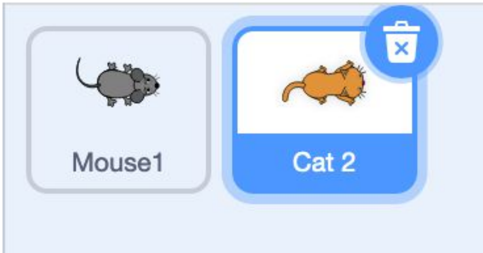
6) Test code. The mouse should follow the computer mouse around the screen.

7) The last part we want to add to this if statement is a “next costumes” block to make it look like the mouse is moving. The final block of code should look like the one below.



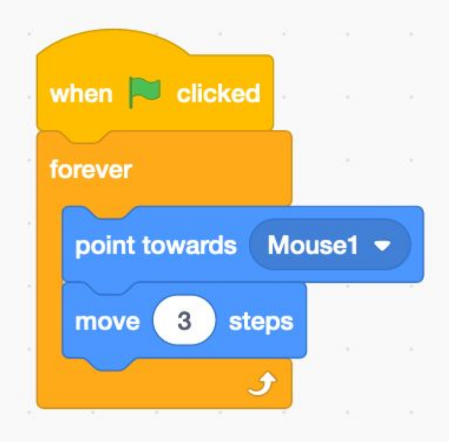
```
when [flag] clicked
forever
    if <(distance to (mouse-pointer ▾)) > (20)> then
        point towards (mouse-pointer ▾)
        move (5) steps
        next costume
```

8) Now that we have coded the mouse, we need a sprite to chase it. We are going to use the cat 2 sprite.



9) We are going to start the code for the cat similar to the mouse so we will need a “when green flag is clicked” block and a “forever” block.

10) Since we want this sprite to chase the mouse we want to use similar code as the mouse but instead of pointing to the computer mouse, we want the cat to point towards the mouse. We also want to reduce the number of steps the cat takes to make it easier for the user.



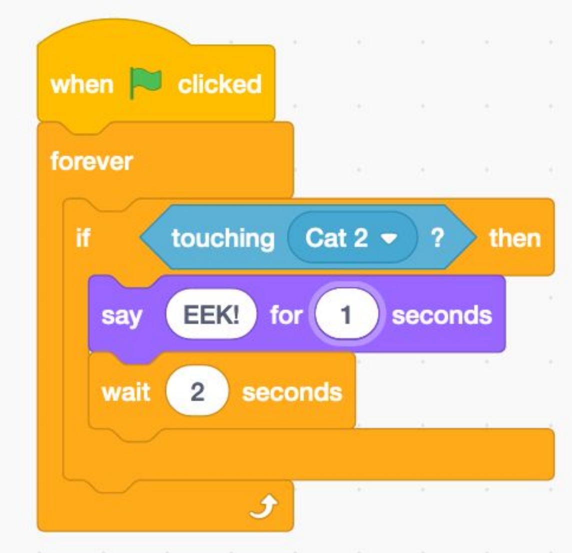
```
when [flag] clicked
forever
    point towards (Mouse1 ▾)
    move (3) steps
```

Scratch Scores

11) Test code. We should now have a game where the cat is chasing the mouse. Play around with the number of steps the sprites take to change the difficulty of the game!

12) To make this game more exciting we are going to add some more sound effects, visuals, and add a score!

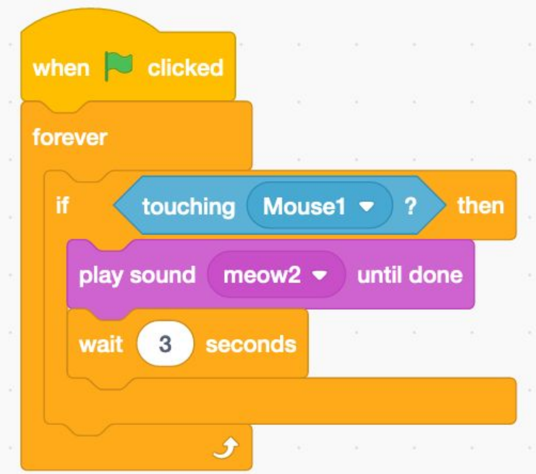
13) Go back to the mouse sprites code and we are going to make the mouse say something when it gets caught. Add another “when green flag is pressed” block and attach an if statement. The condition is going to be when the mouse is touching the cat, if yes then say EEK! Remember the if statement has to be in a forever block because we are constantly checking this condition! The code for this section is as seen below.



```
when 🏳 clicked
forever
    if touching Cat 2 ▾ ? then
        say EEK! for 1 seconds
        wait 2 seconds
```

14) Why did we need a separate block of code for this? Why, couldn’t we have put this if loop inside the original forever loop?

15) Now, we want the cat to react when it catches the mouse. We are going to use a sound effect for this. Go to the cat sprite and add a similar block of code, but instead of saying we are going to use the play sound block.

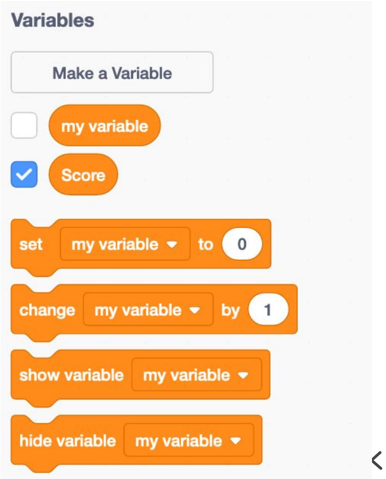


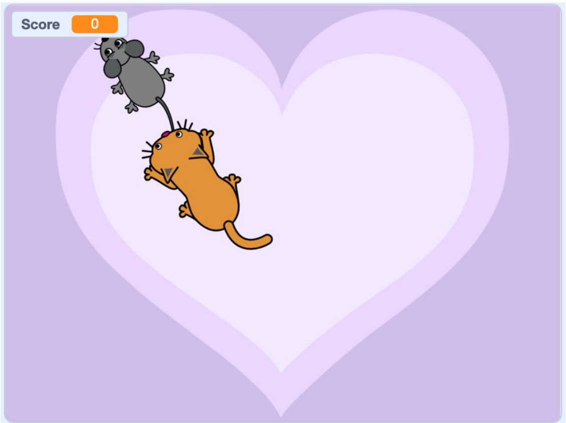
```
when 🏳 clicked
forever
    if touching Mouse1 ▾ ? then
        play sound meow2 ▾ until done
        wait 3 seconds
```

Scratch Scores

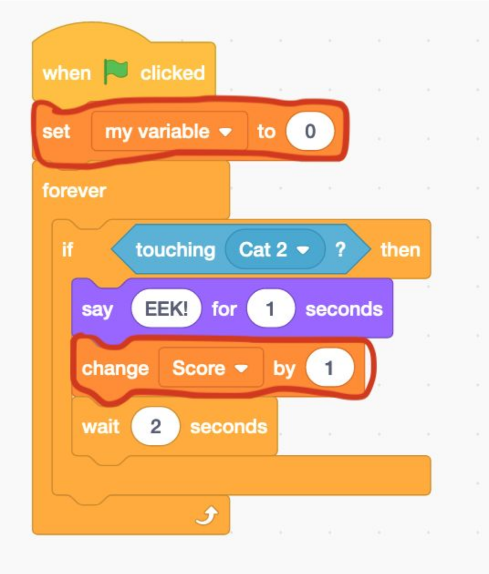
16) Test code. We should now have a game where the cat chases the mouse, and when it gets caught the mouse says eek and the cat meows.

17) Our game is almost done, we now want to add a score! To do this we need to make a new variable and call it Score. Make sure the box beside the variable is blue and has a check mark so it will appear on the stage.


Variables
Make a Variable
□ my variable
☑ Score
set my variable ▾ to 0
change my variable ▾ by 1
show variable my variable ▾
hide variable my variable ▾



18) Now, our scor k of everytime the cat catches the mouse. (The lower the score the better in this case!) To do this in our mouse code when we say EEK! We also want to add a block that adds one to the score. Remember whenever we use a variable we always have to reset it again to zero everytime we start the program. The two blocks needed for this are highlighted in red below.



```
when [flag] clicked
set my variable ▾ to 0
forever
  if touching Cat 2 ▾ ? then
    say EEK! for 1 seconds
    change Score ▾ by 1
    wait 2 seconds
```

19) If you want to view the full code go to https://scratch.mit.edu/projects/413729210/

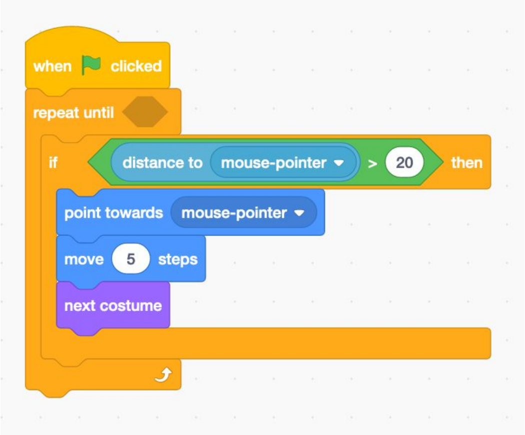
Scratch Scores

Take Home:

For today's take-home activity, we are going to continue working on our final project. If you would like to add a score to your game, follow some of the example code we went over in the live session.

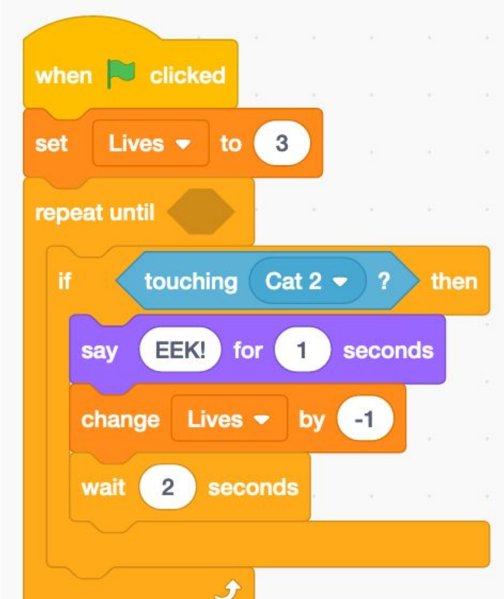
The finishing touch to our game is to make a game over screen. Below is a continued version of the game we were working on today in the live session, but with the addition of a game over screen.

1) To start modifying our program, replace all four forever blocks in our code with "repeat until" blocks. Below is an example of one of the blocks changed.



```
when ⚑ clicked
repeat until ⬡
    if distance to mouse-pointer ▾ > 20 then
        point towards mouse-pointer ▾
        move 5 steps
        next costume
```

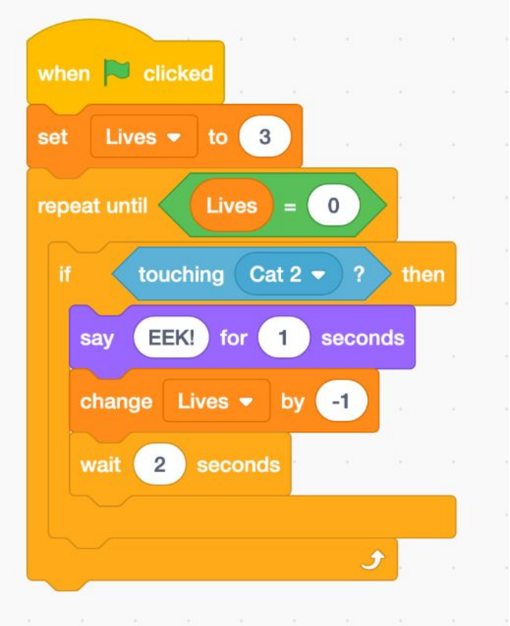
2) The condition we put in the repeat until is up to the game designer since it is based on the score value. However, for this example I am going to give the user 3 lives. Since we are now keeping track of lives instead of the score, we can make a new variable called "Lives". Just make sure in the code we are adding one to the variable "lives" and not the "score" now. Also since we are counting down lives we want our variable "Lives" to start at three and subtract each time the mouse is caught. Therefore, our mouse code should look similar to the code below.



```
when ⚑ clicked
set Lives ▾ to 3
repeat until ⬡
    if touching Cat 2 ▾ ? then
        say EEK! for 1 seconds
        change Lives ▾ by -1
        wait 2 seconds
```

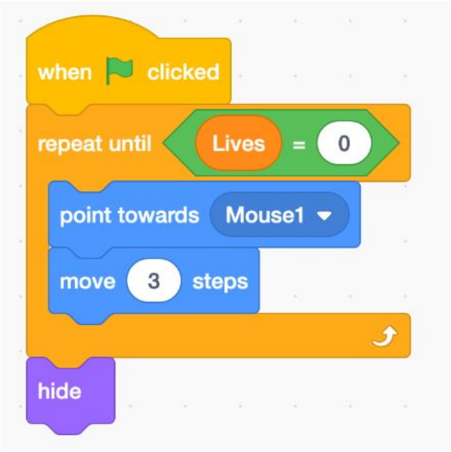
Scratch Scores

3) Now that we know we want to give the user 3 lives, once they are out of lives we want to switch to the game over screen. So we are going to add a condition to the repeat until loop saying we want to repeat until lives = 0. The condition is shown in the green block below.



```
when [flag] clicked
set Lives ▾ to 3
repeat until < Lives = 0 >
    if < touching Cat 2 ▾ ? > then
        say EEK! for 1 seconds
        change Lives ▾ by -1
        wait 2 seconds
```

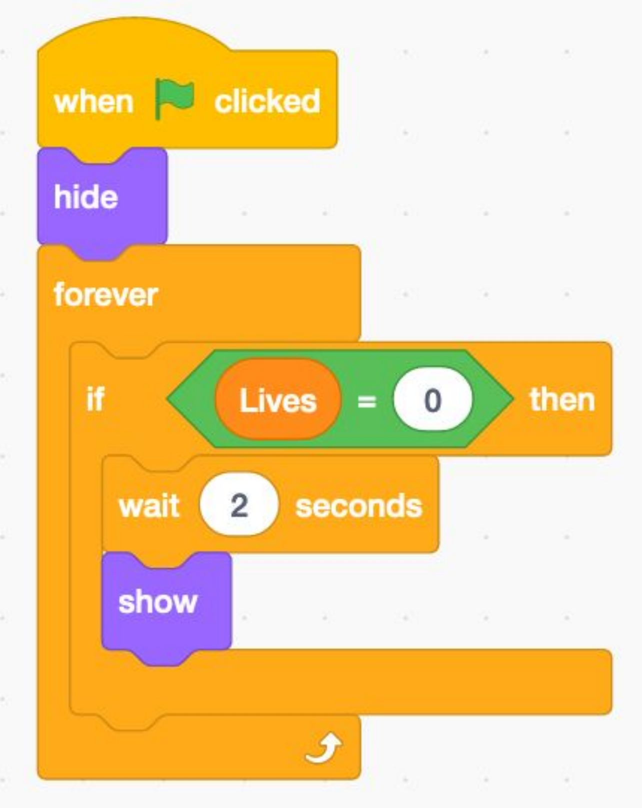
4) Now that we are repeating until we run out of lives, we need to add the code to switch to our game over screen. Outside of the repeat loop we are going to drag a hide block in both sprite characters code. Below is the example for the cat sprite, just add the same block at the end for the mouse sprite



```
when [flag] clicked
repeat until < Lives = 0 >
    point towards Mouse1 ▾
    move 3 steps
hide
```

5) For the final game over screen, I want to keep the same background, just display game over text. I created this text using https://makewordart.com/ once you create the message you would like to display, save it to your computer then upload it to Scratch as a sprite. You can upload from your computer by clicking the sprite dropdown menu at the bottom left of the screen, and then choose upload sprite.

6) We are going to add similar code that we have been using so far to code our game over message. We are going to use the “when green flag is clicked” block then we need to add a hide block right after to make sure that the sprite stays hidden. Then after the hide block, we need to attach a forever loop. Inside the forever loop is going to be an if statement with the condition if “Lives” = 0 then we want to show the sprite. In my code I have also added a wait block so it doesn’t happen all at once. The code for this sprite is shown below.



```
when 🏁 clicked
hide
forever
    if Lives = 0 then
        wait 2 seconds
        show
```

7) If you would like to look through the full finished version of this program, go to https://scratch.mit.edu/projects/413744432/

#SVatHome

Want to share your project or results with us?

Email or tag us @ScienceVenture

Have a question?

Reach us at svcamp@engr.uvic.ca