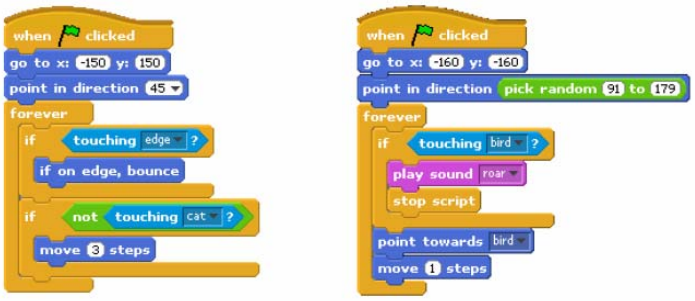
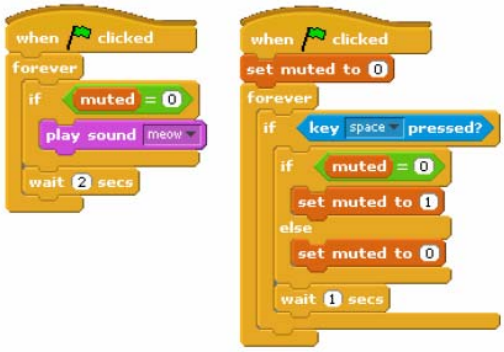
Threads (5:30-11:30)

- Back in the day, programs were not multi-threaded. Microsoft Word would not be able to do anything else as long as it was sending a document for print.
- Today, programs make use of threads to have several processes going on at once.
- Technically, this is just an illusion. The computer is really just switching between tasks repeatedly, spending a few milliseconds on each.
- But to the user, it appears that multiple things are being done at once.
- In Scratch, too, we can use threads to do make it appear that several things are happening at once. We represent threads as scripts and can tell Scratch to execute multiple scripts at once.
- In move2.sb, for instance, we execute a cat script and a bird script in order to have the two sprites move at the same time:



- In hello10.sb, one script allows us to mute the cat, while the other script handles the cat's meowing. The scripts execute at the same time.

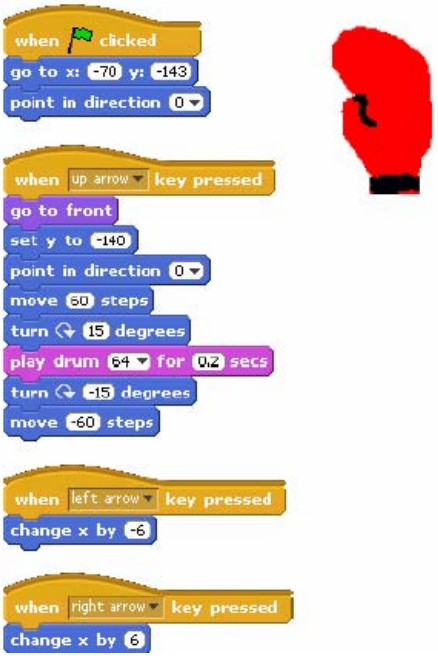


- When we press the space bar, the first script changes the value of the variable muted, and then the second script acknowledges this change in value and does not have the cat meow.
- Here we see an example of scripts sharing memory—the variable muted. Both scripts can check and change its value.

Writing a Program with Multiple Scripts: David.sb (11:30-15:00)

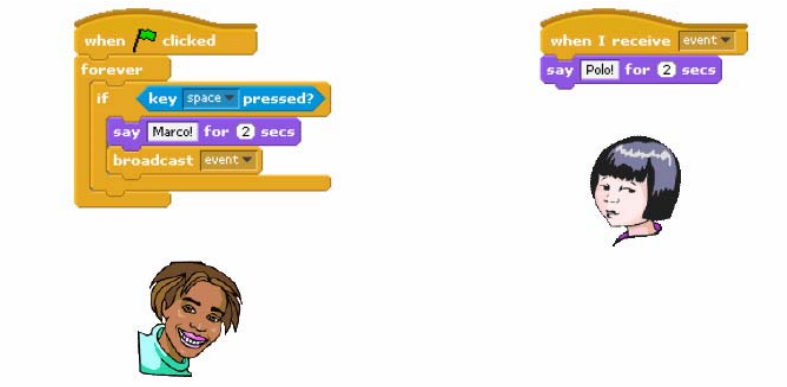
- In David.sb, a picture of a face moves back and forth across the screen and the user controls two boxing gloves to punch the face. After the face is punched several times, it goes through a series of color changes and finally gets "pwned."
- How would we go about implementing this program?

- First, we will need three sprites: two boxing gloves and a face.
  - We might begin by writing the script for the face. It moves back and forth across the screen in a loop.
  - For the boxing gloves, we would write a script that tells them to move only when the user presses a particular key.
- Next, we will set up the stage: a boxing ring background.
- Finally, we would add on the features that actually make this a game:
  - When a glove make contact with the face, the variable is incremented and a sound effect is played.
  - When the variable reaches certain values, the costume of the sprite changes.
  - When the variable reaches some highest value, the game is won.
- As you can see, this program has several scripts running at the same time which depend on some shared memory:



Event-Handling: Marco.sb (15:00-18:00)

- Scripts can communicate with one another by broadcasting events.
- One sprite will throw or trigger an event, and another sprite will listen or catch it.
- For example, in Marco.sb, the girl's script executes only when it receives the event from the boy's script:



Breaking Down a Complex Program: Oscartime.sb (18:00-24:30)

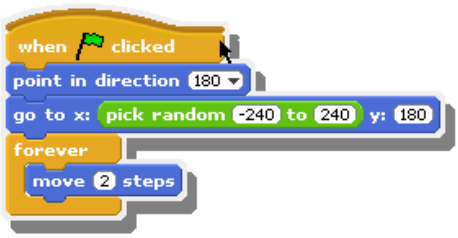
- In this program, trash falls from the sky and the user must click and drag each piece of trash into Oscar's trash can. As he receives trash, Oscar counts the number of pieces that have been collected so far. The object is to collect as much trash as possible before the song finishes.
- At first, Oscartime might seem very complex and difficult to implement. Where would you even begin?
- If we were going to write such a complex program, we would first break it down into its functional and structural components to make it more manageable.
- What functional components must Oscartime contain?
  1. Display instructions.
     - Use a stage
  2. Drop trash from the sky.
     - Represent trash with a gif image.
     - Give trash a location at top of screen with a random x-coordinate.
     - Decrease y-coordinate in a loop.
  3. Detect if the mouse is hovering over trash.
     - Use if statement.
  4. Allow user to drag trash.
     - Use if statement with "mouse-down?" Boolean
  5. Impose time limit
     - Control overall structure of program with Oscar sprite
     - Have this sprite perform actions at specified time intervals
     - Meanwhile other sprites do interesting things
     - When Oscar sprite reaches end of script, game is over
  6. Keep score
     - To change appearance of Oscar sprite as he pops out of the can, use costume changes
  7. Lift Oscar's lid to put trash in
     - Use if statement to determine when mouse is within 40 pixels of can
- Now that we have identified all of the functions it must perform, we can tackle the problem one step at a time.

- We would start by implementing just one of the steps above, and then, as we get small pieces to work properly, put them together to grow a larger and more complex program. In this manner we could ultimately produce the finished product you see here.
- For instance, we might start by implementing the falling trash…

Starting Small: Implement a Falling Cat (24:30-30:00)

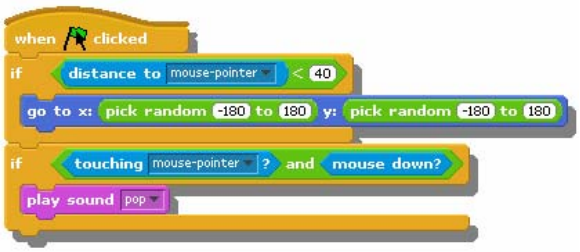
- Suppose we want to implement just one part of Oscartime: having trash fall from the sky. For simplicity, we will have a cat fall from the sky.
- How do we do this?
- First piece will be *when green flag clicked* to initiate the script.
- Next, position cat at top of screen: *go to (0,180)*
- To make cat move, put *move 2 steps* in a *forever* loop.

- Now, we test the program. It is very important to test small pieces as you build up a large program, rather than testing only at the end. This makes it easy to locate and fix bugs.
- Testing reveals that the cat moves the wrong direction.
- Go back and add *point in direction 180* before the movement loop.
- Test again. Now we see that the cat always falls from the same location, which would make the game too easy.
- Change the initial location of the act to (random number between -240 and 240, 180). Now, the user cannot anticipate the initial location of the cat.



Starting Small: Implement a Cat-and-Mouse Game (30:00-38:00)

- Suppose we want to implement a program where the object is to click on the cat, but the cat moves away whenever the mouse comes near it.
- Again we will begin with *when green flag clicked.*
- Next, we want the following in a *forever* loop: if the mouse is close to the cat, the cat should move to a random location; if we click the cat, it should make some sound.
- We can implement this as follows:



- As you can see, programming a large and complex program depends on taking very small steps. Start with a large vision and break it into manageable components. Get each piece working properly on its own, and then begin to put them together.

An Interesting Bug (38:00-44:00)

- When Oscartime was first written, it worked properly
- Complex set of code ensured that user could not pick up two sprites at the same time, even if they were both touching the mouse pointer
- But in newer version of Scratch, this code no longer functions properly
- Now, if two pieces of trash are both touching the mouse, user can pick up two pieces of trash.
- What changed in Scratch to create this bug?
- Scratch must be assuming that if two things happen within a narrow enough time window, they are simultaneous.