CPSC 301: Computing in the Life Sciences
Lecture Notes 3: Tracing, summarizing and writing Scratch code (v2)

Jessica Dawson
jqdawson@cs.ubc.ca
http://www.cs.ubc.ca/~jqdawson

University of British Columbia
Department of Computer Science

2016 W2
Admin

• **Waitlisted students**
  – sign in again today after class

• **Lab02 – Will be posted by Thursday**
  – Will have some before portions to complete ahead of time
Learning Objectives

You should be able to:

• Explain how tracing, summarizing and writing code are each useful skills for a programmer

• Given a piece of Scratch code (a full project or a single script or a sequence of instructions) step through each line of code keeping track of how that line changes the data

• Given a piece of Scratch code (a full project or a single script or a sequence of instructions) provide a summary in natural language that describes the code’s function the way humans understand
Programming

• As we have seen before, a program (sometimes called a “macro,” “procedure,” “function,” “script,” ...) describes to a computer how to perform a task
  – A program consists of “code” (like “statements,” or “blocks” in Scratch)

• The process of creating, testing, fixing, modifying and generally maintaining the program is called programming

• Typically that task will involve manipulating some data
  – How we manipulate the data depends on how we represent it

• We have already discussed a few types of Scratch data. We will discuss more data types later as we study the Python programming language

• For now, let’s discuss the basic programming skills using Scratch
Tracing, Summarizing and Writing Code

- Programmers don’t just write code; they also read and summarize their own code and code written by other people.

Three important skills that we will practice

- Trace code
  - Pretend to be the computer: step through each line, keeping track of how that line changes the data
  - Important when trying to fix broken code

- Summarizing code
  - Explain in regular human language (not a programming language) the purpose of a block of code
  - Important when trying to understand existing code

- Writing code
  - Create new code that solves a particular problem
  - Rarely done “blank screen”. In most cases, new code will refer to or modify existing code
Tracing Scratch Code

• Hand tracing: pretend that you are the computer
  – Write down any relevant data
  – Step through the blocks of a script one at a time, and do exactly what that block would do (not what you meant the block to do)

• Computer aided tracing: let Scratch do it for you
  – You can execute an individual script (a stack of blocks) by clicking on it (click again to stop)
  – You can force Scratch to pause by inserting a wait block, then look at the state of that sprite (and others)

• Both techniques have pros and cons (discuss)
Tracing Example: Dog Moving

• Scratch project “chasing the mouse” from class web site
  – Consider Dog sprite, first script

• Assume that
  – Dog is at $x = 0$, $y = 0$, direction = 90 (center stage, facing right)
  – Mouse pointer is at $x = +200$, $y = +200$

• What will happen?
  – Dog direction will change to 45 (direction of the mouse)
  – Dog will move 1 unit
  – Dog direction will not change
  – Dog will move 1 unit
  – Dog direction will not change
  – Dog will move 1 unit
  – ...

Notes 3: Summarizing, Tracing & Writing
Tracing Example: Dog Barking

• Scratch project “chasing the mouse” from class web site
  – Consider dog sprite, second script

• Assume that
  – Dog is at x = 0, y = 0, direction = 90 (center stage facing right)
  – Mouse pointer is at x = +200, y = +200 (near green flag)

What will happen?

• When flag is clicked, is the dog touching the mouse-pointer?
  No. So:
  • A little speech bubble saying "Bark!" will appear above dog for 1 second.
  • The script will wait for 1 second.
  • Is the dog touching the mouse pointer? If yes do nothing, if not repeat above.
**Tracing Example: Cat Moving**

- Scratch project “chasing” from class web site
  - Consider cat sprite, first script
- Assume that
  - Cat is at $x = 0, y = 0$, direction = 90 (center stage, facing right)
  - Mouse pointer is at $x = 200, y = 200$

What will happen?

- When the space bar is pushed then
  - Is cat touching mouse pointer? No, then:
    - Cat will move 10 steps forward (to the right). ($x = 10, y = 0$, direction = 90.)
    - Cat will turn counter clockwise 90°. ($x = 10, y = 0$, direction = 0.)
    - Cat will move 10 steps forward (up). ($x = 10, y = 10$, direction = 0.)
    - Cat will turn clockwise 90°. ($x = 10, y = 10$, direction = 90.)
    - Cat will wait 1 second.
    - Cat will move 10 steps forward (to the right). ($x = 20, y = 10$, direction = 90.)
      - Since the cat is not touching the mouse-pointer, it will repeat...
- Note: what’s the benefit of recording the data at every step?
Tracing Example: Cat Moving

- Scratch project “chasing the mouse” from class web site
  - consider combination of both scripts

- What is the state of the cat sprite after 3 seconds?

- Why is this so hard to trace?
Tracing Example: Cat Moving Extended

• Group Exercise
  – Consider the following modification to the cat script

• Assume that
  – Cat is at $x = 0$, $y = 0$, direction = 90 (center stage, facing right)
  – Mouse pointer is at $x = 200$, $y = 200$

What will happen?