CPSC 301: Computing in the Life Sciences
Lecture Notes 3:
Summarizing, Tracing & Writing Scratch Code

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

University of British Columbia
Department of Computer Science

2016 W2
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
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 \( x = +200, y = +200 \) (near green flag)

• 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?
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 \( x = 200, y = 200 \) (green flag)
What will happen?

• We can also use a table to keep track of how the data change.
Summarizing Scratch Code

• Describes the action of the code fragment at a higher level than tracing in a form the humans understand
  – A good test of understanding: if you cannot understand the code, you are unlikely to be able to translate it into human language form

• The goal is to develop an overall picture of a piece of code
  – A practical purpose is to decide whether you can reuse that code, or whether you can modify it to suit your needs

• Even in Scratch, programs can be very long
  – When explaining, break them down into appropriate chunks
  – Size of chunk will depend on the degree of detail that is desired: maybe entire sprites, entire scripts, parts of a script, but rarely individual blocks
Summarizing Example: Dog Sprite

• Consider again the two scripts from the dog sprite in Scratch project “chasing the mouse” from class web site.

• Summary of first script: After the flag is clicked, sprite will keep moving relatively slowly towards the mouse-pointer.

• Summary of second script: After the flag is clicked, sprite will keep barking for 1 second and then staying quiet for 1 second as long as it does not touch the mouse-pointer.
Summarizing Example: Cat Sprite

- Try a higher level summary of the Cat sprite in the Scratch project “chasing the mouse” from class web site

Summary: When the space key is pressed, cat will keep moving by 20 steps to the right and 10 steps up until it touches the mouse pointer.
Summarizing Example: Fur Elise

• Consider the project Fur Elise from Scratch gallery (posted on the web site)
  – Summary of the project:
    Plays the right hand theme of the first stanza (verse, part,...) of Beethoven’s Fur Elise displaying the piano keys that play the notes.
  – Summary of the first script of the Piano sprite:

  – Summary of the of the second script of the Piano sprite:
Summarizing Example: Ping Pong

• Another simple project from the Scratch gallery
  – Summarize the action of each of these scripts
  – First script:
    When the simulation starts, put the ball at a particular position. (Note that direction is not set.) Stop the simulation if the ball ever touches the colour red.
  – Second and third script
Writing Scratch Code

• When working with a new language, it is often easiest to start from something that works and modify it
  – Working from the existing code helps you learn by example

• This fact is particularly true for Scratch
  – The Scratch gallery gives you lots of examples of working code
  – The Scratch environment lends itself to experimentation

• We can view the existing code as an incorrect attempt to solve your task
  – It may (or may not) be correct for its original task

• So the modification of Scratch code to accomplish a different task is actually an example of fixing incorrect Scratch code
Summary

• Three basic programming skills are useful in any language for any task:
  • Code Tracing
    – Simulate the code and record the changes it performs on the data
    – Helps to find errors in the code
  • Code Summarization
    – Describe the code in regular human terms
    – Helps the programmer understand the code
  • Code Writing
    – We usually start from existing code and modify it
    – Tracing and summarization help here as well.