# Problem Set 0: Scratch

This is CS50. Harvard University. Fall 2013.

## Table of Contents



*standard edition only, due Fri 9/13 at noon (i.e., includes free late day)*

Questions? Head to cs50.net/discuss[1] or join classmates at office hours[2]!

## Objectives

- Introduce some fundamental programming constructs.
- Empower you to design your own animation, game, or interactive art.
- Impress your friends.

## Academic Honesty

This course's philosophy on academic honesty is best stated as "be reasonable." The course recognizes that interactions with classmates and others can facilitate mastery of the course's material. However, there remains a line between enlisting the help of another and submitting the work of another. This policy characterizes both sides of that line.

The essence of all work that you submit to this course must be your own. Collaboration on problem sets is not permitted except to the extent that you may ask classmates and others for help so long as that help does not reduce to another doing your work for you. Generally

[1] https://www.cs50.net/discuss
[2] https://www.cs50.net/ohs

speaking, when asking for help, you may show your code to others, but you may not view theirs, so long as you and they respect this policy's other constraints. Collaboration on quizzes is not permitted at all. Collaboration on the course's final project is permitted to the extent prescribed by its specification.

Below are rules of thumb that (inexhaustively) characterize acts that the course considers reasonable and not reasonable. If in doubt as to whether some act is reasonable, do not commit it until you solicit and receive approval in writing from the course's heads. Acts considered not reasonable by the course are handled harshly. If the course refers some matter to the Administrative Board and the outcome is Admonish, Probation, Requirement to Withdraw, or Recommendation to Dismiss, the course reserves the right to impose local sanctions on top of that outcome that may include an unsatisfactory or failing grade for work submitted or for the course itself.

## Reasonable

- Communicating with classmates about problem sets' problems in English (or some other spoken language).
- Discussing the course's material with others in order to understand it better.
- Helping a classmate identify a bug in his or her code at Office Hours, elsewhere, or even online, as by viewing, compiling, or running his or her code, even on your own computer.
- Incorporating snippets of code that you find online or elsewhere into your own code, provided that those snippets are not themselves solutions to assigned problems and that you cite the snippets' origins.
- Reviewing past semesters' quizzes and solutions thereto.
- Sending or showing code that you've written to someone, possibly a classmate, so that he or she might help you identify and fix a bug.
- Sharing snippets of your own code on CS50 Discuss or elsewhere so that others might help you identify and fix a bug.
- Turning to the web or elsewhere for instruction beyond the course's own, for references, and for solutions to technical difficulties, but not for outright solutions to problem set's problems or your own final project.
- Whiteboarding solutions to problem sets with others using diagrams or pseudocode but not actual code.

- Working with (and even paying) a tutor to help you with the course, provided the tutor does not do your work for you.

## Not Reasonable

- Accessing a solution in CS50 Vault to some problem prior to (re-)submitting your own.
- Asking a classmate to see his or her solution to a problem set's problem before (re-)submitting your own.
- Failing to cite (as with comments) the origins of code or techniques that you discover outside of the course's own lessons and integrate into your own work, even while respecting this policy's other constraints.
- Giving or showing to a classmate your solution to a problem set's problem when it is he or she, and not you, who is struggling to solve it.
- Looking at another individual's work during a quiz.
- Paying or offering to pay an individual for work that you may submit as (part of) your own.
- Providing or making available solutions to problem sets to individuals who might take this course in the future.
- Redeeming or attempting to redeem someone else's code for a late day.
- Searching for, soliciting, or viewing a quiz's questions or answers prior to taking the quiz.
- Searching for or soliciting outright solutions to problem sets online or elsewhere.
- Splitting a problem set's workload with another individual and combining your work.
- Submitting (after possibly modifying) the work of another individual beyond allowed snippets.
- Submitting the same or similar work to this course that you have submitted or will submit to another.
- Submitting work to this course that you intend to use outside of the course (e.g., for a job) without prior approval from the course's heads.
- Using resources during a quiz beyond those explicitly allowed in the quiz's instructions.
- Viewing another's solution to a problem set's problem and basing your own solution on it.

# Bits

- First curl up with Nate's short on binary, if not too familiar:
  http://www.youtube.com/watch?v=hacBFrgtQjQ
  And then with Nate's short on ASCII:

  http://www.youtube.com/watch?v=UPlR4eMMCmI
  Consider these questions rhetorical for now, but odds are they'll come up again!

  - How do you represent the (decimal) integer 50 in binary?
  - How many bits must be "flipped" (i.e., changed from 0 to 1 or from 1 to 0) in order to capitalize a lowercase `a` that's represented in ASCII?
  - How do you represent the (decimal) integer 50 in, oh, "hexadecimal," otherwise known as "base-16"? Know that decimal is considered "base-10" (since it employs 10 digits, 0 through 9), and binary is considered "base-2" (since it employs 2 digits, 0 and 1). Infer from those base systems how to represent base-16! (We'll see base-16 again in the context of graphics and web design.)
- Next dive into Allison's short on Scratch:
  http://www.youtube.com/watch?v=52JoFF4HMA4
  No questions on that one, though, for now!

# Itching to Program?

- Now join Zamyla for a walkthrough of this problem set, if you'd like a bit of a tour:
  http://www.youtube.com/watch?v=697pD31GCZg
- Next head to http://scratch.mit.edu/ and sign up for an account on MIT's website by clicking **Join Scratch** atop the page. Any username (that's available) is fine, but take care to remember it and your choice of password.
- Then head to http://scratch.mit.edu/help/ and take note of the resources available to you before you dive into Scratch itself. In particular, you might want to skim the Getting Started Guide[3].
- Next head to http://scratch.mit.edu/projects/266919/ to see **Scratch Scratch Revolution** by CS50 alumna Ann Chi, which Vanessa played on stage in Week 0.

[3] http://scratch.mit.edu/scratchr2/static/__1378420408__//pdfs/help/Getting-Started-Guide-Scratch2.pdf

Click the blue square above the game's top-left corner if you'd like to full-screen the user interface (UI). Then click either of the green flags. Per Ann's instructions, as soon as you hit your keyboard's spacebar, the game will begin! Feel free to procrastinate a bit. And if you'd like to try out Frogger, by CS50 alumnus Blake Walsh, head to http://scratch.mit.edu/projects/12221773/.

- If you've no experience (or comfort) whatsoever with programming, rest assured that Ann's and Blake's projects are more complex than what we expect for this first problem set. (Click **See inside** in Scratch's top-right corner to look at each project's underlying "implementation details.") But they do reveal what you can do with Scratch. And if your computer has a webcam, you might also want to try **Move the Butterfly** at http://scratch.mit.edu/projects/10016382/. Recall from Week 0 how that program utilizes a webcam to detect movement to which sprites can respond.
- Let's take a look at one other project. Head to http://scratch.mit.edu/projects/37413/ to see a project you probably haven't yet seen by Carlos Herrera. Take a moment to play the game, then click **See inside** in Scratch's top-right corner to see how it's implemented. Spend some time looking over Carlos's scripts. Don't forget that each sprite has its own set of scripts. Try to get a sense of how the overall program works. Try making some changes, even while the program is running, to see how the program responds. Note that this project is probably a bit simpler than we expect of you for this problem set, but it's a good one to learn from because it's pretty easy to follow. And do appreciate that this game, like all Scratch projects, reduces quite literally to some basic building blocks.

  Feel free to download the source code for a few more projects, either from http://scratch.mit.edu/explore/ or from Week 0 at http://scratch.mit.edu/studios/247678/, even if you already saw some of the latter in lecture. For each program, run it to see how it works overall and then look over its scripts to understand how it works underneath the hood. Feel free to make changes to scripts and observe the effects. Once you can say to yourself, "Okay, I think I get this," you're ready to proceed.
- Now it's time to choose your own adventure! Your mission is, quite simply, to have fun with Scratch and implement a project of your choice (be it an animation, a game, interactive art, or anything else), subject only to the following requirements.
  - Your project must have at least two sprites, at least one of which must resemble something other than a cat.
  - Your project must have at least three scripts total (i.e., not necessarily three per sprite).

- Your project must use at least one condition, one loop, and one variable.
- Your project must use at least one sound.
- Your project should be more complex than most of those demonstrated in lecture (many of which, though instructive, were quite short) but it can be less complex than, say, Scratch Scratch Revolution. As such, your project should probably use a few dozen puzzle pieces overall.

Feel free to peruse additional projects online for inspiration, but your own project should not be terribly similar to any of them. Try to think of an idea on your own, and then set out to implement it. But don't try to implement the entirety of your project all at once: pluck off one piece at a time. Ann, for instance, probably implemented just one arrow first, before she moved on to her game's other three. And Carlos probably implemented a stationary goal before he tried to make it move up and down on its own.

If, along the way, you find it too difficult to implement some feature, try not to fret; alter your design or work around the problem. If you set out to implement an idea that you find fun, you should not find it hard to satisfy this problem set's requirements.

Alright, off you go. Make us proud!

- Once finished with your project, click **See project page** in Scratch's top-right corner. Ensure your project has a title (in Scratch's top-left corner), some instructions (in Scratch's top-right corner), and some notes and/or credits (in Scratch's bottom-right corner). Then click **Share** in Scratch's top-right corner so that others (e.g., your TF!) can see your project. Finally, take note of the URL in your browser's address bar. That's your project's URL on MIT's website, and you'll need to know it later.
- Oh, and if you'd like to exhibit your project in Fall 2013's gallery, head to http://scratch.mit.edu/studios/248024/, then click **Add projects**, and paste in your own project's URL.

# How to Submit

- To submit this problem set, visit https://forms.cs50.net/2013/fall/psets/0/. You'll find that a few questions await. Be extra-sure that your answers are correct, particularly your project's URL on MIT's website, lest we overlook your submission!
- This was Problem Set 0.