UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

CS 39N — Dan Garcia
Fall 2009 — Brian Harvey

CS 39N: The Beauty and Joy of Computing
General Course Information

Introduction

CS39N is a pilot freshman/sophomore seminar version of a new course: *The Beauty and Joy of Computing*. Computing has changed the world in profound ways. It has opened up wonderful new ways for people to connect, design, research, play, create, and express themselves. However, just using a computer is only a small part of the picture. The real transformative and empowering experience comes when one learns how to program the computer, to translate ideas into code. This course will teach you how to do exactly that, using Scratch, one of the friendliest programming languages ever invented. It's purely graphical, which means programming involves simply dragging blocks around, and building bigger blocks out of smaller blocks

But this course is far more than just learning to program. We'll focus on some of the "Big Ideas" of computing, such as abstraction, design, recursion, concurrency, simulations, and the limits of computation. We'll show some beautiful applications of computing that have changed the world, talk about the history of computing, and where it will go in the future. Throughout the course, relevance to the students and to society will be emphasized. As an example, the final project will be your choice, on any topic interesting to you.

The overarching theme is to expose students to the beauty and joy of computing. This course is designed for non-CS majors. This course is serving as a test-the-waters 2-unit (letter-graded) pilot for a full, 4-unit course we plan to offer in the fall of 2010.

Instructors

Dan Garcia
777 Soda Hall
Phone: 642-9595
Computer Mail: ddgarcia@cs.berkeley.edu
Office Hours:
M 4-5pm in lab

Brian Harvey
781 Soda Hall
Phone: 642-8311
Computer Mail: bh@cs.berkeley.edu
Office Hours:
10:00–11:30 Wednesday,
4:00–5:00 Thursday

Office hours are primarily for short questions and administrative problems. We are happy to make appointments for longer periods of time if you feel you need it. Please don't be shy; we'd rather see you as soon as you don't understand something than right before the exam. It's not an imposition—we *like* teaching. Please don't ask us about individual *administrative* problems in lecture.

Do You Belong Here?

This course is intended for students in non-technical majors, with little or no background in computer programming, who are curious about what lies behind Google and Facebook and Twitter and the iPod. The purpose is not directly to prepare you for a career in computer technology, although if as a result of this course you want to pursue computer science further, we encourage you to take CS 61A, which is the first course for CS majors. This course combines some programming experience with discussion of the big, beautiful ideas of computer science.

Another introductory CS course for students without programming experience is CS 3, which comes in two versions: CS 3L is built around scheduled lab sessions, while CS 3S is the self-paced version with no scheduled meetings. CS 3 is more focused on programming than this course, and it uses Scheme, the same programming language as CS 61A. If you already know that you want to take further coursework in CS, you might prefer to start with CS 3 instead.

If you already have previous programming experience that includes writing recursive functions, then you may want to start right off with CS 61A. (Some students, hearing that 61A is the first course for CS majors, think that they have enough experience to skip it. This is very rarely true; previous experience might get you out of CS 61B.)

Course Materials

There is no textbook for this course. We'll hand out readings or post them on our web page from time to time.

We'll be using the Scratch graphical programming language. You can download Scratch for your home computer on the web:

`http://scratch.mit.edu/download`

A little later in the course we'll also use BYOB (Build Your Own Blocks), an experimental, advanced version of Scratch, and we'll provide a download link for that version.

Weekly Schedule

The class meets Monday and Wednesday, 3–5pm, in 200 Sutardja Dai Hall (the new CITRIS building). Ordinarily, Monday will be an hour of lecture and sometimes an hour of discussion, with lab activities all of Wednesday. (It's "sometimes" because four class hours per week is too much for a two-unit course, so discussion will meet roughly every other week, except that weeks with holidays may shift the schedule. In non-discussion weeks we'll hold office hours in the same room Monday 4-5.) But the fact that we're always in the same room gives us some flexibility, if either the nature of the week's topic or a holiday schedule makes some other allocation of the time preferable.

During discussion and lecture times, please don't get distracted by the fact that there's a computer in front of you; stay focused on the activity!

Weekly homework, which will include Scratch programming activities, readings, and sometimes short writing assignments, will be due by the beginning of the first class of the week (Monday, unless Monday is a holiday that week). Since you can't easily turn in Scratch programs on paper, you'll upload your homework to the relevant homework folder within the Assignments tab in bSpace:

`http://bspace.berkeley.edu`

Information Resources

Your first and most important resource for help in learning the material in this course is your fellow students. In the lab, you'll work in pairs, alternating turns with one person at the keyboard and the other helping plan and validate the logic of the program as well as think about the big picture design and goals. Ask your partner! If s/he doesn't know the answer, ask the teams next to you.

We also have a staff consisting of TAs Colleen Lewis and George Wang; readers (and TAs-in-training) Angela Juang and Audris Chiang; and lab assistants not yet recruited as of this writing. Colleen, an experienced Scratch instructor, is the czar of lab time.

During lab, ask any of the staff for help, *after* you've tried asking other students. (It's not that we're lazy; it's that answering each others' questions is a great way to learn.)

During lecture or discussion, if you think your question might be relevant to other students, raise your hand and ask it. Don't be afraid to ask "stupid questions"; there's no such thing. For individual administrative questions, see below.

Buttonholing the instructors before or after class to ask your question *isn't* a good idea. For one thing, it means other students don't get the benefit of your question, and for another, both Dan and Brian have incredibly tight schedules this semester.

If you want to ask a question outside of class time, first decide whether it's an *administrative* question or a *content* question. For administrative questions about your specific situation, come to our office hours, or try email:

Brian Harvey	`bh@cs.berkeley.edu`
Colleen Lewis	`cs39n-tb@inst.eecs.berkeley.edu`
George Wang	`cs39n-tc@inst.eecs.berkeley.edu`
Angela Juang	`cs39n-rb@inst.eecs.berkeley.edu`
Audris Chiang	`cs39n-rc@inst.eecs.berkeley.edu`

(Dan isn't in this list because he prefers that you ask him administrative questions in office hours.)

If what you have is a *content* question (about Scratch, or about some big idea in computer science), try asking it on the class bSpace forum page. That way, other students can both answer your question and read the answers if they have the same question:

`http://bspace.berkeley.edu`

Sample content question: "I want to learn more, but don't want to be a CS major. What class should I take next?" Sample administrative question: "I'm on the badminton team, and we have an away game the night of the midterm. May I reschedule it?"

The course web site at

`http://inst.eecs.berkeley.edu/c̃s39n`

will have the assignments, lecture notes, etc., as we create them. (This is the first time around for this course, so we're just barely ahead of you!)

Lots of information about Scratch is available at the Scratch web site:

`http://scratch.mit.edu`

Computer Resources

Our computer lab consists of 30 Apple Mac Pro workstations, in room 200 Sutardja Dai Hall. (Don't worry if your home computer isn't a Mac; the software we're using runs identically on all platforms.)

The lab will be available for use at all times, once they get the card key issues in the new building settled. Meanwhile, the computers in 330 Soda run Scratch, so you can use those. You'll need a card key for evening access to either lab. To gain card key access to Sutardja Dai Hall and the instructional labs, you must go to 387 Soda Hall to complete the appropriate form to get your CAL ID proximity card or white card key activated (if never activated in previous semesters) or re-activated (if activated in previous semesters). There is **no charge** to get your CAL ID activated for card key access to Sutardja Dai Hall. If you *do not* have a CAL ID proximity card, please go to the CAL ID office located in Lower Sproul Plaza and get a replacement at no charge.

Weekly Assignments

Most weeks there will be problems assigned for you to work on, most of which will involve writing and debugging computer programs. You'll work on some of these problems individually, but most in pairs. These weekly assignments come in three categories:

- **Laboratory exercises** are short, relatively simple exercises designed to introduce a new topic. Most weeks you'll do these during the scheduled lab meetings Wednesday, in groups of two students.

- **Weekly quizzes** are short, usually one question, individual written exercises at the beginning of each lab session. Their purpose is to assure us and yourself that you are keeping up to speed on learning the ideas of the course.

- **Homework assignments** are usually a continuation of the programming topic introduced in lab, and generally you'll do them in pairs. There will also usually be some reading material handed out in lecture that should be read before the following lecture.

Sometimes you will also be asked to write a short (less than a page) paper responding to the reading and the class discussion; these will generally be done individually.

Some weekly assignments may include material labeled as "Extra for Experts." These problems or readings are entirely optional; do them only if you have finished the regular assignment and want to do something more challenging. There is no extra *credit* for these problems; people who need more credit shouldn't even be trying them, and people who are doing well in the course should be motivated by the desire to learn.

The purpose of the lab and homework is for you to learn the course content, not to prove that you already know it. Therefore, those assignments are not graded on the correctness of your solutions, but on effort. **You will get full credit for an entirely wrong answer that shows reasonable effort!** You will get *negative* points for a solution copied from someone else.

Longer Assignments

- **Projects** are larger assignments intended both to teach you the skill of developing a large program and to assess your understanding of the course material. There are two projects during the semester. The programming projects *are* graded on the quality of your work.

- There will also be a **term paper** that will let you explore in more depth some of the big ideas of computer science presented in the lectures, discussions, and reading.

Lost and Found

When people bring us found items from the classroom, we'll take them to the Computer Science office, 387 Soda. Another place to check for lost items is the campus police office in Sproul Hall.

Testing and Grading

If it were up to us, we wouldn't give grades at all. Since we can't do that, the grading policy of the course has these goals: It should encourage you to do the course work and reward reasonable effort with reasonable grades; it should minimize competitiveness and grade pressure, so that you can focus instead on the intellectual content of the course; and it should minimize the time we spend arguing with students about their grades. To meet these goals, your course grade is computed using a point system with a total of 400 points:

```
midterm exam        60                 final exam              100
midterm project     60                 final project            60
term paper          60              15 weekly grades   15 * 4   60
```

The weekly grades are based on class participation, homework, and very short weekly quizzes (except for this first week).

Each letter grade corresponds to a range of point scores:

A+	390-400	A	370-389	A-	360-369
B+	350-359	B	330-349	B-	320-329
C+	310-319	C	290-309	C-	280-289
D	230-279	F	0-239		

A typical C– (barely passing) student would be one who gets 60% scores on exams (90 points), 75% on projects and paper (105 points), and 90% on weekly grades (55 points).

Incomplete grades will be granted only for dire medical or personal emergencies that cause you to miss the final, and only if your work up to that point has been satisfactory.

Cooperative Learning Policy

With the obvious exception of exams, we *encourage* you to discuss *all* of the course activities with your friends as you are working on them.

Each pair of students must solve the weekly homework assignments separately; **it is by struggling with the homework that you learn the course material!** However, before you develop your own solution to each problem you are encouraged to discuss it with other students, in groups as large or small as you like. **When you turn in your solution, you must give credit to any other student(s) who contributed to your work.** Working on the homework in groups is both a good way to learn and a lot more fun! Although the homework is graded on effort rather than on correctness, if you take the opportunity to discuss the homework with other students then you'll probably solve every problem correctly.

The lab activities will also be done in pairs. Feel free also to ask questions of students outside your pair, if you're stuck, but don't let them do the work for you.

Parts of the midterm exams will be done in groups of four; this will be explained further when the time comes.

Working cooperatively in groups is a change from the traditional approach in schools, in which students work either in isolation or in competition. But cooperative learning has become increasingly popular as educational research has demonstrated its effectiveness. One advantage of cooperative learning is that it allows us to give intense assignments, from which you'll learn a great deal, while limiting the workload for each individual student. Another advantage, of course, is that it helps you to understand new ideas when you discuss them with other people. Even if you are the most experienced person in your group, you'll find that you learn a lot by discussing the course with other students. For example, in the past (in other courses) some of our best students have commented that they didn't *really* understand the course until they worked as lab assistants and had to explain the ideas to later students.

If some medical or personal emergency takes you away from the course for an extended period, or if you decide to drop the course for any reason, please don't just disappear silently! You should inform your lab partner and the staff, so that nobody is depending on you to do something you can't finish.

Unlike the homework and projects, the tests in this course (except for the parts specifically designated as group parts) must be your own, individual work. We hope that you will work cooperatively with your friends *before* the test to help each other prepare by learning the ideas and skills in the course. But during the test you're on your own. The EECS Department Policy on Academic Dishonesty says, "Copying all or part of another person's work, or using reference materials not specifically allowed, are forms of cheating and will not be tolerated." The policy statement goes on to explain the penalties for cheating, which range from a zero grade for the test up to dismissal from the University, for a second offense.

In our experience, nobody begins the semester with the intention of cheating. Students who cheat do so because they fall behind gradually, and then panic at the last minute. Some students get into this situation because they are afraid of an unpleasant conversation with an instructor if they admit to not understanding something. We would much rather deal with your misunderstanding *early* than deal with its consequences later. Even if the problem is that you spent the weekend goofing off instead of doing your homework, please overcome your guilt feelings and **ask for help as soon as you need it.**

Lateness

A programming project or term paper that is not ready by the deadline may be turned in until 3pm Tuesday (the day after the due date). These late assignments will count for 2/3 of the earned score. No credit will be given for late homeworks, or for larger assignments turned in after Tuesday. Please do not beg and plead for exceptions. If some personal crisis disrupts your schedule one week, don't waste your time and ours by trying to fake it; just be sure you do the next week's work on time.

Questions and Answers

Q: I am disabled and need special facilities or arrangements to do the course work. What should I do about it?

A: The Disabled Students Program (DSP, ext. 2-0518) certifies students as having special needs; DSP students are entitled to the necessary accomodations in course arrangements; the DSP office will give you a letter to bring to us. Please take the initiative about letting us know what you need. For example, if you are qualified to take tests separately, that's fine, as long as you ensure that we've worked out the arrangements before the test. **If English is not your native language,** and you have trouble understanding the course materials or lectures for that reason, please ask for help about that too.

Q: I'm thinking about buying a personal computer. What do you recommend?

A: For this course, and in general for computer science courses at Berkeley, you don't *need* a computer of your own at all; you can work in the labs on campus. If you just want to be able to connect to the campus computers from home, anything with an Internet connection will do. (If you live in certain dorms, there is an Ethernet connection in your room, and having a computer with an Ethernet adaptor will be very handy.) If you want to work entirely within your home computer, you can get Scratch for PC-compatibles or MacOS X.

Some of our students, especially the ones with a particular interest in system administration, choose to run one of the free versions of Unix at home, either Linux or FreeBSD. This takes more effort than using commercial systems, but you learn a lot in the process. MacOS X is an interesting compromise, with BSD Unix hidden behind a graphical user interface.

Q: My project partner never does any work. What should I do about it?

A: First of all, try to find out why. Sometimes people give up because they're having trouble understanding something. If that's the problem, see if you can teach your partner and get him or her back on track. Also, try to find out what his or her *strengths* are—how he or she can best contribute to the group's efforts. But sometimes people get distracted from coursework for non-academic reasons. If you can't resolve the problem yourselves, talk with a TA. You'll be rotating partners between assignments in any case, so it won't be a long-term problem.

Q: What should we call you?

A: If you're being formal, or don't like the course, "Dr. Garcia" or "Dr. Harvey." Otherwise, "Dan" or "Brian."

Q: I forgot my password. (Or: It won't let me log in.)

Go to the Instructional sys admin staff in 333 Soda, 378 Cory or 386 Cory. Bring your initial class account form or student ID card. Don't ask for another account.

Tentative Topic Outline and Assignment Schedule

The up-to-the-minute version of this chart will be on the class web page.

Underlined Monday dates have discussion meeting 4–5pm.

week	Monday	Wednesday	big idea	Scratch
Abstraction				
1	**holiday**	8/26	abstraction	broadcast, variables, motion
Graphics and Artificial Intelligence in Games				
2	8/31	9/2	video games	random, if, question/answer
3	**holiday**	9/9	strategy games	lists
4	9/14	9/16	artificial intelligence	design tradeoffs
Programming Paradigms				
5	9/21	9/23	functional programming	BYOB, functions
6	9/28	9/30	other paradigms	more functions (start project)
Applications and Their Social Implications				
7	10/5	10/7	great applications	work on project
8	10/12	10/14	privacy	work on project
			first project due 10/19	
9	10/19	10/21	risks	(start paper)
Recursion				
			midterm 10/26 7–9pm	
10	10/26	10/28	recursion	recursion
11	11/2	11/4	recursive patterns	recursion
			term paper due 11/9	
12	11/9	**holiday**	advanced recursion	(start project)
Limits and Future of Computation				
13	11/16	11/18	computability	work on project
14	11/23	11/25	future of computing	work on project
			second project due 11/30	
Summary				
15	11/30	12/2	project presentations	
			Final Tuesday, 12/15, 12:30–3:30pm	