Introduction

Day 1, Session 1
Welcome!
MyCS: Middle-years' Computer Science

July 9 - July 13, 2012
• **Housekeeping**: where? when?
• **Introductions**: who?
• **Welcome**: what? (1) CS, (2) ECS, and (3) MyCS

• Account information, forms, binders, etc.

• Dive in!
### Housekeeping and Schedule

**Restrooms?** ~ SW corner

#### Schedule

<table>
<thead>
<tr>
<th>Session 1: 9am-10:30am</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction: Computers?! Finding facts</td>
<td>Introduction to computational problem solving</td>
<td>Algorithms: searching and sorting them out</td>
<td>Using the web: social responsibility</td>
<td>Wrap-up and looking forward</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Session 2: 10:45am-morning</th>
<th>Session 3: 1pm-2:15pm</th>
<th>Session 4: 2:30pm-4pm</th>
<th>Lunch</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to programming: Making music with Scratch</td>
<td>Computers: Getting the inside view</td>
<td>Scratch: Movement and debugging</td>
<td>noon to 1pm</td>
<td>field trip</td>
<td>field trip</td>
</tr>
<tr>
<td>Scratch: Position (and more debugging!)</td>
<td>Bits and binary... and Lego</td>
<td>Scratch: Images</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scratch: Ifs and Elses</td>
<td>Limits of computing: AI, 20 questions, and life</td>
<td>Scratch: Broadcasting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Creating the web: HTML and CSS</td>
<td>Security and passwords</td>
<td>Javascript and final recipes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Introductions? ~ coming next...
Introductions

- Name tags...
- Our own introductions...
- Saying hello to our pairs around the room...
- Please take the "Welcome!" survey

@ Google docs: login 2012mycs passwd csisfun!

from the 90's:

Mike Erlinger  Zach Dodds  The team!
CS, ECS, and MyCS

Middle-years Computer Science

Exploring Computer Science

Computer Science
We are in the era of Computer Science.

Two of the fastest growing detailed occupations are in the computer specialist occupational group. Network systems and data communications analysts are projected to be the second-fastest-growing occupation in the economy. Demand for these workers will increase as organizations continue to upgrade their information technology capacity and incorporate the newest technologies. The growing reliance on wireless networks will result in a need for more network systems and data communications analysts as well. Computer applications software engineers also are expected to grow rapidly from 2008 to 2018. Expanding Internet technologies have spurred demand for these workers, who

http://www.bls.gov/oco/ocos303.htm
CS, ECS, and MyCS

CS is exciting - and it needs to be!

TBD: Who will consume vs. create it?

Choose a role: consumption vs. contribution

it's not really vs.

January, 2012 at the Victoria Gardens Apple Store
CS, ECS, and MyCS

Course Overview

Goals

Exploring Computer Science is designed to introduce students to the breadth of the field of computer science through an exploration of engaging and accessible topics. Rather than focusing the entire course on learning particular software tools or programming languages, the course is designed to focus the conceptual ideas of computing and help students understand why certain tools or languages might be utilized to solve particular problems. The goal of Exploring Computer Science is to develop in students the computational thinking practices of algorithm development, problem solving and programming within the context of problems that are relevant to the lives of today's students. Students will also be introduced to topics such as interface design, limits of computation, and the role of computers in society.

ECS's Goal: to democratize CS

Everyone should have a chance to help author and understand the computation we use everyday.
CS, ECS, and MyCS

LA Unified students taking ECS:

- 2009: 306
- 2010: 922
- 2011: 1377
- 2012: 2000+

Other cities starting, as well...

We are working to bring this to the I.E. -- and PUSD in particular, only a bit earlier
MyCS is an ECS-inspired course developed for middle-school and "middle-years" audiences.

- engaging CS ideas and activities
- hands-on skills, emphasizing creativity
- can be spliced with technology courses, typing courses, and other existing electives

ECS curriculum with emphasis on skills
we are looking for what will work in one term
we'd love your help creating a usable MyCS!
Exploring Computer Science

ECS 4-5 week units

Non-programming

What is CS? ~ computers and computing

Data and algorithms ~ the heart of CS

Programming

The web ~ web pages, and web applications

Scratch ~ programming with sound and graphics

2 modules of problem-solving

2 modules of programming

18 weeks total
Exploring Computer Science

**ECS**

4-5 week units

---

Non-programming

**What is CS?** ~ computers and computing

**Data and algorithms** ~ the heart of CS

---

Programming

**The web** ~ web pages, and web applications

**Scratch** ~ programming with sound and graphics

---

*Is this "real"?*

2 modules of problem-solving

*too abstract...*

Aargh!

2 modules of programming

*smoother Scratch?*

**Scratch > Web**

18 weeks total
ECS  4-5 week units

Non-programming

What is CS? ~ computers and computing

Data and algorithms ~ the heart of CS

Programming

The web ~ web pages, and web applications

Scratch ~ programming with sound and graphics

MyCS  4-7 days spiraling

CS: what and why?

Scratch ~ sound

What is a computer?

Scratch ~ movement

CS as problem-solving

Encoding data

Scratch ~ coordinates

Algorithms ?!

Scratch ~ if/else

Limits of computers

Scratch ~ projects

Using the web well

Web: HTML/CSS

Web safety/security

Web: Javascript

Games and extras
~15 modules interleaving problem-solving and programming

18 weeks total
Two 18-week schedules

Exploring Computer Science

ECS
4-5 week units

Non-programming

What is CS? ~ computers and computing

Data and algorithms ~ the heart of CS

Programming

The web ~ web pages, and web applications

Scratch ~ programming with sound and graphics

Our 5-day workshop

4-7 days spiraling

MyCS
Middle-years Computer Science

CS: what and why?

Day 1

Scratch ~ sound

What is a computer?

Scratch ~ movement

CS as problem-solving

Day 2

Encoding data

Scratch ~ coordinates

Algorithms ?!

Day 3

Scratch ~ if/else

Limits of computers

Scratch ~ projects

Using the web well

Day 4

Web: HTML/CSS

Web safety/security

Web: Javascript

Day 5

Games and extras
ECS 4-5 week units

Non-programming

What is CS? ~ computers and computing

Data and algorithms ~ the heart of CS

Programming

The web ~ web pages, and web applications

Scratch ~ programming with sound and graphics

MyCS 4-7 days spiraling

Our 5-day workshop

Day 1

Scratch ~ sound

What is a computer?

Day 2

Scratch ~ movement

CS as problem-solving

Encoding data

Day 3

Scratch ~ coordinates

Algorithms ?!

Day 4

Scratch ~ if/else

Limits of computers

Scratch ~ projects

Using the web well

Day 5

Web: HTML/CSS

Web safety/security

Web: Javascript

Games and extras
Natural place to start:

What is computing?

What is a computer?
(1) For each of these items, decide if it's a computer or not.

- Computer
  - Not a computer

- Computer
  - Not a computer

- Computer
  - Not a computer

- Computer
  - Not a computer

- Computer
  - Not a computer

- Computer
  - Not a computer

- Computer
  - Not a computer
Computer? Or, not a computer?
What is a computer / computing?

First, think of two other examples of computers - be creative!

1. 
2. 

Also, think of two other examples of non-computers

1. 
2. 

Now, list three traits or characteristics that make something a "computer":

1. 
2. 
3. 

And -- list three traits that make something not a computer!

1. 
2. 
3.
CS and life?

Are there any connections there?
CS and life...

... and our sponsor!
Web Search

certainly sets the standard for "information interactions"

especially freeing in education

Giving fish vs. teaching fishing!
Web Skills

What are some of the strategies you use to make web-searching fast & effective?

*let's return to this question -- after you try things out!*
Web Scavenger Hunt

With your partner, find answers to these challenge questions...

For the later ones, estimate how difficult you think the question is (both before & after)
Web Scavenger Hunt

Inspired by a Google a Day

Starter Questions:
1. How many results does your name get on Google? ________ on Bing? ________
2. What animal represents your birth year in the Chinese zodiac? __________
3. Which birthday is more common, yours or your partner’s? __________
4. Using google trends, during when is American Idol most popular? __________
5. Using google trends, what language is most used to discuss One Direction? __________

Challenge Questions: For these challenges, before you begin, analyze the question and mark down how hard you think it will be to find the answer, on a scale from 1-10 (1-easy, 10-hard). After you find the answer, mark down how hard it actually was.

<table>
<thead>
<tr>
<th>Hardness Prediction</th>
<th>Question</th>
<th>Answer</th>
<th>How hard was it actually?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>How many days would it take you to walk from Claremont, CA to Claremont, NH?</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you were in the world’s largest annual food fight, what color would your clothes end up?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Google-a-Day

Sample Scavenger Hunt
In your group, use the Internet to find the following items. For each item:

1. A picture of the mayor of your town or city
2. A bus schedule
3. The address of the Chamber of Commerce for your town or city
4. A map of your state—and you have to point out where your town is
5. A copy of the front page of your town’s or city’s web site
6. Something in writing that tells how many people live in the city
7. A picture of any historical landmark in the city
8. A picture of your congressman
9. A program or flyer from a local arts event
10. The names of all the city council members
11. Something that gives information about your local hospital
12. A list of schools in your town or city
13. The phone number of the local police department
14. Anything with the colors or mascot of a local college or community
15. A picture of the state flag
16. A picture of the state bird
17. A schedule of activities or a pamphlet from a local nursing home
18. A sticker or button from a local election
19. A list of safety tips from the local fire department
20. A speech by your governor

ECS's web challenge
Web Skills

What are some of the strategies you use to make web-searching fast & effective?

are your students already good at this -- or is this something worth reinforcing?
CS + life: *other connections?*

What else do your students already do that is determined or dominated by computer science?
Intro to Programming

Scratch: Music

Day 1, Session 2
Computer Programming

- Programming is the *stereotypical* CS activity.
- But CS is really **not** about screenfuls of crazy text...

```
PLEASE DO ,1 <-- #13
DO ,1 SUB #1 <-- #238
DO ,1 SUB #2 <-- #112
DO ,1 SUB #3 <-- #112
DO ,1 SUB #4 <-- #0
DO ,1 SUB #5 <-- #64
DO ,1 SUB #6 <-- #238
DO ,1 SUB #7 <-- #26
DO ,1 SUB #8 <-- #248
DO ,1 SUB #9 <-- #168
DO ,1 SUB #10 <-- #24
DO ,1 SUB #11 <-- #16
DO ,1 SUB #12 <-- #158
DO ,1 SUB #13 <-- #52
PLEASE READ OUT ,1
PLEASE GIVE UP
```

**Intercal**

```

v
>"Hello world!"0<v
,:^_25*,@
```

**Befunge**

What *popular* programming languages have you used/heard of?
Programming languages

- What's popular now? (tiobe.com)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>↑</td>
<td>C</td>
<td>17.725%</td>
<td>+1.45%</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>↓</td>
<td>Java</td>
<td>16.265%</td>
<td>-2.32%</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>=</td>
<td>C++</td>
<td>9.358%</td>
<td>-0.47%</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>↑↑↑</td>
<td>Objective-C</td>
<td>9.094%</td>
<td>+4.66%</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>↓</td>
<td>C#</td>
<td>7.026%</td>
<td>+0.18%</td>
<td>A</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>=</td>
<td>(Visual) Basic</td>
<td>6.047%</td>
<td>+1.32%</td>
<td>A</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>↓↓</td>
<td>PHP</td>
<td>5.287%</td>
<td>-1.31%</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>=</td>
<td>Python</td>
<td>3.848%</td>
<td>-0.05%</td>
<td>A</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>=</td>
<td>Perl</td>
<td>2.221%</td>
<td>-0.09%</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>↑↑</td>
<td>Ruby</td>
<td>1.683%</td>
<td>+0.20%</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>=</td>
<td>JavaScript</td>
<td>1.474%</td>
<td>-0.03%</td>
<td>A</td>
</tr>
</tbody>
</table>
Intro to Computer Programming

Here are two more "Hello, World!" programs:

Python

```python
print 'Hello, World!'
```

C++

```cpp
int main()
{
    std::cout << "Hello, world!";
}
```

CS is fundamentally about problem-solving...
... and programming is just its language

Programming is much more like learning a new human language than many suspect: it's just translation, but easier!
Programming languages

- What does the future of programming look like?

Graphical interfaces are ever more popular in education and industry.
Programming languages

- A "Hello, Kitty" Android Program:

What are these bricks saying?
Programming *ideas*

- *Actions* are the basic building blocks (literally!)
- The data can be simple.
- The actions can be simple.
- The combinations can be powerful!
Scratch!

- Developed at MIT
- Helps anyone learn to program without worrying about syntax
- Its **graphical interface** includes a set of blocks that snap together
**Lingo**

**Block tabs**
Click on these to get different sets of blocks.

**Blocks**
Individual instructions for the sprite. Can be combined to make scripts.

**Script**
A set of instructions for a sprite to follow.

**Sprite**
A picture of a character or object that can be controlled.

**Stage**
The place where the sprites perform your scripts.
The play note block plays a **musical** note. It has 2 **variables**:

1. You can change the **note**.
2. You can change how **long** the note plays (in beats).
Drums!

The play note block plays a **percussion** note. It has 2 variables:

1. You can change the **instrument**.
2. You can change how **long** the note plays (in beats).
Rests!

The rest block makes the script **pause**. It has 1 variable:

1. You can change how long the pause **lasts** (in beats).
Make a song!

Use notes, drums, and rests to create a short song.

When you are done, make sure to save your program -- *saving to the desktop is OK*

Then, have the other partner make a second song -- and play both songs for a nearby group!
Repeats!

The repeat block tells the computer to do the same thing *over and over* again. It has 1 variable:

1. You can change how many times to repeat the **blocks** inside the repeat block.
Repeats, continued!

Other blocks can be placed inside of the repeat block. Everything inside the repeat block will be repeated as many times as you want.
Nested Repeats!

A nested repeat is a repeat block placed inside another repeat block.
Nested Repeats, continued!

Nested repeats multiply together.
1. This drum note will play 6 times.
Tempo!

The tempo block increases the **speed** at which notes play. It has 1 variable:

1. You can change how many beats play **per minute**.
Review: Set Tempo!

The tempo block sets the speed at which notes play. It has 1 variable:

1. You can change how many beats play per minute.

When using the set tempo block, it does not matter what the current tempo is. Set tempo will set the new tempo to the number you type in.
Change Tempo!

The change tempo block *increases* or *decreases* the tempo by whatever amount you want. It has 1 variable:

1. You can change how much to increase or decrease the tempo. You may use *positive* or *negative* numbers.

When using the change tempo block, it *does matter* what the current tempo is. Set tempo will *add* or *subtract* from the current tempo.
Volume works the same way!

1. Use the **checkbox** next to the “volume” block to always see the volume.

2. Use the **set volume** block to reset the volume to the level you want.

3. Use the **change volume** block to increase or decrease the volume by a certain amount.
Song Improvements

Use repeats, tempo, and volume to make improvements to your songs or create a new one...

... either way, explore all of the tools!
“Mapping” Commands to the Keyboard

The “when key pressed” block is a control block. It runs a script when a button on the keyboard is pressed. It has 1 variable:

1. You can change which key will start the script.

The “when key pressed” block only works at the beginning of a script. (You can think of it as a “hat” for the script.)
Mapping Commands to the Keyboard!

To control a script with the keyboard, place a “when key pressed” block on top of it and choose which key will run the script.

When a key on the keyboard starts a script, we say that script has been “mapped” to that key.
Keyboard Challenge

In a new Scratch project, create a **keyboard**.

That is, make each key in a row of keys on your computer keyboard correspond to a note in a scale.  
*Want more? Try different instruments!*

*The MyCS curriculum starts with music -- would that work? How could it be sold more effectively?*
Where we are...

after lunch

Day 2

Day 3

Day 4

Day 5

- CS: what and why?
- Scratch ~ sound
- Scratch ~ movement
- Scratch ~ coordinates
- Algorithms ?!
- Scratch ~ if/else
- Limits of computers
- Scratch ~ projects
- Using the web well
- Web: HTML/CSS
- Web safety/security
- Web: Javascript
- Games and extras
Computer Disassembly

Day 1, Session 3
Behind the "curtain"...

Computers can seem like magic...
- and not necessarily benign magic, at that!
- it's unclear what's happening inside
- leads to anthropomorphizing
- but they're the opposite of us...

they're 100% understandable

Friend, foe, or fork?
Computer Disassembly

Use a search engine to look up the parts of a computer - while taking it apart.

Create a poster with the parts and (briefly) what each part does.

Here's an example...
...you should get something like these computer entrails...
1) Find these components in your computer
2) Remove and/or identify each
3) Label each with **what it does** on your poster

- Memory (RAM)
- Hard Drive
- Transistor
- Motherboard
- Power Supply
- Capacitor
- Processor (CPU)
- CD or DVD Drive
- Graphics Card
- LED
- Heat Sink
- Crystal Oscillator

4) **add two more part names that you find (or know)**
5) **Finally, take a picture and upload it to our Google doc!**
Other parts of computers that you won't find *in* the computer

- Speakers
- Monitor
- Modem
- Keyboard
- Operating System
- Mouse

Which one of these is not like the others?
How do these fit together? (an abstract picture)

If we make an internet search, in what order would these parts be involved in that process?

___ Keyboard
___ Monitor: show results
___ Graphics Card
___ Memory
___ Processor
___ Network cable or wifi
___ Hard Drive
One answer...

If we make an internet search, in what order would these parts be involved in that process?
True or False?

___ If I get a computer with a better processor, I'll be able to download faster.

___ The larger my processor's clock speed, the faster my computer will be.

___ The more space my computer has, the faster it'll run a lot of programs at once.

___ If I add another processor to this computer, it will run Minecraft with less lag.

___ If I'm running out of space on my computer, I need to get more memory.
Scratch: Movement

Day 1, Session 4
The move block makes the sprite move **forward**. It has 1 variable:

1. You can change how many **steps** the sprite takes.

The stage is **480** steps wide (-240 to +240) and **360** steps tall (-180 to +180).

The sprite will move in the direction it is **facing**.
Turn!

The turn blocks make the sprite turn **clockwise** or **counterclockwise**. They each have 1 variable:

1. You can change how many **degrees** the sprite turns.
Movement Practice

Can you make your sprite move in a triangle?

In a square?
Pen down!

Pen blocks can be found in the pen **tab**.

The pen down block makes the sprite **draw a line** as it moves.

The sprite will continue to draw a line until you have it lift the pen back **up**.
Pen up!

The pen up block makes the sprite **stop** drawing lines as it moves.

This does NOT make the lines you’ve already drawn **disappear**.
Clear!

The clear block **erases** all of the pen lines that have already been drawn.
Set pen color!

The set pen color block changes the color of **the line** the sprite draws when the pen is down.

*Clicking on the color box will let you choose the color you want from a palette.*
Set pen size!

The set pen size block changes the **thickness** of the line the sprite draws as it moves. It has 1 variable:

1. You can change how thick the line is. A **bigger** number makes a wider line.
Show and Hide!

If you do not want to see the sprite on the stage, use the hide block to make it **invisible**. The sprite will still move and draw lines while it is invisible.

If you want to make the sprite **visible** again, use the show block.

Both the show and hide blocks can be found in the “**Looks**” tab.
Making a New Variable!

To make a new variable:
1. Click on the “Variables” tab
2. Click “Make a variable”
3. Give your variable a name
Making a New Variable, continued!

You now have **new blocks** to use with your variable!

You can add your variable to **any block** that has a variable.
Using Your Variable!

You can change your variable using the `set variable` and `change variable` blocks, just like you did with tempo and volume.

Remember to use the `checkbox` to keep track of what your variable is at any time.
Choose Your Challenges

Pick a challenge from the worksheets and implement it.
A **bug** is a **mistake** in a computer program that makes it not work the way it is **supposed to**.

If a computer program (or **script**) doesn’t work the way it is supposed to, that is almost always because of **human error**.
**Debugging** is the process of **fixing** the bugs in a computer program so that it works the way it is supposed to.
Using Single Stepping

If your script, or program, isn’t working the way you expect it to, a good debugging strategy can be to run the script in “single stepping” mode. This will cause Scratch to run the script more slowly and highlight each block as it is run. This can help you see which part of the script is causing the problem.

To start single stepping, go to the Edit menu and select “Start Single Stepping”.
Debugging Practice

Use the worksheet to practice debugging.
Scratch: *thoughts*?

Could Scratch be a compelling introduction to programming for middle-school students?
Scratch: *thoughts*?

Could Scratch be a compelling introduction to programming for middle-school students?

DIGITAL DOMAIN

**Computer Science for the Rest of Us**

By RANDALL STROSS

Published: March 31, 2012

READING, writing and — refactoring code?

Marie desJardins, a computer science professor at the University of Maryland, Baltimore County, says her department uses Scratch in its “Introduction to Computers and Programming” course, in which students can try a few basic concepts. About 25 percent of the semester is spent on programming.

Explaining why Scratch is used at the college level, she says that all students arrive on campus having taken high school classes in English, math, biology and so on, but that many have not taken a computer science class.

Michael Littman, who leads the computer science department at Rutgers University, agrees. “Computational thinking should have been covered in middle school, and it isn’t,” he says. “So we in the C.S. department must offer the equivalent of a remedial course.”

It's increasingly popular in colleges' CS1 courses...!
Thanks & see you tomorrow!

Tomorrow's plan

Day 1
- CS: what and why?
- Scratch ~ sound
- What is a computer?
- Scratch ~ movement
- CS as problem-solving

Day 2
- Scratch ~ coordinates
- Algorithms ?!
- Scratch ~ if/else

Day 3
- Limits of computers
- Scratch ~ projects
- Using the web well

Day 4
- Web: HTML/CSS
- Web safety/security

Day 5
- Web: Javascript
- Games and extras