Anjuli Kannan

Google Earth Driving Simulators (3:00-7:00)

- An example of what you can do by learning the GoogleEarth API, once you know how to write code
- Google has published such an API so that people can make programs for their products without having to implement every single detail themselves
- Check out Milktruck linked from Lecture page of website

Introduction to Scratch (7:00-11:30)

- Scratch is a program you can use to make videos, games, animations, and interactive art
- You create a project in Scratch by dragging puzzle pieces from the box on the left into the palette in the center
- The puzzle pieces string together to perform a logical sequence of instructions for the sprite (by default a cat) to follow
- Oscartime (linked from the Lectures page of the website) is an example of something you can build

O Hai, C! (12:30-15:30)

- Here's a simple program in C (Slide 50)
- We could explain all of this to you now, but then we would have to simultaneously be teaching you general programming concepts AND the sometimes unintuitive/confusing syntax unique to C
- Similarly, no matter what language we might choose to begin with, the programming concepts themselves might be lost in confusion about syntax of the language
- Therefore, we're going to start with Scratch, which presents instructions in English.
- This will allow us to focus on how to write a program and how to think algorithmically.
- Later we'll show you how to translate those English instructions into a language like C.

Announcements (15:30-17:00)

- Harvard Computer Society is a student organization that promotes computing, computer science, technology, etc. You can visit their website at http:///hcs.harvard.edu.
- Lunch with David on Friday. If you'd like to attend send an email to rsvp@cs50.net. We hope to see you there!
- In the interest of being green, handouts are on the web.
- Office hours posted on the website.

Algorithms (17:00-23:00)

- In order to communicate with computers, we have to think like them. In algorithms.
- Here's a simple algorithm to look for your socks in the morning (Slide 49)
- Pseudocode is in language that humans can understand, but written methodically so that it can easily be translated into real code.
- In step 1, we set a variable socks_on_feet to the number 0.
- In step 2, we enter a loop that will be repeated as long as the condition (socks_on_feet != 2) is true.
- Inside the loop, we open the sock drawer, look for a sock, and then enter a condition:
    - If a sock is found, put it on.
    - Otherwise, do laundry and replenish sock drawer.
- This code has a bug. If only one sock is in the sock drawer, it will remain in the while loop forever.
- This is called an infinite loop. As you will see later, sometimes programs we write will contain infinite loops. We will recognize these cases because when we run the code, the computer will "hang"—it will keep running and produce no output because it is stuck in a loop.
- This happens in real programs, too. Programs that ordinarily function properly can, in weird corner cases, descend into infinite loops. This happens even in applications you use like Mozilla Firefox or Microsoft Word. You usually deal with it by pressing Ctrl+Alt+Del or shutting down the computer.
- It might seem obvious to us what to do in the case where there is only one sock in the drawer, but a computer cannot make assumptions the way we can.
- Therefore, our algorithms must be extremely *specific* and *precise* and account for all possible cases. Nothing can be assumed.

O Hai, Scratch! The Basics (23:00-39:30)

- When you open up a new project in scratch, you get an empty palette and a single sprite, a Cat
- A sprite is what will execute the instructions in your palette
- So, if we want the cat to say something, we drag the following pieces into the palette and put them together (Slide 52)
- Then when we press the green flag ("go!") the cat says "O hai, world"
- Similarly, we can give other instructions such as "Wait 1 second"
- Stringing statements together, we write the following program, which says "O hai, world" 3 times waiting 1 second in between each (Slide 54)
- But it seems inefficient to simply repeat the same statements over and over again. This is what we call bad style. We will see how to write this more nicely in a moment.

- Boolean Expressions
  - A Boolean expression is something that evaluates to either true or false. For instance, the following are all Boolean expressions (Slide 55)
- Conditions
  - Conditions are branches in the program that do one thing if a particular Boolean expression is true, and otherwise evaluate to false
  - Example (Slide 57). This meows as long as 1<2. So it will always meow.
  - This is a little boring. We can spice this up by picking a random number between 1 and 10 and seeing if its less than 6. This is like asking the computer to pick a random number off the top of its head. (Slide 57)
- Loops
  - Loops allow you to tell the computer to do something a set number of times, or just repeatedly forever
  - This program will meow every 2 seconds forever (Slide 59)
  - This, too, is boring. We can make it continue to meow repeatedly, but only if a certain condition is true, say, if we're touching it (Slide 59)
- Variables
  - We can use variables as placeholders for information, like in algebra
  - The following program simply counts up from 0, indefinitely (Slide 60)
  - We do this by storing the current value of our counter in the variable "counter"
- Arrays
  - An array is a collection of things. You can think of it like a list or vector.
  - (Technically they are vectors because they grow to accommodate your desired size. When you add something, to the end, it grows. C doesn't do this.)
  - So if we were writing an RPG, we could use an array to maintain the collection of objects that the hero is carrying around. Observe the following example (Slide 62)
  - This is used in FruitcraftRPG.sb, which is linked from the Lectures page of the website

More Complex Scratch Concepts (39:30-56:00)

- Threads
  - We can have multiple sequences of instructions being executed at the same time, as in move2.sb. The cat and the bird are each executing a set of instructions.
  - (Actually, the computer is not doing multiple things at once. It is repeatedly doing a little bit of each thing really, really fast. But it can give us the illusion that it is doing two things at once.)
  - In a more interesting example, we have one spirte executing two threads. In Hai10.sb, one thread repeated makes the seal bark so long as the muted variable is set to false, while the other thread "listens" for a key press and changes the muted variable if the space bar is pressed.

    - Again, we get two sets of instructions, for all practical purposes, at the same time.
    - Here we see an example of scripts sharing memory—the variable muted. Both scripts can check and change its value.
- Events
    - Scripts can communicate with one another by broadcasting events.
    - One sprite will throw or trigger an event, and another sprite will listen or catch it.
    - For example, in Marco.sb, the girl's script executes only when it receives the event from the boy's script
    - We see this in programs we use every day. For instance, a browser might respond when the Esc key is pressed, because the key press triggers an event and another part of the code handles it.
- For examples of more complicated programs you can write, see David.sb, cherrera.sb, and others

Sensors (56:00-64:00)

- Scratch also allows user input by means other than keyboard or mouse
- The pico board has a variety of sensory inputs: a button, an audio sensor, a light sensor, a lever that goes back and forth, and alligator clips
- Once you hook the pico board up to your computer (by USB cable), the sensing section of Scratch allows you to make use of these sensory inputs
- As a simple example, we have a man that appears to talk when we make sound into the sensor
- Check out davidwe.sb for an example of a game we can make using the lever to control position
- David has also written a real-world Oscartime by hooking the alligator clips up to tongs used to put a foil ball into a tin can. When the tongs touch the ball, the circuit is completed, and the alligator clips return a "true" to Scratch
- You can see a simplified version of this in Electricity.sb

Breaking Down a Complex Program: Oscartime.sb (64:00-70:30)

- In this program, trash falls from the sky and the user must click and drag each piece of trash into Oscar's trash can. As he receives trash, Oscar counts the number of pieces that have been collected so far. The object is to collect as much trash as possible before the song finishes.
- At first, Oscartime might seem very complex and difficult to implement. Where would you even begin?
- If we were going to write such a complex program, we would first break it down into its functional and structural components to make it more manageable.
- What functional components must Oscartime contain?
    1. Display instructions.
        - Use a stage
    2. Drop trash from the sky.

    - Represent trash with a gif image.
    - Give trash a location at top of screen with a random x-coordinate.
    - Decrease y-coordinate in a loop.
3. Detect if the mouse is hovering over trash.
    - Use if statement.
4. Allow user to drag trash.
    - Use if statement with "mouse-down?" Boolean
5. Impose time limit
    - Control overall structure of program with Oscar sprite
    - Have this sprite perform actions at specified time intervals
    - Meanwhile other sprites do interesting things
    - When Oscar sprite reaches end of script, game is over
6. Keep score
    - To change appearance of Oscar sprite as he pops out of the can, use costume changes
7. Lift Oscar's lid to put trash in
    - Use if statement to determine when mouse is within 40 pixels of can

- Now that we have identified all of the functions it must perform, we can tackle the problem one step at a time.
- We would start by implementing just one of the steps above, and then, as we get small pieces to work properly, put them together to grow a larger and more complex program. In this manner we could ultimately produce the finished product you see here.
- So we start with displaying the instructions. This is a matter of writing a short script for a sprite consisting of the instructions (say, a jpeg) that goes to a particular location, shows itself, waits, and hides itself.
- Next we make trash fall.
- How do we do this?
- First piece will be *when green flag clicked* to initiate the script.
- Next, position trash at top of screen: *go to (0,180)*
- To make trash move, put *move 2 steps* in a *forever* loop.
- Now, we test the program. It is very important to test small pieces as you build up a large program, rather than testing only at the end. This makes it easy to locate and fix bugs.
- An initial bug David encountered was that when we dragged a piece of trash, other pieces of trash would get picked up. Use of the variable my_click fixes this problem.

From Scratch to C (70:30-72:45)

- We can translate all of the familiar structures from Scratch into C
- Hello world (Slide 77)
- Booleans (Slide 78)
- Conditions (Slide 79)
- Loops (Slide 80)

- Variables (Slide 81)
- Arrays (Slide 82)

This week's problem set (72:45-75:30)

- Implement something in Scratch. See spec for details.
- You can optionally use a pico board. Come to office hours to sign one out.