Announcements

- All your clickers should work now. Let’s sync.
- Midterm moved to Tues Feb 12th to allow homework on truth tables, Huffman coding etc (due Feb 5th) to be graded and returned before the midterm
- Sections: you may need to go twice some weeks!
More announcements: Syllabus

More than one homework a week. Broken into smaller pieces.

Week 2: Jan 15th, 17th

- **TUESDAY.** Jan 15th.
  - Part 2: How do Computers represent data? How do they compute?
  - Intro to Processing. Demo the snow angel and the robot. Basic Processing.
  - HW 3, Processing program. Color the Robot introduced in class. Due Tues Jan 22nd.
  - Read [Huffman Coding](#) from Wikipedia for next Tuesday

- **THURSDAY.** Jan 17th. Digital Life, Privacy & the web (You will have already read [Blown to Bits](#) chapter 2)
  - Start thinking about what you think is interesting about this stuff in preparation for your ESSAY, HW4.
  - WATCH [http://www.youtube.com/watch?v=IPg1X8B1Dc](http://www.youtube.com/watch?v=IPg1X8B1Dc). [Wikirebels](#)
  - HW 4, Privacy and Social of the Web, 1500 WORD ESSAY. Due Sat 5:00 PM Jan 26th
  - HW 5, Blinky Stars, Due Thursday Jan 24th; This programming assignment is the first part of two parts. The other part of it is HW6 which is due a week from next Tuesday. We have split it into two parts SO you don't wait until the last minute to get started on it. It is more challenging than what you have done up until now.

Week 3: Jan 22nd, 24th

- **TUESDAY.** Jan 22nd.
  - Part 3: How do Computers represent data? How do they compute?
  - You will have ALREADY read [Huffman Coding](#) from Wikipedia
  - Bits & bytes. Digital Representations. Binary Numbers. Huffman Coding. THIS STUFF WILL BE ON THE MIDTERM.
  - Read [Blown to Bits](#) chapter 3 for Tuesday Jan 29th.

- **THURSDAY.** Jan 24th. Social Media, Lifelogging and Digital Memory. Guest Professor [Steve](#)
Part 2: How do computers compute?
Key ideas from last time

- By 1943, it had been shown that computers could be used for which of the following tasks …..???
  - A. Counting, Arithmetic
  - B. Translation of language
  - C. Logic.
  - D. All of the above.
  - E. None of the above
More Key Ideas

- Representation of data on computers;
  - Presence or Absence of signal: P and A
  - Signal can be mechanical, electrical, color, light, blinking an eye, moving a finger, yes, no, true, false, 0 1
  - Hollerith tabulating machine: mechanical => electrical
Abstraction …

- Formulating blocks of computation as a “concept” is **functional abstraction**
- What we did is important and we do it all the time in everyday life (otherwise we couldn’t deal with life’s complexity)...
  - We break a task down into (one or more) subtasks
  - We solve a subtask using a sequence of instructions
  - We put the solution into a function “package”, gave it a name, “process a riser,” and thus created a new thing, a concept, something we can talk about & use
  - Then we used it to solve something more complicated ... and probably repeat this approach at the next higher level
- This lets us do more complicated things
Recursion is when

- A. You call the same function as many times as you want.
- B. A function calls itself
- C. You use functional abstraction.
- D. None of the above.
Sierpinski: function call when mouse pressed

```java
int size = 600;
int border = 30;
float b = sqrt(size*size - (size/2)*(size/2));
int depth = 3;

void setup()
{
  size(size, (int)b);
  smooth();
  noLoop();
  fill(0);
  noStroke();
  background(255);
}

void draw()
{
  background(255);
  drawTris(0, depth, new PVector(border, h-border), new PVector(width/2, border), new PVector(width-border, h-border));
}

void drawTris(int level, int maxlevels, PVector left, PVector top, PVector right)
{
  level++;
  if(level > maxlevels) {
    triangle(left.x, left.y, top.x, top.y, right.x, right.y);
  } else {
    PVector a = PVector.add(left, PVector.div(PVector.sub(top, left), 2));
    PVector b = PVector.add(right, PVector.div(PVector.sub(top, right), 2));
    PVector c = PVector.add(left, PVector.div(PVector.sub(right, left), 2));
    drawTris(level, maxlevels, a, top, b);
    drawTris(level, maxlevels, left, a, c);
    drawTris(level, maxlevels, c, b, right);
  }
}

void mousePressed()
{
  switch(mouseButton) {
    case LEFT: depth++; break;
    case RIGHT: depth = max(depth-1, 0); break;
  }
  redraw();
}
```
Processing Basics: Recursive Circles

A demonstration of recursion, which means functions call themselves. Notice how the `drawCircle()` function calls itself at the end of its block. It continues to do this until the variable "level" is equal to 1.

```java
void setup() {
  size(640, 360);
  noStroke();
  noLoop();
}

void draw() {
  drawCircle(width/2, 280, 6);
}

void drawCircle(int x, int radius, int level) {
  float tt = 126 * level/4.0;
  fill(tt);
  ellipse(x, height/2, radius*2, radius*2);
  if(level > 1) {
    level = level - 1;
    drawCircle(x - radius/2, radius/2, level);
    drawCircle(x + radius/2, radius/2, level);
  }
}
```
Why does recursion work?
Size of ‘problem’ gets smaller on each call.
Stopping condition.
Will go back to recursion for creativity assignment.
Hollerith’s Tabulating Machine
To Process Data

- A mechanical machine can “read” a card with ... a “metal brush”
Sensing Punch Allows Some Action

- When the brush touches the source of current, the circuit closes,
- The electrical impulse can cause a mechanical action to happen that gives an instruction or records data.
A General Idea

- Digital Information: Detecting the presence or absence of a phenomenon at a specific place and time: PandA
- Phenomena: light, magnetism, charge, mass, color, current, ...
- Detecting depends on phenomenon – but the result must be discrete: it was detected or not; there is no option for “sorta there”
Making Data Digital

- “Digitizing” makes information discrete, it’s either there (1) or not (0), (not sorta there)
- A machine can determine that fact using mechanical or electronic means.
- Once data is digital, it can be processed in many different ways
- Processing power can grow and grow
McCulloch & Pitts Neuron

- Input = (W1 * X1) + (W2 * X2) + ... + (Wn * Xn)
- Output = If Input > Threshold then 1, otherwise 0
<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P and Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
# Truth Table for And (using 0 and 1)

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P and Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
These are the fundamental units of computers
Just like the neurons, but built out of transistors
So computers can do logic, using simple gates (like neurons) for each logic function....

But there are more key ideas.
Bits & Bytes

- P and A is a *binary representation* because it uses 2 patterns
  - Bit -- it’s a contraction for “binary digit”
  - a position in space/time capable of being set and detected in 2 patterns
Bytes: Standard encodings of meaning

- A byte is eight bits treated as a unit
  - Adopted by IBM in 1960s
  - A standard measure ever since
- Bytes encode the Latin alphabet using ASCII -- the American Standard Code for Information Interchange
With 8 places how many different letters?

A. 32
B. 64
C. 128
D. 256
E. 512
With 8 places how many different letters?

A. 32
B. 64
C. 128
D. 256
E. 512
UTF-8: All the alphabets in the world

- Uniform Transformation Format: a variable-width encoding that can represent every character in the Unicode Character set
- 1,112,064 of them!!!
- UTF-8 is the dominant character encoding for the World-Wide Web, accounting for more than half of all Web pages.
- The Internet Engineering Task Force (IETF) requires all Internet protocols to identify the encoding used for character data
- The supported character encodings must include UTF-8.
Encoding Information: There’s more!

- Bits and bytes encode the information, but that’s not all
- Tags encode format and some structure in word processors
- Tags encode format and some structure in HTML
- Tags are one form of meta-data
- Meta-data is information about information
DATA and SOFTWARE DOWNLOADS

PROFESSIONAL EXPERIENCE

Professor of Computer Science, Natural Language and Dialogue Systems Lab, University of California, Santa Cruz, 2009 to present

Professor of Computer Science, Head of Cognitive Systems Group, University of Sheffield, Sheffield, England, 2003 to 2009

Principal Research Staff Member, ATT Labs - Research, Florham Park, N.J., Speech Processing Software and Technology Research, 1996 to 2003

Research Scientist, Mitsubishi Electric Research Laboratories, Cambridge, Ma., Interactive Learning and Entertainment, 1993 to 1996

Consultant, Hewlett Packard Laboratories, Bristol, England, on dialogue systems, speech technology, and personal information systems: 1989-1993

Researcher, Dialogue Modeling Department, Electrotechnical Laboratory, Tsukuba City, Japan: Summer 1991


EDUCATION


B.A. Computer and Information Science, With Honors, University of California Santa Cruz, 1984.
Meta information

Can tell the web browser about the meaning or type of the information

```html
<font face="calibri">

<P>
<font size=+1> <B> <a href= "downloads.html"> DATA and SOFTWARE DOWNLOADS </a> 
</B>

href tells the web browser that downloads.html is a file (another web page)
```
More kinds of meta data all the time

- “The semantic web”
- An “ontology” i.e. “taxonomy” of the kinds of things there are in the world
  - People, place, thing, animal, organization, country
- New tags within web pages
- Could be put there by people
- Could be programmatically identified by text processing algorithms (like what Watson Jeopardy uses)
Representing Information: Summary

- Bits encode numbers using binary representations
  - 11 01 10 00
- Bits encode letters using ASCII for North American and Western European languages
  - 1110 0111
- Bits can be combined with other bits to do logic

- This suggests an principle we will soon argue:
  - All information can be represented with bits
Drawing pictures … It’s not art, it’s fun

Basic Processing …
First: How did Symbolic Lightbot go?

- HW2. Lightbot. Was due today

can be expressed symbolically,

Step  Right  Left  Jump  Power  F.name

- First Processing homework. Due next Tuesday
  - Download processing from www.processing.org. Do the tutorials
Recursion is when

- A. You call the same function as many times as you want.
- B. A function calls itself
- C. You use functional abstraction.
- D. None of the above.
Key idea: Functional Abstraction
The Function Becomes A Concept

- Because $F_1()$ “processes a riser,” I think of the programming task as

  - Process a riser $F_1$
  - Move to next riser $F_1$
  - Process a riser $F_1$
  - Move to next riser $F_1$
  - Process a riser $F_1$

- With $F_1()$ as a concept, I simplify the programming to just 9 steps rather than 21 OR

- It also suggests another concept:
  - Move_to_next_riser()
Now we will use these ideas in Processing …

- Processing is a language for programming graphical and image-based computations
  - Fun!
  - Easier to do because we “see” what’s happening
  - Immediate feedback => bottom up programming style
Get It: http://processing.org/download/

- If you have a personal computer, then grab a copy of the Processing system and put it on your machine
- You will want “Windows” or “Mac” versions
- Following installation instructions ... it takes less than 5 minutes and then you can work on your own computer!
Processing was initiated by Ben Fry and Casey Reas. It is developed by a small team of volunteers.

Processing is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production.

- Free to download and open source
- Interactive programs using 2D, 3D or PDF output
- OpenGL integration for accelerated 3D
- For GNU/Linux, Mac OS X, and Windows
- Projects run online or as double-clickable applications
What You See

- When you start up the Processing system...

programming window

file name

Run
Stop
New
Open
Save
Export
Add Some Code

- Type in instructions that you will learn shortly
- Then run your program

```java
void setup() {
    size(400, 400);
    stroke(192, 64, 0);
    background(192, 64, 0);
}

void draw() {
    if (mousePressed) {
        stroke(255);
    }
    line(150, 150, mouseX, mouseY);
}
```
What is a color in processing?

- A sequence of 0’s and 1’s....
- Red, Green, Blue
- Each color element is represented by one BYTE (8 bits).
- [http://processing.org/learning/color/](http://processing.org/learning/color/)
- So how many shades of Red (Green, Blue) are there?
  A. 8
  B. 16
  C. 256
  D. 512
The Color Purple

- Colors in most Web programming are given as three values: RGB, for red, green, blue
- The Color Purple, for example, is: 128,0,128
- These positions are the intensity of the little lights that make up a pixel on the screen
  - The least intensity is 0, that is, off
  - The greatest intensity is 255, maximum brightness
  - Amazingly, the three max RGB colors make white
  - So, purple is ½ intensity of Red, no Green, and ½ intensity of Blue ... makes sense

© 2010 Larry Snyder, CSE
RGB Color

Remember finger painting? By mixing three "primary" colors, any color could be generated. Swirling all colors together resulted in a muddy brown. The more paint you added, the darker it got. Digital colors are also constructed by mixing three primary colors, but it works differently from paint. First, the primaries are different: red, green, and blue (i.e., "RGB" color). And with color on the screen, you are mixing light, not paint, so the mixing rules are different as well.

- Red + Green = Yellow
- Red + Blue = Purple
- Green + Blue = Cyan (blue-green)
- Red + Green + Blue = White
- no colors = Black
Processing also has a color selector to aid in choosing colors. Access this via TOOLS (from the menu bar) → COLOR SELECTOR.
Questions about “Iron Rich Snow”

- The angel is on a rust-colored background specified by:
  \texttt{background(192, 64, 0); ...}
  which means?

- Stroke sets line color: \texttt{stroke(255, 255, 255);}

- Suppose the angel is “iron rich” and the snow white

- Fill sets color of object:
  \texttt{fill(128, 0, 128);}
It’s All The Same

- When the values for RGB are all the same, it’s some color of gray, or white, or black
- Since writing `background(255, 255, 255)` is kind of a drag, Processing allows us to give just one argument; so `background(255)` is equivalent to giving all three 255s
- What colors are these backgrounds?
  - `background(255, 0, 0);`
  - `background(64);`
  - `background(0, 0, 64);`
Simple Shapes Make Robots

- Reas and Fry, in their book, show us a cute robot they programmed using simple shapes.
- They give their code and we can see how they built it.
- To make the point that all code must “make sense” – it’s not gibberish – let’s look at it even though we don’t know Processing yet.
Robot Code, 1

```javascript
size(720, 480);
smooth();
strokeWeight(2);
ellipseMode(RADIUS);

// Neck
stroke(182);  // Set stroke to gray
line(266, 257, 266, 162);  // Left
line(276, 257, 276, 162);  // Middle
line(286, 257, 286, 162);  // Right

// Antennae
line(276, 155, 246, 112);  // Small
line(276, 155, 306, 56);  // Tall
line(276, 155, 342, 170);  // Medium
```
Looking At Simpler Code

- Drawing a snow angel is straightforward ...

```
void setup() {
  size(400, 400);
  background(192, 64, 0);
  stroke(255, 255, 255);
}

void draw() {
  line(150, 150, mouseX, mouseY);
}
```
Looking At Simpler Code

- Drawing a snow angel is straightforward ...

```java
void setup() {
    size(400, 400);
    background(192, 64, 0);
    stroke(255, 255, 255);
}

void draw() {
    line(150, 150, mouseX, mouseY);
}
```
Coding Is ALL Detail

- Notice everything!

- Two Functions, One Common Form:

  ```
  void <name> () {
  }
  all symbols + placement, matter
  ```

- Every statement ends with a semicolon (;)

- The software colors text it understands – helpful

- Some functions include stuff inside parentheses; these are called *arguments*

- If a function has arguments, each position has a specific meaning: `size(<width>, <height>);` `stroke(<red value>, <green value>, <blue value>);`

- If your cursor is by a closing parenthesis or brace, the matching parenthesis or brace is highlighted

- Keywords are highlighted in blue

- Processing is case sensitive; notice!
Red or White?

This program draws a
A. red rectangle
B. white rectangle

rect(10,10,30,40);
fill(255,0,0);
The Color Purple

- Colors in most Web programming are given as three values: RGB, for red, green, blue.
- The Color Purple, for example, is: \(128, 0, 128\)
- These positions are the intensity of the little lights that make up a pixel on the screen:
  - The least intensity is 0, that is, off.
  - The greatest intensity is 255, maximum brightness.
  - Amazingly, the three max RGB colors make white.
  - So, purple is \( \frac{1}{2} \) intensity of Red, no Green, and \( \frac{1}{2} \) intensity of Blue ... makes sense.
Questions about “Iron Rich Snow”

- The angel is on a rust-colored background specified by:
  `background(192, 64, 0);` ... which means?

- Stroke sets line color:
  `stroke(255, 255, 255);`

- Suppose the angel is “iron rich” and the snow white

- Fill sets color of object:
  `fill(128, 0, 128);`
It’s All The Same

- When the values for RGB are all the same, it’s some color of gray, or white, or black
- Since writing `background(255, 255, 255)` is kind of a drag, Processing allows us to give just one argument; so `background(255)` is equivalent to giving all three 255s
- What colors are these backgrounds?
  - `background(255, 0, 0);`
  - `background(64);`
  - `background(0, 0, 64);`
Simple Shapes Make Robots

- Reas and Fry, in their book, show us a cute robot they programmed using simple shapes.
- They give their code and we can see how they built it.
- To make the point that all code must “make sense” – it's not gibberish – let's look at it even though we don’t know Processing yet.
Robot Code, 1

```java
size(720, 480);
smooth();
strokeWeight(2);
ellipseMode(RADIUS);

// Neck
stroke(102); // Set stroke to gray
line(266, 257, 266, 162); // Left
line(276, 257, 276, 162); // Middle
line(286, 257, 286, 162); // Right

// Antennae
line(276, 155, 246, 112); // Small
line(276, 155, 306, 56); // Tall
line(276, 155, 342, 170); // Medium
```
// Body
noStroke(); // Disable stroke
fill(102); // Set to gray
ellipse(264, 377, 33, 33); // Antigravity Orb
fill(0); // Set to black
rect(219, 257, 90, 120); // Main body
fill(102); // Set back to gray
rect(219, 274, 90, 6); // Gray stripe

// Head
fill(0); // Set to black
ellipse(276, 155, 45, 45); // Head
fill(255); // Set to white
ellipse(288, 150, 14, 14); // Large eye
fill(0); // Set to black
ellipse(288, 150, 3, 3); // Pupil
fill(153); // Set to gray
ellipse(263, 148, 5, 5); // Small eye 1
ellipse(296, 130, 4, 4); // Small eye 2
ellipse(305, 162, 3, 3); // Small eye 3
Knowing Only About Color …

- We “improve” the robot by adding some color
- Then we make it move!
- Detailed instructions in homework PDF

Just Do It!
A Quick Comment on Processing

- We have written two kinds of Processing programs –
  - static, which only draw a picture
  - dynamic, which keep drawing a picture

```java
size(100,100);
background(255);
fill(0, 0, 255);
ellipse(50, 50, 30, 30);
void setup( ) {
    size(100,100);
    background(255);
}
void draw( ) {
    ellipse(50, 50, 30, 30);
}
void mousePressed( ) {
    fill(0, 0, 255);
}
```

What’s The Difference?
If there is time, start on Privacy lecture.