As Experienced Lightbot Hackers

- What are you doing in Lightbot?

- Commanding a robot through a “blocks world”
- Programming is **commanding** an agent

Agent, Instructions, Intent

- Other aspects of “commanding”
  - The **agent** is usually a computer, but it could be a person, or other device (e.g. a robot)
  - The agent follows the commands, the **instructions**, without making a mistake, and without stopping, doing only what it is asked
  - The program implements your intentions – it does what **you tell it to do**
  - **You** try to get the robot to the Blue Tile goal – that’s the point of the instructions that you gave it

Sequencing

- Instructions are given in sequence
- They are **executed** in sequence – essential
  - Instructions ...
    - From a limited repertoire
    - All are within agent’s ability; no JUMP_3
    - Executed one-at-a-time
  - A “program counter” keeps track of agent’s progress

Instructions Formed of Simpler Instructions

- Check out this screen shot of the Lightbot
- It is partway through an instruction … its beacon is lit, but not the tile
- To a programmer the instruction is monolithic (one thing)
- To an agent each instruction is a series of steps

Abstraction

- The word “abstraction” is used a lot in computing
- Remember: it was one of the 7 big ideas
- **Abstraction is a way to understand and solve problems**
- As a general definition, abstraction eliminates details to focus on essential properties
- The instruction example just given illustrates **functional abstraction** meaning that we have given a **name to a series of operations** that perform a coherent (and to us meaningful) activity; the name is the instruction, the series of operations are the bot’s actions to implement it
Abstracting

- Collecting the operations together and giving them a name is functional abstraction
  - The group of operations perform some function but we ignore all of the details
  - Giving it a name is functional abstraction
  - This is AMAZINGLY powerful
  - What makes it powerful, is we can forget about the operations and think only about the function they do; more about this later

- Let’s do some functional abstraction

Example: Abstraction in Everyday life

- Get Dressed
  - Dress Bottom Half
  - Put on Sox
  - Put on Pants
  - Dress Top Half
  - Put on Shirt

Functions abstract by packaging Computation

- F1() packages actions: E.G. “process a riser”
  - Here the “main” method has just 9 commands.

The Function Becomes A Concept

- Because F1() “processes a riser,” I think of the programming task as
  - Process a riser
  - Move to next riser

- With F1() as a concept, I simplify the programming to just 9 steps rather than 21 OR
  - It also suggests another concept:
    - Move to next riser ()
A Five Instruction Program: Define F1, F2, interleave

Here Is What Is Beautiful …

- Did everyone see 1 idea, 2 applications?

• To a programmer the instruction is monolithic (one thing)
• To an agent each instruction is a series of steps

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

True or false: Functional abstraction allows an agent (the robot for example) to complete a task in fewer steps.

The next set of lightbot exercises

Recursion: The repeated application of a recursive procedure or definition.
Recursion in Everyday Life

- Mathematics: 0 ... 9 are digits. A string of digits is a digit followed by a string of digits.

Sierpinski Triangle. See openprocessing.org

What controls when it happens?

Why doesn't it go forever?

Why does recursion work?
Size of ‘problem’ gets smaller on each call.
Stopping condition.
Will go back to recursion on Tuesday.

First Try: Recursion 2
Second Try: Recursion 2. Just keeps going....

Third Try: Recursion 2: Finds stop condition!

Programming Rule 1: Try, try again!!

It only does what you tell it to do

Recursion 3!: First Try

Recursion 3!: First Try

Recursion 3: OOPS!! Second Try
Recursion 3: Third Try...... Wow!

Recursion 4: Whoa!!! What is F2 subroutine?

What do you think?

Recursion 4: Whoa!!! What is F2 subroutine?

Recursion 4: We keep trying
The Function Is Just The Packaging

- Another way to use a function for the risers

Summary: Abstraction

- An instruction (of the Lightbot or any other computer) is abstracted into the command name; functions abstract useful and meaningful operations, functions abstract functions built of functions, etc.

Layer upon layer, we build software solutions

Summary From Lightbot 2.0

- Programming is commanding an agent
  - Agent: usually a computer, person, or other device
  - Agent follows instructions, flawlessly & stolidly
- The program implements human intent
- Instructions are given in sequence
- ... and executed in sequence
  - Limited repertoire, within ability, one-at-a-time
- "Program counter" keeps track current instruction
- Formulating computation as a "concept" is functional abstraction

Homework 2: Lightbot in Symbolic Form

Goal: The purpose of this assignment is to look at a symbolic copy of programming for the bot.

We can give the symbolic solution to the bot as shown. When we program symbolically, we can be expressed symbolically, that is, as text. So, our instruction list can be written symbolically, that is, as text. So, an instruction list

can be expressed symbolically.

Deep Right Left Jump Power F power

as shown. We can give the symbolic solution as above as text. So, we can give the symbolic solution as above as text.

HW 2: more Lightbot. Due Tuesday

Also for Next week. READ Blown to Bit Chapter 2.
How do Computers compute?

The 1890 U.S. Census

- Late 1880s: Herman Hollerith, inspired by conductors using holes punched in different positions on a railway ticket to record traveler details such as gender and approximate age, invented the recording of data on a machine-readable medium.
- First serious advance in digitization: punch cards
- Mechanical method – sensing a hole in a card or not

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
Computing w/o Computers

- Suppose Hollerith coded men as 0, women as 1.

Machine Reads Cards,
- Puts women in this slot
- Puts men in this slot
- Producing 2 piles
- Run each pile through again
- Just to count them – done

Writing As Important As Reading

- After processing based on reading cards, a machine can “save its work” by punching cards.

Making Data Digital

- “Digitizing” makes information discrete, it’s either there (1) or not (0).
- 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.

The first electronic computers: codebreaking during WW II

- COLOSSUS in the U.K.
- ENIAC at Penn.
- Female mathematicians were the first programmers!

The Human Brain: How does it compute?

- Contains approximately ten thousand million basic units, called neurons.
- Each neuron is connected to many others.
- Neuron is basic unit of the brain.
- Neurons specialize in communication.
- Information passes among neurons that result in brain activity.
Important Features of the Brain to Model

- The human brain: capable of immensely impressive tasks.
- Fault tolerant
  - Distributed processing
  - Many simple processing elements sharing each job
  - Can tolerate some faults without producing errors
- Graceful degradation
  - With continual damage, performance gradually falls from high level to reduced level
  - Only catastrophic failure (e.g., death) results in zero performance

Activating a neuron

- Axon: electrically active. Serves as output channel of neuron.
- Axon is non-linear threshold device. Produces electrical pulse, when resting potential within the soma rises above some threshold level.
- Axon terminates in synapse which couples axon with dendrite of another cell.
- Activating gates, when open allow charged ions to flow. Passes pulse to next neuron.

McCulloch and Pitts 1943

- “A Logical Calculus of the Ideas Immanent in Nervous Activity”
- Simple model of human brain, neuron
- Influenced the design of the first computers
- Starts with Logic
  - Ideas from logicians: Carnap, Russell and Whitehead.

Important features to model

- Neuron accepts many inputs, which are all added up (in some fashion).
- If enough active inputs received at once, neuron will be activated, and fire. If not, remains in inactive quiet state. Either ON or OFF.
- The output depends only on the inputs. A certain number must be on at any one time in order to make the neuron fire.
- The efficiency of the synapses at coupling the incoming signal into the cell body can be modelled by having a multiplicative factor (i.e., weights) on each of the inputs to the neuron.
- More efficient synapse has correspondingly larger weight.
McCulloch & Pitts Neuron

- Input = (W1 * X1) + (W2 * X2) + ... + (Wn * Xn)
- Output = If Input > Threshold then 1, otherwise 0

Neuron can model most Logical Functions

- McCulloch and Pitts then proceeded to show (prove mathematically) that such a model of the neuron could calculate most logical functions

- Let's say P = “Today is a sunny day”
- Let's say Q = “I am happy”

- When is (P and Q) true?

Truth Table for And (using True and False)

<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>

McCulloch Pitts Neuron for AND

How the neuron computes logical AND

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X+Y-2</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>0</td>
</tr>
</tbody>
</table>
McCulloch Pitts Neuron for OR

How the Or Neuron computes logical OR

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X+Y-1</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>

Truth Table for And (using True and False)

<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>

These are the fundamental units of computers

Just like the neurons, but built out of transistors

Solid State Electronics

- A transistor is a switch: If the gate (black bar) is neutral, charge cannot pass; if gate is charged, the wires are connected
It All Works Because of Digital

- Key principle: information is the presence or absence of a phenomenon at a given place and time!
- Card example:
  - Phenomenon – hole in the card
  - Present – detected by brush making electrical contact
  - Absent – brush insulated from electrical source
  - Place – there are several on the card, devices can know the positions
- Neuron example:
  - Phenomenon – activation on the axon
  - Present – action potential greater than threshold, neuron fires
  - Absent – action potential not greater
- Transistor example:
  - Phenomenon – gate is charged or neutral
  - Present – gate is charged, switch is on, current passes thru

Transistors

A transistor is most like
A. A complete neuron calculating e.g. the AND function
B. B hole (or lack of) in a punched card
C. An entire punched card with multiple holes
D. A whole bank of light switches

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”
- Place and time apply, but usually default to “obvious” values

Bits

- PandA 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
- PRESENCE
- ABSENCE => PandA

So computers can do logic, using simple gates (like neurons) for each logic function…..

But there are more key ideas
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
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 the binary representation 11 110 0111
- Bits encode letters using ASCII for North American and Western European languages
- This suggests an principle we will soon argue:
  - All information can be represented with bits