We’re underway …

Following Lightbot

Lawrence Snyder
University of Washington, Seattle
What are you doing in Lightbot?

- Commanding a robot through a “blocks world”
- Programming is **commanding** an agent
A Lightbot 2.0 “Computation”

Just Do It!
Agent, Instructions, Intent

- When you are commanding (programming), you direct an agent (by instructions) to a goal
  - The **agent** is usually a computer, but it can be a person, or other device (animated robot?)
  - The agent follows the commands a/k/a **instructions**, flawlessly, and mindlessly, doing only what it is asked
  - The program implements **human intent** – you are trying to get the robot to the Blue Tile goal – it’s the point of your instructions
Sequencing

- Instructions are given in sequence, i.e. in order
- They are followed in sequence, i.e. in order
  - YOU give the instructions ... it’s called programming
  - The AGENT follows them ... it’s called executing or running the program
  - A program counter marks the agent’s place
Order of Events

- The instructions are programmed *ahead of time*
- They are executed *later*, w/o programmer’s intervention
  - Each instruction makes *progress* towards the goal
  - The instructions *must be right* and sufficient to achieve the goal
Programming REQUIRES you to take the **agent’s point of view** ... it’s a essential idea
Programming REQUIRES you to take the agent’s point of view ... it’s a essential idea.

From this cell, a turn is required ... R or L?
The number and type of instructions is always limited – you need a solution using only them

- Instructions ...
  - The agent can do only certain things ... nothing else
  - The Lightbot’s instructions
  - There is no JUMP_3

... Lightbot’s even tougher than normal programming b/c in some LB games, some instructions are unavailable ... but it’s a game!

- Execute the instructions one-at-a-time
The limited repertoire is a fact of all computing, but how limited? A computer’s circuitry (the hardware) has very few instructions ... usually about 100, and many are just different versions of the same idea: `add_2_bytes`, `add_2_ints`, `add_2_decimal_numbers`, etc.

In theory, a computer with just 6 instruction types could compute all known computations.
If that were the end of the story

- Programming would be amazingly tedious if all programming had to use only the basic instructions – I mean REALLY REALLY tediously tedious
  - No one would be a programmer no matter how much it paid
  - Apps as we know them would not exist
  - BTW programming was like this in the beginning
    - This is why they are called the “bad old days”
- Luckily, there are functions
We make new instructions using functions!

F1( ) packages actions: E.G. “process a riser”
Functions Package Computation

Just Do It!
F1(), A Process a Riser Instruction

- We have a new instruction: Process_A_Riser

- Call the function to use the new instruction
The “process a riser” function was an obvious sub-problem of the overall task – we just saw it was a useful operation to perform.

But spotting common patterns is also another place to find “work” that could be turned into functions ... look how the “process a riser” code is used.
“Move To Next Riser”
Move To Next Riser

Perhaps the clearest solution
Yet Another Solution

Just Do It!
Functions may seem “obvious” but they are a HUGE idea ...

They allow us to solve problems by solving parts, naming them (at least in our mind), and putting the part solutions together to solve the whole problem

Sweet!

Really sweet!