Written by Danny Yaroslavski
Created March 07/14

How does Lightbot teach programming?

Document Overview:
This PDF provides an overview for what Lightbot is and what it has to do with Learning Programming and Computer Science.

About the Author:
Danny Yaroslavski has been programming for over eight years, has worked at companies including Electronic Arts, Armor Games Inc, and Side FX Software. He is the primary developer of Lightbot and holds a Bachelors of Computer Science from the University of Waterloo (Canada).

What is Programming? (General Concept)

Programming is the way in which a person tells a computer what to do. A computer does not understand regular English. A computer cannot be told to “walk the dog” or “create a painting of the Mona Lisa”. Computers understand only a small set of instructions. Programming is the way in which a computer is told to execute a series of instructions, from a set of instructions it already knows, to, in general, solve a problem.

The view you may have of the programmers and the programming we see today often combines the visual of typed language written (ie C++, Java, Python) and the result of executing it. For example:

```
int a = 5;
int b = a + 5;
print ( b );
```

Output:
10

However, at the core, programming is not really about the language the computer understands, or about typing expressions like shown above. It is more about the process with which we come to a solution and think algorithmically about how to solve a problem.

What is Lightbot?

Lightbot is a programming puzzle game. This means that at its core, it is a puzzle game, but its game mechanics lend themselves to actually having a one-to-one relationship with programming concepts (more on this later).

The goal of Lightbot is to, in each level, guide a robot to light up all the blue tiles in each level. To do so, you must 'program' the robot using a set of instructions. For example:

An arrow icon tells Lightbot to move forward one space.

A lightbulb icon tells Lightbot to light up the tile he is standing on

Relating Lightbot to Programming

Once you understand what programming means on a general level, and that is about thinking algorithmically to get to a solution, you can begin to see how Lightbot relates to programming.

Here, we'll go over the different concepts that are taught in Lightbot. These concepts will be split into two groups, the second of which has a more 'direct' link to programming.

P**rogramming practices -** the order in which programmers solve problems
C**ontrol-flow -** concepts that deal with the step-by-step sequence of program execution

Programming Practices

Planning

At the start of each level, players must evaluate a level and the instructions they have available to them. Players must imagine some way to put themselves in the robot's position and figure out how to guide the robot to solve the level.

This is the same as how programmers must understand and visualize a problem they are tasked with, evaluate the instructions they have available to them in their language of choice, and create a plan of action.

Programming

Players must sequence instructions according to their plan (using instruction icons).

This is the same as how programmers must write out their program (using code).

Testing

Players run the program they've created and test to see if the solution holds.

This is the same as executing a piece of code and seeing if the result holds with what's expected.

Debugging

When a level is not solved correctly, players must look for what may have caused the problem. This may include re-running Lightbot through the commands, or shifting commands around, or removing commands, to understand where the problem came in.

This is the same as what programmers must do when there is a bug in their code. They must generally re-execute the program, paying special attention to what could have gone wrong, as well as trace through where the mistake can be found.

Control-Flow Concepts

Control flow deals with how program execution occurs, one step at a time. There are often similar structures used in all programming languages which are the building blocks for making programs organized, modular and extendable. This includes procedures and loops.

Sequencing Instructions

Players must place instructions in an order that gets read from first to last.

This is much like how with code, programs execute one line at a time.

Procedures

Players in Lightbot must use procedures when they don't have enough space in the MAIN block to solve a problem. Procedures are helpful for *extracting patterns* and *re-using* a set of commands multiple times. Procedures also execute in a way that, when a P1 icon is run, all the commands in PROC1 are executed, and at the end of PROC1, execution returns to the command following the P1 icon.

This is much like procedures or functions in a typed programming language. Procedures are useful for *re-using* code and *extracting out* code that would otherwise be duplicated in a program, or that deals with a specific set of actions.

Loops

Players in Lightbot can use loops to solve some levels. Loops are useful for repetitive tasks, and are an extension of *extracting a pattern* which occurs *over and over.* In Lightbot, recursion is the type of looping that is used.

This is much like a recursive loop in a typed programming language. Loops are, again, useful for executing code that happens *over and over.* In a typed language, recursion is also performed in the same way as in Lightbot, by calling a procedure from within the procedure itself (ie. putting P1 in PROC1). At the end of the loop, program execution jumps to the start of the loop, like it does in the game. In fact, complicated loop structures like Mutual Recursion can be shown in Lightbot.

If Lightbot had a Typed Language

At its core, Lightbot is about programming, just without a typed language.
Imagining it were a typed language, we could have the following translation:


forward()


turnLeft()


turnRight()

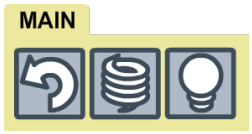

jump()


light()


proc1()


proc2()

Then a program like the following:



Would become

main:
  turnLeft()
  jump()
  light()

And a program like the following:

Would become

main:
  forward()
  proc1()
  proc1()

proc1:
  jump()
  jump()
  light()

Using the vocabulary at the top of the page, all of the levels in Lightbot Hour of Code™ can be re-written using the typed language given.

Glossary of Terms

A glossary of terms and definitions used, including technical terms.

Call	As in ‘call the forward instruction’ or ‘call procedure 1’; execute a specific instruction. eg. Call the forward instruction : The robot will execute moving forward eg. Call P1 : The robot will begin to read the instructions in the PROC1 block.
Control Flow	The order in which a program executes between instructions. In general, it flows from one command to the next. When a procedure is called, however, the program ‘transfers control’ to the instructions in the procedure. At the end of the procedure’s execution, the program transfers control back to the instructions following the call to the procedure.
Execute	Tell a computer to perform the instructions it is given (the program)
Execution	Refers to the state in which a computer is in when executing instructions
Loop	A control-flow concept in which a set of instructions is executed repetitively, over and over.
Mutual Recursion	A special type of recursion in which two or more procedures call each other to make a larger loop. eg. At the end of PROC1, P2 is called, and at the end of PROC2, P1 is called. This creates a large loop in which PROC1 calls PROC2 calls PROC1 calls PROC2 and so on.
Procedure	A smaller program that is used to extract a common set of instructions, or a pattern of instructions from the main program. It can then be called from within the larger program, one or more times. It may also be called from another procedure. eg. Call P1 from MAIN to execute the instructions in PROC1. eg. If P2 appears in PROC1, PROC2 is called from within PROC1.
Program n.	A set of instructions that is written by a programmer
Program v.	Write instructions to create a program.
Recursion	The type of looping used in Lightbot Loops levels. When a procedure is called from within itself. eg. Putting the P1 command inside of PROC1 to cause a loop.
Run	Same as execute: Tell a computer to perform the instructions it is given (the program)