Teaching Coding with Robots From Kindergarten to Career

Shaileen Pokress, Ed.M.
Scansorial, PBC

codewithroot.com

@codewithroot | @shaypokress

Meet Root. A robot that colors outside the lines of the educational robotics category.

But first…

- **What** is computer science?
- **Why** do K–12 students need to learn it?
- **How** will teachers learn to teach it?

What is it?

Experts have outlined *what students should know and be able to do* in computer science in grades K through 12.

We have standards!

- CSTA K–12 Computer Science Standards (csteachers.org)
- K-12 CS Framework (k12cs.org)
- ISTE Standards - Computational Thinking (iste.org/standards)
- Many states have adopted their own versions of standards

CS Scope & Sequence

Grade	K	1	2	3	4	5	6	7	8
Computational Thinking	Create algorithms, or series of ordered steps, to solve problems.								
		Decompose a problem, into smaller, more manageable parts.							
			Collect, analyze, and represent data effectively.						
					Demonstrate an understanding of how information is represented, stored, and processed by a computer.				
						Use abstraction to manage program complexity.			
							Optimize an algorithm for execution by a computer.		
							Create simulations / models to understand natural phenomena		
								Engineer software and/or hardware soluti	
									Evaluate algorithms
Computing Practice and Programmig	Demonstrate dispositions amenable to open-ended problem solving and programming								
	Use hands-on learning and the physical environment to explore computing concepts.								
	Write programs using visual (block-based) programming languages.								
							Write programs using text-based programming languages.		
	Locate and debug errors in a program.								
			Read a program and translate it into English. Explain how a particular program functions.						
					Design, code, test, and execute a program that corresponds to a set of specifications.				
			Modify and create programs, and present work to teammates.				Design, develop, publish, and present products (e.g., web page		
							Create computer programs with a practical, personal, and/or so		
Programming skills	**Implement problem solutions using a programming language, including:**								
	sequence								
		iteration: counted loops		iteration: while loops		iteration: nested loops			
	triggers		event handling & parallelism			event prioritization			
				conditional statements					
						randomization			
			code reuse (boomerang)		functions		functions with parameters		
				variables: integers, strings, booleans			variables: lists, arrays, data structures		
							operators & logic		

Why should K-12 students learn it?

Imagine that we stopped teaching life sciences in K-8 and then offered biology as an elective in high school.

How many kids would be interested in becoming doctors?

- Students develop career interests early.
- **Self-efficacy** directly correlates to interest.
- Broad exposure through schools will increase **diversity** in tech fields.

Computer science is a new literacy.
People use technology to solve problems in every field.

Sylvia Beyer, Kristina Rynes, Julie Perrault, Kelly Hay, and Susan Haller. 2003. Gender differences in computer science students. DOI=http://dx.doi.org/10.1145/792548.611930

How will teachers teach it?

Infusion into regular class time? Special subject?

Where & when?: School structure and “administrivia*”

- elementary lends itself to infusion

- high school leans toward stand-alone subject

- one certainty: must be part of regular school

Who? The teachers are already in place. Give them creative PD and the right tools.

Confidence more important than expertise!

**Hal Abelson, MIT Professor, coined this term, AFAIK*

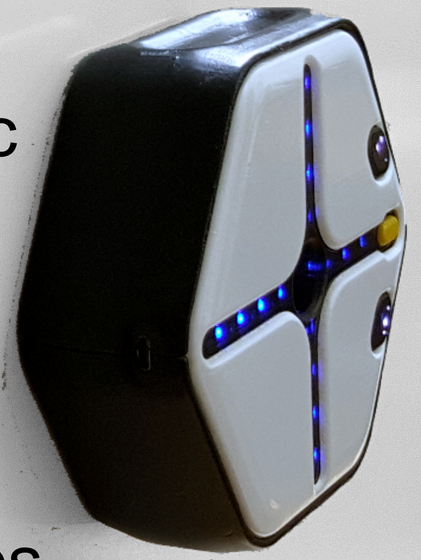
How will teachers learn to teach it?

This is why we created Root!

A robot that brings code to life

Simple setup: start on a **wall** (whiteboards) or use on a **table** (paper)

Many abilities: draws, erases, scans, plays music with over **50 sensors and actuators** and is expandable to add cameras, Arduinos, and customized hardware



Social: brings people together to create new games, activities, and coding challenges in an interactive way

A touch interface for coding

Multiple levels: switch between blocks-based graphical environments up to full text-based programming in the same app

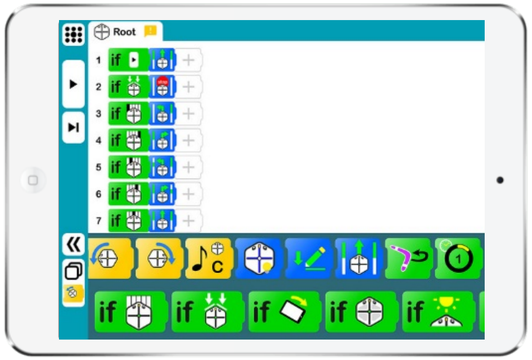
Code interactively: No compiling necessary; see what the robot senses and edit the code in real-time while it is running

Share programs: Users can save, edit, and remix their programs with each other from around the world


Root
1 if
2 if
3 if
4 if
if
if
if
if
if

Access to any age and skill level

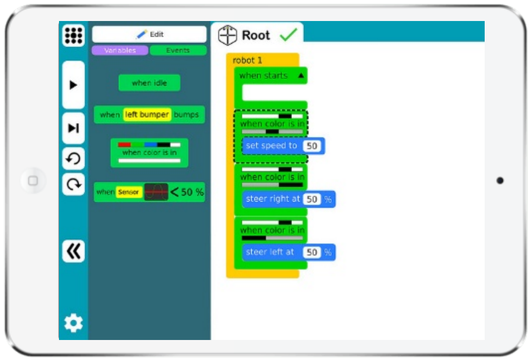
level 1



Graphical Programming

A graphical interface that beginners of any age (even non-readers) can understand. Colored blocks reinforce its basic "if this, then that" framework. It teaches events, sequences, loops, states, functions, priorities, timing, program stepping, and debugging.

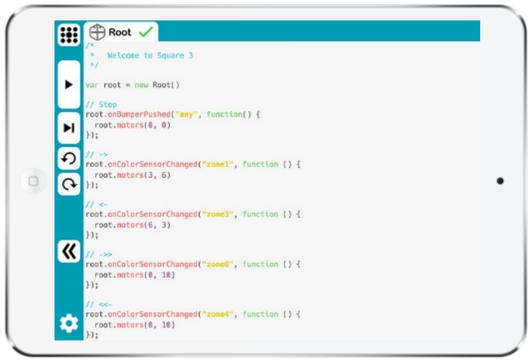
level 2



Computational Fluency

Introduces advanced flow control statements like when, repeat, and if-then-else, in order to foster computational fluency. It teaches variables, sensor values, units, arithmetic operations, recursion, and parallelism.

level 3

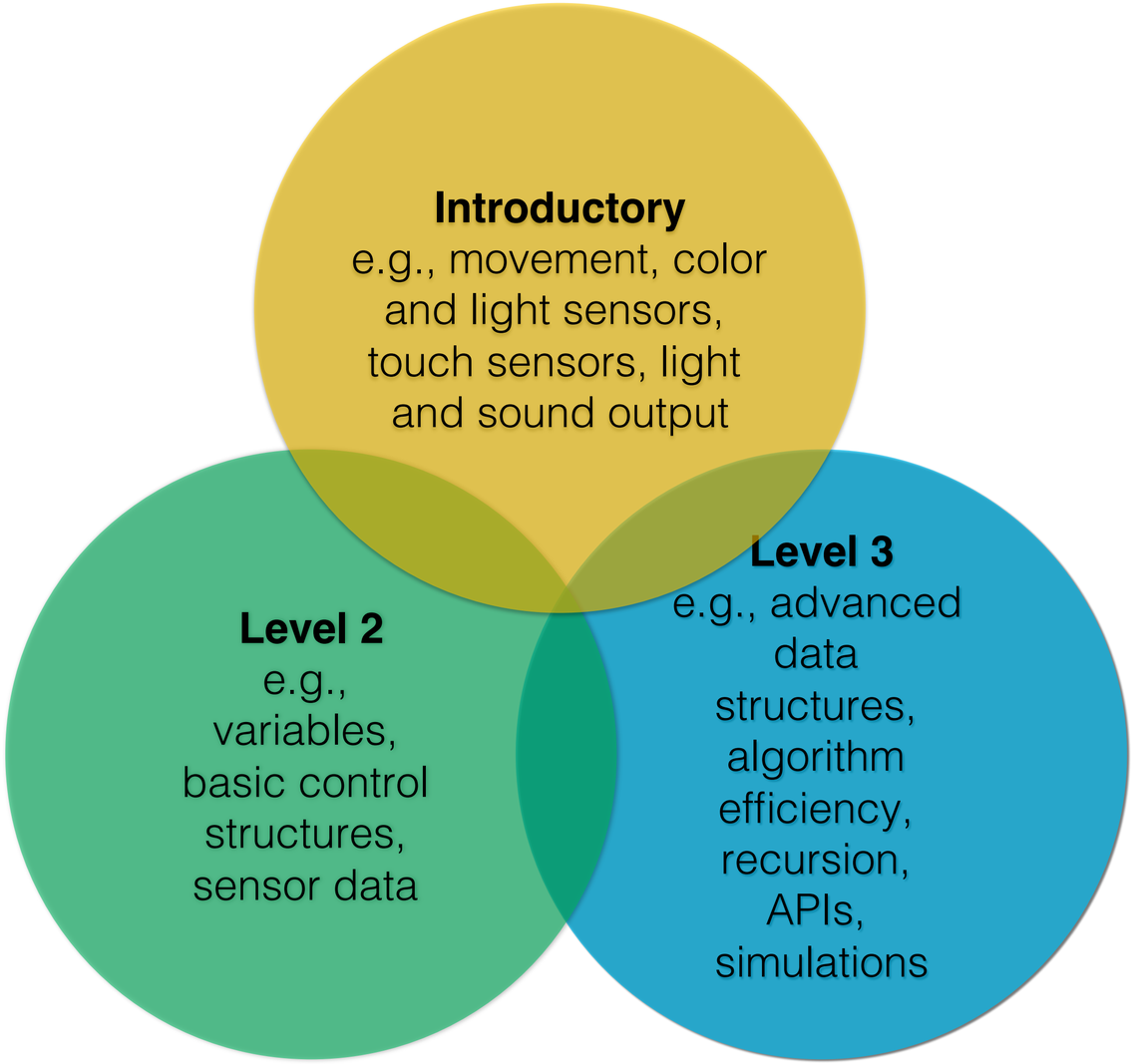


Full text programming

- Python
- JavaScript
- Swift

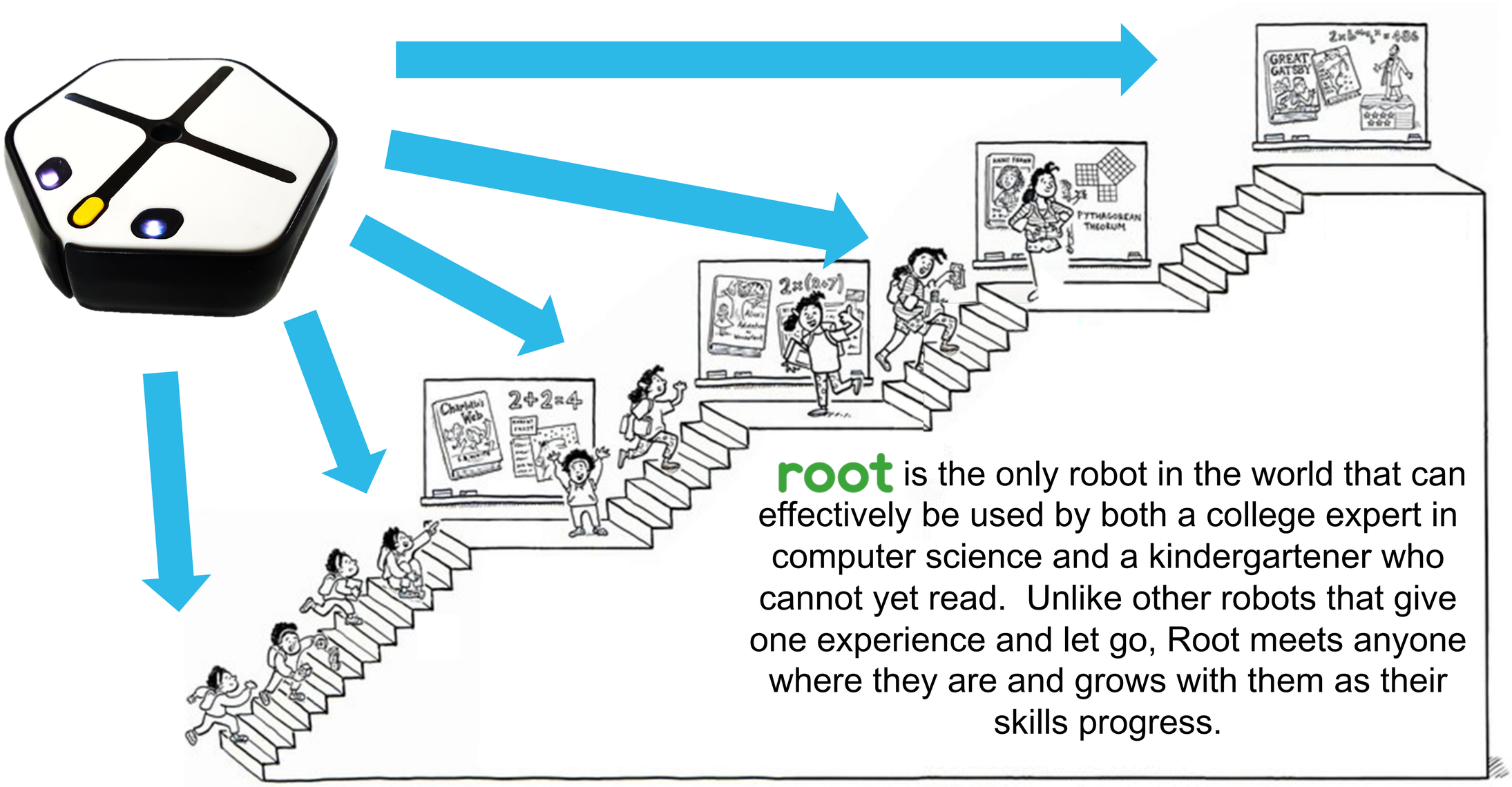
A fully text-based interface that takes programmers to professionally used languages including Python, JavaScript, and Swift. It teaches users the text equivalent of code from the previous levels, and lets them build powerful programs to accomplish new goals.

Pathways from introduction to mastery



Intentional overlap of activities between stages:
A 16-yr-old novice gets started the same way as a 9-yr-old novice, but progresses much more quickly

Start anywhere and grow

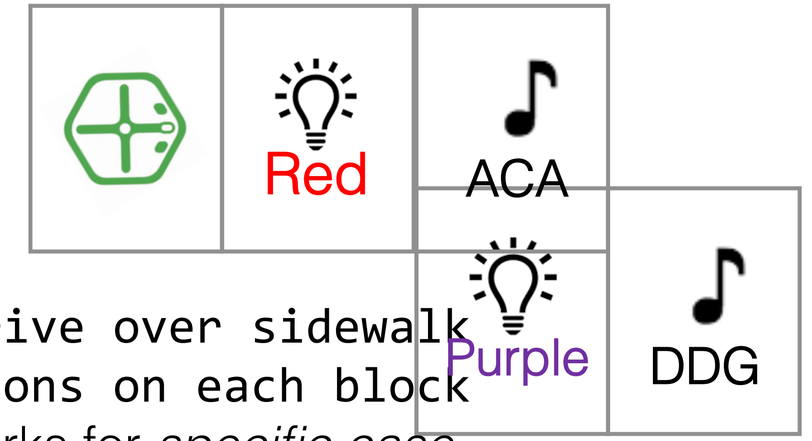


root is the only robot in the world that can effectively be used by both a college expert in computer science and a kindergartener who cannot yet read. Unlike other robots that give one experience and let go, Root meets anyone where they are and grows with them as their skills progress.

Appropriate complexity for different ages

Elementary school intro activity:

- Basic sequencing of move, lights, and sound commands
- Single trigger (when Play button clicked)
  - Debugging by trial and error

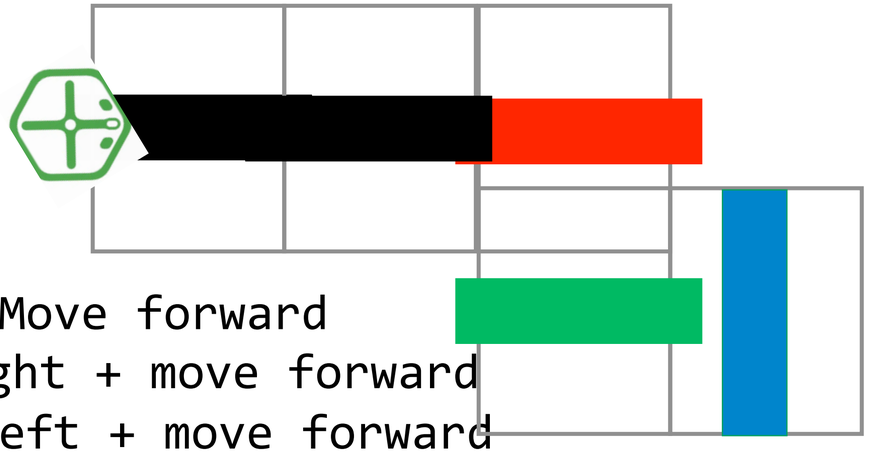



```
Make Root drive over sidewalk
Perform actions on each block
```

*Algorithm works for *specific case*

Middle school intro activity:

- Conditional sequencing
  - Multiple triggers (“when <color> detected”)
    - Debugging by iteration and evidence



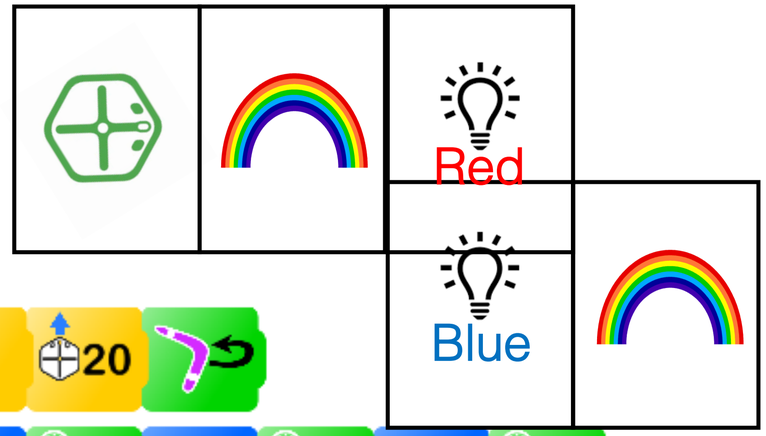
```
See black? Move forward
See red? Turn right + move forward
See green? Turn left + move forward
See blue? Stop
```

*Algorithm works for *any case*

Differentiated progression based on skill

Elementary school skill-building activity:

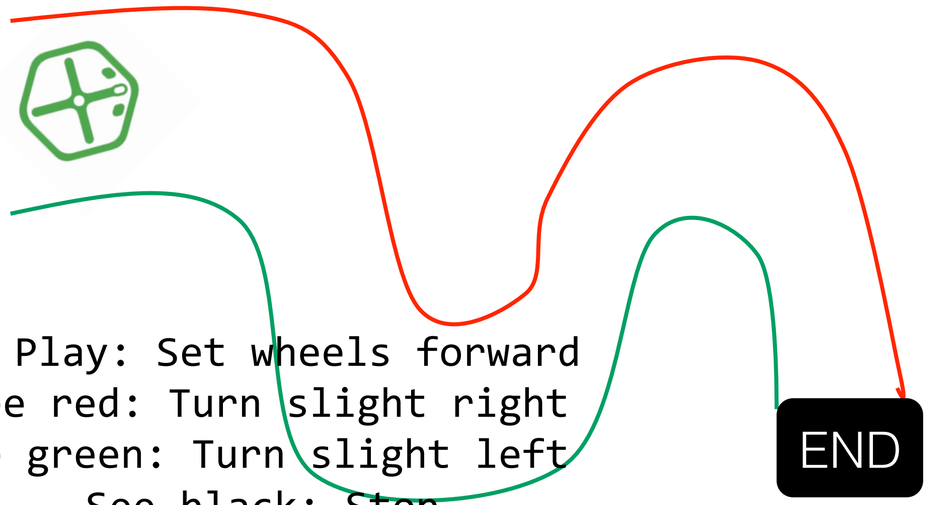
- Reinforce sequencing of moves + *reusing code* with boomerangs



Purple boomerang calls the code that makes the LEDs light up in rainbow order

Middle school skill-building activity:

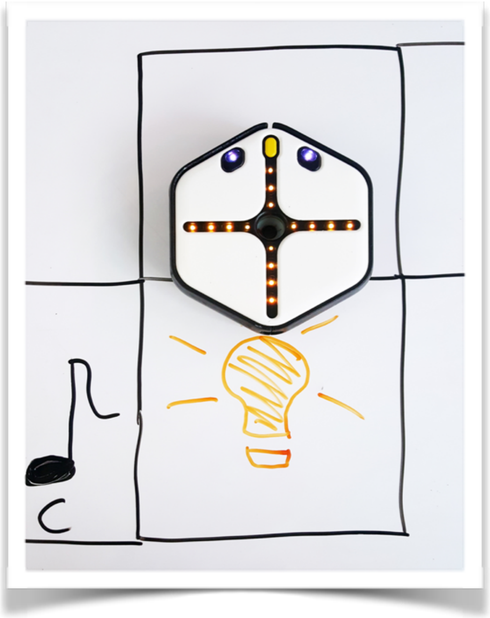
- Reinforce conditional sequencing + explore set state commands vs. action commands
  - *Priorities* of multiple triggers
- Debugging by iteration and evidence




```
On Play: Set wheels forward
See red: Turn slight right
See green: Turn slight left
See black: Stop
```

Must tweak magnitude of turns and priorities of event triggers

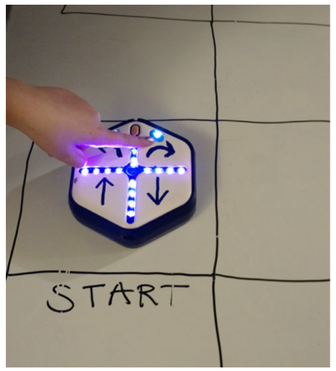
What this looks like in the classroom, early elementary example



Example Activity: **Sidewalk**

Grade-level: **K-1**

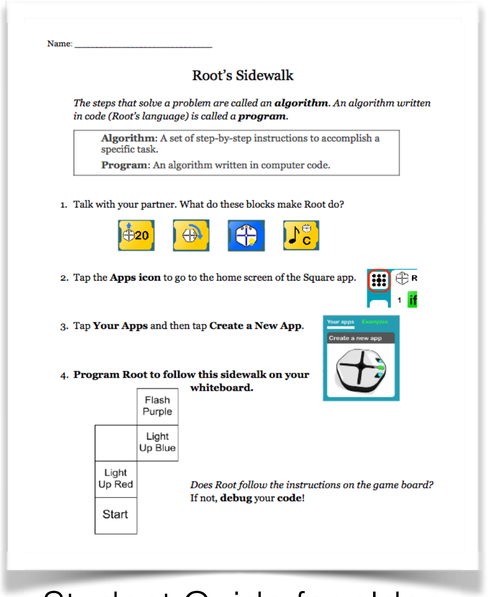
- **Teacher intro:** longer
- **Student guide:** step-by-step instructions and prompts
- **Students can:** Use touch interfaces on top of Root to give real-time instructions.
- **With time, students can:** Pre-plan code.



Touch control for youngest students

Grade-level: **2-3**

- **Teacher intro:** shorter
- **Student guide:** overview and exploration prompts
- **Students can:** Create their own sidewalks.
- **With time, students can:** incorporate looping and reusing code.



Name: _____

Root's Sidewalk

*The steps that solve a problem are called an **algorithm**. An algorithm written in code (Root's language) is called a **program**.*

Algorithm: A set of step-by-step instructions to accomplish a specific task.
Program: An algorithm written in computer code.

1. Talk with your partner. What do these blocks make Root do?

2. Tap the **Apps icon** to go to the home screen of the Square app.

3. Tap **Your Apps** and then tap **Create a New App**.

4. **Program Root to follow this sidewalk on your whiteboard.**

Flash Purple
Light Up Blue
Light Up Red
Start

Does Root follow the instructions on the game board?
If not, **debug** your **code**!

Student Guide for older elementary students

Root in the Classroom

Teachers find value in how Root *engages all students* and enhances student learning. A new differentiation!

Students learn how to communicate, work as a team, iteratively refine, think logically, and solve problems creatively.

Root Scope & Sequence

Grade	K	1	2	3	4	5	6	7	8
Computational Thinking	Create algorithms, or series of ordered steps, to solve problems.								
		Decompose a problem, into smaller, more manageable parts.							
			Collect, analyze, and represent data effectively.						
					Demonstrate an understanding of how information is represented, stored, and processed by a computer.				
						Use abstraction to manage program complexity.			
							Optimize an algorithm for execution by a computer.		
							Create simulations / models to understand natural phenomena an		
								Engineer software and/or hardware solution	
									Evaluate algorithms b
Computing Practice and Programmig	Demonstrate dispositions amenable to open-ended problem solving and programming								
	Use hands-on learning and the physical environment to explore computing concepts.								
	Write programs using visual (block-based) programming languages.								
							Write programs using text-based programming languages.		
	Locate and debug errors in a program.								
			Read a program and translate it into English. Explain how a particular program functions.						
					Design, code, test, and execute a program that corresponds to a set of specifications.				
			Modify and create programs, and present work to teammates.				Design, develop, publish, and present products (e.g., web pages,		
							Create computer programs with a practical, personal, and/or socie		
Programming skills	**Implement problem solutions using a programming language, including:**								
	sequence								
		iteration: counted loops		iteration: while loops		iteration: nested loops			
	triggers		event handling & parallelism			event prioritization			
				conditional statements					
						randomization			
			code reuse (boomerang)		functions		functions with parameters		
				variables: integers, strings, booleans			variables: lists, arrays, data structures		
							operators & logic		


root

Thank you

Web: codewithroot.com

Email: team@codewithroot.com

Twitter: @codewithroot