Gary S. Stager

professorgarystager.com/melbourne

Reasons to Learn to Program Computers

- **Make things**
- **Make things work**
- **Express yourself**
- **Develop habits of mind**
- **Matches young people's capacity for intensity**
- **Solve problems**
- **Concretize abstractions**
- **Contextualize mathematics**
- **Mirrors the writing process and various design cycles**
- **You can do it by yourself or with others**
- **"Hard fun"**
- Jobs / careers

The benefits of computation

- Describes phenomena through formal, numerical, representations
- May produce action
- Makes project-based learning possible in math
- Hard fun
- Models thinking
  - "You can't think about thinking without thinking about thinking about something." – Seymour Papert
- Opportunities for debugging
  - "The question to ask about the program is not whether it is right or wrong, but if it is fixable." – Seymour Papert
- Makes interactivity possible
- Bestows agency on the learner
- Amplifies our potential by the computer working for us instead of us working for the computer
- Sustains democracy

An Embarrassment of Riches
COMPUTATIONAL MENU

STARTERS

Turtle Art
Intuitive & creative web-based block programming environment for learning mathematical thinking while creating beautiful images

Browser-based & iPad app
constructingmodernknowledge.com/turtleart/

Scratch
The MIT Media Lab's block-based multimedia environment for creating animations, games, sharing, and remixing programs - may also interact with the micro:bit and LEGO in a limited fashion

Browser-based & iPad app
scratch.mit.edu

SNACKS

Octostudio
New software from the Scratch team lets you program games and multimedia fun on a cellphone—interacts with micro:bit too

iOS & Android
octostudio.org

microBlocks
Live program micro:bits & similar microcontrollers in real time without downloading code

Browser-based, requires firmware update
microblocks.fun

HARDWARE

BBC micro:bit web site
Learn all about the revolutionary $20 brain board in use by millions of learners

microbit.org

Hummingbird Robotics Kit
Our favorite robotics and physical computing kit for classroom use, powered by the micro:bit and containing everything kids need to create great projects. Their Finch (floor turtle) is cool too.

Birdbraintechnologies.com

MAINS

Microsoft MakeCode
Block-based web environment for programming the BBC micro:bit, arcade games, Minecraft, LEGO EV3 & more—convert blocks to Python & Javascript if you wish

Browser-based
makecode.com

Lynx
Text-based multimedia version of the Logo programming language with robust programming functionality for computationally-rich projects. MicroWorlds descendent.

Browser-based
lynxcoding.club

Snap!
Block-based superset of Scratch designed for teaching computer science—can interact with the micro:bit & share projects online—our preferred programming environment for the Hummingbird Bit. Snap! was developed at UC Berkeley and is used through university CS courses

Browser-based
snap.berkeley.edu

Beauty and Joy of Computing
(AP Computer Science Principles curriculum)

Wolfram Language
The language driving Mathematica and Wolfram Alpha is available for users to embark on their own insanely powerful computational adventures. Now with an LLM "chat" front-end.

Browser-based
wolframcloud.com

K-12 pricing & resources:
www.wolfram.com/computational-literacy-package/

TREATS

Turtle Stitch
A Snap! dialect that outputs to programmable embroidery machines. Functionality may soon be integrated into Snap!

Browser-based
turtlestitch.org

Creating a Microworld in Turtle Art

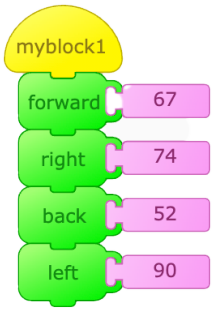
Sadly, Turtle Art does not allow users to hide blocks or customize the block palette. There are ways, however, to create your own blocks and to direct learners to use those specific blocks for the purpose of creating a microworld. **`playfulinvention.com/webturtleart`**

Create a new block

New blocks are created by putting a "hat" on top of a stack of blocks describing a procedure composed of a list of commands (other blocks).

Name a new block

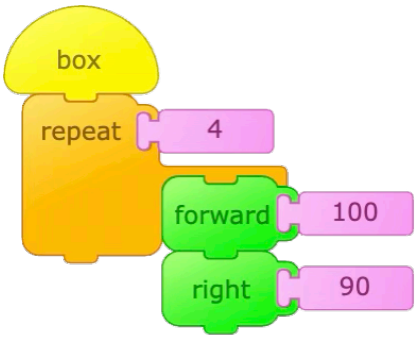
Click on the yellow "hat" and type a one-word name for the new block. Do not use a word already used by a Turtle Art block, or a number, or multiple words. If you wish to name a block *My Block 1,* name it *myblock1.* Most programming languages have limitations and syntactical conventions.

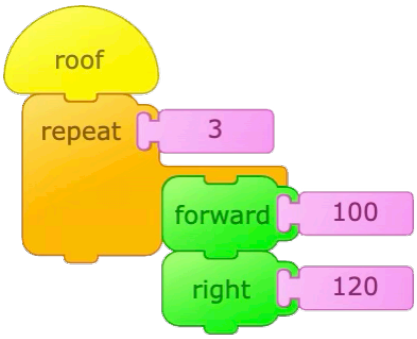


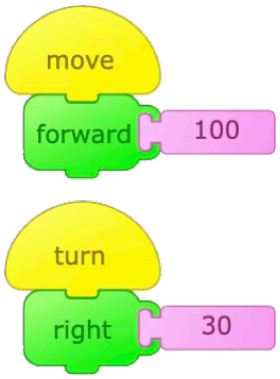
Make sure all the blocks are connected to the naming block (hat) by clicking and dragging the "hat." Every block below the "hat" should remain connected.

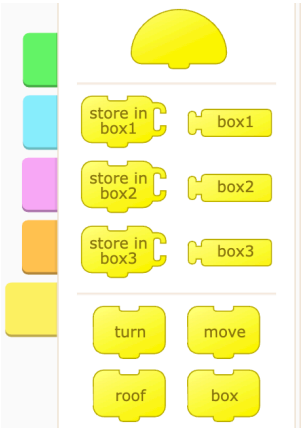
A simple microworld

Give learners these four blocks and ask what they can make with them. (there is no right answer). You can print the blocks on card stock or just ask that their creations are limited to the blocks below the line in the yellow palette.





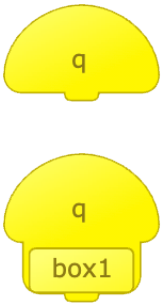




Adding inputs to a custom block

Turtle Art includes three blocks representing global variables, *box1*, *box2*, & *box3*. There's a trick for creating blocks with inputs. For example, the `forward` block has an input, so does `right`. To create a block with an input, do the following.

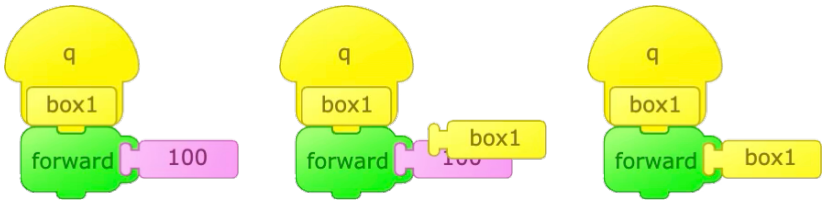
1. Drag a hat/naming block onto the screen or add a hat/naming block to a stack of blocks.
2. Give the stack a name by clicking on the hat block, typing a name, and clicking anywhere else on the screen. You should now see the new block in the (yellow) palette of blocks.
3. Next, drag a `box1`, `box2`, or `box3` block on top of the hat block. It should put a rectangular sign with the name of the input you just added inside the mushroom-shaped hat.
4. That's it. That block now has an input!



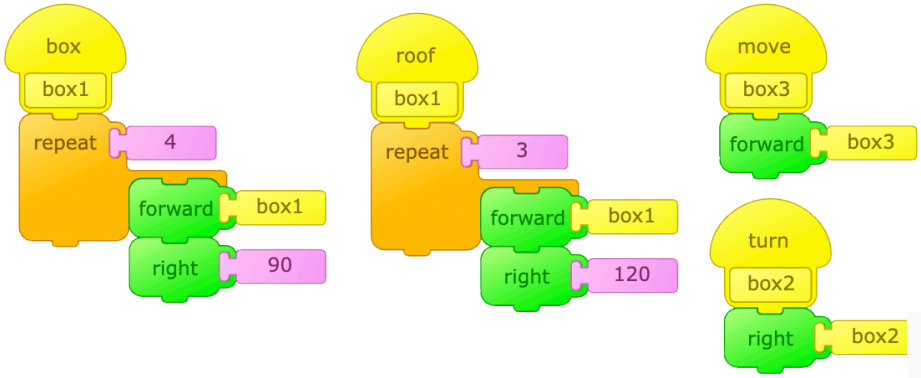
Using a new block containing an input

The `box1`, `box2`, or `box3` you used as an input may be used to represent a value substituting for a fixed or constant number. For example, instead of `forward 10`, you can say `forward box1`, and the thing inside `box1` will be the value handed to `forward`.

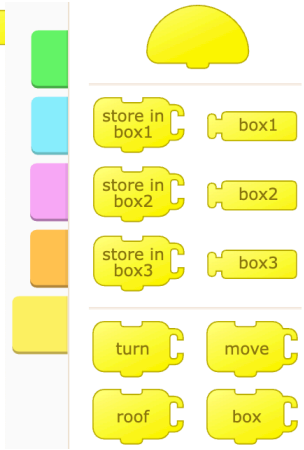
✦ Dragging an input block over a pink numerical block will replace the value with an input.



A new microworld



❖ The input you choose, `box1`, `box2`, or `box3` does not matter since its value is local to the particular block you are creating.



Deploying your microworld

1. Save the project
2. Open Turtle Art on each computer
3. Drag the project you saved into Turtle Art
4. Mess about with the microworld!

Creating a Microworld in Snap!

Snap! is a visual, block-based programming language designed for educational purposes, especially to introduce learners of all ages to computer science and computational thinking in a powerful and accessible way. It is an extension of the Logo family tree and may be thought of as Scratch for computer scientists.

You can design a microworld in Snap! due to its flexibility and large number of primitives. Snap! allows you to hide primitive blocks built into the system and add your own blocks to the programming environment. Some microwords may be intended to assist learners by eliminating confusion or distraction caused by too may blocks to choose from.

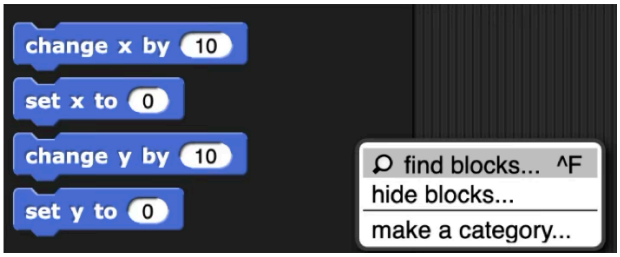
Creating a microworld with a limited number of blocks has obvious applications for working with young children or learners with special needs. In such cases, the Snap! environment is customized in a prosthetic fashion.

A microworld may also add new high-level blocks created to perform complex tasks or simplify functionality. One might imaging a trigonometry microworld or fractal microworld in which new blocks (super procedures) are added to Snap! to allow users to engage with powerful ideas without the need to build the blocks themselves or understand the underlying code.

Best of all, customizing the Snap! environment represents a form of toolmaking. Building one's own tools is a human superpower that allows us to be more efficient or solve more complex problems.

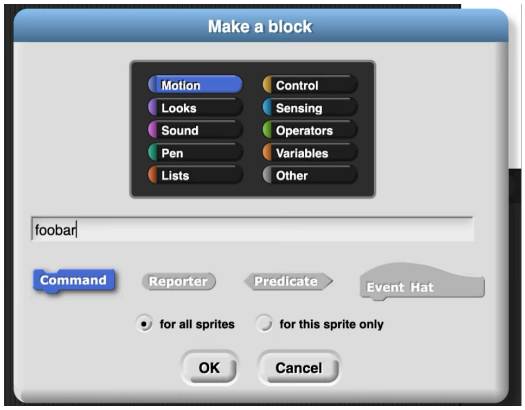
How to hide blocks

1. Open Snap! at snap.berkeley.edu
    1. login to your account if you have one
2. Click the File Icon and select `Hide blocks…`
3. Click on the blocks you wish to hide from the system. Those blocks will disappear from your blocks palette.



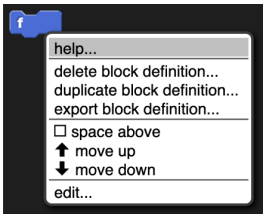
Alternatively, you can CTRL-Click (Mac) or Right-Click (PC) in the blocks palette and choose `hide blocks`. This allows you to hide blocks from that palette of blocks only, not all palettes.

Incidentally, if a block uses another block in its own code, it will still work despite being hidden from view.



How to add blocks

1. Click `Make a block` button in the palette
2. Click on the color/category of block you wish to create
3. Determine the function of the block – command, reporter, predicate, event hat
   Name the new block
4. Click the plus sign to add inputs if your block requires them
5. Drag blocks into the block editor to create the definition for that block
6. Click OK
7. Test your block
8. Debug, edit, or improve your block by CTRL-Click (Mac) or Right Click (PC) on the block and select `edit…`



Share your projects with students or colleagues as a link (cloud sharing)

1. Make sure you're logged into your Snap! account and your project is saved.
2. Go to File → Open…, select your project in the list.
3. Click the Share button and click "Yes" if prompted
4. The address bar will update to a URL containing your username and project name.
5. Copy that URL and send it to anyone — when they open it, they'll see the most recent version saved to the cloud

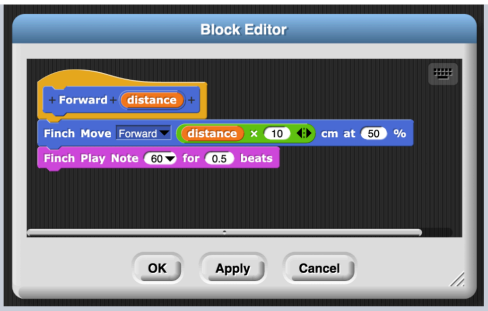
Note: Whenever you save a project, even after sharing, the link updates to reflect your new changes.

MakeCode

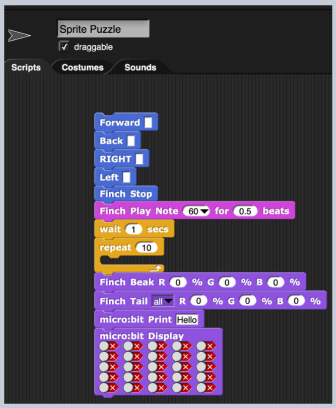
The Invent to Learn Guide to the micro:bit from Constructing Modern Knowledge Press, features a chapter describing how new blocks may be created in MakeCode for use with the BBC micro:bit, video game programming, and other microcontrollers.

Creating a Finch Microworld

1. Get a charged Finch robot
2. Start a new project at snap.birdbraintechnologies.com
3. Establish a connection to your Finch
4. Build your own blocks for making the Finch behave like a Logo turtle, blocks for `forward, back, right, left`...
5. Debug
6. Connect all of the blocks you programmed and any other essential blocks and choose Generate Puzzle from the Snap! file menu. This creates a "new" version of Snap! featuring just the blocks you chose. You may also hide irrelevant blocks. (see Snap! microworld handout)
7. Save the project for the use of students.
8. Observe students using your new software.
9. Make adjustments as necessary.



Build your own block with an input



Connect all the blocks you wish to include in your puzzle

Or hide all of the undesired blocks in each palette

Select Generate puzzle from the file menu to create a puzzle

Challenges

- Make blocks for moving the turtle and turning by reasonable physical distances
- Make blocks for "driving" the turtle with F, B, R, L and no input
- Create turtle movement blocks where 1 = 10 Finch/turtle steps
- Is it possible to use Finch sensors to make sure your turtle does not run off a table?
- Can you create a block for resetting/clearing a turtle (such as `clean, clear, cg`, or `cs`)?
- Change your blocks to
- Design a version of Turtle Art in Snap!

Resources

Check out the turtle graphics/turtle geometry project ideas found in eBooks available at dailypapert.com/logo, including *LogoWorks, Teaching with Logo, Learning with Logo, Turtle Confusion* and *Turtle Speaks Mathematics.* Can you use these ideas in your new Finch Turtle microworld?