Workshop Handout

Additional resources at

inventtolearn.com/questions

Reasons to Program Computers

- Agency over an increasingly complex & technologically sophisticated world
- Make things
- Make things work
- Express yourself
- Develop habits of mind
- 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

What is Logo?

Resources, links, books, & articles

dailypapert.com/logo

Languages for Learning

Language	URL
Turtle Art	turtleart.org
Scratch	scratch.mit.edu
Snap!	snap.berkeley.edu
Lynx	lynxcoding.club
microBlocks	microblocks.fun
Microsoft MakeCode	makecode.com
Turtlestitch	turtlestitch.org
Beetleblocks	beetleblocks.com
Wolfram Language	cmkfutures.com/wolfram

inventtolearn.com/stuff

BBC micro:bit Challenges

Micro:bit and MakeCode Getting Started Prompts

1) Program the micro:bit to display a new image or scrolling text

2) Program the micro:bit to display one image when the A button is pressed and a different image when the B button is pressed.

3) Program the micro:bit to display the temperature. Test your thermometer.

   Super duper gifted and talented extra credit: Display the temperature in Fahrenheit

4) Program the micro:bit to display a smiley face when the temperature is above a certain value and sad face when it gets cold.

5) Program one micro:bit to "pass a message" to another micro:bit, using the radio features, when a user presses a button.

   You can't handle this challenge! Program two more micro:bits to pass messages between each other.

6) Program one micro:bit to cause another micro:bit to produce some action, such as light an LED, drive a servo, or run a program.

7) Make a micro:bit stopwatch. (tricky)

1-hour micro:bit Workshop Challenge

1. Program the micro:bit to behave like a die rolled by a player.

2. Instead of displaying a number, display a die face.

3. Add some effects to make it look (or sound) like rolling a die before it settles on a "side"

4. Roll your die and have it send the value to appear on a friend's micro:bit.

5. Make the die rolled in your hand make something else happen on a friend's die, like flash an LED x times.

6. Use your micro:bit die with Scratch to control an animation or interact with a board game you program.

Read more about the thinking behind the design of this activity at https://inventtolearn.com/1-hour-microbit-workshop-challenges/

The Invent to Learn Guide to the micro:bit

Getting Started with micro:bit Radio Communication

By Gary S. Stager, Ph.D.

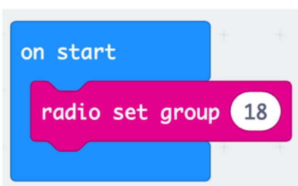
Although the block-based MakeCode environment is quite intuitive, the radio functionality of the micro:bit requires a tiny bit of instruction. You can use the micro:bit's radio functionality to pass messages, as a remote control, or even to model social behavior between robots. Here is a concise guide to get you started.

Imagine the micro:bit as a walkie-talkie

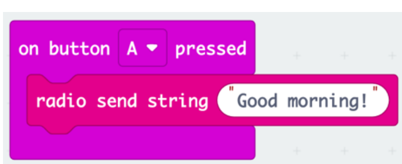
If you talk into a two-way radio, like a walkie-talkie, any other radio within range and using the same frequency or channel should receive your message. Friends should also be able to speak to you. It does not matter how many walkie-talkies are involved. Each radio on the same frequency can participate in the conversation. The micro:bit works in a similar fashion.

Voice is but one of the forms of data a walkie-talkie can share. Some allow users to exchange morse code. The micro:bit doesn't transmit voice, but it does have the ability to share text (strings) and numbers between a seemingly infinite number of micro:bits using the same channel within a limited physical range.

1. Each micro:bit in your "network" needs to be set to the same channel. If you are exchanging private messages, keep the channel extra secret quiet. To set the channel, each micro:bit being used needs to have an `on start` block that sets the *same* channel from 0-255.

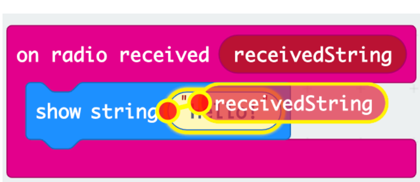


2. Each micro:bit needs code to broadcast messages/data to other listening micro:bits and code for reacting once it "hears" a new message. In this case, when a user presses the A button on the micro:bit, a message of "Good Morning!" is broadcast. If no other micro:bit is listening, that message falls on deaf ears.

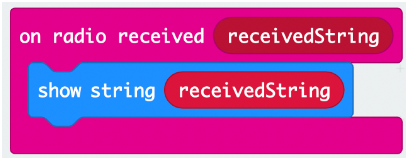


It is critical to decide if you are sending & receiving numbers or strings (text). You must the choose the correct *send* and *receive* blocks according to the data type.

3. If more than one of the micro:bits in your community contain blocks like the first two and the next one, you will have created a simple text-based walkie-talkie system.



Use an `on radio received` block and drag the `receivedString` variable into the input of the `show string` block. This will replace a literal message with the variable one being broadcast by another micro:bit. If dragging the variable into the container is too difficult, the variable may also be found in the blocks under the radio menu.



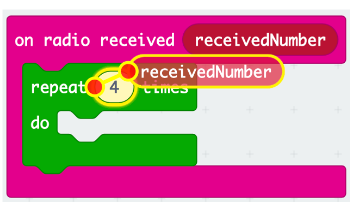
That block has the job of listening until a new message is broadcast (within) range and then doing something. In this case, it displays the message for the owner of the receiving micro:bit to read, just like passing a note or sending a text message.

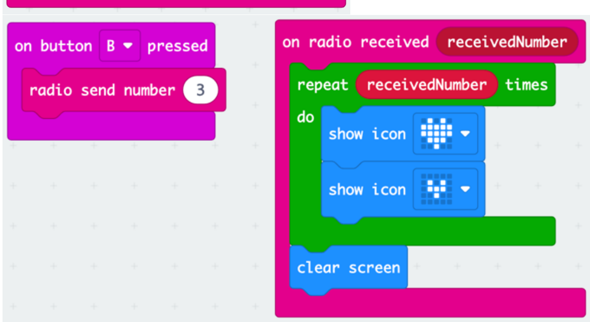
You can of course program other buttons or micro:bit movements to send other messages.

All of the micro:bits participating in this chat need to have the same three blocks, or reasonable facsimiles downloaded to them.

You can also send numbers!
You might send a numerical value between micro:bits when you wish to trigger an event without a textual (string) message being received and displayed. For example, what if you want to tell another micro:bit to start an animation, flash an LED *X* times, or tell your robot to turn left or right? Sending a number might just do the trick.

1. Set the channel as you did in the previous example.
2. Use an `radio received` *`receivednumber`* block to build a simple animation that will repeat the number of times one micro:bit broadcasts to others.





Strings and numbers may be shared via radio in the same program.

Remember that every micro:bit in your network needs to have similar *send* and *receive* programs downloaded to the communicating micro:bits! Any program changes need to be downloaded to all of the micro:bits.

Challenges:
1. Make an LED connected to another micro:bit flash a certain number of times.
2. Shake one micro:bit to display a random dice face on a second micro:bit
3. Tilt one micro:bit to control the motions of a machine containing another micro:bit.
4. Invent a way for micro:bits to send messages across greater distance than their normal range. *Clue*: Think about how telegraphy facilitated Western expansion in the United States.

New & Improved Graph a Mystery Picture #1 – lynxcoding.club

Secret Picture 1 Coordinates	Procedures	Secret Picture 2 Challenge

[1 3]
[2 2]
[3 2]
[2 1]
[3 0]
[2 0]
[2 -4]
[1 -6]
[-1 -7]
[0 -8]
[2 -9]
[-1 -9]
[-1 -8]
[-2 -7]
[-3 -7]
[-4 -8]
[-2 -9]
[-5 -9]
[-5 -8]
[-4 -7]
[-7 -5]
[-8 -3]
[-8 0]
[-6 -1]
[-3 -1]
[-2 0]
[-2 2]
[-1 3]
[1 3]
[1 4]
[0 3]
[-1 4]
[-1 3]
 	
These coordinate points will be used in a procedure to get the turtle to plot them for us.
	`to mypicture`
	`pu`
	`setpos [1 3]`
	`pd`
	`setpos [1 3] dot`
	`setpos [2 2] dot`
	`setpos [3 2] dot`
	`setpos [2 1] dot`
	`setpos [3 0] dot`
	`setpos [2 0] dot`
	`setpos [2 -4] dot`
	`setpos [1 -6] dot`
	`setpos [-1 -7] dot`
	`setpos [0 -8] dot`
	`setpos [2 -9] dot`
	`setpos [-1 -9] dot`
	`setpos [-1 -8] dot`
	`setpos [-2 -7] dot`
	`setpos [-3 -7] dot`
	`setpos [-4 -8] dot`
	`setpos [-2 -9] dot`
	`setpos [-5 -9] dot`
	`setpos [-5 -8] dot`
	`setpos [-4 -7] dot`
	`setpos [-7 -5] dot`
	`setpos [-8 -3] dot`
	`setpos [-8 0] dot`
	`setpos [-6 -1] dot`
	`setpos [-3 -1] dot`
	`setpos [-2 0] dot`
	`setpos [-2 2] dot`
	`setpos [-1 3] dot`
	`setpos [1 3] dot`
	`setpos [1 4] dot`
	`setpos [0 3] dot`
	`setpos [-1 4] dot`
	`setpos [-1 3] dot`
	`end`
	
	to dot
	setpensize 3
	pd fd 0
	setpensize 1
	end
		Can you write a new procedure to connect these points?
		
		Be sure that your procedure has a new name!
		
		[-6 -9]
		[-6 -8]
		[-7 -7]
		[-8 -5]
		[-8 -2]
		[-6 0]
		[-7 3]
		[-5 4]
		[-4 3]
		[-5 3]
		[-5 2]
		[-4 0]
		[-3 3]
		[-1 4]
		[0 3]
		[-1 3]
		[-2 2]
		[-2 0]
		[-1 0]
		[0 -1]
		[0 -2]
		[3 -2]
		[3 -5]
		[-2 -8]
		[-2 -9]
		[-6 -9]
		
		Here's a super version of `dot` for even cooler pictures! What does it do differently?
		
		`to dot`
		`setpensize 3 setc "red`
		`pd fd 0`
		`setpensize 1 setc "black`
		`end`

Mathematician's Mind-boggling Challenge

How can you make the turtle draw a larger version of your connect-the-dots graph?

Getting the Computer to Work for You

Lots of what kids experience while programming is a form of working for the computer. You formalize thinking and communicate precise directions to the computer. Once you develop a bit of programming fluency, the computer can work for you. Let's use the Lynx dialect of Logo for this activity. `http://lynxcoding.club`

In this context, the most obvious shift is in reducing the amount of typing a user must do to get a computer to graph a nonspecific number of coordinate points.

The First Simplification

The `mypicture` procedure has an obvious and immediate need to simplification. Anytime you see a repetitive pattern or set of similar instructions used multiple times, it may be a good time to consider writing a helper procedure to do more of the work. `Setpos [ x  y ] dot` is a great example. Let's create a `go` procedure that takes a position (expressed as a list of two numbers representing *x* and *y*) as input.

```
to go :point
setpos :point
dot
end
```

Now, your `mypicture` procedure can look like this.

```
to mypicture
pu
setpos [1 3]
pd
go [1 3]
go [2 2]
go [3 2]
.
.
.
end
```

That's Still Too Much Work!

Using list processing, we can eat through a list of coordinate points and graph them in sequence, without even knowing how many points there are. The following *recursive* procedure does this.

```
to grapher :list
if empty? :list [stop]
go first :list
grapher bf :list
end
```

`Grapher` takes a list as input, goes to the first set of coordinates in the list, and then runs the same procedure again, but this time passes everything but the first item in the list (a pair of coordinates) back to the `grapher` procedure. The points are a list of lists.

The `if empty? :list [stop]` line is called a *stop rule*. The stop rule tells the computer to stop the procedure as soon as the list of inputs is empty. A programming line such as this is used in countless contexts. Test it out in the command center.

```
Grapher [[0 0] [0 50] [50 50] [50 0]]
```
or
```
Grapher [[0 0] [0 50] [50 50] [50 0][35 35]]
```

The stop rule allows you to use as many or few elements in the list input to the procedure.

My Picture is Too Small!

Now that we've written a procedure to eat through a list of any size and graph the coordinates found in the list, we need a way to change the scale of the drawing. The first obvious improvement to the `grapher` procedure is to add an input for scale.

```
to supergrapher :list :scale
if empty? :list [stop]
.
.
end
```

So far, so good. Our procedure has two inputs and a stop rule, but where's the beef?

We need to take apart our list of coordinates, multiply each element by the scale, and then put the coordinates back together so the turtle may be sent to that new place on the screen.

```
to supergrapher :list :scale
if empty? :list [stop]
setpos list (first first :list) * :scale (last first :list) * :scale
dot
supergrapher bf :list :scale
end
```

If you don't understand what's happening in the setpos line of this procedure, try the following in the command center.

```
show first first [[10 20] [30 40] [50 60]]
show (first first [[10 20] [30 40] [50 60]]) *5
show list (first first [[10 20] [30 40] [50 60]]) * 5 (last first
[[10 20] [30 40] [50 60]]) * 5
show (list (first first [[10 20] [30 40] [50 60]]) * 5 (last first
[[10 20] [30 40] [50 60]]) * 5)
```

What happened?

`list`'s job is to put two items together and report them as a list, like `sentence`, if you have more than two elements, put the entire expression in parentheses (   ) in order to create a list of many items.

Now graph like crazy with the new `supergrapher`!

```
Supergrapher [[-6 -9] [-6 -8] [-7 -7] [-8 -5] [-8 -2] [-6 0] [-7 3]
[-5 4] [-4 3] [-5 3] [-5 2] [-4 0] [-3 3] [-1 4] [0 3] [-1 3] [-2 2]
[-2 0] [-1 0] [0 -1] [0 -2] [3 -2] [3 -5] [-2 -8] [-2 -9] [-6 -9]] 5
```

```
Supergrapher [[-6 -9] [-6 -8] [-7 -7] [-8 -5] [-8 -2] [-6 0] [-7 3]
[-5 4] [-4 3] [-5 3] [-5 2] [-4 0] [-3 3] [-1 4] [0 3] [-1 3] [-2 2]
[-2 0] [-1 0] [0 -1] [0 -2] [3 -2] [3 -5] [-2 -8] [-2 -9] [-6 -9]] -
10
```

If you just want to play with the newly discovered chicken, you could write a procedure like this one.

```
to superchicken :scale
supergrapher [[ 1 3 ] [ 2 2 ] [ 3 2 ] [ 2 1 ] [ 3 0 ] [ 2 0 ] [ 2 -4
] [ 1 -6 ] [ -1 -7 ] [ 0 -8 ] [ 2 -9 ] [ -1 -9 ] [ -1 -8 ] [ -2 -7 ]
[ -3 -7 ] [ -4 -8 ] [ -2 -9 ] [ -5 -9 ] [ -5 -8 ] [ -4 -7 ] [ -7 -5
] [ -8 -3 ] [ -8 0 ] [ -6 -1 ] [ -3 -1 ] [ -2 0 ] [ -2 2 ] [ -1 3 ]
[ 1 3 ] [ 1 4 ] [ 0 3 ] [ -1 4 ] [ -1 3 ]] :scale
end
```

Try

```
cg superchicken 5
cg superchicken 2
cg superchicken 20
```

You may find that specifying the scale after a long list aesthetically displeasing. That's an easypeasy fix. Just reverse the inputs in the procedure and recursive call.

```
to supergrapher :scale :list
if empty? :list [stop]
setpos list (first first :list) * :scale (last first :list) * :scale
dot
supergrapher :scale bf :list
end
```

Now try this new and improved version!

```
Supergrapher -10 [[-6 -9] [-6 -8] [-7 -7] [-8 -5] [-8 -2] [-6 0] [-7
3] [-5 4] [-4 3] [-5 3] [-5 2] [-4 0] [-3 3] [-1 4] [0 3] [-1 3] [-2
2] [-2 0] [-1 0] [0 -1] [0 -2] [3 -2] [3 -5] [-2 -8] [-2 -9] [-6 -
9]]
```

If you want to simplify this process for other users, you can create a visual slider called something like, *magnification*, on the screen (if using Lynx) and then create a button set to run a procedure, such as the following.

```
to sc2
supergrapher coordinates magnification
```

```
end
```

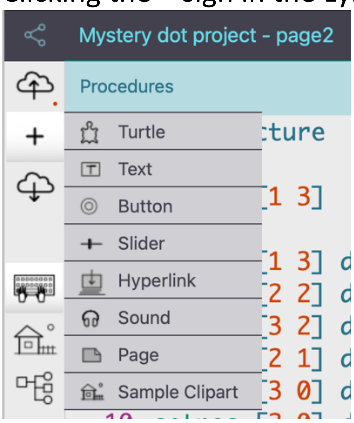
The coordinates procedure is a reporter that just outputs the list of coordinates you're using as an input to other procedures, over and over again.

In this case, the coordinates are the points used to create the original tiny chicken.

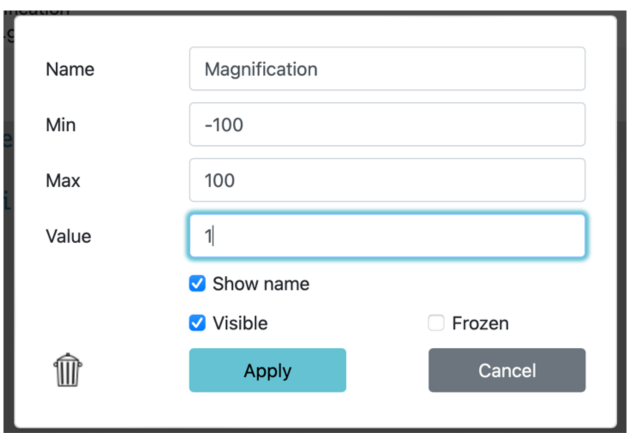
```
to coordinates output [[1 3] [2 2] [3 2] [2 1] [3 0] [2 0] [2 -4]
[1 -6] [-1 -7] [0 -8] [2 -9] [-1 -9] [-1 -8] [-2 -7] [-3 -7] [-4 -8]
[-2 -9] [-5 -9] [-5 -8] [-4 -7] [-7 -5] [-8 -3] [-8 0] [-6 -1]
[-3 -1] [-2 0] [-2 2] [-1 3] [1 3] [1 4] [0 3] [-1 4] [-1 3]]
end
```

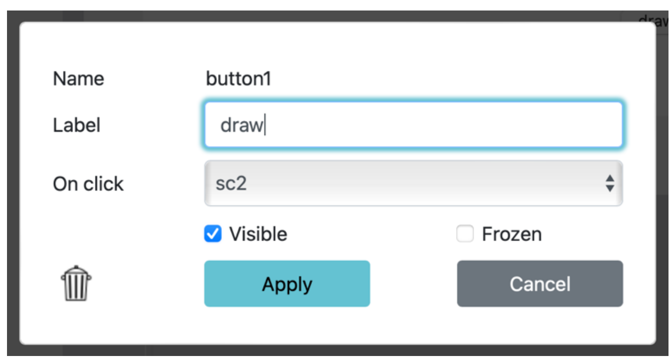
`Magnification` is the name of a slider created by:

- Clicking the + sign in the Lynx tool palette



- Name the slider, set its range, and starting value.





- Next, create a new button from the same tool palette.
- Label the button `draw`
- Choose the `sc2` procedure to run when the button is clicked

Change the value of the slider by clicking and dragging and then click on the draw button to see what happens. Try this several times.

A version of this nonsense is online in Lynx at https://lynxcoding.club/share/NfV6z86P.

List Processing is Universal!

The sort of code found in `grapher` or `supergrapher` are used in countless contexts! Understand it and you can get the computer to work for you in innumerable ways. Here's an example of manipulating computer music.

`Note` takes two inputs, pitch & duration.

Let's say that `Note 60 4` plays a C for a count of 4
`Note 62 4` plays a D for 4 counts
`Note 64 4` plays an E for 4 counts

If we had a play procedure that could take a list of notes and durations, we could manipulate the music just like a composer!

`Play` [[60 4] [62 4] [64 4]]

```
to play :music
If empty? :music [stop]
Note first first :music last first :music
play bf :music
end
```

Another version of this procedure could speed up or slow down the music, just like we did in `supergrapher`.

```
to play :music :tempo
If empty? :music [stop]
note (first first :music) (:tempo * (last first :music))
play bf :music :tempo
end
```

`Play` [[60 4] [62 4] [64 4]] 1
Plays at normal tempo
`Play` [[60 4] [62 4] [64 4]] 2
Plays the same notes twice as slow
`Play` [[60 4] [62 4] [64 4]].5
Plays the same notes twice as fast

Can you add an input to play for *`transposition`*? Its job is to change the pitch by a numerical value (+ or -).

Can you write a procedure that will play a list of notes and durations backwards, or in musical parlance, retrograde?

If you get tired of typing the list of values over and over again, put the computer to work. Create a procedure like the following.

```
to music
output [[60 4] [62 4] [64 4]]
end
```

Then run the following instruction:

```
play music .5
```

or

to points
output [[-6 -9] [-6 -8] [-7 -7] [-8 -5] [-8 -2] [-6 0] [-7 3] [-5 4] [-4 3] [-5 3] [-5 2] [-4 0] [-3 3] [-1 4] [0 3] [-1 3] [-2 2] [-2 0] [-1 0] [0 -1] [0 -2] [3 -2] [3 -5] [-2 -8] [-2 -9] [-6 -9]]
end

Then run the following instruction:

`supergrapher 10 points` or `supergrapher points 10`, depending on which order you decided to use the inputs in your procedure.

The Big Idea

Versions of the grapher/supergrapher procedure are used in music composition, encryption, codes & cryptography, art, and data manipulation of all kinds. If you understand these fundamental list processing techniques of “eating” and “smushing,” you can solve a world of problems and put the turtle to work for you.

To use Turtle Art, go to

http://playfulinvention.com/webturtleart

For Turtle Art resources, go to

constructingmodernknowledge.com/new-turtle-art-cards/

Making Polygons

Super Dooper Really Really Really Hard Challenge

Name	# of sides	amount of turn
Triangle	3	
Square	4	90
Pentagon	5	
	6	
	7	
Octagon	8	
	9	
	10	36
	11	
	12	

Change the number of sides and and amount of the turn to create the polygons.

Turtle Art - Playing with Arithmetic

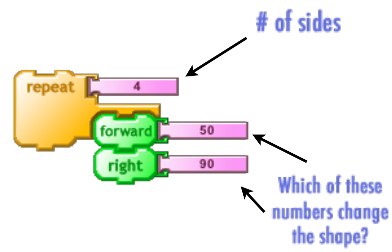
Problem 1
Create the following program:

Can you predict what it will do before you run it?

What does it do?

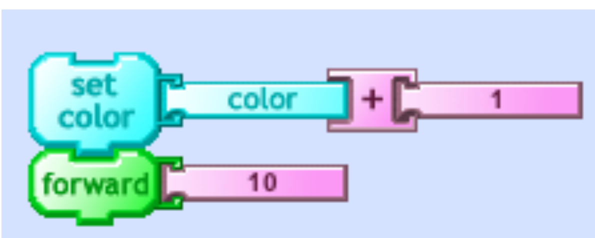
What happens if you change the number 1 to another number?

What happens if you change the **X** to **+, - or /** ?



Answer to other side

Problem 2
Create the following program:



Can you predict what it will do before you run this program?

How does it work?

What happens if you replace the 1 with a larger number, say 10?

When you increase the pen color by 1, does the color get lighter or darker?

What happens if you place a repeat block at the top of the program?

Problem 3
Here's a crazy idea!

What do you predict will happen if you combine program 1 and program 2? Snap them together and fine out!

Thinking About ~~cheating at~~ Tricky Pattern Blocks in Turtle Art

Goal
Write a Turtle Art procedure to draw each of the shapes in a set of pattern blocks! In other words, teach the turtle to draw all of the shapes in a set of pattern blocks.

Suggested Strategies
Think about the shape you want the turtle to draw

How many sides are there? Is there a mirror image?

Look for patterns

Are any of the turns/corners ones you have seen before?
Are all of the sides equal? Are some longer than others? If so, by how much?

Try numbers you know

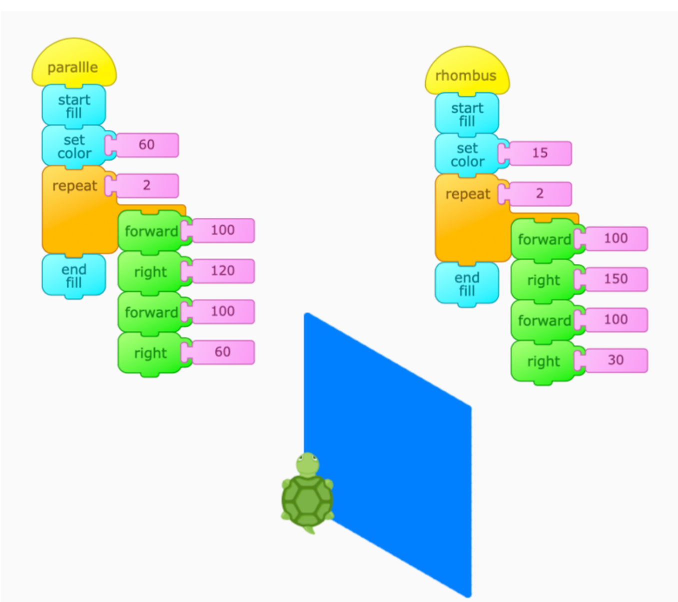
Start with simple numbers for right or left turns. Numbers ending in 0 or 5 often do the trick (those are multiples of 5 or 10). For example, 90, 120, 30, 150, 60, 45 are some of the numbers we have used to turn the turtle.

Hide the turtle to see if the shape is drawn perfectly

You should not see overlapping lines or gaps in the shape.

I really like when the turtle returns to where it started drawing a shape and pointed in its original direction. That's why I use `FORWARD RIGHT or FORWARD LEFT instead of RIGHT FORWARD or LEFT FORWARD.`

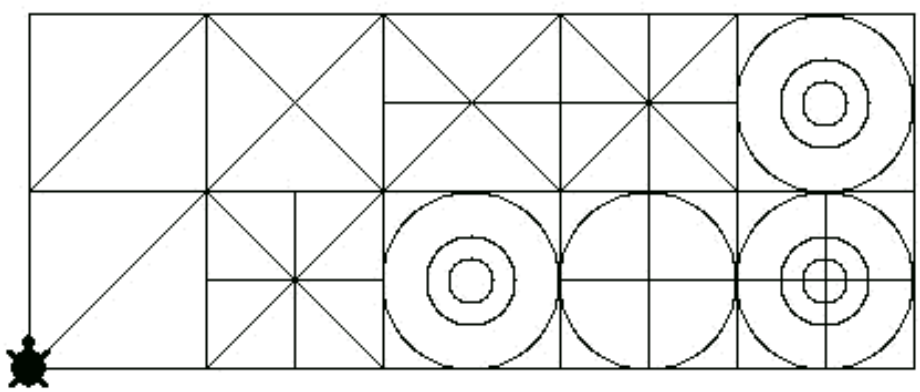
Here are two of the shapes we figured out together. Do you see any patterns?



Challenge
Figure out a way to use the shapes you created to make patterns on the computer screen with your new procedures. You might even think of this as creating art software for little kids to play with.

Turtle Art Quilt Project

An adventure in creativity



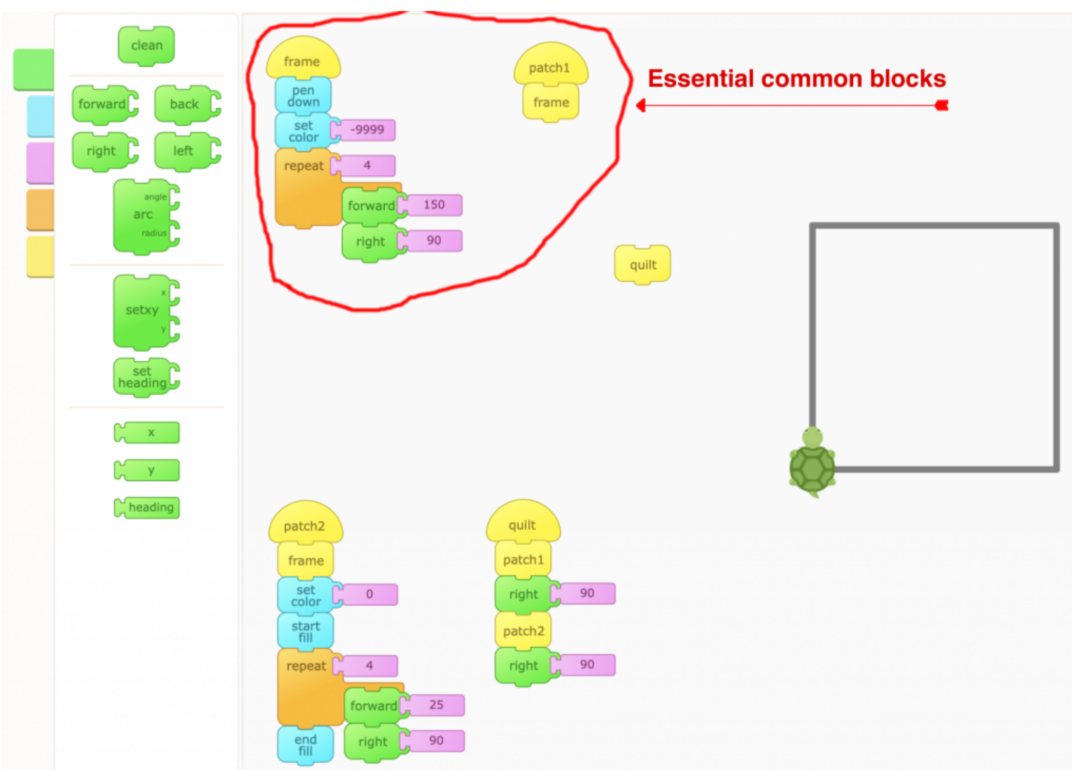
Yours will be much prettier of course!

Goal

Design one or more quilt patches that may be combined with classmates or used to create your own screen quilts in Turtle Art.

Instructions

Start with these blocks. Everyone needs to use these instructions as a common starting place.



Each quilt project needs these fundamental building blocks

Design a procedure named *patch1* to draw a design completely within the frame and be sure that the turtle returns to where it begins with the same orientation as when the procedure started. The frame is a square with sides of *150* turtle steps.

Extra credit

If you are satisfied with *patch1*, design procedures for new and different quilt patches. Name them *patch2, patch3,* etc…

Each patch needs to begin with *frame*.

Extreme Arts and Crafts Challenge

Create a *quilt* procedure that assembles one or more of your patches into a beautiful quilt design. You are of course free to repeat the use of a patch or use a variety of them.

Note: color *-9999* is black in Turtle Art

Remember

- Save often!
- Each patch needs to begin with *frame*.

THE 3N PROBLEM

Gary S. Stager

SCENARIO

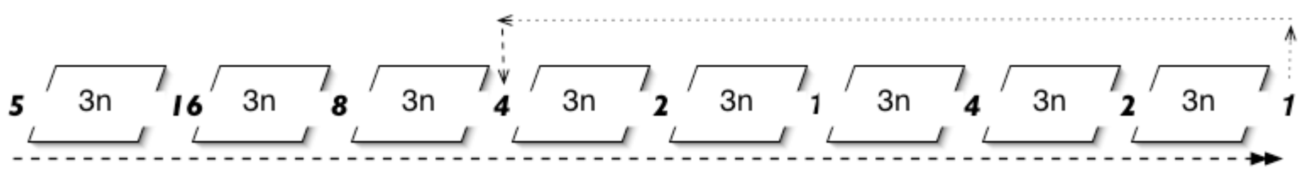
You and your noted mathematician colleagues convene in Geneva to present brilliant theories pertaining to one of the world's great mysteries, the elusive 3n Problem.

BACKGROUND

The 3N problem offers a fantastic world of exploration for students of all ages. The problem is known by several other names, including: Ulam's problem, the Hailstone problem, the Syracuse problem, Kakutani's problem, Hasse's algorithm, Thwaite's Conjecture 3X+1 Mapping and the Collatz problem.

The 3N problem has a simple set of rules. Put a positive integer (1, 2, 3, etc…) in a "machine." If the number is even, cut in half - if it is odd, multiply it by 3 and add 1. Then put the resulting value back through the machine. For example, 5 becomes 16, 16 becomes 8, becomes 4, 4 becomes 2, 2 becomes 1, and 1 becomes 4. Mathematicians have observed that any number placed into the machine will eventually be reduced to a repeating pattern of 4...2...1...

This observation has yet to be proven since only a few billion integers have been tested. The 4…2…1… pattern therefore remains a conjecture.

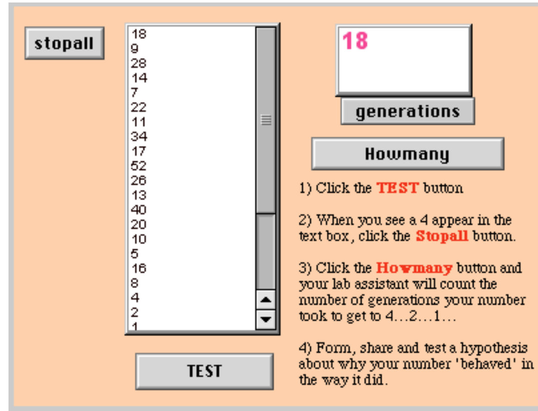


The computer will serve as your lab assistant – smart enough to work hard without sleep, food or pay, but not so smart that it does the thinking for you.

USING THE COMPUTER

1. Point your browser to http://constructingmodernknowledge.com/3n
    a. You may edit the project or look at the code here.
2. Click the *Test* button
3. Enter a positive integer > 0 and click OK

4. As soon as you see the pattern 4…2…1… appear in the data window, click the STOPALL button
5. Click the *Howmany* button and the computer will count many "generations" that number took to reach the repeating pattern.
6. The count will appear in the *generations* window.
7. Think about the results. Record your data and test another number.
8. Repeat steps 1-7



YOUR CHALLENGE

- Work with your teammates to find numbers that take a "long time" to get to the repeating pattern of 4…2…1…
- How did you go discover a number that took a "long time?"
- What is a long time?
- Use any tools at your disposal to learn more about the problem and to record or analyze your data.
- Share your hypotheses with the assembled "conference delegates."
- Defend your hypotheses.
- Disprove the hypotheses of other delegates.

EXTRA TOOLS TO MAKE YOU SAY, "HMMM…"

- Click on the arrow taking you to the web page, http://www.stager.org/3n/3ntools.html
- The first screen is similar to the 3n tools you've been using
- Click on the *Overnight* button to ask your virtual lab assistant to keep track of numbers that take more than a specific number of generations. You may adjust the generations *slider* based on what you determine to be a "long time" and click on the *Experiment* button to specify the number you wish to start with. This tool will then try every number after the value you specify until you stop it.
- Clicking on the *Graph* button will take you to a set of tools designed to graph the number of generations taken by each number in a series beginning with the number you specify. Does the graph tell a story?

DEBRIEFING QUESTIONS

- What did you learn from this experience?
- What did you observe about the learning style(s) of your collaborators?
- Which subject(s) does this project address?
- What might a student learn from this project?
- For age/grade is this project best suited?
- What would a student have to know before successfully engaging in this project?

The tools used in this activity were created using Lynx, a wonderful environment for multimedia authoring, modeling, robotics, animation and exploring powerful ideas. With Lynx, you could customize my tools or build your own. Go to `lynxcoding.club` for more information.

Getting Started with Wolfram Alpha and Wolfram Language

Wolfram Alpha
http://wolframalpha.com

In Wolfram Alpha, click on Open code to open the equation in Wolfram Language

Create a new notebook in Wolfram [Language] Programming Lab
http://lab.open.wolframcloud.com/app/

Fast Introduction to Wolfram Language for Math Students
http://www.wolfram.com/language/fast-introduction-for-math-students

Fast Introduction to Wolfram Language for Math Students

http://www.wolfram.com/language/fast-introduction-for-math-students

Fast Introduction to Wolfram Language for Programmers

http://www.wolfram.com/language/fast-introduction-for-programmers

An Elementary Introduction to Wolfram Language [book]
http://www.wolfram.com/language/elementary-introduction/2nd-ed/ [ebook]
http://amzn.to/2BDhf4B [softcover]

Going from Wolfram Alpha to Wolfram Language - *Launching Wolfram|Alpha Open Code*
http://blog.stephenwolfram.com/2016/12/launching-wolframalpha-open-code/

Important Articles by Stephen Wolfram
What is a Computational Essay?
http://blog.stephenwolfram.com/2017/11/what-is-a-computational-essay/

How to Teach Computational Thinking
http://blog.stephenwolfram.com/2016/09/how-to-teach-computational-thinking/

Wolfram Development Platform [for deploying your Wolfram Language apps]
https://develop.open.wolframcloud.com/app/

Computational Thinking Project Ideas from Wolfram
http://www.computationinitiative.org/resources/teaching/

Classic Steven Levy Wired Profile of Stephen Wolfram
https://www.wired.com/2002/06/wolfram/

Videos
Conrad Wolfram's TED Talk
https://www.youtube.com/watch?v=60OVlfAUPJg

Stephen Wolfram's Intro to Wolfram Language
https://www.youtube.com/watch?v=_P9HqHVPeik

Making Programming Accessible to Everyone with Wolfram Language
https://www.youtube.com/watch?v=ALuQzgDvr2g

Stephen Wolfram in Conversation with Howard Gardner at the Harvard Askwith Forum 11/6/2017
https://youtu.be/sJronwbyFeM

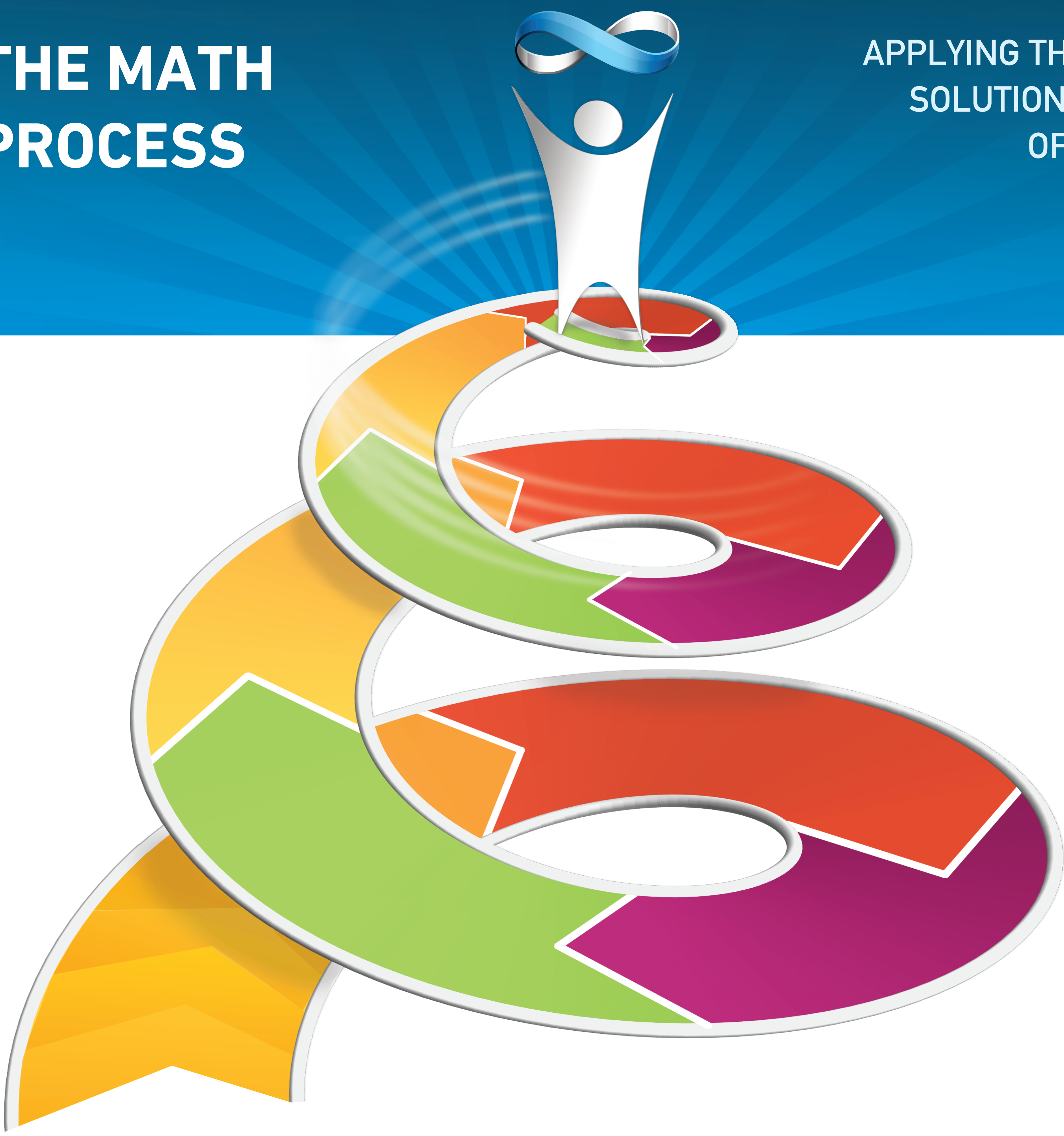
Stephen Wolfram Computational Universe in the MIT Artificial General Intelligence class lecture (March 2018)
https://youtu.be/P7kX7BuHSFI

All of these links are live at cmkfutures.com/wolfram

Curated by Gary Stager – cmkfutures.com/gary

THE MATH PROCESS

APPLYING THE CBM SOLUTION HELIX OF MATH



QUESTIONS

MATH MODELS

MATH ANSWERS

NEW KNOWLEDGE

Think through the scope and details of the problem; define manageable questions to tackle.

Prepare the questions as math models ready for computing the answer. Select from standard techniques or formulate algorithms.

Transform the math models into math answers with the power of computers, or by hand-calculating. Identify and resolve operational issues during the computation.

Did the math answers solve the original problem? Fix mistakes or refine by taking another turn around the Solution Helix.

DEFINE QUESTIONS

TRANSLATE TO MATH

COMPUTE ANSWERS

INTERPRET RESULT

Why Use Math?

Because it's the most powerful way to get answers to a wide range of real-world questions. Several factors contribute to math's power. One is its ability to describe a large number of apparently different situations in precise and standardized ways. Another is because these descriptions come with highly effective methods for working out, or "computing," answers. Math may look cryptic but it's by this "abstraction" from the problem at hand that the same methods can be reused and refined on so many different problems. Math also scales well. Whizz around the CBM Solution Helix in a few seconds for everyday problems like "How fast do I need to go?", or apply it over years at the cutting edge of research to solve problems like "How can I make a car go 1000 mph?"

What Is Computation?

Clearly defined procedures backed up by proven logic for transforming math questions into math answers. For hundreds of years, computation was limited by humans' ability to perform it. Now computers have mechanized computation beyond previous imagination, scaling up to billions of calculations per second, powering math into transforming our societies.

Computer-Based Math (CBM)...

...is building a completely new math curriculum with computer-based computation at its heart, while campaigning at all levels to redefine math education away from historical hand-calculating techniques and toward real-life problem-solving situations that drive high-concept math understanding and experience.

Go interactive with the CBM Solution Helix or get this poster at: **computerbasedmath.org/helix**

computerbasedmath.org

More computational possibilities to explore

MakeCode Arcade

Design, program, and play your own video games on the screen or handheld device.

1. Go to arcade.makecode.com/
2. Try one of the tutorials
3. Embellish or improve the game
4. Play it on the screen or download it to a handheld game system like the Meowbit.

Go to https://inventtolearn.com/program-your-own-gameboy/ for hardware and software resources

MicroBlocks

Microblocks is an ingenious live-coding block-based programming environment you should experience.

1. Go to microblocks.fun
2. Follow the Getting Started instructions
3. MicroBlocks looks a lot like MakeCode. How is it different?
4. What are some advantages of live-coding?

You may also explore programming the micro:bit in Scratch. Go to scratch.mit.edu/microbit to get started.

Turtle Art

Block-based Logo dialect focused on communicating geometric and computational ideas to the computer in pursuit of creating beautiful art.

1. Turtle Art software - playfulinvention.com/webturtleart
2. Visit the Turtle Art resources section of inventtolearn.com/turtleart for links to software, activity cards, and teaching ideas.

Wolfram Language

May just be the future of computing, already powering most serious scientific and mathematical research. Infinite and untapped potential. Wolfram Language powers Wolfram Alpha & Mathematica.

1. Go to inventtolearn.com/wolfram for getting started tutorials, resources, videos, and links to the software.

Continue learning long after the workshop with resources found at inventtolearn.com/questions.

Email gary@stager.org or sylvia@inventtolearn.com to schedule school-based professional development opportunities.

A Math Game Only a Mother Could Love

Version 1

Create two textboxes on the Lynx page. One should be named, Correct and the other should be named Incorrect.

```
to number1
output random 11
end
```

`Number1` will report a random number between `0` and `10`. If you want a number from `1-10`, you say `output 1 + random 10`

```
to number2
output random 11
end
```

`number1` or `number2` may be different in case you want to practice one times table or another.

If you wish to practice a particular "table" change the number1 procedure to `output 5`, if you want to practice your 5 times tables.

Try this line a few times and see what it does.

```
Show (list number1 "* number2)
```

It should make a multiplication problem

```
to quiz
askquestion (list number1 "* number2)

end
```

```
to askquestion :problem
question :problem
ifelse answer = run :problem [setcorrect correct + 1]
[setincorrect incorrect + 1]
end
```

Can you add an announcement with ANNOUNCE, sound-effect or animation when the user answers correctly or incorrectly?

```
to game
setup
repeat 10 [quiz]
end

to setup
setcorrect 0
setincorrect 0
end
```

Can you figure out a way to randomly select the arithmetic operation [+ – * /] ? Hint: `PICK` may be useful here.

Can you figure out a way to display a score (perhaps based on percentage of correct answers) on the page? Hint: You'll need a `score` textbox.

`GAME` is the superprocedure that makes everything work. You may wish to make a button to run the `GAME` instruction.

Version 2 - Timed game

Change the following procedures

```
to game
setup
resett
repeat 10 [quiz]
end
```

`resett` resets the program's clock to 0.

```
to quiz
if timer > 600 [Announce [Time’s up!] stopall]
askquestion (list number1 "* number2)
quiz
end
```

`timer` counts in tenths of a second. So, 10 = 1 second. 600 = 1 minute. You may use any number you wish in the quiz procedure.

The `quiz` procedure now runs over and over again until the time is up and then `stopall` stops all processes.

Introduction to Lynx Words, Lists and Ciphers

Gary Stager, Ph.D.

Words in Lynx begin with quotation marks as in:

```
show “Gary
```

Lynx lists are a collection of words or other lists, such as:

```
show [lemon grape [apple pie] strawberry]
```

The list above has 4 elements, 3 words and 1 list. A good deal of computer programming involves taking things apart and putting things together. In this activity, we will take things apart.

1) `ASCII` is a reporter. Try typing the following in the command center:

```
show ascii “a
show ascii “e
show ascii “z
```

What does `ASCII` do? ______________________________________

2) `CHAR` is another reporter. Try the following in the command center:

```
show char 97
show char 98
show char 111
```

What does `CHAR` do? _____________________________

If `ascii “a = 97`, how can we change that number to equal 1?

97 ☐ _____ = 1

3) Try the following in the command center:

show first “apple
show last “apple

What does the reporter, `first`, do?

What does the reporter, `last`, do?

4) Predict what each of these instructions will do before you try them.

```
show first [apple peach pear]
show last [apple peach pear]
```

How accurate were your predictions?

5) What do you think will happen if you type the following? Make a prediction and then run the instructions in the command center. Write the results next to the instruction.

```
show first first [apple peach pear]
show last first [apple peach pear]
show first last [apple peach pear]
show last last [apple peach pear]
```

How accurate were your predictions?

6) Predict the result of the following instructions before typing them into the command center. Write the results next to the instruction.

```
show bf “lemon
show bl “grape
show bf bf “grape
show bl bf “grape
show bf bl “grape
show bl bl “grape
```

What does bf do? ___________________________

What does bl do? ___________________________

7) Predict the result of the following instructions before typing them into the command center. Write the results next to the instruction.

```
show bf [apple grape peach]
show bl [apple grape peach]
show bf bf [apple grape peach]
show bl bf [apple grape peach]
show bf bl [apple grape peach]
show bl bl [apple grape peach]
show bf bl [apple grape peach]
```

8) Predict the result of the following instructions before typing them into the command center. Write the results next to the instruction.

```
show first bf “grape
show first bl “grape
show last bf bf [apple grape peach]
show first bl bf [apple grape peach]
show first bf bl “grape
show first bl bl “grape
show last bf bl “grape
```

9) Type this procedure in the procedures center:

```
to eat :thing

show first :thing
eat bf :thing
end
```

Try running the procedure above by typing the following in the command center:

```
eat "lemon
eat [apple peach grape lemon]
```

An error message, first does not like as input in eat, is generated. It means that the procedure tried to grab the first thing out of nothing after you ate all of the other items in the word or list. Therefore, we need a common instruction, called a *stop rule* added to the procedure.

Change the `eat` procedure in the procedures center to include the the stop rule (beginning with `IF`)

```
to eat :thing
if empty? :thing [stop]
show first :thing
eat bf :thing
end
```

Try running the procedure above by typing the following in the command center:

```
eat "lemon
eat [apple peach grape lemon]
```

Is the error message gone?

10) Caesar's Cipher Level 1

```
to caesar :word
if empty? :word [stop]
show (ascii first :word) - 96
caesar bf :word
end
```

Try running the procedure above by typing the following in the command center:

```
caesar "touchdown
caesar "school
```

11) Think about how we should improve our cipher program!

Gary's Fraction Challenge

Make a procedure called PIE that will divide a circle into a fraction indicated by two inputs to the procedure, PIE. Below are some procedures to get you started.

`Fraction 2 3`

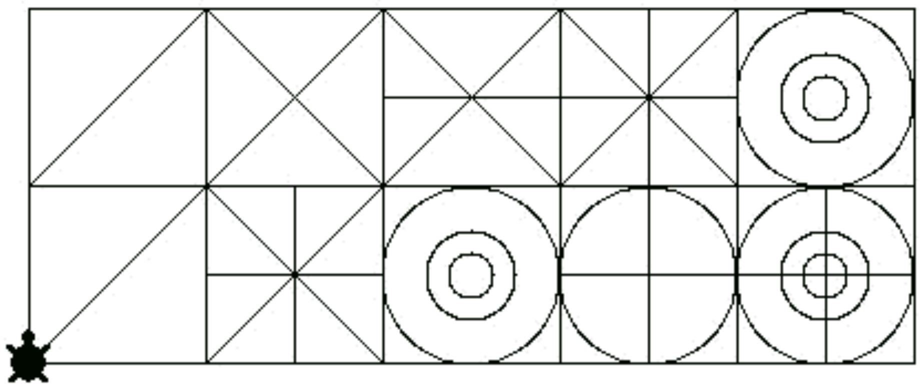
```
to rectangle
pd
repeat 2 [fd 50 rt 90
fd 300 rt 90]
end

to rec :length
repeat 2 [fd 50 rt 90 fd :length rt
90]
end


to fraction :n :d
rectangle
repeat :d [rt 90 fd 300 / :d lt 90
fd 50 bk 50]
lt 90 fd 300 rt 90
repeat :n [fillit pu rt 90 fd 300 /
:d lt 90]
rt 90
bk :n / :d * 300 lt 90
end

to fillit
setc "red
pu
rt 45
fd 2
fill
bk 2
lt 45
setc "black
end
```

Or, in this version, you would type **`fraction 400 3 5`** to draw a rectangle with a length of 400 divided into 3/5

```
to rectangle :length
pd
repeat 2 [fd 50 rt 90
fd :length rt 90]
end

to rec :length
repeat 2 [fd 50 rt 90 fd
:length rt 90]
end

to fraction :l :n :d
rectangle
repeat :d [rt 90 fd :l / :d
lt 90 fd 50 bk 50]
lt 90 fd :l rt 90
repeat :n [fillit pu rt 90
fd :l / :d lt 90]
rt 90
bk :n / :d * :l lt 90
end

to fillit
setc "red
pu
rt 45
fd 2
fill
bk 2
lt 45
setc "black
end
```

Logo Quilt Project

An adventure in creativity using Lynx



Yours will be much prettier of course!

Objective

You will each contribute to a collaborative quilt, programmed in Logo and drawn by the turtle. This is a classic Logo project modified to use a new web-based dialect of Logo called Lynx. (http:// lynxcoding.club)

 [1]  [2]  [3]

Quilting as a craft or art form dates back to ancient Egypt. Quilt making was not only functional as a way of manufacturing blankets, but a collaborative form of expression embraced by Native American, African American, and Amish communities in the United States dating back hundreds of years. There are many styles of quilts, but the combining of different fabric scraps or pieces of uniform size combined to create elaborate

[1] "quilt" by Enid H. W. is licensed under CC BY-NC 2.0

[2] "quilt top" by madelinetosh is licensed under CC BY-NC-ND 2.0

[3] "Alphabet quilt" by tirralirra is licensed under CC BY-NC 2.0

geometric patterns lends itself to Logo programming (and constructionism). Quilting traditions may be found in cultures across the globe.

In this project, each of you will be responsible for creating at least one “patch” that will then be shared with your peers. Each of you will then take some of those square patches and assemble a quilt made of them.

Getting started with Logo

The turtle is a metaphor for yourself. When you give it instructions, the turtle does exactly what you tell it to do. If your instructions were inaccurate or wrong, you will either receive an error message or the result of your instruction will be different than what you anticipated. In either case, you need to debug.

The words built into the Logo vocabulary are called *primitives*. Multiple instructions may be run in sequence from the command center of Lynx as long as there are spaces between the words and numbers.

One of the powerful ideas of Logo is that once you figure out how to do something, you can “teach Logo” or “teach the turtle” a new word that remembers that sequence of instructions. These new words are called *procedures*. Procedures behave exactly like primitives except they are unique to a particular project. In other words, user created procedures are available to use as long as they are defined in that project (file).

Procedures are defined in the procedure pane in Lynx. They always begin with the word, *TO,* and end with the word, *END.* Capitalization is never an issue in Logo.

For example, type *foo* in the Command Center and Logo will present the error message, *I don’t know how to foo.*

We can define foo by typing the following instructions:

```
to foo
fd 57 rt 144
end
```

Now type `cg pd foo` in the command center and hit enter/return.

CG clears the screen and puts the turtle in the center of the screen. `PD` puts the turtle’s pen down. The turtle has a pen stuck in its belly button and when it is down and you command it to move, it leaves a trail. `FD` is the command for forward and it takes a number of turtle steps as its input.

Think of procedure names as infinitive verbs. They produced action when used in Logo.

Procedures and primitives may be combined to create new procedures. Procedures are like building blocks that perform a function and may be combined in infinite variety to produce complexity. Procedures used in other procedures are sometimes called *subprocedures.* There is no limit to the number of procedures you may write. They just all need to be typed in the Lynx procedure pane and follow the rule of beginning with `to` and ending with `end`.

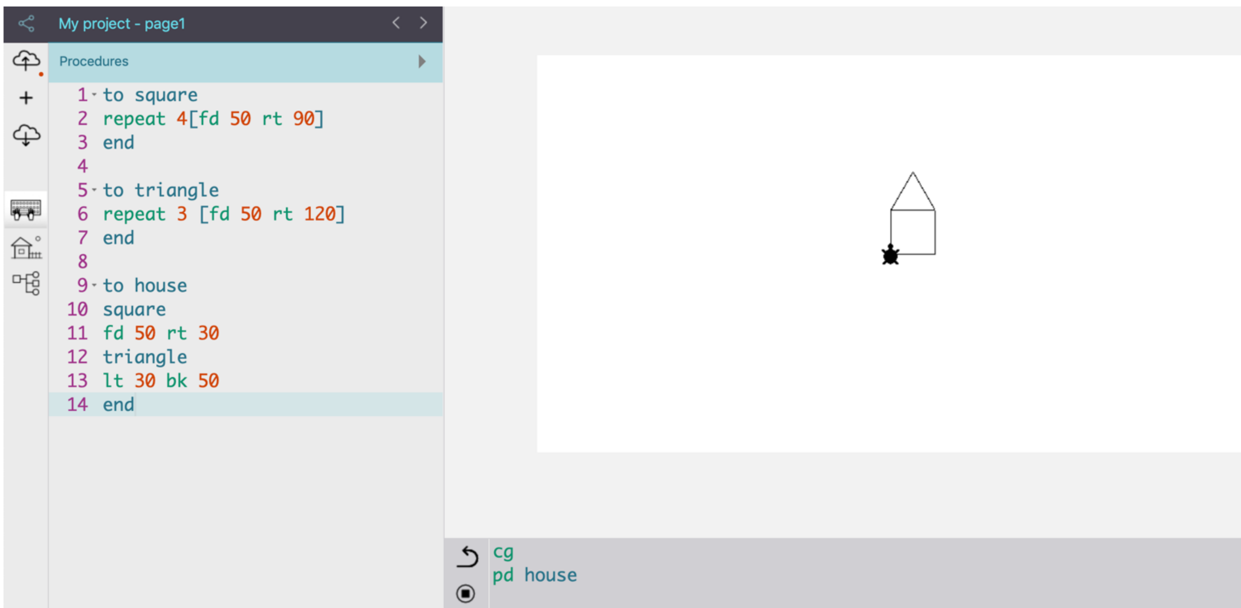
Next, add the following procedure to your procedure pane.

```
to foobar
repeat 5[foo]
end
```

Run `foobar` in the command center. What happened? What does `repeat` do?

Writing and Running Procedures

A procedure is a list of instructions with a name. All procedures begin with `to` and end with `end`.



The order in which procedures are created in the procedure pane does not matter as long as all of the procedures are formatted properly, beginning with `to` and ending with `end`. Putting a blank line between procedures makes them easier to read and debug.

Let's start programming!

1) Each of you must open Lynx, start a new project, name the project, and then type the following procedure into the procedures area.

```
to frame
setcolor "black
pd
repeat 4 [fd 100 rt 90]
end
```

Can you predict what this procedure will do before running it in the command center?

A list of colors the turtle knows may be found here.

2) Next, create a new procedure that is named with your name and perhaps a number (in case you create more than one patch). Each patch will begin with the command, *`frame`*. Then you will tell the turtle what to draw within the constraints of the `patch` (square).

For example:

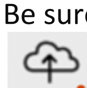
`to gary1`
`frame`
 	
`end`
	`to jose`
	`frame`
	
	`end`
		`to yumi`
		`frame`
		
		`end`

Important rule!
Everything the turtle draws in your patch must be within the square AND the turtle must return to where it began facing in the same direction. Returning to where you began is called *state transparency* in computer science. It is important for making the patches flexible and portable in this project.

3) Use cg patch and then a series of commands in the command center to design a pattern within the square and return the turtle to where it began. Then copy and paste those instructions into a new procedure, for example:

```
to maria1
frame
rt 45 fd 50 bk 50 lt 45
end
```

4) Create as many quilt patches as you can design. Be sure that each procedure has a unique name.

5) Save your project to the cloud by clicking on the  button in Lynx.

6) Copy and paste your procedures (as text) and share them with your friends via email or posting in a collaborative space.

Make a Quilt!

1) Copy and paste the procedures from your friends into your Lynx procedures. (Make sure that there are no duplicate procedure names. Rename some if necessary. You will only need one `patch` procedure since you are all starting with the same one.

2) Try your friends' procedures and see how they look.

3) Decide which of these patches you wish to assemble into your own quilt.

4) Figure out how to assemble the quilt using the patch procedures and other turtle graphics commands.

5) You should use at least four patches in a quilt.

6) Write a new quilt procedure to automatically draw your new quilt!

7) Save your work to the cloud.

8) Share the project with friends by clicking on the  button and sharing the URL via email or collaborative space.

Here is a sample Quilt project

All of these procedures should be in the Lynx procedure pane if you wish to try our sample `quilt`	

`to frame`
`setcolor "black`
`pd`
`repeat 4 [fd 100 rt 90]`
`end`
 	
`to sylvia1`
`frame`
`setc "red`
`pu`
`rt 90`
`fd 30`
`left 90`
`pd`
`repeat 4 [fd 40 rt 90]`
`left 90`
`pu`
`fd 30 rt 90`
`end`
 	
`to sylvia2`
`sylvia1`
`pu setc "blue`
`rt 90 fd 100`
`rt 180`
`sylvia1`
`pu`
`fd 100 rt 90`
`end`
	`to quilt`
	`repeat 4 [sylvia2 rt 90]`
	`rt 90 pu fd 100 left 90`
	`sylvia3`
	`rt 90 sylvia1 left 90`
	`end`
	
	`to sylvia3`
	`frame`
	`pu`
	`fd 50 rt 90 fd 50`
	`repeat 360 [pu fd 50 pd fd 0 pu back 50 rt 1]`
	`back 50 rt 90 fd 50 right 180`
	`end`

`Quilt` is the superprocedure that assembles the quilt you design.

Challenges

- Use one patch procedure as a subprocedure in others.
- What sorts of optical or geometric illusions can you create by just rotating a patch?
- How many patches can you get on the Lynx screen?
- Try the same project with larger or smaller patches.
- Could you program the computer to create random quilts?

Aesthetic tweak

Replace your existing `frame` procedure with this slightly improved version. What does it do differently?

```
to frame
setcolor "black setpensize 3 pd
repeat 4 [fd 100 rt 90]
setpensize 1
end
```

Turtle Cheat Sheet

Here are some turtle graphics primitives to get you started.

Notes:

- # is the sign for inserting a number as the input to a command
- Be sure to use spaces between words and numbers!
- Refrain from using `setpos`. That command makes it hard to move, reorient, or resize quilts.

<table>
<tr><td>Forward #
FD #
For example, <code>fd 50</code></td><td>Back #
BK #</td><td>Right #
RT #</td><td>LEFT #
LT #</td></tr>
<tr><td>CG
clear graphics
Clears the screen and puts the turtle at the center</td><td>Clean
Clears the screen, but leaved the turtle where it is</td><td>PU
Pen up</td><td>PD
Pen down</td></tr>
<tr><td colspan="2">REPEAT # [list of commands]
For example, <code>repeat 4[fd 62 rt 90]</code></td><td>SETC #
set color
SETC 57
<code>SETC "black</code>
<code>SETC "red</code></td><td></td></tr>
<tr><td colspan="2">SETPOS [# #]
For example:
<code>setpos [10 20]</code>
<code>setpos [-25 10]</code>
<code>setpos [-10 -20]</code>
<code>setpos [20 -25]</code></td><td colspan="2">SHOW POS
Displays the current position of the turtle (in coordinates) in the command center</td></tr>
<tr><td colspan="4">SHOW 3 *4
Shows the product of 3 and 4 in the command center. This is the same as asking the turtle to multiple 3 X 4

<code>Show</code> runs a reporter or operation and displays the result in the command center.</td></tr>
</table>

Final Thought

Collaborative expression composed of personal elements created by communicating mathematical ideas to the computer within an extremely open-ended structure makes this project an important "object-to-think-with" for educators.

Resources

- Lynx web site
- Getting Started with Lynx manual

The next several project ideas were written decades ago in LogoWriter, an ancestor of Lynx. You may use Lynx by pointing your browser to http://lynxcoding.club.

The code should work with little to no modification.

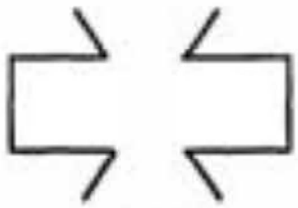
LogoWriter™ Math Reporters

(Primitives)

The following primitive procedures take zero, one, or two values as input and output one number or list of numbers as a result...

No Inputs

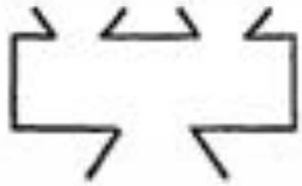
BG
COLOR
COLORUNDER
HEADING
POS
SHAPE
WHO
XCOR
YCOR



One Input

ABS*number*
ARCTAN *number*
COS *number*
COUNT *word,list, or number*
INT *number*
MINUS *number*
RANDOM *number*
ROUND *number*
SIN *number*
SQRT *number*
TOWARDS *[pos]*

OUTPUT *value*
OUTPUT or OP reports its input



Two Inputs

ITEM *item # object*
REMAINDER*divisor dividend*

Infix Reporters with 2 Inputs

\+ - * /

< > = *are predicates which report TRUE or FALSE*
EQUAL? *item 1 item 2*
is the prefix equivalent of =

Useful Mathematical Reporters

SHOW SUM [1 2 3 4]
10

```
to sum :list
if empty? :list [output 0]
output (first :list) +
(sum butfirst :list)
end
```

SHOW AVERAGE [1 2 3]
2

```
to average :list
output sum :list / count
:list
end
```

SHOW FACTORIAL 5
120

```
to factorial :number
if :number = 0 [output 1]
output :number * factorial
:number - 1
end
```

SHOW POWER 2 3
8

```
to power :number :exp
if :exp = 1 [output
:number]
output :number * power
:number (:exp - 1)
end
```

SHOW DIVIDE 10 3
3 remainder 1

```
to divide :divisor
:dividend
output (sentence int
(:divisor / :dividend)
"remainder remainder
:divisor :dividend)
end
```

SHOW DISTANCE [50 50]
70.7

Distance reports the distance between the current turtle's position and a specified set of coordinates

```
to distance :coord
output sqrt (sqr(xcor -
(first :coord)) + (sqr
(ycor - (last :coord)))
end
```

LOGOWriter Dollar Words

By Gary S. Stager
(Idea by Marilyn Burns)

How many dollar words can you think of? What do you mean that you don't know what a dollar word is?

A dollar word is a word in which the sum of it's individual letters equals $1.

Here is how it works... The letter "A" is worth 1¢, "B" = 2¢, "C" = 3¢, "Z" = 26¢, etc...

Type these three short procedures (below) on the flip-side of a LogoWriter page. Be sure to name the page!

In order to find out the value of a word (using the "Dollar Word" rules), type SHOW VALUE "theword (theword should be replaced with the word you wish to evaluate).

```
SHOW VALUE "Gary
$0.51
SHOW VALUE "ELEPHANTS
$1
```

How many dollar words can you think of?

What is the shortest possible word that could be worth $1? What is the longest? Can you think of some quarter words? How about some dime words?

The Procedures:

```
to value :word
output word "$ (getvalue :word) / 100
end

to getvalue :word
if empty? :word [op 0]
output (ascii.value (first :word)) + getvalue bf :word
end

to ascii.value :character
if (ascii :character) > 96 [output (ascii :character) - 96]
if (ascii :character) > 64 [output (ascii :character) - 64]
output ascii :character
end
```

Optional Procedures

Check and see if a word is worth $1.

Type:

```
SHOW DOLLAR? "GARY
False
SHOW DOLLAR? "ELEPHANTS
True

to dollar? :word
output 100 = getvalue :word
end
```

Make a tool to evaluate words worth different amounts of money

```
to worth :word :value
output :value = getvalue
:word
end

to quarter
output 25
end

to nickel
output 5
end

to dime
output 10
end

to dollar
 output 100
end
```

```
SHOW WORTH "GARY QUARTER
FALSE
```


N.A.M.E.
Fallon Education Center
51 CLifford Drive
Wayne, NJ 07470
Voice: (201) 633-3121
Fax #: (201) 628-8837
CIS: 73306,2446
Applelink: K0331 / BITNET:
K0331@Applelink.apple.com

Palindromes

By Gary S. Stager

A palindrome is a word or number in which it's characters or digits are the same backwards and forwards. Bob and 1221 are both examples of palindromes.

In this activity we will focus on numerical palindromes.

Any number can eventually become a palindrome by applying a simple function. If a number is not a palindrome, add the reverse of the number to the number itself. Repeat this process until the sum of the two numbers becomes a palindrome.

157 is not a palindrome, so...

```
  157
+ 751
  908 is not a palindrome, so repeat the process...
+ 809
 1717 is not a palindrome, so repeat the process...
+7171
 8888 is a palindrome!
```

It took four generations to make the number, 157, into a palindrome. All numbers will eventually become a palindrome, but some take longer than others. What kind of numbers are more likely to take several iterations to become palindromes? Do odd numbers take longer? Do prime numbers take longer than composite numbers?

Test your hypotheses and collect data using the following LogoWriter procedures.

<u>The Procedures</u>

The first procedure we need is a simple recursive operation for reporting the reverse of a word (or number)

```
to reverse :word
if empty? :word [output :word]
output word last :word reverse bl :word
end
```

The first set of palindrome procedures work as a command - printing the number of generations it takes for a number to become a palindrome.

```
TO PALINDROME :NUMBER
PRINT (SENTENCE :NUMBER [IS A] FIND.PALINDROME :NUMBER 1 [GENERATION PALINDROME])
END
```

```
to find.palindrome :number :counter
if :number = reverse :number [print :number output :counter]
print :number
output find.palindrome (:number + reverse :number) :counter + 1
end
```

Type `Palindrome 157`

```
    157 is a 4 generation palindrome
```

```
to try.numbers :start :finish
if :start > :finish [stop]
palindrome :start
try.numbers :start + 1 :finish
end
```

Type `Try.numbers 150 160` to printout palindrome information for the numbers 150-160.

Second Palindrome Problem

The second palindrome procedures use the same reverse procedure and function as a reporter. Palindrome now takes a number as input and reports the number of generations it takes before the number becomes a palindrome. Remember that since this new palindrome procedure is a reporter, it must be preceded by a command. Put these procedures on a new page!

```
SHOW PALINDROME 157
3
```

```
TO PALINDROME :NUMBER
OUTPUT FIND.PALINDROME :NUMBER 1
END
```

```
to find.palindrome :number :counter
if :number = reverse :number [output :counter]
output find.palindrome (:number + reverse :number) :counter + 1
end
```

```
to reverse :word
if empty? :word [output :word]
output word last :word reverse bl :word
end
```

An Overnight Problem

The following Record procedure records all of the numbers that take more than two (2) generations to become a palindrome. Two generations was arbitrarily chosen. You may wish to change this number in the Record procedure.

```
to record :start :finish
if :start > :finish [stop]
make "generations palindrome :start
if :generations > 2 [print sentence :start :generations]
record :start + 1 :finish
end
```

Type RECORD 1 100 to record all of the numbers between 1 and 100 which take more than 2 generations to become palindromes.

N.A.M.E.
Fallon Education Center
51 Clifford Drive
Wayne, NJ 07470
(201) 633-3121
Fax: (201) 628-8837
CIS: 73306,2446
Applelink: K0331
Bitnet: K0331@Applelink.apple.com
LogoExpress: GARYS

Thinking Scientifically in Logo
Experimental Math Activities
By: Gary S. Stager

3N Problem

Input a number and if the number is even cut it in half, otherwise multiply the number times 3 and add 1.

Any number inputted will eventually create an infinite pattern of 4 2 1...4 2 1...

```
to 3n :number
pr :number
ifelse even? :number
      [make "number :number / 2]
      [make "number (:number * 3) + 1]
3n :number
end

to even? :number
op member? last :number [0 2 4 6 8]
end
```

Type: `3N` ***some number***
to start the experiment

Controlling the Experiment

EXPERIMENT2 STARTS AT *:number* and PRINTS ALL OF THE NUMBERS THAT TAKE MORE THAN 50 TRIES TO REACH 4 2 1...

```
to experiment2 :number
make "result (3.N :NUMBER [9 9 9] 1)
if :result > 50 [pr (se :number :result)]
experiment2 :number + 1
end
```

Graph the Number of Tries Before 4 2 1 appears

EXPERIMENT STARTS AT *:number* and GRAPHS HOW LONG IT TAKES TO REACH 4 2 1...

Type: `SETUP EXPERIMENT` ***beginning #***

```
TO EXPERIMENT :NUMBER
IF :NUMBER > 40 [STOP]
GRAPH (3N :NUMBER [9 9 9] 1) - 1
EXPERIMENT :NUMBER + 1
END

TO GRAPH :COUNT
TIMES :COUNT
TAB PR :COUNT
PR []
END
```



```
TO 3N :NUMBER :COUNT
IF :NUMBER = 4 [OP :COUNT]
PR :NUMBER
IFELSE EVEN? :NUMBER
      [MAKE "NUMBER :NUMBER / 2 ]
      [MAKE "NUMBER (:NUMBER * 3) + 1]
OP 3N :NUMBER :COUNT + 1
END

TO 3.N :NUMBER :COUNT
IF :NUMBER = 4 [OP :COUNT]
IFELSE EVEN? :NUMBER
      [MAKE "NUMBER :NUMBER / 2 ]
      [MAKE "NUMBER (:NUMBER * 3) + 1]
OP 3.N :NUMBER :COUNT + 1
END

TO EVEN? :NUMBER
OP MEMBER? LAST :NUMBER [0 2 4 6 8]
END

TO SETUP
CG
PU HT
SETPOS [-138 -75]
SETH 0
MAKE "DATA.LIST []
END

TO TIMES :NUMBER
SETY :NUMBER + -75
MAKE "DATA.LIST LPUT :NUMBER :DATA.LIST
PD
FD 0
PU
MOVE
END

TO MOVE
PU
SETY -75
SETX XCOR + 7
END
```

Triangular Fractal

Put three points anywhere on the screen. Randomly choose one of the points and go from where you (the turtle) currently are half the distance to the randomly chosen point. Repeat this process indefinitely.

Type: `SETUP GO`

```
to setup
cg
setc 4
put.dots 0 2
end
to put.dots :start :limit
if :start > :limit [stop]
```

```
pu
setpos list (139 - random 278) (80 - random
160)
dot
make :start pos
put.dots :start + 1 :limit
end

to go
setc 1
find.dot thing random 3
go
end

to find.dot :pos
seth towards :pos
pu
fd (distance :pos) / 2
dot
end

to distance :pos
output sqrt (sq (xcor - first :pos)) + (sq
(ycor - last :pos))
end

to sq :number
op :number * :number
end

to big.one
cg
pu
setc 4
setpos [-120 -60]
dot
make 0 pos
setpos [0 60]
make 1 pos
dot
setpos [120 -60]
make 2 pos
dot
go
end

to dot
pd
fd 0
pu
end

to go2 :limit
find.dot thing random :limit - 1
go2 :limit
end

to setup2 :limit
cg
setc 4
put.dots 0 :limit - 1
end
```

The Ice Cream Scoop Problem

*The following experiment was inspired by a visit to a fourth grade classroom. There was a floor-to-ceiling-high chart containing pictures of ice cream cones. When I inquired about the chart I was told that the students' problem solving book posed the following problem, " **If you had 17 scoops of ice cream and an unlimited number of single, double & triple dip cones, can you make a chart of the 33 possible combinations of cones based on 17 scoops?**" I found the problem intriguing although the activity posed to the students could have been done with brute force by a gorilla. My question was, "Why does 17 scoops generate 33 combinations?"*

If you have X scoops of ice cream and an unlimited supply of single, double, and triple dip cones, how many possible combinations of servings can you make?

Type: `ICE.CREAM` *(starting # of scoops) (limit # of scoops) (number of kinds of cones)*

This first experiment prints out all of the possible combinations for a given number of scoops.

```
TO ICE.CREAM :START :LIMIT :SCOOPS
PR [T\  D\  S]
SCOOPLIST :START :LIMIT :SCOOPS
END

to scooplist :total :limit :SCOOPS
if :total > :limit [stop]
make :total tryeach 0 :SCOOPS :total
PRINTLIST THING :TOTAL
pr (se [There are] count thing :total
[number of combinations of] :total "scoops)
make "data.list lput list :total count
thing :total :data.list
scooplist :total + 1 :limit :SCOOPS
end

to tryeach :howmany :scoops :total
if :scoops = 1 [ op (list (list :total))]
if (:howmany * :scoops) > :total [op []]
op se fputall :howmany tryeach 0 :scoops -
1 :total - :howmany * :scoops tryeach
:howmany + 1 :scoops :total
end

to fputall :first :list
if empty? :list [op []]
op fput fput :first first :list fputall
:first bf :list
end
```

```
to printlist :list
if empty? :list [stop]
pr last :list
printlist bl :list
end
```

```
to startup
make "data.list []
end
```

<u>A Sample of the results generated by typing ...</u>

```
ICE.CREAM 1 5 3

T D S
0 0 1
There are 1 number of combinations of 1 scoops
0 1 0
0 0 2
There are 2 number of combinations of 2 scoops
1 0 0
0 1 1
0 0 3
There are 3 number of combinations of 3 scoops
1 0 1
0 2 0
0 1 2
0 0 4
There are 4 number of combinations of 4 scoops
1 1 0
1 0 2
0 2 1
0 1 3
0 0 5
There are 5 number of combinations of 5 scoops
```

The following experiment prints just the number of scoops and the number of possible combinations so that we can analyze that data without the clutter of the actual combinations

Type: **SAVEMEMORY** *(starting # of scoops) limit*

```
to savememory :total :limit
if :total > :limit [stop]
if member? last :total [0] [savepage bottom]
clearname :total - 1
recycle
startup
scooplist :total :total
insert se char 32 :data.list
savememory :total + 1 :limit
end
```

```
to scooplist :total :limit
if :total > :limit [stop]
make :total tryeach 0 3 :total
make "data.list lput list :total count
thing :total :data.list
scooplist :total + 1 :limit
end
```



```
to tryeach :howmany :scoops :total
if :scoops = 1 [ op (list (list :total))]
if (:howmany * :scoops) > :total [op []]
op se fputall :howmany tryeach 0 :scoops -
1 :total - :howmany * :scoops tryeach
:howmany + 1 :scoops :total
end
```

```
to fputall :first :list
if empty? :list [op []]
op fput fput :first first :list fputall
:first bf :list
end
```

```
to printlist :list
if empty? :list [stop]
pr last :list
printlist bl :list
end
```

```
to startup
make "data.list []
end
```

<u>A Sample of the results...</u>

[1 1] [2 2] [3 3] [4 4] [5 5] [6 7] [7 8]
[8 10] [9 12] [10 14] [11 16] [12 19] [13 21]
[14 24] [15 27] [16 30] [17 33] [18 37] [19 40]
[20 44] [21 48] [22 52] [23 56] [24 61] [25 65]
[26 70] [27 75] [28 80] [29 85] [30 91] [31 96]
[32 102] [33 108] [34 114] [35 120] [36 127] [37 133]
[38 140] [39 147] [40 154] [41 161] [42 169] [43 176]
[44 184] [45 192] [46 200] [47 208] [48 217] [49 225]
[50 234] [51 243] [52 252] [53 261] [54 271] [55 280]
[56 290] [57 300] [58 310] [59 320] [60 331] [61 341]
[62 352] [63 363] [64 374] [65 385] [66 397] [67 408]
[68 420]

Ice Cream Scoop Code by Brian Silverman & Gary Stager

Gary S. Stager
N.A.M.E.
Fallon Education Center
51 Clifford Drive
Wayne, NJ 07470
(201) 633-3121
Compuserve: 73306,2446
NJ ETN: namegs Applelink: K0331

The World's Greatest LogoWriter Function Microworld

©1987 Gary S. Stager
Rough Draft

At the 1987 East Coast Logo Conference, E. Paul Goldenberg presented some ideas for learning the concept of functions or mathematical operations in Logo. I was as always inspired by his presentation and decided to spend the next few days (sleeplessly) extending and embellishing the ideas put forth by Paul.

Paul Goldenberg suggested that students could concretize and understand the concept of mathematical functions by being presented with sufficient tools that would allow them to explore the same problem in a number of domains. His Logo examples demonstrated how a function could be manipulated through the use of "mathematical sentences" and graphs.

My intent was to create a Logo microworld in which the notion of mathematical functions could be explored by students of all ages , not just in two domains (sentences and graphs), but in at least four domains. This allows students, regardless of divergent learning styles to find a comfortable medium for conceptualizing these concepts. The four sets of tools present in this LogoWriter Function Microworld are, Mathematical Sentences, Graph Tools, X/Y Table Tools, and Function Machines.

In the spirit of a Logo microworld, all of these tools are extensible, self-correcting, inheritantly interesting (I hope!), non-threatening, and contain powerful ideas. The student(s) has complete control over the environment and enough memory to build his/her functions -- The entire microworld and student procedures fits comfortably in 4K of memory!

The following is a short narrative on how a problem may be explored and potentially solved by a child or group of children using this microworld. All four aspects of the software will be illustrated, but it is by no means necessary to work in all four domains every time you wish to explore with the microworld. The order for using the particular tools is also inconsequnetial. I have also included a lesson plan for a game which I spontaneously created while working with a group of elementary students. I call the game "Battle of the Functions" and the kids love playing it. "Battle of the Functions" turns out to be a very nice supplementary activity for using this LogoWriter Function Microworld.

Getting Started:

1) Load LogoWriter into your Apple. (Sorry I haven't typed the procedures in other versions yet.)
2) Insert your Project Scrapbook Disk Volume into the disk drive and press ESCAPE.
3) Select the FUNCT.WORLD page by using the arrow keys to pace the cursor on this page and press RETURN.
4) Wait a few seconds while the tool procedures "sneak" into memory.
5) When the cursor is blinking in the command center it is probably a very good idea to rename your page so that the original procedures on the flip-side are not destroyed!

Type
NAMEPAGE "pagename

6) Either the student(s) or teacher should then flip the page to the flip-side and delete any unwantedfunctions (mathematical operations) and create their own, depending on their age, ability, and what they are studying.

Creating a Logo Function:

The power of this microworld lies in its flexibility. First graders (or their teachers) or precalculus students in high school can create appropriate mathematical functions by using the same simple structure. Remember, all LogoWriter procedures are written on the Flip-side of the page. Hit Apple-F to flip the LogoWriter page to the flip-side.

All Functions in this Microworld Have One Numerical Input and One Numerical Output!

This includes the ability to use mathematical primitives or tools already in LogoWriter. For example, +, -, /, *, SQRT, DISTANCE, TOWARDS, ABS, INT, SIN, COS, ROUND, etc...

Elementary Functions

```
TO 2PLUS :NUMBER
OP :NUMBER + 2
END

TO ADD5 :NUMBER
OP :NUMBER + 5
END

TO DOUBLE :NUMBER
OP :NUMBER * 2
END

TO SPLIT :NUMBER
OP :NUMBER / 2
END
```

Note: The CIRCUMFERENCE function uses two other *functions, PI and DIAMETER in calculating a value.*

I. Mathematical Sentences:

Once you have created some mathematical function procedures, you can solve word problems by stacking up these functions and providing a numerical input. This can be done either in the Command Center or in a LogoWriter procedure. The mathematical functions are calculated from right to left. The metaphor is that a number is being dropped into a function machine and the result is dropped into the preceeding function machine until a final value is outputted.

For Example:

```
SHOW DOUBLE DOUBLE DOUBLE ADDS 5
80
```

is the answer outputted

This is the same as saying
$(5 + 5) * 2 * 2 * 2$

```
SHOW TRIPLE ADDS MINUS7 SPLIT TIMES5 SQUARE 2
24
```

is the answer outputted

This is the same as saying
$(((((2 * 2) * 5) / 2) - 7) + 5) * 3$

Note: This is a good time to play the "Battle of the Functions" game.

Problems may be posed by the student or the teacher and can their results can be explored in the mathematical sentence, graph, table, or machine domains. For the sake of discussion, we will try to answer a problem that may be puzzling to some young students (or adults).

Is the output of...

```
DOUBLE ADDS 10
```

equal to...

```
ADDS DOUBLE 10
```

???

You may explore this problem in the way described above or in any of the following ways:

Graph Tools:

If you have already figured out that the two equations are not equal, the following question may be asked:

Is there ANY nunber which can be inputted into both equations and give the same output?

One way to find a solution to this problem is to graph the equations. This set of tools plugs in numbers from -80 to 80 and plots the points which are on the screen. The first problem would be addressed as:

$Y = (X + 5) * 2$

X is the number plugged in by LogoWriter and the point plotted is the coordinate pair of [X Y].

II. Using the Graph Tools:

1) Type
`CG CT`
2)Type
`GRID`

this draws the X Y axis

3) You may then select two scales for the graph; `WIDEANGLE` or `CLOSEUP`
`WIDEANGLE` makes each notch on the axis equal 10 and `CLOSEUP` makes each notch on the axis equal 1.
`WIDEANGLE` is the default scale.

4) Type
`GRAPH [DOUBLE ADDS :X]`

or...

Type

`GRAPH [3 * :X / 2]`
ETC...

5) :X must be included in the brackets of any function you wish to graph. Any function you or others have created or is a LogoWriter primitive may be included in an equation inside GRAPH's list, as long as :X is used.

6) If you wish to see what the equation `ADDS DOUBLE :X` might look like on the same graph, it is probably a good idea to change the turtle's color by typing, `SETC (0-5)` and then use `GRAPH`.

There is a number which makes both equations equal if and only if the two lines intersect on the graph. The point at which they intersect is the number which makes both equations equal. This is a concept that always elluded me through numerous math courses and I suspect that others will experience similar mathematical revelations by using this microworld.

7) You may change the graph's scale at any time by using `WIDEANGLE` or `CLOSEUP`. Sometimes you may need to put a scale factor in your equation so that the result is graphable.

8) Your graph may be printed at any time by typing `PRINTSCREEN`.

9) Clear the screen and repeat the procedure for different functions as often as you wish.

III. Table Tools:

The table tools afford the user the opportunity to create an X|Y table of results from an inputted equation. The X|Y table provides another medium for comparing the results of several functions. I will continue using the previous example in demonstrating how the table tools are used.

TABLE requires 4 inputs; the equation, starting :X value, an ending :X value, and the increment by which you wish :X to change value.

1) Type
CG CT
2) Type
TABLE [DOUBLE ADD5 :X] -5 5 1

This means: run the equation DOUBLE ADD5 :X, the first number plugged in for :X will be -5, no :X value will be higher than 5, and plug in each integer between -5 and 5 if we increase the :X value by 1 each time.

3) Observe the results of the table. If there are a lot of results, hit APPLE - U and use the down arrows to scroll through the data. Then hit APPLE - D.

4) Record the results with either pencil and paper, PRINTSCREEN, or PRINTTEXT80.
PRINTTEXT80 is recommended if there was a wide range of numbers used.

IV. Function Chunks

Another way of exploring the effect of functions on a number is to create a proportional graphic representation of the function using LogoWriter's turtle graphics capabilities.

The simple tool procedure, BAR, requires a numerical input and draws a rectangle the height of the input.

For Example:

BAR 50
draws a rectangle 50 steps high
MOVE.L
BAR DOUBLE 50
draws a rectangle 100 steps high
MOVE.R MOVE.R
BAR 3FOURTHS DOUBLE 50
draws a rectangle 75 steps high

MOVE.L or MOVE.R moves the turtle to the left or the right so that the next bar can be drawn.

Obviously, function chunks are an excellent medium for understanding fraction arithmetic, ratio, and proportion. As in the other four parts of the microworld, any one input mathematical operation may be used with the BAR, MOVE.R., and MOVE.L procedures.

V. Function Machines:

Probably the most exciting and educational aspect of the LogoWriterFunction Microworld is the ability to represent functions and equations in function machines. Function machines are a graphic way of solving a mathematical problem. In this microworld you actually see a numerical input go in the "hopper" (top) of a machine and come out the "spout" (bottom) so that the result of one function can be passed to the next function (machine). The last number displayed is the result of all of the function machines working together (the equation). This microworld has the ability to use up to 12 functions at once. Due to screen limitations, the functions can not be displayed in a vertical line, but rather 4 columns of 3 machines. At the end of a column the result (thus far) is passed to the top machine of the next column. The procedure for using the function machines is as follows:

1) Type
SETUP
this clears the graphic screen and positions the turtle in the proper place for drawing the function machines.

2) Type
DRAW [ADD5 DOUBLE] # *any number or any other combination of functions* (up to 12) in the brackets and give a numerical input.

The function machines will then be drawn and a numerical answer will "drop out" the bottom.

3) Type PRINTSCREEN if you wish a hard copy of your function machines.

4) Repeat Steps 1-3 as many times as you wish to solve function problems.

What would happen if you doubled 1 twelve times??? Will the result be a small number or a large number???

Type
SETUP
DRAW [DOUBLE DOUBLE DOUBLE DOUBLE DOUBLE DOUBLE DOUBLE DOUBLE DOUBLE DOUBLE DOUBLE DOUBLE] 1

Note for programmers: There are no global variables used in the LogoWriter code for the function machine part of the microworld. All values are passed from one procedure to another.

Students may create any function they want. For example:

```
TO 2PLUS :NUMBER
OUTPUT 2 + :NUMBER
END

TO ADD15 :NUMBER
OUTPUT :NUMBER + 15
END

TO DOUBLE :NUMBER
OUTPUT :NUMBER * 2
END
```

```
TO ADD5 :NUMBER
OUTPUT :NUMBER + 5
END

TO TRIPLE :NUMBER
OUTPUT :NUMBER * 3
END

TO SPLIT :NUMBER
OUTPUT :NUMBER / 2
END

TO SQUARE :NUMBER
OUTPUT :NUMBER * :NUMBER
END

TO CUBE :NUMBER
OUTPUT :NUMBER * :NUMBER *
:NUMBER
END

TO DIVIDEBY50 :NUMBER
OUTPUT :NUMBER / 50
END

TO 2THIRDS :NUMBER
OUTPUT :NUMBER * 2/3
END

TO 3FOURTHS :NUMBER
OUTPUT :NUMBER * 3/4
END

TO BAR :HEIGHT
REPEAT 2[FORWARD :HEIGHT
RT 90 FD 30]
END

TO MOVE.L
PU
LT 90
FD 40
RT 90
PD
END

TO MOVE.R
PU
RT 90
FD 40
LT 90
PD
END

TO STARTUP
CLEAREVENTS
CLEARTOOLS
CLEARNAMES
GETTOOLS "FUNCT.TOOLS
NORMAL.SCALE
RECYCLE
END
```

FUNCTION TOOLS

THE PAGE NAME MUST BE CALLED "FUNCT.TOOLS

```
FUNCTION GRAPH AND X/Y
TABLE TOOLS
(C) GARY S. STAGER 1987

TO GRID
PU HOME
PD
REPEAT 30[X.AXIS]
PU HOME
PD
REPEAT 20[Y.AXIS]
PU HOME
END

TO X.AXIS
SETH 90
NOTCH
FORWARD 10
END

TO Y.AXIS
PD
SETH 0
NOTCH
FORWARD 10
END

TO STARTUP
MAKE "SCALE 10
END

TO NOTCH
RT 90
FORWARD 3
BACK 6
FORWARD 3
LT 90
END

TO GRAPH :FUNCTION
CT
IF NOT NAME? "SCALE
[NORMAL.SCALE]
MAKE "X -80 / :SCALE
PU
SETP :X
RUN :FUNCTION
PRINT (SENTENCE :FUNCTION
[FROM] :X CHAR 32 [TO] (0
- :X))
PD
GRAPH1 :FUNCTION :X (0 -
:X) 1
END

TO GRAPH1 :FUNCTION :X
:XMAX :INC
IF :X > :XMAX [STOP]
SETP :X
RUN :FUNCTION
GRAPH1 :FUNCTION :X + :INC
:XMAX :INC
END

TO SETP :X :Y
IF OR (:SCALE * :Y) > 80
(:SCALE * :Y) < -80
[STOP]
IF OR (:SCALE * :X) > 140
(:SCALE * :X) < -140
[STOP]
PU
SETPOS LIST (:SCALE * :X)
(:SCALE * :Y)
PD
SETPOS LIST (:SCALE * :X)
(:SCALE * :Y)
END

TO ZOOM
MAKE "SCALE 10
GRID
END

TO NORMAL.SCALE
MAKE "SCALE 1
GRID
END
```

TABLE TOOLS

```
TO TABLE :FUNCTION :START
:STOP :INC
CT CG HT
CHART
IF NOT NAME? "SCALE
[NORMAL.SCALE]
MAKE "X :START
PRINT (SENTENCE :FUNCTION
[FROM] :X [TO] :STOP)
REPEAT 4[PRINT []]
TABLE1 :FUNCTION :X :STOP
:INC
END
```

```
TO TABLE1 :FUNCTION :X
:XMAX :INC
IF :X > :XMAX [STOP]
PRINTPOINT :X RUN
:FUNCTION
TABLE1 :FUNCTION :X + :INC
:XMAX :INC
END

TO PRINTPOINT :X :Y
INSERT :X
TAB TAB
CB
PR :Y
END

TO CHART
PU
SETPOS [-140 45]
PD
SETH 90
FORWARD 85
BACK 40
LT 90
FORWARD 5
BACK 135
PU
SETPOS [-115 55]
LABEL ":X
PU
SETPOS [-90 55]
LABEL ":Y
END
```

FUNCTION MACHINE TOOLS

```
TO SETUP
CG PU
SETPOS [-135 50]
END

TO MACHINE :NUM
IF MEMBER? :NUM [4 7 10]
   [PU SETPOS LIST (XCOOR
POS) + 70 50]
PD
FORWARD 25
RT 90
FORWARD 22
LT 135
FORWARD 15
BACK 15
RT 135
PU
FORWARD 20
LT 45
PD
FORWARD 15
BACK 15
RT 45
FORWARD 22
RT 90
FORWARD 25
RT 90
FORWARD 22
LT 135
PD
FORWARD 15
BACK 15
RT 135
PU
FORWARD 20
LT 45
PD
FORWARD 15
BACK 15
RT 45
FORWARD 22
RT 90
END

TO NAME.MACHINE :NAME
:FUNCTION :INPUT
MAKE "OLD.POS POS
PU
SETPOS LIST ((XCOOR
:OLD.POS) + 10) ((YCOOR
:OLD.POS) + 15)
LABEL :NAME
INPUT LIST ((XCOOR
:OLD.POS) + 15) ((YCOOR
:OLD.POS) + 25) :INPUT
RESULT LIST (XCOOR
:OLD.POS)  ((YCOOR
:OLD.POS) - 50) (RUN
SENTENCE :FUNCTION
:INPUT)
SETPOS :OLD.POS
PU
BACK 60
PD
OUTPUT (RUN SENTENCE
:FUNCTION :INPUT)
END

TO XCOOR :POS
OUTPUT FIRST POS
END

TO YCOOR :POS
OUTPUT LAST POS
END

TO INPUT :POS :NUMBER
PU SETPOS :POS
LABEL :NUMBER
END

TO RESULT  :POS :NUMBER
PU SETPOS :POS
LABEL :NUMBER
END

TO DRAW :LIST :INPUT
IGNORE DOIT :LIST :INPUT
1
END

TO DOIT :LIST :INPUT :NUM
IF EMPTY? :LIST [IGNORE
:INPUT OUTPUT []]
MACHINE :NUM
OUTPUT DOIT BUTLAST :LIST
(NAME.MACHINE (LAST :LIST)
(LAST :LIST) :INPUT) :NUM
+ 1
END

TO IGNORE :THING
END
```

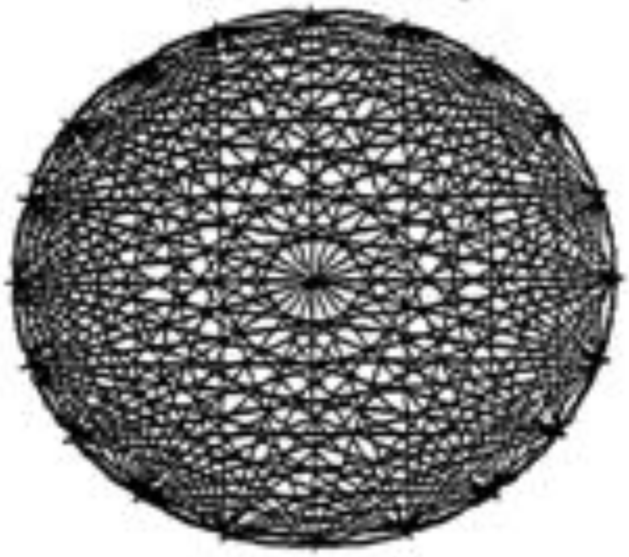
Logo Modulo Designs - © 1986 Gary S. Stager

Recently, I was sitting in a rather mundane college math course and we were studying the topic of mathematical systems and modulo arithmetic. While thumbing through the math text, I came across one of those familiar "challenge problems" found in the bottom corner of a page. The challenge suggested drawing a modulo design for a particular mod. Being a former mathphobic and born-again math student I became intrigued by the idea of teaching Logo to draw a modulo design based on any mod that is inputted.

Below are the procedures I created for drawing modulo designs. The first set of procedures draws a design based on the products of every pair of numbers in the mod. The second set of procedures draws a design based on the product of one factor and the other numbers in the mod.

Modular Arithmetic - an arithmetic constructed to use a *finite* rather than infinite set of numbers. Modular arithmetic is also sometimes called *clock arithmetic*, because the clock face provides a perfect model. *For example:* In MOD 12, (7 + 8) = (Remainder of 15 and 12) = 3

The Program



```
TO START :CIRC :MOD
CIRCLE :CIRC :MOD
DO 0 :MOD
END

TO CIRCLE :CIRC :MOD
PD
CIRCLER :CIRC :MOD :MOD - 1
END

TO CIRCLER :CIRC :MOD :COUNT
IF :COUNT < 0 [STOP]
REPEAT 360 / :MOD [FD :CIRC / 360 RT 1]
MAKE :COUNT POS
NOTCH
CIRCLER :CIRC :MOD :COUNT - 1
END

TO NOTCH
RT 90 FD 3 BK 6 FD 3 LT 90
END

TO DO :NUMBER1 :MOD
IF :NUMBER1 > :MOD - 1 [STOP]
INC :NUMBER1 0 :MOD
DO :NUMBER1 + 1 :MOD
END

TO INC :NUMBER1 :NUMBER2 :MOD
IF :NUMBER2 > :MOD - 1 [PU STOP]
PLOT :NUMBER1 :NUMBER2 :MOD
INC :NUMBER1 :NUMBER2 + 1 :MOD
END

TO PLOT :NUMBER1 :NUMBER2 :MOD
PU
SETPOS THING MOD :NUMBER1 :MOD
SETH TOWARDS THING ( MOD :NUMBER1 *
:NUMBER2 :MOD ) PD
FD DISTANCE THING ( MOD :NUMBER1 *
:NUMBER2 :MOD )
END

TO MOD :NUMBER :MOD
OP REMAINDER :NUMBER :MOD
END
```

For Mac Logo (LCSI) *Use this procedure in place of the other PLOT procedure.*

```
TO PLOT :NUMBER1 :NUMBER2 :MOD
PD
LINE THING MOD :NUMBER1 :MOD THING (
MOD :NUMBER1 * :NUMBER2 :MOD )
END
```

This procedure uses the Mac's Quickdraw Graphic routines and speeds up the drawing of the design.
[For versions of Logo other than Mac Logo (LCSI) , replace CG with CS]

How the Program Works

START is the toplevel procedure for this program and takes two inputs; the circumference of the circle and the mod you wish to draw.

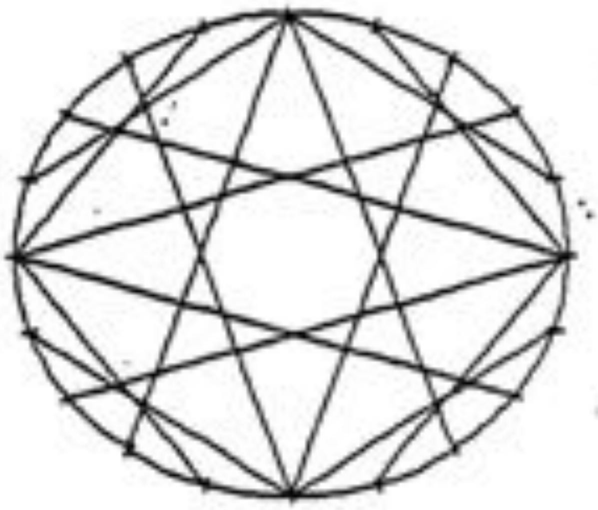
A circle is then drawn and divided by the mod specified in the START procedure. As each segment of the circle is drawn, the position of the NOTCH is made into a global variable.

DO and INC are recursive procedures that determine the product of all the possible factors in a particular MOD. The first factor and the product of the first factor and every other factor in the mod is then passed to the PLOT procedure.

The PLOT procedure is the heart and soul of the modulo design program. PLOT sets the turtle's position to the thing of the first factor (the position is stored under the name of each number in the mod). The turtle's heading is then set TOWARDS the product of the first factor and another number in the mod. The turtle then goes forward the distance between the position on the circle of the factor and the position on the circle of the product of the two factors. *[Note: You may need to write TOWARDS and DISTANCE if your version of Logo doesn't contain these primitives]*

Modulo Designs with Only One Factor

```
TO START :CIRC :MOD :FACTOR
CIRCLE :CIRC :MOD
DO :FACTOR :MOD :MOD
END

TO CIRCLE :CIRC :MOD
PD
CIRCLER :CIRCUMFERENCE :MOD :MOD - 1
END

TO CIRCLER :CIRC :MOD :COUNT
IF :COUNT < 0 [STOP]
REPEAT 360 / :MOD [ FD :CIRCUMFERENCE/ 360
RT 1 ]
MAKE :COUNT POS
NOTCH
CIRCLER :CIRCUMFERENCE :MOD :COUNT - 1
END

TO DO :FACTOR :MOD :BASE
IF :MOD < 1 [STOP]
PLOT :MOD :FACTOR :BASE
DO :FACTOR :MOD - 1 :BASE
END

TO MOD :NUMBER :MOD
OP REMAINDER :NUMBER :MOD
END

TO NOTCH
RT 90 FD 3 BK 6 FD 3 LT 90
END

TO PLOT :NUMBER1 :NUMBER2 :MOD
PD
LINE THING MOD :NUMBER1 :MOD THING ( MOD
:NUMBER1 * :NUMBER2 :MOD )
END
```



How It Works

This program differs from the first modulo design program by drawing only the product of one particular factor and each number in the specified mod. The first set of modulo design procedures draws the products of every possible pair of factors in a particular mod.

The modulo design pictured above is in *MOD 20* with a factor (or multiplier) of *5*. This means that the turtle connects each number in mod 20 (1...20) with the product of that number and 5. This mod would be notated (20 5). To recreate this design type:

START 700 (or any circumference) 20 5

Gary S. Stager
Director of Training
Network for Action in Microcomputer Education
12 Locust Place
Wayne, NJ 07470
(201) 831-0133

A HISTOGRAM OF A ROLLING DIE (DICE)

1 Die

```
TO SETUP
PU SETPOS [-130 -75]
SETH 0
END
```

```
TO ROLL.6 :NUMBER
SETUP
IF 1 > :NUMBER [STOP]
RT 90
PU
FD 30 * (1 + RANDOM 6)
LT 90
BAR
ROLL.6 :NUMBER - 1
END
```

```
TO BAR
IF COLORUNDER = 0 [PD FD 0 STOP]
PU FD 2
BAR
END
```

To graph the rolling of a die, type: **ROLL.6** ***number of rolls***
The result for each number should be about even in height.

To Graph 2 Dice

Use the following procedure with the other procedures listed above:

```
TO ROLL.12 :NUMBER
SETUP
IF 1 > :NUMBER [STOP]
RT 90
PU
FD 15 * (1 + RANDOM 6) + (1 + RANDOM 6)
LT 90
BAR
ROLL.12 :NUMBER - 1
END
```

Type **ROLL.12** ***number of rolls*** to roll 2 dice. This should result in a bell-shaped curve.

GARY S. STAGE
President of ISTE's SIGLo

Educational Computing Consultant
Teacher Trainer

1415 RATZER R
WAYNE, N.J. 074
(201) 942-35

Sites to Explore

Dr. Constance Kamii's Web Site

https://sites.google.com/site/constancekamii/

Conrad Wolfram's TED Talk

https://www.youtube.com/watch?v=60OVlfAUPJg

Stephen Wolfram's Intro to Wolfram Language

https://www.youtube.com/watch?v=_P9HqHVPeik

Making Programming Accessible to Everyone with Wolfram Language

https://www.youtube.com/watch?v=ALuQzgDvr2g

Creating a Video Game in Lynx

http://stager.tv/blog/?p=2436 for video tutorials

Note: The following project starters are written for Lynx and its predecessor LogoWriter. It should be possible to translate them into Scratch (scratch.mit.edu) or SNAP! (snap.berkeley.edu)

Recognize the Superiority of Games Over Worksheets
Kamii, C. (2000). *Young Children Invent Arithmetic* (2nd ed.): New York: Teachers College Press.

It is necessary for children to repeat adding the same numbers if they are to remember sums and build a network of numerical relationships (refer to Figure 5.2), Repetition in games is much better than with worksheets for many reasons. The fact that children are intrinsically motivated in games was discussed earlier in this chapter. Seven other reasons are given below.

First, feedback is immediate in games because children supervise each ' other. By contrast, worksheets are usually returned the next day, and children cannot remember and do not care about what they did yesterday.

Second, when worksheets are used; truth is decided by the teacher, and children get the message that truth can come only from the teacher. In a game, by contrast, the players decide whether an answer is correct. If one child says that 2 + 2 is more than 2 + 3, for example, children try to convince each other and arrive at truth by themselves. In logico-mathematical knowledge, children are bound to arrive at truth if they argue long enough because there is absolutely nothing arbitrary in logico-mathematical knowledge.

Third, games can be played at many levels in a variety of ways, but worksheets encourage children to crank answers out mechanically. In playing Put and Take (see Chapter 11), for example, some children can make 6 only with 6 chips that are each worth one point. Others say that they can make 6 either with 3 2-point chips or with 1 5-point chip and a 1-point chip.

Fourth, having to write answers interferes with the possibility of remembering sums. Children are much more likely to remember sums when they are free to think "2, 3, and 5," for example, without stopping to write "5." Some first graders have to think to make a "5" look different from an "S.'"

Fifth, children are more likely in a game to construct a network of numerical relationships (refer to Figure 5.2). If a player rolls a 3 and a 3, and the next roll is a 3 and a 4, for example, there is a high probability that the answer will be deduced from 3 + 3 = 6. When children fill out worksheets, by contrast, they approach each problem mechanically as a separate and independent problem.

Sixth, children choose the specific games they want to play, but they can seldom choose the worksheets they get. If children can choose an activity that appeals to them, they are likely to work harder. In life outside school, adults constantly make choices, and children need to learn to make wise choices within limits.

Our seventh and last point is that children do not develop sociomorally by sitting alone filling out worksheets. They are well behaved when they are filling out worksheets, but working alone precludes the possibility of sociomoral development. In games, by contrast, children have to interact with others, make decisions together, and learn to resolve conflicts. As stated in Chapter 4, sociomoral education takes place every minute of the school, whether or educators are aware of it. By giving countless worksheets, we unwittingly reinforce children's heteronomy. Thereby, preventing the development of their autonomy.

MATH GAMES WITH PLAYING CARDS
FOR CHILDREN IN GRADES K-3

Constance Kamii
University of Alabama at Birmingham
August, 2015

Kindergarten

1. Lining Up the 5s

Number of players: 2 or 3, preferably 3*

Three suits of cards A-10 are used. All 30 cards are dealt to the 3 players. Each player aligns the 10 cards received, face up, in front of himself.** The players who have 5s put them down in a column in the middle of the table.

The children decide who will go first. (The turns then go clock-wise.)

The players take turns putting one card down at a time. They make a matrix by extending each suit to the right or left, without skipping any number (for example, the 6 of spades followed by the 7 of spades, ... or the 4 of spades followed by the 3 of spades.)

Anyone who does not have a card that can be played must pass. Each time a player passes, he takes a counter. Players can pass only 3 times***. When a player with 3 counters must pass a 4^{th} time, that player is out of the game. He puts down in the matrix all the cards remaining in his hand. In this situation it is often necessary to skip one or more numbers, leaving blank spaces in the matrix between cards that are not consecutive.

The first player to use up all his cards wins.

By playing this game, most kindergartners learn to read numerals without a single lesson on how to read numerals.

---

*If there are more than 3 players in a game, children have to wait longer to get a turn. Having to wait is a waste of time that could be spent thinking.

** "He" and "she" are used alternately throughout this paper. "He" is used in the first, third, and fifth games, etc., and "she" is used in the second, fourth, and sixth games.

***We introduced this rule because (a) many children were passing without systematically examining all their cards for possible use, and (b) the more advanced players passed just to prevent others from using their cards.

2. Before or After
Number of players: 2 or 3

All the numeral cards from one deck (A-10) are used. The cards are dealt to all the players, but the last card is turned up in the middle of the table. The players keep their cards in face-down stacks. The first player turns over the top card of her stack and tries to make a pair with the number that comes immediately before or after the number that is up. (For example, if a 5 is up, a pair can be made with either a 4 or a 6.) If a pair can be made, the player can take both cards and keep them. If not, the card turned over stays in the middle of the table and gets covered up by the next player (or the player after the next player, or the player after her, etc.).

Play continues until pairs cannot be made any more. The winner is the person who collects more cards than anybody else.

3. War (for 2 players)
Number of players: 2

All the number cards from one deck (A-10) are dealt to the 2 players. Without looking at them, each player puts his pile in front of himself, face down. The two players then simultaneously turn over the top cards of their respective piles. The person who turned over the larger number takes both cards. The winner is the person who collected more cards than the other.

If there is a tie, each player turns over the next card, and the person who turned up the larger number takes all 4 of them. (This is a modification of the conventional rule.)

Modification into a fast addition game. The person who announced the correct sum first wins both cards.

4. Find Five (also known as Piggy Bank)
Number of players: 2 or 3

Eight cards each of numbers 1 through 4 (from 2 decks) are used. The object of the game is to make 5 with 2 cards (4+1 or 2+3).

All the cards are dealt. Without looking at them, each player makes a face-down stack with the cards received. On her turn, each player turns over the top card of her stack. The first player always has to discard the card turned over in the middle of the table. If the first player discards a 3, and the second player turns over a 4, she, too, has to discard this card. If, on the other hand, the second player turns over a 2, this 2 can be taken with the 3 on the table. The person who collects more cards than anybody else is the winner.

1st Grade

5. Double War

   Number of players: 2

   This game is played like War except that the cards are dealt so that each player will have 2 stacks. Each player turns over the top cards of both stacks, and the person who announces the larger total first takes all 4 cards.

6. Tens with Nine Cards****

   Number of players: 2 or 3

   Thirty-six cards, 4 each of A (1) through 9, are used. Nine cards are randomly arranged as shown in the figure. The first player takes pairs of cards that make 10 (such as 6+4, 5+5, and 7+3). She then fills the empty spaces with cards from the deck. The second player continues the game in the same way.

6	2	3
5	1	4
5	7	2

   The person who collects the most cards is the winner

7. Find Ten**** (or Find Seven, Eight, Nine, Eleven, etc.)

   Number of players: 2 or 3

   This game is played like Find Fives, but cards 1 through 9 are used (a total of 36 cards), and the object of the game is to find 2 cards that make 10 (9+1, 8+2, etc.).

   In Find Seven, cards 1 through 6 are used. In Find Eight, cards 1 through 7 are used, etc.

8. Draw Ten****

   Number of players: 3

   This game is played like Old Maid, but cards 1-9 are used, and the object of the game is to find 2 cards that make 10. One card is removed from the deck at random, so that there will be a card without a mate at the end of the game. All the other cards are dealt.

   Each player goes through the cards received and puts in front of herself all the pairs that make 10 (6+4, for example).

   The players then hold their cards like a fan and take turns letting the person to the left draw one of them at random. If the person who drew a card can use it to make 10 with one of her cards, the pair is added to her collection of 10s. If a pair cannot be made, the card drawn is kept, and the next person draws a card.

****Becoming able to make 10 with 2 cards facilitates children's changing 8 + 4 to (8 + 2) + 2, for example, and 7 + 5 to (7 + 3) +2. Having to make 10 with 2 cards thus helps children construct tens. It is therefore important not to let children make 10 with 3 cards.

Play continues until one person is left holding the odd card and loses the game.

9. Shut the Box
   Number of players: 2 or 3

   Two dice and 11 cards numbered 1 through J are used. The 11 cards are arranged in a line in sequence from 1 to 11 (J), face up. The players take turns rolling the dice and turning down as many cards as they wish to make the same total. For example, if a 6 and a 2 were rolled, a player can turn down the 8; the 1 and the 7; the 2 and the 6; the 3 and the 5; or the 1, the 3, and the 4. The player keeps playing until it is impossible to make a total with the remaining numbers. The numbers left unused are added and recorded, and the next player takes a turn.

   The points left at the end of each turn are added to the player's previous total. The player who reaches 45 points first is the loser (or the one who has the smallest total is the winner).

2nd Grade

10. Go Ten****
    Number of players: 3

    This game is like Go Fish, but cards 1-9 are used, and the object of the game is to make 10 with 2 cards. All the cards are dealt. (There is no "pond" in this game.) The players first put down all the pairs that make 10. They then ask specific people for specific numbers. For example, John may say to Katie, "Do you have a 5?" If Katie has a 5, she has to give it to John. John then lays this 5 and his 5 in front of himself, face up.

    A player can continue to ask for cards as long as she gets the number requested. If a player is told "I don't have any," the turn passes to the person who said, "I don't have any."

    The person who makes the greatest number of pairs is the winner.

11. Tens Concentration****
    Number of players: 2 or 3

    Cards 1-9 are used, and the object of the game is to find 2 cards that make 10. All the cards are arranged face down in neat rows. The players take turns turning up 2 cards, trying to make a total of 10. When a player succeeds in making 10, he can keep the 2 cards and continue playing. Otherwise, he must turn the 2 cards over so that they are face down again, and the turn passes to the person to the left.

    The game continues until all the pairs have been found. The person who makes the greatest number of pairs is the winner.

12. Salute!
Number of players: 3
Cards 1-10 can be used, but cards going up to 5 might be used at the beginning, when children are not sure about subtraction. The cards are dealt to 2 of the 3 players. The 2 players hold the cards received in a face-down stack. Simultaneously, both take the top cards of their respective piles saying "Salute!" and holding the cards up next to their ears in such a way that each player can see the opponent's card but not her own.

The third player announces the sum of the 2 cards, and each of the other 2 players tries to figure out the number on her own card (by subtracting the opponent's number from the sum that has been announced). The one who announces the difference correctly first takes both cards.

The winner is the person who collected more cards than the other person.

13. Quince
Number of players: 2 or 3

Cards 1-10 are used, and the object of the game is to get as close as possible to a total of 15 without going over it.

The dealer deals 2 cards to each player, including himself, one at a time, face down. Each player looks at the cards received without letting the others see them. The player to the dealer's left begins the game. If his cards add up to less than 15, he may ask the dealer for another card, hoping to get one that will bring his total closer to 15. A player may keep asking for another card every time his turn comes, until he is satisfied with the total and says, "I stand pat," or until he goes over 15 and is out.

For example, in a two-player game, let's say a player receives a 6 and an ace. He asks for another card because 6+1 is too low to win. If the card received is a 2, the total is only 9. If the dealer receives a 9 and a 3, he could stop here but decides to ask for a card, gets a 5, and is out of the game. The other player automatically wins the round and gets a tally mark.

If there are 3 players, the one who has the highest total without going over 15 is the winner of the round and gets a tally mark. The winner of the game is the person who has the most tally marks (or is the first person to get 10 tally marks).

14. Twenty-Twenty
Number of players: 2 or 3

Cards 1-10 and 18 counters are used. The object of the game is to make a total of 20. Each player takes 6 counters and is dealt 5 cards. The remaining cards are placed on the table in a face-down stack. The players take turns putting one card down at a time next to one that is already on the table (see the figure). After putting down a card, each

				O	
				5	
				4	
				3	
O	3	2	7	8	O
				O	

player takes the top card of the stack to have 5 cards again.

When a player puts a card down that makes a total of 20, either vertically or horizontally, she closes the line with 2 counters as shown in the figure. The person who uses up her 6 counters first is the winner.

15. Knock-Knock
Number of players: 2 or 3

A deck of 52 cards is used with the following values: A=1, 2 through 10 are worth the values shown, and the face cards are each worth 10 points. The object of the game is to make the largest total value (or the smallest).

Each player is dealt 4 cards, and the remaining cards make up the drawing pile. The players take turns taking the top card of the drawing pile and discarding one. When a player thinks he has the largest total, he says "Knock-knock," and everybody else has one more turn. The person who has the greatest (or smallest) total is the winner.

3rd Grade

16. Multiplication Salute!
Number of players: 3

This game is played just like Salute! (No. 12 above), except that multiplication and division are used instead of addition and subtraction. When multiplication is still unfamiliar, it is best to use cards going up only to 5. Larger numbers can then be added as small factors become too easy.

17. O'NO 99
Number of players: 2 or 3

The cards have the following values:
All the aces: 99 points
2 through 10: All the spades are "minus" cards. For example, the 2 of spades is worth -2. All the other suits are "plus" cards worth the numbers shown.
Face cards: All the spades are worth -10 points. All the other face cards are worth 10 points.

Five cards are dealt to each player, and the rest of the cards constitute the drawing pile. The object of the game is to avoid making a total of 99 or more.

The first player puts a card down calling out the number (such as "Ten"). He then takes a card from the drawing pile to have 5 cards again. The second player puts down one of his 5 cards announcing the new total (such as "Fifteen"), and draws a card to replace the one used. This procedure is followed around the table. The person who reaches 99 or more loses the round.

The first person to lose 3 rounds is the loser.

Modification into a subtraction game. The same game can be played with subtraction, and the count starts at 99. (The spades become "plus" cards; all the other suits become "minus" cards. The person who reaches zero or less loses the round.)

18. Close to 100 (taken from *Landmarks in the Thousands,* by S. J. Russell & A. Rubin. Palo Alto, CA: Dale Seymour, 1995, p. 109)

Number of players: 2 or 3

Cards 1-9 from one deck are used with a score sheet. Each player is dealt 6 cards. With 4 of the 6 cards, each player makes two numbers that, when added, make a total as close to 100 as possible. For example, a 6 and a 5 can make either 56 or 65. If a 6, a 5, a 4, a 3, a 2, and a 1 are received, 65 + 34 = 99 is as close to 100 as possible. These numbers are written on the score sheet, as well as the difference between the total (99) and 100.

The cards used are discarded, and the 2 unused cards are kept by each player. Four new cards are then dealt to each player so that there will be 6 cards for the next round. When no more cards are available, the discard pile is mixed up and used again. Five rounds are played in this way, and the person with the lowest total score wins.

Close-to-100 Score Sheet

Name: ______________________________

	Diff. from 100
Round 1: ___ ___ + ___ ___ = _____	_____
Round 2: ___ ___ + ___ ___ = _____	_____
Round 3: ___ ___ + ___ ___ = _____	_____
Round 4: ___ ___ + ___ ___ = _____	_____
Round 5: ___ ___ + ___ ___ = _____	_____

Multiplication Games:

How We Made and Used Them

Teachers introduce multiplication in kindergarten and the first two grades in the form of word problems such as the following: "I want to give 2 pieces of chocolate to each person in my family. There are 5 people in my family. How many pieces of chocolate do I need?" Children usually use repeated addition to solve such problems, as Carpenter et al. (1993) and Kamii (2000) describe. By third grade, however, many children begin to use multiplication as they become capable of multiplicative thinking (Clark and Kamii 1996).

Some educators think that teachers should teach for understanding of multiplication rather than for speed. This probably is a reaction to teachers' common practice of making systematic use of timed tests without any reflection, for example, about the relationship between the table of 2s and the table of 4s. In our opinion, children should have an understanding of multiplication and should develop speed. With our advanced third graders in a Title I school, therefore, we have been using games instead of worksheets or timed tests after the children have developed the logic of multiplication. The results have been encouraging. Toward the end of the school year, when the children had played multiplication games for several months, we gave a summative-evaluation test consisting of one hundred multiplication problems to finish in ten minutes. Every child in the class except one (who made two errors) wrote one hundred correct answers within the time limit. This article describes some of the games we used, how we modified commercially made games, and what we learned by using them.

Seven games are described under three headings: a game involving one multiplication table at a time, games involving many multiplication tables and small but increasing factors, and games requiring speed.

A Game Involving One Table at a Time

Rio is a game that is best played by three children. If there are four players, turns come less frequently, and children will be less active mentally. Rio uses ten tiles or squares made with cardboard, fifteen transparent chips (five each of three different colors), and a ten-sided number cube showing the numbers 1–10. For the table of 4s, for example, we wrote the ten products (4, 8, 12, 16, 20, 24, 28, 32, 36, and 40) on the tiles. These tiles are scattered in the middle of the table, and each player takes five chips of the same color.

The first player rolls the number cube, and if a

By Constance Kamii and Catherine Anderson

Constance Kamii teaches at the University of Alabama at Birmingham. Her interest is in figuring out how to improve mathematics education for young children by using Piaget's theory. Catherine Anderson holds a master's degree in education and has taught for more than fifteen years. Twelve of those years have been at Rio Vista Elementary School, a Title I, NCLB, under-performing school. Her focus areas are the development of subtraction and multiplicative thinking.

Children practice a multiplication game.

Figure 1

Easy products and increasingly greater factors

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5				
2	2	4	6	8	10				
3	3	6	9	12	15				
4	4	8	12	16	20				
5	5	10	15	20	25				
6									
7									
8									
9									

5 comes up, for example, he or she puts a chip on the tile marked "20" for 5 × 4. The second player then rolls the number cube, and if an 8 comes up, he or she puts a chip on 32 for 8 × 4. If the third player rolls a 5, the tile marked "20" already has a chip on it, so the player must take it. The third player now has six chips and the first player has four. Play continues in this way, and the person who plays all his or her chips first is the winner.

This is a good introductory game, and most third graders begin by using repeated addition rather than multiplication. As they continue to play Rio, finding products when multiplying by 2 and 10 becomes easy. The next products that they master are multiples of 5 and 3. Multiplying by 6, 7, 8, and 9 is much more difficult. The next category of games is more appropriate after this introduction to all the tables.

Games Involving Many Tables and Small but Increasing Factors

Figure 1 shows easy products of factors up to 5. When children know these products very well, teachers can introduce factors up to 6, 7, and so on.

Examples of games in this category are Salute, Four-in-a-Row, and Winning Touch.

Salute

In Salute, three players use part of a deck of playing cards. At first, we use the twenty cards A–5 and remove all the others (6–K). Ace counts as one. Later, we use the twenty-four cards A–6, then A–7 (twenty-eight cards), and so on.

The dealer holds the twenty cards A–5—or forty cards if two decks are used—and hands a card to each of the two players without letting anyone see the numbers on them. The two players then simultaneously say "Salute!" as they each hold a card to their foreheads in such a way that they can see the opponent's card but not their own. The dealer, who can see both cards, announces the product of the two numbers, and each player tries to figure out the factor on his or her card. The player who announces the correct factor first wins both cards. The winner of the game is the player who has more cards at the end. (We decided that the dealer should hold the deck because when the cards were dealt, the players confused their "winnings" with the cards they had yet to use.)

When this game becomes too easy, children can use cards up to 6, 7, and so on, as stated earlier.

Four-in-a-Row

This is a two-player game that uses a board such as the one in **figure 2a,** eighteen transparent chips of one color, eighteen transparent chips of another color, and two paper clips. Each player takes eighteen chips of the same color to begin the game. The first player puts the two paper clips on any two numbers at the bottom outside the square, such as the 4 and the 5. The same player then multiplies these numbers and puts one of his or her eighteen chips on any 20 because 4 × 5 = 20.

The second player moves one of the two paper clips that are now on the 4 and the 5. If the second player moves one of them from 4 to 3, this person can place one of his or her eighteen chips on any 15 because 3 × 5 = 15. On every subsequent turn, a player must move one of the two paper clips to a different number. Two paper clips can be placed on the same number, to make 5 × 5, for example. The person who is first to make a line of four chips of the same color, vertically, horizontally, or diagonally, is the winner.

The reader may have seen a Four-in-a-Row board such as the one in **figure 2b.** This board is not ideal because some children use only the factors up to 4 or 5. The board in **figure 2a** is better because it does not involve easy factors such as 1 and 2 and more difficult factors such as 7, 8, and 9. The range of factors from 3 to 6 is more appropriate at the beginning because it focuses children's efforts on a few combinations at the correct level of difficulty. When the board in **figure 2a** becomes too easy, teachers can introduce factors 3–7 and a new board made with appropriate products.

We randomly scattered the numbers on the board in **figure 2a** and chose them in the following way. The board includes ten combinations of factors 3–6 because there are four combinations with 3 (3 × 3, 3 × 4, 3 × 5, and 3 × 6), three combinations with 4 (4 × 4, 4 × 5, and 4 × 6), two combinations with 5 (5 × 5 and 5 × 6), and one combination with 6 (6 × 6). Because the board has thirty-six (6 × 6) cells, each product can appear three times and six products can appear more than three times. We usually use the more difficult products for the remaining cells, such as 36, 36, 30, 30, 25, and 24. (We omitted the combinations 4 × 3, 5 × 3, 5 × 4,

Figure 2

Four-in-a-Row boards

24	9	20	15	30	18
12	30	25	36	24	16
36	15	9	18	20	36
16	36	30	25	12	30
12	20	25	15	24	36
24	16	30	9	25	18

3 4 5 6

(a) A Four-in-a-Row board with factors 3–6

1	2	3	4	5	6
7	8	9	10	12	14
15	16	18	20	21	24
25	27	28	30	32	35
36	40	42	45	48	49
54	56	63	64	72	81

1 2 3 4 5 6 7 8 9

(b) A common Four-in-a-Row board

Children run through a practice game before entering into competition.

Figure 3

Two boards for Winning Touch

	3	4	5	6
3				
4				
5				
6				

(a) Winning Touch to 6

	3	4	5	6	7
3					
4					
5					
6					
7					

(b) Winning Touch to 7

6 × 3, 6 × 4, and 6 × 5 from this consideration because 4 × 3, for example, was the same problem as 3 × 4 to our students.)

Winning Touch

Figure 3a shows the board for Winning Touch to 6 and **figure 3b** shows the board for Winning Touch to 7. These boards are modifications of a commercially made game called The Winning Touch (Educational Fun Games 1962). This ready-made game involves all the factors from 1 to 12 and uses a much larger (12 × 12) board than the boards in **figure 3.** A chart on the inside of the cover shows all one hundred forty-four products, and the instructions in the box advise the players to consult this chart when they are unsure of a product.

We took the chart out of the game because it motivates children not to learn products. When children can look up a product quickly, they are deprived of an opportunity to learn it through the exchange of viewpoints among the players. The second modification we made was to eliminate factors less than 3 and reduce the range of factors. For example, when we made the board for factors from 3 to 6, we called it Winning Touch to 6 (see **fig. 3a**). As the class became ready to move on to more difficult factors, we made new boards and called them Winning Touch to 7 (see **fig. 3b**), and so on. We eliminated factors greater than 10, as well as 10, from the game.

Two or three people can play this game. Winning Touch to 6 uses sixteen tiles, on which are written the sixteen products (9, 12, 15, and so on) corresponding to the columns and rows. All the tiles are turned facedown and mixed well, and each player takes two tiles to begin the game. The players look at their two tiles without letting anyone else see them.

The first player chooses one of his or her tiles and places it in the square corresponding to the two factors. For example, 25 must be placed in the column labeled "5" that intersects the row labeled "5." The first player then takes one tile from the facedown pile to have two tiles again. The players take turns placing one tile at a time on the board. To be played, a tile must share a complete side with a tile that is already on the board. Touching a corner is not enough. For example, if the first player has played the tile marked 25, the only products that the second player can use are 20 and 30.

If a player does not have a tile that can be played, he or she must miss a turn, take a tile from the facedown pile, and keep it in his or her collection. In other words, the player cannot play this tile during this turn. The person who plays all his or her tiles first is the winner. If a player puts a tile on an inappropriate square, the person who catches the error can take that turn, and the person who made the error must take the tile back.

When the students are fairly certain about most of the products, it is time to work for mastery and speed. The next section discusses Around the World, Multiplication War, and Arithmetiles.

Games Requiring Speed

Around the World

In this whole-class activity, the teacher shows a flash card and two children at a time compete to see who can give the product of two numbers faster. To begin, the whole class is seated except for the first child, who stands behind the second child to compete. The winner stands behind the third child, and these two wait for the teacher to show the next flash card. The child who wins stands behind the fourth child, and so on, until everyone has had a chance to compete. If the seated child beats the standing child, the two exchange places, and the winner moves to the next person. A child who defeats many others and makes it to the end by moving from classmate to classmate is the champion who has gone "around the world."

Some teachers feel that Around the World benefits only students who already know most of the multiplication facts. When used skillfully, however, this game can motivate students to learn more combinations at home.

Multiplication War

War is a simple game that uses regular playing cards. In the traditional game, the cards are first dealt to two players, who keep them in a stack, facedown, without looking at them. The two players simultaneously turn over the top cards of their respective stacks, and the player who has the greater number takes both cards. The winner of the game is the person who wins the most cards.

Multiplication War is a modification of War. We begin by using cards up to 5 and later add the 6s, 7s, 8s, and 9s gradually. After dealing the cards, the two players simultaneously turn over the top cards of their respective stacks, and the person who announces the correct product first wins both cards. The winner of the game is the player who collects the most cards. It is up to the two players to decide, before beginning the game, what happens in case of a tie.

Arithmetiles

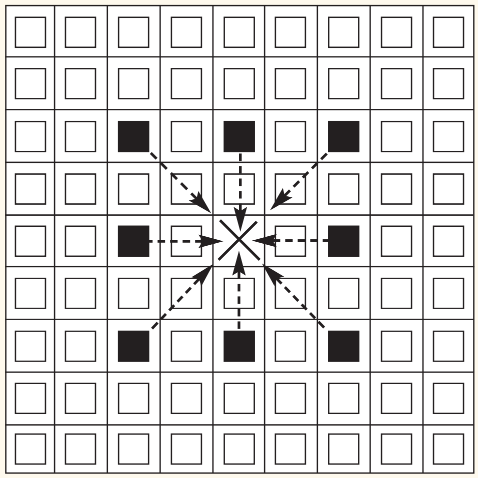
This is a modified version of a commercially made game called Arithmechips (Lang 1990). Arithmechips uses a board that has a grid of eighty-one (9×9) squares and one hundred fifty-six chips. Most of the chips have a multiplication problem on one side and the corresponding product on the other side. To begin the game, eighty chips are randomly placed in every square of the board except the one in the middle marked "X," with the problem side up. The players win chips by jumping over one chip at a time, as in Checkers, reading aloud the problem on the chip they just jumped, stating the answer, and turning the chip over to verify the answer. If the answer is correct, the player can keep that chip.

We modified this game and called it "Arithmetiles." We made the following modifications:

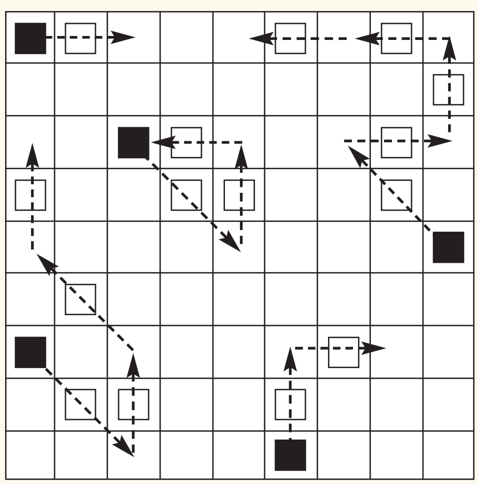
- Eliminating factors of 0, 1, 11, and 12
- Introducing the requirement of speed
- Eliminating the possibility of "self-correction" by not writing a product on each chip

Figure 4

Possible jumps in Arithmetiles



(a) The eight possible jumps at the beginning of the game



(b) Possible moves involving one or more jumps

- Eliminating the requirement of having to read the problem aloud before stating a product
- Introducing levels of difficulty

Arithmetiles is a three-player game played with a 9 × 9 grid that has an "X" in the middle. The game requires eighty problems because players must fill all the squares in the grid except one with tiles that have multiplication problems such as 6 × 7 on them. But because there are only sixty-four combinations of the factors 2–9, sixteen problems must appear on more than one tile. We use the following more difficult combinations on the sixteen tiles: 6 × 6, 6 × 7, 6 × 8, 6 × 9, 7 × 6, 7 × 7, 7 × 8, 7 × 9, 8 × 6, 8 × 7, 8 × 8, 8 × 9, 9 × 6, 9 × 7, 9 × 8, and 9 × 9.

The eighty tiles are placed, facedown, on all the squares except the one marked "X." The first player may play any one of the tiles marked in black in **figure 4a** and jump over a tile into the empty cell marked "X," vertically, horizontally, or diagonally. He or she quickly turns over the jumped tile and announces the product. If the other two players agree with the product and the speed with which the player announced it, the first player can keep the jumped tile. If the product is incorrect, the person who was first to correct it can keep the tile in question. If the other two players agree that the first player gave the answer too slowly, the jumped tile is returned to the grid and the turn passes to the next player.

The X cell is filled after the first play. The second player can choose any tile that he or she wishes to jump vertically, horizontally, or diagonally into the vacated cell. Play continues in this manner, as in Checkers. The person who collects the most tiles is the winner.

As **figure 4b** shows, making two or more jumps is possible. To make multiple jumps, a player must keep his or her hand on the tile while stating the first product and every subsequent product.

Teachers can make Arithmetiles more difficult by eliminating the sixteen easy products of 2–5 that appear in **figure 1.** In this version, we are left with only 64 – 16 = 48 combinations of factors. To have eighty problems, players must use most combinations twice and some combinations only once.

How We Used the Games

Motivation to learn the multiplication tables must come from within the child. The teacher has much to do with the development of this motivation, however. Toward the end of the year, our students' desire to beat the teacher in Multiplication War and Arithmetiles inspired them to learn the tables. A similar motivation was to beat the "stars" in the class. When many students knew the tables rather well, the teacher began to challenge as many groups as possible every day. She briefly played with one group, left the students to continue playing by themselves, and went on to the next group, asking, "Who's going to beat me today?" Some students made flash cards to practice at home, and a few were observed quizzing each other with flash cards on the bus during a field trip.

The children were motivated to learn the multiplication combinations because the games were fun and had a lot of variety. There was no coercion, timed tests, or the threat of a bad grade. Of course, the teacher explained how this knowledge would help in fourth grade, but students largely ignored such talk about next year. When the teacher played every day with small groups of children, they received a stronger message: that games are important enough for the teacher to play.

What about the games was fun to a third grader? Students made decisions every day about which game to play and with whom. Deciding whom to play with was especially a "big deal." Students who had mastered many of the combinations wanted to play against someone at the same level. Those who were not fluent wanted to play against someone at their level so that they still had a chance of winning. A difficult game such as Arithmetiles was not popular with the slower students. They tended to choose games such as Winning Touch, which did not penalize them for lack of speed.

The teacher's role was considerable in giving choices and maximizing learning. We deliberately introduced the more difficult factors one at a time. For example, when we introduced 6 as a multiplier, we played Winning Touch to 6, Four-in-a-Row to 6, Multiplication War with cards only to 6, and Salute, also with cards to 6. We played these games over a two-week period using factors up to 6. After that, we focused on factors up to 7 for about a week, then factors up to 8 and 9.

After a month, when the students had played all these games at four different levels of difficulty, the teacher began to announce on some days that everyone had to play a game with sevens or that everyone had to play Winning Touch at their "just right" level. She also introduced other games such as PrimePak (Conceptual Math Media 2000) and Tribulations (Kamii 1994). The children also benefited from whole-class discussions of strategies. In one of the discussions, for example, one child said that multiplying any number by 8 is easy if "you double it and double it and double it," meaning that 8×6 can be done easily by doing $2 \times 6 = 12$, $2 \times 12 = 24$, and $2 \times 24 = 48$.

As the year progressed, the students selected appropriate partners and games. Some stuck with the same game for a long time; they needed time to develop comfort with certain combinations. Everyone learned the multiplication combinations and enjoyed doing so.

References

Carpenter, Thomas P., Ellen Ansell, Megan L. Franke, Elizabeth Fennema, and Linda Weisbeck. "Models of Problem Solving: A Study of Kindergarten Children's Problem-Solving Processes." *Journal for Research in Mathematics Education* 24 (November 1993): 428–41.

Clark, Faye B., and Constance Kamii. "Identification of Multiplicative Thinking in Children in Grades 1–5." *Journal for Research in Mathematics Education* 27 (January 1996): 41–51.

Conceptual Math Media. *PrimePak*. San Francisco: Conceptual Math Media, 2000.

Educational Fun Games. *The Winning Touch*. Winnetka, Ill.: Educational Fun Games, 1962.

Kamii, Constance. *Young Children Continue to Reinvent Arithmetic, 3rd Grade*. New York: Teachers College Press, 1994.

———. *Young Children Reinvent Arithmetic, 2nd ed.* New York: Teachers College Press, 2000.

Lang, Audrey Clifford. *Arithmechips*. Creative Toys Ltd., 1990. ▲

11

PLAYING WITH NUMBERS

Constance Kamii and Reinventing Arithmetic in Early Childhood Education

Barbara Beatty

Convinced that famed Swiss psychologist Jean Piaget, with whom she studied in Geneva, was right about children's cognitive development, Constance Kamii took on the task of reinventing how young children are taught arithmetic. In this chapter I examine how Kamii came to think that almost everything about traditional arithmetic teaching for preschool through Grade Three was wrong, and how she went on to co-author and write the books *Piaget, Children, and Number* (1976), *Physical Knowledge in Preschool Education* (1978), *Group Games in Early Education* (1980), *Number in Preschool and Kindergarten* (1982), *Young Children Reinvent Arithmetic* (1985), *Young Children Continue to Reinvent Arithmetic, 2nd Grade* (1989), and *Young Children Continue to Reinvent Arithmetic 3rd Grade* (1994), which continue to influence early childhood education today.

One of the leading figures in the movement for constructivist preschool education (the notion that young children construct concepts on their own, through play with materials and games, in carefully planned classroom settings with supportive, interactive teachers), Kamii has tirelessly promoted her beliefs nationally and internationally. Her ideas were perceived as so radical, especially that of the harmfulness of directly teaching young children algorithms, that she eventually had to move from Chicago to Alabama, where she could find a few principals who would allow her to experiment in their schools.

In the tradition of preschool educators such as Friedrich Froebel, Patty Smith Hill, and Harriet Johnson, Kamii believed that children learned basic concepts as well as sophisticated knowledge through manipulation of physical materials. Throughout her long career, Kamii argued that playing with blocks and other preschool materials and games was how children learned arithmetic in a deep and lasting way. Ahead of the times, Kamii's worries about the effectiveness of arithmetic teaching and learning are the subject of great concern currently, when mathematical knowledge and weaknesses in math teaching have been identified as one of the greatest problems in American education.

An International Childhood and Education

Constance Kamii's radical ideas about how young children learn and should be taught were influenced by her international background and education. Initially a Japanese citizen, she was born in Geneva in 1931, where her father was working for the International Labor Organization. Her parents, Kamii says, had "very democratic ideas." She grew up speaking French as her first language, despite her parents' efforts to teach her Japanese (Kamii, 2008). In 1939, when she was eight, her father took the family back to Japan, where Kamii lived during World War II. She remembers the bombings every night. She remembers the "at-ta-ta-ta-ta-ta" sound of machine guns during the day and wondering "am I still alive?" after each attack. Educated in Japanese schools, which during this period were quite regimented (a pedagogical formality she later rejected), Kamii looked back on her early education in Geneva as a time when she was free to explore and learn on her own.

Kamii's life was affected by American prejudice against the Japanese. Her mother, who was Japanese-American, lost her citizenship after World War II, but then regained it, as other Japanese-Americans did. Not naturalized until later in her life, Kamii's legal status as a Japanese citizen had an impact on her career path. Kamii became interested in psychology and education when she came to the United States in the 1950s, where her mother and brother had moved. Kamii attended Pomona College in California, and after graduating in 1955 with a major in sociology, went on to the University of Michigan, which gave her a scholarship, to get a Master's degree in the School of Education. With a student visa that required her to continue studying, she stayed on at Michigan to get her doctorate in psychology and education.

At Michigan, Kamii met fellow student David Weikart who in 1961 helped her get a job as a half-time counselor in a junior high school in the nearby Ypsilanti Public Schools while she was still a graduate student. Weikart, who would go on to become a world-famous preschool researcher, had begun working in Ypsilanti in 1957 as a psychological tester for developmentally delayed children and a year later became the director of special education. With Weikart, Kamii began focusing on the antecedents of learning problems (Kamii & Weikart, 1963; Weikart, 2004).

Piagetian Preschools

Kamii's ideas about arithmetic teaching and learning were grounded in research she did with Weikart at the now iconic Perry Preschool Project in Ypsilanti, Michigan. Kamii then broke with Weikart over how Piaget's concepts should be implemented, and went on to develop her own ideas about young children's learning.

Based on their experiences working with children with special needs, Kamii and Weikart wondered whether something might be done before children entered school that would help prevent later problems. As a counselor, Kamii noticed that the children getting kicked out of class were from low-income backgrounds, "troublemakers," and that the trouble started right away, in kindergarten. Kamii began doing research for her dissertation that gave her more evidence that reaching what today would be called "at-risk children" early was very important. With a list from the welfare department, she studied the child-rearing practices of African-American mothers living in deep poverty and saw how difficult it was for many of them to provide their four-year-olds with the kind of enriched educational environment that young children from middle-class backgrounds received.

With "compensatory education," the idea that schools could make up for the "cultural deprivation" of children from low-income backgrounds, in full sway and growing concerns about the effects of poverty and social inequality, Weikart and Kamii were part of a new wave of researchers looking to preschool education to help the children of the poor (Beatty, 2009, 2012; Bereiter & Engelmann, 1966; Deutsch, 1967; Gray & Klaus, 1965). Determined to prove that preschool education could raise poor children's IQ scores and prevent school failure, Weikart convinced the Ypsilanti school district to let him begin an experimental preschool at the Perry Elementary School in 1962, which became the Perry Preschool Project. Initially seen as a form of remedial preschool intervention, the project combined the ideology of special education with early childhood education. Enabled by the county's forward-looking move of approving new funding for special education, Weikart realized that public money could be spent on three- and four-year olds with special needs (Weikart, 2004).

When Kamii joined the project in 1964, she immediately became immersed in preschool, compensatory, and special education—all major influences on her later work. Sent into the Perry School neighborhood in the summer to recruit low-income African-American three-year-olds whose low IQ test scores, most in the 70–85 range, predicted they would have trouble in school, Kamii helped assign the children randomly for admission to the experimental preschool or a control group, to be followed longitudinally. Working with Perry Preschool social worker Norma Radin, Kamii realized that many African-American mothers living in difficult circumstances felt a strong need to protect their children from harm, and thus "overprotected" and "shielded" them, compared to white middle-class mothers who wanted to expose their children to challenges and were freer to do so. In articles she published with Radin, Kamii described social class differences in the child rearing styles of African-American mothers and argued that social class, not race, was the important variable, providing more evidence that African-American children from low-income backgrounds would benefit from being in a preschool that would challenge them, in a safe environment (Radin & Kamii, 1965; Kamii & Radin, 1967).

The Perry Preschool Project was designed to give three- and four-year-old at-risk African-American children the same kind of enriched preschool education that middle-class children got in nursery school. The children attended three hours a day, five days week, for the length of the school year for two years, and got 90 minute weekly home visits from their teachers, who had to be fully certified. Kamii did pre-tests and post-tests on the children. After one year, the Stanford-Binet IQ test scores of the children in the program went up, way up, an average of 15 points, which put them into the normal range, a big deal in an era when most psychometricians still believed that IQ was an inherited, fixed characteristic (Weikart, 2004, 52–54).

After the second year, however, when the Perry Preschool children entered elementary school, their IQ test scores started to go down. Weikart wondered whether the Perry Preschool curriculum might be the problem. He had initially wanted a curriculum based on John Dewey's philosophy of active learning combined with the Perry Preschool teachers' training in traditional nursery school education, but was disappointed that the teachers did not seem to be doing much planning. The children were given lots of time for free play but were not getting any special academic help. During the first year of the program, after a little boy threw a chair across the room, the teachers realized that they needed to be more proactive. They began to give more guidance and verbal instructions, and talked to the children a lot, in what became known as a "verbal bombardment" approach (Weikart, 2004, 64–65).

The Perry Preschool curriculum evolved further when Weikart discovered Piaget, while reading a review of J. McVicker Hunt's influential 1961 book *Intelligence and Experience*, which summarized Piaget's theories and emphasized the role of the environment in child development and education (Hunt, 1961). Weikart contracted for the teachers to be given Piaget workshops and studied the work of Israeli preschool researcher Sara Smilansky, who focused on how teachers should ask disadvantaged children to plan what they were going to do in their play before they did it (Minkovitch, 1972; Smilansky, 1968). Weikart consulted with psychologist Robert Hess of the University of Chicago, who suggested that the children should review their play after each session. These ideas came together in the Perry Preschool's "plan-do-review" approach, in which children met with a teacher for about 10 to 15 minutes to plan their play, played for about 45 minutes to an hour, and then met with the teacher again to review what they learned from their play (Weikart, 2004, 65–66).

When Kamii joined the Perry Preschool Project as a Research Associate in the second year of the program, she was dissatisfied with the curriculum, too. It still seemed like a traditional nursery school. When she asked the teachers what it was good for, they said language and emotional development. What about the "three Rs?" Kamii asked, knowing that the children needed help with literacy to do well in school. So Kamii started reading curriculum books, and found "generalities," "Nice, sweet generalities." Kamii had heard about the Direct Instruction, academic skills-based preschool program that Carl Bereiter and Siegfried Engelmann had started at the University of Illinois, but worried if children were having trouble learning to read in first grade it would be much harder for them when they were three (Bereiter & Engelmann, 1966).

When Norma Radin gave Kamii a copy of John Flavell's (1963) *The Developmental Psychology of Jean Piaget*, a scholarly exegesis on Piaget's theories, Kamii realized that she had found a "goldmine" that could be applied to early childhood education. She told Weikart that the Perry Preschool program curriculum needed to be even more directly Piagetian. Using the language of compensatory education, Kamii was convinced that "disadvantaged" children had "cognitive deficits," as she later wrote in an article with a Perry Preschool research assistant, because they had not gone through the Piagetian stages. They needed a curriculum that would help them progress through "the transition from sensory-motor intelligence to conceptual intelligence," so that they could acquire cognitive skills (Sonquist & Kamii, 1967).

To create a curriculum that focused on teaching specific Piagetian concepts, Kamii decided she needed to learn more, from Piaget directly. In June of 1965, when she graduated with her doctorate from Michigan, she gave herself the present of going back to Geneva. She got to Geneva just in time to hear Piaget's last lecture of the semester. Mesmerized, she could understand Piaget's French easily. While in Geneva, Kamii met David Elkind, who was finishing up a postdoctoral fellowship. Elkind became an influential professor of early childhood education at the Eliot-Pearson School at Tufts University and would soon become one of the main "popularizers" of Piaget in the United States. She also met many other Piaget researchers with whom she would later collaborate, and was especially impressed by the work of Piaget's close colleague and co-author Barbel Inhelder, who planned the experiments that children were doing with objects, which became the basis for Piaget's increasingly complex theory of logico-mathematical development (Beatty, 2009; Hseuh, 1997).

When Kamii came back to the Perry Preschool project she started applying Piaget's theories in earnest. With Norma Radin, she wrote a framework for how Piagetian stages and sub-stages could form the basis of a preschool curriculum, and then translated the framework into activities. She showed the teachers how they could use regular nursery school activities to help children construct the Piagetian concept of object permanence with games in which the teachers hid objects, as Piaget had done with his children Jacqueline and Laurent. Kamii demonstrated how to make a duck out of clay, to help children understand that the duck was a "symbol" that "represented" a real duck. She told the teachers to ask the children to put blocks in order from smallest to largest, and to organize the doll corner so that the children would order the dishes and sort the doll clothes by size, to teach classification and seriation. She suggested asking the children to put a cup *on* the table and to jump *over* a rope, and what came next in the daily schedule of play-time, outdoor-time, and snack-time, to teach spatio-temporal relationships. She showed how asking the children what would happen when they pushed their juice cup or a block tower hard could be used to teach cause-and-effect relationships. She demonstrated how pointing out that when a cookie was broken into two pieces it was still the same cookie, could be used to teach conservation of quantity. Almost everything in the nursery school environment, Kamii argued, could be manipulated to turn it into an opportunity for disadvantaged children to learn Piagetian concepts and further their cognitive development (Kamii & Radin, 1967; Sonquist & Kamii, 1967).

Indicative of the kinds of tensions that erupt perennially in early childhood education over fine points of pedagogy, relations between Kamii, the teachers, and Wiekart became strained. The teachers objected that they were being told what was theoretically correct and incorrect and what to do in their classrooms. Since Kamii had not been a teacher, they thought that they knew more about the children's individual needs and how to plan for them than she did. Kamii objected that Weikart was not applying Piaget directly enough. Weikart decided that he would trust the teachers' judgment and that the Perry Preschool curriculum would never be a "strictly Piagetian-based program," it would be a "cognitively oriented curriculum." Kamii resigned from the Perry Preschool Project and left for a year of postdoctoral study in Geneva (Weikart, 1971; Weikart, 2004, 67).

Kamii spent 1966–67 in Geneva taking courses with Piaget and Inhelder at the University of Geneva, where Kamii became completely immersed in Piagetian theory. She also began, doing Piagetian experiments with children herself., Not thinking about what she would do next, Kamii was contacted by her Perry Preschool colleague Norma Radin, who had received a federal grant to start another preschool program in the Ypsilanti Public Schools. As Curriculum Director of theYpsilanti Early Education Program for three years, Kamii continued developing Piagtian preschool activities. Her ideas about what to do radically changed. She read a 1964 article "Piaget Rediscovered," by Eleanor Duckworth, a Canadian Piaget researcher who would have a great impact on science education for young children. After reading Duckworth, Kamii began worrying about trying to teach Piagetian concepts too directly. Duckworth said not to teach conservation by having children pour water back and forth from different sized beakers and asking questions or pointing out that the amount of water had not changed, let the children gradually discover it themselves. Piaget did not think that "intensive training of specific tasks" was useful, Duckworth wrote, because it did not affect children's general understanding (Duckworth, 1964).

Duckworth, and especially Hermina Sinclair, a Dutch Piagetian from Geneva who came to consult in Kamii's Ypsilanti preschool program every year, convinced Kamii that her earlier ideas were wrong. Kamii realized that she had been doing what beginners did, trying to teach Piagetian tasks instead of understanding the larger processes of development. Sinclair told Kamii that teaching the tasks, hiding objects, and pouring of water back and forth, was like taking soil samples, fertilizing one sample, and sticking it back, instead of "fertilizing the whole field" (Kamii, 2008) As Kamii put it, it had become:

control group. Although the children's IQ test scores did not go back up, in third grade their achievement test scores and teacher ratings began to rise. In 1984, when they were 19, 59 percent of the former Perry Preschool children were employed, compared to only 32 percent of the group that had not attended preschool; 67 percent had graduated from high school or its equivalent compared to 49 percent; 38 percent compared to 21 percent had gotten college or vocational training; only 31 percent compared to 51 percent had been arrested or detained; and only 16 percent compared to 28 percent had been assigned to special education. The Perry Preschool group also had higher earnings and only about half as many teenage pregnancies (Berrueta-Clement, *et al.*, 1984). Weikart and his associates calculated that every dollar invested in the Perry Preschool gained $7.01, mostly in savings on special education, prisons, and other costly public services. Although criticized by some statisticians, the figures circulated rapidly. Politicians listened. The Perry Preschool Project, with its Piaget-influenced, cognitively-oriented curriculum that Kamii helped design, became a powerful model for why the United States needed to increase support for preschool education (Berrueta-Clement *et al.*, 1984; Schweinhart *et al.*, 2005).

Testing Piaget

In the late 1960s and early 1970s, Kamii and other Piagetians mounted a challenge to behaviorism and the entire edifice of IQ testing that had dominated American psychology since the days of Lewis Terman at the beginning of the twentieth century. By the late 1960s, Piaget was becoming well-known in the United States, and Kamii was becoming known as a Piaget researcher. David Elkind's article "Giant in the Nursery – Jean Piaget," made a splash in *The New York Times* Sunday magazine (Elkind, 1968). Test companies took notice. In 1969, the California Test Bureau, a division of McGraw-Hill, convened a conference to see if developmental and educational psychologists could develop a standardized Piagetian test, an Ordinal Scales of Cognitive Development, based on the kinds of problems Piaget gave children, to measure developmental and intellectual maturity. Piaget and Inhelder were invited, as were many influential American psychologists, psychometricians, and early childhood educators, including Millie Almy of Columbia University's Teachers College, whose 1966 book *Young Children's Thinking* introduced many preschool educators to Piaget, and Selma Greenberg, who directed the Head Start program for African American families in the Mississippi Delta, and Kamii (Green et al., 1971).

Held at the Monterey Institute for Foreign Studies, the conference began with an opening address by Piaget, in which he stated that he was not an expert on ordinal scales, a succession of tasks or questions designed to measure an individual's performance compared to that of subjects in the group upon which the test was based. Nor was he sure, Piaget said through his translator Sylvia Opper, that ordinal scales really measured the abilities they purported to measure (Piaget, 1971). The second day of the conference, held at a hotel in Carmel, began with a paper by David Elkind comparing similarities and differences between Piaget's views on intelligence with those of psychometricians who used IQ testing.

When Kamii found out that Siegfried Engelmann, who, in the early 1960s, with Carl Bereiter, had started a preschool for educationally disadvantaged children, housed at the University of Illinois at Urbana-Champaign, was going to give a paper, she asked to give a comment on it. Antithetical to everything Piaget, and Kamii, stood for, Bereiter's and Engelmann's program, which developed into what is now known as Direct Instruction, was based on behaviorist methods for teaching academic content in language, reading, and arithmetic in short, tightly-scripted, adult-centered lessons. In a lesson on the concept of weapons, for instance, the teacher shows the children a picture of a rifle, praises them if they say it is a gun, especially if they say it in a full sentence in standard English, and has the class repeat the rule and clap rhythmically saying "If you use it to hurt somebody, then it's a weapon." "You use it to POW POW – hurt somebody," the teacher says, and after a series of sing-song question and answers, the preschoolers have supposedly been taught the concept of a weapon, in a quick, two-minute "teaching segment" (Bereiter & Engelmann, 1966, 105–110).

Knowing that Engelmann would claim that he could teach Piagetian concepts directly, not through play, Kamii asked him if she could come to his preschool to test the children. To her surprise, he said yes. Kamii designed some clever experiments that she thought would reveal that Engelmann's preschool children did not really understand physical knowledge about how the world worked, which Piaget said had to be learned through play with objects. So she got a big cake of Ivory soap that would float and a small bar of hard soap that would sink and some other objects, and designed questions to elicit the children's predictions and explanations.

When Kamii and her Ypsilanti Early Education Project assistant Louise Derman arrived at Engelmann's preschool, they soon realized that Engelmann had taught the children basic rules, but that the children could not explain the rules. When asked whether a block would float, for instance, one little boy, Carl, said yes, "Because it is wood." When told it was heavy and allowed to feel it, Carl changed his mind, and put it in a pile of things that he thought would sink, instead of explaining the rule, as a child who understood the concept would. The pieces of soap were especially puzzling to the children. When they saw that the bigger piece of Ivory soap floated they were surprised and said things like "That's not what it's supposed to do." One little girl, Ann, said that both pieces of soap would sink, because they both were soap (Kamii & Derman, 1971, 130). Kamii and Derman concluded that their testing proved that children had to build up sensorimotor knowledge slowly, and that being in a preschool that let them do this was how it happened.

When Engelmann gave his paper at the conference, which Kamii had not seen beforehand, Engelmann critiqued Piaget for lacking an explanation for how children learned. Piaget's theory was nothing "more than a set of accurate descriptions about the performance of children at different ages," Engelmann said. It might as well have been based on "learning-producing" rays from outer space. Piaget did not provide a theory that "clearly implies instruction, lack of instruction, or evaluation of instruction" (Engelmann, 1971, 120–121).

Just as Kamii had expected, Engelmann claimed that he had successfully taught Piagetian conservation tasks directly, through short lectures. Engelmann had found, he said, that kindergarten children could learn the principle of conservation of quantity without playing with objects, without pouring water back and forth, seeing it poured, or even seeing a diagram of it, "after 54 minutes of instruction, distributed over a 5-day period" (Engelmann, 1971, 126). It was simple to teach what Piaget called development, Engelmann claimed, children "are taught."

In the response she gave after Engelmann's presentation at the conference, Kamii disagreed. Young children could not learn logic "without taking into account the natural developmental sequence that Piaget described." In fact, Kamii argued, the verbal rules Englemann had taught the children made it harder because they blocked the children's "intellectual contact" from coming to grips with the real objects. Engelmann had said that the Piagetian model was an inefficient way to teach. On the contrary Kamii said, imposing rules could mask children's multiple explanations, but not eliminate their intuitions, some of which were incorrect. The Piagetian approach to teaching, Kamii said, was not to leave children alone, but to provide situations and materials through which children could build up knowledge interactively and thus progress to the next stage of development (Kamii & Derman, 1971,142, 143, 145, 146).

The confrontation between Kamii and Engelmann was a standoff. Engelmann said that he knew that his instructional methods needed to be improved. The problem, he argued, was that he had not taught a rule that would allow children to generalize sufficiently, so "faulty instruction" was a problem. Engelmann also gave a more basic answer, however, that he thought explained away some of Kamii's results. The reason the little girl Ann had had so much trouble with the soaps was "appallingly simple." She had been absent two of the days when compensating for changes in rectangular objects had been taught (Engelmann, 1971, 147, 126).

Kamii had defended Piagetianism, at a conference Piaget himself had attended. While she had not convinced Engelmann that he was wrong, she got affirmation from Piaget's co-researcher Barbel Inhelder that Kamii had made some good points (Kamii, 2012). Engelmann continued to work on his behaviorist preschool methods, but behaviorism was on the wane. Cognitive-developmental models were rapidly becoming the dominant approach in preschool education.

Back to Geneva

Knowing that she needed to learn more about Piagetian theory, so that she could design better preschool curricula, in 1970 Kamii left Ypsilanti for good and went back to Geneva for another postdoctoral year. This time she had been invited to do research at Piaget's International Center for Genetic Epistemology, a high honor. Each spring, Piaget would announce what the topic would be for the next year. Over the summer, research fellows dreamed up an experiment, a problem related to Piaget's announced topic. All summer the research fellows, Piaget's "slaves" as Kamii called them, of whom she was one, played with the apparatuses they were building, worrying if Piaget would approve them in the fall.

Kamii designed a problem with a balance beam, in which children were to predict what would happen when they tried putting small metal washers at different points on the balance beam and explain why. Kamii brought her balance beam apparatus to the first session of the year, the first Monday in October, when research fellows had to present their plans. She was anxious. To her relief, Piaget, the Patron, as his students called him, approved of her experiment. Kamii took her apparatus to schools in Geneva, a researcher's paradise because Piaget had a standing arrangement that his researchers could simply walk into a school on any afternoon and announce to a teacher that they were going to take children out of the classroom to study them, a blanket permission that Kamii would later find very hard to get.

Strong believers in collaboration, Piaget and Inhelder, who collaborated on everything themselves, insisted that researchers work in pairs, a habit Kamii continued in much of her own research. Kamii's partner for most of the years she kept coming back to Geneva was Sylvia Parrat. To get a feel for the range of development, Piaget required researchers to start by interviewing a four-year-old, a six-year-old, and a ten-year-old, and then fill in more children of different ages to test the theory at different levels. Kamii and Parrat spent hours together talking about their research problems, did a year of one-day-a-week observations, and were critiqued by Piaget and other members of the seminar, weekly. At the end of the year there was a research symposium to which Piaget invited renowned senior researchers from around the world, at which the fellows presented their findings.

Like that of other of Piaget's students, Kamii's research contributed directly and indirectly to Piaget's and Inhelder's own work. At the end of the year, Kamii and her partner would turn in about a 15-page report on their research, which Piaget and Inhelder would take up to their chalet in the mountains, along with the reports of the other "slaves." Kamii never knew where or if pieces of her research might turn up in Piaget's and Inhelder's books. Kamii and the other fellows were credited in references or acknowledgements, but the Patron acted as if he owned their work. Sometimes Kamii would barely recognize her research when she saw it later, in part because Piaget made up theoretical explanations written in long, dense, complicated sentences. Eventually, usually after about three years, Kamii's research would appear in some form in the *Archives de Psychologie*, the journal begun at the University of Geneva in 1902. Soon Kamii was asked to take charge of a research seminar on Piagetian methods herself, which she alternated teaching in the spring and fall at the University of Geneva for twelve years, with Eleanor

Duckworth, another Piaget disciple, during which time Kamii became more and more convinced that Piaget's ideas were scientifically correct.

Playing with Numbers, Objects, and Games

From the mid-1970s to 1980, while Kamii was going back and forth from Geneva, she collaborated with Rheta DeVries, another Piagetian psychologist and educator, to write three very influential books that helped make Kamii widely known in early childhood education. DeVries, who Kamii had met at one of the many Piaget conferences held in the United States throughout the 1970s, helped Kamii get a job at the University of Illinois at Chicago Circle. An elementary school teacher, DeVries had completed her doctorate in psychology at the University of Chicago under Lawrence Kohlberg, who became famous for applying Piaget's stage theory to moral development (DeVries & Kohlberg, 1987). As Kamii and DeVries heard stories from their Masters' students about terrible arithmetic teaching, Kamii and DeVries became convinced of the need for a book on Piagetian approaches to arithmetic for young children. Kamii knew that Piaget's theories were especially strong in the area of logico-mathematical knowledge, and that teaching reading was a crowded field, so she decided to focus on arithmetic. Kamii and De Vries had plenty of time to design Piagetian arithmetic teaching activities because Kamii lived in De Vries's apartment building in the Hyde Park section of the city. They tested their ideas in child care centers in Chicago, Evanston, and at the University of Illinois, Chicago Circle Preschool.

In their 1976 book *Piaget, Children, and Number*, Kamii and DeVries asserted that everything about how young children were traditionally taught numbers was wrong. The names of numerals, number of things in a group, and how to count were arbitrary "number facts," the teaching of which was useless, even potentially harmful. It was rote memorization of arbitrary social knowledge, without real understanding. Numbers are not "out there" in numbers of objects. Children have to play with objects and order and group them mentally, Kamii and DeVries thought, and then see that "eightness" is a relationship. To understand eight or any other number, young children have to construct a concept of eight, and no amount of counting practice, or drill will help. Throw out all of "one, two, three," Kamii and DeVries, said, children have to play with objects to understand numbers, before they can go on to more complicated arithmetic (Kamii & DeVries, 1976, 7–10).

Teachers should not just leave children alone, however, Kamii and DeVries said, but rather teachers should help children construct number concepts by thoughtfully using familiar objects and asking good questions. Arithmetic learning happened all of the time, not just during "math time." At snack time teachers should ask "Do we have enough cups for everyone?" or "Do we have too many cups?" Kamii and DeVries also questioned the usefulness of many existing math "manipulatives," as specially designed objects for children to use to learn arithmetic are called. Cuisenaire rods, the colored wooden rods that come in multiples by length, Montessori's graduated materials, and most other math manipulatives did not help, Kamii and DeVries said, because young children understand number as "one of" an object, not that a longer object means more, or that two is included within a rod that is twice as long.

It was especially important for children to check their own answers, Kamii and DeVries argued. Teachers should not give children the right answer or tell them that they are wrong, a very controversial notion in a field where getting the right answer had long been the goal. Instead, teachers should try to figure out how children themselves are thinking. Did the child get the right answer by accident? Did the child construct how to do it logically, but make a computational error? Getting the wrong answer for the right reason was better than getting the right answer for the wrong reason, Kamii and DeVries stated, flying in the face of how arithmetic was customarily taught (Kamii & DeVries, 1976, 11–26).

Piaget, Children, and Number was an immediate success, even though it almost did not get published. When Kamii and DeVries sent the manuscript to the National Association for the Education of Young Children, NAEYC sat on it for a long time. Kamii thinks this was because it was more theoretical than books NAEYC usually published. When it finally came out, Kamii became famous in the early childhood education community and began giving talks to huge audiences at preschool conferences. Despite the book's popularity, Kamii was dissatisfied. In the 1982 edition that she wrote on her own without DeVries, to "correct the errors and inadequacies" in the original volume, Kamii thanked Hermina Sinclair, and especially Eleanor Duckworth, for helping her see that teachers should not be explicitly teaching Piagetian tasks. In the second edition "teaching" numbers is in quotation marks, because "number is not directly teachable," Kamii says. "How *precisely* the child constructs number is still a mystery," Kamii wrote, just as how children learn language is a mystery (Kamii, 1982, 21, 25; Lascarides & Hinitz, 2000, 134).

Essential to Kamii's approach and part of what made it so original was her emphasis on children's autonomy. Kamii had had an epiphany. Autonomy was the aim of education, not development, an issue about which she and DeVries disagreed. Many in the early childhood education community saw intellectual development as the goal. Kamii did not, and appended a keynote address she had given on autonomy to her 1982 *Number in Preschool and Kindergarten*. Like most Americans, Kamii had been deeply influenced by the events of the late 1960s and 1970s. Martin Luther King Jr. was one of her biggest heroes, along with Copernicus. She praised former Attorney General Elliot Richardson for acting autonomously by defying his boss Richard Nixon in 1973 by refusing to fire Special Prosecutor Archibald Cox who was investigating the Watergate scandal. Piaget's theory of moral development explained why some people were able to act autonomously, Kamii argued. Piaget showed how children could construct a sense of autonomous morality, through interactions with other children and adults, when children were given the opportunity to make decisions and experience the consequences of their decisions (Kamii, 1981).

Following their book on number, Kamii and DeVries went on to write about physical knowledge, concepts about the way the physical world works that children construct from playing with objects and observing reactions and transformations, another type of development that Piaget and Inhelder studied. As Kamii and DeVries explained in their 1978 book, *Physical Knowledge in Preschool Education*, originally published by Prentice-Hall, not NAEYC, the Piagetian approach avoided the "verbalism" of traditional science education. In a traditional textbook lesson on crystals, for instance, the teacher shows children crystals and rocks; explains what they are; gives children salt, bluing, water, and ammonia; and in one hour crystals begin to form. As with their book on number, Kamii learned from observing real teachers and children how children could learn science more effectively. Kamii and DeVries encouraged teachers to let children invent experiments on their own, add different things together and predict what might happen, so that the children would be surprised by some of the results, the way real scientists would be (Kamii & DeVries, 1983, 3–4).

As with understanding of the properties of number, understanding physical knowledge did not develop by leaving children alone, Kamii and DeVries stated. Quoting from *The Having of Wonderful Ideas* by Eleanor Duckworth, Kamii and DeVries argued that content was important, children had to know enough about something to be able to think of other things to do and ask more complicated questions. But, harking back to Engelmann's attempts to directly teach floating and sinking, Kamii and DeVries said that children made "absurd statements precisely because" they "tried to use the specific bits of verbal knowledge that had been stuffed into" their heads. Instead, for example, teachers could give children boards and rollers to sit on and stand on to experience different kinds of movement relationships (an idea Kamii had gotten from a book on the history of engineering that described how rollers and boards were used to build the pyramids); give children balls to aim at different block towers to observe ricocheting and other effects; build inclines from blocks; set up pendulums; and provide for water play (Kamii & DeVries, 1983, 21, 31, 311).

In their third book together, *Group Games in Early Education* (1980), Kamii and DeVries emphasized what was becoming known as "constructivism," the notion that children constructed knowledge themselves through interactions with the environment, peers, and teachers, especially through play. In a foreword to the book, Piaget wrote that play was "a particularly powerful form of activity that fosters the social life and constructive activity of the child," and noted that Kamii and DeVries had been inspired by his famous study of children playing marbles from his 1932 book *The Moral Development of the Child*. Filled with long quotations from Piaget's writings, *Group Games in Early Education*, also contained a single-authored appendix by Kamii in which she explained why Piaget's constructivism was scientifically-derived (Kamii, 1980). Although not a panacea, play,

FIGURE 11.1 Jean Piaget observing Constance Kamii facilitate young children's play with manipulatives at the Perry Elementary School, Ypsilanti, Michigan, October 1967. (Personal collection of Constance Kamii)

Kamii said, was the best way for children to learn, construct knowledge, and become morally autonomous thinkers, and games were a great way for children to do this. The book also contained a photograph taken when Piaget visited Kamii in Chicago while he was on a trip to receive an honorary degree from the University of Michigan.

Kamii and DeVries said that they wrote *Group Games in Early Education* in part because they thought that the pendulum had swung "too far from group instruction to overly individualized instruction." They also thought that the educational benefits of playing games were undervalued. Many teachers and principals were afraid of using group games because "parents complain when children play games and do not bring worksheets home," Kamii and DeVries said. Learning from games was an "alternative to traditional, academic methods," and could be useful with older children, as well, although "*in*struction" became "increasingly necessary and desirable as the child grows older, but older students would learn more if they had *con*structed knowledge when they were young" (Kamii & DeVries, 1980, xii, 33).

Playing games raised the thorny issue of competition, which Kamii tackled head on in a single-authored chapter. She knew that most preschool teachers objected to group games because they were competitive, because they thought there was "already too much competition in our society" and in the upper grades, because children who lost got upset, and because children should compete with themselves, not with each other. Kamii said that teachers could help children see that they were comparing performances, not competing for a "thing," and that teachers could handle competition more casually, by saying that it was OK to lose, so that children did not become obnoxiously boastful. As to competition in the world, the games she and DeVries were suggesting, Kamii wrote, were different because the children decided and agreed on the rules, with help from the teacher, and did not get rewards or prizes. As to feeling badly about losing, Kamii said that teachers should stress that it was just a game, that the loser was not "inferior, incompetent, or worthy of rejection," and not force children who did not want to play. Teachers should help children develop into "fair players" who could "govern themselves" and learn how to "judge their own success." Preschool was a good time to begin this process creatively through games such as block races, tag, marbles, pin the tail on the donkey, card games, and board games (Kamii & DeVries, 1980, 189, 197).

Enormously successful, the books Kamii and DeVries wrote on number, physical knowledge, and group games became classics in early childhood education both nationally and internationally. With Japanese, Korean, Spanish, Portuguese, and Chinese editions, Kamii's work did much to extend Piagetian ideas throughout the world.

Reinventing Arithmetic

After revolutionizing the way many preschool teachers thought about how young children learned about numbers, physical science, and games, Kamii mounted an assault on how all of arithmetic should be taught from preschool to third grade. When, in the early 1980s, Kamii moved up into the primary grades—the *sanctum sanctorum* of "the basics," the three "Rs," the bedrock of American education—she encountered more resistance. Her ideas challenged assumptions that had been in place since the days of one-room schoolhouses in the 1800s. This was territory into which other developmental psychologists had trod, as well, with little lasting impact. In the early 1900s, the father of developmental psychology G. Stanley Hall and progressive educator John Dewey had tried to make arithmetic instruction more natural and practical, with little success in the public schools, where the texts and testing of educational psychologist Edward L. Thorndike ruled the day (Beatty, 2006; Cline, 1982; Finkelstein, 1989; Monroe, 1917). *The Thorndike Arithmetics* laid out how arithmetic should be directly and efficiently taught through practice, word problems, and drills, and how children's learning should be scientifically measured by school achievement tests (Beatty, 2006; Clifford, 1984; Thorndike, 1917, 1922). As Kamii soon discovered, this behaviorist approach, which dominated elementary education in the United States, presented a formidable obstacle to her research.

In a sequence of four books and three videos published by Teachers College Press between 1984 and 2000, Kamii laid out a completely new approach to teaching arithmetic, in which children constructed arithmetical concepts themselves with the help of their teachers. Although similar in some ways to the "new math" of the 1960s, the revolution in math teaching designed by college math professors, Kamii's methods were based on Piaget's theory of cognitive development and collaboration with elementary school teachers. She proposed the radically progressive idea that teachers and parents and schools should trust that children had the ability to learn math through normal, universal processes of development, and that if allowed to do so, they would be confident about their abilities and not suffer from math anxiety or phobia. "Every normal student is capable of good mathematical reasoning," Kamii quoted from Piaget, "if attention is directed to activities of his interest, and if by this method the emotional inhibitions that too often give him a feeling of inferiority in lessons in this area are removed" (Piaget, 1973, 98–99: Kamii, 2000, xii).

Kamii called her approach "reinventing arithmetic," a term she got from Eleanor Duckworth, a notion Kamii based on her own research with children in Geneva. Kamii's new line of research began with one teacher, Georgia DeClark, the only first grade teacher in Kamii's Introduction to Piaget course at the University of Illinois, whom Kamii credited as the second author of the 1985 edition of *Young Children Reinvent Arithmetic*. Constance Kamii and her sister Mieko Kamii from Wheelock College in Boston also collaborated on research on how children learned single digit and double digit addition, which formed part of the basis for Kamii's new work. The Kamiis said that children should not memorize "addition facts" such as $3 + 5 = 8$ or be taught to "carry" from the ones column to the tens column to the hundreds because this was not the way children naturally did addition. On their own, young children did single digit addition up to ten, two ways, either by "counting on" by starting at three and then saying "four, five, six, seven, eight," or by "counting all," counting up to three fingers and then going on to count five more, and then going back to count all 8 fingers, thus combining the group of three and the group of five they had just counted. For double digit addition for sums over 10, Kamii and her sister found that some children rounded up to ten first, as many modern arithmetic texts now recommend (Kamii, 1985, 68; Kamii, 2000, 84). However children approached addition problems, Kamii and her sister argued, the children came up with strategies on their own.

Teachers' reliance on worksheets was one of the stumbling blocks Kamii had to overcome. Georgia DeClark told Kamii that she had been teaching addition successfully to the children in her first grade class using traditional methods—memorization of "addition facts," "carrying," drills, and worksheets—and that this was the way the curriculum she had to cover was supposed to be taught. When Kamii visited DeClark's classroom she asked DeClark if she would be willing to try teaching arithmetic for a year using only activities from the children's daily life and games, no direct instruction, no worksheets, no school math series. DeClark said that she could not promise to make such a "drastic change," but that she would "give it a try" and see how far she could go. Kamii said that DeClark should rely on her own judgment, of course, and do what she thought was necessary if she did not think that Kamii's Piagetian methods were working. Kamii promised to visit DeClark's classroom every week and help her all the way (Kamii, 1985, xiii; DeClark, 1985, 195).

DeClark worried that her children would not learn the basic arithmetic they needed to know with Kamii's methods. DeClark was also worried about how to convince her principal, what she would say to other teachers, and what she would tell parents. DeClark's principal said she could go ahead as long as she reached the achievement goals set by the standard curriculum by the end of the year; the other teachers were busy worrying about their own classes. DeClark explained the new approach to the parents, a little more confidently than she actually felt, and told them to play games at home with their children. They did not challenge her. So at the beginning of the 1980–81 school year, DeClark started using the group games Kamii suggested: Tic Tac Toe, Concentration, Card Dominoes, War, Piggy Bank, Double War, Subtraction Lotto, Sorry, Double Parcheesi, and others. The children loved the games. They focused on them more intently than they had on worksheets and made decisions autonomously, just as Kamii had hoped.

DeClark was still worried, however. On October 29th she gave the children an addition worksheet. They did well on it, just as Kamii had told her they would. DeClark gave out four worksheets in all, and found to her relief that her children could do paper and pencil addition problems on worksheets just fine. Kamii told DeClark that she was probably the only first grade teacher in a public school in America who gave out only four worksheets that year (DeClark, 1985, 195–227).

When Kamii tested DeClark's children on single-digit arithmetic problems, she found that they did as well as a control group of children the same age in another first grade class who had studied arithmetic the traditional way. About the same number in both groups could do double-digit addition problems. DeClark's children had taught themselves arithmetic, by playing games, without lessons, worksheets, flash cards, or adults pushing them. They could explain how they got their answers. The children in the control group could not. Kamii and DeClark repeated the experiment again the next year with the same results (Kamii, 1985, 231–246).

Kamii felt vindicated. She had proved that first graders could reinvent arithmetic on their own. Now she wanted to see if second graders could do it, too. She needed two teachers, one each in first and second grade who were willing to use Piagetian methods. She could not stay at DeClark's school, however, because the principal said he reshuffled the students each year and would not keep DeClark's class together. When Kamii tried to find another principal she encountered resistance. Teachers from her graduate course were eager to try the new methods, but when Kamii talked to their principals, the principals asked one question: Can you promise good achievement test scores? Kamii explained her approach and offered to show her data. None of the principals looked at the data. When Kamii honestly said that she could not absolutely guarantee good test scores, all of the principals said "No." Some asked her if she knew that their jobs depended on getting good test scores. Not one principal in the Chicago area agreed to let Kamii try her arithmetic methods in his school (Kamii, 1989, vii).

Stymied, Kamii was determined to prove that the preschool arithmetic methods based on Piaget and play that she had developed would work with second graders. She was receptive when she met Milly Cowles, the Dean of the School of Education at the University of Alabama in Birmingham, who told Kamii that public schools in the South were much more open to university-based experimenters than public schools in the North. Frustrated in Chicago, Kamii visited Birmingham and moved there in January of 1984, so that she could continue her research. By September, she had a school, the Hall-Kent School in Homewood, in an integrated, moderate-income Birmingham suburb, a supportive school superintendent, Robert Bumpus, and an enthusiastic principal, Gene Burgess, who was so excited about Kamii's research that he wanted her to try it at all grade levels in his school. Burgess thought that the math program he was using was not working, knew about Piaget's work, and never asked Kamii about test scores. Kamii had never met a principal like this. Although the teachers were skeptical at first, Kamii visited their classes and met with them often. Eventually ten teachers signed on, four in kindergarten and three each in first grade and second grade (Kamii, 1989, vii–viii).

Kamii knew how different her approach was from the goals and methods of traditional math texts for second grade. The Harcourt, Brace, Jovanovich text that the Homewood teachers were using required that number facts, addition of whole numbers, subtraction of whole numbers, multiplication of whole numbers, division of whole numbers, fractions, measurement, time, money, geometry, graphing, probability, statistics, and problem solving be taught directly and incrementally, with children writing out correct answers. Kamii had to prove that second graders could learn these concepts and computational skills through constuctivist, play-based methods instead (Abbott & Wells, 1985, 26; Kamii, 1989, 3, 45, 54).

Rather than beginning with specific objectives, as traditional arithmetic programs did, Kamii derived her objectives from carefully observing the children, in the tradition of progressive preschool education going back to the nursery school movement of the 1920s, in which Piaget was imbued from his original work at the nursery school at the Institut Jean-Jacques Rousseau (Beatty, 2009). In her 1989 book *Young Children Continue to Reinvent Arithmetic, 2nd Grade*, written with teacher Linda Joseph, Kamii stated that she eventually arrived at five objectives: addition of one-digit numbers, place value and addition of two-digit numbers, subtraction of one- and two-digit numbers, multiplication, and division. Instead of formally teaching place value first as arithmetic texts recommend, Kamii and the teachers let the children learn it as they did addition (Kamii, 1989, 63).

From her observations, Kamii found that the traditional order of arithmetic teaching—addition, subtraction, multiplication, and division—was not how children reinvented it. Psychologists from the beginning of the twentieth century had been debating the order of arithmetic teaching. In 1911, G. Stanley Hall said that arithmetic learning, what he called "arithmogenesis," was biologically programmed into young children, and should be left to develop naturally, somewhat as Kamii argued (Beatty, 2006; Hall, 1911). In fact, like Hall, in a 1987 article in *Arithmetic Teacher*, Kamii said that children were "born with a natural ability to think and to construct logicomathematical knowledge" (Kamii, 1987). John Dewey argued that children learned arithmetic by constructing concepts through everyday activities, another approach Kamii used (Beatty, 2006; Dewey, 1896). Kamii found that subtraction was much harder for children than multiplication and argued that multiplication, not subtraction, should come after addition.

Like Piaget, Jerome Bruner, Eleanor Duckworth, and other progressives in the science and math curriculum reform movement for older children in the 1960s, Kamii thought that teachers should let children arrive at answers themselves, not correct children when they were wrong, and encourage children to discuss how they got their answers. As with first graders, Kamii suggested that Linda Joseph's second graders learn through games and everyday activities, with the addition of teacher-initiated discussions of computation and story problems. Joseph would put 18 + 13 on the board, ask the children what they thought was a good way to solve it, write their suggestions on the board, and listen to the children's reasons for agreeing or disagreeing with each other's answers. She would not tell them the right answer or correct wrong answers. When some children got the right answer, other children would agree or disagree, and later, sometimes four or seven months later, would reinvent double column addition on their own and be able to say why the right answer was right (Kamii, 1989, 75–79).

Teachers had to be frustrated with traditional methods to be willing to give Kamii's radical approach a try. At first, like Georgia DeClark, Linda Joseph was not convinced. When Kamii visited her classroom, she told Joseph that her children were "not thinking." Joseph had thought this herself sometimes and decided to try Kamii's approach. Joseph stuck with Kamii's Piagetian, play-based methods with the same group of children for four years. After surviving the first year without workbooks and seeing that the children were doing well on tests, Joseph was convinced that games and discussions were better than drill sheets. By the third and fourth year, Joseph was asking her students what they would like to work on, telling time or subtraction, or something else, and letting them choose. She had gone through a "metamorphosis" as a teacher, she said (Joseph, 1989, 151–156).

As in Chicago, Kamii was able to prove that her child-centered, constructivist approach worked, based on the results of standardized tests. When Kamii compared the performance on the Stanford Achievement Test of second graders at the Hall-Kent School, who had learned through her methods, to comparable second graders in another school who had not, she found that their standardized test scores were about the same, but when asked to explain their answers the

Hall-Kent children did much better. The mean Stanford Achievement Test Total Mathematics Score in percentiles for the Hall-Kent second graders was 79; the score for the children in the other school was 85 or above. But the other school enrolled children from higher socio-economic backgrounds, so the scores were comparable, Kamii argued. In contrast, when interviewed orally, the Hall-Kent children could explain the arithmetic they had invented and why; the other children could not. Kamii also made up a paper-and-pencil Math Sampler test of her own in which the children wrote out their answers and showed how they got them, instead of just filling in a blank. On this test, 48 percent of Hall-Kent second graders correctly solved an addition problem on four double-digit numbers adapted from the National Assessment of Educational Progress, the "gold standard" achievement test given to a randomized sample of American children, the exact percent of third graders who got the problem right on the national assessment (Kamii, 1989, 159, 169).

Satisfied that second graders could reinvent arithmetic as first graders did, Kamii moved on to third grade. In her 1994 *Young Children Continue to Reinvent Arithmetic, 3rd Grade*, which she wrote with the help of third-grade teacher Sally Jones Livingston from the Hall-Kent School, Kamii continued to stress the importance of Piagetian constructivist, play-based methods. Kamii included examples of more group games, and meticulous, detailed descriptions of children's own problem-solving techniques. As in her earlier books, when comparing classes taught by her methods versus traditional methods, Kamii found that the children who had been taught arithmetic for three years using her methods were "better in logical and numerical reasoning" and "better thinkers when they are encouraged to do their own thinking" (Kamii, 1994, 207).

In this third book, Kamii set out the most controversial of all of her research on how young children learn and should be taught arithmetic. After introductory chapters on Piaget's theory of logico-mathematical knowledge and on the history of computational techniques going back to the Hindus and Romans, she wrote about "The Harmful Effects of Algorithms." Teaching children algorithms, such as 18 + 17 = 35, actually hurt children's ability to learn arithmetic, Kamii argued, for three reasons. Algorithms forced children to "give up their own numerical thinking;" they "untaught" place value and hindered "children's development of numerical sense;" and they made children "dependent on the spatial arrangement of digits (or paper and pencil) and on other people" (Kamii, 1994, 33). For instance, in addition, subtraction, and multiplication, algorithms forced children to go from right to left, but Kamii observed that when children invented how to solve these types of problem on their own, they always, she said, went from left to right. In division, it was the opposite. With algorithms, Kamii said, children forgot how to use place value and often made illogical mistakes, because they added "all the digits as 1s" (Kamii, 1994, 36). And by using algorithms, children would sometimes avoid trying to solve a problem altogether because they felt dependent on their teachers, or on "paper and pencil" arithmetic (Kamii, 1994, 47).

Kamii's Impact

The impact of Constance Kamii's research on Piagetian theory and pedagogy, especially on teaching arithmetic, continues to be felt in preschool and primary education today She translated Piaget's abstruse ideas into practical activities for teachers, activities that preserved and extended the constructivism of Piaget's theory while remaining grounded in actual classroom application. Kamii was one of a handful of researchers who instantiated Piaget into preschool education, after his psychology had been rejected in academia. Her approach to teaching arithmetic was highlighted in the "bible of preschool education," Sue Bredekamp's ubiquitous 1987 *Developmentally Appropriate Practice in Early Childhood Programs Serving Children From Birth Through Age 8*. Kamii was also mentioned in Bredekamp and Carol Copple's revised 1997 edition, though not in the most recent 2009 edition, although it could be argued that by now many of Kamii's ideas have become so widely accepted that they no longer require specific citation (Bredekamp, 1987; Bredekamp and Copple, 1997; Copple and Bredekamp, 2009). Many of Kamii's books are still in print, sell well, and have been released in innumerable international editions.

Kamii's legacy in arithmetic teaching can still be felt in the primary grades, as well. An expanded version of her chapter on "The Harmful Effects of Algorithms" was reprinted in the National Council of Teachers of Mathematics *Yearbook* in 1998, where it provoked huge controversy (Kamii & Dominick, 1998). Many of Kamii's ideas about how to teach arithmetic through constructivist methods were published in journals of the National Council of Teachers of Mathematics, such as *Teaching Children Mathematics* and the *Journal of Research in Mathematics Education*, where one of her co-authored reports appeared as recently as 2010, giving Kamii's ideas wide currency (Kamii & Russell, 2010). Textbook designers adopted some of Kamii's methods, especially TERC, whose widely-used series *Investigations in Number, Data, and Space*, developed in the 1990s, incorporated much of Kamii's philosophy. In fact, *Investigations* and Kamii's ideas about the superiority of child-centered, constructivist arithmetic teaching were at the center of the "math wars" that raged in the 1990s and reverberate today.

In her 80s and still going strong, Kamii has an abiding faith in the power of Jean Piaget's psychology as the scientific basis of education. She has devoted her long life to promoting constructivist approaches to education for young children from preschool through the primary grades and still wants to find a 4th grade class in which to do more research on her Piagetian approach to teaching arithmetic. She told me that she would also like to get back in touch with Siegfried Engelmann and retest some of the students taught via his Direct Instruction to prove once and for all that she and Piaget are right about how children learn. It is hard to imagine modern early childhood education without the games, math manipulatives, and other child-centered methods that Constance Kamii encouraged preschool teachers to use. A giant in the debate over play that still rages today, Kamii remains firmly convinced that young children should be given the opportunity to learn autonomously.

Bibliography

Abbott, J. S., & Wells, D. W. (1985). *Mathematics today* (Teacher's Ed., Level 2). Orlando: Harcourt Brace Jovanovich.

Beatty, B. (2006). Psychologizing the third *R*: Hall, Dewey, Thorndike, and Progressive-Era ideas on the learning and teaching of arithmetic. In B. Beatty, E. D. Cahan, & J. Grant (Eds.) *When science encounters the child: Education, parenting, and child welfare in 20th-century America* (pp. 35–55). New York: Teachers College Press.

Beatty, B. (2009). Transitory connections: The reception and rejection of Jean Piaget's psychology in the Nursery School Movement in the 1920s and 1930s. *History of Education Quarterly, 49*, May: 442–464.

Beatty, B. (2012). The debate over the young "disadvantaged child": Race, class, culture, language, developmental psychology, and preschool intervention. *Teachers College Record*. June: 1–36.

Bereiter, C., & Englemann, S. (1966). *Teaching disadvantaged children in the preschool*. Englewood Cliffs, NJ: Prentice-Hall.

Berrueta-Clement, J. R., Schweinhart, L. J., Barnett, W. S., Epstein, A. S., & Weikart, D. P. (1984). *Changed lives: The effects of the Perry Preschool Program on youths through age 19*. Ypsilanti, MI: High/Scope Press.

Bredekamp, S. (Ed.). (1987). *Developmentally appropriate practice in early childhood programs serving children from birth through age 8*. Washington, DC: National Association for the Education of Young Children.

Bredekamp, S. & Copple, C. (Eds.). (1997). *Developmentally appropriate practice in early childhood programs* (Revised edn.). Washington, DC: National Association for the Education of Young Children.

Bredekamp, S. & Copple, C. (Eds.). (2009). *Developmentally appropriate practice in early childhood programs* (3rd edn.). Washington, DC: National Association for the Education of Young Children.

Cohen, P. C. (1982). *A calculating people: The spread of numeracy in Early America*. Chicago: University of Chicago Press.

Clifford, G. J. (1984). *Edward L. Thorndike: The sane positivist*. Middletown: Wesleyan University Press.

Deutsch, M. (1967). *The disadvantaged child: Selected papers of Martin Deutsch and Associates*. New York: Basic Books.

DeVries, R. & Kohlberg, L. (1987). *Constructivist early education*. Washington, DC: National Association for the Education of Young Children.

Duckworth, E. (1964). Piaget rediscovered. *Journal of Research in Science Teaching*. September, 172–175.

Elkind, D. (1968, May 26). Giant in the nursery – Jean Piaget. *The New York Times Sunday Magazine*.

Engelmann, S. E. (1971). Does the Piagetian approach imply instruction? In D. R. Green, M. P. Ford, & G. B. Flamer (Eds.), *Measurement and Piaget* (pp. 118–126). New York: McGraw-Hill Book Company.

Engelmann, S. E. (1971). Open discussion. In D. R. Green, M. P. Ford, & G. B. Flamer, (Eds.), *Measurement and Piaget* (p. 147). New York: McGraw-Hill Book Company.

Finkelstein, B. (1989). *Governing the young: Teacher behavior in popular primary schools in 19th Century United States*. New York: Falmer.

Flavell, J. H. (1963). *The developmental psychology of Jean Piaget*. Princeton, NJ: Van Nostrand.

Gray, S. W. & Klaus, R. A. (1965). An experimental preschool program for culturally deprived children. *Child Development, 36*, 887–898.

Green, D. R., Ford, M. P., & Flamer, G. B. (Eds.). (1971). *Measurement and Piaget*. New York: McGraw-Hill Book Company.

Hall, G. S. (1911). *Educational problems*. New York: D. Appleton.

Hunt, J. M. (1961). *Intelligence and experience*. New York: The Ronald Press Company.

Hsueh, Y. (1997). *Jean Piaget, spontaneous development and constructivist teaching*. (Unpublished doctoral dissertation), Harvard Graduate School of Education.

Joseph, L. (1989). Metamorphosis. In C. Kamii, & L. L. Joseph, *Young children continue to reinvent arithmetic, 2nd grade* (pp. 151–156). New York: Teachers College Press.

Kamii, C. (1972a). An application of Piaget's theory to the conceptualization of a preschool curriculum. In R. K. Parker (Ed.), *The preschool in action: Exploring early childhood programs* (pp. 91, 127). Boston: Allyn & Bacon.

Kamii, C. (1972b). A sketch of the Piaget-derived preschool curriculum developed by the Ypsilanti Early Education Program. In S. J. Braun & E. Edwards (Eds.), *History and theory of early childhood education* (p. 295). Worthington, OH: Charles A. Jones.

Kamii, C. (1973). Pedagogical principles derived from Piaget's theory: Relevance for educational practice. In M. Schwebel & J. Raph (Eds.), *Piaget in the classroom* (pp.199–201). New York: Basic Books.

Kamii, C. (1980). *Group games in early education: Implications of Piaget's theory* (Appendix). Washington, DC: National Association for the Education of Young Children.

Kamii, C. (1981, October). Appendix, autonomy as the aim of education: implications of Piaget's theory. Keynote address presented at the annual conferences of the North Carolina Association for the Education of Young Children, Winston-Salem, NC and DuPage Regional Unit of the Chicago Association for the Education of Young Children, Glen Ellyn, IL.

Kamii, C. (1982). *Number in preschool and kindergarten*. Washington, DC: National Association for the Education of Young Children.

Kamii, C. (1987). Arithmetic: Children's thinking or their writing of correct answers? *Arithmetic Teacher, 35*, 2.

Kamii, C., personal communication, March 13/14, 2008.

Kamii, C., personal communication, March 21, 2012.

Kamii, C., with DeClark, G. (1985). *Young children reinvent arithmetic: Implications of Piaget's theory*. New York: Teachers College Press.

Kamii, C. & Derman, L. (1971). Comments on Engelmann's paper. *Measurement and Piaget*. New York: McGraw-Hill, 142, 143, 145, 146.

Kamii, C. & DeVries, R. (1976). *Piaget, children, and number*. Washington, DC: National Association for the Education of Young Children.

Kamii, C. & DeVries, R. (1980). *Group games in early education: Implications of Piaget's theory*. Washington, DC: National Association for the Education of Young Children.

Kamii, C. & DeVries, R. (1983). *Physical knowledge in preschool education: Implications of Piaget's theory* (2nd edn.). New York: Teachers College Press.

Kamii, C. & Dominick, A. (1998). The harmful effects of algorithms in grades 1–4. In L. J. Morrow & M.J. Kenney (Eds.), *The teaching and learning of algorithms in school mathematics: 1998 NCTM Yearbook* (pp. 130–140). Reston, VA: National Council of Teachers of Mathematics.

Kamii, C. & Housman, L. B. (2000). *Young children reinvent arithmetic* (2nd edn.). New York: Teachers College Press.

Kamii, C., with Joseph, L. L. (1989). *Young children continue to reinvent arithmetic, 2nd grade.* New York: Teachers College Press.

Kamii, C., with Livingston, S. J. (1994). *Young children continue to reinvent arithmetic, 3rd grade.* New York: Teachers College Press.

Kamii, C. & Russell, K. A. (2010). The older of two trees: Young children's development of operational time. *Journal for Research in Mathematics Education, 41*, 6–13.

Kamii, C. & Weikart, D. P. (1963). Marks, achievement, and intelligence of seventh graders who were retained (nonpromoted) once in elementary school. *Journal of Educational Research, 56*, 452–459.

Lascarides, V. C. & Hinitz, B. F. (2000). *History of early childhood education.* New York: Falmer.

Minkovitch, A. (1972). Early education in Israel. In S. J. Braun & E. Edwards (Eds.), *History and theory of early childhood education* (pp.132–145). Worthington, OH: Charles A. Jones Publishing Company.

Monroe, W. S. (1917). *Development of arithmetic as a school subject.* Bureau of Education Publication, 10.

Piaget, J. (1971). The theory of stages of cognitive development. In D. R. Green, M. P. Ford, & G. B. Flamer (Eds.), *Measurement and Piaget* (pp. 1–11). New York: McGraw-Hill Book Company.

Piaget, J. (1973). *To understand is to invent* (originally published in 1948). New York: Viking.

Radin, N. L. & Kamii, C. (1965). The child rearing attitudes of disadvantaged Negro mothers and some educational implications. *The Journal of Negro Education, 34*, 138–146.

Radin, N. L. & Kamii, C. (1967). A framework for a preschool curriculum based on some Piagetian concepts, *Journal of Creative Behavior, 1*, 314–324.

Radin, N. L. & Kamii, C. (1967). Class differences in the socialization patterns of negro mothers. *Journal of Marriage and the Family, 29*, 302–310.

Schweinhart, L. J., Montie, J., Xiang., Barnett, W. S., Belfield, C. R., & Nores, M. (2005). *The High/Scope Perry Preschool Study through age 40.* Ypsilanti, MI: High/Scope Press.

Schweinhart, L. J., Barnes, H. V., & Weikart, D. P. (1993). *Significant benefits: The High/Scope Perry Preschool Study through age 27.* Ypsilanti, MI: High/Scope Press.

Sinclair, H. & Kamii, C. (1970). Some implications of Piaget's theory for teaching young children. *The School Review, 78*, 169–183.

Smilansky, S. (1968). *The effects of sociodramatic play on disadvantaged preschool children.* New York: Wiley.

Sonquist, H. D. & Kamii, C. (1967). Applying some Piagetian concepts in the classroom for the disadvantaged. *Young Children, 22*, 232–246.

Thorndike, E. L. (1917). *The Thorndike arithmetics.* Chicago: Rand-McNally.

Thorndike, E. L. (1922). *The psychology of arithmetic.* New York: Macmillan.

Weikart, D., *et al.* (1971). *The cognitively oriented curriculum: A framework for preschool teachers.* Washington, DC: National Association for the Education of Young Children.

Weikart, D. P. (2004). *How High/Scope grew: A memoir.* Ypsilanti, MI: High/Scope Press.

Kamii, C. & Housman, L. B. (2000). *Young children reinvent arithmetic* (2nd edn.). New York: Teachers College Press.

Kamii, C., with Joseph, L. L. (1989). *Young children continue to reinvent arithmetic, 2nd grade.* New York: Teachers College Press.

Kamii, C., with Livingston, S. J. (1994). *Young children continue to reinvent arithmetic, 3rd grade.* New York: Teachers College Press.

Kamii, C. & Russell, K. A. (2010). The older of two trees: Young children's development of operational time. *Journal for Research in Mathematics Education, 41*, 6–13.

Kamii, C. & Weikart, D. P. (1963). Marks, achievement, and intelligence of seventh graders who were retained (nonpromoted) once in elementary school. *Journal of Educational Research, 56*, 452–459.

Lascarides, V. C. & Hinitz, B. F. (2000). *History of early childhood education.* New York: Falmer.

Minkovitch, A. (1972). Early education in Israel. In S. J. Braun & E. Edwards (Eds.), *History and theory of early childhood education* (pp.132–145). Worthington, OH: Charles A. Jones Publishing Company.

Monroe, W. S. (1917). *Development of arithmetic as a school subject.* Bureau of Education Publication, 10.

Piaget, J. (1971). The theory of stages of cognitive development. In D. R. Green, M. P. Ford, & G. B. Flamer (Eds.), *Measurement and Piaget* (pp. 1–11). New York: McGraw-Hill Book Company.

Piaget, J. (1973). *To understand is to invent* (originally published in 1948). New York: Viking.

Radin, N. L. & Kamii, C. (1965). The child rearing attitudes of disadvantaged Negro mothers and some educational implications. *The Journal of Negro Education, 34*, 138–146.

Radin, N. L. & Kamii, C. (1967). A framework for a preschool curriculum based on some Piagetian concepts. *Journal of Creative Behavior, 1*, 314–324.

Radin, N. L. & Kamii, C. (1967). Class differences in the socialization patterns of negro mothers. *Journal of Marriage and the Family, 29*, 302–310.

Schweinhart, L. J., Montie, J., Xiang., Barnett, W. S., Belfield, C. R., & Nores, M. (2005). *The High/Scope Perry Preschool Study through age 40.* Ypsilanti, MI: High/Scope Press.

Schweinhart, L. J., Barnes, H. V., & Weikart, D. P. (1993). *Significant benefits: The High/Scope Perry Preschool Study through age 27.* Ypsilanti, MI: High/Scope Press.

Sinclair, H. & Kamii, C. (1970). Some implications of Piaget's theory for teaching young children. *The School Review, 78*, 169–183.

Smilansky, S. (1968). *The effects of sociodramatic play on disadvantaged preschool children.* New York: Wiley.

Sonquist, H. D. & Kamii, C. (1967). Applying some Piagetian concepts in the classroom for the disadvantaged. *Young Children, 22*, 232–246.

Thorndike, E. L. (1917). *The Thorndike arithmetics.* Chicago: Rand-McNally.

Thorndike, E. L. (1922). *The psychology of arithmetic.* New York: Macmillan.

Weikart, D., *et al.* (1971). *The cognitively oriented curriculum: A framework for preschool teachers.* Washington, DC: National Association for the Education of Young Children.

Weikart, D. P. (2004). *How High/Scope grew: A memoir.* Ypsilanti, MI: High/Scope Press.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
A.I. LABORATORY

July 1971

Artificial Intelligence
Memo No. 249

LOGO
Memo No. 4

TEACHING CHILDREN TO BE MATHEMATICIANS VS. TEACHING ABOUT MATHEMATICS[1]

Seymour Papert

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's education research is provided in part by the National Science Foundation under grant GJ-1049.

[1]To be published in the International Journal of Mathematical Education in Science and Technology (New York: John Wiley & Sons, 1972) and in Proceedings of 1970 CEMREL Conference on Algebra (Carbondale, Illinois: CEMREL, Spring, 1971).

TEACHING CHILDREN TO BE MATHEMATICIANS VS. TEACHING ABOUT MATHEMATICS

by

Seymour Papert

1. Preface

Being a mathematician is no more definable as "knowing" a set of mathematical facts than being a poet is definable as knowing a set of linguistic facts. Some modern math ed reformers will give this statement a too easy assent with the comment: "Yes, they must understand, not merely know." But this misses the capital point that being a mathematician, again like being a poet, or a composer or an engineer, means doing, rather than knowing or understanding. This essay is an attempt to explore some ways in which one might be able to put children in a better position to do mathematics rather than merely to learn about it.

The plan of the essay is to develop some examples of new kinds of mathematical activity for children, and then to discuss the general issues alluded to in the preceding paragraph. Without the examples, abstract statements about "doing," "knowing," and "understanding" mathematics cannot be expected to have more than a suggestive meaning. On the other hand the description of the examples will be easier to follow if the reader has a prior idea of their intention. And so I shall first sketch, very impressionistically, my position on some of the major issues. In doing so I shall exploit the dialectical device employed in the previous paragraph to obtain a little more precision of statement by explicitly excluding the most likely misinterpretation.

It is generally assumed in our society that every child should, and can, have experience of creative work in language and plastic arts.

It is equally generally assumed that very few people can work creatively in mathematics. I believe that there has been an unwitting conspiracy of psychologists and mathematicians in maintaining this assumption. The psychologists contribute to it out of genuine ignorance of what creative mathematical work might be like. The mathematicians, very often, do so out of elitism, in the form of a deep conviction that mathematical creativity is the privilege of a tiny minority.

Here again, it is necessary, if we want any clarity, to ward off a too easy, superficial assent from math ed reformers who say, "Yes, that's why we must use The Method of Discovery." For, when "Discovery" means discovery this is wonderful, but in reality "Discovery" usually means something akin to the following fantasy about a poetry class: the discovery-method teacher has perfected a series of questions that lead the class to discover the line "Mary had a little lamb." My point is not that this would be good or bad, but that no one would confuse it with creative work in poetry.

Is it possible for children to do creative mathematics (that is to say: to do mathematics) at all stages of their scholastic (and even adult!) lives? I will argue that the answer is: yes, but a great deal of creative mathematical work by adult mathematicians is necessary to make it possible. The reason for the qualification is that the traditional branches of mathematics do not provide the most fertile ground for the easy, prolific growth of mathematical traits of mind. We may have to develop quite new branches of mathematics with the special property that they allow beginners more space to romp creatively, than does number theory or modernistic algebra. In the following pages will be found some specific examples which it would be pretentious to call "new pedagogical oriented branches of mathematics" but which will suggest to cooperative readers what this phrase could mean.

Obstreperous readers will have no trouble finding objections. Mathematical elitists will say: "How dare you bring these trivia to disturb our contemplation of the true mathematical structures." Practical people will say: "Romping? Pomping? Who needs it? What about practical skills in arithmetic?"

The snob and the anti-snob are expressing the same objection in different words. Let me paraphrase it, "Traditional schools have found mathematics hard to teach to so-called average children. Someone brings along a new set of activities, which seem to be fun and easy to learn. He declares them to be mathematics! Well, that does not make them mathematics, and it doesn't turn them into solutions to any of the hard problems facing the world of math ed."

This argument raises serious issues, from which I single out a question which I shall ask in a number of different forms:

> In becoming a mathematician does one learn something other and more general than the specific content of particular mathematical topics? Is there such a thing as a Mathematical Way of Thinking? Can this be learned and taught? Once one has acquired it, does it then become quite easy to learn <u>particular</u> topics -- like the ones that <u>obsess our</u> elitist and practical critics?

Psychologists sometimes react by saying, "Oh, you mean the transfer problem." But I do not mean anything analogous to experiments on whether students who were taught algebra last year <u>automatically</u> learn geometry more easily than students who spent last year doing gymnastics. I am asking whether one can identify and teach (or foster the growth of) something <u>other</u> than algebra or geometry, which, once learned, will make it easy to learn algebra and geometry. No doubt, this other thing (let's call it the MWOT) can only be taught by using particular topics as vehicles. But the "transfer" experiment is profoundly changed if the question is whether one can <u>use</u> algebra as a <u>vehicle for deliberately teaching transferable general concepts and skills</u>. The conjecture underlying this essay is a very qualified affirmative answer to this question. Yes, one <u>can</u> use algebra as a vehicle for initiating students to the mathematical way of thinking. But, to do so effectively one should first identify as far as possible components of the general intellectual skills one is trying to teach; and when this is done it will appear that algebra (in any traditional sense) is not a particularly good vehicle.

The alternative choices of vehicle described below all involve using computers, but in a way that is very different from the usual suggestions of using them either as "teaching machines" or as "super-slide-rules". In our ideal of a school mathematical laboratory the computer is used as a means to control physical processes in order to achieve definite goals . . . for example as part of an auto-pilot system to fly model airplanes, or as the "nervous system" of a model animal with balancing reflexes, walking ability, simple visual ability and so on. To achieve these goals mathematical principles are needed; conversely in this context mathematical principles become sources of power, thereby acquiring meaning for large categories of students who fail to see any point or pleasure in bookish math and who, under prevailing school conditions, simply drop out by labelling themselves "not mathematically minded."

The too easy acceptance of this takes the form: "Yes, applications are motivating." But "motivation" fails to distinguish alienated work for a material or social reward from a true personal involvement. To develop this point I need to separate a number of aspects of the way the child relates to his work.

A simple, and important one, is the time scale. A child interested in flying model airplanes under computer control will work at this project over a long period. He will have time to try different approaches to sub-problems. He will have time to talk about it, to establish a common language with a collaborator or an instructor, to relate it to other interests and problems. This project-oriented approach contrasts with the problem approach of most math teaching: a bad feature of the typical problem is that the child does not stay with it long enough to benefit much from success or from failure.

Along with time scale goes structure. A project is long enough to have recognizable phases -- such as planning, choosing a strategy of attempting a very simple case first, finding the simple solution, debugging it, and so on. And if the time scale is long enough, and the structures clear enough, the child can develop a vocabulary for articulate discussion of the process of working towards his goals.

I believe in articulate discussion (in monologue or dialogue) of how one solves problems, of why one goofed that one, of what gaps or deformations exist in one's knowledge and of what could be done about it. I shall defend this belief against two quite distinct objections. One objection says: "it's impossible to verbalize; problems are solved by intuitive acts of insight and these cannot be articulated." The other objection says: "it's bad to verbalize; remember the centipede who was paralyzed when the toad asked which leg came after which."

One must beware of quantifier mistakes when discussing these objections. For example, J.S. Bruner tells us (in his book Towards a Theory of Instruction) that he finds words and diagrams "impotent" in getting a child to ride a bicycle. But while his evidence shows (at best) that some words and diagrams are impotent, he suggests the conclusion that all words and diagrams are impotent. The interesting conjecture is this: the impotence of words and diagrams used by Bruner is explicable by Bruner's cultural origins; the vocabulary and conceptual framework of classical psychology is simply inadequate for the description of such dynamic processes as riding a bicycle! To push the rhetoric further, I suspect that if Bruner tried to write a program to make an IBM 360 drive a radio controlled motorcycle, he would have to conclude (for the sake of consistency) that the order code of the 360 was impotent for this task. Now, in our laboratory we have studied how people balance bicycles and more complicated devices such as unicycles and circus balls. There is nothing complex or mysterious or undescribable about these processes. We can describe them in a non-impotent way provided that a suitable descriptive system has been set up in advance. Key components of the descriptive system rest on concepts like: the idea of a "first order" or "linear" theory in which control variables can be assumed to act independently; or the idea of feedback.

A fundamental problem for the theory of mathematical education is to identify and name the concepts needed to enable the beginner to discuss his mathematical thinking in a clear articulate way. And when we know such concepts we may want to seek out (or invent!) areas of mathematical work which exemplify these concepts particularly well. The next section of this essay will describe a new piece of mathematics with the property that it allows clear discussion and simple models of heuristics that are foggy and confusing for beginners when presented in the context of more traditional elementary mathematics.

2. Turtle Goemetry: A Piece of Learnable and Lovable Mathematics

The physical context for the following discussion is a quintuple consisting of a child, a teletype machine, a computer, a large flat surface and an apparatus called a turtle. A turtle is a cybernetic toy capable of moving forward or back in a particular direction (relative to itself) and of rotating about its central axis. It has a pen, which can be in two states called PENUP and PENDOWN. The turtle is made to act by typing commands whose effect is illustrated in Figure 1.

Figure 1: TURTLE LANGUAGE

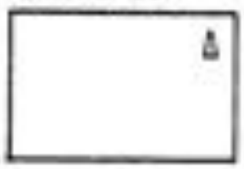
At any time the turtle is at a particular place and facing in a particular direction. The place and direction together are the turtle's geometric state. The picture shows the turtle in a field, used here only to give the reader a frame of reference:

(1)

The triangular picture shows the direction.

FORWARD 50

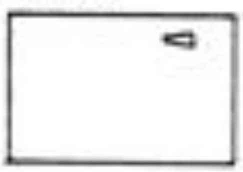
(2)



The turtle advanced 50 units in the direction it was facing.

LEFT 90

(3)



The turtle's position remained fixed. It rotated 90° to the left. So its direction changed.

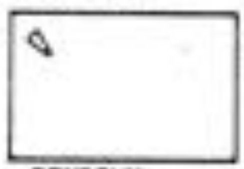
FORWARD 150

(4)

The turtle advanced 150 units in its new direction.

LEFT 135

(5)



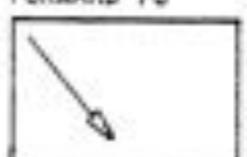
The turtle rotated left 135°.

PENDOWN

(Produces no visible effect. But the next FORWARD instruction will leave a trace.)

FORWARD 70

(6)



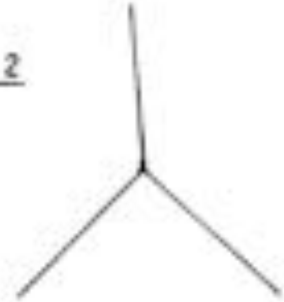
The effect of PENDOWN is to put the turtle in a state to leave a trace: the pen draws on the ground.

(a) Direct Commands

The following commands will cause the turtle to draw Figure 2.

```
PENDOWN
FORWARD 100
RIGHT 60
FORWARD 100
BACK 100
LEFT 120
FORWARD 100
```

Figure 2



PEACE

(b) Defining a procedure

The computer is assumed to accept the language LOGO (which we have developed expressly for the purpose of teaching children, not programming but mathematics). The LOGO idiom for asserting the fact that we are about to define a procedure is illustrated by the following example. We first decide on a name for the procedure. Suppose we choose "PEACE". Then we type:

```
TO PEACE
1  FORWARD 100
2  RIGHT 60
3  FORWARD 100
4  BACK 100
5  LEFT 120
6  FORWARD 100
END
```

These are directions telling the computer how to PEACE. The word "TO" informs the computer that the next word, "PEACE", is being defined and that the numbered lines constitute its definition.

The turtle doesn't move while we are typing this. The word "TO" and the line numbers indicated that we were not telling it to go forward and so on; rather we were telling it how to execute the new command. When we have indicated by the word "END" that our definition is complete the machine echoes back:

PEACE DEFINED

and now if we type

```
PENDOWN
PEACE
```

the turtle will carry out the commands and draw Figure 2. Were we to omit the command "PENDOWN" it would go through the motions of drawing it without leaving a visible trace.

The peace sign in Figure 2 lacks a circle. How can we describe a circle in turtle language?

An idea that easily presents itself to mathematicians is: let the turtle take a tiny step forward, then turn a tiny amount and keep doing this. Thi might not quite produce a circle, but it is a good first plan, so let's begin to work on it. So we define a procedure:

```
TO CIRCUS
1  FORWARD 5
2  RIGHT 7
3  CIRCUS
END
```

Notice two features

(a) The procedure refers to itself in line 3. This looks circular (though not in the sense we require) but really is not. The effect is merely to set up a never-ending process by getting the computer into the tight spot <u>you</u> would be in if you were the kind of person who cannot fail to keep a promise and you had been tricked into saying, "I promise to repeat the sentence I just said."

(b) We selected the numbers 5 and 7 because they seemed small, but without a firm idea of what would happen. However an advantage of having a computer is that we can try our procedure to see what it does. If an undesirable effect follows we can always <u>debug</u> it; in this case, perhaps, by choosing different numbers. If, for example, the turtle drew something like Figure 3a, we would say to ourselve , "It's not turning enough" and replace 7 by 8; on the other hand if it drew Figure 3b we might replace 7 by 6.

Figure 3a

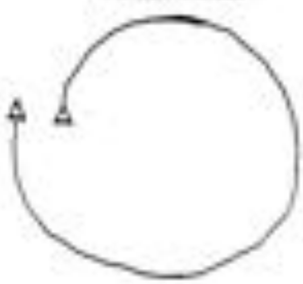


Figure 3b

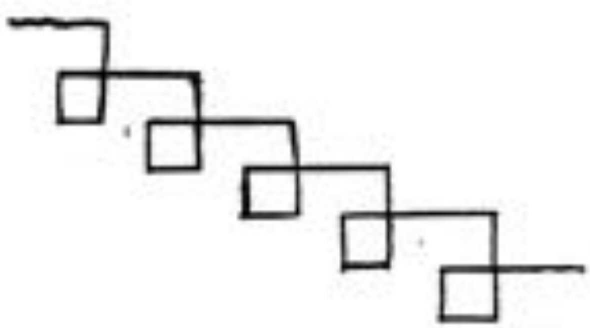
I wish I could collect statistics about how many mathematically sophisticated readers fell into my trap! Experience shows that a large proportion of math graduate students will do so. In fact, the procedure cannot generate either 3a or 3b! If it did, it would surely go on to produce an infinite spiral. And one can easily see that this is impossible since the same sequence of commands would have to produce parts of the curve that are almost flat, and other parts that are very curved. More technically, one can see that the procedure CIRCUS must produce a close approximation to a circle (i.e. what is, for all practical purposes a circle) because it must produce a curve of constant curvature.

One can come to the same conclusion from a more general theorem. We call procedures like CIRCUS "fixed instruction procedures" because they contain no variables.

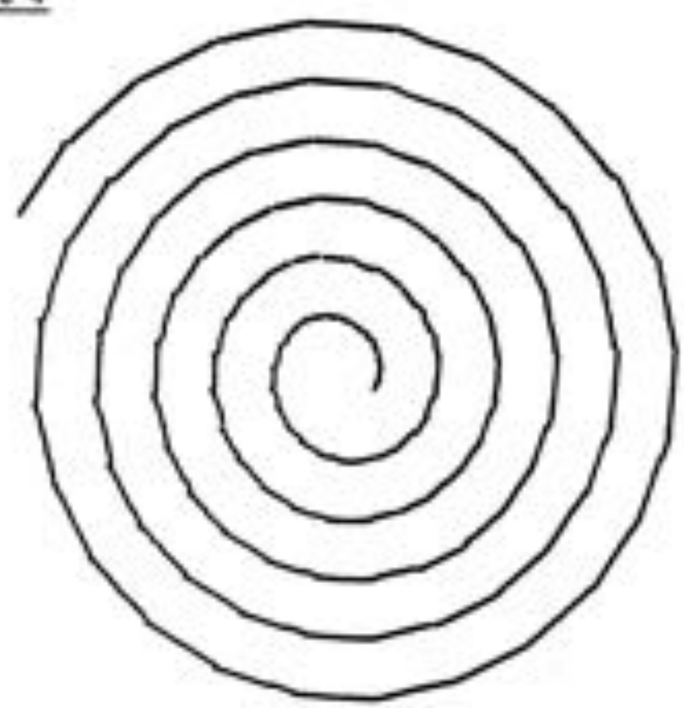
THEOREM: Any figure generated by a fixed instruction procedure can be bounded either by a circle or by two parallel straight lines.

Examples of figures that can and that cannot be so bounded are shown in Figure 4.

Figure 4



A Figure Bounded by parallel lines



A Figure Bounded neither by parallel lines nor by a circle.

We now show how to make procedures with inputs in the sense that the command FORWARD has a number, called an input, associated with it. The next example shows how we do so. (The words on the title line preceded by ":" are names of the inputs, rather like the x's in school algebra.) In the fifth grade class we read <u>:NUMBER</u> as <u>dots NUMBER</u> or as <u>the thing of "NUMBER"</u>, emphasizing that what is being discussed is not the word "NUMBER" but a thing of which this word is the name.

```
TO POLY :STEP :ANGLE
1  FORWARD :STEP
2  LEFT :ANGLE
3  POLY :STEP :ANGLE
END
```

This procedure generates a rather wonderful collection of pictures as we give it different inputs.

Although POLY has provision for inputs it is really a fixed instruction procedure. To create one that is not, we change the last line of POLY. We change the title also, though we do not need to do so.

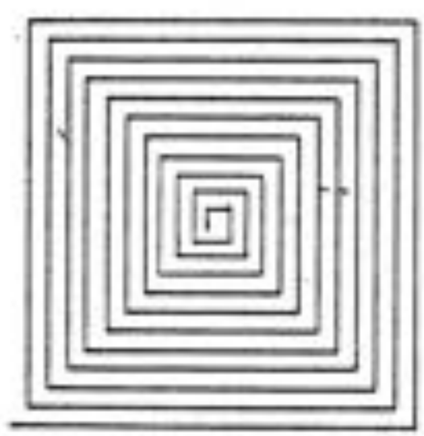
<u>Old Procedure</u>

```
TO POLY :STEP :ANGLE
1  FORWARD :STEP
2  LEFT :ANGLE
3  POLY :STEP :ANGLE
END
```

<u>New Procedure</u>

```
TO POLYSPI :STEP :ANGLE
1  FORWARD :STEP
2  LEFT :ANGLE
3  POLYSPI :STEP+20 :ANGLE
END
```

The effect of POLYSPI is shown in Figure 5.



<u>Figure 5</u>

POLYSPI 5 90

or

Squiral

We have seen we can use POLY to draw a circle. Can we now use it to draw our peace sign? We could, but will do better to make a procedure, here called ARC whose effect will be to draw any circular segment given the diameter and the angle to be drawn as in Figure 6. The procedure is as follows where in line 2 a special constant called "PIE" is used and the asterisk sign is used for multiplication. (Do not assume that :PIE is what its name suggests.)

```
TO ARC :DIAM :SECTOR
1  IF :SECTOR=0 STOP
2  FORWARD :PIE*:DIAM
3  RIGHT 1
4  ARC :DIAM :SECTOR-1
END
```

We can now make a procedure using the old procedure PEACE as a <u>sub-procedure</u>:

```
TO SUPERPEACE
1  ARC 200 360
2  RIGHT 90
3  PEACE
END
```

SUPERPEACE

Figure 6

Better yet we could rewrite PEACE to have inputs. For example:

```
TO PEACE :SIZE
1  FORWARD :SIZE
2  RIGHT 60
3  FORWARD :SIZE
4  BACK :SIZE
5  LEFT 120
6  FORWARD :SIZE
7  RIGHT 90
8  ARC 2*:SIZE 360
```

Then peace signs of different sizes can be made by the commands:

PEACE 100

PEACE 20

and so on.

We can use the command ARC to draw a heart:

```
TO HEART :SIZE
1  ARC :SIZE/2 180
2  RIGHT 180
3  ARC :SIZE/2 180
4  ARC :SIZE*2 60
5  RIGHT 60
6  ARC :SIZE*2 60
END
```

MINITHEOREM: A heart can be made of four circular arcs.

We can also use it to draw a flower. Notice in the following the characteristic building of new definitions on old ones.

A computer program to draw this flower uses the geometric observation that petals can be decomposed (rather surprisingly!) as two quarter circles. So let's assume we have a procedure called TO QCIRCLE whose effect is shown by the examples. Some of them show initial and final positions of the turtle, some do not.

QCIRCLE 50

QCIRCLE 100

Now let's see how to make a petal.

```
TO PETAL :SIZE
1  QCIRCLE :SIZE
2  RIGHT 90
3  QCIRCLE :SIZE
END
```

PETAL 100

```
TO FLOWER :SIZE
1  PETAL :SIZE
2  PETAL :SIZE
3  PETAL :SIZE
4  PETAL :SIZE
END
```

FLOWER 100

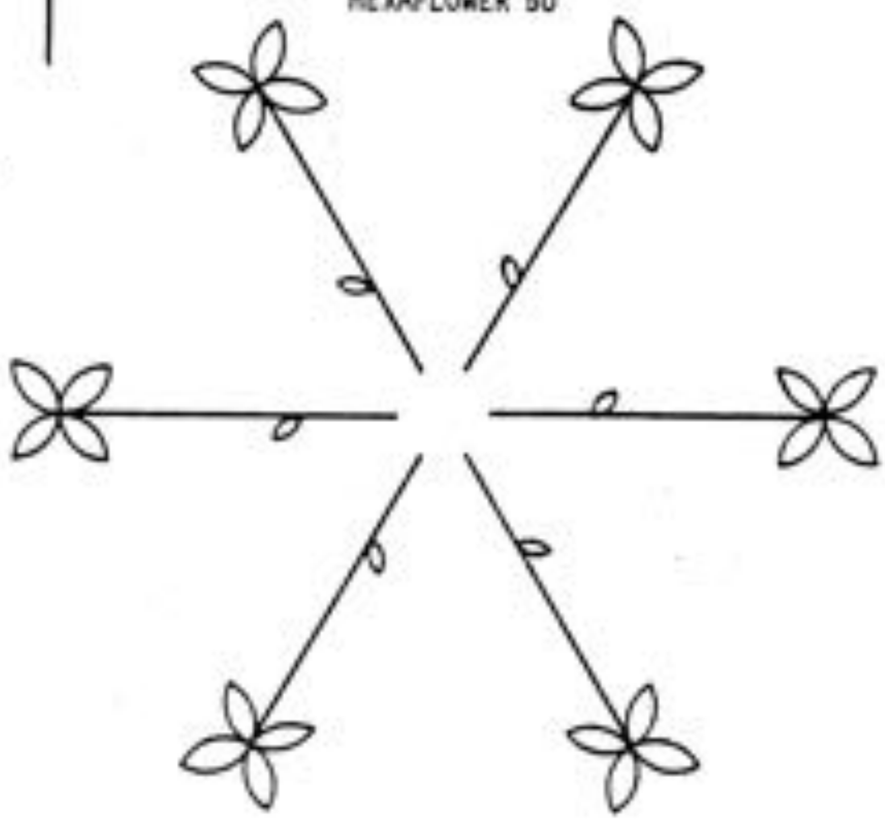
STEM 100

```
TO STEM :SIZE
1  RIGHT 180
2  FORWARD 2*:SIZE
3  RIGHT 90
4  PETAL :SIZE/2
5  FORWARD :SIZE
END
```

PLANT 50

```
TO PLANT :SIZE
1  PENDOWN
2  FLOWER :SIZE
3  STEM :SIZE
4  PENUP
END
```

HEXAFLOWER 50



Now let's play a little.

```
TO HEXAFLOWER :SIZE
1  RIGHT 90
2  FORWARD 4*:SIZE
3  PLANT :SIZE
4  FORWARD :SIZE
5  RIGHT 30
6  HEXAFLOWER :SIZE
END
```

3. Creativity? Mathematics?

In classes run by members of the M.I.T. Artificial Intelligence Laboratory we have taught this kind of geometry to fifth graders, some of whom were in the lowest categories of performance in "mathematics". Their attitude towards mathematics as normally taught was well expressed by a fifth grade girl who said firmly, "There ain't nothing fun in math!" She did not classify working with the computer as math, and we saw no reason to disabuse her. There will be time for her to discover that what she is learning to do in an exciting and personal way will elucidate those strange rituals she meets in the math class.

Typical activities in early stages of work with children of this age is exploring the behavior of the procedure POLY by giving it different inputs. There is inevitable challenge -- and competition -- in producing beautiful or spectacular, or just different effects. One gets ahead in the game by discovering a new phenomenon and by finding out what classes of angles will produce it.

The real excitement comes when one becomes courageous enough to change the procedure itself. For example making the change to POLYSPI occurs to some children and, in our class, led to a great deal of excitement around the truly spontaneous discovery of the figure now called a squiral (Figure 5). (Note: By spontaneous I mean, amongst other things, to exclude the situation of the discovery teacher standing in front of the class soliciting pseudo-randomly generated suggestions. The squiral was found by a child sitting all alone at his computer terminal!) By no means all the children will take this step -- indeed once a few have done so it becomes derivative for the others. Nevertheless, we might encourage them to explore inputs to POLYSPI. There is room here for the discovery of more phenomena. For example, taking :ANGLE as 120 produces a neat triangular spiral. But 123 produces a very different phenomena.

Figure 7

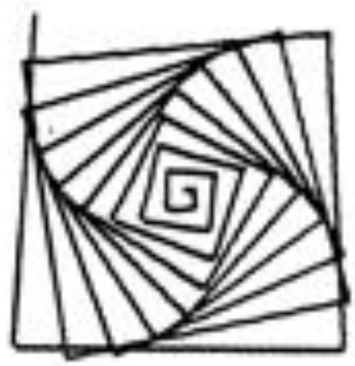
POLYSPI 5 120

POLYSPI 5 123

What else produces similar effects?

Figure 8

POLYSPI 5 121



POLYSPI 5 93

The possibilities for original minor discoveries are great. One girl became excited for the first time about mathematics by realizing how easy it was to make a program for Figure 9 by

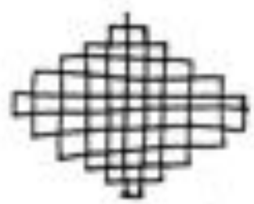
(1) observing herself draw a similar figure

(2) naming the elements of her figure -- "BIG" and "SMALL" -- so that she could talk about them and so describe what she was doing

(3) describing it in LOGO

```
TO GROWSHRINK :BIG :SMALL
1  FORWARD :BIG
2  RIGHT 90
3  FORWARD :SMALL
4  RIGHT 90
5  GROWSHRINK :BIG-10 :SMALL+10
END
```

Figure 9



The possibilities are endless. These are small discoveries. But perhaps one is already closer to mathematics in doing this than in learning new formal manipulations, transforming bases, intersecting sets and drifting through misty lessons on the difference between fractions, rationals and equivalence classes of pairs of integers. Perhaps learning to make small discoveries puts one more surely on a path to making big ones than does faultlessly learning any number of sound algebraic concepts.

4. <u>Some Physical Mathematics</u>

The turtle language is appropriate for many important physical problems. Consider, for example, the problem of understanding planetary orbits as if one were a junior high school student. One would find conceptual barriers of varying degrees of difficulty. Certainly the idea of the inverse square law is simple enough. Somewhat harder is the representation of velocities, accelerations and forces as vectors. But the insuperable difficulty in reading a text on the subject comes from the role of differential equations. The really elegant and intelligible physical ideas give rise to local differential descriptions of orbits; translating those into global ones usually involves going through the messy business called "solving" differential equations.

Turtle geometry helps at all these points. The use of vectors is extremely natural. And the local differential description takes the form of a procedure that can be run so as to produce a drawing of a solution or studied using theorems and analytic concepts about procedures.

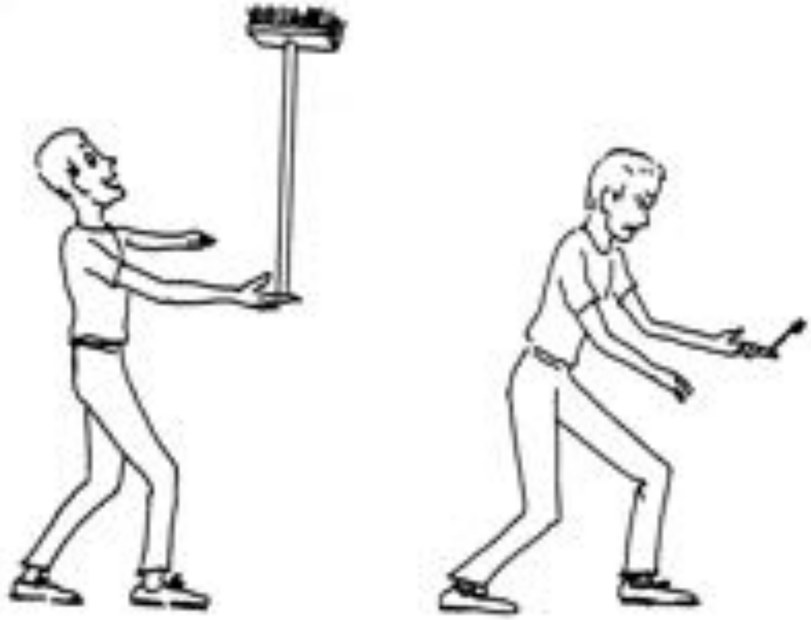
The framework for thinking about orbital theory in turtle terms presupposes prior contact with the concepts of state and of quantized time -- both of which occur very easily and naturally in many computational situations. The state of the "planet" is its position and a certain vector called, say "JUMP". If the planet were left alone it would move by :JUMP at every clock time. Thus it would go off, forever, in a straight line. In the presence of the sun, we think of it as undergoing two movements: it moves by :JUMP and then it falls into the sun! To make this more precise we put these two actions together using a procedure called "VECTORADD", which could be defined by the children or given as a primitive. Thus we obtain a LOGO procedure whose general idea will be intelligible to readers who try hard enough. (Two helpful comments: MAKE is the LOGO idiom for assignment, or setting values, so that line 1 in the procedure will cause the quantity VECTORADD OF :JUMP AND FALL to be computed and given the name "NEWJUMP". This computation assumes the existence of another procedure, called "FALL", which will compute the "fall into the sun vector". These ideas might seem confusing when presented fast; ten year old children understand them fluently when they are presented properly.)

```
TO FLY :JUMP
1  MAKE
       NAME "NEWJUMP"
       THING VECTORADD OF :JUMP AND FALL
2  SETHEADING (DIRECTION :NEWJUMP)
3  FORWARD (LENGTH :NEWJUMP)
4  FLY :NEWJUMP
END
```

Using this same idea one can easily deal in an experimental way with three bodies; one can design space-ship orbits, synchronous satellites and so on endlessly.

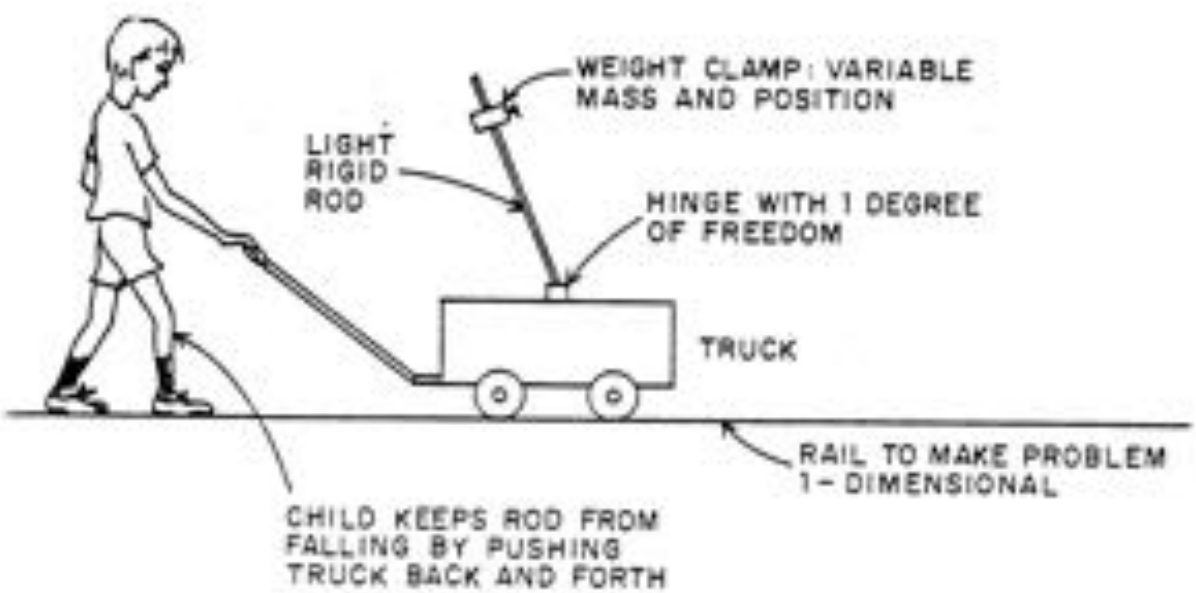
5. Control Theory as a Grade School Subject or Physics in the Finger Tips

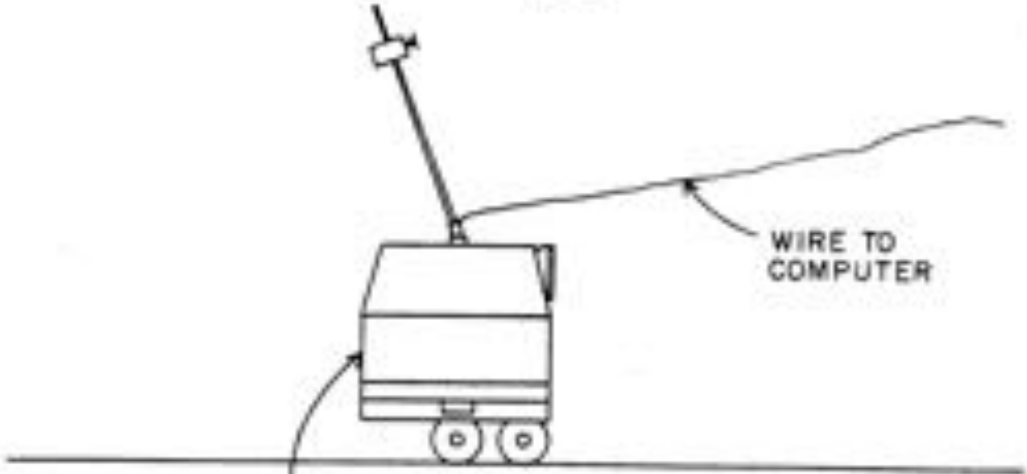
We begin by inviting the reader to carry out the illustrated experiments -- or to recall doing something similar.



One of the goals of this unit of study will be to understand how people do this and particularly to understand what properties of a human being determine what objects he can and what objects he cannot balance.

A "formal physical" model of the stick balancing situation is provided by the apparatus illustrated next:





A computer controlled version replaces the track and the child by a turtle with the angle sensor plugged into its sensor socket. A simple minded procedure will do a fair amount of balancing (provided that the turtle is fast!!):

```
TO BALANCE
1  TEST ANGLE > 10
2  IFTRUE FORWARD 8
3  TEST ANGLE < -10
4  IFTRUE BACK 8
5  WAIT 1
6  BALANCE
END
```

This procedure is written as part of a project plan that begins by saying: neglect all complications, try something. Complications that have been neglected include:

(1) The end of the line bug.

(2) The overshoot bug.
(Perhaps in lines 2 and 4 the value 8 is too much or too little.)

(3) <u>The Wobbly Bug</u>

The TEST in the procedure might catch the rod over to the left while it is in rapid motion towards the right. When this happens we should leave well alone!

One by one these bugs, and others can be eliminated. It is not hard to build a program and choose constants so that with a given setting of the movable weight, balance will be maintained for long periods of time.

6. What are the Primitive Concepts of Mathematics?

To see points and lines as the primitive concepts of geometry is to forget not only the logical primitives (such as quantifiers) but especially the epistemological primitives, such as the notion of a mathematical system itself. For most children at school the problem is not that they do not understand particular mathematical structures or concepts. Rather, they do not understand what kind of thing a mathematical structure is: they do not see the point of the whole enterprise. Asking them to learn it is like asking them to learn poetry in a completely unknown foreign language.

It is sometimes said that in teaching mathematics we should emphasize the process of mathematization. I say: excellent! But on condition that the child should have the experience of mathematizing for himself. Otherwise the word "mathematizing" is just one more scholastic term. The thrust of the explorations I have been describing is to allow the child to have living experiences of mathematizing as an introduction to mathematics. We have seen how he mathematizes a heart, a squiral, his own behavior in drawing a GROWSHRINK, the process of balancing a stick, and so on. When mathematizing familiar processes is a fluent, natural, enjoyable activity, then is the time to talk about mathematizing mathematical structures, as in a good pure course on modern algebra.

But what are the ingredients of the process of mathematizing? Is it possible to formulate and teach knowledge about how one is to tackle for example, the problem of setting up a mathematical representation of an object such as the hearts and flowers we discussed earlier?

Our answer is very definitely affirmative, especially in the context of the kind of work described above. Consider for example, how we would teach children to go about problems like drawing a heart. First step we say: if you cannot solve the problem as it stands, try simplifying it; if you cannot find a complete solution, find a partial one. No doubt everyone gives similar advice. The difference is that in this context the advice is concrete enough to be followed by children who seem quite impervious to the usual math.

A simplification of the heart problem is to settle, as a first approximation, on a triangle; which we then consider to be a very primitive heart.

```
TO TRI
1  FORWARD 100
2  RIGHT 120
3  FORWARD 100
4  RIGHT 120
5  FORWARD 100
END
```

TRI

Now that we have this construction firmly in hand we can allow ourselves to modify it so as to make it a better heart. The obvious plan is to replace the horizontal line by a structure line. So we write a procedure to make this. First choose it a name, say "TOP", then write:

```
TO TOP :SIZE
1  ARC :SIZE/2 180
2  RIGHT 180
3  ARC :SIZE/2 180
END
```

TOP

Replacing line 1 in TO TRI by TOP we get:

```
TO TRI
1  TOP 100
2  RIGHT 120
etc.
```

HEART WITH BUG

The effect is as shown! Is this a failure? We might have so classified it (and ourselves!) if we did not have another heuristic concept: BUGS and DEBUGGING. Our procedure did not fail. It has a perfectly intelligible bug. To find the bug we follow the procedure through in a very FORMAL way. (Formal is another concept we try to teach.) We soon find that the trouble is in line 2. Also we can see why. Replacing line 1 by TOP did what we wanted, but it also produced a SIDE-EFFECT. (Another important concept.) It left the turtle facing in a different direction. Correcting it is a mere matter of changing line 2 to RIGHT 30. And then we can go on to make the fully curved heart. Unless we decide that a straight-sided one is good enough for our purposes.

Straight-sided Heart

Curved Heart

Our image of teaching mathematics concentrates on teaching concepts and terminology to enable children to be articulate about the process of developing a mathematical analysis. Part of doing so is studying good models (such as the heart anecdote) and getting a lot of practice in describing one's own attempts at following the pattern of the model in other problems. It seems quite paradoxical that in developing mathematical curricula, whole conferences of superb mathematicians are devoted to discussing the appropriate language for expressing the <u>formal</u> part of mathematics, while the individual teacher or writer of text-books is left to decide how (and even <u>whether</u>) to deal with heuristic concepts.

In summary, we have advanced three central theses:

(1) The non-formal mathematical primitives are neglected in most discussions of mathematical curricula.

(2) That the choice of content material, especially for the early years, should be made primarily as a function of its suitability for developing heuristic concepts, and

(3) Computational mathematics, in the sense illustrated by turtle geometry, has strong advantages in this respect over "classical" topics.

The Personal Road to Reinventing Mathematics Education

Math education has fascinated me for a very long time. I was always good at arithmetic and despite having a pretty bleak elementary school experience; I could do what they called, “math.” Test scores in the 6th grade indicted that I was mathematically gifted and earned me a place in something called *Unified Math*. “Unified” was an accelerated course intended to rocket me to mathematical superiority between grades 7 and 12. Rather than take discrete algebra, geometry, trigonometry, etc., Unified Math was promised as a high-speed roller-coaster ride through various branches of mathematics.

Then through the miracle of mathematics instruction I was back in a low Algebra track by 9th grade and limped along through terrible math classes until my senior year in high school. In 12th grade, I enrolled in a course called, “*Math for Liberal Arts*.” Today this course might be called, “*Math for Dummies Who Still Intend to Go to College*.” I remember my teacher welcoming us and saying, “Now, let’s see if I can teach you all the stuff my colleagues were supposed to have taught you.”

This led to two observations:

1. Mr. O’Connor knew there was something terribly wrong with math education in his school.

2. I looked around the room and realized that most of my classmates had been in Unified Math with me in 7th grade. These lifeless souls identified as mathematically gifted six years ago were now in the “Math for Dummies Who Still Intend to Go to College” class. If this occurred to me, I wondered why none of the smart adults in the school or district had observed this destructive pattern?

Two things I learned in school between 7th and 12th grade kept me sane. I learned to program computers and compose music. I was actually quite good at both and felt confident thinking symbolically. However, majoring in computer science was a path closed to me since I wasn't good at (school) math – or so I was told.

I began teaching children in 1982 and teachers in 1983. I was 18-19 years old at the time. While teaching others to program, I saw them engage with powerful mathematical ideas in ways they had never experienced before. Often, within a few minutes of working on a personally meaningful programming project, kids and teachers alike would experience mathematical epiphanies in which they learned "more math" than during their entire schooling.

In the words of Seymour Papert, "They were being mathematicians rather than being taught math."

Teaching kids to program in Logo exposed me to Papert's "Mathland," a place inside of computing where one could learn to be a mathematician as casually as one would learn French by living in France, as opposed to being taught French in a New Jersey high school class for forty-three minutes per day.

I met Seymour Papert in 1985 and had the great privilege of working with him for the next 20+ years.

Papert was a great mathematician with a couple of doctorates in the subject. He was the expert Jean Piaget called upon to help him understand how children construct mathematical knowledge. Papert then went on to be a pioneer in artificial intelligence and that work returned him to thinking about thinking. This time, Papert thought that if young children could teach a computer to think (via programming), they would become better thinkers themselves. With Cynthia Solomon and Wally Feurzig, Papert invented the first programming language for children, called Logo. That was in 1968.

What makes Papert so extraordinary is that despite being a gifted mathematician he possesses the awareness and empathy required to notice that not everyone feels the same way about mathematics or their mathematical ability as he does. His life's work was dedicated to a notion he first expressed in the 1960s. Instead of teaching children a math they hate, why not offer them a mathematics they can love?

As an active member of what was known as the Logo community, I met mathematicians who loved messing about with mathematics in a way completely foreign to my secondary math teachers. I also met gifted educators who made all sorts of mathematics accessible to children in new and exciting ways. I fell in love with branches of mathematics I would never have been taught in school and I understood them. Computer programming was an onramp to intellectual empowerment; math class was a life sentence.

It became clear to me that there is no discipline where there exists a wider gap than the crevasse between the subject and the teaching of that subject than between the beauty, power, wonder, and utility of mathematics and what kids get in school – math.

> Papert has accused school math of “killing something I love.”
>
> Marvin Minsky said that what’s taught in school doesn’t even deserve to be called mathematics, perhaps it should just be called “Ma.”
>
> Conrad Wolfram, says that every discipline is faced with the choice between teaching the mechanics of today and the essence of the subject. Wolfram estimates that schools spend 80% of their time and effort teaching hand calculations at the expense of mathematics. That may be a generous evaluation.

Over the years, I’ve gotten to know gifted mathematicians like Brian Silverman, David Thornburg, Seymour Papert, Marvin Minsky, and Alan Kay. I’ve even spent a few hours chatting with two of the world’s most preeminent mathematicians, John Conway and Stephen Wolfram. In each instance, I found (real) mathematicians to embody the same soul, wit, passion, creativity, and kindness found in the jazz musicians I adore. More significantly, math teachers often made me feel stupid; mathematicians never did.

Time for Action

The 1999 National Council of Teachers of Mathematics Standards said, "50% of all mathematics has been invented since World War II." This is the result of two factors; the social science's increasing demand for number and computing.

These new branches of mathematics are beautiful, useful, playful, visual, wondrous, and experimental. Computing makes some of these domains accessible to even young children, and yet you are unlikely to find the likes number theory, chaos, cellular automata, fractal geometry, topology… in the K-12 math curriculum.

Hell, I dream of a day when a math textbook uses the symbol for multiplication used on computer keyboards for a half century. It makes my head explode when a high school student doesn't know how to ask a computer to multiply two numbers.

Since *No Child Left Behind*, parents, politicians, and educators have been engaged in a death match known as the Math Wars. The prescribed algorithmic tricks proscribed by The Common Core have thrown dynamite on the raging fire about how best to teach math. Ignorance, fear, and superstition are a volatile brew and impediment to learning.
Conrad Wolfram estimates that 20,000 student lifetimes are wasted each year by school children engaged in mechanical (pencil and worksheet) calculations. Expressed another way, we are spending twelve years educating kids to be a poor facsimile of a $2 calculator. Forty years after the advent of cheap portable calculators, we are still debating whether children should be allowed to use one.

We are allowing education policy and curriculum to be shaped by the mathematical superstitions of Trump voters. Educators need to take mathematics back and let Pearson keep "math."

Hard, Not Fun
When Barbie said, "Math is hard," the politically-correct class expressed their faux outrage, but Barbie was speaking a ubiquitous truth only tacitly acknowledged by the brave or those severely damaged my school math. If math is hard, fixing mathematics education is even harder.

Study after study tell us that kids hate math, computers are less likely to be used in a math class than anywhere else in school, teachers have little confidence in their own mathematical abilities and were poor math students, formidable gender gaps still exist – even the stupid test scores by which some measure "achievement" are static or worse.

Faced with an abundance of research, personal history, and good old-fashioned intuition while screaming from the rooftops that math education is a shambolic failure, we just double down on what does not work.

Hope
Against this backdrop of panic, misery, and despair there is room for optimism.

There is a renewed attention being paid to the importance of S.T.E.M. and S.T.E.A.M.

We live in a complex society awash in data. Citizenship depends on strong mathematical thinking and modeling skills.

Computational power has never been cheaper, easier to use, or portable. Today, you can ask your phone any question found in the K-12 math curriculum and receive an immediate answer. It can even "show all work." How will math education respond to this reality?

The maker movement has reenergized timeless craft traditions and supercharged such creative human expression with new tools and computational materials.

Kids are miserable. Parents are fed-up. They are not only opting out of standardized testing, but rejecting that which is tested and the way it is taught.

Why Progressive Educators Should Care About Reinventing Mathematics Education

I had a conversation with Dr. Papert in 2004 in which he was on-fire about the need to revolutionize math education with all the urgency our society can muster. When I asked if his focus on math education was because he was a mathematician, Papert rattled off more than a dozen reasons why this was a priority.

One argument in particular stayed with me while I have forgotten others.

Papert said that no pedagogical innovation of the past century has had any real impact on math education and if that were not disconcerting enough, it ultimately meant that in practice, no matter how progressive or learner-centered a school aspired to be, there was one point in the school day when "coercion was reintroduced into the system." Math class was when kids felt badly about themselves and were being taught irrelevant tricks they might need one day.

Papert argued that this scenario was corrosive to any other constructive efforts undertaken by a school, eventually undermining efforts like project-based learning, authentic assessment, student led inquiry, and other aspects of constructivist teaching. There is no way to make a noxious math curriculum more palatable.

Papert would ask how math class could feel more like art class, where students would become lost in their work, think deeply, act creatively, and produce an artifact they were proud of?

Most discussions of math education define "reform" as devising a clever new teaching trick or test intended to fix the kid and make them understand what's in a textbook relatively unchanged since the advent of movable type. This is the time for action.