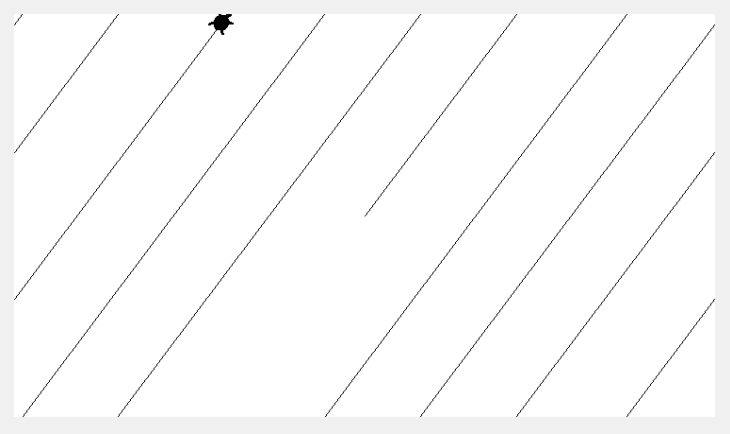
www.logofoundation.org

Rapping About Wrapping

By Seymour Papert and Michael Tempel

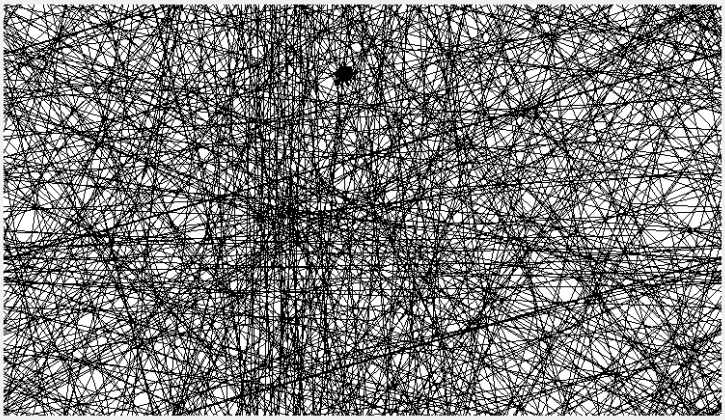
If you're a Logo teacher you've seen something like this

```
right 37
forward 3456
```



many times.

Many, many times. Repeated over and over creating intricate, seemingly haphazard patterns as the turtle goes off the top or side of the screen and reenters at the bottom or opposite side. Kids love to do this. Teachers hate it, feeling that their students are heading into a dead end that has little educational value. Let s stop this nonsense and draw a square!



Is there anything worthwhile going on when kids are playing with wrapping? If there is, how can we bring it out? What can we do to extend the activity into new areas of learning? Since children love wrapping, one essential element of effective learning is taken care of. The learner is engaged and paying attention. But engaged in what? Some careful observation is called for. The paper by Sue Berthouex is the result one teacher's thoughtful look at her first graders' work with wrapping. For example, students increased the size of the input to forward until Logo couldn't

handle the number. The awareness that the size of a number is related to the number of digits, and there mere fact that first graders are using multi-digit numbers is out of the ordinary for that age group. With older students, different learning experiences emerge from the same playful wrapping. A seventh grader wrote these procedures:

```
to neato
rt 91
neato1
end

to neato1
fd 10000
setbg bg + 1
neato1
end
```

Using 91 or 89 as inputs to **forward** produced interesting patterns. With 90 as the input, the result is rather dull. Certain numbers (0, 90, 30, 60, 45, etc.) are boring. Numbers close to the boring numbers produce the most interesting results. Both the first graders and the seventh grader were intuitively learning something mathematical. But a mistake that many Logo teachers make is to leave it at that. These informal learning experiences are just the beginning. What can a teacher do?

Make it Explicit

For starters, talk to the kids about what's going on. Make the mathematical content of what they're doing explicit. For the younger students, talk about how more digits means a larger number; a larger number means more of something. Relate the Logo experience to something else. How many people are there in the class? 34. How many people live on your block? 974. How many people live in your city? 7120000. More digits mean more people.

Ask Some Questions

Let's look more closely at the general wrapping pattern

```
rt <some number >
fd <a very big number >
```

repeated many times. One might ask students if they can tell the effect of **right** alone and of **forward** alone. This is easy enough to check. Try

```
repeat 999 [fd 2345]
```

Then try something like

```
repeat 999 [rt 41]
```

The first just draws a straight line. The second draws nothing as you watch the turtle twirl in place. It is the combination of moving and turning that lets you do interesting things. To emphasize this, play turtle. Try moving around the room strictly obeying turtle commands. What can you do just using right? You might get dizzy, but you won't get too far. Using only forward may be useful in a narrow corridor, but otherwise limits your mobility.

Now here's an interesting observation: With some inputs to **right,** a sufficient number of repetitions of **forward** and **right** will eventually fill the screen completely. With others, no matter how many times the **forward** and **right** combination is repeated, there will be holes. Which numbers fill the screen? Which leave holes? Is there a pattern to the holes? How many repetitions does it take to just fill the screen? Does this depend upon the input to **forward,** the input to **right** or both?

The "boring numbers" leave very large holes. Numbers close to the boring numbers generally cause the screen to be filled. It turns out that the boring numbers are often those that may be divided evenly into 360. When the result of such a division is a small number, the visual product is most uninteresting. For example, 360 ÷ 180 = 2, and

```
rt 180 fd 999
```

just draws a single line. Using 90, 45, 30, 60, or 120 as inputs to **right** also produce rather static patterns Now when the input to **right** is greater than 180, it Will not be evenly divisible into 360 but may produce patterns similar to (or identical to) some of those using numbers less than 180. For example

```
rt 270 fd 999
```

results in the same pattern as

```
rt 90 fd 999
```

although it emerges differently. This leads to another way of looking at the numbers that are inputs to **right.** Does the input to **right** have any common factors with 360? (A common factor is a number that may be divided evenly into both numbers.) If so, how large is the largest common factor? It turns out that if the input to **right** has no common factors with 360, then repeated wrapping will cause the screen to fill completely. The higher the common factor, the less the screen will fill. For example, 37 has no common factors with 360. **Forward** <any big number> **right 37** fills the screen. The highest common factor of 90 and 360 is 90. **Forward** <any big number> **right 90** produces a pattern of perpendicular lines with lots of space around them.

Shrink Wrapping

Here's something to try. Reduce the input to **forward** until a particular pattern no longer wraps. You can do this by simply moving the decimal point. For example, instead of:

```
repeat 999  [fd 3456 rt 67]
```

try

```
repeat 999 [fd 345.6 rt 67]
```

if that still causes wrapping try

```
repeat 999 [fd 34.56 rt 67]
```

What does it look like? The boring numbers generally produce recognizable shapes such as squares, triangles and circles. The wrapping patterns that filled the entire screen produce donuts or disks. It turns out that the wrapping patterns are really polygons folded up on themselves. When the polygon is made with many sides and a small turn it looks like a circle. For example

```
repeat 40 [fd 10 rt 9]
```

makes a 40 sided polygon, but it looks like a circle.

Take a paper disk and fold over the edge. Then make a parallel fold opposite to this one. Make two more parallel folds at right angles to the first two. The result is reminiscent of the pattern on the screen that is produced by

```
repeat 36 [fd 30 rt 10]
```

When the input to forward is very large, a familiar wrapping pattern emerges. The image produced by

```
repeat 999 [rt 3 fd 999]
```

is really a very large circle, (or a 120 sided polygon) folded up on itself. Actually, by making the sides of this polygon large, they appear as straight lines. When shape is small enough to appear whole on the screen, for example using

```
repeat 999 [rt 3 fd 3]
```

the straightness of the sides isn't apparent.

Randomness

Let s start with the original wrapping idea and go off in a different direction. After each large **forward,** the turtle ends up at some point on the screen, but it's hard to predict where that will be. When the turtle is drawing while wrapping, the pattern on the screen obscures the position of the turtle. If we clear the screen and pick the pen up, the positions after each forward are obvious. Try to predict where the turtle will appear next when you type

```
rt 73 fd 6769
```

You probably can't. But is this random? You might argue that it isn't since the position could be calculated, albeit with difficulty. In a more concrete way, however, it is random. You can't predict it. In fact, is the random number generator built into Logo "really" random?

Try this:

```
pu
repeat 999 [rt 73 fd 6769 pd stamp pu]
```

It's a good idea to use a single dot as the turtle shape when doing this so that stamps near each other won't overlap Do some inputs to **right** and **forward** produce recognizable patterns while others generate seemingly random distributions? Do the patterns look anything like a nighttime sky filled with stars?

How about letting Logo's **random** do the work? Try

```
repeat 999 [rt random 360 fd random 1000]
```

How does the result compare with the patterns you just got? Here's another wrapping wrinkle:

```
repeat 999 [fd 10 rt 90 fd 10 lt 90]
```

Instead of wrapping straight lines, this produces wiggly lines that make staircases. Depending upon the computer you are using, this might result in "tiling". The lines meet so that you end up with a screen full of 10 by 10 boxes. If this doesn't happen on your computer, is there a number other than 10 that does produce tiling? Can you generate other tiled patterns using different inputs to **forward** and **right?** One fourth grader went on an interesting exploration based on this tiling effect. In producing the tiled pattern she also set random colors at each step:

```
repeat 999
     [setc 1 + random 15
      fd 10 rt 90
      setc 1 + random 15
      fd 10 rt 90]
```

(Her version of Logo Writer for the IBM PCjr had 16 colors. **1 + random 15** selects any color but 0 which is invisible and would not draw a line) The result is a pattern of squares, the borders of which are random colors. She then wanted to fill in the squares with random colors. After the pattern was complete, she put the turtle back home:

```
pu home
```

and then in the middle of a square:

```
fd 5 rt 90 fd 5 It 90
```

then to fill the square she used a command very similar to the one that drew them in the first place:

```
repeat 999
     [repeat 999
          [pd fill pu fd 10 rt 90 fd 10 lt 90]
     ]
```

The screen began to fill with a colorfully random pattern of squares. When all the squares were filled the turtle just kept on filling new colors over the old. But after a while, something surprising began to happen! Here and there, two squares would merge. Then larger groups of squares all filled in the same color. Why?

Each square has four bordering lines which may be of any color. The square itself may, by chance, be the same color as one of its borders, let's say it's red. The square and its border are then merged. Now, if the square across this border happens to get filled with red at some point, this square is merged with the first. Since there is no longer a border between them, when what used to be either of them is filled, the whole region fills in the same new color. As time goes on this double square region will merge with a neighbor in the same way. Eventually the whole screen becomes uniform. How long does this take? Because of the randomness of the process, we can't be sure. The pattern emerges differently each time, but it is certain that the screen will become one color at some point.

Now that's interesting. An absolutely certain outcome is the result of a random process. Actually, that should not be surprising. Many predictable natural phenomena are the result of random processes. The molecules of air in the room are moving randomly, but it is virtually certain that they will not all move to one corner of the room, leaving a vacuum elsewhere. The positions of electrons in atoms are random, but matter may be quite stable.

Another idea touched upon by this Logo exploration is that of reversibility of process. You may make a stack of blocks, take it down, put it up, any number of times. The turtle may draw a line, then erase it, then draw it again, and so on. There are other processes that are not reversible. If you put a drop of ink in a glass of water it will gradually mix in, never to be separated out again. The merging of squares as they fill cannot be reversed. Once two squares merge, there is no longer a border between them so they may never again be separated according to color.