Turtle, Art, TurtleArt

Paula Bontá, *paula@playfulinvention.com*
Playful Invention Company. Montreal, Quebec, Canada

Artemis Papert, *artemis@turtleart.org*
Independent artist. Aberdeen, Scotland

Brian Silverman, *brians@playfulinvention.com*
Playful Invention Company. Montreal, Quebec, Canada

Abstract

TurtleArt is a microworld for exploring art through turtle geometry. It is similar to the Logo programming language in that the main actor is a turtle that can draw. It is different from Logo and other constructionist systems in that it is focused on art. The vocabulary is small but provides rich control of colours, shades, pen widths, etc.

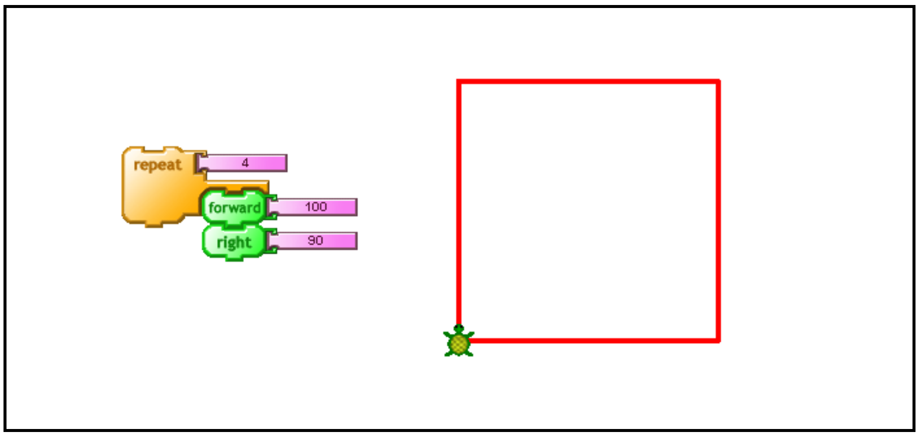
Introduction

TurtleArt does only one thing and tries to do it very simply and very well: bring geometry and art together through programming. Like the Logo programming language turtle geometry has a central role. However the main focus of TurtleArt is static artistic images.

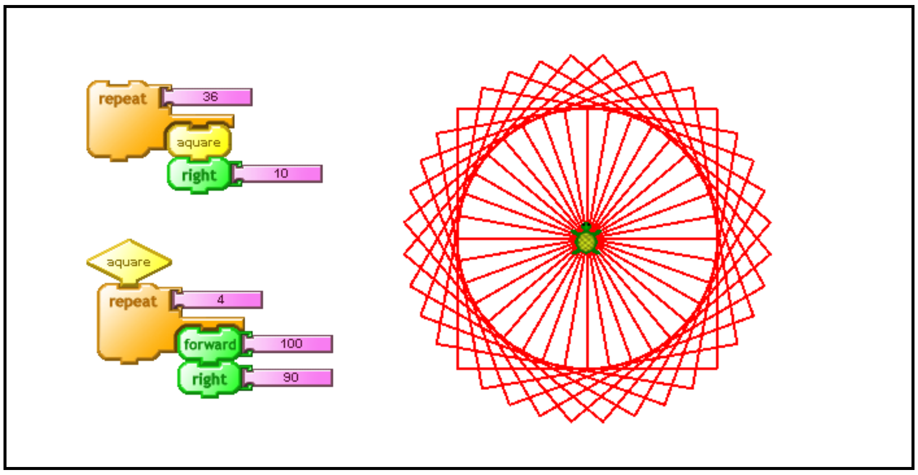
Programming in TurtleArt

TurtleArt follows the recent trend of programming by snapping together blocks. It borrows from the earliest versions of Logo by having a vocabulary centred around Turtle Geometry.

Here is the TurtleArt version of a square:



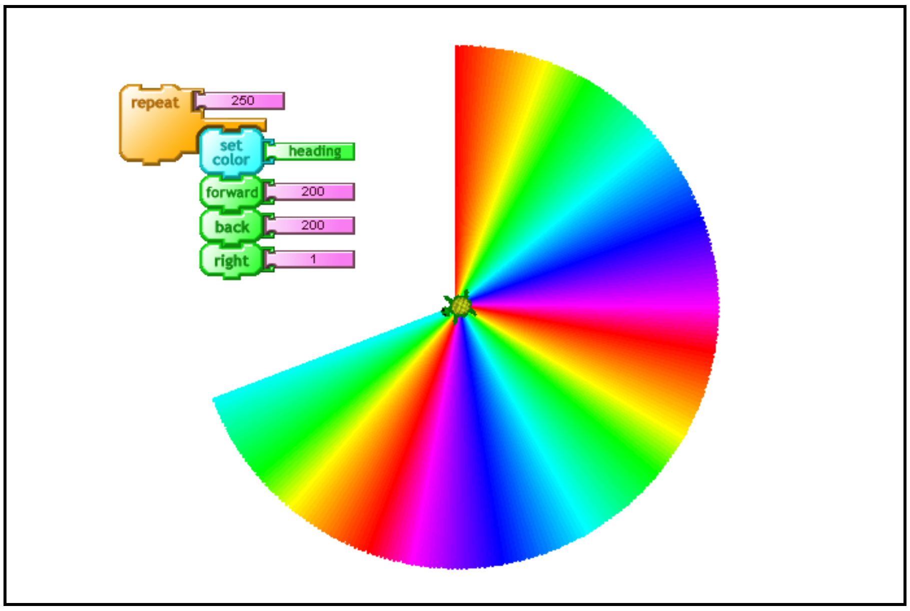
Procedures are defined by adding a "hat" to a stack of blocks. Once defined they can be used in other stacks:



```
repeat 36
  aquare
  right 10

aquare
repeat 4
  forward 100
  right 90
```

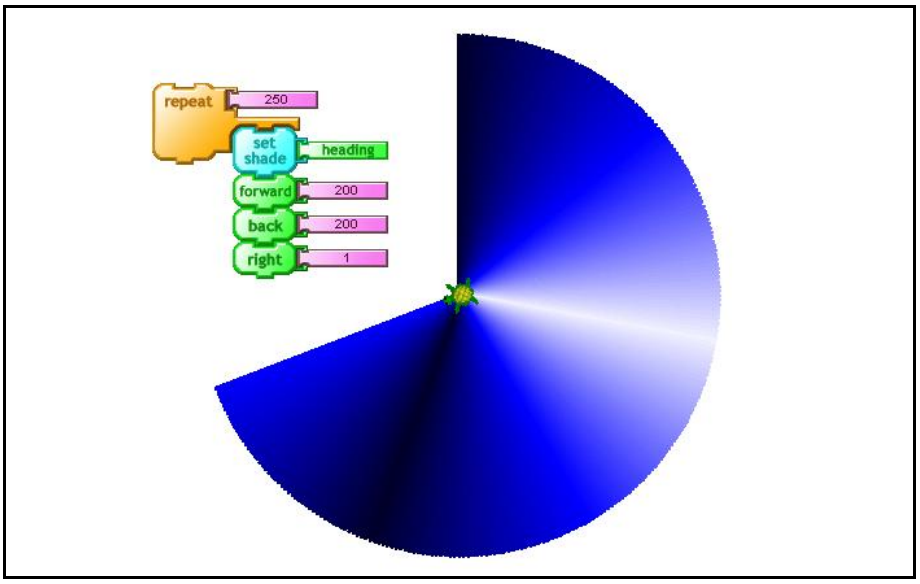
The turtle’s drawing colour is controlled by setting a colour (hue) and a shade (lightness).

The colours:



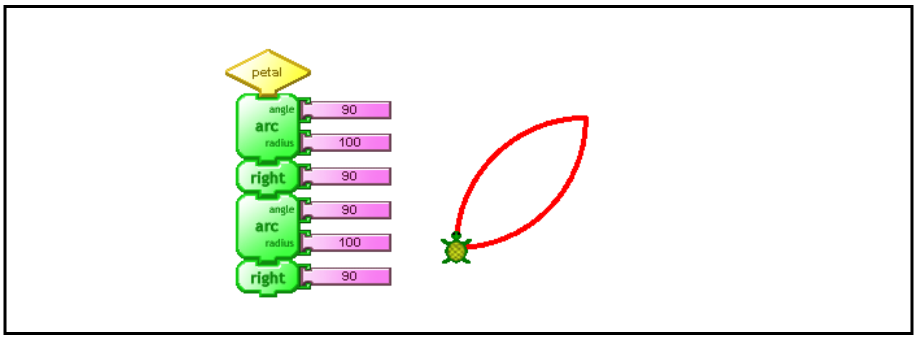
```
repeat 250
  set color heading
  forward 200
  back 200
  right 1
```

and the shades:

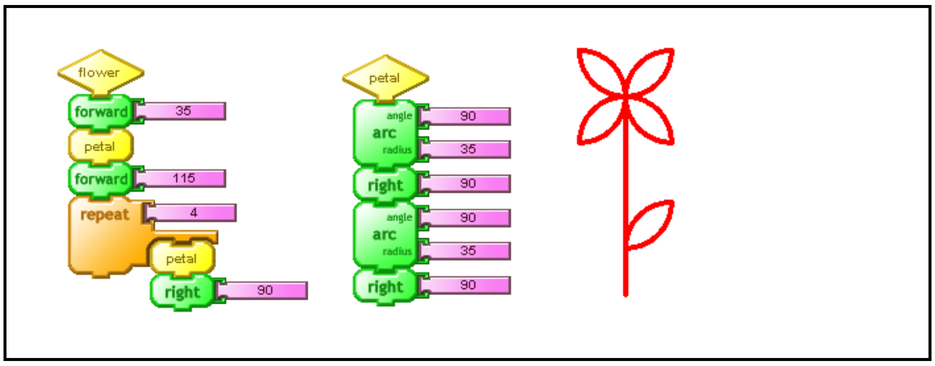


TurtleArt has an arc block. It allows drawing circles and arcs of a desired radius without knowing pi or the formula connecting a radius to a circumference.

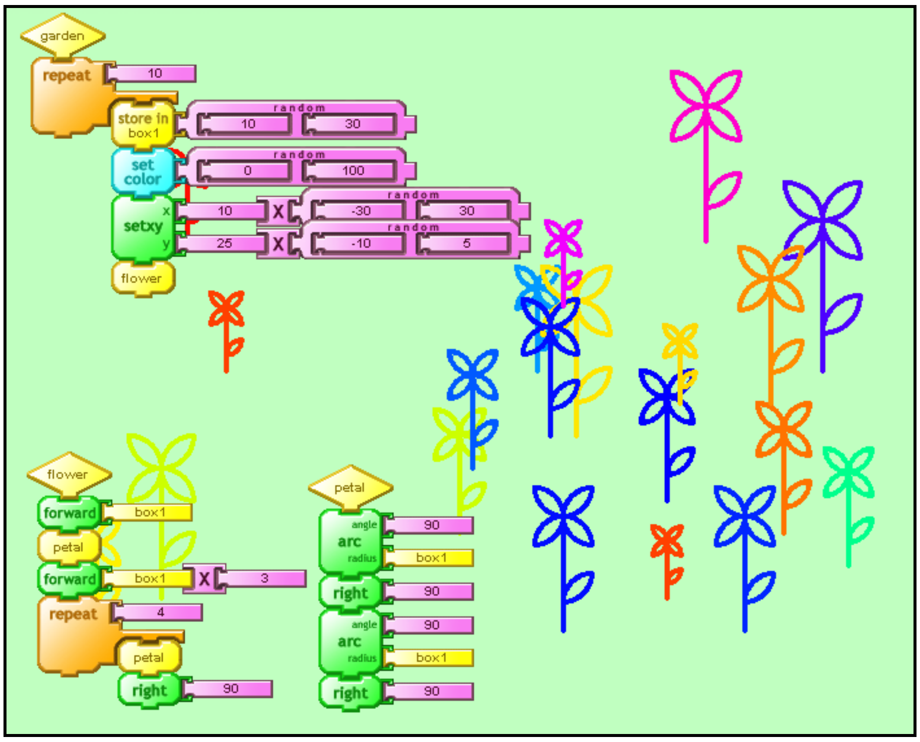
Two arcs put together make a petal:



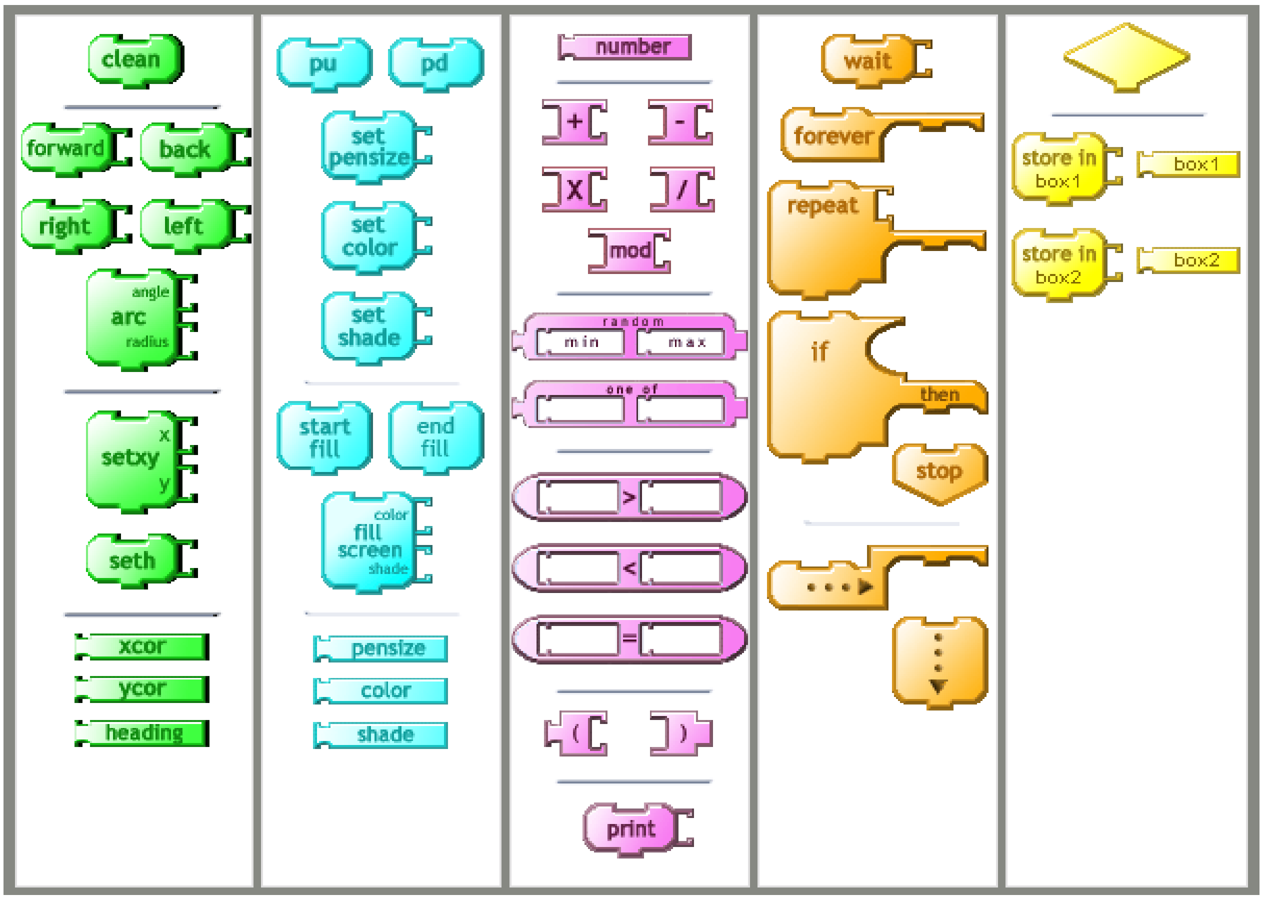
With several petals you can make a flower:



And with several flowers you can make a garden (Papert, 1971):



Here is the entire TurtleArt vocabulary:



Design of TurtleArt

Design is the art of making choices. Some are based on objective criteria. Others are more subjective and based on the preferences of the designers. We'd like to say something about some of our more salient choices.

In TurtleArt we tried to keep everything simple. Period. It is extremely minimalistic. We felt that if something isn't used often then it should not be there. A more common design approach is to feel that it's ok to have things that are useful only in rare cases. The thought is that if you don't need it just don't use it. We prefer otherwise. We agree with the Zen of Palm: "focus on what users do 80 percent of the time and try to ignore the other 20 percent" (PalmSource, Inc., 2003).

TurtleArt has only one object, the turtle. TurtleArt is also single threaded. Most recent versions of Logo have multiple objects of multiple kinds and support multiple threads. We consciously moved "backwards" in the name of simplicity. If the goal is static images, one turtle and one thread is good enough, and in the name of simplicity good enough turns out to be good enough.

The central exploration in TurtleArt is an artistic one. We feel that we can best empower that by a sharp focus on that exploration. There will be people who bump against the walls or the ceiling. Those people have other software choices available to them, some excellent. It is a non-goal for us to have TurtleArt also be good as a video game maker, a simulation language, or a system for multimedia presentations.

Sometimes the hardest part of a design is deciding what not to include. It's easy to invent new features. It's often even relatively easy to implement them. However, a minimalist design aesthetic suggests that things should only be included if they really fit. An example is that we included only a minimal set of mathematical operations. Trigonometric functions and square root are absent. Yes, this a limitation, but a conscious one.

Another important facet of the design is that we explicitly encourage programming. For example, there is no direct manipulation way of setting the turtle's pen colour. Scratch and PicoBlocks have a SETCOLOR block. However they have a “point and click” way of setting a colour in addition to the programmatic way. In TurtleArt we skipped the “point and click” option in order to encourage a description that can be readily manipulated through programs.

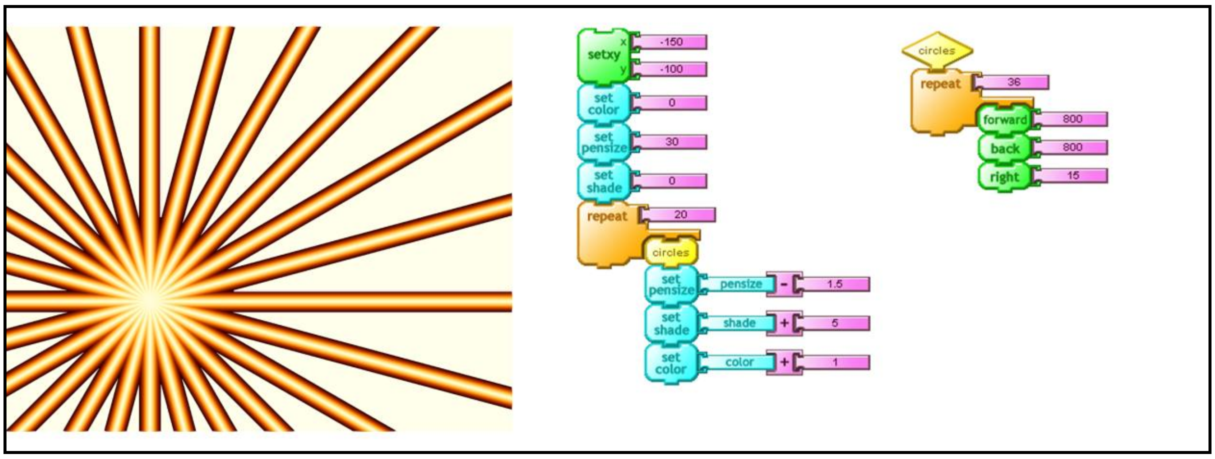
In the design of the user interface and of the blocks language we chose to emphasize visual stability. Nothing flashes, resizes, moves or animates automatically. We feel that this leads to a simpler, friendlier more tangible user interface. We recognise that this is opposite to the modern trend of having animations as a part of the interaction design and having elements that constantly update to provide real time feedback.

Overall, TurtleArt conforms to the design principles described by Resnick and Silverman (2005). Notable exceptions are that the "walls" aren't that wide and there aren't "many paths, many styles".

The Art of TurtleArt

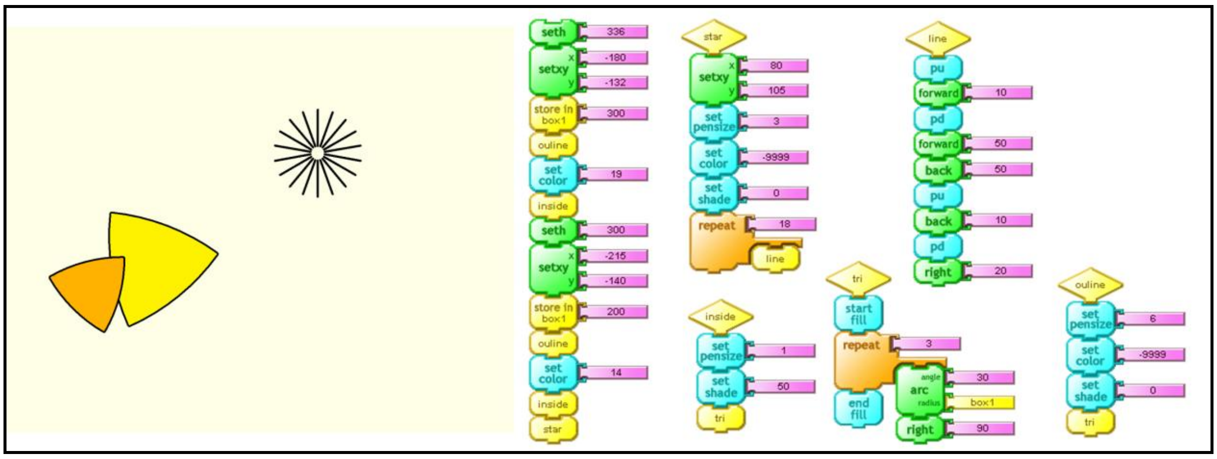
We consider our work with TurtleArt as an exploration. It is an exploration of the interaction of formula, choice, and chance. Formula is very common in mathematical and computational art, e.g. fractals, Julia and Mandelbrodt sets, tilings, tessellations, etc. Choice is the result of an artist making a decision, picking a shape, colour, texture, composition of elements, etc. Chance is added to the mix using randomness, the result may be constrained but is never fully determined.

We'd like to present some images where formula, chance, and choice are mixed in varying proportions.



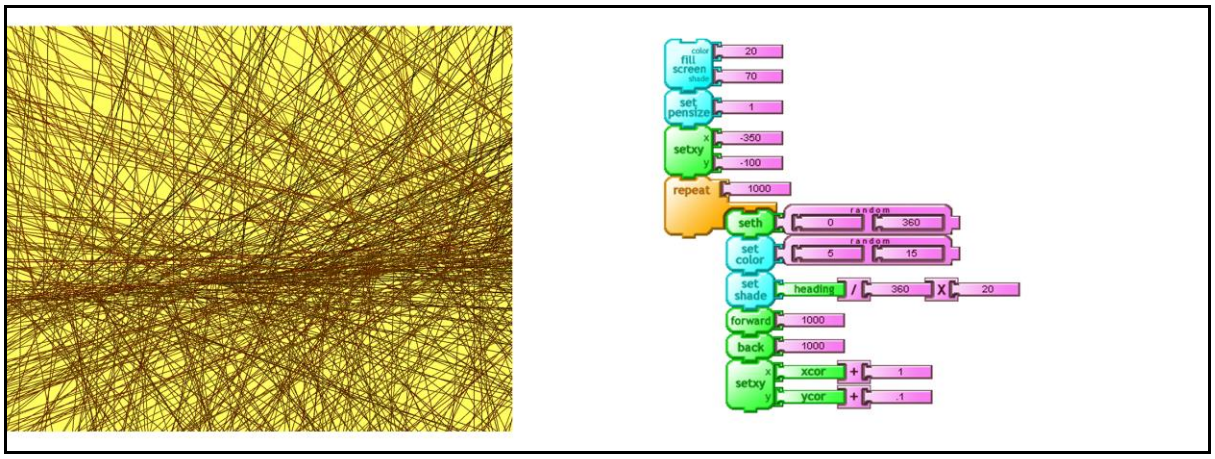
```
setxy x -150 y -100
set color 0
set pensize 30
set shade 0
repeat 20
    circles
    set pensize  pensize - 1.5
    set shade  shade + 5
    set color  color + 1

circles
repeat 36
    forward 800
    back 800
    right 15
```

“Lion’s Teeth” is based on a simple formula. Draw a line, turn a little, repeat. The basic pattern is repeated with different pen widths and colouring.



```
seth 336
setxy x -180 y -132
store in box1 300
ouline
set color 19
inside
seth 300
setxy x -215 y -140
store in box1 200
ouline
set color 14
inside
star

star
setxy x 80 y 105
set pensize 3
set color -9999
set shade 0
repeat 18
    line

line
pu
forward 10
pd
forward 50
back 50
pu
back 10
pd
right 20

inside
set pensize 1
set shade 50
tri

tri
start fill
repeat 3
    arc angle 30 radius box1
    right 90
end fill

ouline
set pensize 6
set color -9999
set shade 0
tri
```

“Simple” has very little formula. Every element is placed by choice. Doing this often requires a fair bit of code.



```
fill screen color 20 shade 70
set pensize 1
setxy x -350 y -100
repeat 1000
    seth random 0 360
    set color random 5 15
    set shade heading / 360 X 20
    forward 1000
    back 1000
    setxy x xcor + 1 y ycor + .1
```

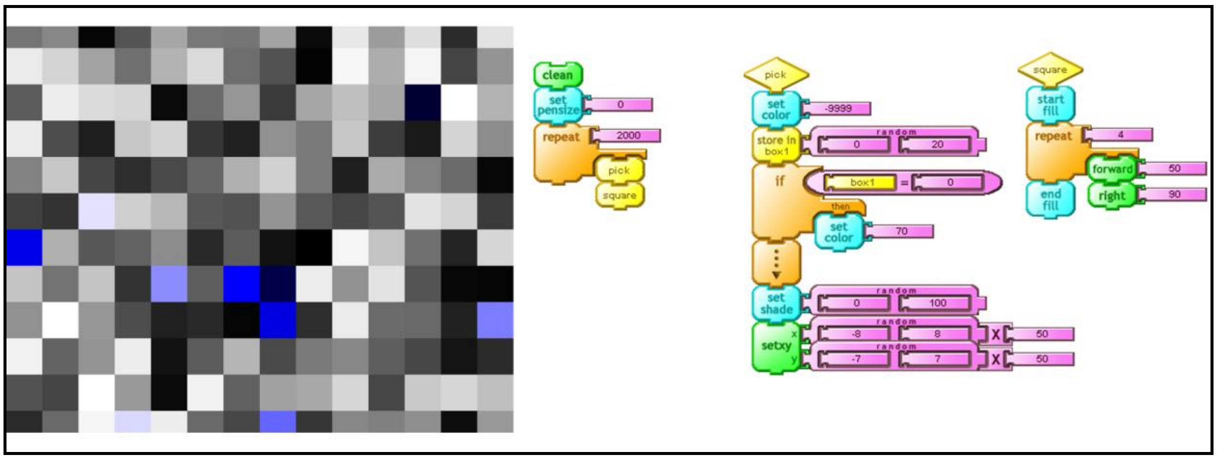
“Bridge to Nowhere” gets its texture from a controlled use of randomness. Run it again and it will be different in detail, though still very similar in texture.

Art through Programming

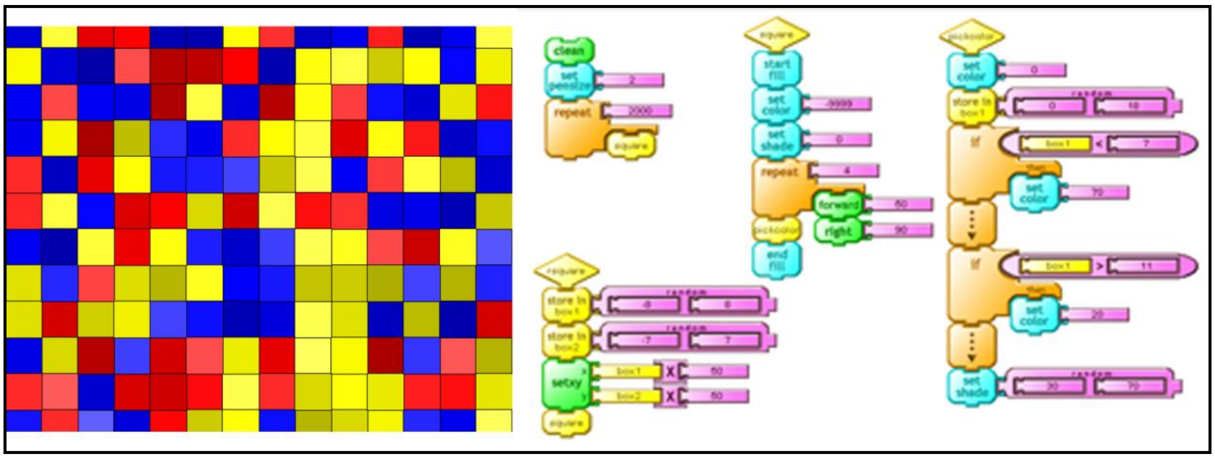
With TurtleArt we create images by creating programs. This has a collection of affordances. One obvious one is that a computer can draw thousands or even millions of strokes in a short amount of time. Less obvious is that programming allows for an evolution of images. We often create images that leave us with the feeling of being a good idea but needing further exploration. It isn't unusual for us to explore variants of a particular theme. We can modify parameters and code sequences in ways that change images to include what we like and leave aside what we don't. This process works exceptionally well with a programmatic description.

Here is an example of an evolution of images.

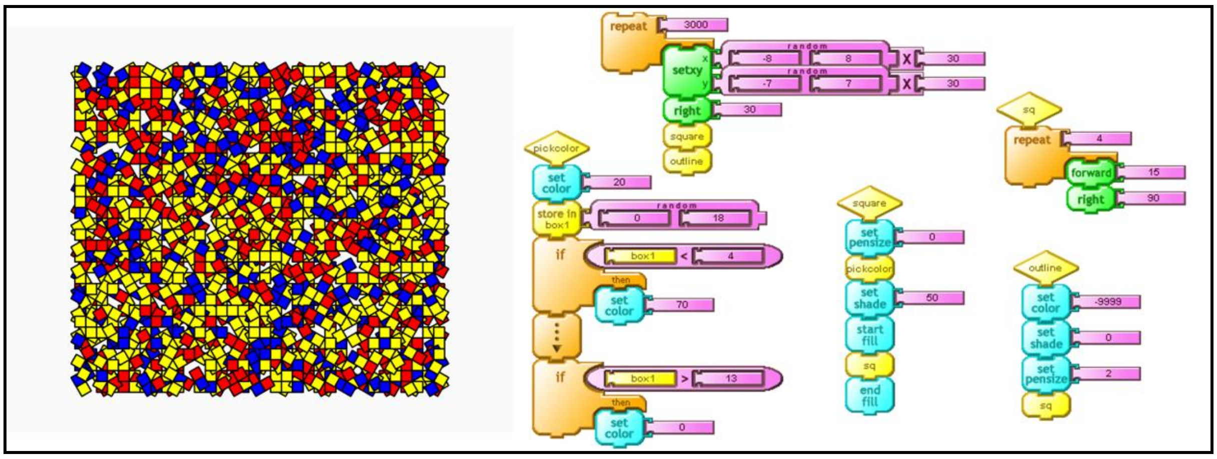
A first image looks like a low resolution picture...



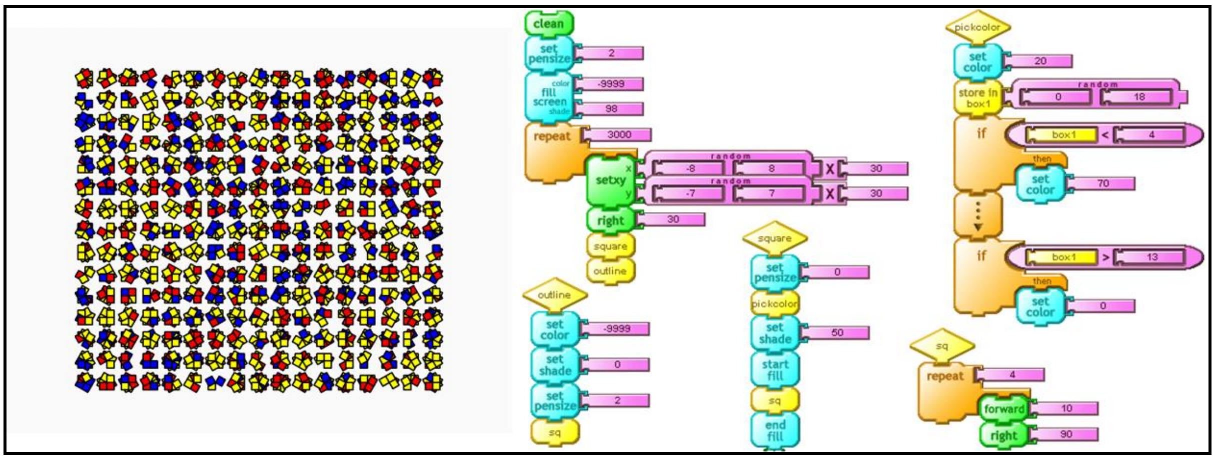
add some colour...



smaller squares, less lined up...



form little groups...



Conclusion

In 1971 Papert and Solomon described twenty things to do with a computer. These included making turtles, drawing pictures, playing music, building balancing robots and other activities. Back then only a few of those activities were possible, Turtle Geometry being one of them. It took a couple of decades before technology made it possible to do most of the other twenty things in a way that was accessible to children. A way of understanding the evolution of Logo and its successors is that Logo grew to be able to do more and more of the things described in that early paper. Along the way Turtle Geometry and drawing with the Turtle slipped to the back burner. We feel that this is unfortunate. TurtleArt tries to bring Turtle Geometry back and to do it in a way that empowers artistic expression.

For more information about TurtleArt, please visit http://www.turtleart.org

Acknowledgments

Mitchel Resnick, Natalie Rusk, Robbie Berg were centrally involved the design of PicoBlocks, essentially the same blocks language as the one in TurtleArt.

References

PalmSource, Inc. (2003) *Zen of Palm*, Document Number 3100-002-HW, June 13, 2003. http://www.accessdevnet.com/docs/zenofpalm.pdf (accessed March 2010).

Papert, S. (1971) *Teaching children to be mathematicians vs. teaching about mathematics.* Artificial Intelligence Memo No. 249 and Logo Memo No. 4.

Papert, S. & Solomon, C. (1971). *Twenty things to do with a computer*. Artificial Intelligence Memo No. 248 and Logo Memo No. 3.

Resnick, M. & Silverman, B. (2005) *Some Reflections on Designing Construction Kits for Kids*. Proceedings of Interaction Design and Children conference, Boulder, CO., USA.