TurtleBlocks: Blocks for Constructing Tangible Turtle Tracings

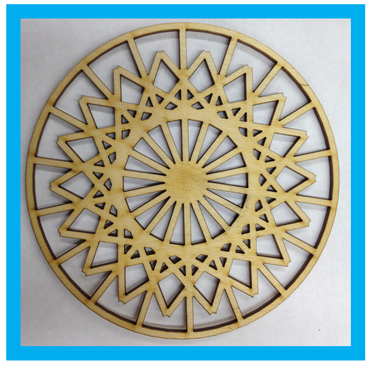
Karishma Chadha, Erin Davis, Emily Erdman
Franklyn Turbak, Advisor
Computer Science Department, Wellesley College

What is TurtleBlocks?

In 1980, Seymour Papert introduced Turtles, abstract creatures that follow simple commands that control their movement around a canvas with "pens" in their bellies. Turtles have long been used in introductory programming classes, allowing novice programmers to create intricate patterns through simple commands.

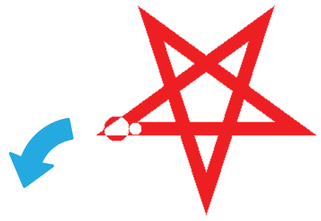
We have further extended Turtles in our block based language, TurtleBlocks, built upon MIT's OpenBlocks. The block structure of TurtleBlocks eliminates common textual syntax errors by having jigsaw-like pieces that strongly suggest how program fragments should be composed.

TurtleBlocks boasts the unique ability to create physical manifestations of turtle designs out of various materials using laser and vinyl cutters. This feature sets TurtleBlocks apart from other block-based turtle programming environments, such as TurtleArt, that focus on creating colored pictures.

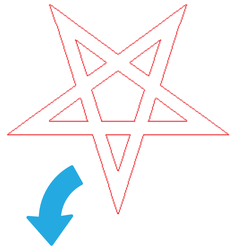


Write the code to describe the turtle's movement.

The turtle draws the image according to the block instructions.

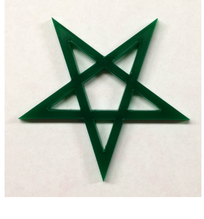


After selecting to print, the boundary of the image is taken to guide the laser and vinyl cutters.

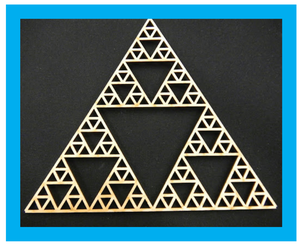


Resulting tangible artifacts

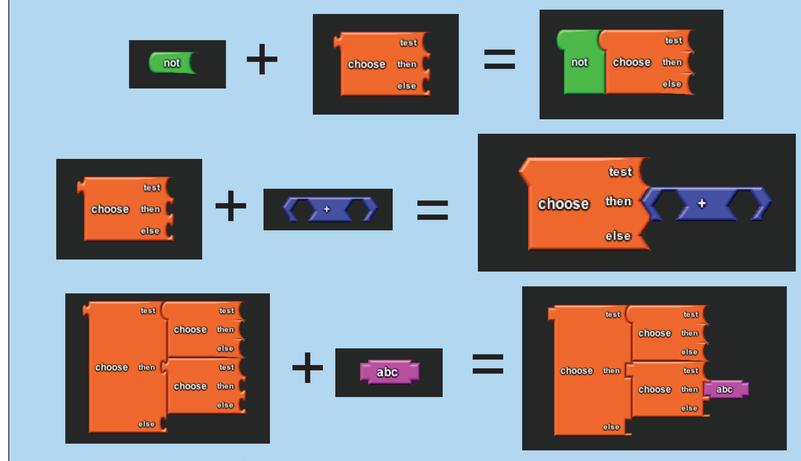
Cardstock from vinyl cutter



Acrylic from laser cutter

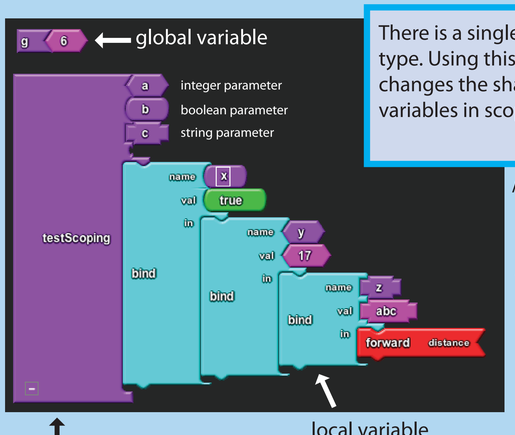


Polymorphism



TurtleBlocks has several primitive typed blocks which are determined by the shape of the plugs and sockets. TurtleBlocks also supports the concept of polymorphic typing which allows generic blocks to be applied to multiple primitive types. We have extended polymorphism to handle situations where multiple polymorphic blocks are linked and introducing one primitive type correctly changes the types of all related plugs and sockets.

Variables and Scoping

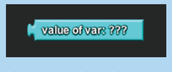


There is a single variable reference block with polymorphic type. Using this block in a socket with definite type not only changes the shape of its plug, but also updates the list of variables in scope to include only variables of that type.

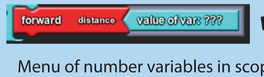
procedure declaration

local variable declaration

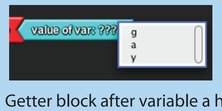
A polymorphic variable reference ("getter")



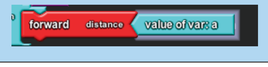
The getter block plugged into a number socket



Menu of number variables in scope at getter

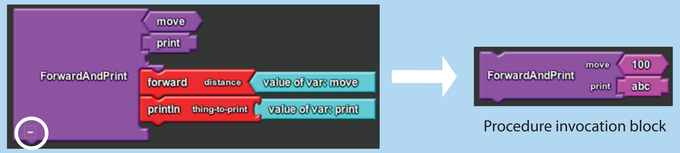


Getter block after variable a has been chosen



Procedures

Users have the option to define their own procedures by wrapping a set of connected blocks in a procedure block. To invoke a defined procedure a user creates an invocation block by clicking a button on the declaration block. The corresponding invocation block includes any defined parameters as they are named in the declaration. As the declaration block is updated, the invocation block changes appropriately in real time.



Procedure declaration block

Procedure invocation block

invocation button