Drawing Trees & Animating Tree Changes

Sandro Badame
<sbadame.student@manhattan.edu>

Peter Boothe
<peter.boothe@manhattan.edu>

Dynamic Trees

Dynamic trees appear in many applications:

- Lisp
- Data Structures
- Tree Automata
- Computer Algorithms

It would be useful to be able to draw trees and animate them changing over time. This will allow students (and anyone else) to visualize the complex algorithmic processes which trees enable. A tree layout and animation library could enable a whole host of new visualizations, both instructor-written and (more excitingly) student-written.

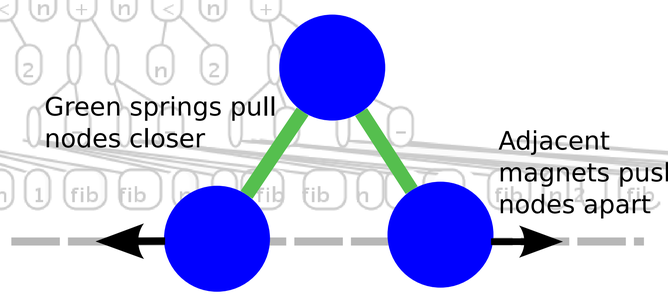
Currently, to visualize a tree process, the programmer must be able to program both complex algorithms code, as well as complex graphics code. This is a rare combination of skills, which means that many visualizations never get written. We provide a library which attempts to simplify the graphics aspect of this tricky task.

A Physical Model for Tree Layout

We lay out trees by fixing each node at a depth corresponding to the number of hops from the root node. We then physically model every vertex of the tree as a magnet, which will repulse the other nodes in the tree according to the inverse square of their distance.

So that our tree does not immediately fly apart, we model every edge of the tree as a spring, which will serve to pull connected edges closer to one another. To further restrict things we only allow tree vertices to move left and right, never up and down.

Our layout algorithm is then to simulate this physical system evolving over time. The system settles into an equilibrium rather quickly, and creates layouts which are pleasing to the eye.



Laying out a Tree

1) Fix the root node to the middle top position.

2) For each node of a given level treat every link as a spring and every node as a magnet and calculate the resultant forces.

3) Simulate the change in position due to these forces and update all node positions by a small amount.

4) Display the system to the user, and repeat

To Animate Tree Changes

1) Layout the new tree using the method above. Do this layout off screen.

2) Fade out any of the nodes of the currently displayed original tree that have been removed.

3) Compare the locations of the nodes in the original tree to those in the new tree, then animate the movement of the original nodes to their new positions.

4) Fade in any nodes that only exist in the new tree.

Multiple changes can be either combined into a single animation step, or each step may be animated individually. This allows artistic flexibility when creating a new visualization.

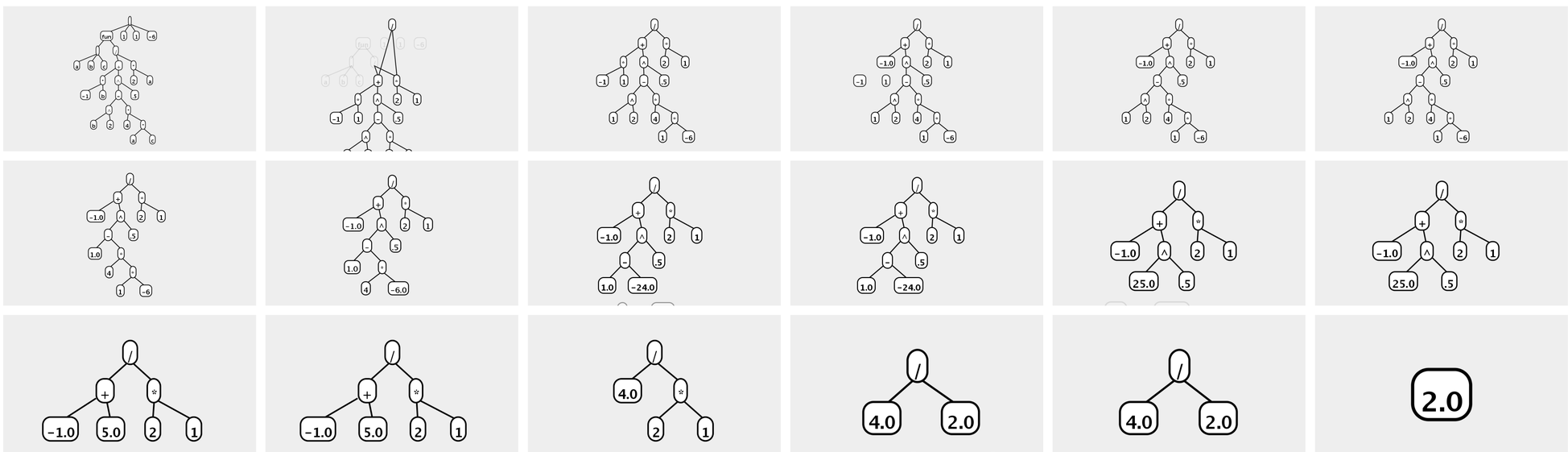
We produced a library and 4+ example visualizations, two of which are shown below.
Our library is available at http://github.com/pboothe/Lispy-Animator

Execution of Lisp Code Calculating the first root of x^2-x+6=0

The lisp code

```
((fun (a b c)
   (/ (+
        (* -1 b)
        (^ (- (^ b 2) (* 4 (* a c))) .5))
      (* 2 a)))
 1 1 -6)
```

calculates the quadratic formula with a=1, b=1, and c=-6. The code eventually evaluates to 2



Execution of a Tree Automata that evaluates logical expressions

Automaton Rules

LEAF	NOT	AND	OR
1→ true	true → false	true false → false	true false → true
0→ false	false → true	true true → true	true true →true
		false false → false	false false →false
		false true → false	false true → false

The rules above are used to reduce the tree to its simplest form. Every step along the way is animated for the user to see. If a tree eventually turns into the value "true", then the original expression was a logical tautology and will be accepted by our automaton.

Output of our visualization system