Circuitry

Was I supposed to cut the red wire or the blue wire?

At BCIT, two of the courses I took were Discrete Mathematics and Computer Architecture. Both of those courses got into the Boolean logic and gates. We talked about a few different circuits: adders, multiplexers and decoders, memory circuits, an ALU. Inevitably, one of the tasks I had to do was trace the path a signal took through a circuit and identify which outputs would be asserted. It felt very tedious, it had very strict set of rules, and it was error-prone. Perfect job for a computer to do.

So I started building a program to calculate the output of circuits for me, with some interactive possibilities. This required some way to represent the different circuits to the computer, so I decided to take it farther and make a reusable node API for circuits. After some though about what I wanted it to be able to do, I came up with this set of goals for what the user would be able to do:

* See a graphical representation of each gate and the links
* Indicate which gates/wires were activated
* To pan across the circuit
* To zoom in and out
* To click an input to activate/deactivate it
* To switch between prebuilt circuits via a drop-down menu
* To auto focus (zoom/pan) on the circuit based on the area of interest it covers

Programmatically, I wanted to be able to write a circuit by typing something like

1. andgate = new GateAnd(x, y)
2. input1 = new Input(x, y)
3. input2 = new Input(x, y)
4. output1 = new Output(x, y)
5. Link(input1, andgate)
6. Link(input2, andgate)
7. Link(andgate, output1)

So I set that as my mission and started programming. The resulting function signatures were a little longer, to be able to specify specific gate ports, scale factors, and link path shapes. The actual code equivalent to the above, looks like this:

1. input1 = new GateInput(0, 0, 3, GateState.OFF);
2. input2 = new GateInput(0, 100, 3, GateState.ON);
3. andgate = new GateAnd(150, 50, 3);
4. output1 = new GatePin(300, 50, 3);
5. Gate.connect(input1, -1, andgate, 0, Link.HVH);
6. Gate.connect(input2, -1, andgate, 1, Link.HVH);
7. Gate.connect(andgate, -1, output1, -1);
8. output1.setLabel("Output");
9. output1.setLabelSide(GatePin.NW);

Specifying my goal at the outset really helped me achieve what I wanted, and design something usable. I feel proud of the API I ended up with, and I reused it once already creating the binary tree visualizer.

Download the jar file here