```
<meta http-equiv="Content-Type" content="text/html;charset=utf-8"/>
<!-- <link type="text/css" rel="stylesheet" href="style.css"/> -->
<script type="text/javascript" src="d3.js"></script>
<script type="text/javascript" src="d3.layout.js"></script>
<script type="text/javascript" src="colorbrewer.js"></script>
<style type="text/css">
```

html { background: #222; font-family: sans-serif; } .node circle { cursor: pointer; /*fill: #fff;*/ stroke: steel ; stroke-width: 1.5px; }

.node text { font-size: 16px; }

path.link { fill: none; stroke: #666; stroke-width: 1.5px; }

```
</style>

<div id="body">
  <div id="footer">
    d3.layout.tree
    <div class="hint">click or option-click to expand or collapse</div>
  </div>
</div>
<script type="text/javascript">
```

var m = [20, 120, 20, 120], w = 1280 - m[1] - m[3], h = 800 - m[0] - m[2], i = 0, root;

var tree = d3.layout.tree() .size([h, w]);

var diagonal = d3.svg.diagonal() .projection(function(d) { return [d.y, d.x]; });

var vis = d3.select("#body").append("svg:svg") .attr("width", w + m[1] + m[3]) .attr("height", h + m[0] + m[2]) .append("svg:g") .attr("transform", "translate(" + m[3] + "," + m[0] + ")");

var o = d3.scale.ordinal() .domain([0,0.5]) .range(colorbrewer.YlOrRd[9]);

d3.json('path.json', function(json) { var robot = vis.selectAll('circle').data(json[0]);

robot.enter() .append("svg:circle") .attr("r", 6) .attr('cx', function(d) {return d[0]}) .attr('cx', function(d) {return d[1]}) .style("fill", 'red')

});

d3.json("trellis.json", function(json) { root = json; root.x0 = h / 2; root.y0 = 0; function toggleAll(d) { if (d.children) { d.children.forEach(toggleAll); toggle(d); } }

// Initialize the display to show a few nodes. root.children.forEach(toggleAll); //toggle(root.children[1]);

update(root); });

function update(source) { var duration = d3.event && d3.event.altKey ? 5000 : 500;

// Compute the new tree layout. var nodes = tree.nodes(root).reverse();

// Normalize for fixed-depth. nodes.forEach(function(d) { d.y = d.depth * 180; });

// Update the nodes... var node = vis.selectAll("g.node") .data(nodes, function(d) { return d.id || (d.id = ++i); });

// Enter any new nodes at the parent's previous position. var nodeEnter = node.enter().append("svg:g") .attr("class", "node") .attr("transform", function(d) { return "translate(" + source.y0 + "," + source.x0 + ")"; }) .on("click", function(d) { toggle(d); update(d); });

nodeEnter.append("svg:circle") .attr("r", 1e-6) .style("fill", function(d) { return o(d.value); });

nodeEnter.append("svg:text") .attr("x", function(d) { return d.children || d.*children ? -10 : 10; }) .attr("dy", ".35em") .attr("text-anchor", function(d) { return d.children || d.*children ? "end" : "start"; }) .text(function(d) { return d.name; }) .style('fill', 'white') .style("fill-opacity", 1e-6);

// Transition nodes to their new position. var nodeUpdate = node.transition() .duration(duration) .attr("transform", function(d) { return "translate(" + d.y + "," + d.x + ")"; });

nodeUpdate.select("circle") .attr("r", 4.5) //.style("fill", function(d) { return o(d.value); });

nodeUpdate.select("text") .style("fill-opacity", 1);

// Transition exiting nodes to the parent's new position. var nodeExit = node.exit().transition() .duration(duration) .attr("transform", function(d) { return "translate(" + source.y + "," + source.x + ")"; }) .remove();

nodeExit.select("circle") .attr("r", 1e-6);

nodeExit.select("text") .style("fill-opacity", 1e-6);

// Update the links... var link = vis.selectAll("path.link") .data(tree.links(nodes), function(d) { return d.target.id; });

// Enter any new links at the parent's previous position. link.enter().insert("svg:path", "g") .attr("class", "link") .attr("d", function(d) { var o = {x: source.x0, y: source.y0}; return diagonal({source: o, target: o}); }) .transition() .duration(duration) .attr("d", diagonal);

// Transition links to their new position. link.transition() .duration(duration) .attr("d", diagonal);

// Transition exiting nodes to the parent's new position. link.exit().transition() .duration(duration) .attr("d", function(d) { var o = {x: source.x, y: source.y}; return diagonal({source: o, target: o}); }) .remove();

// Stash the old positions for transition. nodes.forEach(function(d) { d.x0 = d.x; d.y0 = d.y; }); }

// Toggle children. function toggle(d) { if (d.children) { d.*children = d.children; d.children = null; } else { d.children = d.*children; d._children = null; } }

</script>