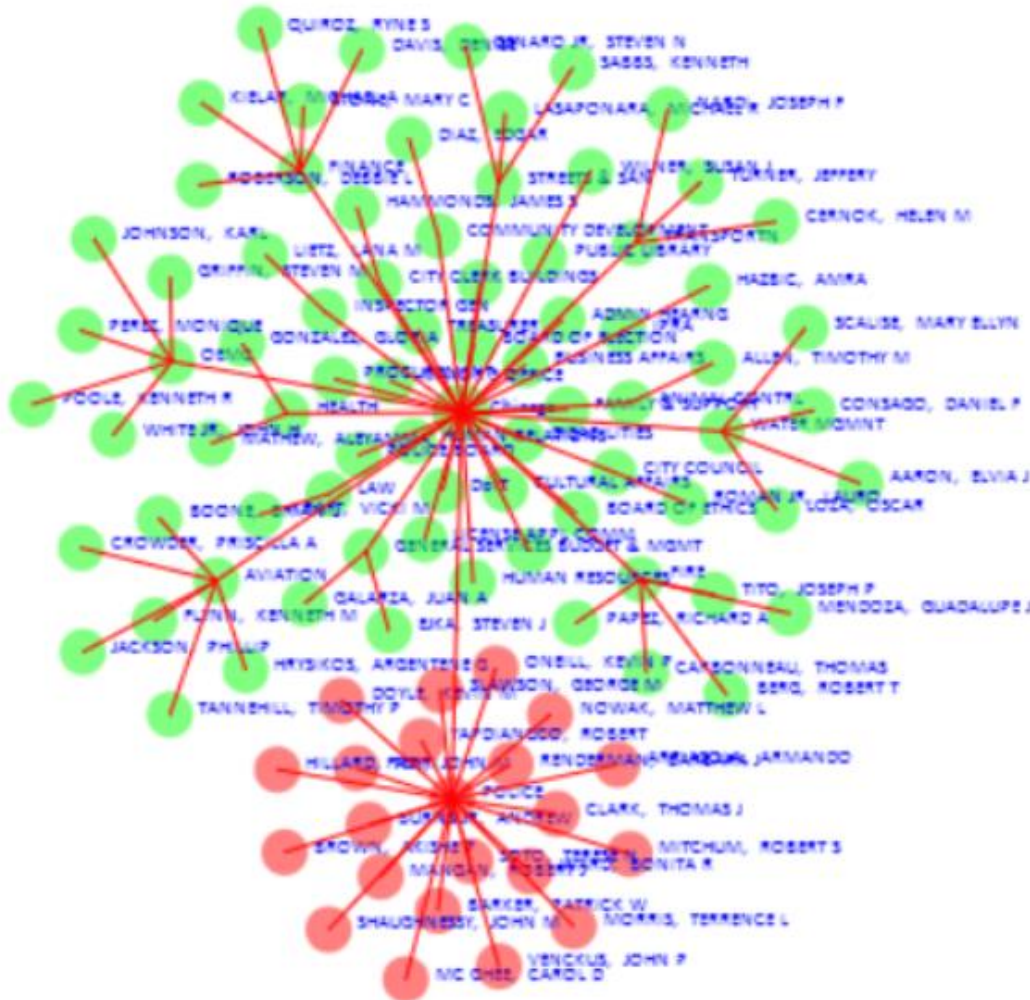


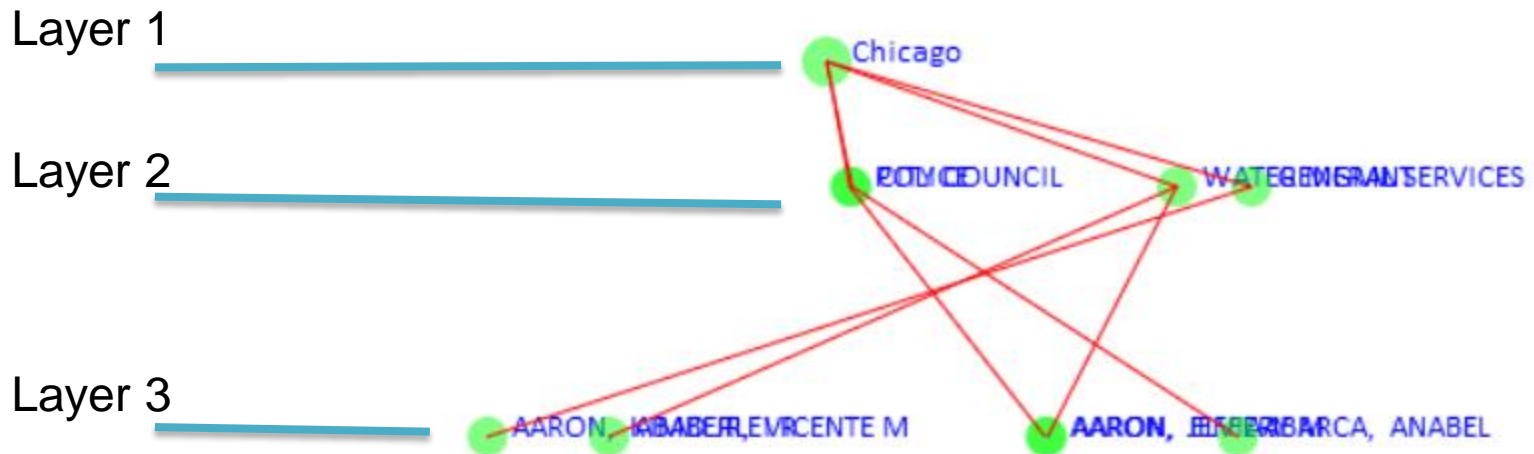
Lecture 29 – Graph Physics (Graphs part 2)

John R. Williams and Abel Sanchez, MIT

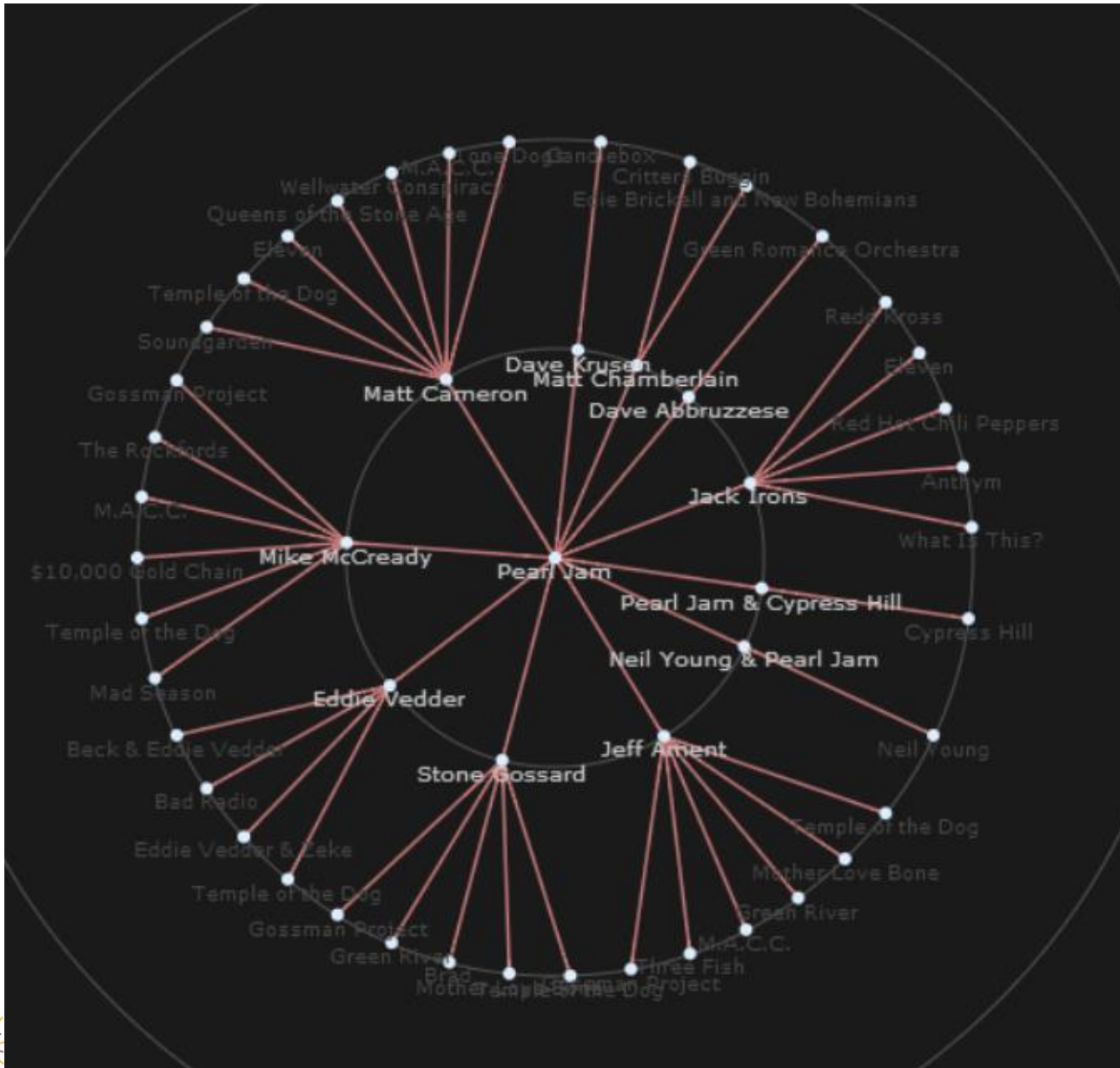


Review of Network Layout Issues

At present we are laying out our nodes in a rather random pattern. Typical strategies might be to lay out in layers as we have done (but with better structure) or on concentric circles.



Concentric Circle Layout



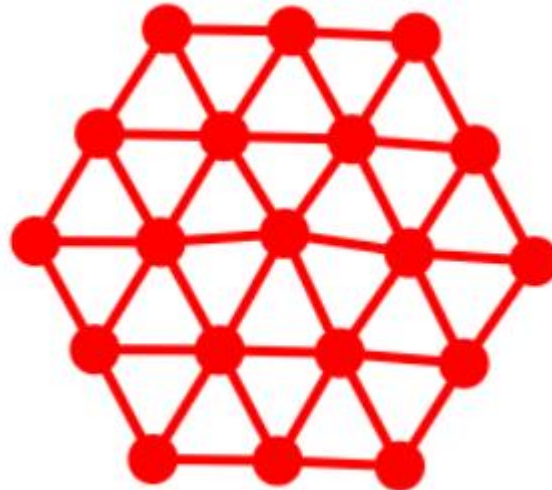
Remember Particle Mechanics

Remember in particle mechanics when we automatically put springs between particles and the result was the particles “self organized” themselves into structures such as below.

The effect of the springs was to keep particles apart and it was controlled by the ‘originalLength’ of the spring.

In our graphs we want to plot the ‘links’ so we might make each link a spring, which we plot.

However, we also want to put in ‘hidden-springs’ that keep particles separate just like the real springs. Lets call these a “repulsive-force”.



In Class Exercise 1

I've provided some code from our Particle Mechanics days so that you can use pieces to put physics into the starter code.

InClassGraphPhysicsEx1 – the starter code (same as we ended up with in last lesson.

Physics Bits - You need to use these to put physics into the starter code. They are taken from our Particle Mechanics days. Here are a few of the bits

```
78 function Node(center,radius){
79   var velocity = Vector(0,0);
80   var force = Vector(0,0);
81   var mass = 1;
82   var children = [];
83   var color = 'Green';
84   var visible = true;
85   var id = null;
86   var updateDisplacements = function updateDisplacements(){
87     var p = this;
88     p.velocity.x = sim.R2/sim.R1*p.velocity.x + (sim.deltaT/sim.R1)* p.force.x/(p.mass);
89     p.velocity.y = sim.R2/sim.R1*p.velocity.y + (sim.deltaT/sim.R1)*(p.force.y - sim.gravity*mass)/(p.mass);
90     //console.log('fx,fy =' +m.force.x+', ' +m.force.y + ' v =' +m.velocity.x+', ' +m.velocity.y);
91
92     // check if either mass will hit the wall
93     checkWallCollision(p);
94
95     // update the position of the masses and draw
96     p.center.x += p.velocity.x *sim.deltaT;
97     p.center.y += p.velocity.y *sim.deltaT;
98     DrawCircle(p.center,3);
99   };
100
101   return {center:center,
102     radius:radius,
103     mass:mass,
104     color:color,
105     id:id,
106     children:children,
107     visible:visible,
108     velocity:velocity,
109     force:force,
110     updateDisplacements: updateDisplacements
111   };
112 }
```

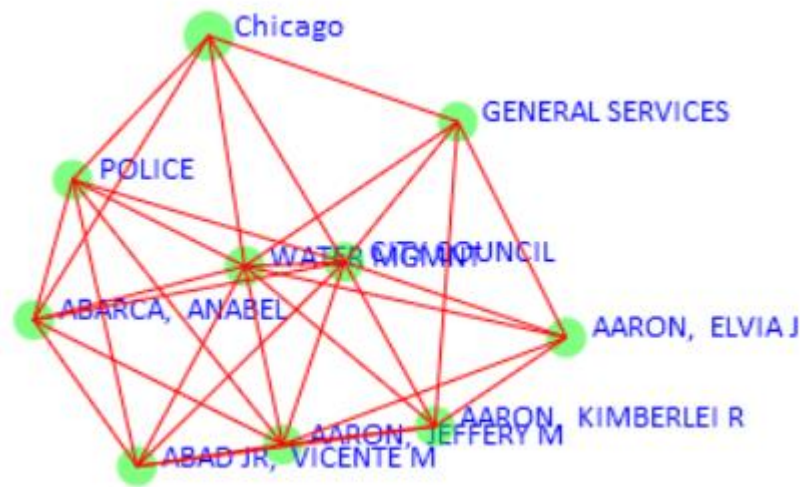
```
14 function RunPhysics(){
15   setInterval(UpdateAll, 10);
16 }
17 function UpdateAll(){
18   // zero all the forces on the masses
19   var n = sim.nodes.length;
20   for(var i=0;i<n;i++){
21     sim.nodes[i].force = Vector(0,0);
22   }
23
24   // update forces due to springs and apply to particles
25   n = sim.links.length;
26   for(i=0;i<n;i++){
27     sim.links[i].updateSpringForces();
28   }
29   // detect neighbors and put springs
30   detectNeighbors();
31
32   // put yield of links in here if needed later
33
34   // update displacements
35   n = sim.nodes.length;
36   for(i=1;i<n;i++){
37     sim.nodes[i].updateDisplacements();
38   }
39   plotAll();
40 }
41 function detectNeighbors(){
42   var n = sim.nodes.length;
43   for(var i=0;i<n;i++){
44     for(var j=i+1;j<n;j++){
45       if(sim.nodes[i].center.distance(sim.nodes[j].center) < sim.interactionDistance){
46         if(noSpring(sim.nodes[i],sim.nodes[j])){ // check if spring exists already
47           var s = Link(sim.nodes[i],sim.nodes[j]);
48           sim.links.push(s);
49         }
50       }
51     }
52   }
53 }
54 function Link(p0,p1){
55   var nodes = [p0,p1]; // these are the particles
56   var originalLength = 20;
57   var color = 'Red';
58   var visible = true;
59
60   var updateSpringForces = function updateSpringForces(){
61     var c0 = nodes[0].center;
62     var c1 = nodes[1].center;
63     var len1 = c1.minus(c0).abs();
64     var forcemag = (len1 - originalLength)*sim.stiffness;
65     color = 'Red';
66     if(forcemag>0)color='Blue';
67
68     var unitVec = c1.minus(c0).unit();
69     if (forcemag > sim.yieldForce ) { // if yieldForce > 0 then
70       forcemag = 0; // this throws away a little energy and damps the system
71       sim.deleteThisSpring.push(this);
72     }
73     var f = unitVec.scale(forcemag);
74
75     nodes[0].force = nodes[0].force.plus(f);
76     nodes[1].force = nodes[1].force.minus(f);
77
78   };
79
80   return {nodes:nodes, color:color, visible:visible,
81     originalLength:originalLength,
82     updateSpringForces:updateSpringForces
83   };
84 }
```

List of things you need.

```
125 function detectNeighbors(){
126     var n = sim.nodes.length;
127     for(var i=0;i<n;i++){
128         for(var j=i+1;j<n;j++){
129             if(sim.nodes[i].center.distance(sim.nodes[j].center) < sim.interactionDistance){
130                 if(noSpring(sim.nodes[i],sim.nodes[j])){ // check if spring exists already
131                     var s = Link(sim.nodes[i],sim.nodes[j]);
132                     sim.links.push(s);
133                     s.color = 'Blue';
134                     //s.visible = false;
135                 }
136             }
137         }
138     }
139 }
140 function noSpring(p0,p1){
141     var n = sim.links.length;
142     for(var i=0;i<n;i++){ // check spring p0 to p1 or p1 to p0
143         if(sim.links[i].nodes[0]===p0 && sim.links[i].nodes[1]===p1) return false;
144         if(sim.links[i].nodes[0]===p1 && sim.links[i].nodes[1]===p0) return false;
145     }
146     return true;
147 }
```


Running physics

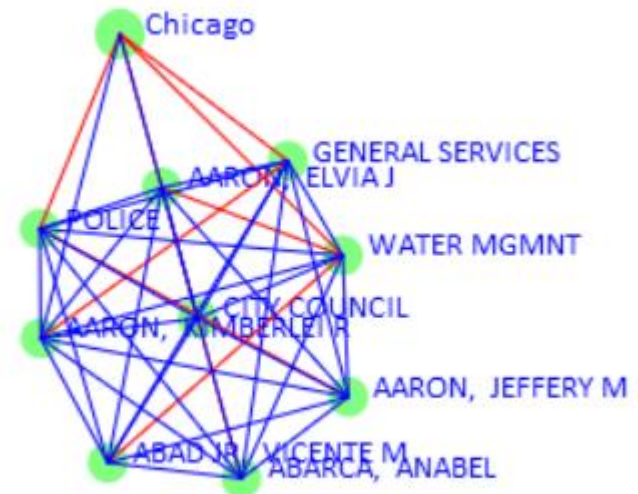
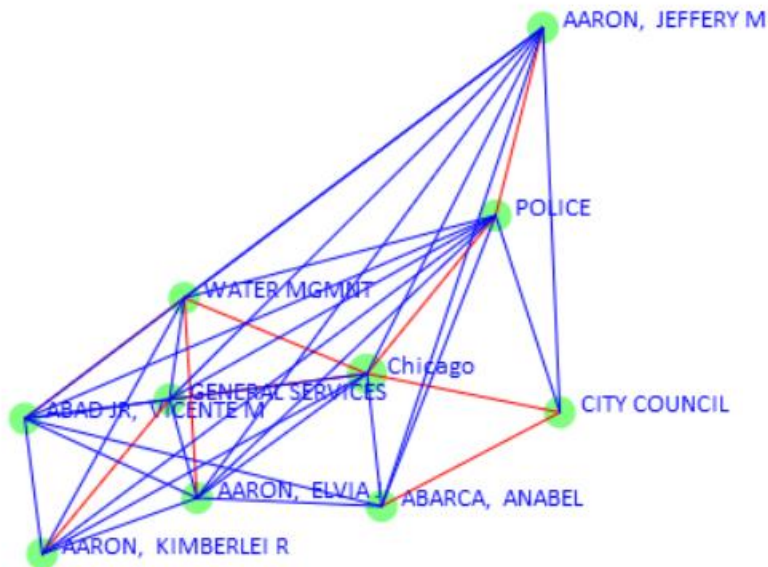
If you plot all the 'new' springs and the 'Links' you get a get something like below when you run physics.



Its not bad but its not great either.

In Class Exercise 2a

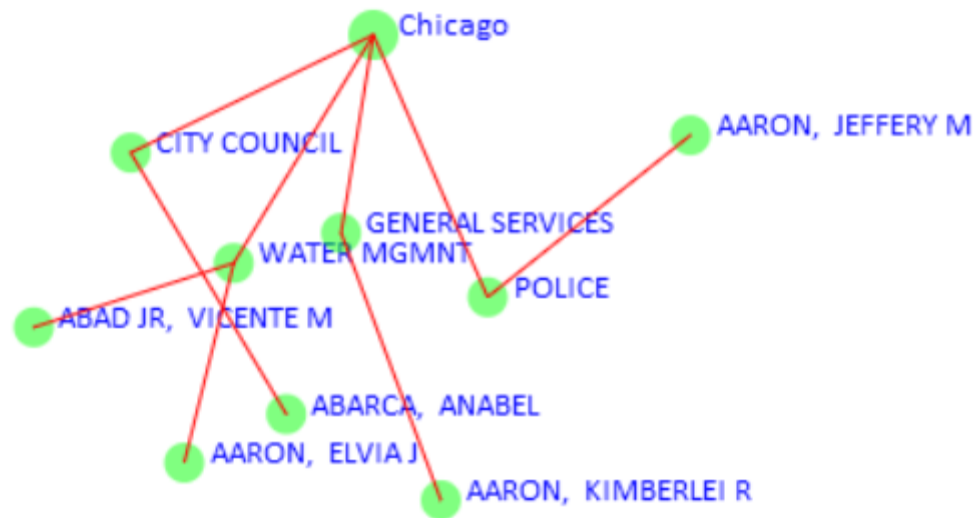
Change the color of the 'new' springs added by physics to 'Blue', leaving the 'old' graph links in red. You should get something like this. This now makes two categories of springs red ones and blue ones.



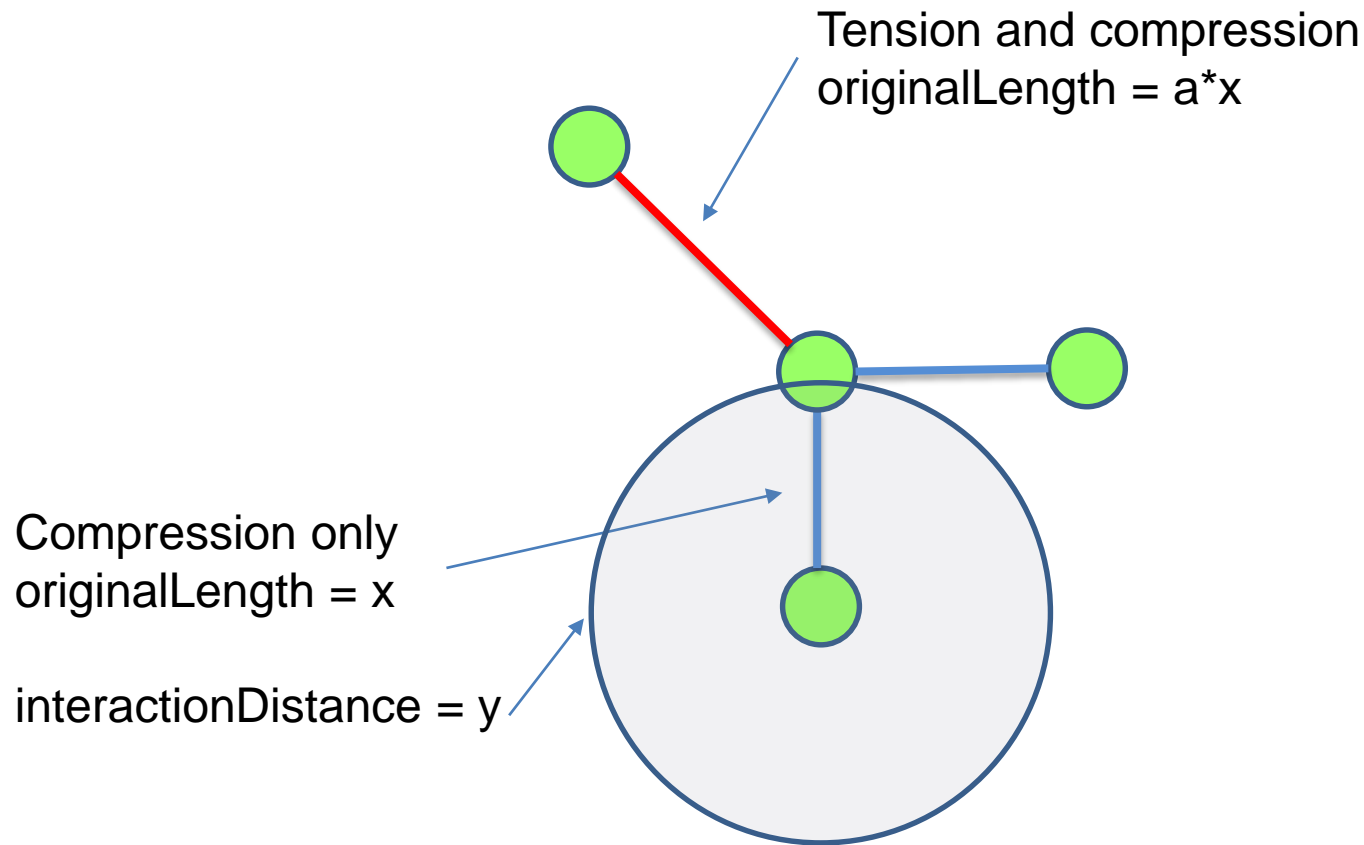
Its still not great.

In Class Exercise 2b

Instead of blue just make the 'new' springs invisible by turning visible = false; for the new springs.



Red and Blue Spring Behavior



In Class Exercise 3

The springs work well to keep the particles apart but they also tend to pull the particles together when they go into tension. What might work better is springs that only work when pushing particles apart but then don't go into tension.

Modify the code so that the behavior of 'Red' springs is unchanged but 'Blue' springs exert zero force once they go into tension.

Once the code is working on the small data set try the larger data set. Change the code in SalaryRipDataLayer.js so that only 1 in 500 employees is read from the data set. (See next slide) Also turn off writing out the names on the nodes so you can see more clearly the overall layout.

Hand in your final code and a screen shots for both data sets.

D:\GitHub\one\john\M29-GraphsPhysicsJSON\InClass29StarterCode\SalaryRipDataLayerSolution.js - S...

File Edit Selection Find View Goto Tools Project Preferences Help

OPEN FILES

- × InClassGraphPhysicsEx1Solution.htm
- × SalaryRipDataLayerSolution.js
- × InClassRandomWalkRev2Solution.htm
- × SpringMassDetectNeighbors.html
- × InClassGraphPhysicsEx1.html
- × spatialReasoning.js
- × salaryGraphTraversal.html
- × modelParams.js
- × PhysicsBits.html
- × untitled
- Vector2D.js

```
5 // 1) find all departments
6 // 2) create nodes and link to department
7 function initDataParse(){
8     var name;
9     var department;
10    var salary;
11    // create the root node
12
13    var anode = Node(rootPos, 10);
14    anode.id = 'Chicago';
15    sim.nodes.push(anode);
16    addDepartmentNodes(anode);
17    var n = data.length;
18    var index;
19    var deptNode;
20    for(var i=0;i<n;i++){
21        name = data[i][8];
22        department = data[i][10];
23        salary = data[i][11];
24        deptNode = findNodeName(department);
25        if(deptNode!=null && parseInt(salary) > 8000){
26            processNode(deptNode,name,3);
27        }
28    }
29 }
30 function findNodeName(name){
31     var n = sim.nodes.length;
32     for(var i=0;i<n;i++){
33         if(sim.nodes[i].id === name)return sim.nodes[i];
34     }
35     return null;
36 }
37
```

DrawCharAt

INSERT MODE, 21 characters selected

Find Find Prev Find All

Tab Size: 2 JavaScript

Result

You should get something like this.

