Force Directed Graph Layout
You will implement the force-directed graph layout algorithm of Fruchterman and Reingold. The original paper can be found at: http://courses.engr.illinois.edu/cs519/rf.pdf

Implementation Requirements:
1. You will use an HTML5 canvas and JavaScript to implement the algorithm.
   Your code should consist of the following files:
   GraphForceLayout.html
   You may base your code off of the code available on GitHub at https://github.com/shaffer1/Ullinois_SciVis/blob/master/MP3/GraphForceLayout.html
   You can grab a copy of GraphForceLayout.html from the repo to serve as starter code.

2. The node positions in the graph are generated iteratively by calculating attractive and repulsive forces. For the vertex \( n_i \) the forces are:

\[
F_a(n_i, n_j) = \frac{||p_j - p_i||^2}{k} (p_j - p_i)
\]

\[
F_r(n_i, n_j) = \frac{-k^2}{||p_j - p_i||} (p_j - p_i)
\]

- \( k \) is a constant typically set to \( k = C \sqrt{\frac{A}{N}} \)
  - \( A \) is the area of the canvas
  - \( N \) is the number of nodes
  - \( C \) is a constant you pick to make things work well...try \( C=1 \) to begin
- \( F_a \) is the attractive force calculated along graph edges
- \( F_r \) is the repulsive force calculated between all pairs of nodes
- \( n_i \) and \( n_j \) are nodes in the graph
- \( p_i \) and \( p_j \) are the positions in space of \( n_i \) and \( n_j \)

3. You may need to cap the maximum movement of a node per iteration as some constant \( t \) and the decrease that by some fraction \( \Delta t \) at each iteration

4. You should animate the layout computation using the techniques described here:
Data

For the graph, use a randomly positioned cycle graph. You can generate an $n$ node random cycle with the following JavaScript:

```javascript
function random_cycle(n,lim_x,lim_y)
{
    var edges=[];
    for(var i=0;i<n;i++)
    {
        edge = {"target":(i+1) % n, "source":i}
        edges.push(edge)
    }
    var nodes=[]
    for(var i=0;i<n;i++)
    {
        var rand_x = Math.random()*lim_x;
        var rand_y = Math.random()*lim_y;
        node = {"x": rand_x, "y": rand_y}
        nodes.push(node)
    }
    var g = {"nodes":nodes,"edges":edges};
    return g;
}
```

In addition, you should include a test of one other graph of your choice. The user interface on the webpage should allow someone to choose which test to run.

Submission

Submit using Compass. Upload the following:

`GraphForceLayout.html`