Bridge SImulator

Interface for Drawing Bridge

Use HTML5 Canvas and Javascript to animate bridge

A Bridge object contains nodes and beams. All interfaces for nodes and beams are accessed through the bridge. A bridge responds to moveForTime:, which moves all beams and nodes
A Beam object represents a beam as a massless spring. A beam does have mass, where a node gets its mass. A beam has a restLength,
Class Node

Bridge Builder will be a website located at hhscp.org/u/leed/bridgeBuilder.html.

User Interface
Drawing a Bridge

When the mouse button goes down on the canvas, the function coord(event, object) finds the location of the click in model coordinates. The function objectAtCoord(coord) finds the object clicked, if any, and returns a list including the object clicked and the point clicked on the object.

Procedure in the mouseDown(event, object) function is determined by subsequent calls to mouseMoved(event, object) and mouseUp(event, object). The results from objectAtCoord(coord(event, object)) are stored in global variables. If the mouse moves more than a certain distance before being released, the gesture becomes the addition of a truss. Otherwise, if the mouse is released without much movement, the gesture is to select the object originally clicked.
For selection, the selected object is indicated by a change in color or size.
For truss addition, the clicked object type matters more. If the object is a node, set the global startingNode to this object. If the object is undefined or a truss, set the startingNode to be a new node at the given coordinate. If the object is a truss, the truss must be deleted with the deleteTruss(truss) function, and the startingNode should be connected with connectNode(node1, node2) to the original endpoints of the deleted truss.
When an object is selected, it can be deleted by pressing the delete key. When a node has been selected, it can be moved by click-and-drag.

Model
The model is composed of three classes.
The Bridge class is used for representing model bridges. A bridge object has an array of nodes and another array of beams. The bridge class encapsulates everything needed to store a bridge. It responds to the following method: update(timestep). It implements update by calling the corresponding update method on each of its nodes sequentially.

The Beam class is used for representing a single truss on a model bridge.
Properties
Beams reference two nodes, one at each endpoint. Beams also are initialized with width, a density, and a stress-strain coefficient.

When a beam is initialized, it sets its k value to the spring constant determined by beam area and stress-strain constant. This k value is used for determining the spring force and the damping force caused by the beam.
Methods

Beams respond to the methods area(), length(), and volume().

Area is calculated by squaring the beam's width.

Length is calculated by calling distanceToNode(other) on one of the beam's nodes, passing the other node as the parameter. Upon initialization, a beam stores its current length in startingLength.

Volume is calculated by multiplying area by length. A beam's mass is set by multiplying density by initial volume. When a beam's mass is set, it increases the mass of each of its nodes by half of the new mass. This is important because, for simulation, nodes move based on forces applied to their own mass, regardless of the masses of beams.

A Beam object responds to the method force() by multiplying its spring constant value by its extension, which is length()-startingLength. A beam's force is a scalar quantity and is negative for compression forces and positive for tension forces.

A beam also responds to the convenience method otherNode(node), which is given one of its nodes and returns the other.

The Node class represents the intersection of bridge beams.
Properties
A node has an array of beams which connect to it.
A Node has coordinates x and y as well as velocity vx and vy. A node also has a weight, which is a constant force pulling down and initializes the node's mass as weight/GRAVITY.
Methods

A node responds to the method distanceToNode(other) by finding the length of the hypotenuse of a right triangle which has a node at the end of each leg.

A node responds to the method velocity() by using the Pythagorean theorem on its component velocities to return a scalar representing the magnitude of the node's velocity.

A node responds to the method directionToNode(other) by using Math.atan2() on the differences in the y and x coordinates of the two nodes.

A node responds to the method netForce() by adding the forces and dampening forces of each beam, separated using directionToNode(beam.otherNode(this)) into component forces fx and fy. The method netForce() then returns a dictionary {"x":fx, "y":fy-weight}.

A node responds to the method update(timestep). First the net force on the beam is calculated with netForce(). Then this force is used to calculate acceleration in the x and y directions by dividing the force by the node's mass. Acceleration is then used to change the node's velocity. Acceleration and velocity are used to change the node's position.