Mass-Spring Locomotion

An implementation in Processing by Kaylah Facey

Controls

The buttons to the right of the simulation control the simulation. Some of the buttons have equivalent keyboard commands:

- "play/pause" or the SPACE key: Play/pause
- "step" or 's': Advance the simulation by 1 timestep.
- "reset" or 'r': Reset the simulation
- Creature checkboxes: Click these checkboxes to select the creature to simulate.

The Simulation

Point Masses and Springs

The core of the locomotion simulator is a mass-spring system. Each spring uses the same spring constant and damping constant, and each point mass has the same mass. Springs act on masses with the following equations (from Sticky Feet: Evolution in a Multi-Creature Physical Simulation.):

- a.p and b.p are position vectors for a and b.
- a.v and b.v are velocity vectors for a and b.
- P = a.p - b.p; V = a.v - b.v
- ks = spring constant; kd = damping constant; L = rest length
- spring force Fs = -ks(|P| - L)
- damping force Fd = -kd(V . P) / |P|
- damped spring force Fds(a) = (Fs + Fd) * P/|P|
- Fds(b) = -Fds(a)

Each spring adds its damped force to the gravitational force for each mass it is connected to.

With only damped spring forces and gravity acting on the creatures, they would oscillate briefly and come to a stop. To use the mass-spring system for *locomotion*, more was required:

Oscillation

Firstly, I added oscillators to some springs. Oscillators affect the rest length of springs as follows (from Sticky Feet: Evolution in a Multi-Creature Physical Simulation.):

- Lbase = the base rest length
- amplitude a = the % by which the rest length changes
- frequency f = the speed at which the rest length changes
- phase p = an offset to the oscillator's cycle, allowing two springs with the same L, a, and f to oscillate at slightly different times
- time t = the current timestep
- Lnew = Lbase(1 + asin(ft + 2PIp))

Adding oscillation was the beginning of interesting-looking movement for my virtual creatures, but to get the forward motion I wanted, I added a few more things to my simulation.

The Creatures

The Whirligig and The Pinwheel

The first creature I designed was a pinwheel with 6 spokes, christened "The Whirligig". With oscillation, I was able to get it to roll forward for a few turns, but because the spokes rotate to different positions on the wheel, and spokes at the top move at the same time as spokes on the ground, the creature would unbalance itself and tip backwards in the opposite direction.

I solved this problem by adding rest periods to the oscillators, which works as follows:

- active time at = time the oscillator spends changing the rest length before resting
- rest time rt = time that the oscillator spends at rest before resuming its affects on the rest length
- phase p = an offset that defines the starting point in the active-rest cycle for a given oscillator

In this way, the oscillator cycles between active and resting. To determine whether an oscillator is currently active or at rest:

- cycle time ct = at + rt
- position in cycle pc = (t + p) % ct

- if pc < at then the oscillator is active. Otherwise, the oscillator is at rest.

Using active and rest times for the oscillators in the whirligig, I was able to tune its springs so that each spring in turn pushes the whirligig forward without unbalancing it. Permitting oscillating springs to rest allows greater control over a simulated creature's movements without using hard-coded animation frames.

Once "The Whirligig" was moving satisfactorily, I was able to, with minimal tweaks, extend the same system for a 10-spoked creature I christened "The Pinwheel".

The Inchworm

For my other creature, I designed an accordian-shaped creature that extends and contracts to crawl along the floor like a snake or worm. Initially the creature, christened "The Inchworm", simply expanded and contracted in place, without moving forward. Its problem was a lack of friction, and I added a friction force to the point masses in the simulation. Each point mass has a friction "oscillator" that determines whether the friction is set to the min value or the max value. I found that setting friction to toggle rather than slowly increase and decrease had a nicer-looking movement. Friction is determined as follows:

- min = the min friction coefficient
- max = the max friction coefficient
- frequency f = how often the coefficient switches from min to max and back
- If cos((f * t) + p) > 0 then the value of the friction coefficient would be increasing if it were oscillating slowly (cos is the derivative of sin). In this case, the friction coefficient is set to max. Otherwise, it is set to min.

The friction force on the point masses is calculated using the resulting friction coefficient kf as follows:

- First, determine if the point mass is touching the ground (we only consider friction against the ground in the x direction)
- If the point mass is touching the ground, friction is applied as a damping force against the x direction of the velocity of the point mass.
- Let v be the velocity; then the friction force ff is -kf * v.x
- Let F be the current forces on the point mass.
- If |ff| > |F.x| set ff = -F.x (friction should not send a point moving in the opposite direction)
- Add ff to F.x.

With friction "The Inchworm" is able to inch forward. Many simulated creatures can achieve forward momentum and realistic-looking motion with a friction force, as demonstrated by Sticky Feet: Evolution in a Multi-Creature Physical Simulation.