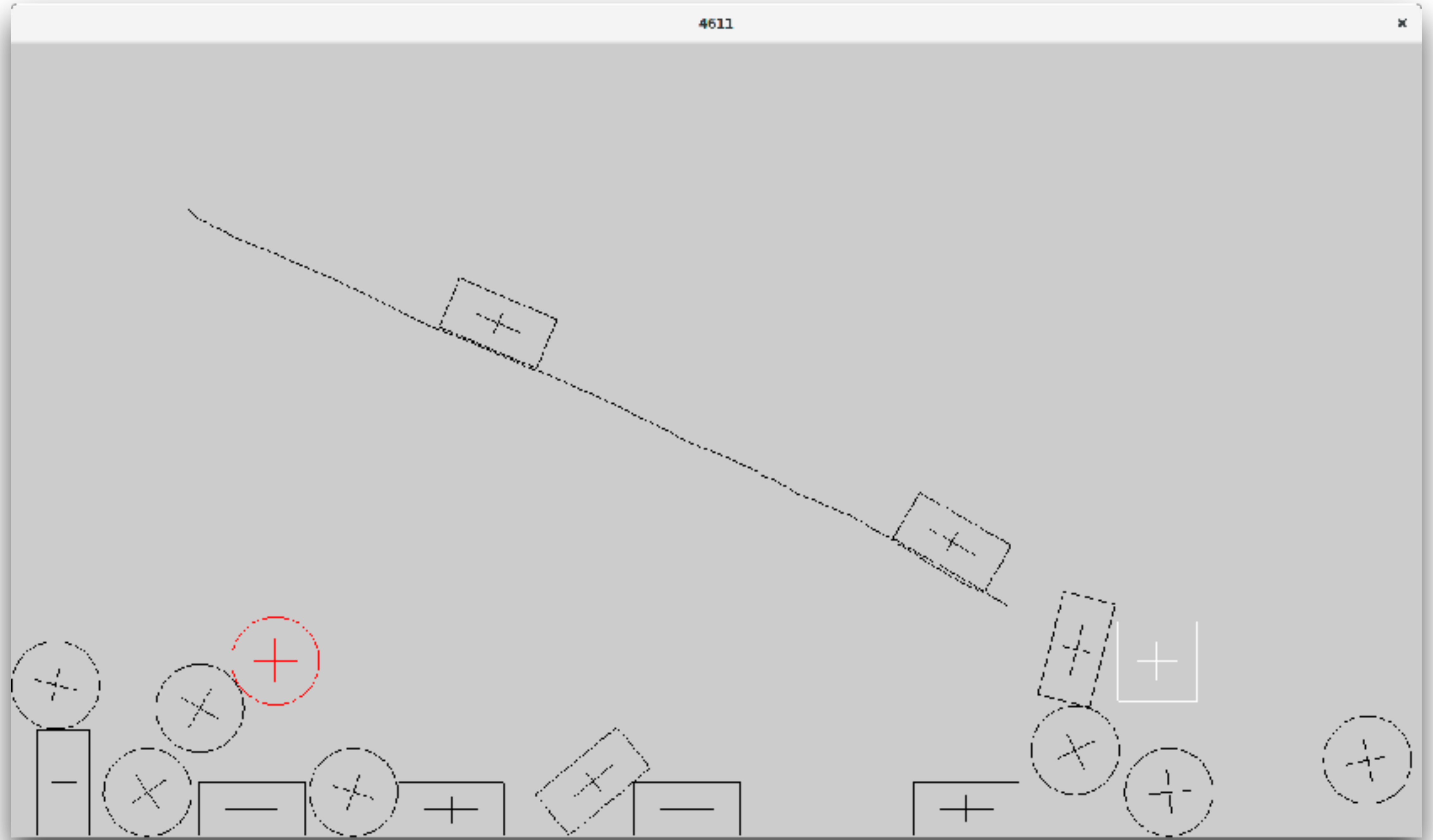
CSCI 4611

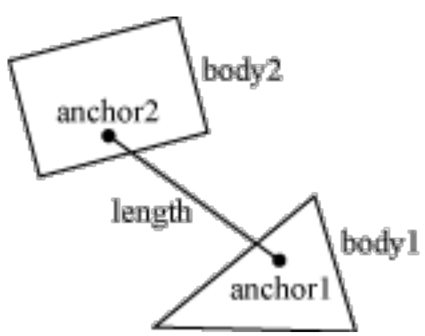
Particles and mass-spring systems

Assignment 6 demo

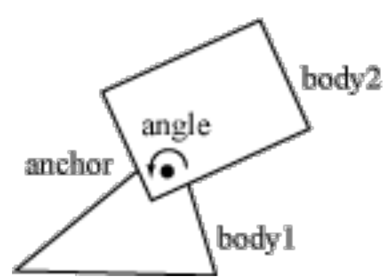


Box2D joints

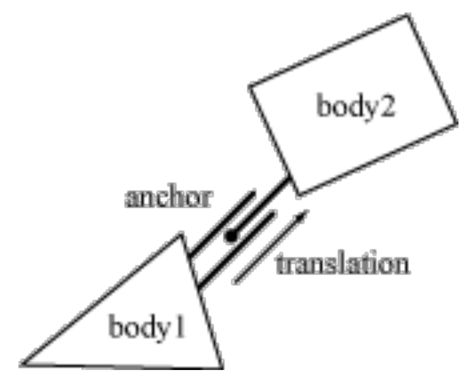
Joints connect two rigid bodies together. Box2D has lots of useful ones:



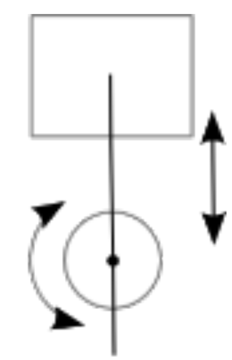
distance



revolute



prismatic



wheel

Specified by two bodies, anchor point(s), [other params]

Box2D joints

Creating a joint is similar to creating anything else in Box2D:

```
b2RevoluteJointDef jointDef;
jointDef.Initialize(
  bodyA, bodyB, anchorPoint);
b2RevoluteJoint *joint =
  (b2RevoluteJoint*)
    world->CreateJoint(&jointDef);
```

Note: The mouse joint doesn't have an `Initialize()` method.
Set members `bodyA` etc. directly.

Particles and mass-spring systems

Particles

A particle is just a point mass.

- Position $\mathbf{x}$
- Velocity $\mathbf{v}$
- Mass m

(Like the ball in Assignment 2, except “smaller”)

Particles

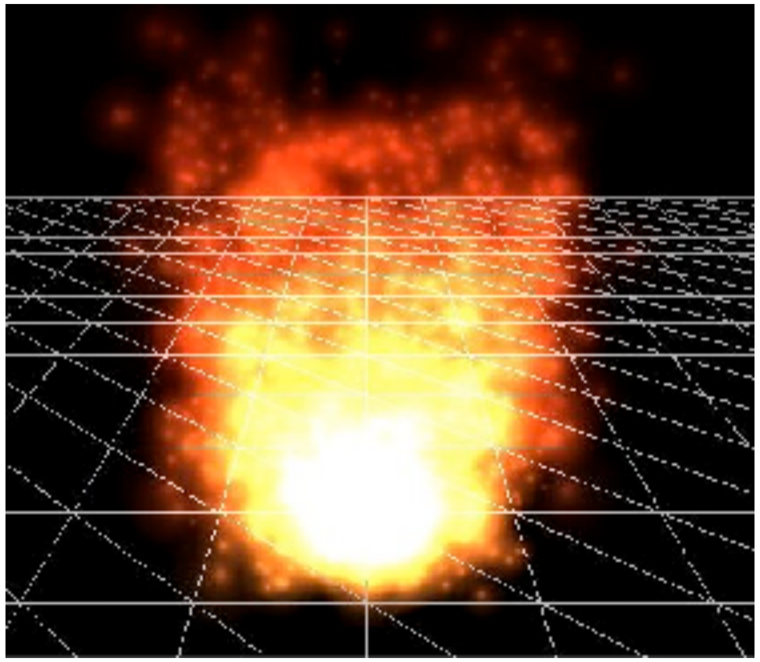
A particle can be acted upon by multiple forces (gravity, springs, repulsions, collisions, …)

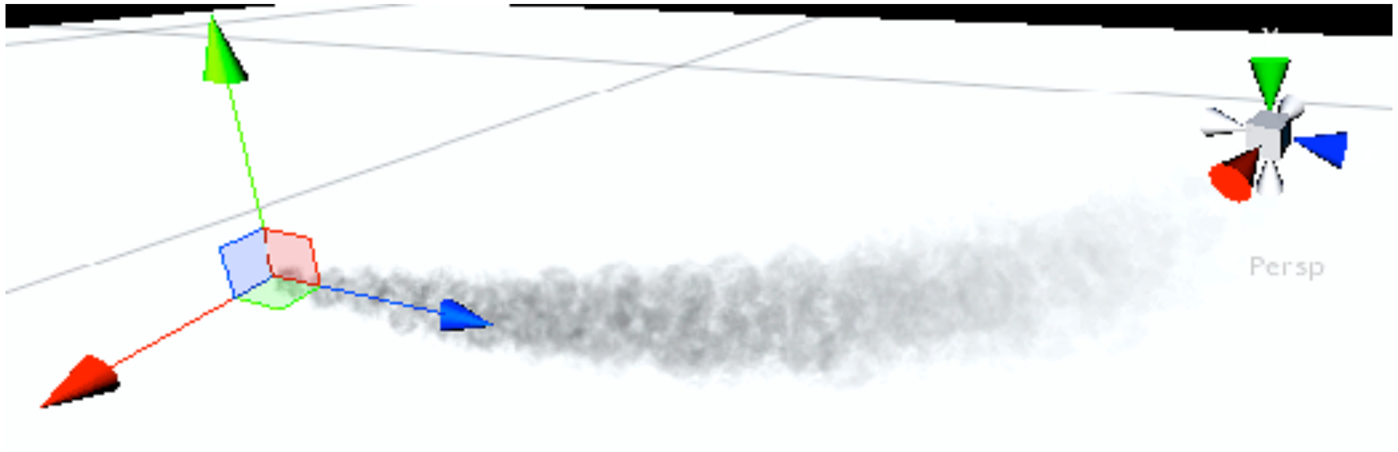
- Total force $\mathbf{f} = \mathbf{f}_1 + \mathbf{f}_2 + \mathbf{f}_3 + \cdots$
- Acceleration $\mathbf{a} = \mathbf{f}/m$
- Equations of motion:

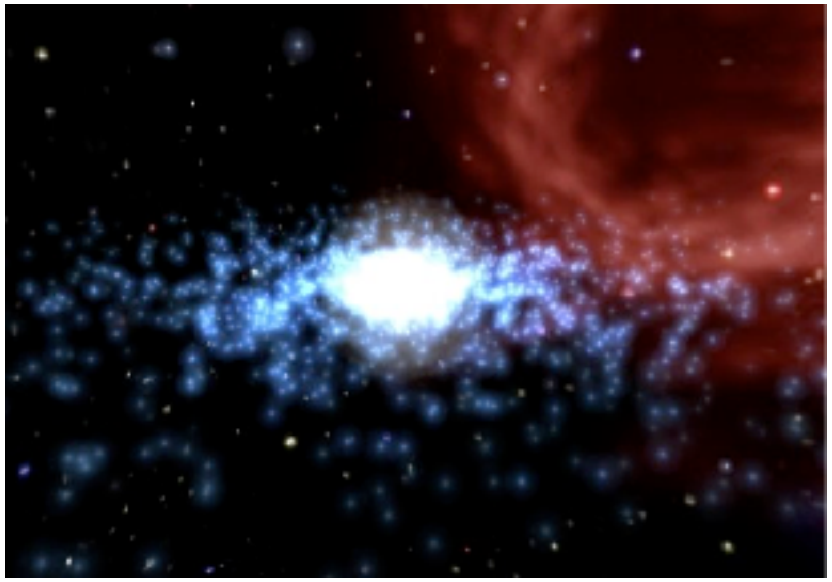
$$\mathrm{d}\mathbf{v}/\mathrm{d}t = \mathbf{a}$$
$$\mathrm{d}\mathbf{x}/\mathrm{d}t = \mathbf{v}$$

Particle systems

Create lots of particles with randomized initial state,
apply some ad-hoc forces







Some forces

- Gravity: $\mathbf{f} = m\,(0, -9.8)$
- Buoyancy: $\mathbf{f} = (T - T_{\text{air}})\,(0, 1)$
- Drag: $\mathbf{f} = -k_d\,\mathbf{v}$
- Repulsion-based collisions: If particle is at depth d inside object, apply force $\mathbf{f} = k_r\,d\,\mathbf{n}$

Time stepping

For each particle:

- Compute total force $\mathbf{f} = \mathbf{f}_1 + \mathbf{f}_2 + \mathbf{f}_3 + \cdots$
- Update velocity $\mathbf{v} \mathrel{+}= \mathbf{f}/m \, \Delta t$
- Update position $\mathbf{x} \mathrel{+}= \mathbf{v} \, \Delta t$

Mass-spring systems

Mass-spring systems

So far, the particles don't interact, each one moves independently. So it behaves more like a gas than a solid.

Connect particles to each other with springs:

Each spring applies a force on two particles.

Spring forces

Springs:

- Rest length ℓ_0
- Spring constant k_s
- Particles p_1, p_2

$$\ell = ||\mathbf{x}_2 - \mathbf{x}_1||$$

$$\mathbf{e} = \mathrm{normalize}(\mathbf{x}_2 - \mathbf{x}_1)$$

$$\mathbf{f}_2 = -k_s\,(\ell - \ell_0)\,\mathbf{e}$$

$\mathbf{f}_1$ is equal and opposite

Spring forces

Springs:

- Rest length ℓ_0
- Spring constant k_s
- Damping coefficient k_d
- Particles p_1, p_2

$$\ell = \|\mathbf{x}_2 - \mathbf{x}_1\|$$

$$\mathbf{e} = \mathrm{normalize}(\mathbf{x}_2 - \mathbf{x}_1)$$

$$v_{\mathrm{rel}} = (\mathbf{v}_2 - \mathbf{v}_1) \cdot \mathbf{e}$$

$$\mathbf{f}_2 = -k_s\,(\ell - \ell_0)\,\mathbf{e} - k_d\,v_{\mathrm{rel}}\,\mathbf{e}$$

$\mathbf{f}_1$ is equal and opposite

Spring implementation

Create a Spring class that has these members:

- Rest length (`float`)
- Spring constant (`float`)
- Damping coefficient (`float`)
- Two connected particles (`Particle*`)

…and a `computeForce()` method that computes the force on the second particle (but doesn’t do anything with it yet)

Time stepping

For each particle:

- Compute total force $\mathbf{f} = \mathbf{f}_1 + \mathbf{f}_2 + \mathbf{f}_3 + \cdots$
- Update velocity $\mathbf{v} \mathrel{+}= \mathbf{f}/m \, \Delta t$
- Update position $\mathbf{x} \mathrel{+}= \mathbf{v} \, \Delta t$

Time stepping

For each particle:

- Initialize force $\mathbf{f} = m\mathbf{g}$
- For each spring connected to particle:
  - Compute force due to spring, then update $\mathbf{f} \mathrel{+}= \mathbf{f}_{\text{spring}}$
- Update velocity $\mathbf{v} \mathrel{+}= \mathbf{f}/m\,\Delta t$
- Update position $\mathbf{x} \mathrel{+}= \mathbf{v}\,\Delta t$

Time stepping

For each particle:

- Initialize force $\mathbf{f} = m\mathbf{g}$

For each spring:

- Compute force due to spring
- Update $\mathbf{f}$ of connected particles

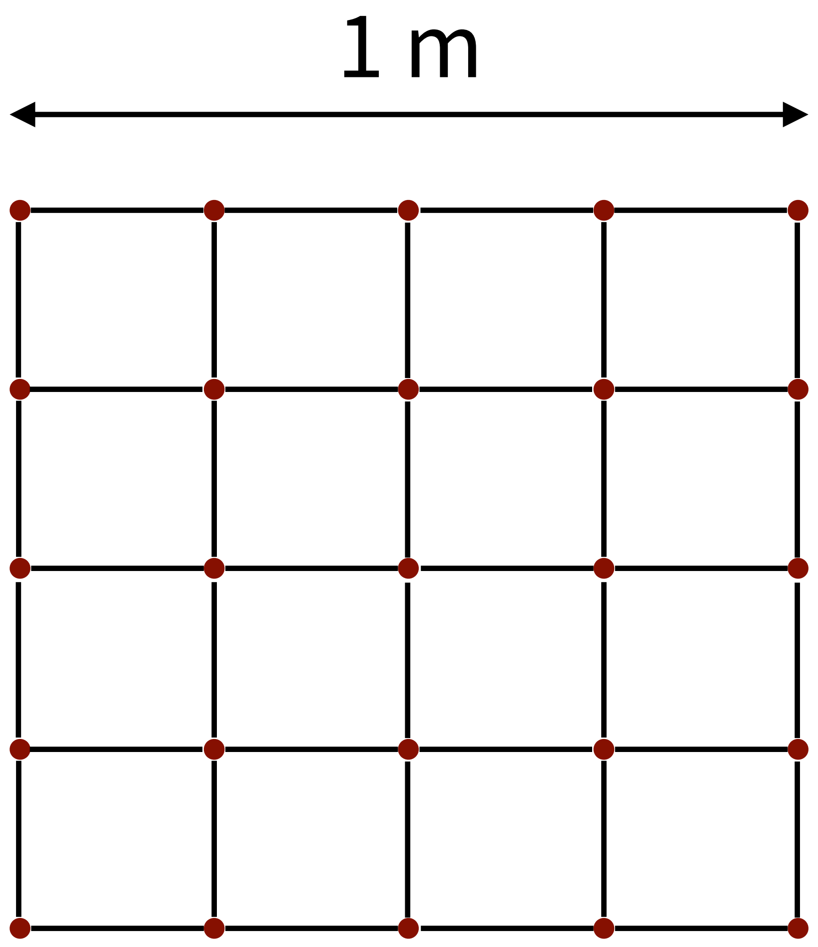
For each particle:

- Update velocity $\mathbf{v} \mathrel{+}= \mathbf{f}/m\, \Delta t$
- Update position $\mathbf{x} \mathrel{+}= \mathbf{v}\, \Delta t$

Mass-spring cloth

Create an $(n+1)\times(n+1)$ grid of particles, connected with springs.

- Connect particle (i, j) to $(i+1, j)$ and to $(i, j+1)$
- Rest length = $1/n$



We will see two problems…

Instability

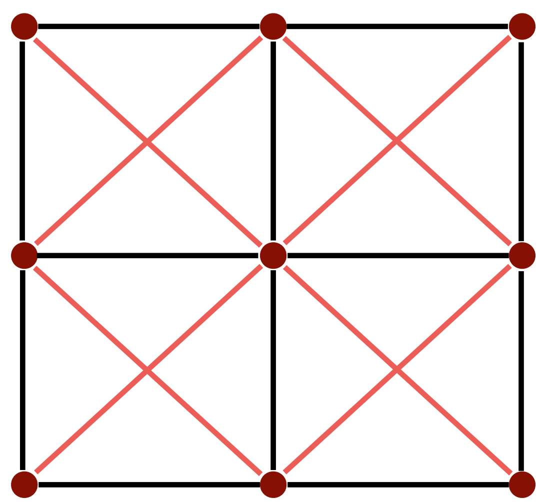
If the forces are too stiff, the system blows up.
(Accelerations vary too much in a single time step)

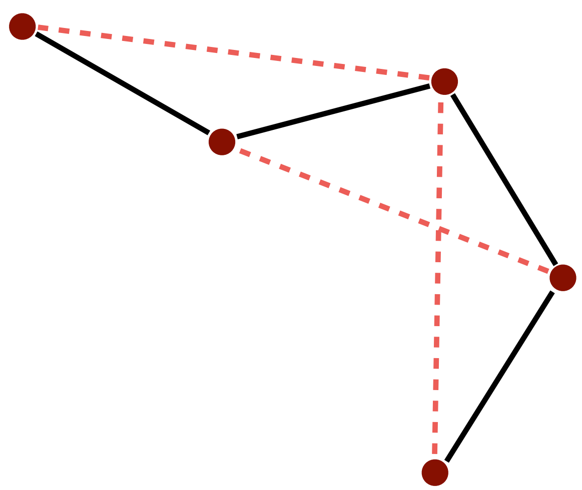
- **Solution:** Reduce the time step.
  Each frame, instead of taking a single step of length Δt, take n substeps of length $\Delta t/n$. (Try $n = 10$.)

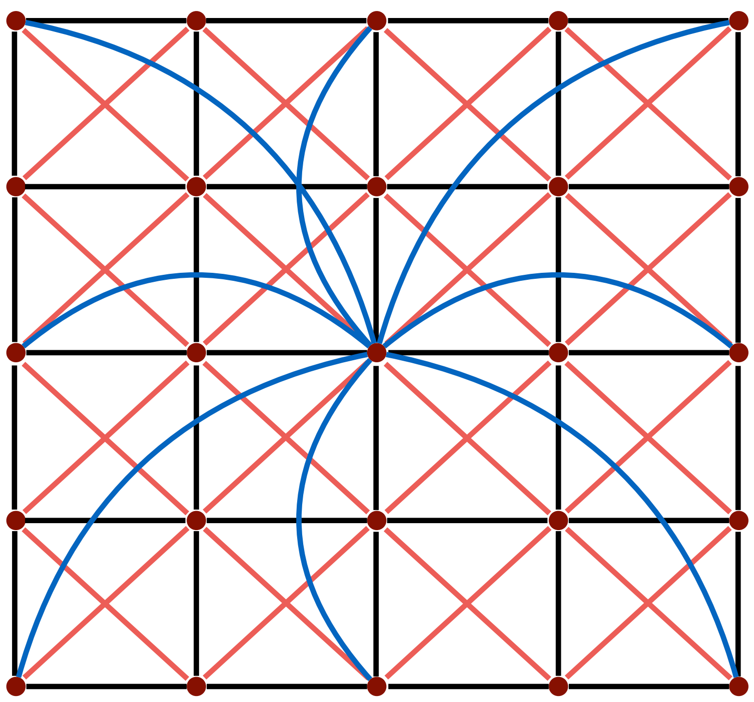
- Other solution: Use a more robust integration scheme. But this generally requires solving a system of nonlinear equations…

Extra springs

Need to add springs for all desired behaviors (shear, bending, …)





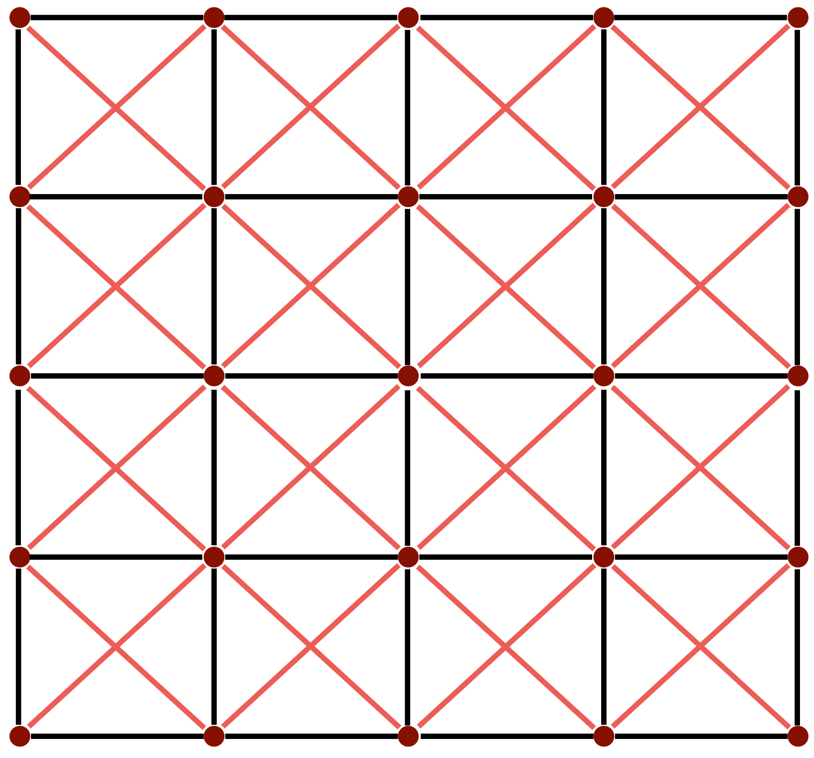


(Hard to control stretching, shearing, bending resistance independently)

Cloth springs

Shear springs:

- Connect (i, j) to $(i+1, j+1)$ and $(i+1, j-1)$
- Rest length is different!
- Stiffness and damping should be lower than of structural springs



Learn more

A classic intro to physics-based animation:

Witkin and Baraff,
"Physically Based Modeling: Principles and Practice",
SIGGRAPH 1997 course notes
http://www.cs.cmu.edu/~baraff/sigcourse/