Computer Animation

Sylvia Pan

March 2013

Lecture slides heavily based on previous versions produced by Marco Gillies

Computer Animation

- videos\SIGGRAPH 2011 Computer Animation Festival Video Preview.flv

- If we have internet:

Siggraph Computer Animation Festival 2012

This course will:

- Outline the major techniques used in animation
- Discuss general animation, physical simulation, and character animation
- Go into detail on a few key methods

Books

- “Computer Animation – algorithms & techniques” Rick Parent
- “Real-time 3D Character Animation” Nik Lever – Focal Press
  - Very good, but only handles characters
- “Advanced Animation and Rendering Techniques: Theory and Practice” Alan Watt and Mark Watt – Addison-Wesley
  - More general but sometimes hard to follow

Everything you need is in the slides

Overview

- Animation:
  - Animate: bring to life
  - The rapid display of a sequence of images to create an illusion of movement

- Computer Animation:
  - A key aspect of computer graphics
  - Generate the sequence of images with computers

Computer Animation: Applications

- Entertainment
  - Films (non-real-time) and Games (real-time)
- Commercial
  - Advertisement, online agents (ikea agent)
- Simulation
  - Training (driving, flight simulation), Scientific data visualisation
- Medical Animation
  - Medical training, education, research

Course Outline

- Physical Simulation
  - Particles, spring, rigid bodies
  - Particle systems
- Traditional animation
- Key frame and interpolation
- Character animation
  - Body and face
  - Behaviour simulation

Physical Simulation

Particles

- Particle: a small object (cannon ball, or a particle in a cloud)
- Ignore the size and rotational dynamics of the object

Kinematics and Dynamics

- “Kinematics”: when we’ve just used velocities and positions (motion)
- “Dynamics”: when we deal with forces and accelerations (cause of motion)

Physical Simulation (Dynamical Simulation)

- In computer animation also called physically based animation
- Application: smoke, fluid, explosions, car collision, cloth…
- Animators are concerned with general quality of the motion rather than precise location and orientation of each object
- May not be physically accurate
- Believability (physical realism)

Physical Simulation

- The motion of objects is governed by the laws of physics
- Physical Simulation is about finding the physics behind the movement of objects (and decide which forces to include)
- Those laws of physics can be expressed using mathematical formulae

Newton's Second Law

- The basis of physical simulation is Newton's second law: the net force applied to a body produces a proportional acceleration.

$$\vec{f} = m\vec{a}$$

Particle Simulation with Newtown's Second Law

- Simply two steps:
  - Work out the forces acting on an object
    - Gravity
    - Impulse forces (explosions or collisions)
    - Forces from springs, friction, other objects
  - Use Newton's law to work out the position of an object at each time step

Example: an object with a initial velocity under gravity

$$\vec{f} = m\vec{a}$$

$$\vec{a} = \frac{d\vec{v}}{dt}$$

$$\vec{v} = \frac{d\vec{p}}{dt}$$

$$\Delta t \to 0 \Longrightarrow$$

$$\vec{a} = \frac{\vec{f}}{m}$$

$$\vec{v}_{t+\Delta t} = \vec{v}_t + \Delta t\vec{a}$$

$$\vec{p}_{t+\Delta t} = \vec{p}_t + \Delta t\vec{v}_t$$

Example: an object with a initial velocity under gravity

- A ball launched from the ground under gravity
- Time step:

$$\Delta t = 0.1s$$

- Initial conditions:

$$f = [0, -9.81m, 0]$$

$$v_0 = [10, 10, 0]$$

$$p_0 = [0, 0, 0]$$

Example: an object with a initial velocity under gravity

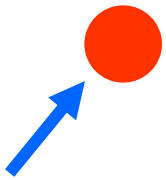
$$t = t_0$$

$$a = [0, -9.81, 0]$$

$$v_0 = [10, 10, 0]$$

$$p_0 = [0, 0, 0]$$

Example: an object with a initial velocity under gravity



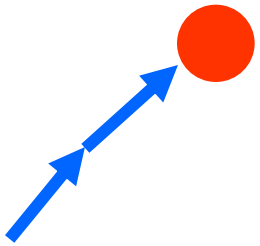
$$t = t_1$$

$$a = [0, -9.81, 0]$$

$$v_1 = [10, 10 - 0.981, 0]$$

$$p_1 = [1, 1, 0]$$

Example: an object with a initial velocity under gravity



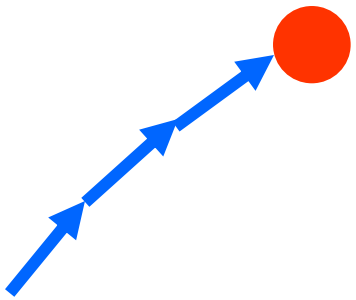
$$t = t_2$$

$$a = [0, -9.81, 0]$$

$$v_2 = [10, 8.0380, 0]$$

$$p_2 = [2, 1.9019, 0]$$

Example: an object with a initial velocity under gravity



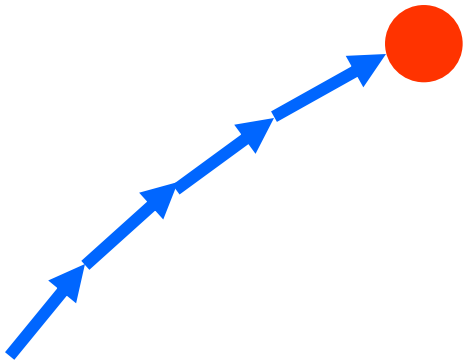
$$t = t_3$$

$$a = [0, -9.81, 0]$$

$$v_3 = [10, 7.0570, 0]$$

$$p_3 = [3, 2.7057, 0]$$

Example: an object with a initial velocity under gravity



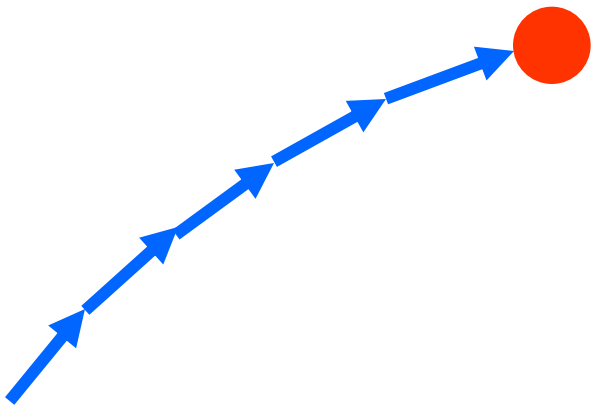
$$t = t_4$$

$$a = [0, -9.81, 0]$$

$$v_4 = [10, 7.0570, 0]$$

$$p_4 = [4, 3.4114, 0]$$

Example: an object with a initial velocity under gravity



$$t = t_5$$

$$a = [0, -9.81, 0]$$

$$v_5 = [10, 5.0950, 0]$$

$$p_5 = [5, 4.0190, 0]$$

What if we use a bigger time stamp?

- A ball launched from the ground under gravity
- Time step:

$\Delta t = 0.5s$

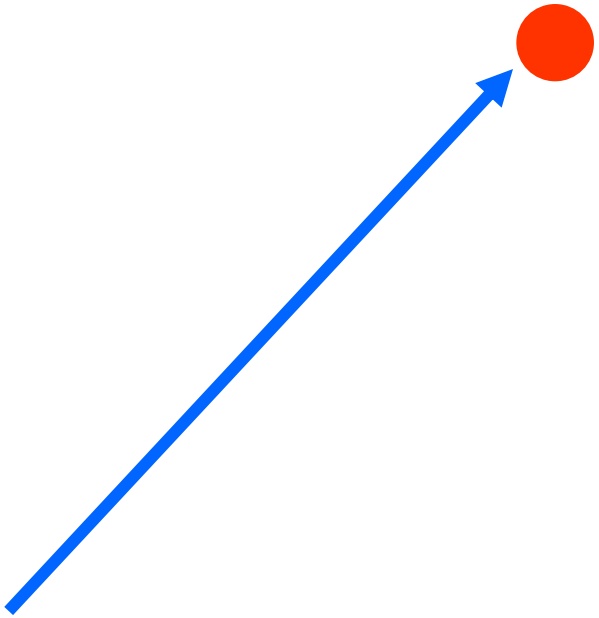
- Initial conditions:

$$f = [0, -9.81m, 0]$$

$$v_0 = [10, 10, 0]$$

$$p_0 = [0, 0, 0]$$

Example: an object with a initial velocity under gravity



$$t = t_1$$

$$a = [0, -9.81, 0]$$

$$v_1 = [10, 5.0950, 0]$$

$$p_1 = [5, 5, 0]$$

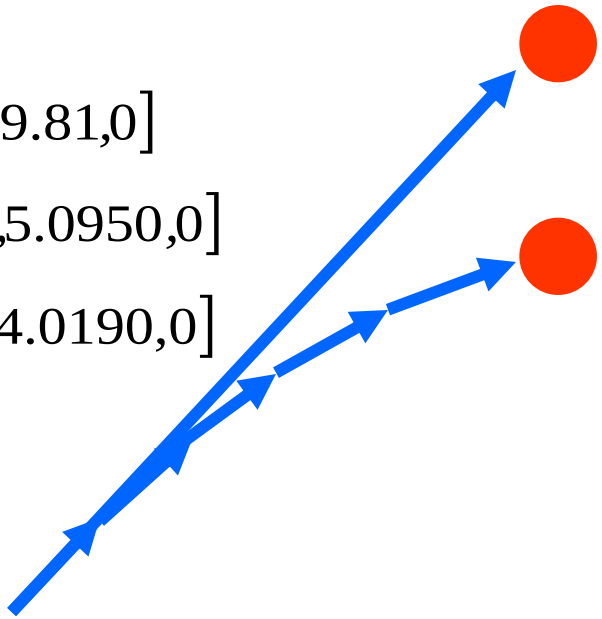
Example: an object with a initial velocity under gravity

$t = t_5$

$a = [0,-9.81,0]$

$v_5 = [10,5.0950,0]$

$p_5 = [5,4.0190,0]$



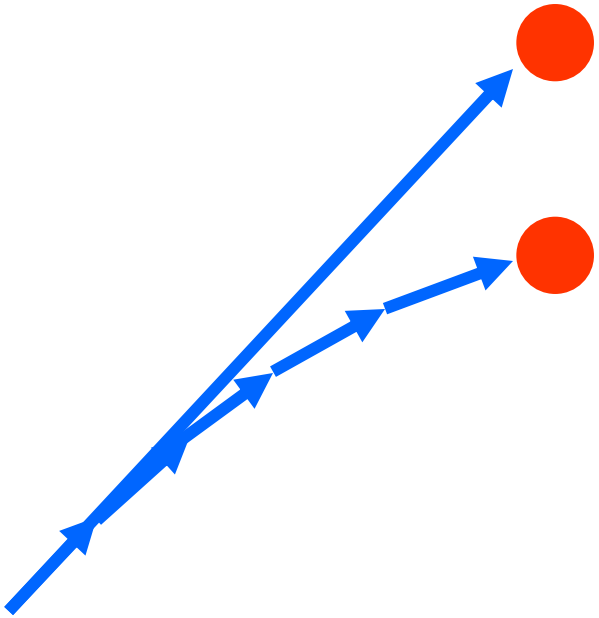
$t = t_1$

$a = [0,-9.81,0]$

$v_1 = [10,5.0950,0]$

$p_1 = [5,5,0]$

Example: an object with a initial velocity under gravity



- The accuracy is highly sensitive to the step sizes!

Example: an object with a initial velocity under gravity

- Euler's Method:

$$\vec{a} = \frac{\vec{f}}{m}$$

$$\vec{v}_{t+\Delta t} = \vec{v}_t + \Delta t \vec{a}$$

$$\vec{p}_{t+\Delta t} = \vec{p}_t + \Delta t \vec{v}_t$$

Euler's Method

- Initial Value Problem:

$$\frac{dy}{dx} = f(x, y)$$
$$y(0) = y_0$$

- Euler's Method gives a first-order numerical solution

$$y_{t+\Delta t} = y_t + f(x_t, y_t)\Delta t$$

Maths

- Taylors Expansion:

$$y_{t+\Delta t} = y_t + \Delta t \frac{dy_t}{dt} + \frac{(\Delta t)^2}{2!} \frac{d^2 y_t}{dt^2} + \cdots$$

Maths

- If Δt is small we can ignore higher terms:

$$y_{t+\Delta t} = y_t + \Delta t \frac{dy_t}{dt}$$

- In our case we have:

$$\vec{p}_{t+\Delta t} = \vec{p}_t + \Delta t \vec{v}_t$$

Euler's Method

$$\vec{p}_{t+\Delta t} = \vec{p}_t + \Delta t \vec{v}_t$$

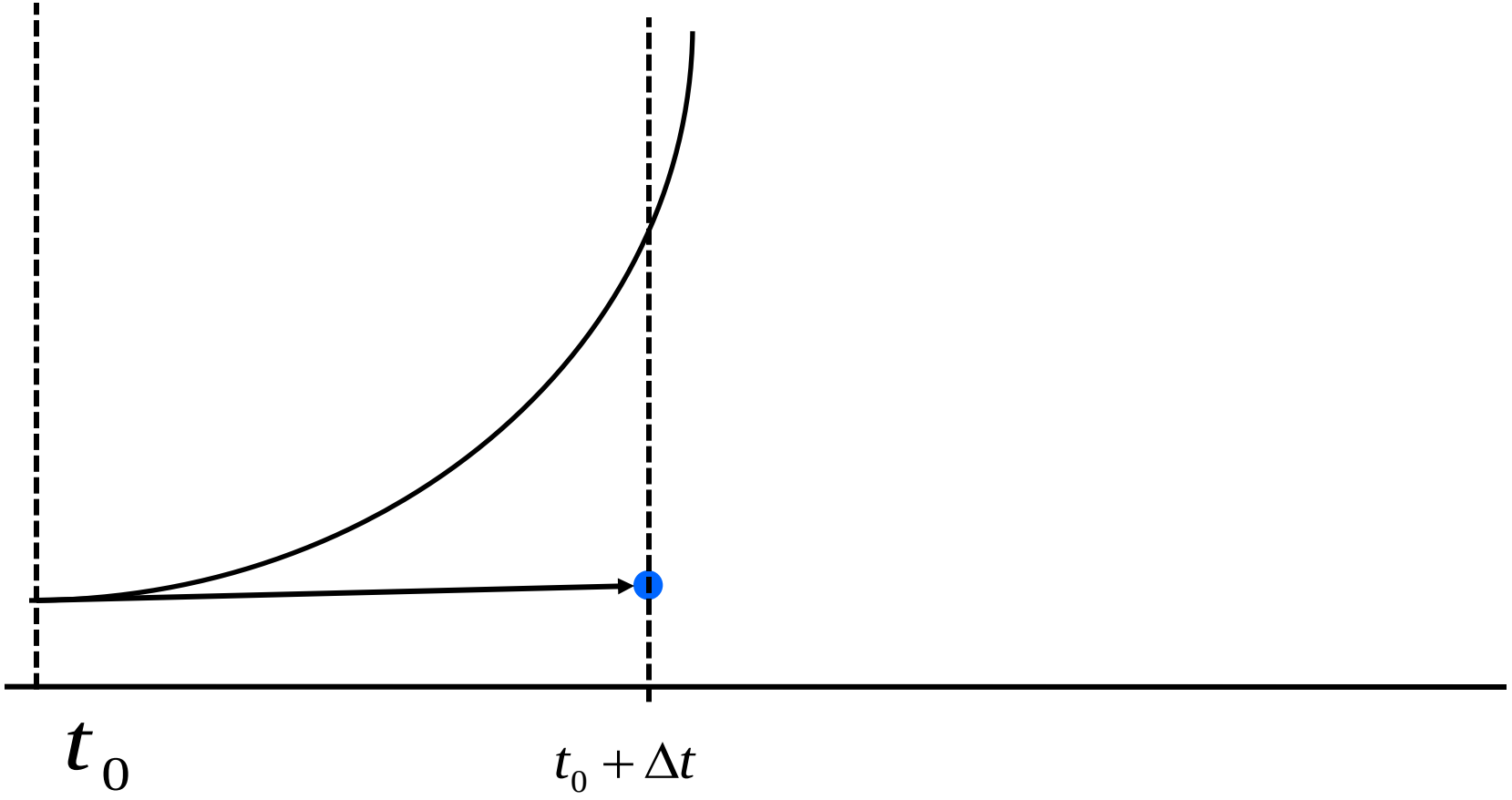
- Euler's method gives a 1st order numerical solution to the differential equation
- We can using 2nd and higher derivatives to increase accuracy

Other numerical integration methods

- Vast field with huge number of methods
- Other numerical integration methods
  - Midpoint
  - Adaptive stepsizes

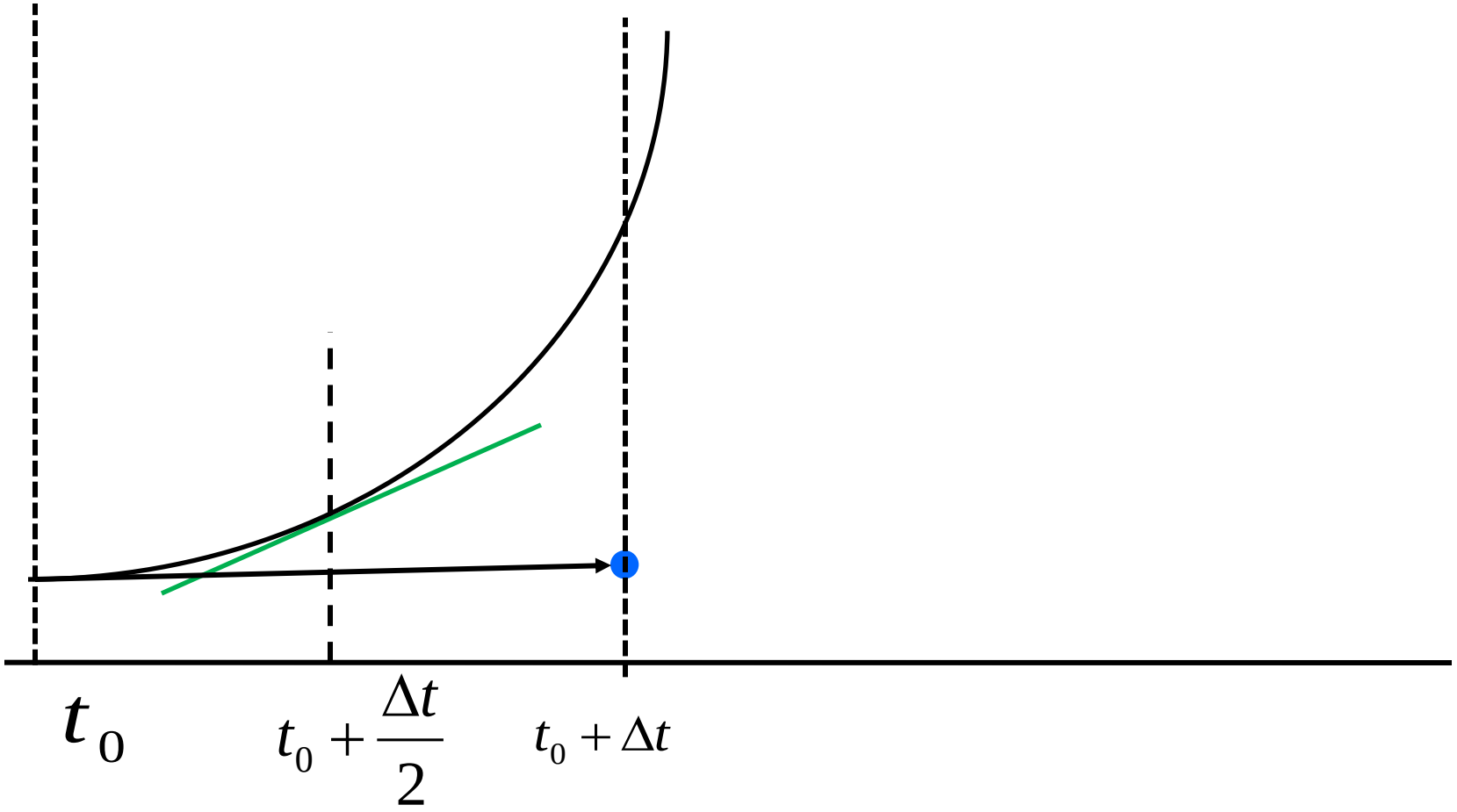
Midpoint method

- Euler's method evaluates the derivative at t_0 (The start of the step)



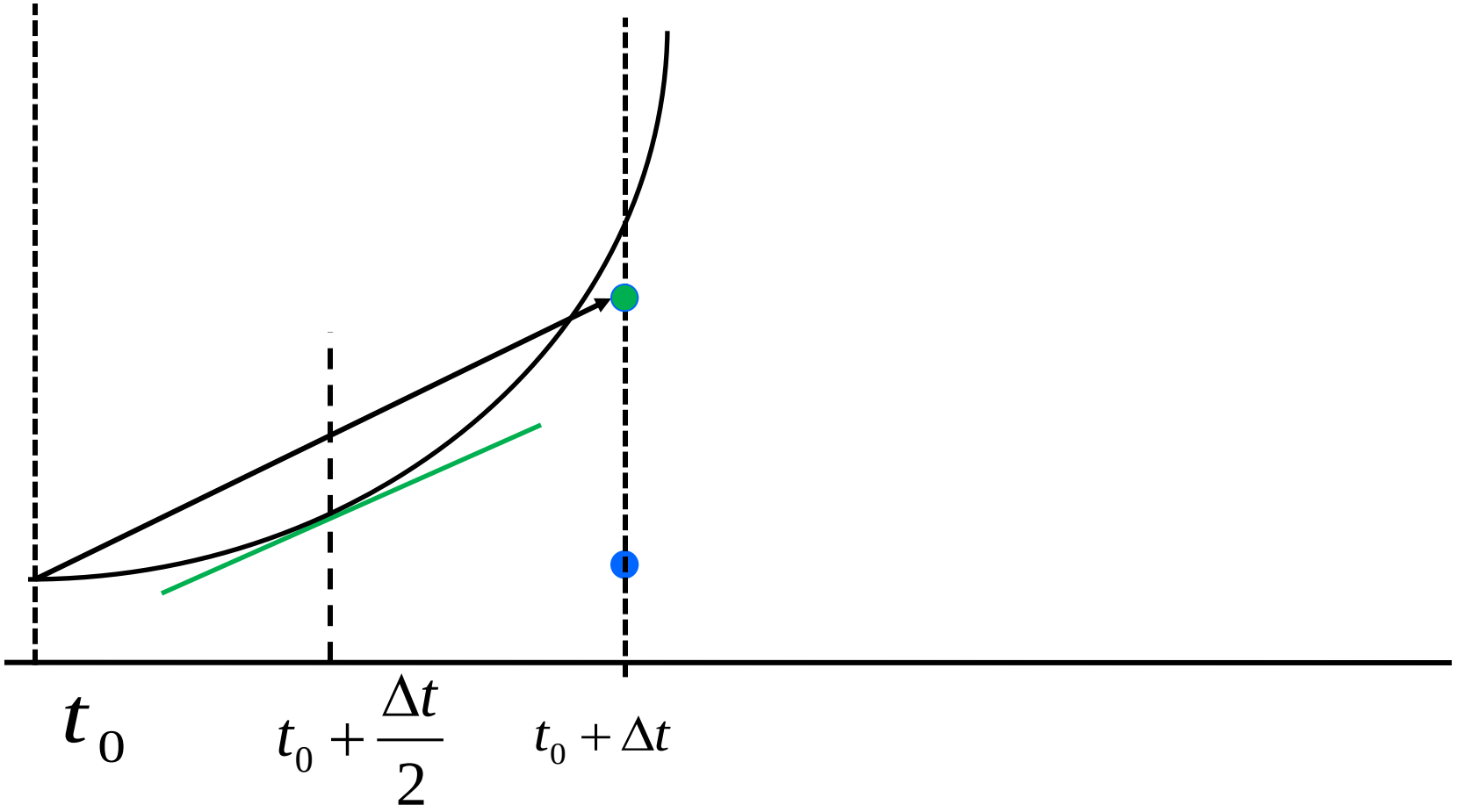
Midpoint method

- Instead we can gain more accuracy by evaluating it at t + Δt/2 (the midpoint)



Midpoint method

- Take a step from t_0 using the midpoint value



Midpoint method

- The formula becomes:

$$p_{t+\Delta t} = p_t + \Delta t \frac{dp_{t+\Delta t/2}}{dt}$$

- We have gained an extra order in accuracy by evaluating the derivative at a different time.
- “Double” application of Euler to refine accuracy
- This is one example from a many types of improvement

Stepsize

- Δt is called the step size
- No matter which integration method you are using, it is very important
- If it is too large then the solution will be inaccurate
- If it is too small then you will need a lot of processing power

Stepsize

- Error can build up over time
- So for long(ish) animations you need a very small step size
- Can get very expensive

Adaptive Stepsizes

- Only use as small a step size as you need
- Dynamically change the step size as we need it
- Every so often test the different between using step size Δt and $\Delta t /2$
- If the difference is big enough start using $\Delta t /2$
- Do the same for $\Delta t *2$

Summary

- A physical simulation may consist of
  - A set of particles
  - A set of forces on those particles
  - Initial condition (position, velocity) of those particles
- Methods introduced:
  - Euler's method
  - Midpoint method
  - Adaptive stepsizes
- The algorithm:
  - Use the forces to update the acceleration
  - Integrate to update velocities and positions

Examples

- The best way to understand it is to program and plot the trajectory

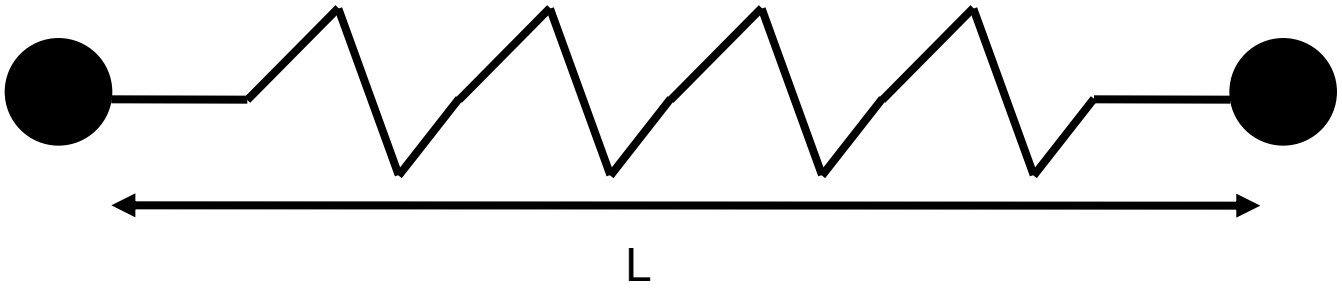
Springs

Springs

- Springs are a system that opposes deformation
- It always try to maintain a desired length
- Common tool for modelling flexible objects or to keep two objects at a prescribed distance
  - Correspond to an actual geometric element (a thread of a cloth)
  - Or defined simply to maintain a desire relationship between two objects (virtual spring)

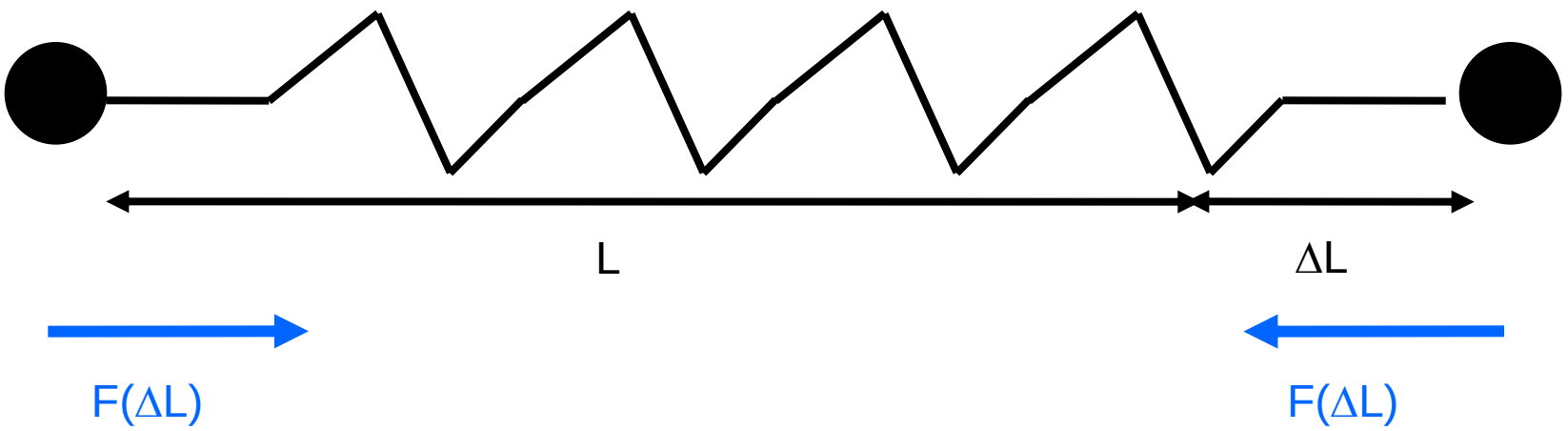
Springs

- Here we model it as two particles connected by a spring
- When attached to an object, a spring imparts a force on that object depending on the object's location relative to the other end of the spring



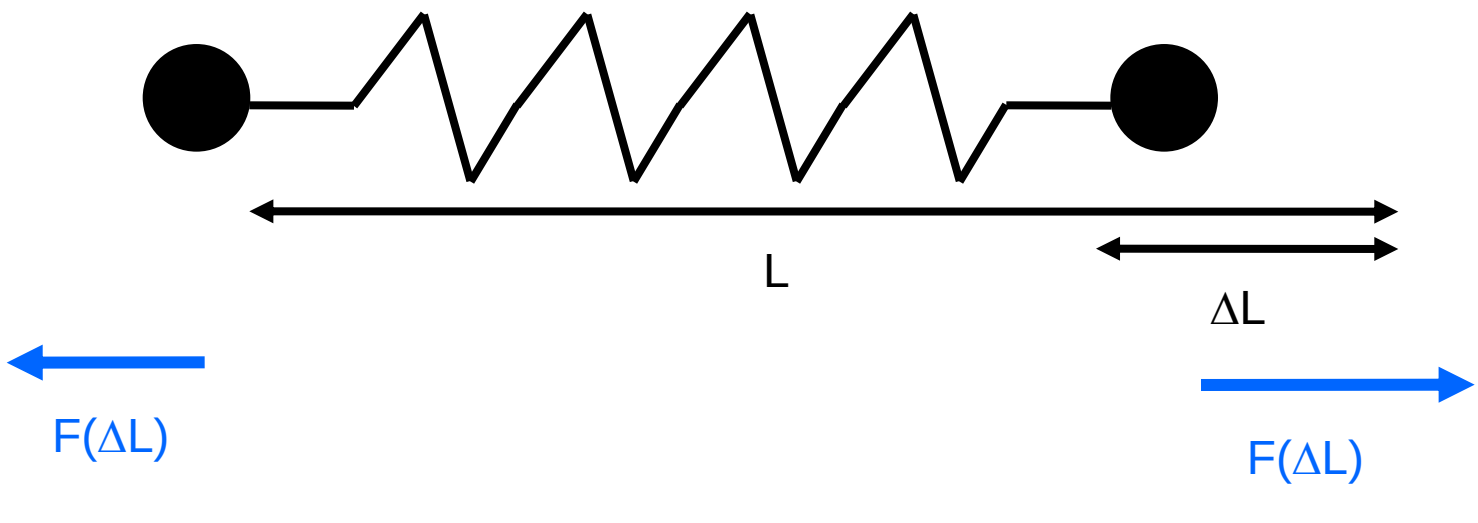
Springs

- When you stretch the spring it creates a force on both particles in the opposite direction of the deformation
- These forces are proportional to the change in length



Springs

- Ditto for compression



Springs

- The Formula for spring forces on the particles are given by Hooke's Law
- The spring force (fs) is proportional to the difference between the springs rest length (Lr) and its current length (Lc)

$$f_s = k_s(L_c - L_r)$$

- Convention: the direction of the force applied on the particle is the same as the relative position of the spring to the particle

Springs – stiffness

$$f_s = k_s(L_{current} - L_{rest})$$

- k_s is a spring constant, which describes how much the spring reacts to a change in length (*stiffness*)

Springs

- Given an change of length, using the stiffness formula we can easily generate animation of a spring
- The problem with this formula is that the spring will bounce about too much and never come to rest
- In reality this will not happen as there is friction
- Damping forces are introduced to reduce the motion of an object (like friction)

Springs – Damper

- Damper (or damping force, fd) is negatively proportional to the relative velocity of the two particles

$$f_d = -k_d v_{spring}$$

- k_d is a damper constant, which determines how much resistance there is to a change in spring length

Springs-mass-damper system

- Springs-mass-damper system

$$f = k_s(L_{current} - L_{rest}) - k_d v_{spring}$$

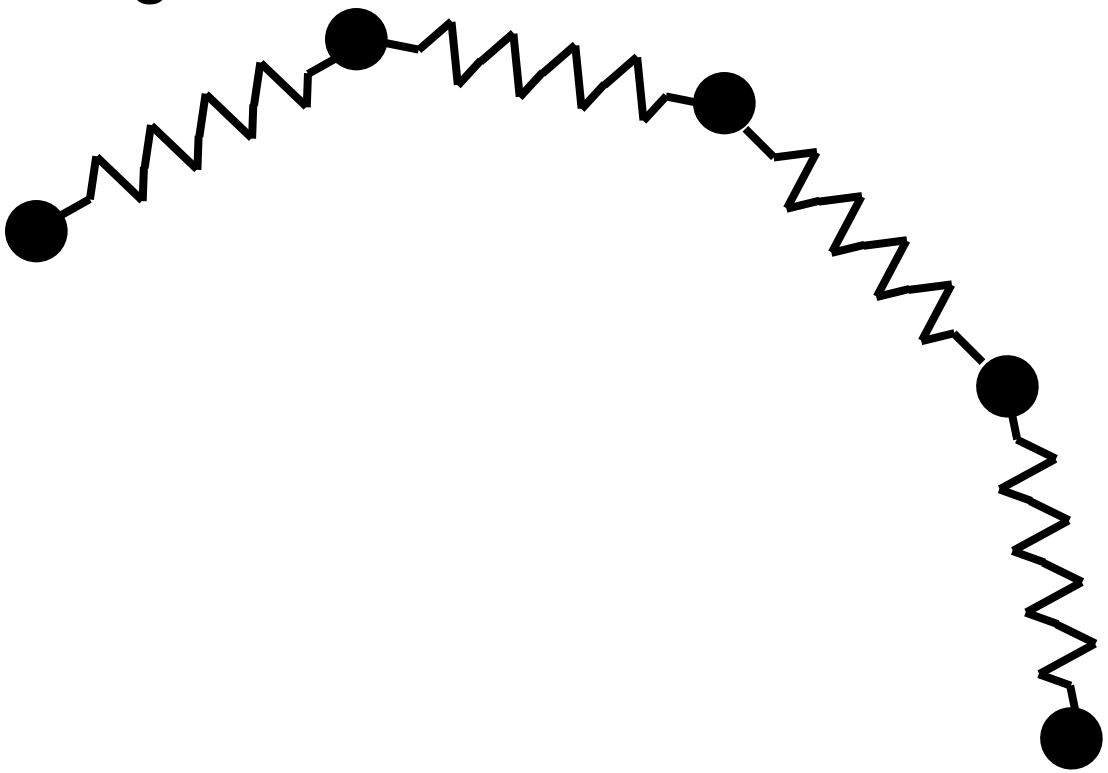
- Keep two objects at a certain distance
- Maintain an angle between two polygons that share an edge

Using Springs

- A string of springs are commonly used to model flexible shapes
  - Hair
  - Cloth
  - Grass

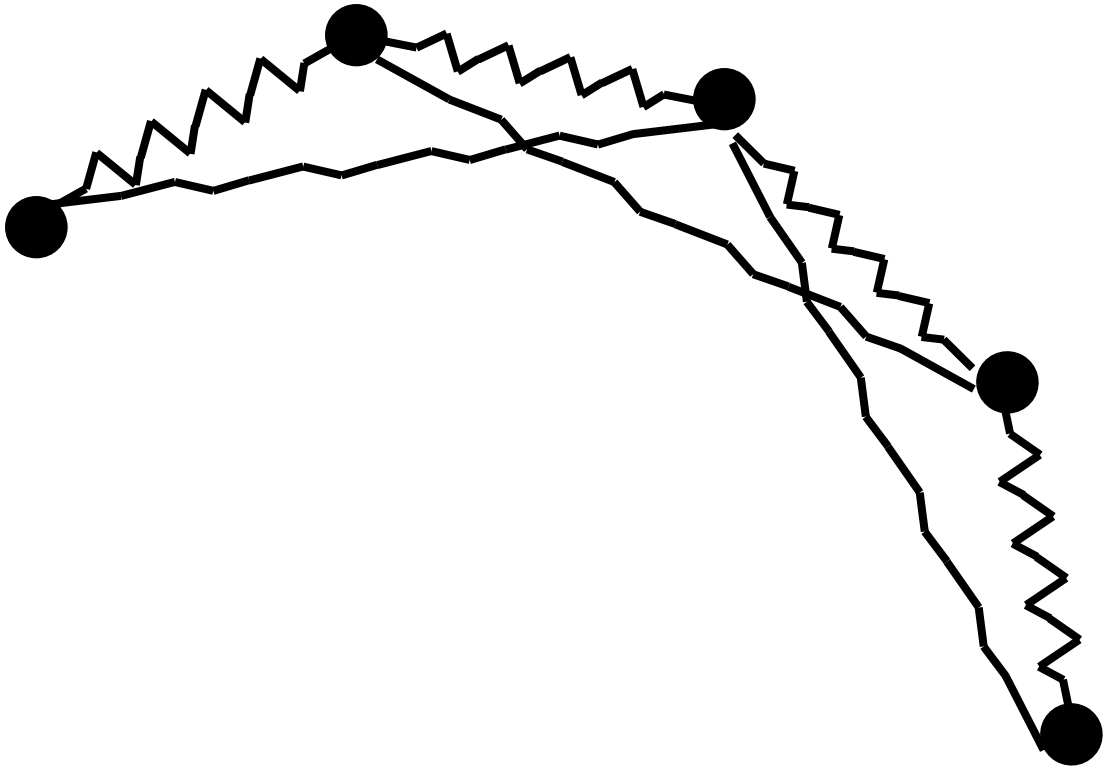
Hair

- Problem: there is nothing to stop the hair bending



Hair

- Add in springs forces to prevent bending
- Counteracts bending between 3 particles

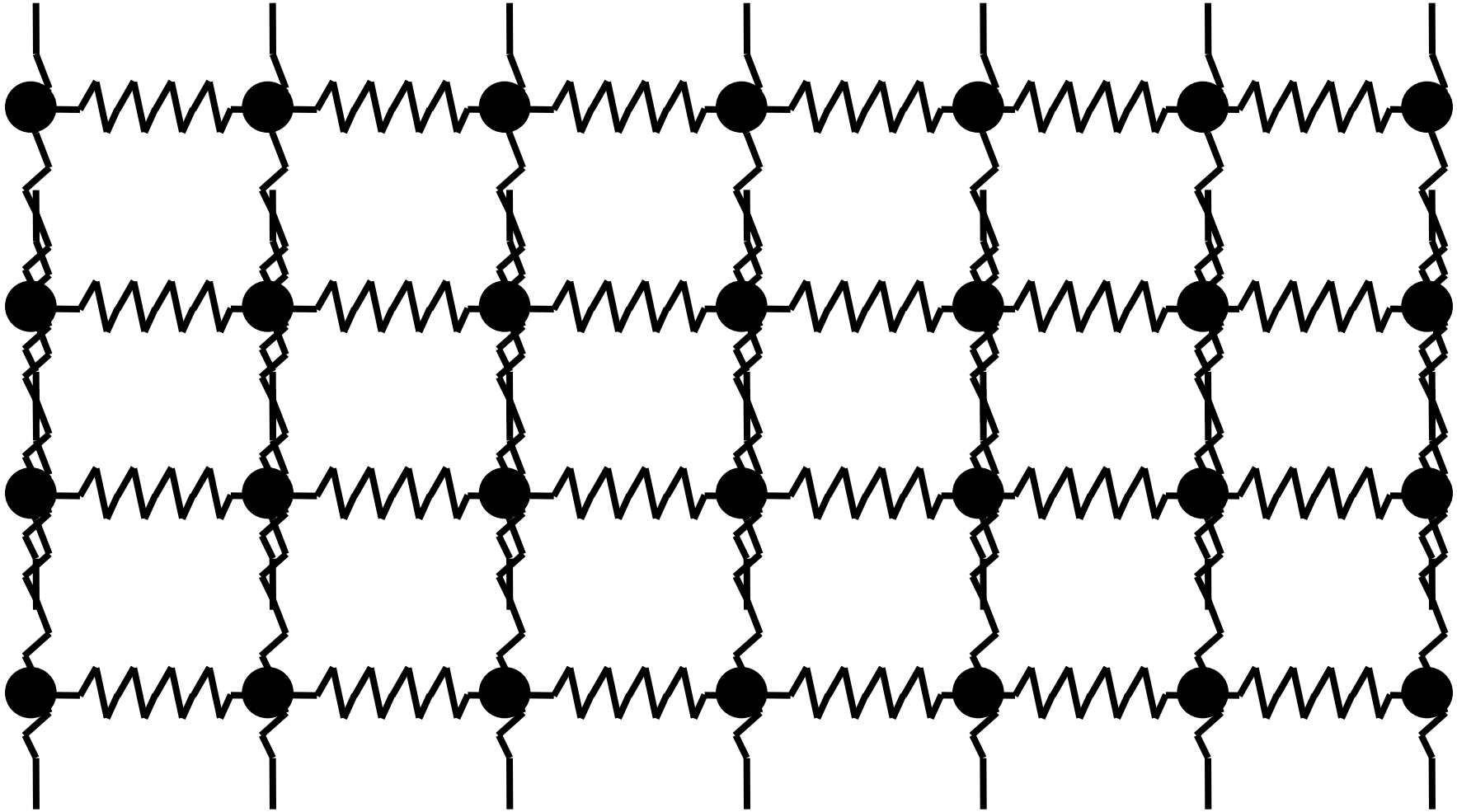


Hair

- Animating each hair would be hugely expensive
- Generally methods simulate a small number of strands and render each one as a clump of hair rather than an individual hair
- videos\Mei Hair Test.flv

Cloth

- Structure Spring

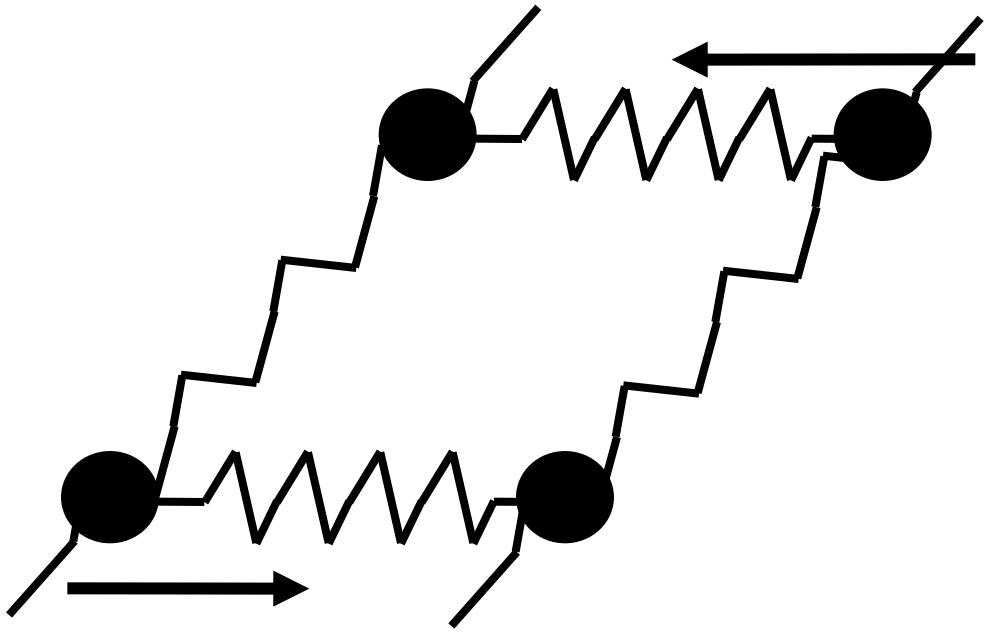


Cloth

- This simple lattice allows you to simulate the in plane stretching properties of cloth well, but there are other forces that go on in cloth
- But the result is not ideal
- videos\Cloth Simulation (Without Shear and Bend Springs).flv

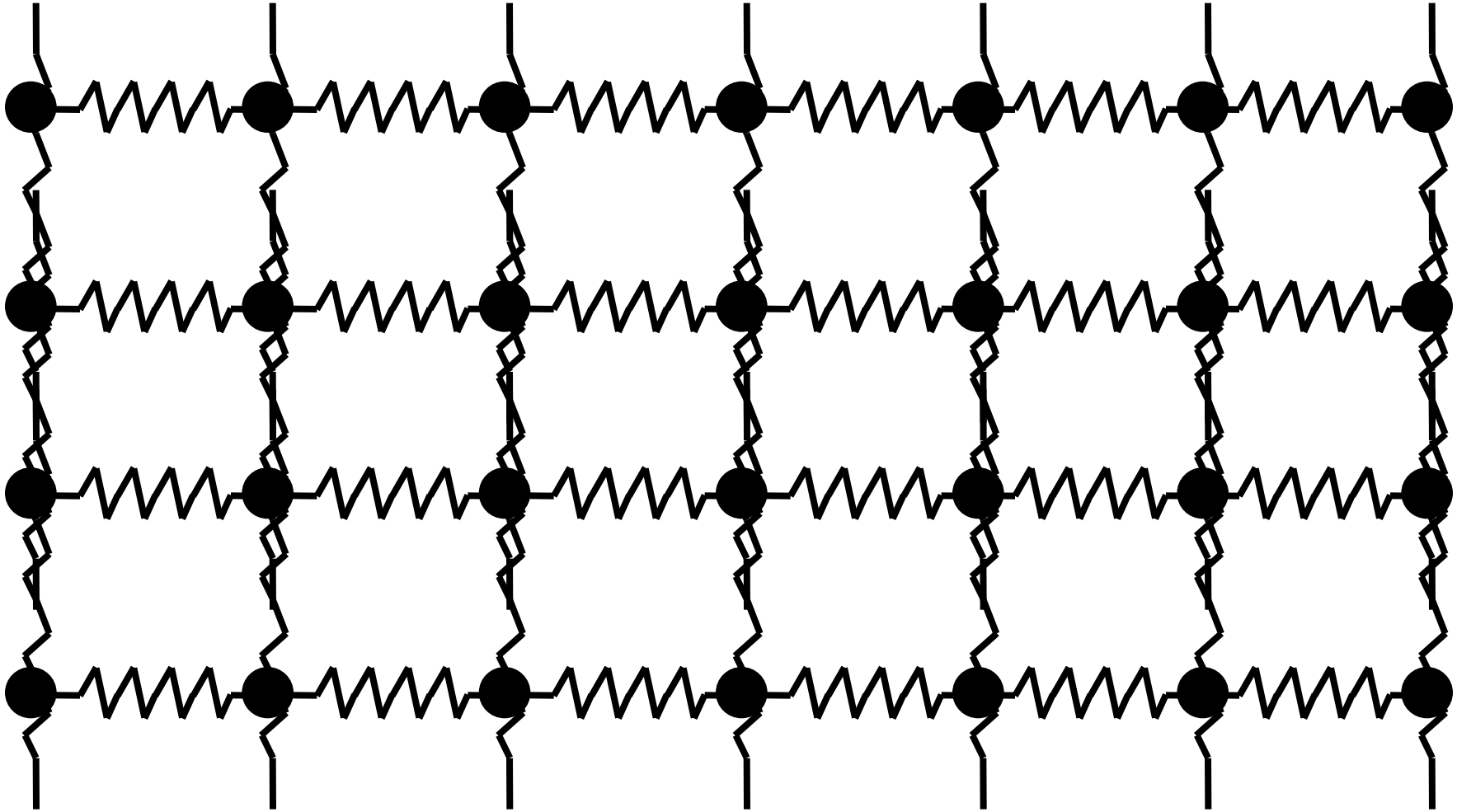
Cloth

- Shearing: this becomes an issue going from 1D hair to 2D cloth
- Need to add a force that prevents the cloth pulling against itself



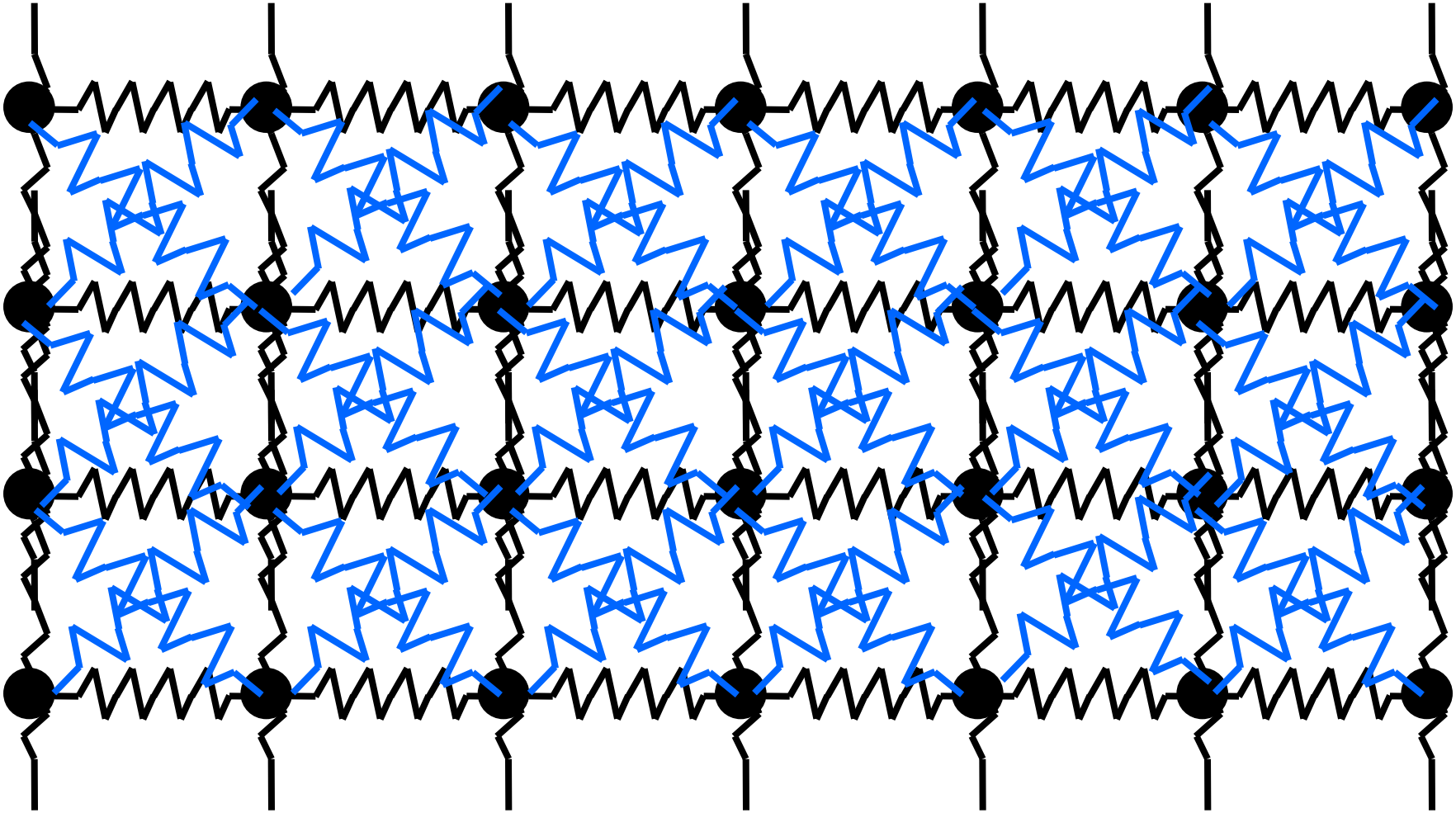
Cloth

- Add in diagonal springs for shearing



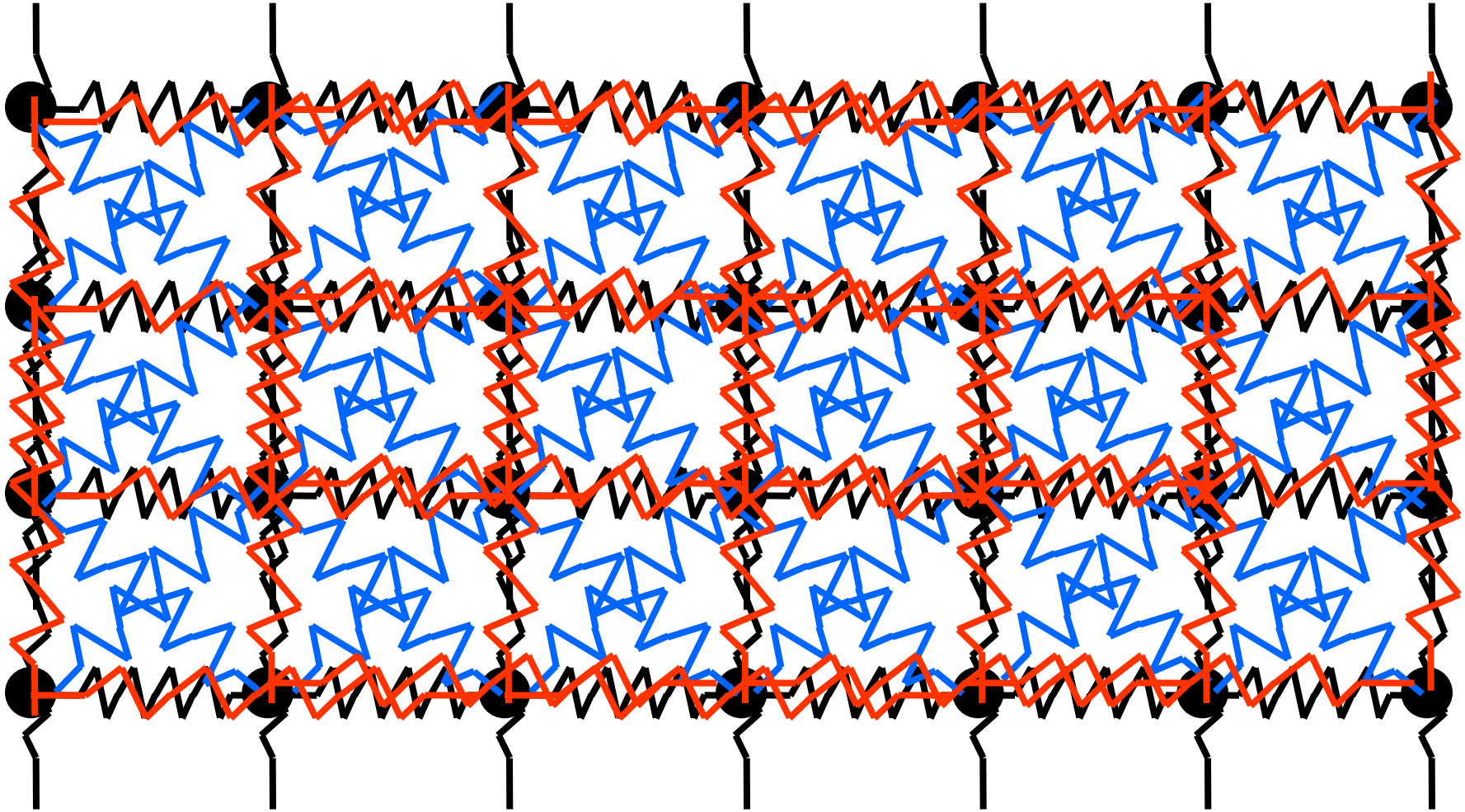
Cloth

- Add in diagonal springs for shearing

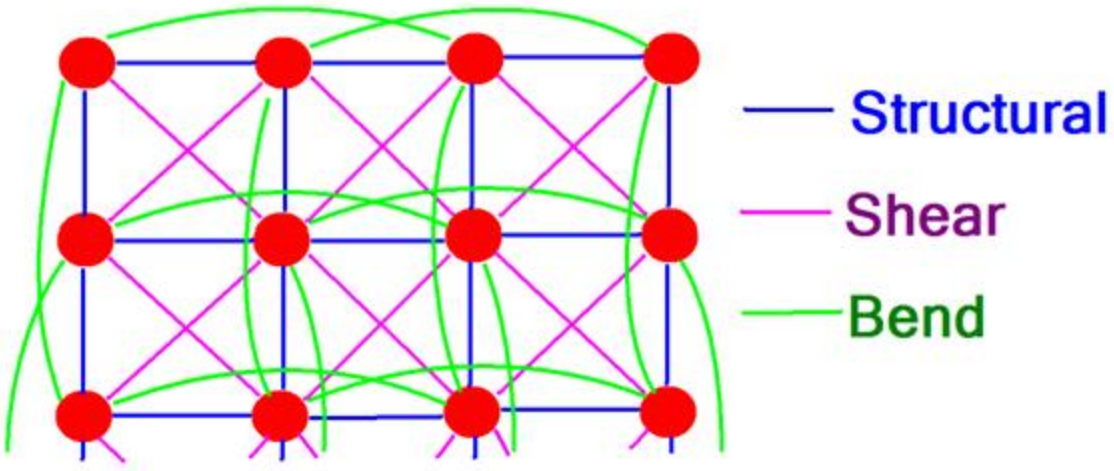


Cloth

- And springs for bending (same as hair)



Cloth

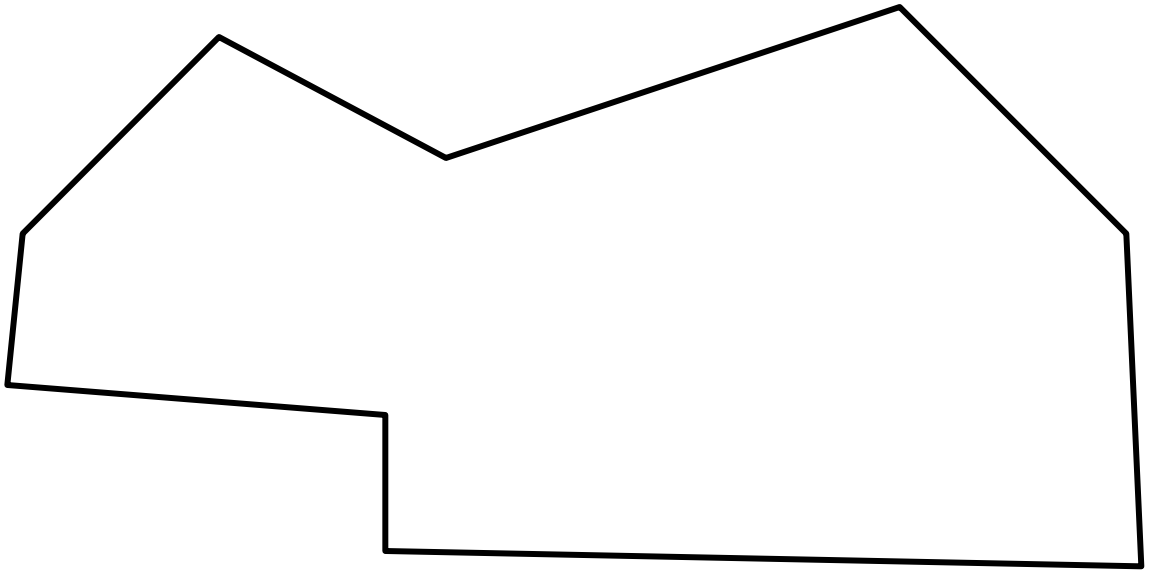


- With only structural springs: videos\Cloth Simulation (Without Shear and Bend Springs).flv
- Add shearing springs: videos\Cloth Simulation (Without Bend Springs).flv
- Add bending springs: videos\Cloth Simulation (Without Length Constraints).flv
- With both shearing and bending: videos\Cloth Simulation.flv

Rigid Bodies

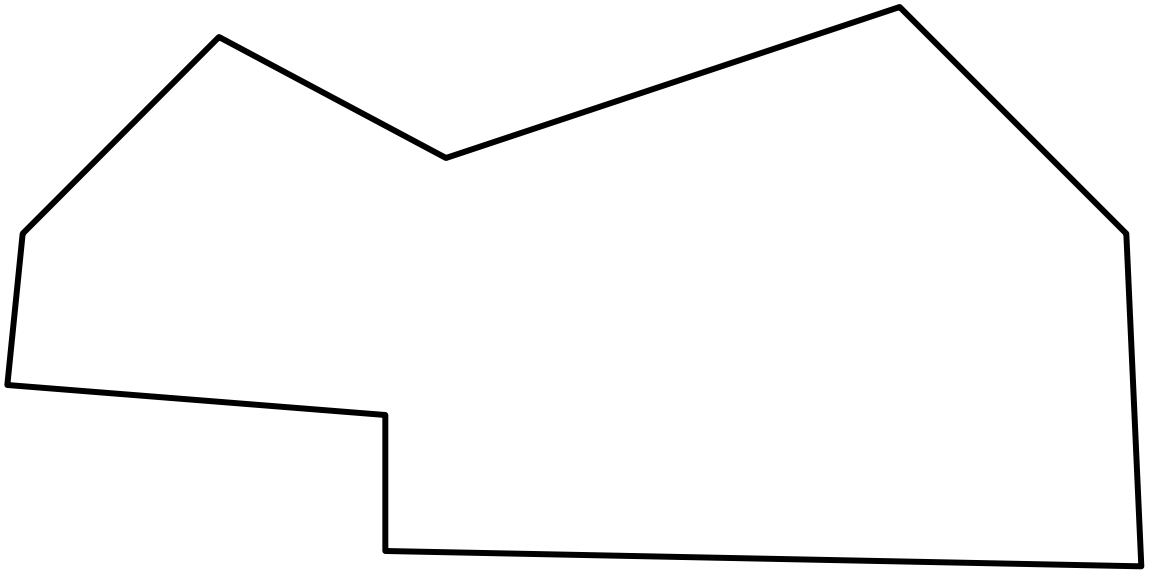
Rigid bodies

- Up to now we have dealt with forces on particles (points)
- What about more complex object?
- A rigid body:



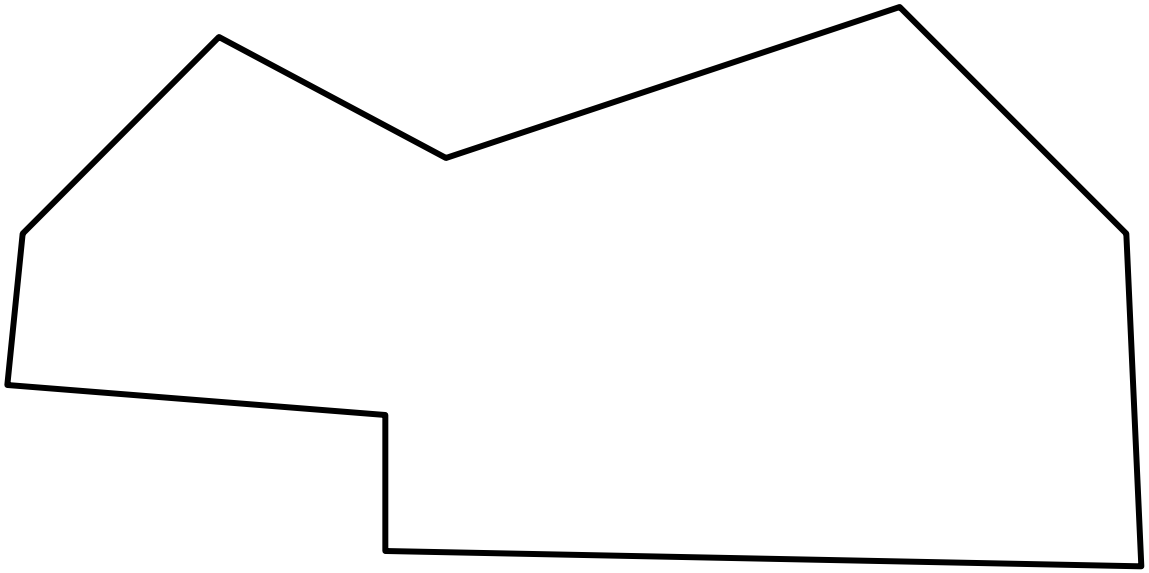
Rigid bodies

- Can move around
- But cannot deform



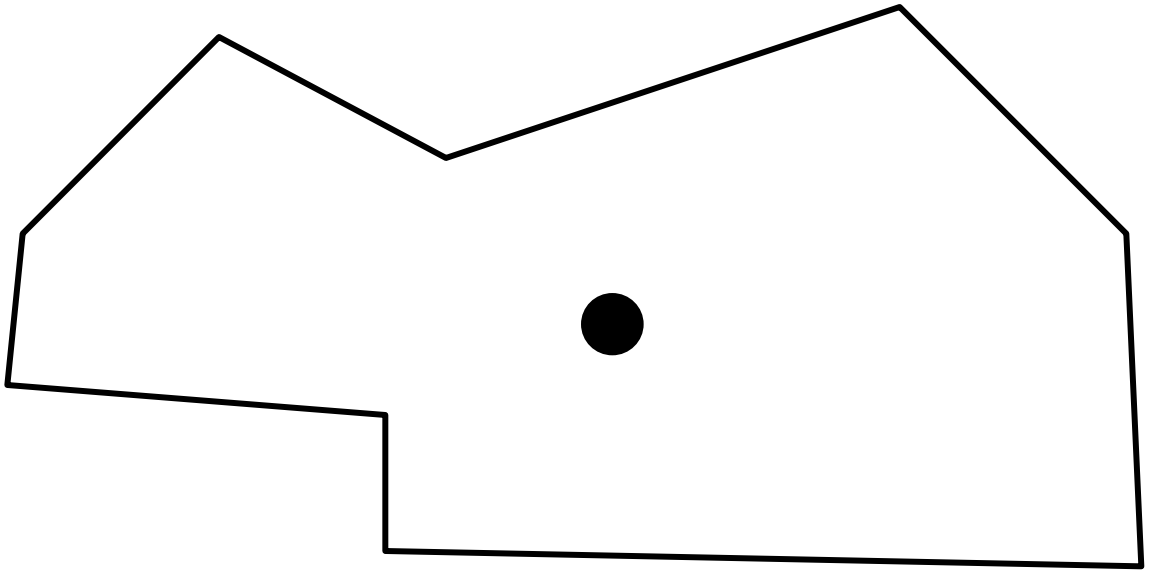
Rigid bodies

- Could make it out of springs
- But as you don’t want it to deform the spring constant must be very high
- Very expensive to integrate



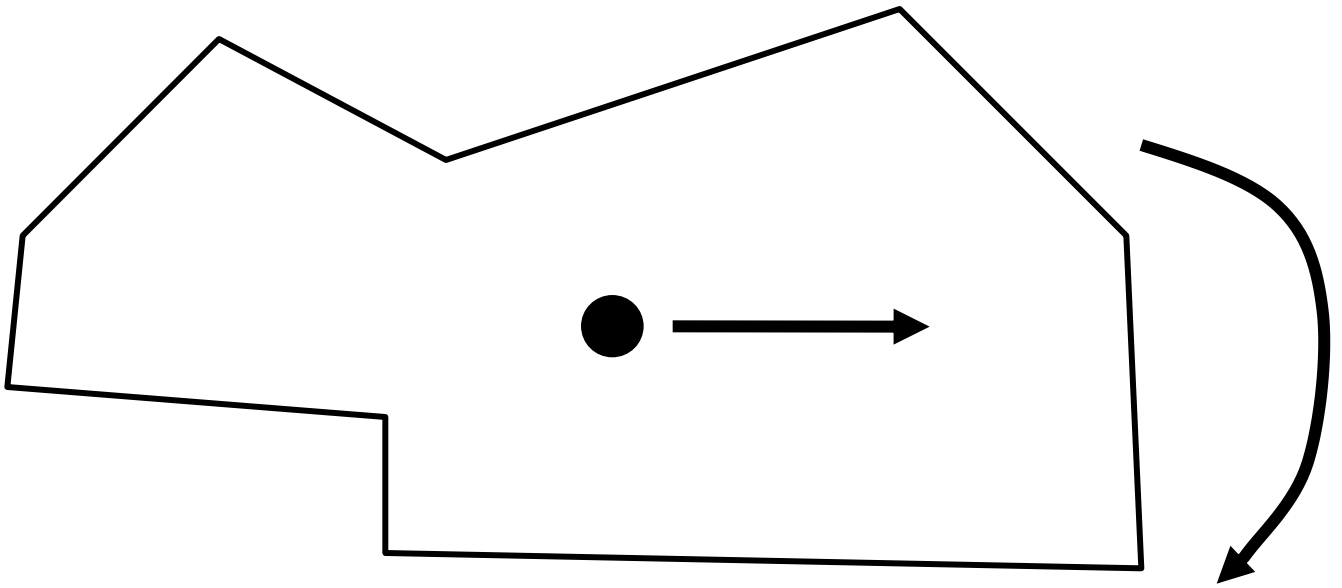
Rigid bodies

- Has a centre of mass that can be treated as a particle



Rigid bodies

- The movement of a rigid body can be decomposed into 2:
  - Motion of the centre of mass
  - Rotation about the centre of mass



Rigid bodies

- The motion of the centre of mass is treated the same as the particle dynamics discussed above
- The rotation adds a new set of components above the x,y,z position components
- Won’t go into much detail about the maths here

Demo

- videos\Blender Cloth, Particle, Physics, and Softbody Animations.flv

Particle Systems

Particle Systems

- Modelling natural phenomena could be difficult
  - Wind, clouds, water, fire, snow, explosions
- “Fuzzy” objects
  - Irregular surface (not well-defined)
  - Dynamic and fluid (non-rigid objects)
  - Their motions cannot be described by simple transformations

videos\Particle Systems - Snow.flv

Particle Systems

- A more ad-hoc simulation method
- Good for phenomena that are not well represented by solid, rigid surfaces/bodies
- Advantages:
  - Creating complex systems with little human effort
  - The level of detail can be easily adjusted (closer to camera more particles, further away less particles)

Particle Systems

- These systems can all be modelled as large sets of particles
- Not (necessarily) physically modelled in the way described above
- The structure of all particle systems is the same
- The details of particle behaviour can differ

Basic Model of Particle Systems

- At each frame:
  - Any new particles that are born during this frame are generated
  - Each new particle is assigned attributes
  - Any particles that have exceeded their allocated life span are terminated
  - The remaining particles are animated and their shading parameters changed according to the controlling processes
  - Particles are rendered

Particle Creation

- Particles are randomly created
- They are all created within a certain area
- The mean number of particles created controls the size of the system
- This mean can vary over time to have the system grow or shrink

Particle Attributes

- The particle system is assigned:
  - The particle system origin
  - Orientation
  - A generation shape which defines the region around the origin where new particles are placed
- Each new particle is assigned:
  - Initial status (position, velocity, size, colour, transparency)
  - Shape
  - Lifetime

Particle Deletion

- Particles are also deleted
- Particles have a certain lifetime, if the particle has exceeded its lifetime it is destroyed
- Also sometimes particles are deleted if they leave a certain area (visible area)

Particle Dynamics

- This step controls what the particles actually do
- Varies depending on what you are simulating (e.g. snow fall vs fire)
- Examples:
  - Particles move according to their velocity
  - Particles change colour (e.g. get brighter/darker)

Particle Rendering

- Particles can be treated as light sources
  - Don’t have to worry about occlusion
- Or particles are small primitives, often partially transparent
- Can apply motion blur or other types of blurring to make individual particles less visible
- Don’t need to consider individual shadows but for some application when shadowing effects are an important visual cue, we can used the density of particles to estimate shadowing

Example: Smoke generation

- Particle Creation:
  - Particles are created at the source of the smoke (often a point)

Example: Smoke generation

- Particle Attributes:
  - Velocity: Upwards, slightly to the side
  - Appearance: grey and transparent

Example: Smoke generation

- Particle Deletion:
  - Delete at end of lifespan
  - Limited lifetime allows for smoke dissipation
  - Delete if they go out of the view

Example: Smoke generation

- Particle Motion and transformation:
  - Particles move according to their velocity
  - It is also useful to have a wind force
    - Accelerates the particles in the direction of the wind
  - Turbulence makes the particles more realistic
    - Random variation in wind force
  - Particles increase in size

Example: Smoke generation

- Particle Rendering:
  - Particles are rendered as small polygons
  - The polygons are blurred at the edges so that its less obvious that they are polygons
  - They are transparent

Example: Smoke generation

- videos\3Ds Max Fire and smoke particle system.flv
- videos\Maya Fluid Effects Explosion Testing.flv
- videos\Maya Fluids effects_ Fire & Smoke.flv
- Examples in Unity