Inside LiquidFun

Kentaro Suto
Senior Software Engineer
Google

Contents

Overview of LiquidFun

What is particle simulation?

Data structures

Simulation Algorithm

An open source game that uses LiquidFun

Sample LiquidFun app for Android and IOS

Overview of LiquidFun

- 2D rigid body and fluid simulation library
- Extension of Box2D
- Written in C++, bindings for Java, translated to JavaScript
- Runs on Android, iOS, Windows, OS X, Linux, browsers
- Distinguishing feature: particle simulation

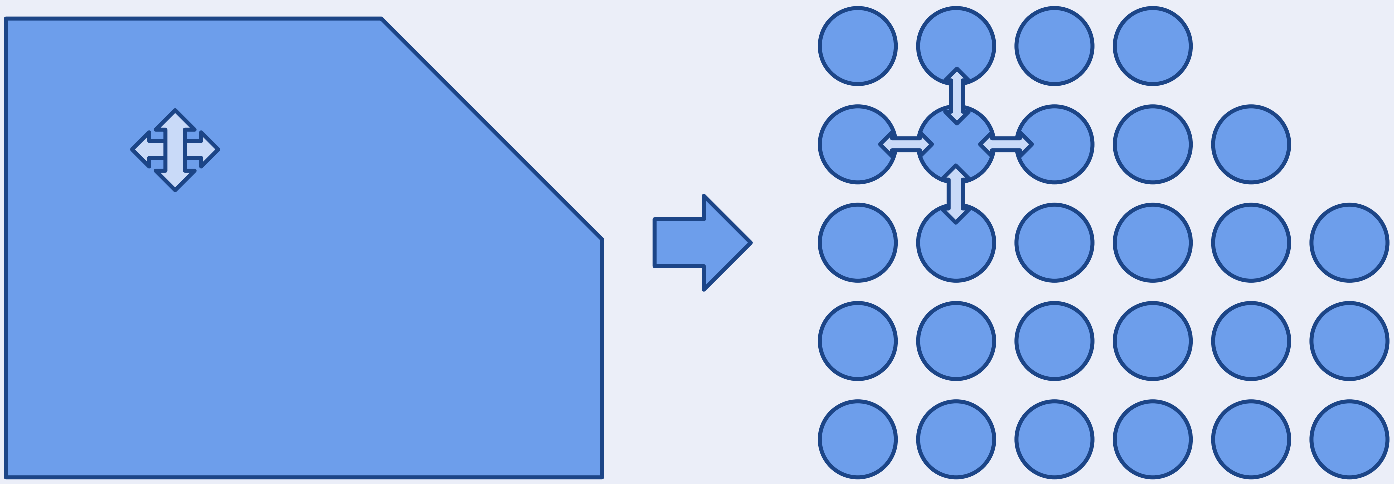
What is particle simulation?

In particle simulation...

- objects are composed of particles (circles, in LiquidFun)
- physics is described by interactions between particles

Particle simulation works well with...

- Fluids--liquids and gasses
- Deformable objects
- Interactions of different materials



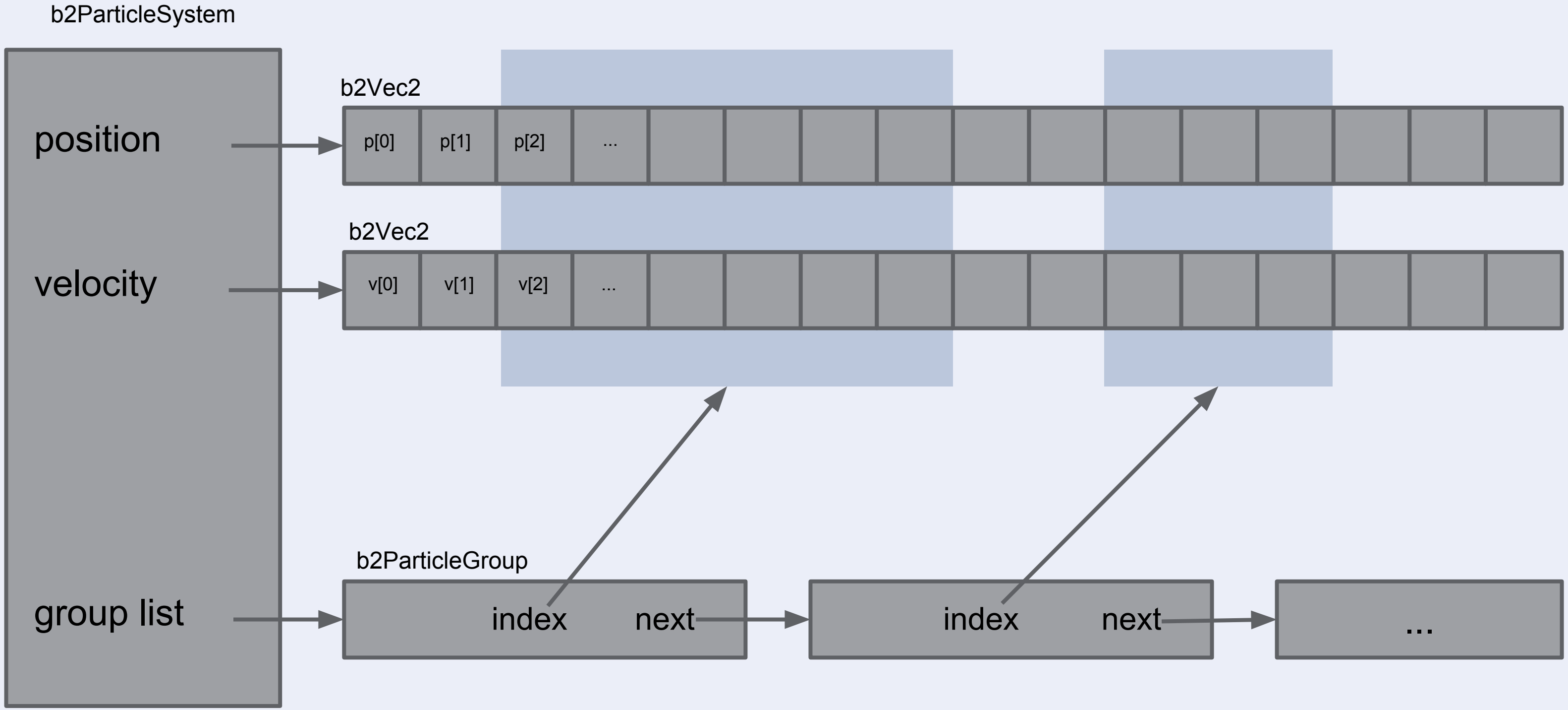
Particle simulation

Data structures

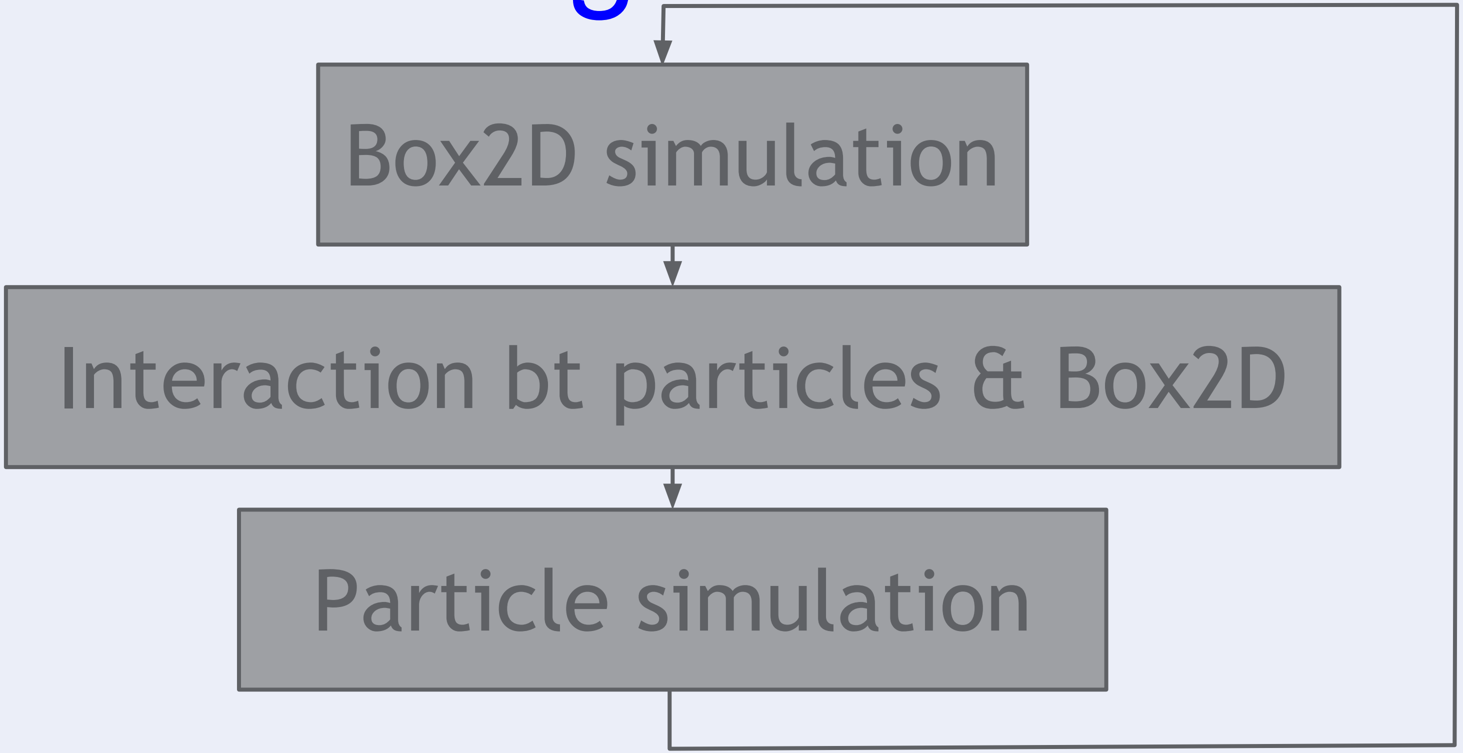
Particle data
- flags
- positions
- velocities
- ...

Particle group
- flags
- first index
- last index
- ...

Data structures



Simulation Algorithm



Legend

p[i]	position of particle i
v[i]	velocity of particle i
Δt	simulation time step
R	particle radius
D	particle diameter = 2*R

These symbols will be used in the next few slides.

Particle simulation

Collision detection

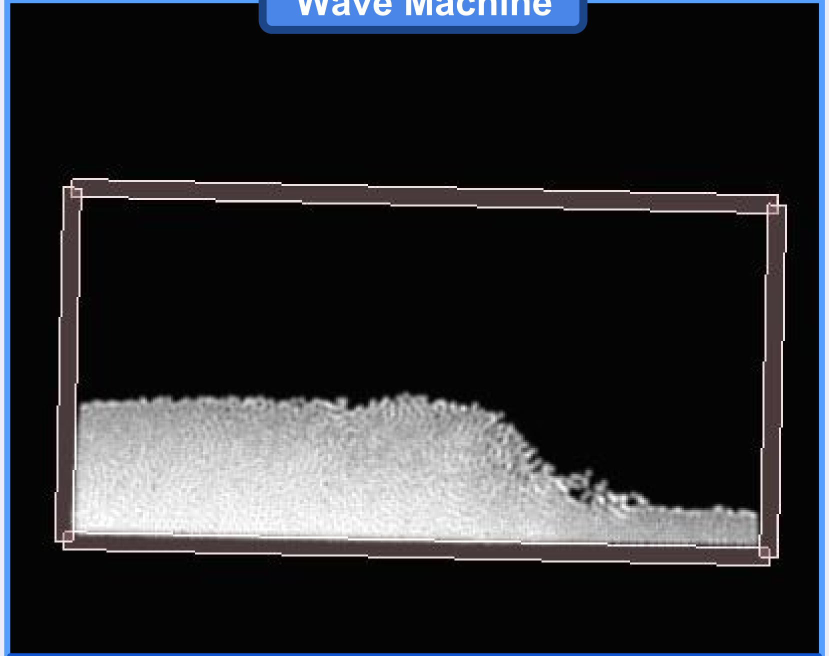
Apply pressure

Apply additional forces (viscous, etc.)

Restrict particle velocity (rigid, wall, etc.)

Move particles

Wave Machine



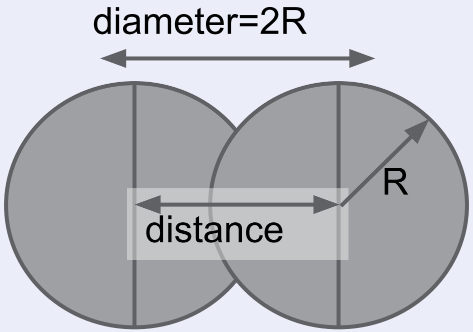
The wave machine, from LiquidFun's testbed

Collision detection

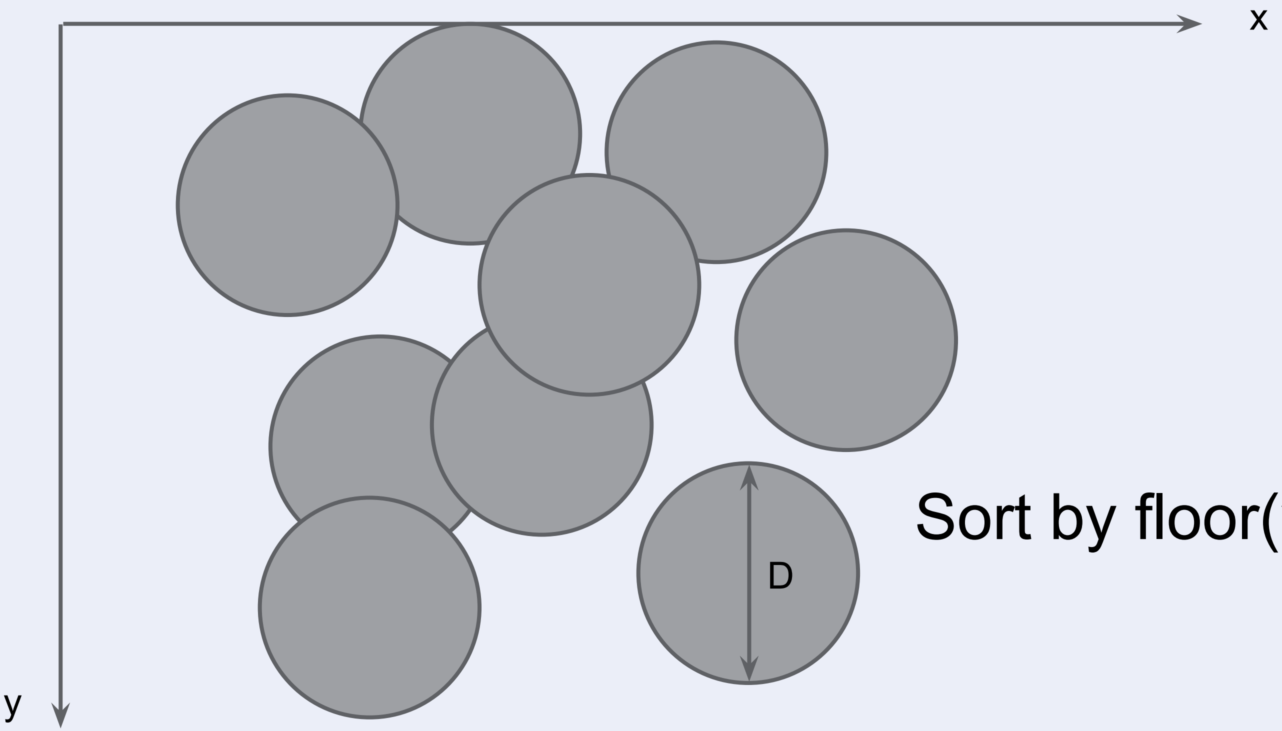
Find pairs of particles closer than the particle diameter

Record the following values:

- i,j: indices of colliding particles
- n[i,j]: normalized relative position
- w[i,j]: calculated as 1-distance/diameter
  - 0 if barely contacting, 1 if overlapping

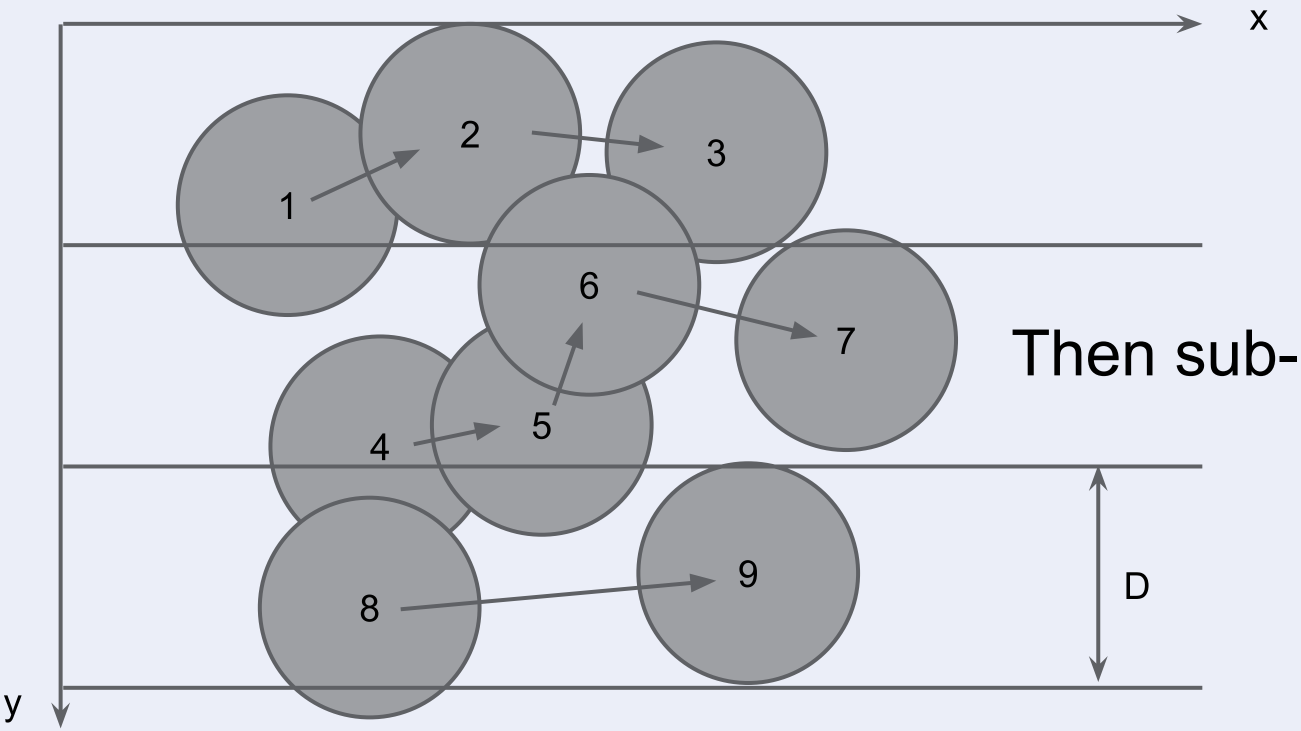


Collision detection -- particle sort



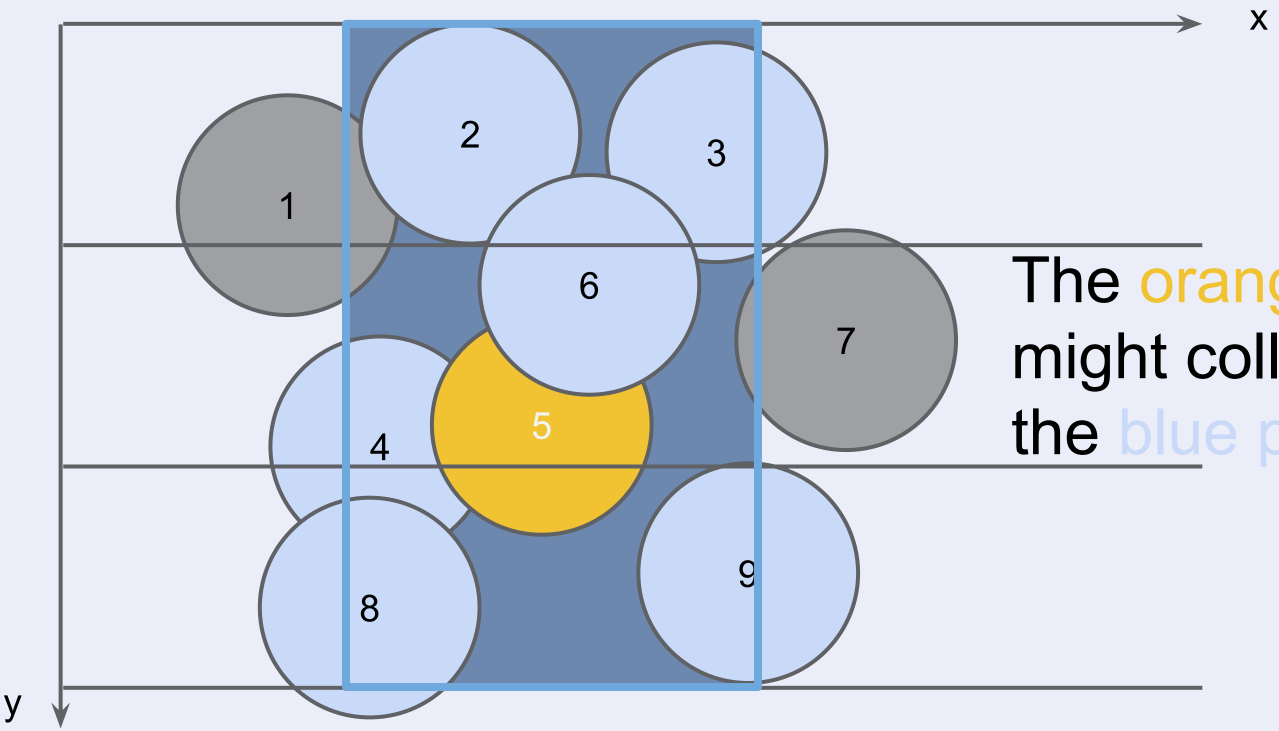
Sort by floor(y/D)

Collision detection -- particle sort



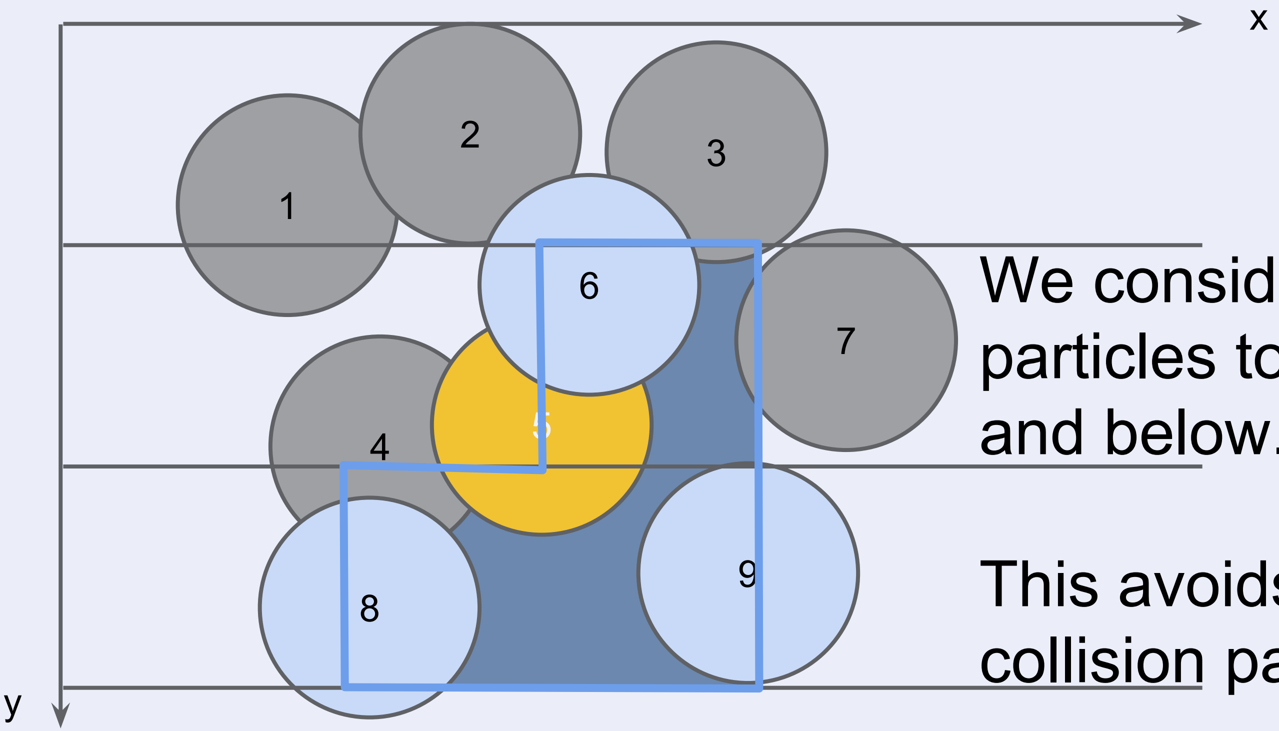
Then sub-sort by x

Collision detection -- broad pass collision detection



The orange particle might collide with the blue particles

Collision detection -- broad pass collision detection



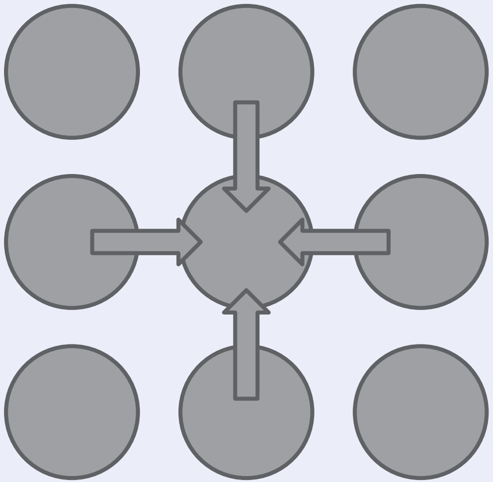
We consider only particles to the right and below.

This avoids duplicate collision pairs.

Apply Pressure

Sum up the weight of contacts for each particle

w[i] = sum(w[i,j],j)

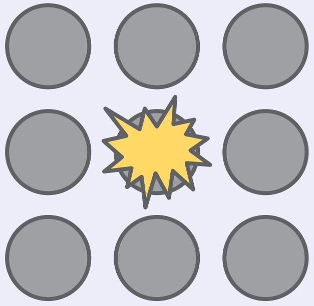


Apply Pressure

Calculate the pressure of each particle using the weight sum

$$h[i] = \max(0, \eta * (w[i] - w0))$$

h[i]	pressure of particle i
η	constant coefficient of weight
w0	constant average weight

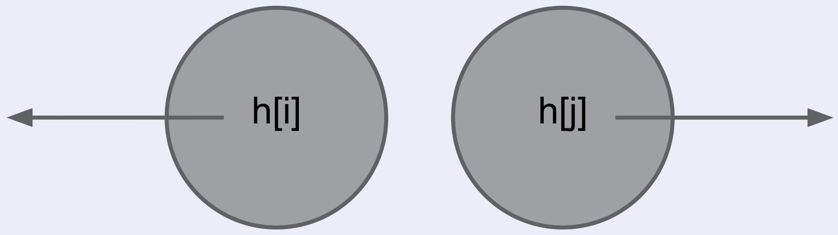


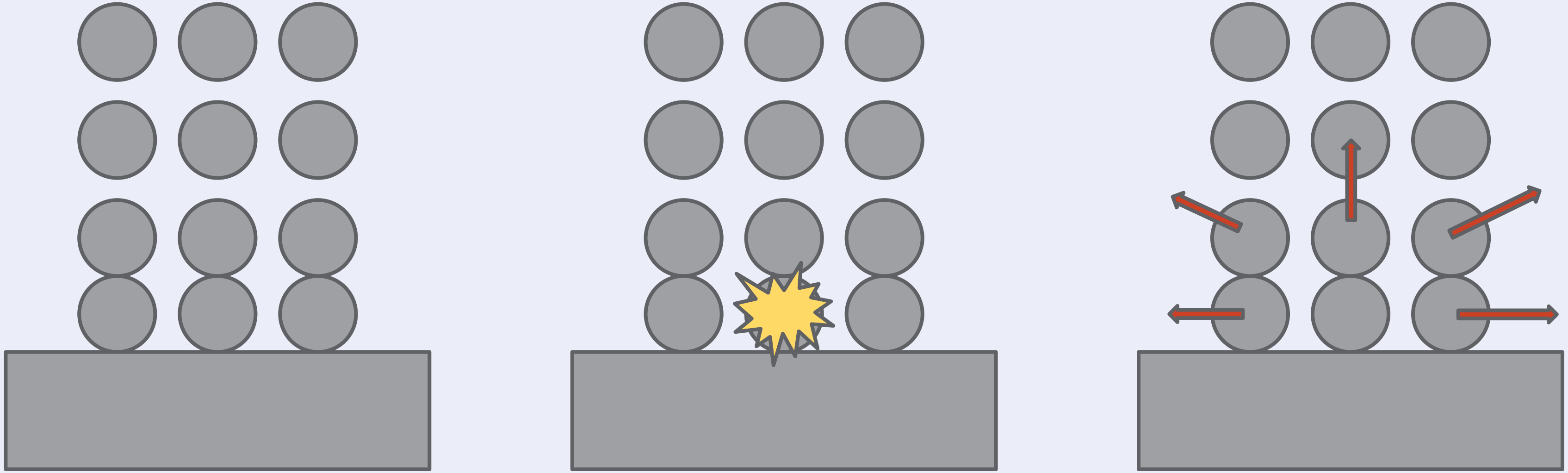
Apply Pressure

Apply a repulsive force to each contacting particle according to pressure

$$v[i] \leftarrow v[i] + \Delta t * a * (h[i] + h[j]) * w[i,j] * n[i,j]$$

a: constant coefficient of repulsion





compression pressure force

Apply pressure

Apply Viscous Force

If a particle is viscous, mix velocity with its contacting particles

$$v[i] \leftarrow v[i] + \Delta t * \mu * (v[j] - v[i])$$

μ : constant coefficient of viscosity

Apply Spring Force

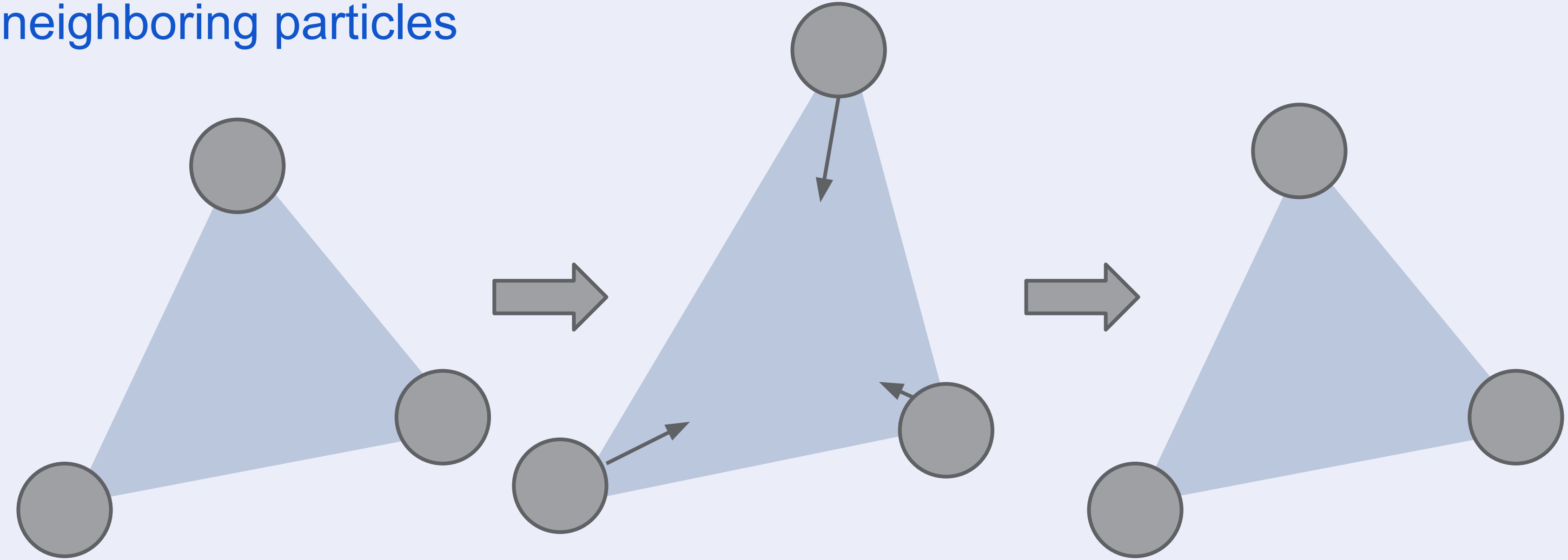
If a particle is springy, apply force to restore distance between neighboring particles

$$v[i] \leftarrow v[i] + \Delta t * k * (\mathrm{distance}(p[j],p[i]) - L[i,j]) * n[i,j]$$

k	constant spring coefficient
L[i,j]	initial distance

Apply Elastic Force

If a particle is elastic, apply force to restore relative positions among neighboring particles



Apply Powder Force

If a particle is powder, apply simple potential force instead of pressure

$$v[i] \leftarrow v[i] + \Delta t * c * \max(0, w[i,j] - w1) * n[i,j]$$

c: constant potential coefficient

w1: constant threshold weight

Apply Tensile Force

If a particle is tensile, apply force to simulate surface tension

$$w[i] = \mathrm{sum}(w[i,j],\ j)$$

$$s[i] = \mathrm{sum}((1 - w[i,j]) * w[i,j] * n[i,j],\ j)$$

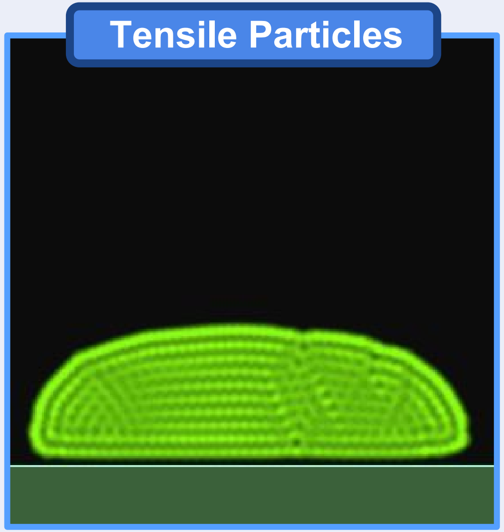
$$A[i] = a * (w[i] + w[j] - 2 * w0)$$

$$B[i] = b * \mathrm{dot}(s[j] - s[i],\ n[i,j]))$$

$$v[i] \leftarrow v[i] - \Delta t * (A[i] + B[i]) * n[i,j]$$

a,b: constant coefficients

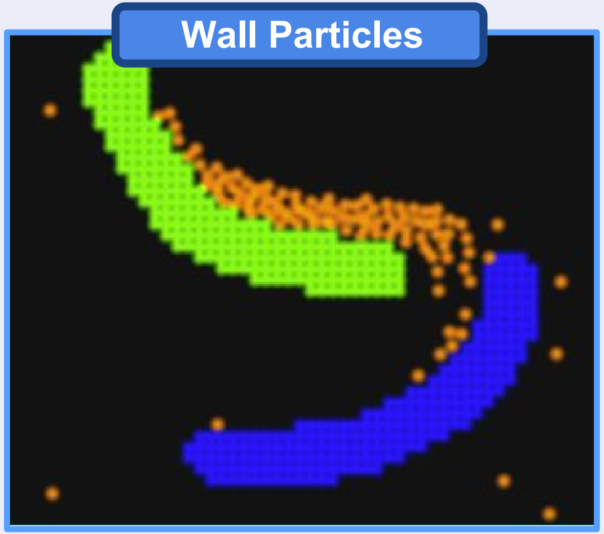
w0: constant average weight



Restrict Particle Velocity

For wall particles, set velocity to zero.

$$v[i] \leftarrow 0$$

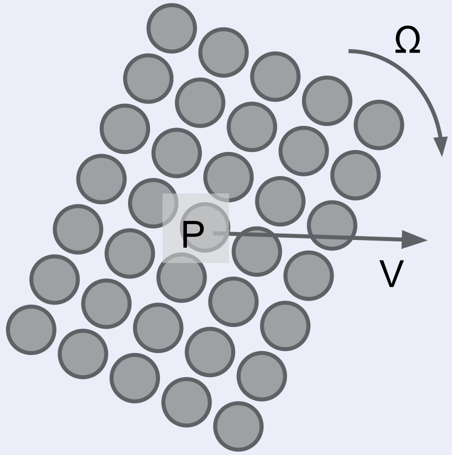


Restrict Particle Velocity

For particles in a rigid group, set velocities to maintain relative positions.

$$v[i] \leftarrow V + \mathrm{cross}(\Omega, p[i] - P)$$

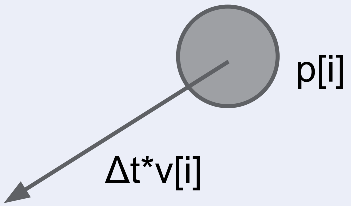
V: linear velocity of the group
Ω: angular velocity of the group
P: center of mass of the group



Move Particles

Use velocity to update particle positions

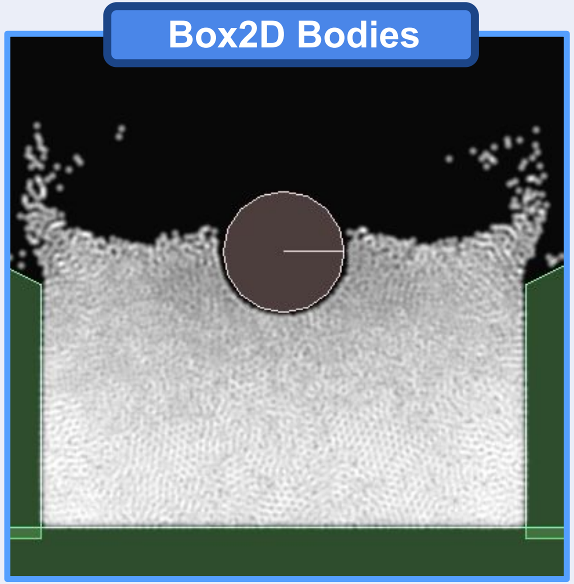
$$p[i] \leftarrow p[i] + \Delta t * v[i]$$

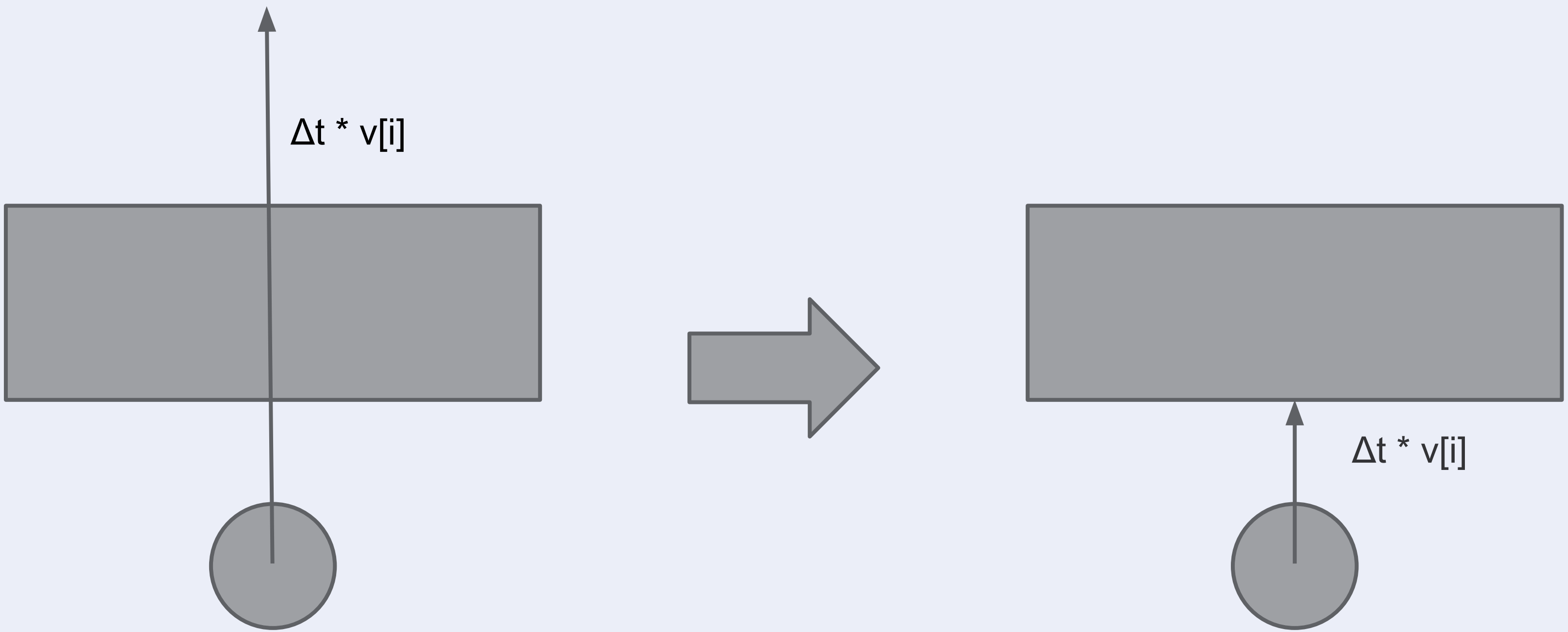


Interaction with Box2D

Prevent penetration:

If a particle is about to penetrate a Box2D body during this step, stop the particle at the surface

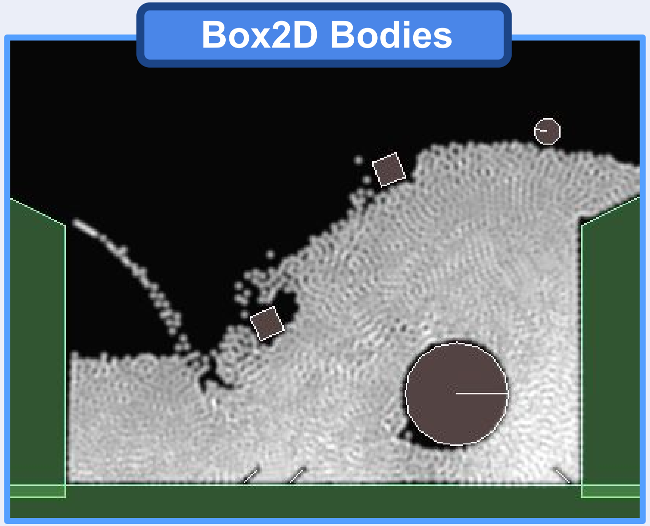


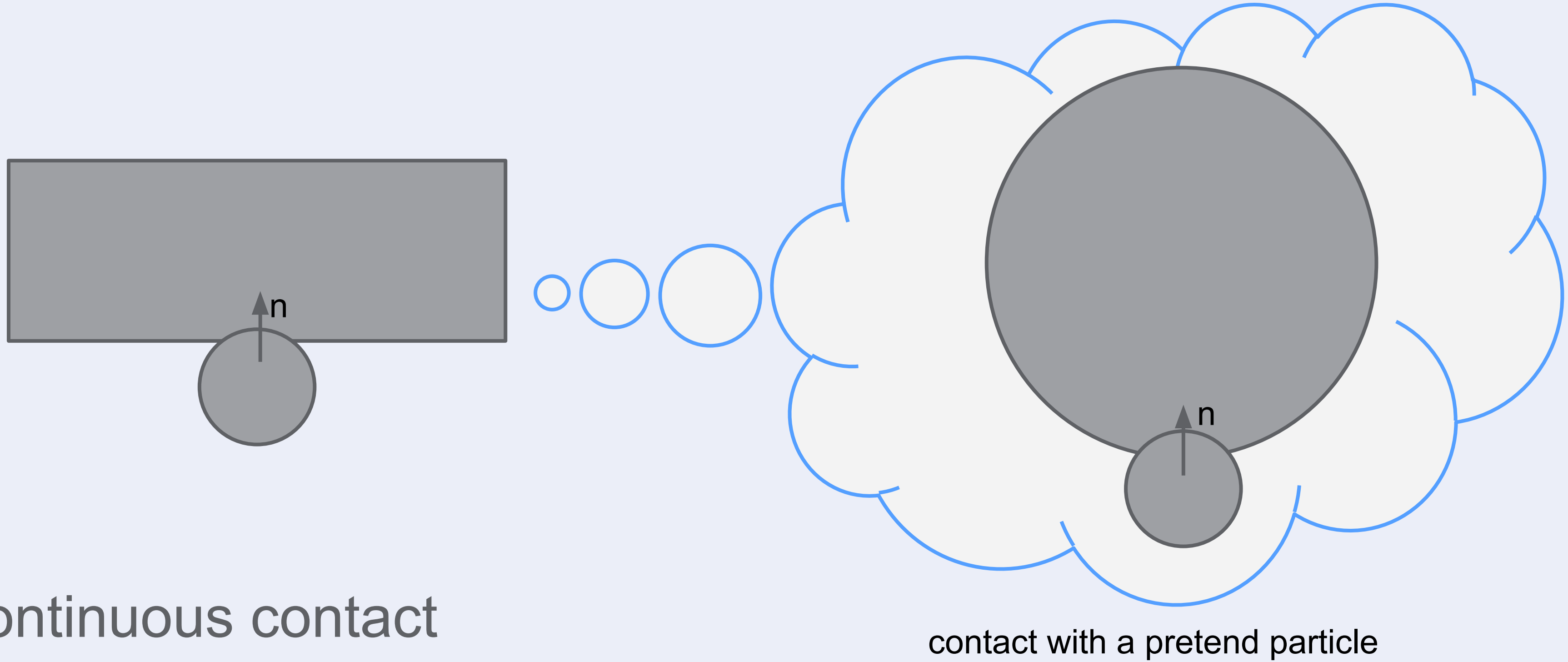


Prevent penetration

Interaction with Box2D

If a particle is in contact with a Box2D body, execute the subsequent particle simulation as if the particle is in contact with another particle





Continuous contact

Summary

- Particle systems are suitable for fluids, deformable objects, and interactions between different materials
- Collision testing is made tractable by ordering the particles top-to-bottom, left-to-right
- Viscous, spring, elastic, powder, and tensile forces can be added to create different kinds of particles