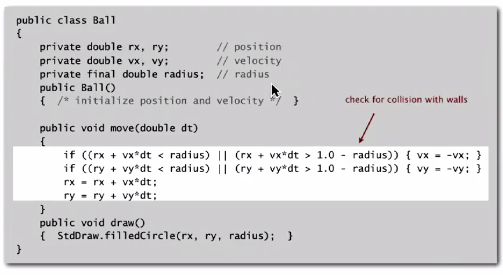
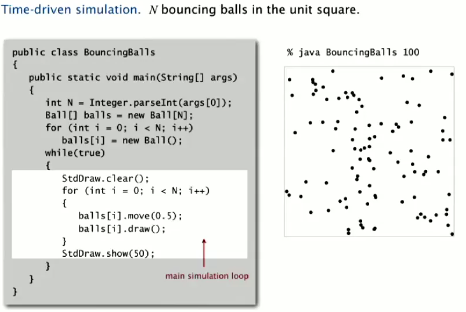
Event-Driven Simulation

*Study a property of the natural world by simulating it*

Simulate the motion of n moving particles:

* Hard disc model
  + Moving particles collide with each other and the walls
  + Each particle is a disc with known position, velocity and radius
  + No other forces
* Significance: macroscopic observables to microscopic dynamics
  + Macro: e.g. temp, pressure, diffusion constant, etc.
  + Micro: motion of individual atoms and molecules, etc.

Implementation with only collisions with walls (not other discs):



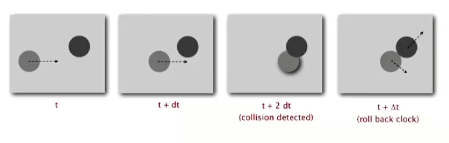
What’s missing from the implementation?

* Physics problems: When balls collide? What effect?
* CS problems: how and when do we do these computations for each ball? How can we do efficiently? N log N rather than N2 , which does not scale

Option 1:

Time-driven simulation:

* Update time every so many *dt* seconds
* Update position of all particles every *dt* seconds and check for overlaps
* If overlap, roll back clock to collision, update velocities of colliding particles and continue sim



Many problems:

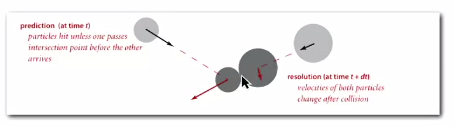
* N2 / 2 checks per time (slow)
* If N is small and *dt*  is small, there is too much computation
* If *dt* is too large, may miss collisions

Option 2 (BETTER):

Event-driven simulation:

*Change state only when something happens*

* Particles move in a straight line between collisions
* Focus only on times when collisions occur
* Maintain a PQ of collision events, prioritized by time
* Remove min = get next collision

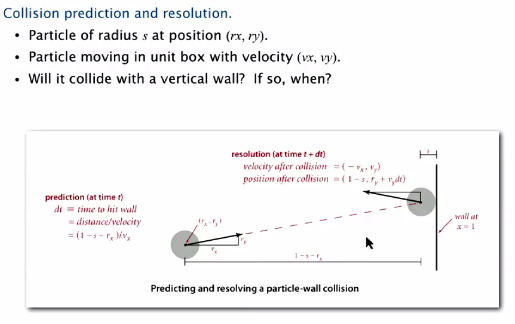


Need to compute the following:

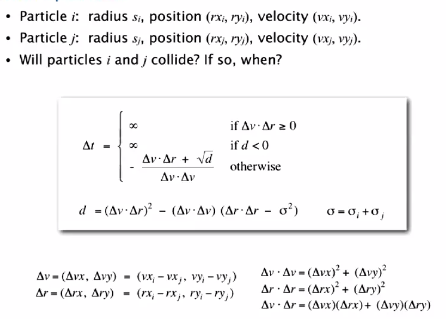
* Collision prediction: given position, velocity and radius of particle, when will it collide next with a wall or another particle?
* Collision resolution: If collision occurs, update colliding particle(s) according to laws of elastic conditions

Predicting a particle-wall collision:

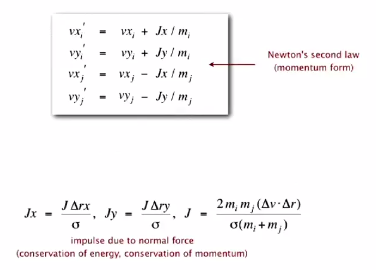
* Distance to wall divided by velocity == time
* When it impacts the wall, change the velocity



Predicting a particle-particle collision:

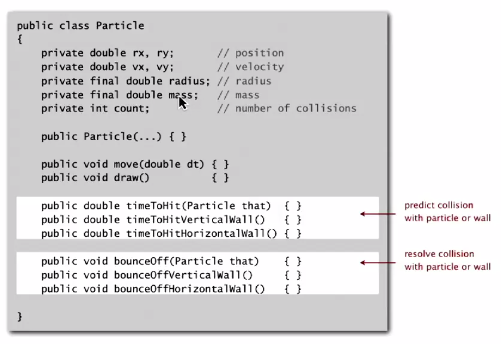


Collision resolution

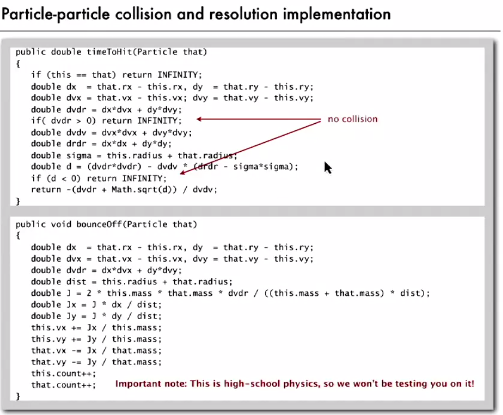


Particle data type:

\*\* Note: adds mass



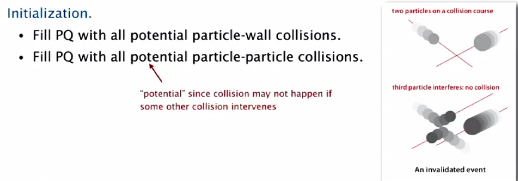
Has multiple procedures to predict collision and handle collision resolution. Implemented below:



Main loop:

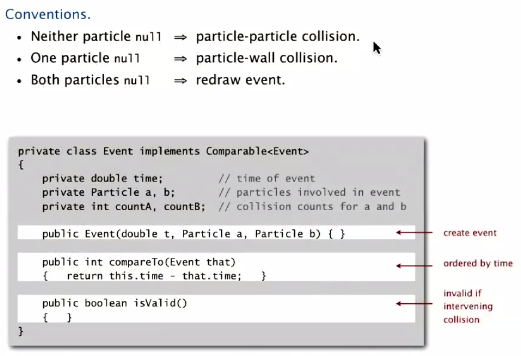
Initialization is quadratic:

* Fill PQ with all particle-wall collisions
* Fill PQ with all potential particle-particle collisions

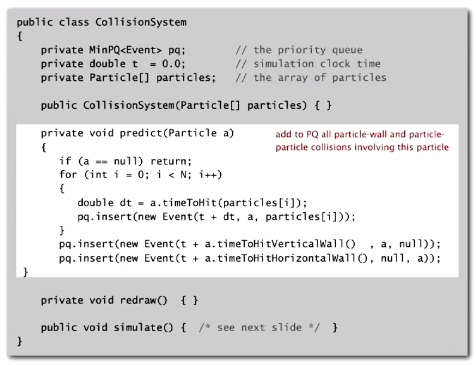


Process:

1. Take (delete) next event from PQ (the next collision)
2. Test whether event has been invalidated by another event
3. Go through all particles and change positions
4. Update velocities of two particles that collided
5. Predict future collisions of particles with walls and other particles, then push to PQ



Collision system implementation:



Full main loop implementation

