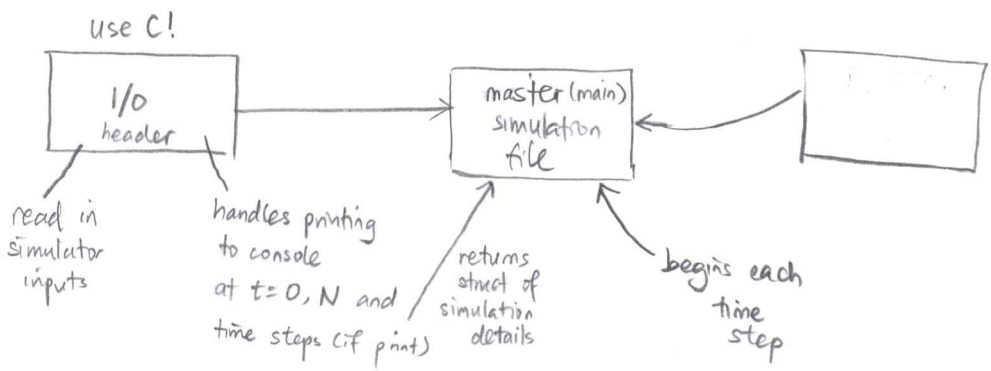
# Program Design

each C file has a header file

testwriter
generates input and output for tests




# Overall Simulation Design

Assumptions:

① Even though $\frac{L}{8r} \leq |\vec{v}| \leq \frac{L}{4}$,
$t \ll 1$ such that $|\vec{v}|t \ll L$
$\Rightarrow$ particles move only a v. small amount for each time step $\Rightarrow$ P(collision) < 1
$\Rightarrow \dot{N}_{collisions} \ll N$

② Particles will not collide twice with a wall in the same time step; if it happens we place the particle at the wall, to collide at time $T+1$

③ Need to check all $\frac{N(N-1)}{2}$ possible particle pairs, and additional $4N$ collisions with walls

$\Rightarrow$ Assume not.

Consider N particles in a line moving with the same $\vec{v}$

We can nudge one particle along the line $\Rightarrow$ collision with any of the other N-1 particles.

✕ Collision not detected since it can be any one of the N particles selected and any one of the (N-1) other particles colliding

④ $N \gg N_{cores} \Rightarrow$ not wise to simulate each particle in a seperate thread

Structs: simParams
particle
collision → vector<collision>

State Diagram

N, L, r, S, type

find candidates → filter candidates → resolve

initialise simulation parameters → initial particle data

YES → read particle data

NO → initialise with random positions and velocities

IO.c

print details of all particles

T=0 or T=N or type='print'

simulate time step T

T=0

simulator.c

O(N²)

all operations here are read-only

these loops are //able

i=0→N

For particle $p_i$

And other particle $p_j$

j=i→N

total $\frac{N(N-1)}{2}$ possible pairs

no

check if $p_i$ collides w walls

Determine if $p_i$, $p_j$ collide

Yes → enqueue j into array

No

enqueue w into array

array w collision pairs

O(N²)

N

0 N

$\frac{N}{2}$ threads each doing

N

"work" to check other particles

CS on array to prevent concurrent modification (mutex)

candidates.c

collision candidates

BAD IDEA!

sort collision candidates

potentially // ⇒ mutex on each particle

no items (empty)

//ise to resolve all valid collisions

//ise

avoids concurrent modificatio

sort by time then break ties by index

no

not empty

is valid collision?

yes

if both not collided

mark both particles as collided

schedule into final array

POSSIBLE RACE if we //ise and two threads try updating hasCollided of $P_i$

yes

if either collided

discard

filter.c

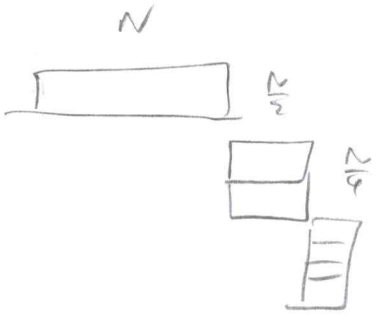
//ise

update pos's & velocities of all particles

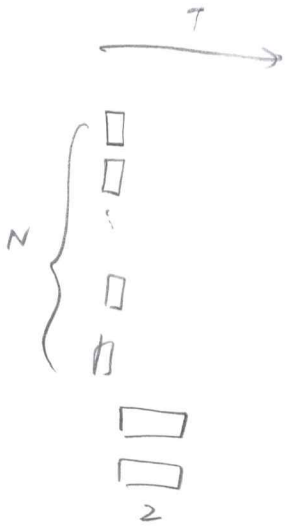
(incl those that do not collide

use true-false boolean array)

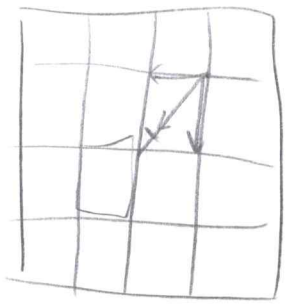
kinetics.c

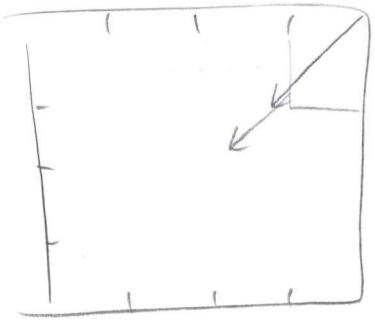



... $2N = \Theta(N)$

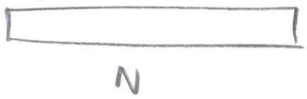


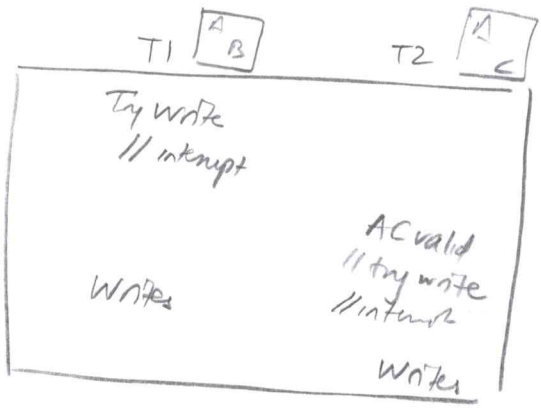

particle cycle can only collide with neighboring blocks



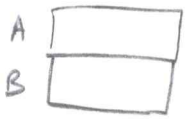


... $2N = \Theta(N)$.

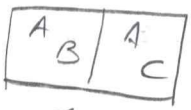



T1 value's get overwritten.

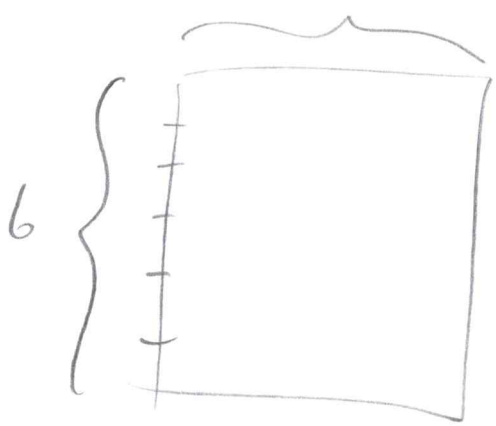


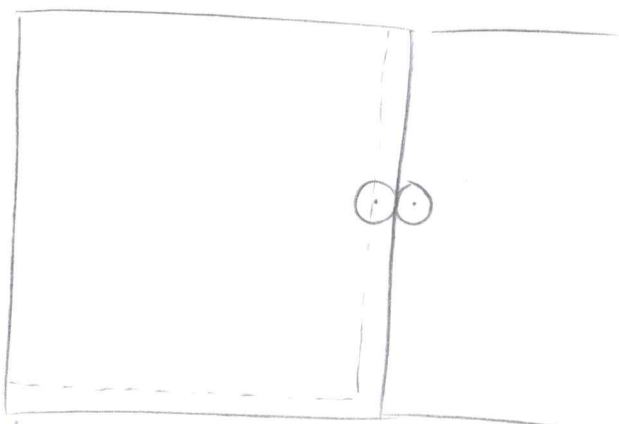

t=0 t=0.5

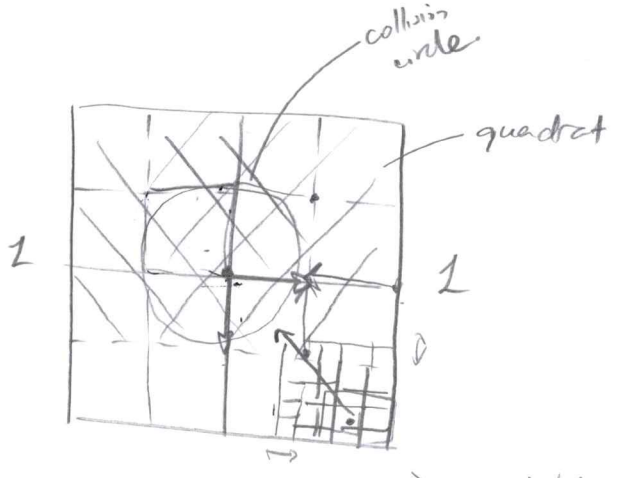



✓.

6

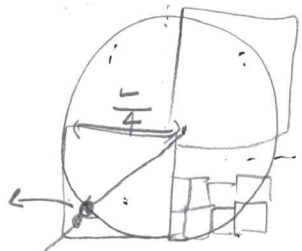


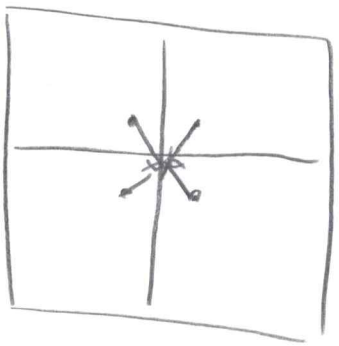



4 16 64 256 1024

$\frac{L}{4}$




each block can filter out some unlikely candidates.



$\left|\frac{L}{4}\right| = |\vec{v}|$

$\frac{1}{2}N^2 - \frac{N}{2}$

$\frac{N(N-1)}{2} \rightarrow |Q_i|$ · expected to 25% better?

$= \frac{N}{4} \cdot \left(\frac{25N}{64}\right)$ ← quad

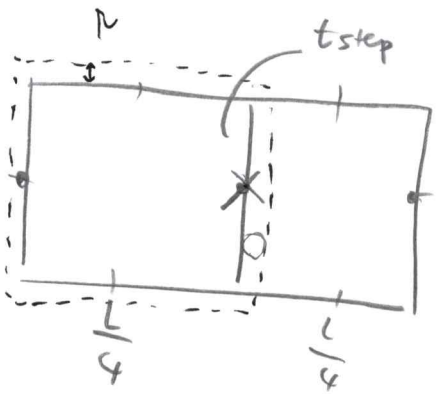
$= \frac{25N^2}{64}$

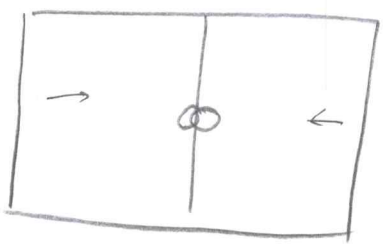
4×4 blocks of $L' = \frac{L}{4}$ for each → check collisions only in

$x \pm 1$
$y \pm 1$

"slightly" probabilistic

"average"



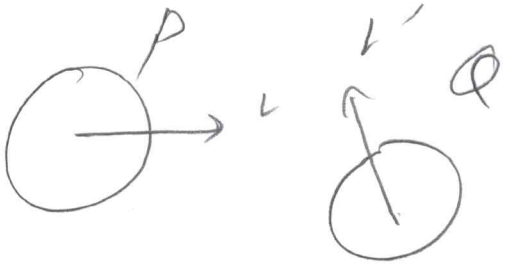



point particles-

$$\frac{\begin{cases} \text{best optimist} \\ \frac{N}{16}\left(\frac{4N}{16} + \frac{6N}{16} + \frac{6N}{16} + \frac{9N}{16}\right) \end{cases}}{4}$$

each quadrat of quadtree on avg requires checking

$= \frac{N}{4}\left(\frac{25N}{64}\right)$ collisions




$$2r = d = \sqrt{(x_p + \dot{x}_p \Delta t}$$

$$\sqrt{\left((x_q + \dot{x}_q \Delta t) - (y_p - \dot{y}_p \Delta t)\right)^2 + \left((y_q - \dot{y}_q \Delta t) - (y_p - \dot{y}_p \Delta t)\right)^2}$$

$$(\Delta x)_t^2 + (\Delta y)_t^2$$

Tues

Sequential.

io.c

candidates.c

filter.c

simulation.c ←

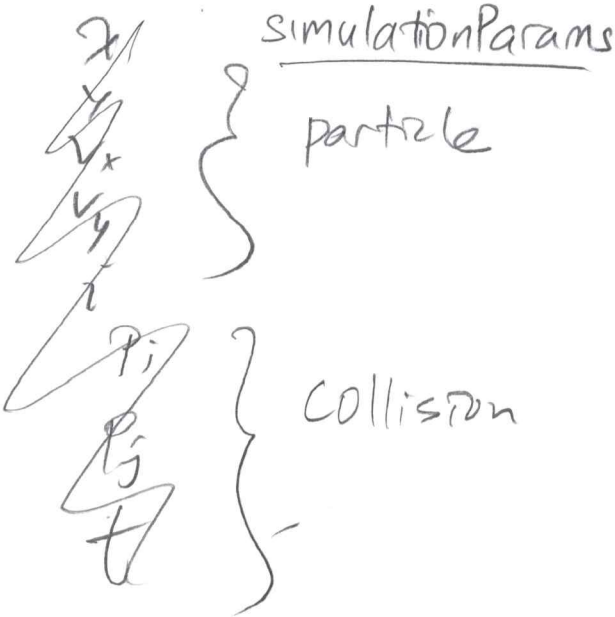
structs

random.c

collisions.c — collision detection fns

kinetics.c

structs




C++ { "vector" collisionCandidates
"vector" validCollisions

bool[] hasCollided.