Molecular dynamics simulation of hard discs

Goal. Simulate the motion of $N$ moving particles that behave according to the laws of elastic collision.
Time-driven simulation

- Discretize time in quanta of size $dt$.
- Update the position of each particle after every $dt$ units of time, and check for overlaps.
- If overlap, roll back the clock to the time of the collision, update the velocities of the colliding particles, and continue the simulation.
Time-driven simulation

Main drawbacks.

• $\sim N^2/2$ overlap checks per time quantum.
• Simulation is too slow if $dt$ is very small.
• May miss collisions if $dt$ is too large.
  (if colliding particles fail to overlap when we are looking)
Event-driven simulation

Change state only when something interesting happens.
- Between collisions, particles move in straight-line trajectories.
- Focus only on times when collisions occur.
- Maintain PQ of collision events, prioritized by time.
- Delete min = get next collision.

Collision prediction. Given position, velocity, and radius of a 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 collisions.
Collision system: event-driven simulation main loop

Initialization.
- Fill PQ with all potential particle-wall collisions.
- Fill PQ with all potential particle-particle collisions.

Main loop.
- Delete the impending event from PQ (min priority = $t$).
- If the event has been invalidated, ignore it.
- Advance all particles to time $t$, on a straight-line trajectory.
- Update the velocities of the colliding particle(s).
- Predict future particle-wall and particle-particle collisions involving the colliding particle(s) and insert events onto PQ.
Particle collision simulation: example 1

% java CollisionSystem 100
Particle collision simulation: example 2

% java CollisionSystem < billiards.txt
Particle collision simulation: example 3

% java CollisionSystem < brownian.txt
Particle collision simulation: example 4

% java CollisionSystem < diffusion.txt
2.4 Priority Queues

- API and elementary implementations
- Binary heaps
- Heapsort
- Event-driven simulation
Molecular dynamics simulation of hard discs

**Goal.** Simulate the motion of $N$ moving particles that behave according to the laws of elastic collision.
Molecular dynamics simulation of hard discs

Goal. Simulate the motion of $N$ moving particles that behave according to the laws of elastic collision.

Hard disc model.
- Moving particles interact via elastic collisions with each other and walls.
- Each particle is a disc with known position, velocity, mass, and radius.
- No other forces.

Significance. Relates macroscopic observables to microscopic dynamics.
- Einstein: explain Brownian motion of pollen grains.
Warmup: bouncing balls

**Time-driven simulation.** \( N \) bouncing balls in the unit square.

```java
public class BouncingBalls {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        Ball[] balls = new Ball[N];
        for (int i = 0; i < N; i++)
            balls[i] = new Ball();
        while (true)
        {
            StdDraw.clear();
            for (int i = 0; i < N; i++)
            {
                balls[i].move(0.5);
                balls[i].draw();
            }
            StdDraw.show(50);
        }
    }
}
```

% java BouncingBalls 100

main simulation loop
Warmup: bouncing balls

```java
public class Ball {
    private double rx, ry; // position
    private double vx, vy; // velocity
    private final double radius; // radius
    public Ball(...) {
        /* initialize position and velocity */
    }

    public void move(double dt) {
        if ((rx + vx*dt < radius) || (rx + vx*dt > 1.0 - radius)) { vx = -vx; }
        if ((ry + vy*dt < radius) || (ry + vy*dt > 1.0 - radius)) { vy = -vy; }
        rx = rx + vx*dt;
        ry = ry + vy*dt;
    }

    public void draw() {
        StdDraw.filledCircle(rx, ry, radius);
    }
}
```

**Missing.** Check for balls colliding with each other.
- Physics problems: when? what effect?
- CS problems: which object does the check? too many checks?
Time-driven simulation

- Discretize time in quanta of size $dt$.
- Update the position of each particle after every $dt$ units of time, and check for overlaps.
- If overlap, roll back the clock to the time of the collision, update the velocities of the colliding particles, and continue the simulation.

![Diagram](image-url)
Main drawbacks.
- $\sim N^2/2$ overlap checks per time quantum.
- Simulation is too slow if $dt$ is very small.
- May miss collisions if $dt$ is too large.
  (if colliding particles fail to overlap when we are looking)
Event-driven simulation

Change state only when something interesting happens.
- Between collisions, particles move in straight-line trajectories.
- Focus only on times when collisions occur.
- Maintain PQ of collision events, prioritized by time.
- Delete min = get next collision.

Collision prediction. Given position, velocity, and radius of a 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 collisions.
Collision prediction and resolution.

- Particle of radius $s$ at position $(r_x, r_y)$.
- Particle moving in unit box with velocity $(v_x, v_y)$.
- Will it collide with a vertical wall? If so, when?

**Prediction (at time $t$)**

$$dt = \text{time to hit wall} = \frac{\text{distance/velocity}}{v_x} = \frac{1 - s - r_x}{v_x}$$

**Resolution (at time $t + dt$)**

$$\text{velocity after collision} = (-v_x, v_y)$$

$$\text{position after collision} = (1 - s, r_y + v_y dt)$$
Particle-particle collision prediction

Collision prediction.

- Particle $i$: radius $s_i$, position $(rx_i, ry_i)$, velocity $(vx_i, vy_i)$.
- Particle $j$: radius $s_j$, position $(rx_j, ry_j)$, velocity $(vx_j, vy_j)$.
- Will particles $i$ and $j$ collide? If so, when?
Particle-particle collision prediction

Collision prediction.

- Particle $i$: radius $s_i$, position $(rx_i, ry_i)$, velocity $(vx_i, vy_i)$.
- Particle $j$: radius $s_j$, position $(rx_j, ry_j)$, velocity $(vx_j, vy_j)$.
- Will particles $i$ and $j$ collide? If so, when?

\[
\Delta t = \begin{cases} 
\infty & \text{if } \Delta v \cdot \Delta r \geq 0, \\
\infty & \text{if } d < 0, \\
- \frac{\Delta v \cdot \Delta r + \sqrt{d}}{\Delta v \cdot \Delta v} & \text{otherwise}
\end{cases}
\]

\[
d = (\Delta v \cdot \Delta r)^2 - (\Delta v \cdot \Delta v) (\Delta r \cdot \Delta r - s^2), \quad s = s_i + s_j
\]

\[
\Delta v = (\Delta vx, \Delta vy) = (vx_i - vx_j, vy_i - vy_j) \quad \Delta v \cdot \Delta v = (\Delta vx)^2 + (\Delta vy)^2
\]

\[
\Delta r = (\Delta rx, \Delta ry) = (rx_i - rx_j, ry_i - ry_j) \quad \Delta r \cdot \Delta r = (\Delta rx)^2 + (\Delta ry)^2
\]

\[
\Delta v \cdot \Delta r = (\Delta vx)(\Delta rx) + (\Delta vy)(\Delta ry)
\]

Important note: This is physics, so we won’t be testing you on it!
Particle-particle collision resolution

Collision resolution. When two particles collide, how does velocity change?

\[
\begin{align*}
    v_{x_i}' &= v_{x_i} + \frac{Jx}{m_i} \\
    v_{y_i}' &= v_{y_i} + \frac{Jy}{m_i} \\
    v_{x_j}' &= v_{x_j} - \frac{Jx}{m_j} \\
    v_{y_j}' &= v_{y_j} - \frac{Jy}{m_j}
\end{align*}
\]

Newton's second law
(momentum form)

\[
J_x = J \frac{\Delta r_x}{s}, \quad J_y = J \frac{\Delta r_y}{s}, \quad J = \frac{2 m_i m_j (\Delta v \cdot \Delta r)}{s (m_i + m_j)}
\]

Impulse due to normal force
(conservation of energy, conservation of momentum)

Important note: This is physics, so we won’t be testing you on it!
public class Particle
{
    private double rx, ry;       // position
    private double vx, vy;       // velocity
    private final double radius; // radius
    private final double mass;   // mass
    private int count;           // number of collisions

    public Particle( ... ) { ... }

    public void move(double dt) { ... }
    public void draw() { ... }

    public double timeToHit(Particle that) { }
    public double timeToHitVerticalWall() { }
    public double timeToHitHorizontalWall() { }

    public void bounceOff(Particle that) { }
    public void bounceOffVerticalWall() { }
    public void bounceOffHorizontalWall() { }

    http://algs4.cs.princeton.edu/61event/Particle.java.html
Collision system: event-driven simulation main loop

Initialization.

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

Main loop.

- Delete the impending event from PQ (min priority = t).
- If the event has been invalidated, ignore it.
- Advance all particles to time $t$, on a straight-line trajectory.
- Update the velocities of the colliding particle(s).
- Predict future particle-wall and particle-particle collisions involving the colliding particle(s) and insert events onto PQ.
Event data type

Conventions.

- Neither particle null ⇒ particle-particle collision.
- One particle null ⇒ particle-wall collision.
- Both particles null ⇒ redraw event.

```java
private static class Event implements Comparable<Event> {
    private final double time;       // time of event
    private final Particle a, b;     // particles involved in event
    private final int countA, countB; // collision counts of a and b

    public Event(double t, Particle a, Particle b) {
        ... }

    public int compareTo(Event that) {
        return this.time - that.time;
    }

    public boolean isValid() {
        return ... }
}
```
Particle collision simulation: example 1

% java CollisionSystem 100
% java CollisionSystem < billiards.txt
Particle collision simulation: example 3

% java CollisionSystem < brownian.txt
Particle collision simulation: example 4

% java CollisionSystem < diffusion.txt