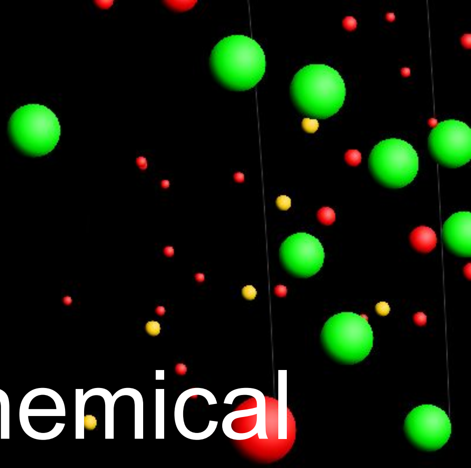
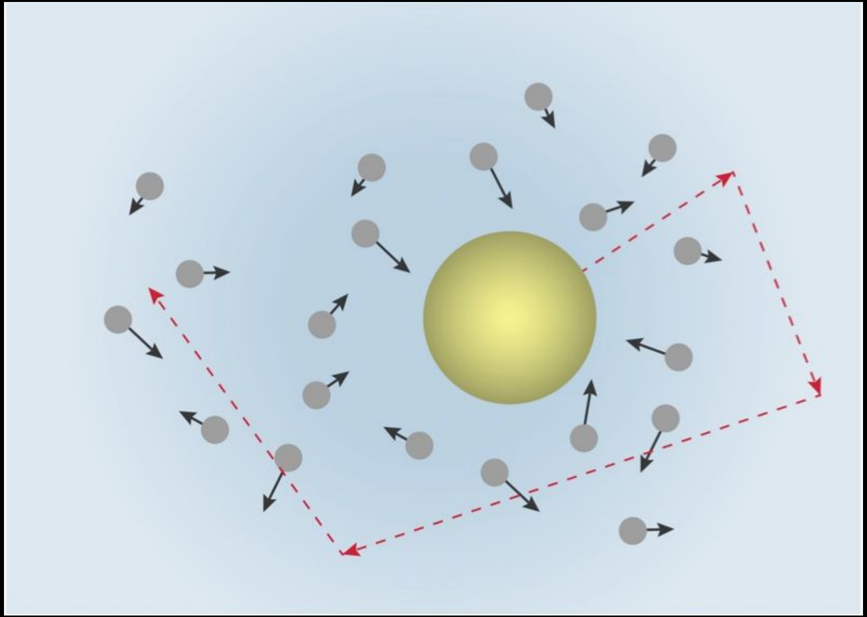
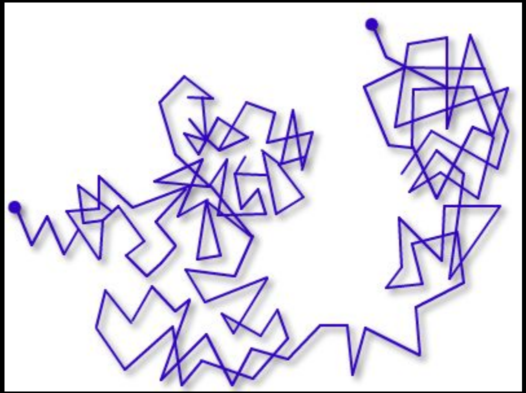


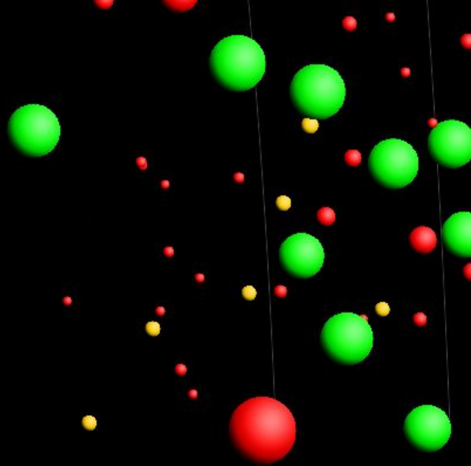
A stochastic particle-based chemical system simulator for the web

By Herman Bergwerf

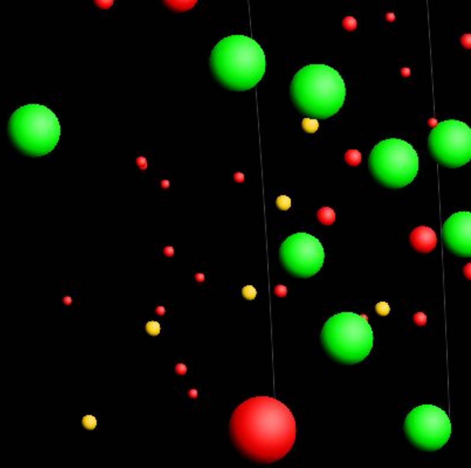
Brownian motion



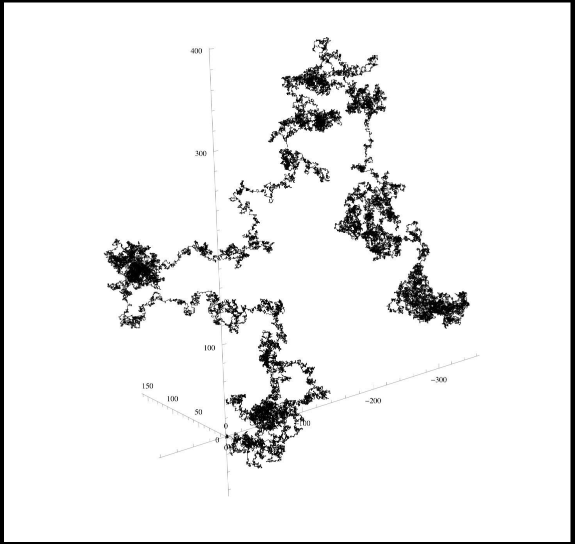


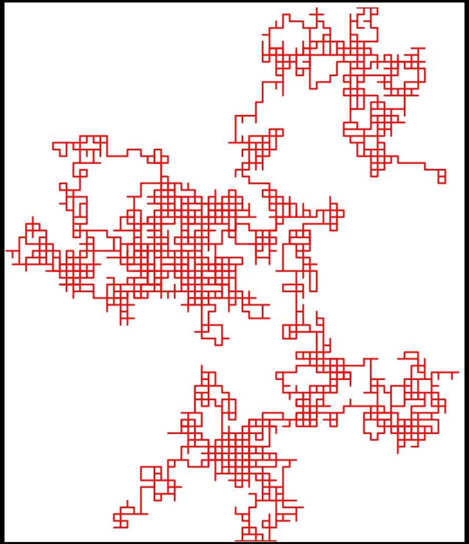


Random walk

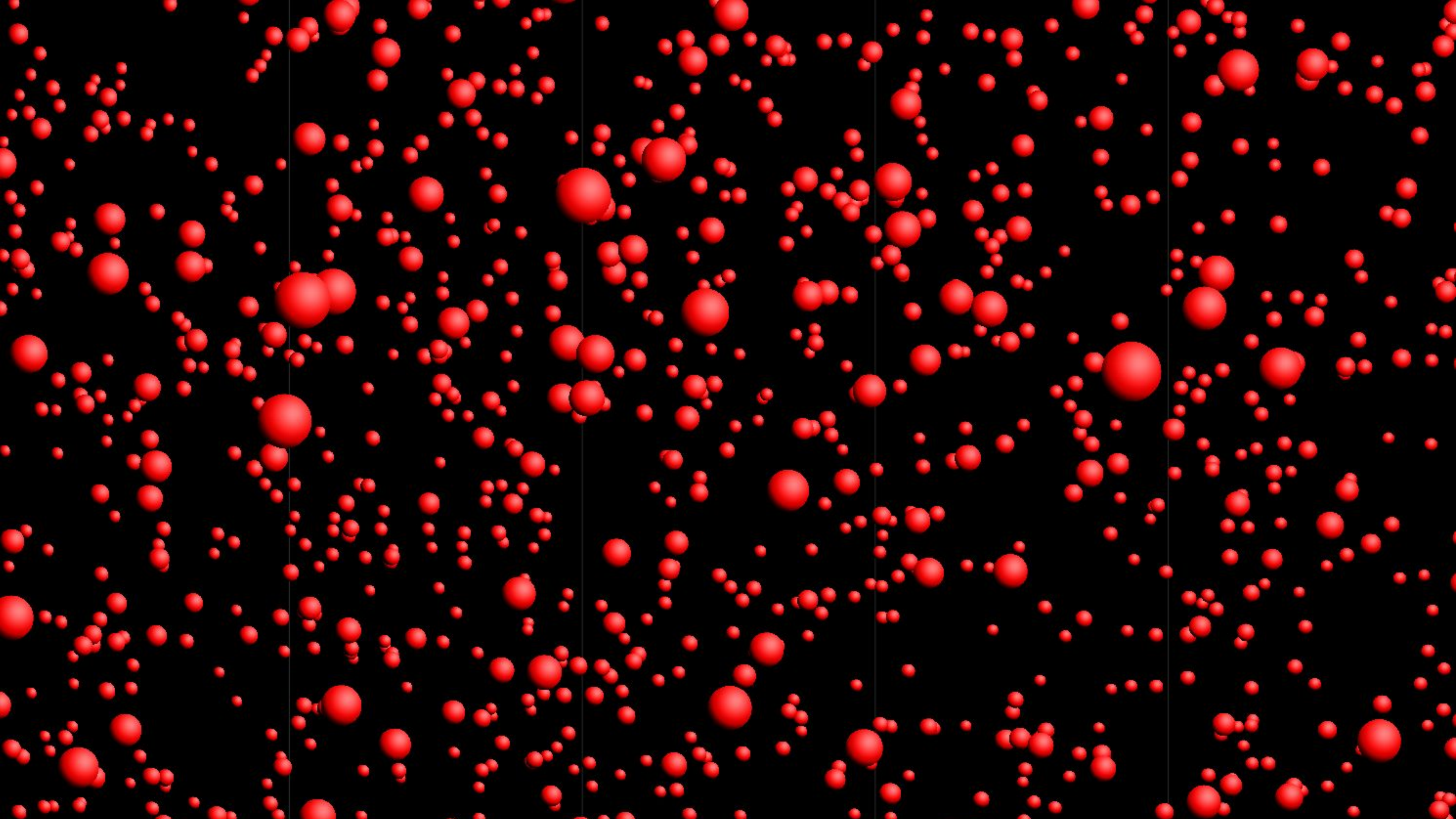


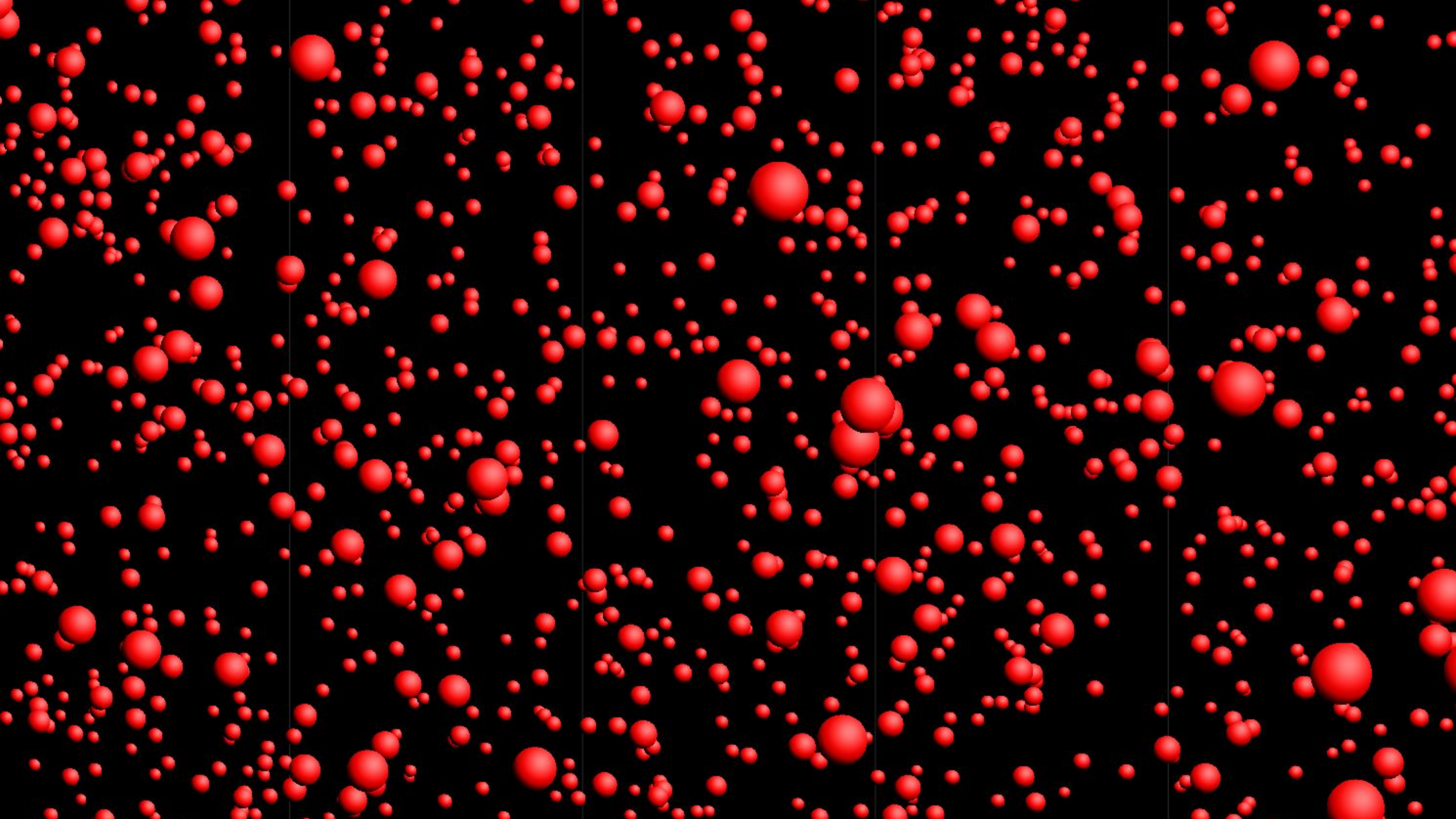
Every step, choose a random new direction and speed*

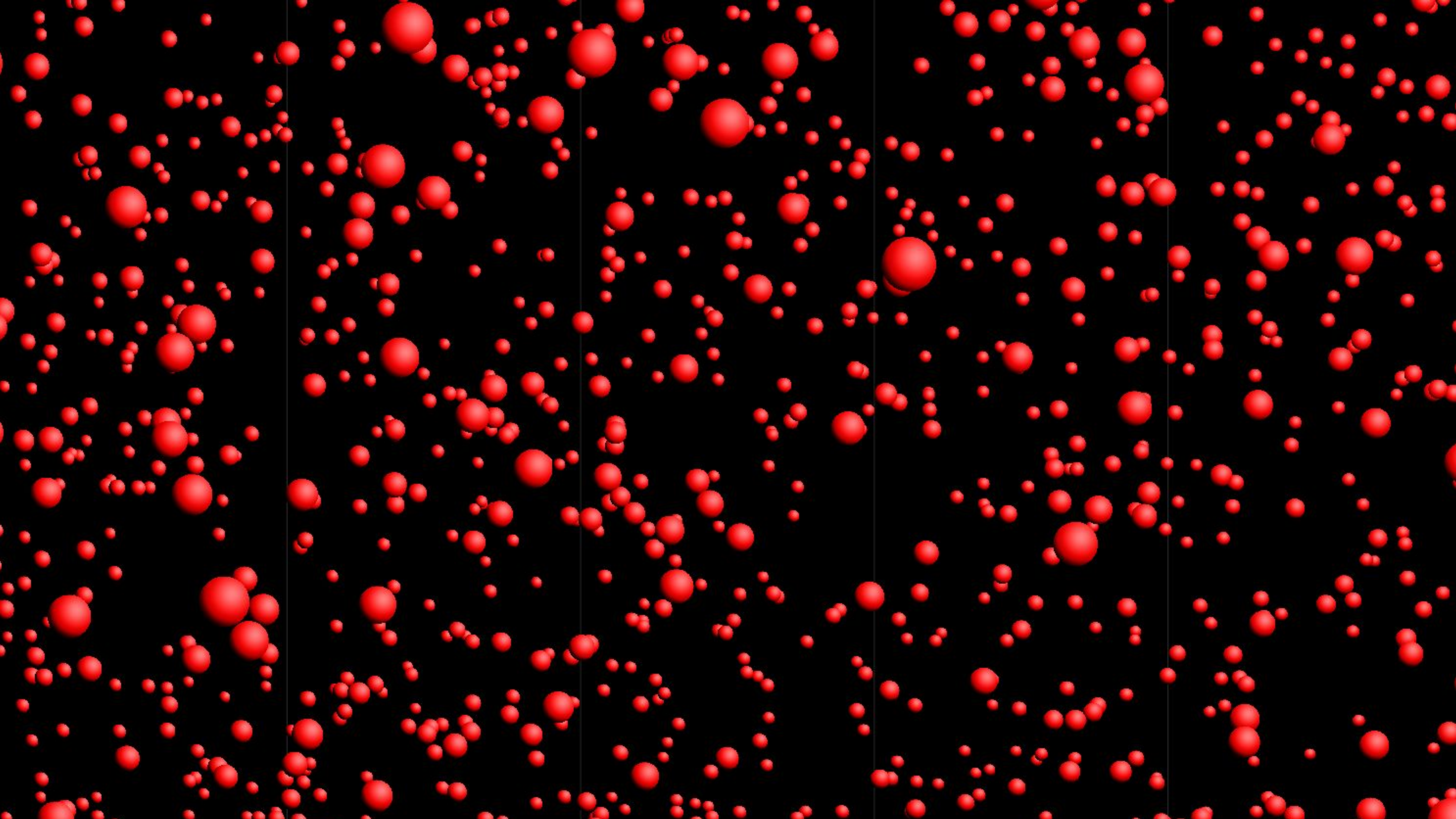




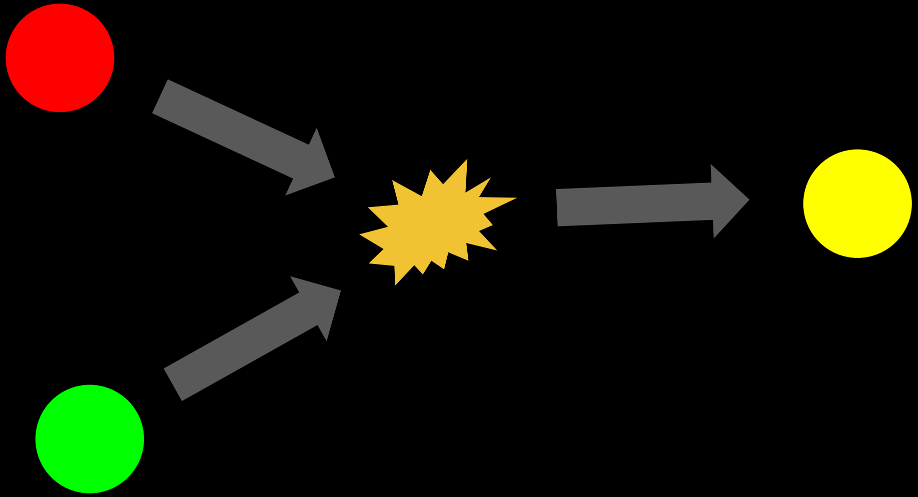
**Distributed around an average value (similar to Gaussian distribution)*

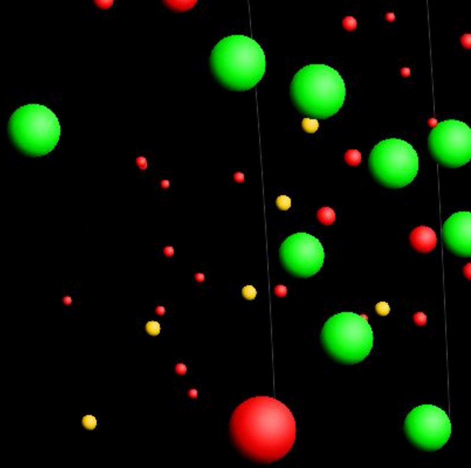


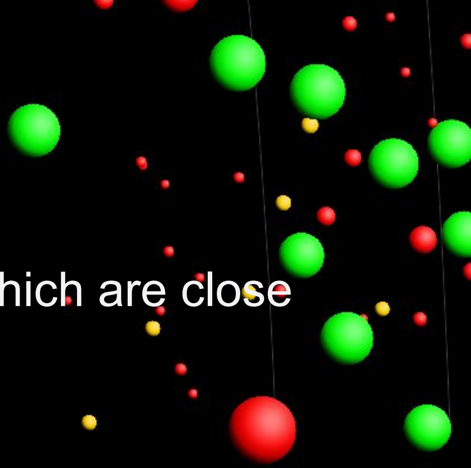




Reactions



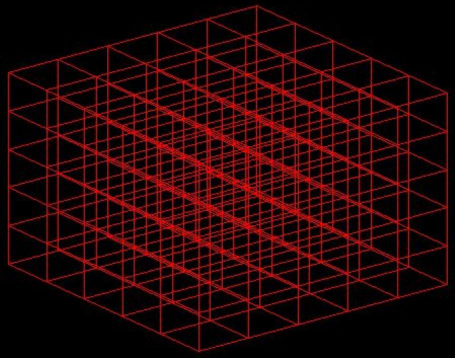




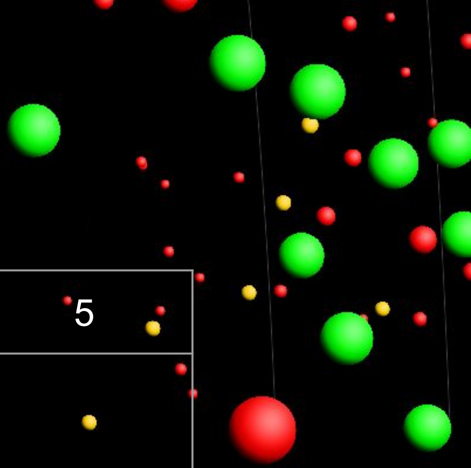
Collision algorithm

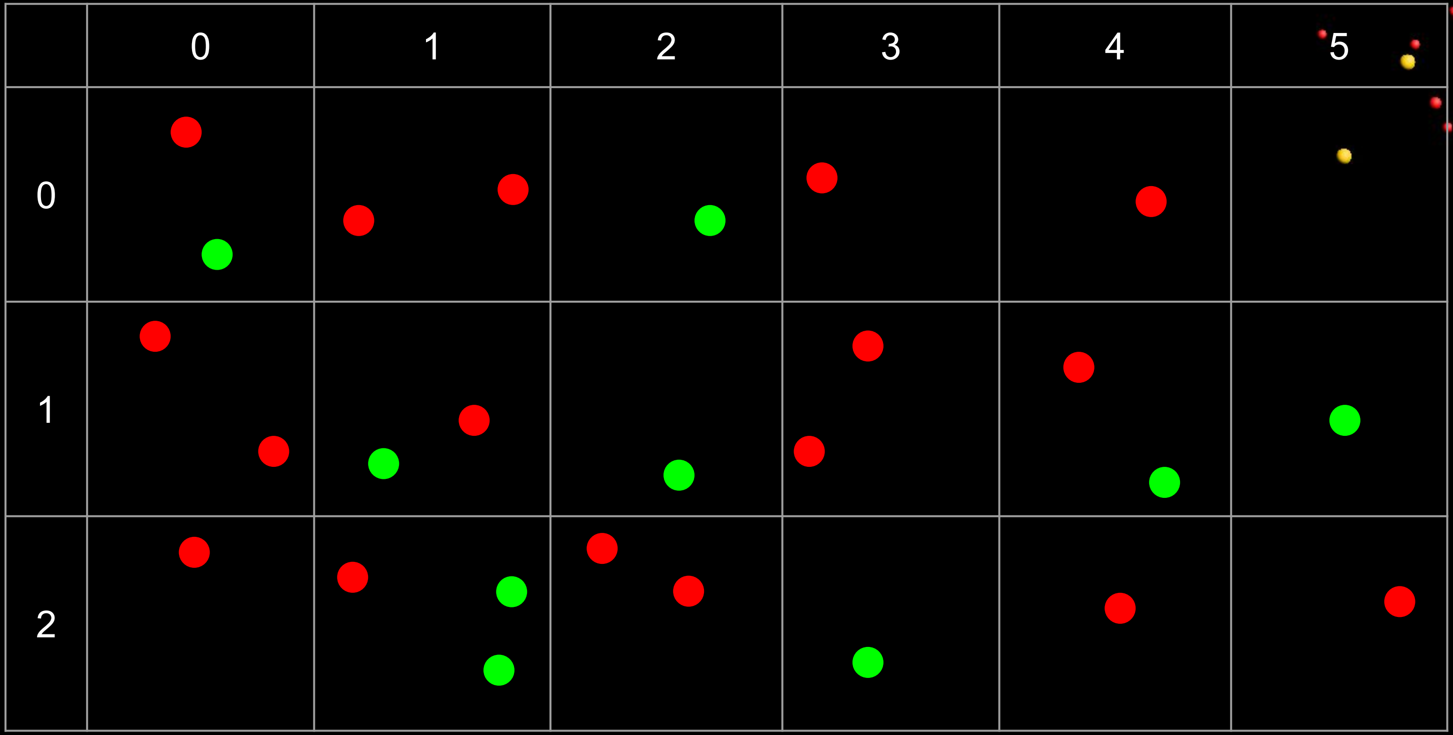
Problem: computing the distances between all particles to see which are close enough to collide is very slow (N^2), so:

1. Compute a voxel for each particle
2. Only reaction with other particles in the same voxel
3. Less accurate, but way faster

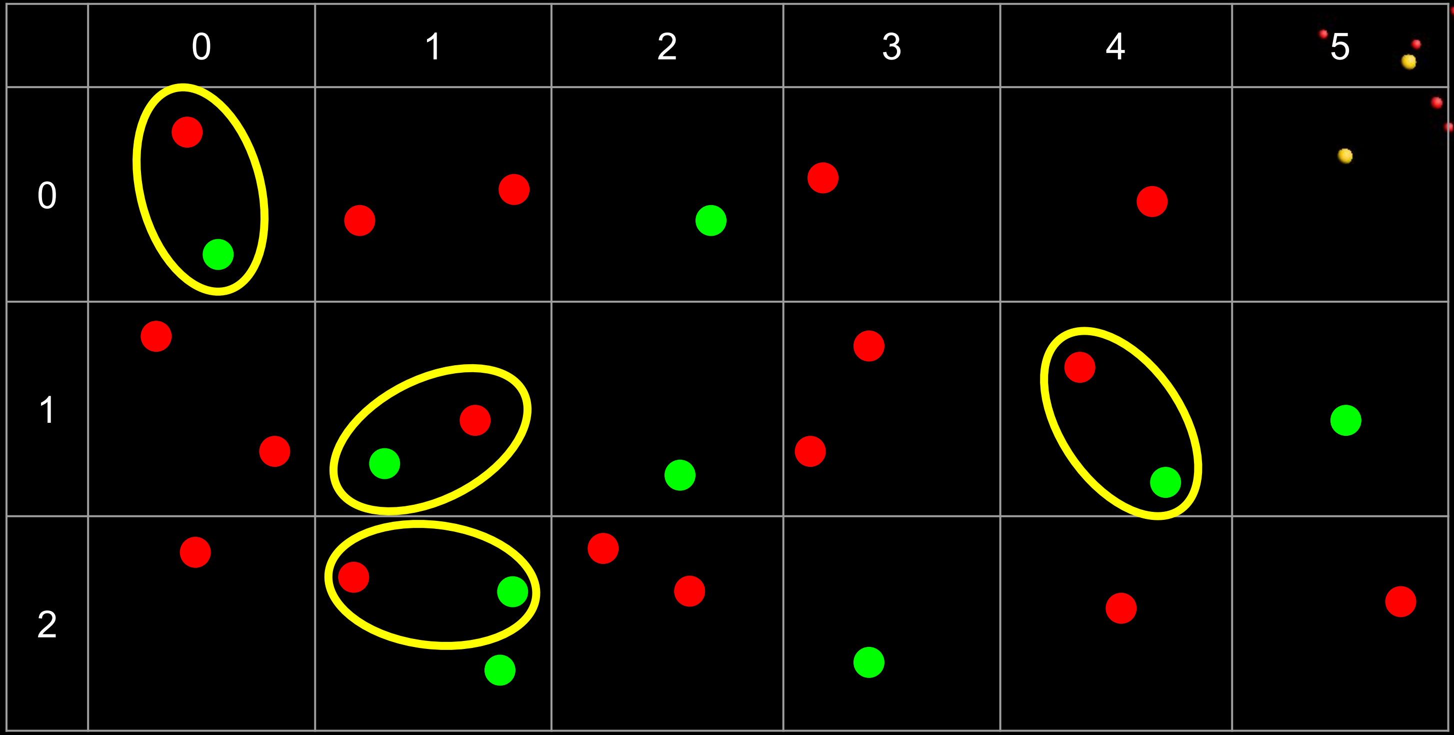


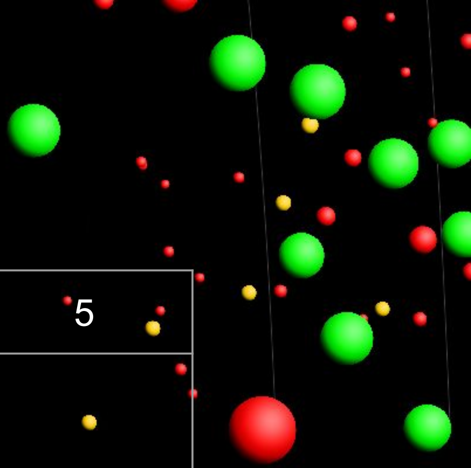
Particles in voxels: 2D example





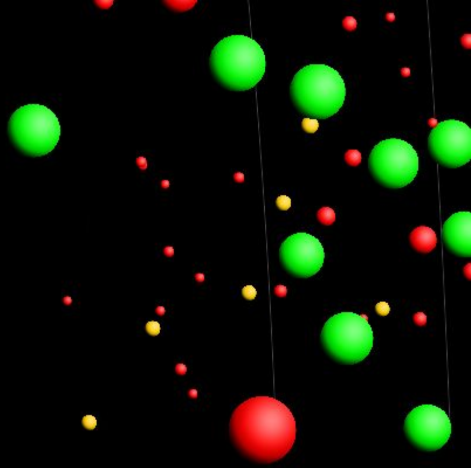
Particles in voxels: 2D example





Data structure

1. **First pass:** assign voxel position to each particle
2. **Second pass:** find particles that are in the same voxel



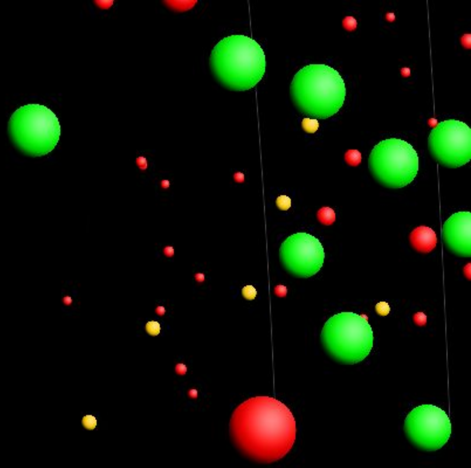
Slow data structure for this:

```
HashMap<int x, HashMap<int y, HashMap<int z, List<Particle>>>>
```

Alternative

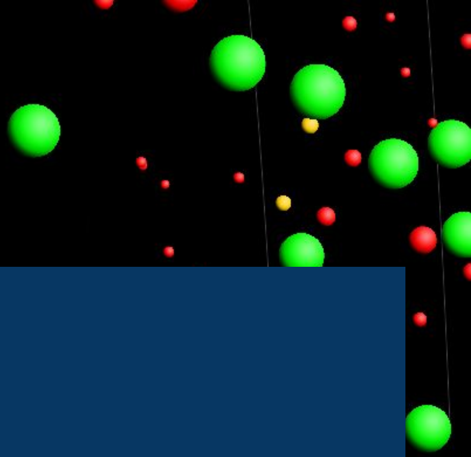
- 10 bits per position
- packed together: 30 bits to represent a voxel position.

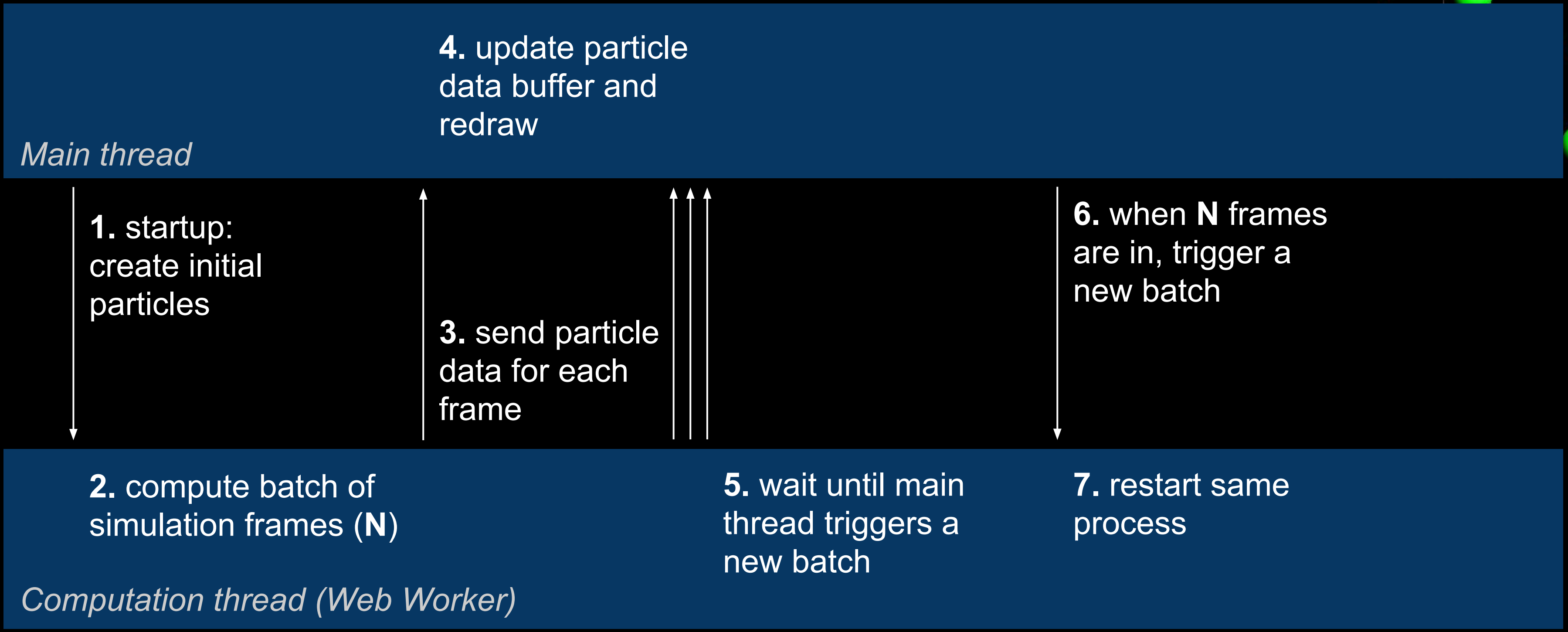
Faster data structures:



1. `HashMap<int position, List<Particle>>`
2. `List<Tuple2<int position, Particle>>`
   a. Sort by position
   b. Iterate through array: particles that are in the same voxel are next to each other in the array!
   c. Remember sorting order as starting points for the next cycle, particles move only slightly so the sorting order doesn’t change that much (small but significant difference for larger number of particles, **`>10k`**)

Web Workers (multi-threading, kinda)





4. update particle data buffer and redraw

Main thread

1. startup: create initial particles

3. send particle data for each frame

6. when **N** frames are in, trigger a new batch

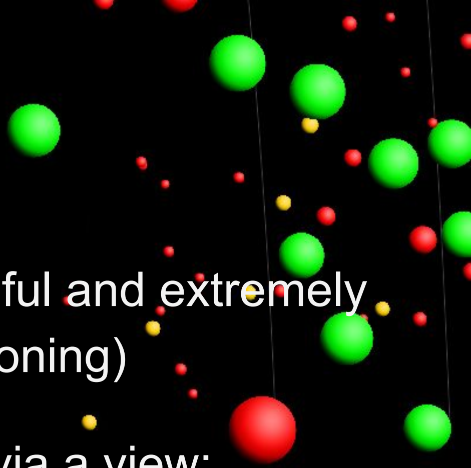
2. compute batch of simulation frames (**N**)

5. wait until main thread triggers a new batch

7. restart same process

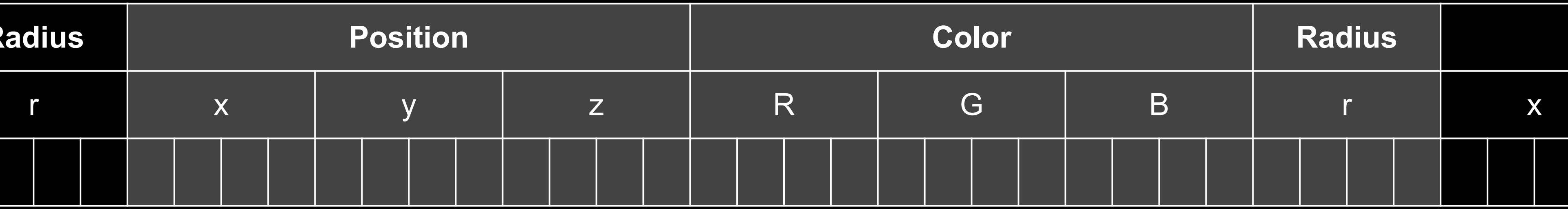
Computation thread (Web Worker)

ByteBuffer



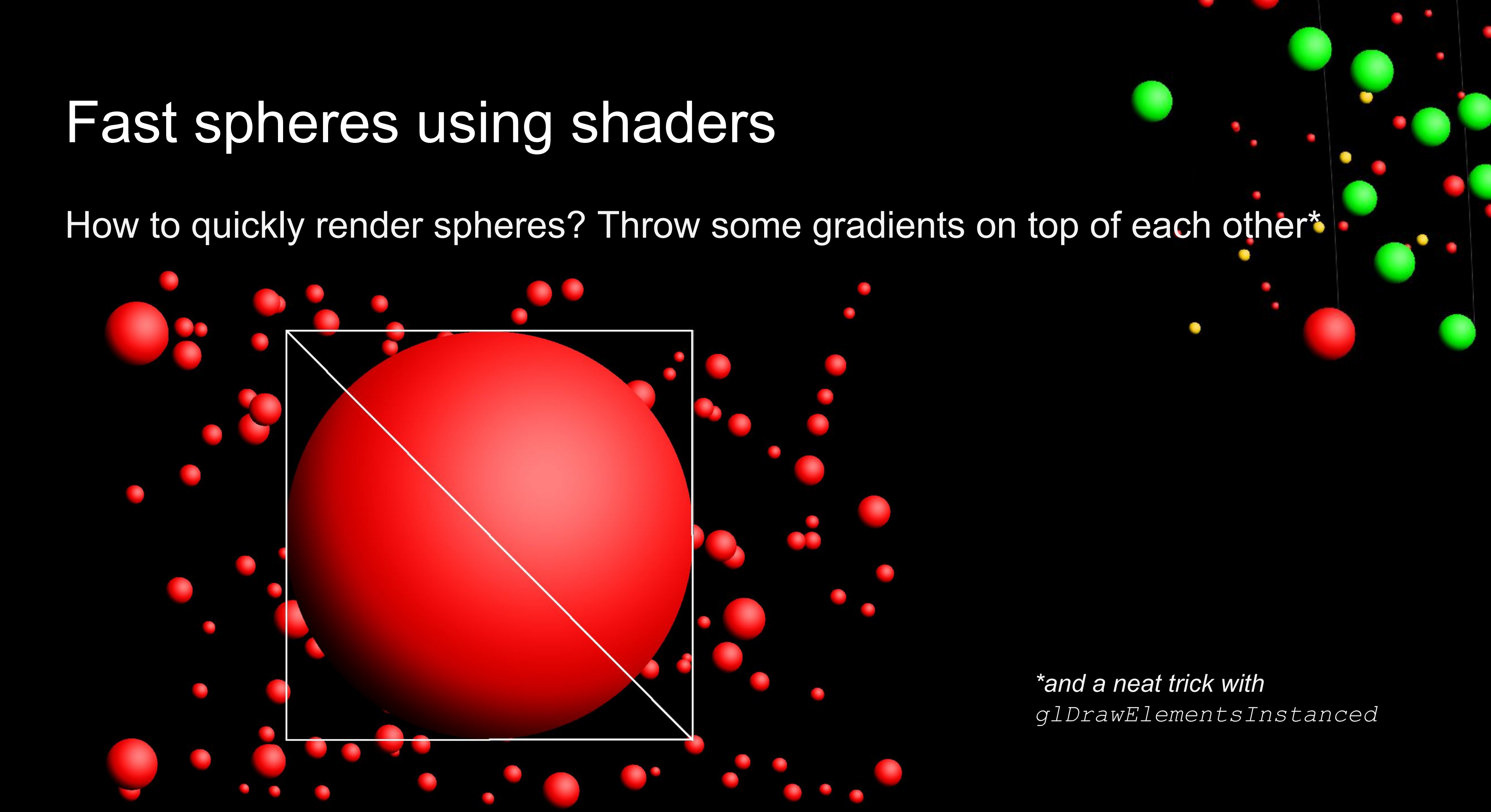
Passing an array with 100.000 particles to another thread is painful and extremely slow in JavaScript, but fortunately we have ArrayBuffer (faster cloning)

All data is constantly packed into an ArrayBuffer, and accessed via a view:



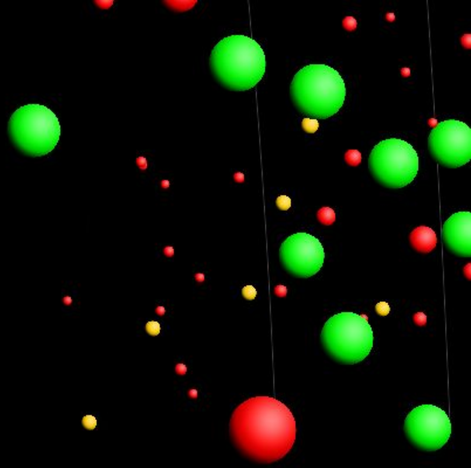
Fast spheres using shaders

How to quickly render spheres? Throw some gradients on top of each other*



**and a neat trick with* `glDrawElementsInstanced`

More information



Source code:

github.com/molview/bromium (written in the Dart language)

Live demo:

molview.github.io/bromium-deploy/