Particle collisions

Andone Alexandru

June 2022

1 The problem

The problem I am trying to solve is a collision detection problem. Given a canvas of any size and a number of particles randomly generated and randomly moving, I want to highlight all the particles that touch each other.

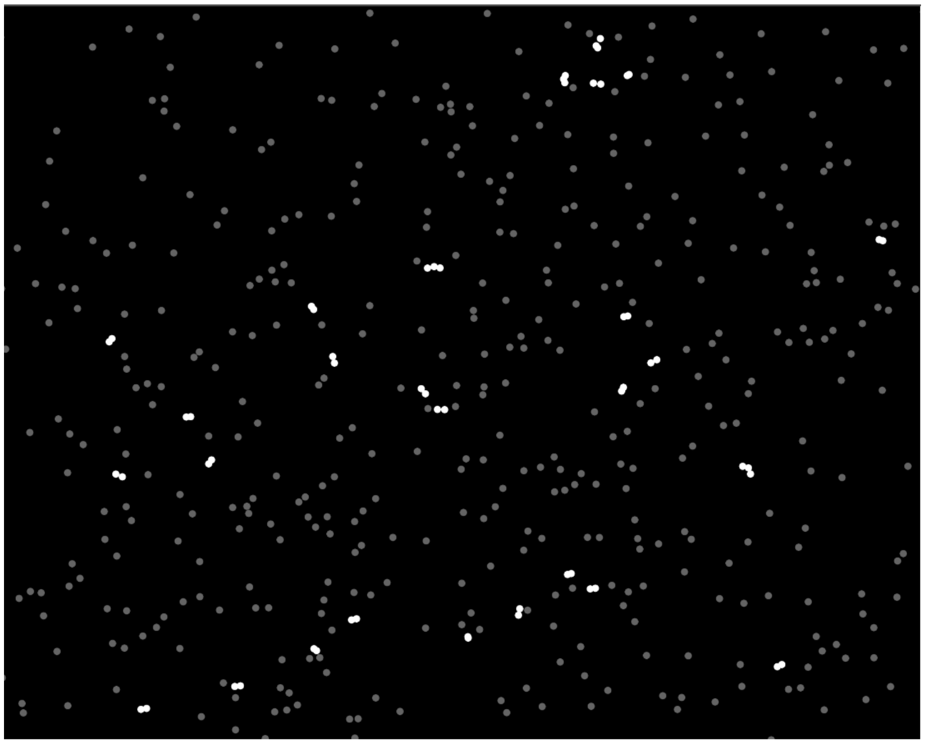

Figure 1: Particles touching each other are highlighted

This would be a naive implementation with the complexity of O(n^2):

```
for particle in particles:
    for otherParticle in particles:
        if particle != otherParticle and
           particle.intersects(otherParticle):
            particle.highlight()
```

There are some optimisations that can be done, but this is still the overall idea, and the complexity remains O(n^2). This algorithm checks for every particle if it intersects another particle.

2 Optimisations that can be done

The main problem with the naive algorithm is that for a particle, it checks every other particle if it intersects it, including the ones that are far away. So, a simple approach is to create an algorithm that for a particle, checks only the particles that are nearby for collisions. One great idea for this kind of algorithm is a quadtree.

3 What is a quadtree?

A quadtree is a tree data structure in which each internal node has exactly four children. This data structure is most often used to partition a two-dimensional space by recursively subdividing it into four quadrants.

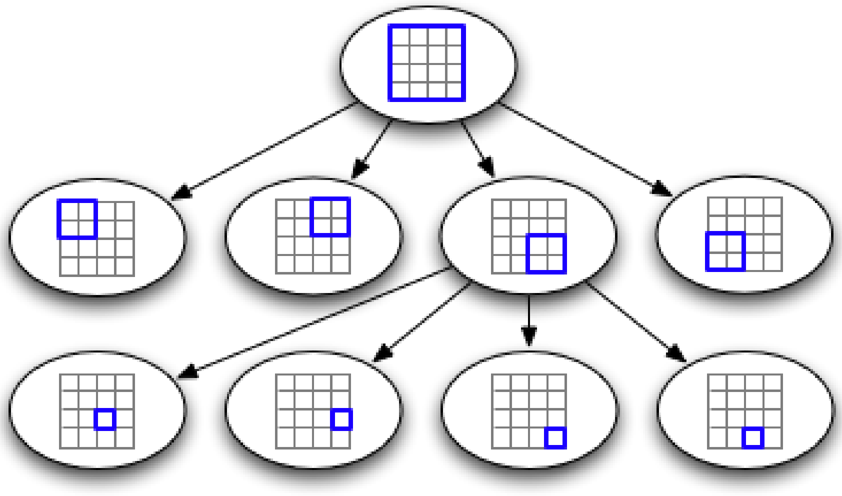

Figure 2: A quadtree recursively dividing a two-dimensional space

4 Quadtrees in our problem

How can we use quadtrees in our problem? Dividing the canvas into regions of particles will allow us to check for a particle only the particles in the nearby regions for collisions, instead of all the particles. We can also only divide a region if it contains a certain number of particles, so that where there are more particles close to each other, there are more regions, and where there are less particles close to each other, there are less regions. With these changes, we can reduce the complexity of the problem all the way to O($n * log(n)$)

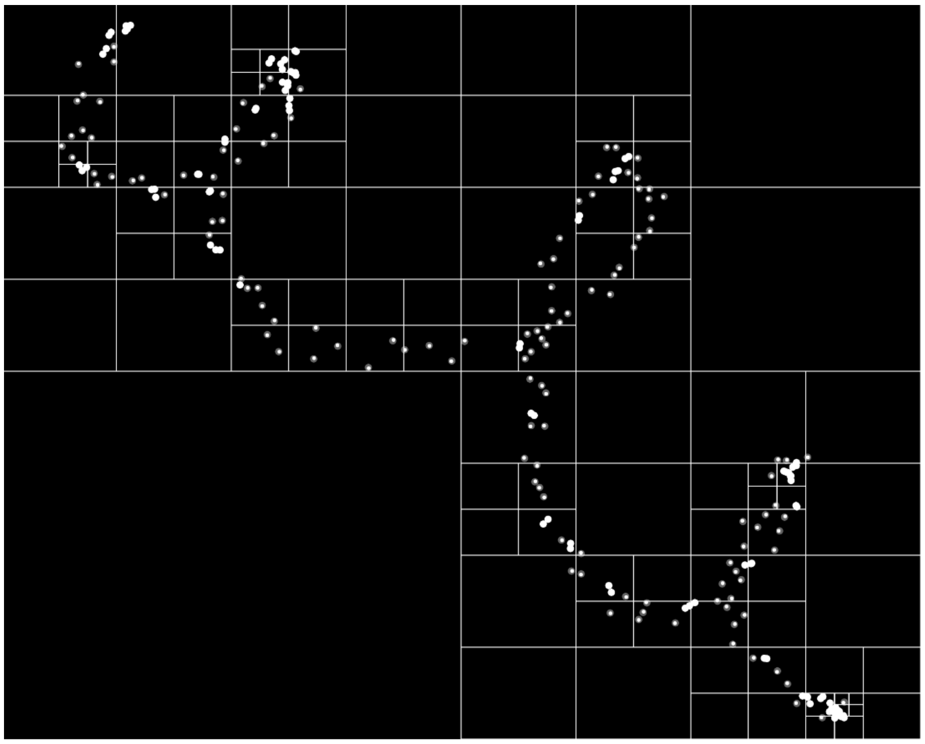

Figure 3: A quadtree dividing a two-dimensional space according to the number of particles