Exercise 8

Implement simulation of 2D disks moving inside a square box with elastic collisions.

1. The program takes two command line arguments: the number of disks N and the disk radius R.
2. A disk is characterized by its position and velocity. (The radius of all disks is the same.)
3. All disks are moving inside the $[0,1000] \times [0,1000]$ box.
4. The disks are generated in the following way:
    a. The center of the i-th disk is generated according to the uniform distribution in $[R, 1000 - R] \times [R, 1000 - R]$. The generation of the i-th disk center repeats until the i-th disk does not overlap with the disks $1, \dots, i - 1$.
    b. The direction α of the disk velocity is generated according to the uniform distribution in $[0,2\pi)$. Then, the disk velocity is computed as $100 \cdot (\cos\alpha, \sin\alpha)$.
    c. To generate uniformly distributed numbers, the class `std::uniform_real_distribution` can be utilized.
5. When a disk does not collide with other disks, no forces are applied on it. That means, it moves with a constant velocity.
6. Collisions with the box walls are processed as follows:
    a. If the x-coordinate of the disk center is out of range $[R, 1000 - R]$, the x-coordinate of the disk velocity is inverted.
    b. If the y-coordinate of the disk center is out of range $[R, 1000 - R]$, the y-coordinate of the disk velocity is inverted.
    c. The disk position is not changed.
7. Collisions of two disks are processed as follows:
    a. A collision is detected If the distance between the centers of two disks is smaller than $2R$.
    b. The collision line is defined as the line connecting the centers of these disks.
    c. The velocity of each disk is split into two components: one parallel to the collision line, the other orthogonal to the collision line.
    d. Since it is assumed that the disks have equal masses, the velocity components parallel to the collision line are swapped.
    e. The disk positions are not changed.
8. The time step is 0.001. In order to compute the disk positions for the next frame, 100 steps should be applied. The disk positions for each frame should be printed to the file frame_NNNN.txt, where NNNN is the 4-digit frame index going from 0 to 499. The format of this file is as follows:
    x_1 y_1 R
    x_2 y_2 R
    …
    x_N y_N R
    where (x_i, y_i) is the center of the i-th disk. 500 frames should be generated in total. Then, the motion of the disks can be visualized using the gnuplot command `load "animate_disk.plt"`, which creates a gif-file.
9. The collision detection should be optimized using the sweep and prune approach:
    a. At each step, the disks are sorted by the x-coordinate of the disk center using the insertion sort.
    b. The collision of the disks i and j is checked only if $|x_i - x_j| < 2R$. Since the disk array is sorted, all such disks j lie near the disk i is the array; so that the traversal of the whole array is not required.

10. In order to measure the total time to process disk-disk collisions, a class for time measurement should be implemented using `std::chrono::steady_clock`. (See `std::chrono::duration` and `std::chrono::duration_cast` as well.)
11. For 400 disks, the total time to process disk-disk collisions is about 0.5 s on a PC with CPU Intel Core i7 920 2.67 GHz. In order to be accepted, a solution should spend at most 1 s in the processing of disk-disk collisions on the same PC.

References:

1. http://en.wikipedia.org/wiki/Elastic_collision
2. http://en.wikipedia.org/wiki/Sweep_and_prune
3. http://en.wikipedia.org/wiki/Insertion_sort