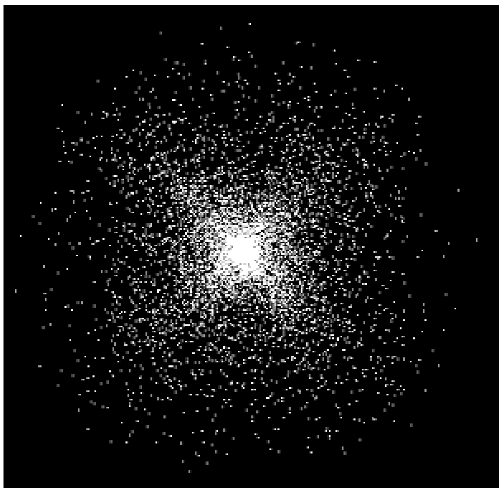
GPU Programming
TP4 – Nbody

An N-body simulation numerically approximates the evolution of a system of bodies in which each body interacts with every other body.

A typical example is an astrophysical simulation in which each body represents a galaxy or an individual star. The bodies attract each other through the gravitational force.

The all-pairs approach to N-body simulation is a brute-force technique that evaluates all pair-wise interactions among the N bodies. It is a relatively simple method, but it is rarely applied because of its O(N^2) computational complexity.



At each simulation step, the body i is accelerated by $acc_i = F_i / m_i$. The force F_i acting on body i, due to gravitational interactions with all other N-1 bodies, is obtained by the sum:

$$\mathbf{F}_i \approx Gm_i \cdot \sum_{1\le j\le N} \frac{m_j \mathbf{r}_{ij}}{\left(\left\|\mathbf{r}_{ij}\right\|^2 + \varepsilon^2\right)^{3/2}}.$$

where m_i and m_j are the masses of bodies i and j, respectively, $r_{ij} = x_j - x_i$ is the vector from body i to body j, and G is the gravitational constant. ε is a softening factor in order to avoid singularities during the simulation when two bodies get too close to each other.

This is the pseudo-code for an all-pairs N-body simulation:

```
// pos[N] = current position of the bodies
// vel[N] = current velocity of the bodies
// mass[N] = mass of the bodies

initialize pos, vel, mass
while (simulating)
      for (i=0;i<N;i++) {
            acc = 0
            for (j=0;j<N;j++) {
                  r = pos[j]-pos[i]
                  d = ||r||² + EPS²;
                  acc += G*r*mass[j]/sqrt(d*d*d);
            }
            new_pos[i] = pos[i] + vel[i];
            new_vel[i] = vel[i] + acc;
      }
      pos = new_pos
      vel = new_vel
end while
```

Execute the provided base code.

- Random bodies are created: position, velocity and mass arrays are initialized.
- No acceleration: the bodies follow a constant velocity.
- The arrow keys up/down allow increasing/decreasing the number of bodies
- GPU mode = CPU mode

Exercices (1 point per question)

1) Write a CPU version of the N-body algorithm.
2) GPU version 1. Write a kernel which receives body masses, current positions and current velocities, and which computes new pos/vel. Fetch the new data back to the host.
3) GPU version 2. Stop copying current pos/vel to the device every time you run the kernel. Copy the initial values once, then compute on the device from pos1/vel1 to pos2/vel2, and next time from pos2/vel2 back to pos1/vel1 (“double buffer”). Fetch the proper data back to the host.
4) GPU version 3: use shared memory for faster memory access. In each thread block, preload the data of NBTHREADS bodies into shared memory and start summing up partial accelerations. Then preload the next NBTHREADS bodies, continue summing up the partial accelerations, etc., until all bodies are processed.