The Wind Trap

Story

Since the Race Trap did not work Wile need a new approach. His reasoning is that if the Roadrunner is so fast, then it must be very light, and if it is so light is can easily be caught by the wind. All that is needed is something to help the wind blow the Roadrunner into a net - and it is dinnertime!

Wile has bought a huge funnel from ACME. The idea is that Wile can use the natural occurring wind as a source, and by concentrating this wind the Roadrunner can be blown into the net.

The problem is that the concentrated wind need to hit the Roadrunner at a distance of at least 5 meters and the concentrated wind must be at least 90% of the wind that was collected, otherwise there is not enough power to catch the Roadrunner. To tell if the trap works we need to simulate it in a wind-tunnel, fortunately Wile lives in a two dimensional world, so the wind-tunnel may be simulated via a Lattice Gas Automaton.

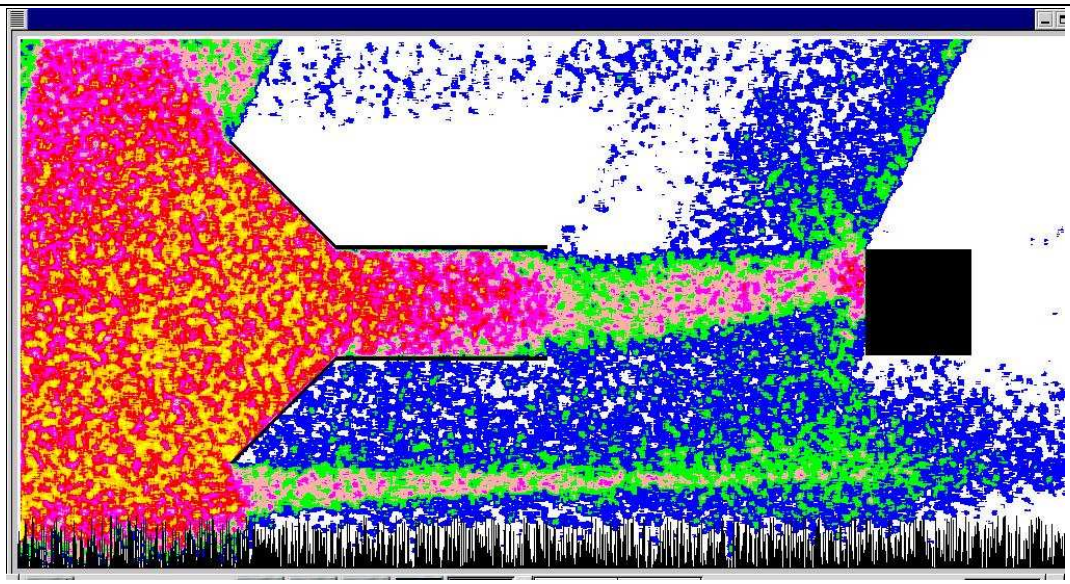

Figure 1 The result of the wind-trap

Considerations on the parallel version

The Lattice Gas Automaton is a complete application that models airflow in a two-dimensional wind tunnel. The flow is modeled as particles rather than using gas dynamics. This model is both simple and efficient, and is to some degree easy to parallelize. The idea is to model the two-dimensional space as a set of large matrixes of bits, where 1 indicates the presence and 0 the absence of a gas particle. Eight matrixes are used, one representing the object in the wind tunnel, one representing gas particles that have no motion, and six matrixes representing particles moving in six different directions (right, right-up, right-down, left, left-up, left-down).

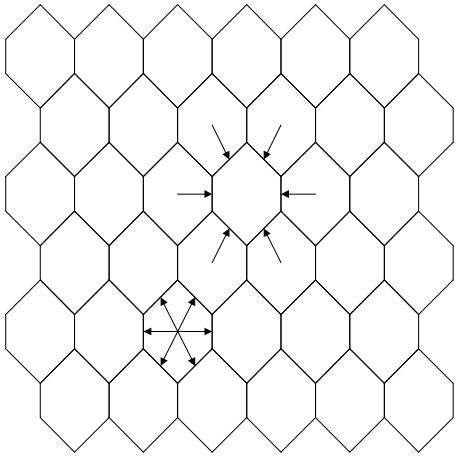

Figure 2 Layout of the Lattice Gas Automaton

The simulation runs for a predetermined number of time steps. Each such step consists of three distinct phases. First all points in the space are checked to see if more than one particle occupies this space, indicating a collision of two or more particles. The object that is being tested is modeled as a set of immovable particles. A set of collision-rules determines the implications of collisions on the particles involved. This first step accounts for most of the time spent on each step. Following the collision-test, all gas particles are moved in the matrixes. Finally new particles are injected into the system.

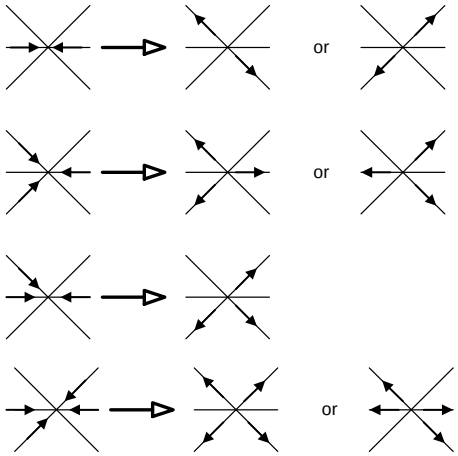

Figure 3 Particle collision rules in the Lattice gas automaton [Orme 95]

The detection of collisions depends on the same point in all eight matrixes so the same part of the different matrixes should be kept at the same worker-process. To move particles it is necessary to access different points within the same matrix. The space can be partitioned along the left-right axis, so that communication is needed in four dimensions only, e.g. particles moving right-up, right-down, left-up, and left-down but not straight left and right. Pseudo code for the Gas Automaton is found in Example 1.

```
for(i=0; i<iterations;i++) {
  Do collision check
  Calculate particle movement
}
```

Example 1. Pseudo code for the Lattice gas automaton.

Writing a parallel version of the Lattice gas automaton is primarily a question of dividing the matrices in an efficient manner and ensuring latency-hiding by overlapping communication and calculation, the collision test code don't need to be parallelized.

Programming Task

The solution should be implemented using RMI, and based on the sequential version below. The code should be run on from one through 32 nodes.

The report should identify the various choices that have been made as well as individual techniques that have been applied to improve performance. And the impact of each should be documented. In addition the scalability of your implementation should be discussed and the achieved performance curve should be discussed.

Real World Relevance

Particle simulation

- Directly usable
- Discrete event simulations
- Particle base systems