UNIVERSITY OF PORTSMOUTH

Cellular Automata and Fluid Dynamics

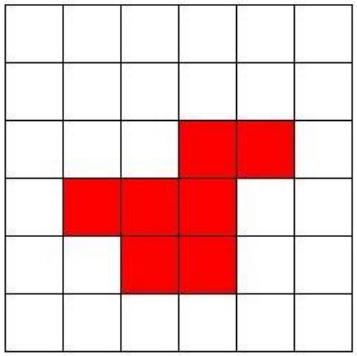
HAMIDREZA KHALEGHZADEH

Cellular Automata

You will probably have encountered cellular automata in your earlier studies of theoretical computer science.

In general the state of a cellular automaton is defined on some *regular grid* in one or more dimensions. At each site or cell it will have some more or less simple (finite) state – e.g. one or more Boolean values, perhaps mixed with some bounded integer values.

From one generation to the next, update of state at each site is a function that depends on the cell's old value and the old values of its neighbouring sites.



Cellular automata for science

Aside from their theoretical importance (see for example the early work of von Neumann and others on *self-replicating "machines"*), cellular automata have various applications as models for physical systems.

This week we will talk about use of cellular automata in *Computational Fluid Dynamics*. This material also serves as an entry point into discussion of *Lattice Boltzmann Models* later in the module.

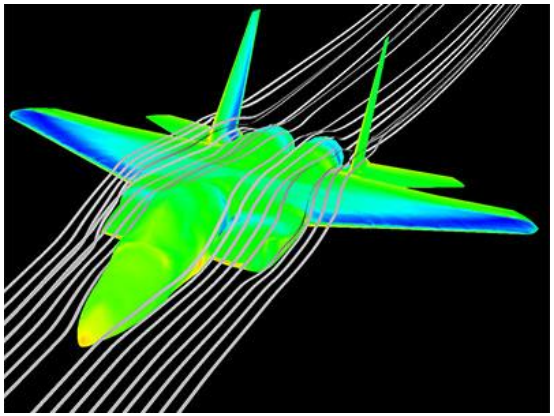
Next week we will see an example from *cardiac simulation* in biological sciences.

Fluid Dynamics

Fluid dynamics is an important area in science and engineering. It is the study of the motion of fluids.

Subdisciplines are *aerodynamics* and *hydrodynamics* for gases and liquids, respectively. They are governed by rather similar equations – the main difference being that liquids are not readily compressible (while gases are).

Applications are manifold and in many cases obvious – e.g. aerodynamics is vital for understanding behaviour of aircraft and other objects moving through air, weather prediction, combustion simulation, and so on.



Navier-Stokes Equation

I will display a version of this equation once, without explanation, and leave it at that:

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}\right) = -\boldsymbol{\nabla}p + \mu\boldsymbol{\nabla}^2\boldsymbol{u} + \frac{1}{3}\mu\boldsymbol{\nabla}(\boldsymbol{\nabla}\cdot\boldsymbol{u}) + \rho\boldsymbol{g}$$

Here:

- ρ is the density of fluid as a function of position
- p is pressure of fluid as a function of position
- $\boldsymbol{u}$ is velocity of fluid (flow) as a function of position
- ∇ is the divergence (partial derivatives)

and μ is "bulk viscosity" and $\boldsymbol{g}$ is acceleration due to gravity (I have neglected "volume viscosity", for simplicity).

Solving Navier-Stokes

A popular and direct approach to Computational Fluid Dynamics is to discretize the partial differential equations directly – e.g. by finite differences for the derivatives, or by *finite elements* – and solve the resulting finite system of equations on a computer.

Unfortunately the mathematical complexities of these approaches seemingly take them outside the scope of this module.

An alternative solution is to use *lattice-based approaches* to approximate the Navier-Stroke equations.

Two of interest in this module are *Lattice Gas Models* and *Lattice Boltzmann Models*.

Lattice Gas Models

Lattice Gas Model (LGM) is used to simulate fluid flows.

A *Lattice Gas Model* in general is a *cellular automaton* with finite state at each site (although in some cases it may have a pseudorandom update rule).

The main interest in Lattice Gas Models today is that they were the historical precursor to the *Lattice Boltzmann Model (LBM)* approach, to which we will return in a later week.

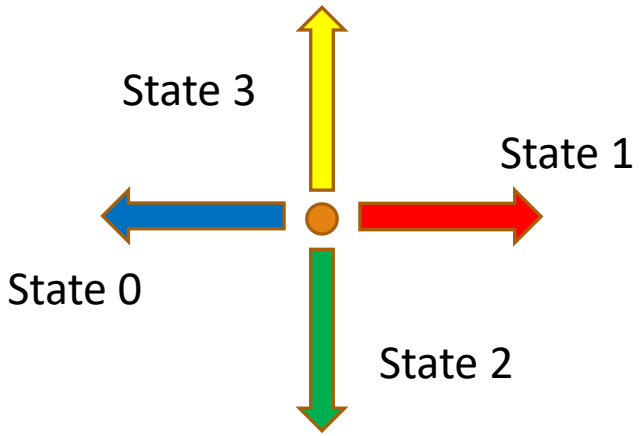
Lattice Gas Models remain pedagogically useful because they are significantly easier to understand than LBM, while sharing several of their characteristics.

The HPP Model

The simplest interesting Lattice Gas Model is the HPP model introduced by Hardy, Pomeau and de Pazzis (1973-76).

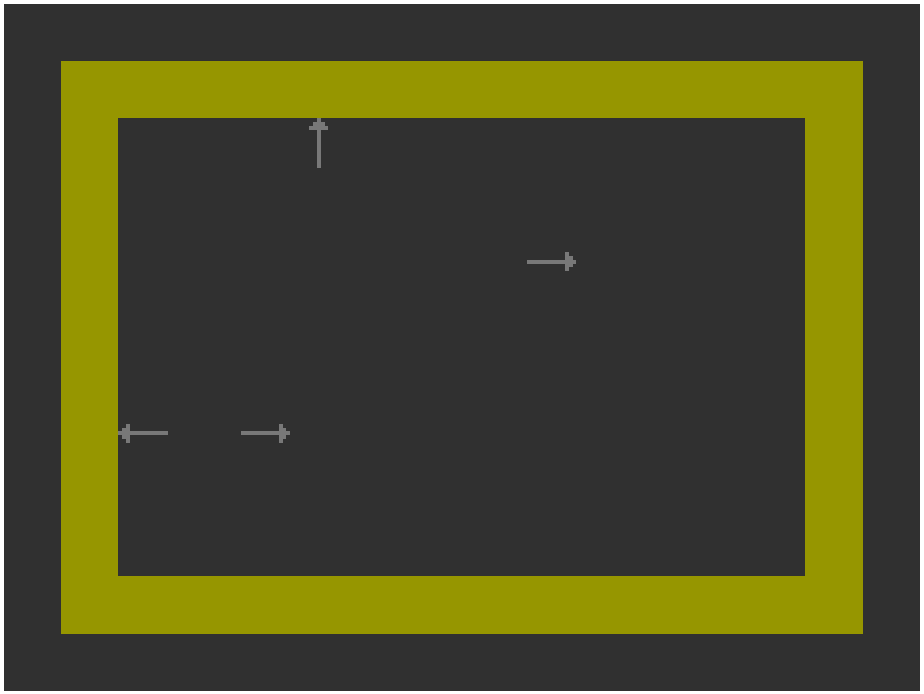
Based on a two dimensional square grid. Conceptually, grid is inhabited or populated by a set of particles with different velocities. Particles can interact to change velocity, but only four velocity states are allowed.

All four velocities have same magnitude, and four directions are: positive along x axis, negative along x axis, and similarly for y axis.



HPP - Demonstration

A small scale demonstration of the square lattice HPP model

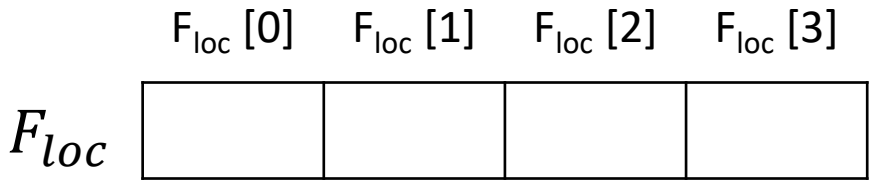


Cell state in HPP model

At any given cell at any given time, there is *at most* one particle in any of the four possible states (this is an arbitrary constraint that is relaxed in the later Lattice Boltzmann models).

So, any, local cell state can therefore be modelled by four Boolean variables:

- F_{loc} [0] = true if there is a particle here in velocity state 0, otherwise false
- F_{loc} [1] = true if there is a particle here in velocity state 1, otherwise false
- F_{loc} [2] = true if there is a particle here in velocity state 2, otherwise false
- F_{loc} [3] = true if there is a particle here in velocity state 3, otherwise false



“Macroscopic” quantities

To eventually relate the lattice gas model back to a fluid it is supposed to simulate, we will need some definition of net density and net flow or velocity of the fluid.

In the HHP model, if we treat Booleans, F_{loc}, as having values 0 or 1, then at a particular site we can say:

Local density (total particles) =

$$\sum_{i=0}^{3} F_{loc}[i]$$

Local momentum density =

$F_{loc}[1] - F_{loc}[0]$ (x-component), $F_{loc}[3] - F_{loc}[2]$ (y-component)

Update rule

One can write an update rule as a function of the local state and neighbouring states, as for any cellular automaton. But it is conventional to break the update down into two steps – both steps must be completed in order to compute the next generation of cell states.

Two steps are:

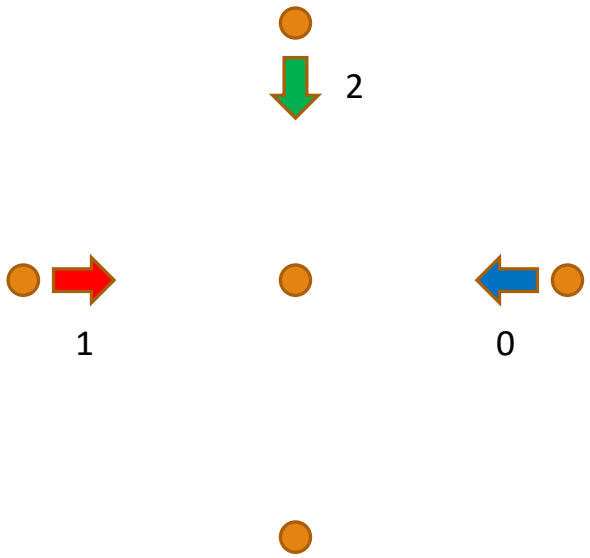
1. *Collision step*
2. *Streaming step*

Step 2 goes by various other names, like *transport step* or *propagation step*. We call it the streaming step for consistency with common use in Lattice Boltzmann Models.

Streaming step (before)

Since this is rather simple, describe it first (though usually put second in main loop of program – only really matters that two steps alternate).

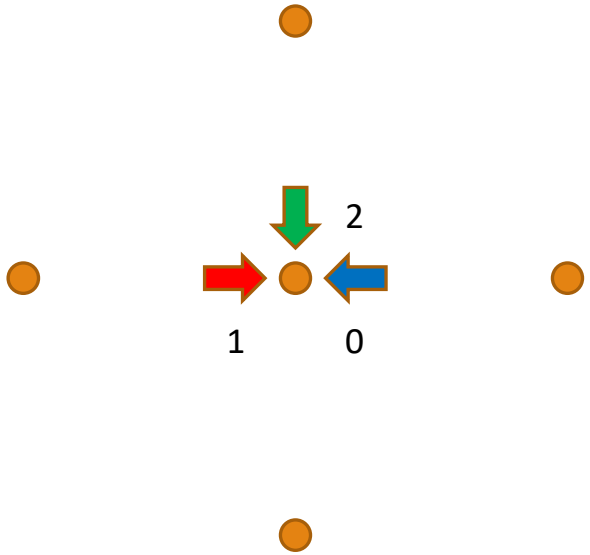
If there is a particle with velocity *in a particular direction* at a particular site, it moves one step to the *next site in that direction*:



Streaming step (after)

Since this is rather simple, describe it first (though usually put it second in main loop of program – only really matters that two steps alternate).

If there is a particle with velocity *in a particular direction* at a particular site, it moves one step to the *next site in that direction*:



Streaming, “programatically”

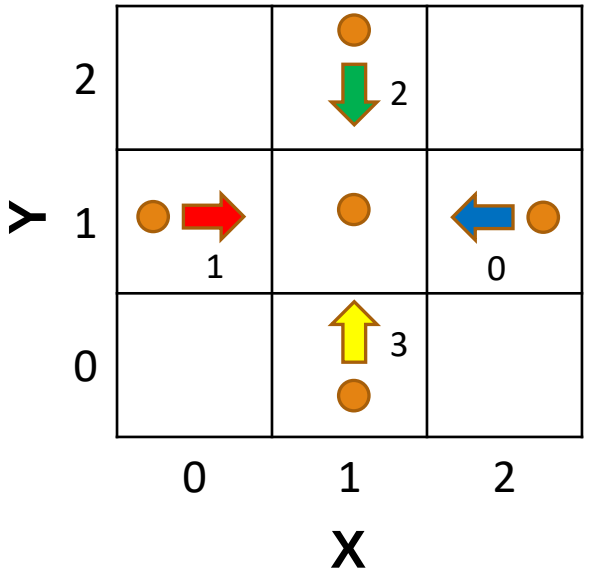
If we call the arrays of local Booleans *before* streaming $F_{x,y}^{out}$ (outgoing) and those after streaming $F_{x,y}^{in}$ (incoming), then the streaming step looks like:

$F_{x,y}^{in}$ [0] = $F_{x+1,y}^{out}$[0] // moving in –ve x direction

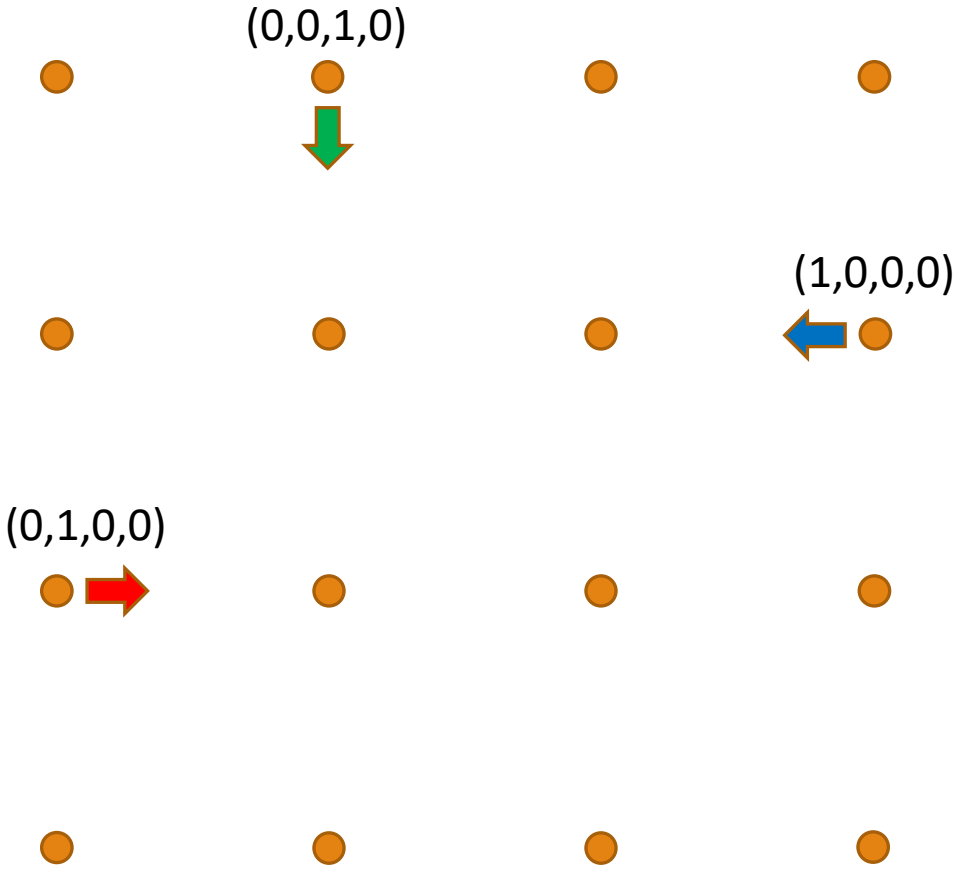
$F_{x,y}^{in}$ [1] = $F_{x-1,y}^{out}$[1] // moving in +ve x direction

$F_{x,y}^{in}$ [2] = $F_{x,y+1}^{out}$[2] // moving in –ve y direction

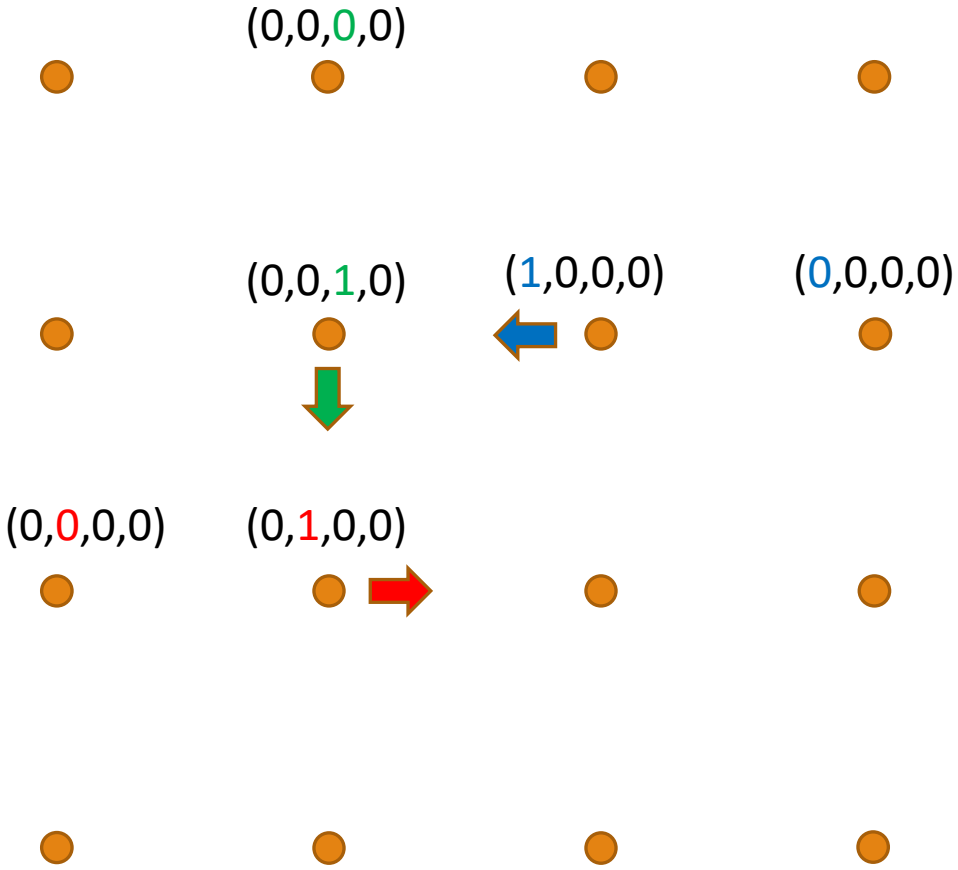
$F_{x,y}^{in}$ [3] = $F_{x,y-1}^{out}$[3] // moving in +ve y direction



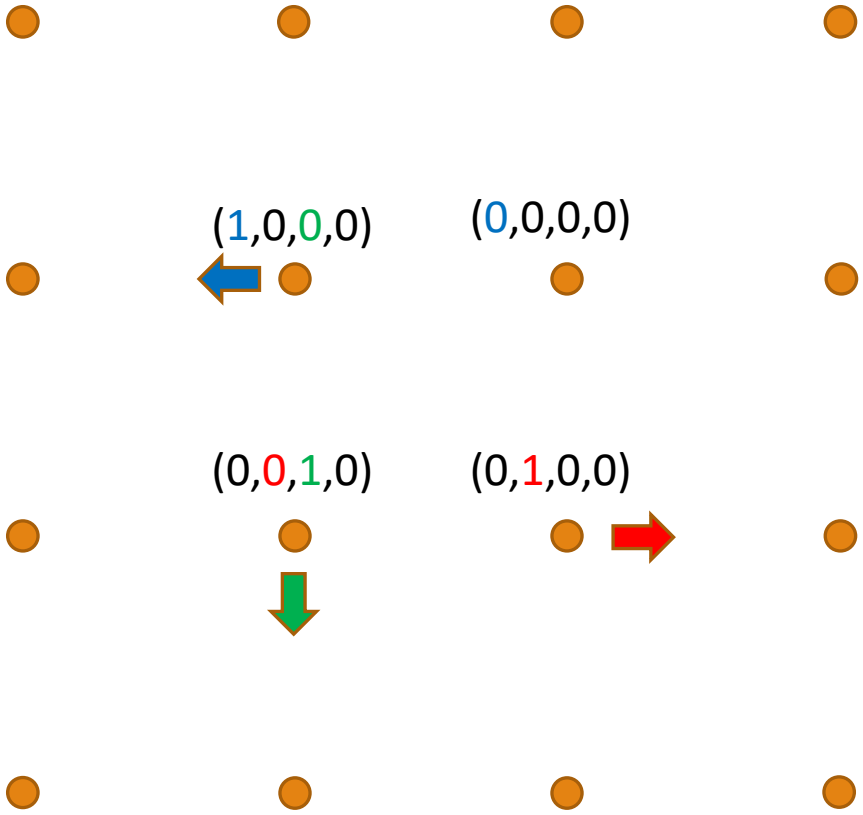
Some particles streaming



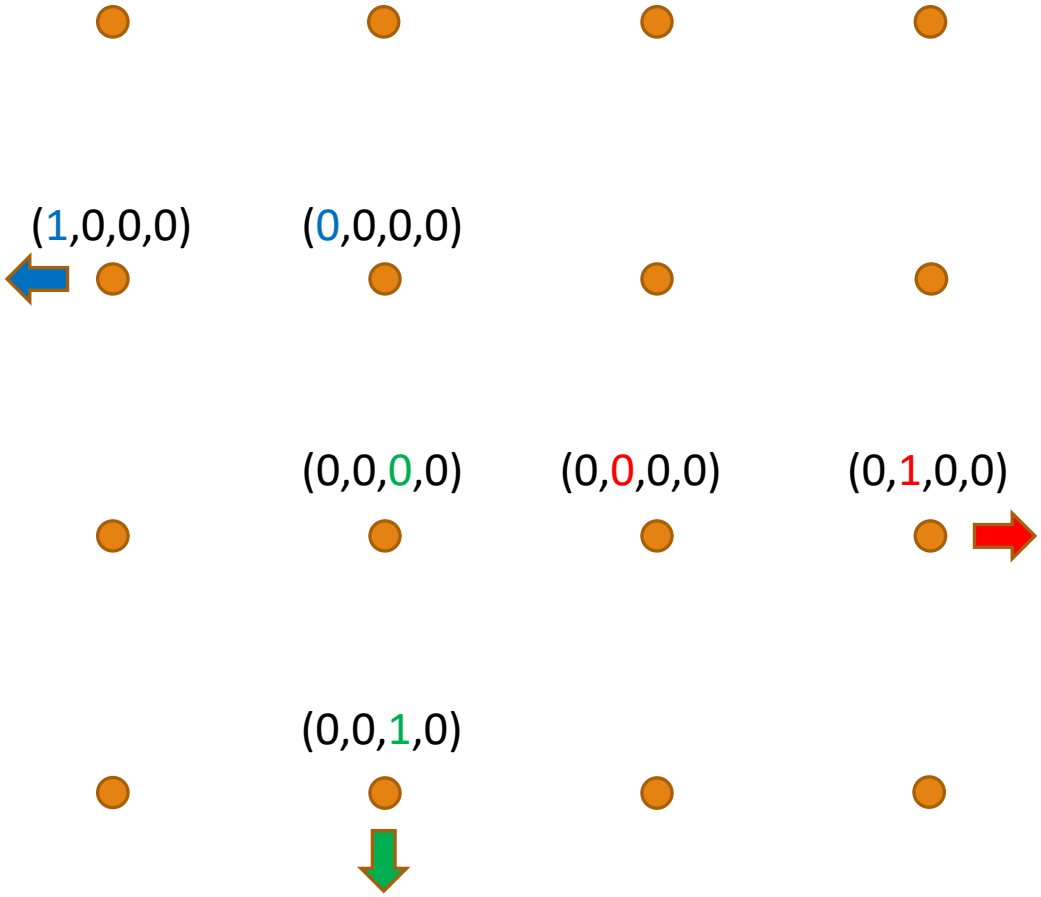
Some particles streaming



Some particles streaming



Some particles streaming



Collision

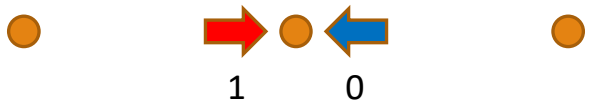
After a streaming step, the incoming particles at a particular site can interact through a *collision*.

In words, the rule for collision in the original HPP model is just:

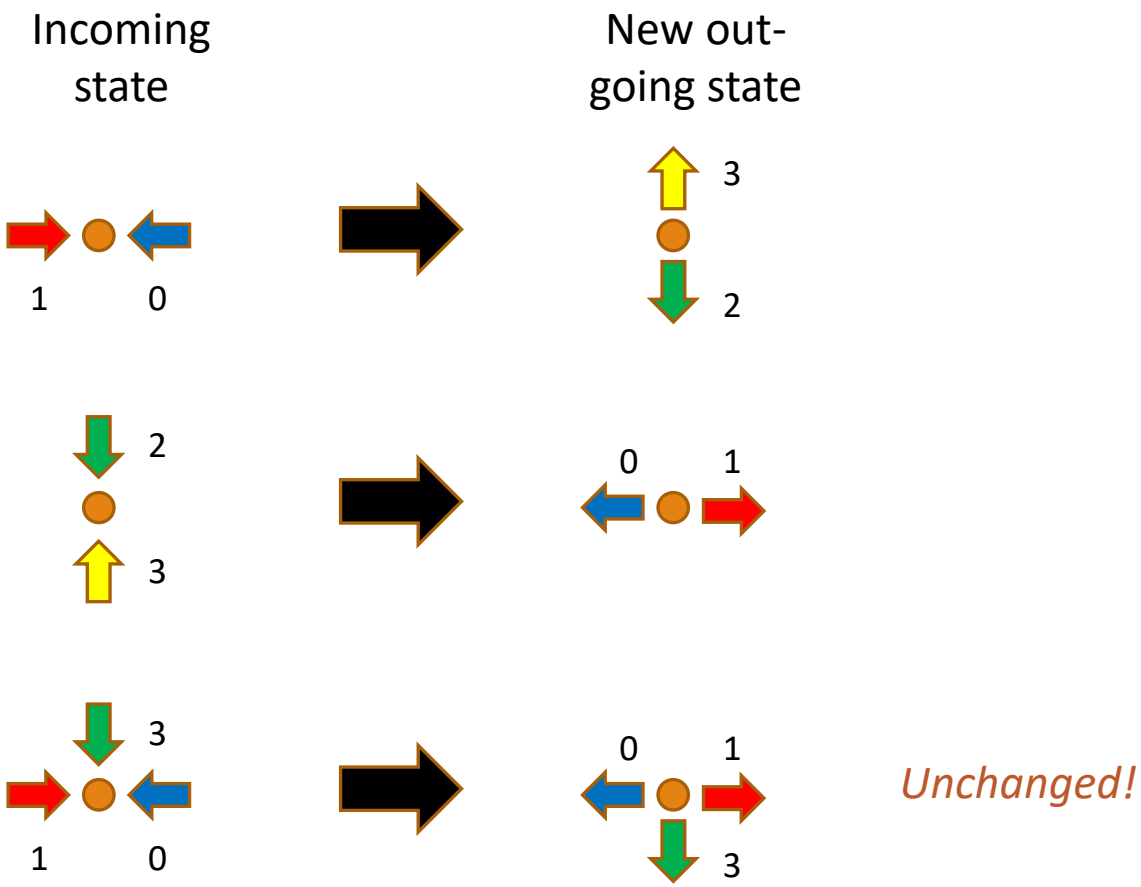
Two incoming particles involved in a head-on collision are deflected perpendicularly

- (but only if there are no particles occupied at that cell).

Deflection does not change the velocity magnitude.



Collision in pictures



Programmatically

In some kind of non-Java pseudocode:

if (only states 0 and 1 populated in $F_{x,y}^{in}$)

$F_{x,y}^{out}$ = [false, false, true, true] // states 2 and 3 populated

else if (only states 2 and 3 populated in $F_{x,y}^{in}$)

$F_{x,y}^{out}$ = [true, true, false, false] // states 0 and 1 populated

else

$F_{x,y}^{out} = F_{x,y}^{in}$

Note that in 14 out of 16 *possible* cases, collision produces no change!

(In the other cases no there is no rearrangement that would conserve momentum and particle number, and not put more than one particle in any local velocity state).

Limitations of HPP model

Although reportedly one can reproduce *some* features of realistic fluid dynamics with the HPP, the results don't closely resemble those of a continuous fluid even for very large lattices.

Crucially, the large scale behaviour of HPP models are *not* described by the Navier Stokes equation.

Technically, the square grid simply doesn't have enough symmetry to yield to an *isotropic* continuum behaviour.

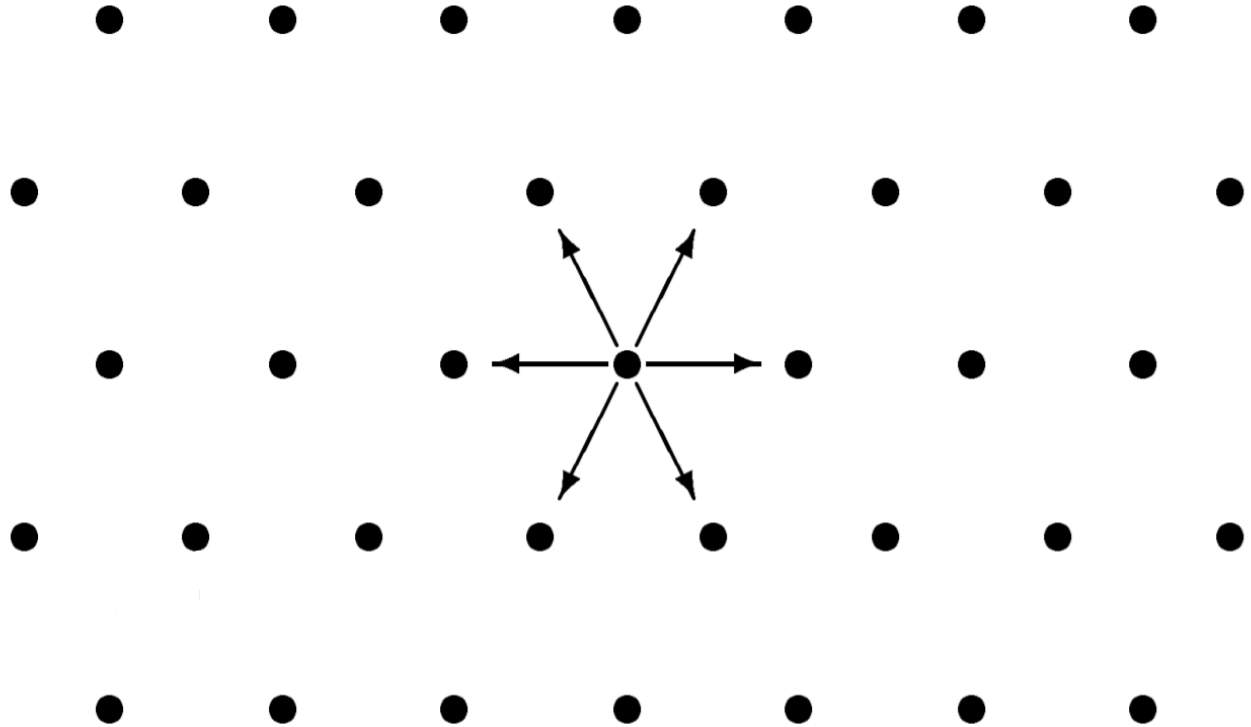
The FHP Model

A big step forward on the HPP model was introduced a decade later (1986) by Frisch, Hasslacher and Pomeau.

The *FHP* model works very similarly to the HPP, but it is defined on a triangular or hexagonal lattice instead of a square lattice.

Remarkably it was shown that the large scale behaviour of this new lattice gas model *is* described by Navier Stokes equations, and thus it can be used to simulate fluids (at least in two dimensions).

Lattice and velocity states†

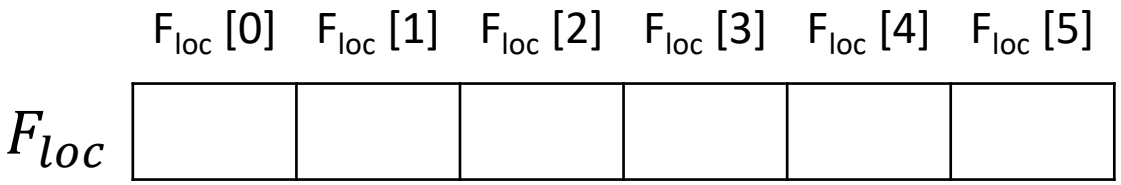


†Dieter A. Wolf-Gladrow, *Lattice-Gas Cellular Automata and Lattice Boltzmann Models - An Introduction*, Fig 3.2.1

Streaming in FHP

In the simplest versions of the FHP model there are six states instead of four in the HPP model. Thus local state is represented by 6 Boolean variables describing which states are populated locally.

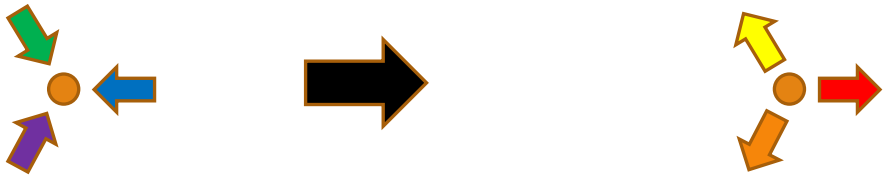
Apart from that, the streaming step works in exactly the same way, with a particle moving along one of the outgoing links to its neighbour, according to the direction of its velocity.



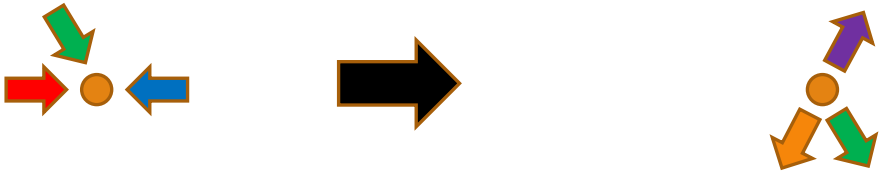
Example collision rules

Head on collision

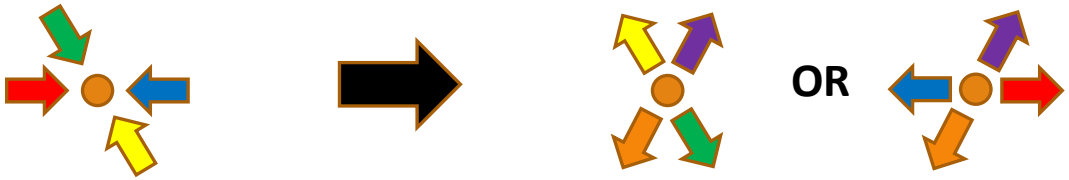
Three way collision



Head on with spectator



Double head on collision



Collision in FHP

This is a little more complex than in HPP, partly because there are now 64 possible cases in total to consider.

Perhaps most novel feature, illustrated on previous slide, is that head on collisions can lead to two possible outcomes. In practice the outcomes are chosen with 50/50 probability – the "cellular automaton" is no longer strictly deterministic.

In simplest FHP model, rotation and reflection symmetry give 3 variants of first collision, 2 of second collision, 12 of third, and 3 of fourth. The other 44 incoming states are unchanged after the collision step (but compare 20/64 states leading to collision in FHP with 2/16 for earlier HPP model).

LGM for CFD

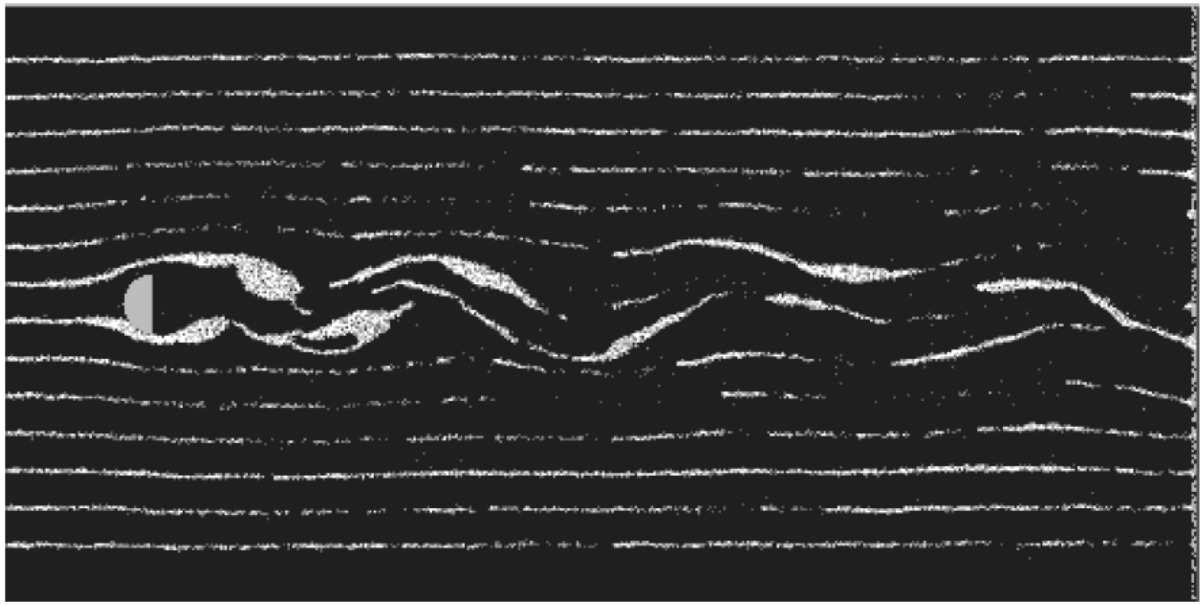


Fig. 1.13. Flow past a half-cylinder using a six-direction lattice gas on a triangular lattice. We simulate "smoke" streamers to visualize the flow.

From Norman Margolus, *Crystalline Computation*, in *Feynman and Computation*, 1999, edited by Tony Hey. Lattice is 2000 by 1000.