7. Traffic Simulation via Cellular Automata

From Chapter 12 (mainly) of "Computer Simulations with Mathematica", Gaylord & Wellin, Springer 1995 [DCU library ref 510.2855GAY]

■ 7.0 Overview of Cellular Automata (p91 etc of reference)

■ 7.0.1 What is a Cellular Automaton?

A **Cellular Automaton** (CA) consists of a discrete system of **lattice sites** having various **initial values**.

These sites evolve in discrete **time steps** as each site assumes a new value based on

(the values of some local neighbourhood of sites) + (a finite number of previous time steps).

■ 7.0.2 Kinds of CA lattices

In one dimension, a cellular automaton is a linear list of numbers or symbols.
In two dimensions, various lattices are possible (e.g. triangular, hexagonal, or rectangular).
In higher dimensions, the approach is similar but lattices are obviously more complex.

■ 7.0.3 CA Neighbourhoods

The neighbourhood of a lattice site consists of the site and its nearest neighbour sites.

For the rectangular lattice, two kinds of neighbourhood are commonly used,
- Von Neumann (site + 4 nearest neighbours (N, E, S, W))
- Moore (site + 8 nearest neighbours (N, NE, E, SE, S, SW, W, NW)).

■ 7.0.4 Treatment of sites at the edge of a lattice (focus on rectangular lattices)

Various boundary conditions are possible. For example,

Absorbing boundaries: Nearest neighbour to left of a site on the left border of the lattice is zero.
Similarly, for sites on right, top and bottom borders.

Periodic boundaries: Nearest neighbour to left of a site on the left border of the lattice equals site on same row on right border.
Similarly, for sites on right, top and bottom borders.

■ 7.1 Introduction to traffic modeling using cellular automata

Our model is that of particles located on a lattice such that ***no lattice site is occupied by more than one particle at a time***.

The particles have different probabilities of moving left or right so that, overall, there is a **net movement** of particles in a direction.

We focus on modeling one-way traffic with one-dimensional cellular automata.

Our purpose is (simply) to examine the effect of stopping on the flow of traffic as a function of the density of cars on the road.

A somewhat less simple problem would be to include the effect of passing (overtaking).

■ 7.2 One-Lane Road with Car Stopping

■ 7.2.1 Overview

● Traffic takes place on a one-dimensional lattice of length **s**.

● Periodic boundary conditions are assumed which means that the system has a ***ring*** geometry.

● Values of sites are
either
  car speeds - integers ranging from 0 to **vmax** (the speed limit!)
or
  symbol **e**, representing an empty site.

● The system evolves over a specified number of time steps by updating each lattice site, based on its value and on the values of neighbouring sites to its right (that is, in front).

■ 7.2.2 The One-Lane Algorithm

Step 1:
The initial road configuration consists of cars whose speeds and locations are both randomly distributed.
This is done by first placing cars on the road and then determining their speeds.
As well as **s** and **vmax**, a needed input parameter is the probability that a location on the road is occupied by a car (call it **p**). (We note that in the limit of large **s**, the density of cars on the road will approach the value of **p**).

The movement of cars along the road is carried out by simultaneously updating all of the lattice sites according to Steps 2 (***adjusting car speed***) and 3 (***moving the car***). Doing Step 2 before Step 3 ensures that no car is crashed into from behind! Steps 2 and 3 will be carried out for a specified number (**t**) of times.

Step 2: (a model for speed sdjustment)
Let **d** = distance from car of interest to the car ahead, let **vel** = velocity of the car of interest.
If **d** $\leqslant$ **vel** then slow down to **newvel = d - 1**, else speed up to **newvel = vel + 1** (but not beyond speed limit **vmax**)

Step 3: (a model for car motion)
A car moves to the right, a distance equal to the car's velocity, **newvel**.
[*Note: In general, the units of velocity are distance/time; here, we are taking the duration of each time step to be one time unit which allows us, in effect, to compare distance and velocity directly*].

■ 7.2.3 Mathematica implementation of the algorithm (don't worry about code details)

```
OneLane[s_, p_, vmax_, t_] :=
 Module[{emptyRoad, road, carDensity, followTheLeader},
  emptyRoad = Table[e, {s}];
  road = Table[Floor[p + Random[]], {s}] *
     Table[Random[Integer, {1, vmax + 1}], {s}] /.
    {0 → e, x_Integer → (x - 1)};
  carDensity = N[Count[road, _Integer] / s];
  followTheLeader = Function[y,
    vels = DeleteCases[y, e];
    locs = Complement[Range[s], Flatten[Position[y, e]]];
    distances = Join[Rest[locs - RotateRight[locs]],
      {s - Last[locs] + First[locs]}];
    newVels = Map[(Min[#[[1]], #[[2]], vmax]) &,
      Transpose[{vels + 1, distances - 1}]];
    newLocs = Map[(Mod[# - 1, s] + 1) &,
      (newVels + locs)];
    Fold[ReplacePart[#1, #2[[1]], #2[[2]]] &,
     emptyRoad, Transpose[{newVels, newLocs}]]];
  NestList[followTheLeader, road, t]
 ]
```

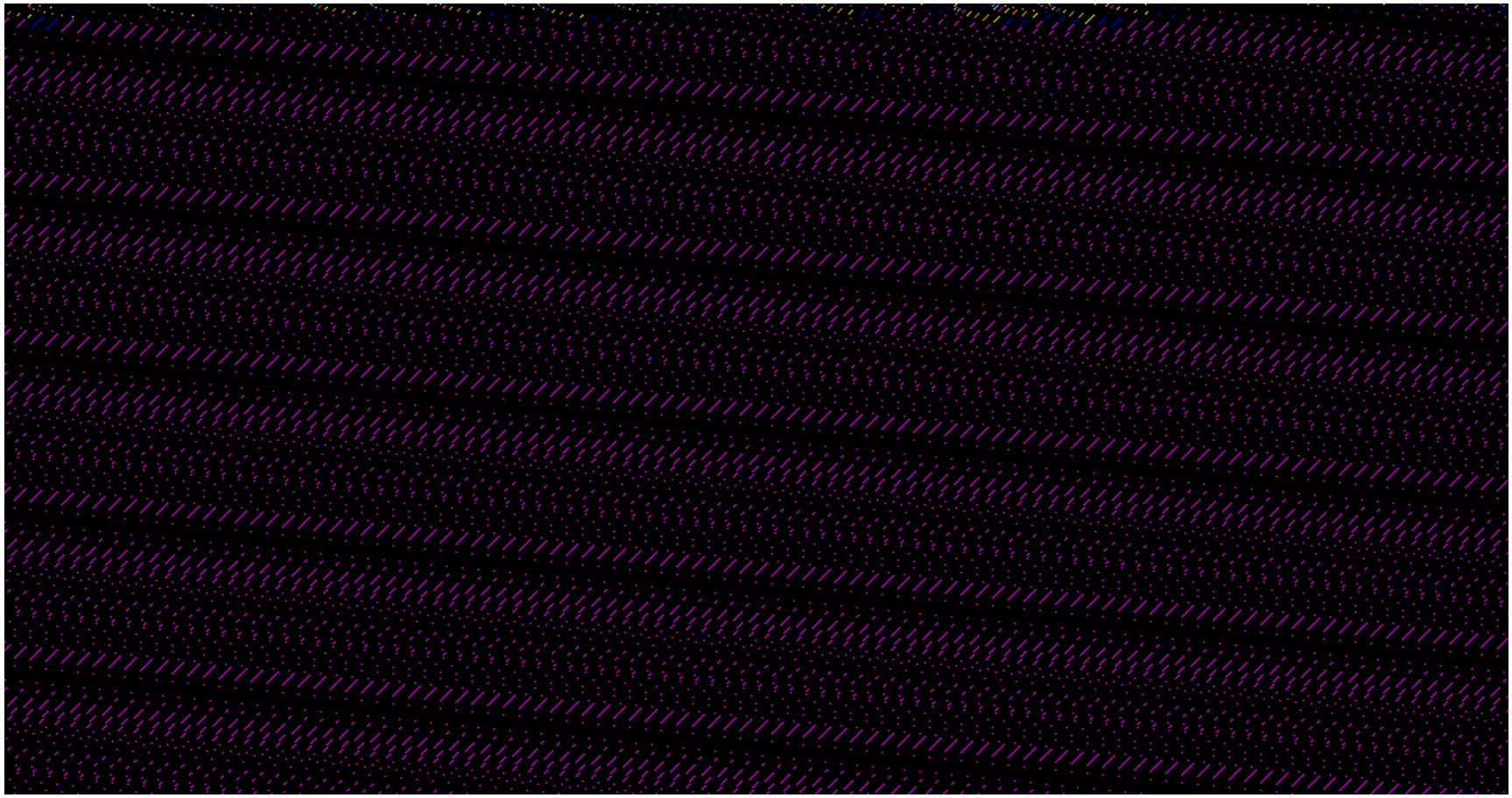
■ 7.2.4 Basis of displaying results graphically (Don't worry about the code detail)

The (empty) road will be shown in black.
Cars will be shown as hues, with different colours representing different velocities.
Successive snapshots in time will be shown to allow the evolution of the traffic velocities to be observed.

```
ShowTraffic[list_List, opts___] :=
 Module[{vmax = Max[list /. e → -1]},
  Show[Graphics[RasterArray[Reverse[list] /.
      Join[{e → Hue[0, 0, 0]},
       Thread[Range[0, vmax] →
         (Map[Hue, Table[Random[], {vmax + 1}]])]
      ]]],
   opts, AspectRatio → Automatic]]
```

■ 7.2.5 Road length 1000, Maximum velocity 10, 500 time steps, Approx car density (p) = 0.05

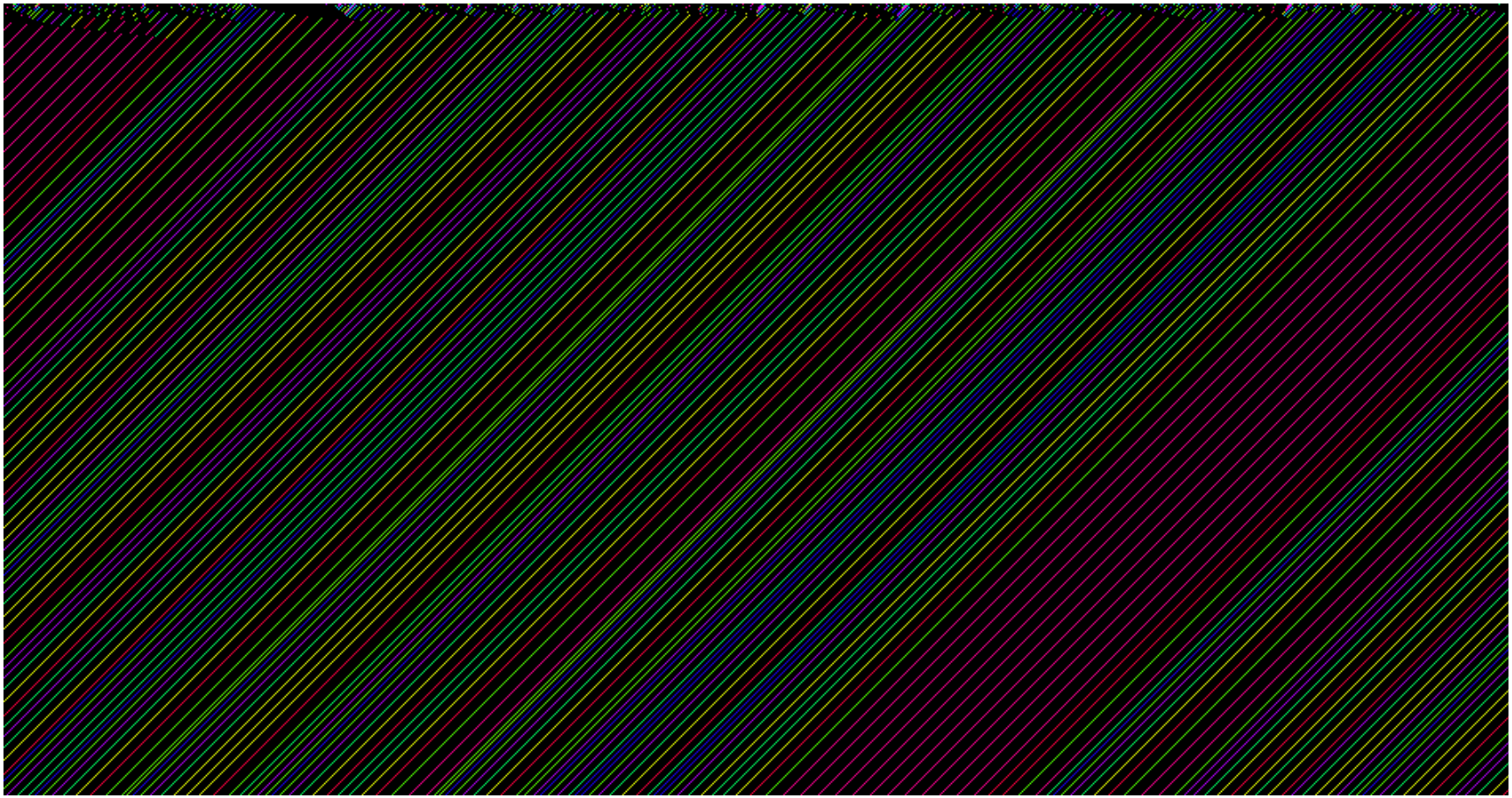
```
ShowTraffic[OneLane[1000, 0.05, 10, 500]]
```



- Graphics -

■ 7.2.6 Road length 1000, Maximum velocity 10, 500 time steps, Approx car density (p) = 0.15

```
ShowTraffic[OneLane[1000, 0.15, 10, 500]]
```



- Graphics -

■ 7.2.7 Road length 1000, Maximum velocity 10, 500 time steps, Approx car density (p) = 0.25

```
ShowTraffic[OneLane[1000, 0.25, 10, 500]]
```

- Graphics -

■ 7.2.8 Road length 1000, Maximum velocity 10, 500 time steps, Approx car density (p) = 0.35

```
ShowTraffic[OneLane[1000, 0.35, 10, 500]]
```

- Graphics -

■ 7.2.9 Road length 1000, Maximum velocity 10, 500 time steps, Approx car density (p) = 0.50

```
ShowTraffic[OneLane[1000, 0.50, 10, 500]]
```

- Graphics -

■ 7.2.10 Comments on results

Above figures (distance on horizontal towards the right, time vertically downwards) can be interpreted in terms of kinematics waves which move at different speeds and meet to produce shock waves.
The kinematic waves are created as cars adjust their speed to the cars ahead, creating waves of constant flow:
- packs of fast-moving, widely spaced cars
- caravans of slow-moving, closely-spaced cars

A shock wave is created when a fast-moving car must brake suddenly to avoid running into a slow-moving car.