CITS4403 – May 1, 2013

Computational Modelling

Discrete models – Cellular Automata

Pascal Buenzli
pascal.buenzli@uwa.edu.au
Lab G17, CSSE
The University of Western Australia

Overview

- Assignments
- CA: reminder
  - Self-replication
  - Conway's game of life
- CA definition extensions
  - probabilistic CAs
- The parity rule
  - analytical development [next week]
- CA program: graphical visualisation
  - a worked-through example using gnuplot

Assignments

Projects:

http://undergraduate.csse.uwa.edu.au/units/CITS4403

http://undergraduate.csse.uwa.edu.au/units/CITS4403/assignment2.htm

- 5 different projects
- You can choose whichever you want
- To be done in pair (2 students), but individually also possible

Format:

- Written report + source code:
- Report: pdf format, 5-10 A4 pages
  - Intro, model, results & discussion, conclusions, refs
  - Aim: "computational modelling" – task sheet gives possible thread but you are allowed to deviate *within reason* to follow your own ideas
- Code: preferably

  C/C++, Java, D, Matlab/Octave, Fortran, Mathematica, Python
- Submission: Friday 31th May on cssubmit (pref.) or email to me
- Criteria (based on report – code is only for checking purposes):

  correctness: 50%, discussion: 30%, presentation: 20%

CA and complexity

- von Neumann + Ulam (40s)
  - used before, but not formalised
    - cellular spaces, tessellation automata, homogeneous structures, cellular structures, tessellation structures, iterative arrays, lattice gases
  - extract abstract mechanisms leading to self-reproduction of biological organisms
  - universal computation property
    - CA are a nonrestrictive computation technique

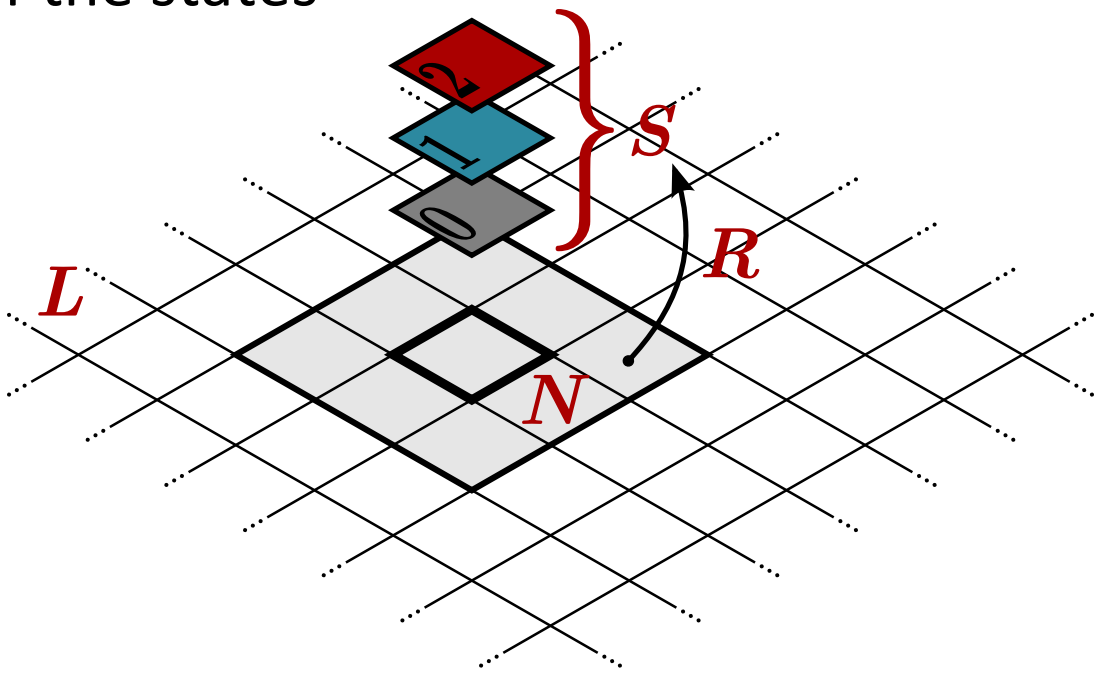
- Chaos, emergence and self-organisation
  - emergence of structures without pre-existing plan (`design') in the rules that the cells are using
  - complexity is not encoded in the rules
  - collective behaviour emerges out of the sum of many simple interacting components

`The whole is more than the sum of its parts'

CA: Formal Definition

A Cellular Automaton requires in general

1. A regular lattice *L* of ordered cells
2. A finite state space *S* for each cell;
3. A neighbourhood *N* that each cell interacts with
4. An evolution rule *R* which specifies the time evolution of the states



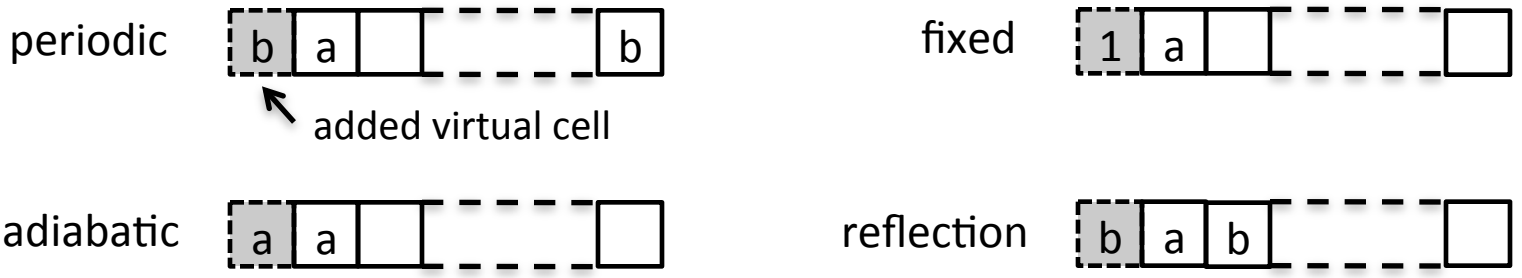
- The rule is applied simultaneously for all cell: synchronous dynamics

- The rule is identical for all sites: it is homogeneous (independent of $\boldsymbol{r}$)

- Spatial and temporal inhomogeneities can be introduced:
  - Define additional bits of the state $s(\boldsymbol{r},t)$ set to e.g. 1 for given $\boldsymbol{r}$ and t to mark particular locations and times where different rules may apply; E.g. Boundary cells

- New state at t+1: only a function of previous state at t
  - Longer memories can be introduced: dependence on states at t-1, t-2, ...
  - keep a copy of the previous states in the current state by using extra bits: E.g: $s_{t+1}=R(s_t,s_{t-1})$. Define $u_t=\{s_t,s_{t-1}\}$. Then $u_{t+1} = R(u_t)$.

- Boundary conditions
  - Computer: finite lattice -> boundaries.
  - a boundary site does not have the same neighbourhood -> different evolution rule
  - Being or not at the boundary: code at the site and choose rule appropriately!
  - Another possibility: extend the neighbourhood for the sites at the boundary (virtual cells):



Self-reproducing CAs

- Two fundamental questions (von Neumann 1949)
  - Self-reproduction. Can any automaton construct other automata that are exactly like it? Can it be made, in addition, to perform further tasks, e.g. also construct other, prescribed automata?
  - Evolution. Can the construction of automata by automata progress from simpler types to increasingly more complicated types? Can it go from "less efficient" to "more efficient" automata?

Trivial self-reproducing CAs

- Problem: there are trivial examples...
  - State space S={𝕆,1,2}
  - Neighbourhood N_2={self,N,E,S,W}
  - Evolution rule:

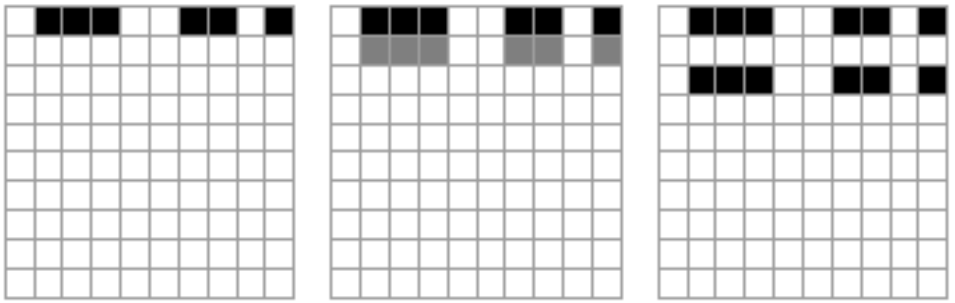
    R(𝕆,1,*, 𝕆,*)=2,

    R(𝕆,1,*,1,*)=𝕆,

    R(𝕆,2,*,*,*)=1,

    R(1,*,*,*,*)=1,

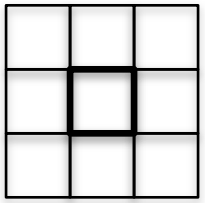
    R(2,*,*,*,*)=𝕆



- What is a good definition of trivial/nontrivial?
  - Computational universality?

Conway's `Game of Life'

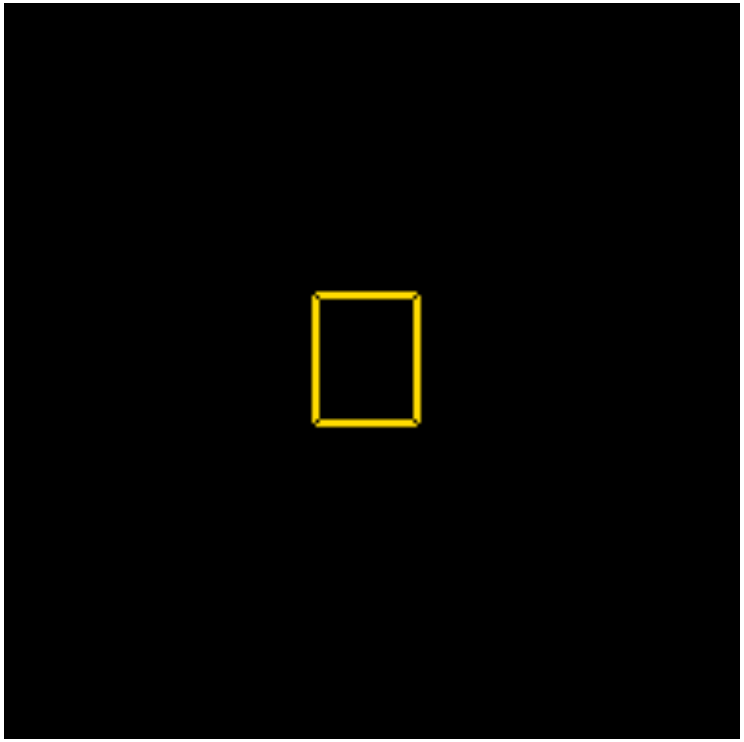
- John Conway (1970)
- Each cell is either alive (state 1) or dead (state 0)
- Lattice 2D, 8 nearest neighbours (Moore)



- Evolution rule (B3/S23)
  - A dead cell comes back to life if surrounded by (exactly) 3 live cells
  - A live cell dies if surrounded by less than 2 (isolation) or more than 3 (overcrowdness) live cells.
- Simulation example:
  - Java Applet: http://www.ibiblio.org/lifepatterns/

A simple CA: The parity rule

$$s_{t+1}(i,j) = s_t(i+1,j) \oplus s_t(i-1,j) \oplus s_t(i,j+1) \oplus s_t(i,j-1)$$



Chopard & Droz, http://theory.physics.unige.ch/cours/outils/

A simple CA: The parity rule

- Properties:
  - Initial pattern is replicated at specific times
    - times at which this happens are a power of 2
  - If system size ℓ is a power of 2: the state of each cell vanishes after $\ell/2$ iterations!
- Can we understand this? *[derivation in class: next week]*

Modelling physical systems with CA

- CA can provide a modeling environment for physical systems
- Physics: often nonlinear phenomena (PDEs)
  - very complex evolution
  - chaotic behaviour
- Similar complications occur in discrete dynamical systems
- CA models provide an alternative approach to study dynamical systems
  - microscopic simplicity: potentially easier to analyse
  - numerical studies: free of rounding approximation, exact results

CA: extensions of the definition

- Non-uniform or non-homogeneous CA:
  - cellular rules need not be identical for all cells
  - use extra bits to code which rule is chosen
- Asynchronous CA:
  - laws of Physics are `parallel'. But several processes occur randomly in time.
  - Cells are not updated simultaneously, but e.g. sequentially at each time step, or randomly.
  - Collective behaviour is lost
- Probabilistic CA:
  - important and common generalisation of CA
  - similar to non-uniform CA: randomness enters the rule through extra bits which are 1 with probability p.
  - random bit(s) generated by `data-blind' processes
  - allows to relax some artefacts of the discrete nature of CA

Assignment

CA program – visual output?

- Computational part
  - Calculate CA evolution
    - Evolving array of states
- Graphical part
  - Represent the generated data; e.g. pixmap
  - Snapshots of the lattice at various times
    - Post-processing
    - Real-time processing (animations)
- User interface?
  - Can be nice and handy (start/pause/step/reset)
  - But not strictly necessary for our purposes

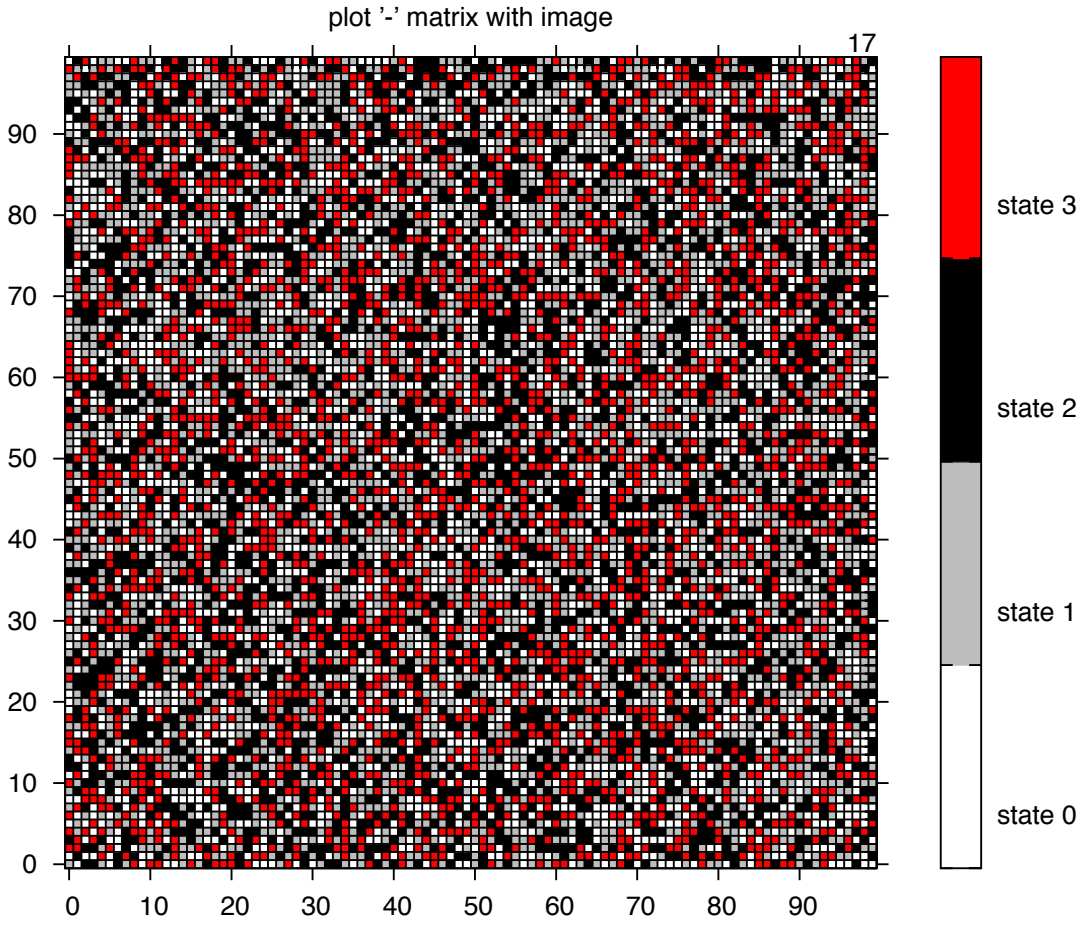
Solution 1: Graphical toolkits

- Matlab/Octave
  - Inbuilt graphical toolkits
  - Octave's: based on gnuplot
- Java
  - abstract windowing toolkit (AWT)
- C/C++, D, other programming languages?

  Several GUI libraries are freely available
  - GTK/GTK+/GTKD, QT, Tcl/Tk, WxWidgets, FLTK, …
  - Drawback: learning full-feature library…

Solution 2: Use external tools

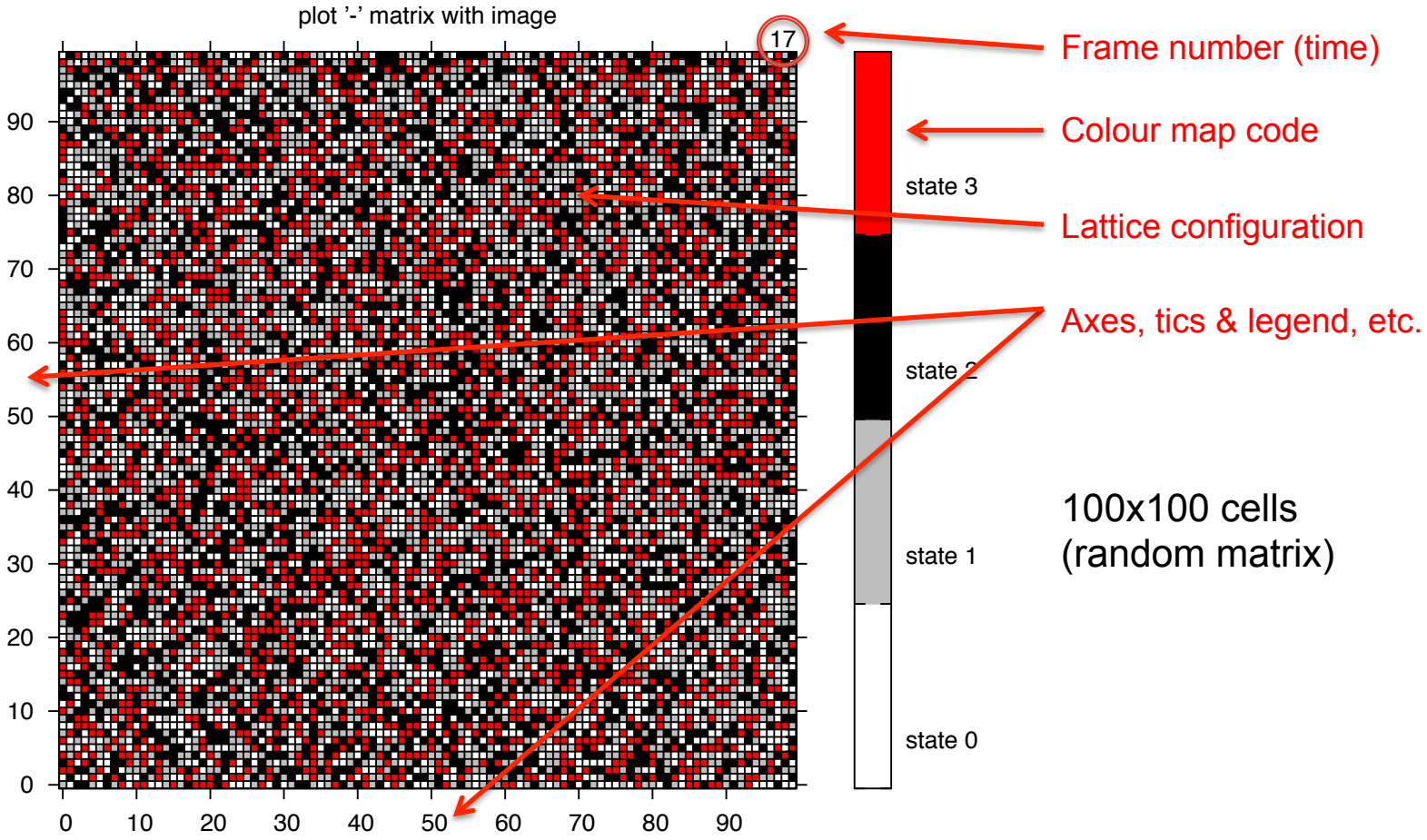
- Post-processing
- Real-time: visualise streaming/dynamic data
- A worked-through example: gnuplot



100x100 cells
(random matrix)

Solution 2: Use external tools

- Post-processing
- Real-time: visualise streaming/dynamic data
- A worked-through example: gnuplot



Gnuplot to visualise streaming data

- Useful features of gnuplot
  - Script language, batch mode/interactive mode
  - Cross-platform (Unix, Mac, Cygwin)
  - Publication-quality figures (*not by default!*)
  - Many options, plotting styles, and output formats:
    - GUI: x11, wxt, aqua, (xterm…!)
    - eps, eps+latex, epscairo, pdfcairo, cairolatex, svg… (vector figure formats)
    - png, jpeg, gif, … (rasterised figure formats)
- Visualise streaming data?
  - Redirect program output (raw or ascii) to gnuplot

```
$ program | gnuplot [-persist]
```

Gnuplot to visualise streaming data

- 2D CA output = time series of matrix of numbers
- Procedure:

1. Output gnuplot setup commands. Extract:

```
set term x11 size 800,600 # which "device" to plot onto: x11 window
set size ratio 1 # aspect ratio of the axes
set palette model RGB defined (0 'white', 0.99 'white', 1 'gray', 1.99
'gray', 2 'black', 2.99 'black', 3 'red', 3.99 'red') # cell state colorcode
set cbrange [0:3.99] # range of colormap box (otherwise inferred from data)
set cbtics ("state 0" 0, "state 1" 1, "state 2" 2, "state 3" 3) offset 0,2
set title "My Cellular Automaton"
set key at graph 1,1 bottom Right reverse # used to show current 'time'

# Grid to delimit the cells
set xtics out -200,10,200 #only shown when in range
set ytics out -200,10,200
set x2tics -200.5,1,200.5 format "" # invisible, used to offset grid by 0.5
set y2tics -200.5,1,200.5 format "" # invisible, used to offset grid by 0.5
set grid noxtics noytics x2tics y2tics front linetype -1
```

2. At each iteration step,

a) Output the gnuplot plotting cmd: `plot '-' … with image`

b) Output the matrix of numbers and flush stdout

ascii format (slow, but readable)

```
plot '-' matrix using (0+1*$1):(-1-1*$2):3 title '…' with image
0    0    0    0    0    …
1    0    2    3    0    …
3    0    1    2    3    …
…
e
e
```

raw format (fast, but binary)

For 2D CA, Nx rows x Ny columns, ≤ 256 possible states/cell (alt.: `with rgbimage`):

```
plot '-' binary array=100x100 flipy format='%byte' title '…' with image
```

C++ code:

```
char * bytedata = new char[Nx*Ny];
for(int j=0; j<Ny; j++)
    for(int i=0; i<Nx; i++)
        bytedata[j*Nx+i] = (char)CA_lattice[i][j];
// Send data to stdout in raw (binary) format:
cout.write(bytedata, Nx*Ny);
cout << flush;
```

Gnuplot to visualise streaming data

- Examples of usage:

```
$ ./ca [1000 100 100 setup-gnuplot.gp] | gnuplot -persist

$ ./ca [1000 100 100 setup-gnuplot.gp] | tee output.log | gnuplot –persist
$ gnuplot < output.log

$ ./ca [1000 100 100 setup-gnuplot.gp] | tee output.bin | gnuplot –persist
$ gnuplot < output.bin
```

- Export as figure? Replot to new terminal device:

```
# setup a new plotting terminal device and output:
set term postscript eps # other possible: set term svg, set term png…
set output "file.eps"

# replot the current frame (lattice configuration) after this new setup:
plot '-' binary array=100x100 flipy format='%byte' title '…' with image
…

# reset previous terminal device and output:
…
```

- Alternatively: open pipe stream feeding gnuplot:

in D:

```
File gp;
gp.popen("gnuplot –noraise", "w");
gp.write(…);
gp.flush();
```

⇒ Several gnuplot windows can be opened at once

Simple event loop

- stdout: redirected to gnuplot
- Use stderr for console msg
- Use stdin to wait for user input
- While simulation runs in a separate thread

in D:

```
import core.thread;
…
int main(string[] args) {
    …
    Player ca = new Player(…);
    string menu="Press <RET> for …";
    string s;
    while(1) {
        stderr.writeln(menu);
        stdin.readln(s);
        switch(s) {
            case "p\n": // play
                ca.start(); // play in separate thread to keep control
                stdin.readln(s); // wait for user to say stop/pause
                ca.stop(); // send stop signal to playing thread. Join.
                break;
            …
        }
    }
}
```