Moving On in Swarm

Paul Johnson
Ecological Society of America
Portland, OR
2004/08/02

Where do you stand?

- Expert programmer? No problem, dig in
- Complete Novice: Swarm's not a bad way to learn (IMHO)
  - Get a good book on C:
    - Kochan, *Programming in ANSI C*
    - Kernighan & Richie: *The C Programming Language*
  - Work hard on 1st part of Swarm Tutorial
- Intermediate: Swarm's a good place to learn ideas of Object-Oriented Programming
- Java users: still need to master Obj-C Swarm

Gathering Tools

- Install Swarm
  - pre-built “binary packages”: various platforms
  - get archive and compile: swarm-2.2.tar.gz
  - binary packages preferred for neophytes
- Make sure you have a good editor:
  - Emacs
- gdb : The GNU debugger
- Miscellaneous GNU tools
  - wget
  - ftp

Web sites to remember

Swarm Development Group:

- http://www.swarm.org (old)
- http://wiki.swarm.org (new)

My Swarm HQ:

http://lark.cc.ku.edu/~pauljohn/Swarm

- Swarm packages for Linux users
- Links to many resources (discussed next)

Ecoswarm (Steve Railsback):

http://www.humboldt.edu/~ecomodel/software.htm

Gather Manuals

Put copies in ~/swarm/docs

- The “Objective-C” book in pdf form
- The reference guide for Swarm
  - Don't take the source code for the guide
  - Look for a bland name like “set-html-2.2.tar.gz”
- Swarm User Guide
  - A [immodesty alert] pretty good discussion of Swarm in Objective-C
  - Look for it in html or postscript or pdf format
- SwarmOnlineFaq
- Keep a copy of the Swarm Source code, even if you don't compile Swarm

Should also get...

- R http://www.r-project.org
  - The wonderful free/open statistical powerhouse.
- Drone: Ted Belding (U. Mich) tool for batch processing simulations
- Some programs may require addon libraries:
  - GSL: The GNU Scientific Library
  - BLAS (linear algebra)
  - Swarm GraphLib
  - UM-EXPtools

Email Lists

- join swarm-support and swarm-modelling (via www.swarm.org)

Swarmapps

- Swarmapps is a tar.gz file containing the Swarm Tutorial as well as other demonstration programs.
- Most recent official release:
  - swarmapps-2.1.1.tar.gz
- Newer snapshots are available from Paul Johnson <pauljohn@ku.edu> or directly from Swarm's online code (CVS) repository

Read through the Swarm Tutorial

- Steps from elementary C to the design of Swarm models.
- Recently added elements
  - “batch” processing of simulations
  - parameter classes & command line arguments
  - data collection
- Get the newest version of swarmapps, because it has new components
  - simpleObserverBug3
  - simpleBatchBug1-3

Shop for working programs

- Swarm changes, programs change, not all work all of the time
- Get programs from authors or on the web
- Swarm ftp site has
  - apps/objc/sdg apps/objc/contrib
  - apps/java/sdg apps/java/contrib
- Make sure a program compiles & runs before you exert any effort on it.
- If you find a program is out of date, contact the author directly. Don't be bashful.

Small Working Examples

- Marcus Daniels (SDG) wrote many small programs that illustrate usage of specific Swarm elements.
- Best way to learn “how to” use a particular thing.
- Best way to get help and report bugs
- These and others are collected in the WorkingExampleCode directory referred to in SwarmFAQ

Questions to ask about a model

- What do these agents “do”?
- How do they interact?
  - meet each other?
  - detect changes in environment?
- How are their actions “interleaved” in time?
  - **synchronous**: all step at same time, don't impact environment until all have acted.
  - **asynchronous**: each one steps and registers its impact on the environment
  - event-driven (dynamic) scheduling

Scheduling

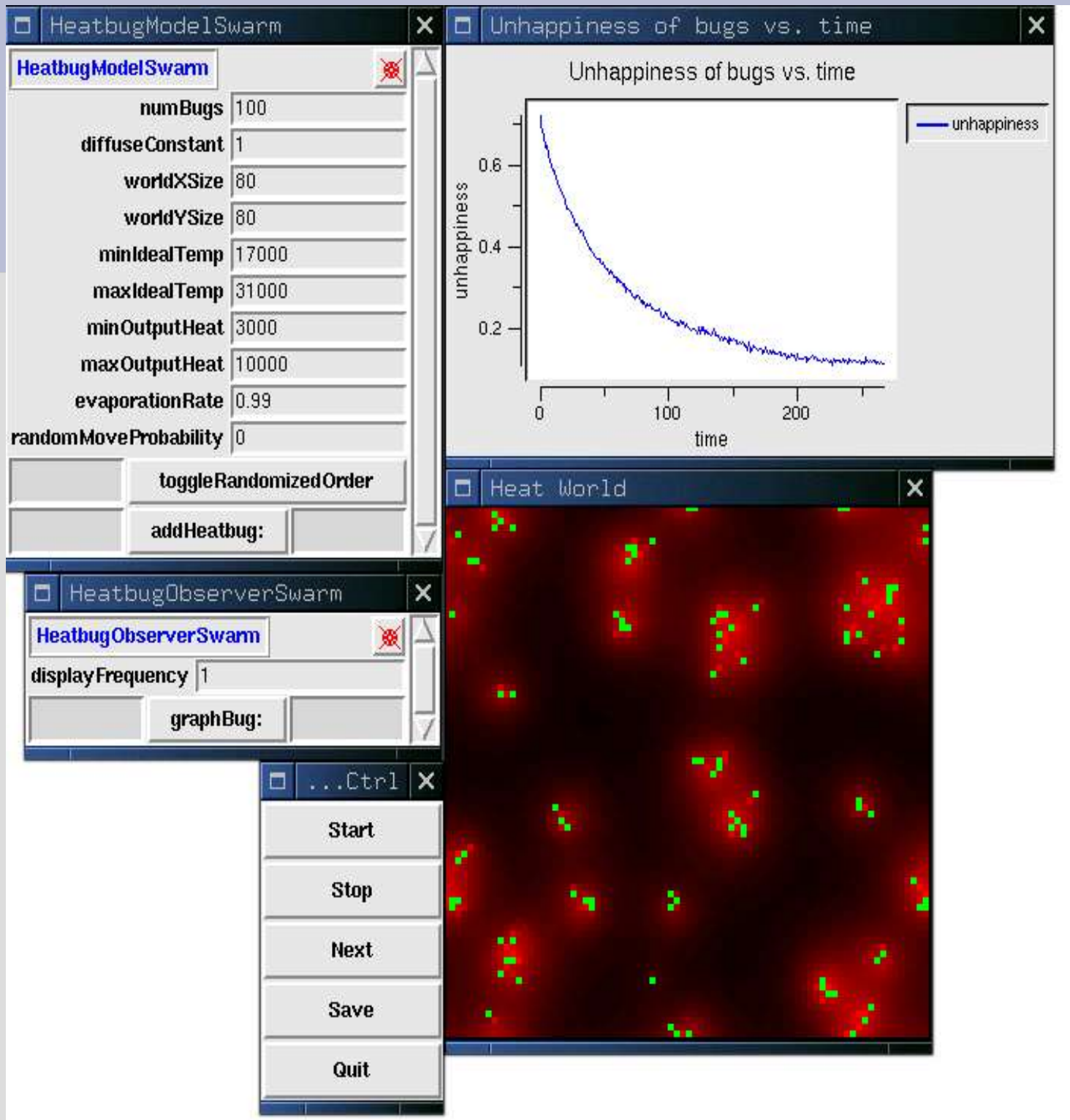
- Regular (process a collection of agents)

or

- Dynamic (Event-driven)

Heatbugs: Prototype Swarm Application

- After tutorial, Heatbugs should be the first model you run
- Agents are bugs seeking “just the right” temperature
- Each bug deposits heat onto a “HeatSpace”
- Each bug moves in a 2d grid that is “overlaid” on the HeatSpace


HeatbugModelSwarm
HeatbugModelSwarm
numBugs 100
diffuseConstant 1
worldXSize 80
worldYSize 80
minIdealTemp 17000
maxIdealTemp 31000
minOutputHeat 3000
maxOutputHeat 10000
evaporationRate 0.99
randomMoveProbability 0
toggleRandomizedOrder
addHeatbug:
Unhappiness of bugs vs. time
Unhappiness of bugs vs. time
unhappiness
unhappiness
0.6
0.4
0.2
0
100
200
time
Heat World
HeatbugObserverSwarm
HeatbugObserverSwarm
displayFrequency 1
graphBug:
...Ctrl
Start
Stop
Next
Save
Quit

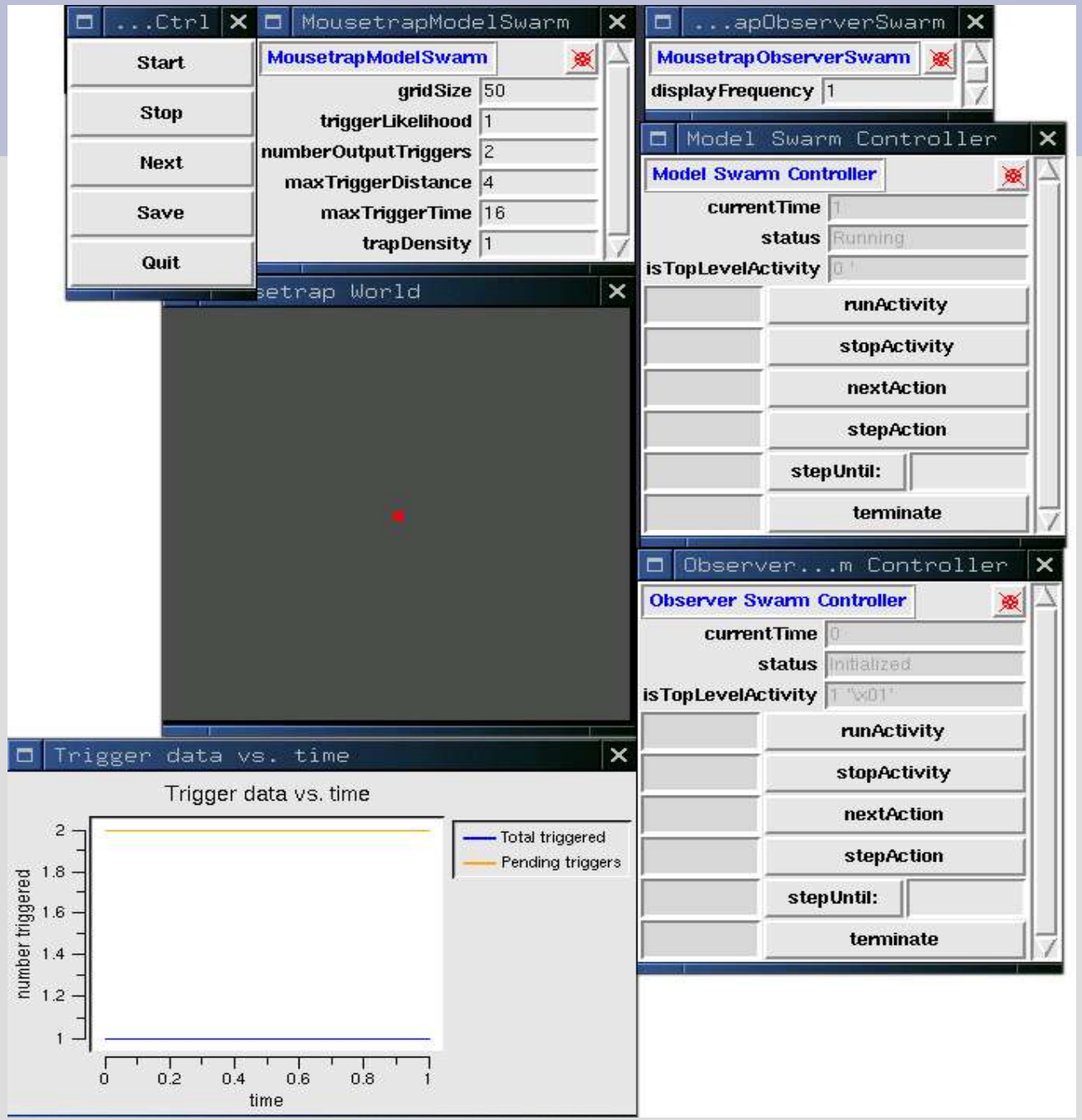
Heatbugs, cont.

- Bug Interactions:
  - No direct interaction
  - Prevented from “overlapping” on grid
  - Bugs create and adjust to heat in HeatSpace
- Schedule: repeated 'trips through the list'
  - Possibly randomized
- Batch mode: run with -b, note the Graphs write out their number streams to files
  - limited usefulness (IMHO), except it demonstrates “fork” in main.m between GUI ObserverSwarm and BatchSwarm

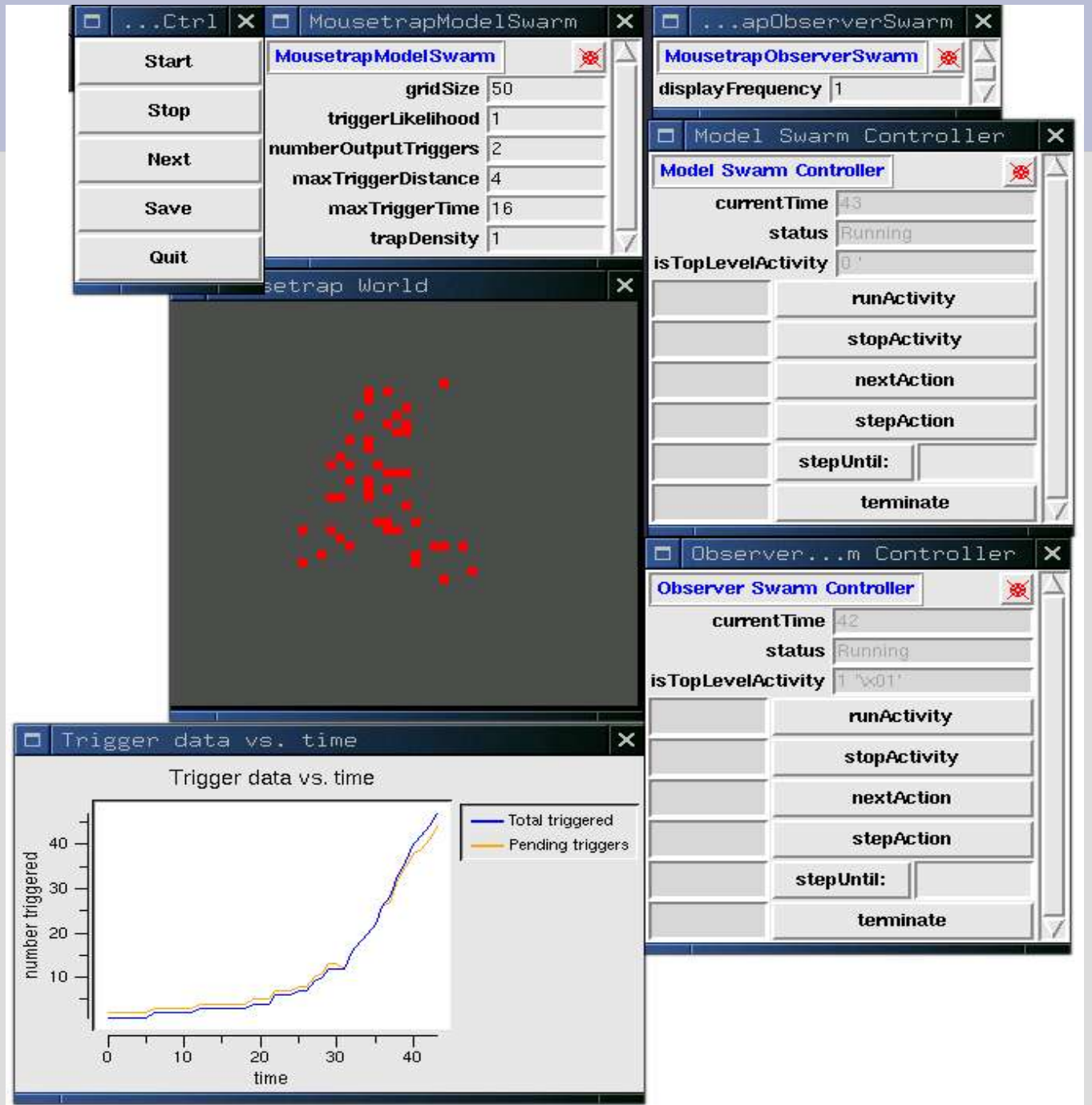
Dynamic Scheduling: Mousetrap

- Most notable event-driven Swarm simulation
- There's a “master schedule” in ModelSwarm
- Mouse traps “go off” and then notify ModelSwarm that other traps are supposed to go off at a future time
- Not completely “decentralized” in the bottom-up sense
- A true bottom-up scheduling arrangement is possible (pjrepeater* examples), but technically equivalent

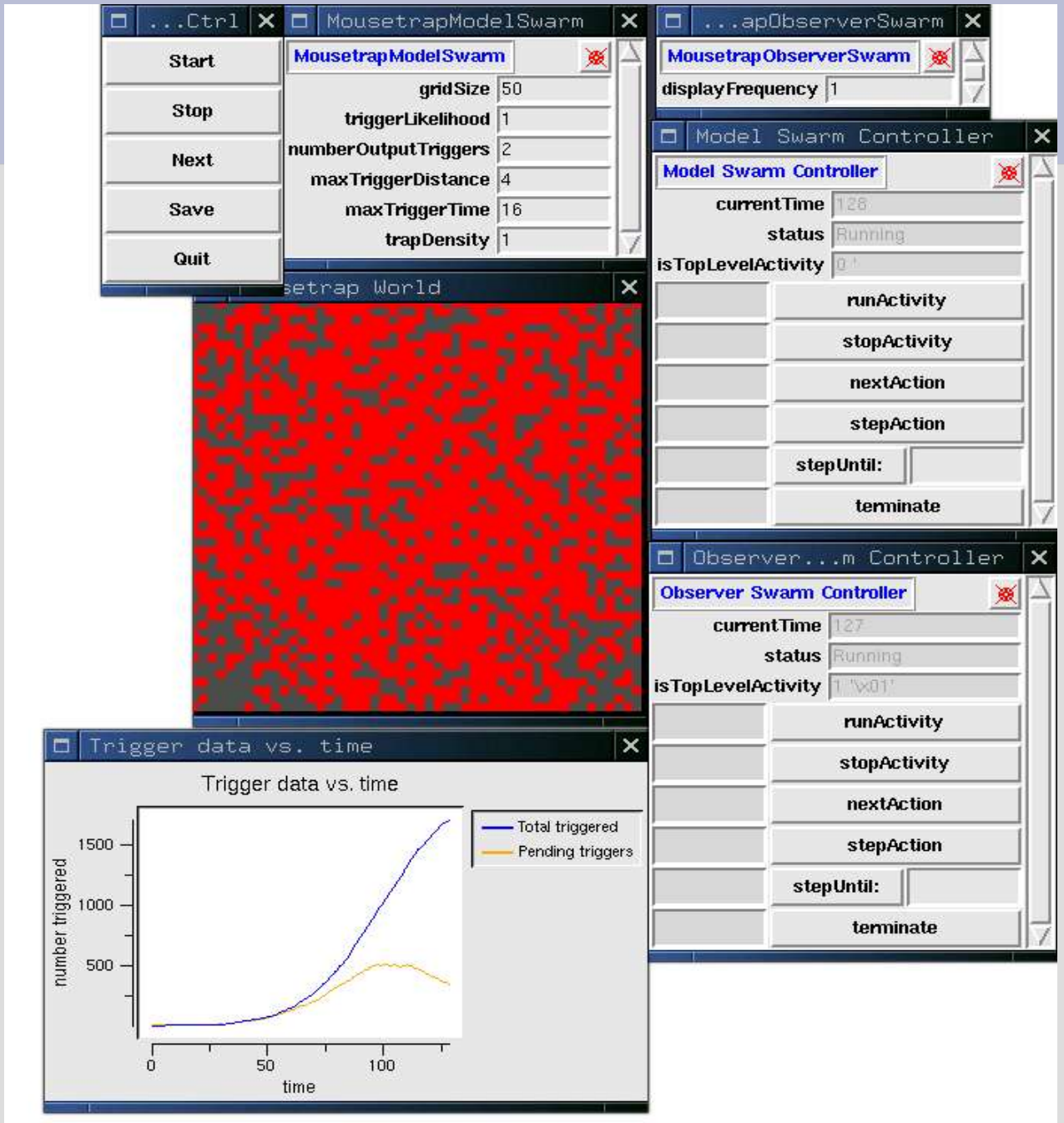
Mousetrap start



Mousetrap: midpoint



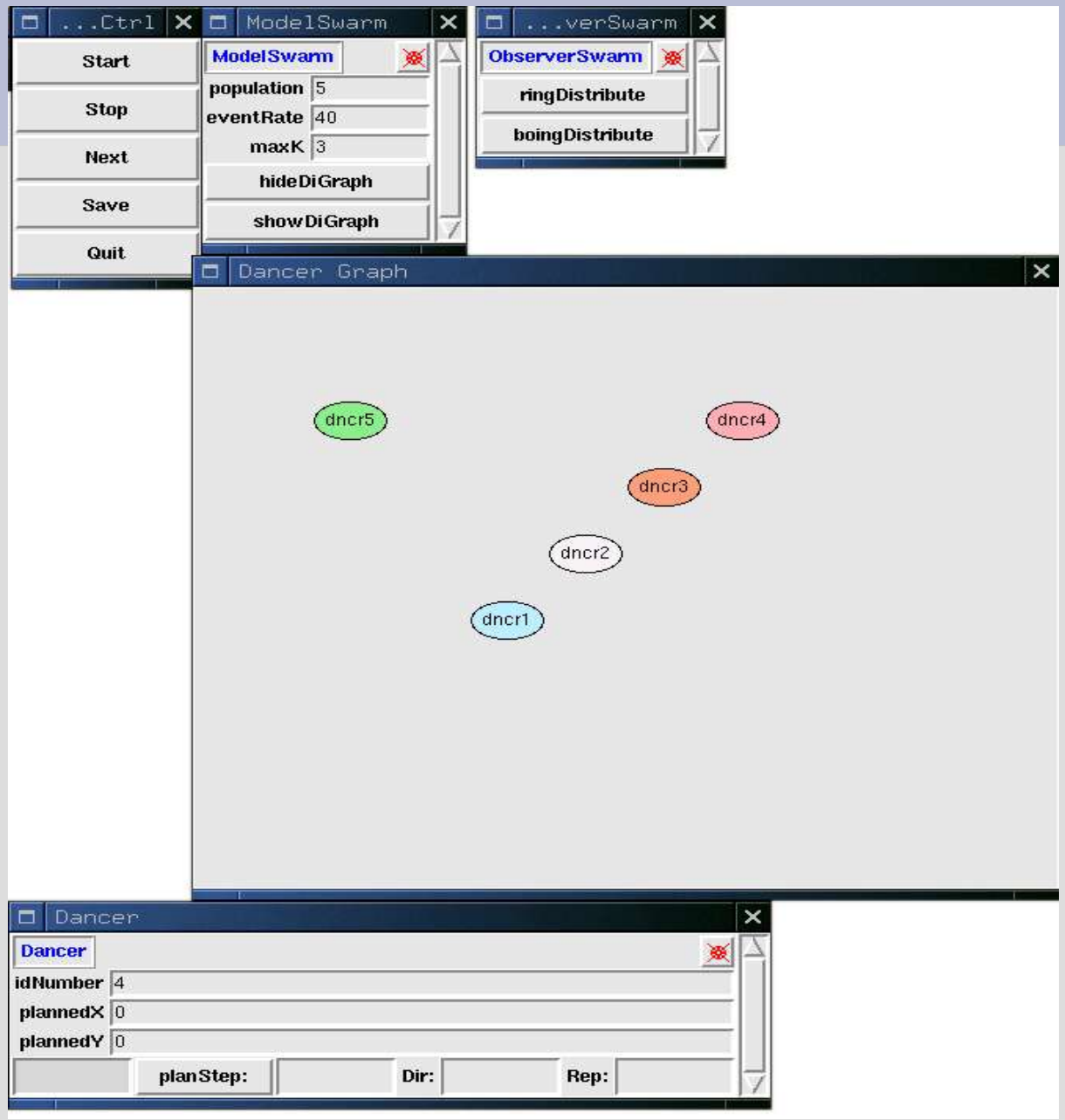
Mousetrap: finished



Dynamic Scheduling: Ballet

- Tina Yu & Paul Johnson, “Tour Jeti, Pirouette: Dance Choreographing by Computers,” YELM Journal (2003).
- Dancers have a list of dance steps and a “transition matrix”
- Dance Step take a variable number of time steps
- Swarm model has dancers “schedule themselves” for new steps X timesteps into future (asynchronous, dynamic scheduling).

Dancer


...Ctrl
Start
Stop
Next
Save
Quit
ModelSwarm
ModelSwarm
population 5
eventRate 40
maxK 3
hideDiGraph
showDiGraph
...verSwarm
ObserverSwarm
ringDistribute
boingDistribute
Dancer Graph
dncr5
dncr4
dncr3
dncr2
dncr1
Dancer
Dancer
idNumber 4
plannedX 0
plannedY 0
planStep:
Dir:
Rep:

Brief Interlude for Whining about nonDynamic Scheduling

- Scheduling: “createActionForEach” often causes more trouble than its worth
- Easier to
  - create a for loop that processes agents
  - myLoop {
    ...[do something for each element in a collection];
    }
- In buildActions:

```
[schedule at: 0 createActionTo: self
                        Message: M(myLoop)];
```

Scheduling Opinion, cont.

- Reasons to take “loop” approach
  - keeps agent actions “together in time”
  - faster because it does not invoke the “deep down” scheduling apparatus so much
  - avoids major hassles, especially when writing models in Java
- Counter argument:
  - Sometimes you want to throw actions onto the pile at a given time and want them all “mixed up”

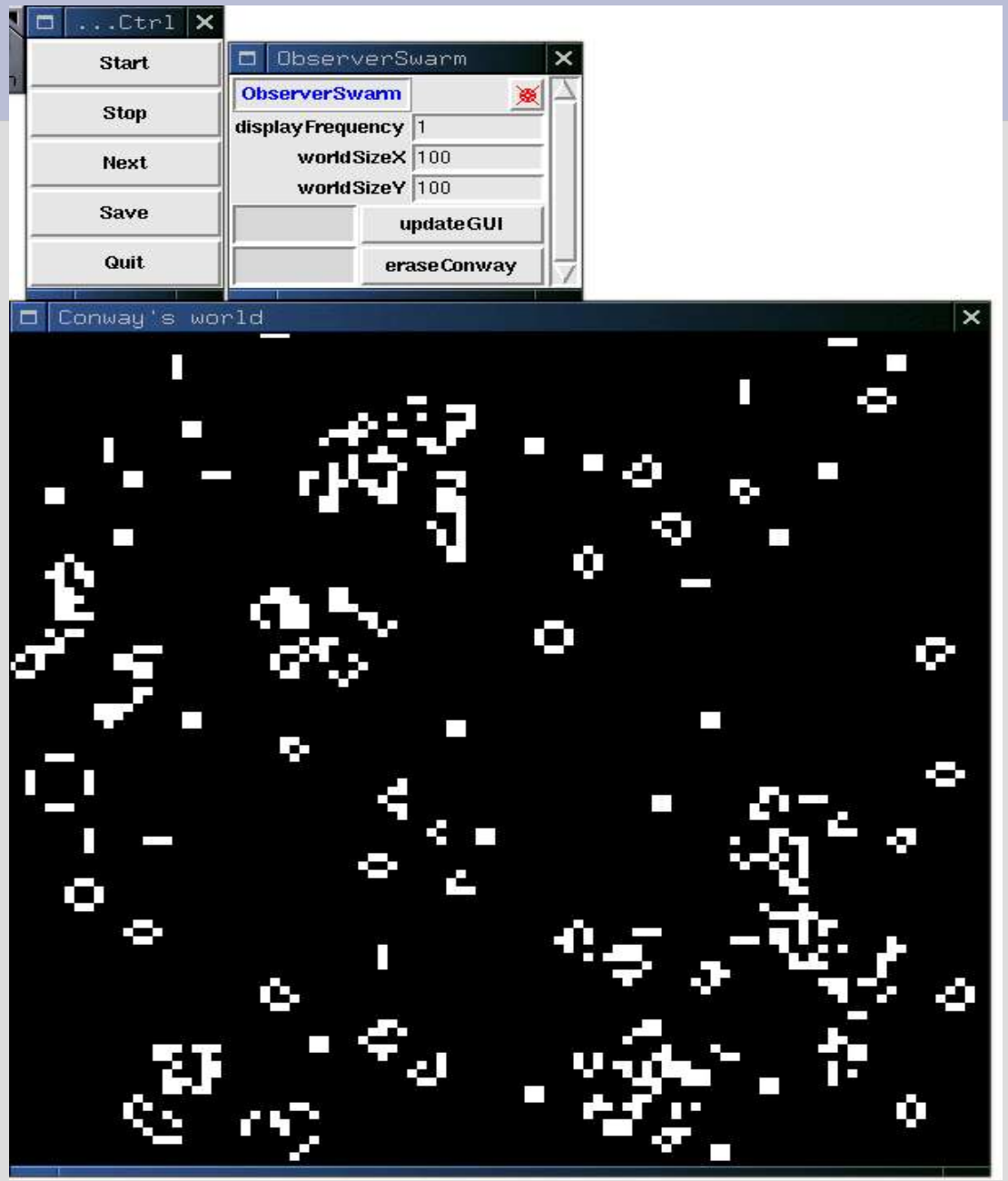
Asynchronous And Synchronous

- Commonly mistaken as a Swarm library issue.
- Actually, its an issue of conceptualization and user model design
- Sudden Impact: Does programmer intend agents to have impacts on environment/other agents that are immediately?

Cellular Automata

- CA can be written in Swarm
- Conway Game of Life (conway-1.1-Swarm-2.2.tar.gz available online

Conway


...Ctrl
Start
Stop
Next
Save
Quit
ObserverSwarm
ObserverSwarm
displayFrequency
1
worldSizeX
100
worldSizeY
100
updateGUI
eraseConway
Conway's world

Scheduling in Game of Life

- Game of life has no “agents”
- The cells are updated at each step
- Double-buffered “grid”
  - each cell is updated against a snapshot of the grid from the previous period
  - after all cells are updated, then their status is drawn onto the grid
- This is SYNCHRONOUS updating

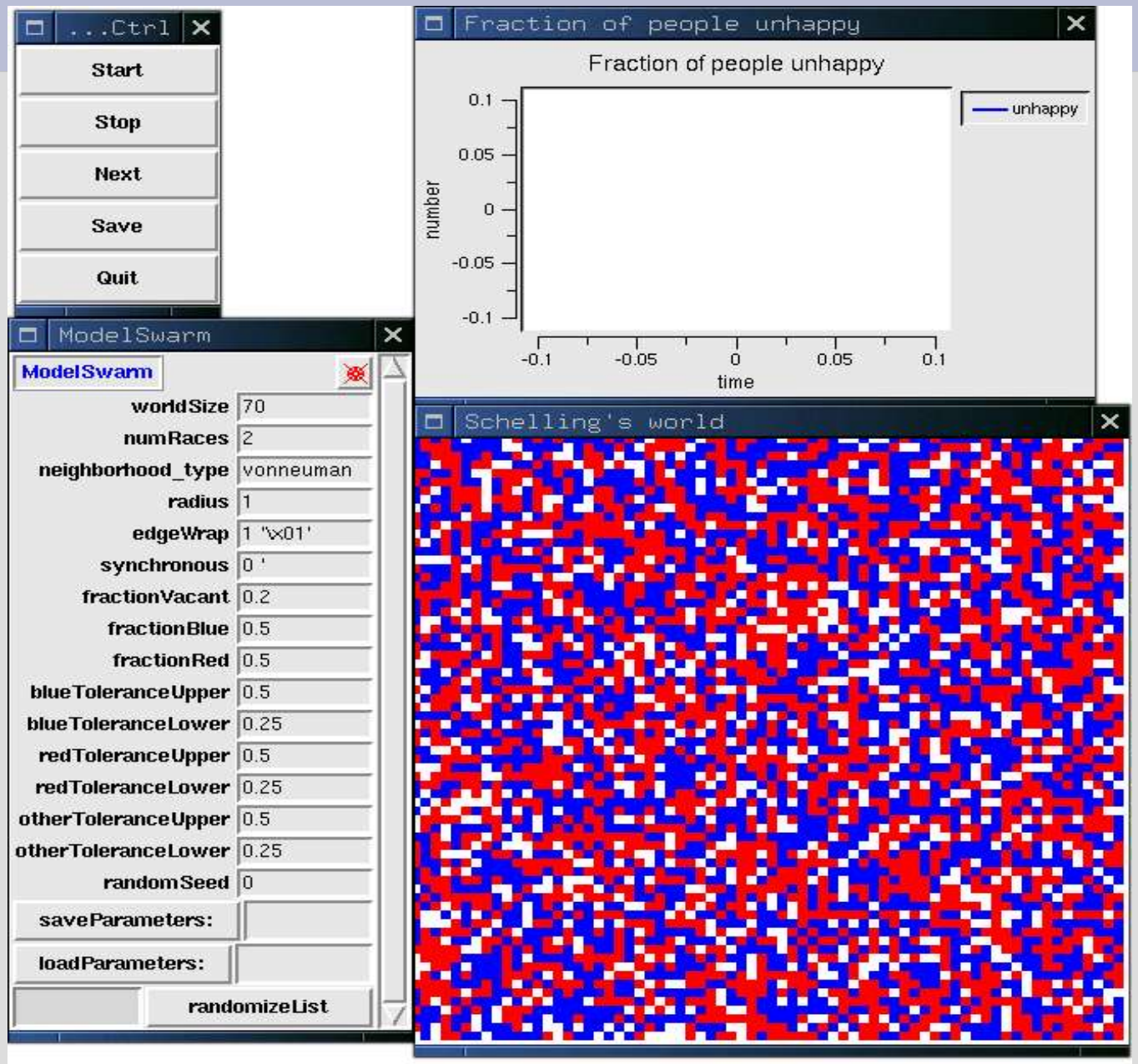
Schelling2

- Thomas Schelling, “Dynamic Models of Segregation”, J. Math. Soc, 1971
- Agents move in response to hi/low levels of diversity in local environment
- schelling2 Code available MySwarmCode

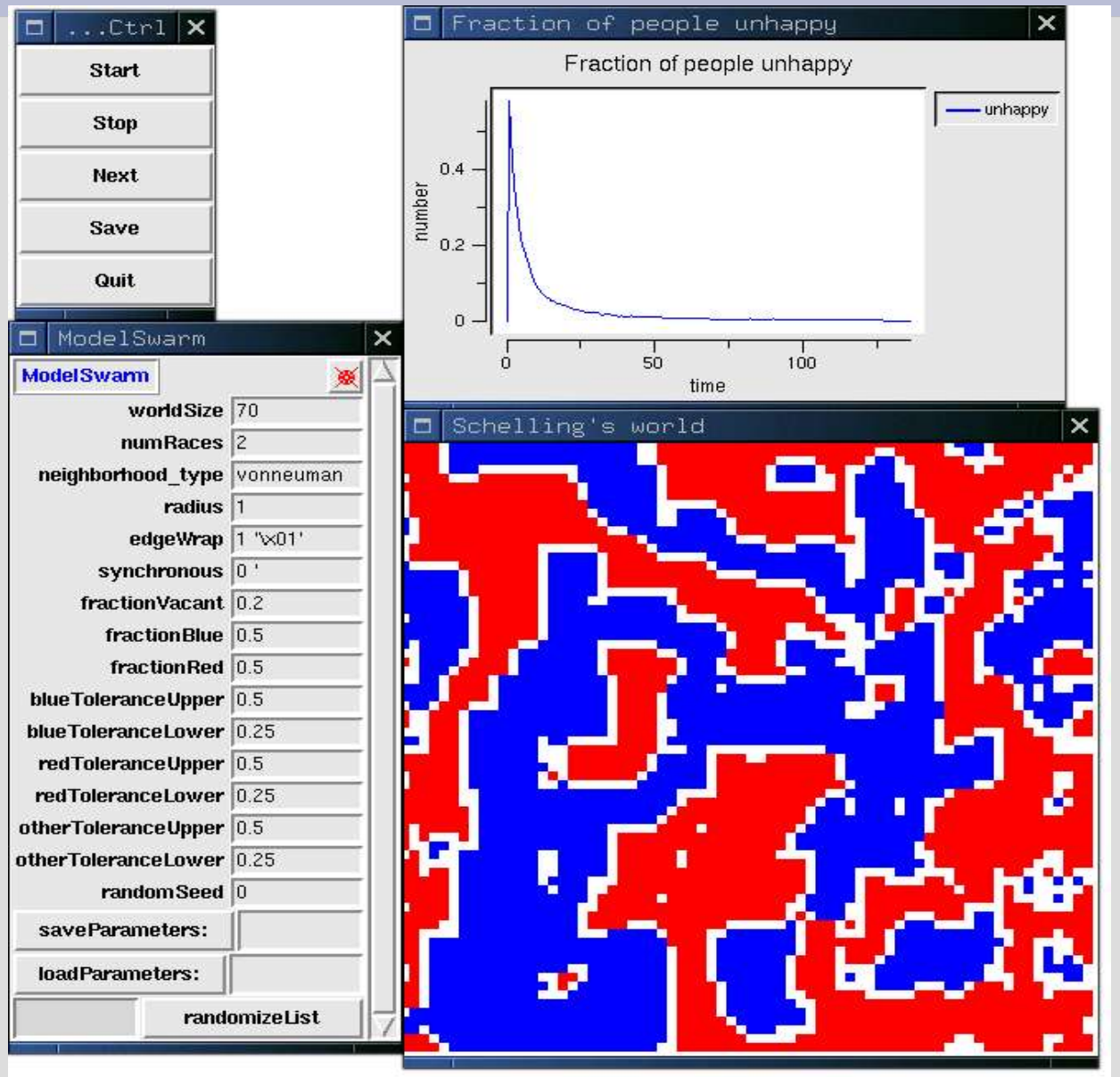
Schelling2 Runtime Options

- ASYNCHRONOUS or SYNCHRONOUS
- Load & save parameter files
- Set Neighborhood type- Moore or VonNeumann
- Radius of neighborhood
- Edge effects & Wrap Around
- Randomized ordering of agent actions at each step

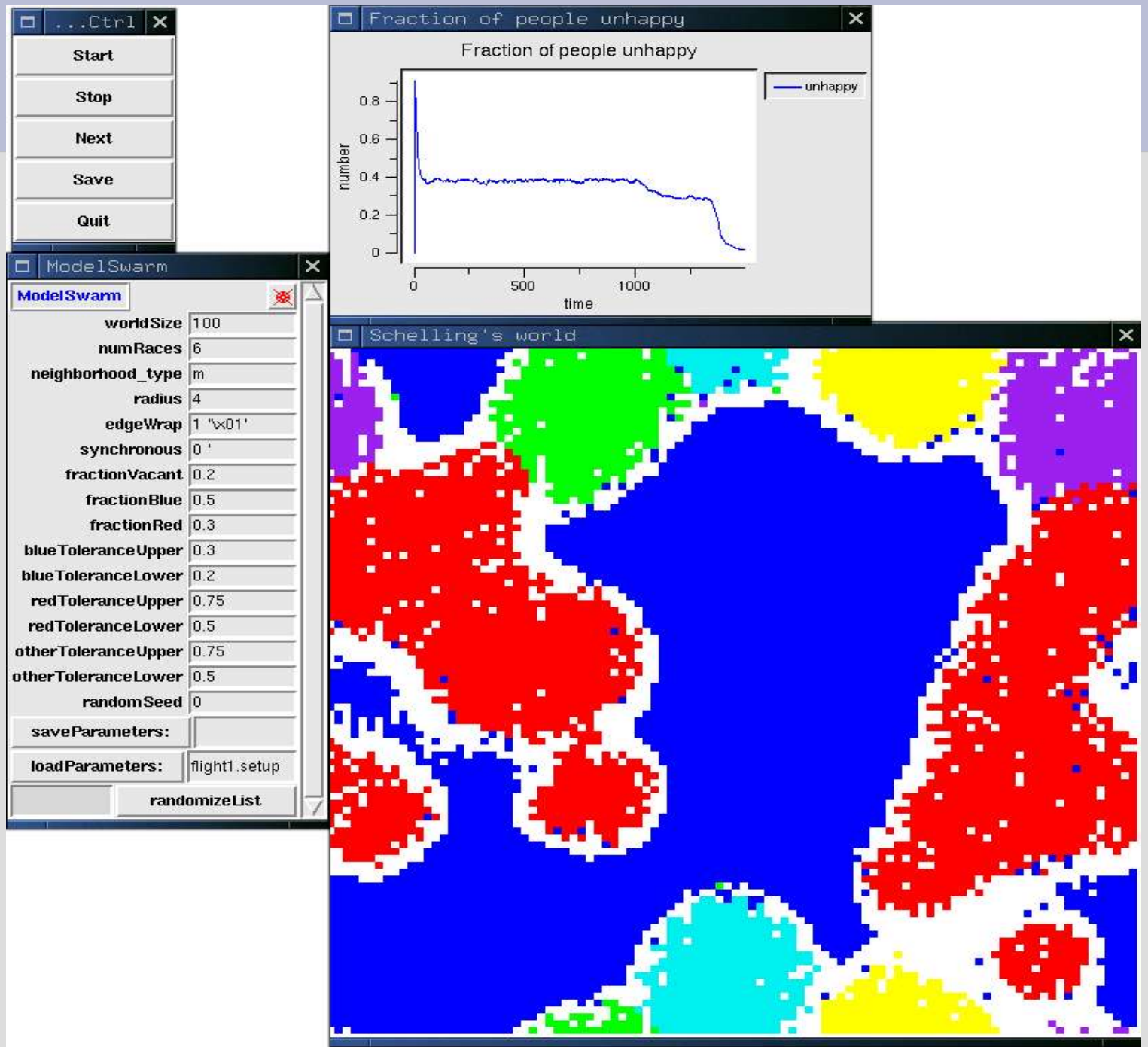
Standard Schelling Start



Standard Schelling End



Explore: flight1.setup



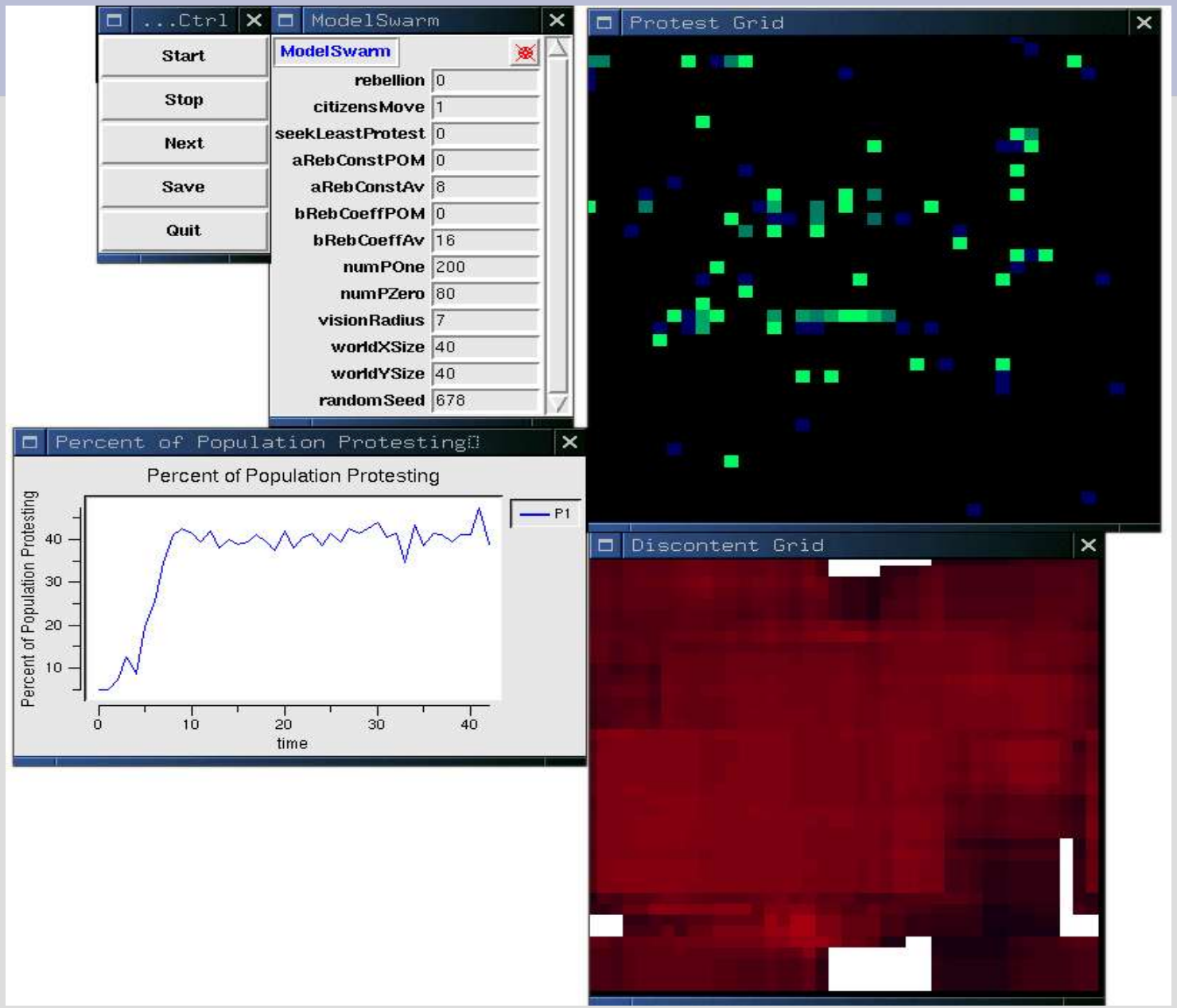
Protest Activist Model

- Brichoux and Johnson, “Power of Commitment in Collective Action”, JASS (2002).
- “Activists” code available PJ's “MySwarmCode/Protest”
- Agents on a grid
- Can (optionally) move
- Can protest if they are unhappy or want change
- Agents “view” limited number of cells in their vicinity

Protest #2

- SYNCHRONOUS compiler flag
  - each agent chooses next behavior on the basis of a “snapshot” of community at previous instant
  - SYNC can produce “modeling artifacts” (Huberman and Glance, ,)
- ASNCHRONOUS model:
  - each agent's action registers in eyes of others “right away”
  - more realistic?

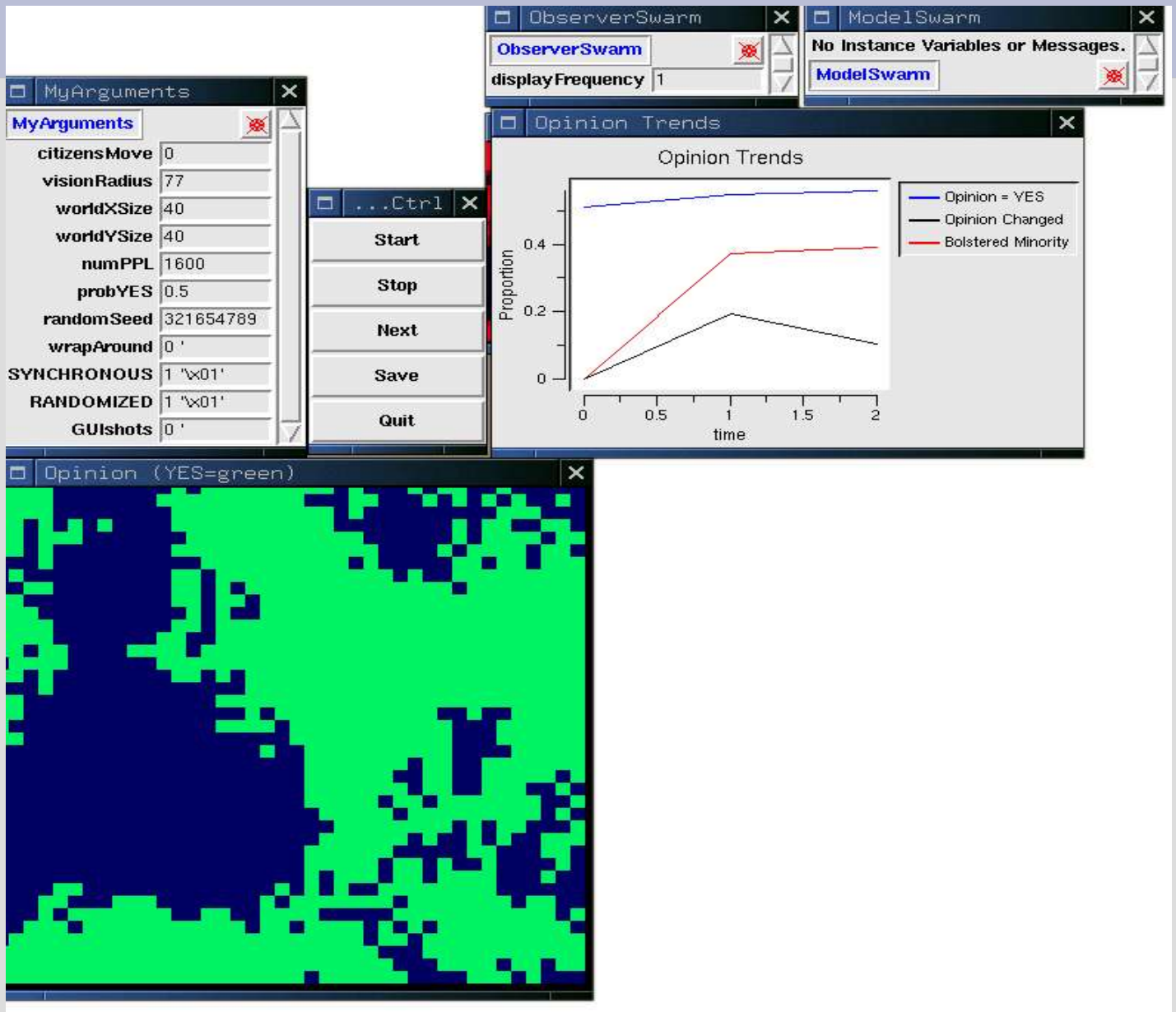
Protest snapshot



Social Impact Model

- Nowak & Latane, Social Impact Model
- A classic cellular automaton
- Agents change YES or NO depending on social pressure (distance weighted)
- Swarm “SIM” available PJ's MySwarmCode
- Swarm SIM model implements ASYNCHRONOUS option
- Swarm SIM implements “variable neighborhood size”

Social Impact Model



Collector Grids™

- Speed: Swarm Library problem or User problem?
- Activists, SIM, Schelling2 use “collector grids” to register the actions of agents.
- Too slow to have each agent search each neighboring cell for each step
- Faster to have agents “take action” and register that action on all cells within “eyesight”.
- Other agents can obtain “visible activity” with a single check or a Grid position.

“Full Service” Swarm models

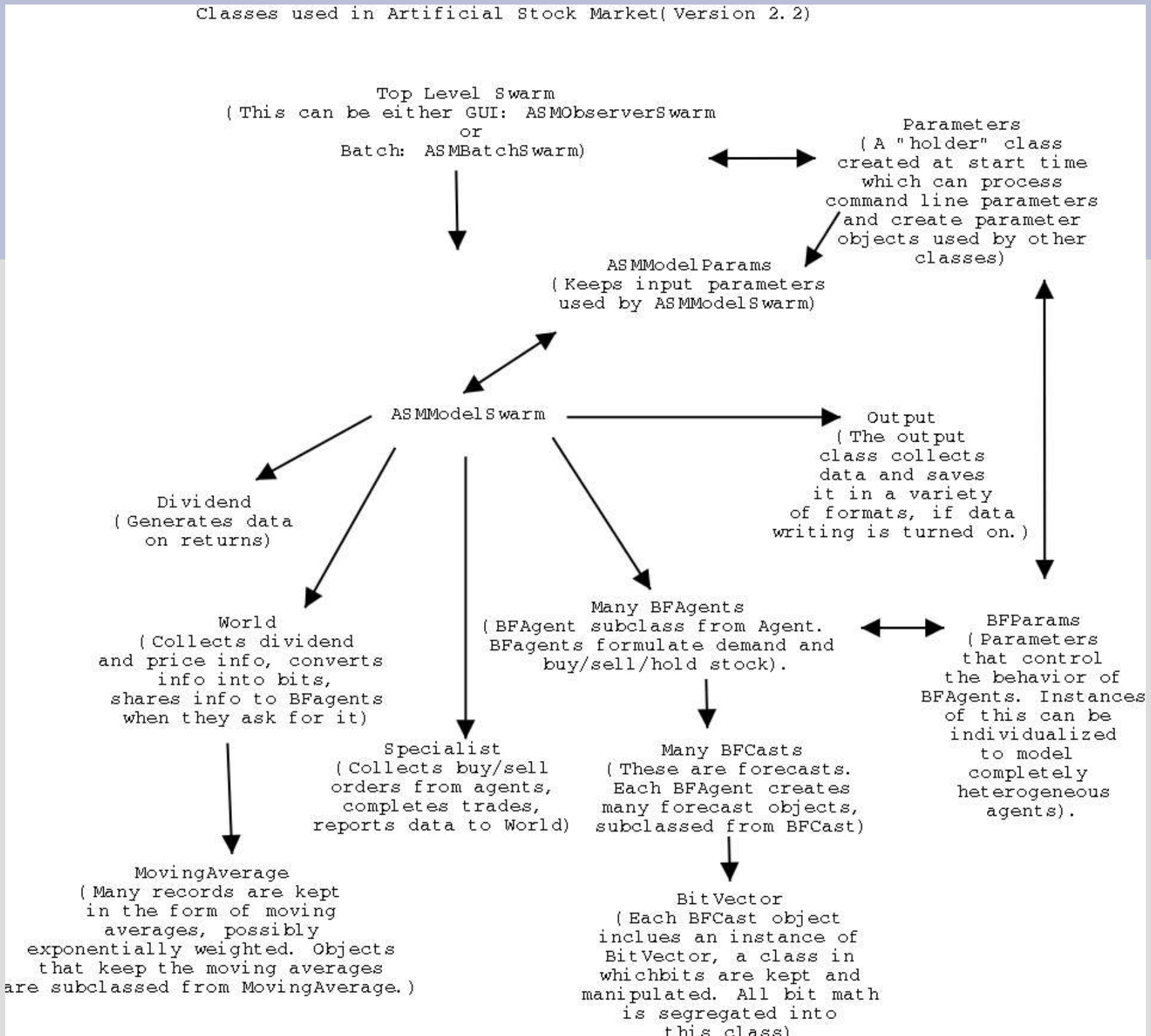
- GUI output for diagnostics and interaction
- Clear Summary Measures & Indicators
- Batch output
  - run model repeatedly
  - allows “command line” arguments for “parameter sweeps” (look for Parameter classes)
- Serialization & stability analysis
  - details, see details in presentation Sfest03_serialization.pdf
- Scheduling variations
- Documentation

Artificial Stock Market

- Pioneering ABM study (LeBaron, et al).
- Swarm project on Sourceforge
  http://ArtStkMkt.sf.net
  Code revisions discussed Johnson, “Agent-based Modeling...”, Soc. Sci. Computer Review, 2001.

What's in the ASM?

- Agents buy or sell a single stock
- Agents receive info on the world and on stock price patterns
- Each agent has an intricate “mental model” of the world (Genetic Algorithm)
- Agents invest in isolation: never meet
- Runs for hours in order for agents to “learn”


Classes used in Artificial Stock Market(Version 2.2)
Top Level Swarm
(This can be either GUI: ASMObserverSwarm
or
Batch: ASMBatchSwarm)
Parameters
(A "holder" class
created at start time
which can process
command line parameters
and create parameter
objects used by other
classes)
ASMModelParams
(Keeps input parameters
used by ASMModelSwarm)
ASMModelSwarm
Output
(The output
class collects
data and saves
it in a variety
of formats, if data
writing is turned on.)
Dividend
(Generates data
on returns)
World
(Collects dividend
and price info, converts
info into bits,
shares info to BFagents
when they ask for it)
Many BFAgents
(BFAgent subclass from Agent.
BFagents formulate demand and
buy/sell/hold stock).
BFParams
(Parameters
that control
the behavior of
BFAgents. Instances
of this can be
individualized
to model
completely
heterogeneous
agents).
Specialist
(Collects buy/sell
orders from agents,
completes trades,
reports data to World)
Many BFCasts
(These are forecasts.
Each BFAgent creates
many forecast objects,
subclassed from BFCast)
MovingAverage
(Many records are kept
in the form of moving
averages, possibly
exponentially weighted. Objects
that keep the moving averages
are subclassed from MovingAverage.)
BitVector
(Each BFCast object
inclues an instance of
BitVector, a class in
whichbits are kept and
manipulated. All bit math
is segregated into
this class)

ASM In Action

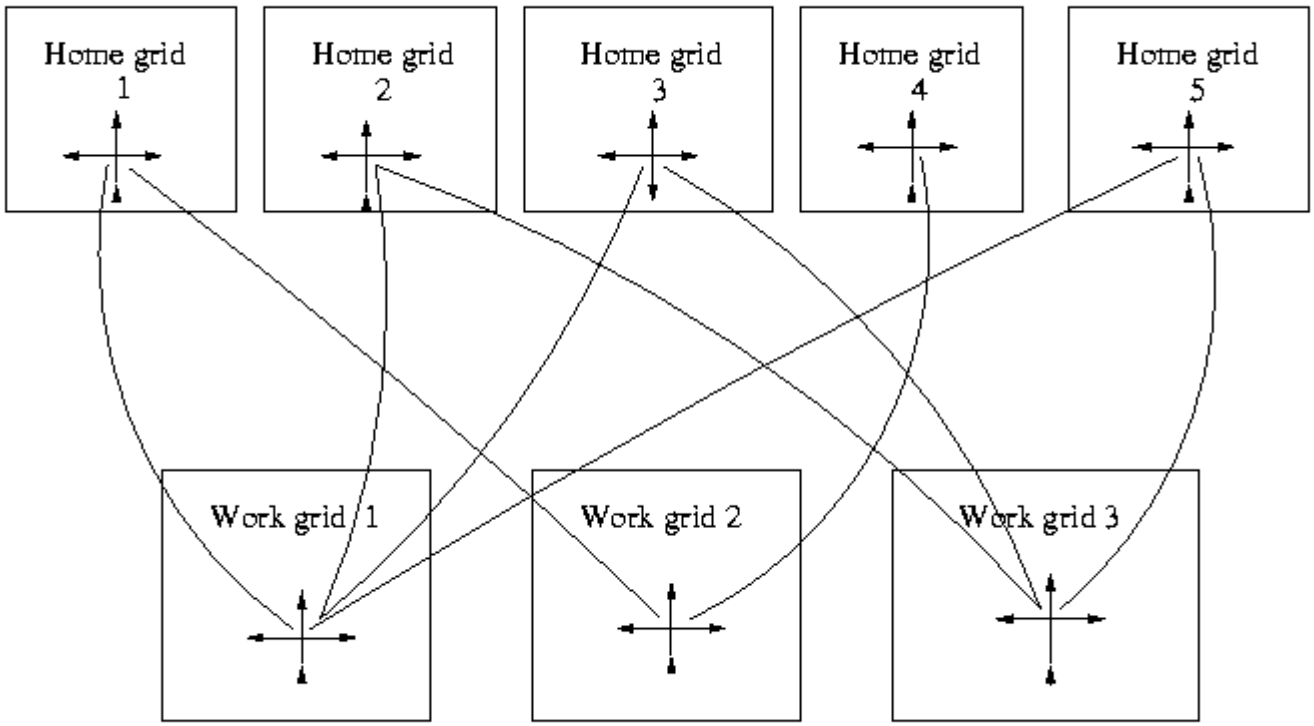
ASM: Serialization

- ASM-2.4 implements Serialization:
  - able to save entire state of simulation and restart
  - valuable because of long “burn in” time for ASM
- Serialization allows one to change agent behavioral assumptions within a “stabilized” context.
- Developing “Social ASM” in which agents can copy from each other

Public Opinion (home & work)

- Huckfeldt, Johnson, Sprague, *Political Disagreement: The Survival of Diverse Opinions within Communication Networks* (Cambridge, 2004)
- Code available PJ's MySwarmCode
- Agents interact only when they
  - find another available agent and
  - choose to initiate interaction
- Various behavioral premises
- (Comparatively) complete documentation

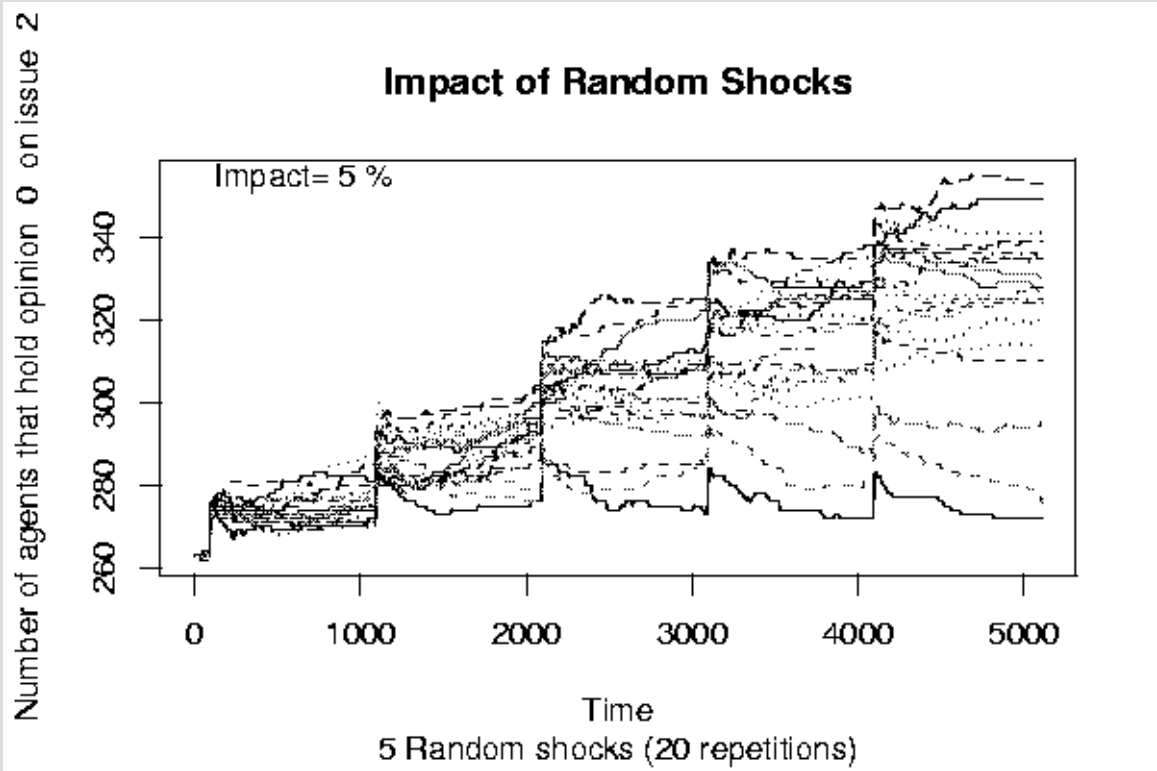
Many agents per cell allowed



Opinion Model #2

- Full implementation of Swarm serialization in LispArchiver format
- Run model to equilibrium
- Restart repeatedly after small random shocks.

20 restarts


Impact of Random Shocks
Impact= 5 %
Number of agents that hold opinion 0 on issue 2
260
280
300
320
340
0
1000
2000
3000
4000
5000
Time
5 Random shocks (20 repetitions)

Opinion Model #3

- Thorough example of batch processing.
- Makes picture (png format) snapshots of grids at designated intervals.
- Text output: use C commands to write text into files
- Unix tools for post-processing data files (tail, etc) & R scripts for graphs
- Some (smarter) users prefer HDF5 output which can be obtained from EZGraph

Multi-Agent Grids

- Original Swarm designers always considered Grid2d with one agent per cell
- Sometimes we want multi-agent cells
- Sven Thommesen developed 1st prototype of multi-agent grid (MoGrid2d)
- PJ's MultiGrid2d is MoGrid2d on steroids.
  - answers all ordinary Swarm instructions suitable for grids
  - allows full customization of “cell sites” to allow diagnostic information collection