Power Tour of Swarm Apps

Paul Johnson
Swarmfest 2005 Tutorial
Torino, Italy
2005/06/05

3 Questions to ask about a model

- 1. What do these agents “do”?
- 2. How do they interact & get information?
  - meet each other?
  - detect changes in environment?

Scheduling

- 3. How are their actions “interleaved” in time?

Ordinary Models: repeatedly process a collection of agents (perhaps shuffle)

Scheduling Details

A. Synchronization

- **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

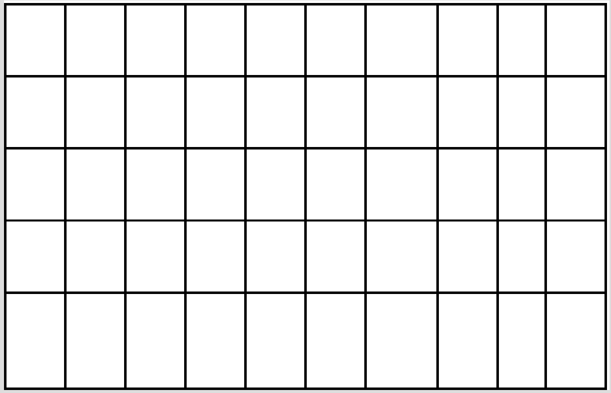
B. Dynamic Scheduling: Events add items to schedules

Cellular Automata

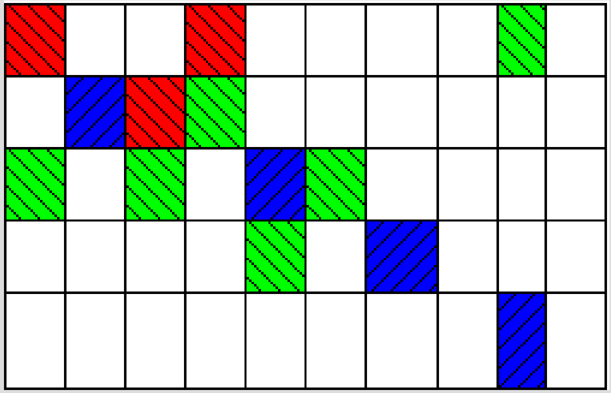
- CA can be written in Swarm
- Conway Game of Life

Cellular Automata (CA)

- Can be written with Swarm
- World is a grid of cells

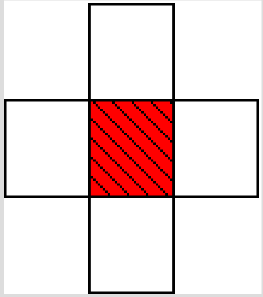


- Colors represent condition (state)

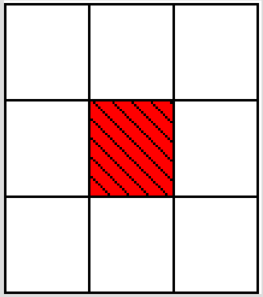


Rules for Updating Cells

- Rules specify state transition
- Usually depend on neighborhood



VonNeumann (4)

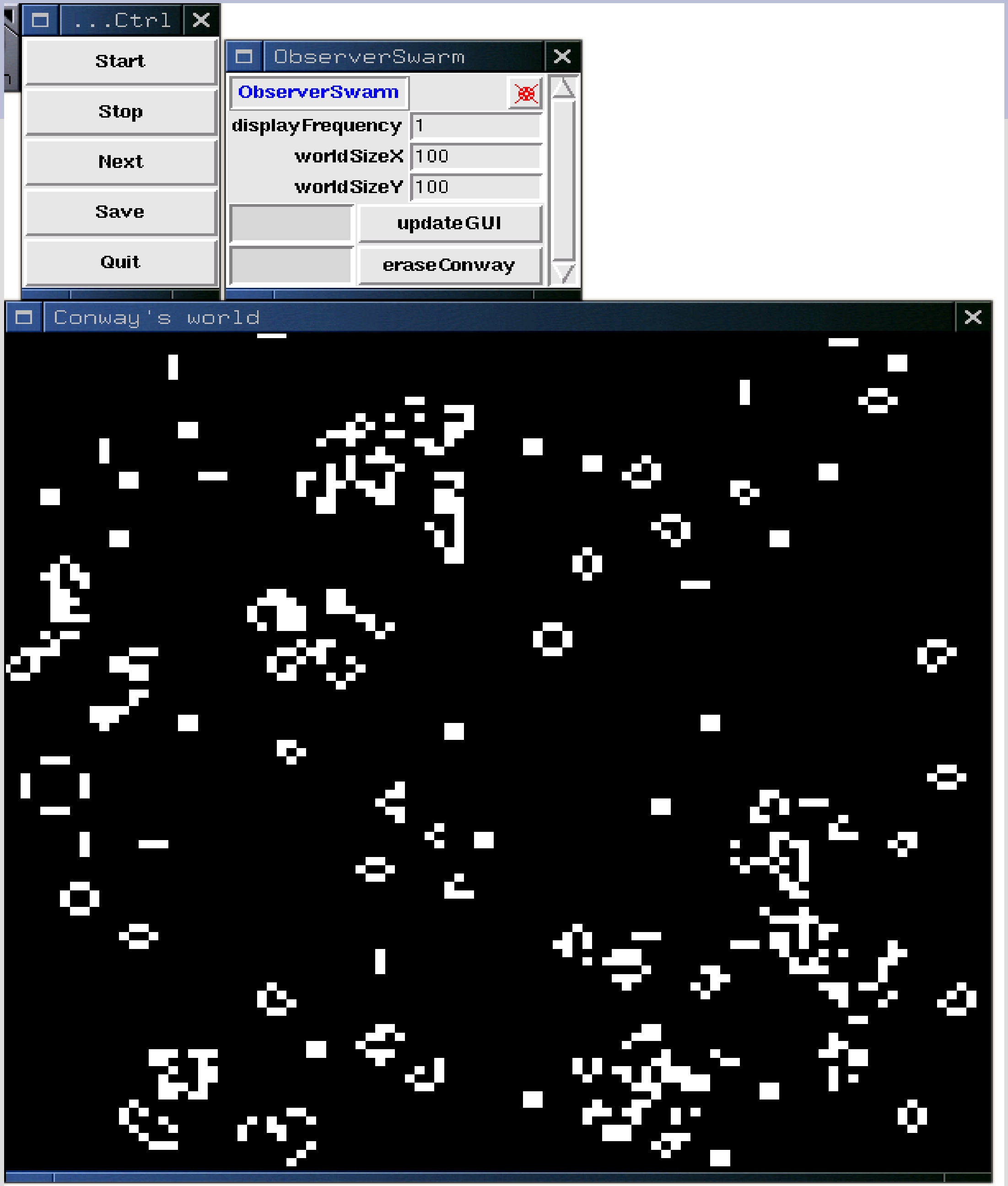


Moore (8)

Conway's Game of Life

- Martin Gardner, “The Fantastic Combinations of John Conway's new solitar game “life”” *Scientific American*, 223, (1970)
- 2 States: on / off (alive / dead)
- Cells die if they are lonely (< 2 neighbors)
- Cells die if too crowded (> 3 neighbors)
- Cells turn on if neighbors = 3

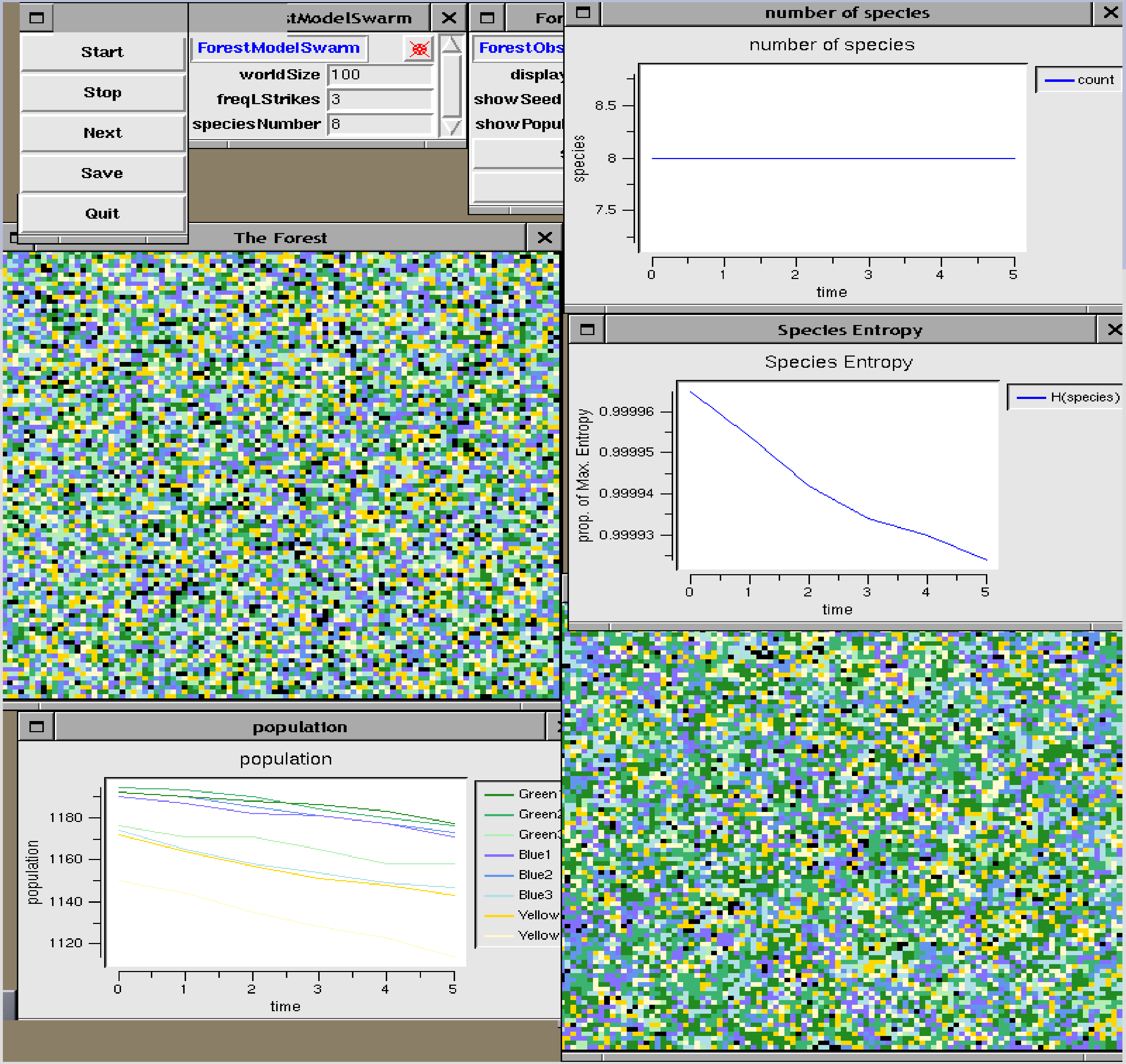
Conway



displayFrequency 1
worldSizeX 100
worldSizeY 100

Arborgames

- Melissa Savage & Manor Askenazi (SFI)
- Several Species of Tree
- Tree occupies cell & sheds seeds in neighbors
- Seeds may grow on open cells in “young forest”
- Die if “mature forest” has no opening
- Fires
- Code significantly revised for inclusion in swarmapps-objc-2.2

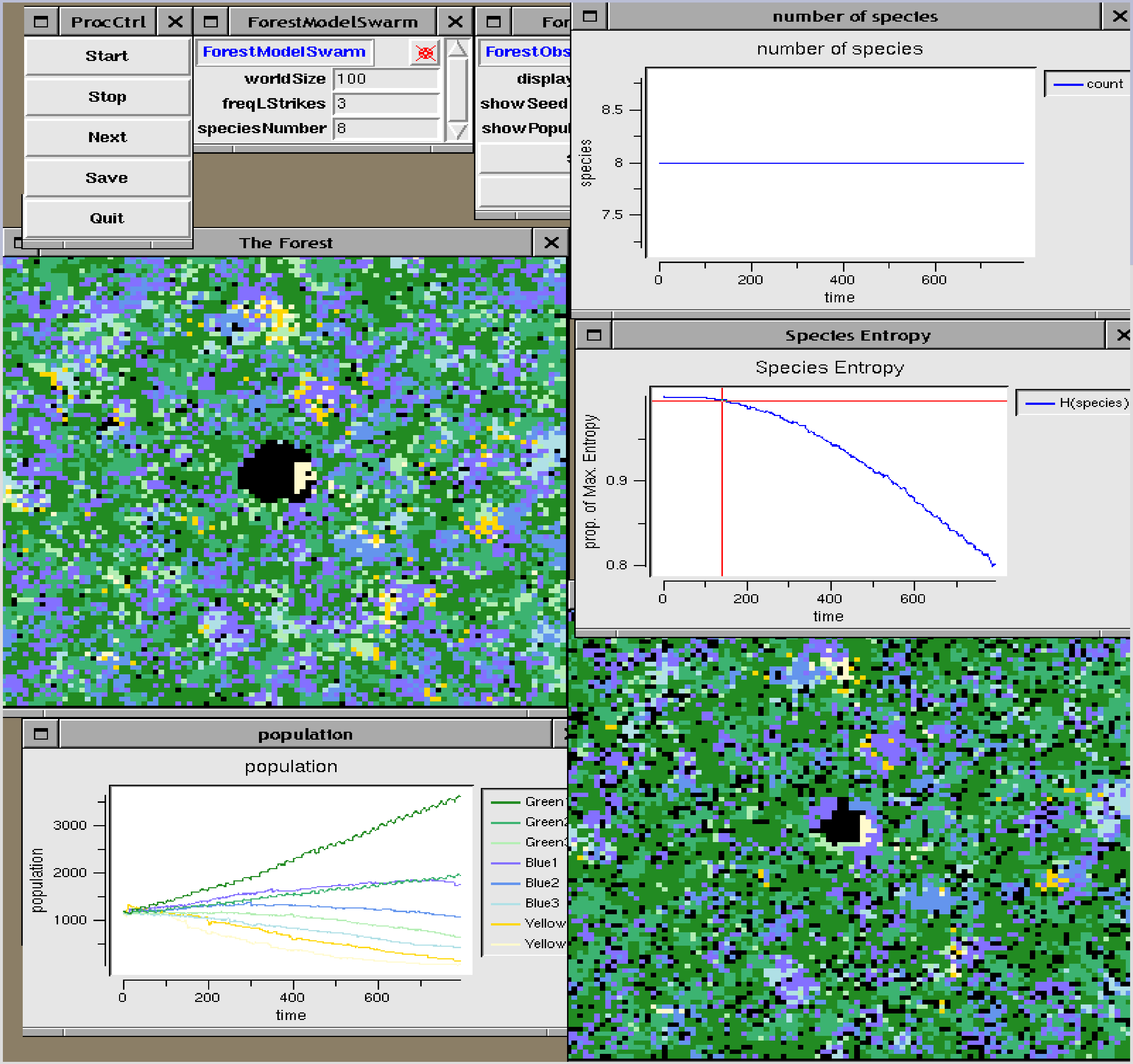


ForestModelSwarm

worldSize 100

freqLStrikes 3

speciesNumber 8



ForestModelSwarm

worldSize 100

freqLStrikes 3

speciesNumber 8

Look Under the Hood

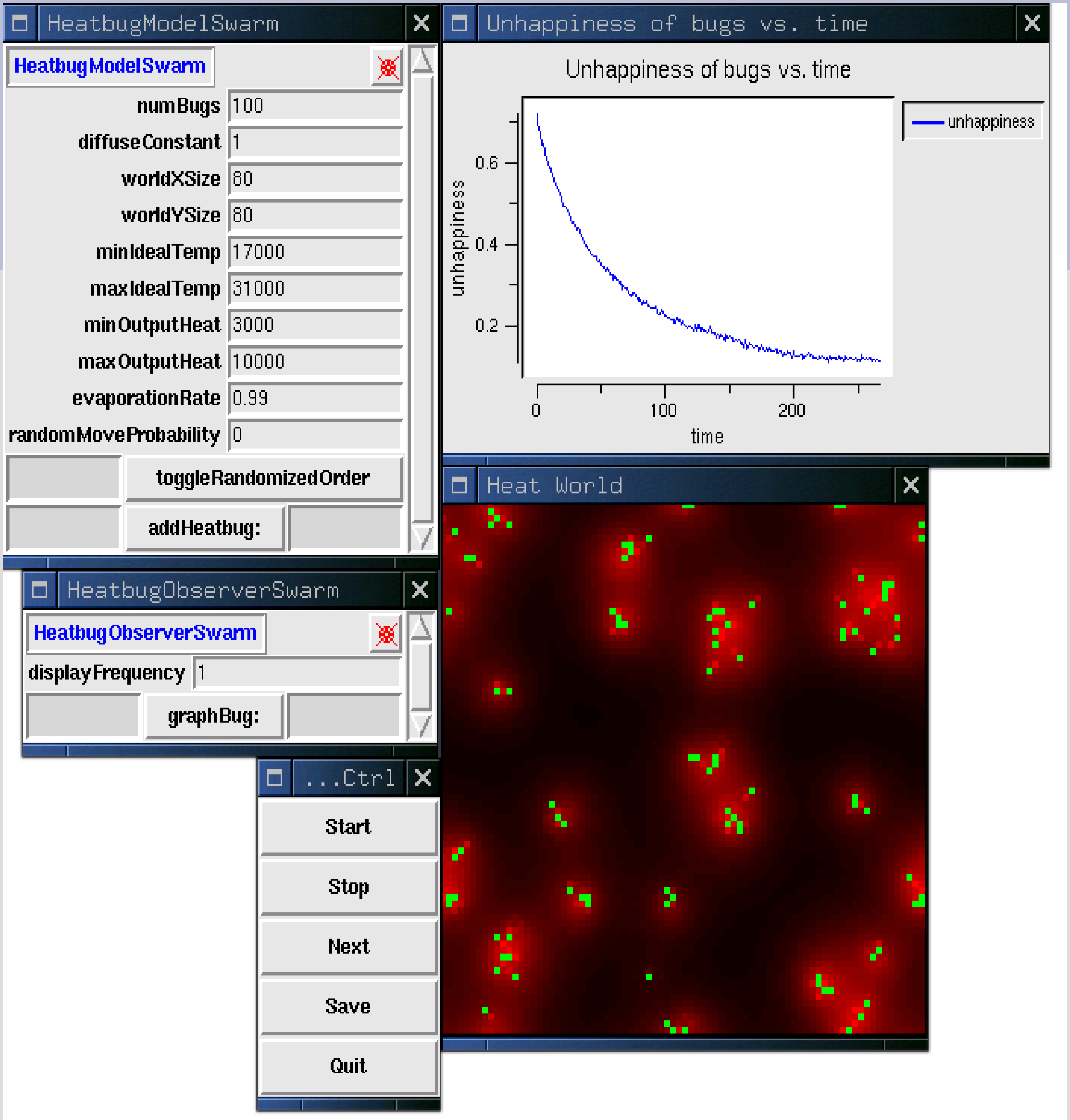
- A whole bunch of cellular automata running at once!
- Young forest
- Mature forest
- Fire grid
- Seed Grids (1 for each Species)

Scheduling in Game of Life

- All 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

Heatbugs: Prototype Swarm Application

- 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

numBugs	100
diffuseConstant	1
worldXSize	80
worldYSize	80
minIdealTemp	17000
maxIdealTemp	31000
minOutputHeat	3000
maxOutputHeat	10000
evaporationRate	0.99
randomMoveProbability	0

HeatbugObserverSwarm

displayFrequency	1

Bug Interactions

- Look at “step” method in Heatbug.m
- Bugs adjust to heat in HeatSpace
- No direct interaction between bugs
- Prevented from stepping onto occupied cells

Heatbugs: Synchronous?

- Both ASYNC and SYNC elements
- SYNCHRONOUS: Heat grid is not updated until all bugs (agents) move

Look in HeatbugModelSwarm.m for "updateLattice" message sent to "heat"

-

Heatbugs: Asynchronous Aspects

- Agent Schedule: repeated 'trips through the list'
- Possibly randomized
- Agents reposition themselves one-at-a-time
- Agents will not move onto an occupied square

Heatbugs: Cool Gadgets

- GUI buttons interact with agents
- Pixmaps for bugs (compiler flags)
- Batch mode: run with -b: demonstrates “fork” in main.m between GUI ObserverSwarm and BatchSwarm

“createActionForEach”

- HeatbugModelSwarm “buildActions”
- Simple Old-style Method

```
actionForEach =
   [modelActions createActionForEach:
      heatbugList message: M(step)];
```

• Faster, new “multilanguage” method used if Compiler flag FAST is set

```
#make EXTRACPPFLAGS=-DFAST
```

•5% difference on my laptop

A Simpler Approach

- Create method that processes agents

```
- myLoop {
    id anAgent, index;
    index= [heatbugList begin: self];
    for (anAgent=[index next]; [index
      getLoc]==Member; anAgent = [index next])
        ...[do something for each element in a collection];
        }
```

- In buildActions add:

```
[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”

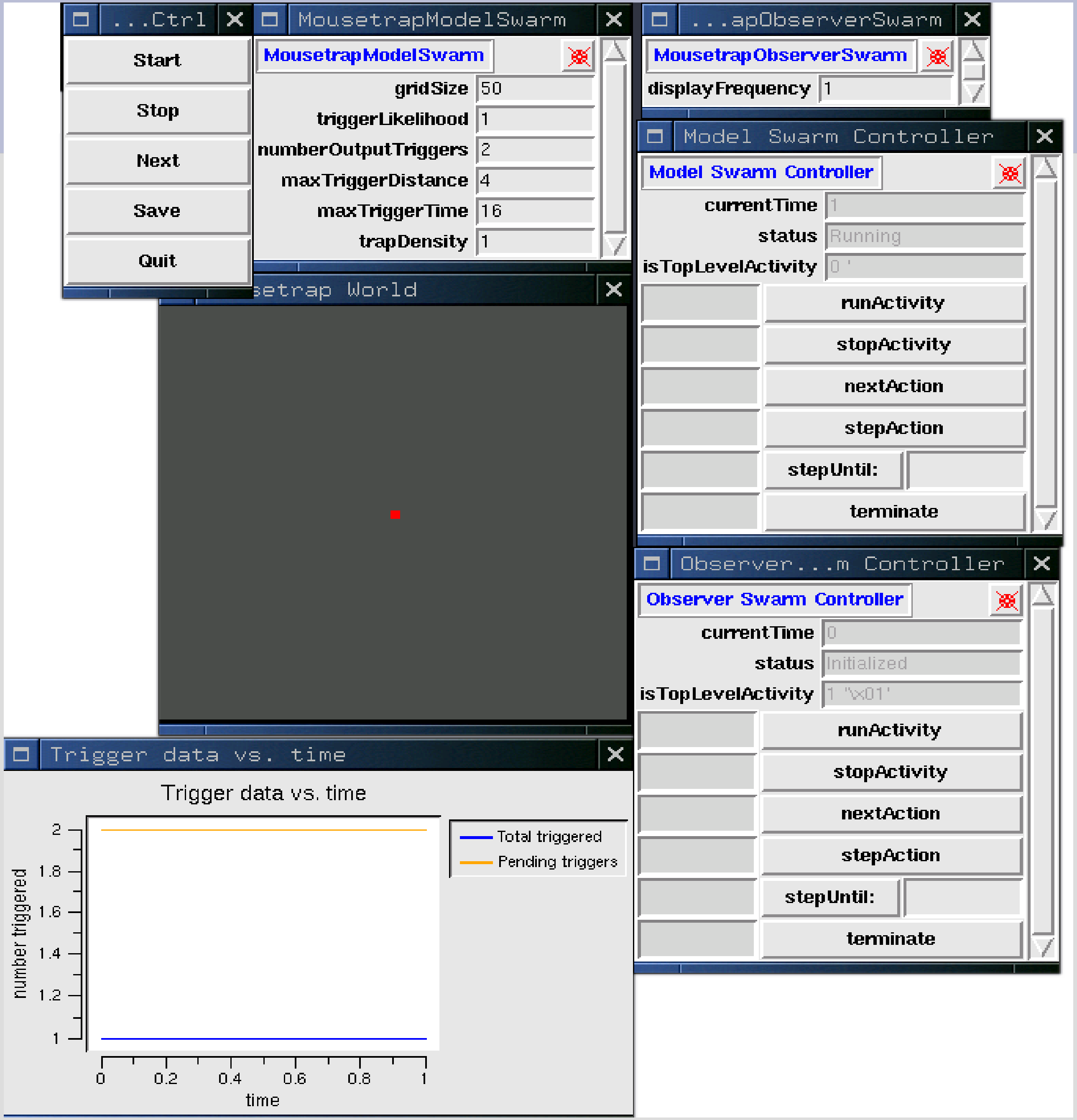
But if you really want speed

- Use gcc profiler to find slow parts of model
- Revise code:
  - Reduce use of % operator makes model much faster (about 1/3)

Dynamic Scheduling: Mousetrap

- 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
-

Mousetrap start



MousetrapModelSwarm

gridSize	50
triggerLikelihood	1
numberOutputTriggers	2
maxTriggerDistance	4
maxTriggerTime	16
trapDensity	1

MousetrapObserverSwarm

displayFrequency	1

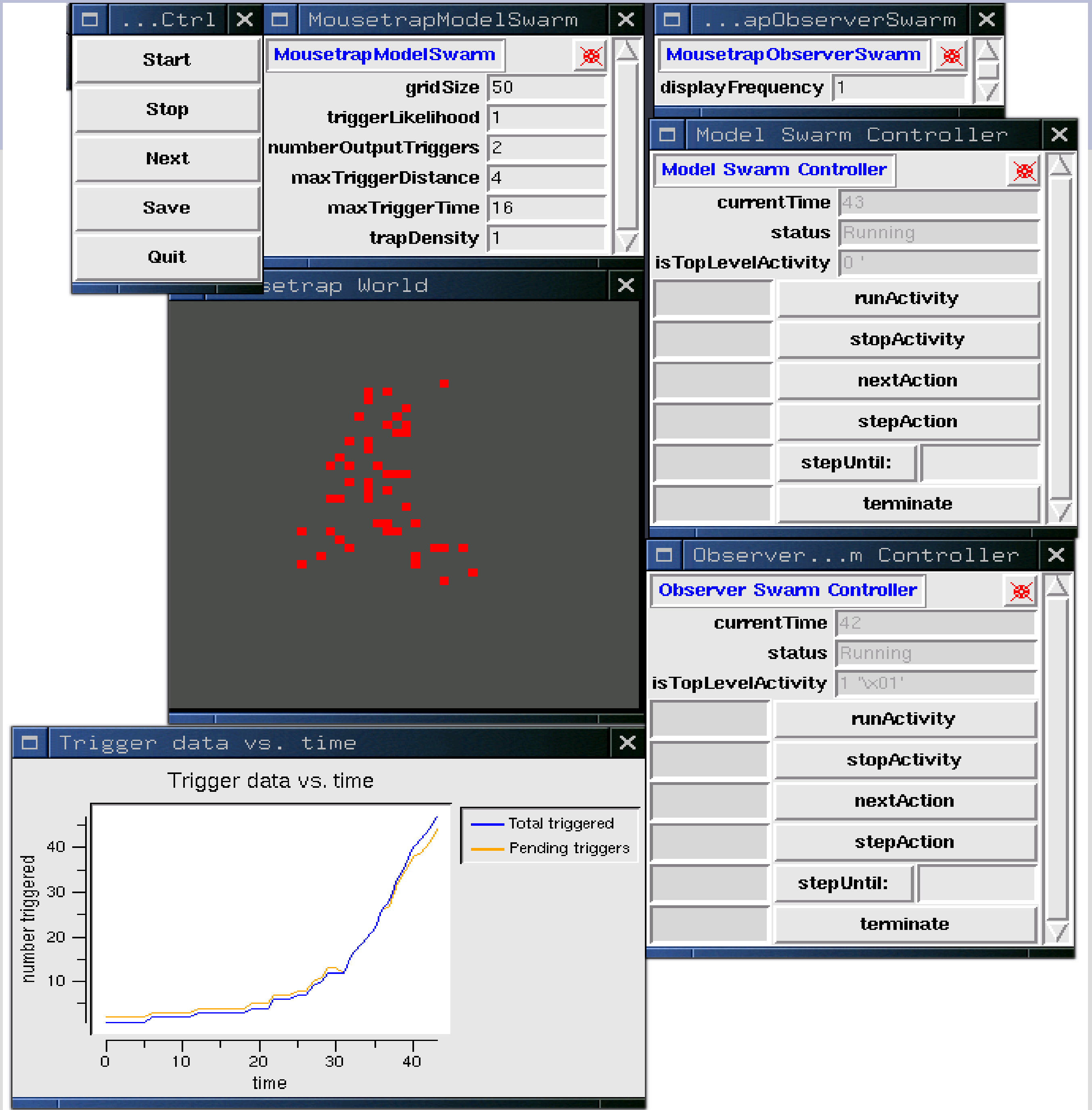
Model Swarm Controller

currentTime	1
status	Running
isTopLevelActivity	0 '

Observer Swarm Controller

currentTime	0
status	Initialized
isTopLevelActivity	1 '\x01'

Mousetrap: midpoint



MousetrapModelSwarm

gridSize 50
triggerLikelihood 1
numberOutputTriggers 2
maxTriggerDistance 4
maxTriggerTime 16
trapDensity 1

MousetrapObserverSwarm

displayFrequency 1

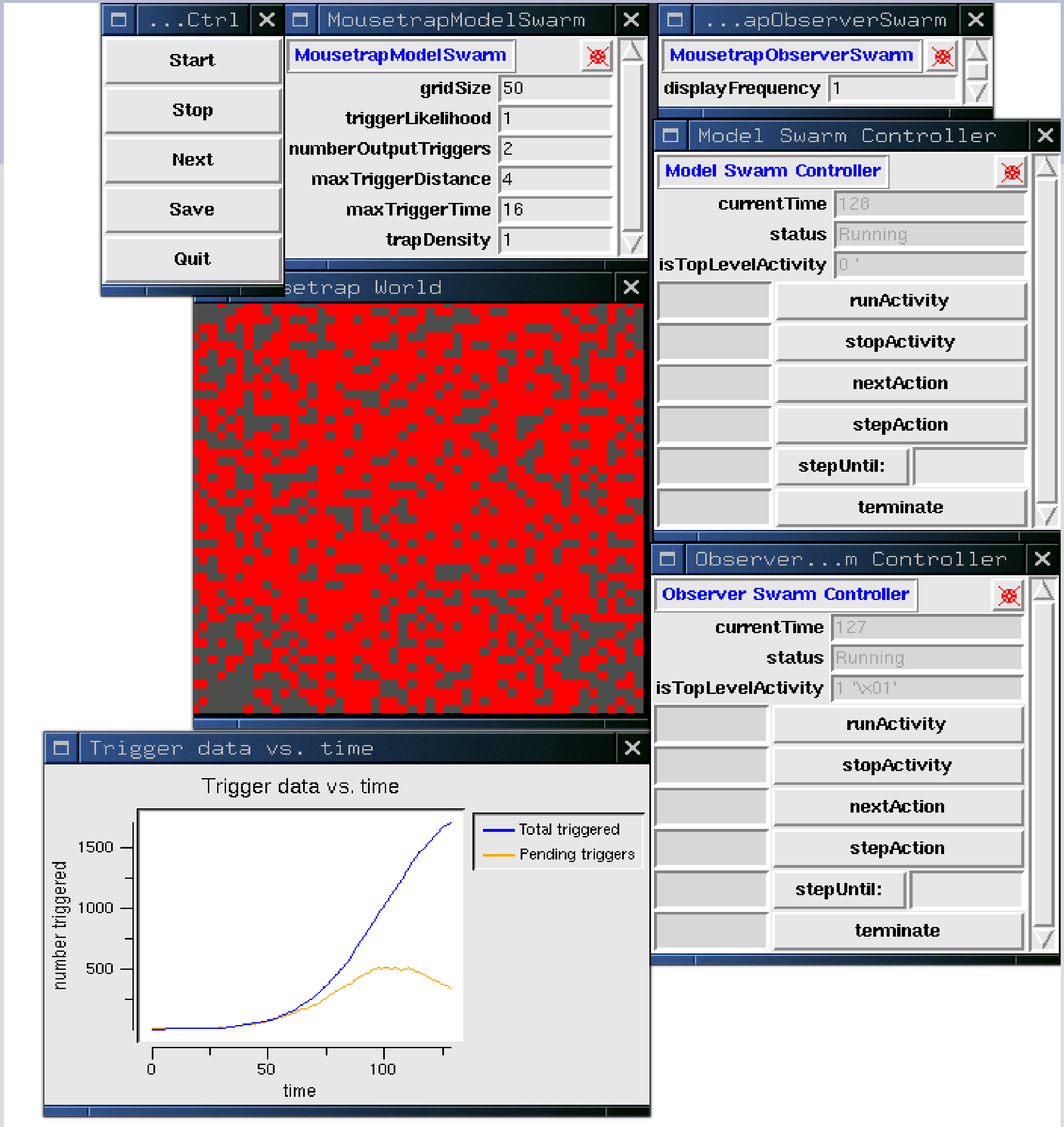
Model Swarm Controller

currentTime 43
status Running
isTopLevelActivity 0 '

Observer Swarm Controller

currentTime 42
status Running
isTopLevelActivity 1 '\x01'

Mousetrap: finished



MousetrapModelSwarm

gridSize 50
triggerLikelihood 1
numberOutputTriggers 2
maxTriggerDistance 4
maxTriggerTime 16
trapDensity 1

MousetrapObserverSwarm

displayFrequency 1

Model Swarm Controller

currentTime 128
status Running
isTopLevelActivity 0 '

Observer Swarm Controller

currentTime 127
status Running
isTopLevelActivity 1 '\x01'

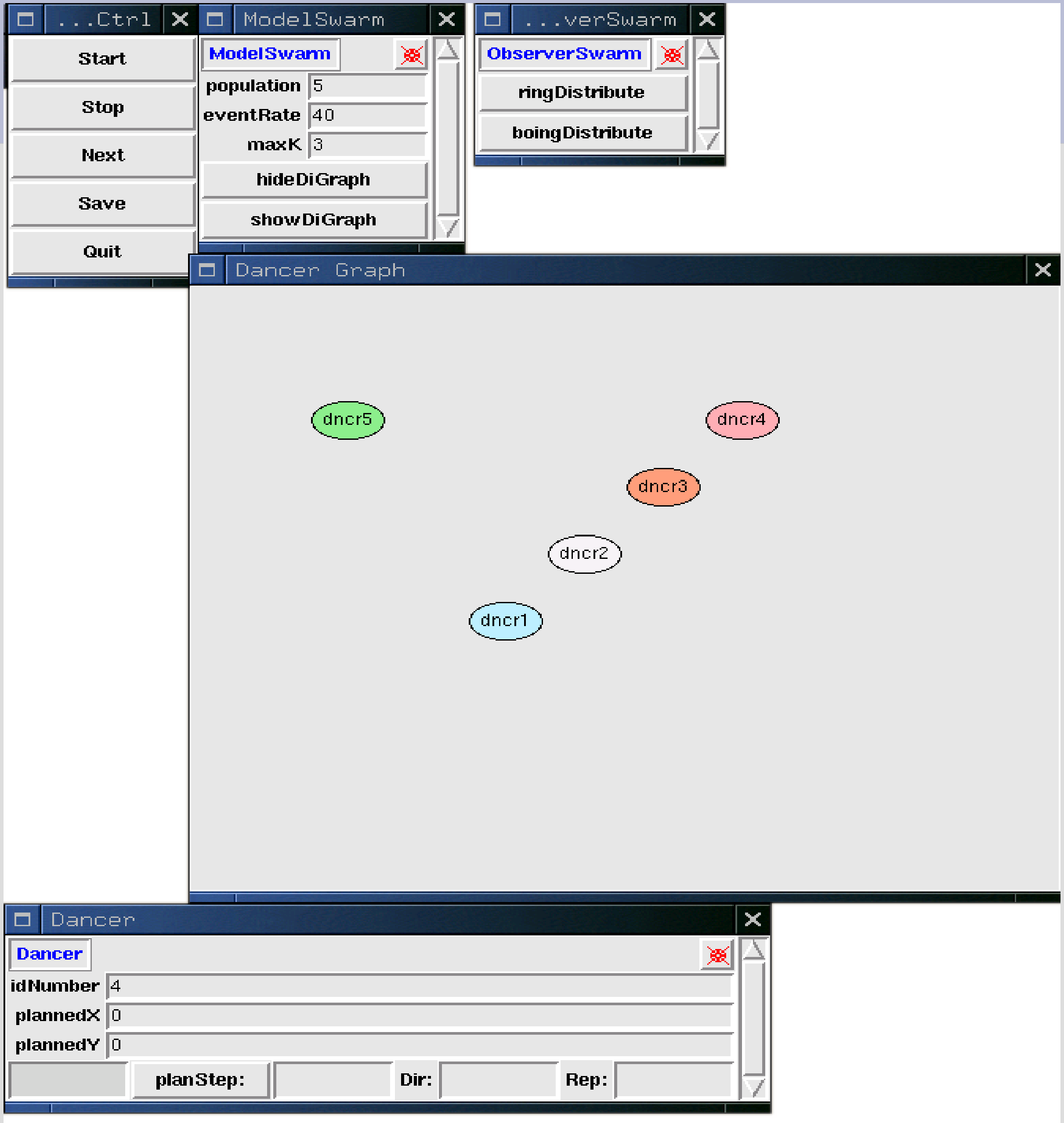
How Decentralized is it?

- Schedule in ModelSwarm manages timing
- Not completely “decentralized” in the bottom-up sense
- A true bottom-up scheduling is possible (pjrepeater* examples)
- “activateIn:” is hierarchical “time harmonization” tool

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 Steps (Behaviors) take a variable number of time steps
- Swarm model has dancers “schedule themselves” for new steps X timesteps into future (asynchronous, dynamic scheduling).

Dancer



ModelSwarm

population 5

eventRate 40

maxK 3

Dancer

idNumber 4

plannedX 0

plannedY 0

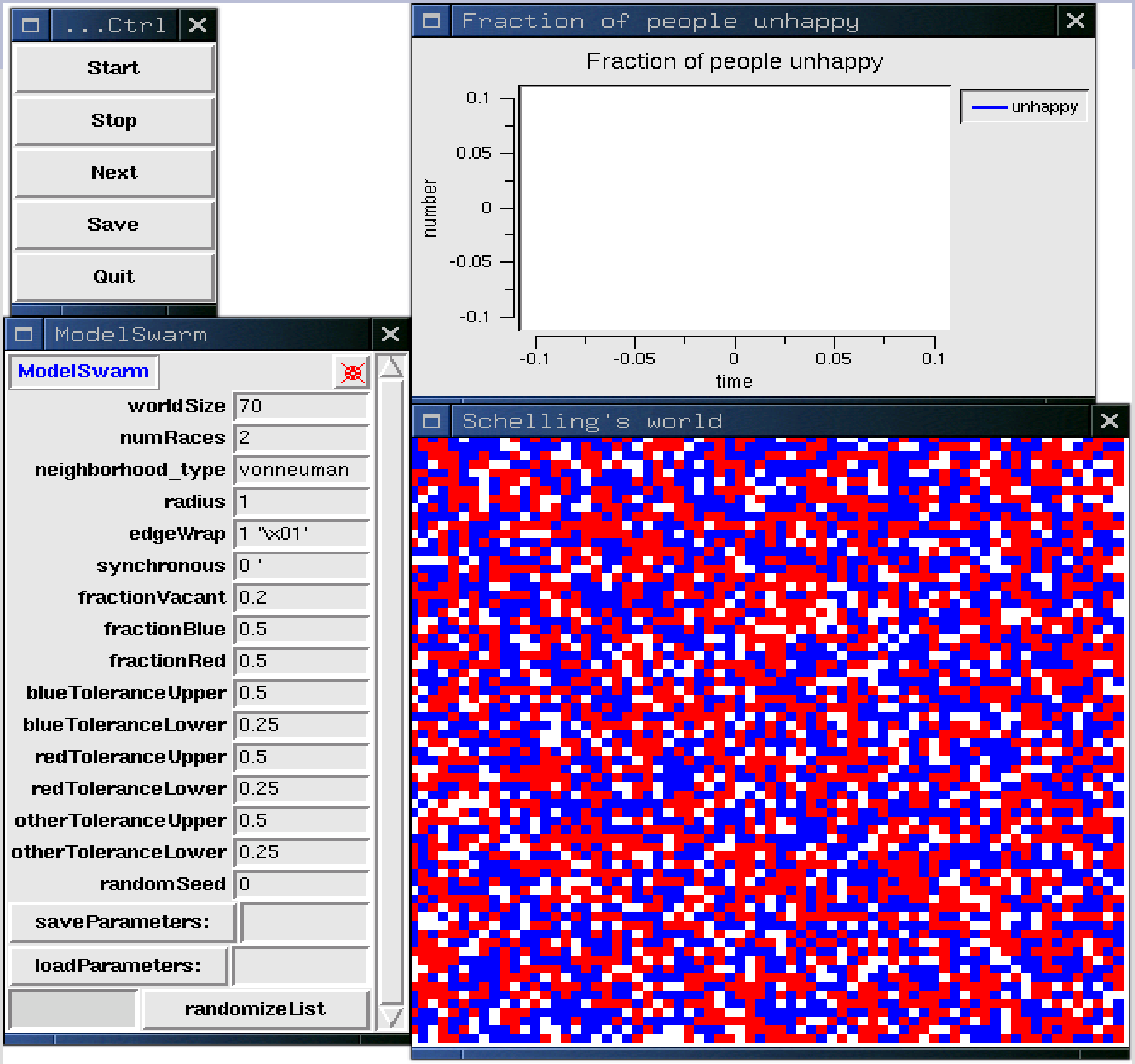
Schelling2

- Thomas Schelling, “Dynamic Models of Segregation”, *Journal of Mathematical Sociology*, 1971
- Cells are “houses”
- White cells are empty
- Agents are “colored” and move about
- Can tolerate some diversity
- move if

tolerance < diversity in neighborhood

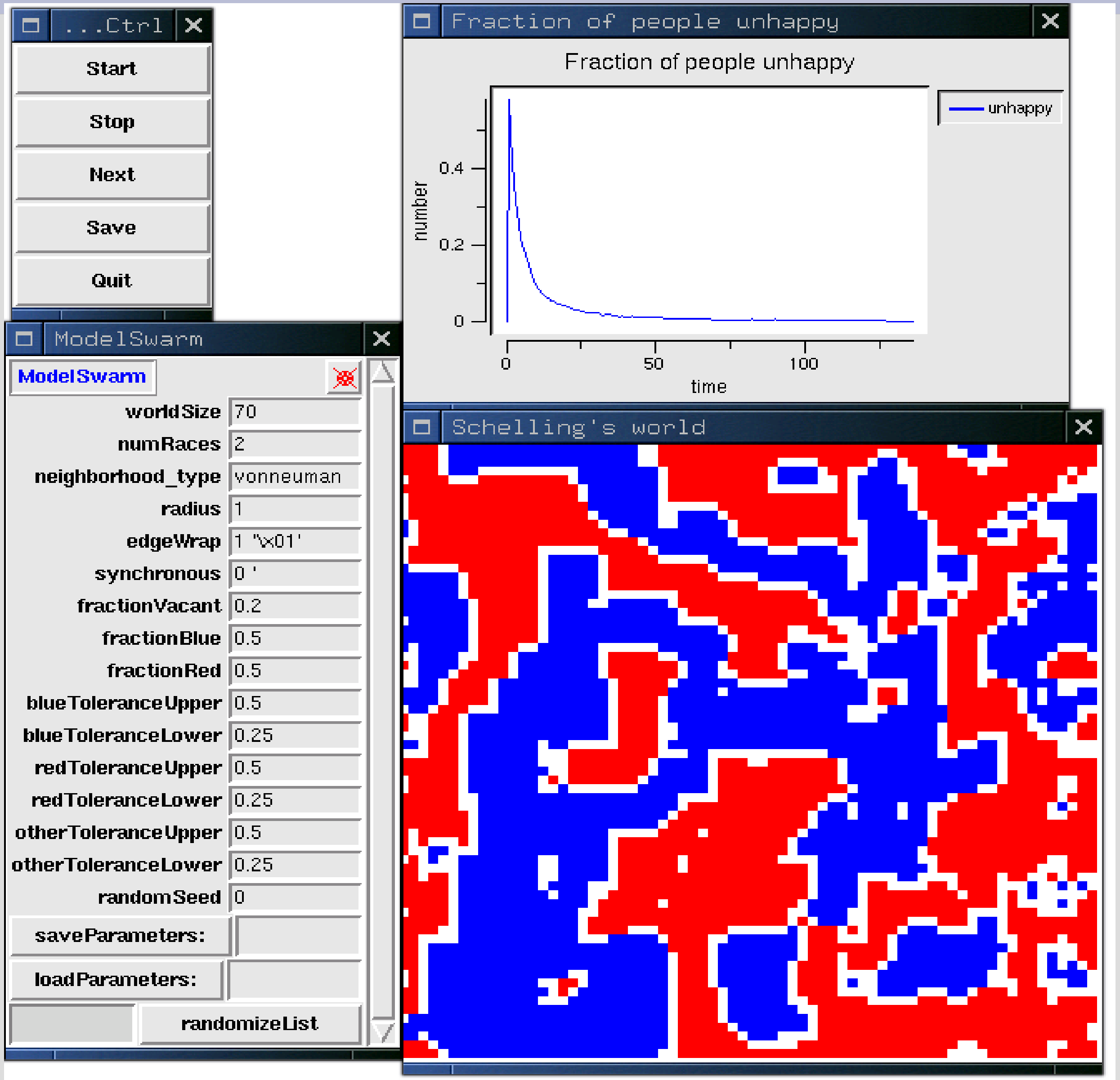
diversity = 1 - fractionOwnType

Standard Schelling Start



worldSize	70
numRaces	2
neighborhood_type	vonneuman
radius	1
edgeWrap	1 '\x01'
synchronous	0 '
fractionVacant	0.2
fractionBlue	0.5
fractionRed	0.5
blueToleranceUpper	0.5
blueToleranceLower	0.25
redToleranceUpper	0.5
redToleranceLower	0.25
otherToleranceUpper	0.5
otherToleranceLower	0.25
randomSeed	0

Standard Schelling End



worldSize	70
numRaces	2
neighborhood_type	vonneuman
radius	1
edgeWrap	1 '\x01'
synchronous	0 '
fractionVacant	0.2
fractionBlue	0.5
fractionRed	0.5
blueToleranceUpper	0.5
blueToleranceLower	0.25
redToleranceUpper	0.5
redToleranceLower	0.25
otherToleranceUpper	0.5
otherToleranceLower	0.25
randomSeed	0

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

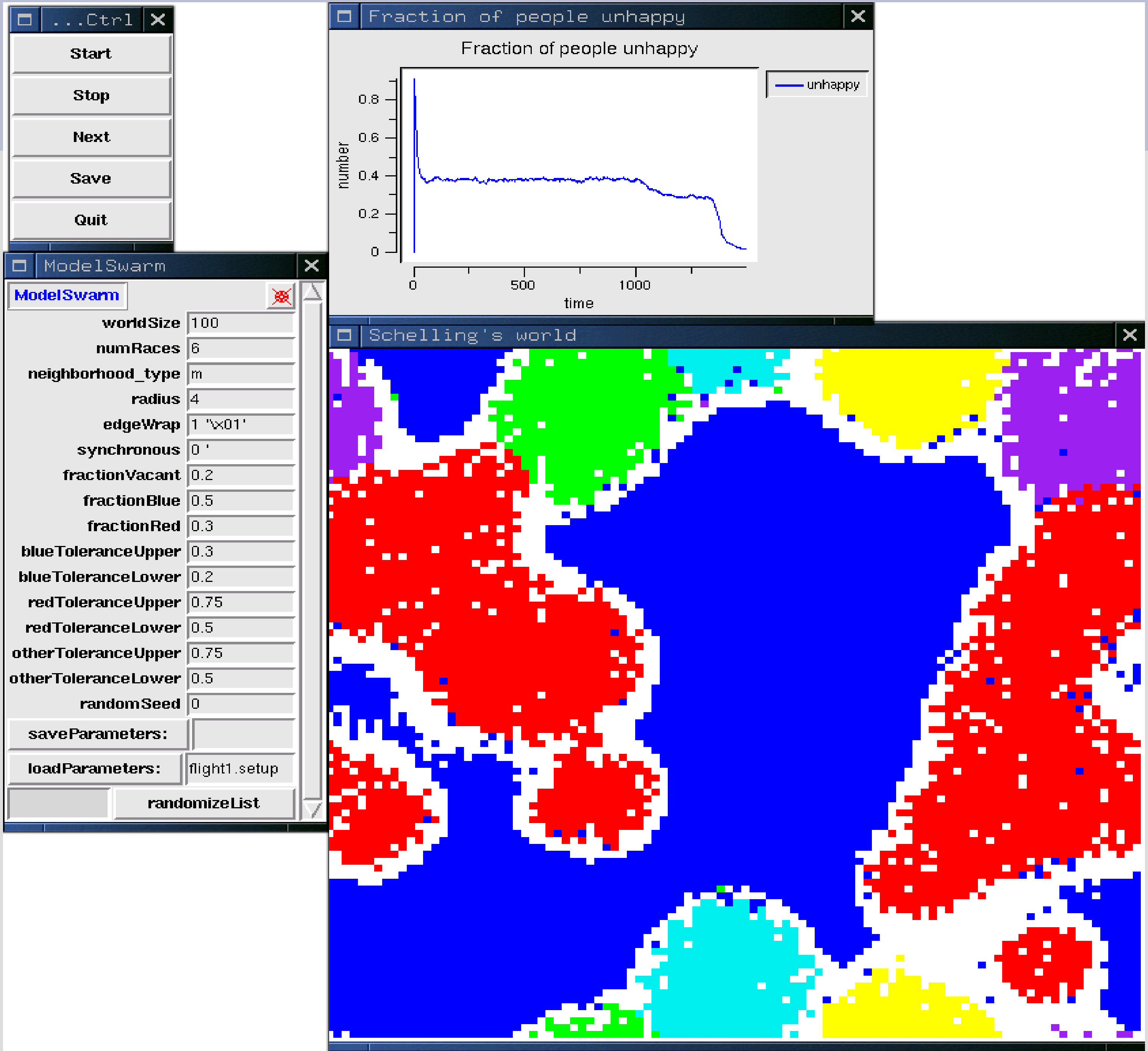
Many Options can be considered

- Number of races
- Tolerance of individuals
- Set Neighborhood type- Moore or VonNeumann
- Radius of neighborhood
- Edge effects & Wrap Around
- Randomized ordering of agent actions at each step
- ASYNCHRONOUS or ASYNCHRONOUS
-

Bells & Whistles

- Note Files:
  - Parameter file: load or save
  - Output file
- Screenshot of raster: turn on “writeGUIRaster” in the GUI, watch what happens
- Full BatchSwarm implementation, including BatchPixmap

Explore: flight1.setup



ModelSwarm

worldSize	100
numRaces	6
neighborhood_type	m
radius	4
edgeWrap	1 '\x01'
synchronous	0 '
fractionVacant	0.2
fractionBlue	0.5
fractionRed	0.3
blueToleranceUpper	0.3
blueToleranceLower	0.2
redToleranceUpper	0.75
redToleranceLower	0.5
otherToleranceUpper	0.75
otherToleranceLower	0.5
randomSeed	0
saveParameters:	
loadParameters:	flight1.setup
	randomizeList

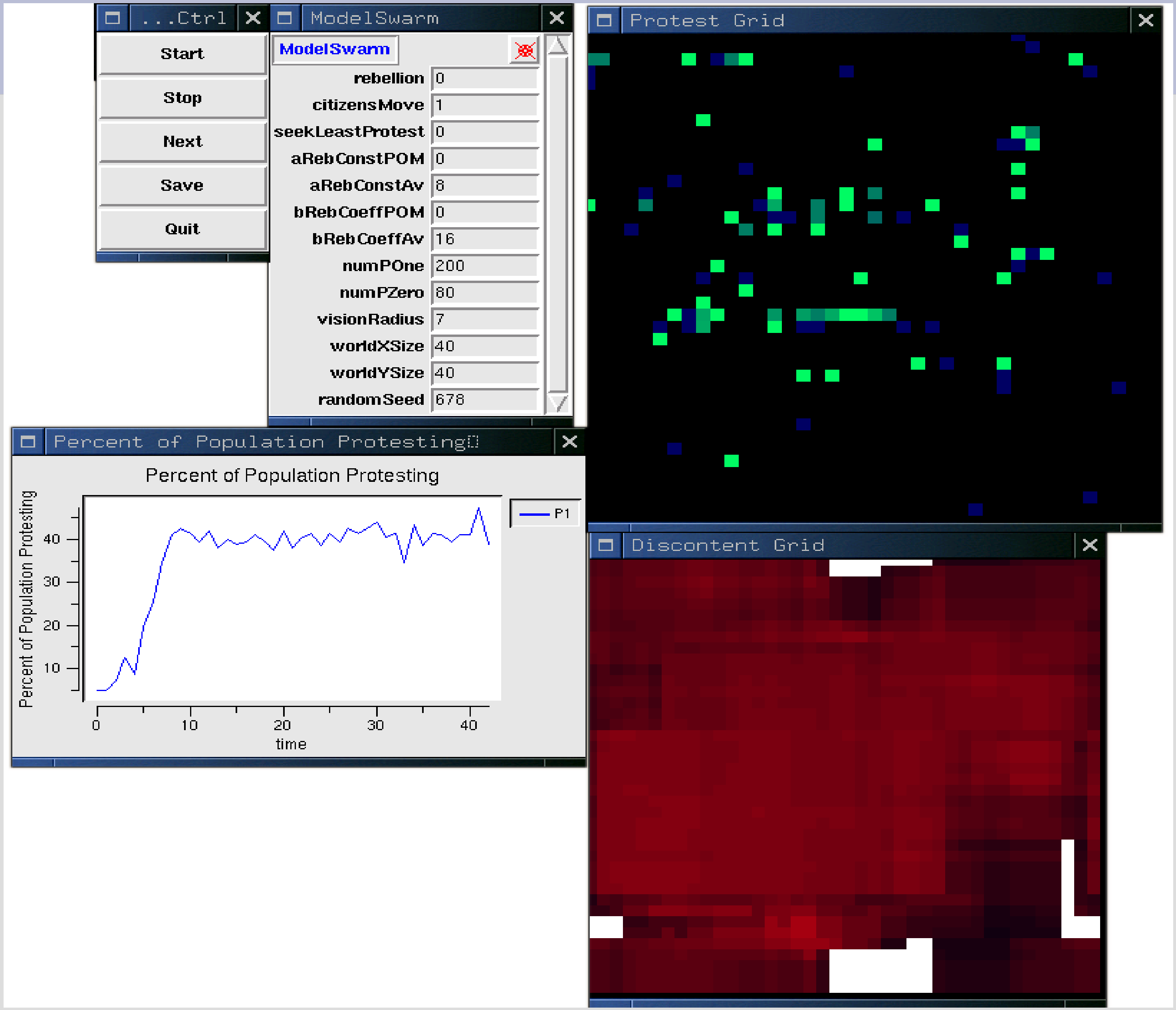
Protest Activist Model

- 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



rebellion	0
citizensMove	1
seekLeastProtest	0
aRebConstPOM	0
aRebConstAv	8
bRebCoeffPOM	0
bRebCoeffAv	16
numPOne	200
numPZero	80
visionRadius	7
worldXSize	40
worldYSize	40
randomSeed	678

Social Impact Model

- Nowak & Latane: social psychologists

A. Nowak, J. Szamrej, B. Latane. “From private attitude to public opinion: A dynamic theory of social impact” Psychological Review 97 (1990)

- A well-known cellular automaton
- Agents change YES or NO depending

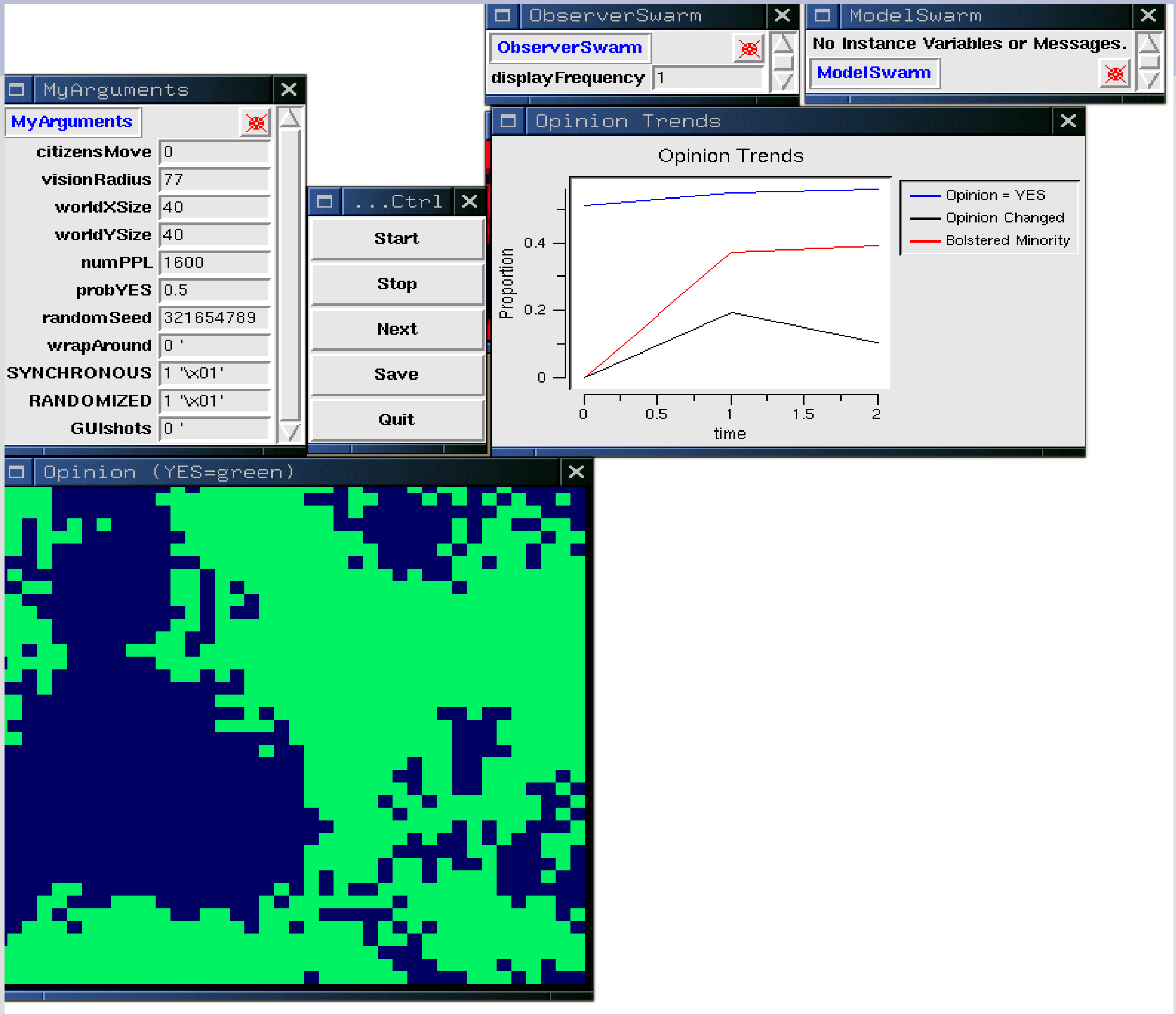
Latane's theory

- Agents change opinion YES or NO depending on social pressure
- Agents gather “support” from like-minded others
- Agents subjected to pressure from other-minded agents
- Influence is distance weighted: closer agents have more influence

Social Impact Model

- Swarm “SIM” available
- Swarm SIM model implements ASYNCHRONOUS option
- Swarm SIM implements “variable neighborhood size”

Social Impact Model



MyArguments

citizensMove	0
visionRadius	77
worldXSize	40
worldYSize	40
numPPL	1600
probYES	0.5
randomSeed	321654789
wrapAround	0 '
SYNCHRONOUS	1 '\x01'
RANDOMIZED	1 '\x01'
GUIshots	0 '

ObserverSwarm

displayFrequency	1

ModelSwarm

No Instance Variables or Messages.

Speed Note

- Heatbug style cell search TOO SLOW
- Activists, SIM, Schelling2 use “collector grids” to register the actions of agents.
- When agents “make change” they register that action withworld
- World applies impact on all cells within “eyesight”.
- Other agents can obtain “visible activity” with a single check or a Grid position.

Artificial Stock Market

- Pioneering study.

R.G. Palmer, Brian Arthur, John Holland, Blake LeBaron, & Paul Taylor, “Artificial economic life: a simple model of a stockmarket” Physica D 75: 264-274.

- Swarm project on Sourceforge

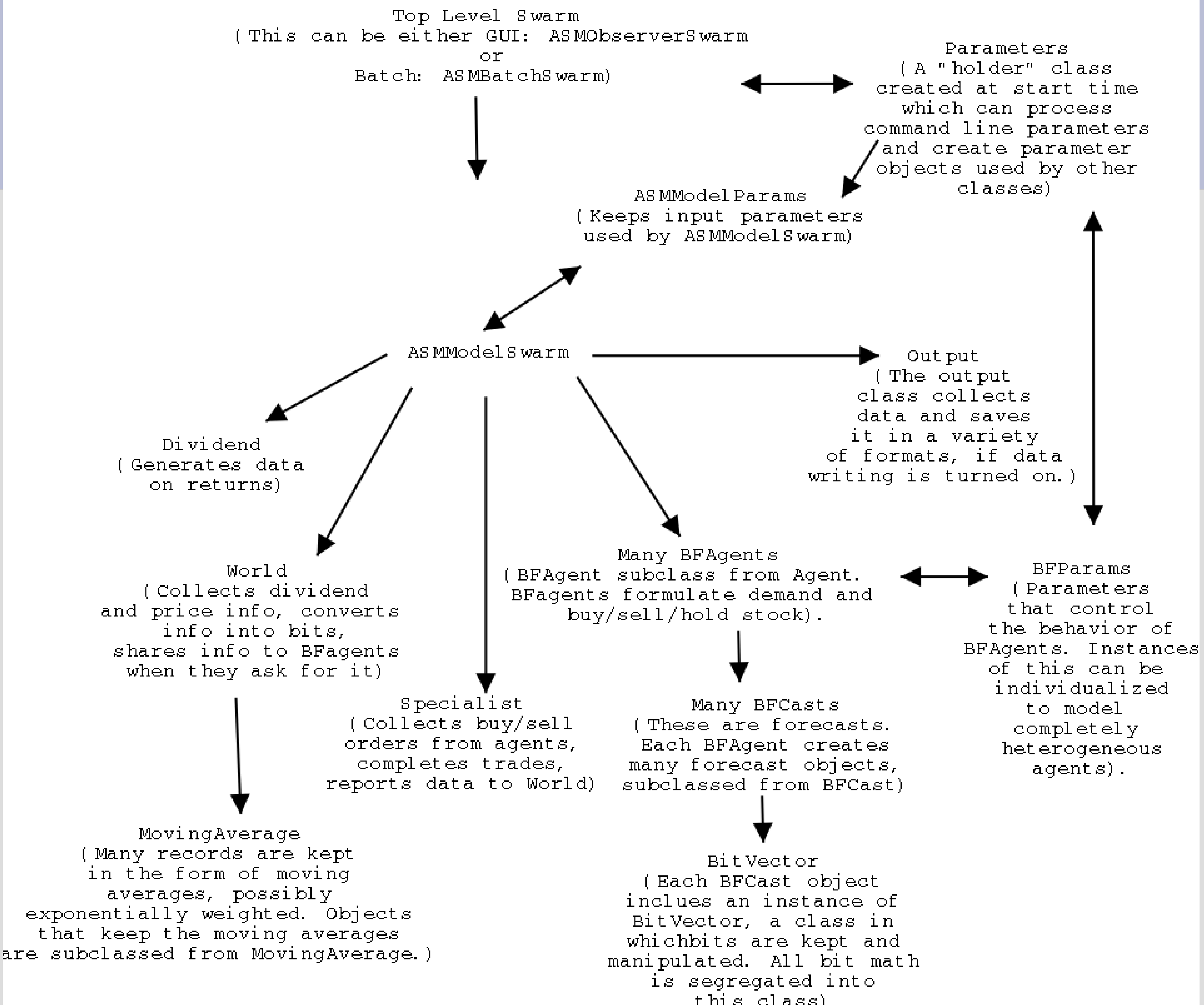
  http://ArtStkMkt.sf.net

  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).

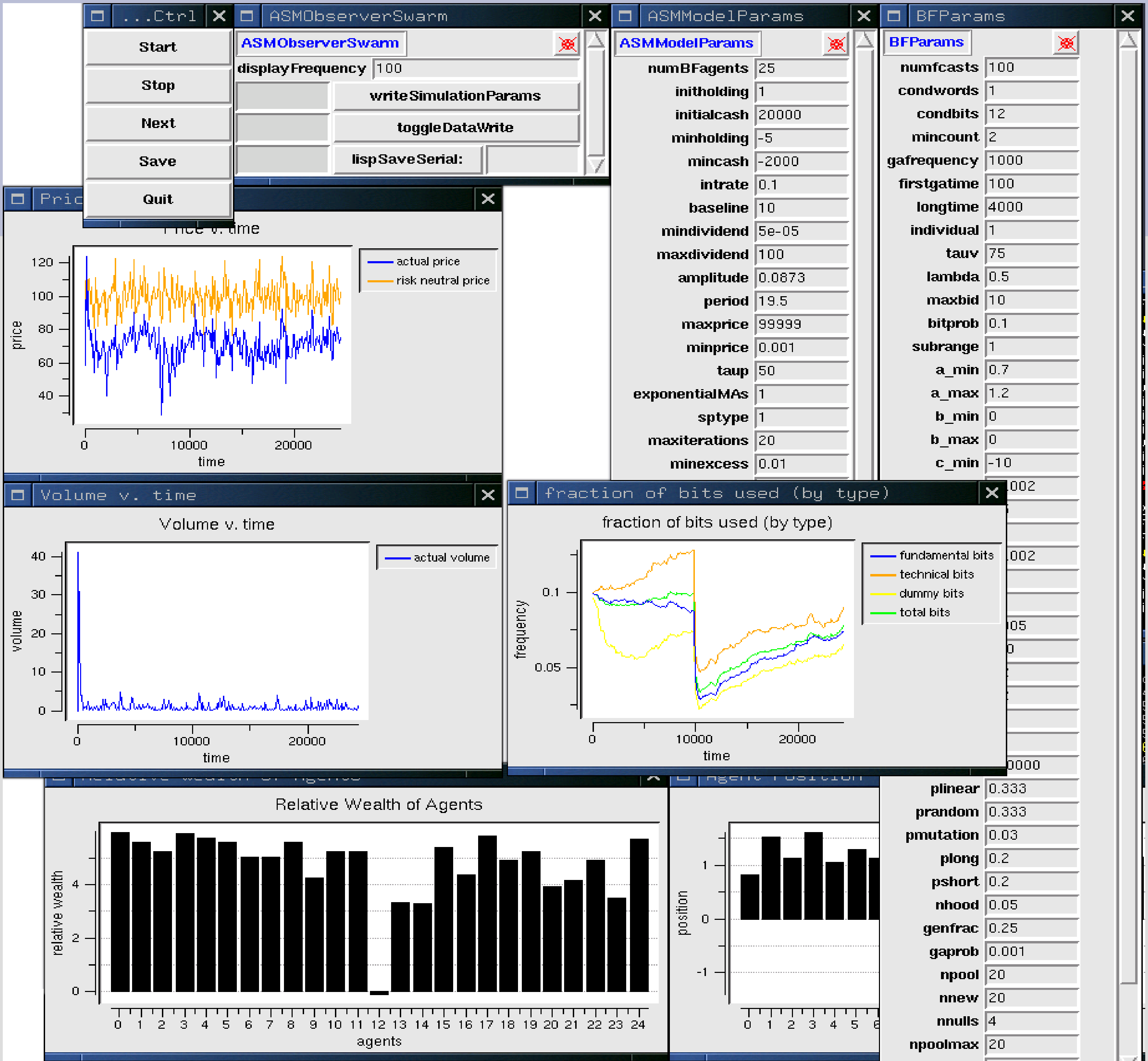
Specialist
(Collects buy/sell
orders from agents,
completes trades,
reports data to World)

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



displayFrequency 100

numBFagents	25
initholding	1
initialcash	20000
minholding	-5
mincash	-2000
intrate	0.1
baseline	10
mindividend	5e-05
maxdividend	100
amplitude	0.0873
period	19.5
maxprice	99999
minprice	0.001
taup	50
exponentialMAs	1
sptype	1
maxiterations	20
minexcess	0.01

numfcasts	100
condwords	1
condbits	12
mincount	2
gafrequency	1000
firstgatime	100
longtime	4000
individual	1
tauv	75
lambda	0.5
maxbid	10
bitprob	0.1
subrange	1
a_min	0.7
a_max	1.2
b_min	0
b_max	0
c_min	-10
plinear	0.333
prandom	0.333
pmutation	0.03
plong	0.2
pshort	0.2
nhood	0.05
genfrac	0.25
gaprob	0.001
npool	20
nnew	20
nnulls	4
npoolmax	20

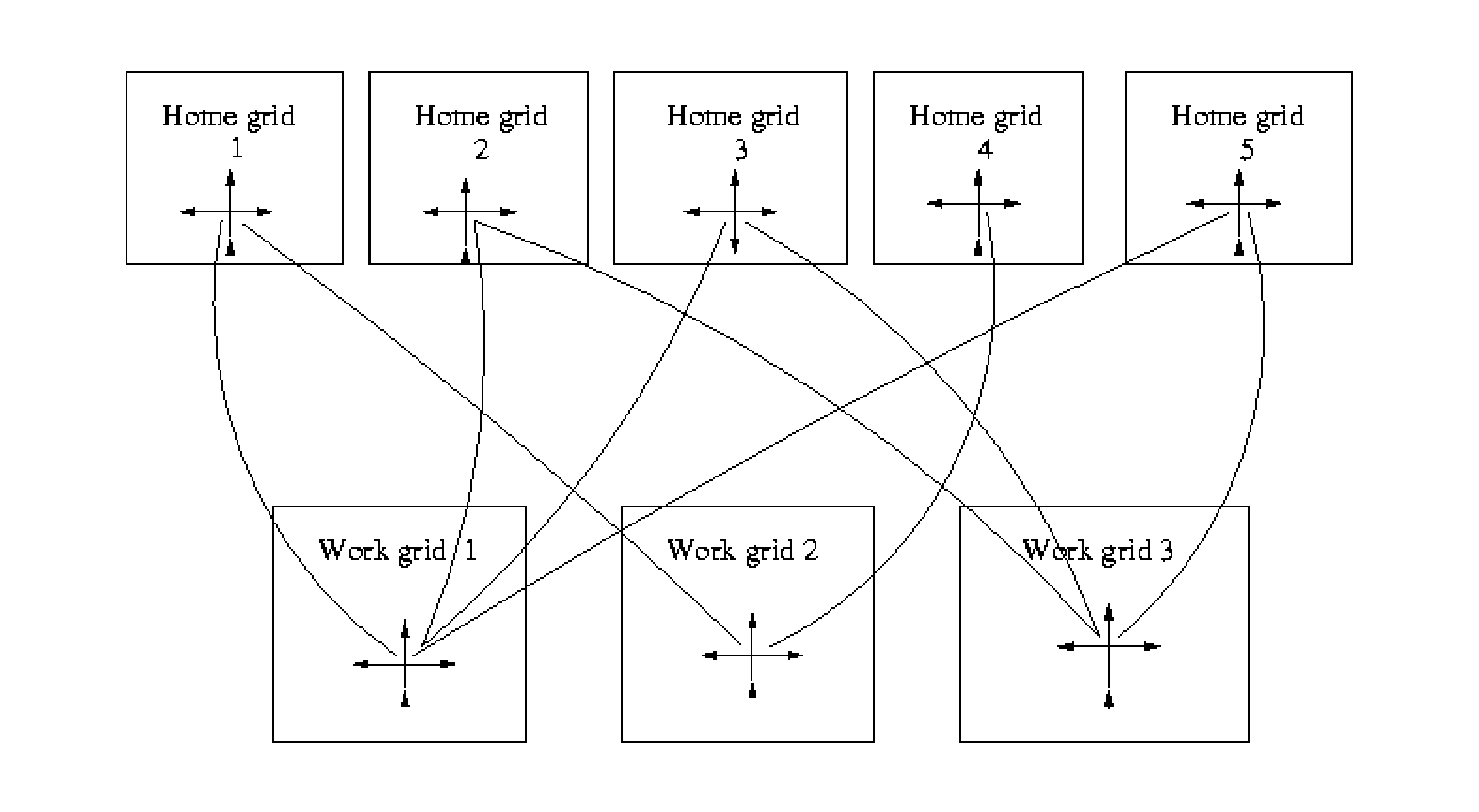
ASM: Serialization

- 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)
- Agents interact only when they
  - find another available agent and
  - choose to initiate interaction
- Various behavioral premises
- (Comparatively) complete documentation

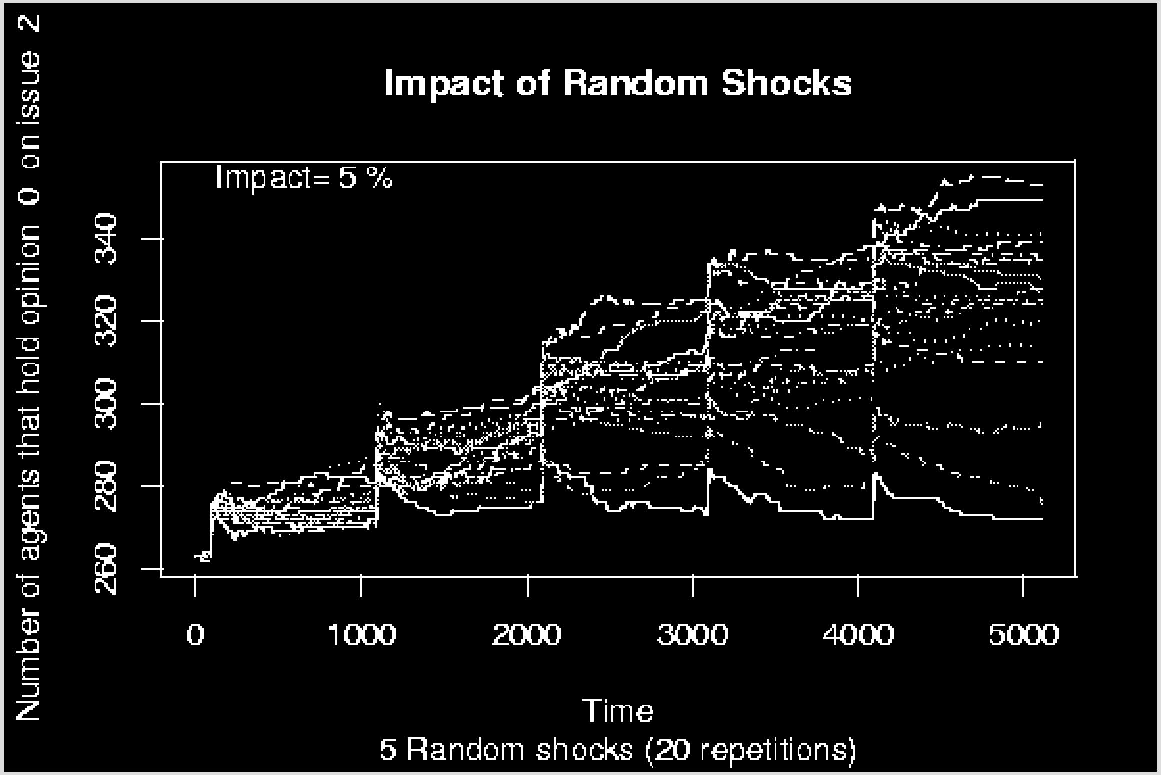
Many agents per cell allowed



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

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?