MASON

Sean Luke

Department of Computer Science
George Mason University
Washington, DC

About Myself

- Associate Professor
  Department of Computer Science
  George Mason University
- Co-director
  GMU Evolutionary Computation Laboratory
- *Interests*
  Multiagent Systems and Simulation
  Machine Learning and Stochastic Search
  Autonomous Robotics

Presentation Overview

- Quick Introduction to MASON
- Projects Involving MASON
- MASON Architecture Overview
  [note to the jet-lagged: you can sleep here]
- Model Parameter and Agent Behavior Optimization

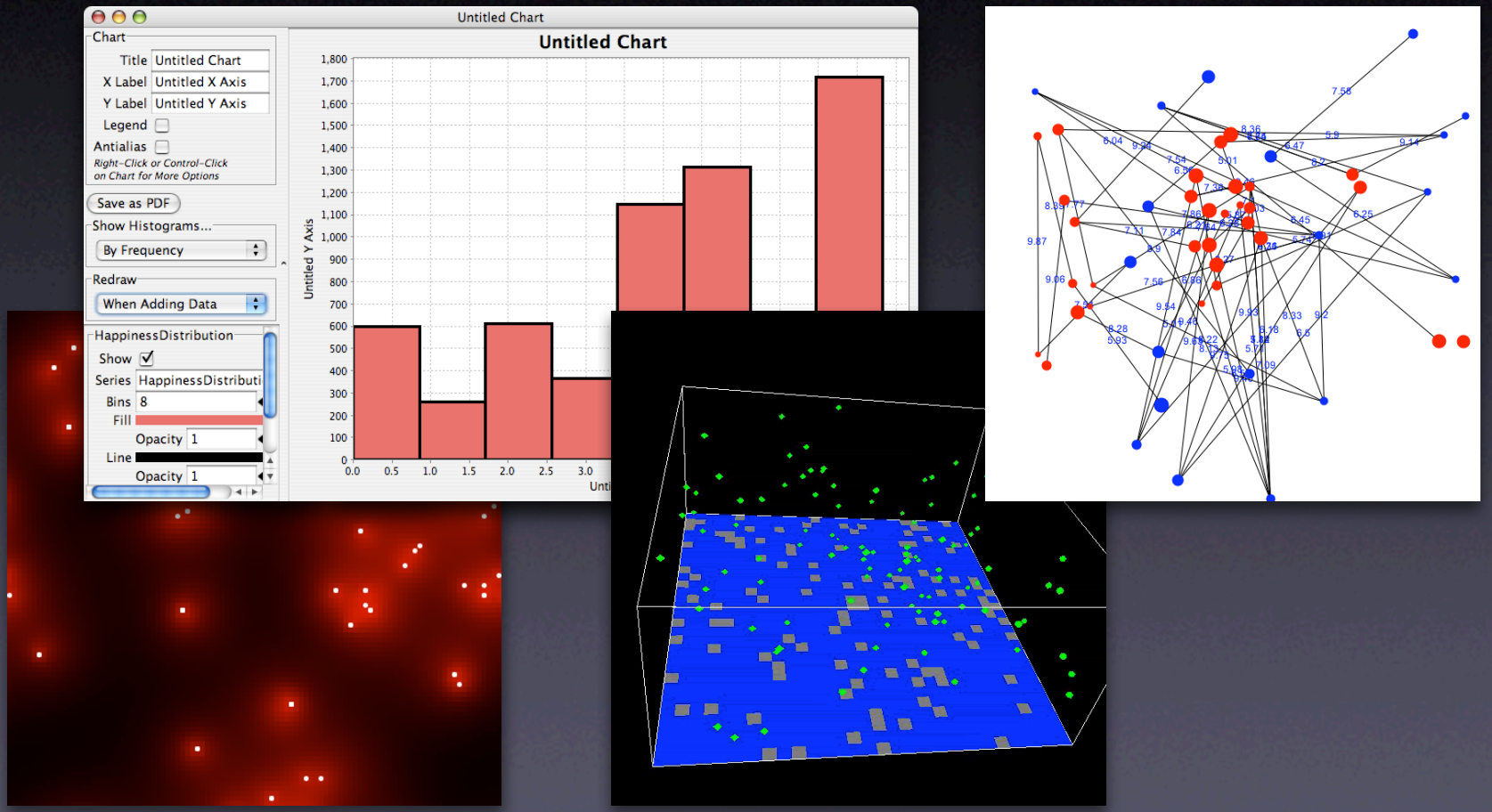
- *Later this week:* Hands-On Programming

Introduction

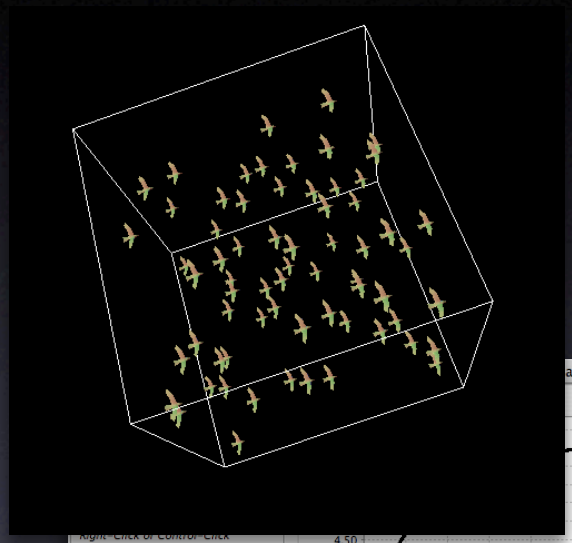
What is MASON?

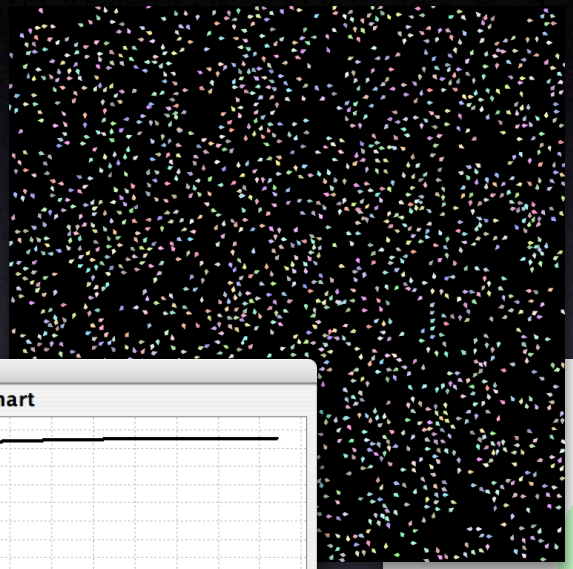
- Open source Java discrete-event simulator
- For multiagent simulations requiring:
  - speed
  - replicability
  - large numbers of agents
  - high degree of flexibility and "hackability"
  - simulations run on back-end servers

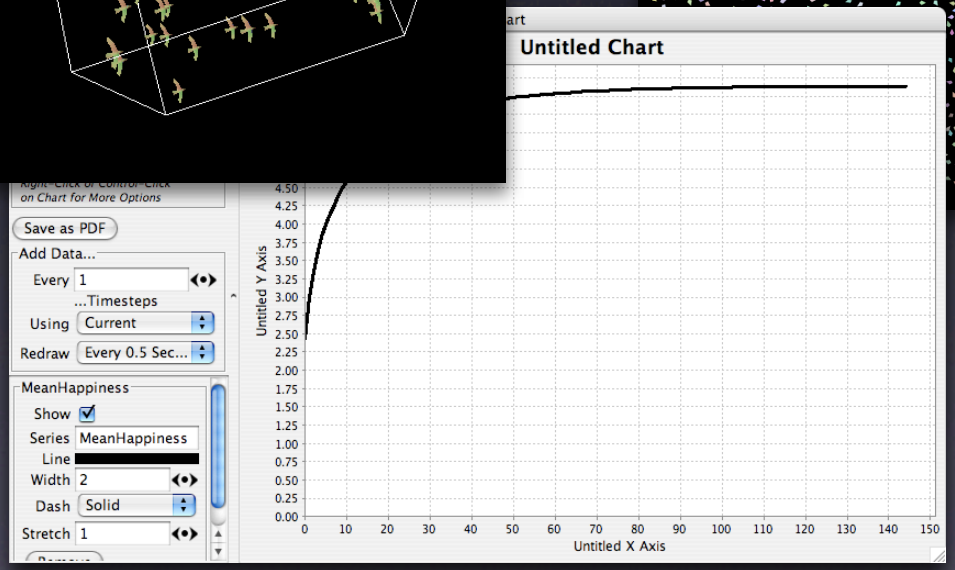
Pretty Demo Pictures

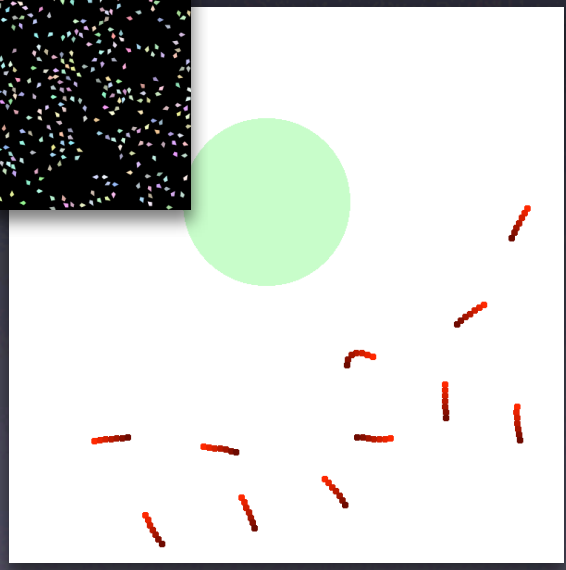


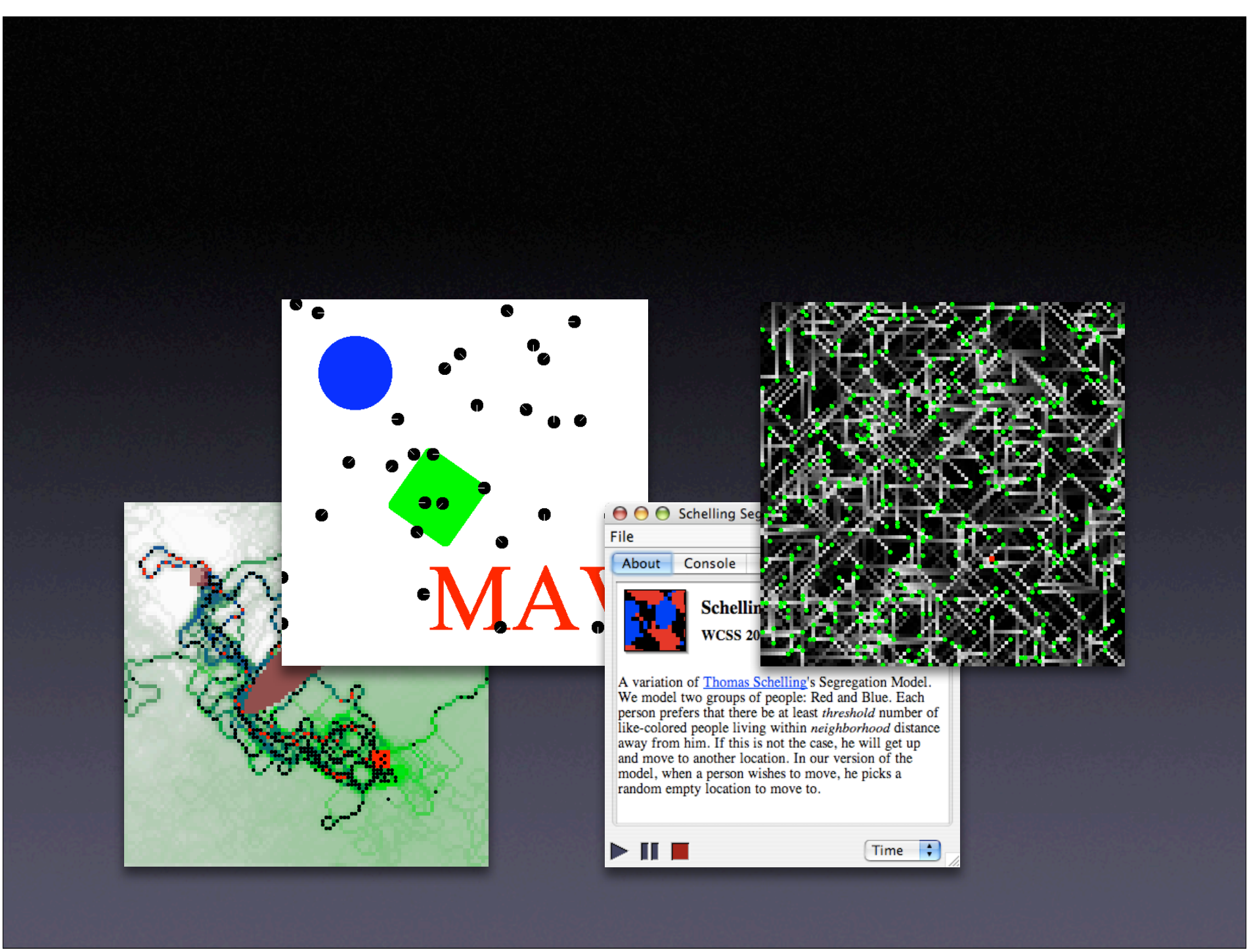
Pretty Demo Pictures










File
About
Console
A variation of Thomas Schelling's Segregation Model. We model two groups of people: Red and Blue. Each person prefers that there be at least *threshold* number of like-colored people living within *neighborhood* distance away from him. If this is not the case, he will get up and move to another location. In our version of the model, when a person wishes to move, he picks a random empty location to move to.
Time

Who Built MASON?

- *Design*
  ***The* George Mason University**
  Center for Social Complexity
  & Department of Computer Science
  Sean Luke, Claudio Cioffi-Revilla

- *Coding*
  Sean Luke, Gabriel Balan, Liviu Panait, Keith Sullivan, Joey Harrison, Sean Paus, Christian Thompson, Daniel Kuebrich, Ankur Desai

- *Funding*
  GMU, NSF, DARPA, General Electric Global Research, Office of Naval Research (ONR)

Trends in "Swarm-style" MAS Simulation

- *Small, rapid-prototyped experiments*
  Higher-level, slower, more domain-specific
  - Repast, NetLogo
- *Large experiments*
  General-purpose, high-performance, modular, highly flexible, strong guarantees, parameter optimization
  - MASON

Why "Big Iron" Matters

- *Reason #0:* our simulations are getting more detailed
- *Reason #1:* we need to demonstrate solution robustness despite parameter perturbations
  - Parameter sweeps are *expensive*
- *Reason #2:* the design space of simulations is getting nasty. Automated design help is welcome. But also expensive!

What I Use MASON For

- Fast evaluation of many simulations on large numbers (100+) of back-end clustered computers
  - Optimization of agent behaviors
  - Parameter sweeps
- Visualization on front-end workstations
- Application to a wide range of problems

Large Experiments aren't Just in Robotics

- Multiagent modeling of the emergence of empire in Inner Mongolia
  GMU Center for Social Complexity
  Smithsonian Institution (NSF)
- Agents: households, political entities
- Agent interactions: trade routes, communication, migration, rise of horse culture
- We only know part of the design space!

We Use MASON for...

- Social Science MAS Models
- Physical Systems Modeling
- Robotics
- Machine Learning and Optimization
- Artificial Life and Systems Biology
- Traffic Engineering

How They're Different

- Most are *engineering* fields. They want to figure out MAS *solutions to problems*.
  - "How do I get a swarm of unmanned aerial vehicles to photograph as much territory in Afghanistan as possible?"
- MAS in the social sciences (and biological sciences) wants to use MAS to *describe phenomena*.
- It turns out the tools aren't that different!

Current Simulations Done with MASON

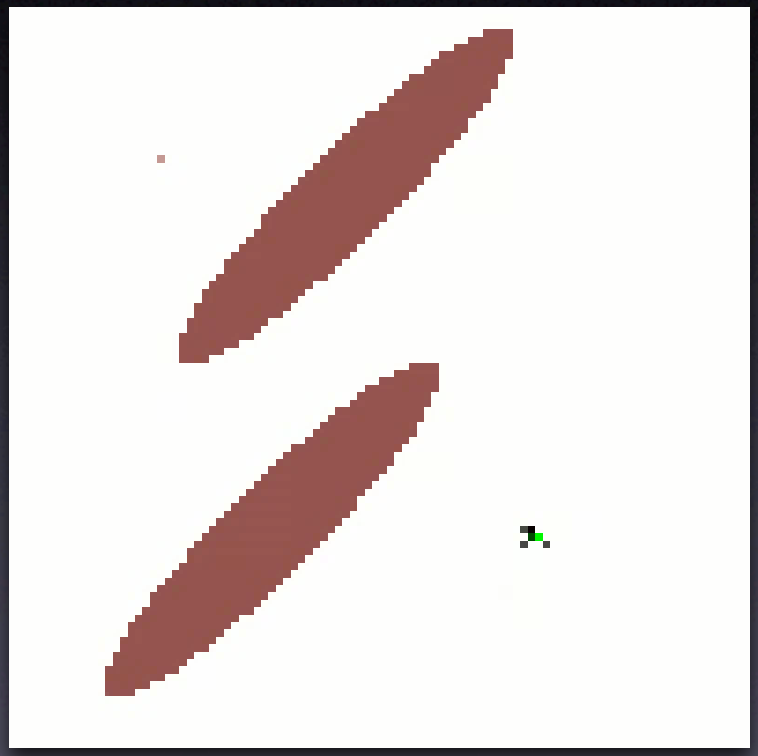
- Harbor Defense
- Cultural Transmission, Memory, Leadership, and Collective Action
- Large-Scale (million-agent) economies of microeconomically-motivated agents
- Models of the Development of the Mongolian Empire

A Few of My Own Ongoing Projects

- New Multi-Pheromone Algorithms for Ant-Robot Foraging
- Cooperative Target Observation with Unmanned Aerial Vehicles
- Multiagent Traffic Control
- Robotics Simulation
- Rigid Body Physics Simulation

Swarm Foraging with Pheromones

- Thousands of robotic ant agents discover and ferry "food" back to a nest
- Ants communicate information via depositing *pheromones*
- Pheromones direct other ants to food and nest locations



Pheromones and Value Functions

- Strong relationship with dynamic-programing-style value functions:

$$U_p(s') = R(s') + \gamma \max_{a \in A} \sum_{s'' \in S''} T(s', a, s'') U_p(s'')$$

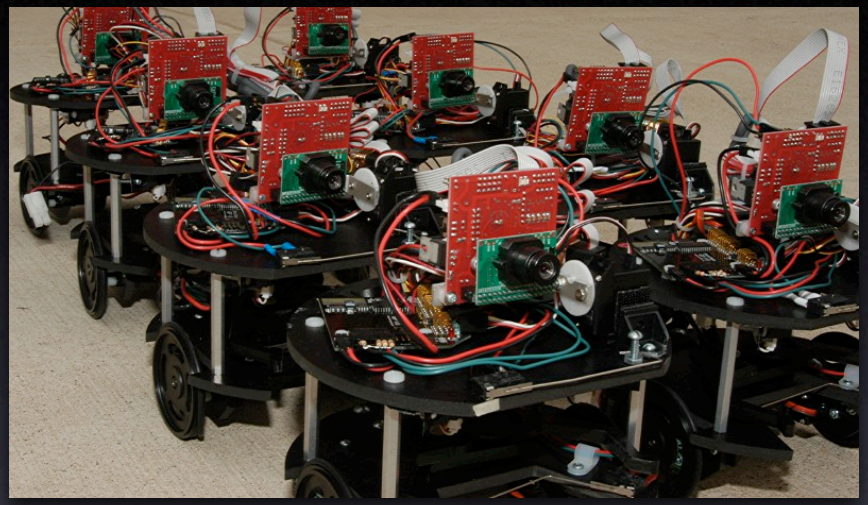
$$U_p(s') = \max(U_p(s'), (1-\alpha)U_p(s') + \alpha(R(s') + \gamma U_p(s))$$

- Of strong interest to the MAS machine learning community

Multiagent Traffic Light Control

- You've waited at many reds today. Shouldn't future lights cut you a break?
- Lights dispense *credits* to cars waiting at red
- Cars given the green must pay back credits
- Green lights given to lanes holding cars totaling the most credits
- Multiagent simulation: as many as 16,000 cars, 10x10 grid of traffic light intersections

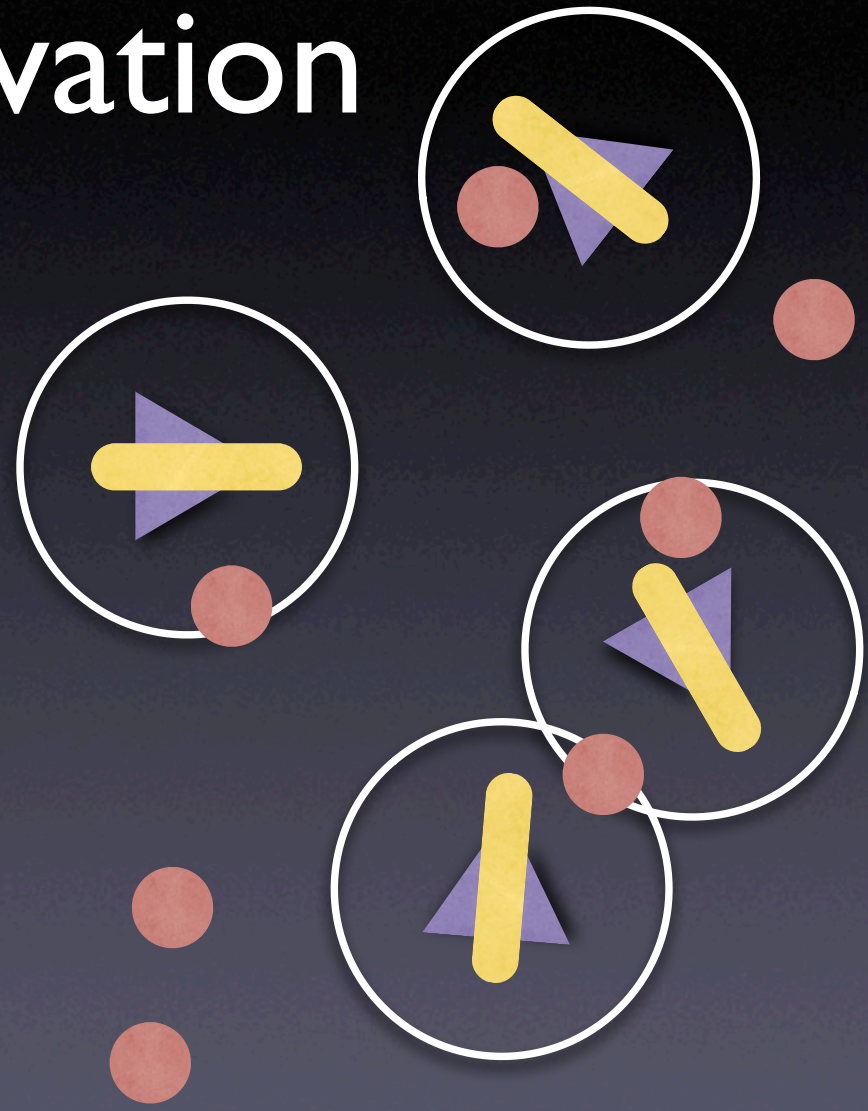
FlockBots



- 7” mobile swarm robots w/gripper, camera, wireless linux, sensors
- Running experiments on 10–1000 physical robots is slow and tedious. Fast simulation is crucial.

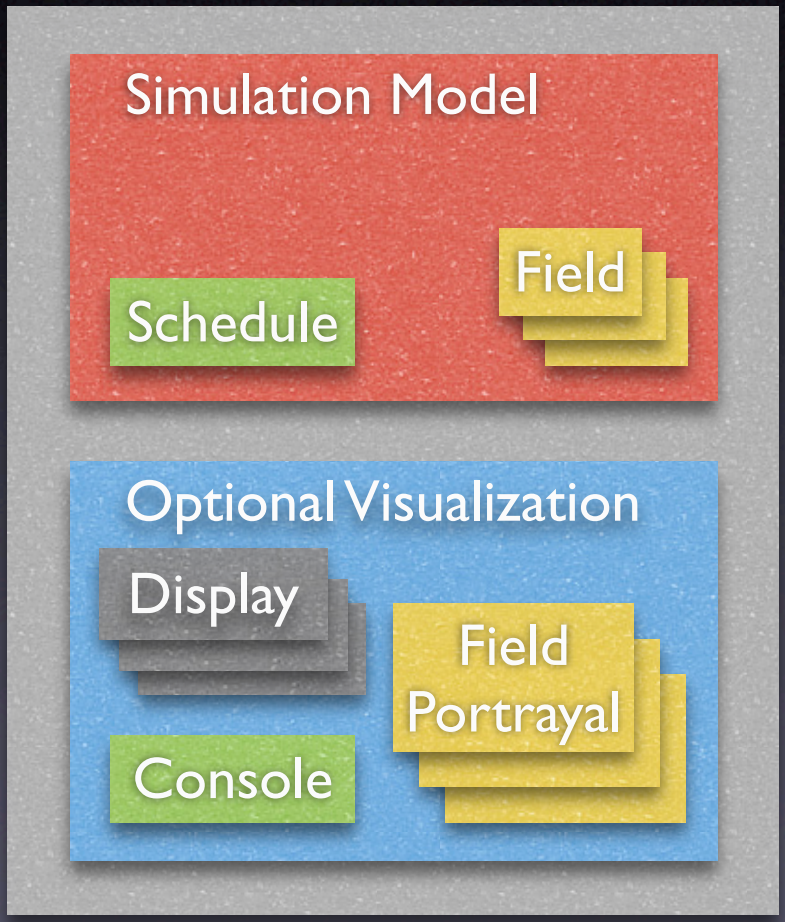
Cooperative Target Observation

- Many slow targets
- A few fast observers, each with an *observation range*
- *Partially-decentralized* algorithms to maximize the number of targets within range of at least one observer



MASON App Architecture

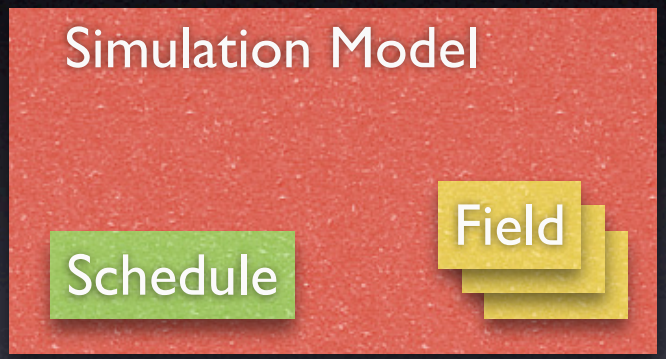
- MASON simulations are divided into two parts:
  - The *model*, which can run by itself on the command line, or be attached to...
  - The *optional visualization toolkit*, for analyzing and manipulating the model.



Architecture Overview

MASON Models

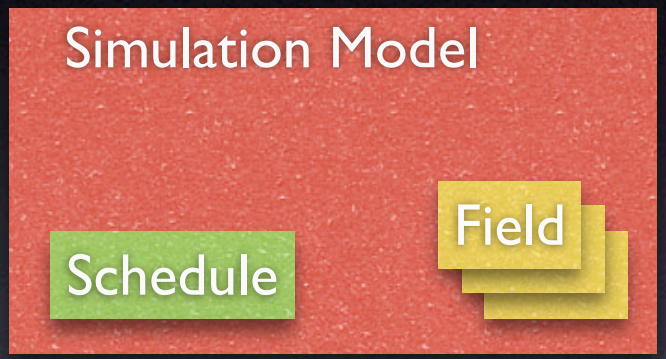
- Models often consist of a *schedule* representing time, and *fields* representing space.
- ***Schedule:*** discrete-event schedule such as is found in Swarm or Repast.
- ***Fields:*** MASON has square and hex grids, continuous space, and networks; in 2D, 3D, toroidal, bounded, or unbounded space; holding Objects, ints, or doubles.



You can make your own fields if you like.

Where are the Agents?

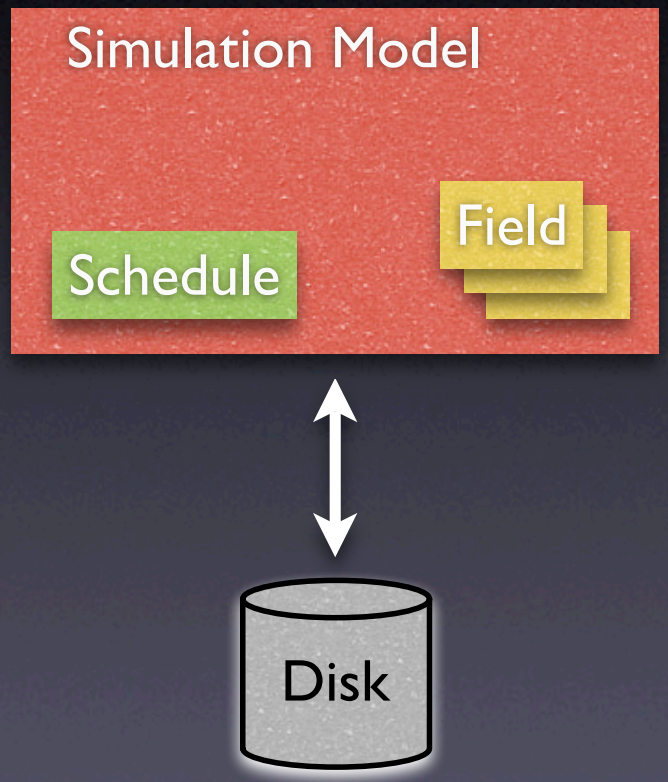
- Agents are the objects scheduled on the Schedule to manipulate the world.
- Fields can store any arbitrary object or data value: they don't just have to store agents.
- Indeed, many agents might not be part of a Field at all. They're "in the world but not of the world".



This is a very AI way of looking at things...

Checkpointing

- The entire model is self-contained in a subclass of *SimState* which you create.
- This allows you to occasionally *checkpoint* the model (save its current running state to disk).
- You can restore from a checkpoint on a different computer running a different operating system if you like.

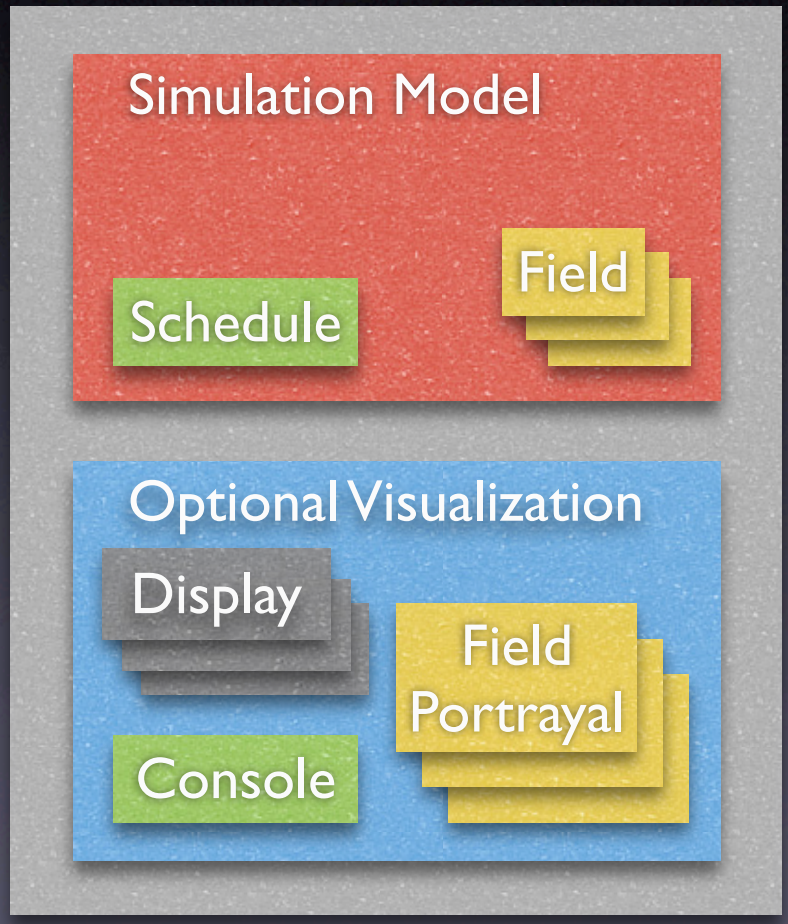


MASON Model Goals

- Efficiency
- Optional Model Replicablity
- Self-Containment
- Modularity and Extensibility
- [and of course] Checkpointing

Visualization and GUI

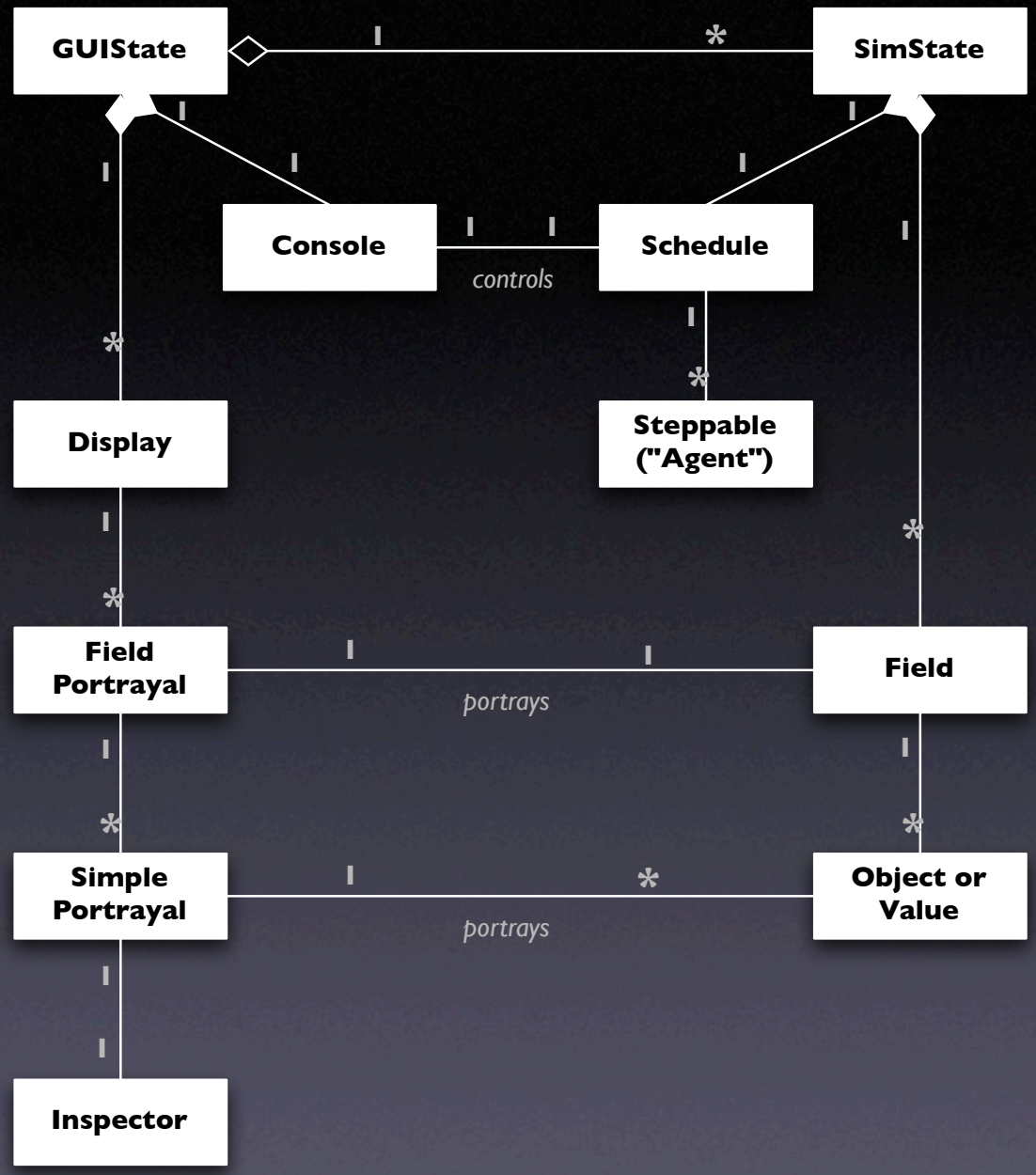
- An optional visualization toolkit holds onto the model and visualizes it while it is running.
- *The Console*: lets the user manipulate the Schedule
- *Field Portrayals:* let the user visualize / probe fields and their objects.
- *Displays:* windows which hold field portrayals



Displays and Portrayals

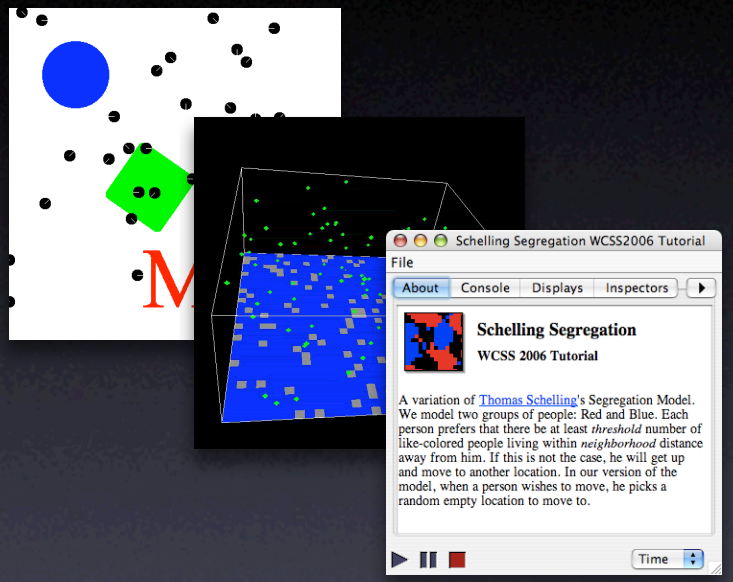
- Each 2D or 3D *display* is a window which is associated with one or more *field portrayals*
- Each field portrayal draws/inspects one *field*
- Field portrayals draw/inspect the objects or values the field by calling up *simple portrayals* you've registered for the objects
- Simple portrayals can draw objects or values, and produce *inspectors* (probes) for those objects or values.

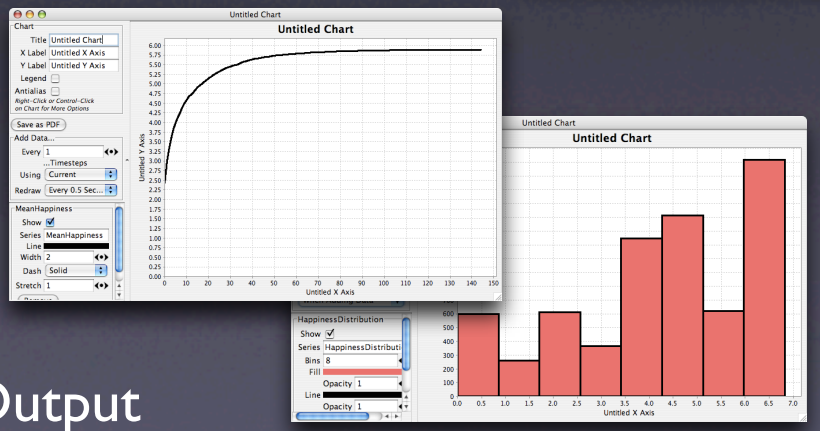
Requisite UML Diagram



Visualization Gizmos

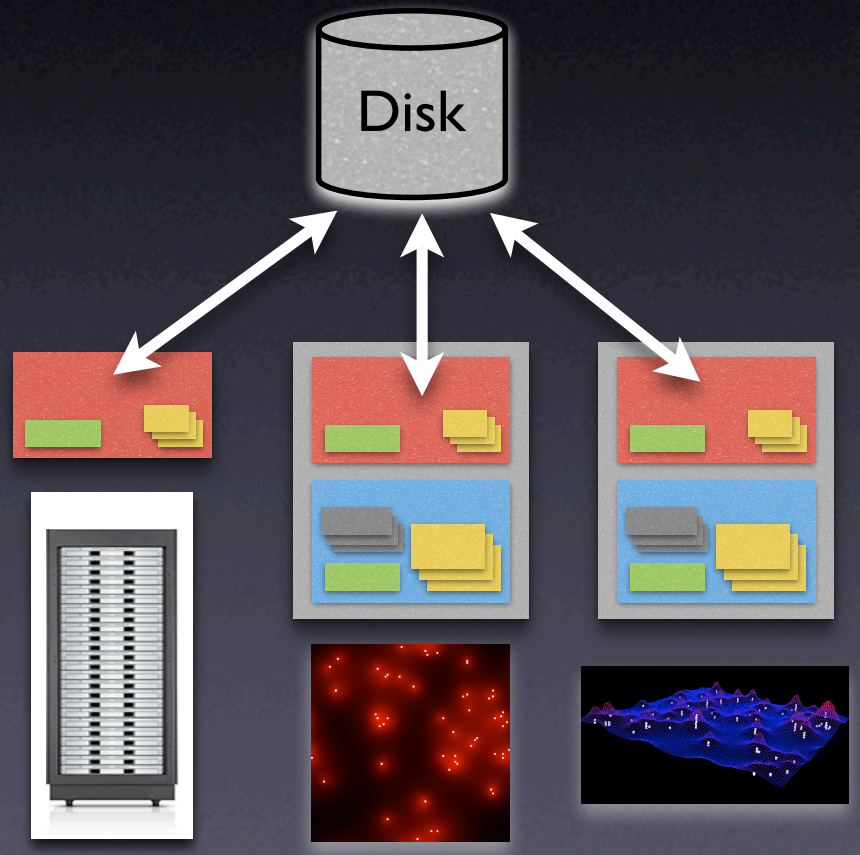
- 2D and 3D Space
- Movie and Image Generation
- Inspectors
  (What Swarm called "Probes")
- Sophisticated Console
- Histograms
- Time-Series Charts
- Publication-Quality PDF Output





Checkpointing Works in Visualization Too

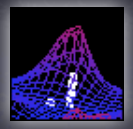
- You can checkpoint the model from, and recover to, a visualization toolkit
- The same checkpointed model runs by itself or under different visualizers.
- You can change platforms: run on a back-end server, occasionally visualize on front-end workstation.



Extensions

- 2D Rigid Body Physics (pure Java)
- Social Networks
- Scheme / Kawa
- *In Design*
  - 3D Rigid Body Physics (using ODE)
  - GIS
  - Swarmbot API

Optimization

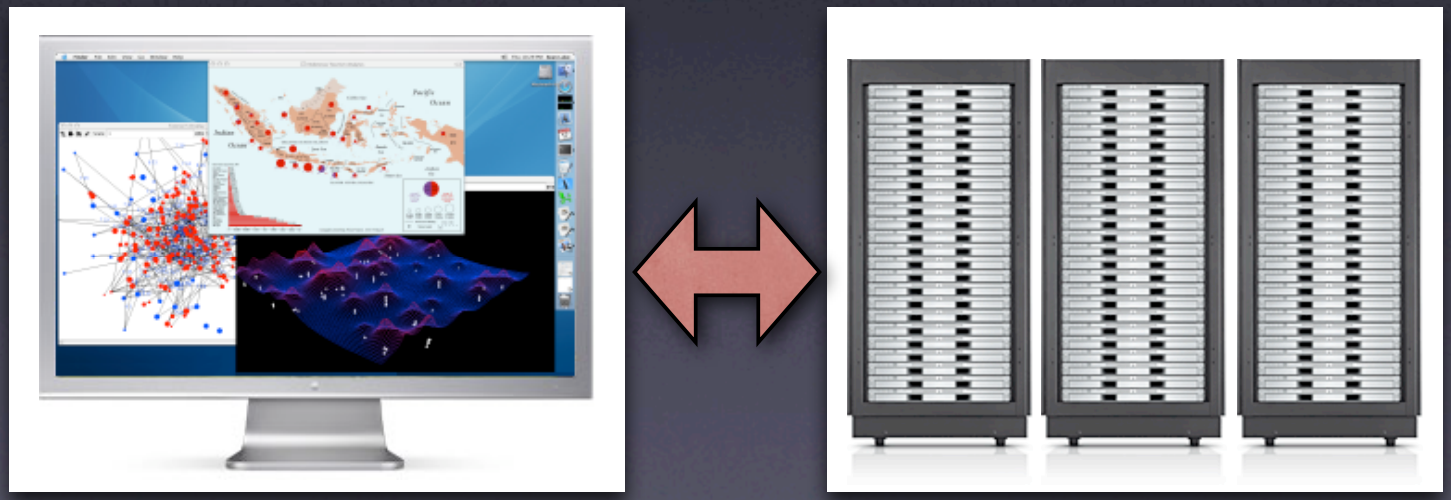


Multiagent Systems Design is Hard

- Design space increases dramatically with...
  - More agents and their interactions
  - Complex behaviors
- Unexpected macro-phenomena *emerge* from simple underlying agent behaviors
  - How to produce the effect you want?

Can Computers Help?

- It'd be nice to automate the design process!
- Let the computer discover and optimize the solution for you



Which Automation Method?

- Solutions are unknown [supervised learning]
- No directly computable gradient [gradient descent]
- Solutions take arbitrary form, neighborhood function is ad-hoc [reinforcement learning, linear optimization, etc.]
- Many local optima (due in part to "emergence") [hill-climbing]

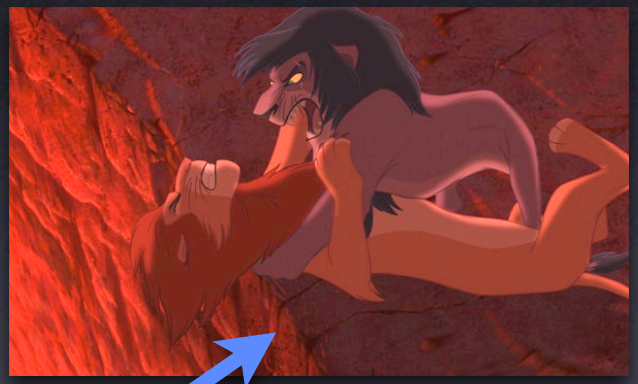
Evolutionary Computation

- When you don't know what the best solution is, but you *know a good one when you see it*

- Set of candidate solutions (a *population* of *individuals*)
- Each individual is tested and assigned a quality (*fitness*)
- A new set (the next *generation*) is formed by
  - *Selecting* and copying fitter individuals, then
  - Randomly tweaking them and mixing and matching pieces of them (*breeding*)
- Repeat

Evolutionary Computation

Selection and
Breeding



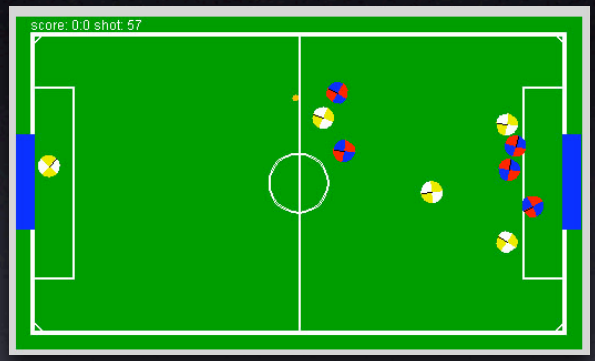
Fitness
Assessment

Population
Reassembly

MASON
Fits Here

Example: Robot Soccer Team Behaviors

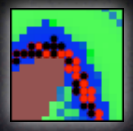
- What is are the optimal robot behaviors for good team performance?
- Each population is responsible for one robot type (goalie, forward, etc.)
- Individuals' fitnesses are assessed by teaming them up and playing a match against another team



The Good and Bad

- Evolutionary Computation methods...
  - Are highly parallelizable
  - Are representation-free
  - May be applied to many problems
- But require **many, *many*** evaluations
  - 100,000 simulation runs not uncommon
  - In this field, industrial grade *really matters!*

The End



- http://cs.gmu.edu/~eclab/projects/mason/
- *WCSS Tutorial:* Simple Schelling Segregation
  - Ask me for tutorial documentation, tutorial classes, copies of MASON, etc. (on my USB drives)