# Planning and Designing Agent-Based Models

Aaron L Bramson

# Philosophy for Complex Systems

- The goals of complex system models are conceptually distinct from other modeling methods and application domains.

- There is a focus on bottoms-up approaches, generative science, emergent properties, wholistic views, systems thinking, and systems being more than the sum of their parts.

- But what does it all mean? These terms and ideas are deployed without any rigorous treatment or specific definitions.

- By examining the landscape of the related ideas we can build a more solid foundation for our complexity framework.

# Glossary of Terms

"You keep using that word.
I don't think it means
what you think it means."

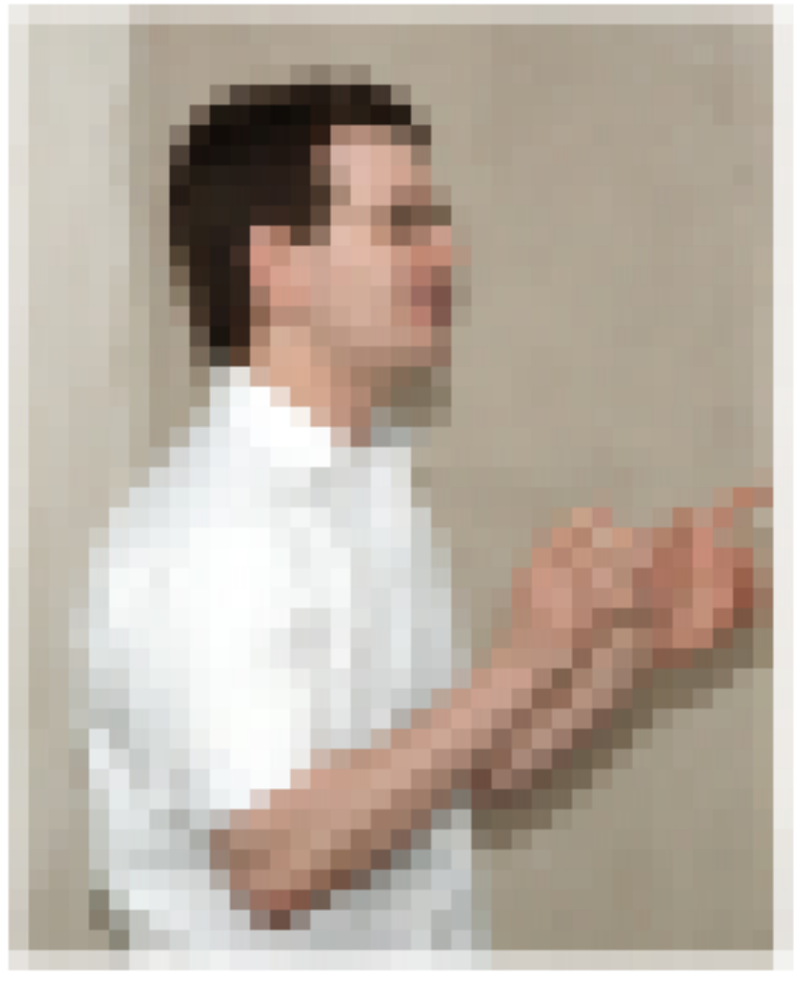
-Inigo Montoya

# *Mereology*

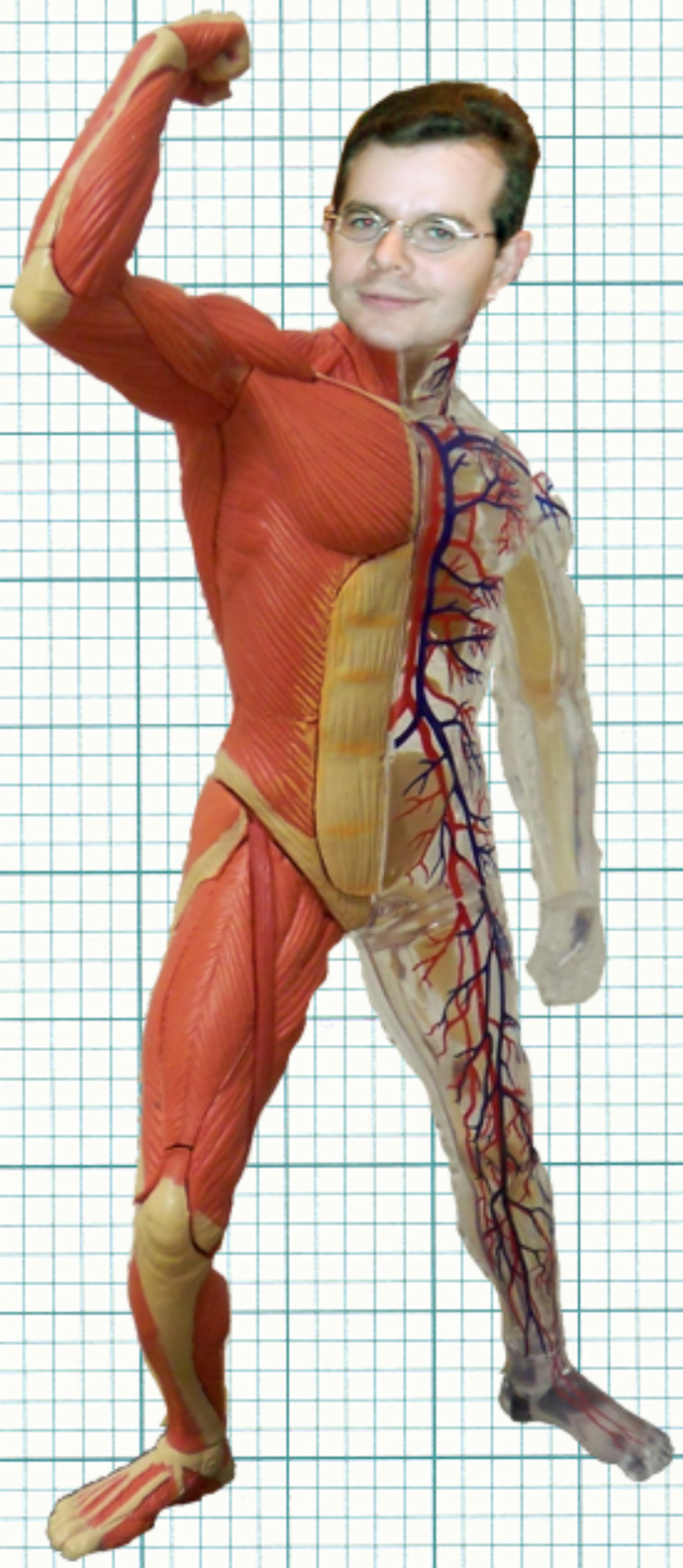
The study of part-part and part-whole relationships.

# *Lossy Compression*

A description made from another description that omits some details.



# *Reduction*



Take a whole system,
determine its parts,
and how those parts interact.

# *Constitution*

Determine what parts
make up (constitute)
a whole.

# Deep Thoughts

by
Jack Handey

Maybe in order to understand mankind,
we have to look at the word itself.
'Mankind'.
Basically, it's made up of two separate words
--- 'mank' and 'ind'.
What do these words mean?
It's a mystery, and that's why so is mankind.

# *Emergence*

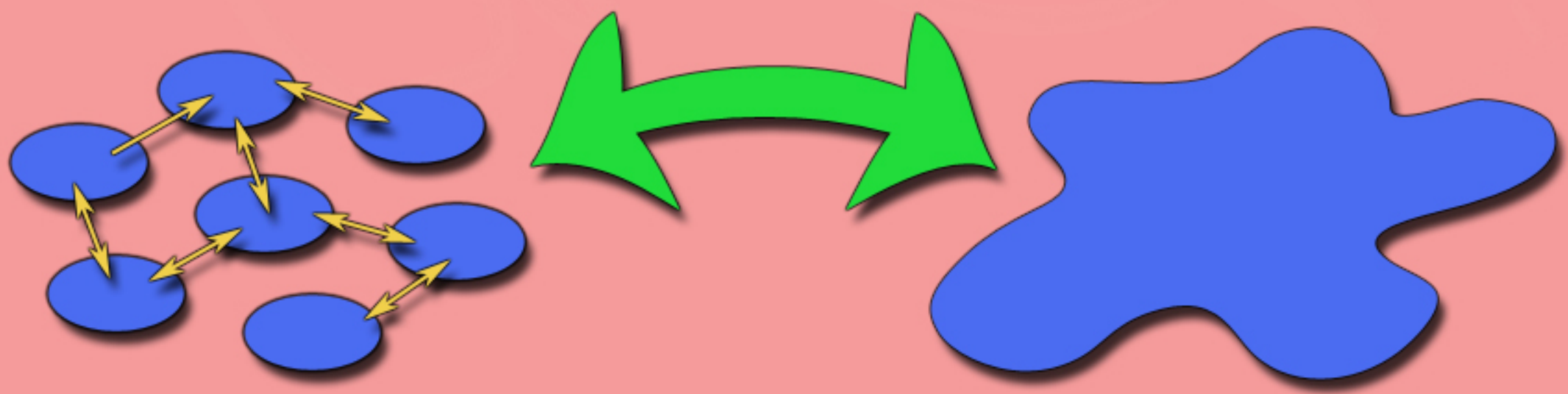
Take some parts and their relations,
and determine the whole system they generate.

# *Bridge Laws*

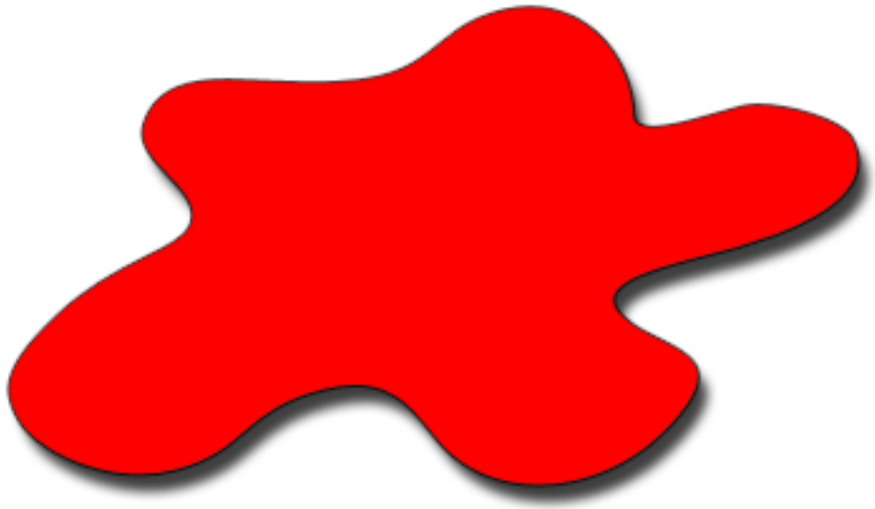
A translation of phenomena from
one domain to another domain,
or from one scale to another scale.

# Reduction ♥ s Emergence

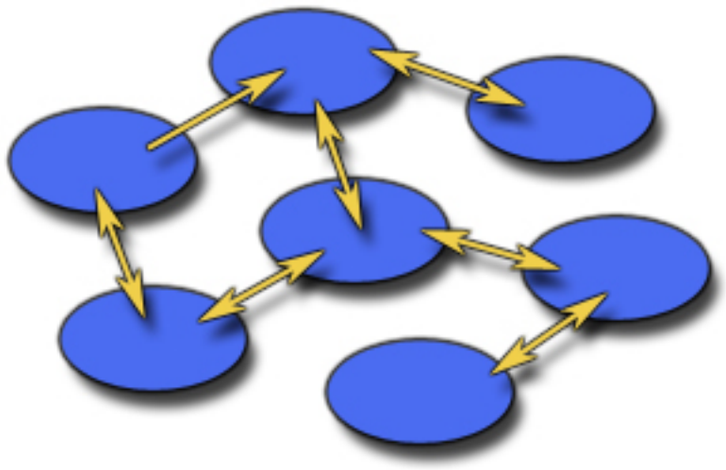
Reduction and emergence are mereologically inverse bridge laws.



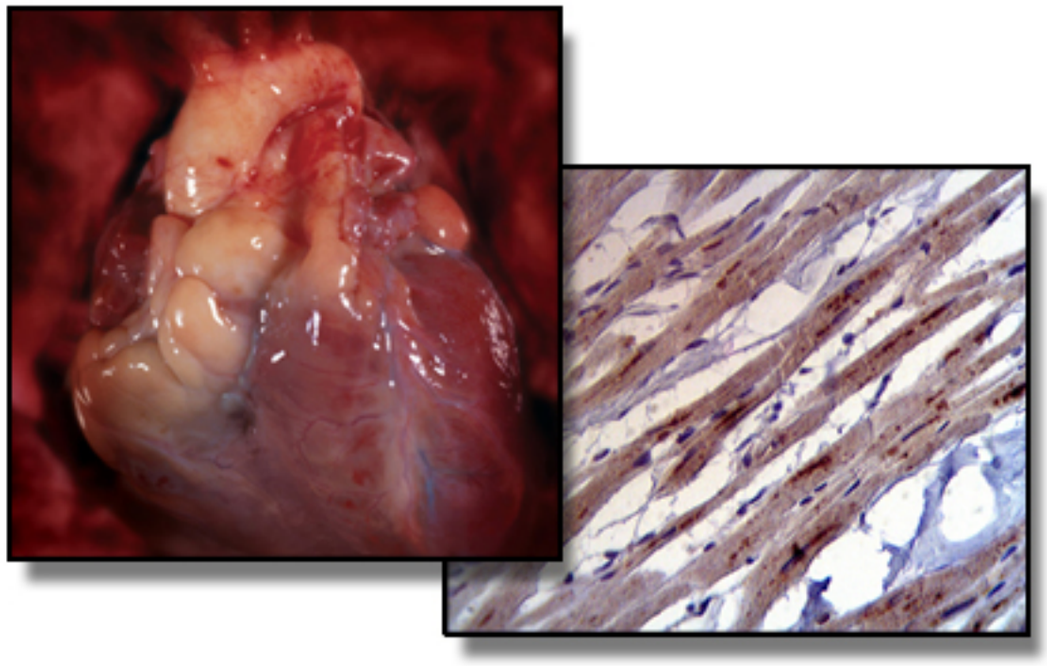
# *Supervenience*



***A* supervenes on *B***
if and only if there can be
***no change in A without a change in B.***



# *Supervenience*



# Make Emergence Safe for Science

- Emergence is a cornerstone concept of complexity theory, but without any formal meaning it does no work for us.

- Or worse, it obscures confusion and is pure rhetoric.

- Given what has just been said, how can we refine that concept?

- Science (any kind of science) works on testable claims, repeatable experiments, and reproducible results.

- How can we make claims of emergence scientific claims?

# *Develop Tests for Emergence*

Truth evaluable claims about
repeatable and reproducible features
of relationships between
specific micro-level phenomena
and their generated macro-level phenomena.

# Designing an Agent-Based Model

- Designing is different from building, but building develops intuitions for better ABM design.

- The technique doesn't have tried and true standards to rely on.

- You should have some justification for every modeling decision, although many features are implicit and ignored in practice.

- Furthermore, some decisions you won't be able to justify yourself, and they are beyond the scope of your modeling interests.

- Thus good modeling is a bit of an art, a bit of luck, and a bit of marketing; but finding appropriate methods is possible.

# Snowball & Goldilocks Programming

- Agent-based models can include any level of detail, but you may not need to include everything to explore/explain/predict your target phenomenon.

- Start as simple as possible, and build increasing complication.

- Remember that the goal is to be useful, and a model that is as complicated as the real system may be less useful.

- Complexity requires a certain amount of complication, the key is to include the “right” features and components.

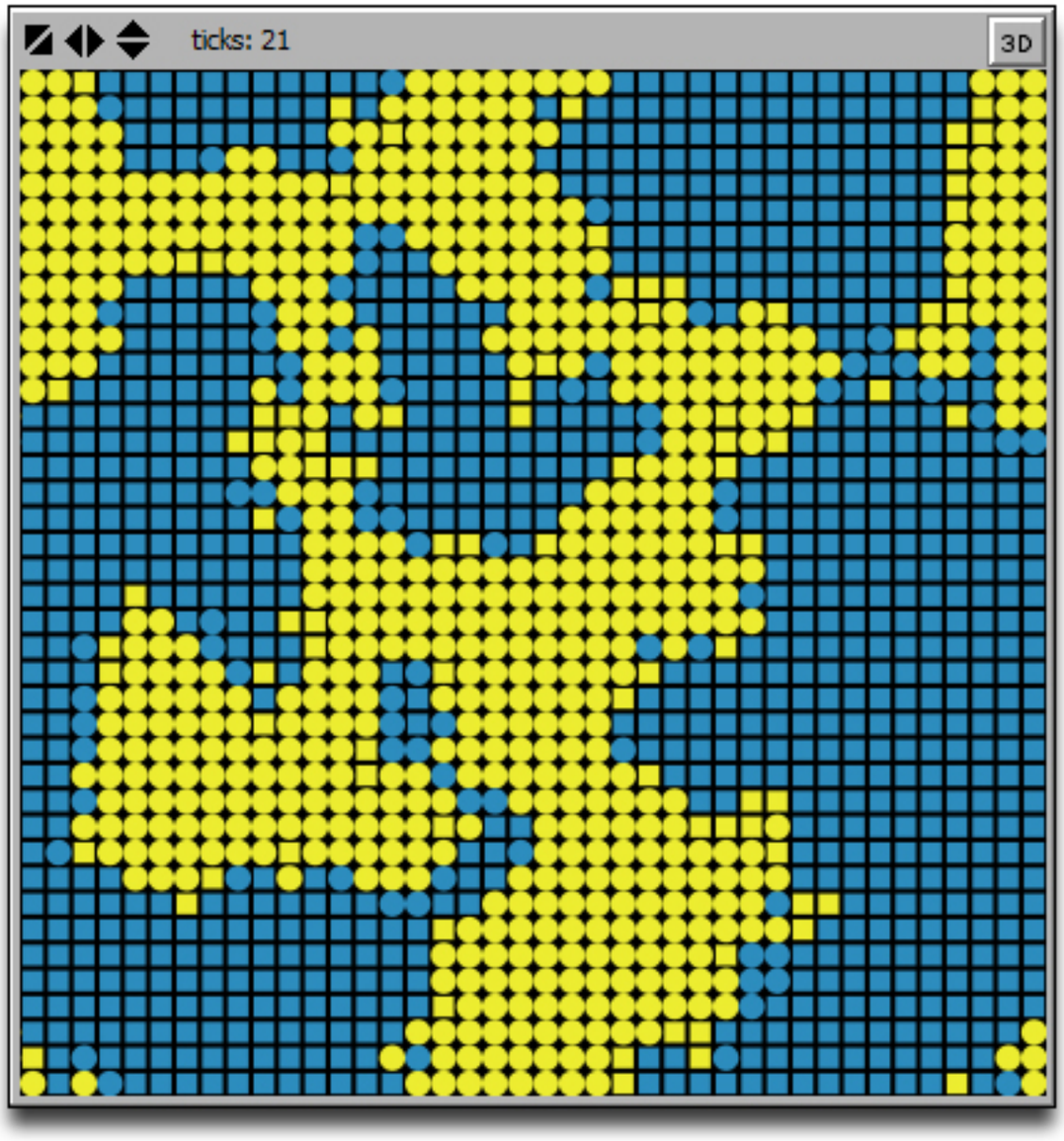
# Build for and with Science

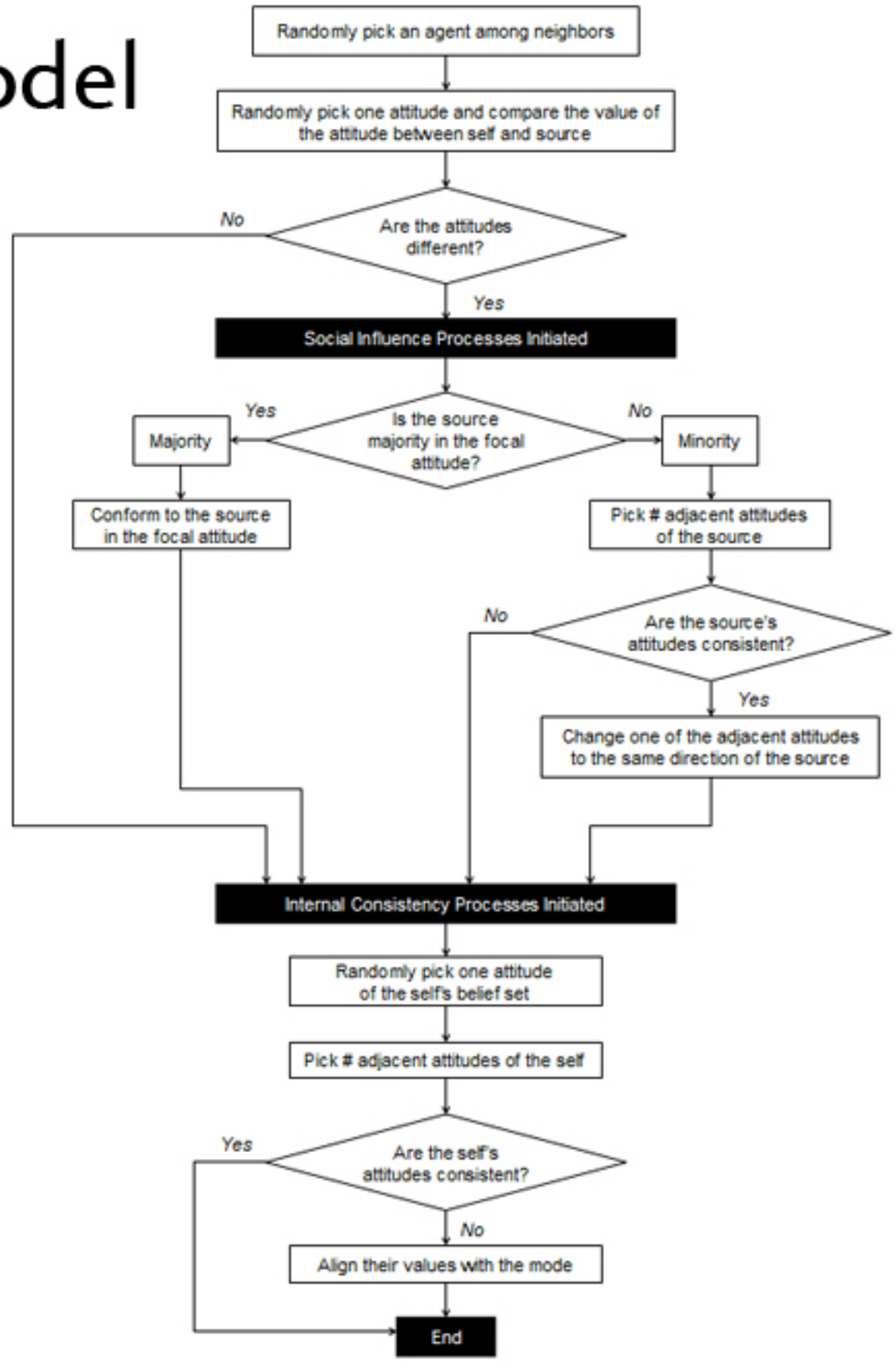
- Start with deep analysis of your system to isolate essential components and mechanisms.
- Have clear expectations about model behavior and output.
- Have hypotheses and ways to test them - model validation.
- Plan model with flowcharts, mathematics, and concise descriptions of algorithms.
- With experience, these tasks become easier too.

# Plan Algorithms before Writing Code

- Start with pencil (or pen) and paper.
- Describe each module in words first.
- Plan each module as an independent function with well-defined inputs and outputs.
- Use a workflow diagram to integrate the components.
- Write the code and reevaluate that it is what you need.
- Repeat these steps to build a model step by step.

# Building an Agent-Based Model



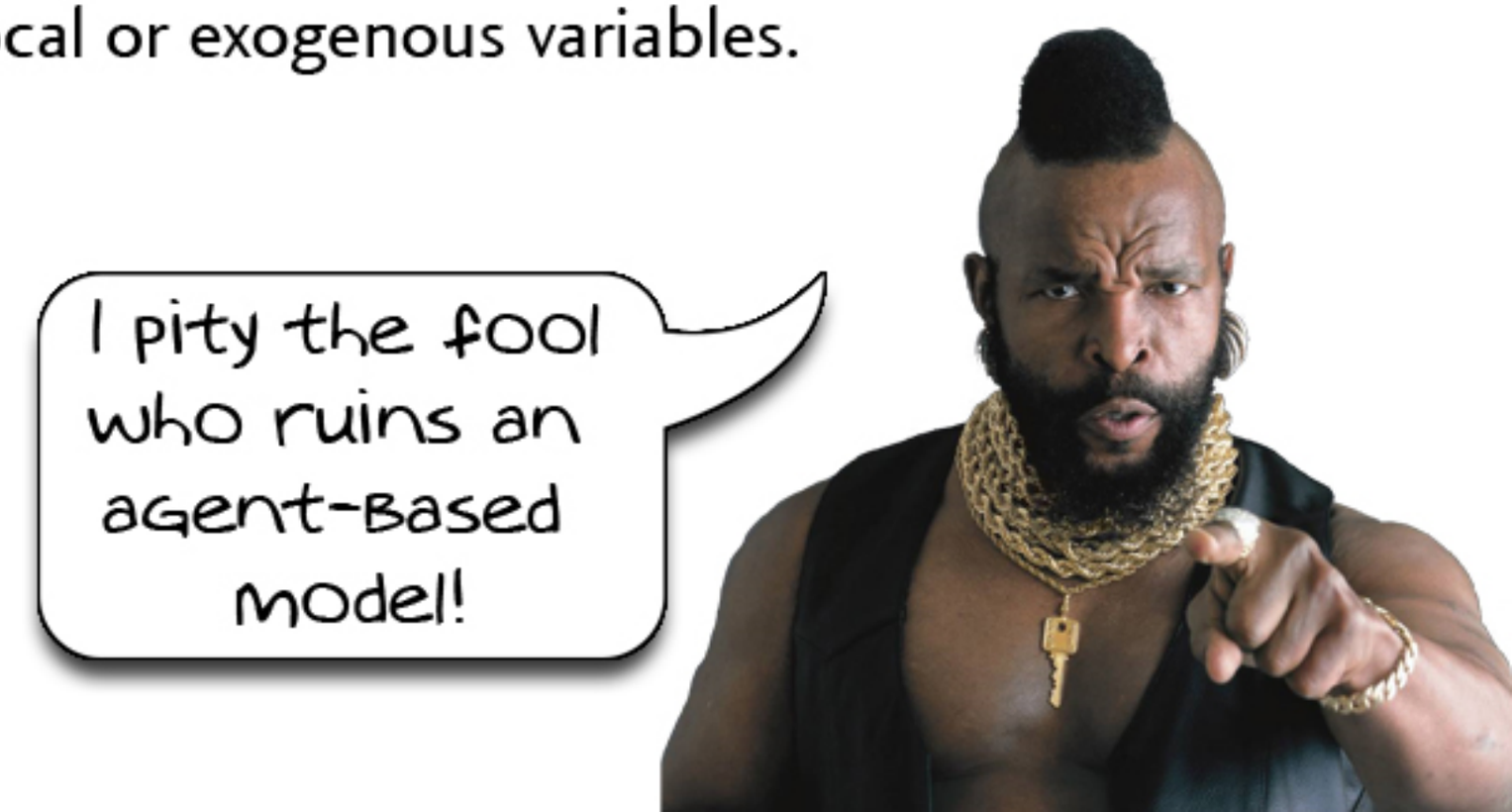



# Modularity and Code Reuse

- Save time and brain power by reusing methods already available.
- Use the same code for the same behavior each time.
- However, consider alternate methods from time to time.
- Encapsulated methods can be used across models.
- Have two Netlogo applications running side by side.
- Learn and assign keyboard shortcuts and built-in functionality.

# Things that Ruin an Agent-Based Model

- Aggregates of micro-level agents are also agents.
- Agents react to macro-level measures or phenomena (that they couldn't directly know about).
- Agents directly affect macro-phenomena through behavior.
- Agents react to non-local or exogenous variables.



# Putting Your Models to Work

- Use simple, abstract models to prove that some mechanism is sufficient to generate a macro-level phenomenon.

- Add more complication to encode your theory of agent behavior.

- Calibrate the model to data and use micro behaviors to explain patterns at the macro level, and how they are generated.

- Explore the ramifications of changes in the theory and changes in the environment...which includes interventions.

- Build and analyze your models with complexity in mind.

# Deciding What to Include

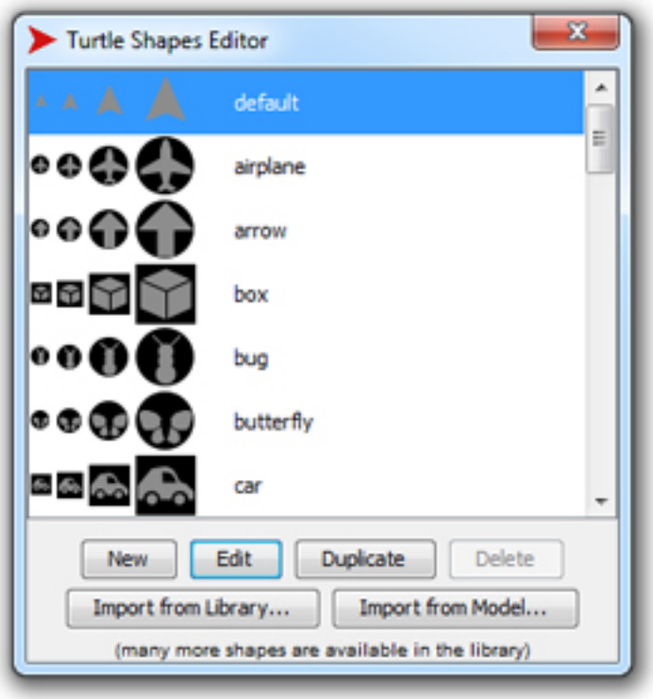
- Like anything else, people exploit the techniques they know well.
- This is especially true for ABM, because people want to reuse code.
- Therefore you have to be especially self-critical, and always check that your model's components are appropriate for your problem.
- It helps to read others' approaches with an open mind.
- And sometimes, if you can write solvable games or equation models, you are better off NOT using an ABM.

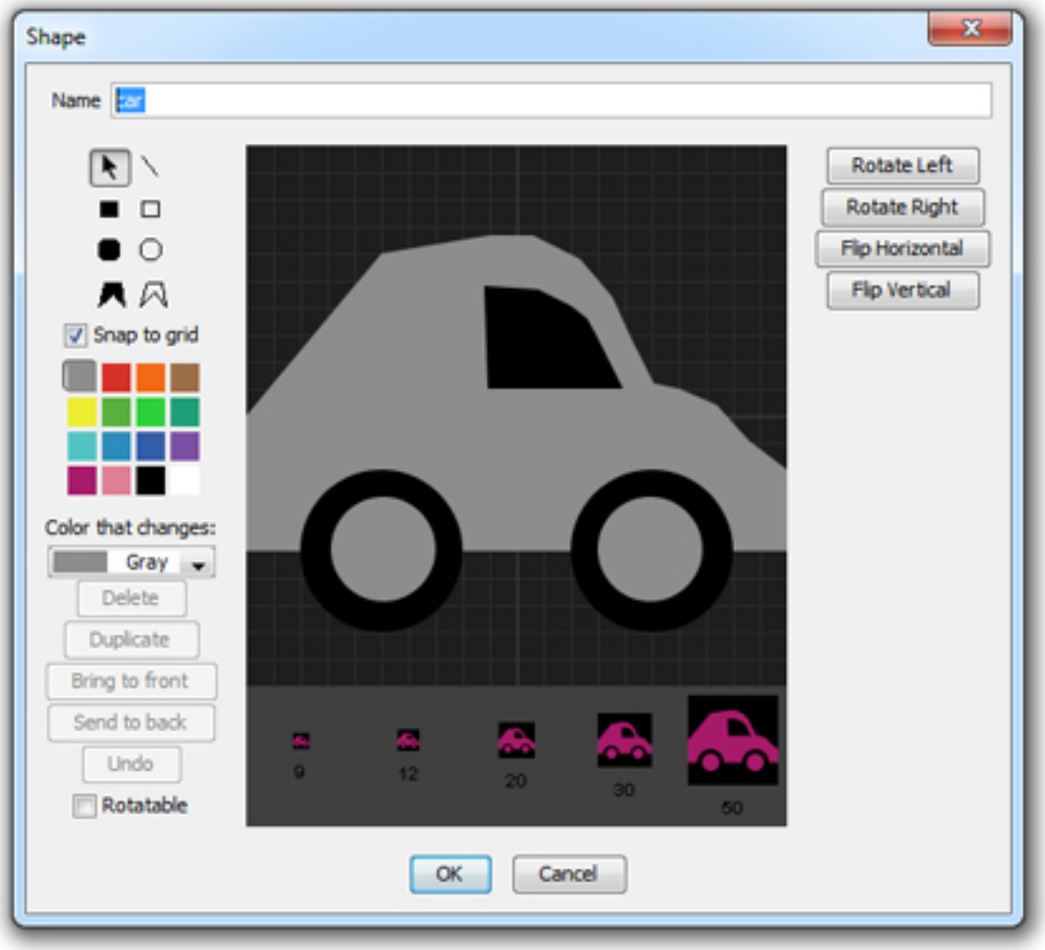
# Netlogo Tools and Advanced Techniques

- Shapes Editor
- Java Extensions
- Importing Data, Images, & 2D Data Files
- Network Models and the Extension
- Audio Feedback
- Mouse Interaction
- Systems Dynamics Modeler
- Hubnet - Participatory Models
- Mathematica Link
- Netlogo 3D
- Geographic Information Systems

# Turtle Shapes Editor

- Vector driven for speed, but it's still better to keep them simple.
- Large shapes library included, plus you can import from other models.
- Limited color options, and only one color is controlled by code.
- Can be a rotatable image or not.
- Limited tool, but good enough.



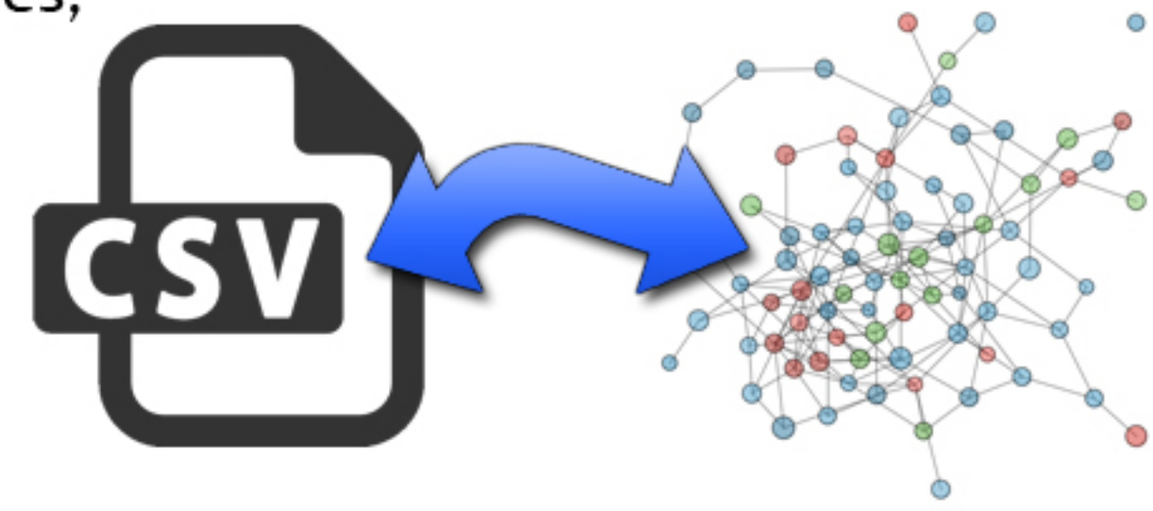



# Java Extensions

- Netlogo API for writing Java interfaces (incomplete).
- Extensions work in applet mode too.
- Import existing Java capabilities into Netlogo.
- Use Netlogo interface for feedback of Java model.
- Provide access to advanced analytics.

# Importing Data into Netlogo

- Import picture (bmp, jpg, png, gif) and use as backdrop, or to feed information to the patches.
- Import data file and assign values to patches.
- Import all the variables and settings for a model (import-world).
- Import network data with “attributes” and “links” files.
- See models library, code examples, and user manual for details.



# Network Models

- Edges are called “links” in Netlogo.
- Can be directed or symmetric (or ties).
- Built in support for multiple layouts, accessing link-neighbors, multiple link “breeds”, and few other things.
- No native network measures, limited visualization, and can run slow because links are agents.
- The Network Java Extension adds a lot of analysis capabilities, but also imposes restrictions on the model.

# Audio Feedback with Sound Extension

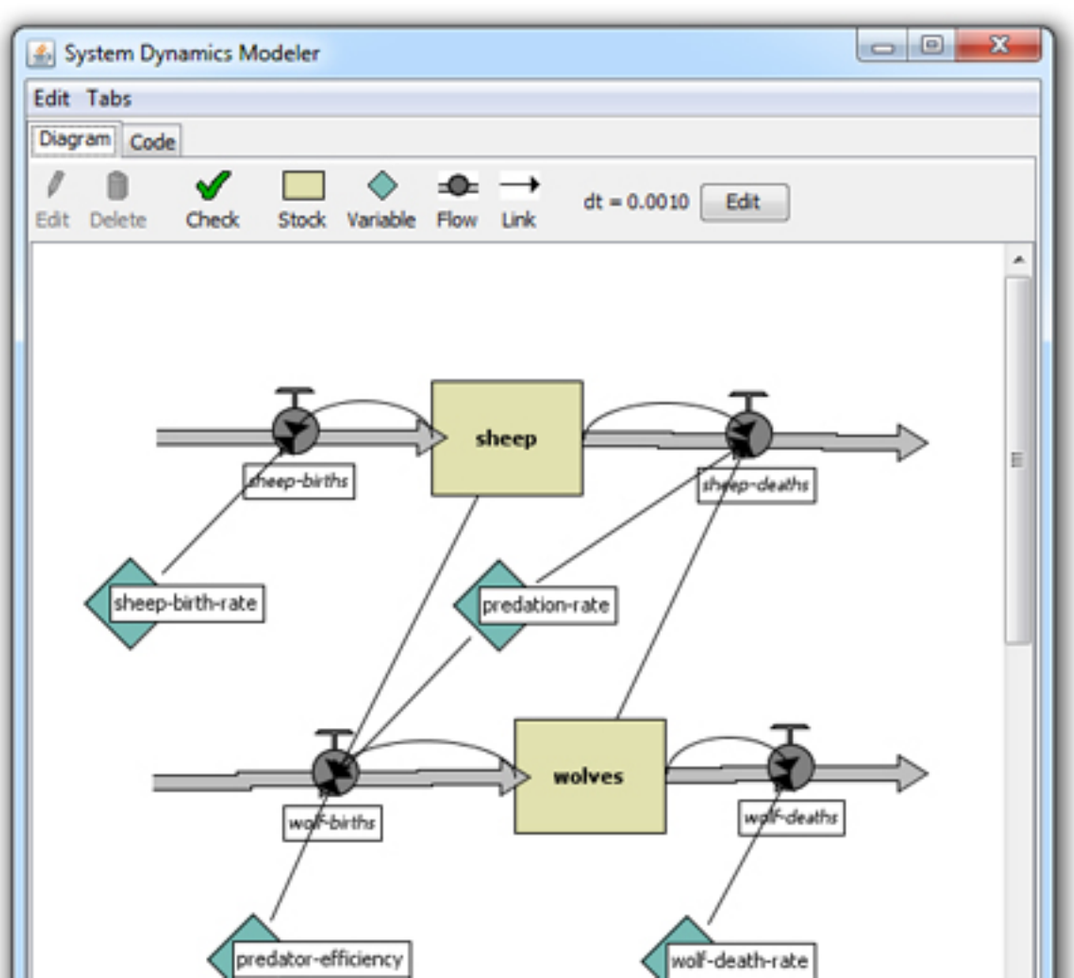
- Create sound effects and alerts
- Auralization of data stream
- Preinstalled - see Extensions section of user manual
- MIDI method simulates a 128-key electronic keyboard with 47 drums and 128 melodic instruments
- Supports 15 instruments and one percussion channel
- Pitch and volume adjustable on 0 to 127 scales
- Plays a recorded sound file, e.g. your voice!
- It supports WAV, AIFF, and AU files.

# Mouse Interaction

- Select, drag, add, remove, etc agents and other model features.
- Draw aspects of the world that agents see.
- Manually position agents, objects, and patch values.
- User interaction with a computer...like a game.
- And what else?

# System Dynamics Modeler

- Population-based dynamics; not ABM.
- Define stocks, flows, variables, and links.
- Intuitive Netlogo interface (variables and plots).
- Generates code for you.
- Port and compare with ABM.
- See user manual and examples.

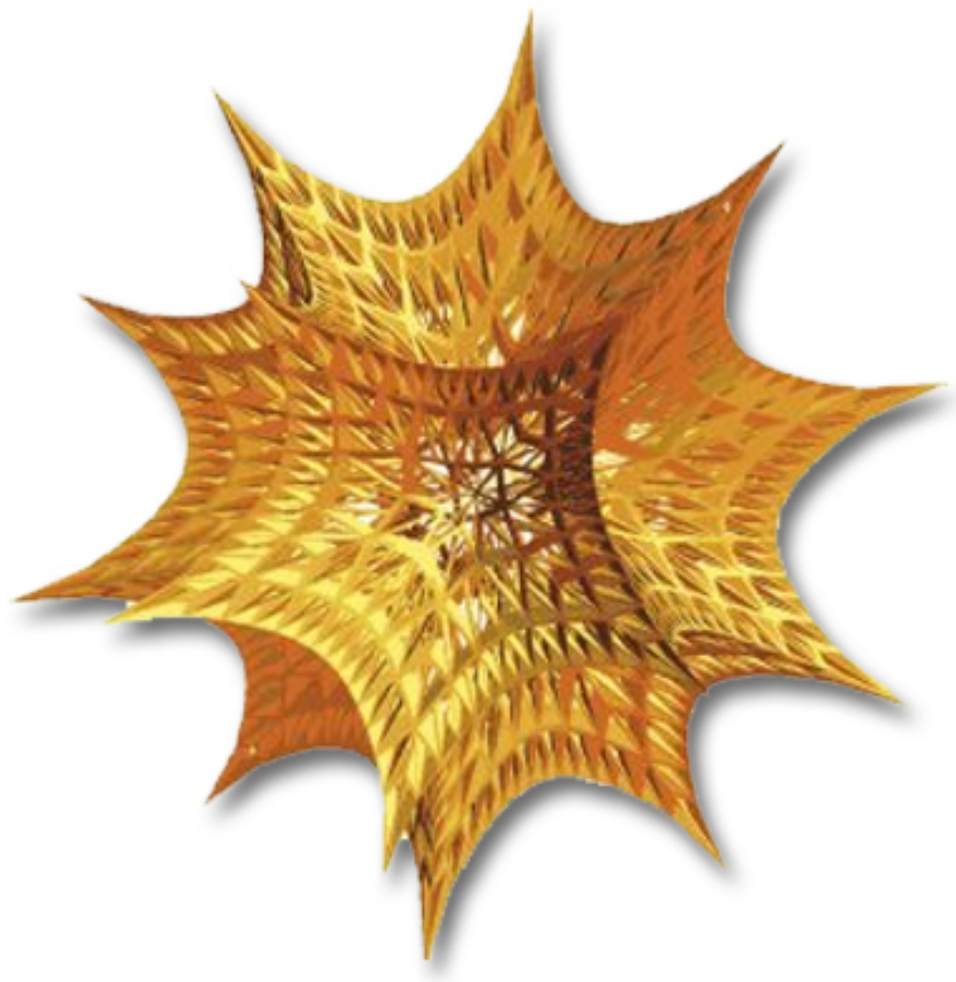


# HubNet Participatory Models

- Allow humans to control agents and/or parameters.
- Support for computers, calculators, cell phones, PDAs.
- Remote data collection and model integration.
- Teach concepts on distributed action and emergence.
- Run social science experiments.
- Still limited and somewhat difficult to set up and use.

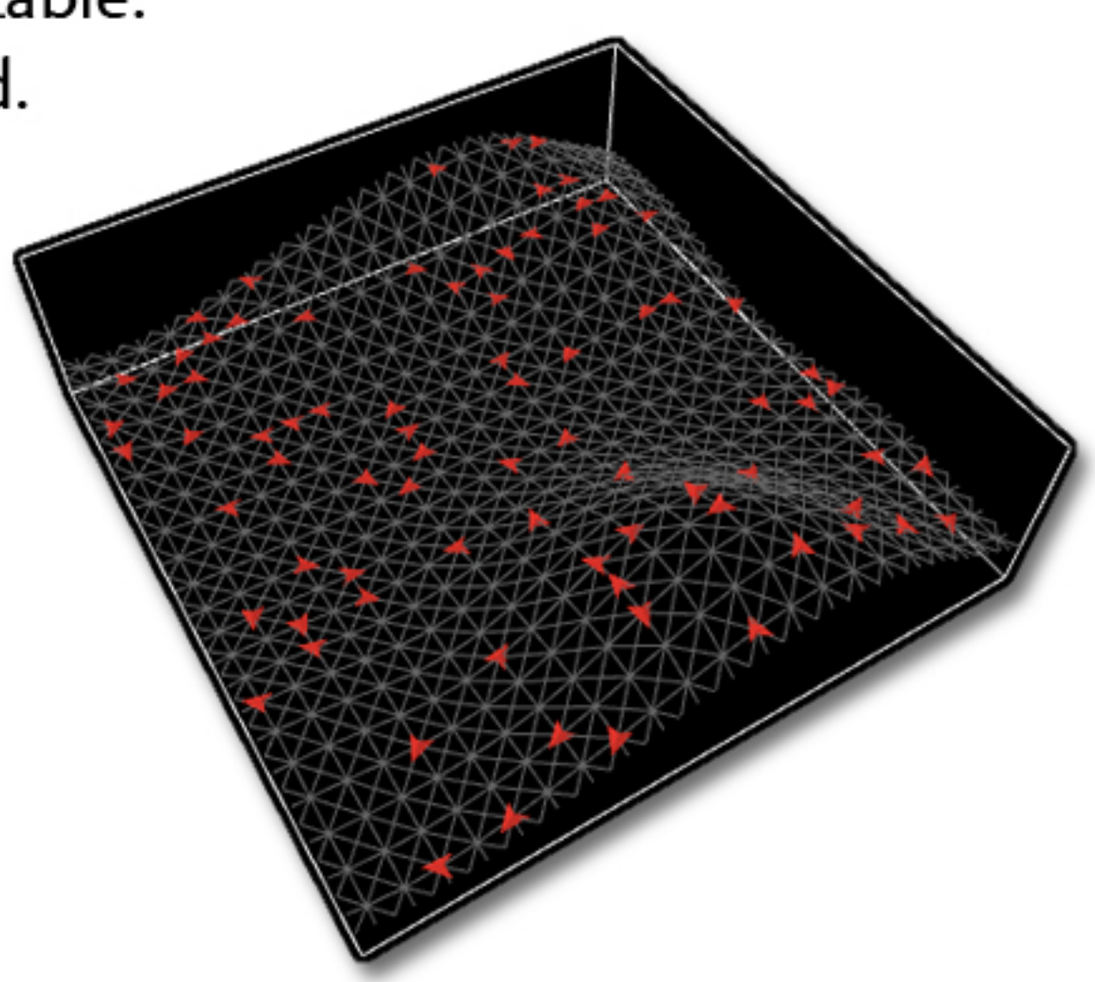
# Mathematica Link

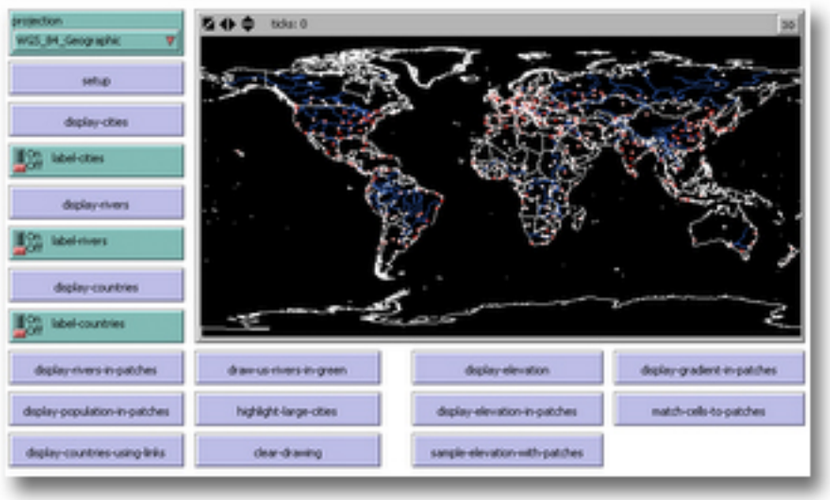
- Real-time interaction between applications.
- Provides a highly interactive self-documenting workflow.
- Replace the Behavior Space with adaptive sweeping.
- Useful in model debugging.
- Utilize Mathematica for:
  - data importing
  - statistical functions
  - network analysis
  - document creation
  - visualization
  - and much, much more.



# Netlogo 3D

- Add another spatial dimension to your model.
- Separate program but downloaded together.
- 3D world and interface/procedure tabs in separate windows.
- Most commands are the same, but no real manual.
- No shapes editor GUI, but generatable.
- Transparency for turtles supported.
- No applet support.
- Full integration planned.



# Geographic Information Systems

- Import and Interact with GIS Data.
- Import real-world geographic and demographic data.
- Supports ESRI .shp files to import vectors.
- Supports ESRI .grd and .asc files for raster grids.
- Not just converting to patch values (though that's possible).
- Maintain and manage vectors directly in Netlogo.
- Extract features to assign values, create agents, etc.
- Still young, but quickly growing.