Sharif University of Technology
Department of Mathematical Sciences
Complex and Multi-Agent Systems Lab

Agent Based Modelling and Simulation Tools An Overview

Meysam Madani

November 20, 2014

A Workshop on Mechanism Design and Computational Modeling of Social Systems, IPM, Tehran,Iran

Outline

Outline

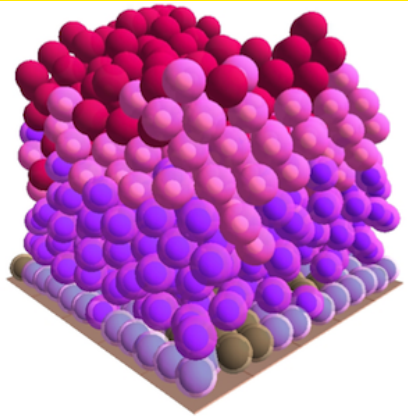
- Simulation tools are useful for accelerating the development and verification of switching converters and their control systems.
- In this talk we overview some commonly used simulation software and applications in Agent-based and complex systems.
- Particularly we will overview the features and capabilities of NetLogo Software and code in practice a simple model.
- Finally we will study the programming of game theoretic Dove-Hawk problem, which was programmed by Rick O'Gorman.

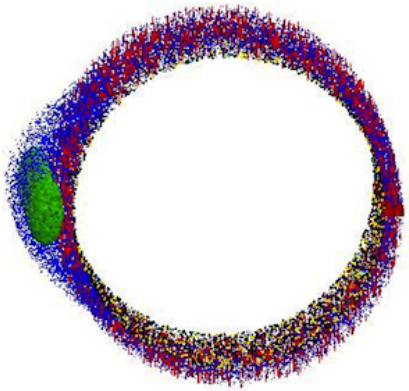
To find these slides and some more Just Google **Meysam**

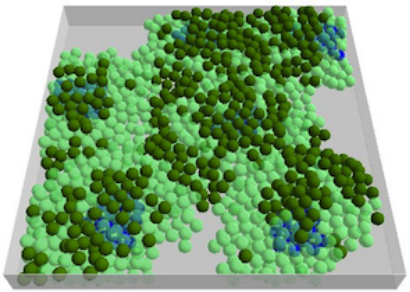
On Agent Based Modelling and Simulation Tools

FLAME

Flexible Large-scale Agent Modelling Environment

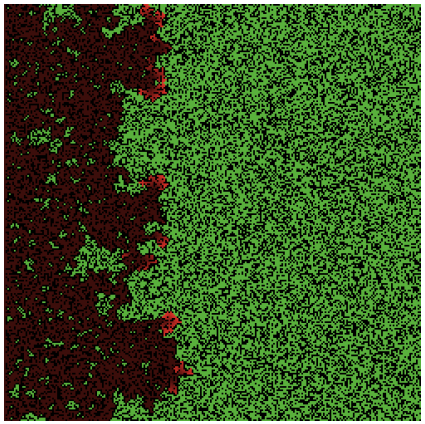


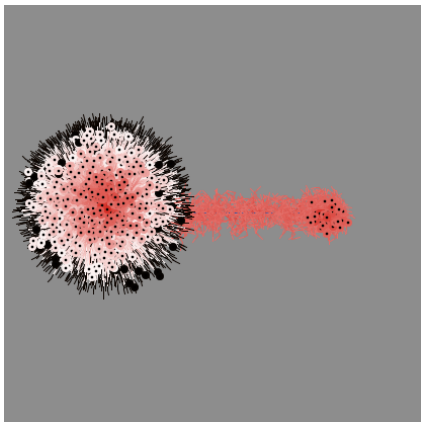


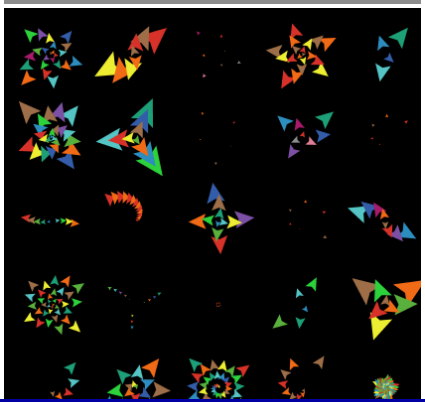


- FLAME is a generic agent-based modelling system which can be used to development applications in many areas.
- Models are created based upon a model of computation called (extended finite) state machines.
- FLAME framework can automatically generate simulation programs that can run models efficiently on HPCs.
- It generates a complete agent-based application which can be compiled and built on the majority of computing systems ranging from laptops to HPC super computers.
- The FLAME Model Library is a collection of relatively simple models that illustrate the use of FLAME in different applications.
- **Download** Programs to run FLAME models (xparser and libmboard) and Programs to edit FLAME models and visualise data (FLAME editor and visualiser) are currently available from the CCPForge website.
- **Free, Good Model Library, Hard for beginners, C Based,**
- Start From http://www.flame.ac.uk/docs/overview.html

NetLogo

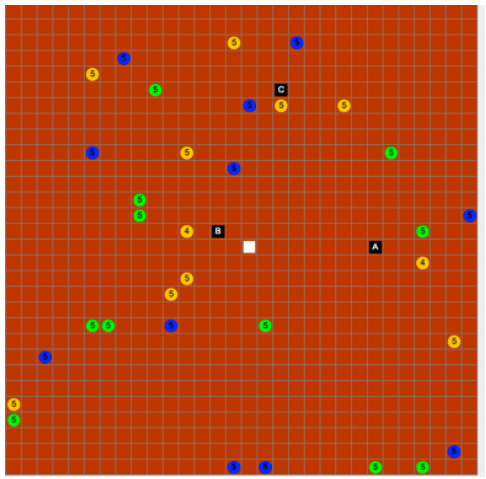


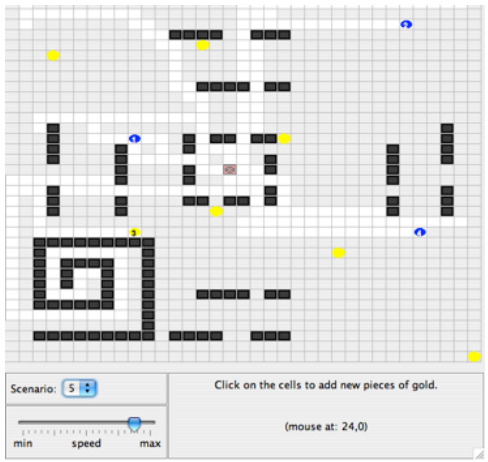


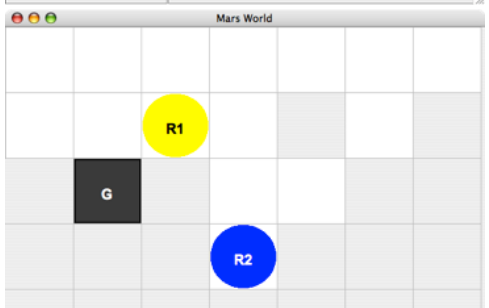


- NetLogo is a multi-agent programmable modeling environment.
- It also powers HubNet participatory simulations.
- Help beginning users get started authoring models
- **Free, Great Model Library, Easy for beginners, Lisp and Logo Based,**
- Start From http://ccl.northwestern.edu/netlogo

Jason
Agent Programming

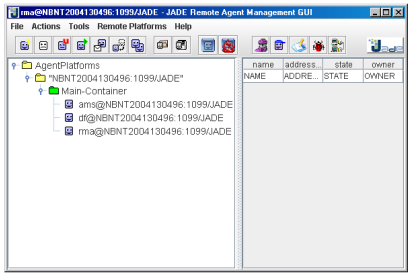


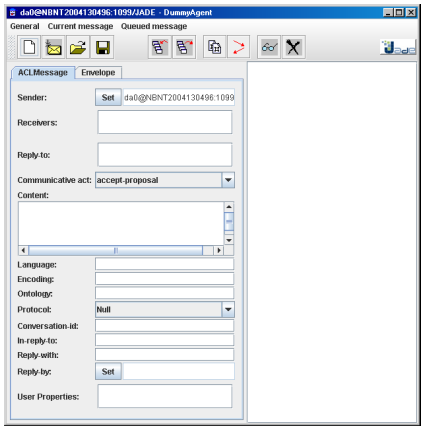




- Jason is an interpreter for an extended version of AgentSpeak.
- It implements the operational semantics of that language,
- Strong negation, so both closed-world assumption and open-world are available
- Annotations in beliefs used for meta-level information and annotations in plan labels
- One of the best known approaches to the development of cognitive agents is the BDI (Beliefs-Desires-Intentions) architecture.
- AgentSpeak has been one of the most influential abstract languages based on the BDI architecture.
- the possibility to run a multi-agent system distributed over a network (using Saci or JADE);
- An IDE in the form of a jEdit or Eclipse plugin; the IDE includes a mind inspector that helps debugging.
- **Open Source, Manual and Refs, GUI, Java Based**
- Start From http://jason.sourceforge.net/wp/

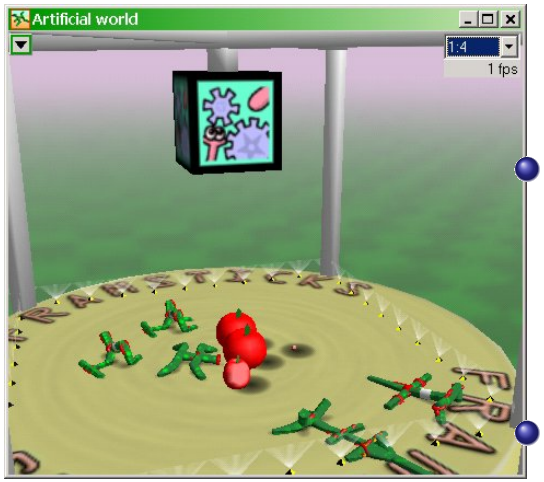
JAVA Agent Development Framework





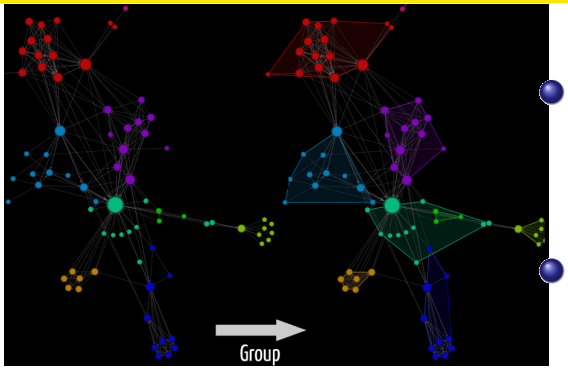
- Jade is an open source platform for peer-to-peer agent based applications
- It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that support the debugging and deployment phases
- A JADE-based system can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI.
- The configuration can be even changed at run-time by moving agents from one machine to another, as and when required.
- JADE is completely implemented in Java language and the minimal system requirement is the version 5 of JAVA
- ADE is free software and is distributed by Telecom Italia, LGPL
- An IDE in the form of a jEdit or Eclipse plugin; the IDE includes a mind inspector that helps debugging.
- **Open Source, Manual and Refs, GUI, Java Based**
- Start From http://jason.sourceforge.net/wp/

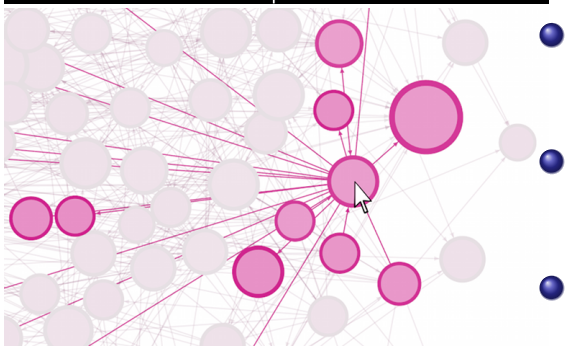
Framsticks

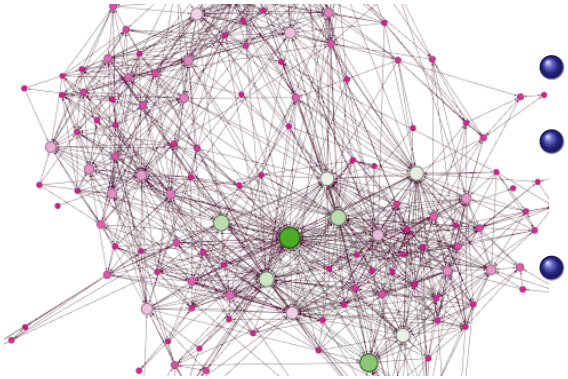


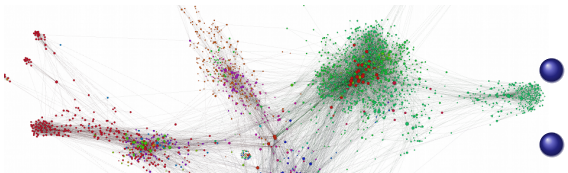
- Framsticks is a three-dimensional life simulation project.
- Both mechanical structures ("bodies") and control systems ("brains") of creatures are modeled.
- It is possible to design various kinds of experiments, including simple optimization, coevolution, open-ended and spontaneous evolution, distinct gene pools and populations and modeling of species and ecosystems.
- Users of this software work on evolutionary computation, artificial intelligence, neural networks, biology, robotics and simulation, cognitive science, neuroscience, medicine, philosophy, virtual reality, graphics, and art.
- The system can be interesting for experimenters who would like to evolve their own artificial creatures and see them in a three-dimensional, virtual world.
- **Free, Tutorial step by step, Manual, A graphical interface, Java Based**
- Start From http://www.framsticks.com

Gephi
makes graphs handy



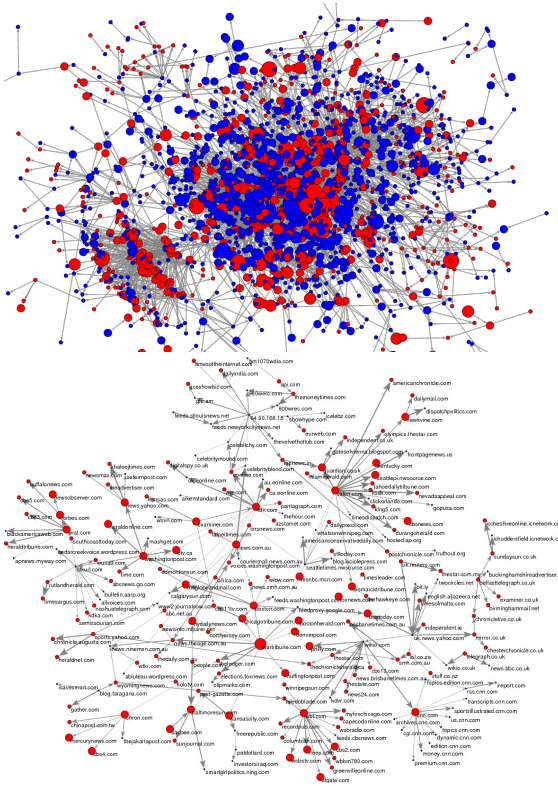


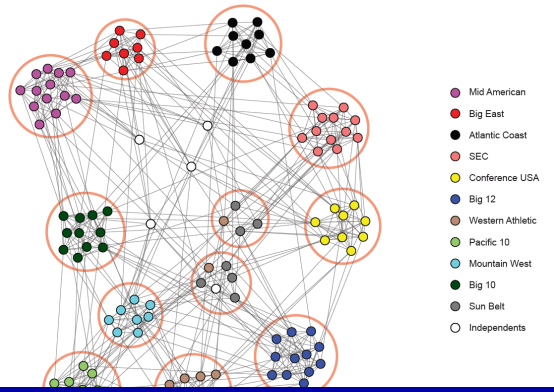




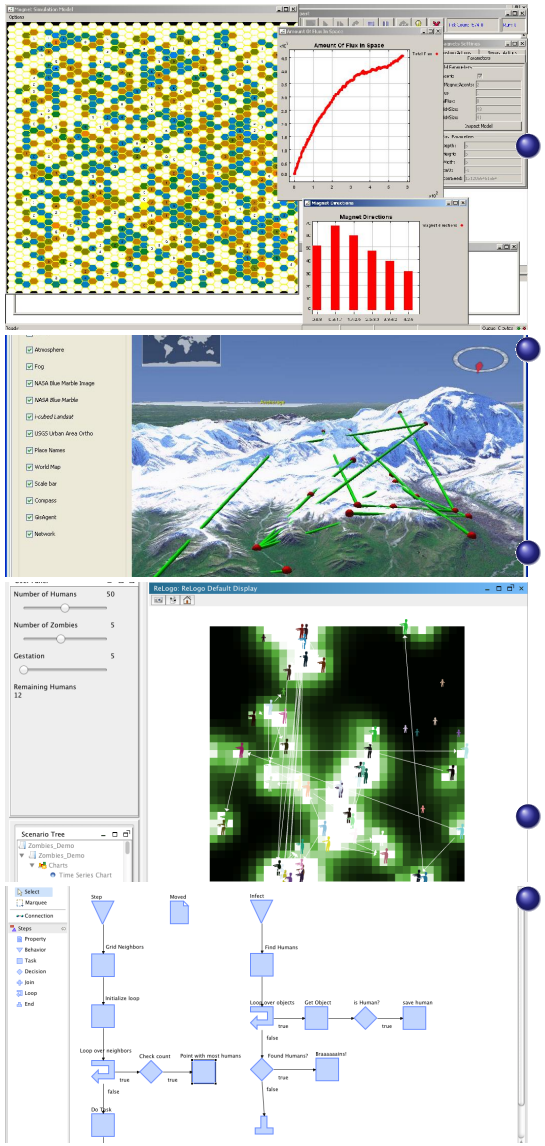
- Gephi is an interactive visualization and exploration platform for all kinds of **networks** and **complex systems**, **dynamic and hierarchical graphs**.
- **Runs** on Windows, Linux and Mac OS X. Gephi is **open-source** and free.
- **Exploratory Data Analysis**: intuition-oriented analysis by networks manipulations in real time.
- **Link Analysis**: revealing the underlying structures of associations between objects, in particular in scale-free networks.
- **Social Network Analysis**: easy creation of social data connectors to map community organizations and small-world networks.
- **Biological Network analysis**: representing patterns of biological data.
- **Poster Creation**: scientific work promotion with hi-quality printable maps.
- **Metrics**: Centrality, degree (power-law), betweenness, closeness, density, path length, diameter, HITS, modularity, clustering coefficient.
- Very good **DataBases** For research.
- Start From https://gephi.github.io/

Stanford Network Analysis Projec



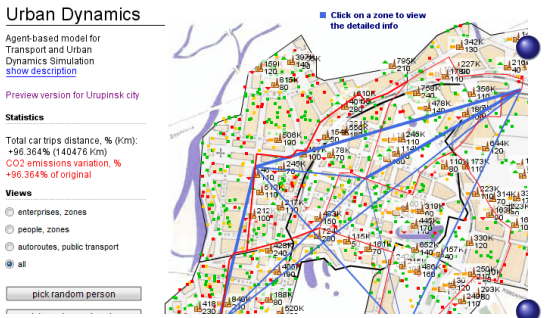


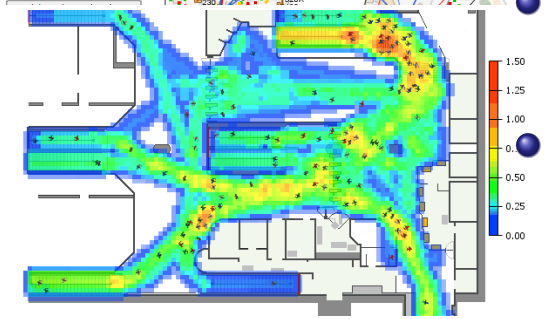
- Stanford Network Analysis Platform (SNAP) is a general purpose, **high performance** system for analysis and manipulation of **large networks**.
- It easily scales to massive networks with **hundreds of millions** of nodes, and billions of edges.
- It efficiently manipulates large graphs, **calculates structural properties**, **generates** regular and random graphs, and supports attributes on nodes and edges.
- Complete source code for the core SNAP and GLib libraries is available under the BSD license.
- A large set of DataBases of Networks (some with millions of nodes and edges) http://snap.stanford.edu/data/index.html
- Start From http://graphlab.org

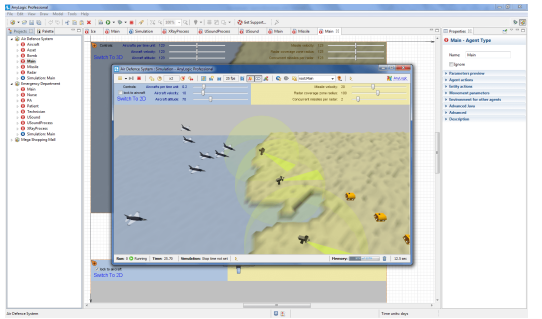


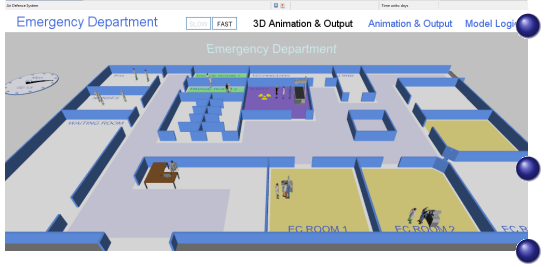
- The Repast Suite is a family of advanced, free, and open source agent-based modeling and simulation platforms that have collectively been under continuous development for over 14 years
- Repast Simphony 2.2, released on 26 June 2014, is a richly interactive and easy to learn Java-based modeling system that is designed for use on workstations and small computing clusters.
- Repast for High Performance Computing 2.0, released on 12 August 2013, is a lean and expert-focused C++-based modeling system that is designed for use on large computing clusters and supercomputers.
- Start From http://repast.sourceforge.net/

AnyLogic 6 A Multimethod Simulation Software









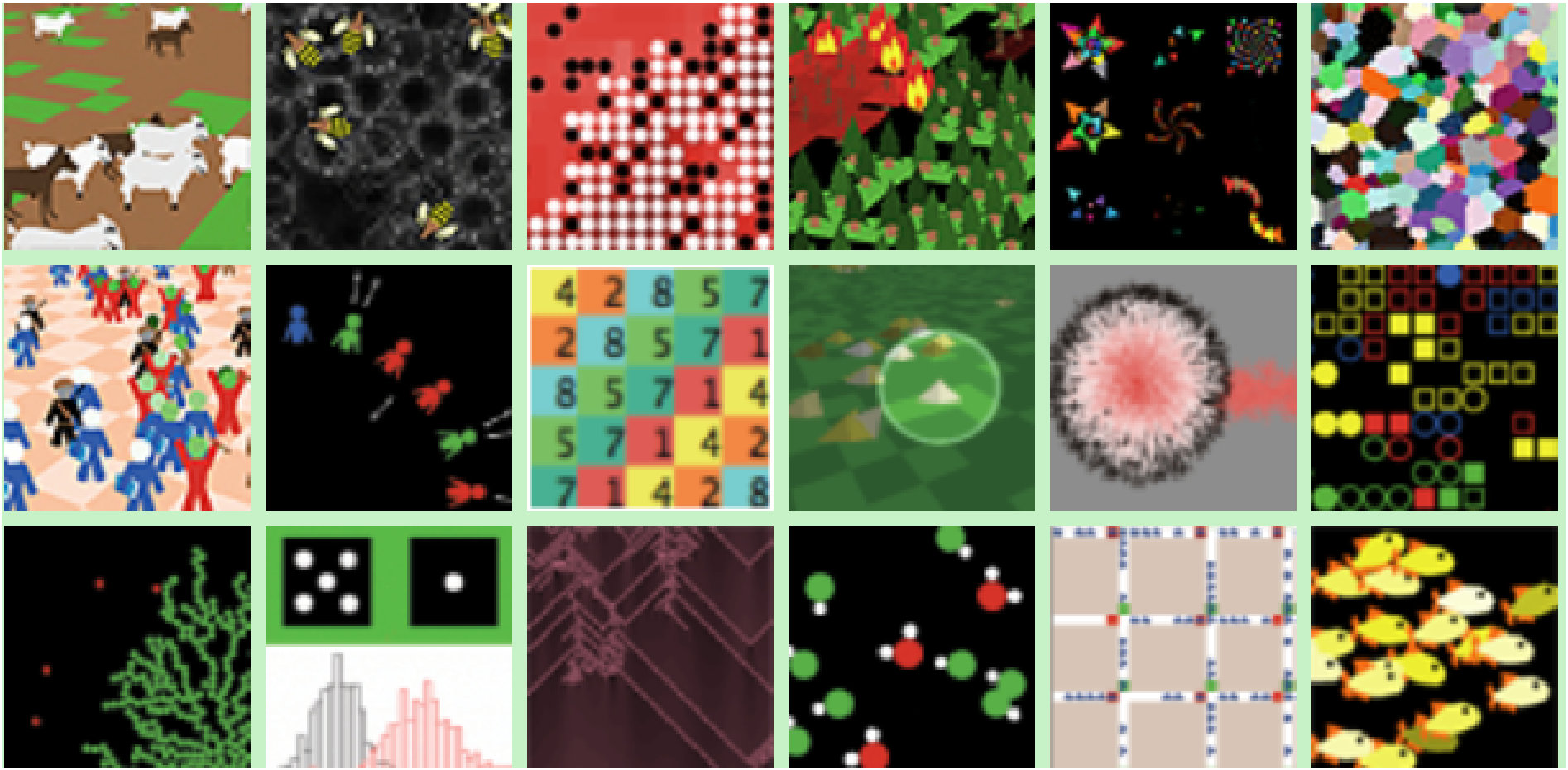
- AnyLogic is the only simulation tool that supports all the most common simulation methodologies in place today: System Dynamics, Process-centric (AKA Discrete Event), and Agent Based modeling.
- AnyLogic's visual development environment significantly speeds up the development process
- It has **applicational models** in Manufacturing— Logistics and Supply Chains— Markets and Competition— Business Processes Modeling— Healthcare and Pharmaceuticals Simulation— Pedestrian Traffic Flows— Information and Telecommunication Networks Simulation Modeling— Social Process and Marketing Simulation— Asset Management and Financial Operations with Simulation Modeling— Warehouse Operations and Layout Optimization
- AnyLogic is a professional software and has 4 sort of licences, Professional, Advanced, University Researcher, Educational.
- It supports Windows, Mac, and Linux
- Start From http://www.anylogic.com

Programming and Graphic in NetLogo

From NetLogo Site, I confirm them!

- NetLogo is a programmable modelling environment for simulating natural and social phenomena.
- It was authored by Uri Wilensky in 1999 and has been in continuous development ever since at the Center for Connected Learning and Computer-Based Modeling.
- Modelers can give instructions to hundreds or thousands of "agents" all operating independently.
- This makes it possible to explore the connection between the micro-level behaviour of individuals and the macro-level patterns that emerge from their interaction.
- NetLogo lets students open simulations and "play" with them, exploring their behavior under various conditions.
- It also comes with the Models Library, a large collection of pre-written simulations that can be used and modified.
- NetLogo can also power a classroom participatory-simulation tool called **HubNet**
- Through the use of networked computers or handred devices, each student can control an agent in a simulation.

Netlogo Library



Just choose **Model Library** from **File** Tab.
About **350** Sample Models to Work, From Art and Economic Models to Earth science and chemistry Models.
Anyone can build his/her own correct **Model** and upload it to its library.

Programming in NetLogo

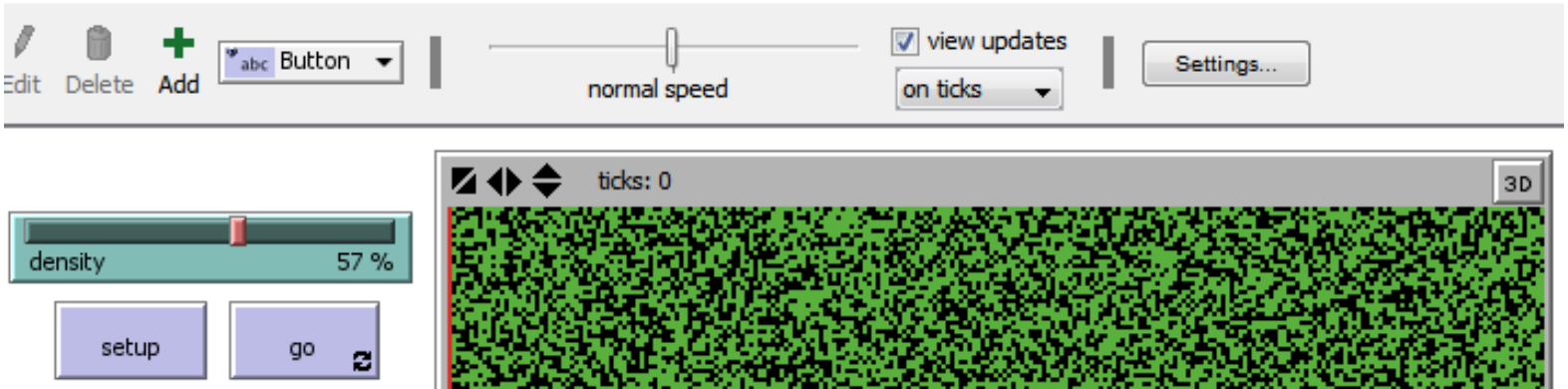
- It supports International character set.
- Fully programmable
- Approachable syntax
- Language is Logo dialect extended to support agents
- Mobile agents (turtles) move over a grid of stationary agents (patches).
- Link agents connect turtles to make networks, graphs, and aggregates.
- Double precision floating point math
- First-class function values (aka tasks, closures, lambda)

```
to setup
  clear-all
  set-default-shape turtles "square"
  ;; make some green trees
  ask patches with [(random-float 100) < density]
    [ set pcolor green ]
  ;; make a column of burning trees
  ask patches with [pxcor = min-pxcor]
    [ ignite ]
  ;; set tree counts
  set initial-trees count patches with [pcolor = green]
  set burned-trees 0
  reset-ticks
end

to go
  if not any? turtles  ;; either fires or embers
    [ stop ]
  ask fires
    [ ask neighbors4 with [pcolor = green]
```

Environment of NetLogo

- Command center for on-the-fly interaction
- Interface builder w/ buttons, sliders, switches, choosers, monitors, text boxes, notes, output area
- Info tab for annotating your model with formatted text and images
- Mobile agents (turtles) move over a grid of stationary agents (patches).
- Agent monitors for inspecting and controlling agents
- Export and import functions (export data, save and restore state of model, make a movie)
- BehaviorSpace, an open source tool used to collect data from multiple parallel runs of a model
- NetLogo 3D for modeling 3D worlds
- Headless mode allows doing batch runs from the command line



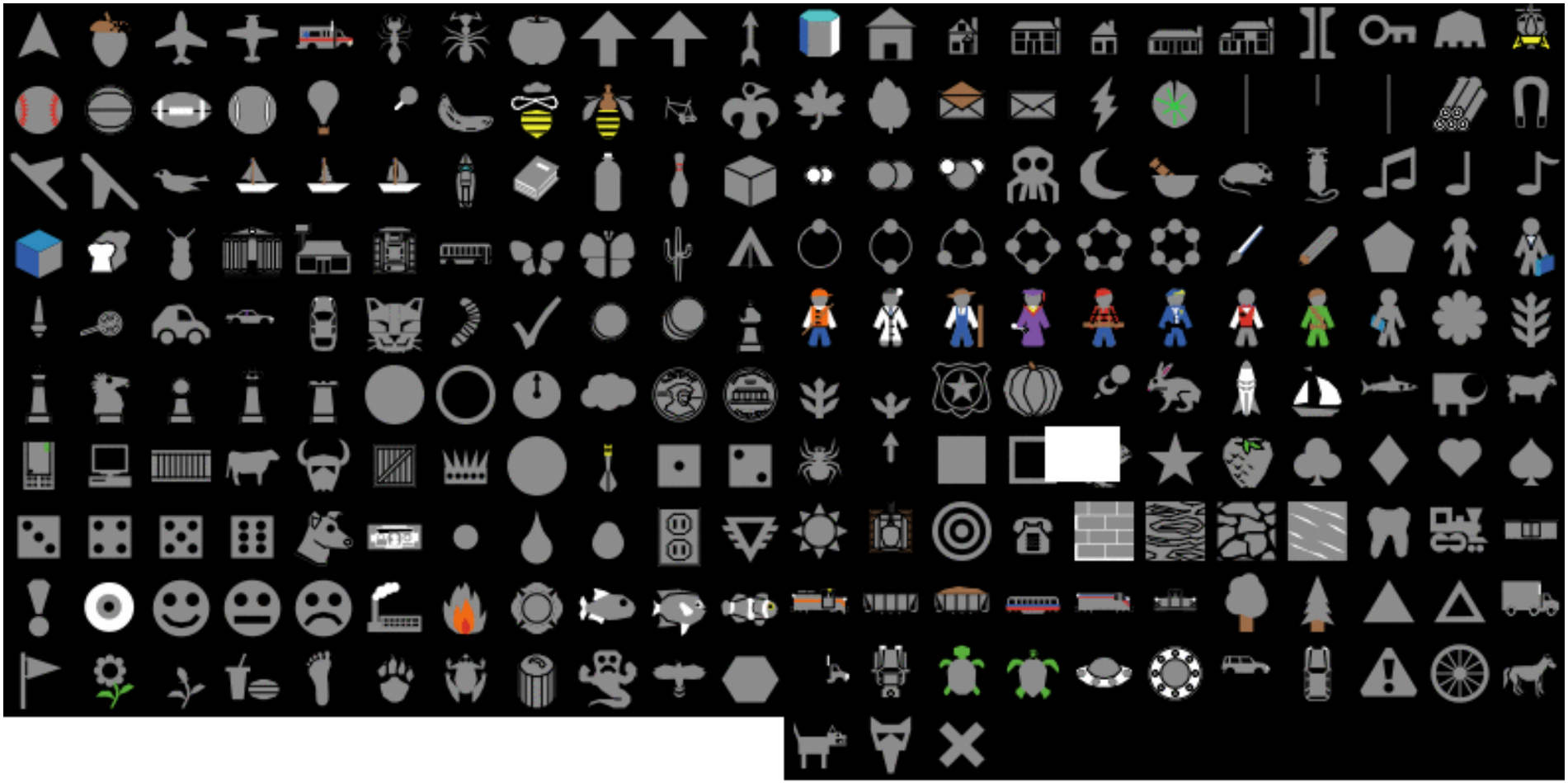
Building Blocks of Programming in NetLogo:Turtles

- **create-turtles** 2 (or crt 2)
- **ask turtle** 0
- **ask turtles**
- [**set color** blue]
- **clear-all**
- [**forward** 1] (or [fd 1])
- [**lt** 40] (or turn left 40 degree) set heading 45 & set heading heading + 40 & set heading towards turtle 1
- [**rt** 30] (or turn right 30 degree)
- [random 30] (choose a random number between 0 and 30)
- set-default-shape turtles "square" ask turtles with[xcor>0] [set shape "wolf"] & show shapes & ask turtles [set shape one-of shapes]

Example: Type and run line to line in Observer

clear-all create-turtles 3 ask turtles [forward 3]
ask turtle 2 [set color blue] ask turtle 0 [set color yellow rt 40]
ask turtles [fd random 5 lt random 40]

Default and New Shapes



To begin making shapes or links, choose **Turtle Shapes Editor** or **Link Shapes Editor** in the **Tools menu**.

Choose a shape and **edit** it as a new shape or Click **new** to draw your own shape.

Building Blocks of Programming in NetLogo: Patches

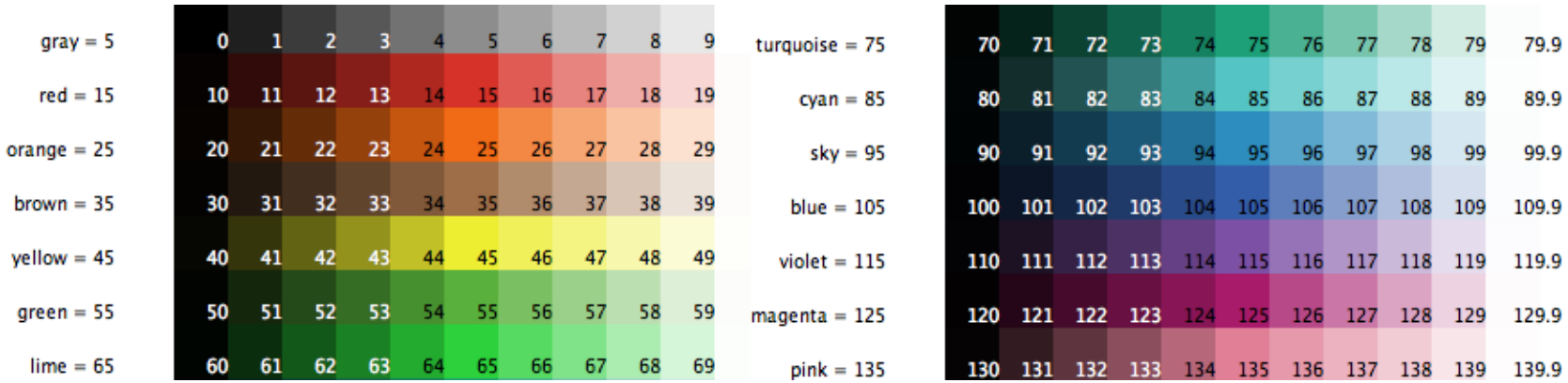
- min-pxcor , max-pxcor, min-pycor, and max-pycor (all are 16 r -16)
- pxcor and pycor both range from -16 to 16 (you can change them in UI)
- ask patches with [pxcor = min-pxcor][set pcolor green]
- ask patches with [pycor > 0][set pcolor red]
- ask patches with [pxcor > 0 and pycor > 0] [set pcolor red]
- ask patches with [(random-float 100) < 50][set pcolor blue]
- ask patches with [count turtles-here = 1] [set pcolor black]
- ask one-of turtles-here
- show count neighbors with [pcolor = 77]
- ask patches [set n count neighbors with [pcolor = yellow]]
- if any? neighbors with [any? turtles-here] neighbors neighbors4
- **agentset at-points [[x1 y1] [x2 y2] ...]** ask turtles at-points [[2 4] [1 2] [10 15]] [fd 1]
- uphill patch-variable & uphill4 patch-variable (also downhill, downhill4)

Ticks and Repeat

- **tick** Advances the tick counter by one and updates all plots.
- **reset-ticks** to start tick
- **repeat number [commands]** Ex pd repeat 36 [fd 1 rt 10]
- **foreach** foreach [1.1 2.2 2.6] [show (word ? " -> " round ?)]
- **stop** This agent exits immediately from the enclosing procedure, ask, or ask-like construct. if not any? turtles [stop]
- **while [reporter] [commands]** while [any? other turtles-here] [fd 1]
- ask turtles [print who]
- **sort-by reporter-task list** show sort-by < [3 1 4 2] & show sort-by [length ?1 < length ?2] ["Grumpy" "Doc" "Happy"]
- **to repeat a function with its button just choose forever**
- there is not for loop in NetLogo, Don't seek it

Variables, Strings and Plot

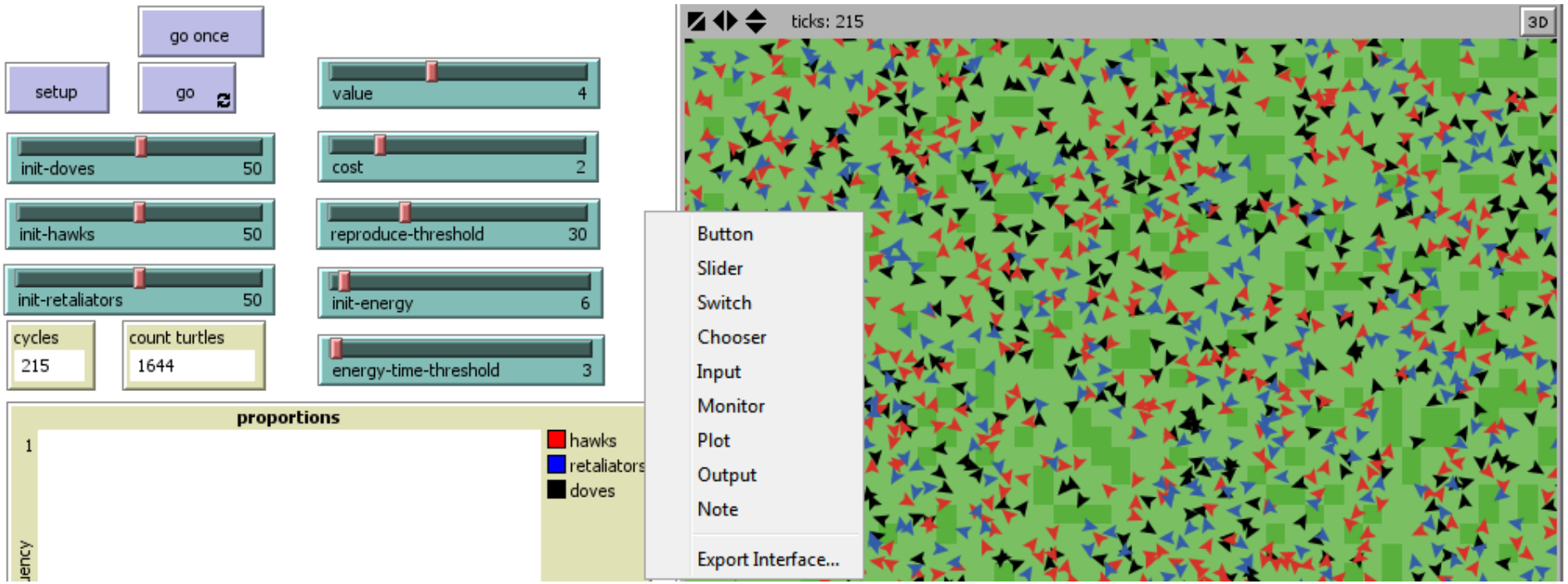
- **word value1 value2** Concatenates the inputs together show word "tur" "tle"
- set variable value set i 9
- to make a plot just by right click on main interface window and make a plot with some variables.
- **wait** repeat 10 [fd 1 wait 0.5]
- Shades of Colors



Input Variables

- One can take any input of user use v as variable and make a slider, input box ... just by right click on main interface window
- To build a slider just click right and choose your type, and type the functions

create-doves **init-doves** & create-hawks **init-hawks**



Function

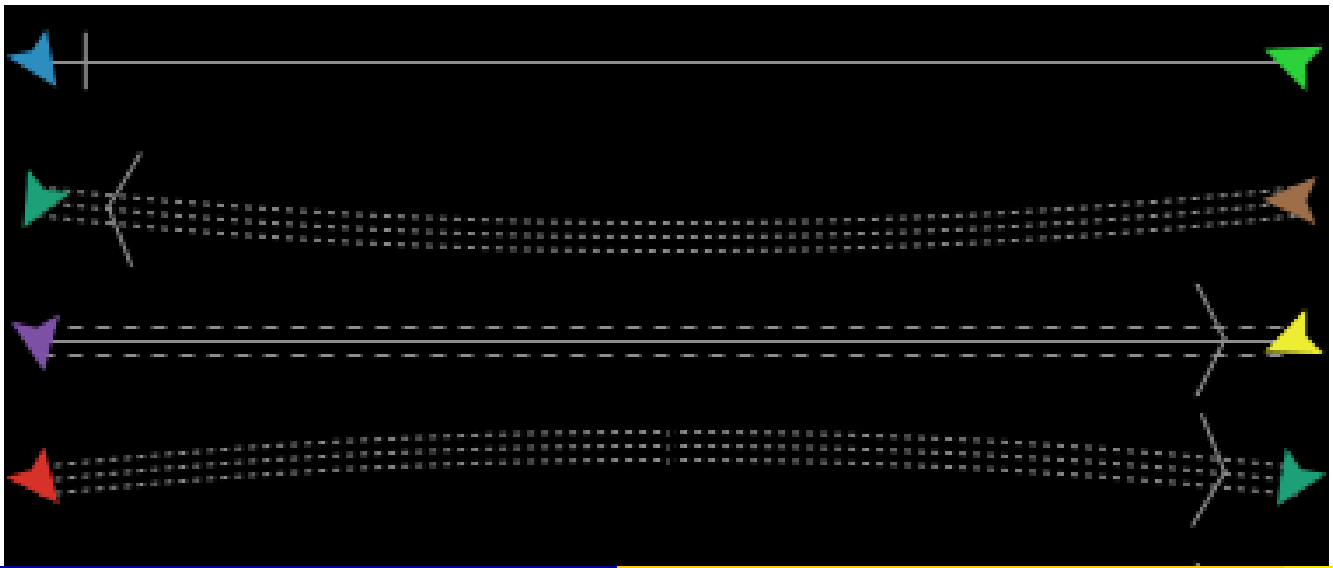
- **to setup**
  clear-all
  crt 500
  end
- **to circle [radius]**
  crt 100 [fd radius]
  end
- **to-report average** [a b]
  report (a + b) / 2
  end

Bread and mor advanced

- **breed [breeds breed]** breed [cats cat] breed [dogs dog] breed [hamsters hamster]
- **turtles-own [eyes legs]** cats-own [fur kittens] hamsters-own [fur cage] dogs-own [hair puppies]

Links

- **link end1 end2 <breed>end1 end2** (ask link 0 1 [set color green])
- **link-heading** (ask link 0 1 [print link-heading])
- **link-length** (ask link 0 1 [print link-length])
- **link-shapes** (show link-shapes)
- **links-own [var1 ...]** (links-own [traffic])
- **directed-link-breed [streets street]**
- **streets-own [cars bikes]**
- **Link Shape** (links>set shape "road")
- One can choose any shape or Create his/her own shape using Shape and Link Editor.



Pen Mode

- **pen-mode** possible values ("up", "down", and "erase".) (show pen-mode)
- One can use **pu, pd** and **pe**.
- When a turtle's pen is **down**, all movement commands cause lines to be drawn, including jump, setxy, and move-to.
- These commands are equivalent to setting the turtle variable "pen-mode" to "down" , "up", and "erase".
- **pen-size**

A Case Study: Dove Hawk

Code Info

- This is a model based on evolutionary game theory.
- Author Rick OGorman SUNY-Binghamton Binghamton NY 13902-6000 USA 2003
- Turtles wander from patch to patch in a somewhat random fashion
- Each move costs energy, but they can get energy from patches.
- If they arrive alone, then they get all the energy but if there is another turtle, then they get a payoff depending on thee type of turtle they are there with.
- If the energy of a turtle reaches a certain level, it reproduces asexually, and if its energy reaches zero, it dies.
- The reason that a turtle gets energy if alone is that otherwise, in situations where there is a net cost for hawks interacting with other hawks and there are two strategies, hawk and dove, then the hawks would become fixed, and then all die!!!
- Patches require a certain amount of time before they recover their resource value.
- This controls the population of turtles. Patches with resources available are green; they are a lighter color if their resources are not available.

Programming Procedure of Dove Hawk

globals [cycles]↵ breed [doves] ↵ breed [hawks] ↵ breed [retaliators]
turtles-own [turtle-energy xhere yhere trys] ↵ patches-own [energy energy-time]

to setup
set cycles 0 ↵ ask patches [set pcolor green set energy-time 1]
create-doves init-doves ↵ create-hawks init-hawks ↵ create-retaliators
init-retaliators
ask doves [set color black] ↵ ask hawks [set color red] ↵ ask retaliators [set
color blue]
ask turtles [setxy random world-width random world-height ↵ set turtle-energy
init-energy]
do-plot ↵ reset-ticks
end

Programming Procedure of Dove Hawk

```
to go
ask patches with [energy-time >= energy-time-threshold and pcolor = green + 1]
[set pcolor green]
ask turtles [move] get-energy ↵ ask turtles [reproduce perish]
if not any? turtles [do-plot-zero stop]
ask patches [set energy-time energy-time + 1] ↵ tick
 end
```

* * * * * * * * * * * **

```
to move
set xhere xcor ↵ set yhere ycor ↵ set trys 1
while [xhere = xcor and yhere = ycor and trys < 9]
[rt random 46 - random 46 ↵ if count turtles-at dx dy < 2
[ fd 1] ↵ set trys trys +1] ↵ set turtle-energy turtle-energy - 1
end
```

and ...

Any Question?