PyCX

A Python-Based Simulation Code Repository for Complex Systems Education

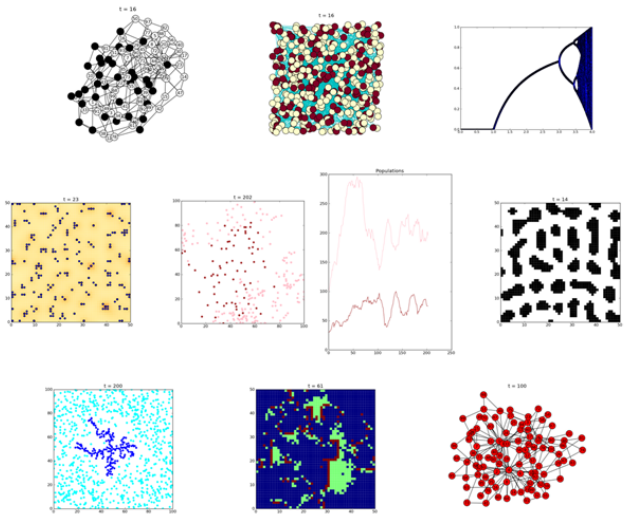
Hiroki Sayama
sayama@binghamton.edu
http://pycx.sf.net/

This Tutorial Is For…

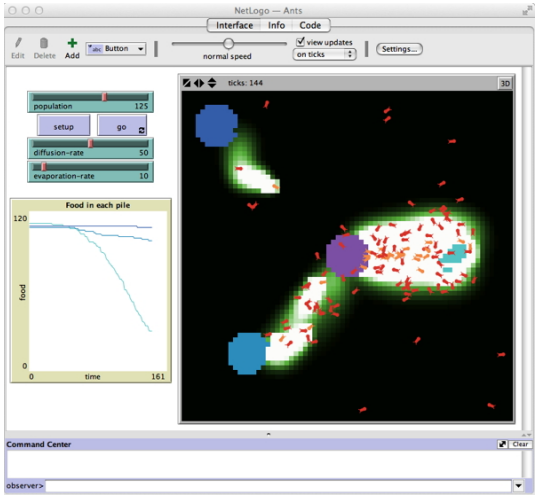
- Educators who teach complex systems-related courses and thus need simple, easy-to-understand examples of complex systems simulations
- Students and researchers who want to learn basics of writing complex systems simulations themselves

What is PyCX?

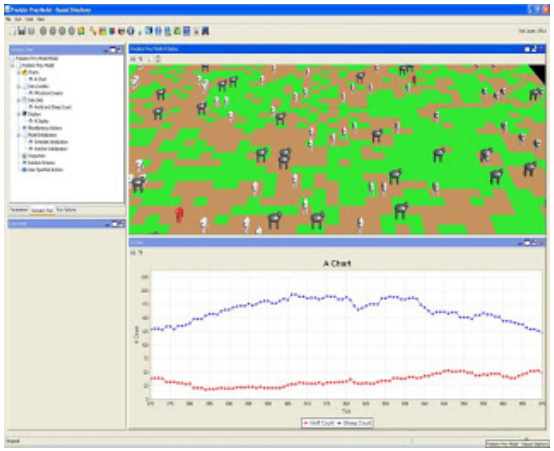
- "**Py**thon-based **C**omple**X** systems simulations"
  - Online repository (**http://pycx.sf.net/**) of sample codes of complex systems simulations written in plain Python
    - Iterative maps
    - Cellular automata
    - Dynamical networks
    - Agent-based models



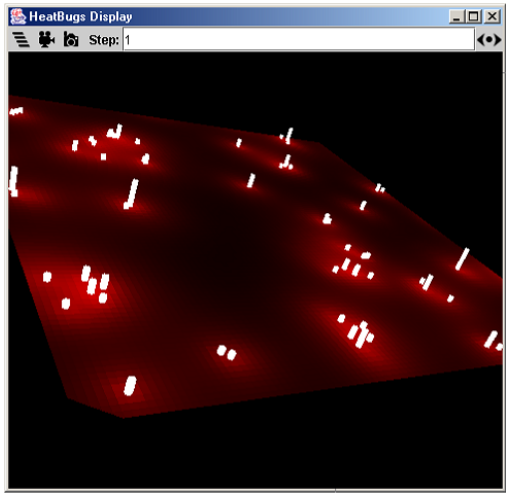
Yet Another Simulation Software?



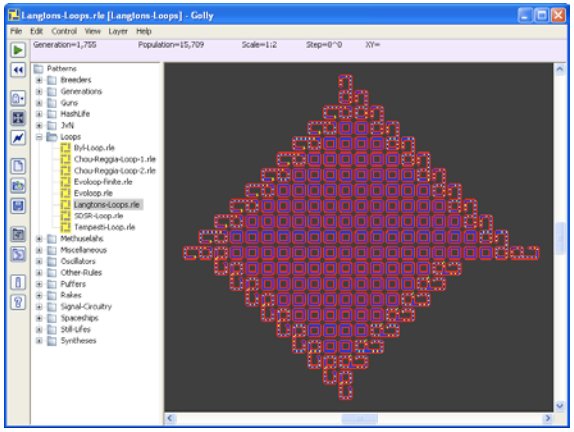
NetLogo



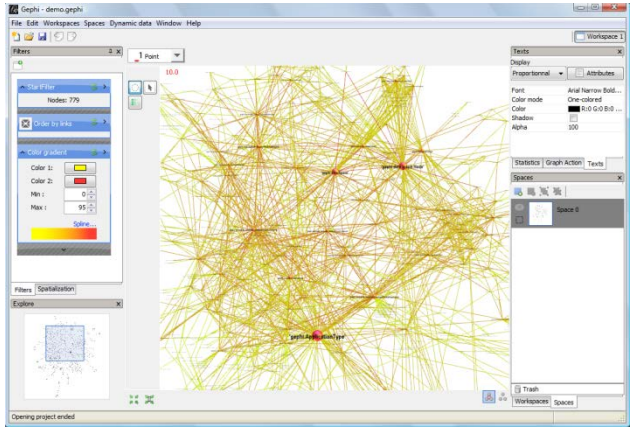
RePast



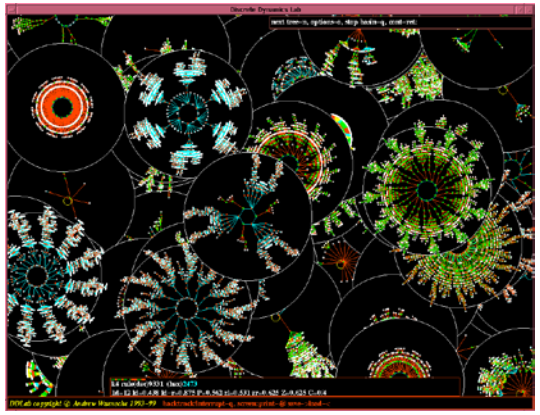
MASON



Golly



Gephi



DDLab

No, It's Not Software

- PyCX is nothing more than a collection of very simple Python sample codes
- It can run on plain Python

Problems in Teaching Complex Systems with Pre-Built Software

- Students would not advance general technical skills
- Choice of software varies greatly from discipline to discipline
- Students would not have access to details of model implementation
- Use of existing software puts unrecognized limitations to "creativity" in modeling

PyCX As a Potential Solution

- Students can learn generalizable technical skills (i.e., programming in Python)
- Python has become very popular in a number of scientific domains
- Students will be able to see every detail of their model by coding it themselves
- Using a programming language itself as a modeling tool allows open-ended modeling

PyCX Philosophy

- **Emphasized:**
  - **Simplicity**
  - **Readability**
  - **Generalizability**
  - **Pedagogical values**
- Not emphasized:
  - Computational speed
  - Efficiency
  - Maintainability

PyCX Coding Style

- One simulation model, one .py file
- Same three-part structure for all dynamic simulations
  - Initialization, visualization, updating
- No object-oriented programming
- Frequent use of global variables

What To Do

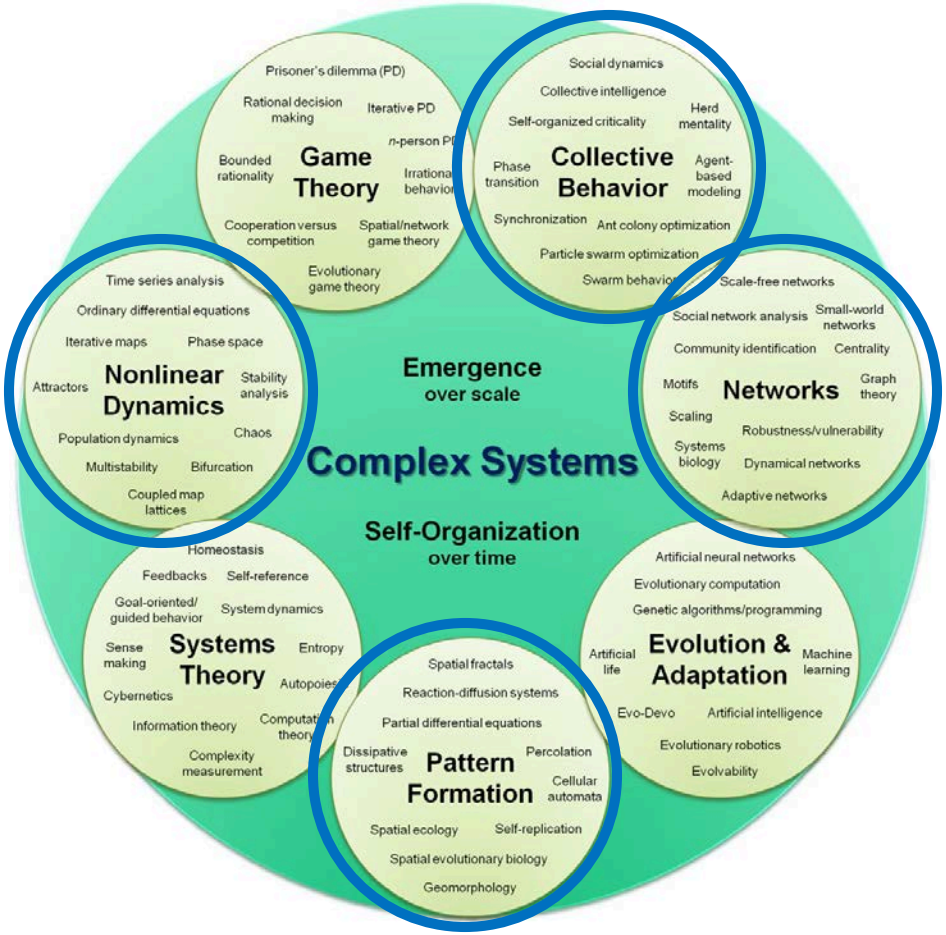
1. Install Python 2.7, NumPy, SciPy, matplotlib and NetworkX
2. Download a PyCX sample code of your interest
3. Run it
4. Read it
5. Change it as you like

Python Installation and Basics

- If you don't have Python on your laptop, get installers online or from the flash drives circulated in the room and install it now

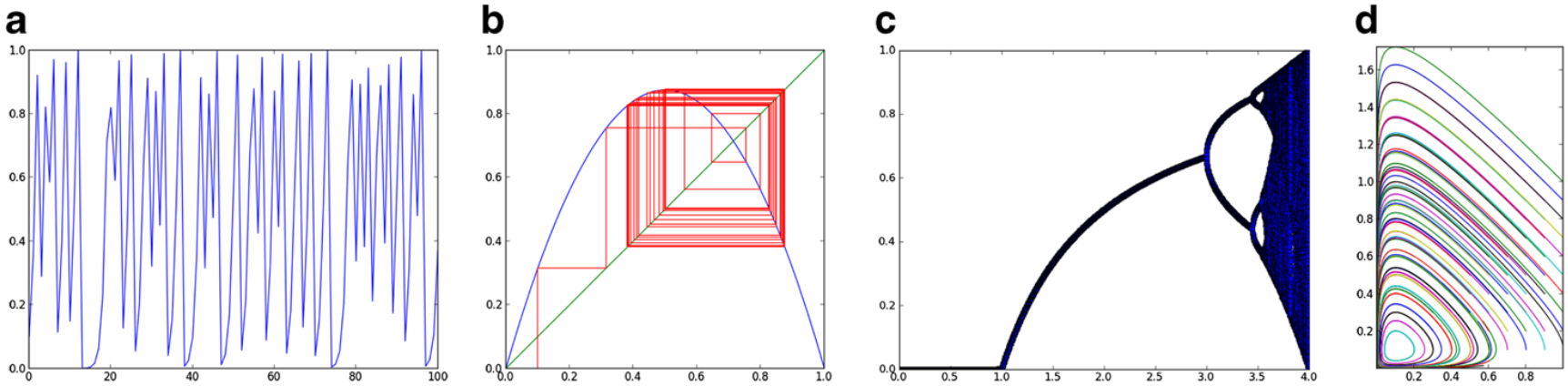
Sample Simulations

- Dynamical systems
- Cellular automata
- Dynamical networks
- Agent-based models



Dynamical Systems

- Logistic map
- Cobweb plot
- Bifurcation diagram
- Lotka-Volterra equations

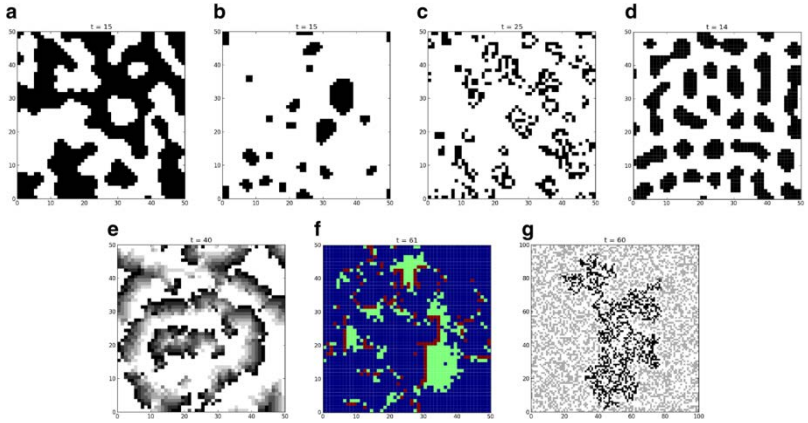


Dynamic, Interactive Simulations

- Use "**pycxsimulator.py**" to produce simple interactive GUI
  - New GUI by Przemyslaw Szufel & Bogumil Kaminski at the Warsaw School of Economics!!
  - The file should exist in the same folder where your simulator code is located
  - See "**realtime-simulation-template.py**" for how to use it

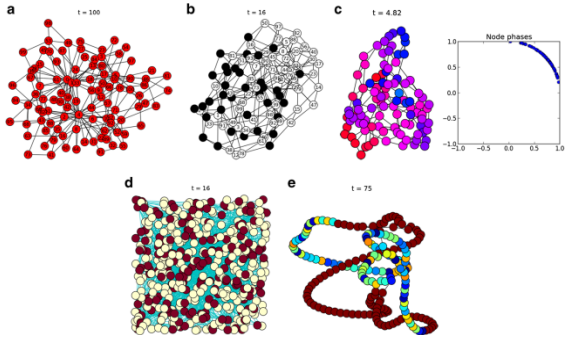
Cellular Automata

- Local majority rule
- Droplet rule
- Game of Life
- Turing pattern formation
- Excitable media
- Host-pathogen interaction
- Forest fire
- Spread of rumor
- Schelling's segregation model (technically ABM, but…)



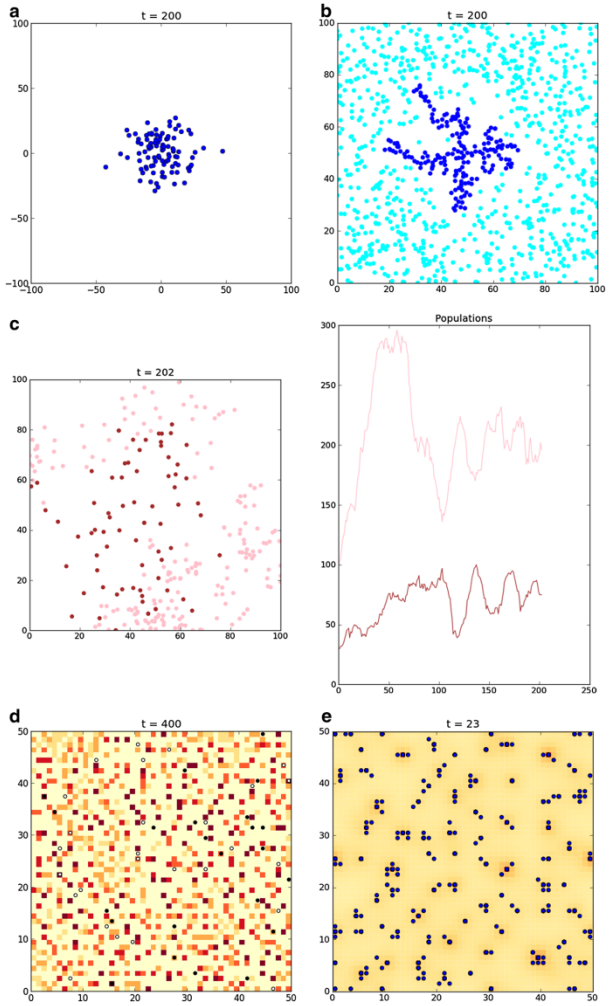
Dynamical Networks

- Basic network construction and analysis
- Network growth by preferential attachment
- Local majority rule on a network
- Synchronization on a network
- Random walk on a network
- Cascade of failures on a network
- Voter model of opinion formation on a network
- Epidemics on a network
- Epidemics on a network with adaptive link cutting



Agent-Based Models

- Random walk of particles
- Diffusion-limited aggregation
- Predator-prey ecosystem
- Garbage collection by ants
- Aggregation of ants via pheromone-based communication



Actual Use in Classroom

- Binghamton University Graduate Courses
  - BME-510: Modeling Complex Biological Systems (2009, 2010)
  - SSIE-518X/523: Collective Dynamics of Complex Systems (2011-)
  - BME-523X/523: Dynamics of Complex Networks (2012-)
- New England Complex Systems Institute Summer/ Winter Schools
  - CX 202: Complex Systems Modeling and Networks (2008-)
  - CX 102: Computer Programming and Complex Systems (2010-)
- NWO (Netherlands Organisation for Scientific Research) Complexity Winter School (2011)
- NetSci High Summer Workshop (2012-)

Typical Instruction Methods

- BYOL course
- **First 3~6 hours: Python installation and basics**
- **Rest: Modular instructions, using each sample code as a curricular module**

Instruction of a Module (30~60 min)

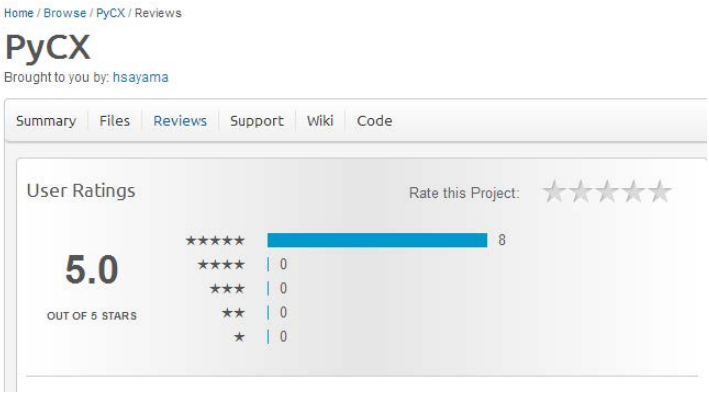
1. Describe the key concepts of the phenomenon being modeled, as well as the basic model assumptions
2. Run the simulation sample code, show the results and have a brief discussion on the observations
3. Open the code in an editor and give a line-by-line walk-through, explaining how the model is implemented in detail and addressing any technical questions as needed

Instruction of a Module (30~60 min)

4. Have a couple of in-class exercises that require students to understand and then modify the code to implement some model variations
5. Summarize the learning experience of the curricular unit and have open Q&A's and/or try further model extensions

Testemonials

- 100% "thumbs-up" user rating scores on SF



- Very positive comments from those who took courses with PyCX or who used PyCX
- Several publications of papers on Python-based computer simulation by students and faculty who took courses with PyCX

Further Developments

- **WPyCX** – a Warsaw School of Economics version of PyCX written in Python 3.3 has been released just last night!!
  - Thanks to Przemyslaw Szufel and Bogumil Kaminski at Warsaw School of Economics

For More Info

http://pycx.sf.net/

Sayama, H. (2013) PyCX: A Python-based simulation code repository for complex systems education, *Complex Adaptive Systems Modeling*, 1:2, 2013. (open access) http://www.casmodeling.com/content/1/1/2