Complex Systems Made Simple

1. **Introduction**

2. **A Complex Systems Sampler**

3. **Commonalities**

4. **NetLogo Tutorial**

   a. What is NetLogo?

   b. Graphical interface

   c. Programming concepts

   d. Termites $\rightarrow$ NetLogo project?

Complex Systems Made Simple

1. Introduction

2. A Complex Systems Sampler

3. Commonalities

4. **NetLogo Tutorial**

    a.   **What is NetLogo?** *Modeling, flash history, the world*

    b.   Graphical interface

    c.   Programming concepts

    d.   Termites $\rightarrow$ NetLogo project?

4. NetLogo Tutorial
a. What is NetLogo? – *Modeling complex systems*

✓ **programmable modeling environment for simulating natural and social phenomena**

- well suited for complex system modeling that evolves over time

- hundreds or thousands of independent agents operating concurrently

- exploring the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from the interaction of many individuals

4. NetLogo Tutorial

a. What is NetLogo? – *Modeling complex systems*

✓ **easy-to-use application development environment**

- opening simulations and playing with them

- creating custom models: quickly testing hypotheses about self-organized systems

- models library: large collection of pre-written simulations in natural and social sciences that can be used and modified

- simple scripting language

- user-friendly graphical interface

4. NetLogo Tutorial
a. What is NetLogo? – *Flash history*

➤ **LOGO (Papert & Minsky, 1967)**

- theory of education based on Piaget's constructionism ("hands-on" creation and test of concepts)
- simple language derived from LISP
- turtle graphics and exploration of "microworlds"

➤ **StarLogo (Resnick, 1991), MacStarLogo, StarLogoT**

- agent-based simulation language
- exploring the behavior of decentralized systems through concurrent programming of 100s of turtles

➤ **NetLogo (Wilensky, 1999)**

- further extending StarLogo (continuous turtle coordinates, cross-platform, networking, etc.)
- most popular today (growing cooperative library of models)

René Doursat: "Complex Systems Made Simple"

4. NetLogo Tutorial
a. What is NetLogo? – *The world of NetLogo*

✓ NetLogo is a 2-D world made of 3 kinds of agents:

- *patches* – make up the background or "landscape"

- *turtles* – move around on top of the patches

- *the observer* – oversees everything going on in the world

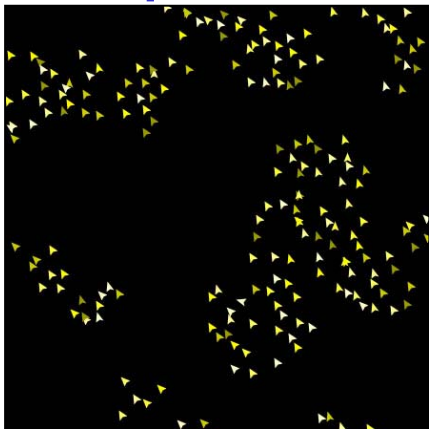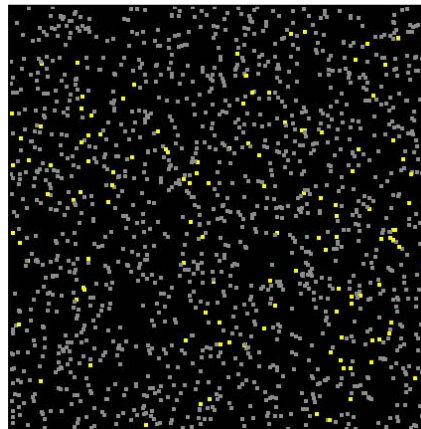➢ examples of patch-only models

B-Z reaction

Fur

4. NetLogo Tutorial
a. What is NetLogo? – *The world of NetLogo*
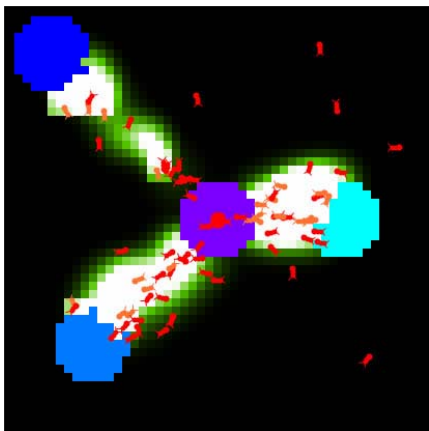
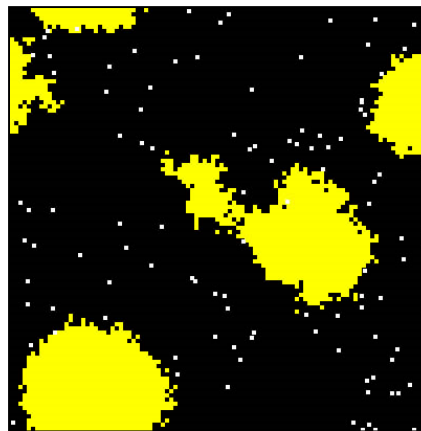➤ examples of turtle-only models

Flocking

Fireflies

➤ examples of patch-&-turtle models

Ants

Termites

Complex Systems Made Simple

1. **Introduction**

2. **A Complex Systems Sampler**

3. **Commonalities**

4. **NetLogo Tutorial**

    a. What is NetLogo?

    b. **Graphical interface:** *Controls, settings, views*

    c. Programming concepts

    d. Termites → NetLogo project?

4. NetLogo Tutorial

b. Graphical interface – *Controls, Settings, Views*

✓ **controls** (BLUE) – allow to run and control the flow of execution
  - buttons
  - command center

✓ **settings** (GREEN) – allow to modify parameters
  - sliders
  - switches
  - choosers

✓ **views** (BEIGE) – allow to display information
  - monitors
  - plots
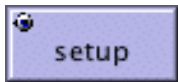 - output text areas
  - graphics window

4. NetLogo Tutorial
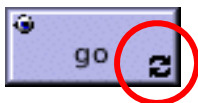## b. Graphical interface – *Controls*

✓ **controls** (BLUE) – allow to run and control the flow of execution
  - buttons
  - command center

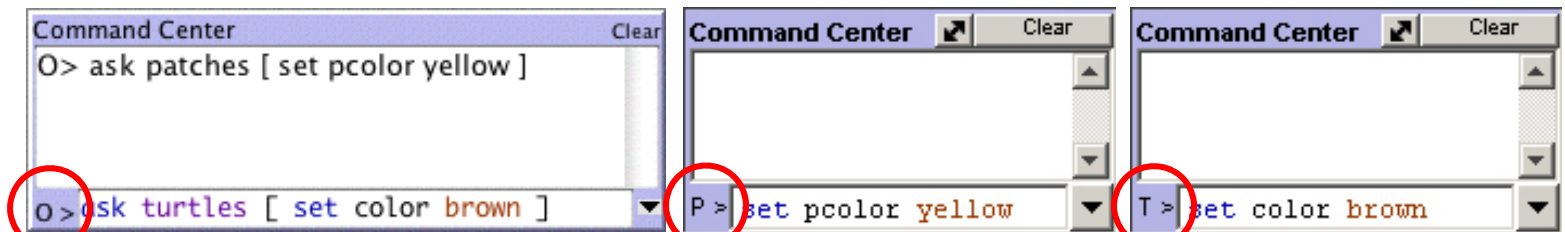➤ **buttons** – initialize, start, stop, step through the model

  - "once" buttons execute one action (one piece of code)
  - "forever" buttons repeat the same action (the same piece of code) until pressed again

➤ **command center** – ask observer, patches or turtles to execute specific commands "on the fly"

- O>**ask patches [*commands*]** ⟺ P>*commands*
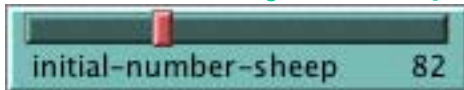- O>**ask turtles [*commands*]** ⟺ T>*commands*

4. NetLogo Tutorial
b. Graphical interface – *Settings*

✓ **settings** (GREEN) – allow to modify parameters
- sliders
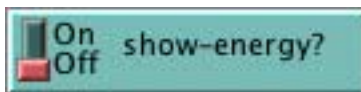- switches
- choosers

➤ **sliders** – adjust a quantity from *min* to *max* by an *increment*
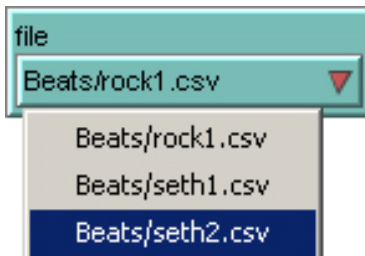
initial-number-sheep   82

- `initial-number-sheep = 82`

➤ **switches** – set a Boolean variable (true/false)

On/Off show-energy?

- `show-energy? = false`

➤ **choosers** – select a value from a list

file
Beats/rock1.csv ▼
Beats/rock1.csv
Beats/seth1.csv
Beats/seth2.csv

- `file = "Beats/seth2.csv"`

4. NetLogo Tutorial
b. Graphical interface – *Views*

✓ **views** (BEIGE) – allow to display information
- monitors
- plots
- output text areas
- graphics window

➢ **monitors** – display the current value of variables

time-ticks	sheep	wolves	grass / 4
0	0	0	0
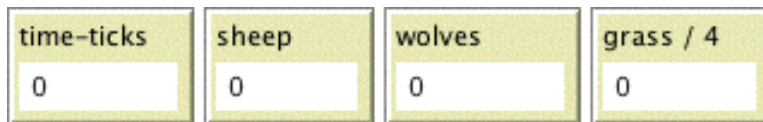
➢ **plots** – display the history of a variable's value

Turtles at Peaks / Pens / 39.6 / Number / 0 / 0 / Time / 33.8
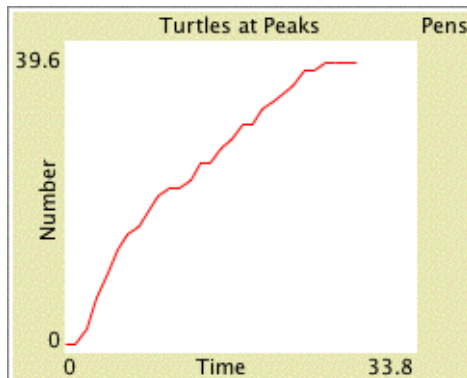
➢**output text areas** – log text info

Last turn your partner defected
Choose your action
Last turn your partner cooperated
Choose your action
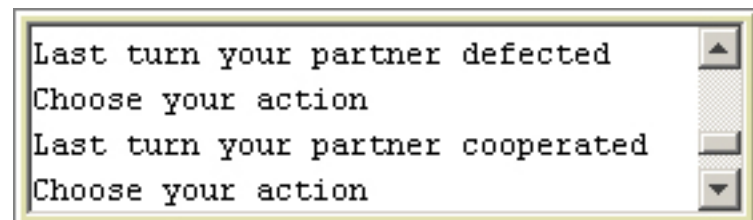
4. NetLogo Tutorial
b. Graphical interface – *Views*

➢ **graphics window** – the main view of the 2-D NetLogo world

adjust speed

turn turtle shapes on/off

freeze/unfreeze display

change width & height
in number of patches

right-click brings up
turtle/patch inspector

turtle 0	
who	0
color	15.0
heading	90.0
xcor	-14.237661106278512
ycor	0.0
shape	"car"
pen-down?	false
label	
label-color	9.9999
breed	turtles
hidden?	false
size	1.0
speed	9.1
speed-limit	1
speed-min	0

Complex Systems Made Simple

1. Introduction

2. A Complex Systems Sampler

3. Commonalities

4. NetLogo Tutorial

    a. What is NetLogo?

    b. Graphical interface

    c. **Programming concepts:** *Agents, procedures, variables, ask, agentsets, breeds, synchronization*

    d. Termites → NetLogo project?

4. NetLogo Tutorial
c. Programming concepts – *Agents*

✓ **agents** – carry out their own activity, all simultaneously
- patches
- turtles
- observer

➢ **patches**
- don't move, form a 2-D wrap-around grid
- have *integer* coordinates (`pxcor`, `pycor`)

➢ **turtles**
- move on top of the patches, not necessarily in their center
- have *decimal* coordinates (`xcor`, `ycor`) and orientation (`heading`)

➢ **observer**
- can create new turtles
- can have read/write access to all the agents and variables

4. NetLogo Tutorial
c. Programming concepts – *Procedures*

➢ **commands**

- actions for the agents to carry out ("void" functions)
- example:

```
to setup
  ca
  crt 10
end
```

- example with 2 input arguments:

```
to draw-polygon [ num-sides size ]
  pd
  repeat num-sides
    [ fd size
      rt (360 / num-sides) ]
end
```

4. NetLogo Tutorial
c. Programming concepts – *Procedures*

➢ reporters

- report a result value (functions with return type)
- example with 1 input argument:

```
to-report absolute-value [ number ]
  ifelse number >= 0
    [ report number ]
    [ report 0 - number ]
end
```

➢ primitives

- *built-in* commands or reporters (language keywords)
- some have an abbreviated form: **create-turtles** ⟺ **crt**, **clear-all** ⟺ **ca**, etc.

✓ procedures

- *custom* commands or reporters (user-made)

4. NetLogo Tutorial
c. Programming concepts – *Variables*

✓ **variables** – places to store values (such as numbers or text)
- global variables
- turtle & patch variables
- local variables

➢ **global variables**
- only one value for the variable
- every agent can access it

➢ **turtle & patch variables**
- each turtle/patch has its own value for every turtle/patch variable

➢ **local variables**
- defined and accessible only inside a procedure
- scope = narrowest square brackets or procedure itself

René Doursat: "Complex Systems Made Simple"

4. NetLogo Tutorial
c. Programming concepts – *Variables*

➢ **built-in variables**

- ex. of built-in turtle variables: `color`, `xcor`, `ycor`, `heading`, etc.
- ex. of built-in patch variables: `pcolor`, `pxcor`, `pycor`, etc.

➢ **custom variables**

- defining global variables:

```
global [ clock ]
```

- defining turtle/patch variables:

```
turtles-own [ energy speed ]
patches-own [ friction ]
```

- defining a local variable:

```
to swap-colors [ turtle1 turtle2 ]
  let temp color-of turtle1
  ...
```

4. NetLogo Tutorial
c. Programming concepts – *Variables*

➢ **setting variables**

- setting the color of all turtles:

  ```
  ask turtles [ set color red ]
  ```

- setting the color of all patches:

  ```
  ask patches [ set pcolor red ]
  ```

- setting the color of the patches under the turtles:

  ```
  ask turtles [ set pcolor red ]
  ```

- setting the color of one turtle:

  ```
  ask turtle 5 [ set color green ]
  ```

  or:

  ```
  set color-of turtle 5 red
  ```

- setting the color of one patch:

  ```
  ask patch 2 3 [ set pcolor green ]
  ```

4. NetLogo Tutorial
c. Programming concepts – *Ask*

✓ **"ask"** – specify commands to be run by turtles or patches
- asking all turtles:

```
ask turtles [ fd 50 ... ]
```
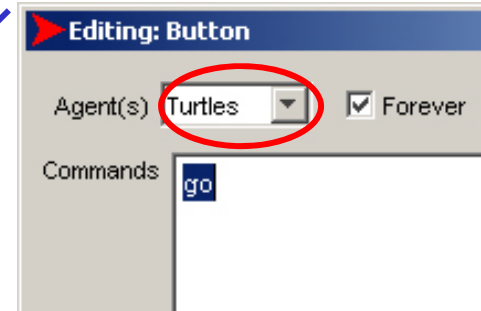
- asking all patches:

```
ask patches [ diffuse ... ]
```

- asking one turtle:

```
ask turtle 5 [ ... ]
```

✓ can be factored out in button specs

```
to go [
    ask turtles [ ... ]
]
```


➢ observer code *cannot* be inside any "ask" block
- ex: creating 100 turtles:

```
crt 100
```

4. NetLogo Tutorial

c. Programming concepts – *Agentsets*

✓ **agentset** – definition of a subset of agents (*not* a keyword)
- all red turtles:

```
turtles with [ color = red ]
```

- all red turtles on the patch of the current caller (turtle or patch):

```
turtles-here with [ color = red ]
```

- all patches on right side of screen:

```
patches with [ pxcor > 0 ]
```

- all turtles less than 3 patches away from caller (turtle or patch):

```
turtles in-radius 3
```

- the four patches to the east, north, west, and south of the caller:

```
patches at-points [[1 0] [0 1] [-1 0] [0 -1]]
```

- shorthand for those four patches:

```
neighbors4
```

4. NetLogo Tutorial
c. Programming concepts – *Agentsets*

➢ using agentsets

- ask such agents to execute a command

```
ask <agentset> [ ... ]
```

- check if there are such agents:

```
show any? <agentset>
```

- count such agents:

```
show count <agentset>
```

- example: remove the richest turtle (with the maximum "assets" value):

```
ask max-one-of turtles [ sum assets ] [ die ]
```

singleton agentset containing the richest turtle

4. NetLogo Tutorial
c. Programming concepts – *Breeds*

✓ **breed** – a "natural" kind of agentset (other species than turtles)

- example:

```
breed [ wolves sheep ]
```

- a new breed comes with automatically derived primitives:

```
create-<breed>
create-custom-<breed>
<breed>-here
<breed>-at
...
```

- the breed is a turtle variable:

```
ask turtle 5 [ if breed = sheep ... ]
```

- a turtle agent can change breed:

```
ask turtle 5 [ set breed sheep ]
```
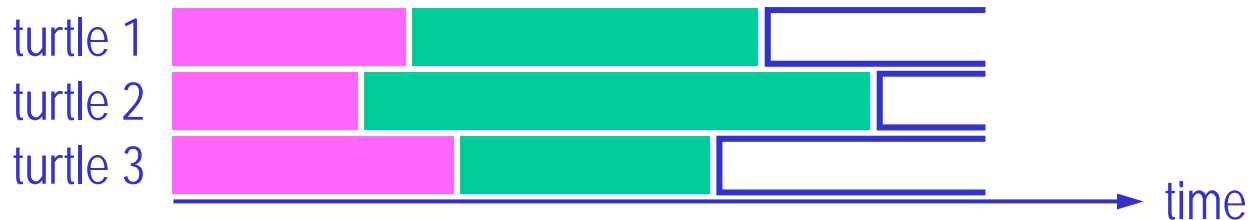
4. NetLogo Tutorial

c. Programming concepts – *Synchronization*

✓ agents run in parallel (each agent is an independent thread)
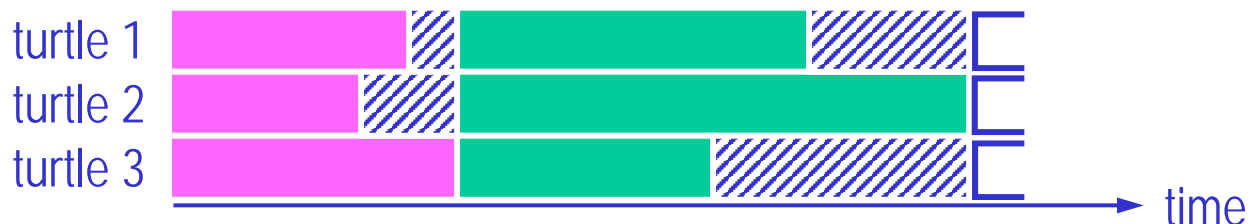
  - asynchronous commands:

  ```
  ask turtles [ fd random 10
                do-calculation ... ]
  ```


turtle 1
turtle 2
turtle 3
time

✓ agent threads wait and "join" at the end of a block

  - synchronous commands:

  ```
  ask turtles [ fd random 10 ]
  ask turtles [ do-calculation ] ...
  ```
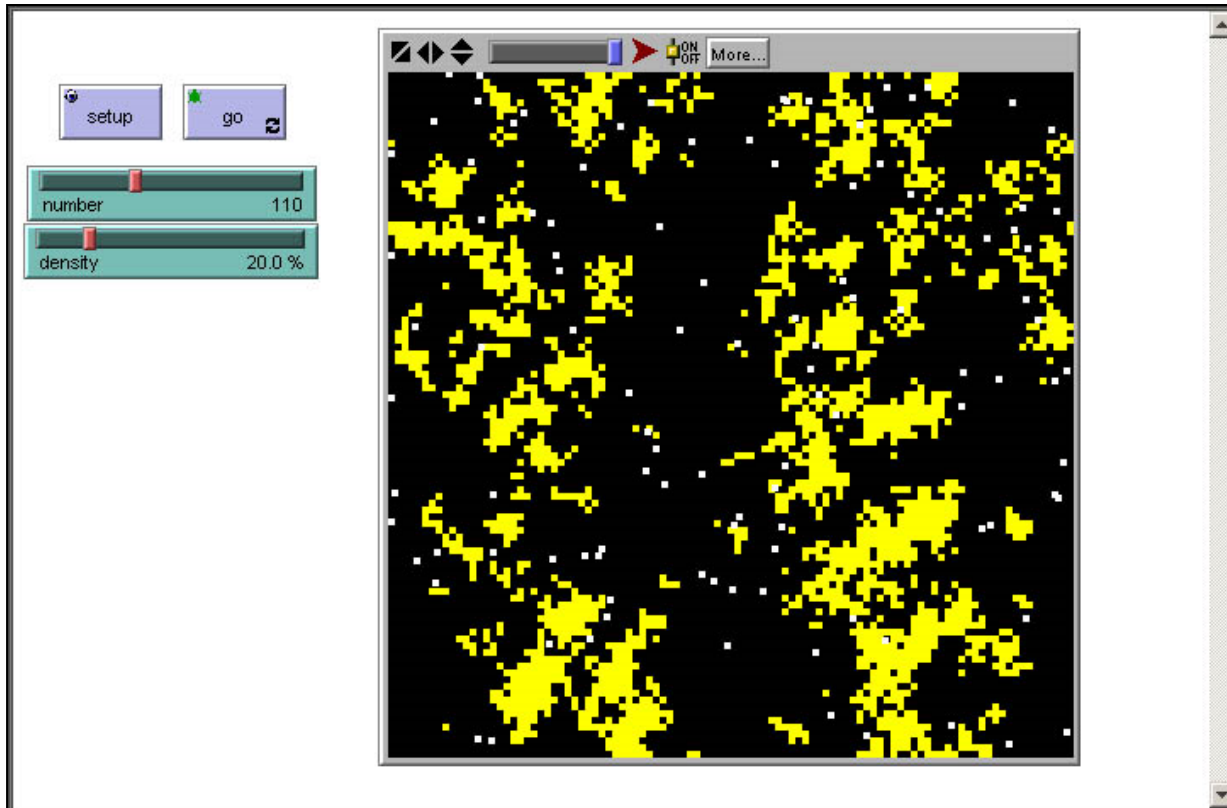

turtle 1
turtle 2
turtle 3
time

René Doursat: "Complex Systems Made Simple"

Complex Systems Made Simple

1. Introduction

2. A Complex Systems Sampler

3. Commonalities

4. NetLogo Tutorial

    a. What is NetLogo?

    b. Graphical interface

    c. Programming concepts

    d. Termites: *Interface, setup, go, full code, adding a plot*

René Doursat: "Complex Systems Made Simple"

4. NetLogo Tutorial
c. Termites – *Interface*

✓ build interface

➢ two buttons

- **setup**
  observer, once

- **go**
  turtles, forever

➢ two sliders

- **number**
  $1 \rightarrow 300$ (1)

- **density**
  $0 \rightarrow 100\%$ (1)

4. NetLogo Tutorial
c. Termites – *Setup*

✓ randomly strew yellow wood chips (patches) with given density

```
to setup-chips
    ask patches [ if random-float 100 < density
                      [ set pcolor yellow ] ]
end
```

✓ randomly position given number of white termites (turtles)

```
to setup-termites
    create-turtles number
    ask turtles [ set color white
                      setxy random-xcor
                            random-ycor ]
end
```

✓ setup all

```
to setup ca setup-chips setup-termites
end
```

4. NetLogo Tutorial
c. Termites – *Go*

✓ termites (turtles) follow 3 rules:

1. look around for a wood chip and pick it up

2. look around for a pile of wood chips

3. look around for an empty spot in the pile and drop off the chip

```
to go                ; turtle code
  pick-up-chip
  find-new-pile
  drop-off-chip
end
```

4. NetLogo Tutorial
c. Termites – *Go: explore*

✓ termites (turtles) explore the environment through random walk

```
to explore
  fd 1
  rt random-float 50
  lt random-float 50
end
```

```
to explore
  fd 1
    rt random-float 50
        - random-float 50
end
```

4. NetLogo Tutorial
c. Termites – *Go: pick up chip*

✓ find a wood chip, pick it up and turn orange (recursive versions)

```
to pick-up-chip
   ifelse pcolor = yellow
     [ set pcolor black
       set color orange ]
     [ explore
       pick-up-chip ]
end
```

```
to pick-up-chip
   if pcolor = yellow
     [ set pcolor black
       set color orange stop ]
   explore
   pick-up-chip
end
```

→ nonrecursive version

```
to pick-up-chip
   while [ pcolor != yellow ]
     [ explore ]
   set pcolor black
   set color orange
end
```

4. NetLogo Tutorial
c. Termites – *Go: find new pile*

✓ find a new pile of chips (recursive versions)

```
to find-new-pile
  if pcolor != yellow
    [ explore
      find-new-pile ]
end
```

```
to find-new-pile
  if pcolor = yellow [ stop ]
  explore
  find-new-pile
end
```

→ nonrecursive version

```
to find-new-pile
  while [ pcolor != yellow ]
    [ explore ]
end
```

4. NetLogo Tutorial
c. Termites – *Go: drop off chip*

✓ find an empty spot, drop off chip and get away (recursive versions)

```
to drop-off-chip
   ifelse pcolor = black
     [ set pcolor yellow
       set color white
       fd 20 ]
     [ explore
       drop-off-chip ]
end
```

```
to drop-off-chip
   if pcolor = black
     [ set pcolor yellow
       set color white
       fd 20 stop ]
   explore
   drop-off-chip
end
```

→ nonrecursive version

```
to drop-off-chip
   while [ pcolor != black ]
     [ explore ]
   set pcolor yellow
   set color white     fd 20
end
```

4. NetLogo Tutorial
c. Termites – *Full code*

```
to setup
  ca
  setup-chips
  setup-termites
end

to setup-chips
  ask patches [
    if random-float 100 < density
      [ set pcolor yellow ] ]
end

to setup-termites
  create-turtles number
  ask turtles [
    set color white
    setxy random-float screen-size-x
          random-float screen-size-y ]
end

to explore
  fd 1
  rt random-float 50
  lt random-float 50
end
```

```
to go                 ; turtle code
  pick-up-chip
  find-new-pile
  drop-off-chip
end

to pick-up-chip
  while [ pcolor != yellow ]
    [ explore ]
  set pcolor black
  set color orange
end

to find-new-pile
  while [ pcolor != yellow ]
    [ explore ]
end

to drop-off-chip
  while [ pcolor != black ]
    [ explore ]
  set pcolor yellow
  set color white
  fd 20
end
```
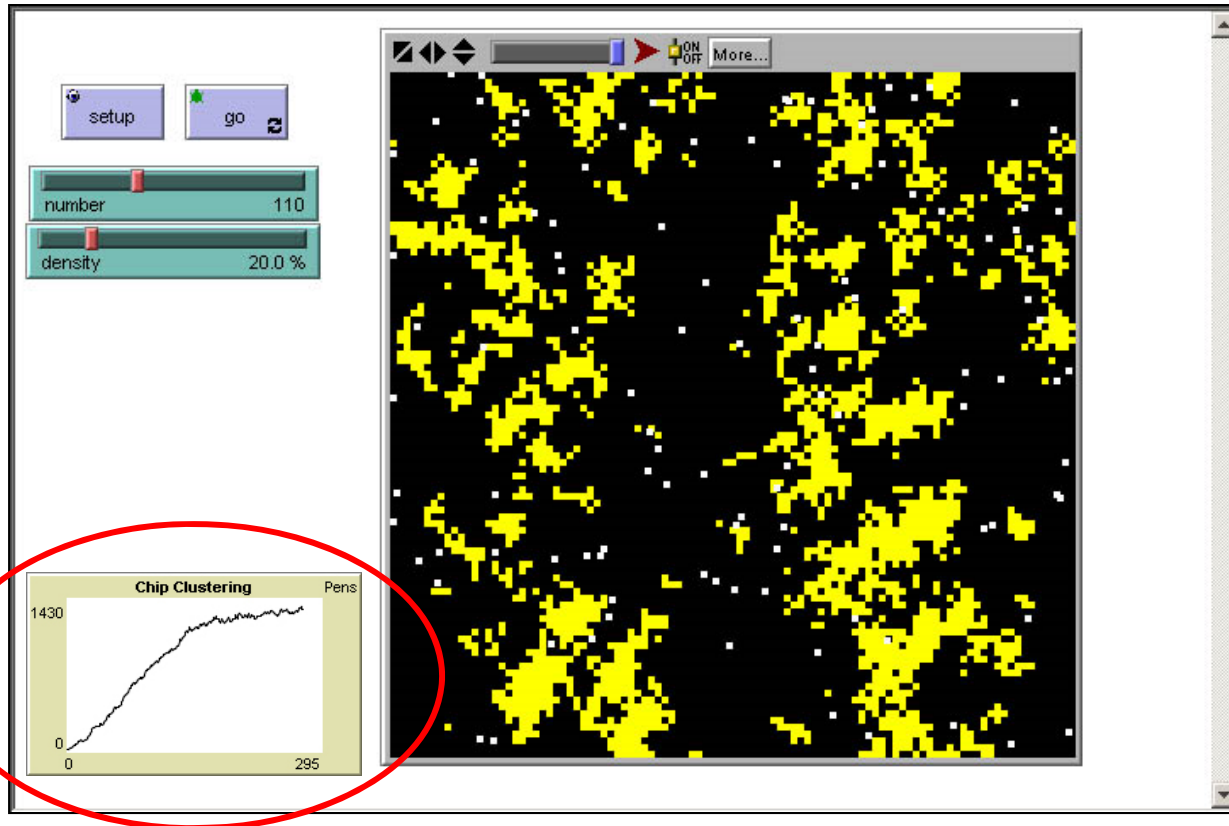
4. NetLogo Tutorial
c. Termites – *Adding a plot*

➤ one plot

- "Chip Clustering"
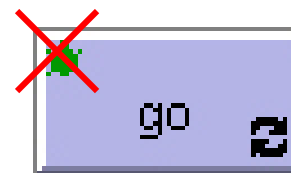
4. NetLogo Tutorial
c. Termites – *Adding a plot*

✓ plotting

```
to draw-plot
   set-current-plot "chip clustering"
   plot count patches with
      [ count neighbors4 with [ pcolor = yellow] = 4 ]
end
```

✓ modifying "go" to become observer code

```
to go              ; turtle code
   ask turtles [
      pick-up-chip
      find-new-pile
      drop-off-chip ]
   draw-plot
end
```

Complex Systems Made Simple

1. Introduction

2. A Complex Systems Sampler

3. Commonalities

4. NetLogo Tutorial