2-Day Introduction to Agent-Based Modelling

Day 2: Session 6

Mutual adaption

Manchester Metropolitan University

Q&A of experienced modellers

Mutual Adaption and Emergence

- Many interesting cases come about when agents are mutually adapting, so that the resultant organisation or social structure results from this mutual adaption
- However such a process can be difficult to predict from the initial conditions, this is called “emergence”
- Chance developments during the development of such organisation can determine which of several possible outcomes result
- Sometimes there are several, quite different, kinds of outcome that can occur from the same start
- In such situations, averaging the results from many runs is not helpful, indeed can be very misleading – better to try to characterise the different “phases”

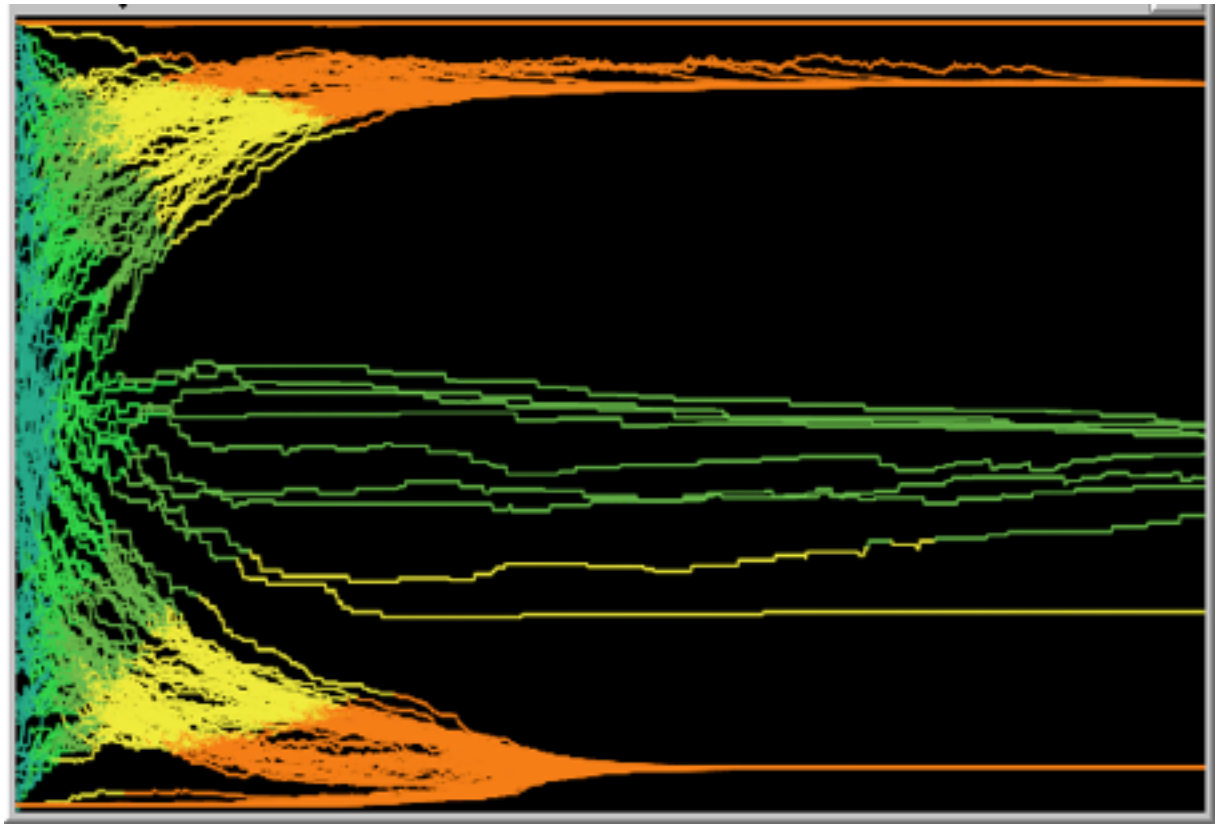
Simulation of Influence with a Group

- Model originated from an EU project looking at how information disseminated to farmers
- They noticed in meetings that opinions often diverged into contrasting groups
- They made an abstract simulation to try and capture this phenomena
- Now a great family of related models along these lines, called “opinion dynamic” models
- This is a simplified version of one of these

Details of this Influence Simulation

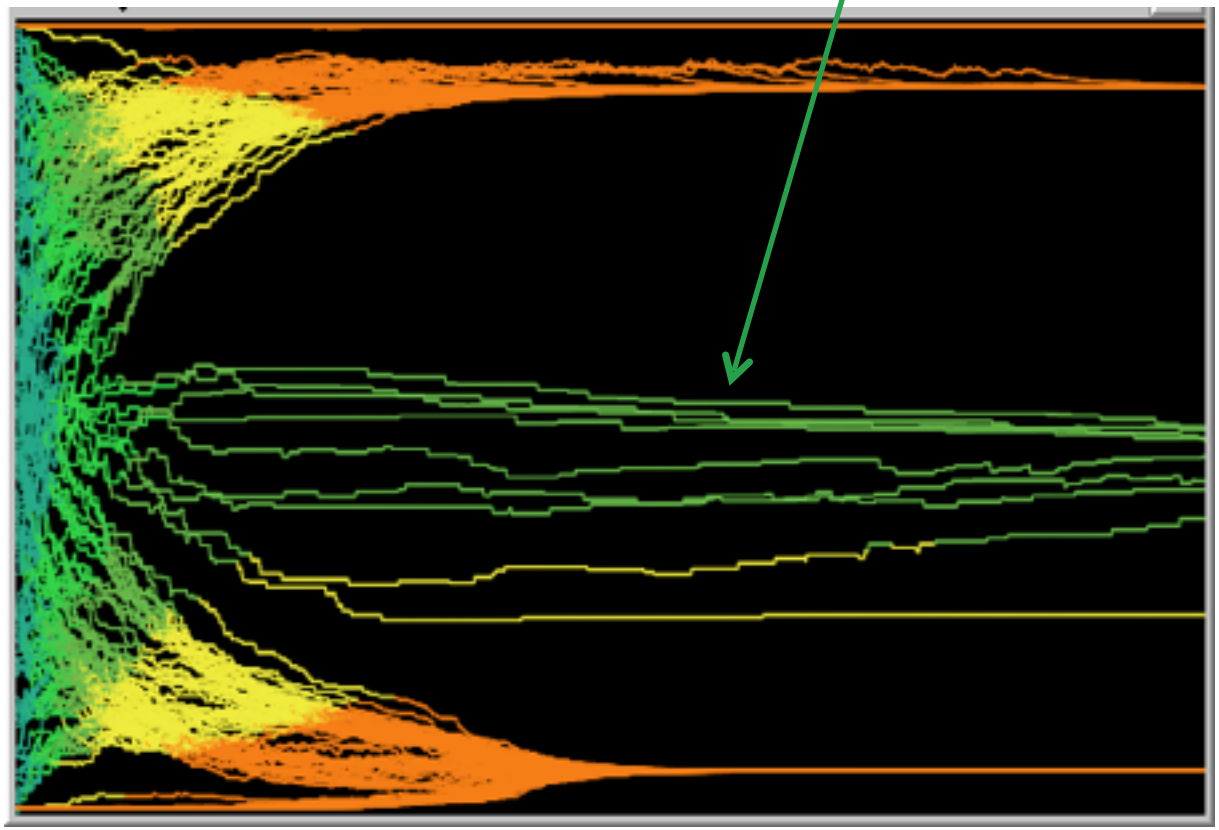
- Agents all have different levels of:
  - agreement on an issue, represented by a number -1 to 1
  - uncertainty about their opinion, represented by a number from 0 to 2
- Each iteration one (randomly picked) agent is randomly paired with another
- That other influences their opinion and uncertainty, but only if the other's opinion is sufficiently close to their own (difference is less than their uncertainty)
- There are some "extremists" who are divided between those with opinion 1 and -1 initially
- And "moderates" who have a random opinion initially
- This is a simple version of an existing model (see Info)
- There are many, many variants of these!

An example of what happens



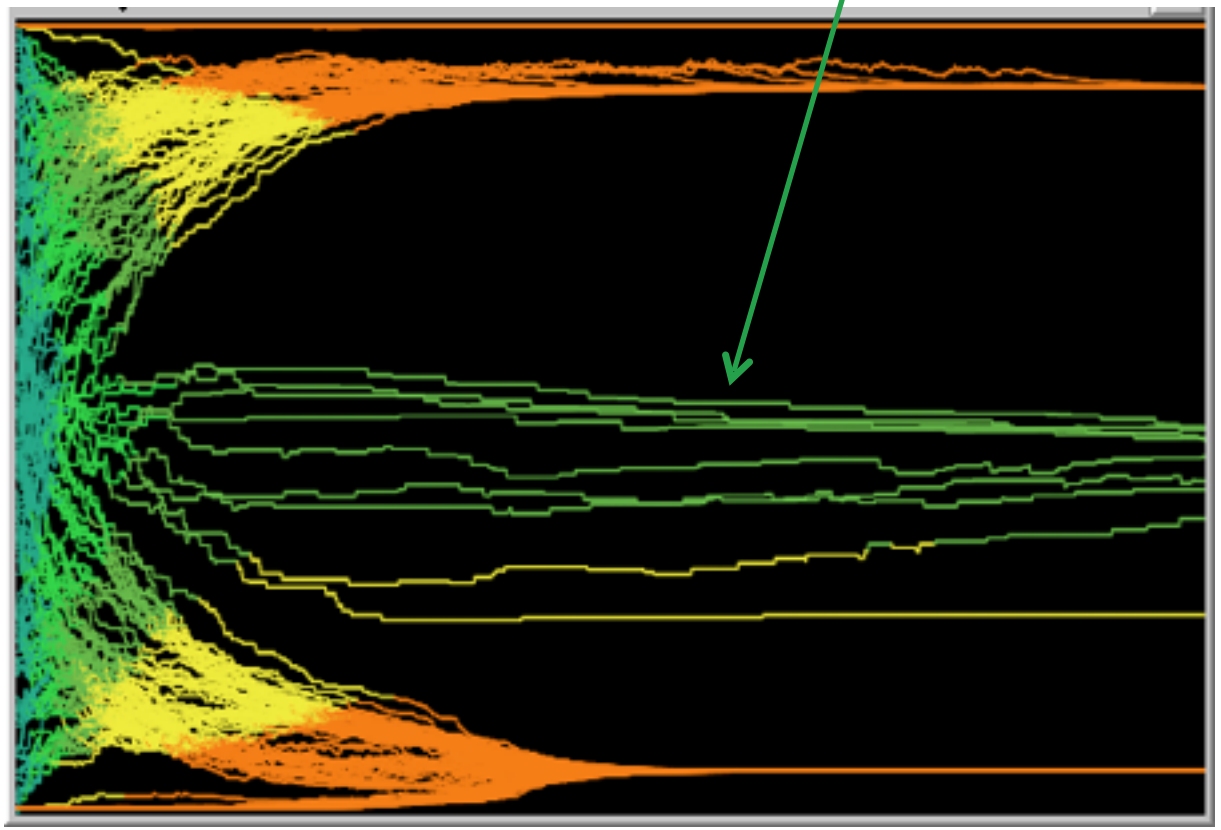
An example of what happens

Each line shows the "trajectory" of a single agent



An example of what happens

Each line shows the "trajectory" of a single agent

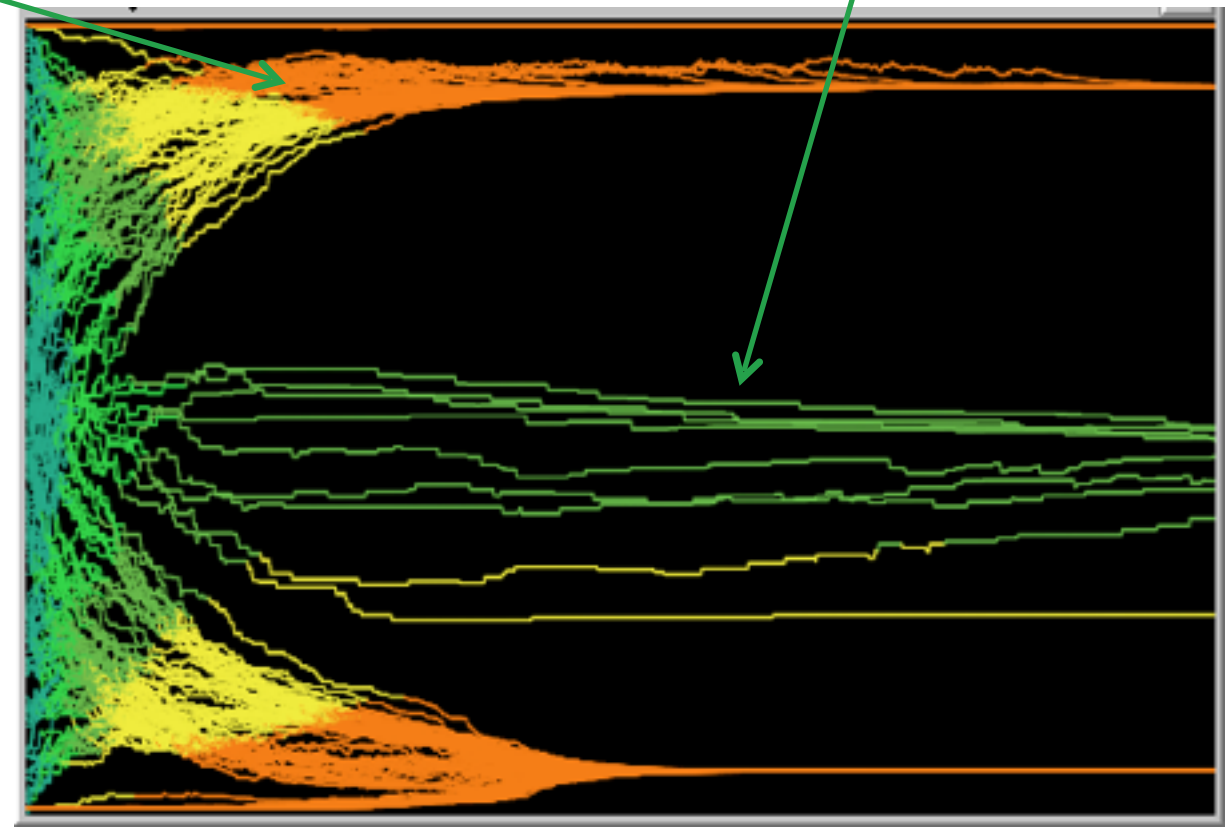


The vertical scale represents the opinion of each, from -1 up to 1

An example of what happens

The COLOUR of each is their level of uncertainty, from blue (maximally uncertain) to red (minimally uncertain)

Each line shows the "trajectory" of a single agent

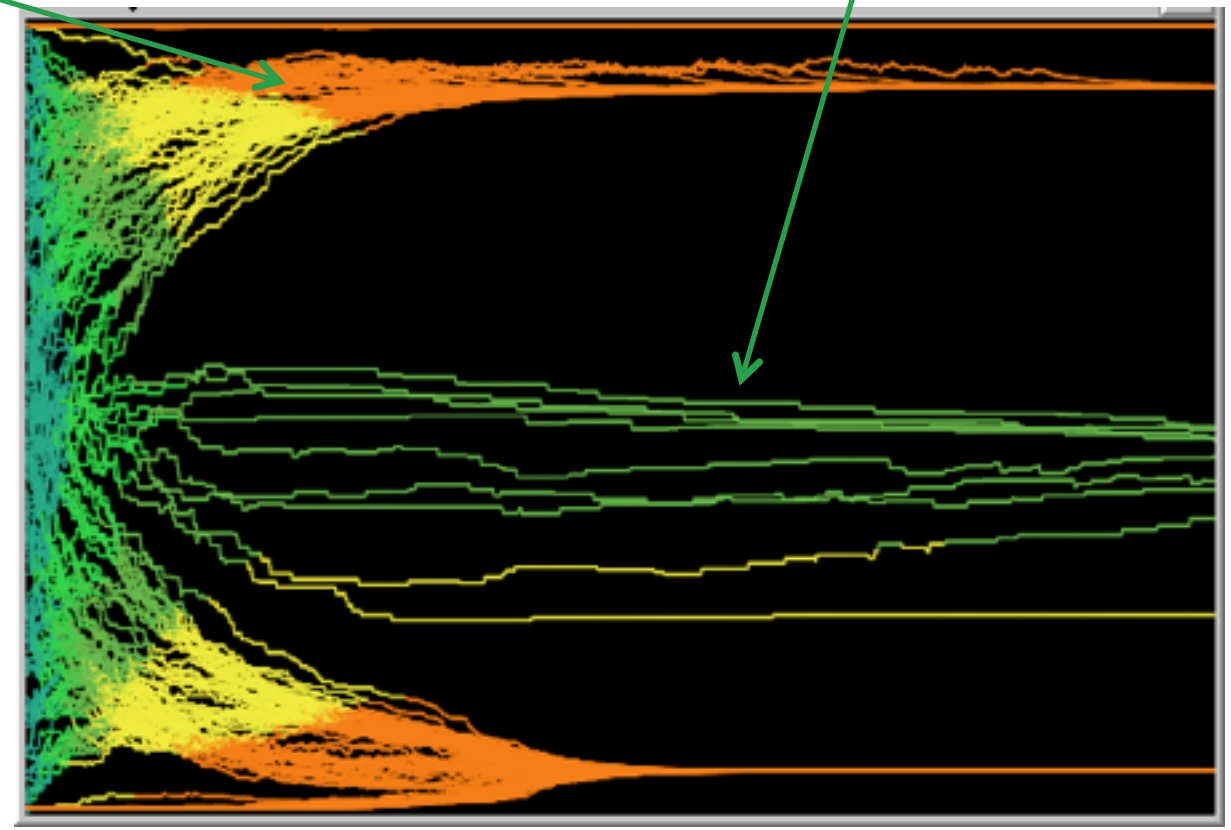


The vertical scale represents the opinion of each, from -1 up to 1

An example of what happens

The COLOUR of each is their level of uncertainty, from blue (maximally uncertain) to red (minimally uncertain)

Each line shows the "trajectory" of a single agent



The vertical scale represents the opinion of each, from -1 up to 1

(Simulation) time is this axis

An example of what happens

The COLOUR of each is their level of uncertainty, from blue (maximally uncertain) to red (minimally uncertain)

Each line shows the "trajectory" of a single agent

The vertical scale represents the opinion of each, from -1 up to 1

In this case (roughly) two groupings with "extreme" certain views emerged

(Simulation) time is this axis

The Consensus Simulation

- Load the "6-consensus.nlogo" simulation
- "prop-of-extremists" is the proportion of extremists in the initial population
- "uncert-of-moderates" is the initial uncertainty of the moderates (initial uncertainty of extremists is fixed at 0.05)
- "speed-uncertainty-change" is how much an agent's uncertainty is changed if influenced by another (opinion is always changed 5%)
- Play with the settings, run the simulations, see how many qualitatively different kinds of outcome there are and under what conditions they tend to occur

Different kinds of procedure

- Since the context of commands matters, *whether the commands are being done within the context of an agent, the observer, a patch (or even a link)* …
- …it is useful to keep track of which procedure (or chunk of code) is within which kind of context
- Some primitives and variables can only be used within an agent (turtle) context, others only within a patch context and others only within the observer context, etc.

Some Global Procedures in the code

```
;;;;;;;;;;;;;;;;;;;;;;;;
;;; global procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

to setup
  ;; note that parameter names have suffixes of the paramter name

  clear-all

  ;; later used in colouring the plots
  set uncertainty-colours [red orange yellow green lime turquoise]
  set uncert-of-extremists 0.05

  ;; calculations as to the size of the various sub-populations
  let num-extremists round num-of-agents * prop-of-extremists
  let num-moderates num-of-agents - num-extremists
  let num-upper-extremists num-extremists / 2
  let num-lower-extremists num-extremists - num-upper-extremists

  ;; create moderates with random opinions
  create-turtles num-moderates [
    set uncertainty uncert-of-moderates
    set opinion (random-float 2) - 1
  ]
```

Some Global Procedures in the code

The "**setup**" and "**go**" procedures are within the observer (the global) context

```
;;;;;;;;;;;;;;;;;;;;;;;;
;;; global procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

to setup
  ;; note that parameter names have suffixes of the paramter name

  clear-all

  ;; later used in colouring the plots
  set uncertainty-colours [red orange yellow green lime turquoise]
  set uncert-of-extremists 0.05

  ;; calculations as to the size of the various sub-populations
  let num-extremists round num-of-agents * prop-of-extremists
  let num-moderates num-of-agents - num-extremists
  let num-upper-extremists num-extremists / 2
  let num-lower-extremists num-extremists - num-upper-extremists

  ;; create moderates with random opinions
  create-turtles num-moderates [
    set uncertainty uncert-of-moderates
    set opinion (random-float 2) - 1
  ]
```

Some Global Procedures in the code

```
;;;;;;;;;;;;;;;;;;;;;;;;
;;; global procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

to setup
  ;; note that parameter names have suffixes of the paramter name

  clear-all

  ;; later used in colouring the plots
  set uncertainty-colours [red orange yellow green lime turquoise]
  set uncert-of-extremists 0.05

  ;; calculations as to the size of the various sub-populations
  let num-extremists round num-of-agents * prop-of-extremists
  let num-moderates num-of-agents - num-extremists
  let num-upper-extremists num-extremists / 2
  let num-lower-extremists num-extremists -

  ;; create moderates with random opinions
  create-turtles num-moderates [
    set uncertainty uncert-of-moderates
    set opinion (random-float 2) - 1
  ]
```

The "**setup**" and "**go**" procedures are within the observer (the global) context

But some chunks of code are within the agent context

Agent procedures

```
;;;;;;;;;;;;;;;;;;;;;;;;
;;; agent procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

to be-influenced-from [oth]
  ;; only influenced if difference between opinion and other's opinion i
  ;; my uncertainty...
  if abs (opinion - [opinion] of oth) < uncertainty [
    ;; ...in which case shift my opinion and uncertainty towards other's
    set opinion 0.95 * opinion + 0.05 * [opinion] of oth
    set uncertainty (1 - speed-uncertainty-change) * uncertainty
                    + speed-uncertainty-change * [uncertainty] of oth
  ]
end

to initialise-agent-display
  ;; what each turtle does in setup
  hide-turtle
  penup
  colour-agent
  position-agent 0
  pendown
end
```

Agent procedures

A lot of the other procedures are within the agent context

```
;;;;;;;;;;;;;;;;;;;;;;;;
;;; agent procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

to be-influenced-from [oth]
  ;; only influenced if difference between opinion and other's opinion i
  ;; my uncertainty...
  if abs (opinion - [opinion] of oth) < uncertainty [
    ;; ...in which case shift my opinion and uncertainty towards other's
    set opinion 0.95 * opinion + 0.05 * [opinion] of oth
    set uncertainty (1 - speed-uncertainty-change) * uncertainty
                    + speed-uncertainty-change * [uncertainty] of oth
  ]
end

to initialise-agent-display
  ;; what each turtle does in setup
  hide-turtle
  penup
  colour-agent
  position-agent 0
  pendown
end
```

Agent procedures

```
;;;;;;;;;;;;;;;;;;;;;;;;
;;; agent procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

to be-influenced-from [oth]
  ;; only influenced if difference between opinion and other's opinion i
  ;; my uncertainty...
  if abs (opinion - [opinion] of oth) < uncertainty [
    ;; ...in which case shift my opinion and uncertainty towards other's
    set opinion 0.95 * opinion + 0.05 * [opinion] of oth
    set uncertainty (1 - speed-uncertainty-change) * uncertainty
                    + speed-uncertainty-change * [uncertainty] of oth
  ]
end

to initialise-agent-displ
  ;; what each turtle does in setup
  hide-turtle
  penup
  colour-agent
  position-agent 0
  pendown
end
```

A lot of the other procedures are within the agent context

Local agent variables are only usable within the agent context

Agent procedures

```
;;;;;;;;;;;;;;;;;;;;;;;;
;;; agent procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

to be-influenced-from [oth]
  ;; only influenced if difference between opinion and other's opinion i
  ;; my uncertainty...
  if abs (opinion - [opinion] of oth) < uncertainty [
    ;; ...in which case shift my opinion and uncertainty towards other's
    set opinion 0.95 * opinion + 0.05 * [opinion] of oth
    set uncertainty (1 - speed-uncertainty-change) * uncertainty
                    + speed-uncertainty-change * [uncertainty] of oth
  ]
end

to initialise-agent-displ
  ;; what each turtle does in setup
  hide-turtle
  penup
  colour-agent
  position-agent 0
  pendown
end
```

A lot of the other procedures are within the agent context

Local agent variables are only usable within the agent context

Some primitives are only usable within the agent context

Things to try in this simulation

- What might one add to help understand what is happening in the simulation?
- What happens if you change code in the procedure: "**be-influenced-from**"?
- What happens if everyone is only influenced by the nearest other?
- How might you change the simulation so that everybody is influenced exactly once per simulation tick rather than one agent every tick?

Related models

- Other models that are similar (but more complex) include:
  - "Bounded Confidence Opinion Dynamics" – the full opinion dynamics model with lots of options and references to original papers
  - "Dissemination of Culture" – a reimplementation of Axelrod's model on this, done by programming patches rather than agents
- Found in the "Extra Examples" directory

The End

2-Day Introduction to Agent-Based Modelling

http://cfpm.org/simulationcourse

Manchester Metropolitan University