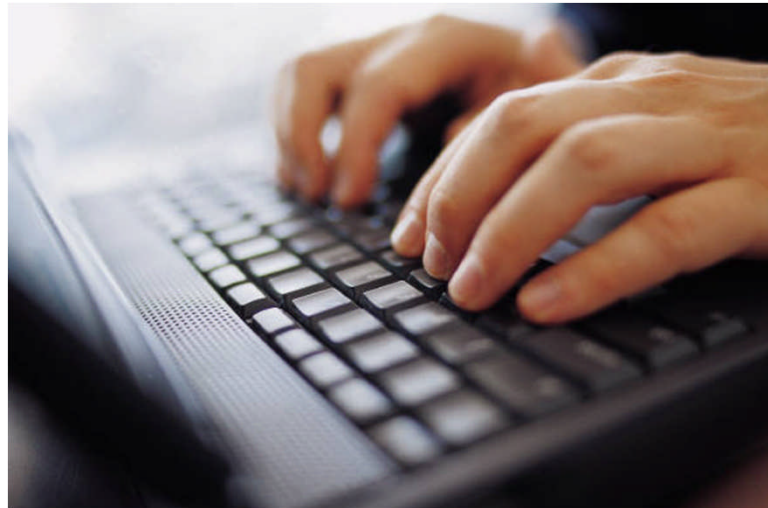




## Hands on Model Use Ahead



Load model: Schelling Segregation.alp

# A Model to Examine the Emergence of Segregation

The screenshot shows the AnyLogic Advanced software interface for a Schelling Segregation model. The main workspace displays a yellow square representing a population, with a red border and a slider for "Preference to have neighbors of the same color" set at 30%. The interface includes a project tree on the left, a palette on the right, and a properties panel at the bottom.

**Project Tree (Left):**

- Plain Variables
- Statecharts
- Presentation
- Simulation: Main
- Schelling Segregation
  - Main
  - Parameters
  - Environments
    - environment
  - Embedded Objects
    - Presentation
  - Person
    - Parameters
    - Plain Variables
      - satisfied
    - Presentation
  - Simulation: Main

**Properties Panel (Bottom):**

**Main - Active Object Class**

**General** Name:   Ignore

**Advanced**

**Agent**  Agent  Generic

**Parameters**

Startup Code:

Destroy Code:

**Palette (Right):**

- Model
  - Parameter
  - Flow Aux Variable
  - Stock Variable
  - Event
  - Dynamic Event
  - Plain Variable
  - Collection Variable
  - Function
  - Table Function
  - Port
  - Connector
  - Entry Point
  - State
  - Transition
  - Initial State Pointer
  - Branch
  - History State
  - Final State
  - Environment
- Action
- Analysis
- Presentation
- Connectivity
- Enterprise Library
- More Libraries...

# A Discrete Spatial Environment with Random Agent Positioning

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a grid-based environment with a red box labeled "Preference to have neighbors of the same color:" and a yellow box below it. A red box labeled "environment" is positioned on the grid, with a "Threshold" label above it and "people [...]" below it. The Properties panel on the right shows the "environment - Environment" configuration. The "Advanced" tab is selected, showing the following settings:

- Space type:  Continuous  Discrete  GIS
- Width: 400
- Height: 400
- Columns: 100
- Rows: 100
- Neighborhood type: Moore
- Layout type: Random  Apply on startup
- Network type: User-defined  Apply on startup
- Connections per agent: 2
- Connection range: 50
- Neighbor link fraction: 0.95

Annotations in the image include a red arrow pointing to the "Width" and "Height" fields, and a blue box labeled "Width & Height in Discrete Cells" with arrows pointing to the "Columns" and "Rows" fields. The text "Spatial Width & Height" is written in red above the Properties panel.

# Population Dependence on the Population

The screenshot displays the AnyLogic Advanced software interface, titled "AnyLogic Advanced [EDUCATIONAL USE ONLY]". The main workspace shows a simulation model for "Schelling Segregation". The model is composed of several elements: a "Threshold" element, an "environment" element, and a "people [...]" element. A prominent red box in the workspace contains the text "Preference to have neighbors of the same color:" and a slider control.

The left sidebar shows a project tree with the following structure:

- Plain Variables
- Statecharts
- Presentation
- Simulation: Main
- Schelling Segregation
  - Main
    - Parameters
    - Environments
      - environment
    - Embedded Objects
      - people
    - Presentation
  - Person
    - Parameters
    - Plain Variables
    - satisfied
    - Presentation
  - Simulation: Main

The bottom panel shows the "Properties" window for the "people - Person" element. The properties are as follows:

- General:** Name: people,  Show Name,  Ignore,  Public,  Show At Runtime, [Create Presentation](#)
- Parameters:** Type: Person, Package: schelling\_segregation, Environment: environment
- Color:** randomTrue( 0.5 ) ? Color.red : Color.black
- Replication:** 8000

The right sidebar shows a palette of simulation elements:

- Model
  - Parameter
  - Flow Aux Variable
  - Stock Variable
  - Event
  - Dynamic Event
  - Plain Variable
  - Collection Variable
  - Function
  - Table Function
  - Port
  - Connector
  - Entry Point
  - State
  - Transition
  - Initial State Pointer
  - Branch
  - History State
  - Final State
  - Environment
- Action
- Analysis
- Presentation
- Connectivity
- Enterprise Library
- More Libraries...

# Slider Input Sets Parameter Value

AnyLogic Advanced [EDUCATIONAL USE ONLY]

File Edit View Model Window Help

80%

Project Search

- Simulation: Main
  - Schelling Segregation
    - Main
      - Parameters
      - Environments
        - environment
      - Embedded Objects
        - people
      - Presentation
        - rect
        - rect1
        - people\_presentation
        - text: Prefer...
        - text1: 30%
        - button
        - button1
        - slider

Person

environment

people [..]

“Threshold” parameter

Preference to have neighbors of the same color: 30%

slider - Slider

General

Name: slider  Show Name  Ignore  Public  Icon

Advanced

Orientation:  Horizontal  Vertical

Dynamic

Description

Minimum Value: 0

Maximum Value: 1

Default Value: Threshold

Enabled:

Action:

Threshold = value;

Default value is that of Threshold parameter

Sets Threshold Parameter Value

Palette

- Model
  - Parameter
  - Flow Aux Variable
  - Stock Variable
  - Event
  - Dynamic Event
  - Plain Variable
  - Collection Variable
  - Function
  - Table Function
  - Port
  - Connector
  - Entry Point
  - State
  - Transition
  - Initial State Pointer
  - Branch
  - History State
  - Final State
  - Environment
- Action
- Analysis
- Presentation
- Connectivity
- Enterprise Library
- More Libraries...

# Person is Assigned a Randomly Picked Color

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a simulation model with a red square representing a person and a blue circle representing a color parameter. A red arrow points from the text "Person's Visual Representation" to the red square. A blue arrow points from the text "Color is set to either red or black with equal likelihood" to the blue circle. The Properties window for the "color - Parameter" is open, showing the following configuration:

- Name: color
- Show Name:
- Ignore:
- Public:
- Show At Runtime:
- Type:  void (just action)  boolean  int  double  String  Other: Color
- Default Value: `randomTrue( 0.5 ) ? Color.red : Color.black`
- Dynamic:  Save in snapshot:
- On Change: (empty field)

The left sidebar shows a project tree with the following structure:

- people\_presentation
  - AA text: Prefer...
  - AA text1: 30%
  - button
  - button1
  - slider
- Person
  - Parameters
    - color: random...
  - Plain Variables
    - satisfied
  - Presentation
    - Simulation: Main
      - Presentation
        - frame
        - rect1
        - rect
        - AA text: Schell...

# Core Segregation (Movement) Logic

**Person - Active Object Class**

Space type:  Continuous  Discrete  GIS

**Agent**

Environment defines initial location ← **Person's Initial Location**

Initial coordinates:

X:

Y:

Movement parameters:

Velocity:

Rotation:

On Arrival:

On Message Received:

On Before Step:

```
//calc hbow many neighbors have same color as me
int nname = 0;
Agent[] neighbors = getNeighbors();
if( neighbors == null ) {
    satisfied = true; //no neighbors is good too
    return;
}
for( Agent a : neighbors )
    if( ((Person)a).color.equals( color ) )
        nname++;
//satisfied if percent of same color is greater than a given threshold
satisfied = nname >= get_Main().Threshold * neighbors.length;
```

On Step:

```
if( ! satisfied && randomTrue( 0.3 ) )
    jumpToRandomEmptyCell();
```

**Count neighbors Sharing same colour (should be in diff. Function).**

**Only satisfied if fraction of surrounding individuals Sharing color exceeds threshold**

**if dissatisfied, 30% chance of moving**

# Experiment: Simulation Sets Parameter Assumptions

The screenshot displays the AnyLogic Advanced software interface. The main workspace is titled "Schelling Segregation Model" and contains the following text:

**Schelling Segregation Model**

The Schelling Segregation Model was first developed by Thomas C. Schelling (Micromotives and Macrobehavior, W. W. Norton and Co., 1978, pp. 147-155). It represents one of the first constructive models of a dynamical system capable of self-organization.

Schelling placed pennies and dimes on a chess board and moved them around according to various rules. He interpreted the board as a city, with each square of the board representing a house or a lot. He interpreted the pennies and dimes as agents representing any two groups in society, such as two different races of people, boys and girls, smokers and non-smokers, etc. The neighborhood of an agent occupying any location on the board consisted of the squares adjacent to this location. Thus, interior agents had eight neighbors while boundary agents had either three or five neighbors. Rules could be specified that determined whether a particular agent was happy in its current location. If it was unhappy, it would try to move to another location on the board, or possibly just exit the board entirely.

As can be expected, Schelling found that the board quickly evolved into a strongly segregated location pattern if the agents' "happiness rules" were specified so that segregation was heavily favored. Surprisingly, however, he also found that initially integrated boards tipped into full segregation even if the agents' happiness rules expressed only a mild preference for having neighbors of their own type.

Run the model and switch to Main view

The interface includes a left-hand palette with a tree view showing the model structure: slider, Person, Parameters (color: random...), Plain Variables (satisfied), Presentation (frame, rect1, rect, Aa text: Schell..., Aa text1: The Sc..., image, Aa text2: AnyLog..., button), Problems (table with Description and Location columns), Properties, and Console. The Console shows "Simulation - Simulation Experiment" with a Threshold parameter set to 0.7. A right-hand palette lists various model components like Parameter, Flow Aux Variable, Stock Variable, Event, Dynamic Event, Plain Variable, Collection Variable, Function, Table Function, Port, Connector, Entry Point, State, Transition, Initial State Pointer, Branch, History State, Final State, and Environment. At the bottom right, there are buttons for Action, Analysis, Presentation, Connectivity, Enterprise Library, and More Libraries...



# Add a Parameter to Main

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a grid with several objects: a slider for 'Threshold' (set to 0.7), an 'environment' object, a 'people' object, and a 'likelihoodOfMovementIfDissatisfied' parameter. A red box highlights a slider for 'Preference to have neighbors of the same color:' set to 30%. The 'Properties' panel at the bottom shows the configuration for the 'likelihoodOfMovementIfDissatisfied' parameter, including its name, type (double), and default value (0.3).

**AnyLogic Advanced [EDUCATIONAL USE ONLY]**  
File Edit View Model Window Help

Project Search

Model

- Parameter
- Flow Aux Variable
- Stock Variable
- Event
- Dynamic Event
- Plain Variable
- Collection Variable
- Function
- Table Function
- Port
- Connector
- Entry Point
- State
- Transition
- Initial State Pointer
- Branch
- History State
- Final State
- Environment

Properties Console

**likelihoodOfMovementIfDissatisfied - Parameter**

**General**

Name:   Show Name  Ignore  Public  Show At Runtime

**Editor**

Type:  void (just action)  boolean  int  double  String  Other:

Default Value:

Dynamic  Save in snapshot

On Change:

Selection

# Experiment: Add a Slider!

The screenshot displays the AnyLogic Advanced software interface. The main workspace is a dark red area titled "Schelling Segregation Model". It contains two paragraphs of text and a button labeled "Run the model and switch to Main view".

**Schelling Segregation Model**

The Schelling Segregation Model was first developed by Thomas C. Schelling (Micromotives and Macrobehavior, W. W. Norton and Co., 1978, pp. 147-155). It represents one of the first constructive models of a dynamical system capable of self-organization.

Schelling placed pennies and dimes on a chess board and moved them around according to various rules. He interpreted the board as a city, with each square of the board representing a house or a lot. He interpreted the pennies and dimes as agents representing any two groups in society, such as two different races of people, boys and girls, smokers and non-smokers, etc. The neighborhood of an agent occupying any location on the board consisted of the squares adjacent to this location. Thus, interior agents had eight neighbors while boundary agents had either three or five neighbors. Rules could be specified that determined whether a particular agent was happy in its current location. If it was unhappy, it would try to move to another location on the board, or possibly just exit the board entirely.

As can be expected, Schelling found that the board quickly evolved into a strongly segregated location pattern if the agents' \*happiness rules\* were specified so that segregation was heavily favored. Surprisingly, however, he also found that initially integrated boards tipped into full segregation even if the agents' happiness rules expressed only a mild preference for having neighbors of their own type.

Run the model and switch to Main view

The interface includes a menu bar (File, Edit, View, Model, Window, Help), a toolbar with various icons, and a palette on the right side. The palette is divided into sections: Model, Action, Analysis, and Presentation. The "Presentation" section is highlighted in red, and the "Slider" icon is selected. Other icons in the palette include Line, Polyline, Curve, Rectangle, Round Rectangle, Oval, Arc, Pixel, Text, Image, Group, Button, Check Box, Edit Box, Radio Buttons, Combo Box, List Box, File Chooser, Progress Bar, CAD Drawing, and GIS Map. The bottom of the interface shows a Properties panel and a Console panel.

# Setting the Slider Properties

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a red text box with the following text:

(Micromotives and Macrobehavior, W. W. Norton and Co., 1978, pp. 147-155). It represents one of the first constructive models of a dynamical system capable of self-organization.

Schelling placed pennies and dimes on a chess board and moved them around according to various rules. He interpreted the board as a city, with each square of the board representing a house or a lot. He interpreted the pennies and dimes as agents representing any two groups in society, such as two different races of people, boys and girls, smokers and non-smokers, etc. The neighborhood of an agent occupying any location on the board consisted of the squares adjacent to this location. Thus, interior agents had eight neighbors while boundary agents had either three or five neighbors. Rules could be specified that determined whether a particular agent was happy in its current location. If it was unhappy, it would try to move to another location on the board, or possibly just exit the board entirely.

As can be expected, Schelling found that the board quickly evolved into a strongly segregated location pattern if the agents' \*happiness rules\* were specified so that segregation was heavily favored. Surprisingly, however, he also found that initially integrated boards tipped into full segregation even if the agents' happiness rules expressed only a mild preference for having neighbors of their own type.

Below the text box is a button labeled "Run the model and switch to Main view" and a slider control. The slider is currently set to 0.3.

The Properties window at the bottom shows the configuration for the "sliderMovementChance - Slider" property:

- Name: sliderMovementChance
- Show Name:
- Ignore:
- Public:
- Icon:
- Orientation:  Horizontal  Vertical
- Minimum Value: 0
- Maximum Value: 1
- Default Value: 0.3
- Enabled:  true
- Action: (empty)

The left sidebar shows the Project tree with the following structure:

- Person
  - Parameters
    - color: random...
  - Plain Variables
    - satisfied
  - Presentation
    - Simulation: Main
      - Presentation
        - frame
        - rect1
        - rect
        - Aa text: Schell...
        - Aa text1: The Sc...
        - image
        - Aa text2: AnyLog...
        - button
        - sliderMovementChance

The right sidebar shows the Palette with the following items:

- Model
  - Parameter
  - Flow Aux Variable
  - Stock Variable
  - Event
  - Dynamic Event
  - Plain Variable
  - Collection Variable
  - Function
  - Table Function
  - Port
  - Connector
  - Entry Point
  - State
  - Transition
  - Initial State Pointer
  - Branch
  - History State
  - Final State
  - Environment
- Action
- Analysis
- Presentation
- Connectivity
- Enterprise Library
- More Libraries...

# Setting Value for Parameter from Slider

The screenshot displays the AnyLogic Advanced [EDUCATIONAL USE ONLY] interface. The main workspace shows a simulation model with a red rectangular area containing a slider. The slider is labeled "Preference to have neighbors of the same color:" and is currently set to 30%. Below the slider is a yellow rectangular area. The left sidebar shows a project tree with a "Simulation: Main" folder expanded, containing a "Presentation" folder with a "sliderMovementChance" parameter. The bottom panel shows the "Simulation - Simulation Experiment" properties, with the "Threshold" parameter set to 0.7 and the "likelihoodOfMovementIfDissatisfied\*" parameter set to "sliderMovementChance.value".

**Simulation - Simulation Experiment**

**General**

Name:  Main active object class (root):   Ignore

**Advanced**

Random number generation:

Random seed (unique simulation runs)

Fixed seed (reproducible simulation runs) Seed Value:

**Model Time**

**Presentation**

**Window**

**Parameters**

**Description**

Threshold

likelihoodOfMovementIfDissatisfied\*

# Modify Person's Behavior to Depend on New Parameter

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a model diagram with two objects: 'color' and 'satisfied'. The 'Person' class is selected in the Project pane. The Properties pane shows the 'Agent' tab, and the Console pane displays the following code:

```
Person - Active Object Class  
General  
Advanced  
Agent  
Parameters  
Description  
  
if( neighbors == null ) {  
    satisfied = true; //no neighbors is good too  
    return;  
}  
for( Agent a : neighbors )  
    if( ((Person)a).color.equals( color ) )  
        nsame++;  
//satisfied if percent of same color is greater than a given threshold  
satisfied = nsame >= get_Main().Threshold * neighbors.length;  
  
On Step:  
if( ! satisfied && randomTrue( get_Main().likelihoodOfMovementIfDissatisfied ) )  
    jumpToRandomEmptyCell();
```

Updated Code ("get\_Main()") required  
Because new parameter is global  
And lives in Main class rather than in  
Person class.)

# Movement in Discrete Space

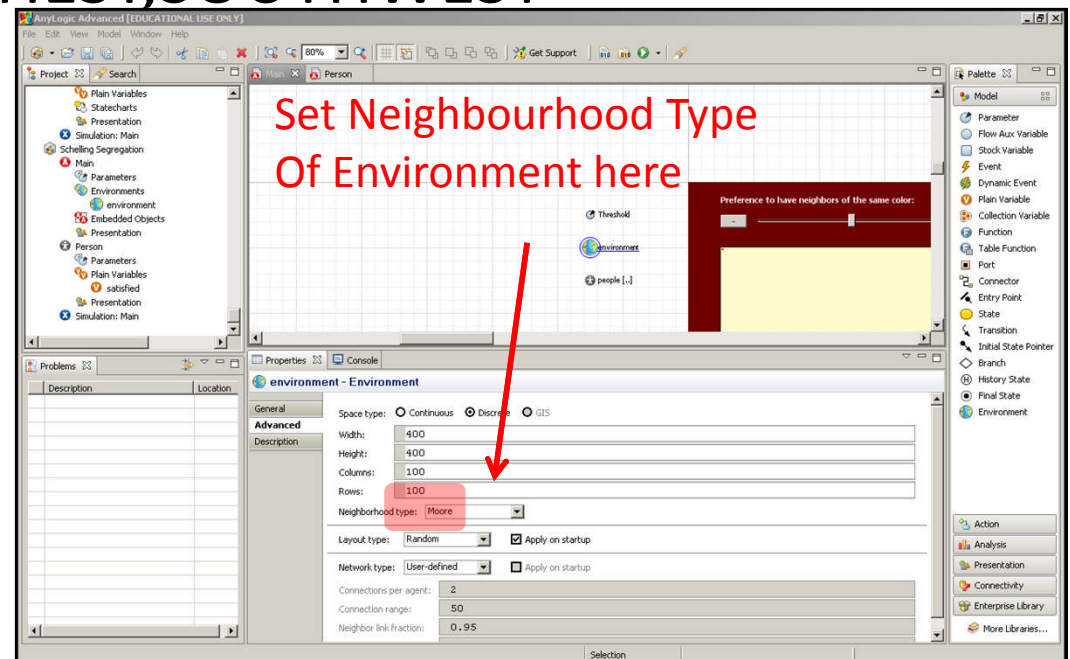
- `jumpToCell(int row, int column)`
  - Jumps to a particular unoccupied cell
  - Precondition: destination cell is unoccupied
- `moveToNextCell(int direction)`
  - Moves agent into neighbouring cell in a given direction
  - Directions: NORTH, SOUTH, EAST, WEST, NORTHEAST, NORTHWEST, SOUTHEAST, SOUTHWEST
    - Precondition: destination cell is unoccupied
- `jumpToRandomEmptyCell`
  - Jumps to randomly selected empty cell (returning true), returns false if no empty cell can be located

# Discovery in Discrete Space

- `int []findRandomEmptyCell`
  - Returns row & column of an unoccupied cell
- Getting agents in cell or direction
  - `getAgentAtCell(int row, int column)`
  - `getAgentNextToMe(int direction)`
  - `getNeighbors()`

# Neighbourhood Models

- Moore: Cardinal directions
  - NORTH,SOUTH,EAST, WEST
- Euclidean
  - NORTH, SOUTH, EAST, WEST, NORTHEAST, NORTHWEST, SOUTHEAST,SOUTHWEST





# Important Distinction

- Suppose an agent is moving in discrete 2D space and need to be concerned about moving into the same cell as another agent
- We can readily prevent this agent from moving into another cell currently occupied
- But can we prevent this agent from colliding with another agent that wishes to move into the same cell?
  - To answer this, we need to be clear about the model of time used by agents

# Two Key Models of Time in Anylogic: Synchronous Time

- Here, agents all change in lockstep, separated by fixed “time steps”
- When computing agent behavior (to determine agent state in the next timestep), our enquiries about agent state (e.g. using *getAgentAtCell* or *getAgentNextToMe*) need to ask about the situation in the current timestep
  - We gather needed information regarding current state in “onBeforeStep”, and changes are performed in “OnStep”.
- This is similar to what we saw in System Dynamics – the changes over the next small interval of time ( $\Delta t$ ) depend on the current value of the stocks
  - These changes are then applied at once, and all stocks are updated

# Enabling Synchronous Time

- Unless enable the steps, the various handlers for synchronized time (e.g. “On before step”, “On step”, “On after step”) etc.) are executed
  - Both environment and agents have “On before step” and “On after step” handlers
  - “On before step” for environments is executed before the corresponding method for agents
  - “On after step” for environments is executed after the corresponding method for agents
- Synchronous time can be enabled via the environment “General” page
  - Click checkbox “Enable steps”

# Two Key Models of Time in Anylogic: Asynchronous Time

- Here, every agent is updated at a different time
- No two agents are typically likely to be updated at exactly the same time, so when considering the state of other agents they “see” the last situation where the other agent has been updated

# Synchronization & Discrete Agent Movement

- In Synchronous mode, it is difficult to know if two agents will collide using data on the current timestep
  - Even if we know where the other object was during the current timestep, it's possible it will move into the cell we wish to occupy in the next timestep
- It is simpler to handle this asynchronously
  - Here, we can have each agent update at slightly different times, and observe the location of the other agents at the current time – without any significant chance that they will move to the same place at the same time.
- This issue only arises for discrete agent movement, as this is the only case where cells only contain 1 agent