

Agent Mobility in 2D Landscapes (Bonus: Some UI Customization)

Nathaniel Osgood

MIT 15.879

March 23, 2012

Agent Mobility

- Thus far, we have looked at spatial dynamics where each agent remains stationary
 - Continuous space (static & dynamic populations)
 - Discrete space (cellular automata)

2D Spatial Embedding: Mobility Implications

- Continuous embedding (e.g. Wandering elephants)
 - No physical exclusion: Agents are assumed to be small compared to landscape scale, and exhibit arbitrary spatial density without interfering
 - Agents move
 - In a direction
 - With some speed
- Discrete cells (e.g. Agent-based predator prey, Schelling Segregation)
 - Divided into “Columns” and “Rows”
 - Physical exclusion: Only one agent in a cell at a time
 - Agents move continuously or discontinuously from cell to cell



Hands on Model Use Ahead



Load model: Wandering Elephants.alp

Environment

The screenshot displays the AnyLogic Advanced software interface, specifically the "environment - Environment" configuration window. The interface is divided into several panels:

- Project Panel (Left):** Shows a hierarchical tree of the model structure, including "Main", "Parameters", "Functions", "Environments", and "Embedded Objects". The "environment" object is selected under "Environments".
- Diagram Canvas (Center):** Displays a grid-based workspace with various objects and connections. A pink box highlights the "environment" object. Other visible objects include "SmokingInitiationByAgeAndSmokingStatusForSexualActivityGroup1", "makeUpVegetation", "placeElephants", "altitude", "vegetation", "mapDrawing", "altitudesDrawn", "DisplacementTable", "AngleTable", "DistrDisplacement", "DistrAngle", "updateVegetation", "vegetationToColor", "altcolor", and "viewVegetation".
- Properties Panel (Bottom):** Shows the configuration for the selected "environment" object. The "General" tab is active, displaying the following settings:
 - Space type:** Continuous (selected), Discrete, GIS
 - Width:** 500
 - Height:** 500
 - Columns:** 100
 - Rows:** 100
 - Neighborhood type:** Moore
 - Layout type:** User-defined (selected), Apply on startup (unchecked)
 - Network type:** User-defined (selected), Apply on startup (unchecked)
 - Connections per agent:** 2
 - Connection range:** 50
- Palette (Right):** A vertical toolbar containing various modeling elements such as "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".
- Problems Panel (Bottom Left):** A table with columns "Description" and "Location", currently empty.

Landscape Information

The screenshot displays the AnyLogic Advanced software interface, specifically the 'Wandering Elephants' model. The main workspace shows a grid of components including 'makeUpVegetation', 'environment', 'vegetationToColor', 'placeElephants', 'DisplacementTable', 'altitude', 'AngleTable', 'vegetation', 'DistrDisplacement', 'mapDrawing', 'DistrAngle', 'altitudesDrawn', and 'updateVegetation'. A color palette is visible on the right side of the workspace.

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

- Wandering Elephants*
 - Elephant
 - Parameters
 - drinkingPeriod: 100
 - smokingInitiationRateByAgeAn
 - Plain Variables
 - Statecharts
 - behavior
 - behavior
 - FreeWandering
 - GotThirsty
 - GoToWater
 - DrinkWater
 - initialState
 - state
 - NewDir
 - Functions
 - Presentation
 - Main
 - Parameters
 - NumberOfElephants: 50
 - Plain Variables

The bottom panel shows the 'vegetation - Plain Variable' properties:

- General**
 - Name: vegetation
 - ☒ Show Name ☐ Ignore ☐ Public ☐ Show At Runtime
 - Access: public ☐ Static ☐ Constant ☒ Save in snapshot
 - Type: ☐ boolean ☐ int ☐ double ☐ String ☒ Other: double []
 - Initial Value: new double [100] [100]

The bottom status bar indicates 'environment - Environment' and 'Selection'.

Agent Movement: Periodic Movement Changes

The screenshot displays the AnyLogic Advanced software interface, specifically the statechart editor for an agent named "Elephant". The main workspace shows a statechart with a "FreeWandering" state and a "NewDir" transition. The "FreeWandering" state contains a "NewDir" transition. The "NewDir" transition is triggered by a "Timeout" event and has an action of "headingRandom()". The "FreeWandering" state also has a "GotThirsty" transition leading to a "GoToWater" state, and a "DrinkWater" transition leading back to "FreeWandering".

The left sidebar shows the project structure, including "Wandering Elephants*", "Elephant", "Parameters", "Plain Variables", "Statecharts", "behavior", "FreeWandering", "GotThirsty", "GoToWater", "DrinkWater", "InitialState", "state", "NewDir", "Functions", "Presentation", "Main", "Parameters", "NumberOfElephants: 50", and "Plain Variables".

The bottom panel shows the properties for the "NewDir - Transition".

NewDir - Transition

General

Name: NewDir ☒ Show Name ☐ Ignore ☐ Public ☒ Show At Runtime

Triggered by: Timeout

Timeout: 12

Action: headingRandom() ;

Guard:

The right sidebar shows the palette with various 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, and More Libraries...

New Direction Change Function Info

The screenshot displays the AnyLogic Advanced [EDUCATIONAL USE ONLY] interface. The main workspace shows a statechart for an elephant's behavior. The statechart includes a 'FreeWandering' state with a self-loop labeled 'NewDir'. Transitions from 'FreeWandering' include 'GotThirsty' leading to a 'GoToWater' state and 'DrinkWater' leading back to 'FreeWandering'. Other elements include a 'drinkingPeriod' parameter, a 'thirsty' plain variable, and a 'headingRandom' function. The 'headingRandom' function is highlighted in the workspace.

The left sidebar shows the project structure, including 'Wandering Elephants*', 'Elephant', 'Parameters', 'Plain Variables', 'Statecharts', 'behavior', 'FreeWandering', 'GotThirsty', 'GoToWater', 'DrinkWater', 'initialState', 'state', 'NewDir', 'Functions', 'Presentation', 'Main', 'Parameters', and 'NumberOfElephants: 50'.

The bottom panel shows the 'headingRandom - Function' properties. The 'General' tab is active, displaying the following information:

- Name: headingRandom
- Access: default
- Return Type: void
- Function arguments: (empty table)

Name	Type

The right sidebar shows the 'Palette' with various modeling 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, and Environment. The bottom right corner shows a 'Selection' tool.

New Direction Change: Function “Body”

The screenshot displays the AnyLogic Advanced interface. On the left, a project tree shows the hierarchy: Wandering Elephants* > Elephant > Parameters > headingRandom. The main workspace shows a statechart for an elephant's behavior. It includes states like 'FreeWandering' (a yellow circle) and 'GoToWater' (a yellow rectangle). Transitions are labeled 'GotThirsty' and 'DrinkWater'. A red arrow points from the text 'Setting Agent Speed (set so as to reach target in fixed time until next target shift)' to the 'setVelocity' line in the function body. A blue arrow points from the text 'Initiates movement towards (randomly chosen) destination' to the 'moveTo' line in the function body.

Setting Agent Speed (set so as to reach target in fixed time until next target shift)

Initiates movement towards (randomly chosen) destination

```
Function body:
stop();
//new velocity (note that 12 is the length of time until stop moving in this direction; we'
setVelocity( get_Main().DistrDisplacement.get() / 12 );
//new heading
double heading = getHeading();
heading += get_Main().DistrAngle.get() * ( randomTrue( 0.5 ) ? 1 : -1 );
//move
moveTo( getX() + 1000*cos( heading ), getY() + 1000*sin( heading ) );
```

(Main) Defining a Custom Angle Distribution

The screenshot displays the AnyLogic University software interface, specifically the 'Main' workspace. The interface is divided into several panels:

- Projects Panel (Left):** Shows a tree view of the project structure. Under the 'Main' folder, the 'Variables' section is expanded, and 'DistrAngle' is selected.
- Workspace (Center):** A grid-based workspace containing various elements. The 'DistrAngle' variable is highlighted with a blue circle. Other elements include 'altitude', 'vegetation', 'mapDrawing', 'altitudesDrawn', 'altitudeImage', 'AngleTable', 'DistrDisplacement', 'viewVegetation', and 'updateVegetation'.
- Properties Panel (Bottom):** Displays the properties for the selected 'DistrAngle' variable. The 'General' tab is active, showing the following settings:
 - Name:** DistrAngle
 - Access:** public
 - Type:** CustomDistribution
 - Initial value:** new CustomDistribution(AngleTable, getDefaultRandomGenerator());
- Palette (Right):** A vertical panel on the right side of the interface, showing a list of available components and libraries, including 'General', 'System Dynamics', 'Statechart', 'Actionchart', 'Analysis', 'Presentation', '3D', 'Controls', 'Connectivity', 'Enterprise Library', 'Pedestrian Library', 'Rail Library', 'Road Traffic Library - Preview', 'Pictures', '3D Objects', and 'Palettes...'.

Data for Custom Distribution

The screenshot displays the AnyLogic University software interface, which is used for building and simulating agent-based models. The main workspace shows a simulation model with various components like 'altitude', 'vegetation', 'DistrDisplacement', 'mapDrawing', 'altitudesDrawn', 'AngleTable', and 'viewVegetation'. The 'AngleTable' component is highlighted, and its properties are shown in the 'Properties' pane.

The 'AngleTable - Table Function' properties pane is open, showing the following settings:

- Name: AngleTable
- Access: public
- Static: checked
- Interpolation: Linear
- Out of Range: Error

The 'Table Data' section shows a table with two columns: 'Argument' and 'Value'.

Argument	Value
0	0
120	0.118
150	0.134
180	0.217
30	0.22
60	0.175
90	0.133

A line graph is also visible, plotting the values from the table against the arguments. The x-axis represents the argument (0 to 180) and the y-axis represents the value (0 to 0.4). The graph shows a red line connecting the data points, with a peak around 180 degrees.

Looking at body of this function
(method)

Heading Towards Resource

The screenshot shows the AnyLogic Advanced interface. On the left, a project tree lists 'Wandering Elephants*' with sub-items like 'Elephant', 'Parameters', 'drinkingPeriod: 100', 'smokingInitiationRateByAgeAn', 'Plain Variables', 'Statecharts', 'behavior', 'FreeWandering', 'GotThirsty', 'GoToWater', 'DrinkWater', 'initialState', 'state', 'NewDir', 'Functions', 'Presentation', 'Main', 'Parameters', 'NumberOfElephants: 50', and 'Plain Variables'. The 'Main' statechart is open, showing a state 'thirsty' with a 'headingRandom' function and a 'headingToWater' function. A red arrow points from the text 'Looking at body of this function (method)' to the 'headingToWater' function in the statechart. Another red arrow points from the text 'Determining current position & Searching for quickest way to find water from that position. (should be in separate function!)' to the 'headingToWater' function. A blue arrow points from the text 'Initiates movement towards chosen destination' to the 'moveTo' function in the 'headingToWater' function body.

Determining current position & Searching for quickest way to find water from that position.
(should be in separate function!)

headingToWater - Function

```
stop();
double x = getX();
double y = getY();

//find nearest water and set heading there
double dmin = Double.POSITIVE_INFINITY;
double heading = 0;
for ( double a = 0; a < 2 * Math.PI; a += Math.PI / 16 ) { // try 16 directions
    for ( double d = 0; d < 750; d += 5 ) {
        if ( d >= dmin )
            break; // we know better direction
        int c = (int) ( ( x + d * cos( a ) ) / 5 );
        int r = (int) ( ( y + d * sin( a ) ) / 5 );
        if ( c < 0 || 100 <= c || r < 0 || 100 <= r )
            break; // this is outside the area
        if ( get_Main().altitude[c][r] < 0 ) {
            dmin = d;
            heading = a;
            break;
        }
    }
}

//fixed high velocity
setVelocity( 5 );
//and start moving in the new direction to a virtual distant target - this will be stoppe
moveTo( x + 1000*cos( heading ), y + 1000*sin( heading ) );
```

Handling Agent Arrival at Destination (Not Currently Used in this Model)

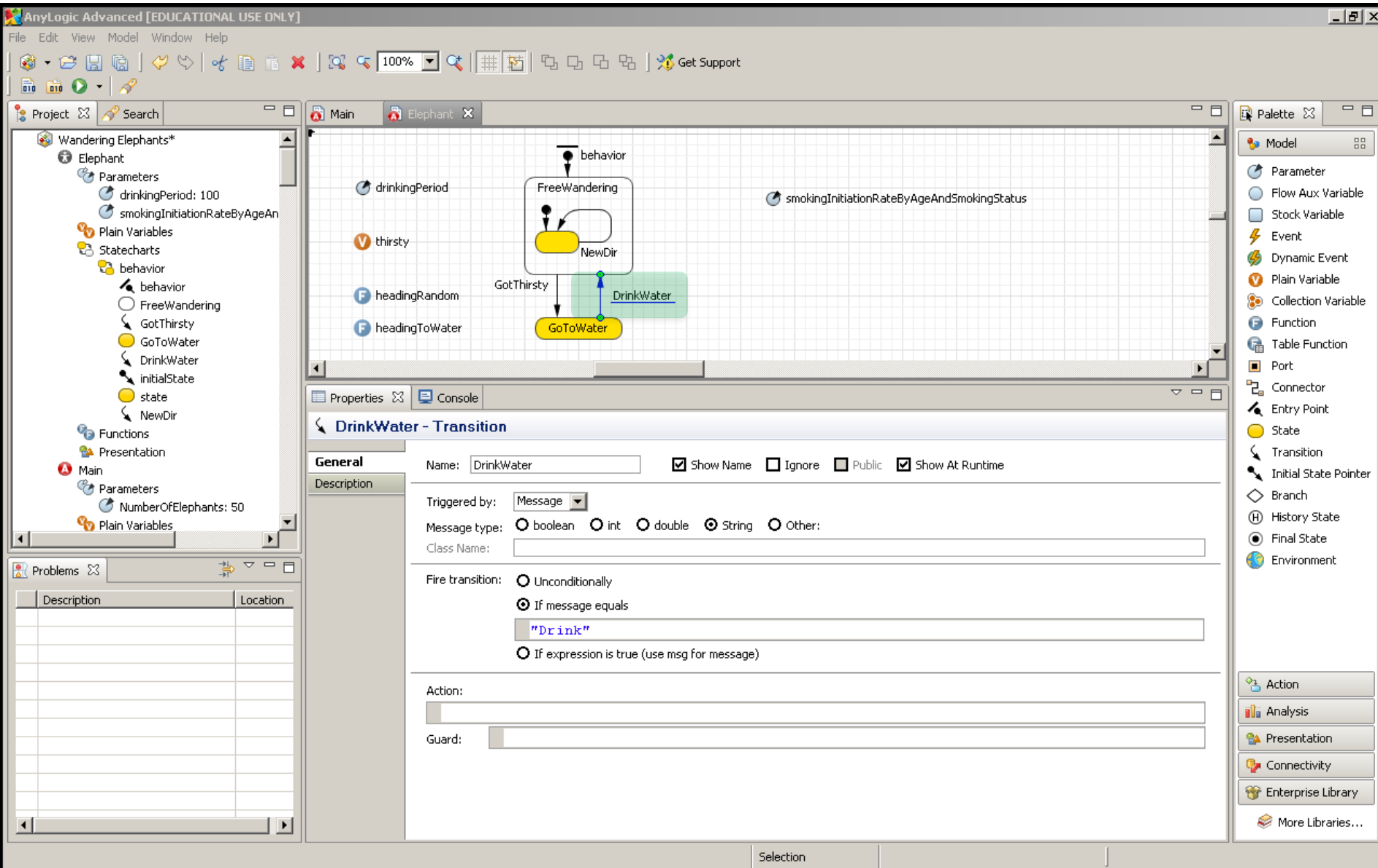
The screenshot displays the AnyLogic Advanced [EDUCATIONAL USE ONLY] interface. The main workspace shows a statechart for an agent named 'Elephant'. The statechart includes a 'FreeWandering' state with a self-loop labeled 'NewDir'. A transition labeled 'GotThirsty' leads from 'FreeWandering' to a 'GoToWater' state. The 'GoToWater' state has a return transition to 'FreeWandering'. The left sidebar shows a project tree with various components like 'Events', 'Presentation', 'Simulation: Main', and 'AgentFactory'. The bottom panel shows the 'Properties' window for the 'Elephant - Active Object Class'. The 'General' tab is selected, showing 'Space type' as 'Continuous' and 'Environment defines initial location' checked. The 'On Arrival:' section is highlighted in red, indicating the area for handling agent arrival at a destination.

“Handler”: Code fires when the specified event (here, arrival at a destination) occurs.

Handling Arrival Events in Statecharts

The screenshot displays the AnyLogic software interface. On the left is the 'Projects' tree showing a hierarchy of models and components. The main workspace shows a statechart for a 'Person' agent. The statechart includes two states: 'Susceptible' and 'Infective'. A red arrow points from the text 'Transition contingent on agent arrival' to the 'Agent arrival' option in the 'Fire transition' dropdown menu of the 'transition - Transition' properties panel. The properties panel also shows options for 'Triggered by' (Message, Timeout, Rate, Condition) and 'Message type' (int, double, String, Other). The 'Console' tab is visible below the properties panel.

Resumption of Wandering After Slaking Thirst



Handling of Movement Logic

AnyLogic Advanced [EDUCATIONAL USE ONLY]

File Edit View Model Window Help

100%

Get Support

Project Search

Elephant

- Parameters
 - drinkingPeriod: 100
 - smokingInitiationRateByAgeAn
- Plain Variables
- Statecharts
 - behavior
 - behavior
 - FreeWandering
 - GotThirsty
 - GoToWater
 - DrinkWater
 - initialState
 - state
 - NewDir
- Functions
- Presentation

Main

- Parameters
 - NumberOfElephants: 50
- Plain Variables
- Functions

Problems

Properties Console

Elephant - Active Object Class

General

Advanced

Agent

Parameters

Description

On Step:

```
if( ! isMoving() )
    error( "Not moving!" );

Main m = get_Main();

//where am I?
double x = getX();
double y = getY();
int c = min( max( 0, (int) (x/5) ), 99 );
int r = min( max( 0, (int) (y/5) ), 99 );

//drink if thirsty if in water
if( thirsty && m.altitude[c][r] < 0 )
    behavior.receiveMessage( "Drink" );

//demolish trees at current cell, if any
if( m.vegetation[c][r] > 10000 )
    m.vegetation[c][r] -= 10000;
```

Finding location in continuous space (x,y) & in terms of Discrete vegetation Space (c,r).

Handling the case of reaching water when thirsty

Poor style -- Should be In separate function

Rerouting Around Barriers (Boundaries & Water)

Poor Style – entire logic, conditions (checks on boundaries, whether water) & rerouting
Logic should all be in separate functions from this & from each other). Remove constants

The screenshot displays the AnyLogic Advanced software interface, specifically the 'Elephant' model. The interface is divided into several panes:

- Project Pane (Left):** Shows the project structure for 'Elephant', including Parameters (drinkingPeriod: 100, smokingInitiationRateByAgeAn), Plain Variables, Statecharts, behavior (FreeWandering, GotThirsty, GoToWater, DrinkWater, initialState, state, NewDir), Functions, Presentation, and Main (NumberOfElephants: 50, Plain Variables, Functions).
- Main Canvas (Center):** Displays a statechart for the 'Elephant' model. It features a 'FreeWandering' state with a self-loop labeled 'NewDir'. Transitions include 'GotThirsty' leading to a 'GoToWater' state and 'DrinkWater' leading back to 'FreeWandering'. Other variables shown are 'drinkingPeriod', 'thirsty', 'headingRandom', 'headingToWater', and 'smokingInitiationRateByAgeAndSmokingStatus'.
- Properties Pane (Bottom Left):** Shows the 'Elephant - Active Object Class' properties, including General, Advanced, Agent, Parameters, and Description tabs.
- Console Pane (Bottom Center):** Displays the Java code for the 'Elephant' model, showing logic for avoiding bounds and water, changing direction if needed, and moving to a new location.
- Palette (Right):** Lists various modeling elements such as Parameter, Flow Aux Variable, Stock Variable, Event, Dynamic Event, Plain Variable, Collection Variable, Function, Table Function, Port, Connector, Entry Point, Entry Point, State, Transition, Initial State Pointer, Branch, History State, Final State, and Environment.

The Java code in the Console pane is as follows:

```
m.vegetation[c][r] -= 10000;  
  
//avoid bounds and water, change direction if needed  
if( x < 0 || x >= 500 || y < 0 || y >= 500 || m.altitude[c][r] < 0 ) {  
    stop();  
    //try new heading until find a valid one  
    double heading;  
    double xtry, ytry;  
    int count = 0;  
    do {  
        if( count >= 100 ) {  
            error( "Count not find way out!" );  
        }  
        heading = uniform( -Math.PI, Math.PI );  
        xtry = x + 10 * cos( heading );  
        ytry = y + 10 * sin( heading );  
        count++;  
    } while( xtry < 0 || xtry >= 500 || ytry < 0 || ytry >= 500 || m.altitude[(int)(xtry/  
//and start moving in the new direction to a virtual distant target - this will be st  
moveTo( x + 1000*cos( heading ), y + 1000*sin( heading ) );  
}
```

Environment: Updating Vegetation

The screenshot displays the AnyLogic Advanced software interface, specifically the 'updateVegetation' function editor. The main workspace shows a grid with various variables and functions, including 'vegetation', 'DistrDisplacement', 'mapDrawing', 'DistrAngle', 'altitudesDrawn', 'altitudeImage', and 'vegetationDrawn'. The 'updateVegetation' function is highlighted in the workspace and its code is visible in the 'Code' tab of the Properties panel.

Project Explorer (Left):

- Main
 - Parameters
 - NumberOfElephants: 50
 - Plain Variables
 - Functions
 - AngleTable
 - DisplacementTable
 - altitudeToColor
 - makeUpAltitudes
 - makeUpVegetation
 - placeElephants
 - updateVegetation
 - vegetationToColor
 - Environments
 - environment
 - Embedded Objects
 - elephants
 - Presentation
 - Simulation: Main
 - Presentation
 - ABMClinicModelV6
 - Intern

Properties Panel (Bottom Left):

- General
- Code
 - Function body:

```
for ( int i = 0; i < 100; i++ )
    for ( int j = 0; j < 100; j++ )
        if ( vegetation[i][j] > 0 )
            vegetation[i][j] = limitMax( vegetation[i][j] + 15, ( 40 - altitude[i][j] ) * 1
//reset flag
vegetationDrawn = false;
```
- Description

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...

Continuous Space: Relevant Methods (To call on *Agent*)

- Already covered
 - moveTo(x,y) : initiates agent movement to location
 - setVelocity(v)
- Basic info
 - getX()/getY()
 - setXY(x,y): initial location
 - jumpTo(x,y): moves agent to location
 - isMoving()
 - getTargetX()/getTargetY()
 - Where heading to?
 - setRotation()/ getRotation()

Environment Happens to Handle Process of Maintaining Environmental Dynamics

The screenshot displays the AnyLogic Advanced software interface, specifically the 'environment' component configuration window. The interface is divided into several panels:

- Project Panel (Left):** Shows a hierarchical tree of the project structure. The 'environment' component is selected under the 'Main' project.
- Diagram Canvas (Center):** Displays a grid with various components and their connections. Components include 'vegetation', 'DistrDisplacement', 'mapDrawing', 'DistrAngle', 'altitudesDrawn', 'updateVegetation', 'altitudeImage', and 'vegetationDrawn'. A small map preview is visible on the right side of the canvas.
- Properties Panel (Bottom):** Shows the configuration for the 'environment' component. The 'General' tab is active, displaying the following settings:
 - Name: environment
 - Show Name: ☒ (checked)
 - Ignore: ☐ (unchecked)
 - Public: ☐ (unchecked)
 - Show At Runtime: ☐ (unchecked)
 - Enable steps: ☒ (checked)
 - Step duration (in model units): [empty field]
 - On before step: [empty field]
 - On after step: `updateVegetation();`
- Palette (Right):** Lists various components available for use in the model, including 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.
- Problems Panel (Bottom Left):** A table with columns 'Description' and 'Location' for reporting errors or warnings.

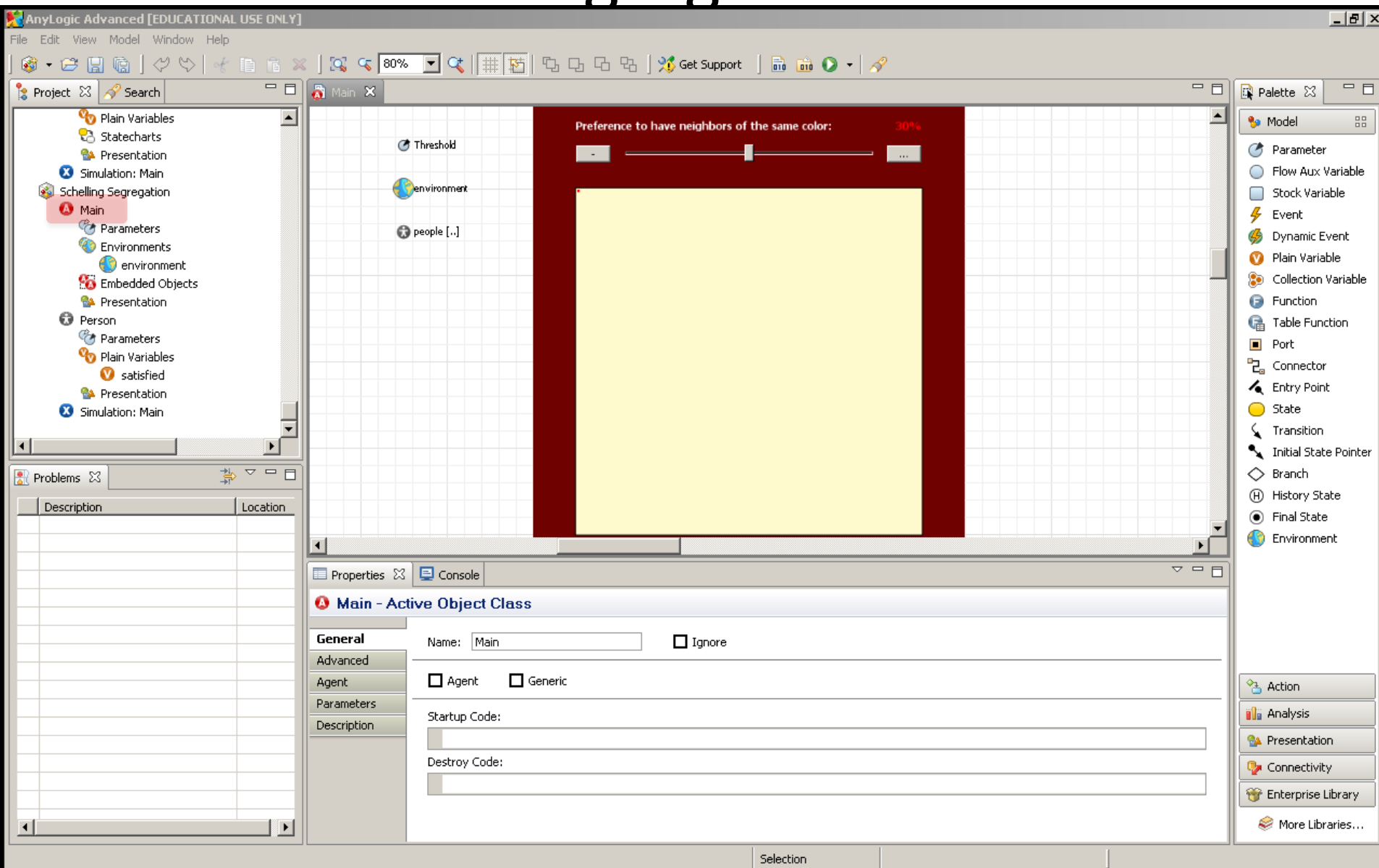


Hands on Model Use Ahead



Load model: Schelling Segregation.alp

A Model to Examine the Emergence of Segregation



A Discrete Spatial Environment with Random Agent Positioning

The screenshot displays the AnyLogic Advanced software interface, specifically the 'environment - Environment' properties window. The main workspace shows a grid-based environment with a red rectangle labeled 'Preference to have neighbors of the same color:' and a yellow rectangle. A 'Person' agent is positioned on the grid, with a 'Threshold' parameter and a 'people [...]' collection variable associated with it.

The 'environment - Environment' properties window is open, showing the 'Advanced' tab. The 'Space type' is set to 'Discrete'. The 'Width' and 'Height' are both set to 400. The 'Columns' and 'Rows' are both set to 100. The 'Neighborhood type' is set to 'Moore'. The 'Layout type' is set to 'Random', and the 'Apply on startup' checkbox is checked. The 'Network type' is set to 'User-defined', and the 'Apply on startup' checkbox is unchecked. The 'Connections per agent' is set to 2, the 'Connection range' is set to 50, and the 'Neighbor link fraction' is set to 0.95.

Annotations highlight the 'Spatial Width & Height' (400) and the 'Width & Height in Discrete Cells' (100 columns and 100 rows).

On the left, the 'Project' pane shows a tree structure with 'Main' selected. On the right, the 'Palette' pane shows various modeling elements 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'.

Population Dependence on the Population

The screenshot displays the AnyLogic Advanced software interface, titled "AnyLogic Advanced [EDUCATIONAL USE ONLY]". The main workspace shows a grid-based model of a Schelling Segregation simulation. A red rectangular area on the right side of the grid is labeled "Preference to have neighbors of the same color:". Below this area, a yellow rectangular area is visible. The workspace contains several objects: a "Threshold" object, an "environment" object, and a "people [...]" object. 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 of the interface features a "Properties" panel and a "Console" panel. The "Properties" panel is currently showing the properties for the "people - Person" object. The "Console" panel is empty.

people - Person Properties:

- General:** Name: people, ☒ Show Name, ☐ Ignore, ☐ Public, ☒ Show At Runtime, [Create Presentation](#)
- Parameters:**
- Statistics:**
- Description:**
- Type: Person
- Package: schelling_segregation
- Environment: environment
- Color: `randomTrue(0.5) ? Color.red : Color.black`
- Replication: 8000

The right sidebar shows a "Palette" with various model components:

- 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

The screenshot displays the AnyLogic Advanced interface for a Schelling Segregation model. A slider input is connected to a 'Threshold' parameter. The slider's 'Default Value' is set to 'Threshold', and its 'Action' is 'Threshold = value;'. A green arrow points from the 'Threshold' parameter to the slider, and a blue arrow points from the 'Default Value' field to the 'Threshold' parameter. A red arrow points from the 'Action' field to the 'Threshold' parameter.

“Threshold” parameter

Default value is that of Threshold parameter

Sets Threshold Parameter Value

Properties of slider - Slider:

- Name: slider
- Orientation: ☒ Horizontal ☐ Vertical
- Minimum Value: 0
- Maximum Value: 1
- Default Value: Threshold
- Enabled: ☐
- Action: Threshold = value;

Person is Assigned a Randomly Picked Color

The screenshot displays the AnyLogic Advanced [EDUCATIONAL USE ONLY] interface. The main workspace shows a 'Person' object with a 'color' parameter. A red arrow points to the 'Person' object, labeled 'Person's Visual Representation'. A blue arrow points to the 'color' parameter, labeled 'Color is set to either red or black with equal likelihood'. The 'color - Parameter' properties window is open, showing the 'Name' as 'color', 'Type' as 'Color', and 'Default Value' as `randomTrue(0.5) ? Color.red : Color.black`. The 'Properties' tab is selected, and the 'General' section is visible. The 'Type' is set to 'Color' and the 'Default Value' is set to `randomTrue(0.5) ? Color.red : Color.black`. The 'Dynamic' checkbox is unchecked, and the 'Save in snapshot' checkbox is checked. The 'On Change' field is empty.

AnyLogic Advanced [EDUCATIONAL USE ONLY]

File Edit View Model Window Help

100%

Get Support

Project Search

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...

Properties Console

color - Parameter

General

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:

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...

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:

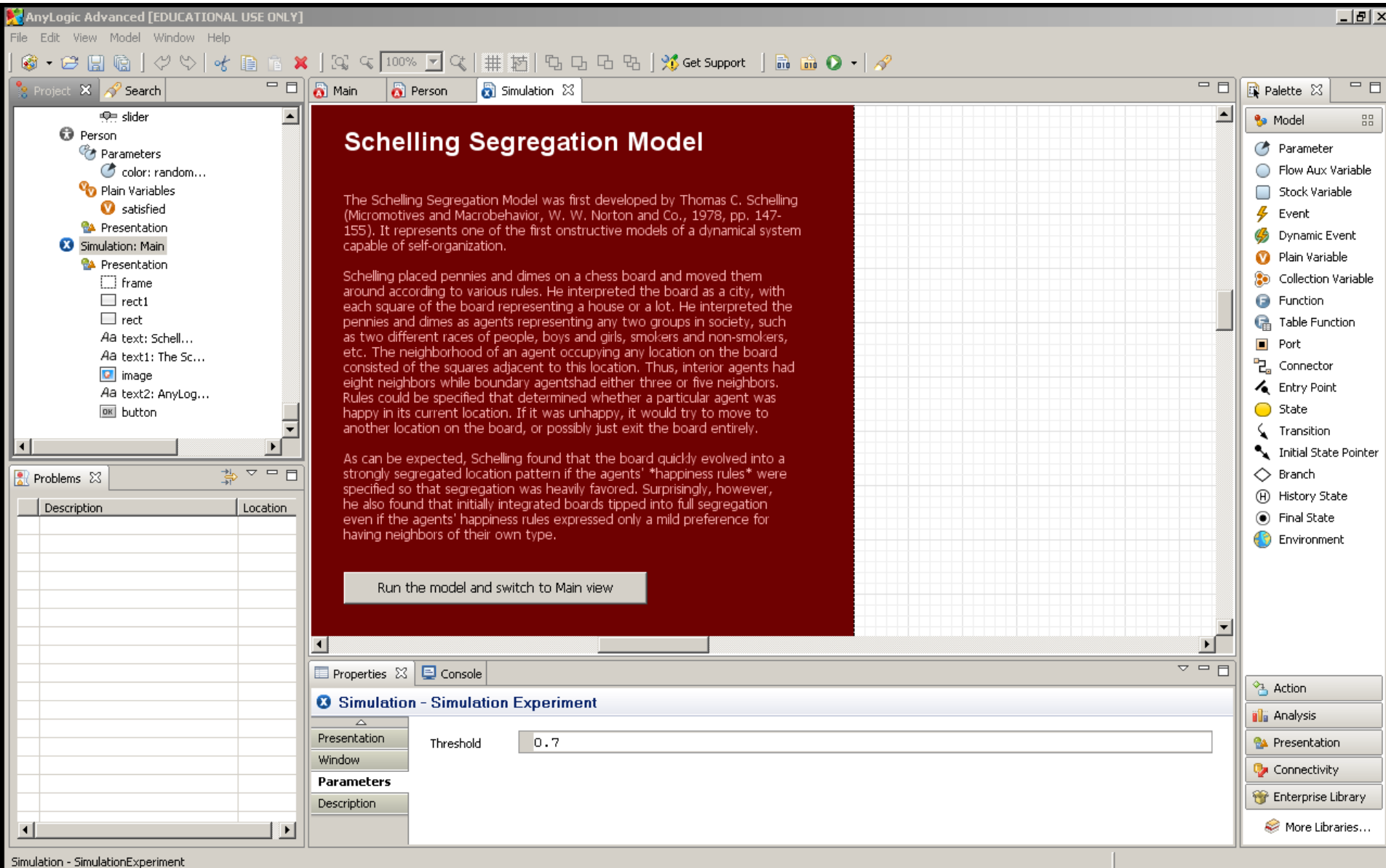
```
if( ! satisfied && randomTrue( 0.3 ) )
    jumpToRandomEmptyCell();
```

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

Parameter Assumptions



Add a Parameter to Main

The screenshot displays the AnyLogic Advanced software interface, specifically the 'Main' model window. The interface includes a menu bar (File, Edit, View, Model, Window, Help), a toolbar with various icons, and a project browser on the left. The project browser shows a hierarchy of objects: Threshold (0.7), likelihoodOfMovementIfDissatisfied (C), Environments, environment, Embedded Objects, people, Presentation, rect, rect1, people_presentation, text: Prefer..., text1: 30%, button, button1, slider, Person, Parameters, and color: random... The main workspace shows a diagram with a red rectangle labeled 'Preference to have neighbors of the same color: 30%' and a yellow rectangle. The 'likelihoodOfMovementIfDissatisfied' parameter is highlighted in the project browser. The 'Properties' panel at the bottom shows the details for the 'likelihoodOfMovementIfDissatisfied' parameter, including its name, type (double), default value (0.3), and options for dynamic behavior and saving in snapshots.

AnyLogic Advanced [EDUCATIONAL USE ONLY]

File Edit View Model Window Help

100%

Get Support

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

Action

Analysis

Presentation

Connectivity

Enterprise Library

More Libraries...

Properties Console

likelihoodOfMovementIfDissatisfied - Parameter

General

Name: likelihoodOfMovementIfDissatisfied ☒ Show Name ☐ Ignore ☐ Public ☒ Show At Runtime

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

Default Value: 0.3

☐ Dynamic ☒ Save in snapshot

On Change:

Description

Selection

Experiment: Add a Slider!

The screenshot displays the AnyLogic Advanced software interface, titled "AnyLogic Advanced [EDUCATIONAL USE ONLY]". The main window shows the "Schelling Segregation Model" simulation. The interface includes a menu bar (File, Edit, View, Model, Window, Help), a toolbar with various icons, and a status bar at the bottom.

The central workspace is divided into two main sections. The left section, titled "Schelling Segregation Model", contains text describing the model's history and rules. The right section is a large grid area for the simulation.

The left sidebar contains a "Project" pane with a tree view showing the model's structure:

- Model
 - Parameters
 - color: random...
 - Plain Variables
 - satisfied
 - Presentation
 - frame
 - rect1
 - rect
 - Simulation: Main
 - Aa text: Schell...
 - Aa text1: The Sc...
 - image

The bottom-left pane shows a "Problems" table with columns "Description" and "Location".

The bottom-right pane shows a "Properties" and "Console" area.

The right sidebar contains a "Palette" with various UI elements and a "More Libraries..." button.

The main text in the simulation window reads:

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

Setting the Slider Properties

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a red background with text describing Schelling's model of segregation. A slider control is visible in the workspace, and its properties are shown in the bottom panel.

Project Explorer (Left):

- 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

Properties Panel (Bottom):

sliderMovementChance - Slider

General

Name: ☐ Show Name ☐ Ignore ☒ Public ☐ Icon

Advanced

Orientation: ☒ Horizontal ☐ Vertical

Minimum Value:

Maximum Value:

Default Value:

Enabled: ☒ true

Description

Action:

Console (Bottom):

Run the model and switch to Main view

Setting Value for Parameter from Slider

The screenshot displays the AnyLogic Advanced software interface, which is used for building and simulating agent-based models. The interface is divided into several panes:

- Project Pane (Left):** Shows the model's structure, including a 'Person' entity with parameters like 'color: random...', 'Plain Variables' like 'satisfied', and a 'Simulation: Main' entity with various presentation elements like 'frame', 'rect1', 'rect', 'text', 'image', 'button', and 'sliderMovementChance'.
- Model Canvas (Center):** Displays the simulation environment. A red rectangular area represents a 'Preference to have neighbors of the same color:' with a slider set to 30%. Below this, a yellow rectangular area is visible. The canvas also shows a 'Threshold' parameter and a 'likelihoodOfMovementIfDissatisfied' parameter.
- Properties Pane (Bottom):** Shows the 'Simulation - Simulation Experiment' properties. The 'General' tab is active, displaying the 'Name' as 'Simulation' and the 'Main active object class (root)' as 'Main'. The 'Random number generation' section shows 'Fixed seed (reproducible simulation runs)' selected with a 'Seed Value' of 1. The 'Threshold' is set to 0.7, and the 'likelihoodOfMovementIfDissatisfied' is set to 'sliderMovementChance.value'.
- Palette (Right):** Contains a list of model components such as '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'.

The 'Simulation - Simulation Experiment' properties pane is expanded, showing the following details:

- General:** Name: Simulation, Main active object class (root): Main, Ignore: ☐
- Advanced:** Random number generation: ☐ Random seed (unique simulation runs), ☒ Fixed seed (reproducible simulation runs), Seed Value: 1
- Model Time:**
- Presentation:**
- Window:**
- Parameters:**
- Description:** Threshold: 0.7, likelihoodOfMovementIfDissatisfied*: sliderMovementChance.value, Paste from clipboard

Modify Person's Behavior to Depend on New Parameter

The screenshot displays the AnyLogic Advanced [EDUCATIONAL USE ONLY] interface. The main workspace shows a grid with two objects: 'color' and 'satisfied'. A green text box with an arrow points from the text to the 'get_Main()' method call in the code.

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

Person - Active Object Class

```
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();
```

The interface includes a Project tree on the left, a Palette on the right, and a Console window at the bottom.

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 a 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()`

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

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.
- Issue only arises for discrete agent movement, as this is the only case where cells are limited to contain 1 agent

Irregular Spatial Embedding



Realizing Irregular Spatial Embedding in AnyLogic

- Basic idea: people moving around follow networks of *paths*
- Irregular spatial embedding is supported directly by “Network Based Modeling” (Discrete Event Simulation)
 - This approach is individual-based, but treats agents either as flowing through and being operated on by a process or as (often interchangeable) process resources
 - We will have a brief introduction to this approach later in the week, showing how it can be combined with ABM
- With a modest amount of custom coding, irregular spatial embedding can be achieved within ABM
 - A guest lecture with an Alzheimer’s application will give a glimpse as to how this can be achieved