

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

Nathaniel Osgood

Using Modeling to Prepare for Changing
Healthcare Needs

Duke-NUS

April 16, 2014

Reminder: Agent Spatial Embedding

- Spatial embedding of agents is key to
 - Expressing essential dynamics for problems
Locality of influence/Transmission
 - Insight into certain phenomena (spatial concentration, percolation, spatial reference modes)
- Spatial embedding can permit GIS integration

2D Spatial Embedding: Two Options

- Continuous embedding (e.g. Wandering elephants, our built-up model)
 - No physical exclusion: Agents are assumed to be small compared to landscape scale, and exhibit arbitrary spatial density without interfering
 - We have seen this much with distributing agents initially around the space, adding agents in
- Discrete cells (e.g. The Game of Life, Agent-based predator prey, Schelling Segregation)
 - Divided into “Columns” and “Rows”
 - Physical exclusion: Only one agent in a cell at a time

The Locus of Control: Environment

- The Anylogic Environment sets the parameters for the nature of the 2D landscape
 - Width
 - Breadth
 - Continuous vs. Discrete
 - Character of discrete neighbourhoods (cardinal directions vs. Euclidian { N,NE,E,SE,S,SW,W,NW})

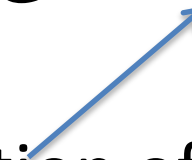
Reminder: Common Division

- Endogenous
 - Things whose dynamics are calculated as part of the model
- Exogenous
 - Things that are included in model consideration, but are specified externally
 - Time series
 - Constants
- Ignored/Excluded
 - Things outside the boundary of the model

Motivations for Including Endogenous Factors

- Maintaining factors as endogenous (rather than pre-specified as exogenous) lends
 - Extra flexibility for more accurately capturing effects of
 - Interventions
 - Alternative exogenous scenarios
 - Greater robustness in the context of changes
 - Support for translations to other contexts
- Keeping greater detail requires more data & implementation work, but allows our models to be translated to other contexts & times

Example

$$\begin{bmatrix} x_{11} & x_{12} & 1 - x_{11} - x_{12} \\ x_{21} & x_{22} & 1 - x_{21} - x_{22} \\ x_{31} & x_{32} & 1 - x_{31} - x_{32} \end{bmatrix}$$


- Mixing Matrix (specifies fraction of population A's contact that occur with populations B & C)
- Preference matrix
 - Scales to capture fluctuating population captures relative preference
 - can't specify where to test
- Mobility-based methods with mobility patterns hard-coded
 - this is challenged for interventions which change e.g. mixing opportunities and mobility
- Mobility-based methods with preference-based mobility model

Agent Mobility

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

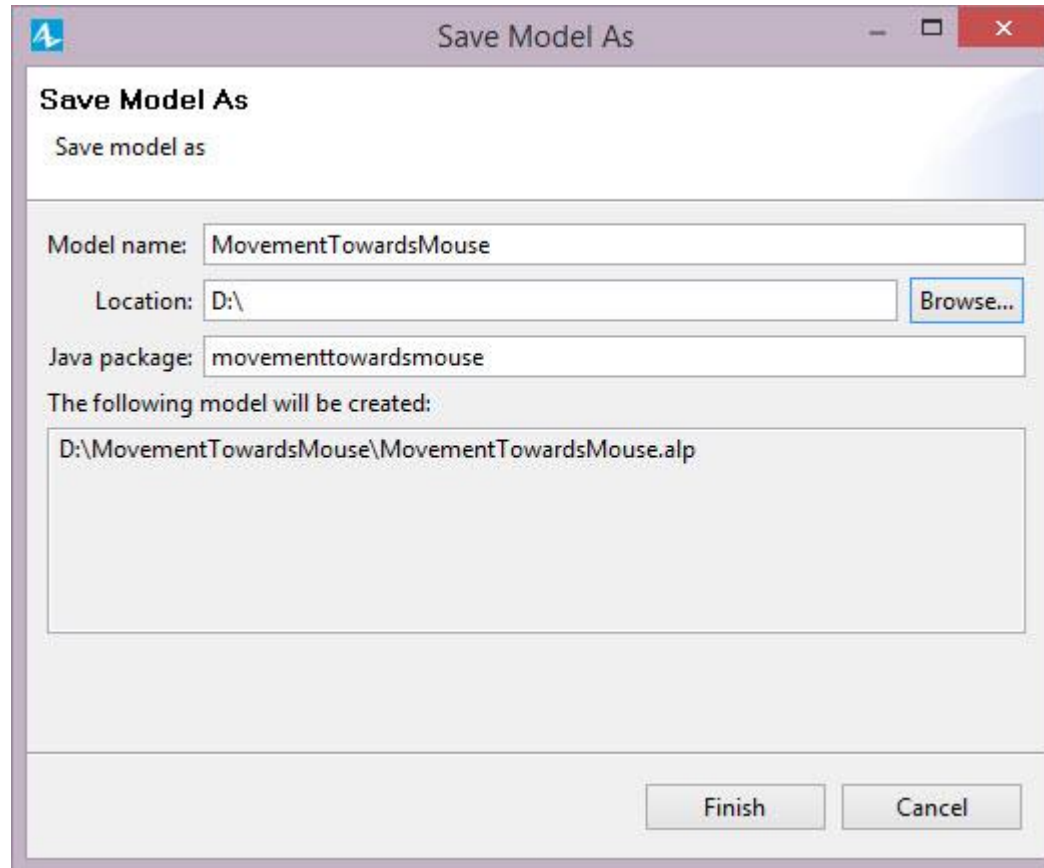
2D Spatial Embedding: Mobility Implications

- 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

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

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

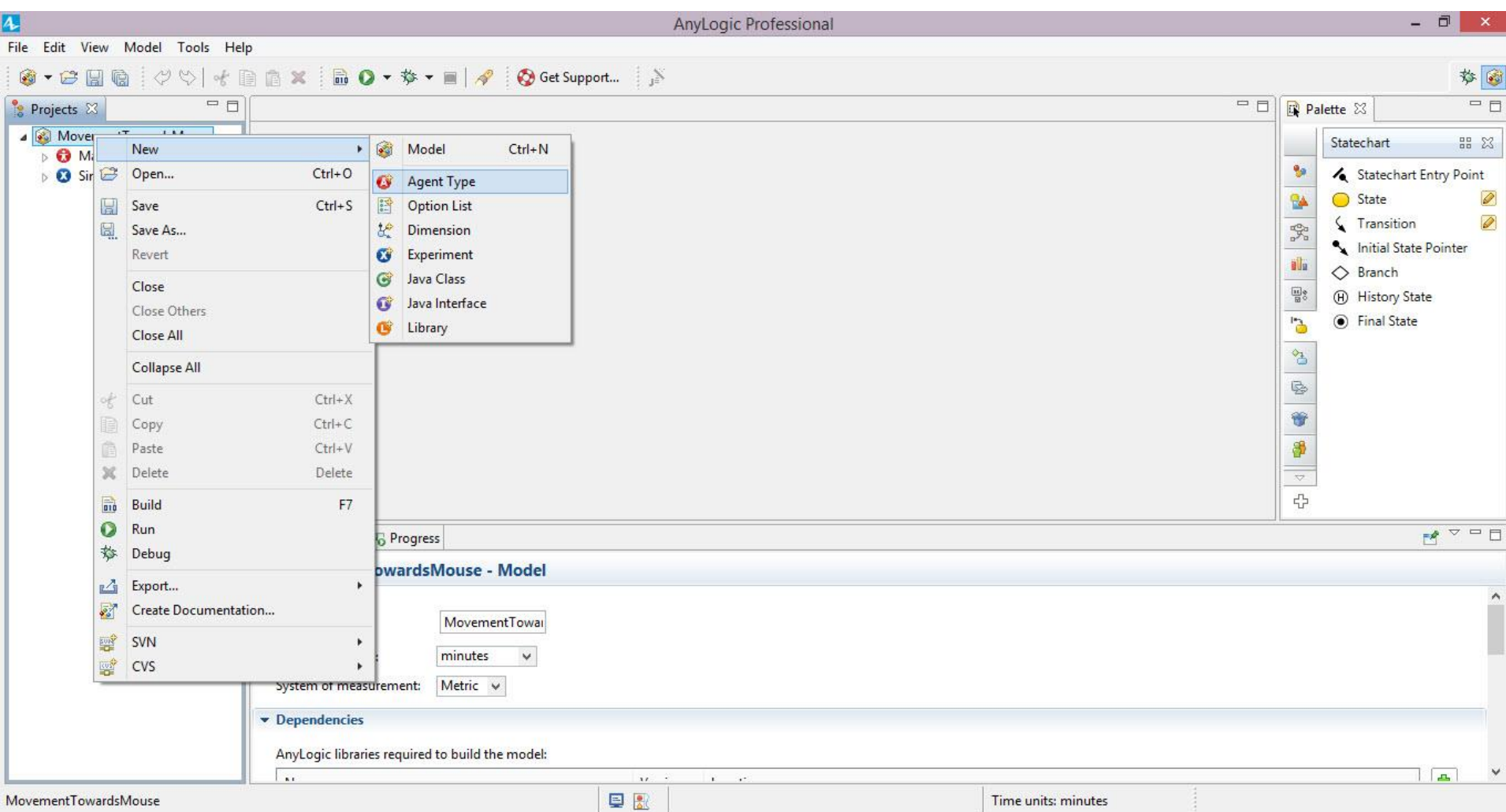
Create a New Project Called “MovementTowardsMouse”



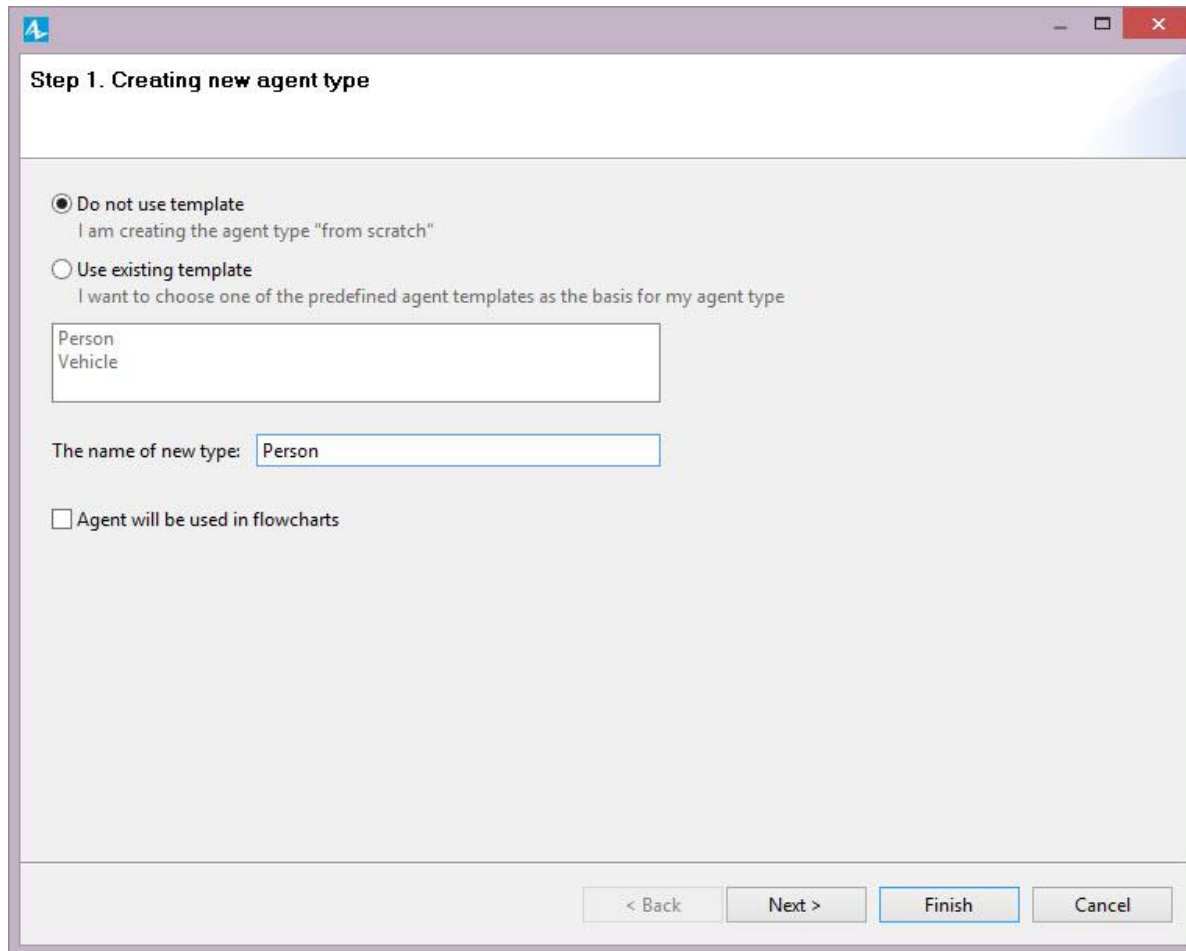
The screenshot shows a 'Save Model As' dialog box with the following fields and options:

- Model name:** MovementTowardsMouse
- Location:** D:\ (with a 'Browse...' button to the right)
- Java package:** movementtowardsmouse
- The following model will be created:**
D:\MovementTowardsMouse\MovementTowardsMouse.alp
- Buttons:** 'Finish' and 'Cancel' at the bottom right.

Add a New Active Object Class



Class Properties



The screenshot shows a software window with a title bar containing a blue icon with the number '4' and standard window controls (minimize, maximize, close). The window's title is 'Step 1. Creating new agent type'. The main area contains two radio button options. The first option, 'Do not use template', is selected and includes the text 'I am creating the agent type "from scratch"'. The second option, 'Use existing template', is unselected and includes the text 'I want to choose one of the predefined agent templates as the basis for my agent type'. Below the second option is a list box containing 'Person' and 'Vehicle'. Further down, a text field is labeled 'The name of new type:' and contains the text 'Person'. At the bottom left, there is an unchecked checkbox labeled 'Agent will be used in flowcharts'. The bottom right of the window features four buttons: '< Back', 'Next >', 'Finish' (which is highlighted with a blue border), and 'Cancel'.

Step 1. Creating new agent type

☒ **Do not use template**
I am creating the agent type "from scratch"

☐ **Use existing template**
I want to choose one of the predefined agent templates as the basis for my agent type

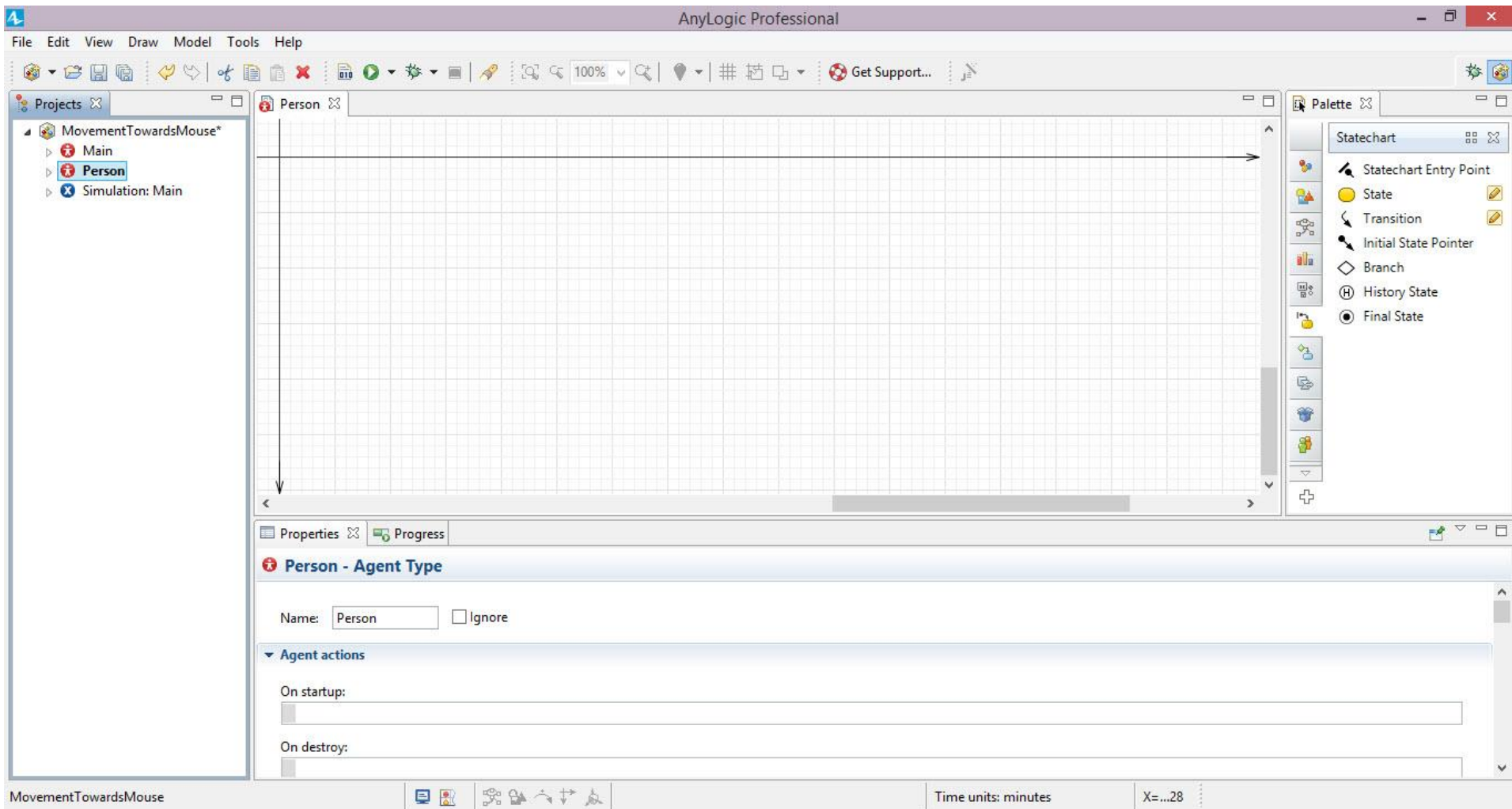
Person
Vehicle

The name of new type: Person

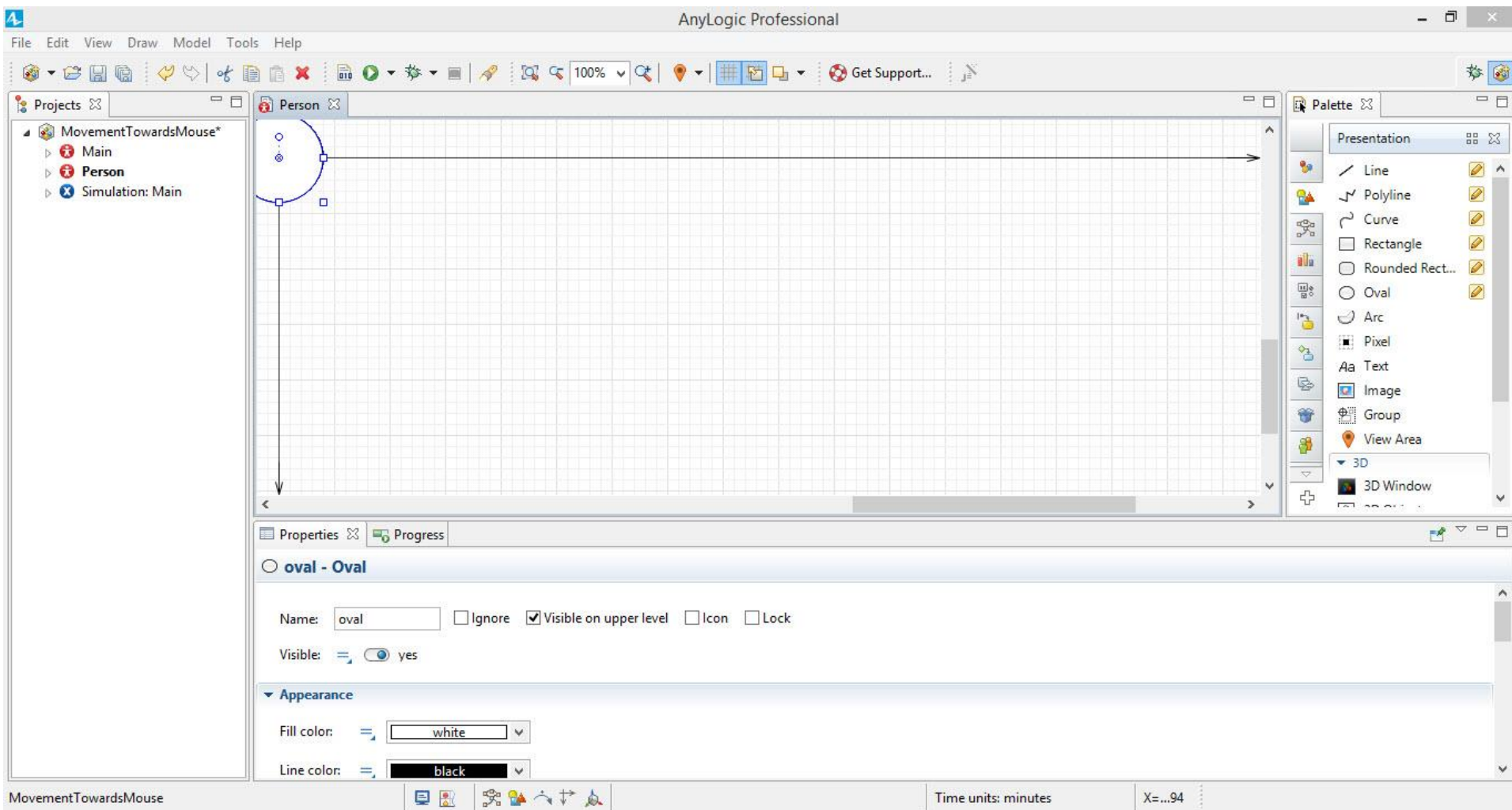
☐ Agent will be used in flowcharts

< Back Next > **Finish** Cancel

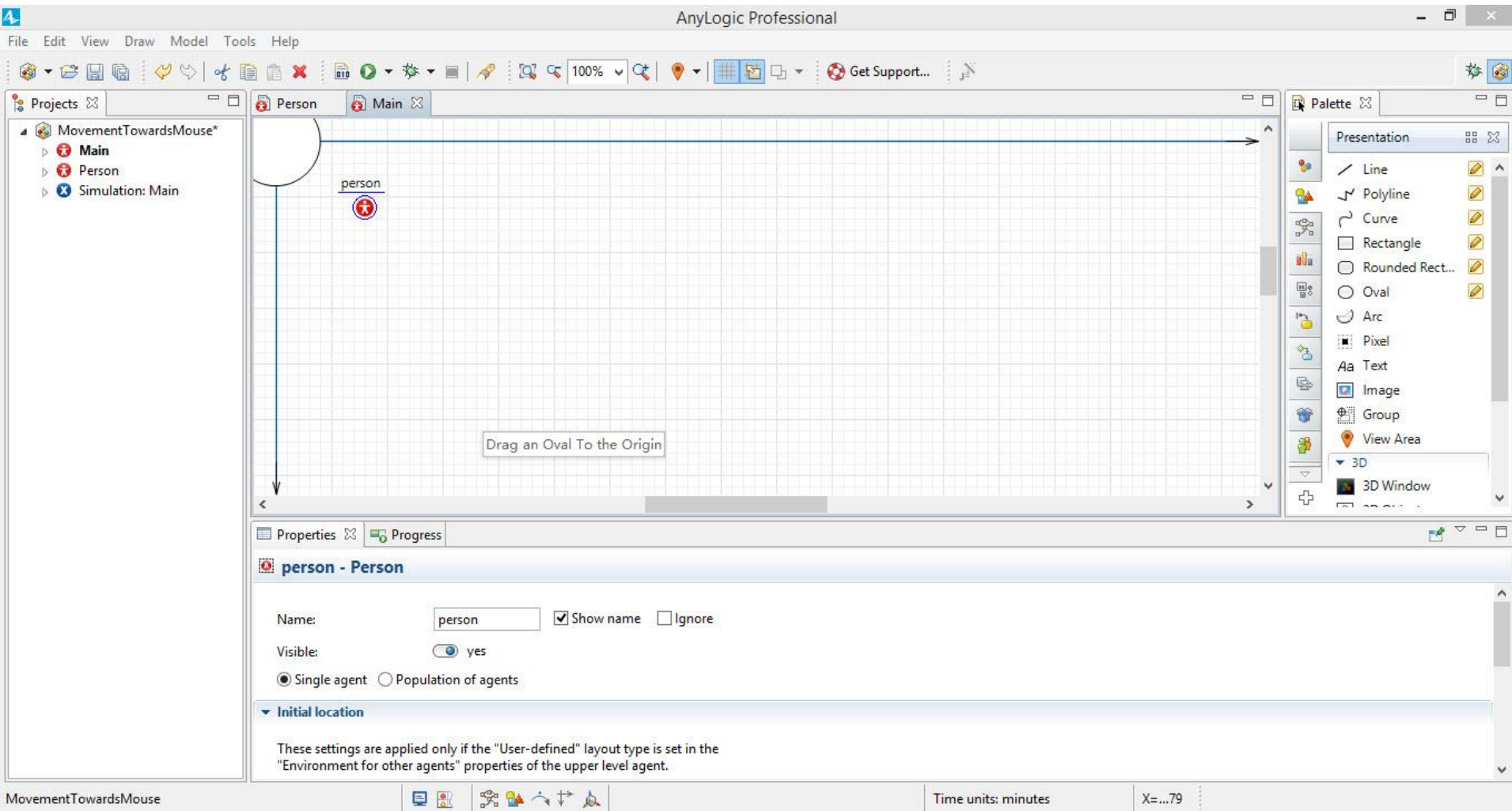
Once “Person” Class is Added



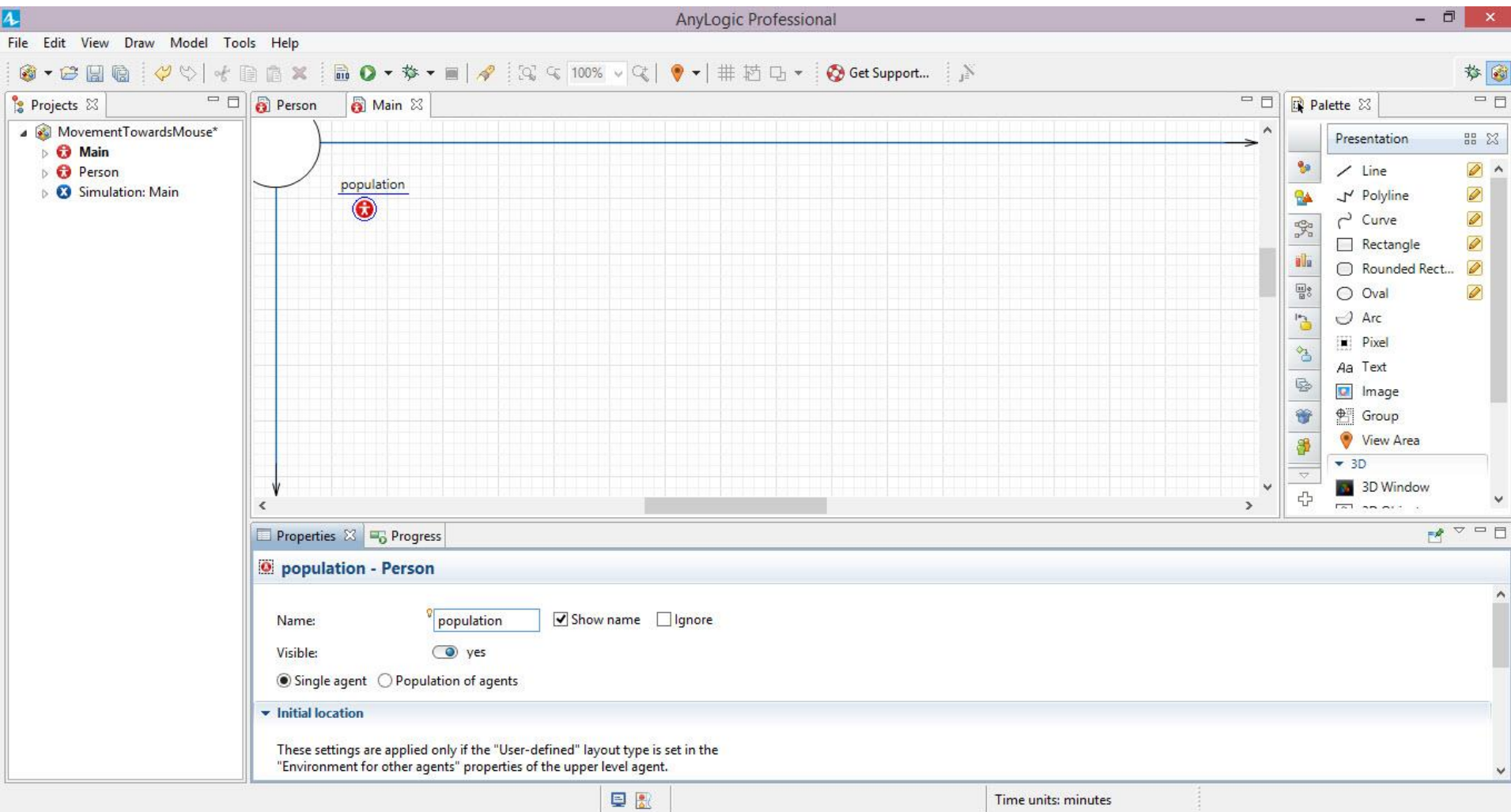
Drag an Oval To the Origin



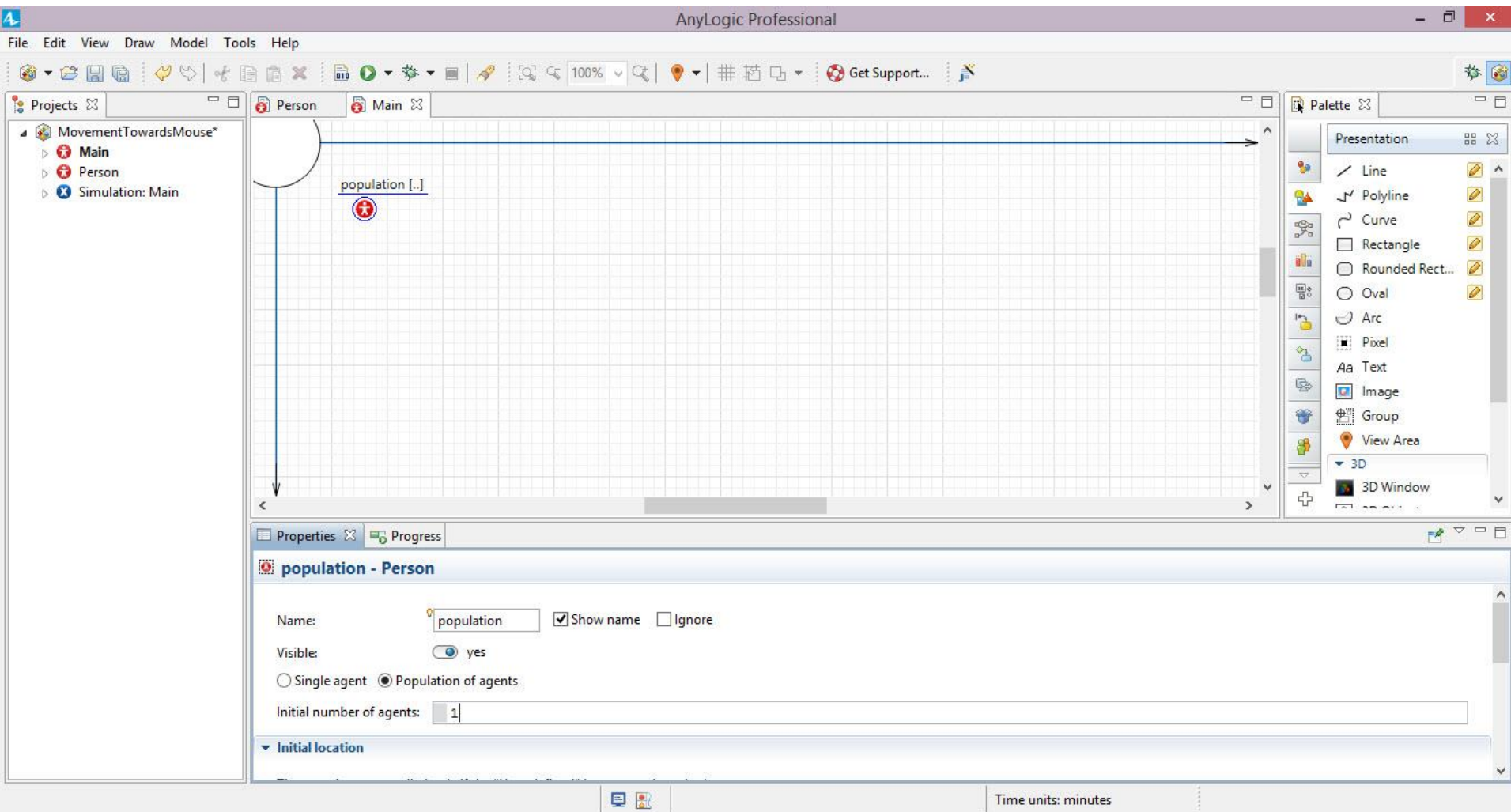
Drag “Person” Class into “Main” to Create a Population



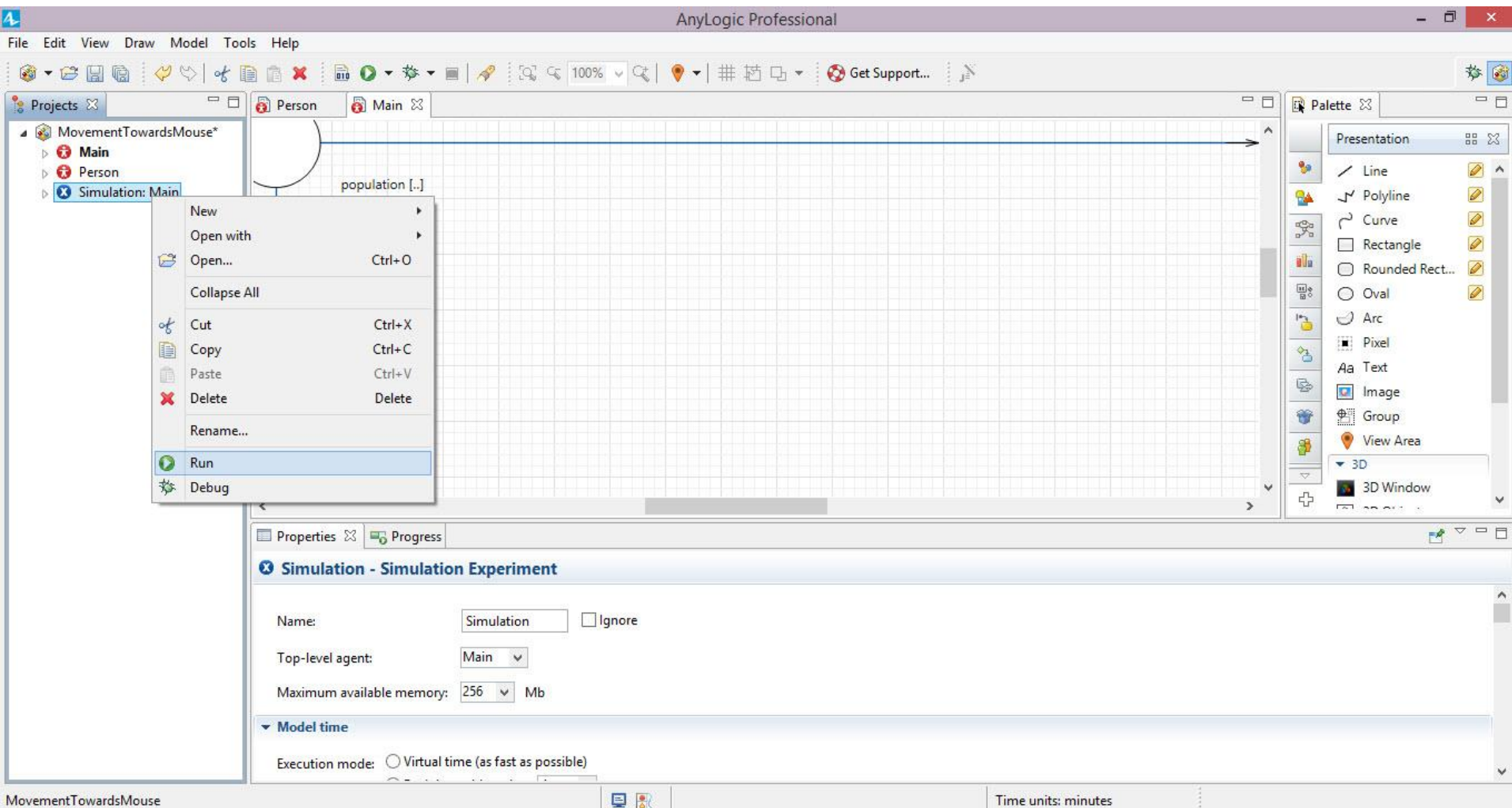
Name it “population”



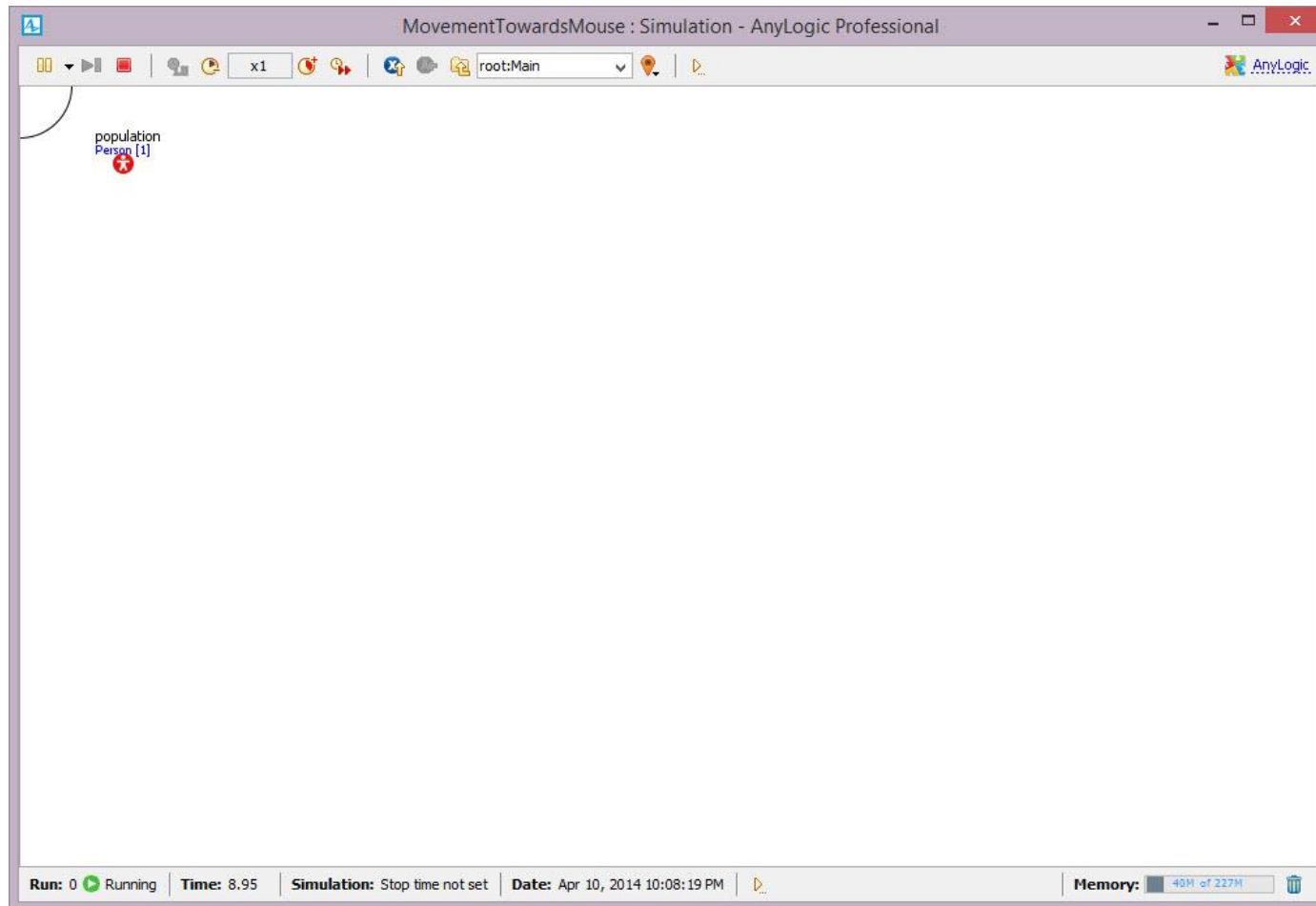
Set Population as Replicated (size 1 – for now)



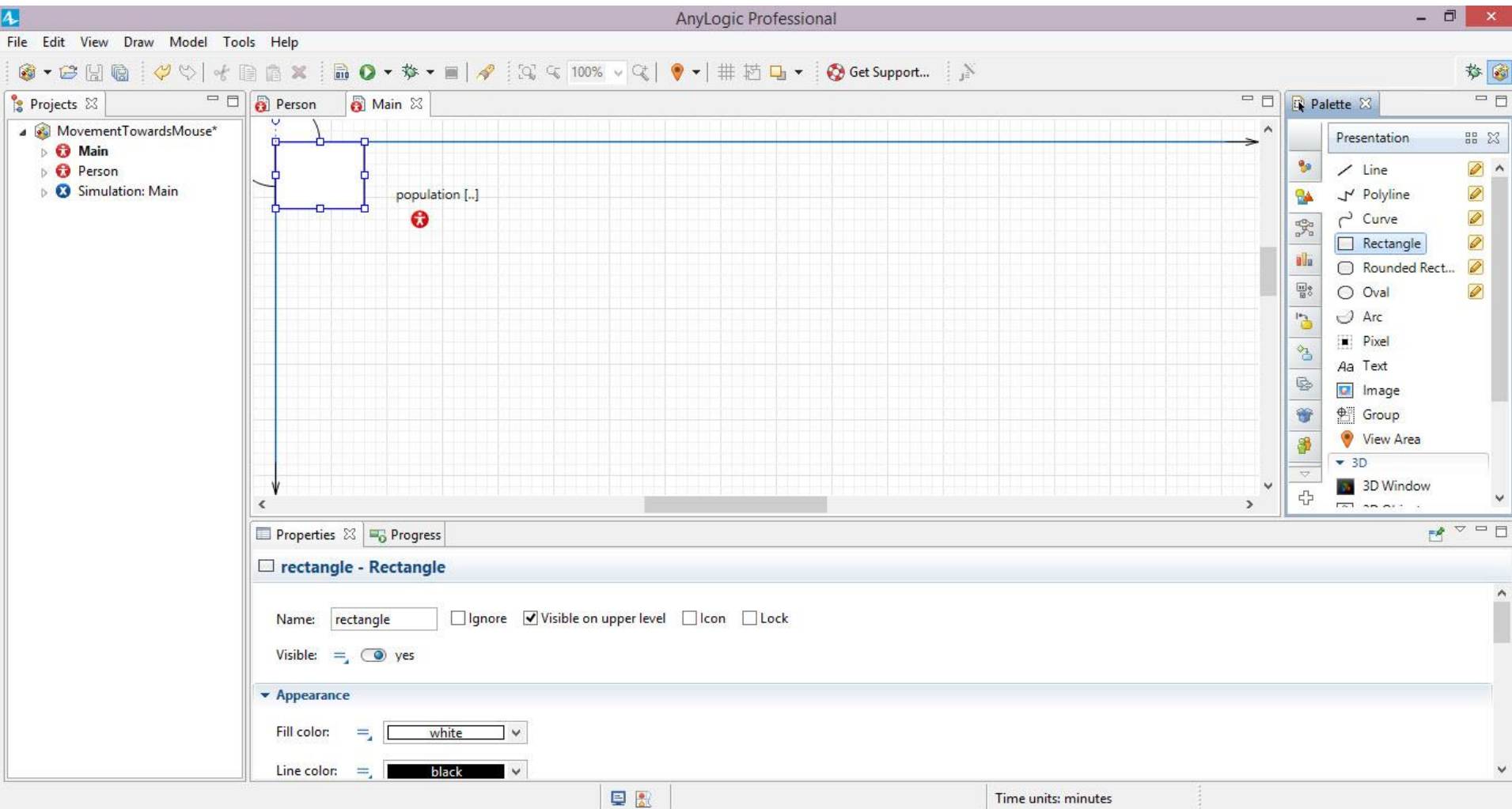
Run Experiment



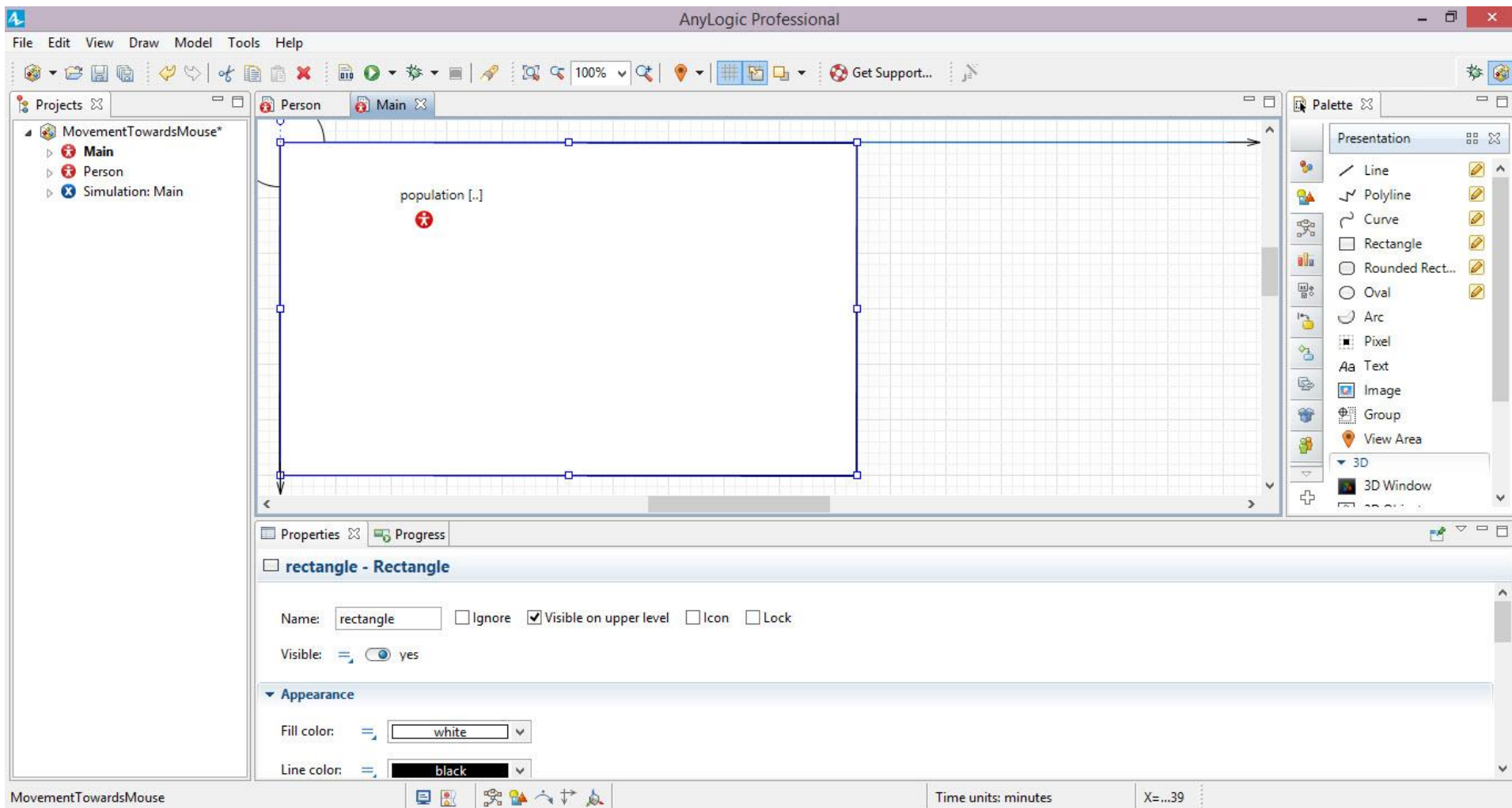
Model Appearance



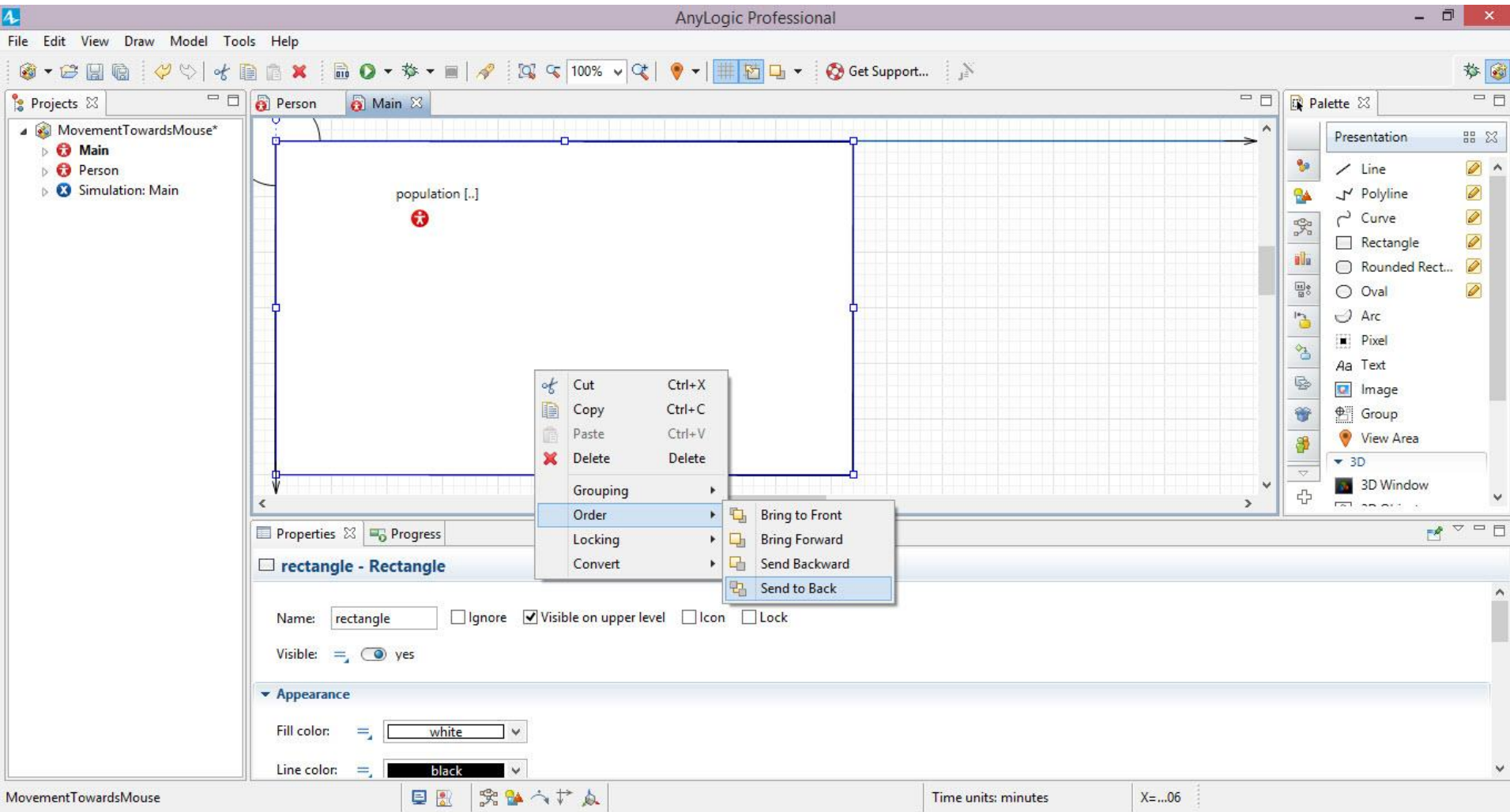
In “Main” Add a Rectangle (Origin at Same Place as Pop Origin)



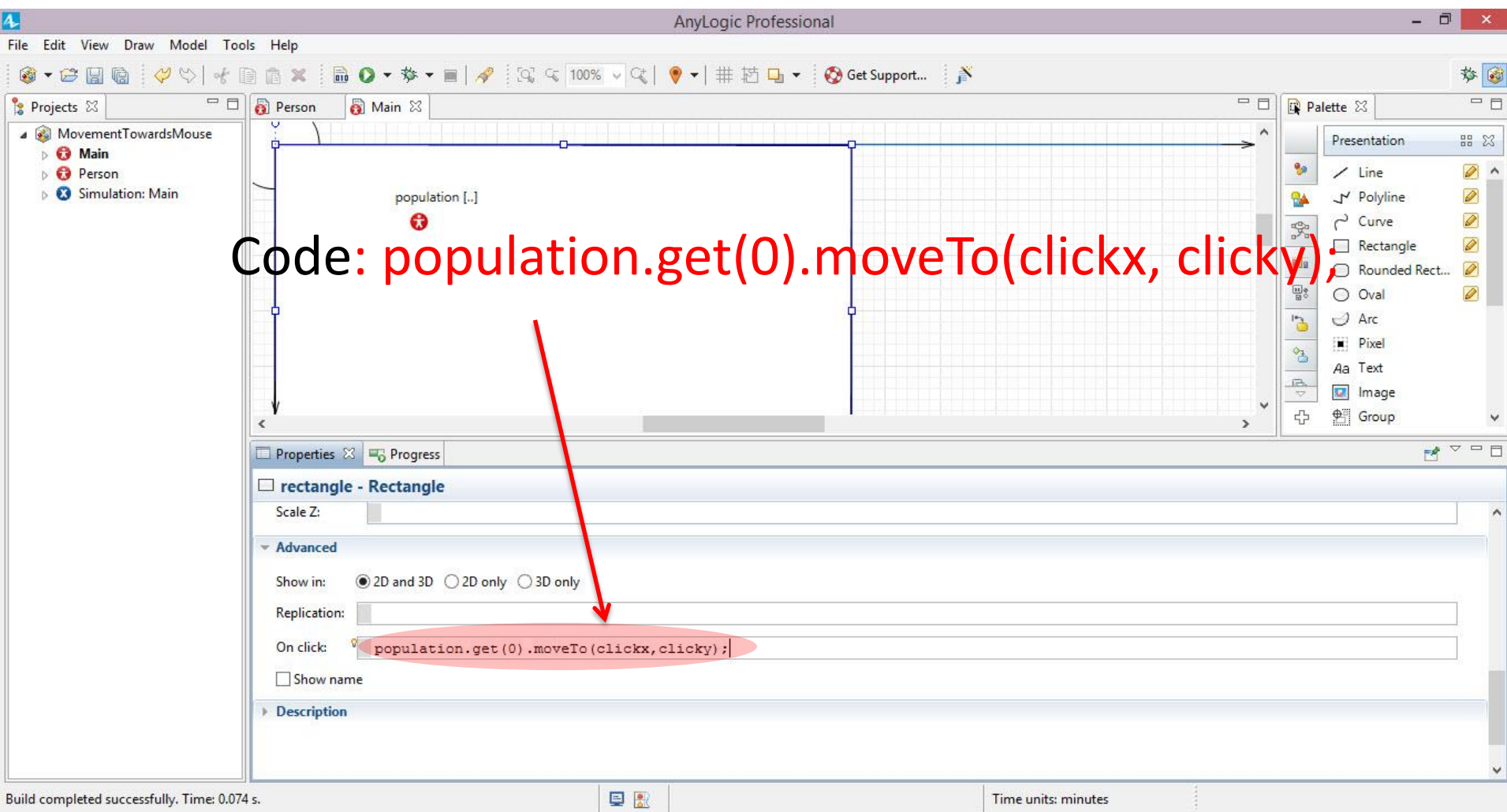
Enlarge Rectangle



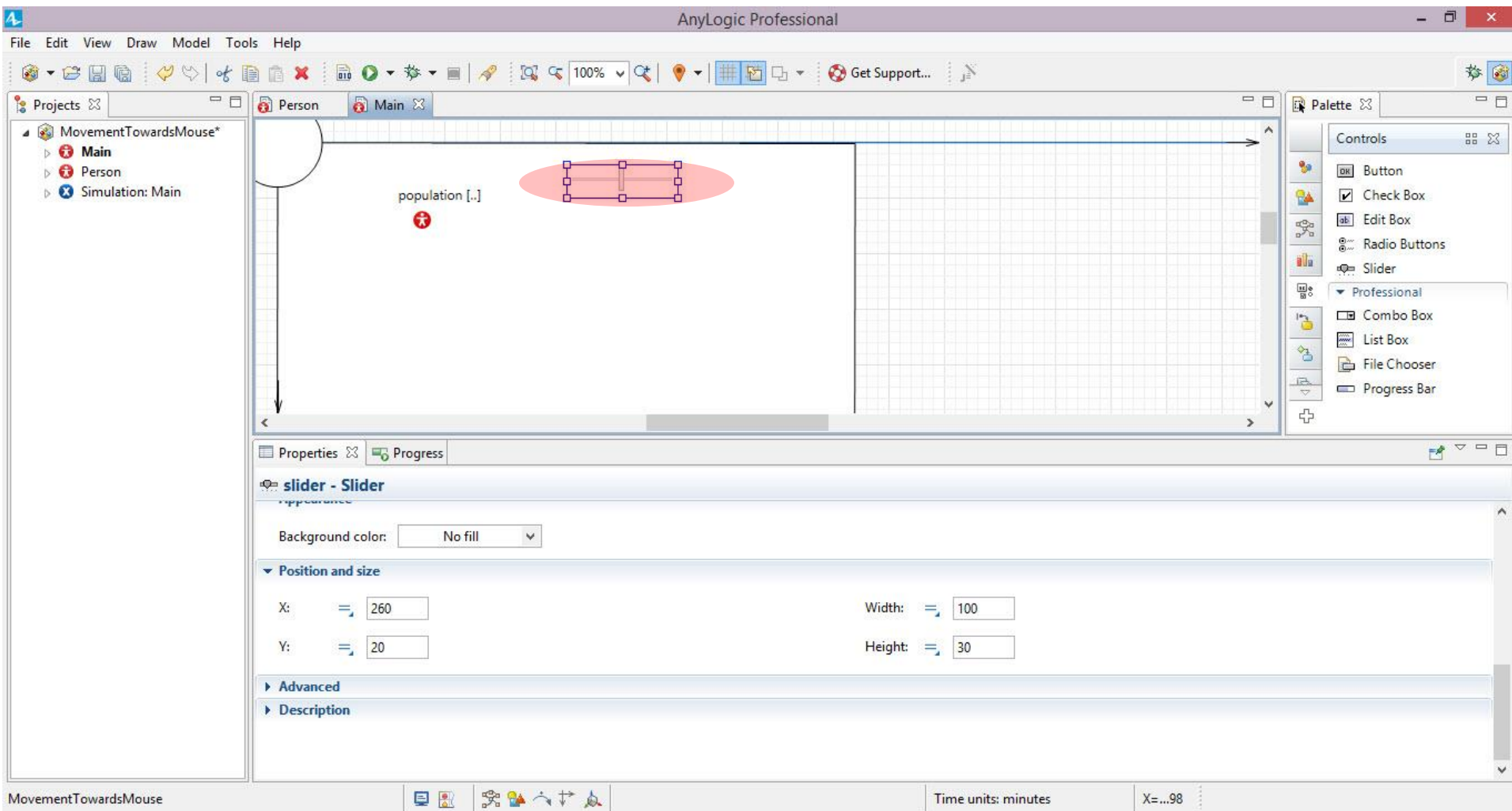
Set Rectangle's Ordering to Back



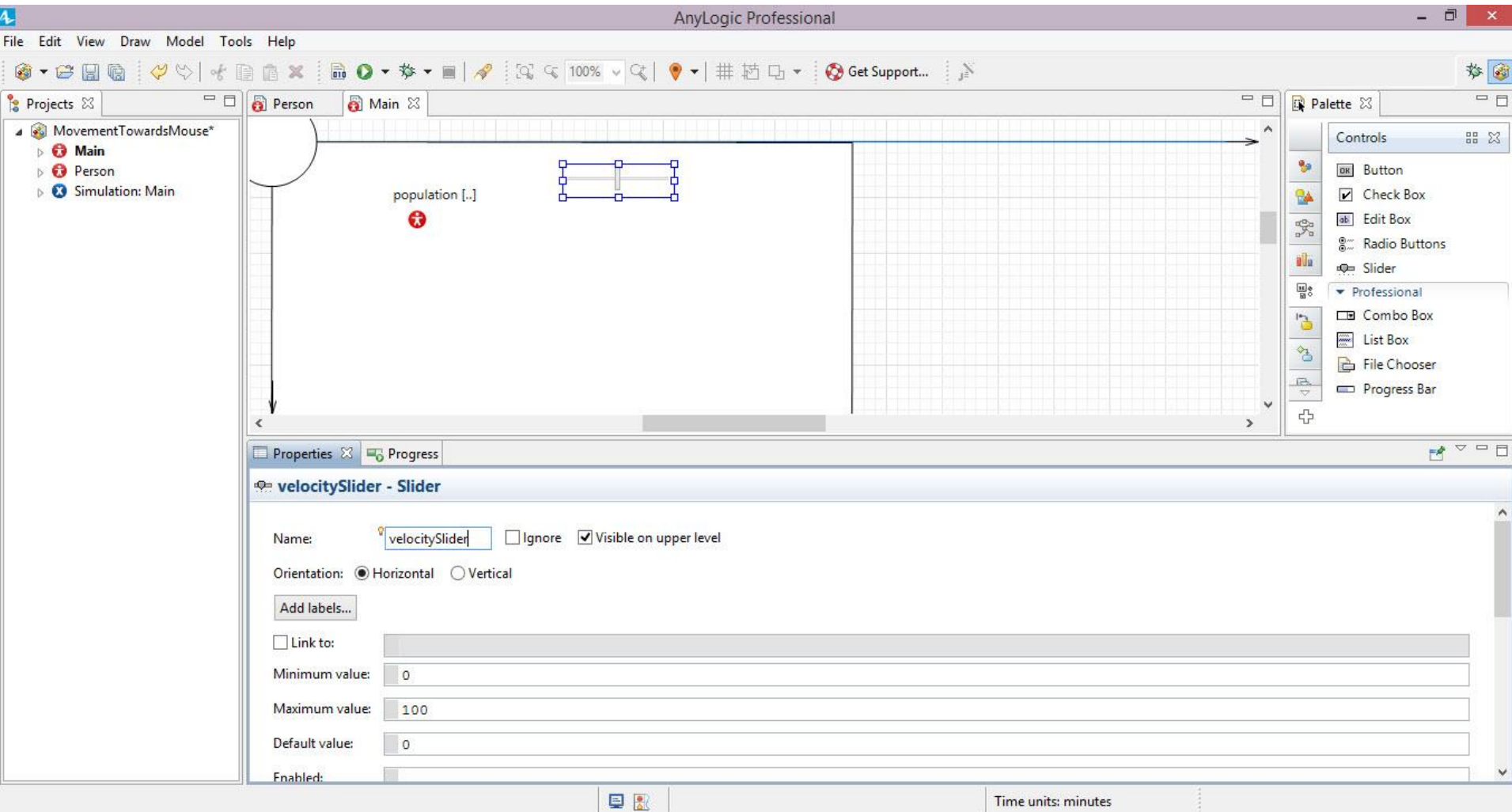
Insert Code



Drag a Slider from the “Controls” Area

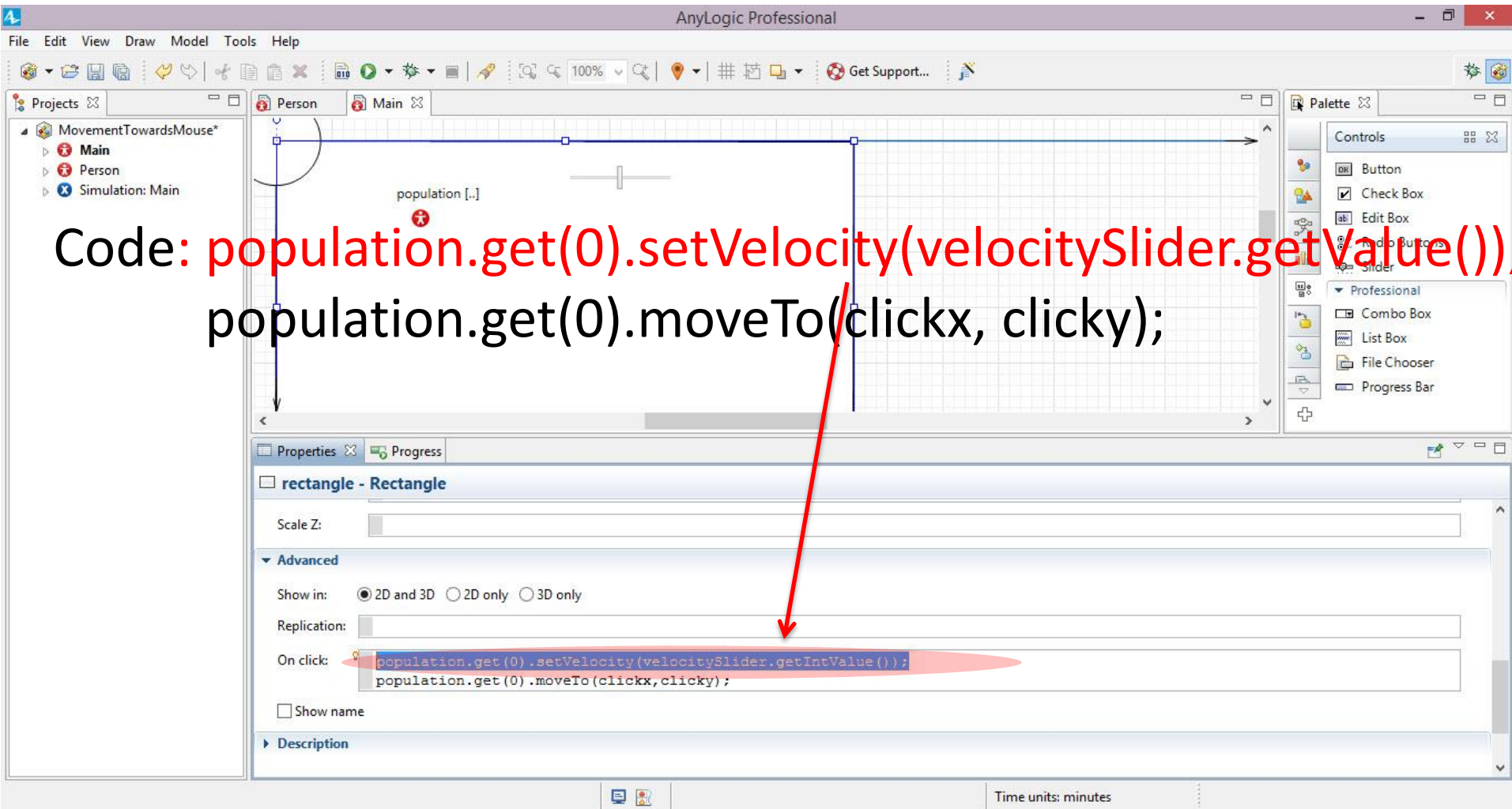


Name it “velocitySlider” (in “Properties”, under “General” tab)



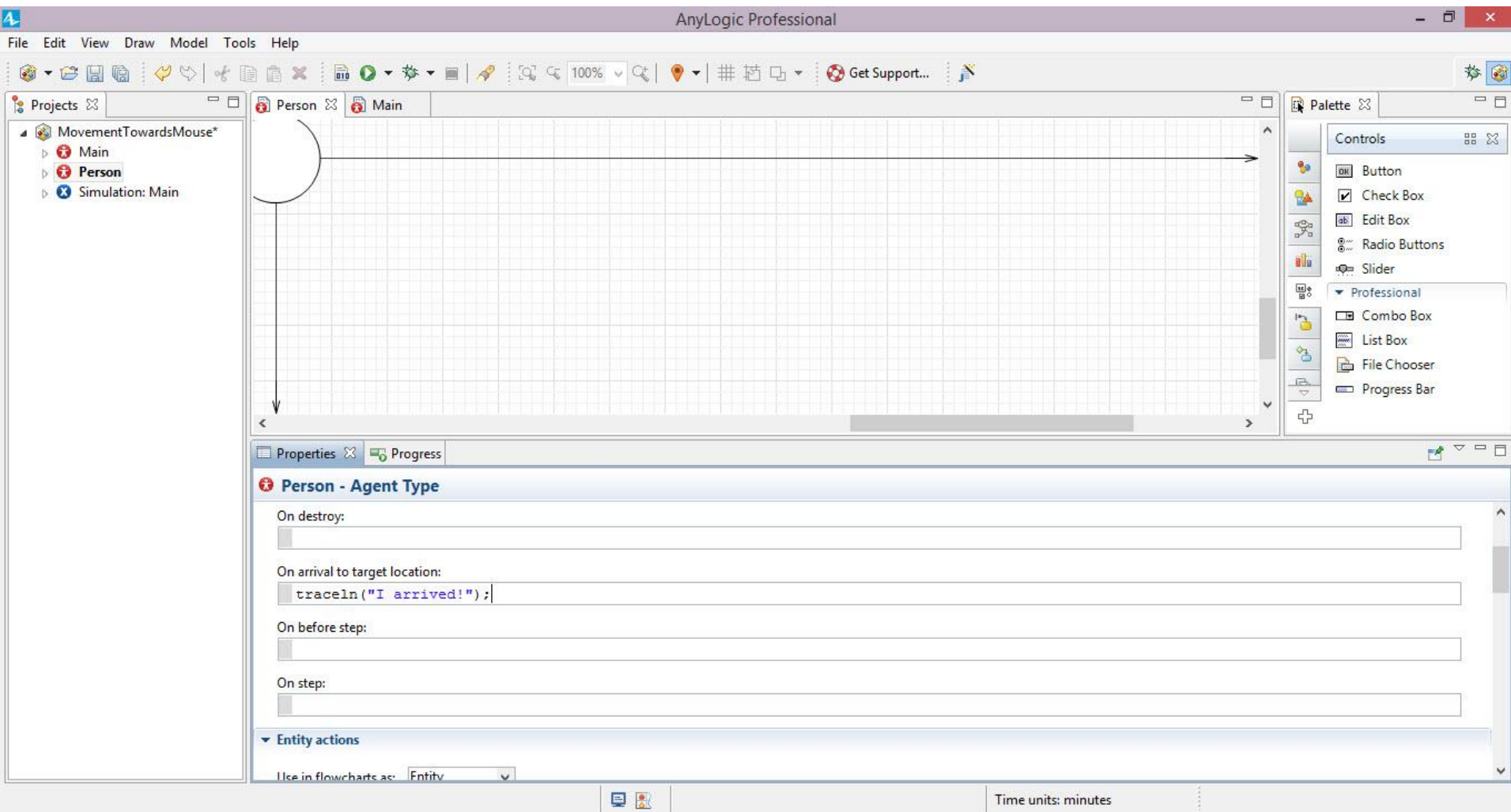
Add a New Line of Code to “On Click” Handler of Rectangle

Code: `population.get(0).setVelocity(velocitySlider.getValue());`
`population.get(0).moveTo(clickx, clicky);`



On “Person” “Agent” Properties

Add code to indicate Arrival



Add Random Walks

AnyLogic Professional

File Edit View Draw Model Tools Help

Projects

- MovementTowardsMouse*
- Main
- Person
- Simulation: Main

Person Main

mobilityStatechart

inMotion

Statechart

- Statechart Entry Point
- State
- Transition
- Initial State Pointer
- Branch
- History State
- Final State

Properties Progress

mobilityStatechart - Statechart Entry Point

Name: ☒ Show name ☐ Ignore

Visible: ☒ yes

Action:

Description

MovementTowardsMouse

Time units: minutes X=...30

The screenshot displays the AnyLogic Professional interface. The main workspace shows a statechart diagram with a yellow state labeled 'inMotion' and a statechart entry point labeled 'mobilityStatechart'. The 'Properties' panel at the bottom shows the configuration for 'mobilityStatechart - Statechart Entry Point', including its name, visibility, and an action 'this.setVelocity(100);'. The 'Palette' on the right lists various statechart elements like Statechart Entry Point, State, Transition, etc. The bottom status bar indicates 'Time units: minutes' and 'X=...30'.

Random Walk Models

The screenshot displays the AnyLogic Professional interface for modeling a random walk. The main workspace shows a statechart with a state named **inMotion** (represented by a yellow rounded rectangle). A transition labeled **mobilityStatechart** leads into the **inMotion** state. The statechart is connected to a larger model structure on the left, which includes a **Person** object and a **Simulation: Main** component.

The **Properties** panel at the bottom shows the configuration for the **inMotion - State**:

- Name:** inMotion
- Fill color:** Default
- Entry action:** `this.moveTo(this.getX()+normal(50,0),this.getY()+normal(50,0));`
- Exit action:** (empty)

The **Palette** on the right lists the components used in the statechart: Statechart Entry Point, State, Transition, Initial State Pointer, Branch, History State, and Final State.

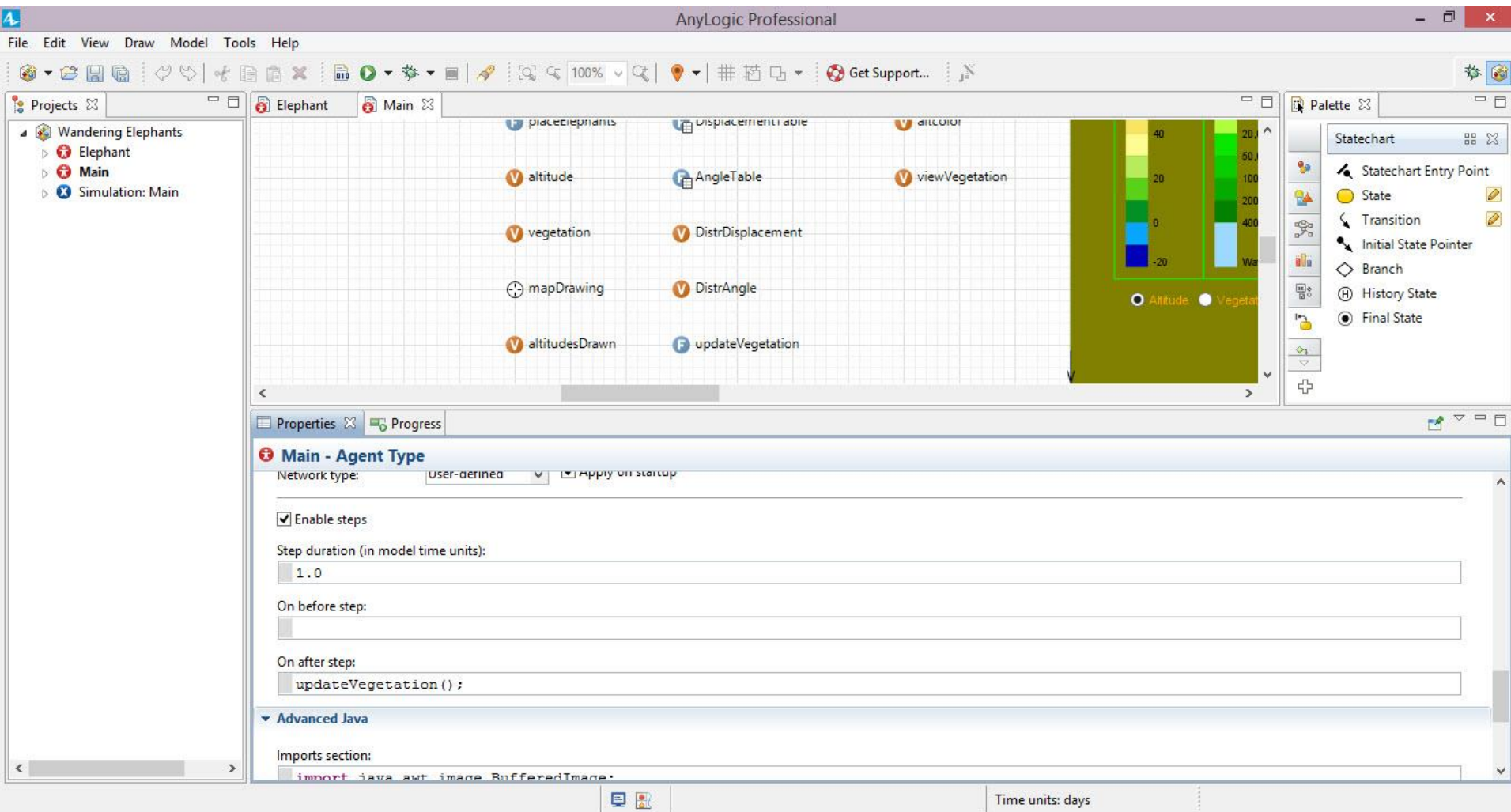
At the bottom of the interface, the status bar indicates **Time units: minutes**.

Continuous Space: Relevant Methods

(To call on Instances of *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



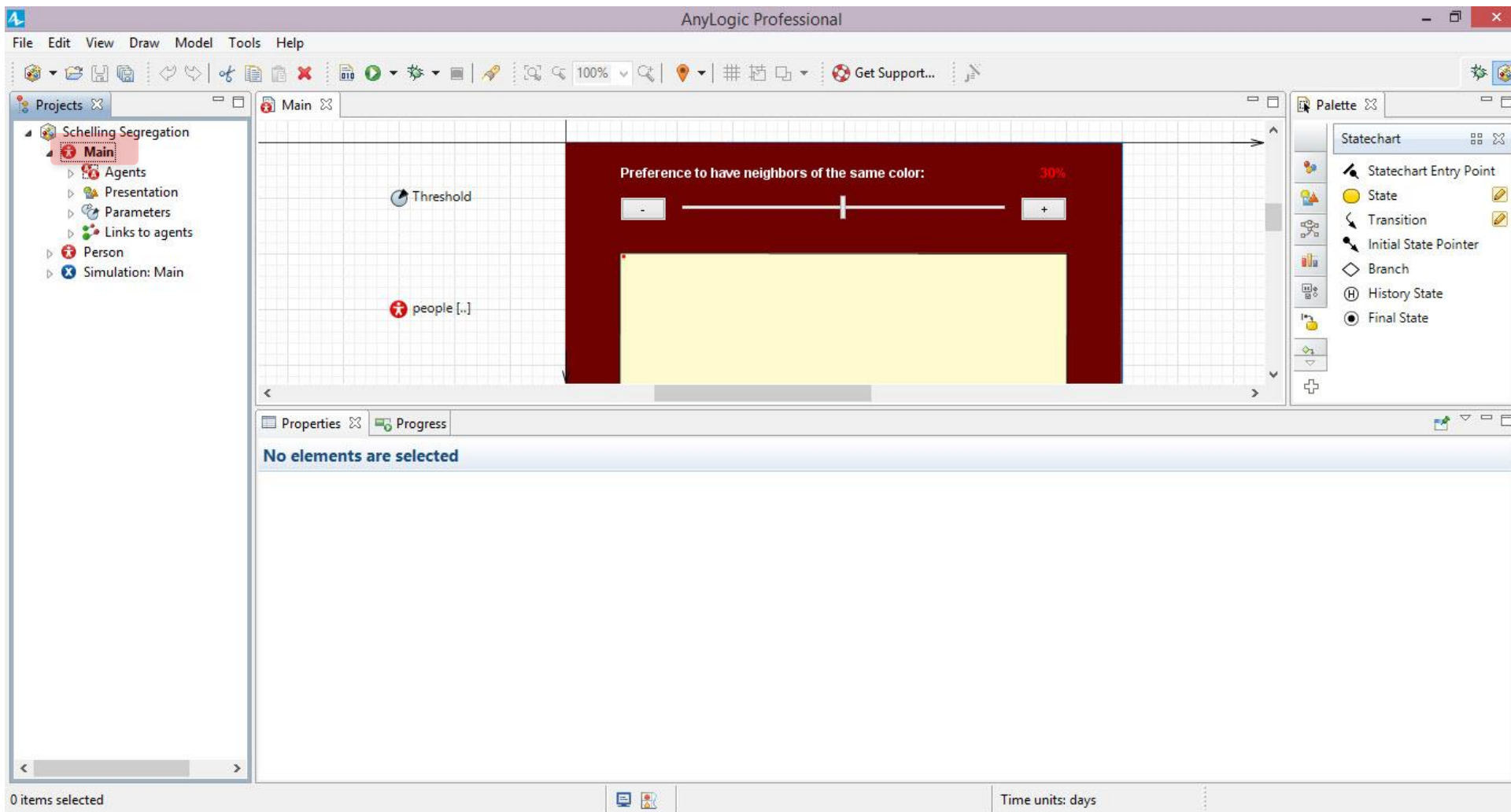


Hands on Model Use Ahead



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 Professional interface for configuring a discrete spatial environment. The main window shows a grid with a yellow rectangular area and a red border. The 'Main - Agent Type' configuration window is open, showing the following settings:

- Select the agents you want to place in the environment:** ☒ people
- Space type:** ☐ Continuous ☒ Discrete ☐ GIS
- Space dimensions:**
 - Columns:** 100
 - Rows:** 100
 - Width:** 400
 - Height:** 400
- Neighborhood type:** Moore
- Layout type:** Random ☒ Apply on startup
- Network type:** User-defined ☐ Apply on startup

Annotations in the image highlight the spatial dimensions:

- A red arrow points to the text "Spatial Width & Height".
- A blue arrow points to a box labeled "Width & Height in Discrete Cells".

Population Dependence on the Population

The screenshot displays the AnyLogic Professional software interface for a Schelling Segregation model. The main workspace shows a grid with a yellow rectangular area representing the environment, surrounded by a dark red border. A small red star icon labeled "people [...]" is positioned on the left side of the grid.

The left sidebar shows the project structure:

- Schelling Segregation
 - Main
 - Agents
 - Presentation
 - Parameters
 - Links to agents
 - Person
 - Simulation: Main

The right sidebar shows the Statechart palette with the following items:

- Statechart
 - Statechart Entry Po...
 - State
 - Transition
 - Initial State Pointer
 - Branch
 - History State

The bottom panel shows the Properties window for the "people - Person" agent:

people - Person

Name: ☒ Show name ☐ Ignore

Visible: ☒ yes

☐ Single agent ☒ Population of agents

Initial number of agents:

Color: =

Initial location

These settings are applied only if the "User-defined" layout type is set in the "Environment for other agents" properties of the upper level agent.

Row:

Column:

Statistics

The bottom status bar shows "Time units: days".

Slider Input Sets Parameter Value

The screenshot displays the AnyLogic Professional interface. The main workspace shows a model with a red rectangular area labeled "Preference to have neighbors of the same color:" and a yellow rectangular area below it. A green box labeled "Threshold" is connected to the red area. A green arrow points from the text "Threshold parameter" to the "Threshold" box. The Properties panel on the right shows the "slider - Slider" component. The "Link to:" field is set to "Threshold". The "Minimum value:" field is set to 0, and the "Maximum value:" field is set to 1. A red arrow points from the text "Sets Threshold Parameter Value" to the "Minimum value:" field. The "Action" and "Appearance" sections are also visible in the Properties panel.

AnyLogic Professional

File Edit View Draw Model Tools Help

Projects

- Schelling Segregation
 - Main
 - Agents
 - Presentation
 - Parameters
 - Links to agents
 - Person
 - Simulation: Main

Main

Threshold

Preference to have neighbors of the same color: 30%

people [...]

Properties

slider - Slider

Name: slider ☐ Ignore ☐ Visible on upper level

Orientation: ☒ Horizontal ☐ Vertical

Add labels...

☒ Link to: Threshold

Minimum value: 0

Maximum value: 1

Enabled: ☐

Action

Appearance

Time units: days

Sets Threshold Parameter Value

Person is Assigned a Randomly Picked Color

The screenshot displays the AnyLogic Professional interface. The main workspace shows a statechart for a 'Person' agent. A red rectangle represents the 'Person's Visual Representation'. A blue circle labeled 'color' is connected to a 'satisfied' state. A red arrow points from the text 'Person's Visual Representation' to the red rectangle. A blue arrow points from the text 'Color is set to either red or black with equal likelihood' to the 'color' parameter. The 'Properties' panel at the bottom shows the configuration for the 'color' parameter.

Person's Visual Representation

Color is set to either red or black with equal likelihood

color - Parameter

Name: ☒ Show name ☐ Ignore

Visible: ☒ yes

Type:

Default value:

☐ System dynamics array

Value editor

Label:

Control type:

Hide conditions:

Schelling Segregation

Time units: days

X=...16

Core Segregation (Movement) Logic

The screenshot displays the AnyLogic Professional interface for the 'Schelling Segregation' model. The main workspace shows a grid with two agents: 'color' (a blue circle) and 'satisfied' (an orange circle). The 'Person' agent type is selected in the 'Properties' pane, showing its logic. The logic is divided into 'On before step' and 'On step' sections. The 'On before step' section contains code to count neighbors with the same color and check if the fraction exceeds a threshold. The 'On step' section contains a condition to move the agent if not satisfied and a 30% chance of moving.

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

```
On before step:
//calc hbow many neighbors have same color as me
int nsame = 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 ) )
        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( 0.3 ) )
    jumpToRandomEmptyCell();
```

Experiment: Simulation Sets

Parameter Assumptions

The screenshot displays the AnyLogic Professional software interface. The main workspace shows a text-based description of the Schelling Segregation model. The text describes how Schelling placed pennies and dimes on a chess board to represent a city, with each square representing a house or lot. It details the rules for agent movement based on happiness and neighborhood composition. A button at the bottom of the text area reads "Run the model and switch to Main view".

The interface includes a menu bar (File, Edit, View, Draw, Model, Tools, Help), a toolbar with various icons, and a project tree on the left. The project tree shows the following structure:

- Projects
 - Schelling Segregation
 - Main
 - Agents
 - Presentation
 - Parameters
 - Links to agents
 - Person
 - Simulation: Main

The right side of the interface features a Palette with a Statechart section containing the following elements:

- Statechart
 - Statechart Entry Point
 - State
 - Transition
 - Initial State Pointer
 - Branch
 - History State
 - Final State

The bottom of the interface shows the Properties and Progress tabs. The Properties tab is active, displaying the "Simulation - Simulation Experiment" section. Under the "Parameters" sub-section, the "Threshold" is set to 0.7. A "Paste from clipboard" button is also visible.

The status bar at the bottom indicates "Time units: days".

Add a Parameter to Main

The screenshot displays the AnyLogic Professional interface. The main workspace shows a model diagram with a grid background. A red rectangular area on the right side of the grid contains a slider control labeled "Preference to have neighbors of the same color:" with a value of 30%. Below this slider is a yellow rectangular area. To the left of the red area, there are several elements: a "Threshold" icon, a "people [...]" icon, and a "likelihoodOfMovementIfDissatisfied" icon. The "Projects" panel on the left shows a tree structure with "Main" selected. The "Properties" panel at the bottom shows the properties for the "likelihoodOfMovementIfDissatisfied" parameter, including its name, visibility, type (double), and default value (0.3).

AnyLogic Professional

File Edit View Draw Model Tools Help

Projects

- Schelling Segregation*
 - Main
 - Agents
 - Presentation
 - Parameters
 - Links to agents
 - Person
 - Simulation: Main

Main Person Simulation

connections

Threshold

people [...]

likelihoodOfMovementIfDissatisfied

Preference to have neighbors of the same color: 30%

palette

General

- Agent
- Parameter
- Event
- Dynamic Event
- Variable
- Collection
- Function
- Table Function
- Custom Distribution
- Schedule
- Port
- Connector
- Link to agents

Properties Progress

likelihoodOfMovementIfDissatisfied - Parameter

Name: likelihoodOfMov ☒ Show name ☐ Ignore

Visible: ☒ yes

Type: double

Default value: 0.3

☐ System dynamics array

Value editor

Time units: days

Experiment: Add a Slider!

The screenshot displays the AnyLogic Professional software interface. The main workspace shows a grid with a red rectangular area titled "Schelling Segregation Model". This area contains text describing the model's history and rules. The text reads: "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 onstructive 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

The interface includes a menu bar (File, Edit, View, Draw, Model, Tools, Help), a toolbar with various icons, and a palette on the right side. The palette is open to the "Controls" tab, which lists various UI elements: Button, Check Box, Edit Box, Radio Buttons, Slider (highlighted in red), Professional, Combo Box, List Box, File Chooser, and Progress Bar. The bottom panel shows the "Simulation - Simulation Experiment" configuration area, which includes fields for Name (Simulation), Top-level agent (Main), Maximum available memory (64 Mb), and Parameters (Threshold: 0.7, likelihoodOfMovementIfDissatisfied: 0.3). The status bar at the bottom indicates "Time units: days".

Setting the Slider Properties

The screenshot displays the AnyLogic Professional software interface. The main workspace shows a grid-based simulation environment with a red overlay text box containing the following text:

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".

The interface includes a menu bar (File, Edit, View, Draw, Model, Tools, Help), a toolbar, and a palette on the right side with various controls like Button, Check Box, Edit Box, Radio Buttons, Slider, Combo Box, List Box, File Chooser, and Progress Bar.

The Properties panel at the bottom shows the configuration for the **sliderMovementChance - Slider**:

- Name: sliderMovementChance
- Orientation: ☒ Horizontal ☐ Vertical
- Link to: ☐ (empty)
- Minimum value: 0
- Maximum value: 1
- Default value: 0.3
- Enabled: true

The Action section is currently collapsed.

Setting Value for Parameter from Slider

The screenshot displays the AnyLogic Professional interface. The main workspace shows a simulation model with a grid background. A red rectangular area on the right side of the grid contains a slider control labeled "Preference to have neighbors of the same color:" with a value of 30%. Below the slider is a yellow rectangular area. The left sidebar shows a project tree for "Schelling Segregation*" with sub-items: Main, Agents, Presentation, Parameters, Links to agents, Person, and Simulation: Main. The bottom panel shows the "Properties" tab for the "Simulation - Simulation Experiment".

Simulation - Simulation Experiment

Name: ☐ Ignore

Top-level agent:

Maximum available memory: Mb

Parameters

Threshold:

likelihoodOfMovementIfDissatisfied:

Time units: days

Modify Person's Behavior to Depend on New Parameter

AnyLogic Professional

File Edit View Draw Model Tools Help

Projects

- Schelling Segregation*
- Main
 - Agents
 - Presentation
 - Parameters
 - Links to agents
- Person
 - Presentation
 - Parameters
 - Variables
 - Links to agents
- Simulation: Main

Main Person Simulation

Schelling Segregation/Person

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

Properties Progress

Person - Agent Type

```
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( get_Main().likelihoodOfMovementIfDissatisfied ) )
    jumpToRandomEmptyCell();
```

Entity actions

Use in flowcharts as: Entity

On enter flowchart block:

Time units: days

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 by routing the agent along a network of “polylines”