

Introduction to the Anylogic Interface & Supporting Concepts

Announcements

- Lecture recording links posted
- Tutorial time: Extended class hours on Tuesday or Thursday
 - Choice will depend on other classes following ours
 - Thursday is likely

The AnyLogic User Interface

The **Project View** (Common Configuration overview of projects & components)

Note: Double-Clicking on a Tab opens view as Full-Screen

Palette for adding items to canvas

Problem area
(indicates problem Building/running Model)

Properties area
(shows info on selected element in project or palette window)

The screenshot displays the AnyLogic Advanced software interface. The top window shows a project view for 'Person' with a tree structure of parameters, plain variables, dynamic variables, and statecharts. The central canvas displays a statechart diagram with states like 'TBSusceptible', 'LTDI', 'UnDiagnosedActiveTB', and 'DiagnosedActiveTB', connected by transitions. The bottom-left panel shows a 'Problems' list with several error messages, such as 'The constructor DataSet() is undefined'. The bottom-right panel shows the 'Properties' window for a selected transition, with fields for 'Name', 'Condition', and 'Action'.

The “Project View” – Hierarchically Shows the Project Components

The screenshot displays the AnyLogic Advanced interface. On the left, the 'Project View' shows a hierarchical tree of components. The top-level project is 'TBv1*', which is highlighted with a red oval. Below it is the 'Main' component, followed by 'Parameters', 'Functions', 'Environments', 'Embedded Objects', 'Presentation', and 'Person'. The 'Person' component has its own 'Parameters' list, including 'DaysPerTimeUnit: 365.25', 'Ethnicity: 1', 'MeanDaysToNaturallyClearInfection: 180.00', 'ReactivationRateForNormoGlycemicPeople: 1/100....', and 'Sex: true'. The main workspace shows a statechart diagram with several states: 'TBSusceptible', 'LTBI', 'UnDiagnosedActiveTB', and 'DiagnoseActiveTB'. Transitions between these states are labeled with conditions like 'WhetherInfected' and 'WhetherPrimaryProgression'. A red arrow points from the 'TBv1*' project name in the hierarchy to the statechart. Overlaid on the diagram is red text: 'The “*” means that the model has changed since the last time it was saved. You should consider saving the model when you see this!'. At the bottom, the 'Person - Active Object Class' properties window is visible, showing 'Name: Person' and 'Agent' checked.

The “*” means that the model has changed since the last time it was saved. You should consider saving the model when you see this!



Hands on Model Use Ahead

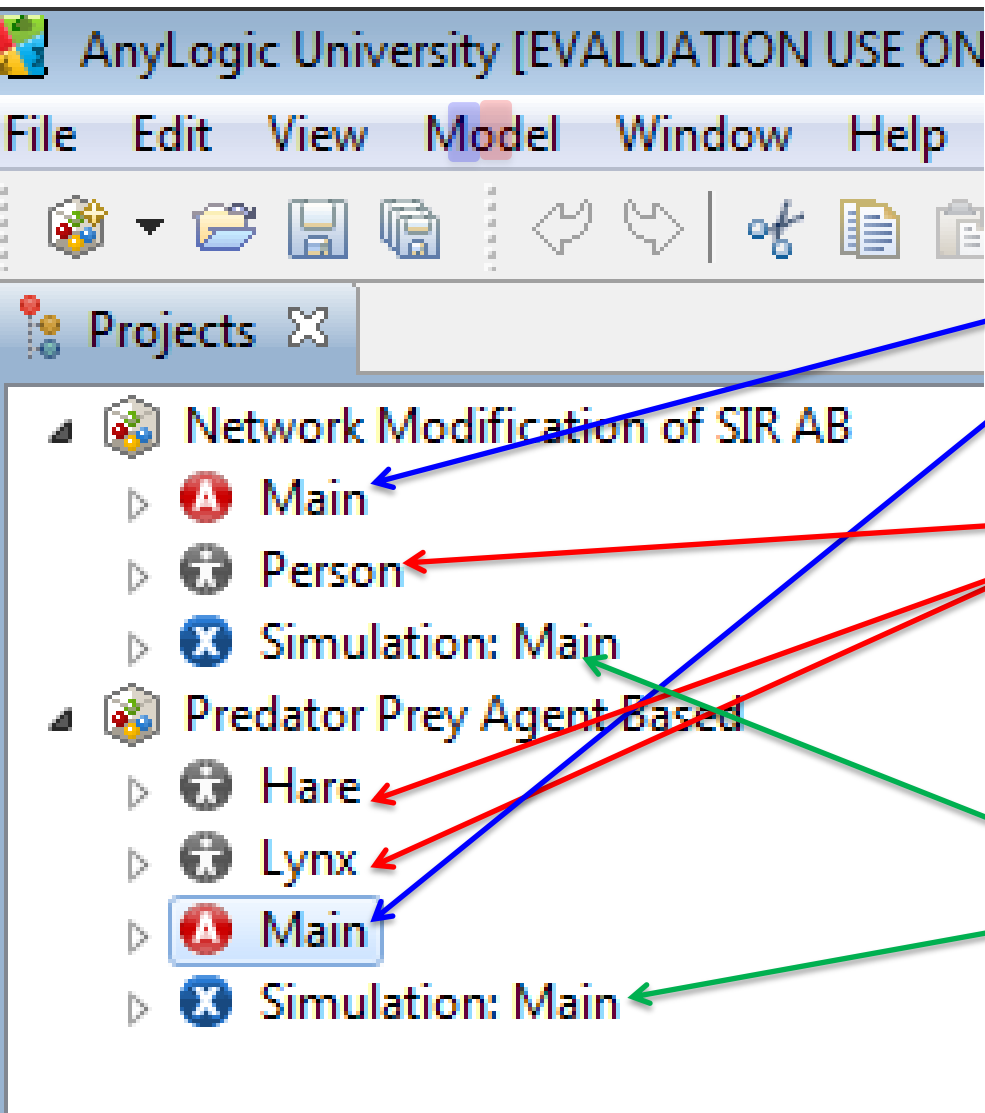


Load Sample Model:

Predator-Prey Agent Based

(Via “Sample Models” under “Help” Menu)

Example “Classes”

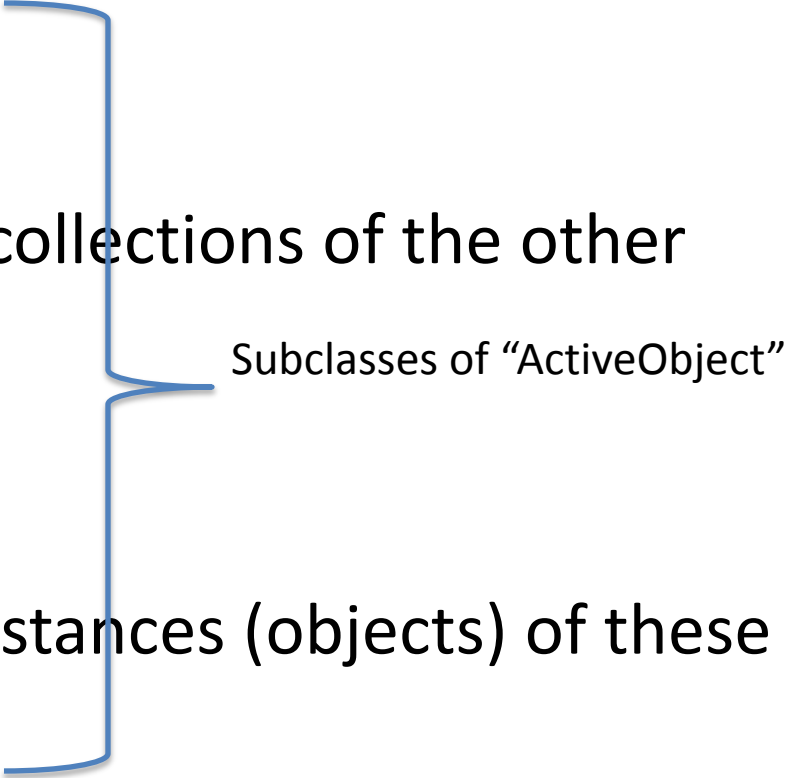


“Main” classes
(Define the “Stage”)

“Agent” classes
(Define the *actors*)

“Experiment” classes
(Define the *scenarios*)

Key Customized “Classes”

- The structure of the model is composed of certain key user-customized “classes”
 - “Main” class
 - Normally just one instance
 - This will generally contain collections of the other classes
 - “Agent” classes
 - Your agent classes
 - There are typically many instances (objects) of these classes at runtime
 - “Experiment” classes
 - These describe assumptions to use when running the model
- 
- Subclasses of “ActiveObject”

Double Click on "Main" Class Name to View this Class (Should Appear on Top Tab)

The screenshot displays the AnyLogic University software interface. The main window shows a simulation titled "Predator Prey Agent Based M...". The simulation area is divided into two main sections: a control panel and a graph.

Control Panel: Titled "Change parameters on-the-fly", it contains four sliders, each with a numerical value and a slider handle:

- Hare Births per Year: 123
- Hare Babies per Birth: 123
- Lynx Births per Year: 123
- Lynx Babies per Birth: 123

Graph: A line graph showing the population of Lynx (red line) and Hares (green line) over time (0 to 10). The y-axis ranges from 0 to 1.0. The Lynx population starts at approximately 0.7, peaks at 1.0 around time 7, and then declines. The Hare population starts at approximately 0.7, dips to a minimum of about 0.2 around time 6, and then rises back towards 1.0.

Project Tree (Left): Shows a hierarchy of projects. The "Main" class under the "Predator Prey Agent Based M..." project is highlighted with a red circle and a red arrow pointing to it. The text "Double Click Here!" is overlaid in red on the project tree.

Simulation Area (Center): The "Main" class name is highlighted with a white circle. A red arrow points from the "Main" class in the project tree to this circle. The text "Double Click Here!" is overlaid in red on the simulation area.

Palette (Right): Shows various simulation elements like Parameter, Event, Dynamic Event, etc.

“Main” Class

- Defines the environment where agents interact
- Defines interface & cross-model mechanisms
- The Main object normally contains one or more populations of “replicated” agents
 - Each population consists of agents of a certain class (or a subclass therefore), e.g.
 - “Hares”
 - “Lynxes”
 - The agent classes are defined separately from the Main class

Agent Populations in the Main Class

- Through the “Replication” property, the number of these agents can be set
- The “Environment” property can be used to associated the agents with some surrounding context (e.g. Network, embedding in some continuous space, with a neighborhood)
- Statistics can be computed on these agents
- Within the Main class, you can create representations of subpopulations by dragging from an Agent class into the Main class area

Elements of a “Main” Class

These “parameters” specify static model-wide characteristics

The screenshot shows the AnyLogic University interface. On the left, the 'Main' class editor displays several sections: 'Parameters' (a list of model-wide characteristics like HaresInitial, LynxInitial, etc.), 'Variables' (HaresInCell, auxGoodCells), 'Functions' (NotOverpopulatedCellAround, RandomCellAround, XGlobal, YGlobal), and 'Embedded Objects' (lynx [..], hares [..]). A purple arrow points to the Parameters list, and a blue double-headed arrow connects the lynx and hares objects. On the right, the simulation interface is titled 'Predator Prey Agent Based Model'. It features a control panel with sliders for 'Hare Births per Year', 'Hare Babies per Birth', 'Lynx Births per Year', and 'Lynx Babies per Birth', all set to 123. A red arrow points to this control panel. Below the sliders is a black simulation window. At the bottom, a line graph shows the population of Lynx (red line) and Hares (green line) over 10 time steps. A red arrow points to the graph. A green arrow points from the Functions list to the graph.

Visual input elements used during simulation (param. setting)

Visual output elements used during simulation

These represent the agent populations

These “functions” Calculate things or can change model behavior

Agent Class Defines the Characteristics & Behaviour of Agent Population Members

The screenshot displays the AnyLogic University interface. On the left, the 'Projects' pane shows a tree structure with 'Lynx' highlighted under 'Predator Prey Agent'. A red arrow points from this 'Lynx' entry to a statechart diagram in the main workspace. The statechart, titled 'statechart', is contained within a yellow box and shows a cycle of states: 'NoLuck' leads to 'Eat', which leads to 'Hunt', which leads back to 'NoLuck'. A transition labeled 'HaveBabies' is also shown. The right side of the interface features a 'Palette' with various modeling elements like 'Parameter', 'Event', and 'Dynamic Event'. The bottom status bar shows 'Selection'.

Double Click on “Lynx”!

A Critical Distinction:

Design (Specification) vs. Execution (Run) times

- The computational elements of Anylogic support both design & execution time presence & behaviour
 - Design time: Specifying the model
 - Execution time (“Runtime”): Simulating the model
- It is important to be clear on what behavior & information is associated with which times
- Generally speaking, design-time elements (e.g. in the palettes) are created to support certain runtime behaviors

A Familiar Analogy

- The distinction between model design time & model execution time is like the distinction between
 - Time of Recipe Design: Here, we're
 - Deciding what exact set of steps we'll be following
 - Picking our ingredients
 - Deciding our preparation techniques
 - Choosing/making our cooking utensils (e.g. a cookie cutter)
 - Time of Cooking: When we actually are following the recipe
 - A given element of the recipe may be enacted many times
 - One step may be repeated many times
 - One cookie cutter may make many particular cookies

Cooking Analogy to an Agent Class: A Cookie Cutter

- We only need one cookie cutter to bake many cookies
- By carefully designing the cookie cutter, we can shape the character of many particular cookies
- By describing an Agent class at model design time, we are defining the cookie cutter we want to use

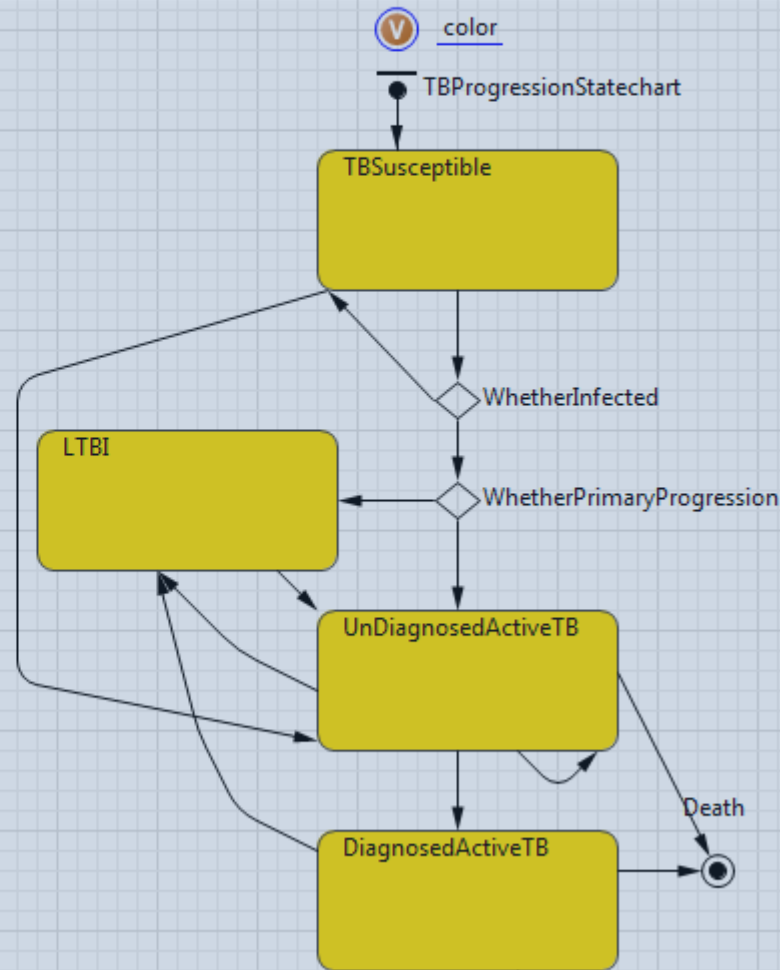
This defines the visual elements to be used for this object when it is displayed at runtime.

Common Agent-Class Elements

These introduce “methods” (“functions”) that include some Java code

- CirclePerimeterColorFromState
- CirclePerimeterWidthFromState
- ReactivationRateCoefficientForCKDStage
- ReactivationRateForCKDStage
- getDegree

- Sex
- Ethnicity
- DaysPerTimeUnit
- MeanDaysToNaturallyClearInfection



These “parameters” specify static agent characteristics

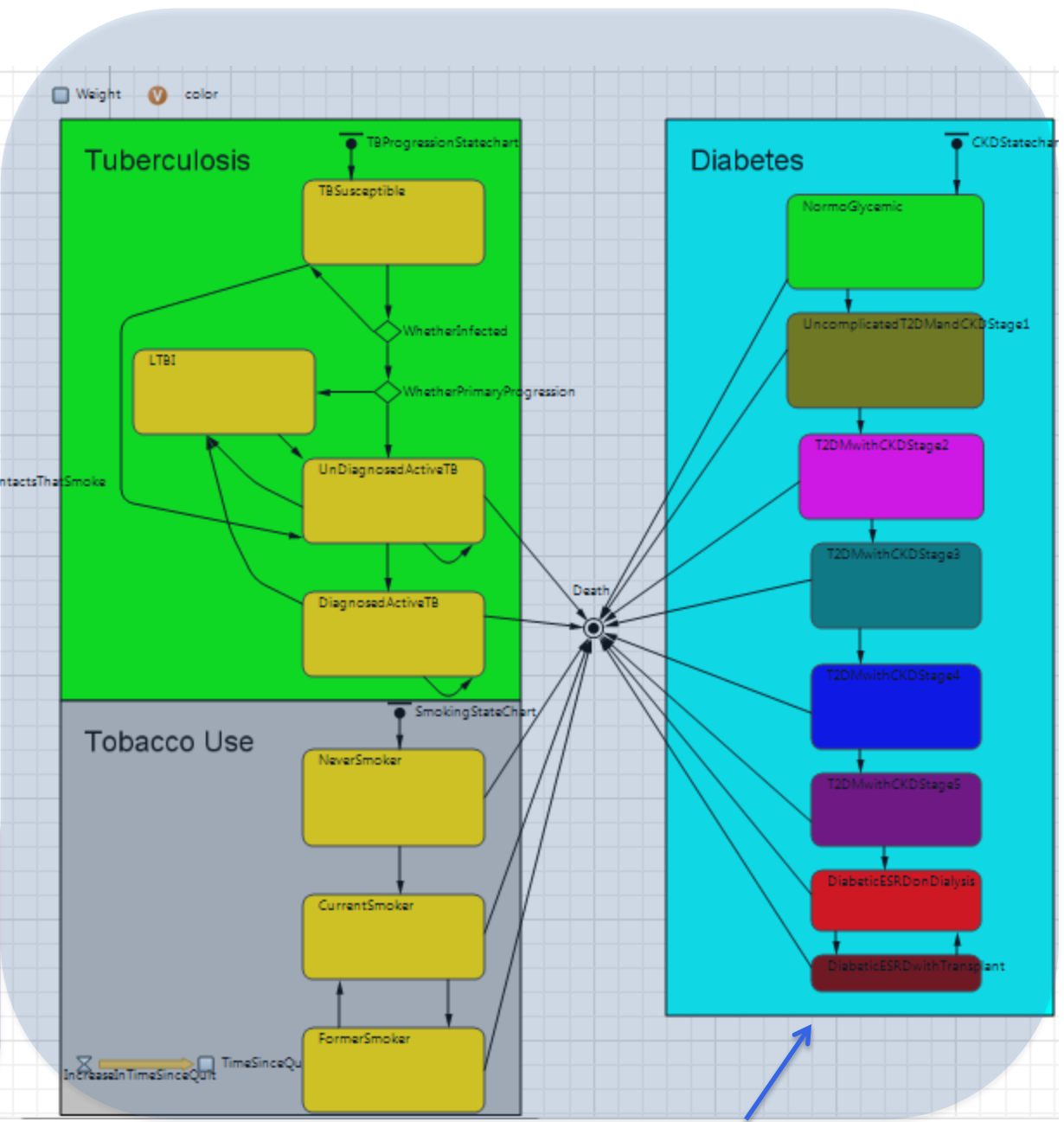
These describe the agent state & behaviour – the mechanisms that will govern agent dynamics

This defines the visual elements to be used for this object when it is displayed at runtime.

These introduce "methods" ("functions") That include some Java code for custom behaviours

- CirclePerimeterColorFromState
- CirclePerimeterWidthFromState
- SmokingInitiationHazardCoefficientAsAFunctionOfFractionOfContactsThatSmoke
- CountSmokingContacts
- CountContacts
- FractionOfContactsThatSmoke
- SmokingInitiationHazard
- ReactivationRateCoefficientForSmokingStatus
- ReactivationRateCoefficientForCKDStage
- ReactivationRateForSmokingStatusAndCKDStage
- IsCurrentSmoker
- AgeCoefficientForSmokingInitiation
- getDegree

- Sex
- Ethnicity
- MeanDaysToNaturallyClearInfection
- ReactivationRateForNormoGlycemicPeople
- SmokingInitiationHazardLogisticSteepnessCoefficient
- SmokingInitiationHazardLogisticValueWhenNoContactsSmoke
- SmokingInitiationHazardLogisticValueWhenAllContactsSmoke
- ReactivationRateHazardForNeverSmoker
- ReactivationRateHazardForCurrentSmoker
- RapidityOfDecreaseInReactivationRateWithTimeSinceQuit
- SmokingInitiationHazardLogisticMidpoint
- RapidityOfDecreaseInChanceOfRelapseWithTimeSinceQuit
- DaysPerTimeUnit



These "parameters" give static characteristics of the agent

These describe the "behaviours" – the mechanisms that will govern agent dynamics

Experiment Classes

- Experiment classes allow you to define & run scenarios in which global parameters (i.e. parameters defined in *Main*) may hold either default or alternative values
- Experiment classes are also used to set
 - The time horizon for a simulation
 - Memory limits (important for large models)
 - Details of simulation run
 - Details on random number generation
 - Virtual machine arguments
- “Properties” allow one to set the values for each parameter
- Right click on these & choose “Run” to run such a scenario

Setting Memory & Virtual Machine Arguments

The screenshot displays the AnyLogic Advanced [EDUCATIONAL USE ONLY] interface. The main window shows the 'Simulation - Simulation Experiment' properties. The 'Advanced' tab is selected, showing the following settings:

- Application options (will not be applied when model runs as applet)
- Maximum Available Memory: 64 Mb
- Java Machine Arguments: (empty field)
- Command-line Arguments: (empty field)
- Load root object from snapshot: (empty field)

The 'Imports section:' and 'Additional Class Code:' fields are empty. The 'Initial Experiment Setup:' field is empty. The 'Before Each Experiment Run:' field contains the following code:

```
Date dateOpen = toDate(2010,0,4,7,0, 0); // (new java.util.GregorianCalendar(2000, 0, 1,
traceln(dateOpen);
getEngine().setStartDate(dateOpen);
```

The left sidebar shows the project structure for 'Wandering Elephants' and 'ABMClinicModelV6'. The 'Properties' window is open, and the 'Advanced' tab is selected. The 'Maximum Available Memory' is set to 64 Mb. The 'Java Machine Arguments' and 'Command-line Arguments' fields are empty. The 'Load root object from snapshot' checkbox is unchecked. The 'Imports section:' and 'Additional Class Code:' fields are empty. The 'Initial Experiment Setup:' field is empty. The 'Before Each Experiment Run:' field contains the following code:

The Notion of a “Build”

- We prepare a fully specified model to run a simulation using a “build”
 - If all goes well, this translates project to executable Java
 - This may alert you to errors in the project
- A “Compiler” is a tool to convert from a program’s specification (e.g. state charts, Action diagrams, etc.) to a representation that can be executed
 - Normally a compiler is applied to each of several components of a program (e.g. classes)
 - AnyLogic’s “build” process applies a compiler to the components of the AnyLogic model

Cooking Analogy to “Build”ing: Obtaining & Preparing the Ingredients

- Before we can actually realize the recipe, we need to go collect & prepare all ingredients
- We’re not yet cooking, but what we are doing makes the cooking possible
- The “cooking” here is running the modle

A Bit on “Java” ...

- “Java” is a popular cross-platform “object oriented” programming language introduced by Sun Microsystems
- Anylogic is written in Java and turns models into Java
- AnyLogic offers lots of ways to insert snippets (“hooks”) of Java code
- You will need these if you want to e.g.
 - Push AnyLogic outside the envelop of its typical support
 - e.g. Enabling a network with diverse Agent types
 - Exchange messages between Agents
 - Put into place particular initialization mechanisms
 - Collect custom statistics over the population

Stages of the Anylogic Build

Modification Possible



Modification Not Possible

Java Code

```
double initialPrevalenceOfInfection ) {
    if (initialPrevalenceOfInfection == this.initialPrevalenceOfInfection) {
        return;
    }
    this.initialPrevalenceOfInfection = initialPrevalenceOfInfection;
    onChange_initialPrevalenceOfInfection();
    onChange();
}

void onChange_initialPrevalenceOfInfection() {
    int index;
    index = 0;
    for ( Person object : Population ) {
        object.set_isInitiallyInfected((uniform() < initialPrevalenceOfInfection));
        index++;
    }
}
```

JVM
Byte
Code

Person.class

“Build” Buttons

(One just for this project, one for all projects)

The screenshot displays the AnyLogic University software interface. The main workspace shows a statechart with two states: 'Susceptible' and 'Infected'. The 'Person' class is selected in the Project palette, and its properties are visible in the Properties panel. The Properties panel shows the 'Agent' section with a 'statechart.receiveMessage(msg);' event. The Console panel is empty. The System Dynamics palette is visible on the right, and the Welcome screen is partially visible on the far right.

Build just this project (blue arrow pointing to the 'Build' button in the toolbar)

Build all projects (red arrow pointing to the 'Build All' button in the toolbar)

Build completed successfully. Time: 3.000 s.

Selection X=83, Y=181

Alternative: Building via Context Menu

The screenshot displays the AnyLogic Advanced [EDUCATIONAL USE ONLY] interface. The main workspace shows a state transition diagram with states: Susceptible, Infective, NonPregnant, and Pregnant. A context menu is open over the diagram, listing standard file operations such as New, Open..., Save, Save As..., Revert, Close, Close Others, Close All, Cut, Copy, Paste, Delete, Refresh, Build (F7), Export..., and Team. The 'Build' option is highlighted. The diagram includes variables like circleSize, appearanceTime, sex, ethnicity, pregnancyStatus, and various functions like FertilityRateAgeSexEthnicity, PerformBirth, and FinalizeDeath. The bottom panel shows the 'EclipseDebuggingExample - Model' configuration with fields for Name, Package, and File.

Builds Gone Bad: The “Problems View”

The screenshot displays the AnyLogic Advanced [EDUCATIONAL USE ONLY] interface. The main workspace shows a statechart diagram for a TB progression model. The diagram includes states: **TBSusceptible**, **LTBI**, **UnDiagnosedActiveTB**, and **DiagnosedActiveTB**. Transitions are labeled with events like **WhetherInfected** and **WhetherPrimaryProgression**. A **Death** state is also present. A red arrow points from the title to the **Problems View** window in the bottom-left corner.

The **Problems View** window shows the following error messages:

- ✗ The constructor DataSet() is undefined
- ✗ The constructor DataSet() is undefined
- ✗ The constructor DataSet() is undefined
- ✗ The constructor DataSet() is undefined
- ✗ Engine.log cannot be resolved
- ✗ mViewer cannot be resolved
- ✗ mViewer cannot be resolved

The **Person - Active Object Class** properties window is visible at the bottom, showing the **Additional class code** section with the following code:

```
public static final int MsgInfectiousTBContact=1;  
public static final int MsgForceInitialTBInfection=2;
```

Builds Gone Good: Model Execution

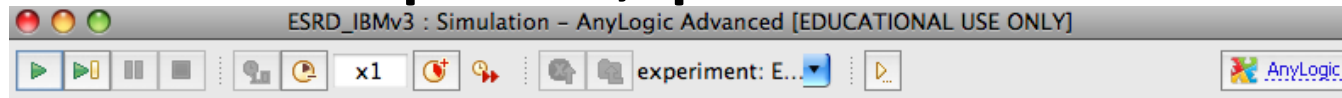
- The simulation is running
- Time is advancing in steps or as necessary to handle events
- Each agent class will typically have many particular agents in existence
 - Each agent will have a particular state
 - This population may fluctuate
- Variables will be changing value
- Presentation elements will be knit together into a dynamic presentation

Press this button to run an experiment (a simulation)

You can pull down the menu to choose which experiment to simulate

The screenshot displays the AnyLogic software interface. At the top, the menu bar includes 'AnyLogic', 'File', 'Edit', 'View', 'Model', 'Window', and 'Help'. The title bar reads 'AnyLogic Advanced [EDUCATIONAL USE ONLY]'. The main workspace shows a state transition diagram for 'TBProgressionStatechart' with states: 'TBSusceptible', 'LTBI', 'UnDiagnosedActiveTB', and 'DiagnosedActiveTB'. Transitions are labeled 'WhetherInfected' and 'WhetherPrimaryProgression'. A 'Death' state is also shown. A 'Recent Experiment' dropdown menu is open, listing various simulation models such as 'TBv1 / Simulation', 'ESRD_IBMv4 / Simulation', and 'SIR / Simulation'. The left sidebar contains a project tree for 'TBv1*' with sections for 'Main', 'Parameters', 'Functions', 'Environments', 'Embedded Objects', 'Presentation', and 'Person'. The bottom right shows the 'Person - Active Object Class' properties panel, with 'Agent' checked and 'Generic' unchecked. The bottom left shows a 'Problems' table with columns for 'Description' and 'Location'.

Initial Screen: Experiment Set up (Use to set speed, parameters via UI)

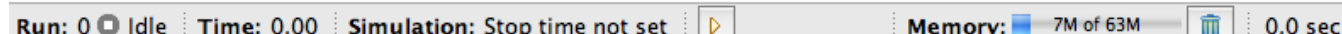


ESRD_IBMv3

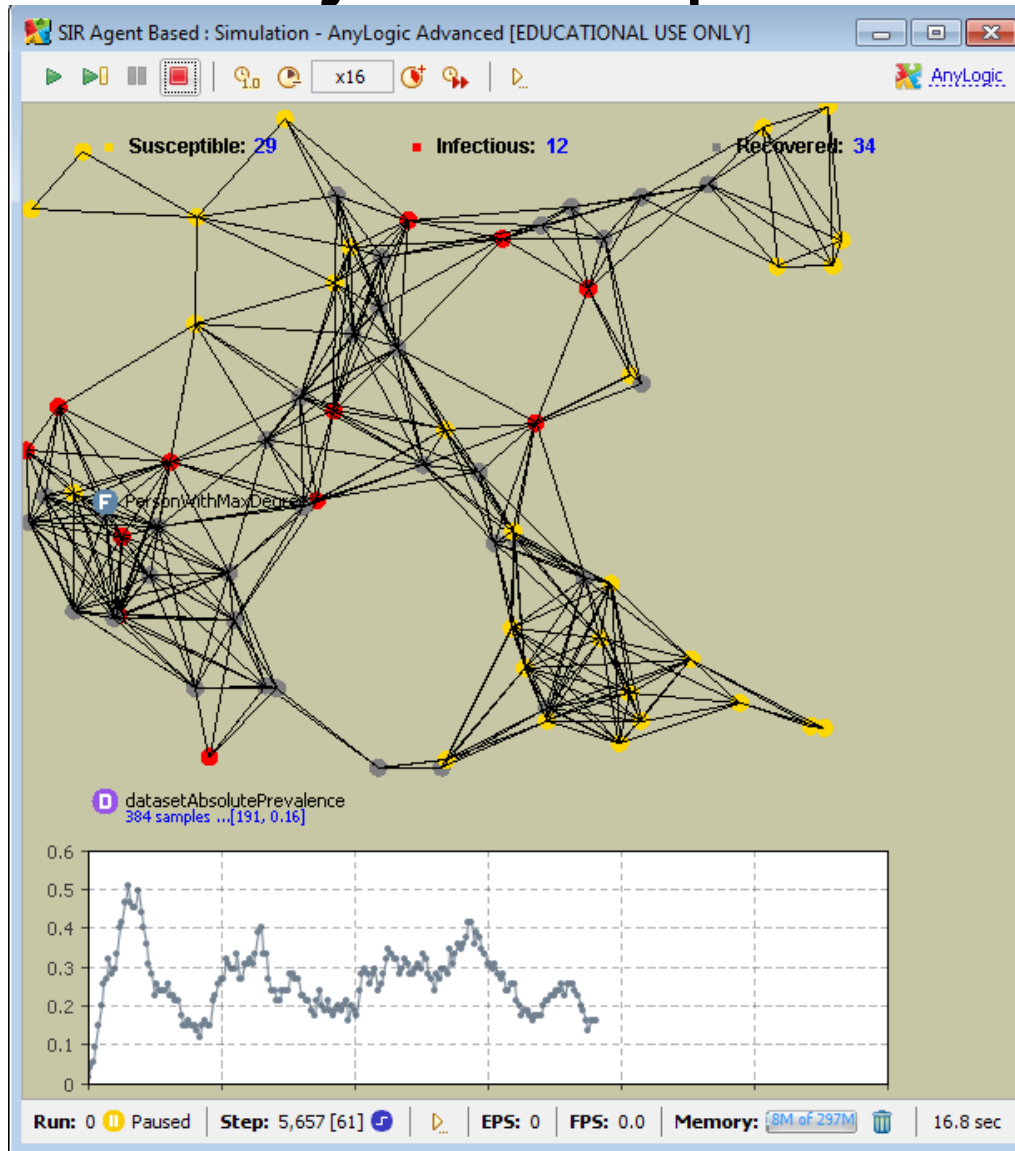
Experiment setup page

Run the model and switch to Main view

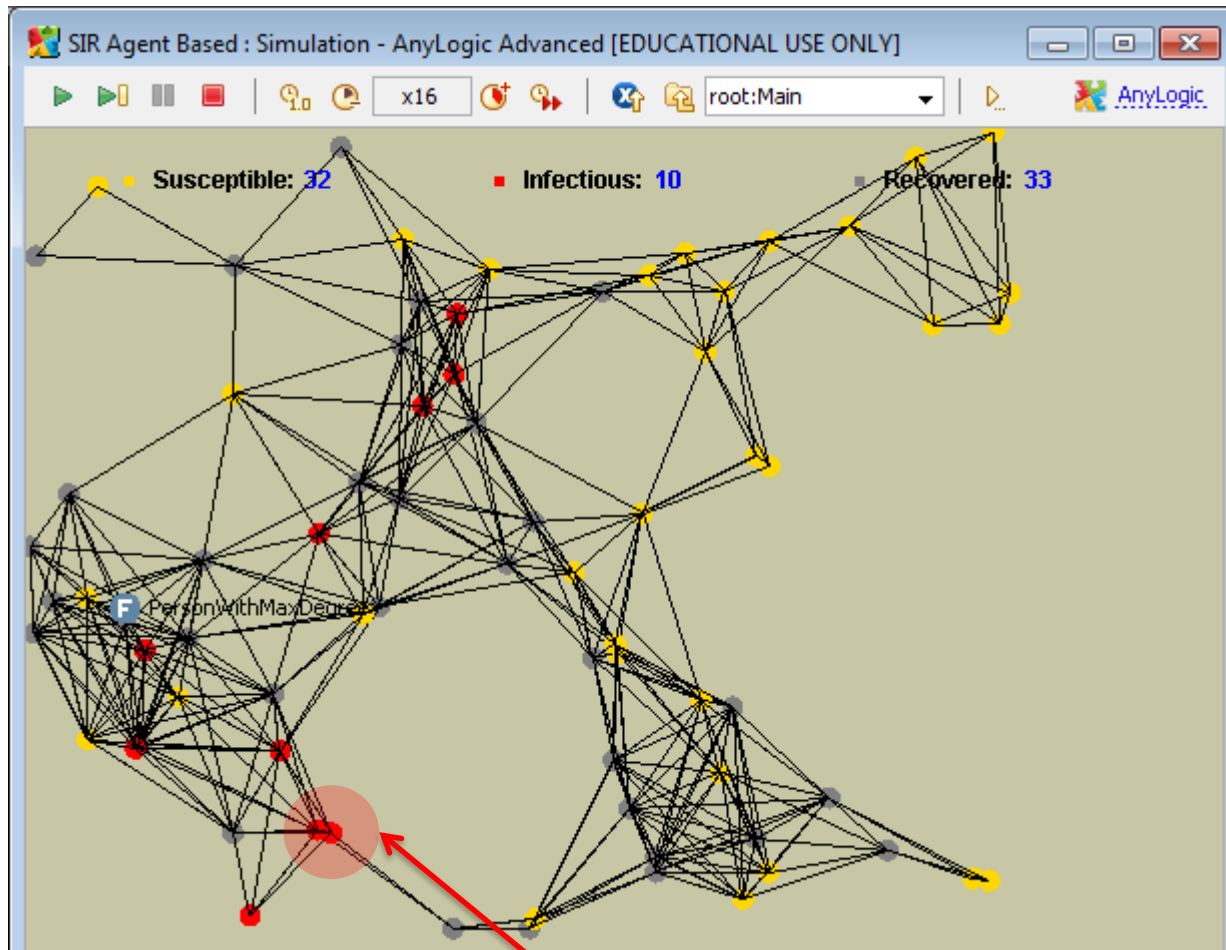
Press this button to switch to the model presentation display



Presentation of the Model “Main” Object in Operation

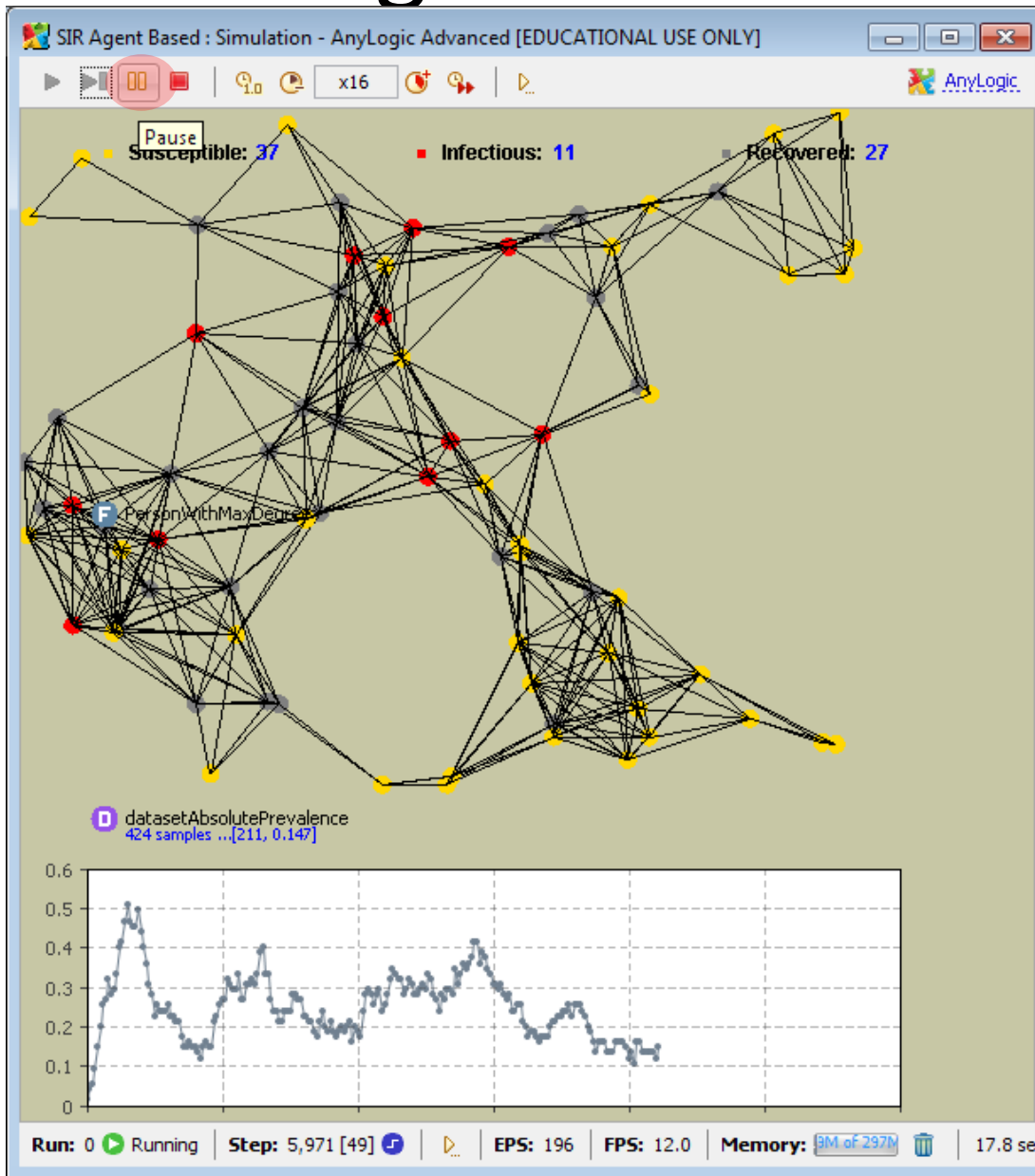


Network Embedding of Agents

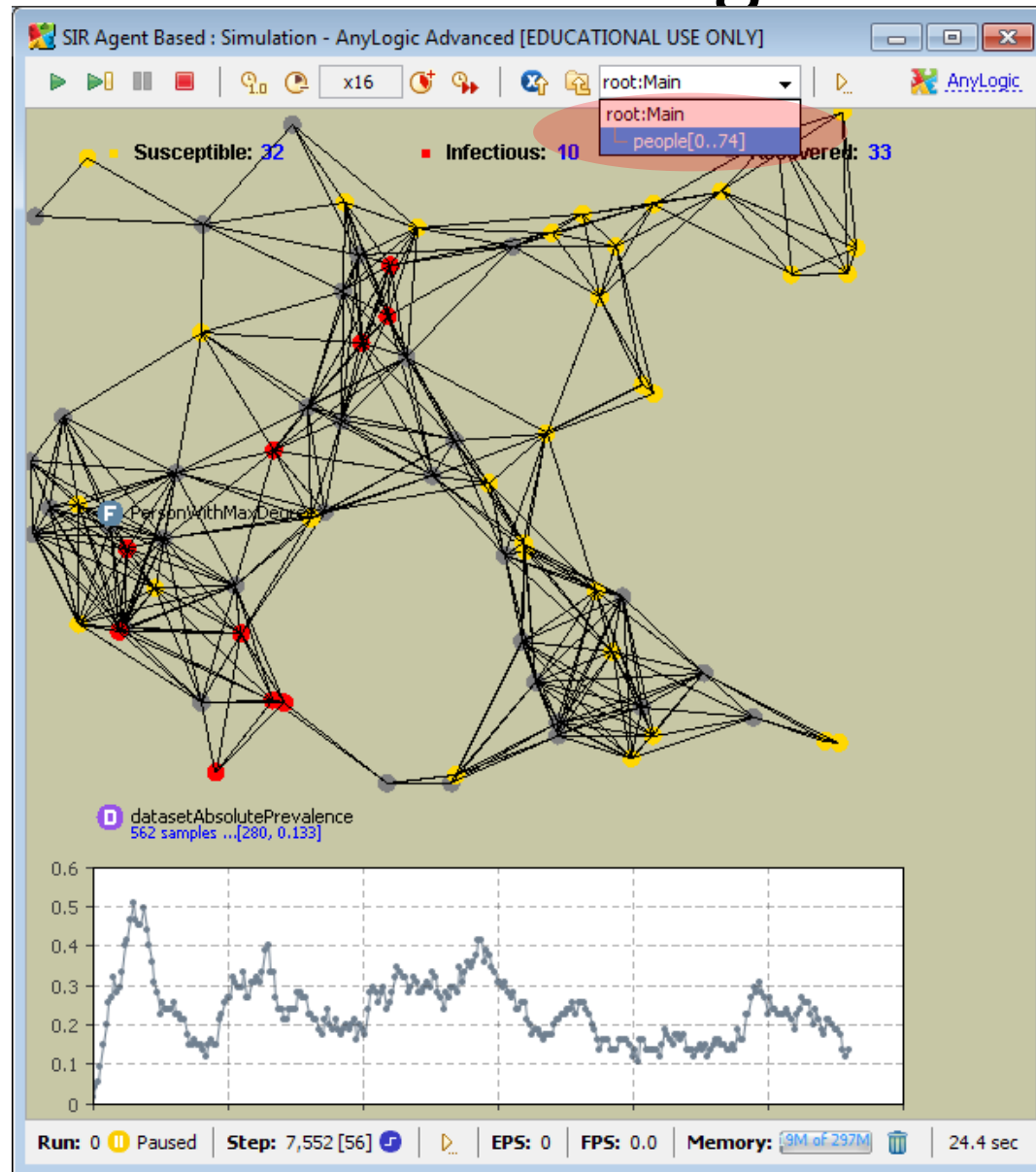


Dynamic color updates via agent logic

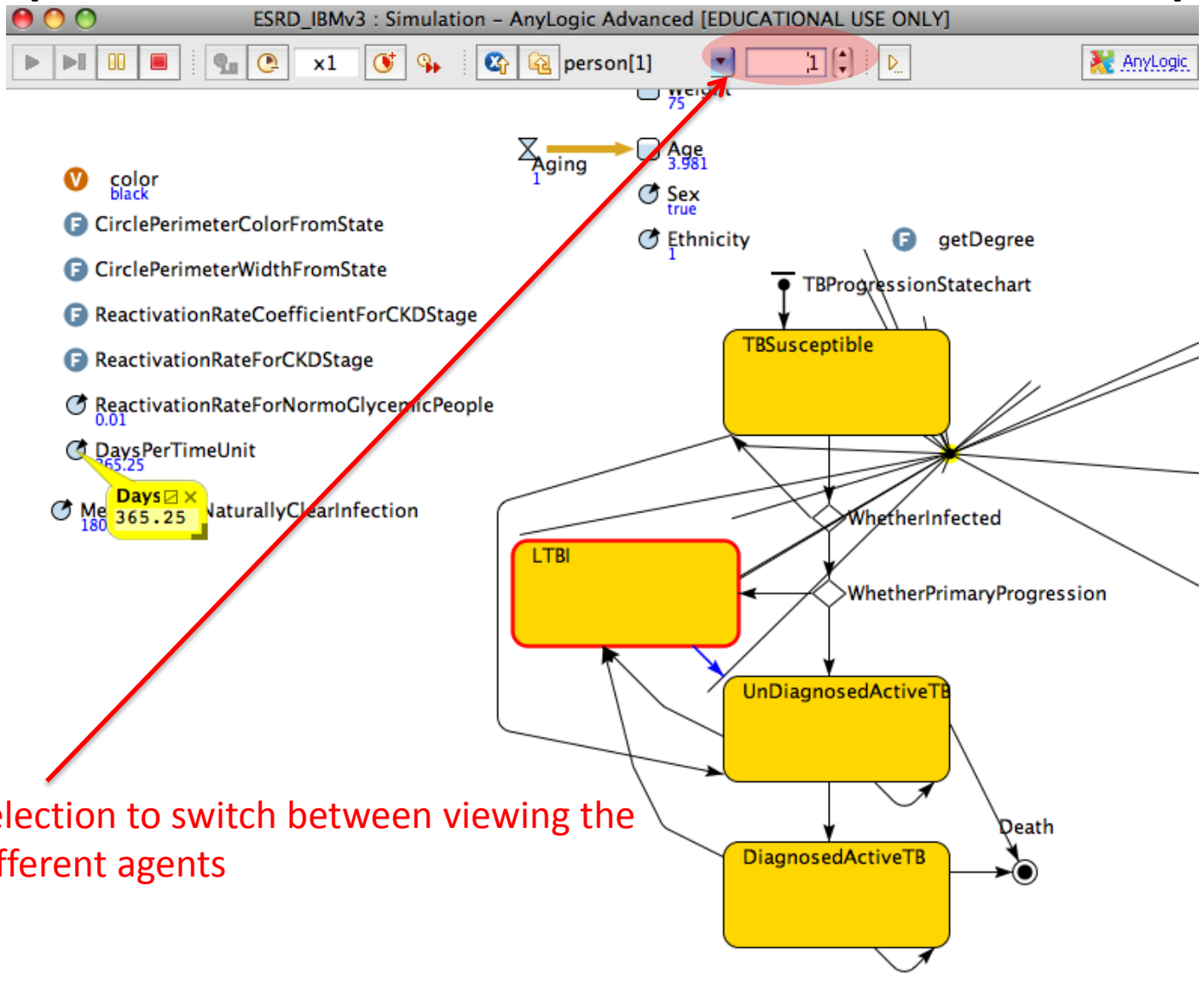
Pausing the Model



Drill Down from the Model to Particular Agents



Runtime View of Particular Agent (Drill Down from Previous View)



Use this selection to switch between viewing the State of different agents

Customizing the Model Running User Interface

The screenshot displays the AnyLogic Advanced simulation environment. The main window title is "ESRD_IBMv3 : Simulation - AnyLogic Advanced [EDUCATIONAL USE ONLY]". The interface includes a toolbar with various simulation controls, a parameter list on the left, and a state transition diagram in the center.

Parameter List (Left):

- color: black
- CirclePerimeterColorFromState
- CirclePerimeterWidthFromState
- ReactivationRateCoefficientForCKDStage
- ReactivationRateForCKDStage
- ReactivationRateForNormoGlycemicPeople: 0.01
- DaysPerTimeUnit: 365.25
- MeanDaysToNaturallyClearInfection: 180

State Transition Diagram (Center):

- TBSusceptible** (yellow rounded rectangle) is the starting state.
- Transitions from **TBSusceptible** lead to decision diamonds: **WhetherInfected** and **WhetherPrimaryProgression**.
- WhetherInfected** leads to **LTBI** (yellow rounded rectangle with a red border).
- WhetherPrimaryProgression** leads to **UnDiagnosedActiveTB** (yellow rounded rectangle).
- LTBI** transitions to **UnDiagnosedActiveTB**.
- UnDiagnosedActiveTB** transitions to **DiagnosedActiveTB** (yellow rounded rectangle).
- DiagnosedActiveTB** transitions back to **UnDiagnosedActiveTB** and also leads to a **Death** state (represented by a circle with a dot).
- A **TBProgressionStatechart** is connected to the **TBSusceptible** state.

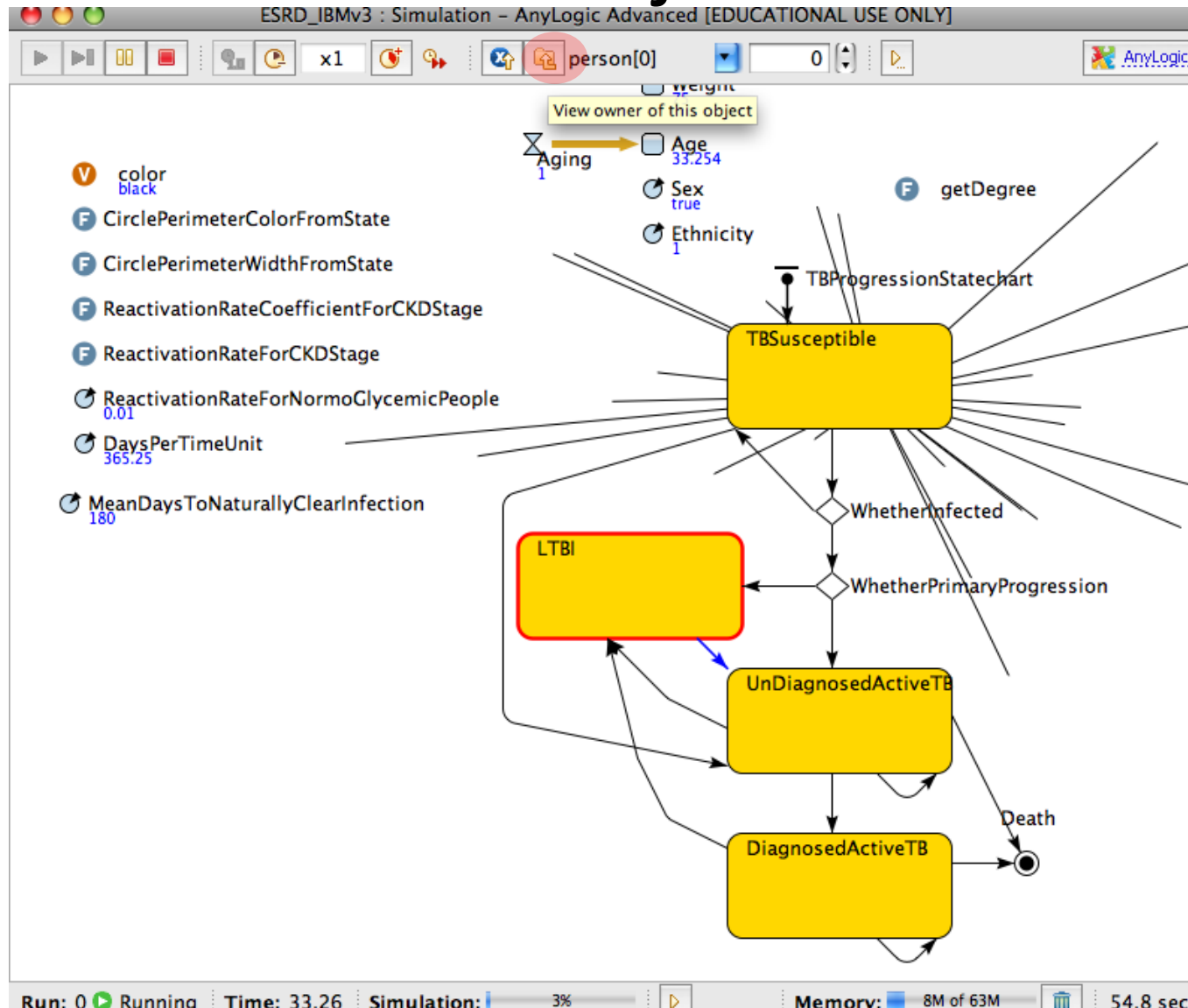
File Menu (Right):

- File
- ✓ Execution control
- ✓ Time scale
- Animation setup
- View
- ✓ Model navigation

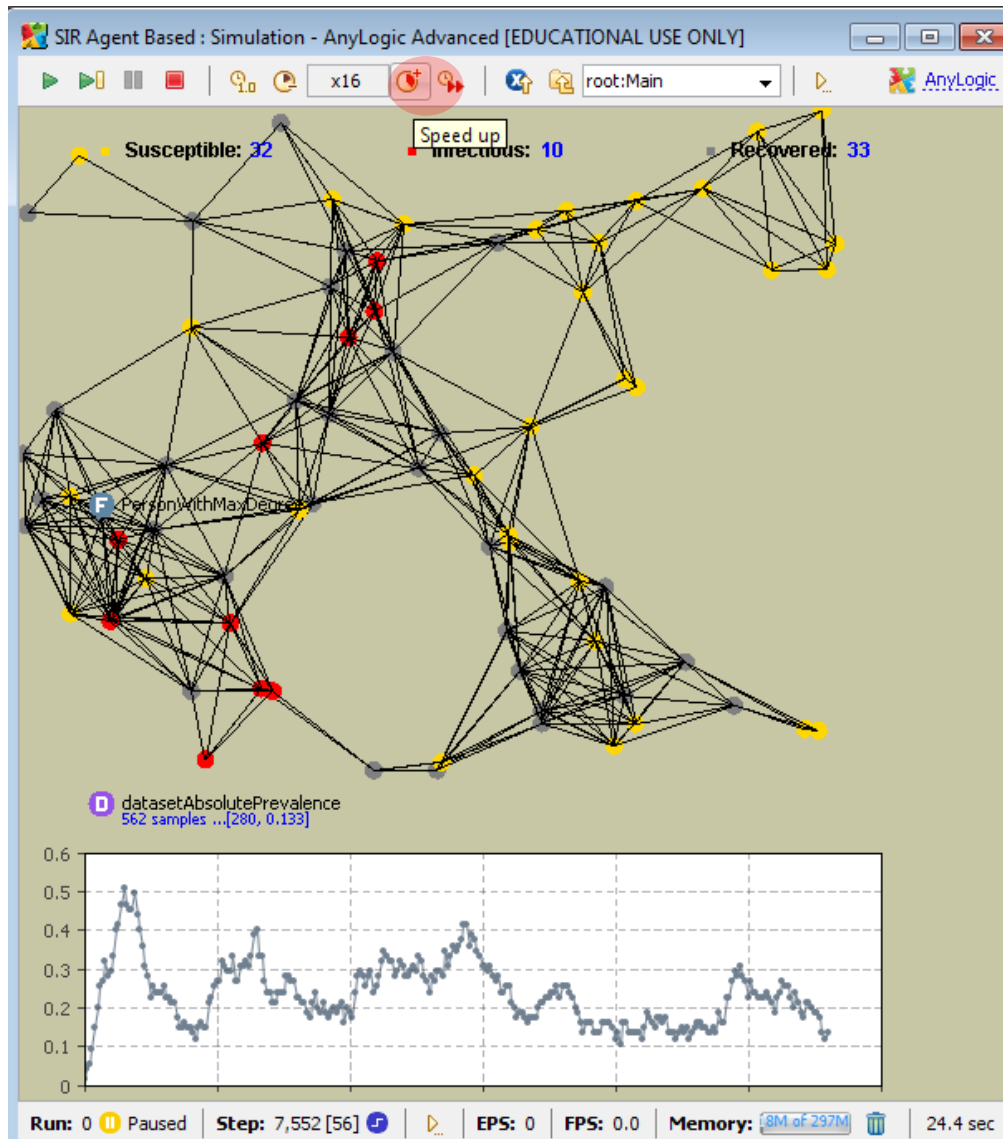
Bottom Status Bar:

- Run: 0 Running
- Time: 15.04
- Simulation: 2%
- Memory: 8M of 63M
- 30.8 sec

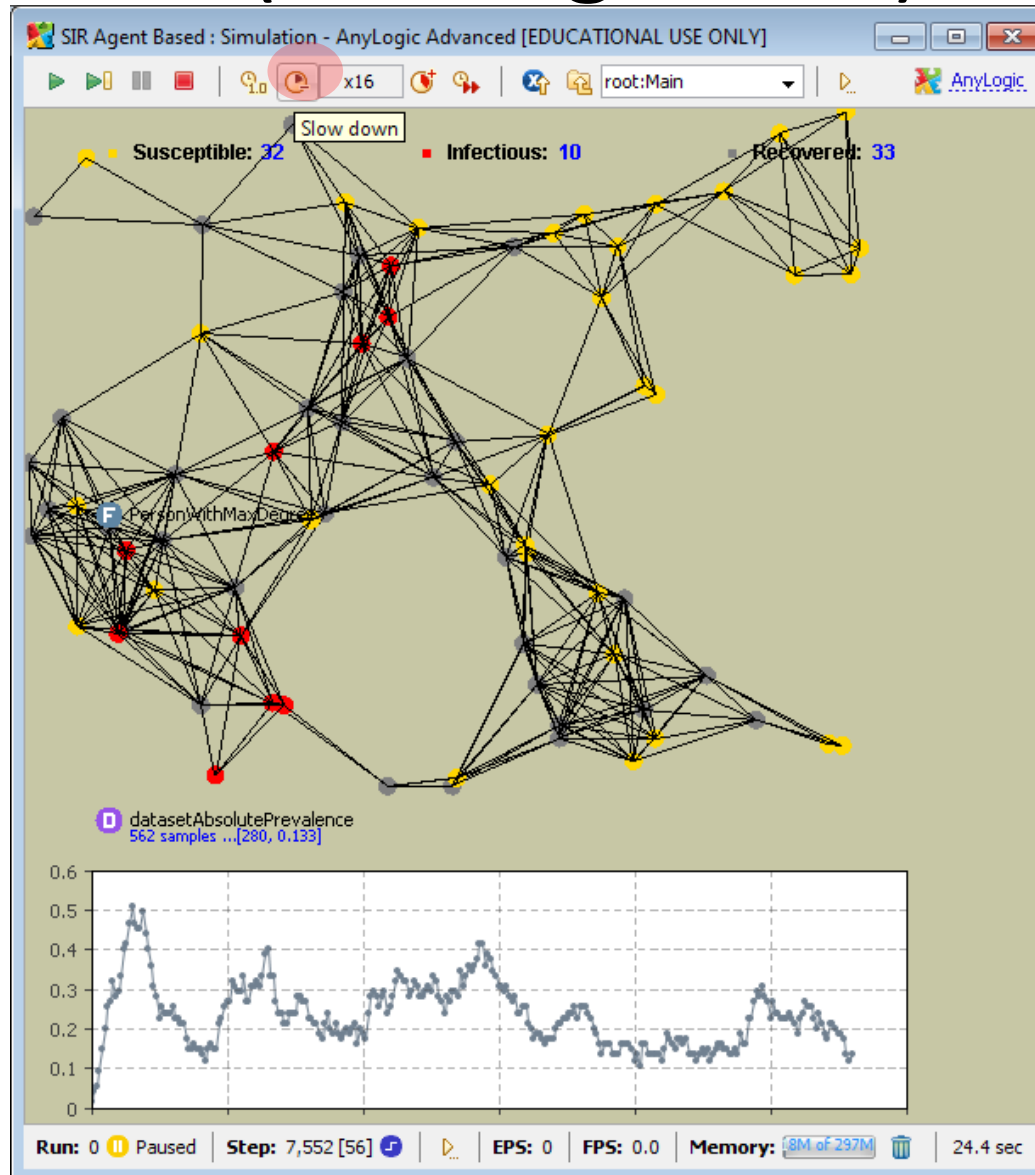
Switching Back to View the Main Object



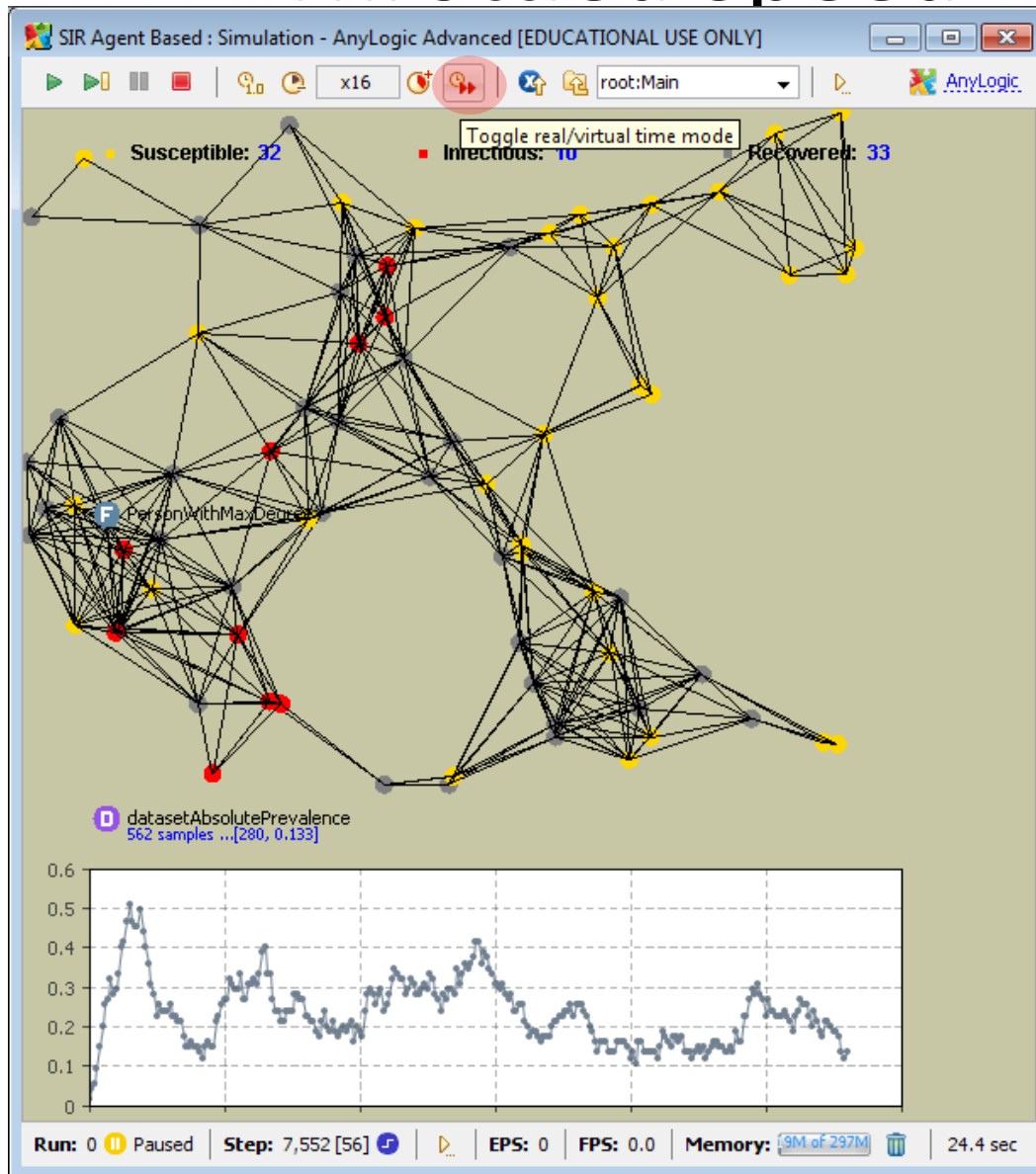
Controlling Simulation Speed (Speeding up)



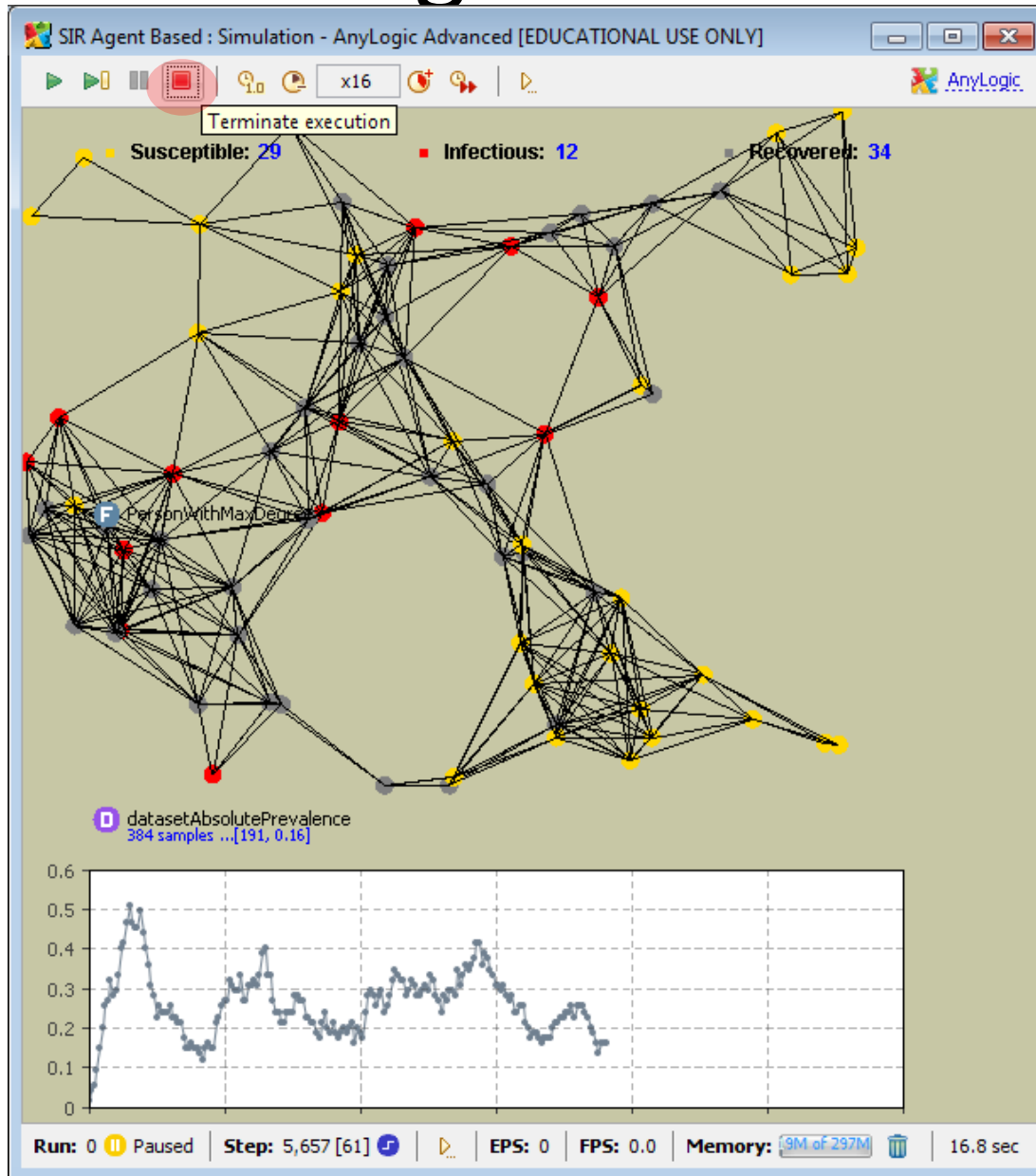
Controlling Simulation Speed (Slowing Down)



toggling between Maximum and a Throttled Speed



Terminating Model Execution



Another Way to Terminate a Simulation

Use this Console “stop” button to terminate the simulation

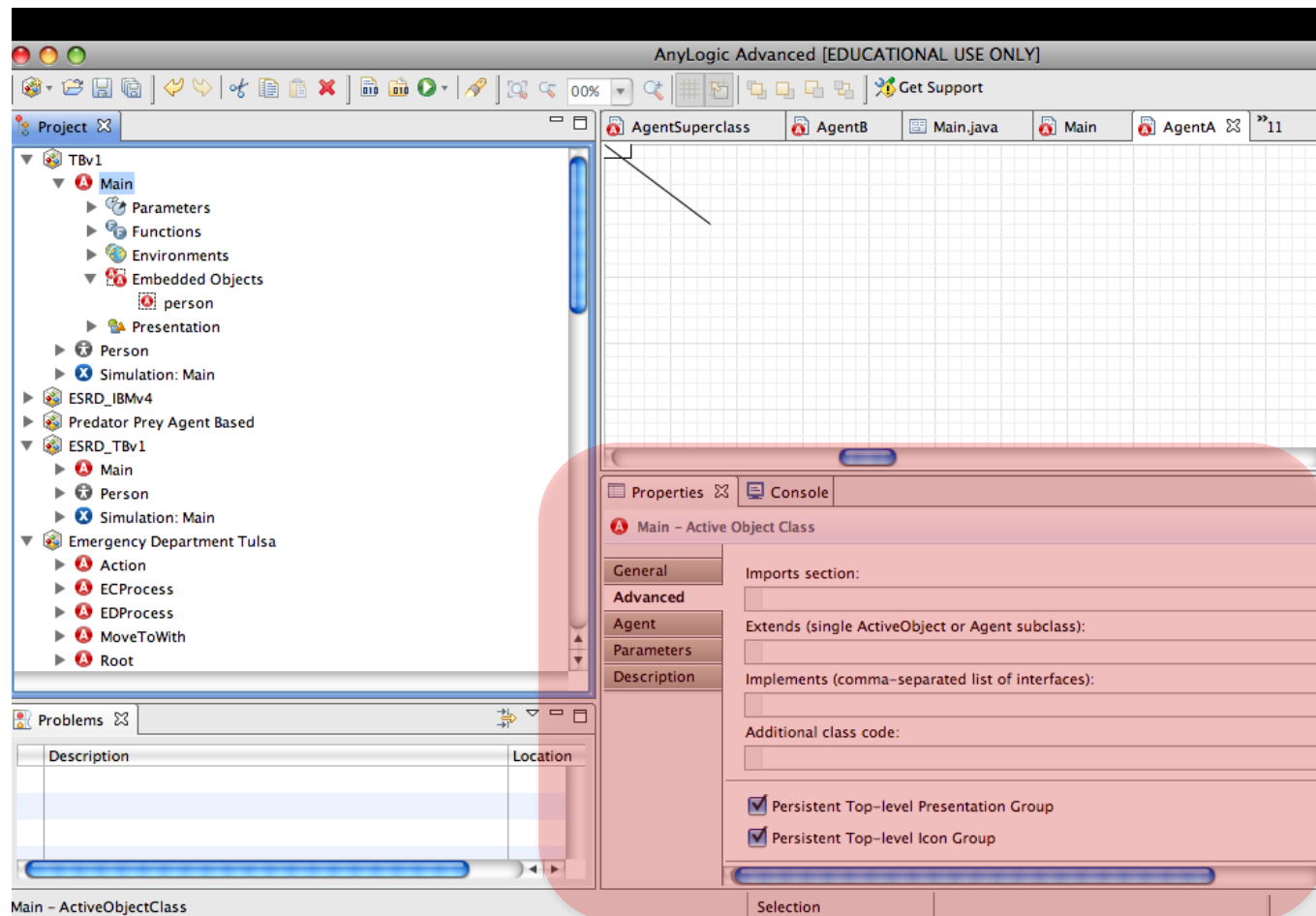
The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a simulation model with several components: 'PersonWithMaxDegree', 'DaysUntilDiagnosis', 'DaysFromDiagnosisUntilRecovery', and 'LikelihoodOfPrimaryProgression'. A red arrow points from the top of the console window to a red circle highlighting a 'stop' button in the console toolbar. The console window shows the following output:

```
anylogic config [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home/bin/java (Oct 29, 2009)
Reactivated
Naturally recovering to LTBI...
Reactivated
Naturally recovering to LTBI...
Reactivated
Developing LTBI directly from primary infection...
Reactivated
Developing LTBI directly from primary infection...
Naturally recovering to LTBI...
Naturally recovering to LTBI...
Naturally recovering to LTBI...
Reactivated
Reactivated
Naturally recovering to LTBI...
```

The interface also includes a Project Explorer on the left, a Properties window, and a Problems window at the bottom left.

Examples of Where to Insert Code Object Properties

- “Advanced”



Examples of Where to Insert Code

Object Properties

- “General”

The screenshot displays the AnyLogic Advanced software interface. The top toolbar includes icons for file operations, navigation, and a 'Get Support' button. The main workspace shows a project tree on the left with the 'Person' object class selected. The right pane is divided into 'Properties' and 'Console' tabs. The 'Properties' tab is active, showing the 'Person - Active Object Class' configuration. The 'General' section is expanded, displaying the 'Name' field set to 'Person' and an 'Ignore' checkbox. The 'Agent' section shows the 'Agent' checkbox checked and the 'Generic' checkbox unchecked. The 'Description' section contains 'Startup Code' and 'Destroy Code' text areas.

AnyLogic Advanced [EDUCATIONAL USE C

AgentSuperclass AgentB Main.java AgentFactory.java

Project TBv1

- Main
 - Parameters
 - Functions
 - Environments
 - Embedded Objects
 - person
 - Presentation
 - person_presentation
- Person
 - Parameters
 - Plain Variables
 - Dynamic Variables
 - Statecharts
 - Functions
 - Presentation
- Simulation: Main
- ESRD_IBMv4
- Predator Prey Agent Based
- ESRD_TBv1
 - Main
 - Person

Properties Console

Person - Active Object Class

General Name: Person Ignore

Advanced

Agent Agent Generic

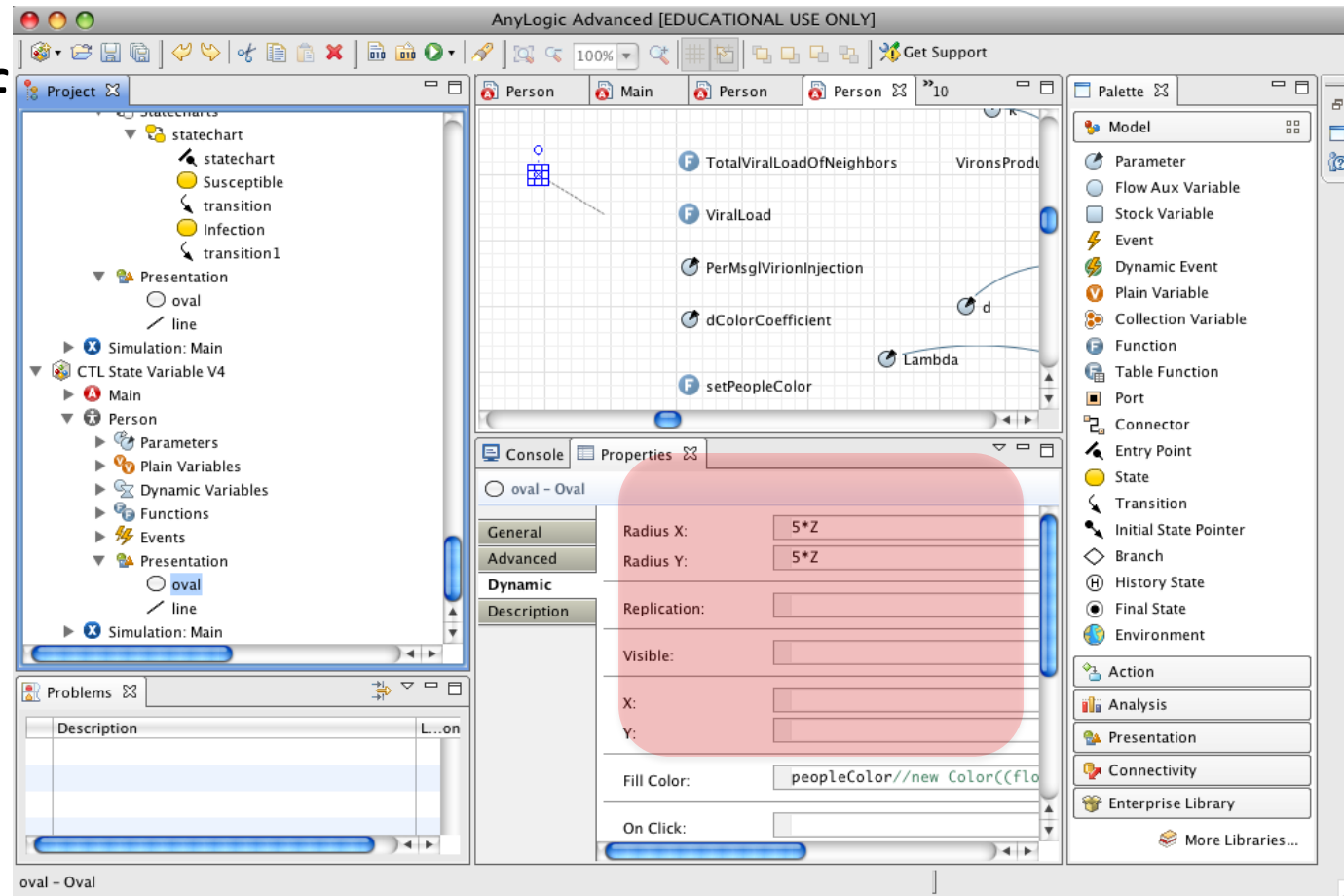
Parameters

Description Startup Code:

Destroy Code:

Example of Where to Insert Code Presentations Properties

- “Dynamic” properties of presentation elements (especially of Agents)



Tips to Bear in Mind While Writing Code

- Click on the “light bulb” next to fields to get contextual advice (e.g. on the variables that are available from context)
- While typing code, can hold down the Control key and press the “Space” key to request autocompletion
 - This can help know what parameters are required for a method, etc.
- Java is case sensitive!
- Can press “Control-J” to go to the point in Java code associated with the current code snippet
- Can press “build” button after writing snippet to increase confidence that code is understood

Example of Contextual Information

The screenshot displays the AnyLogic software interface for an educational simulation. The main workspace shows a statechart with three states: 'people [...]', 'environment', and 'CountInfectious'. The 'CountInfectious' state is selected, and its properties are shown in the Properties window. The Properties window has tabs for General, Parameters, Statistics, and Description. The Statistics tab is active, showing the state's name as 'CountInfectious', its type as 'Count', and its expression as 'Use: item: the embedded object'. The condition for the state is 'item.statechart.isStateActive(Person)'. The left sidebar shows a project tree with a hierarchy: Simulation: Root > Ophthalmology Department > MainPhase1, MainPhase2, MainPhase3 > Simulation: MainPhase3 > NetworkSEIR > Main > Parameters, Plain Variables, Functions, Environments (environment), Embedded Objects, Analysis Data (datasetAbsolutePrevalence, CountInfectious), Presentation, and Person (Plain Variables, Statecharts (statechart)). The right sidebar shows a Palette with 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, Environment, Action, Analysis, Presentation, Connectivity, and Enterprise Library.

AnyLogic Advanced [EDUCATIONAL USE ONLY]

Project

Simulation: Root

Ophthalmology Department

MainPhase1

MainPhase2

MainPhase3

Simulation: MainPhase3

NetworkSEIR

Main

Parameters

Plain Variables

Functions

Environments

environment

Embedded Objects

Analysis Data

datasetAbsolutePrevalence

CountInfectious

Presentation

Person

Plain Variables

Statecharts

statechart

statechart

Person

Main

Person

Person

people [...]

environment

CountInfectious

Console

Properties

people - Person

General

Parameters

Statistics

Description

Name: CountInfectious

Type: Count Sum Average Min Max

Expression: Use: item: the embedded object

Condition: item.statechart.isStateActive(Person)

Add Statistics

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...

Autocompletion Info (via Control-Space)

The screenshot displays the AnyLogic University interface. The main workspace shows a statechart diagram with a 'statechart' node connected to 'Susceptible' and 'Infected' states. The 'Person - Active Object Class' properties window is open, showing the 'On message received' field with the text 'statechart.rece' entered. An autocompletion tooltip is visible, providing details for the 'receiveMessage' method.

Person - Active Object Class

General: X: [], Y: []

Advanced: []

Agent

Preview: []

Description: []

Movement parameters:

Velocity: []

Rotation: []

On arrival: []

On message received: statechart.rece

On before step: []

On step: []

receiveMessage

public boolean receiveMessage(int msg)

Same as receiveMessage(Object msg) but with an integer as message

Parameters:

msg - the integer posted to the statechart

Press 'Tab' from proposal table or click for focus

Selection X=246, Y=275