Events in AnyLogic

Nathaniel Osgood

October 29, 2013

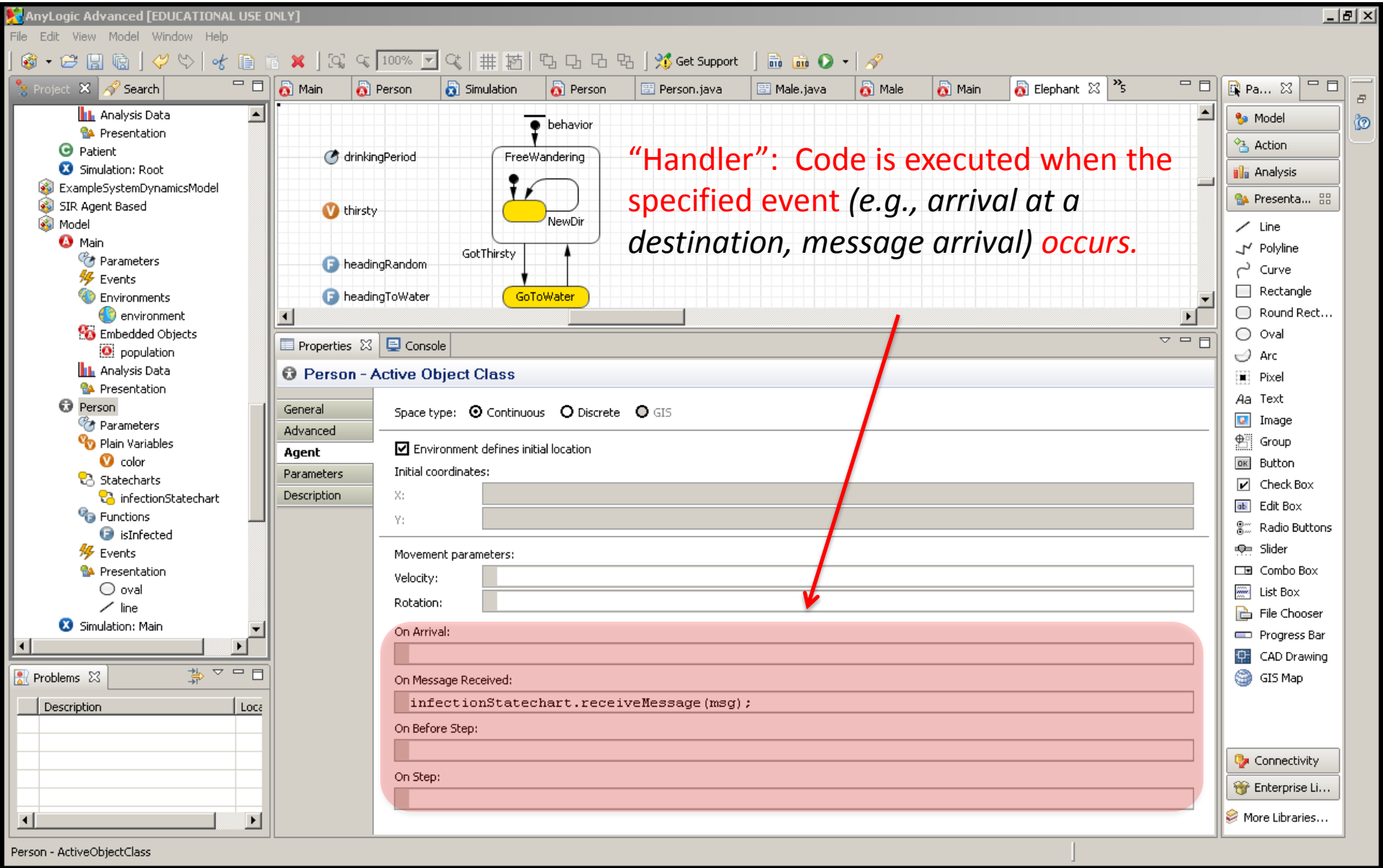
Events & Scheduling in AnyLogic

- Reminder: In simulating stock & flow models, time advances in steps
  - Euler integration: Fixed-sized Steps
  - Runga-Kutta: Fixed or variable sized steps
  - For each timestep, we compute the flows & update the stocks
- AnyLogic jumps from “event” to “event”
  - The data structure that keeps track of such events is called the “schedule”
  - The associated process is called the “scheduler”

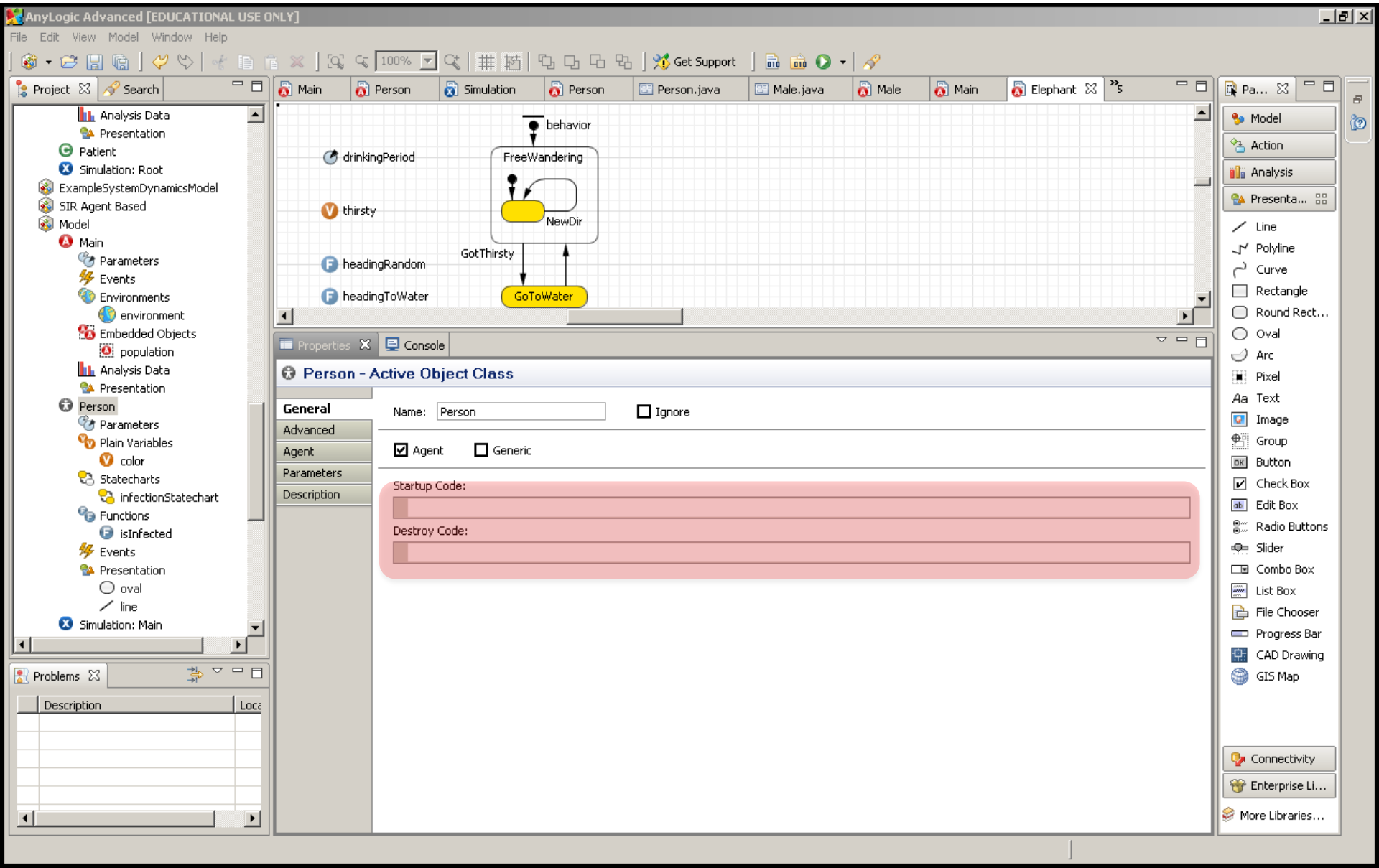
Implicit Events we've Seen

- Transitions
  - Fixed rate (Poisson arrival)
  - Timeout
  - Condition
  - Message transmission (schedules event for the receiver)
- Starting a model
- Stopping a model
- In this course, we term these *implicit events* because they are not reified as objects in the model
- To handle these events, code is inserted into certain handler areas for each of different sorts of classes

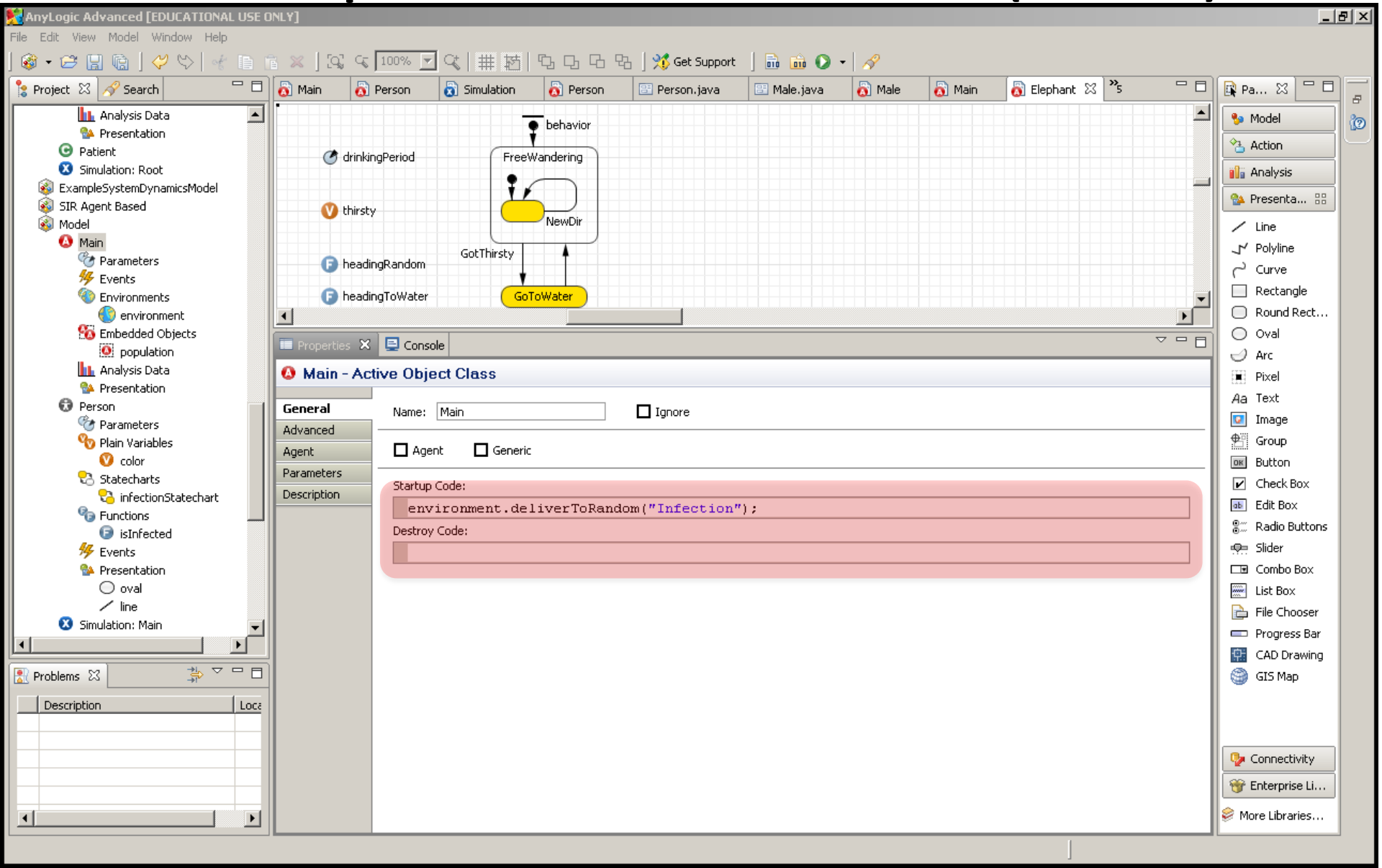
Example: Built-In Events (Agent 1)



Example: Built-In Events (Agent 2)



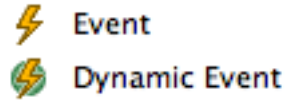
Example: Built-In Events (Main)



The Schedule

- At a given time, the schedule keeps track of a number of queued events
- Events may get added to the schedule (e.g. when we enter a new state)
- Events get deleted from the schedule
  - When they fire off and are complete
  - When another mutually exclusive event preempts them (e.g. a person dies before they recover from an infection)

Explicit Events

- Explicit Events can also be declared from the pallette 

  - Dynamic events can have multiple instances
    - Each instance can be scheduled at different times
    - The instances disappear after event firing
  - Regular (static) events can be rescheduled, enabled/disabled, but can only have one scheduled firing at a time
- There are some subtleties with explicit events

(Explicit) Event Subtleties

- Be very careful of what you count on for recomputation of rate – may think was recomputed, but hasn't been
- Event rates (and likely event timeout times) are only computed occasionally, not continuously
  - These are computed when
    - Explicitly call event methods
      - start()
      - restart()
      - onChange()
        - » e.g. if wish to update rates associated with transitions, *Main* can periodically call *onChange()* on each agent
          - An event in Main can take care of this task
    - When event fires and requires restarting
    - (For outgoing transitions) when enter a state in a statechart
- Calling "reset" will disable a rate until re-enable (e.g. with call to *restart()*)

Event Times: Common Options for Event Scheduling

- At a specified rate (Poisson arrivals)
  - Interarrival time is exponentially distributed!
  - Mean time between events is reciprocal of rate (i.e. 1/rate)
- One-time
  - Can go off at a particular time (specified as a calendar time or as a double-precision value)
- At some initial time and then cyclically beyond with set "timeout" period
  - The timeout period is set according to the time unit
  - This goes off after *exactly* the timeout time
- When boolean condition changes (depends on *onChange* being called)
- Manually (via restart() – see following slides)

Event Subtleties

- Be very careful of what you count on for recomputation of rates – may think was recomputed, but hasn't been
- Event rates (and likely event timeout times) are only computed occasionally, not continuously
  - These are computed when
    - Explicitly call event methods
      - start()
      - restart()
      - onChange()
    - When event fires and requires restarting
    - (For outgoing transitions) when enter a state in a statechart
- Calling "reset" will disable a rate until re-enable (e.g. with call to *restart()*)

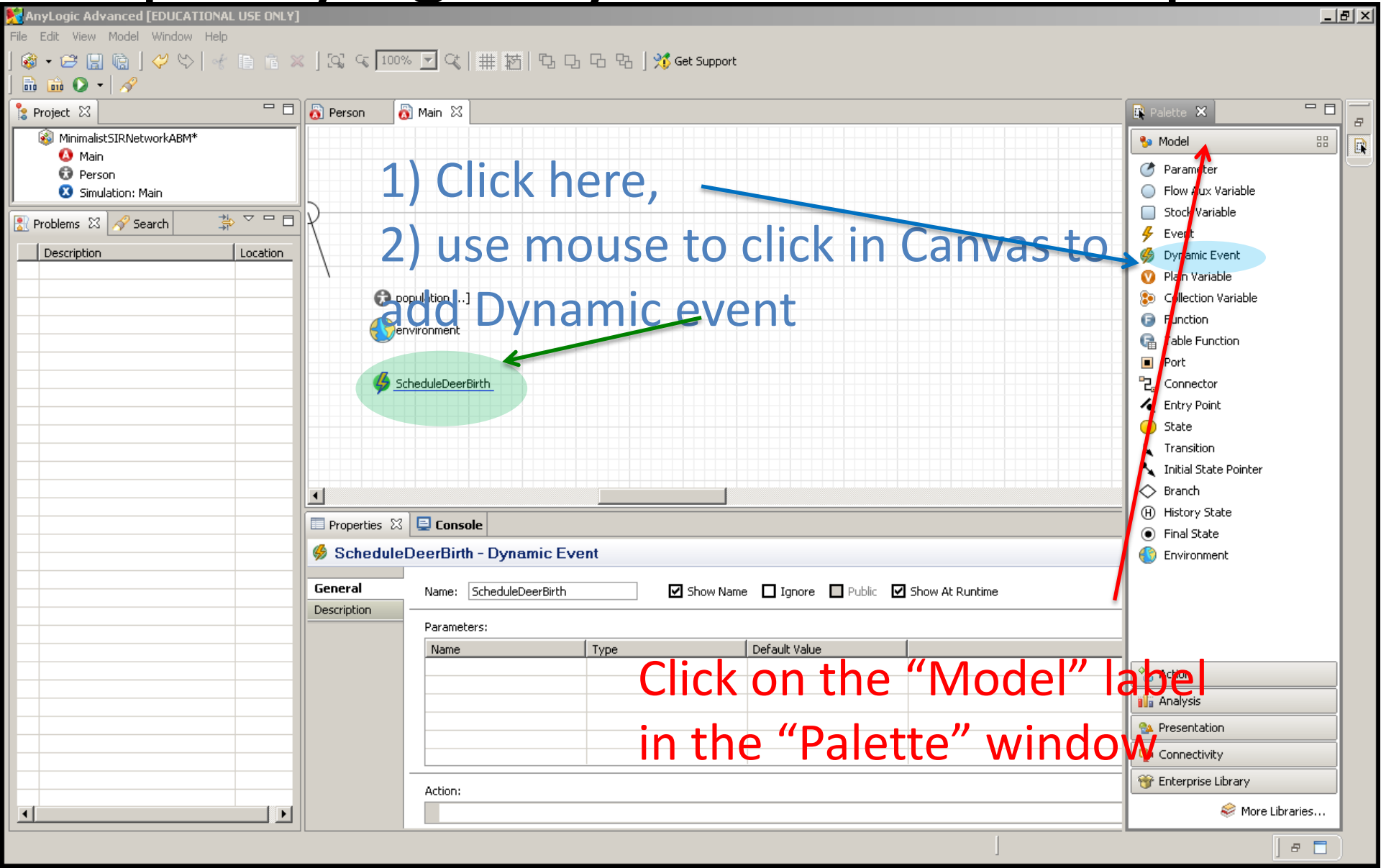
Dynamic Events (Closure-Like)

- Like a static event, a dynamic event is associated with an *action* to invoke when it occurs
- A *static* event has a single associated schedule
- Just as a class can be associated with multiple instances, Dynamic events can have *multiple instances*
  - Each instance can be scheduled at different times
  - The schedule for each different instance proceed in parallel
  - The instances disappear after event firing
    - We can think of each dynamic event instance as its own one-time ("one-shot") event
- Schedule a dynamic event with create_event(*timeout*, parameters…)
  - The event will be "awoken" time *timeout* from now!

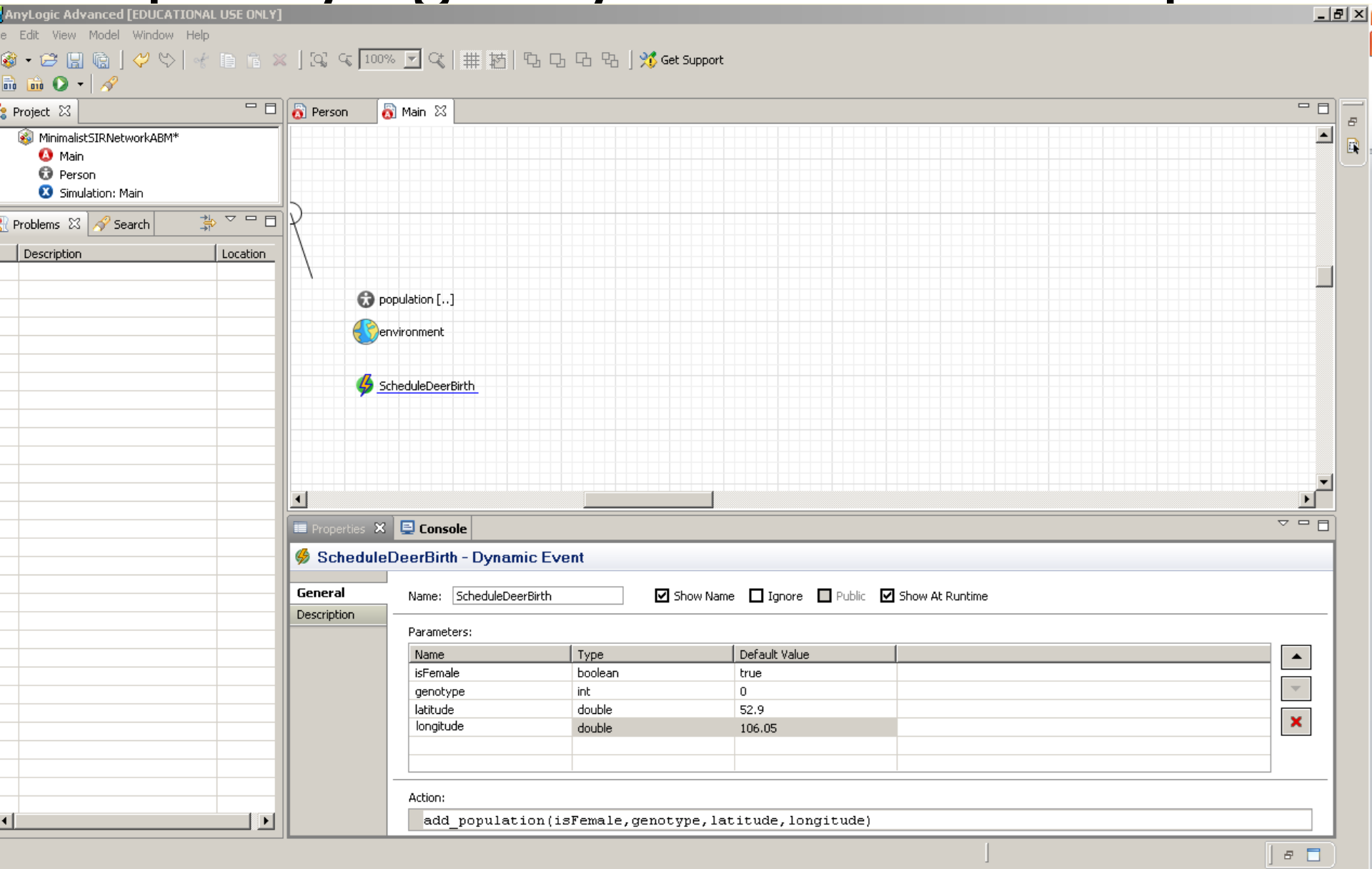
Parameterization of Dynamic Events

- With a dynamic event, we create the event during simulation, but at a different time than it occurs
- Frequently the action we want to performed in a dynamic event depends on specific context known at the time that it was created
  - For example, we want to create or delete a particular person, or a person with particular characteristics
- Specification of dynamic events at design time defines custom ‘parameters’ (‘arguments’)
  - Parameters values can be used to communicate context from time of creation of the dynamic event until when it fires
  - Particular values for these parameters are then given at time when dynamic event instance is created

Specifying a Dynamic Event Step 1



Specifying a Dynamic Event Step 2



Attractive Use of Dynamic Events 1

Scheduling Future Birth at time of Conception

- Mating of deer during rut occurs long before births of fawns
- Contacts between deer during rut could be simulated in the model
  - At time of contact, create single dynamic event to schedule associated future birth
  - Could save away information of history relevance e.g.
    - Characteristics of parents
      - Infection status
      - Genotype
      - Stress level
    - Location of where conception occurred

Attractive Use of Dynamic Events 2

Adding in Individuals to Population over a Time Interval

- Dynamic events can be very handy if have a known number of actions that need to take place spread out over some period of time
- Example: Given: Known count of Immigrants with particular characteristics to be added to model population over course of each month
  - Suppose we don't know when these individuals arrive during the month
  - We can simply create the same count of dynamic events, whether each dynamic event takes care of
    - Creating a person with known characteristics
    - Adding that person to the model population

This approach will be discussed in an upcoming guest lecture