Basic Energy User Tutorial v03 (23/04/2013)

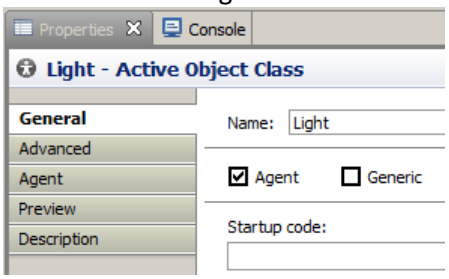
Peer-Olaf Siebers
Nottingham University
peer-olaf.siebers@nottingham.ac.uk

Functions are provided in the attached model! Just copy and paste them as required!

Create a new model
Create active object classes

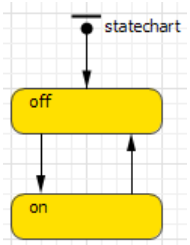
- Light
- Office
- User

Mark them all as agents



Add simple state charts to all agent classes

- Light states: off/on
- Office states: vacant/occupied
- User states: outOfOffice/inOffice



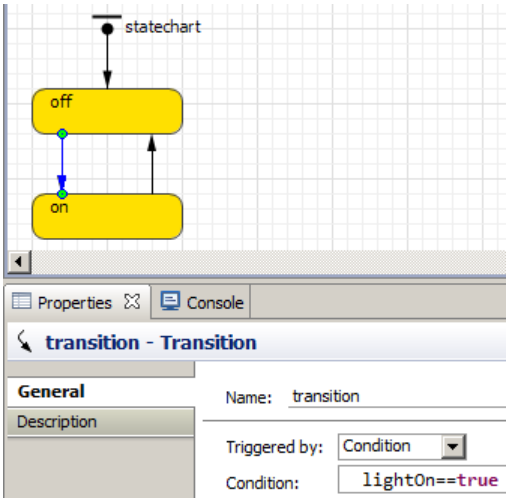
Light Class

Add variables

- {Name} power {Type} double (for displaying the power consumption in a time plot)
- {Name} lightOn {Type} boolean (for controlling transitions)
- {Name} office {Type} Other : Office (for storing information about the office instance the light is installed in)

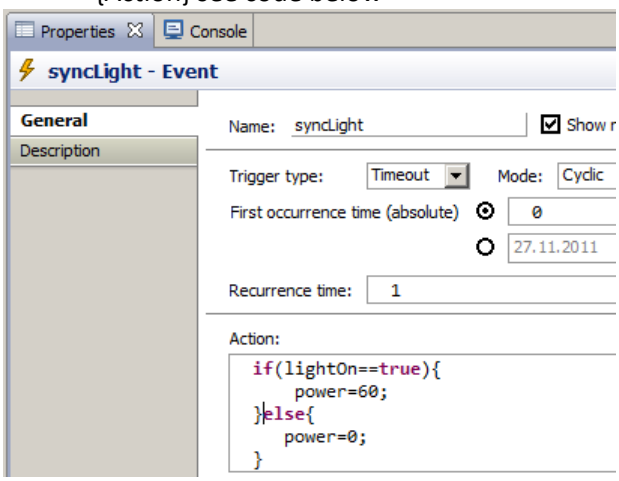
Add conditions for transitions

- From state "off" to state "on": {Triggered by} Condition; Condition: lightOn==true
- From state "on" to state "off": {Triggered by} Condition; Condition: lightOn==false



Add event

- {Name} syncLight (required for correctly updating the power consumption time plot)
- {Mode} Cyclic
- {Action} see code below



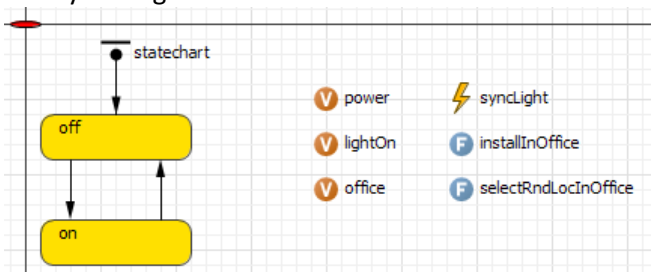
Add the two functions provided (these functions have additional code which we ignore for now)

- installInOffice (chooses the office the light will be installed in)
- selectRndLocInOffice (chooses the location for displaying the light)

Add an Oval from the [Presentation] pallet

- {Fill color} red
- [Advanced] {Position x} 0; {Position y} 0
- [Advanced] {Radius x} 10; {Radius y} 2
- [Dynamic] {Visible} statechart.isStateActive(on)

Finally the Light class looks like this



Office class

Add variables

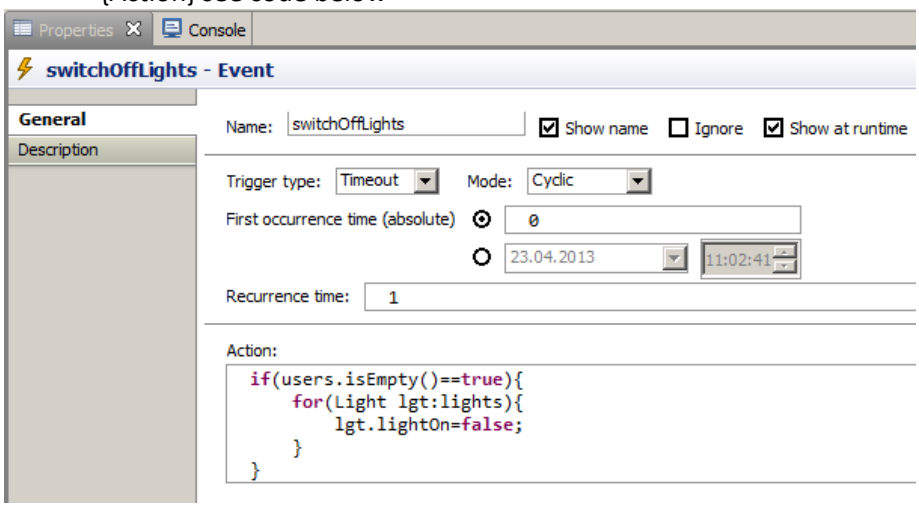
- {Name} officeCapacity {Type} int (max. number of users)
- {Name} officeLights {Type} int (number of office lights)
- {Name} shapeLine {Type} Other : ShapePolyLine (defining the boundaries of the office)

Add collection variables

- {Name} lights {Elements class} Light (light instances related to this office)
- {Name} users {Elements class} User (user instances related to this office)

Add event

- {Name} switchOffLights (calculates energy consumption and switches off lights if room is empty)
- {Mode} Cyclic
- {Action} see code below

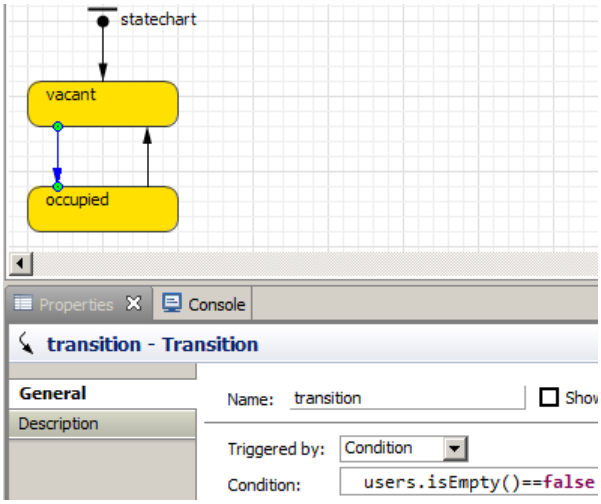


Add the two functions provided

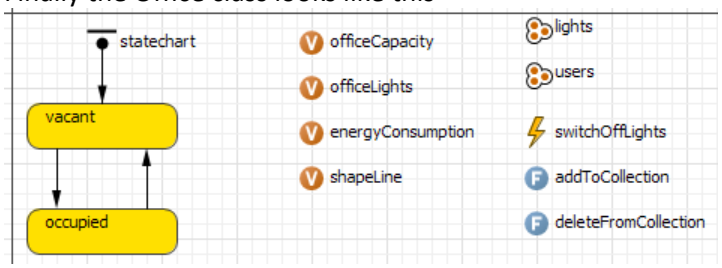
- addToCollection (adds objects to the collection variables)
- deleteFromCollection (deletes objects from the collection variables)

Add conditions for transitions

- From state "vacant" to "occupied": {Triggered by} Condition; Condition: users.isEmpty()==false
- From state "occupied" to "vacant": {Triggered by} Condition; Condition: users.isEmpty()==true



Finally the Office class looks like this

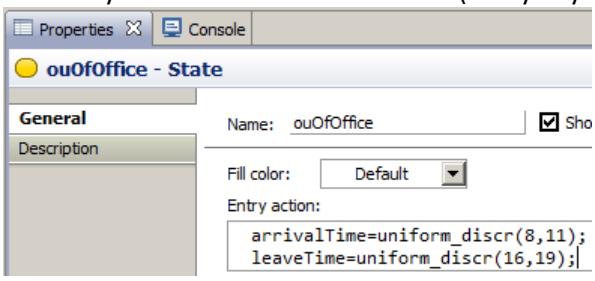


User class

Add variables

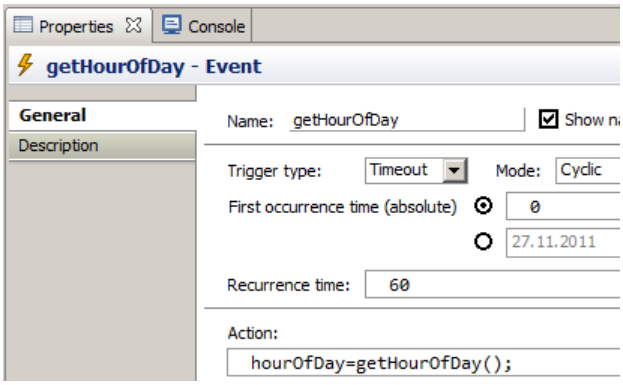
- {Name} ownOffice {Type} Other : Office (the user's office)
- {Name} arrivalTime {Type} int (arrival time (drawn from uniform distribution))
- {Name} leaveTime {Type} int (arrival time (drawn from uniform distribution))
- {Name} hourOfDay {Type} int (auxiliary variable storing the time of day)

Add Entry action for outOfOffice state (every day a new random time for arrival/leave is produces)



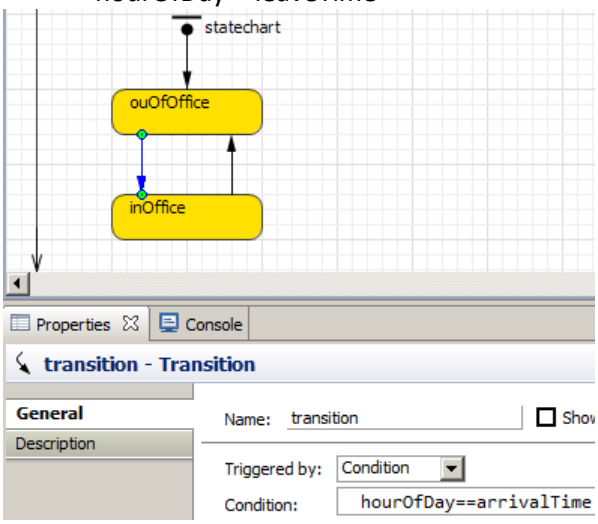
Add event

- {Name} getHourOfDay (update auxiliary variable required for the transitions)
- {Mode} Cyclic
- {Recurrence time} 60
- {Action} see code below
- (perhaps later you want to try "`(hourOfDay==arrivalTime)&&(randomTrue(0.7))`") as some people might not come to work every day!



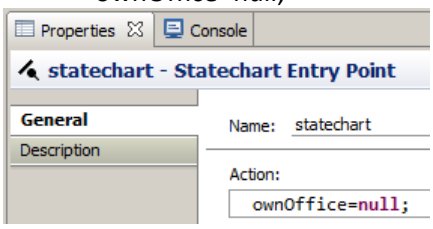
Add conditions for transitions

- From state "outOfOffice" to "inOffice": {Triggered by} Condition; Condition: hourOfDay==arrivalTime
- From state "inOffice" to "outOfOffice": {Triggered by} Condition; Condition: hourOfDay==leaveTime



Add Statechart Entry Point Action

- ownOffice=null;



Add a person picture from the [Pictures] pallet

- [Advanced] {Position x} 0; {Position y} 0
- [Dynamic] {Visible} statechart.isStateActive(inOffice)

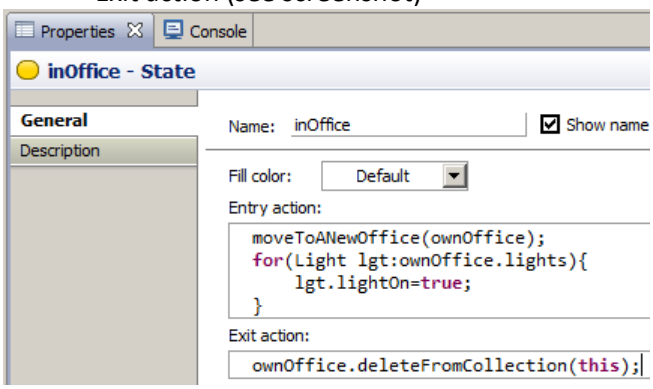
Add the two functions provided (these functions have additional code which we ignore for now)

- moveToANewOffice (chooses the office the light will be installed in)
- selectRndLocInOffice (chooses the location for displaying the light)

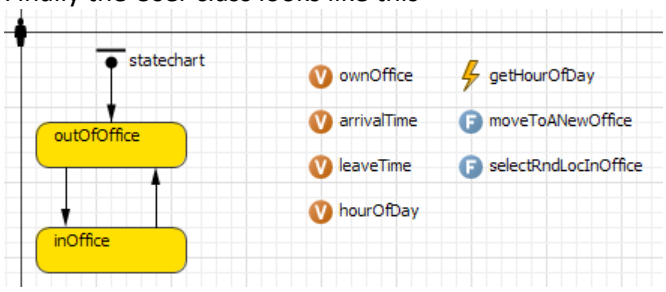
Add code to the inOffice state

- Entry action (see screenshot)

- Exit action (see screenshot)



Finally the User class looks like this



Main class

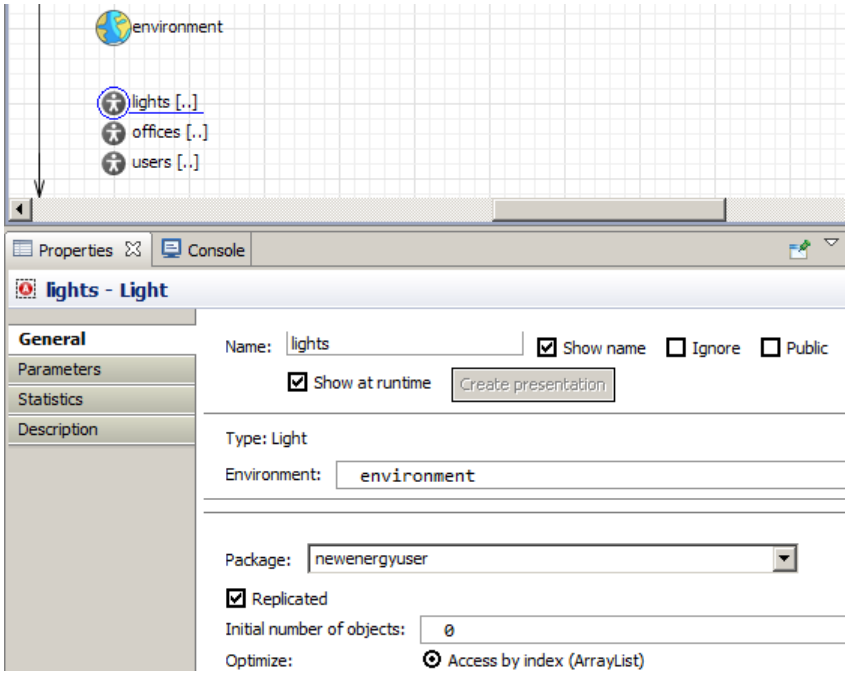
{Startup code}

- Add the code provided (for creating offices, lights, and users)

Add an Environment from the [General] pallet

Instantiate the Light, Office, and User classes and name the instances lights, offices, and users
For all three:

- Choose "environment" for {Environment}
- {Replicated} needs to be switched on (but we create our object in Main : Startup code)

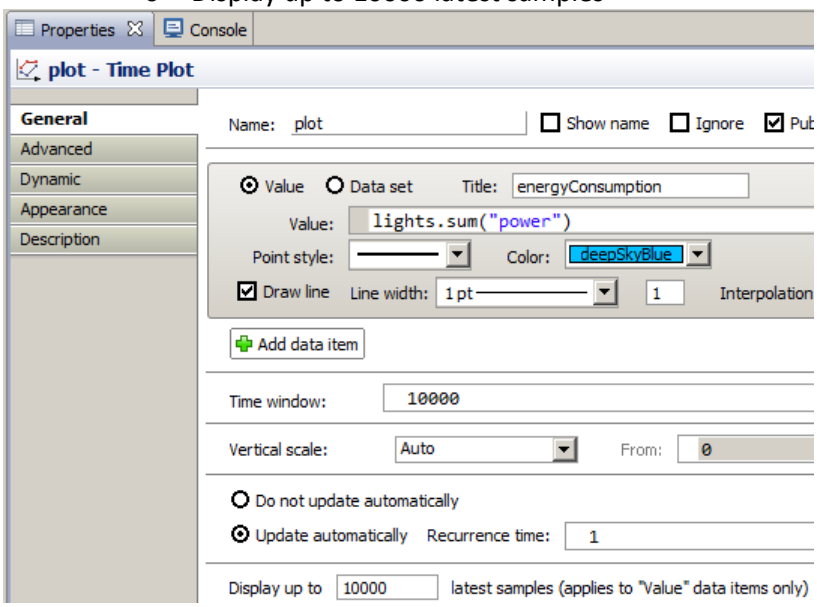


Draw four boxes (three smaller ones and one larger one) using polylines from the [Presentation] tab to represent the offices and name them p1 ... p4

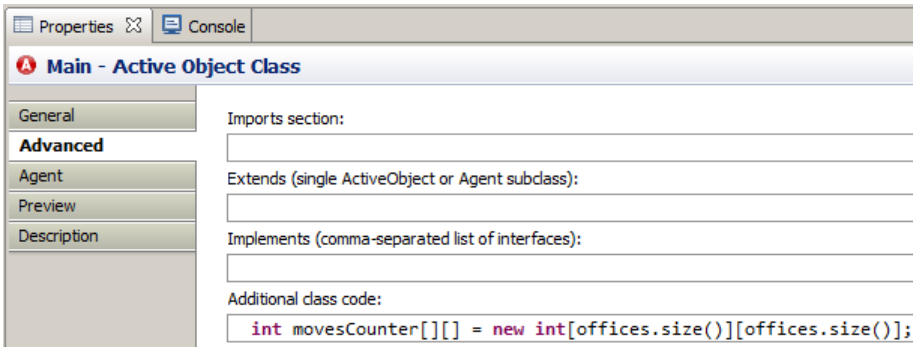
Add an image place holder from the [Presentation] pallet as the background for the offices and do not change the name.

Add a TimePlot from the [Analysis] pallet

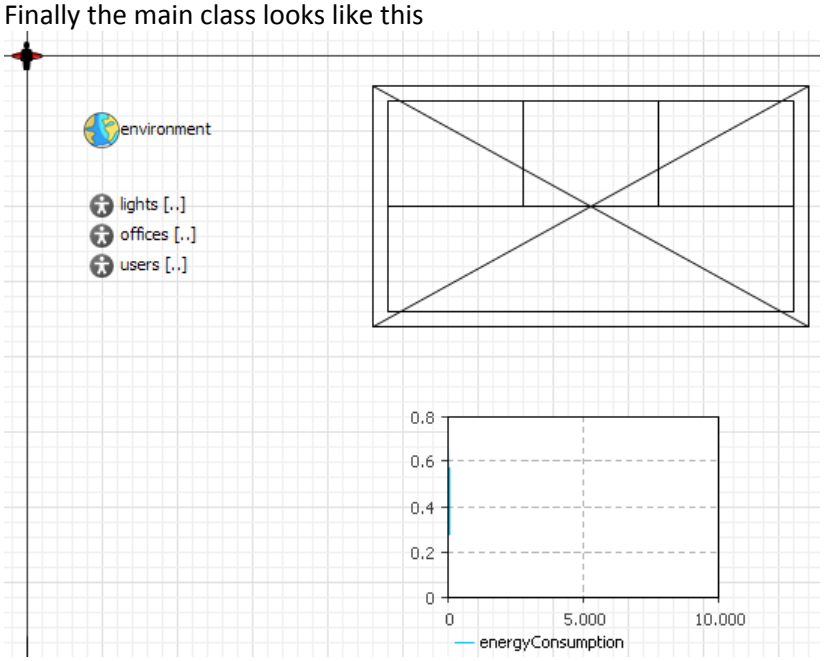
- [General] Add a DataItem
    - Title = energConsumption
    - Value = lights.sum("power")
    - TimeWindow = 10000
    - Display up to 10000 latest samples



Add code in Advanced/AdditionalClassCode



Finally the main class looks like this



Run the model
Set simulation speed to "x250"