ASSS 2014 Tutorial: Laptop Simulation

Task

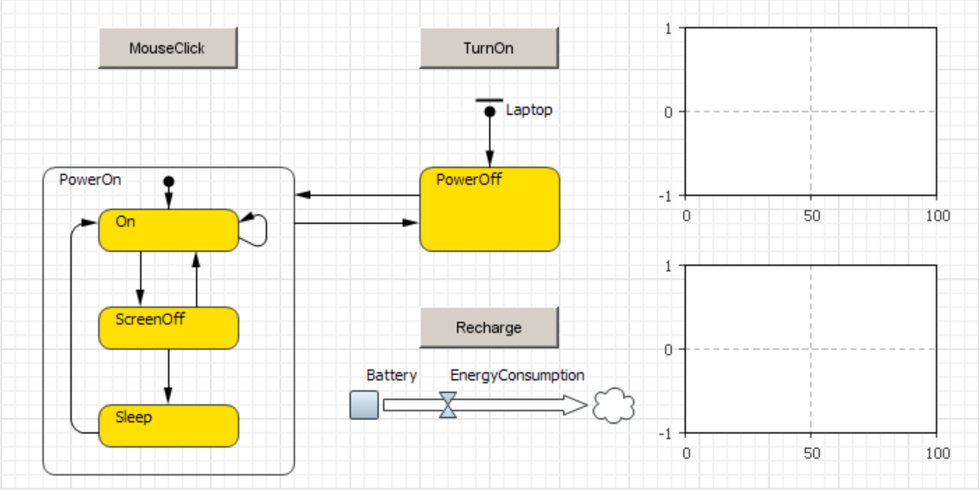
- Build a simulation model of a laptop operation. When the laptop is on and the user is working, i.e. is pressing the keyboard keys and moving the mouse, the laptop stays on. After 5 minutes of user inactivity the laptop turns off the screen to save power. If the user remains inactive for another 10 minutes, the laptop switches to sleep mode to further minimise the battery usage. To wake the laptop up you need to press the power button. In any of the three states On, Screen Off and Sleep the laptop consumes the battery power. When the battery level falls down to 10% the laptop is forced to shut down regardless of the state.

Planning

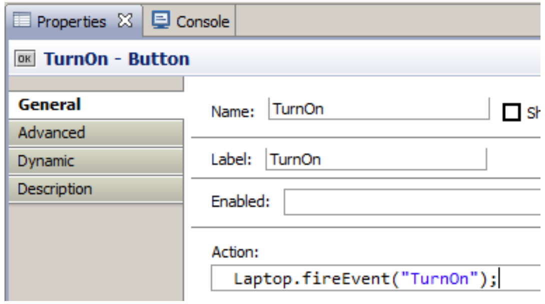
- With your colleagues: Create a conceptual model using a UML state chart

Implementation

- Create new blank project
- Drag all required elements into the main window
  - States and transitions are in the Statechart palette
  - Buttons are in the Control palette
  - Battery elements are in the System Dynamics palette
  - Time Plot is in the Analysis palette
  - Tip: To bend the transition arrows double click on the line – this will produce a bending point which you then can move into the desired position



- Button triggered events:
  - Add code to the button {TurnOn}: [Action: Laptop.fireEvent("TurnOn"]



Name: TurnOn

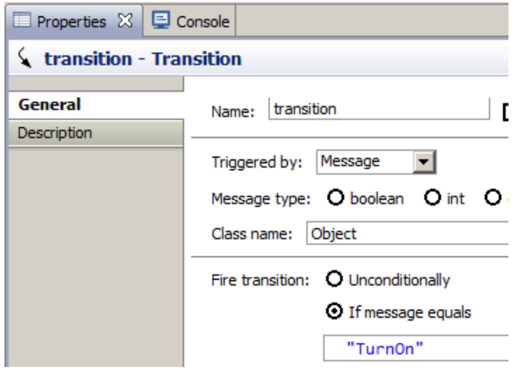
Label: TurnOn

Enabled:

Action:

```
Laptop.fireEvent("TurnOn");
```

- ○ Add code to the transitions {PowerOff=>PowerOn}: [Triggered by: Message]; [File transition: If message equals: "TurnOn"]



Name: transition

Triggered by: Message

Message type: ○ boolean ○ int ○

Class name: Object

Fire transition: ○ Unconditionally

● If message equals

"TurnOn"

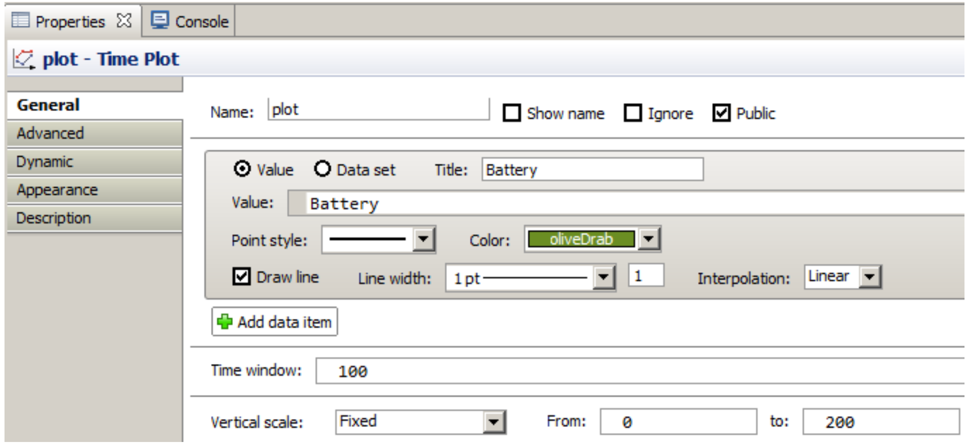
- ○ Compile and run
- ○ Add code to the button {MouseClick}: [Action: Laptop.fireEvent("MouseClick"]
- ○ Add code to the transitions {PowerOn/On=>PowerOn/On}: [Triggered by: Message]; [File transition: If message equals: "MouseClick"]
- ○ Do the same for transition {PowerOn/ScreenOff=>PowerOn/On} [Triggered by: Message]; [File transition: If message equals: "MouseClick"]
- ○ Add code to the transition {PowerOn/Sleep=>PowerOn/On} [Triggered by: Message]; [File transition: If message equals: "TurnOn"]

- Time triggered events:
  - ○ Add code to the transitions {PowerOn/On=>PowerOn/ScreenOff}: [Triggered by: Timeout]; [Timeout: 5]
  - ○ Add code to the transitions {PowerOn/ScreenOff=>PowerOn/Sleep}: [Triggered by: Timeout]; [Timeout: 10]
  - ○ Compile and run [it goes from {PowerOn/On} directly to {PowerOff} and never reaches the other {PowerOn} states :-(]
- Battery
  - ○ Add code to the stock {Battery}: [Initial value: 100]; (the formula [d(Battery)/dt=: -EnergyConsumption] is automatically created for you :-)]
  - ○ Add code to the flow {EnergyConsumption} [EnergyConsumption=:

```
        Laptop.isStateActive(On)?5 :
                Laptop.isStateActive(ScreenOff)?3 :
                        Laptop.isStateActive(Sleep)?1:0
```

  - ○ Add code to button {Recharge} [Action: if(Laptop.isStateActive(PowerOff))Battery=100]

  - Add code to the transition {PowerOn=>PowerOff}: [Triggered by: Condition]; [Condition: Battery<=10]
  - Compile and run ... mine works now as expected
- Graphs
  - Choose top graph (we use it to display the battery load) [Click +Add data item]; [Value Title: Battery; Value: Battery] [Vertical scale: Fixed; From 0; To: 200]



plot - Time Plot

Name: plot □ Show name □ Ignore ☑ Public

● Value ○ Data set Title: Battery

Value: Battery

Point style: Color: oliveDrab

☑ Draw line Line width: 1 pt 1 Interpolation: Linear

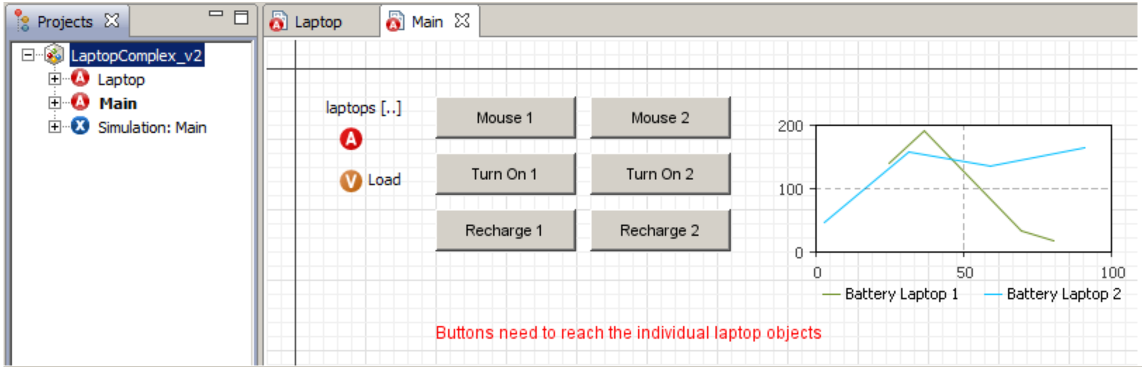
Time window: 100

Vertical scale: Fixed From: 0 to: 200

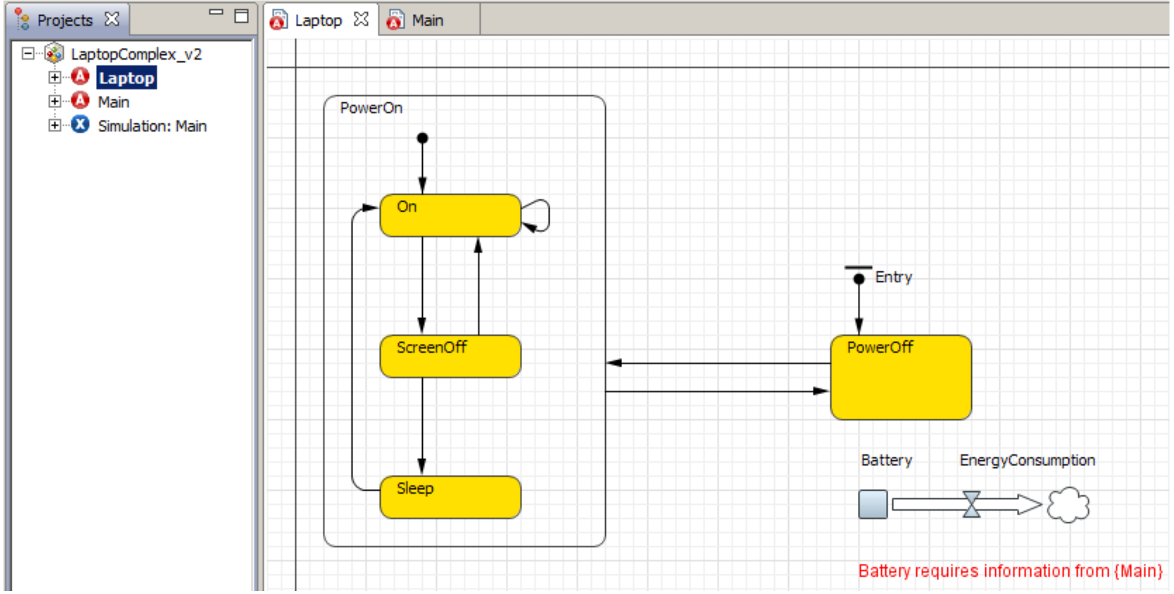
  - Choose bottom graph (we use it to display the energy consumption) [Click +Add data item]; [Value Title: EnergyConsumption; Value: EnergyConsumption] [Vertical scale: Fixed; From 0; To: 10]
  - Compile and run ... that's it :-)

Challenges

1. Try to add off-switch
2. Try to add a combined on/off switch
3. Try to create multiple laptops using an object oriented approach (you might want to come back to this once you learned a bit more about AnyLogic)
   - Create a laptop class
   - Create several instances of the class in the main class



Buttons need to reach the individual laptop objects



Battery requires information from {Main}