Simulation Visualization (KViz)

- Overview
- Creating a Configuration File

Overview

A simple table with all variables is sometimes not enough to test and visualize the behavior of a model. If one programs a big system, such as a railway controller, it might help to see on which track the trains are and how points are set. KViz is used to achieve this by using javascript to manipulate an svg (or html in general).

This is done by generating an html page that is displayed in a browser. Therefore used SVGs and scripts are dependent on the used browser.

Creating a Configuration File

Example KViZ File

```
image "Lights.svg"

load "some-js-library" // These libraries are copied in a script tag for the corresponding html page that is
build.
load "some-other-js-library"

init
'
// javascript global variable initialization
// and function definition
// e.g. var hugo = "hugo"
'

handle showLight
in "#theRect"
with (elem, status)
'
// elem is the element referenced via "in"
// status is the value of showLight
elem.style.fill = status === 0 ? "red" : status === 1 ? "yellow" : "green"
'

handle showLight
in "#theRedOne"
with (elem, status)
'
// elem is the element referenced via "in"
// status is the value of showLight
elem.style.opacity = status === 1 || status === 2 ? 0 : 1
'

handle showLight
in "#theRedOne"
with (elem, status)
'
// elem is the element referenced via "in"
// status is the value of showLight
elem.style.opacity = status === 0 || status === 2 ? 0 : 1
'

handle showLight
in "#theRedOne"
with (elem, status)
'
// elem is the element referenced via "in"
// status is the value of showLight
elem.style.opacity = status === 0 || status === 1 ? 0 : 1
'
```

Simulation Interaction

Interaction example

```
image "Lights.svg"

handle showLight
in "#theRect"
with (elem, status, datapool) // datapool can always be added to have access to every value in the datapool
'
// elem is the element referenced via "in"
// status is the value of showLight
elem.style.fill = status === 0 ? "red" : status === 1 ? "yellow" : "green"
'

event "click"
on "#theRect"
simulation step // executes an actual simulation step

event "click"
on "#theRect"
set showLight
'
if (showLight != 1) {
    return 1 // this sets showLight to 1
}
'

// Only set this boolean variable in this tick
event "click"
on "#theRedOne"
signal thisIsABoolean
```

Scripts that are not associated to a specific value

Script example

```
image "Lights.svg"

script (datapool)
'
// You are able to write to inputs of the model via the datapool.
datapool.onInput = true
'
```

Scripts can always be associated to a value instead via the *handle* keyword. Not doing so shows that it is a general script that is invoked every tick and does not depend on the change of a specific value.