R3 GUI Actor/Reactor design changes

Author: Richard Smolak

Revision History

20-Jan-2013	ARS

Table of Contents

1. Introduction

This document describes the changes related to the decision to simplify the current actor/reactor model.

2. No more Reactors

The Reactors have been completely removed from the system because we (the RMA team) agreed they make the usage of the framework confusing and not flexible enough.

After the changes made to the R3GUI the programmer will need to deal only with Actors.

3. Actor Overriding

Here is an example how to override a style (default) actor of a face:

```
view [
        my-field: field on-key [
                ;arg holds event object
                if arg/type = 'key [
                        print ["field" face/name "key:" mold arg/key]
                ]
        ]
]
```

Running the above code you should see that the `on-key` actor has been overridden in the `my-field` face and the new code is executed instead.

But now the typed characters aren't shown in the field. That's because the style `on-key` actor is not called anymore.

Here is a modified code of the above example executing the style actor after the newly added code:

```
view [
```

```
        my-field: field on-key [
                ;arg holds event object
                if arg/type = 'key [
                        print ["field" face/name "key:" mold arg/key]
                ]

                ;now call the FIELD style actor
                do-actor/style face 'on-key arg face/style
        ]
]
```

This techinque is flexible enough so the programmer can call the style actor at any possible place in the new code that overrides the style actor.

.

Note

NOTES:

1. Should we introduce API function wrapper for calling the style actor in some easier way?
2. If yes, what syntax form and name should have such function wrapper?

4. ON-ACTION actor

Because the DO reactor doesn't exist in the system, the ON-ACTION actor has been added as *system actor* for the default style terminal action.

The ON-ACTION actor is useful if the style needs to call some default action from multiple places (actors) in the style definition.

For example, the BUTTON style needs to call the default style action from the ON-KEY actor and also from the ON-CLICK actor, so it is better to call the ON-ACTION actor from the both code points to avoid the necessity to override multiple style actors.

The ON-ACTION actor can be overridden in the layout block without the need to call the original style actor code. Here is an example:

```
view [
        button on-action [print "button pressed"]
]
```

.

Note

NOTES:

Styles should not define the ON-ACTION actor. This way the layout programmer doesn't need to deal with any interferences.

Do we need a better name for this actor?

5. Face ATTACH mechanism

The face attaching subsystem has been reworked. The ATTACH reactor doesn't exist. The ATTACH keyword has been added to the layout dialect to keep the current syntax.

Summary of main changes in the subsystem:

- the face/targets field contains faces that are attached FROM the face
- the face/attached field contains faces that are attached TO the face
- the ON-ATTACH actor is called in the target face during the time the target face is being attached from any source face. The ON-ATTACH actor is optional.

for example:

```
view [
        t1: toggle "toggle 1" attach 't2
        t2: toggle "toggle 2"
        t3: toggle "toggle 3" attach 't1
]
```

In the example above:

t1/targets	contains t2
t1/attached	contains t3
t2/attached	contains t1
t3/targets	contains t1

- the attached actions are chained by default (if the default ON-ATTACHED actor is used). So if we use the example above:

pressing t1	triggers t2 only
pressing t2	no effect
pressing t3	triggers t1 but since t1 is attached to t2, t2 is triggered as well

- if any triggered face attaches back to the initial face in the chain, such a feedback attachment is skipped and ignored
- the ON-ATTACHED actor is called in every face that is triggered in the attach reaction chain.

If the ON-ATTACHED actor returns TRUE the reaction is chained. Otherwise the reaction is terminated in such face.

The default ON-ATTACHED actor contains following code:

```
on-attached: [;arg: dst-face no-show-flag
        apply :set-face [arg/1 get-face face arg/2]
        true
]
```

Where the ARG is the source face that triggers the actual face.

.

Note

NOTES:

1. Are the face/targets, face/attached and other naming good enough or are there better names?
2. Any feedback to the chaining is appreciated.
3.

6. API function changes

Following table describes all API functions that have been changed, added or removed:

Function name	state	Description
DO-FACE	changed	Function name has been reused for the changed behaviour. Now the function executes the standard action code sequence of a face.The sequence contains the code that tries to call all attached targets of the face and then the ON-ACTION terminal actor (see below)
DO-STYLE	renamed to DO-ACTOR	DO-ACTOR now tries to call the face (overriding) actor first. If the face actor is not defined it calls style actor. DO-ACTOR/STYLE calls only the specified style actor code. It can be used to mix different actor calls.
DO-TARGETS	added	This function calls DO-FACE function on all target faces (the faces that have been attached FROM the calling face). Use DO-TARGETS/CUSTOM to execute custom code instead of the DO-FACE call on every target face.
DO-ATTACHED	changed	This function calls DO-FACE function on all faces which are attached TO the calling face. Use DO-ATTACHED/CUSTOM to execute custom code before the DO-FACE call on every attached face.

.

Note

Should be any function name changed to better one?