Notes to the %resizing-shortcuts.mdp document

Author: Ladislav Mecir, Richard Smolak

Revision History
18-Mar-2011/8:06:42+1:00 A

Table of Contents

1. Introduction 1
1.1. Simple horizontal divider bar 1
1.2. Simple box 1
1.3. Button 2
1.4. Panels 2
1.5. Text and other styles with calculated dimensions 2
2. Flex 2
2.1. Pair notation 2
2.2. Word notation 4
3. Other problems of the proposal 4
4. Summary 4

1. Introduction

Henrik had a nice idea to "throw in" three examples for comparison purposes. Let' see them:

1.1. Simple horizontal divider bar

```
bar: box [
        about: "Simple horizontal divider bar."
        facets: [
                init-size: 1x2
                min-size: 1x2
                max-size: -1x2
        ]
]
```

The code is incorrect. In our opinion, the correct code was meant to be:

```
bar: box [
        about: "Simple horizontal divider bar."
        facets: [
                init-size: 1x2
                min-size: 1x2
                max-size: as-pair 1 guie/max-coord
        ]
]
```

1.2. Simple box

Henrik wrote: "A derivative that tries not to destruct size information from the parent:"

We do not understand what is meant by "tries not to destruct…".

```
box: [
        about: "Simple box"
        facets: [
                init-size: 100x100
                max-size: guie/max-coord
        ]
]
```

The code is incorrect. In our opinion, the correct code was meant to be:

```
box: [
        about: "Simple box"
        facets: [
                init-size: 100x100
        ]
]
```

1.3. Button

Button in R3-GUI has the following dimensions:

```
init-size: 130x24
min-size: 24x24
max-size: 260x24
```

1.4. Panels

In the case of panels the INIT-SIZE, MIN-SIZE and MAX-SIZE are calculated after every content change. Thus, to allow the user to influence these, the Hint dialect was designed. As opposed to the above Flex dialect, the Hint dialect is usable to influence every of the above panel dimensions in an unrestricted way, which was found to be necessary.

1.5. Text and other styles with calculated dimensions

In case of the text style, the INIT-SIZE, MIN-SIZE and MAX-SIZE dimensions are calculated after every content change. Thus, a way how to influence the result of the calculation is needed as well for the text style and similar styles. The Hint dialect is suitable for this allowing the user to adjust any available dimension.

2. Flex

2.1. Pair notation

This notation uses some arbitrary numbers, that are not readable without documentation.

In our opinion, the users are likely to make errors in them, since the meaning is not obvious. Moreover, such errors are hard to detect, due to the non-obvious meaning.

Simple box

```
box: [
        about: "Simple box"
        facets: [
                init-size: 100x100
                flex: 1x1
        ]
]
```

Comparing this example to the code in the "Introduction" section we prefer the one in there, since it is:

- shorter
- simpler

Simple horizontal divider bar.

```
bar: box [
        about: "Simple horizontal divider bar."
        facets: [
                init-size: 1x2
                flex: 1x-1
        ]
]
```

When compared to the example in the "Introduction" section it spares one line, i.e. it is shorter, but much more cryptic at the same time.

Button

The current button style cannot be defined using the Flex dialect.

Panels

While the Hint dialect has been found necessary (see the reasons above), the introduction of an incompatible Flex dialect does not help to simplify the situation of the user, which is forced to learn a new dialect even when:

- the Flex dialect does not allow him to set all the properties demonstrated in the r3-gui/tests/panels/ examples
- the Flex dialect does not make the code more readable

Text and other styles with calculated dimensions

The same objections as for panels apply.

Layout examples

Box example

In the code example:

```
view [box options [max-size: as-pair guie/max-coord 2]]
; the number 2 is necessary knowledge. MAX-COORD is necessary knowledge.
```

, the comment is misleading. For the box style defined above the example can be:

```
view [box]
```

No Flex dialect can make this code simpler or shorter.

Second box example

In the example

```
view [box options [max-size/x: guie/max-coord/x]]
; this is not possible, currently
```

The comment is correct. This does not work, because the REDUCE/SET function is used to process the options.

However, this code works:

```
view [box options [max-size: guie/max-pair]]
```

Compared to the Flex example it is roughly of the same length, and it looks less cryptic, since:

- it does not use arbitrary numbering
- it does not require the knowledge of a new dialect

2.2. Word notation

Henrik stated: "… word notation may be easier to read, but may in this form be too easy to produce illegal char combinations."

We do agree with this objection and consider it serious.

We don’t like the word notation. Why X, Y should mean *unlimited* and W,H *restricted*… We don’t think such a dialect would make the life of a style or layout writer easier.

3. Other problems of the proposal

We are worried, that this additional layer will unnecessarily slow down the system, because if Flex is defined in the face style, the init/min/max settings would need to be recalculated on every resize although currently this recalc is not really neccessary. Such a recalculation would in fact be disastrous for the performance of the resizing algorithm for (h/v)panel as well as the resizing algorithm for (h/v)group.

4. Summary

The most serious problems of the proposed Flex dialect are, that it is incompatible with:

- the way how the dimensions of panels and groups are calculated

- the way how the dimensions of buttons are currently specified
- the way how the dimensions of text and other styles with calculated dimensions will be specified

Takíng all the above disadvantages and the comparisons of the examples into account we do not recommend to use the dialect.