Resizing for R3 GUI framework

Author: Ladislav Mecir, Richard Smolak

Revision History
13-Jan-2013/15:50:55+1:00 A

Table of Contents

1. Purpose 1
2. Overview 1
    2.1. Original version 1
    2.2. Improvements 2
3. Hints 4
    3.1. INIT-HINT, MIN-HINT and MAX-HINT 4
    3.2. The 'AUTO hint value 4
    3.3. The 'INIT hint value 4
    3.4. The 'KEEP hint value 5
    3.5. Hint values can be specified for every coordinate 5
4. Styles and resizing 5
    4.1. Hpanel and vpanel 5
    4.2. Hgroup and vgroup styles 7
5. Box model 9
6. Additional GOB attributes 11
    6.1. Special atributes shared by layout(panel,group) faces 12
    6.2. Special group face attributes 12
    6.3. Special layout face attributes 12
    6.4. Attribute formats and units 13
7. The UPDATE-FACE function 14

1. Purpose

The purpose of this document is to describe the R3 GUI resizing subsystem.

2. Overview

2.1. Original version

The original resizing in R3 GUI had the following bugs and limitations:

- problematic offset computation
- the MAX-SIZE field influenced the expansion ratio
- box model not implemented
- low flexibility

What was needed, was to define an algorithm that would:

- be flexible
- use a built-in box model
- easy to use by layout makers
- easy to understand by style makers
- simple consistent rules to compute positions
- simple consistent rules to compute sizes

2.2. Improvements

Objects expand evenly

Since the idea to use the MAX-SIZE field to determine the expansion ratio was not accepted well, we needed to use a different approach, looking more "natural" to the user.

The most natural expansion principle seemed to be the principle to expand objects "evenly", i.e. proportionally to their original dimensions. The attribute used is called INIT-SIZE.

Even expansion example

Let’s consider two objects with no resizing limits, object A having INIT-SIZE 50x20, object B having INIT-SIZE 100x20. Moreover, let’s suppose that the objects are in a hpanel consisting of just one row and containing no additional space.

Initially, both objects will have the dimensions 50x20 and 100x20, and the hpanel will have the dimensions 150x20. If it is needed to expand the hpanel to 300x20, the object A will be resized to 100x20, while the object B will be resized to 200x20, i.e. proportionally to their INIT-SIZE values.

MIN-SIZE is used as the lowest boundary

For every object it is possible to specify the smallest dimensions the object is allowed to have after resizing. The attribute is called MIN-SIZE. When the size computed by the resizing algorithm would be smaller than the MIN-SIZE, the calculated size is clipped to the MIN-SIZE.

0x0 used as "no limit to shrink"

If we do not want to limit the object’s ability to shrink, we can set the MIN-SIZE of the object to 0x0. That does not limit the object’s ability to shrink.

MAX-SIZE is used as the highest boundary

For every object it is possible to specify the greatest dimensions the object is allowed to have after resizing. The attribute is called MAX-SIZE. When the size computed by the resizing algorithm would be greater than the MAX-SIZE, the calculated size is clipped to the MAX-SIZE.

GUIE/MAX-PAIR used as "no limit to expand"

If we do not want to limit the object's ability to expand, we can set the MAX-SIZE of the object to GUIE/MAX-PAIR value, which is large enough to not limit the object's ability to expand.

Notes

- The initial size is the size that is suggested as the size when the object is first displayed. Due to the dynamic nature of resizing, the actual dimensions of the object may differ.
    - For example, if the object is in a vpanel, and the column width of the vpanel is larger than the suggested initial size, the resizing algorithm automatically tries to magnify the object so that its width becomes as large as the column width if possible.
    - Another reason, why the object size might differ from its INIT-SIZE may be that the object dimensions are clipped to either its MIN-SIZE, or MAX-SIZE
- There is not much sense (although it is not forbidden) to having the initial size equal to zero, since the object with zero initial size is invisible, and it cannot be enlarged to a nonzero size.
- To specify that an object has a fixed size it suffices to specify that its minimal as well as maximal sizes are equal to its initial size.

Hints are used to specify the dimensions of layouts

To specify the way, how the dimensions of layouts, layout rows and columns are calculated, it is possible to use hints.

Box model is used to specify margins, etc.

Every graphic object has a content area and optional surrounding padding, border, and margin areas.

Object visibility can be specified

For every object in a layout it is possible to specify that the object is:

- VISIBLE, meaning that the object is:
    - visible,
    - and it normally resizes with the layout
- HIDDEN, meaning that the object is
    - invisible,
    - and it normally resizes with the layout reserving the necessary empty space
- FIXED, meaning that the objects is:
    - visible,

    - and it does not resize with the layout,
        - i.e. it changes neither its size,
        - nor its position, when the layout resizes
- IGNORED, meaning that the objects is:
    - invisible,
    - and it does not take up any space in the layout

3. Hints

In a hpanel, vpanel, hgroup, or vgroup the initial, minimal, and maximal sizes are computed using the dimensions of the graphic elements the layout contains and the hints user has given.

3.1. INIT-HINT, MIN-HINT and MAX-HINT

These attributes are the hints used to calculate the layout dimensions, the INIT-HINT is used to calculate the INIT-SIZE, the MIN-HINT is used to calculate the MIN-SIZE, and the MAX-HINT is used to calculate the MAX-SIZE.

3.2. The 'AUTO hint value

Since the contents of a layout can be examined, it is possible to calculate the INIT-SIZE, MIN-SIZE and MAX-SIZE values for a layout based on its contents.

For example, if we set the INIT-HINT layout attribute to 'AUTO, we are suggesting that the INIT-SIZE of the layout shall be calculated automatically, from the dimensions of the layout contents.

This is the default value, i.e., if not specified otherwise by the user, the layout dimensions will be calculated automatically from dimensions of the layout contents.

3.3. The 'INIT hint value

For the MAX-SIZE and MIN-SIZE, the default hint is the 'AUTO hint as well, i.e. the values are calculated automatically from the contents of the layout.

When calculating the MIN-SIZE, the 'INIT hint can be used, if the user does not want the layout to become smaller than its INIT-SIZE is.

Similarly, when calculating the MAX-SIZE, the 'INIT hint can be used, if the user does not want the layout to become larger than its INIT-SIZE is.

Example of a non-resizing layout

To specify that the layout does not resize at all, it is sufficient to use the 'INIT hint both for the layout MIN and MAX sizes. This way, the layout size cannot change.

3.4. The 'KEEP hint value

When used e.g. as the MAX-HINT, this value specifies that the MAX-SIZE of the layout should be kept as it is now, i.e. not recalculated.

3.5. Hint values can be specified for every coordinate

For every coordinate it is possible to specify a separate hint. For example, the INIT-HINT = 100x200 value specifies that the INIT-SIZE of the layout should always be 100x200. Another way, how to specify the same is to set the INIT-HINT to [100 200]. In the latter case, any of the numbers can be replaced by one of the above word hints.

4. Styles and resizing

Every style can have its own resizing. Currently, there are three resizing methods available:

1. the simplest resizing method is the method adjusting the size of the object respecting the box model properties of the graphic object
2. the method used in the hpanel and vpanel styles adjusts the positions of subobjects so that they are organized into both rows and columns
3. the resizing method defined for the hgroup and vgroup styles uses a layout, in which the subobjects are organized into either rows, or columns, but not both at the same time

4.1. Hpanel and vpanel

In a hpanel or vpanel, the graphic objects are arranged into both rows and columns at the same time. As opposed to the group layout, layout resize always respects both rows as well as columns, producing a "perfectly tabular" layout.

LAYOUT-MODE

Similarly as for hgroup/vgroup, the hpanel and vpanel differ by their LAYOUT-MODE attribute, which can be changed at run time. This attribute is the main difference between a hpanel and vpanel.

When the LAYOUT-MODE is set to HORIZONTAL (for hpanel), the primary orientation is left-to-right, rows are layed out in top-down manner; when the LAYOUT-MODE is set to VERTICAL (for hpanel), the primary orientation is top-down, columns are layed in left-to-right manner.

The LAYOUT-MODE can be changed at run-time, changing effectively a hpanel to vpanel or vice versa.

Row and column resizing

Initial, minimal and maximal sizes of all rows and columns are computed from initial, minimal and maximal sizes of their graphic elements.

Hints

Similarly as for layouts, the init, min, and max sizes of rows and columns are calculated using hints. The hints used are ROW-INIT - used to calculate the init height of row, ROW-MIN - used to calculate the min height of row, ROW-MAX - used to calculate the max height of a row, COLUMN-INIT - used to calculate the init width of a column, COLUMN-MIN used to calculate the min width of a column, and COLUMN-MAX - used to calculate the max width of a column.

The 'MAX hint value

When used as the value of the ROW-INIT hint, it suggests that all initial heights shall be calculated as the maxima of the initial heights of their contents.

The 'MIN hint value

When used as the value of the ROW-MAX hint, it suggests that all the max heights of the rows shall be calculated as the minima of the max heights of their contents.

The 'KEEP hint value

When used as the value of the ROW-MIN hint, it suggests that all the min heights of the rows shall be kept "as is".

A number used as a hint value

When a number such as 100 is used as the COLUMN-INIT hint, it suggests that the initial widths of all columns shall be set to 100.

A block used as a hint value

It is possible to specify a hint for every column individually, by specifying a block containing one of the above hints for every layout column. (or row)

Size

- When a layout is resized, its rows as well as its columns are resized proportionally, i.e. so that their new dimensions have the same ratios as their initial sizes.
- If, using the above rule, the size of a row/column would exceed its maximum, the respective row/column is magnified to its maximal size, and the "unused magnification" is used by other rows/columns proportionally.
- If, using the above rule, the size of a row/column would become smaller than its minimum, the row/column is minified to its minimal size, and the "unused minification" is used by other rows/columns proportionally.

Positions of graphic objects

For every object, the resizing function computes the "layout cell" that contains it, which is an intersection of object’s row with object’s column. In the "layout cell", the horizontal position of

the object is chosen in accordance with object’s ALIGN attribute, the vertical position is chosen in accordance with object’s VALIGN attribute.

Position of the pane

Respecting the PANE-ALIGN and PANE-VALIGN attributes, the whole "tabular layout" is positioned in the layout viewport. For example, if the PANE-ALIGN is left and PANE-VALIGN is top, the top left point of the layout is positioned to the top left point of the layout viewport, similarly for other cases.

4.2. Hgroup and vgroup styles

LAYOUT-MODE

In a group, the graphic objects are primarily arranged depending on its LAYOUT-MODE attribute. The LAYOUT-MODE attribute can have two possible values: HORIZONTAL and VERTICAL.

If the LAYOUT-MODE is HORIZONTAL, the graphic object in the hgroup are arranged into rows in a left to right manner. Rows are positioned in a top-down manner.

If the LAYOUT-MODE is VERTICAL, the graphic object in the vgroup are arranged into columns in a top to down manner. Columns are positioned in a left-to-right manner.

By changing the LAYOUT-MODE of a hgroup/vgroup it is possible to change the way how the subobjects are arranged, which actually transforms a hgroup into a vgroup and vice versa.

Longitudinal positions of graphic objects

Every graphic object in a row (or column, depending on the LAYOUT-MODE) is supposed to lay next to another, so, if we eventually want to have a "gap" between two subsequent graphic objects, we may need to insert a "filler object" "occupying" the "empty space" between the objects.

The resizing algorithm works so that for a pair of graphic objects laying next to each other holds that they will remain to lay next to each other after any resizing operation.

Transverzal positions of graphic objects

If the LAYOUT-MODE is HORIZONTAL, the height of a row is the maximum of the heights of all graphic objects in the row.

Analogically, if the LAYOUT-MODE is VERTICAL, the width of a column is the maximum of the widths of all graphic objects in the column.

If the LAYOUT-MODE is HORIZONTAL, the height of a row can be adjusted by inserting a "flexible" graphic object into the row (which may even be invisible), allowing the row to grow more, than "ordinary graphic objects" would allow. This suggestion works analogically columns, if the LAYOUT-MODE is VERTICAL.

To compute the transverzal position of a graphic object, its alignment attributes are taken into account.

If the LAYOUT-MODE is HORIZONTAL, the the VALIGN attribute is used to align the object relative to its row as follows:

top	the top edge of the graphic object is aligned to the top edge of the row
middle	the vertical center of the graphic object is aligned to the vertical center of the row
bottom	the bottom edge of the graphic object is aligned to the bottom edge of the row

If the LAYOUT-MODE is VERTICAL, the the ALIGN attribute is used to align the object relative to its column as follows:

left	the left edge of the graphic object is aligned to the left edge of the column
center	the horizontal center of the graphic object is aligned to the horizontal center of the column
right	the right edge of the graphic object is aligned to the right edge of the column

Longitudinal positions of rows/columns

If the LAYOUT-MODE is HORIZONTAL, the row ALIGN attribute specifies the horizontal alignment of the row relative to the group viewport. Three alignment alternatives are available:

left	the left edge of the row (i.e. the left edge of the first graphic object in the row) is aligned to the left edge of the group viewport
center	the horizontal center of the row is aligned to the horizontal center of the group viewport
right	the right edge of the row (i.e. the right edge of the last graphic object in the row) is aligned to the right edge of the group viewport

If the LAYOUT-MODE is HORIZONTAL, the PANE-ALIGN attribute is used to set the ALIGN attributes of all rows in the group. In case the PANE-ALIGN attribute is a word, all group rows get the same ALIGN attribute as specified by PANE-ALIGN. If the PANE-ALIGN attribute is a block, every row obtains its own ALIGN attribute from the PANE-ALIGN block.

If the LAYOUT-MODE is VERTICAL, the VALIGN attribute specifies the vertical alignment of the column relative to the group viewport. Three alignment alternatives are available:

top	the top edge of the first column (i.e. the top edge of the first graphic object in the column) is aligned to the top edge of the layout viewport
middle	the vertical center of the column is aligned to the vertical center of the layout viewport
bottom	the bottom edge of the column (i.e. the bottom edge of the last graphic element in the column) is aligned to the bottom edge of the layout viewport

If the LAYOUT-MODE is VERTICAL, the PANE-VALIGN attribute is used to set the VALIGN attributes of all columns in the group. In case the PANE-VALIGN attribute is a word, all group columns get the same VALIGN attribute as specified by PANE-VALIGN. If the PANE-VALIGN attribute is a block, every column obtains its own VALIGN attribute from the PANE-VALIGN block.

Transversal positions of rows/columns

The resizing algorithm is working so that if two rows/columns lay next to each other, they will remain to lay next to each other (without any gap) after any resizing operation.

To specify the vertical position of all group rows, three alignment alternatives using the PANE-VALIGN attribute of the group are available:

top	the top edge of the first row is aligned to the top edge of the group viewport
middle	the vertical center of the central row is aligned to the vertical center of the group viewport
bottom	the bottom edge of the last row is aligned to the bottom edge of the group viewport

To specify the horizontal position of all group columns, three alignment alternatives using the PANE-ALIGN attribute of the group are available:

left	the left edge of the first column is aligned to the left edge of the layout viewport
center	the horizontal center of the central column is aligned to the horizontal center of the layout viewport
right	the right edge of the last column is aligned to the right edge of the layout viewport

Size

- When a group is resized, the graphic objects it contains are resized proportionally, i.e., so that their new dimensions have the same ratios as their initial sizes.
- If, using the above rule, the size of a graphic object would exceed its maximum, the object is magnified to its maximal size, and the "unused magnification" is used by other graphic objects proportionally.
- If, using the above rule, the size of a graphic object would become smaller than its minimum, the object is minified to its minimal size, and the "unused minification" is used by other graphic objects proportionally.

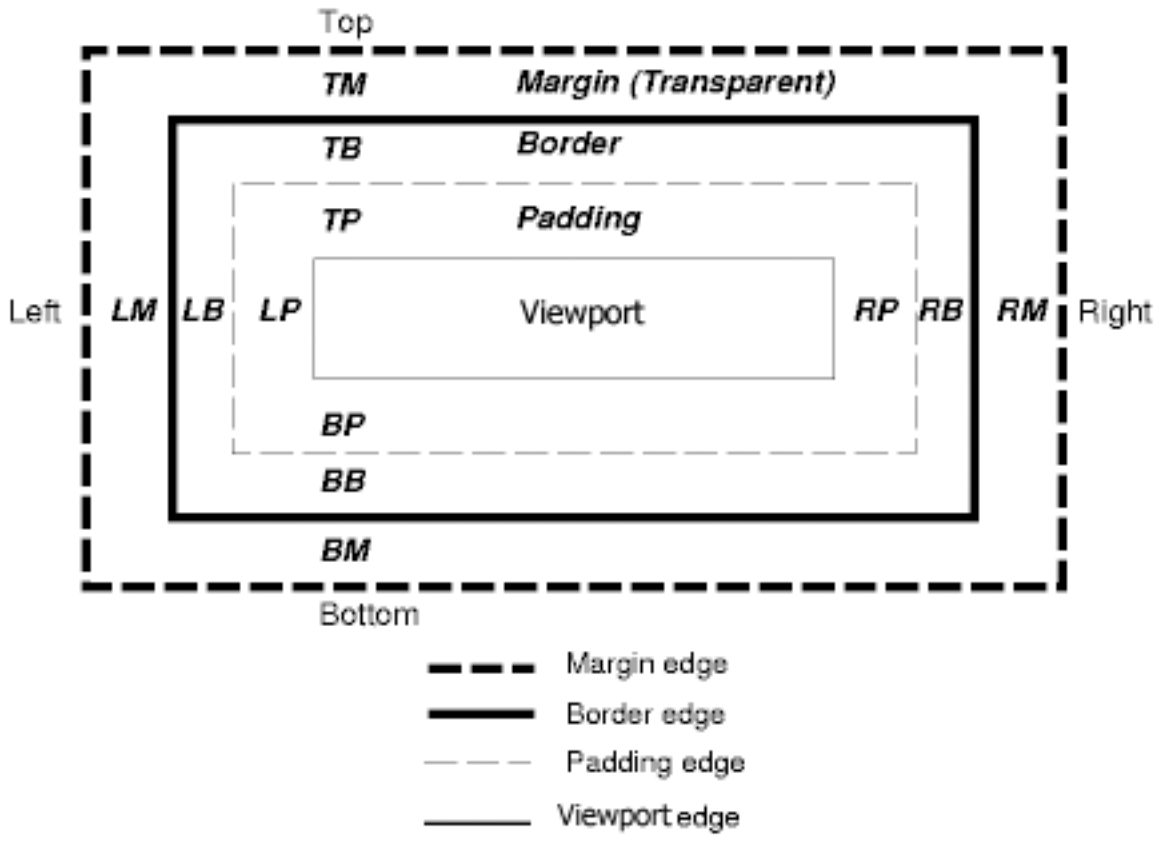
Hints

Similarly as for (h/v)panels, it is possible to specify hints for group lines. The hints used are: LINE-INIT - used to calculate the initial transversal dimensions of lines, LINE-MIN - used to calculate the minimal transversal dimensions of lines, LINE-MAX - used to calculate the maximal transversal dimensions of line.

The same types of hint values as for layout rows/columns are available.

5. Box model

The box model used is similar to the well known CSS box model. Every GOB has a viewport(content) area and optional surrounding padding, border, and margin areas.



The margin, border, and padding can be broken down into top, right, bottom, and left segments.

The perimeter of each of the four areas (viewport, padding, border, and margin) is called an "edge", so each box has four edges:

- viewport edge*

The viewport edge surrounds the viewport area of GOB. The four viewport edges define the GOB’s viewport box.

- padding edge*

The padding edge surrounds the GOB padding. If the padding has 0 width, the padding edge is the same as the viewport edge. The four padding edges define the GOB’s padding box.

- border edge*

The border edge surrounds the GOB border. If the border has 0 width, the border edge is the same as the padding edge. The four border edges define the GOB’s border box.

- margin edge*

The margin edge surrounds the GOB margin. If the margin has 0 width, the margin edge is the same as the border edge. The four margin edges define the GOB’s margin box.

.

Note

The margin area has a transparent background, so it reveals the content under the GOB, while the background of the padding area is of the same color as the background of the GOB.

Each edge may be broken down into a top, right, bottom, and left edge. The offset/size values for each edge have to be specified in pixel units.

- GOB *viewport* area size*

The dimensions of the viewport area of a GOB can be affected by the settings of all edges. The final width and height of the GOB is always equal to the defined size of the gob.

The following equation is always true:

```
viewport-size = gob-size - margin-top-left - border-top-left - padding-top-left -
margin-bottom-right - border-bottom-right - padding-bottom-right
```

For example GOB of size 100x100 pixels with border of 1 pixel on all edges will have viewport area of size 98x98 pixels.

6. Additional GOB attributes

The resizing system needs to define a couple of additional atributes(accessible thru FACETS object) for each GOB to be able to work properly.

align	holds a word describing how the graphic object is aligned horizontally in its column
valign	holds a word describing how the graphic object is aligned vertically in its row
init-size	holds the initial size (given as size, when the GOB is created) of the GOB (default is 100x100)
min-size	defines minimum GOB size value boundary that will be used during resizing (default is 0x0, which means the same as "no limit")
max-size	defines maximum GOB size value boundary that will be used during resizing (default is guie/max-pair constant, which means the same, as "no limit")
resizes	flags if the face is handled by resizing system (default is TRUE)
border-color	defines the color of the GOB border (default is NONE which means the border color is turned off)
border-size	defines the size(s) of the GOB border (default is [0x0 0x0])
margin	defines the margin offset(s) of the GOB (default is [0x0 0x0])
padding	defines the padding offset(s) of the GOB (default is [0x0 0x0])
bg-color	defines the color of the GOB *viewport* background (default is NONE which means the background color is turned off)
space	read-only value, holds sum of margin, padding and border sizes of the GOB (default is [0x0 0x0] used internally for box model calculations)
margin-box	read-only value, a block! containing top-left, top-right, bottom-left, bottom-right and center pair!s coordinates describing the margin area of the GOB

border-box	read-only value, a block! containing top-left, top-right, bottom-left, bottom-right and center pair!s coordinates describing the border area of the GOB
padding-box	read-only value, a block! containing top-left, top-right, bottom-left, bottom-right and center pair!s coordinates describing the padding area of the GOB
viewport-box	read-only value, a block! containing top-left, top-right, bottom-left, bottom-right and center pair!s coordinates describing the viewport area of the GOB
gob	read-only value, reference to GOB object (can be used in DRAW block defeinition of a face)
gob-size	can be set in layout OPTIONS block if the face is displayed in FIXED mode, otherwise used internally by the resizing system, don’t use it.

6.1. Special atributes shared by layout(panel,group) faces

layout-mode	specifies whether the graphic objects are preferably laid in rows or in columns; if this value is changed, it is necessary to call the UPDATE-FACE function, since some low-level values need to be recalculated
spacing	the X coordinate specifies space between columns and the Y coordinate space between rows
min-hint	used to calculate the MIN-SIZE
max-hint	used to calculate the MAX-SIZE
init-hint	used to calculate the INIT-SIZE
min-size	read only value, defines minimum GOB size value boundary that will be used during resizing
max-size	read only value, defines maximum GOB size value boundary that will be used during resizing
init-size	read only value, defines the initial size of GOB

6.2. Special group face attributes

pane-align	specifies the horizontal alignment of group columns, can be changed
pane-valign	specifies the vertical alignment of group rows, can be changed
line-min	used to calculate the minimal transversal dimensions of line
line-max	used to calculate the maximal transversal dimensions of line
line-init	used to calculate the initial transversal dimensions of line

6.3. Special layout face attributes

break-after	specifies the count of columns in a vpanel, or the count of rows in a hpanel; can be changed manually, negative values are not allowed
pane-align	specifies the horizontal alignment of the layout relative to the layout viewport, can be changed manually

pane-valign	specifies the vertical alignment of the layout relative to the layout viewport, can be changed manually
row-min	used to calculate the min height of a row
row-max	used to calculate the max height of a row
row-init	used to calculate the init height of a row
column-min	used to calculate the min width of a column
column-max	used to calculate the max width of a column
column-init	used to calculate the init width of a column

6.4. Attribute formats and units

This section will describe data formats used by specific attributes.

LAYOUT-MODE

format			description	

ALIGN

format			description	

PANE-ALIGN

format			description	

VALIGN

format			description	

PANE-VALIGN

format			description	

SIZE, INIT-SIZE, MIN-SIZE, MAX-SIZE

format			description	

BREAK-AFTER

format			description	

BORDER-COLOR

format			description	

BORDER-SIZE, MARGIN, PADDING

format			description	
integer!			all edges will have the same integer value, in pixels	
pair!			top/bottom and left/right edge pairs will have the same value, in pixels	
[pair!]			only the top/left edges are set, in pixels	

DRAW

format			description	

SPACE

format			description	

7. The UPDATE-FACE function

When a face was updated by changing its dimensions, or otherwise affecting its appearance/handling by its parent layout, the UPDATE-FACE function should be called to "signal" that event to its parent.

```
update-face: funct [
        face [object!]
        /no-show
        /contents
        /contents-only
]
```

When called with the /NO-SHOW refinement, the function just sets the respective flags for the layout resizing algorithm to be able to update the layout.