Le dialecte Rich Text

Table of Contents

1. Aperçu

Rich-Text (texte enrichi) est un format de texte qui permet la mise en forme. Différents styles graphiques peuvent être appliqués au texte ou à des segments de texte.

L'API Rich-Text a trois niveaux différents, du plus simple au plus optimisé:

- En utilisant RTD (depuis VID, ou lorsqu'on construit une figure manuellement).
- Un dialecte bas niveau de style enrichi pour une chaîne de texte (pour une performance rapide).
- Des paragraphes de texte enrichi multiples sur une même figure (pour des agencements complexes).

2. Le dialecte Rich-Text de haut niveau (RTD)

Grammaire

```
nested: [ahead block! into rtd]
color:  [
    s: tuple!   (v: s/1)                            ;-- couleur sous forme de tuple
R.G.B
    | issue!    (v: hex-to-rgb s/1)                 ;-- couleur sous forme de valeur
hexadécimale #rgb ou #rrggbb
    | word!     if (tuple? attempt [v: get s/1])
]
f-args: [
    ahead block! into [integer! string! | string! integer!]
    | integer!
    | string!
]
style!: make typeset! [word! tag! tuple! path!]
style: [ahead style! [
      ['b | 'bold      | <b>] (push 'b)   [nested | rtd [/b | /bold      | </b>]] (pop
'b)
    | ['i | 'italic    | <i>] (push 'i)   [nested | rtd [/i | /italic    | </i>]] (pop
'i)
    | ['u | 'underline | <u>] (push 'u)   [nested | rtd [/u | /underline | </u>]] (pop
'u)
    | ['s | 'strike    | <s>] (push 's)   [nested | rtd [/s | /strike    | </s>]] (pop
's)
    | ['f | 'font      | <font>]
       s: f-args (push either block? s/1 [head insert copy s/1 'f][reduce ['f s/1]])
       [nested | rtd [/f | /font | </font>]]
       (pop 'f)
    | ['bg | <bg>] color (push reduce ['bg v]) [nested | rtd [/bg | </bg>]] (pop 'bg)
    | color (push-color v) opt [nested (pop-color)]
    | ahead path!
      into [
        (col: 0 insert/only mark tail stack) some [                 ;@@ implement any-
single
            (v: none)
            s: ['b | 'i | 'u | 's | word! if (tuple? attempt [v: get s/1])]
            (either v [col: col + 1 push-color v][push s/1])
        ](insert cols col)
      ]
    nested (pop-all take mark)
]]
rtd: [some [pos: style | s: [string! | char!] (append text s/1 s-idx: tail-idx?)]]
```

NOTE	La syntaxe par chemin permet de combiner différents styles pour les appliquer ensemble au bloc qui vient après le chemin. Il est permis de mixer l'utilisation de délimiteurs et de blocs pour différents styles.

Usage

Les entrées RTD sont traitées par une fonction spécifique `rtd-layout` qui renverra une figure rich-

text d'un seul élément, dans lequel le code RTD sera compilé dans une unique chaîne de texte (stockée dans la facette /text) et une description de style de bas niveau (stockée dans la facette /data).

La spécification complète de la fonction est:

```
rtd-layout func [
        "Returns a rich-text face from a RTD source code"
        spec [block!]   "RTD source code"
        /only           "Returns only [text data] facets"
        /with           "Populate an existing face object"
            face [object!] "Face object to populate"
        return: [object! block!]
]
```

Exemple:

```
rt: rtd-layout [<i> <b> "Hello" </b> <font> 24 red " Red " </font> blue "World!" </i>]

view [rich-text with [text: rt/text data: rt/data]]
```

Le code RTD peut être fourni directement au sein de VID pour la facette de texte enrichi `data`.

Exemples:

```
view compose [rich-text 200x100 data [i b "Hello" /b font 24 red " Red " /font blue
"World!" /i]]

view compose [rich-text 200x100 data [i [b ["Hello"] red font 24 [" Red "] blue
"World!"]]]

view compose [rich-text 200x100 data [i/b/u/red ["Hello" font 32 " Red " /font blue
"World!"]]]

view compose [rich-text 200x100 data [i/blue ["Hello " b/u/red [font [32 "Arial"] "Red
" /font] "World!"]]]
```

3. Dialecte de mise en forme de bas niveau

Ce dialecte décrit une liste de styles à appliquer sur la chaîne de caractères référencée par la facette /text dans une figure rich-text. L'objectif de ce dialecte est de fournir une solution pour des changements dynamiques et une retrait d'informations aussi rapides que possibles. Il correspond également bien aux APIs sous-jacentes d'accélération matérielle (il s'appuie sur Direct2D dans Windows).

Usage

La grammaire du dialecte est une simple liste de segments de texte (définis en utilisant une position de départ et une longueur combinés dans une valeur de type pair!), suivie par une liste de styles. Ainsi, la structure typique est:

```
[
    <range1> style1 style2 ...        ;-- range1: start1 x length1
    <range2> style1 style2 ...        ;-- range2: start2 x length2
    ...
]
```

Les styles peuvent se recouvrir, et les styles postérieurs ont une priorité supérieure (styles en cascade).

Les styles suivants sont supportés:

```
[
      tuple!                              ;-- couleur de texte
    | backdrop tuple!                     ;-- couleur de fond
    | bold                                ;-- police en gras
    | italic
    | underline tuple! (color)  lit-word! ('dash, 'double, 'triple) ;@@ la couleur et
le type ne sont pas encore supportés
    | strike tuple! (color) lit-word! ('wave, 'double)              ;@@ la couleur et
le type ne sont pas encore supportés
    | border tuple! (color) lit-word! ('dash, 'wave)                ;@@ pas implémenté
    | integer!                            ;-- taille de la nouvelle police
    | string!                             ;-- nom de la nouvelle police
]
```

NOTE La couleur du texte ne devrait pas suivre immédiatement `strike` ou `underline`. La couleur et le type de `strike` et `underline` modifieront les styles de leurs lignes, et non le texte. Comme ils ne sont pas encore implémentés, la spécification d'une couleur (ou d'un type) après ces mots-clés n'aura aucun effet.

4. Le type de figure Rich Text

Un nouveau type de figure rich-text supporte les fonctionnalités de texte enrichi avec l'accélération matérielle sous-jacente. La figure a deux modes pour l'affichage de texte enrichi.

4.1. Mode boîte unique

Toute la surface de la figure est utilisée pour afficher le texte enrichi, en partant du coin supérieur gauche, et en utilisant les facettes spécifiques suivantes:

- /data (block!): un bloc d'instructions de mise en forme de bas niveau à appliquer à la facette text.

- /text (string!): une chaîne de texte à afficher en utilisant la description de styles de la facette /data.

La facette Draw peut toujours être utilisée et elle sera rendue par-dessus l'affichage du texte enrichi.

Exemples:

```
view [
    rich-text with [
        text: "Hello Red World!"
        data: [1x17 0.0.255 italic 7x3 255.0.0 bold 24 underline]
    ]
]
view [
    rich-text "Hello Red World!"
    with [data: [1x17 0.0.255 italic 7x3 255.0.0 bold 24 underline]]
]
```

4.2. Mode boîtes multiples

Dans ce mode, un nombre arbitraire de zones de texte enrichi peut être affiché à l'intérieur de la même figure rich-text. Pour réaliser cela, chaque zone de texte enrichi est spécifiée en utilisant le mot-clé text dans le dialecte Draw.

Facettes spécifiques:

- /draw (block!): un bloc d'instructions text, éventuellement mélangées avec des instructions Draw normales.
- /text (none!): cette facette doit être fixée à none pour permettre ce mode.

Extension de Draw

```
text <pos> <text>

<pos>  : une valeur de type pair! indiquant le coin supérieur gauche de la boîte de
texte
<text> : une chaîne de caractères, ou un objet figure rich-text avec une description
rich-text en boîte unique.
```

Exemple:

```
view compose/deep [
    rich-text 200x200 draw [
        text 10x10 (rt1: rtd-layout ["Some^/" b "text^/" /b "here"] rt1/size: 50x80
rt1)
        text 100x90 (rt2: rtd-layout [red "Other^/" b "text^/" /b "there"] rt1/size:
50x80 rt2)
        pen gold box 90x80 160x180
    ]
]
```

5. Fonctions de retrait d'informations

Les fonctions suivantes sont fournies pour retirer des informations sur le contenu d'une figure rich-text. Ces fonctions peuvent être utilisées pour implémenter facilement:

- la navigation du curseur
- un test de contact

D'après le code de `system/words` par défaut:

```
caret-to-offset: function [
    "Given a text position, returns the corresponding coordinate relative to the top-
left of the layout box"
    face    [object!]
    pos     [integer!]
    return: [pair!]
]

offset-to-caret: function [
    "Given a coordinate, returns the corresponding text position"
    face    [object!]
    pt      [pair!]
    return: [integer!]
]

size-text: function [
    "Returns the area size of the text in a face"
    face [object!]
    /with                   ;-- inutilisé avec rich-text
        text [string!]
    return: [pair! none!]
]
```

D'après le contexte rich-text:

```
line-height?: function [
    "Given a text position, returns the corresponding line's height"
    face    [object!]
    pos     [integer!]
    return: [integer!]
]

line-count?: function [
    "number of lines (> 1 if line wrapped)"
    face    [object!]
    return: [integer!]
]
```

Exemples:

```
view [
    rich-text data [font 16 "Sélectionnez du texte avec votre souris" /font]
    on-down [
        bkg: reduce [ ; Couleur de fond du texte sélectionné
            as-pair caret: offset-to-caret face event/offset 0
            'backdrop sky
        ]
        either 2 = length? face/data [ ; A la première sélection
            pos: tail face/data
            append face/data bkg
        ][ ; Changement de la position de départ lors des sélections suivantes
            change pos bkg/1
        ]
    ] all-over
    on-over [
        if event/down? [ ; Si le bouton de souris reste pressé ne changer que la
longueur
            pos/1/2: (offset-to-caret face event/offset) - caret
        ]
    ]
]
```

```
view compose/deep [
    rich-text draw [
        text 10x10 (rt: rtd-layout [i/blue ["Hello " red/b [font 24 "Red " /font]
"World!"]])
        line-width 5 pen gold
        line ; Dessin d'une ligne sous les mots à l'aide d'un couple des fonction
utilitaires ci-dessus
            (as-pair 10 h: 10 + rich-text/line-height? rt 1) ; Starting-point y -> 10
+ line-height
            (as-pair 10 + pick size-text rt 1 h) ; End-point x -> 10 + length-of-text-
size
    ]
]
```