Dialecte d’Interfaces Visuelles

Table of Contents

1. Vue d’ensemble

VID signifie Dialecte d’Interfaces Visuelles (Visual Interface Dialect). Il a pour objet de fournir un dialecte (DSL) simple pour la description d’interfaces graphiques utilisateur en surcouche du moteur View de Red.

VID vous permet de spécifier chaque composant graphique à afficher, vous donnant le choix entre différentes méthodes d’agencement:

- succession horizontale ou verticale

- positionnement sur une grille
- positionnement absolu

VID créera automatiquement un conteneur pour vous, pour contenir les descriptions de figures que vous fournissez. Par défaut, la figure conteneur est du type `window`.

Le code de VID est traité par la fonction `layout` (qui est appelée en interne par la fonction `view`). Le code VID est ensuite compilé en une arborescence de figures, adapté à un affichage direct.

NOTE	Utilisez `help view` et `help layout` à partir de la console Red pour voir comment traiter un bloc VID.

2. Structure du code

Un bloc de code VID typique a la structure suivante:

```
[
    <paramètres du conteneur>
    <description de l'agencement>
]
```

- **paramètres du conteneur**: paramètres qui affectent l'objet conteneur - qui peut être un panneau (`panel`) ou une fenêtre (`window`).
- **description de l'agencement**: commandes de positionnement de l'agencement, définitions de styles et descriptions de figures.

NOTE	Toutes les sections sont optionnelles, il n'y a aucun contenu obligatoire à fournir dans un bloc VID.

3. Paramètres du conteneur

NOTE	le mot-clé `react` peut également être utilisé au niveau des paramètres du conteneur en sus des options de figure, voir sa description ici.

3.1. title (titre)

Syntaxe

```
title <text>

<text> : texte de titre (string!).
```

Description

Définit le texte de titre de la figure conteneur.

3.2. size (taille)

Syntaxe

```
size <value>

<value> : largeur et hauteur en pixels (pair!).
```

Description

Définit la taille de titre de la figure conteneur. Si la taille n'est pas explicitement donnée, la taille du conteneur est automatiquement calculée pour correspondre à son contenu.

3.3. backdrop (fond)

Syntaxe

```
backdrop <color>

<color> : nom ou valeur d'une couleur (word! tuple! issue!).
```

Description

Définit la couleur de fond de la figure conteneur.

3.4. Définition d'acteurs

Les acteurs du conteneur peuvent également être définis dans cette zone du code. Voir la section Acteurs pour la définition des acteurs.

4. Layout (agencement)

Par défaut, VID place les figures dans la figure conteneur suivant des règles simples:

- la direction peut être horizontale ou verticale
- les figures sont positionnées les unes après les autres dans la direction courante en utilisant l'espaçage courant.

Valeurs par défaut:

- origin: `10x10`
- space: `10x10`
- direction: `across`

- alignment: `top`

Voici comment les figures sont disposées en mode `across`:

[across] | *across.png*

Voici comment les figures sont disposées en mode `below` (en utilisant l'alignement par défaut `left`):

[below] | *below.png*

4.1. across (à côté)

Syntaxe

```
across <alignment>

<alignment> : (optionnel) valeurs possibles: top | middle | bottom.
```

Description

Définit une direction d'agencement des figures horizontale, de gauche à droite. Un modificateur d'alignement peut éventuellement être fourni pour changer l'alignement des figures de la rangée, par défaut alignées au sommet (`top`).

4.2. below (au dessous)

Syntaxe

```
below <alignment>

<alignment> : (optionnel) valeurs possibles: left | center | right.
```

Description

Définit une direction d'agencement des figures verticale, du haut vers le bas. Un modificateur d'alignement peut éventuellement être fourni pour changer l'alignement des figures de la colonne, par défaut alignées à gauche (`left`).

4.3. return

Syntaxe

```
return <alignment>

<alignment> : (optionnel) valeurs possibles: left | center | right | top | middle |
bottom.
```

Description

Déplace la position (de placement des figures) à la rangée ou colonne de figures suivante, suivant la direction d'agencement courante. Un modificateur d'alignement peut éventuellement être fourni pour changer l'alignement courant des figures dans la rangée ou dans la colonne.

4.4. space (espacement)

Syntaxe

```
space <offset>

<offset> : nouvelle valeur d'espacement (pair!).
```

Description

Définit la nouvelle valeur d'espacement qui sera utilisée pour le placement des figures suivantes.

4.5. origin

Syntaxe

```
origin <offset>

<offset> : nouvelle valeur d'origine (pair!).
```

Description

Définit la nouvelle position d'origine, relative à la figure conteneur.

4.6. at (à)

Syntaxe

```
at <offset>
at <expr>

<offset> : position de la figure suivante (pair!).
<expr>   : expression Red qui renvoie une valeur de type `pair!` utilisée comme
position.
```

Description

Place la figure suivante à une position absolue. Ce mode de positionnement n'affecte que la figure immédiatement suivante, et ne change pas la position courante du flux d'agencement. Ainsi, les figures suivant la prochaine, seront placées de nouveau dans la continuité des précédentes dans le

flux d'agencement.

4.7. pad (écartement)

Syntaxe

```
pad <offset>

<offset> : déplacement relatif (pair!).
```

Description

Modifie la position courante d'agencement d'un déplacement relatif. Toutes les figures suivantes de la même rangée (ou colonne) sont affectées.

4.8. do

Syntaxe

```
do <body>

<body> : code à évaluer (block!).
```

Description

Evalue un bloc de code Red normal, pour les besoins d'initialisation de dernière minute. Le bloc est lié à la figure conteneur (fenêtre ou panneau), donc un accès direct aux facettes du conteneur est possible. On peut faire référence au conteneur lui-même en utilisant le mot-clé `self`.

5. Styles additionnels

Le moteur View offre de nombreux widgets intégrés, le dialecte VID étend ceux-ci en définissant des styles additionnels couramment utilisés avec des mots-clés associés. Ils peuvent être utilisés avec les mêmes options que leur type de figure sous-jacente. Ils peuvent également être restylés à volonté en utilisant la commande `style`.

5.1. h1

Le style `H1` est un type `text` avec une taille de police fixée à 32.

5.2. h2

Le style `H2` est un type `text` avec une taille de police fixée à 26.

5.3. h3

Le style `H3` est un type `text` avec une taille de police fixée à 22.

5.4. h4

Le style `H4` est un type `text` avec une taille de police fixée à 17.

5.5. h5

Le style `H5` est un type `text` avec une taille de police fixée à 13.

5.6. box

Le style `box` est un type de `base` avec une couleur par défaut transparente.

5.7. image

Le style `image` est un type de `base` de taille par défaut 100x100. Il attend une option `image!`; si aucune n'est fournie, une image vide sur fond blanc, et de la même taille que la figure, est produite.

6. Définition de figures

Une figure peut être insérée dans l'agencement, à la position courante, en utilisant simplement le nom d'un type de figure existant ou de l'un des styles disponibles.

Syntaxe

```
<name>: <type> <options>

<name>    : nom optionnel pour le nouveau composant (set-word!).
<type>    : un type de figure valide ou un nom de style (word!).
<options> : liste d'options optionnelle.
```

Si un nom est fourni, le mot fera référence à l'objet `face!` créé par VID à partir de la description de figure.

Des valeurs par défaut sont fournies pour chaque type de figure ou chaque style, ainsi une nouvelle figure peut être utilisée sans qu'il soit nécessaire de spécifier aucune option. Lorsque des options sont requises, les sections suivantes décrivent les différents types d'options acceptées:

- Mots-clés
- Types de données
- Acteurs

Toutes les options peuvent être spécifiées dans un ordre arbitraire, suivant le nom de la figure ou du style. Un nouveau nom de figure ou un mot-clé d'agencement marquent la fin de la liste d'options pour une figure donnée.

NOTE `window` ne peut pas être utilisé comme type de figure.

6.1. Mots-clés

6.1.1. left

Syntaxe

```
left
```

Description

Aligne le texte de la figure sur le coté gauche.

6.1.2. center

Syntaxe

```
center
```

Description

Centre le texte de la figure.

6.1.3. right

Syntaxe

```
right
```

Description

Aligne le texte de la figure sur le coté droit.

6.1.4. top

Syntaxe

```
top
```

Description

Aligne verticalement le texte en haut de la figure.

6.1.5. middle

Syntaxe

```
middle
```

Description

Aligne verticalement le texte au centre de la figure.

6.1.6. bottom

Syntaxe

```
bottom
```

Description

Aligne verticalement le texte en bas de la figure.

6.1.7. bold

Syntaxe

```
bold
```

Description

Met en gras le texte de la figure.

6.1.8. italic

Syntaxe

```
italic
```

Description

Met en italique le texte de la figure.

6.1.9. underline

Syntaxe

```
underline
```

Description

Souligne le texte de la figure.

6.1.10. extra

Syntaxe

```
extra <expr>

<expr> : toute valeur ou expression Red (any-type!).
```

Description

Donne une valeur à la facette `extra` de la figure.

6.1.11. data

Syntaxe

```
data <list>
data <expr>

<list> : liste littérale d'éléments (block!).
<expr> : expression Red qui renvoie une liste sous forme de valeur de type `block!`
```

Description

Définit une liste de valeurs comme facette `data` de la figure. Le format de la liste dépend de ce qui est requis par ce type de figure.

6.1.12. draw

Syntaxe

```
draw <commands>
draw <expr>

<commands> : liste littérale de commandes (block!).
<expr>     : expression Red qui renvoie un `block!` de commandes.
```

Description

Donne une valeur à la facette `draw` de la figure.

Définit une liste de commandes du dialecte Draw comme facette `draw` de la figure. Voir la Documentation du dialecte Draw pour les commandes valides.

6.1.13. font

Syntaxe

```
font <spec>

<spec> : une spécification de police valide (block! object! word!).
```

Description

Définit un nouvel objet `font` (police) comme facette `font` de la figure. L'objet Font! est décrit ici.

> **NOTE** Il est possible d'utiliser `font` avec d'autre paramètres concernant les polices, VID les fusionnera ensemble, donnant la priorité au dernier spécifié.

6.1.14. para

Syntaxe

```
para <spec>

<spec> : une spécification para valide (block! object! word!).
```

Description

Définit un nouvel objet `para` (paragraphe) comme facette `para` de la figure. L'objet Para! est décrit ici.

> **NOTE** Il est possible d'utiliser `para` avec d'autre paramètres concernant les paragraphes, VID les fusionnera ensemble, donnant la priorité au dernier spécifié.

6.1.15. wrap

Syntaxe

```
wrap
```

Description

Renvoie à la ligne le texte de la figure lors de l'affichage.

6.1.16. no-wrap

Syntaxe

```
no-wrap
```

Description

Ne renvoie pas à la ligne le texte de la figure lors de l'affichage.

6.1.17. font-size

Syntaxe

```
font-size <pt>

<pt> : taille de police en points (integer! word!).
```

Description

Définit la taille de police courante pour le texte de la figure.

6.1.18. font-color

Syntaxe

```
font-color <value>

<value> : couleur de la police (tuple! word! issue!).
```

Description

Définit la couleur de police courante pour le texte de la figure.

6.1.19. font-name

Syntaxe

```
font-name <name>

<name> : nom valide d'une police disponible (string! word!).
```

Description

Définit le nom de la police courante pour le texte de la figure.

6.1.20. react

Ce mot-clé peut être utilisé à la fois comme une option de figure ou comme un mot-clé global. Un nombre arbitraire d'instances de `react` peut être utilisé.

Syntaxe

```
react [<body>]
react later [<body>]

<body> : code Red normal (block!).
```

Description

Crée un nouveau réacteur à partir du bloc fourni. Lorque `react` est utilisé comme une option de figure, le corps du bloc peut faire référence à la figure courante en utilisant le mot `face`. Lorsque `react` est utilisé globalement, les figures cibles doivent être référencées en utilisant un nom. Le mot-clé optionnel `later` ignore la première réaction se produisant immédiatement après le traitement du corps du bloc.

NOTE	Les réacteurs font partie du support de programmation réactive dans View, dont la documentation est à venir. En bref, le bloc peut décrire une ou plusieurs relations entre les propriétés des figures en utilisant des chemins. Les `set-path` définissant une propriété d'une figure sont traitées comme la **cible** du réacteur (la figure à mettre à jour), tandis que les chemins accédant à une propriété d'une figure sont traités comme la **source** du réacteur (un changement de la source déclenche une relecture du code du réacteur).

6.1.21. loose

Syntaxe

```
loose
```

Description

Permet de faire glisser la figure avec le bouton gauche de la souris.

6.1.22. all-over

Syntaxe

```
all-over
```

Description

Lève le drapeau `all-over` permettant la réception de tous les événements `over` de la souris.

6.1.23. hidden

Syntaxe

```
hidden
```

Description

Rend la figure invisible par défaut.

6.1.24. disabled

Syntaxe

```
disabled
```

Description

Désactive la figure par défaut (la figure ne traitera aucun événement jusqu'à ce qu'elle soit activée).

6.1.25. password

Syntaxe

```
password
```

Description

Cache l'entrée de l'utilisateur dans un champ de texte.

6.1.26. tri-state

Syntaxe

```
tri-state
```

Description

Active le mode à 3 états d'une case à cocher.

6.1.27. select

Syntaxe

```
select <index>

<index> : index de l'élément sélectionné (integer!).
```

Description

Définit la facette `selected` de la figure courante. Utilisé principalement avec des listes pour indiquer quel élément est présélectionné.

6.1.28. focus

Syntaxe

```
focus
```

Description

Donne le focus à la figure courante lorsque la fenêtre est affichée pour la première fois. Une seule figure peut avoir le focus. Si plusieurs options `focus` sont utilisées sur des figures différentes, uniquement la dernière obtiendra le focus.

6.1.29. hint

Syntaxe

```
hint <message>

<message> : texte d'aide (string!).
```

Description

Fournit un message d'aide dans les figures `field` (champ), Lorsque le contenu du champ est vide. Ce texte disparaît lorsque un nouveau contenu est fourni (par action de l'utilisateur ou définition de la valeur de la facette `face/text`).

6.1.30. rate

Syntaxe

```
rate <value>
rate <value> now

<value>: durée ou fréquence (integer! time!).
```

Description

Définit un minuteur pour la figure, à partir d'une durée (time!) ou d'une fréquence (integer!). A chaque déclenchement du minuteur, un événement `time` sera généré pour cette figure. Si l'option `now` est utilisé, un premier événement est généré immédiatement.

6.1.31. default

Syntaxe

```
default <value>

<value>: une valeur par défaut pour la facette `data` (any-type!).
```

Description

Définit une valeur par défaut pour la facette `data` lorsque la conversion de la facette `text` renvoie `none`. Cette valeur par défaut est stockée dans la facette `options` en tant que paire clé/valeur.

NOTE	n'est actuellement utilisé que pour les figures de type `text` et `field`.

6.1.32. with

Syntaxe

```
with <body>

<body>: un bloc de code Red lié à la figure courante (block!).
```

Description

Evalue un bloc de code Red lié à la figure en cours de définition. Cela permet une définition directe des champs de la figure, en passant outre les autres options de VID.

6.2. Types de données

Additionnellement aux mots-clés, il est permis de passer des paramètres aux figures en utilisant des valeurs littérales des types suivants:

Type de données	Usage
integer!	Spécifie la largeur de la figure. Pour les panneaux, indique le nombre de rangées ou de colonnes pour l'agencement, suivant la direction courante.
pair!	Spécifie la largeur et la hauteur de la figure.
tuple!	Spécifie la couleur de fond de la figure (lorsque c'est applicable).
issue!	Spécifie la couleur de fond de la figure en utilisant la notation hexadécimale (#rgb, #rrggbb, #rrggbbaa).

Type de données	Usage
string!	Spécifie le texte à afficher par la figure.
date!	Définit la facette `data` (utile pour le type `calendar`)
percent!	Définit la facette `data` (utile pour les types `progress` et `slider`).
logic!	Définit la facette `data` (utile pour les types `check` et `radio`).
image!	Définit l'image à afficher comme fond de la figure (lorsque c'est applicable).
url!	Charge la ressource pointée par l'URL, puis traite la ressource en fonction du type chargé.
block!	Définit l'action pour l'événement par défaut de la figure. Pour les panneaux, définit leur contenu.
get-word!	Utilise une fonction existante comme acteur.
char!	*(réservé pour un usage futur).*

6.3. Acteurs

Un acteur peut être attaché à une figure en spécifiant une valeur de bloc littérale ou le nom d'un acteur suivi par une valeur de bloc.

Syntaxe

```
<actor>
on-<event> <actor>

<actor> : bloc du corps de l'acteur ou référence à l'acteur (block! get-word!).
<event> : nom d'événement valide (word!).
```

Description

Il est possible de spécifier des acteurs d'une manière simplifiée en fournissant juste le bloc du corps de l'acteur, le bloc de spécifications étant implicite. La fonction acteur est alors construite et ajoutée à la facette `actor` de la figure. Plusieurs acteurs peuvent être spécifiés de cette manière.

La spécification complète de la fonction acteur créée est:

```
func [face [object!] event [event! none!]][...body...]
```

La liste des noms d'événements valides est disponible ici.

Lorsqu'un bloc ou un get-word est passé sans aucun nom d'acteur en préfixe, l'acteur par défaut pour ce type de figure est créé, suivant les définitions données ici.

7. Panneaux

Il est possible de définir des panneaux enfants pour regrouper des figures, et éventuellement appliquer des styles particuliers. La taille du nouveau panneau, si elle n'est pas spécifiée explicitement, est calculée automatiquement pour s'adapter au contenu.

Les types de figures de la classe panneau de View sont supportés en VID avec une syntaxe particulière:

7.1. panel

Syntaxe

```
panel <options> [<content>]

<options> : liste optionnelle de paramètres pour chaque panneau.
<content> : description VID du contenu du panneau (block!).
```

Description

Construit un panneau enfant à l'intérieur du conteneur courant, dont le contenu est un autre bloc VID. Additionnellement aux autres options de figure, un diviseur entier peut être fourni, définissant un agencement en mode grille:

- si la direction est `across`, le diviseur représente le nombre de colonnes.
- si la direction est `below`, le diviseur représente le nombre de rangées.

7.2. group-box (boîte de groupement)

Syntaxe

```
group-box <divider> <options> [<body>]

<divider> : nombre optionnel de rangées ou de colonnes (integer!).
<options> : liste optionnelle de paramètres pour le panneau.
<body>    : description en VID du contenu du panneau (block!).
```

Description

Construit une un panneau enfant "boîte de groupement" à l'intérieur du conteneur courant, lorsque le contenu est un autre bloc VID. Un diviseur peut être donné en argument, pour instaurer un agencement en mode grille:

- si la direction est `across`, le diviseur représente le nombre de colonnes.
- si la direction est `below`, le diviseur représente le nombre de rangées.

NOTE	On peut donner une valeur de `string!` en option qui définira le texte de titre de la boîte de groupement.

7.3. tab-panel (panneau à onglets)

Syntaxe

```
tab-panel <options> [<name> <body>...]

<options> : liste optionnelle de paramètres pour le panneau.
<name>    : le titre d'un onglet (string!).
<body>    : le contenu d'un onglet sous forme de description VID (block!).
```

Description

Construit une un panneau à onglets à l'intérieur du conteneur courant. Le bloc de spécifications doit contenir un couple nom/description pour chaque onglet. Le contenu du corps de chaque onglet est une nouvelle figure panneau enfant, se comportant comme n'importe quel autre panneau.

8. Mise en forme

8.1. style

Syntaxe

```
style <new> <old> <options>

<new>     : nom du nouveau style (set-word!).
<old>     : nom de l'ancien style (word!).
<options> : liste optionnelle de paramètres pour le nouveau style.
```

Description

Définit un nouveau style dans le panneau courant. Le nouveau style peut être créé à partir de types de figures existants, ou à partir d'autres styles. Le nouveau style est valide uniquement dans le panneau courant et ses panneaux enfants.

Des styles peuvent être définis en cascade des panneaux parents vers les panneaux enfants, si bien que le même style peut être redéfini ou étendu dans les panneaux enfants, sans affecter les définitions dans les panneaux parents.

9. Métrique d’interface graphique multi-plateforme

Afin de gérer les différentes règles générales d’interface graphique qui varient selon les plateformes, VID inclut un moteur de réécriture de l’interface graphique orienté-règles qui est capable de modifier dynamiquement une arborescence de figures en fonction de règles préétablies. Ce moteur est intégré au dernier stade du traitement VID.

Règles de Windows:

- color-backgrounds (coloriage de fond): colorie le fond de certaines figures incolores pour correspondre à la couleur de leur parent
- color-tabpanel-children (coloriage des enfants d’un panneau à onglets): comme color-backgrounds, mais spécifique aux panneaux à onglets
- OK-Cancel (OK - Annule): ordre des boutons, qui place les boutons Cancel/Delete/Remove en dernier

Règles de macOS:

- adjust-buttons (ajuste les boutons): utilise les sous-classes de boutons standard lorsque les boutons sont suffisamment étroits
- capitalize (majuscule): Met une majuscule au début du texte du widget text suivant les règles générales de macOS
- Cancel-OK (Annule - OK): ordre des boutons, qui place les boutons Ok/Save/Apply en dernier

---

Un exemple simple, qui met en jeu les règles d’ordre des boutons et de capitalisation:

```
    view [
        text "Name" right 50 field return
        text "Age"  right 50 field return
        button "ok" button "cancel"
    ]
```

Remarquez le texte et l’ordre des boutons sur les formulaires générés pour macOS et Windows.

[mac]

[windows]

Les règles d’interface graphique ont garanti que:

- Les boutons sont ordonnés suivant les règles générales de chaque plateforme, "Ok" en dernier sur macOS, "Cancel" en dernier sur Windows.

- Les labels des boutons commencent bien par une majuscule sur macOS.

Vous pouvez désactiver les règles d'interface graphique en fixant `system/view/VID/GUI-rules/active?` à `no`.

```
system/view/VID/GUI-rules/active?: no
```

Vous pouvez également supprimer des règles sélectivement, en modifiant le contenu des listes suivantes:

```
system/view/VID/GUI-rules/OS/Windows
== [
    color-backgrounds
    color-tabpanel-children
    OK-Cancel
]
```

```
system/view/VID/GUI-rules/OS/macOS
== [
    adjust-buttons
    capitalize
    Cancel-OK
]
```

Cela vous permet un contrôle total lorsque c'est nécessaire, mais vous aide aussi à suivre les règles générales sans effort.