Le moteur graphique Red/View

Table of Contents

1. Objectifs de conception . . . 2
2. Objet figure . . . 2
   2.1. La facette options . . . 4
3. Objet police . . . 5
4. Objet paragraphe . . . 6
5. L'arbre des figures . . . 6
6. Types de figures . . . 7
   6.1. Base . . . 7
   6.2. Text (texte) . . . 7
   6.3. Button (bouton) . . . 8
   6.4. Check (case à cocher) . . . 8
   6.5. Radio (bouton de radio) . . . 8
   6.6. Field (champ de saisie) . . . 9
   6.7. Area (champ multiligne) . . . 10
   6.8. Text-list (liste de textes) . . . 11
   6.9. Drop-list (liste déroulante) . . . 11
   6.10. Drop-down (liste déroulante éditable) . . . 12
   6.11. Calendar . . . 13
   6.12. Progress (barre de progression) . . . 13
   6.13. Slider (curseur) . . . 13
   6.14. Camera (caméra) . . . 14
   6.15. Panel (panneau) . . . 14
   6.16. Tab-panel (panneau à onglets) . . . 14
   6.17. Window (fenêtre) . . . 15
   6.18. Screen (écran) . . . 16
   6.19. Group-box (boîte de groupement) . . . 16
7. Cycle de vie d'une figure . . . 16
8. la fonction SHOW . . . 17
9. Mise à jour en temps réel vs différée . . . 18
10. Liaison bidirectionnelle . . . 18
11. Evénements . . . 19
   11.1. Noms des événements . . . 19
   11.2. Le type de données Event! . . . 20
   11.3. Acteurs . . . 22
   11.4. Parcours de l'événement . . . 23
   11.5. Gestionnaires globaux d'événements . . . 24

11.5.1. insert-event-func . . . 24
11.5.2. remove-event-func . . . 24
12. L'objet System/view . . . 25
13. Inclusion du composant View . . . 25
14. Fonctions supplémentaires . . . 26

1. Objectifs de conception

Le composant Red/View (ou simplement View) est un système graphique pour le langage de programmation Red. Les objectifs de conception sont:

- Orienté-données, API minimale
- Arbre virtuel d'objects comme interface de programmation
- Synchronisation en temps réel ou différée entre l'arbre virtuel et le système d'affichage
- Une liaison bidirectionnelle très facile à supporter
- La capacité d'avoir des processus de fond différents, pour des plateformes différentes
- Le support de jeux de widgets fournis par le système opératoire, par des tiers ou personnalisés
- Un faible impact sur les performances

L'arbre virtuel est construit en utilisant des objets figures. Chaque objet figure correspond à un composant graphique de l'affichage dans une liaison bidirectionnelle.

2. Objet figure

Les objets figures sont des clones de l'objet modèle `face!`. Un champ d'un objet figure est appelé une *facette* (facet).

Liste des facettes disponibles:

Facette	Type de données	Obligatoire?	Applicable à	Description
type	word!	oui	tous	Type du composant graphique (voir ci-dessous).
offset	pair!	oui	tous	Déplacement de la position à partir du coin supérieur gauche du parent.
size	pair!	oui	tous	Taille de la figure.
text	string!	non	tous	Texte d'étiquette affiché dans la figure.
image	image!	non	certains	Image affichée comme fond de la figure.

Facette	Type de données	Obligatoire?	Applicable à	Description
color	tuple!	non	certains	Couleur de fond de la figure au format R.G.B ou R.G.B.A.
menu	block!	non	tous	Barre de menu ou menu contextuel.
data	any-type!	non	tous	Données contenues par la figure.
enabled?	logic!	oui	tous	Active ou désactive le traitement des événements par la figure.
visible?	logic!	oui	tous	Affiche ou cache la figure.
selected	integer!, pair!, object!	non	certains	Pour les types liste, index de l'élément actuellement sélectionné. Pour les champs d'entrée de texte, le texte sélectionné. Pour les fenêtres, la figure qui a le focus.
flags	block!, word!	non	certains	Liste de mot-clés spéciaux qui altèrent l'affichage ou le comportement de la figure.
options	block!	non	certains	Propriétés supplémentaires de la figure au format [name: value].
parent	object!	non	tous	Référence arrière vers la figure parent (s'il y en a une).
pane	block!	non	certains	Liste des figures enfant affichées dans la figure.
state	block	non	tous	Information d'état interne de la figure *(utilisé uniquement par le moteur View)*.
rate	integer!, time!	non	tous	Minuteur de la figure. Génère périodiquement des événements `time`. Un entier définit une fréquence, un horaire (`time`) définit un intervalle, `none` arrête le minuteur.
edge	object!	non	tous	*(réservé pour un usage futur)*
para	object!	non	tous	Référence à un objet para pour le positionnement du texte.

Facette	Type de données	Obligatoire?	Applicable à	Description
font	object!	non	tous	Référence à un objet font pour fixer les propriétés de la police de la facette texte.
actors	object!	non	tous	Gestionnaires d'événements fournis par l'utilisateur.
extra	any-type!	non	tous	Données utilisateur optionnelles attachées à la figure (usage libre).
draw	block!	non	tous	Liste de commandes Draw à exécuter pour dessiner sur la figure.

Liste des drapeaux utilisables globalement pour la facette `flags`:

Drapeau	Description
all-over	Envoie tous les événements `over` à la figure.

D'autres drapeaux spécifiques aux types de figures sont documentés dans leurs sections respectives.

NOTE

- Les facettes non obligatoires peuvent être fixées à `none`.
- `offset` et `size` sont définis en pixels d'écran.
- `offset` et `size` peuvent parfois être fixés à `none` avant l'affichage. Le moteur View se chargera de fixer les valeurs (comme pour les panneaux dans le type panneau à onglets).
- Ordre d'affichage (de l'arrière vers l'avant): color, image, text, draw.

La création d'une nouvelle figure se fait par clonage de l'objet `face!` en fournissant **au moins** un nom de `type` valide.

```
button: make face! [type: 'button]
```

Une fois qu'une figure est créée, le champ `type` ne peut plus être changé.

2.1. La facette options

La facette options regroupe des facettes optionnelles qui sont utilisées pour des comportements spécifiques:

Option	Description
drag-on	Peut être l'une des valeurs suivantes: `'down`, `'mid-down`, `'alt-down`, `'aux-down`. Utilisé pour permettre une opération de glisser-déposer.

3. Objet police

Les objets police de caractères sont des clones de l'objet modèle `font!`. Un objet police peut être référencé par une ou plusieurs figures, ce qui permet de contrôler les propriétés de police d'un groupe de figures de manière centralisée.

Champ	Type de données	Obligatoire?	Description
name	string!	non	Nom d'une police valide installée sur le système d'exploitation.
size	integer!	non	Taille de police en points.
style	word!, block!	non	Mode de mise en forme ou bloc de modes de mise en forme.
angle	integer!	oui	Angle d'écriture du texte en degrés (la valeur par défaut est `0`).
color	tuple!	oui	Couleur de la police au format R.G.B ou R.G.B.A.
anti-alias?	logic!, word!	non	Mode anti-crénelage (actif/inactif ou mode spécial).
shadow	*(réservé)*	non	*(réservé pour un usage futur)*
state	block!	non	Information sur l'état interne de la figure *(utilisé uniquement par le moteur View)*.
parent	block!	non	Référence arrière interne à la/les figure(s) parent(s) *(utilisé uniquement par le moteur View)*.

NOTE

- Les facettes non obligatoires peuvent être fixées à `none`.
- Le champ `angle` ne fonctionne pas encore correctement.
- Les valeurs de tous les champs devraient devenir optionnelles à l'avenir.

Styles de polices disponibles:

- `bold`
- `italic`
- `underline`
- `strike`

Modes anti-crénelage disponibles:

- actif/inactif (`anti-alias?: yes/no`)
- mode ClearType (`anti-alias?: 'ClearType`)

4. Objet paragraphe

Les objets paragraphe sont des clones de l'objet modèle `para!`. Un objet paragraphe peut être référencé par une ou plusieurs figures, ce qui permet de contrôler les propriétés paragraphe d'un groupe de figures de manière centralisée.

Champ	Type de données	Description
origin	*(réservé)*	*(réservé pour un usage futur)*
padding	*(réservé)*	*(réservé pour un usage futur)*
scroll	*(réservé)*	*(réservé pour un usage futur)*
align	word!	Contrôle l'alignement horizontal du texte: `left`, `center`, `right`.
v-align	*(réservé)*	Contrôle l'alignemente vertical du texte: `top`, `middle`, `bottom`.
wrap?	logic!	Active/désactive le retour à la ligne automatique du texte dans la(les) figure(s).
parent	block!	Référence arrière interne à la(les) figure(s) parent(s) *(utilisé uniquement par le moteur View)*.

NOTE

- Tous les champs du paragraphe peuvent être mis à `none`.

5. L'arbre des figures

Les figures sont organisées en un arbre qui correspond aux composants graphiques de l'affichage. Les relations dans l'arbre sont définies à partir de:

- la facette `pane`: liste d'une ou plusieurs figures enfant dans un bloc.
- la facette `parent`: référence à la figure parent.

L'ordre des objets figure dans un `pane` est important, il détermine l'ordre de superposition des objets graphiques (la figure en tête du `pane` est affichée au-dessous de toutes les autres figures, la dernière figure est affichée au-dessus de toutes les autres).

La racine d'un arbre de figures est une figure `screen`. Une figure `screen` ne peut afficher que des figures `window` à partir de son bloc `pane`.

Pour qu'une figure quelle qu'elle soit puisse être affichée à l'écran, elle *doit* être connectée à une figure `screen` directement (pour les fenêtres) ou indirectement (pour les autres types de figures).

[Arbre des figures] | *face-tree.png*

6. Types de figures

6.1. Base

Le type `base` est le type de figure le plus basique, mais aussi le plus versatile. Par défaut, il n'affichera qu'un fond de couleur `128.128.128`.

Facette	Description
`type`	`'base`
`image`	Une valeur de type `image!` peut être spécifiée, le canal alpha est supporté.
`color`	Une couleur de fond peut être spécifiée, le canal alpha est supporté.
`text`	Un texte optionnel à afficher dans la figure.
`draw`	La transparence est totalement supportée dans les primitives Draw.

NOTE

- Toutes les facettes suivantes peuvent être combinées et seront rendues dans l'ordre suivant: `color`, `image`, `text`, `draw`.
- La transparence peut être obtenue dans `color`, `image`, `text` et `draw` en spécifiant une composante de canal alpha dans les t-uplets de couleurs: `R.G.B.A` où `A = 0` indique une opacité totale et `A = 255` une transparence totale.

Ce type de face devrait être utilisé pour toute implémentation de composant graphique personnalisé.

6.2. Text (texte)

Le type `text` est un label statique à afficher.

Facette	Description
`type`	`'text`
`text`	Texte du label.
`data`	Valeur à afficher sous forme de texte.
`options`	Champs supportés: `default`.

La facette `data` est synchronisée en temps réel avec la facette `text` en utilisant les règles de conversion suivantes:

- lorsque `text` change, `data` prend la valeur de `load` appliqué à `text`, ou `none`, ou `options/default` si celui-ci est défini.
- lorsque `data` change, `text` prend la valeur de `form` appliqué à `data`.

La facette `options` accepte les propriétés suivantes:

- `default`: peut prendre n'importe quelle valeur, qui sera utilisée par la facette `data` si la conversion de `text` retourne `none`, comme dans le cas des chaînes de caractères ne pouvant être

chargées par `load`.

6.3. Button (bouton)

Ce type représente un bouton simple.

Facette	Description
`type`	`'button`
`text`	Le texte du bouton.
`image`	L'image sera affichée dans le bouton. Peut être combiné avec un texte.

Type d'événement	Gestionnaire	Description
`click`	`on-click`	Déclenché lorsque l'utilisateur clique sur un bouton.

6.4. Check (case à cocher)

Ce type représente une case à cocher, avec un texte de label optionnel, affiché du côté gauche ou du côté droit.

Facette	Description
`type`	`'check`
`text`	Texte du label.
`para`	Le champ `align` contrôle si le texte est affiché du côté gauche ou du côté droit.
`data`	`true`: coché; `false`: décoché; `none`: décoché pour une case à cocher à 2 états, indéterminé pour une case à cocher à 3 états (par défaut);
`flags`	Active la fonctionnalité de case à cocher à 3 états (`word!`).

Drapeaux supportés:

- `tri-state`: active le troisième état, état indéterminé représenté par la valeur `none` dans la facette `data`.

Type d'événement	Gestionnaire	Description
`change`	`on-change`	Déclenché lorsque l'état de la case à cocher est changé par une action de l'utilisateur.

6.5. Radio (bouton de radio)

Ce type représente un bouton de radio, avec un texte de label optionnel, affiché du côté gauche ou du côté droit. Un seul bouton de radio par panneau peut être coché.

Facette	Description
`type`	`'radio`
`text`	Texte du label.
`para`	Le champ `align` contrôle si le texte est affiché du côté gauche ou du côté droit.
`data`	`true`: coché; `false`: décoché (par défaut).

Type d'événement	Gestionnaire	Description
`change`	`on-change`	Déclenché lorsque l'état du bouton de radio est changé par une action de l'utilisateur.

6.6. Field (champ de saisie)

Ce type représente un champ de saisie sur une seule ligne.

Facette	Description
`type`	`'field`
`text`	Texte saisi; valeur en lecture/écriture.
`data`	Valeur à afficher comme texte.
`selected`	Texte sélectionné (pair! none!).
`options`	Champs supportés: `default`.
`flags`	Active/désactive certaines propriétés spéciales du champ de saisie (block!).

La facette `selected` contrôle le surlignage du texte (en lecture/écriture). Une valeur de type pair indique le premier et le dernier des caractères surlignés. Une valeur `none` indique qu'aucun texte n'est sélectionné dans le champ de saisie.

Drapeaux supportés:

- `no-border` : supprime les décorations de bordure faites par le système sous-jacent d'interface graphique.
- `password` : au lieu des caractères tapés, des astérisques (*) sont affichées.

La facette `data` est synchronisée en temps réel avec la facette `text` en utilisant les règles de conversion suivantes:

- lorsque `text` change, `data` prend la valeur de `load` appliqué à `text`, ou `none`, ou `options/default` si celui-ci est défini.
- lorsque `data` change, `text` prend la valeur de `form` appliqué à `data`.

La facette `options` accepte les propriétés suivantes:

- `default`: peut prendre n'importe quelle valeur, qui sera utilisée par la facette `data` si la conversion de `text` retourne `none`, comme dans le cas des chaînes de caractères ne pouvant être

chargées par `load`.

Type d'événement	Gestionnaire	Description
`enter`	`on-enter`	Se produit chaque fois que la touche Entrée est pressée dans le champ de saisie.
`change`	`on-change`	Se produit chaque fois qu'une saisie est faite dans le champ de saisie.
`select`	`on-select`	Se produit chaque fois qu'un texte a été sélectionné en utilisant la souris ou le clavier.
`key`	`on-key`	Se produit chaque fois qu'une touche est pressée dans le champ de saisie.

6.7. Area (champ multiligne)

Ce type représente un champ de saisie multiligne.

Facette	Description
`type`	`'area`
`text`	Texte saisi; valeur en lecture/écriture.
`selected`	Texte sélectionné (pair! none!).
`flags`	Active/désactive certaines propriétés spéciales du champ de saisie (block!).

La facette `selected` contrôle le surlignage du texte (en lecture/écriture). Une valeur de type pair indique le premier et le dernier des caractères surlignés. Une valeur `none` indique qu'aucun texte n'est sélectionné dans le champ de saisie.

Drapeaux supportés:

- `no-border`: supprime les décoration de bordure faites par le système sous-jacent d'interface graphique.

NOTE

- Une barre de défilement verticale peut apparaître si toutes les lignes de texte ne peuvent pas être visibles dans le champ multiligne (cela pourrait être contrôlé par une option `flags` dans le futur).

Type d'événement	Gestionnaire	Description
`change`	`on-change`	Se produit chaque fois qu'une saisie est faite dans le champ multiligne.
`select`	`on-select`	Se produit chaque fois qu'un texte a été sélectionné en utilisant la souris ou le clavier.
`key`	`on-key`	Se produit chaque fois qu'une touche est pressée dans le champ multiligne.

6.8. Text-list (liste de textes)

Ce type représente une liste verticale de chaînes de texte, affichée dans un cadre fixe. Une barre de défilement verticale apparaît automatiquement si le contenu ne rentre pas dans le cadre.

Facette	Description
`type`	`'text-list`
`data`	Liste de chaînes à afficher (`block! hash!`).
`selected`	Index de la chaîne sélectionnée ou valeur none s'il n'y a aucune sélection (lecture/écriture).

Type d'événement	Gestionnaire	Description
`select`	`on-select`	Se produit lorsqu'une entrée de la liste est sélectionnée. La facette `selected` contient l'index de **l'ancienne** entrée sélectionnée.
`change`	`on-change`	Se produit après un événement `select`. La facette `selected` contient l'index de la **nouvelle** entrée sélectionnée.

NOTE

- Le nombre d'entrées visibles ne peut pas encore être défini par l'utilisateur.

6.9. Drop-list (liste déroulante)

Ce type représente une liste verticale de chaînes de texte, affichée dans un cadre repliable. Une barre de défilement verticale apparaît automatiquement si le contenu ne rentre pas dans le cadre.

Facette	Description
`type`	`'drop-list`
`data`	Liste de chaînes à afficher (`block! hash!`).
`selected`	Index de la chaîne sélectionnée ou valeur `none` s'il n'y a aucune sélection (lecture/écriture).

La facette `data` accepte des valeurs arbitraires, mais seulement les valeurs de type chaîne seront ajoutées à la liste et affichées. Des valeurs supplémentaires de type de données autre que chaîne peuvent être utilisées pour créer des tableaux associatifs, en utilisant les chaînes comme clés. La facette `selected` est un index entier en base 10 indiquant la position de la chaîne sélectionnée dans la liste, et non pas dans la facette `data`.

Drapeaux supportés:

PAS ENCORE IMPLÉMENTÉ

- `scrollable`: active manuellement une barre de défilement verticale.

Type d'événement	Gestionnaire	Description
`select`	`on-select`	Se produit lorsqu'une entrée de la liste est sélectionnée. La facette `selected` contient l'index de **l'ancienne** entrée sélectionnée.
`change`	`on-change`	Se produit après un événement `select`. La facette `selected` contient l'index de la **nouvelle** entrée sélectionnée.

NOTE

- Le nombre d'entrées visibles ne peut pas encore être défini par l'utilisateur.

6.10. Drop-down (liste déroulante éditable)

Ce type représente un champ éditable avec une liste verticale de chaînes de texte, affichés dans un cadre repliable. Une barre de défilement verticale apparaît automatiquement si le contenu ne rentre pas dans le cadre.

Facette	Description
`type`	`'drop-down`
`data`	Liste de chaînes à afficher (`block! hash!`).
`selected`	Index de la chaîne sélectionnée ou valeur `none` s'il n'y a aucune sélection (lecture/écriture).

La facette `data` accepte des valeurs arbitraires, mais seulement les valeurs de type chaîne seront ajoutées à la liste et affichées. Des valeurs supplémentaires de type de données autre que chaîne peuvent être utilisées pour créer des tableaux associatifs, en utilisant les chaînes comme clés. La facette `selected` est un index entier en base 10 indiquant la position de la chaîne sélectionnée dans la liste, et non pas dans la facette `data`.

Drapeaux supportés:

PAS ENCORE IMPLÉMENTÉ

- `scrollable`: active manuellement une barre de défilement verticale.

Type d'événement	Gestionnaire	Description
`select`	`on-select`	Se produit lorsqu'une entrée de la liste est sélectionnée. La facette `selected` contient l'index de **l'ancienne** entrée sélectionnée.
`change`	`on-change`	Se produit après un événement `select`. La facette `selected` contient l'index de la **nouvelle** entrée sélectionnée.

NOTE

- Le nombre d'entrées visibles ne peut pas encore être défini par l'utilisateur.

6.11. Calendar

Ce type représente un calendrier Grégorien mensuel couvrant l'intervalle du 1er janvier 1601 au 31 décembre 9999.

Facette	Description
`type`	`'calendar`
`data`	valeur de type `date!` qui représente le jour sélectionné.

Type d'événement	Gestionnaire	Description
`change`	`on-change`	Se produit lorsqu'une date est sélectionnée dans le calendrier.

NOTE

- By default, la facette `data` est initialisée à la date du jour.
- Une valeur de `date!` inférieure ou supérieure aux limites spécifiées du calendrier sélectionnera respectivement la date minimale ou maximale supportée.

6.12. Progress (barre de progression)

Ce type représente une barre de progression horizontale ou verticale.

Facette	Description
`type`	`'progress`
`data`	Valeur représentant la progression (valeur de type `percent!` ou `float!`).

NOTE

- Si une valeur de type `float` est utilisée pour `data`, celle-ci doit être entre 0.0 et 1.0.

6.13. Slider (curseur)

Ce type représente un curseur qui peut être déplacé selon un axe horizontal ou vertical.

Facette	Description
`type`	`'slider`
`data`	Valeur représentant la position du curseur (valeur de type `percent!` ou `float!`).

NOTE

- Si une valeur de type `float` est utilisée pour `data`, celle-ci doit être entre 0.0 et 1.0.

6.14. Camera (caméra)

Ce type est utilisé pour afficher le flux d'une caméra vidéo.

Facette	Description
`type`	`'camera`
`data`	Liste de nom(s) de caméra(s) sous forme d'un bloc de chaînes de caractères.
`selected`	Sélectionne la caméra à afficher dans la liste `data`, en utilisant un index entier. Si fixé à `none`, le flux de caméra est désactivé.

NOTE

- La facette `data` est initialement à `none`. La liste de caméras est consultée durant le premier appel à `show` sur la figure caméra.
- Il est possible de capturer le contenu d'une figure caméra en utilisant `to-image` sur la figure.

6.15. Panel (panneau)

Un panel est un conteneur pour d'autres figures.

Facette	Description
`type`	`'panel`
`pane`	Bloc de figures enfants. L'ordre dans le bloc définit l'ordre de superposition lors de l'affichage.

NOTE

- Les coordonnées d' `offset` des enfants sont relatives au coin supérieur gauche du panneau parent.
- Les faces enfants sont coupées aux limites du cadre du panneau.

6.16. Tab-panel (panneau à onglets)

Un tab-panel est une liste de panneaux dont un seul peut être visible à la fois. Une liste de noms de panneaux est affichée sous forme d'onglets, et utilisée pour basculer entre les panneaux.

Facette	Description
`type`	`'tab-panel`
`data`	Bloc de noms de panneaux (valeurs chaînes de caractères).
`pane`	Liste de panneaux correspondant à la liste d'onglets (`block!`).
`selected`	Index du panneau sélectionné ou valeur `none` (`integer!`) (lecture/écriture).

Type d'événement	Gestionnaire	Description
`change`	on-change	Se produit lorsque l'utilisateur sélectionne un nouvel onglet. `event/picked` contient un index de l'onglet nouvellement sélectionné. La propriété `selected` est mise à jour juste après cet événement.

NOTE

- Les deux facettes `data` et `pane` doivent être remplies afin que le panneau à onglets puisse être affiché correctement.
- Si `pane` contient plus de panneaux que le nombre spécifié d'onglets, ils seront ignorés.
- Lors de l'ajout/la suppression d'un onglet, le panneau correspondant doit être ajouté à/supprimé de la liste `pane`.

6.17. Window (fenêtre)

Représente une fenêtre affichée sur le bureau du système d'exploitation.

Facette	Description
`type`	`'window`
`text`	Titre de la fenêtre (`string!`).
`offset`	Déplacement à partir du coin supérieur gauche de l'écran du bureau, sans prendre en compte les décorations de bordure de la fenêtre. (`pair!`)
`size`	Taille de la fenêtre, sans prendre en compte les décorations de bordure de la fenêtre. (pair!)
`flags`	Active/désactive certaines propriétés spéciales de la fenêtre (`block!`).
`menu`	Affiche une barre de menu dans la fenêtre (`block!`).
`pane`	Liste de figures à afficher dans la fenêtre (`block!`).
`selected`	Sélectionne la fenêtre qui recevra le focus (`object!`).

Drapeaux supportés:

- `modal`: rend la fenêtre modale, désactivant toutes les fenêtres précédemment ouvertes.
- `resize`: active le redimensionnement de la fenêtre (par défaut elle est de taille fixée, non redimensionnable).
- `no-title`: n'affiche pas de titre à la fenêtre.
- `no-border`: supprime les décorations de bordure de la fenêtre.
- `no-min`: supprime le bouton de minimisation de la barre d'en-tête de la fenêtre.
- `no-max`: supprime le bouton de maximisation de la barre d'en-tête de la fenêtre.
- `no-buttons`: supprime tous les boutons de la barre d'en-tête de la fenêtre.

- `popup`: décoration de bordure alternative plus petite (Windows seulement).

NOTE	• L'emploi du mot-clé `popup` au début du bloc de spécification du menu forcera un menu contextuel dans la fenêtre, au lieu d'une barre de menu par défaut.

6.18. Screen (écran)

Représente une unité d'affichage graphique connectée à l'ordinateur (habituellement un moniteur).

Facette	Description
`type`	`'screen`
`size`	Taille d'affichage de l'écran en pixels. Défini par le moteur de View au démarrage (`pair!`).
`pane`	Liste de fenêtres à afficher sur l'écran (`block!`).

Toutes les figures "fenêtre" qui sont affichées doivent être des enfants d'une figure "écran".

6.19. Group-box (boîte de groupement)

Une boîte de groupement est un conteneur pour d'autres figures, entouré d'une bordure visible. *Il s'agit d'un style temporaire qui sera supprimé lorsque la facette* `edge` *sera supportée.*

Facette	Description
`type`	`'group-box`
`pane`	Bloc de figures enfants. L'ordre dans le bloc définit l'ordre de superposition lors de l'affichage.

NOTE	• Les coordonnées `offset` des enfants sont relatives au coin supérieur gauche de la boîte de groupement. • Les faces enfant sont coupées aux limites du cadre de la boîte de groupement.

7. Cycle de vie d'une figure

1. Création d'un objet figure à partir du prototype `face!`.
2. Insertion de l'objet figure dans un arbre connecté à une figure écran.
3. Utilisation de `show` pour rendre la figure sur l'écran.
   a. des ressources système sont allouées à ce moment.
   b. le bloc `face/state` est défini.
4. Suppression de la figure de la liste `pane` pour la supprimer de l'affichage.
5. Le nettoyeur (garbage collector) se charge de libérer les ressources système associées lorsque la figure n'est plus référencée.

NOTE

- Une fonction `free` pourra être fournie pour un contrôle manuel de la libération des ressources systèmes dans les applications gourmandes en ressources.

8. la fonction SHOW

Syntaxe

```
show <face>

<face>: clone de l'objet face! ou bloc d'objets figures ou de noms de figures
(utilisant des valeurs de type word!).
```

Description

Cette fonction est utilisée pour mettre à jour une figure ou une liste de figures à l'écran. Seule une figure qui est référencée dans un arbre de figures connecté à un écran peut être correctement rendue à l'écran. Lors du premier appel de la fonction, des ressources système seront allouées, la facette `state` sera définie et le composant graphique sera affiché à l'écran. Les appels ultérieurs reflèteront à l'écran tout changement fait à l'objet figure. Si la facette `pane` est définie `show` sera également appliqué récursivement aux figures enfants.

La facette State

L'information qui suit est fournie uniquement pour référence, en usage normal la facette `state` ne devrait pas être manipulée par l'utilisateur. Cependant, on peut y accéder si des API du système d'exploitation sont appelées directement par l'utilisateur ou si le comportement du moteur de View doit être modifié.

Position/Field	Description
1 (handle)	Pointeur spécifique au système d'exploitation vers l'objet graphique (`integer!`).
2 (changes)	Tableau de drapeaux booléens qui indiquent quelles facettes ont été changées depuis le dernier appel à `show` (`integer!`).
3 (deferred)	Liste de changements différés depuis le dernier appel à `show`; lorsque les mises à jour en temps réel sont désactivées (`block! none!`).
4 (drag-offset)	Enregistre la position de départ du curseur de la souris lorsqu'on entre en mode de déplacement d'une figure à la souris (`pair! none!`).

NOTE

- Après un appel à `show`, le champ `changes` est remis à 0 et le bloc de champs `deferred` est vidé.
- Un type de données `handle!` sera utilisé dans le futur pour des pointeurs de système d'exploitation opaques.

9. Mise à jour en temps réel vs différée

Le moteur View a deux modes différents pour mettre à jour l'affichage après que des changements aient été faits à l'arbre des figures:

- Mise à jour en temps réel: tout changement à une figure est immédiatement rendu à l'écran.
- Mise à jour différée: tous les changements à une figure ne sont pas propagés à l'écran, jusqu'à ce que `show` soit appelé sur la figure ou sur la figure parent.

Le basculement entre ces deux modes est contrôlé par le mot `system/view/auto-sync?` : s'il prend la valeur `yes`, le mode de mise à jour en temps réel (mode par défaut) est activé; s'il prend la valeur `no`, le moteur View différera toutes les mises à jour.

Les motivations pour une mise à jour en temps réel par défaut sont:

- Un code source plus simple et plus court, pas besoin d'appeler `show` après chaque changement de face.
- Moindre difficulté d'apprentissage pour les débutants.
- Assez bon pour des applications simples ou des prototypes.
- Simpllifie l'expérimentation depuis la console.

Le mode différé met à jour beaucoup de changements en même temps à l'écran afin d'éviter les ralentissements ou lorsque l'objectif est une meilleure performance.

NOTE	• C'est une grande différence avec le moteur de Rebol/View qui ne supporte que le mode différé.

10. Liaison bidirectionnelle

Les objets figure s'appuient sur le système de propriété de Red pour lier l'objet avec la série utilisée dans les facettes, de telle sorte que tout changement dans l'une des facettes (même un changement profond) est détecté par l'objet figure et traité suivant le mode de synchronisation courant (temps réel ou différé).

D'un autre côté, les changements faits aux objets graphiques affichés sont instantanément reflétés sur la figure correspondante. Par exemple, taper du texte dans une figure `field` reflètera immédiatement l'entrée sur la facette `text` de la figure.

Cette liaison bidirectionnelle simplifie l'interaction avec les objets graphiques pour le programmeur, sans avoir besoin d'aucune API spécifique. Il suffit de modifier les facettes en utilisant les actions sur les séries.

Exemple:

```
view [
    list: text-list data ["John" "Bob" "Alice"]
    button "Add" [append list/data "Sue"]
    button "Change" [lowercase pick list/data list/selected]
]
```

11. Evénements

11.1. Noms des événements

Nom	Type d'entrée	Cause
down	souris	Le bouton gauche de la souris a été pressé.
up	souris	Le bouton gauche de la souris a été relâché.
mid-down	souris	Le bouton du milieu de la souris a été pressé.
mid-up	souris	Le bouton du milieu de la souris a été relâché.
alt-down	souris	Le bouton droit de la souris a été pressé.
alt-up	souris	Le bouton droit de la souris a été relâché.
aux-down	souris	Le bouton auxiliaire de la souris a été pressé.
aux-up	souris	Le bouton auxiliaire de la souris a été relâché.
drag-start	souris	Démarrage d'un déplacement de figure à la souris.
drag	souris	Une figure est en cours de déplacement à la souris.
drop	souris	Une figure déplacée à la souris a été relâchée.
click	souris	Clic gauche de la souris (widgets boutons seulement).
dbl-click	souris	Double clic gauche de la souris.
over	souris	Le curseur de la souris passe sur une figure. Cet événement est produit une fois lorsque la souris entre sur la figure, et une fois lorsqu'elle la quitte. Si la facette `flags` contient le drapeau **all-over**, alors tous les événements intermédiaires sont également produits.
move	souris	Une fenêtre a été déplacée.
resize	souris	Une fenêtre a été redimensionnée.
moving	souris	Une fenêtre est en cours de déplacement.
resizing	souris	Une fenêtre est en cours de redimensionnement.
wheel	souris	La roue de la souris est tournée.
zoom	tactile	Un geste de zoomage (pincement) a été reconnu.
pan	tactile	Un geste de déplacement (balayage) a été reconnu.
rotate	tactile	Un geste de rotation a été reconnu.

Nom	Type d'entrée	Cause
two-tap	tactile	Un geste de double tapotement a été reconnu.
press-tap	tactile	Un geste de pression-et-tapotement a été reconnu.
key-down	clavier	Une touche est pressée.
key	clavier	Un caractère a été entré ou une touche spéciale a été pressée (à l'exception des touches control, majuscule et menu).
key-up	clavier	Une touche pressée est relâchée.
enter	clavier	La touche Entrée est pressée.
focus	any	Une figure vient de recevoir le focus.
unfocus	any	Une figure vient de perdre le focus.
select	any	Une sélection est faite sur une figure offrant des choix multiples.
change	any	Un changement est intervenu sur une figure acceptant des entrées utilisateur (saisie de texte ou sélection dans une liste).
menu	any	Une entrée de menu est choisie.
close	any	Une fenêtre se ferme.
time	timer	Le délai fixé par la facette `rate` de la figure a expiré.

NOTE

- Les événements de type tactile ne sont pas disponibles dans Windows XP.
- Un ou plusieurs événements `moving` précèdent toujours un événement `move`.
- Un ou plusieurs événements `resizing` précèdent toujours un événement `resize`.

11.2. Le type de données Event!

Une valeur du type event est un objet opaque contenant toute l'information sur un événement donné. On accède aux champs de l'événement en utilisant la notation par chemin.

Champ	Valeur retournée
`type`	Type d'événement (`word!`).
`face`	Objet figure sur lequel l'événement s'est produit (`object!`).
`window`	Objet fenêtre sur lequel l'événement s'est produit (`object!`).
`offset`	Coordonnées du curseur de la souris par rapport à l'objet figure lorsque l'événement s'est produit (`pair!`). Pour les événements tactiles, retourne les coordonnées du point central.
`key`	Touche pressée (`char! word!`).

Champ	Valeur retournée
`picked`	Nouvel élément sélectionné dans une figure (`integer! percent!`). Pour un événement de souris `down` sur une `text-list`, retourne l'élément sous la souris ou `none`. Pour un événement `wheel`, retourne le nombre de pas de rotation. Une valeur positive indique que la roue a été tournée vers l'avant, en s'éloignant de l'utilisateur; une valeur négative indique que la roue a été tournée vers l'arrière, vers l'utilisateur. Pour un événement `menu`, retourne l'ID (`word!`) du menu correspondant. Pour un geste de zoomage, retourne une valeur de pourcentage représentant l'augmentation/la diminution relative. Pour les autres gestes, la valeur dépend du système pour l'instant (Windows: `ullArguments`, champ de GESTUREINFO).
`flags`	Retourne une liste d'un ou plusieurs drapeaux (voir liste ci-dessous) (`block!`).
`away?`	Retourne `true` si le curseur de la souris quitte les limites de la figure (`logic!`). Ne s'applique que si l'événement `over` est actif.
`down?`	Retourne `true` si le bouton gauche de la souris a été pressé (`logic!`).
`mid-down?`	Retourne `true` si le bouton du milieu de la souris a été pressé (`logic!`).
`alt-down?`	Retourne `true` si le bouton droit de la souris a été pressé (`logic!`).
`ctrl?`	Retourne `true` si la touche CTRL a été pressée (`logic!`).
`shift?`	Retourne `true` si la touche MAJ a été pressée (`logic!`).

Liste des drapeaux possibles dans `event/flags`:

- `away`
- `down`
- `mid-down`
- `alt-down`
- `aux-down`
- `control`
- `shift`

NOTE

- Tous les champs (à l'exception de `type`) sont en lecture seule. La valeur de `type` n'est changée qu'en interne par le moteur View.

Voici la liste des touches spéciales retournées sous forme de mots par `event/key`:

- `page-up`
- `page-down`
- `end`
- `home`
- `left`
- `up`

- right
- down
- insert
- delete
- F1
- F2
- F3
- F4
- F5
- F6
- F7
- F8
- F9
- F10
- F11
- F12

Les noms supplémentaires suivants peuvent être retournés par event/key uniquement dans le cas des messages key-down et key-up:

- left-control
- right-control
- left-shift
- right-shift
- left-menu
- right-menu

11.3. Acteurs

Les acteurs sont des fonctions de gestion des événements de View. Ils sont définis dans un objet de forme libre (aucun prototype n'est fourni) référencé par la facette actors. Tous les acteurs ont le même bloc de spécifications.

Syntaxe

```
on-<event>: func [face [object!] event [event!]]

<event> : tout nom d'événement valide (de la table ci-dessus)
face    : objet figure qui reçoit l'événement
event   : valeur de l'événement.
```

Additionnellement aux événements de l'interface graphique, il est possible de définir un acteur `on-create` qui sera appelé lorsque la figure est affichée pour la première fois, juste avant que des ressources système ne lui soient allouées. Contrairement aux autres acteurs, `on-create` n'a qu'un argument, `face`.

Valeur retournée

```
'stop : quitte la boucle d'événements.
'done : interrompt le passage de l'événement à la figure suivante.
```

Les autres valeurs retournées n'ont pas d'effet.

11.4. Parcours de l'événement

Les événements sont habituellement générés à une position spécifique de l'écran et assignés à la figure supérieure la plus proche. Cependant, l'événement voyage d'une figure à une autre dans la hiérarchie des parents, dans deux directions communément appelées:

- **capture** de l'événement: l'événement va de la figure fenêtre vers la figure d'où il provient. Pour chaque figure, un événement `detect` est généré et le gestionnaire correspondant est appelé s'il en a été fourni un.
- **remontée** de l'événement: l'événement va de la figure vers la fenêtre parent. Pour chaque figure, le gestionnaire d'événement local est appelé.

[Event flow] | *event-flow.png*

Chemin typique parcouru par un événement:

1. Un événement de click est généré sur un bouton, les gestionnaires globaux sont traités (voir section suivante).
2. L'étape de capture de l'événement démarre:
   a. La fenêtre reçoit l'événement en premier, son gestionnaire `on-detect` est appelé.
   b. Le panneau reçoit ensuite l'événement. Le gestionnaire `on-detect` du panneau est appelé.
   c. Le bouton reçoit l'événement en dernier. Le gestionnaire `on-detect` du bouton est appelé.
3. L'étape de remontée de l'événement démarre:
   a. Le bouton reçoit l'événement en premier, son gestionnaire `on-click` est appelé.
   b. Le panneau reçoit ensuite l'événement. Le gestionnaire `on-click` du panneau est appelé.
   c. La fenêtre reçoit l'événement en dernier, son gestionnaire `on-click` est appelé.

NOTE

- L'annulation d'un événement est obtenue en retournant le mot `'done` depuis n'importe quel gestionnaire d'événement.
- La capture des événements n'est pas activée par défaut pour des raisons de performance. Définir `system/view/capturing?: yes` pour l'activer.

11.5. Gestionnaires globaux d’événements

Avant l’entrée dans le parcours d’événement, un pré-traitement spécifique peut être accompli en utilisant ce qu’on appelle les "gestionnaires globaux d’événements". L’API suivante est fournie pour les ajouter et les supprimer.

11.5.1. insert-event-func

Syntaxe

```
insert-event-func <handler>

<handler> : une fonction gestionnaire ou un bloc de code pour le pré-traitement de(s)
(l')événement(s).

Spécification de la fonction gestionnaire: func [face [object!] event [event!]]
```

Valeur de retour

```
La fonction gestionnaire nouvellement ajoutée (`function!`).
```

Description

Installe une fonction gestionnaire global d’événements, qui peut pré-traiter les événements avant qu’ils n’atteignent les gestionnaires des figures. Tous les gestionnaires globaux sont appelés sur chaque événement, aussi le corps du gestionnaire doit être optimisé pour la rapidité et l’usage mémoire. Si un bloc est passé en argument, il sera converti en une fonction en utilisant le constructeur `function`.

Valeur de retour de la fonction gestionnaire:

- `none` : l’événement peut être traité par d’autres gestionnaires (`none!`).
- `'done` : les autres gestionnaires globaux sont ignorés mais l’événement est propagé aux figures enfants (`word!`).
- `'stop` : quitte la boucle d’événements (`word!`).

Une référence à la fonction gestionnaire est renvoyée et devrait être sauvegardée si la fonction doit être supprimée plus tard.

11.5.2. remove-event-func

Syntaxe

```
remove-event-func <handler>

<handler> : une fonction gestionnaire d'événements prédédemment installée.
```

Description

Désactive un gestionnaire d'événements global précédemment installé en le retirant de la liste interne.

12. L'objet System/view

Word	Description
`screens`	Liste de figures écrans représentant les affichages connectés.
`event-port`	*réservé pour un usage futur*
`metrics`	*réservé pour un usage futur*
`platform`	Code bas-niveau du moteur View de la plateforme (inclut du code backend).
`VID`	code de traitement de VID.
`handlers`	Liste des gestionnaires d'événements globaux.
`reactors`	Tableau associatif interne pour les figures réactives et leurs blocs d'actions.
`evt-names`	Tableau interne pour la conversion des événements en noms d'acteurs.
`init`	Fonction d'initialisation du moteur View, peut être appelée par l'utilisateur si nécessaire.
`awake`	Fonction - principal point d'entrée de haut niveau pour les événements.
`capturing?`	`yes` = permet l'étape de capture des événements et la génération d'événements `detect` (`no` par défaut).
`auto-sync?`	`yes` = mises à jour en temps réel des figures (par défaut), `no` = mise à jour différée des figures.
`debug?`	`yes` = affiche des logs détaillés des événements internes de View (`no` par défaut).
`silent?`	`yes` = ne signale pas les erreurs de traitement des dialectes VID ou Draw (`no` par défaut).

13. Inclusion du composant View

Le composant View n'est pas inclus par défaut à la **compilation**. Pour l'inclure, le script Red principal doit déclarer la dépendance dans l'en-tête en utilisant le champ `Needs`:

```
Red [
    Needs: 'View
]
```

NOTE L'usage des consoles automatiquement générées par l'exécutable `red` incluera le composant View sur les plateformes sur lesquelles il est disponible, l'en-tête `Needs` n'est donc pas requis dans les scripts utilisateur exécutés depuis ces consoles.

14. Fonctions supplémentaires

Fonction	Description
view	Rend une fenêtre à l'écran à partir d'un arbre de figures ou d'un bloc de code VID. Entre dans une boucle d'événements sauf si le **raffinement** `/no-wait` est employé.
unview	Détruit une ou plusieurs fenêtre(s).
layout	Convertit un bloc de code VID en un arbre de figures.
center-face	Centre une figure par rapport à son parent.
dump-face	Affiche une description compacte de la structure d'un arbre de figures (à des fins de débogage).
do-actor	Evalue manuellement un acteur de figure.
do-events	Lance une boucle d'événements (optionnellement, traite juste les événements en attente et se termine).
draw	Rend un bloc de dialecte Draw dans une image.
to-image	Convertit toute figure affichée en une image.
set-focus	Donne le focus à une figure spécifique.
size-text	Mesure la taille en pixels du texte dans une figure (en prenant en compte la police sélectionnée).

A ajouter:

- spécifications de la facette `menu`
- description du type de données `image!`