Aide de GizmoRBControls

30 juin 2013

FIGURE 1 – **GizmoRBControls** Version 2.0.2

Table des matières

Table des figures

1 Quel est l'usage de GizmoRBControls ?

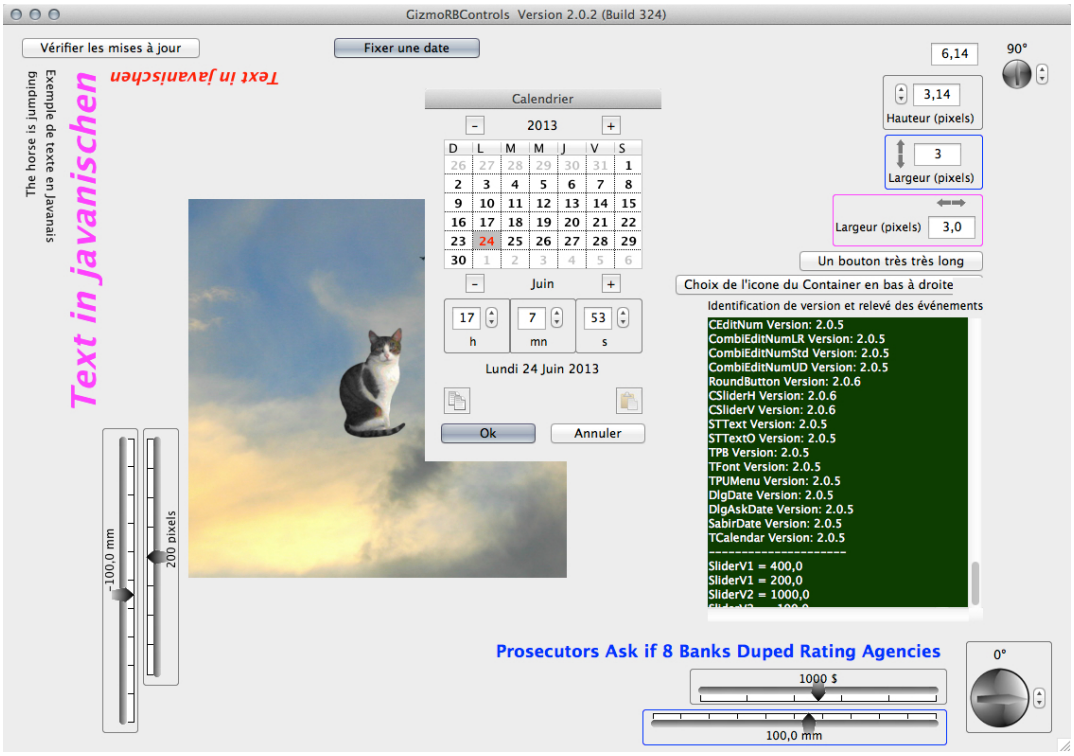

FIGURE 2 – GizmoRBControls sous Macintosh

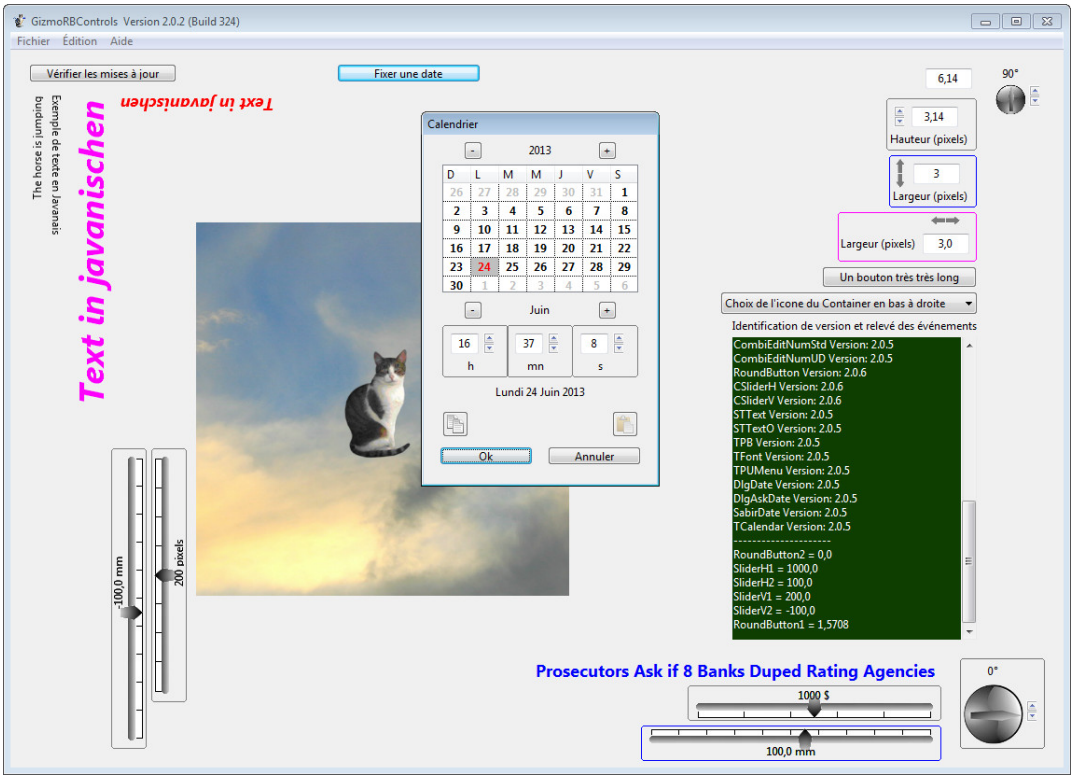

FIGURE 3 – GizmoRBControls sous Windows

Le programme décrit l'utilisation d'un certain nombre de **Classes**, de **Contrôles** ou de **Containers** utilisables avec **Xojo/RealBasic**. Ces composants sont disponibles pour **Macintosh**, **Windows** et **Linux** comme le montre les figures [2,3,4] qui illustrent GizmoRBControls sur les 3 plateformes.

GizmoRBControls est un gratuiciel et vous disposez des sources des contrôles et containers (situés dans le dossier **ExportGizmoCRB 2.0.2** figure [6]) que vous êtes libre de modifier. Les contrôles dans

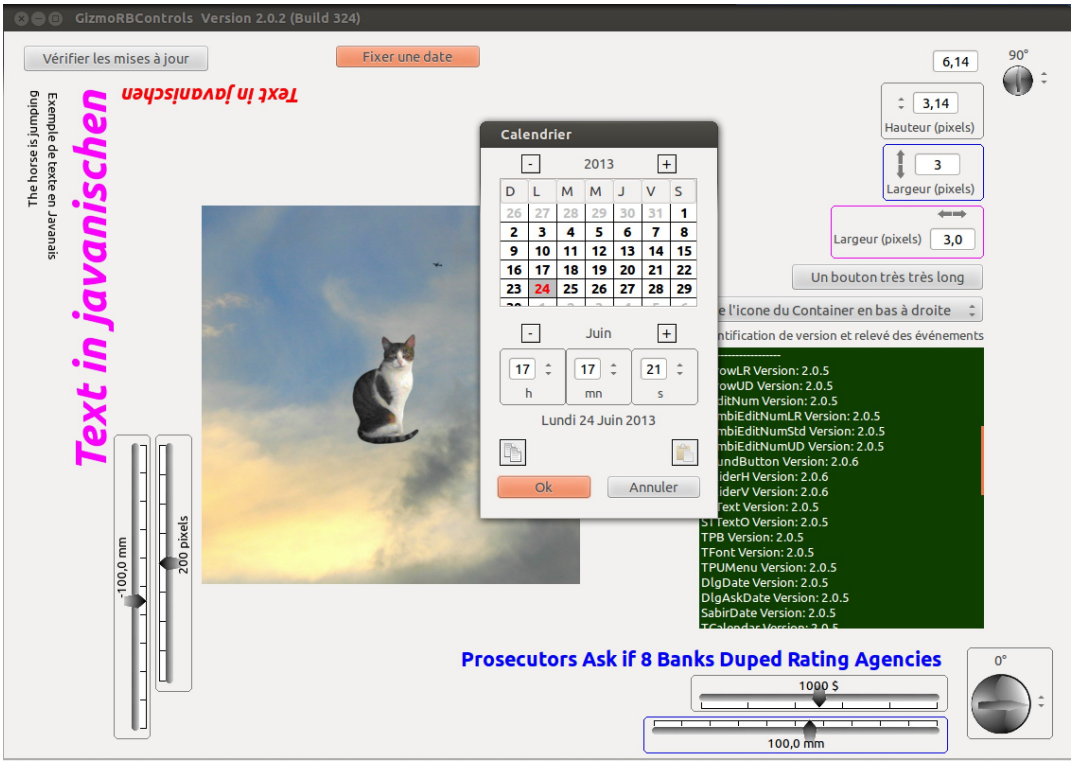

FIGURE 4 – **GizmoRBControls** sous Linux

l'interface de **GizmoRBControls** sont actifs. Vous pouvez vérifier leur comportement dans la zone de texte qui note tous les événements. Un des «Sliders »vertical permet de déplacer l'image de Gizmo le chat dans un canvas.

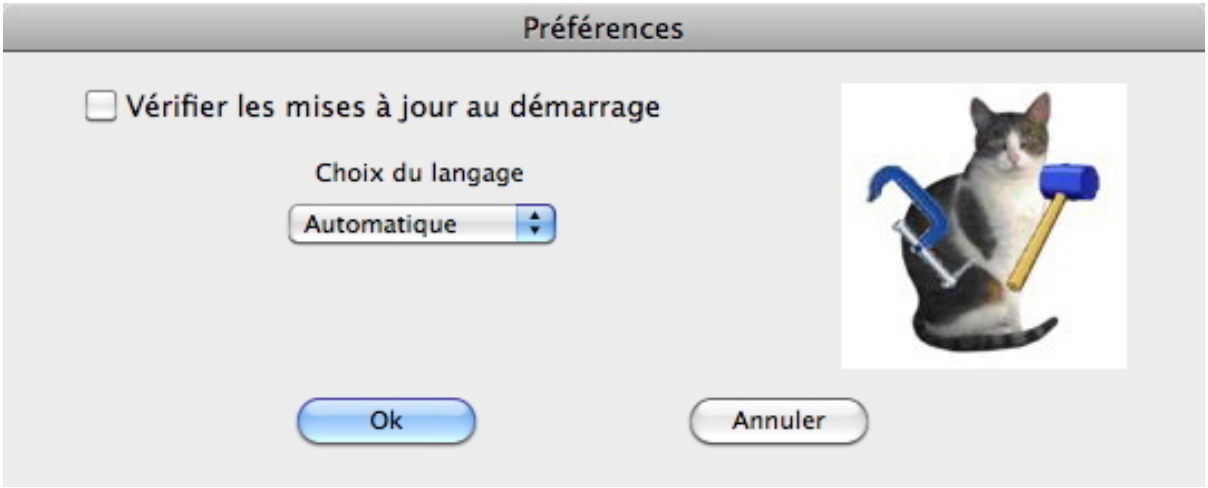

FIGURE 5 – Le dialogue Préférences permet de choisir le langage de l'interface : **Automatique**→ langage de la plateforme. **Anglais**, **Français** ou **Allemand** impose le langage.

1.1 Nouveautés de la version 2.0.2

Cette version introduit un nouveau dialogue (**DlgDate**) permettant le choix d'une date.
On a corrigé de nombreuses erreurs.
On a simplifié la gestion du langage et ajouté l'allemand.
Comme les éléments de **ExportGizmoCRB 2.0.2** sont modifiables par l'utilisateur, on a introduit un système élémentaire de gestion des versions (voir [1.4.2]).
Ce programme a été testé avec la nouvelle version **Xojo** de RealBasic. Les contrôles fournis sont **compatibles PowerPC, Intel, MacIntosh/Carbon, MacIntosh/Cocoa, Windows XP-Windows 8, Linux (Ubuntu version 8 à 12.X)**.

1.2 Limitations

Les classes et les contrôles sont utilisables par toutes les versions de Xojo/RealBasic. En revanche, les Containers ne sont utilisables *que dans les versions Pro de Xojo/RealBasic.*

1.3 Installation.

Avec **Windows**, l'ouverture de **SetupGizmoRBControls.exe** crée le dossier **ExportGizmoCRB 2.0.2** dans votre dossier Documents. Le contenu du dossier **ExportGizmoCRB 2.0.2** est illustré sur la figure[6]. Les fichiers qu'il contient sont soit des classes, soit des contrôles (extention .rbo), soit des containers (.rbw) ou des images (.png).

FIGURE 6 – Contenu du dossier **ExportGizmoCRB 2.0.2**. À noter que le dossier **PictureCRB 2.0.0** contient les différentes images nécessaires à l'utilisation de certains containers.

1.4 Utilisation

Pour utiliser les éléments disponibles dans le dossier **ExportGizmoCRB 2.0.2** (voir figure [6]), il suffit de faire un glisser déposer de ces éléments dans le volet **Projet** de l'EDI (Environnements de Développement Intégré) de votre programme en **Xojo/RealBasic**. La documentation qui suit vous précise les éléments à importer suivant le but recherché.

1.4.1 Localisation

Certains modules affichent des messages ou dépendent de la langue utilisé (les éléments utilisant **CEditNum** et **SabirDate**). Il est important de pouvoir localiser ces messages. Dans ce but il est conseillé d'utiliser le module **SabirDate** qui défini 3 constantes :

- English = 0
- French = 1
- German = 2

et une variable (*Langage* As Integer).

Ensuite, il suffit avant tout emploi d'un des contrôles, container ou dialogue de **GizmoRBControls** de définir *Langage = English, French ou German.* Il est possible d'introduire d'autres langages en modifiant le module **SabirDate** la procédure *SetSabirDate* de **SabirDate** ou la procedure *Setlangage* de **CEditNum**.

1.4.2 Maintenance

Afin de faciliter la maintenance le module **CRBGlobals** contient tous les numéros de version des éléments. Chaque élément contient également sont propre numéro de version. En mode Debug, ces numéros sont comparés pour vous signaler une inconsistance : si le numéro de version de l'élément est inférieur au numéro contenu dans **CRBGlobals**, le programme vous signale que l'élément utilisé n'est pas à jour.

À vous de maintenir ces numéros de version pour vous assurer (en ayant toujours la dernière version de **CRBGlobals** dans votre programme en développement) de la cohérence de l'utilisation de **ExportGizmoCRB 2.0.2**.

2 Le PushButton TPB

2.1 Utilité et mise en place.

FIGURE 7 – Instances de PushButton dérivé de TPB (Mac, Windows, Linux).

L'intérêt de ce type de bouton est que sa largeur et sa hauteur s'ajustent automatiquement à la largeur et à la hauteur du texte. C'est une propriété utile pour les boutons utilisés avec différentes localisations ou sous différents systèmes.

Il suffit de faire un glisser-déposer de l'élément TBP.rbo du dossier **ExportCRB 2.0.0** (voir figure [6]) dans le Projet de votre programme.

Soit TPB1 une instance de **TPB**. On dispose de 2 fonctions :

- **TPB1.Open** : cette fonction redimensionne automatiquement le bouton suivant le texte placé dans le contrôle au moment de la conception.
- **TPB1.SetText(S As String)** : équivallent à TPB1.Caption = S, mais redimensionne le bouton suivant la longueur du texte. Les propriétés TextFont, TextSize, Bold, Italic Underline et LockRight fixées à la conception sont conservées.

2.2 Exemple d'utilisation :

```
' TPB ----------------------------------------
  TPB1.SetText("A very very long button")
' Positionnement -----------------------------------
  TPB1.Top = CombiEditNumLR1.Top + CombiEditNumLR1.Height + 5
  TPB1.Left = Width - 20 - TPB1.Width
```

FIGURE 8 – Instance de PopUpMenu dérivé de TPUMenu (Mac, Windows, Linux).

3 Le PopUpMenu TPUMenu

3.1 Utilité et mise en place.

L'intérêt de ce type de PopUpMenu est que sa largeur et sa hauteur s'ajustent automatiquement à la largeur et à la hauteur du texte. C'est une propriété utile pour les PopUpMenus utilisés avec différentes localisations ou sous différents systèmes.

Il suffit de faire un glisser-déposer de l'élément TPUMenu.rbo du dossier **ExportCRB 2.0.0** (voir figure [6]) dans le Projet de votre programme.

Soit TPUMenu1 une instance de **TPUMenu**. On dispose de la fonction :

- **TPUMenu1. SetPopup(StrList As String, Index As Integer)** : équivallent à
  ```
  for i = 1 to CountFields(StrList,",")
     TPUMenu1.AddRow = NthField(StrList,'','',i)
  next i
  ```
  mais redimensionne le menu suivant la longueur des lignes de texte. Les propriétés **TextFont**, **TextSize**, **Bold**, **Italic**, **Underline** et **LockRight** sont conservées. **LastIndex** est fixé à Index.

```
'TPUMenu ----------------------------------------
  TPUMenu1.SetPopup("Choix de l'icone du Container en bas à droite,RButtonYingYang,
  RButtonBlue,RButtonBlueButterfly,RButtonWithArrow,Mythomaniac,Psychopathe",0)
' Positionnement ----------------------------------
  TPUMenu1.Top = TPB1.Top + TPB1.Height + 5
  TPUMenu1.Left = Width - 20 - TPUMenu1.Width
```

3.2 Récupération de la valeur.

Il suffit d'utiliser l'événement **Change** de l'instance TPUMenu1 de **TPUMenu**.

4 La classe TFont

4.1 Utilité et mise en place.

Cette classe a pour intérêt de passer les arguments relatifs à une fonte de façon compacte. Elle est utilisée dans le Label **STText** auto-dimensionnable, dans le Label orientable **STTextO** et d'autres contrôles.

Il suffit de faire un glisser-déposer de l'élément **TFont.rbo** du dossier **ExportCRB 2.0.0** (voir figure [6]) dans le Projet de votre programme.

Cette classe a pour propriétés :

- **Name As String** : le nom de la fonte.
- **Size As Integer** : la taille de la fonte.
- **Bold As Boolean**.
- **Italic As Boolean**.
- **Underline As Boolean**.
- **C As Color**.

Cette classe a une fonction **Init** :

```
Sub Init
  Name = "System"
  Size = 0
  Bold = false
  Italic = false
  Underline = false
  Colour = &c000000
```

4.2 Exemple d'utilisation :

```
'TFont ------------------------------------------
  Dim Fonte As TFont

  Fonte = new TFont
  Fonte.Init
  ...

  Fonte.Size = 18
  Fonte.Colour = rgb(0,180,180)
```

5 Le Label STText

5.1 Utilité et mise en place.

FIGURE 9 – Instances de Label dérivé de STText (Mac, Windows, Linux).

L'intérêt de ce type de Label est que sa largeur et sa hauteur s'ajustent automatiquement à la largeur et à la hauteur du texte. C'est une propriété utile pour les Label utilisés avec différentes localisations ou sous différents systèmes.

Il suffit de faire un glisser-déposer de la classe **TFont.rbo** et l'élément STText.rbw du dossier **ExportCRB 2.0.0** (voir figure [6]) dans le Projet de votre programme.

Soit **STText1** un contrôle instance de **STText**. On dispose des fonctions :

- **STText1.SetText(S As String)** redimensionne le Label en fonction du contenu de **S**.
- **STText1.ReSetText(S As String)** affiche **S** sans re-dimensionner le contrôle.
- **STText1.SetTextAlign(A As Integer)** fixe l'alignement (Label.AlignLeft, Label.AlignCenter, Label.AlignRight).
- **STText1.SetTextBold(Yes As Boolean)**.
- **STText1.SetTextColor(C As Color)** fixe la couleur du texte.
- **STText1.SetTextFont(font As String)** fixe la fonte du texte, font est le nom de la fonte.
- **STText1.SetTextItalic(Yes As Boolean)**.
- **STText1.SetTextMargin(m As Integer)** fixe la marge autour du texte.
- **STText1.SetTextSize(Size As Integer)** fixe la taille du texte.
- **STText1.SetTextTransparent(Yes As Boolean)** effectif seulement pour Windows.
- **STText1.SetTextUnderLine(Yes As Boolean)**.

De manière plus compacte on peut utiliser la classe **TFont**[4] :

- **STText1.SetTextFont(f As TFont)** fixe la fonte du Label.
- **STText1.SetText(S As String, f As TFont)** fixe le texte et la fonte du Label.

5.2 Exemple d'utilisation :

```
'STText -----------------------------------------------------
  STText1.SetTextBold(true)
  STText1.SetText("Prosecutors Ask if 8 Banks Duped Rating Agencies")
' Positionnement ----------------------------------
  STText1.Left = RoundButton2.Left - STText1.Width-20
  STText1.Top = RoundButton2.Top
' Réutilisation sans redimensionnement ----
  STText1.ResetText("The answer was no"
```

6 Le Label STTextO

6.1 Utilité et mise en place.

L'intérêt de ce type de Label est que sa largeur et sa hauteur s'ajustent automatiquement à la largeur et à la hauteur du texte. C'est une propriété utile pour les Label utilisés avec différentes localisations ou sous différents systèmes. **De plus le tracé du texte est orientable** à 0^0, 90^0, 180^0 ou -90^0. Voir les figures [2,3,4].

Il suffit de faire un glisser-déposer de l'élément STTextO.rbw du dossier **ExportCRB 2.0.0** (voir figure [6]) dans le Projet de votre programme.

Soit **STTextO1** un contrôle dérivé de **STTextO**. On dispose des fonctions :

- **STText.SetText(S As String,Angle As Integer)** redimensionne le Label en fonction du contenu de **S** et le trace avec l'orientation donnée par Angle. Angle est un entier compté en degré, positif dans le sens trigonométrique (sens inverse des aiguilles d'une montre). Seuls les angles 0^0, 90^0, 180^0 et -90^0 sont acceptés.
- **STText.SetText(S As String,Angle As Integer, AColor As Color)** redimensionne le Label en fonction du contenu de **S**, le trace avec l'orientation donnée par Angle et avec la couleur donnée par AColor.
- **STText.SetText(S As String,Angle As Integer, AColor As Color, Fonte As TFont)** redimensionne le Label en fonction du contenu de **S**, le trace avec l'orientation donnée par Angle, avec la couleur donnée par AColor et la police donnée par Fonte (voir la classe TFont [4]).

6.2 Exemple d'utilisation :

```
'STTextO ------------------------------------------
  STTextO1.SetText("The horse is jumping",90)
' Positionnement ----------------------------------
    STTextO1.Left = 20
  STTextO1.Top = PBCheckUpdate.Top + PBCheckUpdate.Height + 10
' Autre exemple -----------------------------------
  Dim Fonte As TFont

  Fonte = new TFont
  Fonte.Init
  Fonte.Size = 36
  Fonte.Bold = true
  Fonte.Italic = true
  STTextO3.SetText("Text in javanischen",90,&cFF0000,Fonte)
  ' Positionnement ----------------------------------
  STTextO3.Left = STTextO2.Left + STTextO2.Width + 5
  STTextO3.Top = STTextO1.Top
```

7 Le container CEditNum.

7.1 Utilité et mise en place.

Ce container permet de saisir des nombres entiers ou réels. On peut également fixer une valeur minimale et/ou une valeur maximale aux nombres saisis.

Il suffit de faire un glisser-déposer de l'élément **CEditNum.rbw** du dossier **ExportCRB 2.0.0** (voir figure [6]) et éventuellement la classe **TFont** (**TFont.rbo**)dans le Projet de votre programme.

Pour l'initialiser, il suffit d'utiliser la fonction **Resize** qui fixe ses dimensions et la fonction SetValue qui initialise son contenu :

- **CEditNum1.Resize(Marge As Integer, TextFieldWidth As Integer)**. Marge représente la marge autour du TextField et TextFieldWidth la largeur du TextField.
- **CEditNum1.SetValue(X)**. Si **X** est entier, on ne pourra saisir que des nombres entiers. Si **X** est double on pourra saisir des entiers ou des rééls.
- **CEditNum1.SetMaxValue(Max As Integer ou double)**. Spécifie qu'il existe une valeur maximale.

- **CEditNum1.SetMinValue(Max As Integer ou double)**. Spécifie qu'il existe une valeur minimale.
- **CEditNum1.Decrease**. Diminue la valeur de la quantité positive **Pas**.
- **CEditNum1.Increase**. Augmente la valeur de la quantité positive **Pas**.
- **CEditNum1.SetStep(p As Integer or double) :** fixe le **Pas**.
- Dans le cas des nombres réels on peut fixer leur format en spécifiant : **CEditNum.AFormat = "-###.##"** (par exemple).
- **CEditNum1.SetTextAlign(A As Integer)** fixe l'alignement (Label.AlignLeft, Label.AlignCenter, Label.AlignRight).
- **CEditNum1.SetTextBold(Yes As Boolean)**.
- **CEditNum1.SetTextColor(C As Color)** fixe la couleur du texte.
- **CEditNum1.SetTextFont(fontname As String)** fixe la fonte du texte, fontname est le nom de la fonte.
- **CEditNum1.SetTextFont(font As TFont)** fixe la fonte du texte, font donne les paramètres de la fonte[4].
- **CEditNum1.SetTextItalic(Yes As Boolean)**.
- **CEditNum1.SetTextMargin(m As Integer)** fixe la marge autour du texte.
- **CEditNum1.SetTextSize(Size As Integer)** fixe la taille du texte.
- **CEditNum1.SetTextTransparent(Yes As Boolean)** effectif seulement pour Windows.
- **CEditNum1.SetTextUnderLine(Yes As Boolean)**.
- **CEditNum1.SetBackColor(C As Color)** fixe la couleur du fond.

Il est également possible de fixer un minimum ou un maximum pour les nombres à saisir :
- **CEditNum1.SetMinValue(5.2)** pour spécifier la valeur minimale.
- **CEditNum1.SetMaxValue(15.0)** pour spécifier la valeur maximale.

Enfin, **CEditNum** *génère un évènement lorsque la valeur du nombre a changé* :
- ValueChanged : qui fourni la valeur X modifiée en double précision. Si vous manipulez des entiers posez Dim k As Integer = X dans le cas où vous devez réutiliser cette valeur.

7.2 Exemple d'utilisation :

```
' CEditNum ----------------------------------------
  CEditNum1.Resize(3,50) ' Marge As Integer, TextFieldWidth As Integer
  CEditNum1.SetValue(6.14)
  CEditNum1.SetMinValue(5.2)
  CEditNum1.SetMaxValue(15.0)
' Positionnement -----------------------------------
  CEditNum1.Top = RoundButton1.Top + RoundButton1.Height + 5
  CEditNum1.Left = Width - 20 - CEditNum1.width
```

7.3 Récupération de la valeur :

On peut récupérer la valeur contenue dans le champ d'édition soit par la fonction **GetValue**, soit par l'événement **ValueChanged** :
- **CEditNum1.GetValue** As double : renvoie la valeur numérique du TextField.
- **CEditNum1.ValueChanged(X As double)** : X est la valeur numérique du TextField.

8 Le container CombiEditNumStd

8.1 Utilité et mise en place.

Ce container permet de saisir des nombres entiers ou réels. On peut également fixer une valeur minimale et/ou une valeur maxumale aux nombres saisis. **De plus on dispose de flèches actives pour modifier l'entrée du nombre** (voir figure [10]). Le contrôle peut optionnellement être entouré d'une bordure.

Ce container contient :
- Un container **CEditNum**.
- Un contrôle **STText** pour la légende.
- Un contrôle **UpDownArrow**.

Il suffit de faire un glisser-déposer de l'élément CombiEditNumStd.rbw du dossier **ExportCRB 2.0.0** (voir figure [6]) dans le Projet de votre programme.
+ CEditNum + STText + TFont

Comme le montre la figure 10 le container **CombiEditNumStd** combine un **CEditNum**, une **légende** et un contrôle **UpDownArrow** standard.

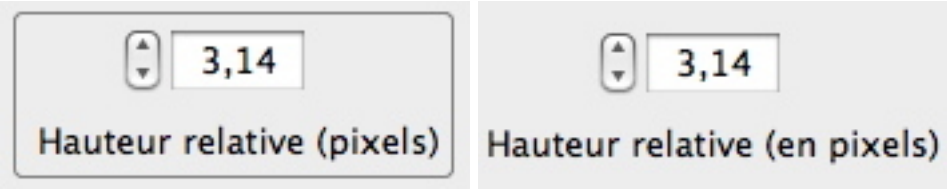

FIGURE 10 – CombiEditNum Standard avec et sans bordure.

8.2 Exemple d'utilisation :

```
' CombiEditNumStd -----------------------------
  CombiEditNumStd1.DispoLeg = CombiEditNumStd1.kLegBottom
  CombiEditNumStd1.DispoArrow = CombiEditNumStd1.kArrowLeft
  CombiEditNumStd1.SetText("Hauteur relative (pixels)")

  CombiEditNumStd1.Resize(3,50,&c7F7F7F)
  CombiEditNumStd1.SetValue(6.0)
  CombiEditNumStd1.SetMinValue(5.2)
  CombiEditNumStd1.SetMaxValue(15.0)
  CombiEditNumStd1.SetStep(1.0)
' Positionnement -----------------------------------
  CombiEditNumStd1.Top = CEditNum1.Top + CEditNum1.Height + 5
  CombiEditNumStd1.Left = Width - 20 - CombiEditNumStd1.Width
```

Les variables DispoLeg et DispoArrow fixent la disposition des éléments :
- **CombiEditNumStd1.kLegTop** : légende au dessus du Textfield.
- **CombiEditNumStd1.kLegLeft** : légende à gauche du Textfield.
- **CombiEditNumStd1.kLegRight** : légende à droite du Textfield.
- **CombiEditNumStd1.kLegBottom** : légende au dessous du Textfield.
- **CombiEditNumStd1.kArrowLeft** : contrôle UpDownArrow à gauche du Textfield.
- **CombiEditNumStd1.kArrowRight** : contrôle UpDownArrow à droite du Textfield.

À noter que c'est la position de la légende qui a la priorité.
Quant aux fonctions :

- **CombiEditNumStd1.SetText("Hauteur relative (pixels)")** elle permet de fixer la légende.
- **CombiEditNumStd1.Resize(Marge As Integer,TextFieldWidth As Integer,BorderColor As Color)** elle permet de fixer la marge, la largeur du TextField et la couleur du rectangle entourant le contrôle. **Si la couleur est omise, le retangle n'est pas tracé**.
- **CombiEditNumStd1.SetValue(Value As Integer/double)** elle fixe la valeur initiale du Textfield.
- **CombiEditNumStd1.SetValue(Value As Integer/double, AFormat As String)** elle fixe la valeur initiale du Textfield. Le format par défaut pour les réels est "-###.0##".
- **CombiEditNumStd1.SetMinValue(MinVal As Integer/double)** (optionelle) elle fixe la valeur minimale acceptable.
- **CombiEditNumStd1.SetMaxValue(MaxVal As Integer/double)** (optionelle) elle fixe la valeur maximale acceptable.
- **CombiEditNumStd1.SetStep(Step As Integer/double)** elle fixe la valeur utilisée par le contrôle UpDownArrow pour faire varier le TextField.

À noter que les appels à **SetMinValue** et **SetMaxValue** doivent être placés après **SetValue** qui annule les valeurs min et max.

8.3 Récupération de la valeur :

On peut récupérer la valeur contenue dans le champ d'édition soit par la fonction **GetValue**, soit par l'événement **ValueChanged** :

- **CombiEditNumStd1.GetValue** As double : renvoie la valeur numérique du TextField.
- **CombiEditNumStd1.ValueChanged(X As double)** : X est la valeur numérique du TextField.

9 Le container CombiEditNumUD

9.1 Utilité et mise en place.

À part le graphisme des flèches UpDown, le fonctionnement de ce contrôleur est identique au contrôleur **CombiEditNumStd** [8].

Ce container contient (voir figure [11]) :
- Un container **CEditNum**.
- Un contrôle **STText** pour la légende.
- Un container **ArrowUD**.
- Une image avec un masque (type PNG) : **ArrowUpN12x15**.
- Une image avec un masque (type PNG) : **ArrowUpG12x15**.
- Une image avec un masque (type PNG) : **ArrowDownN12x15**.
- Une image avec un masque (type PNG) : **ArrowDownG12x15**.

Il suffit de faire un glisser-déposer de l'élément CombiEditNumUD.rbw du dossier **ExportCRB 2.0.0** (voir figure [6]) dans le Projet de votre programme.

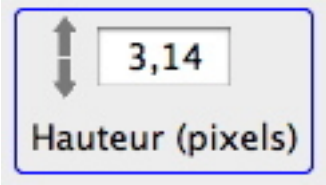

FIGURE 11 – CombiEditNumUD (UpDown modifié)

10 Le container CombiEditNumLR

À part le graphisme des flèches UpDown, le fonctionnement de ce contrôleur est identique au contrôleur **CombiEditNumStd** [8].

Ce container contient (voir figure [12]) :
- Un container **CEditNum**.
- Un contrôle **STText** pour la légende.
- Un container **ArrowLR**.
- Une image avec un masque (type PNG) : **ArrowLeftN12x15**.
- Une image avec un masque (type PNG) : **ArrowLeftG12x15**.
- Une image avec un masque (type PNG) : **ArrowRightN12x15**.
- Une image avec un masque (type PNG) : **ArrowRightG12x15**.

Il suffit de faire un glisser-déposer de l'élément CombiEditNumLR.rbw du dossier **ExportCRB 2.0.0** (voir figure [6]) dans le Projet de votre programme.

FIGURE 12 – CombiEditNumLR (LeftRight modifié) avec et sans bordure.

Les variables DispoLeg et DispoArrow fixent la disposition des éléments :
- **CombiEditNumStd1.kLegTop** : légende au dessus du Textfield.
- **CombiEditNumStd1.kLegLeft** : légende à gauche du Textfield.
- **CombiEditNumStd1.kLegRight** : légende à droite du Textfield.
- **CombiEditNumStd1.kLegBottom** : légende au dessous du Textfield.
- **CombiEditNumStd1.kArrowTop** : contrôle LeftRightArrow en haut du Textfield.
- **CombiEditNumStd1.kArrowBottom** : contrôle LeftRightArrow en bas du Textfield.

À noter que c'est la position de la légende qui a la priorité.

11 Container RoundButton

11.1 Utilité et mise en place.

Ce container permet de choisir un angle entre 0^0 et 360^0. On règle la valeur soit avec le contrôle UpDownArrow, soit en cliquant sur l'icone du bouton qui tourne suivant la valeur. L'icone du bouton peut être modifiée comme l'illustre la figure [13].

Il suffit de faire un glisser-déposer de l'élément RoundButton.rbw du dossier **ExportCRB 2.0.0** (voir figure [6]) dans le Projet de votre programme, ainsi que les images **RButtonYingYang.png, RButtonBlue.png, RButtonBlueButterfly.png et RButtonWithArrow.png**.

Ce container contient :

- Une image avec un masque (type PNG) :RButtonYingYang.png, RButtonBlue.png, RButtonBlue-Butterfly.png or RButtonWithArrow.png.
- Un canvas : **Cnvs**.
- Un contrôle UpDownArrow : **UDRot**.
- Un Label : **STDegre**.

FIGURE 13 – Container RoundButton. On peut modifier l'angle avec le contrôle UpDownArrow, le bouton circulaire ou avec la souris cliquant dans le bouton. L'aspect du contrôle peut être paramétré et l'utilisateur peut créer ses propres logos.

11.2 Exemple d'utilisation :

```
' RoundButton ---------------------------------------
  RoundButton1.Resize(48,3,0)
' Positionnement -------------------------------
  RoundButton1.Top = 14
  RoundButton1.Left = Width - 20 - RoundButton1.Width
```

On dispose de 3 fonctions pour l'initialisation :

- **RoundButton1.Resize(ButtonSize As Integer,Margin As Integer,InitialAngle As Integer)** qui fixe la taille du bouton (de 32 à 128 pixels), la marge autour du container et l'angle initial.
- **RoundButton1.Resize(ButtonSize As Integer,Margin As Integer,InitialAngle As Integer, PictureName As String)** qui fixe la taille du bouton (de 32 à 128 pixels), la marge autour du container, l'angle initial et le choix de l'image du bouton. Par défaut **PictureName** est **RButtonYingYang**). On dispose des valeurs suivantes :

  ```
  "RButtonYingYang"
  "RButtonBlue"
  "RButtonBlueButterfly"
  "RButtonWithArrow"
  ```

- **RoundButton1.Resize(ButtonSize As Integer,Margin As Integer,InitialAngle As Integer, PictureName As String, AColor As Color)** qui fixe la taille du bouton (de 32 à 128 pixels), la marge autour du container, l'angle initial, le choix de l'image du bouton ainsi qu'une bordure avec le choix de la couleur.

11.3 Récupération de la valeur :

On peut récupérer la valeur contenue dans le champ d'édition soit par la fonction **GetValue**, soit par l'événement **ValueChanged** :

- **RoundButton1.GetValue** As double : renvoie la valeur numérique de l'angle en radian. Les angles sont comptés dans le sens trigonométrique (inverse des aiguilles d'une montre).

– **RoundButton1. ValueChanged(X As double)** : X est la valeur numérique de l'angle en radians.

12 Container CSliderH

12.1 Utilité et mise en place.

Ce container permet de choisir une valeur entre une valeur min et une valeur max en faisant glisser le curseur sur le long de la règle graduée horizontale à l'aide de la souris[14]. On peut également, lorsque le container a le focus, augmenter la valeur en tapant sur la **touche +** et la diminuer en tapant sur **la touche -**. Le curseur devient bleu *lorsque le container a le focus.*

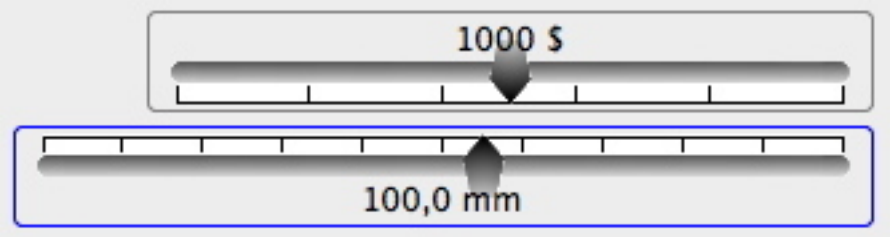

FIGURE 14 – Container **CSliderH**. On peut placer l'indicateur de valeur (1000 $ ou 100,0 mm) en haut (kGradUp) ou en bas (kGradDown) du curseur.. .

Ce container contient :
– Une image avec un masque (type PNG) : **LineLeft**.
– Une image avec un masque (type PNG) : **LineHorM**.
– Une image avec un masque (type PNG) : **LineRight**.
– Une image avec un masque (type PNG) : **PointeurUp**.
– Une image avec un masque (type PNG) : **PointeurUpOn**.
– Une image avec un masque (type PNG) : **PointeurDown**.
– Une image avec un masque (type PNG) : **PointeurDownOn**.
– Un canvas : **Cnvs**.

Il suffit de faire un glisser-déposer de l'élément CSliderH.rbw du dossier **ExportGizmoCRB 2.0.2** (voir figure [6]) dans le Projet de votre programme, ainsi que les images indiquées ci-dessus.

12.2 Exemple d'utilisation :

```
' CSliderH ------------------------------------
CSliderH1.Resize(250,500,1500,"$",5,CSliderH1.kGradUp,&c7F7F7F)
' Arguments: longueur de la règle graduée,Mini, Maxi,Unité,Nombre de graduations,Position de l'indica
' Valeur initiale
CSliderH1.SetValue(1000)
' Positionnement --------------------------------
CSliderH1.Left = RoundButton2.Left - CSliderH1.Width - 20
CSliderH1.Top = RoundButton2.Top + (RoundButton2.Height - CSliderH1.Height)/2
```

On dispose de 6 fonctions pour l'initialisation, 3 pour des valeurs réelles (double) :
– **CSliderH1.Resize(L As Integer, MinR As double, MaxR As double, Unity As String, NTics As Integer, PosLeg As Integer)**. Défini une barre de glissement. horizontale.
  – L = longueur de la barre de glissement (en pixels).
  – MinR = valeur minimale acceptable (double).
  – MaxR = valeur maximale acceptable (double).
  – Unity = unité des valeurs (par exemple mm, $, etc.).
  – NTics = nombre de graduations : l'intervalle MinR,MaxR est divisé en NTics.
  – PosLeg = position de l'indicateur de valeur. Voir ci-dessous pour les valeurs de PosLeg.
– **CSliderH1.Resize(L As Integer, MinR As double, MaxR As double, Unity As String, NTics As Integer, PosLeg As Integer, AFormat As String** Défini une barre de glissement. horizontale. Arguments indentiques à ceux de la fonction ci-dessus, plus un nouvel argument :
  – AFormat = format des nombres pour l'indicateur de valeur.
    Par défaut, AFormat = -###.0##.
– **CSliderH1.Resize(L As Integer, MinR As double, MaxR As double, Unity As String, NTics As Integer, PosLeg As Integer, AFormat As String, AColor As Color)** Défini une

barre de glissement. horizontale. Arguments indentiques à ceux de la fonction ci-dessus, plus un nouvel argument :
- AColor : si cet argument est présent, une bordure est tracée autour du container.

et 2 pour des valeurs entières :
- **CSliderH1.Resize(L As Integer, MinI As Integer, MaxI As Integer, Unity As String, NTics As Integer, PosLeg As Integer)**
- **CSliderH1.Resize(L As Integer, MinI As Integer, MaxI As Integer, Unity As String, NTics As Integer, PosLeg As Integer, AColor As Color)**

La seule différence avec les fonctions pour valeurs réelles est qu'ici les arguments MinI et MaxI sont des entiers et que l'argument AFormat a disparu.

Valeurs de la constante PosLeg (Slider horizontal) :
- **CSliderH1.kGradUp** : Indicateur de valeur au-dessus de la barre de glissement.
- **CSliderH1.kGradDown** : Indicateur de valeur en-dessous de la barre de glissement.

De plus on peut fixer la valeur initiale avec la fonction **SetValue** :
- **CSliderH1.SetValue(X As Integer/double) :** à utiliser **après** la fonction **Resize(...)**.

12.3 Récupération de la valeur :

On peut récupérer la valeur contenue dans le champ d'édition soit par la fonction **GetValue**, soit par l'événement **ValueChanged** :
- **CSliderH1.GetValue** As double : renvoie la valeur de l'indicateur de valeur.
- **CSliderH1.ValueChanged(X As double)** : X est la valeur de l'indicateur de valeur.

13 Container CSliderV

13.1 Utilité et mise en place.

Ce container permet de choisir une valeur entre une valeur min et une valeur max en faisant glisser le curseur sur le long de la règle graduée verticale à l'aide de la souris[15]. On peut également, lorsque le container a le focus, augmenter la valeur en tapant sur la **touche +** et la diminuer en tapant sur **la touche -**. Le curseur devient bleu *lorsque le container a le focus.*

Ce container contient :
- Une image avec un masque (type PNG) : **LineTop**.
- Une image avec un masque (type PNG) : **LineVerM**.
- Une image avec un masque (type PNG) : **LineBottom.**
- Une image avec un masque (type PNG) : **PointeurLeft**.
- Une image avec un masque (type PNG) : **PointeurLeftOn**.
- Une image avec un masque (type PNG) : **PointeurRight**.
- Une image avec un masque (type PNG) : **PointeurRightOn**.
- Un canvas : **Cnvs**.

Il suffit de faire un glisser-déposer de l'élément CSliderV.rbw du dossier **ExportCRB 2.0.0** (voir figure [6]) dans le Projet de votre programme, ainsi que les images indiquées ci-dessus.

L'utilisation de ce container est semblable à celle de **CSliderH**[12].

13.2 Exemple d'utilisation :

```
' CSliderV-------------------------------------
CSliderV1.Resize(250,500,1500,"$",5,CSliderV1.kGradLeft,&c7F7F7F)
' Arguments: longueur de la règle graduée,Mini, Maxi,Unité,Nombre de graduations,Position de l'indica
' Valeur initiale
CSliderV1.SetValue(1000)
' Positionnement --------------------------------
 CSliderV1.Left = CSliderV2.Left + CSliderV2.Width + 5
 CSliderV1.Top = CSliderV2.Top
```

On dispose de 6 fonctions pour l'initialisation, 3 pour des valeurs réelles (double) :
- **CSliderV1.Resize(L As Integer, MinR As double, MaxR As double, Unity As String, NTics As Integer, PosLeg As Integer)**. Défini une barre de glissement. horizontale.
  - L = longueur de la barre de glissement (en pixels).
  - MinR = valeur minimale acceptable (double).
  - MaxR = valeur maximale acceptable (double).

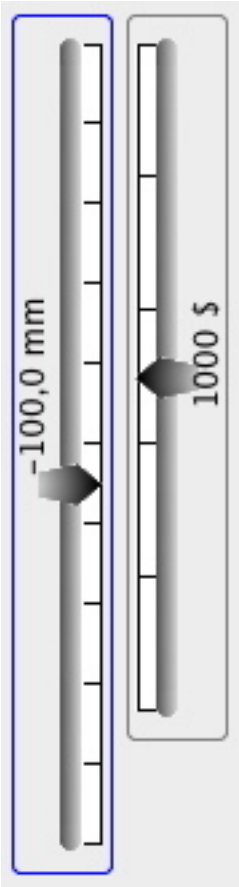

FIGURE 15 – Container **CSliderV**. On peut placer l'indicateur de valeur (1000 $ ou 100,0 mm) à gauche (kGradLeft) ou à droite (kGradRight) du curseur.

  - Unity = unité des valeurs (par exemple mm, $, etc.).
  - NTics = nombre de graduations : l'intervalle MinR,MaxR est divisé en NTics.
  - PosLeg = position de l'indicateur de valeur. Voir ci-dessous pour les valeurs de PosLeg.
- **CSliderV1.Resize(L As Integer, MinR As double, MaxR As double, Unity As String, NTics As Integer, PosLeg As Integer, AFormat As String** Défini une barre de glissement. horizontale. Arguments indentiques à ceux de la fonction ci-dessus, plus un nouvel argument :
  - AFormat = format des nombres pour l'indicateur de valeur.
    Par défaut, AFormat = -###.0##.
- **CSliderV1.Resize(L As Integer, MinR As double, MaxR As double, Unity As String, NTics As Integer, PosLeg As Integer, AFormat As String, AColor As Color)** Défini une barre de glissement. horizontale. Arguments indentiques à ceux de la fonction ci-dessus, plus un nouvel argument :
  - AColor : si cet argument est présent, une bordure est tracée autour du container.

et 2 pour des valeurs entières :
- **CSliderV1.Resize(L As Integer, MinI As Integer, MaxI As Integer, Unity As String, NTics As Integer, PosLeg As Integer)**
- **CSliderV1.Resize(L As Integer, MinI As Integer, MaxI As Integer, Unity As String, NTics As Integer, PosLeg As Integer, AColor As Color)**

La seule différence avec les fonctions pour valeurs réelles est qu'ici les arguments MinI et MaxI sont des entiers et que l'argument AFormat a disparu.

Valeurs de la constante PosLeg (Slider vertical) :
- **CSliderV1.kGradLeft** : Indicateur de valeur à gauche de la barre de glissement.
- **CSliderV1.kGradRight** : Indicateur de valeur à droite de la barre de glissement.

De plus on peut fixer la valeur initiale avec la fonction **SetValue** :
- **CSliderV1.SetValue(X As Integer/double) :** à utiliser **après** la fonction **Resize(...)**.

13.3 Récupération de la valeur :

On peut récupérer la valeur contenue dans le champ d'édition soit par la fonction **GetValue**, soit par l'événement **ValueChanged** :

- **CSliderV1.GetValue** As double : renvoie la valeur de l'indicateur de valeur.
- **CSliderV1.ValueChanged(X As double)** : X est la valeur de l'indicateur de valeur.

14 Dialogue DlgDate

14.1 Utilité et mise en place.

Ce dialogue permet de choisir commodément la date et l'heure comme le montre la figure [16]. Les boutons + et - permettent de naviguer suivant les mois et les années. En cliquant sur l'année ou le mois on peut accéder au choix d'une date quelquonque. En cliquant dans la grille on sélectionne la date et on peut choisir l'heure avec les champs relatifs au temps.

FIGURE 16 – Dialogue DlgDate. À noter les 2 boutons en bas du dialogue pour effectuer un **copier** ou un **coller**.

Ce dialogue contient :

- Le dialogue **DlgDate**.
- La classe **TCalendar**.
- Le container **CombiEditNumStd**.
- Le dialogue **DlgAskDate**.
- Le Label **STText**.
- Le container **CEditNum**.
- Le module **SabirDate**.

Il suffit donc de faire un glisser-déposer ou d'importer ces élément du dossier **ExportCRB 2.0.0** (voir figure [6]) dans le Projet de votre programme.

14.2 Récupération de la valeur :

On peut récupérer la valeur par la fonction **GetValue() As double**.

14.3 Exemple d'utilisation :

```
DlgDate.TotalSeconds = CurrentDate.TotalSeconds
DlgDate.ShowModal
```

```
 if DlgDate.IsOk then
    CurrentDate.TotalSeconds = DlgDate.GetValue
end if
```

où *CurrentDate* est une instance de *Date*.
N'oubliez pas de fixer la valeur de **Langage** avant l'appel au dialogue[1.4.1].