GizmoRBControls Help

June 30, 2013

Figure 1: GizmoRBControls Version 2.0.2

Contents

List of Figures

1 What is the use of GizmoRBControls ?

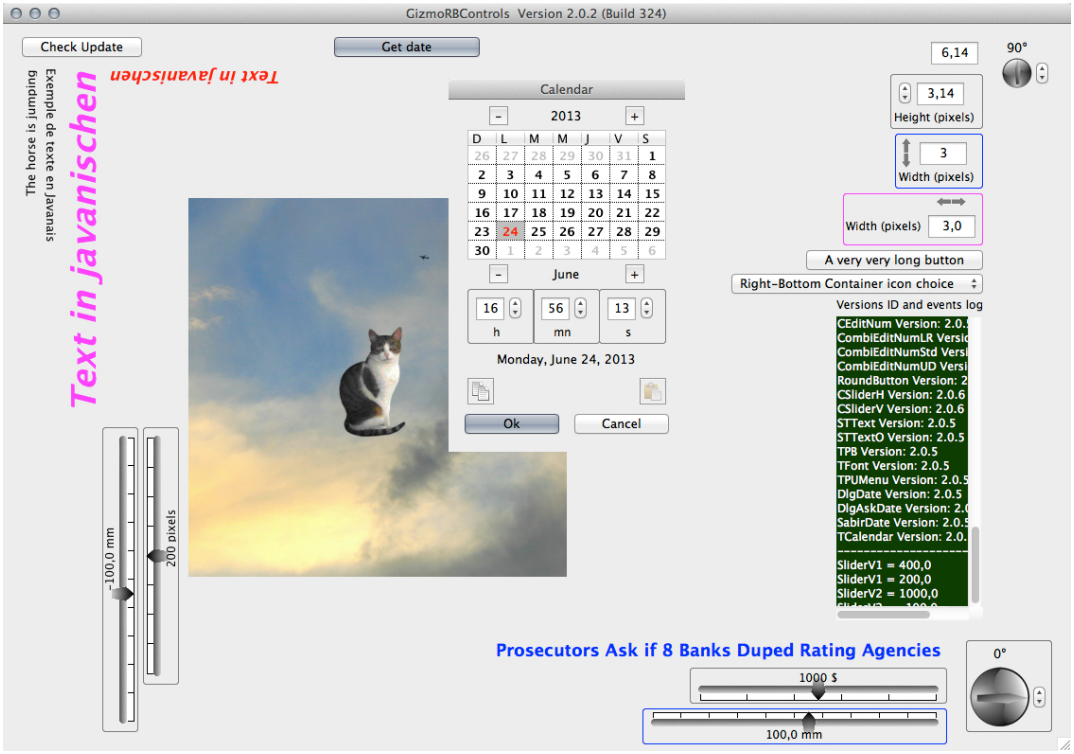

Figure 2: GizmoRBControls with Mac OS X

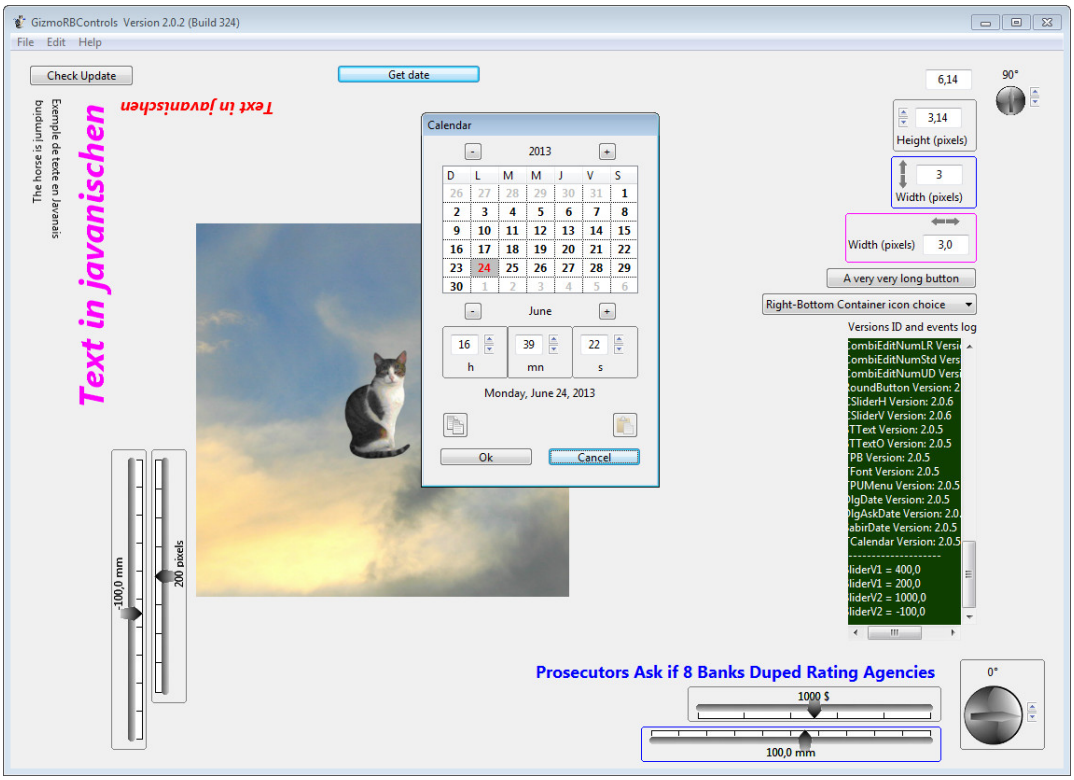

Figure 3: GizmoRBControls with Windows

This application illustrates some original **Class**, **Controls** or **Containers** usable with **Xojo/RealBasic**. These components can be deployed on the three platforms **Macintosh**, **Windows** and **Linux** as shown on figures [2,3,4].

GizmoRBControls is a freeware and is open source. You have at your disposal the sources (located in the folder **ExportGizmoCRB 2.0.2** figure [6]) that you may use and modify freely. In GizmoR-

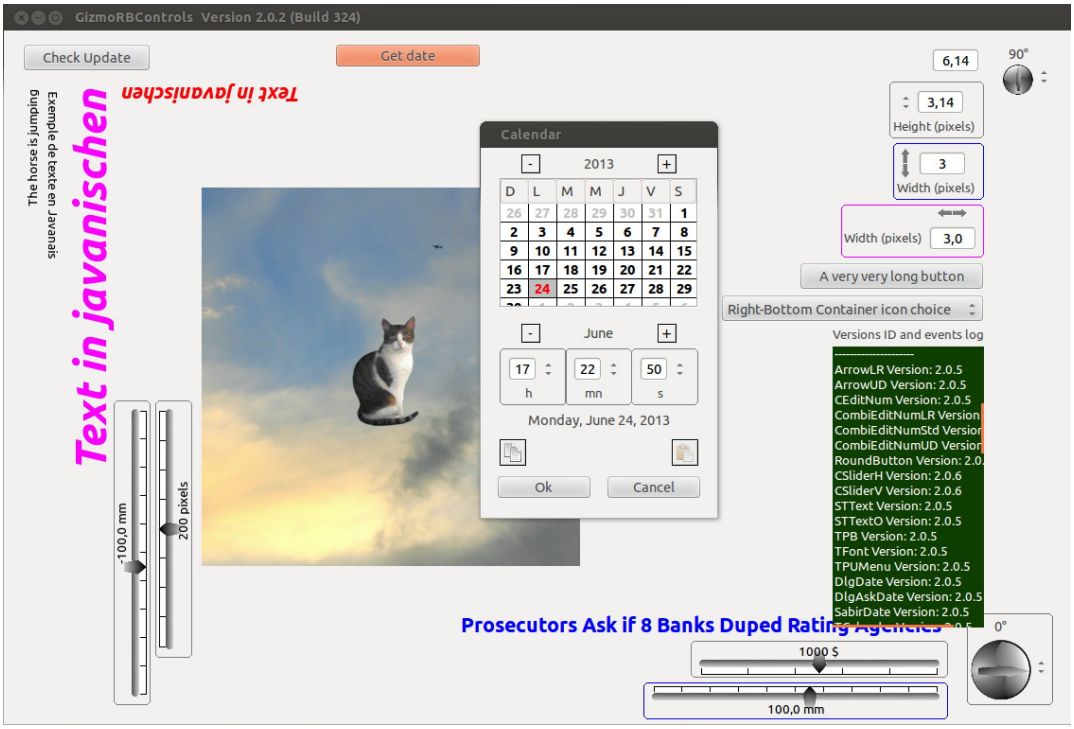

Figure 4: **GizmoRBControls** with Linux

BControls all controls are live. You may check their behavior and the text zonz is a log of all events. One of the vertical slider on the left allows to move the Gizmo's picture in a canvas.

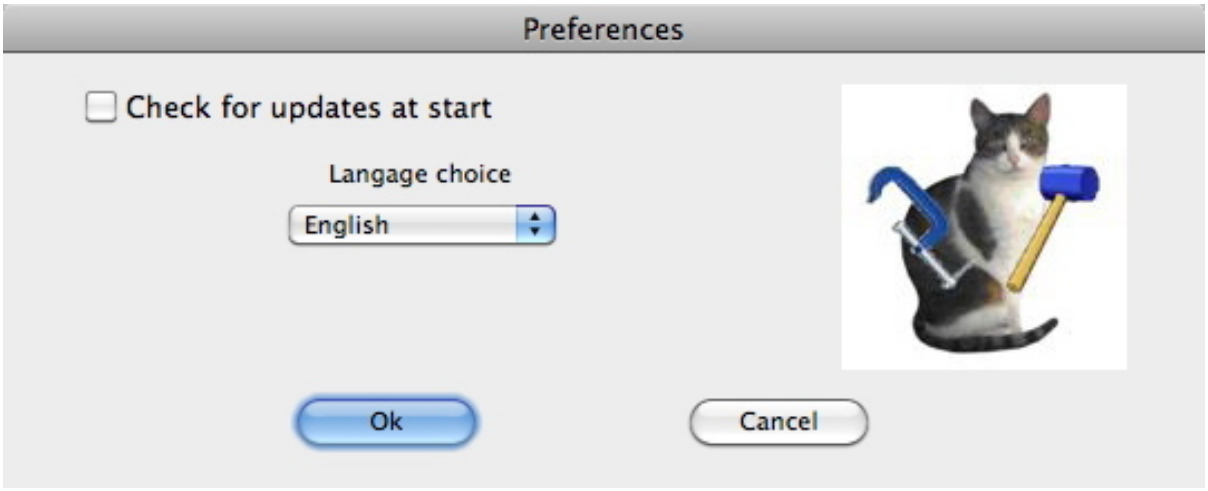

Figure 5: The Preference Dialog allows to select the interface language: **Automatic**→ for the platform language. **English**, **French** or **German** set independently the language.

1.1 What is new in version 2.0.2

This release introduces a new dialog (**DlgDate**) for choosing a date.
Many errors have been corrected.
It has simplified the management of language and added the German.
As the elements of **ExportGizmoCRB 2.0.2** are modified by the user, we introduced a basic version control (see [1.4.2]).

1.2 Limitations

The class and controls are usable with all versions of Xojo/RealBasic. But the container can only be used *with the Pro versions of Xojo/RealBasic*.

This program has been tested with the new version **Xojo** of REALbasic. The controls provided are **compatible PowerPC, Intel, Macintosh/Carbon, MacIntosh/Cocoa, Windows XP, Windows 8, Linux (Ubuntu Version from 8 to 12.X)**.

1.3 Installation.

Under Linux, the unpacking of **GizmoRBControls.tar.gz** can be done in any folder. One obtains the folder **GizmoRBControls** that contains GizmoRBControls application, folders GizmoRBControls Libs, Resources, ExportGizmoCRB 2.0.2, and the icon Icon256.png. Make sure that the program **GizmoRB-Controls** has permission to be executed and, optionally, associate the program icon.

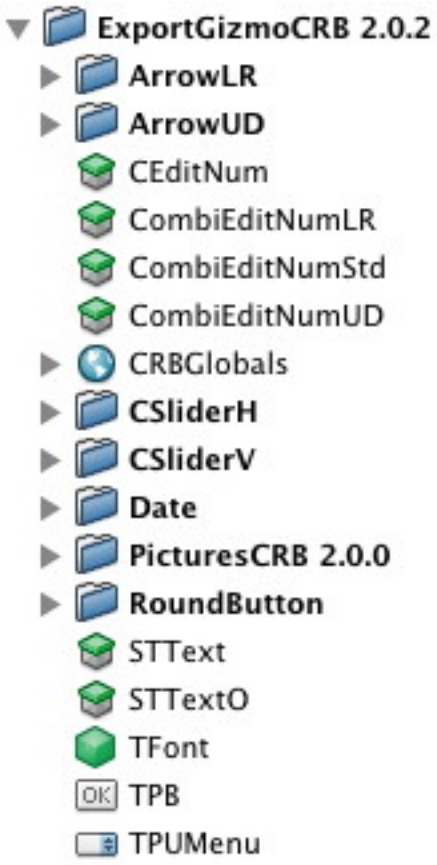

Figure 6: Contents of the **ExportGizmoCRB 2.0.2** folder

1.4 Use

To use the components of the **ExportGizmoCRB 2.0.2** (see figure [6]) folder, you simply do a drag and drop of theses components in the project tab of the Xojo/RealBasic EDI of your source program in **Xojo/RealBasic**. The following documentation explains wich elements to import and how to use then according to your goals.

1.4.1 Localization

Some modules display messages or depend on the language used (the elements using **CEditNum** and **SabirDate**). It is important to localize these messages. For this purpose it is advisable to use the module **SabirDate** who defined three constants:

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

and the variable (*Langage* As Integer).

Then just before making any use of controls, container or dialog of **GizmoRBControls** to define *Language = English, French or German*. It is possible to introduce other languages by modifying in the module **SabirDate** the procedure *SetSabirDate* and in **CEditNum**, the procedure *SetLangage*.

1.4.2 Maintenance

To facilitate maintenance, **CRBGlobals** module contains all elements version numbers. Each element contains also its own version number. In **Debug mode**, these numbers are compared to alert you of an inconsistency: if the version number of the element is less than the number contained in CRBGlobals, the program notifies you that the element used is outdated.

Its to you to take care of these version numbers to make sure (always with the latest version of your program **CRBGlobals** in developpement) of the consistency of your use of **ExportGizmoCRB 2.0.2**.

2 The PushButton TPB

2.1 Usefulness and implementation.

Figure 7: Instances of PushButton derived from TPB (Mac, Windows, Linux).

The interest of this kind of button is its width and height being automatically adjusted to the text length and type of font. This property is useful for buttons used in multiple localization or with different operating systems.

You only have to make a drag and drop of the TPB.rbo element from folder **ExportGizmoCRB 2.0.2** (see figure [6]) in the Project tab of your program

Let TPB1 be an instance of **TPB**. You have 2 functions at your disposal:

- **TPB1.Open**: this function resize automatically the button according to the text defined at design time.
- **TPB1.SetText(S As String)**: equivalent to TPB1.Caption = S, but resize the button according to S length. The properties TextFont, TextSize, Bold, Italic Underline and LockRight set at design time are preserved.

2.2 Example of use:

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

3 The PopUpMenu TPUMenu

3.1 Usefulness and implementation.

Figure 8: Instances of PopUpMenu derived from TPUMenu (Mac, Windows, Linux).

The interest of this kind of PopUpMenu is its width and height being automatically adjusted to the text length and type of font. This property is useful for buttons used in multiple localization or with different operating systems.

You only have to make a drag and drop of the TPUMenu.rbo element from folder **ExportGizmoCRB 2.0.2** (see figure [6]) in the Project tab of your program.

Let TPUMenu1 be an instance of **TPUMenu**. You have 1 function at your disposal:

- **TPUMenu1. SetPopup(StrList As String, Index As Integer)**: equivalent to

```
for i = 1 to CountFields(StrList,",")
   TPUMenu1.AddRow = NthField(StrList,'','',i)
next i
```

button is its width and height being automatically adjusted to the text length and type of font. The properties TextFont, TextSize, Bold, Italic Underline and LockRight set at design time are preserved. **TPUMenu1.LastIndex** is set to Index.

```
'TPUMenu ----------------------------------------
  TPUMenu1.SetPopup("Right-Bottom Container icon choice,RButtonYingYang,RButtonBlue,RButtonBlueButte
  RButtonWithArrow,Mythomaniac,Psychopathe",0)
' Positionnement ----------------------------------
  TPUMenu1.Top = TPB1.Top + TPB1.Height + 5
  TPUMenu1.Left = Width - 20 - TPUMenu1.Width
```

3.2 Retrieval of the component value.

Simply use the **TPUMenu1.Change** event of the instance TPUMenu1 of **TPUMenu**.

4 The Label STText

4.1 Usefulness and implementation.

Figure 9: Instances of Label derived from STText (Mac, Windows, Linux).

The interest of this kind of Label is its width and height being automatically adjusted to the text length and type of font. This property is useful for buttons used in multiple localization or with different operating systems.

You only have to make a drag and drop of the STText.rbw element from folder **ExportGizmoCRB 2.0.2** (see figure [6]) in the Project tab of your program.

Let **STText1** be an instance of **STText**. You have 1 function at your disposal:

- **STText1.SetText(S As String)** resize the Label acording to the **S** content.

4.2 Example of use:

```
'STText ------------------------------------------------
  STText1.SetText("Prosecutors Ask if 8 Banks Duped Rating Agencies")
' Positionnement -----------------------------------
  STText1.Left = RoundButton2.Left - STText1.Width-20
  STText1.Top = RoundButton2.Top
```

5 The class TFont

5.1 Usefulness and implementation.

The interest of this class is to pass font argument in a compact way. This class is used in the adjustable Label **STTextO**.

You only have to make a drag and drop of the TFont.rbo element from folder **ExportGizmoCRB 2.0.2** (see figure [6]) in the Project tab of your program.

Class properties:

- **Name As String**: font name.
- **Size As Integer**: font size.
- **Bold As Boolean**.
- **Italic As Boolean**.
- **Underline As Boolean**.

Class function **Init**:

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

5.2 Example of use:

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

  Fonte = new TFont
  Fonte.Init
  ...
```

6 The Label STTextO

6.1 Usefulness and implementation.

The interest of this kind of Label is its width and height being automatically adjusted to the text length and type of font. This property is useful for buttons used in multiple localization or with different operating systems. In addition, you can choose the text orientation:0^0, 90^0, 180^0 ou -90^0. See figures [2,3,4].

You only have to make a drag and drop of the STTextO.rbw element from folder **ExportGizmoCRB 2.0.2** (see figure [6]) in the Project tab of your program.

Let **STTextO1** be an instance of **STTextO**. You have the following functions at your disposal:

- **STText.SetText(S As String,Angle As Integer)** resize Label has a function of **S** and draw it with the orientation given by Angle. Angle is an integer counted in degree, positif in the trigonometric sense (counter clockwise). Only Angle 0^0, 90^0, 180^0 and -90^0 are permitted.

- **STText.SetText(S As String,Angle As Integer, AColor As Color)** resize Label has a function of **S** and draw it with the orientation given by Angle and the color set by AColor.
- **STText.SetText(S As String,Angle As Integer, AColor As Color, Fonte As TFont)** resize Label has a function of **S** and draw it with the orientation given by Angle, the color set by AColor and the font set by Fonte (see class TFont [5]).

6.2 Example of use:

```
'STText0 ----------------------------------------
  STText01.SetText("The horse is jumping",90)
' Positionnement ----------------------------------
    STText01.Left = 20
  STText01.Top = PBCheckUpdate.Top + PBCheckUpdate.Height + 10
' Other example -----------------------------------
  Dim Fonte As TFont

  Fonte = new TFont
  Fonte.Init
  Fonte.Size = 36
  Fonte.Bold = true
  Fonte.Italic = true
  STText03.SetText("Text in javanischen",90,&cFF0000,Fonte)
  ' Positionnement ----------------------------------
  STText03.Left = STText02.Left + STText02.Width + 5
  STText03.Top = STText01.Top
```

7 Le container CEditNum.

7.1 Usefulness and implementation.

This container is useful for numbers input. You can also set a minimum and/or a maximum value for number entry.

You only have to make a drag and drop of the CEditNum.rbw element from folder **ExportGizmoCRB 2.0.2** (see figure [6]) in the Project tab of your program.

Let **CEditNum1** be an instance of **CEditNum**. You have the following functions at your disposal:

- **CEditNum1.Resize(Marge As Integer, TextFieldWidth As Integer)**. Marge represent the margin arround the TextField and TextFieldWidth is the width of the TextField.
- **CEditNum1.SetValue(X)**. If **X** is an integer, the inputs can be only integers. If **X** is double the inputs can be only integers or reals.
- In the case of real numbers, you can specify their format:
  CEditNum.AFormat = "-###.##" (for example).

You can also set a minimum and/or a maximum for numbers input:

- **CEditNum1.SetMinValue(5.2)** specify a minimum value.
- **CEditNum1.SetMaxValue(15.0)** specify a maximum value.
- **CEditNum1.Setlanguage** Set the language of the warnings in case the input value exceed the range set by min,max.(See [8.2].

7.2 Example of use:

```
' CEditNum ---------------------------------------
  CEditNum1.Resize(3,50) ' Marge As Integer, TextFieldWidth As Integer
  CEditNum1.SetValue(6.14)
  CEditNum1.SetMinValue(5.2)
  CEditNum1.SetMaxValue(15.0)
  CEditNum1.Setlanguage
' Positionnement ----------------------------------
```

```
CEditNum1.Top = RoundButton1.Top + RoundButton1.Height + 5
CEditNum1.Left = Width - 20 - CEditNum1.width
```

7.3 Retrieval of the component value:

You can retrieve the value of the TextField by the function **GetValue**, or by the event **ValueChanged**:

- **CEditNum1.GetValue** As double: return the TextField value.
- **CEditNum1.ValueChanged(X As double)**: X is the TextField numerical value.

8 The container CombiEditNumStd

8.1 Usefulness and implementation.

This container is useful for numbers only input. You can also set a minimum and/or a maximum value for number entry. Moreover you can change the number input with an UpDownArrow control. This container has a caption and may be surrounded by an edge (optional) (see figure [10]).

Container content:

- A container **CEditNum**.
- A control **STText** for a caption.
- A control **UpDownArrow**.

You only have to make a drag and drop of the CombiEditNumStd.rbw element from folder **Export-GizmoCRB 2.0.2** (see figure [6]) in the Project tab of your program.

Figure 10: CombiEditNum Standard with and without edge.

8.2 Example of use:

```
' 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.Setlanguage
  CombiEditNumStd1.SetStep(1.0)
' Positionnement ----------------------------------
  CombiEditNumStd1.Top = CEditNum1.Top + CEditNum1.Height + 5
  CombiEditNumStd1.Left = Width - 20 - CombiEditNumStd1.Width
```

Variables DispoLeg and DispoArrow set the relative element positions:

- **CombiEditNumStd1.kLegTop**: caption above to Textfield.
- **CombiEditNumStd1.kLegLeft**: caption left to Textfield.

- **CombiEditNumStd1.kLegRight**: caption right to Textfield.
- **CombiEditNumStd1.kLegBottom**: caption at bottom of Textfield.
- **CombiEditNumStd1.kArrowLeft**: control UpDownArrow left to Textfield.
- **CombiEditNumStd1.kArrowRight**: control UpDownArrow right to Textfield.

Note that the caption position has priority.
You have the following functions at your disposal:

- **CombiEditNumStd1.SetText("Hauteur relative (pixels)")** set the caption text.
- **CombiEditNumStd1.Resize(Marge As Integer,TextFieldWidth As Integer,BorderColor As Color)** set the margin, the TextField width and the edge color. **If BorderColor is omitted, the edge arround the container is not drawn.**.
- **CombiEditNumStd1.SetValue(Value As Integer/double)** set the initial value of the Textfield.
- **CombiEditNumStd1.SetValue(Value As Integer/double, AFormat As String)** set the initial value of the Textfield. The default format for real numbers is "-###.0##".
- **CombiEditNumStd1.SetMinValue(MinVal As Integer/double)** (optional) set the minimum value.
- **CombiEditNumStd1.SetMaxValue(MaxVal As Integer/double)** (optional) set the maximum value.
- **CombiEditNumStd1.Setlanguage** set the language for the warnins (when the input is below the min value or above the max value). The function Setlanguage needs 3 global constants: English = 0, French = 1 and German = 2 and the global variable language (Integer) which has one of the 3 previous values.
- **CombiEditNumStd1.SetStep(Step As Integer/double)** set the value of the steps used by the UpDownArrow control to change the value in TextField.

Keep in mind that calls to **SetMinValue** and **SetMaxValue** should occur after **SetValue** which reset the min max values.

8.3 Retrieval of the component value:

You can retrieve the value of the TextField by the function **GetValue**, or by the event **ValueChanged**:

- **CombiEditNumStd1.GetValue** As double: return the TextField value.
- **CombiEditNumStd1.ValueChanged(X As double)**: X is the TextField numerical value.

9 The container CombiEditNumUD

9.1 Usefulness and implementation.

Except the appearance of the UpDown arrows the behavior of the container is identical to **CombiEditNumStd** [8].
Container contents (see figure [11]):

- A container **CEditNum**.
- A control **STText** for caption.
- A container **ArrowUD**.
- A picture with mask (type PNG): **ArrowUpN12x15**.
- A picture with mask (type PNG): **ArrowUpG12x15**.
- A picture with mask (type PNG): **ArrowDownN12x15**.
- A picture with mask (type PNG): **ArrowDownG12x15**.

You only have to make a drag and drop of the CombiEditNumUD.rbw element from folder **Export-GizmoCRB 2.0.2** and the associated pictures (see figure [6]) in the Project tab of your program.

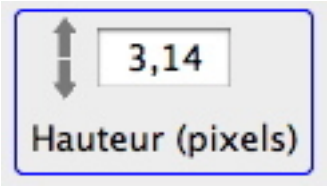

Figure 11: CombiEditNumUD (UpDown modified)

10 The container CombiEditNumLR

Except the appearance of the UpDown arrows the behavior of the container is identical to **CombiEditNumStd** [8].

Container contients (see figure [12]):

- A container **CEditNum**.
- A control **STText** for caption.
- A container **ArrowLR**.
- A picture with mask (type PNG): **ArrowUpN12x15**.
- A picture with mask (type PNG): **ArrowUpG12x15**.
- A picture with mask (type PNG): **ArrowDownN12x15**.
- A picture with mask (type PNG): **ArrowDownG12x15**.

You only have to make a drag and drop of the CombiEditNumLR.rbw element from folder **ExportGizmoCRB 2.0.2** and the associated pictures (see figure [6]) in the Project tab of your program.

Figure 12: CombiEditNumLR (LeftRight modified) with and without border.

Les variables DispoLeg et DispoArrow fixent la disposition des éléments:

- **CombiEditNumLR1.kLegTop**: caption above to Textfield.
- **CombiEditNumLR1.kLegLeft**: caption left to Textfield.
- **CombiEditNumLR1.kLegRight**: caption right to Textfield.
- **CombiEditNumLR1.kLegBottom**: caption at bottom of Textfield.
- **CombiEditNumLR1.kArrowTop**: control UpDownArrow to top of Textfield.
- **CombiEditNumLR1.kArrowBottom**: control UpDownArrow to bottom of Textfield.

Note that the caption position has priority.

11 Container RoundButton

11.1 Usefulness and implementation.

This container is designed for angles choice. The container allows to set an angle between 0^0 and 360^0. The value is choosen or by clicking with the mouse on the rotating button or using the UpDownArrow control. As shown on figure [13] the container appearance can be changed.

You only have to make a drag and drop of the CombiEditNumLR.rbw element from folder **ExportGizmoCRB 2.0.2** and the associated pictures **RButtonYingYang.png, RButtonBlue.png, RButtonBlueButterfly.png and RButtonWithArrow.png** (see figure [6]) in the Project tab of your program.

Container contents:

- A picture with a mask (type PNG): RButtonYingYang.png, RButtonBlue.png, RButtonBlueButterfly.png or RButtonWithArrow.png
- A canvas: **Cnvs**.
- A control UpDownArrow: **UDRot**.
- A Label: **STDegre**.

Figure 13: Container RoundButton. You can change the angle with the UpDownArrow control, the circular button by the mouse. The appearence of the container can be set and the user may create its own logos.

11.2 Example of use:

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

You have the following functions at your disposal:

- **RoundButton1.Resize(ButtonSize As Integer,Margin As Integer,InitialAngle As Integer)** set the button size (32 to 128 pixels), the margin arround the container and the initial angle.
- **RoundButton1.Resize(ButtonSize As Integer,Margin As Integer,InitialAngle As Integer, PictureName As String)** set the button size (32 to 128 pixels), the margin arround the container, the initial angle and the choice of the button picture. By default **PictureName** is **RButtonYingYang**). The following values are available:

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

- **RoundButton1.Resize(ButtonSize As Integer,Margin As Integer,InitialAngle As Integer, PictureName As String, AColor As Color)** set the button size (32 to 128 pixels), the margin arround the container, the initial angle, the choice of the button picture and the bordure color which will be drawn.

11.3 Retrieval of the component value:

You can retrieve the value of the TextField by the function **GetValue**, or by the event **ValueChanged**:

- **RoundButton1.GetValue** As double: return the angle value in radians (positif in the trigonometric sense (counter clockwise).

- **RoundButton1. ValueChanged(X As double)**: X is the numerical value of the angle in radians (positif in the trigonometric sense (counter clockwise).

12 Container CSliderH

12.1 Usefulness and implementation.

This container allows to set a number bounded by a minimum and a maximum value by dragging with the mouse a cursor (see fig. [14]) along an horizontal ruler. When the container has the focus, you can also use the keyboard: typing **key +** increase the number value and typing **key -** decrease it. The cursor of the container becomes blue *when the container has the focus.*

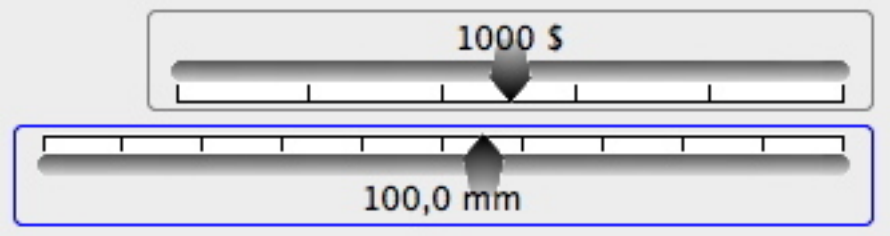

Figure 14: Container **CSliderH**. You can set the value indicator (1000 $ ou 100,0 mm) above (kGradUp) or below (kGradDown) the container's cursor.. .

Container contents:

- A picture with a mask (type PNG): **LineLeft**.
- A picture with a mask (type PNG): **LineHorM**.
- A picture with a mask (type PNG): **LineRight**.
- A picture with a mask (type PNG): **PointeurUp**.
- A picture with a mask (type PNG): **PointeurUpOn**.
- A picture with a mask (type PNG): **PointeurDown**.
- A picture with a mask (type PNG): **PointeurDownOn**.
- A canvas: **Cnvs**.

You only have to make a drag and drop of the CSliderH.rbw element from folder **ExportGizmoCRB 2.0.2** and the associated pictures (listed above) in the Project tab of your program.

12.2 Example of use:

```
' CSliderH ------------------------------------
CSliderH1.Resize(250,500,1500,"$",5,CSliderH1.kGradUp,&c7F7F7F)
' Arguments: Ruler's length,Mini, Maxi,Unity,Number of tics,Value Indicator position,Format,Border c
' Initial value
CSliderH1.SetValue(1000)
' Positionnement --------------------------------
CSliderH1.Left = RoundButton2.Left - CSliderH1.Width - 20
CSliderH1.Top = RoundButton2.Top + (RoundButton2.Height - CSliderH1.Height)/2
```

At your disposal you have 6 functions for initialisation, 3 for real values (MinR and MaxR, double):

- **CSliderH1.Resize(L As Integer, MinR As double, MaxR As double, Unity As String, NTics As Integer, PosLeg As Integer)**. Set an horizontal slider.
  - L = length of the sliding bar (in pixels).
  - MinR = minimum value (double).
  - MaxR = maximum value (double).
  - Unity = unity for the values (for example mm, $, etc.).
  - NTics = number of tics: the interval MinR,MaxR is linearly divide in NTics.

– PosLeg = Value indicator position. See below for the PosLeg values.

- **CSliderH1.Resize(L As Integer, MinR As double, MaxR As double, Unity As String, NTics As Integer, PosLeg As Integer, AFormat As String** Set an horizontal slider. Arguments identical to the function above. New argument:
  – AFormat = number's format for the value indicator.
  The default value is, 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)** Set an horizontal slider. Arguments identical to the function above. New argument:
  – AColor: when this argument is present, a border arround the container is drawn with this color value.

and 2 for integer values (MinI and MaxI):

- **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)**

The only difference with the functions for real arguments is that the AFormat variable has disapeared (MinI et MaxI are integers).

Values of the PosLeg constant (Slider horizontal) :

- **CSliderH1.kGradUp**: Value indicator above the slide bar.
- **CSliderH1.kGradDown**: Value indicator below the slide bar.

In addition, you can set the initial value with the function **SetValue**:

- **CSliderH1.SetValue(X As Integer/double):** use only **after** function **Resize(...)**.

12.3 Retrieval of the component value:

You can retrieve the value of the TextField by the function **GetValue**, or by the event **ValueChanged**:

- **CSliderH1.GetValue** As double: return the value of the indicator value.
- **CSliderH1. ValueChanged(X As double)**: X is the numerical value of the indicator value

13 Container CSliderV

13.1 Usefulness and implementation.

This container allows to set a number bounded by a minimum and a maximum value by dragging with the mouse a cursor (see fig. [15]) along an vertical ruler. When the container has the focus, you can also use the keyboard: typing **key +** increase the number value and typing **key -** decrease it. The cursor of the container becomes blue *when the container has the focus.*

Container contents:

- A picture with a mask (type PNG): **LineTop**.
- A picture with a mask (type PNG): **LineVerM**.
- A picture with a mask (type PNG): **LineBottom.**
- A picture with a mask (type PNG): **PointeurLeft**.
- A picture with a mask (type PNG): **PointeurLeftOn**.
- A picture with a mask (type PNG): **PointeurRight**.
- A picture with a mask (type PNG): **PointeurRightOn**.
- A canvas: **Cnvs**.

You only have to make a drag and drop of the CSliderV.rbw element from folder **ExportGizmoCRB 2.0.2** and the associated pictures (listed above) in the Project tab of your program.

The use of this container is similar to the use of **CSliderH**[12].

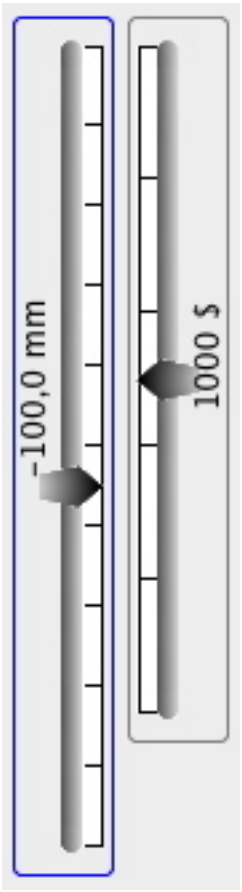

Figure 15: Container **CSliderV**. You can set the value indicator (1000 $ ou 100,0 mm) to the left (kGradLeft) or to the right (kGradRight) the container's cursor..

13.2 Example of use:

```
' CSliderV-------------------------------------
CSliderV1.Resize(250,500,1500,"$",5,CSliderV1.kGradLeft,&c7F7F7F)
' Arguments: Ruler's length,Mini, Maxi,Unity,Number of tics,
' Value Indicator position,Format,Border color.
' Initial value
CSliderV1.SetValue(1000)
' Positionnement ---------------------------------
 CSliderV1.Left = CSliderV2.Left + CSliderV2.Width + 5
 CSliderV1.Top = CSliderV2.Top
```

At your disposal you have 6 functions for initialisation, 3 for real values (MinR and MaxR, double):

- **CSliderV1.Resize(L As Integer, MinR As double, MaxR As double, Unity As String, NTics As Integer, PosLeg As Integer)**. Set an horizontal slider.
  - L = length of the sliding bar (in pixels).
  - MinR = minimum value (double).
  - MaxR = maximum value (double).
  - Unity = unity for the values (for example mm, $, etc.).
  - NTics = number of tics: the interval MinR,MaxR is linearly divide in NTics.
  - PosLeg = Value indicator position. See below for the PosLeg values.
- **CSliderV1.Resize(L As Integer, MinR As double, MaxR As double, Unity As String, NTics As Integer, PosLeg As Integer, AFormat As String** Set an horizontal slider. Arguments identical to the function above. New argument:
  - AFormat = number's format for the value indicator.
    The default value is, 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)** Set an horizontal slider. Arguments identical to the function above. New argument:
  - AColor: when this argument is present, a border arround the container is drawn with this color value.

and 2 for integer values (MinI and MaxI):

- **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)**

The only difference with the functions for real arguments is that the AFormat variable has disapeared (MinI et MaxI are integers).

Values of the PosLeg constant (Slider vertical) :

- **CSliderV1.kGradLeft**: Value indicator to the left of the slide bar.
- **CSliderV1.kGradRight**: Value indicator to the right of the slide bar.

In addition, you can set the initial value with the function **SetValue**:

- **CSliderV1.SetValue(X As Integer/double):** use only **after** function **Resize(...)**.

13.3 Retrieval of the component value:

You can retrieve the value of the TextField by the function **GetValue**, or by the event **ValueChanged**:

- **CSliderV1.GetValue** As double: return the value of the indicator value.
- **CSliderV1. ValueChanged(X As double)**: X is the numerical value of the indicator value

14 Dialog DlgDate

14.1 Usefulness and implementation.

With this dialog you can conveniently choose the date and time as shown in Figure [16]. The buttons + and - allow to navigate along the months and years. Clicking on the year or month you can access the choice of an arbitrary date. By clicking on the grid you select the date and you can choose the time with the time-related fields.

This dialog contents:

- The dialog **DlgDate**.
- The Class **TCalendar**.
- The container **CombiEditNumStd**.
- The sub-dialog **DlgAskDate**.
- The Label **STText**.
- The container **CEditNum**.
- The module **SabirDate**.

You only have to make a drag and drop or to import these elements from folder **ExportGizmoCRB 2.0.2** in the Project tab of your program.

14.2 Retrieval of the value:

You can retrieve the value by the function **GetValue() As double**.

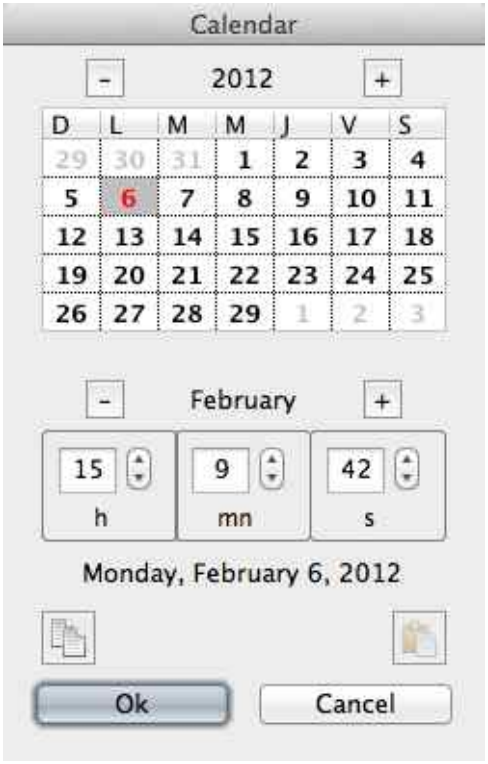

Figure 16: Dialog **DlgDate**. Note the two buttons at the bottom of the dialog to make a **copy** or **paste**.

14.3 Example of use:

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

where **CurrentDate** which is an instance of **Date**.
Don't forget to set the value of **Langage** before using the dialog[1.4.1].