RubberViews™

Fit all screen sizes without extra code

RubberViews is a single class for Xojo that takes care of resizing all controls on a Window or on a ContainerControl, including font size, image sizes for ImageWell, Canvas Backdrop and BevelButton Icon. Moreover, ListBox RowHeight is resized as well to accommodate the new font height. Also, if a window or ContainerControl backdrop image is used, it is resized automatically, with option to have the picture stretched to fill the entire window. If you have downloaded the evaluation package, it will display a MsgBox telling about it's nature at every run.
To remove that MsgBox, dash by our site http://RubberViews.com to acquire the license.

RubberViews is extremely convenient when screen size varies considerably, between desktop computer usually at 1920x1080, laptops often used at 1280x1080, Windows tablets at 1280x800 or even 1024x765, and on the other size of the spectrum high dpi screens at 2550x1600 or more. And those are standard resolutions. Any resizeable window can adopt any size in between, as well as any aspect ratio. It is impossible to do with fixed size windows. RubberViews lets you design one window, and have it displayed in the best of conditions for all these resolutions.

RubberViews is very simple to use. Here is how to start right away. Elastic Window users can use the Drop-In replacement, see page 4.

3 minutes Start :

- Drag the RubberViews_Classes folder into the project
- Open the folder and drag the RubberViews class over the Window or ContainerControl,
- In App, add two properties : MSNameL and MSRubberViewsL as string
- In the Window or ContainerControl Open event, call the Init method :

```
RubberViews1.Init(Self)
```

If no width and height is specified, RubberViews will take the size of the window at the time of the call as reference. Problem is, when that size is smaller than the design time size, resize is thrown off. This happens mostly under Windows, where a window cannot exceeed the screen size, so a window designed for instance at 1920x1080, ran on a 1024x768 device will not have the required size at opening of the window, and RubberViews will not work adequately. To avoid this issue, enter the design width and height of the window, for instance :

```
RubberViews1.Init(Self, 1200, 800)
```

Then to have the controls resized

```
RubberViews1.SizAll(Self)
```

Once again, there are options available for that method. The complete parameters are :

```
SizAll(Win as window, optional noContentResize as Boolean,
optional KeepRatio as Boolean)
```

noContentResize as its name indicates, makes RubberViews resize controls but not font sizes or pictures.

```
RubberViews1.SizAll(Self, True)
```

KeepRatio will insure that no matter the shape of the resized window, controls will keep their original aspect ratio. This is important for circles and squares, for instance.

```
RubberViews1.SizAll(Self, False, True)
```

Note that these options apply to all controls on the page. See next paragraph to select controls.

KeepRatios
It is possible to apply KeepRatio only to selected controls. For instance, in the demo project, all controls resize freely, but the oval remains circular.To obtain that, add the name of the control to the *RubberViews KeepRatios* property. Please note that it is plural, as opposed to the SizAll keepRatio property which is singular.

```
RubberViews1.KeepRatios = RubberViews1.KeepRatios + "Oval1"
```

This property is available in the IDE, so you can add all controls you want there as well. Important : If you indicate only part of the control name that is common to several, for instance "Oval", all controls that have "Oval" in their name will be concerned.

Ignore
You also may want RubberViews not to resize and move some controls. This is achieved with the Ignore property.

RubberViews1.Ignore = RubberViews1.Ignore + "CheckBox1"

This property is available in the IDE, so you can add all controls you want there as well. Global ignore : If you indicate only part of the control name that is common to several, for instance "Label", all controls that have "Label" in their name will be ignored.

StretchWindowBackdrop
If you are using a Window or ContainerControl property, it will be resized as well, unless noContentResize is True. However, we added an additional facility for that particular picture. Often one wants to have the backdrop picture fill the entire window background. This is achieved by setting StretchWindowBackdrop to True before calling SizAll :

```
RubberViews1.StretchWindowBackdrop = True
```

NoContentResizes
You may want to prevent the resize of the content for a particular control. For instance, in the demo, the RubberViews version indicated in the lower right corner. Simply add the name of the control to the NoContentResizes property :

```
RubberViews1.NoContentResizes = RubberViews1.NoContentResizes +
"Label5"
```

This property is available in the IDE, so you can add all controls you want there as well. Important : If you indicate only part of the control name that is common to several, for instance "Label", all controls that have "Label" in their name will be concerned.

RubberCanvas
Canvas has two different ways to display pictures. RubberViews takes care of resizing canvas backdrop automatically, but not if the drawing is done in the Paint event. In order to allow automatic resize of drawings made into Canvas.Paint, we provide a class called RubberView. To use, copy to the project navigator, and simply change the super of the Canvas to RubberCanvas. Everything works the same, and most of the code stays unchanged. You got to make sure, though, to keep all parameters absolute. For instance, if you want to draw a rectangle along the edge of the canvas, do not use the current size, but the design size instead. See the demo project RubberView1 Paint event code for reference :

```
~~g.DrawRect(0,0,me.width,me.height)~~ // Not this

g.DrawRect(0,0,161,168) // Use the control absolute design size
```

Cross Platform

RubberViews is cross platform and can be used on Mac, Windows and Linux projects.

On Mac, double buffering makes it extremely smooth and practically flicker free. SizAll can be used in the Window Resizing event practically without limit. RubberViews takes advantages of some of Macintosh features. For instance, since the ProgressWheel can be sized from the design size 16 up to 24, it will resize accordingly in SizAll.

On Windows on a physical PC (as opposed to VM), because the platform is intrisically more flicker prone, it is necessary to make some precautions. One of them is to avoid placing RubberViews.SizAll()) in the Resizing event, or to do so with the noContentResize True, to avoid pictures flickering. Then eventually, use SizAll with all options in Resized. That is the way the demo works. If your window has a backdrop image, it will make all controls flicker, especially labels, groupboxes, checkboxes, canvases. The best way to mitigate that is to avoid placing SizAll in the Resizing event.

On Linux, since there are many different environments, some better than others with graphics, flicker may not show too much. However, most of the Windows precautions should be applied.

Elastic Window Direct Drop-In Replacement

Users of Elastic Window have been orphaned by its disappearance last year, so RubberViews comes with a direct drop-in replacement to substitute in place of Elastic Window.

- Drag the RubberViews classes into your project
- Make the super of your windows RubberWindow instead of ElasticWindow.

  Alternatively, if you have many windows, you can remove the ElasticWindow class, and rename RubberWindow inside the RubberViews Classes folder to ElasticWindow.

RubberWindow emulates all standard Elastic Window methods. However, some have been placed there for compatibility, as they have no equivalent. For instance ResizeMode, which is not necessary (RubberViews manages that transparently), or Register, managed differently. Others, such as Ignore, will work just the same, but you can have more possibilities with RubberViews.

Basically, RubberWindow is a window class which wraps RubberViews .

If you need access to all the additional features of RubberViews such as keepRatio, StretchWindowBackdrop or global ignore for instance, you can access the internal RubberViews class properties and methods like so :

```
MsgBox Self.RubberViews.Version
```

RubberContainerControl is the drop-in Super for Container Controls. You can proceed the same and rename it as ElasticContainerControl if you have many ContainerControls.

Supported versions

RubberViews is supported on all versions of Xojo since 2013R1. Although it is not supported on RealStudio, it should work with Enterprise versions of RS that have ContainerControl support (RubberViews is a ContainerControl). You can probably modify the code to work on other versions of RealStudio or even RealBasic by removing it from the ContainerControl, so if you absolutely must use these older versions, you can acquire the non encrypted version of the class to do so.

Limitations

If you store a lot of text in a styled TextArea, the styled text will not resize past 15K. This is here to work around the limitations of Xojo RTFData parsing, which hangs past that size. If you really need to us very large amounts of styled text in TextArea, you should consider using RTFDataMBS which is cross platform, or some declares on Mac. And manage the resizing yourself eventually. Although managing StyleRuns is very time consuming, and there are practical limits to how much a user is able to wait.

One control seems to do strange things when resized : Image Well. Under versions older than 2015R1, it suddenly shows the picture upside down at random. The bug appears to be a Xojo bug, so we are unable to mitigate it. If you need to display pictures, better use the Canvas control under versions affected by the bug.

Each platform has it's own limitationss. On Mac, the height of some controls cannot be increased. For instance PopupMenus. As a result, when scaling up, such controls content may exceed the available room. The solution is to add the control name to the *noContentResizes* property.

On Windows, apart from the flicker issue, there are still some bugs in GroupBoxes, CheckBoxes , RadioButtons and Sliders that take the color of the window, so they do not look good over a backdrop picture. Unfortunately, RubberViews cannot mitigate this issue. With some luck, Xojo will fix that in a future version. RubberCanvas is a nice solution to have automatic resize without major rewrite on the Paint event. It works great for pictures and lines. However, text is not rendered very well under Windows. Better use a label laid over the control to display text, so you will benefit from the font full resolution and system antialiasing.

RubberViews resizes and positions all current controls. If Xojo was to add a new one, we will make sure to issue an update that takes care of it. All subclasses based on regular controls are managed, like for instance RubberCanvas. So most custom controls should behave fine. If, however, you have a control based on something else entirely, RubberViews will not be able to manage it. To have it position, you can place it over a parent canvas. If you needed often that custom control, better acquire the non encrypted class and add it there.

We welcome all suggestions to improve RubberViews. Send an email through the dedicated web site http://RubberViews.com

Michel Bujardet

Match Software

2207 Concord Pike #816

Wilmington, DE 19803

http://Match-Software.com

Properties of RubberViews

Name	Type	Description
Ignore	String	List of the controls that must not be changed or moved by RubberViews.
KeepRatios	String	List of the controls that must keep their aspect ratio when all the controls are resized. Contrary to the SizAll Boolean parameter below, it lets you specify which controls should keep their aspect ratio.
NoContentResizes	String	List of the controls that must keep their content untouched when all the controls are resized. Contrary to the NoContentResize SizAll Boolean parameter below, it lets you specify which controls should keep their content the same.
StretchWindowBackdrop	Boolean	When True, the window backdrop image will extend on all the surface of the window, stretching if necessary to match the aspect ratio.
Version	String	RubberViews current version.
ReleaseNotes	String	The history of this development

Methods and parameters

Name	Type	Description
Init(Win as Window, Width as Integer, Height as Integer)		This method must be called to initialize RubberViews so it knows all the controls sizes and positions.
Win	Window or ContainerControl	The window or ContainerControl onto which an instance of RubberViews has been dragged to manage controls size and position.
Width	Integer	The width of the window at design time
Height	Integer	The height of the window at design time
SizAll(Win as Window, NoContentResize as Boolean, KeepRatio as Boolean)		This method resizes all the contents and sizes when the window is resized, maximized, or resizing.
Win	Window or ContainerControl	The window or ContainerControl onto which an instance of RubberViews has been dragged to manage controls size and position.
NoContentResize	Boolean	When True, all the content of controls, text or pictures, is not resized.
KeepRatio	Boolean	When True, all the content of controls are resized and positioned to match the window size change, but their aspect ratio remains the same.