Creating VSTGUI Custom Views 4: **Custom View Objects for any VSTGUI4 Object**

Will Pirkle

In the last module, you learned how to create Custom Views for the *CViewContainer* and how to cache object pointers. In this module, you will learn how to make Custom Views with *CView* for creating VST-GUI4 objects that are not included in the RackAFX GUI Designer. These include the following views and controls:

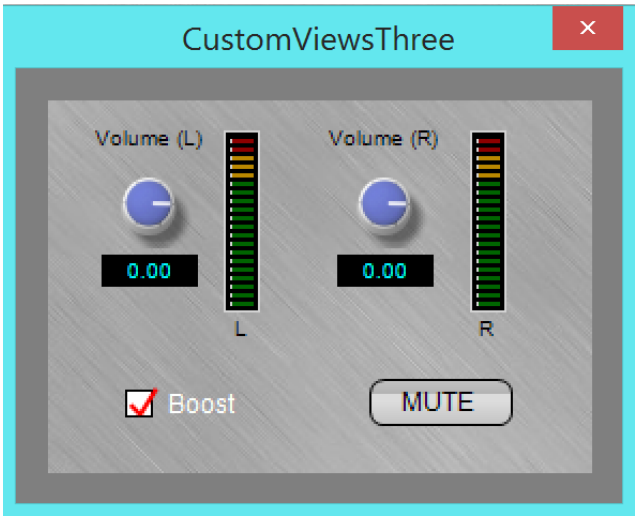
Views:
CAnimationSplashScreen
CGradientView
CRowColumnView
CScrollView
CSplitView
COpenGLView
CSplashScreen

Controls:
CSegmentButton (for making radio-button banks)
CCheckBox
CKnob
CMovieButton (uses CMovieBitmap)
CRockerSwitch
CTextButton
CSpecialDigit

There are also two more types of view containers that you can implement by creating Custom Views for the *CViewContainer* object:

View Containers:
CLayeredViewContainer
CShadowViewContainer

There are some very cool and powerful objects you can use in your advanced plugin GUIs including *COpenGLView* and *CShadowViewContainer.* In this module, we will create the *CCheckBox* and *CTextBut-ton* objects to give simple examples of creating these non-RackAFX GUI Designer items.

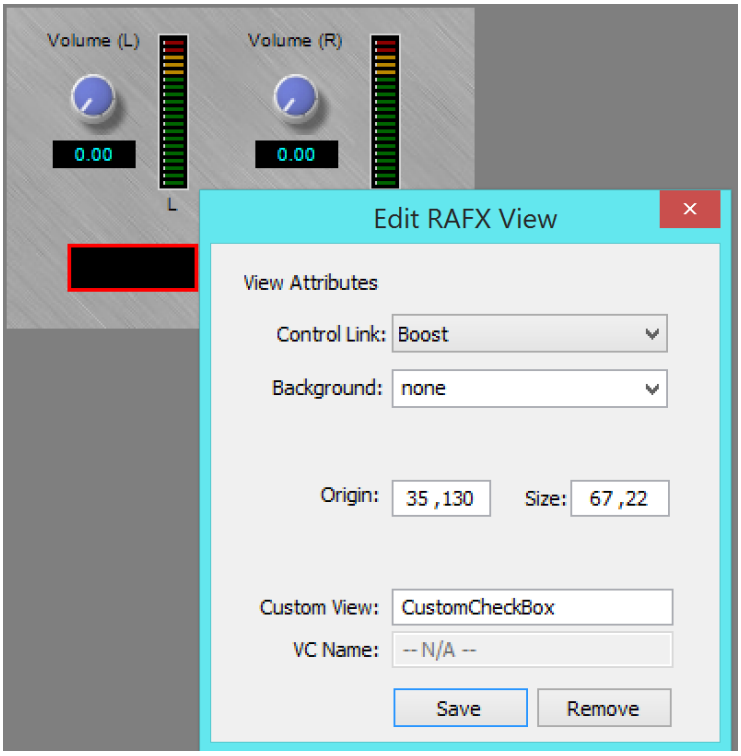


Here is the final GUI for the **CustomViewsThree** project that accompany this module The CCheckBox implements the boost function and the CTextButton implements mute.

These are neat and simple objects because they are self-drawing — they do not require graphics to render the control.

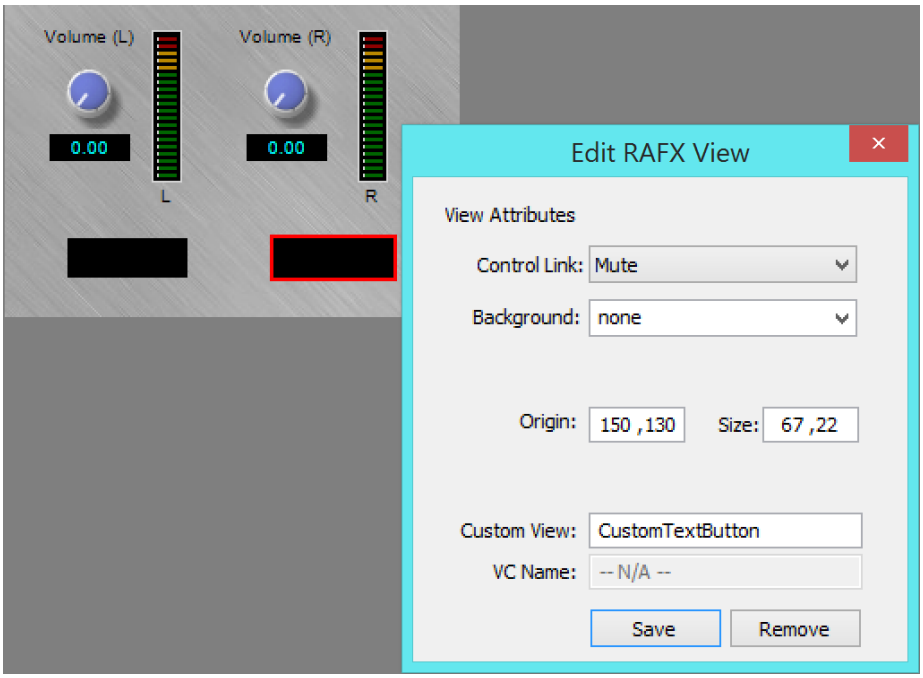
CustomViewsThree

The CustomViewsThree project accompanies this module. In the GUI Designer you can see that I have dragged two instances of *CView* onto the GUI and assigned both Custom View names as well as control tags — this is the first time we've added a control-tag to a *CView*. The control-tag will only have meaning if the Custom View you assign is derived from *CControl* — all the control-type objects are derived from *CControl*, which is subclassed from *CView*. I assign the boost tag to the check box, and the mute tag to the text button.



The views appear as black boxes in the GUI Designer and will stay that way as the rendering of the controls occurs in your plugin.

Both CView's are assigned Custom View names and control-tags.



There is no reason to cache the view pointers in this project so that simplifies things greatly. All you need to do is implement the custom view creation in *showGUI()* for the two controls. Notice that the text for each control is set in the constructor. The *CTextButton* constructor also includes a constant that defines the behavior as either on/off or kick-style and these are defined in:

```
..\vstgui4\vstgui\lib\controls\cbuttons.h

class CTextButton : public CControl
{
public:
	enum Style ///< CTextButton style
	{
		kKickStyle = 0,
		kOnOffStyle
	};

	etc…


case GUI_CUSTOMVIEW:
{
	if(info->customViewName.compare("CustomCheckBox") == 0)
	{
		// --- get the needed attributes with the helper
		const CRect rect = m_GUIHelper.getRectWithVSTGUIRECT(
                                                    info->customViewRect);

		// --- construction: note that Boost text is set here
		CCheckBox* pCheck = new CCheckBox(rect,
                                    (IControlListener*)info->listener,
                                     info->customViewTag,
                                     "Boost");

		// --- return control cloaked as a void*
		return (void*)pCheck;
	}
	if(info->customViewName.compare("CustomTextButton") == 0)
	{
		// --- get the needed attributes with the helper
		const CRect rect = m_GUIHelper.getRectWithVSTGUIRECT(
                                                    info->customViewRect);

		// --- construction: note that Mute text is set here
		CTextButton* pTbutton = new CTextButton(rect,
                                   (IControlListener*)info->listener,
                                    info->customViewTag,
                                    "MUTE",
                                    CTextButton::kOnOffStyle);

		// --- return control cloaked as a void*
		return (void*)pTbutton;
	}
	return NULL;
}
```

Final Thoughts

With this technique you can create any of the VSTGUI4 objects you wish — for the two extra view containers, you would need to make Custom Views for the *CViewContainer* object, not the *CView*. This allows you to extend the RackAFX GUI Designer to the same level as the VSTGUI designer found in all VST3 clients (and upon which the RackAFX GUI Designer was based). So far, we have only created Custom Views for existing VSTGUI objects or simple subclassed versions. In the next module, we will subclass *CView* and make a completely custom object that displays a scrolling waveform, unlike any of the built-in VSTGUI4 objects.

References:

VSTGUI4 Files and Documentation: http://sourceforge.net/projects/vstgui/