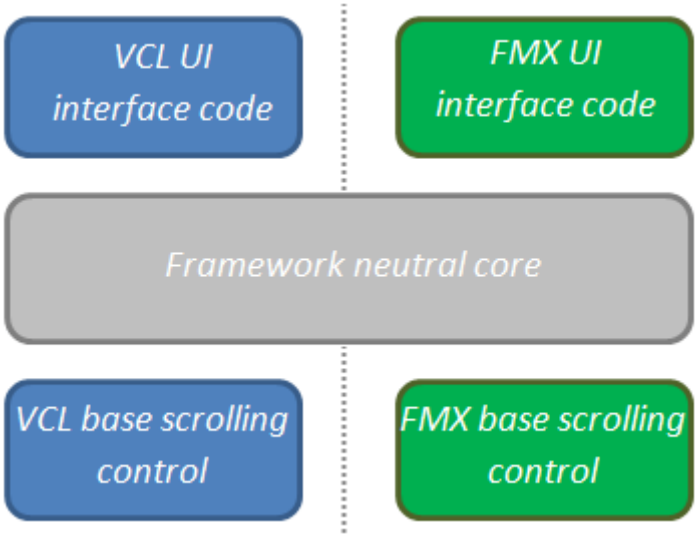
WYSIWYG formatted text & images editor for VCL & FireMonkey

Background

The standard rich text editor that ships with Delphi for over 15 years now is based on the Windows riched20.dll library control. While Microsoft introduced meanwhile newer versions 3.0 and 4.1 of this rich editor control, the VCL version continued to be based on a wrapper of the v2.0. This means that the feature set of this rich editor control is quite limited. On the other side, in the Delphi cross platform framework FireMonkey, there is no edit control for rich text at all. The only editor control that is available there in the framework is TMemo. As many of our customers requested both more flexibility from a Windows VCL based rich editor and also a rich editor for FireMonkey, the TMS team decided to take on these projects. From the start, we want to create a control for FireMonkey that would fit neatly the FMX paradigm of one source code to rule all platforms (Windows, iOS, Android, Mac OSX at this time) but realizing this would be a huge effort, we wanted that this effort would also be usable for VCL developers targetting Windows only.

Architecture

After a lot of research was done on performance (as we knew from our experience with other FMX controls this would be difficult) we decided on what we internally disrespectfully call a sandwich architecture. This sandwich architecture should enable us to isolate a core formatted text handling layer from the framework, i.e. the VCL and FireMonkey framework. The benefit clearly is that work on the core only needs to be done once and further evolutions of the core and its maintenance automatically benefits both VCL and FireMonkey framework versions. Visually, this can be seen as:



The core middle layer not only deals with rendering the formatted text and images but also with all the manipulations that can happen with it. Non-trivial areas in this core middle layer abstraction are: dealing with font handling, coordinates and images that are all treated in a different way in VCL and FireMonkey frameworks. Between this architecture and its final implementation is of course lots and lots of hard work both in development and testing.

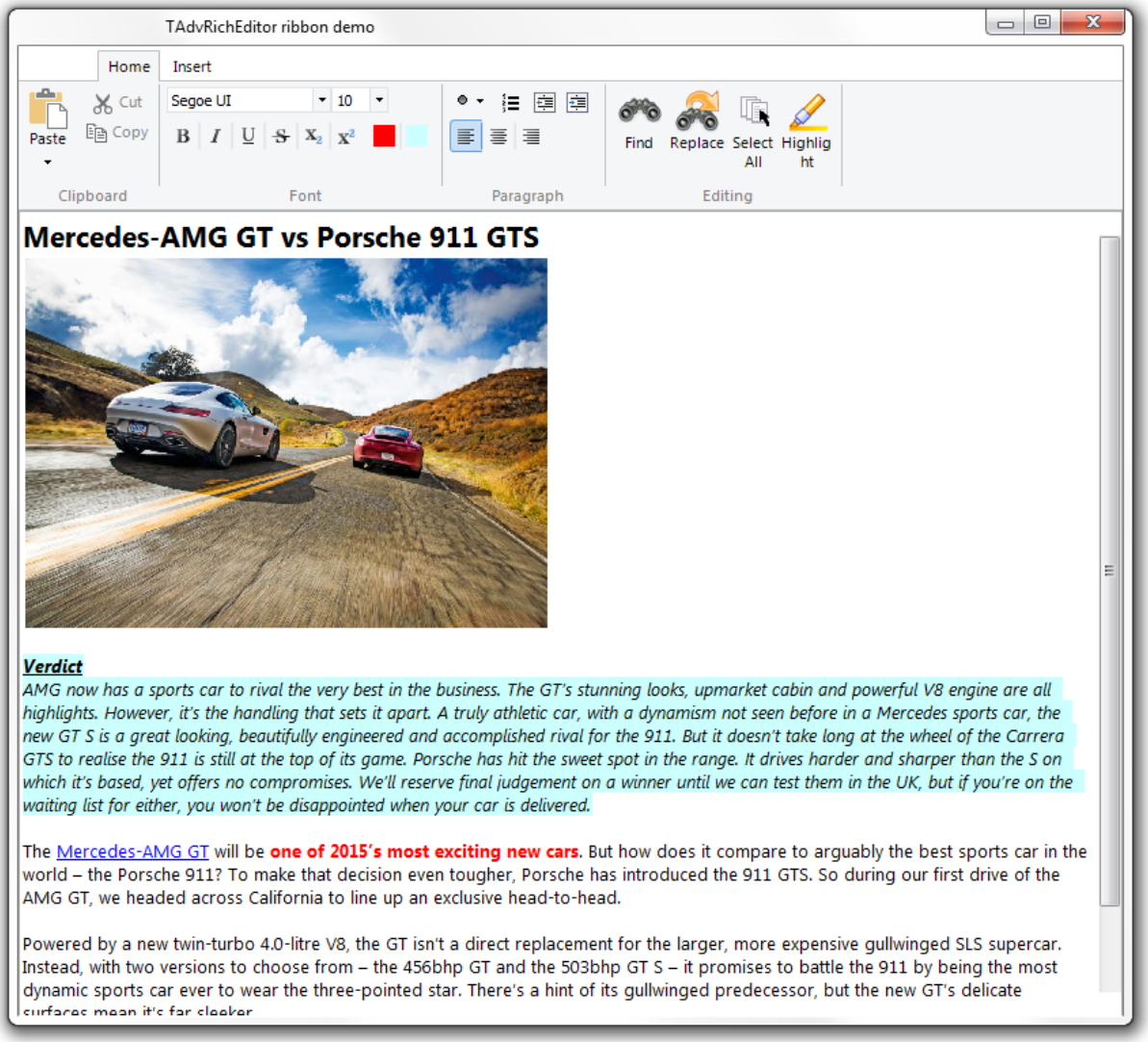
Feature set

For the first iteration of the new rich editor controls, we decided on the following feature set:

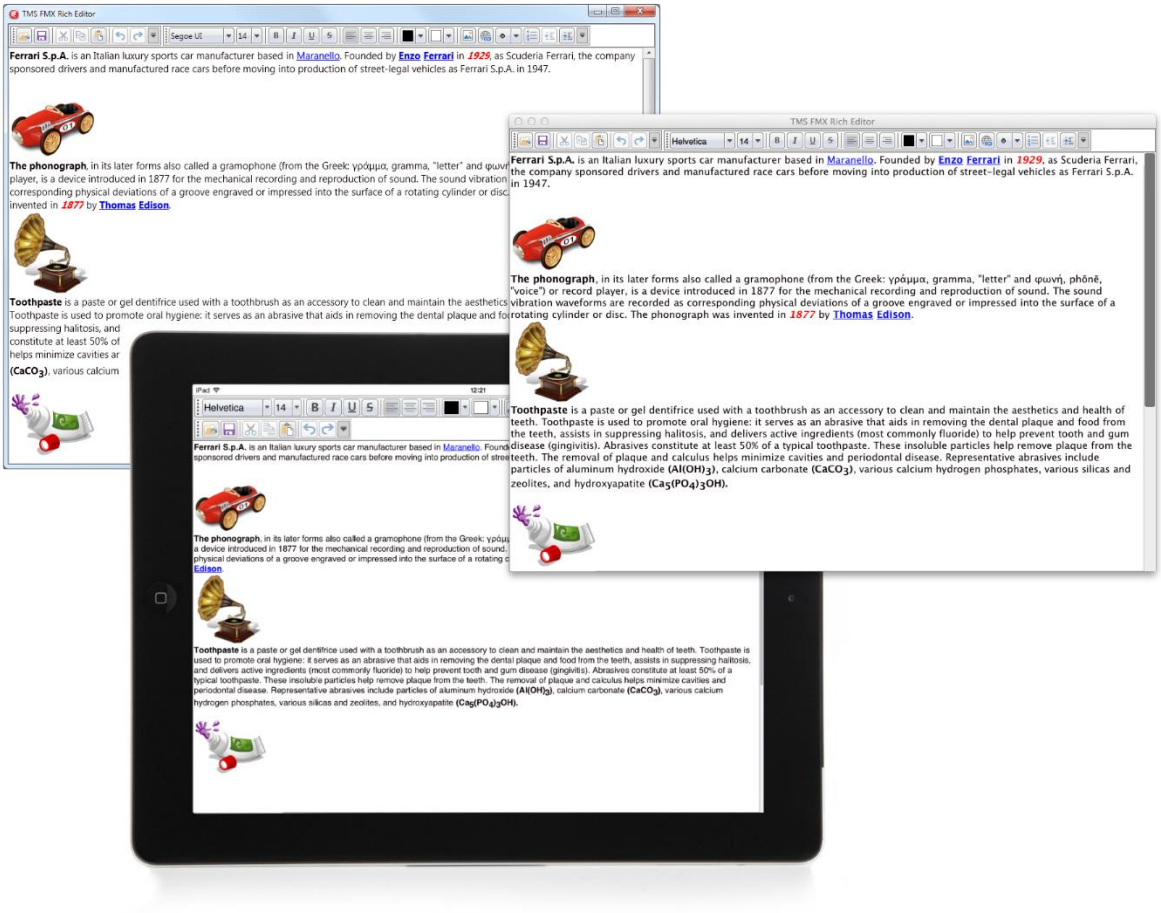
- Compact light-weight WYSIWYG editor for formatted text
- Supports formatted text with bullets, hyperlinks, indenting
- Paragraphs with left, center, right alignment
- Support for images, plain text and formatted text on clipboard
- Images (BMP, JPEG, GIF, PNG) natively built-in
- Custom graphics elements / owner draw elements can be embedded
- Undo/redo support
- Find & replace + text highlighting
- Mailmerge fields, mailmerge function
- Printing (on Windows)
- Exports to .TXT, .RTF, .HTML files.
- Many action classes to minimize work on building GUIs around the control
- Includes different toolbars (docking, ribbon, regular) ready to use for building GUIs

Of course, the team has a lot more ideas and already started work on the next iterations of this rich editor control and as always, we are of course listening to all your feedback & requests to steer this further development.

VCL TAdvRichEditor with ribbon toolbar UI also included



FireMonkey versions TTMSFMXRichEditor running on 3 platforms: Windows, Mac OS-X, iOS:

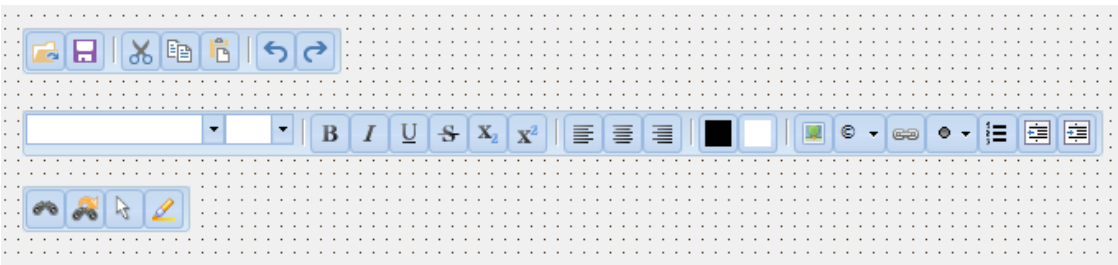


Getting started

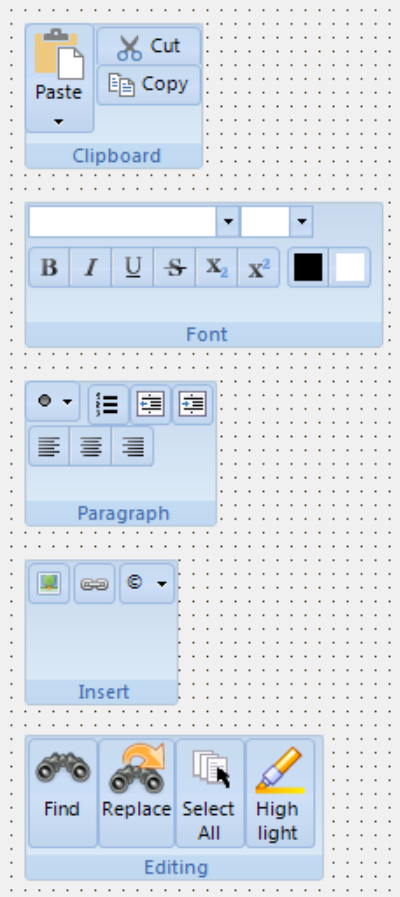
Drop a TAdvRichEditor (in VCL) or TTMSFMXRichEditor (in FMX) on the form. The component with its default settings is ready for use. Entering of text can be done with default font & alignment.

In VCL, for ease of use, connect a TAdvRichEditorEditButtonBar and TAdvRichEditorFormatButtonBar that presents most of the built-in actions as a button bar or use the docking toolbars TAdvRichEditorEditToolBar, TAdvRichEditorFormatToolBar, TAdvRichEditorEditingToolBar, TAdvRichEditorParagraphToolBar to apply all kinds of formatting to the text without writing any code or use its ribbon equivalents for a WYSISWYG editor with ribbon UI.

Included docking toolbars for VCL



Included ribbon toolbars for VCL



In FireMonkey, drop a TTMSFMXRichEditorEditToolBar and TTMSFMXRichEditorFormatToolBar on the form and connect these to the TTMSFMXRichEditor and you have a ready to use rich editor without writing a single line of code.

Included toolbars for FireMonkey:

Actions

If you want to hookup the various possible interactions with the rich editor to your own UI or UI controls, you can use the many actions that come with the rich editor:

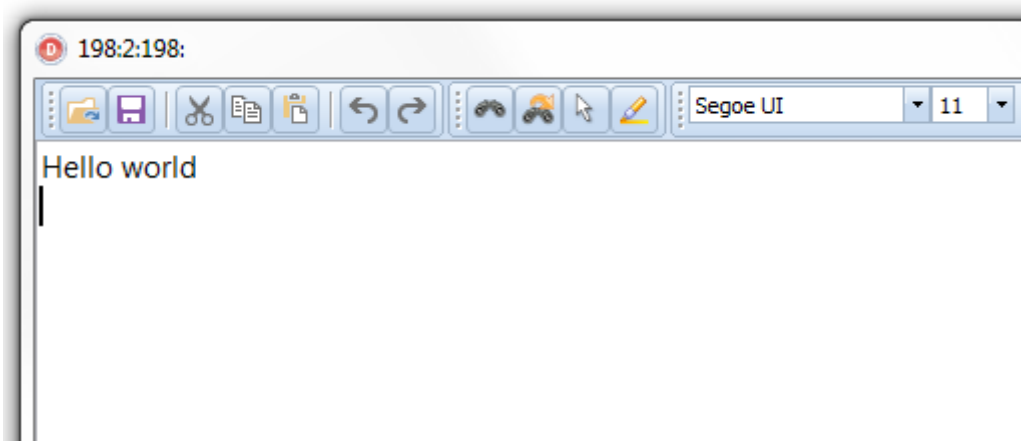
TAdvRichEditorClear, TAdvRichEditorCut, TAdvRichEditorCopy, TAdvRichEditorPaste, TAdvRichEditorSelectAll, TAdvRichEditorAlignRight, TAdvRichEditorAlignCenter, TAdvRichEditorAlignLeft, TAdvRichEditorBold, TAdvRichEditorItalic, TAdvRichEditorUnderline, TAdvRichEditorStrikeOut, TAdvRichEditorSubScript, TAdvRichEditorSuperScript, TAdvRichEditorTextColor, TAdvRichEditorFontName, TAdvRichEditorFontSize, TAdvRichEditorBulletType, TAdvRichEditorNumberedBulletType, TAdvRichEditorColor, TAdvRichEditorIndent, TAdvRichEditorUnIndent, TAdvRichEditorUndo, TAdvRichEditorRedo

Programmatic access

The API for adding and manipulating formatted text in the rich editor controls is exactly the same for the VCL version as for the FireMonkey version. When referring to RichEditor in this section, this can as such refer to both TAdvRichEditor (VCL) as to TTMSFMXRichEditor (FMX).

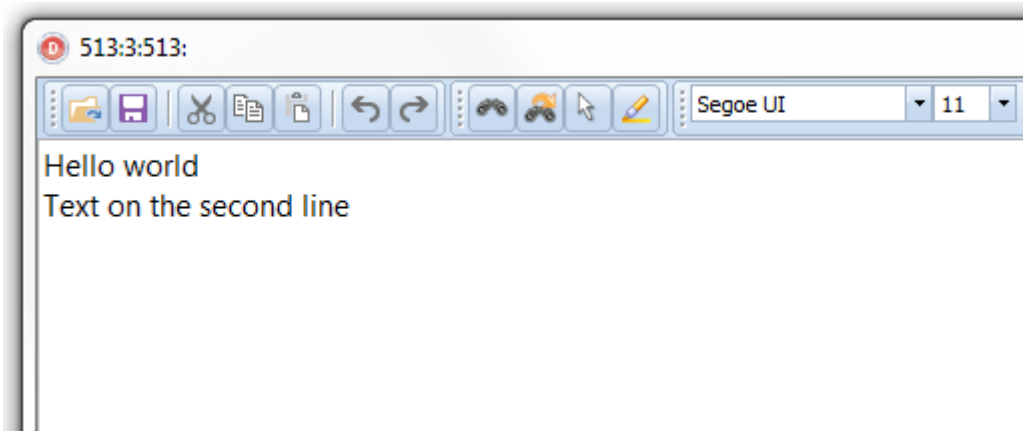
Text can be inserted in RichEditor in various ways. To start with call:

```
RichEditor.AddText('Hello world');
```



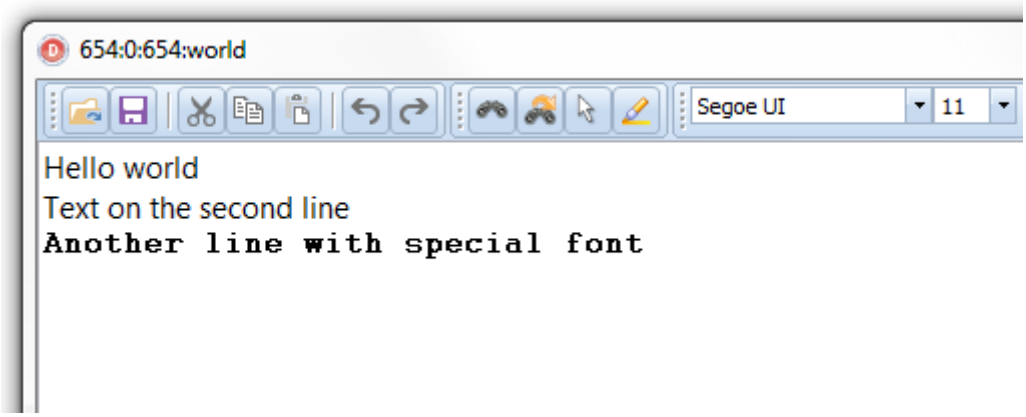
Add text on the next line with:

```
RichEditor.AddLineBreak;
RichEditor.AddText('Text on the second line');
```



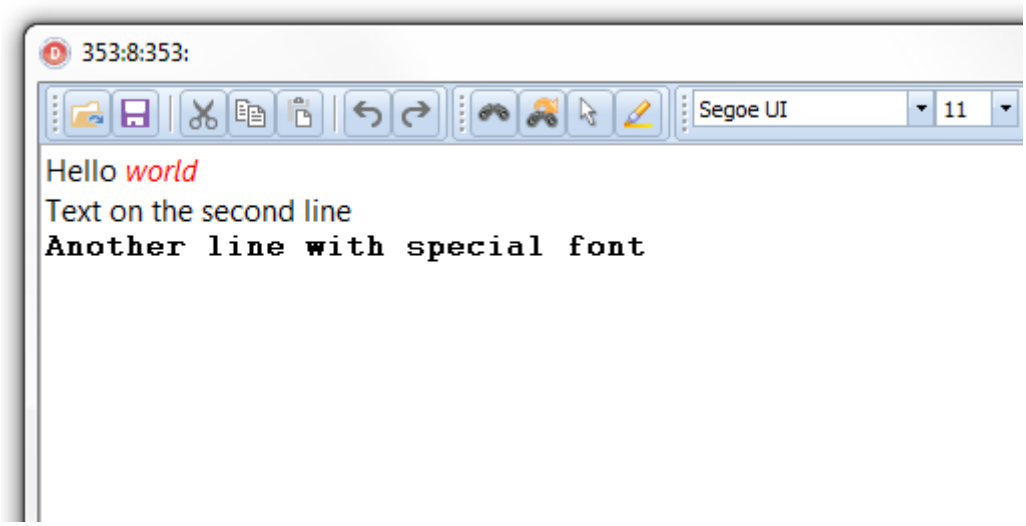
To add text with a different font than default font, use:

```
RichEditor.AddLineBreak;
RichEditor1.AddText('Another line with special
font',12,'Courier',[fsBold]);
```



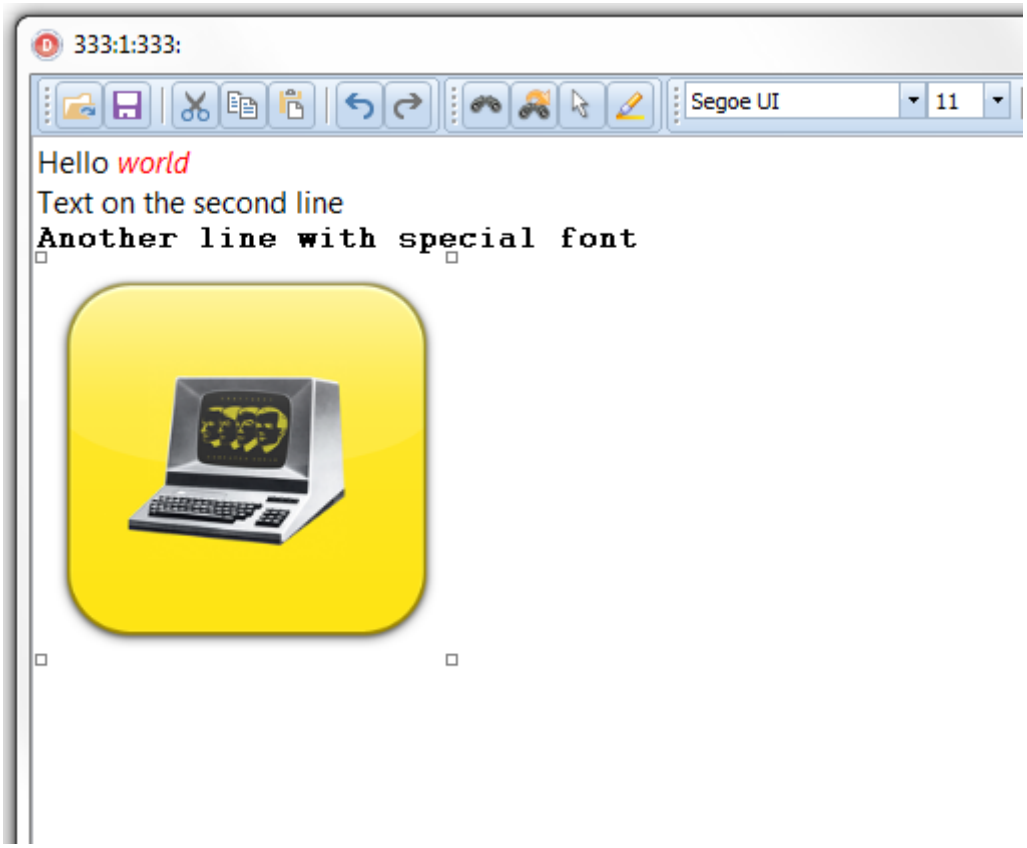
To change attributes of text in the RichEditor, perform a selection based on index of the text and length. For example, to change the color of “world” on the first line, set a selection from character 6 for 5 characters (character index starts at zero) and set an attribute for the selection followed by remove the selection itself:

```
RichEditor1.SelectText(6,5);

RichEditor1.SetSelectionColor(PlatformRed);  //PlatformRed = clRed
for VCL and claRed for FMX

RichEditor1.SetSelectionItalic(True);

RichEditor1.ClearSelection;
```



To add images to the RichEditor, use:

```
RichEditor1.AddImage('.\sample.png');
```



Persisting and exporting formatted text from the rich editor controls

Clipboard

From TAdvRichEditor or TTMSFMXRichEditor, there is built-in support to copy its text to the clipboard in 2 ways: plain text and rich text (RTF). As such, you can copy all text or a selection of text from the TAdvRichEditor or TTMSFMXRichEditor and paste it in popular applications such as MS Word or MS Outlook with its formatting and images applied.

Persist in file or stream

The rich editor controls offers the methods SaveToFile(), SaveToStream() and equivalent LoadFromFile(), LoadFromStream() to persist the formatted text and images to a binary file or binary stream. This is a proprietary binary format.

Export to RTF or HTML file

For generating an RTF or HTML file from TAdvRichEditor or TTMSFMXRichEditor, two non-visual exporter components are offered: TAdvRichEditorHTMLIO and TAdvRichEditorRTFIO (for VCL) TTMSFMXRichEditorHTMLIO and TTMSFMXRichEditorRTFIO (for FireMonkey). To use these controls, simply drop the non-visual control on the form and assign the rich editor control to the property RichEditorHTMLIO.RichEditor or RichEditorRTFIO.RichEditor. Then you can simply call

RichEditorRTFIO.Save(FileName);

RichEditorHTMLIO.Save(FileName);

Notice that for HTML export, the default behavior is that all images used in the document are exported as separate linked image files in the same folder where the .HTML file is generated. If it is preferred that images are generated in a different folder, use the 2nd default parameter ImagePath:

```
RichEditorHTMLIO.Save(FileName, ImagePath);
```

Conclusion

With the new VCL and FireMonkey rich editor controls, the TMS team tried to offer a comprehensive component set to empower you to add a formatted text / images editor in your traditional VCL Windows applications or cross platform FireMonkey applications running on Windows, Mac OS-X, iOS, Android with a minimum effort, often barely writing any line of code.

More information, trial downloads, documentation can be found at:

VCL version
http://www.tmssoftware.com/site/advricheditor.asp

FireMonkey version
http://www.tmssoftware.com/site/tmsfmxpack.asp?s=fmxricheditor#features