Biochemfusion Rendering Tools

Usage and build instructions

2011-01

This document describes how to incorporate the Biochemfusion rendering tools into your own applications or web pages.

A change log can be found at the end of this document.

1 Renderers

A renderer is a class or function that translates Biochemfusion rendering info text into on-screen graphics. Biochemfusion rendering info text is specified in the Biochemfusion Rendering Info format specification which you may download from the Biochemfusion web site.[1]

Renderers have been implemented in Delphi/Free Pascal[2], Microsoft .NET, Adobe Flash, Java, and JavaScript. You will find a directory for each of these implementations that holds the source code and example build scripts.

In the `test/` directory you will find HTML test pages that show how each renderer implementation handles rendering info text examples. The pages `moltest.html` and `seqtest.html` test molecule renderers and sequence renderers respectively.

2 HTML encoding

The characters percent, colon, and comma ('%', ':', ',') are already percent-encoded[3] in the rendering info text produced by Proteax in order to simplify parsing.

When embedding rendering info text in an HTML page you may have to do an additional layer of encoding to ensure that the rendering info text is passed correctly to the underlying implementation. Each implementation description will have a brief section about HTML encoding.

3 Implementations

3.1 Delphi / Free Pascal

The Delphi version may be compiled with Delphi or Lazarus/Free Pascal. The code was developed using Delphi 2007 but should work with older Delphi versions too. It has been successfully tested with Lazarus IDE v0.9.28.2-0 beta on Linux Ubuntu 9.04.

The example projects are `RenderPascalMolGraphics.dpr` and `RenderPascalSeqGraphics.dpr` that produce a bitmap file from an input file containing rendering info text. The applications created by these projects are used by the script that builds the HTML test pages.

To compile the example projects on Linux use Lazarus's "Convert Delphi Project" button and the .DPR files should convert smoothly to .LPI files. Before you can compile an .LPI file, open the

Project Inspector and add the LCL package as a Required Package. Then the compile should go smoothly too - with a single warning that "*APPTYPE is not supported by the target OS*".

The renderer classes draw their output on a normal `TCanvas` graphics abstraction. In that respect they can be used in any context - not necessarily in GUI applications. The graphics can be drawn on e.g. an off-screen `TBitmap` (as in the example projects) or on a visual component's canvas.

Scaling

The molecule renderer's graphics size is controlled by the `ZoomFactor` passed to it.

The sequence renderer's graphics size is controlled by the size (and type) of the font assigned to the canvas. A larger font will produce larger graphics.

Both renderers can return the graphics bounding box dimensions in pixels through the `BoundingBox()` function.

HTML encoding

Not applicable.

3.2 .NET

C# code is found in the `dotnet/` directory. As for the Delphi version two example solutions `RenderDotNetMolGraphics.sln` and `RenderDotNetSeqGraphics.sln` generate bitmap files from an input file containing rendering info text.

The code was developed using Visual Studio 2005 and should run on any Windows system that has a .NET 2.0 runtime. The code has very few dependencies and may thus compile and work with Mono[4] but no attempt has yet been made to test or verify this.

The code draws graphics on a standard .NET System.Drawing.Graphics abstraction. It has no dependencies on Windows.Forms or any particular GUI framework.

Scaling

The scaling of the .NET renderers works the same way as the Delphi renderers above.

HTML encoding

Not applicable.

3.3 Flash

The Flash version has been created in ActionScript 3. Adobe's Flex Builder 2 SDK was used to build the final .SWF files. Example build scripts can be found in `make_mol.bat` and `make_prot.bat`.

Scaling

Both the molecule and the sequence renderer will scale the graphics to fit the size of the stage area. The sequence renderer will however only scale the width of the graphics and will add a scrollbar to accomodate a larger height. The molecule renderer has no such abilities at present.

HTML encoding

Plus-signs, ampersands, and double quotes need escaping to survive all the way from the web page to the Flash component.

Double quotes can be encoded as either '"' or '%22'. Ampersands can however *not* be encoded as '&' - they must be percent-encoded as '%26'. So for consistency's sake it is recommended to use only percent-encoding when passing data to the Flash components.

For an encoding example, see `moltest.html` line 29.

3.4 Java

The Java version builds with a JDK version 1.5 or higher.

The example applets are the simplest possible wrappers that enable the renderers to be used in a web page.

Scaling

The molecule renderer will scale its vector graphics to `zoomFactor`. The renderer assumes that the caller will have scaled the font size to match `zoomFactor`.

The protein renderer will scale its graphics to match the font size of the graphics context. This is the same behavior as the Delphi and .NET renderers.

Both type of renderers have a `boundingBox()` function that returns the dimensions of the graphic bounding box in pixels.

HTML encoding

When passing rendering info text to a Java applet via a `<param>` tag double quotes need to be encoded as '"' (percent-encoding them as '%22' does *not* work). Otherwise no additional encoding seems to be necessary.

For an encoding example, see `moltest.html` line 34.

3.5 JavaScript

The JavaScript renderers require a browser with Canvas support. This means most current browsers except Internet Explorer. Firefox, Chrome, Safari, and the browsers on iPhones and Android phones should be OK.

The renderers draw the graphics on the canvas element that has the id given by the `canvas_id` parameter.

Scaling

The molecule renderer will scale both its vector graphics and font to the `zoomFactor`.

The protein renderer graphics size is controlled by the `font_height` parameter which sets the font size used to draw the sequence.

If you set the `scale_canvas` parameter to `true` the canvas element will have its dimensions reset

to match the bounding box of the graphics.

HTML encoding

Assuming that you embed the data as JavaScript code, single or double quotes will need to be backslash-escaped.

This type of quote escaping is done in `make_mol_tests.py` line 23.

4 Building the HTML test pages

If you want to build the HTML test pages yourself follow these instructions. You will no doubt have to do minor changes to the build scripts to make the scripts match your local paths.

- Delete all files and directories in the `test/` directory - except for the two .PY and the two .HTML files.
- Compile the two Delphi example projects.
- Compile the two .NET example solutions.
  - Note that the Python scripts in `test/` per default uses the Debug version of the compiled .NET applications.
- Compile the Flash components if you want to update the .SWF files.
  - The `make_mol.bat` and `make_prot.bat` scripts do this.
- Compile the Java renderers and applets and copy the compiled classes to `test/`.
  - The `make_mol.bat` and `make_seq.bat` scripts do this.
- Go to the `test/` directory and run the Python scripts `make_mol_tests.py` and `make_seq_tests.py`.
  - The `moltest_data.js` and `seqtest_data.js` files with test data are generated.
  - The Delphi and .NET example applications are called to produce a bitmap file for each of the rendering info text examples. The produced bitmap files are then referenced by the HTML test pages.
- You should now be able to open the `moltest.html` and `seqtest.html` test pages and compare the renderer graphics.

5 Change log

2011-01-14

- Fixed a NumberFormatException thrown by the Java molecule renderer when pseudo atoms did not have associated chain numbers. This will be the case when Proteax generates molecule rendering info text directly from a V2000 condensed molfile. Ensured that all renderers will use a consistent pseudo atom coloring in this case.

2010-12-27

- First release.

6 References

(1) The Rendering Info format specification can be freely downloaded from http://www.biochemfusion.com/doc/Biochemfusion_RenderingInfo_formats_1.0.pdf.

(2) Free Pascal's web site is http://www.freepascal.org/. The Lazarus IDE can be found at http://www.lazarus.freepascal.org/.

(3) Percent-encoding is commonly used for URI data: http://en.wikipedia.org/wiki/Percent-encoding.

(4) The Mono project http://www.mono-project.com/Main_Page aims to provide an open source implementation of C# that is binary compatible with Microsoft .NET.