Sources

Here you can find all my published sources that you can use as libraries in your projects. Only components for Delphi are on another page. You can use, change, and distribute all sources published under the terms of the MPL, GPL, or BSD-license in any way you want. Although in the case of the MPL or GPL, you must publish changes and you must not change the license. In the case of the GPL, you must also publish extensions and the source of all (your) programs using it under the terms of the GPL. If any of the files is published under the terms of more than one license, you can choose one of them. You can find more and legally valid information in the source files.

Title	Description	Language	License
Internet Tools	An XPath 2/3/XQuery 1/3/JSONiq engine, a pattern matcher for HTML/XML, an auto update class, and an HTTP connection wrapper.	FPC	GPL
Farb-auswahl-dialog	A color select dialog with HLS- and RGB-mode.	Delphi	GPL
HAMT	Mutable and immutable persistent maps and sets as a hash array mapped trie	FPC	(LC)LGPL
TCommand-LineReader	An unit for parsing of command line parameters	FPC, Delphi	GPL
Big Decimal Math	An arbitrary precision BCD floating point library	FPC	(LC)LGPL
Diagram-Tools	A diagram component based on a model/view system.	FPC/Lazarus	GPL
BBUtils	Often needed functions missing in FPC.	FPC	LGPL (+FPC)
Non-VCL Canvas	A replacement for Delphi's TCanvas not depending on the VCL.	Delphi	MPL, GPL
Diffreader	A unit to read and write unified diffs.	FPC	GPL
TCVirus	An unit/component for the development of joke programs	Delphi	MPL, GPL
TAuto-Updater	An unit for an automatic update	FPC	GPL
Simple-Browser	A simple IE-based browser	(Delphi), (EXE)	GPL
Huffman-komprimierer	A collection of functions for Huffman (de-)compression.	Delphi	MPL, GPL

Internet Tools

GPL

2006 - 2024

This package consists of several units working together:

- The *simple HTML parser* at the lowest level parses an XML/HTML like file and calls callback functions for every tag and text it has read.
- The *HTML tree parser* uses the simpler parser to read an XML/HTML like file and create a linked element-stream from it, which is equivalent to a dom-like tree. This parser takes much care to make the tree usable, e.g. by correcting errors in invalid HTML files and auto detecting its encoding.
- With the *XPath / XQuery engine*, you can then run arbitrary XPath 2.0, XPath 3.0, XQuery 1.0, and XQuery 3.0 queries on the dom-like representation. It is completely implemented in Pascal and pretty much standard conformant. (compare its XQuery Testsuite results: between 100% for XPath 2 and over 99% for XQuery 3). It also supports JSONiq.
- The parser for *CSS 3 selectors* is part of the XQuery class, converting the CSS selector to an XPath expression and evaluating that.
- The *pattern-matching HTML template engine* at the highest level can be used to process an HTML file by annotating a subset of it and applying the resulting template to the HTML file. All elements in the template are searched in the HTML file and template instructions in matching elements can be used to extract data. For example a template like

  ```
  <a>{@href}</a>*
  ```

  will read the URLs of all links in an HTML file.
  A template like this:

  ```
  <table id="foobar"><tr><td>abc</td><td>{.}</td>*</tr>*</table>
  ```

  will read all table cells after the cell containing abc in all rows of the table with id foobar. And a template like this:

  ```
  <table id="foobar"><tr>{temp := 0}<td>abc</td><td>{temp := $temp + .}
  </td>*{result := $temp}</tr>*</table>
  ```

  will calculate the sum of all table cells after the abc cell in all rows.
  You can also try these templates online or with Xidel.
- The *multi page template engine* can be used to download several pages, and apply a single-page template to each of them. Contrary to the single-page template it does not do any pattern matching itself (of course the single-page templates contained in a multi-page template do pattern matching against each page), but simply executes the actions in the order given in the template. It also supports variables, condition, and loops, so it is even Turing complete.
- The *HTTP/S-wrapper* lets you switch dynamically between Wininet (default library installed on every Windows computer), Synapse (platform-independent, but not always installed and problematic with HTTPS), Apache HttpComponents, and OkHttp (latter two called through JNI for Android).
- The *auto update* class is a simple example for the other classes and can be used to add an auto update function to a program which will check online for a more recent version, download that version automatically, and (optionally) install it.

Summary of features:

- **Parsers/interpreters:**
  - for XML/HTML files, with a sax or dom-like interface
  - for XPath / XQuery expressions
    - standard compatible: passes over 99.9% of the XPath 2 only tests and 99.5% of the XQuery 3.0 tests in the XQuery Testsuite
    - Some extensions for objects, CSS selectors, regular expressions, pattern matching and JSONiq.
  - Some syntax from XPath/XQuery 3.1.
  - for JSONiq: An extension to XQuery to handle JSON data
  - for CSS 3 Selectors
    - standard compatible (afaik)
    - standalone or within XPath queries `//foobar/css("p#x a.class")/@href`
  - for pattern-matching HTML/XML templates:
    - Templates process HTML like regular expressions process strings.
    - The template languages allow the inclusion of regular, XPath/XQuery, JSONiq, and CSS 3 Selectors expressions
    - The language has variables, optional elements, loops and conditions
    - e.g. `<p class="test"><span>Caption:</span><a>{.}</a>*</p>` will read all links after the `span`-element containing `Caption:` in a paragraph with class `test`
  - for multi-page "scraping" templates
    - A collection of patterns applied to several webpages.
    - has variables, conditions, and loops => Turing complete.
- **HTTP/S-Wrapper:**
  - Can be used for transmissions over HTTP and HTTPS
  - Allows arbitrary HTTP methods, e.g. GET, POST, PUT, ...
  - Supported platforms:
    - Windows: using WinINet with the standard system configuration
    - Android: using OkHttp or the Apache HttpComponents through JNI
    - Others (Linux, Mac): using the platform-independent Synapse
    - Offline: simulating an internet connection locally for unit tests
  - Supports HTTP, HTTPS and SOCKS-Proxy (depending on the platform)
- **Auto update:**
  - Uses an XML format to store the most recent version and a changelog on the server.
  - Supports different updates for 32/64-bit platforms and Linux, Windows or BSD systems.
  - Checks if a new version is available without downloading the whole, large changelog.
  - Allows the execution of an arbitrary command line after the update, runas administrator if necessary.

Mercurial repository

Online Documentation

Source on GitHub

Download (1665 KB)jump to top

discuss / share

Farbauswahldialog

GPL

2005

This color select dialog is similar to the default one, but has more options:

- All components of RGB and HLS are shown.
- You can change any component of RGB/HLS in a color gradient
- The selected color component is shown as a function depending on the other two components
- You can change the other two components in a two-dimensional gradient
- This one is drawn depending on the first component
- Additional you can enable a list of colors to change

Source on GitHub

Download (14 KB)jump to top
discuss / share

HAMT

(LC)LGPL

A HAMT is a hashmap/set stored as a trie, which provides an update and lookup performance similar to a normal hashmap/set, but needs no rehashing and also allows one to copy the entire map/set in constant time. This implementation uses a 32 bit hash and trie nodes with 32 children, so 5 bits of the hash are consumed to choose the next child. When there are no hash collisions, this HAMT can store 2^32 items with a maximal tree depth of (log_32 2^32) = 6, i.e., you need 6 memory accesses to find any key/value which is practically O(1). (When there are hash collisions, they are put in an array)

Each HAMT node carries a reference counter since FreePascal has no garbage collector. If the reference count is 1, the node can mutate, otherwise, it is immutable with a copy-on-write semantic like strings. The counter is updated atomically, so the map could be shared across threads.

Everything is implemented using generics, so it can be used with all types.

Example:

```
type TImmutableMapStringString = specialize TImmutableMap<string, string,
THAMTTypeInfo>;
var map, map2, map3: TImmutableMapStringString;
    p: TImmutableMapStringString.PPair;
begin
  map := TImmutableMapStringString.create;
  map2 := map.Insert('hello', 'world');
  map3 := map2.insert('foo', 'bar');

  writeln(map.get('hello', 'default')); // default
  writeln(map.get('foo', 'default')); // default

  writeln(map2.get('hello')); // world
  writeln(map2.get('foo', 'default')); // default

  writeln(map3['hello']); // world
  writeln(map3['foo']); // bar

  //enumerate all
  for p in map3 do
    writeln(p^.key, ': ', p^.value);

  map.free;
  map2.free;
  map3.free;
end
```

Online Documentation

Source on GitHub

Download (113 KB)jump to top
discuss / share

TCommandLineReader

GPL

This is a command line parser working with Delphi as well as with Free Pascal, and on Windows and Linux.
Before you can read the command line arguments, all allowed arguments need to be declared. So the parser can ensure that the user only uses allowed arguments and each given argument has the correct

type (e.g. int or string). Furthermore, the parser can automatically generate a help summary displaying all allowed arguments including a description. This again ensures that there are no hidden, undocumented command line arguments.
There are three different versions of the parser. The first two read the arguments from the command line and show the help list of all allowed arguments on stdout or respectively in a message box. The third version (fpc only) runs as a cgi service and reads the argument from GET/POST requests.

Here is a nice comparison between TCommandLineReader and the standard pascal command line functions, when the program is called as `executable '" a "'`

System	TCommandLineReader	ParamStr	string(cmdline)
Windows (Delphi 4)	`" a "`	`' a '`	`"executable" '" a "'`
Windows (fpc 2.6)	`" a "`	`' a '`	`executable '" a "'`
Linux/bash (fpc 2.6)	`" a "`	`" a "`	`executable "" a ""`

The feature list of rcmdline is:

- supports linux (`--xyz=bar` or `-x bar`) and windows (/) style parameters
- different input possibilities for boolean flags (`-f`, `--flag`, `--enable-flag`, `--disable-flag`)
- understands single (') and double (") quotation marks (even on Windows)
- type conversion of the string arguments to the declared types
- automatic user notification about wrong parameters
- existing files with names containing spaces can be parsed without quotation marks
- can run as a cgi service and answer to GET/POST requests (fpc only)
- Arguments can be read from a simple string (e.g. cmdline) or an array of arguments (e.g. paramstr)

Online Documentation

Source on GitHub

Download (45 KB)jump to top
discuss / share

Big Decimal Math

(LC)LGPL

2013, 2015

This unit provides a BigDecimal-Typ which stores arbitrary precision (BCD) decimal numbers and can be used like any native numeric type. For example:

```
var bd: BigDecimal;

bd := 12.34;
bd := bd * 1000 - 42;  // bd = 12298
bd := bd / 7.0;        // bd = 1756.85714285714286
```

```
  bd := StrToBigDecimal('12345678901234567890123456789') + 1; // bd
= 12345678901234567890123456790
```

Summary of features:

- Supports at least numbers between $10^{-19327352814}$ to $10^{19327352814}$ with up to 4831838208 decimal digit precision
- And all standard arithmetic and comparison operators
- The operators are overloaded, so it can be used like normal numeric types
- Rounding functions (floor, ceil, to-even, ..)
- Some more advanced operations, e.g. power and sqrt
- Also provides exact native float to string conversion functions
- Written in pure Pascal, therefore platform-independent and without dependencies
- Designed for correctness and simplicity (rather than speed)

Online Documentation

Source on GitHub

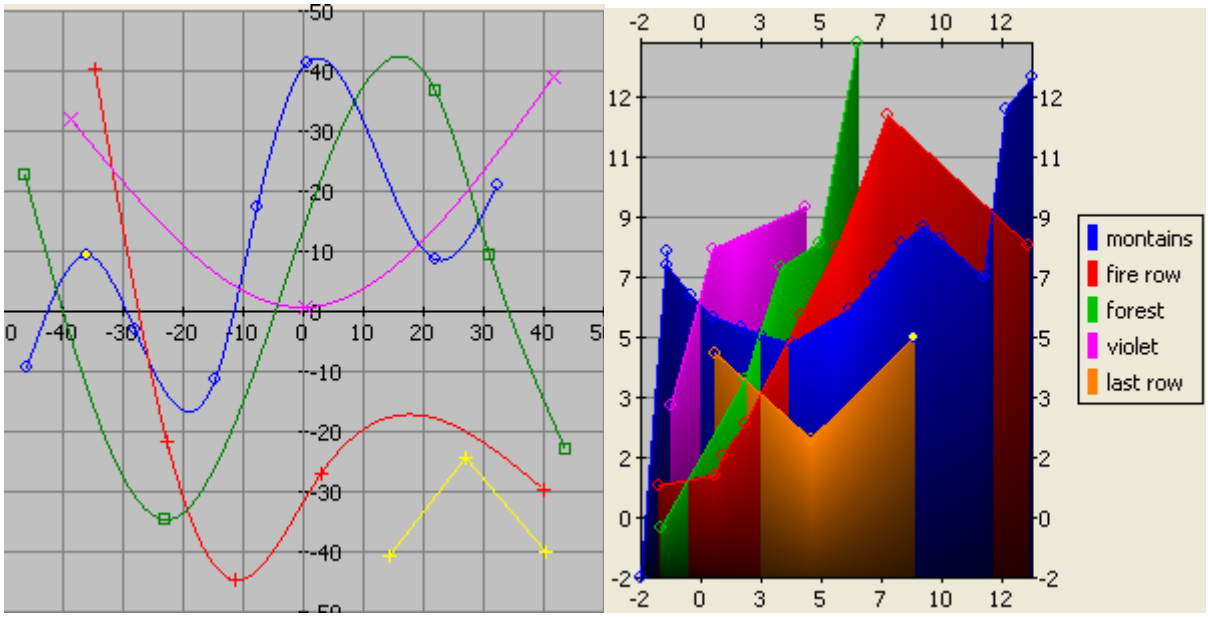
Download (50 KB)jump to top
discuss / share

Diagram-Tools

GPL

This diagram drawing unit follows a model/view concept like qt: There is a view component responsible to show the data, and a model class to store them.
This is more flexible than the traditional way, in which the data is handled within the viewing class.
Features:

- Model/view-concept with automatic synchronization between different views showing the same model
- Filling with horizontal/vertical color gradient and different line/point-styles
- Interpolation using linear lines or splines
- The diagram data can be modified by the user if wanted
- Multiple axes and point rows as well as a legend naming the latter
- Automatic calculation of the view range and support for floats
- Standard models with lists of points, cyclic lists, and a model to merge other ones
- There is a help file



Online Documentation

Source on GitHub

Download (64 KB)jump to top
discuss / share

BBUtils

LGPL (+FPC)

These units consist of low-level functions missing in FPC. For example:

- Various array functions (add, delete, search, prealloc, ...)
- Various string functions (pchar/ansistring, compare, split, utf8-functions, 1-bit/unicode encoding conversions, search, convert HTML entities, ...)
- Various date time functions, parsing/formatting supporting timezones, years before 0 and after 65535 (compatible to XML Schema 1.1, using the date time model of ISO 8601:2004)
- Stable sorting function for arbitrary sized data arrays (merge+insert sort)
- Mathematical functions (gcd, primes, Bernoulli statistics,...)
- Automatic translation of with tr['marked strings'] and components
- A Pascal template language which is "compiled" to Pascal (similar to the C preprocessor)

This unit has the same license as the runtime libraries of FPC, so anyone using FPC/Lazarus can use it without license issues.

A note about Delphi compatibility: This unit was written for Delphi 4 and most functions still work with Delphi 4. However, I no longer test new functions with Delphi, as I have no newer Delphi version and people are unlikely to still use Delphi 4. If you want to use it with Delphi, you can just remove the non-compiling functions.

Online Documentation

Source on GitHub

Download (233 KB)jump to top
discuss / share

Non-VCL Canvas

MPL, GPL

2003

This package provides a TCanvas replacement which does not depend on the VCL.
The most important features are:

- Functions to draw texts, ellipses, lines, rectangles and polygones.
- Functions to copying parts of pictures.
- Classes to set font, pen and brush properties.
- An int-int hash map implementation and a resources manager.

In some areas, they are even better than the functions of Delphi, for example, they can draw rotated text.
There is a good, German help, and you could use the documentation of TCanvas from Delphi.

Download (26 KB)jump to top
discuss / share

Diffreader

GPL

2009

You can use this unit to read a Unified Diff, modify its tree structure, and save it finally.

- Reads Unified Diffs of one or many files
- Creates valid diffs
- Has functions to remove unnecessary data (see also Simplify Diff)

Download (3 KB)jump to top
discuss / share

TCVirus

MPL, GPL

2000, 2003

These are the most important features of TCvirus:

- It gives you a canvas for direct drawing on the screen.
- It can mirror the screen.
- It returns handles of important objects like the task button, the task list, and the desktop.
- It can hide your program from the task manager (on Win9x).
- It can open and close the CD tray.

Because TCvirus has originally been a component, there are 2 units, a component for the VCL and a unit for non-VCL.

Download (60 KB)jump to top
discuss / share

TAutoUpdater

GPL

This allows an automatic update, downloaded from the internet.
To use the class you need a wininet unit containing the windows internet functions (above is a new class that uses a better customizable XML format).
Features:

- check for new versions
- download them in a compressed zip archive
- extract this archive
- replace the current executable with the new one on every Windows
- detect missing access rights on Windows NT...

Download (9 KB)jump to top
discuss / share

SimpleBrowser

GPL

2006

This is a small browser based on the Internet Explorer.
It is completely useless for a normal user, but you can use it to implement a browser for a certain web page in your program. The appearance and behavior can be controlled over command line switches, so it allows for example restricted access.

Download (188 KB)jump to top
discuss / share

Huffmankomprimierer

MPL, GPL

2003

This unit can (de-)compress strings with the Huffman algorithm. After compression frequently occurring characters are stored with less than 8 bit.
It is also possible to compress Unicode strings, arrays, and files, by treating them as byte strings.
This unit contains a copy of TBits from the 32-Bit VCL, which only works on 32 systems. On other platforms, the normal VCL can be used by enabling a define.
You get the best compression rates in data which contains many repeated characters, but it is usually worse than modern compressions like ACE or RAR.

Download (12 KB)jump to top
discuss / share
