Pixmap

The alpha channel is now optional. Its presence is controlled by a new boolean parameter (called `alpha`). This has the following consequences:

- The size of one pixel can be two different values. For e.g. colorspace RGB, this size may be 3 (no alpha) or 4 bytes. The size of a pixmap is therefore determined not only by its colorspace, but also by its alpha value.
- Handling of pixmaps needs to take the alpha parameter into account. An decision has to be made during pixmap creation. Pixmaps coming from other sources or generated by some functions may or may not contain an alpha, which therefore needs to be checked.
- In general, the alpha channel should be avoided to benefit from significant memory savings.
- The `savealpha` parameter disappeared: all methods saving pixmaps (`writePNG` and friends) now always store the complete pixmap.
- Pixmaps created from PDF-internal images may or may not contain an alpha – this solely depends on how the image was stored in the PDF.
- The `Pixmap` and `Colorspace` classes have been extended with properties that help determine their characteristics – see the following list:
    - `Pixmap.alpha` – bool
    - `Pixmap.stride` – integer containing the number of bytes of one line of the pixmap's IRect
    - `Pixmap.n` – integer containing the number of bytes per pixel
    - `Colorspace.nbytes` – integer containing the number of bytes used to determine the color (1 = `DeviceGray`, 3 = `DeviceRGB`, 4 = `DeviceCMYK`)
    - `Colorspace.name` – string naming the colorspace, one of `DeviceGray`, `DeviceRGB`, `DeviceCMYK`
    - `Pixmap.colorspace = Colorspace.name`
    - `Pixmap.n - Pixmap.alpha = Colorspace.nbytes`

PyMuPDF Design Decision

Where PyMuPDF constructors require an alpha parameter, we assume `alpha = False` by default.

API Change: Display List

Constructor now requires the page's mediabox.

API Change: Text Page

Constructor now requires the page's mediabox.

API Change: Links

This contains significant changes:

- Link destinations objects are no longer maintained by MuPDF and information about destinations in general has been reduced. The only two variables containing such information now are `isExternal` (a bool) and `uri`, a string.
- The base class for PyMuPDF's `linkDest`, `fz_link_dest_s` has been deleted from MuPDF. In order to maintain backward compatibility, PyMuPDF provides an own `linkDest` class from available information as closely as possible.
- Document `outline` now additionally also contains `page`, `isExternal` and `uri` properties.
- The following shows MuPDF's behavior concerning links and how we interpret this in PyMuPDF.

isExternal	uri	Links	Outlines
True	Starts with `file://`	If `uri` ends with a page number (format: `#page=n`), generate a `gotor`, else as a `launch`. Strip off prefix and suffix in any case.	Behave as described in Links column. Page must be `-1`.

isExternal	uri	Links	Outlines
True	Does not start with `file://` (may instead be `http://`, `https://`, `mailto:`, `ftp://` a probably incomplete list)	Generate a `launch`.	Generate a `launch`. Page must be `-1`.
True	None (empty)	Should not happen and will be ignored.	Generate a dummy outline entry. Page must be `-1`.
False	May contain a named destination or a 1-based page number (however, format varies by document type: e.g. `#n,x,y` or `#n` in case of PDF.	Generate a `goto`. If PDF and `uri` is in page number format, use page number to construct a direct destination. If coordinates `x,y` are provided, use them as `linkDest.lt`. If not a PDF or a different format, assume a named destination with `uri` as its name.	The outline structure contains `page` (0-based) for all document types. We use this (and potentially `x,y` coordinates) to construct the destination.

Configuration Changes

There is a new file `mupdf/include/mupdf/fitz/config.h` to be used for MuPDF generation. It contains `#define` statements for switching off unwanted features. Our recommendations are contained in the file version that follows.

Of course, everyone using MuPDF is free to decide what to generate for himself – PyMuPDF should work in any case. In so far the following is just a recommendation. It does, however, have an impact on the Windows binaries we generate – our decisions are “burnt in” into them.

Basically, we disable all but the standard fonts as before in v1.9.2.

You may disable JavaScript features, too, because they are not used in PyMuPDF. This will save you another 300 KB binary file size.

```
#ifndef FZ_CONFIG_H
#define FZ_CONFIG_H
/*
	Choose which plotters we need.
	By default we build the greyscale, RGB and CMYK plotters in,
	but omit the arbitrary plotters. To avoid building
	plotters in that aren't needed, define the unwanted
	FZ_PLOTTERS_... define to 0.
*/
/* #define FZ_PLOTTERS_G 1 */
/* #define FZ_PLOTTERS_RGB 1 */
/* #define FZ_PLOTTERS_CMYK 1 */
/* #define FZ_PLOTTERS_N 0 */
/*
	Choose which document agents to include.
	By default all but GPRF are enabled. To avoid building unwanted
	ones, define FZ_ENABLE_... to 0.
*/
/* #define FZ_ENABLE_PDF 1 */
/* #define FZ_ENABLE_XPS 1 */
/* #define FZ_ENABLE_SVG 1 */
/* #define FZ_ENABLE_CBZ 1 */
/* #define FZ_ENABLE_IMG 1 */
/* #define FZ_ENABLE_TIFF 1 */
/* #define FZ_ENABLE_HTML 1 */
/* #define FZ_ENABLE_EPUB 1 */
/* #define FZ_ENABLE_GPRF 1 */
/*
	Choose whether to enable JavaScript.
	By default JavaScript is enabled both for mutool and PDF interactivity.
*/
// #define FZ_ENABLE_JS 0  // <=== potential save of 300 KB
/*
	Choose which fonts to include.
	By default we include the base 14 PDF fonts,
	DroidSansFallback from Android for CJK, and
	Charis SIL from SIL for epub/html.
	Enable the following defines to AVOID including
	unwanted fonts.
*/
/* To avoid all noto fonts except CJK, enable: */
#define TOFU            // <=== PyMuPDF
/* To skip the CJK font, enable: */
#define TOFU_CJK        // <=== PyMuPDF
/* To skip CJK Extension A, enable: */
#define TOFU_CJK_EXT    // <=== PyMuPDF
/* To skip the Emoji font, enable: */
#define TOFU_EMOJI      // <=== PyMuPDF
/* To skip the ancient/historic scripts, enable: */
#define TOFU_HISTORIC   // <=== PyMuPDF
/* To skip the symbol font, enable: */
/* #define TOFU_SYMBOL */
/* To skip the SIL fonts, enable: */
#define TOFU_SIL        // <=== PyMuPDF
/* To skip the Base14 fonts, enable: */
/* #define TOFU_BASE14 */
/* (You probably really don't want to do that except for measurement purposes!) */
/* ---------- DO NOT EDIT ANYTHING UNDER THIS LINE ---------- */
#ifndef FZ_PLOTTERS_G
#define FZ_PLOTTERS_G 1
#endif /* FZ_PLOTTERS_G */
#ifndef FZ_PLOTTERS_RGB
#define FZ_PLOTTERS_RGB 1
#endif /* FZ_PLOTTERS_RGB */
#ifndef FZ_PLOTTERS_CMYK
#define FZ_PLOTTERS_CMYK 1
#endif /* FZ_PLOTTERS_CMYK */
#ifndef FZ_PLOTTERS_N
#define FZ_PLOTTERS_N 0
#endif /* FZ_PLOTTERS_N */
/* We need at least 1 plotter defined */
#if FZ_PLOTTERS_G == 0 && FZ_PLOTTERS_RGB == 0 && FZ_PLOTTERS_CMYK == 0
#undef FZ_PLOTTERS_N
#define FZ_PLOTTERS_N 1
#endif
#ifndef FZ_ENABLE_PDF
#define FZ_ENABLE_PDF 1
#endif /* FZ_ENABLE_PDF */
```

```
#ifndef FZ_ENABLE_XPS
#define FZ_ENABLE_XPS 1
#endif /* FZ_ENABLE_XPS */
#ifndef FZ_ENABLE_SVG
#define FZ_ENABLE_SVG 1
#endif /* FZ_ENABLE_SVG */
#ifndef FZ_ENABLE_CBZ
#define FZ_ENABLE_CBZ 1
#endif /* FZ_ENABLE_CBZ */
#ifndef FZ_ENABLE_IMG
#define FZ_ENABLE_IMG 1
#endif /* FZ_ENABLE_IMG */
#ifndef FZ_ENABLE_TIFF
#define FZ_ENABLE_TIFF 1
#endif /* FZ_ENABLE_TIFF */
#ifndef FZ_ENABLE_HTML
#define FZ_ENABLE_HTML 1
#endif /* FZ_ENABLE_HTML */
#ifndef FZ_ENABLE_EPUB
#define FZ_ENABLE_EPUB 1
#endif /* FZ_ENABLE_EPUB */
#ifndef FZ_ENABLE_GPRF
#define FZ_ENABLE_GPRF 0
#endif /* FZ_ENABLE_GPRF */
#ifndef FZ_ENABLE_JS
#define FZ_ENABLE_JS 1
#endif /* FZ_ENABLE_JS */
/* If Epub and HTML are both disabled, disable SIL fonts */
#if FZ_ENABLE_HTML == 0 && FZ_ENABLE_EPUB == 0
#undef TOFU_SIL
#define TOFU_SIL
#endif
#endif /* FZ_CONFIG_H */
```