Annotations

An annotation is not page content. Nothing here is drawn by XGraphics - a reader paints it, and can hide it.

TEXT MARKUP

Highlight marks a run of text with a wash of colour.
PdfHighlightAnnotation - PDFKit's highlight()

Underline draws a line along the foot of the run.
PdfUnderlineAnnotation - PDFKit's underline()

Strike out draws ~~through the middle~~ of it instead.
PdfStrikeOutAnnotation - PDFKit's strike()

Squiggly draws the wavy line a spell checker uses.
PdfSquigglyAnnotation - PDFKit has no squiggly()

One run, ~~marked twice~~: green underneath and a strike over the top.
Color is the annotation's, not the page's. Opacity applies to the whole markup.

A markup that runs past the end of a line is one annotation with
two quadrilaterals in it, not two annotations.
AddQuad twice. /Rect is recomputed as the box around every quad.

NOTES - PDFKIT'S NOTE()

A note has no appearance of its own: the reader draws the icon, at whatever size it likes.


Comment


Note


Help


Key


Insert


NewParagraph


Paragraph

Open = true on this one - its popup should already be showing.

Links, attachments and stamps

LINKS - PDFKIT'S LINK() AND GOTO()

The PdfPinata repository

gfx.AddWebLink(rect, url) - a URI action. PDFKit calls this link().

Jump to the parity table on page three

gfx.AddDocumentLink(rect, 3) - a GoTo by one-based page number.

Back to the text markup on page one

gfx.AddNamedLink(rect, "markup") against gfx.AddNamedDestination on page one - PDFKit's goTo().

A link with a tooltip

AddWebLink returns the annotation, so /Contents can be set on it afterwards.

FILE ATTACHMENT - PDFKIT'S FILEANNOTATION()

PdfEmbeddedFile holds the bytes, PdfFileSpecification names them, and the annotation points at it.
The constructor sets PdfAnnotationFlags.Locked, so a reader will not let it be dragged off the page.
Like a note's, the paperclip is the reader's own drawing - so a renderer that paints only appearance streams shows nothing above.

RUBBER STAMP - PDFKIT HAS NO EQUIVALENT

Draft Confidential ForComment Final

Fifteen standard names, drawn by the reader. A stamp with artwork of its own would need an appearance stream.

Parity with PDFKit's annotations

pdfkit.org/docs/annotations.html documents eleven methods. Nine of them have something here; two do not.

PDFKit	PdfPinata
`note(x, y, w, h, contents)`	`PdfTextAnnotation`
`link(x, y, w, h, url)`	`gfx.AddWebLink / PdfLinkAnnotation.CreateWebLink`
`goTo(x, y, w, h, name)`	`gfx.AddNamedLink / gfx.AddDocumentLink`
`highlight(x, y, w, h)`	`PdfHighlightAnnotation`
`underline(x, y, w, h)`	`PdfUnderlineAnnotation`
`strike(x, y, w, h)`	`PdfStrikeOutAnnotation`
`fileAnnotation(x, y, w, h, file)`	`PdfFileAttachmentAnnotation`
`lineAnnotation(x1, y1, x2, y2)`	`MISSING - no /Line annotation`
`rectAnnotation(x, y, w, h)`	`PdfSquareAnnotation`
`ellipseAnnotation(x, y, w, h)`	`PdfCircleAnnotation`
`textAnnotation(x, y, w, h, text)`	`MISSING - no /FreeText annotation`

AND WHAT THIS HAS THAT PDFKIT DOES NOT

`PdfSquigglyAnnotation`	the fourth text markup subtype
`PdfRubberStampAnnotation`	fifteen standard stamp names
`PdfAnnotation.Opacity`	/CA, applied to the whole annotation
`PdfAnnotation.Flags`	Hidden, Print, Locked, NoZoom and the rest
`PdfTextMarkupAnnotation.AddQuad`	many quads under one annotation

The two missing subtypes are appearance-bearing: a viewer will not draw a /Line from its endpoints alone, so adding them means writing appearance streams the way PdfTextMarkupAnnotation already does. Until then, draw the shape with XGraphics.

Nor can a caller supply one: PdfAnnotation is abstract with no public way to set /Subtype, PdfGenericAnnotation is internal, and PdfAnnotations.Add takes neither - so a subtype this library has no class for cannot be added through the typed API.