Introducing Gamut, a D image library

A long-term goal for the Dplug library.
Dplug “winter” is nearly over, since Auburn Sounds next plug-in is ready => 6th September

That means: bug fixes and enhancements for the next six months.

DConf 2022 happened.
The context

dplug:graphics is all based upon a forked ae.utils.graphics

- Was presented in [https://blog.cy.md/2014/03/21/functional-image-processing-in-d/](https://blog.cy.md/2014/03/21/functional-image-processing-in-d/)
- Groundbreaking article, served us well (draw ellipses in L16 and RGBA8 with the same code!)
- Top speed.

I used and forked this in 2015 as I thought it would be easy for you peeps to understand.
Early widgets were complicated to draw

...before the smaller `dplug:canvas` API became the preferred way to draw widgets.

Therefore, marginalizing the older style of writing to “Voldemort” lazy image chains.
But actually we need something like Cairo

- This kind of open-ended API is best when you don’t know what you will need to compute.
- Plenty of `OwnedImage!RGBA`, `ImageRef!RGBA`, `toRef()`, templates...
- Templates are inherently public and lead to large API surface => bad for learning.
- Pretty sure `dplug:graphics` is confusing for new and ancient D programmers.
- I’ve fought to make easy things easier in Dplug, and that means less powerful.

Let’s follow the hugely simplifying example of `dplug:canvas`!
Next step: remake the image basics

- **dplug:graphics** has pretty good image-loading (fast, low memory, best 4:2:0 quality) in pure D.
  - Important to be able to load and convert to another number of channels at the same time.
  - 16-bit support important for PBR (our “depth” is a “displacement map” in video games parlance).

- **dplug:graphics** has best image resizing in pure D (fast, low memory, quality)
  - Basically `stb_image_resize.d` with better kernels (lanczos)

- Make that available to the larger community.

- Plenty of small refactoring steps, will take years.
No more of that

- No more *pixel type* like `!RGBA`: this is a runtime type.

- No more `ImageRef` vs `OwnedImage`: there is only a single `Image` type
  - Able to do both
  - Own its data, or borrow it
  - Advanced layout options to replace `OwnedImage` completely.

- One `Image` type to Rule Them All
Adding exotic codecs

- QOI, lossless codec invented by Dominic Szablewski, faster than PNG and in some cases smaller (typically: large transparent overlays). See https://qoiformat.org/
  Already supported today in Dplug (use the image converter in Gamut).
Adding more exotic codecs

- **QOIX**, a Gamut-specific custom format *(no fixed bitstream as of today Aug 2022)*
  - uses a better compression scheme than regular QOI (at least QOI2AVG)
  - followed by LZ4 compression
  - Is made of 3 different sub-codecs: QOI2AVG, QOI-Plane, QOI-10b... most good ideas came from people creating QOI2AVG in [https://github.com/nigeltao/qoi2-bikeshed](https://github.com/nigeltao/qoi2-bikeshed)
  - Supports 1/2/3/4 channels in 8-bit and 10-bit.

- In most cases, QOIX win against PNG in decoding speed, memory usage.
  - and in some cases in coding efficiency.
  - QOIX 10-bit is lossy, so it “wins” much more often against expensive 16-bit PNG.
  - QOIX is always smaller than QOI.

- The main usage of QOIX will be 16-bit PBR knobs in **Issue #457** (renovating ImageKnob)
  - Otherwise, each knob may take 200kb).
  - At which point, we will have rotating PBR knobs with proper depth, enabling very pretty knobs.
Example 1: 8-bit image knob

Consider this `UIImageKnob` used as Attack knob in Renegate plug-in.
Example 1: 8-bit image knob

800 x 550 8-bit knob image.

- as 16-bit PNG => 94.2 kb
- as 10-bit QOIX => 90.6 kb
- as 8-bit PNG => 75.6 kb
- as 8-bit QOIX => 70.3 kb

Decodes in 1.05 ms instead of 2.67 ms (8-bit, see examples/qoix in Gamut repo).
Encodes 10x faster than PNG.
Example 2: 8-bit transparent overlay

This 800 x 550 8-bit overlay in Auburn Sounds next plugin:

- as 16-bit PNG => 1343 kb (!)
- as 10-bit QOIX => 441 kb
- as 8-bit PNG => 156 kb
- as 8-bit QOIX => 174 kb
- as 8-bit QOI => 252 kb

Likewise, faster encoding and decoding in QOIX. (QOI is even faster).
Example 3: 10-bit UIImageKnob of the future

This 768 x 384 10-bit knob will enable better textured rotating knobs.

- as 16-bit PNG => 467 kb
- as 10-bit QOIX => 193 kb (lossy!)

Also, faster encoding and decoding in QOIX.

The hope is that 10-bit is enough for elevation.
Future endeavours?

- Actually finish Gamut and start using it in Dplug.
- (Why not) Keeping QOI-encoded backgrounds in memory instead of decoding them from JPG on resize.
- (Why not) Programmatic screenshots.
Questions?