Dplug architecture and D ecosystem

Why so much libraries?

Meeting
Oct 24th 2023

At the beginning

- Dplug (2014) quickly expanded to cover features without too much regards for **how easy it was to use and learn.**
  There was so much to do.

- First, make things possible
  - “There is no adequate bootstrap”

~~good~~

capable

8 years later

- Okay. But *Usability* and *Learnability* is the only way to put Dplug on the map (also: it must be productive).
  - *Fortunate that D = easier to learn than C++*
  - *But everything else must be easier too.*

- Doing “nothing” for DSP side was an odd win.

  No concept to learn, no identifier, no “you must make a SoundProcessor with a ChannelLayout that has the bufferSplit flag”.

  => whatever we add for responsible DSP later should be optional.

Names are a kind of debt

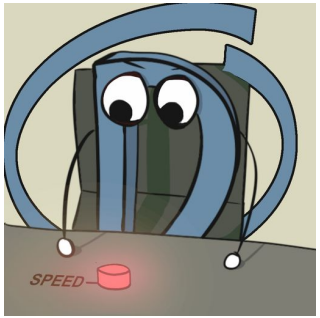
- Familiarity wins: already-known names look easy.
  Question: who knows that linmap and logmap exist in Dplug?

- People don't *like* to learn new names.
  eg: OwnedImage!RGBA vs ImageRef!RGBA vs View!RGBA
  A new name is like an investment that must pay-off.

- Ideally: forcing people to learn things to use a library is a violence that ought to be minimized.

Started `intel-intrinsics` in 2016 out of necessity.

- With already-known names for people that optimize stuff in native space.

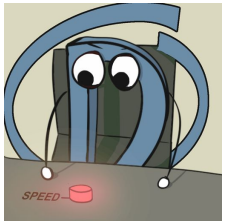


The idea was to “solve” SIMD in D and insulate from compiler bugs. Less assembly = less debt.

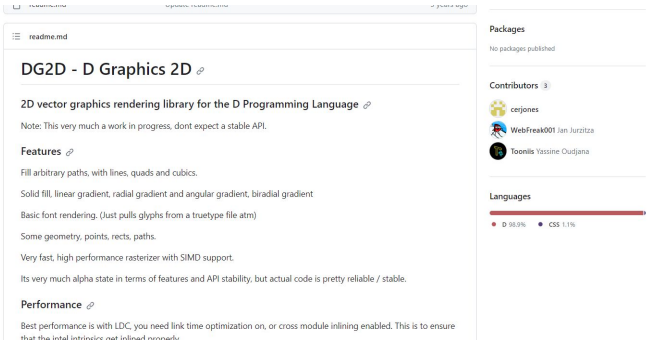
=> Lots of effort, became dependable for consumer apps.

=> companies that sell server software (or do not need speed) less interested.

But: unexpected benefit



@cerjones wrote the very fast rasterizer we use in dplug:canvas *using intel-intrinsics*.


readme.md

DG2D - D Graphics 2D

2D vector graphics rendering library for the D Programming Language

Note: This very much a work in progress, dont expect a stable API.

Features

Fill arbitrary paths, with lines, quads and cubics.

Solid fill, linear gradient, radial gradient and angular gradient, biradial gradient

Basic font rendering. (Just pulls glyphs from a truetype file atm)

Some geometry, points, rects, paths.

Very fast, high performance rasterizer with SIMD support.

Its very much alpha state in terms of features and API stability, but actual code is pretty reliable / stable.

Performance

Best performance is with LDC, you need link time optimization on, or cross module inlining enabled. This is to ensure

Packages

No packages published

Contributors 3

cerjones

WebFreak001 Jan Jurzitza

Toonik Yassine Oudjana

Languages

D 98.9% CSS 1.1%

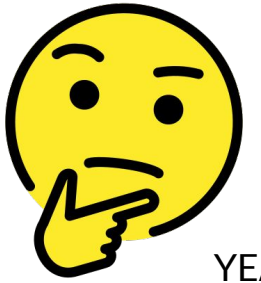
https://github.com/cerjones/dg2d

Personal lesson = library seen as “good”, and not easy to re-do, may bring more value to others, and thus have better network effect.

Started `printed` package

- With no use case at first other than being interested in PDF format. Used the familiar HTML5 Canvas API.

Why am I doing it?



YEAR 2018

Started `printed` package

- With no use case at first other than being interested in PDF format. Used the familiar HTML5 Canvas API.

- **Nowadays (5 years later) “printed” is used to:**
  - generate invoices and B2B quotes
  - generate user manuals in 1 sec *(will present this on another meeting)*
  - generate plugin datasheets…
  - low maintenance, upside is non-negligible
  - thankfully pure D libraries are easy to maintain at low operating cost

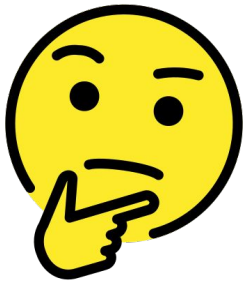
Personal lesson

What if I had just used PrinceXML and paid a licence? I'd have spent 3x less time. When should you buy or build? I like to build!

***Code as investment** can easily lose capital (and lead to debt), except when it has **unexpected** benefits, or a need for detailed control.*

Code sharing with the larger D community will make beneficial network effects more likely to happen.

***Hence**, documenting and well defined interfaces are an economic necessity. Because if it's not documented it might as well not exist.*



YEAR 2020

and indeed JUCE comes with Image, String… with extensive API.

The new plan for the future of Dplug

- Build ONE abstraction per domain with ONE name:
  - Image (with `gamut`)
  - AudioStream
  - etc…

- Use regular software-engineering such as:
  small high-quality interface (very important)
  always *domain* abstractions: language things is NOT a domain, “utils” is NOT a domain
  always **topdown** design
  always documented

- Publish DUB packages that ***covers most use cases*** in a domain
  yes that means betterC-compatible / @nogc / nothrow else fragmentation

The plague of templatitis

How to spot typical templatitis in a D library?

- Library can potentially do everything.
- Large number of short unspecific names
  like “take” or “fill”.
- Not sure what is *internal details*
  and what is the *interface*.
  Hard to tell what can be removed.
- Usually built bottom-up to “*solve the domain*” so the examples look bad on the screen.
- linear build cost with the number of cases.
- build slow

It is still a popular style of library in D, and this is **a problem,** popular C libraries do the exact reverse!

The new plan for the future of Dplug

- Build ONE abstraction per domain with ONE name:
  - Image
  - AudioStream
  - etc…

Use r

Need more domain libraries!

- Font
- Color
- I/O Stream
- AudioBuffer
- GPU compute

Specifically in coming years

Font library

- used in printed
- used in dplug:canvas in a new fillText call
- used for legacy Font in dplug:graphics (drawText)
- can do registry, CSS fontface lookup,
  glyph, output bezier curves
  Benefits: font rendering without glyphs, system font use, text with gradients

Color library

- used for colorspaces in gamut
- follow a CSS Color specification, started this
- used in dplug:canvas
  Benefits: easier to specify colors in Wren code and Dplug, same syntax than in a browser,
  more usable colorspaces

Specifically in coming years

Audio Buffer abstraction

- **Benefit**: easier multi-channel audio and buffer copy
  Needed for AudioProcessor

AudioProcessor abstraction

- **Benefit**: Needed to redo the dplug:client and support channel layouts, ambisonics properly.
- Much needed for dplug:host also.
- Scary transition period to new AudioProcessor but can implement the former client API.

I/O stream library

- no D library has a runtime type for this.
- kinda less needed, but would be very nice.
- Each of above libraries implement own binary parsing and emitting.

Building each of these makes it more likely that people create an UI library or game engine in D.

Also would be very useful

An alternative druntime based upon Hipreme, LWDR and arsd work

- **Benefit**: WebASM support.
  Freedom from druntime/Phobos. Would be cool to get freedom from libc also.
  Getting `new` again, array literals, a sort of limited GC.
- Hipreme engine has shown it was the way to be very portable.
- **Reference**: https://github.com/MrcSnm/webassembly/

All this doesn't have too much Dplug benefit, but it can have an effect on the range of things you can do with D in general.

Question?