Benchmark your UI in 3 easy steps

Happy new year!!!

Meeting
Jan 24th 2023
STEP ONE => Open this page

https://github.com/AuburnSounds/Dplug/wiki/More-Options

- The place for Dplug “options” that meant to be long-lived, versus being just for transitions.

```java
version(Dplug_ProfileUI)
```

- **Goal:** Produce a flame graph to benchmark your UI.
- **Usage:** In your `dub.json` add "Dplug_ProfileUI" to the list of version identifiers.

Now the UI context holds a `traceProfiler()` object that can record events, and does so by default. Every event is recorded for the duration of the session.

In your UI destructor, put this line:

```
version(Dplug_ProfileUI)
{
    import dplug.core.file;
    writeFile("/home/mysis/plugin-trace.json", context.profiler.tobytes());
}
```

This JSON file can be open in https://ui.perfetto.dev/ or chrome://tracing/ to explore visually the UI CPU consumption.
STEP TWO => Do the 2 steps in there

- Step 2.1 => Add this to `dub.json`

```json
"versions": [  
  "legacyMouseCursor",
  "legacyAUHighResolutionParameters",
  "legacyVST2Chunks",
  "legacyZOrder",
  "Dplug_ProfileUI"
],
```
STEP TWO => Do the 2 steps in there

- **Step 2.2 =>** Add this to your `gui.d` destructor

```plaintext
~this()
{
    _fontCouture.destroyFree();
    _fontLato.destroyFree();

    version(Dplug_ProfileUI)
    {
        import dplug.core.file;
        writeFile(`C:\Users\guill\Desktop\plugin-trace.json`,
            context.profiler.toBytes());
    }
}
```

*Use a file path that can be written from the plugin.*
STEP THREE => Run, close, and open the profiler

Demo time.
Example: opening of Couture
Now what to do with bottlenecks?

- **Solution 1:** Former advice: draw less, no PBR updates, optimize.

- **Solution 2:** NEW! For large widgets, use the graphics thread pool for your own work in `onDrawPBR` and `onDrawRaw`
Widgets are drawn in parallel, but how to use parallelism in a single large widget?

2 new UIElement flags =

```java
/// Is not drawn in parallel with other widgets, when drawn to the Raw layer.
flagDrawAloneRaw = 8,

/// Is not drawn in parallel with other widgets, when drawn to the PBR layer.
flagDrawAlonePBR = 16,
```

Can access thread pool JUST in onDrawPBR and onDrawRaw! NOT outside of it. Do NOT use in reflow().
Example of PBRBackgroundGUI: constructor

```java
this(SizeConstraints sizeConstraints) {
    super(sizeConstraints, flagPBR | flagAnimated | flagDrawAlonePBR);
    _diffuseResized = mallocNew!(OwnedImage!RGBA);
    _materialResized = mallocNew!(OwnedImage!RGBA);
    _depthResized = mallocNew!(OwnedImage!L16);
    version(decompressImagesLazily)
}
```

It means: I can't be drawn in parallel with other widgets in the PBR layer.
// Potentially resize all 3 backgrounds in parallel
void resizeOneImage(int i, int threadIndex) noexcept @noc
{
    ImageResizer resizer;
    if (i == 0)
    {
        version(Dplug_ProfileUI) context.profiler.begin("resize Diffuse background");
        resizer.resizeImageDiffuse(_diffuse.toRef, _diffuseResized.toRef);
        version(Dplug_ProfileUI) context.profiler.end;
    }
    if (i == 1)
    {
        version(Dplug_ProfileUI) context.profiler.begin("resize Material background");
        resizer.resizeImageMaterial(_material.toRef, _materialResized.toRef);
        version(Dplug_ProfileUI) context.profiler.end;
    }
    if (i == 2)
    {
        version(Dplug_ProfileUI) context.profiler.begin("resize Depth background");
        resizer.resizeImageDepth(_depth.toRef, _depthResized.toRef);
        version(Dplug_ProfileUI) context.profiler.end;
    }
}
context.globalThreadPool.parallelFor(3, &resizeOneImage);
Lots of small Dplug news

- Latest Dplug uses Gamut, so you can load QOIX in addition to JPEG, PNG, and QOI
- `dplug` master tool, in the future you won’t have to build other Dplug tools
  
  ```bash
  $ dplug build <stuff>
  ```
  instead of `$ dplug-build <stuff>`
- `dplug-build --root` can build plugins from other directories
- Z-order mostly fixed and available from Wren (like `.visibility`)
- `tailSizeInSeconds()` fixed, doesn’t default to 2 seconds anymore => set it
- macOS Ventura, AAX native M1 support
- events for parameter hovering
- Windows cloud signing support (Certum)
- a bit better image resizing (speed and quality)
- fix laggy controls in non-VST2 in some hosts

Those points are all in [https://github.com/AuburnSounds/Dplug/wiki/Release-notes](https://github.com/AuburnSounds/Dplug/wiki/Release-notes)

No real big item here, but there is a LOT more work to do.
When you find a bug in Dplug

- Gives as much information as possible on what you were doing when you saw a problem

- No reproducible instructions => what to do?
Questions?