

Grease Pencil 2.8 Merge Meeting - 18 April 2018

Participants: Bastien, Campbell, Dalai, Joshua, William

Location: BI 2.0 Kitchen/Meeting Room

Summary

There are a number of big architectural-level issues currently blocking a merge.

To resolve some of these, meeting agrees we need both a *Short Term Solution* and *Long Term Solution* for each of these - a *Short Term* solution so that we can move forward with a merge into 2.8 (as it's going to get increasingly hard to keep a branch up to date as Code Quest proceeds), and a *Long Term* solution that ensures that Blender code/design quality standards are maintained across the codebase.

Key Issues Identified

0) Compatibility with *Hero* files/content

In an ideal world, *Hero* files would continue to work automatically in 2.8 without any extra effort. However, this is unlikely to be the case here, as there are significant architecture-level here that need to be resolved.

Resolutions:

1. *Hero* .blend files “should” be able to be converted to post-merge 2.8 (“soft-goal”)
2. Maintaining compatibility with *Hero* files **should not** constrain architectural decisions

1) Palettes

There are concerns about the use of Blender's Palettes (originally a datablock for a preset-like system for tool settings used during Texture Paint) for what is essentially a Material-like system for GP objects. There are two main issues:

- a) **Overlap with Material Functionality** - The Palettes + Palette Slot system basically implements something like Materials + Material Slots (or to be specific, Material Collections + Materials + Material Collection Slots). There were also concerns that the GP extensions here exceed what a “Palette” should be, as the options + animation support effectively make them more of like “Stroke Style” (similar to Freestyle Line Style datablocks) instead.
- b) **Implementation violates Blender design** - See DNA/Library Handling notes

2) Modifiers System (in general)

There is disagreement over what should happen here on a broad level (i.e. “*Should these ‘GP modifiers’ even be ‘Blender modifiers?’*”). The concerns center on how GP modifiers can essentially only be used on GP objects (while standard modifiers can’t be used on GP objects), leading to some operators being duplicated/implemented separately. There are arguments for and against keeping GP modifiers in the standard modifier stack, but a final solution still needs to be found.

The meeting agrees however, that there are technical concerns here (e.g. regarding Depsgraph/COW).

3) “VFX” Modifiers

Meeting agrees that these should not be part of the “Modifiers” list/stack, as they do not really function as modifiers, but instead work more as placeholders for a post-process object effect system (for which there are no current analogues in Blender).

4) “Array” Modifier

There are concerns that the “Array” modifier differs too much from the “normal” Array modifier (i.e. no end cap support, follow curve, etc.) for it to still be called an “Array Modifier”. A secondary/related issue was that if they are meant to behave the same, why the GP ones implementing similar functionality are in separate files/modifiers - since GP modifiers have their own callbacks for operating on data.

5) Modes

It is a bit contentious whether these operate per-Object or per-Workspace.

6) DNA / Library / Data Handling

There were concerns about the way that some things are handled in the data structures (see notes by Bastien in the [patch \(D2889\) / tracker](#)). In particular, these concern:

- Palette / PaletteColor stored in each stroke (i.e. PaletteColor is non-ID data - can be dangerous with various lib handling tools)
- Weight Paint data storage could also use some attention (e.g. it may be better as something more similar to CustomData?)

Resolutions

Meeting identified the following paths forward towards resolving the issues identified. Joshua oversees fixes with help from GP team.

Item	Short Term Solutions/Actions <i>These are actions that need to be taken before a merge can be performed. These mostly represent a minimal-standard of implementation/design required for inclusion in Blender.</i>	Long Term Solutions/Actions <i>For some of these topics, even though the short term solutions may be ok to apply to get something into 2.8 so that users can get some benefit from the work done for the Open Movie project, we need a better design for the long-term sake of keeping Blender's design sane.</i>
1	a) Rename/split "Palettes" struct as used for GP Materials away from Blender Palettes (*1), and work on final design. b) Must fix these issues	Implement final solution (TBD still what this will be - meeting could not reach agreement on best way forward. Will reconsider options after consulting GP team via videochat)
2	Isolate from rest of Blender	Integrate with DEG/COW
3	No VFX Modifiers	Make this into a general per-Object effect stack
4	No "array" modifier or (anything that conflicts with existing modifiers)	Rename offending modifiers, or add missing functionality to bring them in line with standard
5	Try to integrate (otherwise ignore the issue for now)	Integrate with workspace mode and multi-object editing
6	Must fix. Dangerous/bad code design.	N/A

(*1) Meeting could not reach agreement on best course of action here (for short-term solution to palettes). Specifically, we have the following two main options:

Rename Palettes to <SomethingElse>	Keep Current Name
<ul style="list-style-type: none">- No do_version problems down the track (i.e. no-one should have "production files" saved using "Palettes", meaning we don't need to try to patch everyone's files- Allows to leave hacks in a bubble- More work in short term fixing all code	<ul style="list-style-type: none">- Allows for a merge sooner- No duplication of effort when we do figure out what we want to replace this with (i.e. as we don't try to change all existing code using Blender Palettes, we won't have to then change from those to whatever our final solution is)

(Joshua will be responsible for deciding which of these approaches we adopt in the short term)

The best course of action long term also needs to be decided still. Meeting discussed several options - more input is needed from GP team regarding the suitability of each option with regard to 2D artist workflows though:

- 1) **Palette Colors -> Blender Materials (i.e. 1 Palette Color = 1 Material)** - PaletteColor's are in many ways similar to Blender Materials. One of the big unknowns though is whether doing this would impose too much overhead on artists when importing/hooks up all the colors needed to work with a particular character (or characters).
- 2) **Palette -> Blender Material with nested "GP Stroke Style" struct, with "sub colors" stored in this struct to represent Palette Colors** - The benefit of this approach is that GP objects will still be able to have a Palette-like thing to cart around a bunch of colors, and all the existing tools for working with Materials will still work. Non-GP objects can safely ignore and be unaware of the existence of the GP specific stuff. Only problem though is that this does feel slightly hacky.
- 3) **Stick to a similar system to what we have now, but call it something like "StrokeStyles" or "GPLineStyles" - basically, anything other than "Palette"**

It's also worth noting a few other things that were discussed with regard to these options:

- If we go with Blender Materials here, there is a potential problem with confusion when you've got a bunch of "GP materials" vs "Cycles/Eevee/normal" materials. It could be argued though that this is a more general problem though.
- There was a brief mention of how Freestyle Line Styles end up doing similar things in some regards, and whether we could/should do some combining of systems at some point?
- There was also a brief discussion about whether it would make sense to have GP evaluation work by converting geometry to Blender Meshes. Then we could easily reuse existing modifiers for GP tools.

