2.8 Before the Quest

Introduction

March is coming to an end, and Spring is upon us. With April we also get started on the Blender 2.8 code quest. Let's now take a moment to provide a reference guide to the main projects that will be tackled during the code quest.

But before diving into it, we shall revisit Blender 2.8 goal:

"Blender 2.8 will enable artists to work faster and more efficient. Emphasis is on helping specialists or to enable specialist tasks better. 2.8 will allow Blender template/configurations to start with reduced or different user interfaces."

Keep that in mind while you go through this document, and check the Blender 2.8 design if you need further information.

Projects

- User experience and user interface design
- Workspaces
- Render Engines
    - Cycles
    - EEVEE
    - Workbench
    - Clay
    - Blender Render
    - Non-Render Engines
- Grease Pencil
- Asset Management
- Dependency Graph
- Overrides
    - Static Overrides
    - Dynamic Overrides
- Templates
- Python
- Other Projects

UI/UX - user experience and user interface design

During the code quest we will have designers, developers and an entire animation studio sharing the same building. This will allow us to find production friendly workflows that combine usability, visual appeal and are technically feasible.

Current Status: In each project here we list their own status in regard to UI/UX. There are unlisted known projects however (e.g., collections and outliner) which are mostly UI/UX centered and will get a proper treatment during the code quest.

Quest Goals: Finish the begin-to-end workflow required for working with collections. Polish the outliner centric pipeline for collections and scene management. Make sure we have solid workflows for the most common use cases for an animation studio.

Challenge: Design a moving vehicle while driving it at the same time we construct it.

Read more:

- Highlights of workflow workshop
- View layers and collections
- Outliner improvements

Workspaces

Blender 2.8 is all about improving workflows. And workspaces bring the biggest change in workflow in Blender 2.8. They encapsule not only Layouts (former *Screens*) but also modes, per workspace add-ons and in the future even keymaps.

And since mode is no longer part of the object, it is time to try multi-object editing. Initially for mesh editing and multiple characters posing.

Current Status: Basic functionality is there. Including per workspace add-on filtering. We still need to move tool settings into workspaces.

Quest Goals: Polish the usability based on production workflow feedback.

Challenge: Provide good built-in workspaces and have users excited about creating their own. Find a way for scene and workspace engine settings to live side-by-side in harmony.

Read more:

- Workflow workshop write up: Workspaces
- T54242 - Multi-object editing

Render Engines

Cycles

Cycles is still the go-to engine when it comes to photo-realism with Blender. Aside from its regular development agenda Cycles will be fully integrated with Blender 2.8. Making use of dynamic overrides, new dependency graph, view layers, and collections.

Current Status: Integrated with the new features (e.g., new dependency graph and collections) yet missing functionality from 2.7x (point density, motion blur, …).

Quest Goals: To fix regressions and unify settings with EEVEE.

Challenge: Polish the pipeline for users with Cycles and EEVEE in the same files. See if we can have a fast viewport preview to can be used with editing tools.

Read more:

- T54180 - Cycles Blender 2.8 tasks
- T53214 - UI/UX implement per engine material outputs

EEVEE

EEVEE is a realtime rendering engine direct from the viewport. It uses what is most modern in the video game industry. This engine replaces Blender Internal as a fast alternative to Cycles while also working within the PBR (physically-based rendering) paradigm.

High-end and low-end computers alike should be able to use this engine. Although we require a few tricks such as the recently implemented progressive shader compilation to make sure most computers can handle a full shader oriented pipeline.

Current Status: Initial feature set is complete, with integration with the render pipeline (offline rendering, sequencer, compositor).

Quest Goals: To make it production ready - bug fix, polishing and optimizations.

Challenge: Find the balance between high quality looks and playback performance. Try to support headless (background) rendering for realtime based engines such as EEVEE.

Read more:

- EEVEE FAQ

Workbench

The workbench engine is a general purpose drawing engine. It will allow someone to model, sculpt, … with features such as studio light, reflective floor, procedural object colors, matcaps and more.

Current Status: Missing design and implementation.

Quest Goals: Decide on final design early on, and finish implementation by the end of it.

Challenge: Leverage keeping old functionality (wireframes) with modern editing techniques.

Clay Engine

The clay engine was created as a working prototype to get the other parts of Blender going until EEVEE and Workbench were ready.

Current Status: It misses offline rendering (F12).

Quest Goals: Replace it with the Workbench engine.

Challenge: Nothing, really. From the beginning we knew we would remove this, in fact you can even build Blender without it.

Read more::

- Viewport plan of action

Blender Render

The Blender Render (as known as Blender Internal) engine will be replaced by EEVEE. However this engine is still required for the procedural texture (Cloud, Wood, Musgrave, …) which in turn are supported as brush textures.

Current Status: Scheduled to be deprecated, not supposed to around 2.8 anymore.

Quest Goals: Remove it from the codebase.

Challenge: To separate the procedural texture code from the rest of the engine may not be doable.

Non-Render Engines

Object mode, edit mode, sculpt have dedicated engines that are overlaid on top of the viewport. This allows for any of those modes to be fully functional regardless of the underneath render engine and how slow it is.

Current Status: Basic functionality implemented. We need a design for which options we want for the different modes.

Quest Goals: Get a robust implementation for the modes to be used by the Spring project, while delivering a clear design to any of the missing ones.

Challenge: Get people excited to work in 2.8 from scratch.

Grease Pencil

Grease pencil evolved from being a 2D annotation tool, into a full fledge drawing engine. It is currently being developing on its own branch - greasepencil-object, on top of Blender 2.8.

Current status: The open movie Hero is using grease pencil fully, relying on its integration with EEVEE and Cycles. It is almost complete tool-wise.

Quest goal: Merge with Blender 2.8 as early as possible. Create and polish its own dedicated Workspace.

Challenges: Finish Hero within the Code Quest as the first open movie made with Blender 2.8.

Read more:

- 2D Animation in Blender

Asset Management

The underlying technology is in place and will land in Blender 2.8 soon. The next step are the engines, built on top of the asset management system. The first one will be aimed at the open movies using the Spring project as a users case.

Current status: Ready for merge with Blender 2.8

Quest goal: Collaborate with the Spring team to design and implement Amber

Challenge: Extensive testing and user interface

Read more:

- Asset management and Pipeline
- T46049 - Assets Integration in Blender

Dependency Graph

The backbone of Blender! Connecting everything together. Blender 2.8 is still missing working modifiers among them bone animation. This is a great showstopper for the adoption of 2.8 in production.

Current status: Aside from operators everything works, although in a hacky way. That includes skeletal animation, modifiers, constraints and shapekeys.

Quest goal: Modifiers and operators fully working and fast animation playback.

Challenge: The pending technical designs are very tricky. And the port of all the modifiers and operators can use help from the community.

Read more:

- Cowherd challenges [temporary link - still waiting for wiki]

Overrides

Static Overrides

Any linked datablock can now have local properties. Technically is it as if everything can be treated as armature proxies in Blender 2.7x.

Current Status: Core implemented, missing changes in RNA diff and a workflow for proxy.

Quest Goals: Polishing and wrap up.

Challenge: Design an armature proxy workflow superior to 2.7x.

Read more:

- D2417 - Initial ID static override proposal

Dynamic Overrides

View layers and collections can override draw and datablock options. Those overrides are local to the view layers, but handled as part of the dependency graph rewrite.

Current Status: We have a prototype of dynamic overrides working for clay shaders. We need an interface for any datablock as well as the dependency graph implementation.

Quest Goals: Collection and view layer dynamic overrides fully working.

Challenge: Easy of use interface.

Read more:

- The override manifesto
- Depsgraph and dynamic overrides

Templates

Templates allow users to customize the Blender interface to the extent of making it work as a new stand-alone app. This can be used for 3d printing, basic sculpting, tinker-cad projects. For example, basic sculpting could be a finger painting app for kids.

This may not be directly tackled during the code quest, but we are open for people providing feedback so we can make the technology for it. So basically we need designer work.

Current Status: It is working well. We need people using it to expose missing features.

Quest Goals: Templates may not be directly tackled during the quest. However we can polish it if we get a clear picture of new expected features or bugs.

Challenge: Find projects ready to put template to use with well designed valid applications for it.

Read more:

- Application templates - Blender Manual

Python

The Blender Python API reflects the functionality of Blender itself. Therefore it is expected that all the changes in Blender 2.8 require Python to play some catch up. Specially in the drawing part of the API since we no longer immediate mode drawing.

Some of those tasks don't necessarily need production feedback and iterations between design and code. As such may be postponed for after the code quest and tackled remotely.

Current Status: New features are exposed in the API, however the change to OpenGL 3.3 wasn't followed up by Python.

Quest Goals: Update the drawing and render API to reflect the new designs in Blender.

Challenge: Determine the level of abstraction we want to expose for drawing. And make sure scripts that rely heavily on the old drawing system can be ported over.

Read more:

- Updating scripts from 2.7x
- Blender 2.8 API changes

Other Projects

Once the main projects are in good enough shape we can re-visit other projects that were started during the Blender 2.8 development process.

For instance:

- Top bar
- Manipulators
- Tools
- Workspace specific keymaps, add-ons.

During the code quest we will have an opportunity to re-evaluate the state of these projects. Depending on availability and relevance of the project they can be rebooted during the code quest itself.

Current Status: These projects are currently on hold, some of them with initial implementations in branches (e.g., top-bar), while some (e.g, tools) have their initial work already merged in 2.8 itself.

Quest Goals: Usability and user interface designs for the projects we will carry on.

Challenge: Find the killing features among feature creeping ones to allocate our resources wisely.

Read more:

- Blender 2.8 - priorities reconsideration proposal
