A New **Developer Documentation** Platform

blender

Why Change? (1/4)

Let’s talk about the Wiki...

- **Not the best experience**
  Developers are not exactly excited about it.
- **Navigation & structure needs rethinking**
  Important content is elusive, we’re not guiding people to it
- **Disconnected from the code**
  Context switch required

Why Change? (2/4)

Let’s talk about the Wiki…

- **Closed platform**
  No real community involvement, no feedback loops.

- **Better writing experiences/tools available**
  WYSIWYG, collaborative editing, review changes, … – HackMD :)

Why Change? (3/4)

We’re not good at high level technical documentation.

😐

Why Change? (4/4)

Developers don’t really use the documentation either.

🤔

Can we establish a new **developer documentation culture**?

Well, others did!

Doctave

‹ All posts

How Google, Twitter, and Spotify built a culture of documentation

Niklas Begley

Sep 7, 2021

Many technical problems ultimately turn out to be people problems, and a lack of good documentation is no exception. Writing and maintaining documentation is a habit that

https://www.doctave.com/blog/2021/09/07/how-google-twitter-and-spotify-build-culture-of-documentation.html

Twitter, Google, Spotify

Before

- Technical documentation identified as **big problem**
- **Everybody's problem**, but **nobody's job**
- **Technical writers didn't solve it**
  Jumped from project to project, docs outdated quickly.

Twitter, Google, Spotify

After

- **Greatly improved documentation culture**
  Technical documentation used and updated all the time - by engineers!
- **Technical writers help & empower**
  Maintain docs infrastructure, make strategic decisions.

How did they do it?

- **Culture of docs**
  Documentation sprints, education, lead by example

- **Standardize & centralize**
  Common platforms, templates

How did they do it?

- **Feedback loops**
  Easy bug reporting for docs, “Was this page useful?”, ...

- **Keep it simple**

How did they do it?

Empower developers:
"Fiercely optimize for the engineer"!

Docs as Code

Docs as Code

Docs as Code

Treat documentation like code
Version control, collaboration, and automation

- Simple markup language
- Close to the code
- Pull requests, versioning, branching
- Forge integration (Gitea, Github, etc.)
- Continuous delivery, automated checks

Sounds Familiar?

Blender 4.0 Manual

Search docs

GETTING STARTED

About Blender

Installing Blender

Configuring Blender

Help System

SECTIONS

User Interface

Editors

Scenes & Objects

Modeling

Sculpting & Painting

Introduction

/ Sculpting & Painting / Brush / Brushes

Brushes

Reference

Mode: Sculpt Mode

Panel: Sidebar ▸ Tools ▸ Brushes

For painting/sculpting modes each brush type is exposed as a tool in the toolbar. The brush on the other hand is a saved preset of all the brush settings, including a name and thumbnail.

All these settings can be found and changed here in the tool setting (brush, texture, stroke, falloff & cursor).

Brushes

Clicking on the brush thumbnail will open the Data-Block Menu to select a brush.

Add Brush (Duplicate icon)

When you add a brush, the new brush is

Sounds Familiar?

Blender Studio

Search Ctrl K

Films ↗ Training ↗ Blog ↗ Pipeline Characters ↗ •••

On this page >

Blender SVN

blender-svn is a Blender add-on to interact with the Subversion version control system from within Blender.

Blender-SVN Demo Video

Installation

1. Download latest release
2. Launch Blender, navigate to `Edit > Preferences` select `Addons` and then `Install`,
3. Navigate to the downloaded add-on and select `Install Add-on`

Sounds Familiar?

Flamenco

About Download Documentation FAQ Get Involved

Search

Edit page

Navigation

Shared Storage

Flamenco needs some form of *shared storage*: a place for files to be stored that can be accessed by all the computers in the farm.

Basically there are three approaches to this:

Approach	Simple	Efficient	Render jobs are isolated
Work directly on the shared storage	✅	✅	❌
Create a copy for each render job	✅	❌	✅
Shaman Storage System	❌	✅	✅

Proposal

- **Material for MkDocs***
- **Continuous delivery via buildbot**
- **Edit with preview in Gitea**
- **Git (LFS?) repository**
- **Pulled with `make update`?**
- developer.blender.org/docs

* Alternatives: Sphinx, VitePress, Hugo

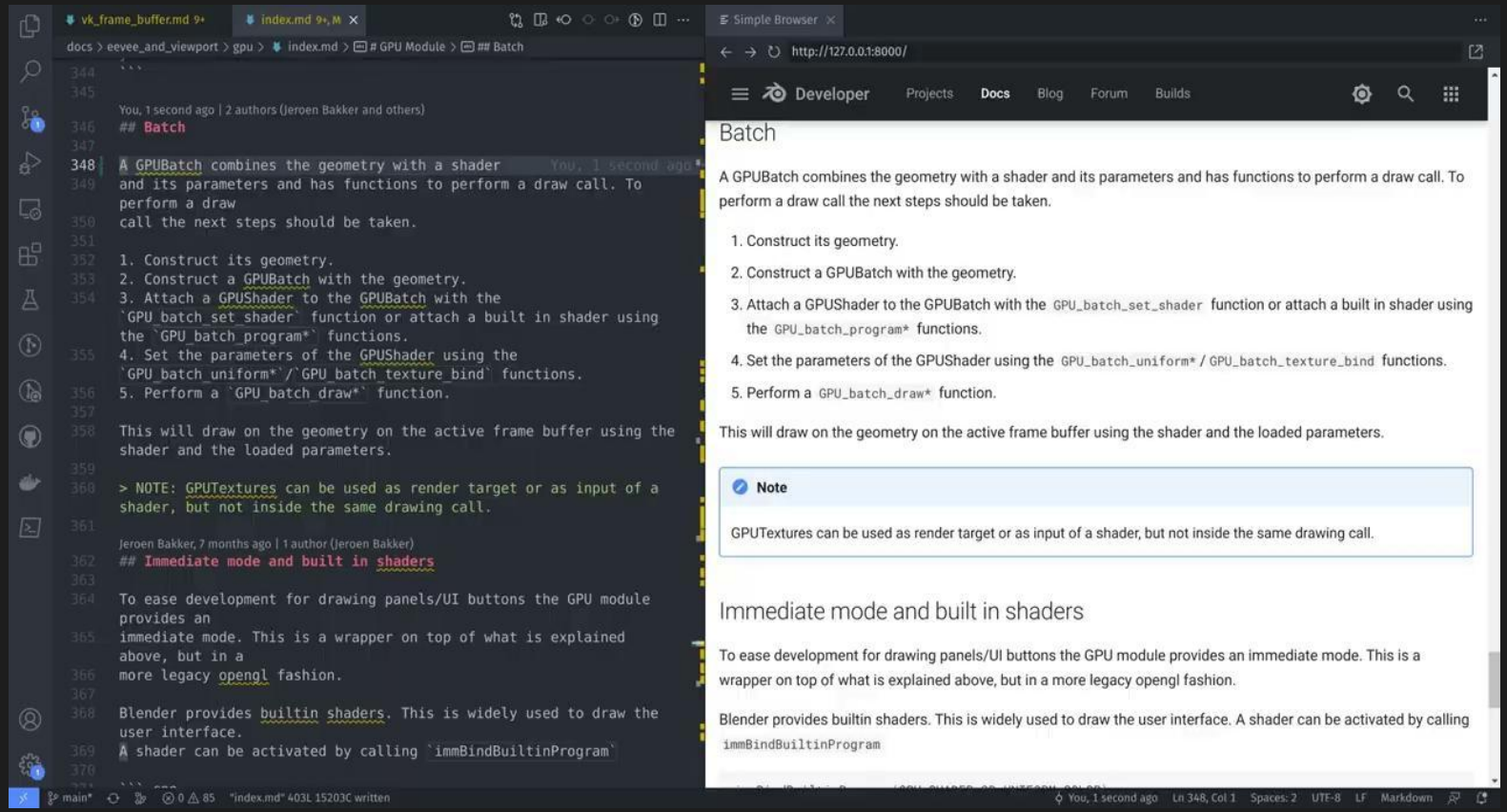
Goodbye Wiki?

- **What moves to the new platform?**
  Most pages, including release notes
- **Personal pages & weekly reports:**
  Personal repository on Gitea
- **Transition requires manual work**
  Basic Wiki to Markdown converters available (https://projects.blender.org/brecht/wiki-to-markdown).
  People volunteered to help.

Demo Time

developer.blender.org/docs

Demo: Offline Editing



Demo: Offline Editing

vk_frame_buffer.md 9+ | index.md 9+

docs > eevee_and_viewport > gpu > index.md > # GPU Module > ## Geometry

```
(attributes, uniforms, uniform buffers, textures and shader storage buffer objects).

Jeroen Bakker, 7 months ago | 1 author (Jeroen Bakker)
## Geometry

Geometry is defined by a `GPUPrimType`, one index buffer (IBO) and one or more vertex
buffers (VBOs). The GPUPrimType defines how the index buffer should be interpreted.

Indices inside the index buffer define the order how to read elements from the vertex
buffer(s). Vertex buffers are a table where each row contains the data of an element.
When multiple vertex buffers are used they are considered to be different columns of
the same table. This matches how GL backends organize geometry on GPUs.      Jeroen Bakker, 7 months ago • Initial gpu and vulkan tech d

Index buffers can be created by using a `GPUIndexBufferBuilder`

``` cpp title="Create Index Buffer"
GPUIndexBufBuilder ibuf
/* Construct a builder to create an index buffer that has 6 indexes.
 * And the number of elements in the vertex buffer is 12. */
GPU_indexbuf_init(&ibuf GPU_PRIM_TRIS, 6, 12);

GPU_indexbuf_add_tri_verts(&ibuf, 0, 1, 2);
GPU_indexbuf_add_tri_verts(&ibuf, 2, 1, 3);
GPU_indexbuf_add_tri_verts(&ibuf, 4, 5, 6);
GPU_indexbuf_add_tri_verts(&ibuf, 6, 5, 7);
GPU_indexbuf_add_tri_verts(&ibuf, 8, 9, 10);
GPU_indexbuf_add_tri_verts(&ibuf, 10, 9, 11);

GPUIndexBuf *ibo = GPU_indexbuf_build(&builder)
```
```

Simple Browser

http://127.0.0.1:8000/

Developer | Projects | Docs | Blog | Forum | Builds

Geometry

Geometry is defined by a `GPUPrimType`, one index buffer (IBO) and one or more vertex buffers (VBOs). The GPUPrimType defines how the index buffer should be interpreted.

Indices inside the index buffer define the order how to read elements from the vertex buffer(s). Vertex buffers are a table where each row contains the data of an element. When multiple vertex buffers are used they are considered to be different columns of the same table. This matches how GL backends organize geometry on GPUs.

Index buffers can be created by using a `GPUIndexBufferBuilder`

Create Index Buffer

```
GPUIndexBufBuilder ibuf
/* Construct a builder to create an index buffer that has 6 indexes.
 * And the number of elements in the vertex buffer is 12. */
GPU_indexbuf_init(&ibuf GPU_PRIM_TRIS, 6, 12);

GPU_indexbuf_add_tri_verts(&ibuf, 0, 1, 2);
GPU_indexbuf_add_tri_verts(&ibuf, 2, 1, 3);
GPU_indexbuf_add_tri_verts(&ibuf, 4, 5, 6);
GPU_indexbuf_add_tri_verts(&ibuf, 6, 5, 7);
GPU_indexbuf_add_tri_verts(&ibuf, 8, 9, 10);
GPU_indexbuf_add_tri_verts(&ibuf, 10, 9, 11);

GPUIndexBuf *ibo = GPU_indexbuf_build(&builder)
```

Vertex buffers contain data and attributes inside vertex buffers should match the attributes of the shader. Before a buffer can be created, the format of the buffer should be defined.

And Beyond...

- We need to **lead by example**
- **Technical documentation days**
  Last Friday of every month.
- **Education**
  Google's technical writing courses:
  https://developers.google.com/tech-writing
- **Examples & templates**

And Beyond...

Documentation Structure

Search...

New Developer Introduction
Welcome! Advice on how to get started.

Communication
The most important thing.

Code & Design Documentation
Technical documentation about the code and big picture design.

Building Blender
Instructions for compiling Blender locally.

Modules
Blender components and their owners.

Style Guide
Coding guidelines and committer etiquette.

Tools
Setup your development environment.

Process
Release cycle, BugTracker, Code Reviews and Testing.

Release Notes
What changed in each Blender version.

Google Summer of Code
A program that introduces students to open source software development.

Python
Learn about scripting and add-ons.

Translation
Blender UI internationalization

Infrastructure
Details about the online ecosystem that supports Blender development.

FAQ
Common questions about the development process.

Search...

New Developer Introduction
Welcome! Advice on how to get started.

Communication
The most important thing.

Code & Design Documentation
Technical documentation about the code and big picture design.

Building Blender
Instructions for compiling Blender locally.

Modules
Blender components and their owners.

Style Guide
Coding guidelines and committer etiquette.

Tools
Setup your development environment.

Process
Release cycle, BugTracker, Code Reviews and Testing.

Release Notes
What changed in each Blender version.

Google Summer of Code
A program that introduces students to open source software development.

Python
Learn about scripting and add-ons.

Translation
Blender UI internationalization

Infrastructure
Details about the online ecosystem that supports Blender development.

FAQ
Common questions about the development process.

Blender Developer **Handbook**

Blender Developer Handbook

This is the Blender Developer Handbook. It tries to provide all the necessary general information needed to develop Blender. As such it is not only aimed at beginner developers, experienced developers use this handbook too.

Proof of Concept

The entire structure of the handbook is just another proof of concept. Pages are empty or don't even exist. Expect a bunch of 404 links.

Last update: 5 minutes ago

Created: 5 minutes ago

Blender Developer Handbook

This is the Blender Developer Handbook. It tries to provide all the necessary general information needed to develop Blender. As such it is not only aimed at beginner developers, experienced developers use this handbook too.

Proof of Concept

The entire structure of the handbook is just another proof of concept. Pages are empty or don't even exist. Expect a bunch of 404 links.

Last update: 5 minutes ago

Created: 5 minutes ago

Learning **Flow**

Asset System

The asset system brings a native understanding of assets (entities packaged for sharing/reuse) to Blender's core design, and enriches it with a number of features for great asset based workflows.

For example it includes: The asset browser, asset shelfs, asset libraries, asset library loading, asset catalogs, asset metadata, etc.

The backend implements all the core types and functionality, which various parts Blender can access. The design is user experience driven, and as such, the backend very much serves the user interface. They work in close collaboration to provide an experience that makes assets feel like first-class citizens in Blender.

Note

The asset system is still in early development so expect this documentation to receive regular updates. In various places, it will refer to designs that are not there yet (at least not in the `master` branch), partially there or just temporary. This will be clearly indicated.

Last update: 18 minutes ago

Created: 18 minutes ago

Important Concepts

What is an *Asset* anyway?

The Blender reference manual gives a user level answer to this question. As far as the asset system design goes, **assets are arbitrary entities that are packaged for organized sharing/reuse**.

The term *entity* is used here, because the design is meant to support assets that are not Blender data-blocks (also called IDs or ID Datablocks on a technical level), even if the current implementation is limited to that. In future it should be possible to let the asset system deal with any data as assets, like files on disk, USD prims, SQL data-base entities, data fetched from the web, etc. This only works because the asset system doesn't deal with the actual underlying entity itself (the object, the material, the brush, etc.). It only deals with the package of the entity.

Asset Representation

If assets are packaged entities, what does the package look like? **An *asset representation* is the package for an asset, which enables the asset system to work with it.** When loading an asset library, an asset representation is created for each detected asset and put into the asset library storage.

Note that an asset representation is just the package itself, and usually doesn't contain the actual entity. It contains information on how/where to find the entity, so that the asset can be loaded when the user asks for it.

Back to top

`asset-library-hash`

Hash of the absolute file path of the asset library.

`asset-index-hash`

Hash of the absolute file path of the asset file.

`asset_file`

Filename of the asset file. Not used by Blender, but is added for discoverability convenience.

Content

```
{
  "version": <file version number>,
  "entries": [{
    "name": "<asset name>",
    "catalog_id": "<catalog_id>",
    "catalog_name": "<catalog_name>",
    "description": "<description>",
    "author": "<author>",
    "tags": ["<tag>"],
    "properties": [..]
  }]
}
```

`version`

the version of the asset index file. It is used to identify the structure of the content. Asset indexes with a different version than used by Blender would be regenerated. Blender 3.1-3.4 expect version attribute to be `1`.). Later versions might require to change it.

is a proper asset system API for add-ons to use.

Back to top

Python API

At this point, there is no proper Python API for the asset system yet, it is still in the planning. There are still ways to achieve commonly requested functionality through other parts of the Python API however.

How do I load custom previews? ¶

There is no simple `asset.load_custom_preview(filepath)` method or similar available. But there are still two ways to do the job:

- `bpy.ops.ed.lib_id_load_custom_preview()` : Attempts to load an image file from a `filepath` property (opens a File Browser if not set) to the active ID set in context.

```
override = context.copy()
# Set context "id" member to some ID, e.g. a material.
override["id"] = my_material
with context.temp_override(**override):
   bpy.ops.ed.lib_id_load_custom_preview(filepath="path/to/image.png")
```

  Or to call the operator from a button:

```
# Set layout context "id" member to some ID, e.g. a material.
layout.context_pointer_set("id", my_material)
props = layout.operator("ed.lib_id_load_custom_preview")
props.filepath = "path/to/image.png"
```

- `bpy.types.ID.preview` : ID assets (and currently all assets have to be IDs) share the ID's

Recap

- **New developer documentation culture?**
- Others did it, let's learn from them
- **Docs as code** based developer documentation, replacing Wiki
- Simple workflows, **optimized for engineers**, **open to the community**
- Restructured: Blender Developer handbook & learning flows

Status

Phase 0:

- ✅ Research, experiments & testing setup
- ✅ Get buy-in

Phase 1 - Setup:

- ✅ Buildbot CD setup
- ✅ Hosting on developer.blender.org/docs
- ▢ Custom theme, navigation & landing page
- ▢ Setup repository (Git LFS setup?)

Status

Phase 2 - Transition:

- ▢ Move relevant pages from Wiki
- ▢ Setup developer handbook
- ▢ Archive Wiki

Phase 3 - Source code integration:

- ▢ Checkout docs with source code?