renderlet

PAINT BY NUMBERS

HIGH PERFORMANC DRAWING IN WASM

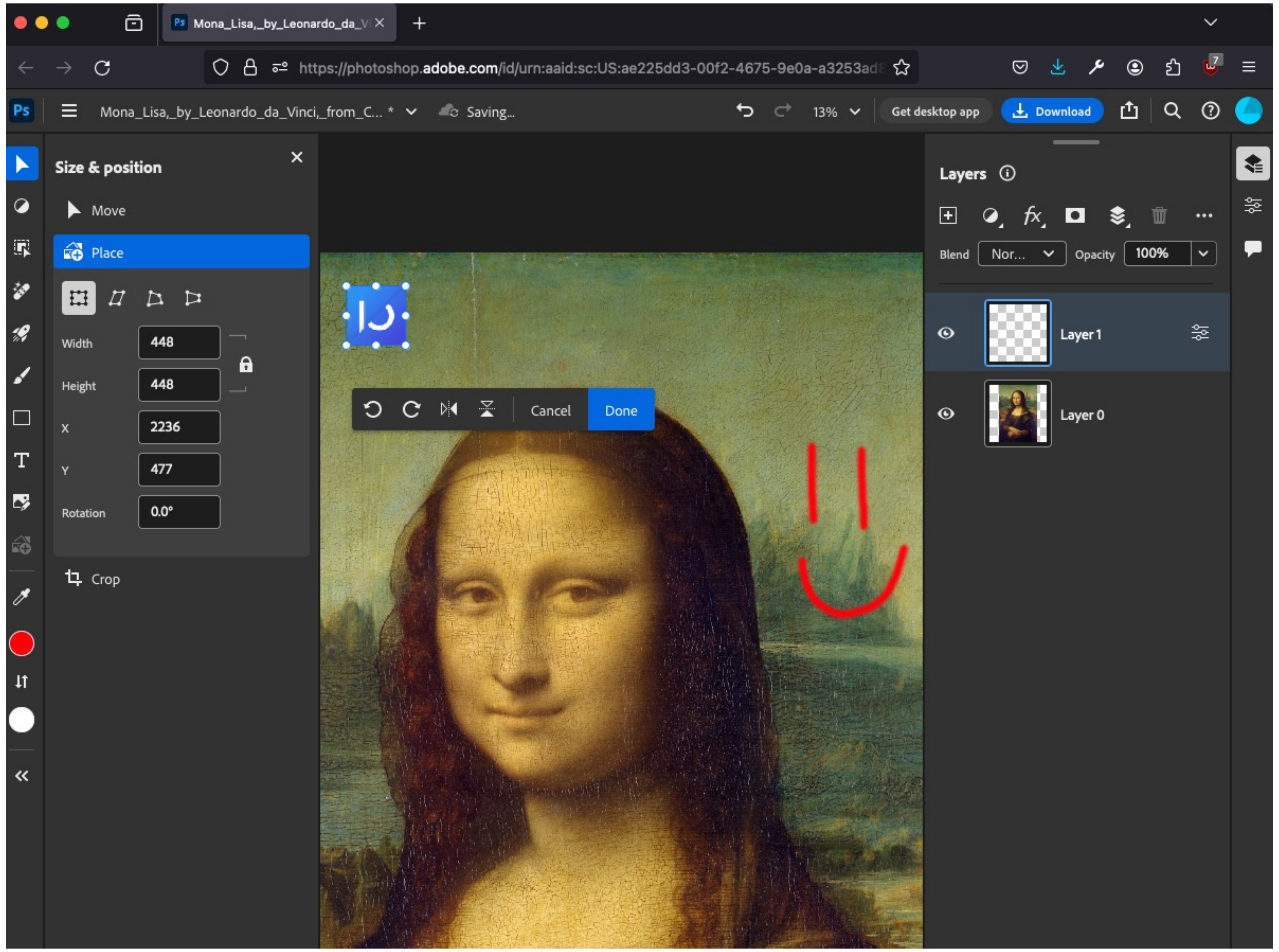
Sean Isom

CEO @ renderlet

linkedin.com/in/isom | @theisomizer | github.com/seanisom

DRAWING IN (BROWSER) WASM

- Solved problem?
- In production
  - Photoshop, AutoCAD, GMaps
- Almost a rewrite
  - Web is just different



https://photoshop.adobe.com/id/urn:aaid:sc:US:ae225dd3-00f2-4675-9e0a-a3253ad

Mona_Lisa,_by_Leonardo_da_Vinci,_from_C... *

Width 448

Height 448

X 2236

Y 477

Rotation 0.0°

Blend Nor...

Opacity 100%

THE TREACHERY OF CROSS PLATFORM

- Took 10+ years
- Canvas, WebGL, and WebGPU
  - It’s all just JavaScript…
    - Still a platform
- Uses bindings & shims

DRAWING IN (RAW) WASM

- … you can’t.
- Framebuffers?
- ML (wasi-nn)?
- WASI? Component Model?
- An idea…

INTRODUCING WANDER

- **WA**sm ren**DER**er
- Compile pipeline to Wasm
  - Graphics code and data
- Cross-platform runtime
  - Embed inside app (C++ API)
- Expose GPU to Wasm

wander

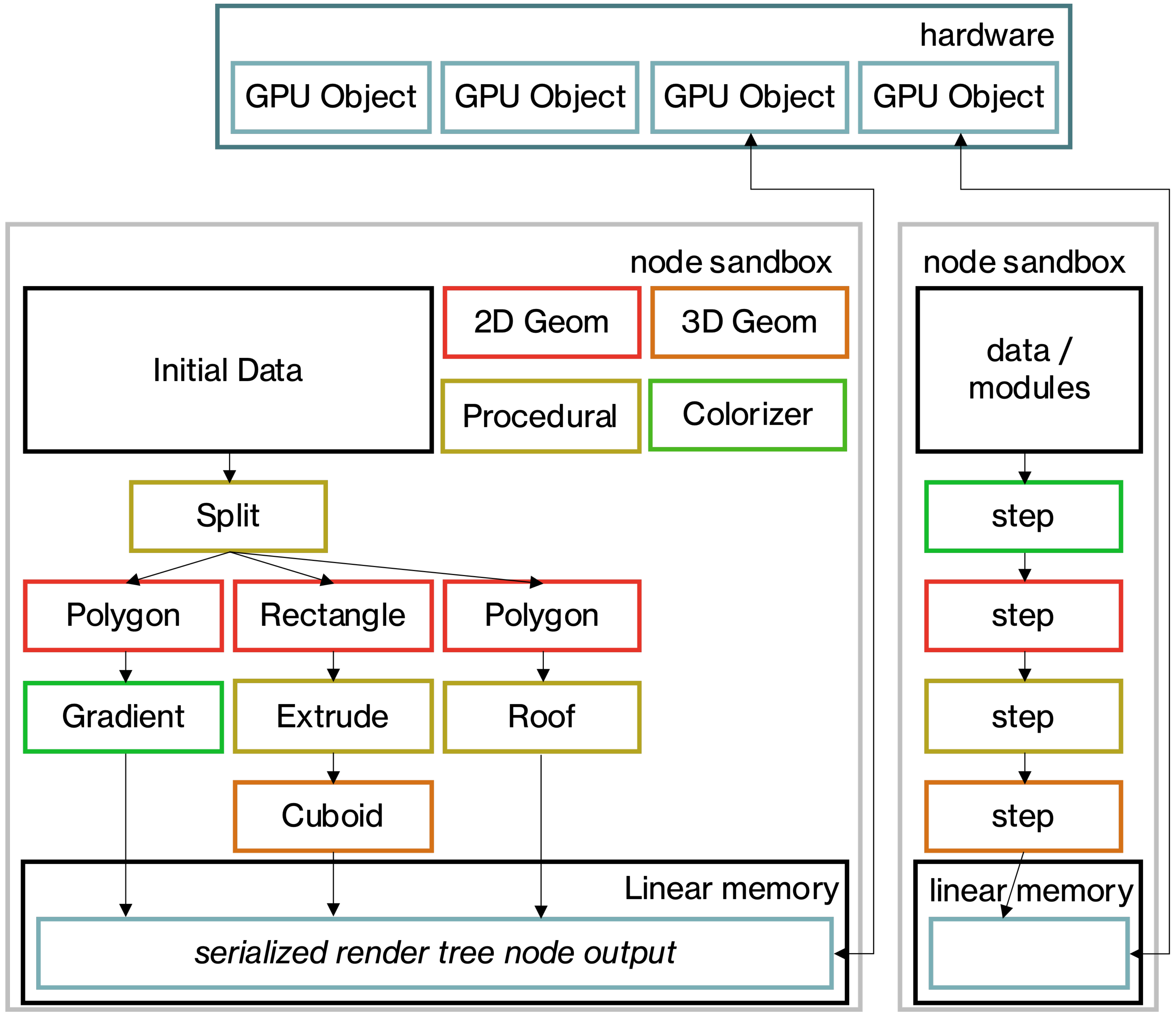
https://github.com/renderlet/wander

DESIGNING A RENDERER

- Don’t reinvent the wheel (GPU APIs & formats)
- Extremely high-performance (throughput & latency)
- Massively parallel (tasks & data)
- Incremental with Wasm:
  - Host: Talk to platform’s GPU, Guest: Talk to host via wire format

NODE / DOM MODEL

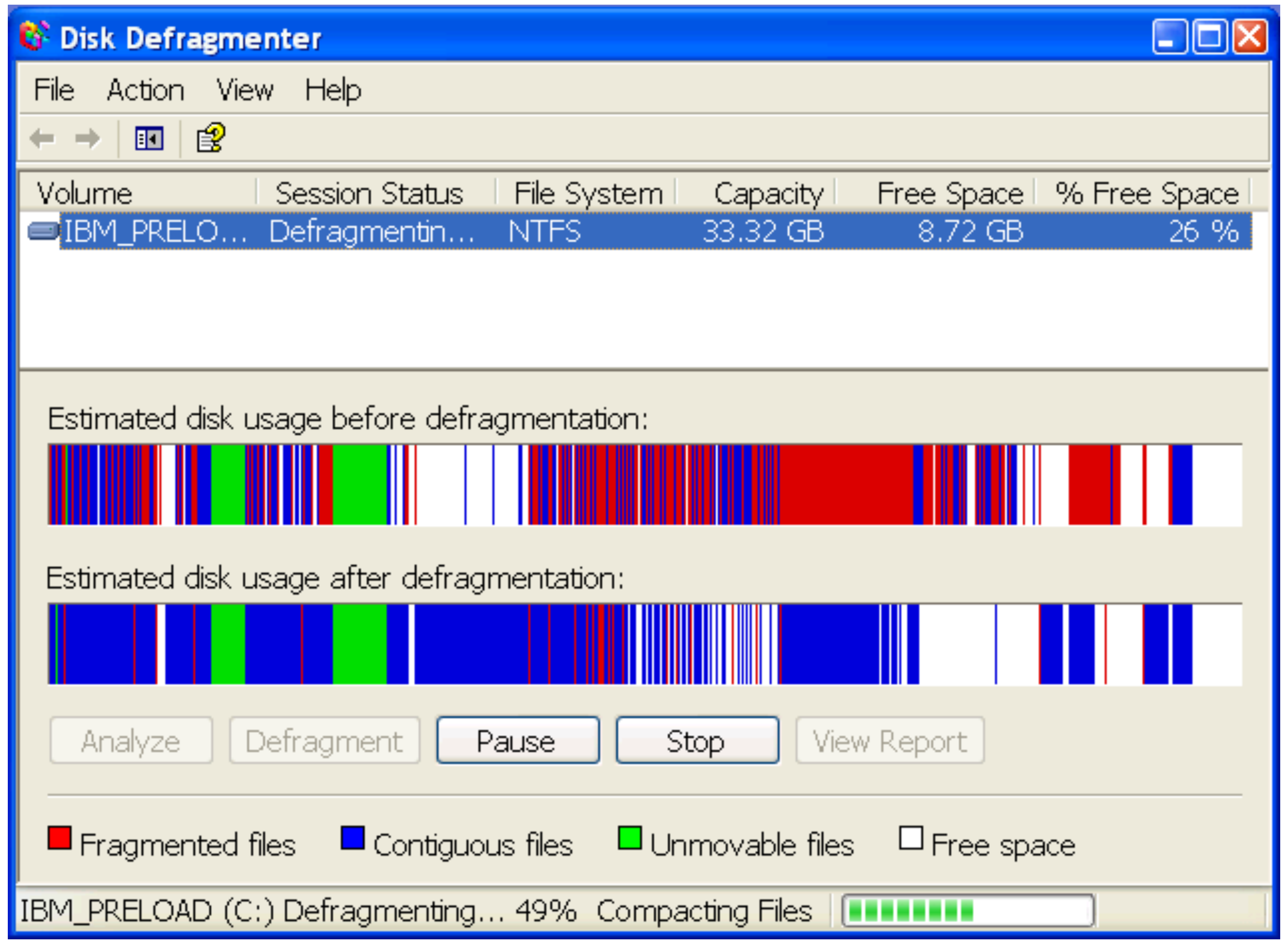
- Render Tree (DOM)
- Node
  - GPU object
- Steps
  - Access requires interrupt
- Linking



MEMORY MODEL

- Bad: individual buffers

- Better: host pooled buffers



Volume	Session Status	File System	Capacity	Free Space	% Free Space
IBM_PRELO...	Defragmentin...	NTFS	33.32 GB	8.72 GB	26 %

Estimated disk usage before defragmentation:

Estimated disk usage after defragmentation:

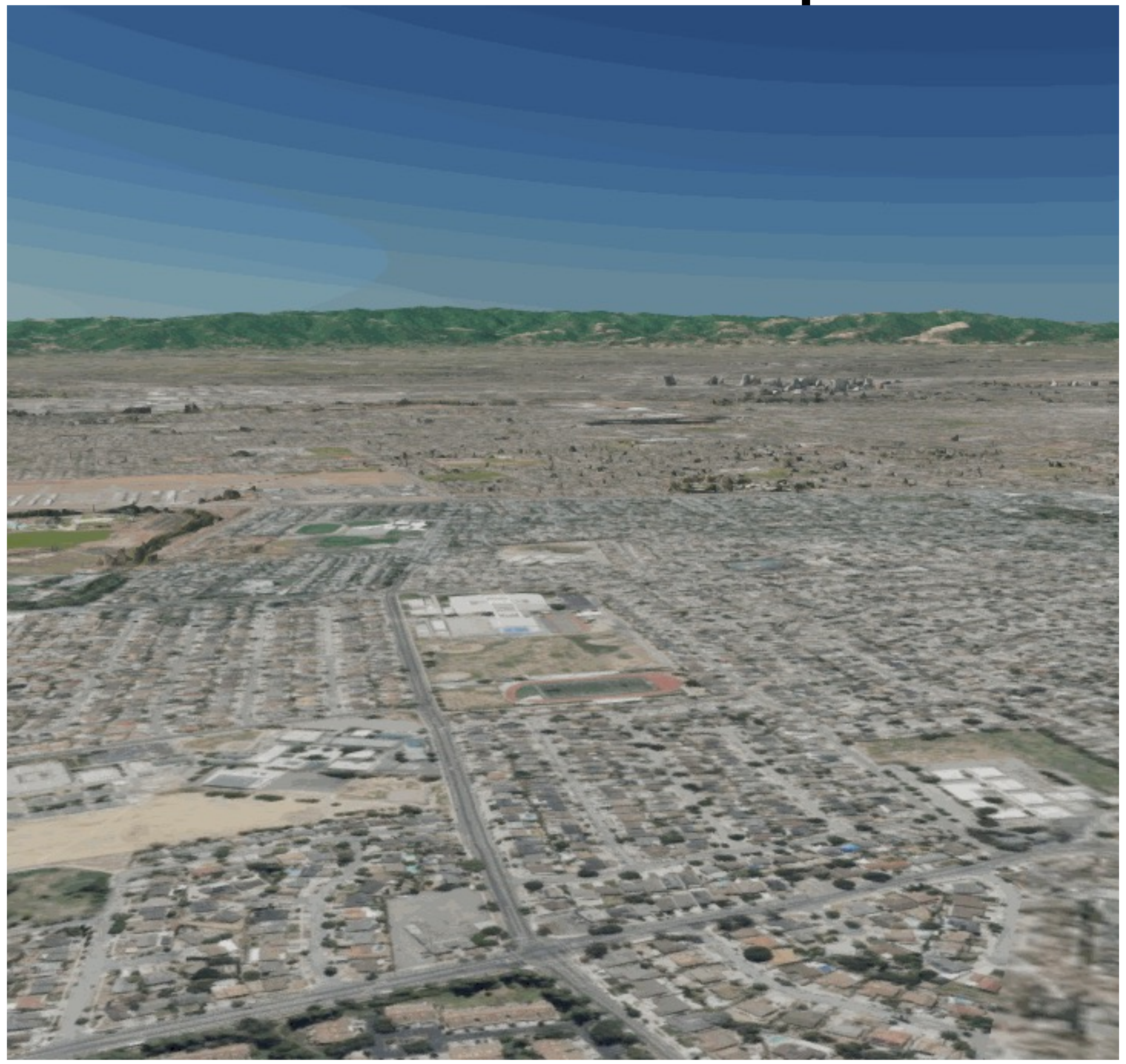
🟥 Fragmented files 🟦 Contiguous files 🟩 Unmovable files □ Free space

IBM_PRELOAD (C:) Defragmenting... 49% Compacting Files

- Future: shared memory across Wasms (multi-memory?)

PERFORMANCE

- ~75% of native speed – Wasm invocation overhead > Wasm opcodes
- Managing wasmtime is critical
  - Memory/Store sizing
  - Linker is slow
  - Reuse stacks



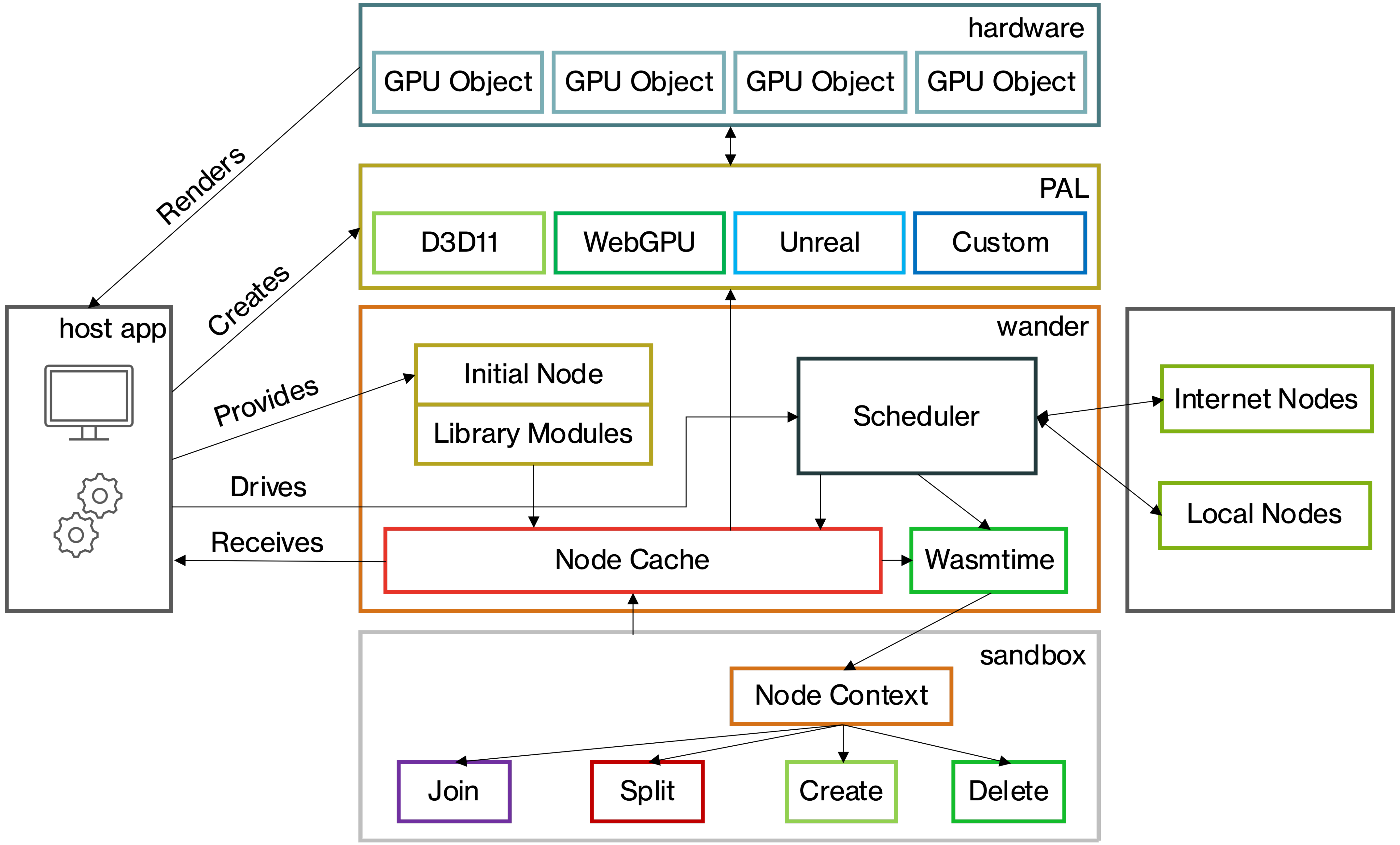
PARALLELISM

- Data Level Parallelism
  - Fine-grained threads per operation – context switches
    - Best left to shaders
- Task Level Parallelism
  - Specific – schedule large render trees with DAG
  - General – split trees per core

ARCHITECTURE V1

Wasm code on CPU generates buffers and wander uploads to GPU using platform's API

Host app owns shader and draws buffer data of a known format

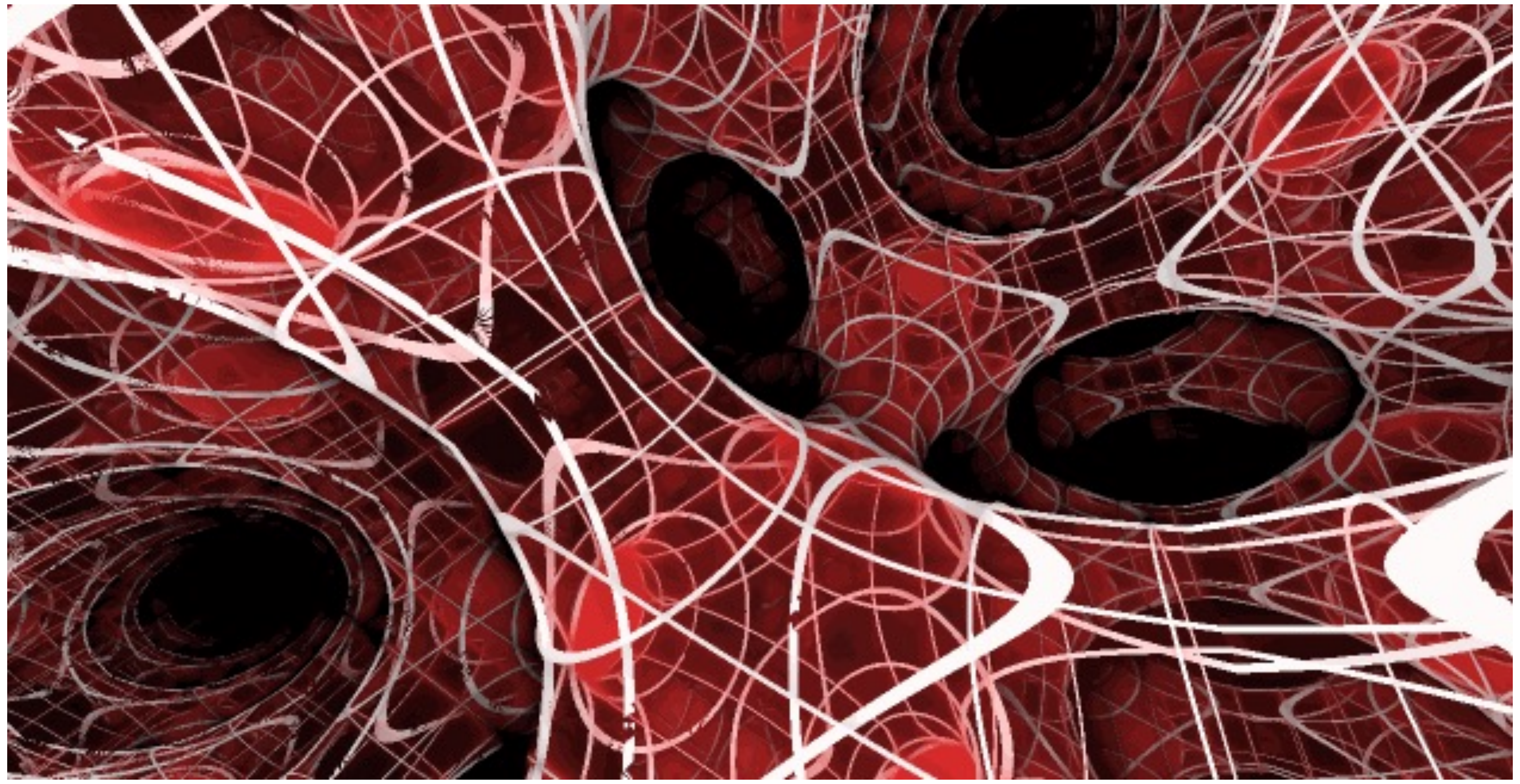


WAIT! HOW DO WE DRAW???

- Naïve: expose framebuffer to Wasm CPU Code
- Better: create GPU Code from Wasm
  - https://github.com/Aandreba/wasm2spirv
- Sophisticated: build shaders programmatically in Wasm guest

DESIGNING FOR SHADERS

- Dynamic compilation
- Uniforms
- Attachments
- Pipeline state

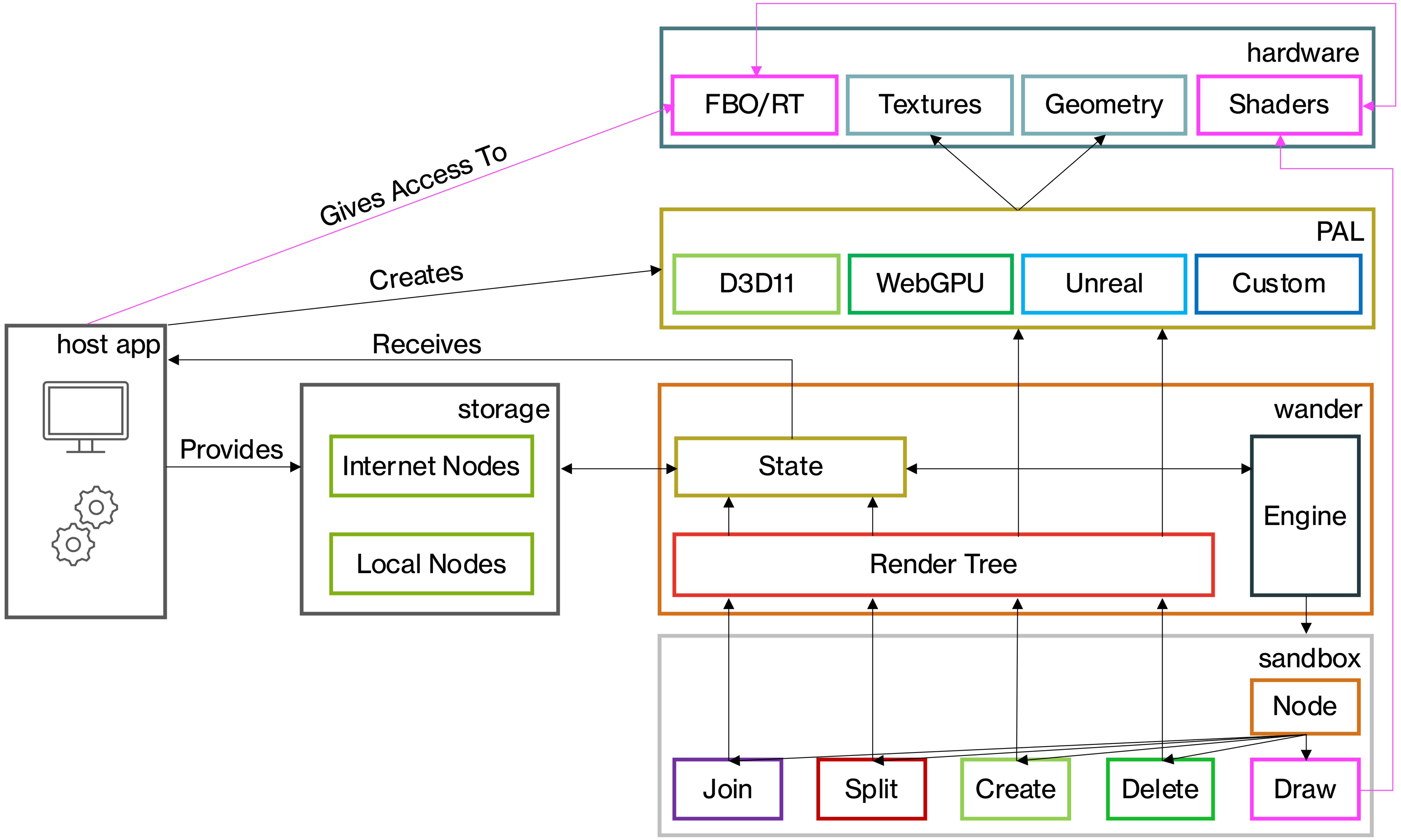


Credit: Fabrice Neyret - https://www.shadertoy.com/view/4lfSDn

ARCHITECTURE V2

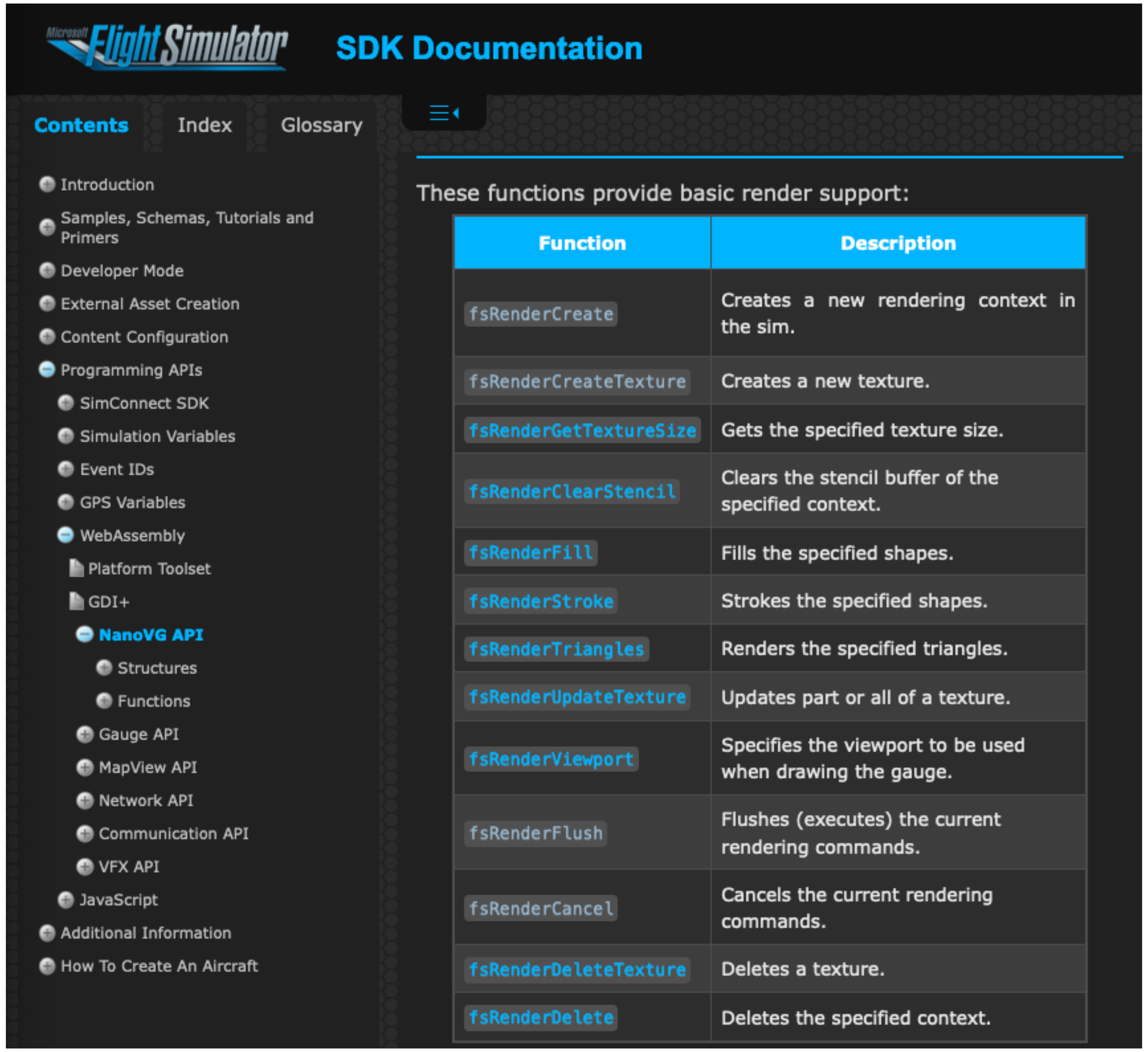
Host app gives wander its framebuffer and scene state

GPU gives wasm functions to manage shaders and draw directly to host app's framebuffer



HOW TO ACTUALLY EXPOSE THE GPU

- Host Functions (MSFS+, Flutter*)?
- Platform's API?
- Translation layer?
  - Bounds checking
  - Resource ownership validation



SDK Documentation

These functions provide basic render support:

Function	Description
`fsRenderCreate`	Creates a new rendering context in the sim.
`fsRenderCreateTexture`	Creates a new texture.
`fsRenderGetTextureSize`	Gets the specified texture size.
`fsRenderClearStencil`	Clears the stencil buffer of the specified context.
`fsRenderFill`	Fills the specified shapes.
`fsRenderStroke`	Strokes the specified shapes.
`fsRenderTriangles`	Renders the specified triangles.
`fsRenderUpdateTexture`	Updates part or all of a texture.
`fsRenderViewport`	Specifies the viewport to be used when drawing the gauge.
`fsRenderFlush`	Flushes (executes) the current rendering commands.
`fsRenderCancel`	Cancels the current rendering commands.
`fsRenderDeleteTexture`	Deletes a texture.
`fsRenderDelete`	Deletes the specified context.

* https://github.com/flutter/flutter/wiki/Flutter-GPU

+ https://docs.flightsimulator.com/html/Programming_Tools/WASM/Low_Level_API/NanoVG_API.htm

INTRODUCING WASI-WEBGPU

- Level 1 WASI Proposal
  - https://github.com/WebAssembly/wasi-webgpu
  - https://github.com/MendyBerger/wasi-webgpu
- Convert WebIDL to WIT
- Component Model wraps native WGPU

WANDER: WHAT'S NEXT

- Does wasi-webgpu solve everything?
- wander thesis - rendering should be:
    - Embeddable
    - Platform independent
    - Higher level

DEMO

wander

THANKS!

Sean Isom

More info

renderlet

linkedin.com/in/isom | @theisomizer | github.com/seanisom