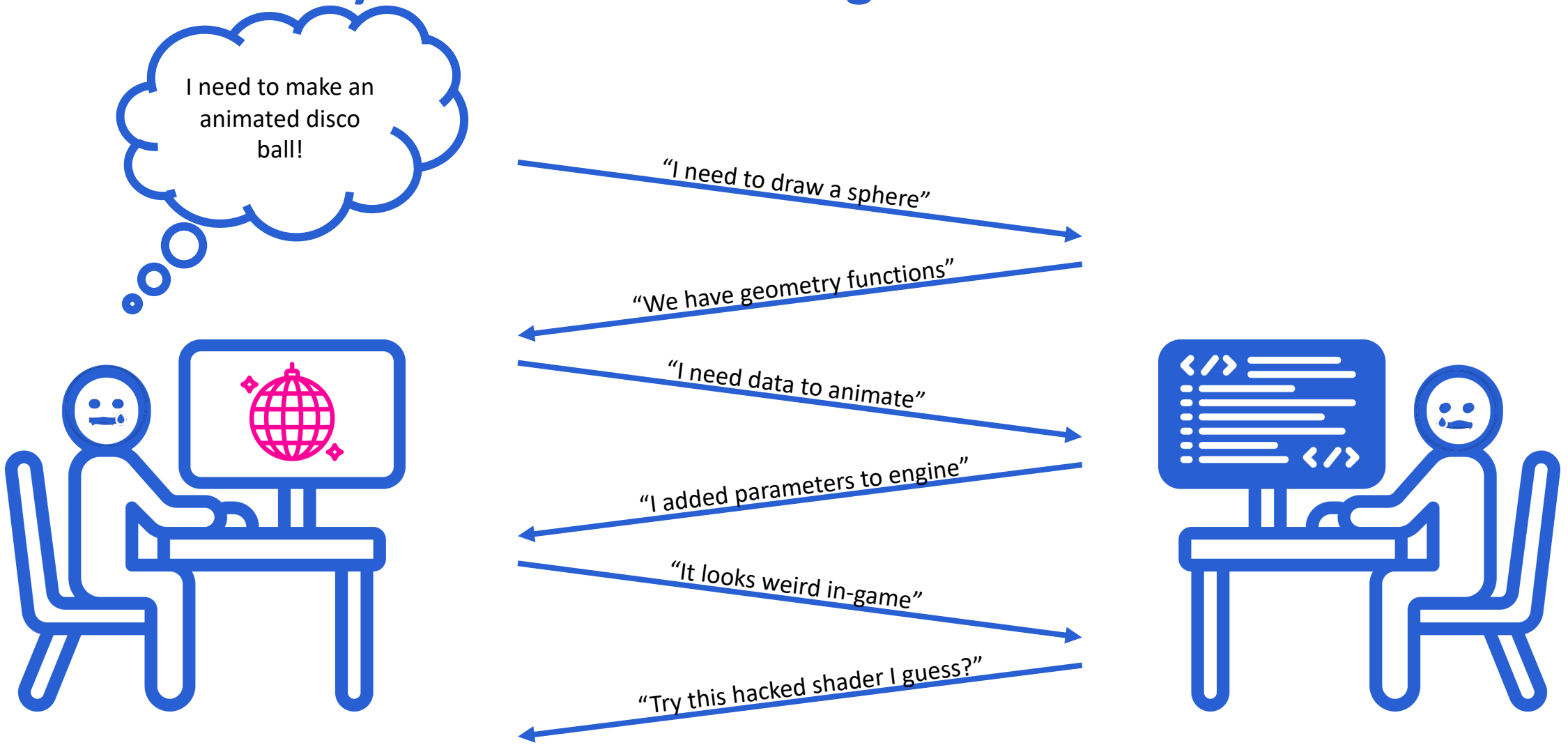
renderlet

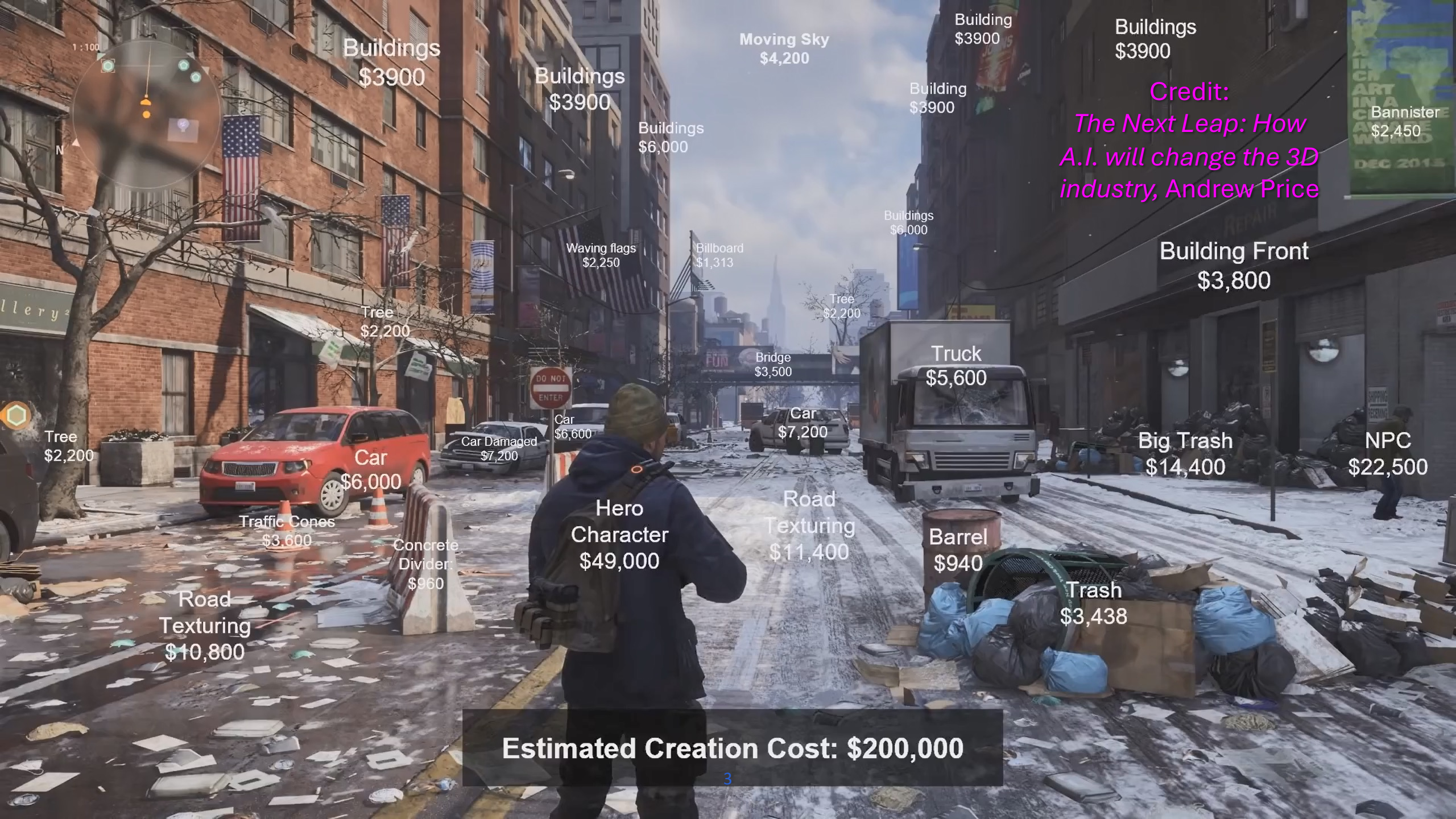
GPUs Unleashed!
Make Your Games More Powerful With wasi-gfx

Sean Isom, CEO @ Renderlet

Mendy Berger, Sr. Software Engineer @ Renderlet

The bits and bytes of 3D scenes in games



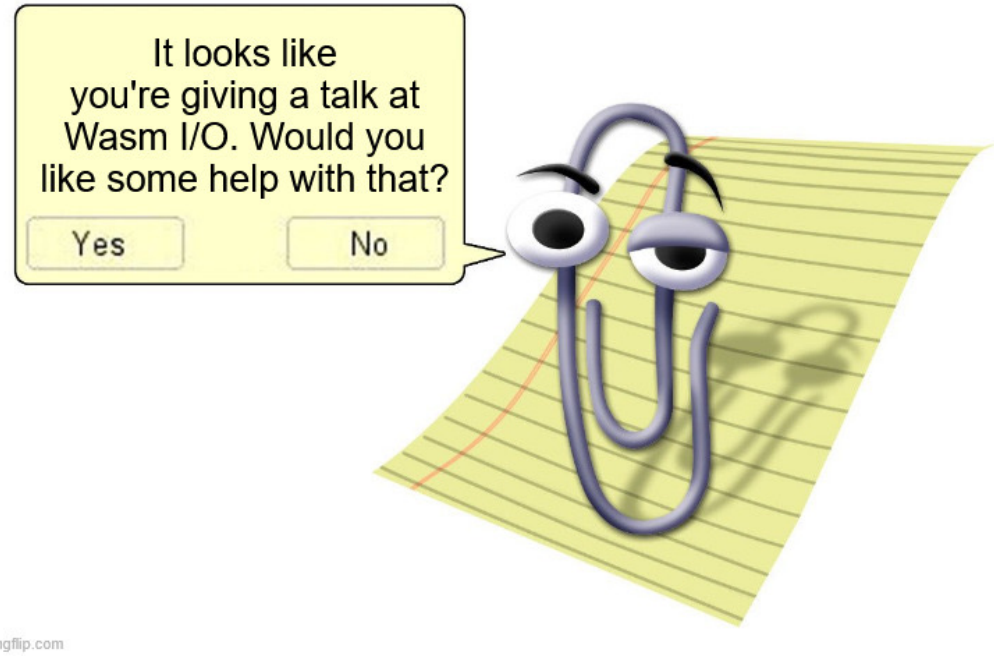

Buildings
$3900
Buildings
$3900
Moving Sky
$4,200
Building
$3900
Buildings
$3900
Credit:
The Next Leap: How A.I. will change the 3D industry, Andrew Price
Bannister
$2,450
Buildings
$6,000
Building
$3900
Buildings
$6,000
Waving flags
$2,250
Billboard
$1,313
Building Front
$3,800
Tree
$2,200
Tree
$2,200
Bridge
$3,500
Truck
$5,600
Car
$7,200
Car
$6,600
Car Damaged
$7,200
Tree
$2,200
Car
$6,000
Big Trash
$14,400
NPC
$22,500
Traffic Cones
$3,600
Hero
Character
$49,000
Road
Texturing
$11,400
Barrel
$940
Concrete
Divider
$960
Trash
$3,438
Road
Texturing
$10,800
Estimated Creation Cost: $200,000

The big problem: content

- Content is expensive to build!
- Exploding scope, quantity, and quality bar
- Data is intertwined with code for 3D
- Not necessarily reusable
- GLTF is for static data

Plugins to the rescue

- Rendering becomes plugin-driven
- Definition: 3rd party code called over a defined interface
- Plugins come in many shapes and sizes
  - WinAmp
  - Photoshop
  - Kubernetes CRDs
  - Browser APIs
  - Clippy
- Games are different! Plugins need a GPU



Wasm + GPUs = ?

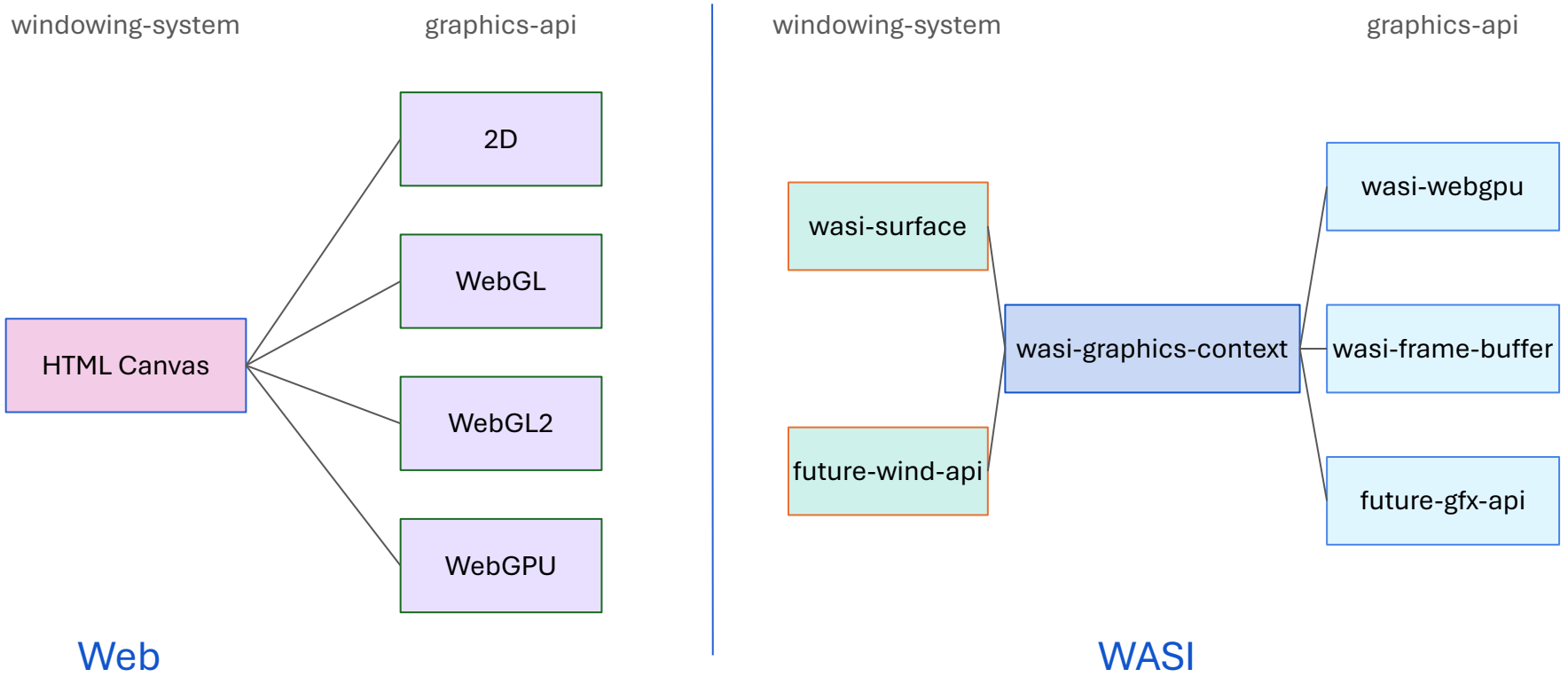
- WebGPU: multitenant GPU security
- Wasm: a true native sandbox (not Flash)
- In-browser:
  - Browser GPU APIs over JS Bridge
  - Emscripten / wasm-bindgen
- Outside browser:
  - ?
  - Wasm Component Model to the rescue!

Introducing wasi-gfx

- Machine translate WebGPU's WebIDL to wit
- Extend with support to connect with surfaces and IO
- Window and graphics API agnostic
- Reference host implementation: wasi-gfx-runtime
  - https://github.com/wasi-gfx/wasi-gfx-runtime

Demo: Bevy

wasi-gfx architecture



Considerations for drawing into a window

- “Hooking” backbuffer is straightforward
  - Consider subregions for “applets” like UI
  - Fullscreen quad / not copy
- Drawing into scene in 3D is difficult
  - Host context – deferred targets, screen-space effects, etc.
  - Shared camera/materials between host and guest
  - Consider rendering into own texture in 3D with alpha
  - Also consider using host app’s shader on plugin data

Demo: Maps

Using wasi-gfx in a game

- Like a box of “Lego” parts
- Need a plugin runtime
  - Renderlet is building this
- Adoption hurdles:
  - Games are C++
  - Games demand performance
  - Games aren’t WGPU


C++/DX12
Rust/WGPU

Hurdle: Integrating C++ and Rust

- Plugins written in any language
- Host for wasi-gfx requires Rust*
  - WGPU
  - Wasmtime (Component model)
  - wasi-gfx-runtime
- Maintain separate build tree
  - C ABI “DLL”
  - Bindings via cbindgen / cxx
  - Lots of #[repr(C)] and void*
  - Clean C++ API wrapper

Hurdle: Real-time performance

- Wasm usually is fast on CPU (~80%)
- Wasm shouldn’t be the bottleneck
- Manage resources intentionally
  - Don’t readback data to CPU
  - Move to GPU-driven rendering
  - Avoid allocs with memory mapping

Hurdle: Integrating DX12 and WebGPU

- WebGPU implemented through WGPU in wasi-gfx-runtime
- WGPU uses DX12 backend on Windows
- Change WGPU to accept device created by host
- “as_hal” -> Get guest-created WGPU resource as DX12 for host
- “from_hal” -> Wrap host-created DX12 resource for WGPU guest
- Only required to share small number of resources

Introducing:
Renderlet Plugin Runtime

Demo: Adria

Next: Sane data models for plugins

Don’t

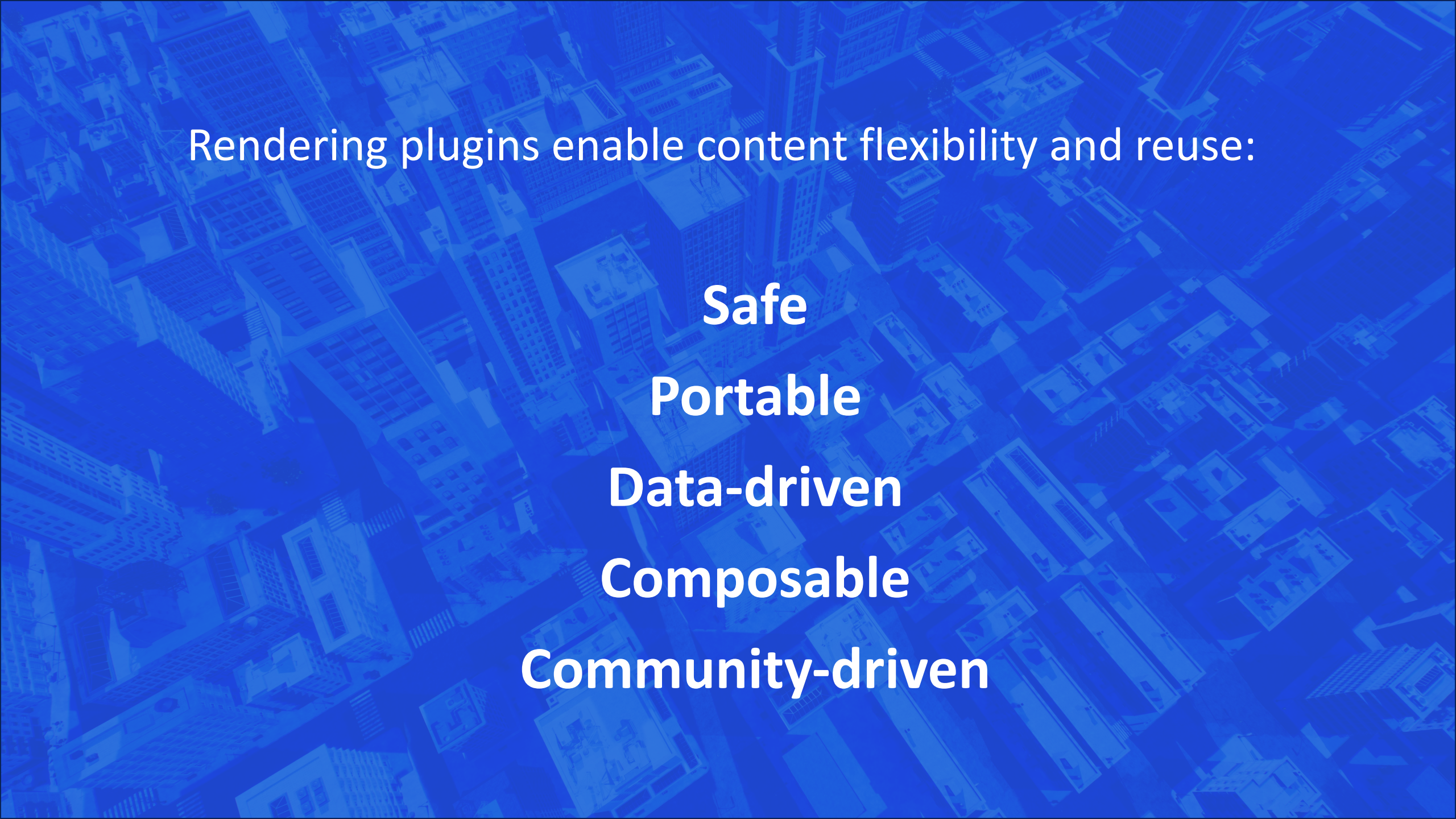
- Export entire host app API (headers)
- Build bespoke, one-off mappings

Do

- Use automatic mapping systems
- Use concurrency model of host app

Capability and resource restriction

- Capability restriction depends on host
  - “Does this plugin need full WebGPU API”
  - Can it mine bitcoin with arbitrary shaders?
  - Maybe restrict
- Resource restriction depends on frame / object budget
  - CPU time – “Fuel”
  - GPU time – Queries / Indirect?
  - VRAM – “heap / plugin”
  - Object count / format


Rendering plugins enable content flexibility and reuse:
Safe
Portable
Data-driven
Composable
Community-driven

renderlet

Interested in plugins? Come chat with us!

Sign up for updates: **renderlet.com**

Follow along on Twitter: **@_renderlet**

sean@renderlet.com