

Arcan

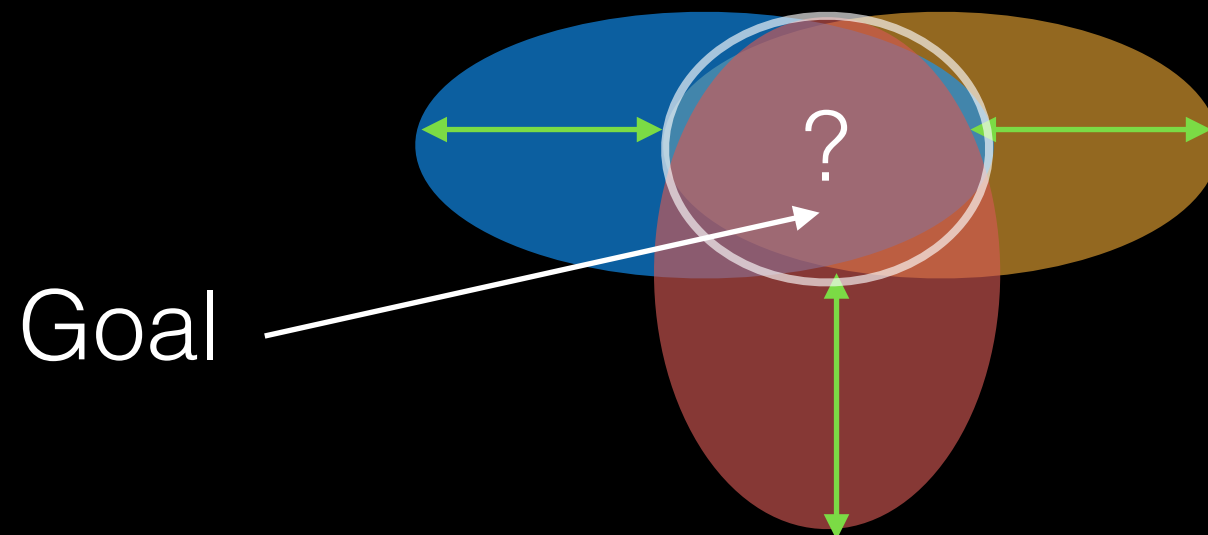
Free (BSDv3+a little GPLv2) { display “server”
portable, scriptable { game engine
realtime multimedia framework }

Forms of Contact ordered by estimated success-rate (high -> low)

Github	github.com/letoram/arcan
IRC	#arcan @ irc.freenode.net
Twitter	@arcan_fe
Web	arcan-fe.com
E-Mail	contact@arcan-fe.com

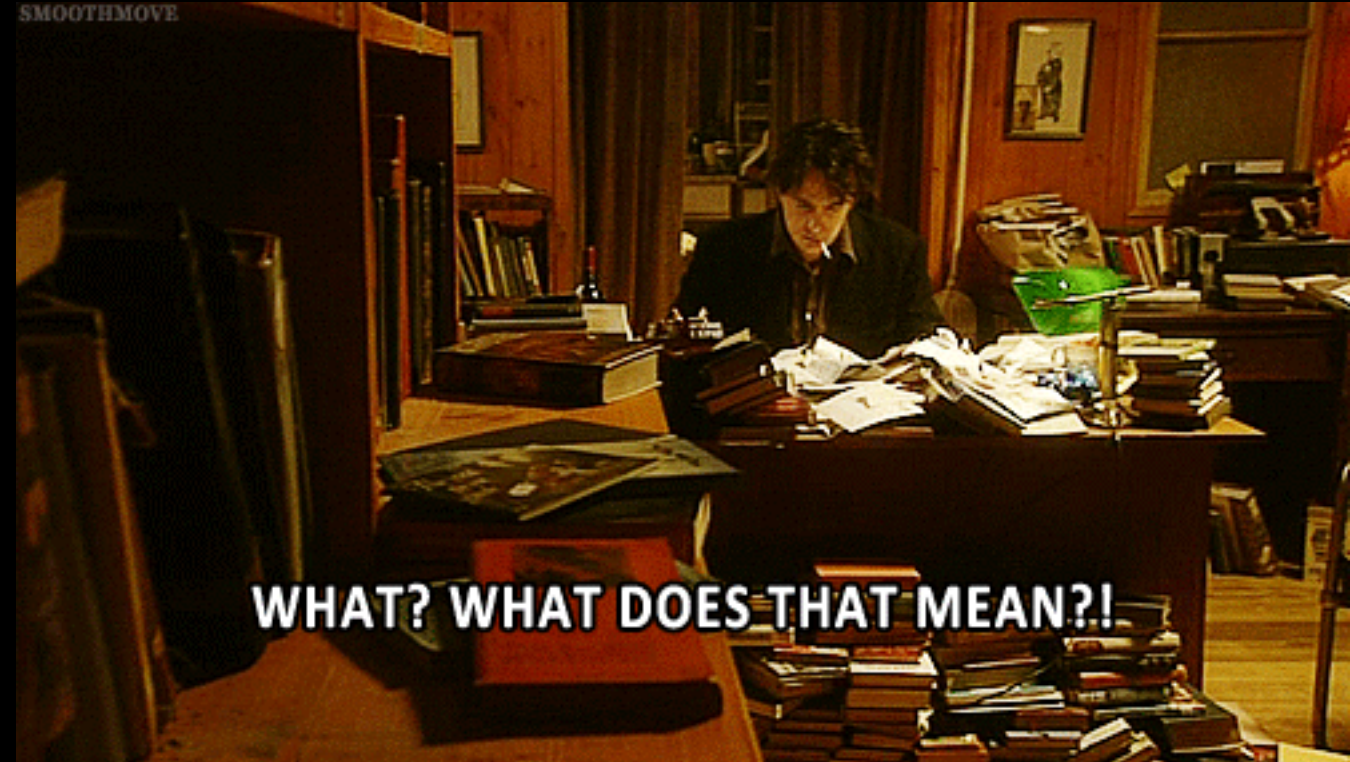
Idea

- * Look for a useful intersection between *typically* distinct (display server, game engine, streaming multimedia processing / low- mid- level graphics)
- * Make the 'last mile' scriptable
- * Emphasize minimalism and portability



“Special” Challenges

- * **Display Server** (X.org, DWM, Quartz, SurfaceFlinger)
 - * Privileged (it's not just about *root*)
 - * External producers & consumers (bad mix with privileged)
 - * Low level device integration (Monitors, Keyboards, ...)
 - * Power Management
- * **Game engine**
 - * Complex input models
 - * Adaptive soft realtime (Quality of Experience)
 - * High variability in GPUs (and their drivers and APIs)
- * **Multimedia processing**
 - * Assymetric Loads
 - * Complex / Insane Data Formats
 - * Timing sensitive, stream de-multiplexation
 - * Heavy / unsatisfiable buffering requirements

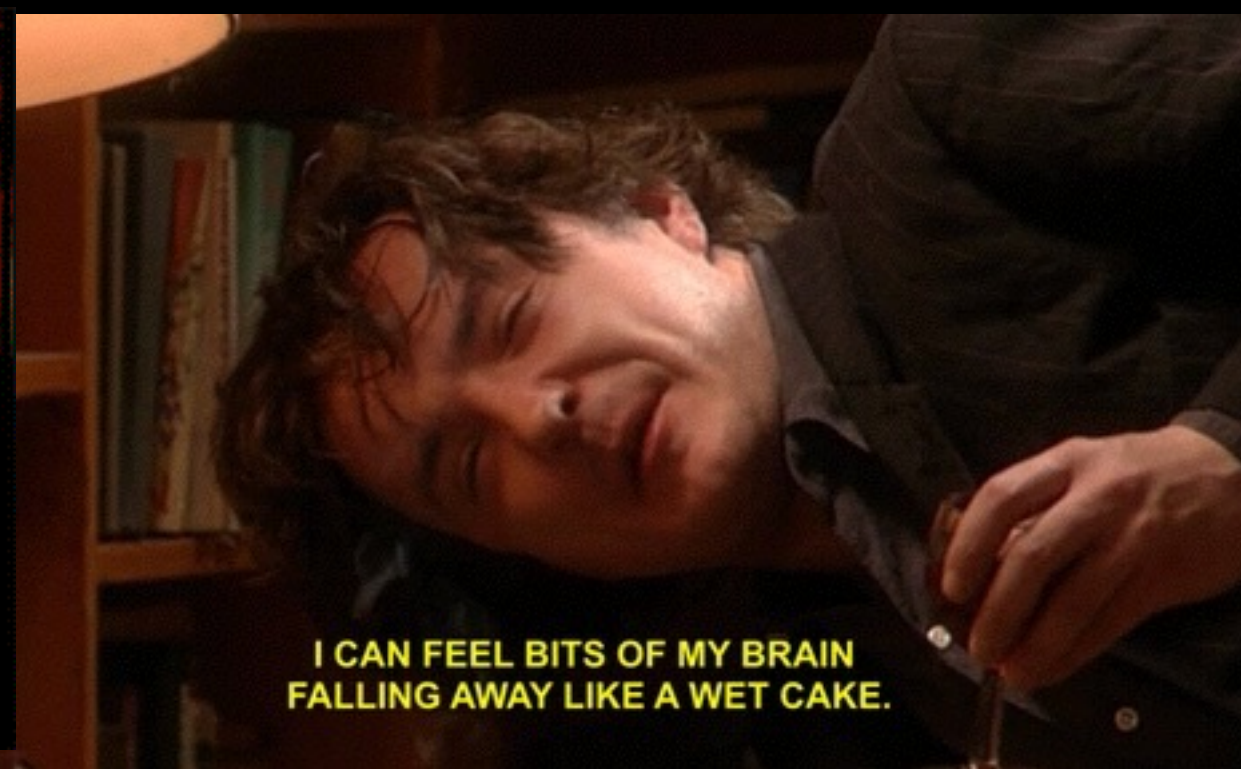


WHAT? WHAT DOES THAT MEAN?!



What the hell is this?

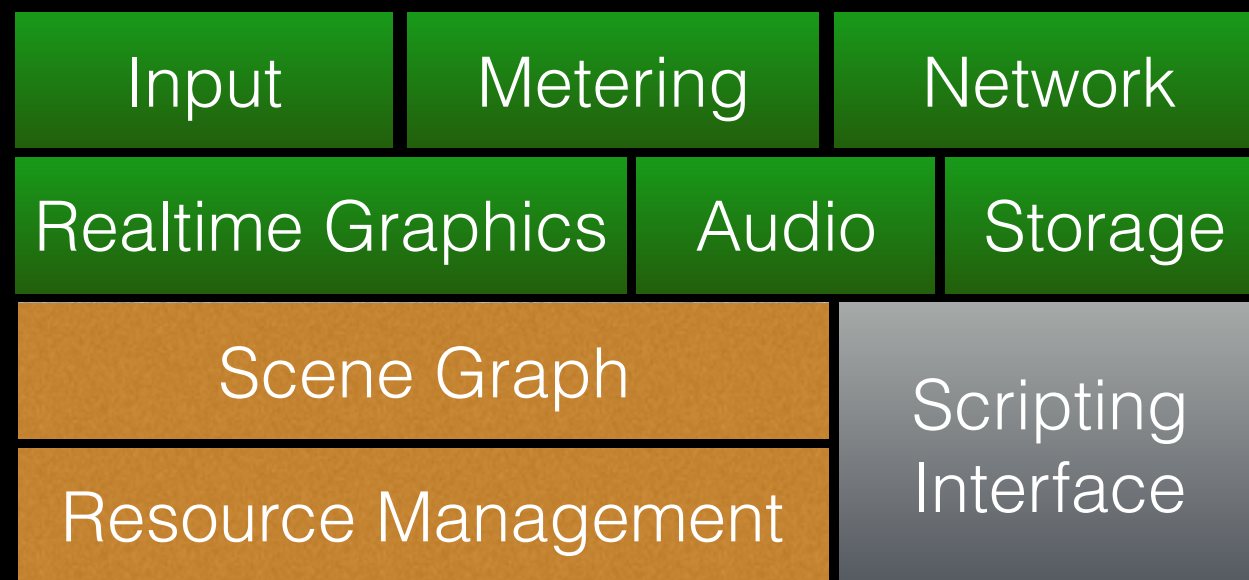
Fast Forward A Few Thousand Hours
(and a terrifying amount of *wine* and *coffee*)



I CAN FEEL BITS OF MY BRAIN
FALLING AWAY LIKE A WET CAKE.

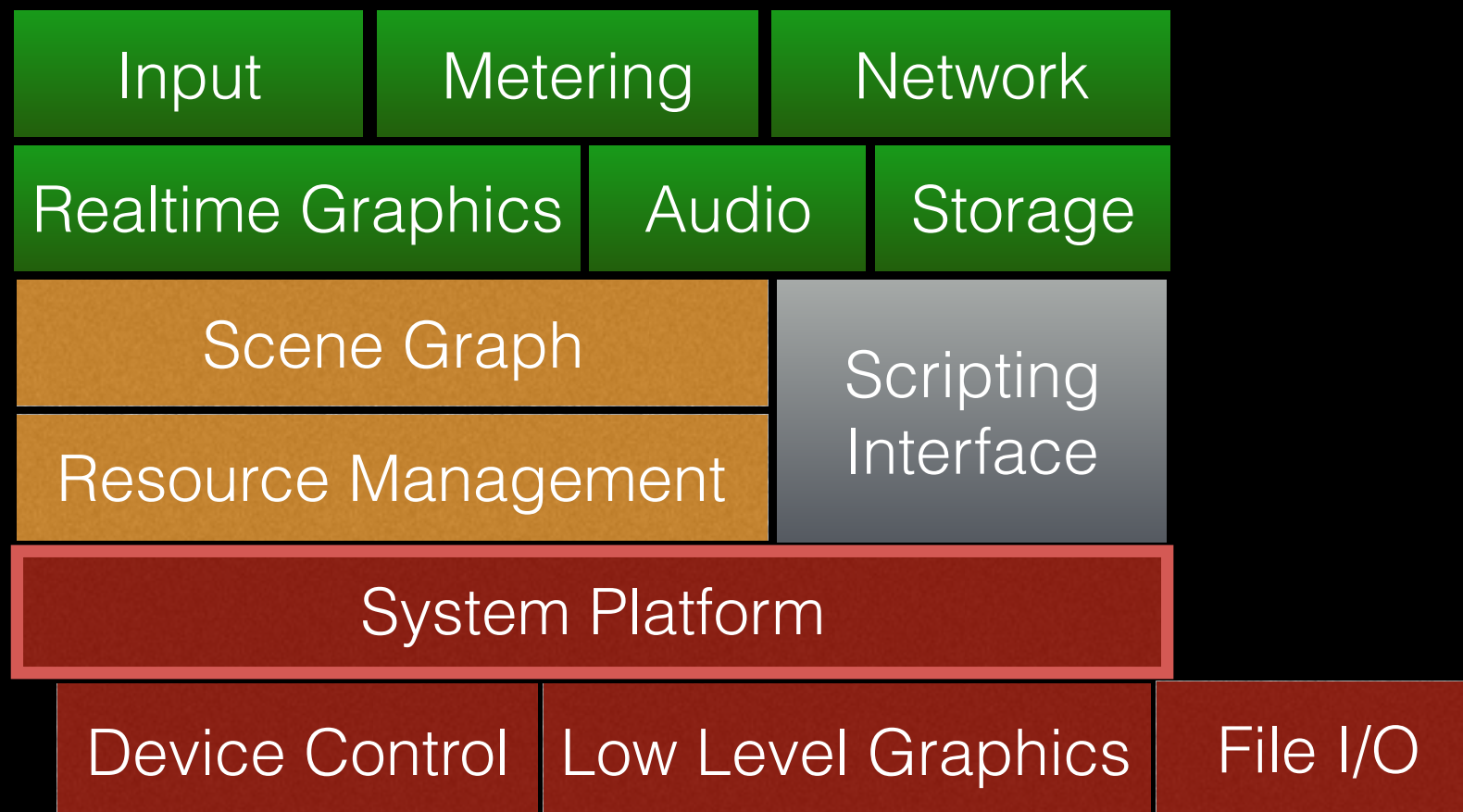
Recipe

1. Take a game-engine



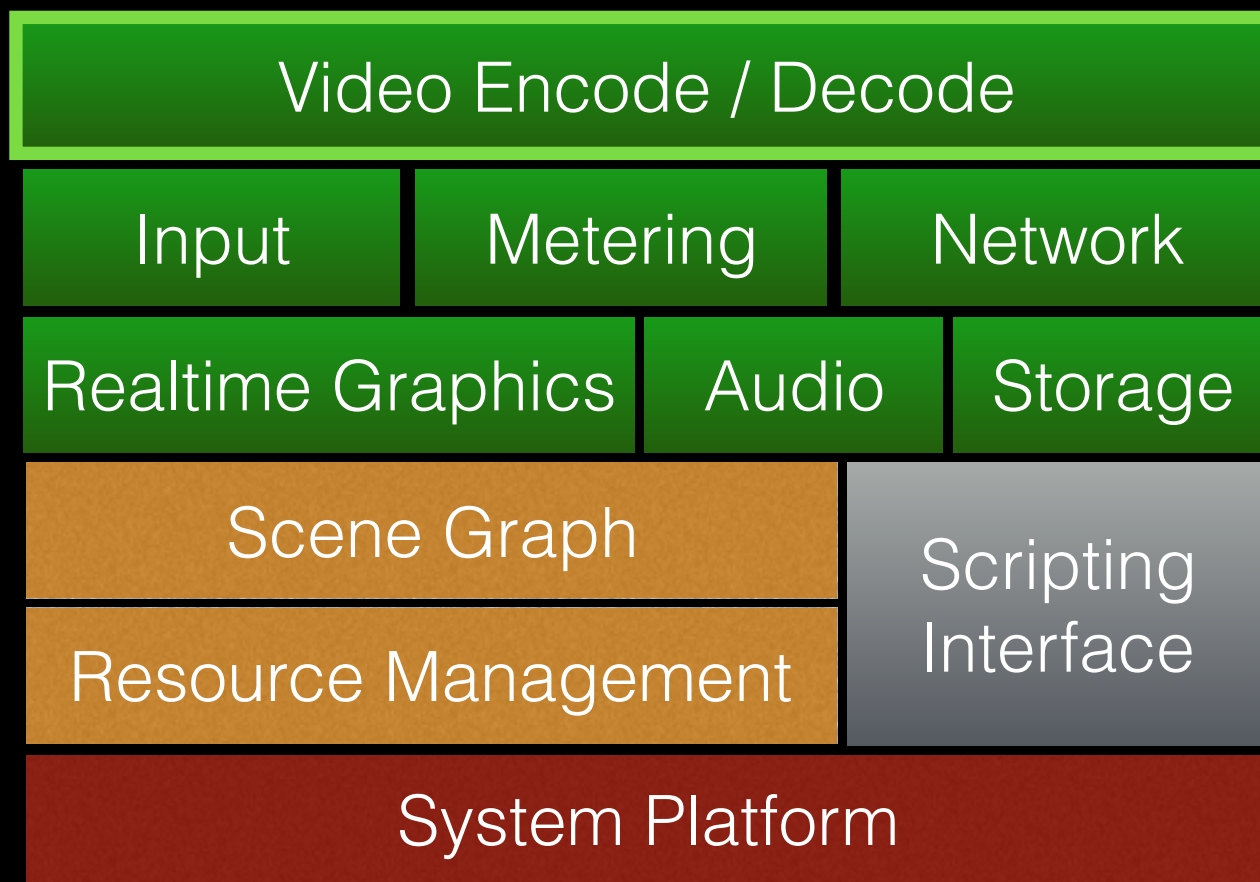
Recipe

2. Make it **Portable**



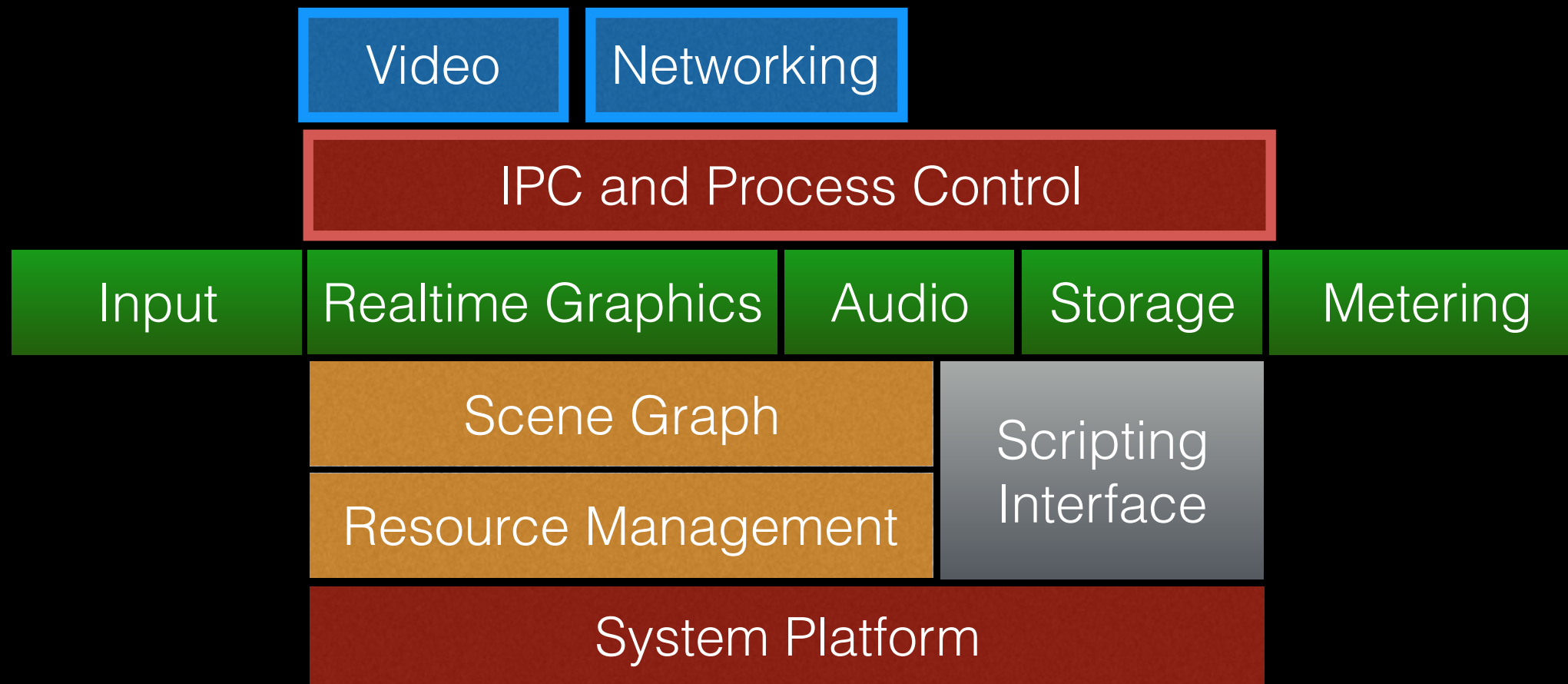
Recipe

3. Add **Streaming Media** Support



Recipe

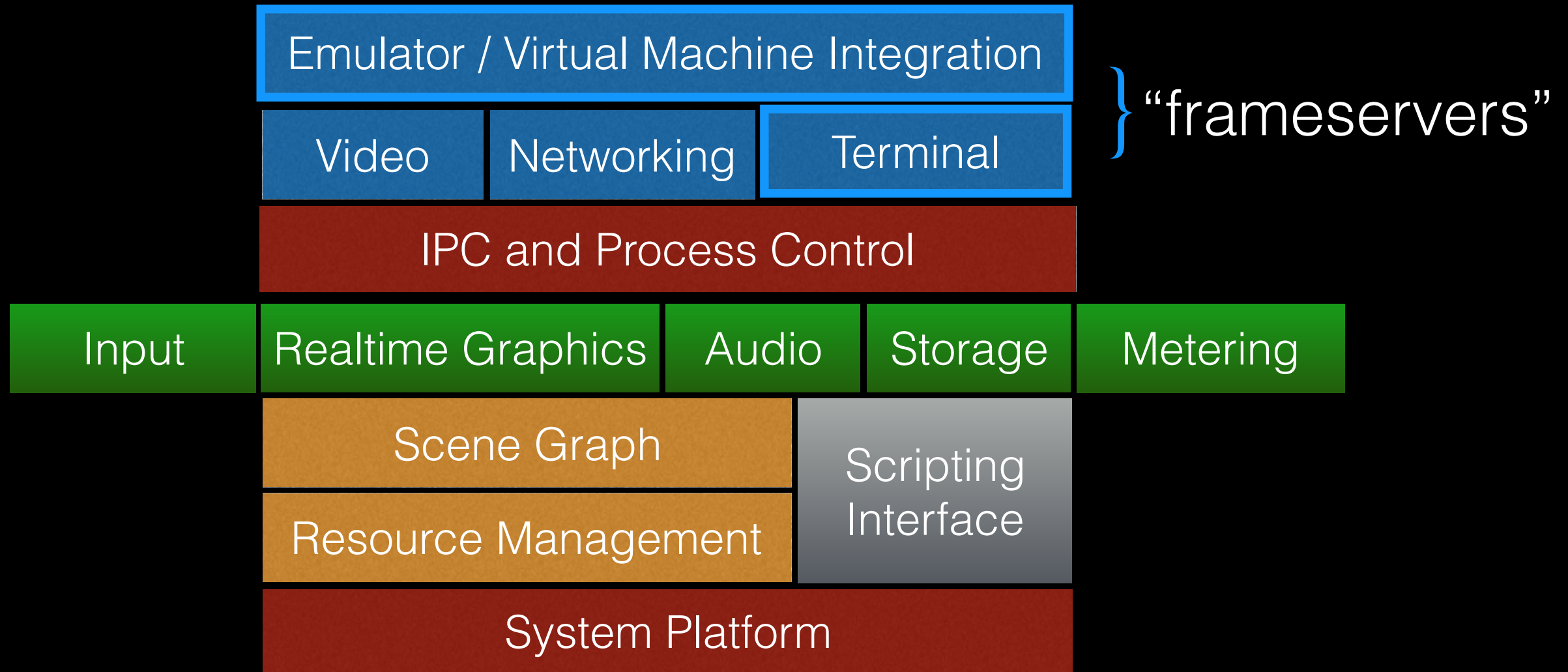
4. Add **Process Separation** (for resilience)



Recipe

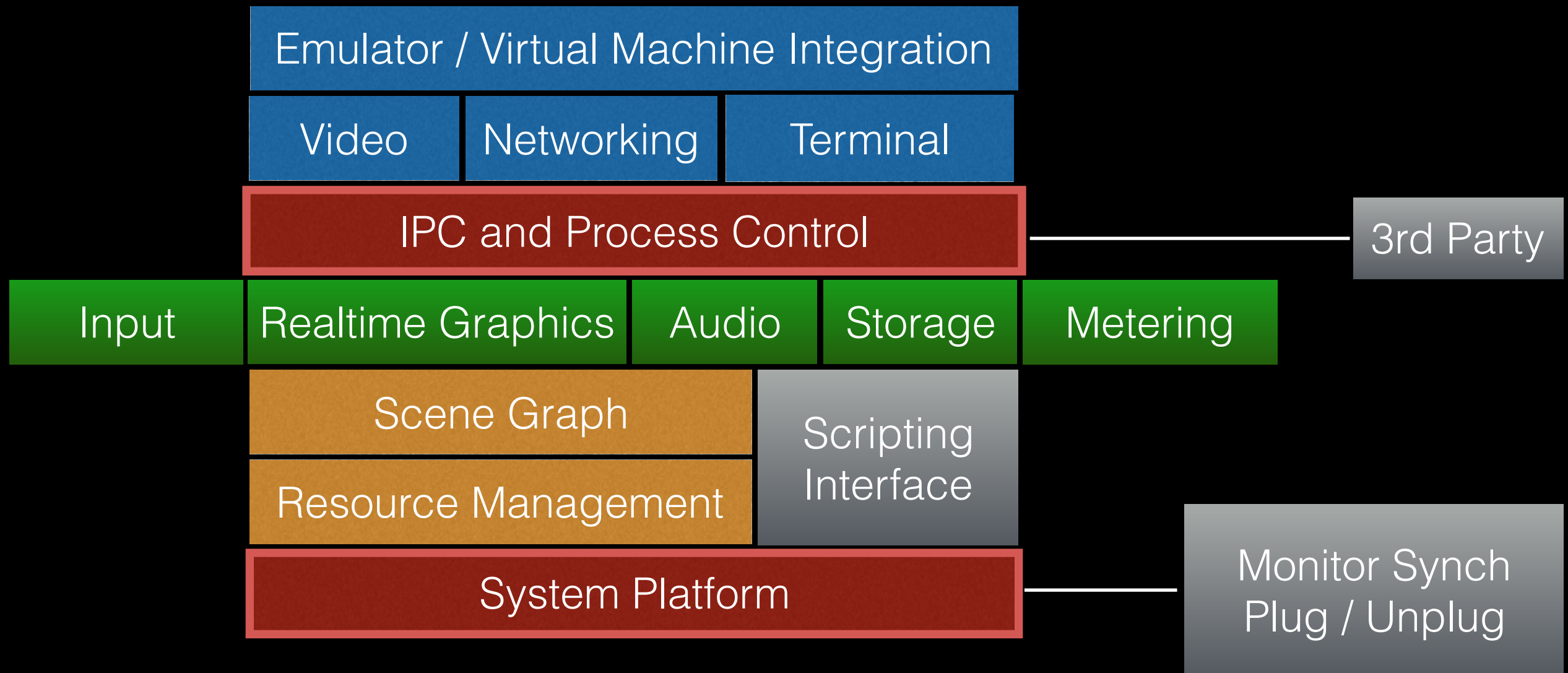
5. Expand Feature Set

[indirectly improve and harden IPC and related API]



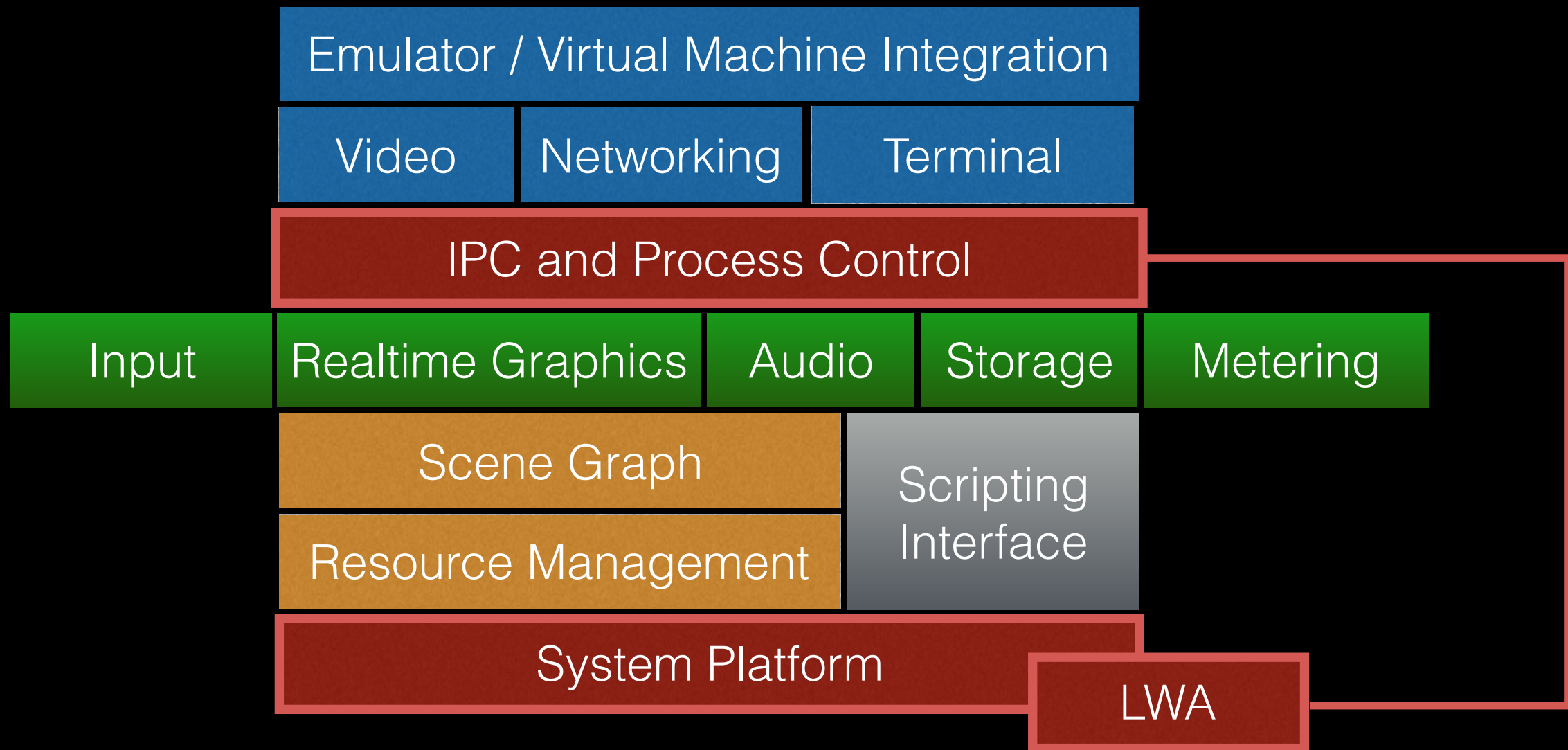
Recipe

6. Display Control + External Connections



Recipe

7. Allow nesting, chaining



"Lightweight Arcan"

- build where A/V/I platform outputs to IPC interface

Meanwhile....

- * Iteratively develop **proof-of-concepts**
- * to **(de,re)fine** scripting interface
- * establish support- scripts, code patterns
- * **locate, evaluate** and **improve** design rough spots

PoC Name:

Gridle

AWB

Senseye

Durden

Role:

Home-theater / Graphical FE

Classic “Fun” Desktop Interface

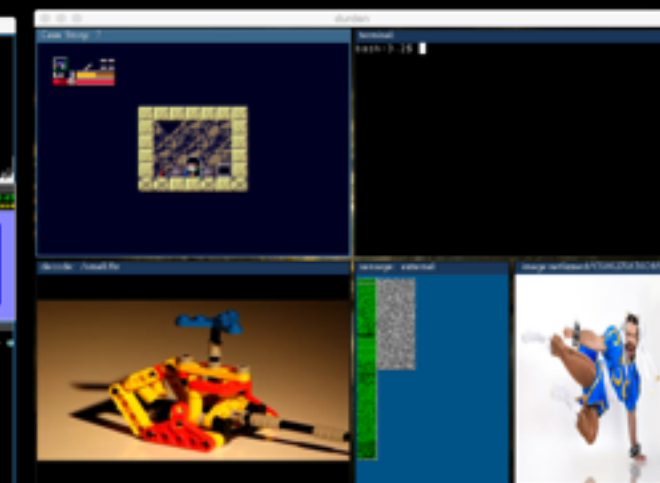
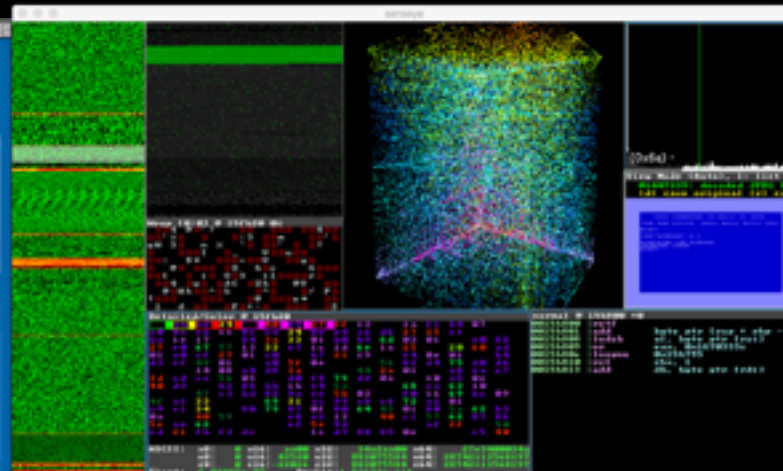
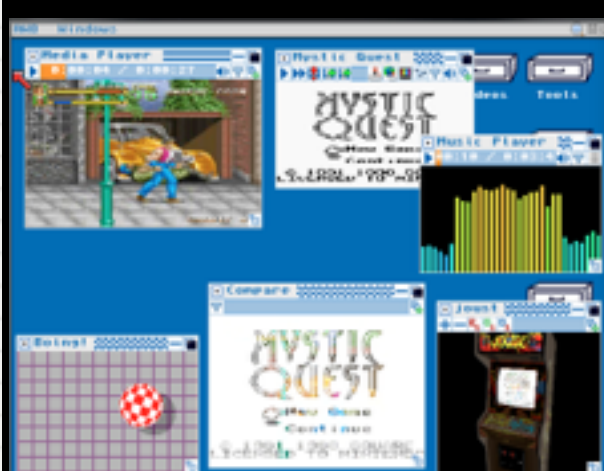
Debugging / Reversing tool

Desktop Environment

Status:

Abandoned

Supported



Arcan<Gridle>

~2011

HTPC- like interface



Improved:

- **Input Model** (support custom usb gamepads, multiple keyboards)
- Tons of **asynchronous- related bugs** squashed (background tiles are all videos from separate processes)
- **State Management** (suspend/resume/serialize external processes, minimizing resource footprint)
- Helped **Define the graphics API** that was needed for the advanced effects (simulating damaged CRTs, ..)

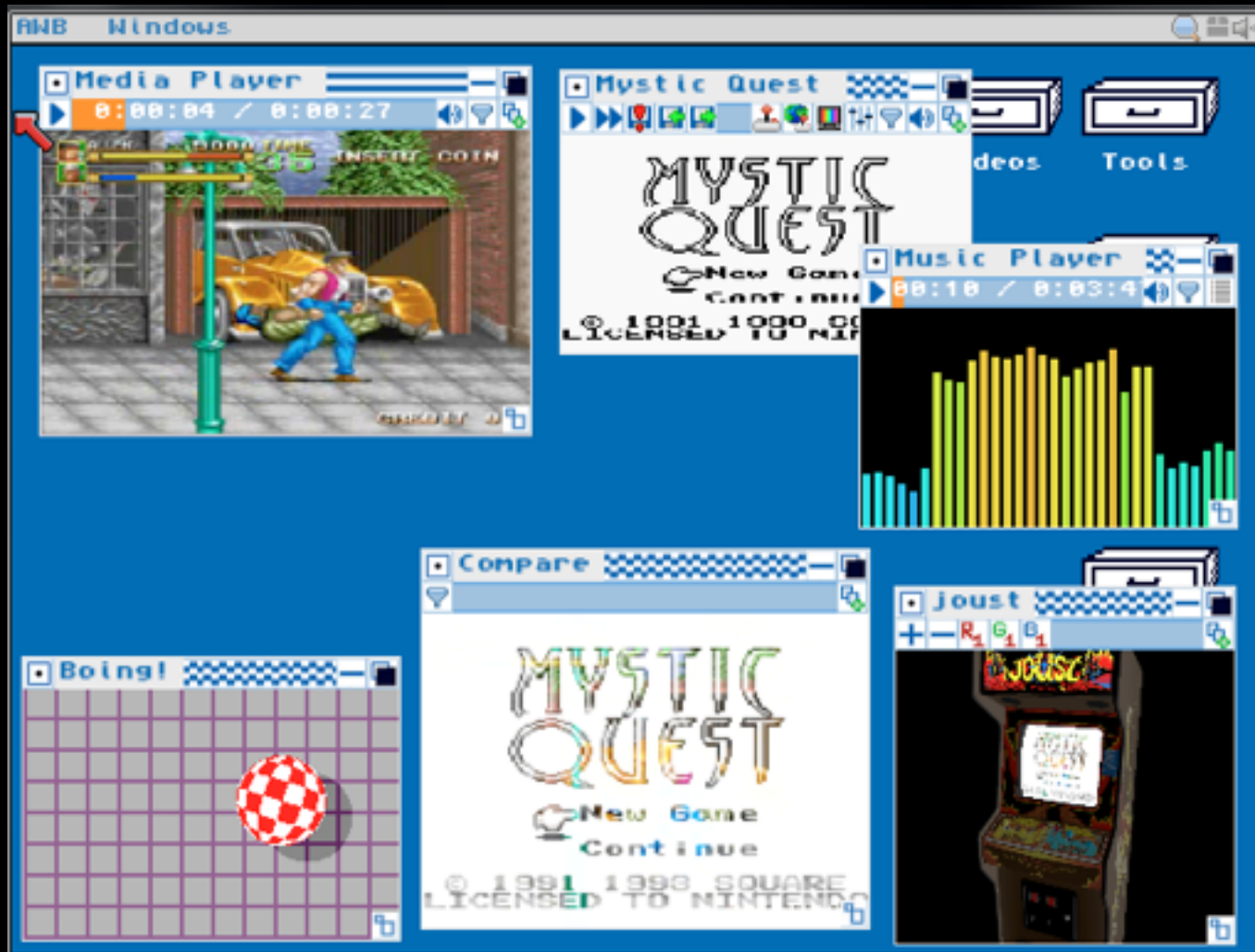
Arcan<AWB>

~2013

Inspired by some desktop from a more civilized age

Improved:

- Performance / caching for complex hierarchies
- Analog device management
- Synchronization between multiple producers/consumers
- Mouse gesture scripts
- API simplification
- A/V mixing when recording/streaming/sharing

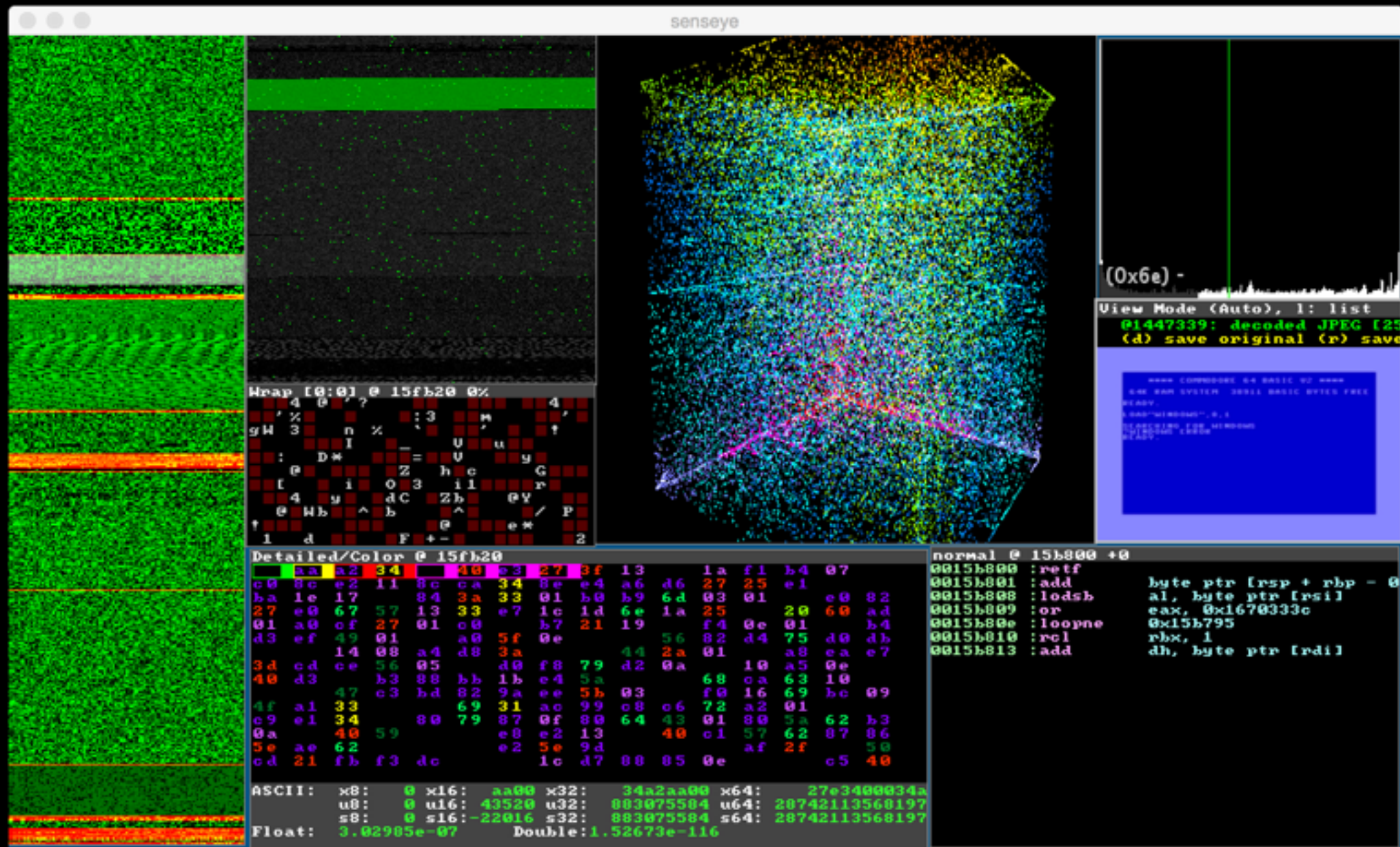


Demo Video @: <https://www.youtube.com/watch?v=3O40cPUqLbU>

Arcan<Senseye>

~2015

Intersection between rev.eng, data-vis, debugging, forensics ...



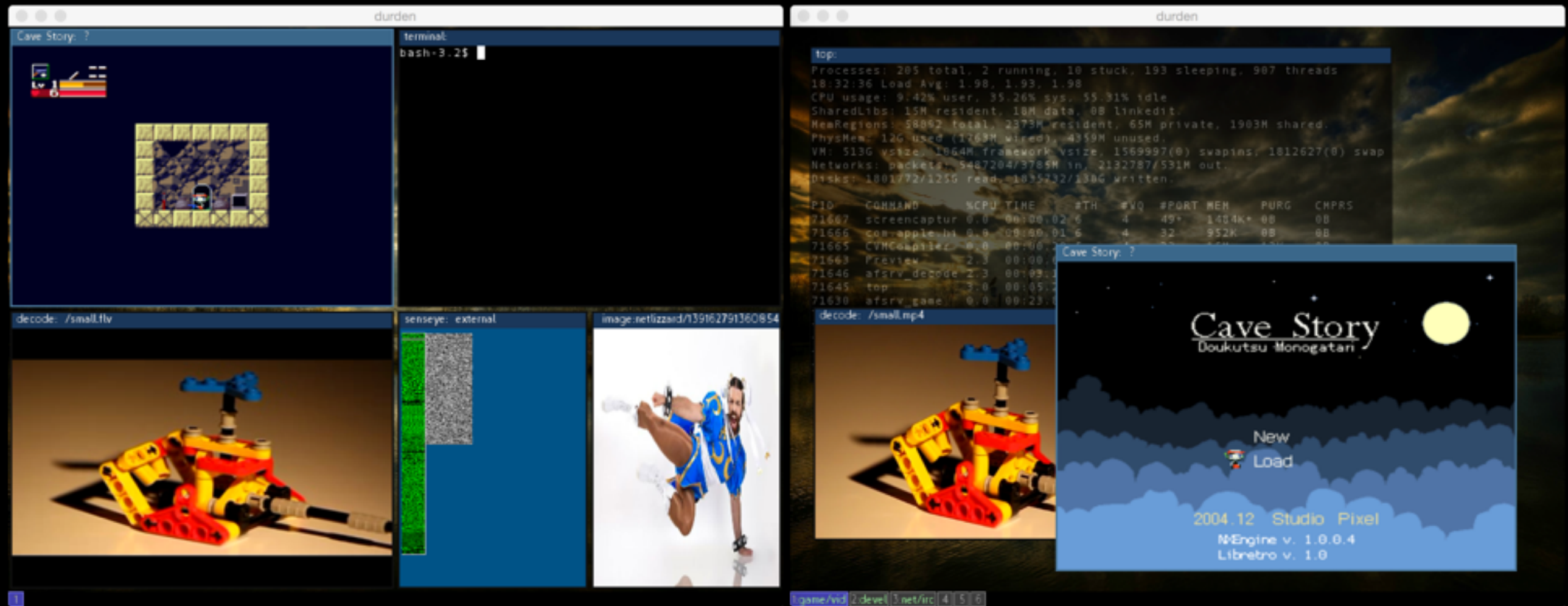
Presentation: <https://speakerdeck.com/letoram/senseye>

Details: <https://github.com/letoram/senseye/wiki>

Arcan<Durden>

~2015

(*primarily*) tiling++/keyboard driven desktop environment



Presentation: <https://speakerdeck.com/letoram/durden>

Features (rough overview)

Basic Graphics

- Rotate/Blend/Scale
- Animations
- Hierarchical Relations
- Clipping
- 3D Models & basic geometry
- Picking, Measuring
- Image Loading / Saving
- Draw Order Control
- Filtering / Blending Controls

Moderately Advanced Graphics

- Shaders + Uniform Mgmt
- Offscreen Rendering
- Streaming transfers
- Recording
- Allocation Contexts
- Custom Resampling
- Transform Scheduling

Audio

- Streaming Sources
- Sample Playback
- Gain Control
- Input Mixing

Process Control

- State transfers
- Life tracking
- Configuration
- Launching

Database

- Key / Value
- Execution Model

Display Management

- Hotplug
- Resolution Switching
- Mapping Output
- Synchronization

Device Control

- Keyboards, Gamepads, Mice, Touch
- Configurable Filtering
- LEDs

Media Control

- Video Playback
- Video Recording
- Webcams, Streams, ...

Networking (*experimental*)

- Client / Server
- Local Discovery
- Simple Messaging
- Block Transfer
- Streaming

Hopes & Ambition

or “what would this (ideally) be used for/bring”

- Key Component for “different” Desktop Environments:
 - Customizing support for Specific / Complex Disabilities
 - Loosely coupled support scripts, pick and place / share
 - Virtual Reality (useful ones, not *just* ‘lets make it 3D’)
 - Increasing public interest for graphics on (*BSDs & Linux*)
 - Enabling the Security Paranoid *e.g. alpine-linux* (good: *grsec*, *musl-libc*, *minimal*), *direct boot to signed/static arcan on ro- base-system, dev whitelist, that’s how I use it...*)
- Embedded And Specialized Graphics Applications:
 - Lightweight Computer Vision
 - UI for low-end (*raspberry pi*-level) electronics projects
 - Research Targets *e.g.* Secure UI design - data sharing in sandboxed environments, Data Visualization, Monitoring Systems, Debugging)

Status / Roadmap

(past releases, roadmap @ github.com/letoram/arcan/wiki)

(2012) emulators via “libretro” (see libretro.com)

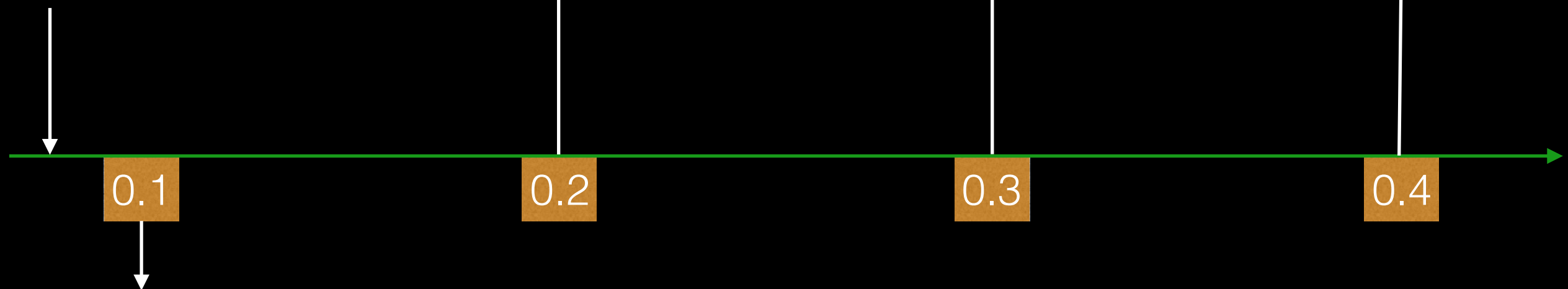
used as testing model for performance, latency, audio, I/O
video encoding (offscreen gpu + readback over IPC)

(~2003+) Private

Mostly learning experiments from
old programming experiences
(90ies emulators, software 3D,
codec development etc.)

(2013-14) Lua api improvement focus

AWB developed, tons of performance and design
quirks fixed, analog device filtering and mouse gestures
documentation and refactoring work. **Last versions that
supported Microsoft OSes...**



(2011) “Public” Release

First refactor into ‘not entirely embarrassing’ state
API feature set @ gridle level
no ‘real’ dissemination: upload to sforge
preload- hacks on SDL1.2 for games + video decode

Status / Roadmap

(current + future releases, roadmap @ github.com/letoram/arcan/wiki)

(2016 - may) current release

- * Nested (arcan_lwa connecting to arcan disp. server)
- * Remoting, (VNC client + server)
- * Heavy refactoring (db/namespace/platforms)
- * tons of doc / Q&A work
- * non-auth connections
- * much improved egl/dri/kms (multimonitor, ...)

(q2 2017)
"feature complete"

- * Audio rework / improvements
- * HMD integration /support scripts
- * 3D pipeline improvements
- * Wayland Client / Server

(q1 2018)
"secure"

- * memory allocations type-pooled
- * image/font parsers sandboxed
- * reproducible builds
- * lib-ify engine components
- * fuzzers and models for all privsep interfaces
- * critical path CFG enforcement

(q2-q3 2016)

- * Qemu (soon, Bhyve?) integration
- * Shmif- improvements (proxy, monitor, debug)
- * New backends: Vulkan

0.5

0.5.1

0.6

0.7

0.8

0.9

Durden 0.1

Senseye 0.1 - 0.3

(q4 2016)

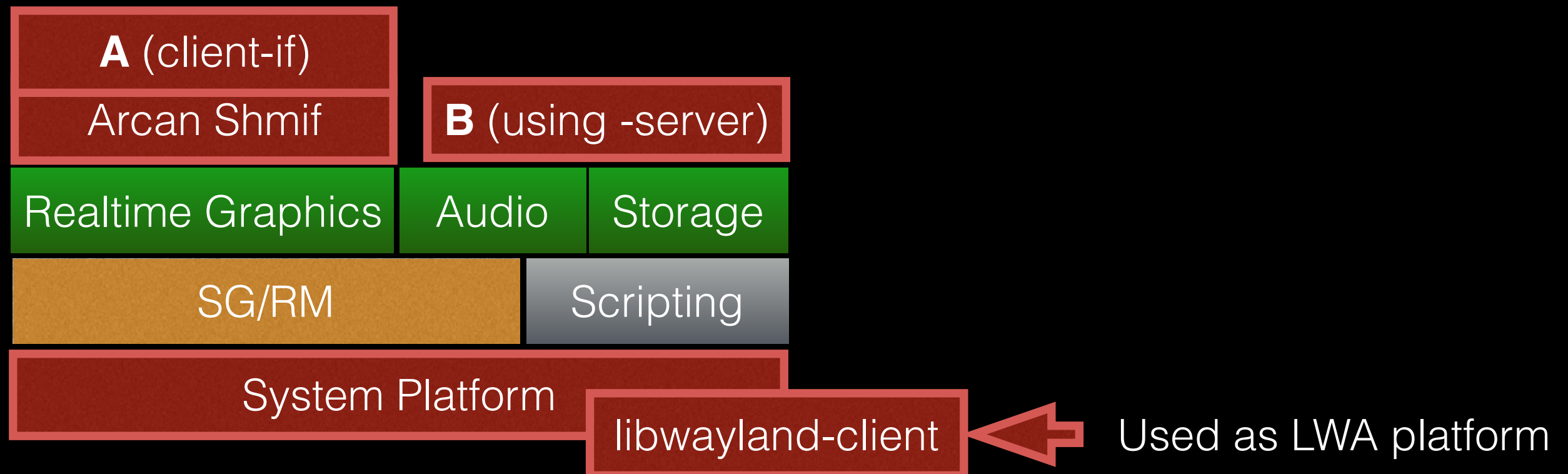
- * Finish Networking Support
- * Package Format / Loader
- * FUSE- based i/o intercept
- * Seccomp, CloudABI, Capsicum ...

(q3 2017)
"optimal"

- * all testing automated
- * profile- based optimization builds
- * heavy functions all vectorized
- * structures reordered and compacted for cache

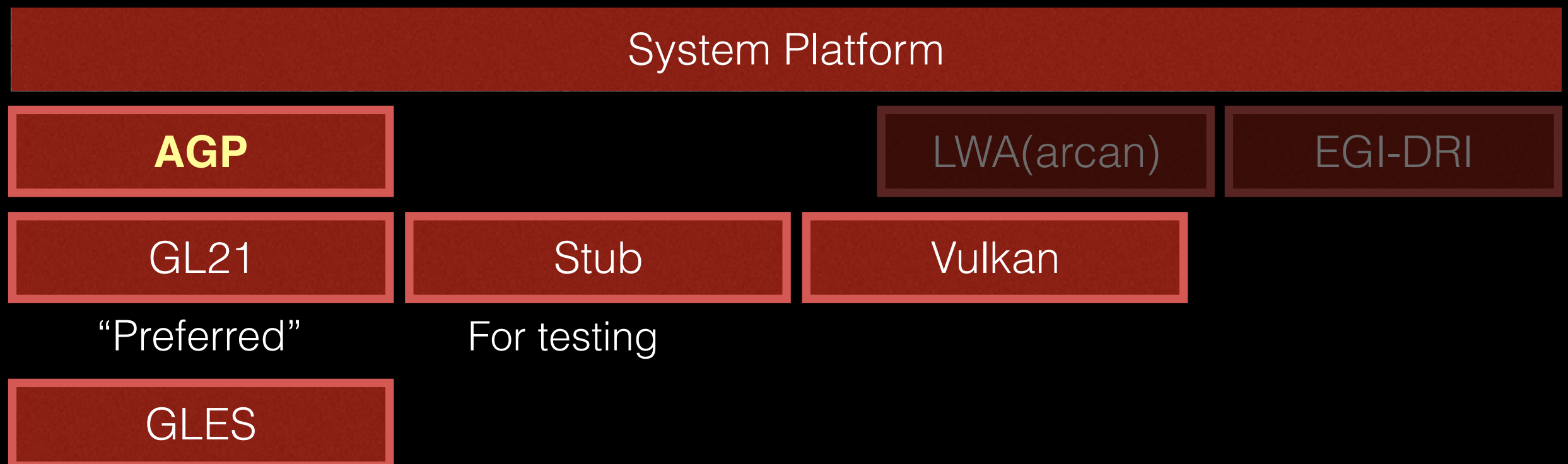
Obvious Questions #1 - Wayland

- Support **Planned**, lack of resources / time / motivation / ... / - contributors welcome :-)
 - Tight QEmu/KVM integration higher priority as means for legacy X/etc. support
- Heavy lifting (API model, input device management, **EGL/KMS/DRI**) done
- Arcan internal IPC (Shmif), feature superset - same internal code-paths can be used
 - Either by adding support for an optional libarcan_shmif build path that enabled libwayland-client needed- entry points (**A**) and have clients dynamic link to that *or* mapping engine features to libwayland-server ("proper" but interface- design mix very poorly with engine codebase)



Obvious Questions #2 - Vulkan

- Planned for arcan- side support in next release
- Used graphics operations already abstracted as part of **AGP** platform layer
 - With GL21, GLES2, GLES3 backends
- Vulkan **benefits** will **primarily** be in GPU<->CPU transfer coordination and storage management, where current GL cost is bad/broken to “insane” but still(?) missing things for ideal conditions (MAP_SHARED)



“Works” as long as decent FBO/PBO isn’t needed, full feature-set not available

Other References

Slides

Online:

Design: <https://speakerdeck.com/letoram/arcan-design>

Devel-intro: <https://speakerdeck.com/letoram/arcan-appl>

or offline in the arcan-git @:

doc/slides_devintro.pdf

doc/slides_devmodel.pdf

doc/arcan_presentation.pdf (these slides)

Much More @ Wiki

<https://github.com/letoram/arcan/wiki>