Arcan

Free (BSDv3+a little GPLv2)
portable, scriptable

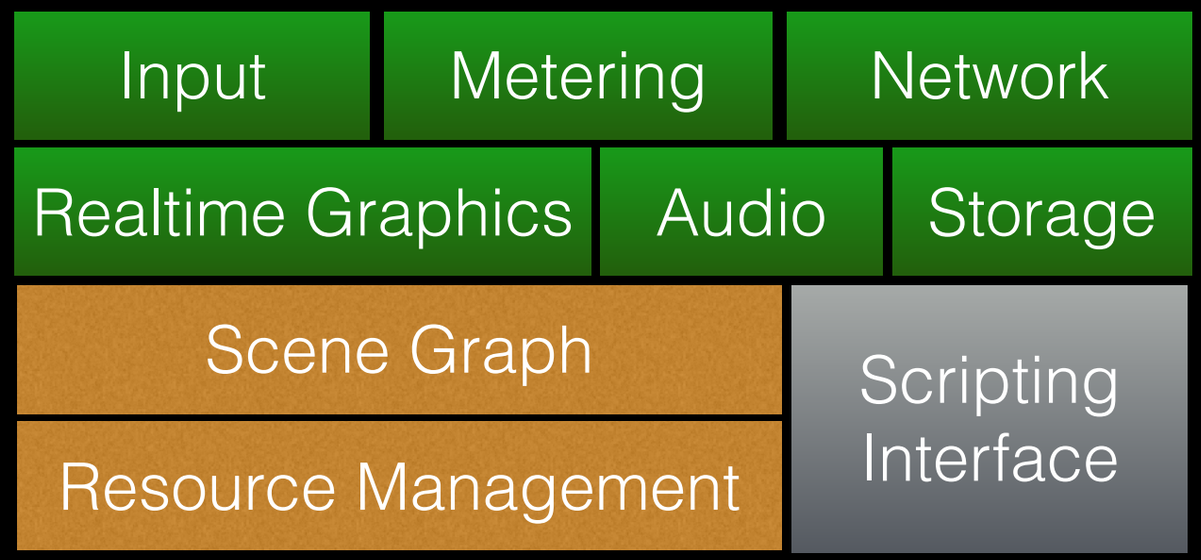
{ display server
game engine
realtime multimedia framework }

Other Sources

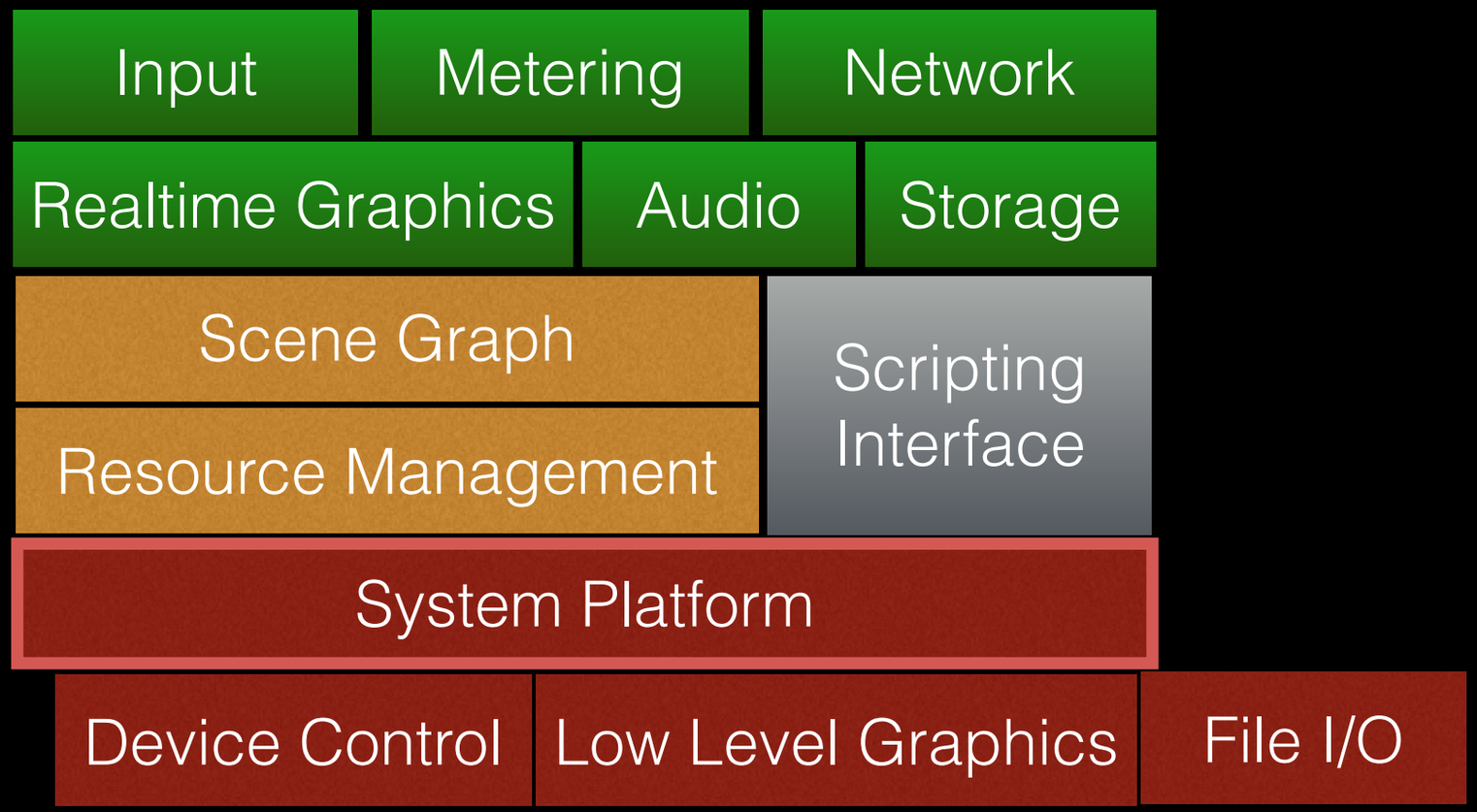
Github	github.com/letoram/arcan
Web	arcan-fe.com
IRC	#arcan @ irc.freenode.net
E-Mail	contact@arcan-fe.com

Recipe

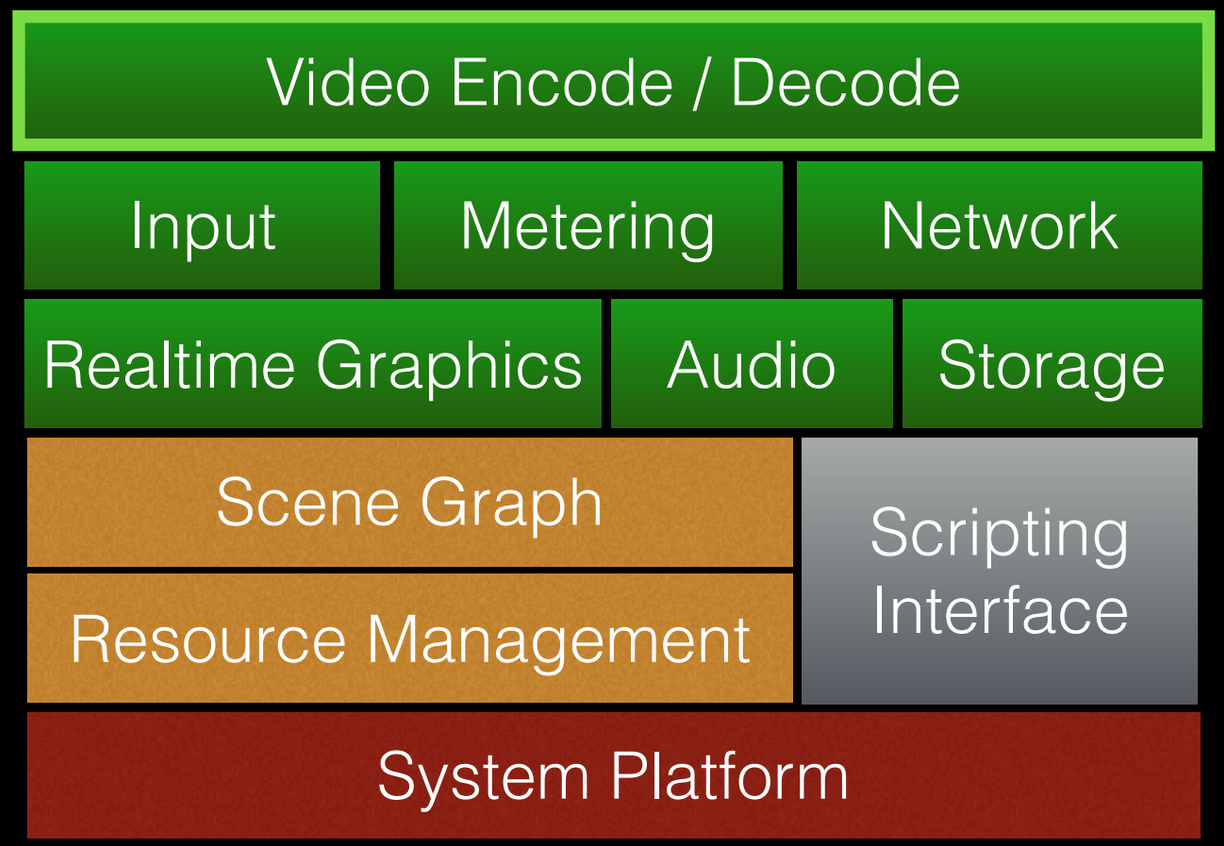
1. Take a game-engine



Recipe
2. Make it Portable

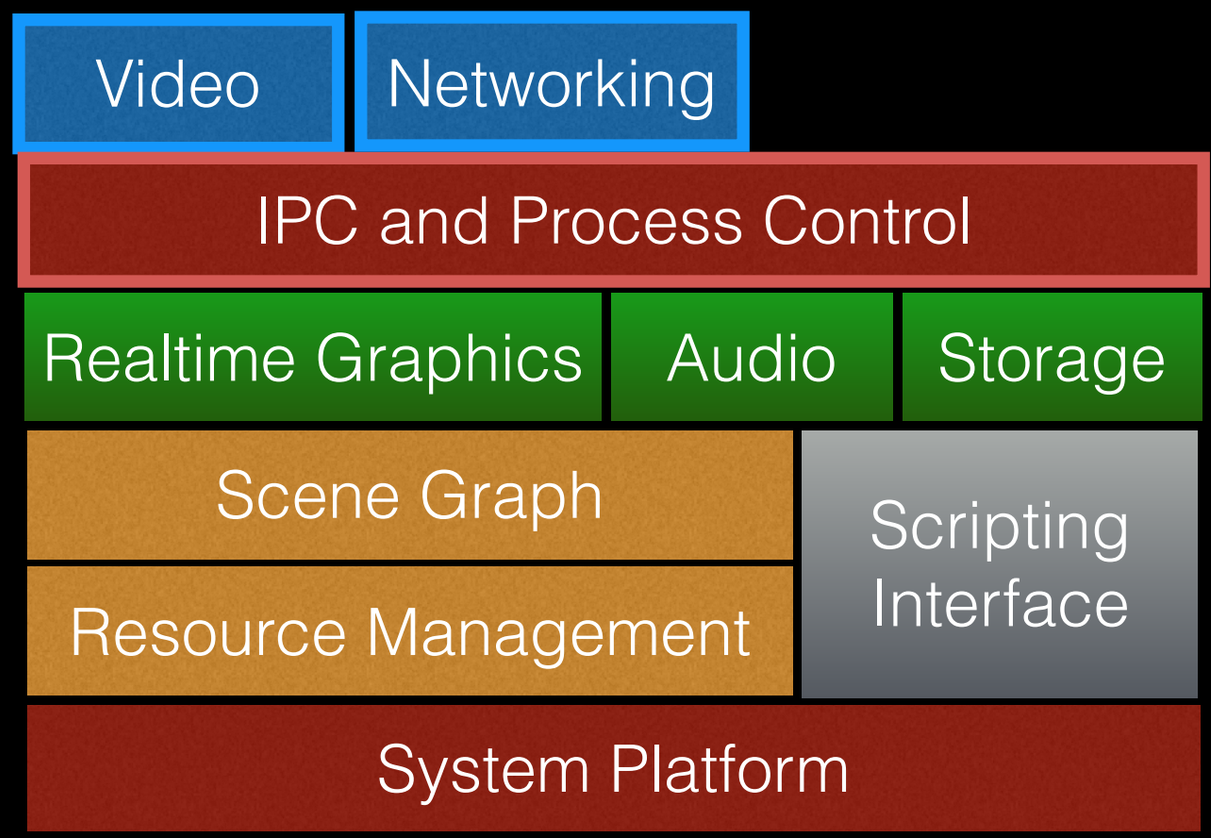


Recipe
3. Add Streaming Media Support



Recipe

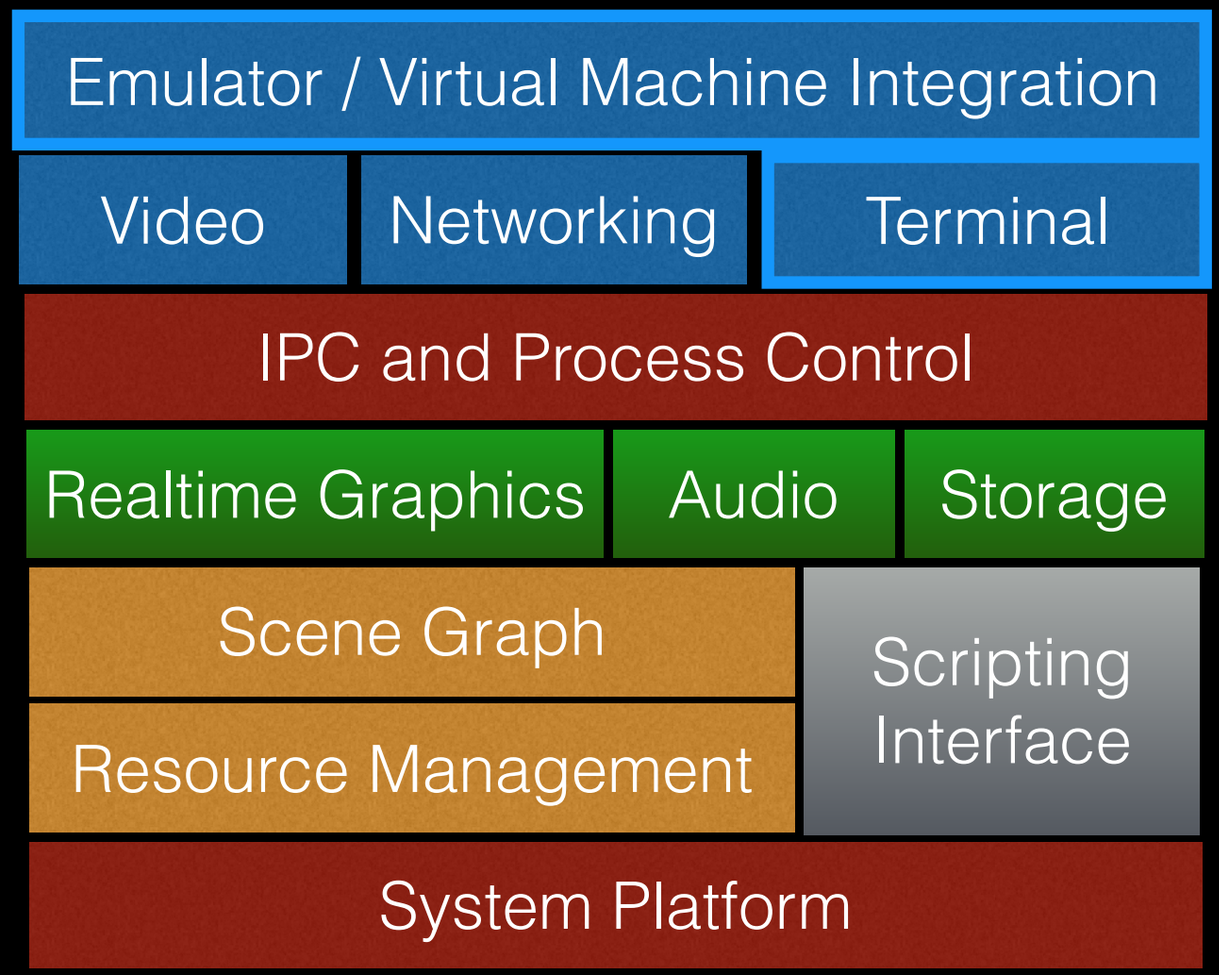
4. Add Process Separation (for resilience)



Recipe

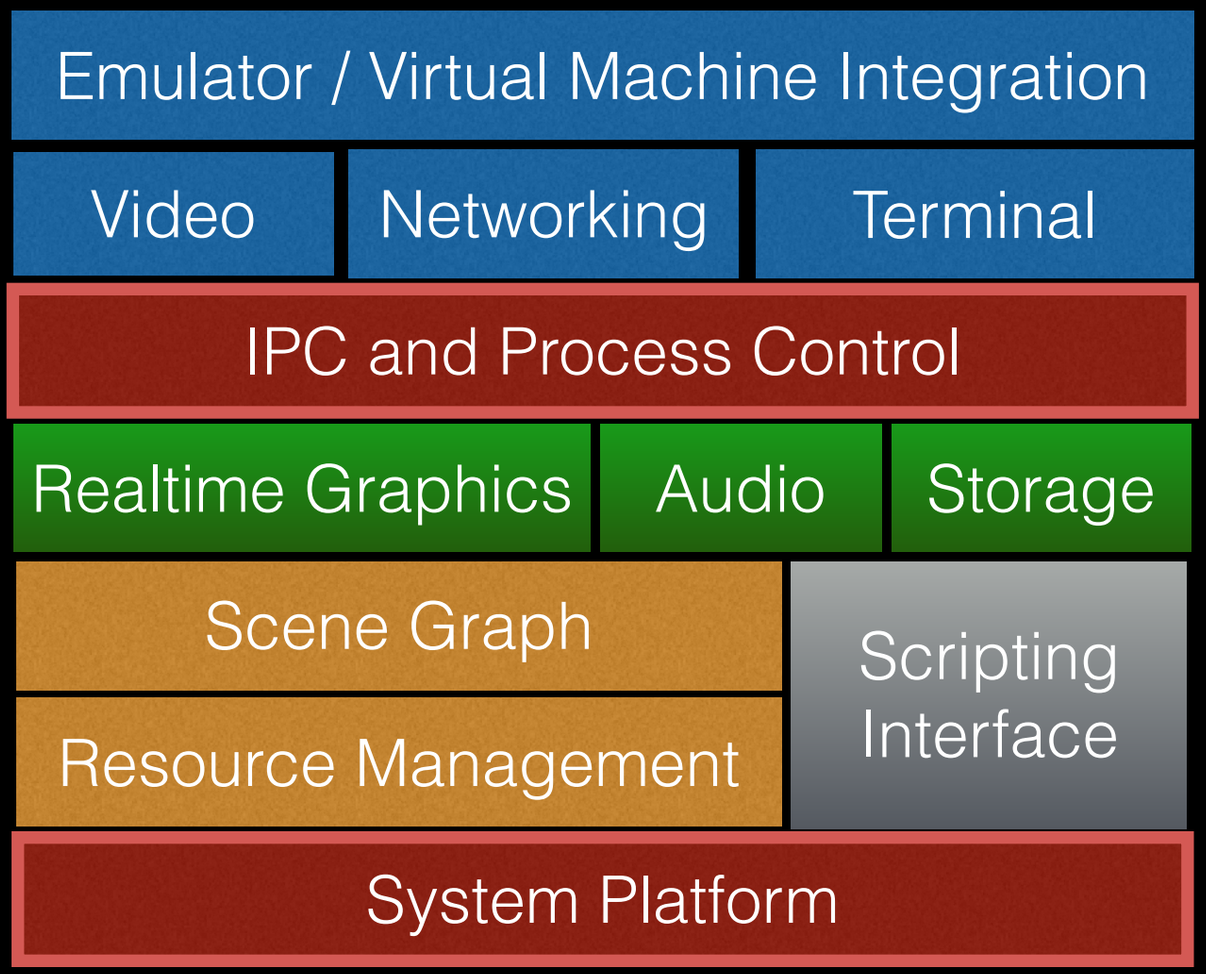
5. Expand Feature Set

[indirectly improve and harden IPC and related API]



Recipe

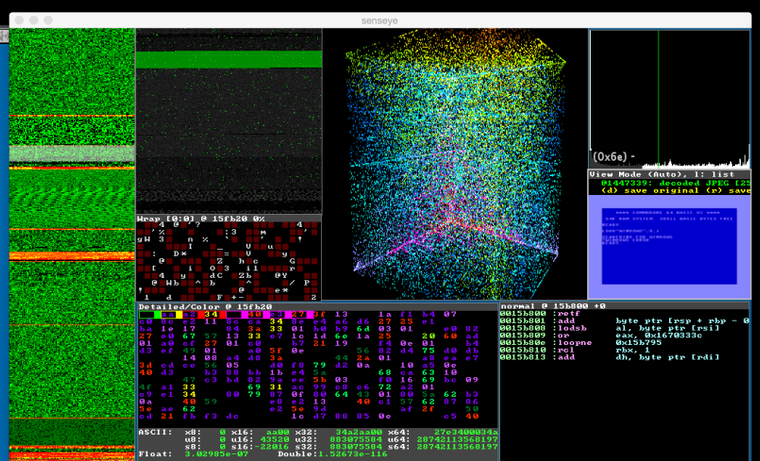
6. Display Control + External Connections

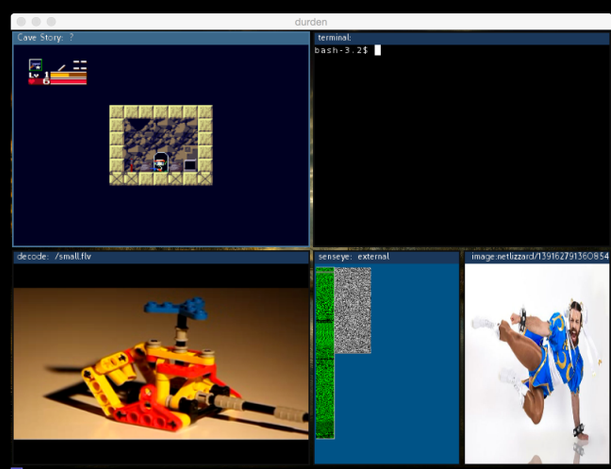


Meanwhile…

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

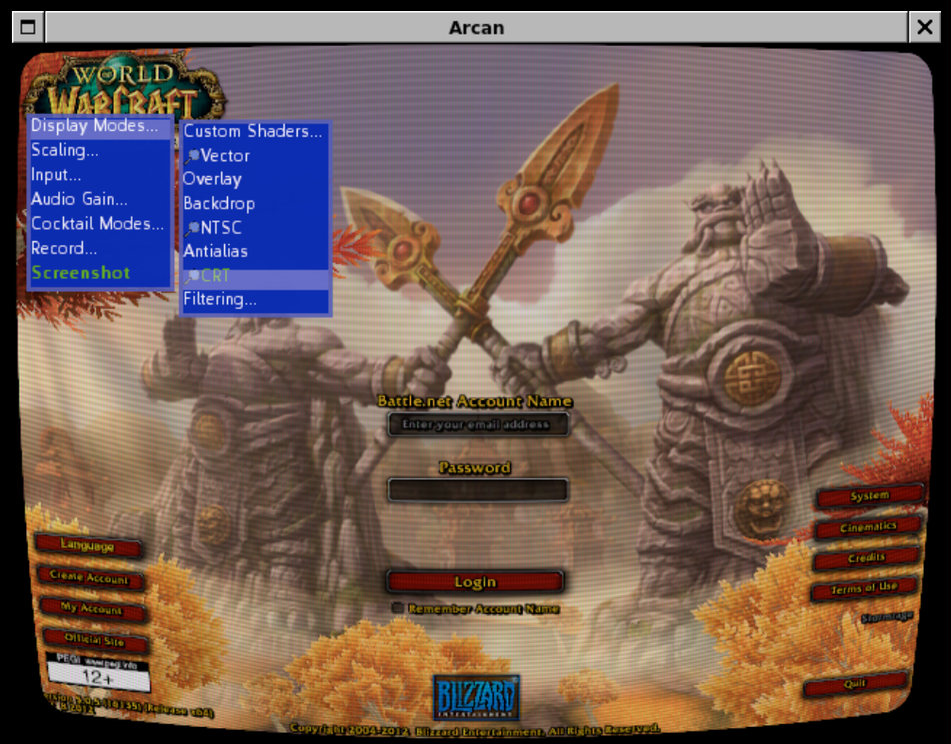
PoC Name:	Role:	Status:
Gridle	*Home-theater / Graphical FE*	Abandoned
AWB	*Classic "Fun" Desktop Interface*	
Senseye	*Debugging / Reversing tool*	Supported
Durden	*Desktop Environment*	





Arcan<Gridle>

HTPC- like interface

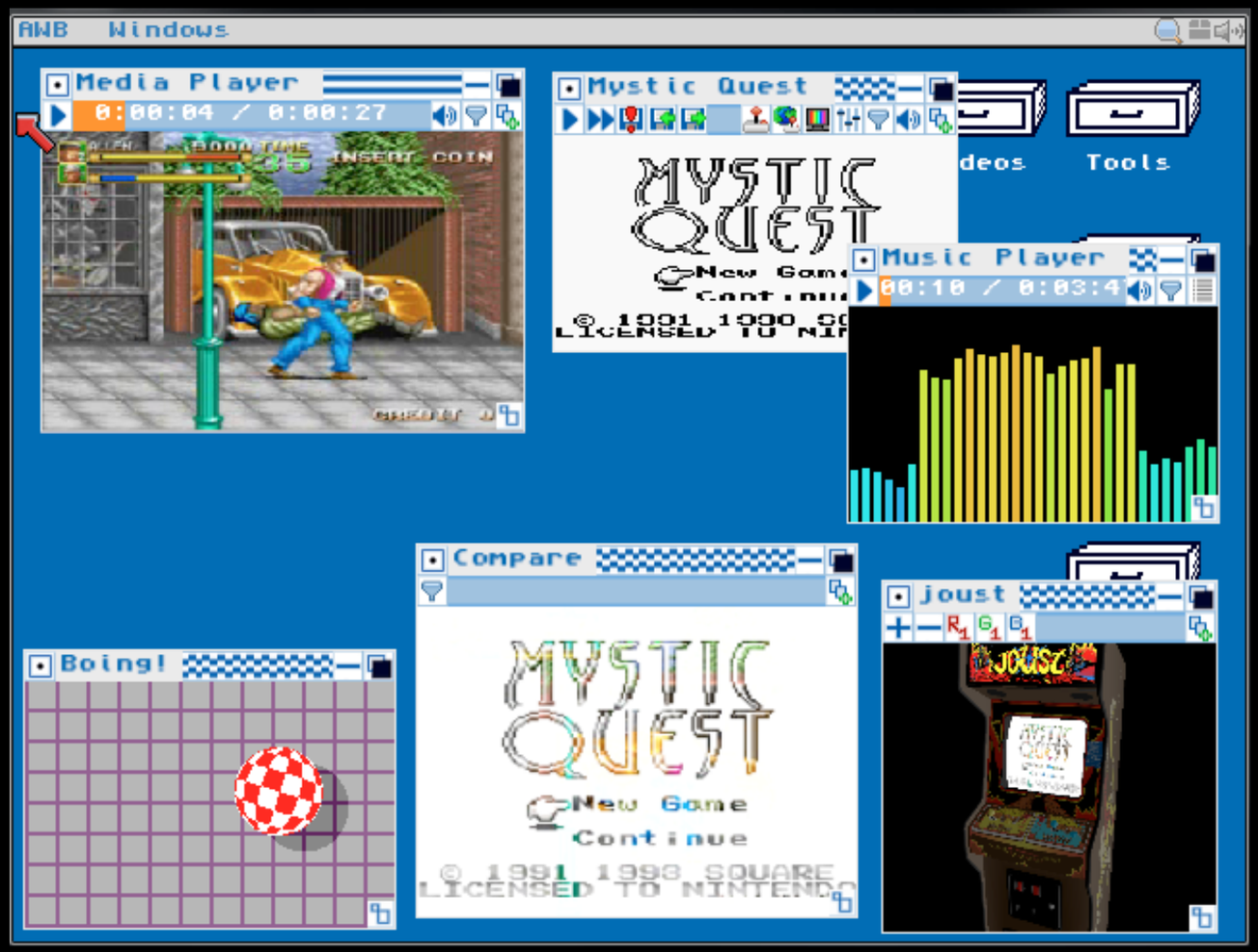


Improved:

- Input Model (support custom usb gamepads, multiple keyboards)
- Tons of asynchronous- related bugs squashed (background tiles all videos from separate processes)
- Defined much of the regular graphics APIs that was needed for the advanced effects (simulating damaged CRTs, ..)

Arcan<AWB>

Inspired by some desktop from a more civilized age



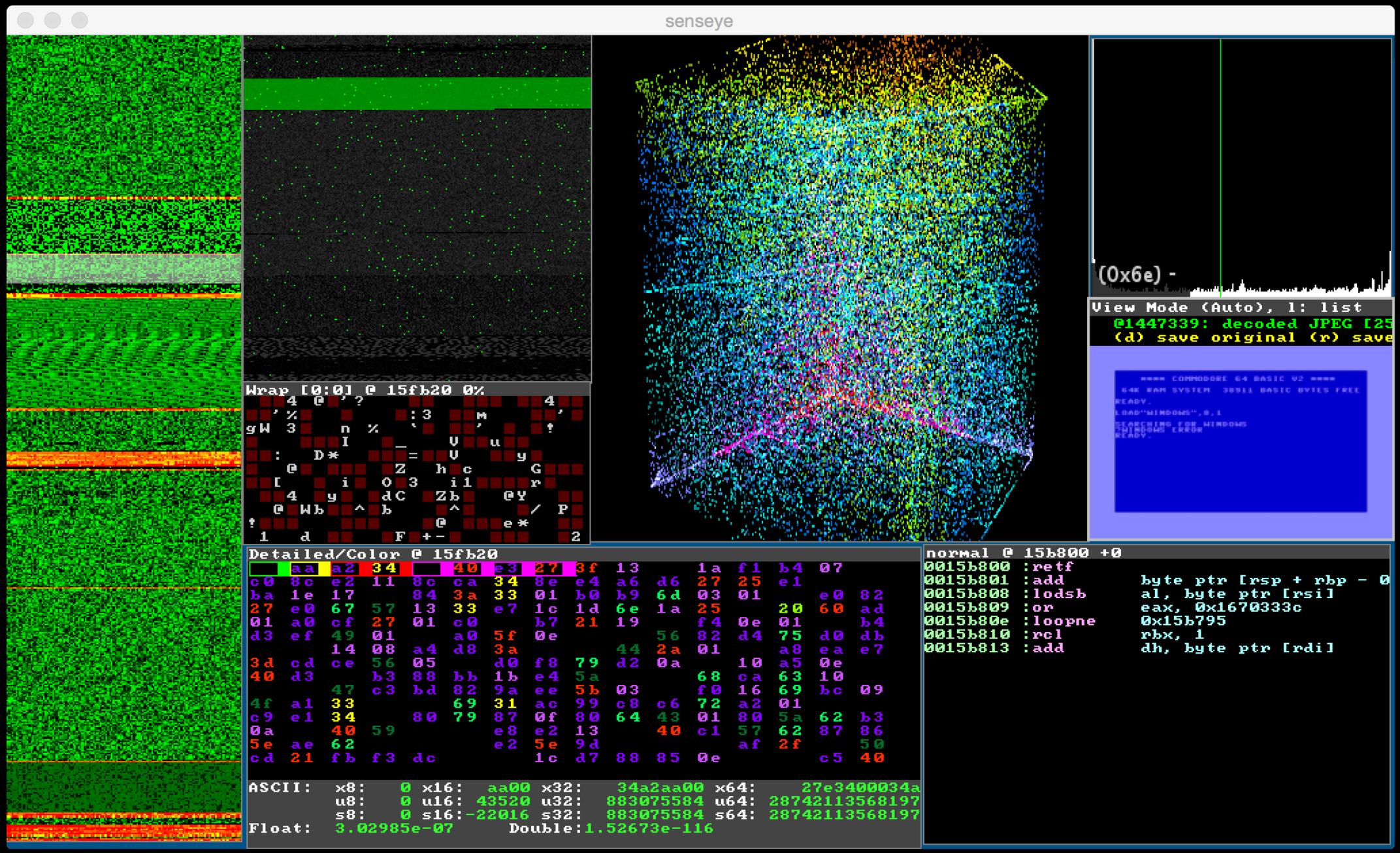
Improved:

- Performance / caching for complex hierarchies
- Analog device calibration
- 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>

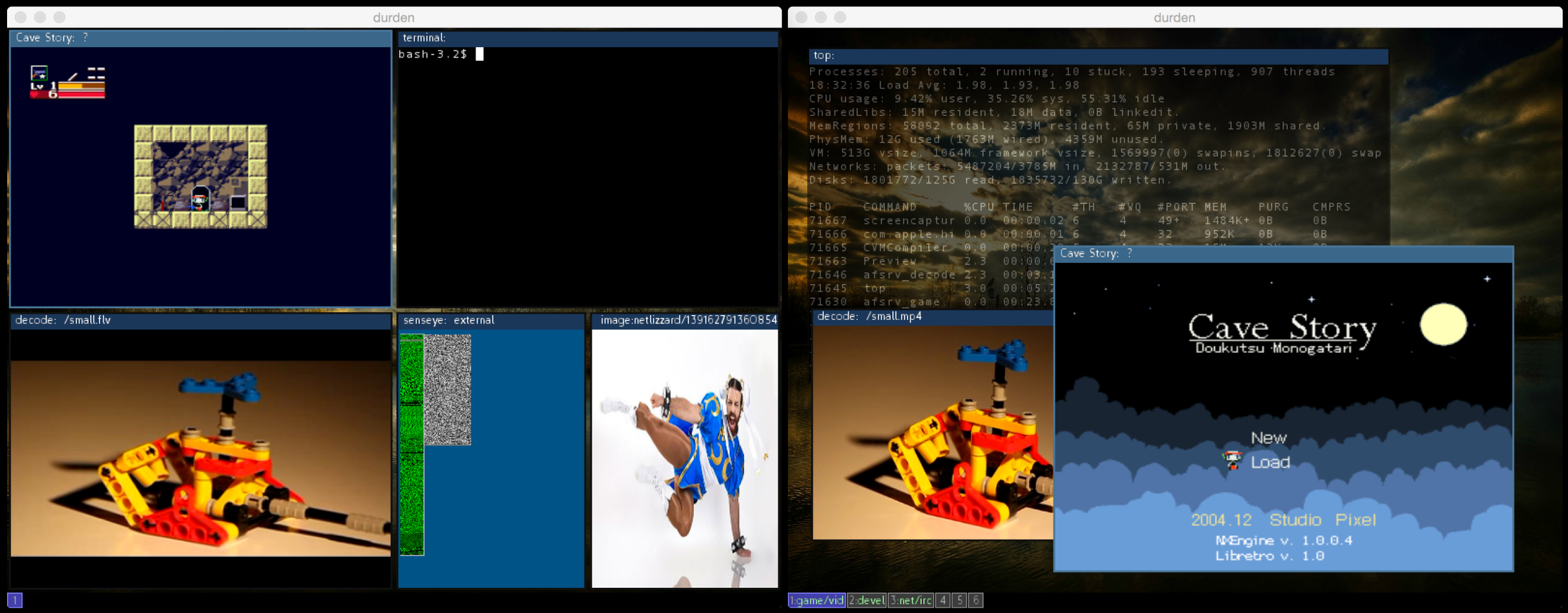
Intersection between rev.eng, data-vis, debugging, forensics …



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

Arcan<Durden>

(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

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

Audio
- Streaming Sources
- Sample Playback
- Gain
- 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:
  - **Supporting Specific / Complex Disabilities**
  - **Virtual Reality HMDs (useful ones, not just ‘lets make it 3D’)**
  - **Increasing interest for graphics on BSDs**
  - **Enabling the Security Paranoid** *e.g.* ***alpine-linux*** *(good: grsec, musl-libc, minimal, no systemd), direct boot to signed/static arcan on ro- filesystem, dev whitelist*

- Embedded And Specialized Graphics Applications:
  - **Lightweight Computer Vision**
  - **Home-Theater Projects**
  - **UI for low-end (raspberry pi-like) electronics projects**
  - **Research Targets** e.g. Securing User Interfaces, Data Visualization, Monitoring Systems, Debugging)

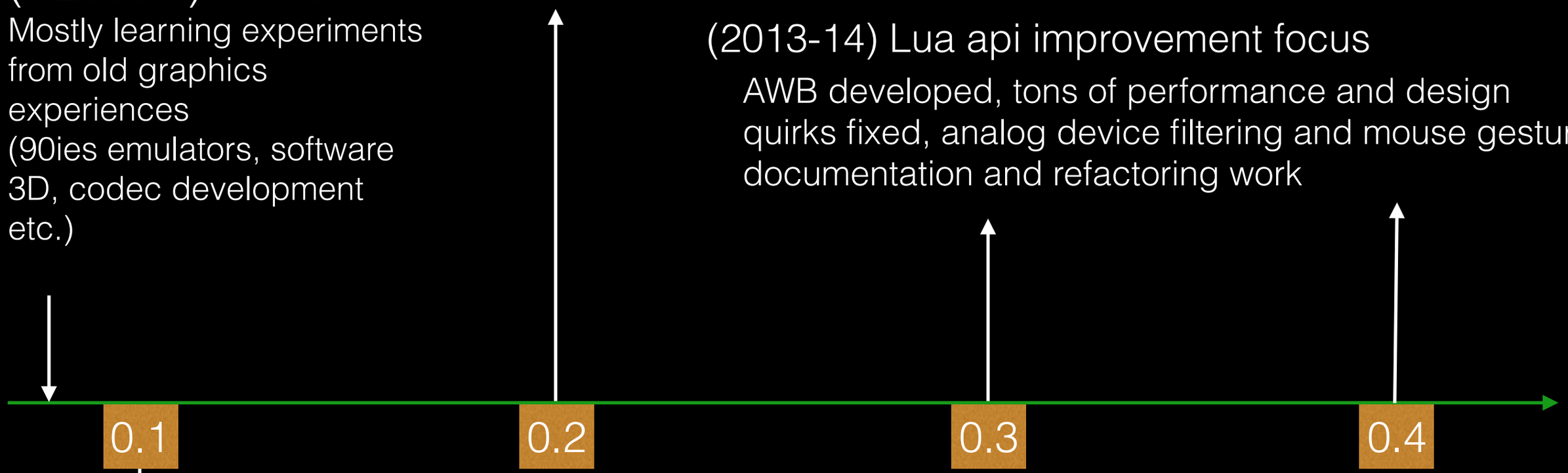
Status / Roadmap

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

(2012) emulators via “libretro”
used as testing model for performance, latency, audio, I/O
video encoding (offscreen gpu + readback over IPC)

(~2003+) Private
Mostly learning experiments
from old graphics
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



(2011) “Public” Release
First refactor into ‘not entirely embarrassing’ state
API feature set @ gridle level
i.e. upload to sforge + forum post
preload- hacks on SDL1.2 for games + video decode

Status / Roadmap

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

(2015) 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, …)

(late 2015 / early 2016)

* Pixman AGP backend
* Qemu integration (A/V/input/State)
* Text rendering improvements
* Virtual (networked, …) secondary displays

“feature complete”

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

“secure”

* memory allocations type-pooled
* image/font parsers sandboxed
* reproducible builds
* libify engine components
* fuzzers and models for all privsep interfaces

0.5 0.5.1 0.6 0.7 0.8 0.9

Durden 0.1

Senseye 0.1 - 0.3

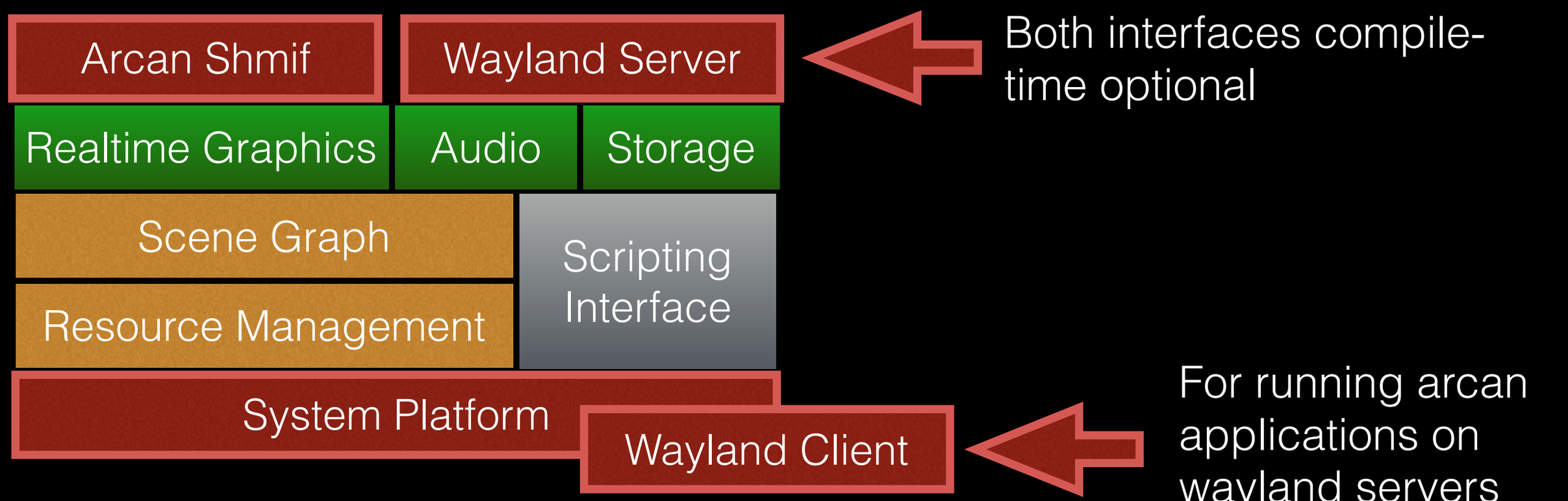
* Networking Support
* Package Format
* FUSE- based i/o intercept
* Seccmp, Capsicum …

“optimal”

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

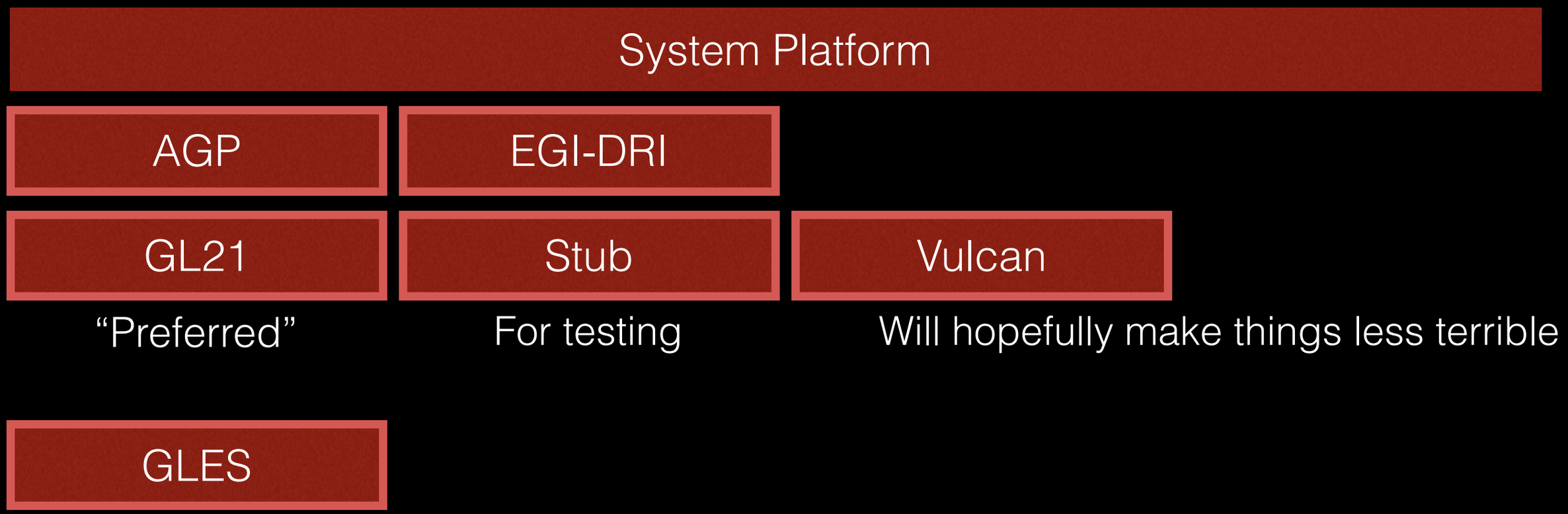
Obvious Questions #1 - Wayland

- Support Planned, but lack of resources / time - contributors welcome :-)
- Heavy lifting (API model,, input device management, EGL/KMS/DRI) already done
- Arcan IPC (Shmif) only intended for specialized projects / tight integration
- Wayland for new- linux applications and partial legacy compatibility
- Tight QEmu integration and hacker tricks (SDL1.2 preload hack for example) for specialized legacy (or 1 X server with Shmif- backend)



Obvious Questions #2 - Vulcan

- Will support when “coordinated release” !#@!#!” happens (*sigh*)
- Used graphics operations already abstracted in AGP platform layer
  - With GL21, GLES2, GLES3 backends
  - Pixman backend “planned but not there”
- Vulcan benefits will primarily be in GPU<->CPU transfer coordination and storage management, where current GL cost is bad/broken to “insane”.



Other References

Slides

Online:

https://speakerdeck.com/letoram/arcan-appl
https://speakerdeck.com/letoram/arcan-design

or offline in the arcan-git @:

doc/slides_devintro.pdf
doc/slides_devmodel.pdf
doc/arcan_presentation.pdf (these slides)