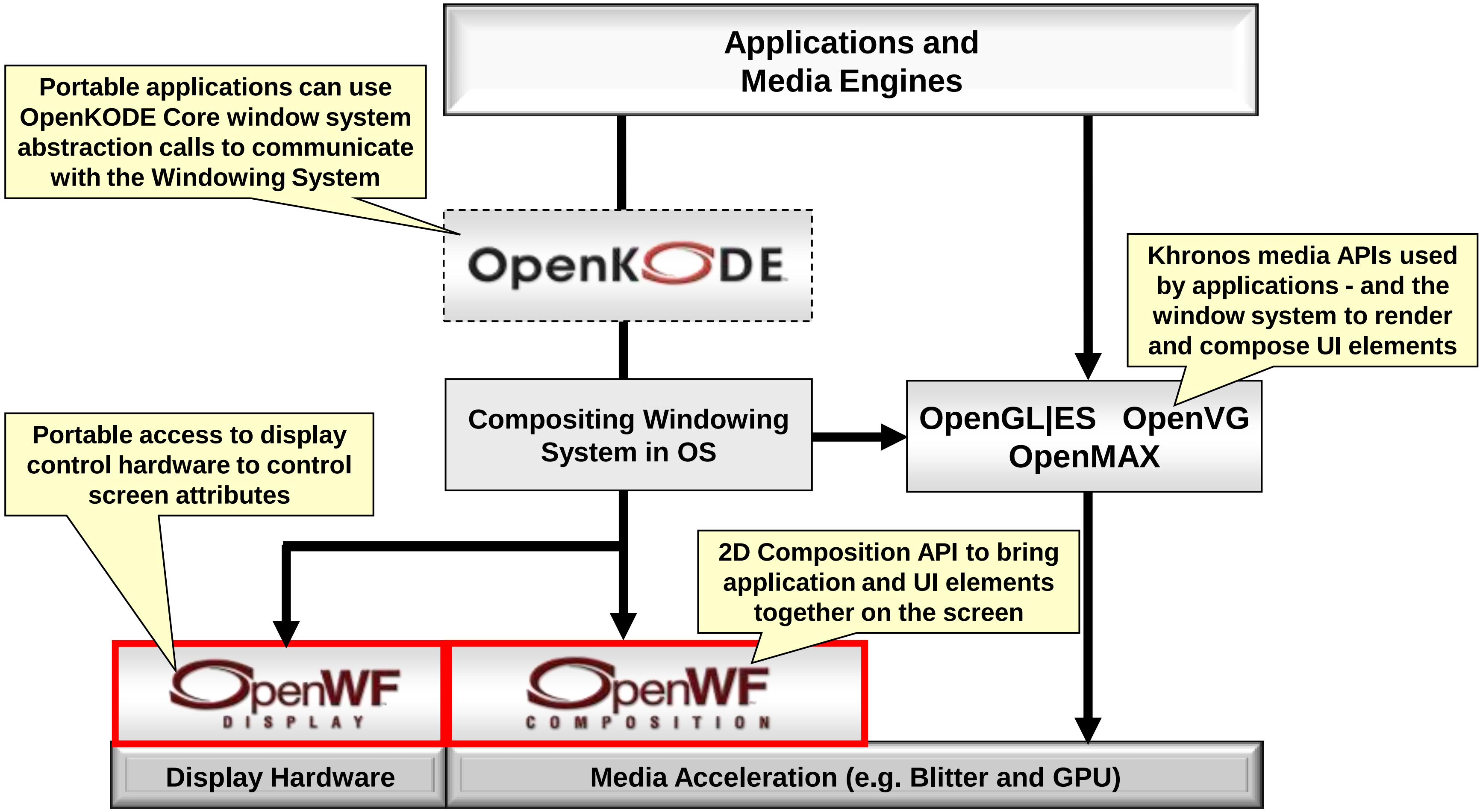
OpenWF

Overview

OpenWF in the Khronos Ecosystem

Unified access to compositing and display resources



Applications and Media Engines
Portable applications can use OpenKODE Core window system abstraction calls to communicate with the Windowing System
OpenKODE
Khronos media APIs used by applications - and the window system to render and compose UI elements
Compositing Windowing System in OS
OpenGL|ES OpenVG OpenMAX
Portable access to display control hardware to control screen attributes
2D Composition API to bring application and UI elements together on the screen
OpenWF DISPLAY
OpenWF COMPOSITION
Display Hardware
Media Acceleration (e.g. Blitter and GPU)

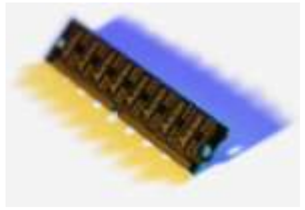
Target User of OpenWF

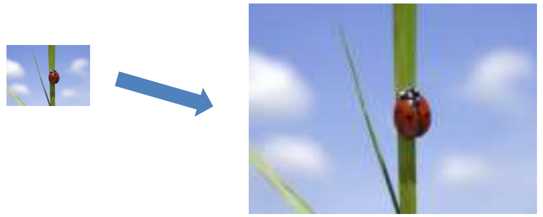
- **Compositing Window Manager (single user, probably the OS)**

More on this later...

Benefits compared to OpenGL compositing (using OpenWF hardware)

- **Memory bandwidth**
- **Power consumption**
- **Filtering/scaling quality**

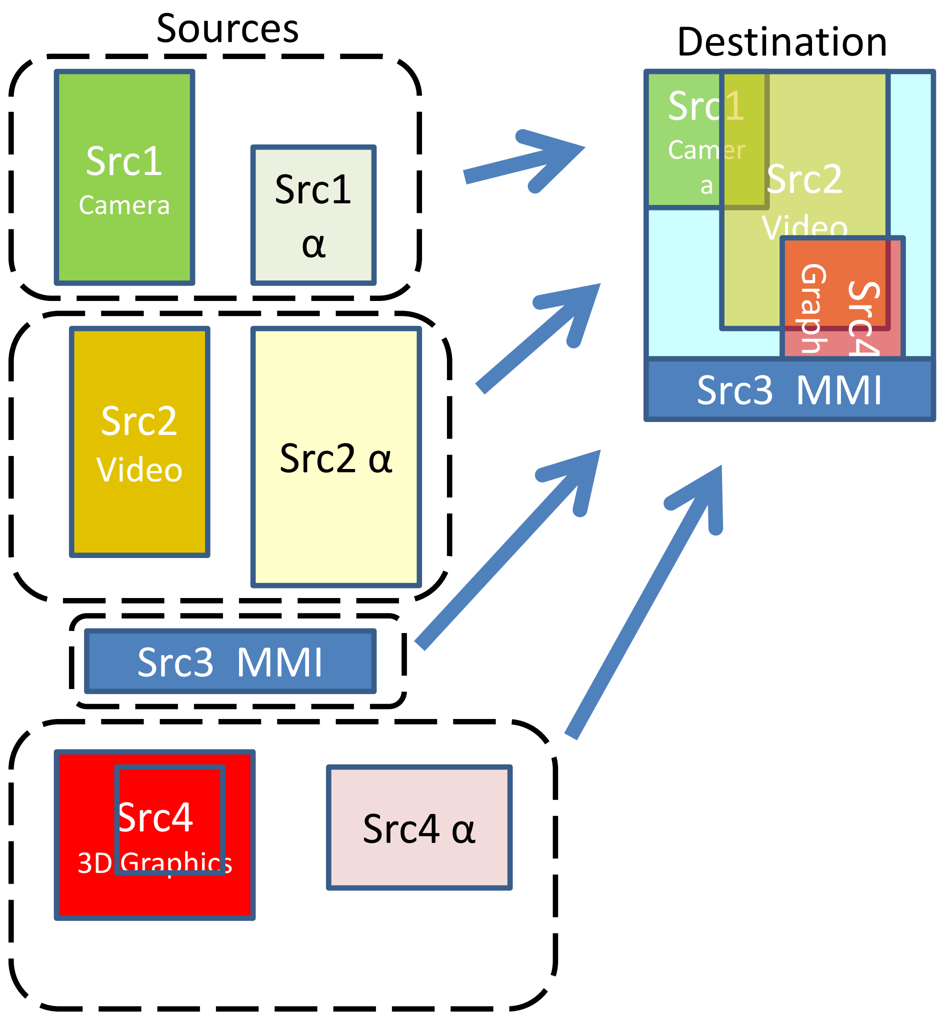




What is Compositing?

Combining several source ”images” with respect to order, transparacy (or possibly masks), scaling, rotation/flipping, cropping into one destination image.

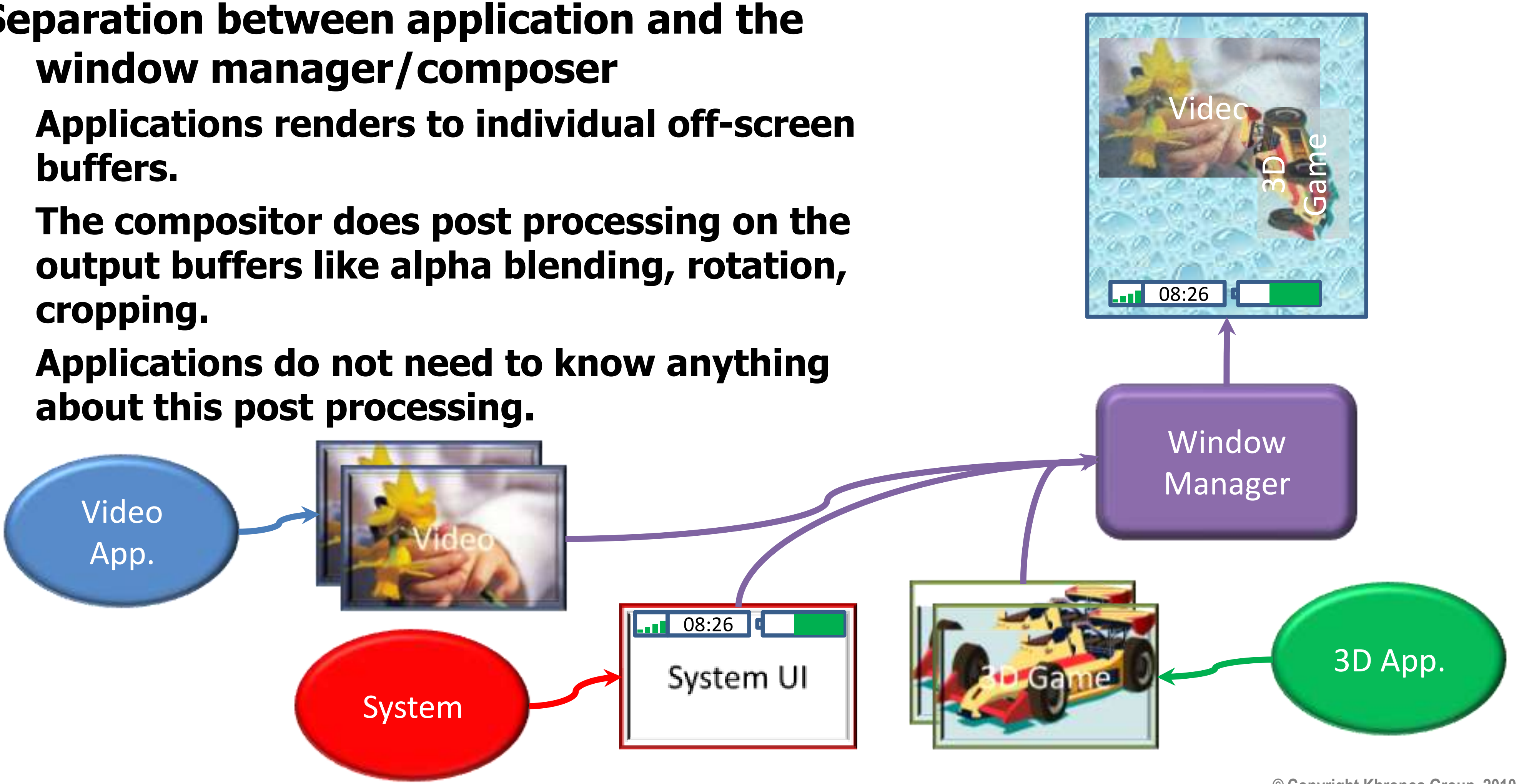
In this example all images except the MMI has transparacy information (α) in separate buffers.



What's a Compositing Window System?

Separation between application and the window manager/composer

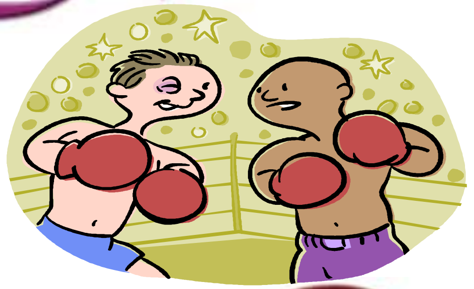
- **Applications renders to individual off-screen buffers.**
- **The compositor does post processing on the output buffers like alpha blending, rotation, cropping.**
- **Applications do not need to know anything about this post processing.**



GL vs WF compositing

- **Why not use only OpenGL for compositing?**
  - OpenWF is implementable on top of OpenGL
  - You could do effects in GL that are not possible to do in WF

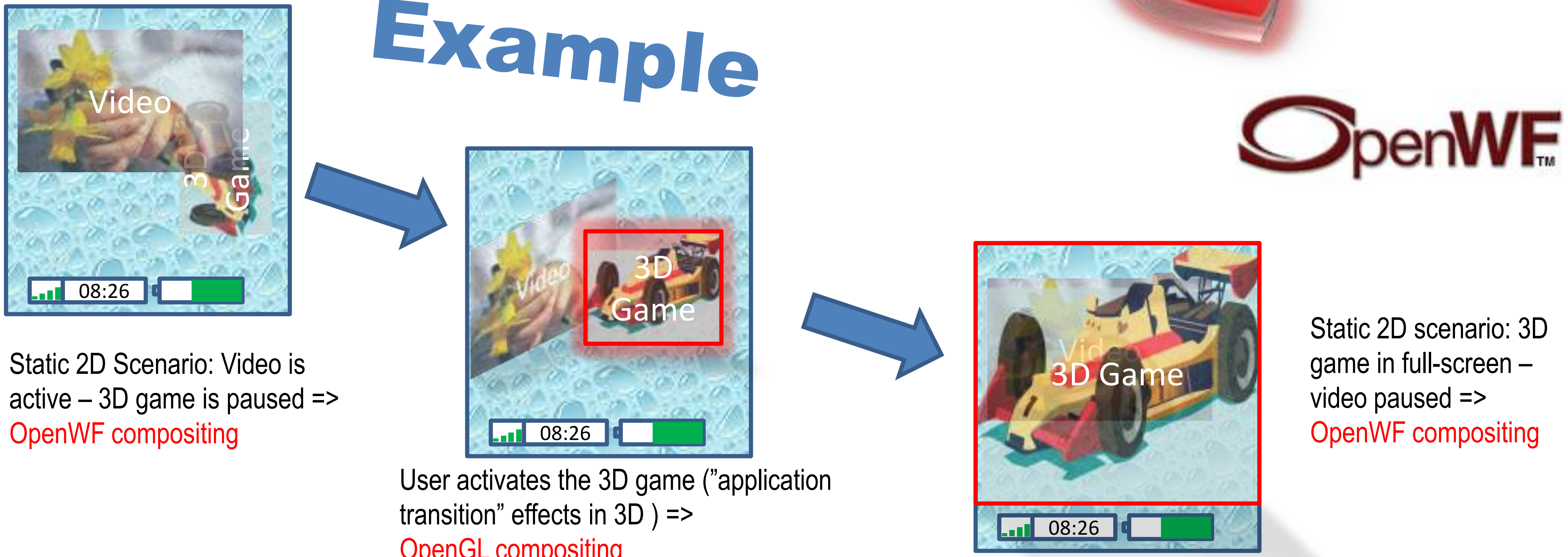
- The gross part of compositing will be for ”static scenarios” like video playback, full screen gaming etc. In this case, a specific compositing hw (using overlay compositing) could perform the work with lesser memory bandwidth and could to it in a more power efficient way
- There are higher requirements for scaling quality in OpenWF then in OpenGL – scaling images and windows is an important part of compositing (requires hw support)
- OpenWF will make it possible to utilize hardware from different venders in a standardized way



GL + WF compositing = TRUE

- **It is actully possible to use both GL and WF compositing**
  - Use GL for advanced transition effects and WF for more simple and static scenarioes

Example



Static 2D Scenario: Video is active – 3D game is paused => OpenWF compositing

User activates the 3D game (”application transition” effects in 3D) => OpenGL compositing

Static 2D scenario: 3D game in full-screen – video paused => OpenWF compositing

Why 2 APIs?

WFC

- Possible to implement on existing <u>GPU or specific hardware </u>for better power consumption
- Both "memory to memory" and "on-screen" compositing
- Unlimited number of pipelines

WFD

- Should be possible to support with <u>legacy display control hardware</u>
- <u>On-screen compositing only</u>
- Limited number of pipelines
- Optional support for rotation and scaling/filtering in the pipeline
- Standardized handling of displays, including attachable displays (like HDMI)

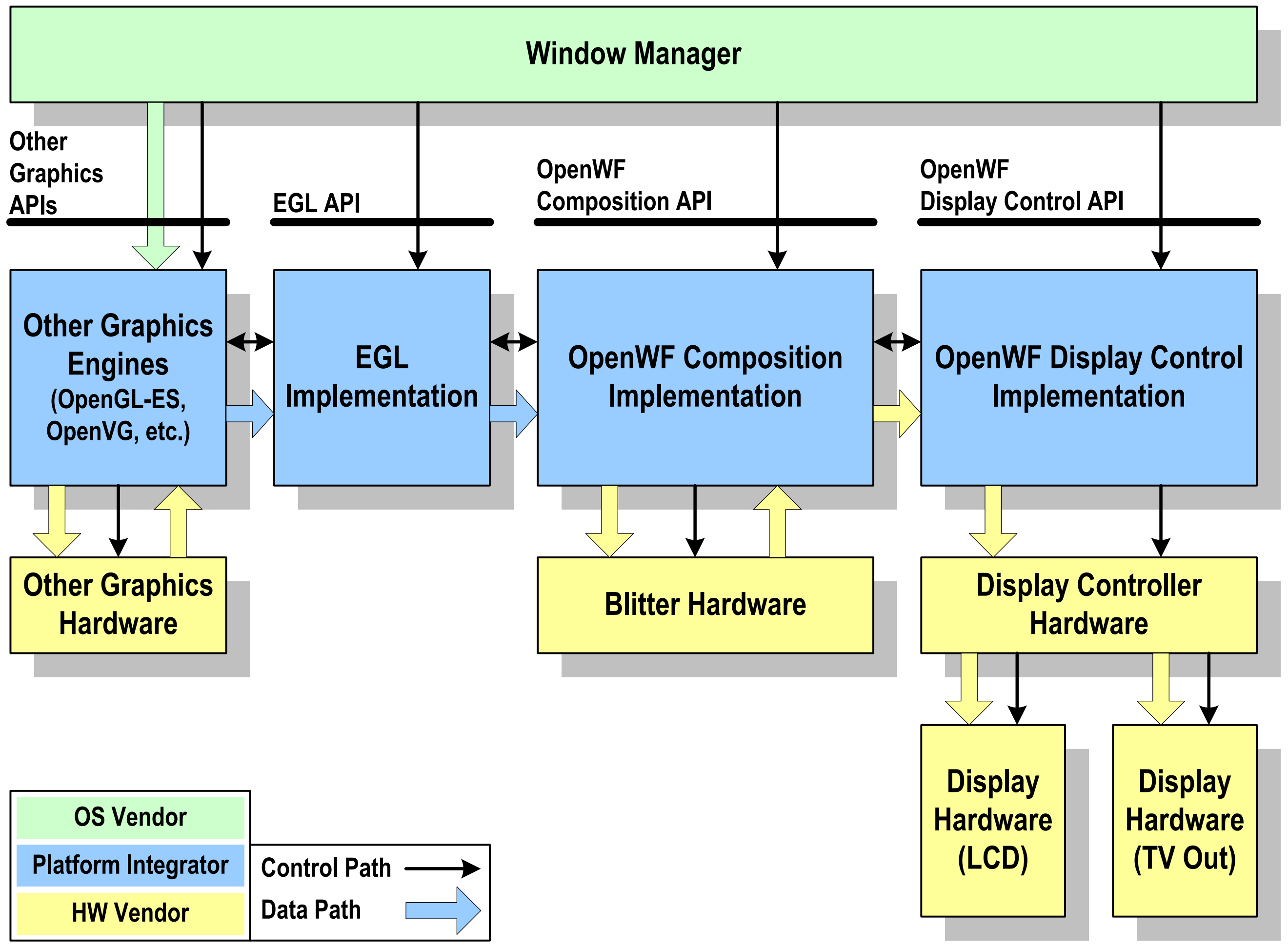
Deployment Possibilities

- OpenWF Composition and OpenWF Display are distinct interfaces reflecting the traditional separation of the hardware. The APIs can be used together in the same system or as standalone APIs.

- This independence enables system builders to take a phased approach to adopting OpenWF, for example by retaining an existing proprietary API for display control whilst migrating to OpenWF Composition for composition activities, or vice versa.

A typical system based on WFC & WFD

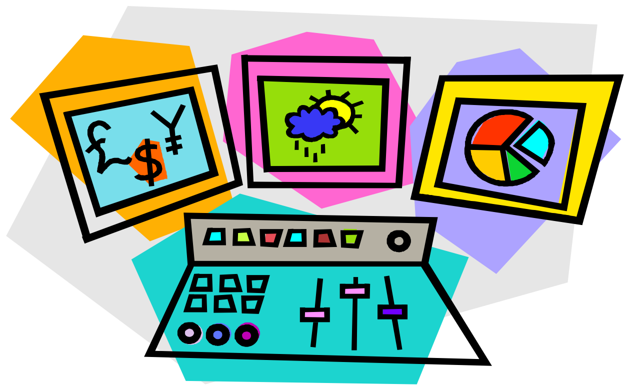
- OpenWF C - blitter hardware
- OpenWF D – a single display controller.
- Graphical and multimedia pixel routed by the windowing system to the compositor - result to the display controller.



OpenWF Compositing

OpenWF Composition

A wide range of visual scenarios can be implemented in terms of 2D layering. OpenWF Composition offers a clean API for achieving system-wide composition of layered content, such as 3D content rendered through OpenGLES hardware and video content decoded using OpenMAX-IL hardware.

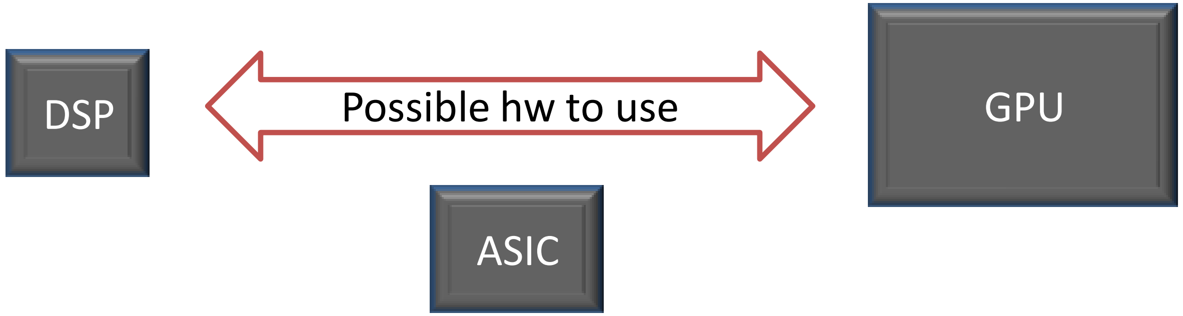


Key rendering operations are:

- Scaling (with control over filtering)
- Rotations (90-degree increments)
- Mirroring
- Alpha-blending (per-pixel and global)
- Alpha masking
- Solid background color

Scalable on different HW

The scalable nature of the Composition API allows system adaptors to use acceleration hardware from low-end DSP-based implementations all the way through to high-end Graphics Processing Units. Both render-to-memory and render-to-screen (a.k.a. overlay) hardware can be used to process the content. The technology can be used on systems with unified and non-unified graphics memory.

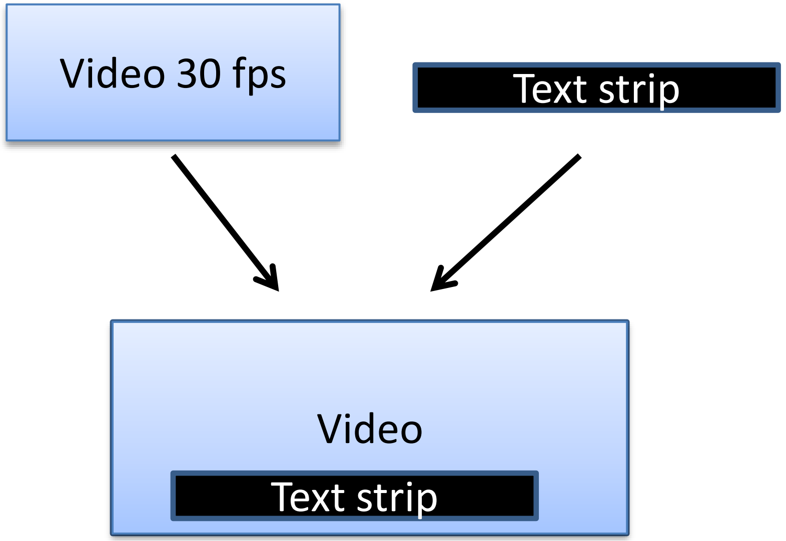


Optimize resource useage

This implementation flexibility enables the driver writer to dynamically optimize the way content gets merged onto the display according to the specific hardware platform being used. Along with performance/throughput gains, memory bandwidth usage and power consumption can be reduced. This is particularly useful for maximizing battery life during long-running use-cases such as viewing HD video with subtitles. No need for 3D HW to be engaged.

User-driven/Autonomous rendering

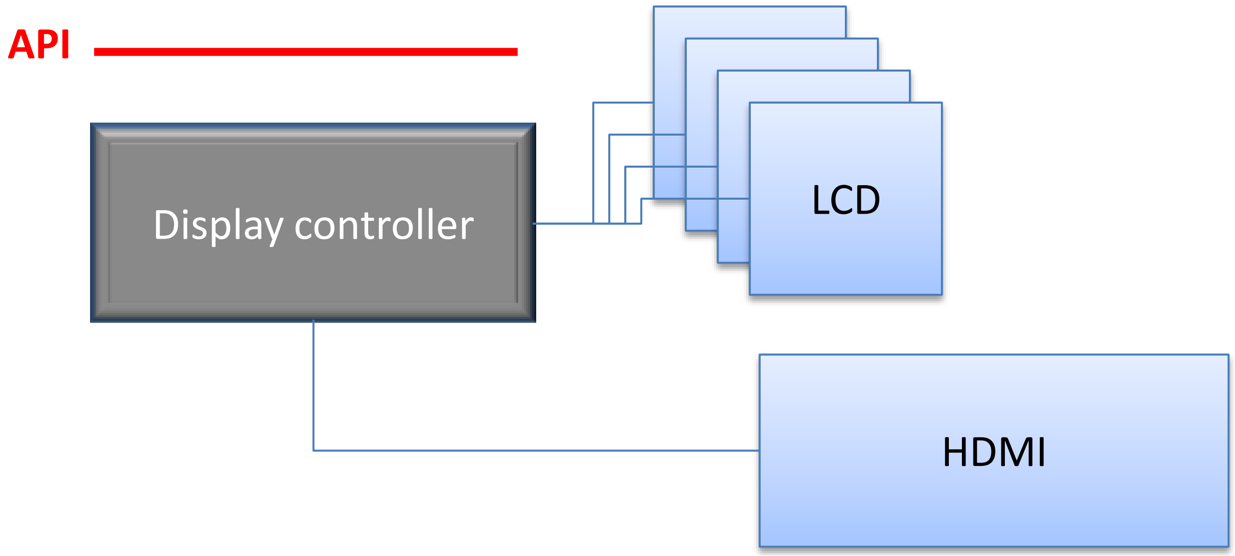
The Composition API supports both user-driven and autonomous rendering. User-driven rendering means that the windowing system decides when to recompose the scene. Autonomous rendering means that content can be rendered, composed and displayed without the windowing system being involved. On platforms with the appropriate hardware, this can be used to allow hardware video playback directly to the display without per-frame CPU intervention.



OpenWF Display

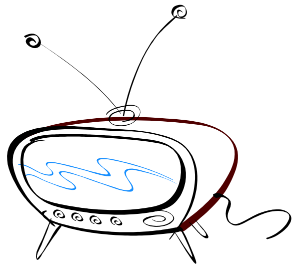
OpenWF Display

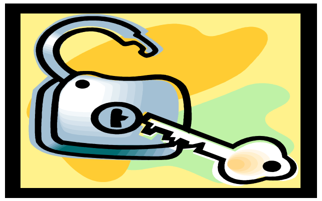
Embedded devices with multiple displays and connections for external displays are becoming increasingly common. These displays offer various degrees of configurability and commonly require custom routines for set up and mode selection. OpenWF Display provides a consistent way to query and control the state of these displays.



Key features are:

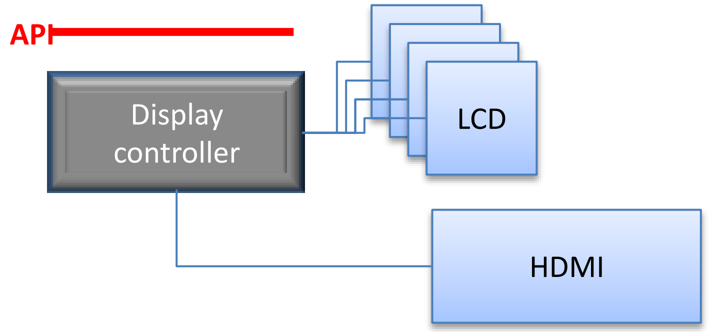
- Dynamic discovery of external displays, e.g. cable attach detection
- Power control
- Mode-setting (resolution, refresh rate)
- Rotation and flipping control
- Pipeline management (scaling, rotation, mirroring, alpha masking, alpha blending)
- Retrieval of standardized display information (EDIDv1, EDIDv2, DisplayID)
- Content protection control, e.g. HDCP





Display Management

The Display API is designed to allow the windowing system to be the focal point for display management. Support is provided for built-in displays, such as embedded LCD panels, and external displays, such as those connected by HDMI/DVI/S-Video. The cable attach sequence allows the system to tailor the display mode based on information dynamically reported from the display device that was connected. The wide range of display types is abstracted so that the windowing system can easily offer higher-level services, such as spanning of windows across displays.



Compatible with Existing HW

OpenWF Display is a low-level abstraction allowing it to be compatible with a broad range of display control hardware without the need for costly software fallbacks. The Display API reflects common hardware constraints and behaviors such as limitations on the number of display pipelines a device can offer.

OpenWF DISPLAY

OpenWF COMPOSITION

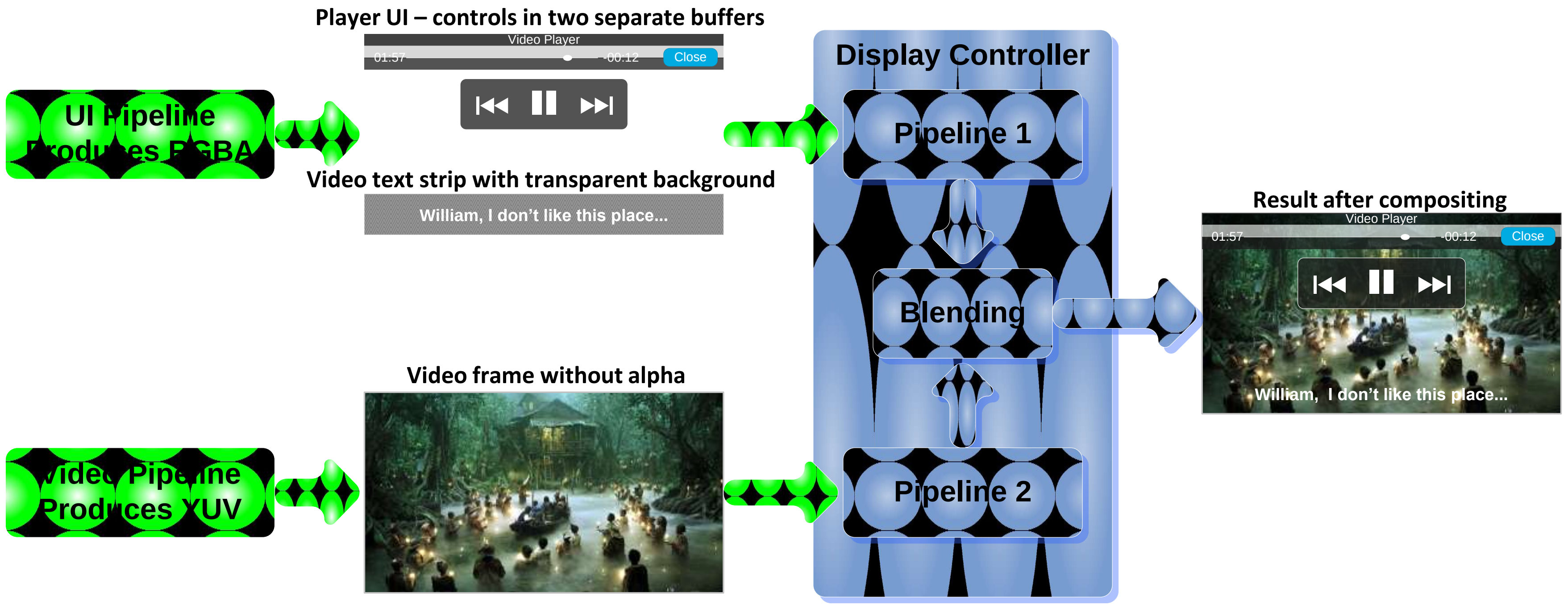
Features and Benefits

Feature	Benefit
Standardized Interface A well-defined standard abstraction for accessing composition and display control functionality.	***For implementers:*** • Reduced hardware and software design costs. • Increased marketability of features. • Maximized opportunity of hardware functionality being utilized. ***For users:*** • Decreased time to integrate new hardware. • Decreased hardware switching costs. • Portability: Minimized rework of higher level software through consistent driver interfaces.
Extensive Conformance Tests Khronos provides adopters with a rich set of tests that verify compliance with the specification and provide an assessment of visual quality.	***For implementers:*** • Reduced costs associated with developing in-house testing infrastructure. ***For users:*** • Provides a guarantee of quality. • Tests can be reused as a form of acceptance criteria when sourcing implementations.

Feature	Benefit
Conformant Sample Implementations Khronos publishes an open source Sample Implementation of both OpenWF Composition and OpenWF Display that passes the Conformance Tests and can be used to generate reference images for comparison during testing.	***For implementers:*** • Provides a concrete example to guide new implementations. • Acts as a focal point for resolving questions over intended behavior. ***For users:*** • Provides a quick way to try out and learn about the technology. • Offers a foundation to build a Windowing System in preparation for switching to a hardware-based implementation.
Optimal processing & data paths OpenWF describes the intended visual result rather than making assumptions about how the content gets onto the screen. This enables the driver writer to choose best data path and best processing hardware for the job.	• Increased battery life via the ability to use dedicated hardware with fewer gates than full GPUs. • Decreased memory bandwidth usage via the use of overlay composition. • Better performance via reduced processing overheads and ability to run composition in parallel with rendering.
Optimal control paths OpenWF supports streaming content directly to the display. For example, video data can be generated, composed and displayed without the need for per-frame intervention by the windowing system.	• Increased battery life via increased sleep time for the CPU that runs the windowing system; especially during long-running use-cases such as video playback. • Decreased latency between content rendering and display.

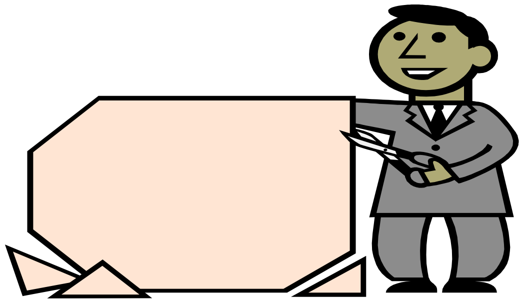
Memory Bandwidth

OpenWF Composition and OpenWF Display allow driver writers to take advantage of the full range of hardware available for composition. Choosing wisely between the different types of GPU composition and overlay composition enables memory bandwidth saving of over 69% on some use-cases.

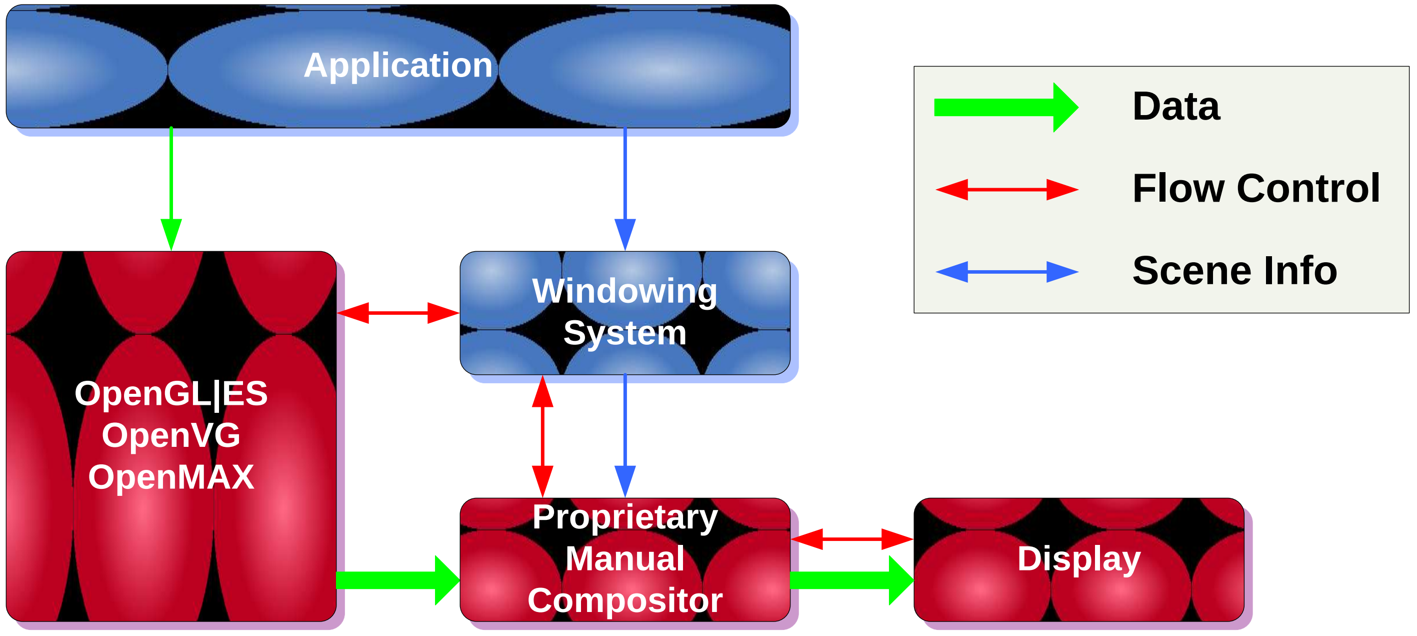

Player UI – controls in two separate buffers
Video Player
01:57
-00:12
Close
UI Pipeline Produces RGBA
Video text strip with transparent background
William, I don't like this place...
Video frame without alpha
Video Pipeline Produces YUV
Display Controller
Pipeline 1
Blending
Pipeline 2
Result after compositing
Video Player
01:57
-00:12
Close
William, I don't like this place...

Optimized Control Paths

Traditional integration of composition systems relies on the graphics/multimedia driver signaling the windowing system every time a new frame of content is ready to be composed onto the screen. The OpenWF APIs do not have this limitation and permit composition to occur without per-frame windowing system interaction whilst still keeping the windowing system in control of where content is displayed.

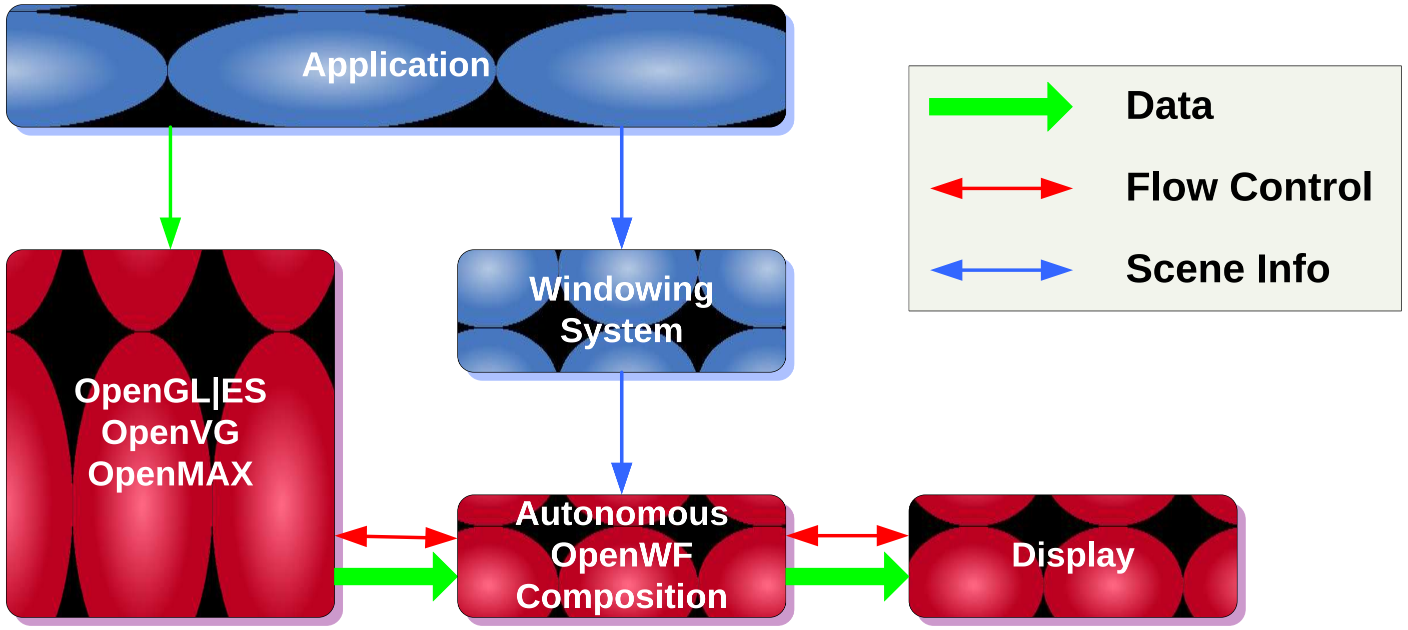


Non-optimized control-path



The traditional control-path in red. This consists of the signaling that provides notification of new content, transfers ownership of buffers throughout the chain of components and may also communicate audio-visual synchronization information.

Optimized control-path



This is the optimized control-path enabled by OpenWF. Using this model, the windowing system need only be active whilst changing the scene geometry. On systems that contain dedicated hardware for graphics and multimedia, the main benefit is that it offers more time for the main CPU to sleep during long-running use-cases with mostly static scene geometry, e.g. video playback with subtitles. Such systems can also benefit from the reduced latency that comes from new frame notifications bypassing the windowing system.

Compliance and Conformance Testing

- **Only products that have been certified by Khronos as being conformant with the specifications may use the OpenWF logos or trademarks. This means that any product displaying the logo or advertising compliance has successfully passed the extensive Conformance Test Suite (CTS) defined and provided by Khronos. The purpose of the test suite is to promote consistent, high-quality cross-vendor implementations.**

- **The test suite includes comprehensive interface testing, rendering functionality and quality tests, stress tests and animated use-case based testing. Tests can be run in automated and interactive modes, with full recording of visual results. The framework is extensible so that members may contribute further test cases and improvements over time. The test suite is OS-agnostic and is designed to be ported to multiple operating systems.**

OpenWF Summery

- Integrating new graphics and display hardware with an existing software stack, such as an OS windowing system or application-specific engine, can take considerable time and resources. OpenWF Composition and OpenWF Display reduce this integration effort, giving OEMs a wider choice of hardware and a quicker time to market for devices.

- OpenWF provides access to high-performance low-cost graphics functionality that can otherwise be left underused. Using the right hardware for the job can have a significant quantitative effect on the graphical and multimedia use-cases a device can support, as well maximizing device battery life.