



COLLABORA

Cross-Process Sharing and Direct Mode with Vulkan*

*on Linux.





COLLABORA

Jakob Borencrantz

[FDO GH @Wallbraker](#)
jakob@collabora.com

Open First



- **Why**
- **Cross Process Sharing**
- **Direct Mode**
- **Wrapping up**



- > Why <
- Cross Process Sharing
- Direct Mode
- Wrapping up

Why - Sharing?

- Security
 - Process Isolation
- API
 - Not only inter-process
 - Vulkan -> OpenGL

Why - Direct Mode?

- Speed
 - By-pass compositor
 - Almost exclusively for VR
- Vulkan only
 - Need cross API

Why - Monado!

- OpenXR
 - VR/AR API
 - Runtime
 - Open Source! :D
- Compositor
 - Cross-Process sharing
 - Direct Mode



- **Why**
- **> Cross Process Sharing <**
- **Direct Mode**
- **Wrapping up**

Sharing

- Cross Process
 - Get native handle
 - Hand wave to other process
 - Use native handle
- Cross API
 - Get native handle
 - Use native handle

Sharing

- Fence
- Memory
- Semaphore

Sharing <thing>

- Fence
 - `VK_KHR_external_fence_<platform>`
- Memory
 - `VK_KHR_external_memory_<platform>`
- Semaphore
 - `VK_KHR_external_semaphore_<platform>`

Sharing <thing> on <platform>

- Linux
 - int fd
 - int fd // (DMA-buf)
- Windows
 - HANDLE pHandle



Sharing <thing> on <platform>

- Linux
 - VK_EXT_external_<thing>_fd
 - VK_EXT_external_memory_dma_buf
- Windows
 - VK_KHR_external_<thing>_win32

Example - Image

```
VkExternalMemoryImageCreateInfoKHR external_memory_image_create_info = {  
    .sType = VK_STRUCTURE_TYPE_EXTERNAL_MEMORY_IMAGE_CREATE_INFO_KHR,  
    .handleTypes = VK_EXTERNAL_MEMORY_HANDLE_TYPE_OPAQUE_FD_BIT_KHR,  
};
```

```
VkImageCreateInfo info = {  
    .sType = VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO,  
    .pNext = &external_memory_image_create_info,  
    .sharingMode = VK_SHARING_MODE_EXCLUSIVE,  
};
```

```
ret = vkCreateImage(device, &info, NULL, &image);
```

Example - Memory

```
VkMemoryDedicatedAllocateInfoKHR dedicated_memory_info = {
    .sType = VK_STRUCTURE_TYPE_MEMORY_DEDICATED_ALLOCATE_INFO_KHR,
    .image = image,
};

VkExportMemoryAllocateInfo export_alloc_info = {
    .sType = VK_STRUCTURE_TYPE_EXPORT_MEMORY_ALLOCATE_INFO_KHR,
    .pNext = &dedicated_memory_info,
    .handleTypes = VK_EXTERNAL_MEMORY_HANDLE_TYPE_OPAQUE_FD_BIT_KHR,
};

VkMemoryAllocateInfo alloc_info = {
    .sType = VK_STRUCTURE_TYPE_MEMORY_ALLOCATE_INFO,
    .pNext = &export_alloc_info,
    .allocationSize = size,
    .memoryTypeIndex = memory_type_index,
};

ret = vkAllocateMemory(device, &alloc_info, NULL, &device_memory);
```



Example - fd get!

```
int fd;
```

```
VkMemoryGetFdInfoKHR fd_info = {  
    .sType = VK_STRUCTURE_TYPE_MEMORY_GET_FD_INFO_KHR,  
    .memory = device_memory,  
    .handleType = VK_EXTERNAL_MEMORY_HANDLE_TYPE_OPAQUE_FD_BIT_KHR,  
};
```

```
ret = vkGetMemoryFdKHR(device, &fd_info, &fd);
```


Example - Use fd!

```
VkMemoryDedicatedAllocateInfoKHR dedicated_memory_info = {
    .sType = VK_STRUCTURE_TYPE_MEMORY_DEDICATED_ALLOCATE_INFO_KHR,
    .image = image,
};

VkImportMemoryFdInfoKHR import_memory_info = {
    .sType = VK_STRUCTURE_TYPE_IMPORT_MEMORY_FD_INFO_KHR,
    .pNext = &dedicated_memory_info,
    .handleType = VK_EXTERNAL_MEMORY_HANDLE_TYPE_OPAQUE_FD_BIT_KHR,
    .fd = fd,
};

VkMemoryAllocateInfo alloc_info = {
    .sType = VK_STRUCTURE_TYPE_MEMORY_ALLOCATE_INFO,
    .pNext = &import_memory_info,
    .allocationSize = memory_requirements.size,
    .memoryTypeIndex = memory_type_index,
};

ret = vk->vkAllocateMemory(vk->device, &alloc_info, NULL, &device_memory);
```

Gotchas?

- Not available
 - Fence and semaphore
- Mesa vs NVIDIA
 - No dma-buf, just use fd variant
- Perf issues?
 - Not seen any

- **Why**
- **Cross Process Sharing**
- **> Direct Mode <**
- **Wrapping up**

Basics

- VK_KHR_display
 - Extra extensions for direct mode
- How is direct mode formed?
 - Find display
 - Acquire display
 - Set mode

Direct mode

- Window
 - No driver with VK_KHR_display
 - Windows specific API exists
- Linux
 - Mesa
 - nVidia

Direct Mode

- Extra extensions
 - VK_EXT_acquire_xlib_display
 - VK_EXT_direct_mode_display
- X11
 - Xrandr
 - Thankfully can use XCB

Common

- Get VkDisplayKHR
 - Different on FOSS and NVIDIA
- Get mode, plane and and more...
 - VkDisplayModeKHR
 - ...
- Create VkSurface
 - Continue as normal

Mesa

- Find display and mode
 - Xrandr
 - Non-desktop flag
- vkGetRandROutputDisplayEXT
 - To get VkDisplayKHR
- vkAcquireXlibDisplayEXT

NVIDIA

- Find display and mode
 - vkGetPhysicalDeviceDisplayPropertiesKHR
 - EDID name
 - Get VkDisplayKHR directly
- vkAcquireXlibDisplayEXT

Gotchas?

- Will acquire any display
 - So make sure you got the right one
- Mesa vs NVIDIA
 - NVIDIA will hide non-desktop display on X



- **Why**
- **Cross Process Sharing**
- **Direct Mode**
- **> Wrapping up <**



Open First



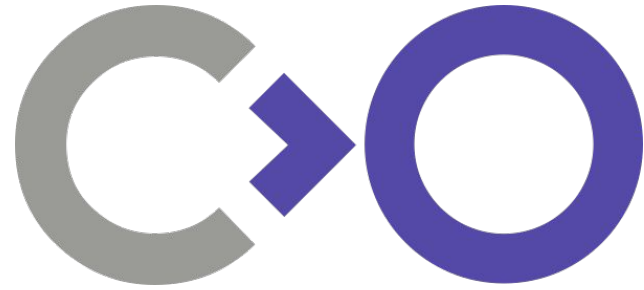
COLLABORA

Talk to me about

- Questions about this talk
- XR, VR & AR
- [Volt Programming Language](#)
- FOSS & FPGAs
- Amiga (FPGA), mc68k (LLVM)
- Voxel/SVO rendering
- FOSS & Society
- Joining Collabora!

Open First





Thank you!



COLLABORA

Open First