Understanding Composition#

There are two types of composition; composition at the hardware level using a display controller's layers and composition into a frame-buffer. Hardware level composition can composite one buffer per layer. The buffers can be either a window's buffer or a composited frame-buffer. Composited frame-buffer's on the other hand consist of multiple elements being combined into a single frame-buffer to be displayed.

A window who's buffer is displayed directly on a layer is said to be autonomous, as composition manager does not need to composite a window's buffers into a composited frame-buffer. For a window to run autonomously on a layer, the window's buffer format must be supported by the layer it's trying to be displayed on.

Hardware level composition capabilities are controlled by a layer's capabilities which varies platform to platform. There are three forms of transparency for layers that composition manager uses. These form of transparency determine how buffers on layers can be composited. The three forms of transparency are destination view port, source chroma, and source alpha blending. Destination view ports have an implicit transparency in that anything outside the view-port is transparent, allowing contents on layers below to be displayed. Source chroma allows for pixels of a particular colour to be interpreted as transparent. Unlike a destination view-port, chroma allows for transparent pixels with in the buffer. The last form of transparency, source alpha, allows for blending pixel's based on their alpha channel. Source alpha is the most powerful form of transparency, as it allows for semi transparent pixels, or completely transparent pixels like chroma.

Many of these capabilities can be achieved when composting windows into a frame-buffer. Using the same set of window properties, we're able to provide the same capabilities for windows regardless of whether they are composited or running autonomously. However when a window is composited, the contents of a window's buffer must be copied to the frame-buffer, which can negatively impact performance. But if a platform does not have enough layers to composited the number of elements desired or a layer does not support a particular behaviour, it can still be done with composition regardless of the hardware's capabilities.

Composition can also provide additional flexibility. With composition it's possible to create windows with a buffer format which is not supported by a layer, since composition can convert the format when it copies the window buffer. It's also possible to display a software cursor, or draw a background. Most importantly, it can combine multiple windows in a single buffer, which means a platform only requires one layer.

Both solutions have multiple advantages and disadvantages, some of which are very situation, sometimes depending on the rate at which window's contents are updated. In general, using hardware level composition provides the best performance. However, depending on the frequency and size of updates to windows, composition can sometime display the same as layer, but whiling using less bandwidth.

Here's a quick list of advantages and disadvantages for both hardware level composition vs composited frame-buffers.

Hardware level composition#

Advantages:#

- Performs well
- Window buffers don't need to be copied
- Does not require a lot of processing power(CPU and/or GPU)
- Deals well with a high frequency of updates

Disadvantages:#

- Can only display one buffer per layer
- Layer capabilities vary greatly, and can sometimes be lacking
- Requires a lot of memory bandwidth

Composited frame-buffer#

Advantages:#

- Can composited various buffers to be displayed with just a single layer
- Not limited by layer's capabilities
- Requires less memory bandwidth if frequency of updates are low
- Can display a software cursor or background

Disadvantages:#

- Requires copy of buffers
- Requires processing power(CPU and/or GPU) to composite buffers

Layers, WFD pipelines and EGL levels#

Now with an understanding of how composition works, it's important to know how the OpenKODE/EGL APIs map to layers. Composition manager implements a subset of the OpenWFD specification which is another Khronos Group API for communicating with display controllers. Composition manager uses WFD to do all it's interactions with display controllers. Layers are exposed as wfd-pipelines, the only difference is that layers are index starting at 0, where as wfd-pipelines begin at 1 (e.g. layer 0 is wfd-pipelines 1). Some of the WFD related functionality is configurable via configuration files such winmgr.conf or graphics.conf. Layer/Pipeline properties such as chroma and source alpha blending are examples of configurable properties. For applications, the layer which their window will be displayed on, is determined by the EGL level attribute of a windows egl config. The EGL level is an index to the layer, and it also indexed starting at 0 (e.g. layer 0 is egl level 0 which is wfd pipelines 1).

Composition Modules#

Composition Manager currently has 3 composition modules. The first, is the WFD module, which allows for hardware level composition as described in the previous section. The second, is a gf composition module which takes advantage of any GF acceleration on a platform. The final third module is a GLES2 composition module, which can take advantage of GPU's which support GLES2 to do composition.

Cursors#

With the current version of composition manager there's various ways to configure a cursor. One possible cursor is a software cursor which must be on a plane which is composited. Another possible way to get a cursor, is by placing a hardware cursor on a plane which has access to a wfd-pipeline, however keep in mind access to the pipeline cannot be shared. Lastly, is configuring WFD to use the GF hardware cursor on the platform. This is done by telling WFD to use the GF cursor as an additional pipeline. This gives composition manager access to an additional pipeline to use as a cursor.

Configuring CM#

The various winmgr config directives won't be covered here as they are all documented in the winmgr.conf file. However, this section will show various example configurations for displays in winmgr and provide some background on how they relate to the subjects discussed in the previous sections.

Configuration 1:#

```
begin display
```

```
 begin plane
  wfd-pipeline = 1
 end plane
end display
```

This configuration would gives access for applications to layer 0, as egl level 0. Any windows created on this plane (or layer or egl level) would run autonomously. However only one window can be displayed at a time. Visible windows are queued up into a fifo and only the first visible window is displayed. If the displayed window is destroyed or it's visible is toggled to invisible the next window in the fifo will be displayed if there's a window queued.

Configuration 2:#

```
begin display
 begin plane
  wfd-pipeline = 1
 end plane
 begin plane
  wfd-pipeline = 2
  source-alpha = on
 end plane
end display
```

This configuration is similar to the first configuration, however an additional plane corresponding to wfd pipeline 2 was added. Source alpha blending was also enabled on the pipeline. Adding another plane with a pipeline, allows for creating two windows on separate layers accessible via egl levels 0 and 1. Source alpha is also enabled on wfd-pipeline 2 (egl level 1), which means any window placed on layer 1 will have it's alpha channel blended. This is a basic example of composition at the hardware level. Note that to enable source alpha or source chroma on a pipeline, the layer must support those capabilities.

Configuration 3:#

```
begin display
 begin plane
  egl-config = native rgba8888
  level = 0
 end plane
end display
```

This configuration enables gf composition on the layer 0. Any windows created with EGL level 0, will be composited by the gf composition module. Since composition manager is handing composition of windows for this layer, multiple windows can be created on the same layer and can be composited together and displayed on a single layer. Windows properties such as size, global alpha, source alpha blending and visibility to name a few, can be controlled to get the desired composited effect.

Configuration 4:#

```
begin display
 begin plane
  wfd-pipeline = 1
 end plane
 begin plane
  egl-config = glesv2 rgb565
  wfd-pipeline = 2
  source-chroma = on
  cursor = sw-arrow
  background = FFFFFF
```

```
 end plane
end display
```

This configuration is similar to the previous in that we're still using composition however we're compositing with GLES2. We've also enabled GLES2 compositing, source chroma, the background and a software cursor all on the second layer. Note that software cursor's can only be placed on composited planes. We also added a another plane corresponding to layer 0. This allows for having a GLES2 composited frame-buffer be composited at the hardware level with the other autonomous window below. Since source chroma is enabled, the background colour will be used as the transparent colour key. Any pixels which match the background colour will be transparent.