

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light green. They are positioned diagonally, with the blue one partially covering the green one.

Foveated Rendering in Turnip

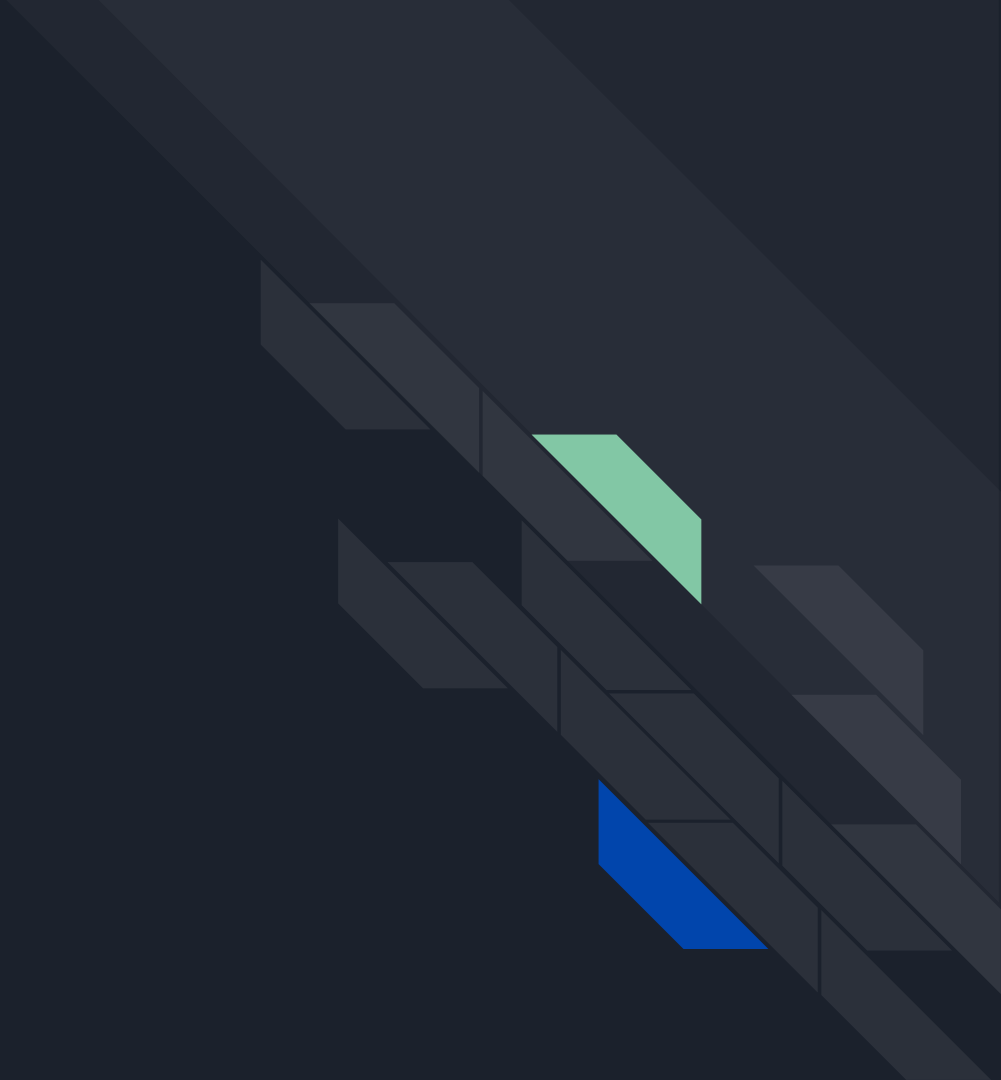
Connor Abbott, Valve



What is Turnip

- The open-source Vulkan driver for Qualcomm Adreno GPUs in Mesa
- Under the freedreno project
- I joined the project in 2020
- Now a mature, stable driver

Meanwhile...







Gimme More FPS

The goal:

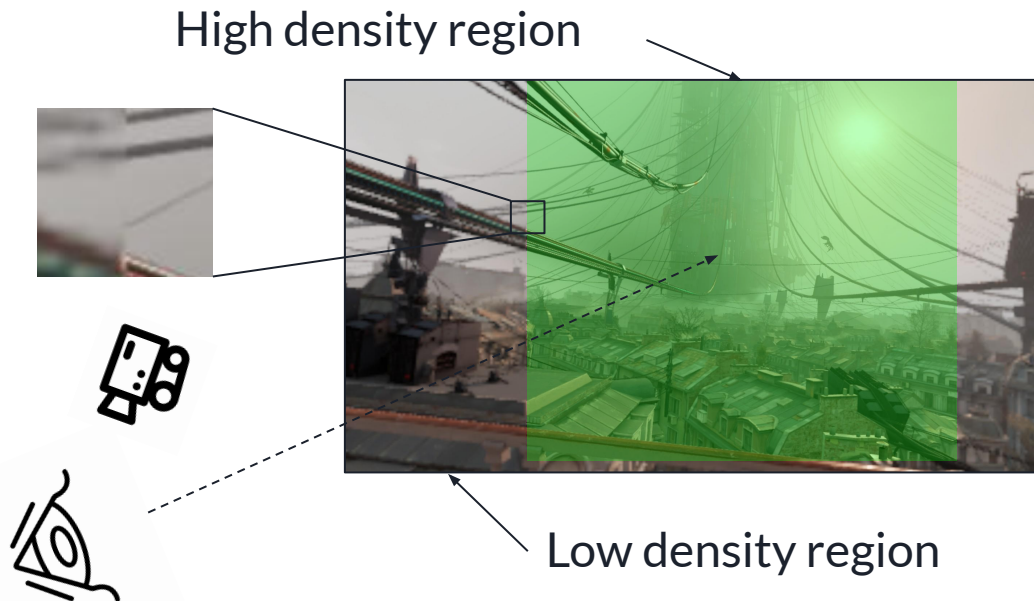
- Run existing PC VR content
 - Designed for desktop class GPUs!
- Smoothly
- On a smartphone chip



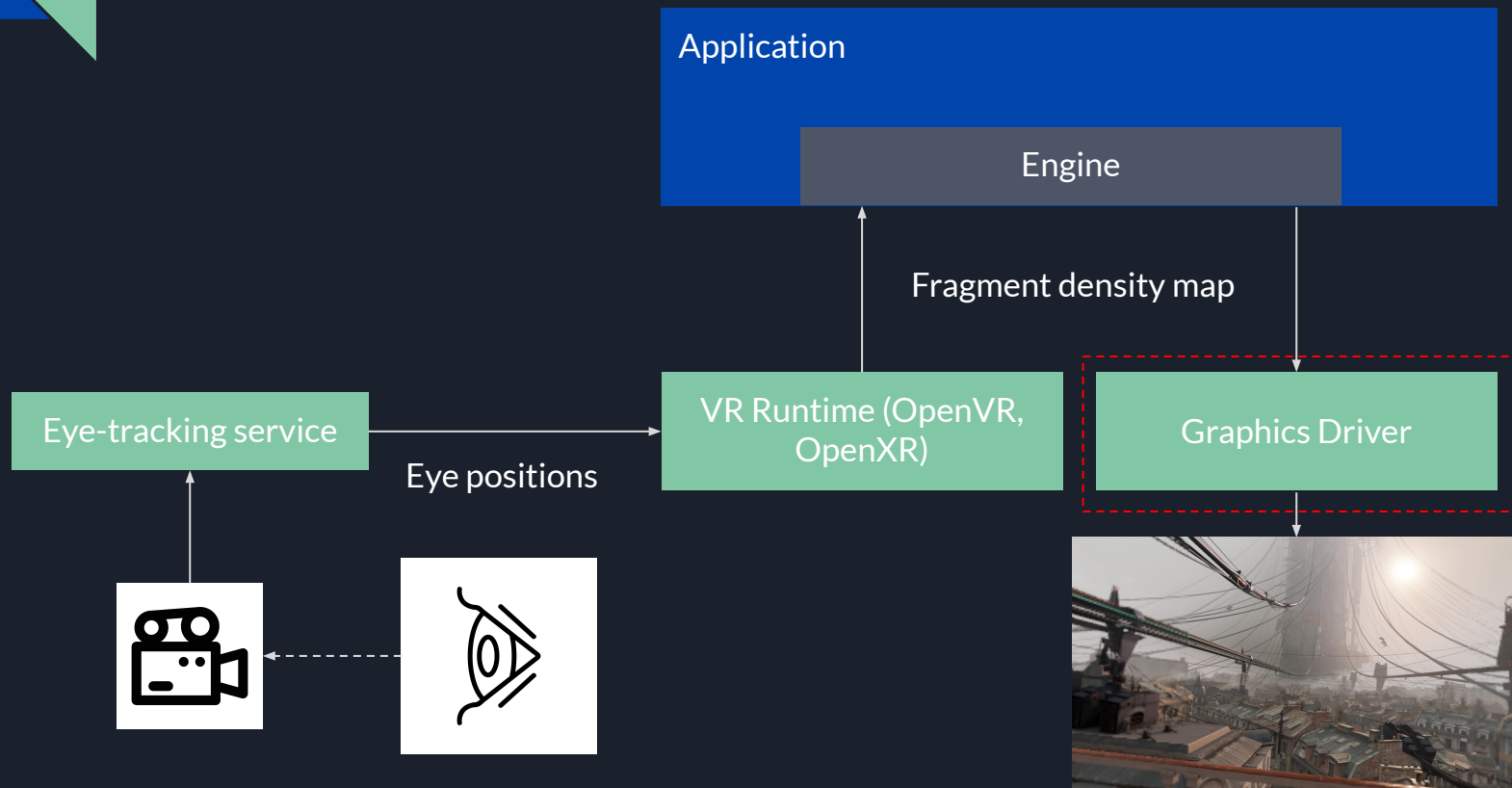
Foveated rendering

- Human eyes have a pretty small high-resolution area
 - The "fovea"
- The rest cannot make out fine detail
- Your brain fills in the rest

Foveated rendering



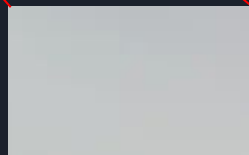
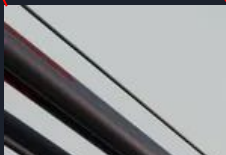
Foveated Rendering Software Stack



Tiled rendering



GMEM



Binning

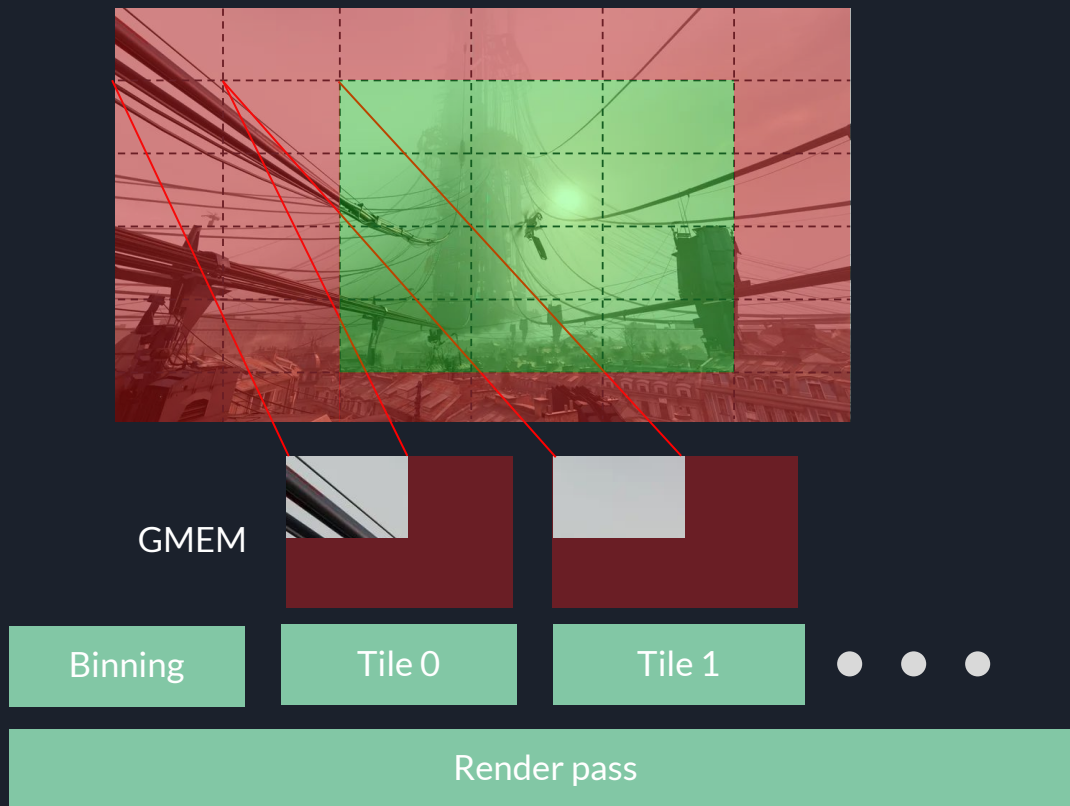
Tile 0

Tile 1



Render pass

Foveated rendering



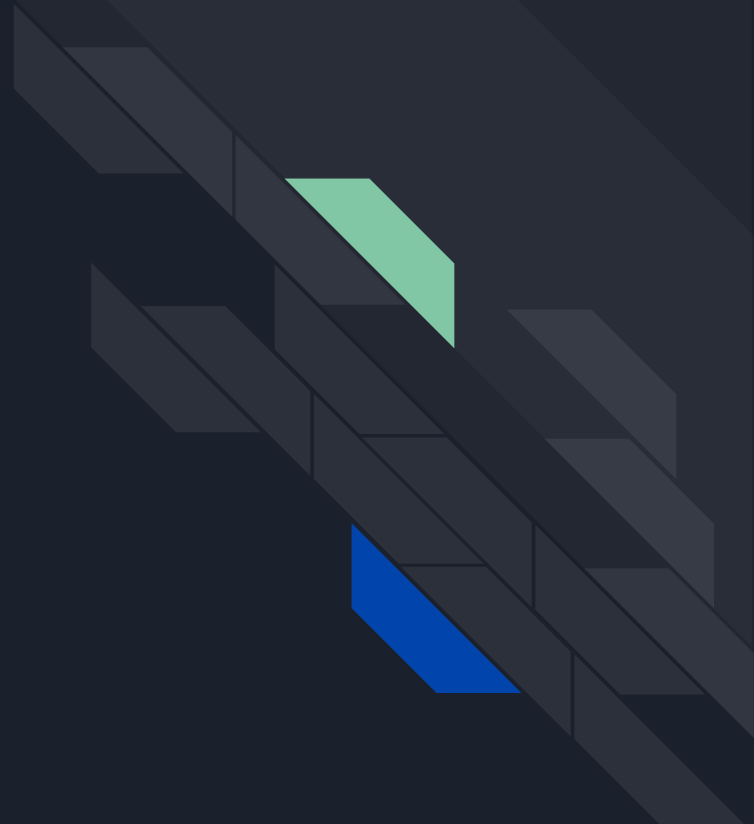


VK_EXT_fragment_density_map

- Cross-vendor Vulkan extension
- Application supplies "fragment density map"
- For each bin ("local framebuffer region"):
 - Take bin center (x, y)
 - Scale down to FDM coordinates
 - Sample FDM: get a *fragment area*
 - Round down to supported fragment area
 - Render bin at that resolution
- Supported fragment area may be only (1, 1)!
 - Means FDM is effectively unsupported

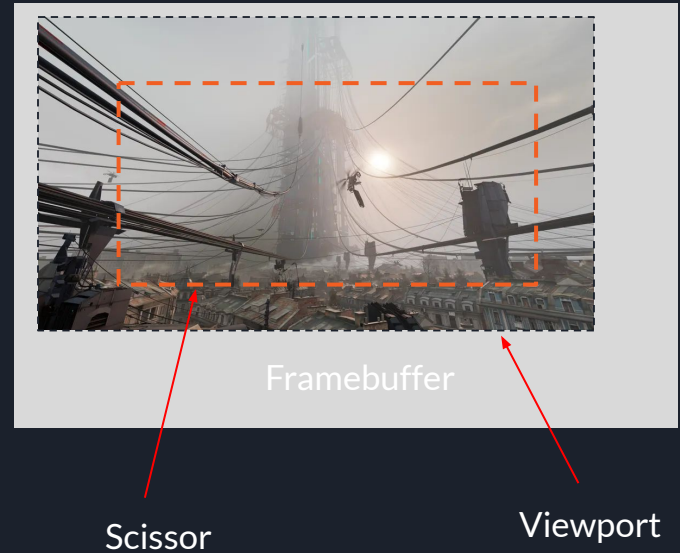
How it Actually Works

Descend into the madness with me



Recap: Viewport and Scissor

- Viewport: the transform from *Normalized Device Coordinates* to final coordinates
 - Determines the final size of the rendered image
 - Normally covers the entire framebuffer
- Scissor: Restricts rendering to a part of the viewport
 - Normally equal to the viewport size, i.e. has no effect
- User may have multiple viewports:
Multi-viewport
 - Vertex shader selects which viewport to use
 - *Not* the same as multiview!

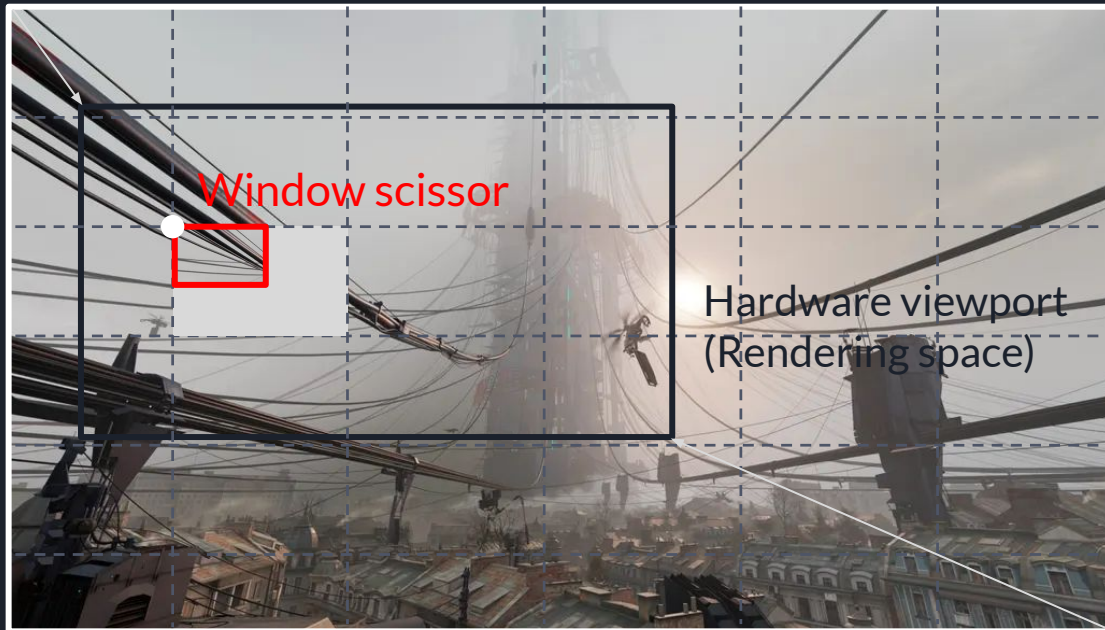


Tiled Rendering on Adreno

- Window offset: Subtract when accessing GMEM
- Window scissor: combine with user's scissor
- Together, these features make tiled rendering mostly transparent
- Driver programs these per tile and can ignore tiled rendering elsewhere

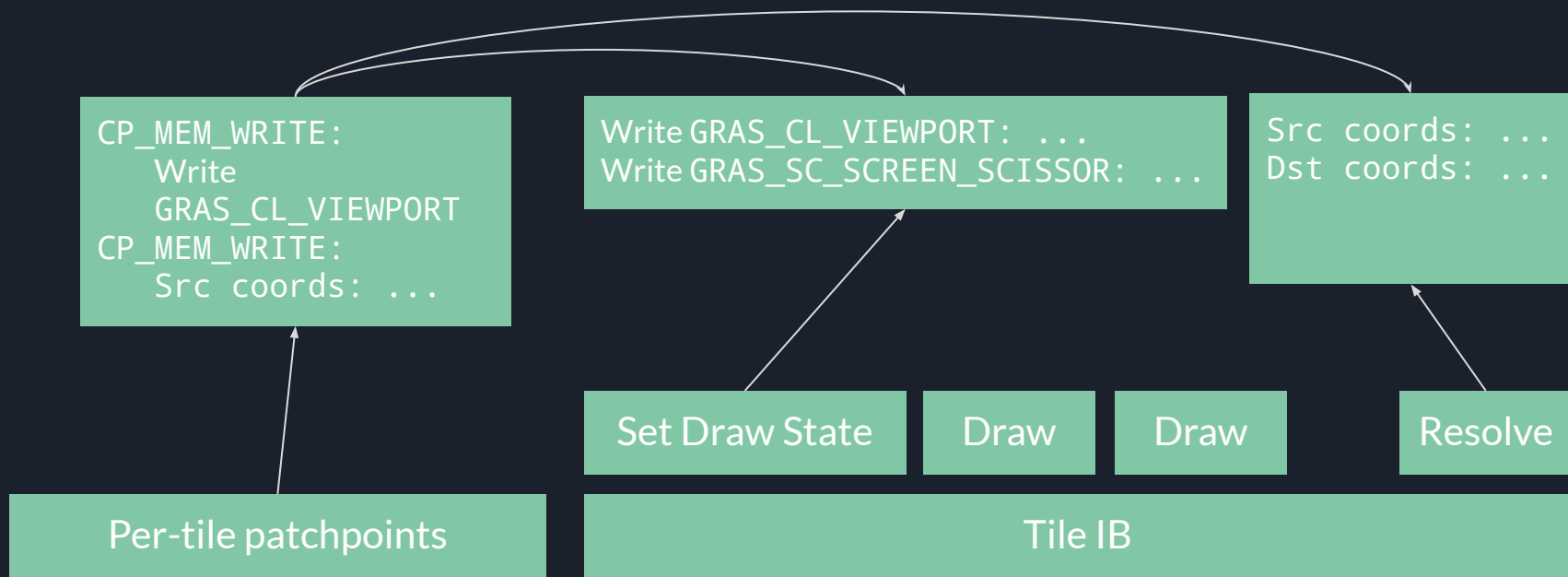


Foveated Rendering on Adreno



User viewport
(Framebuffer space)

Foveated Rendering Patchpoints





Multiview Rendering

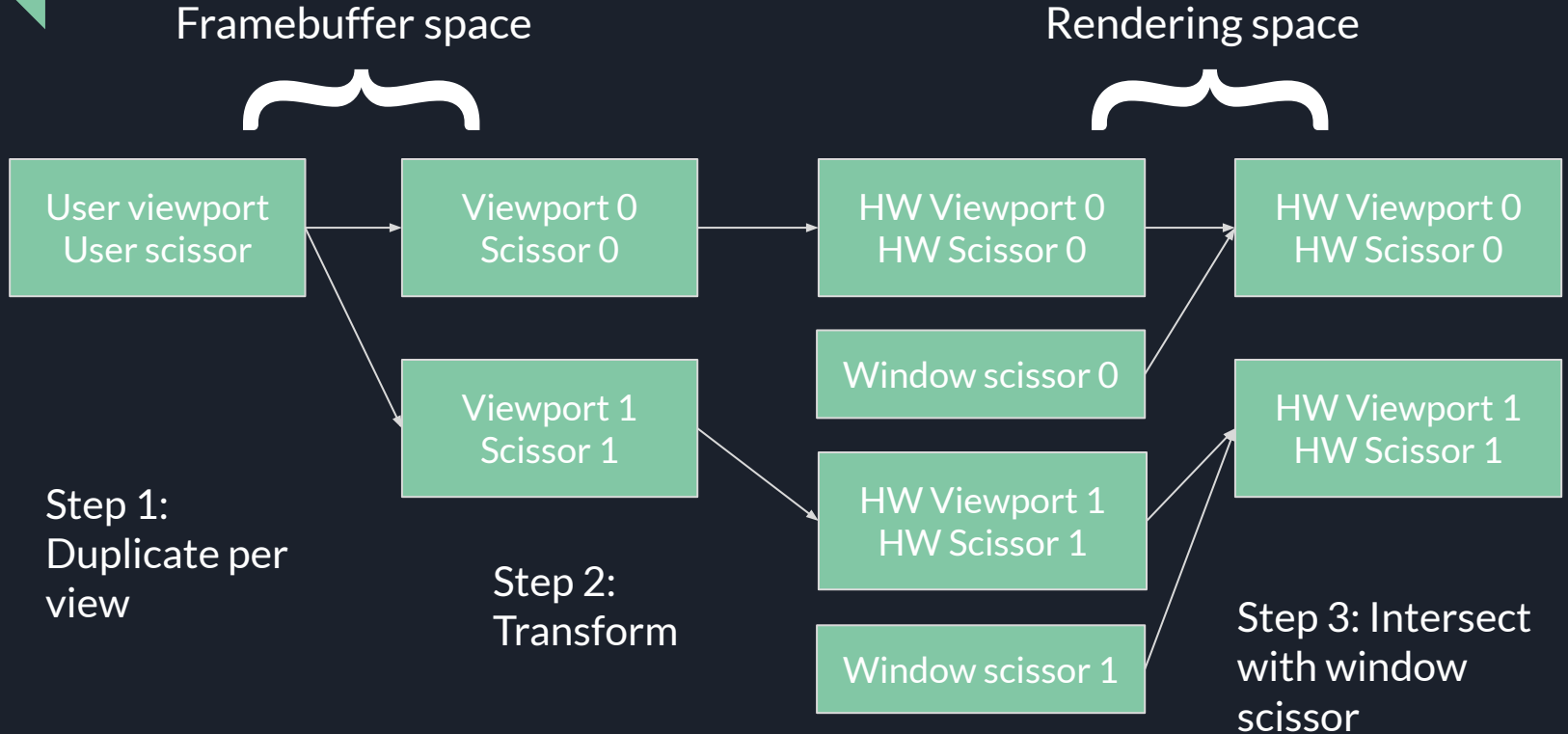
- Layered rendering: each eye is a separate layer
 - As opposed to *doublewide*, where both eyes are side-by-side in the same image
- Multiview: Each draw renders to both layers at once
 - Each layer is a *view* in Vulkan parlance
- HW window scissor, window offset are *not* per-view
- Viewports, scissors are *not* per-view
- But the fragment area may be:
 - Different eyes may look at different things!



Multiview Rendering

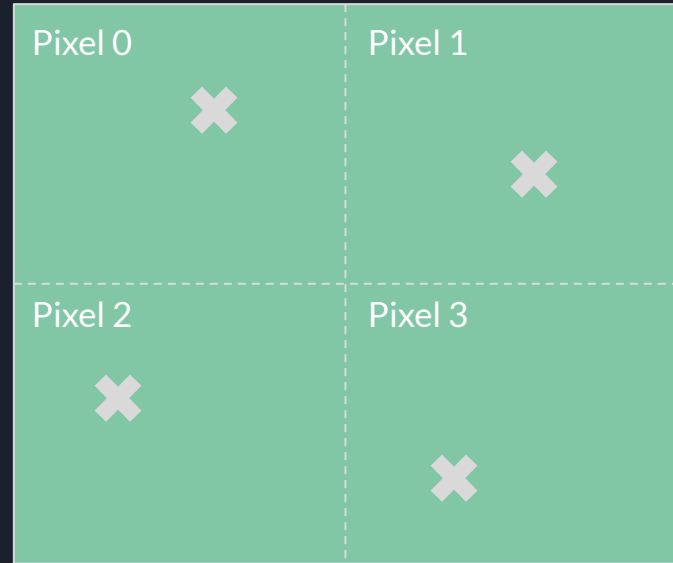
- HW bit: add view index to viewport index
- HW viewport can now vary per view
- Hidden from the user
- User uses multiple viewports: game over
 - Fall back to intersecting fragment area over all views
 - Don't do this!

Multiview Rendering



Multisample Rendering

- Sample positions are effectively scaled up
- When resolving: average, then duplicate
- When storing: nothing we can do, effectively disables FDM
 - Sample positions are not compatible
 - Kills FDM with RenderDoc 😞
- Don't use `STORE_OP_STORE/LOAD_OP_LOAD` with multisample!



Fragment

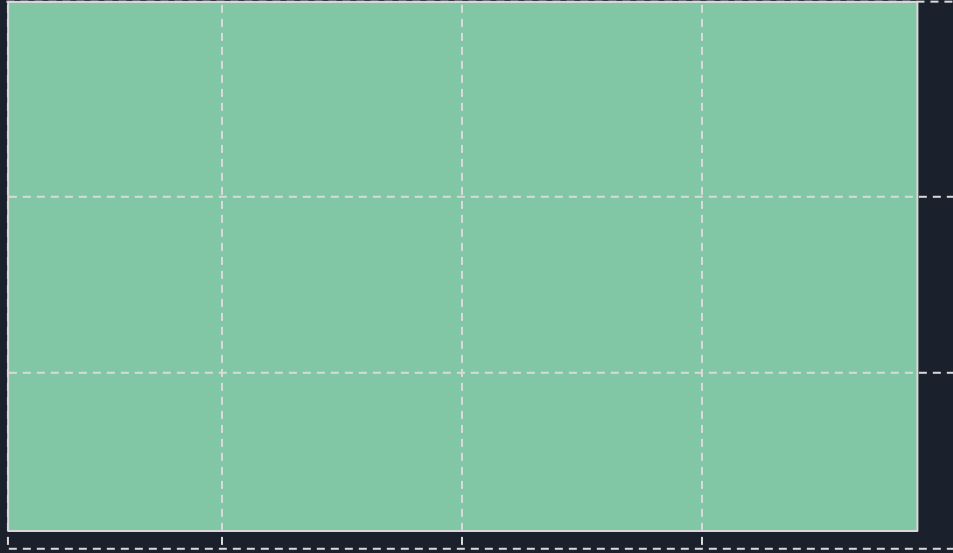


Fragment Density Map Offset

- Bins are too coarse: ~200-300 pixels
- Need smooth transitions when eye gaze is moved
- Enter `VK_EXT_fragment_density_map_offset`
 - Originally `VK_QCOM_fragment_density_map_offset`
 - The cross-vendor EXT version adds support for dynamic rendering
- The actual fragment density map is now fixed
- The user supplies an offset for each frame
- Can shift the pattern by a fraction of a tile



Fragment Density Map Offset

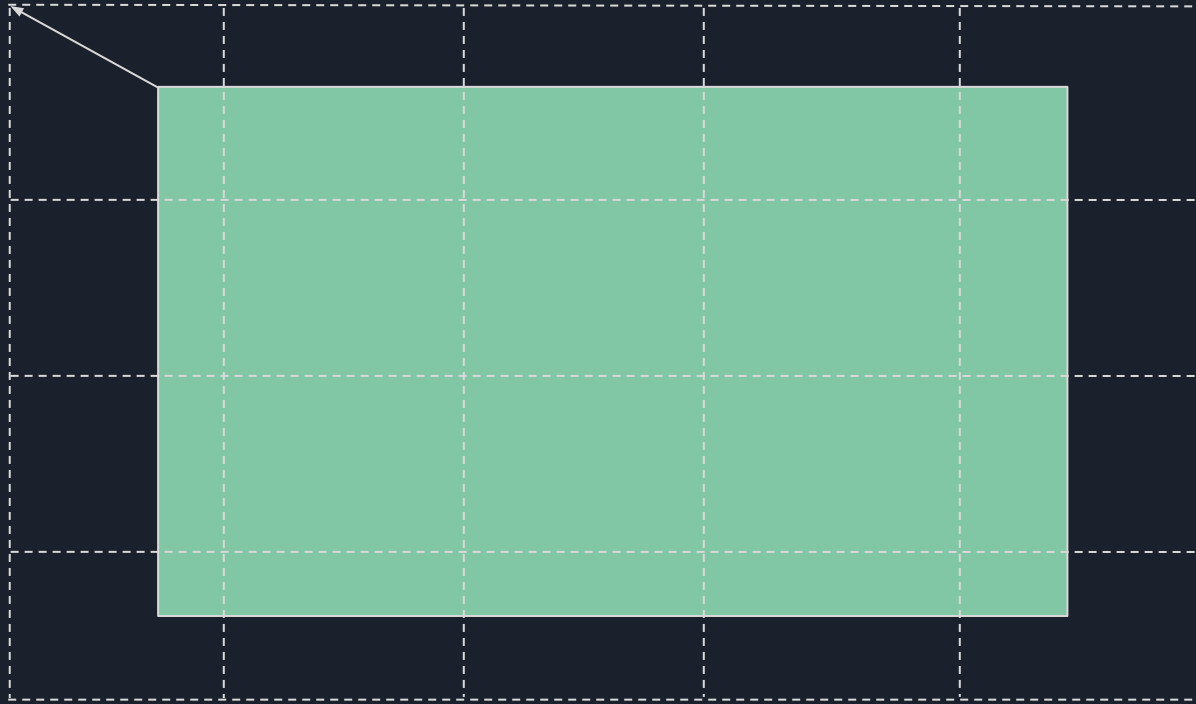




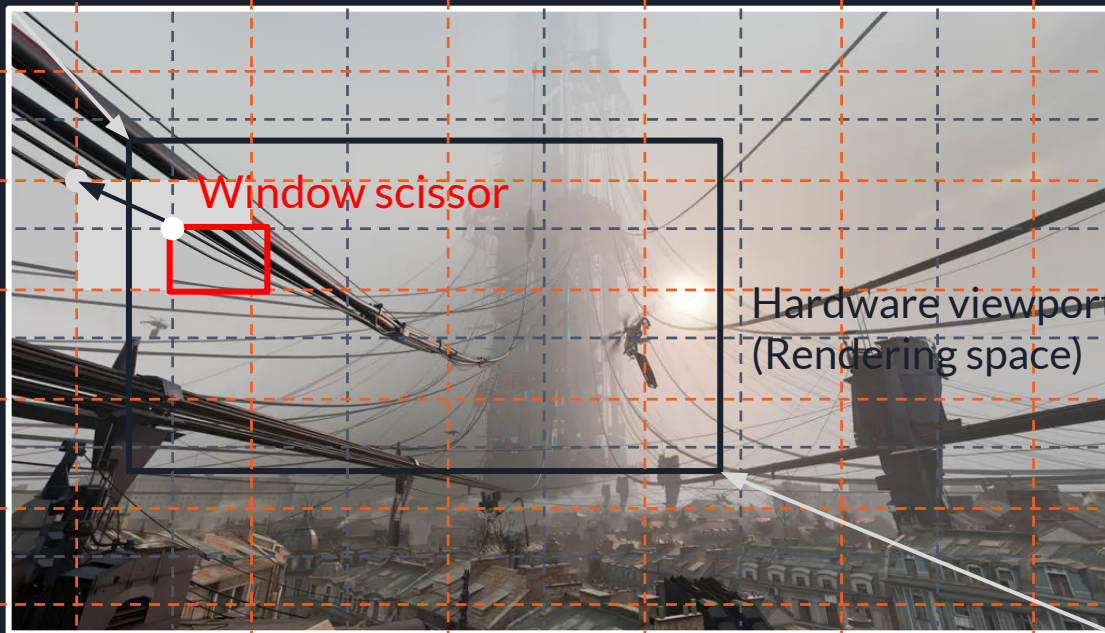
Fragment Density Map Offset



Fragment Density Map Offset



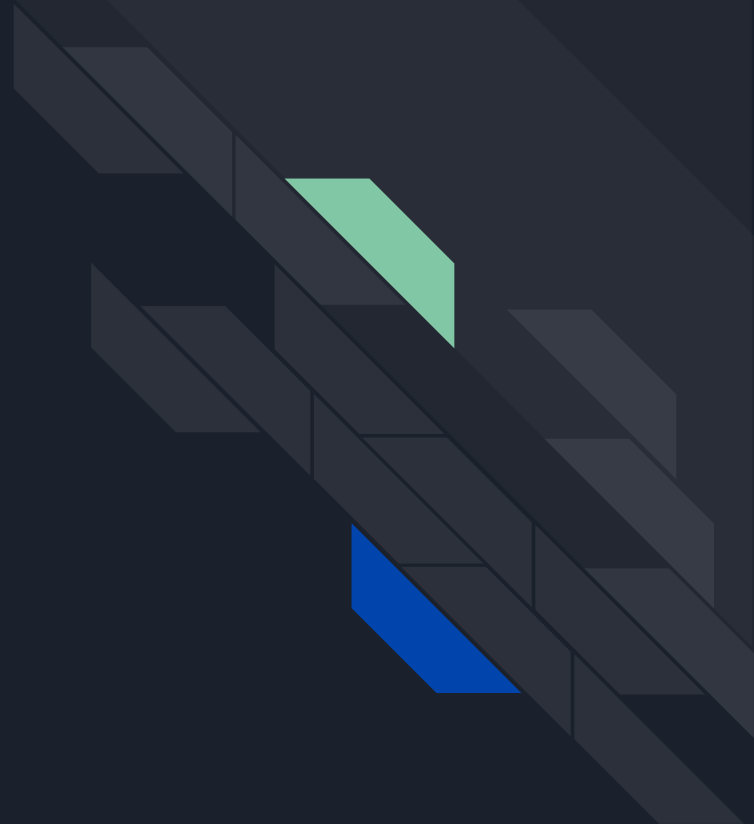
Fragment Density Map Offset



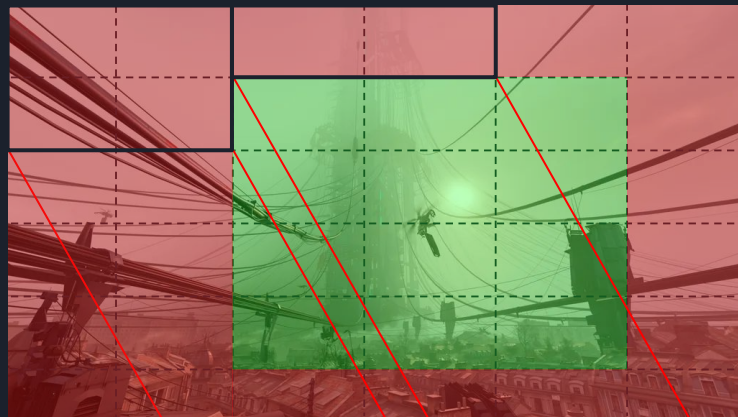
Hardware viewport
(Rendering space)

User viewport
(Framebuffer space)

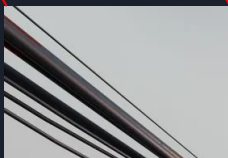
Advanced Features



Bin Merging



GMEM



Binning

Tile 0

Tile 1



Render pass



Bin Merging

- Merge visibility information when rendering bins
 - The only actual HW feature required
 - The rest is handled by the driver
- Only possible if bins are in the same *visibility stream*
 - Rectangular groups of tiles encoded at the same time
- Introduced in A650
- Enhanced on A750 (GPU used in Steam Frame)
 - Ability to use all tiles in visibility stream
- Non-rectangular merged tiles since A8xx+
 - Not implemented yet on turnip

Bin merging

Bin Layout

Viewport 0 - View 0



Viewport 1 - View 1



<https://gitlab.steamos.cloud/frame-public/frame-developer-tools/-/blob/main/perfetto/README.md>

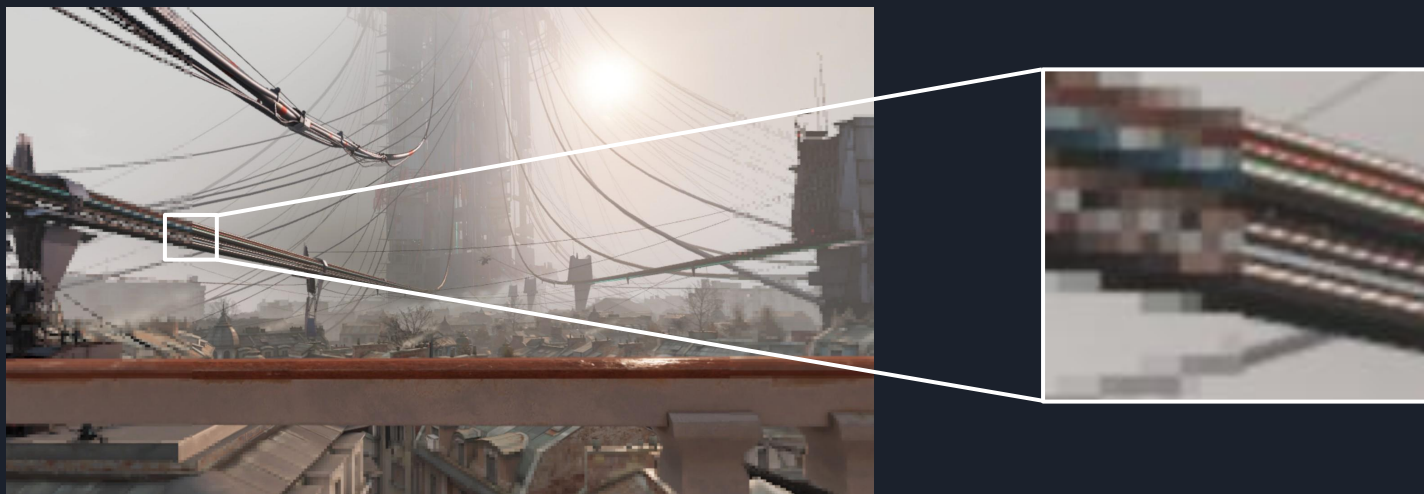


Subsampled Images

- Scaling up when resolving uses point sampling
 - Each fragment in render space is duplicated
 - Otherwise, there would be artifacts along bin edges
- However, this reduces image quality in low-resolution areas

Subsampled images

Subsampled OFF:



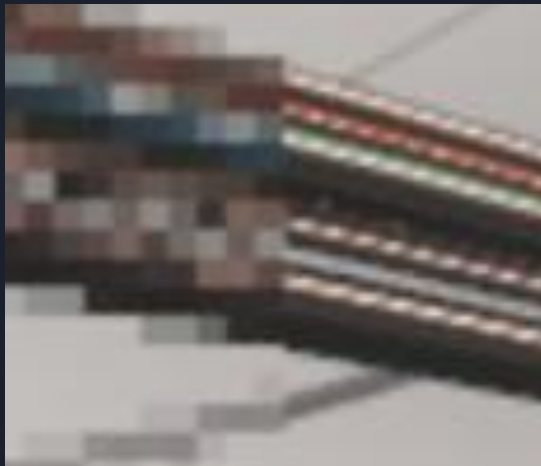


Subsampled Images

- Store low-resolution image directly in memory
- Scale up when sampling
 - Either in-game in a post-pass, or in the VR compositor (not yet on Steam Frame!)

Subsampled Images

Subsampled OFF:

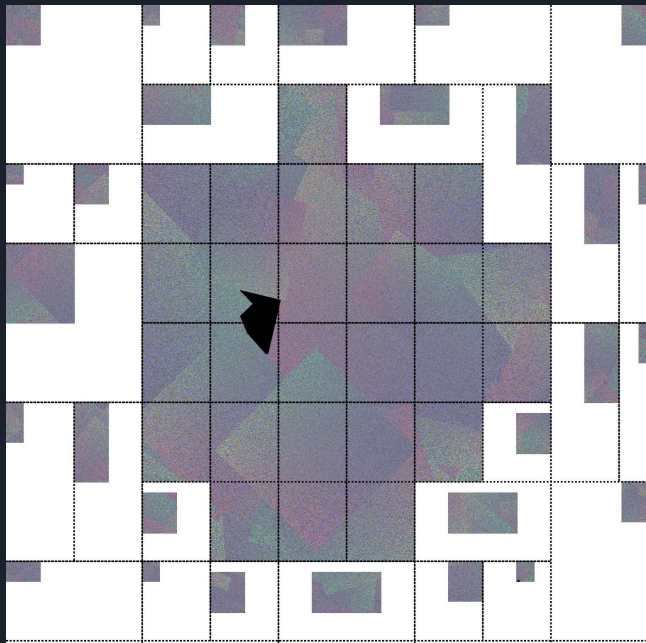


Subsampled ON:

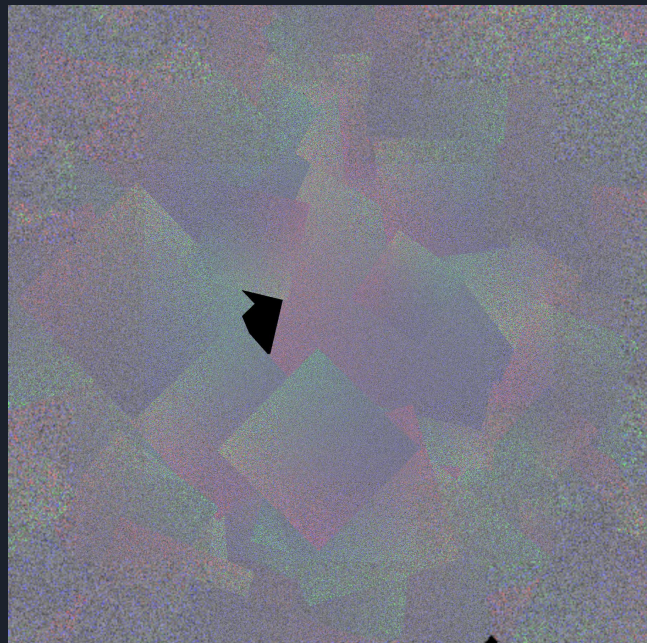


Subsampled Images

Subsampled space



Framebuffer space





Subsampled Images

```
uniform sampler2D mySampler;  
  
texture(mySampler, coord);
```

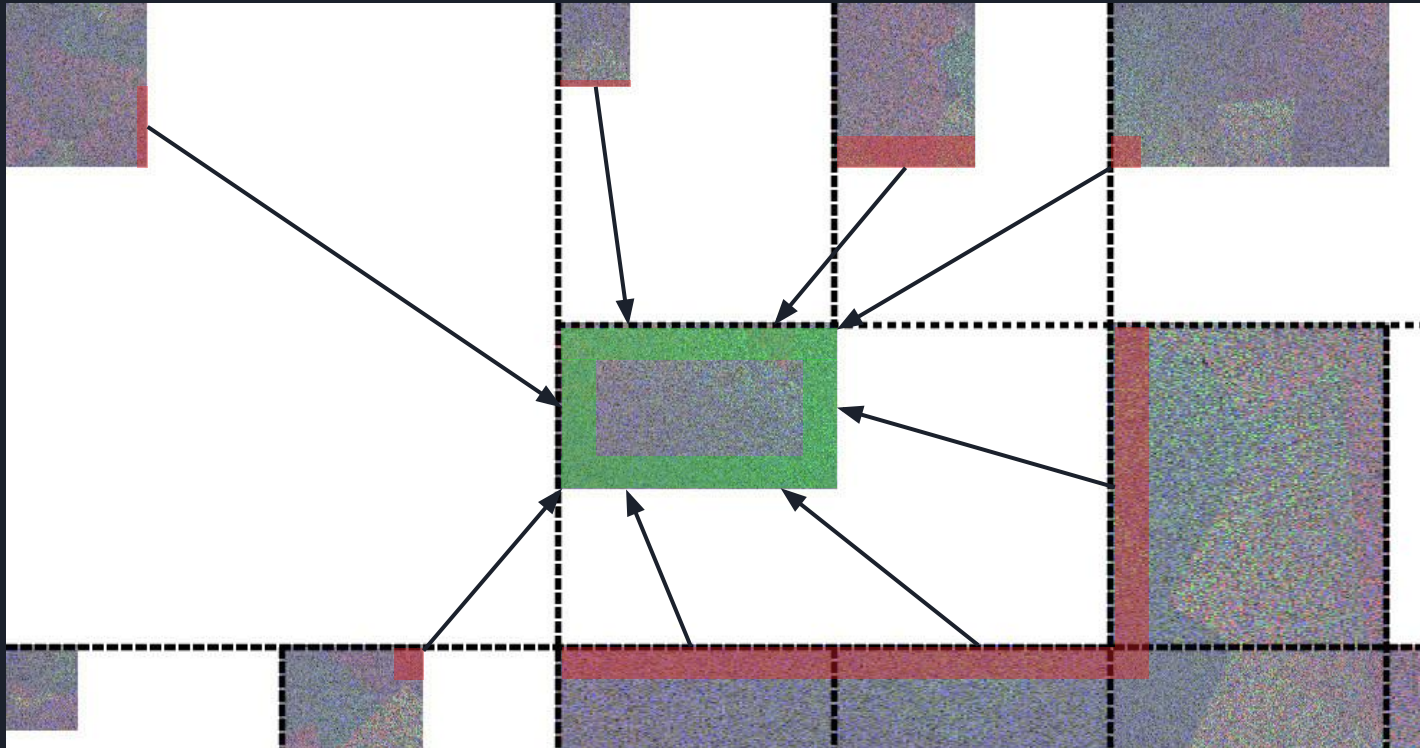
```
uniform sampler2D mySampler;  
uniform Metadata metadata;  
  
ivec2 bin =  
    getBin(metadata, coord);  
vec4 scale =  
    getScale(metadata, bin);  
coord.xy = coord.xy * scale.xy  
+ scale.zw;  
  
texture(mySampler, coord);
```



Subsampled Images

- What happens when a sample straddles two bins?
- Create *aprons* around each bin
- Blit from surrounding bins

Subsampled Images





Subsampled Images Restrictions

- Need an immutable sampler
 - `VK_SAMPLER_CREATE_BIT_SUBSAMPLED_EXT` in the immutable sampler signals driver to insert extra code
- Need combined image+sampler
 - Need extra space for metadata descriptor



Subsampled Images Restrictions

- Cannot draw the same image twice in different renderpass
 - i.e. no LOAD_OP_LOAD in render pass
 - Seriously constricting
 - Can't do overlays etc.
- Currently not supported in SteamVR
 - App must do a copy itself
 - In *Half-Life: Alyx*, done in the bloom post-processing pass



Layered Fragment Density Map

- DX11 does not have multiview
- Existing content uses layered rendering instead
 - Looping over draws is done manually
 - Vertex shader writes `gl_Layer` to indicate which eye to write to
 - No limit on number of layers
- Base `VK_EXT_fragment_density_map` does not work well with layered rendering
 - The fragment density map and offsets must be shared between layers
 - Makes using it for foveated rendering impossible!



Layered Fragment Density Map

- The fix: `VK_VALVE_fragment_density_map_layered`
- App promises to only render to a limited number of layers
- In return, driver samples fragment density map per-layer
- Used by for existing PCVR content
- Native apps encouraged to use multiview



Layered Fragment Density Map

```
void main() {  
    ...  
    mat4 xform =  
    transforms[gl_InstanceIndex];  
    gl_Position = xform *  
    in_Position;  
    gl_Layer = gl_InstanceIndex;  
}
```

```
void main() {  
    ...  
    mat4 xform =  
    transforms[gl_InstanceIndex];  
    gl_Position = xform *  
    in_Position;  
    gl_Layer = gl_InstanceIndex;  
  
    // inserted by driver  
    gl_ViewportIndex = gl_Layer;  
}
```



Future Work

- Implement on A8xx
 - Many features now done in HW
 - Need to get it working
- Make FDM work with RenderDoc record/replay
- Implement subsampled images in SteamVR & engine integrations
- Smoother images without subsampled images
 - Downsample and upsample in the compositor?

Questions?

