Surface Aggregation

jbauman@

Overview

- What are Surfaces?
- History
- Surfaces in Chrome
- Compositor Frames
- Surface Aggregation
- Future work

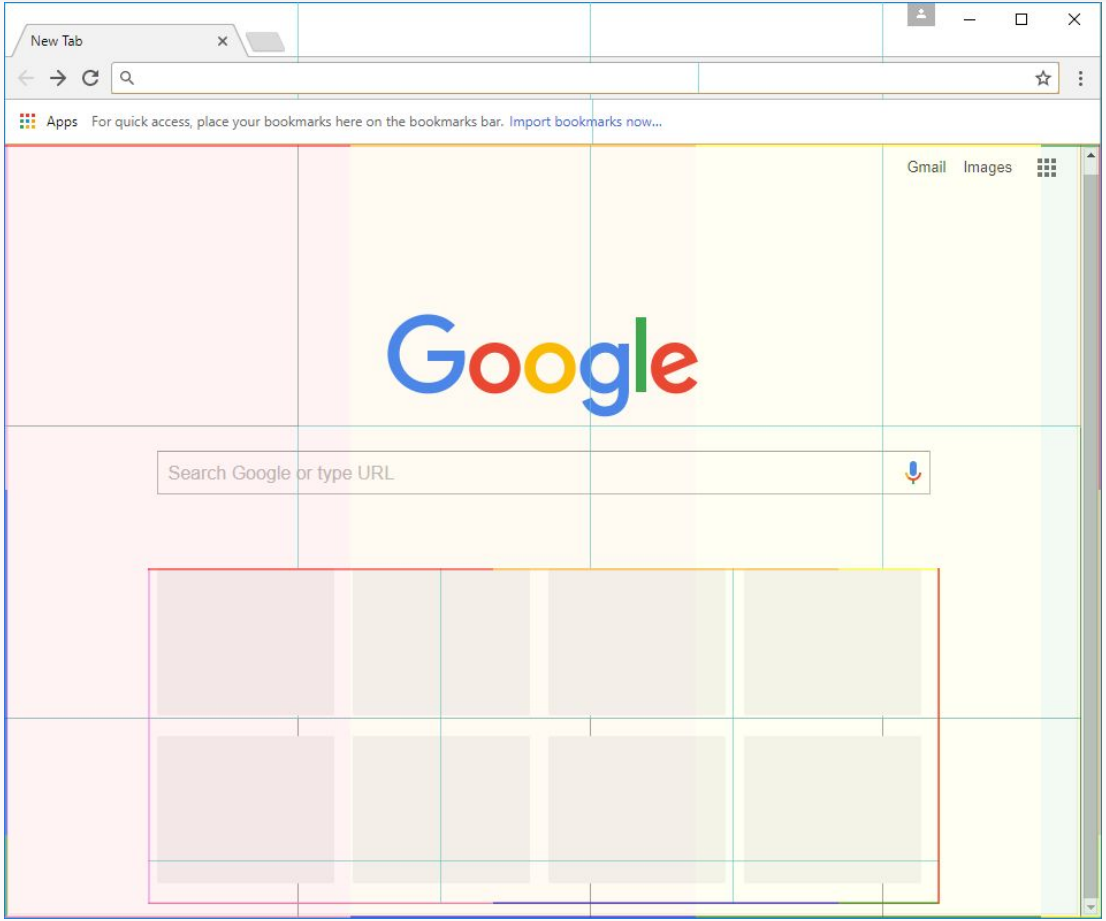
Surfaces

What are Surfaces?

Surfaces are a concept to allow graphical **embedding** of **heterogeneous** **untrusting** clients **efficiently** into one scene

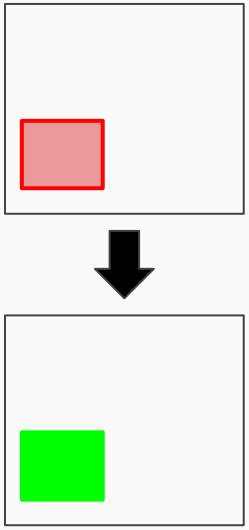
Still in flux

What are they used for?



Surface

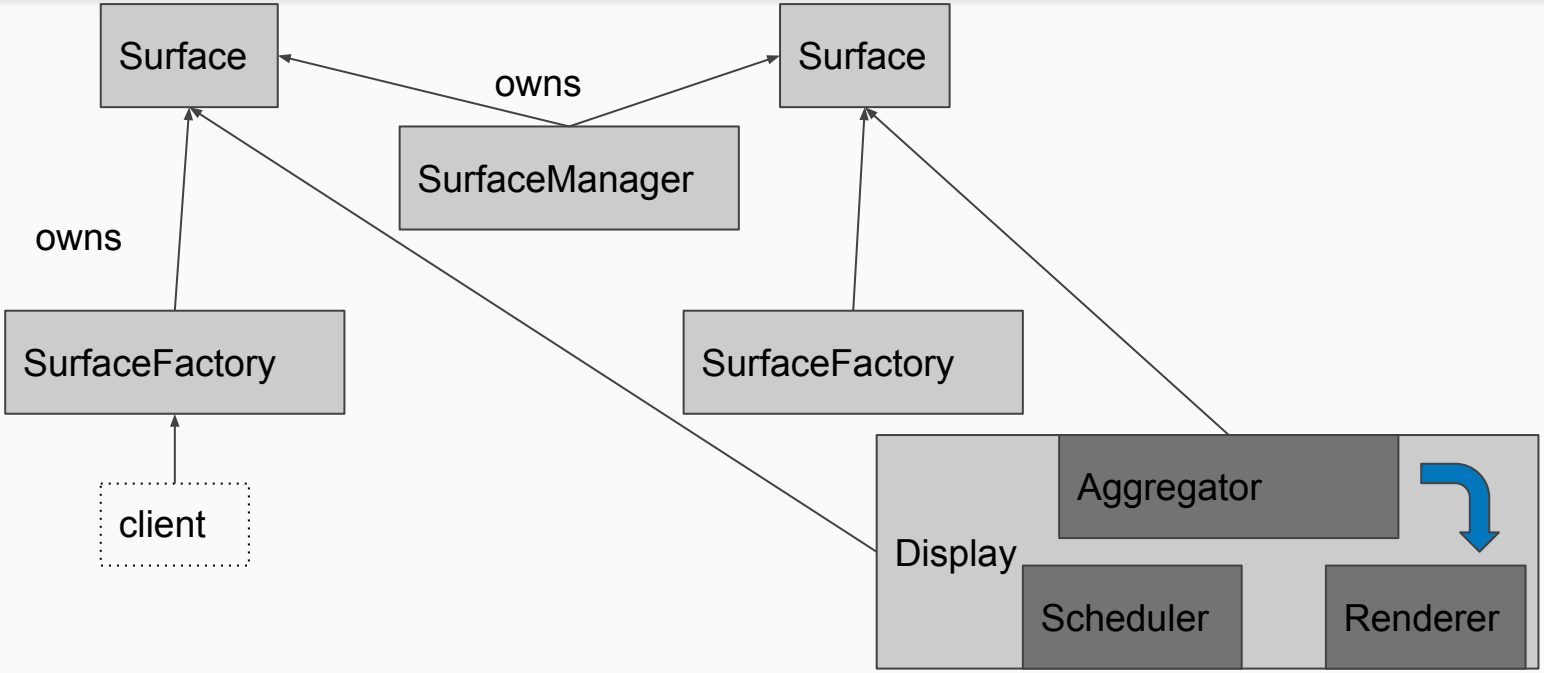
- Has an ID
- Represents instructions about how to render an area.
- Can embed other surfaces
- Represent a sequence of frames
- Updated independently
- (Conceptually) have size and scale factor



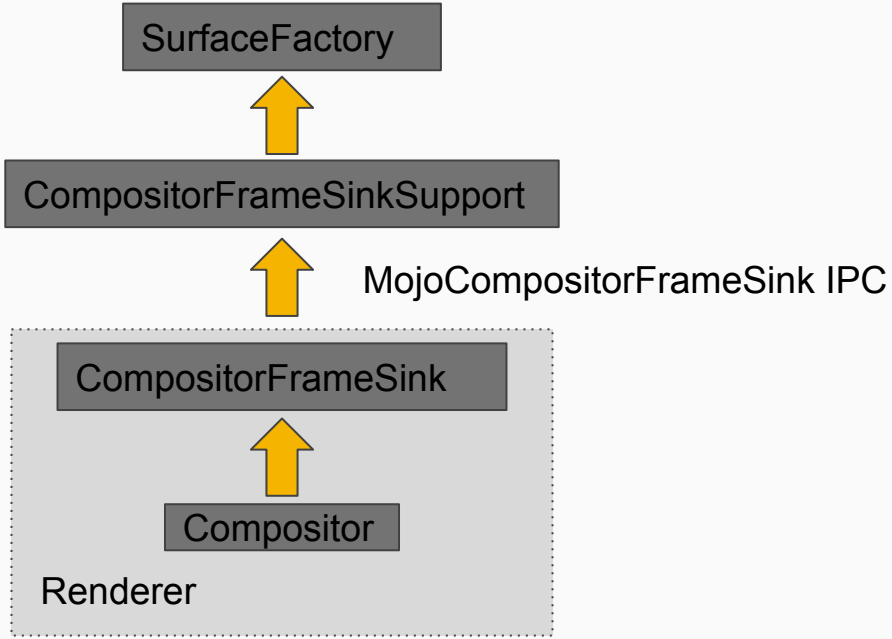
Surface classes (in cc/Surfaces)

- **Surface** represents a single surface, has draw callback
- Can put **CopyOutputRequests** on surfaces
- Has one active frame and one pending frame
- **SurfaceSequence** handles refcounting

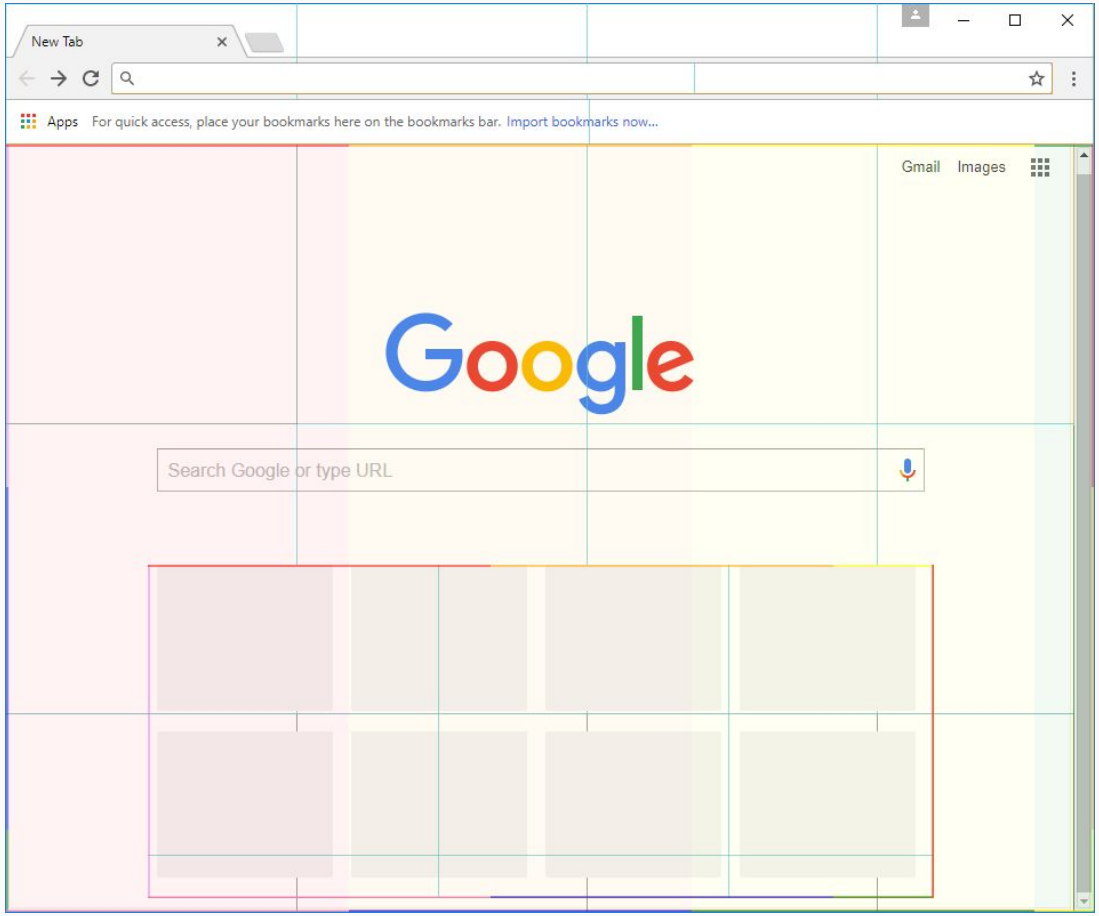
Surface classes



Surface classes



Surfaces in Chrome



History

Originally started for mojo - many untrusted embedders

Added in Chrome

Previously child frames went into the browser compositor, and browser compositor spit out CompositorFrames directly into final Renderer

Surfaces in Mus

Move Display to GPU process.

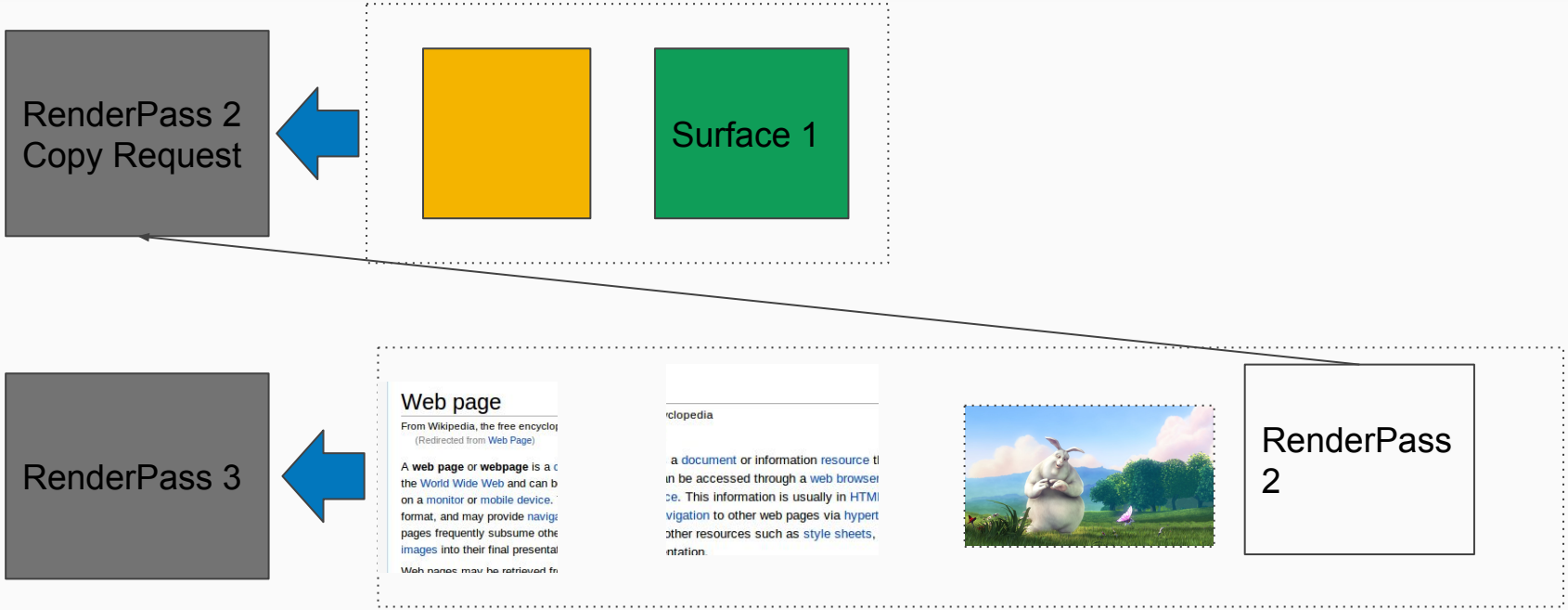
Mus window server takes place of browser compositor.

What’s a Frame?

What’s in a Frame?

- Metadata
- A list of resources (with IDs)
  - GPU mailbox ID
  - Texture type
  - etc
- A list of RenderPasses

Render passes



Quad

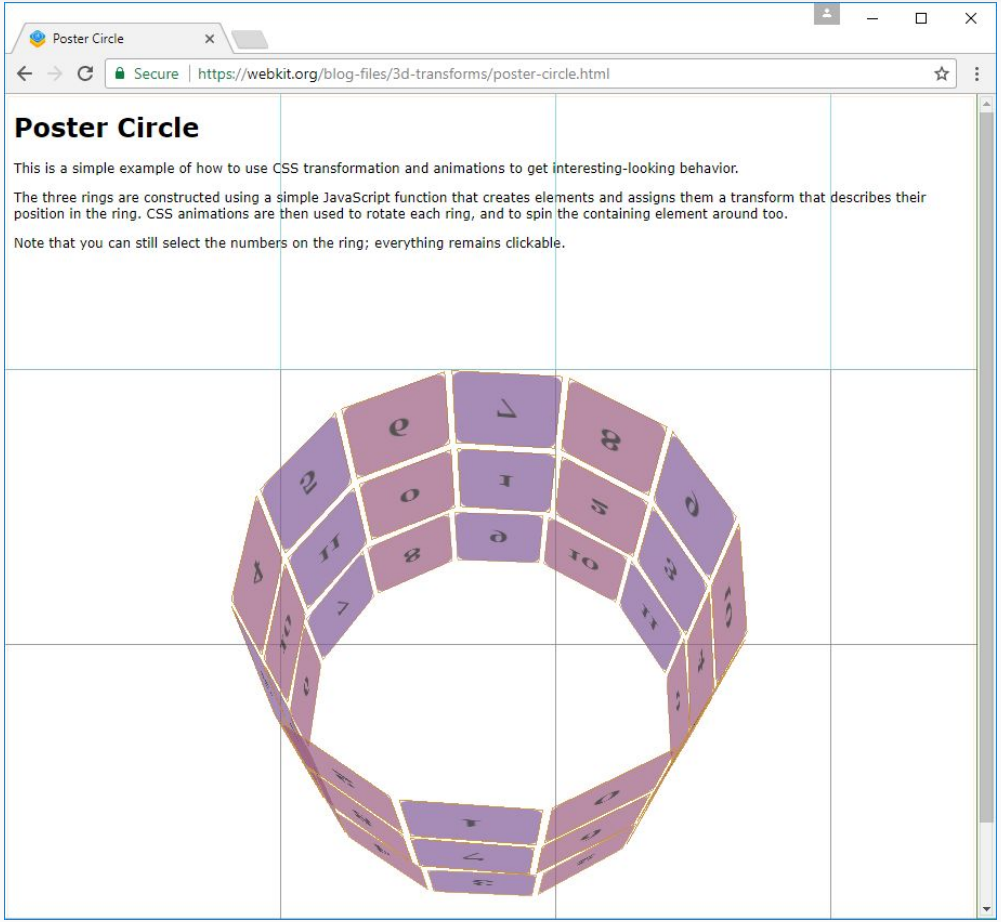
Rectangle

References SharedQuadState

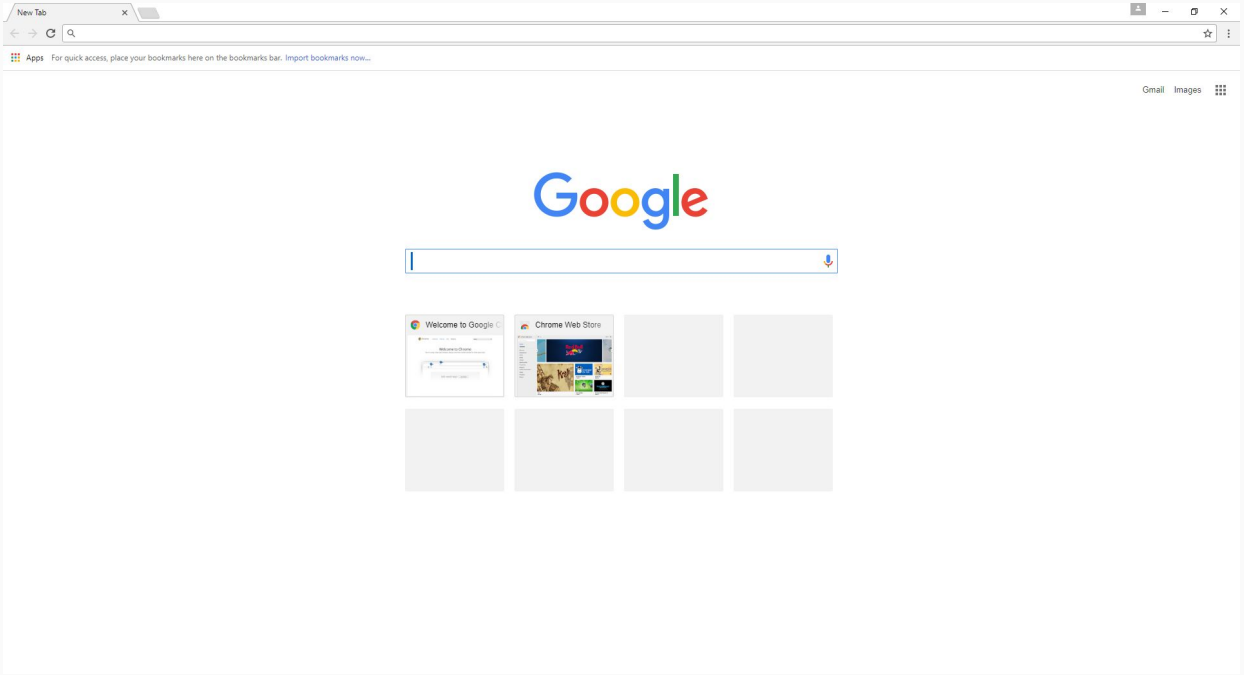
- Clip rect
- Transform to RenderPass
- Opacity

Resource references

Complicated quad transforms



Damage Rect


New Tab
Apps
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
Gmail
Images
Google
Welcome to Google
Chrome Web Store

Metadata

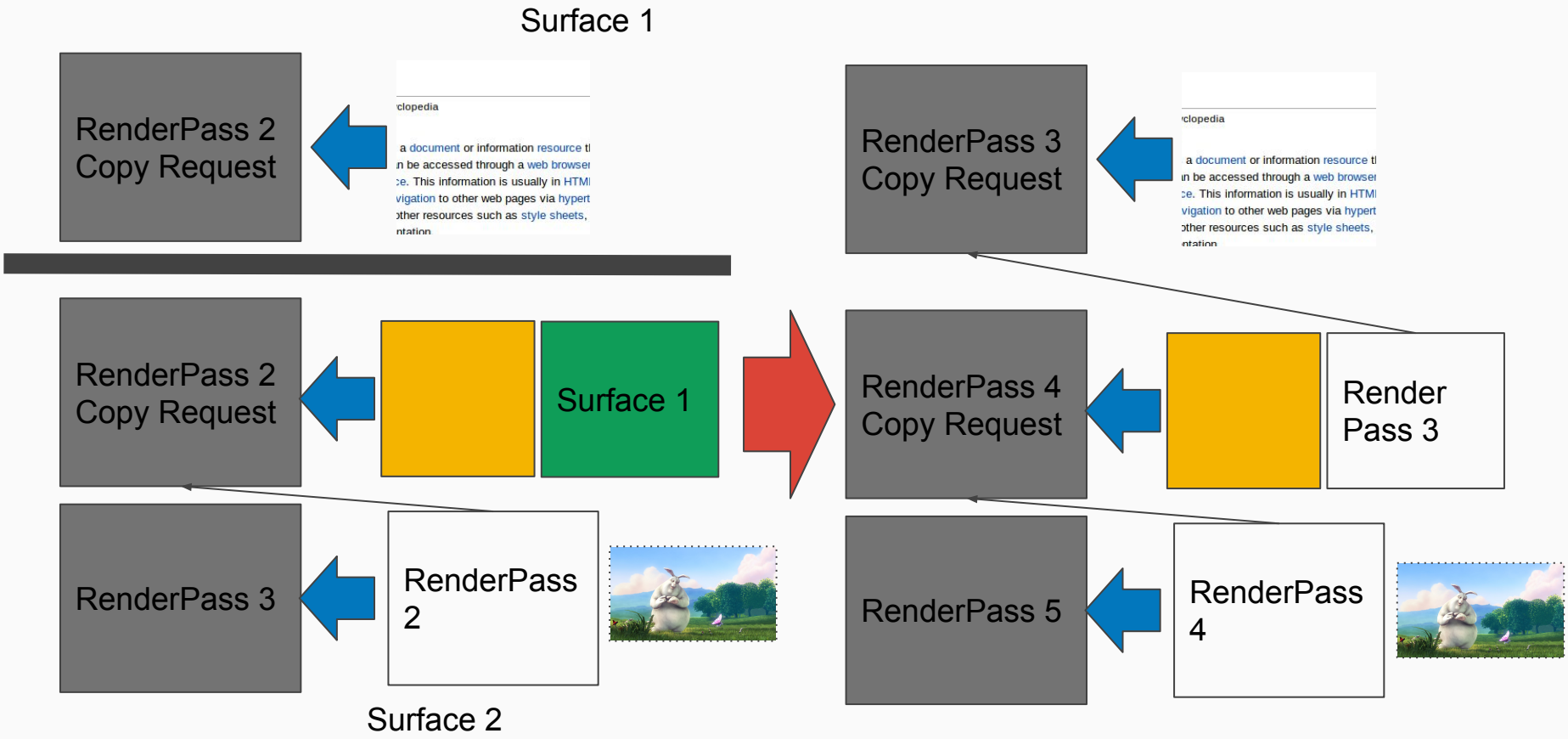
Latency info

Top controls info

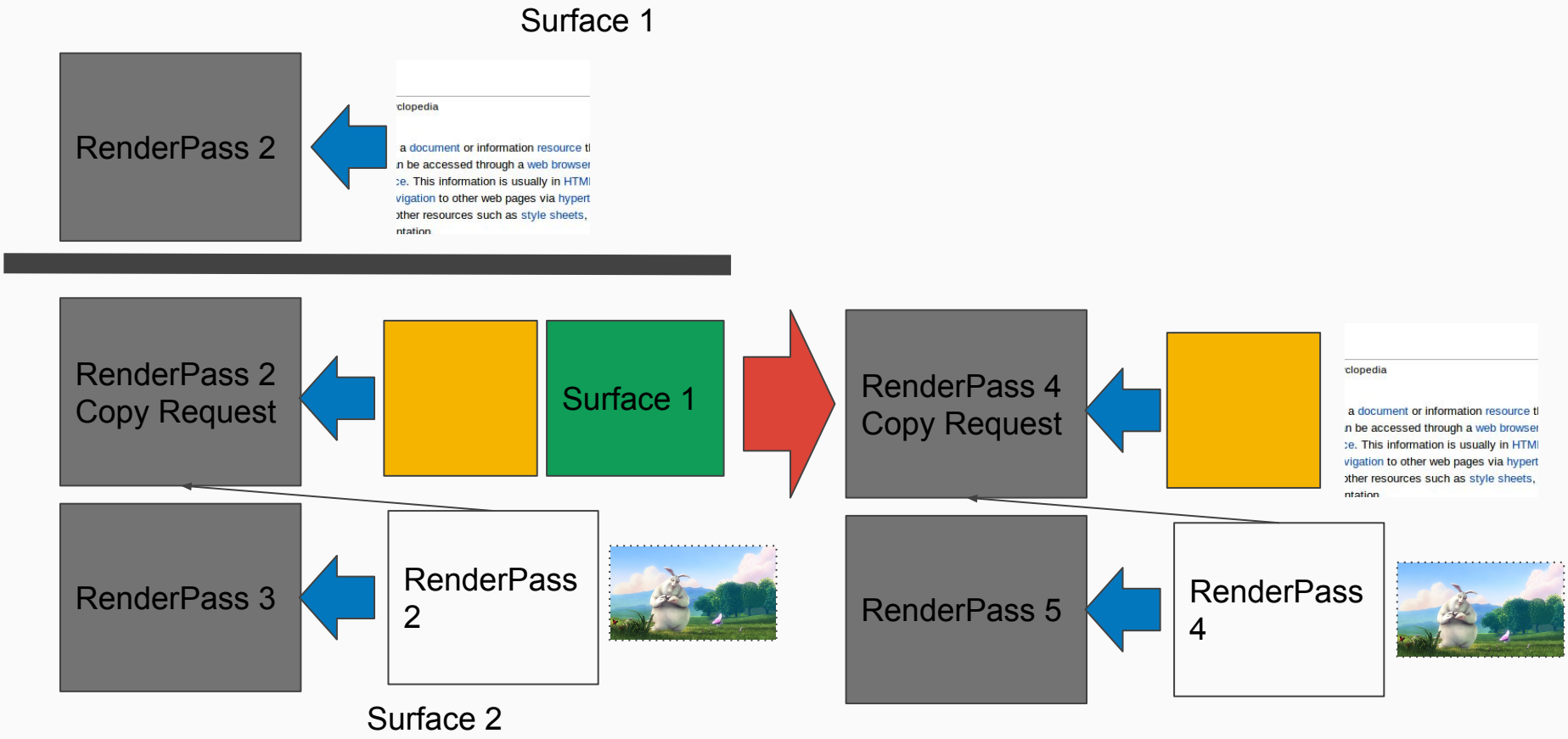
Child Surfaces

Surface Aggregation

Surface Aggregation



Optimizations



Damage Rectangle

Calculate minimal damage rect

RenderPass filters and CopyOutputRequest make complicated

Output one per Renderpass - tells renderer what to draw

Calculated using frame index

Try to avoid aggregating outside of it.

First pass

Start at root surface

Walk tree

Calculate damage rect

Map child to parent resource ids

Second pass

Walk tree again

Keep track of transform to current RenderPass (multiple Surfaces)

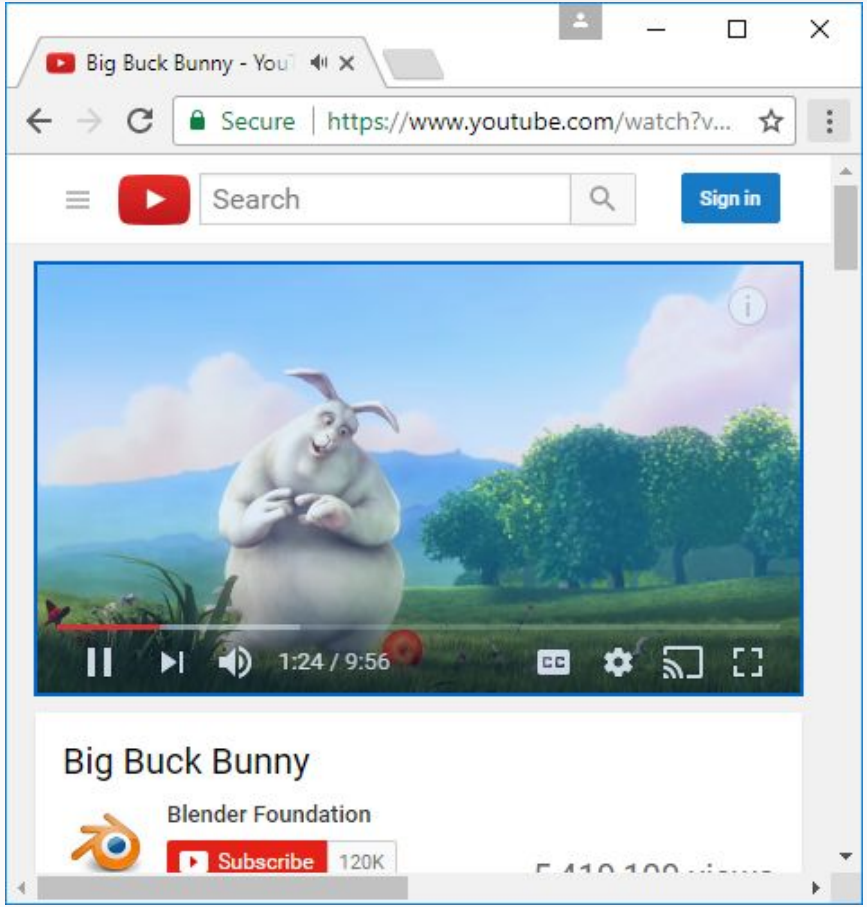
Aggregate everything within damage rect.

Add quads to list.

Hidden CopyOutputRequests

Some Surfaces may not actually be referenced by a quad, but have a CopyOutputRequest. Track them in first pass, and, aggregate them completely first.

Side note: overlays



Draw scheduling

Damage-triggered

If Surface gets new frame, propagate to Displays containing it.

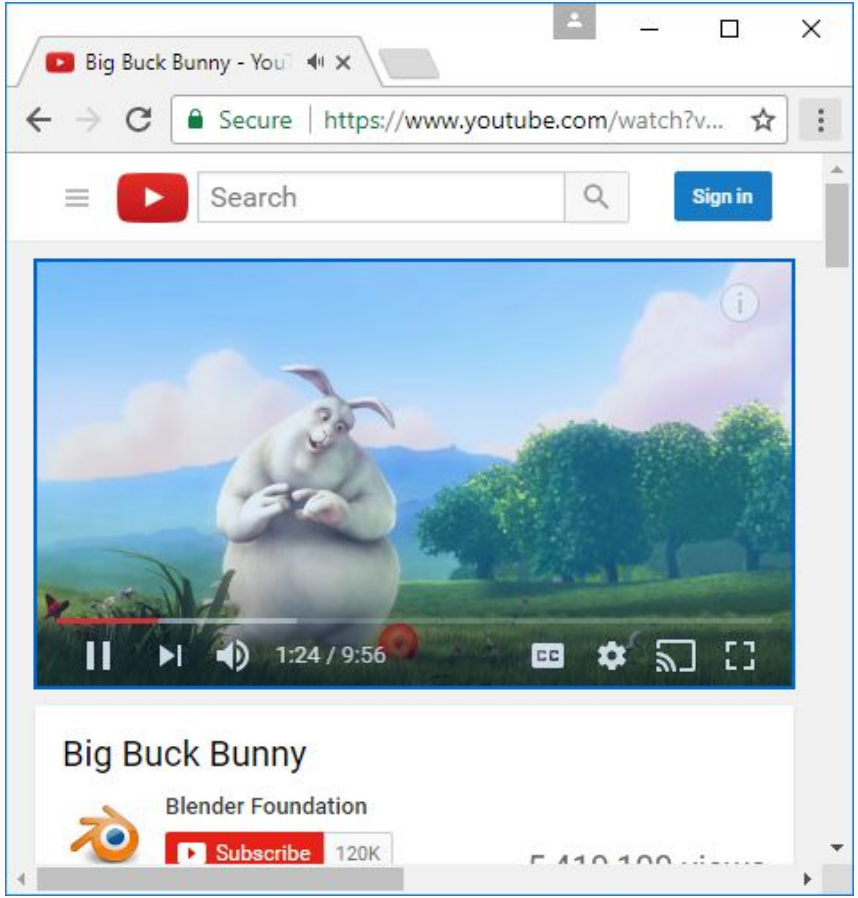
If it’s added to parent, parent will cause draw.

Smart timer scheduling

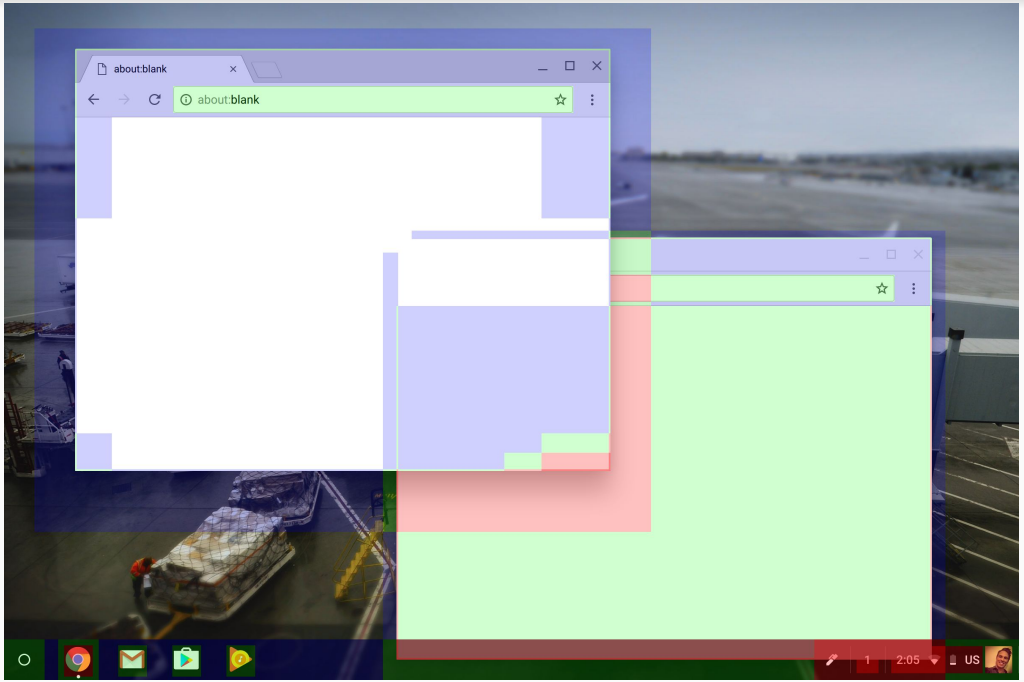
Draw callback

Future work

Move damage rect calculation later



Occlusion culling



Improve synchronization

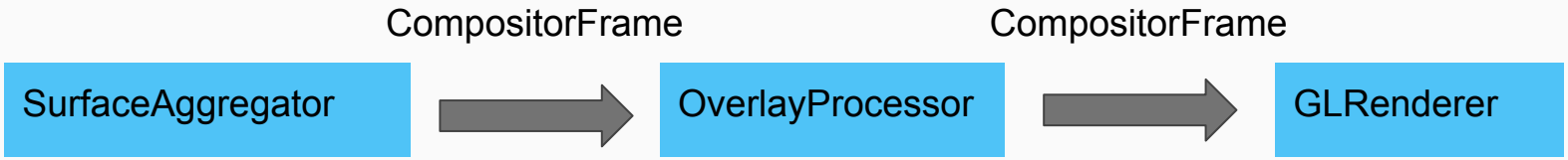
For fsamuel's talk@ soon

Current summary - surfaces draw as soon as they get there - missing surfaces don't draw anything.

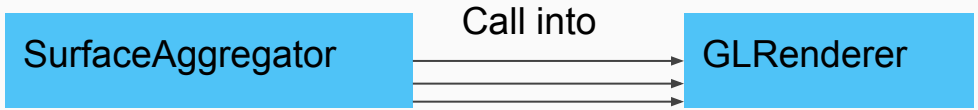
Minimal synchronization to prevent surfaces from dying before they're used

Combine with final rendering

Now



Future



CompositorFrame diffing

IPC of update to entire frame is expensive - attempt to diff it

May aggregation can process only diff sometimes?

Maybe a better format that allows finding RenderPassId->RenderPass mappings

Headless displays

Used only for consistent CopyOutputRequests. Looks like a real display.

Doesn’t require artificial references

Tries to piggyback off of other compatible display (software vs. GPU)

Questions?