Slimming Paint

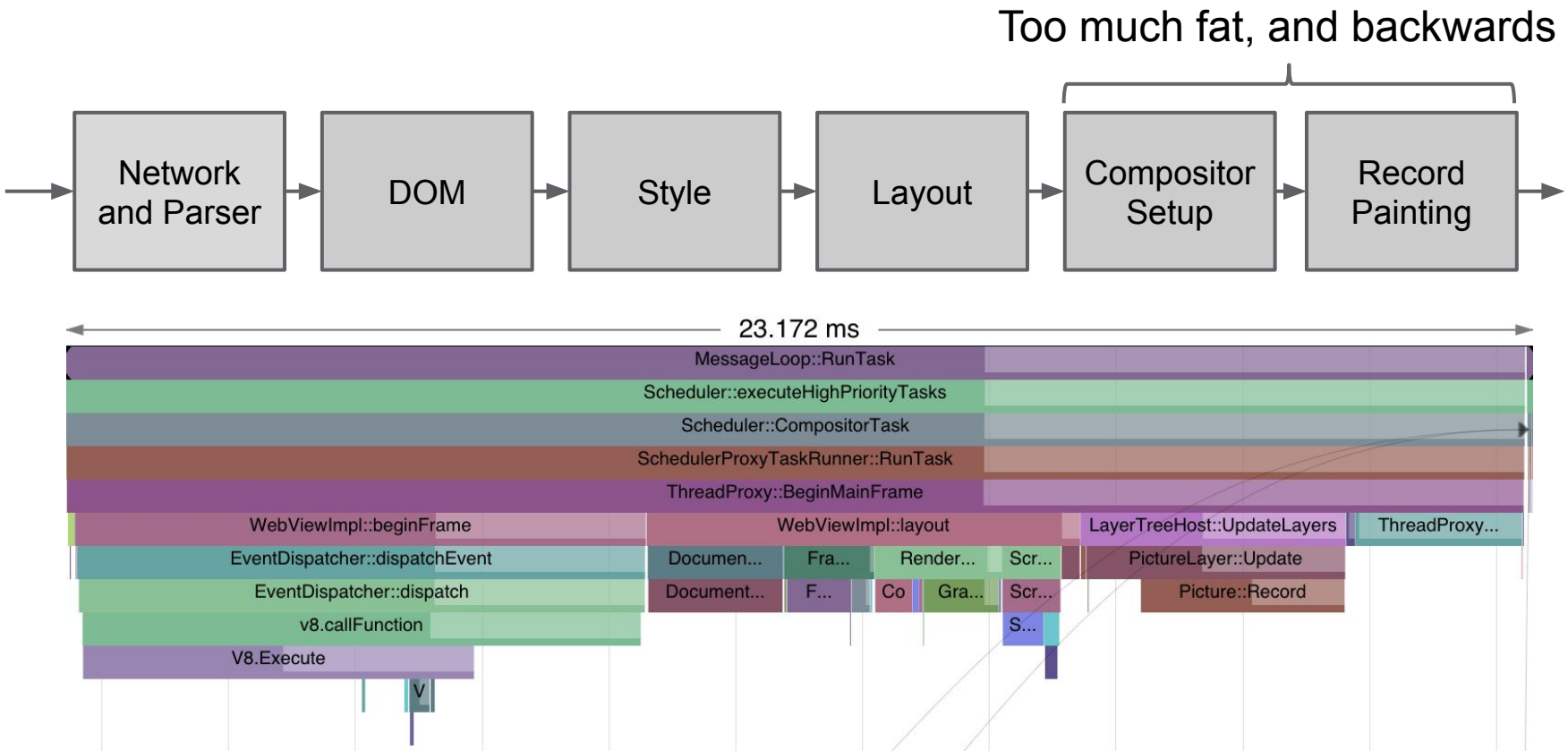
tl;dr

1. Cache recordings
2. Layerize in cc

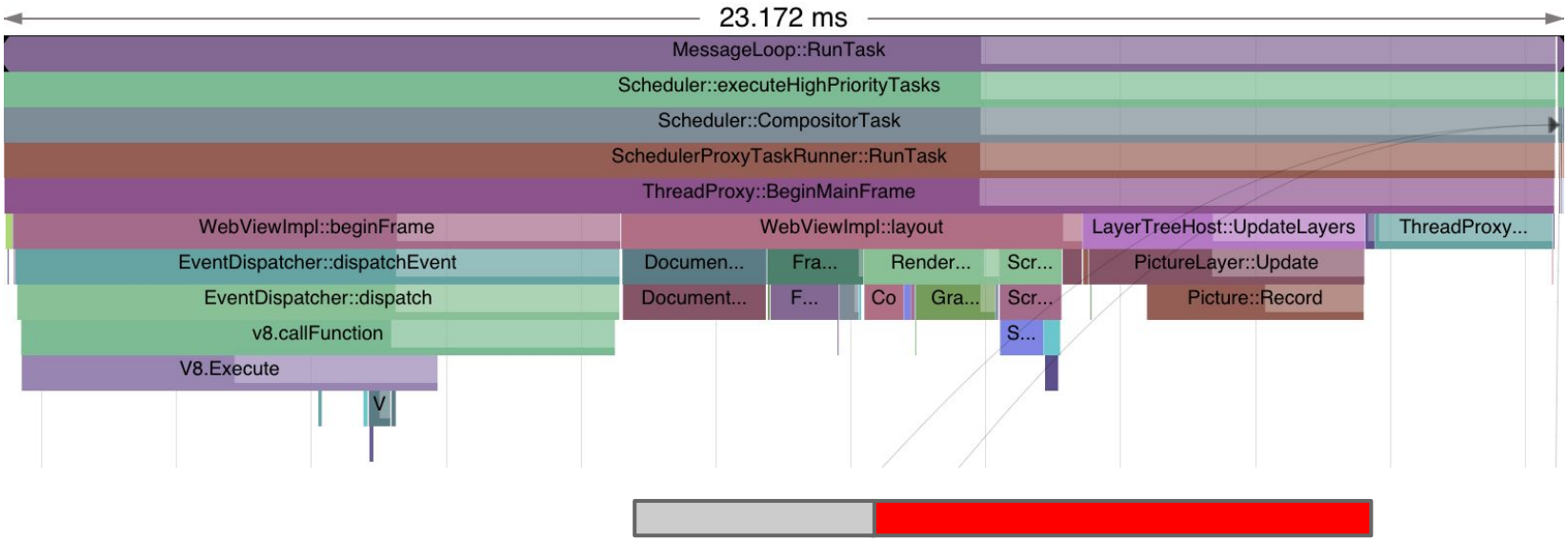
Why?

- Re-recording is expensive
- Blink shouldn’t do layers

From Content to Pixels



Painting is too expensive



Issuing draw calls should not take longer than style and layout!

Project Goals

- Faster recording/painting
- Correct and faster compositing
- Healthier, better tested code

Painting Bottlenecks

- Expensive layer management
- Blink paint code not optimized for our compositor architecture nor Skia
  - Inherited 2 code paths: CG/CA & Skia/cc
- We cache pixels, repeat work to define them

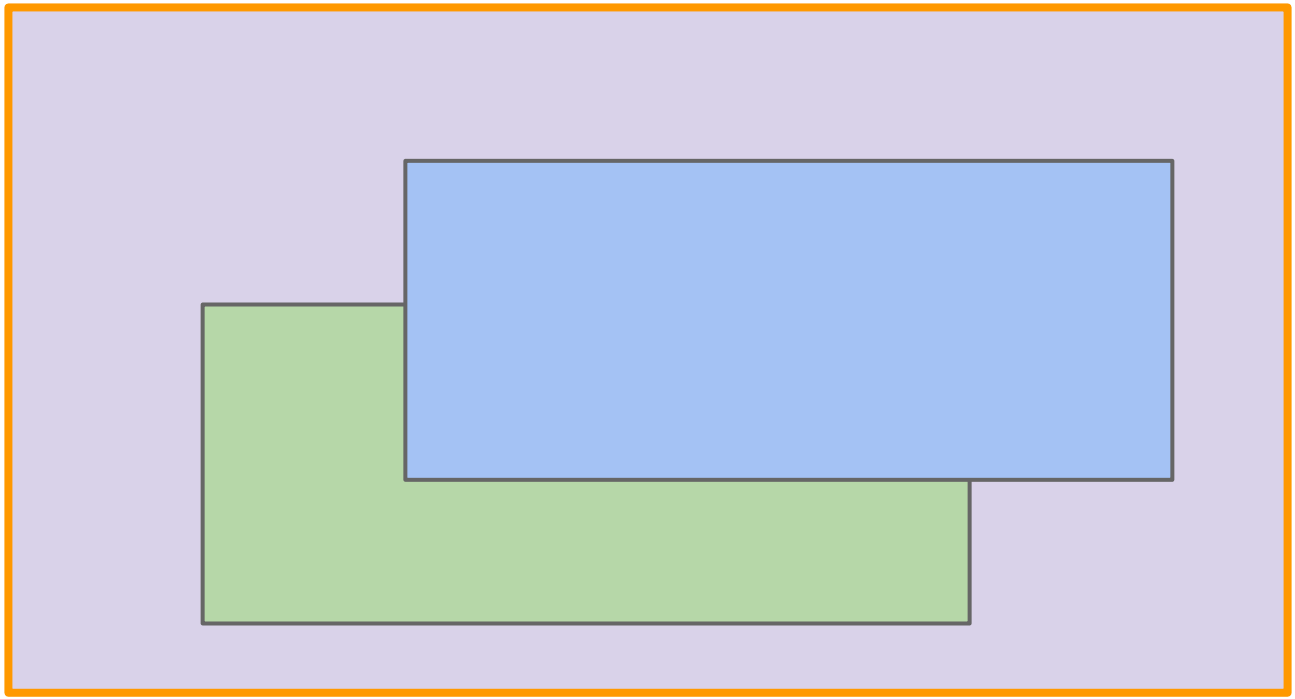
Potential for re-use

- On basic interactions, we repaint the same thing multiple times
  - Images, for example, do not change on about 80% of their calls
  - Image overlaps some changed rect, but it is not itself damaged

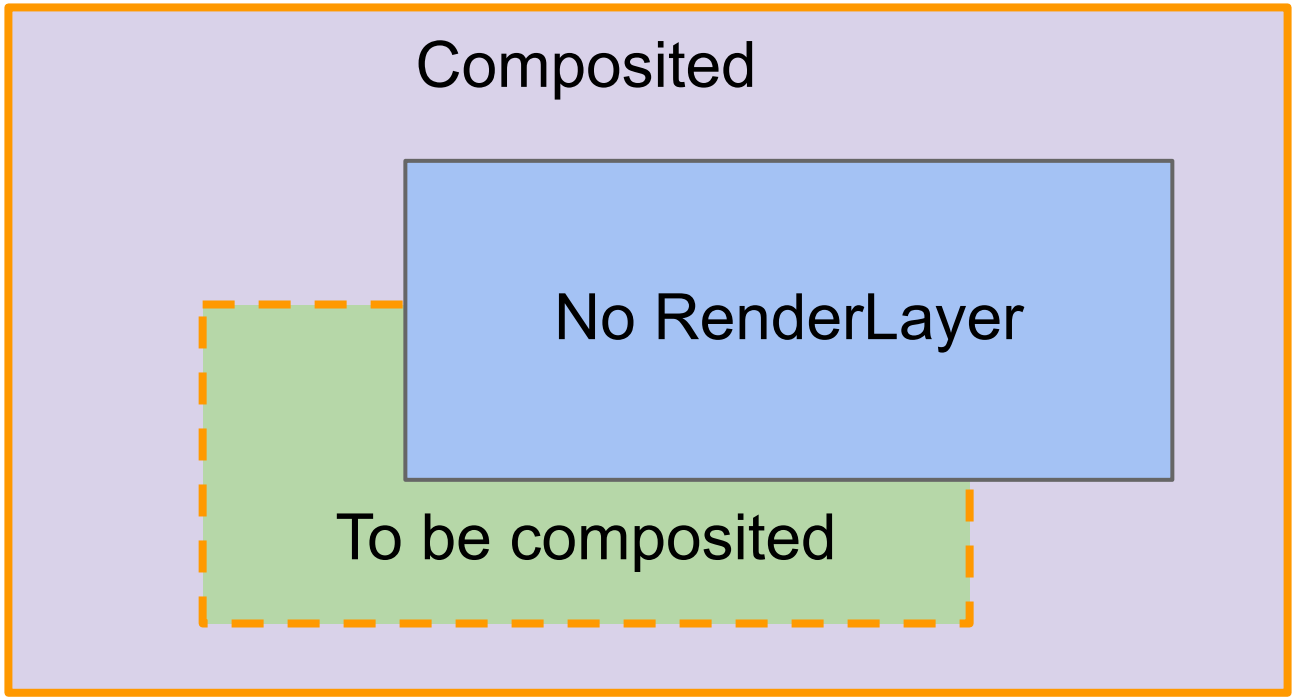
Compositing Limitations

- RenderLayer is the compositing “atom”
  - Causes “fundamental compositing bug”
  - Prevents layers in SVG
- Layer layering violation
- Squashing was forced to be a retrofit

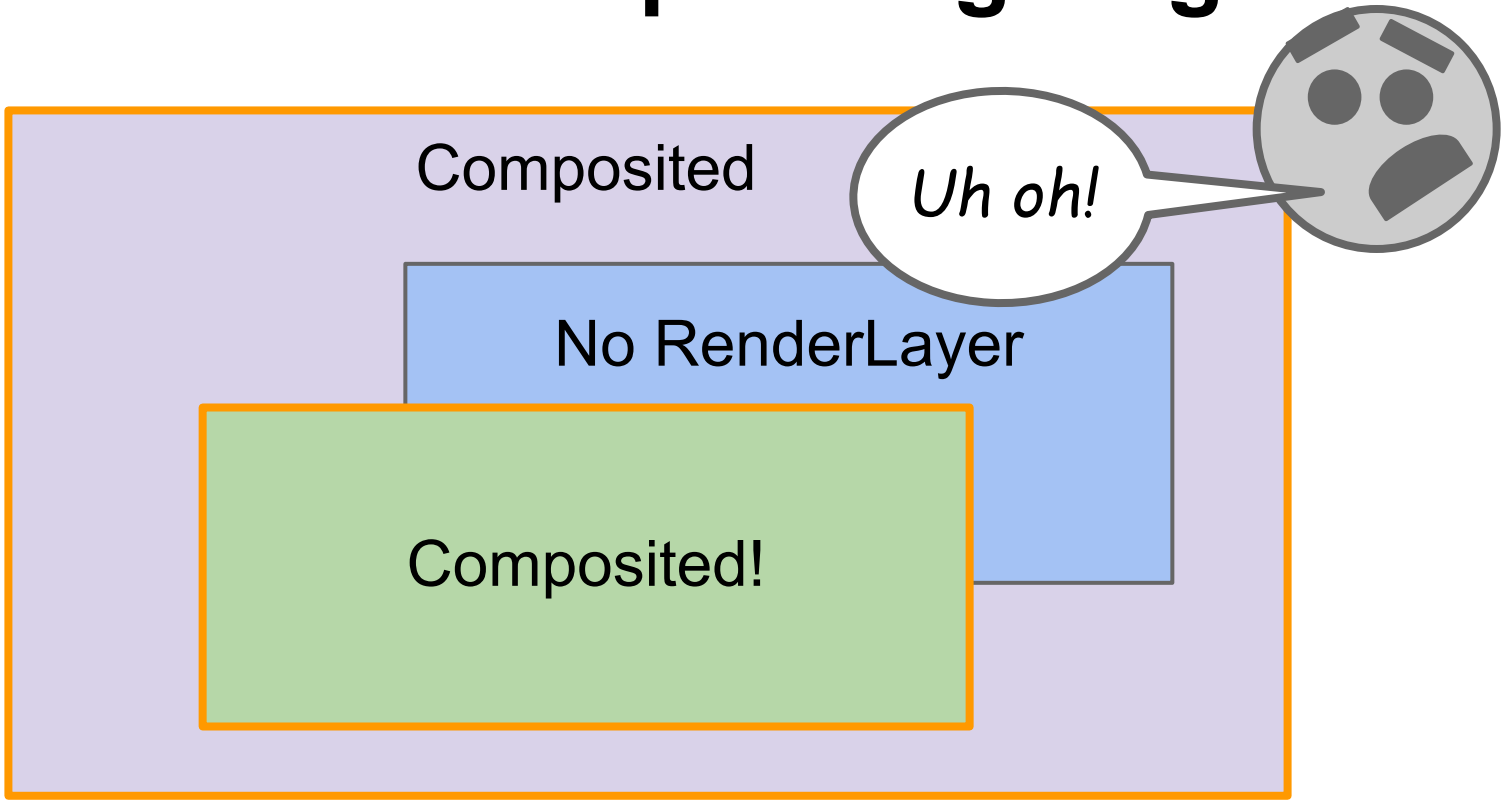
Fundamental Compositing Bug



Fundamental Compositing Bug



Fundamental Compositing Bug

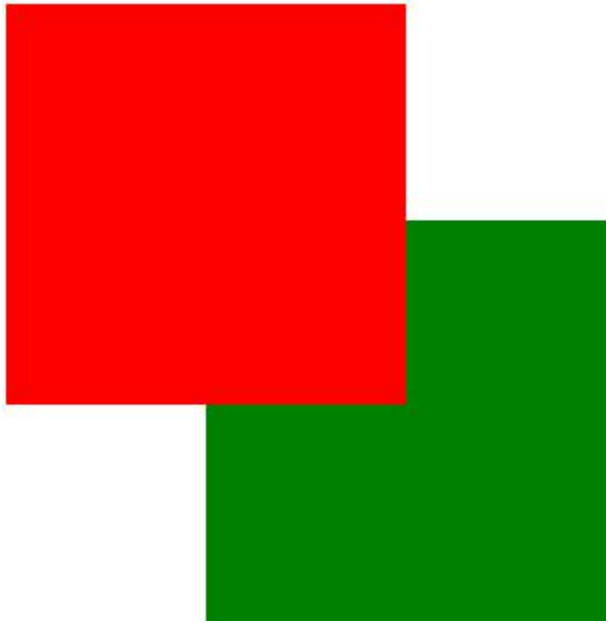


Example: Incorrect compositing

Courtesy of esprehn:

```
<canvas id="canvas"
        style="background:red;...">
</canvas>
<div id="overlap"
     style="background:green;margin-top:-50px;...">
</div>
```

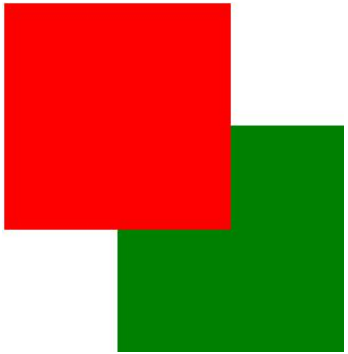
Example: Incorrect compositing



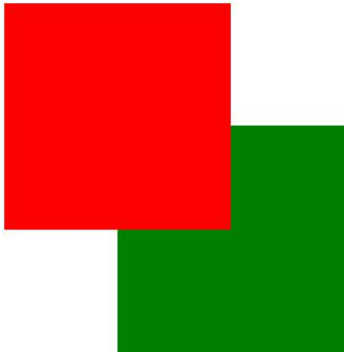
Example: Incorrect compositing

Why is red on top?

- canvas got own backing
  - Has its own RenderLayer
- div didn’t
  - Shares body RenderLayer



Example: Incorrect compositing



Why?

Since green had no RenderLayer,

- It didn’t participate in overlap detection, and
- Even if we wanted to promote it, we couldn’t!

Simplify Code

- Match code to logical units
  - CSS painting algorithm in Blink
  - Graphics layer logic in compositor
    - *Remove Source/core/rendering/compositing*
  - Better hit testing
- Rationalize paint phases
- Compositing on the right data structures

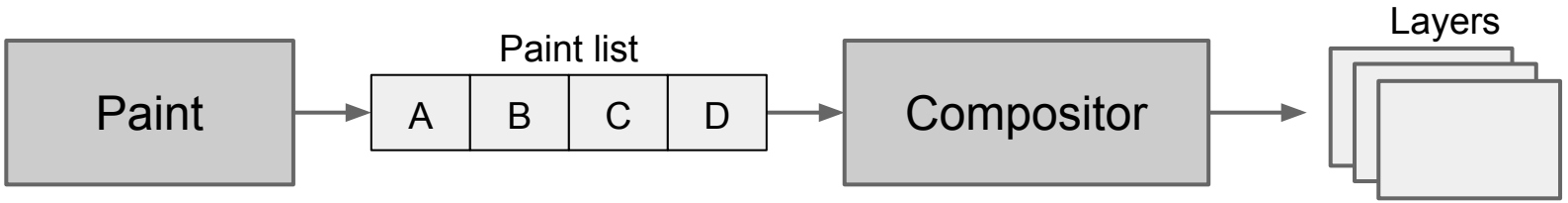
Getting to the goal ...

What Code is Affected?

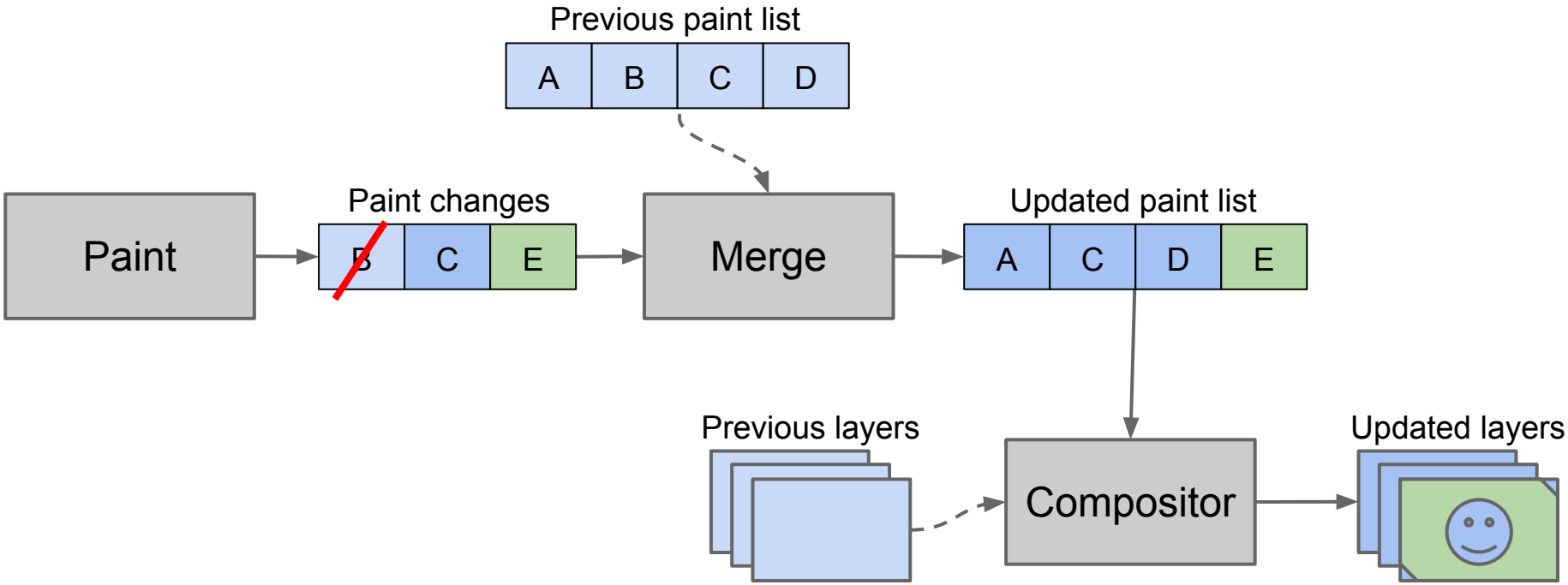
- Blink paint
  - Break painting into its own module
  - Painting converts render objects to display items
- Compositor
  - Entirely responsible for layering decisions
- Rasterizer
  - Better texture management, GPU targeting

Top Level Blink Design

- Blink paint produces a list of display items
- Compositor uses this to layerize and raster

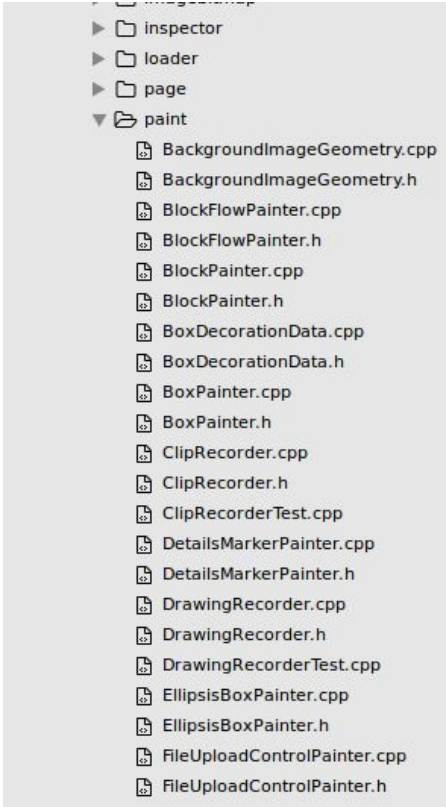


Top Level Design



Blink Changes: Painting refactor

- Painting code moved out of RenderObject, into Painters
- Code will be re-arranged for improved comprehension
  - e.g. Break out code paths per phase
  - e.g. Better map CSS paint spec onto code

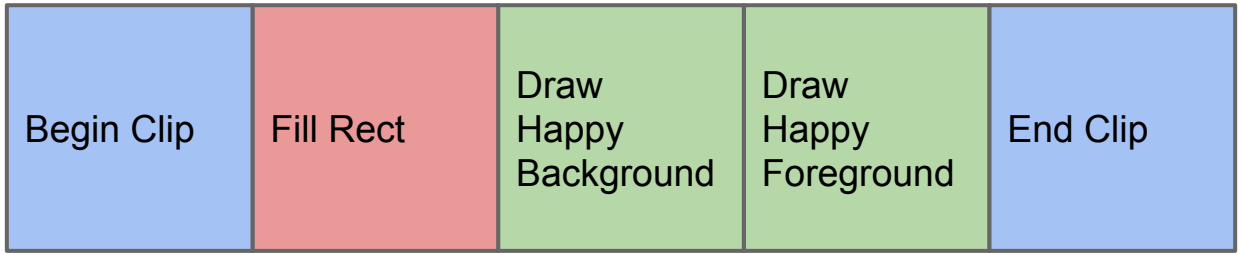


Blink Changes: Paint list

- All painting data inserted into a list of `DisplayItem`
  - Drawing commands, clips, transforms, stacking, …
- Only invalid renderers create new data
- Post-paint update merges new items with existing

Paint list example

```
<div style="clip: rect(...); bg-color: papayawhip;">
  <happy></happy>
</div>
```



Blink Changes: Future

- Remove `GraphicsLayer`, `CompositedLayerMapping`, etc
  - These structures are Blink's attempt to set up compositing, but the compositor knows more about compositing
- Remove `RenderLayer`
  - Reduce to hints in the paintlist.

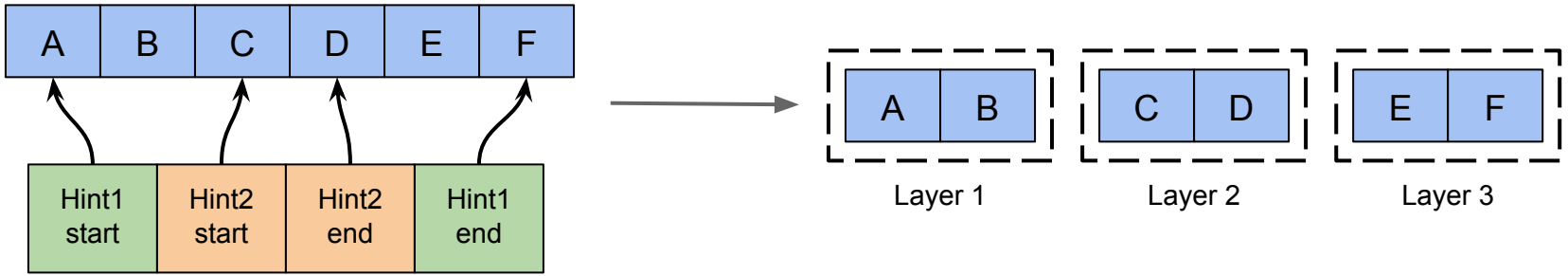
Compositor Change: Replace CDP

- CalcDrawProps walks the layer tree
  - Recursively computes visible rects, clips, screen-space transforms, screen-space opacity
- Compute from tiny property trees instead
  - Intuition: # of “significant” transforms, clips, or filters is << total # of layers

Compositor Change: Layerization

Layerization = clump draw ops in layers

- Historically: done in Blink with RenderLayers
- Now: done by the Compositor
  - using hints sent by Blink, knowledge of hardware



Compositor Changes: Future

Might move other consumers to display lists

Measuring Success

- Improve frame time
  - Particularly in dynamic situations
  - Composite more content, GPU more content
  - Telemetry tests
- Reduce bug count
  - Things we can’t fix now
  - Fewer bugs in the long run
- Increase developer productivity

Related Projects

- TextBlobs: Compute text paint information once, pass all the way to Skia
- GPU Rasterization (Ganesh): Draw all content with the GPU
  - Also reduce code, texture consumption, etc
- Various other perf projects

People

Blink

- Chris Harrelson (chrishtr)
  - pdr
  - leviw
  - schenney
  - trchen

Compositor

- Ali Juma (ajuma)
  - vollick
  - enne
  - weiliangc
  - awoloszyn

Following Along

- silk-dev@chromium.org mailing list for Silk Project development info
- slimming-paint-reviews@chromium.org for code reviews
- Slimming Paint Overview document
- Slimming Paint wiki (go/slimming-paint)