# Composited Recordings Requirements

Obsolete: This document has been merged into Slimming Paint

*Plugging paint right into the compositor*

*tl;dr The compositor should consume recordings, not layers.*

## Background

We are investigating major changes to the way that we record and integrate with the compositor. At a high level, the idea is that we'll teach the compositor to consume the output from Blink's render tree directly, sidestepping and simplifying a large portion of our current pipeline.

Specifically, the Blink code that manages layers (both Render- and Graphics-) will be entirely replaced, in part by a data container approximately equivalent to a scene graph, and in part by compositor code that processes the container to assign content to layers for compositing.

## Why?

Good question. Since the rationale for such a big project is so important, we've created a separate document to hash it out. The high level argument is that this refactor will fix a fundamental correctness issue, improve raster, record and compositing update performance and put us in a good place, architecturally, for the future.

## Prior Art

Blue Sky Compositing
Rethinking Blink Painting

## Requirements

### General Requirements

- Compositing decisions such as texturization may never affect web compatibility, only performance.
- Performance must be equal to or better than the existing system at each shipped phase, or known to be only marginally and temporarily slower.

### Blink's Requirements

- Must be possible to locally record. e.g. mutate a recording or replace some portion of the entire recording. The most effective way we have to improve painting performance is to invoke painting methods on fewer objects.
- Must be able to determine what to record in a region of interest. In response to an invalidation of a composited layer due to a content change, we currently paint every contributing renderer that intersects the invalidation. If we can make content updates (previous bullet), we should be able to re-record just the dirty properties that involve the region of interest. Note, to permit this Blink will need:

    - Efficient ways to compute the region of visual effect for any style or layout change (including semantic invalidations)
    - A method to plumb window system dirty rects back to Blink
    - Viewporting (project has been started; still in early days).
    - Addition of a spatial data structure (part of the viewporting project).
- Must produce artifacts that Skia and cc can consume without overhead. For example, culling information should be computed once in the form Skia needs.
- Must reduce the total cost of painting for key benchmarks
- Must simplify or remove layer related code
- TODO: more here!

### Compositor Requirements

- Must be able to efficiently group subrecordings into textures.
- Must be able to efficiently answer geometry queries about what parts of recordings intersect with a given rectangle.
- Must be able to efficiently update the data structure on the thread.
- Cannot regress raster or record times.
- Cannot increase video memory requirements when texturizing.
- Must contain enough information to implement compositor driven effects (animation, scrolling)
- Must be possible to associate the recording with multiple hierarchies (e.g., transform, clip, opacity, scroll).
- Recording must be efficient to traverse. It is usually read from many more times than written, so it can't, eg, be pointer-happy.
- TODO: more here!

### Raster Requirements

- Unify spatial data structures and region related code
- Must be able to reorder draw operations
- Must not enable fully mutable pictures or otherwise hobble efficient recording and playback

### Out of scope: requirements for another time

- Ability to record at the *semantic* level rather than the draw op level.