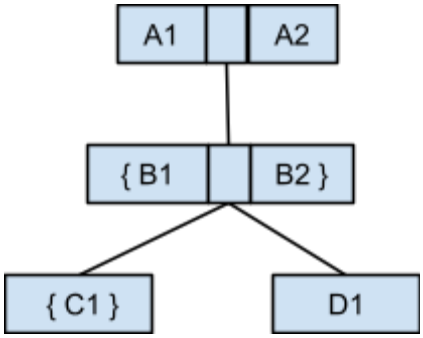
Pitfalls in DisplayList Creation for Slimming Paint

Consider a call graph where painting code at multiple stack frames each creates a PaintCommandRecorder to create paint chunks containing DisplayLists. There are several aspect of DisplayLists and the SkPictures within them that must be managed correctly to get the right sequence of commands.

I'll refer to this call graph throughout.

- Method A instantiates a PaintCommandRecorder, makes some GraphicsContext drawing calls (A1), then invokes B before drawing some more, A2.
- Method B pushes a clip (or transform, it doesn't matter which), makes drawing calls B1, invokes C and D, draws B2 and pops the clip. Push and pop in this sense refer to save/restore on the graphics context, which in turn invokes save/restore on the Skia canvas.
- Method C saves a transform, draws C1 and then restores.
- Method D just draws D1



Nesting

When A invokes B, the expected draw order is A1, whatever B does, then A2. Indeed, if a DisplayList records for the entirety of A, and the entirety of B, the result will be A's DisplayList containing a reference to B's DisplayList at the appropriate point in the drawing sequence.

The only correct way to replay this is to invoke A's DisplayList only, which internally invokes B's display list at the right time.

Paint chunks assume that they will be replayed in a linear ordering, with each paint chunk replaying its DisplayList. So when we generate paint chunks we must instead generate at least 3 chunks, one for A1, one for B and its contents, and one for A2.

The simplest fix for the current problem is to have paint chunks be emitted every time a new recording starts. Note that having a DisplayList for A, and one for B, where A's does not contain B's, does not solve our problem because we cannot in that case play B's partway within A's.

Clips and Transforms

SkPicture, and hence DisplayList, enforces a requirement that save and restores be balanced within the picture. If not, playback will insert or ignore the appropriate number of restores (I hope it also throws an error).

Consider B in detail now. It clips, draws, invokes C and D, then draws and restores the clip. We cannot include the clip in a DisplayList for B1 because it will be restored automatically before drawing C and D, removing the clip on those nodes. The restore in B2 will also generate an error. Note this problem does not occur for C because C can be recorded with a single DisplayList.

The DisplayList class contains an explicit clip and transform intended to deal with this problem, but it is insufficient in the current architecture. When originally created, it was assumed that DisplayList would be extended or subclassed contain the children DisplayList and could appropriate manage the clips and transforms. In other words, we would generate a paint tree rather than a paint list.

In the paint list architecture, we need explicit paint chunks for clips and transforms that are intended to bridge multiple paint chunks.

Short Term Mitigation

For an initial round we can be conservative and assume that whenever a paint method invokes another paint method that creates a paint chunk we must create the sub-lists, clip chunks and transform chunks described above. A method that invokes no other paint methods can naively emit a single display list.

Implicit in this is the assumption that we can track which methods create chunks and whether any given call site will ever lead to another method that generates chunks. I think we have to assume that all methods create chunks.

Medium Term Mitigation

A more complex system would be lazy about creating new DisplayLists. For example, initially we could generate a single DisplayList for A and all it's contents, only breaking it up if we encounter a situation in which we must create another DisplayList (e.g. for compositing reasons).

This would not be intractable to implement from the recording perspective. One hard part is maintaining the paint chunks such that A didn't have to be concerned about how it's paint chunks are being manipulated, or how many there are or what its child methods are doing. It argues for making the PaintCommandRecorder an intelligent thing that keeps track of active clips and transforms and layering reasons so that it can retroactively generate the right set of lists after paint commands have already been issued. This in turn would require SkPicture introspection (which is frowned upon by the Skia team - just ask the compositor team). Or we would need to create our own command buffer (at which point, do we want to back by Skia at all?).

Long Term Mitigation

Start with medium term mitigation. As the page dynamically evolved and paint became invalid, we could insert the necessary DisplayLists and re-record or otherwise break up A. This would become messy, but has the advantage of finding the "right" degree of paint command discretization depending the page actions.