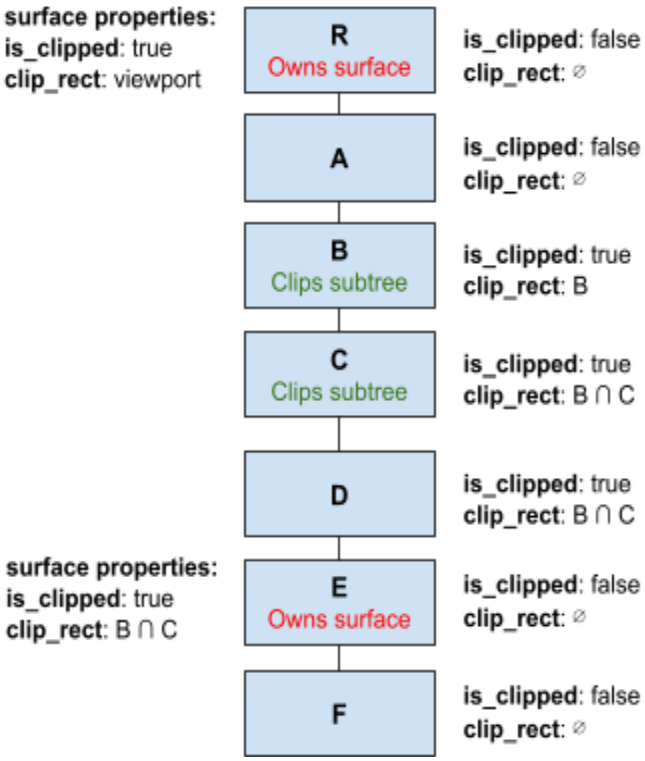
Draw-time clipping

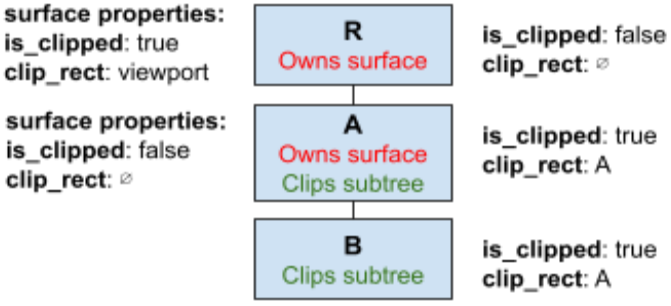
clip_rect and *is_clipped*

The **clip_rect** and **is_clipped** draw properties are used to determine the scissor rect (if any) used when drawing each quad. Changing the scissor rect can cause an expensive GPU pipeline flush (see GLRenderer::SetScissorTestRect), so the goal is to minimize the number of scissor rect changes while preserving correct rendering. In particular, this means that preserving the existing scissor rect is preferable to changing it to a tighter rect, as long as this doesn't break correctness. Similarly, implicitly clipping a render surface by taking a clip rect into account when computing the surface's size is preferable to setting a scissor rect when drawing the quads of each individual layer that contributes to the surface, as long as there aren't any contributing layers that escape this clip (that is, as long as no contributing layer has a ClipParent).

Some examples

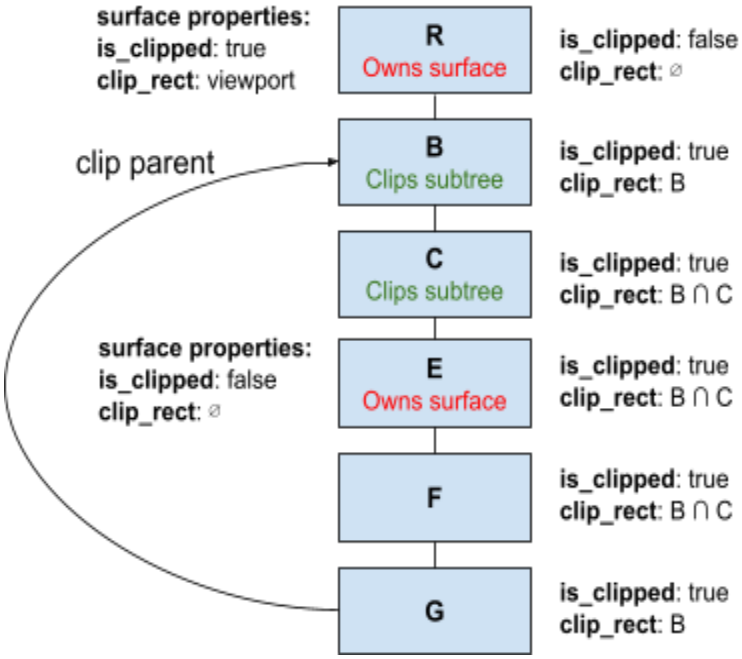


At each layer that clips it subtree, the layer bounds are intersected into the current clip_rect. However, once we reach the surface owned by E, the clip_rect gets cleared; the surface itself is drawn with this clip_rect, so layers contributing to the surface don't need to be clipped when drawn. Layer D could be drawn with a tighter clip_rect (B ∩ C ∩ D), but this isn't done since it would introduce an unnecessary clip_rect change between D and C.



Layer A owns a surface and also clips its subtree. This clip cannot be applied when drawing the surface itself, since the surface might need to grow beyond the clip. For example, if the surface is used to apply a blur filter, the output of the blur will grow beyond the bounds of A even though

its input is clipped by by the bounds of A. So the clip is applied when drawing each layer that contributes to the surface.



In this example, layer G has clip parent B, so escapes the clip established by C. This means that surface E cannot be drawn with clip B ∩ C (since that would incorrectly clip G). Instead, the layers that contribute to this surface need to be clipped individually.

Property tree implementation

The **layers_are_clipped** and **target_is_clipped** values of each ClipTreeNode are used to determine the is_clipped values for layers and surfaces. Clipped layers use the **clip_in_target_space** of their ClipTreeNode as their clip_rect. Clipped surfaces use the **clip_in_target_space** of their ClipTreeNode's parent node as their clip rect.

Algorithm to determine draw content rects

Property tree state = { transform node, effect node, clip node, scroll node }

Input: the four property trees, plus a notation of which effect tree nodes induce a render surface
Output: draw_clip_rects to apply, if any before drawing each layer in order within its render surface

1. For each render surface, recursively from the root render surface:
    a. render_surface_clip_node = clip node for the property tree state of the render surface
    b. For each layer contributing to that surface:
        i. render_surface_clip_node = least_common_ancestor(render_surface_clip_node, clip node for layer)
    c. After recursing into child render surfaces:
        i. For each child render surface:
            1. render_surface_clip_node = least_common_ancestor(render_surface_clip_node, render_surface_clip_node for child)
2. For each render surface, recursively from the root render surface:
    a. For each layer contributing to that surface:
        i. compute combined clip rect from clip nodes between this layer's clip node and render_surface_clip_node
        ii. Transform this combined clip layer down into the property tree space of the render surface. Record the result as clip_rect_in_render_surface_space for this layer.
        iii. Intersect it with the layer's bounds mapped into the property tree state of the render surface
        iv. Record this rect as the draw_content_rect of the layer
3. For each render surface, compute the union of the bounds computed in step 2, and record the result as the draw_content_rect of the render surface.

This is a linear time algorithm.

Algorithm to optimize out unnecessary clips

Input: output of algorithm above.
Output: scissor rects if necessary when sequentially painting layers.

scissor_rect = nullptr
For each render surface/layer in draw order:
1. Set the scissor_rect for this layer to current_clip_rect for the current layer if clip_rect_in_render_surface_space does not contain the current scissor_rect
2. (Otherwise retain the existing scissor_rect)

This is a linear time algorithm.