Summary of compositing inputs in Blink

Ancestor-dependent compositing inputs

Existing use-cases

1. Opacity ancestor (ok to go)
    a. Used to decide whether layers can't squash **[SPv1-specific]**
    b. Used to determine whether a layer is opaque for possible promotion of fixed-position elements. **[SPv1-specific]**
    c. Used to determine opaqueness for LCD text in promoting fixed-position objects
2. Transform ancestor (ok to go)
    a. Used to decide whether layers can't squash **[SPv1-specific]**
    b. Used to position squashed layers **[SPv1-specific]**
    c. Used to help determine promotion of possible fixed-position elements
    d. Used to determine opaqueness for LCD text in promoted scrolling
3. Filter ancestor (ok to go)
    a. Used to decide whether layers can't squash **[SPv1-specific]**
4. Ancestor scrolling layer (ok to go)
    a. Promote elements for whom their containing scroller is not in the stacking context chain **[SPv1-specific]**
    b. Turn off overlap testing inside of composited scrollers **[SPv1-specific]**
    c. Figure out if an element is scrolling w.r.t the viewport when scrolling it onscreen via web apis
    d. Avoid promoting fixed-position elements that are fixed w.r.t. a transformed `element but not the view

    e. Determine whether a “slow repaint” region for scrolling of a composited layer needs to include a particular element (i.e. if it scrolls) **[SPv1-specific]**
    f. Used to decide whether layers can’t squash (don’t scroll together) **[SPv1-specific]**
    g. Apply assumed-overlap compositing trigger to descendant PaintLayers which scroll w.r.t. a composited ancestor **[SPv1-specific]**
5. Ancestor fixed position layer (ok to go)
    a. Used to decide whether layers can’t squash **[SPv1-specific]**
6. Scroll parent (ok to go)
    a. Used to decide whether layers can’t squash **[SPv1-specific]**
    b. Detect if the top-most child of a scroller changed (and therefore composited layers for some reason need rebuilding) **[SPv1-specific]**
    c. Composite layers which have a composited scrolling ancestor which is not a stacking ancestor **[SPv1-specific]**
    d. Find scrolling parent to check if the scroll controls need to be reparented in the graphics layer tree **[SPv1-specific]**
7. Clip parent (ok to go)
    a. Sets clip parent of graphics layers which have a non-stacking clip ancestor **[SPv1-specific]**
    b. Detect out-of-flow (non-stacking) clipping as a direct compositing reason **[SPv1-specific]**
8. Clipping container (ok to go)
    a. Determine whether a graphics layer is clipped by a non-stacking ancestor **[SPv1-specific]**
    b. Determine the local clip rect for squashing layers **[SPv1-specific]**
    c. Used to decide whether layers can’t squash **[SPv1-specific]**
9. Clipped absolute bounding box (ok to go)
    a. Bounds for determining overlap in overlap testing for compositing
    b. Bounds of a squashing layer. Used to determine if a squashing layer has to be split because it’s too sparse **[SPv1-specific]**
10. Unclipped absolute bounding box (ok to go)
    a. Bounds for determining overlap of elements with a composited scroller **[SPv1-specific]**
11. Has ancestor with clip path (ok to go)
    a. Used to avoid composited scrolling on a low-DPI screen if there is an ancestor clip path **[SPv1-specific?]**

New implementations

Store required ancestor-dependent flags on either pre-paint tree walk context (if not needed outside of the walk), or on ObjectPaintProperties. Record direct and indirect compositing reasons in ObjectPaintProperties (*not* PaintLayerRareData like today)

Descendant-dependent properties

Existing use-cases

1. hasDescendantWithClipPath
    a. Used to avoid composited scrolling on a low-DPI screen if there is an ancestor clip path **[SPv1-specific?]**
2. hasNonIsolatedDescendantWithBlendMode
    a. One of the triggers of compositing with blend-mode (has to happen at the direct enclosing ancestor stacking node of the blend-mode object).
    b. Indicates whether a PaintLayer has to us a “compositing” save layer to properly mix colors with the blend-mode child.
3. hasRootScrollerAsDescendant
    a. Used to not set masking to bounds of a scrolling layer if it has a root scroller descendant
    b. Used to not clip composited children if it has a root scroller descendant
4. hasVisibleDescendant
    a. Determining if the contents of a composited layer are visible, as an optimization to avoid drawing contents if false
    b. Return an empty rect for the bounding box for compositing if there are no visible descendants
    c. Skipping invisible subtrees when invalidating viewport-constrained objects
    d. Skipping composited overflow controls for invisible scrolling contents
    e. Skips setting clip parents for invisible subtrees
    f. Skips squashing for invisible subtrees **[SPv1-specific]**
    g. Used to skip returning true for “can be composited” (unless there is a compositor animation)
    h. Skip main-thread scrolling for invisible subtrees
5. hasSelfPaintingLayerDescendant
    a. Early-out from hit testing recursion for PaintLayers if there is no self-painting descendant
    b. Early-out from paint recursion for PaintLayers if there is no self-painting descendant

New implementations

2a: Add an effect node for blending at the stacking context ancestor during paint property tree building. Use that node to apply blending. Compute this bit during updateLayerPositionsAfterLayout()

3a, 3b. There will have to be special logic for root scrollers? Or maybe root-layer-scrolls will solve this.

Compositing triggers

PaintLayerCompositor::updateDirectCompositingReasons appears to be dead code

* Every frame has a CompositingReasonFinder, which stores state bits for the CompositingTriggerFlags on the containing frame.

* On style update, a PaintLayer calls CompositingReasonFinder::potentialCompositingReasonsFromStyle, by using CompositingReasonFinder,
and stores it on itself.

* CompositingRequirementsUpdater reads CompositingReasonFinder::directReasons and uses it to force compositing.

* In addition, if during the recursion it determines that there are composited descendants, then the triggers in CompositingReasonComboCompositedDescendants
force compositing. If it determines there are 3D-transformed descendants, then the triggers in CompositingReasonCombo3DDescendants force compositing.

Direct reasons

CompositingReason3DTransform | CompositingReasonVideo |
CompositingReasonCanvas | CompositingReasonPlugin |
CompositingReasonIFrame | CompositingReasonBackfaceVisibilityHidden |
CompositingReasonActiveAnimation | CompositingReasonTransitionProperty |
CompositingReasonScrollDependentPosition |
CompositingReasonOverflowScrollingTouch |
CompositingReasonOverflowScrollingParent |
CompositingReasonOutOfFlowClipping | CompositingReasonVideoOverlay |
CompositingReasonWillChangeCompositingHint |
CompositingReasonCompositorProxy | CompositingReasonBackdropFilter;

Potential reasons from style not listed above

```
    CompositingReasonInlineTransform | CompositingReasonComboCompositedDescendants |
CompositingReasonCombo3DDescendants

    CompositingReasonComboCompositedDescendants =
        CompositingReasonTransformWithCompositedDescendants |
    CompositingReasonIsolateCompositedDescendants |
    CompositingReasonOpacityWithCompositedDescendants |
    CompositingReasonMaskWithCompositedDescendants |
    CompositingReasonFilterWithCompositedDescendants |
    CompositingReasonBlendingWithCompositedDescendants |
    CompositingReasonReflectionWithCompositedDescendants |
    CompositingReasonClipsCompositingDescendants |
    CompositingReasonPositionFixedWithCompositedDescendants;

    CompositingReasonCombo3DDescendants =
        CompositingReasonPreserve3DWith3DDescendants |
    CompositingReasonPerspectiveWith3DDescendants;
```