BlinkGenPropertyTrees perf analysis

BlinkGenPropertyTrees (BGPT) is a project to pass property trees and a layer list from blink to cc. Before BGPT, a layer tree was passed from blink to cc, and cc computed property trees from this layer tree. BGPT is a staging towards CompositeAfterPaint and is not intended to be a performance win on its own.

Finch trial

The finch trial is setup to run for M75+ and will run at 10% enabled / 10% disabled / 80% default (enabled) for canary / dev / beta, and 1% enabled / 1% disabled / 98% default (enabled) for stable. The finch trial is setup to run through June 7th. The finch trial details can be found on BlinkGenPropertyTrees.gcl. When looking at finch results, it is important to use the "*_20190422" study group (i.e., Control_20190422, Enabled_20190422).

UMA (canary, M76, 1 day):
https://uma.googleplex.com/variations?sid=aafac3fed416458fc1af4a39551b340c

Timeline view (canary, M76, 1 day):
https://uma.googleplex.com/p/chrome/timeline_v2/?sid=93b2df0d7ad0ded909d82f3d6dee268e

UMA (canary+dev, M75+M76, 7 day):
https://uma.googleplex.com/variations?sid=63c4f6dce329ce6fc7ba36a22ddd4ca8

UMA (beta, M75, 7 day):
https://uma.googleplex.com/variations?sid=d709f78ff46fc422f59ac32d196d50c4

Performance bots

Perf regression bugs:

- crbug.com/926327 - General BGPT regression bug for *CPU*
  Perf bot tracking spreadsheet:
  https://docs.google.com/spreadsheets/d/1lGdFw4mMGfSewfQK448ZdjVkgtcjuepEi0mUylfe5yU/edit#gid=0
    - crbug.com/954961 - Fix performance of infinite_scroll_root_n_layers_99 and infinite_scroll_root_fixed_n_layers_99
    - crbug.com/954520 - Investigate regression in thread_total_all_cpu_time_per_frame/js_poster_circle
    - crbug.com/954493 - Investigate regression in avg_surface_fps/css_transitions_triggered_style_element

    - crbug.com/941031 - regression in rendering.mobile/thread_raster_cpu_time_per_frame
- crbug.com/947050 - BlinkGenPropertyTrees causes more render surfaces
- crbug.com/900241 - Additional nodes are created for animations.

Fundamental performance differences

BlinkGenPropertyTrees is known to have some progressions:

- Because cc::Layers no longer store scroll-offset-relative values, compositing updates can be skipped for some scroll offset changes (see: PaintLayerScrollableArea::UpdateCompositingLayersAfterScroll). This can be seen on perf bots as a ~30% drop in thread_total_all_cpu_time_per_frame / infinite_scroll_element_n_layers_99.
- By more closely tracking dirty state (PaintArtifactCompositor::SetNeedsUpdate), full updates can be reduced. This causes roughly 15% fewer calls to LayerTreeHost::UpdateLayers (see drop in Compositing.Renderer.LayersUpdateTime UMA count).
- By removing non-drawable cc::Layers that were used for positioning, roughly 30% fewer cc::Layers exist (see drop in Compositing.Renderer.NumActiveLayers UMA count).
- By building the complete property trees in blink, the cc property tree builder no longer runs (see drop in Compositing.Renderer.LayersUpdateTime UMA count).

BlinkGenPropertyTrees is known to have some regressions:

- To make more granular layerization decisions for CompositeAfterPaint, more property tree nodes are created. For example, separate transform nodes are created for scroll offset and css transform, whereas these were combined into a single transform node in the past. After BlinkGenPropertyTrees launches, the cc property trees will be simplified so the cost is lowered.
- Additional render surfaces are created for rounded corners (see: https://crbug.com/947715). This is addressed in a followup in M76 with FastBorderRadius which uses a shader for drawing rounded corner masks.