Detecting animation updates crbug.com/1111392

Authors: flackr@

Problem

For a general web metric, we need a standardizable way to classify whether a given frame update factors into smoothness. From a standardization point of view, it doesn't matter whether this update is on the compositor or main thread, just whether it should be considered important. However, from an implementation point of view, we are currently considering all compositor thread updates important.

Examples of animation updates include:

- Animations (e.g. CSS animations)
- Videos
- Javascript updating a CSS property of an element regularly
- Scrolling

Examples of non-animation updates:

- Page loading
- Clicking on a button takes a long time to produce view in response (this will be classified under a separate metric)
- Button appearance responding to mousedown.
- Background loading / insertion of new content

Goals

It is likely not be possible to perfectly solve this problem, however the solution should:

- Detect the majority of existing use cases
- Be difficult to animate without it being detected (i.e. obscuring slow animations from reporting)
- Be possible to specify in a standard

As such, it's expected that there will be false positives however we expect these to be infrequent compared to true positives.

Proposal

Consider a main frame update important if it:

1. has a CSS/web animation update not reflected on the compositor (i.e. main thread animation) or
2. has an active scroll which is blocked on the main thread or
3. has a playing video which does not support accelerated rendering or
4. has a canvas invalidation on a canvas element which **was visible** prior to the current frame or
5. has a **direct style** update of an **animatable property** (see Animating properties) on an element which **was visible** prior to the current frame.

The last condition is intended to detect cases where the developer is running their own animations such as:

- A pointermove handler which updates the transform of an element to match the pointer.
- A requestAnimationFrame updating element properties every frame to implement a custom animation.
- An onscroll handler updating the background position of a parallax background.

Note that this will include false positives such as single frame direct style updates, but these are expected to be far less frequent than animating cases.

Why direct style mutations?

- Note this includes element.style = '...' or element.style.property = '...'.
- Without great difficulty (e.g. creating 100s of class names), this is the only way to smoothly animate a property.

Why animatable properties?

- Matches properties which could be animated by CSS. Updating the display or position property every frame does not produce a smooth animation.

Why not exclude discrete properties?

- Properties which are discretely animated in CSS may be smoothly animated in javascript (e.g. developer could polyfill gradient interpolation before css-images-4 defined a method for this).

Why elements which were visible?

- We already track whether elements were previously visibility: visible and this combines the box having been rendered and having been visible to the developer.
- It is common for developers to set inline style on newly added elements when constructing views. These should not be considered animations.

Implementation

We have an existing mechanism to count main thread animations, so the two new cases are canvas invalidations and style updates.

Canvas invalidations

Instrument CanvasRenderingContext::DidDraw?

Inline style mutations

Instrument Element::SetInlineStyleFromString and Element::SetInlineStyleProperty?
Checking if Element has a box? Call GetLayoutObject().

Evaluation Metrics

We should measure the relative proportions of the following:

1. Total frames
2. Number of animating frames (by type)
3. Potential false positives - Number of direct style updates frames which were not back to back (i.e. incremented when we no longer detect animation after a single animating frame).
4. Potential false negatives - Number of indirect style updates?

This should allow us to calculate an upper bound for the false negative rate and false positive rate to determine whether we should attempt to further refine the metric and in which direction.

Alternatives considered

Looking for N of M frame / node updates
Pros

- Doesn't include single frame updates

Cons

- Easy to cheat (e.g. update less frequently than the chosen rate)
- Could miss rate limited updates if N/M is too large (e.g. low fps animation, events not showing up often enough)

Every main frame update is important
Pros

- Very difficult to cheat

Cons

- Includes loading and long response updates making high percentiles less valuable.
- Includes progressive and asynchronous loading (e.g. images, infinite scroller) which do not affect user experience during smooth scrolls