BeginFrame sequence numbers + acknowledgements

PUBLIC
bit.ly/beginframeacks
Authors: eseckler@
June 9th, 2017
Tracking bug: https://crbug.com/697086

Motivation

Chromium's display compositor currently relies on a heuristic to determine which surfaces are *active* and thus likely to be damaged during a vsync (`BeginFrame`). As soon as all these active surfaces have been damaged, the `DisplayScheduler` will trigger the completion of the `BeginFrame` by triggering its deadline early. This is to provide as much time as possible for the drawing of the damages onto screen.

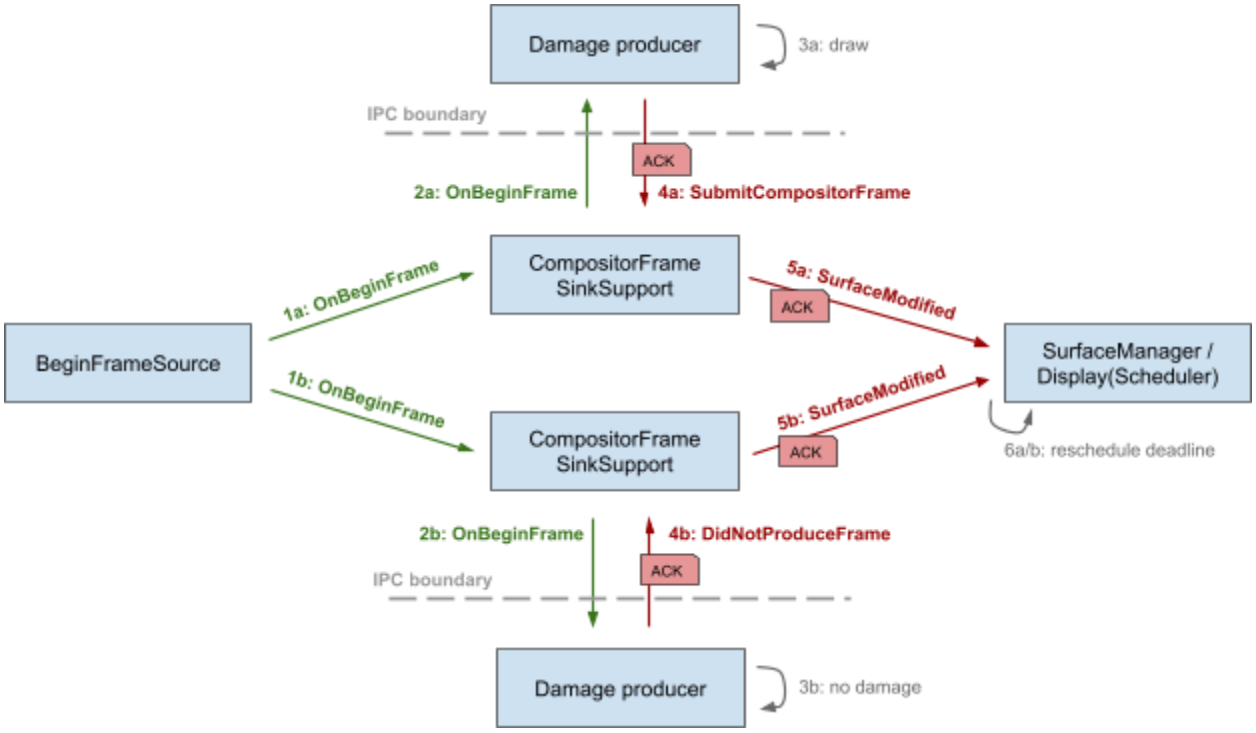
The current heuristic simply considers all surfaces that were damaged during the previous `BeginFrame` to be likely to be damaged in the following one. This is inaccurate and can lead to the `DisplayScheduler` triggering the deadline too early or too late, which incurs a higher latency of changes reaching the screen. In some cases, it can even cut the frame rate in half.

Further, Headless Chrome is looking to control the issuing of BeginFrames via DevTools and requires a way to reliably track when an individual BeginFrame has been completed by all damage producers.

Approach

We will introduce an acknowledgement mechanism (`BeginFrameAcks`) for `BeginFrames`, which is used to signal to the compositor that a damage producer has completed a `BeginFrame`. A damage producer will bundle its `BeginFrameAck` with a `CompositorFrame` submitted in response to a BeginFrame. If there was no need to produce a new `CompositorFrame`, it will send an individual `BeginFrameAck` through the same layers instead.

The `DisplayScheduler` can later use these `BeginFrameAcks` to replace its heuristics for active surfaces by triggering an early `BeginFrame` deadline as soon as all relevant damage producers have acknowledged the current `BeginFrame`.



Implementation

BeginFrame sequence numbers

To track and acknowledge individual `BeginFrames`, we need to identify them. For this purpose, we introduce two new fields to `BeginFrameArgs`:

```
struct CC_EXPORT BeginFrameArgs {
  [...]

  // |source_id| and |sequence_number| identify a BeginFrame within a single
  // process and are set by the original BeginFrameSource that created the
  // BeginFrameArgs. When |source_id| of consecutive BeginFrameArgs changes,
  // observers should expect the continuity of |sequence_number| to break.
  uint32_t source_id;
  uint64_t sequence_number;
};
```

We also need to identify the source that a `BeginFrame` originated at, because it may change while prior `BeginFrames` are still in-flight. The new source can then identify stray acknowledgements that belong to a `BeginFrame` sent by the old source.

BeginFrame acknowledgements

Damage producers will send a `BeginFrameAck` struct either bundled within the metadata of a `CompositorFrame` or individually to confirm each `BeginFrame` they receive.

```
// Sent by a damage producer as acknowledgment of completing a BeginFrame.
struct CC_EXPORT BeginFrameAck {
  // Source id and sequence number of the BeginFrame that is acknowledged.
  uint32_t source_id;
  uint64_t sequence_number;

  // |true| if the producer had damage (e.g. sent a CompositorFrame or
  // damaged a surface) as part of responding to the BeginFrame.
  bool has_damage;
};
```

Tracking producer / surface state for DisplayScheduler

Ultimately, the `Display/DisplayScheduler` needs to know when all "relevant" damage producers have completed the current `BeginFrame`, so that it can then trigger its `BeginFrame` deadline early. A damage producer is only relevant to a specific `Display` and to the current BeginFrame if

1) the producer received said `BeginFrame`, and
2) the producer's damage would be visible on the `Display`, i.e. its `Surface` is included in the `Display`'s surface hierarchy.

If it a producer is relevant, the `DisplayScheduler` will consider its corresponding `Surface` ready-to-draw if: the `Surface` has damage[1], or

3) the producer has acknowledged the current `BeginFrame`.

For this purpose, the `DisplayScheduler` will track for each `Surface`:

a) the last `BeginFrame` handled by its producer, and
b) the last `BeginFrameAck` received from its producer.

The `DisplayScheduler` will use this state to determine if all surfaces included in its hierarchy are ready to draw, and thus, the current `BeginFrame`'s deadline should be triggered immediately.

Fully unifying BeginFrame distribution

We can only remove the `DisplayScheduler`'s old heuristics if

- all damage producers, on all layers, consistently acknowledge `BeginFrames`, and
- `BeginFrames` are only created in and distributed from one place.

In particular, `RenderWidgetHostViewAndroid` needs to receive `BeginFrames` from the `Display`'s `BeginFrameSource`, the browser in `Mus` needs to receive `BeginFrames` from `Mus`' `BeginFrameSource`, and all `CompositorFrame` creators (in all display compositor clients and on all platforms) need to issue acknowledgments for `BeginFrames` they receive.

Alternatives considered

We first prototyped a similar approach, which transported `BeginFrameAcks` via the `BeginFrameSources`, rather than the `FrameSink` hierarchy. This approach was replaced by the one described above, because the `BeginFrameSource` hierarchy doesn't necessarily reflect the surface hierarchy and - because of this -

[1] We need consider any surface with damage as ready-to-draw, regardless of its producer's last BeginFrameAck, because its producer will be CompositorFrameAck-blocked until the damage was drawn.

the `DisplayScheduler` would have had difficulties in determining which producers (`BeginFrameObservers`) to wait for.