Better Video Frame Scheduling

Andrew Scherkus (scherkus@chromium.org)
March 25, 2014
Bug: https://code.google.com/p/chromium/issues/detail?id=336733

Background

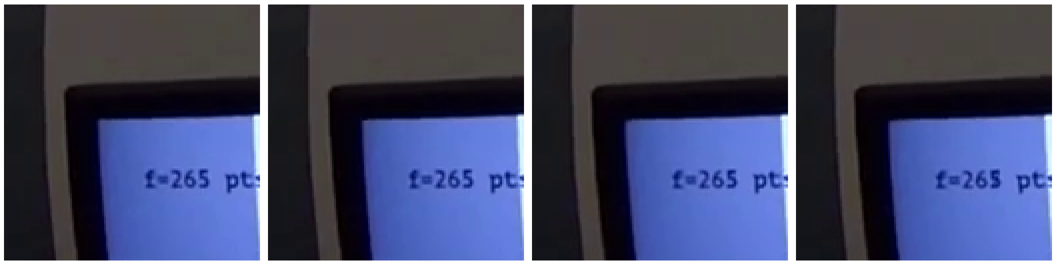
Today the media stack notifies the compositor whenever a new frame is available.

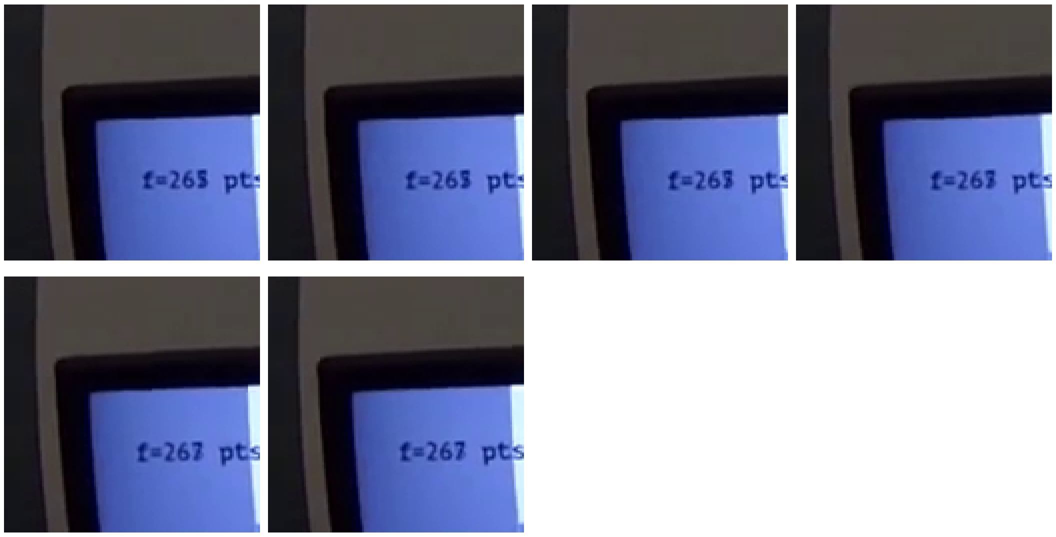
If there is nothing else on the page that would cause the compositor to keep compositing each vsync (e.g., no animations visible during video playback), then it's possible the media stack will drop the new frame before the next vsync arrives resulting in dropped frames and stuttering/janky video.

Here's a contrived example where for whatever reason (e.g., scheduling hiccups) the compositor doesn't show frame 4 because the media stack swaps in frame 5 by time the compositor decides it's appropriate to generate new output.

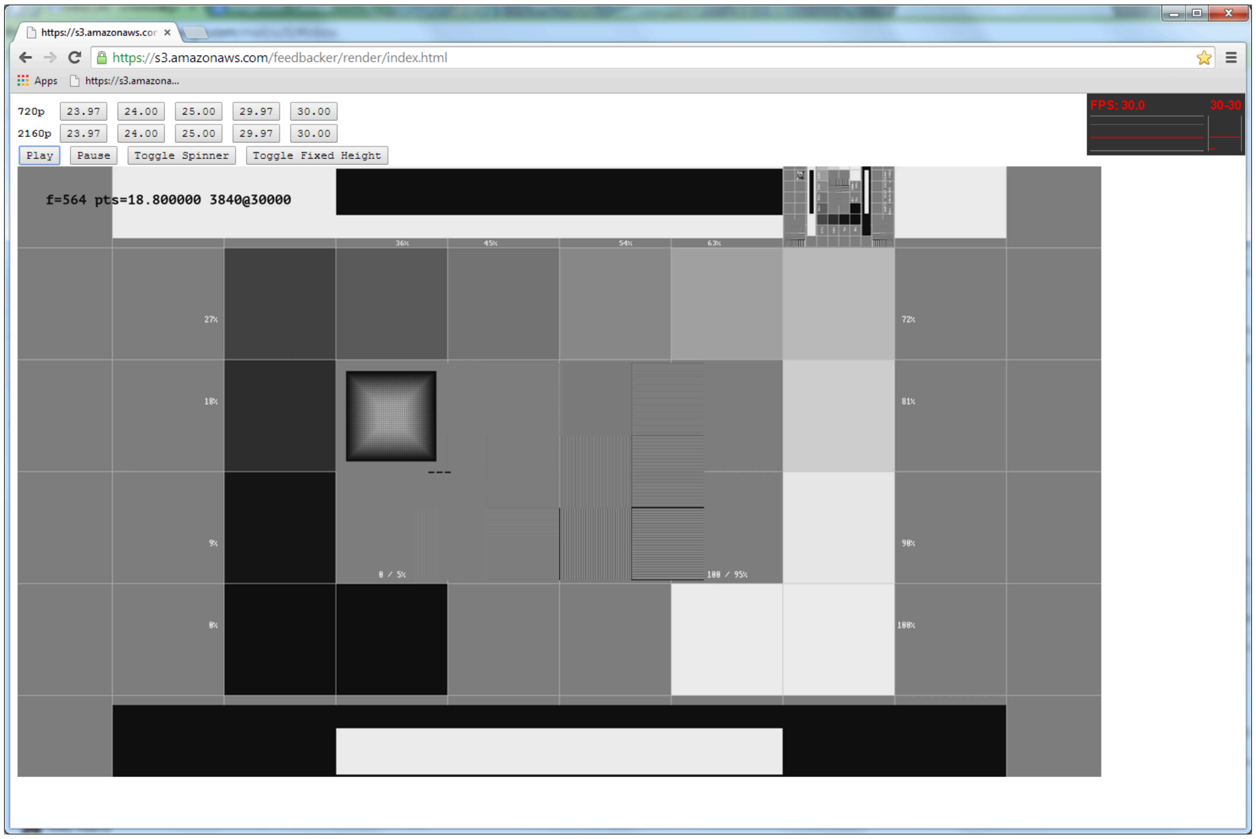
```
frame displayed     1       2       3               5           6
vsync @ 60 Hz   |---|---|---|---|---|---|---|---|---|---|---|---|---|
video @ 25 Hz   |--------|--------|--------|--------|--------|-------
frame # ready   1        2        3        4        5        6
```

Here's a [real world example](#) recorded with a high speed camera using [special test content](#). Note frame 266 never appears. Recording a trace while playing back this content reveals that the video frame was ready and did notify the compositor, but that the scheduler decided to skip out compositing.





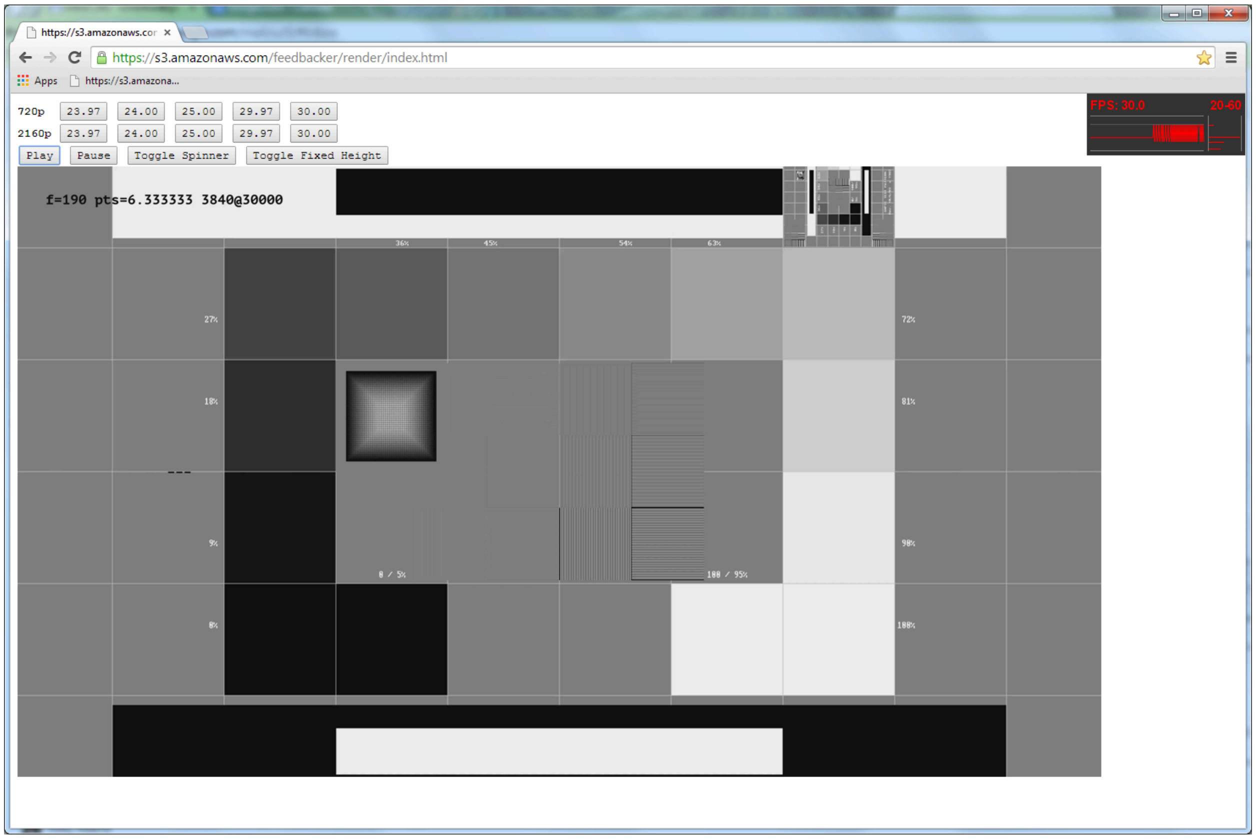
Stuttering/janky video shows up in the FPS counter as thick bands of oscillating frame rates:



https://s3.amazonaws.com/feedbacker/render/index.html

```
f=564 pts=18.800000 3840@30000
```

Rock solid 30 frames per second. Yay!



```
f=190 pts=6.333333 3840@30000
```

Jittery video rendering and compositor oscillating between 20-60 frames per second. Boo!

The media stack is capable of producing video frames well in advance, so ideally we should never be dropping video frames due to missing vsync. We can do better!

Option 1: Scheduler polls VideoLayerImpls for new frames

Treat video similar to animations. The compositor is told when there is video content playing, which causes the scheduler to keep waking up every vsync to do a lightweight poll of VideoLayerImpls for the presence of new frames. If there are no new frames, it bails out as early as possible to avoid doing unnecessary work.

```
  class LayerImpl {
+   // Defaults to false, VideoLayerImpl would return true if
+   // media stack notifies its VideoLayerImpl that video is playing
+   // or not.
+   virtual bool HasContinuouslyUpdatedContent() const;
  };

  void LayerTreeHostImpl::AnimateScrollbarsRecursive(...) {
```

```
+   if (layer->HasContinuouslyUpdatedContent()) {
+     // Mark damage for the VideoLayerImpl.
+   }
  }
```

Prototype CL: https://codereview.chromium.org/112623004/

Pros:
- Fairly simple addition to existing animation path
- Avoids having the scheduler continuously flip-flop between active and inactive states
- Single video frame is outstanding at any given time (keeps decoding pipeline running)

Cons:
- Compositor will be waking up more often than necessary since video frame rates are typically less than display refresh rate
- Additional plumbing of media-specific state into compositor

Option 2: Media stack waits until frame is composited

The media stack continues to notify the compositor when frames are available. Instead of only keeping a single frame outstanding, the media stack will queue up to a single extra frame if we detect the compositor has not composited the previous frame. If a third frame comes along before anything has been composited, the media stack starts dropping frames. Ideally the compositor should be able to consume video frames fast enough to avoid queuing any frames at all.

Prototype CL: https://codereview.chromium.org/213123005/

Pros:
- Compositor will only wake up as necessary
- Avoids plumbing media-specific state into compositor

Cons:
- I don't think this fixes the problem as the scheduler is still suspect to flip-flopping between active and inactive states
- Up to two video frames are outstanding at any given time (can stall decoding pipeline)

Option 3: Plumb deadline information into media stack

This is a little hand-wavey, but if the media stack knows when the deadlines are going to happen, it could make better judgements as to when it can notify the compositor.

I have no idea whether this is possible.

Option 4: Figure out why the scheduler keeps oscillating

In https://codereview.chromium.org/112623004/#msg14 brianderson@ pondered the following:

If the video is making FrameRateController::SetActive toggle on and off, we could be deciding whether or not to retroactively process the missed BeginImplFrame (and thus when to draw) in a jittery manner. I wonder what it would look like if we always retroactively processed a missed BeginImplFrame immediately, which would cause the compositor to pick up and draw new video frame notifications immediately.

If traces show that we are alternating between retroactive and non-retroactive BeginImplFrames, we could try changing the following line in FrameRateController::SetActive to see if it makes a difference:

```
  return BeginFrameArgs::Create(missed_tick_time, deadline + deadline_adjustment_,
interval_);
```

to:

```
  return BeginFrameArgs::Create(missed_tick_time, missed_tick_time + interval_, interval_);
```

I tried the above but it didn’t fix anything. I need brianderson@ to help me out here. Here’s a trace where the stuttering is happening during ~3 seconds until ~9 seconds:
https://s3.amazonaws.com/feedbacker/render/video_jank_trace.json

Build info: Release build of Chromium, Windows 32-bit, r259634,
GYP_DEFINES="ffmpeg_branding=Chrome use_proprietary_codecs=1
component=shared_library"

Update April 4, 2014

Trace using software decoding. Lots of stuttering happening during ~14 seconds until ~21 seconds of the trace:
https://s3.amazonaws.com/feedbacker/render/video_jank_trace_sw_decode.json

Build info: Release build of Chromium, Windows 32-bit, r261815,
GYP_DEFINES="ffmpeg_branding=Chrome use_proprietary_codecs=1
component=shared_library"