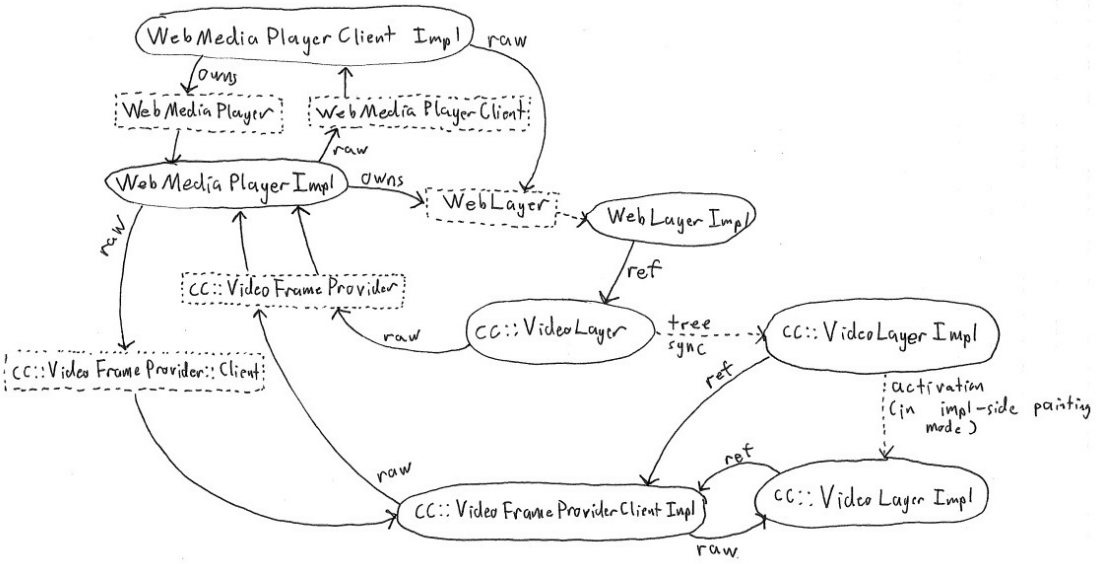
Video Playback and Compositor

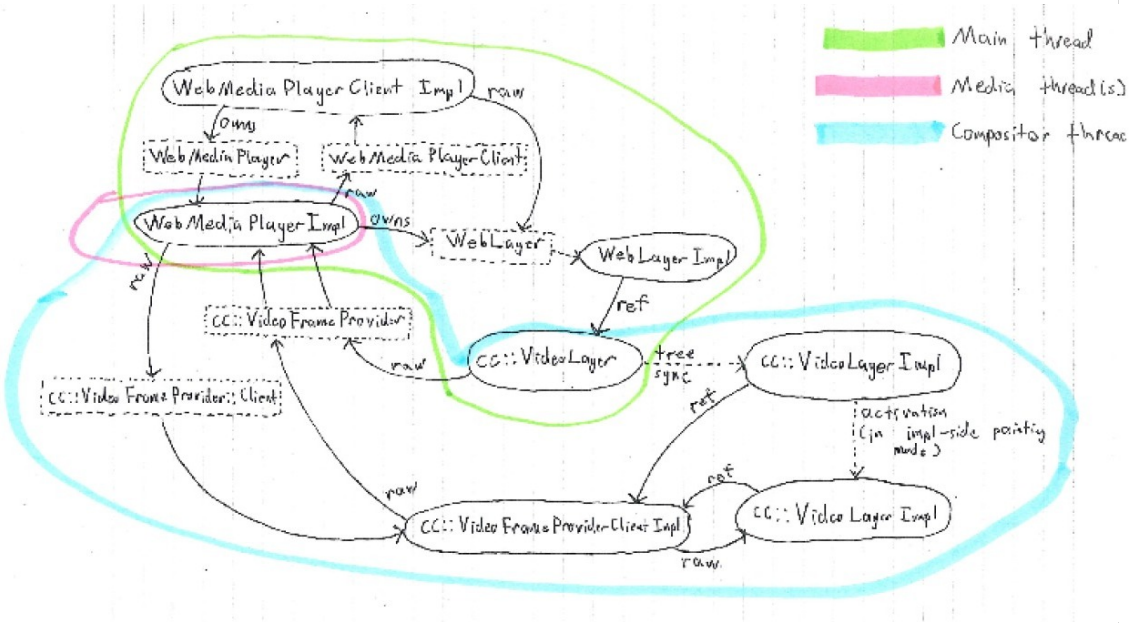
Authors: jamesr@chromium.org, danakj@chromium.org

The Chromium compositor has support for video playback to support offloading work to the GPU and displaying video frames while the main thread is blocked. There are a few different media engine implementations in Chromium, but they interact with the compositor in similar ways. Here's a diagram of the different classes and how they interact:



Things in dotted square boxes are interfaces, things in ovals are concrete classes. Arrows represent pointers between objects.

Objects by thread (note that a few span multiple threads):



Before video playback:
1.) WebMediaPlayerClientImpl constructed by WebCore
At start of video load:
2.) WebMediaPlayerClientImpl constructs WebMediaPlayer
In SetReadyState, if the video has loaded enough to know we want composited playback:
3.) WebMediaPlayerImpl constructs a WebLayerImpl wrapping a cc::VideoLayer with the provider pointer set to the WebMediaPlayerImpl

4.) WebMediaPlayerImpl provides WebLayer* to WebMediaPlayerClientImpl to register in WebCore's compositing tree
On the next compositor commit during tree sync (on compositor thread with main thread blocked):
5.) cc::VideoLayer creates a cc::VideoLayerImpl and passes the cc::VideoFrameProvider*
6.) cc::VideoLayerImpl creates a cc::VideoFrameProviderClientImpl with the provider pointer
7.) cc::VideoFrameProviderClientImpl calls SetVideoFrameProviderClient(this) on the cc::VideoFrameProvider*, which is the WebMediaPlayerImpl
On tree activation (immediately after tree sync when not in impl-side painting, at some point later on in impl-side painting)
8.) Active tree cc::VideoLayerImpl sets itself as the cc::VideoFrameProviderClientImpl's active layer

If we fully initialize without anything shutting down, then the pathway for the video layer to grab a new frame is:

On the compositor thread when preparing to draw a frame:
1.) cc::VideoLayerImpl::WillDraw calls VideoFrameProviderClientImpl::AcquireLockAndCurrentFrame()
2.) VFPCI::ALACF takes its provider_lock_ and calls GetCurrentFrame() on its provider
3.) WebMediaPlayerImpl takes its internal lock_ and returns its current_frame_
4.) cc::VideoLayerImpl uploads data into textures (if needed)
On the compositor thread when drawing a frame:
5.) cc::VideoLayerImpl::AppendQuads adds quads into the draw list referencing the video texture(s)
On the compositor thread after drawing a frame:
6.) cc::VideoLayerImpl::DidDraw calls VideoFrameProviderClientImpl::PutCurrentFrame
7.) VFPCI::PCF calls PutCurrentFrame() on its provider
8.) cc::VideoLayerImpl::DidDraw calls VFPCI::ReleaseLock which releases provider_lock_

The way the media system can notify the compositor of new video frames on the compositor thread is (currently implemented only on android):

On the compositor thread when the media system has a new frame available
1.) Something in the media playback system (content::StreamTextureProxyImpl for android) calls cc::VideoFrameProvider::DidReceiveFrame()
2.) cc::VideoFrameProviderClientImpl checks if it has an active cc::VideoLayerImpl and calls SetNeedsRedraw() on it if so
3.) cc::VideoLayerImpl::SetNeedsRedraw tells the compositor to schedule a new frame

Shutdown

There are a few ways this system can shut down. All shutdown paths start on the main thread, but they can propagate out to other threads in different ways depending on the type of shutdown.

Video layer removed from compositing tree while playing

This can happen if a <video> element is detached from the DOM tree or display:none is set on an ancestor of the element.

There are two cases to consider here - impl-side painting off and impl-side painting on. With impl-side painting off, there is only one cc::VideoLayerImpl associated with a given cc::VideoLayer. The cc::VideoLayerImpl is always created or destroyed during the tree sync part of commit which happens on the compositor thread with the main thread blocked.

With impl-side painting on, there are potentially two cc::VideoLayerImpls associated with a given cc::VideoLayer. One is in the pending/recycle tree and is always created/destroyed during the commit tree sync on the compositor thread with the main thread blocked. The other is in the active tree and is created/destroyed during tree activation, which happens on the compositor thread while the main thread is **not** blocked. In impl-side painting mode both cc::VideoLayerImpls hold a reference to the same cc::VideoFrameProviderClientImpl.

Preconditions:

1.) WebMediaPlayerClientImpl owns a WebMediaPlayer (which is a WebMediaPlayerImpl)
2.) WebMediaPlayerImpl owns a WebLayerImpl, which has a ref to a cc::VideoLayer
3.) cc::VideoLayer has an associated cc::VideoLayerImpl (two in impl-side painting mode) which reference a cc::VideoFrameProviderClientImpl
4.) WebMediaPlayerImpl's provider_client_ pointer points to the cc::VideoFrameProviderClientImpl
5.) cc::VideoFrameProviderClientImpl's provider_ pointer points to the WebMediaPlayerImpl

Sequence:

On the main thread at any point in time
1.) cc::VideoLayer removed from the compositing tree and/or destroyed

In the next compositor commit on the compositor thread with the main thread blocked:
2.) Tree sync starts destruction of the cc::VideoLayerImpl (only cc::VideoLayerImpl when not in impl-side painting, pending/recycle layer in impl-side painting)
3.) ~cc::VideoLayerImpl() calls cc::VideoFrameProviderClientImpl::Stop()
4.) cc::VFPCI::Stop() calls SetVideoFrameProviderClientImpl(NULL) on its cc::VideoFrameProvider*, which is a WebMediaPlayerImpl
5.) cc::VFPCI::StopUsingProvider() takes its provider_lock_ and nulls out its provider_
6.) WebMediaPlayerImpl nulls out its provider_client_ pointer
7.) cc::VFPCI::Stop() nulls out its provider_ pointer (again)
If in impl-side painting mode, on tree activation in the compositor thread:
8.) cc::VideoLayerImpl in active tree destroyed, dropping last reference to cc::VideoFrameProviderClientImpl

Ordering:

Steps 2-7 happen on the compositor thread with the main thread blocked. This means that the compositor cannot be in a frame for any of these steps and nothing on the main thread can advance (i.e. we can't start shutting down the WebMediaPlayerImpl).

Postconditions:
1.) WebMediaPlayerImpl's provider_client_ pointer is NULL (as of step 7)
2.) cc::VideoFrameProviderClientImpl's provider_ pointer is NULL (as of step 6)
When impl-side painting is not enabled, then
3.) cc::VideoLayerImpl and cc::VideoFrameProviderClientImpl destroyed (as of step 9)

Media engine shut down while video layer is in compositing tree

Preconditions:

1.) WebMediaPlayerClientImpl owns a WebMediaPlayer (which is a WebMediaPlayerImpl)
2.) WebMediaPlayerImpl owns a WebLayerImpl, which has a ref to a cc::VideoLayer
3.) cc::VideoLayer has an associated cc::VideoLayerImpl (two in impl-side painting mode) which reference a cc::VideoFrameProviderClientImpl
4.) WebMediaPlayerImpl's provider_client_ pointer points to the cc::VideoFrameProviderClientImpl
5.) cc::VideoFrameProviderClientImpl's provider_ pointer points to the WebMediaPlayerImpl
6.) We're outside a compositor commit on the main thread
7.) The compositor thread may be in any state, including activation or drawing a frame.

Sequence:

On the main thread at any point in time:
1.) WebMediaPlayerClientImpl destroys its WebMediaPlayer (which is a WebMediaPlayerImpl)
2.) ~WebMediaPlayerImpl() calls SetVideoFrameProviderClient(NULL) on its provider_client_, which is a cc::VideoFrameProvierClientImpl
3.) cc::VideoFrameProviderClientImpl::SetVideoFrameProviderClient() takes its provider_lock_
4.) cc::VideoFrameProviderClientImpl::SetVideoFrameProviderClient() nulls out its provider_ pointer, which pointed to the WebMediaPlayerImpl

5.) cc:VFPCI::SetVideoFrameProviderClient() releases its provider_lock_
6.) ~WebMediaPlayerImpl() completes and the object is destroyed

Ordering:

1.) In step (3) the cc::VFPCI takes its provider_lock_. If the compositor thread is in the middle of drawing a frame (specifically between WillDraw() and DidDraw()), this call will block until the frame is complete

Postconditions:

1.) WebMediaPlayerImpl's provider_client_ pointer is NULL (as of step 4)
2.) cc::VideoFrameProviderClientImpl's provider_ pointer is NULL (as of step 5)

Comments

You do not have permission to add comments.