Chrome Windows Direct Composition Explainer

sunnyps@chromium.org

PUBLIC DOCUMENT

Overview

This document explains the display pipeline used by display compositor in Chrome on Windows platform.

Background

Renderer processes produce CompositorFrames which contain a list of RenderPasses. A RenderPass contains a list of DrawQuads, shared state between quads, post-processing filters, CopyOutputRequests for reading back rendered pixels, damage rect, color space, etc.

A DrawQuad is a bag of data used to render a quad in some content space interpreted according to DrawQuad type to a target space using an arbitrary transform with a set of textures. There are different DrawQuad types e.g. tiled content (from renderer compositor layers), video (typically YUV textures), solid color, single texture (from webgl), etc.

Chrome's display compositor is responsible for collecting CompositorFrames from all sources such as renderers, OOPIFs, browser UI, etc. and issuing drawing commands. This code runs on the browser process currently, but it will move to the viz process in the future. GLRenderer processes RenderPasses and issues GL commands for drawing.

Recent graphics hardware has the ability to scanout multiple buffers (overlays) and "composite" them on the fly. This is usually a power and performance improvement especially for video since there is often hardware support for YUV overlays. However, there are restrictions on the count, buffer format, and transforms of overlays.

DirectComposition is a new API on Windows 8 and above that supports drawing layers of content (IDCompositionVisual) organized into a tree. These visuals are transactionally updated using "Commit". Rendering updates can be synchronized with commit using special drawing surfaces (IDCompositionSurface). DWM, the system compositor on Windows, can map swap chain visuals to hardware overlays. Chrome uses visuals for video to use hardware overlays and reduce power consumption.

Other docs:

- RenderPasses and Quads: https://docs.google.com/presentation/d/14FIKgkh0-4VvM5vLeCV8OTA7YoBasWIwKIJyNnUJItM/edit?usp=sharing and https://drive.google.com/file/d/0BwPS_JpKyELWTURjMS13dUJxR1k/view (video of a presentation)
- Windows video overlays doc: https://docs.google.com/a/chromium.org/document/d/1hmjMKb1DU9ZD2_NzPfFQYGFS1Urc4dqr_mdGdQxPT-M/edit?usp=sharing

Legacy Pipeline

The legacy pipeline uses GLSurfaceEGL which wraps a native EGL window surface. ANGLE implements this using a swap chain created for the window. RenderPasses are composited to the back buffer in GLRenderer. This is similar to other platforms such as Linux GLX and Android EGL.

Direct Composition Pipeline

Root surface

In the GPU process, DirectCompositionSurfaceWin holds the root surface (DirectCompositionChildSurfaceWin), and the tree of visuals (DCLayerTree). The root surface is backed by either a swap chain, or a direct composition surface. The latter is used when there are overlays in DCLayerTree because rendering to the root surface must be synchronized with visual updates for overlays to prevent guttering in root surface exposed when an overlay is resized. Swap chains are used otherwise because they consume less power.

If a direct composition surface is used, drawing commands have to be demarcated using BeginDraw and EndDraw calls, and a damage rect has to be provided. Therefore, the client has to call SetDrawRectangleCHROMIUM before drawing to issue BeginDraw. EndDraw is called automatically by SwapBuffers. Drawing commands must render using an offset returned by BeginDraw which is exposed via GetDrawOffset, and used by command decoder to update viewport and scissor rect.

Video overlays

DirectRenderer::BeginDrawingFrame uses a platform specific OverlayProcessor to process all render passes and quads. For DirectComposition, DCLayerOverlayProcessor produces a list of DCLayerOverlays. DCLayerOverlay is a bag of data needed to produce a DirectComposition visual. It is roughly equivalent to a DrawQuad but has some additional information like z-order.

Currently, only YUVVideoDrawQuads in the root render pass with 2d axis aligned preserving transforms and SrcOver blending are considered as overlay candidates. These quads fall into two general categories: overlays and underlays.

Overlays are unoccluded by other quads so they can be placed on top of other visuals. The quad is removed from the render pass and its damage is subtracted.

Underlays are occluded by other quads so they are placed on bottom of other visuals. The quad is removed from the render pass, and a transparent solid color quad is punched through in the render pass. This is behind a flag today.

Overlays are submitted to DirectCompositionSurfaceWin in GPU process using ScheduleDCLayerCHROMIUM. These overlays are mapped to direct composition visuals during SwapBuffers. Previous frame's visuals are reused, if possible. Overlays are sorted using z order in DCLayerOverlay, and properties like transform, clip, etc. are applied.

Video content comes from accelerated decoder as DXGI images, or from software decoder in renderer as shared memory images. Shared memory images are uploaded to textures. Since only YUY2 is supported for hardware overlays, and video images are typically NV12, a VideoProcessor is used to convert from NV12 textures to YUY2. The video images can be scaled up using hardware overlays.

Future Work

- Make video overlays work for more cases such as underlays and non-root render pass quads (magchen@). Add support for a "require overlay" flag on video quads, and gracefully fallback when complex transforms are used (spec allows ignoring some CSS properties).
- Render tiles, webgl, canvas, etc. into direct composition visuals directly (sunnyps@, zmo@): Similar to Mac where we render into IOSurface backed textures which are set as content for CALayers. Direct composition surfaces are similar but don't allow fallback compositing in Chrome or copy to texture for CopyOutputRequest.