Behind Canvas

fserb@ (Feb/2020)

Canvas team: aaronhk@ fserb@ juanmihd@ yiyix@

Overview

Canvas Context (2D, WebGL, WebGPU, ImageBitmap)

HTML Canvas & OffscreenCanvas

Canvas Resource (texture) Management

What are we working on now

How Canvas 2D works

2D API

Maps JS Canvas2D API to cc:PaintCanvas (or SkCanvas).

It’s always deferred. Commands are stored and only executed once JS context is done.

“Raster” (converting high level 2D commands into GL calls) happens on the “main” thread and sent to the GPU command buffer.

We are moving towards OOP-R, where we will send 2D commands directly.

HTML Canvas & OffscreenCanvas

HTML Canvas is a TextureLayer. Resources are sent through TextureLayer as part of Paint.

Transferred OffscreenCanvas is a SurfaceLayerBridge. Resources are sent directly to the compositor by CanvasResourceDispatcher. By design, they are not in sync with main page paint, as they can be sent from workers.

Canvas Resources

Creates the **SharedImage** (or other backing textures, like SharedBitmap, GPUMemoryBuffer, etc..).

Lifecycle of the resource (sync token, compositor release frame, recycling)

For 2D, we provide a **PaintCanvas** paint interface for the texture and also do copy-on-write of previous texture.

Otherwise, we provide a Mailbox.

Also used by other clients, like drag&drop image.

Canvas loop

a day in the life of a Canvas Resource:

2D: Page has a 2D Canvas being modified on a RequestAnimationFrame
2D: All commands get recorded on a PaintCanvas until RAF/JS ends
CRP: Resource gets created or recycled
CRP: Previous frame gets copied into (if needed for 2D)
CRP: Texture is available for the clients to draw (either as PaintCanvas for 2D or as a Mailbox)
2D: Replay the PaintCanvas onto the texture
CRP: prepare resource to send to the compositor
(at this point, a new RAF may begin)
Compositor: uses the resource to render the page and eventually releases it
CRP: mark texture for recycling

Canvas Printing

(a small side note)

One of our recent launches is support vector printing of Canvases.

In supported cases (no image-rendering:pixelated, post onbeforeprint render), we replay the CanvasPaint on the printing paint, instead of the final canvas image.

Where things go wrong

Performance corner cases of deferral (1000s of source images).

Canvases can be bigger than the maximum GPU texture size.

There are still legitimate use cases for non SharedImage resources (desynchronized, for example).

getImageData.

Roundtrip means that each canvas must allocate at least 3 texture of the whole canvas.

Other Contexts not using CanvasResource may require a copy to send to compositor.

What are we working on now

New Canvas 2D API: batchDrawImage, filters, roundRect, conics, 3D transforms, records, batchFillText, willReadFrequently, Canvas2DContextLoss, more text modifiers, etc, etc.

New CanvasResource management. Remove a lot of complexity, make WebGL/WebGPU always use CanvasResources, make copy-on-write smarter, support dirty-rect for compositing. OOP-R.

Remove SetNextCommitWaitsForActivation from Canvas.

Eventually: release Canvas color managed.

gg