Firefox OS Graphics inside

16/Dec/2015

about:me

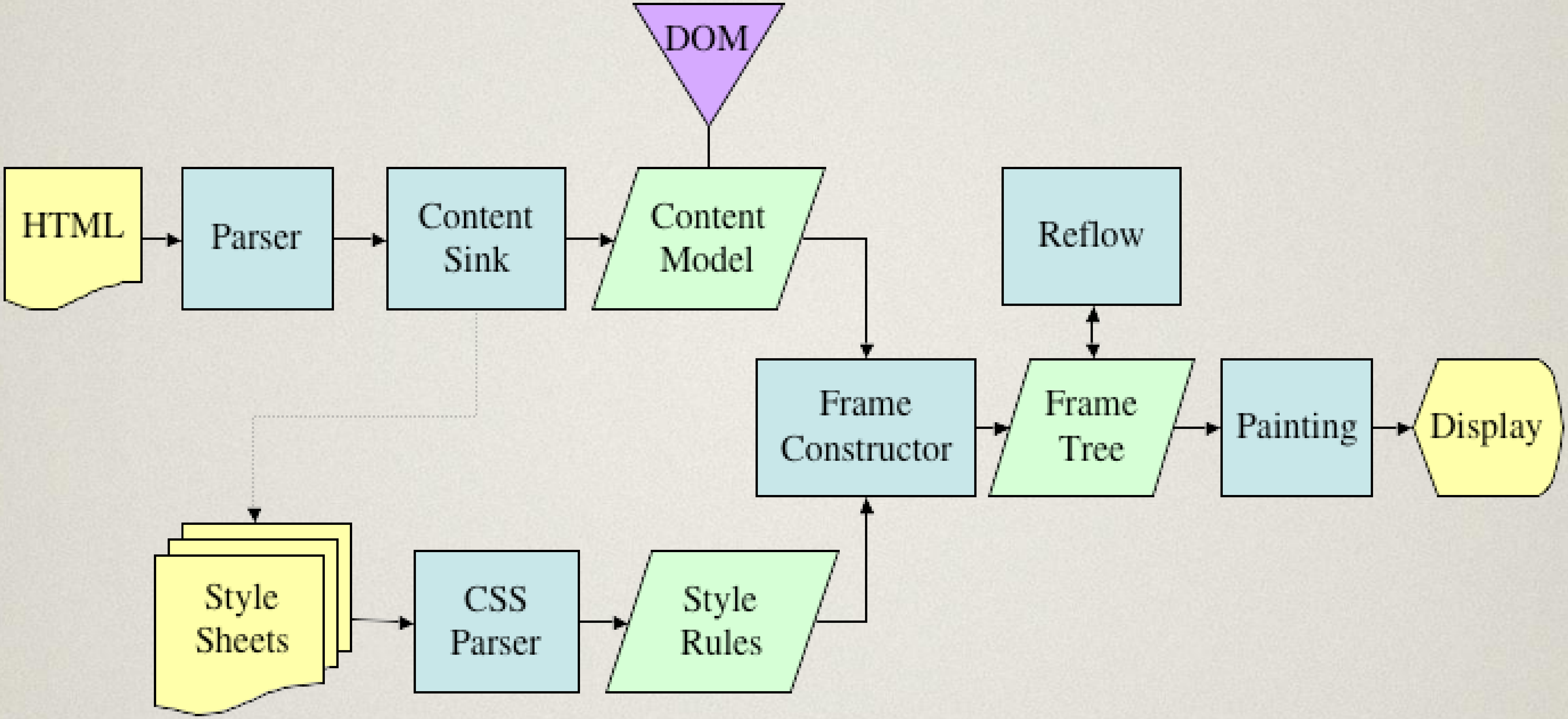
Sotaro Ikeda (池田総太郎)

- Mozilla Corporation (Since 2013)
- Current Work :
  - Graphics and Media
- Software Diagrams
  - Firefox-diagrams
    - https://github.com/sotaroikeda/firefox-diagrams/wiki/Firefox-Diagrams
  - Android Diagrams
    - https://github.com/sotaroikeda/android-diagrams/wiki/Android-Diagrams

about:document

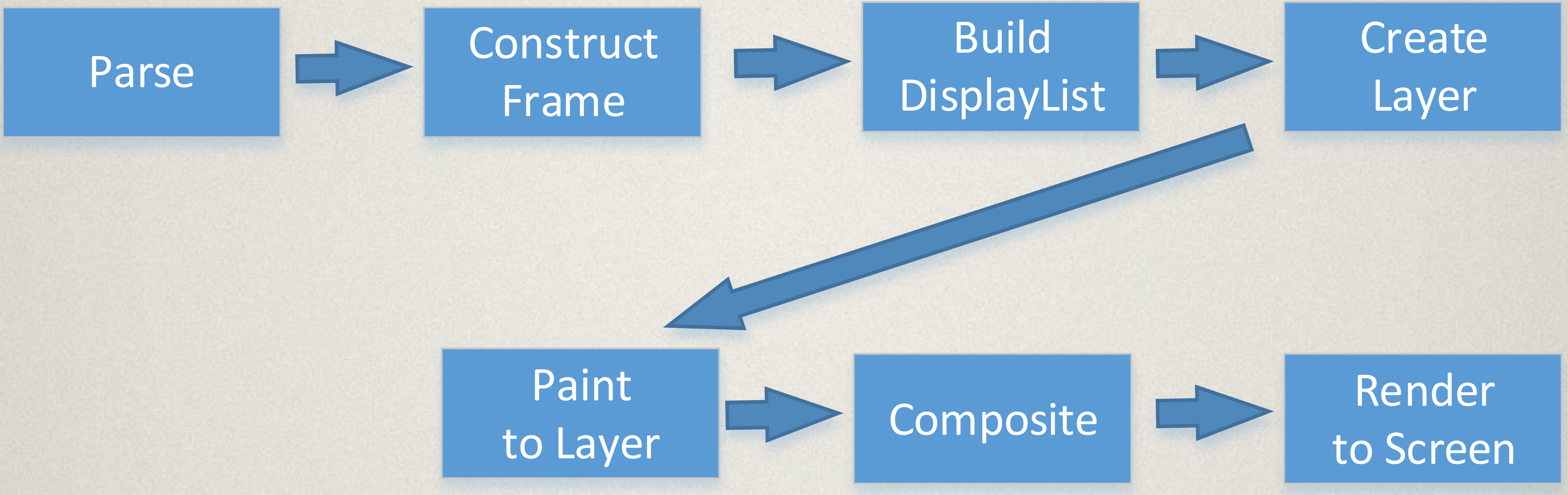
- Explains about internal software structures of Firefox OS graphics mainly about Layers and Compositing
- Diagrams are based on latest master gecko (gecko 45) on 16/Dec/2015
- More detailed diagrams are at
  - https://github.com/sotaroikeda/firefox-diagrams/wiki/Firefox-Diagrams
- Platform/GFX mozilla wiki
  - https://wiki.mozilla.org/Platform/GFX/

Rendering data flow

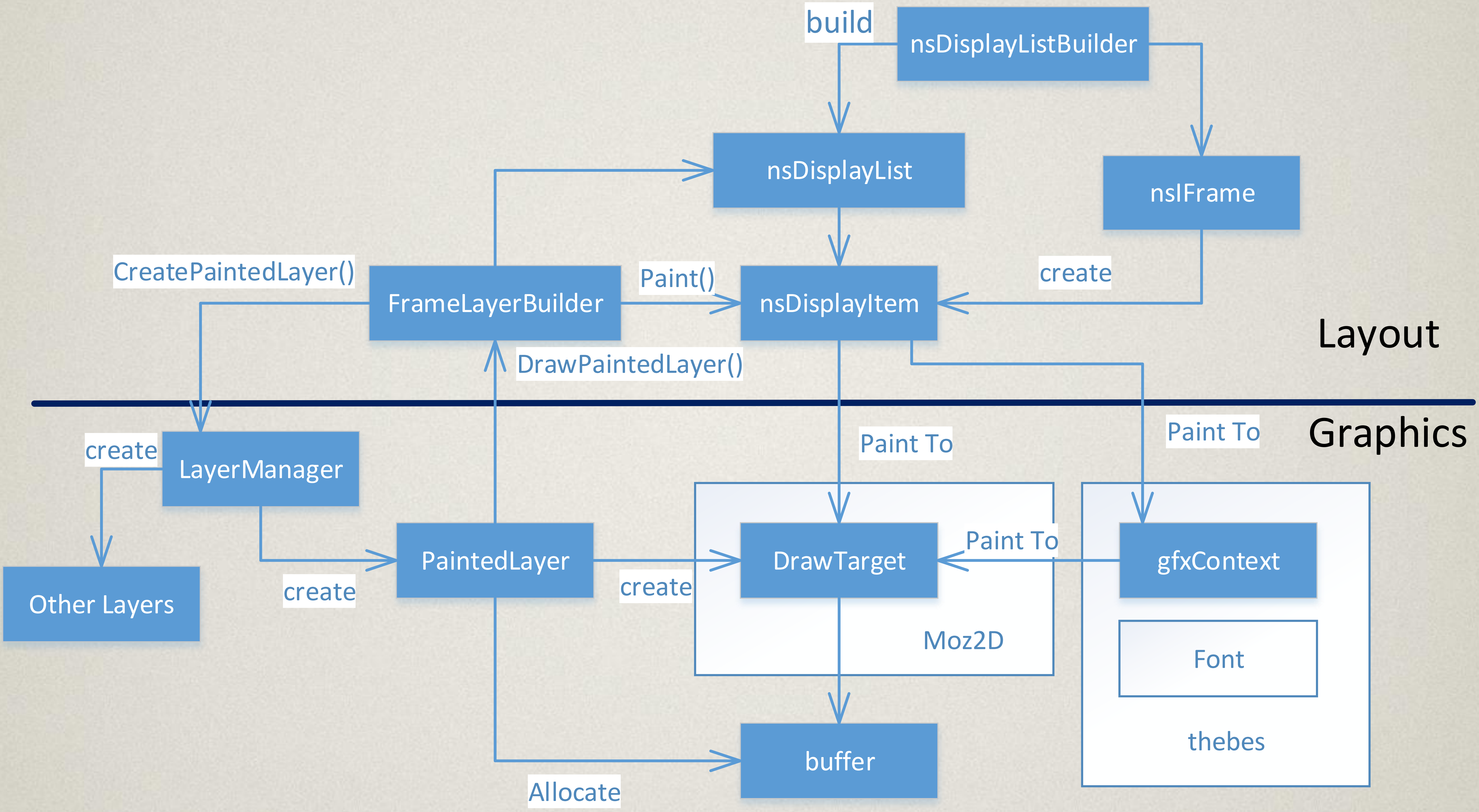


https://developer.mozilla.org/ja/docs/Introduction_to_Layout_in_Mozilla

Rendering data flow(added gfx steps)



Boundary between Layout and Graphics



Boundary between Layout and Graphics

- nsDisplayListBuilder
  - Manages a display list
  - Contains the parameters that don't change from frame to frame
  - nsIFrame can have many different visual parts. Constructs a display list for a frame tree that contains one item for each visual part
  - Display list could improve performance than traversing frame tree directly
- nsDisplayList
  - Manages a singly-linked list of display list items
  - The display list items are sorted by z-order
  - Can be used to paint the frames, to determine which frame is the target of a mouse event, and to determine what areas need to be repainted when scrolling
  - Role is similar to 'scene graph'
- nsDisplayItem
  - Unit of rendering and event testing
  - Each instance represents an entity that can be drawn on the screen e.g., a frame's CSS background, or a frame's text string

Boundary between Layout and Graphics

- FrameLayerBuilder
  - Responsible for converting display lists into layer trees
  - Every LayerManager needs an unique FrameLayerBuilder to build layers
- LayerManager
  - Controls a tree of layers
  - Uses transaction to update layers and painting of
- PaintedLayers
  - A state of the layer tree at the end of a transaction is rendered to the target
- Layer
  - Represents anything that can be rendered onto a destination surface
  - Layers are primarily used to minimize invalidating and repainting
- PaintedLayer
  - A Layer which we can paint into
- DrawTarget
  - Class used for all drawing of Moz2D
  - Layout paints into it directly or indirectly via gfxContext

Boundary between Layout and Graphics

- Moz2D
  - Cross-platform interface onto the various graphics backends that Gecko uses for rendering
  - Mostly stateless—better suited to CSS rendering and eliminates overhead
  - Floating-point—better suits platform APIs
  - API methods line up with HTML canvas
  - https://wiki.mozilla.org/Platform/GFX/Moz2D
- Thebes
  - C++ wrapper around Moz2D/Cairo, along with some Gecko-specific utility code, and a text API that uses platform text handling.
  - Cairo dependencies are going to be replaced by Moz2D
  - Predecessor of Moz2D, but still in use
  - Named after Cairo

LayerManager

Controls a tree of layers and uses transaction to update layers

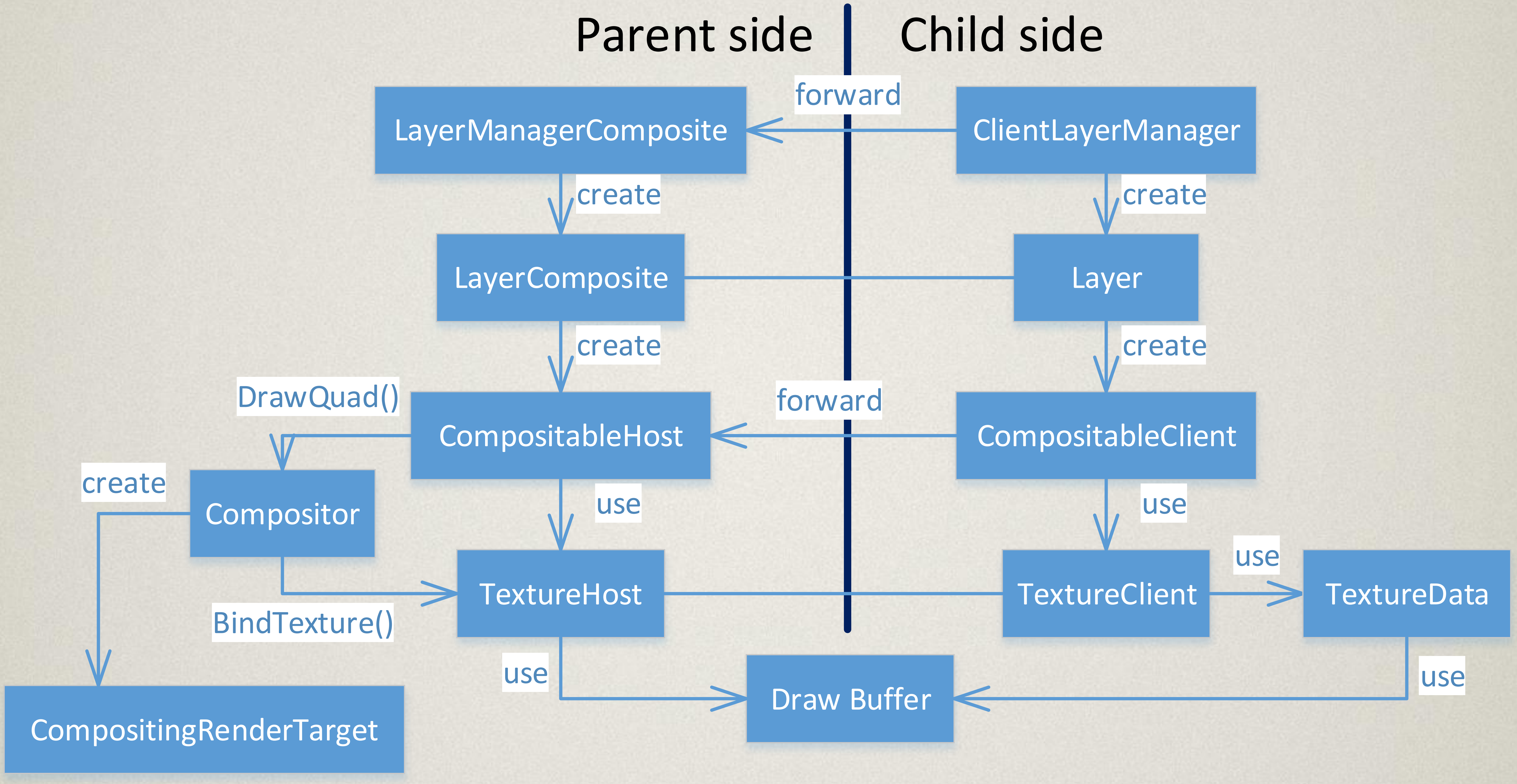
- ClientLayerManager
  - Used for off main thread composition for widget(screen)
  - Handles only active layers
  - Final composition is done by compositor on compositor thread
- BasicLayerManager
  - Used for the following use cases
    - Paints Inactive layers in PaintInactiveLayer()
    - Off screen document rendering
    - Main thread composition of widget (Not supported on Firefox OS)
  - During the drawing phase, each PaintedLayer is painted directly into the target

Layers

Represents anything that can be rendered onto a destination surface. Primarily used to minimize invalidating and repainting

- PaintedLayer
  - A Layer which we can paint into
- ImageLayer
  - A Layer which renders Images or video frames
  - From ClientLayerManager point of view, majority of Images are rendered into PaintedLayer
- ContainerLayer
  - A Layer which other layers render into. It holds references to its children
- ColorLayer
  - A Layer which just renders a solid color in its visible region
- CanvasLayer
  - A Layer for HTML Canvas elements
- RefLayer
  - ContainerLayer that refers to a "foreign" layer tree, through an ID.
  - Used to refer to a tree in a different process
- ReadbackLayer
  - It is created only by nsPluginFrame. Firefox OS does not use it

Compositing



https://wiki.mozilla.org/Gecko:Overview#Compositing

Compositing

Action of flattening Layers into the final image that is shown on the screen. ContainerLayers might be painted to intermediate surfaces during Compositing

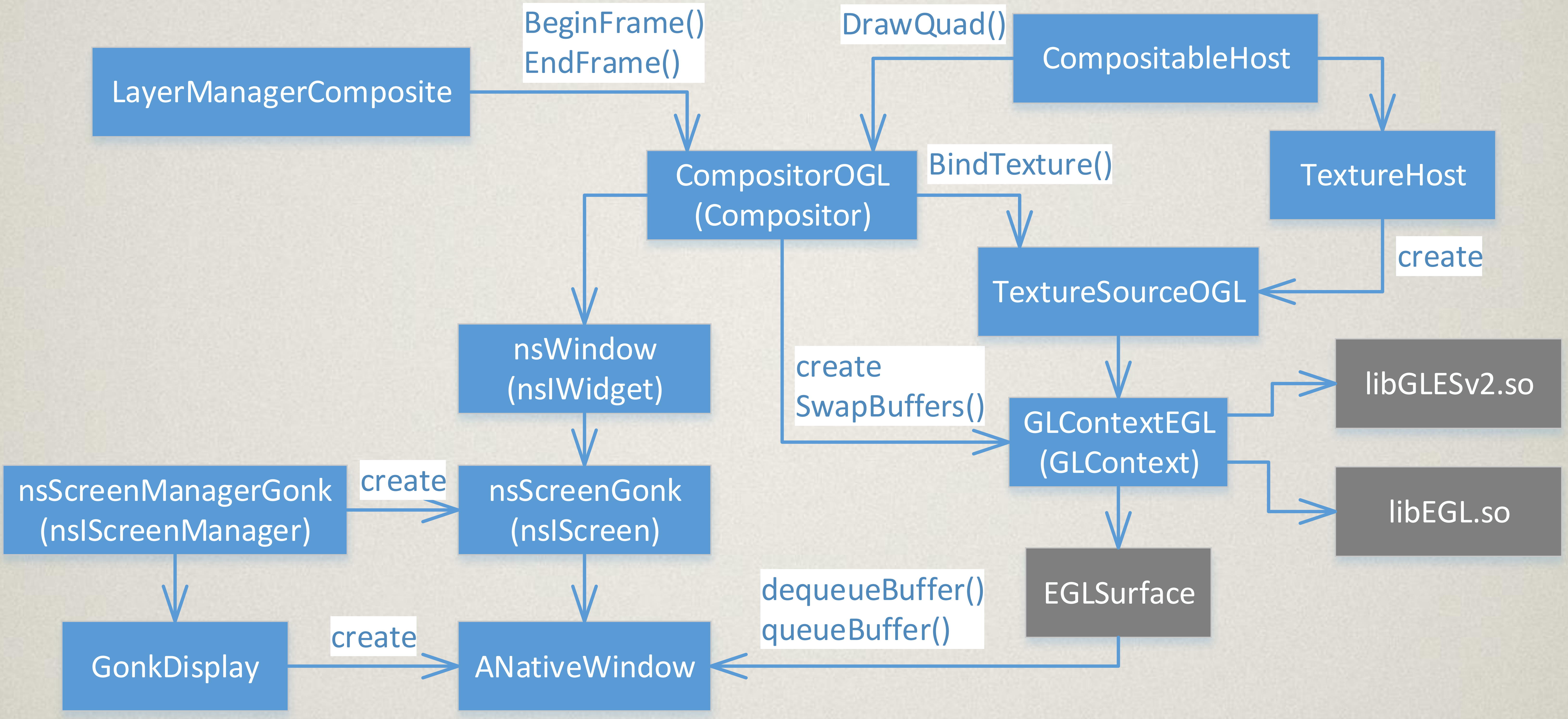
- Compositor
  - An object that can draw quads on the screen (or on an off-screen render target)
- Texture:
  - An object that contains image data
  - By Bug 1200595 fix, TextureClient's platform specific parts were split to TextureData
- Compositable
  - An object that can manipulate one or several textures, and knows how to present them to the compositor
  - Handle all the logic around texture transfer
- Layer
  - A layer usually doesn't know much about compositing. It uses a compositable to do the work

Compositors on Firefox OS

Two compositors are available on Firefox OS.

- CompositorOGL
  - Use OpenGL for Compositing
  - Firefox OS on gonk use it by default
- BasicCompositor
  - Use Moz2D for Compositing
  - Used when CompositorOGL is disabled
  - Can be used when platform does not have GPU
  - Can be tested by pref("layers.acceleration.disabled", true);

CompositorOGL

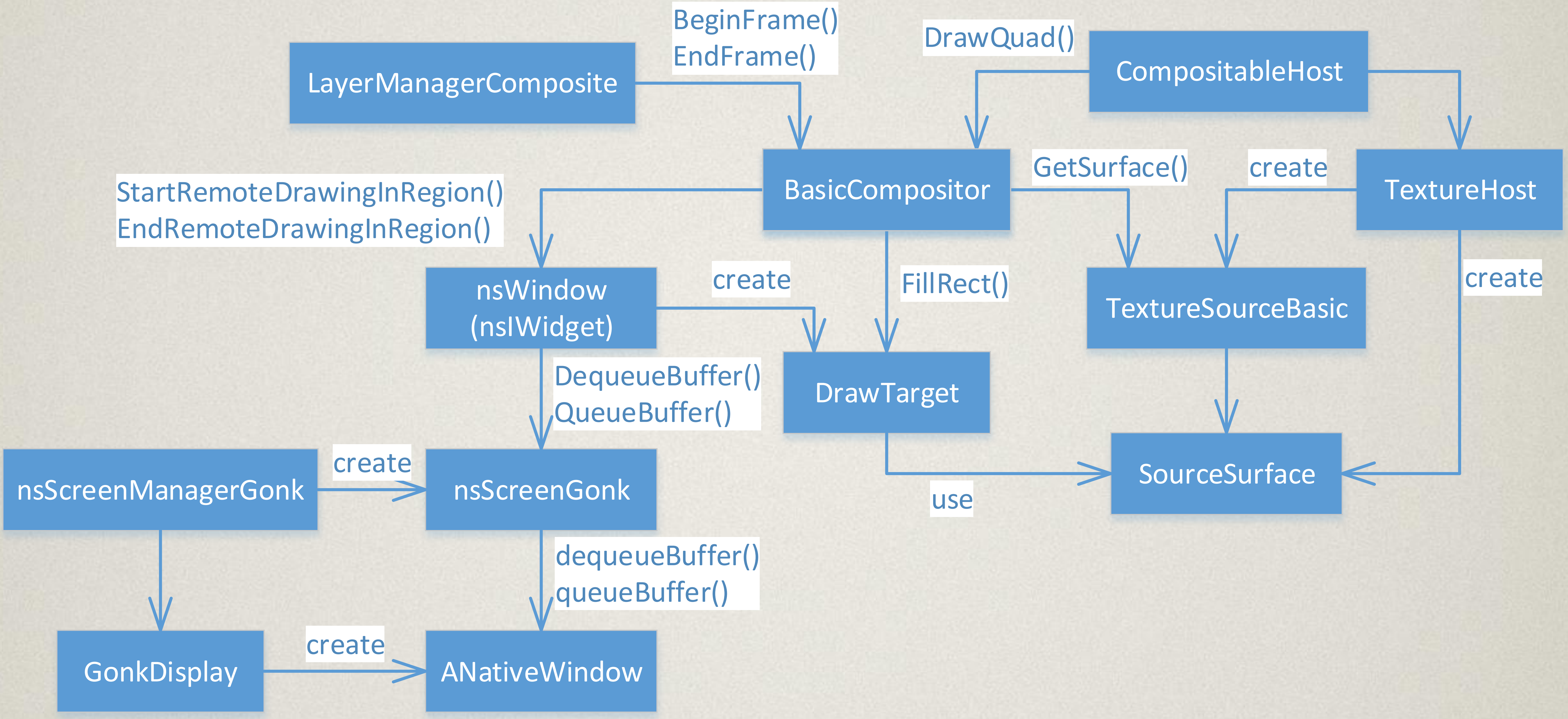


CompositorOGL

Uses OpenGL for Compositing. Its compositing is controlled by LayerManagerComposite. BeginFrame() starts a new frame compositing and EndFrame() flushes the current frame to the screen and tidy up

- nsWindow
  - Gonk’s implementation of nsIWidget. nsIWidget is a wrapper of OS platform related things. Roles of nsIWidget are different between os platforms. On gonk, it handles touch inputs and screen
- ANativeWindow
  - Android's struct to provide access to a native window and window buffers.
- nsScreenGonk
  - Gonk’s implementation of nsIScreen
  - It wraps android’s display surface
- GonkDisplay
  - Creates android’s display surface and renders boot animation
  - It exits to start boot animation as soon as possible before xpcom initialization
  - After boot completes, its roles are superseded by nsScreenGonk and HwcComposer2D

BasicCompositor



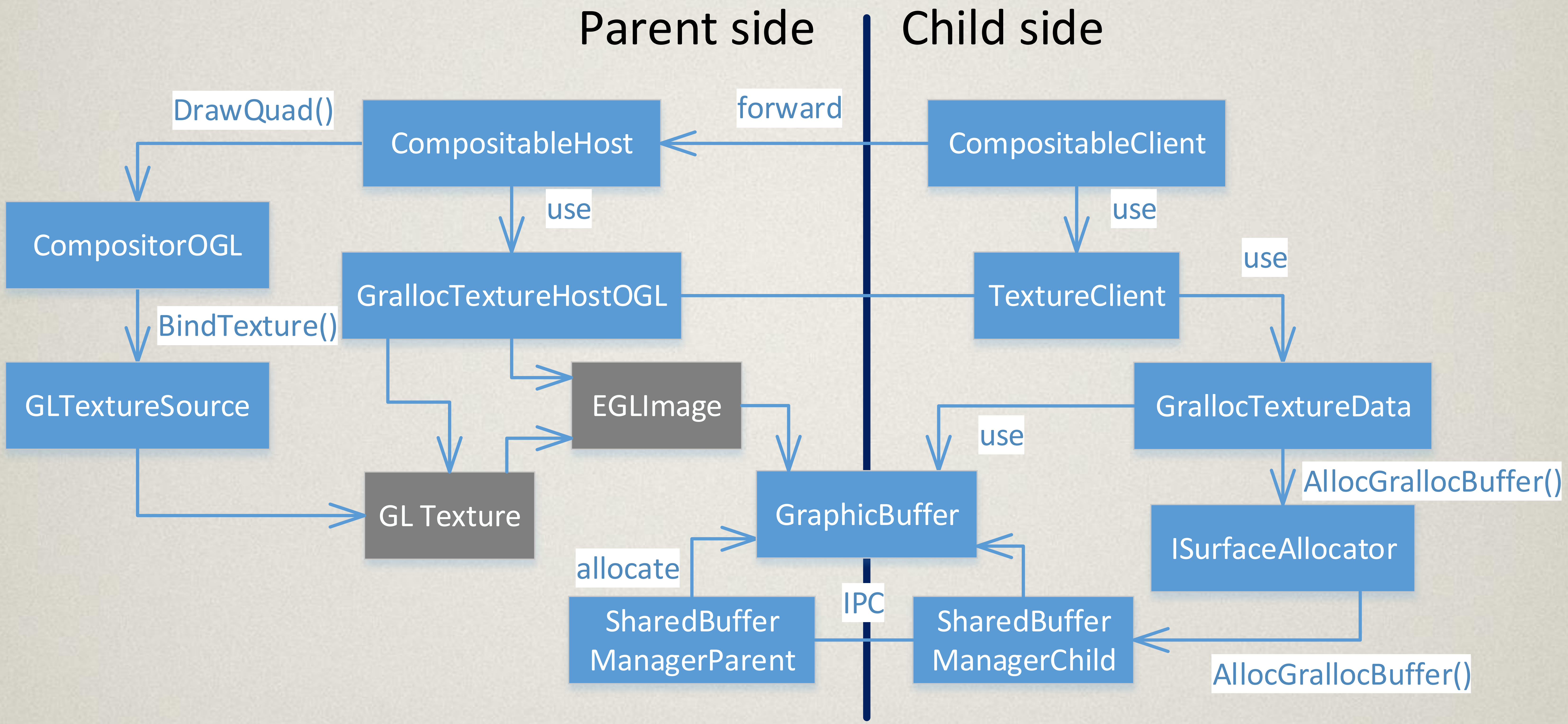
BasicCompositor

- Uses Moz2D for compositing
- On gonk, there is no valid use case except no GPU devices
- BasicCompositor gets DrawTarget from nsWindow
- Extra color conversions might happen during compositing on current gonk (like Flame)
  - Android applications draw with RGB color. But gecko draws contents with BGR

Types of TexutureData/TextureHost on Firefox OS

- GrallocTextureData/GrallocTextureHostOGL
  - A wrapper of android gralloc buffer
  - Always use GrallocTextureData if gralloc could be allocated
  - Disable gralloc for gfx::SurfaceFormat::A8
  - Disable gralloc if width/height is more than 4096. Many devices do not support more than 4096
  - GrallocTextureHostOGL is used when CompositorOGL is used
- ShmemTextureData/ShmemTextureHost
  - A wrapper of Shmem
  - Used when TextureClient is not in chrome process
  - Shmem is gecko's platform independent memory for cross process
  - On gonk, android's ashmem is used for Shmem
- MemoryTextureData/MemoryTextureHost
  - A Wrapper of raw memory
  - Used when TextureClient is in chrome process
- GrallocTextureHostBasic
  - GrallocTextureHostBasic is used when BasicCompositor is used
  - Used with GrallocTextureData

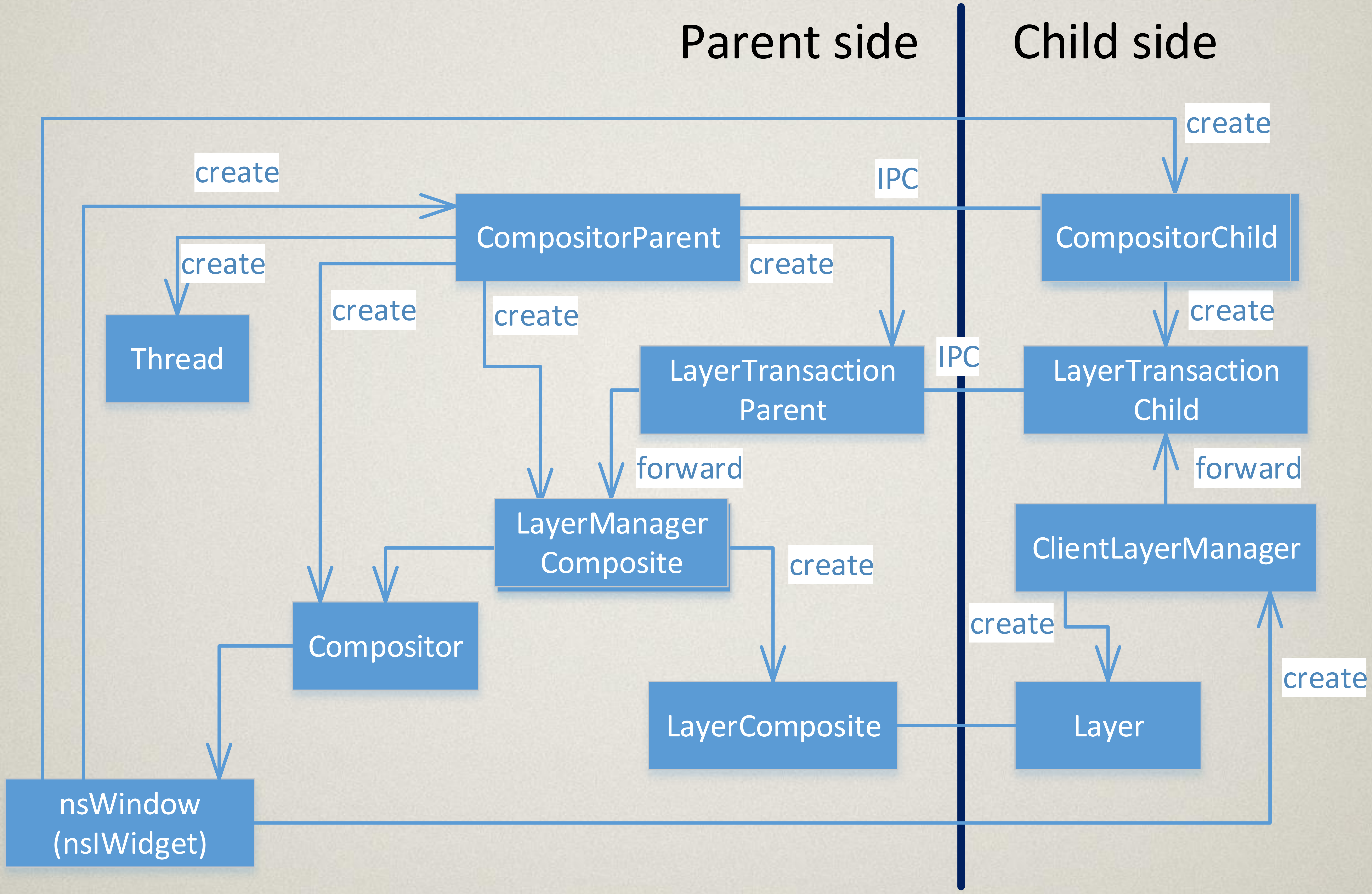
GrallocTextureData/GrallocTextureHostOGL



GrallocTextureData/GrallocTextureHostOGL

- Gralloc (android::GraphicBuffer) is shared between GrallocTextureData and GrallocTextureHostOGL
- gralloc could be directly bounded to Open GL texture by using EGLImage
- SharedBufferManagerParent allocates gralloc in chrome process and delivers it to child side
- SharedBufferManagerParent/Child pairs is allocated for chrome process and for each content process
- SharedBufferManagerParent/Child owns a thread to handle gralloc allocation, since the allocation takes a long time

Off-main-thread compositing (OMTC)

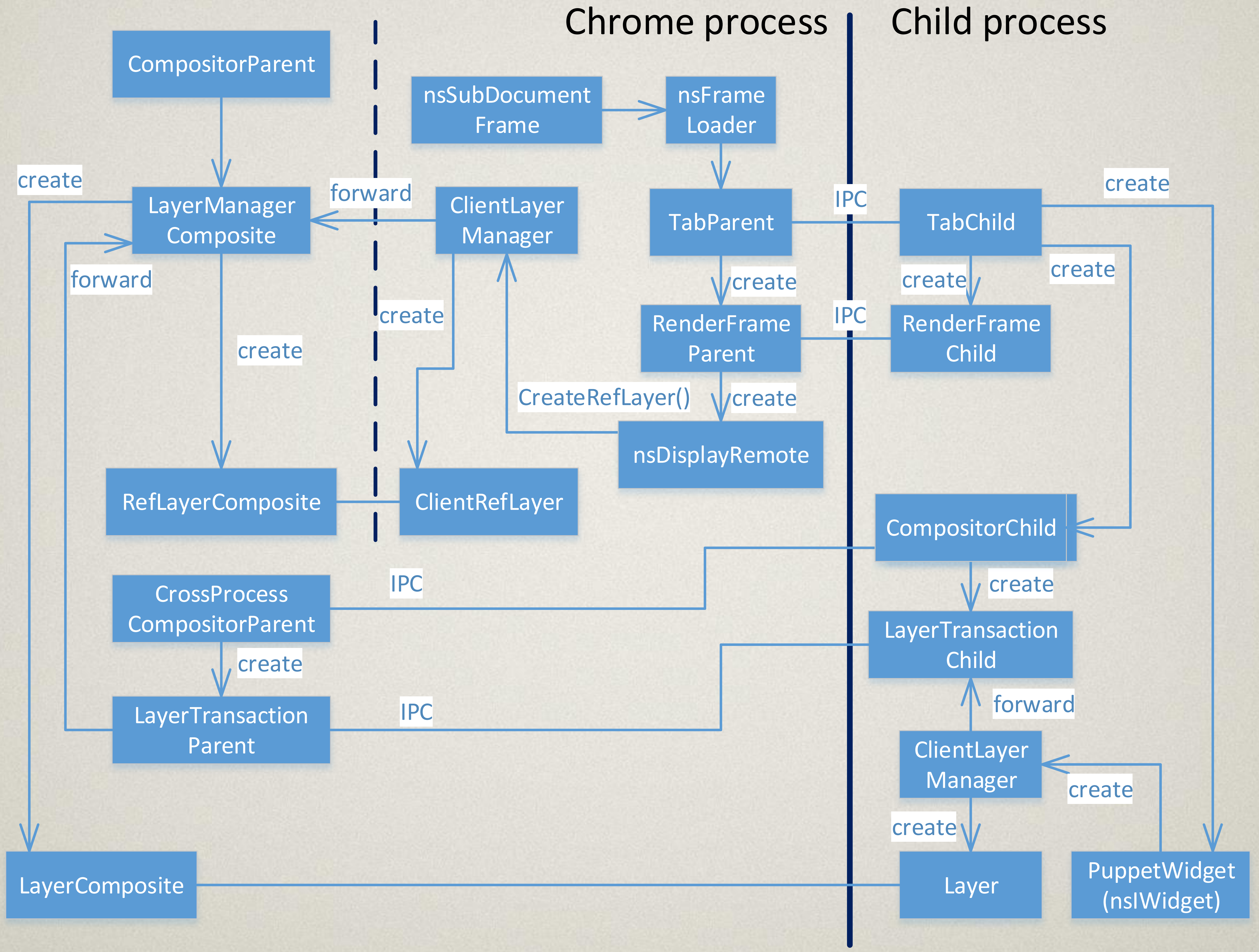


Off-main-thread compositing (OMTC)

CompositorParent creates a thread for compositing(compositor thread) and run compositing tasks and related IPC on the thread

- PCompositor protocol
  - Used to manage communication between the main thread (CompositorChild side) and the compositor thread (CompositorParent side). It's primary purpose is to manage the PLayerTransaction sub protocol
- PLayerTransaction protocol
  - Atomically publishes layer subtrees from main thread (LayerTransactionChild side) to a "shadow“(LayerTransactionParent side) and atomically updating a published subtree. ("Atomic" in this sense is wrt painting)

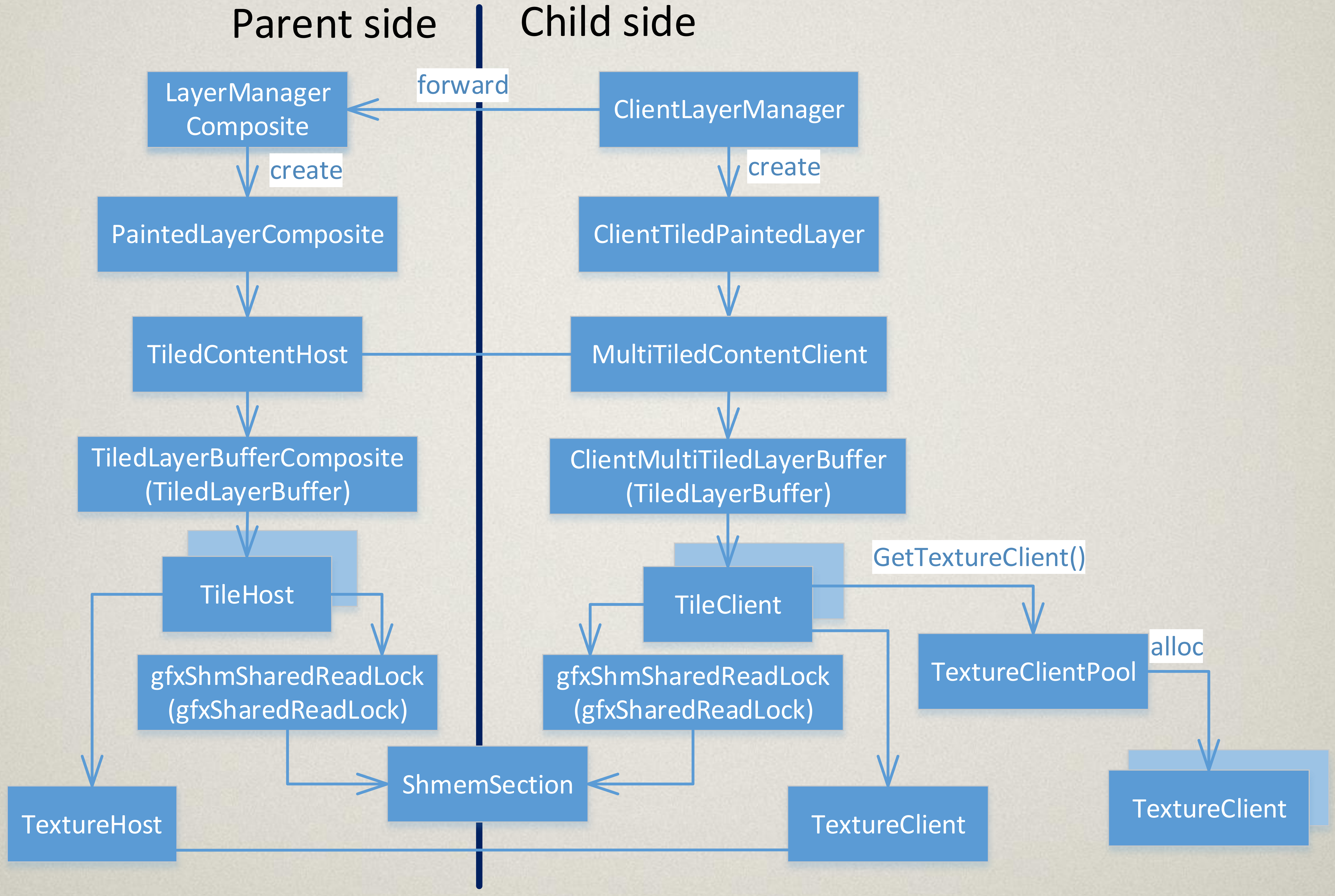
Compositing of child process layers



Compositing of child process layers

- When a document is loaded in child process (like by <iframe>), layers are created in the process
- nsSubDocumentFrame could be created by several tags like <iframe> by NS_NewSubDocumentFrame()
- PRenderFrame (in the layout sense of "frame") represents one web "page". It's used to graft content processes' layer trees into chrome's rendering path. RenderFrameParent is for chrome side
- ClientRefLayer and RefLayerComposite are created for the RenderFrameParent. RefLayer refers to a layer tree in child process through an ID.
- LayerComposites of the layer trees are created by LayerManagerComposite via CrossProcessCompositorParent
- The LayerComposites are connected to RefLayerComposite only during compositing by using AutoResolveRefLayers

PaintedLayer (scrollable)

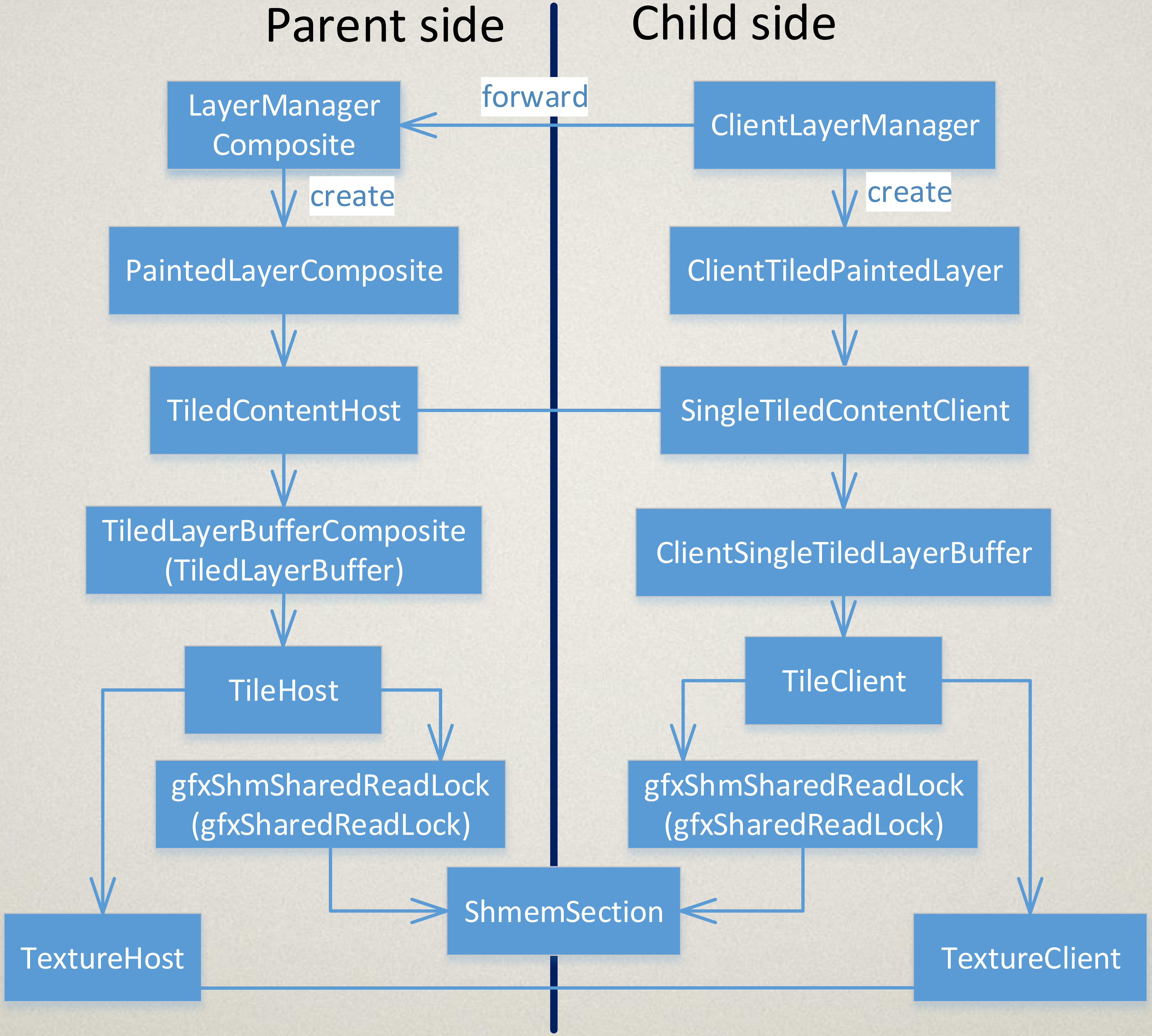


PaintedLayer (scrollable)

ClientTiledPaintedLayer is always created as PaintedLayer on gonk. If ClientTiledPaintedLayer is scrollable, MultiTiledContentClient is created as CompositableClient. Tile size is choosen so that there are between 2 and 4 tiles per screen width (tile size max: 1024)

- ClientTiledPaintedLayer
  - An implementation of PaintedLayer that only supports remote composition that is backed by tiles
- MultiTiledContentClient
  - An implementation of TiledContentClient that supports multiple tiles and a low precision buffer
- TileClient
  - Represent a single tile in tiled buffer. The buffer keeps tiles, each tile keeps a reference to a texture client and a read-lock. This read-lock is used to help implement a copy-on-write mechanism
- TextureClientPool
  - Caches TextureClients to improve performance
  - Used by TileClient

PaintedLayer (non scrollable)

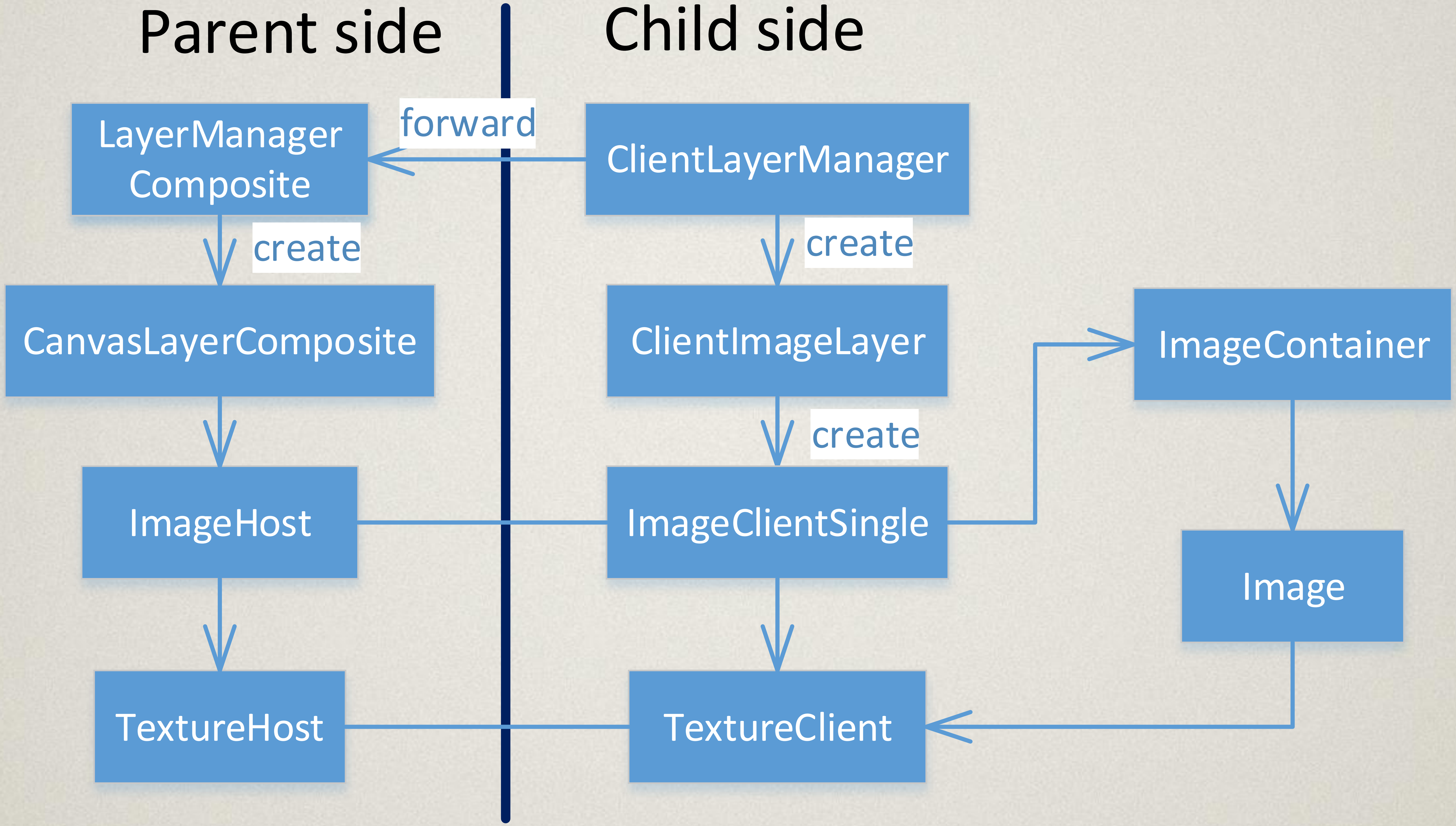

Parent side
Child side
LayerManager Composite
forward
ClientLayerManager
create
create
PaintedLayerComposite
ClientTiledPaintedLayer
TiledContentHost
SingleTiledContentClient
TiledLayerBufferComposite (TiledLayerBuffer)
ClientSingleTiledLayerBuffer
TileHost
TileClient
gfxShmSharedReadLock (gfxSharedReadLock)
gfxShmSharedReadLock (gfxSharedReadLock)
ShmemSection
TextureHost
TextureClient

PaintedLayer (non scrollable)

If ClientTiledPaintedLayer is non scrollable, SingleTiledContentClient is created as CompositableClient. Host side works same to MultiTiledContentClient

- SingleTiledContentClient
  - Allocate one tile for whole PaintedLayer
  - Does not use TextureClientPool

ImageLayer (synchronous)



Parent side

Child side

LayerManager Composite

forward

ClientLayerManager

create

CanvasLayerComposite

ClientImageLayer

create

ImageContainer

ImageHost

ImageClientSingle

Image

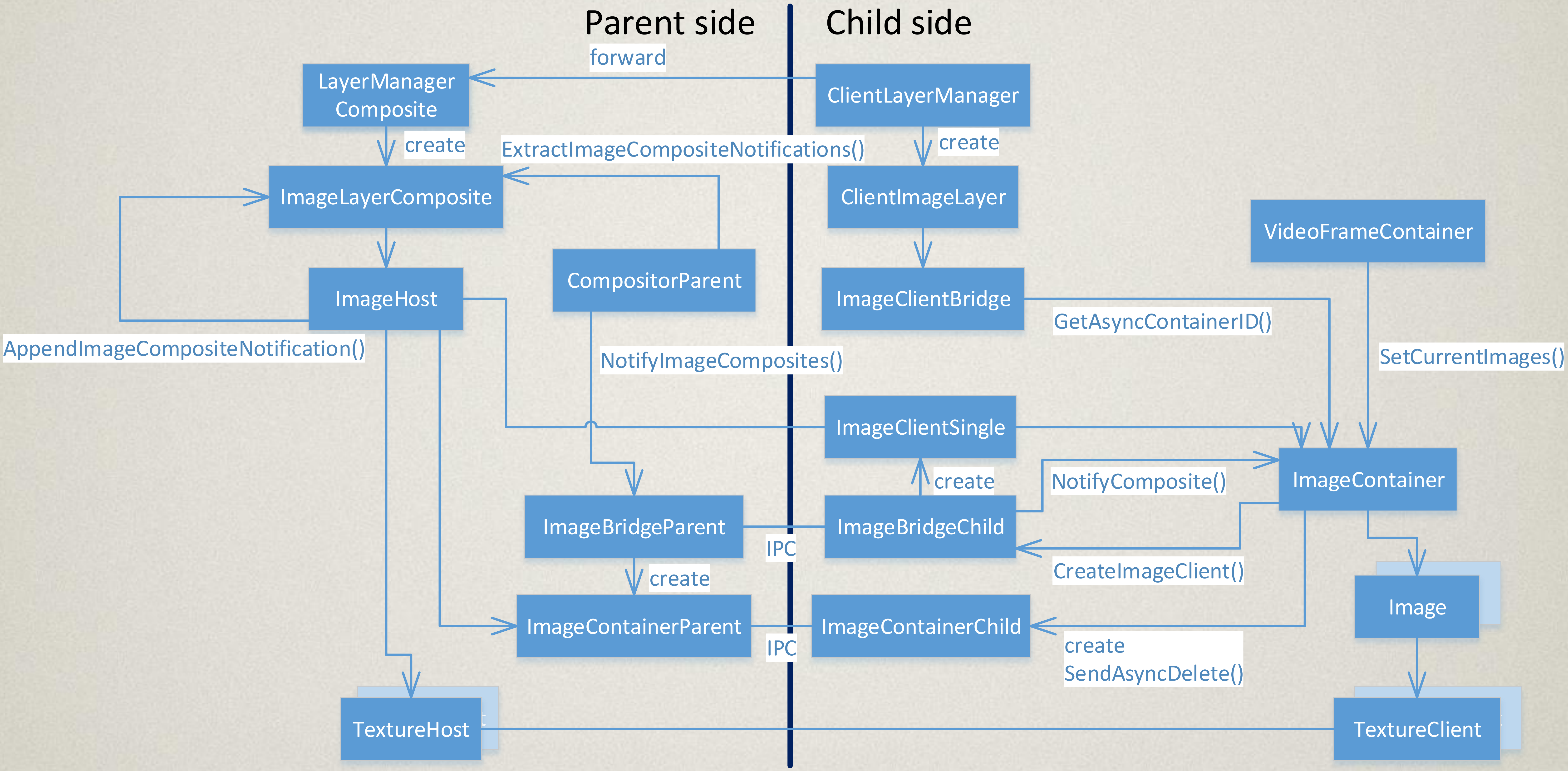
TextureHost

TextureClient

ImageLayer (synchronous)

- On Firefox OS, all ImageContainers uses synchronous mode except video frame renderings.
- Delivers images to Compositor side via main thread
- In the majority of cases, ClientImageLayer is not used for image rendering. Instead, they are painted into PaintedLayer
- ClientImageLayers are typically created for animated images

ImageLayer (asynchronous)



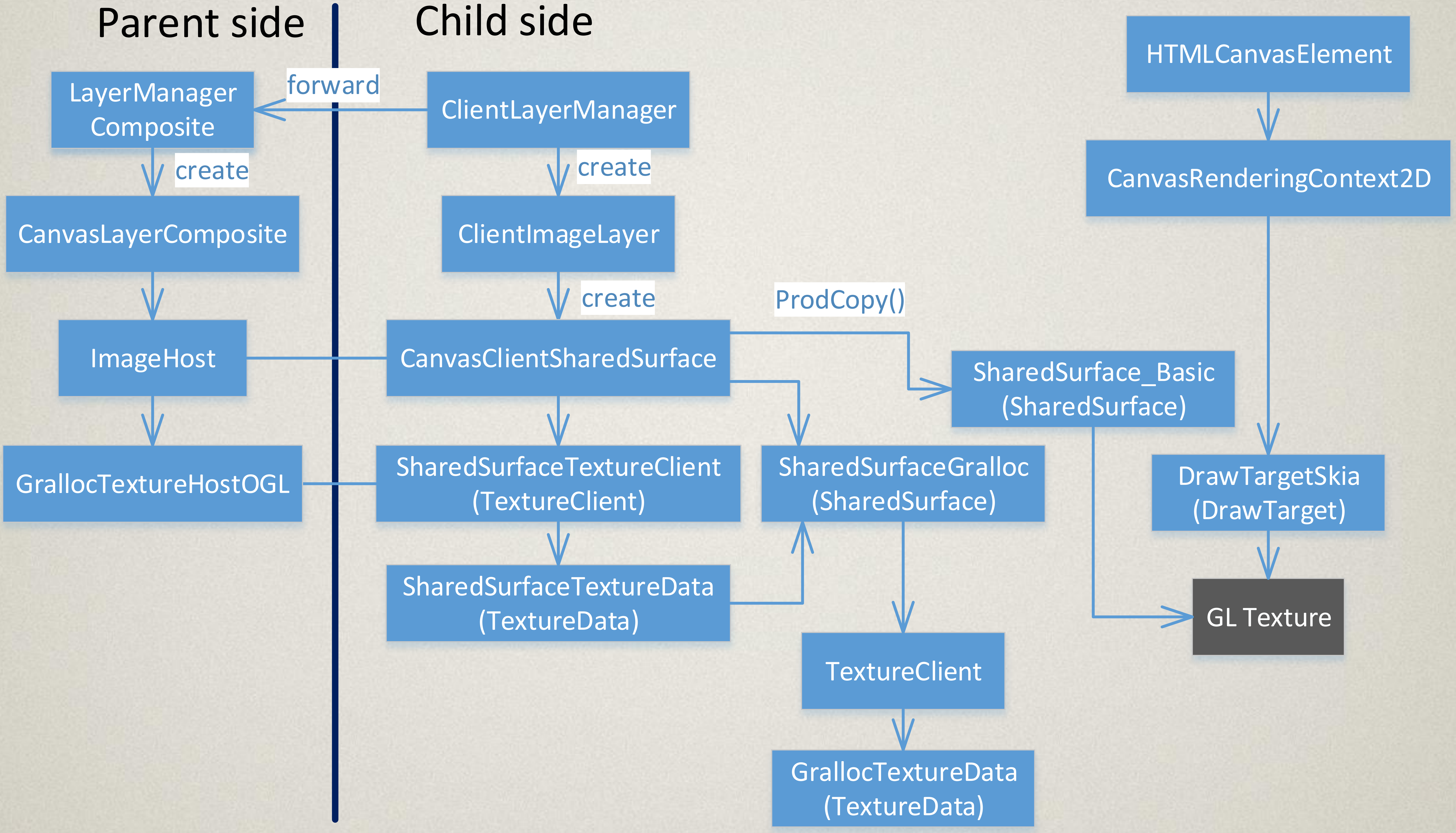
ImageLayer (synchronous)

- On Firefox OS, asynchronous mode of ImageComtainer is used for video frames renderings. TextureClients are delivered to Compositor side via ImageBridge thread

- To reduce video frames drops, ImageContainer, ImageClientSingle and ImageHost could hold multiple images with timestamps(Bug 1143575). ImageHost renders Images based on timestamps.

- Image
  - Represents a buffer of pixel data. The data can be in one of various formats including YCbCr
  - If the buffer is not TextureClient, its content is copied to TextureClient at ImageClientSingle
- PImageBridge protocol
  - Used to allow isolated threads or processes to push frames directly to the compositor thread/process without relying on the main thread which might be too busy dealing with content script.
  - ImageBridgeChild is child side object that run on ImageBridge thread. ImageBridgeParent is parent side object that run on Compositor thread

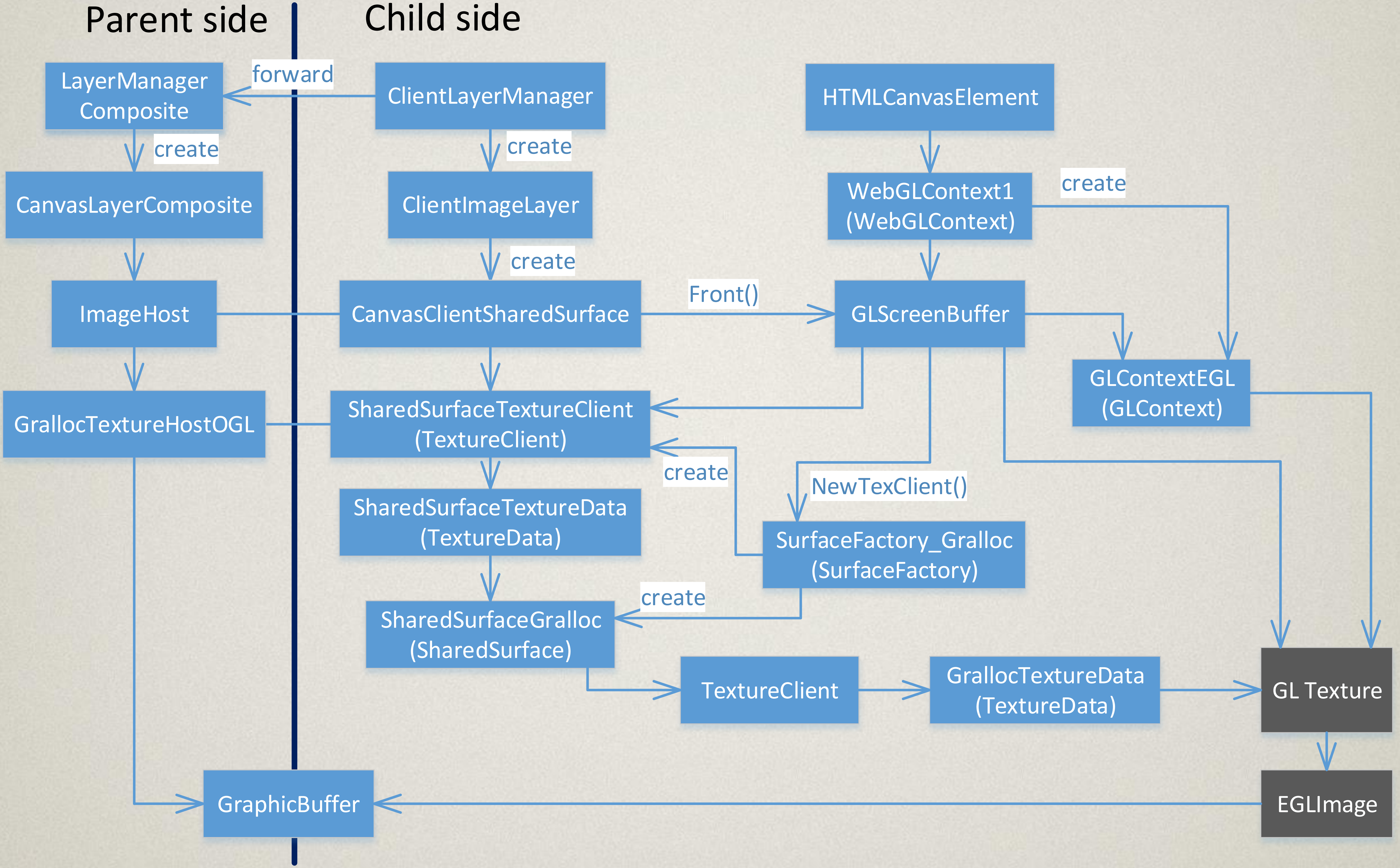
2D Canvas (with GPU)


Parent side
Child side
LayerManager Composite
forward
ClientLayerManager
create
create
CanvasLayerComposite
ClientImageLayer
create
ProdCopy()
ImageHost
CanvasClientSharedSurface
SharedSurface_Basic (SharedSurface)
GrallocTextureHostOGL
SharedSurfaceTextureClient (TextureClient)
SharedSurfaceGralloc (SharedSurface)
SharedSurfaceTextureData (TextureData)
TextureClient
GrallocTextureData (TextureData)
HTMLCanvasElement
CanvasRenderingContext2D
DrawTargetSkia (DrawTarget)
GL Texture

2D Canvas (with GPU)

- CanvasRenderingContext2D renders to DrawTargetSkia. It actually renders to OpenGL texture
- CanvasClientSharedSurface gets the OpenGL texture as SharedSurface_Basic
- The texture is copied from SharedSurface_Basic to SharedSurfaceGralloc by using GPU
- SharedSurfaceGralloc is wrapped by SharedSurfaceTextureClient. But SharedSurfaceGralloc wraps TextureClient. It is sad. It might be changed in future

WebGL


Parent side
Child side
LayerManager Composite
forward
ClientLayerManager
HTMLCanvasElement
create
create
CanvasLayerComposite
ClientImageLayer
WebGLContext1 (WebGLContext)
create
create
ImageHost
CanvasClientSharedSurface
Front()
GLScreenBuffer
GLContextEGL (GLContext)
GrallocTextureHostOGL
SharedSurfaceTextureClient (TextureClient)
create
NewTexClient()
SharedSurfaceTextureData (TextureData)
SurfaceFactory_Gralloc (SurfaceFactory)
create
SharedSurfaceGralloc (SharedSurface)
TextureClient
GrallocTextureData (TextureData)
GL Texture
EGLImage
GraphicBuffer

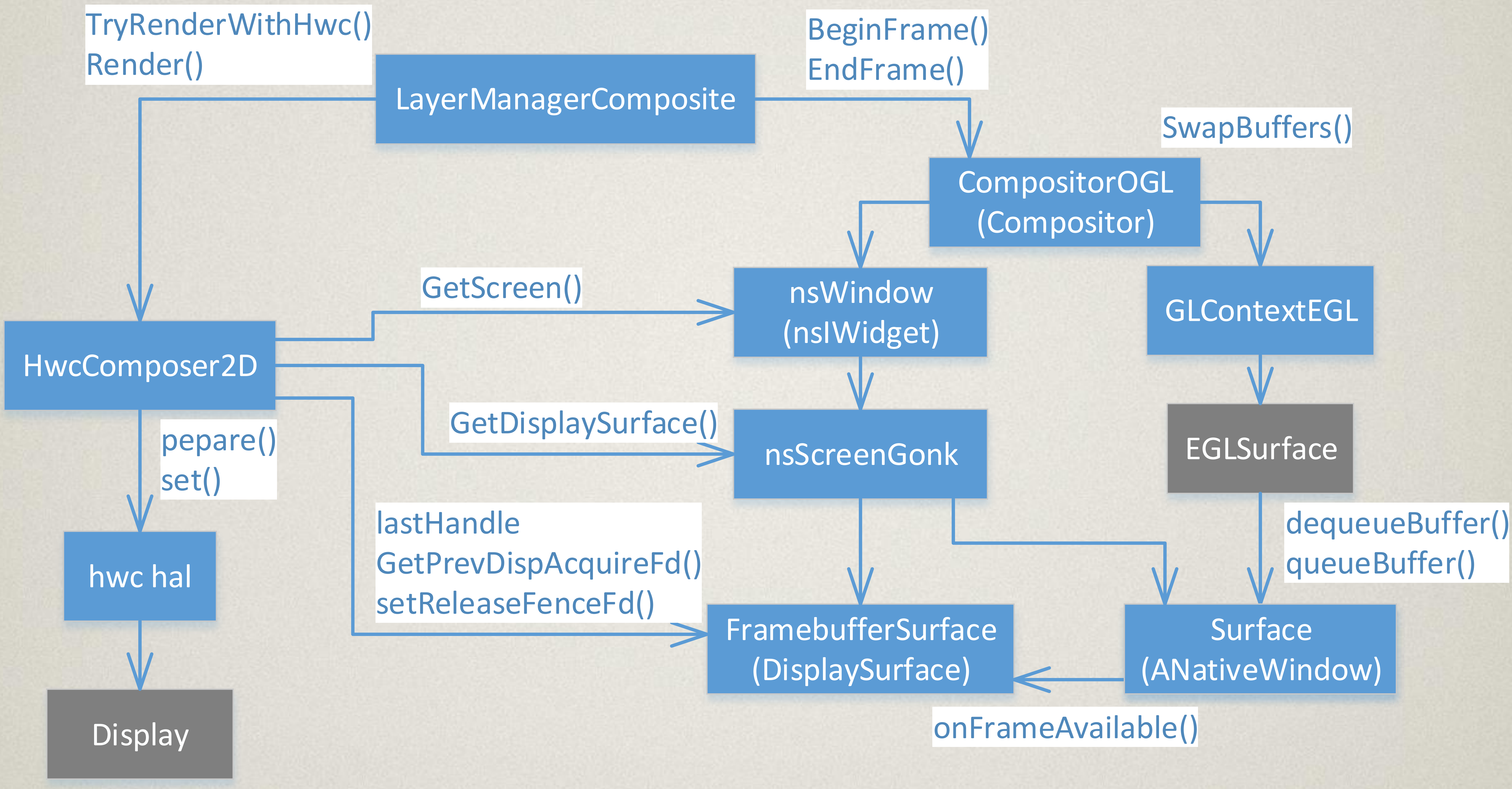
WebGL

- WebGLContext renders to GLScreenBuffer
- CanvasClientSharedSurface gets front buffer(SharedSurfaceTextureClient) by GLScreenBuffer::Front()
- The SharedSurfaceTextureClient is sent to Compositor side for composition

- GLScreenBuffer
  - Abstraction for the "default framebuffer" used by an offscreen GLContext
  - Rendering is published as SharedSurfaceTextureClient by GLScreenBuffer::PublishFrame()

- OffscreenCanvas is wip
  - Bug 709490, Bug 801176, Bug 1172796, Bug 1203382
  - https://wiki.whatwg.org/wiki/OffscreenCanvas

Render to screen



Render to screen(with CompositorOGL)

- On gonk, Compositing and "render to screen" are different steps. Hwc (hardware compooser) does "render to screen". Hwc also supports 2d compositing for limited number of layers
  - https://source.android.com/devices/graphics/architecture.html#hwcomposer
- When compositing is done only by CompositorOGL, resultant buffers are delivered to FrameBufferSurface via ANativeWindow
- When Composer2D::Render() is called, Composer2D gets the buffer from FrameBufferSurface and renders it to screen via hwc hal.

Types of Compositing on gonk

Compositing by using Compositor is not the only compositing on gonk. LayerManagerComposite::Render() tries ‘Full hwc composition’ at first. If it failed, then fallbacks to ‘ using OpenGL for compositing’. The fallback could be ‘OpenGL compositing’ or ‘Hwc and OpenGL mixed compositing’

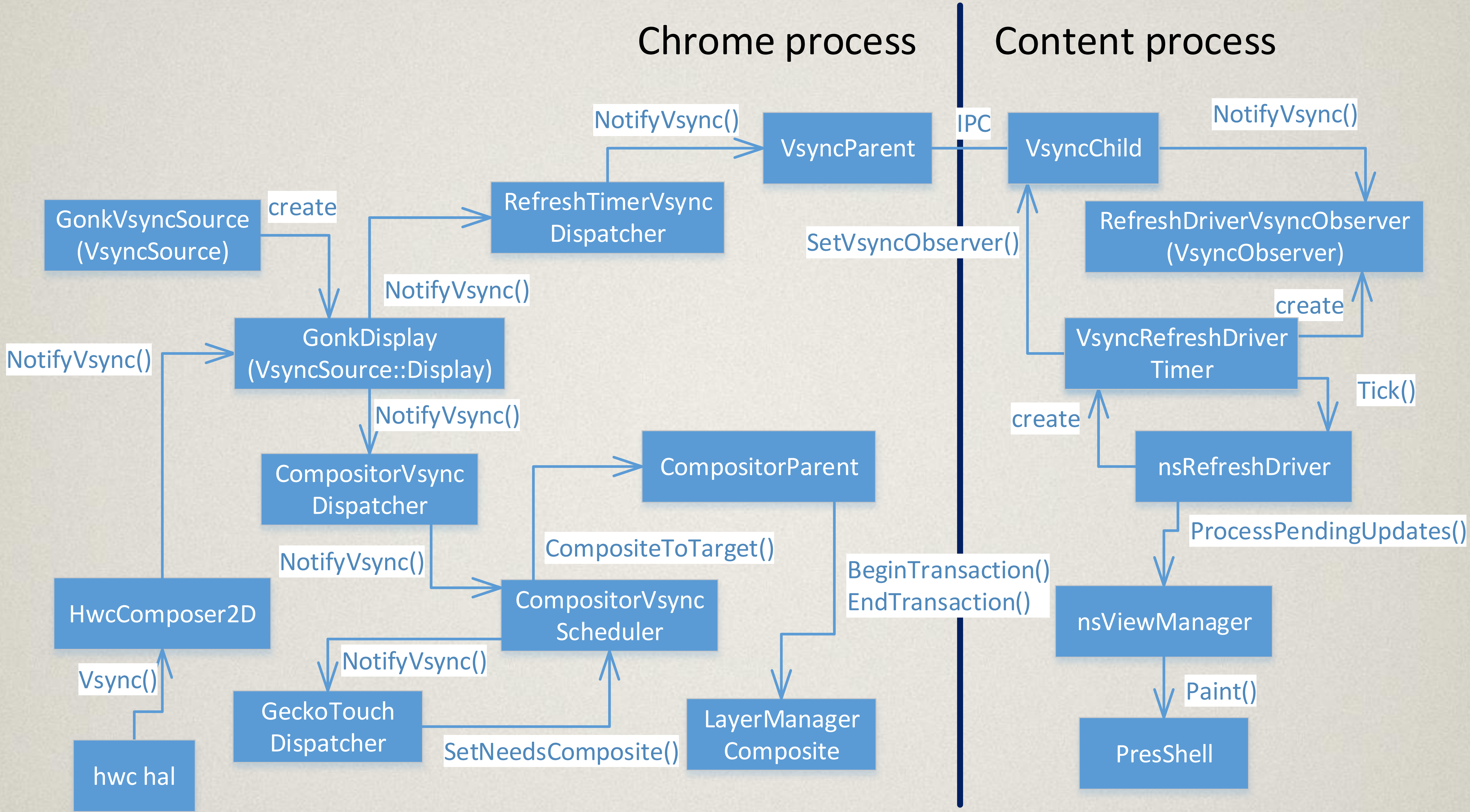
- Full hwc compositing
  - All layers are composed by hwc. Open GL composition does not happen
- Open GL Compositing
  - All layers are composed by Open GL
  - Hwc just renders the composed result
- Hwc and OpenGL mixed compositing
  - Some layers are composed by hwc and another layers are composed by OpenGL
  - Supported only by overlay hwc
  - Uses it mainly for performance and power consumption especially of video rendering
  - Android uses it also for rendering DRM protected contents. Firefox OS does not supports it. It is going to be added to Firefox OS(Bug 1049296)

Types of HWC

- Copybit HWC
  - Majority of Firefox OS mobile products use it
  - Use copybit hw for composition
  - Does not have much power consumption advantage to OpenGL. If over drawing exceed a specific level(150%), it fallbacks to OpenGL composition
  - Can handle multiple rectangles for each layer
  - Does not support OpenGL+HWC mixed composition
- Overlay HWC
  - Fx0 use it
  - Better performance
  - Not good at handling multiple rectangles in a layer
  - Support OpenGL+HWC mixed composition

mozilla

VsyncSource



https://wiki.mozilla.org/Project_Silk

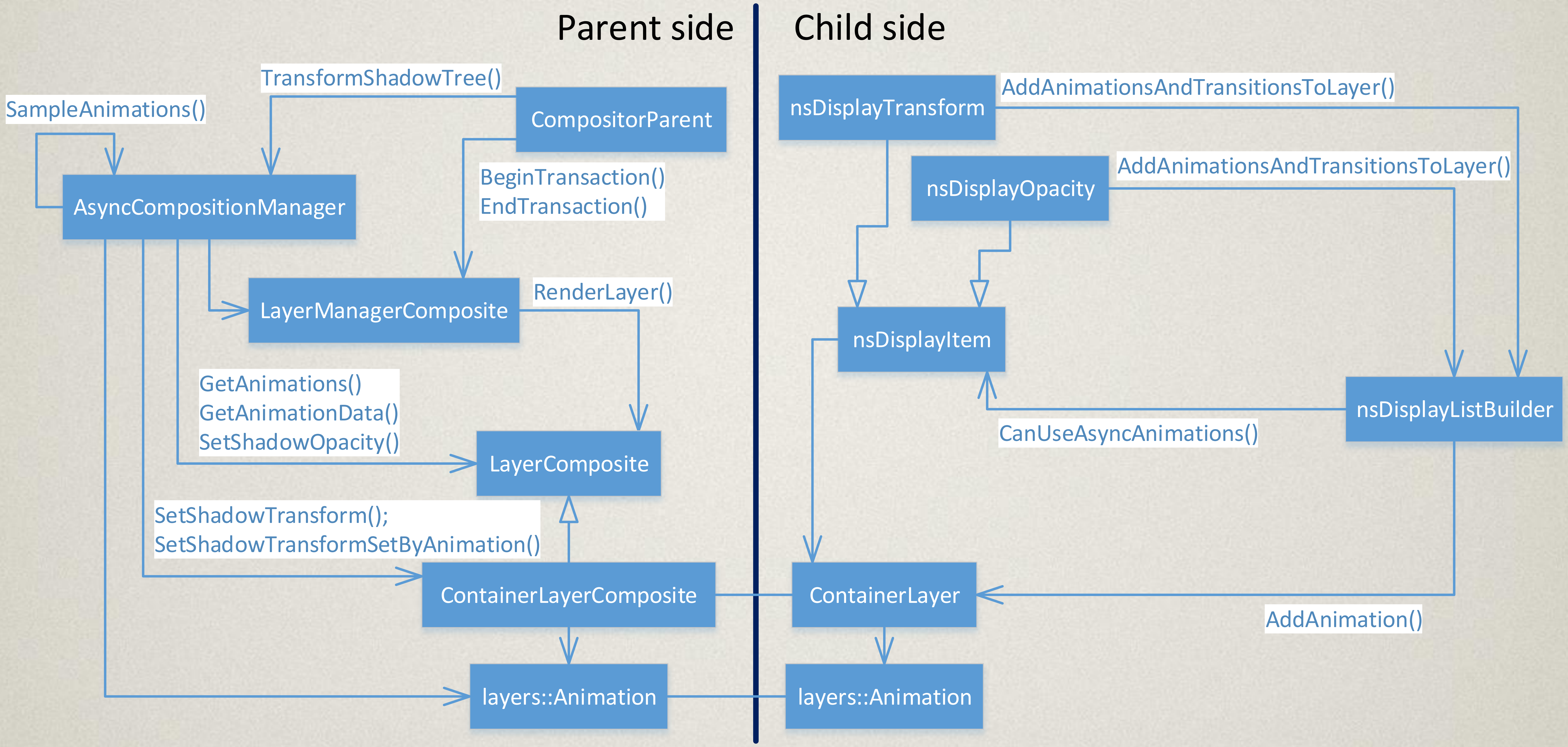
VsyncSource

- VsyncSource is used to deliver vsync timing to CompositorVsyncScheduler, nsRefreshDriver and GeckoTouchDispatcher
- CompositorVsyncScheduler schedules CompositorParent's compositing
- nsRefreshDriver schedules PresShell's painting or other layout’s tasks like requestAnimationFrame(rAF).
- GeckoTouchDispatcher resamples touch events whenever a vsync event occurs for smooth scrolling
- GonkVsyncSource uses hwc hal's vsync callback as vsync source. If it is disabled, SoftwareVsyncSource is used.

Off-main-thread Animation(OMTA)

- Allows css animation to be performed asynchronously (on the compositor thread rather than the main thread)
- Also referred as “Async Animations”
- Only supports opacity and transform of animations
- Implemented by Bug 768440 and Bug 1166173

Off-main-thread Animation(OMTA)



Off-main-thread Animation(OMTA)

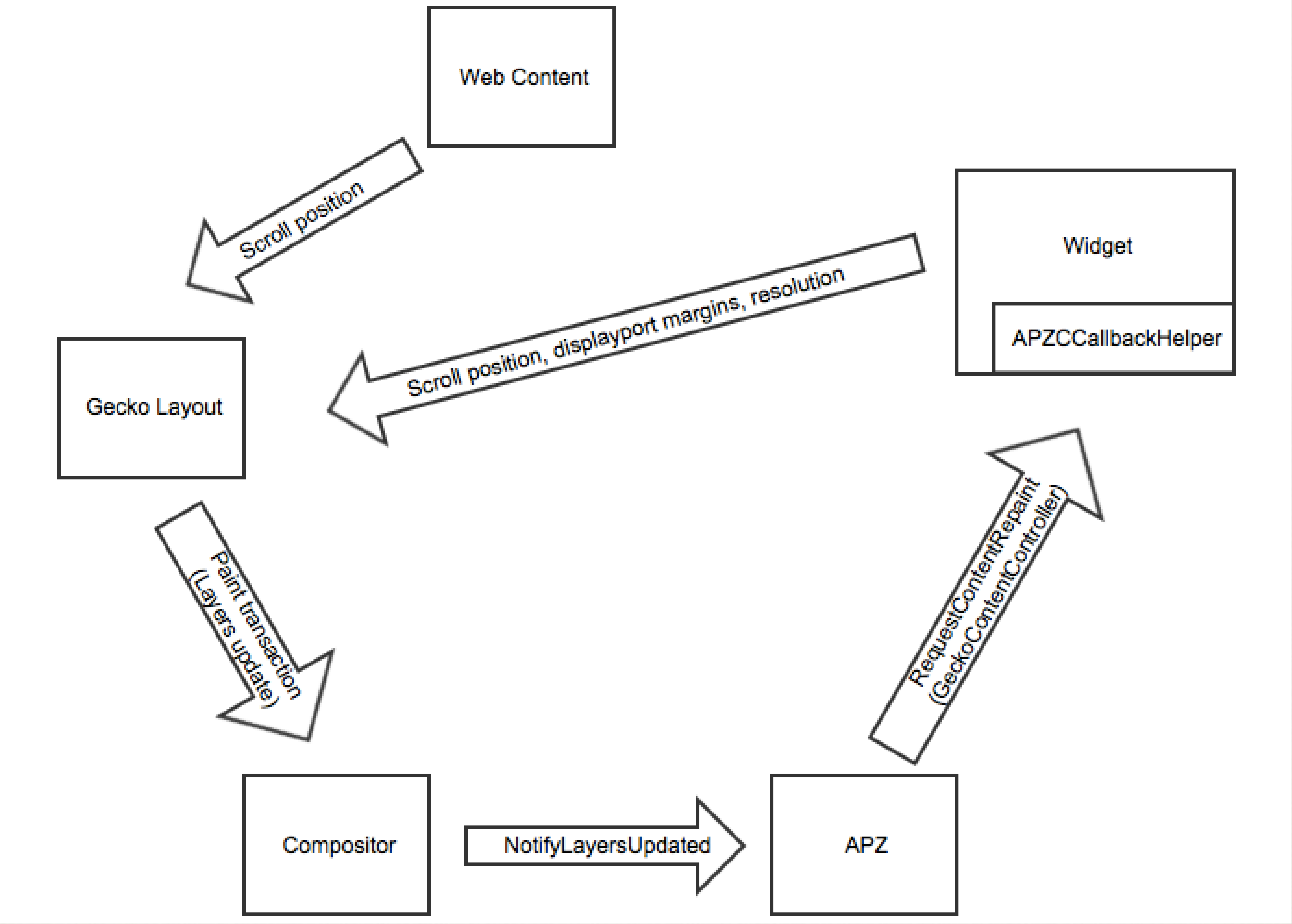
- AddAnimationsAndTransitionsToLayer() adds layers::Animation to Layers. Only nsDisplayTransform and nsDisplayOpacity calls it among nsDisplayItems
- layers::Animation is delivered to LayerComposite
- Just before compositing, CompositorParent calls AsyncCompositionManager::TransformShadowTree(). It transforms LayerComposite tree based on layers::Animation
- If more Composition is necessary to update animation, CompositorParent schedules next compositing

Async Pan/Zoom(APZ)

- Allows panning and zooming to be performed asynchronously (on the compositor thread rather than the main thead).
- Reference
  - https://dxr.mozilla.org/mozilla-central/source/gfx/doc/AsyncPanZoom-HighLevel.png
  - https://github.com/mozilla/gecko-dev/blob/master/gfx/doc/AsyncPanZoom.md
  - https://wiki.mozilla.org/Platform/GFX/APZ
  - https://wiki.mozilla.org/Mobile/AsyncSubframePanning
  - https://staktrace.com/spout/entry.php?id=834

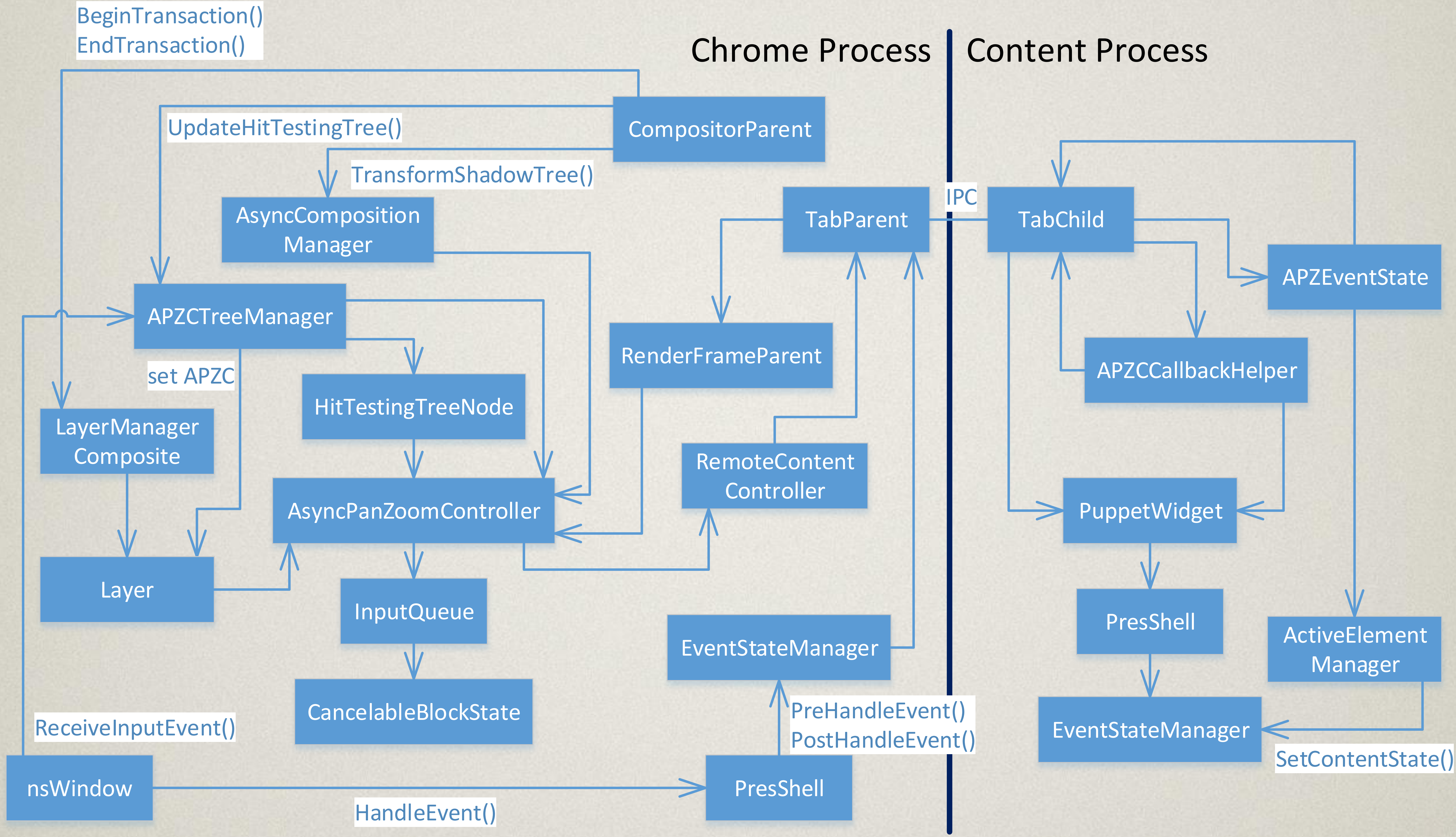
mozilla

Async Pan/Zoom(APZ)


Web Content
Scroll position
Widget
Scroll position, displayport margins, resolution
APZCCallbackHelper
Gecko Layout
Paint transaction (Layers update)
RequestContentRepaint (GeckoContentController)
Compositor
NotifyLayersUpdated
APZ

https://dxr.mozilla.org/mozilla-central/source/gfx/doc/AsyncPanZoom-HighLevel.png

Async Pan/Zoom(APZ)



Async Pan/Zoom(APZ)

- When layer transactions are receved at LayerTransactionParent, CompositorParent requests APZCTreeManager to update hit-testing tree based on the layer update by APZCTreeManager::UpdateHitTestingTree()
- nsWindow at first, sends an input event to APZCTreeManager by APZCTreeManager::ReceiveInputEvent(). It manipulates frame metrics of AsyncPanZoomControllers based on what type of input it is
- If the APZCTreeManager says to drop it, then nsWindow drops it
- If the APZCTreeManager does not say to drop it, nsWindow sends it to normal event delivery route (to PresShell)
- Like OMTA, AsyncCompositionManager::TransformShadowTree() transforms LayerComposite(Layer) tree based on AsyncPanZoomController that is set to the Layer
- AsyncPanZoomController requests content repaint via RemoteContentController

Q&A?