MAXIMIZING THE PERFORMANCE OF HTML5 VIDEO IN RPI2

Gwang Yoon Hwang

yoon@igalia.com

WHAT IS THE HTML5 VIDEO?

CSS

```
body {
  padding: 0;
  margin: 0;
  width: 100%;
  height: 100%;
  overflow: hidden;
}

#framerate {
...
```

JavaScript

```
  g.mvpMatrix.load(g.perspectiveMatrix);
  g.mvpMatrix.multiply(g.mvMatrix);
  g.mvpMatrix.setUniform(gl, g.u_modelViewProjMatrixLoc, false);

  gl.bindTexture(gl.TEXTURE_2D, tuxTexture);
  gl.drawElements(gl.TRIANGLES, g.box.numIndices, gl.UNSIGNED_BYTE, 0);
```

HTML

```
<video width="1280" height="720" autoplay loop>
<source src="big_buck_bunny_720p_h264.mov" type="video/mp4" />
</video>
<canvas id="example"></canvas>
<div id="framerate"></div>
```

WEBKIT - A PORTABLE WEB RENDERING ENGINE

- PCs, phones, TVs, IVI, smartwatches, **Raspberry Pi**
- Mac, iOS, EFL, **GTK**

WEBKITGTK+

- Full-featured port of the WebKit rendering engine
- Useful in a wide rage of systems from desktop to embedded

WEBKIT FOR WAYLAND

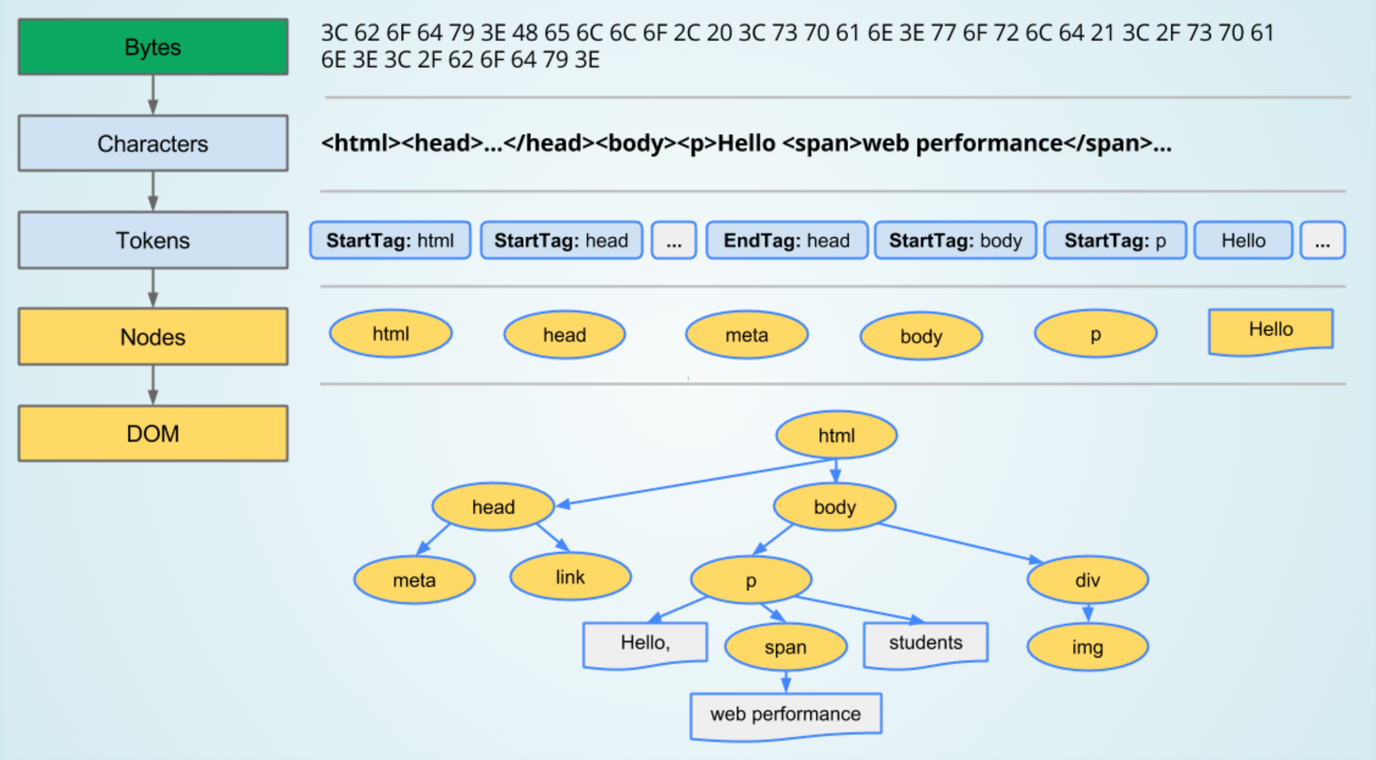
- Avoids using any toolkit
- Defaults to EGL, OpenGL ES for accelerated rendering of Web content
- Optimized for displaying the fullscreen web content -- Youtube TV, Information displays, IVI, STBs..

HOW WEBKIT RENDERS WEBPAGE

STEPS FOR RENDERING

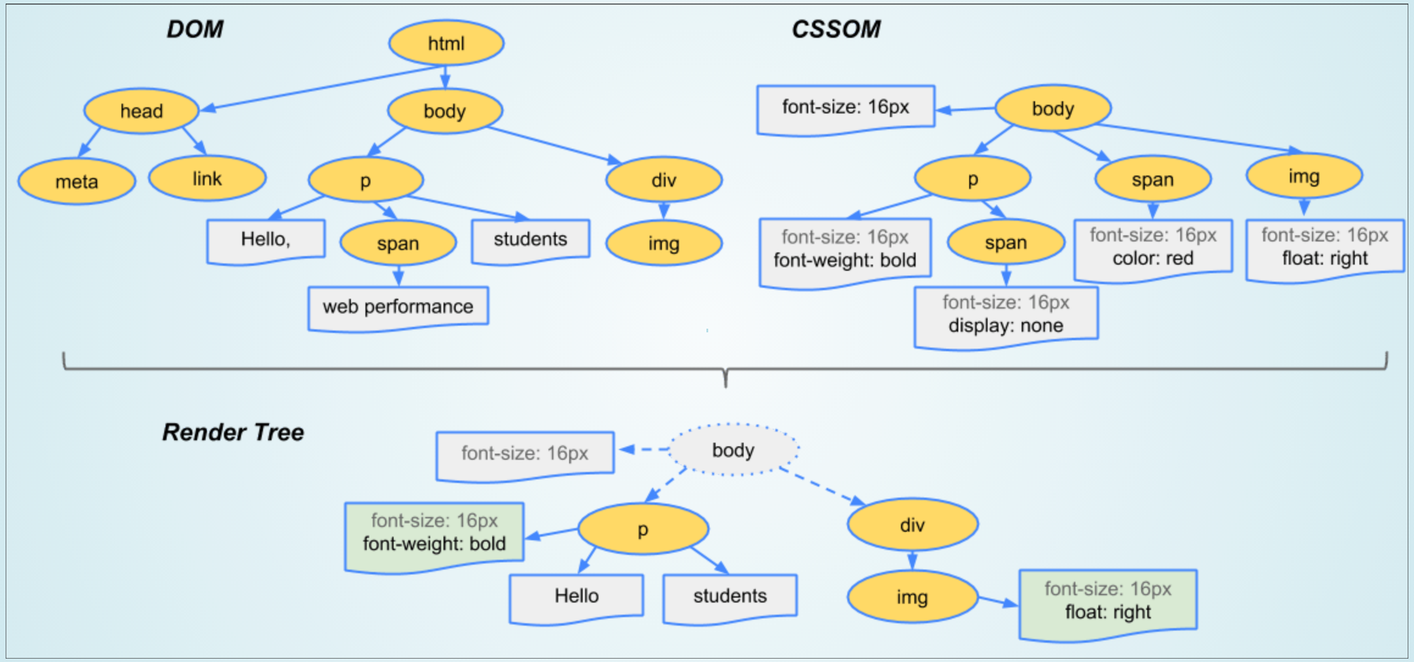
- Parsing: Nodes to DOM Tree
- Constructing RenderObject Tree to RenderLayer Tree

PARSING: CREATES THE DOM TREE FROM SOURCE



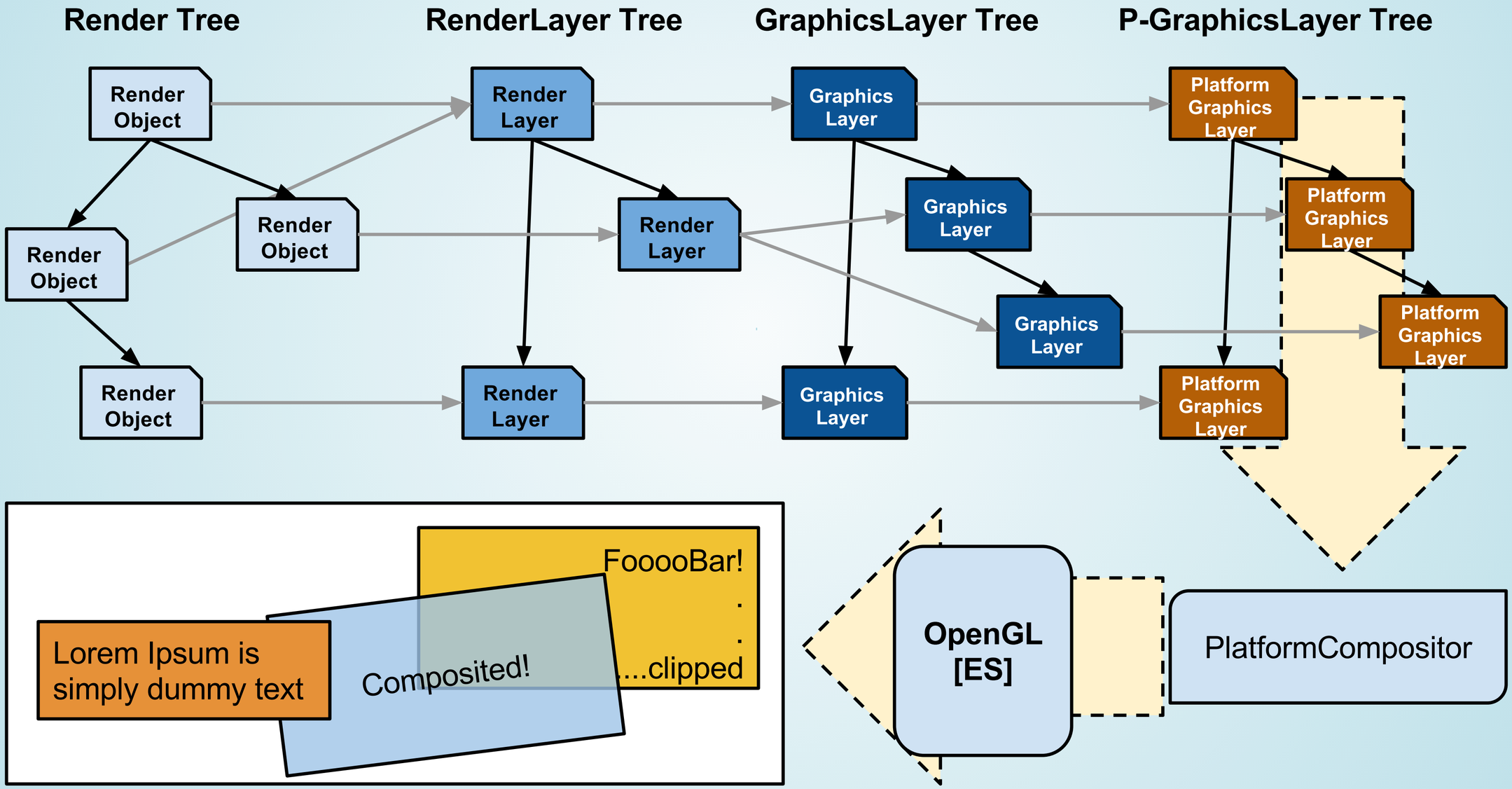
From: https://developers.google.com/web/fundamentals/performance/critical-rendering-path/

CREATING THE RENDER TREE



From: https://developers.google.com/web/fundamentals/performance/critical-rendering-path/

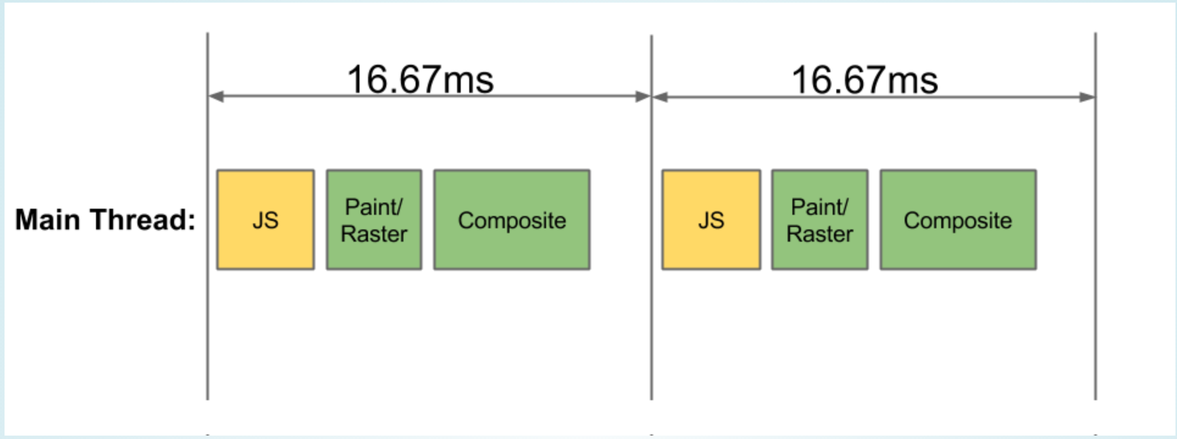
CREATING THE GRAPHICSLAYER TREE AND COMPOSITING IT



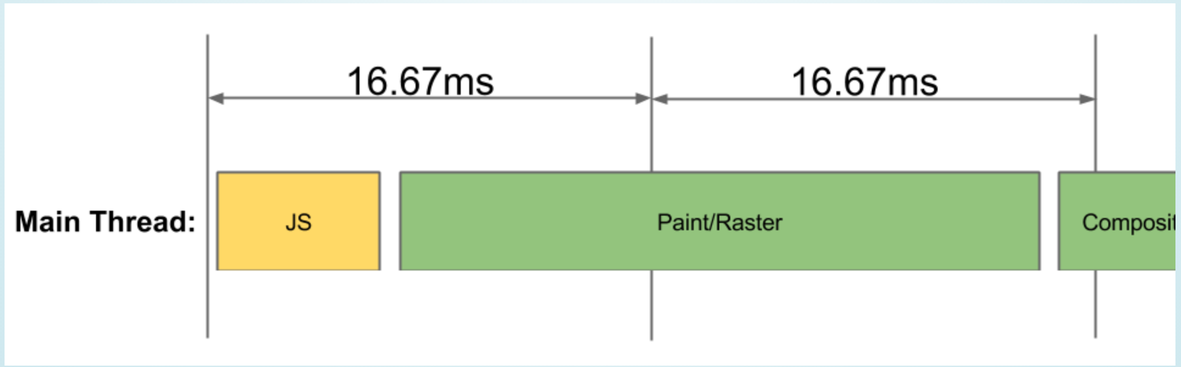
PROBLEMS

- The main-thread is always busy (Parsing, Layout, JS ...)
- The main-thread can be blocked by VSync
- And we want awesome webpages which uses HTML5 Video

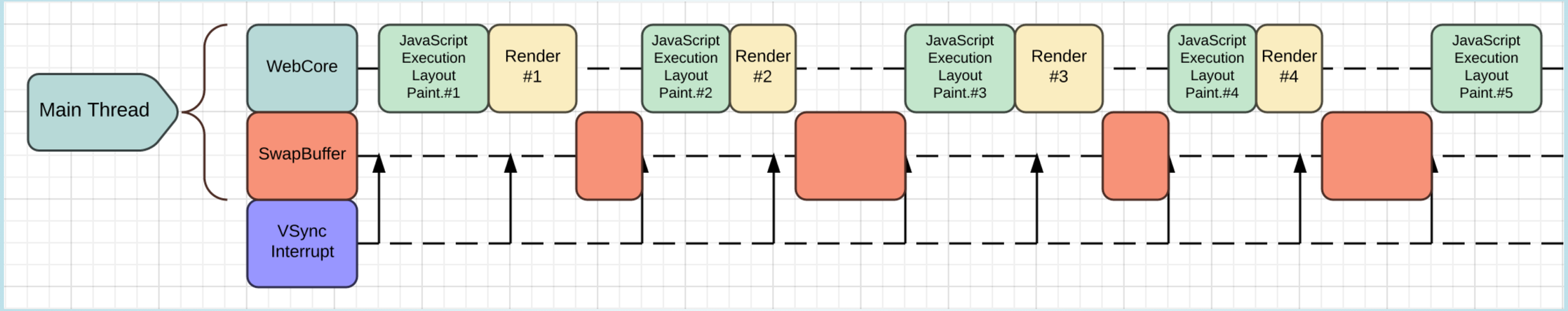
VSYNC : BEST CASE



VSYNC : WORSE CASE



VSYNC : WORST CASE



RASPBERRY PI2

- It is not a Rasberry Pi!
  - A 900MHz quad-core ARM Cortex-A7 CPU
- Why not utilize multi cores?

OFF-THE-MAIN-THREAD COMPOSITING

Main Thread in Web Process

Loading

Parsing

Scripting

Styling

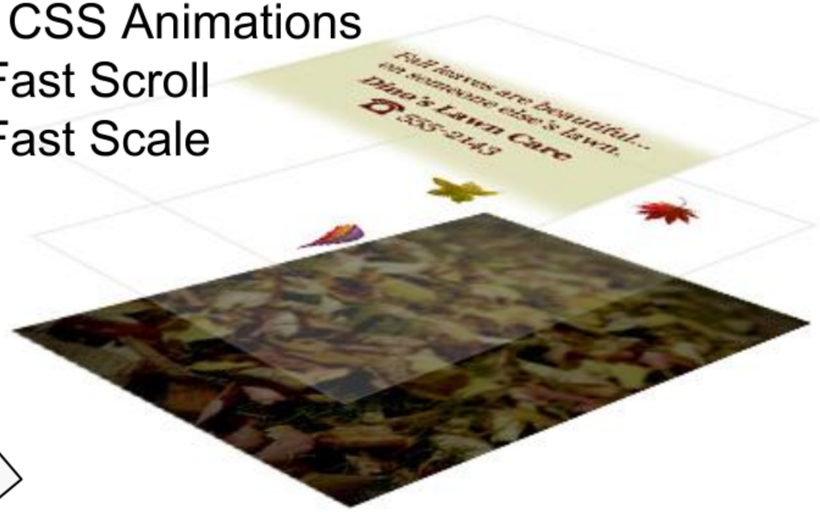
Layout

Compositing Thread or Compositing Process

Running CSS Animations

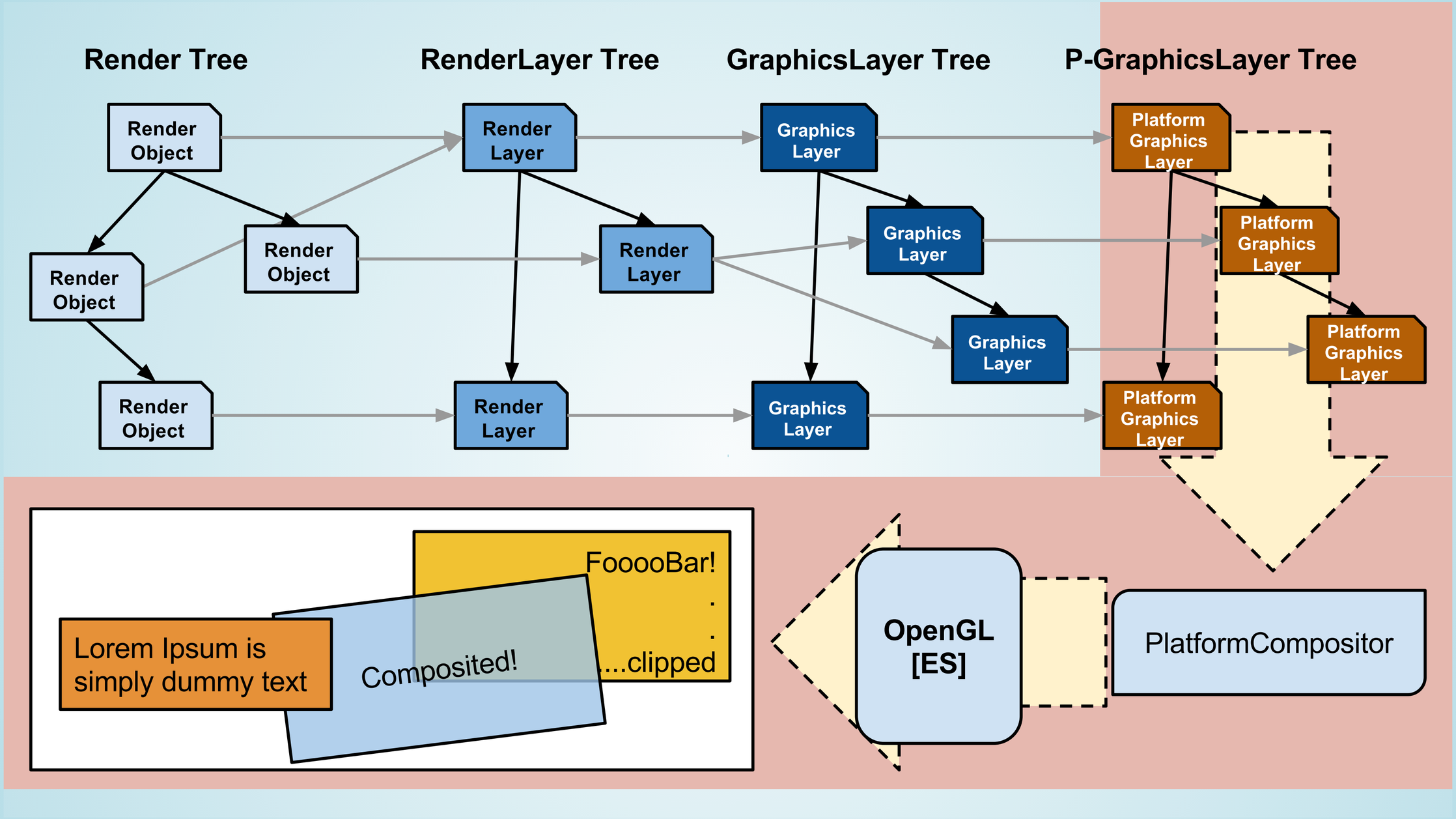
Handle Fast Scroll

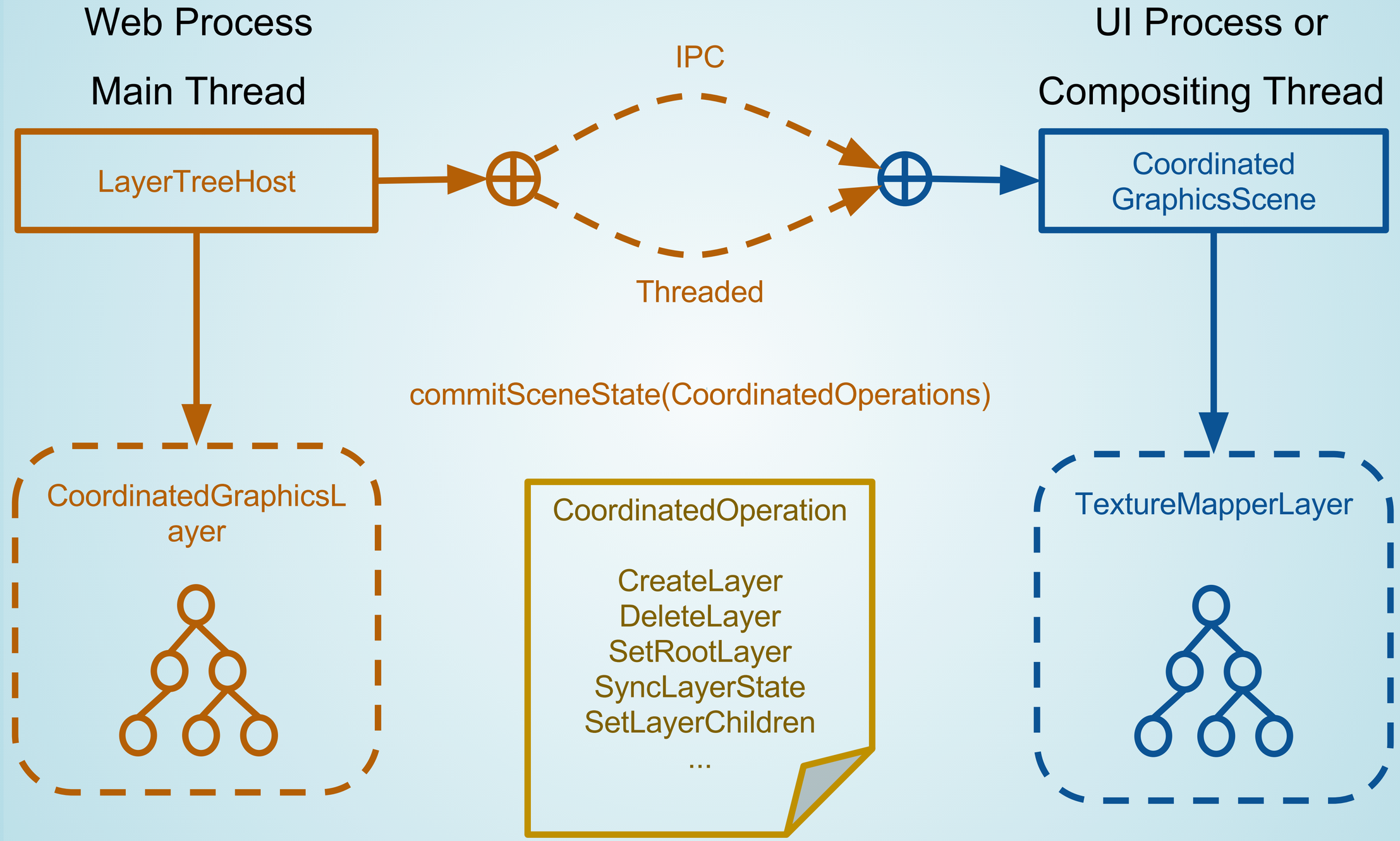
Handle Fast Scale



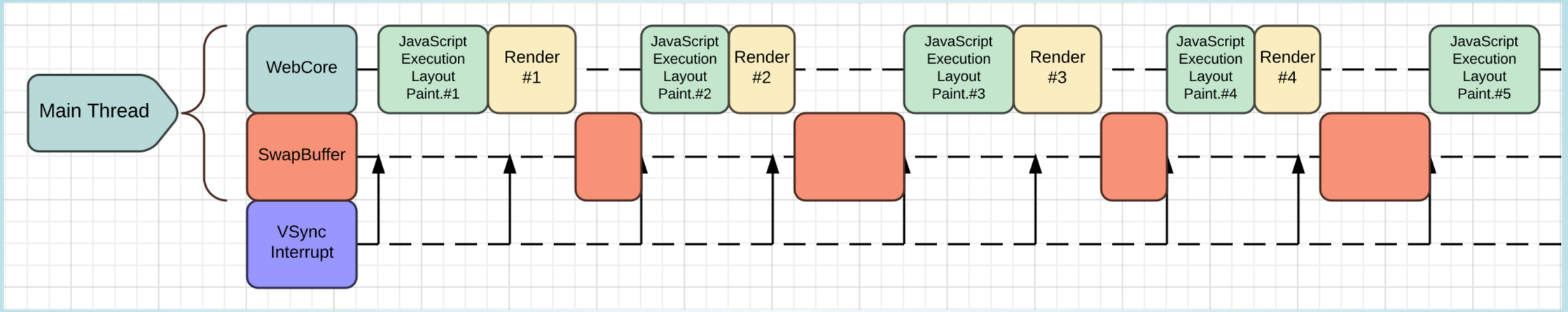
COMPOSITING IN THE DEDICATED THREAD

- Free the main-thread from the Vsync and compositing operations
- It shows more smooth CSS animations, zoom, scrolling, and scale operations.

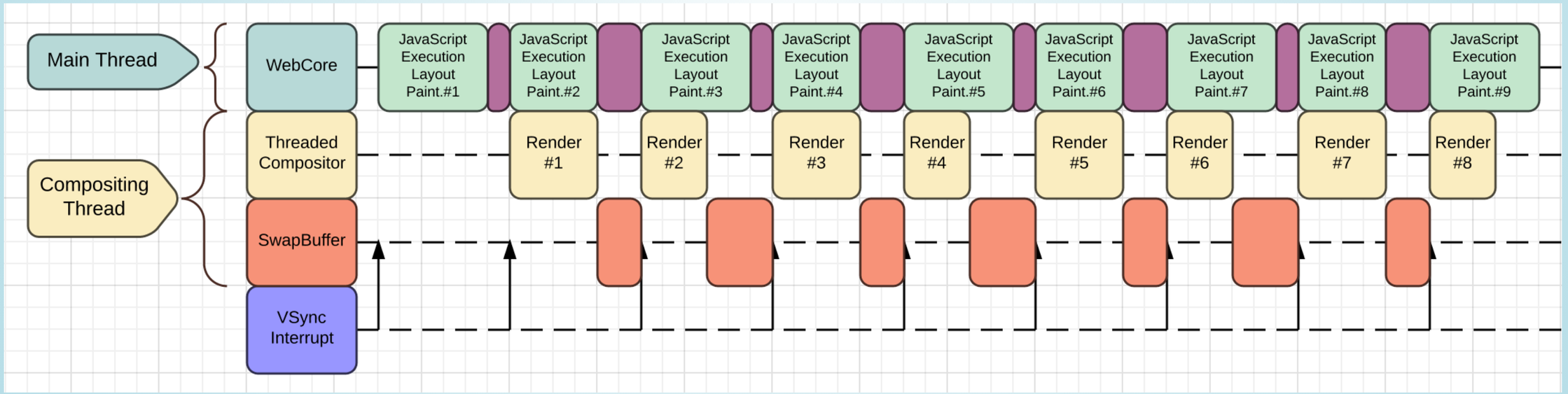

Render Tree
RenderLayer Tree
GraphicsLayer Tree
P-GraphicsLayer Tree
Render Object
Render Object
Render Object
Render Object
Render Layer
Render Layer
Render Layer
Graphics Layer
Graphics Layer
Graphics Layer
Graphics Layer
Platform Graphics Layer
Platform Graphics Layer
Platform Graphics Layer
Platform Graphics Layer
Lorem Ipsum is simply dummy text
Composited!
FooooBar!
.
.
....clipped
OpenGL [ES]
PlatformCompositor


Web Process
Main Thread
IPC
UI Process or
Compositing Thread
LayerTreeHost
Coordinated
GraphicsScene
Threaded
commitSceneState(CoordinatedOperations)
CoordinatedGraphicsL
ayer
CoordinatedOperation
CreateLayer
DeleteLayer
SetRootLayer
SyncLayerState
SetLayerChildren
...
TextureMapperLayer

WORST CASE



SAME CASE WITH DEDICATED COMPOSITING THREAD



WHAT WE HAVE DONE : WEBKITGTK / WEBKITFORWAYLAND

- Split compositing operations into the dedicated thread
- Utilize multi-core CPUs and GPU
- Play CSS Animation off-the-main-thread
- Reduce latencies of scrolling and scaling operations

VIDEO RENDERING

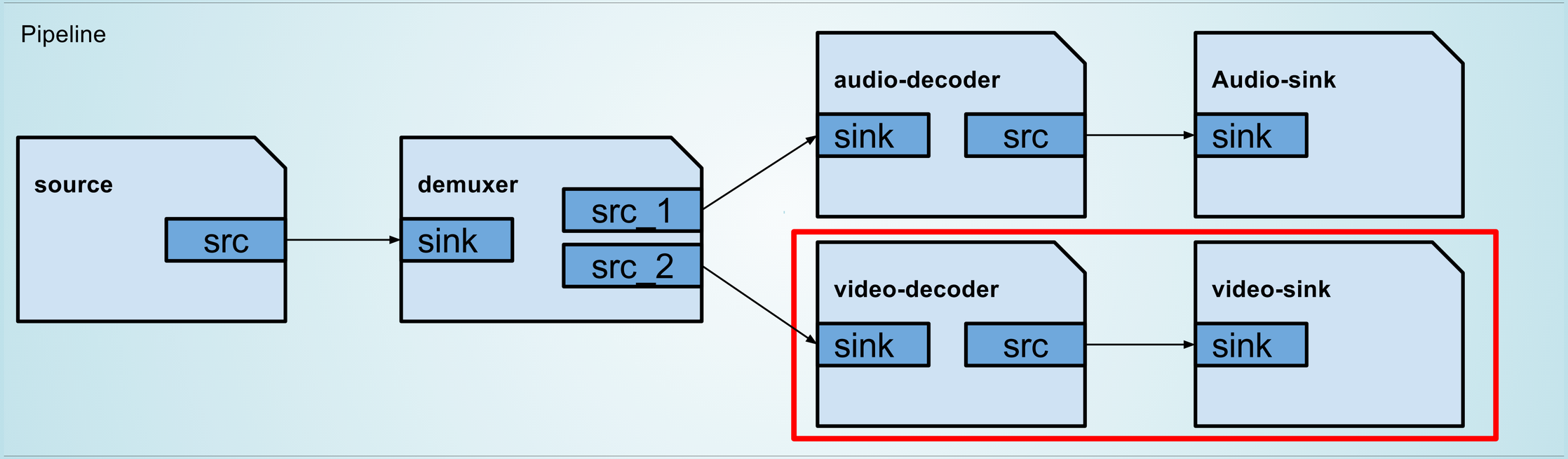
GSTREAMER

- Open source media framework for multimedia playback.
- GStreamer constructs graphs of media-handling components.
- Supports playback, streaming, complex audio mixing and non-linear video processing
- Can handle muxers/demuxers and codecs transparently.
- Add codecs/filters by writing plugins with a generic interface
- A major version is API and ABI stable.

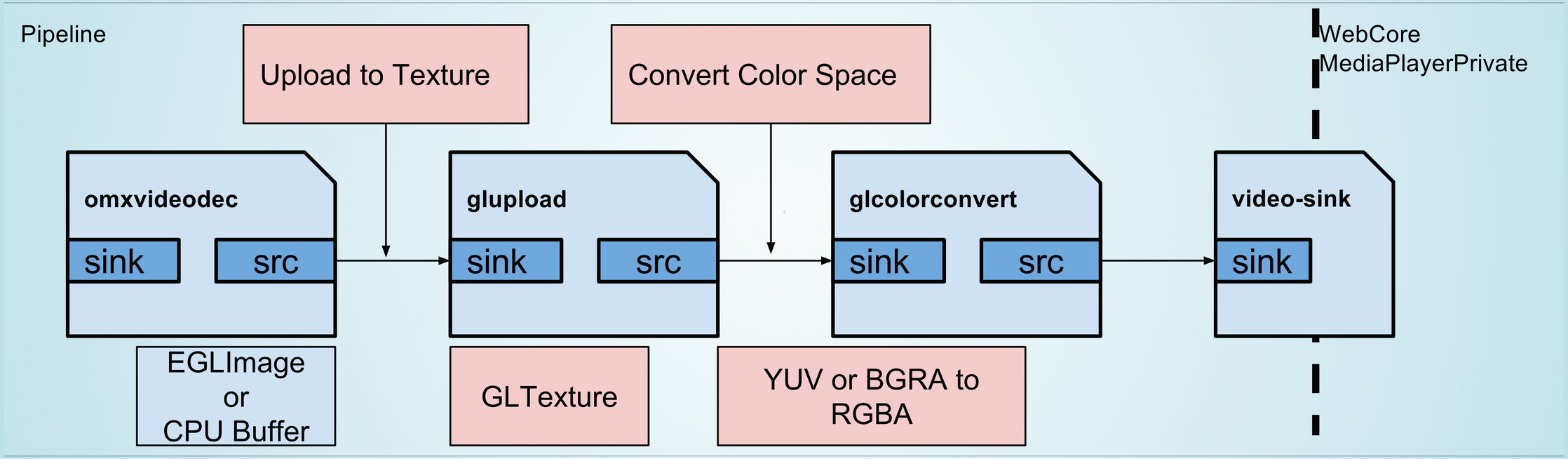
GSTREAMER OPENMAX (GST-OMX)

- Hardware decoders for H.264, VP8, Theora
- meta:EGLImage
- Custom audio sinks: HDMI and analog
- Very good integration within playbin

GSTREAMER PIPELINE



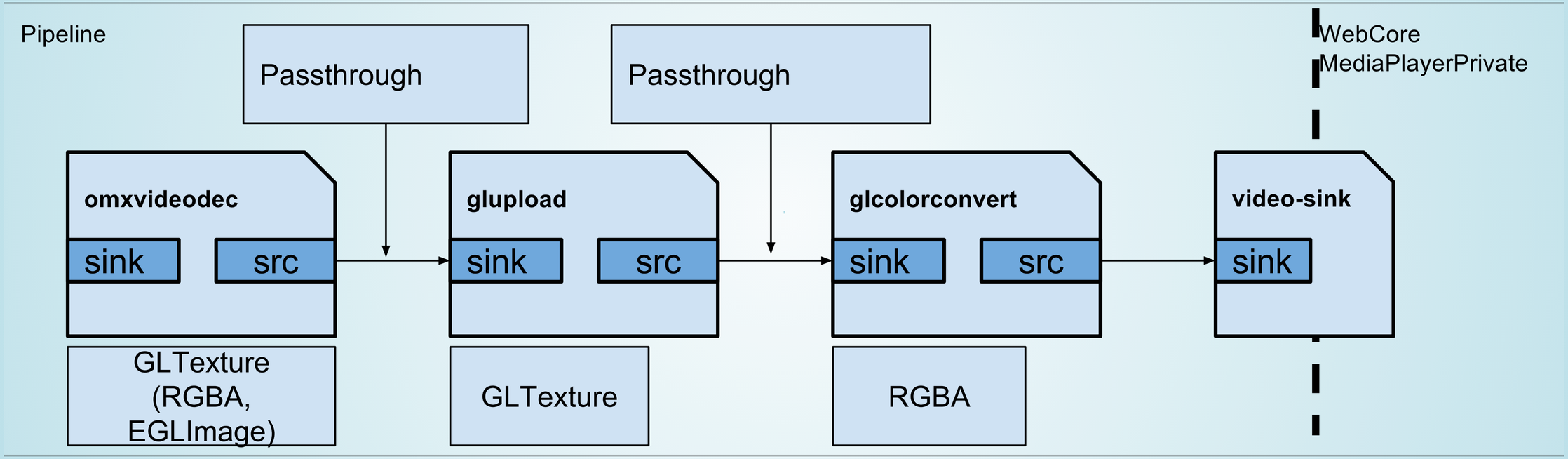
GSTREAMER PIPELINE - INEFFICIENT



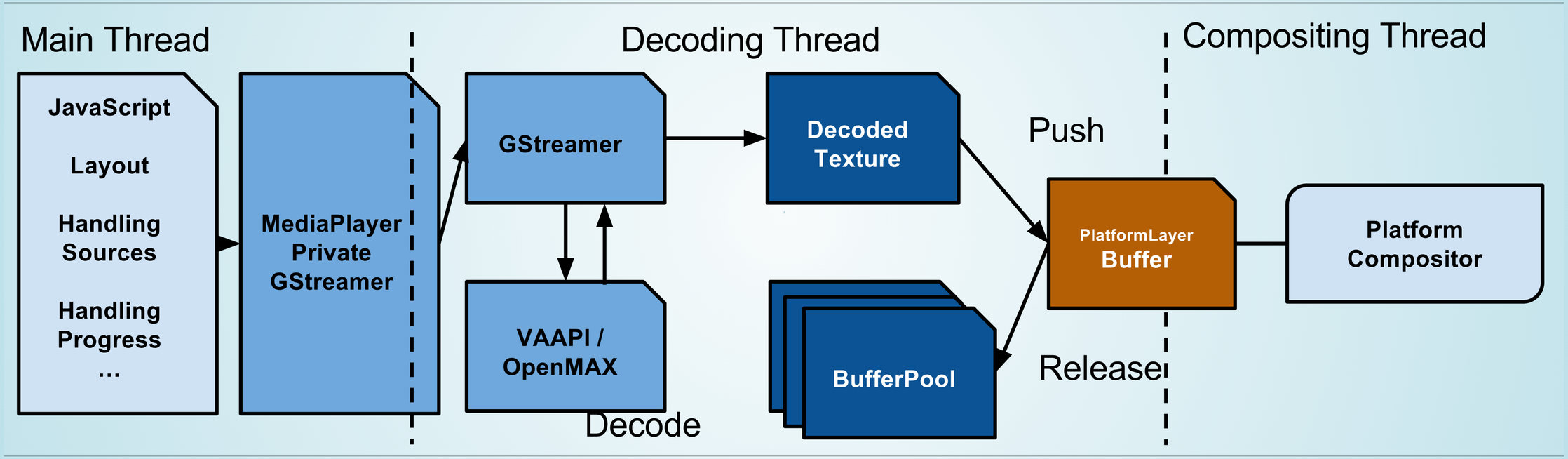
POLISH GST-OMX AND GST-GL TO REMOVE OVERHEADS

- GstGLMemoryEGL (EGLImage + GLMemory)
- Remove additional texture allocations and copy operations
  - 3.5 Mb for each frame (720p, RGBA)
- **Passthrough**

GSTREAMER PIPELINE - EFFICIENT

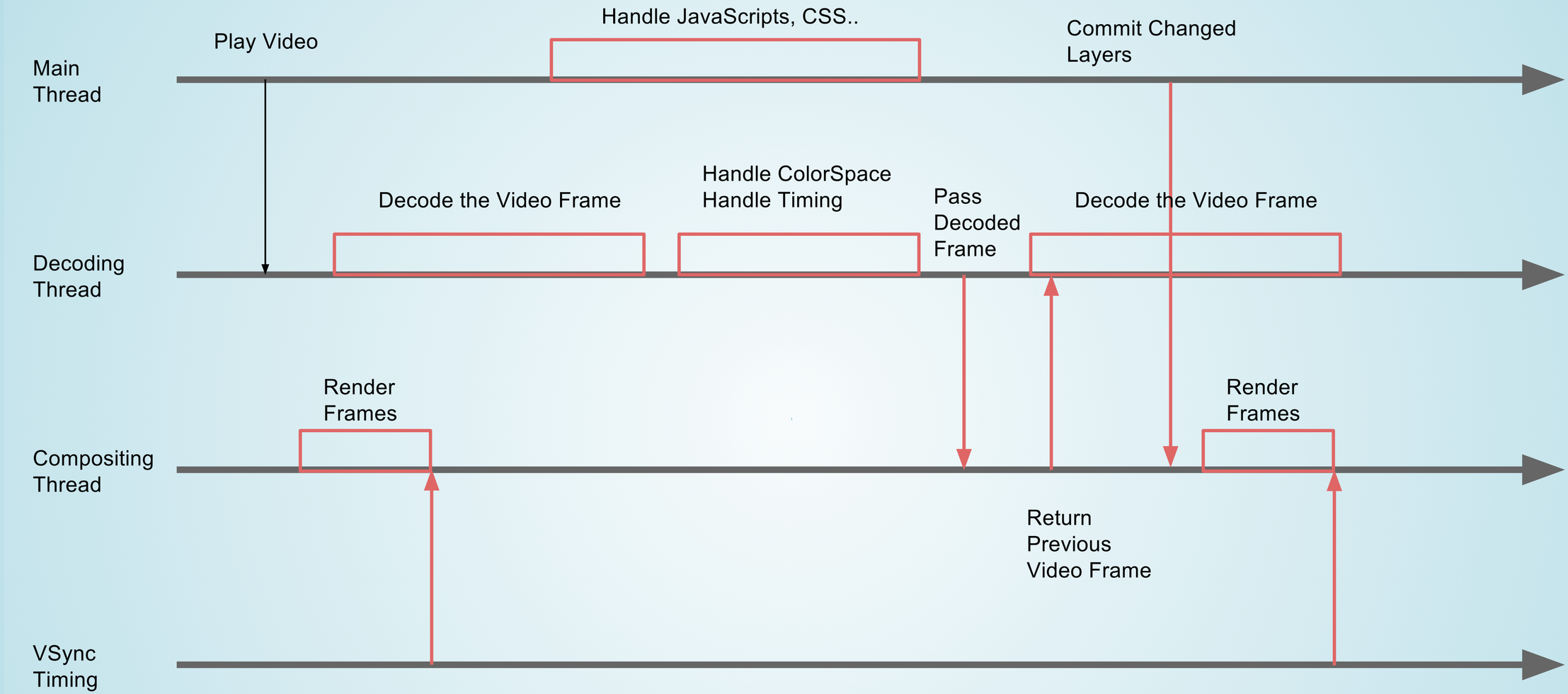


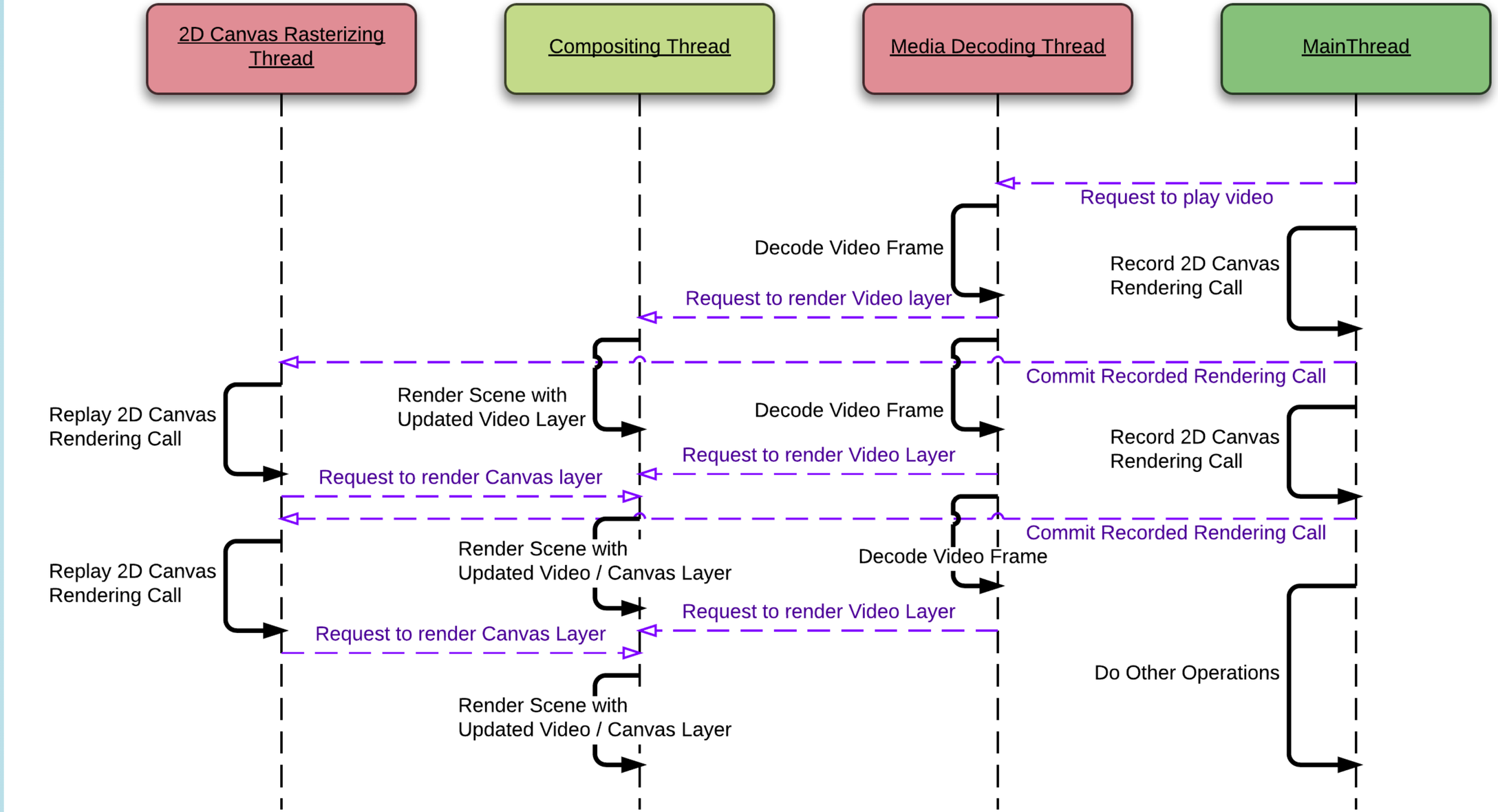
COMPOSITE DECODED FRAME



COMPOSITE DECODED FRAME

- Pass the decoded frame to the compositor directly
- Compositor composites the video without waiting main-thread


Main Thread
Play Video
Handle JavaScripts, CSS..
Commit Changed Layers
Decoding Thread
Decode the Video Frame
Handle ColorSpace
Handle Timing
Pass Decoded Frame
Decode the Video Frame
Compositing Thread
Render Frames
Render Frames
Return Previous Video Frame
VSync Timing


2D Canvas Rasterizing Thread
Compositing Thread
Media Decoding Thread
MainThread
Request to play video
Decode Video Frame
Record 2D Canvas Rendering Call
Request to render Video layer
Commit Recorded Rendering Call
Render Scene with Updated Video Layer
Decode Video Frame
Record 2D Canvas Rendering Call
Replay 2D Canvas Rendering Call
Request to render Video Layer
Request to render Canvas layer
Commit Recorded Rendering Call
Render Scene with Updated Video / Canvas Layer
Decode Video Frame
Replay 2D Canvas Rendering Call
Request to render Video Layer
Request to render Canvas Layer
Do Other Operations
Render Scene with Updated Video / Canvas Layer

TEST RESULTS ON THE RASPBERRY PI2

- Targeting 720p, 1080p
- 30 FPS on HTML5 Video playback
- 40-50 FPS with a 720p HTML5 Video and WebGL at same time
- Reduced memory consumption
- Still, needs to reduce ghost copies of decoded frame

THANK YOU

QUESTIONS?