Copy Report to Clipboard

Graphics Feature Status

- Canvas: Hardware accelerated
- Canvas out-of-process rasterization: Disabled
- Custo Wallpaper Animation: Enabled
- Direct Rendering Display Compositor: Disabled
- Compositing: Software only. Hardware acceleration disabled
- Multiple Raster Threads: Enabled
- OpenGL: Enabled
- Rasterization: Hardware accelerated
- Raw Draw: Disabled
- Video Decode: Software only. Hardware acceleration disabled
- Video Encode: Software only. Hardware acceleration disabled
- Vulkan: Enabled
- WebGL: Hardware accelerated but at reduced performance
- WebGL2: Hardware accelerated but at reduced performance
- WebGPU: Disabled

Driver Bug Workarounds

- adjust_src_dst_region_for_blitframebuffer
- clear_uniforms_before_first_program_use
- count_all_in_varyings_packing
- disable_post_sub_buffers_for_onscreen_surfaces
- enable_webgl_timer_query_extensions
- exit_on_context_lost
- msaa_is_slow
- disabled_extension_GL_KHR_blend_equation_advanced
- disabled_extension_GL_KHR_blend_equation_advanced_coherent
- disabled_extension_GL_MESA_framebuffer_flip_y

Problems Detected

- WebGPU has been disabled via blocklist or the command line.
  Disabled Features: webgpu
- Accelerated video encode has been disabled, either via blocklist, about:flags or the command line.
  Disabled Features: video_encode
- Accelerated video decode has been disabled, either via blocklist, about:flags or the command line.
  Disabled Features: video_decode
- Gpu compositing has been disabled, either via blocklist, about:flags or the command line. The browser will fall back to software compositing and hardware acceleration will be unavailable.
  Disabled Features: gpu_compositing
- Clear uniforms before first program use on all platforms: 124764, 349137
  Applied Workarounds: clear_uniforms_before_first_program_use
- Mesa drivers in Linux handle varyings without static use incorrectly: 333885
  Applied Workarounds: count_all_in_varyings_packing
- On Intel GPUs MSAA performance is not acceptable for GPU rasterization: 527565, 1298585
  Applied Workarounds: msaa_is_slow
- Disable partial swaps on Mesa drivers (detected with GL_VERSION): 339493
  Applied Workarounds: disable_post_sub_buffers_for_onscreen_surfaces
- adjust src/dst region if blitting pixels outside framebuffer on Linux Intel: 664740
  Applied Workarounds: adjust_src_dst_region_for_blitframebuffer

- Disable KHR_blend_equation_advanced until cc shaders are updated: 661715
  Applied Workarounds: disable(GL_KHR_blend_equation_advanced), disable(GL_KHR_blend_equation_advanced_coherent)
- Expose WebGL's disjoint_timer_query extensions on platforms with site isolation: 808744, 870491
  Applied Workarounds: enable_webgl_timer_query_extensions
- Some drivers can't recover after OUT_OF_MEM and context lost: 893177
  Applied Workarounds: exit_on_context_lost
- Disable GL_MESA_framebuffer_flip_y for desktop GL: 964010
  Applied Workarounds: disable(GL_MESA_framebuffer_flip_y)

ANGLE Features

- **allowCompressedFormats** (Frontend workarounds): Enabled: true
  Allow compressed formats
- **cacheCompiledShader** (Frontend features) anglebug:7036: Disabled
  Enable to cache compiled shaders
- **disableAnisotropicFiltering** (Frontend workarounds): Disabled
  Disable support for anisotropic filtering
- **disableDrawBuffersIndexed** (Frontend features) anglebug:7724: Disabled
  Disable support for OES_draw_buffers_indexed and EXT_draw_buffers_indexed
- **disableProgramBinary** (Frontend features) anglebug:5007: Disabled: IsPowerVrRogue(functions)
  Disable support for GL_OES_get_program_binary
- **disableProgramCachingForTransformFeedback** (Frontend workarounds): Disabled: IsAndroid() && isQualcomm
  On some GPUs, program binaries don't contain transform feedback varyings
- **emulatePixelLocalStorage** (Frontend features) anglebug:7279: Disabled: false
  Emulate ANGLE_shader_pixel_local_storage using shader images
- **enableCaptureLimits** (Frontend features) anglebug:5750: Disabled
  Set the context limits like frame capturing was enabled
- **enableProgramBinaryForCapture** (Frontend features) anglebug:5658: Disabled
  Even if FrameCapture is enabled, enable GL_OES_get_program_binary
- **forceDepthAttachmentInitOnClear** (Frontend workarounds) anglebug:7246: Disabled
  Force depth attachment initialization on clear ops
- **forceGlErrorChecking** (Frontend features) https://issuetracker.google.com/220069903: Disabled
  Force GL error checking (i.e. prevent applications from disabling error checking
- **forceInitShaderVariables** (Frontend features): Disabled
  Force-enable shader variable initialization
- **forceRobustResourceInit** (Frontend features) anglebug:6041: Disabled
  Force-enable robust resource init
- **loseContextOnOutOfMemory** (Frontend workarounds): Enabled: true
  Some users rely on a lost context notification if a GL_OUT_OF_MEMORY error occurs
- **scalarizeVecAndMatConstructorArgs** (Frontend workarounds) 1165751: Disabled: false
  Always rewrite vec/mat constructors to be consistent
- **singleThreadedTextureDecompression** (Frontend workarounds): Disabled
  Disables multi-threaded decompression of compressed texture formats
- **RGBA4IsNotSupportedForColorRendering** (OpenGL workarounds): Enabled: functions->standard == STANDARD_GL_DESKTOP && isIntel
  GL_RGBA4 is not color renderable
- **RGBDXT1TexturesSampleZeroAlpha** (OpenGL workarounds) anglebug:3729: Disabled: IsApple()
  Sampling BLACK texels from RGB DXT1 textures returns transparent black on Mac.
- **addAndTrueToLoopCondition** (OpenGL workarounds): Disabled: IsApple() && isIntel
  Calculation of loop conditions in for and while loop has bug
- **adjustSrcDstRegionForBlitFramebuffer** (OpenGL workarounds) 830046: Enabled: IsLinux() || (IsAndroid() && isNvidia) || (IsWindows() && isNvidia) || (IsApple() && functions-

>standard == STANDARD_GL_ES)
Many platforms have issues with blitFramebuffer when the parameters are large.

- **allowAstcFormats** (OpenGL workarounds): Enabled: !isMesa || isIntel && (Is9thGenIntel(device) || IsGeminiLake(device) || IsCoffeeLake(device) || Is11thGenIntel(device) || Is12thGenIntel(device))
  Enable ASTC on desktop OpenGL
- **allowClearForRobustResourceInit** (OpenGL workarounds) 848952: Disabled: IsApple()
  Using glClear for robust resource initialization is buggy on some drivers and leads to texture corruption. Default to data uploads except on MacOS where it is very slow.
- **allowETCFormats** (OpenGL workarounds): Enabled: isIntel && !IsSandyBridge(device) && !IsIvyBridge(device) && !IsHaswell(device)
  Enable ETC2/EAC on desktop OpenGL
- **alwaysCallUseProgramAfterLink** (OpenGL workarounds) 110263: Enabled: true
  Always call useProgram after a successful link to avoid a driver bug
- **alwaysUnbindFramebufferTexture2D** (OpenGL workarounds) anglebug:5536: Disabled: isNvidia && (IsWindows() || IsLinux())
  Force unbind framebufferTexture2D before binding renderbuffer to work around driver bug.
- **avoid1BitAlphaTextureFormats** (OpenGL workarounds): Disabled: functions->standard == STANDARD_GL_DESKTOP && isAMD
  Issue with 1-bit alpha framebuffer formats
- **bindTransformFeedbackBufferBeforeBindBufferRange** (OpenGL workarounds) anglebug:5140: Disabled: IsApple()
  Bind transform feedback buffers to the generic binding point before calling glBindBufferBase or glBindBufferRange.
- **clampArrayAccess** (OpenGL workarounds) anglebug:2978: Disabled: IsAndroid() || isAMD || !functions->hasExtension("GL_KHR_robust_buffer_access_behavior")
  Clamp uniform array access to avoid reading invalid memory.
- **clampFragDepth** (OpenGL workarounds): Disabled: isNvidia
  gl_FragDepth is not clamped correctly when rendering to a floating point depth buffer
- **clampMscRate** (OpenGL workarounds) 1042393: Disabled: IsLinux() && IsWayland()
  Some drivers return bogus values for GetMscRate, so we clamp it to 30Hz
- **clampPointSize** (OpenGL workarounds): Disabled: IsAndroid() || isNvidia
  The point size range reported from the API is inconsistent with the actual behavior
- **clearToZeroOrOneBroken** (OpenGL workarounds) 710443: Disabled: IsApple() && isIntel && GetMacOSVersion() < OSVersion(10, 12, 6)
  Clears when the clear color is all zeros or ones do not work.
- **clipSrcRegionForBlitFramebuffer** (OpenGL workarounds) 830046: Disabled: IsApple() || (IsLinux() && isAMD)
  Issues with blitFramebuffer when the parameters don't match the framebuffer size.
- **decodeEncodeSRGBForGenerateMipmap** (OpenGL workarounds) anglebug:4646: Disabled: IsApple()
  Decode and encode before generateMipmap for srgb format textures.
- **disableBlendFuncExtended** (OpenGL workarounds) anglebug:1085: Enabled: isAMD || isIntel
  ARB_blend_func_extended does not pass the tests
- **disableClipCullDistance** (OpenGL workarounds) anglebug:7763: Disabled: isQualcomm
  Shader compiler does not handle redeclared built-ins.
- **disableDrawBuffersIndexed** (OpenGL workarounds): Disabled: IsWindows() && isAMD
  Disable OES_draw_buffers_indexed extension.
- **disableGPUSwitchingSupport** (OpenGL workarounds) 1091824: Disabled: isDualGPUMacWithNVIDIA
  Disable GPU switching support (use only the low-power GPU) on older MacBook Pros.
- **disableMultisampledRenderToTexture** (OpenGL workarounds) anglebug:2894: Disabled: isAdreno4xxOnAndroidLessThan51 || isAdreno4xxOnAndroid70 || isAdreno5xxOnAndroidLessThan70 || isAdreno5xxOnAndroid71 || isLinuxVivante
  Many drivers have bugs when using GL_EXT_multisampled_render_to_texture
- **disableNativeParallelCompile** (OpenGL workarounds) 1094869: Disabled: isTSANBuild && IsLinux() && isNvidia

Do not use native KHR_parallel_shader_compile even when available.

- **disableSemaphoreFd** (OpenGL workarounds) 1046462: Disabled: IsLinux() && isAMD && isMesa && mesaVersion < (std::array<int, 3>{19, 3, 5})
  Disable GL_EXT_semaphore_fd extension
- **disableSyncControlSupport** (OpenGL workarounds) 1137851: Disabled: IsLinux() && isIntel && isMesa && mesaVersion[0] == 20
  Speculative fix for issues on Linux/Wayland where exposing GLX_OML_sync_control renders Chrome unusable
- **disableTextureClampToBorder** (OpenGL workarounds) anglebug:7405: Disabled: isImagination
  Imagination devices generate INVALID_ENUM when setting the texture border color.
- **disableTimestampQueries** (OpenGL workarounds) 811661: Disabled: (IsLinux() && isVMWare) || (IsAndroid() && isNvidia) || (IsAndroid() && GetAndroidSdkLevel() < 27 && IsAdreno5xxOrOlder(functions)) || (IsAndroid() && IsMaliT8xxOrOlder(functions)) || (IsAndroid() && IsMaliG31OrOlder(functions))
  Disable GL_EXT_disjoint_timer_query extension
- **disableWorkerContexts** (OpenGL workarounds) 849576: Disabled: (IsWindows() && (isIntel || isAMD)) || (IsLinux() && isNvidia) || IsIOS() || IsAndroid() || IsAndroidEmulator(functions)
  Some tests have been seen to fail using worker contexts
- **doWhileGLSLCausesGPUHang** (OpenGL workarounds) 644669: Disabled: IsApple() && functions->standard == STANDARD_GL_DESKTOP && GetMacOSVersion() < OSVersion(10, 11, 0)
  Some GLSL constructs involving do-while loops cause GPU hangs
- **doesSRGBClearsOnLinearFramebufferAttachments** (OpenGL workarounds): Enabled: isIntel || isAMD
  Issue clearing framebuffers with linear attachments when GL_FRAMEBUFFER_SRGB is enabled
- **dontInitializeUninitializedLocals** (OpenGL workarounds) anglebug:2046: Disabled: IsAndroid() && isQualcomm
  Initializing uninitialized locals caused odd behavior in a few WebGL 2 tests
- **dontRelinkProgramsInParallel** (OpenGL workarounds) anglebug:3045: Disabled: IsAndroid() || (IsWindows() && isIntel)
  Relinking a program in parallel is buggy
- **dontUseLoopsToInitializeVariables** (OpenGL workarounds) 809422: Disabled: (IsAndroid() && isQualcomm) || (isIntel && IsApple())
  For loops used to initialize variables hit native GLSL compiler bugs
- **emulateAbsIntFunction** (OpenGL workarounds) 642227: Disabled: IsApple() && isIntel
  abs(i) where i is an integer returns unexpected result
- **emulateAtan2Float** (OpenGL workarounds) 672380: Disabled: isNvidia
  atan(y, x) may return a wrong answer
- **emulateCopyTexImage2D** (OpenGL workarounds): Disabled: isApple
  Replace CopyTexImage2D with TexImage2D + CopyTexSubImage2D.
- **emulateCopyTexImage2DFromRenderbuffers** (OpenGL workarounds) anglebug:4674: Disabled: IsApple() && functions->standard == STANDARD_GL_ES && !(isAMD && IsWindows())
  CopyTexImage2D spuriously returns errors on iOS when copying from renderbuffers.
- **emulateImmutableCompressedTexture3D** (OpenGL workarounds) 1060012: Disabled: isQualcomm
  Use non-immutable texture allocation to work around a driver bug.
- **emulateIsnanFloat** (OpenGL workarounds) 650547: Disabled: isIntel && IsApple() && IsSkylake(device) && GetMacOSVersion() < OSVersion(10, 13, 2)
  Using isnan() on highp float will get wrong answer
- **emulateMaxVertexAttribStride** (OpenGL workarounds) anglebug:1936: Disabled: IsLinux() && functions->standard == STANDARD_GL_DESKTOP && isAMD
  Some drivers return 0 when MAX_VERTEX_ATTRIB_STRIED queried
- **emulatePackSkipRowsAndPackSkipPixels** (OpenGL workarounds) anglebug:4849: Disabled: IsApple()

GL_PACK_SKIP_ROWS and GL_PACK_ṠKIP_PIXELS are ignored in Apple's OpenGL driver.

- **emulatePrimitiveRestartFixedIndex** (OpenGL workarounds) anglebug:3997: Disabled: functions->standard == STANDARD_GL_DESKTOP && functions->isAtLeastGL(gl::Version(3, 1)) && !functions->isAtLeastGL(gl::Version(4, 3))
  When GL_PRIMITIVE_RESTART_FIXED_INDEX is not available, emulate it with GL_PRIMITIVE_RESTART and glPrimitiveRestartIndex.
- **emulateRGB10** (OpenGL workarounds) 1300575: Enabled: functions->standard == STANDARD_GL_DESKTOP
  Emulate RGB10 support using RGB10_A2.
- **finishDoesNotCauseQueriesToBeAvailable** (OpenGL workarounds): Disabled: functions->standard == STANDARD_GL_DESKTOP && isNvidia
  glFinish doesn't cause all queries to report available result
- **flushBeforeDeleteTextureIfCopiedTo** (OpenGL workarounds) anglebug:4267: Disabled: IsApple() && isIntel
  Some drivers track CopyTex{Sub}Image texture dependencies incorrectly. Flush before glDeleteTextures in this case
- **flushOnFramebufferChange** (OpenGL workarounds) 1181068: Disabled: IsApple() && Has9thGenIntelGPU(systemInfo)
  Switching framebuffers without a flush can lead to crashes on Intel 9th Generation GPU Macs.
- **initFragmentOutputVariables** (OpenGL workarounds) 1171371: Disabled: IsAdreno42xOr3xx(functions)
  No init gl_FragColor causes context lost
- **initializeCurrentVertexAttributes** (OpenGL workarounds): Disabled: isNvidia
  During initialization, assign the current vertex attributes to the spec-mandated defaults
- **keepBufferShadowCopy** (OpenGL workarounds): Disabled: !CanMapBufferForRead(functions)
  Maintain a shadow copy of buffer data when the GL API does not permit reading data back.
- **limitMax3dArrayTextureSizeTo1024** (OpenGL workarounds) 927470: Disabled: limitMaxTextureSize
  Limit max 3d texture size and max array texture layers to 1024 to avoid system hang
- **limitMaxMSAASamplesTo4** (OpenGL workarounds) 797243: Disabled: IsAndroid() || (IsApple() && (isIntel || isAMD || isNvidia))
  Various rendering bugs have been observed when using higher MSAA counts
- **limitWebglMaxTextureSizeTo4096** (OpenGL workarounds) 927470: Disabled: IsAndroid() || limitMaxTextureSize
  Limit webgl max texture size to 4096 to avoid frequent out-of-memory errors
- **packLastRowSeparatelyForPaddingInclusion** (OpenGL workarounds) anglebug:1512: Disabled: IsApple() || isNvidia
  When uploading textures from an pack buffer, some drivers count an extra row padding
- **packOverlappingRowsSeparatelyPackBuffer** (OpenGL workarounds): Disabled: isNvidia
  In the case of packing to a pixel pack buffer, pack overlapping rows row by row
- **passHighpToPackUnormSnormBuiltins** (OpenGL workarounds) anglebug:7527: Disabled: isQualcomm
  packUnorm4x8 fails on Pixel 4 if it is not passed a highp vec4.
- **preAddTexelFetchOffsets** (OpenGL workarounds) 642605: Disabled: IsApple() && isIntel
  Intel Mac drivers mistakenly consider the parameter position of nagative vaule as invalid even if the sum of position and offset is in range, so we need to add workarounds by rewriting texelFetchOffset(sampler, position, lod, offset) into texelFetch(sampler, position + offset, lod).
- **promotePackedFormatsTo8BitPerChannel** (OpenGL workarounds) anglebug:5469: Disabled: IsApple() && hasAMD
  Packed color formats are buggy on Macs with AMD GPUs
- **queryCounterBitsGeneratesErrors** (OpenGL workarounds) anglebug:3027: Disabled: IsNexus5X(vendor, device)
  Drivers generate errors when querying the number of bits in timer queries

- **readPixelsUsingImplementationColorReadFormatForNorm16** (OpenGL workarounds) anglebug:4214: Disabled: !isIntel && functions->standard == STANDARD_GL_ES && functions->isAtLeastGLES(gl::Version(3, 1)) && functions->hasGLESExtension("GL_EXT_texture_norm16")
  Quite some OpenGL ES drivers don't implement readPixels for RGBA/UNSIGNED_SHORT from EXT_texture_norm16 correctly
- **reapplyUBOBindingsAfterUsingBinaryProgram** (OpenGL workarounds) anglebug:1637: Disabled: isAMD || IsAndroid()
  Some drivers forget about UBO bindings when using program binaries
- **regenerateStructNames** (OpenGL workarounds) 403957: Disabled: IsApple()
  All Mac drivers do not handle struct scopes correctly. This workaround overwrites a structname with a unique prefix.
- **removeDynamicIndexingOfSwizzledVector** (OpenGL workarounds) 709351: Disabled: IsApple() || IsAndroid() || IsWindows()
  Dynamic indexing of swizzled l-values doesn't work correctly on various platforms.
- **removeInvariantAndCentroidForESSL3** (OpenGL workarounds): Disabled: functions->isAtMostGL(gl::Version(4, 1)) || (functions->standard == STANDARD_GL_DESKTOP && isAMD)
  Fix spec difference between GLSL 4.1 or lower and ESSL3
- **resetTexImage2DBaseLevel** (OpenGL workarounds) 705865: Disabled: IsApple() && isIntel && GetMacOSVersion() >= OSVersion(10, 12, 4)
  Reset texture base level before calling glTexImage2D to work around pixel comparison failure.
- **rewriteFloatUnaryMinusOperator** (OpenGL workarounds) 308366: Disabled: IsApple() && isIntel && GetMacOSVersion() < OSVersion(10, 12, 0)
  Using '-<float>' will get wrong answer
- **rewriteRepeatedAssignToSwizzled** (OpenGL workarounds): Disabled: isNvidia
  Repeated assignment to swizzled values inside a GLSL user-defined function have incorrect results
- **rewriteRowMajorMatrices** (OpenGL workarounds) anglebug:2273: Disabled: false
  Rewrite row major matrices in shaders as column major as a driver bug workaround
- **sanitizeAMDGPURendererString** (OpenGL workarounds) 1181193: Disabled: IsLinux() && hasAMD
  Strip precise kernel and DRM version information from amdgpu renderer strings.
- **setPrimitiveRestartFixedIndexForDrawArrays** (OpenGL workarounds) anglebug:3997: Disabled: features->emulatePrimitiveRestartFixedIndex.enabled && IsApple() && isIntel
  Some drivers discard vertex data in DrawArrays calls when the fixed primitive restart index is within the number of primitives being drawn.
- **setZeroLevelBeforeGenerateMipmap** (OpenGL workarounds): Disabled: IsApple()
  glGenerateMipmap fails if the zero texture level is not set on some Mac drivers.
- **shiftInstancedArrayDataWithOffset** (OpenGL workarounds) 1144207: Disabled: IsApple() && IsIntel(vendor) && !IsHaswell(device)
  glDrawArraysInstanced is buggy on certain new Mac Intel GPUs
- **supportsFragmentShaderInterlockARB** (OpenGL features) anglebug:7279: Disabled: functions->isAtLeastGL(gl::Version(4, 5)) && functions->hasGLExtension("GL_ARB_fragment_shader_interlock")
  Backend GL context supports ARB_fragment_shader_interlock extension
- **supportsFragmentShaderInterlockNV** (OpenGL features) anglebug:7279: Disabled: functions->isAtLeastGL(gl::Version(4, 3)) && functions->hasGLExtension("GL_NV_fragment_shader_interlock")
  Backend GL context supports NV_fragment_shader_interlock extension
- **supportsFragmentShaderOrderingINTEL** (OpenGL features) anglebug:7279: Disabled: functions->isAtLeastGL(gl::Version(4, 4)) && functions->hasGLExtension("GL_INTEL_fragment_shader_ordering")
  Backend GL context supports GL_INTEL_fragment_shader_ordering extension
- **supportsShaderFramebufferFetchEXT** (OpenGL features) anglebug:7279: Disabled: functions->hasGLESExtension("GL_EXT_shader_framebuffer_fetch")
  Backend GL context supports EXT_shader_framebuffer_fetch extension

- **supportsShaderFramebufferFetchNonCoherentEXT** (OpenGL features) anglebug:7279: Disabled: functions->hasGLESExtension("GL_EXT_shader_framebuffer_fetch_non_coherent")
  Backend GL context supports EXT_shader_framebuffer_fetch_non_coherent extension
- **supportsShaderPixelLocalStorageEXT** (OpenGL features) anglebug:7279: Disabled: functions->hasGLESExtension("GL_EXT_shader_pixel_local_storage")
  Backend GL context supports EXT_shader_pixel_local_storage extension
- **syncVertexArraysToDefault** (OpenGL workarounds) anglebug:5577: Disabled: !nativegl::SupportsVertexArrayObjects(functions)
  Only use the default VAO because of missing support or driver bugs
- **unbindFBOBeforeSwitchingContext** (OpenGL workarounds) 1181193: Disabled: IsPowerVR(vendor)
  Imagination GL drivers are buggy with context switching.
- **unfoldShortCircuits** (OpenGL workarounds) anglebug:482: Disabled: IsApple()
  Mac incorrectly executes both sides of && and || expressions when they should short-circuit.
- **unpackLastRowSeparatelyForPaddingInclusion** (OpenGL workarounds) anglebug:1512: Disabled: IsApple() || isNvidia
  When uploading textures from an unpack buffer, some drivers count an extra row padding
- **unpackOverlappingRowsSeparatelyUnpackBuffer** (OpenGL workarounds): Disabled: isNvidia
  In the case of unpacking from a pixel unpack buffer, unpack overlapping rows row by row
- **unsizedSRGBReadPixelsDoesntTransform** (OpenGL workarounds) 550292: Disabled: IsAndroid() && isQualcomm
  Drivers returning raw sRGB values instead of linearized values when calling glReadPixels on unsized sRGB texture formats
- **uploadTextureDataInChunks** (OpenGL workarounds) 1181068: Disabled: IsApple()
  Upload texture data in <120kb chunks to work around Mac driver hangs and crashes.
- **useUnusedBlocksWithStandardOrSharedLayout** (OpenGL workarounds): Disabled: (IsApple() && functions->standard == STANDARD_GL_DESKTOP) || (IsLinux() && isAMD)
  Unused std140 or shared uniform blocks will be treated as inactive
- **vertexIDDoesNotIncludeBaseVertex** (OpenGL workarounds): Disabled: IsApple() && isAMD
  gl_VertexID in GLSL vertex shader doesn't include base vertex value

DAWN Info

<Integrated GPU> Vulkan backend - Intel(R) Graphics (ADL GT2)
[Default Toggle Names]

- **lazy_clear_resource_on_first_use:** https://crbug.com/dawn/145*: Clears resource to zero on first usage. This initializes the resource so that no dirty bits from recycled memory is present in the new resource.*
- **use_temporary_buffer_in_texture_to_texture_copy:** https://crbug.com/dawn/42*: Split texture-to-texture copy into two copies: copy from source texture into a temporary buffer, and copy from the temporary buffer into the destination texture when copying between compressed textures that don't have block-aligned sizes. This workaround is enabled by default on all Vulkan drivers to solve an issue in the Vulkan SPEC about the texture-to-texture copies with compressed formats. See #1005 (https://github.com/KhronosGroup/Vulkan-Docs/issues/1005) for more details.*
- **vulkan_use_d32s8:** https://crbug.com/dawn/286*: Vulkan mandates support of either D32_FLOAT_S8 or D24_UNORM_S8. When available the backend will use D32S8 (toggle to on) but setting the toggle to off will make it use the D24S8 format when possible.*
- **vulkan_use_s8:** https://crbug.com/dawn/666*: Vulkan has a pure stencil8 format but it is not universally available. When this toggle is on, the backend will use S8 for the stencil8 format, otherwise it will fallback to D32S8 or D24S8.*
- **disallow_unsafe_apis:** http://crbug.com/1138528*: Produces validation errors on API entry points or parameter combinations that aren't considered secure yet.*

- **use_vulkan_zero_initialize_workgroup_memory_extension:** https://crbug.com/dawn/1302*: Initialize workgroup memory with OpConstantNull on Vulkan when the Vulkan extension VK_KHR_zero_initialize_workgroup_memory is supported.* [WebGPU Forced Toggles - enabled]
- **disallow_spirv:** https://crbug.com/1214923*: Disallow usage of SPIR-V completely so that only WGSL is used for shader modules. This is useful to prevent a Chromium renderer process from successfully sending SPIR-V code to be compiled in the GPU process.* [Supported Features]
- texture-compression-bc
- texture-compression-etc2
- texture-compression-astc
- pipeline-statistics-query
- timestamp-query
- timestamp-query-inside-passes
- depth-clip-control
- depth32float-stencil8
- chromium-experimental-dp4a
- indirect-first-instance
- rg11b10ufloat-renderable
- dawn-internal-usages
- dawn-native

<CPU> Vulkan backend - llvmpipe (LLVM 11.0.1, 256 bits)
[Default Toggle Names]

- **lazy_clear_resource_on_first_use:** https://crbug.com/dawn/145*: Clears resource to zero on first usage. This initializes the resource so that no dirty bits from recycled memory is present in the new resource.*
- **use_temporary_buffer_in_texture_to_texture_copy:** https://crbug.com/dawn/42*: Split texture-to-texture copy into two copies: copy from source texture into a temporary buffer, and copy from the temporary buffer into the destination texture when copying between compressed textures that don't have block-aligned sizes. This workaround is enabled by default on all Vulkan drivers to solve an issue in the Vulkan SPEC about the texture-to-texture copies with compressed formats. See #1005 (https://github.com/KhronosGroup/Vulkan-Docs/issues/1005) for more details.*
- **vulkan_use_d32s8:** https://crbug.com/dawn/286*: Vulkan mandates support of either D32_FLOAT_S8 or D24_UNORM_S8. When available the backend will use D32S8 (toggle to on) but setting the toggle to off will make it use the D24S8 format when possible.*
- **vulkan_use_s8:** https://crbug.com/dawn/666*: Vulkan has a pure stencil8 format but it is not universally available. When this toggle is on, the backend will use S8 for the stencil8 format, otherwise it will fallback to D32S8 or D24S8.*
- **disallow_unsafe_apis:** http://crbug.com/1138528*: Produces validation errors on API entry points or parameter combinations that aren't considered secure yet.*
- **use_vulkan_zero_initialize_workgroup_memory_extension:** https://crbug.com/dawn/1302*: Initialize workgroup memory with OpConstantNull on Vulkan when the Vulkan extension VK_KHR_zero_initialize_workgroup_memory is supported.* [WebGPU Forced Toggles - enabled]
- **disallow_spirv:** https://crbug.com/1214923*: Disallow usage of SPIR-V completely so that only WGSL is used for shader modules. This is useful to prevent a Chromium renderer process from successfully sending SPIR-V code to be compiled in the GPU process.* [Supported Features]
- texture-compression-bc
- pipeline-statistics-query
- timestamp-query
- timestamp-query-inside-passes
- depth-clip-control
- depth32float-stencil8
- indirect-first-instance
- rg11b10ufloat-renderable

- dawn-internal-usages
- dawn-native

<CPU> Vulkan backend - SwiftShader Device (Subzero)
[Default Toggle Names]

- **lazy_clear_resource_on_first_use:** https://crbug.com/dawn/145*: Clears resource to zero on first usage. This initializes the resource so that no dirty bits from recycled memory is present in the new resource.*
- **use_temporary_buffer_in_texture_to_texture_copy:** https://crbug.com/dawn/42*: Split texture-to-texture copy into two copies: copy from source texture into a temporary buffer, and copy from the temporary buffer into the destination texture when copying between compressed textures that don't have block-aligned sizes. This workaround is enabled by default on all Vulkan drivers to solve an issue in the Vulkan SPEC about the texture-to-texture copies with compressed formats. See #1005 (https://github.com/KhronosGroup/Vulkan-Docs/issues/1005) for more details.*
- **vulkan_use_d32s8:** https://crbug.com/dawn/286*: Vulkan mandates support of either D32_FLOAT_S8 or D24_UNORM_S8. When available the backend will use D32S8 (toggle to on) but setting the toggle to off will make it use the D24S8 format when possible.*
- **vulkan_use_s8:** https://crbug.com/dawn/666*: Vulkan has a pure stencil8 format but it is not universally available. When this toggle is on, the backend will use S8 for the stencil8 format, otherwise it will fallback to D32S8 or D24S8.*
- **disallow_unsafe_apis:** http://crbug.com/1138528*: Produces validation errors on API entry points or parameter combinations that aren't considered secure yet.*
- **use_vulkan_zero_initialize_workgroup_memory_extension:** https://crbug.com/dawn/1302*: Initialize workgroup memory with OpConstantNull on Vulkan when the Vulkan extension VK_KHR_zero_initialize_workgroup_memory is supported.*
[WebGPU Forced Toggles - enabled]
- **disallow_spirv:** https://crbug.com/1214923*: Disallow usage of SPIR-V completely so that only WGSL is used for shader modules. This is useful to prevent a Chromium renderer process from successfully sending SPIR-V code to be compiled in the GPU process.*
[Supported Features]
- texture-compression-bc
- texture-compression-etc2
- texture-compression-astc
- timestamp-query
- timestamp-query-inside-passes
- depth-clip-control
- depth32float-stencil8
- indirect-first-instance
- rg11b10ufloat-renderable
- dawn-internal-usages
- dawn-native

Version Information

Data exported	2023-05-19T07:00:26.157Z
Chrome version	YaBrowser/23.3.1.946 (corp)
Operating system	Linux 6.1.29-un-def-alt1
Software rendering list path	Path=/src/gpu/config/software_rendering_list.json RevisionId=694ad6c9e86008a1a7f5b19dfd05c1971af3b905
Driver bug list path	Path=/src/gpu/config/gpu_driver_bug_list.json RevisionId=694ad6c9e86008a1a7f5b19dfd05c1971af3b905
ANGLE commit id	unknown hash
2D graphics backend	Skia/110 694ad6c9e86008a1a7f5b19dfd05c1971af3b905
Command Line	/usr/bin/yandex-browser-stable --flag-switches-begin --disable-accelerated-video-decode --ignore-gpu-blocklist --enable-features=Vulkan --flag-switches-end --desktop-startup-id=mate-menu.py-3234-strepkovaas.ipa.basealt.ru-/usr/bin/yandex-browser-

	stable-1_TIME3485543 --external-app-null-path --external-app-data=null_data

Driver Information

Initialization time	69
In-process GPU	false
Passthrough Command Decoder	true
Sandboxed	false
GPU0	VENDOR= 0x8086 [Google Inc. (Mesa/X.org)], DEVICE=0x46a6 [ANGLE (Mesa/X.org, llvmpipe (LLVM 11.0.1 256 bits), OpenGL 4.5 (Core Profile) Mesa 22.3.6)], DRIVER_VENDOR=Mesa, DRIVER_VERSION=22.3.6 *ACTIVE*
Optimus	false
AMD switchable	false
GPU CUDA compute capability major version	0
Pixel shader version	1.00
Vertex shader version	1.00
Max. MSAA samples	4
Machine model name	
Machine model version	
GL_VENDOR	Google Inc. (Mesa/X.org)
GL_RENDERER	ANGLE (Mesa/X.org, llvmpipe (LLVM 11.0.1 256 bits), OpenGL 4.5 (Core Profile) Mesa 22.3.6)
GL_VERSION	OpenGL ES 2.0.0 (ANGLE 2.1.0 git hash: unknown hash)
GL_EXTENSIONS	GL_AMD_performance_monitor GL_ANGLE_base_vertex_base_instance GL_ANGLE_base_vertex_base_instance_shader_builtin GL_ANGLE_client_arrays GL_ANGLE_compressed_texture_etc GL_ANGLE_depth_texture GL_ANGLE_framebuffer_blit GL_ANGLE_framebuffer_multisample GL_ANGLE_get_serialized_context_string GL_ANGLE_get_tex_level_parameter GL_ANGLE_instanced_arrays GL_ANGLE_logic_op GL_ANGLE_memory_size GL_ANGLE_multi_draw GL_ANGLE_program_cache_control GL_ANGLE_provoking_vertex GL_ANGLE_request_extension GL_ANGLE_robust_client_memory GL_ANGLE_texture_compression_dxt3 GL_ANGLE_texture_compression_dxt5 GL_ANGLE_texture_external_update GL_ANGLE_texture_rectangle GL_ANGLE_translated_shader_source GL_APPLE_clip_distance GL_ARB_sync GL_CHROMIUM_bind_generates_resource GL_CHROMIUM_bind_uniform_location GL_CHROMIUM_color_buffer_float_rgb GL_CHROMIUM_color_buffer_float_rgba GL_CHROMIUM_copy_texture GL_CHROMIUM_lose_context GL_CHROMIUM_sync_query GL_EXT_base_instance GL_EXT_blend_minmax GL_EXT_color_buffer_half_float GL_EXT_compressed_ETC1_RGB8_sub_texture GL_EXT_debug_label GL_EXT_debug_marker GL_EXT_discard_framebuffer GL_EXT_disjoint_timer_query GL_EXT_draw_buffers GL_EXT_draw_elements_base_vertex GL_EXT_float_blend GL_EXT_frag_depth GL_EXT_instanced_arrays GL_EXT_map_buffer_range GL_EXT_memory_object

	GL_EXT_memory_object_fd GL_EXT_multi_draw_indirect GL_EXT_multisample_compatibility GL_EXT_occlusion_query_boolean GL_EXT_read_format_bgra GL_EXT_robustness GL_EXT_sRGB GL_EXT_sRGB_write_control GL_EXT_shader_texture_lod GL_EXT_shadow_samplers GL_EXT_texture_compression_bptc GL_EXT_texture_compression_dxt1 GL_EXT_texture_compression_rgtc GL_EXT_texture_compression_s3tc_srgb GL_EXT_texture_filter_anisotropic GL_EXT_texture_format_BGRA8888 GL_EXT_texture_norm16 GL_EXT_texture_rg GL_EXT_texture_sRGB_decode GL_EXT_texture_storage GL_EXT_texture_type_2_10_10_10_REV GL_EXT_unpack_subimage GL_KHR_debug GL_KHR_parallel_shader_compile GL_KHR_texture_compression_astc_ldr GL_KHR_texture_compression_astc_sliced_3d GL_MESA_framebuffer_flip_y GL_NV_depth_buffer_float2 GL_NV_fence GL_NV_framebuffer_blit GL_NV_pack_subimage GL_NV_pixel_buffer_object GL_NV_read_depth GL_NV_read_stencil GL_OES_compressed_EAC_R11_signed_texture GL_OES_compressed_EAC_R11_unsigned_texture GL_OES_compressed_EAC_RG11_signed_texture GL_OES_compressed_EAC_RG11_unsigned_texture GL_OES_compressed_ETC1_RGB8_texture GL_OES_compressed_ETC2_RGB8_texture GL_OES_compressed_ETC2_RGBA8_texture GL_OES_compressed_ETC2_punchthroughA_RGBA8_texture GL_OES_compressed_ETC2_punchthroughA_sRGB8_alpha_texture GL_OES_compressed_ETC2_sRGB8_alpha8_texture GL_OES_compressed_ETC2_sRGB8_texture GL_OES_depth24 GL_OES_depth32 GL_OES_depth_texture GL_OES_draw_elements_base_vertex GL_OES_element_index_uint GL_OES_fbo_render_mipmap GL_OES_get_program_binary GL_OES_mapbuffer GL_OES_packed_depth_stencil GL_OES_rgb8_rgba8 GL_OES_standard_derivatives GL_OES_surfaceless_context GL_OES_texture_3D GL_OES_texture_border_clamp GL_OES_texture_float GL_OES_texture_float_linear GL_OES_texture_half_float GL_OES_texture_half_float_linear GL_OES_texture_npot GL_OES_texture_stencil8 GL_OES_vertex_array_object GL_WEBGL_video_texture
Disabled Extensions	GL_KHR_blend_equation_advanced GL_KHR_blend_equation_advanced_coherent GL_MESA_framebuffer_flip_y
Disabled WebGL Extensions	
Window system binding vendor	Google Inc. (Mesa/X.org)
Window system binding version	1.5 (ANGLE 2.1.0 git hash: unknown hash)
Window system binding extensions	EGL_KHR_create_context EGL_KHR_get_all_proc_addresses EGL_ANGLE_create_context_webgl_compatibility EGL_CHROMIUM_create_context_bind_generates_resource EGL_EXT_pixel_format_float EGL_KHR_surfaceless_context EGL_ANGLE_display_texture_share_group EGL_ANGLE_display_semaphore_share_group

	EGL_ANGLE_create_context_client_arrays EGL_ANGLE_program_cache_control EGL_ANGLE_robust_resource_initialization EGL_ANGLE_create_context_extensions_enabled EGL_ANDROID_blob_cache EGL_ANDROID_recordable EGL_ANGLE_create_context_backwards_compatible EGL_KHR_create_context_no_error EGL_NOK_texture_from_pixmap EGL_KHR_reusable_sync
XDG_CURRENT_DESI	MATE
XDG_SESSION_TYPE	x11
GDMSESSION	mate
Ozone platform	x11
Direct rendering version	unknown
Reset notification strategy	0x8261
GPU process crash count	0
gfx::BufferFormats supported for allocation and texturing	R_8: not supported, R_16: not supported, RG_88: not supported, RG_1616: not supported, BGR_565: not supported, RGBA_4444: not supported, RGBX_8888: not supported, RGBA_8888: not supported, BGRX_8888: not supported, BGRA_1010102: not supported, RGBA_1010102: not supported, BGRA_8888: not supported, RGBA_F16: not supported, YVU_420: not supported, YUV_420_BIPLANAR: not supported, YUVA_420_TRIPLANAR: not supported, P010: not supported

Compositor Information

Tile Update Mode	One-copy
Partial Raster	Enabled

GpuMemoryBuffers Status

R_8	Software only
R_16	Software only
RG_88	Software only
RG_1616	Software only
BGR_565	Software only
RGBA_4444	Software only
RGBX_8888	Software only
RGBA_8888	Software only
BGRX_8888	Software only
BGRA_1010102	Software only
RGBA_1010102	Software only
BGRA_8888	Software only
RGBA_F16	Software only
YVU_420	Software only
YUV_420_BIPLANAR	Software only
YUVA_420_TRIPLANA	Software only
P010	Software only

Display(s) Information

Info	Display[0] bounds=[0,0 1920x1200], workarea=[0,0 1920x1169], scale=1, rotation=0, panel_rotation=0 external.
Color space (all)	{primaries:BT709, transfer:SRGB, matrix:RGB, range:FULL}

Buffer format (all)	BGRA_8888
Color volume	{name:'srgb', r:[0.6400, 0.3300], g:[0.3000, 0.6000], b:[0.1500, 0.3300], w:[0.3127, 0.3290]}
SDR white level in nits	203
HDR relative maximum luminance	1
Bits per color component	8
Bits per pixel	24

Video Acceleration Information

Decoding	
Encoding	

Vulkan Information

info	{ "apiVersion": "1.3.236", "usedApiVersion": "1.1.0", "instanceExtensions": { "VK_KHR_device_group_creation": 1, "VK_KHR_external_fence_capabilities": 1, "VK_KHR_external_memory_capabilities": 1, "VK_KHR_external_semaphore_capabilities": 1, "VK_KHR_get_physical_device_properties2": 2, "VK_KHR_get_surface_capabilities2": 1, "VK_KHR_surface": 25, "VK_KHR_surface_protected_capabilities": 1, "VK_KHR_wayland_surface": 6, "VK_KHR_xcb_surface": 6, "VK_KHR_xlib_surface": 6, "VK_EXT_debug_report": 10, "VK_EXT_debug_utils": 2, "VK_KHR_display": 23, "VK_KHR_get_display_properties2": 1, "VK_EXT_acquire_drm_display": 1, "VK_EXT_acquire_xlib_display": 1, "VK_EXT_direct_mode_display": 1, "VK_EXT_display_surface_counter": 1, "VK_KHR_portability_enumeration": 1 }, "enabledInstanceExtensions": ["VK_KHR_external_memory_capabilities", "VK_KHR_external_semaphore_capabilities", "VK_KHR_surface", "VK_KHR_xcb_surface", "VK_EXT_debug_report"], "instanceLayers": [{ "layerName": "VK_LAYER_VALVE_steam_fossilize_64", "specVersion": 4206799, "implementationVersion": "0.0.1", "description": "Steam Pipeline Caching Layer" }, { "layerName": "VK_LAYER_VALVE_steam_fossilize_32", "specVersion": 4206799, "implementationVersion": "0.0.1", "description": "Steam Pipeline Caching Layer" }, { "layerName": "VK_LAYER_VALVE_steam_overlay_32", "specVersion": 4206799, "implementationVersion": "0.0.1", "description": "Steam Overlay Layer" }, { "layerName": "VK_LAYER_VALVE_steam_overlay_64", "specVersion": 4206799, "implementationVersion": "0.0.1", "description": "Steam Overlay Layer" }, { "layerName": "VK_LAYER_MESA_device_select", "specVersion": 4206803, "implementationVersion": "0.0.1", "description": "Linux device selection layer" }, { "layerName": "VK_LAYER_MESA_overlay", "specVersion": 4206803, "implementationVersion": "0.0.1", "description": "Mesa Overlay layer" }], "physicalDevices": [{ "properties": { "apiVersion": "1.3.230", "driverVersion": "22.3.6", "vendorID": 32902, "deviceID": 18086, "deviceType": 1, "deviceName": "Intel(R) Graphics (ADL GT2)", "pipelineCacheUUID": "dddf85b3-eef3-b44c-b6ea-27e2ff6b5b01", "limits": { "maxImageDimension1D": 16384, "maxImageDimension2D": 16384, "maxImageDimension3D": 2048, "maxImageDimensionCube": 16384, "maxImageArrayLayers": 2048, "maxTexelBufferElements": 134217728, "maxUniformBufferRange": 1073741824,

"maxStorageBufferRange": 1073741824, "maxPushConstantsSize": 128, "maxMemoryAllocationCount": 4294967295, "maxSamplerAllocationCount": 65536, "bufferImageGranularity": 1, "sparseAddressSpaceSize": 0, "maxBoundDescriptorSets": 32, "maxPerStageDescriptorSamplers": 65535, "maxPerStageDescriptorUniformBuffers": 64, "maxPerStageDescriptorStorageBuffers": 65535, "maxPerStageDescriptorSampledImages": 65535, "maxPerStageDescriptorStorageImages": 65535, "maxPerStageDescriptorInputAttachments": 64, "maxPerStageResources": 4294967295, "maxDescriptorSetSamplers": 393210, "maxDescriptorSetUniformBuffers": 384, "maxDescriptorSetUniformBuffersDynamic": 8, "maxDescriptorSetStorageBuffers": 393210, "maxDescriptorSetStorageBuffersDynamic": 8, "maxDescriptorSetSampledImages": 393210, "maxDescriptorSetStorageImages": 393210, "maxDescriptorSetInputAttachments": 256, "maxVertexInputAttributes": 29, "maxVertexInputBindings": 31, "maxVertexInputAttributeOffset": 2047, "maxVertexInputBindingStride": 4095, "maxVertexOutputComponents": 128, "maxTessellationGenerationLevel": 64, "maxTessellationPatchSize": 32, "maxTessellationControlPerVertexInputComponents": 128, "maxTessellationControlPerVertexOutputComponents": 128, "maxTessellationControlPerPatchOutputComponents": 128, "maxTessellationControlTotalOutputComponents": 2048, "maxTessellationEvaluationInputComponents": 128, "maxTessellationEvaluationOutputComponents": 128, "maxGeometryShaderInvocations": 32, "maxGeometryInputComponents": 128, "maxGeometryOutputComponents": 128, "maxGeometryOutputVertices": 256, "maxGeometryTotalOutputComponents": 1024, "maxFragmentInputComponents": 116, "maxFragmentOutputAttachments": 8, "maxFragmentDualSrcAttachments": 1, "maxFragmentCombinedOutputResources": 131078, "maxComputeSharedMemorySize": 65536, "maxComputeWorkGroupCount": [65535, 65535, 65535], "maxComputeWorkGroupInvocations": 1024, "maxComputeWorkGroupSize": [1024, 1024, 1024], "subPixelPrecisionBits": 8, "subTexelPrecisionBits": 8, "mipmapPrecisionBits": 8, "maxDrawIndexedIndexValue": 4294967295, "maxDrawIndirectCount": 4294967295, "maxSamplerLodBias": 16, "maxSamplerAnisotropy": 16, "maxViewports": 16, "maxViewportDimensions": [16384, 16384], "viewportBoundsRange": [-32768, 32767], "viewportSubPixelBits": 13, "minMemoryMapAlignment": 4096, "minTexelBufferOffsetAlignment": 1, "minUniformBufferOffsetAlignment": 1, "minStorageBufferOffsetAlignment": 1, "minTexelOffset": -8, "maxTexelOffset": 7, "minTexelGatherOffset": -32, "maxTexelGatherOffset": 31, "minInterpolationOffset": -0.5, "maxInterpolationOffset": 0.4375, "subPixelInterpolationOffsetBits": 4, "maxFramebufferWidth": 16384, "maxFramebufferHeight": 16384, "maxFramebufferLayers": 2048, "framebufferColorSampleCounts": 31, "framebufferDepthSampleCounts": 31, "framebufferStencilSampleCounts": 31, "framebufferNoAttachmentsSampleCounts": 31,

"maxColorAttachments": 8, "sampledImageColorSampleCounts": 31,
"sampledImageIntegerSampleCounts": 31,
"sampledImageDepthSampleCounts": 31,
"sampledImageStencilSampleCounts": 31,
"storageImageSampleCounts": 1, "maxSampleMaskWords": 1,
"timestampComputeAndGraphics": true, "timestampPeriod":
52.08333206176758, "maxClipDistances": 8, "maxCullDistances": 8,
"maxCombinedClipAndCullDistances": 8, "discreteQueuePriorities": 2,
"pointSizeRange": [0.125, 255.875], "lineWidthRange": [0, 8],
"pointSizeGranularity": 0.125, "lineWidthGranularity": 0.0078125,
"strictLines": false, "standardSampleLocations": true,
"optimalBufferCopyOffsetAlignment": 1,
"optimalBufferCopyRowPitchAlignment": 1, "nonCoherentAtomSize": 1
}, "sparseProperties": { "residencyStandard2DBlockShape": false,
"residencyStandard2DMultisampleBlockShape": false,
"residencyStandard3DBlockShape": false, "residencyAlignedMipSize":
false, "residencyNonResidentStrict": false } }, "extensions": {
"VK_KHR_8bit_storage": 1, "VK_KHR_16bit_storage": 1,
"VK_KHR_bind_memory2": 1, "VK_KHR_buffer_device_address": 1,
"VK_KHR_copy_commands2": 1, "VK_KHR_create_renderpass2": 1,
"VK_KHR_dedicated_allocation": 3,
"VK_KHR_deferred_host_operations": 4,
"VK_KHR_depth_stencil_resolve": 1,
"VK_KHR_descriptor_update_template": 1, "VK_KHR_device_group":
4, "VK_KHR_draw_indirect_count": 1, "VK_KHR_driver_properties": 1,
"VK_KHR_dynamic_rendering": 1, "VK_KHR_external_fence": 1,
"VK_KHR_external_fence_fd": 1, "VK_KHR_external_memory": 1,
"VK_KHR_external_memory_fd": 1, "VK_KHR_external_semaphore":
1, "VK_KHR_external_semaphore_fd": 1,
"VK_KHR_format_feature_flags2": 2,
"VK_KHR_fragment_shading_rate": 2,
"VK_KHR_get_memory_requirements2": 1,
"VK_KHR_image_format_list": 1, "VK_KHR_imageless_framebuffer": 1,
"VK_KHR_incremental_present": 2, "VK_KHR_maintenance1": 2,
"VK_KHR_maintenance2": 1, "VK_KHR_maintenance3": 1,
"VK_KHR_maintenance4": 2, "VK_KHR_multiview": 1,
"VK_KHR_pipeline_executable_properties": 1,
"VK_KHR_pipeline_library": 1, "VK_KHR_push_descriptor": 2,
"VK_KHR_relaxed_block_layout": 1,
"VK_KHR_sampler_mirror_clamp_to_edge": 3,
"VK_KHR_sampler_ycbcr_conversion": 14,
"VK_KHR_separate_depth_stencil_layouts": 1,
"VK_KHR_shader_atomic_int64": 1, "VK_KHR_shader_clock": 1,
"VK_KHR_shader_draw_parameters": 1,
"VK_KHR_shader_float16_int8": 1, "VK_KHR_shader_float_controls":
4, "VK_KHR_shader_integer_dot_product": 1,
"VK_KHR_shader_non_semantic_info": 1,
"VK_KHR_shader_subgroup_extended_types": 1,
"VK_KHR_shader_subgroup_uniform_control_flow": 1,
"VK_KHR_shader_terminate_invocation": 1, "VK_KHR_spirv_1_4": 1,
"VK_KHR_storage_buffer_storage_class": 1, "VK_KHR_swapchain":
70, "VK_KHR_swapchain_mutable_format": 1,
"VK_KHR_synchronization2": 1, "VK_KHR_timeline_semaphore": 2,
"VK_KHR_uniform_buffer_standard_layout": 1,
"VK_KHR_variable_pointers": 1, "VK_KHR_vulkan_memory_model": 3,
"VK_KHR_workgroup_memory_explicit_layout": 1,
"VK_KHR_zero_initialize_workgroup_memory": 1,
"VK_EXT_4444_formats": 1, "VK_EXT_border_color_swizzle": 1,

"VK_EXT_buffer_device_address": 2,
"VK_EXT_calibrated_timestamps": 2, "VK_EXT_color_write_enable":
1, "VK_EXT_conditional_rendering": 2,
"VK_EXT_conservative_rasterization": 1,
"VK_EXT_custom_border_color": 12,
"VK_EXT_depth_clamp_zero_one": 1, "VK_EXT_depth_clip_control":
1, "VK_EXT_depth_clip_enable": 1, "VK_EXT_descriptor_indexing": 2,
"VK_EXT_display_control": 1, "VK_EXT_extended_dynamic_state": 1,
"VK_EXT_extended_dynamic_state2": 1,
"VK_EXT_extended_dynamic_state3": 2,
"VK_EXT_external_memory_dma_buf": 1,
"VK_EXT_external_memory_host": 1,
"VK_EXT_fragment_shader_interlock": 1, "VK_EXT_global_priority": 2,
"VK_EXT_global_priority_query": 1, "VK_EXT_host_query_reset": 1,
"VK_EXT_image_2d_view_of_3d": 1,
"VK_EXT_image_drm_format_modifier": 2,
"VK_EXT_image_robustness": 1, "VK_EXT_image_view_min_lod": 1,
"VK_EXT_index_type_uint8": 1, "VK_EXT_inline_uniform_block": 1,
"VK_EXT_line_rasterization": 1, "VK_EXT_memory_budget": 1,
"VK_EXT_multi_draw": 1, "VK_EXT_mutable_descriptor_type": 1,
"VK_EXT_non_seamless_cube_map": 1, "VK_EXT_pci_bus_info": 2,
"VK_EXT_physical_device_drm": 1,
"VK_EXT_pipeline_creation_cache_control": 3,
"VK_EXT_pipeline_creation_feedback": 1,
"VK_EXT_post_depth_coverage": 1,
"VK_EXT_primitive_topology_list_restart": 1,
"VK_EXT_primitives_generated_query": 1, "VK_EXT_private_data": 1,
"VK_EXT_provoking_vertex": 1, "VK_EXT_queue_family_foreign": 1,
"VK_EXT_robustness2": 1, "VK_EXT_sample_locations": 1,
"VK_EXT_sampler_filter_minmax": 2, "VK_EXT_scalar_block_layout":
1, "VK_EXT_separate_stencil_usage": 1,
"VK_EXT_shader_atomic_float": 1, "VK_EXT_shader_atomic_float2":
1, "VK_EXT_shader_demote_to_helper_invocation": 1,
"VK_EXT_shader_module_identifier": 1,
"VK_EXT_shader_stencil_export": 1,
"VK_EXT_shader_subgroup_ballot": 1,
"VK_EXT_shader_subgroup_vote": 1,
"VK_EXT_shader_viewport_index_layer": 1,
"VK_EXT_subgroup_size_control": 2,
"VK_EXT_texel_buffer_alignment": 1, "VK_EXT_tooling_info": 1,
"VK_EXT_transform_feedback": 1, "VK_EXT_vertex_attribute_divisor":
3, "VK_EXT_ycbcr_image_arrays": 1, "VK_GOOGLE_decorate_string":
1, "VK_GOOGLE_hlsl_functionality1": 1, "VK_GOOGLE_user_type": 1,
"VK_INTEL_shader_integer_functions2": 1,
"VK_NV_compute_shader_derivatives": 1,
"VK_VALVE_mutable_descriptor_type": 1 }, "features": {
"robustBufferAccess": true, "fullDrawIndexUint32": true,
"imageCubeArray": true, "independentBlend": true, "geometryShader":
true, "tessellationShader": true, "sampleRateShading": true,
"dualSrcBlend": true, "logicOp": true, "multiDrawIndirect": true,
"drawIndirectFirstInstance": true, "depthClamp": true,
"depthBiasClamp": true, "fillModeNonSolid": true, "depthBounds": true,
"wideLines": true, "largePoints": true, "alphaToOne": true,
"multiViewport": true, "samplerAnisotropy": true,
"textureCompressionETC2": true, "textureCompressionASTCLDR":
true, "textureCompressionBC": true, "occlusionQueryPrecise": true,
"pipelineStatisticsQuery": true, "vertexPipelineStoresAndAtomics": true,
"fragmentStoresAndAtomics": true,

"shaderTessellationAndGeometryPointSize": true,
"shaderImageGatherExtended": true,
"shaderStorageImageExtendedFormats": true,
"shaderStorageImageMultisample": false,
"shaderStorageImageReadWithoutFormat": false,
"shaderStorageImageWriteWithoutFormat": true,
"shaderUniformBufferArrayDynamicIndexing": true,
"shaderSampledImageArrayDynamicIndexing": true,
"shaderStorageBufferArrayDynamicIndexing": true,
"shaderStorageImageArrayDynamicIndexing": true,
"shaderClipDistance": true, "shaderCullDistance": true,
"shaderFloat64": false, "shaderInt64": true, "shaderInt16": true,
"shaderResourceResidency": false, "shaderResourceMinLod": true,
"sparseBinding": false, "sparseResidencyBuffer": false,
"sparseResidencyImage2D": false, "sparseResidencyImage3D": false,
"sparseResidency2Samples": false, "sparseResidency4Samples":
false, "sparseResidency8Samples": false,
"sparseResidency16Samples": false, "sparseResidencyAliased": false,
"variableMultisampleRate": true, "inheritedQueries": true },
"featureSamplerYcbcrConversion": true, "featureProtectedMemory":
false, "queueFamilies": [{ "queueFlags": 7, "queueCount": 1,
"timestampValidBits": 36, "minImageTransferGranularity": { "width": 1,
"height": 1, "depth": 1 } }] }, { "properties": { "apiVersion": "1.3.230",
"driverVersion": "0.0.1", "vendorID": 65541, "deviceID": 0, "deviceType":
4, "deviceName": "llvmpipe (LLVM 11.0.1, 256 bits)",
"pipelineCacheUUID": "76616c2d-2573-0000-0000-000000000000",
"limits": { "maxImageDimension1D": 16384, "maxImageDimension2D":
16384, "maxImageDimension3D": 4096, "maxImageDimensionCube":
32768, "maxImageArrayLayers": 2048, "maxTexelBufferElements":
134217728, "maxUniformBufferRange": 65536,
"maxStorageBufferRange": 134217728, "maxPushConstantsSize": 128,
"maxMemoryAllocationCount": 4294967295,
"maxSamplerAllocationCount": 32768, "bufferImageGranularity": 1,
"sparseAddressSpaceSize": 0, "maxBoundDescriptorSets": 8,
"maxPerStageDescriptorSamplers": 32,
"maxPerStageDescriptorUniformBuffers": 15,
"maxPerStageDescriptorStorageBuffers": 32,
"maxPerStageDescriptorSampledImages": 128,
"maxPerStageDescriptorStorageImages": 64,
"maxPerStageDescriptorInputAttachments": 8,
"maxPerStageResources": 128, "maxDescriptorSetSamplers": 32768,
"maxDescriptorSetUniformBuffers": 256,
"maxDescriptorSetUniformBuffersDynamic": 256,
"maxDescriptorSetStorageBuffers": 256,
"maxDescriptorSetStorageBuffersDynamic": 256,
"maxDescriptorSetSampledImages": 256,
"maxDescriptorSetStorageImages": 256,
"maxDescriptorSetInputAttachments": 256, "maxVertexInputAttributes":
32, "maxVertexInputBindings": 32, "maxVertexInputAttributeOffset":
2047, "maxVertexInputBindingStride": 2048,
"maxVertexOutputComponents": 128,
"maxTessellationGenerationLevel": 64, "maxTessellationPatchSize": 32,
"maxTessellationControlPerVertexInputComponents": 128,
"maxTessellationControlPerVertexOutputComponents": 128,
"maxTessellationControlPerPatchOutputComponents": 128,
"maxTessellationControlTotalOutputComponents": 4096,
"maxTessellationEvaluationInputComponents": 128,
"maxTessellationEvaluationOutputComponents": 128,

"maxGeometryShaderInvocations": 32,
"maxGeometryInputComponents": 64,
"maxGeometryOutputComponents": 128,
"maxGeometryOutputVertices": 1024,
"maxGeometryTotalOutputComponents": 1024,
"maxFragmentInputComponents": 128,
"maxFragmentOutputAttachments": 8,
"maxFragmentDualSrcAttachments": 2,
"maxFragmentCombinedOutputResources": 104,
"maxComputeSharedMemorySize": 32768,
"maxComputeWorkGroupCount": [65535, 65535, 65535],
"maxComputeWorkGroupInvocations": 1024,
"maxComputeWorkGroupSize": [1024, 1024, 1024],
"subPixelPrecisionBits": 8, "subTexelPrecisionBits": 8,
"mipmapPrecisionBits": 4, "maxDrawIndexedIndexValue": 4294967295,
"maxDrawIndirectCount": 4294967295, "maxSamplerLodBias": 16,
"maxSamplerAnisotropy": 16, "maxViewports": 16,
"maxViewportDimensions": [16384, 16384], "viewportBoundsRange": [
-32768, 32768], "viewportSubPixelBits": 0,
"minMemoryMapAlignment": 64, "minTexelBufferOffsetAlignment": 1,
"minUniformBufferOffsetAlignment": 1,
"minStorageBufferOffsetAlignment": 1, "minTexelOffset": -32,
"maxTexelOffset": 31, "minTexelGatherOffset": -32,
"maxTexelGatherOffset": 31, "minInterpolationOffset": -2,
"maxInterpolationOffset": 2, "subPixelInterpolationOffsetBits": 8,
"maxFramebufferWidth": 16384, "maxFramebufferHeight": 16384,
"maxFramebufferLayers": 2048, "framebufferColorSampleCounts": 5,
"framebufferDepthSampleCounts": 5,
"framebufferStencilSampleCounts": 5,
"framebufferNoAttachmentsSampleCounts": 5, "maxColorAttachments":
8, "sampledImageColorSampleCounts": 5,
"sampledImageIntegerSampleCounts": 5,
"sampledImageDepthSampleCounts": 5,
"sampledImageStencilSampleCounts": 5,
"storageImageSampleCounts": 5, "maxSampleMaskWords": 1,
"timestampComputeAndGraphics": true, "timestampPeriod": 1,
"maxClipDistances": 8, "maxCullDistances": 8,
"maxCombinedClipAndCullDistances": 8, "discreteQueuePriorities": 2,
"pointSizeRange": [0, 255], "lineWidthRange": [1, 255],
"pointSizeGranularity": 0.125, "lineWidthGranularity": 0.0078125,
"strictLines": true, "standardSampleLocations": true,
"optimalBufferCopyOffsetAlignment": 1,
"optimalBufferCopyRowPitchAlignment": 1, "nonCoherentAtomSize": 1
}, "sparseProperties": { "residencyStandard2DBlockShape": false,
"residencyStandard2DMultisampleBlockShape": false,
"residencyStandard3DBlockShape": false, "residencyAlignedMipSize":
false, "residencyNonResidentStrict": false } }, "extensions": {
"VK_KHR_8bit_storage": 1, "VK_KHR_16bit_storage": 1,
"VK_KHR_bind_memory2": 1, "VK_KHR_buffer_device_address": 1,
"VK_KHR_copy_commands2": 1, "VK_KHR_create_renderpass2": 1,
"VK_KHR_dedicated_allocation": 3, "VK_KHR_depth_stencil_resolve":
1, "VK_KHR_descriptor_update_template": 1,
"VK_KHR_device_group": 4, "VK_KHR_draw_indirect_count": 1,
"VK_KHR_driver_properties": 1, "VK_KHR_dynamic_rendering": 1,
"VK_KHR_external_fence": 1, "VK_KHR_external_memory": 1,
"VK_KHR_external_memory_fd": 1, "VK_KHR_external_semaphore":
1, "VK_KHR_format_feature_flags2": 2,
"VK_KHR_get_memory_requirements2": 1,

"VK_KHR_image_format_list": 1, "VK_KHR_imageless_framebuffer": 1,
"VK_KHR_incremental_present": 2, "VK_KHR_maintenance1": 2,
"VK_KHR_maintenance2": 1, "VK_KHR_maintenance3": 1,
"VK_KHR_maintenance4": 2, "VK_KHR_multiview": 1,
"VK_KHR_pipeline_library": 1, "VK_KHR_push_descriptor": 2,
"VK_KHR_relaxed_block_layout": 1,
"VK_KHR_sampler_mirror_clamp_to_edge": 3,
"VK_KHR_separate_depth_stencil_layouts": 1,
"VK_KHR_shader_atomic_int64": 1, "VK_KHR_shader_clock": 1,
"VK_KHR_shader_draw_parameters": 1,
"VK_KHR_shader_float16_int8": 1, "VK_KHR_shader_float_controls":
4, "VK_KHR_shader_integer_dot_product": 1,
"VK_KHR_shader_subgroup_extended_types": 1,
"VK_KHR_shader_terminate_invocation": 1, "VK_KHR_spirv_1_4": 1,
"VK_KHR_storage_buffer_storage_class": 1, "VK_KHR_swapchain":
70, "VK_KHR_swapchain_mutable_format": 1,
"VK_KHR_synchronization2": 1, "VK_KHR_timeline_semaphore": 2,
"VK_KHR_uniform_buffer_standard_layout": 1,
"VK_KHR_variable_pointers": 1, "VK_KHR_vulkan_memory_model": 3,
"VK_KHR_zero_initialize_workgroup_memory": 1,
"VK_EXT_4444_formats": 1,
"VK_EXT_attachment_feedback_loop_layout": 2,
"VK_EXT_border_color_swizzle": 1, "VK_EXT_calibrated_timestamps":
2, "VK_EXT_color_write_enable": 1, "VK_EXT_conditional_rendering":
2, "VK_EXT_custom_border_color": 12, "VK_EXT_depth_clip_control":
1, "VK_EXT_depth_clip_enable": 1,
"VK_EXT_depth_range_unrestricted": 1,
"VK_EXT_extended_dynamic_state": 1,
"VK_EXT_extended_dynamic_state2": 1,
"VK_EXT_extended_dynamic_state3": 2,
"VK_EXT_external_memory_host": 1,
"VK_EXT_graphics_pipeline_library": 1, "VK_EXT_host_query_reset":
1, "VK_EXT_image_2d_view_of_3d": 1, "VK_EXT_image_robustness":
1, "VK_EXT_index_type_uint8": 1, "VK_EXT_inline_uniform_block": 1,
"VK_EXT_line_rasterization": 1, "VK_EXT_multi_draw": 1,
"VK_EXT_multisampled_render_to_single_sampled": 1,
"VK_EXT_non_seamless_cube_map": 1,
"VK_EXT_pipeline_creation_cache_control": 3,
"VK_EXT_pipeline_creation_feedback": 1,
"VK_EXT_post_depth_coverage": 1,
"VK_EXT_primitive_topology_list_restart": 1,
"VK_EXT_primitives_generated_query": 1, "VK_EXT_private_data": 1,
"VK_EXT_provoking_vertex": 1,
"VK_EXT_rasterization_order_attachment_access": 1,
"VK_EXT_robustness2": 1, "VK_EXT_sampler_filter_minmax": 2,
"VK_EXT_scalar_block_layout": 1, "VK_EXT_separate_stencil_usage":
1, "VK_EXT_shader_atomic_float": 1,
"VK_EXT_shader_atomic_float2": 1,
"VK_EXT_shader_demote_to_helper_invocation": 1,
"VK_EXT_shader_stencil_export": 1,
"VK_EXT_shader_subgroup_ballot": 1,
"VK_EXT_shader_subgroup_vote": 1,
"VK_EXT_shader_viewport_index_layer": 1,
"VK_EXT_subgroup_size_control": 2,
"VK_EXT_texel_buffer_alignment": 1, "VK_EXT_transform_feedback":
1, "VK_EXT_vertex_attribute_divisor": 3,
"VK_EXT_vertex_input_dynamic_state": 2,
"VK_ARM_rasterization_order_attachment_access": 1,

	"VK_GOOGLE_decorate_string": 1, "VK_GOOGLE_hlsl_functionality1": 1 }, "features": { "robustBufferAccess": true, "fullDrawIndexUint32": true, "imageCubeArray": true, "independentBlend": true, "geometryShader": true, "tessellationShader": true, "sampleRateShading": true, "dualSrcBlend": true, "logicOp": true, "multiDrawIndirect": true, "drawIndirectFirstInstance": true, "depthClamp": true, "depthBiasClamp": true, "fillModeNonSolid": true, "depthBounds": false, "wideLines": true, "largePoints": true, "alphaToOne": true, "multiViewport": true, "samplerAnisotropy": true, "textureCompressionETC2": false, "textureCompressionASTCLDR": false, "textureCompressionBC": true, "occlusionQueryPrecise": true, "pipelineStatisticsQuery": true, "vertexPipelineStoresAndAtomics": true, "fragmentStoresAndAtomics": true, "shaderTessellationAndGeometryPointSize": true, "shaderImageGatherExtended": true, "shaderStorageImageExtendedFormats": true, "shaderStorageImageMultisample": true, "shaderStorageImageReadWithoutFormat": false, "shaderStorageImageWriteWithoutFormat": true, "shaderUniformBufferArrayDynamicIndexing": true, "shaderSampledImageArrayDynamicIndexing": false, "shaderStorageBufferArrayDynamicIndexing": true, "shaderStorageImageArrayDynamicIndexing": false, "shaderClipDistance": true, "shaderCullDistance": true, "shaderFloat64": true, "shaderInt64": true, "shaderInt16": true, "shaderResourceResidency": false, "shaderResourceMinLod": false, "sparseBinding": false, "sparseResidencyBuffer": false, "sparseResidencyImage2D": false, "sparseResidencyImage3D": false, "sparseResidency2Samples": false, "sparseResidency4Samples": false, "sparseResidency8Samples": false, "sparseResidency16Samples": false, "sparseResidencyAliased": false, "variableMultisampleRate": false, "inheritedQueries": false }, "featureSamplerYcbcrConversion": false, "featureProtectedMemory": false, "queueFamilies": [{ "queueFlags": 7, "queueCount": 1, "timestampValidBits": 64, "minImageTransferGranularity": { "width": 1, "height": 1, "depth": 1 } }] }] }

Device Performance Information

Log Messages

- [7509:7509:0519/100008.933899:WARNING:sandbox_linux.cc(393)] : InitializeSandbox() called with multiple threads in process gpu-process.
- [7509:7509:0519/100008.934891:ERROR:vulkan_device_queue.cc(234)] : samplerYcbcrConversion is not supported.
- [7509:7509:0519/100008.935792:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100008.935872:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100008.935940:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100008.935959:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100008.986984:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100008.987037:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState

- [7509:7509:0519/100008.987266:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100008.987307:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100008.990809:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100008.990872:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100008.991017:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100008.991060:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100009.029599:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100009.029665:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100009.029788:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100009.029823:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100009.030325:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100009.030476:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100009.030651:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100009.030700:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100009.511184:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100009.511316:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100009.511512:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100009.511654:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100009.519180:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100009.519350:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100009.519584:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100009.519625:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100010.642157:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100010.642470:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100010.642717:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100010.642770:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100010.656984:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100010.657101:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100010.657324:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context

- [7509:7509:0519/100010.657485:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100012.216657:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100012.217000:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100012.217337:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100012.217529:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100012.229469:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100012.229591:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100012.229679:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100012.229707:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100015.840849:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100015.841062:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100015.841356:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100015.841417:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100015.849622:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100015.849722:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100015.849810:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100015.849840:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100026.110843:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100026.111008:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100026.111160:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100026.111260:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub
- [7509:7509:0519/100026.120929:ERROR:shared_context_state.cc(363)] : OOP raster support disabled: GrContext creation failed.
- [7509:7509:0519/100026.121115:ERROR:gpu_channel_manager.cc(974)] : ContextResult::kFatalFailure: Failed to InitializeGrContext for SharedContextState
- [7509:7509:0519/100026.121360:ERROR:shared_image_stub.cc(475)] : SharedImageStub: unable to create context
- [7509:7509:0519/100026.121459:ERROR:gpu_channel.cc(589)] : GpuChannel: Failed to create SharedImageStub