VK_QCOM_render_pass_transform Extension

Developer Guide

80-PT676-1 Rev. B

May 29, 2020

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	November 2019	Initial release
B	May 2020	Updated 4.1 “How to use the extension” Updated A “Extension definition”

Contents

1 Introduction

1.1 Purpose

This document describes a new vendor extension from Qualcomm® that reduces the burden on the application in handling pre-rotation in Vulkan. A general technical overview is also provided for better understanding. In the end, the extension specification is listed, and its usage illustrated.

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered appear in a different font, for example, **`copy a:*.* b:`**.

Button and key names appear in bold font, for example, click **Save** or press **Enter**.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.Qualcomm.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.Qualcomm.com.

2 Extension overview

When doing pre-rotation, the extension `VK_QCOM_render_pass_transform` frees the applications from some of the burden required.

A general information and background of is described, along with the changes provided by the implementation of the extension.

2.1 Pre-rotation

Mobile devices can be rotated, and mobile applications need to render properly when a device is held in a landscape or portrait orientation. When the current (landscape) orientation differs from the device’s native (portrait) orientation, a rotation is required so that the "up" direction of the rendered scene matches the current orientation.

The rotation can be handled differently. The **Vulkan** presentation engine can handle this rotation in a separate composition pass. Alternatively, the application can render frames "pre-rotated" to avoid this extra pass. The latter is preferred to reduce power consumption and achieve the best performance because it avoids tasking the GPU with extra work to perform the copy/rotate operation.

Unlike OpenGL ES, which entails the burden on the driver, the pre-rotation burden in Vulkan falls on the application. To implement pre-rotation, the application renders into swapchain images matching the device’s native aspect ratio of the display and then "pre-rotates" the rendering content to match the device’s current orientation.

However, this pre-rotation burden is greater than adjusting the Model View Projection (MVP) matrix in the vertex shader to account for rotation and aspect ratio. The coordinate systems of scissors, viewports, derivatives and several shader built-ins may need to be adapted to produce the correct result. It is difficult for some game engines to manage this burden; many choose to simply accept the performance/power overhead of performing rotation in the presentation engine.

2.2 Implementation of the extension

This extension allows applications to achieve the performance benefits of pre-rotated rendering by moving much of the above-mentioned burden to the graphics driver.

The following is unchanged:

- Applications create a swapchain matching the native orientation of the display. Applications must also set the `VkSwapchainCreateInfoKHR::preTransform` equal to the `currentTransform` as returned by `vkGetPhysicalDeviceSurfaceCapabilitiesKHR`.

The following are changes with this extension:

- At `vkCmdBeginRenderpass`, the application provides extension struct `VkRenderPassTransformBeginInfoQCOM` specifying the render pass transform parameters.

- At `vkBeginCommandBuffer` for secondary command buffers, the application provides extension struct `VkCommandBufferInheritanceRenderPassTransformInfoQCOM` specifying the render pass transform parameters.
- The renderArea, viewPorts and scissors are all provided in the current (non-rotated) coordinate system. The implementation will transform those into the native (rotated) coordinate system.
- The implementation is responsible for transforming shader built-ins (`FragCoord`, `PointCoord`, `SamplePosition`, `interpolateAt()`, `dFdx`, `dFdy`, `fWidth`) into the rotated coordinate system.
- The implementation is responsible for transforming the position to the rotated coordinate system.

2.3 Proposal

Rather than modifying their shader pipelines or vertex data, applications can leverage this extension from Qualcomm.

3 Extension specifications

Name String	VK_QCOM_render_pass_transform
Extension Type	Device extension

New Enum Constants	▪ Extending VkStructureType:
	`VK_STRUCTURE_TYPE_RENDER_PASS_TRANSFORM_BEGIN_INFO_QCOM`
	`VK_STRUCTURE_TYPE_COMMAND_BUFFER_INHERITANCE_RENDER_PASS_TRANSFORM_INFO_QCOM`
	▪ Extending VkRenderPassCreateFlagBits:
	`VK_RENDER_PASS_CREATE_TRANSFORM_BIT_QCOM`

New Structures	`VkRenderPassTransformBeginInfoQCOM` `VkCommandBufferInheritanceRenderPassTransformInfoQCOM`
New Functions	None

4 Extension usage

4.1 How to use the extension

Follow these steps to use the extension:

1. Enable "VK_QCOM_RENDER_PASS_TRANSFORM_EXTENSION_NAME" extension when creating device (`vkCreateDevice`).
2. Create rotated swapchain if `currentTransform` as returned by `vkGetPhysicalDeviceSurfaceCapabilitiesKHR` is 90 or 270.
3. Swap width and height.
4. Set VkSwapchainCreateInfoKHR::preTransform equal to the currentTransform.
5. Set `VK_RENDER_PASS_CREATE_TRANSFORM_BIT_QCOM` flags when creating the render pass.
6. Create Framebuffer with the swapped width and height.
7. Provide the render pass transform info to driver at record time.

- Via `VkRenderPassTransformBeginInfoQCOM` in the call to `vkCmdBeginRenderPass` for primaries and secondaries outside a render pass.
- Via `VkCommandBufferInheritanceRenderPassTransformInfoQCOM` for secondary command buffers entirely in a Render Pass (e.g. `VK_COMMAND_BUFFER_USAGE_RENDER_PASS_CONTINUE_BIT` is set)
- The transform in `VkRenderPassTransformBeginInfoQCOM` and `VkCommandBufferInheritanceRenderPassTransformInfoQCOM` must be consistent with the preTransform specified on creating the swapchain.

4.2 Validity check

To ensure valid usage of the extension, ensure the following:

- RenderPass's renderArea must be equal to the unrotated Framebuffer dimensions with renderArea::offset equal to (0,0)

Notes

- The driver will rotate state as needed during a render pass. The application will need to handle rotation outside of a render pass (e.g., image clears, copies, blits, compute, etc.).
- If the rotation angle changes, all command buffers (primary & secondary) will need to be re-recorded.
- The transform will be applied to the back framebuffer's color and depth attachments used in the render pass. For example, a depth buffer used with the on-screen color buffer ("swap image") will need to have matching extents.

4.3 Simple test

- Tested a simple native application on the SDM730 device with GPU frequency fixed to 355 MHz. A performance improvement of 6.6% was achieved.

	GPU FPS	**GPU time**	**GPU composition**
Enable `VK_QCOM_render_pass_transform`	59.8	16.7ms	/
Disable `VK_QCOM_render_pass_transform`	56.3	17.8ms	2.2ms

A Extension definition

- The extension was published on 2020-02-05, developers can search and get the definitions on khronos vkspec from https://www.khronos.org/registry/vulkan/specs/1.0-extensions/html/vkspec.html.
- To cover all the QCOM devices in the market, it is recommended to use `VK_STRUCTURE_TYPE_RENDER_PASS_TRANSFORM_BEGIN_INFO_QCOM = 1000282000` when driverVersion <=#468, and use `VK_STRUCTURE_TYPE_RENDER_PASS_TRANSFORM_BEGIN_INFO_QCOM = 1000282001` when driverVersion >=#469"

```
#define VK_QCOM_render_pass_transform                        1
#define VK_QCOM_RENDER_PASS_TRANSFORM_SPEC_VERSION           1
#define VK_QCOM_RENDER_PASS_TRANSFORM_EXTENSION_NAME     "VK_QCOM_render_pass_transform"

#define VK_STRUCTURE_TYPE_RENDER_PASS_TRANSFORM_BEGIN_INFO_QCOM                   1000282000
#define VK_STRUCTURE_TYPE_COMMAND_BUFFER_INHERITANCE_RENDER_PASS_TRANSFORM_INFO_QCOM  1000282001
#define VK_RENDER_PASS_CREATE_TRANSFORM_BIT_QCOM                                  0x00000002
typedef struct VkRenderPassTransformBeginInfoQCOM {
    VkStructureType                  sType;
    void*                            pNext;
    VkSurfaceTransformFlagBitsKHR    transform;
} VkRenderPassTransformBeginInfoQCOM;

typedef struct VkCommandBufferInheritanceRenderPassTransformInfoQCOM {
    VkStructureType                  sType;
    void*                            pNext;
    VkSurfaceTransformFlagBitsKHR    transform;
    VkRect2D                         renderArea;
} VkCommandBufferInheritanceRenderPassTransformInfoQCOM;
```