Nathan Li
Arm
Nan Wei
Tencent Games

WHITE PAPER

Building the Next-Gen Mobile Renderer with Tencent Games

Introduction

Tencent Games is building the next-generation mobile renderer with support from Arm that will power most of its future high-fidelity games. The idea is to take today's mobile graphics to the next level through unleashing the full potential of Vulkan® API by leveraging the advantages of Mali™ processor design. All of this under the guidance of the ***Total Compute*** strategy from Arm.

However, even with one of the world's most powerful game engines, there are still limitations when developing for mobile platforms, particularly with lighting. Delivering a production-ready real-time lighting solution is the challenge that Arm and Tencent Games are attempting to solve with the new mobile renderer for today's hardware and tomorrow's needs.

With endless efforts to optimize the software implementation for Mali, we finally reach the point where rendering hundreds of movable lights on mobile is no longer limited by existing engine solutions. The renderer developed by Tencent Games is now fully capable of delivering this without compromising either performance or visual quality on 2020 premium smartphones. It brings the support to various types of rendering processes, including an incredibly efficient shading pipeline implemented and optimized for devices with limited power and thermal budgets.

Real-time lighting on mobile devices with advanced VFX

The story

The Arm Mali family of GPUs is a broad range of highly effective graphics solutions, with some significant improvements from generation to generation, which includes a continually evolving tile-based graphics architecture. Mali GPUs have been tested by numerous different workloads. These cover high fidelity mobile games, the heaviest loaded benchmarks, all kinds of shading techniques, such as forward and forward plus, and the new challenge brought by the renderer from Tencent.

Thanks to the lightning-fast on-chip tile cache, adaptive pipeline design, and millions of tri-pipes built-in Mali GPUs, complex graphics workloads can happen while consuming minimal battery life of smartphones. We would like to show our great respect to the engineers from Tencent Games, who have been working on this project day in and day out. In each step of the pipeline, there is plenty of optimization, including geometry processing with optimal methods, maximizing the efficiency of each pass, saving bandwidth with Vulkan subpasses, and, finally, a fine-tuned lighting pass to cull light sources. Arm has been providing design advice, tools, and libraries for Tencent Games to accelerate the development of the renderer. Following this work, we are now able to talk more about the journey.

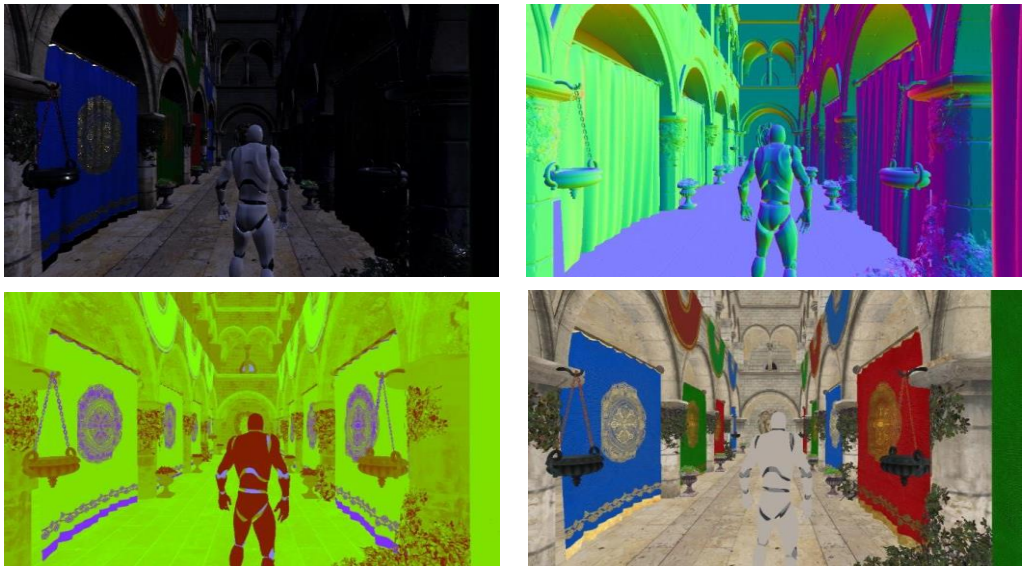
Highlights of the Pipeline

Overview of the deferred shading pipeline

When lights fall on objects, shadows form. In the digital world, lighting is all about computing for the gain and loss of energy transmitted between surfaces throughout millions of iterations per second. In the end, a well-painted picture is achievable by compositing the contribution of lights and shadows of every light source into each pixel. Developers often use textures to represent the brightness and darkness information varying from frame to frame, called shadow maps.

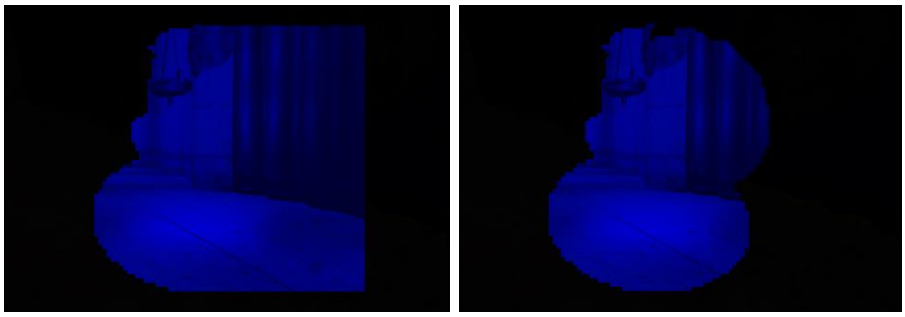
It is worth noting that by dividing a complicated problem into smaller pieces can often save a typical day of a programmer. This is especially true for Tencent engine developers, who have been dealing with the endless challenge of performance optimization since day one. Coming to the topic, we see many kinds of lights in video gaming, as well as in the real world. The essential types of light supported by the renderer include but are not limited to, point lights, spotlights, and directional lights, with various types of **Cascaded Shadowmap** and **Shadowmap atlas**, respectively:

Shadow map atlas for various types of lights



Main stages of the deferred shading pipeline

Better **overdraw reduction** (by the fewer primitives, fewer pixels, or fewer color attachments) is the magic that the renderer plays with lighting, along with making use of the masks at various stages for different purposes. For example, a stencil mask generated in previous GBuffer pass will also be used later for culling pixels for directional lights in Decal pass. It can also be the input further to the lighting pass or optional later stages to accelerate full-screen directional lighting at the far end, where translucent and post-process stages are placed to complete a classical deferred pipeline. There are many more exciting topics left for us to explore to achieve advancements in visual quality and performance.



Lights culling with fast intersection tests

One thing that is particularly remarkable is how the new renderer takes full advantage of the architecture used by Mali GPUs in its lighting pass. This is achieved by keeping relevant geometric and shading information in the on-chip tiled cache, with no redundant load or store operation. It also improves the precision of light culling by optimizing **intersection testing algorithms**, which significantly boosts shading performance. Everything is implemented by fragment shaders, providing the best chance to run smoothly on every device with no compatibility issues.

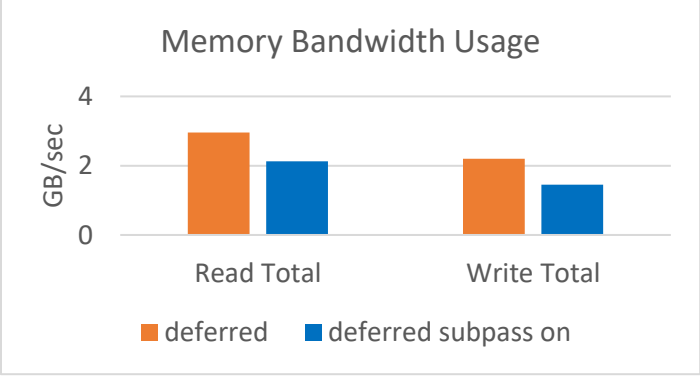
Rendering result of lighting pass

Vulkan: Barrier, Layout and Subpass

Mali GPUs expose two hardware processing slots. Each slot implements a subset of the rendering pipeline stages and runs in parallel with one another. Therefore, synchronizations within a single hardware processing slot and across the two hardware processing slots are both critical to any performance-sensitive application. We are confident to say that the renderer from Tencent Games has successfully passed the most critical reviews done by our GPU experts in terms of the efficiency of pipeline design against the criteria about synchronization listed in Mali developer guides.

In late 2019, Arm started looking into the prototype of the renderer, with the focus on reviewing all the **Image Layouts** used in the pipeline. We started from the point that *VK_IMAGE_LAYOUT_GENERAL* was seen everywhere and ended up with a bunch of optimal layouts (i.e., *VK_IMAGE_LAYOUT_DEPTH_STENCIL_READ_ONLY_OPTIMAL*) when applicable. By carefully following the principles listed in our Vulkan Best Practices, we got the most out of **Vulkan Subpass Merge** in each of these cases. It indeed took some time to sort out the clean and robust relationships between all the input and output attachments linked to those "naughty" subpasses.

We have been going back and forth here to check the settings of the pipeline to make sure that the data formats of color attachments can be merged – with no redundant depth and stencil changes – for all the subpasses that potentially share the data. The Mali GPU driver can combine them intelligently for batch processing at runtime to avoid unnecessary write-out or read-back and then reduce power usage quite a bit. After such a long process, we are very excited to see that all the considerable efforts invested in our GPU hardware and driver to enable the features are worth the cost, especially in an intensive use case, which means at least a 30% **bandwidth saving**.

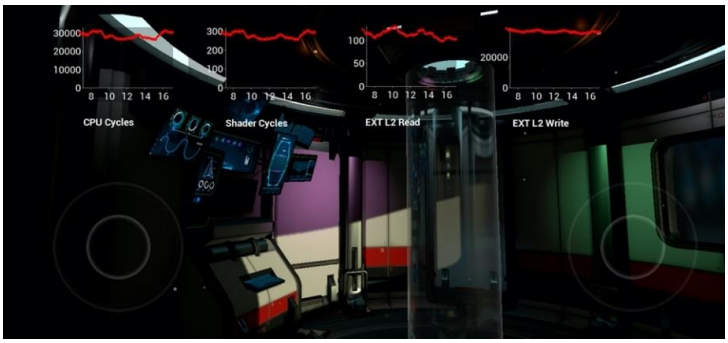


Bandwidth saving on a Mali G7x platform

In the meantime, Arm compiler engineers studied many shaders used in the renderer. They evaluated how good the shaders fit into the design of Mali hardware registers and shader cores, as well as delivering in-depth performance analysis reports for engine developers to do things right. In the end, we provided some advice on how to make the right choice when people are programming the shaders to implement specific rendering algorithms.

Tech demos: 64-bit and Vulkan ready

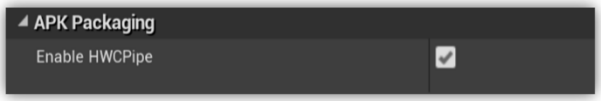
At the very beginning of the project, Arm and Tencent Games both agreed to make the 64-bit and Vulkan build options priorities, as our partners are moving towards a pure 64-bit content ecosystem rapidly. Tencent Games arranged most of the QA work around this setting until the final stage targeting legacy devices. Looking back at the critical moment when the highly customized renderer started running for the first time, we needed challenging content to validate its essential functionalities. Developers could not wait to compare it with what can be achieved on a desktop PC with stock solutions in the market. Therefore, Arm decided to remake a tech demo of **CircuitVR** for Tencent Games.



Arm Toolchain

One of the biggest challenges for software developers who fight for better performance is always how to make use of tools to achieve more with less effort. Fortunately, we provide various options to choose from, for instance, the HWCPipe library, which is free for developers to download and use. Tools and libraries from Arm have played a significant role throughout the development of the renderer to keep things on the right track.

With support from Tencent Games, developers can now access Mali GPU performance counters in real-time by enabling this option in "**APK Packaging**" session on the project settings panel, along with simple clicks in the visual scripting language editor as follows (subject to engine support):



The Future

Now it is time for Tencent Games to take things one step further through using the technologies that we talked about for commercial game titles beyond 2020. Arm is committed to ensuring that mobile devices powered by our technologies are ready to provide the compute performance for future digital immersion that gamers have been dreaming of experiencing for decades, as part of the *Arm Total Compute Solution*.

Additional Resources

Arm recommends visiting some of the following resources for more specific information about our technologies:

Arm Total Compute Strategy

Arm Technologies Overview

64-bit Android Development