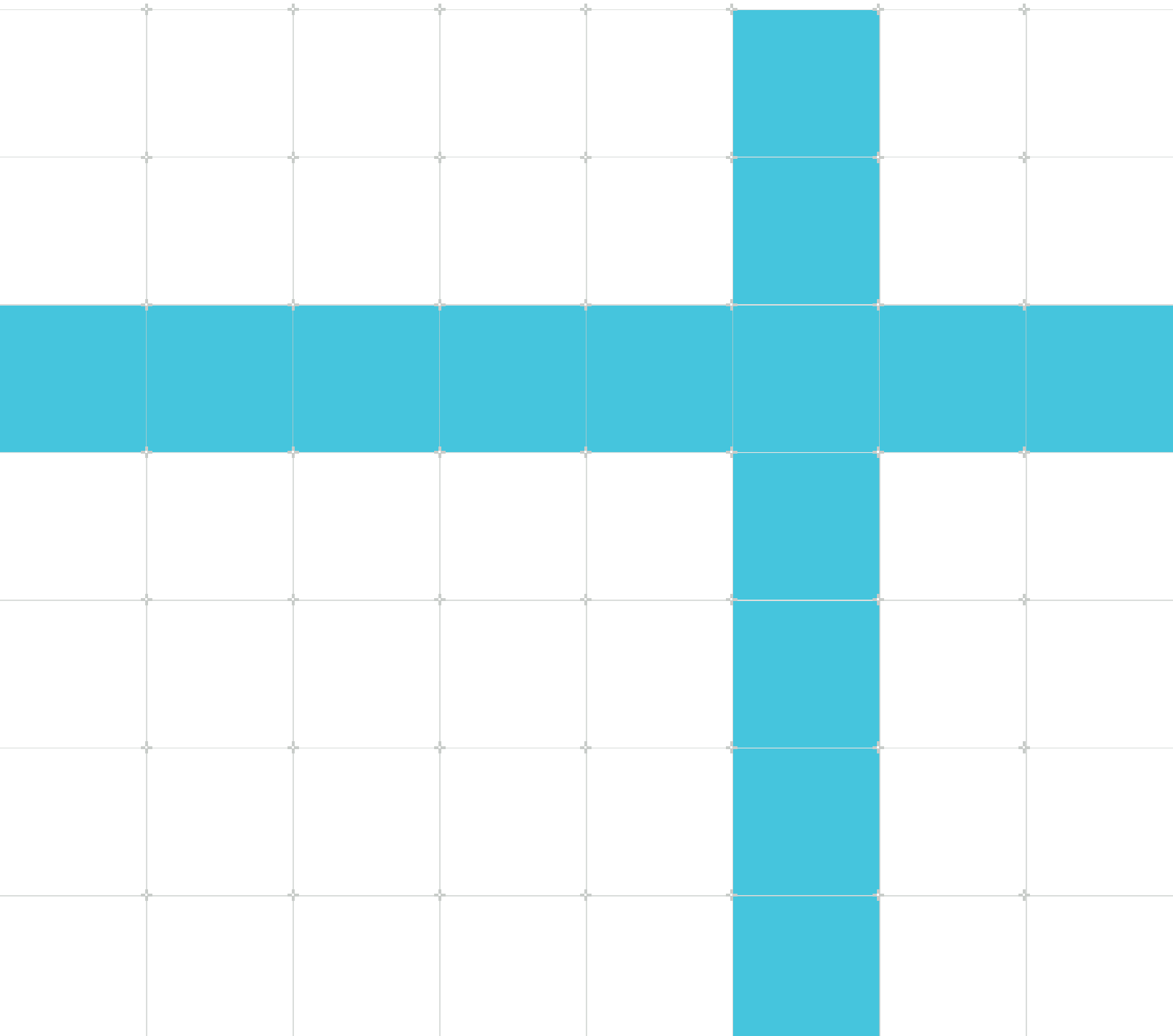
arm

Principles of High Performance

Version 1.0

guide

Issue 02
102544_0100_02_en



Principles of High Performance
guide

Release information

Document history

Issue	Date	Confidentiality	Change
0100-02	7 October 2020	Non-Confidential	First release

Proprietary Notice

and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm® welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Overview

Optimizing the graphics for your target GPU should be a fundamental goal during your application's development. And as graphics optimization is more of an art than a science, it is important to put time into learning how to balance the goal of creating vibrant and exciting on-screen visuals, with the optimization tweaks that happen 'under the hood'.

This guide is aimed at developers who are new to developing applications.

In this guide we will walk you through: The key differences between effectiveness and efficiency, How to be effective with your energy - in terms of performance, How to be efficient with your energy - in terms of performance, and why frequently analyzing the data you pass to the GPU is essential.

2. It's all about energy

The most significant performance limitation of smartphones are their form factor. Passively cooling a chip inside a sealed case is a challenge, and the heat dissipation rate determines how much power can be sustainably drawn during gameplay.

For most devices, the System on Chip (SoC) power budget is between 2.5 and 3.5 Watts. This only leaves 1 to 1.5 Watts for the GPU. The challenge for you, when trying to get the best performance and rendering quality out of these platforms, is how to be as efficient as possible with that power budget.

3. Principle 1: Be effective

Considering the energy and thermal constraints of a smartphone, what does effectiveness mean for high performance rendering? The answer is spending energy on CPU cycles, GPU cycles, and memory accesses, resulting in a valuable improvement in the screen's output.

Any cycle or byte spent on something that is not visible, or which simply doesn't justify its cost, is a waste of energy and results in a loss in quality. The first step for GPU optimization is not to make the current rendering faster by fine tuning. It is to ensure that the overall rendering pipeline, and choice of algorithm, is suitable for both the device's performance capability and power budget.

In addition, remove workloads if they are not contributing a useful output to the current frame.

Some examples of such workloads may include:

- Reviewing your overall algorithm choices,
- Reviewing the output resolution, and color format, of each render pass,
- Ensuring the CPU is culling draw calls that are off-screen, or which are known to be occluded, before they are sent through the graphics API.

4. Principle 2: Be efficient

What does efficiency mean for high performance rendering? By this point in the development cycle, you should have already applied effective optimization to ensure that only useful work is sent to the GPU.

It is now time to focus on optimizing the remaining workload that contributes to the final render.

These optimizations can include activities such as:

- Ensuring API usage is efficient, with minimal state changes.
- Checking models are well structured, with good locality, and minimal precision in data buffers.
- Ensuring textures are using appropriate data formats, texture compression, and filtering modes.
- Reviewing shader programs and their execution cost.

5. Principle 3: Data matters

Developers are used to writing and reviewing code and spend a lot of their optimization time looking at API calls and shader code. This can easily be done without really looking at the data they are passing to the GPU. This can be a lost opportunity because GPUs are data-plane processors.

Graphics rendering performance can be strongly influenced by data efficiency problems. Accessing external DDR memory is very energy intensive and so poorly sized, or inefficiently packed data resources, can rapidly consume valuable energy.

A useful rule of thumb is that 1GB/s of memory access uses approximately 100mW of power. If you spend 600mW of your 2.5W power budget on memory access, an application running at 60 Frames-Per-Second (FPS) only gets 100MB per frame to play with.

The data optimization stage can include the following review activities:

- Total bandwidth needs of the scene render pass graph.
- Resolution of textures and framebuffers.
- The use of texture compression and mipmapping.
- Models for good data locality and minimal data precision.
- Shader programs and their execution cost.

6. Principle 4: It's an art, not a science

It is important to remember that graphics optimization is an art, not a science. In most cases, there is no single right answer on how to fully optimize graphics as the route you take will depend on what you are trying to achieve.

In general, a common goal to aim for is to render something on-screen that is both visually appealing, and does not hinder your application's performance.

So don't be afraid to experiment with the algorithms and use faster approximations if it helps streamline performance. It's unlikely that anyone will notice if an optimized version of something on-screen is not bit-exact.

7. Conclusions

High quality real-time rendering in a mobile form factor is a reality on modern smartphones today, but requires the developer to write effective and efficient applications to make the best use of the available system power budget.

There are many details to get right, ranging from high-level application algorithm choices, all the way down to the fine detail of object mesh data encoding. Other articles on this site will explore key topics related to efficient use of Mali GPUs in more detail.

References: Drucker, Peter F. 1967. The effective executive.