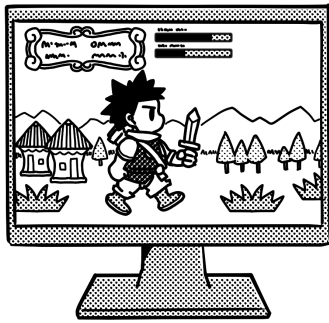


Arm Mobile Studio: a Developer's Best Friend

Yeah, finally ported my
game to mobile.



But the
performance
is terrible...

And the
phone
becomes
super hot...

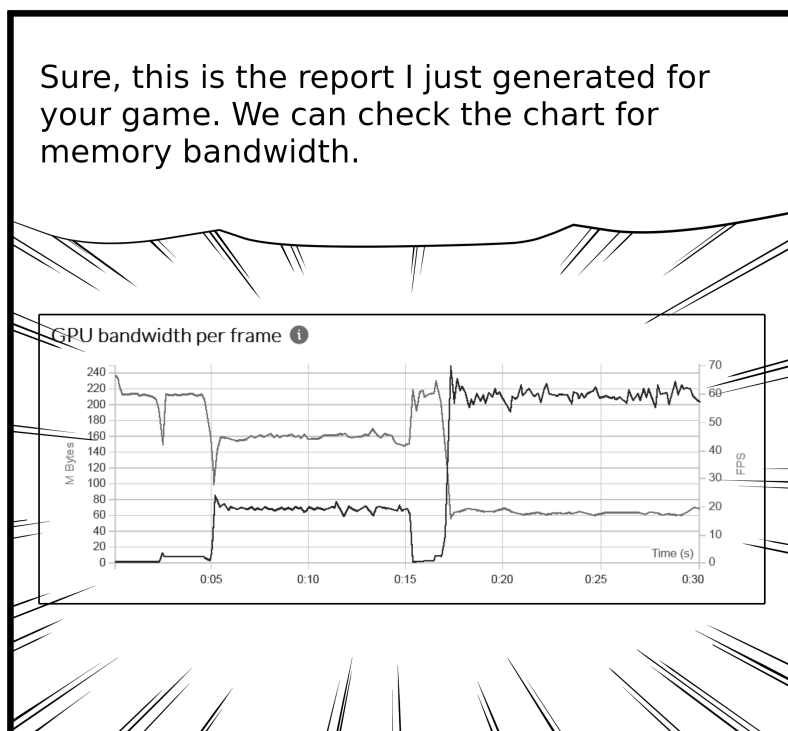
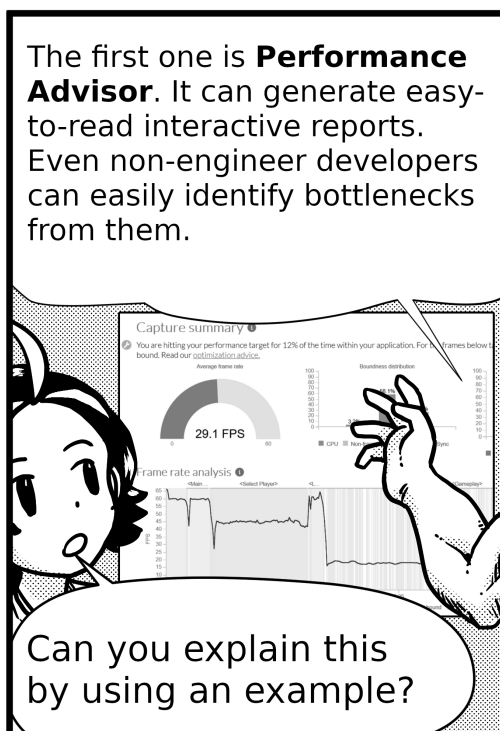
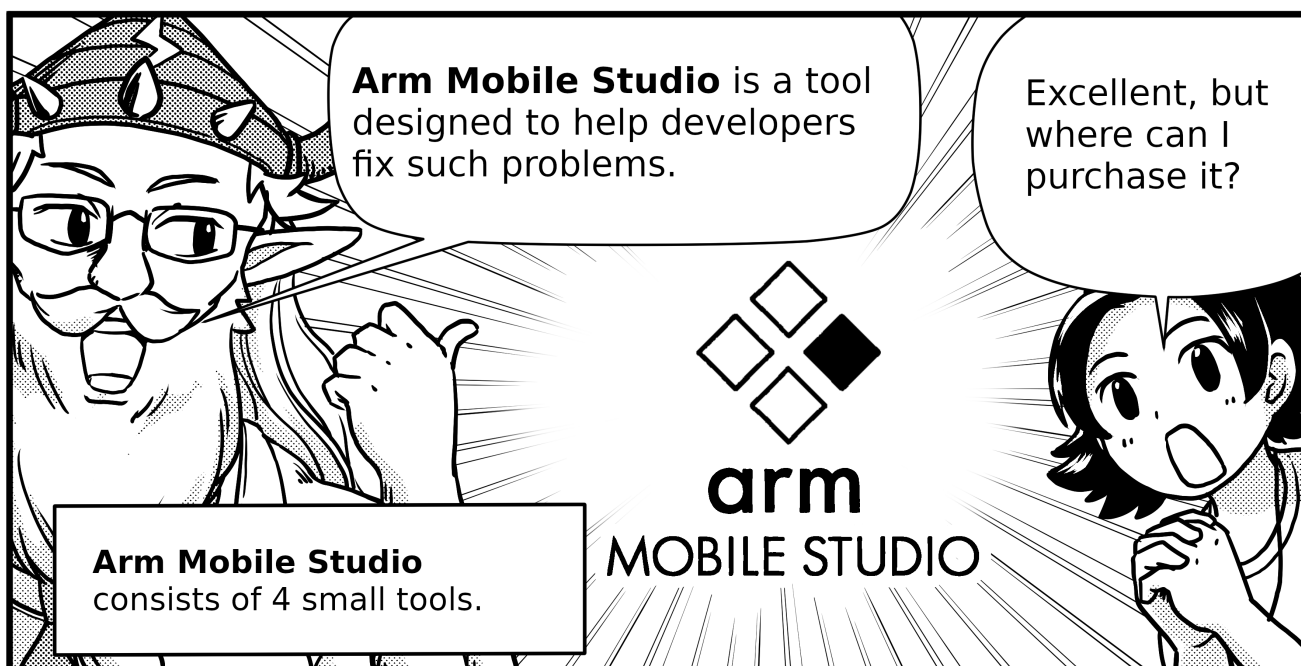
Feels like I
can fry an
egg on the
phone...

No worries. Do you know what
Arm Mobile Studio can help
you fix this?

Dr. Arm !
What is Arm
Mobile Studio?

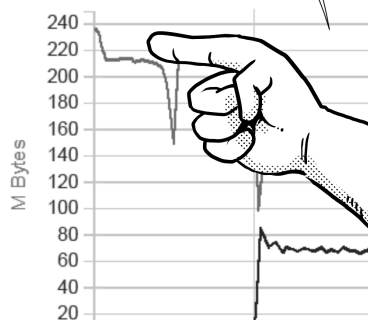
arm

developer.arm.com/graphics - 1 -



We can see from the chart, the bandwidth is as high as **7GB/s**.

GPU bandwidth per frame

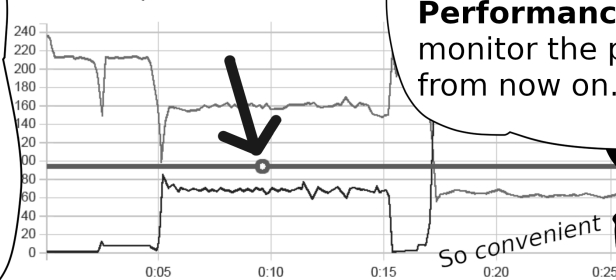


The memory bandwidth on mobile is very limited, so higher bandwidth causes higher power consumption and heat.

Oops, I've never been aware of that before....

Having the bandwidth around **2 to 3 GB/s** is recommended on mobile. **Performance Advisor** has a feature which lets you mark the budget on the chart, so anything over that budget can be found easily.

GPU bandwidth per frame ①



Cool! I will let **Performance Advisor** monitor the performance from now on.

The next tool is **Streamline**. This is for engineer developers because it allows them to know exactly what's happening inside the GPU.

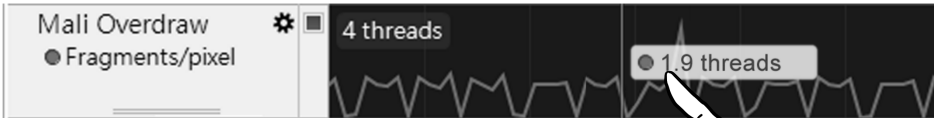


Yes! The GPU is like a black box. It's really hard to do optimizations without knowing what's happening inside.

This tool uses charts to show the counters inside the GPU and makes the bottleneck investigation much easier.

See, this is what I just captured using **Streamline**. You can see here that the number of **Overdraw** is around 1.9.

This means that one pixel is drawn 1.9 times on average. So this will cause some performance waste.



Why will one pixel be drawn more than once?

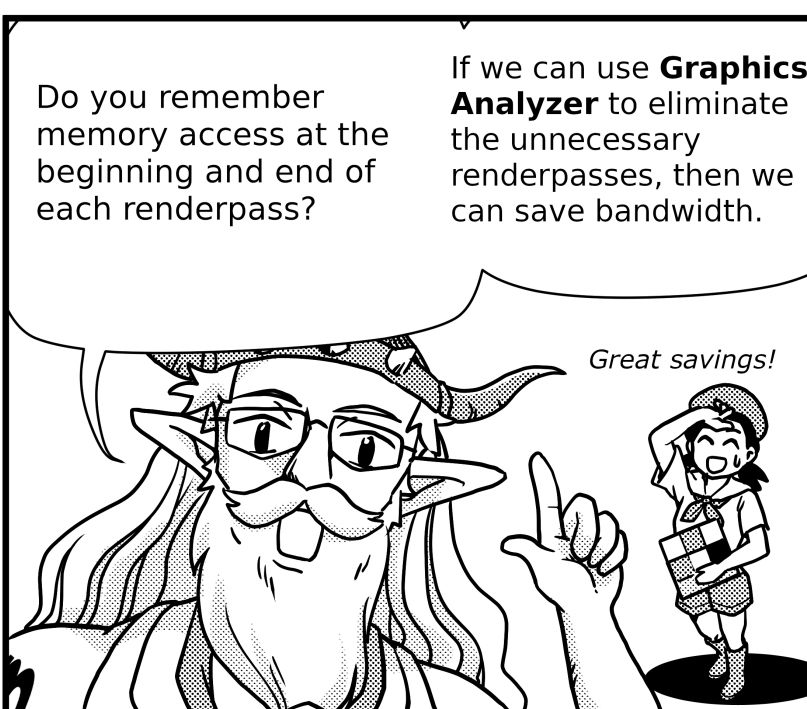
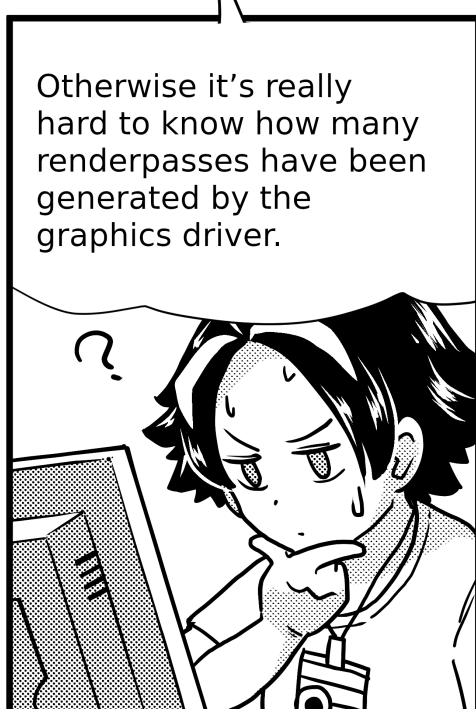
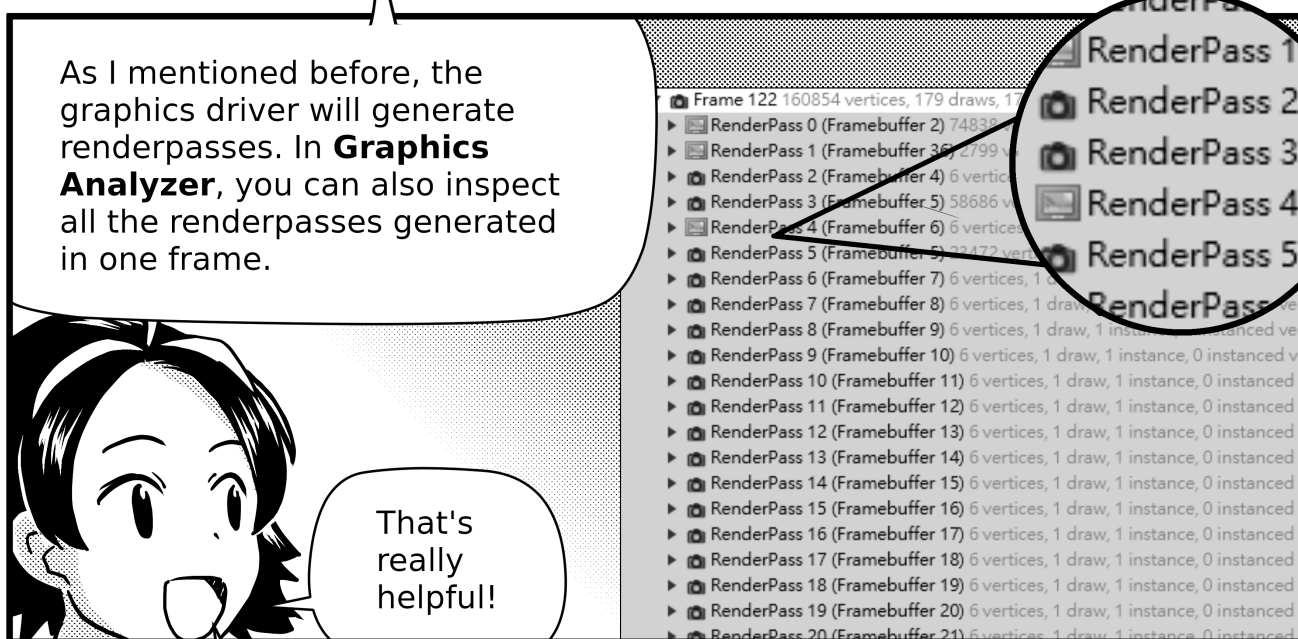
It happens when there are overlapping objects in the scene! For example, rendering the skybox first then the scene objects can easily cause overdraw.

I see.

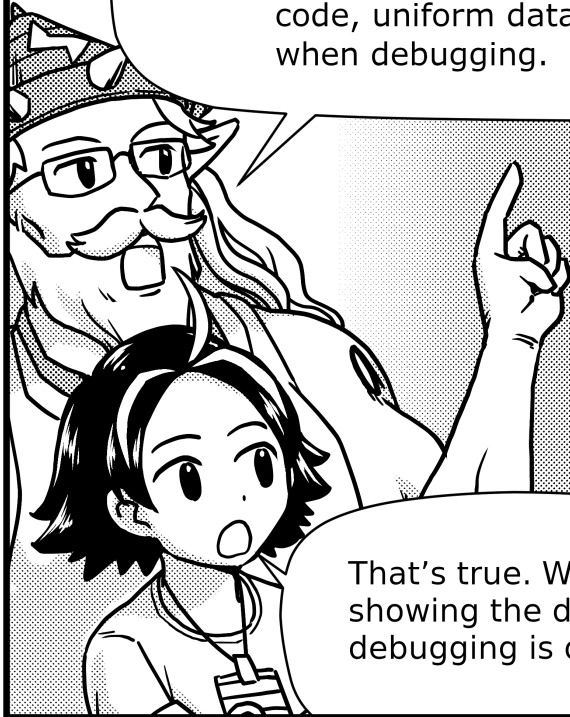
So it's better to render objects front-to-back and not back-to-front?

Occluded pixels are not drawn

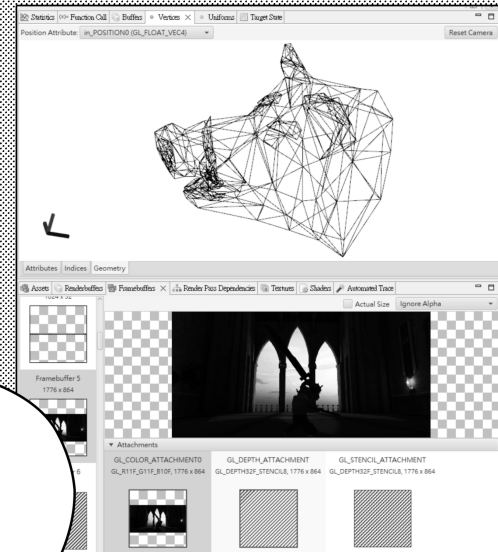
Correct, doing this can reduce the overdraw and get a performance boost!



Besides showing the renderpasses, **Graphics Analyzer** can also show the geometry data, shader code, uniform data, etc. It's a very convenient tool when debugging.



That's true. Without showing the data, debugging is quite tricky.



The last tool is **Mali Offline Compiler**. This is for shader performance analysis.

```
Mali Offline Compiler v7.3.0 (Build 102baf)
Copyright 2007-2020 Arm Limited, all rights reserved

configuration
=====
Hardware: Mali-G72 r0p3
Architecture: Bifrost
Driver: r25p0-00rel0
Shader type: OpenGL ES Fragment

Main shader
=====
Work registers: 46
Uniform registers: 28
Stack spilling: false
16-bit arithmetic: 0%

Total instruction cycles:  5.00  0.00  1.50  1.00  A
Shortest path cycles:     5.00  0.00  1.50  1.00  A
Longest path cycles:     5.00  0.00  1.50  1.00  A

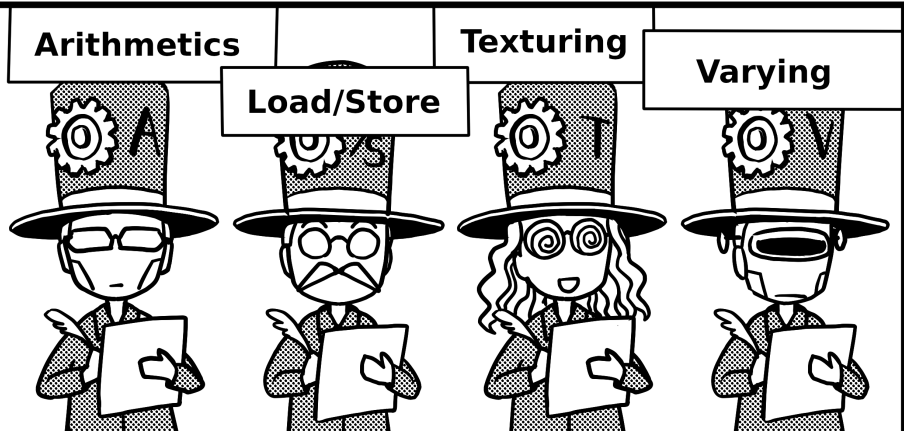
A = Arithmetic, LS = Load/Store, V = Varying, T = Texture

Shader properties
=====
Has uniform/computation: true
Has side effects: false
Modifies coverage: false
Uses late ZS: false
Uses late ZS/updates: false
```

Does this mean I can use it to help optimize complex shaders?



There are 4 units inside the execution engine of a shader core. This tool can help you understand the workload of each unit when running the shader.



What's the meaning of these numbers?

These are the workload cycles of each unit when running your shader.

```

: 46
ters: 28
ng: false
hmetic: 0%

Instruction cycles:
est path cycle:
gest path cycle:

```

	A	LS	V	T	Bound
Instruction cycles:	5.00	0.00	1.50	1.00	
est path cycle:	5.00	0.00	1.50	1.00	
gest path cycle:	5.00	0.00	1.50	1.00	

Arithmetic: LS = Load/Store, V = Varying, T = Texture

Because all the units are independent, the one with the heaviest workload will decide the performance of your shader.

Can't get off work because of you.

Hurry up! We all are waiting for you...

Not my fault! My workload is much heavier than yours!

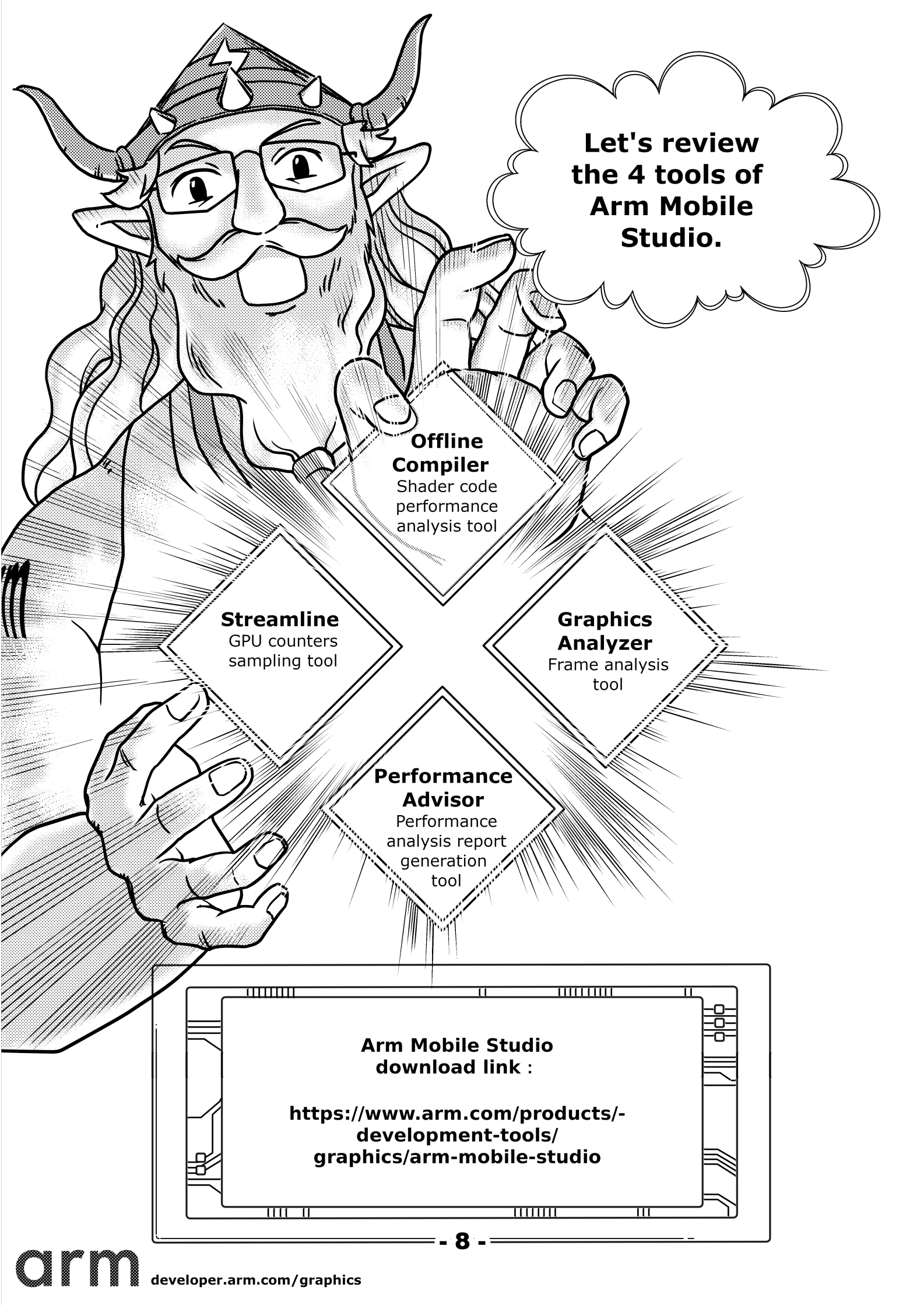
Because the arithmetic unit has the heaviest workload, it means the arithmetics are the bottleneck. So we must optimize it first.

Relieved! ^_^

With this tool, you will never optimize the wrong place.

Excellent! I'm going to optimize my game using **Arm Mobile Studio** now!

Go! Go! Go!



**Let's review
the 4 tools of
Arm Mobile
Studio.**

**Offline
Compiler**

Shader code
performance
analysis tool

Streamline

GPU counters
sampling tool

**Graphics
Analyzer**

Frame analysis
tool

**Performance
Advisor**

Performance
analysis report
generation
tool

**Arm Mobile Studio
download link :**

**[https://www.arm.com/products/-
development-tools/
graphics/arm-mobile-studio](https://www.arm.com/products/-development-tools/graphics/arm-mobile-studio)**