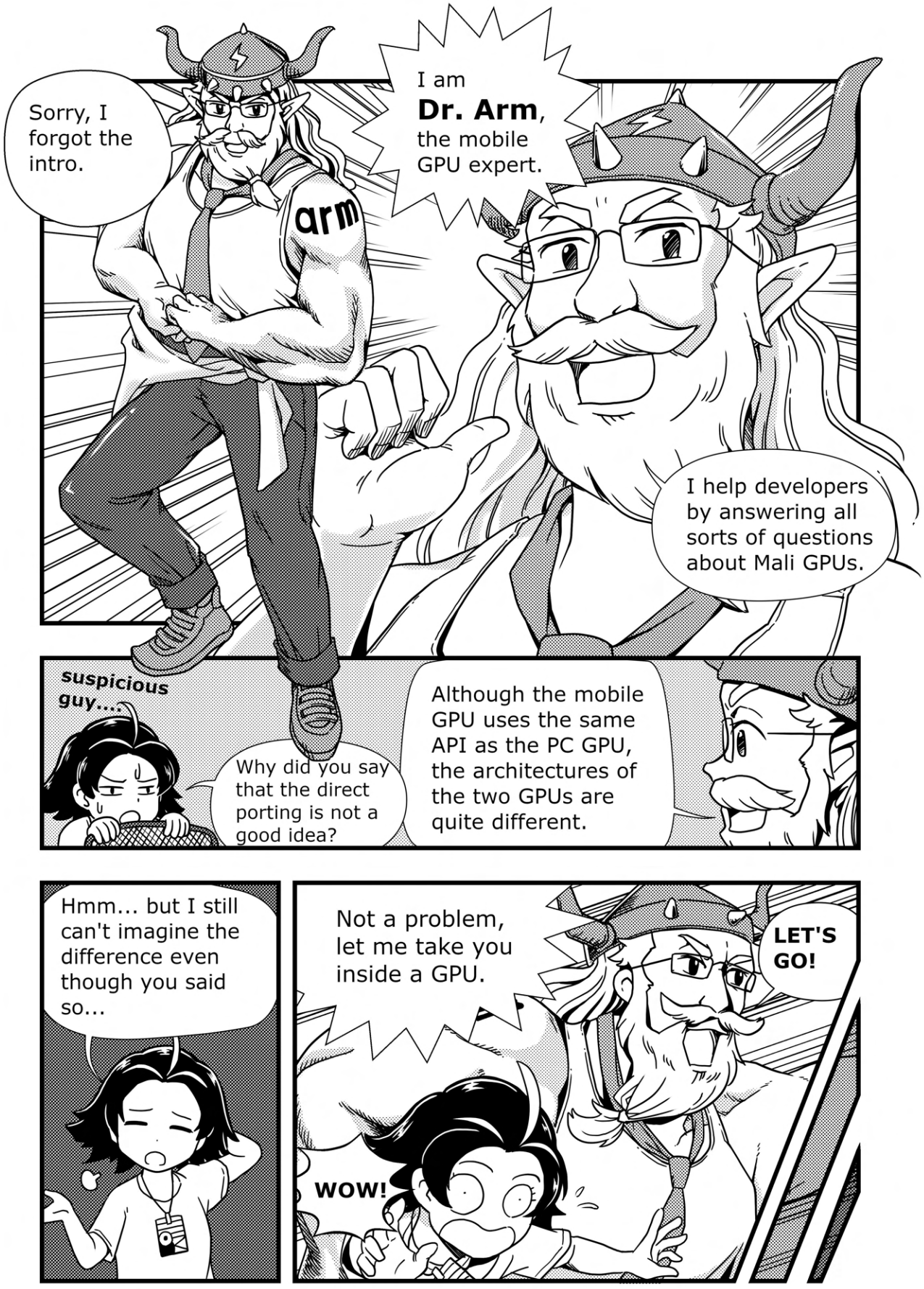
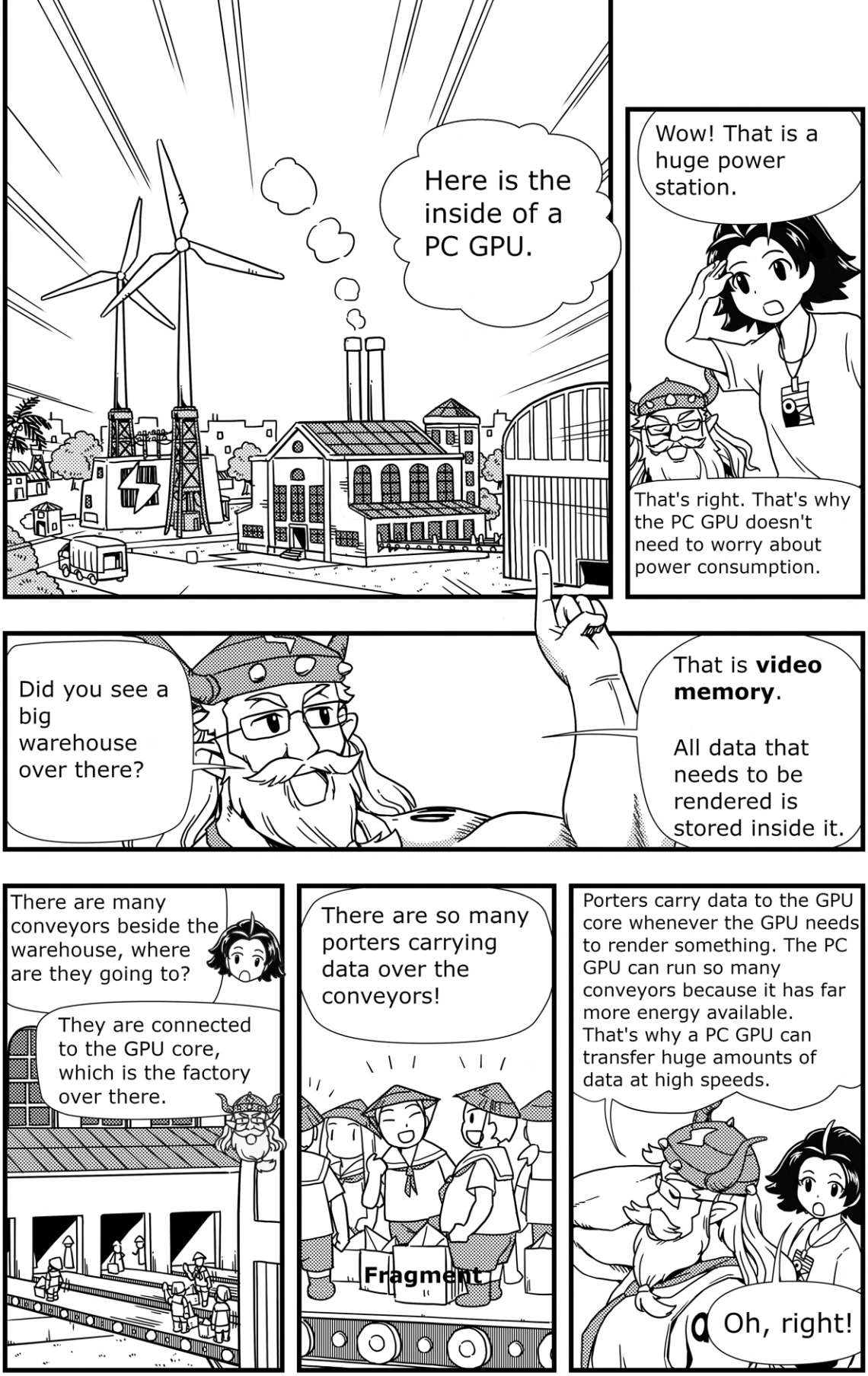
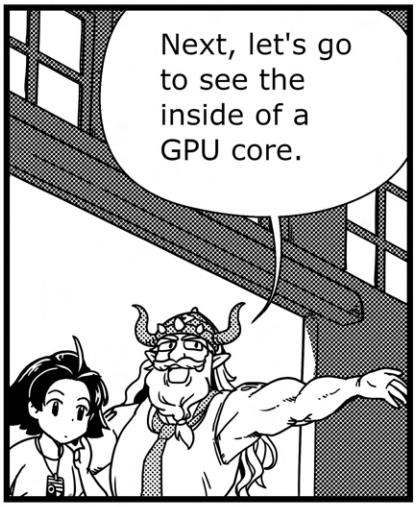
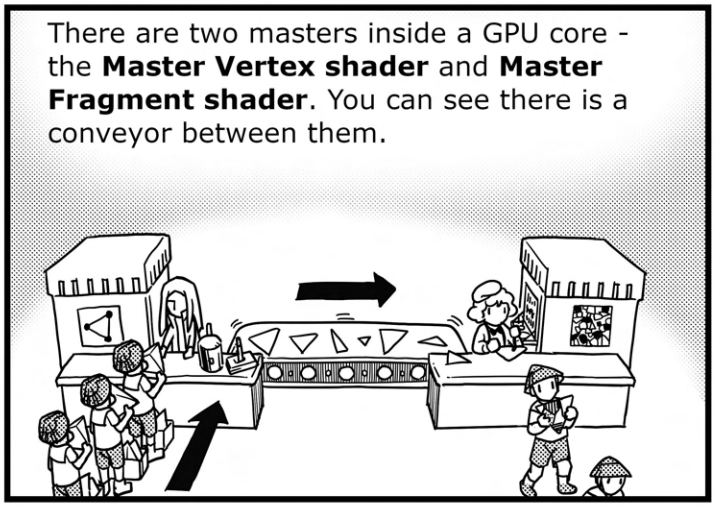
How does a Mobile GPU Work?

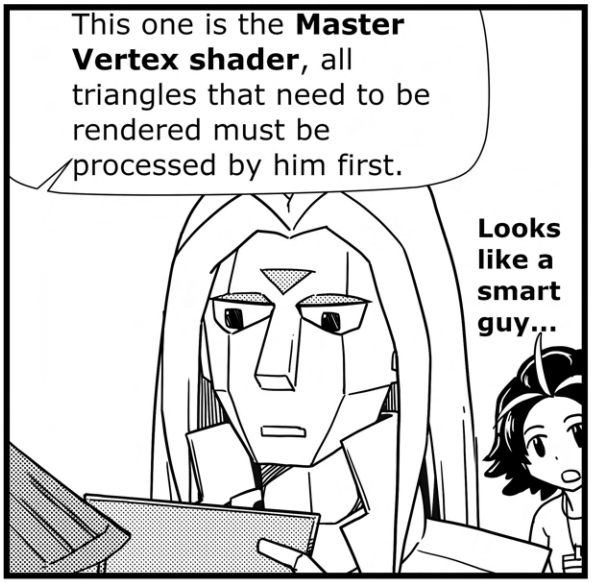
Comic: Ikaridon Yu

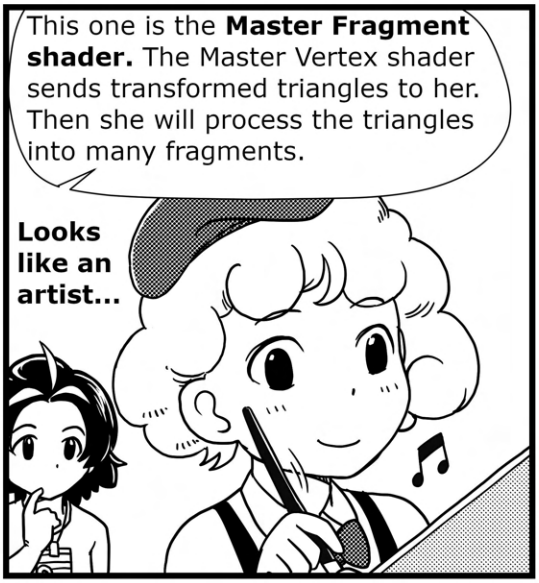

Sorry, I forgot the intro.
I am **Dr. Arm**, the mobile GPU expert.
arm
I help developers by answering all sorts of questions about Mali GPUs.
suspicious guy...
Why did you say that the direct porting is not a good idea?
Although the mobile GPU uses the same API as the PC GPU, the architectures of the two GPUs are quite different.
Hmm... but I still can't imagine the difference even though you said so...
Not a problem, let me take you inside a GPU.
LET'S GO!
WOW!

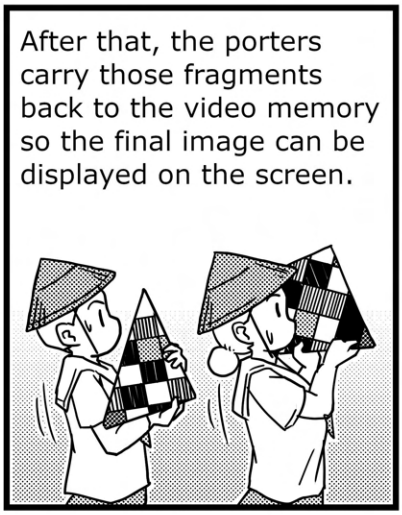

Here is the inside of a PC GPU.
Wow! That is a huge power station.
That's right. That's why the PC GPU doesn't need to worry about power consumption.
Did you see a big warehouse over there?
That is **video memory**.
All data that needs to be rendered is stored inside it.
There are many conveyors beside the warehouse, where are they going to?
They are connected to the GPU core, which is the factory over there.
There are so many porters carrying data over the conveyors!
Fragment
Porters carry data to the GPU core whenever the GPU needs to render something. The PC GPU can run so many conveyors because it has far more energy available. That's why a PC GPU can transfer huge amounts of data at high speeds.
Oh, right!

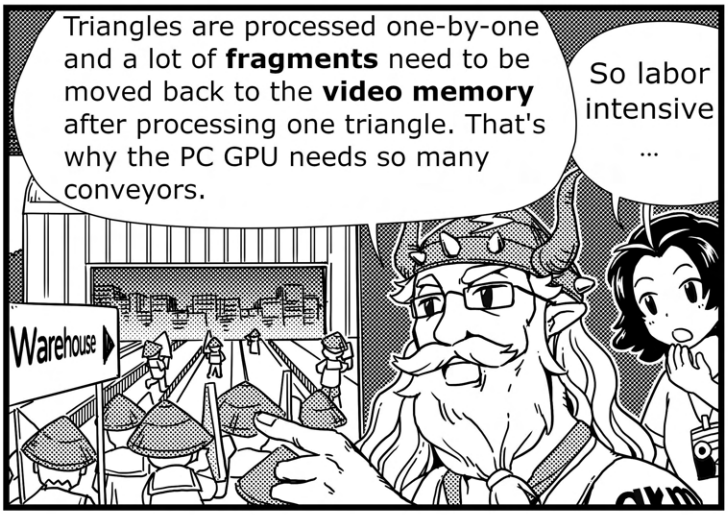

Next, let's go to see the inside of a GPU core.

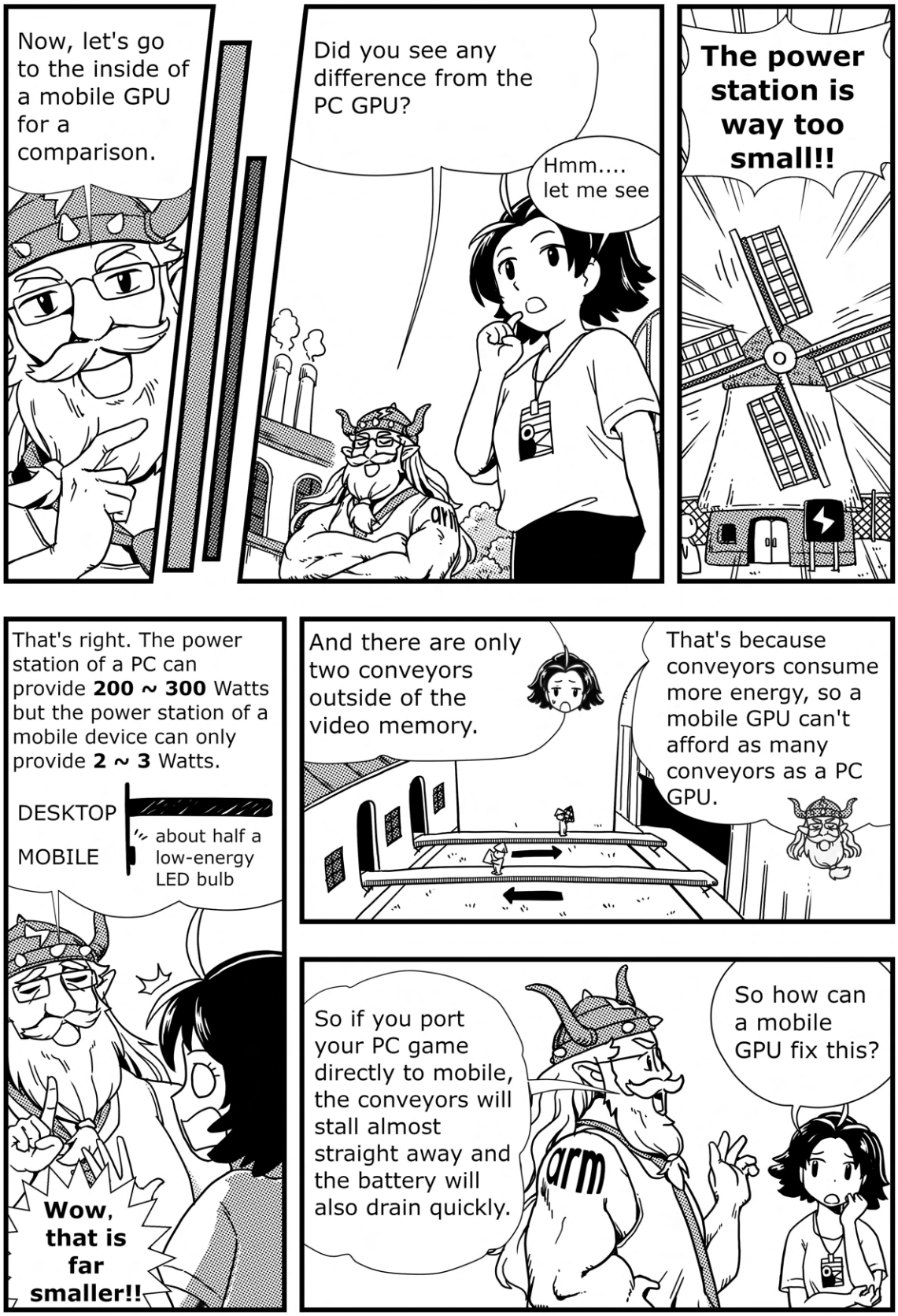

There are two masters inside a GPU core - the **Master Vertex shader** and **Master Fragment shader**. You can see there is a conveyor between them.


This one is the **Master Vertex shader**, all triangles that need to be rendered must be processed by him first.
Looks like a smart guy...


This one is the **Master Fragment shader.** The Master Vertex shader sends transformed triangles to her. Then she will process the triangles into many fragments.
Looks like an artist...


After that, the porters carry those fragments back to the video memory so the final image can be displayed on the screen.

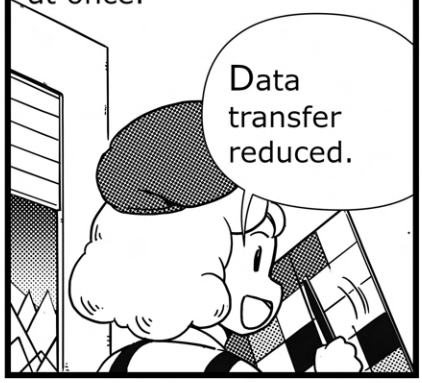

Triangles are processed one-by-one and a lot of **fragments** need to be moved back to the **video memory** after processing one triangle. That's why the PC GPU needs so many conveyors.
So labor intensive ...
Warehouse

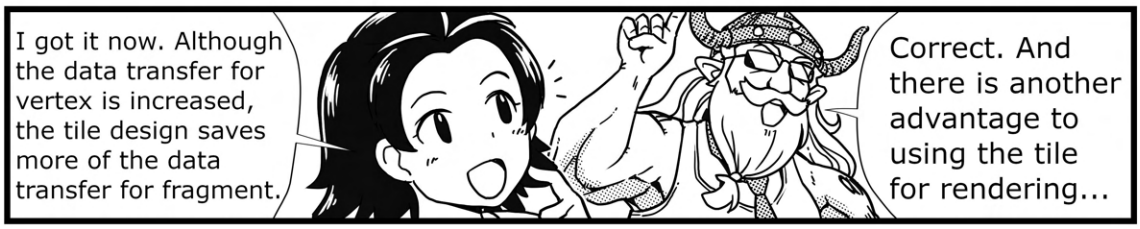

Now, let's go to the inside of a mobile GPU for a comparison.
Did you see any difference from the PC GPU?
Hmm.... let me see
The power station is way too small!!
That's right. The power station of a PC can provide **200 ~ 300** Watts but the power station of a mobile device can only provide **2 ~ 3** Watts.
DESKTOP
MOBILE
about half a low-energy LED bulb
Wow, that is far smaller!!
And there are only two conveyors outside of the video memory.
That's because conveyors consume more energy, so a mobile GPU can't afford as many conveyors as a PC GPU.
So if you port your PC game directly to mobile, the conveyors will stall almost straight away and the battery will also drain quickly.
So how can a mobile GPU fix this?


Let's take a close look at the mobile GPU core. It has the **vertex shader** to process triangles, but...
THUD
The transformed triangles will not pass directly to the **fragment shader**. Instead, the triangles will be transferred to the **video memory**.
What?! So what should the **fragment shader** do then?
Zzz
The **Fragment shader** will get the transformed triangles from the video memory then.
But would it increase the size of the data transfer?
Exactly! Did you see a small warehouse besides the **fragment shader**?
Yes, I didn't see such a thing at the PC GPU core.
It's **tile memory** a special design for mobile GPUs.
Tile
On a mobile GPU, the screen is split into many **tiles**. Each **tile** contains **16x16 ~ 64x64** fragments.
16
16

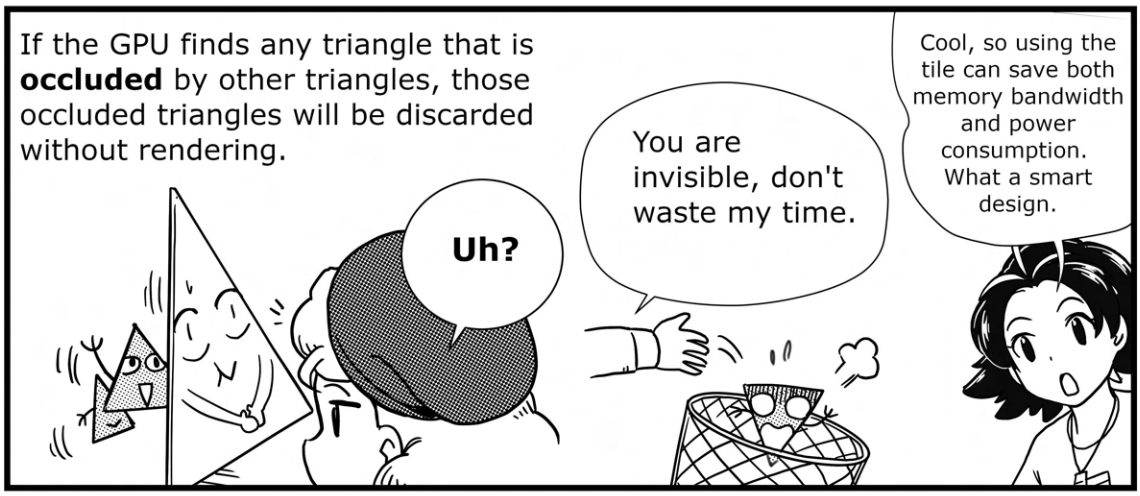
The **fragment shader** processes one **tile** at a time and then processes all triangles in the tile at once. So the frame buffer data just needs to be transferred to the **tile memory** at the beginning of the tile processing.

Then the **fragment shader** can directly access the data in the **tile memory** and process all the triangles at once.





If the GPU finds any triangle that is **occluded** by other triangles, those occluded triangles will be discarded without rendering.



The whole tile will be transferred back to the video memory when all triangles in this tile are processed.

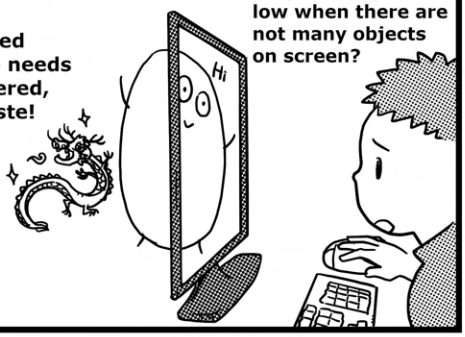
Because the number of **tiles** on a screen is fixed, the size of the data transfer is also fixed.

My size is fixed

But on a PC GPU, the data transfer size is not fixed. It varies depending on the number of triangles that are rendered.

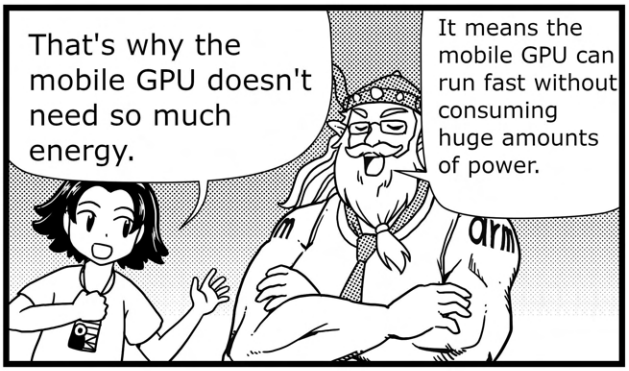
Why is the fps so low when there are not many objects on screen?

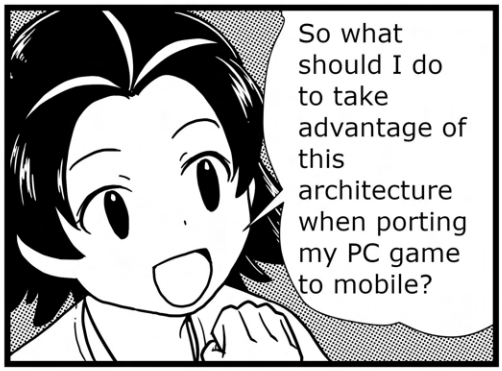
The occluded object also needs to be rendered, what a waste!



That's why the mobile GPU doesn't need so much energy.

It means the mobile GPU can run fast without consuming huge amounts of power.

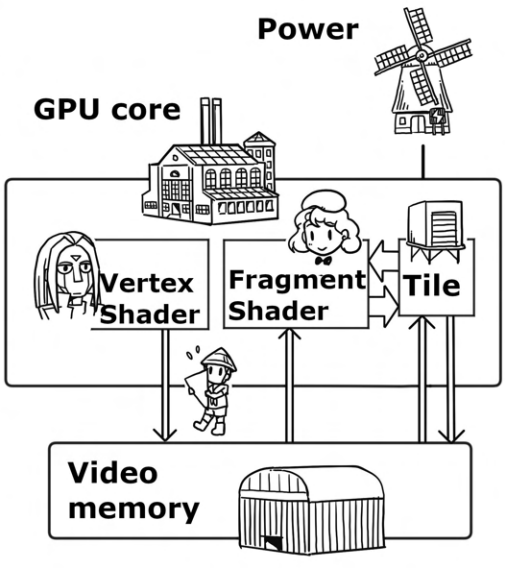




So what should I do to take advantage of this architecture when porting my PC game to mobile?

Good question. But we have learned enough today.

Let's review what we've learned so far and I will show you how to do that in the next episode.



Go to the Arm developer site for more detailed information :
https://developer.arm.com/solutions/-graphics-and-gaming/developer-guides/-learn-the-basics/tile-based-rendering