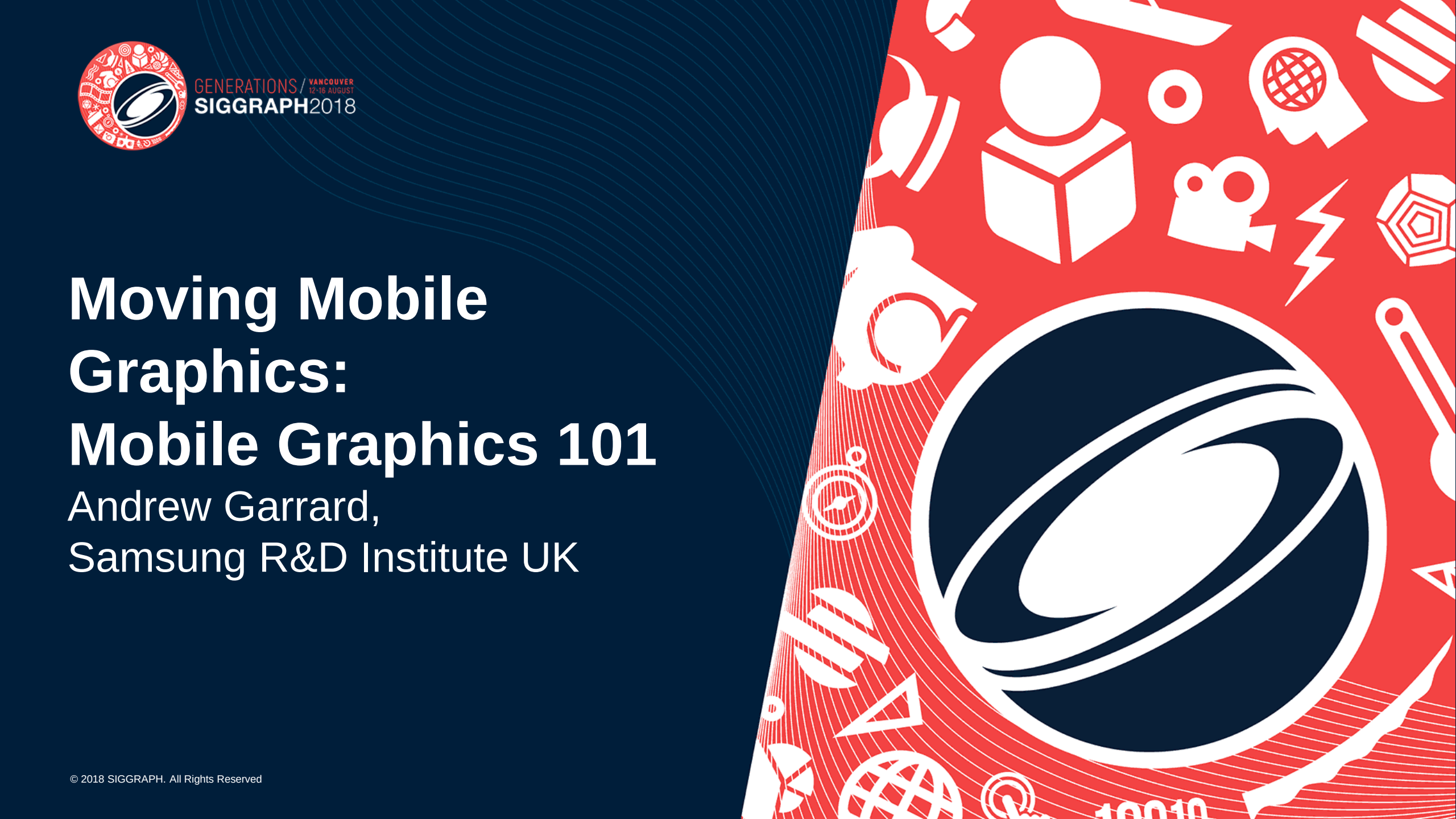


Moving Mobile Graphics: Mobile Graphics 101

Andrew Garrard,
Samsung R&D Institute UK

GENERATIONS / VANCOUVER 12-16 AUGUST
SIGGRAPH2018

What makes a mobile GPU*?

The more things change, the more they stay the same

* Graphics Processing Unit

“Mobile” vs “Desktop”

Mobile

Desktop

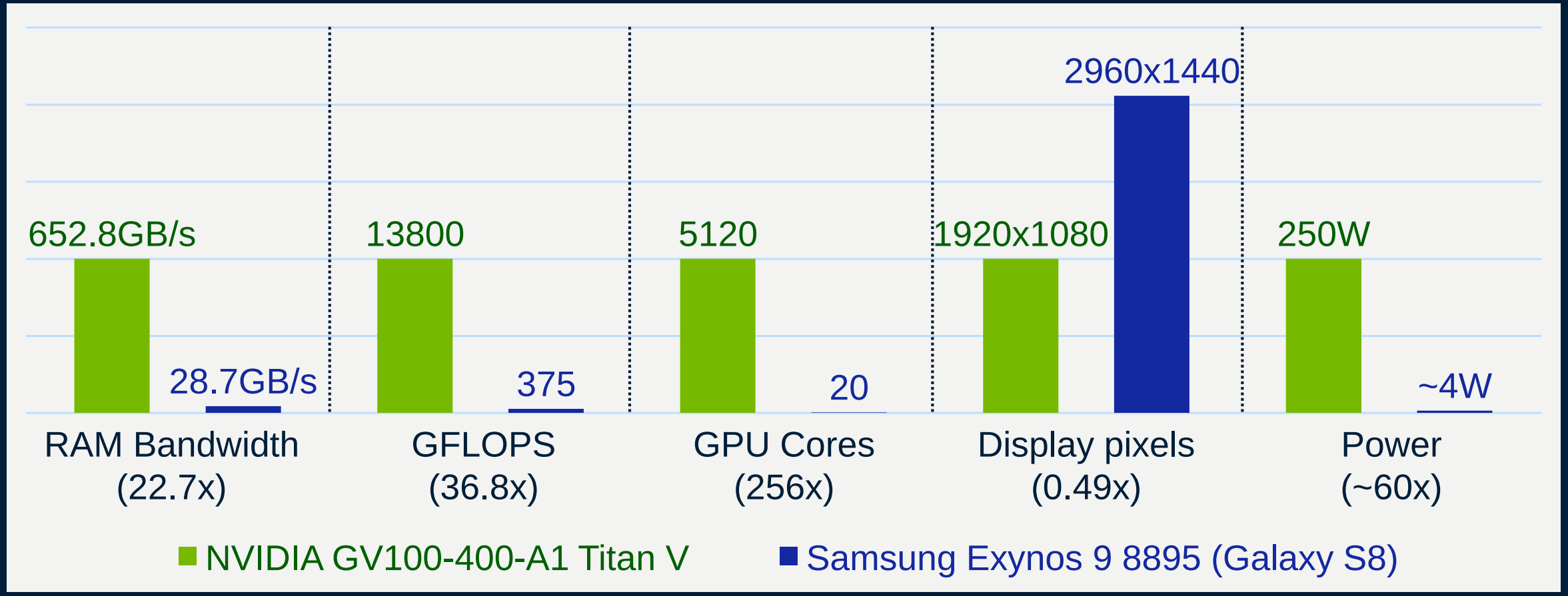
What's "desktop"?

- Relatively unlimited power budget
  - Mains powered or big batteries
- Active cooling (usually)
- Dedicated memory interface (usually)
- Might be "mobile" (laptops, tablets)

What's "mobile"?

- Limited power budget
  - Small batteries or low cable power
- Passive cooling (usually)
- Shared memory interface (usually)
- Might not be "mobile" (embedded, STB, automotive)

Totally unfair comparison



Source: Wikipedia

How do you do that?

- Thoughts are cheap
- Memories are golden
- It all depends what you're doing
  - "Have you tried optimising the software?"
  - If you can't do it right, cheat

GENERATIONS / VANCOUVER 12-16 AUGUST
SIGGRAPH2018

How do *they* do that?

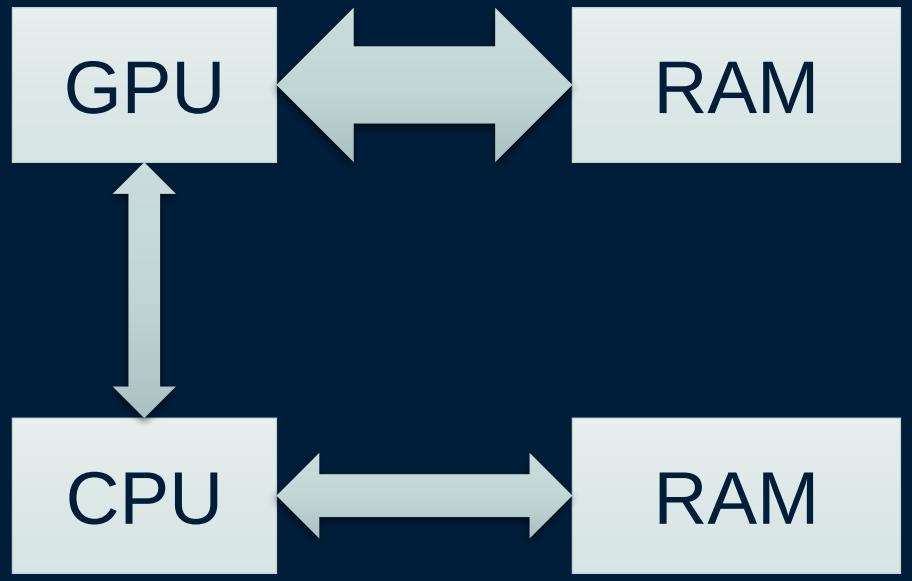
I get by with a little help from my friends

Main differences

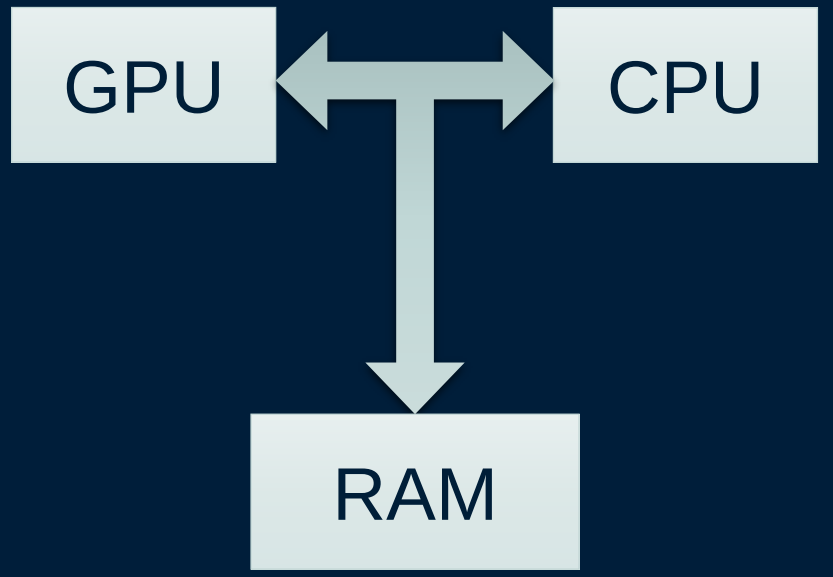
- Shared memory
- Tiled rendering
- Cache usage
- Thermal throttling

Shared memory

Desktop (esp. discrete)



Mobile (almost all)



Shared memory

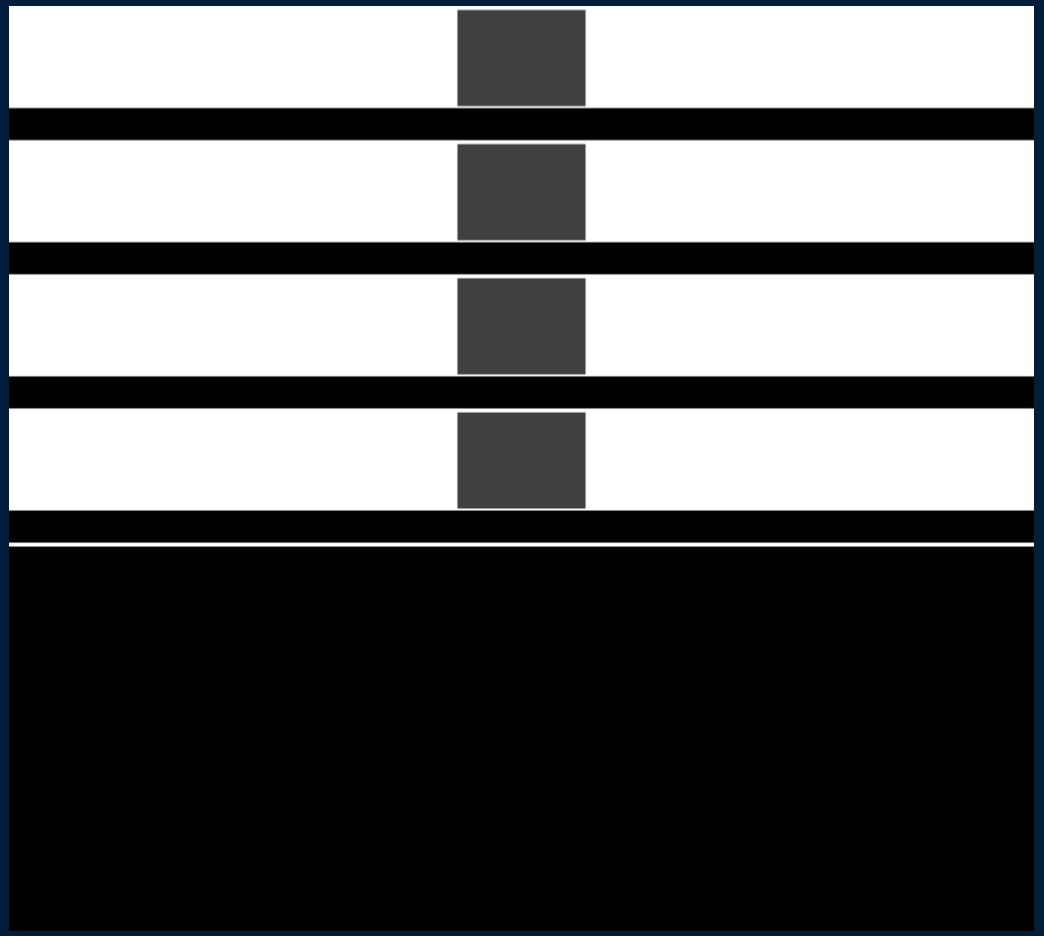
- There probably isn’t a small special area for xfers
- You probably don’t need a transfer buffer
  - But you might for layout/tiling
- Shared memory doesn’t mean shared caches!

Tiling: Classical IMR

- “Immediate mode” rendering:
  triangles render in the order submitted
- Depth updated as you go
- It’s never this simple
  - We need more parallelism!

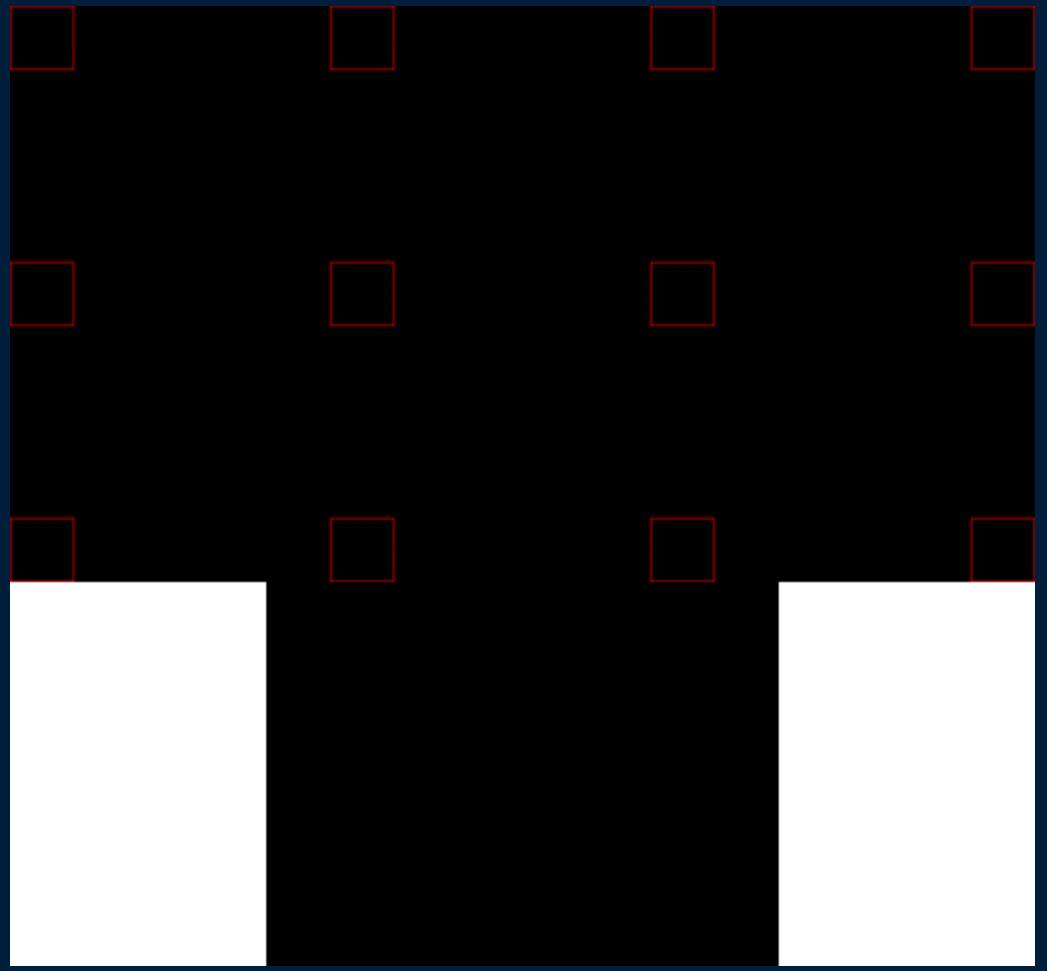
Tiling: Classical IMR

- IMR uses lots of bandwidth
- Triangle order doesn’t suit memory access
- Sane hardware uses a tiled frame buffer, not linear cache
  - But it still doesn’t help much



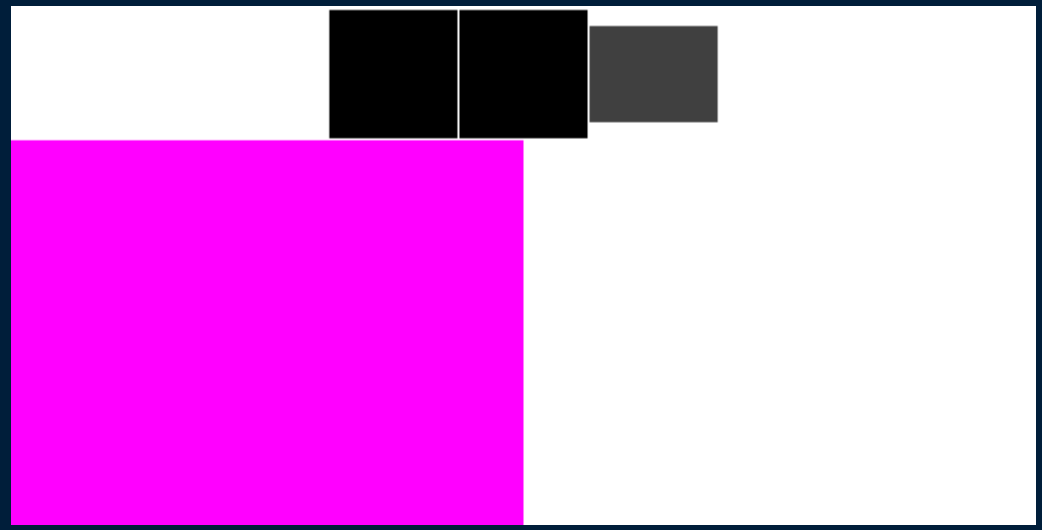
Tiling: Binning pass

- Tile the frame buffer
- Work out which triangles fall in which tiles
- Details are proprietary



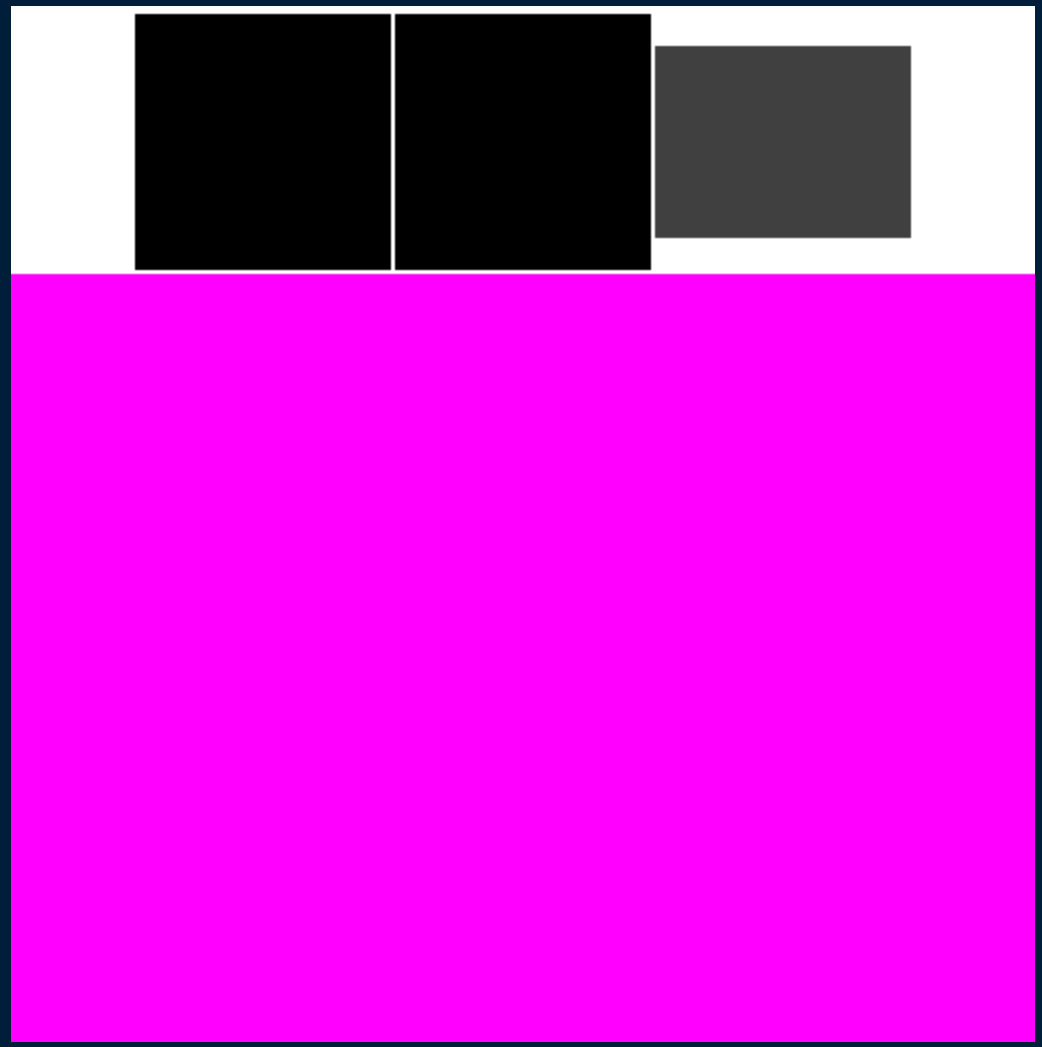
Tiling: Rasterising

- Rasterise all the triangles in each tile
- Tiles get processed independently
- Only touch the framebuffer pixels once



Tiling: Benefits

- No off-chip cost for MSAA
- May not need to write Z

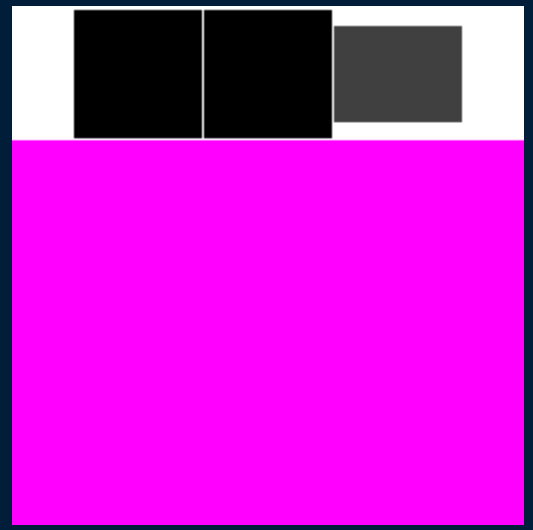


Tiling: Problems

- Geometry has to be stored
- Latency (can’t rasterise until you’re done binning)*
- Can’t access the frame buffer during rendering**
  - Resources stay in use for the whole frame

Tiling: deferred shading

- You *can* access the *current* pixel
- Subpasses in Vulkan
- PLS in OpenGL ES
- Metal2 imageblocks
- Not a panacea (blur/flare, other post-processing)



Cache

- Bandwidth is not just the framebuffer
- Cache is not large
- Use compressed textures where appropriate
- Reduce resolutions
- Use mipmapping where possible

Thermal throttling

- Throttle back when it’s too hot
- Peak speed cannot be maintained
  - Passive cooling (usually)
  - Cases make it worse
- Degrade gracefully
- Consider bursty workloads

Stay cool

It’s easy to be cool with enough airflow

API porting

- Start with a next-gen API and back-port
  - Easier to retrofit a simpler API than route data
  - Easier to serialise than parallelise
- Pick the most optimised path
  - *Please* use Vulkan subpasses!

GENERATIONS / VANCOUVER 12-16 AUGUST
SIGGRAPH2018

Next-gen graphics API features

How mobile GPUs use all the extra typing you had to do

Render passes

- Distribute work over the framebuffer
  - All targets have to be the same size!
  - On a tiler, determines tile traversal
  - May determine tile shape
- Targets written to in parallel

Subpasses

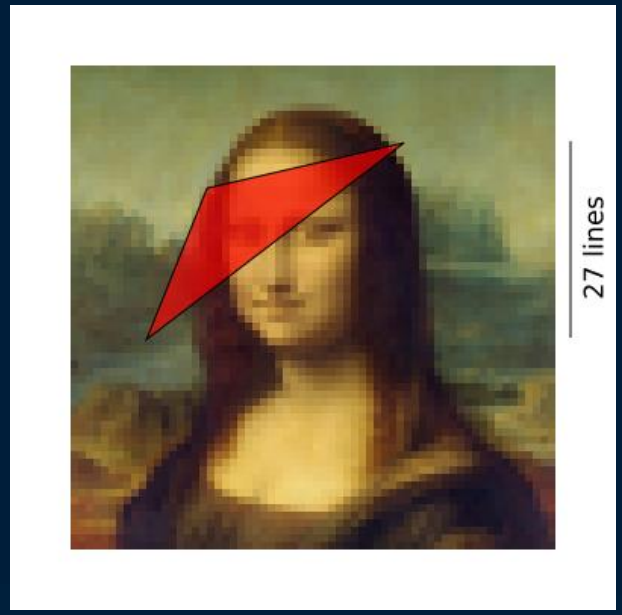
- Vulkan feature, see also Metal2 Imageblocks
- Order work within the tile
- Allow tile memory to be reused
- Can be reordered (dependencies)
- Local access only

(Don’t) load and store

- If you don’t need off-chip data, don’t ask for it
  - Easy to leave transferring accidentally when porting
- DONT_CARE better than CLEAR better than LOAD
- DONT_CARE better than STORE
- N.B. Overriding in later passes

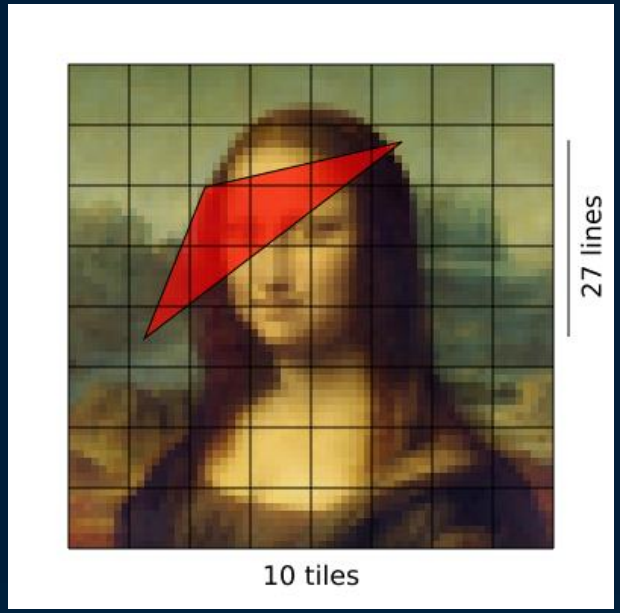
Tiling (of images)

- Raster order doesn't usually suit textures



Tiling (of images)

- Raster order doesn't usually suit textures



Tiling (of images)

- Raster order doesn’t usually suit textures
- Linear “tiling” is useful for frequent updates
- Use TILING_OPTIMAL for better GPU cache access
  - Aka swizzled textures
- Details are proprietary (to fit caches)

Layout (of images)

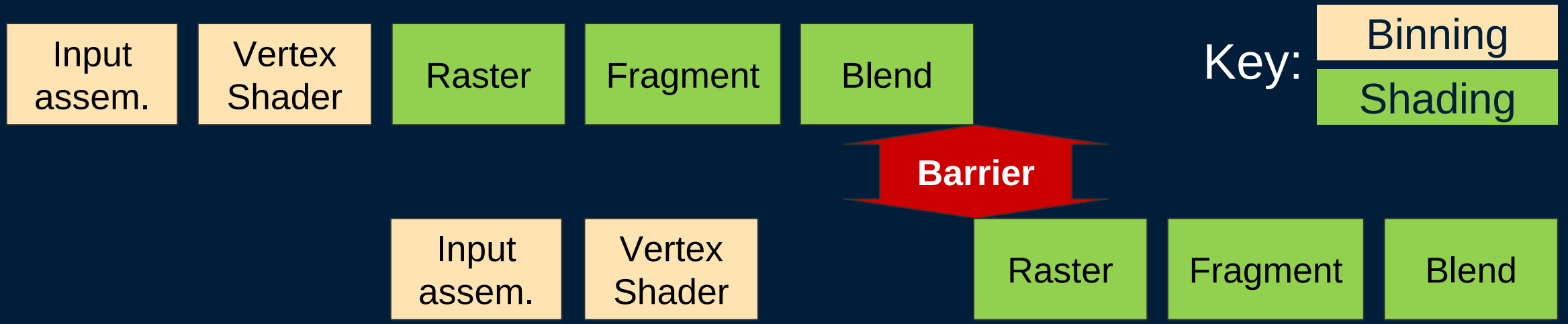
- Not the same thing as tiling
- Intended for things like compression schemes
- Don’t assume LAYOUT_GENERAL is always best
  - Future hardware/drivers might have custom options
- Details are proprietary

Texture formats

- Next gen API drivers shouldn’t lie to you
  - WYAFIWYG - some formats might be “missing”
- Compressed textures can help *a lot*
- ASTC can be a lot smaller
- Back to bandwidth

Synchronisation

- The mobile pipeline has a big latency gap
- Using the right pipeline stage is important



Synchronisation

- The mobile pipeline has a big latency gap
- Using the right pipeline stage is important
  - Don’t leave too big a gap (performance)
  - Don’t leave too small a gap! (errors)
  - Don’t assume all hardware is the same
  - Tobias Hector has a helpful library!

Command buffers

- Command buffer building is the slow bit
  - Don’t do it on the main submission thread
- Parallelise (there are lots of cores)
- Use secondary command buffers only if needed
- Watch lifetimes – don’t kill a buffer that’s in use

Memory heaps

- It’s complicated (AMD has a library)
- Getting an allocation is slow
  - *Please* sub-allocate (except for very big images)
- Different requirements might need different heaps
- Different hardware handles this differently

Frame buffers

- Framebuffers are tied to render passes
  - Hardware needs this interaction to be configured properly
- Ensure you can get the information you need
- Don’t forget you can reuse memory

Pipelines

- Drivers need the full shader sequence to build the pipeline (there's cross-stage optimisation)
- Mix-and-match shaders can be a problem
  - Some engines have thousands of shaders that are unused
- If you can simplify up front, do
- Use the caches (deltas are less important)

GENERATIONS / VANCOUVER 12-16 AUGUST
SIGGRAPH2018

Problems

Opportunities!

Driver updates

- The chain from hardware vendor to customer is long
- Bug fixes and new features may take months
- On older/low-end devices they may never arrive
- Sorry, you probably need to work around issues
- This is getting better (slowly)

Driver quality

- Mobile driver teams are generally small
- There are a lot of SKUs
- It’s hard to push out frequent updates
- There’s a concerted effort to improve this (CTS)
- Please bear with us

Feature limitations

- Don’t expect desktop limits and features on mobile
- New APIs won’t emulate functionality
  - “But you can do this in GL”
  - Developers want predictable performance, not this
- Check per device – limits are raising

Tooling

- Vendor-specific tools are often quite good
- General tools are more limited (WIP)
- Use validation
  - It won’t catch everything, but it’ll catch a lot
  - Unless you’re embedded, future devices may change

Legacy engines

- "But our engine doesn't work like that"
  - We've been warning you…
- Drivers don't do hidden optimisations any more
  - Predictable performance is predictably *low*!
- Much better news if you're starting from scratch

GENERATIONS / VANCOUVER 12-16 AUGUST
SIGGRAPH2018

Design for the target

Inconsolable: *adj., won't run on PlayStation, Xbox or Switch*

Don’t render as much

- Drop the resolution (and upscale)
- Drop the frame rate
- Hide things behind the UI
- Hide things behind the hands
- You can do this selectively

Let things cool down

- Design bursty sections
- Cool off in loading screens
- Do hard work early to avoid stuttering

Resource limits

- Use memory wisely (OOM killer!)
- Bandwidth isn’t always free
- Large games need to earn their storage
- Games that kill the battery get deleted
  - Make sure you’re not running in the background!

GENERATIONS / VANCOUVER 12-16 AUGUST
SIGGRAPH2018

Summary

Wake up now

Take-home messages

- The RAM is evil

Take-home messages

- The RAM is evil
- All GPUs are different
- Some are more different than others
- The situation is improving
- We need your help

More information

- https://developer.samsung.com/game/
- a.garrard at samsung.com
- Talk to us, your chip vendors and Khronos!