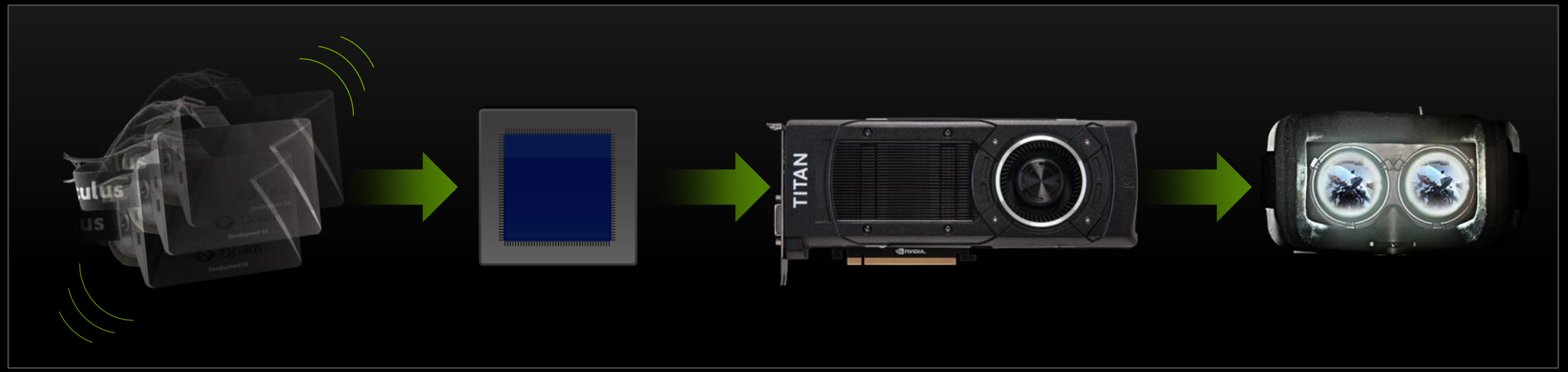
GAMEWORKS VR

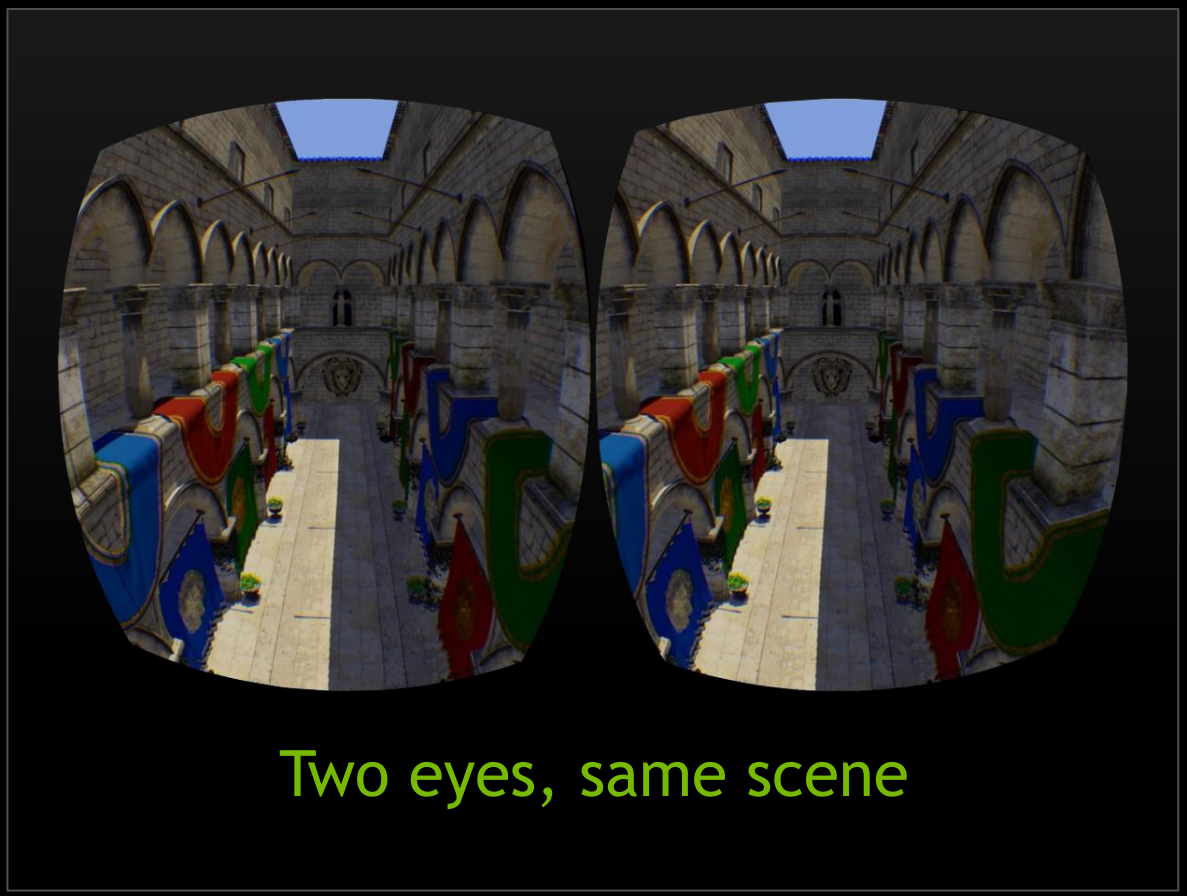
Nathan Reed — Developer Technology Engineer, NVIDIA

LATENCY



Motion to photons in ≤ 20 ms

STEREO RENDERING



Two eyes, same scene

GAMEWORKS VR

SDK for VR headset and game developers

MULTIRES SHADING

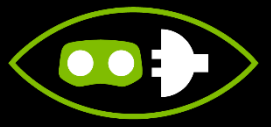
Increase performance via an innovative new way to render for VR

VR SLI

Scale performance with multiple GPUs

CONTEXT PRIORITY

Minimize head tracking latency with asynchronous time warp



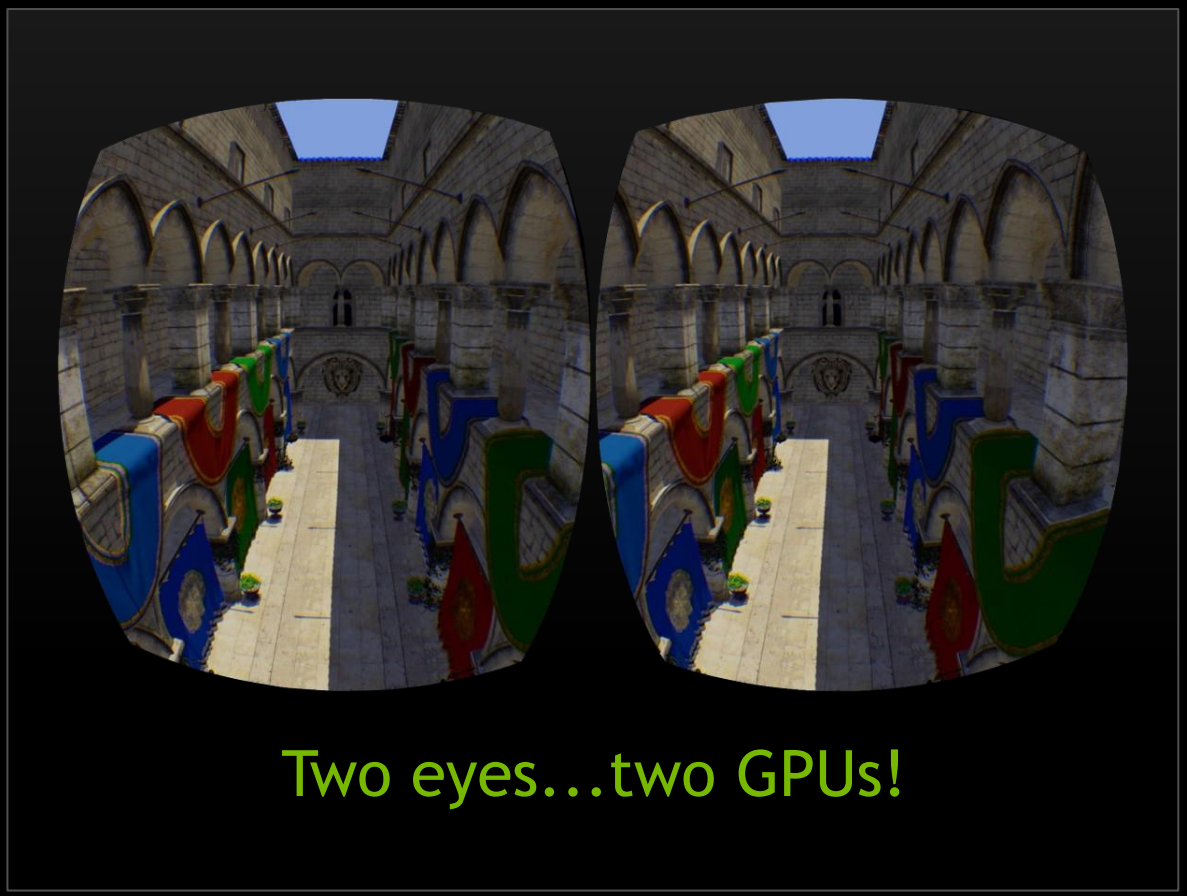
DIRECT MODE

Plug and play compatibility from GPU to HMD

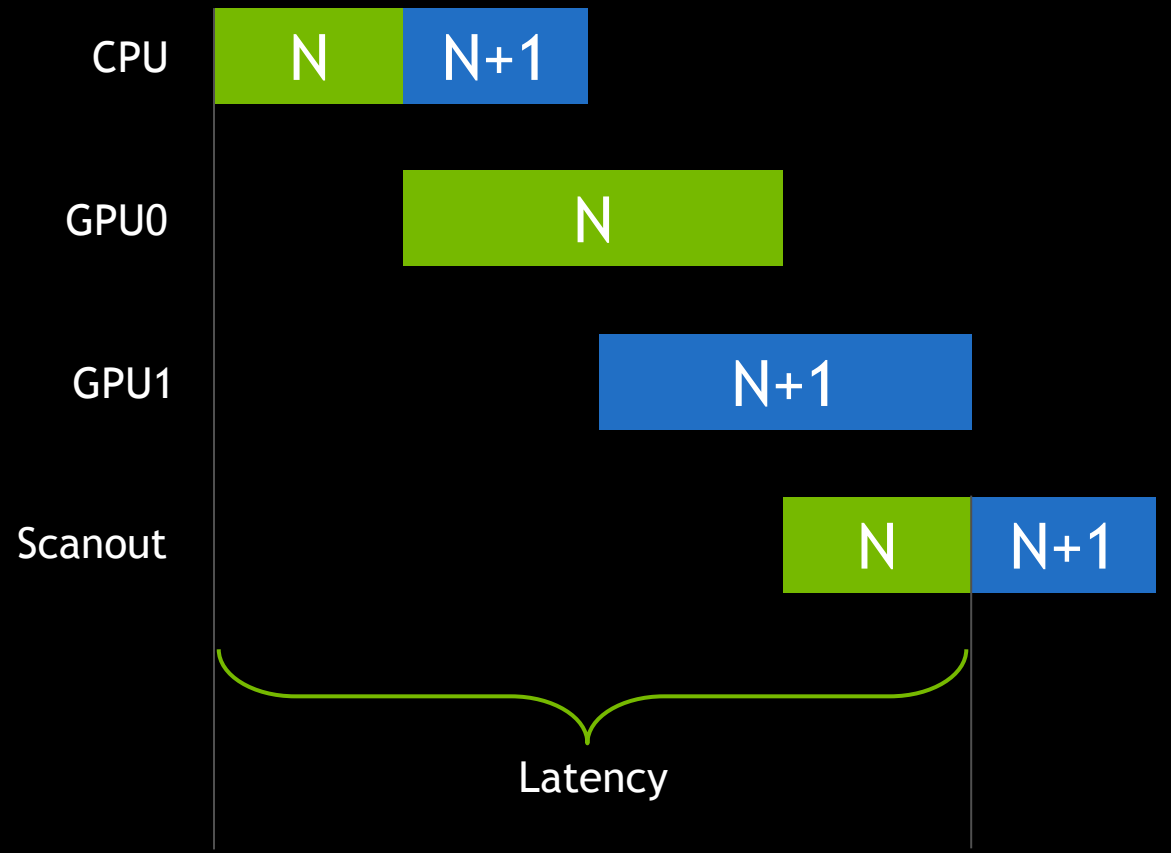
FRONT BUFFER RENDERING

Reduce latency by rendering directly to the front buffer

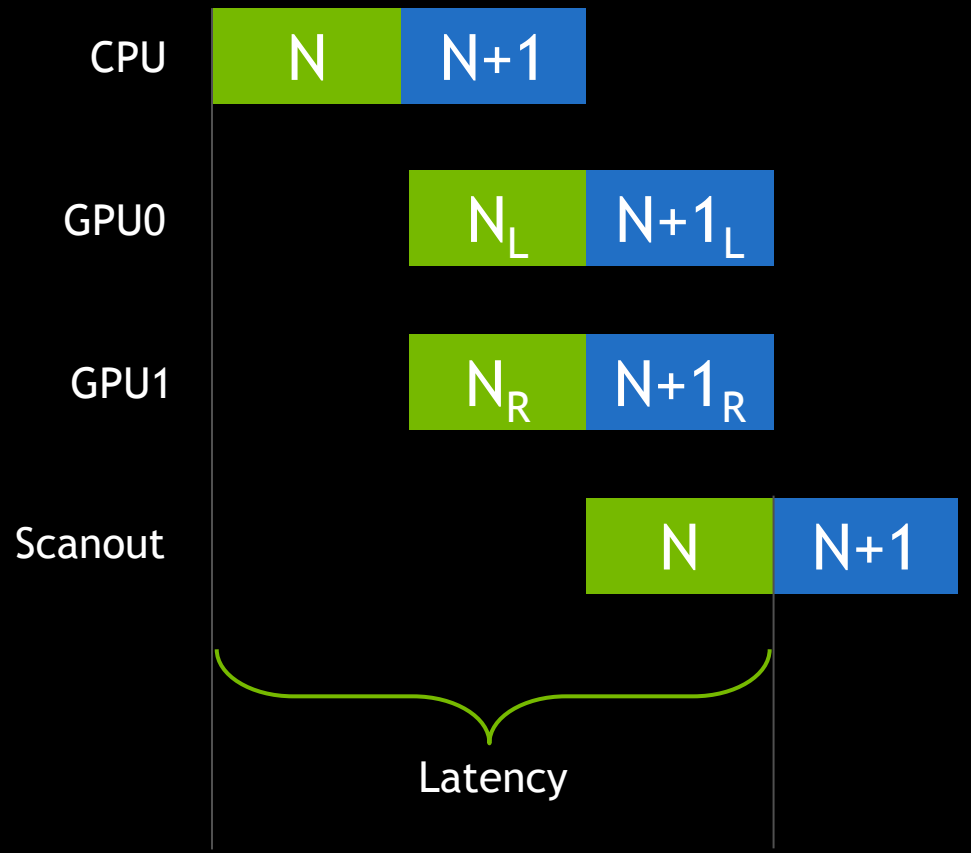
VR SLI



INTERLUDE: AFR SLI



VR SLI



VR SLI

Shared command stream

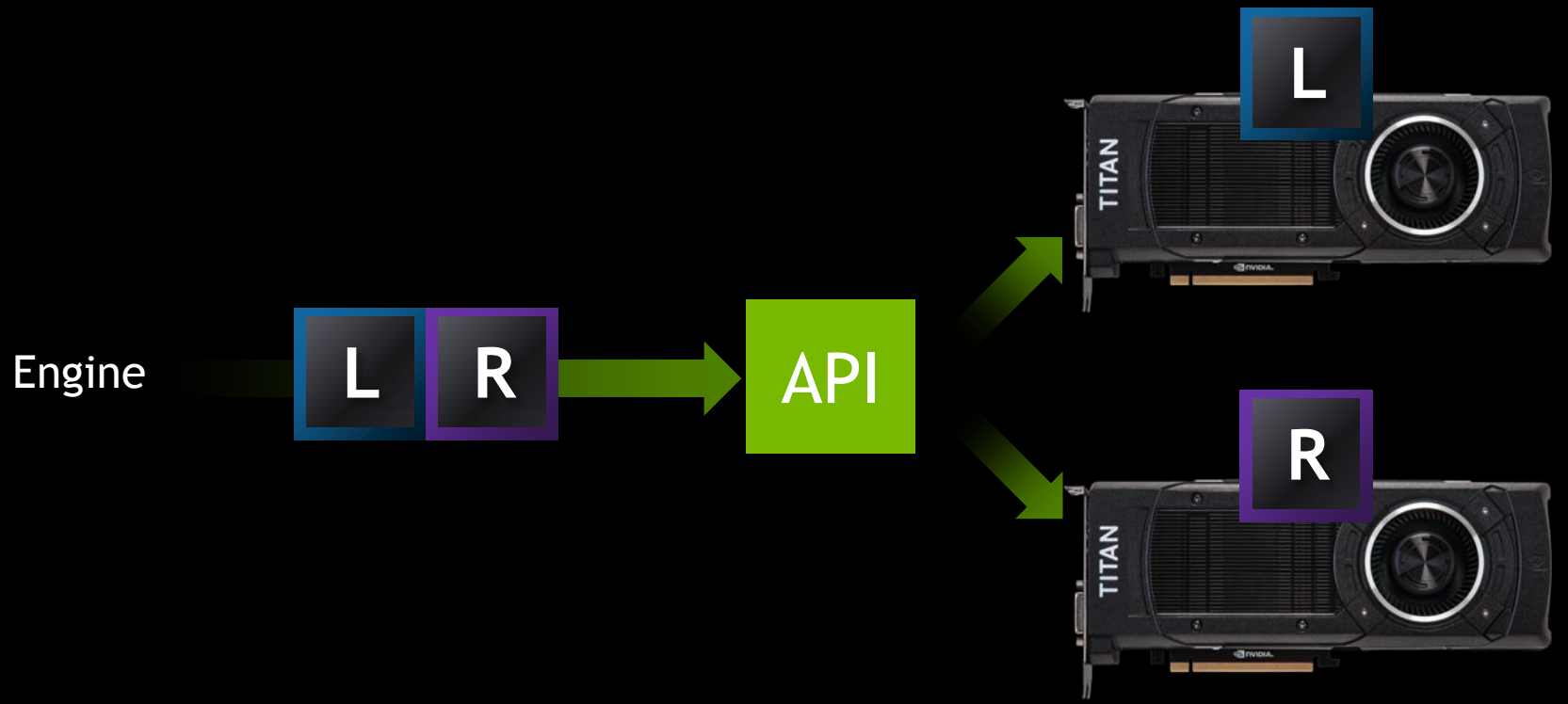
API

L

R

VR SLI

Per-GPU state | Constant buffers | Viewports/scissors



VR SLI

GPU affinity masking

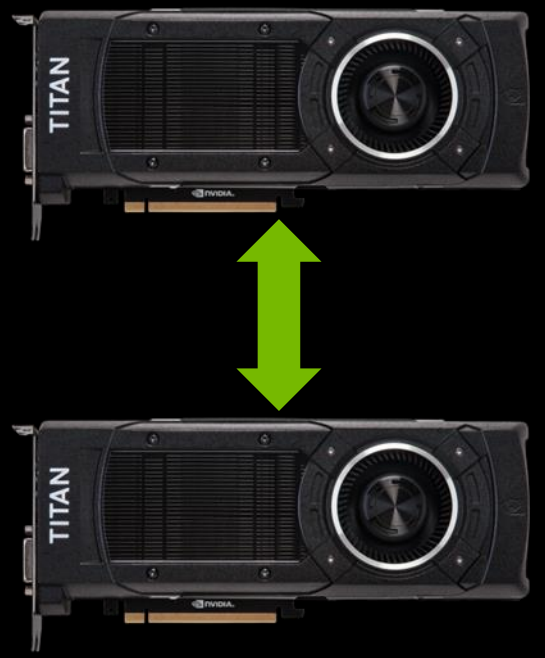
SetGPUMask(1)

SetGPUMask(3)

SetGPUMask(2)

VR SLI

Cross-GPU data copies, via PCIe



VR SLI PERFORMANCE SCALING

- Up to the app to decide how to use GPUs
  - Needs engine integration
- Scaling depends on the app
- Duplicating work → less scaling
  - Shadow maps
  - GPU particles, physics sims

DEVELOPER GUIDANCE

- Teach your engine to render both views at once
- Currently:

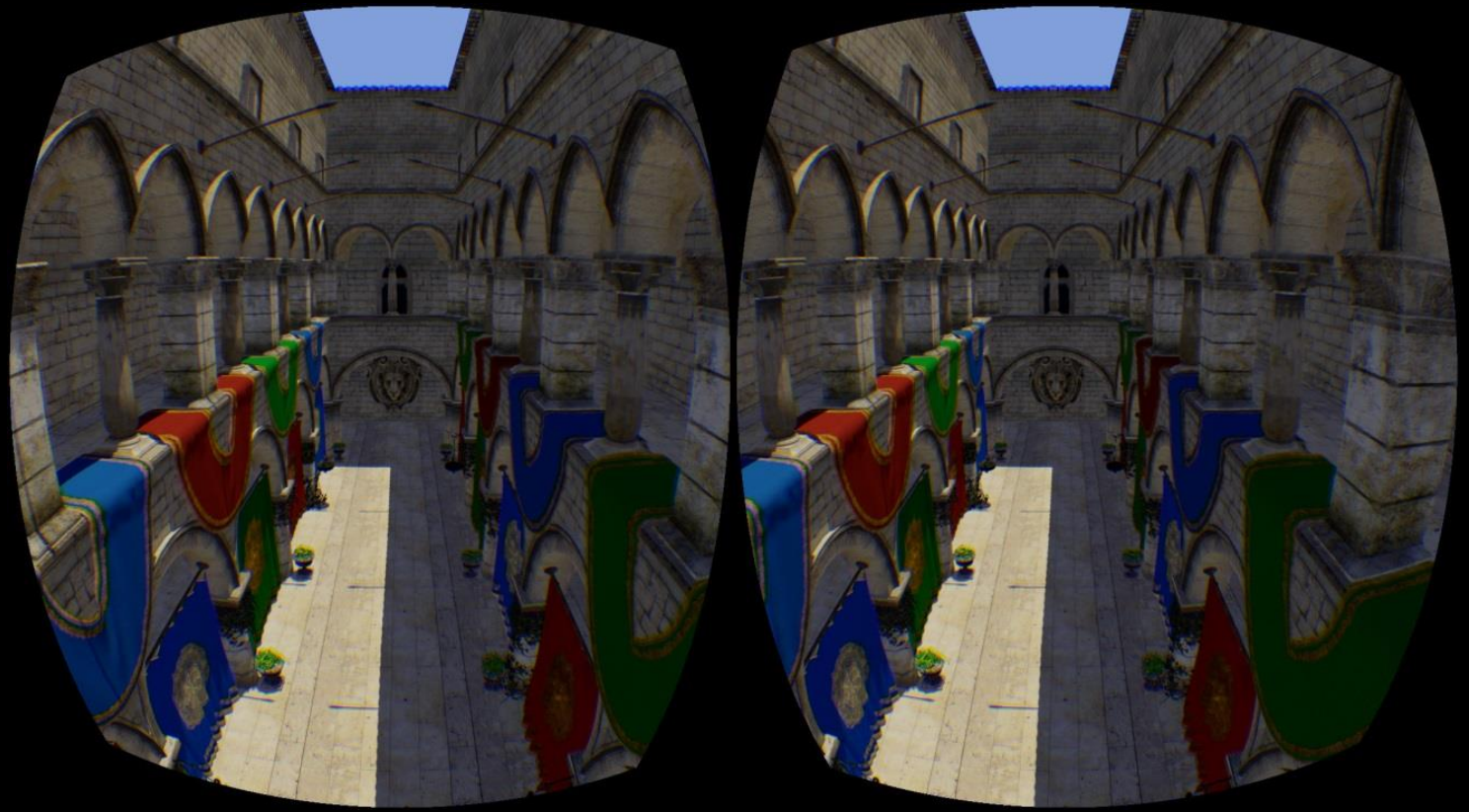
```
for (each view)
 find_objects();
 for (each object)
   update_constants();
   render();
```


DEVELOPER GUIDANCE

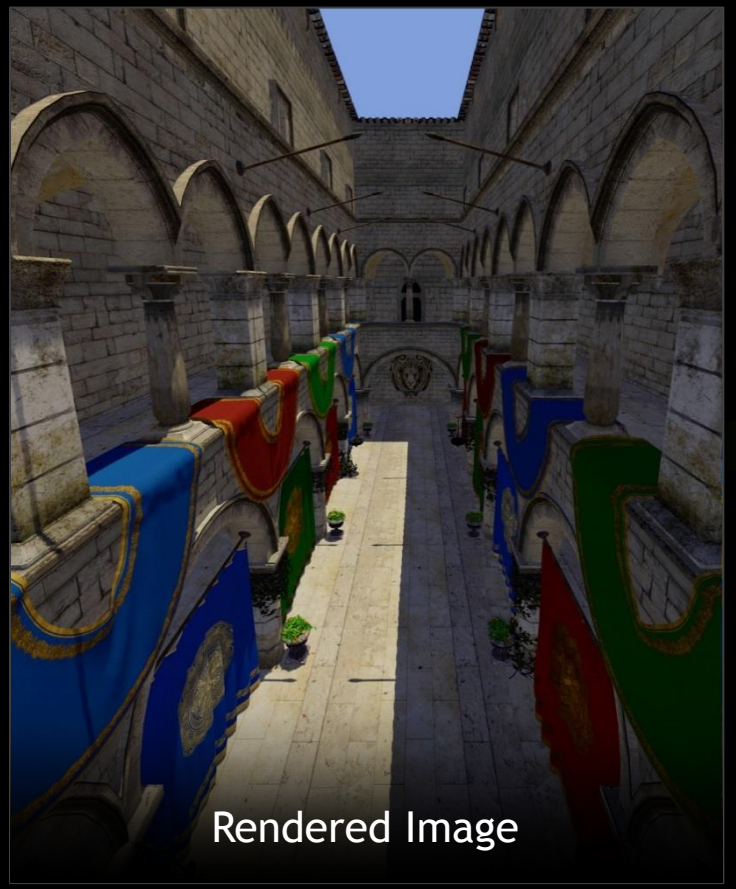
- Where you want to end up:

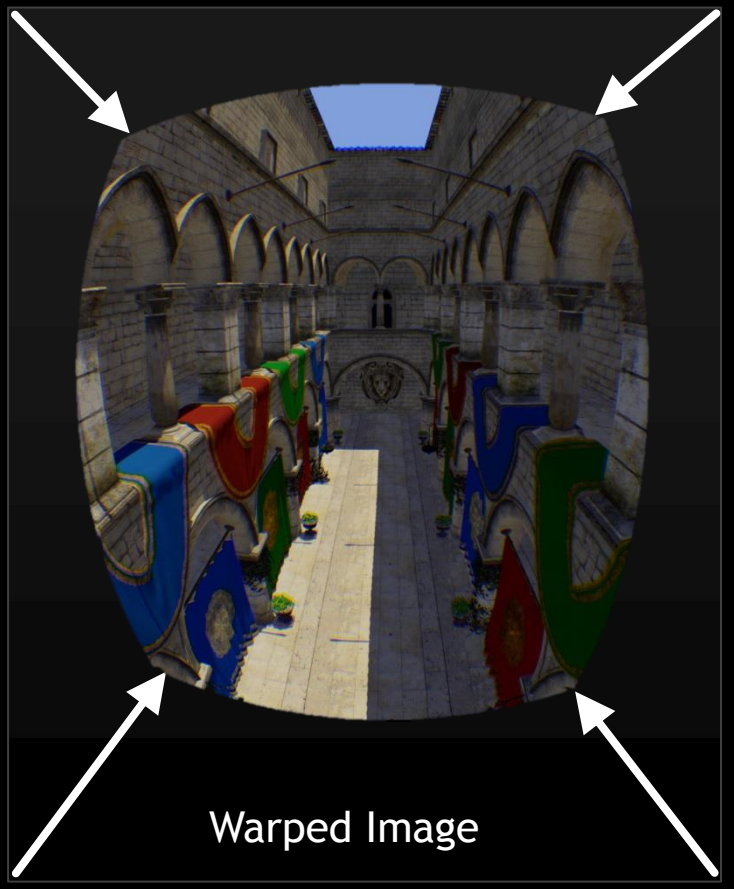
```
find_objects();
for (each object)
  for (each view)
    update_constants();
  render();
```


MULTI-RESOLUTION SHADING

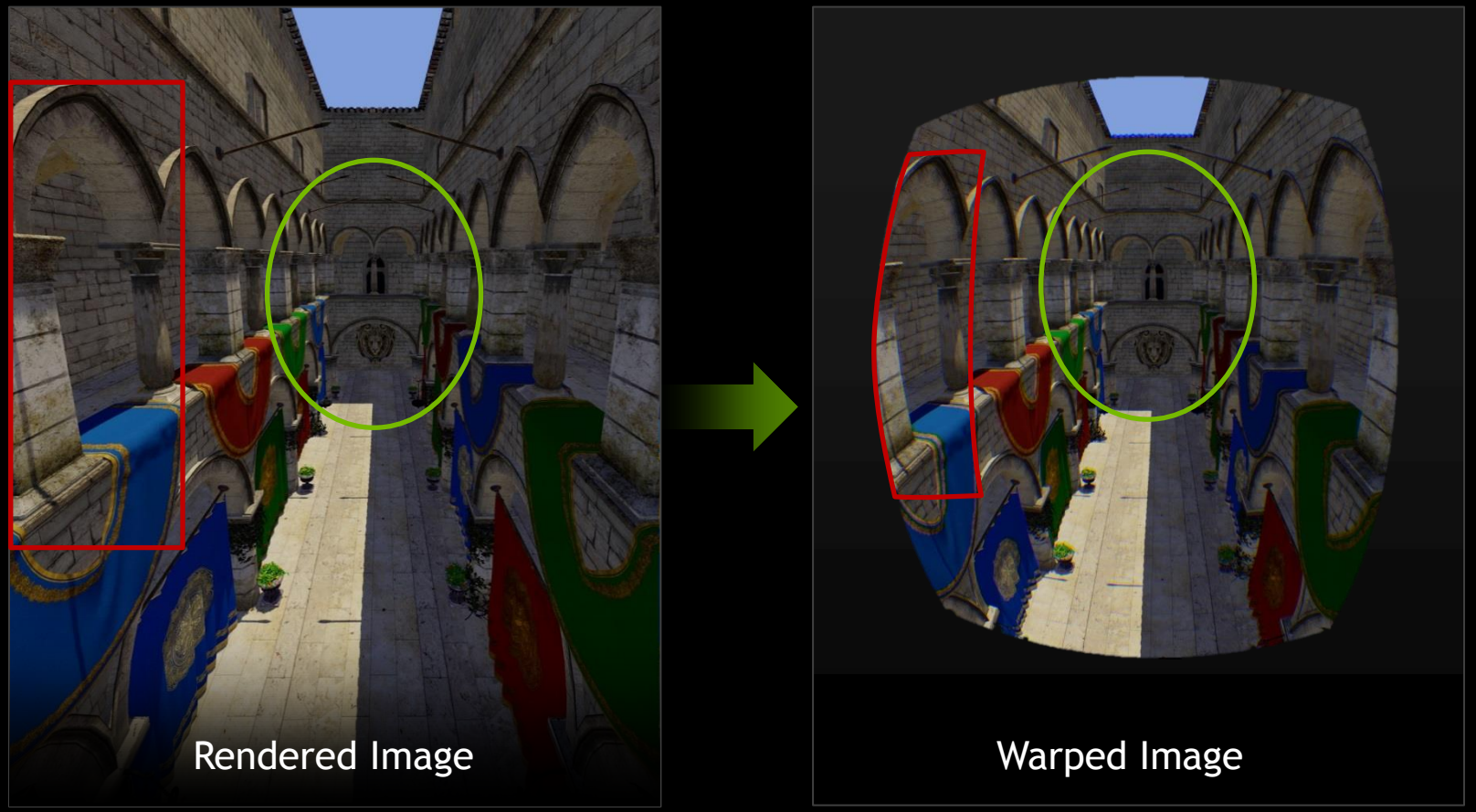


LENS DISTORTION

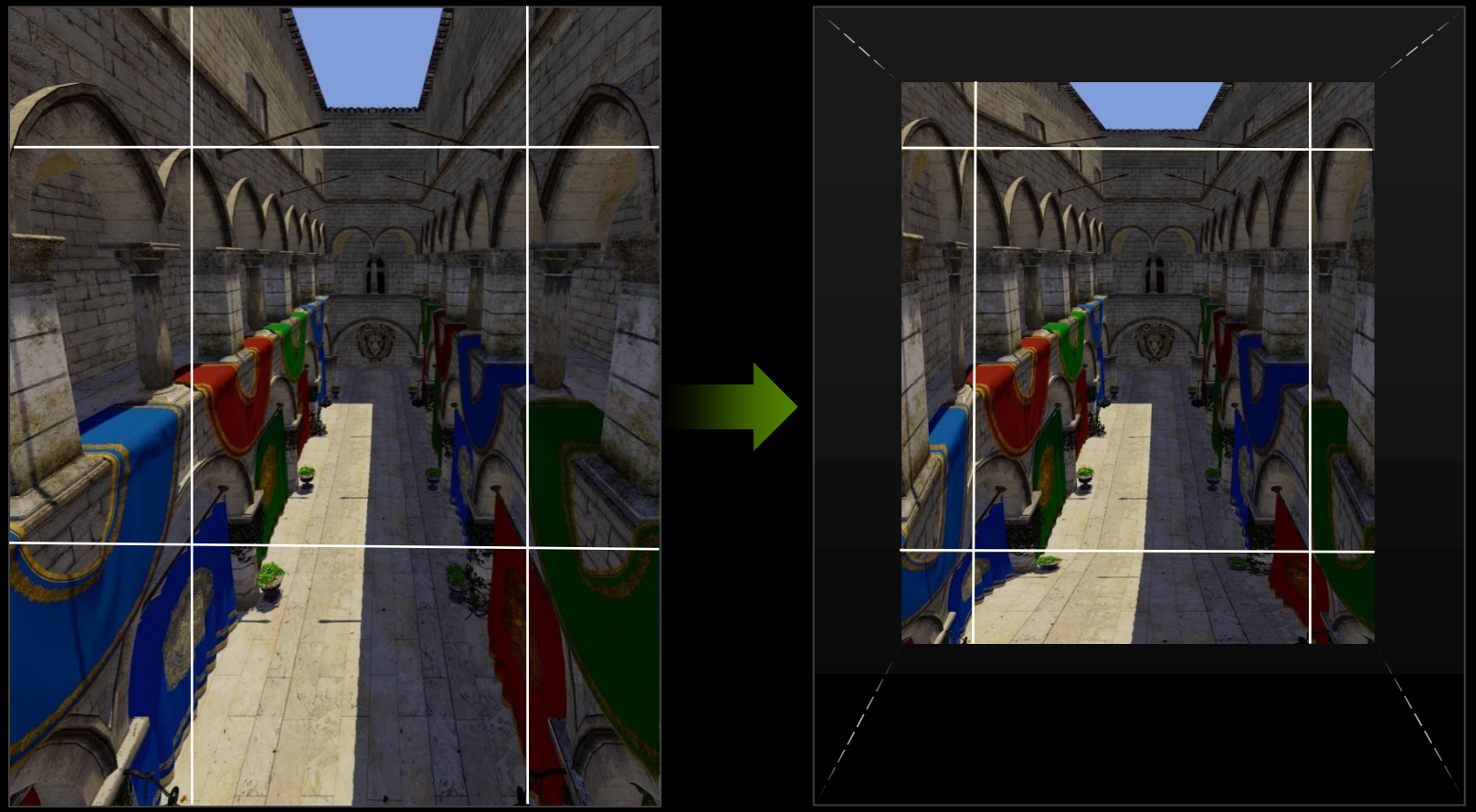




LENS DISTORTION

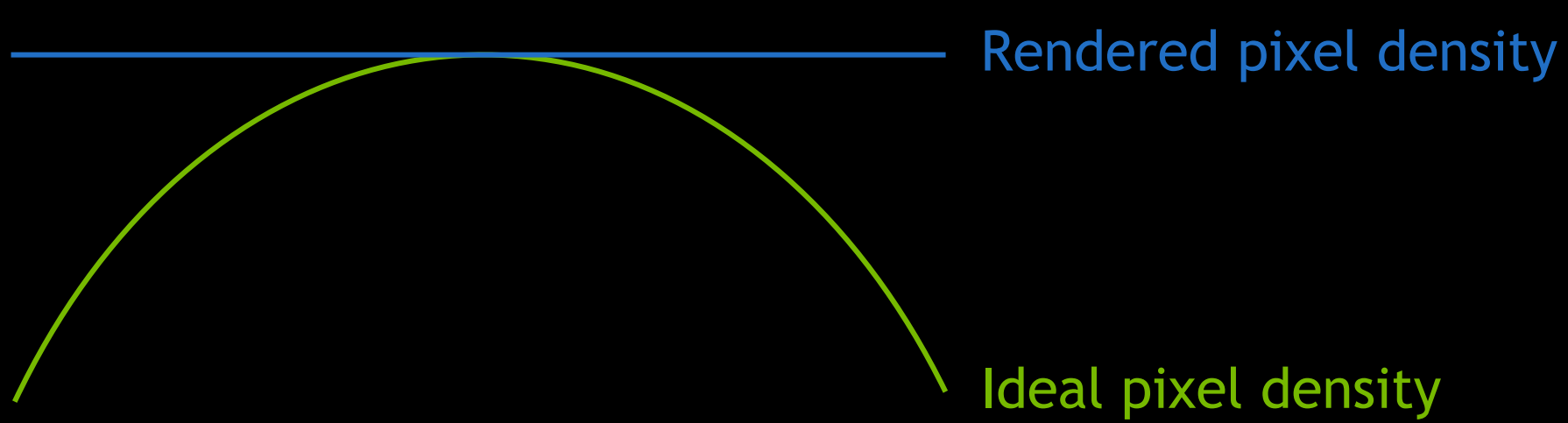


MULTI-RESOLUTION SHADING



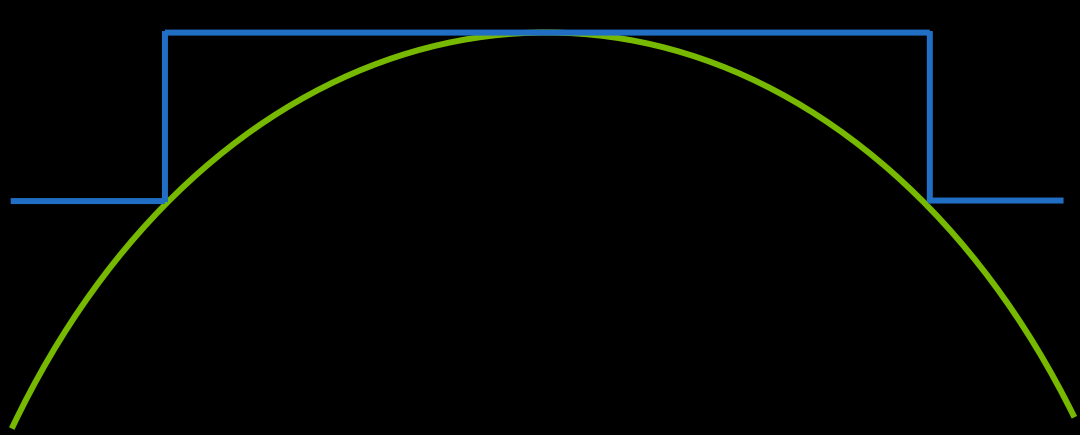
STANDARD RENDERING

Maximum density everywhere



CONSERVATIVE MULTI-RES

25% pixels saved = 1.3x pixel shading speedup

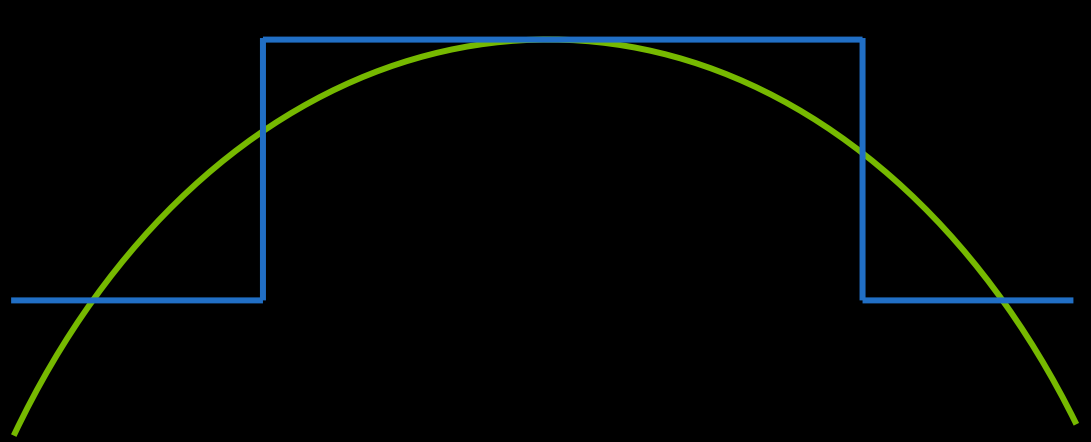


Rendered pixel density

Ideal pixel density

AGGRESSIVE MULTI-RES

50% pixels saved = 2x pixel shading speedup

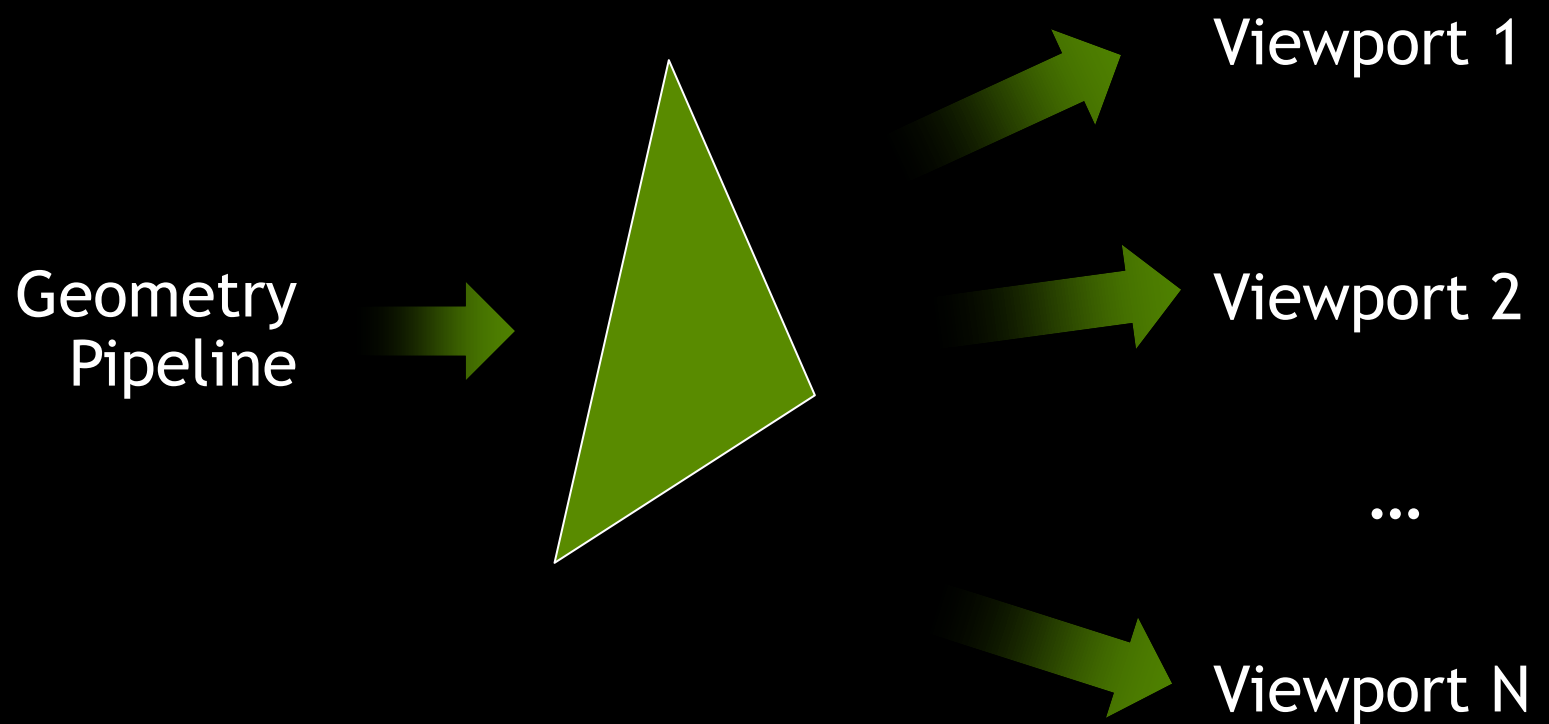


Rendered pixel density

Ideal pixel density

FAST VIEWPORT BROADCAST

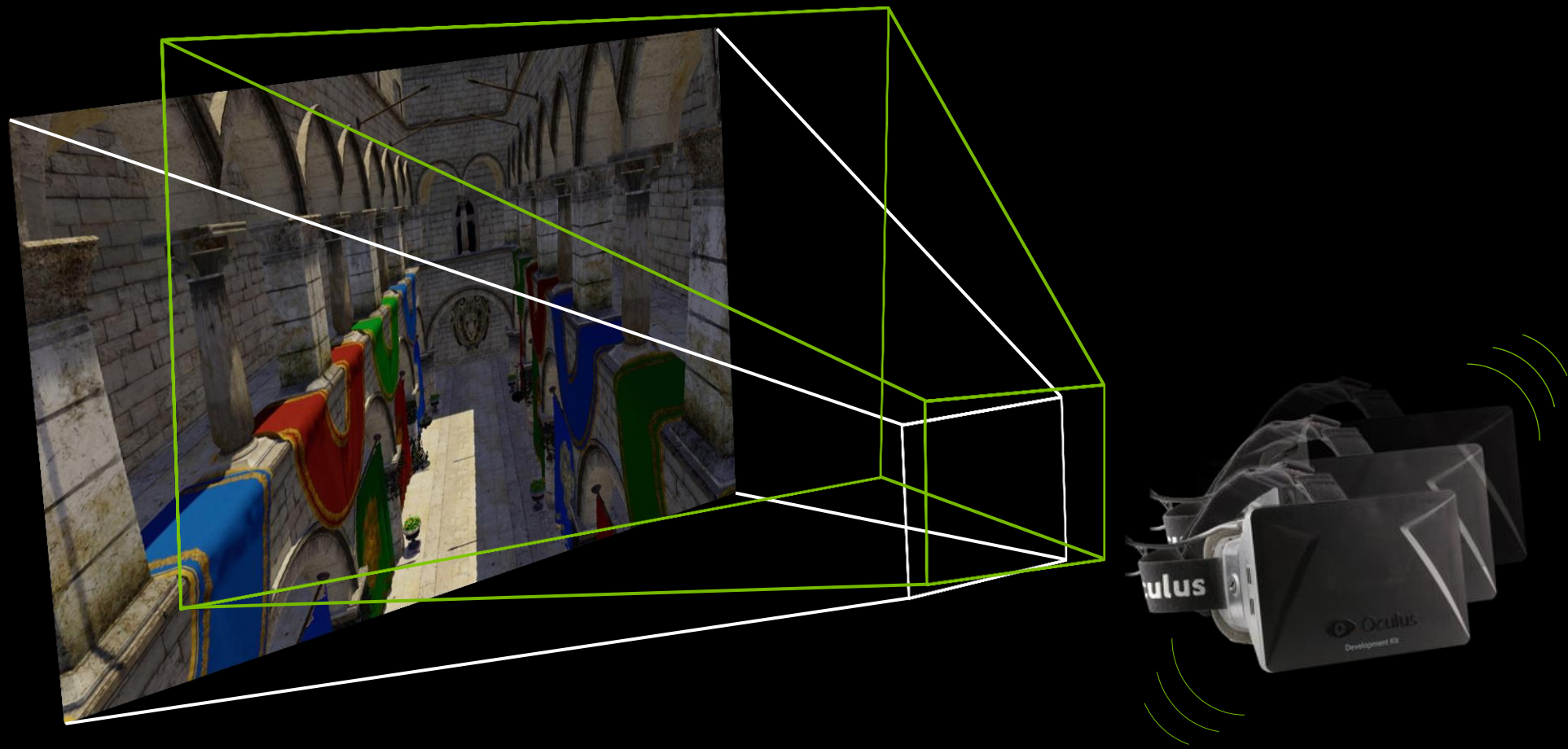
Maxwell multi-projection



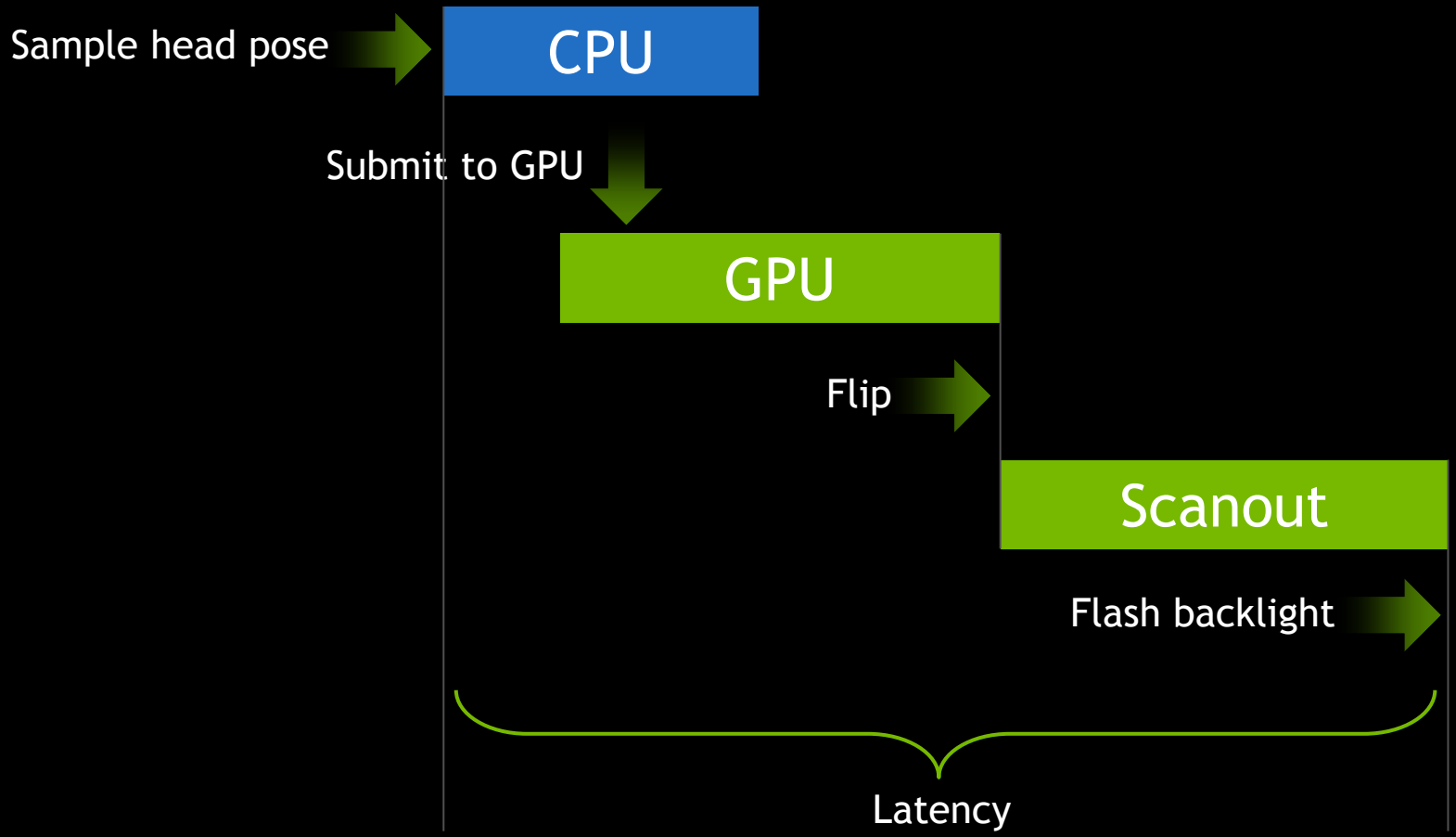
CONTEXT PRIORITY

- Enable VR platform vendors to implement asynchronous timewarp
- Via GPU preemption

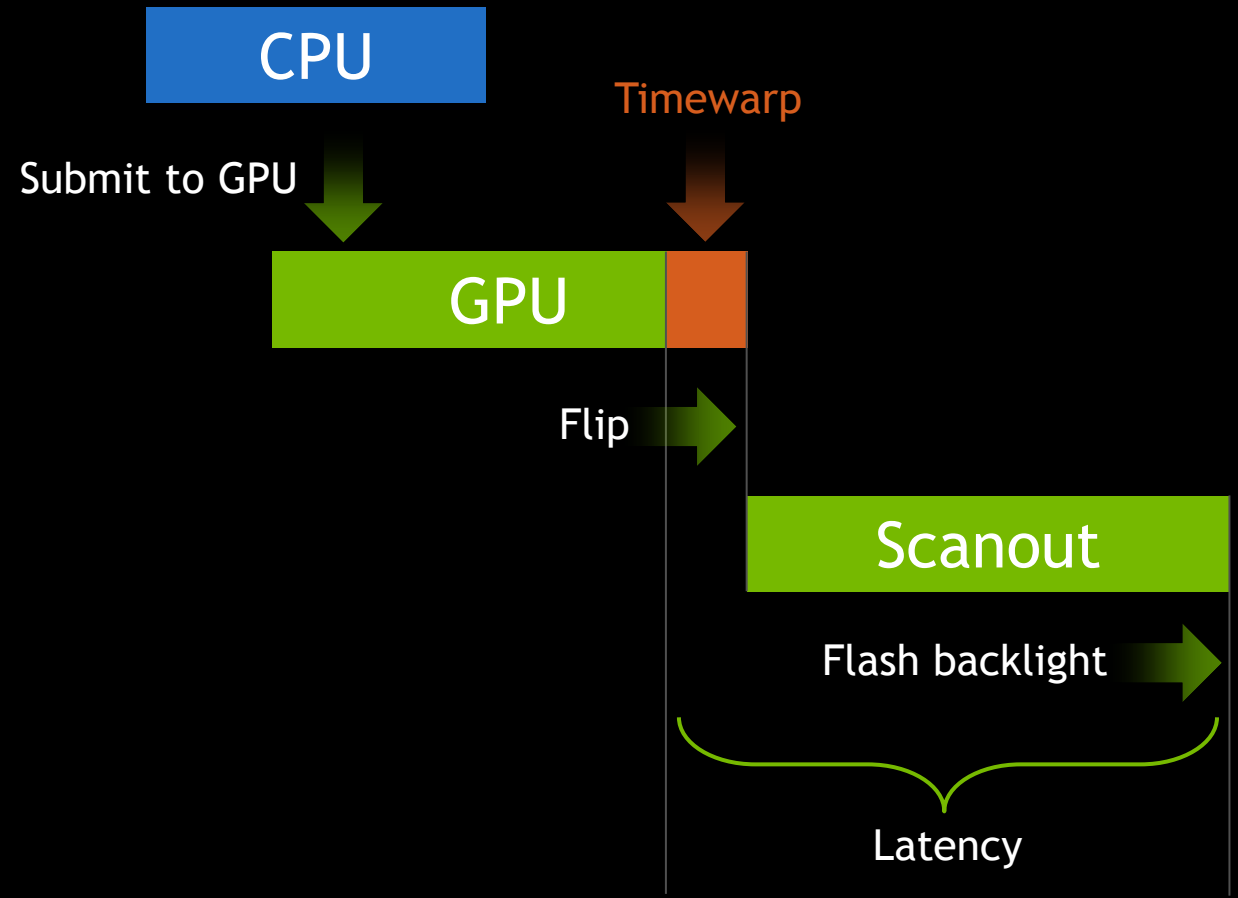
TIMEWARP



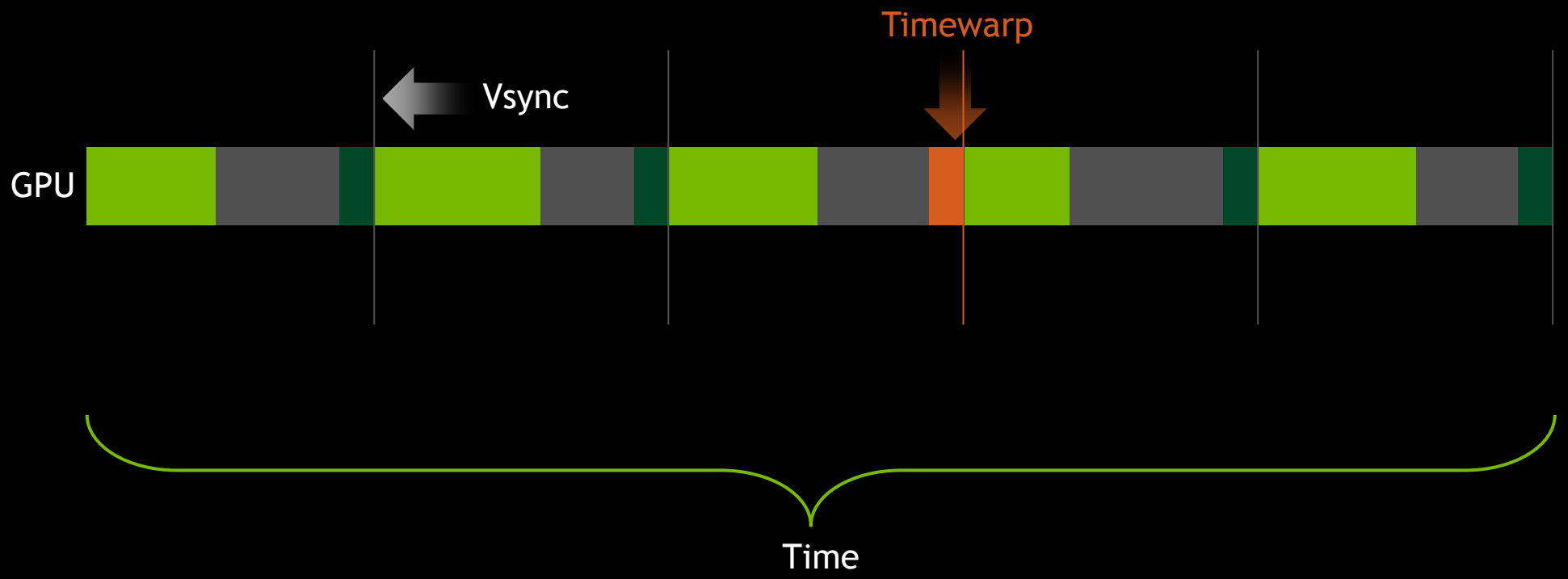
WITHOUT TIMEWARP



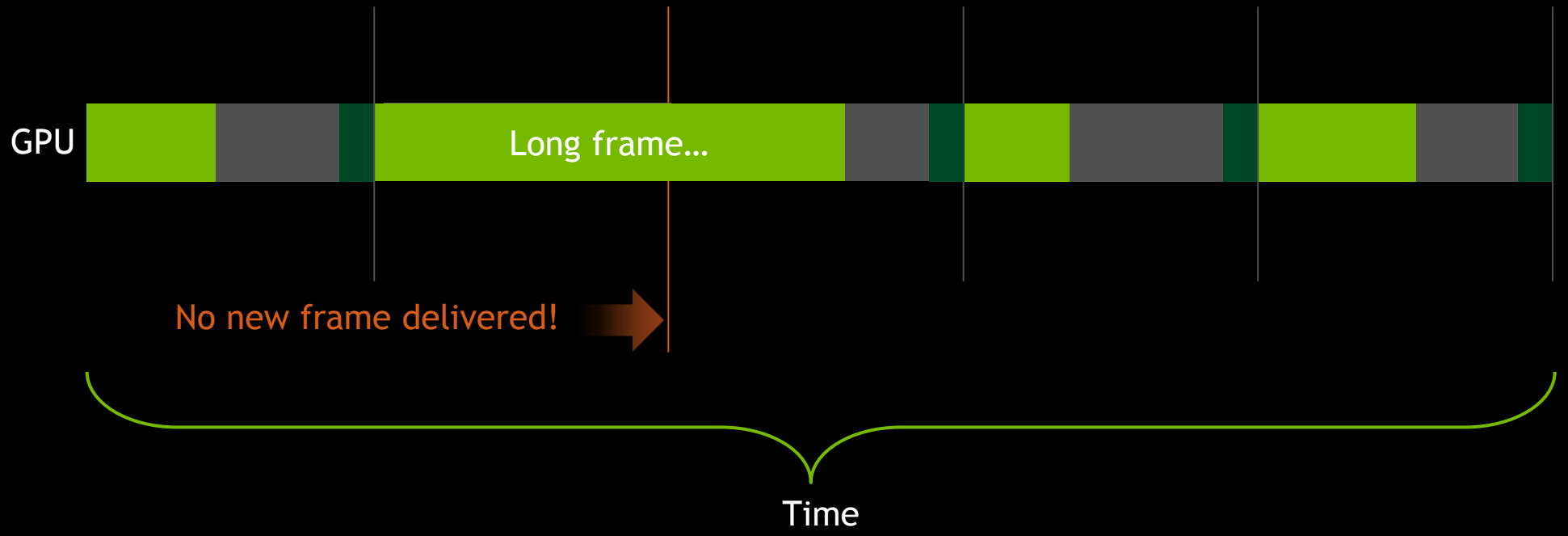
WITH TIMEWARP



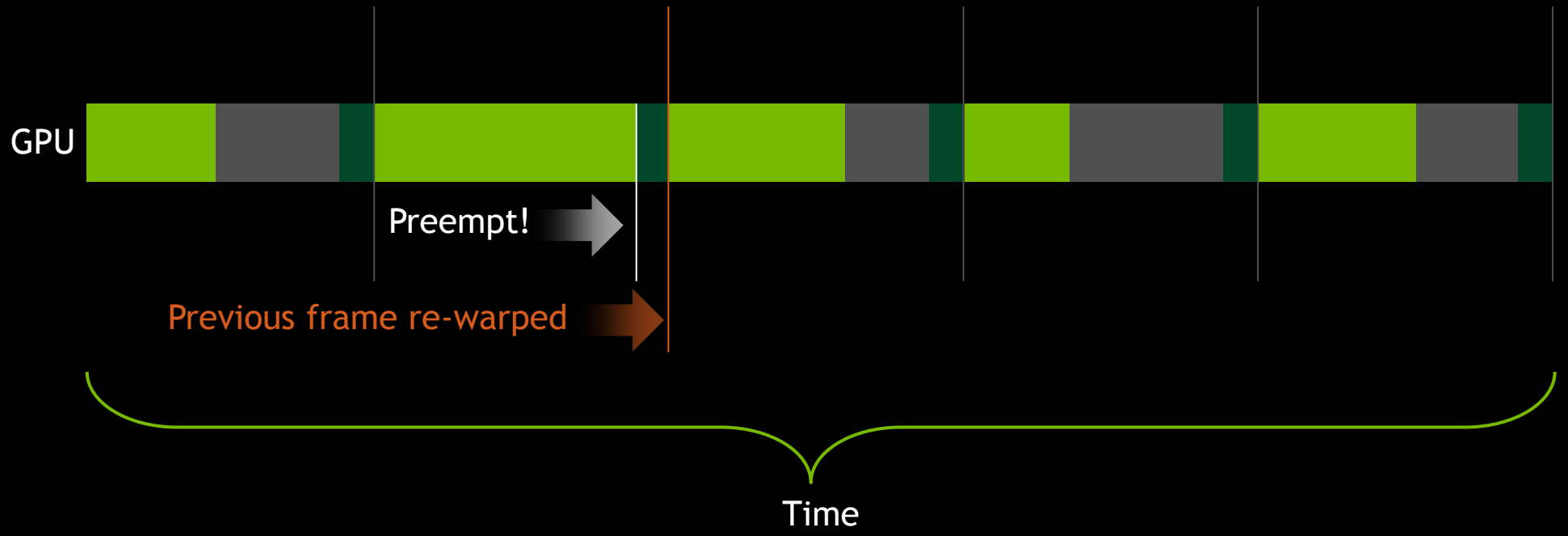
STEADY FRAMERATE



HITCHING



ASYNC TIMEWARP



HIGH-PRIORITY CONTEXT

- NVIDIA supports high-priority graphics context
  - Preempts other GPU work
- Main rendering → normal context
- Timewarp rendering → high-pri context

PREEMPTION

- Current GPUs: draw-level preemption
- Can only switch at draw call boundaries!
- Long draw can delay context switch

DEVELOPER GUIDANCE

- Still try to render at native framerate! (90 Hz)
  - Better experience
  - Async timewarp is a safety net
- Long draws could cause hitches
  - Split up draws that take >1 ms or so
  - E.g. heavy postprocessing: split in screen-space tiles

DIRECT MODE

- Prevent desktop from extending onto VR headset
- Hide display from OS, but let VR apps render to it
- Better user experience

FRONT BUFFER RENDERING

- Normally not accessible in D3D11
- Direct Mode enables access to front buffer
- Enables low-level latency optimizations
  - Render during vblank
  - Beam-racing

GAMEWORKS VR

Faster performance, lower latency, and better compatibility

MULTIRES SHADING

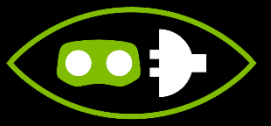
Increase performance via an innovative new way to render for VR

VR SLI

Scale performance with multiple GPUs

CONTEXT PRIORITY

Minimize head tracking latency with asynchronous time warp



DIRECT MODE

Plug and play compatibility from GPU to HMD

FRONT BUFFER RENDERING

Reduce latency by rendering directly to the front buffer

API/PLATFORM/HW SUPPORT

- Currently D3D11 only
  - OpenGL and other APIs: later
- Windows 7+
- Multi-res shading: GTX 900 series+ only!
- Everything else: GTX 500 series+
- NDA developer SDK available now