VR Direct: How NVIDIA Technology Is Improving the VR Experience

Nathan Reed — Developer Technology Engineer, NVIDIA
Dario Sancho — Lead Programmer, Crytek

Who We Are

- Nathan Reed
  - NVIDIA DevTech — 2 yrs
  - Previously: game graphics programmer at Sucker Punch
- Dario Sancho
  - Crytek — 2 ½ yrs
  - Previously: academy, system and platform programming

Hard Problems of VR

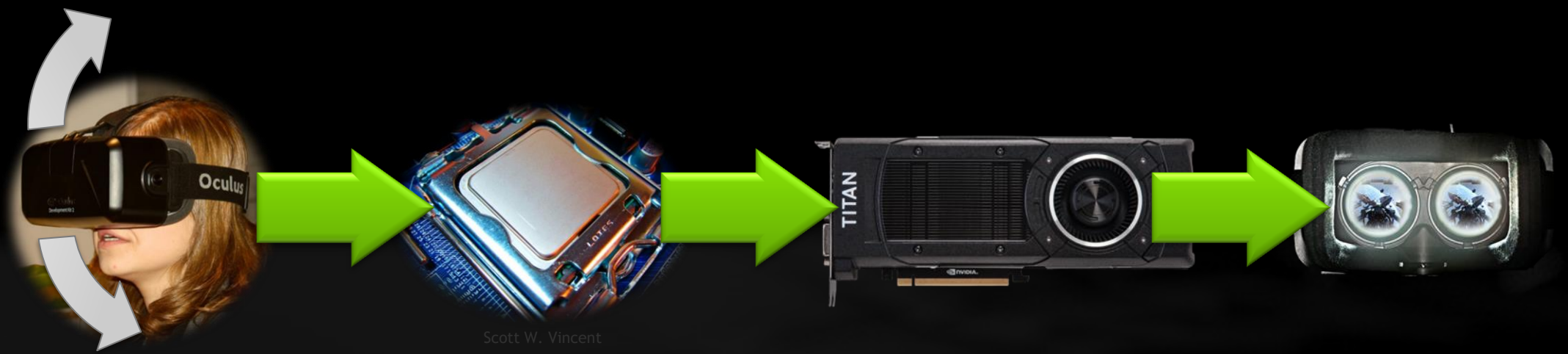
Headset design

Input

Rendering performance

Experience design

Latency

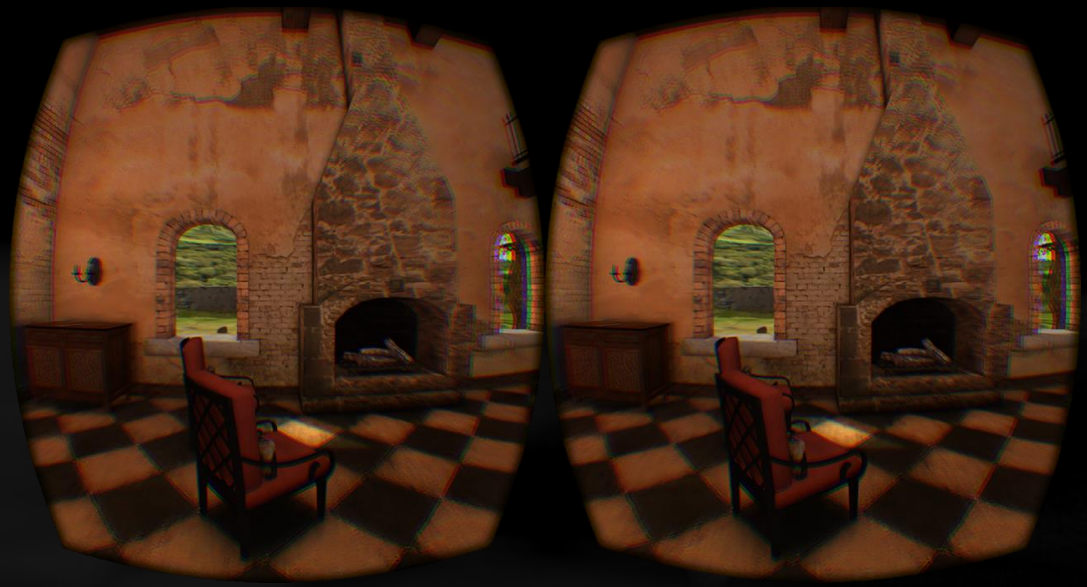


Franklin Heijnen

Scott W. Vincent

Motion to photons in ≤ 20 ms

Stereo Rendering



Two eyes, same scene

What Is VR Direct?

- Various NV hardware & software technologies
- Targeted at VR rendering performance
  - Reduce latency
  - Accelerate stereo rendering

VR Direct Components

In This Talk

Asynchronous Timewarp

VR SLI

Latency

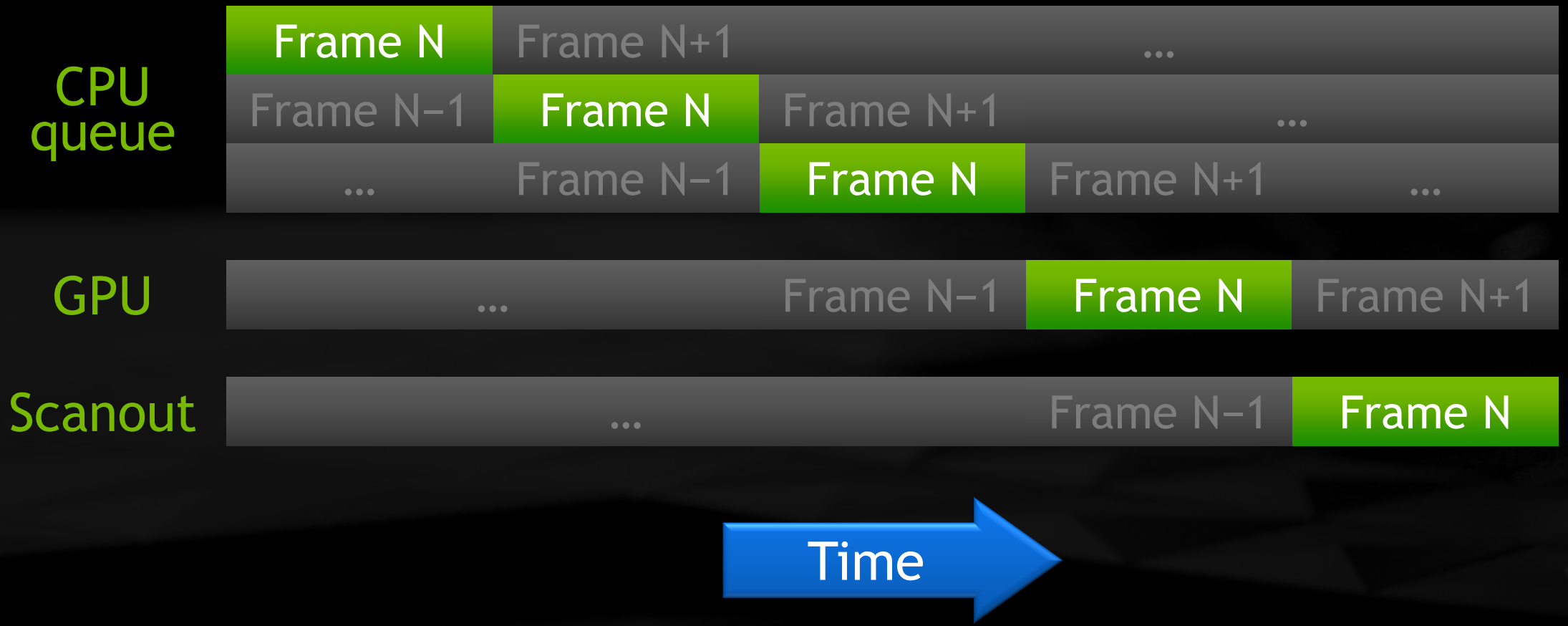
Frame Queuing

Timewarp

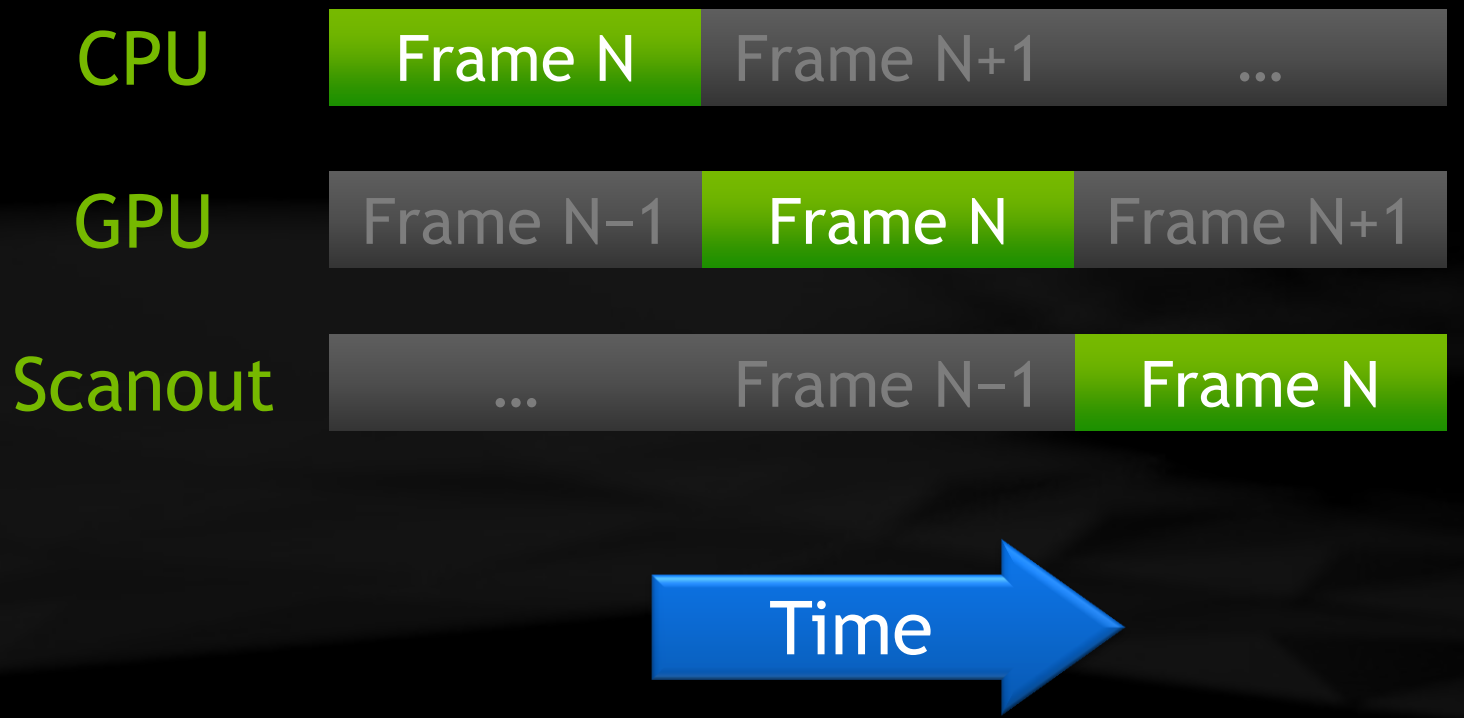
Late-Latching Constants

Asynchronous Timewarp

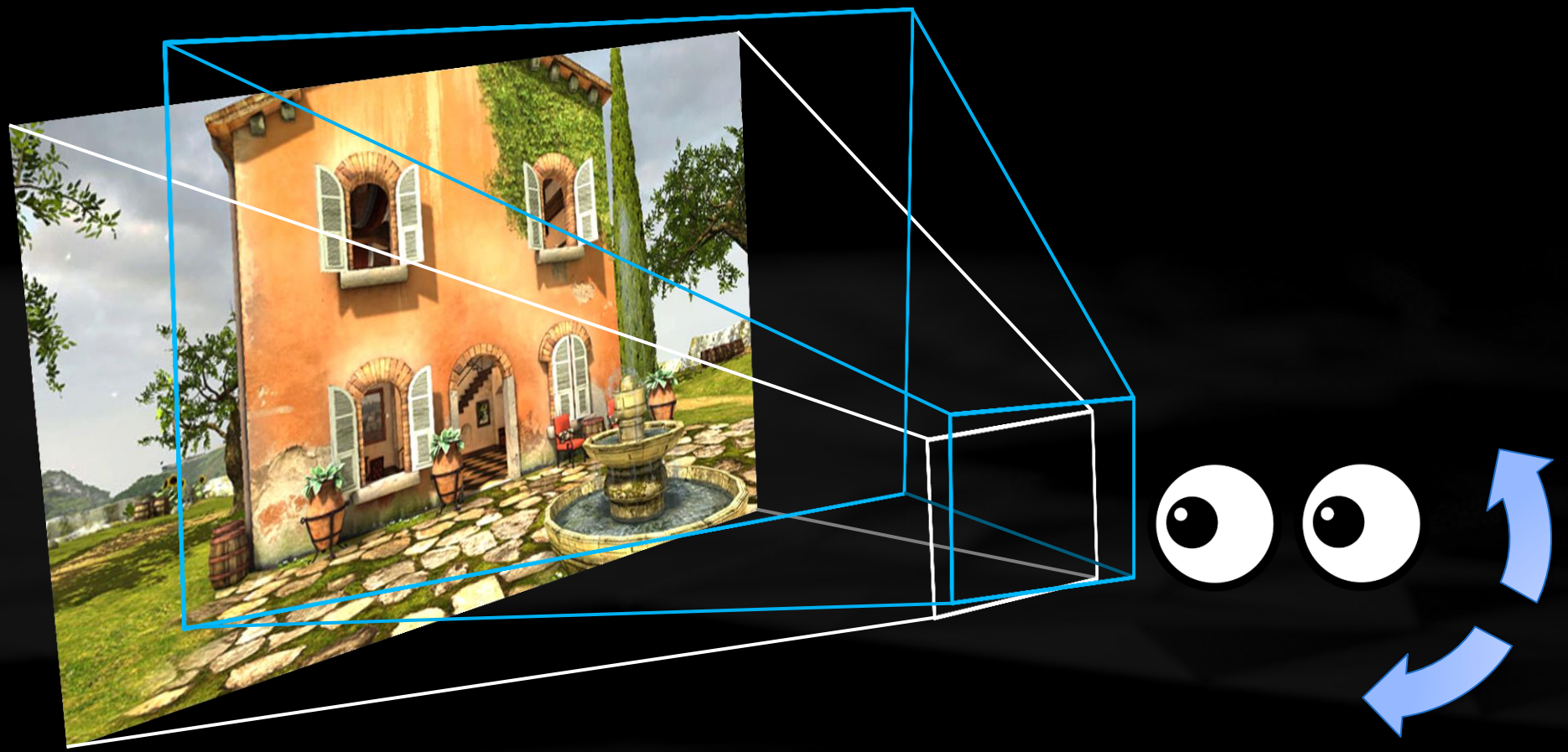
Frame Queuing



Frame Queuing



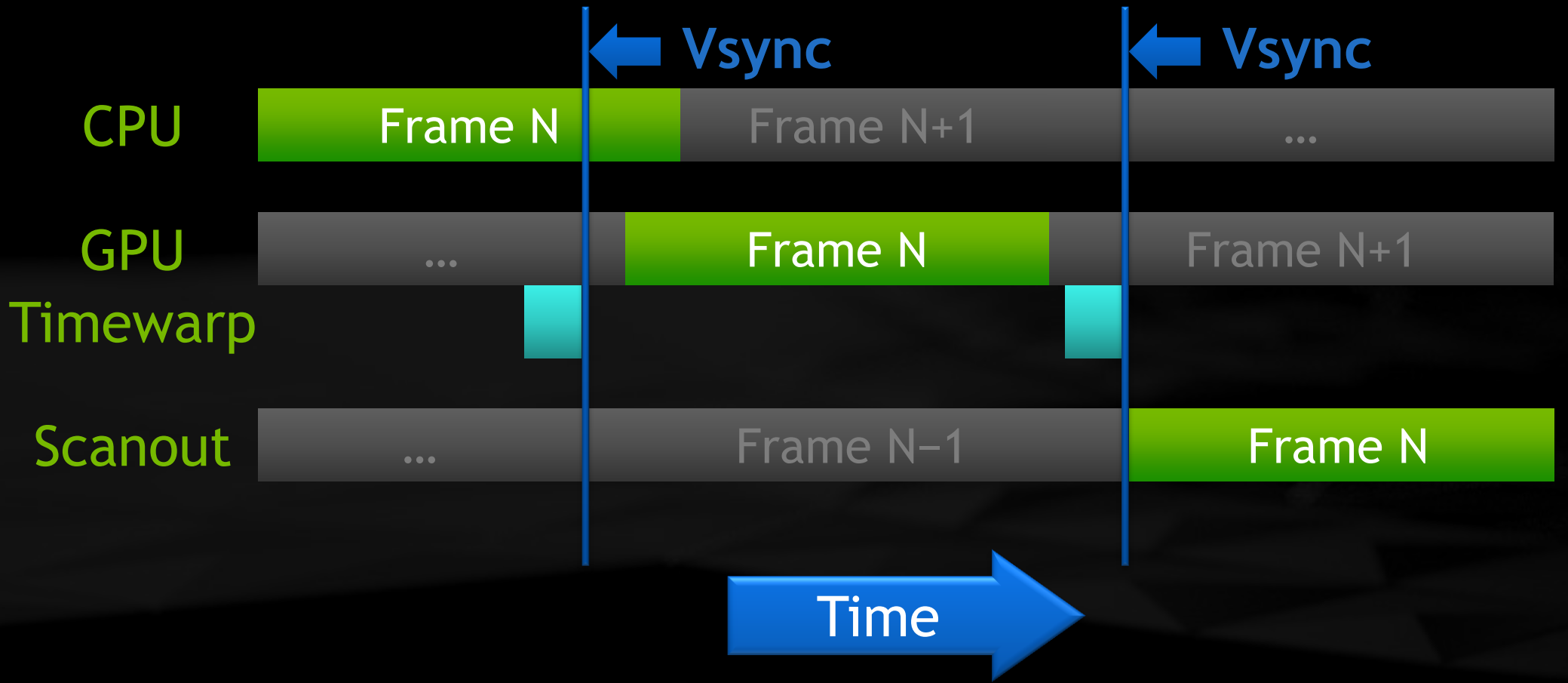
Timewarp



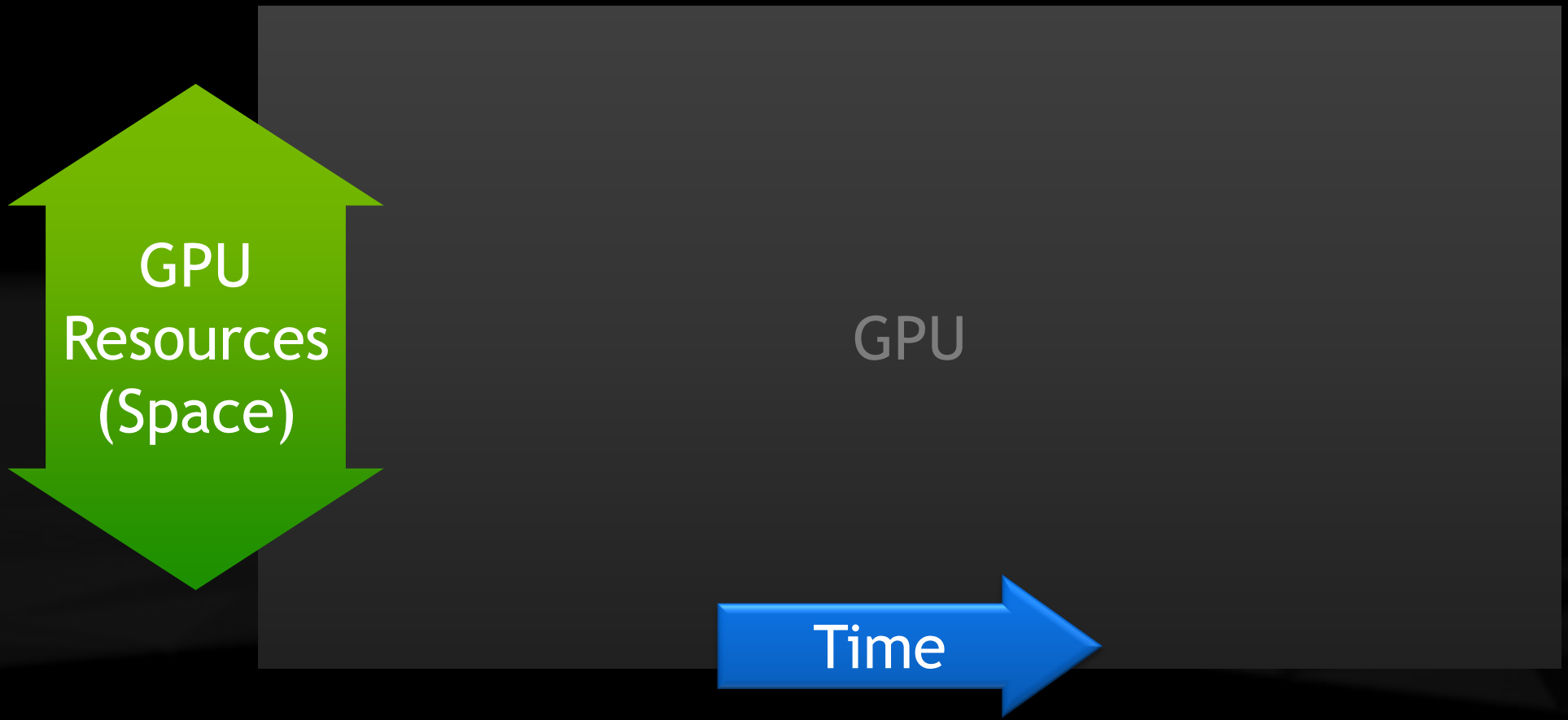
Timewarp Pros & Cons

- Very effective at reducing latency...of rotation!
  - Fortunately, that's the most important
- Doesn't help translation!
- Doesn't help other input latency
- Doesn't help if vsync is missed

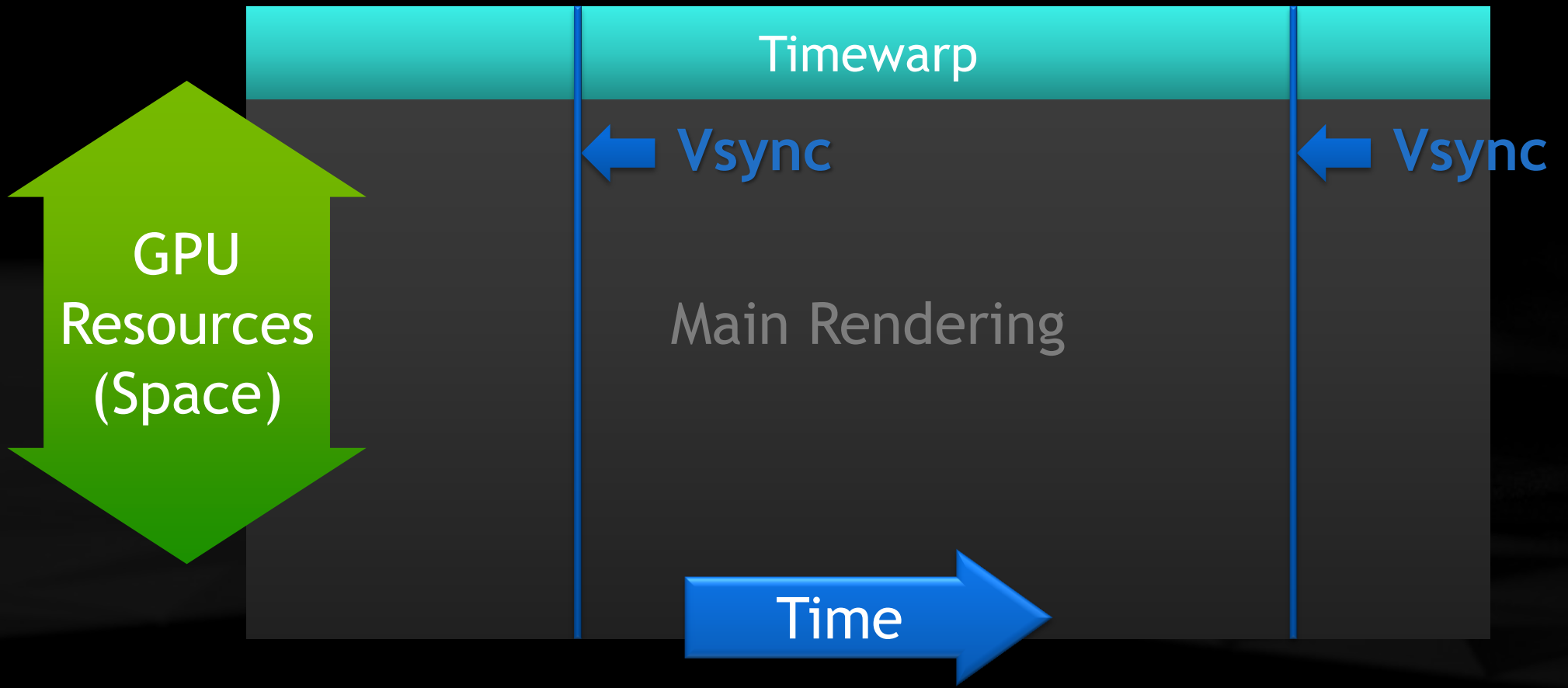
Asynchronous Timewarp



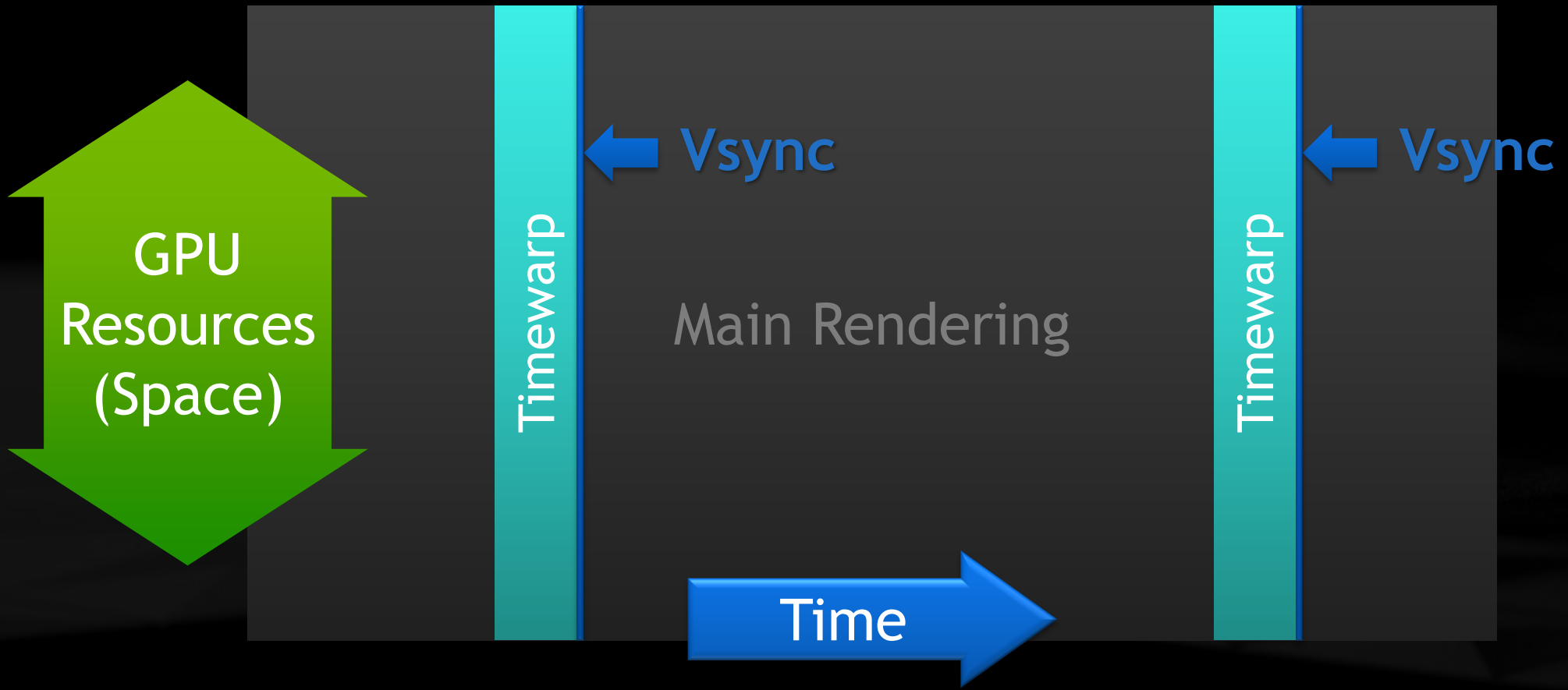
Space Vs Time



Space-Multiplexing



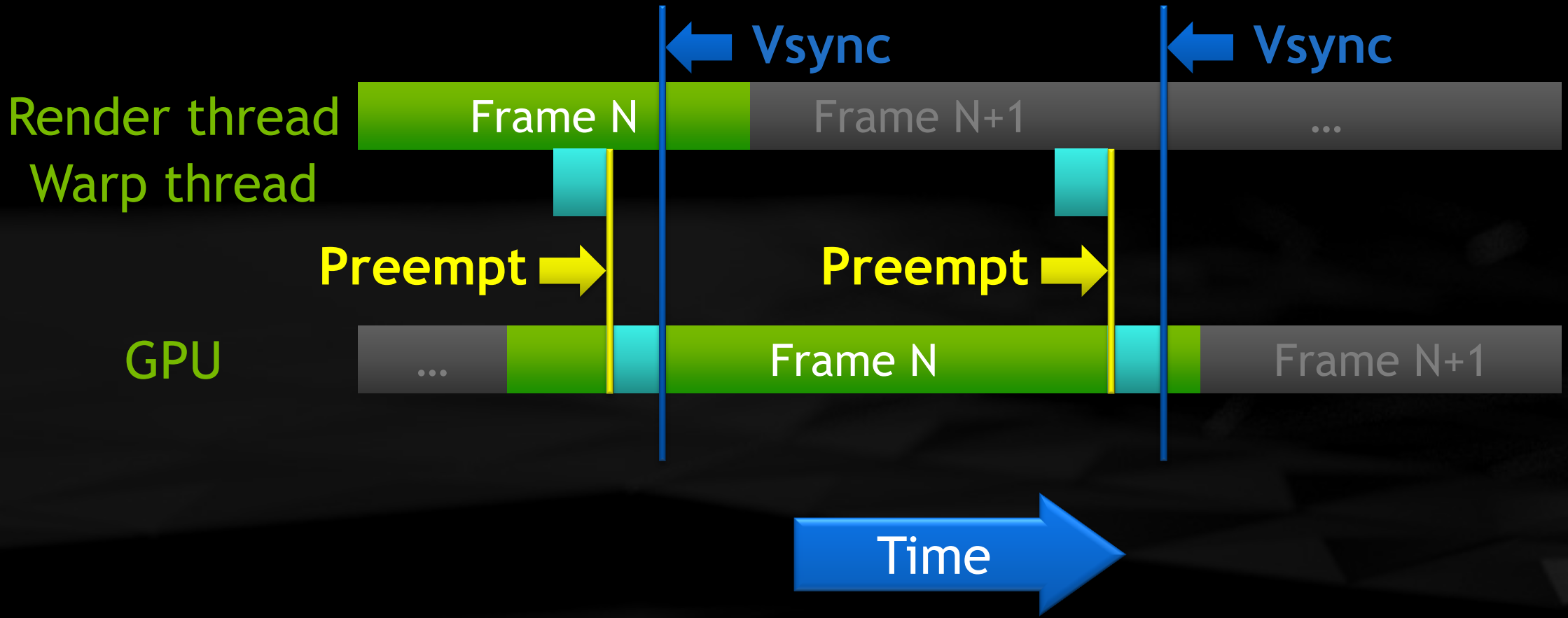
Time-Multiplexing



High-Priority Context

- NV driver supports high-priority graphics context
  - Time-multiplexed — takes over entire GPU
- Main rendering → normal context
- Timewarp rendering → high-pri context

Async Timewarp With High-Pri Context



Preemption

- Fermi, Kepler, Maxwell: draw-level preemption
- Can only switch at draw call boundaries!
  - Long draw will delay context switch
- Future GPU: finer-grained preemption

Direct3D High-Priority Context

- NvAPI_D3D1x_HintCreateLowLatencyDevice()
- Applies to next D3D device created
- Fermi, Kepler, Maxwell / Windows Vista+
- NDA developer driver available now

OpenGL High-Priority Context

- EGL_IMG_context_priority
- Adds priority attribute to eglCreateContext
- Available on Tegra K1, X1
  - Including SHIELD console
- Only for EGL (Android) at present
  - WGL (Windows), GLX (Linux) to come

Developer Guidance

- Still try to render at headset native framerate!
- Async timewarp is a safety net
  - Hide occasional hitches / perf drops
  - Not for upsampling framerate

Developer Guidance

- Avoid long draw calls
  - Current GPUs only preempt at draw call boundaries
  - Async timewarp can get stuck behind long draws
- Split up draws that take >1 ms or so
  - E.g. heavy postprocessing
  - Split into screen-space tiles

Latency TL;DR

- Reduce queued frames to 1
- Timewarp: adjusts rendered image for late head rotation
- Async timewarp: safety net for missed vsync
- NVIDIA enables async timewarp via high-pri context

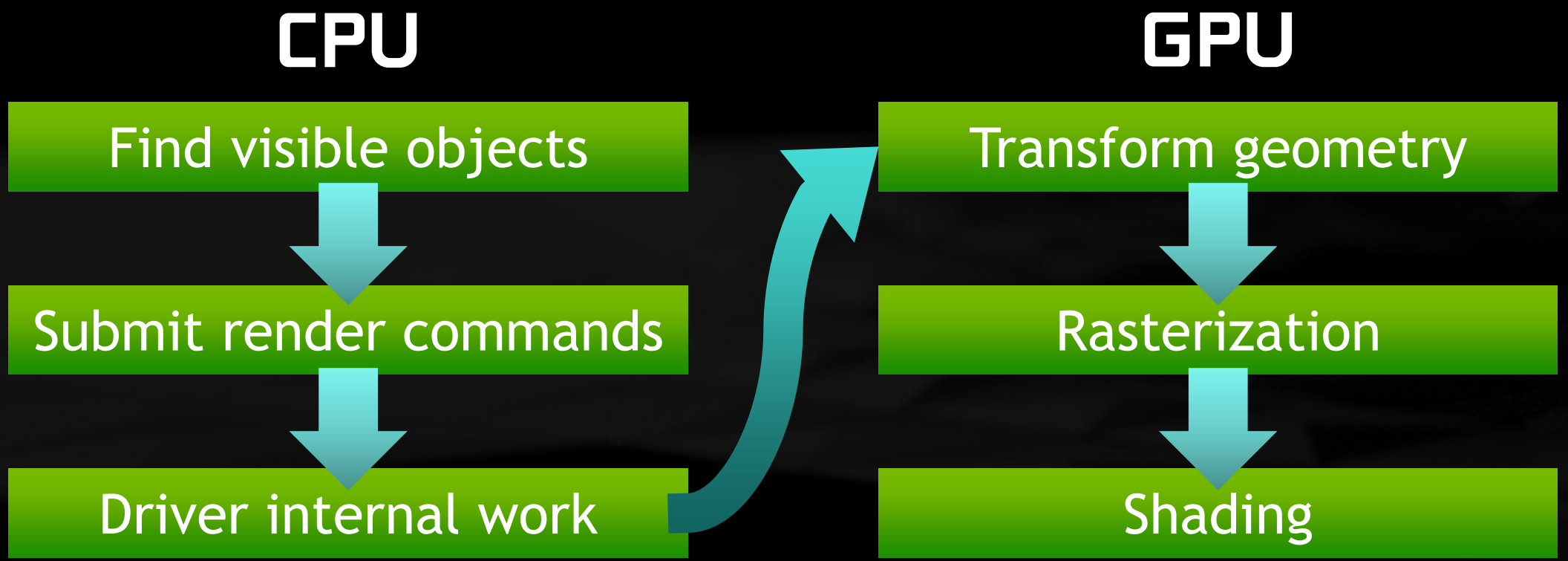
Stereo Rendering

Multiview Rendering

VR SLI

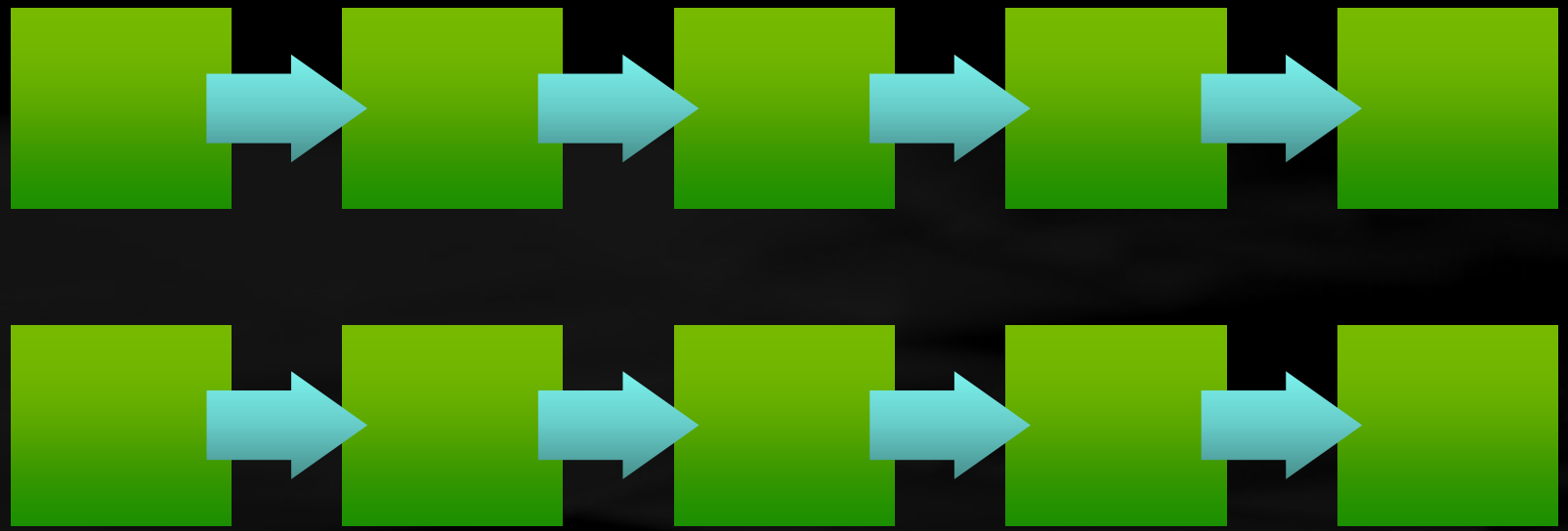
Frame Pipeline

Which stages must be done twice for stereo?



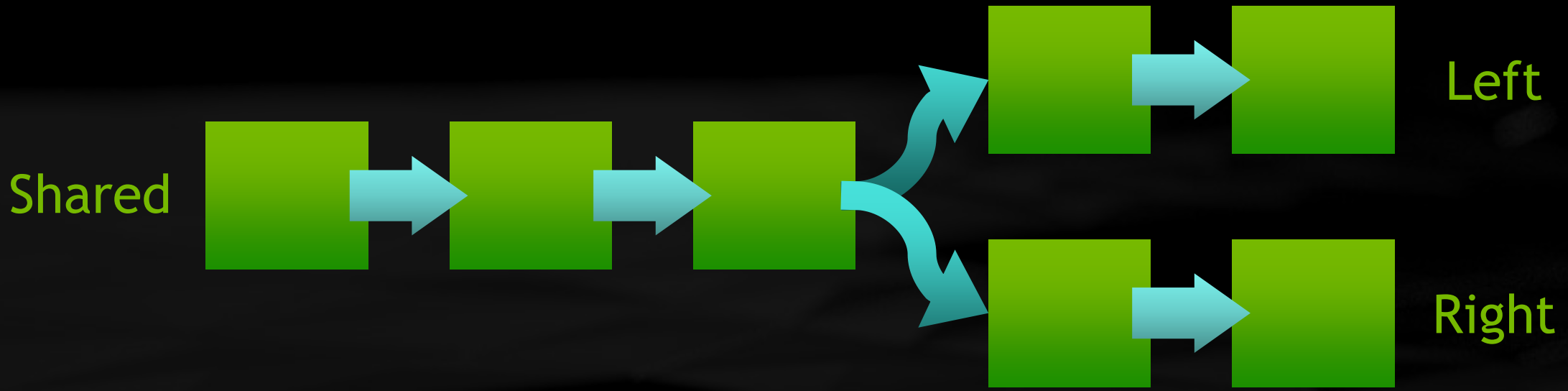
Flexibility vs Optimizability

More flexible – all stages separate



Flexibility vs Optimizability

More optimizable — some stages shared



Stereo Views

- Almost the same visible objects
- Almost the same render commands
- Almost the same driver internal work
- Almost the same geometry rendered

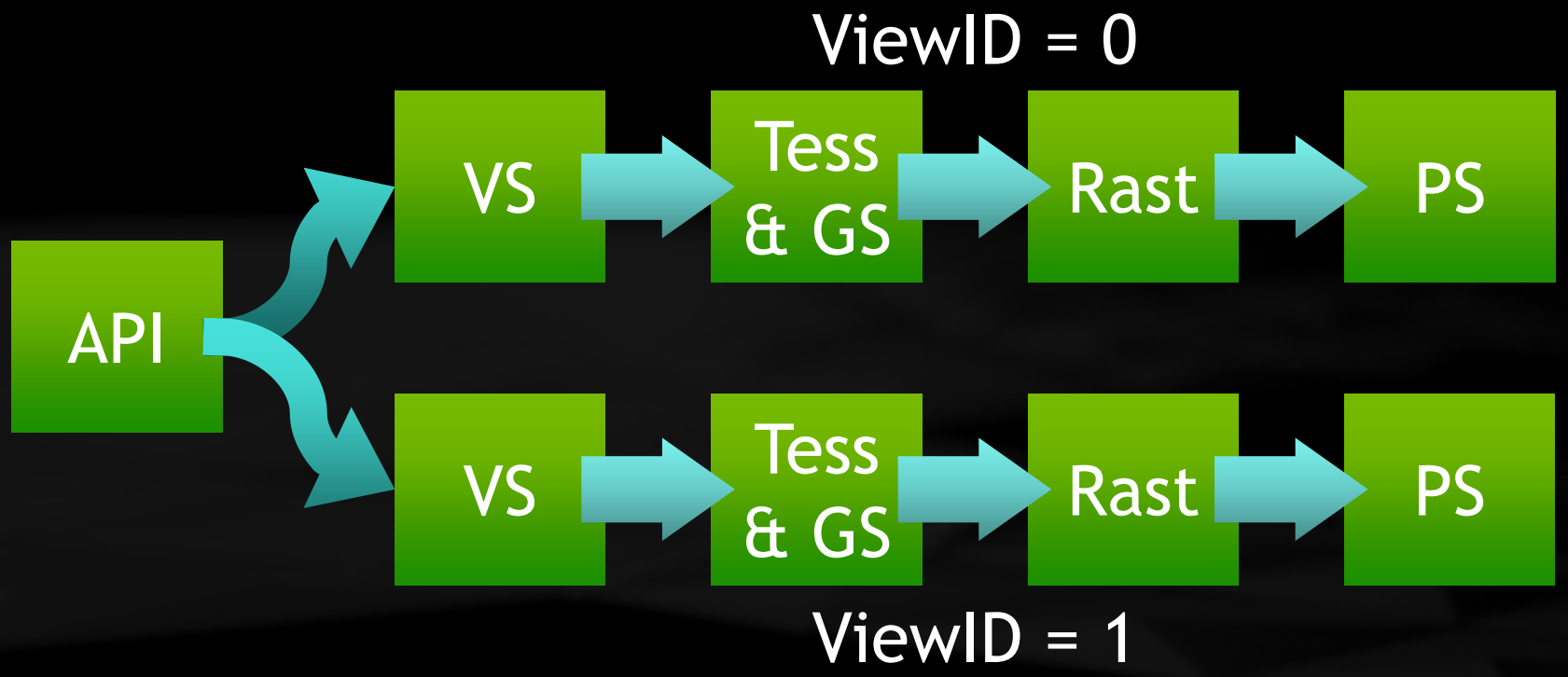
Other Multi-View Scenarios

- Cubemaps: 6 faces
- Shadow maps
  - Several lights in one scene
  - Slices of a cascaded shadow map
- Light probes for GI
  - Many probe positions in one scene

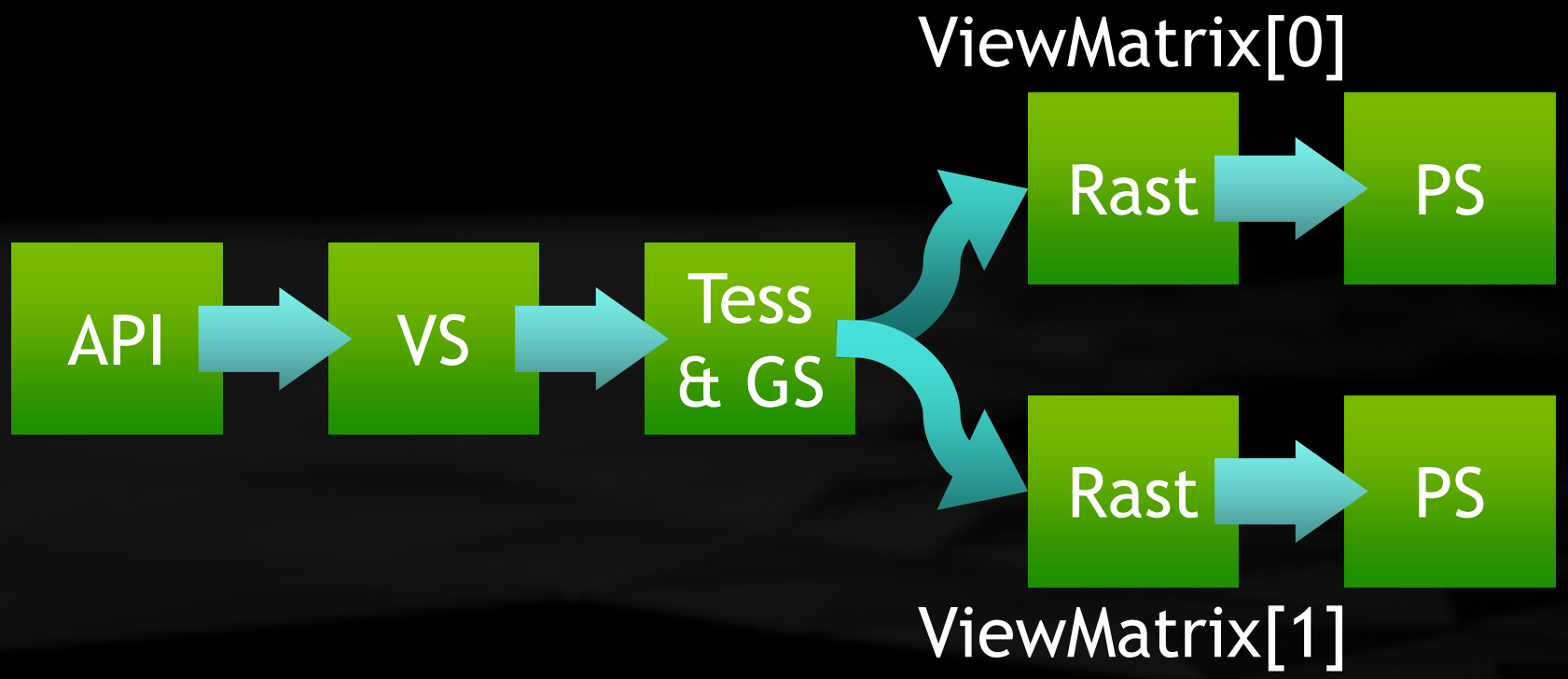
Multiview Rendering

- Submit scene render commands *once*
- All draws, states, etc. broadcast to all views
- API support for limited per-view state
- Saves CPU rendering cost
- Maybe GPU too — depending on impl!

Shader Multiview

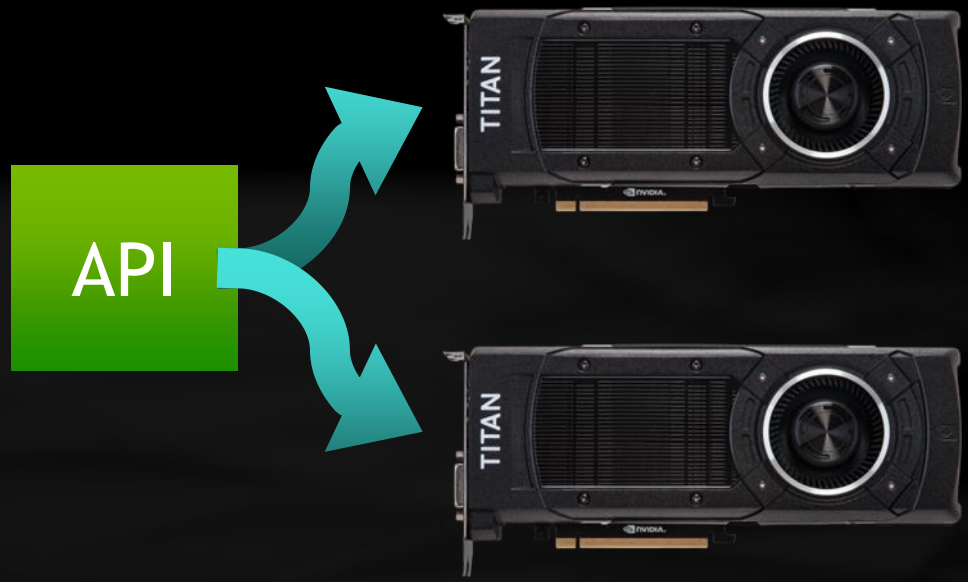


Hardware Multiview



VR SLI

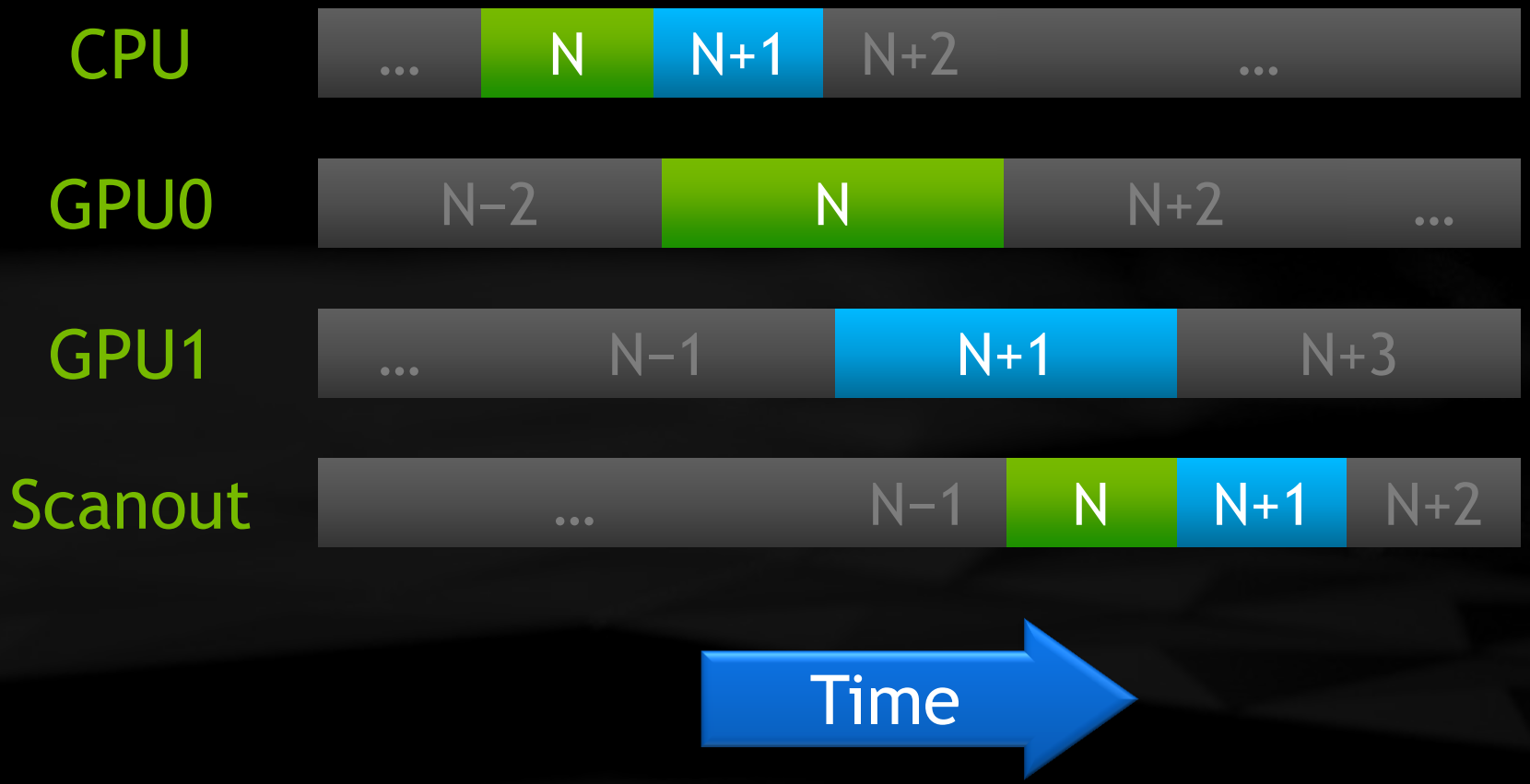
Shared command stream



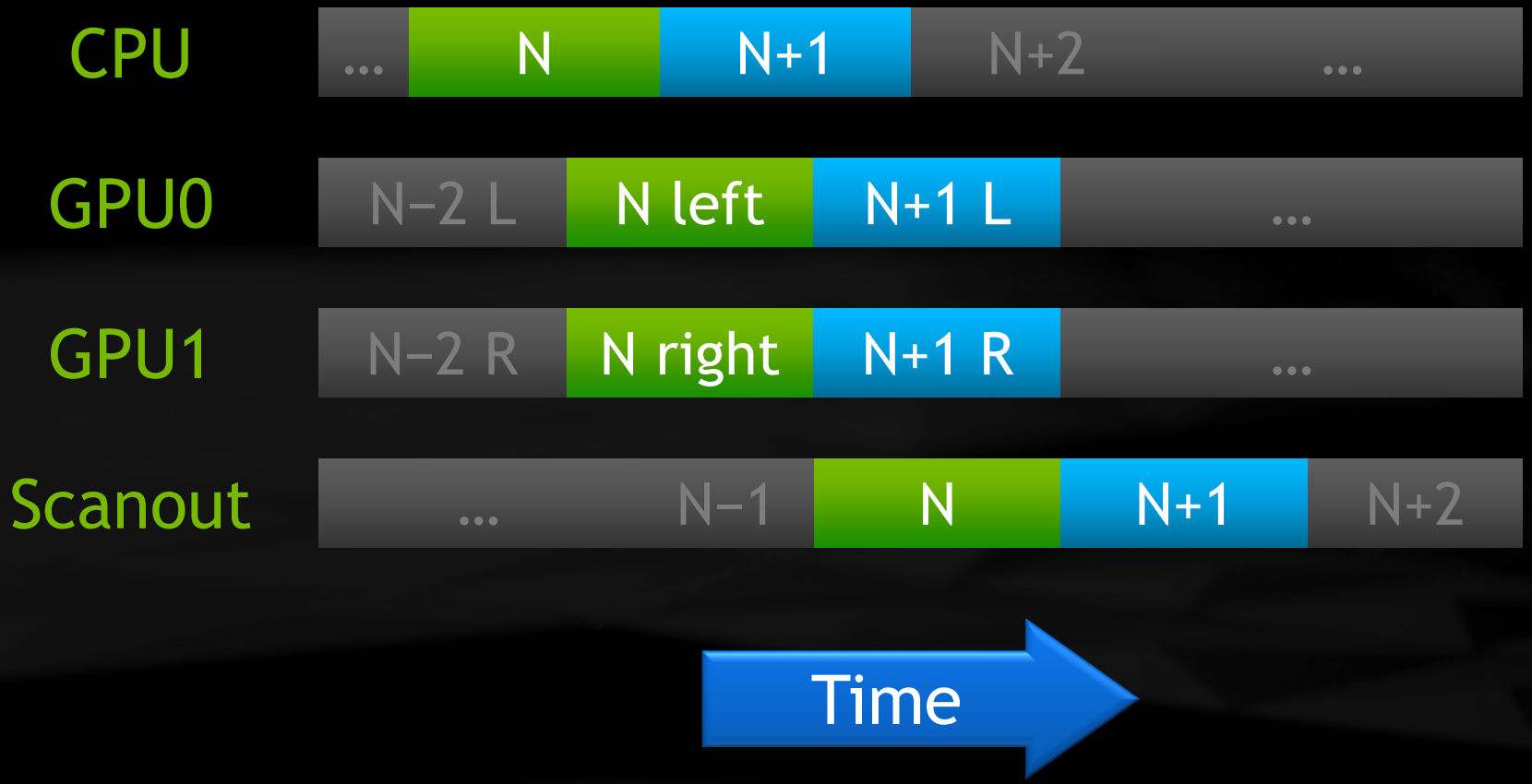
Left

Right

Interlude: AFR SLI



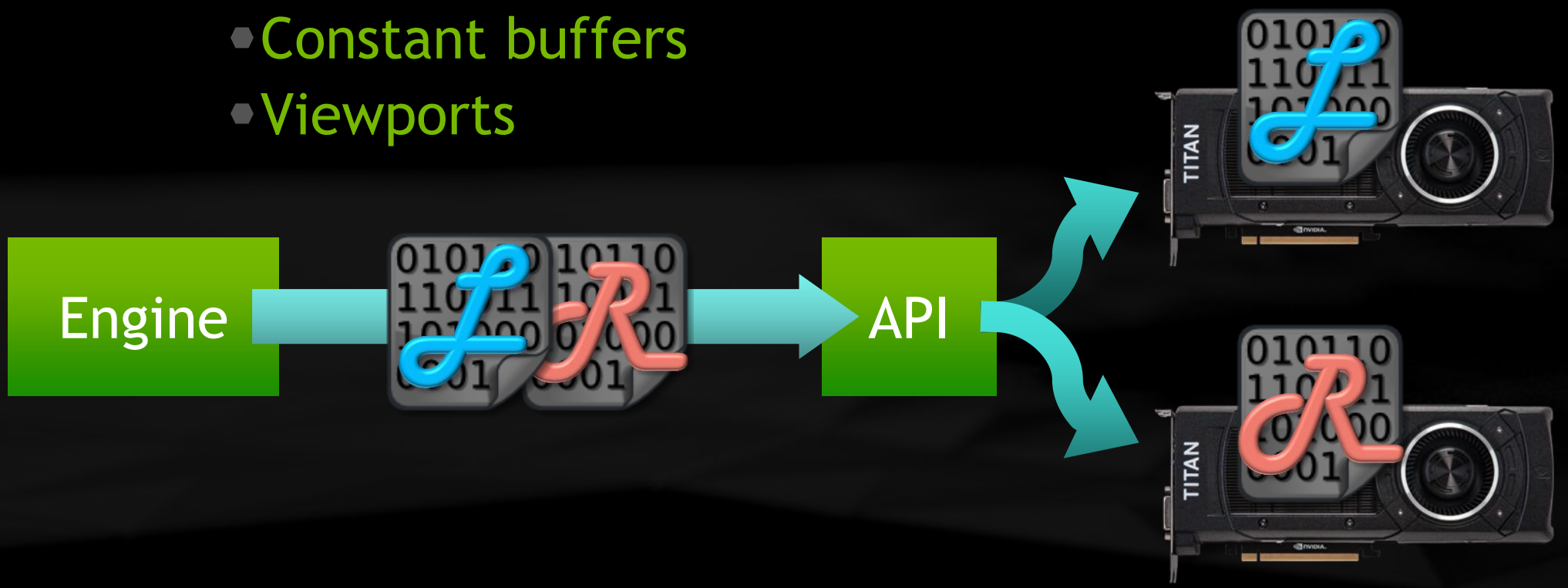
VR SLI


CPU
...
N
N+1
N+2
...
GPU0
N–2 L
N left
N+1 L
...
GPU1
N–2 R
N right
N+1 R
...
Scanout
...
N–1
N
N+1
N+2
Time

VR SLI

Per-GPU state:

- Constant buffers
- Viewports



VR SLI

Blit GPU1→GPU0 over PCIe bus

VR SLI Scaling

- View-independent work (e.g. shadow maps) is duplicated
- Scaling depends on proportion of view-dependent work

API Availability

- Currently D3D11 only
- Fermi, Kepler, Maxwell / Windows 7+
- Developer driver available now
- OpenGL and other APIs: to come

Developer Guidance

- Teach your engine the concept of a “multiview set”
  - Related views that will be rendered together
- Currently:

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


Developer Guidance

- Multiview:

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


Developer Guidance

- Keep track of which render targets store stereo data
  - May need to be marked or set up specially
  - Or allocated as a texture array, etc.
- Keep track of sync points
  - Where you need all views finished before continuing
  - May need to blit between GPUs

Stereo Rendering TL;DR

- Multiview: submit scene once, save CPU overhead
  - Requires some engine integration
- Range of possible implementations
  - Trade off flexibility vs optimizability
- VR SLI: a GPU per eye

VR Direct Recap

- Variety of VR-related APIs coming in near future
- Reduce latency
  - Reduced frame queuing
  - Enable async timewarp & other improvements
- Accelerate stereo rendering
  - Multiview APIs
  - VR SLI

VR Direct API Availability

- Fermi, Kepler, Maxwell
- D3D11: context priorities and VR SLI
  - NDA developer driver available now
- Android: EGL_IMG_context_priority
- Other APIs/platforms: to come

What Next?

- All this stuff is hot out of the oven!
- Will need more iterations before it settles
  - See what works, revise APIs as needed
  - Consolidate & standardize across industry

How VR Is Shaping CryEngine

Our VR Challenges

- As **Developer**
  - Focus on results: Best possible VR demo for GDC 2015 (presence, interaction, performance…)
  - Focus on the platform to be shown
  - Short development time

- As **Technology Provider**
  - Solid implementation
  - Multiplatform and support for multiple head set vendors
  - Focus on performance and seamless integration for users

CRYENGINE®

Exploring the key aspects of VR

- Presence
  - Convincing rich environments, 3D audio, etc.
- Interactivity
  - Allow the player to manipulate the world instead of just watching
- Input devices
  - Experimenting withtraditional and next-gen input devices
- Movement
  - Believable, stable, non-sickening

Our Rendering Challenges

- High & stable frame rate
  - Oculus requires 90+ FPS (drops → physical discomfort)
- Resolution: Full HD and beyond
- Quality: Bringing our signature visuals to VR
- Dual rendering vs Reprojection
- Minimum latency

VR Demo Rendering Tale

REPROJECTION

GPU

Single GPU solution – 90+ FPS at full HD

✓ 3D TV

Excellent performance, just a post-process
Worked well on 3D TVs

X VR (R&D)

Introduces artefacts
Reduced depth perception (to minimize artefacts)
→ “Presence” is not fully achieved

VR Demo Rendering Tale

REPROJECTION

DUAL RENDERING

MGPU

GPU GPU

Single GPU (brute force)

GPU

CPU, GPU: 2x Rendering Effort!!

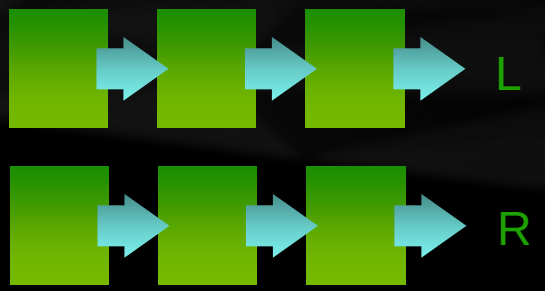
✓ 3D TV

X VR (R&D)

AFR

VR SLI (engine pipeline)

GPU Masking (good proof of concept)



VR Demo Rendering Tale

REPROJECTION

DUAL RENDERING

MGPU

Single GPU (brute force)

GPU GPU

GPU

CPU, GPU: 2x Rendering Effort!!

3D TV

X VR (R&D)

No driver support

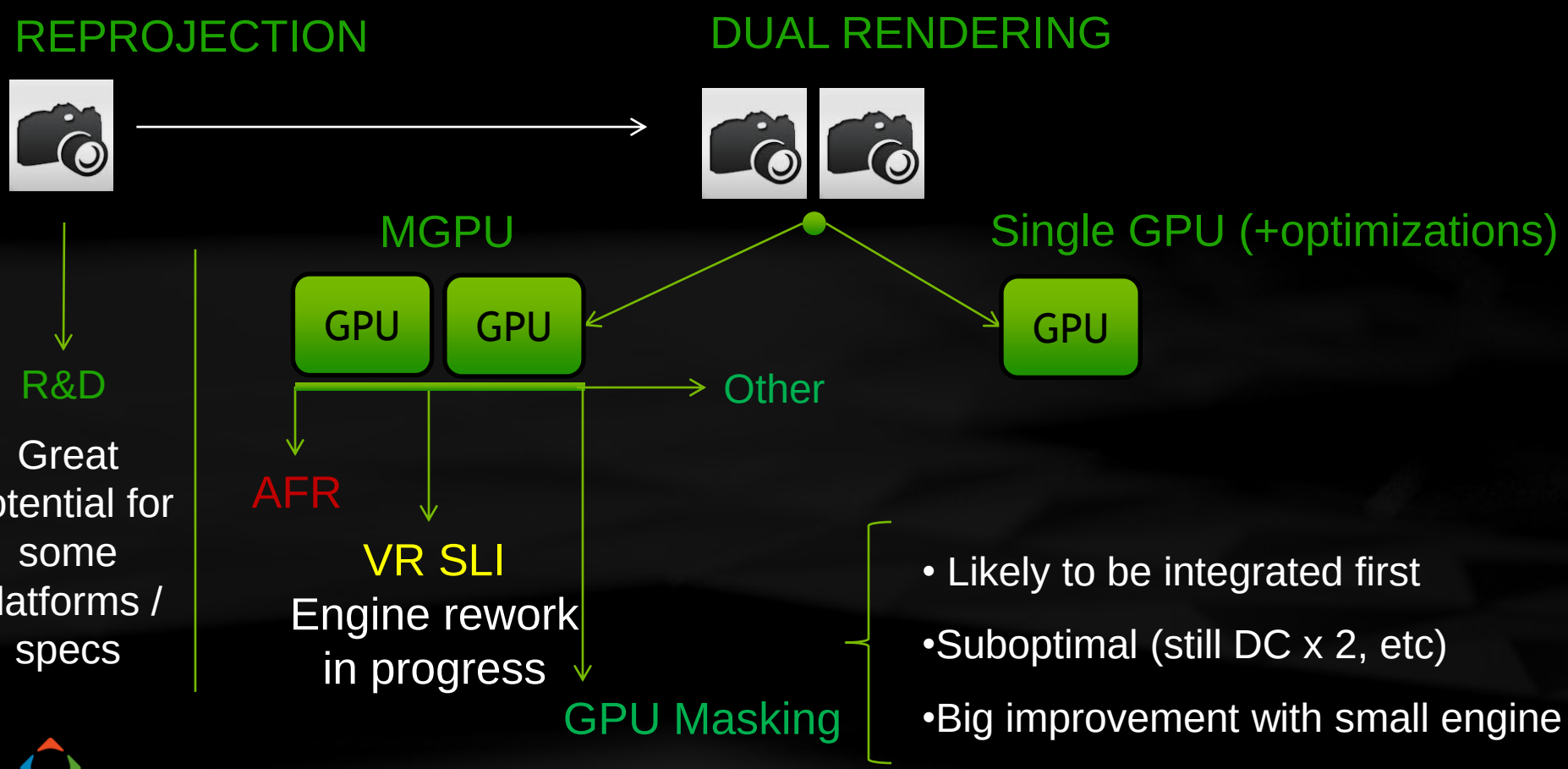
Solution: Use Next-gen GPU

Titan X (**GM200 GPU**)

Engine Optimization
Art optimization

90+ FPS (around 100 FPS avg.)

CryEngine VR Integration



CryEngine VR Integration

REPROJECTION

DUAL RENDERING

R&D

Great potential for some platforms / specs

Not just a tech demo

Community wants to work with these technologies ASAP

Licensees and potential partners' requests are the proof

Questions & Comments?

Email us:

nreed@nvidia.com

dariop@crytek.com

Slides will be posted:

http://gputechconf.com