
RADEON
TECHNOLOGIES GROUP
LIQUIDVR™ TODAY AND TOMORROW
GUENNADI RIGUER, SOFTWARE ARCHITECT
AMD

Bootstrapping the industry for better VR experience

Complimentary to HMD SDKs

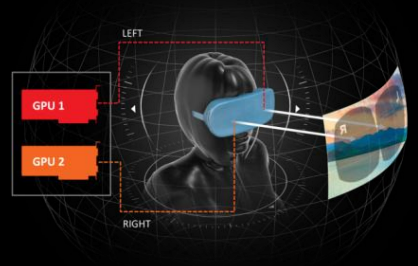
It’s all about giving developers the tools they want!

AMD LIQUIDVR™ FEATURES

Latest data latch

Efficient low-latency GPU head tracking

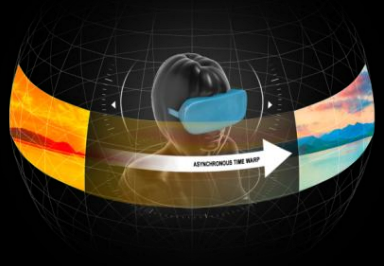
Affinity multi-GPU



Lower latency and increases quality with multiple GPUs

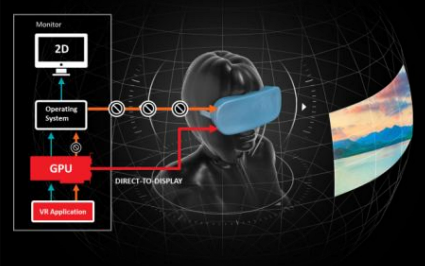
Content developers

Asynchronous shaders



Minimizes latency and stuttering

Direct-to-display



Delivers a seamless plug & play VR experience

HMD vendors

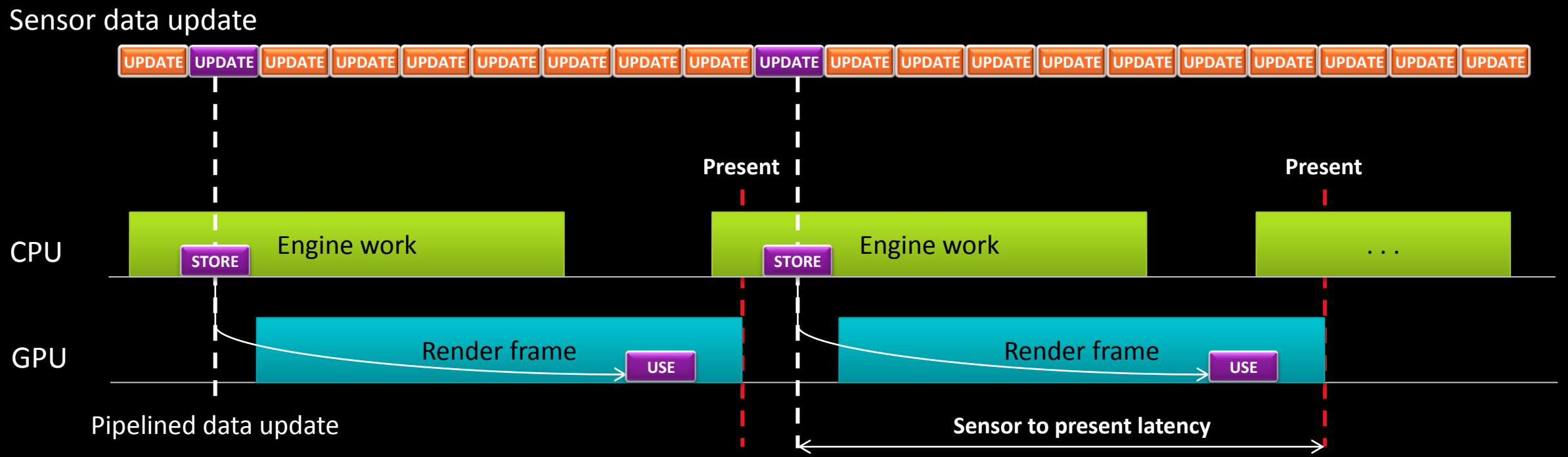
LATEST DATA LATCH

- Using up-to-date head tracking inputs for VR rendering and image warp

- Post-submission data update
  - GPU pull instead of CPU push

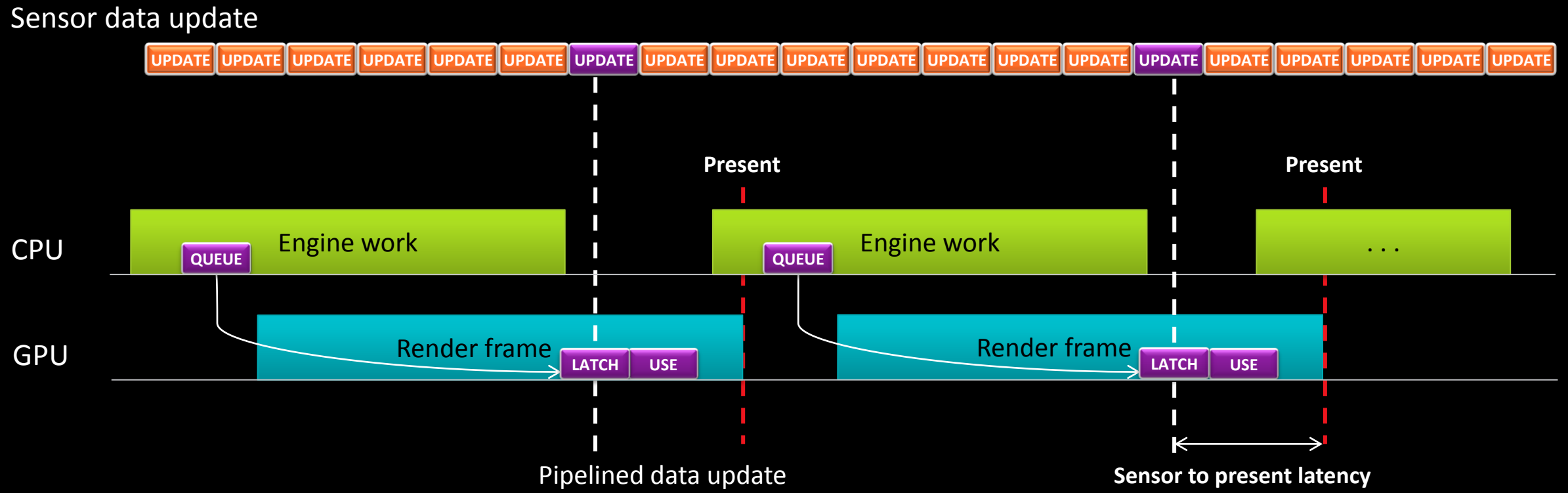
SENSOR DATA WITHOUT LATEST LATCH

- Data pipelined through API at frame building time



LATEST DATA LATCH

- Data latched right before use
- Latch is queued through rendering API



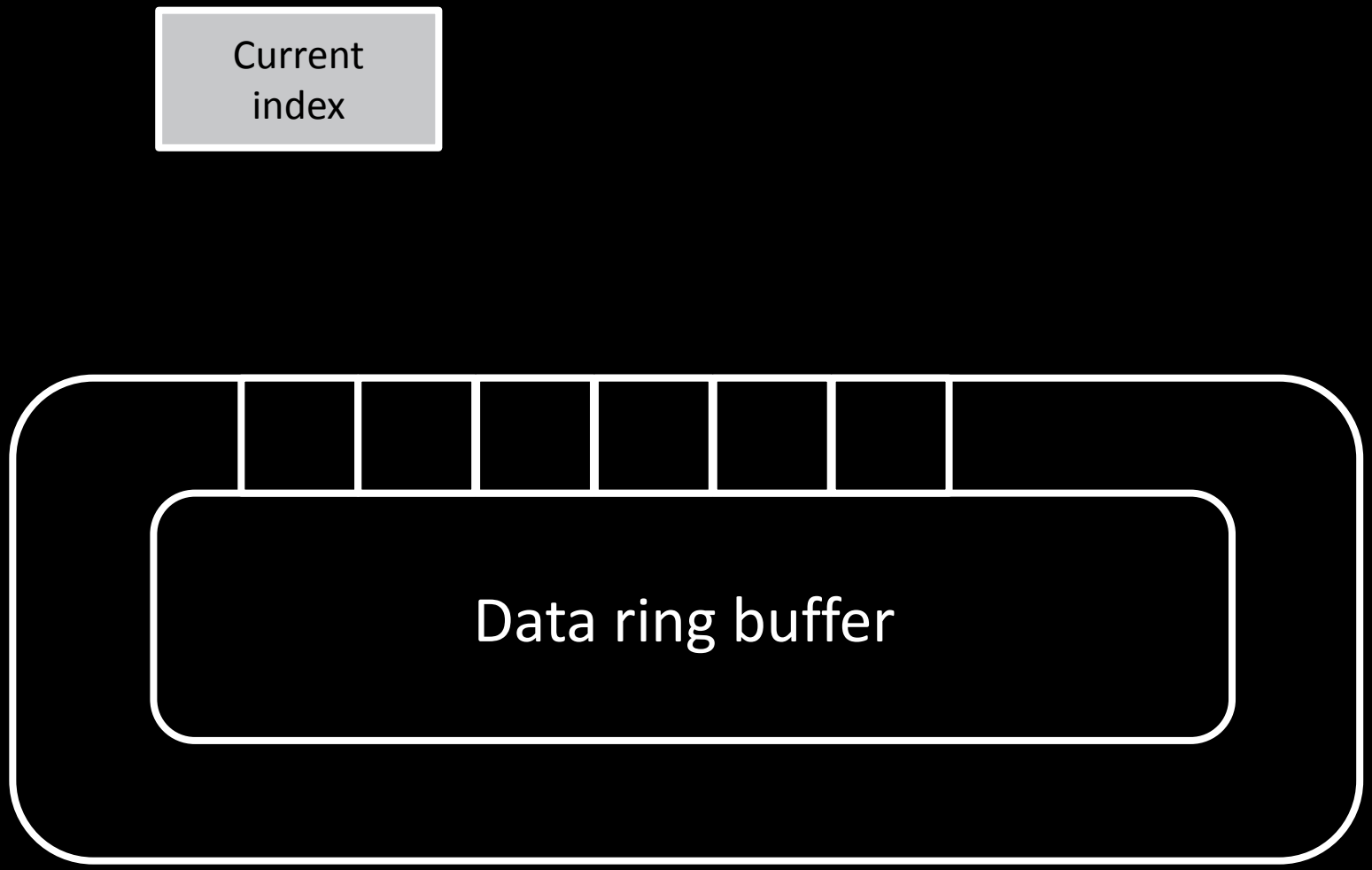
LATE LATCH OPERATION

```
// Setup late latch
ALVRLateLatchConstantBufferD3D11* pLatchBuf;
m_pLvrDevEx->CreateLateLatchConstantBufferD3D11(
    sizeof(HeadData), recordCount, 0, &pLatchBuf);

// Sensor update loop
while (updateSensor)
{
    HeadData latestData;
    GetSensorHeadData(&latestData);
    pLatchBuf->Update(&latestData, 0, sizeof(HeadData));
}
```

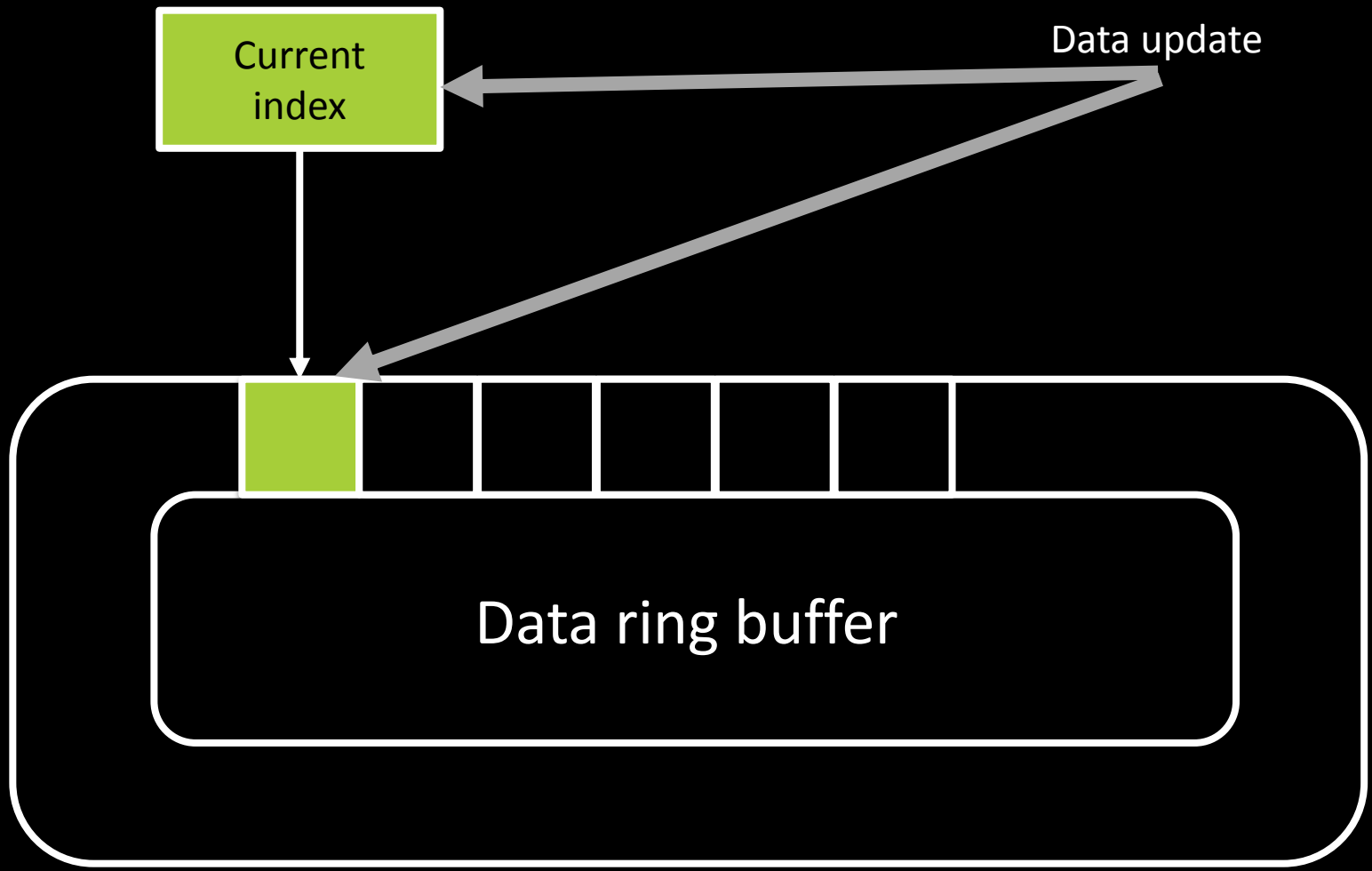
LATE LATCH OPERATION

GPU Queue



LATE LATCH OPERATION

GPU Queue



LATE LATCH OPERATION

GPU Queue

…

Latch

Shader data access

…

```
// Queue latch of latest index
pLatchBuf->QueueLatch();
// Const buffers for shader access of latched data
ID3D11Buffer* pDataCb  = pLatchBuf->GetDataD3D11();
ID3D11Buffer* pIndexCb = pLatchBuf->GetIndexD3D11();
pDxCtx->VSSetConstantBuffers(0, 1, &pIndexCb);
pDxCtx->VSSetConstantBuffers(1, 1, &pDataCb);
. . .
pDxCtx->Draw();
```

LATE LATCH OPERATION

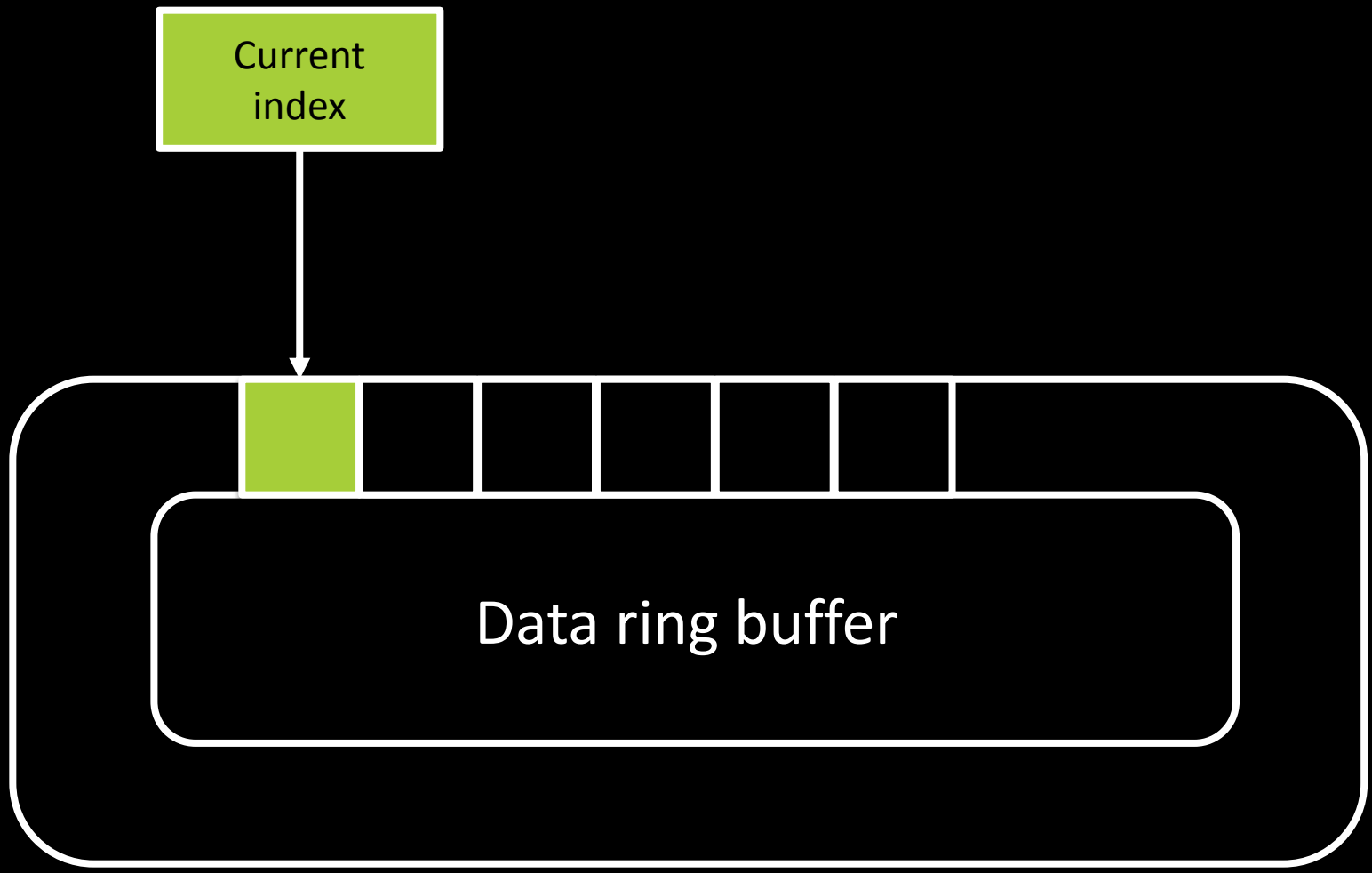
GPU Queue

…

Latch

Shader data access

…



LATE LATCH OPERATION

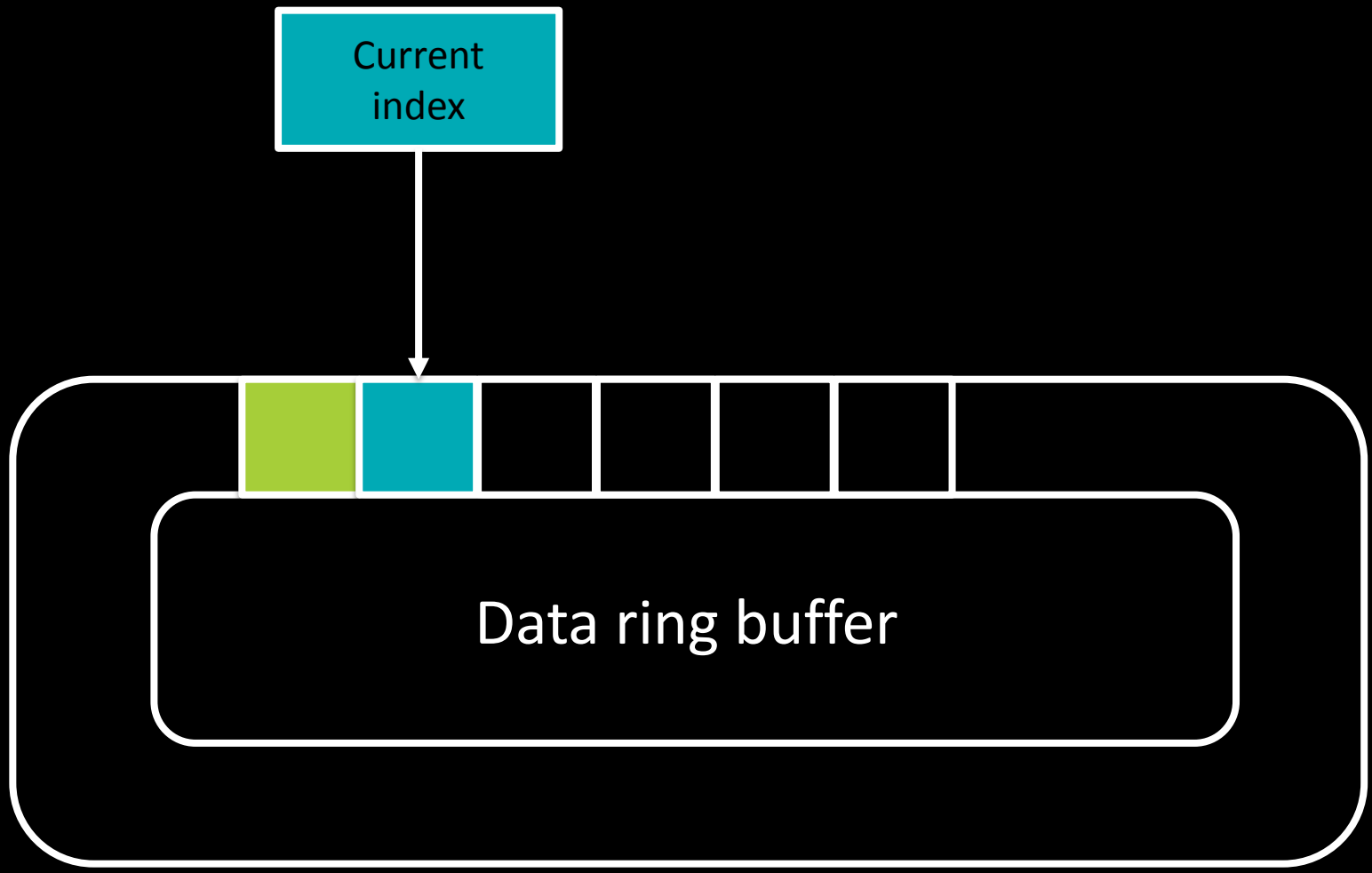
GPU Queue

…

Latch

Shader data access

…



LATE LATCH OPERATION

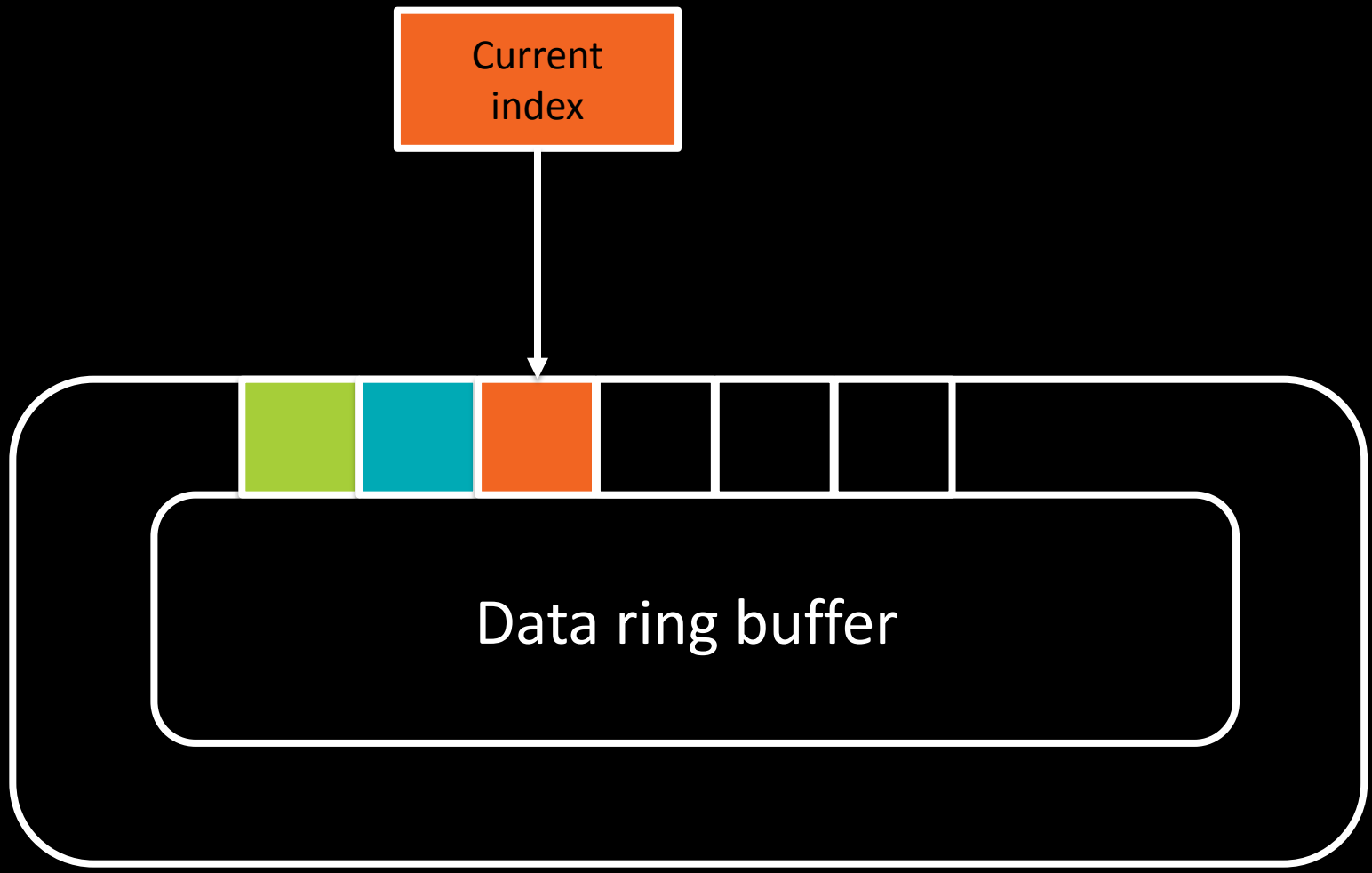
GPU Queue

…

Latch

Shader data access

…



LATE LATCH OPERATION

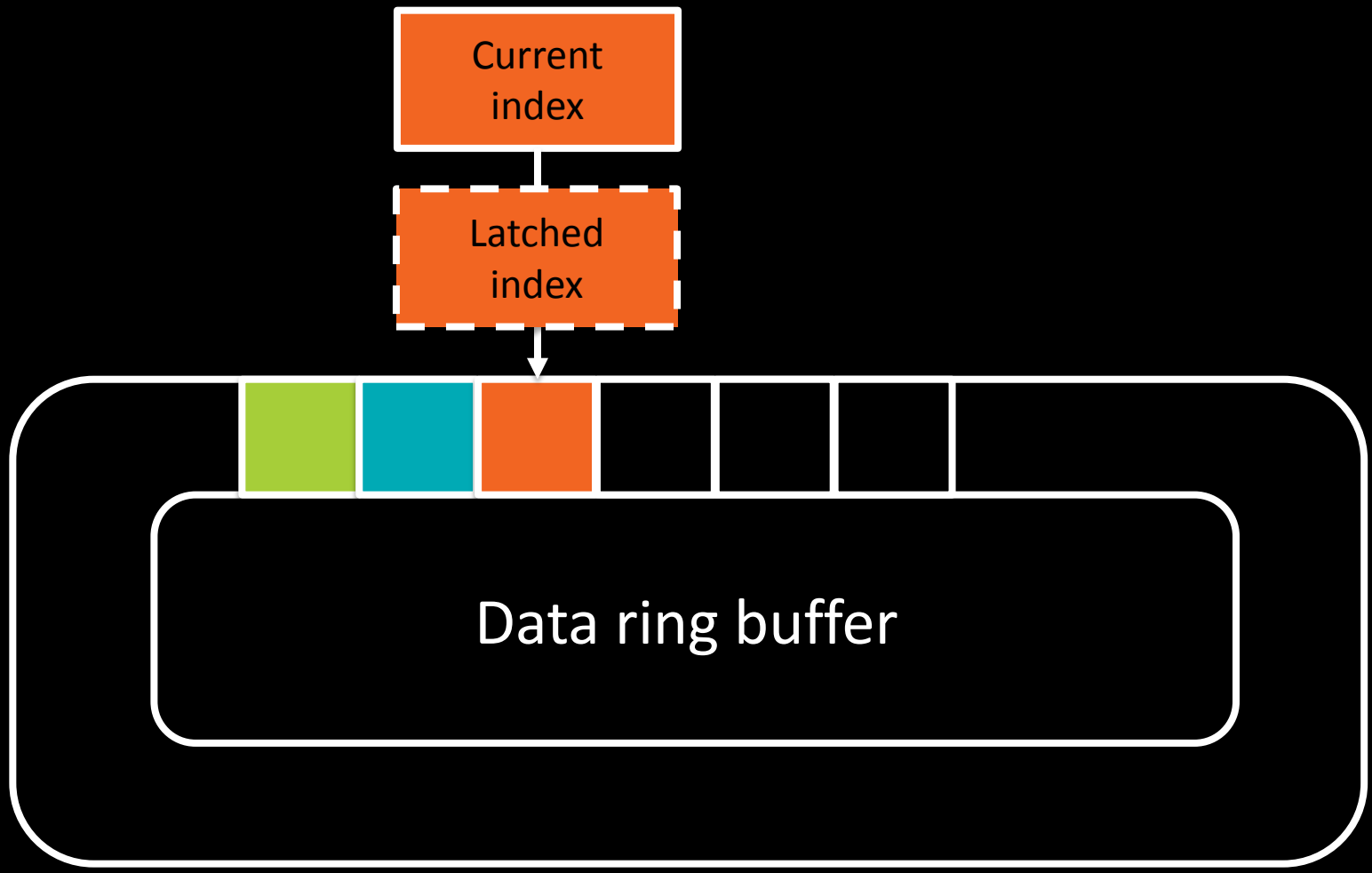
GPU Queue

…

Latch

Shader data access

…



LATE LATCH OPERATION

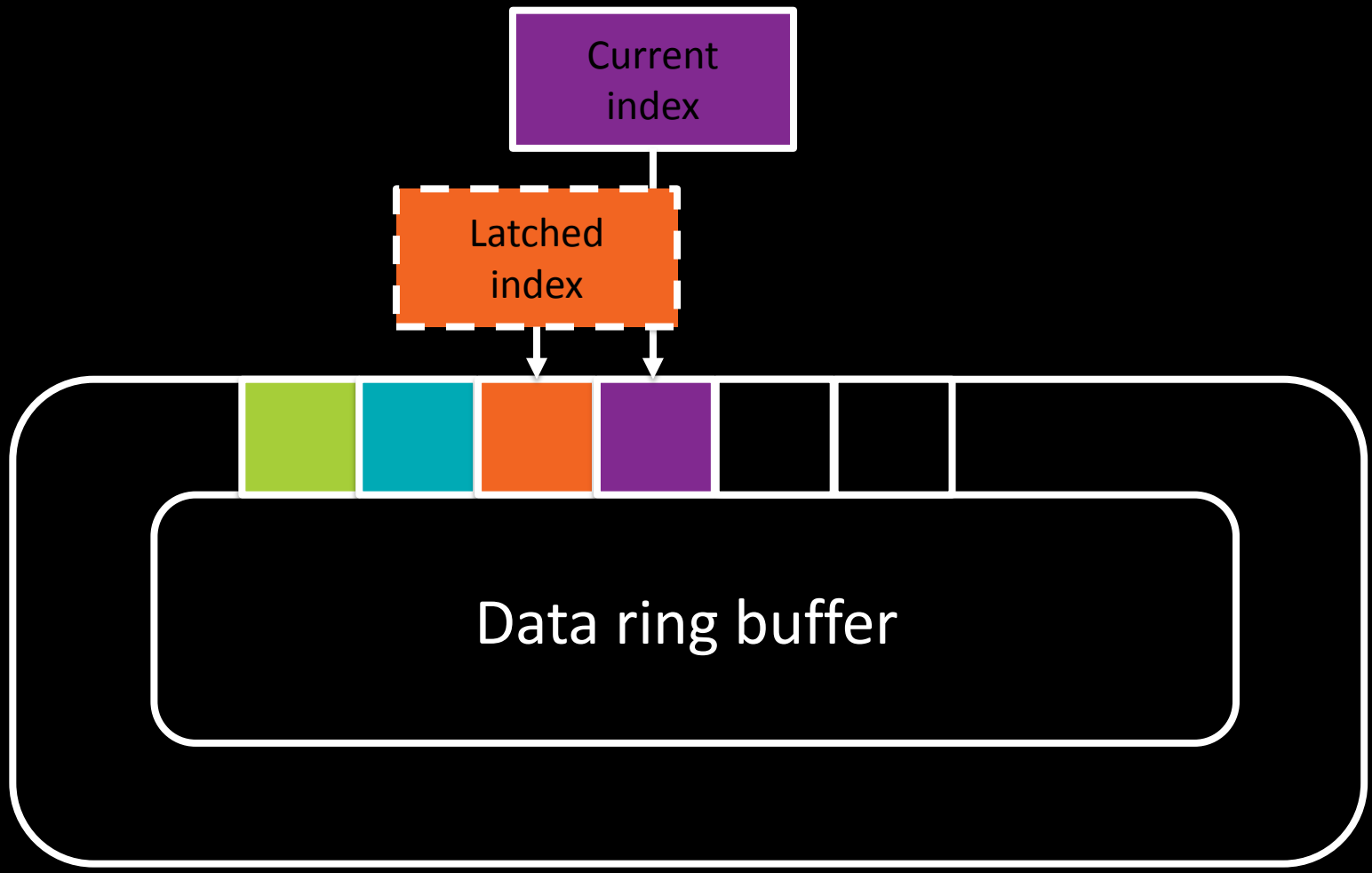
GPU Queue

…

Latch

Shader data access

…



LATE LATCH OPERATION

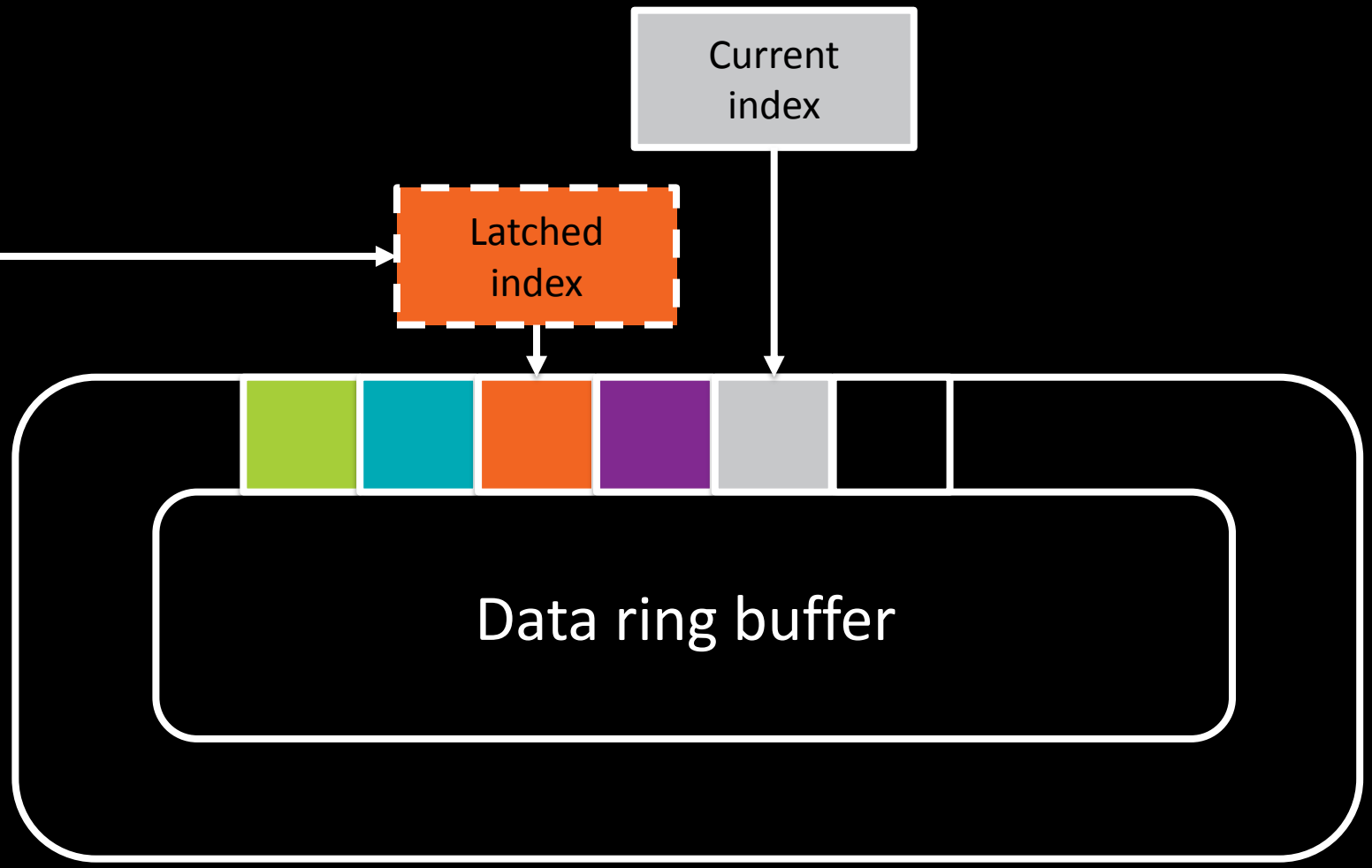
GPU Queue

...

Latch

Shader data access

...

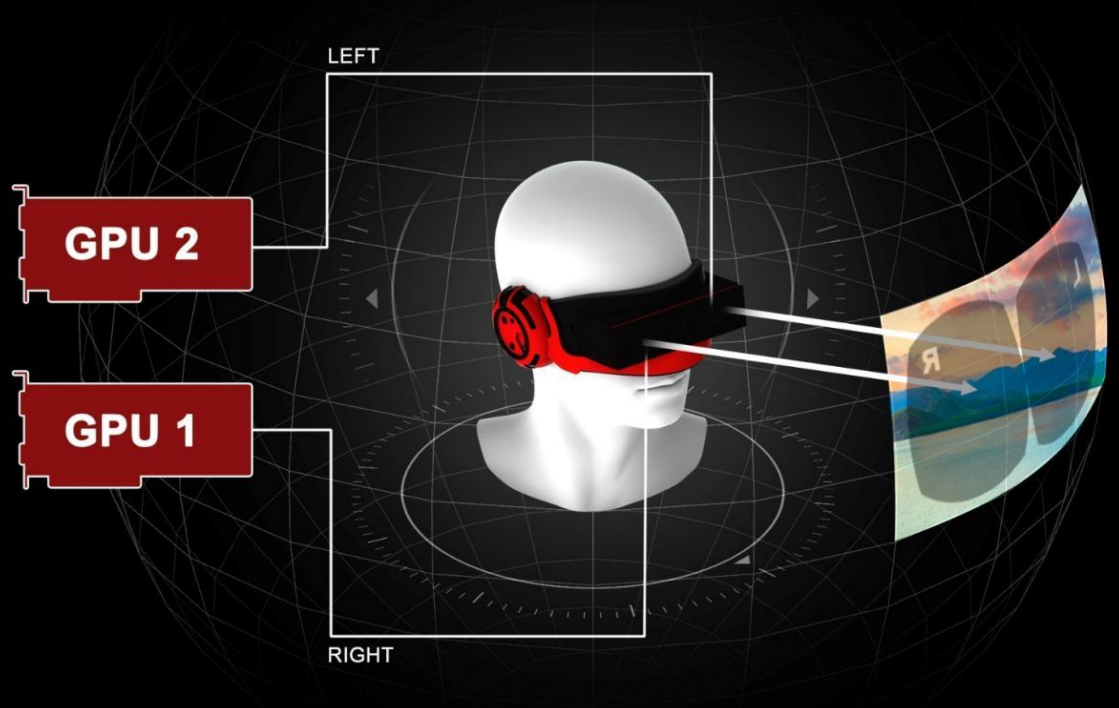


SHADER ACCESS OF LATCHED DATA

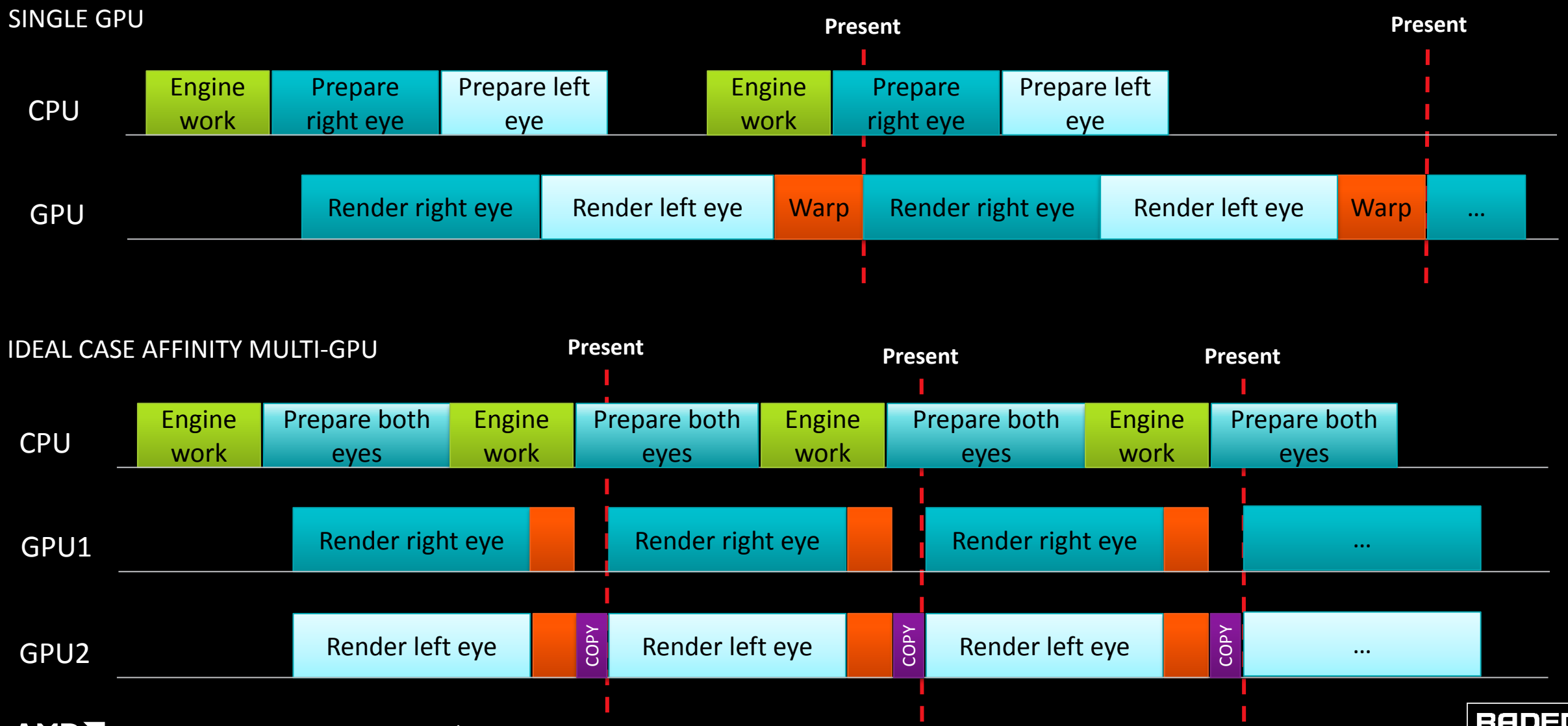
```
struct HeadData
{
    matrix xform;
};
cbuffer LatchIndexBuffer : register(b0)
{
    unsigned int dataIndex;
};
cbuffer LatchDataBuffer : register(b1)
{
    HeadData dataSlots[512];
};
VS_OUTPUT VSMain(VS_INPUT Input)
{
    VS_OUTPUT Output = (VS_OUTPUT)0;
    Output.Pos = mul(Input.Pos, dataSlots[dataIndex].xform);
    return Output;
}
```

AFFINITY MULTI-GPU

- Explicit control of GPUs
  - Broadcast API commands to GPUs
  - Application controls GPU affinity
  - Explicit transfers and synchronization
  - Control of resource instancing
- Assign a GPU per eye for 2 GPUs
- Supports more than 2 GPUs

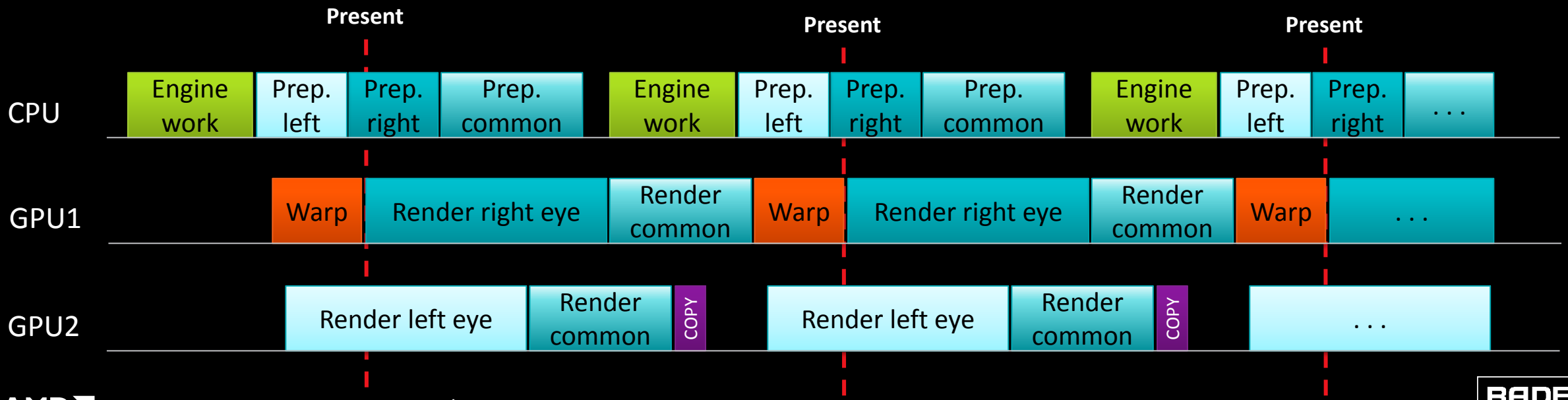


AFFINITY MULTI-GPU IDEAL SCENARIO



MORE TYPICAL MULTI-GPU SCENARIO

- Some inefficiencies due to:
  - Compositor work
  - Eye-independent work
- Illustrates some best practices…



AFFINITY MGPU BEST PRACTICES – DEVICE WRAPPING

- Must wrap device and context
- Can’t use original device and context once wrapped
- Don’t forget to disable prior to creation of non-affinity D3D11 device

AFFINITY MGPU BEST PRACTICES – EYE SWITCHING

- Avoid excessive switching between eyes
- Broadcast to both GPUs when possible

```
for (auto &obj : Scene)
{
    for (int eye = 0; eye < 2; i++)
    {
        pLvrDevEx->SetGpuRenderAffinity(1 << eye);
        pDxCtx->Map(pCb, ...);
        UpdateEyeXform(pCb, eye);
        pDxCtx->Unmap(pCb);
        DrawObject(obj, pCb);
    }
}
```

AFFINITY MGPU BEST PRACTICES – EYE SWITCHING

- Avoid excessive switching between eyes - √
- Broadcast to both GPUs when possible

```
for (int eye = 0; eye < 2; i++)
{
    pLvrDevEx->SetGpuRenderAffinity(1 << eye);
    pDxCtx->Map(pCb, ...);
    UpdateEyeXform(pCb, eye);
    pDxCtx->Unmap(pCb);
    for (auto &obj : Scene)
    {
        DrawObject(obj, pCb);
    }
}
```

AFFINITY MGPU BEST PRACTICES – EYE SWITCHING

- Avoid excessive switching between eyes - √
- Broadcast to both GPUs when possible - √

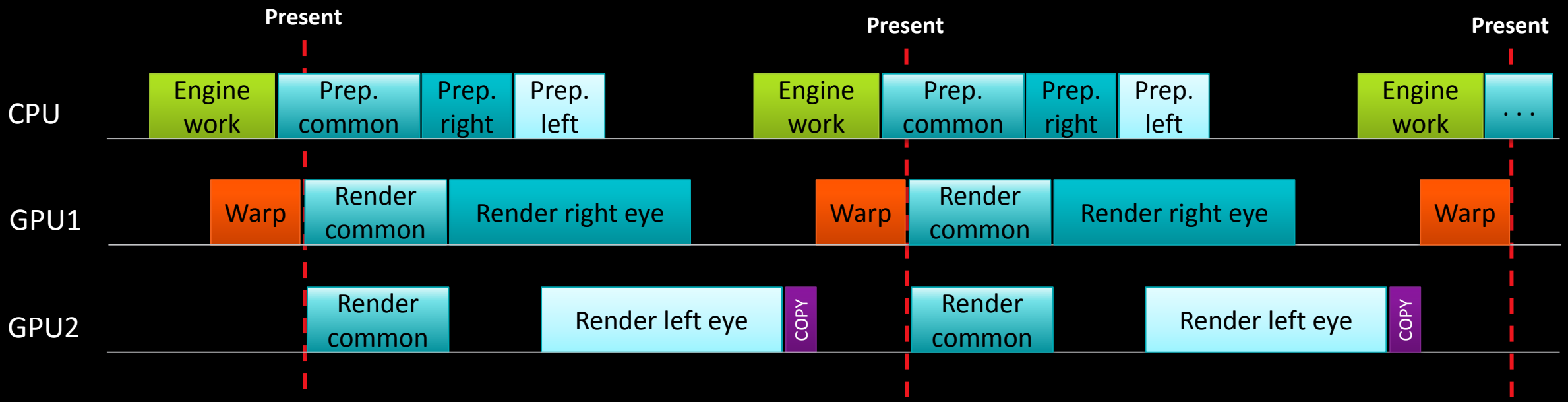
```
for (int eye = 0; eye < 2; i++)
{
    pLvrDevEx->SetGpuRenderAffinity(1 << eye);
    pDxCtx->Map(pCb, ...);
    UpdateEyeXform(pCb, eye);
    pDxCtx->Unmap(pCb);
}
pLvrDevEx->SetGpuRenderAffinity(0x3); // Broadcast to both eyes
for (auto &obj : Scene)
{
    DrawObject(obj, pCb);
}
```

AFFINITY MGPU BEST PRACTICES – CONST BUFFERS

- Minimize const buffers with eye-dependent data
- Minimize size of const buffers
- Max 64K const buffers

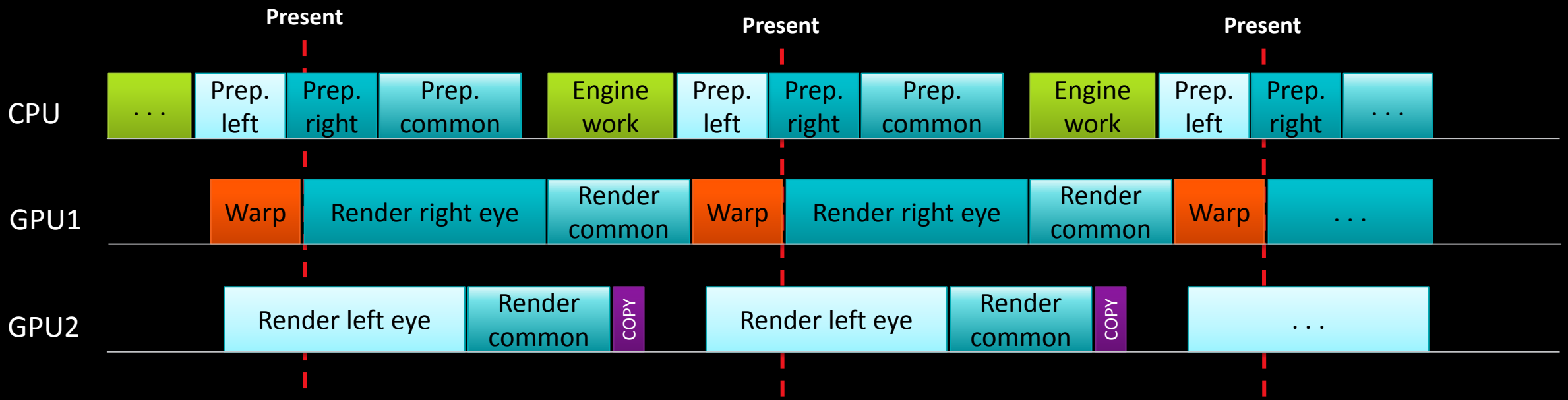
AFFINITY MGPU BEST PRACTICES – WORK ORDERING

- Ordering of operations matters



AFFINITY MGPU BEST PRACTICES – WORK ORDERING

- Push eye-independent work to end of frame
- Get slave running earlier to hide copy

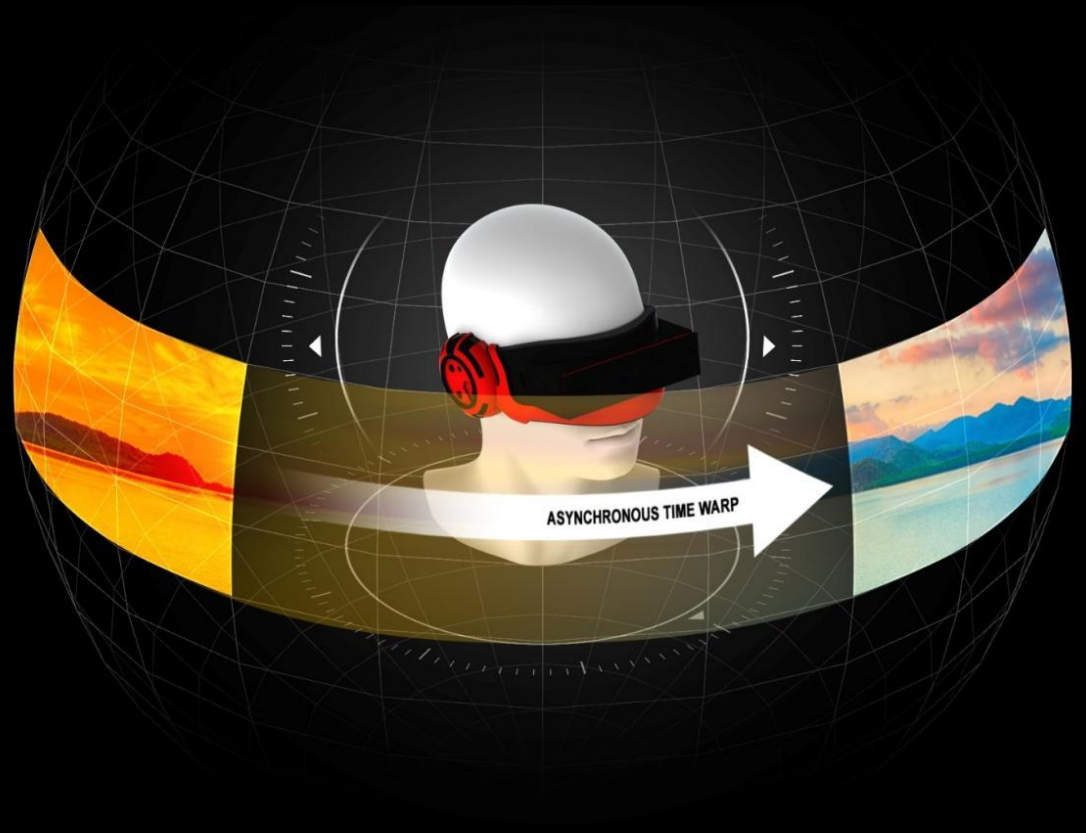


AFFINITY MGPU BEST PRACTICES - TRANSFERS

- Application's use of sync objects is more optimal
- 3D engine transfers are safer choice
  - –Supports partial subresource transfers
- DMA only for experts
  - –Supports whole subresource transfers

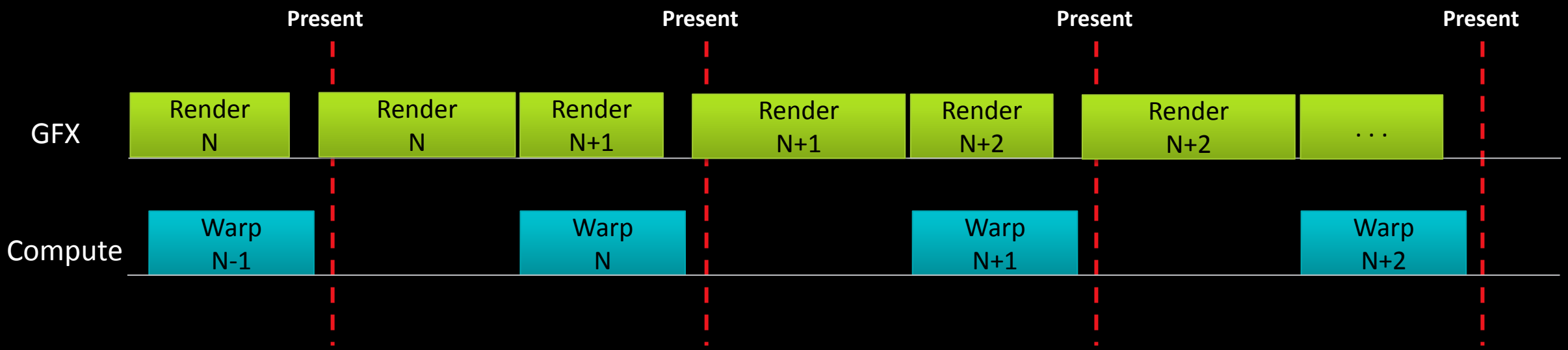
ASYNCHRONOUS COMPUTE FOR VR

- Execute VR image processing while rendering
- Creates superb VR experience
  - Minimizes latency
  - Eliminates judder
- Execution priority controls (QoS)



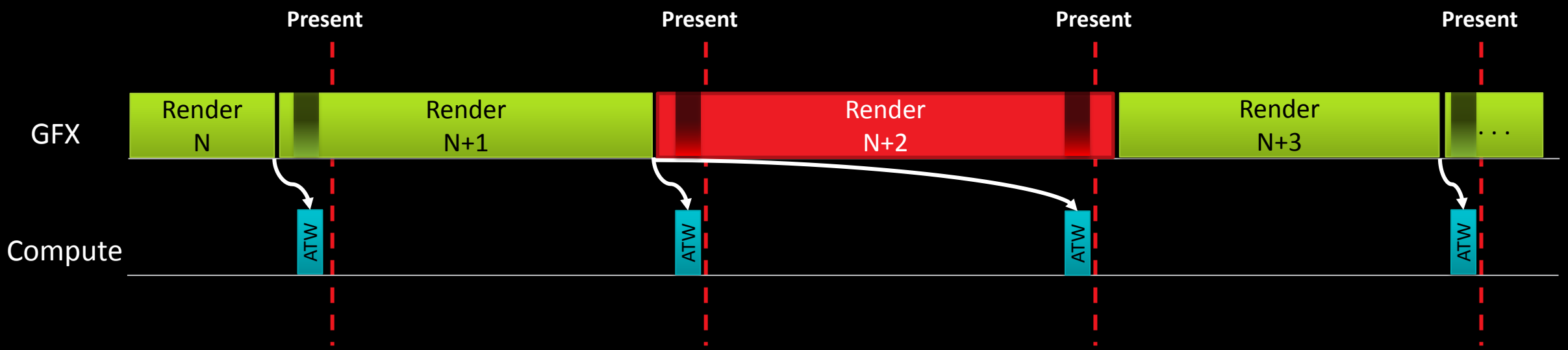
REGULAR PRIORITY COMPUTE

- Helps hide cost of image warp, but…
- No QoS = graphics could delay image warp
- Need knowledge and control of graphic workload



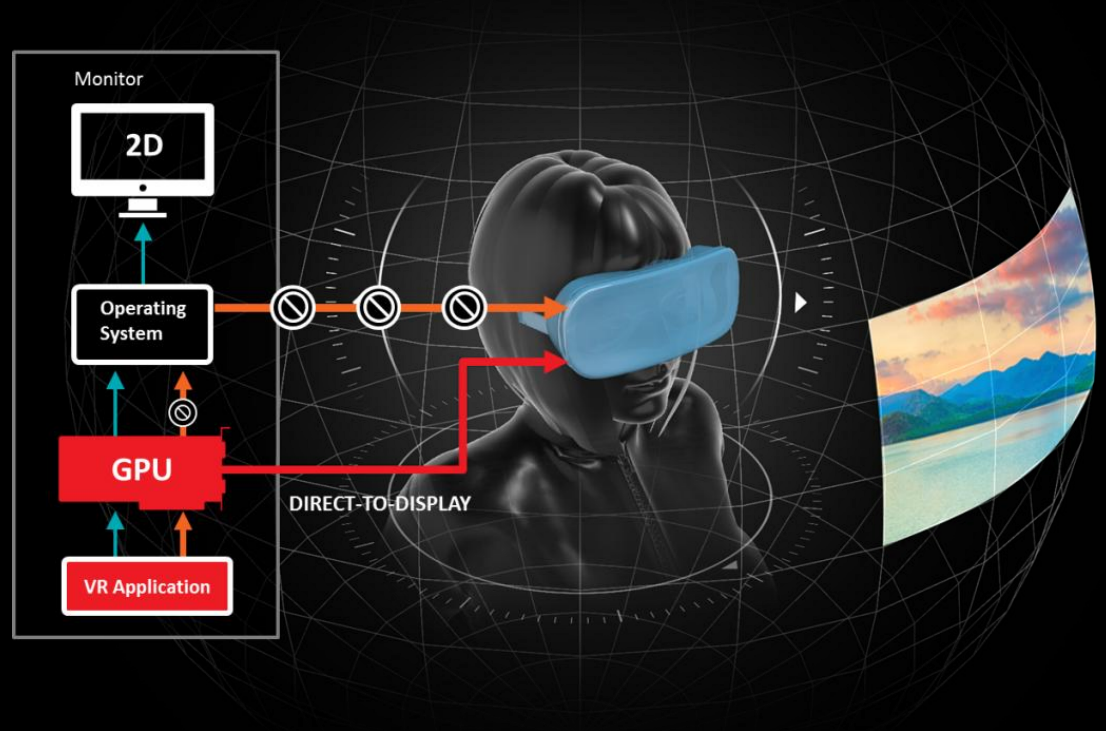
ASYNCHRONOUS COMPUTE QOS

- Favor compute vs. other workloads
- Better than draw call granularity
- Key for good asynchronous time-warp
  - –Judder compensation



DIRECT-TO-DISPLAY

- Brings native HMD support to OS
  - –Hide display from OS
  - –Doesn’t mess up desktop
- Lots of low level controls
  - –Perfect for embedded solutions
- Simplifies user experience



DIRECT-TO-DISPLAY

- Available to HMD vendors (under NDA)
- Contact us for more info…

MORE TOOLS FOR YOUR TOOLBOX

- API interoperability
- Synchronization
- Timeline building
- …


FUTURE DEVELOPMENTS

FUTURE DIRECTIONS

- Improving efficiency for lower end GPUs and mobile solutions
- Better API interoperability
- Untethered experience
- Sound integration

POWER MANAGEMENT

- Extended operation for mobile systems
- Thermal event avoidance
- Part of async compute QoS solution for guaranteed performance


RADEON
TECHNOLOGIES GROUP
TRUEAUDIO NEXT
CARL WAKELAND, FELLOW DESIGN ENGINEER
AMD

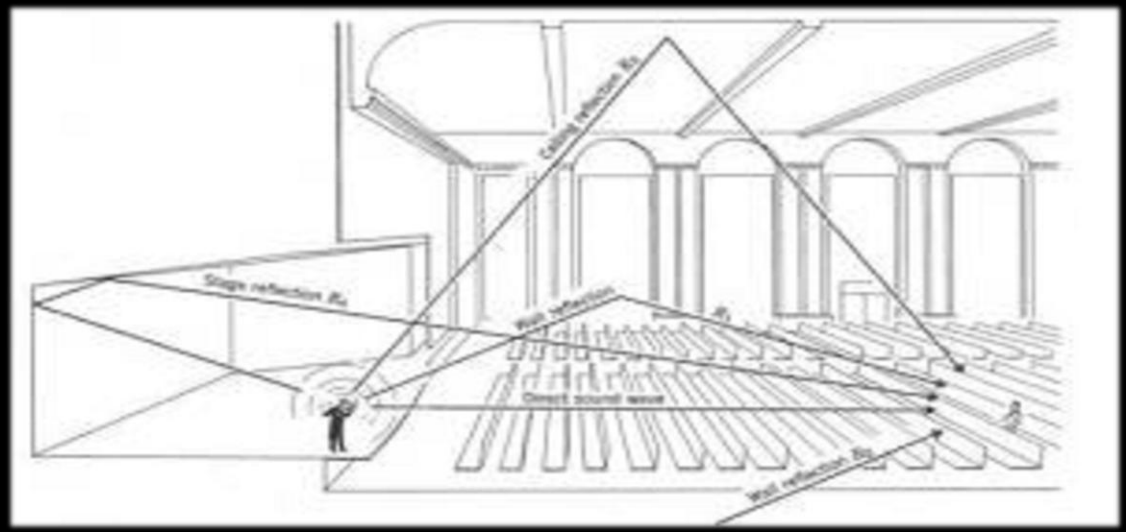
TRUEAUDIO NEXT

THE **RIGHT** NEXT STEP IN AUDIO RENDERING FOR VR

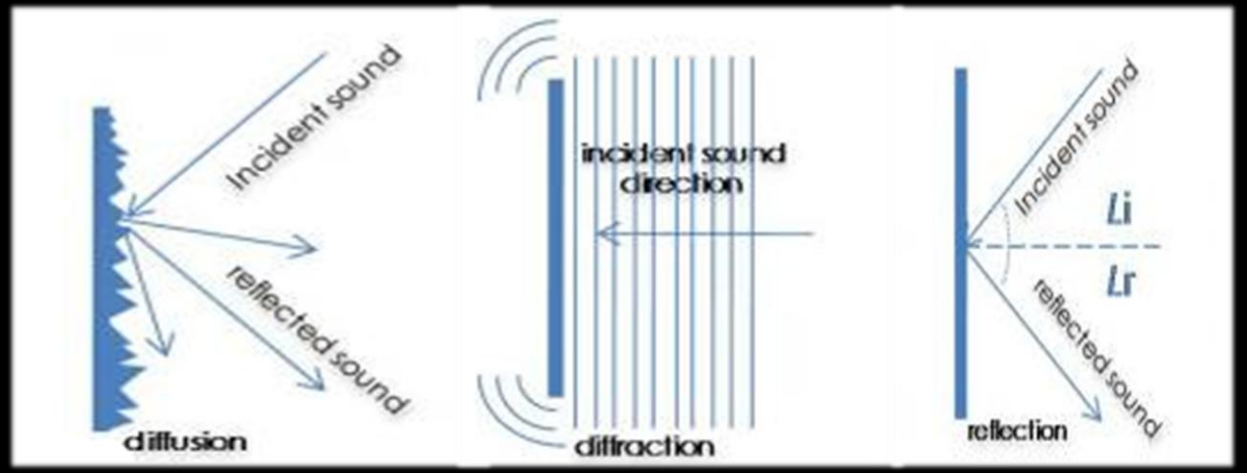
- 50% of consensus reality perception comes from what we hear.
- Studio-derived sound approximations are not reality, and our brains know it.
- We're ready to cross the chasm to full real-time physics-based audio rendering.
- TrueAudio Next is a future AMD technology that allows real-time audio to leverage the powerful resources of GPU Compute, safely coexisting with the world's best graphics, with performance not possible on CPU alone.
- TrueAudio Next is not an embedded DSP. VR Audio is gaining a seat at the grown-ups table -- GPU audio compute, with scalability.

CREATING PRESENCE IN VR REQUIRES PHYSICAL ACOUSTICS-BASED SOUND MODELING

- When we hear a sound in the real world, we hear a dynamic superposition of many reflected paths plus the direct path.

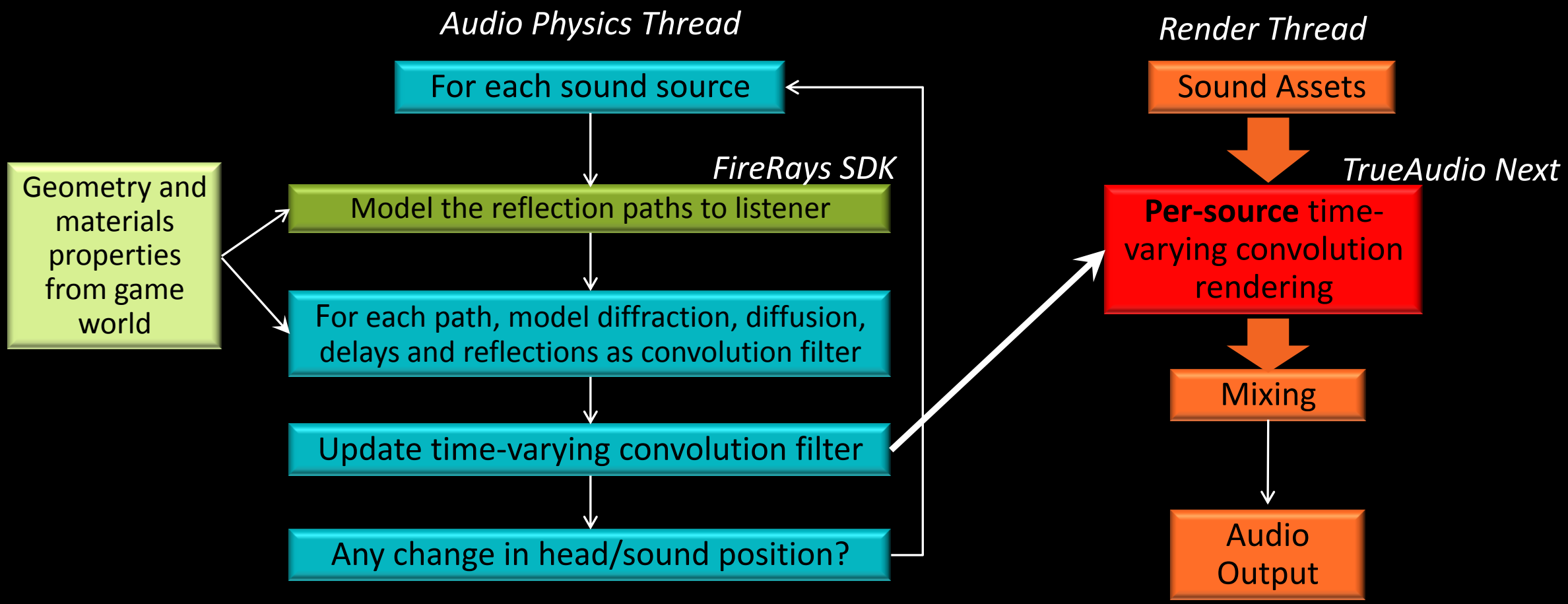


- Diffusion, diffraction and absorption/dampening by reflecting materials add more complexity to reflections



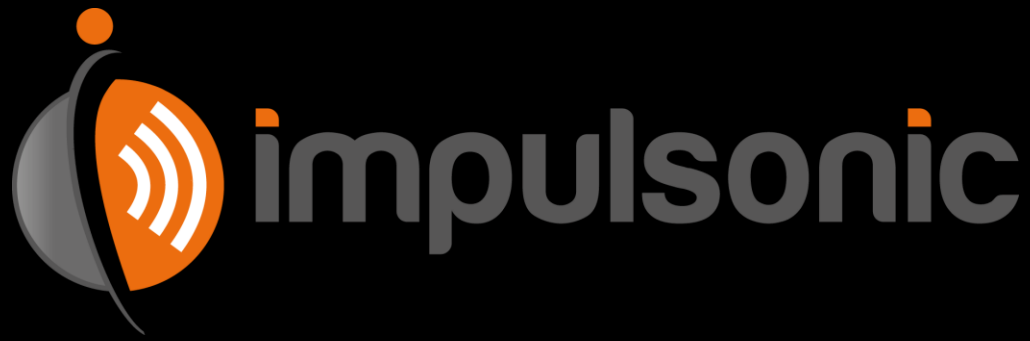
EXAMPLE OF VR AUDIO RENDERING

- Geometric acoustics adds **realistic physically modelled** real-time occlusion and acoustics to conventional binaural audio for VR



TRUEAUDIO NEXT PARTNERS

- TrueAudio Next cutting-edge partners are actively engaged, with more on the way. **Support our partners!**
- Ask us about plugin and engine integration – **be a partner!**
- Preview SDK available to NDA partners – **try it!**
- Get more details under NDA
- Hear the DEMOs



NEXT STEPS

- More features and optimizations
- Next generation APIs are great choice for VR
  - Not all pieces are there yet…
- **We want to hear from you!**


THANK YOU

DISCLAIMER & ATTRIBUTION

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION


AMD