Virtual Reality and Getting the Best from Your Next-Gen Engine

Ian Bickerstaff
Technical Director
Sony Computer Entertainment Europe
Immersive Technology Group

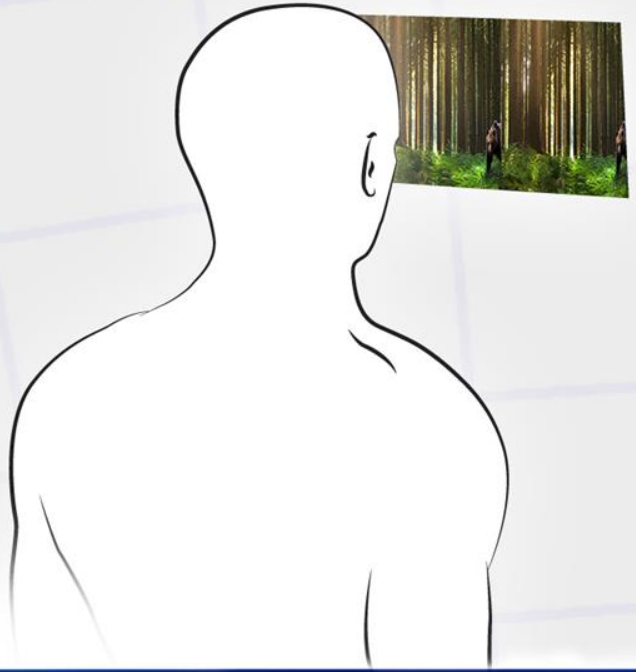
Patrick Connor
Principal Engineer
Sony Computer Entertainment Europe R&D

GAME DEVELOPERS CONFERENCE™ EUROPE
CONGRESS-CENTRUM OST KOELNMESSE · COLOGNE, GERMANY
AUGUST 11–13, 2014 · EXPO: AUGUST 11–12, 2014

Virtual Reality

believing you are really there…

Virtual Reality



display

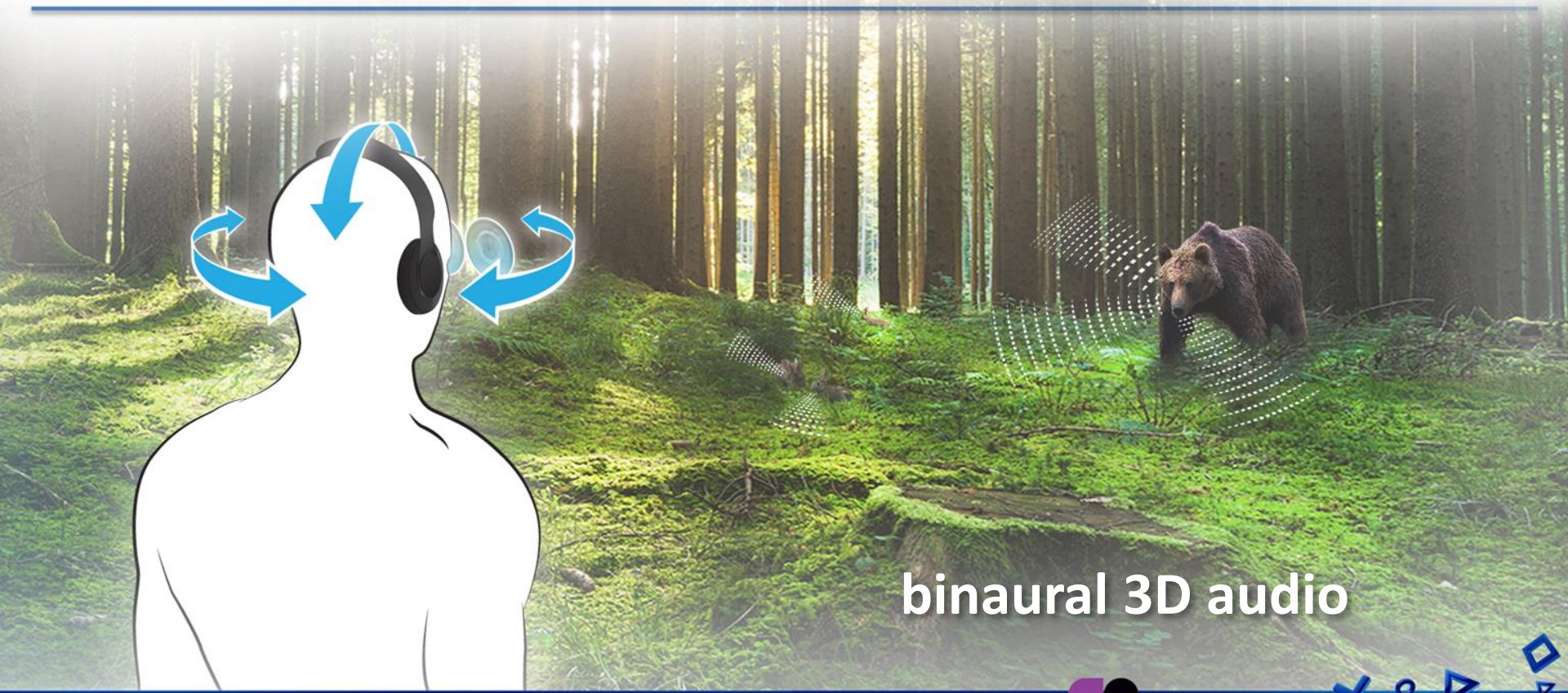
Virtual Reality

optics

Virtual Reality

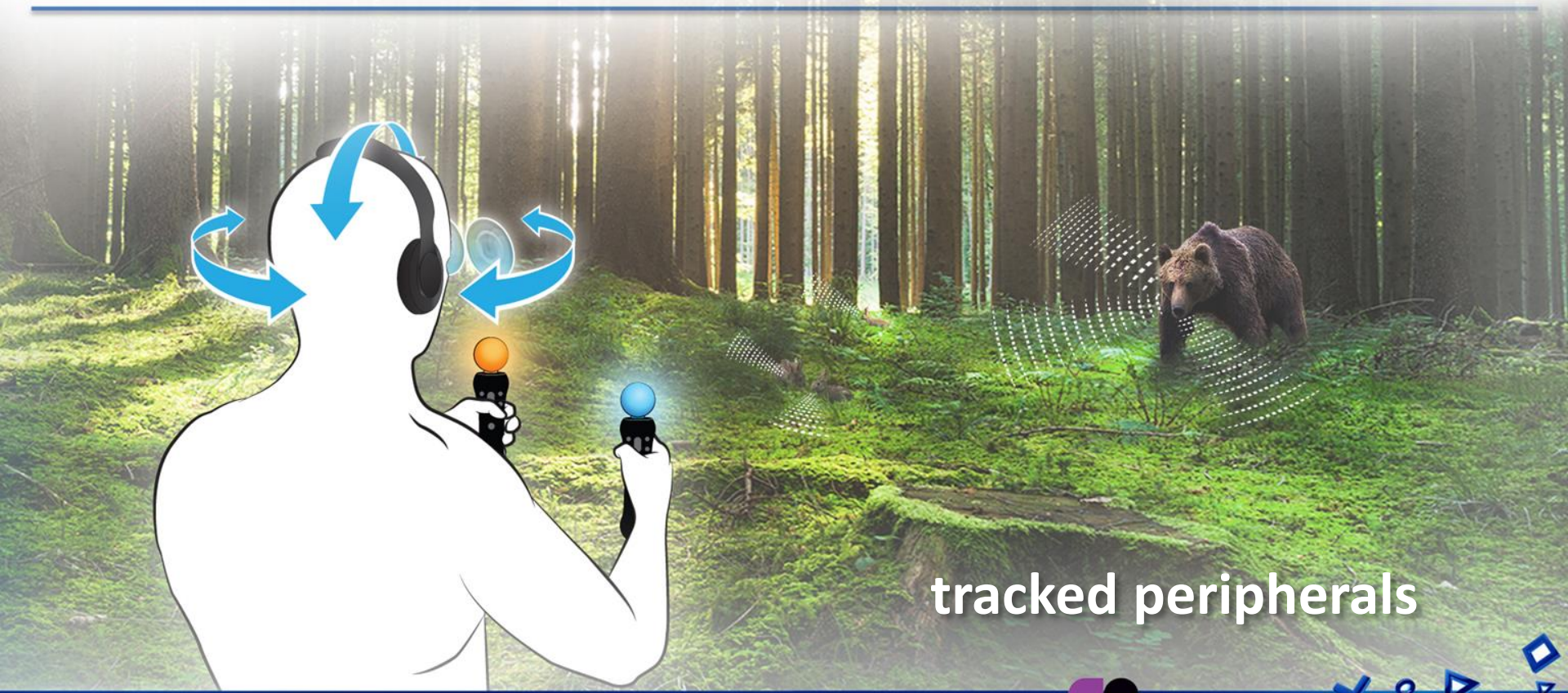
SONY

Virtual Reality

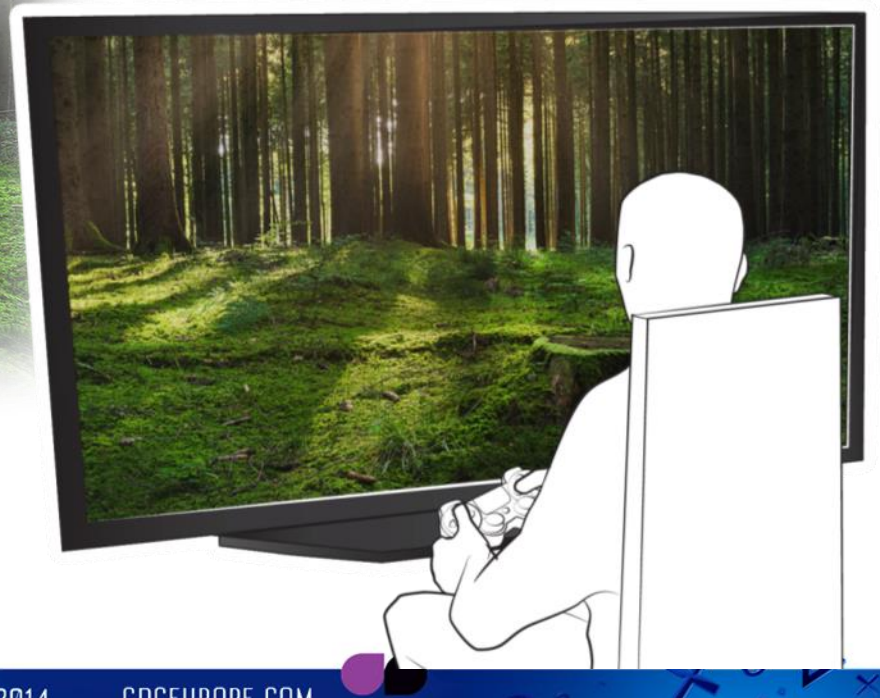


Virtual Reality

SONY



Virtual Reality



social screen

The virtual window

Why is looking at an image on a screen…

The virtual window

…not like looking through a window?

The virtual window

camera's view

move our viewpoint and the image looks wrong

The virtual window

camera's view

the focus distance is wrong too

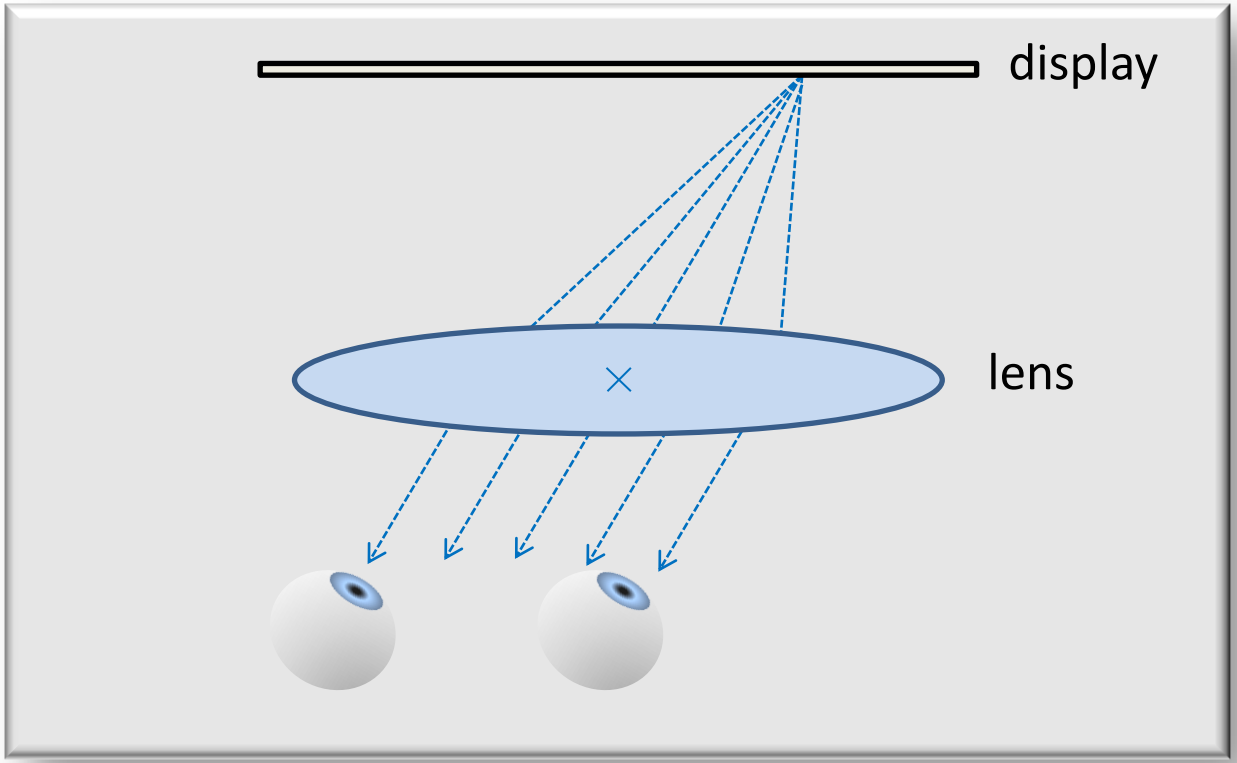
The virtual window

SONY

The solution?

The virtual window

The virtual window

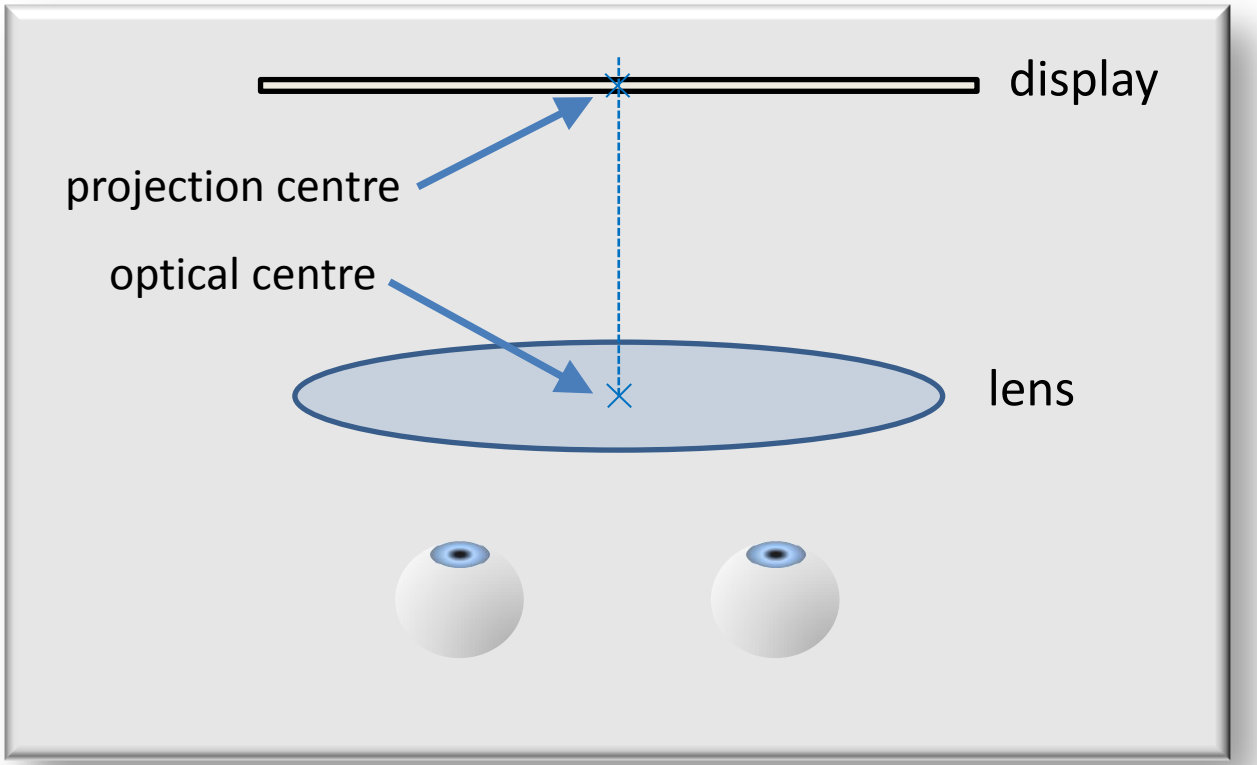


The virtual window

camera's view

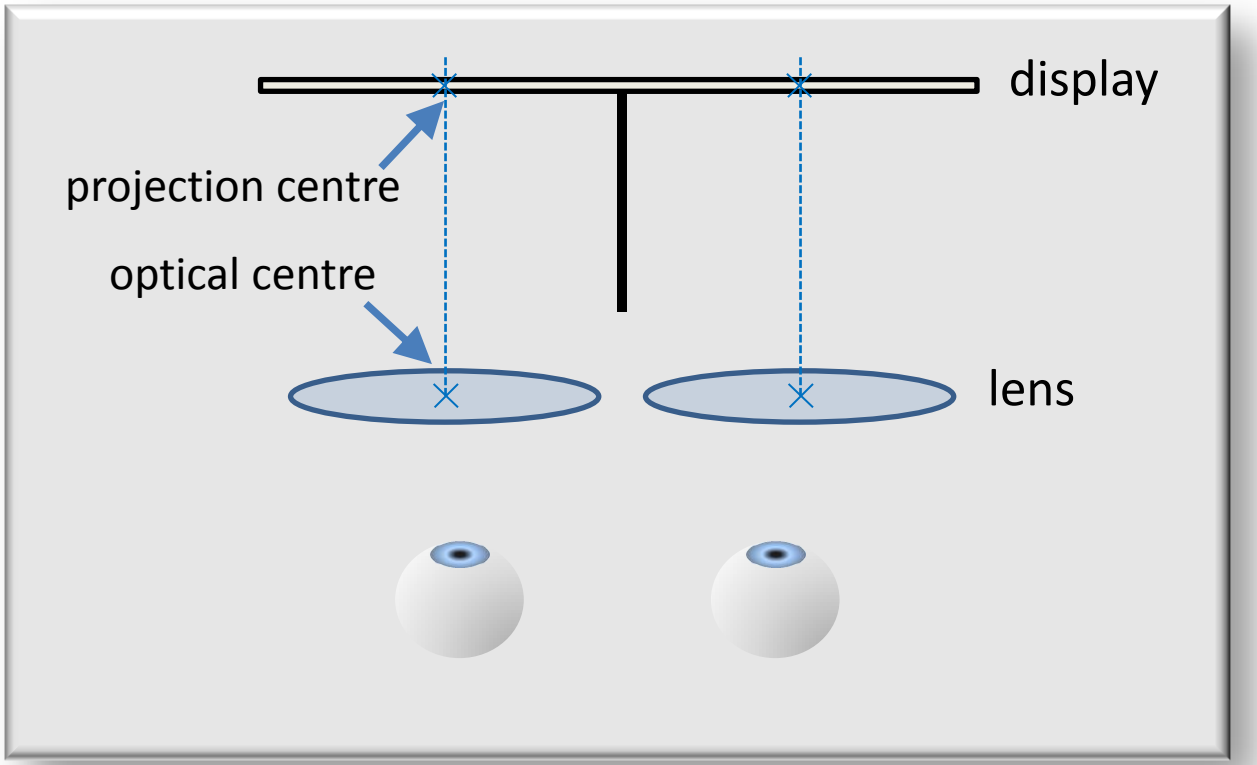
…a seamless transition from real to virtual!

The virtual window



alignment is critical

The 3D virtual window (stereoscope)



a different image for each eye

The 3D virtual window

left eye

right eye

adjustable for glasses wearers

Side-to-side movement

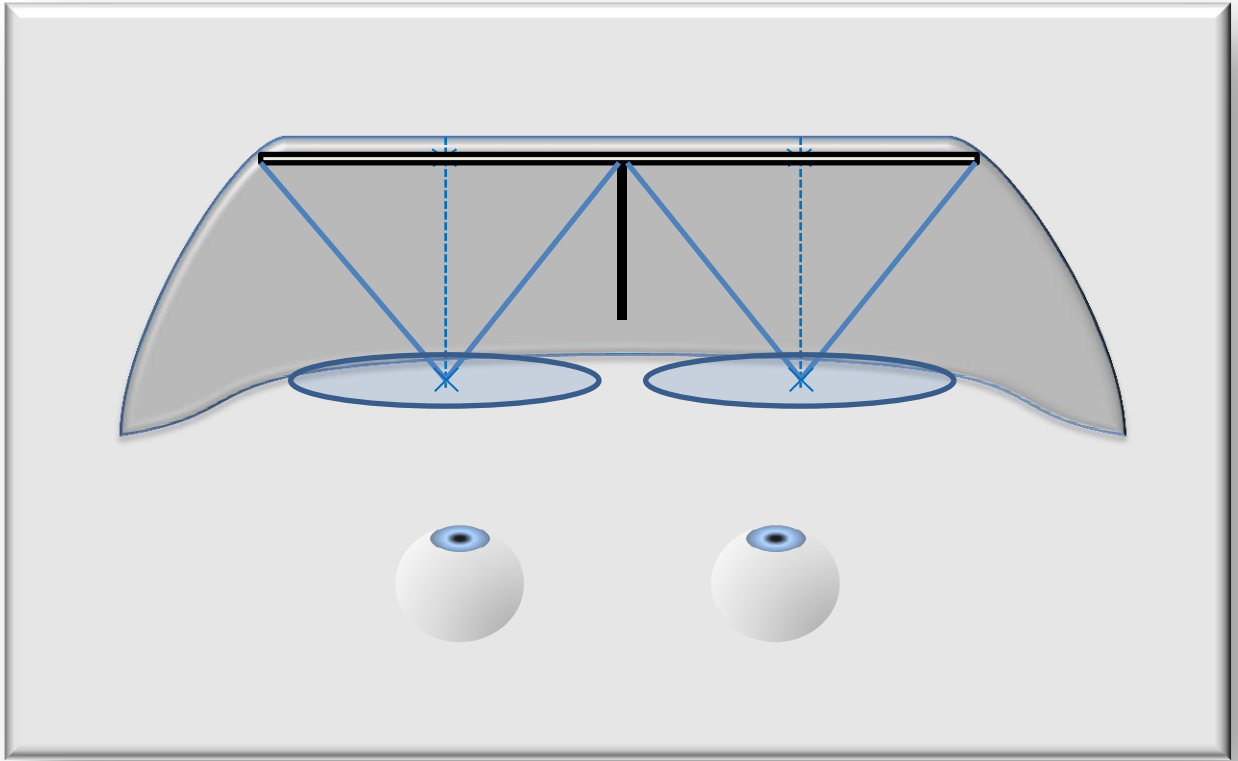
left eye

right eye

tolerant to variations in inter-pupillary distance without adjustment

Optics Summary

critical
(in SDK)

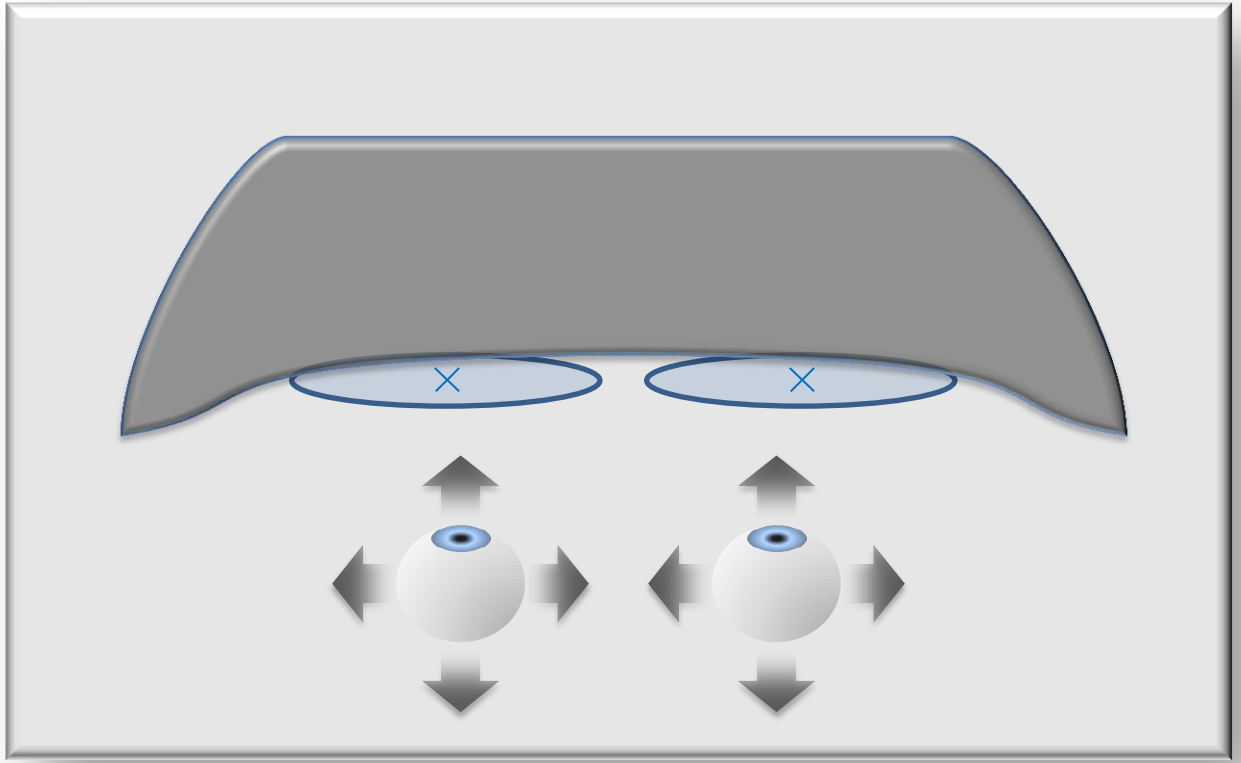


Optics Summary

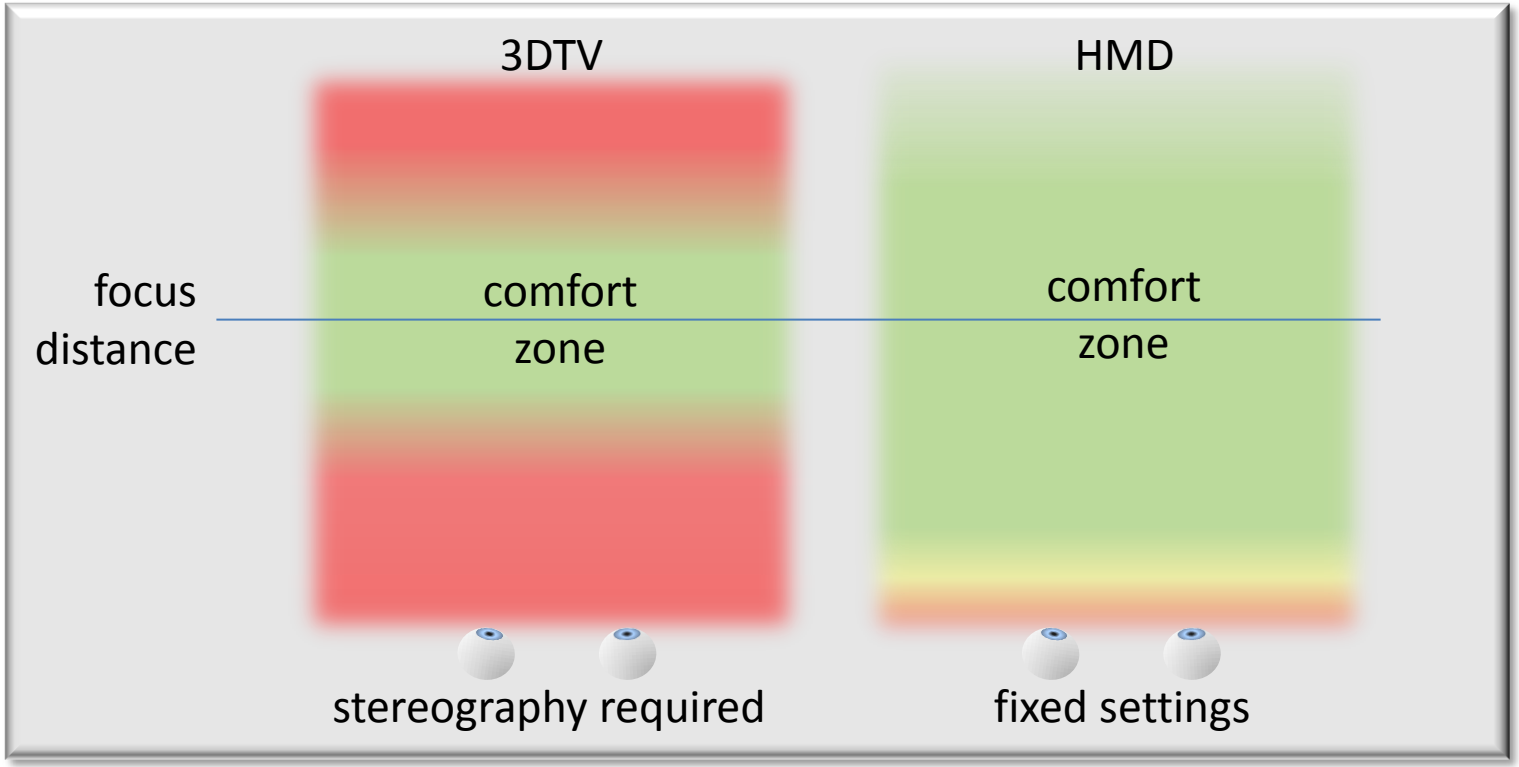
SONY

critical
(in SDK)

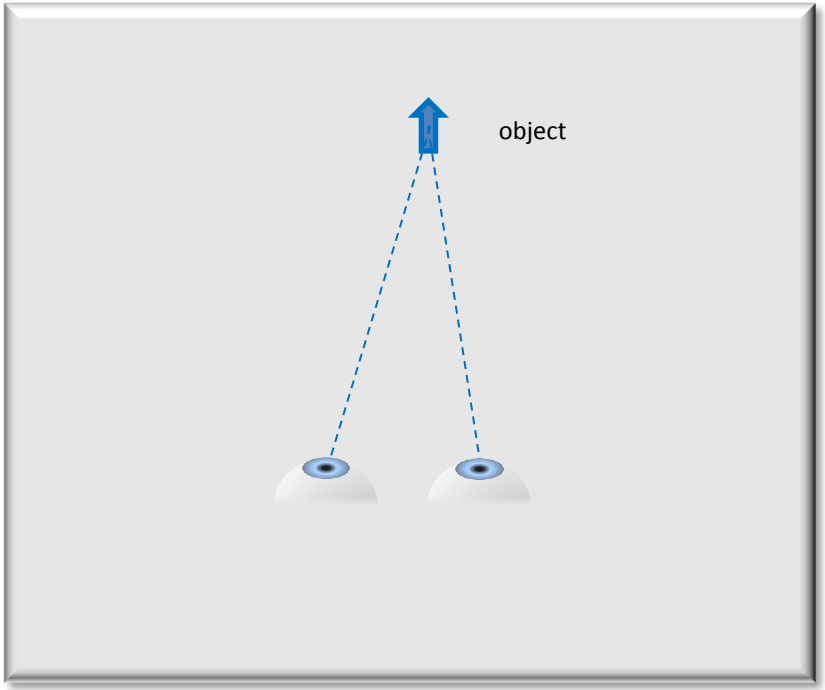
tolerant



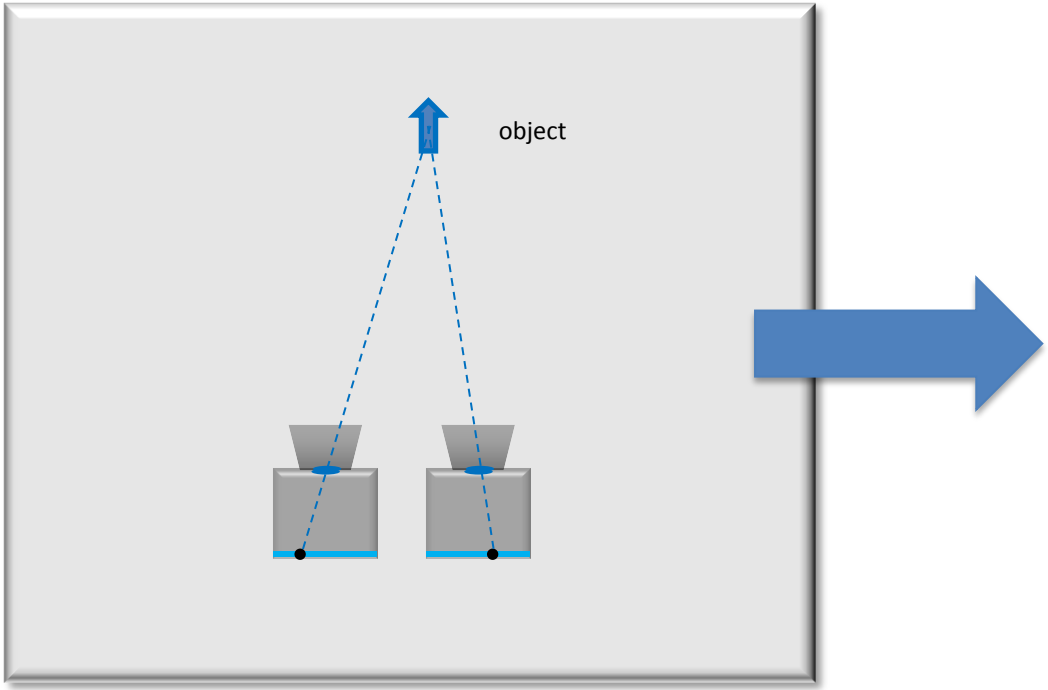
Stereoscopic 3D



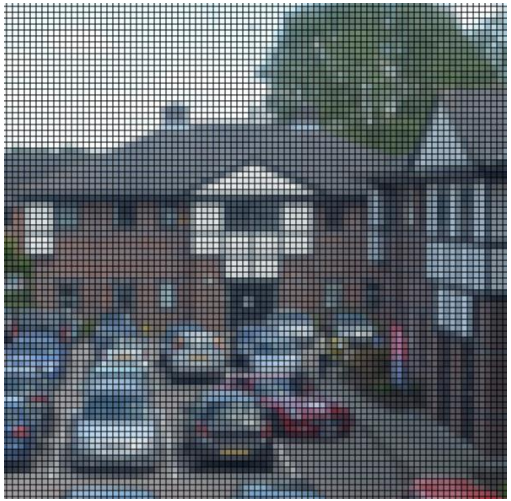
Ortho-stereoscopic viewing



Ortho-stereoscopic viewing



Wide field of view optics

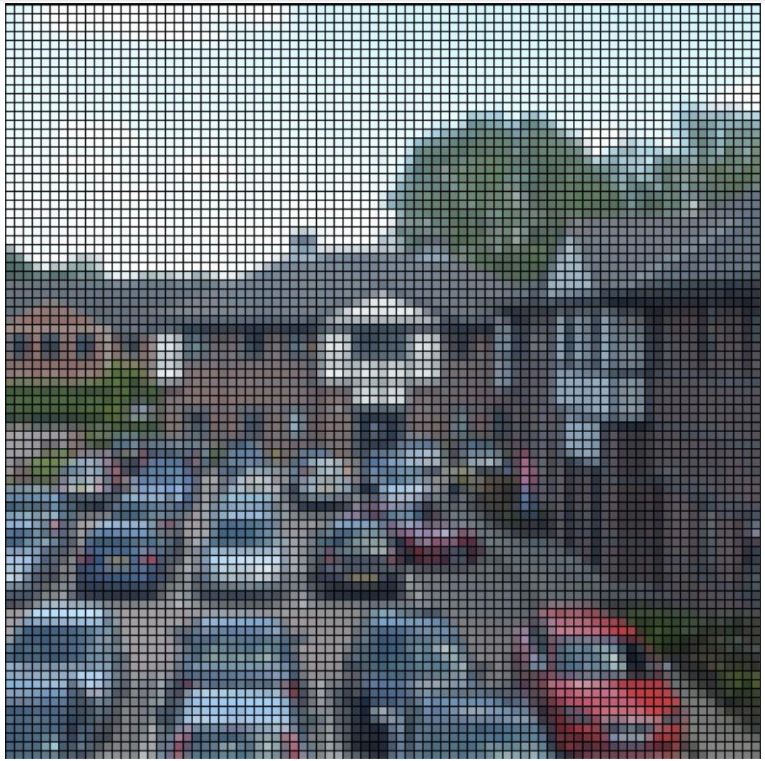


most HMDs have a narrow FOV…

Wide field of view optics

achieving a wide FOV requires a higher resolution display

Wide field of view optics

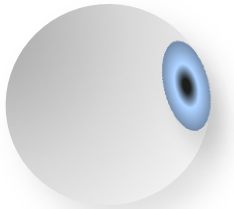


..or larger pixels

Wide field of view optics

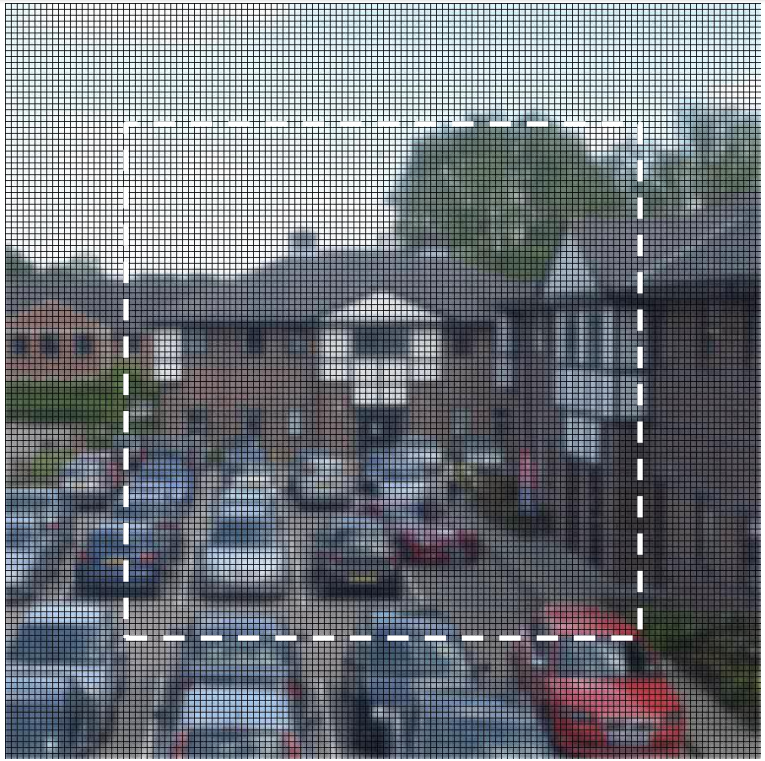
SONY

how to have the best of both worlds?



exploit the eye's variable acuity…

Wide field of view optics



compress the edges of the image with a distortion shader

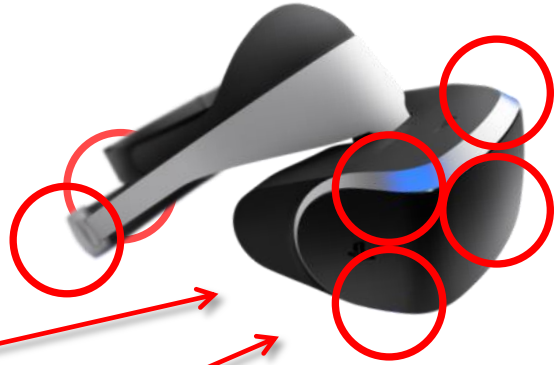
Wide field of view optics

the optics apply the inverse distortion so the edges look correct again

Wide field of view optics

small pixels in the centre, larger pixels at the edges

Tracking



accelerometers

gyroscopes

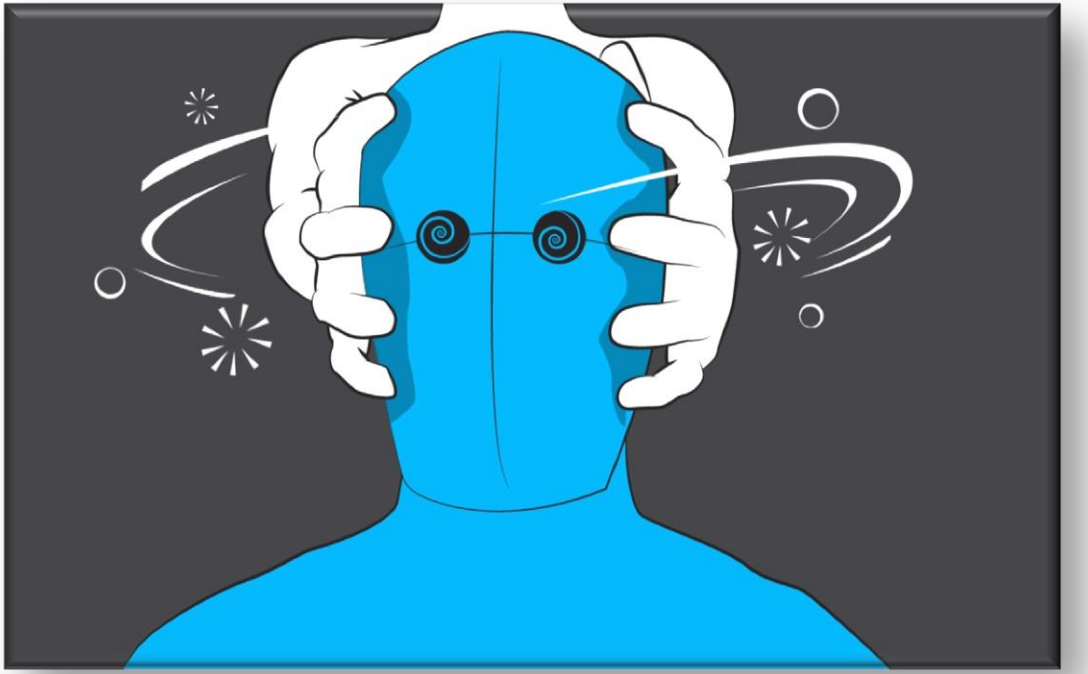
360° stereoscopic optical tracking

Tracking

sensor fusion

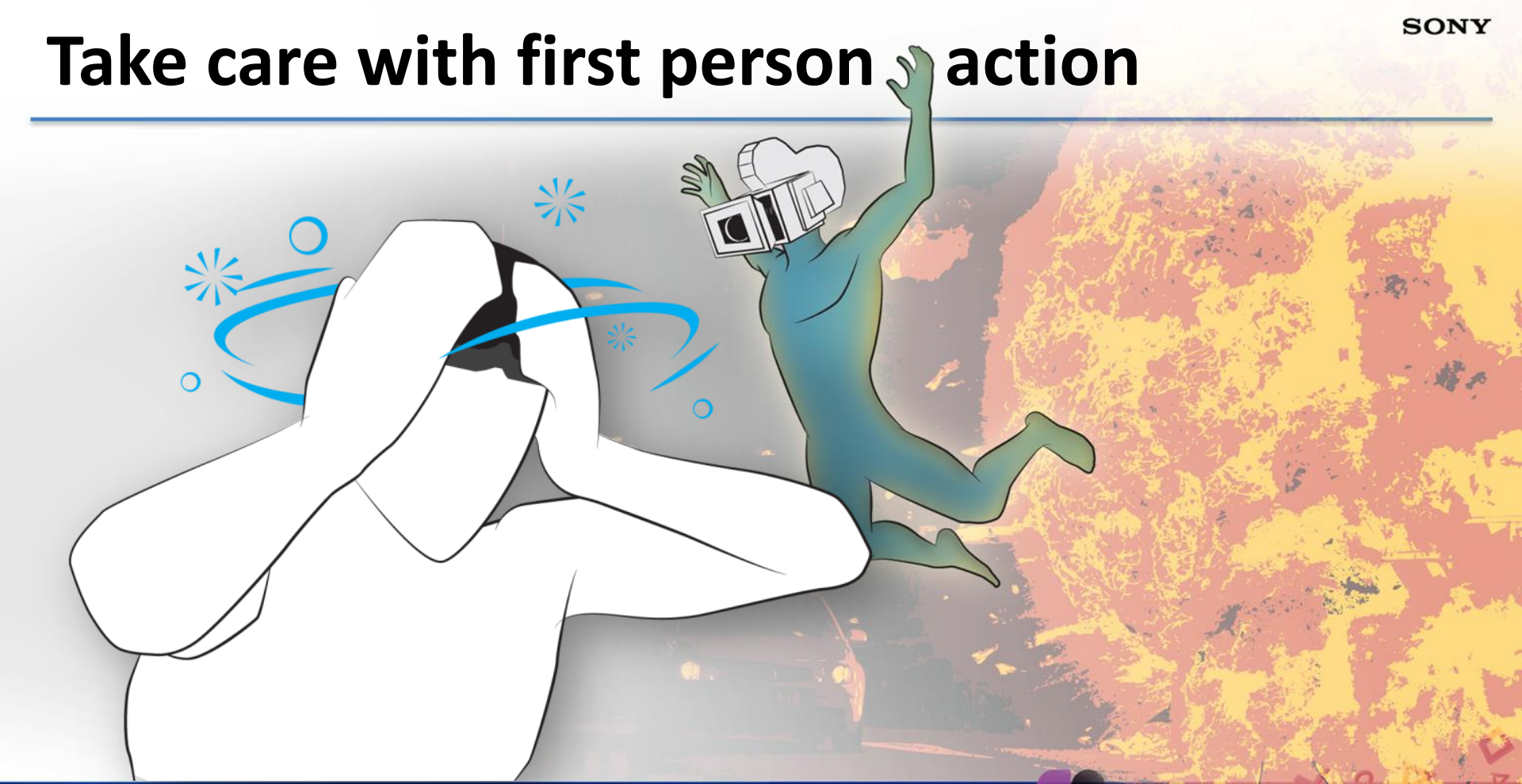
head movement

Don’t take control of the player’s head!



Take care with first person action

Take care with first person action



Take care with first person action

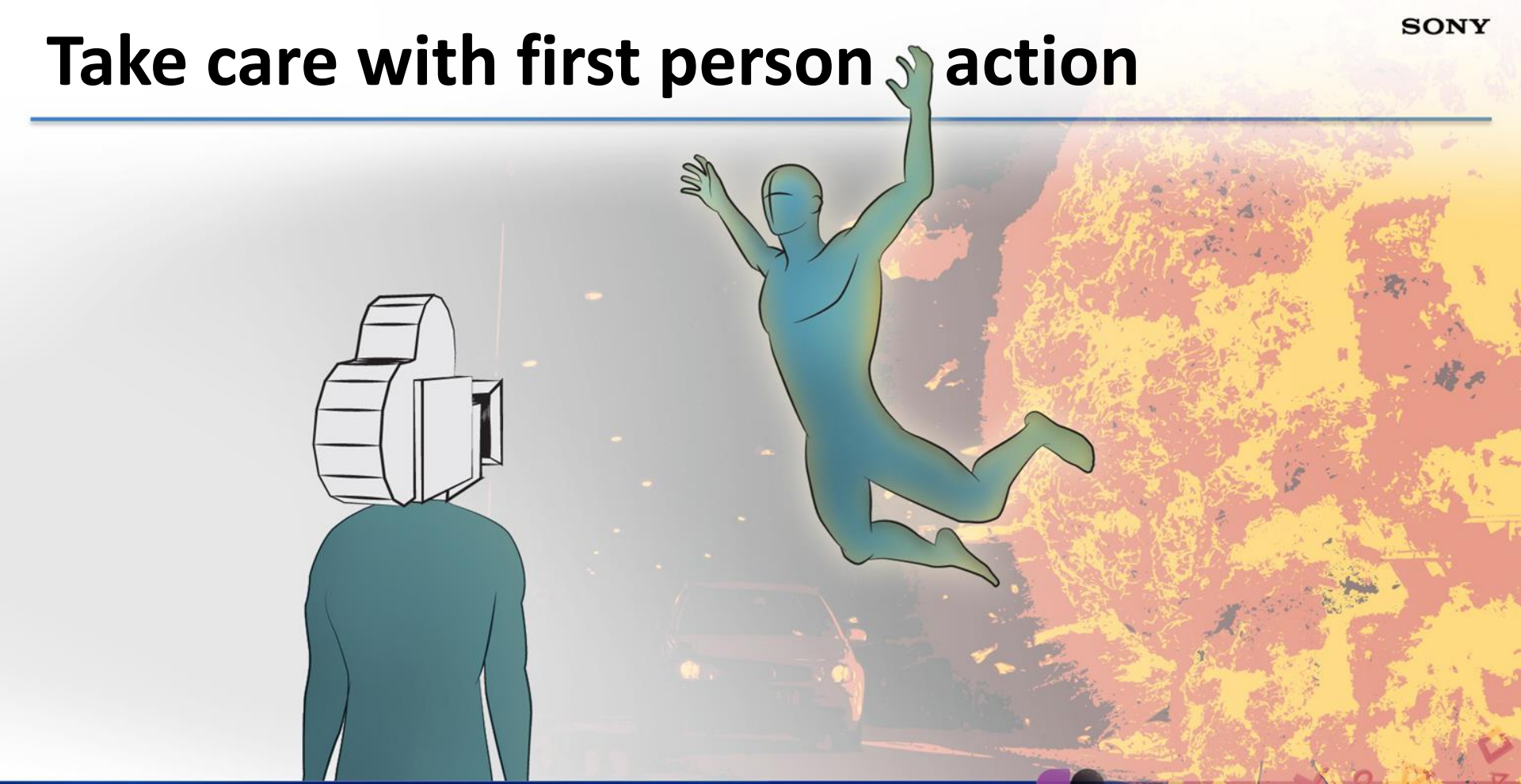
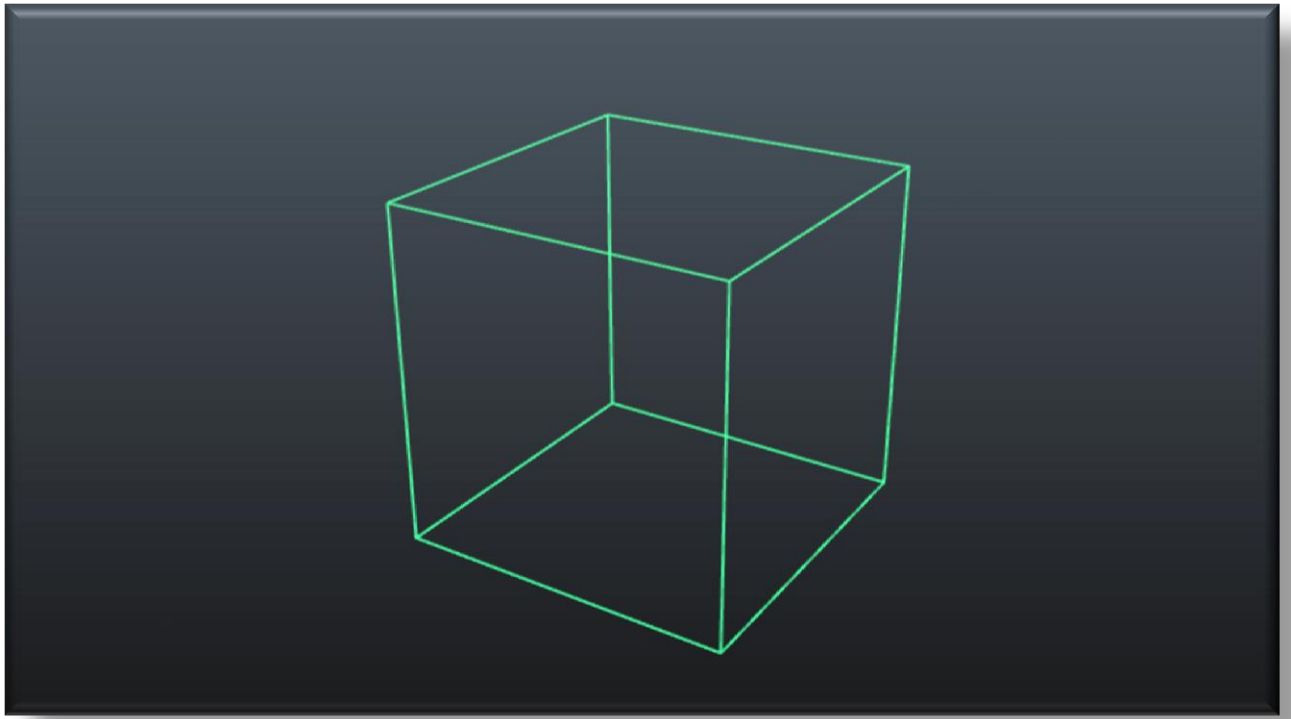


Photo-realism is not necessary



Don’t use cinematography!

SONY

~~*Variable Focal lengths*~~

~~*Filters*~~

~~*Lens flares*~~

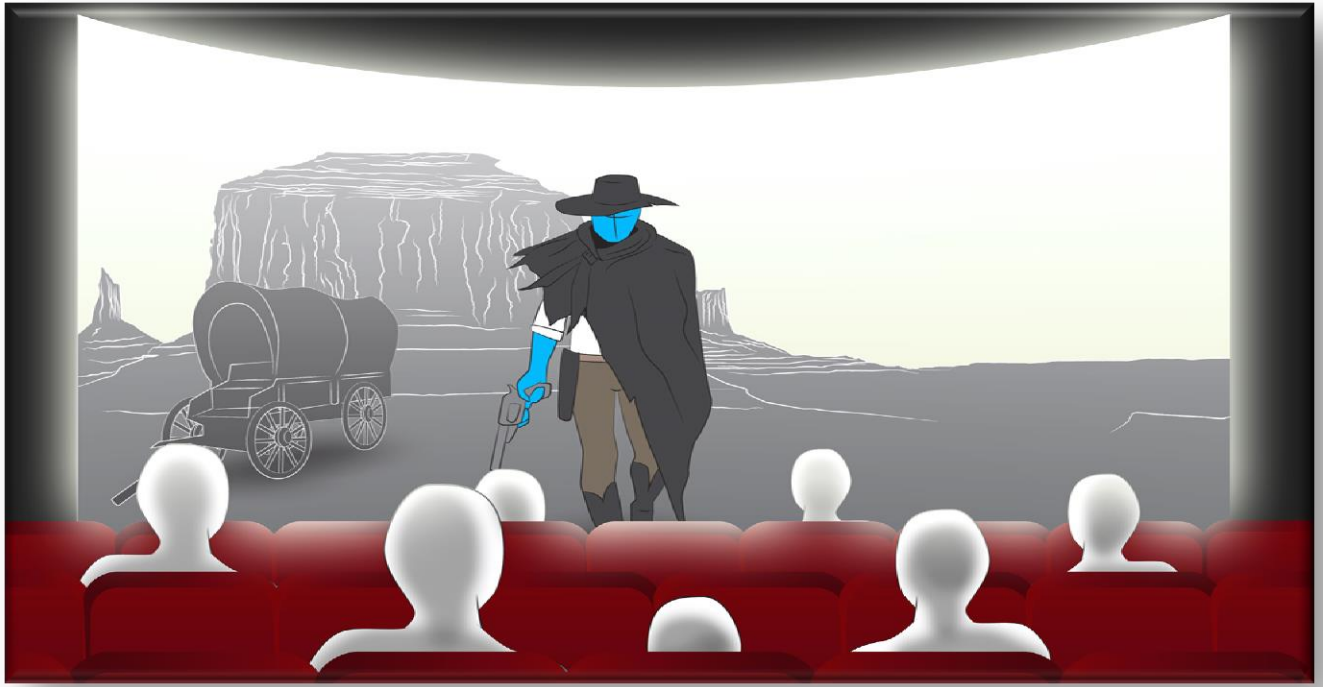
~~*Bloom*~~

~~*Film grain*~~

~~*Vignetting*~~

~~*Depth of field(?)*~~

Virtual cinema?



Stereoscopic Quality checks

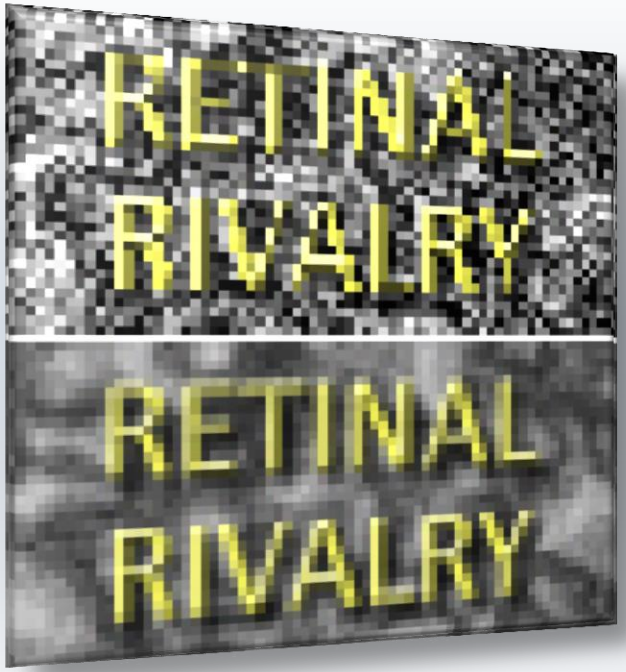
Stereoscopic Quality checks

1. Left and right the correct way round?

Stereoscopic Quality checks

1. Left and right the correct way round?
2. Same elements in both eyes?

Stereoscopic Quality checks



1. Left and right the correct way round?
2. Same elements in both eyes?

Stereoscopic Quality checks

1. Left and right the correct way round?
2. Same elements in both eyes?
3. Both images represent the same time?

Stereoscopic Quality checks

1. Left and right the correct way round?
2. Same elements in both eyes?
3. Both images represent the same time?

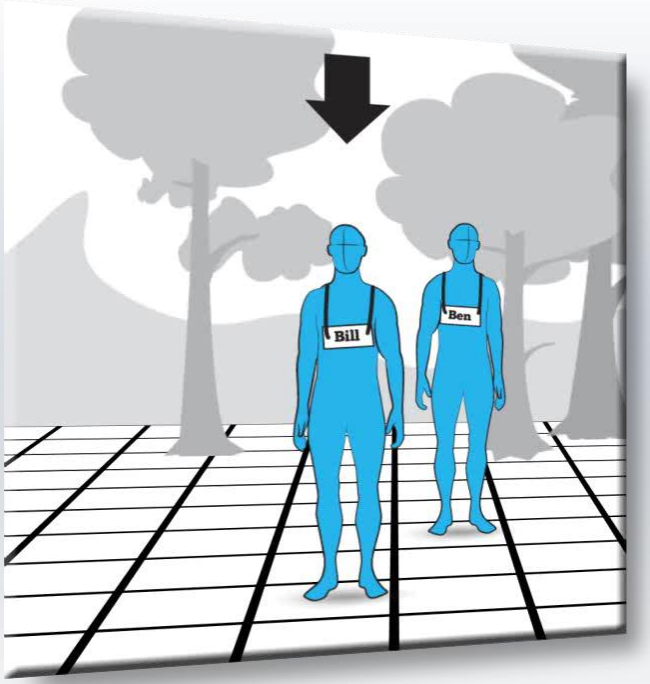
Stereoscopic Quality checks

1. Left and right the correct way round?
2. Same elements in both eyes?
3. Both images represent the same time?
4. The scale is correct?

Stereoscopic Quality checks

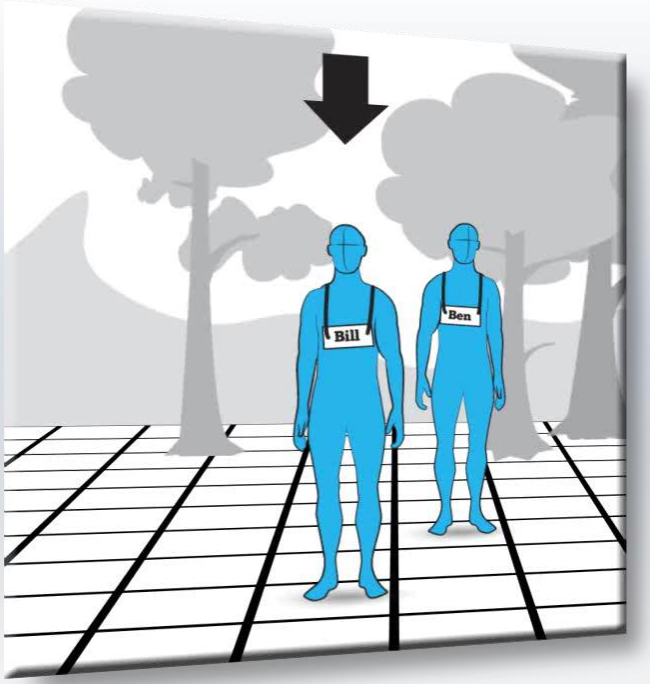
1. Left and right the correct way round?
2. Same elements in both eyes?
3. Both images represent the same time?
4. The scale is correct?
5. The depth is consistent?

Stereoscopic Quality checks



1. Left and right the correct way round?
2. Same elements in both eyes?
3. Both images represent the same time?
4. The scale is correct?
5. The depth is consistent?

Stereoscopic Quality checks

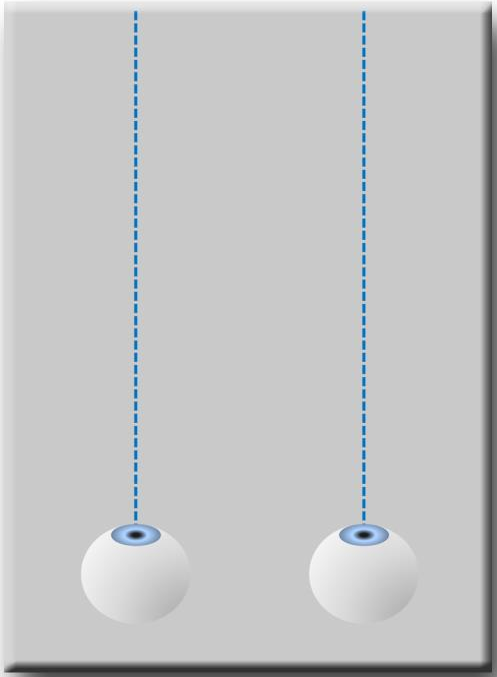


1. Left and right the correct way round?
2. Same elements in both eyes?
3. Both images represent the same time?
4. The scale is correct?
5. The depth is consistent?

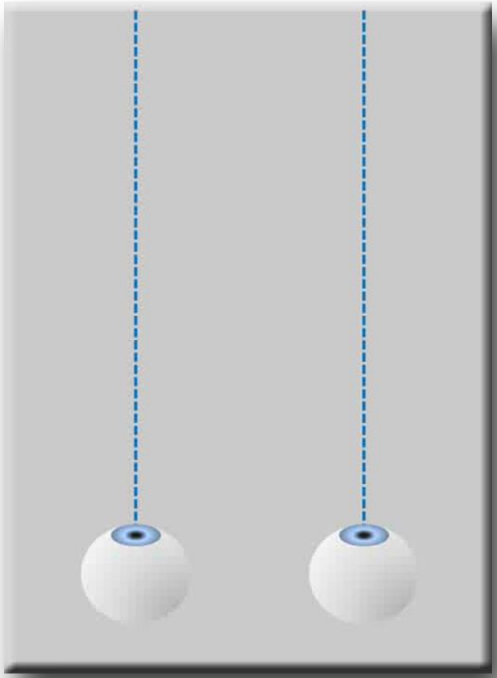
Stereoscopic Quality checks

1. Left and right the correct way round?
2. Same elements in both eyes?
3. Both images represent the same time?
4. The scale is correct?
5. The depth is consistent?
6. Rapid depth changes avoided?

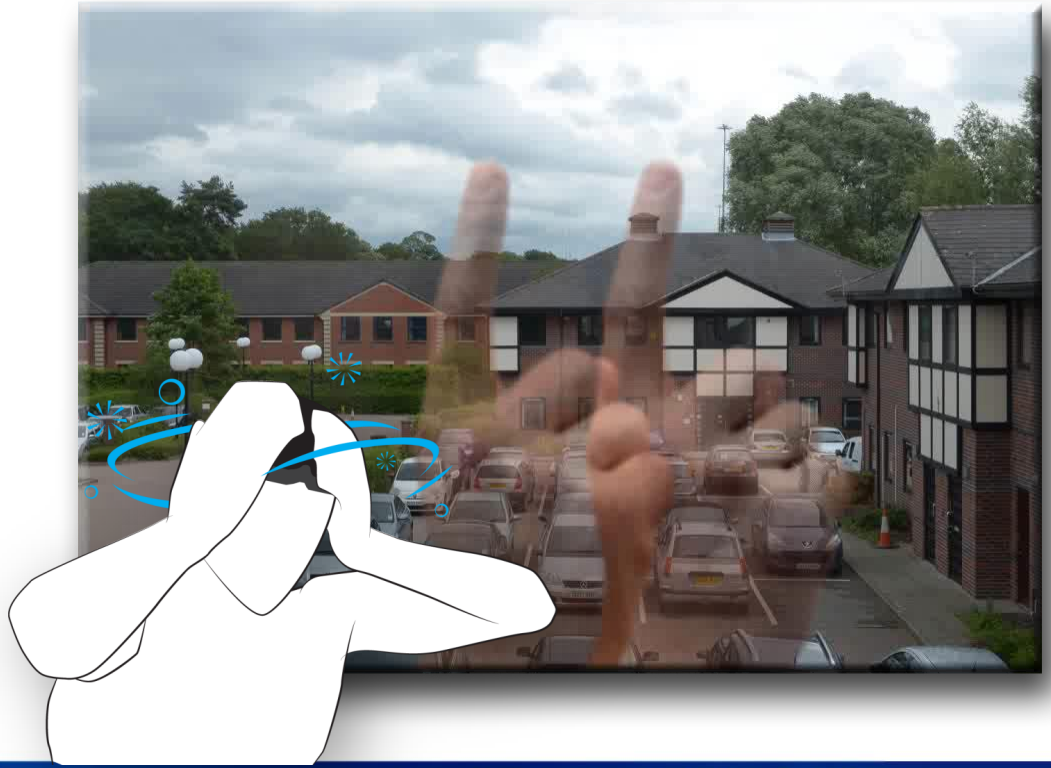
Fusion zones

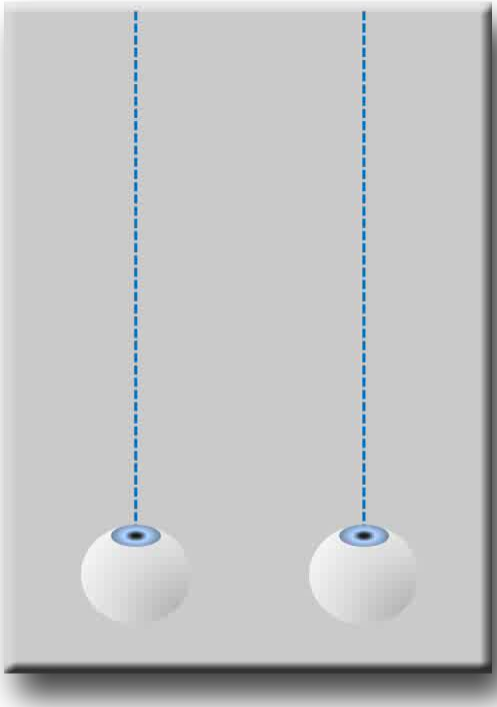


Fusion zones



Fusion zones





Virtual reality

believing you are really there…

Getting The Best
From Your Next-Gen Engine

Overview

- Every engine is different
- But there will always be some similarities
- The considerations I'll discuss today are really important for VR…
- …but a lot of them are also important for any next-gen engine

What Makes A Good Engine?

- High-quality visuals
- Consistent, high frame rate
- Low latency

What Makes A Good VR Engine?

- High-quality visuals
- Consistent, high frame rate
- Low latency

What Makes A Good VR Engine?

- High-quality visuals
- Consistent, high frame rate
- Great tracking and calibration
- Low latency

Testbed Engine

- I created a very simple testbed engine to test my experiments:
  - Stereo 3D rendering
  - Large range of depths
  - Multiple anti-aliasing solutions
  - Geometry optimisations
  - Multi-context rendering

High-Quality Visuals

High-Quality Visuals

- What do we mean by high-quality visuals?
  - Nothing distracting you from being immersed in the game
  - Good shading (but not necessarily photorealistic)
    - This often means good anti-aliasing

High-Quality Visuals

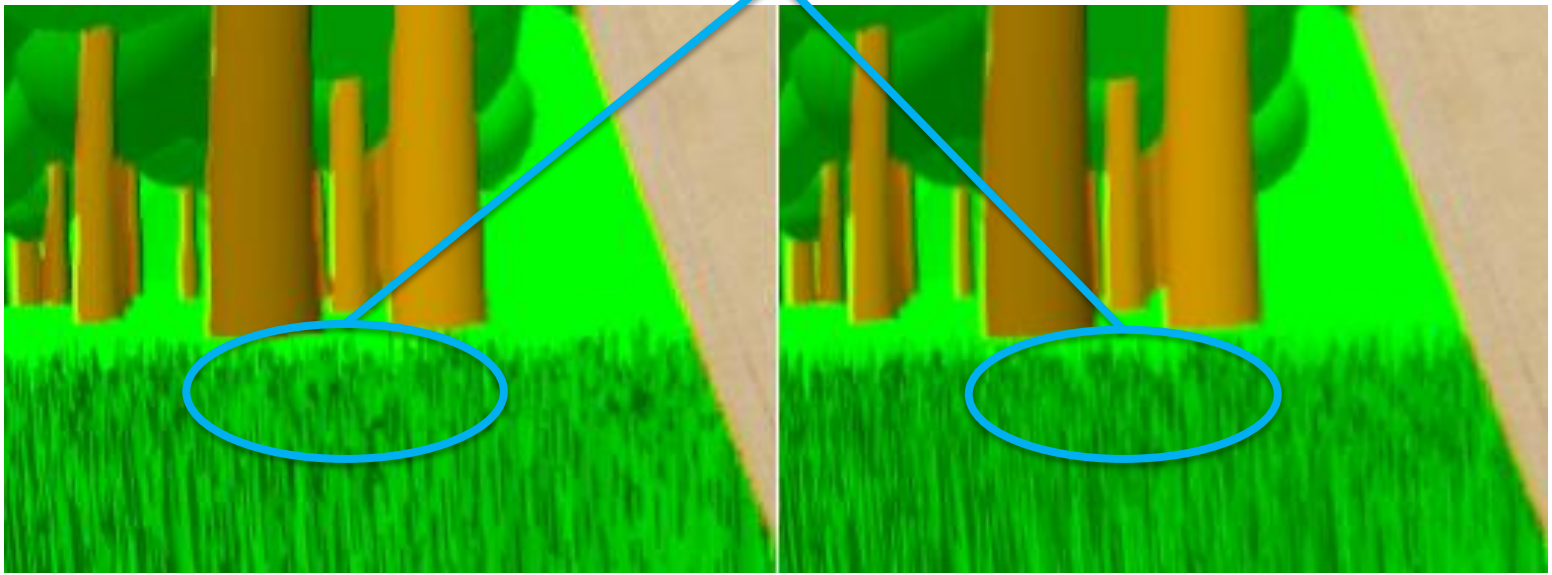
- Why is good anti-aliasing essential?
  - The nature of human perception means we're easily distracted by high frequency noise
  - Distraction reduces the sense of presence
  - Aliasing artifacts can be even worse with stereo rendering
    - They can cause retinal rivalry
- **Good anti-aliasing can be more important than native resolution**

Anti-Aliasing Methods

- Edge geometry AA
  - Often hardware accelerated
- Image-space AA
  - Fit well with most rendering pipelines
  - FXAA[1], MLAA[2], SMAA[3] etc.
  - Temporal AA
    - Use re-projection for temporal supersampling
- See References at the end of the presentation for more info[4]

Edge-Geometry Anti-Aliasing

high-frequency geometry looks better

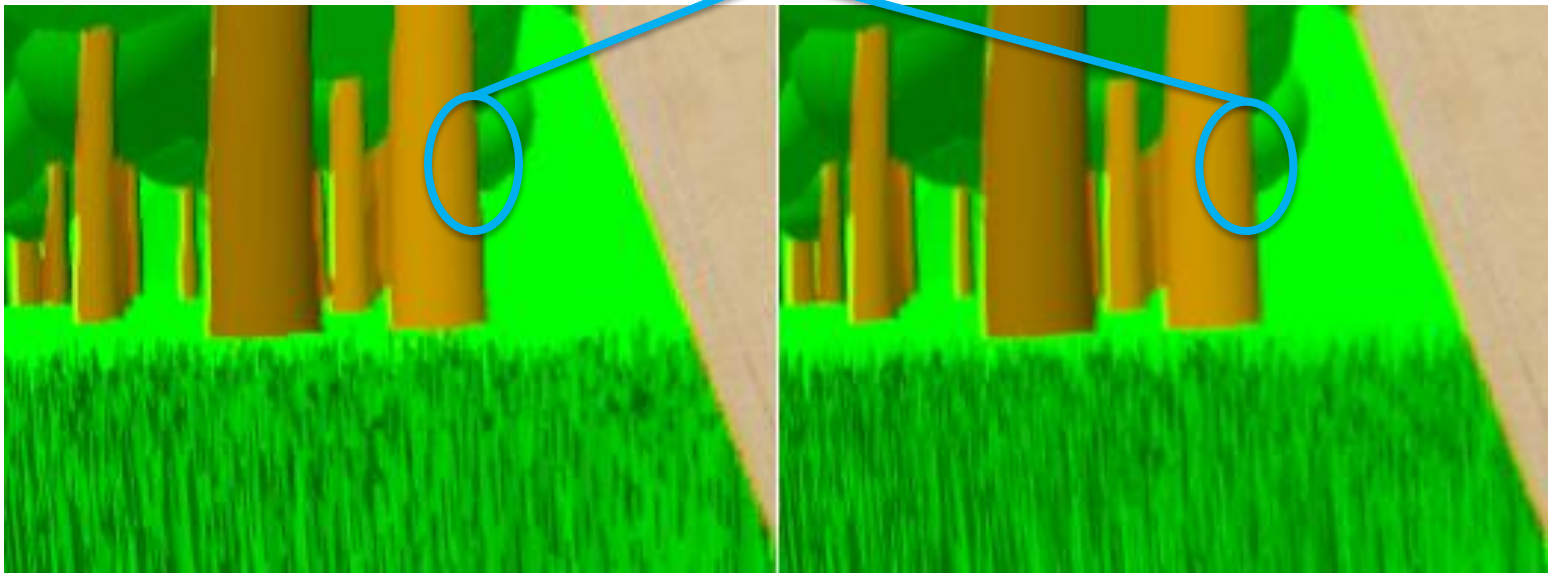


no anti-aliasing

2x MSAA

Edge-Geometry Anti-Aliasing

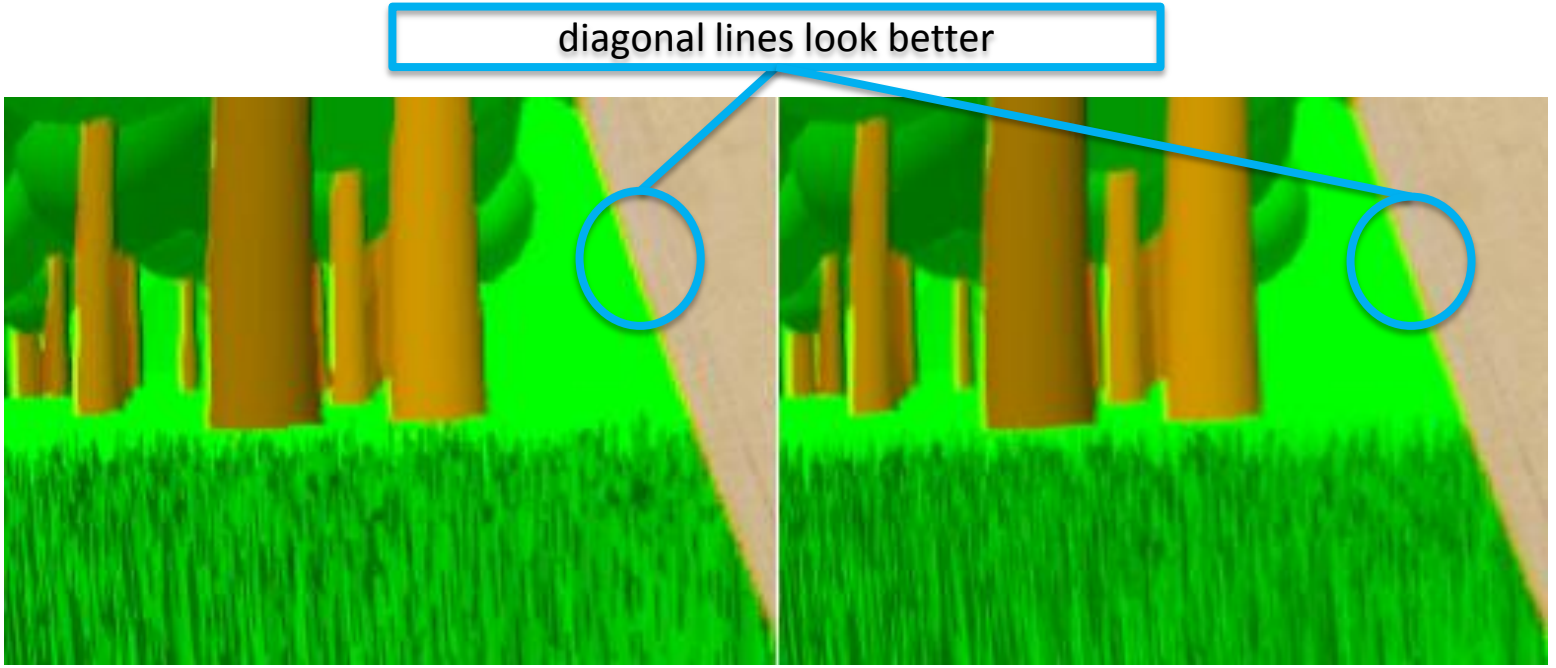
almost-vertical lines look better



no anti-aliasing

2x MSAA

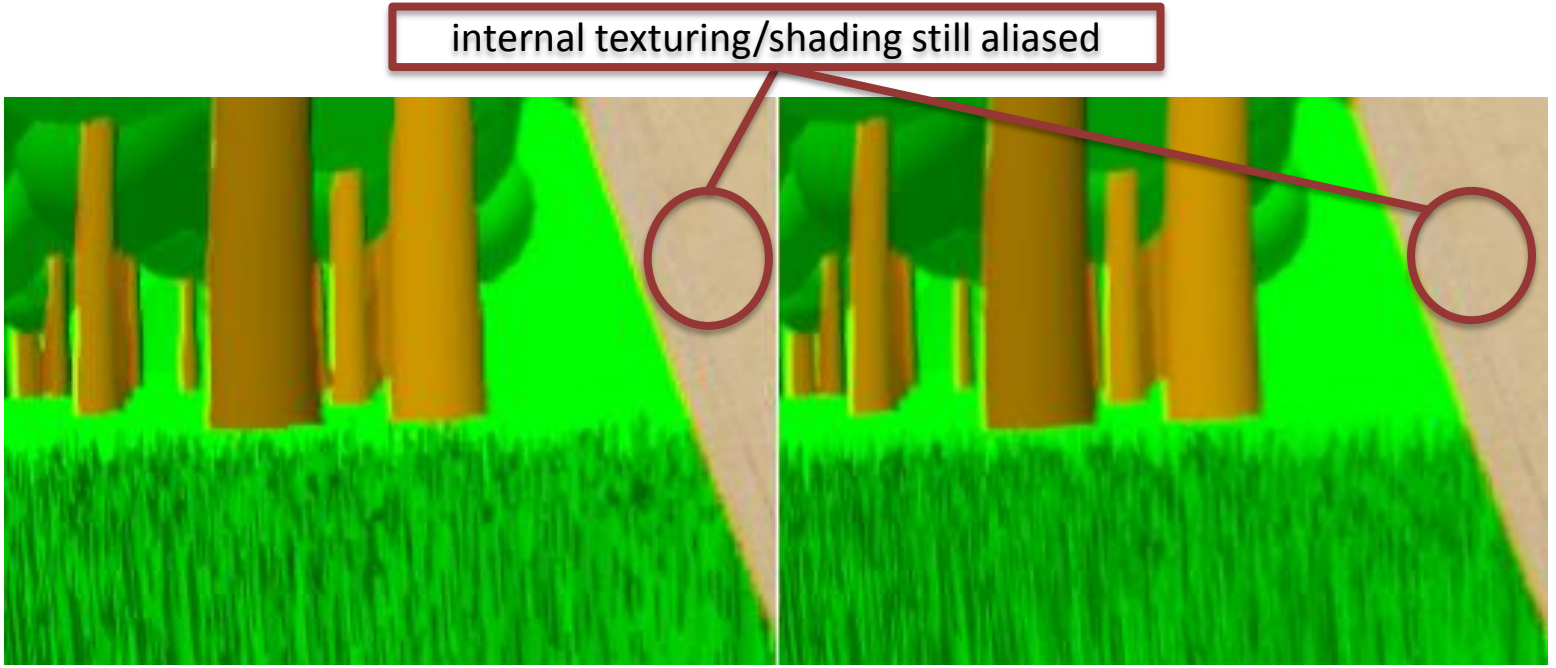
Edge-Geometry Anti-Aliasing



no anti-aliasing

2x MSAA

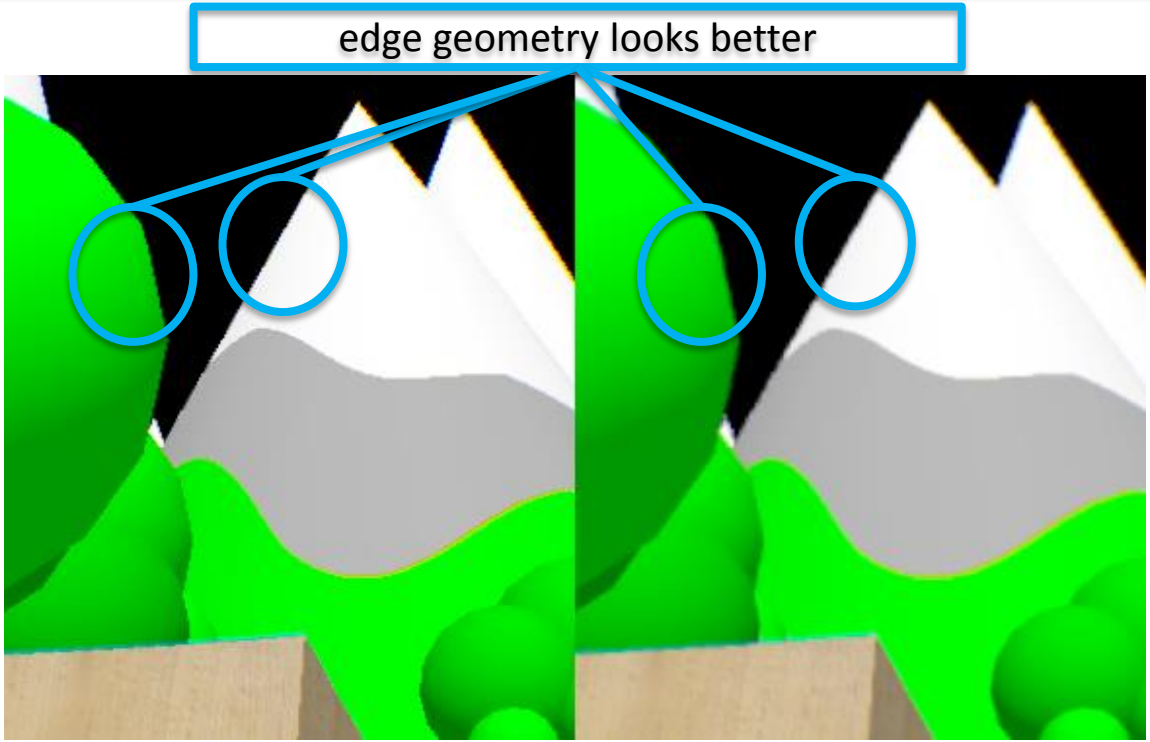
Edge-Geometry Anti-Aliasing



no anti-aliasing

2x MSAA

Image-Space Anti-Aliasing

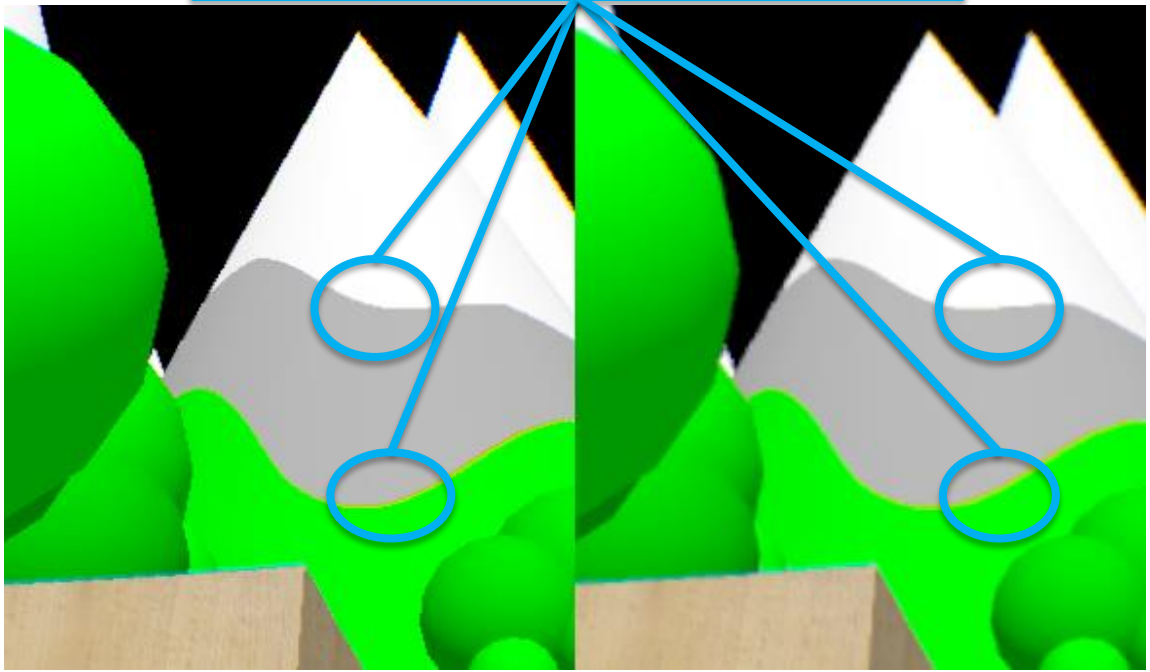


no anti-aliasing

FXAA

Image-Space Anti-Aliasing

texture/shading detail looks better

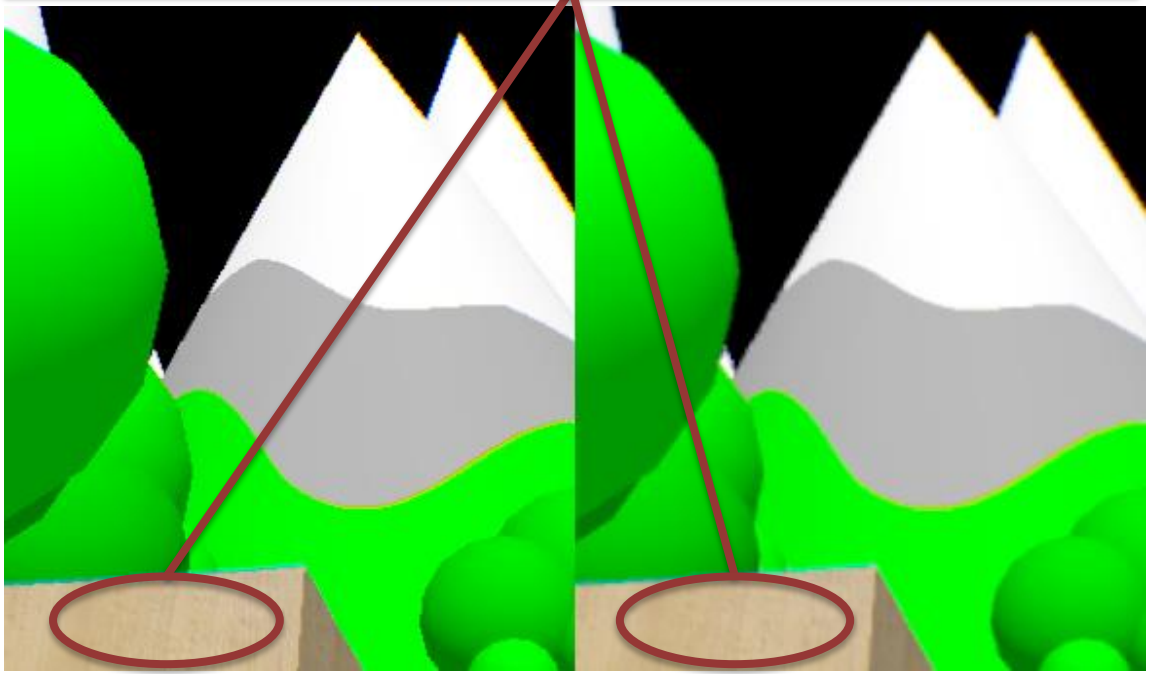


no anti-aliasing

FXAA

Image-Space Anti-Aliasing

can sometimes lose detail in high-frequency data



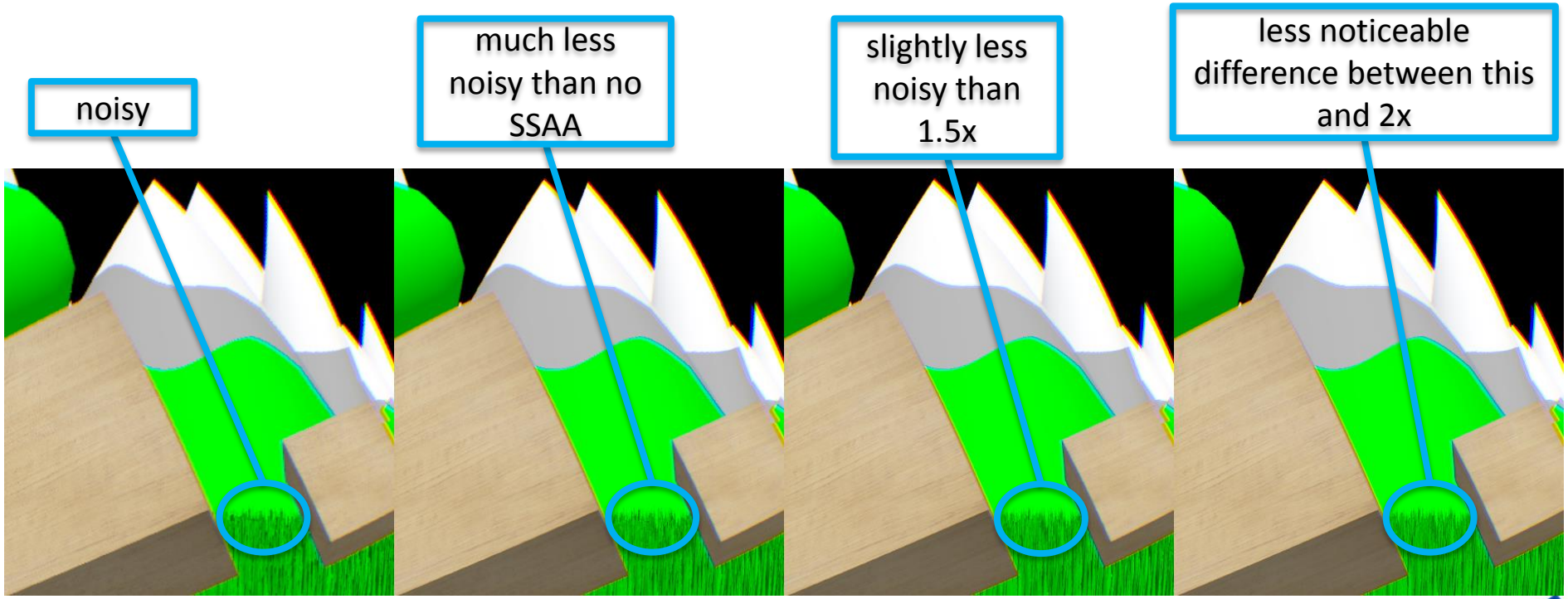
no anti-aliasing

FXAA

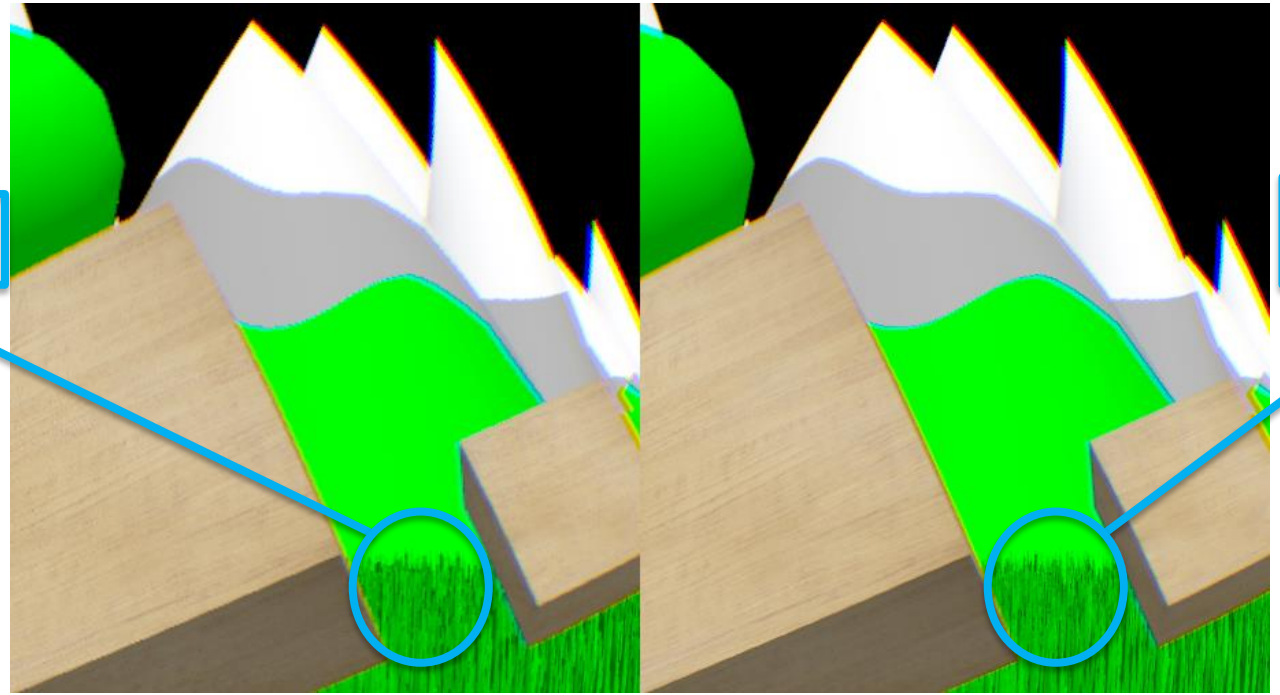
Supersample Anti-Aliasing

- Good results from rendering to a larger buffer…
- …If you can afford it
- Use with a good downsample filter

Supersample Anti-Aliasing



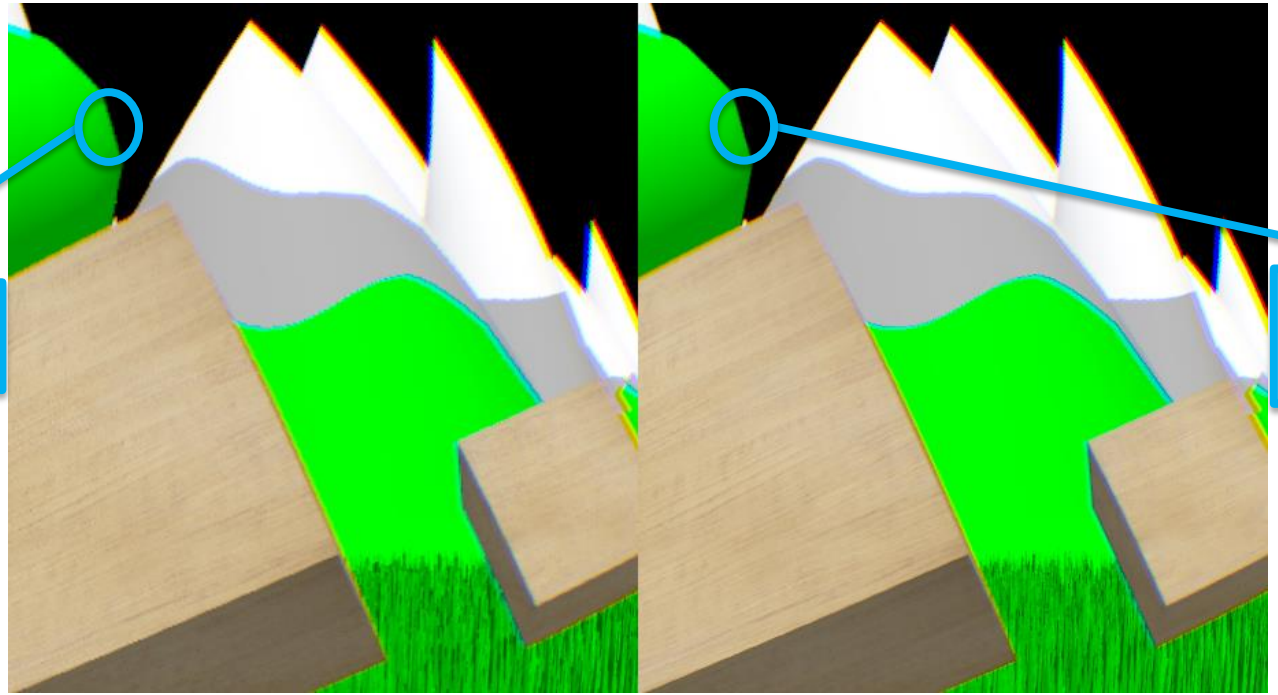
Supersample Anti-Aliasing



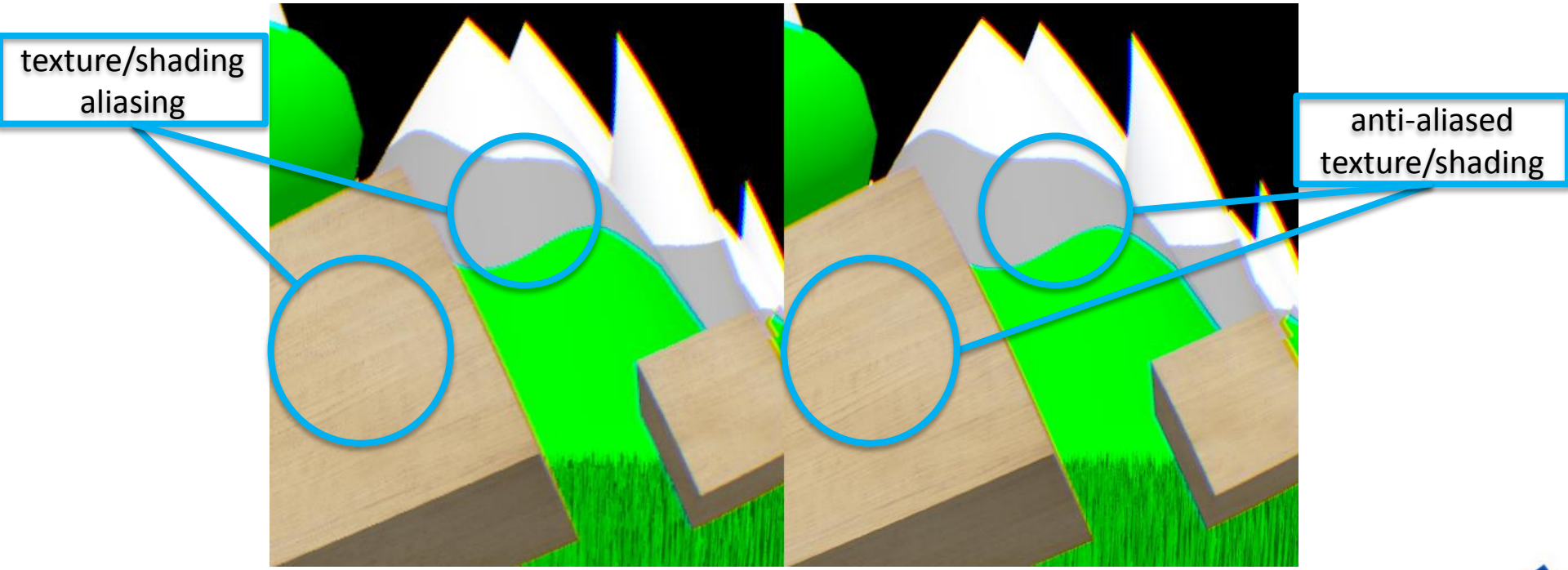
no SSAA

1.5x SSAA

Supersample Anti-Aliasing



Supersample Anti-Aliasing



Anti-Aliasing Findings

- Specular AA can also improve the image a lot
  - A great starting point is to look at LEAN, Cheap LEAN (CLEAN) and Toksvig AA[5]
- Distortion shader reduces edge aliasing
- In some games, more attention may need to be given to levels of detail

Anti-Aliasing Findings

SONY

- **A combination of several AA methods may give better results**
  - Each different AA solution combats different aspects of aliasing
  - Use whichever methods work best for your engine

High Frame Rate

High Frame Rate

- Why is a consistent, high frame rate essential?
  - Low frame rate looks and feels bad in VR
  - Testing is difficult without a high frame rate
    - Maintain high frame rate throughout development
  - Lack of V-sync is also much more noticeable
    - So make sure V-sync is enabled

High Frame Rate

- The concept of a “pass” is widely-accepted in current engines
  - Reflection rendering, shadow rendering, post-processing etc.
- Each pass has different requirements
- Always work out where the bottleneck is for each pass
  - CPU?
    - Draw calls?
    - State setting?
    - Resource setting?
  - GPU?
    - Vertex processing limited?
    - Geometry processing limited?
    - Pixel processing limited?

High Frame Rate

- CPU bound on draw calls, state setting or resource setting?
  - Consider how you can utilise the Geometry Shader
    - Can reduce total number of draw calls
      - Shadow cascade render: drawCallCount / n, where n is number of cascades
      - Cube map render: drawCallCount / 6
    - Reduces resource setting cost
    - It has other features that can help move processing off the CPU

High Frame Rate

- Geometry Shader
  - Converts a stream of primitives into another, possibly larger stream of primitives
  - Occurs before the pixel shader
    - i.e. after the vertex shader in a straightforward vertex-pixel draw call
    - After the hull-domain shaders if tessellation is enabled

High Frame Rate

- Geometry Shader Features
  - Render target index/viewport index
    - Use for single-pass cubemap rendering, shadow cascades, S3D etc.
  - GS Instancing
    - Allow multiple executions of the same geometry shader to be run per-primitive without the previous shader stage being run again

High Frame Rate

- Geometry Shaders for Stereo 3D Rendering
  - An easy method of making your engine stereoscopic 3D compatible
  - Add a GS to every material (or adjust the GS of existing materials that already have one) as follows:

GS for Stereo 3D Rendering

SONY

```
[maxvertexcount(3)]
void main(
     inout TriangleStream<GS_OUTPUT> triangleStream,
     triangle GS_INPUT input[3])

{
     for(uint i = 0; i < 3; ++i)
     {
          GS_OUTPUT output;
          output.position = (input[i].worldPosition , g_ViewProjectionMatrix);

          triangleStream.Append(output);
     }
}
```

GS for Stereo 3D Rendering

SONY

```
[instance(2)]
[maxvertexcount(3)]
void main(
      inout TriangleStream<GS_OUTPUT> triangleStream,
      triangle GS_INPUT input[3],
      uint gsInstanceId : SV_GSInstanceID)
{
      for(uint i = 0; i < 3; ++i)
      {
            GS_OUTPUT output;
            output.position = (input[i].worldPosition , g_ViewProjectionMatrix[gsInstanceId]);
            output.renderTargetId = gsInstanceId;
            triangleStream.Append(output);
      }
}
```

High Frame Rate

- Vertex/geometry bound?
  - Reduce your vertex size by compressing your attributes
  - Pack any attributes between shader stages
    - Important if geometry shaders for amplification or tessellation pipeline are being used
  - Consider using a late fetch method
    - Bind vertex attribute data as a buffer in the shader stage it is used
    - Highly dependent on hardware
    - Always profile to see if it makes a difference!
  - **Reduce the data being moved around the GPU**

High Frame Rate

- Pixel bound?
  - Reduce the complexity of pixel shaders
  - Reduce the number of pixels shaded every frame
    - An experiment using a smaller render target with upsampling conflicted with high-quality visuals, introducing haloing, shimmering and retinal rivalry

High Frame Rate

- Consider using re-projection to speed up aspects of stereo 3D rendering[7]
  - Used with great success on PlayStation®3 stereo 3D games
  - However, it can only be used successfully where parallax is small

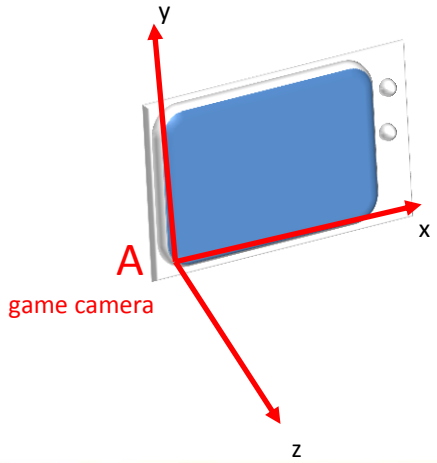
Great Tracking and Calibration

Great Tracking and Calibration

- The Project Morpheus SDK handles tracking
- Use the tracking matrices as supplied by the SDK

Tracking Matrices

existing game



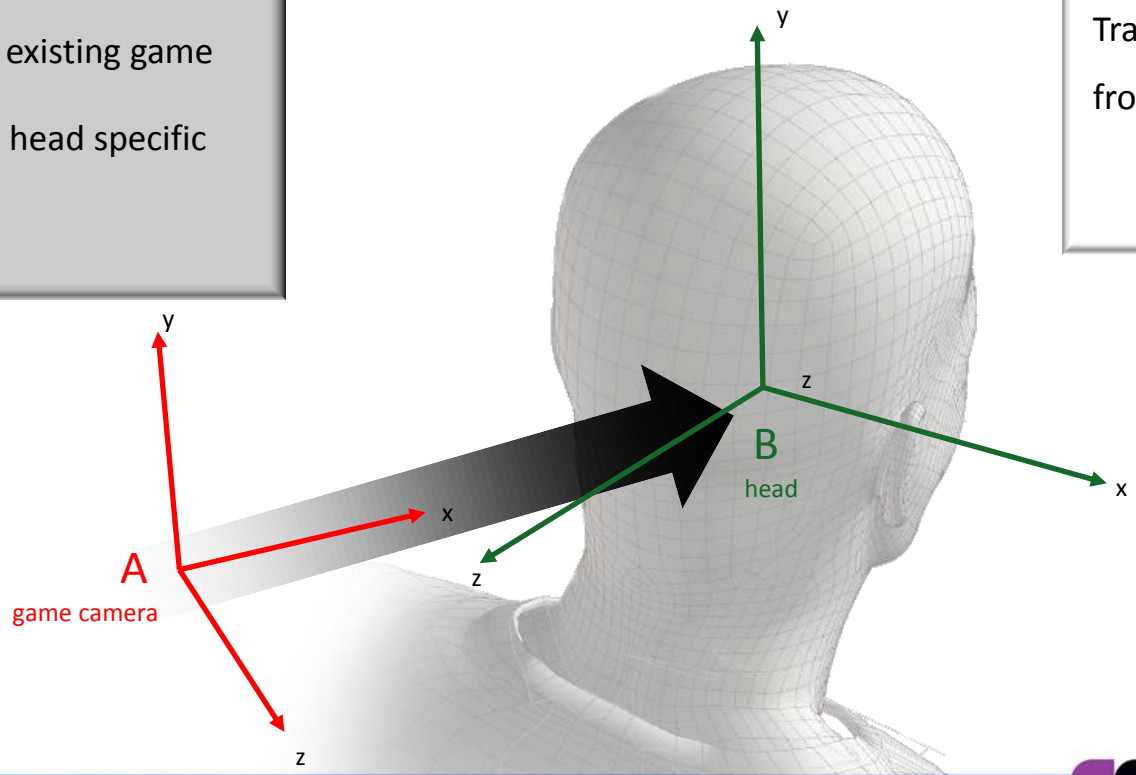
Default viewing position and orientation defined by the game

Tracking Matrices

existing game

head specific

Tracked offset of the players head from the camera



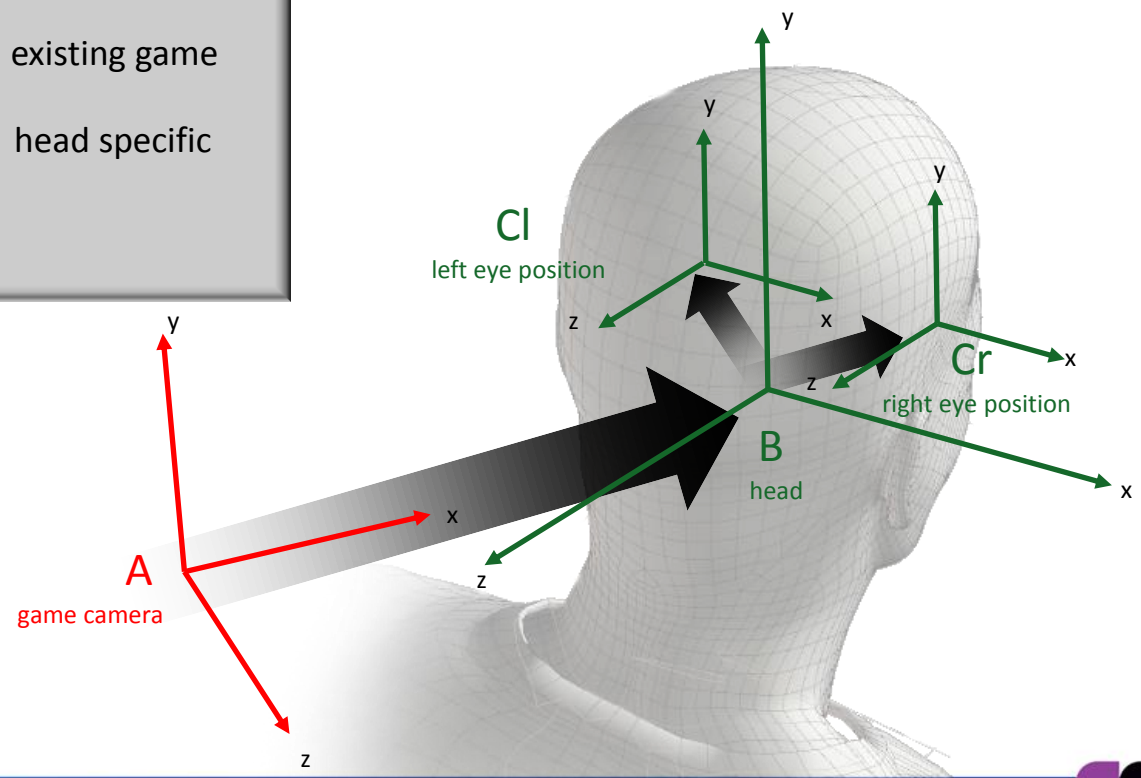
A x B

Multiplication

Tracking Matrices

existing game

head specific



Offset of the player's eyes relative to the head matrix

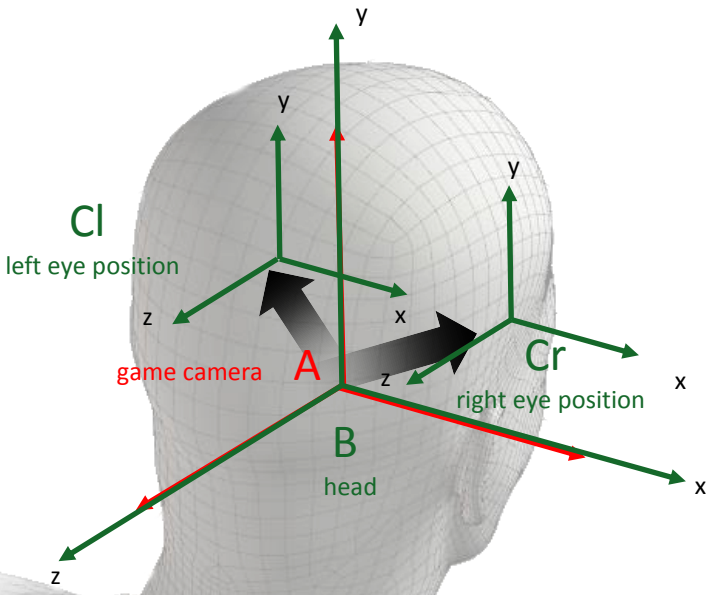
A x B x C

Multiplication

Tracking Matrices

existing game

head specific



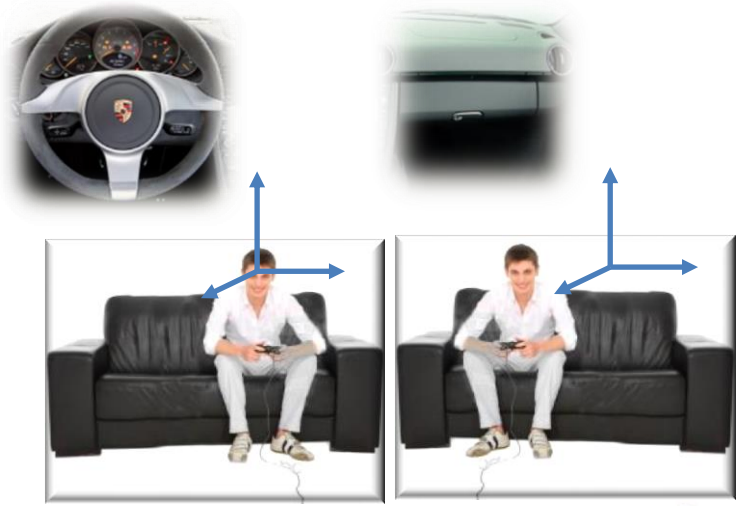
Tracker reset function: sets the head position to line up with the game camera in position and yaw.

A x B x C

Multiplication

Calibration

- Reset positional and orientation tracking
- Re-align the game world with the real world for:
  - Fixed playing positions
  - Pass-and-play
  - Players of different heights



Low Latency

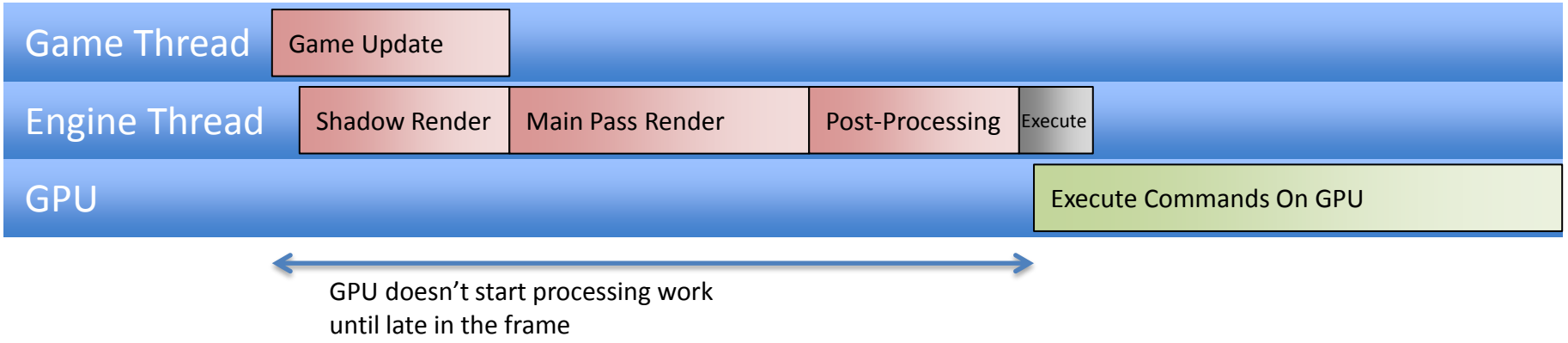
Latency

- Why is the reduction of latency so important?
  - Latency is the time interval between an input and the response
  - It isn't just a consistent, high frame rate that's important
- Not only for VR head-tracking; increasing responsiveness is well understood in gaming
  - Gameplay programmers understand the need for responsive controls
  - Network programmers understand the need for responsive opponents
  - etc.

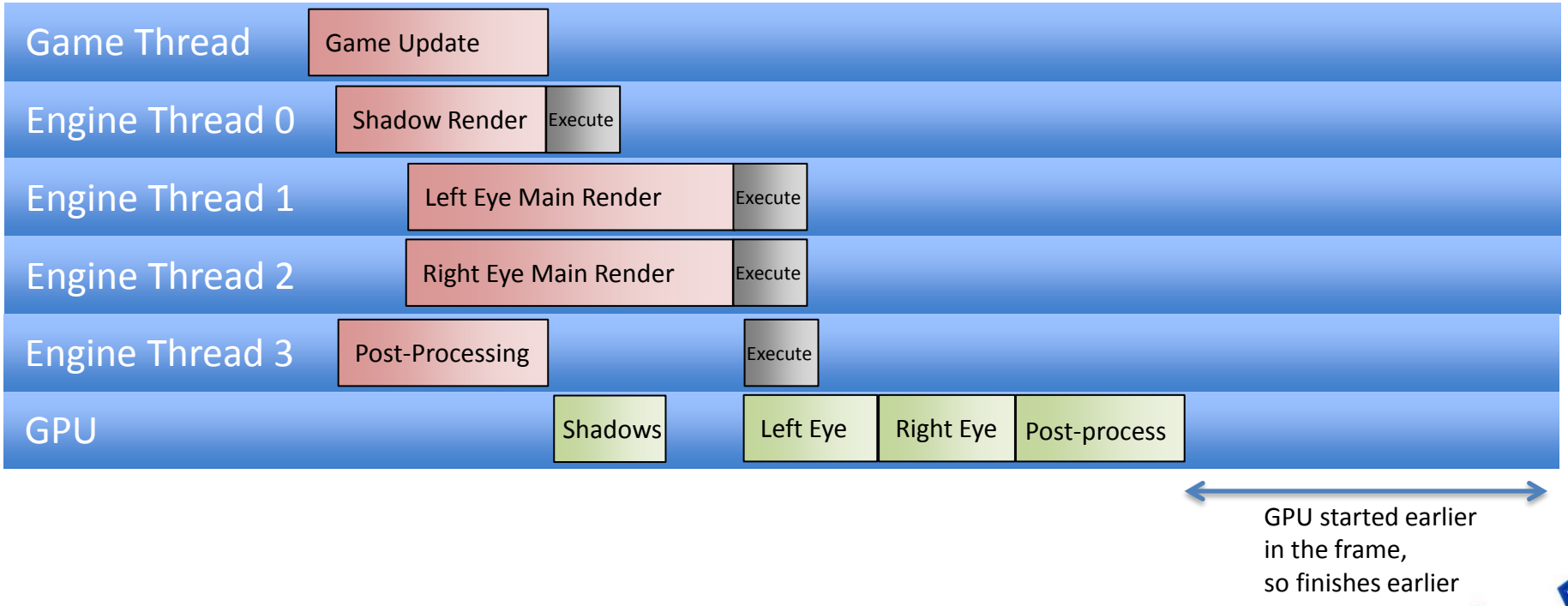
Multi-Context Rendering

- One way to reduce latency from an engine perspective is to use deferred contexts to build command lists (AKA command buffers) on multiple threads asynchronously
  - As an immediate context “generates rendering overhead when it queues up commands in the command buffer. In contrast, a command list executes much more efficiently during playback.”[6]
  - Works well with the concept of a “pass”
- **Multi-context rendering reduces latency by allowing the GPU to start processing earlier in the frame**

Single-Context Rendering



Multi-Context Rendering



Multi-Context Rendering

SONY

- Simplest test case
  - Built and submitted command lists for each eye's view in parallel
  - Immediate reduction in CPU frame time
  - If the engine is CPU-bound, this translates as an immediate reduction in frame latency
  - If the engine is GPU-bound but the GPU is now finishing earlier in the frame because it started earlier, this translates as an immediate reduction in frame latency

Latency Considerations for VR

- Are there any VR-specific ways to combat latency?
  - The time between sampling the tracking data and using that data to render a frame needs to be as low as possible
  - Don’t use more than double buffering
  - Re-projecting the image with the latest orientation data can improve the apparent latency and frame rate
- Keep tracked peripherals’ latency as low as possible too
- Are there any platform-specific ways to combat latency?

Prediction

- PlayStation®4 with Project Morpheus is a known system
  - We know any latencies in the hardware
  - We know any latencies in the libraries/software
  - We're continuing to reduce these latencies all the time
  - We give you performance analysis tools for CPU and GPU that enable you to calculate and reduce the latency in your game
- We can use this to predict where the HMU will be by the time the image gets displayed

Prediction

- Reducing latency in your engine is key
  - But using prediction to mask any small remaining latencies can work well
  - The smaller the amount of prediction you specify, the better its quality

What's Next?

What's Next?

- So now you have great tracking in an incredibly efficient, high-frame rate, low-latency, super-high-quality next-gen engine that's optimised for virtual reality…your job's done, isn't it?
- Of course not!
  - More platform-specific optimisation
  - Tracked peripherals
  - Social aspects
  - Gameplay/design elements

Asynchronous Compute

- Still CPU-bound?
  - Maybe Compute can help offload parallelisable tasks to the GPU
- Still GPU-bound?
  - Compute allows you to think about GPU tasks from a different, more generic perspective
- Use it where your GPU is not being fully utilised
  - Shadow rendering is usually vertex/geometry-heavy so it's a good place to schedule async Compute tasks

Tracked Peripherals

- DUALSHOCK®4
  - Motion sensors
  - Light bar
- PlayStation®Move
- Multiple devices and multiple users

Social

- Social Screen
  - Asymmetric gameplay
  - Tracked controllers
- Companion Apps
- Online multiplayer
  - Head-tracked opponents

REBOOTING GAME DESIGN FOR VIRTUAL REALITY

Offenbachsaal, 1st Level
Tuesday, August 12, 10:00-11:00

Format: Lecture
Track: Design
Pass Type: All Access Pass, Student Pass

Jed Ashforth
Senior Game Designer, WWS Immersive Technology Group
Sony Computer Entertainment Europe

GAME DEVELOPERS CONFERENCE™ EUROPE
CONGRESS-CENTRUM OST KOELNMESSE · COLOGNE, GERMANY
AUGUST 11–13, 2014 · EXPO: AUGUST 11–12, 2014

Tools and Middleware


THIS IS FOR
THE CREATORS
PS4™

Thanks

- Sony Computer Entertainment Europe
  - Simon Benson, Immersive Technology Group
  - Sharwin Raghoebardayal, World Wide Studios
  - Domenico Troiano, Research & Development

References

- [1] “FXAA” Nvidia whitepaper by Timothy Lottes – http://developer.download.nvidia.com/assets/gamedev/files/sdk/11/FXAA_WhitePaper.pdf
- [2] “Practical Morphological Anti-Aliasing” – Jorge Jimenez, Belen Masia, Jose I. Echevarria, Fernando Navarro, Diego Gutierrez – http://www.iryoku.com/mlaa/
- [3] “SMAA: Enhanced Subpixel Morphological Anti-Aliasing” – Jorge Jimenez, Jose I. Echevarria, Tiago Sousa, Diego Guierrez – http://www.iryoku.com/smaa/
- [4] “Filtering Approaches for Real-Time Anti-Aliasing” – SIGGRAPH 2011 course – http://iryoku.com/aacourse/
- [5] “Specular Showdown in the Wild West” by Stephen Hill (Self Shadow) – http://blog.selfshadow.com/2011/07/22/specular-showdown/
- [6] “Immediate and Deferred Rendering” – MSDN Direct3D 11 Graphics documentation – http://msdn.microsoft.com/en-gb/library/windows/desktop/ff476892(v=vs.85).aspx
- [7] “Optimization for Making Stereoscopic 3D Games on PlayStation® (PS3™)” – Sebastien Schertenleib, Sony Computer Entertainment Europe R&D – http://develop.scee.net/files/presentations/nordic/OptimizationforMakingStereoscopic3DGamesonPlayStationPS3.pdf