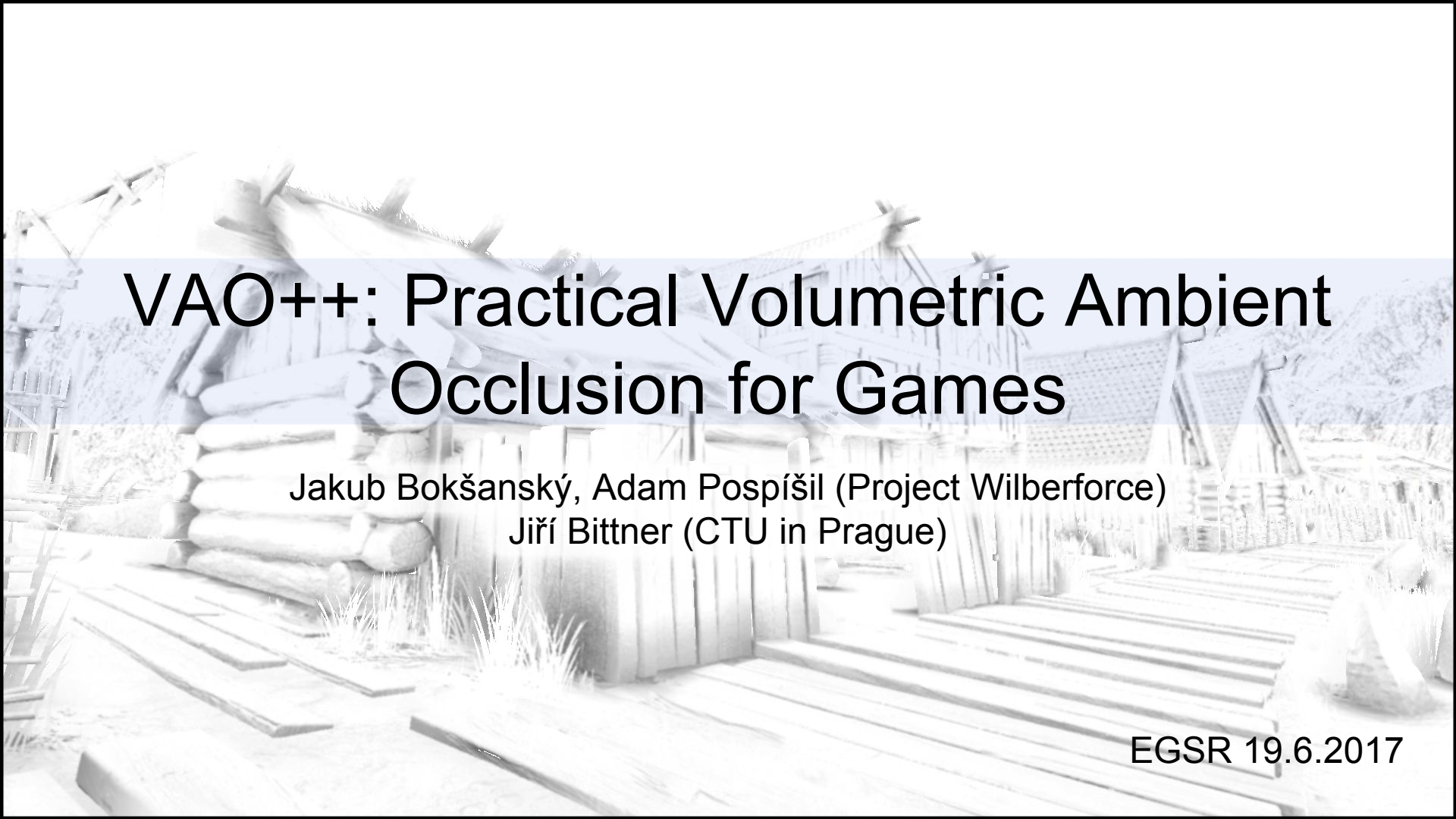


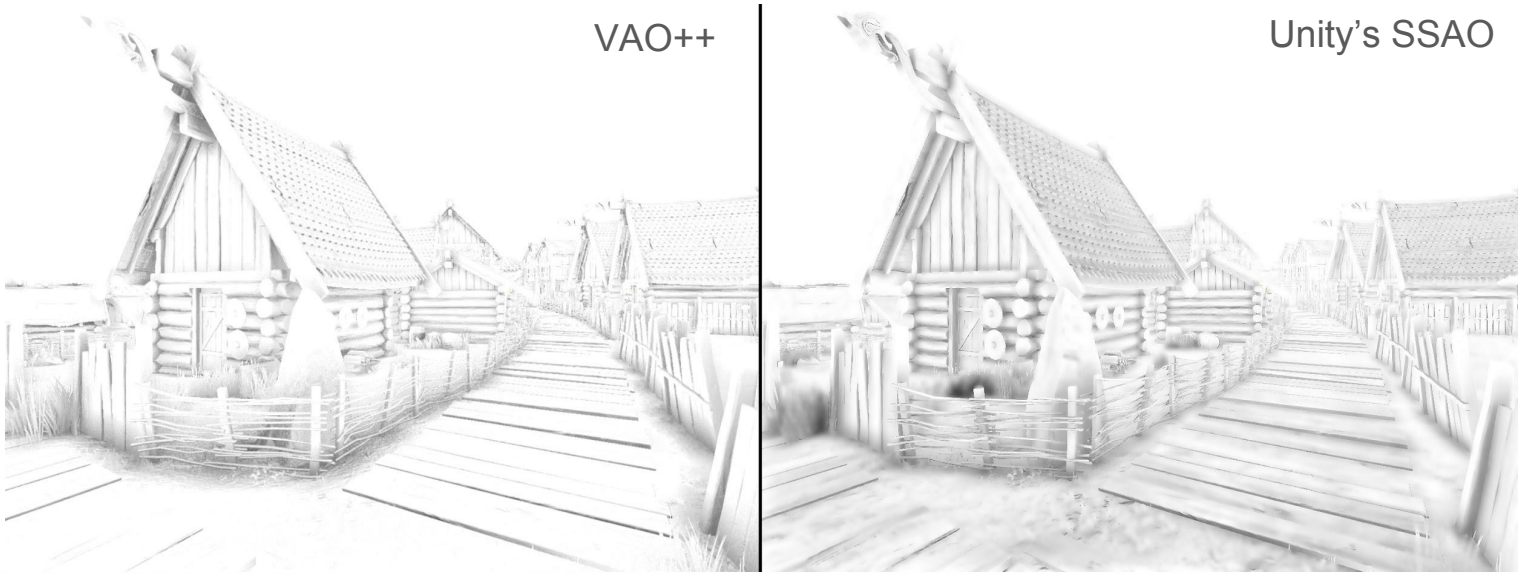
VAO++: Practical Volumetric Ambient Occlusion for Games

Jakub Bokšanský, Adam Pospíšil (Project Wilberforce)
Jiří Bittner (CTU in Prague)

EGSR 19.6.2017

Motivation

- Focus on performance (highly optimized implementation in Unity)
- Unity's built-in SSAO not fast enough and contains artifacts
- VR compatibility (single pass stereo, explicit stereo support)



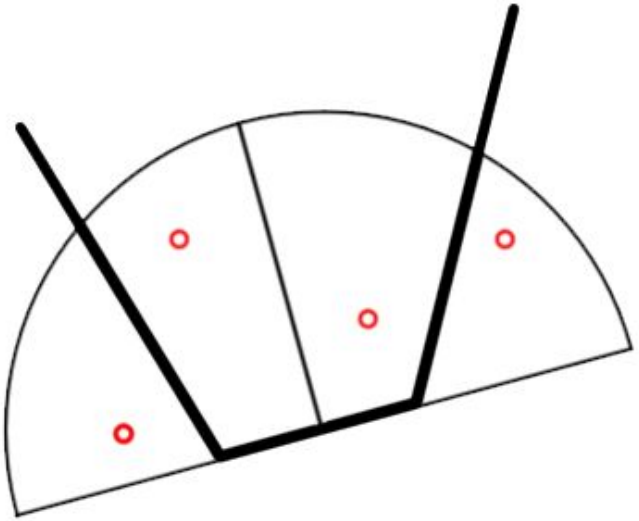
SSAO

- Adds proximity shadows (enclosed areas, creases, corners…)
- Contributes significantly to perceived realism
- Calculated in screen space
- Limitations:
  - No information about geometry what is not seen
- Advantages
  - Feasible in real time on current GPUs
  - Easy integration with renderers
  - No pre-processing needed
- [Mittring07], [Bavoil08], [Ritschel09]

Scene without effects VAO++

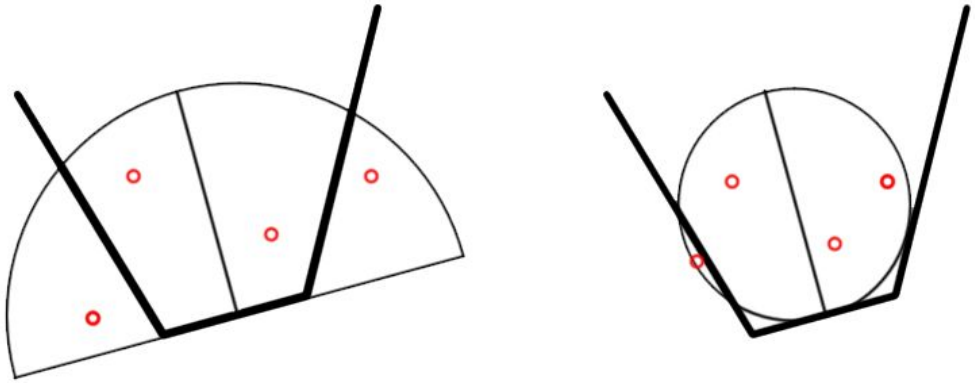
SSAO Implementation

- For each shaded point, calculate visibility (how much is it obscured by surrounding geometry)
- Surrounding geometry is sampled in normal oriented hemisphere of radius R



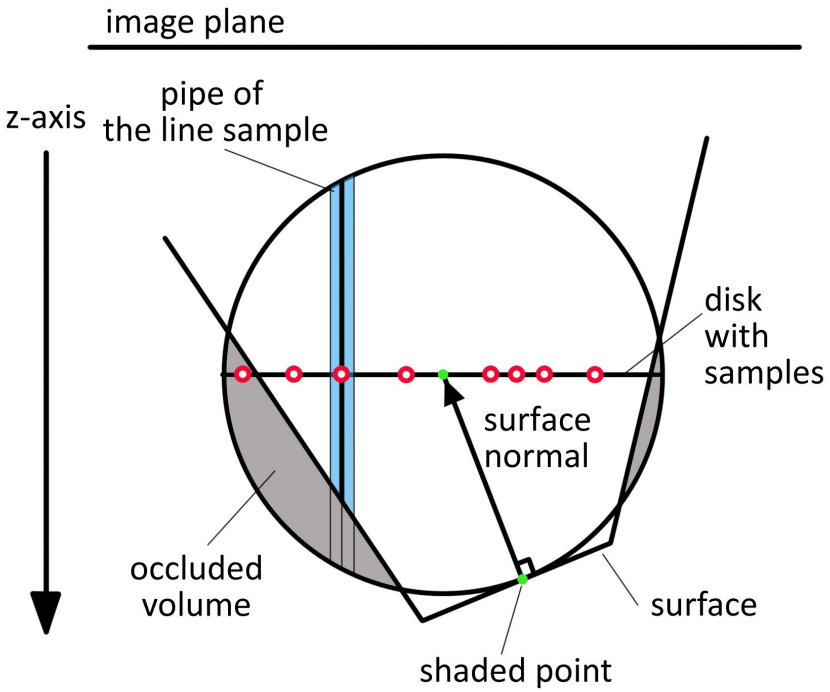
Volumetric Ambient Occlusion [Szirmay-Kalos10]

- Transforms hemispherical domain into smaller tangent sphere
  - Uses samples within a disc located in the center of tangent sphere oriented towards camera (2D samples instead of 3D)
  - Each sample represents a line segment inside the sphere in given sampling point
    - Length of the line segment is taken as weight of the sample
  - Samples never overlap



Volumetric Ambient Occlusion II.

- Algorithm calculates where geometry lies wrt. tangent sphere
  - A - Inside (see image)
  - B - Behind (completely unoccluded)
  - C - In front of (completely occluded)
- Visibility (occlusion factor) is calculated as ratio of unoccluded volume to total volume of the tangent sphere

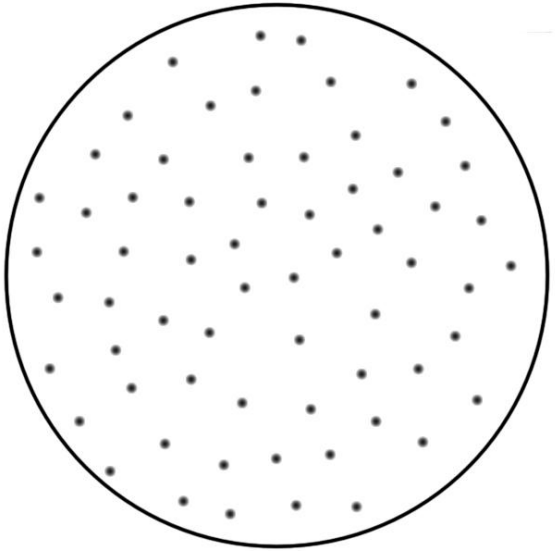


VAO++

- Extensions and improvements to VAO algorithm
- Improvements of visual quality
  - Samples generation
  - Luminance sensitivity
  - Depth range check
- Performance optimizations
  - Adaptive sampling
  - Culling pre-pass

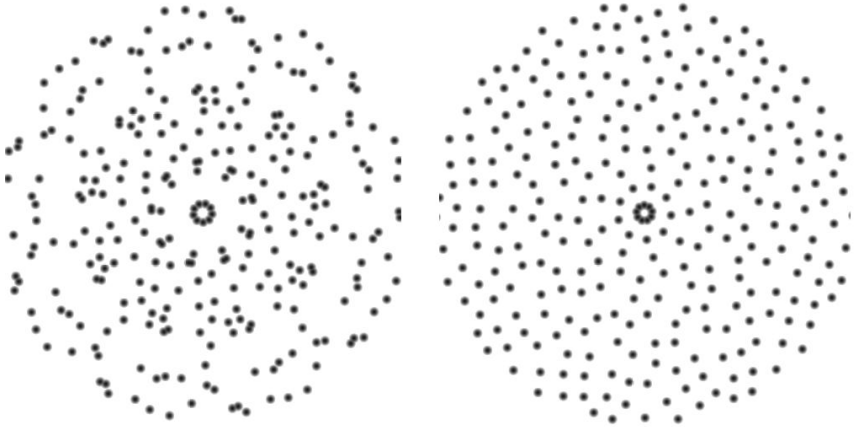
Samples generation

- VAO algorithm requires uniformly distributed samples within a disk
- Poisson distribution is suitable
- Interleaved sampling
  - Size 3x3 - good improvement in quality (number of samples increased virtually 9 times)
    - easy to remove HF noise by low pass filter of matching size
  - Problem with selecting suitable rotation angles
  - Points should not overlap after rotations (prevent sampling of same area)



Samples generation - selecting rotations

- Selecting rotations after generating Poisson distribution yields suboptimal results
  - Goal is to select angles that maximize the distance between positions of rotated samples



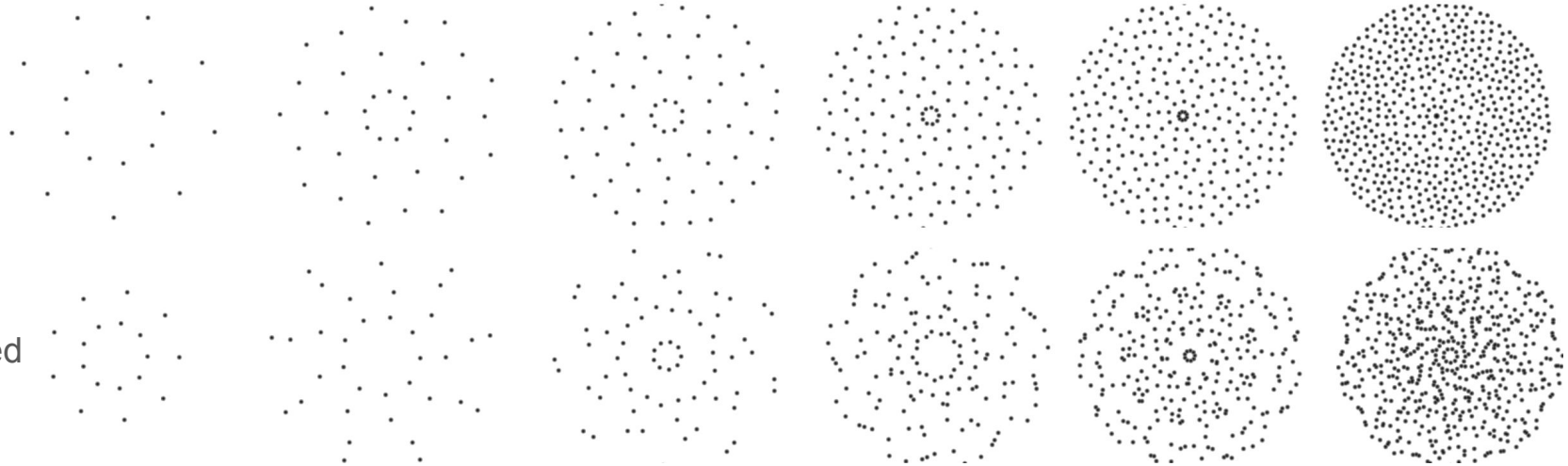
(left) rotations selected after generating sample-set. (right) our method

Samples generation - selecting rotations II.

- Take rotations into account when generating Poisson disk distribution
  - We use uniform-step 9 rotations
- Union of sample-set rotated by all angles satisfies Poisson disk distribution requirements

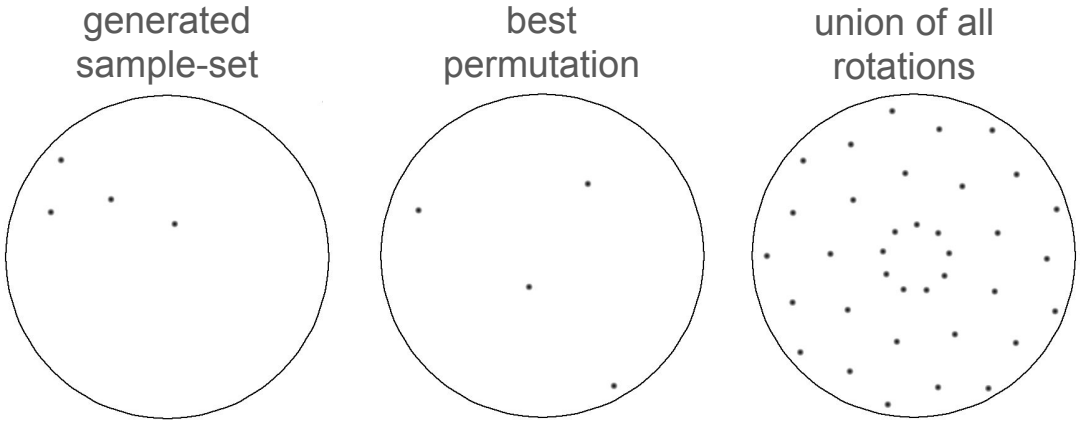
optimized

non-optimized

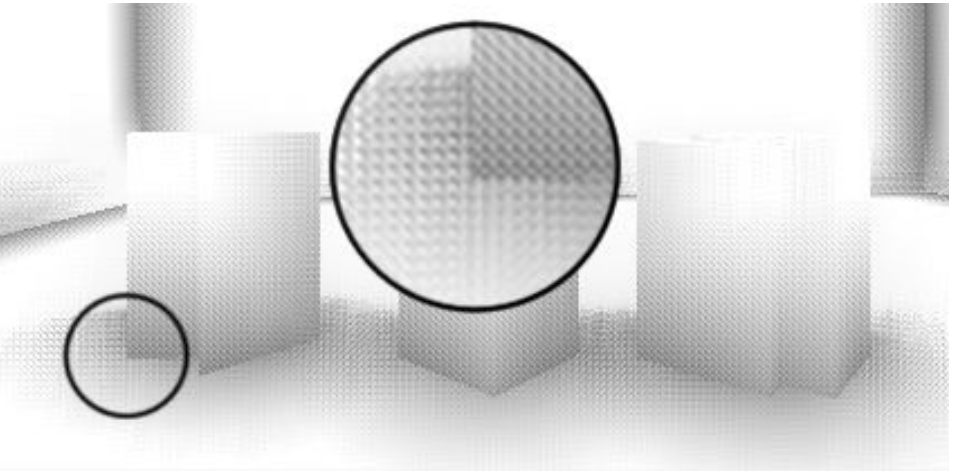


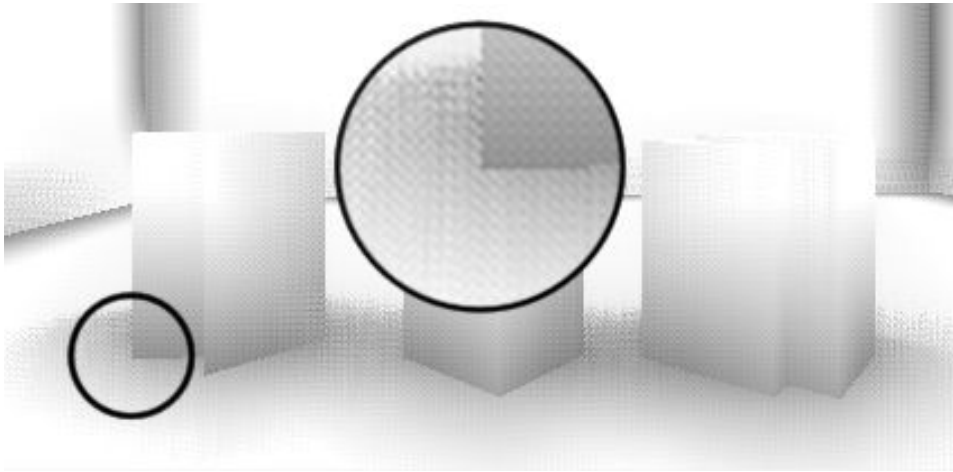
Samples generation - selecting rotations III.

- Final step is to select best permutation of rotated samples
  - Maximize distance between all samples
  - Feasible for 8 samples or less, for larger sample-sets we calculate best permutation for groups of size 8…
- Morton-Ordered for better cache coherence



Samples generation - selecting rotations IV.





Comparison of effect results with sample-set non-optimized (left) vs optimized (right)
Loupe contains enlarged area

Luminance sensitivity I.

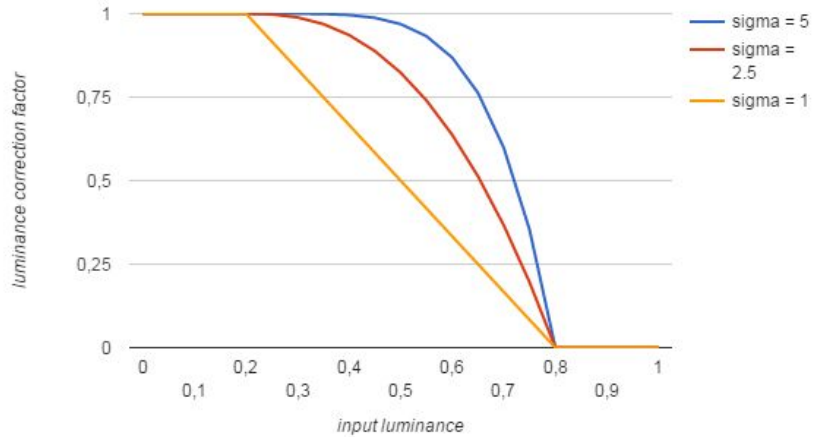
- SSAO should be applied on ambient color component before lighting (BRDF) is calculated - not always possible or practical
- VAO++ applies occlusion as final step to rendered image
  - can put occlusion on brightly lit areas, specular highlights, light sources etc.

Scene without effects

VAO++ applied

Luminance sensitivity II.

- To counteract this defect consider luminance of target surface when applying VAO component
- Falloff function to suppress occlusion on bright surfaces
- User can set brightness threshold and slope

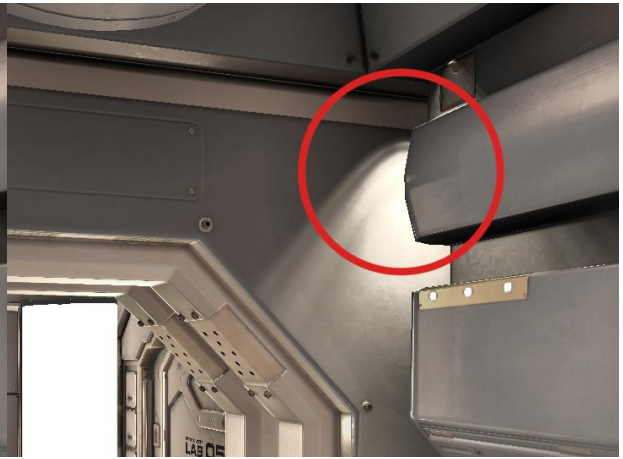


Luminance sensitivity III.

- As a luminance use either relative luminance function or Value component of HSV model

Scene without effects

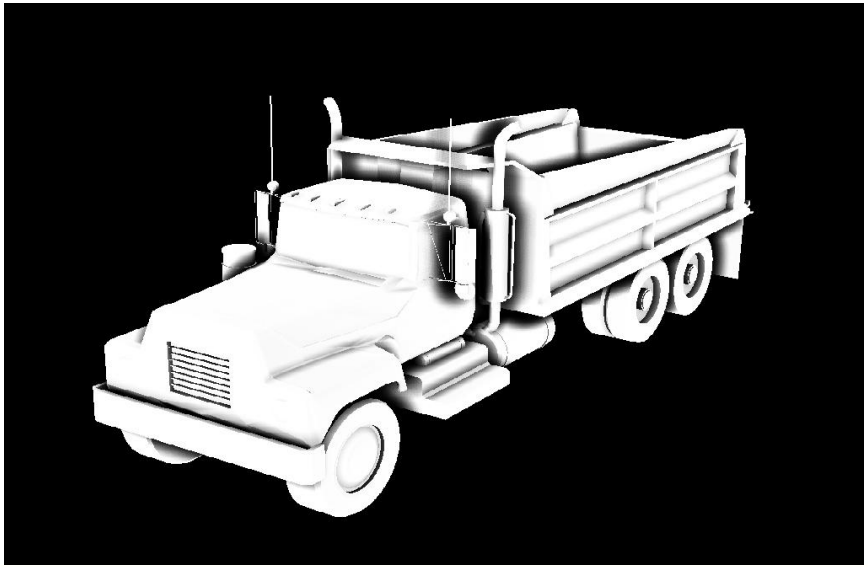
VAO++



VAO++ with luminance sensitivity

Depth Range Check

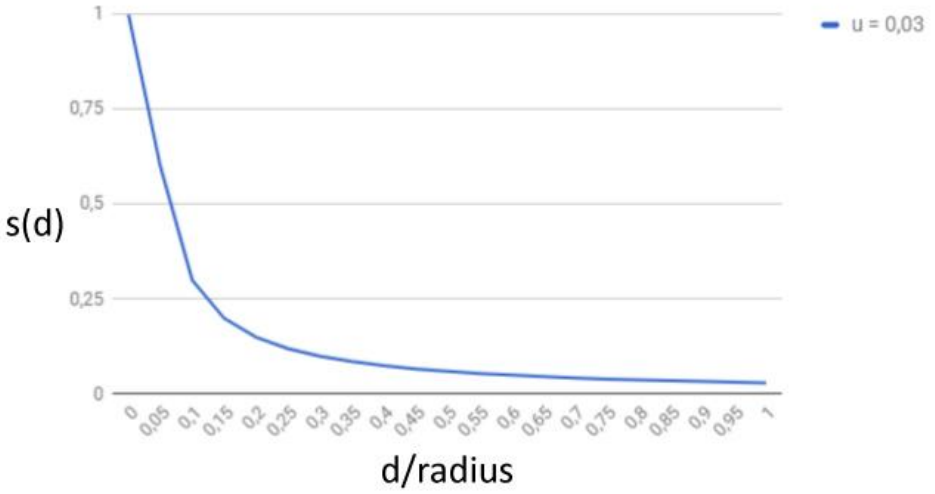
- No geometry information behind what is seen by camera
- Thickness of occluders unknown
  - VAO considers all occluders infinitely thick - leading to unpleasant halo artifact



Depth Range Check II.

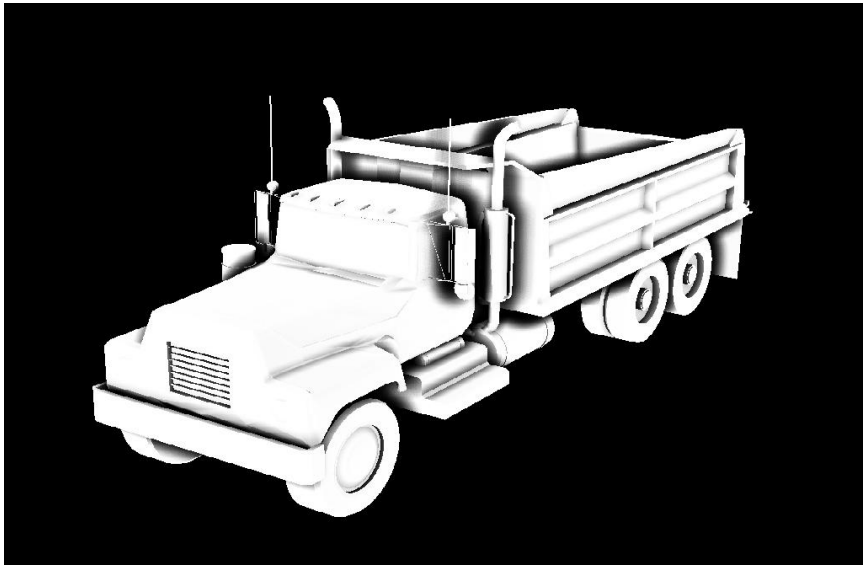
- Implement a falloff function describing thickness of geometry seen by camera
  - smoothly decreases thickness with increasing distance
- Resulting occlusion also smoothly decreases around depth discontinuities
- This suppresses halo significantly and works well for dynamic scenes

$$s(d) = \frac{u}{max(u, \frac{d}{radius})}$$

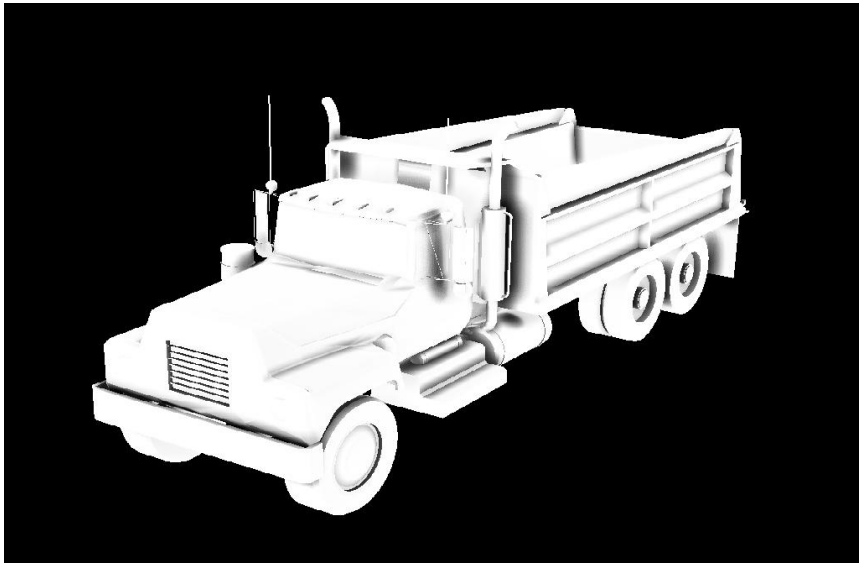


Depth Range Check III.

without depth range check



with depth range check



VAO++

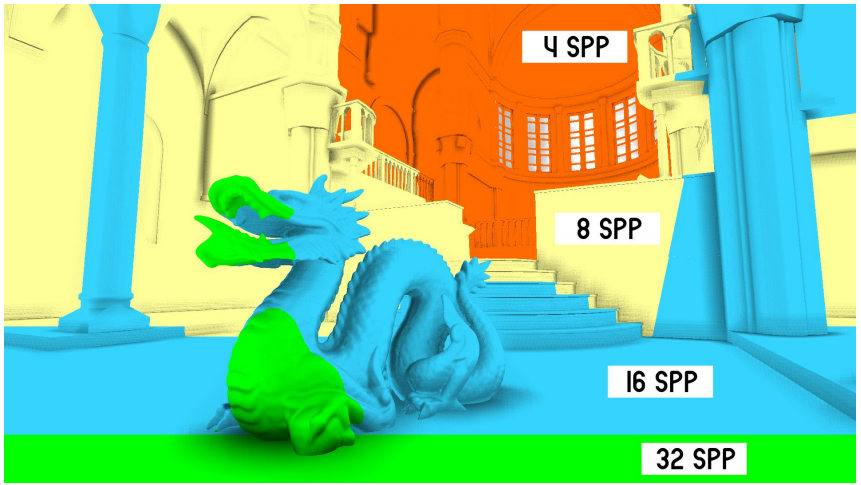
- Extensions and improvements to VAO algorithm
- Improvements of visual quality
  - Samples generation
  - Luminance sensitivity
  - Depth range check
- Performance optimizations
  - Adaptive sampling
  - Culling pre-pass

Adaptive Sampling

- To improve performance use only as many samples as needed
- VAO uses kernel that always displays as a disk on the screen
  - This disk surface shrinks with increasing distance from camera
  - Use fewer samples far away
- Aim is to achieve same density of samples in post-perspective image space

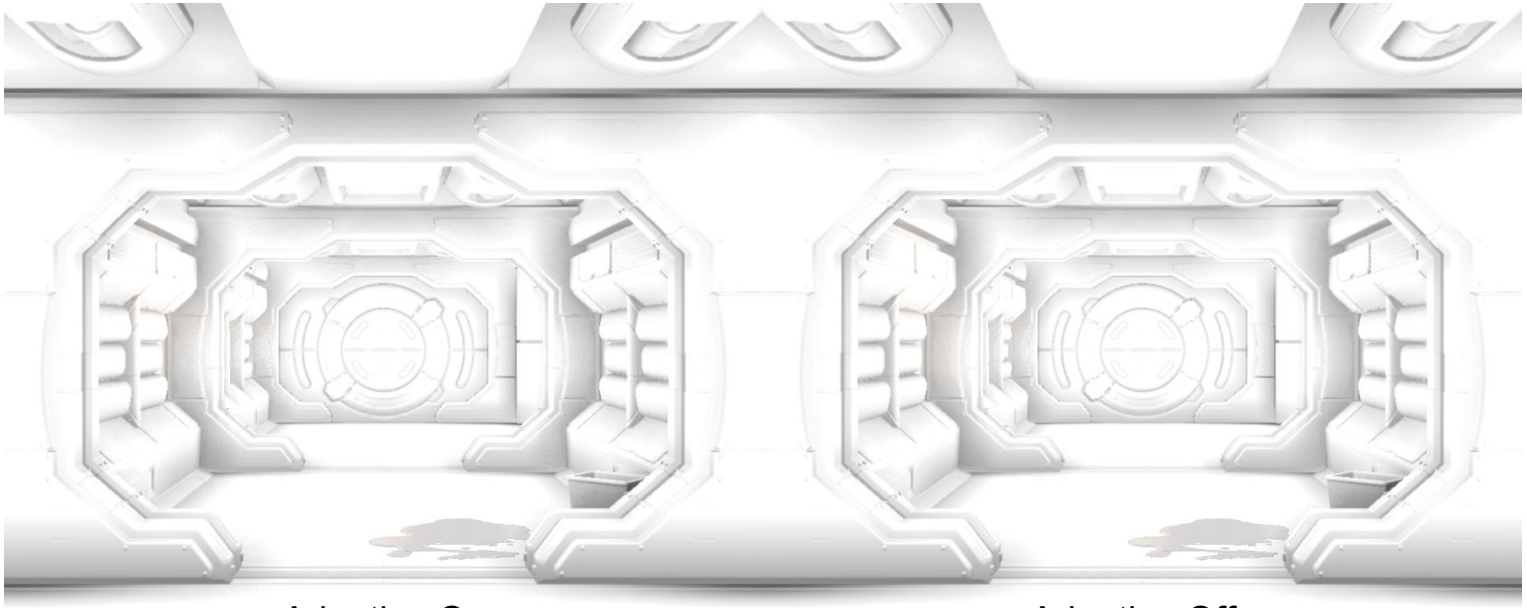
Adaptive Sampling II.

- Pick number of samples to use based on kernel disk surface area after projection onto the screen
- User selects surface area for which lowest number of samples is used
- Algorithm doubles number of samples used when surface area doubles



Adaptive Sampling III.

- Comparison of adaptive sampling on/off



Adaptive On

Adaptive Off

Culling pre-pass

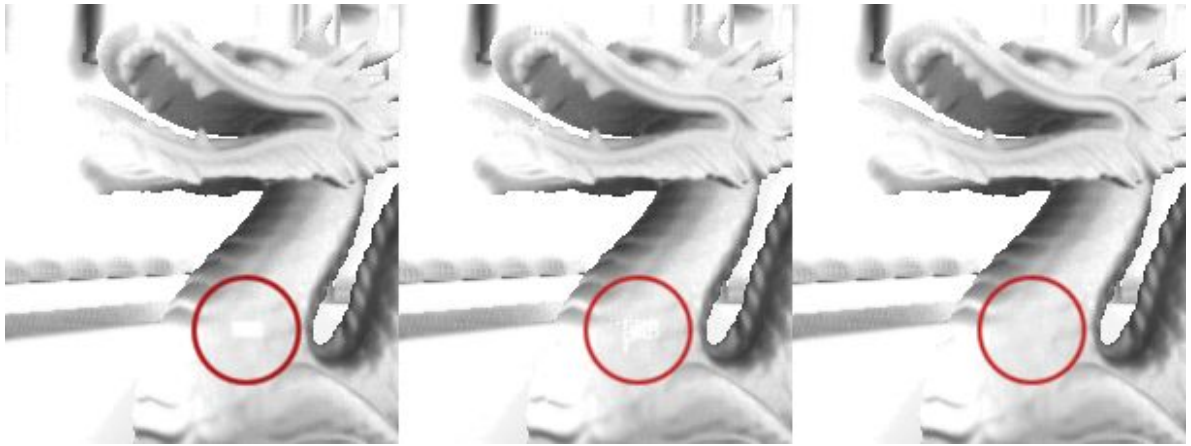
- Areas where no occlusion occurs can be omitted
- Estimate such areas first, calculate VAO only where needed
- Calculate lower quality VAO in a separate pass
- Use result as an estimation

Occlusion estimation texture

Rendered scene

Culling pre-pass II.

- Areas without estimated occlusion could contain undetected details
- Handle culling candidates in one of following ways:
  - Greedy variant - complete omission
  - Careful variant - eight samples regardless of quality settings

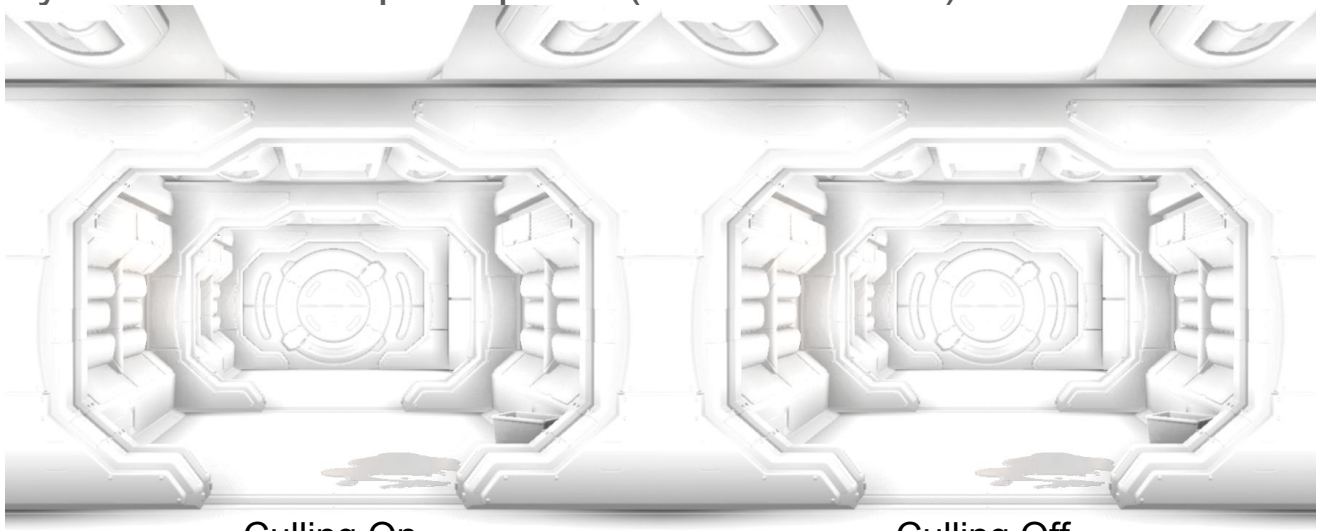


Greedy Careful Off

Images are
8x zoomed
and gamma corrected

Culling pre-pass III.

- Performance gain scales with resolution
- Especially useful for large screens (4k, VR, etc.)
- Quality loss almost imperceptible (careful variant)



Culling On

Culling Off

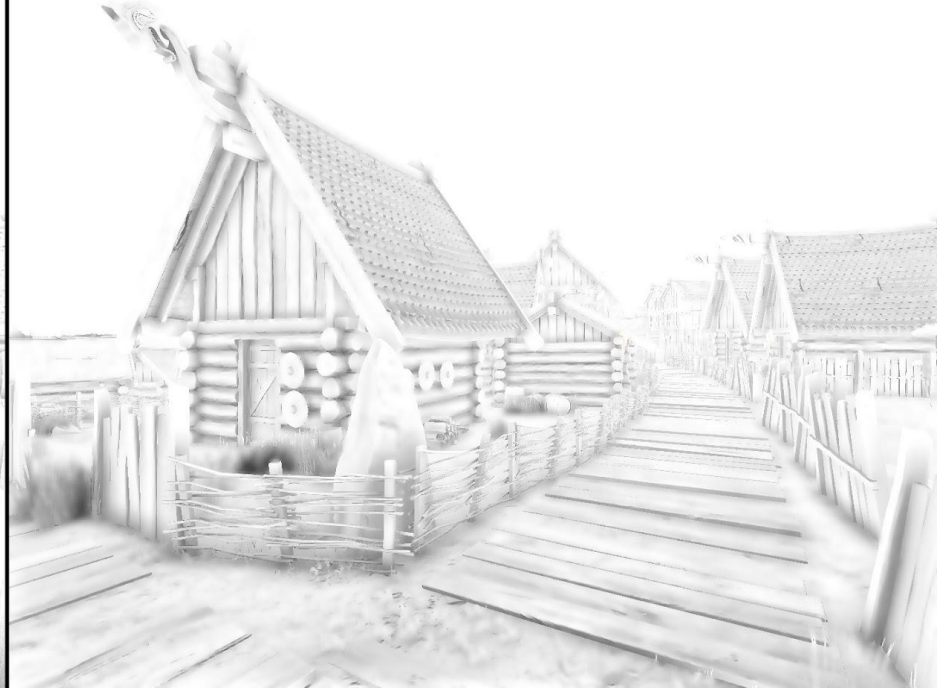
Implementation in Unity

- Image effect attached to camera
  - Inject into pipeline - after opaque rendering pass
  - Read source texture (rendering pipeline output)
  - Write into destination texture
- C# Script + CG/HLSL shader file
- Alternative Command Buffer implementation
- Unity provides depth and normals texture
  - Octa encoded normals + linearized depth packed into 32 bits
- Single Pass Stereo Rendering (VR)
  - Both eyes in single texture
  - Special macros for handling split between eyes
  - Need to calculate effect in world-space

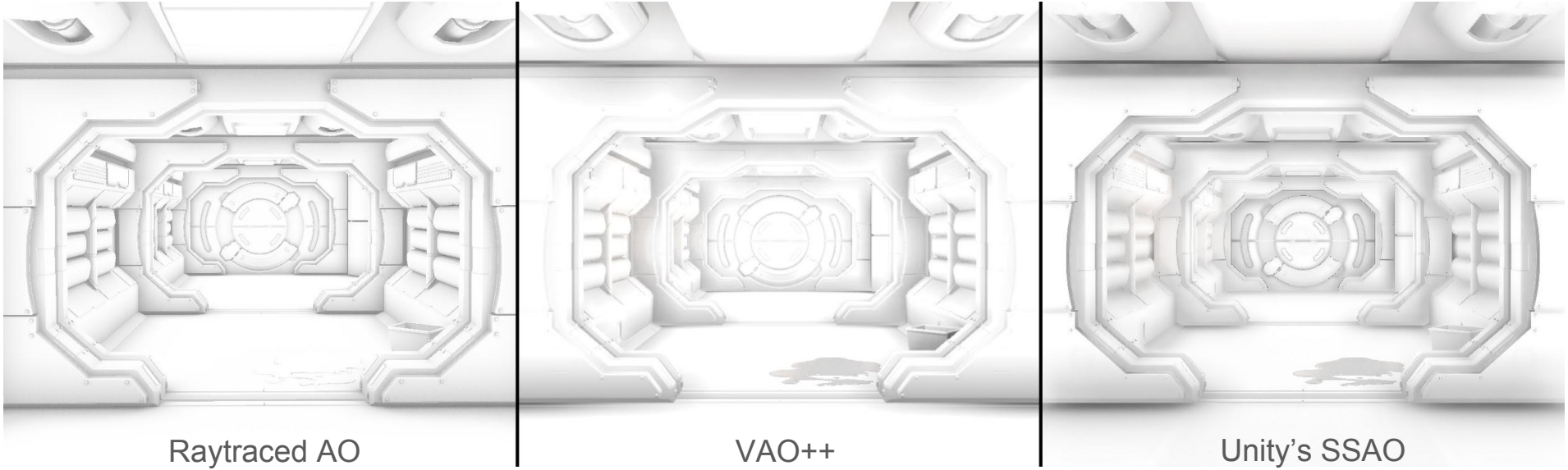
Comparison - 16 SPP, 1920x1080, GeForce GT555M, same radius

VAO++ (48ms) - Speedup 1,25x

Unity's SSAO (60ms)



Comparison



Raytraced AO | VAO++ | Unity's SSAO

Comparison

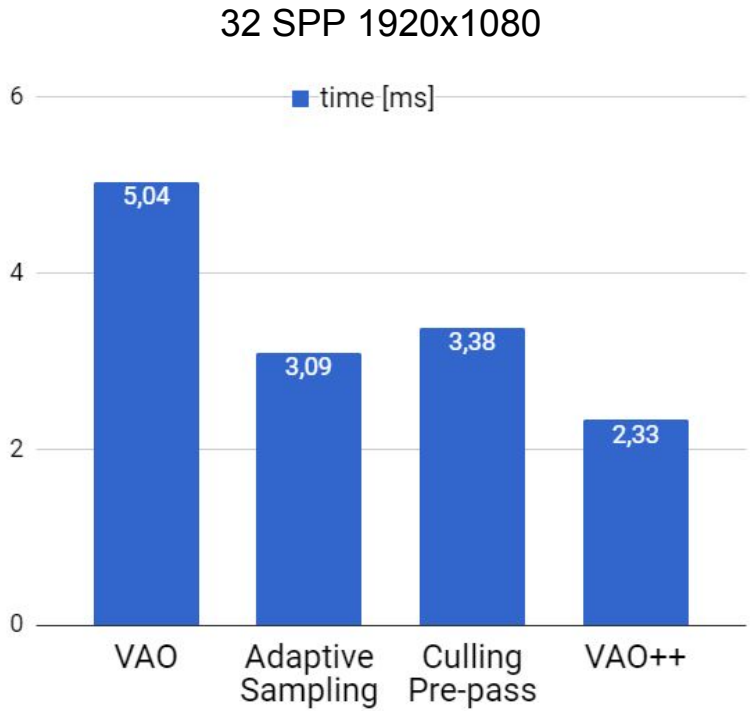
Effect Off

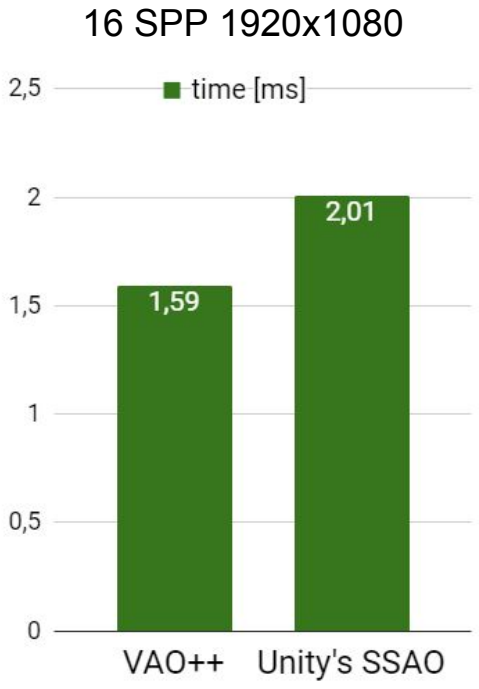
Comparison

Effect On

Comparison - Video

Performance results





Conclusion

- Samples generation taking rotations for interleaved sampling into account
- Luminance sensitivity and depth range check for more believable results
- Adaptive sampling and culling pre-pass for higher performance with minimal quality loss
- Optimized Unity implementation (available on Asset Store)

Scene by Evgenia

- **Project Wilberforce**
- projectwilberforce.github.io
- projectwilberforce@gmail.com
- Demo available at: https://projectwilberforce.github.io/vaodemo/

Thank you!