Ambient Occlusion in 4D

Maria R. Lily Djami
Bachelor's thesis
Heidelberg, Germany, July 19, 2021

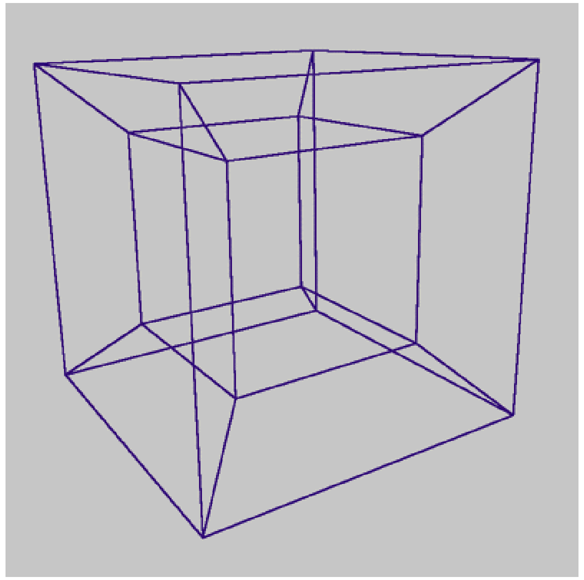
Outline

- Introduction
- Methods
- Implementation
- Results & Discussion
- Conclusion

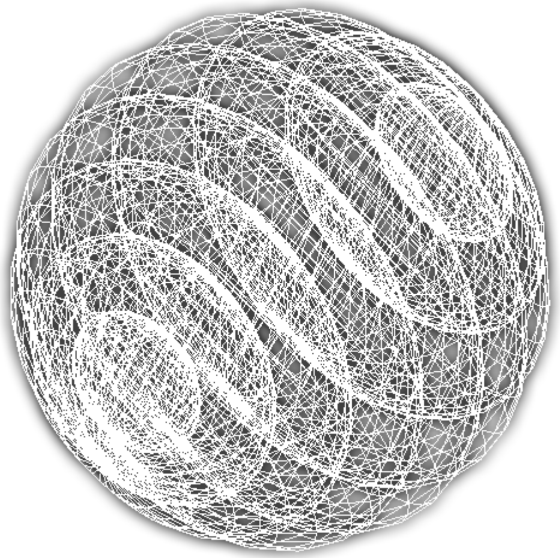
Introduction

Motivation:

- 4D structures are difficult to perceive
- Improve perception of 4D structures using illumination



4-cube represented as a 2D wireframe image



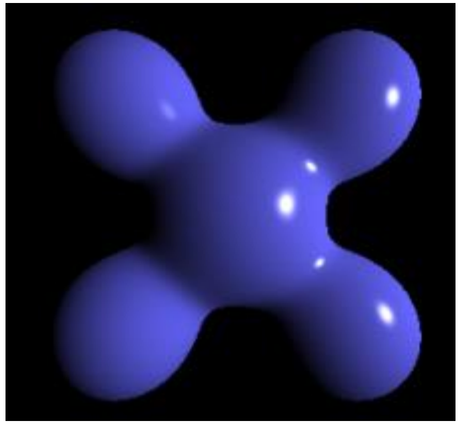
Direct projection of hypersphere in 4D into 3D space, rendered as a 2D image

Source: HOLLASCH, Steven R., 1991. Four-Space Visualization of 4D Objects. MA thesis. Arizona State University; Antipov, Eugene. (7 September 2006). 2D-projection of 3D-projection of Hypersphere of 4D-space. Retrieved July 2, 2021 from https://en.wikipedia.org/wiki/File:Hypersphere.png

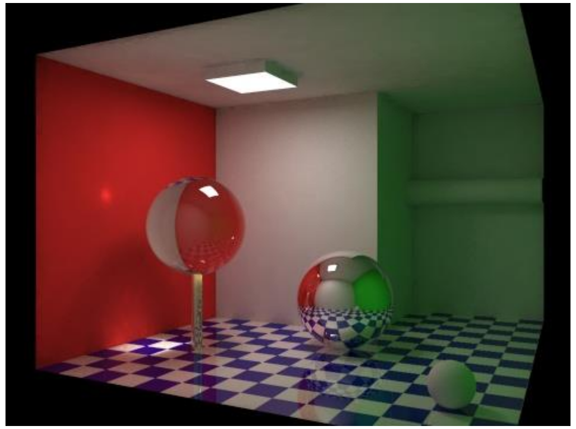
Introduction

Motivation:

- existing illumination methods in 3D visualization:
  - Phong shading
  - global illumination
    - ambient occlusion, approximation of global illumination



Phong shading



global illumination

ambient occlusion

Image source: Smith, Brad. (7 August 2006). Illustration of the Phong reflection model. Retrieved July 2, 2021 from https://commons.wikimedia.org/wiki/File:Phong_components_version_4.png, Barahag. (24 March 2020). 3 spheres in the small room. Retrieved July 2, 2021 from https://en.wikipedia.org/wiki/File:Global_illumination1.png, MITTRING, Martin, 2007. Finding next Gen: CryEngine 2. In: ACM SIGGRAPH 2007 Courses. San Diego, California: Association for Computing Machinery. SIGGRAPH 07. Available from doi: 10.1145/1281500.1281671.

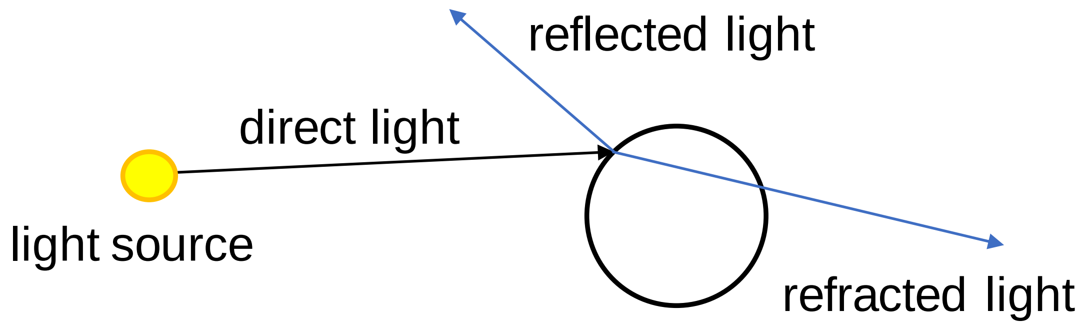
Introduction

Problem:

expand ambient occlusion method to 4D space

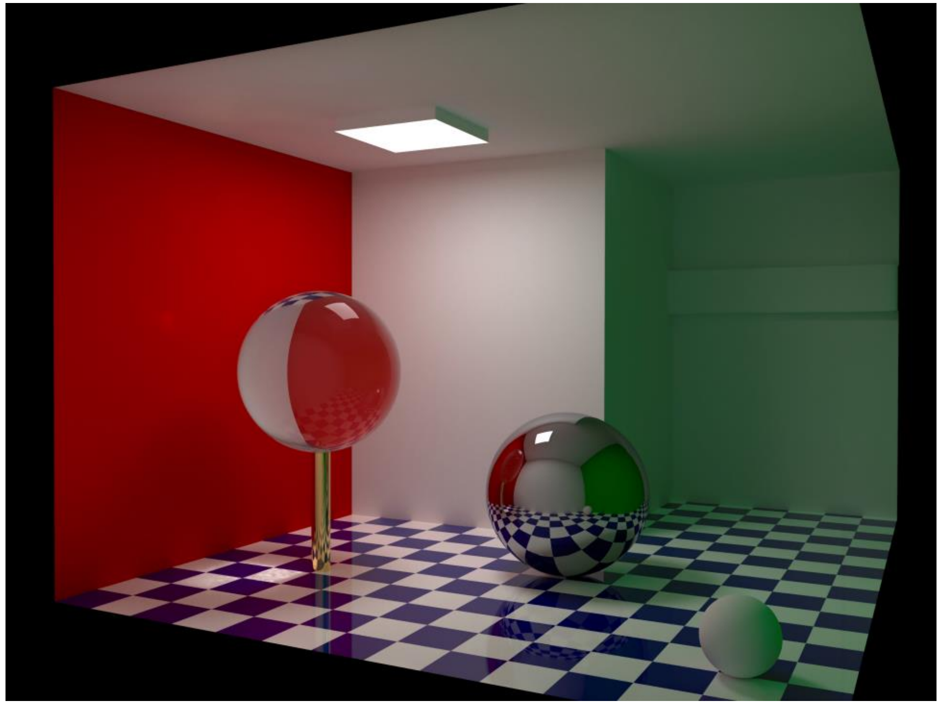
Global Illumination

- Shading technique, uses both direct and indirect light
- Photorealistic results
- Global illumination takes the entire scene into consideration
- Direct light: light coming directly from a light source
- Indirect light: reflected/refracted rays
- Calculated recursively, expensive to calculate

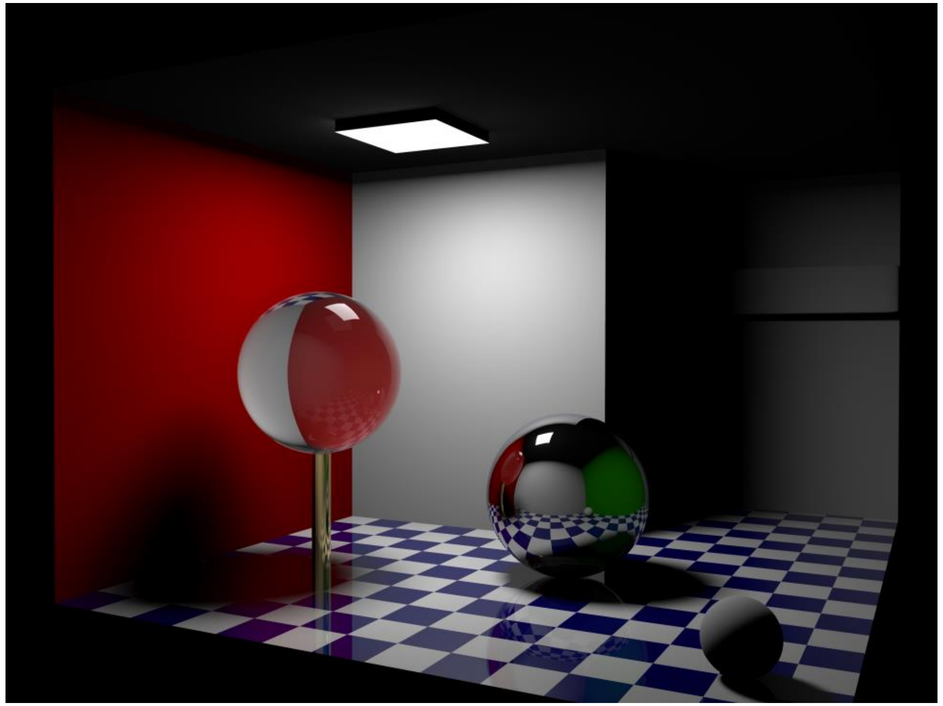


Global Illumination

Example of global illumination:



A scene with global illumination



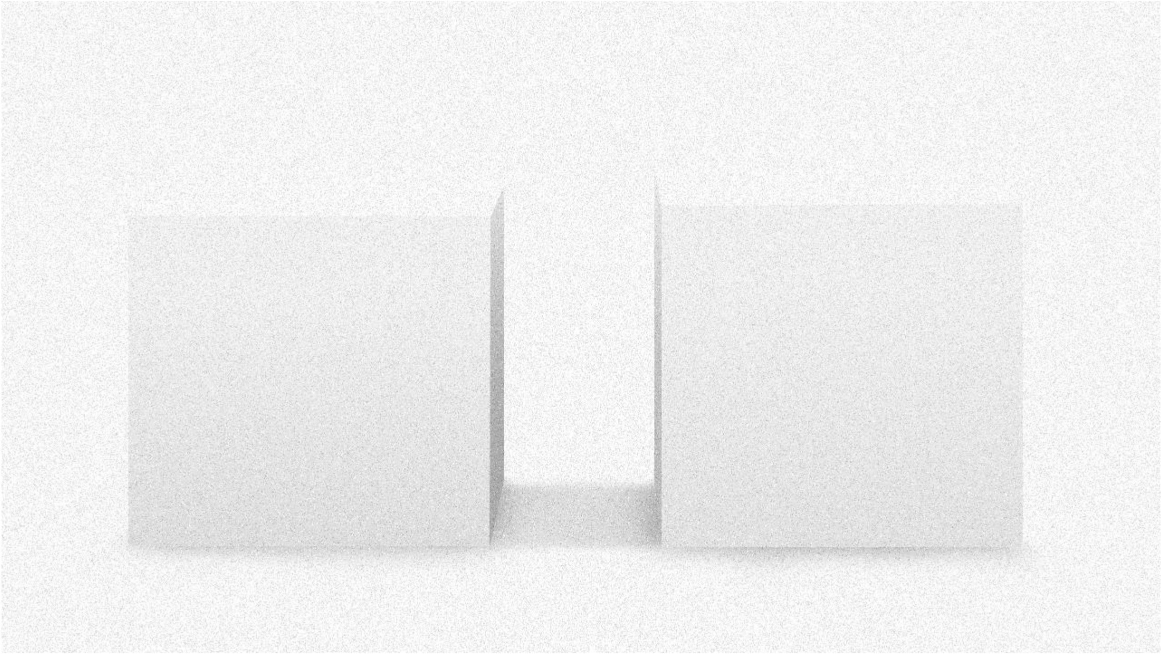
A scene without global illumination

Image source: Barahag. (24 March 2020). Three spheres lit only by direct lighting algorithms. Retrieved July 2, 2021 from https://en.wikipedia.org/wiki/File:Direct_lighting.png, Barahag. (24 March 2020). 3 spheres in the small room. Retrieved July 2, 2021 fromhttps://en.wikipedia.org/wiki/File:Global_illumination1.png

Ambient Occlusion

- Approximates global illumination, also takes the entire scene into consideration
- Shading on surfaces independent of light sources
- Examples of ambient occlusion maps:

Ambient occlusion of a scene from *Crysis* using *CryEngine 2*

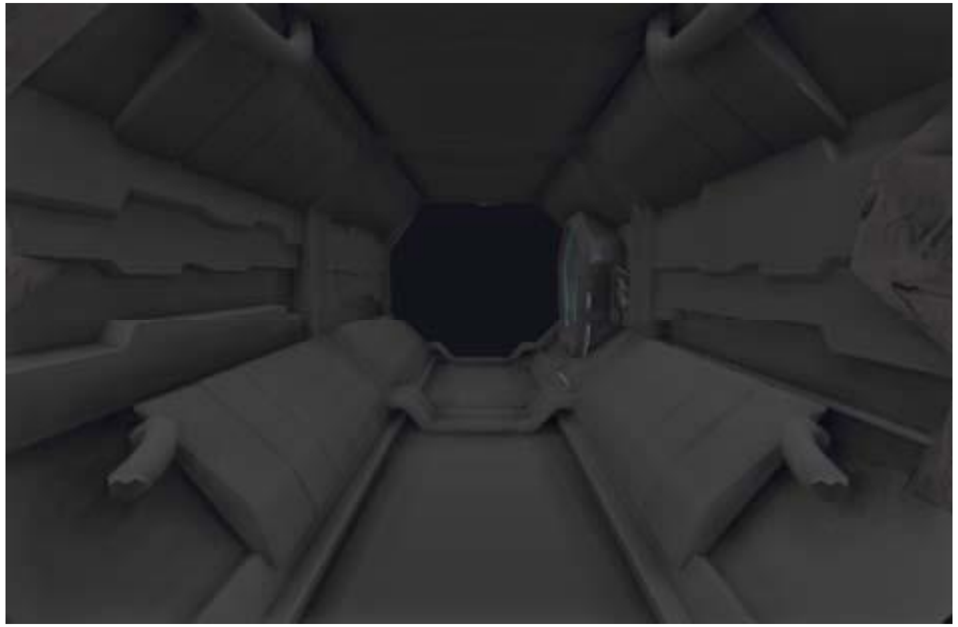


A simple scene of 2 cubes on a plane with ambient occlusion

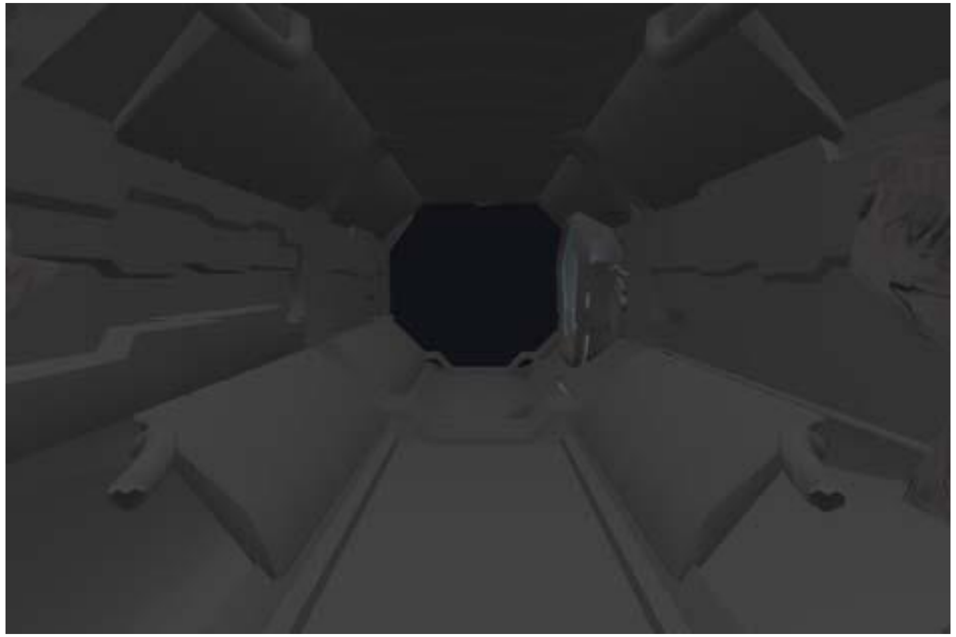
Image source: MITTRING, Martin, 2007. Finding next Gen: CryEngine 2. In: ACM SIGGRAPH 2007 Courses. San Diego, California: Association for Computing Machinery. SIGGRAPH 07. Available from doi: 10.1145/1281500.1281671.

Ambient Occlusion

- Ambient occlusion maps calculated separately from the scene
- Scene combined with ambient occlusion afterwards



A scene with ambient occlusion



A scene without ambient occlusion

Image source: MITTRING, Martin, 2007. Finding next Gen: CryEngine 2. In: ACM SIGGRAPH 2007 Courses. San Diego, California: Association for Computing Machinery. SIGGRAPH 07. Available from doi: 10.1145/1281500.1281671.

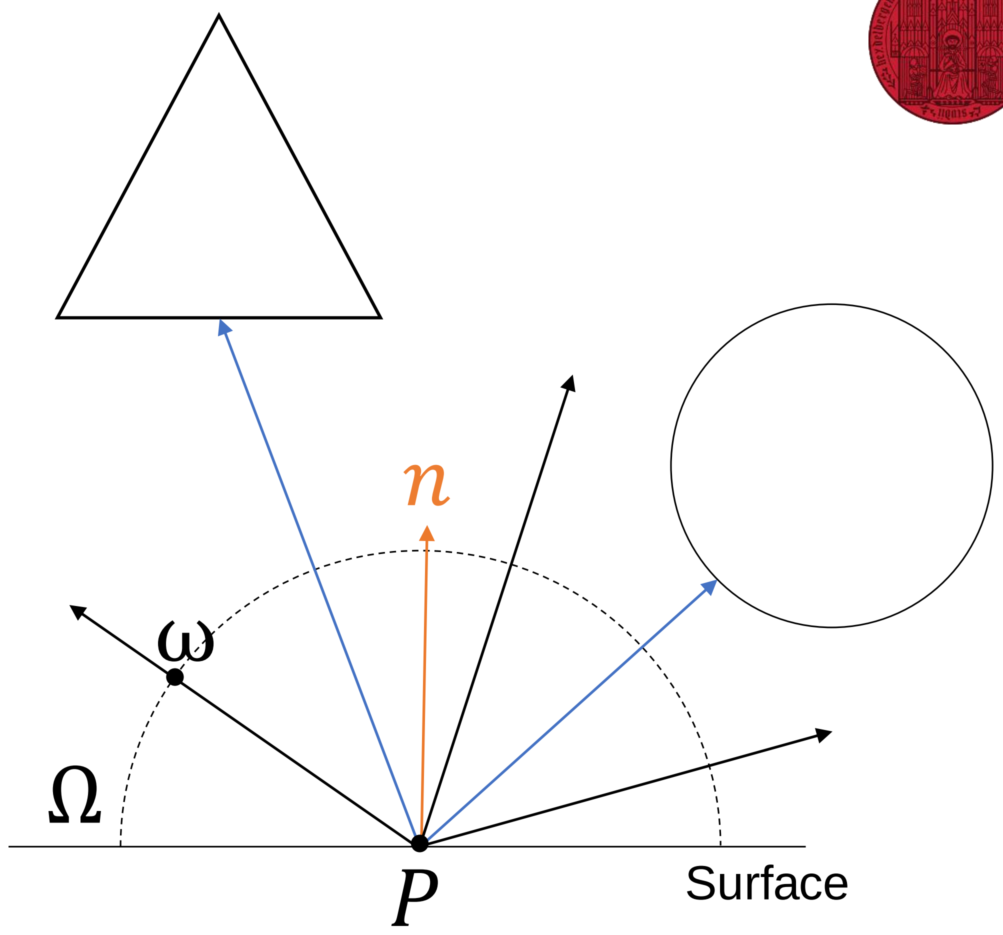
Variations of Ambient Occlusion

- Screen Space Ambient Occlusion (SSAO):
  uses only currently visible scene and depth buffer to calculate occlusion
- Ray-traced Ambient Occlusion:
  uses ray tracing to calculate occlusion

Ambient Occlusion (AO)

$$A(p, n) = \frac{1}{\pi} \int_{w \in \Omega} V(p, w) |w \cdot n| dw$$

p = *point in scene*
n = *surface normal*
Ω = hemisphere
$V(p, \omega)$ = visibility function, returns 1 if there's a ray-object intersection for ray from p to ω, otherwise 0



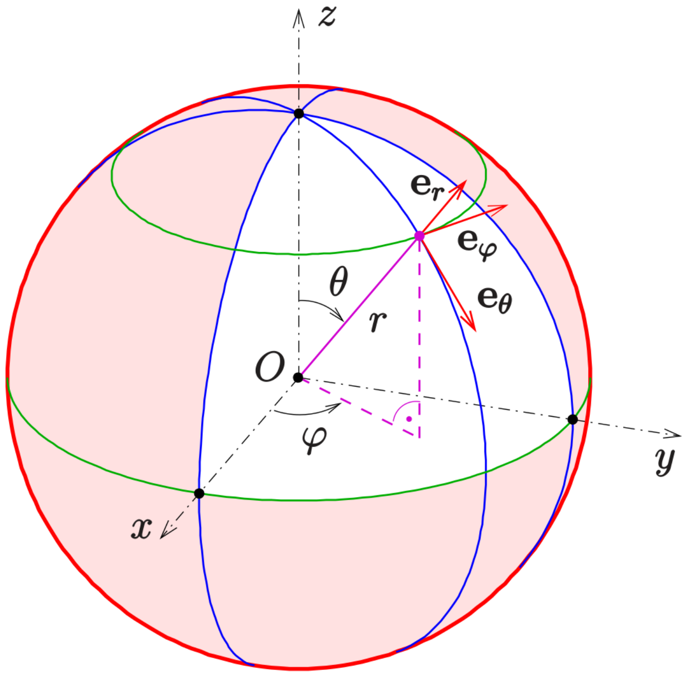
- Ambient occlusion calculates the average of the visibility function
- Practically, AO is calculated using hemispheric sampling

Hemispheric Sampling

- Create a hemisphere centered on a point p
- Shoot rays to random points on the hemisphere
- Points on the hemisphere calculated using spherical coordinates

Point $P = (x, y, z)$ on a sphere with center point c and radius r can be defined as

$$x = c + r\cos\varphi\sin\theta$$
$$y = c + r\sin\varphi\sin\theta$$
$$z = c + r\cos\theta$$

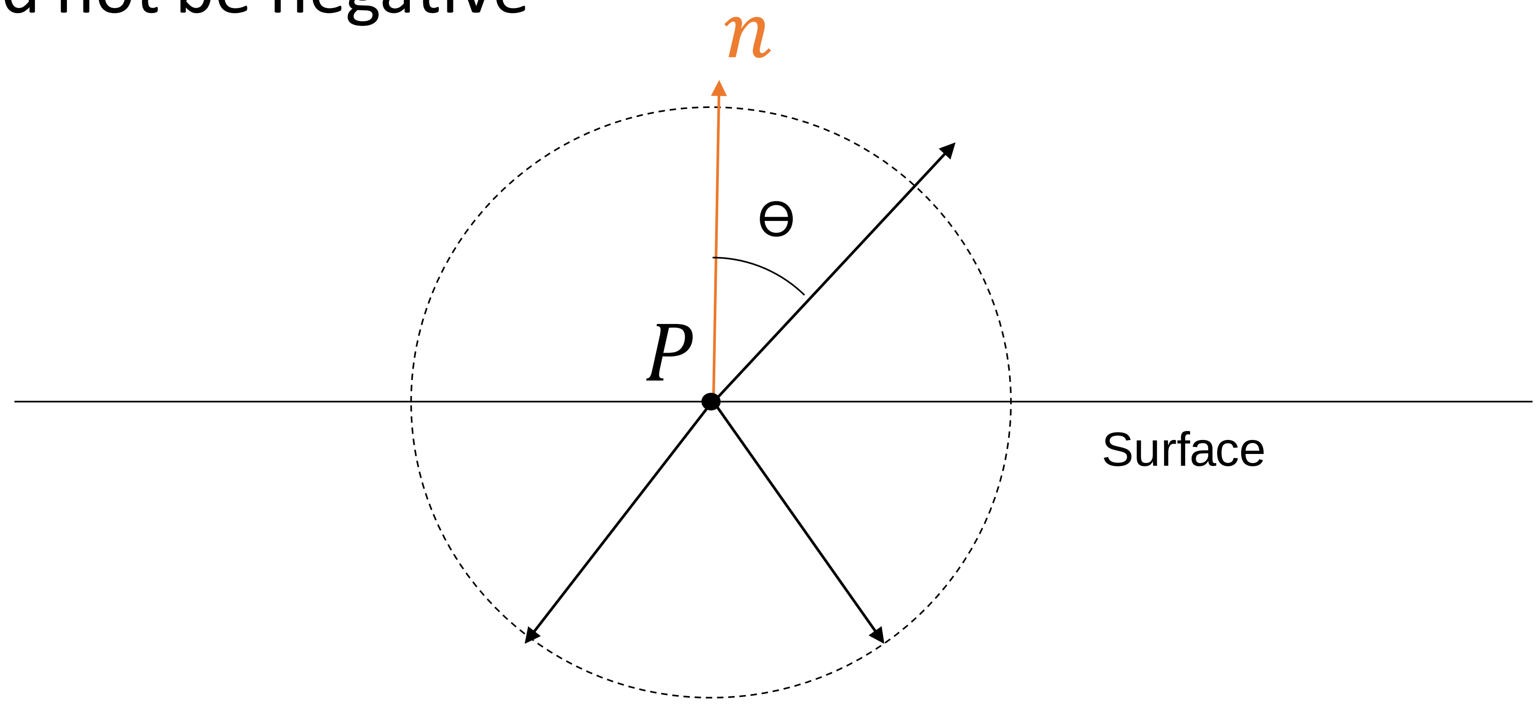


Spherical coordinates in 3D

Image source: Ag2gaeh. (3 October 2015). Kugelkoordinaten: lokale Basis. Retrieved July 2, 2021 from https://commons.wikimedia.org/wiki/File:Kugelkoord-lokale-Basis-s.svg

Hemispheric Sampling

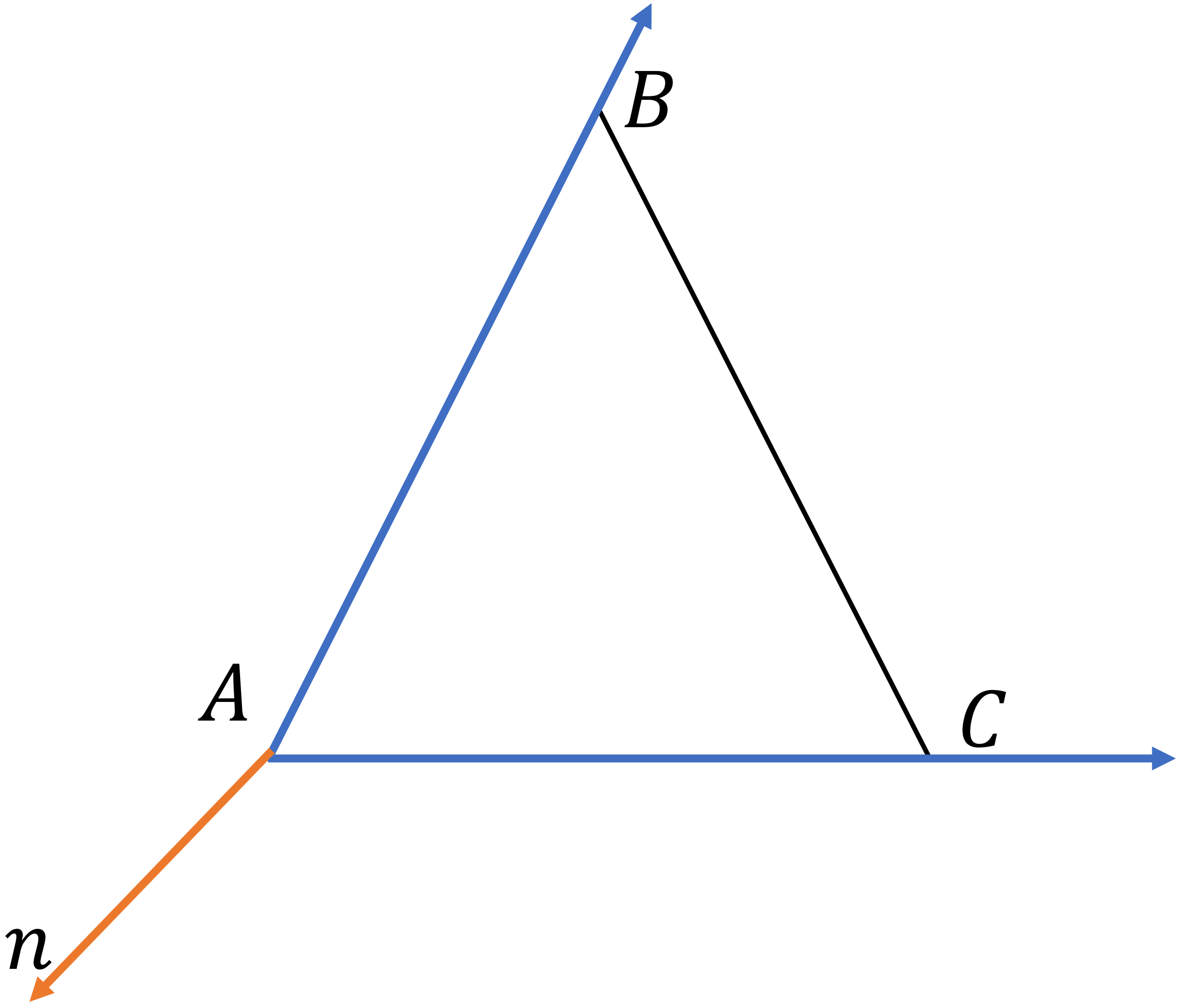
- Problem: Using spherical coordinates we get points on a sphere, not hemisphere. Sampled rays may point under the surface
- Solution: Ray must point to the same direction as surface normal, angle between ray and surface normal cannot be more than 90⁰ → cross product of ray and surface normal should not be negative



Finding Surface Normal

In 3D: find surface normal using cross product of two sides of a triangle

$$n = \frac{(B-A)\times(C-A)}{[(B-A)\times(C-A)]}$$



Ray-Object Intersection in 3D

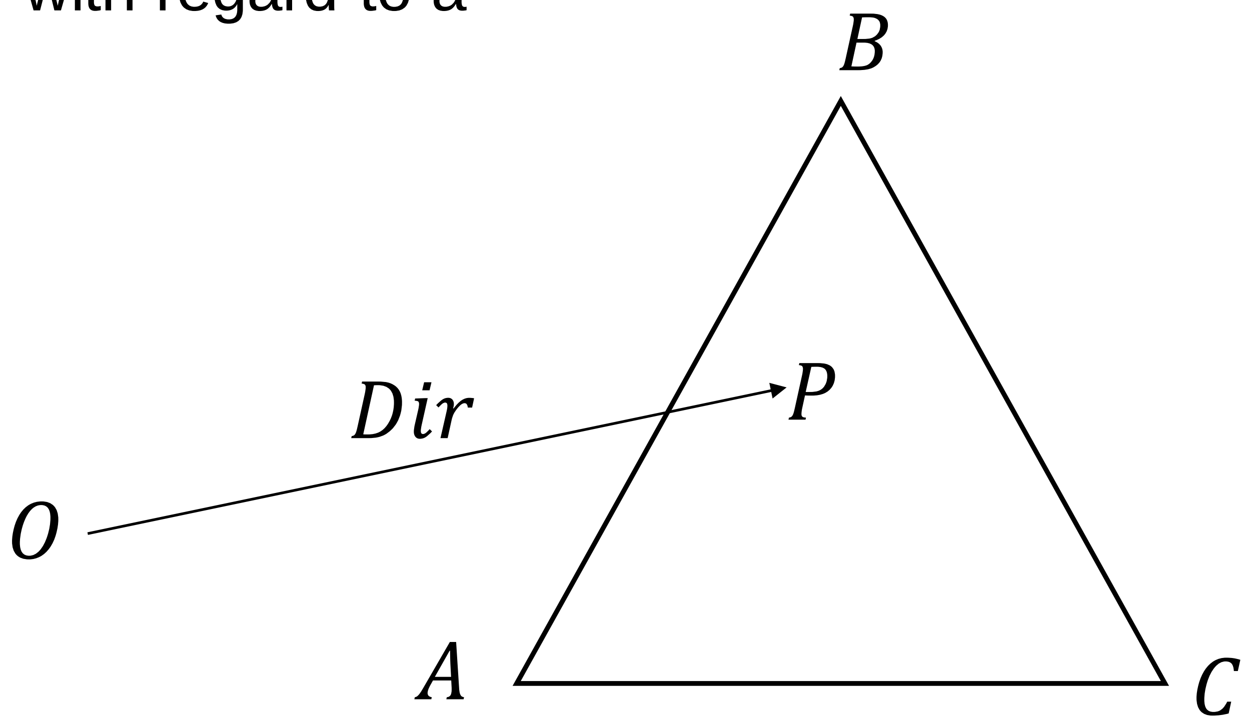
- Ray-object intersection is solved using ray-triangle intersection
- Ray-triangle intersection calculated using barycentric coordinates

$\lambda_{1,}\, \lambda_2,\, \lambda_3$ barycentric coordinates for a point P with regard to a triangle ABC

$$P = \lambda_1 A + \lambda_2 B + \lambda_3 C$$
$$\text{with } \lambda_1 + \lambda_2 + \lambda_3 = 1$$

Barycentric coordinates is calculated using definition of a point on a ray with origin O and direction Dir:

$$P = O + tDir$$



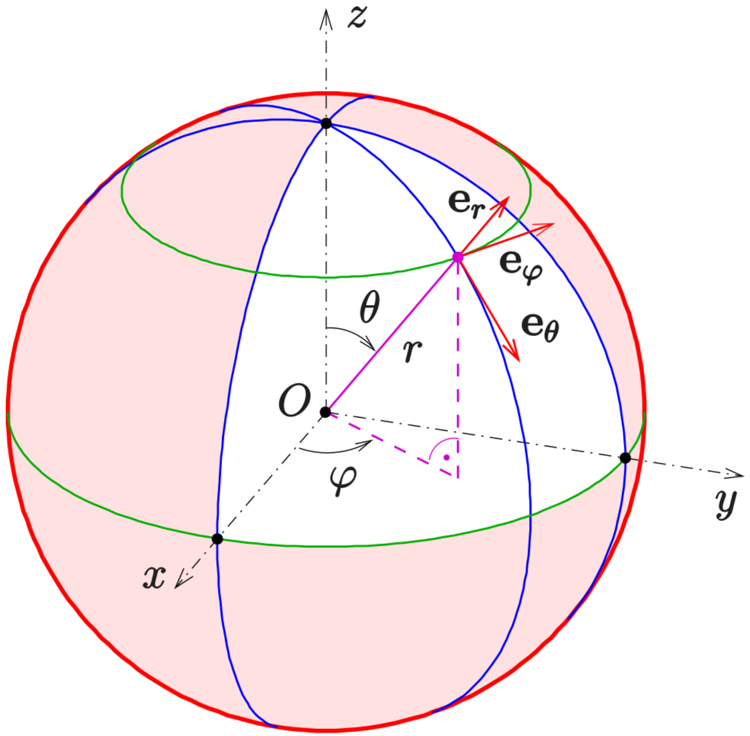
Ambient Occlusion in 4D

Subproblems:

- Hemispheric Sampling using hypersphere
- Ray-Object Intersection in 4D

Hemispheric Sampling in 4D

Generalized spherical coordinates



Spherical coordinates in 3D

$$p = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} r\cos(\phi_1) \\ r\sin(\phi_1)\cos(\phi_2) \\ r\sin(\phi_1)\sin(\phi_2)\cos(\phi_3) \\ \vdots \\ r\sin(\phi_1)\ldots\sin(\phi_{n-2})\cos(\phi_{n-1}) \\ r\sin(\phi_1)\ldots\sin(\phi_{n-2})\sin(\phi_{n-1}) \end{bmatrix}$$

Image source: Ag2gaeh. (3 October 2015). Kugelkoordinaten: lokale Basis. Retrieved July 2, 2021 from https://commons.wikimedia.org/wiki/File:Kugelkoord-lokale-Basis-s.svg

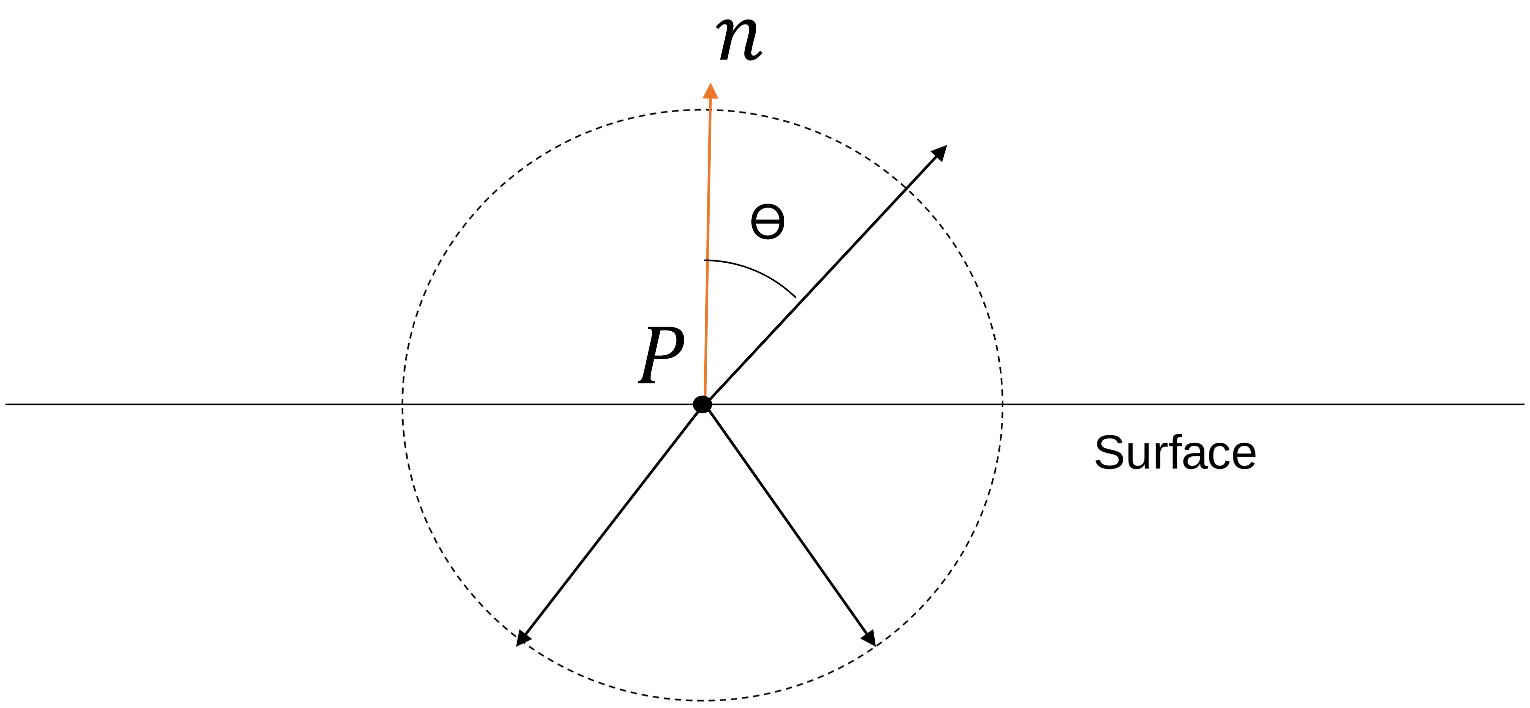
Hemispheric Sampling in 4D

Spherical coordinates for $n = 4$, center point c

$$p = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} c + r\cos(\phi_1) \\ c + r\sin(\phi_1)\cos(\phi_2) \\ c + r\sin(\phi_1)\sin(\phi_2)\cos(\phi_3) \\ c + r\sin(\phi_1)\sin(\phi_2)\sin(\phi_3) \end{bmatrix}$$

Hemispheric Sampling in 4D

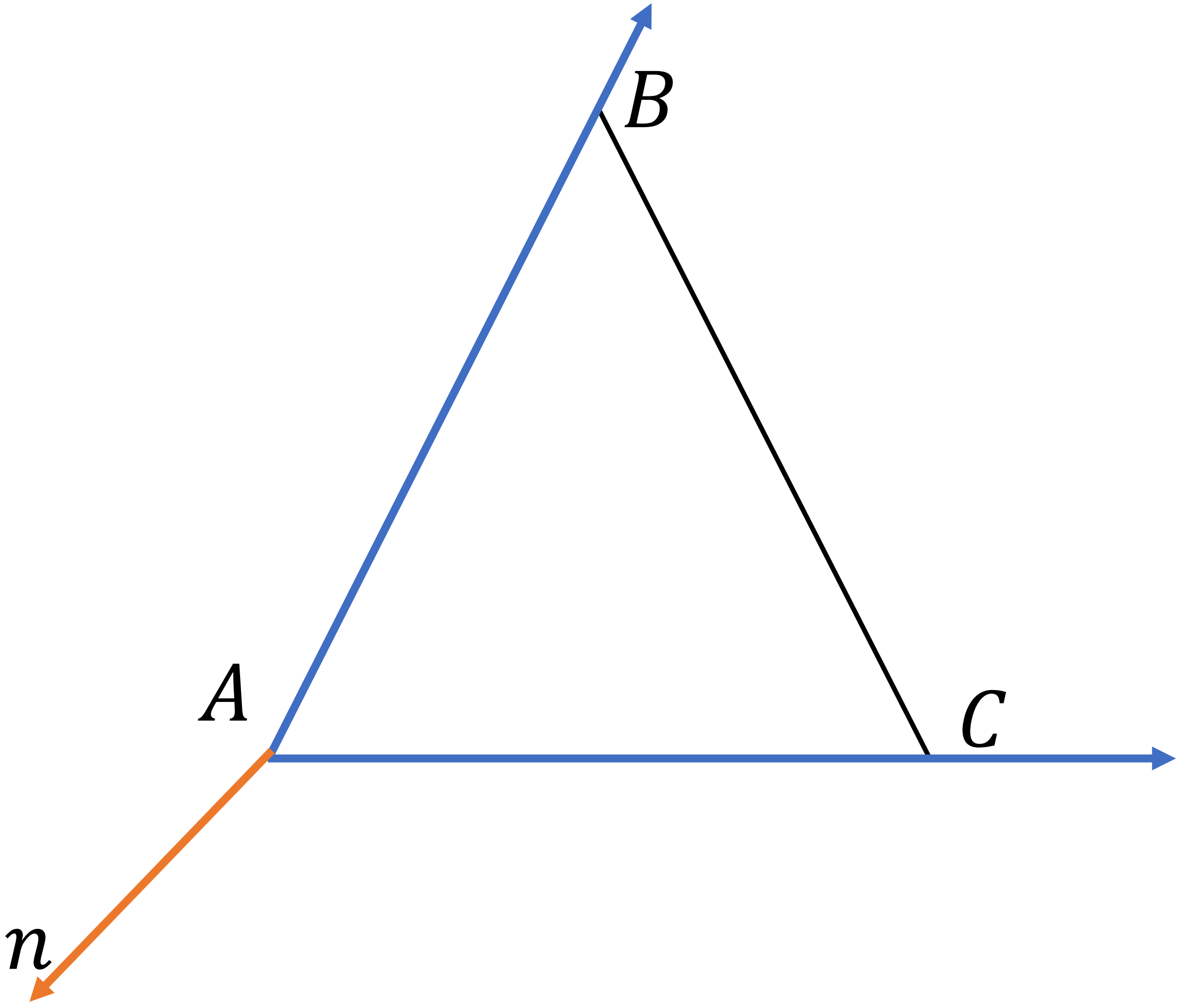
- The same problem as in 3D, ray must point to the same direction as surface normal, angle between ray and surface normal cannot be more than 90^o → cross product of ray and surface normal should not be negative



Finding Surface Normal

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

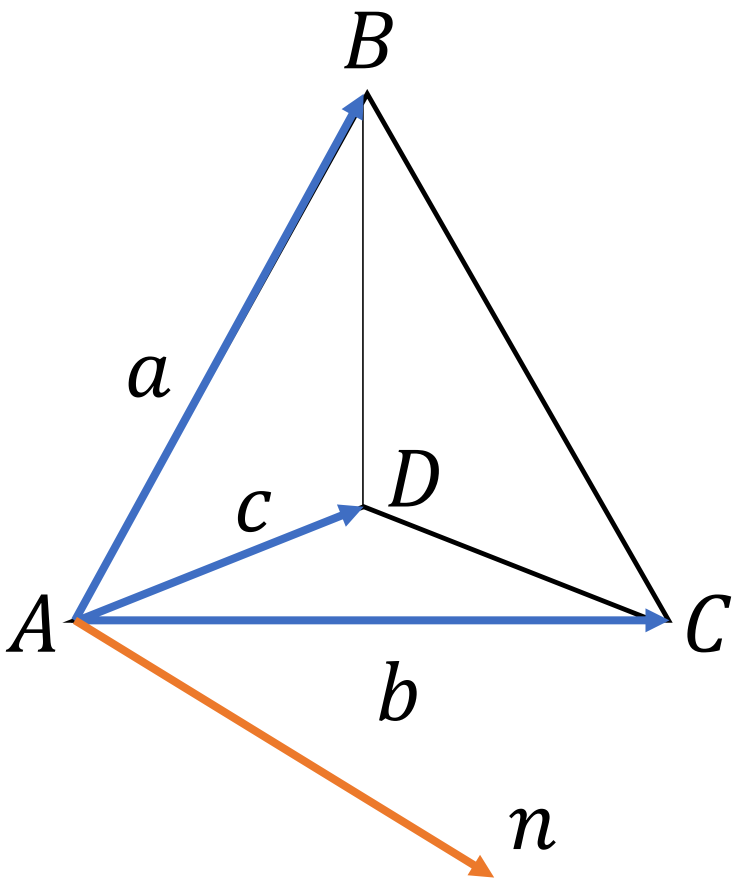
- In 3D: find surface normal using cross product of two sides of a triangle
- In 4D: cross product not defined



VISUAL COMPUTING GROUP HEIDELBERG UNIVERSITY

Finding Surface Normal

- Finding surface normal: Finding an orthogonal vector
- In 4D possible to find a vector that is orthogonal to 3 other vectors
- 3 vectors selected from 3 sides of the tetrahedron



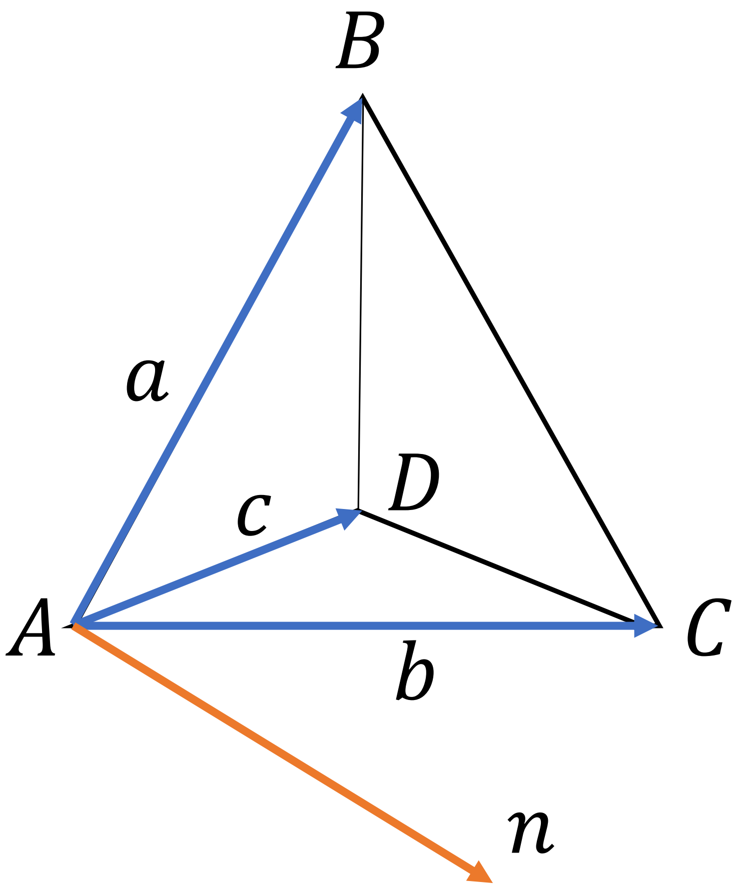
Finding Surface Normal

Use formal determinant to find orthogonal vector

$$a \times b \times c = \begin{vmatrix} x & y & z & w \\ a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \end{vmatrix}, n = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

$$x = \det \begin{bmatrix} \boldsymbol{a}_2 & \boldsymbol{a}_3 & \boldsymbol{a}_4 \\ \boldsymbol{b}_2 & \boldsymbol{b}_3 & \boldsymbol{b}_4 \\ \boldsymbol{c}_2 & \boldsymbol{c}_3 & \boldsymbol{c}_4 \end{bmatrix} \quad y = \det \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_3 & \boldsymbol{a}_4 \\ \boldsymbol{b}_1 & \boldsymbol{b}_3 & \boldsymbol{b}_4 \\ \boldsymbol{c}_1 & \boldsymbol{c}_3 & \boldsymbol{c}_4 \end{bmatrix}$$

$$z = \det \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \boldsymbol{a}_4 \\ \boldsymbol{b}_1 & \boldsymbol{b}_2 & \boldsymbol{b}_4 \\ \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{c}_4 \end{bmatrix} \quad w = \det \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \boldsymbol{a}_3 \\ \boldsymbol{b}_1 & \boldsymbol{b}_2 & \boldsymbol{b}_3 \\ \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{c}_3 \end{bmatrix}$$

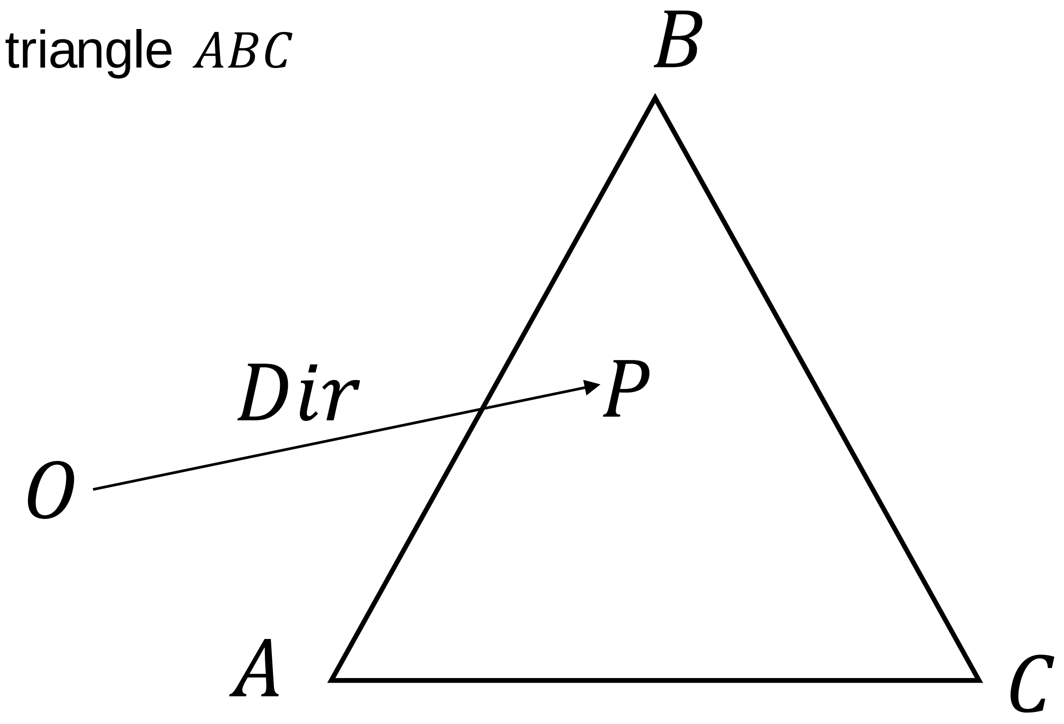


Review: Ray-Object Intersection in 3D

- Ray-object intersection is solved using ray-triangle intersection
- Ray-triangle intersection calculated using barycentric coordinates

$\lambda_1, \lambda_2, \lambda_3$ barycentric coordinates for a point P with regard to a triangle ABC

$$P = \lambda_1 A + \lambda_2 B + \lambda_3 C$$
$$\text{with } \lambda_1 + \lambda_2 + \lambda_3 = 1$$



Barycentric coordinates are solved using definition of a point on a ray with origin O and direction Dir:

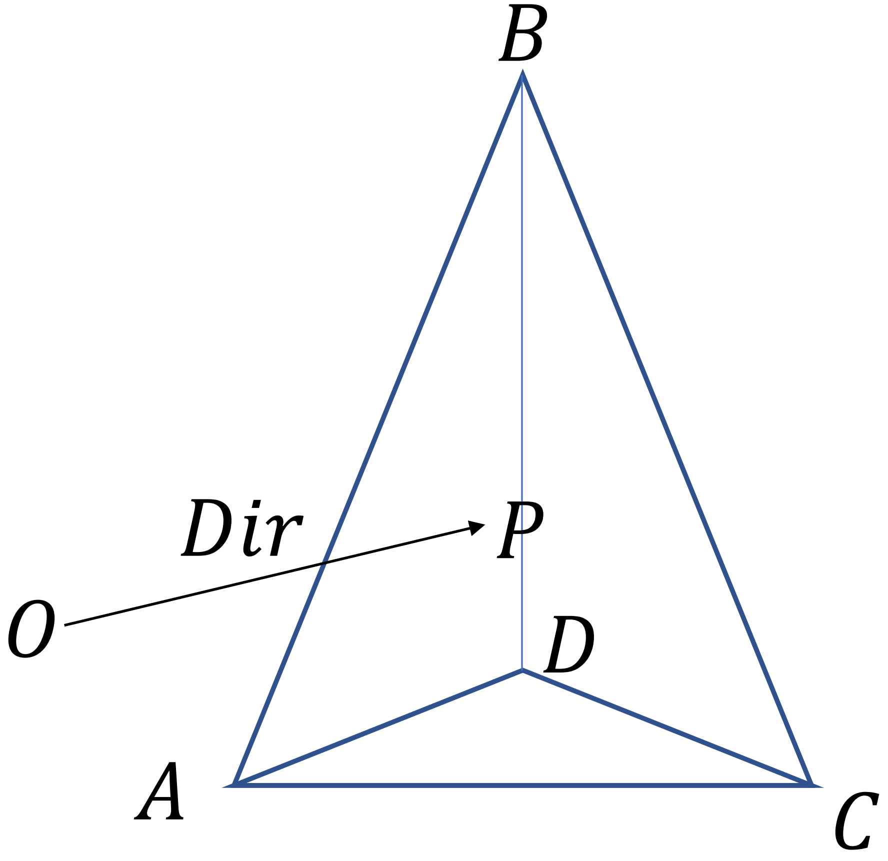
$$P = O + tDir$$

Ray-Object Intersection in 4D

- Same principles like in 3D but with a 3-simplex (tetrahedron)
- Use barycentric coordinate with tetrahedron to find ray intersection point

Barycentric coordinate with 3-simplex (tetrahedron)

$$P = \lambda_1 A + \lambda_2 B + \lambda_3 C + \lambda_4 D$$
$$\text{with } \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$$



Barycentric coordinates in 4D can also be solved using definition of a point on a ray with origin O and direction Dir

$$P = O + tDir$$

Ray-Object Intersection in 4D

Using the barycentric coordinate with 3-simplex (tetrahedron) and definition of point on a ray, we get the following linear system

$$
\begin{aligned}
O + tDir &= \lambda_1 A + \lambda_2 B + \lambda_3 C + \lambda_4 D \\
&= A(1 - \lambda_2 - \lambda_3 - \lambda_4) + \lambda_2 B + \lambda_3 C + \lambda_4 D \\
&= A + \lambda_2 (B - A) + \lambda_3 (C - A) + \lambda_4 (D - A)
\end{aligned}
$$

$$O - A = tDir + \lambda_2 (B - A) + \lambda_3 (C - A) + \lambda_4 (D - A)$$

Ray-Object Intersection in 4D

Linear system solved using Cramer's rule (using determinants)

Cramer's Rule

for $a = (a1, a2)$, $b = (b1, b2)$,
$c = (c1, c2)$ x and y can be calculated as follows:

$$ax + by = c$$

$$x = \det(c\ b)\ /\det(a\ b)$$
$$y = \det(a\ c)\ /\det(a\ b)$$

Ray-Object Intersection in 4D

Solution of the linear system:

$$t = \frac{\det((O-A) \quad (B-A) \quad (C-A))}{\det(Dir \quad (B-A) \quad (C-A) \quad (D-A))}$$

$$\lambda_2 = \frac{\det(Dir \quad (O-A) \quad (C-A))}{\det(Dir \quad (B-A) \quad (C-A) \quad (D-A))}$$

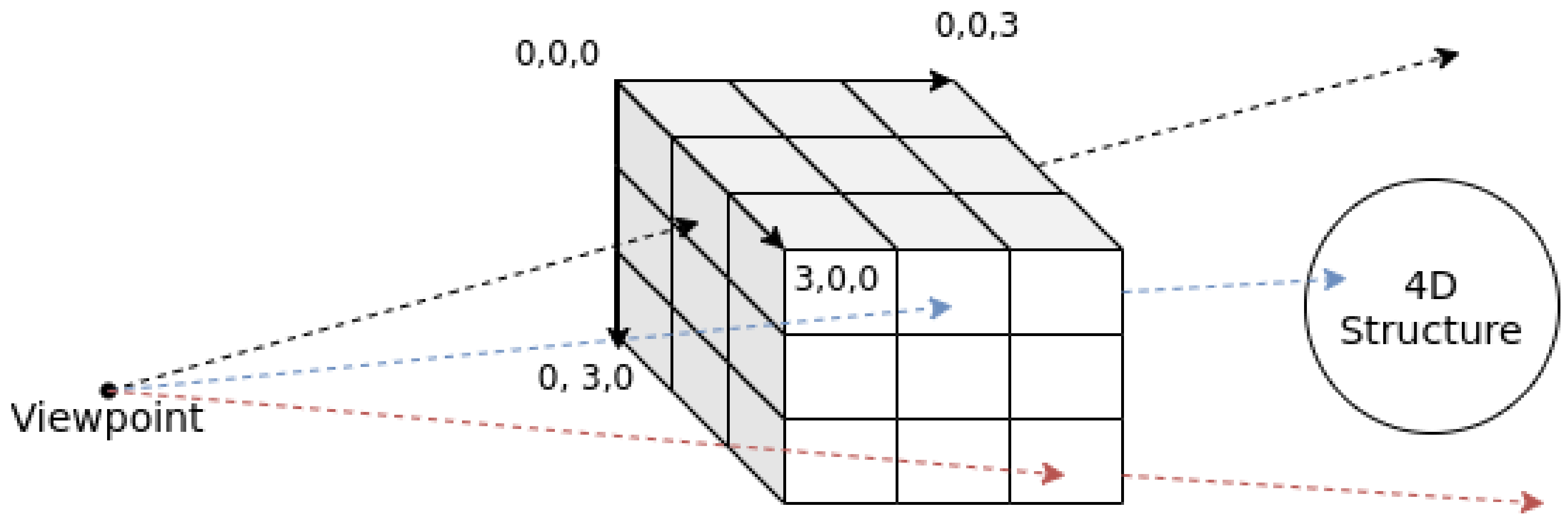
$$\lambda_3 = \frac{\det(Dir \quad (B-A) \quad (O-A))}{\det(Dir \quad (B-A) \quad (C-A) \quad (D-A))}$$

$$\lambda_4 = \frac{\det(Dir \quad (B-A) \quad (C-A) \quad (O-A))}{\det(Dir \quad (B-A) \quad (C-A) \quad (D-A))}$$

Implementation

C++, OpenCL for parallel computing

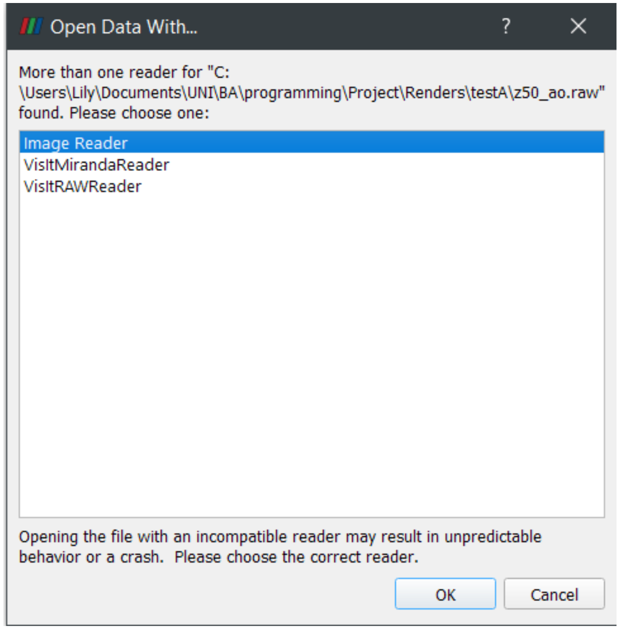
Experimental Result

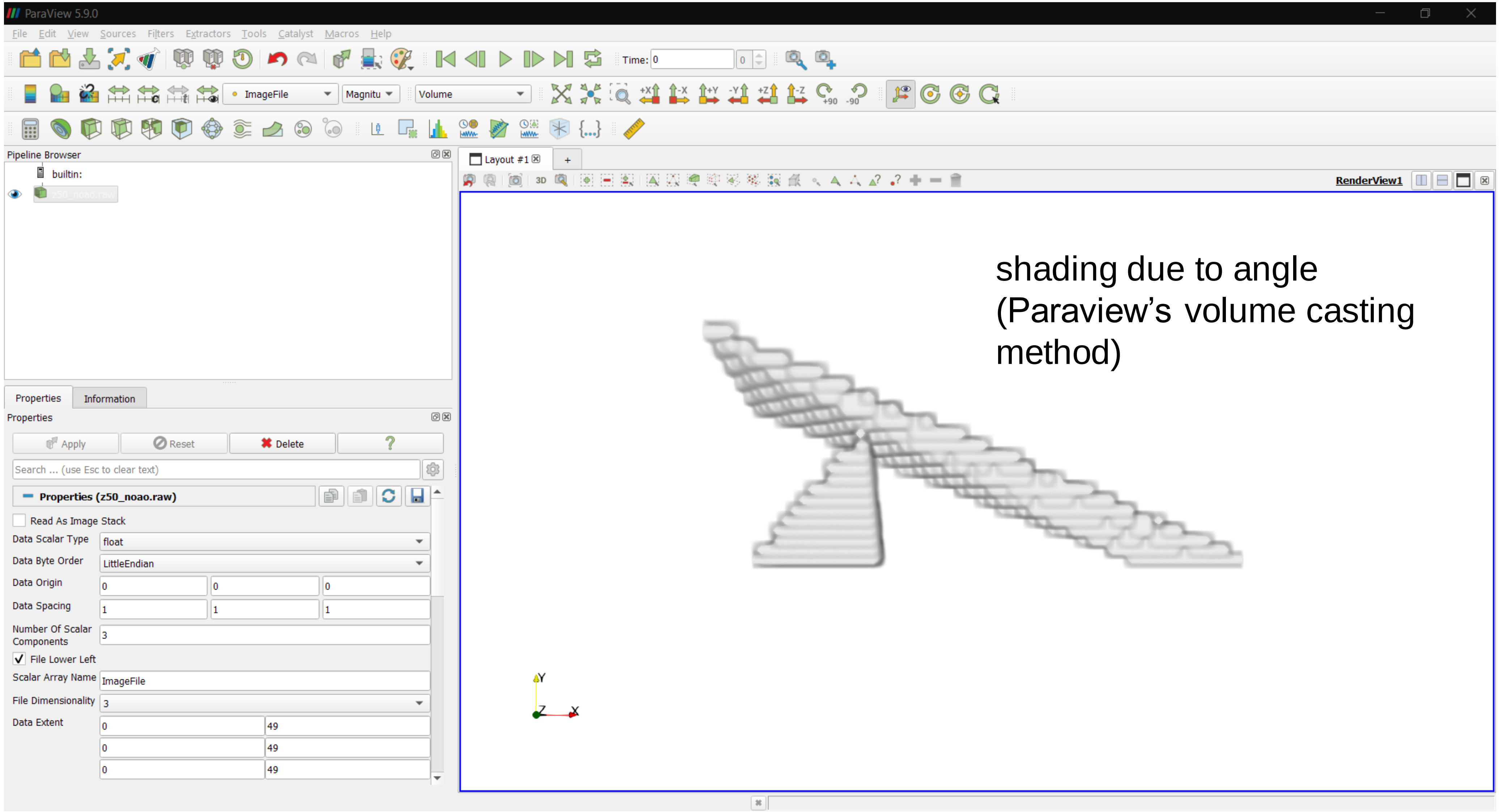


4D scenes projected to 3D image space (50 x 50 x 50)

Experimental Result

Results rendered in ParaView using RAW Image Reader





VISUAL COMPUTING GROUP HEIDELBERG UNIVERSITY

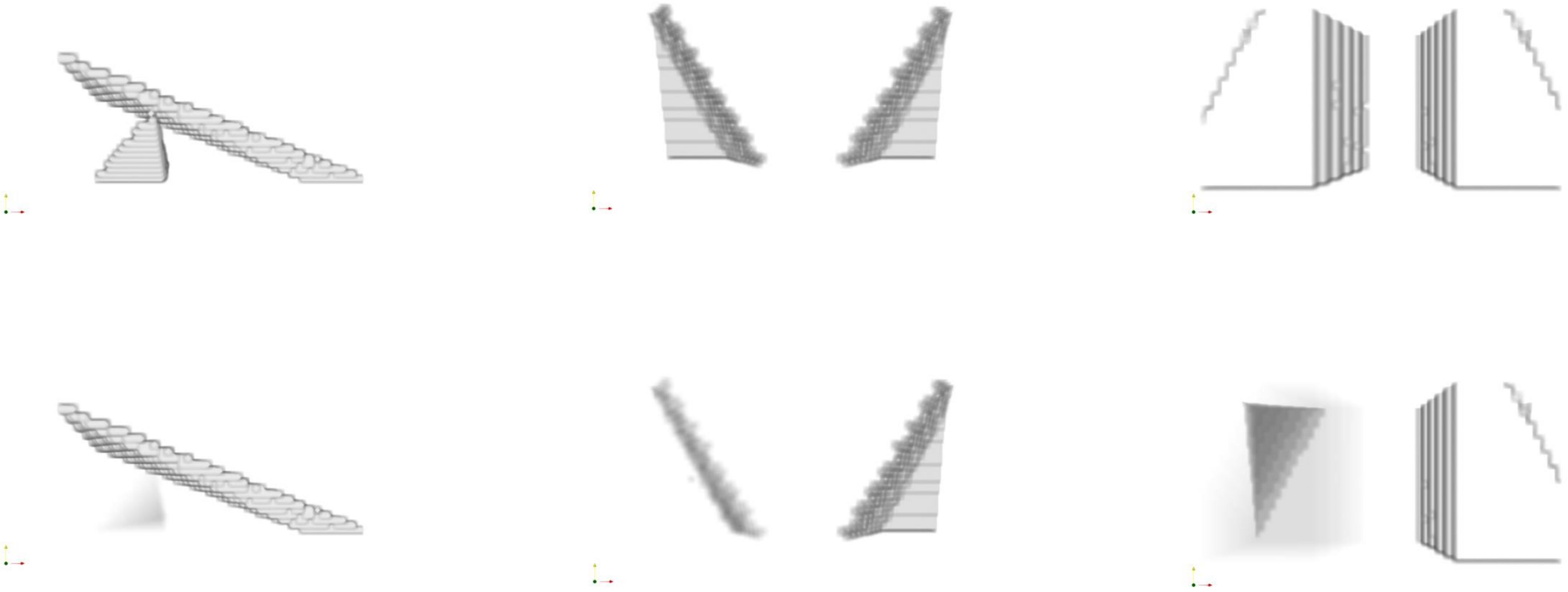
UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Experimental Result

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

without ambient occlusion

with ambient occlusion



2 tetrahedra

2 structure out of 2 tetrahedra

Cube-like structures with an indent

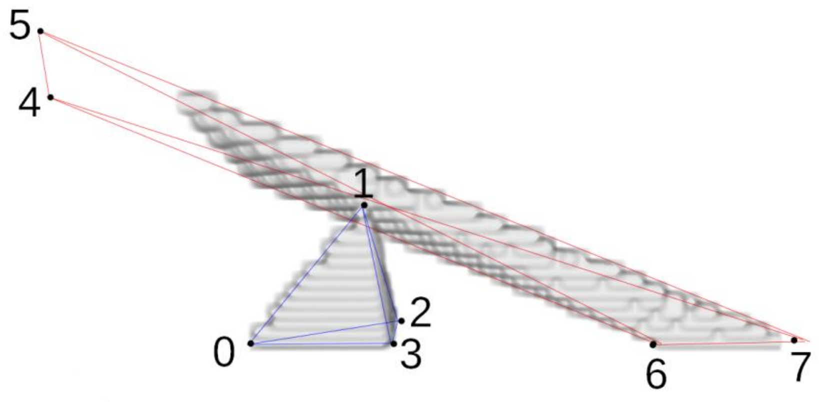
VISUAL COMPUTING GROUP HEIDELBERG UNIVERSITY

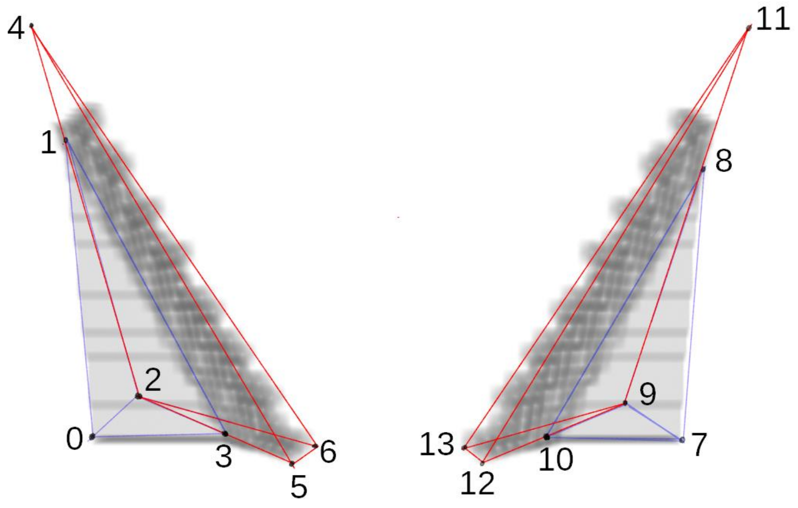
Unexpected Result in Our Implementation

- Entirely occluded tetrahedra, values either 0 or 1.
- Left object always affected differently than the right

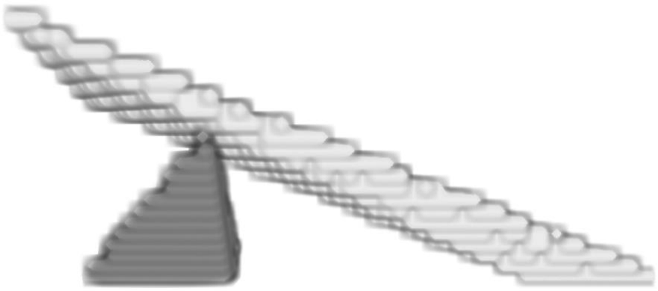
Theoretical Result

without ambient occlusion, marked vertices





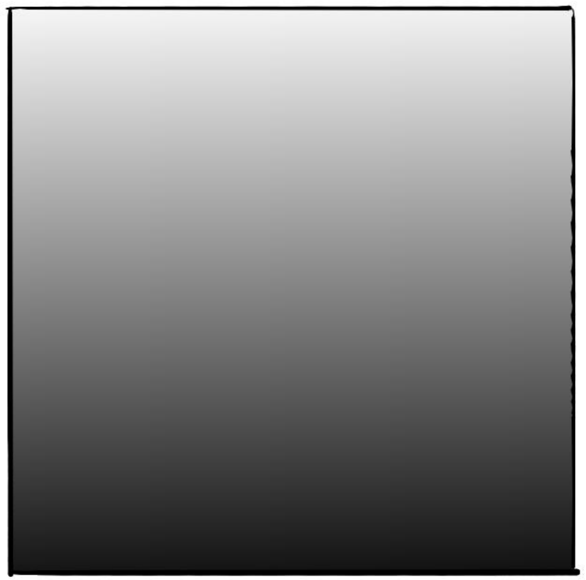
with theoretical ambient occlusion



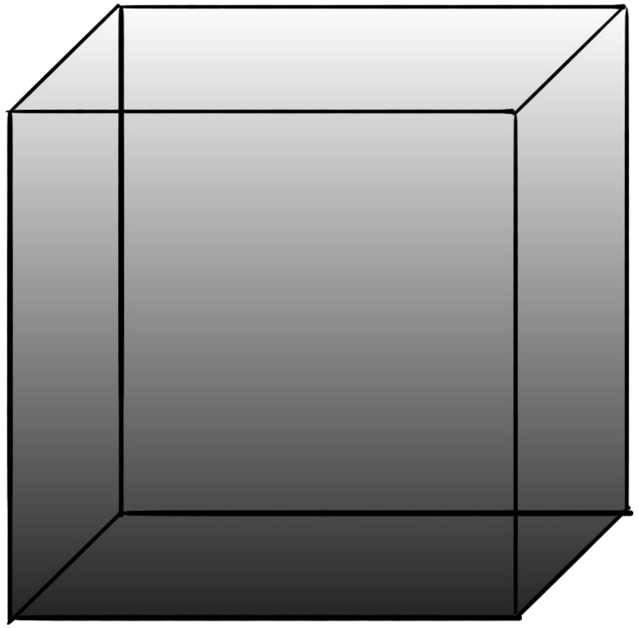
2 tetrahedra

2 structure out of 2 tetrahedra

Theoretical Result



In 3D: surface of an object is a plane,
occlusion is calculated plane-wise



In 4D: surface of an object is a volume,
occlusion is calculated volume-wise

Insufficient Test Data

- Test data uses structures build out from tetrahedra
- Test data does not represent surfaces of 4D structures
- Cannot observe true characteristics of ambient occlusion in 4D

Finding Better Test Data

- Test data should represent surfaces of 4D structures
- Surface of a 4D structure is a volume
- Volume represented as tetrahedral mesh
- Finding surface of a 4D structure is difficult

Finding Surfaces of 4D Structures

- Use isosurface to find surface of a 4D structure
- Isosurface represents points of constant values in a volume
- Possible solution: marching cubes in 4D

Marching Cubes

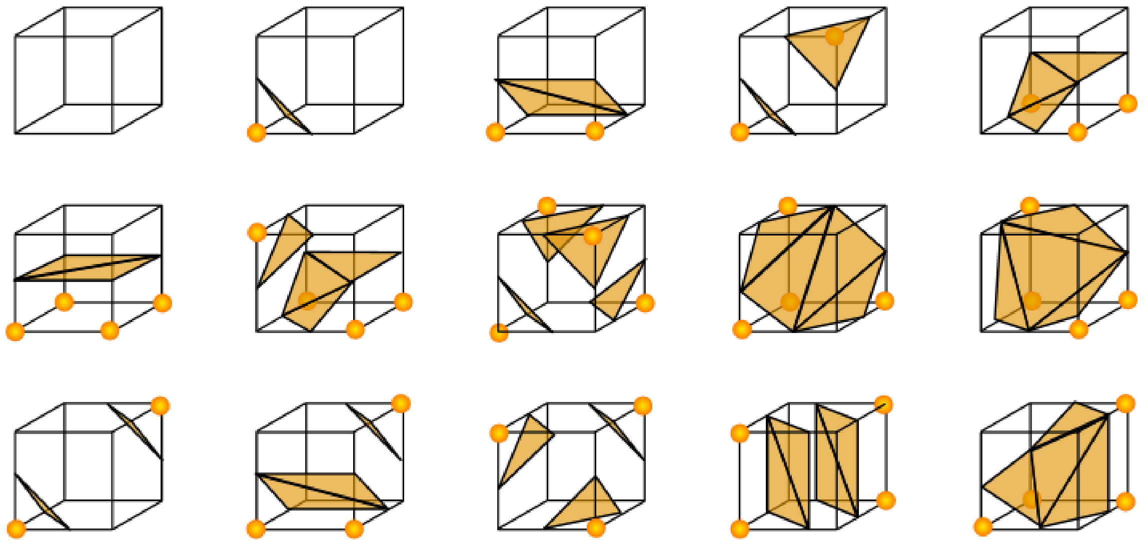
UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

In 3D:

- Extract polygonal mesh of an isosurface of a discrete 3D structure
- Uses pre-defined polygonal configurations in a cube saved as a look-up table

In higher dimensions:

- Introduced by Bhaniramka et. al., 2000
- Generates d-simplex mesh for dimensions $\mathrm{d} \geq 3$
- Look-up table generation, size of look-up table 2^{2^d}



15 configurations from the original marching cubes algorithm in 3D

VISUAL COMPUTING GROUP
HEIDELBERG UNIVERSITY

Image source: Jmtrivial. (16 October 2006). Marching cubes cases. Retrieved July 1, 2021 from https://commons.wikimedia.org/wiki/File:MarchingCubes.svg

Conclusion

- It is possible to extend ambient occlusion to 4D directly
- Our implementation has unexpected results
- Tetrahedral mesh of surfaces of 4D structures required to fully observe characteristics of ambient occlusion
- Tetrahedral mesh can be generated using marching cubes

- Future work:
  - Illumination in 4D including specular value

Sources

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

- MITTRING, Martin, 2007. Finding next Gen: CryEngine 2. In: ACM SIGGRAPH 2007 Courses. San Diego, California: Association for Computing Machinery. SIGGRAPH 07. Available from doi: 10.1145/1281500.1281671.
- GANOVELLI, Fabio; CORSINI, Massimiliano; PATTANAIK, Sumanta; DI BENEDETTO, Marco, 2014. Introduction to Computer Graphics. Chapman and Hall/CRC.
- KHRONOS GROUP, 2020. OpenCL. Version 3.0. Available also from: https://www.khronos.org/opencl/.
- Wikipedia contributors. Global Illumination. Wikipedia, The Free Encyclopedia. Available at: https://en.wikipedia.org/w/index.php?title=Plagiarism&oldid=5139350. Accessed July 6, 2021.
- HOLLASCH, Steven R., 1991. Four-Space Visualization of 4D Objects. MA thesis. Arizona State University.
- CRAMER, Gabriel, 1750. Introduction à l'Analyse des lignes courbes algébriques.
- BHANIRAMKA, Praveen; WENGER, Rephael; CRAWFIS, Roger, 2000. Isosurfacing in higher dimensions. In: Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145), pp. 267–273.
- Lorensen, W. E., & Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *ACM siggraph computer graphics*, *21*(4), 163-169.

VISUAL COMPUTING GROUP
HEIDELBERG UNIVERSITY

Thank you for your attention!