Playing with Real-Time Shadows

Nikolas Kasyan
Crytek

SIGGRAPH2013

Shadows in Games: Crysis 1

Shadows in Games: Crysis 2

Shadows in Games: Crysis 3

Shadows in Games: Ryse

Shadow Methods & Techniques

- Deferred Shadows
- Cascaded shadow maps
- Soft Shadows Approximation
- Shadows & Transparency
- Contact Shadows/SSDO
- Screen Space Self-Shadowing
- Volumetric shadows
- Area Light Shadows

Typical Shadows Frame Budgets

- 33 ms typical videogame frame budget (30 FPS)
- 4000 Draw calls (average PC)
- ~5-7ms shadows frame budget
- 10Mb shadowmap texture pools (x360, ps3)
  - PC can go much higher
- 10+ shadow casting lights

Deferred Shadows

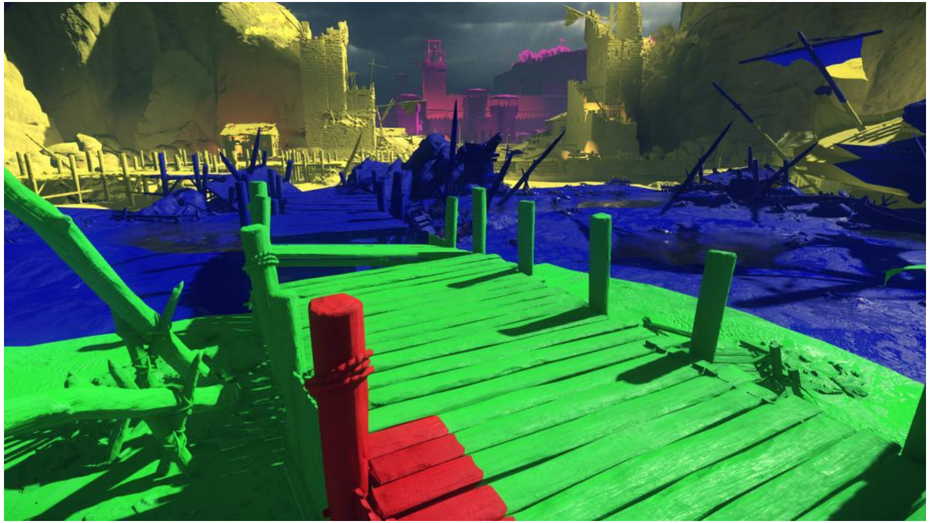
- Shadow mask for the sun
  - Special render target to accumulate shadow occlusion
  - Shadow mask combines multiple shadowing technique on top of each other before using in actual shading
    - VSM, per-object shadows, clouds shadows
- Point light shadows rendered directly to the light buffer


Sun shadow mask


10+ shadow casting lights

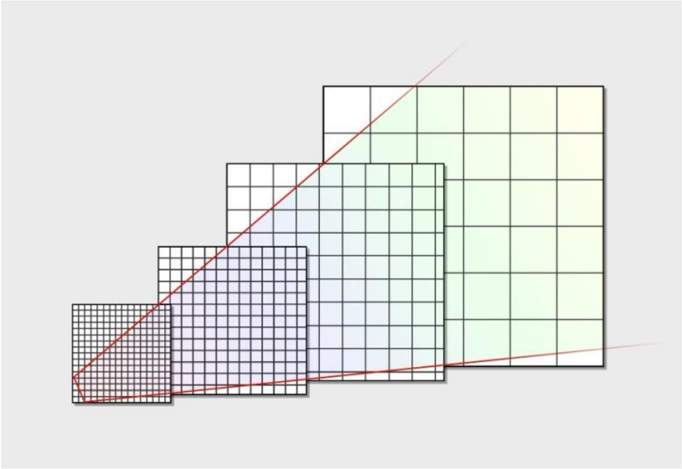
Cascaded Shadows Maps

- View Frustum is covered with multiple shadow frustums
  - Usually distance-based splitting



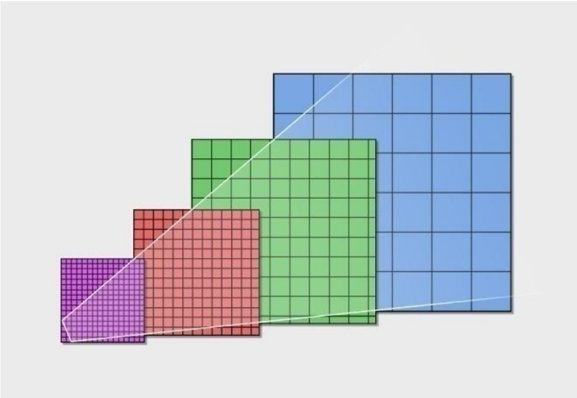
Cascaded Shadows Maps

- Cascades Splitting Scheme
  - Approximate Logarithmic texel density distribution
  - Shadow frustums adjusted to cover the camera view frustum conservatively
  - Orientation for shadow frustums is fixed in world space
- Having more cascades allows
  - Improved texel density, reduced aliasing and improved self-shadowing for wider shadow range due to a better approximation of the logarithmical distribution
- For each cascade snap the shadow frustum to the SM’s texel grid



Deferred Shadow Passes

- Shadow passes for cascades/point lights are rendered in a deferred way
- Potential shadow-receiving areas are tagged in the stencil buffer by rendering frustum volumes
  - Allows to have a more sophisticated splitting into cascades
  - Picks a cascade with the highest resolution in overlapping regions
  - Uses shadow map space more efficiently



Shadow Cascades Caching

- Not all the cascades are updated in one frame
  - Update cost distributed across several frames
  - Performance reasons
- Allows us to have more cascades – better shadow map density distribution
- Distant cascades are updated less frequently
- Cached Shadow Maps are not updated but are used for shadowing
- Can handle dynamic objects with additional memory
- Last cascade uses VSM and blends additively with the shadow mask
  - Allows to have large penumbras from huge distant objects

Point Light Shadows

- We always split omni-directional lights into six independent projectors
- Shadow map for each projector is scaled separately
  - Based on the shadow projection coverage
  - Final scale is a result of
    a logarithmic shadow map density distribution function,
    which uses the coverage as a parameter
- Use cascades for large projectors
- Texture atlas to pack all shadow maps
  each frame after scaling
  - Texture atlas is allocated permanently to avoid
    memory fragmentation
  - Size on consoles: 1024x1024 (4 MB)
- Receiving areas tagged by stencil

Shadow atlas

Per–Object Shadow Maps

- Used to increase self-shadowing details in cutscenes and for very large and detailed objects in game (first person weapon)
- Separate hi-resolution shadow map for dedicated objects
- Global and per-objects shadows are blended to the shadowmask with a max() filter.
- Huge per-object shadow map bias to eliminate low-res self-shadowing with global shadow maps (CSM, point lights)
  - Objects still cast global low-resolution shadows
  - Self-shadowing comes from hi-resolution shadow map only

Per–Object Shadows

Per-Object Shadows

Per–Object Shadows

First Person Weapon Self-Shadowing

- Different first-person and third-person models for rendering and global shadow map generating
  - Proper self-shadowing is achievable only with separate per-object shadow maps

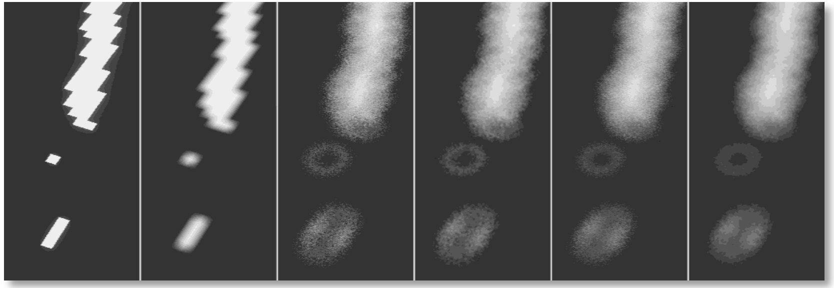
First-Person Weapon Self-Shadowing

- Problem with deferred shadowing
  - uses different view frustum (near/far planes, FOV)
  - General case - need to re-project weapon depth from the “weapon” space
  - When FOV difference is not large – approximate re-projecting with simple depth re-scaling

Soft Shadows Approximation

Soft Shadows Approximation

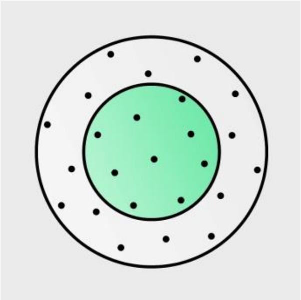
- We use Poisson PCF taps with randomized rotations in shadow space



- Adjusting the kernel size at runtime gives a good approximation of soft shadows with variable penumbra
- Soft shadows idea: Estimate average distance ratio to shadow casters
  - Similar to Percentage-Closer Soft Shadows [Randima05]

Soft Shadows Approximation

- Basic Algorithm:
  - Poisson-distributed taps are presorted by distance from the kernel center
  - Initial kernel radius set to match the maximum range (= largest/longest penumbra)
  - Use this kernel to estimate the average distance ratio
  - The amount of samples is reduced proportionally to the avg. distance ratio
    - This affects the radius of the kernel since the taps are sorted
  - Use only the reduced amount of samples for final shadow computation
- Cascade shadow maps need custom kernel scale adjustment
  to handle transitions between cascades
- Compute Shader option: fetch all taps to CS shared memory
  and reuse them for both distance estimation and shadow computation



Area Lights Shadowing

No light

Area Lights Shadowing

Simple soft-shadows approximation

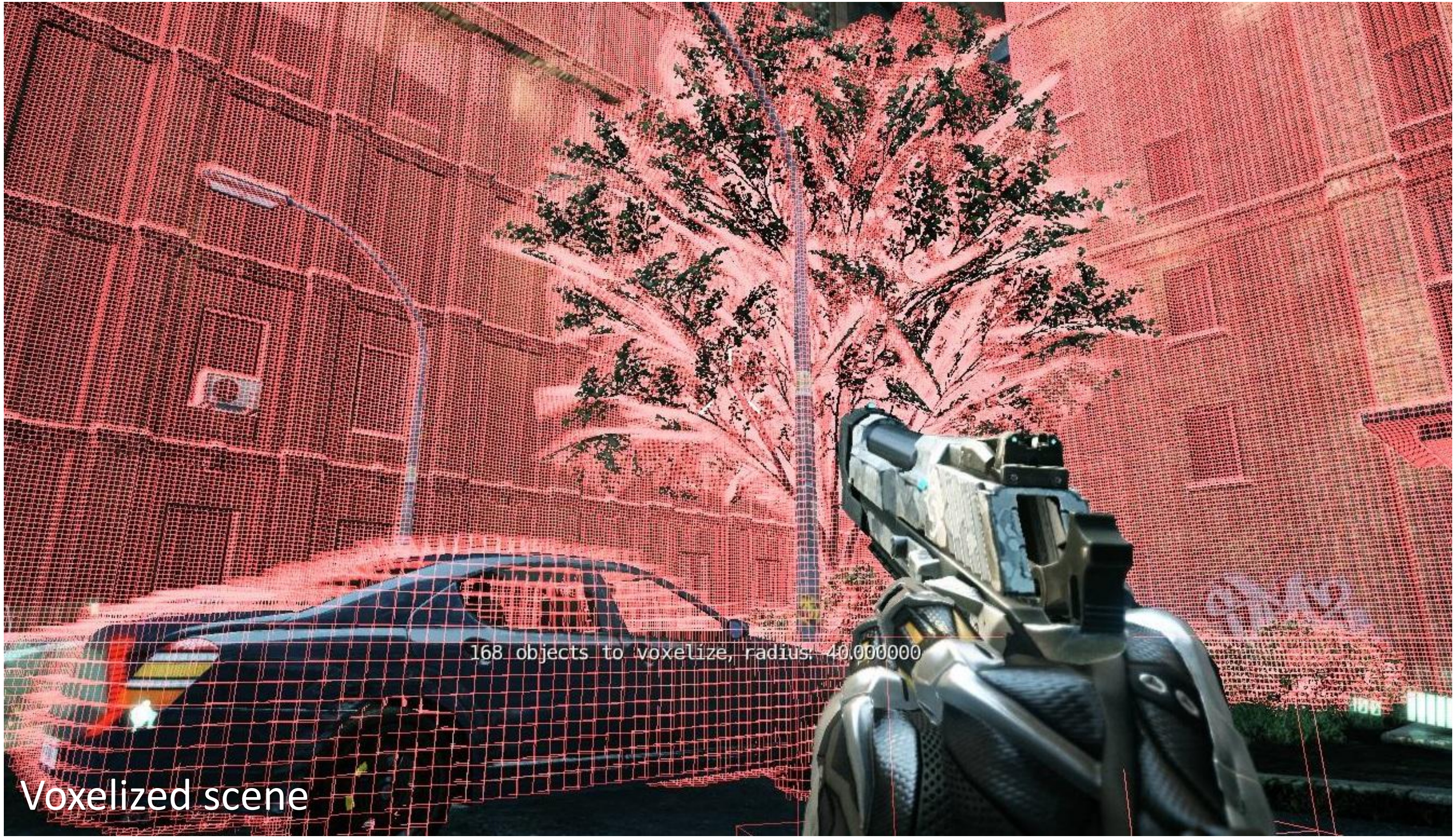
Area Lights Shadowing

Voxel based area light shadowing

Area Light Shadows

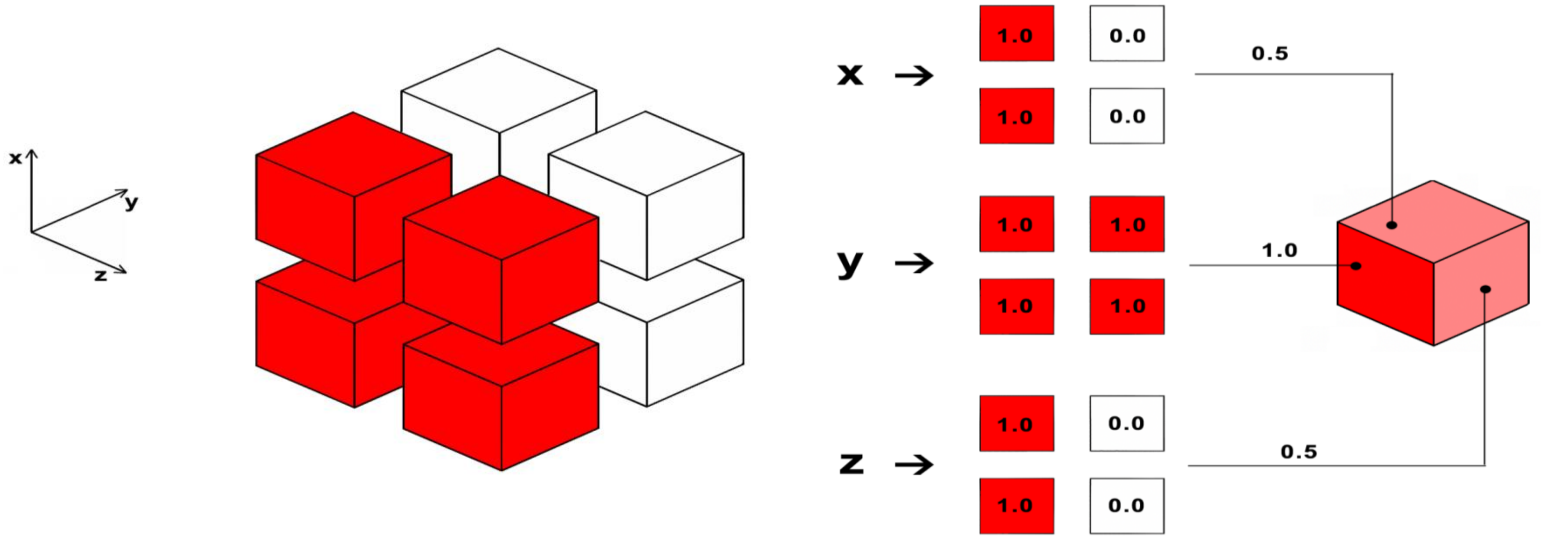
- Multi-resolution uniform voxel data
  - Efficient occlusion sampling for very large volumes
  - Adaptive resolution for ray traversal
  - Multiple distance-based cascades are an option
- Dynamic surface voxelization and downsampling
  - Downsampling “directional occlusion” values
- Adaptive downsampling
  - Avoids updating static parts of the scene each frame
  - Bit-masked change-aware downsampling (XOR with the previous frame’s voxel data)

Area Light Shadows

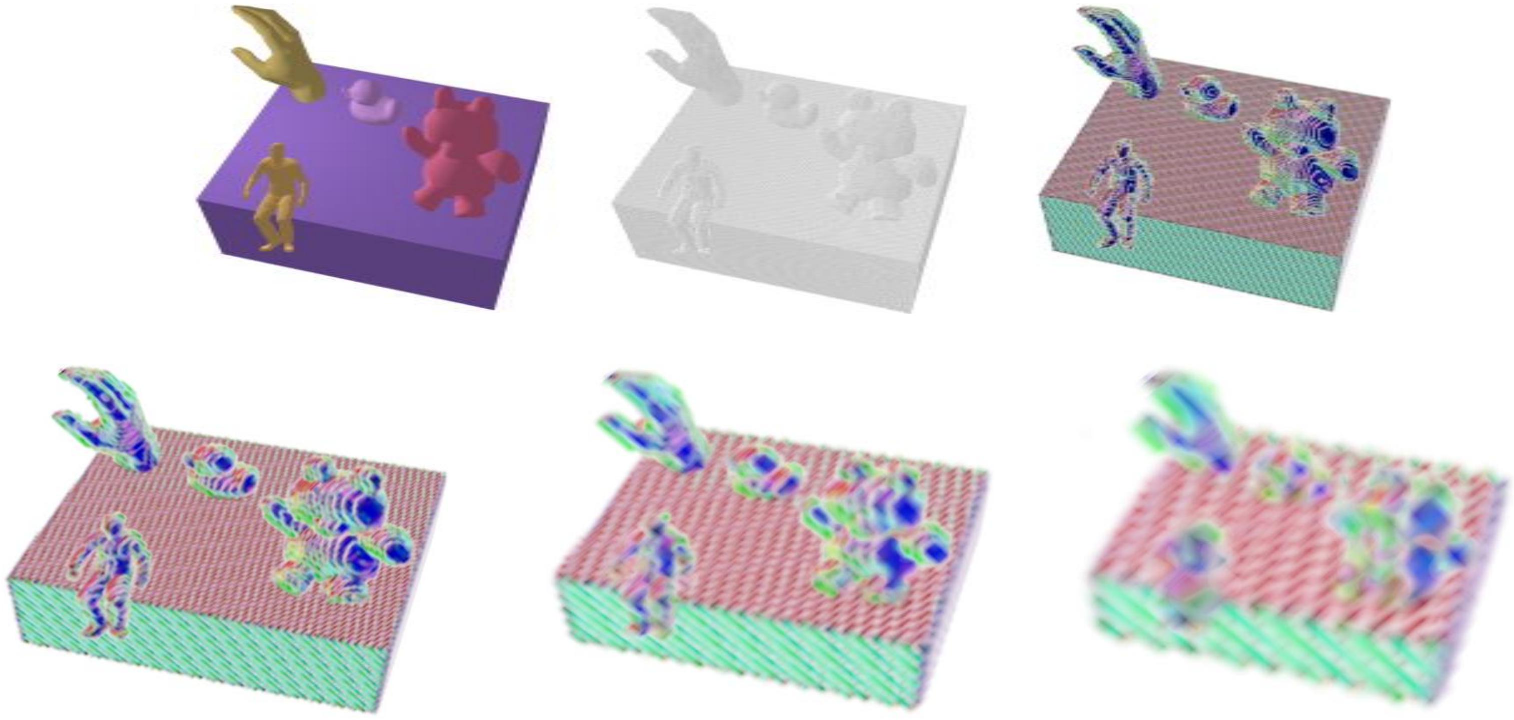


Voxel Data Downsampling

- Directional occlusion Concept
  - Downsample light occlusion
  - Bi-directional; 3 component

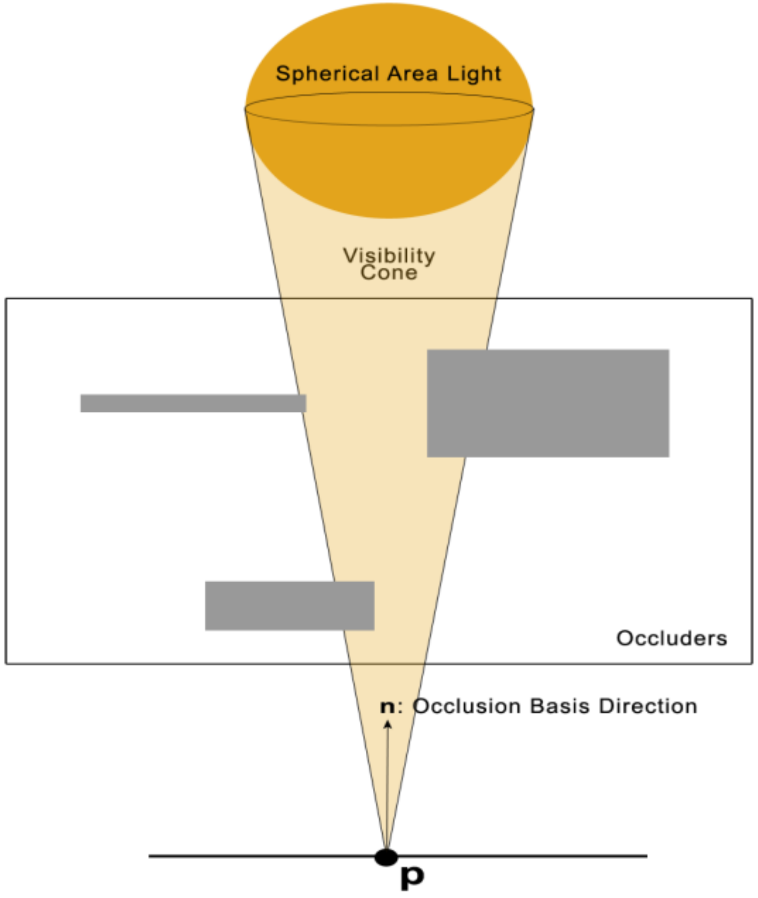


Voxel Data Downsampling



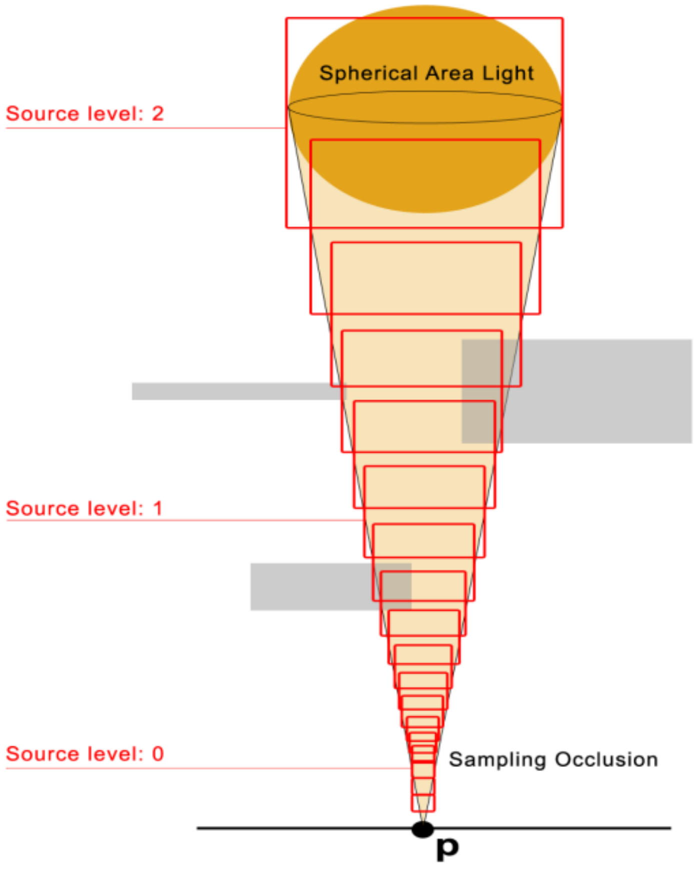
Area Light Shadows: Cone Tracing

- Approximation of grouped rays



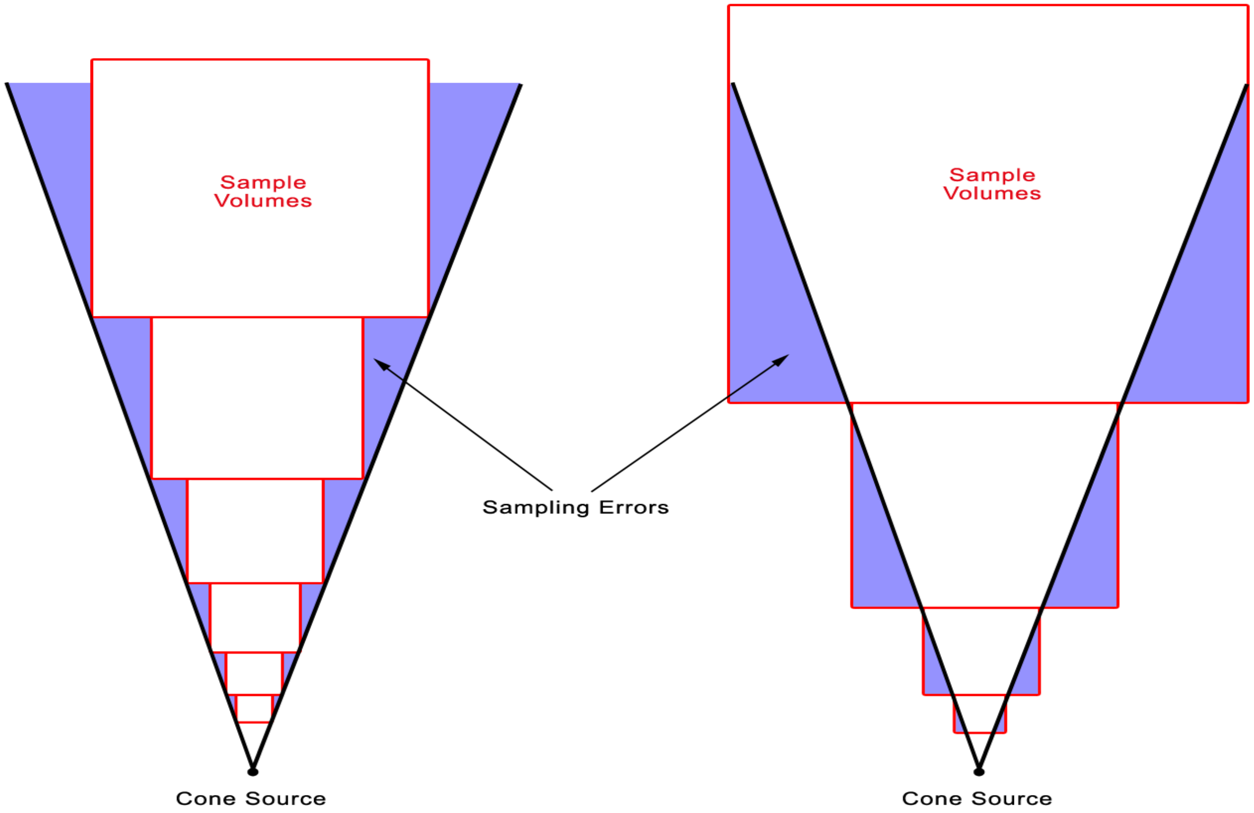
Area Light Shadows: Cone Tracing

- Sample different Voxelization levels
- Adjust voxel level along the ray
- Directional Occlusion Gathering



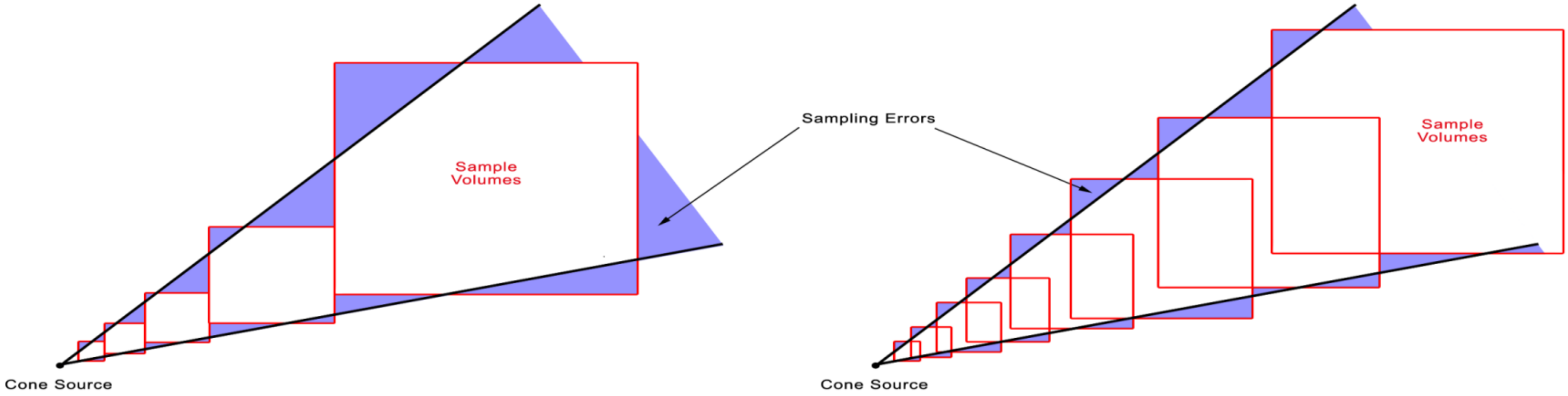
Area Light Shadows: Cone Tracing

- Sampling Errors

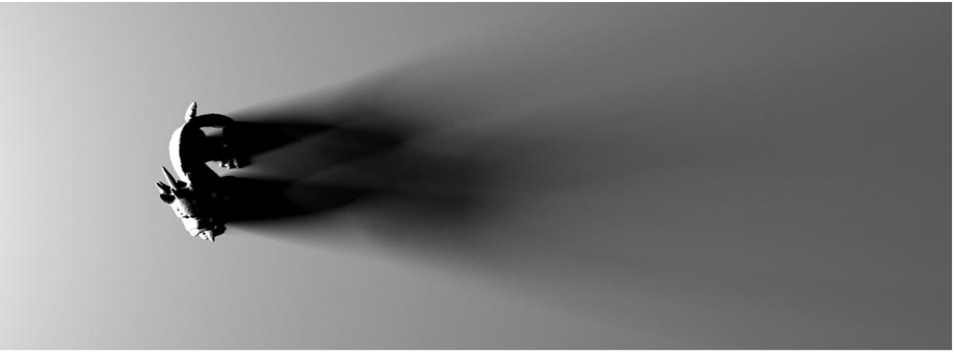


Area Light Shadows: Cone Tracing

- Sampling Errors



Area Light Shadows

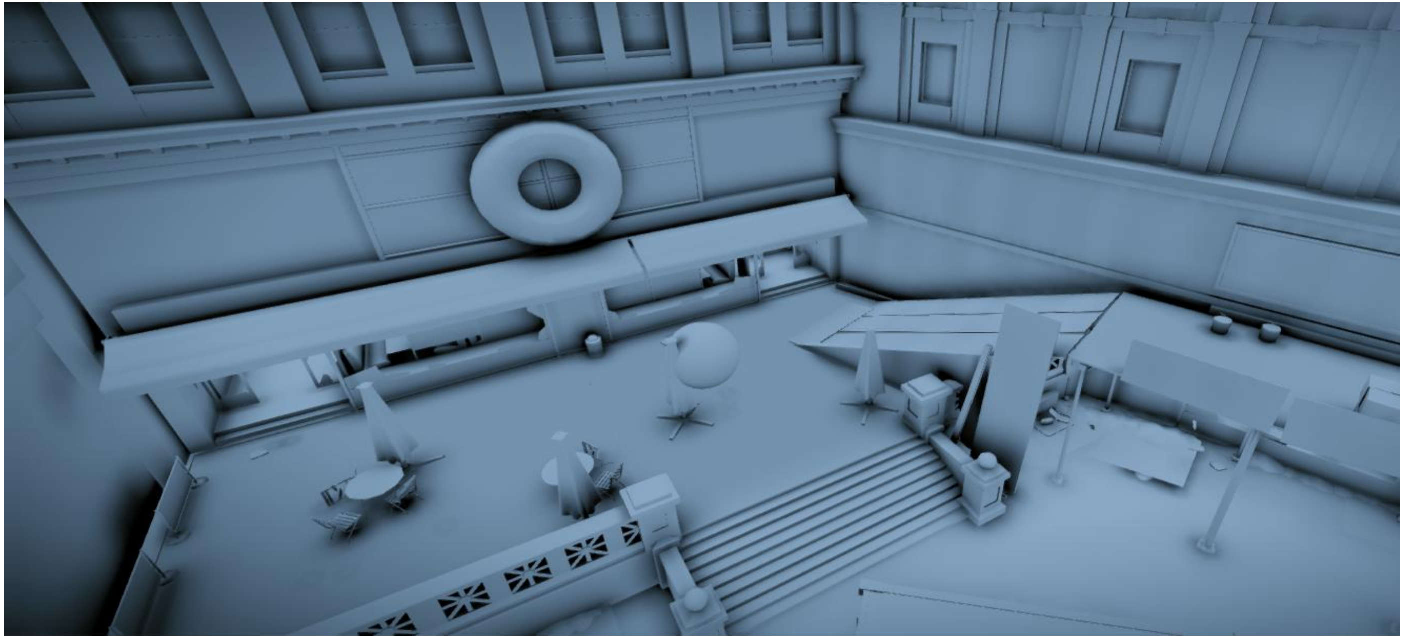


Ray-traced Spherical Area Light
Autodesk Maya 2012 ~40s

Real-time Area light shadows ~20ms
(Voxelization + Cone Tracing)

Area Light Shadows

- World Space Ambient Occlusion with cone tracing
  - Approximated with 8 uniformly distributed cones



Shadows & Transparency

- For alpha blended shadow receivers
  - Forward passes to apply shadows
- For transparent shadow casters(e.g. hair, smoke) we accumulate alpha values of the casters
  - Stored in a 8-bit render target

Shadows & Transparency

- Translucency map generation:
  - Depth testing using depth buffer from a regular opaque shadow map to avoid back projection/leaking
  - Transparency alpha is accumulated only for objects that are not in “opaque” shadows
  - Alpha blended shadow generation pass to accumulate translucency alpha (sorted back to front)
  - In case of cascaded shadow maps, generate translucency map for each cascade
  - Shadow terms from shadow map and translucency map are both combined during deferred shadow passes with max() operation

Contact Shadows/SSDO

- Contact Shadows/SSDO
  - Applied to all light sources and ambient, via screen space bent normals (average unoccluded direction)

SSDO off

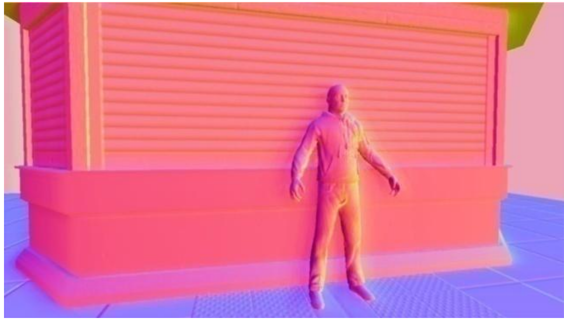
SSDO on

Contact Shadows

- Core idea the same as SSDO [Ritschel 2010]
  - Modulate lighting with computed screen space occlusion
  - Produces soft contact shadows
  - Can also hide shadow bias issues
  - Considerable quality gain over just SSAO
- Directional occlusion information is accessible in a deferred way
  - Fits better into the existing lighting pipeline
  - Can be applied efficiently to every light source

Contact Shadows

- Occlusion information generation
  - Compute and store bent normal N' during SSAO pass
    - Bent normal is average unoccluded direction
  - Requires clean SSAO without any self-occlusion and a relatively wide radius
- For each light
  - Compute N dot L as usual
  - Compute N' dot L
  - Center depth is full resolution, all other taps are FP16 half-resolution depth
  - Attenuate lighting with the occlusion amount multiplied by a clamped difference between the two dot products



Screen Space Self-Shadowing

- Simple trick/approximation
  - Ray casting along screen space light vector
  - For cutscenes specify the affected depth buffer range
  - Ray length tracking allows to even compute proper soft shadowing

Volumetric Fog Shadows

- Based on TÓTH09: accumulate not in-scattered light but shadow contribution along the view ray instead

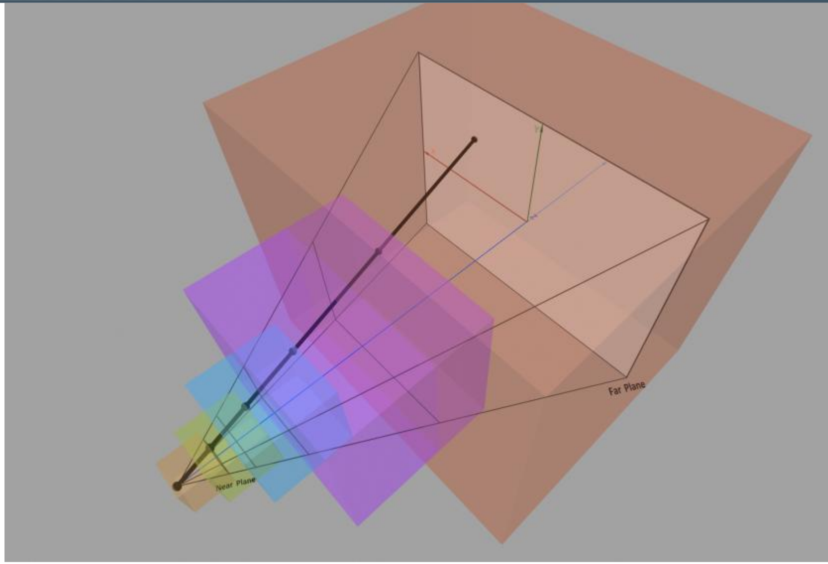
Volumetric Fog Shadows

- Ray casting in shadow space
- Interleave pass distributes shadow samples along the view direction on a 8x8 grid shared by neighboring pixels
    - Half-resolution destination target for performace
- Gather pass computes final shadow value
    - Bilateral filtering was used to minimize ghosting and halos
    - Shadow stored in alpha, 8 bit depth in red channel
    - Used 8 taps to compare against center full resolution depth
- Max sample distance configurable *(~150-200m in C3 levels)*
- Cloud shadow texture baked into final result
- Final result modifies fog height and radial color

Volumetric Fog Shadows: Naive Upscale

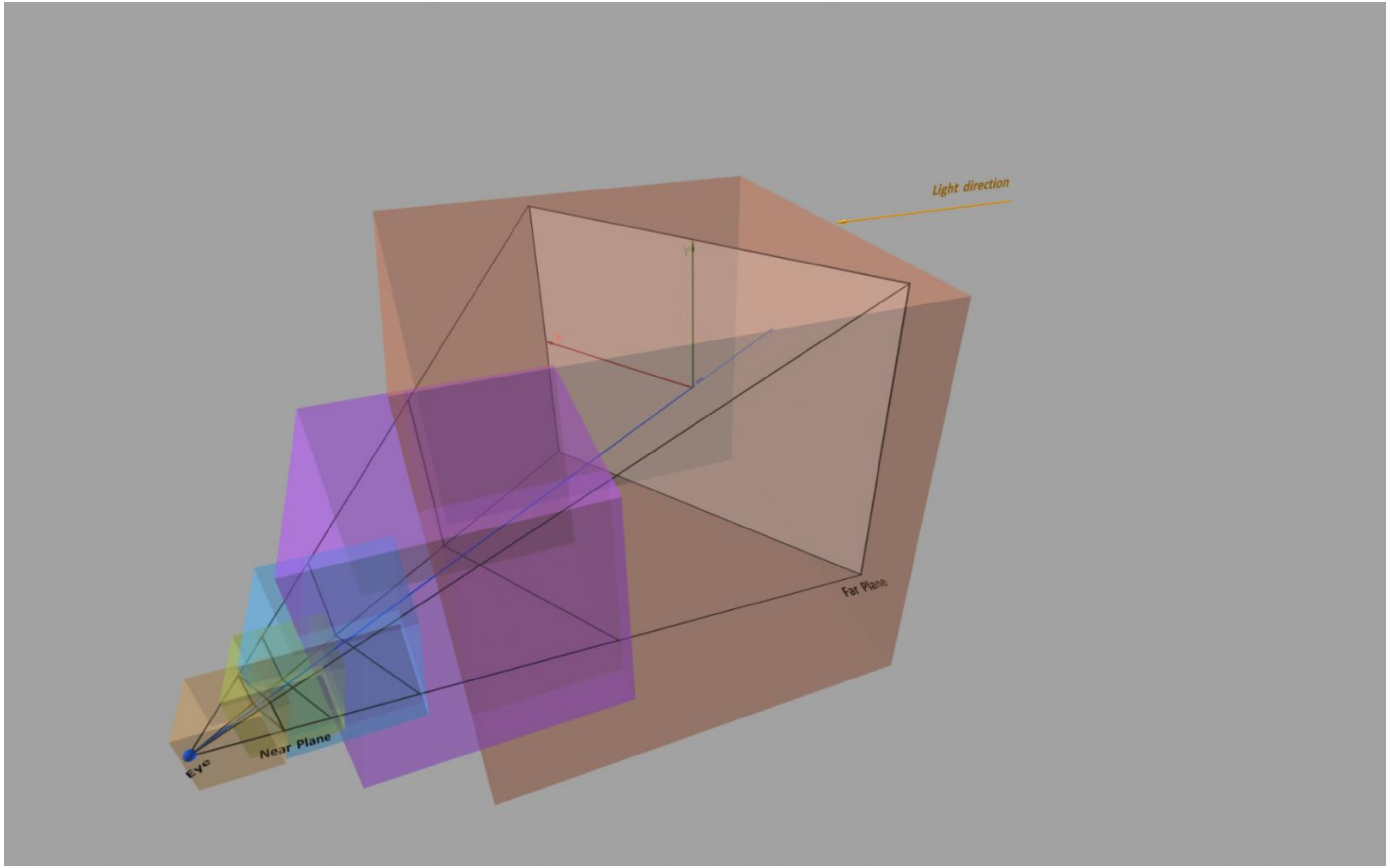
Volumetric Fog Shadows: Bilateral Upscale

Volumetric Fog Shadows



- Cascaded shadow map ray casting
  - Ray casting happens in shadow space
  - Adaptive ray casting - more samples taken in the near space
  - Arbitrary shadow frustums for cascades
  - Frustums are overlapped
  - Always pick the cascade with the highest sampling density
  - Use global parametric coordinate to store the current ray’s intersection points with the shadow frustum
  - No need to re-project between cascades
  - Optimized ray clip function that directly modifies the global parametric coordinate

CSM frustum split schemes analysis



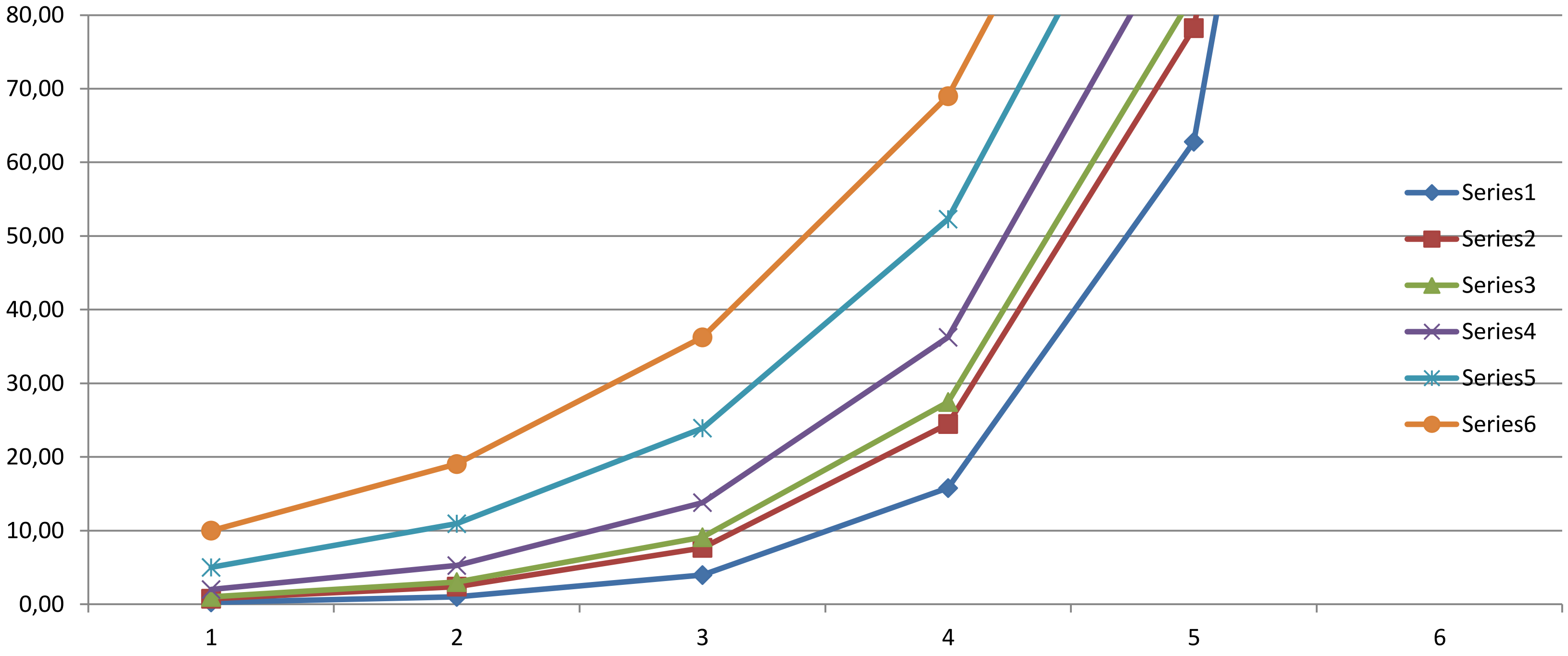
Logarithmical Split Scheme

- Multiple split schemes for variable CSM's bounds

	zn	zf	split0	split1	split2	split3	split4	split5	
	0,25	250	0,25	1,00	3,96	15,77	62,80	250,00	
	0,75	250	0,75	2,40	7,66	24,48	78,23	250,00	
	1	250	1,00	3,02	9,10	27,46	82,86	250,00	
	2	250	2,00	5,25	13,80	36,24	95,18	250,00	
	5	250	5,00	10,93	23,91	52,28	114,33	250,00	
	10	250	10,00	19,04	36,24	68,99	131,33	250,00	
	0,25	50	0,25	0,72	2,08	6,01	17,33	50,00	
	0,25	100	0,25	0,83	2,75	9,10	30,17	100,00	
	0,25	150	0,25	0,90	3,23	11,61	41,73	150,00	
	0,25	200	0,25	0,95	3,62	13,80	52,53	200,00	
	0,25	250	0,25	1,00	3,96	15,77	62,80	250,00	

Typical Logarithmical Split Scheme

- Splits overlapping dependency on the CSM's near bound



Cascaded Shadows Maps Split Consideration

- Cascaded splits‘ frustums overlapping
  - Using accurate logarithmic distribution is difficult for the splits that are close to the near plane
  - Closest splits are adjusted manually
- Efficient cascades splitting is very sensitive to the camera’s near plane and **FOV**
  - Larger FOV increases shadow frustums overlapping
  - Larger FOV increases shadow map waste on invisible parts of the scene
  - Closer near plane increases shadow frustums overlapping
- Tighter CSM bounds for cutscenes with limited depth range

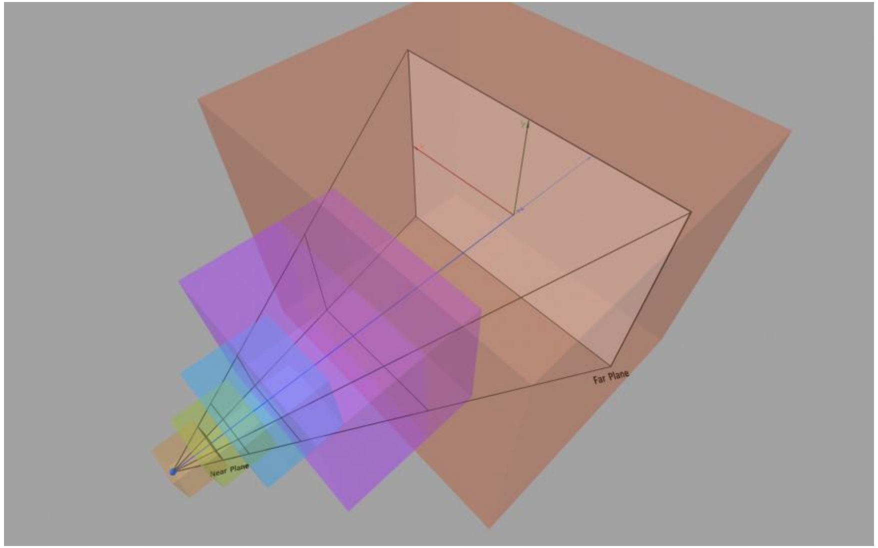
Tighter CSM Bounds for Cutscenes

Tighter CSM Bounds for Cutscenes

Tighter CSM Bounds for Cutscenes

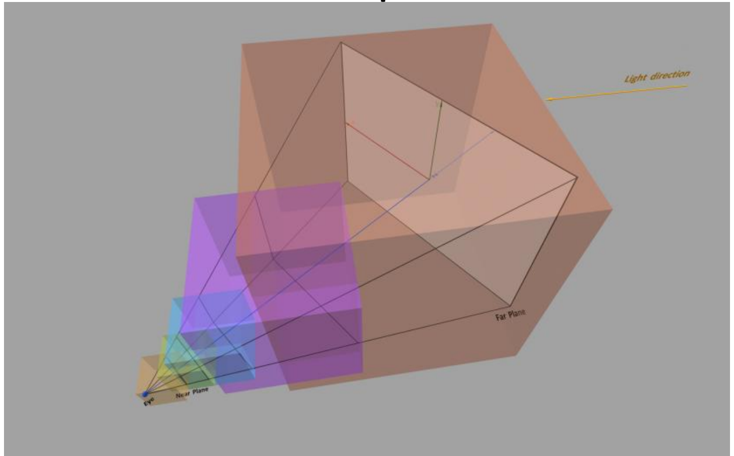
Light Space vs. View Space Shadow Frustums Alignment

- View space aligned
  - Better shadow space usage
  - Less frustum overlapping
  - Higher Shadow map sampling density
  - Shadows are not stable in case of shadowmap under-sampling (shadow aliasing - shimmering for moving camera)



Light Space vs. View Space Shadow Frustums Alignment

- Light space aligned shadow frustums
  - Less efficient shadow map usage due to increased frustums overlapping
  - More efficient for Shadow Cascades Caching
  - Allows to use shadow map texel size snapping
  - Stable shadows for under-sampled shadow maps with moving camera



Shadow Aliasing with Cascaded Shadow Maps

- Influencing factors:
  - Low shadow map sampling density
  - Precision of the depth buffers
  - Direction of the light source relative to the camera

Shadow Aliasing

- Different scenarios to overcome aliasing
  - Sun shadows: front faces rendered with slope-scaled depth bias
  - Point light shadows: back face rendering, works better for indoors
  - Variance shadows for distant LODs - render both faces to shadow maps
- Constant depth bias during deferred shadow passes to overcome depth buffer precision issues

Current Situation

- Mostly undersampled shadows are used in games nowadays
- Cascades splitting is not efficient
- Tricks like shadowmap texels snapping, per-object shadows
- Per-level/cutscene tweaked solutions

Main Goals – What We Are Trying to Achieve

- Eliminate/minimize overlapping of cascaded shadow map frustums
- Eliminate/minimize unused regions in the shadow map
- Minimize shadow map waste on invisible parts of the scene
- Aim for the very high shadow map sampling density – makes tricks like shadow map texel snapping unnecessary
- Guarantee close to constant shadow map sampling density for all regions of the scene
  - Having close to constant shadowmap density helps to address the shadow aliasing problem

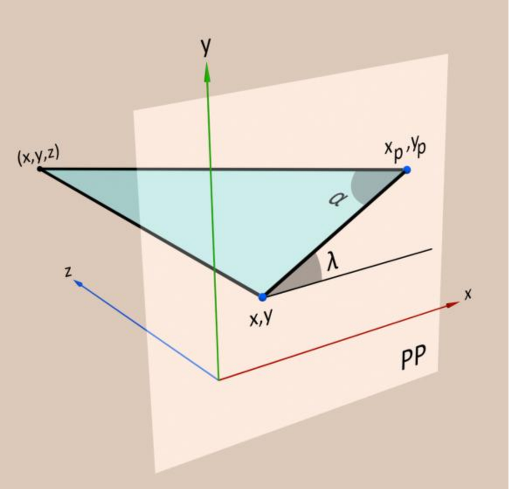
Oblique Projection for Cascaded Shadow Maps

- A type of parallel projection
- Projects an image by intersecting parallel rays ("projectors") from a three-dimensional source object with a target projection plane
- The projectors are not perpendicular to the projection plane

Oblique Projection for Cascaded Shadow Maps

- The projectors are defined by the two angles α and λ where
  α is the angle between the line (x,y,xp,yp) and the projection plane,
  λ is the angle between the line (x, y, xp, yp) and the x axis on the projection plane
  L = the length of the line (x,y,xp,yp). L1 = L / z

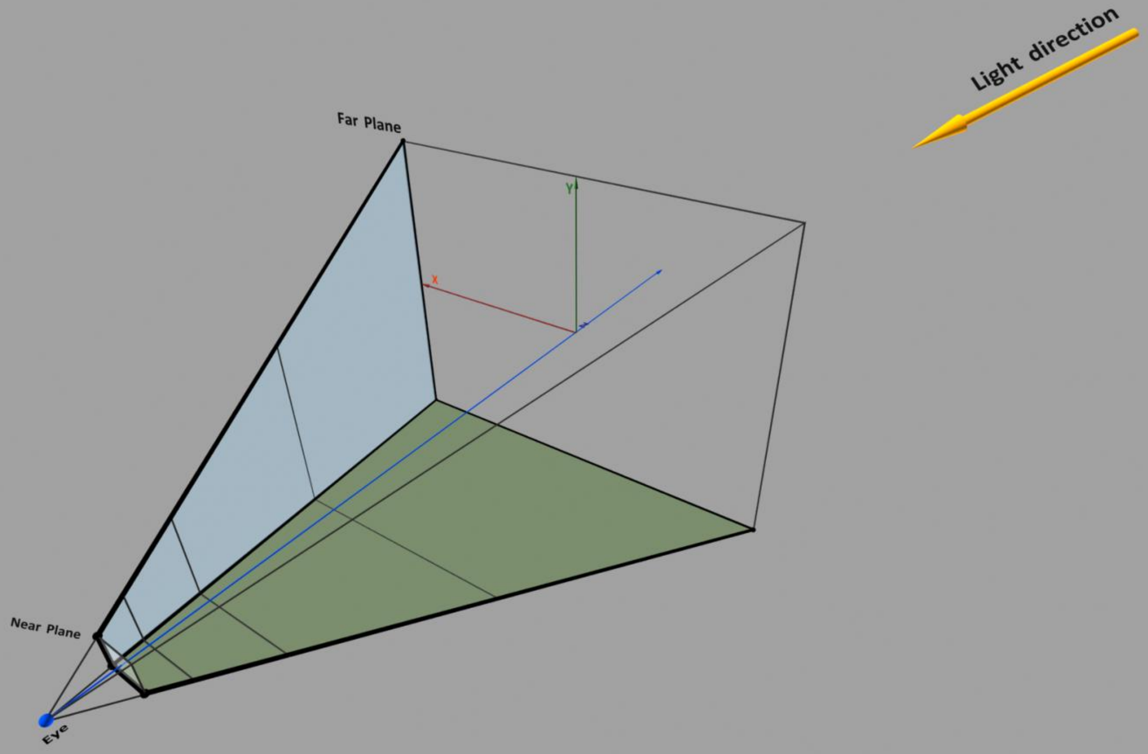
$$P\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ L1\cos\alpha & L1\sin\alpha & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Oblique Projection for Cascaded Shadow Maps

- Use oblique projectors
- Use view frustum clip planes as a shadowmap projection planes
- Projection planes are selected from the 5 view frustum planes (Far plane is irrelevant)
- Oblique projection planes for shadow projections are selected based on the light direction
    - Select planes that have the same sign of the dot product between the plane normal and the light direction as the nearest plane

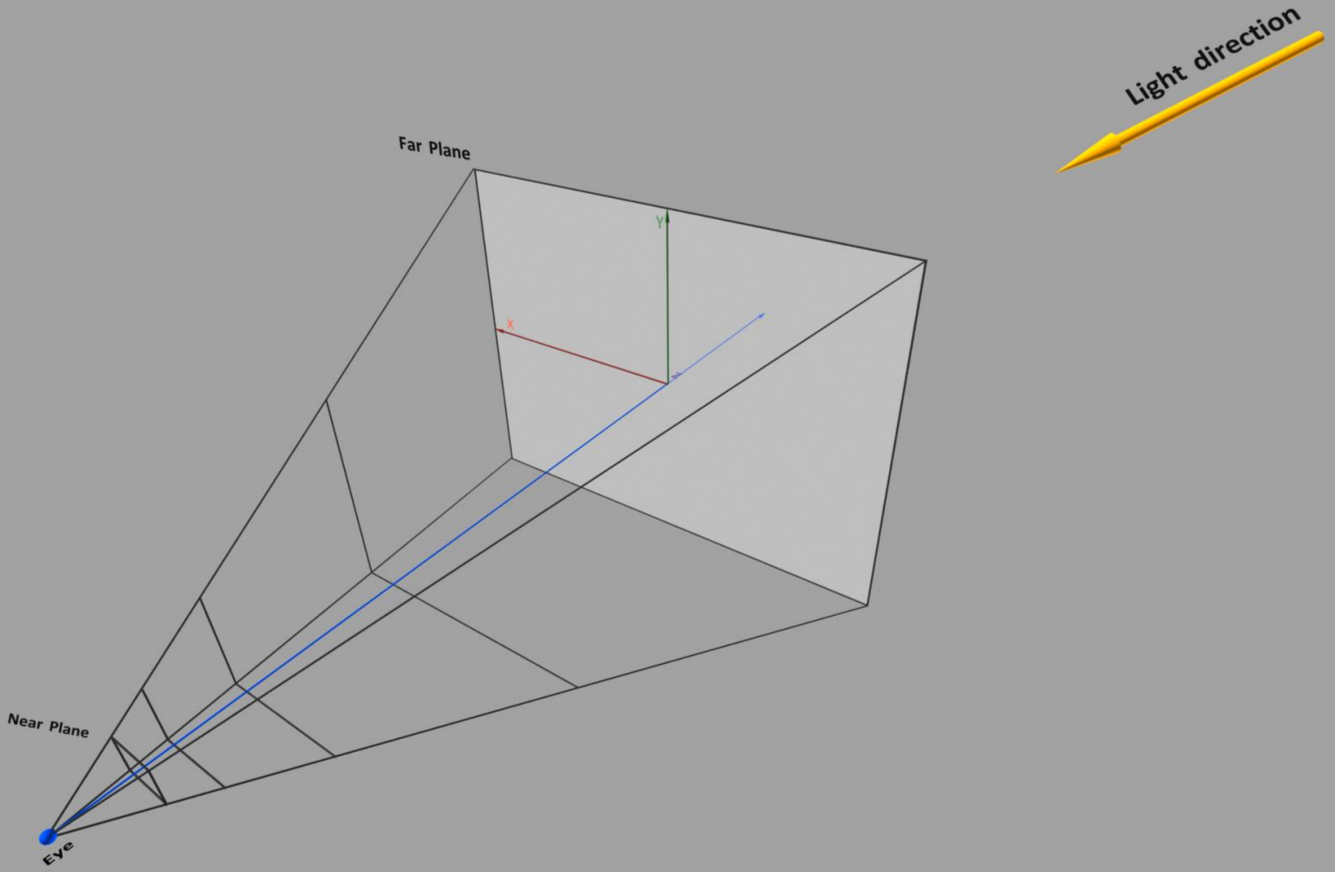
Planes Selection



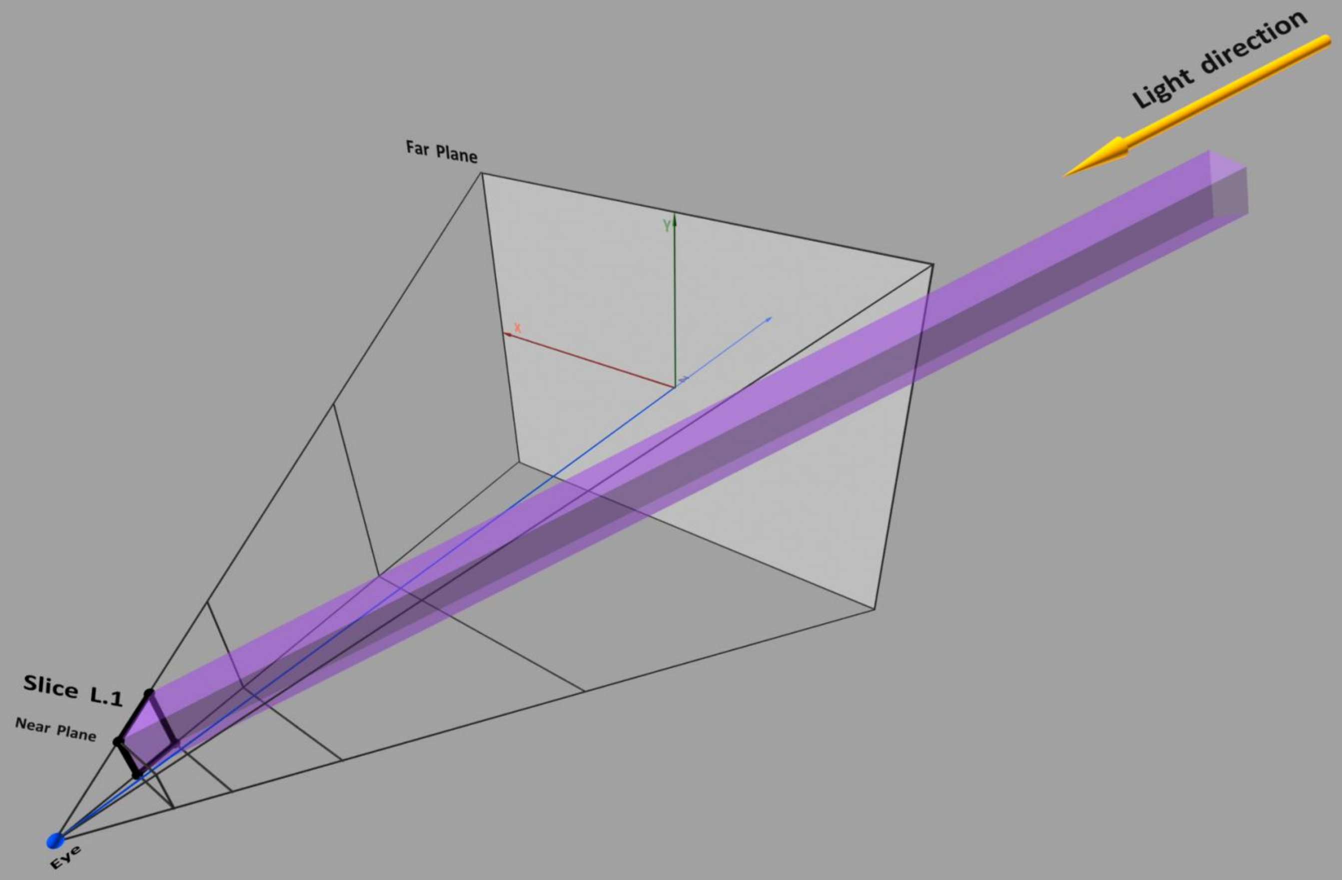
Oblique Projection for Cascaded Shadow Maps

- Projection planes are split into segments to get an approximation of a logarithmic distribution
  - Plane segments are essentially shadowmap cascades
- Faraway segments cover more area with the same shadow map resolution
- CPU culling of shadow casters is performed with a set of oblique frustums

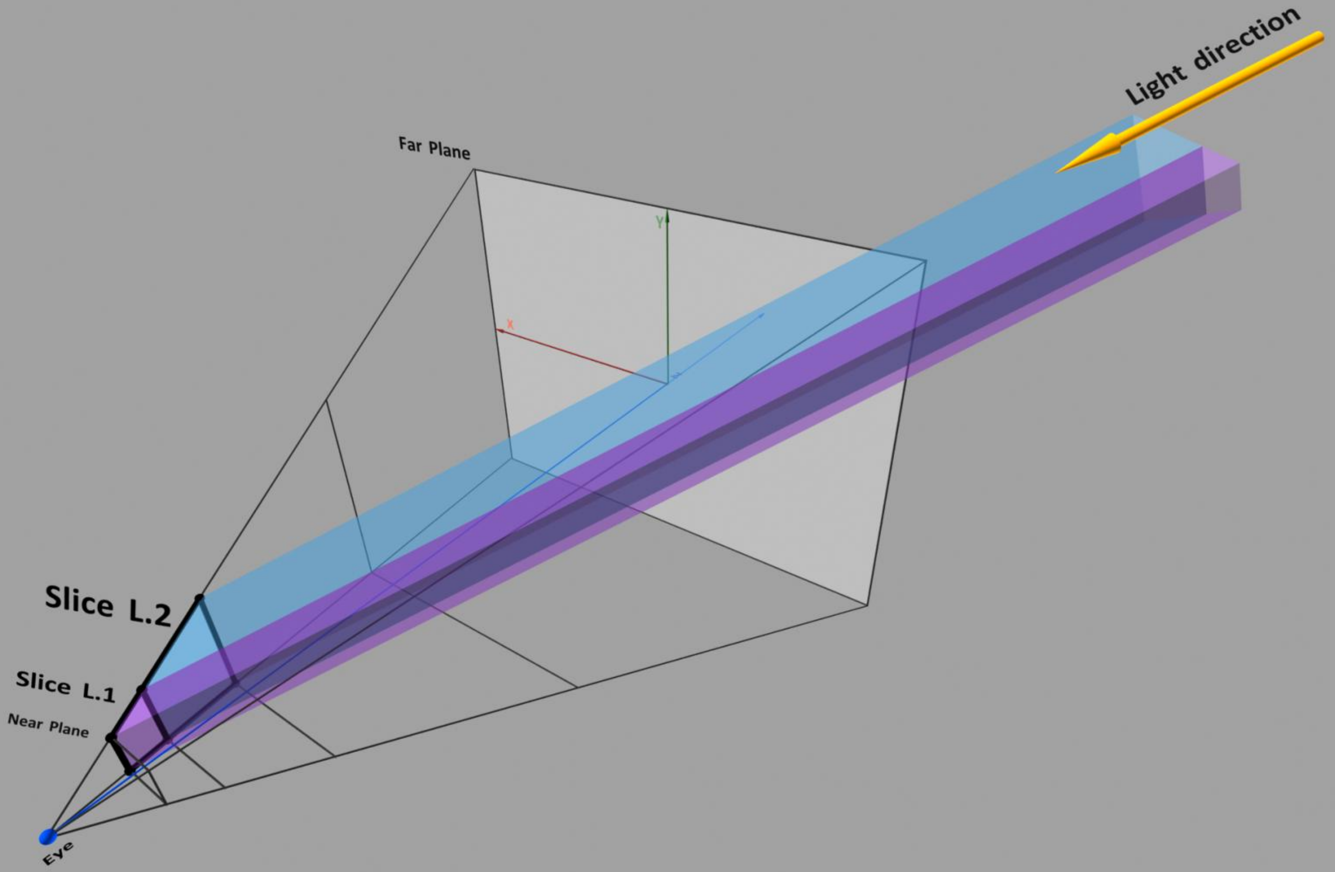
Plane Segments Selection



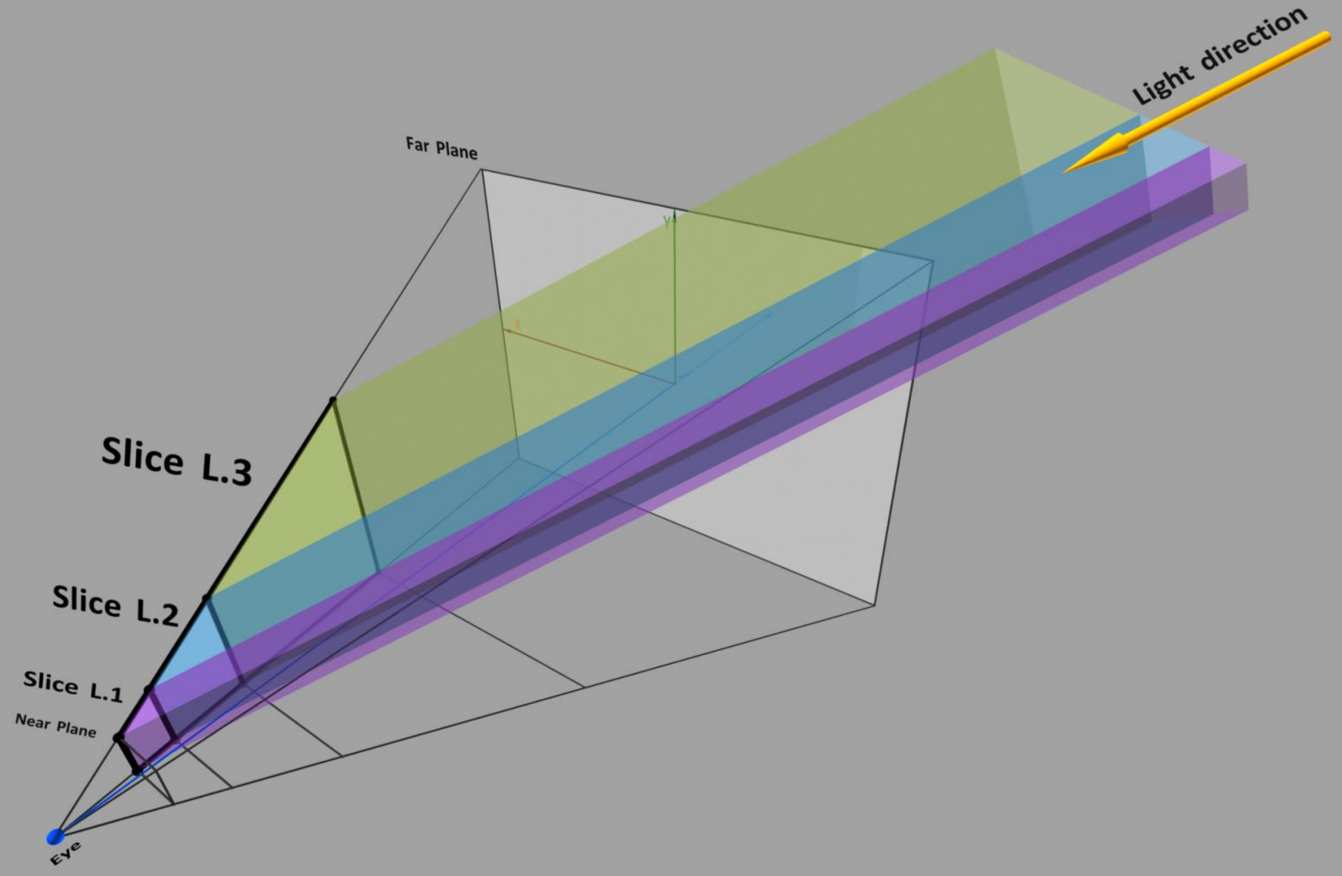
View Frustum Slices (Left Plane)



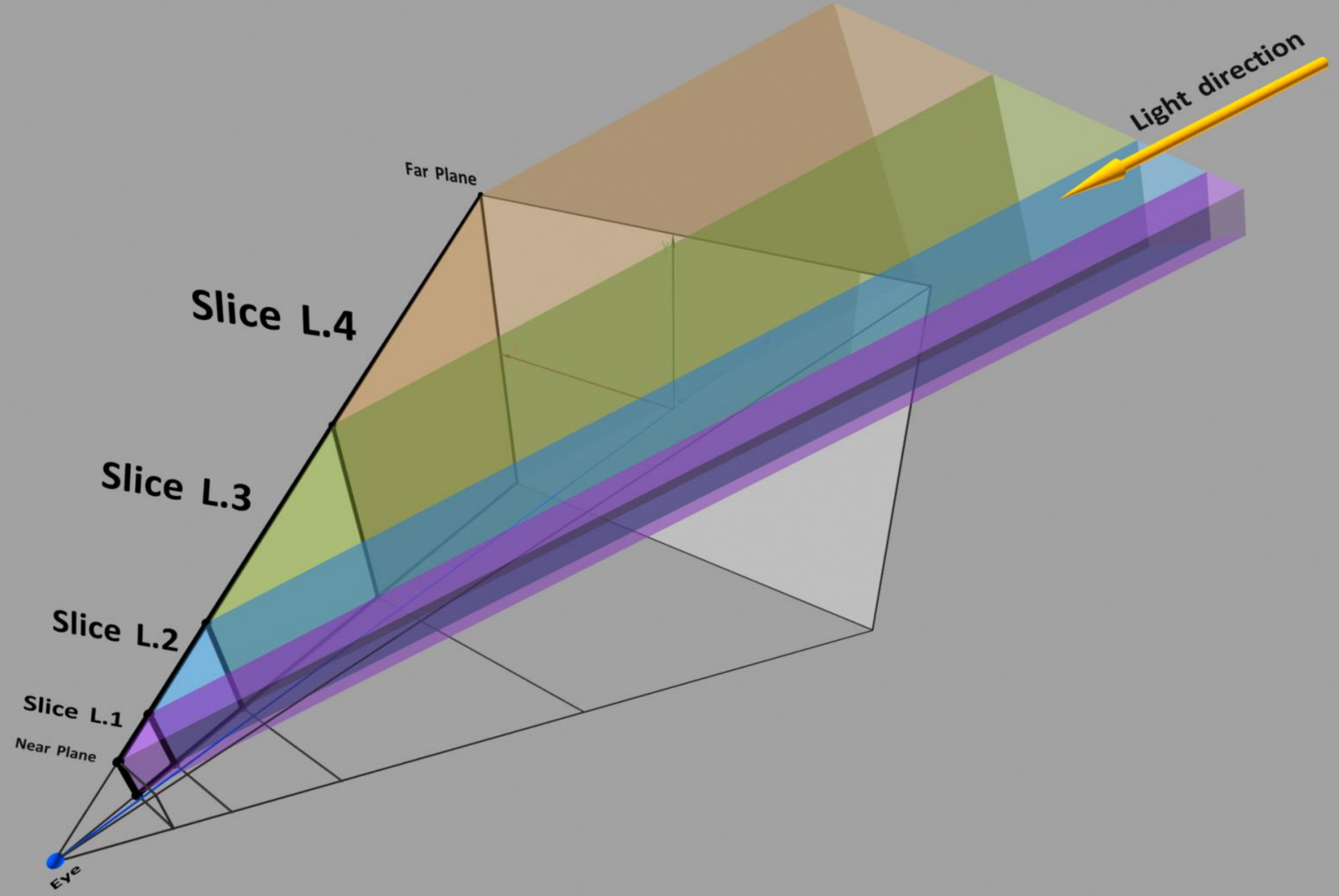
View Frustum Slices (Left Plane)



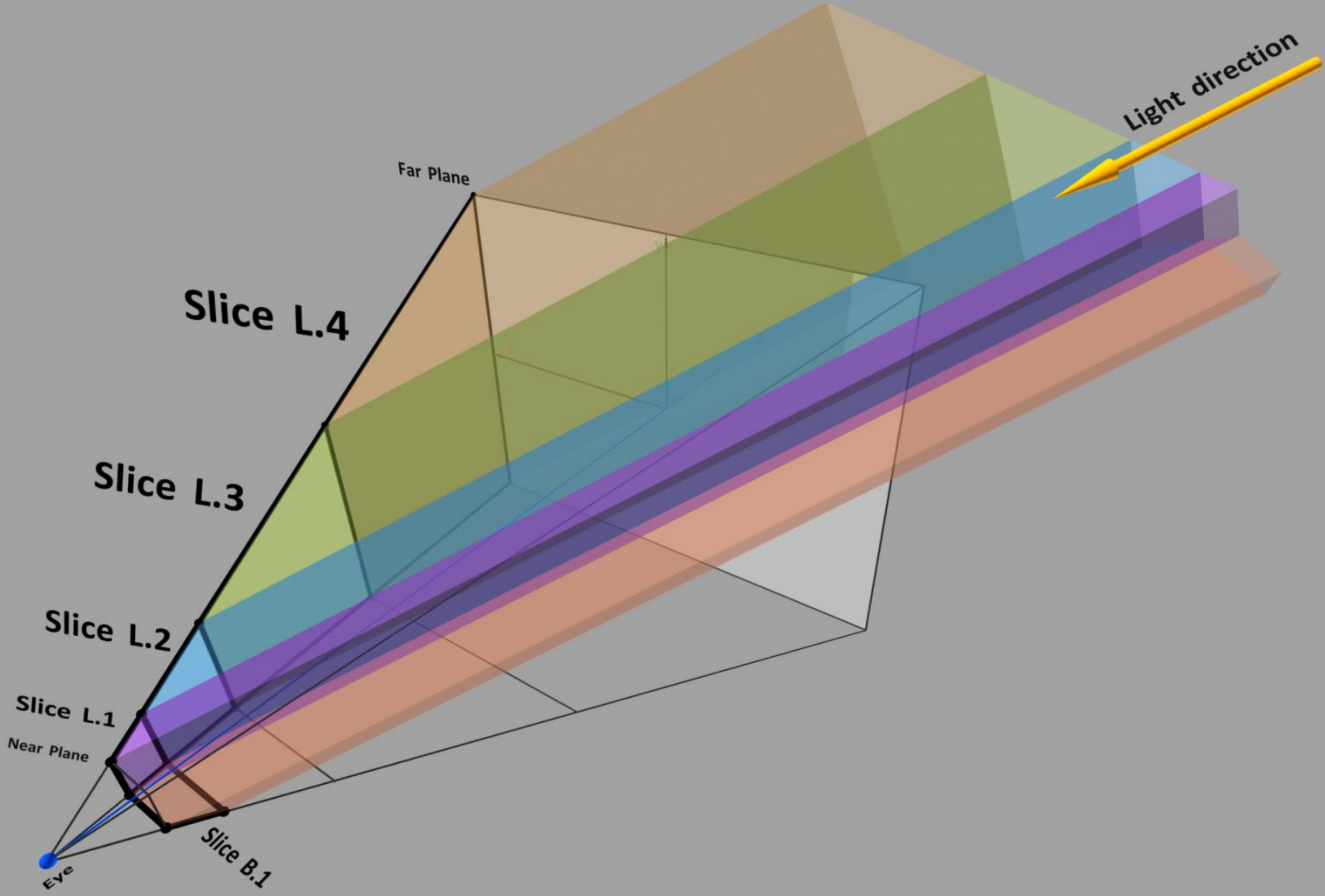
View Frustum Slices (Left Plane)



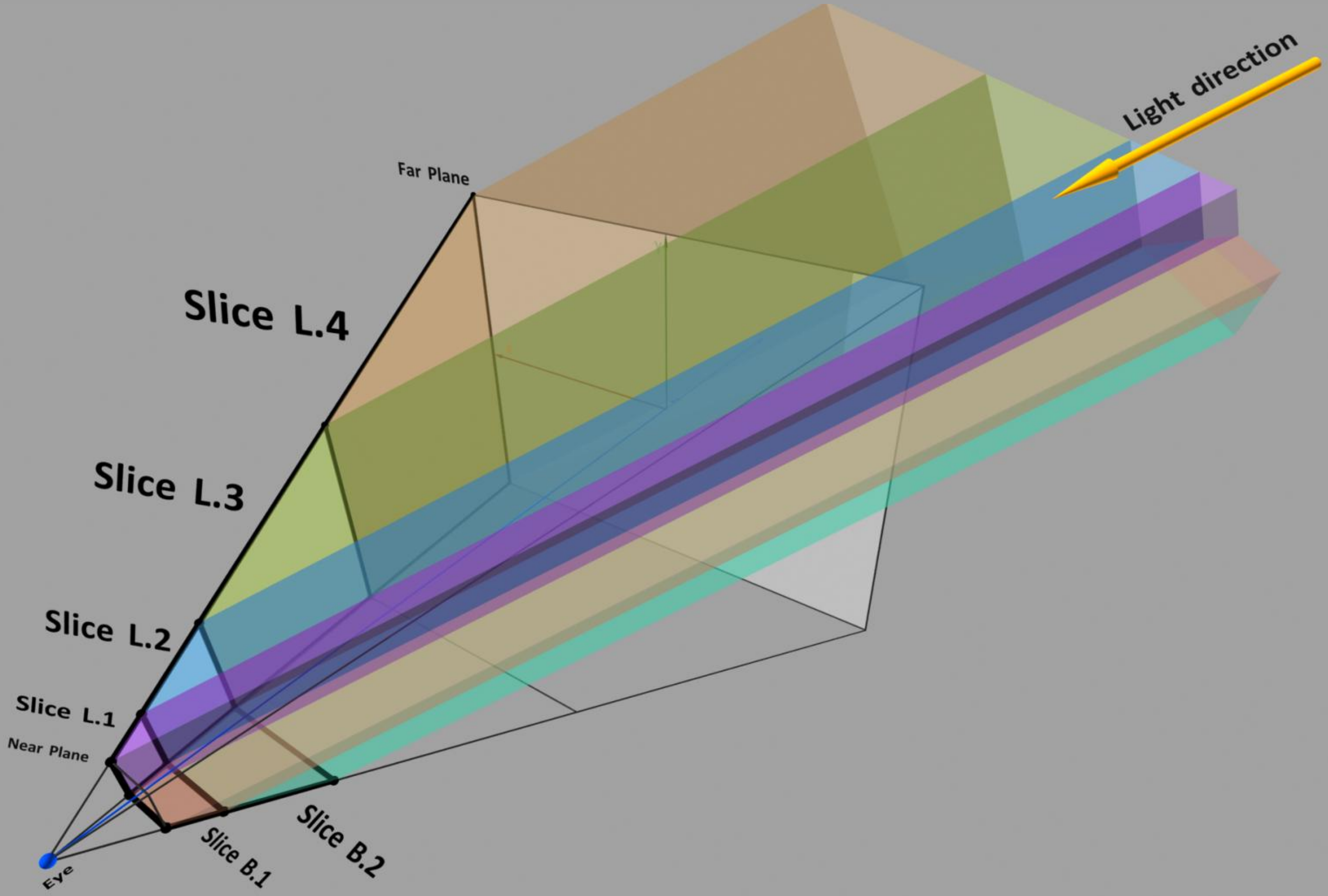
View Frustum Slices (Left Plane)



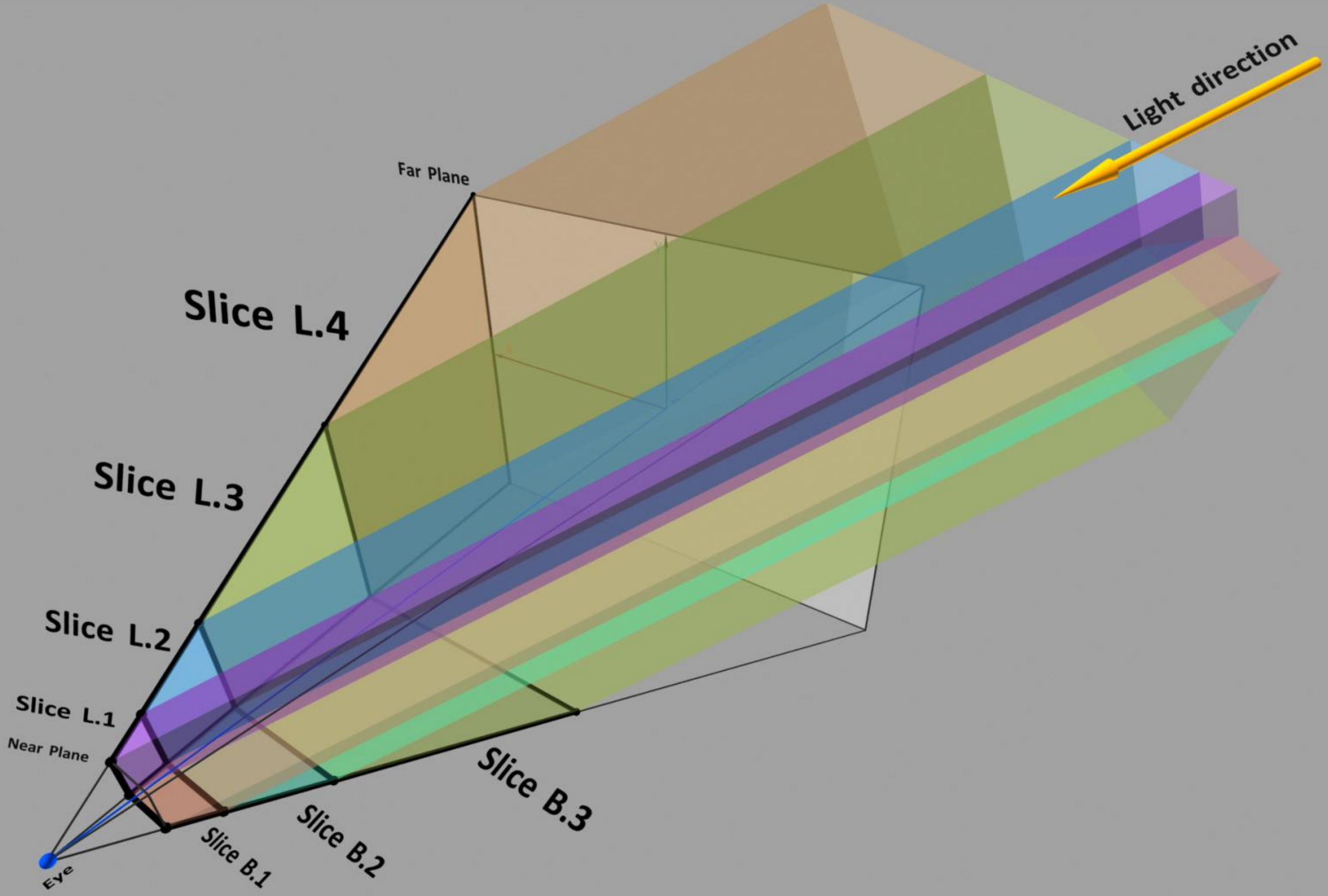
View Frustum Slices (Bottom Plane)



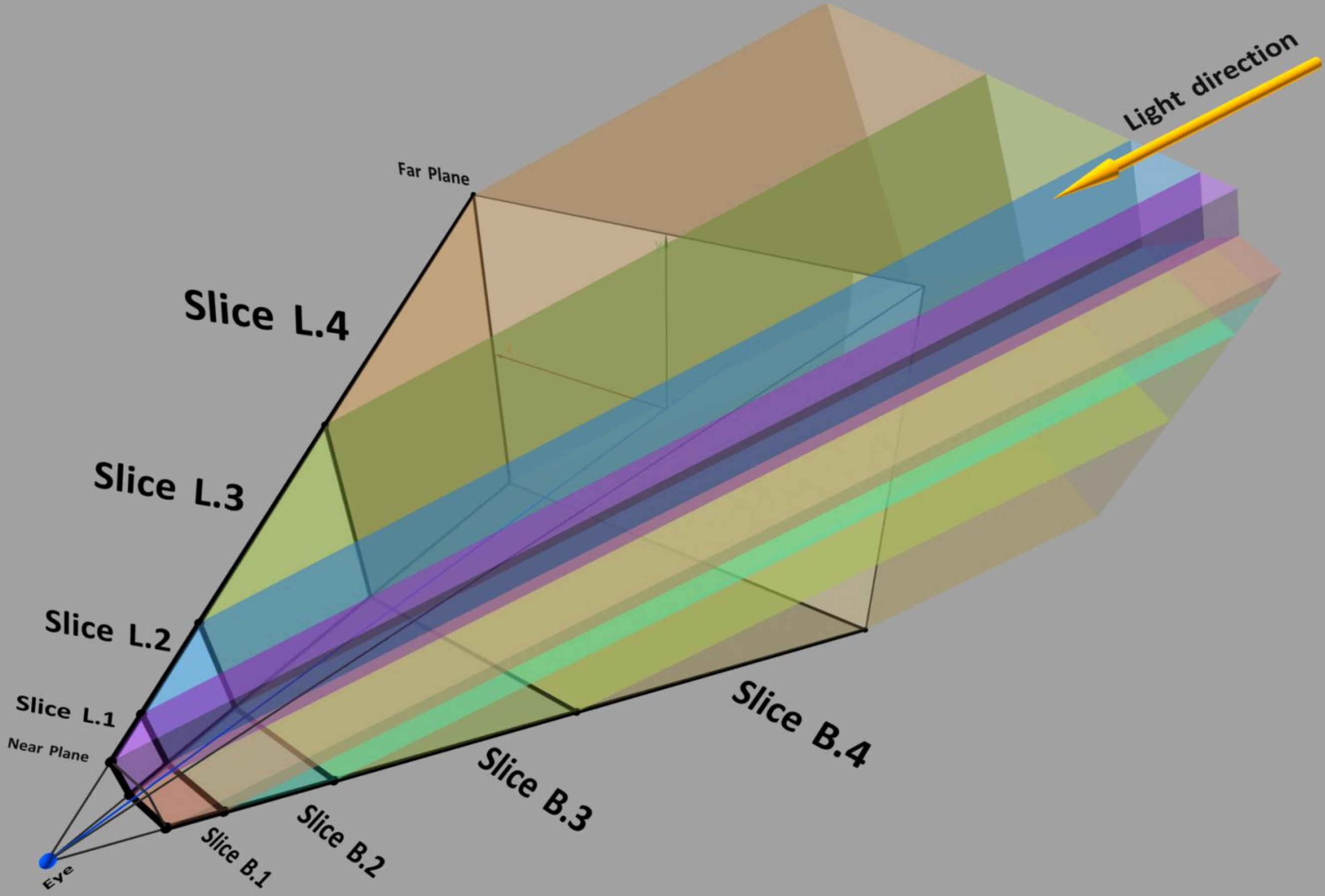
View Frustum Slices (Bottom Plane)



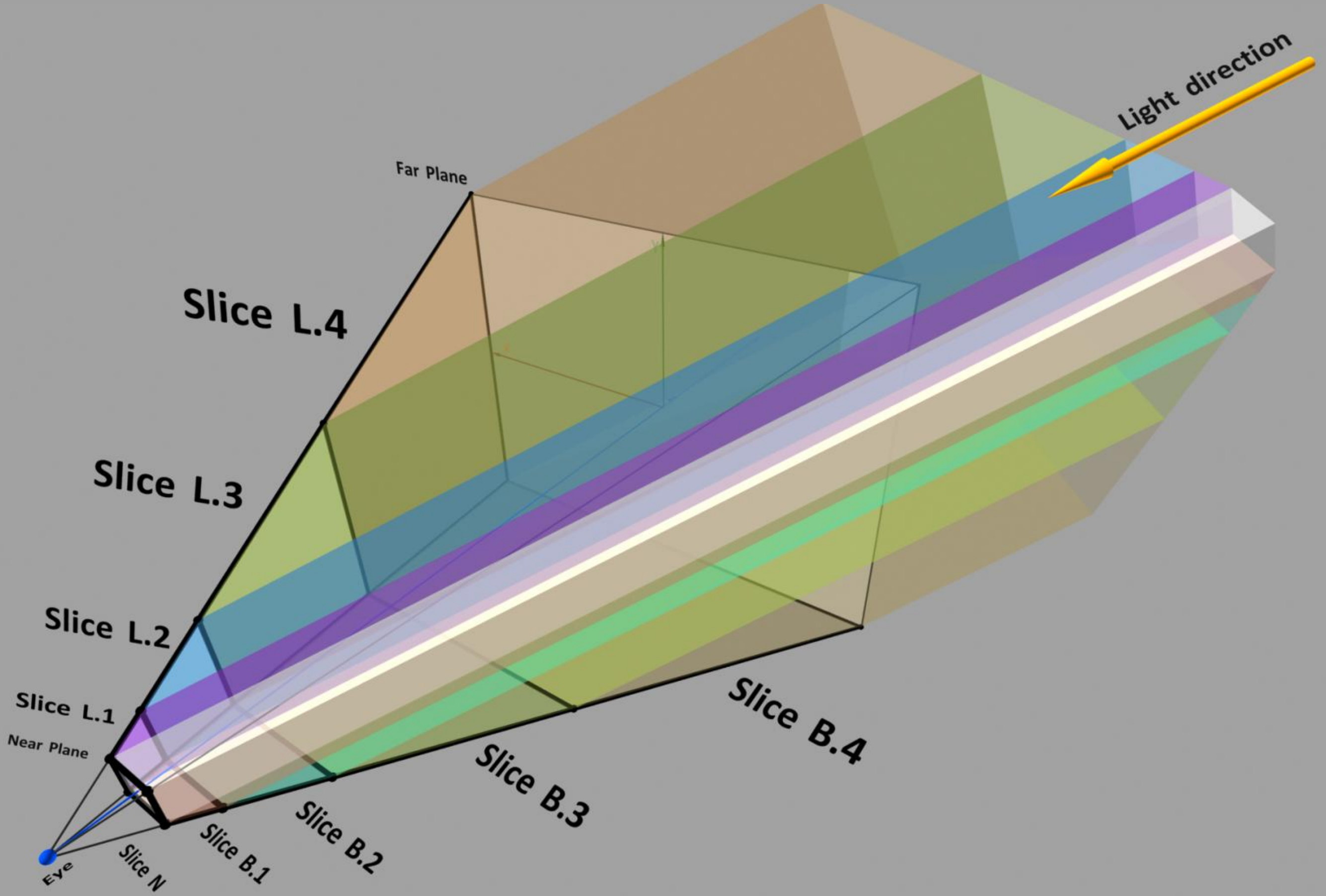
View Frustum Slices (Bottom Plane)



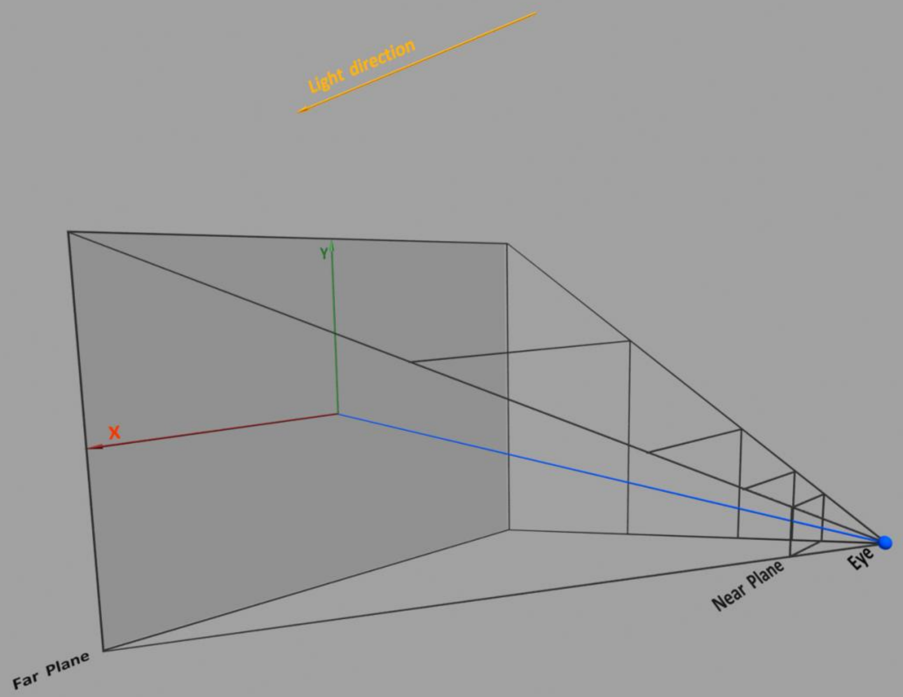
View Frustum Slices (Bottom Plane)



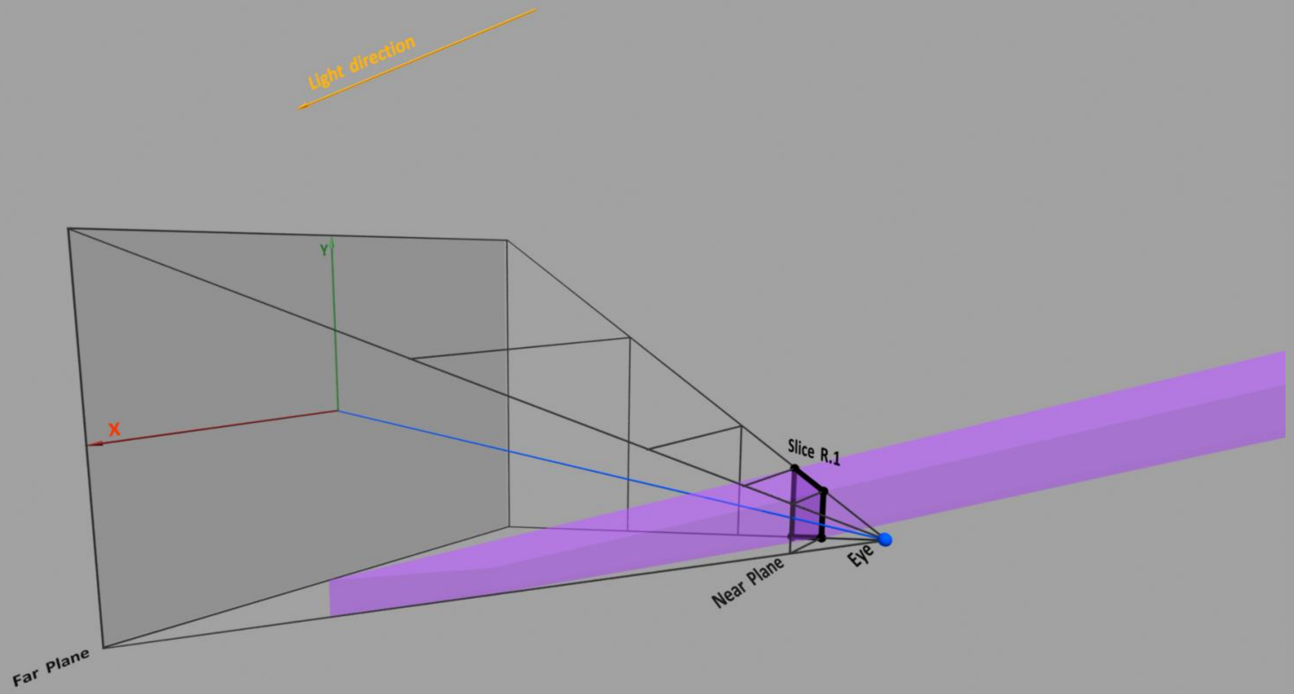
View Frustum Slices (Near Plane)



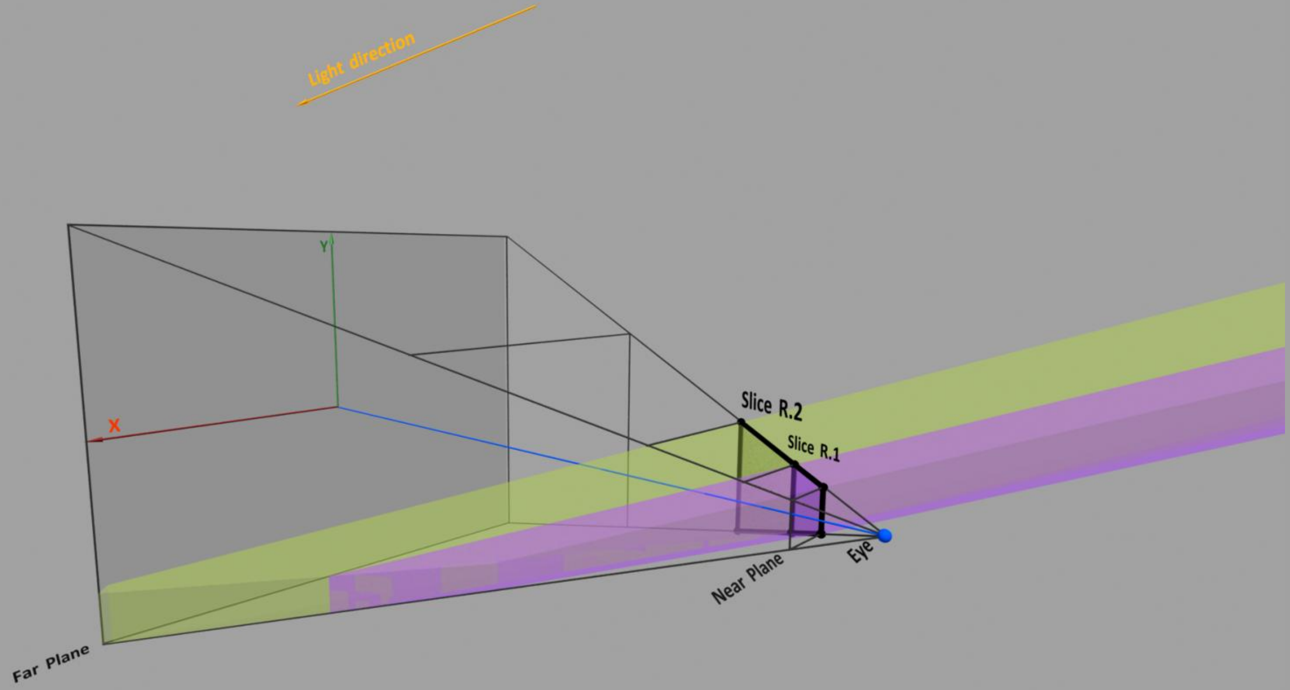
Plane Segments Selection



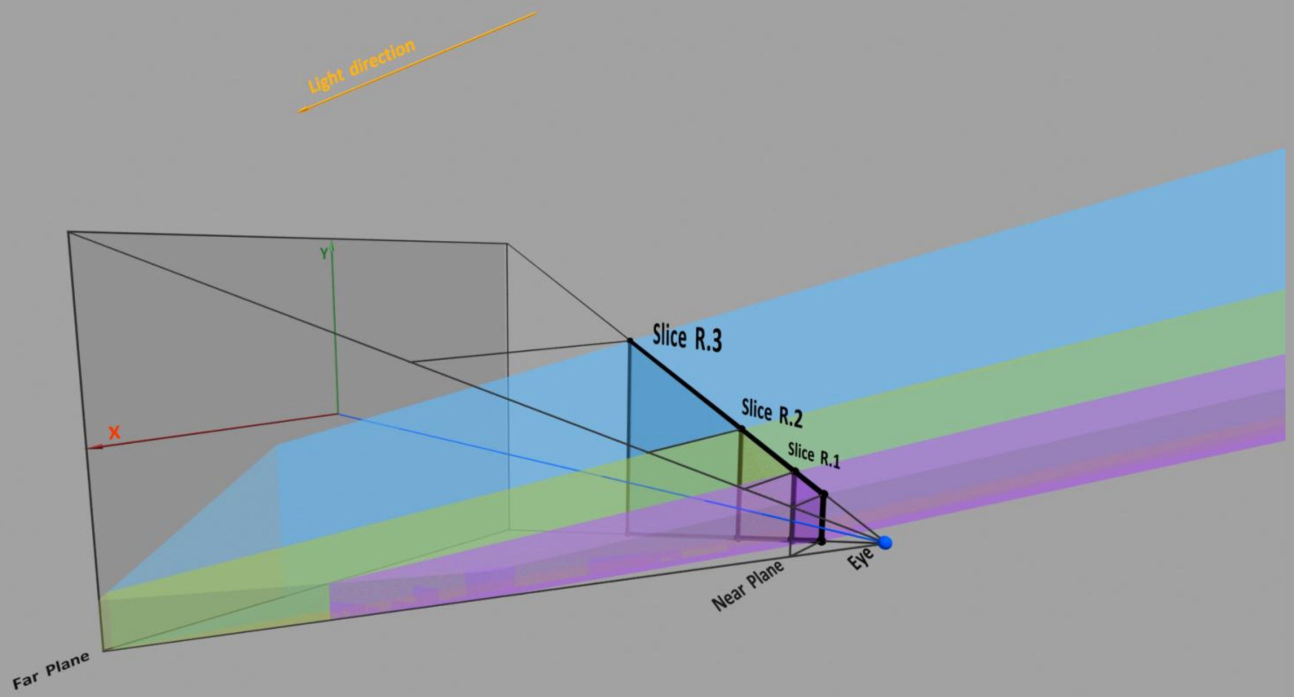
View Frustum Slices (Right Plane)



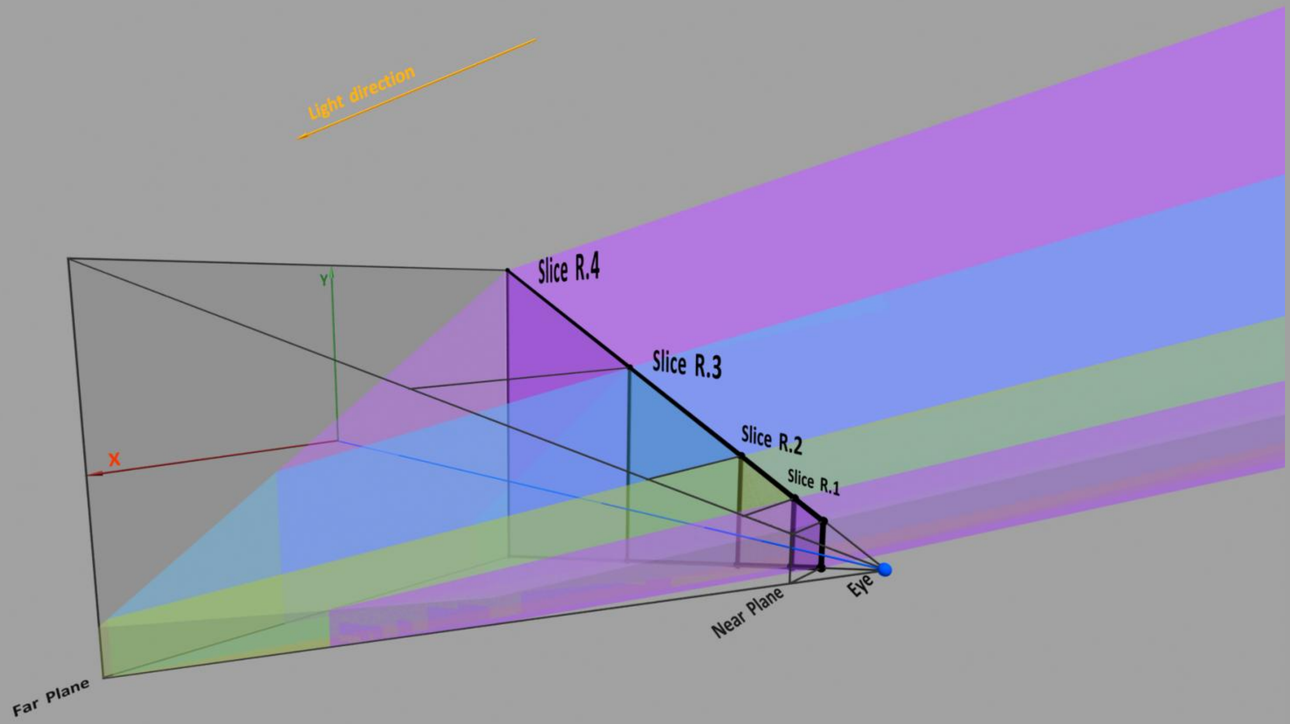
View Frustum Slices (Right Plane)



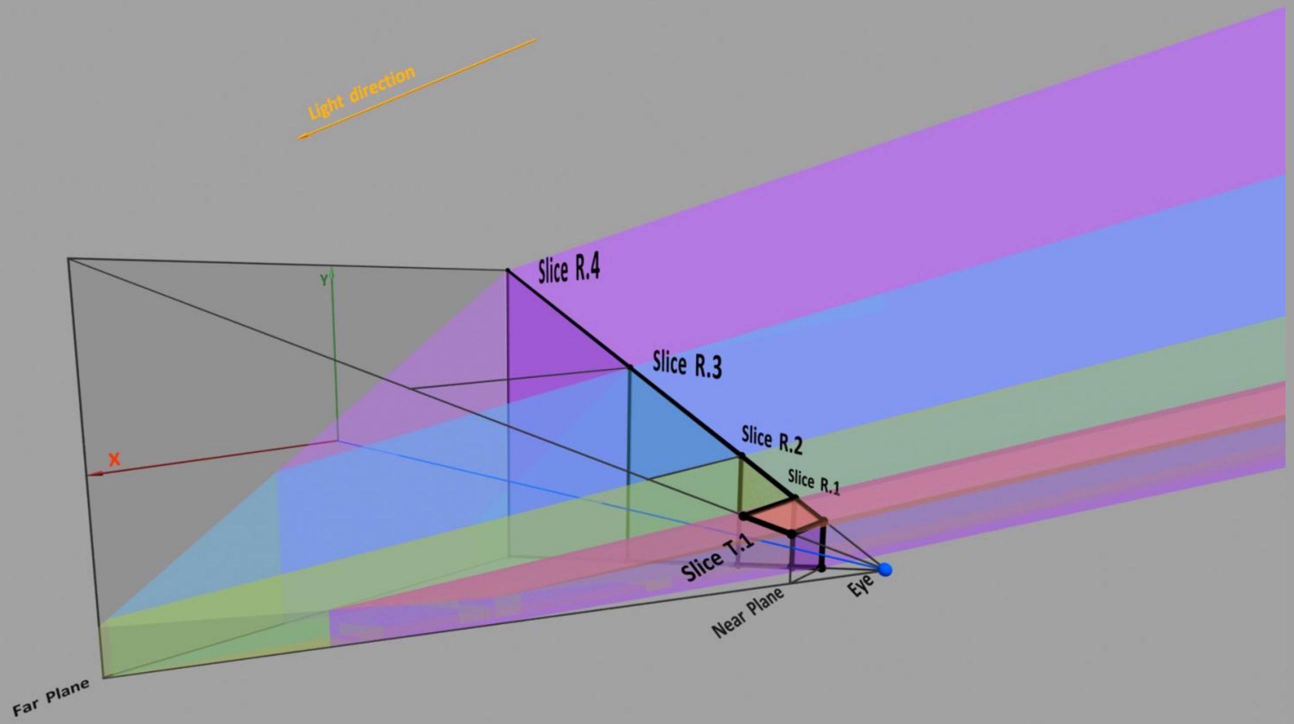
View Frustum Slices (Right Plane)



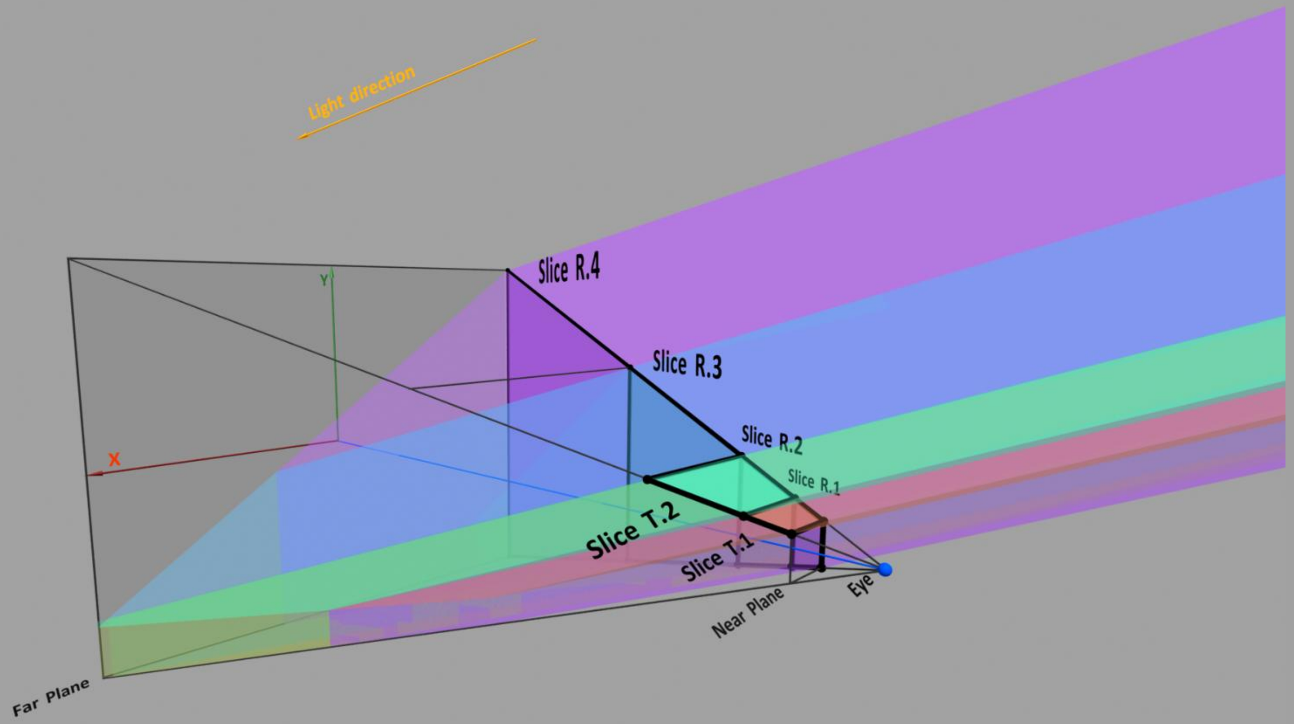
View Frustum Slices (Right Plane)



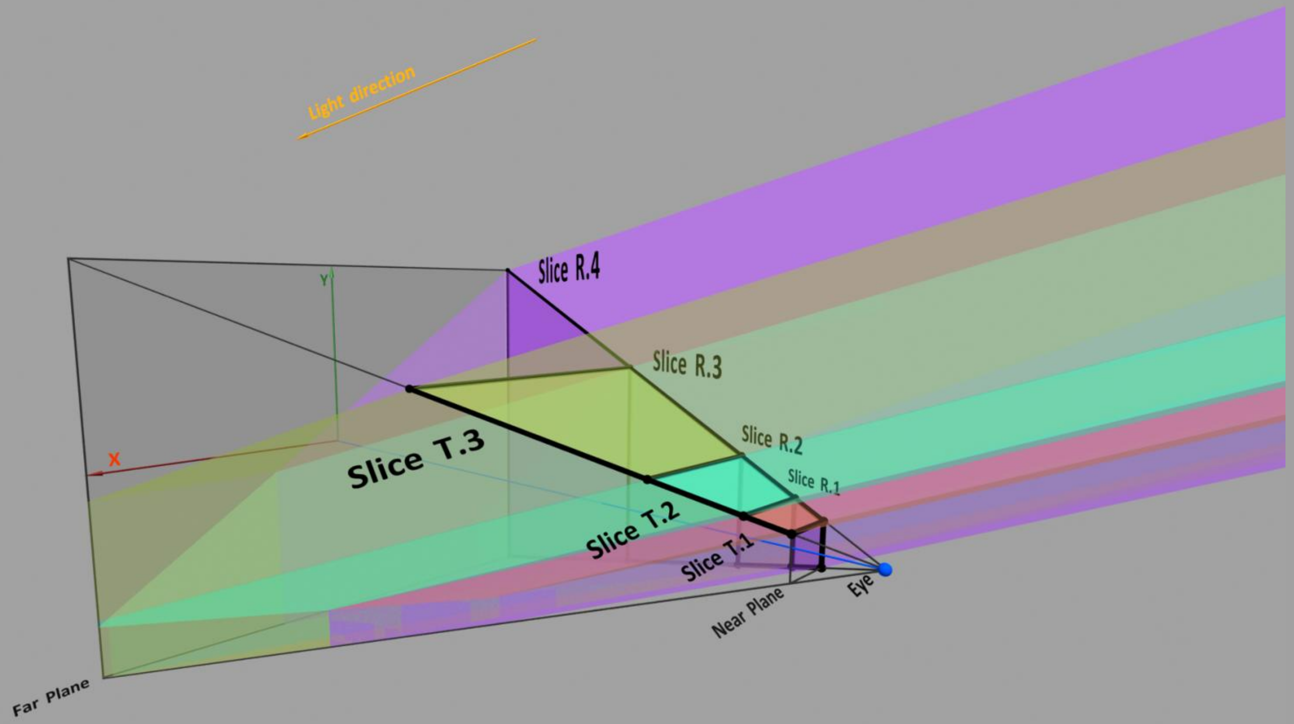
View Frustum Slices (Top Plane)



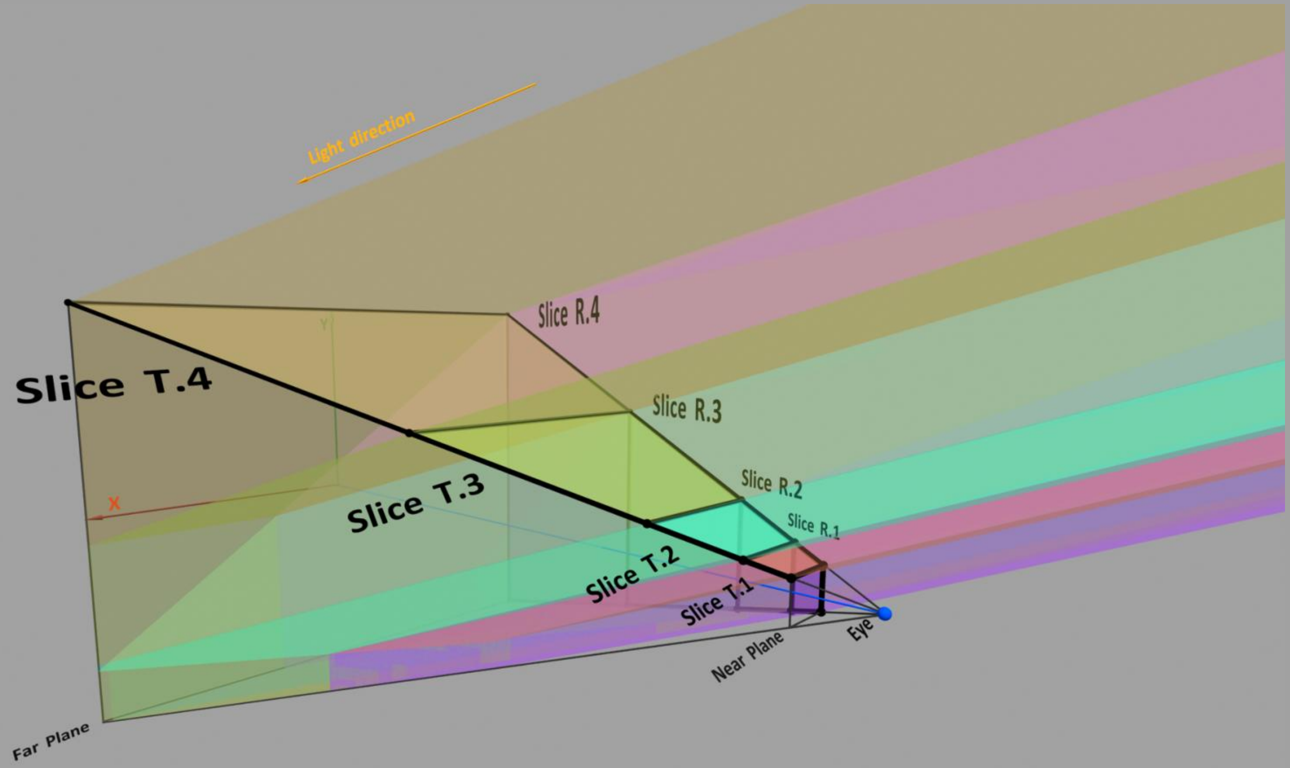
View Frustum Slices (Top Plane)



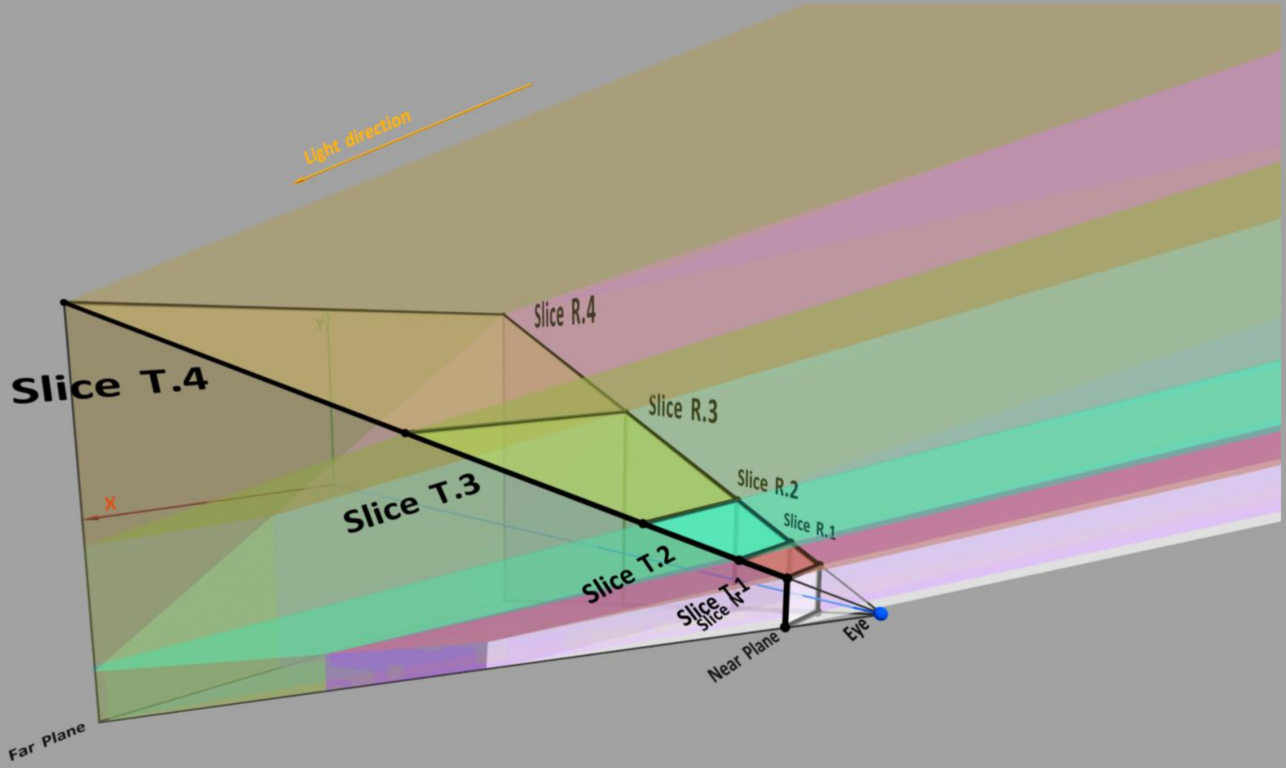
View Frustum Slices (Top Plane)



View Frustum Slices (Top Plane)



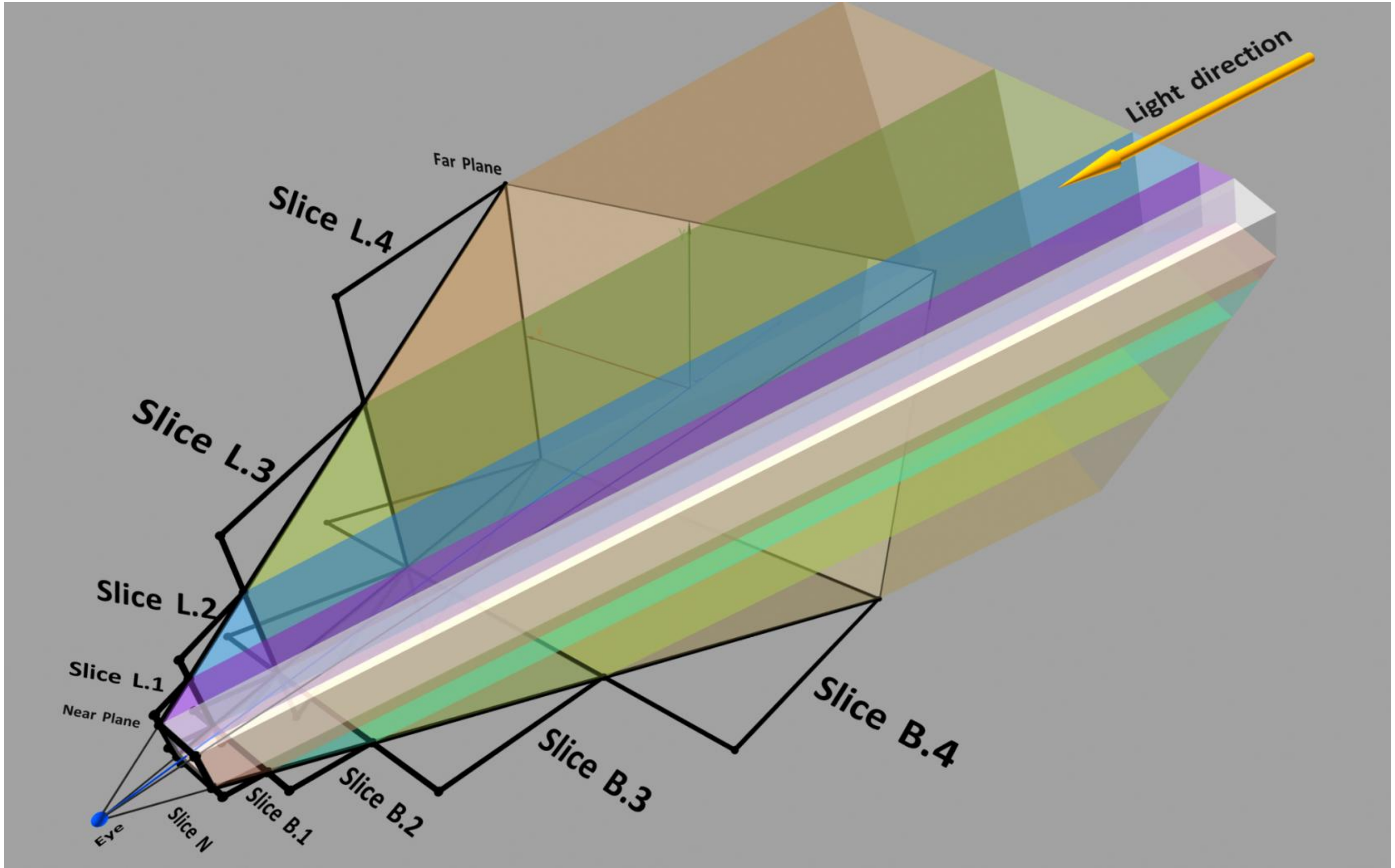
View Frustum Slices (Near Plane)



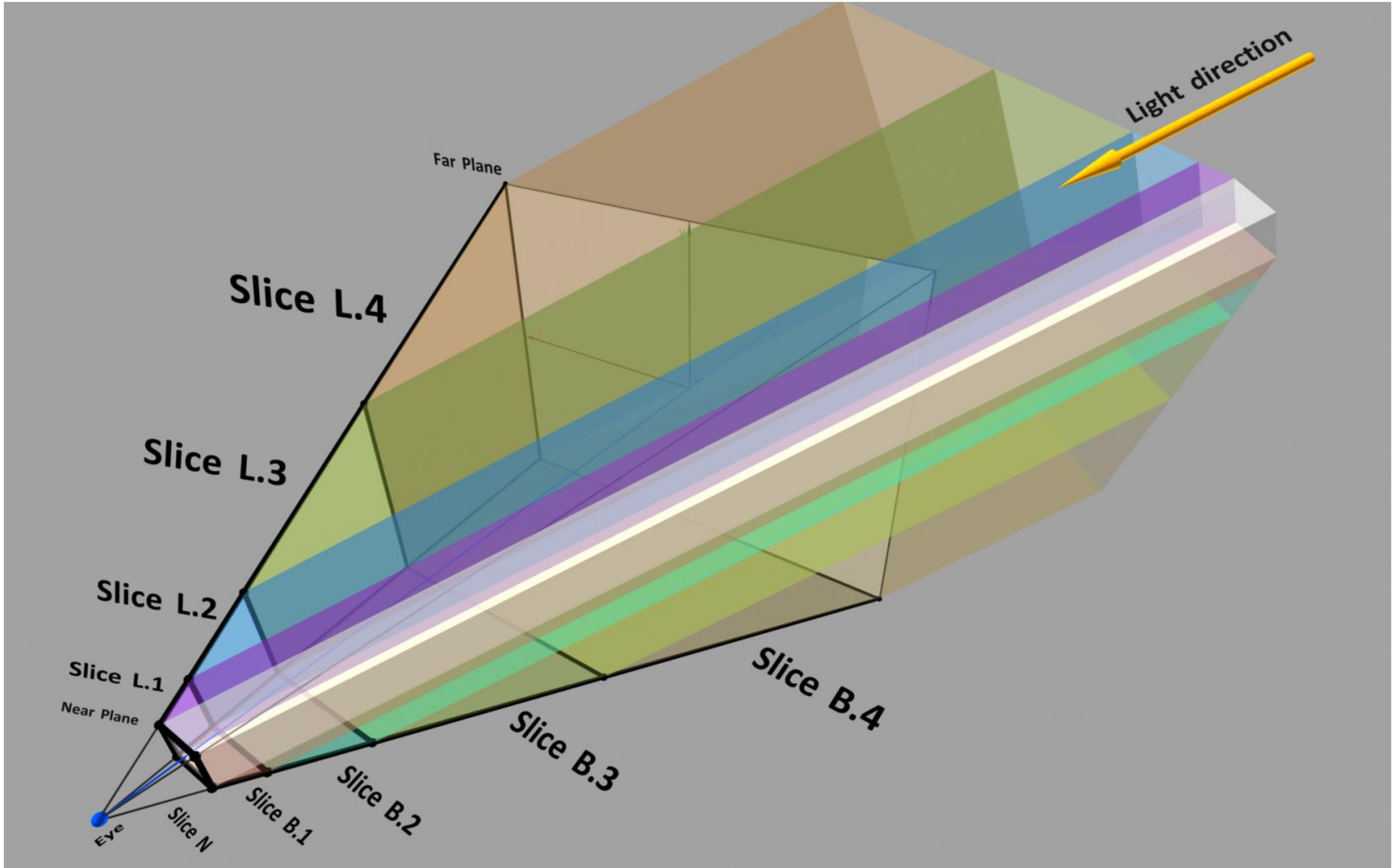
Shadow Map Parameterization

- Extend frustum segments to quads and use a rectangular shadow map
  - Requires more plane segments to get an acceptable approximation of a logarithmic splitting
  - wasted shadow space
- Use view camera perspective warp together with oblique projection
  - Virtual view camera with shifted and expanded near plane
  - Almost no wasted shadow space
  - Can be used successfully when we have enough shadow map sampling density to overcome shimmering
- Logarithmic distribution for projection planes segments

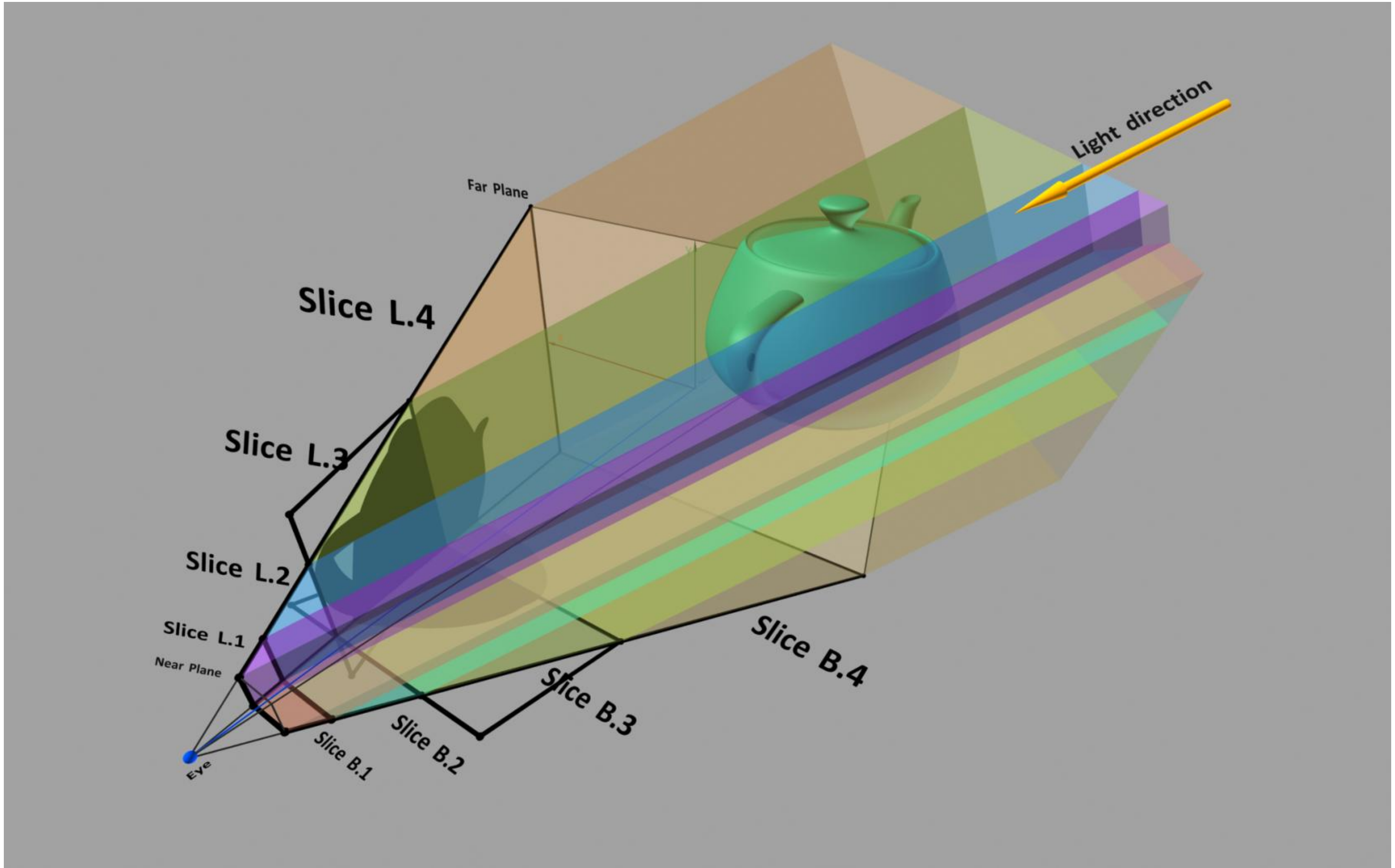
Shadow Map Parameterization



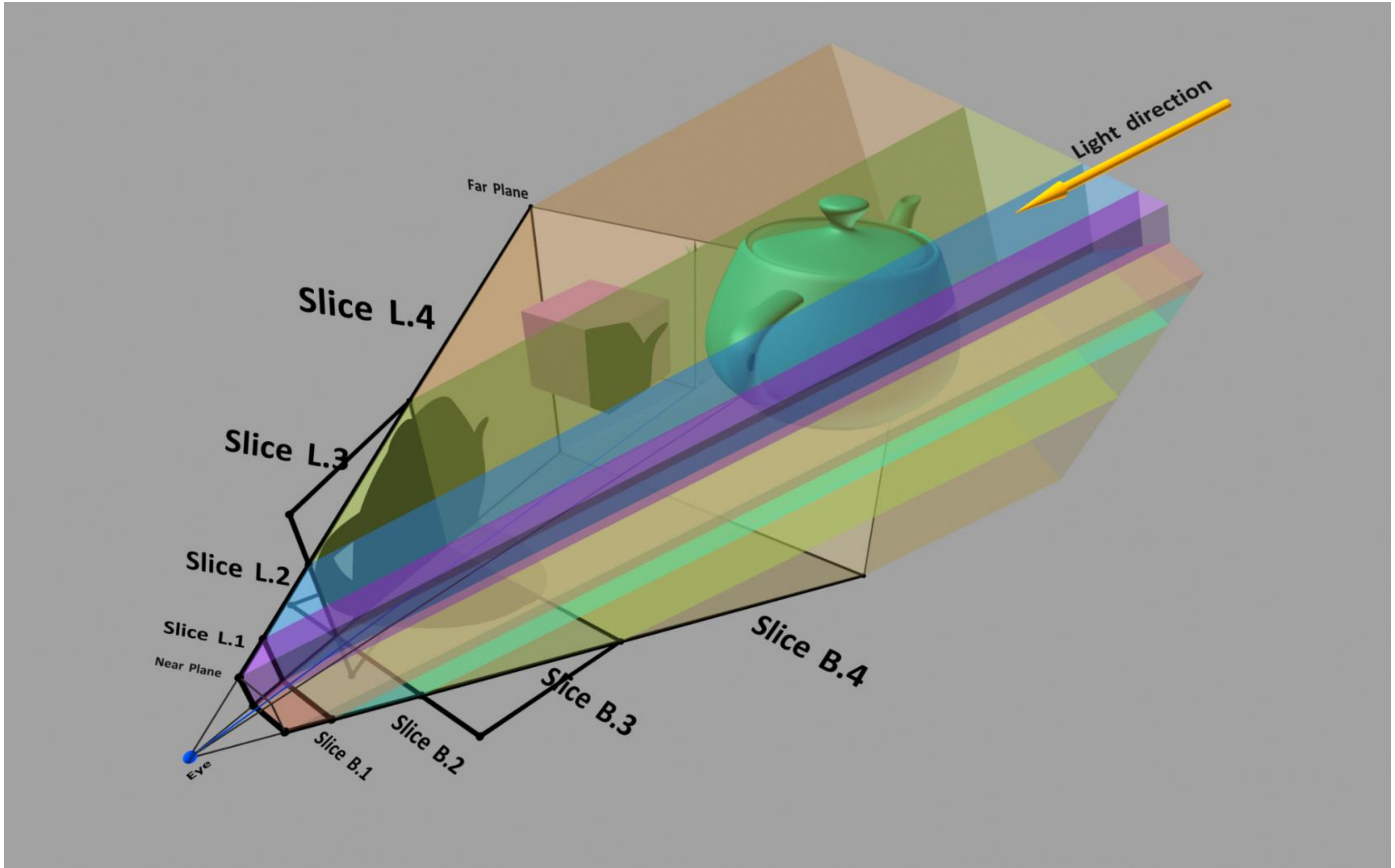
Shadow Map Parameterization



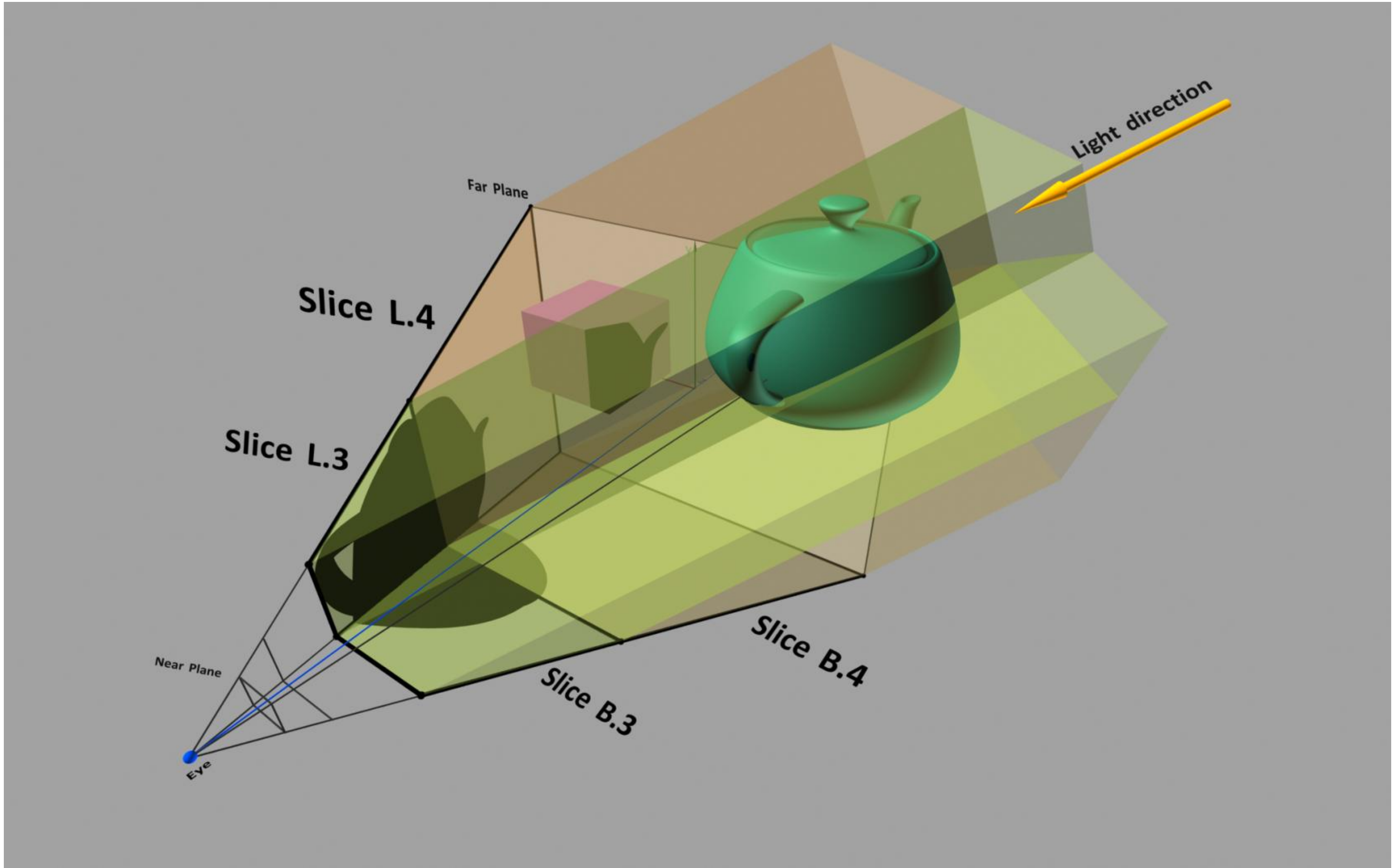
Oblique Shadow Projection



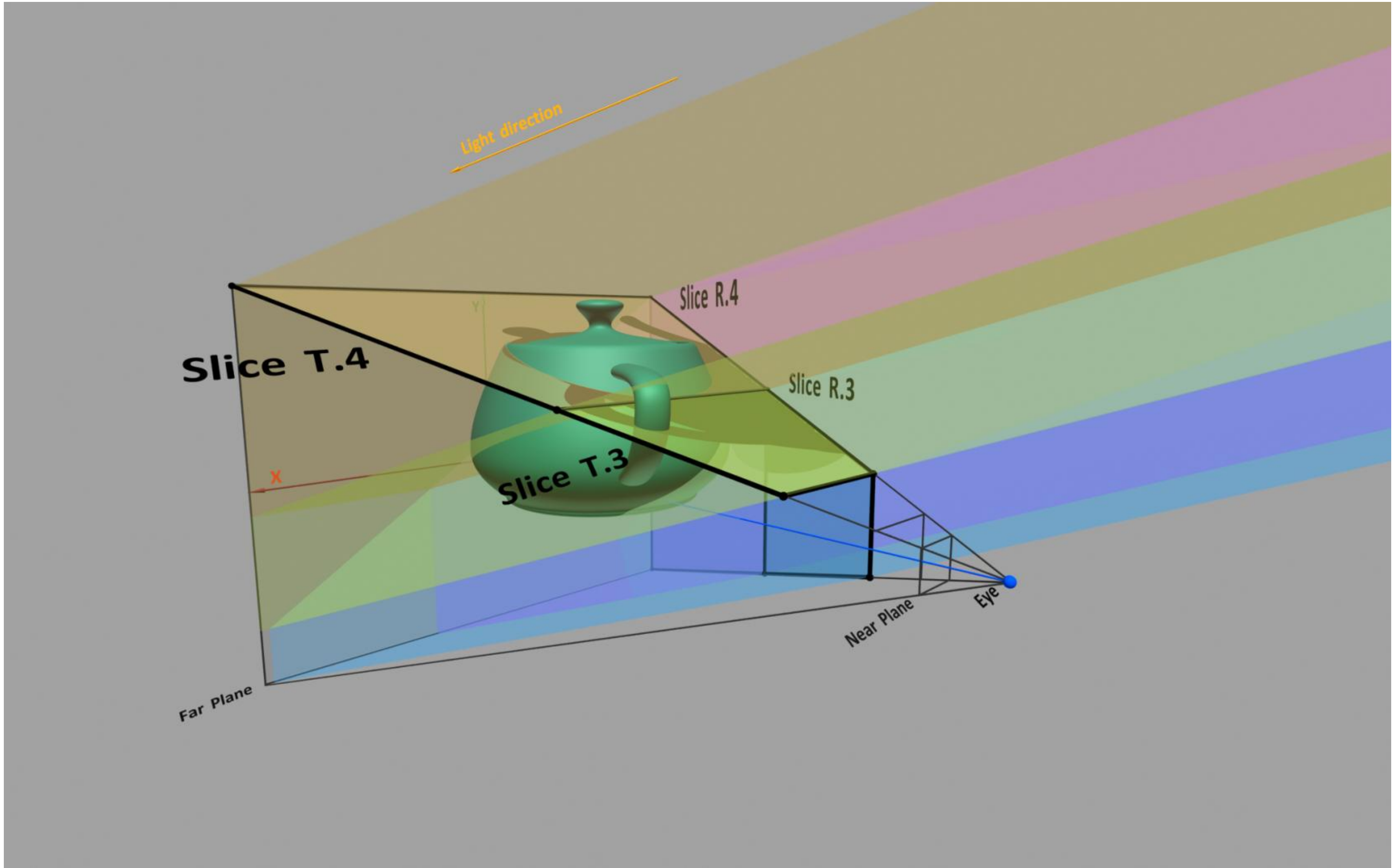
Oblique Shadow Casting



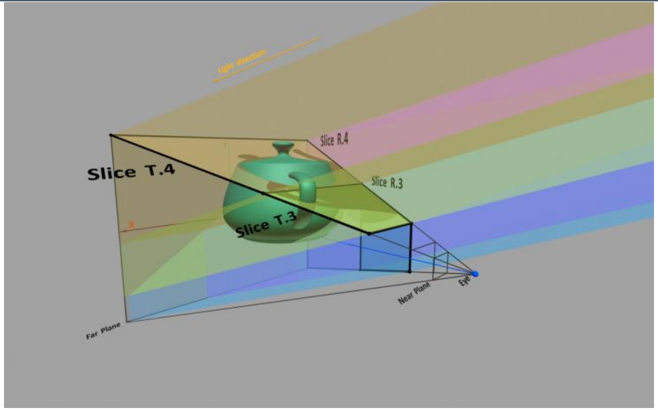
Affected Projection Planes Segments and Frustum Slices



Oblique Shadow Projection



Oblique Shadow Projection



- Shadow maps accurately cover the view frustum
- Only small parts of shadow maps are wasted on invisible areas of the scene
- No cascades overlapping
- Potentially shadow-receiving areas get guaranteed shadow sampling density since casters are projected on the most appropriate plane segments
- The approach is designed to maintain close to constant shadow map sampling density independent of light direction
  - Helps to address shadow aliasing problem

Implementation Details

- Shadow map rendering
  - Every split of view frustum plane is processed independently
  - Geometry shader replicates triangles between several segments on a plane when necessary
  - Proper plane’s segment is selected based on the Z coordinate in the view space

Use Cases

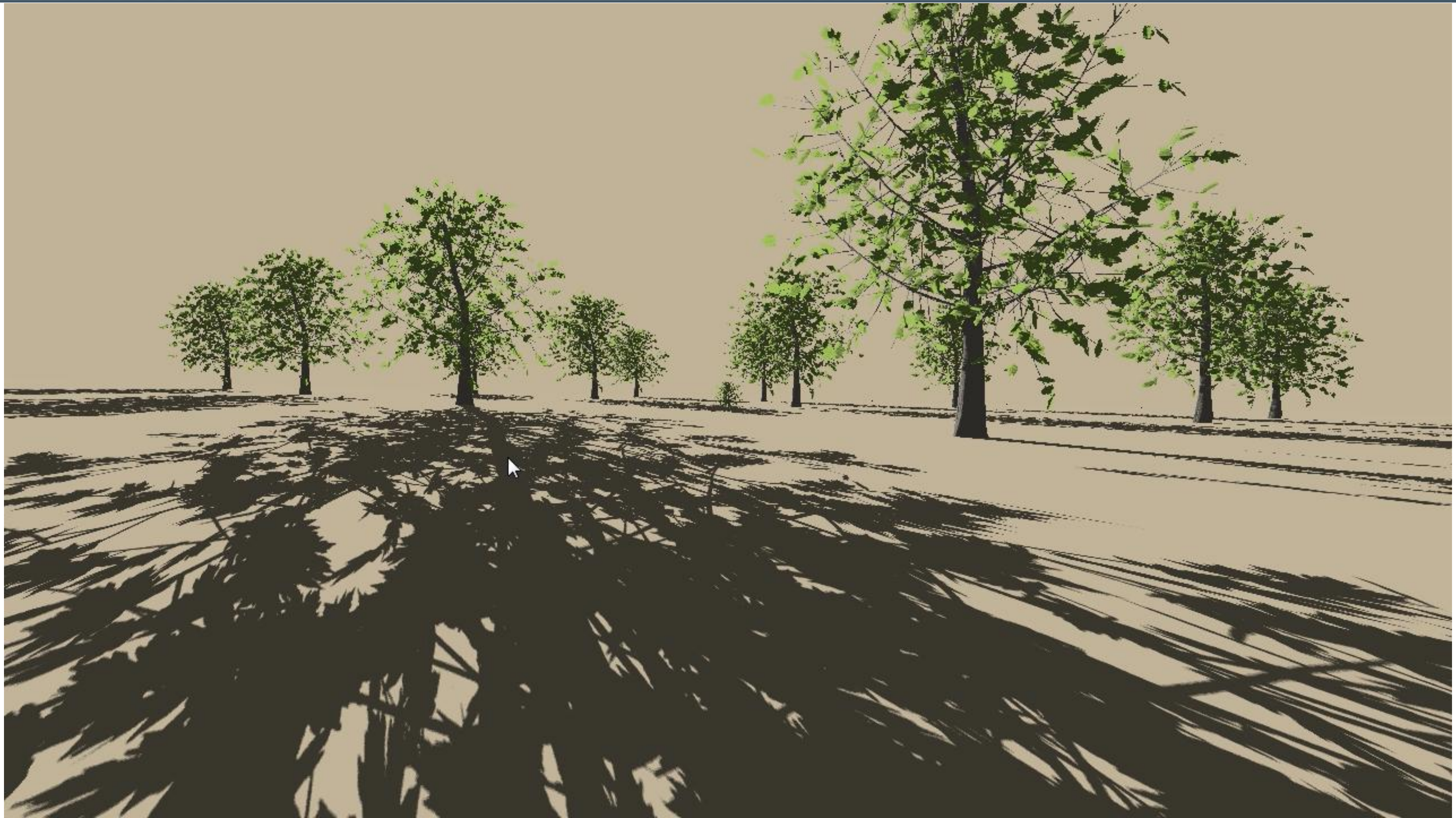
- Deferred shadow rendering
  - Consequently apply shadows from all oblique shadow maps
  - No complex stenciling between cascaded is needed as oblique frustums do not overlap
  - Texture arrays to index global shadow map segments/cascades
- Forward shadow rendering
  - straightforward shadow maps indexing with a one-to-one correspondence of the shadowed regions in to the shadow map regions
- Clustered Forward and Deferred Shading
  - No overlapping for shadow frustums
  - straightforward shadow maps indexing with a one-to-one correspondence of the clusters to the shadow map regions with oblique projection

Oblique Shadow Projection Features

- Efficient texture space usage
- Better addressed shadow aliasing problem
- Allows to approach guaranteed shadowmap sampling density
- Aliasing-free if hi-resolution shadow maps are used

  (1-2K, optimal 4K)

Oblique Shadow Projection: Demo



Summary

- Deferred Shadows
- Cascaded shadow maps
- Soft Shadows Approximation
- Voxel based area light shadows
- Shadows & Transparency
- Contact Shadows/SSDO
- Screen Space Self-Shadowing
- Volumetric shadows
- Oblique projection for cascaded shadow maps

Special Thanks

- Tiago Sousa, Carsten Wenzel, Anton Knyazyev, Michael Kopietz, Theodor Mader, Vladimir Kajalin, Dmitry Gait, Nicolas Schulz, Serhat Eser Erdem, Elmar Eisemann

- And to the entire Crytek Team !

Questions

Nick@crytek.com / NickKasyan@googlemail.com