Rasterization Rendering Effects Review

Aaron Lefohn, Intel / University of Washington
Mike Houston, AMD / Stanford

Overview

- Surface shaders
- Light shaders
  - Shadow maps
- Reflections
  - Planar reflections
  - Environment maps
- [z,w,g,a,k,f,n]-buffer review
- Billboards
  - Hair, foliage, smoke, etc.

Approach for This Lecture

- Since many of you are more familiar with ray tracing than rasterization…
- This lecture describes how basic material, illumination, and visibility problems are solved in real-time rasterization-based renderers (compared to how they are solved in a simple ray tracer)

(Surface shaders)

Surface Properties

Surface Shaders

- In ray tracing
  - Your "materials" class defines the BRDF and provides the surface properties to the BRDF such as diffuse color, specularity, etc.
- In rasterization
  - A batch of primitives with the same "material" are rendered at the same time
  - "Material class" implemented in pixel shader

(Light shaders)

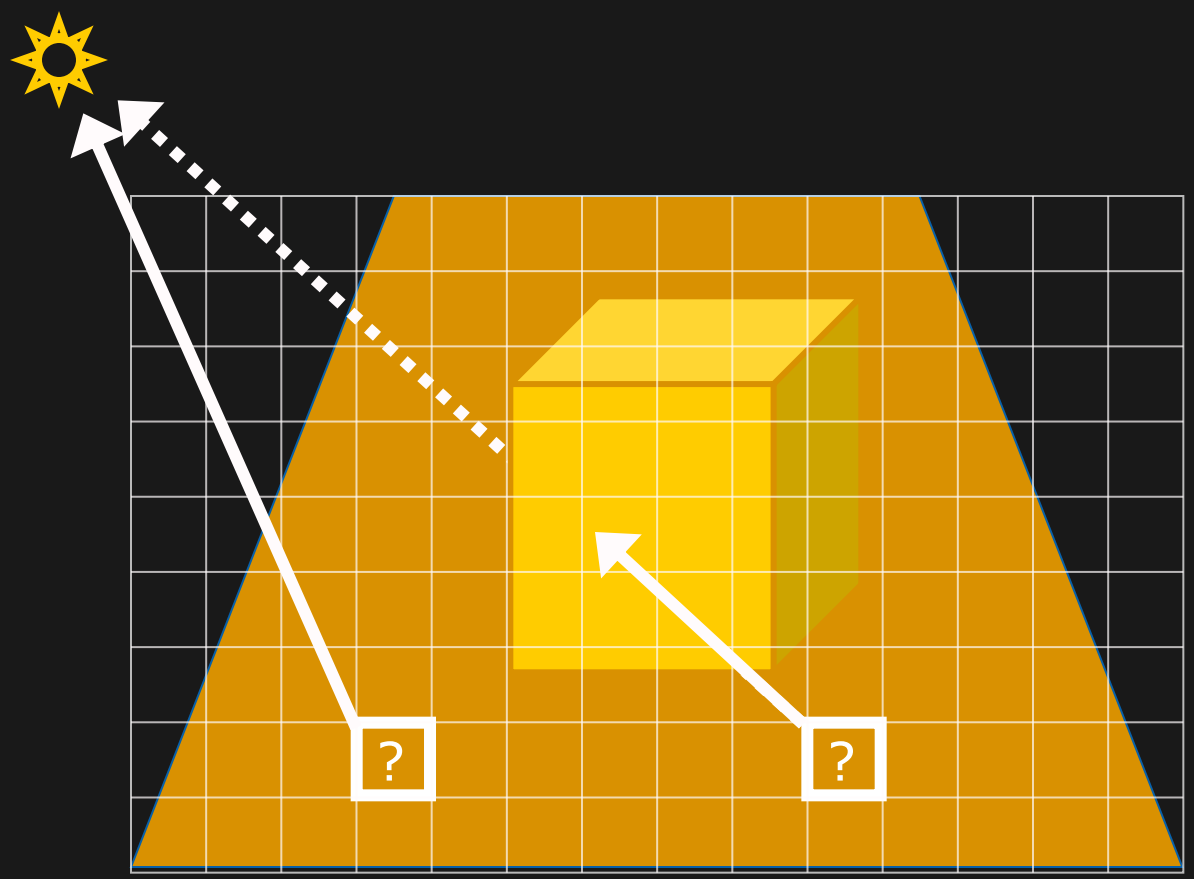
Direct Illumination

Direct Illumination

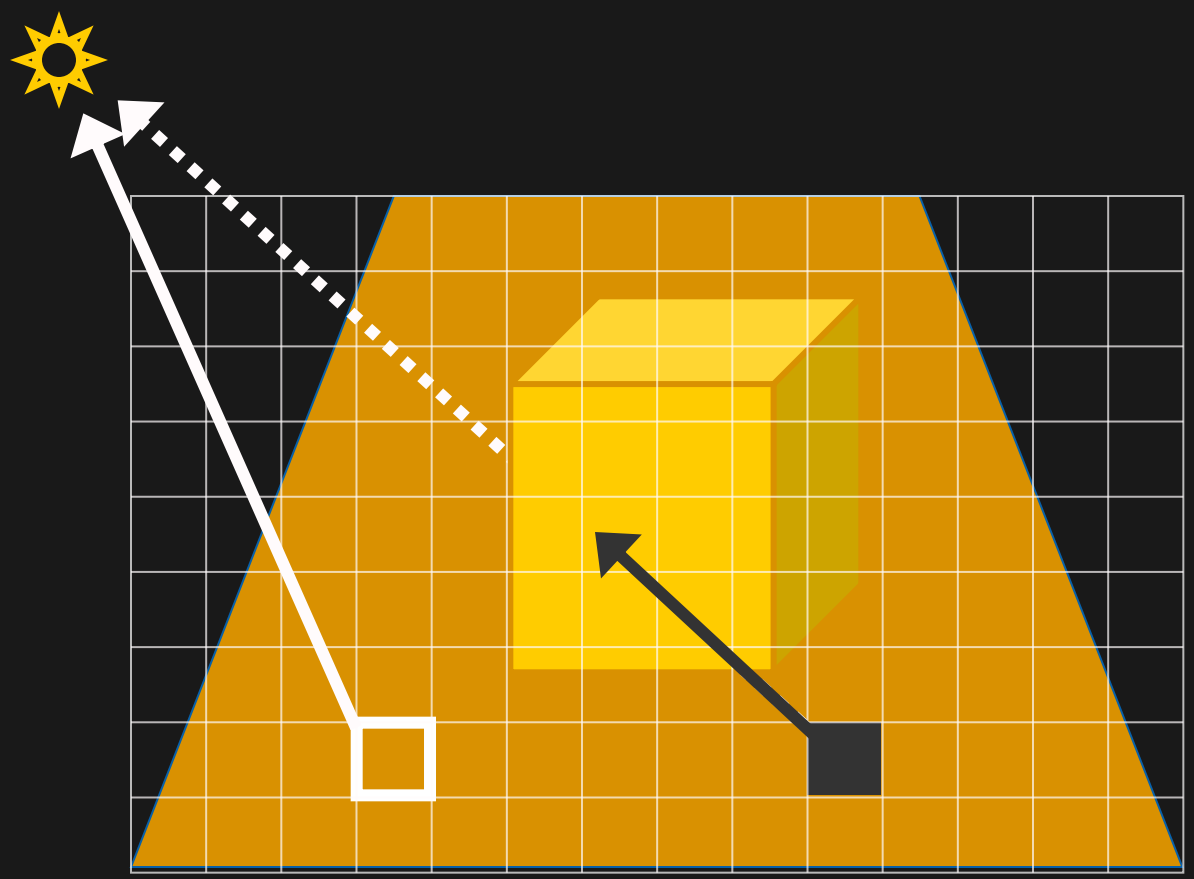
- In ray tracing
  - "At a ray-surface intersection, trace rays to all lights and combine with BRDF to compute final color"
- In rasterization
  - Query visibility data computed in pre-pass for each light (pixel shader) E.g., shadow mapping
  - Combine lighting result with BRDF to compute final color (pixel shader)

Shadow Mapping

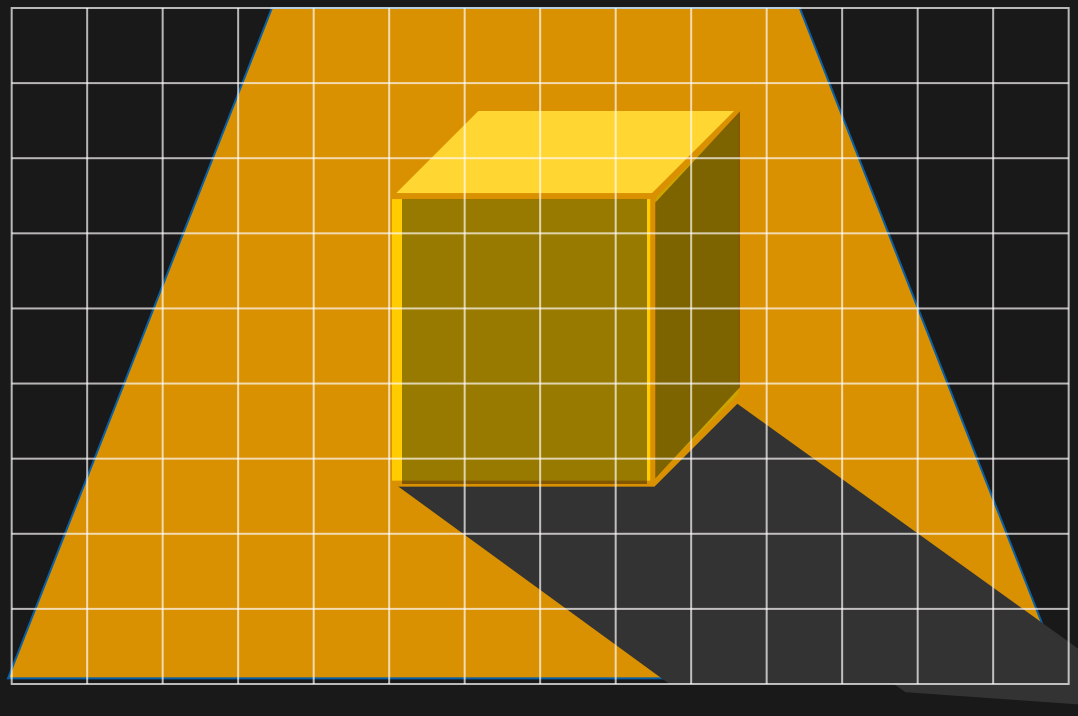
The Shadowing Problem



The Shadowing Problem

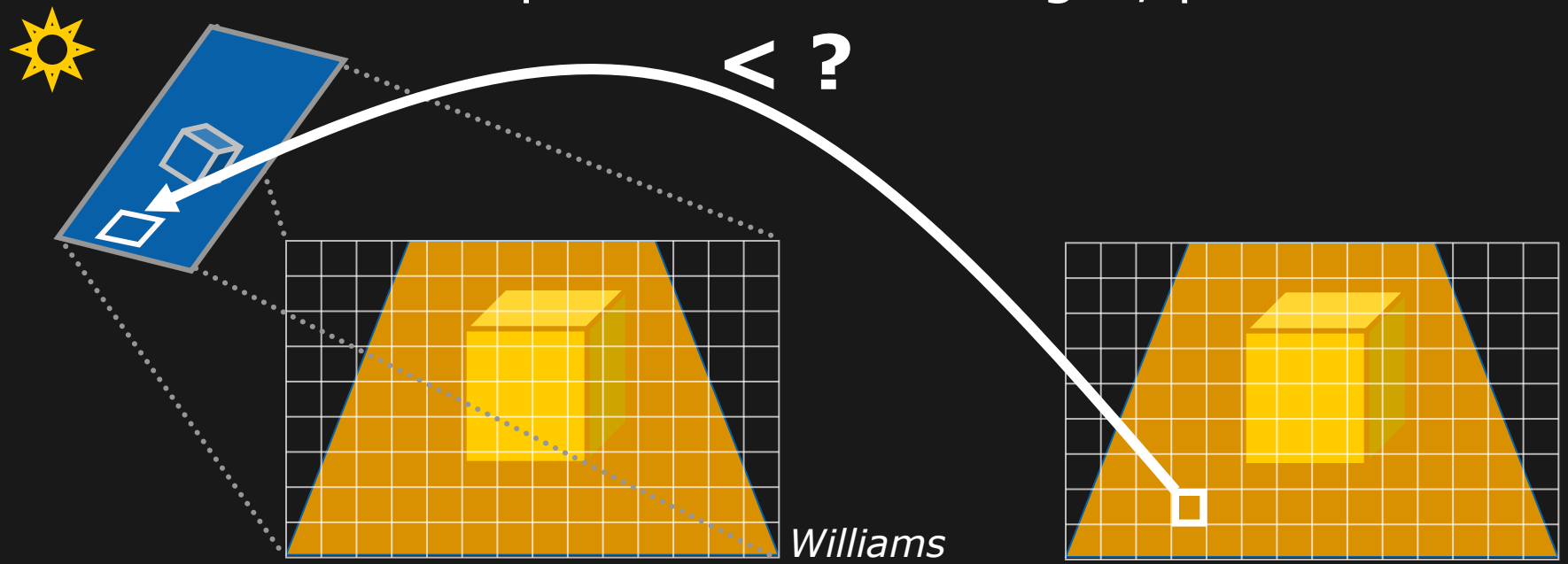


The Shadowing Problem



Shadow Mapping

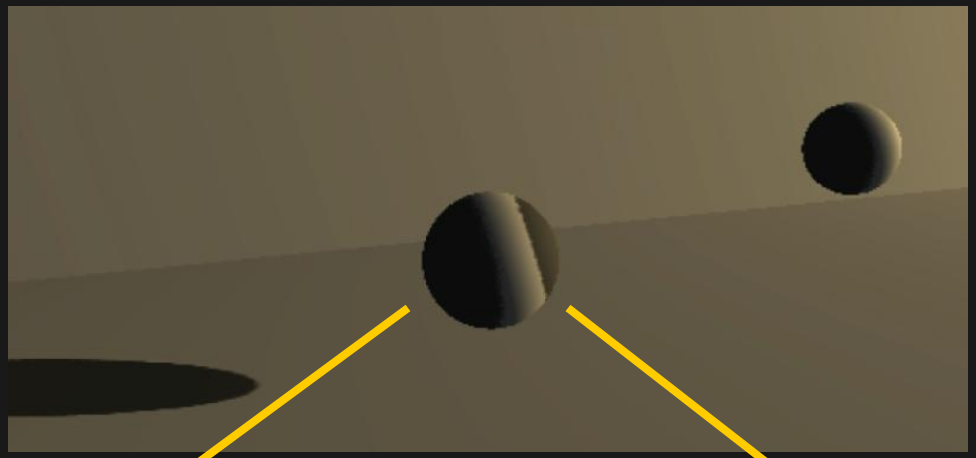
- Render depth image from light position
- Shadow lookup
  - Transform eye samples to shadow map
  - If shadow map value closer to light, pixel in shadow



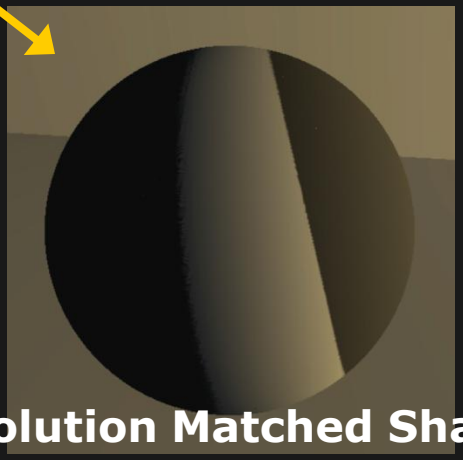
Williams SIGGRAPH 1978

Projective Aliasing

- Occluder normal nearly orthogonal to light rays



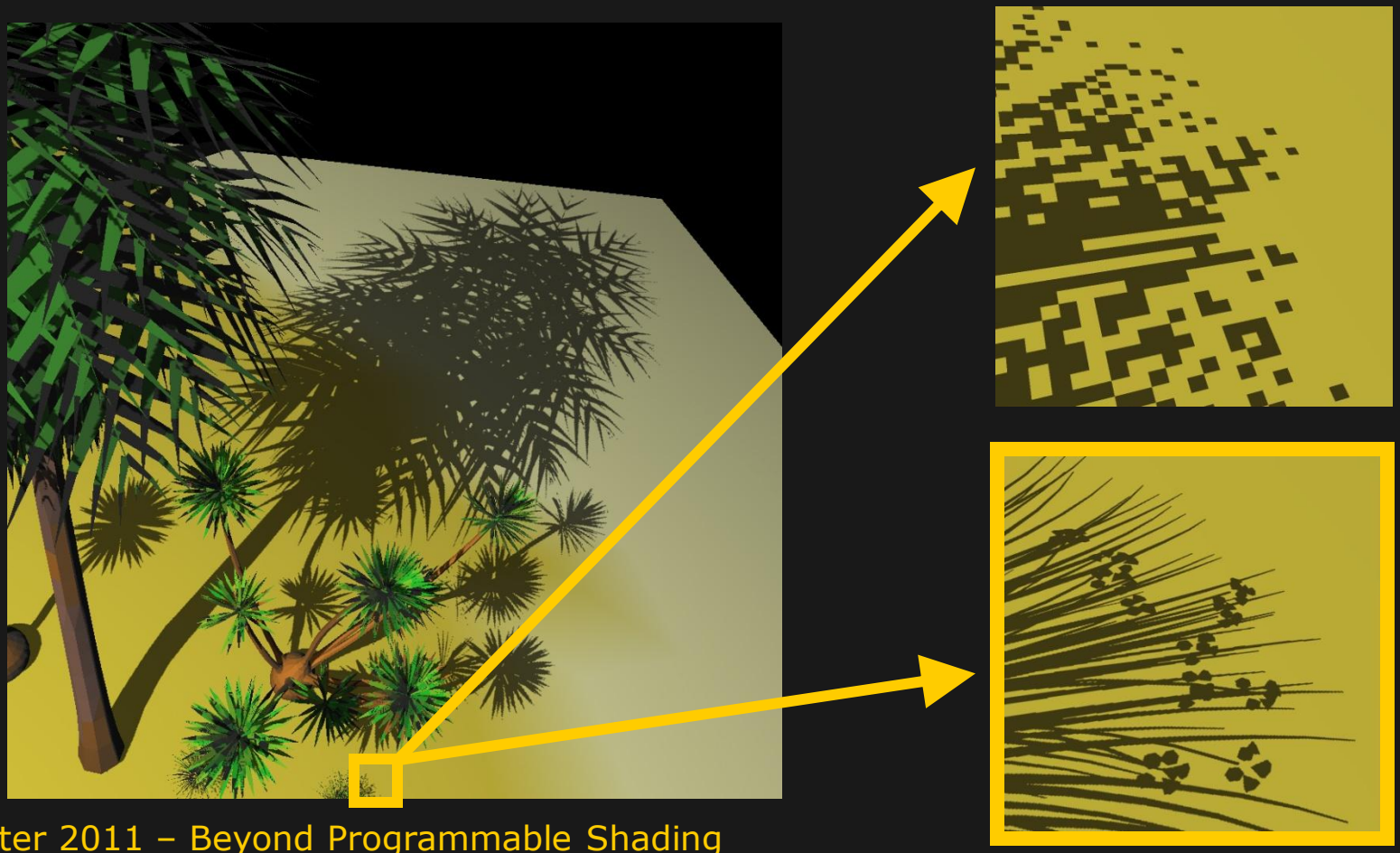
2048^2 Standard Shadow Map



$32{,}768^2$ Resolution Matched Shadow Ma

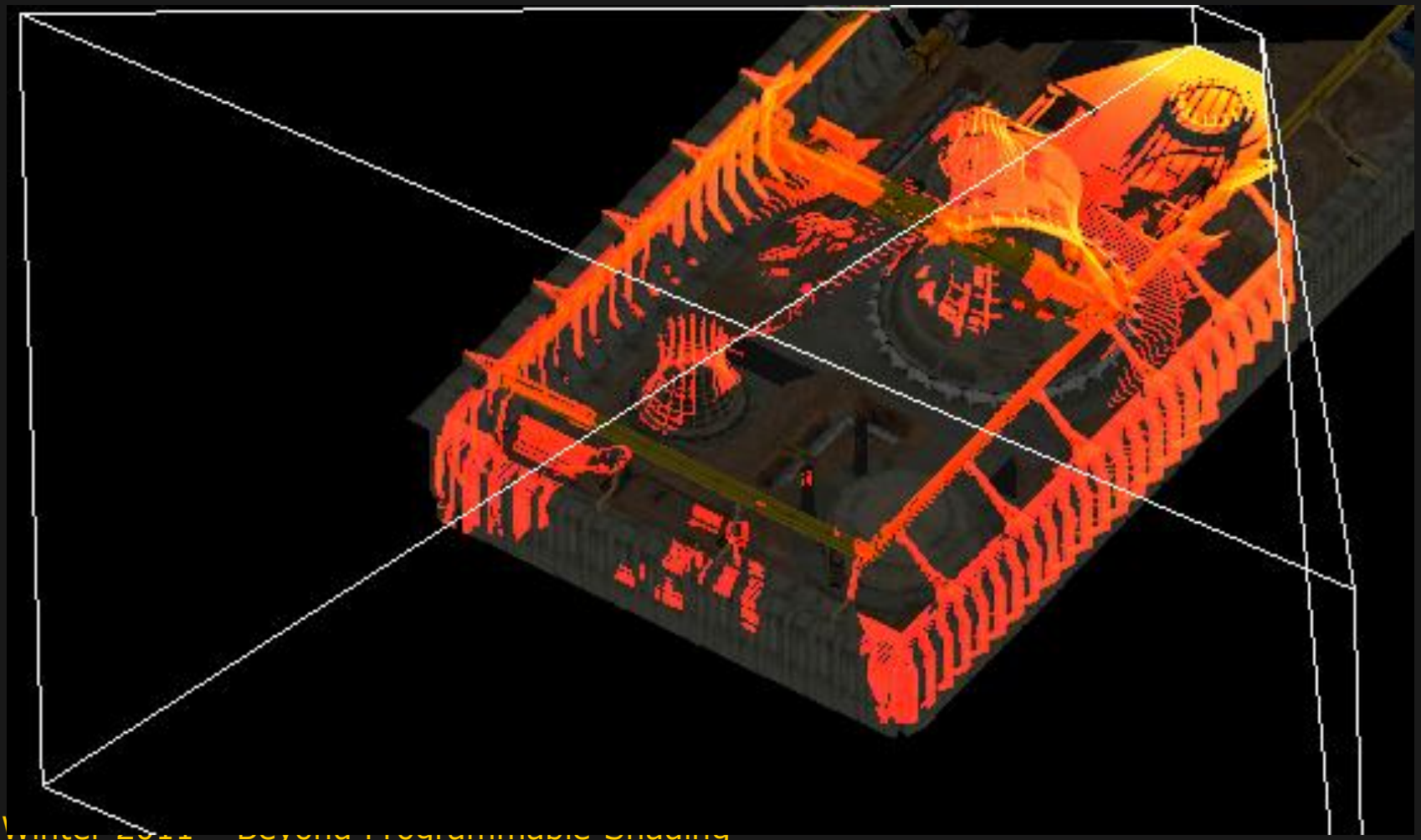
Perspective Aliasing

- Mismatch between sampling distribution of eye-space samples and shadow samples



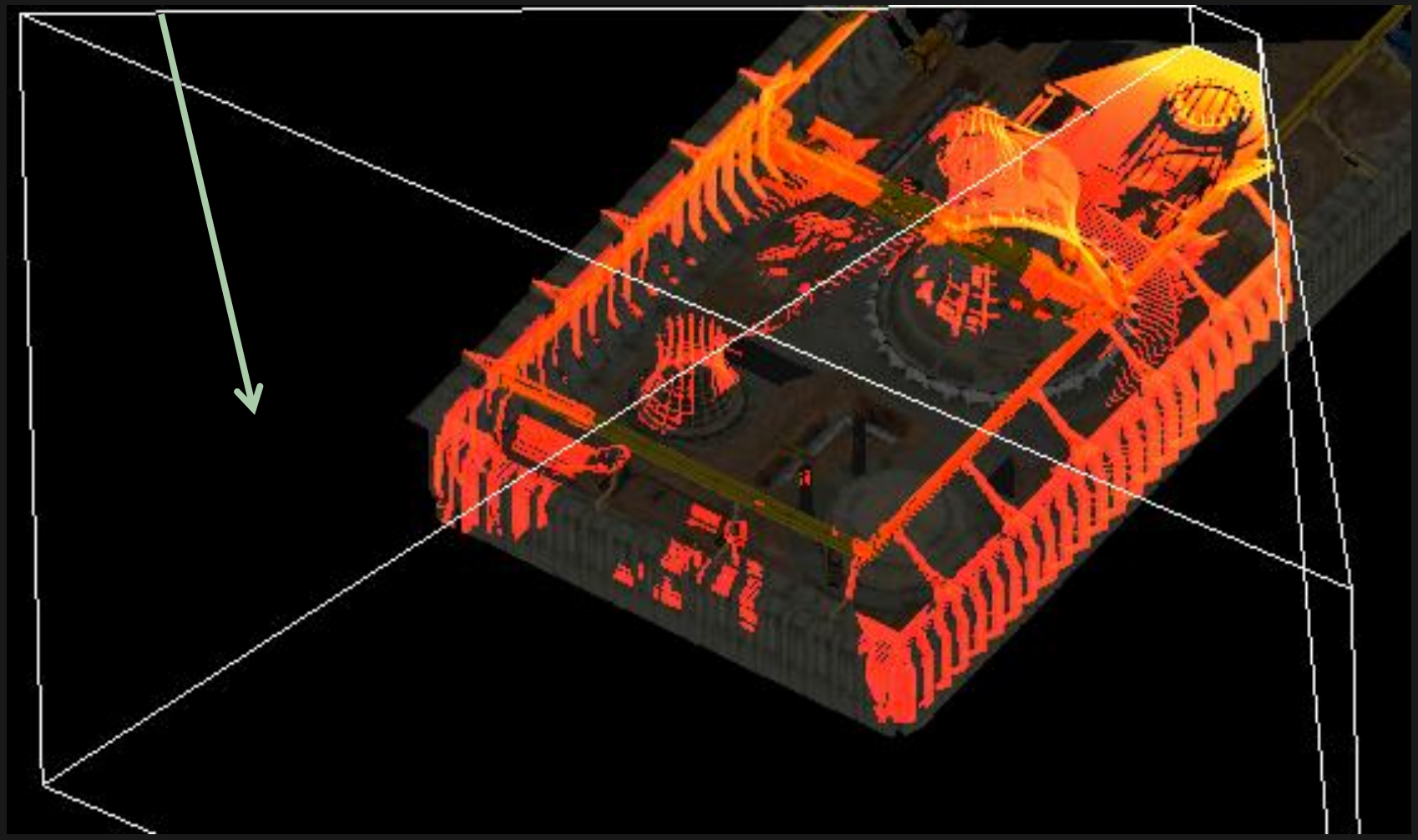
Light Space Sample Distribution (Shadow Rays)

- Samples colored; yellow = denser samples



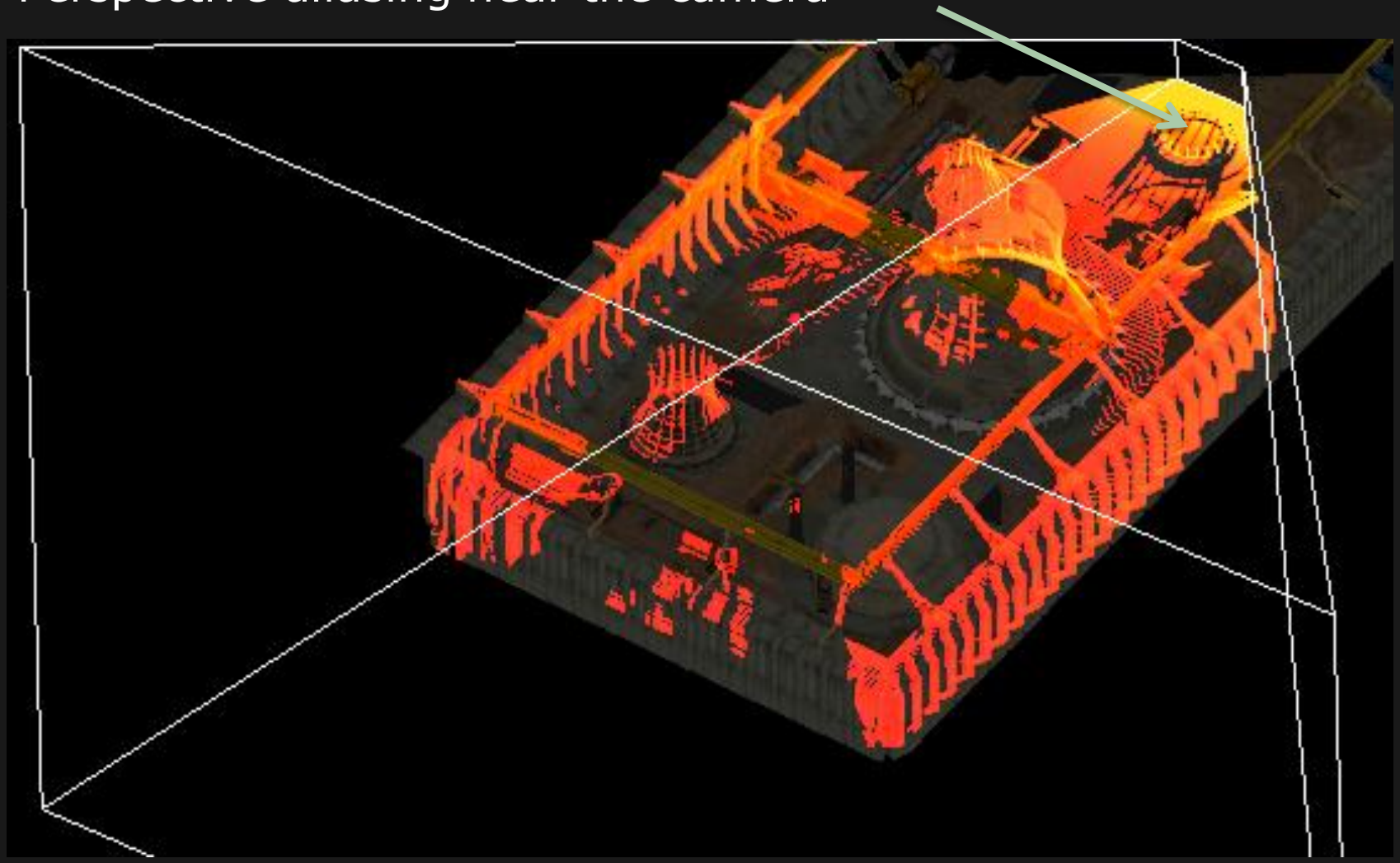
Naïve Shadow Mapping

- Wastes lots of space that is never sampled



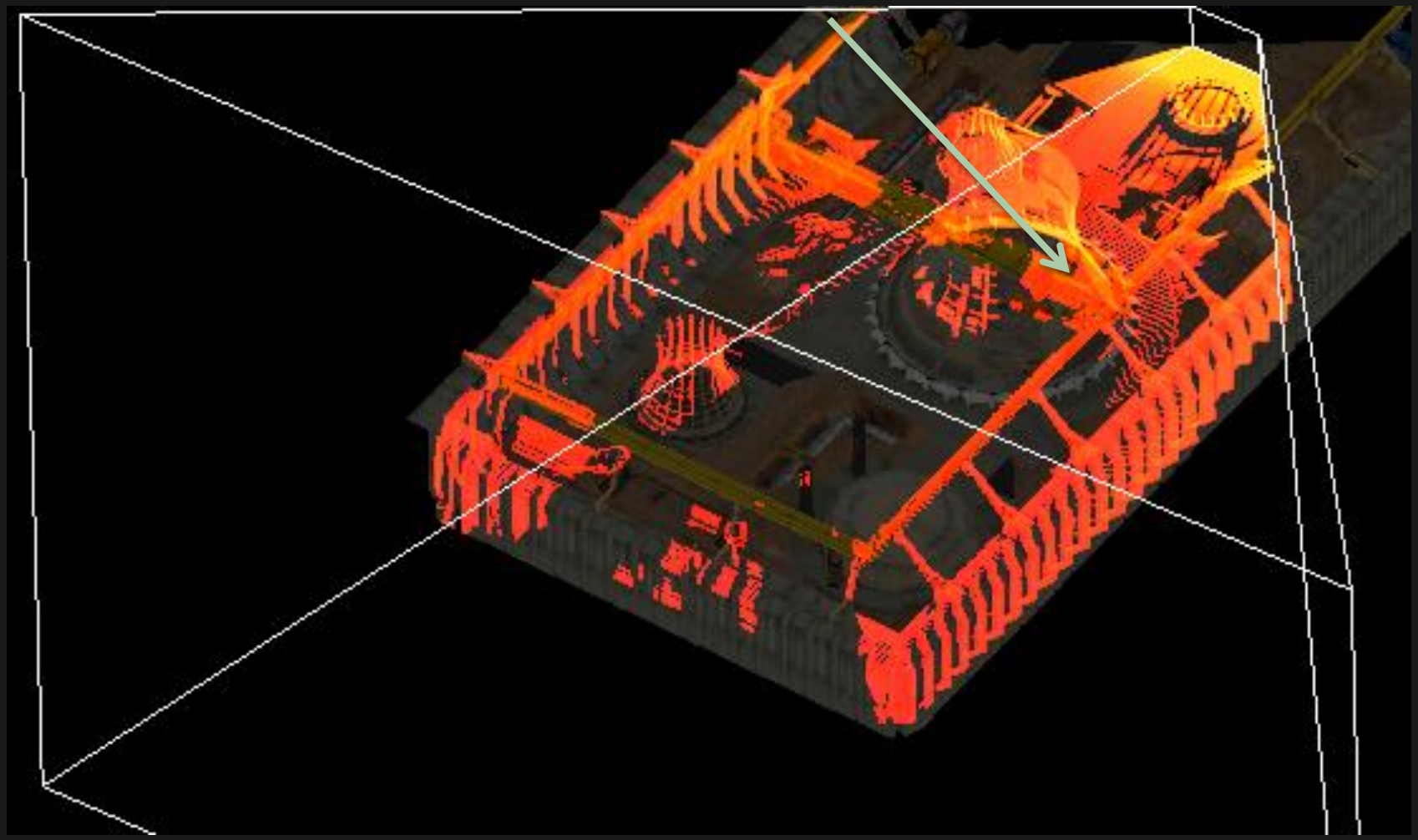
Naïve Shadow Mapping

- Perspective aliasing near the camera



Naïve Shadow Mapping

- Projective aliasing on surfaces aligned with light rays



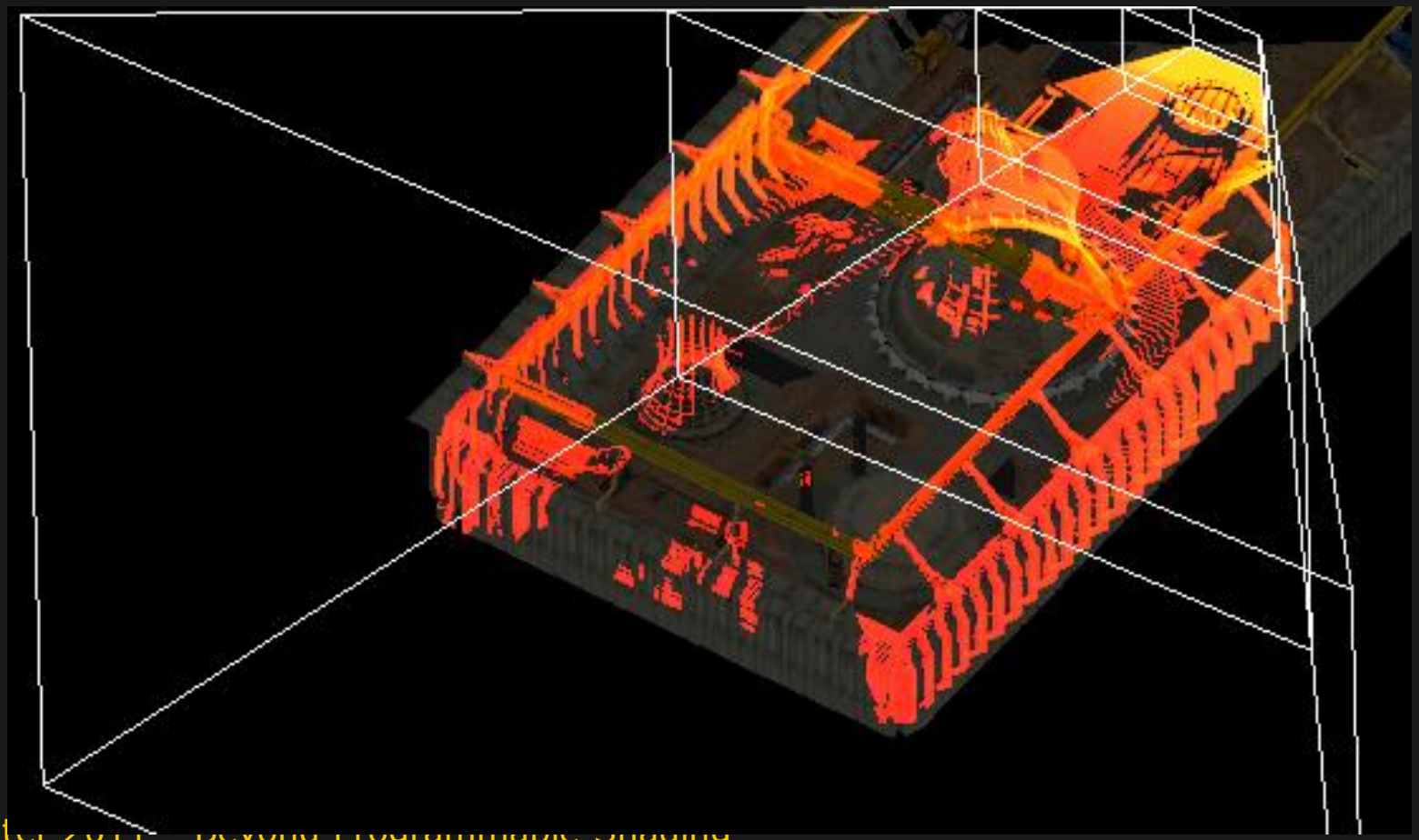
Shadow Map Techniques

- Hundreds of shadow map papers address perspective, projective, and depth representation aliasing. For example:
  - Perspective shadow maps (and many follow-up ideas)
    - Warp shadow map to match receiver samples)
  - Adaptive quadtree shadow maps
    - Generate hundreds of small shadow maps at the correct resolution to match receivers
  - Cascaded shadow maps (“Z-Partitioning”)
    - Render small number of shadow maps (~2-4) that split eye-space view frustum so each shadow map covers a smaller depth range and is therefore a better fit for the receivers in that partition
  - (and the list goes on, and on, and on)

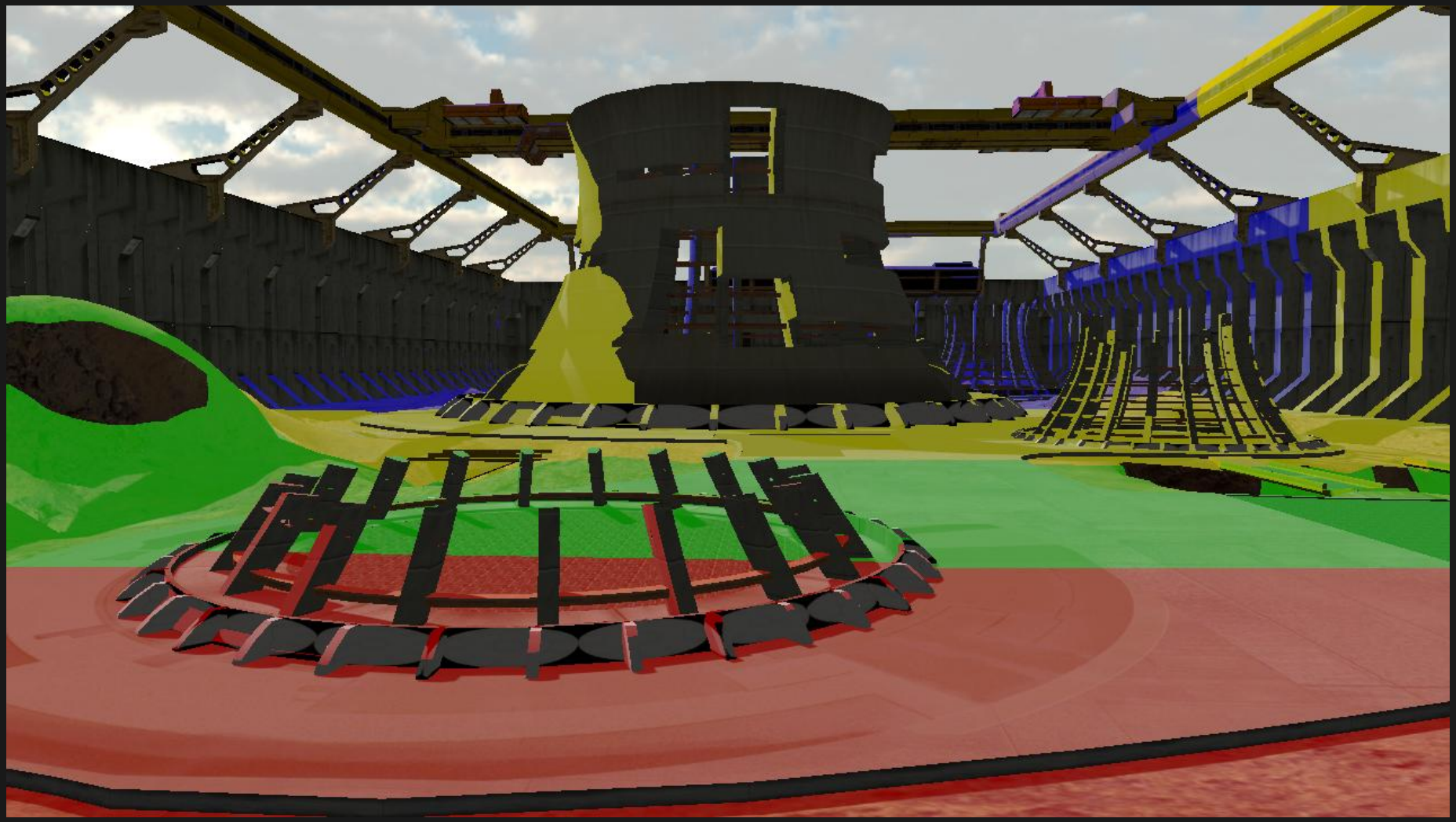
- The only approaches that directly address both perspective and projective aliasing are
  - Irregular rasterization
  - Adaptive grid-based methods

Z-Partitioning

- Split camera frustum in Z
- Use a different shadow map for each frustum partition



Z-Partitioning



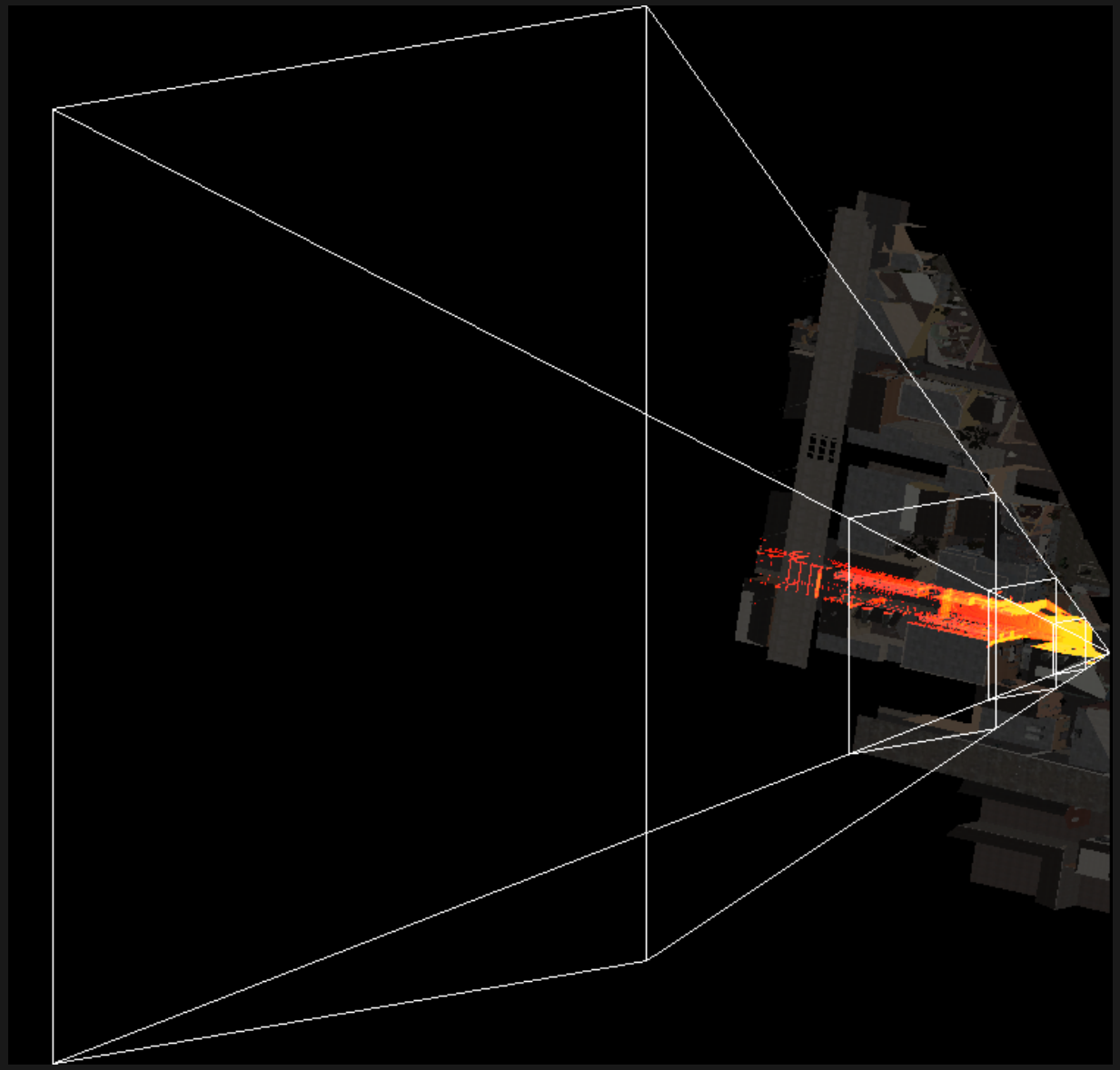
Z-Partitioning

Scene from Left 4 Dead 2, courtesy of Valve Corporation

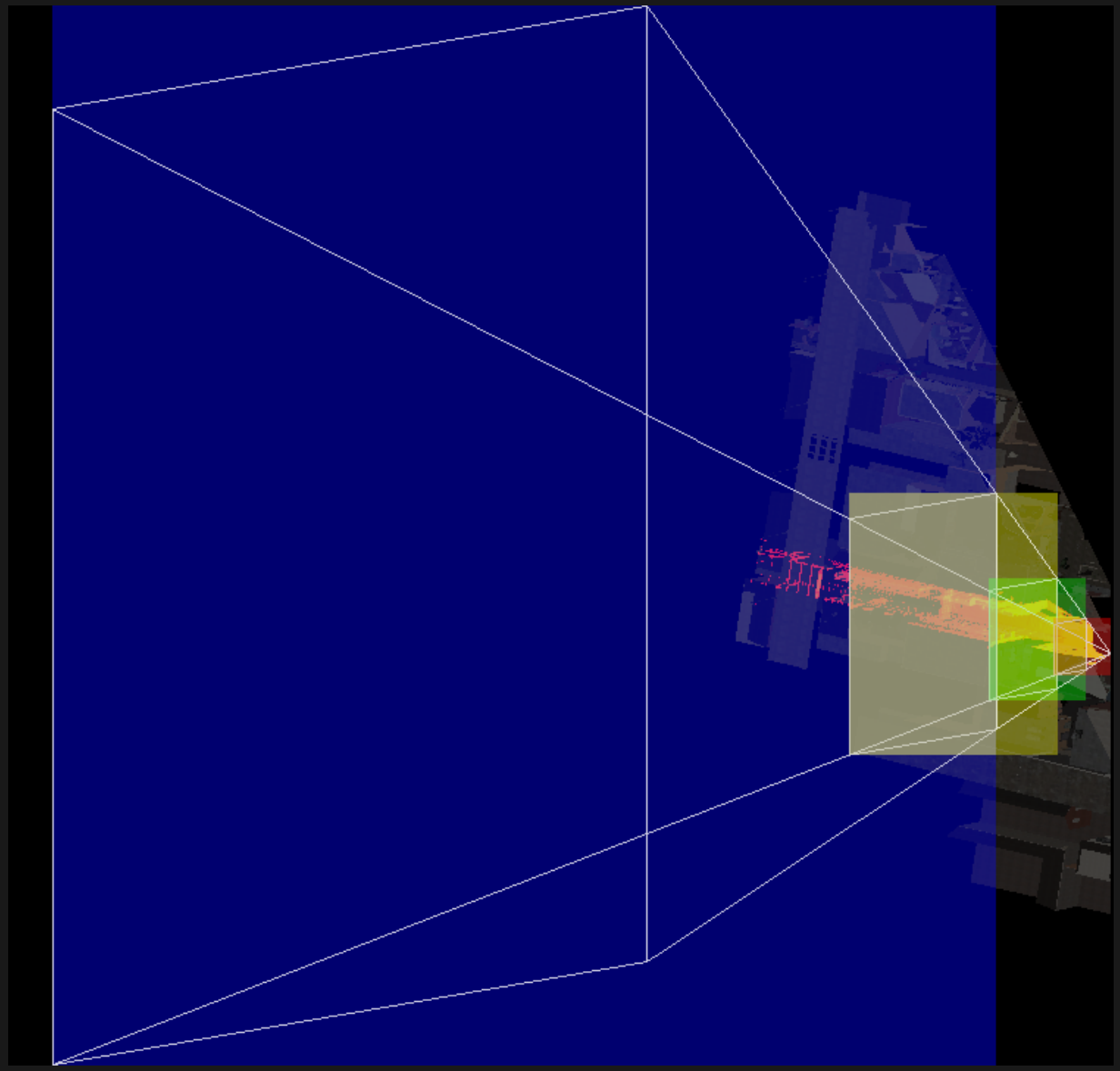
Z-Partitioning

Scene from Left 4 Dead 2, courtesy of Valve Corporation

Z-Partitioning in Light Space



Z-Partitioning Light Space Partitions



Reflections

Planar Reflections

- Ray tracing
  - "When hit specular surface, shoot new ray in direction determined by sampling specular lobe of BRDF"

- Rasterization
  - If planar surface (e.g., rear-view mirror in car), render image from back side of surface, clipped by bounding box of planar model (pre-pass)
  - In final rendering pass, query reflected-surface texture (pixel shader)

Reflections from Arbitrary Surfaces

- Ray tracing
  - "When hit specular surface, shoot new ray in direction determined by sampling specular lobe of BRDF"
- Rasterization
  - Render environment map (cube, dual paraboloid, etc) in pre-pass
  - In final rendering pass, query environment map based on reflected ray direction

Graphics *-Buffer Glossary

Overview

- Single depth layer
  - Z buffer
  - W bufffer
  - G buffer
- Multiple depth layers
  - A buffer
  - K buffer
  - F buffer

Z-Buffer (aka "Depth Buffer")

- Purpose
  - "Render geometry in any order and capture front-most depth layer"
- Key Attributes
  - Fixed memory regardless of amount of geometry
  - Accelerated in all current GPUs

W-Buffer

- Purpose
  - "Just like z-buffer but store depth in eye space (linear) rather than post-projective screen space."
- Key Attributes
  - Similar storage to z-buffer (but always floating point)
  - Different precision distribution across depth range

G-Buffer

- Purpose
  - Deferred rendering
  - "Render to an image-space buffer that captures per-pixel surface information such that the lighting can be computed in a post-processing image-space computation pass"

- Key Attributes
  - Fixed memory requirements
  - Decouples geometry from lighting

A-Buffer

- Purpose
  - "Render translucent and opaque geometry in any order, capture all depth layers, and resolve to final image"
  - Also capture per-sample coverage information for anti-aliasing
- Key Attributes
  - Unbounded memory requirements
  - Used in REYES / RenderMan

K-Buffer

- Purpose
  - "Render geometry that will generate fragments that are no more than k out of order, and use k-buffer to do final streaming sort"
- Key Attributes
  - Fixed memory requirements
  - Requires read-modify-write operations on framebuffer or custom blending logic

F-Buffer

- Purpose
  - "Capture all rendered fragments in a linear output stream"

- Key Attributes
  - Unbounded memory requirements
  - Indexed by re-rendering geometry
  - Does not support random indexing by pixel position without sorting entire f-buffer
  - (Much like geometry shader's "stream out")

N-Buffer

- Purpose
  - Pre-blurred images that don’t suffer from down-sampling artifacts
- Key Attributes
  - Recursively blurred stack of images that are all the same size
  - Like mipmaps, but with no down-sampling
  - Takes huge amount of memory unless image size is small

Billboards

- Fine geometry (sub-pixel) and volumetric media are usually handled with “billboards”
  - A “billboard” is a camera-aligned, texture-mapped, partially transparent quad
  - Used for hair, fences, smoke, foliage, grass, …
  - No depth test. Alpha blending. Must render billboards in depth order.

Billboards

Summary

- Many of the illumination and surface material effects supported in ray tracing or REYES can be implemented in the current programmable shading pipeline
  - Often involves a pre-pass to cache non-local visibility
  - These caches almost always introduce artifacts, but greatly speed up rendering
- Boundaries between rasterization and ray tracing are blurring
  - (Limited) ray tracing in pixel shaders is increasingly common
  - Ray tracing framebuffers is common
  - Rasterization is highly-optimized special-case ray tracing

Homework 1

- Will be on the web page this evening
- Due Monday, 1/24 (1.5 weeks)

- Join the class mailing list to get help from me and support each other with logistics/systems problems

Backup

Sample Distribution Shadow Maps

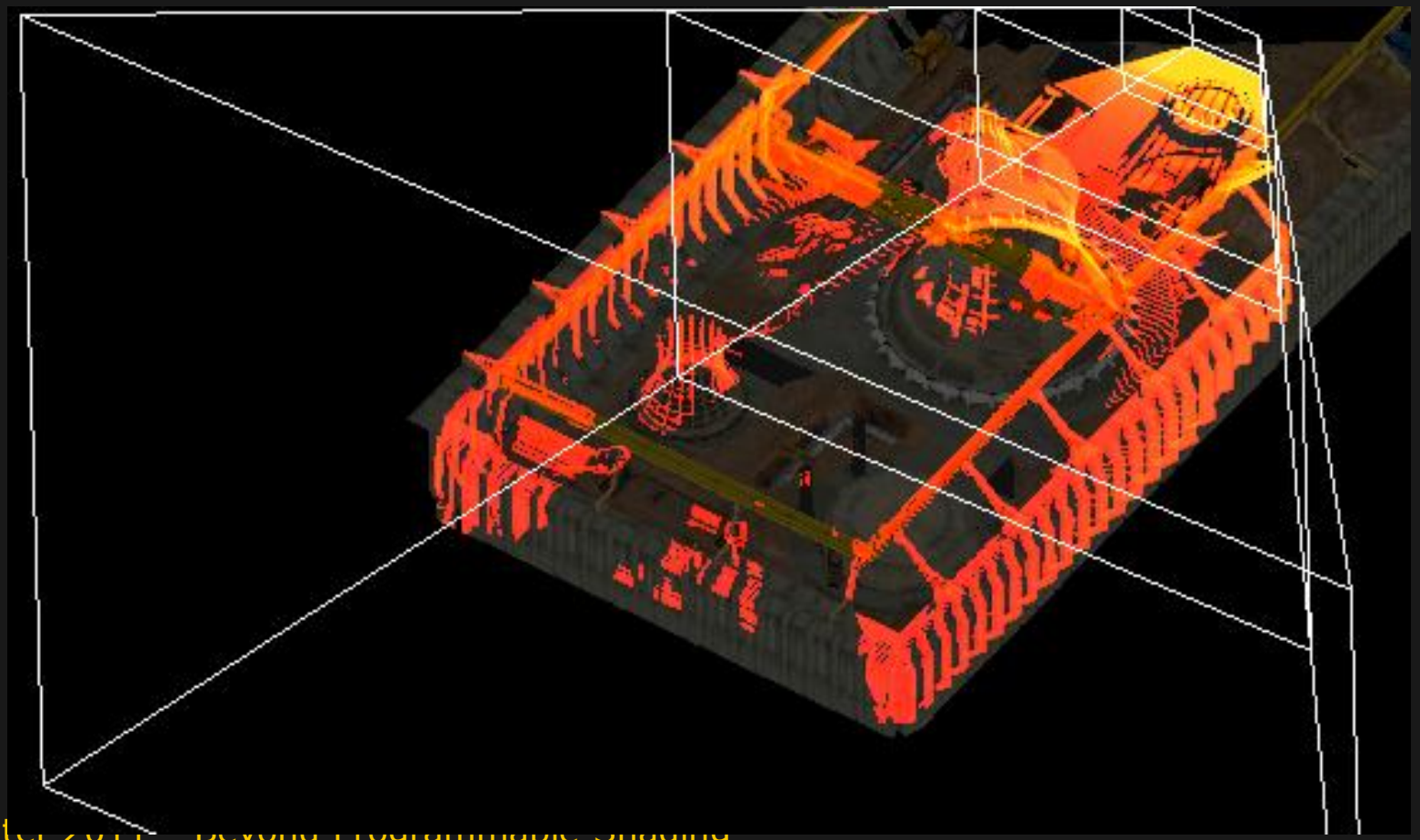
Slides by Andrew Lauritzen, Intel

Sample Distribution Shadow Maps

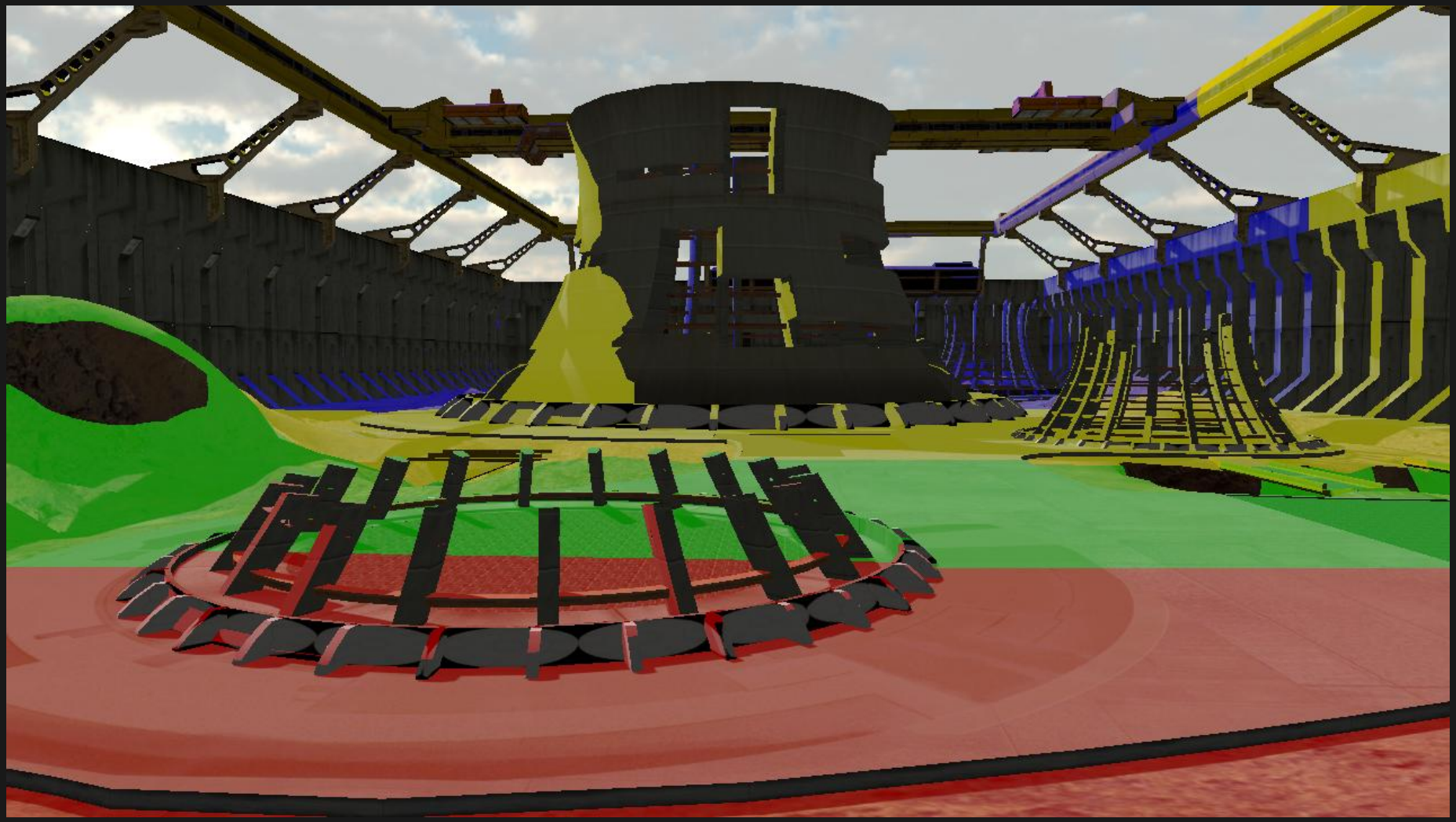
- Needs of real-time applications
  - Real-time applications need to constrain memory and time: "Authorable performance"
  - RMSM and IZB guarantee quality but vary time/memory
- SDSM idea
  - "What is the best shadow quality we can deliver using a fixed amount of memory and time?"
  - Automatically place a fixed number of shadow map partitions based on shadow receiver samples (same input as IZB and RMSM but different optimization)
- Addresses perspective aliasing directly and projective aliasing "when we get lucky"

Z-Partitioning

- Split camera frustum in Z
- Use a different shadow map for each frustum partition



Z-Partitioning

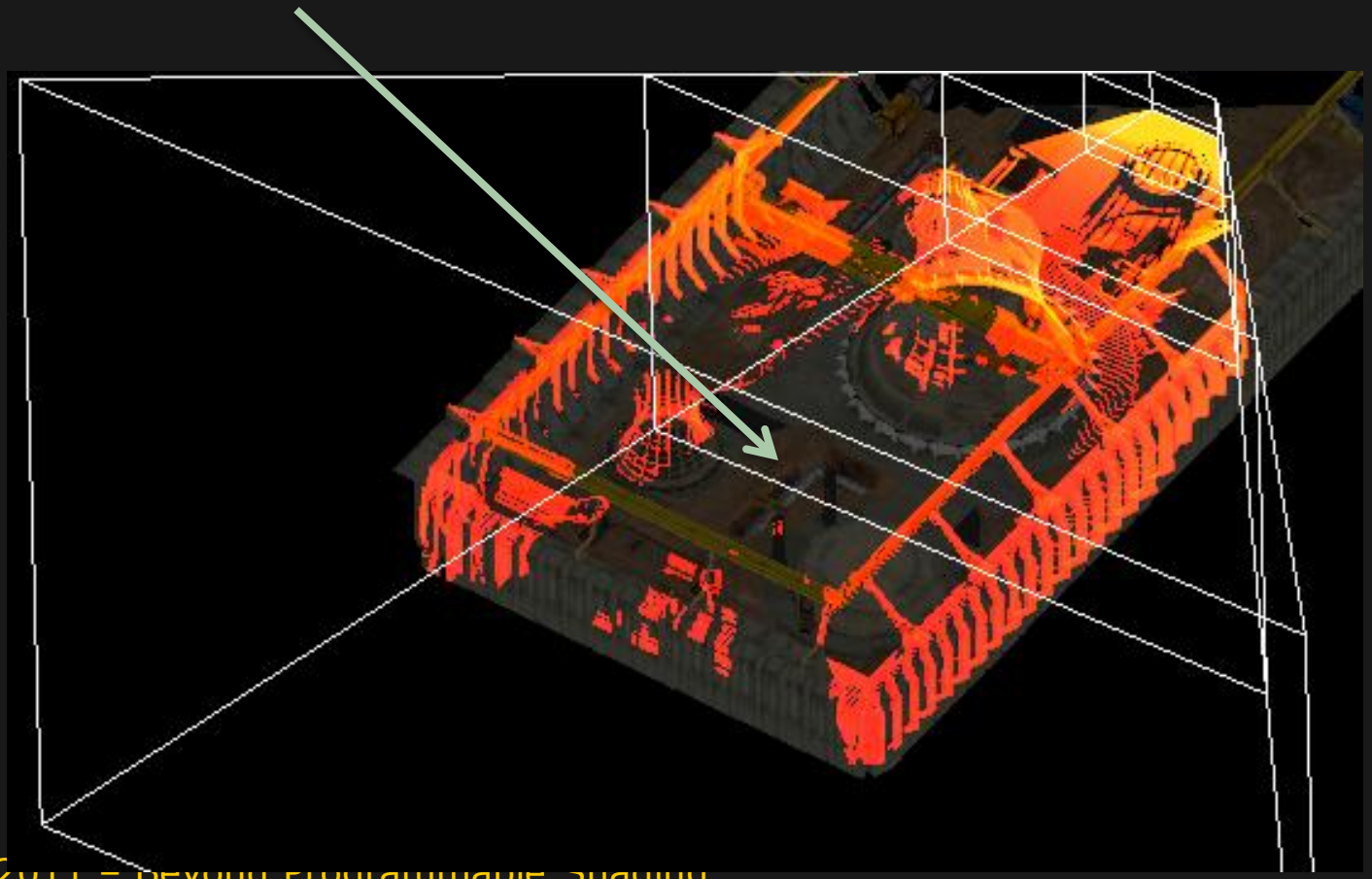


Where to Partition Z?

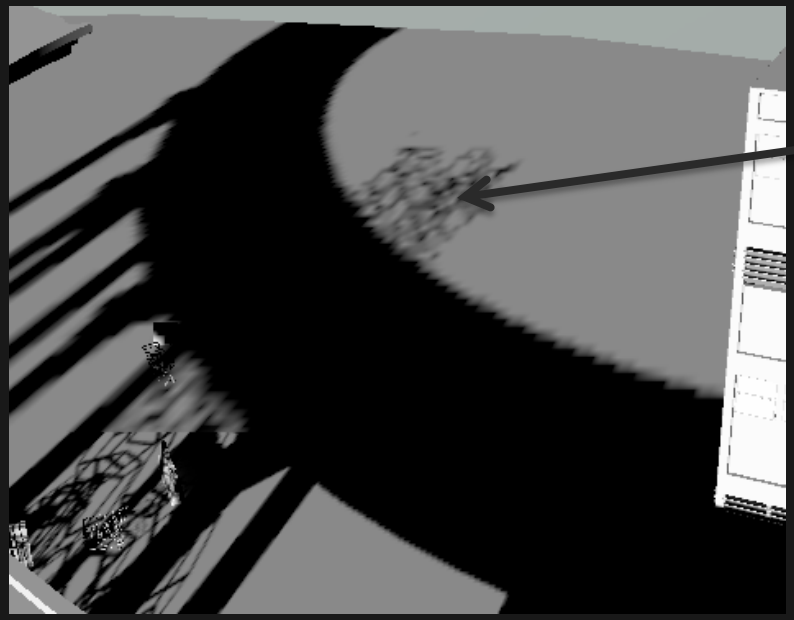
- Logarithmic is the best [Lloyd et al. 2006]
  - But only if the entire Z range is covered!
  - Needs tight near/far planes
- Parallel-Split Shadow Maps [Zhang et al. 2006]
  - Mix of logarithmic and uniform
  - Requires user to tuneable a parameter
    - Optimal value related to tight near plane…
- In practice, artists tune for specific views
  - Tedious and not robust to scene/camera changes
  - Ultimately suboptimal for arbitrary views

Where to place shadow maps?

- Axis-aligned bounding box of frustum segment in light
- Does not consider vast segments of the shadow map that are occluded

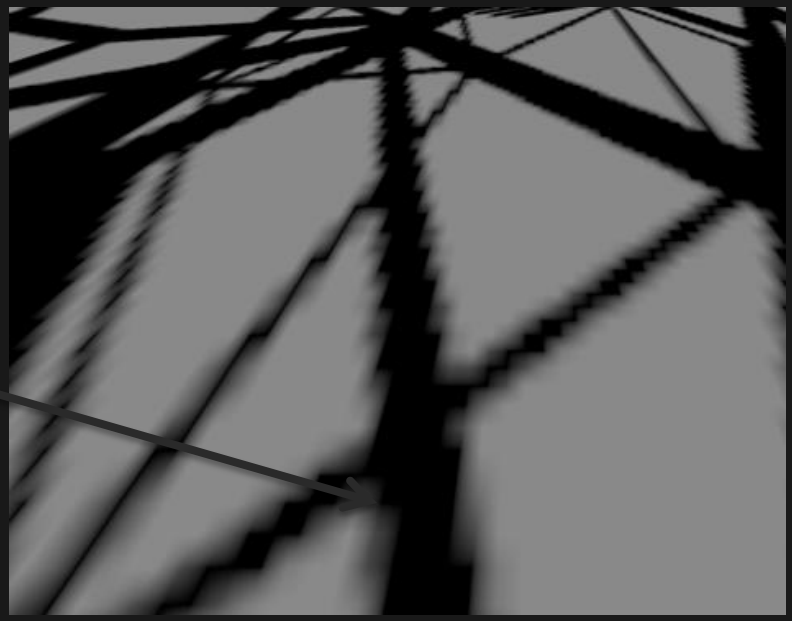


Static Partitions (PSSM)



Too little resolution far!

Too little resolution close!



Sample Distribution Shadow Maps

- Analyze the light-space sample distribution
  - Find tight Z min/max
  - Partition logarithmically based on tight Z bounds
  - Fully automatic; adapts to view with no need for tuning

- Compute tight light space bounds for each partition
  - Min/max of sample coordinates in light space
  - Avoids including occluded samples in shadow map
  - Greatly increases useful shadow resolution

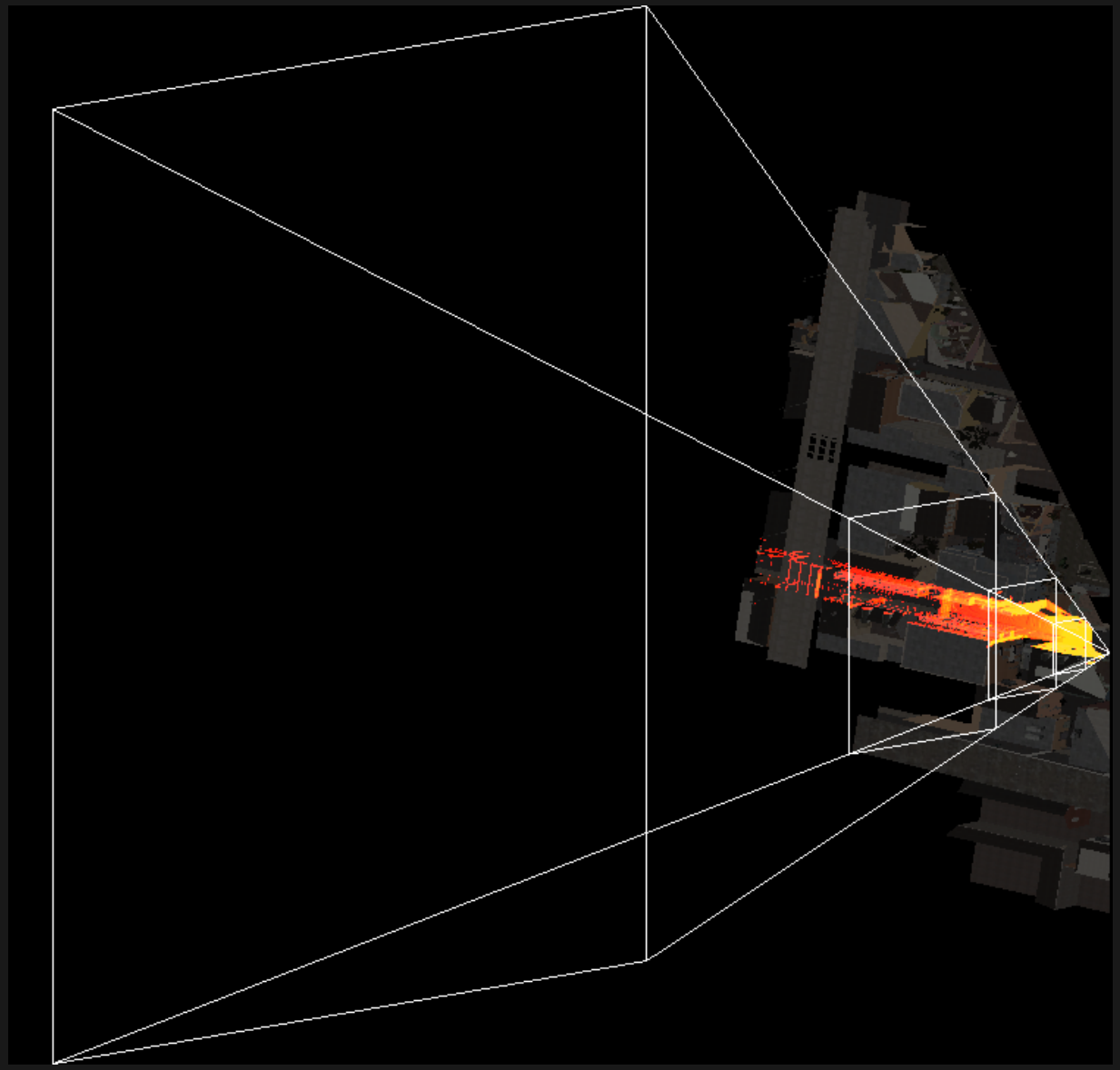
Example: PSSM

Scene from Left 4 Dead 2, courtesy of Valve Corporation

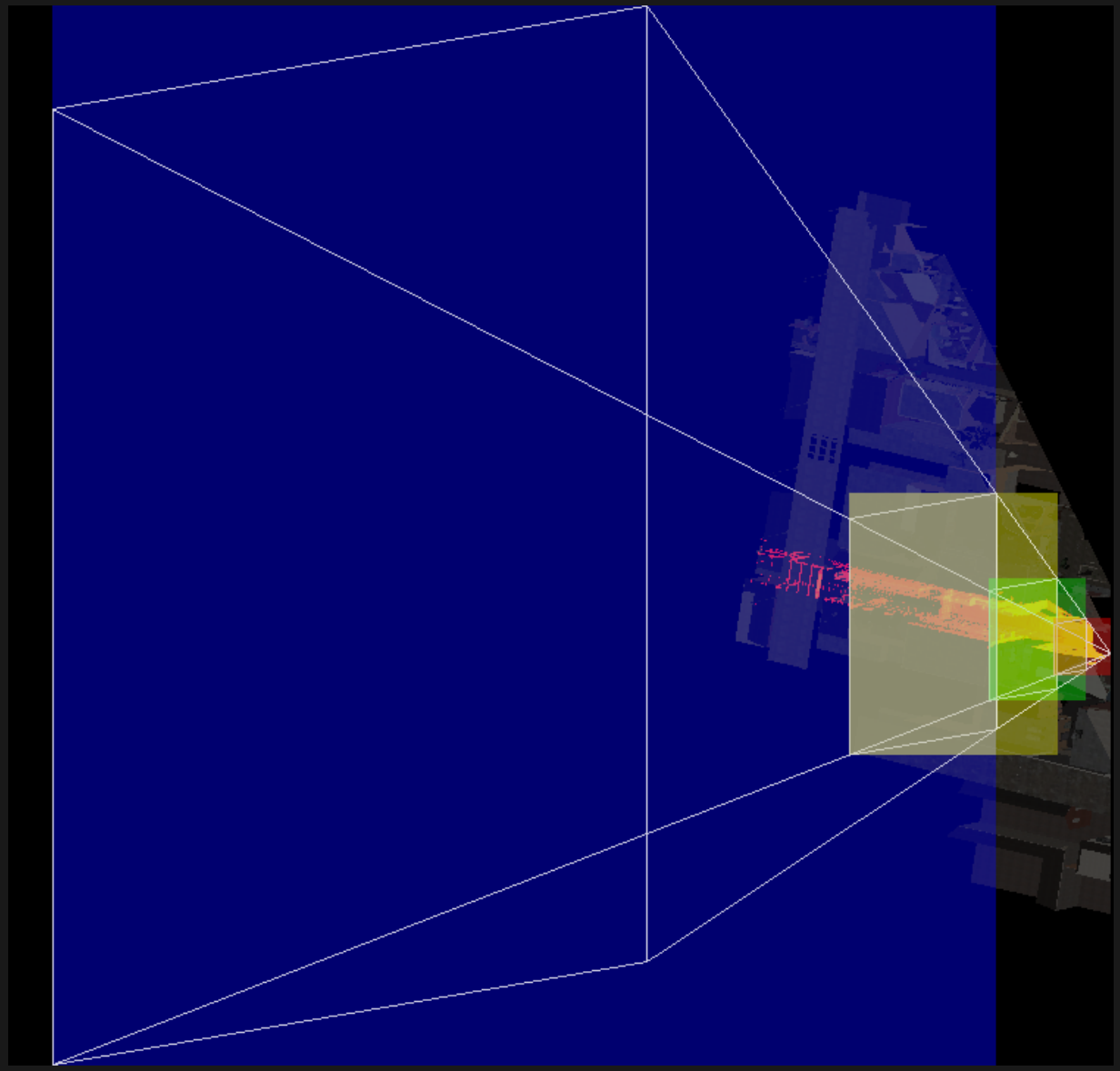
Example: PSSM Partitions

Scene from Left 4 Dead 2, courtesy of Valve Corporation

Example: PSSM Light Space



Example: PSSM Light Space Partitions



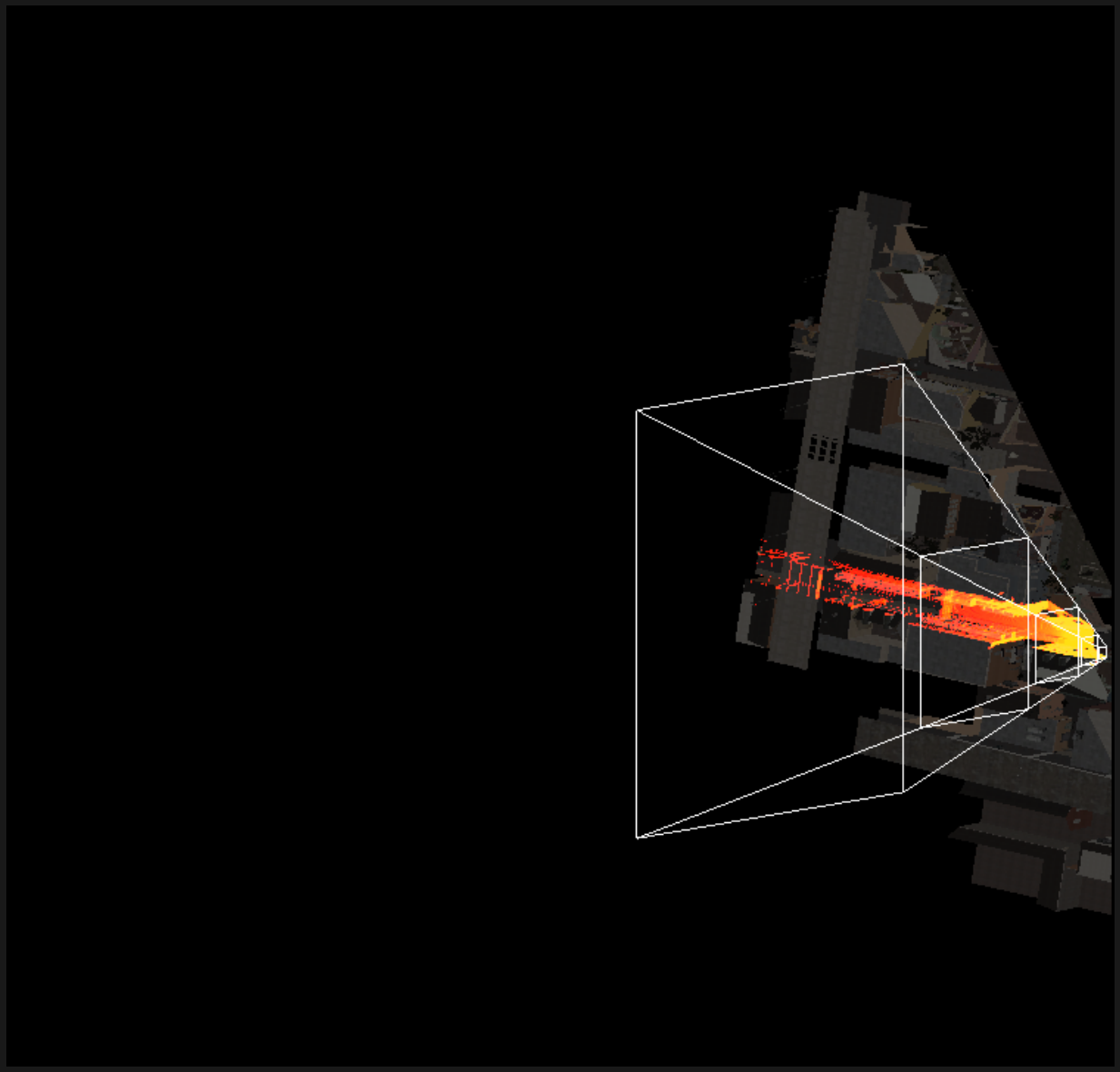
Example: SDSM

Scene from Left 4 Dead 2, courtesy of Valve Corporation

Example: SDSM Partitions

Scene from Left 4 Dead 2, courtesy of Valve Corporation

Example: SDSM Light Space



Example: SDSM Light Space Partitions