



DelphiDay
italian conference

3D Rendering Techniques for CAD applications

Leveraging advanced rendering techniques in CAD applications,
powered by Afterwarp Framework



Yuriy Kotsarenko

Afterwarp Interactive - Owner



<https://afterwarp.io>



ykot@afterwarp.io



<https://github.com/yunkot>



DelphiDay

italian conference

19-20 Giugno 2025
Piacenza



wintech
italia

OPEN-SOURCE PROJECTS

<https://github.com/yunkot>

Afterwarp Framework

<https://afterwarp.io>

MicroPXL

<https://github.com/yunkot/MicroPXL>

Other Projects and Resources

<https://asphyre.net>



19-20 Giugno 2025
Piacenza





3D Rendering: introduction

- A 3D model is represented as a triangular mesh
 - **Vertex Buffer**: each vertex has position, normal, texture coordinates, etc.
 - **Index Buffer**: an array of indices to vertex buffer forming triangles
- 3D vertex coordinates are “transformed” by multiplying them with a 4x4 matrix, assuming 4th vertex coordinate as 1
 - This performs an Affine and/or Projective transformation



3D Rendering: introduction (cont'd)

- Vertices (normals, tangents, etc.) are transformed between:
 - **Object** (or “Model”) Space
 - **World** Space
 - **View** (or “Camera”, or “Eye”) Space
 - **Clip** (or “Projection”) Space
 - **NDC** (Normalized Device Coordinates) Space
 - **Screen** space



Rendering pipeline simplified

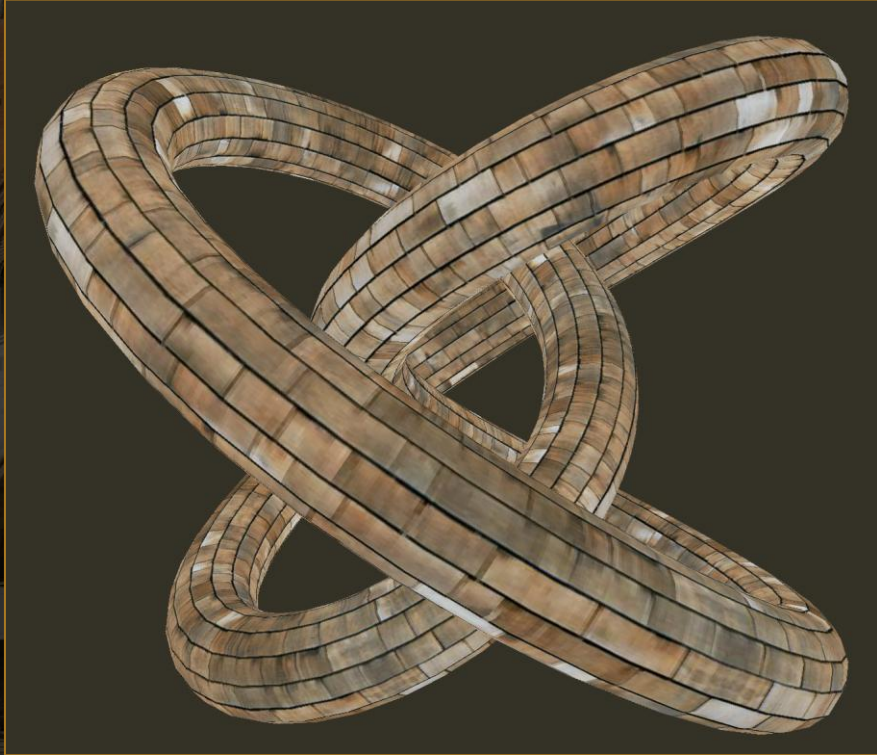
- **Vertex** and **Index buffers** are sent to the GPU
- **Draw call** is issued to the GPU
- **Vertices** from **vertex buffer** are transformed by **Vertex Shader** to **Screen Space**
- Each **triangle** is **rasterized** on the screen
- For **every pixel** in the triangle, **Pixel Shader** is executed on the GPU to calculate the final pixel color



Rendering of triangles

- Earlier software renderers sorted triangles by depth
- Triangles were rasterized from back to front
 - Works reasonably well for very basic 3D shapes
 - Requires direct and immediate access to ALL triangles in the whole scene for correct rendering
 - Does not work when two or more triangles intersect
 - Prone to “triangle fighting”

Triangle sorting: practical example





The magic of a Depth Buffer

- Target rendering surface also stores depth in addition to color
- Typically implemented using two separate rendering surfaces
 - Color rendering surface
 - Depth buffer
- Eliminates the limitations of triangle sorting
- Requires more CPU work, but “free” on the GPU



Forward rendering technique

- DirectX 7 and earlier
 - Fixed-Function Pipeline (FFP) or Software Rendering
 - 3D lighting is performed per vertex
- DirectX 8 and later
 - Programmable pipeline: shaders!
 - 3D lighting is performed per pixel



Forward rendering technique (cont'd)

- Easy to learn and understand, simple to work with
- Easy to apply different types of materials
- Multisampling antialiasing, semi-transparency
- Suffers from overdraw, many shader permutations
- Number of simultaneous lights is limited
 - Multiple passes are required for many lights



Deferred rendering technique

→ Target rendering surface: **G-Buffer**

- Color, depth, normal, material ID, etc. (specular, reflection)

→ First pass fills G-Buffer

- 3D scene geometry is rendered

→ Second pass performs lighting on G-Buffer

- Lights are “rendered” as cones/spheres



Deferred rendering technique (cont'd)

- Decoupled pipeline is easier to maintain
- Number of light sources is practically unlimited
- Requires more GPU memory and bandwidth
- Antialiasing and semi-transparency are difficult
- Total number of materials could be limited

Deferred rendering: practical example





Forward vs deferred rendering

→ Forward rendering

- Unlimited materials, but very limited number of lights
- Antialiasing through multisampling, direct semi-transparency
- Shader complexity, significant overdraw

→ Deferred rendering

- Practically unlimited number of lights, limited materials
- Antialiasing: super-sampling, selective blurring (FXAA, SMAA, TAA)
- Semi-transparency: stochastic-like tricks or through forward-rendering



Hybrid forward/deferred rendering

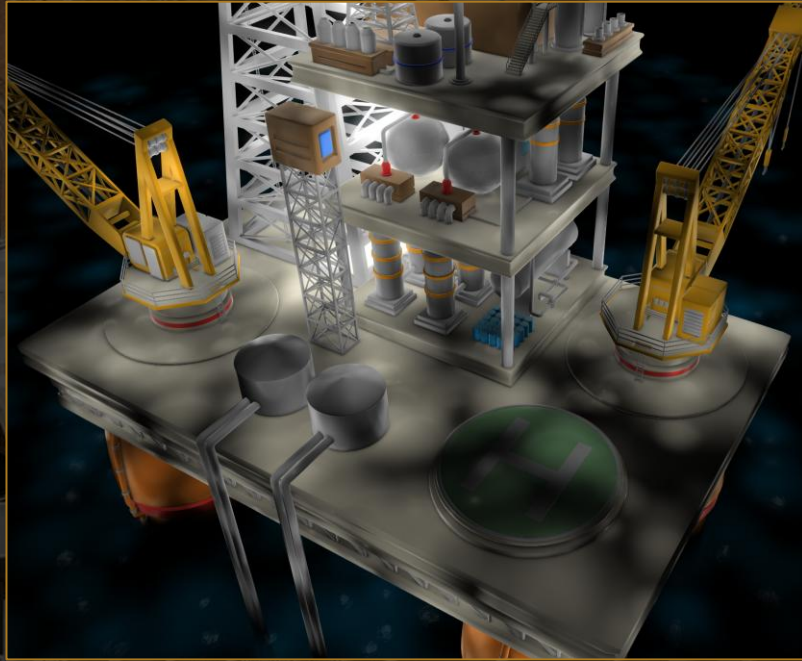
- Usually involves a separate **depth pre-pass**
 - Requires a smaller G-Buffer: depths and normals
 - Practically eliminates overdraw, just like in deferred rendering
- G-Buffer can be used for screen-space effects:
 - Depth of field (DoF), Bokeh, Ambient Occlusion (AO), Fog
- The screen is partitioned and only lights affecting a particular partition are used during **forward pass**
 - Depth comparison function is set to “equal”



Afterwarp v3.x: rendering technique

- Hybrid deferred and forward rendering
 - **Depth pre-pass** is optional
 - Required for fog and water, reduces overdraw
 - Supports up to 65 536 simultaneous lights
 - Multisampling antialiasing, unlimited materials
 - Order-independent transparency (OIT)
 - Approximated and/or accurate in a single pass

Hybrid forward – deferred rendering: practical example





Real-time shadows

→ Common techniques:

→ **Shadow Volumes** and **Shadow Mapping**

→ Typical characteristics:

→ Hard penumbra around the shadow shape

→ Noticeable “jumping” of shadow during movement

→ Other visible artifacts



Modern techniques for soft penumbra

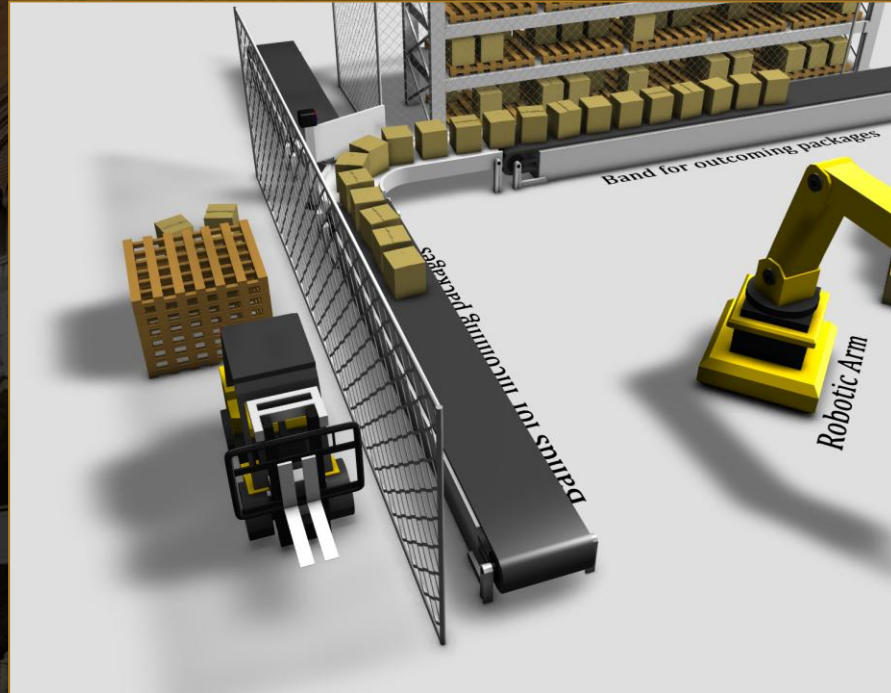
- Percentage-Closer Filtering (PCF) Shadows
 - Efficient on modern GPU
 - Penumbra is noisy
- Variance Shadow Maps (VSM)
 - Noticeable depth artifacts between different objects
- Exponential Shadow Maps (ESM)
 - Noticeable light leakage at objects positions



Realistic soft and natural shadows

- Afterwarp v3 supports **ESM** and **E(V)SM** shadows
 - High quality shadows with soft penumbra
 - Smooth shadow transition during object movement
 - Relatively low memory usage
 - Uses a single atlas for multiple shadow maps
 - Many shadow-casting light sources

Realistic soft and natural shadows: practical example





Order-independent transparency

- Naïve rendering of semi-transparent objects
 - Does not work with multiple transparent layers
 - ❖ Otherwise, these have to be depth-sorted!
- **Depth-peeling**: traditional multi-pass approach
 - Relatively slow and heavy on the GPU
 - No antialiasing other than super-sampling
- Afterwarp v1 supported **dual depth-peeling**



Order-independent transparency (cont'd)

→ **Alpha-to-coverage trick**

- Very fast, but requires multisampling
- Antialiasing works naturally “out of the box”
- Fixed transparency levels: 2, 4 or 8

→ Supported in all Afterwarp versions!

- Works “in place”, not accurate in HDR rendering

→ **Stochastic technique**: somewhat similar



Order-independent transparency (cont'd)

→ **Weighted Average (WAVG) technique**

- Relatively fast, but is an approximation!
- Super-sample antialiasing only
- Works best for transparency less or equal to 50%

→ Supported in all Afterwarp versions!

- HDR accurate! Rendered separately in a single-pass

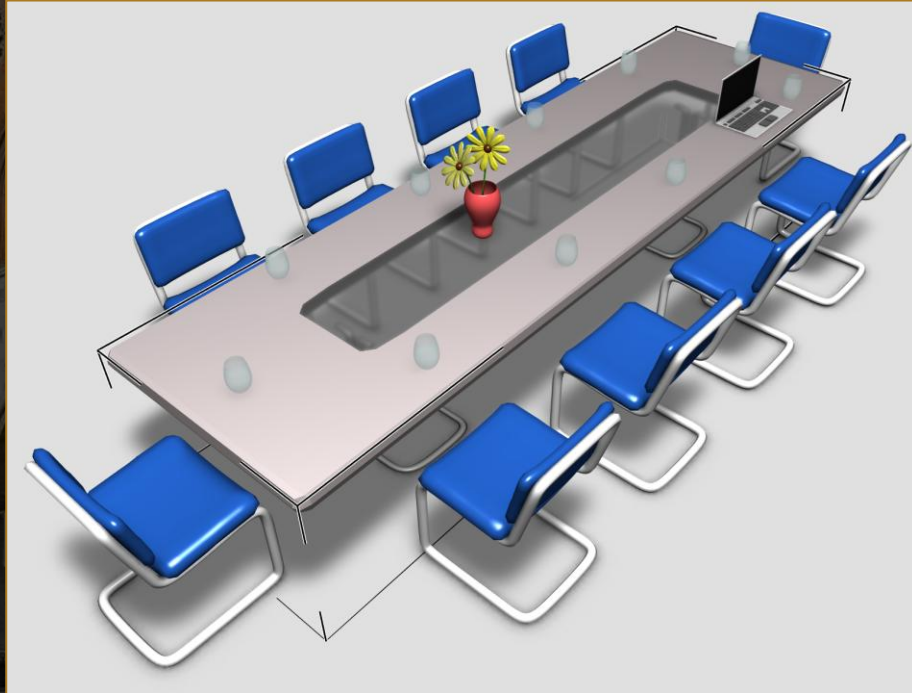


Order-independent transparency (cont'd)

→ **Afterwarp v3 hash-based implementation**

- Accurate with anti-aliasing in a single-pass!
- Optionally supports “Frosted Glass” effect
- Needs a discrete GPU with fast memory (GDDRx)
- Some integrated GPUs also work well (e.g. AMD)
- Memory usage depends on the scene, controllable

Order-independent transparency (OIT): practical example





Text in 3D: techniques

→ Naïve drawing

- Blurry or pixelated text, almost unreadable

→ Signed Distance Fields (SDF) w/SuperSample

- Sharp looking text, even at angles, optional outline

→ Text as a real 3D mesh with depth

- Real-time! Can be optionally curved.

Text in 3D: practical examples



Text in 3D can also be flat (no extrusion).

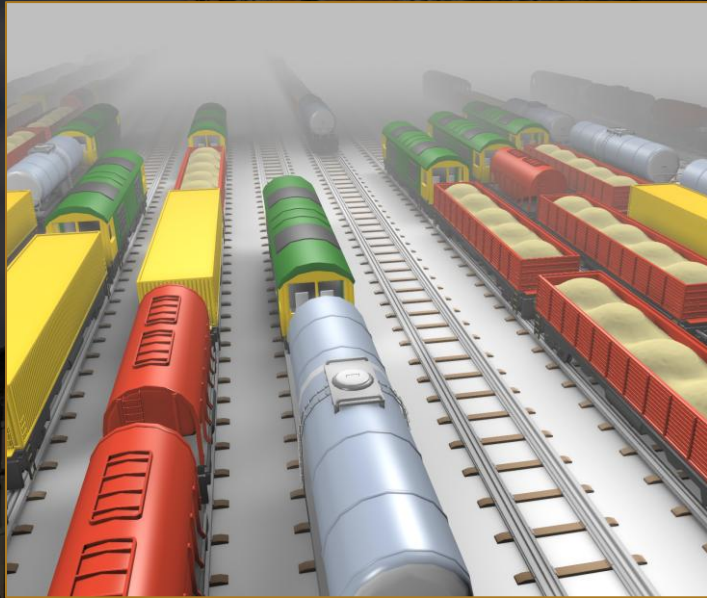
e is a standard text ex
frame index is (629), skippe
rendering test appli
可以用多种不同的语言
is rendered at cer



Fog

- Aesthetically pleasing, hides artifacts at distance
- Fixed-Function Pipeline and typical forward renderers apply fog directly when drawing geometry
- Fog is naturally a screen-space effect
 - In deferred renderer it is relatively cheap to do on GPU
- In Afterwarp requires a **depth pre-pass**

Fog: practical examples

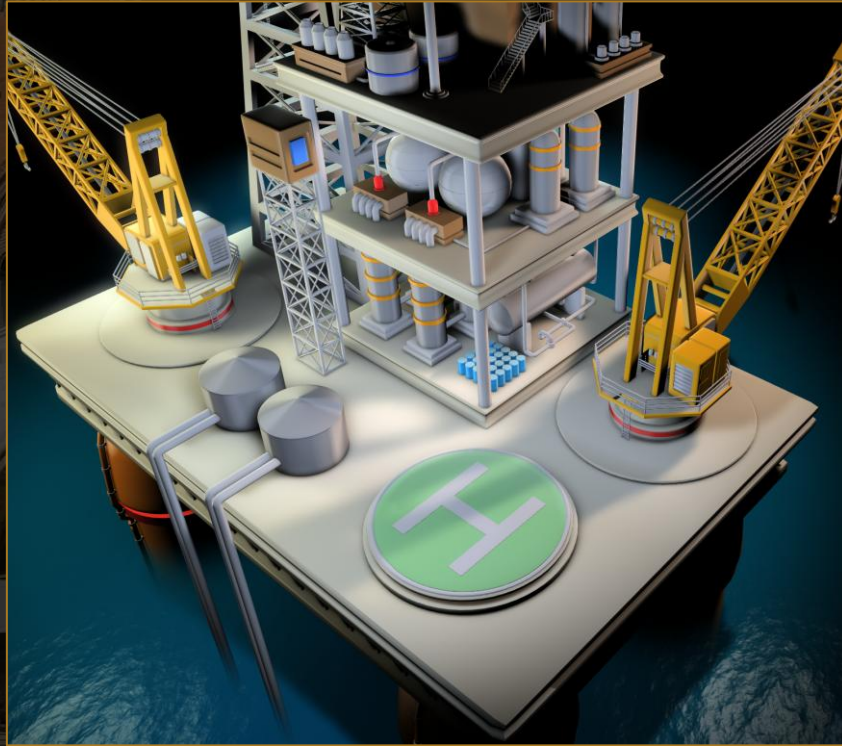




Water

- Requires special care to combine with “solid” geometry, otherwise has ugly “cuts” at seams
- Many different techniques exist
- In Afterwarp, this requires a **depth pre-pass**
 - Simulated and rendered as an infinite plane with configurable wave animation parameters

Water: practical example





Where to go from here?

- Global illumination, radiosity
 - Many recent research and implementations exist
 - Can be costly, but improves lighting significantly
- Ray-tracing
 - Supported on newer GPUs and APIs
 - Direct3D 12 and Vulkan
 - Shadows, reflections, refractions, etc.



Discussion