Game Rendering Techniques

Greg Turk
School of Interactive Computing
and GVU Center

Game Engines

- Software for creating games
  - Physics
  - Collision detection
  - Artificial intelligence
  - Scripting (e.g. game mechanics)
  - Rendering

Game Engine Rendering

- Many rendering techniques incorporated
- Tailored for use in games
- Tuned for fast rendering

Popular Game Engines

- Unity 3D
- Unreal
- CryEngine
- Source (from Valve)
- GameMaker

- Often free educational versions

Real-Time Techniques

- Two classes of techniques:
  - Faster Rendering
  - Higher Quality
- Often work in opposition to each other
- Balance shifts based on hardware

Immediate Mode Rendering

- CPU sends polygons to GPU one-by-one
- Allows per-frame changes to objects (e.g. animated waves)
- Slow, due to communication speed between CPU and GPU
- We use immediate mode in Processing

Retained Mode Rendering

- Send polygons to GPU *once* (store on GPU)
- Collection of polygons called *Vertex Buffer Object* (VBO)
- CPU asks GPU to render given VBO
- Fast, because polygons already on GPU
- Processing has VBO commands, too

Games use Retained Mode

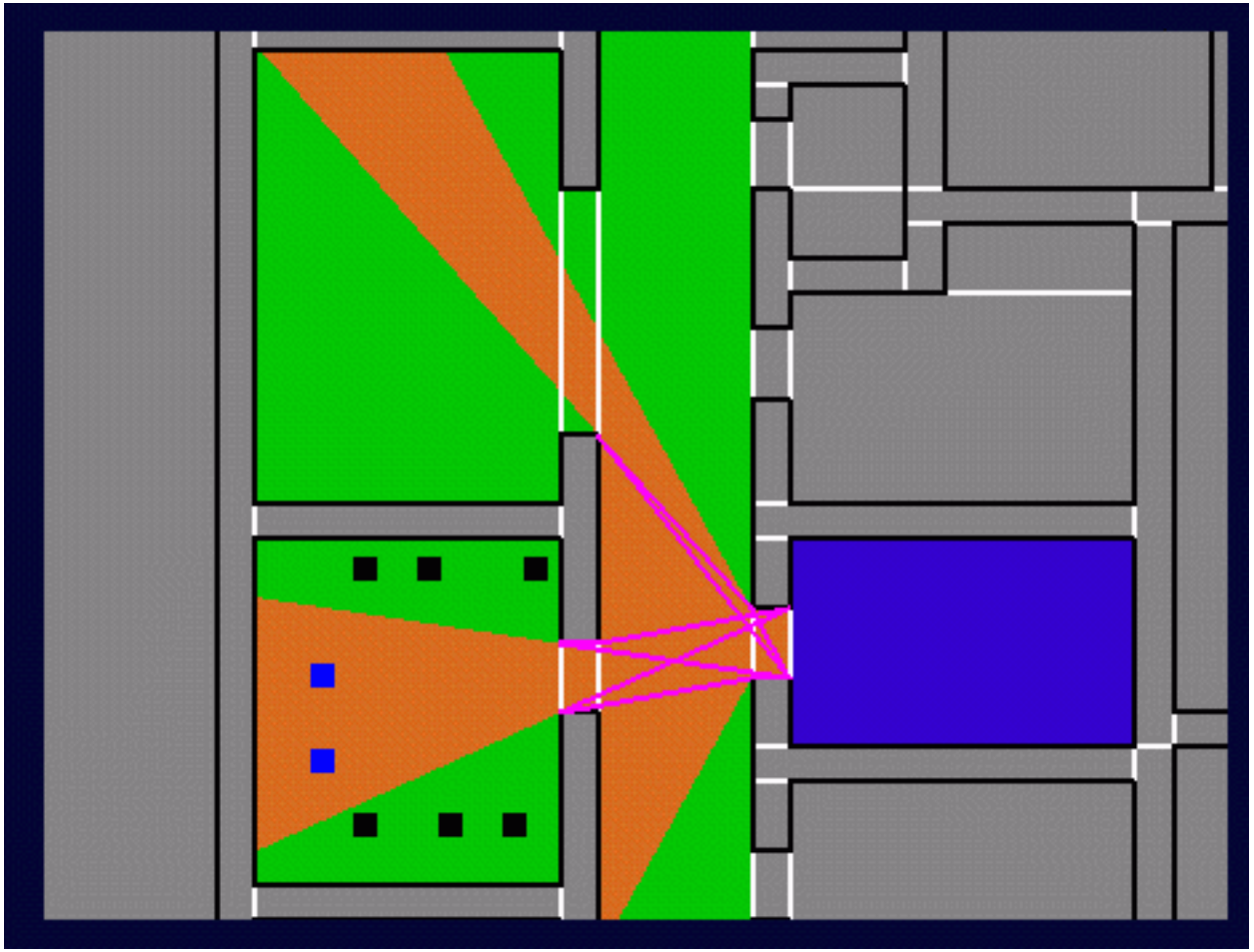
- All games use retained mode graphics
- Speeds up rendering dramatically
- Harder to change object's geometry
  - Use vertex shaders
  - More complicated than CPU programming

Draw Fewer Polygons

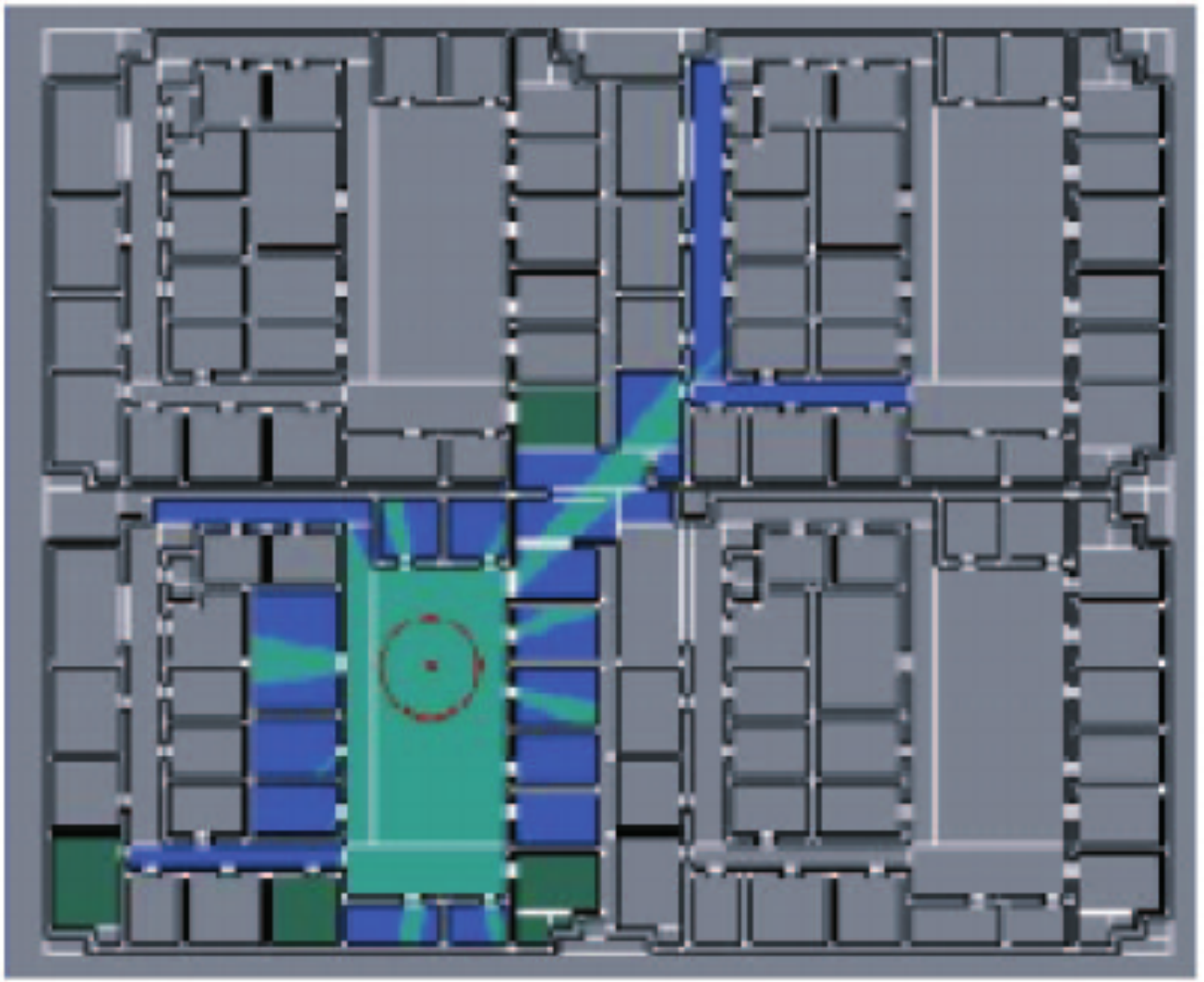
Potentially Visible Sets

- Used for indoor scenes
- Divide interior into rooms
- Calculate between-room visibility
- Only draw subset of all rooms
- Two variations:
  - Pre-calculate room visibility
  - Calculate visibility on-the-fly (portals)

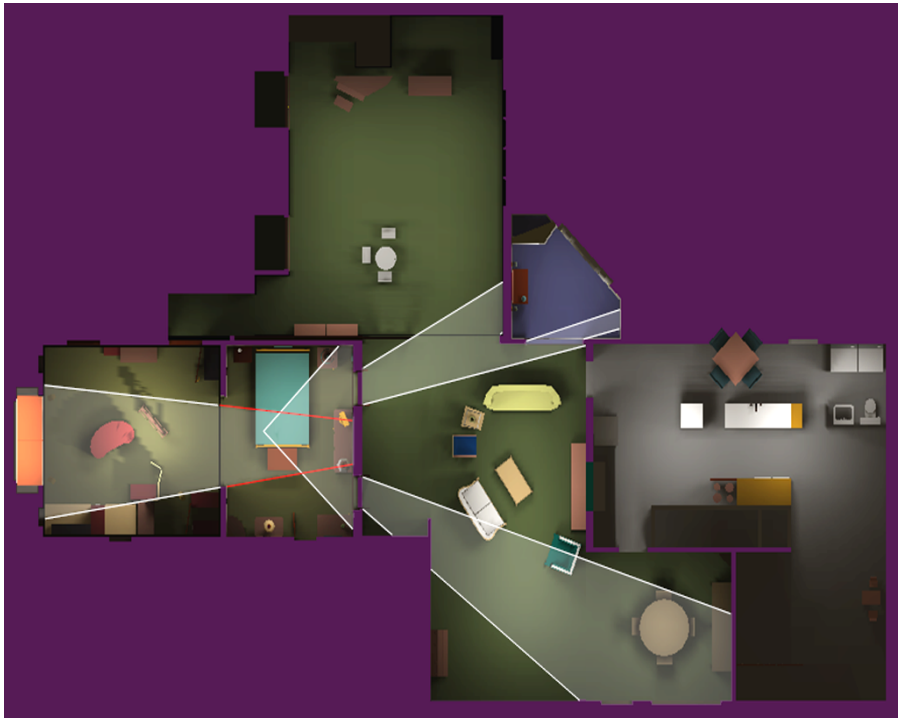
Potentially Visible Sets



Potentially Visible Sets



Portals



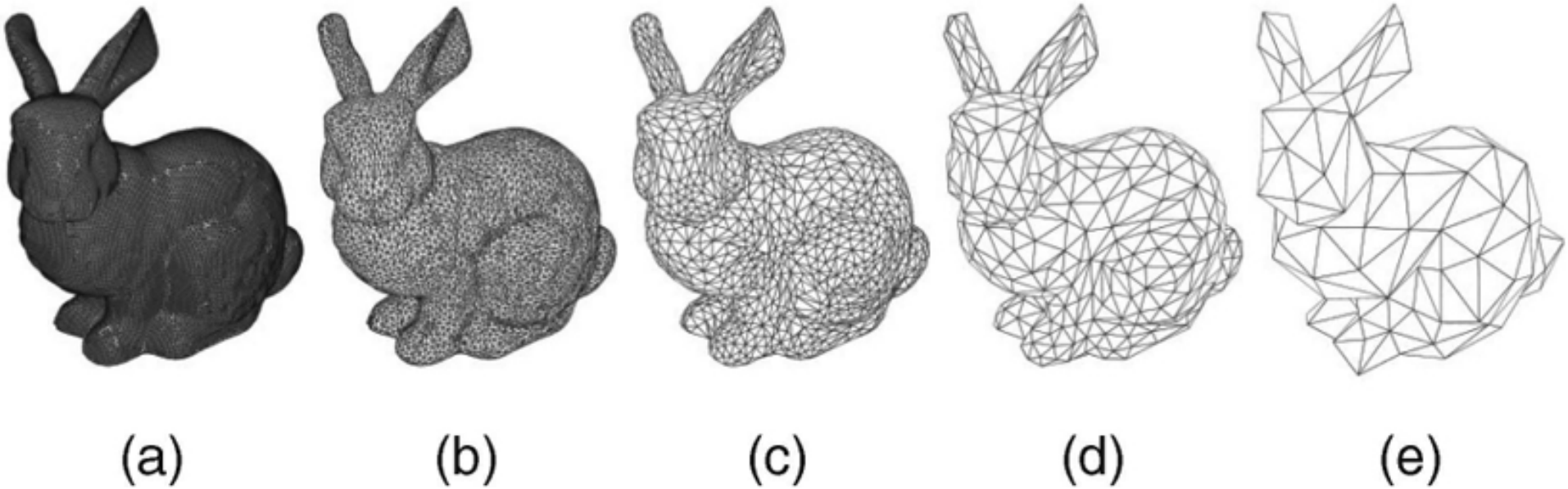
Level of Detail

- Create several meshes for an object
- Versions have different polygon counts
- Automatic or human-created models

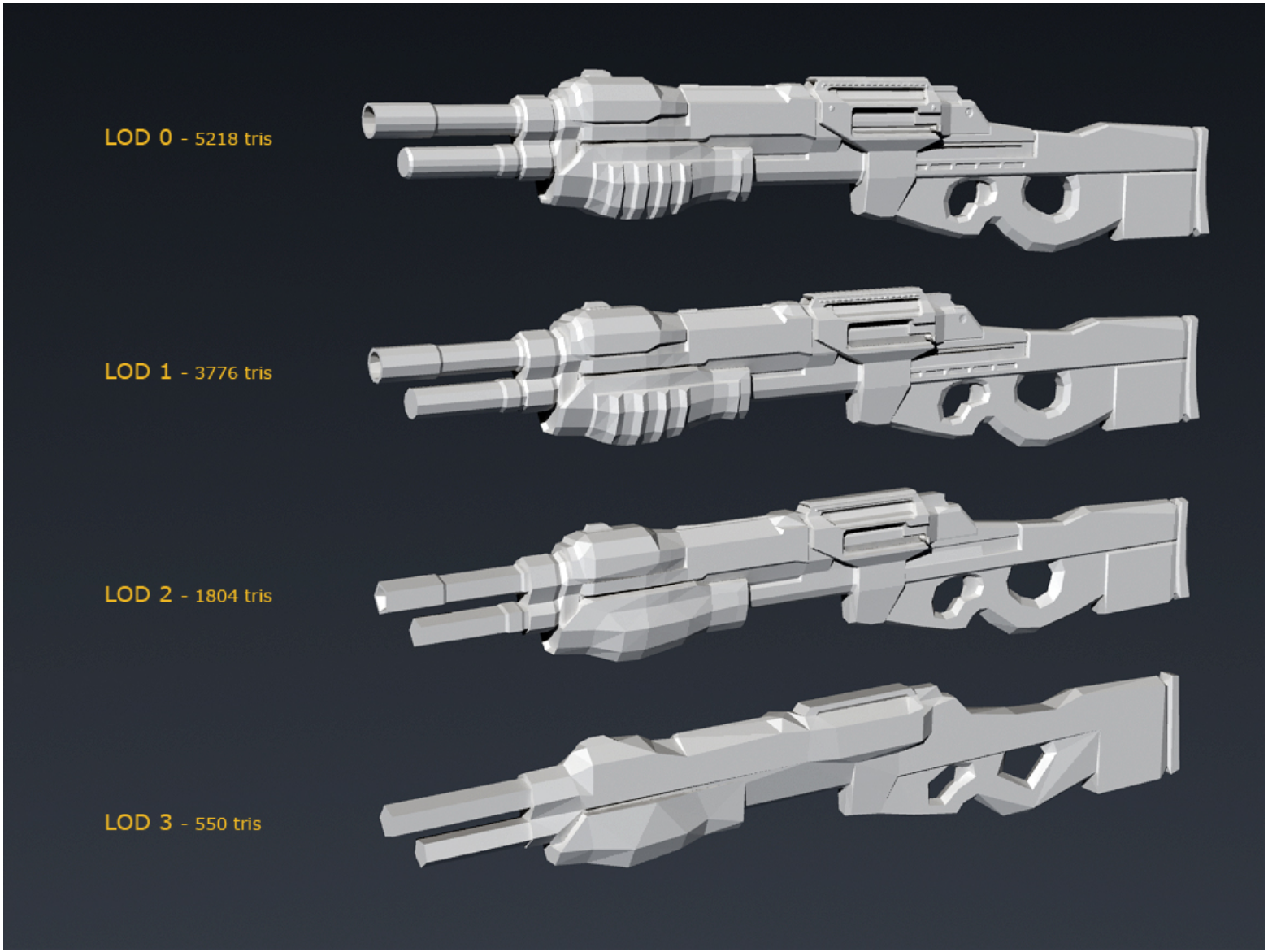
- High-resolution for close-up views
- Low-resolution for far-away views

Level of Detail

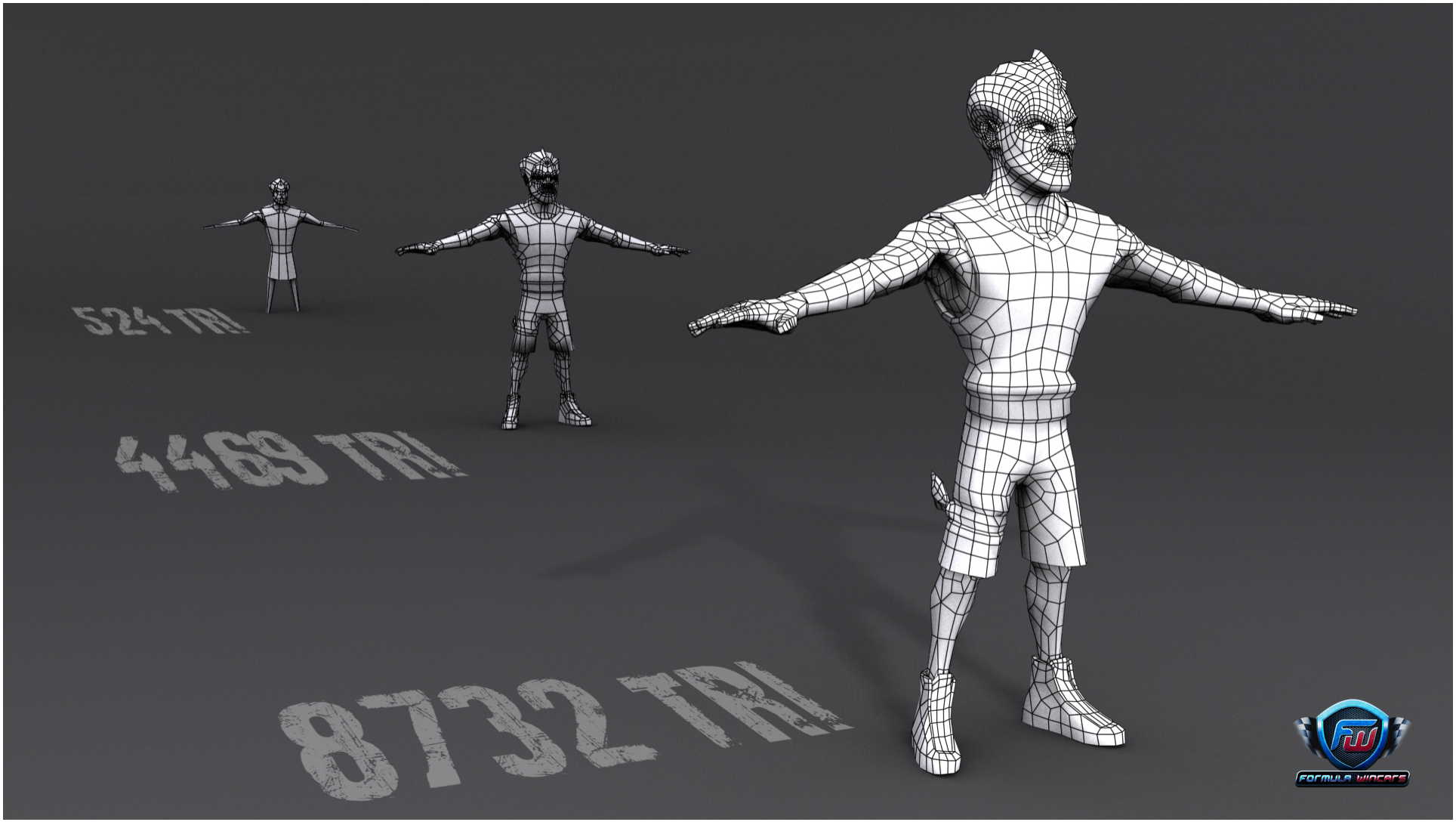
Level of Detail



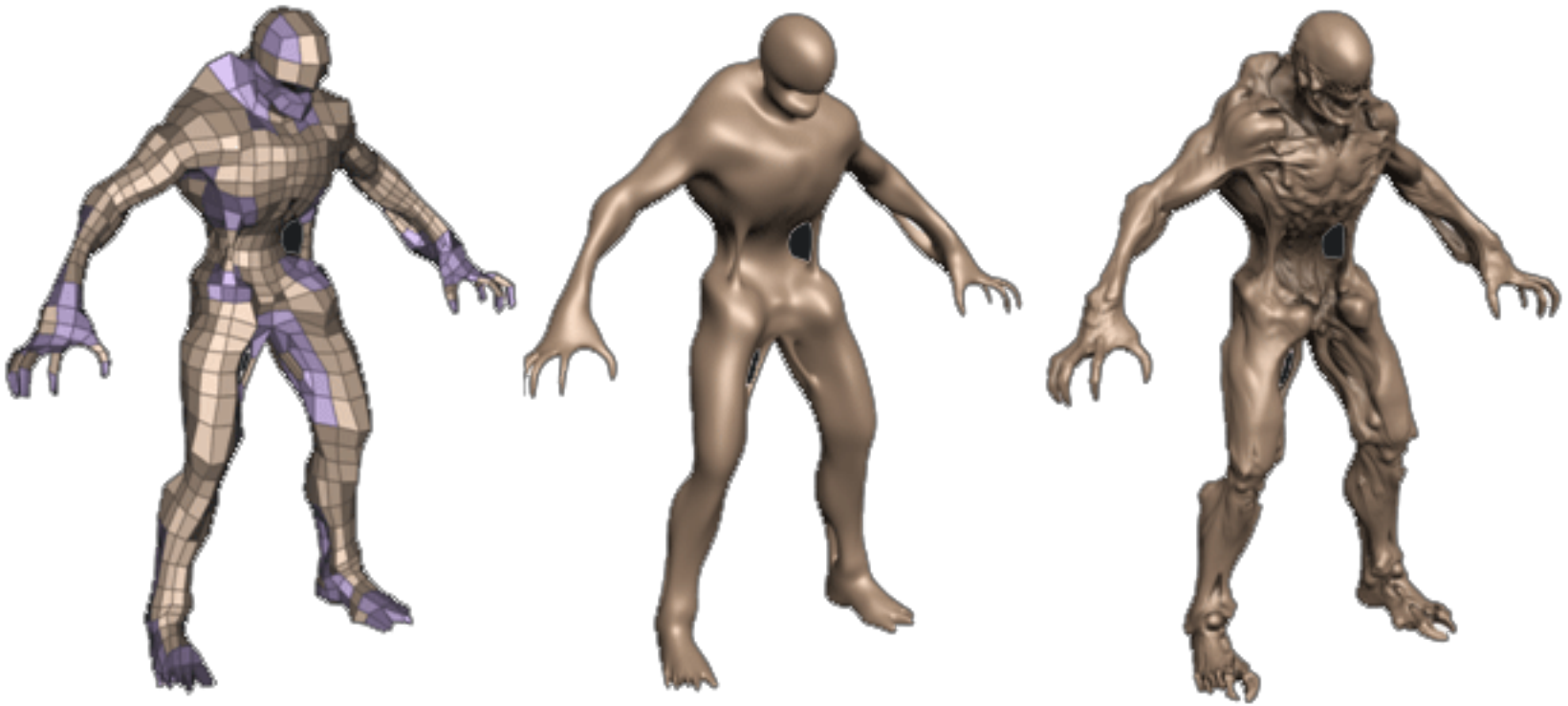
Level of Detail



Level of Detail



Tessellation & Displacement

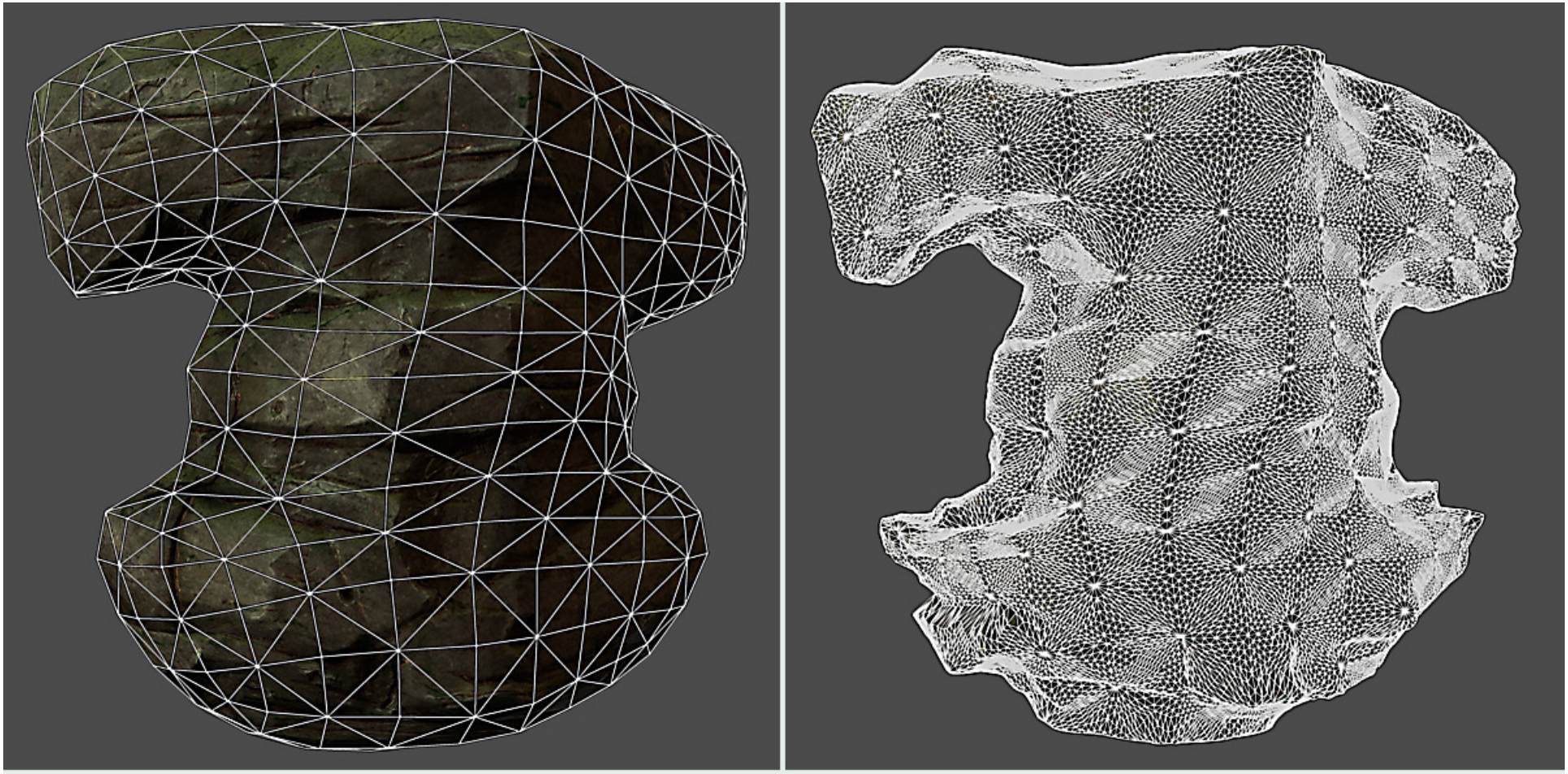


Low Polygon Count Model

Tessellated (More Polygons)

After Displacement

Tessellation & Displacement



Low Poly Model Topology

Low Poly Model Tesselation

Tessellation

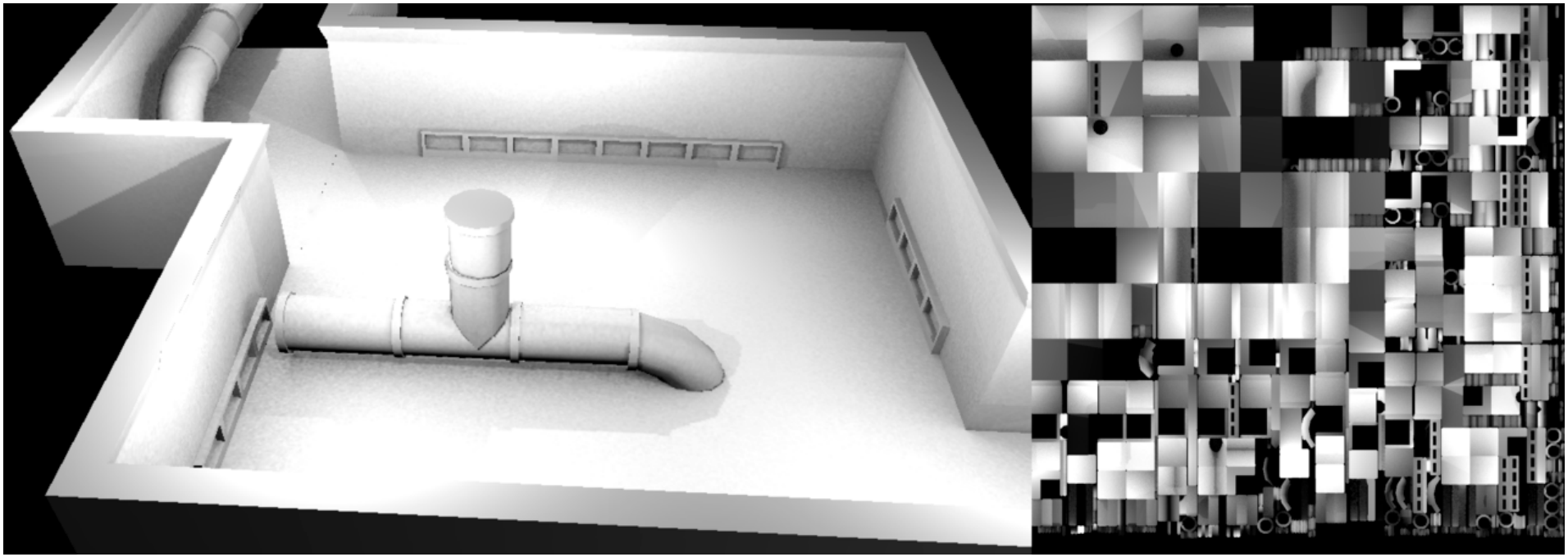
Burned-In Lighting

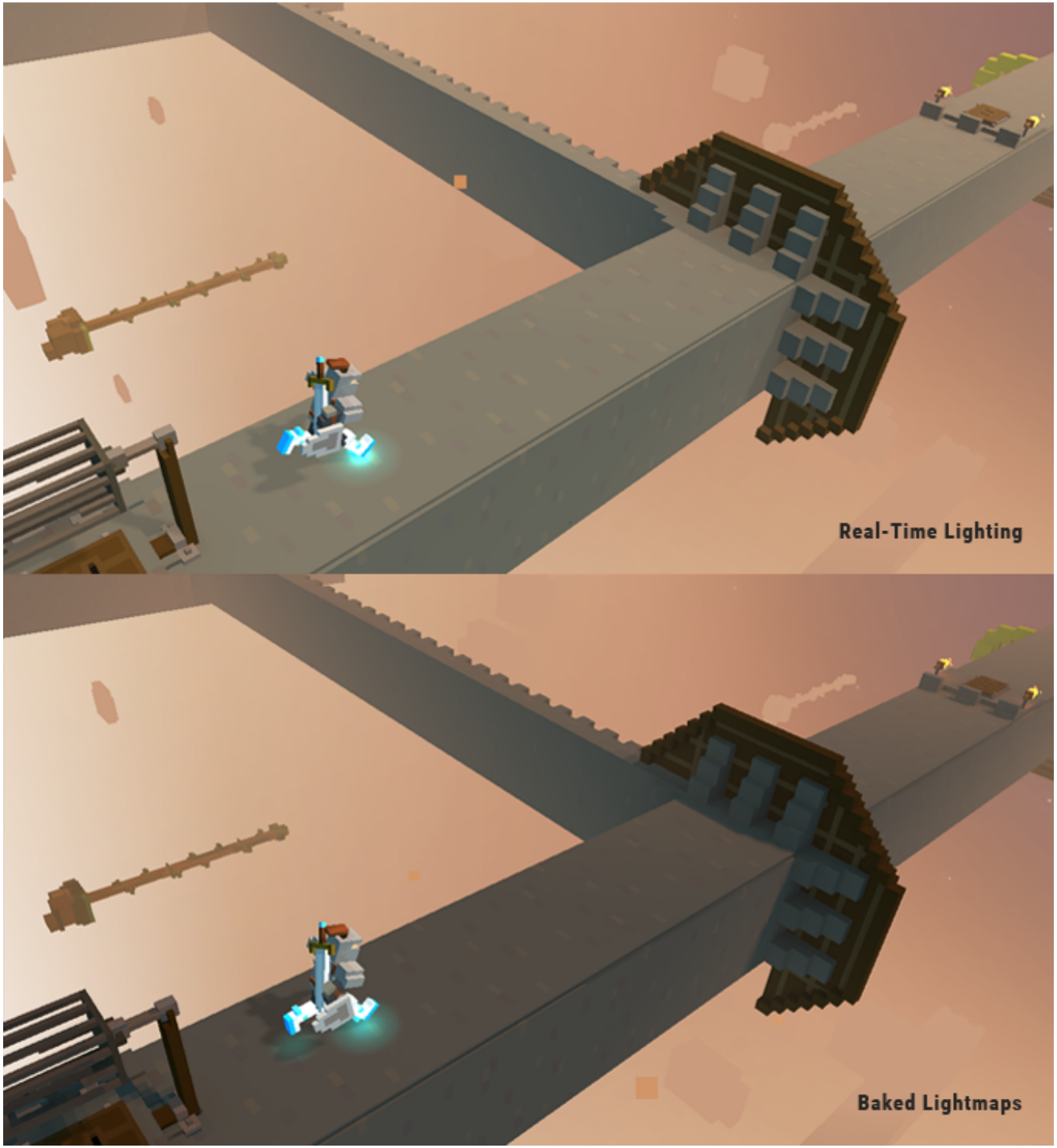
- Calculate expensive lighting off-line
- May include: shadows, indirect illumination
- Create texture map *that includes lighting result*
- Best for Lambertian surfaces (no specular reflection)
- Only works for static objects and lights (no motion)

Burned-In Lighting

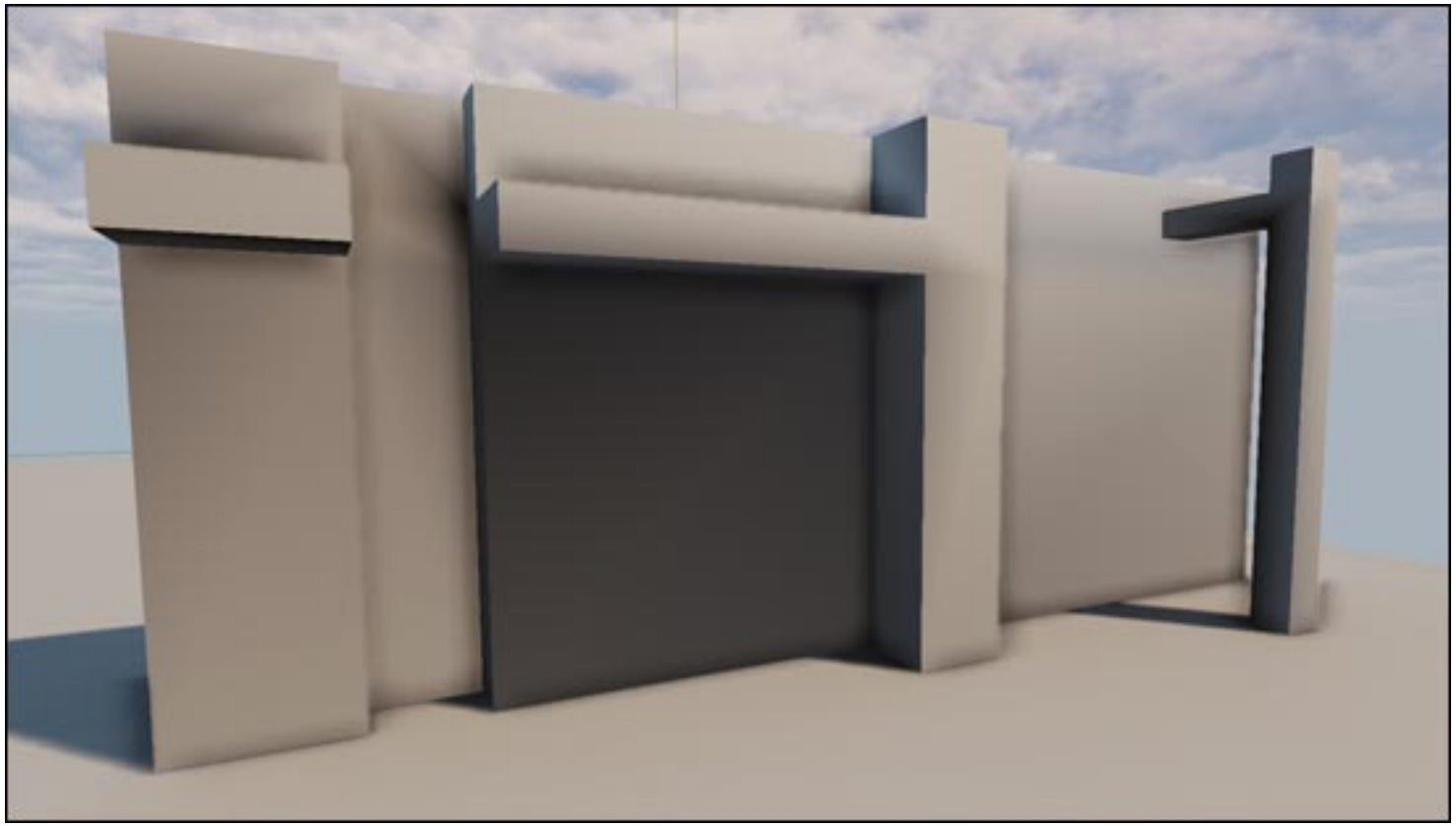
Lit Scene

Lightmap

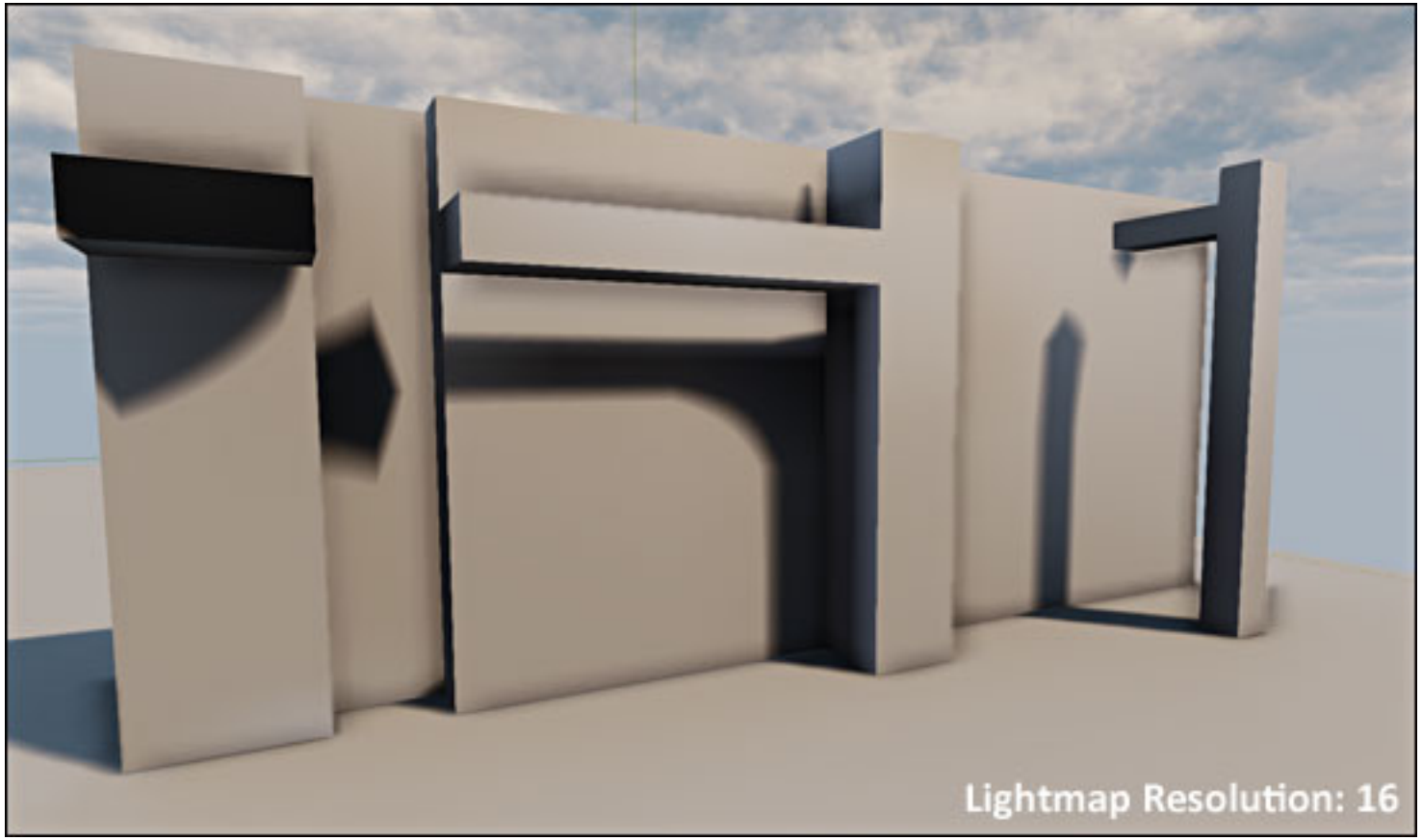



Real-Time Lighting
Baked Lightmaps

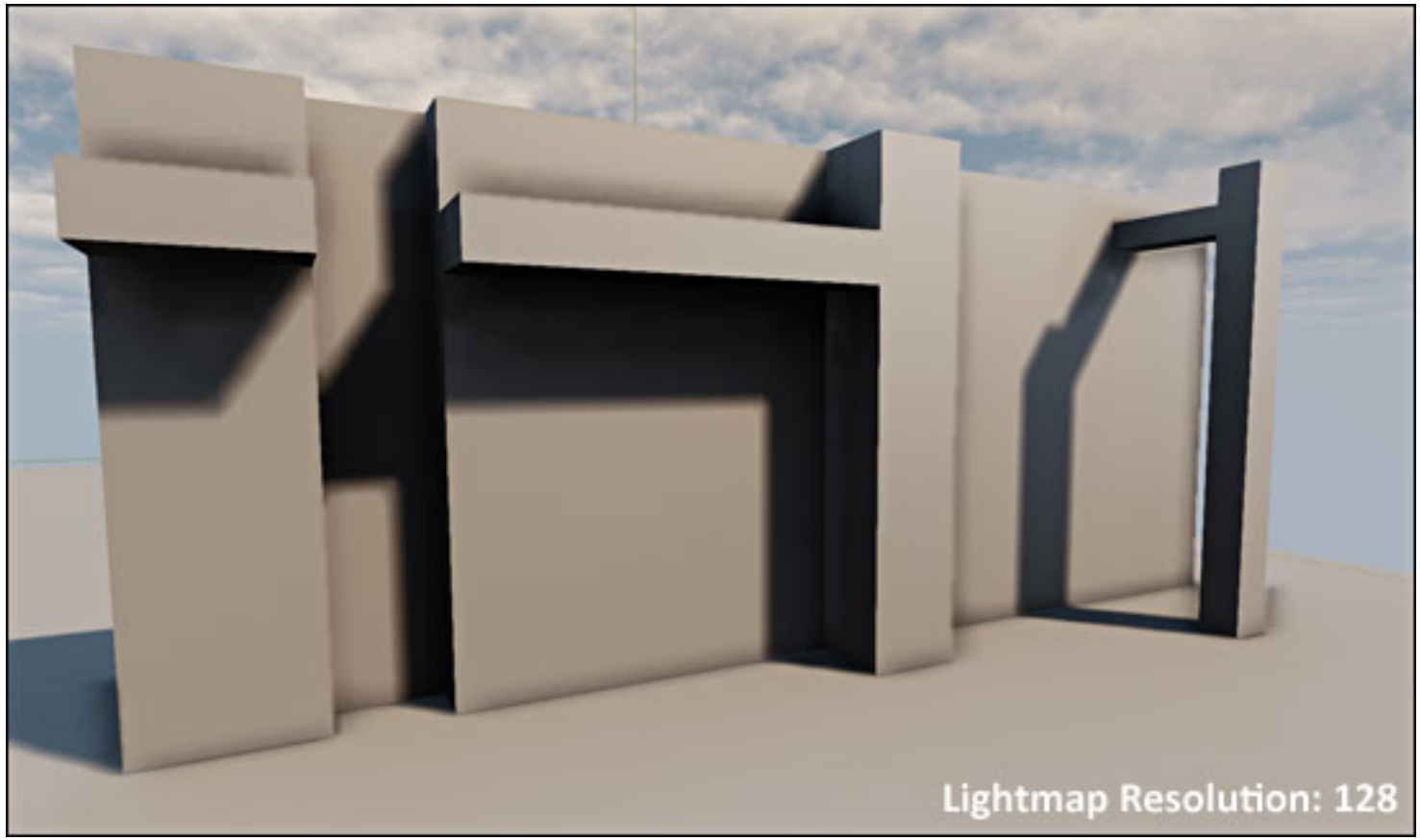
Before Lightmap



Low Resolution Lightmap



High Resolution Lightmap

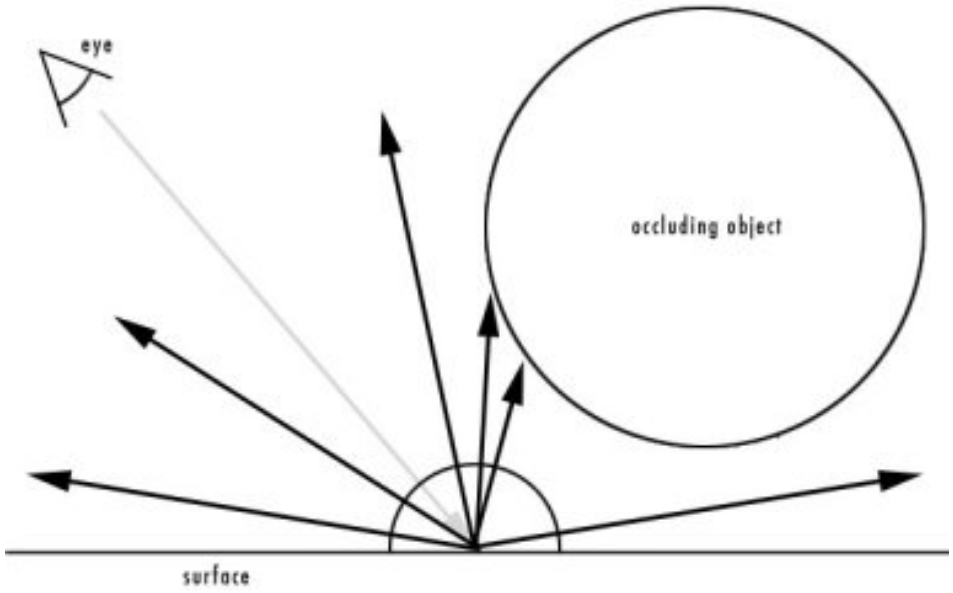


Lightmap and Texture

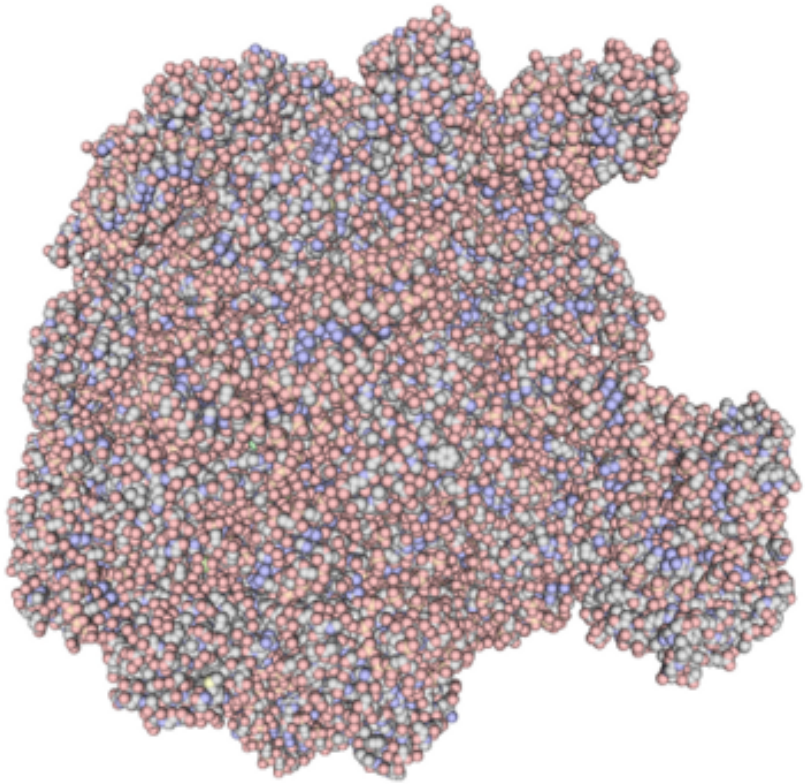
Ambient Occlusion

- Estimate amount of sky visible from point
- Mimics light from cloudy day
- Darkens the creases in objects
- Soft shadows on ground plane

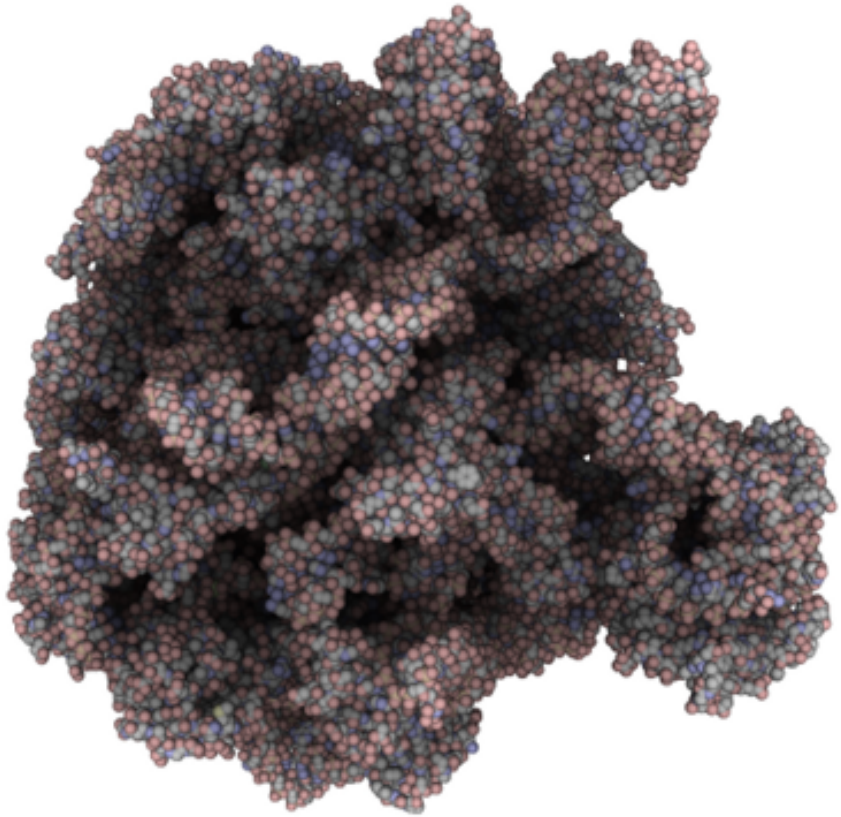
Ambient Occlusion



Ambient Occlusion - Molecules



Uniform Illumination



Ambient Occlusion

Ambient Occlusion - Off-Line

Ambient Occlusion - GPU

Post-Processing

- Calculate image of scene
- Use pixel shaders (on GPU) to modify image *after* rendering
- Often treat original scene as a texture
- Sometimes separate elements into different images; recombine later
- Such images are called G-buffers

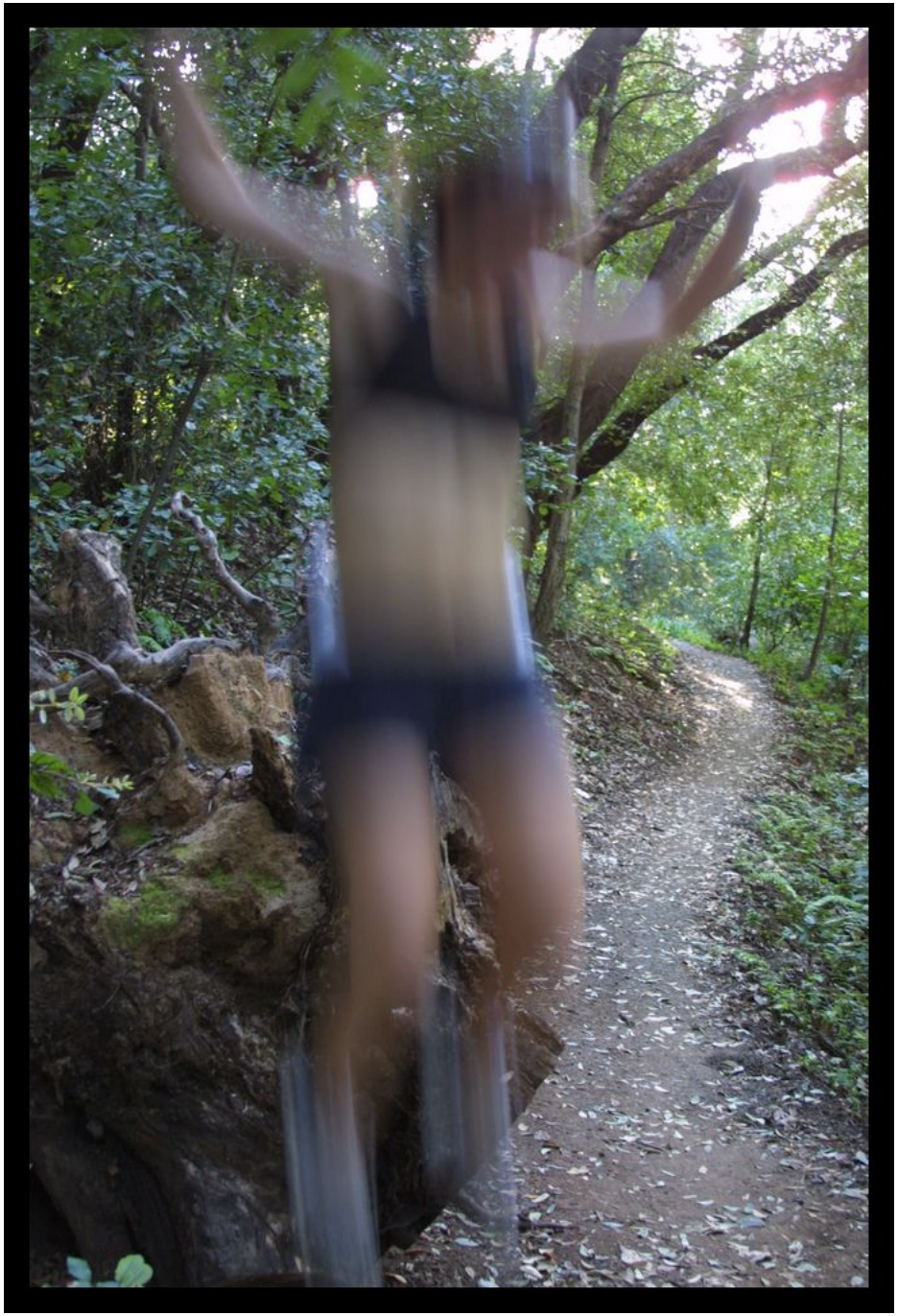
Post-Processing

- Motion Blur
- Depth of Field
- Haze and fog
- Vignette
- Blooming, Neon
- Lens Flare
- High Dynamic Range Images

Motion Blur

- Shutter open for non-zero duration
- Produces streaks in direction of motion
- Also give translucency appearance

Real Motion Blur



Motion Blur - Offset

Motion Blur

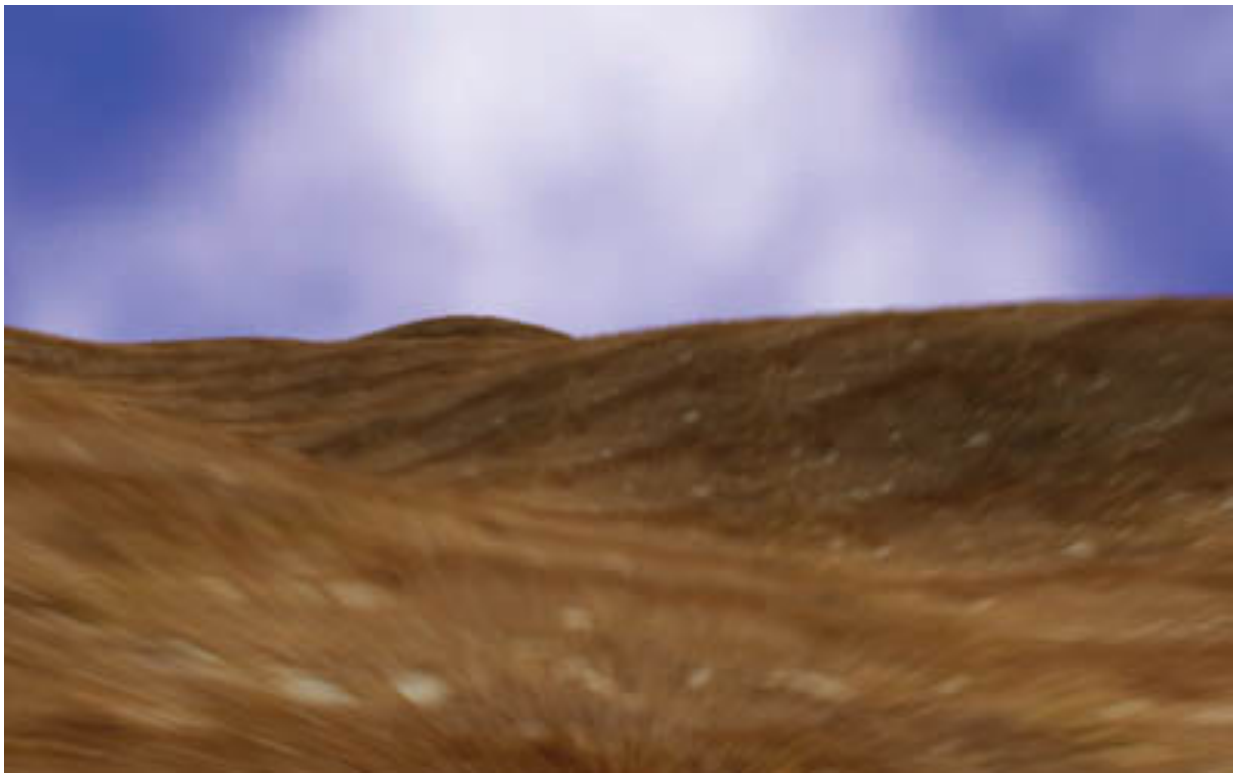
- Save depth map
- Depth + (x,y) gives 3D position
- Use old & new camera position to find how each pixel moves
- Create per-pixel offset vectors
- Blur image along vectors (average multiple texture lookups)

http://http.developer.nvidia.com/GPUGems3/gpugems3_ch27.html

No Blur

Blur

Motion Blur



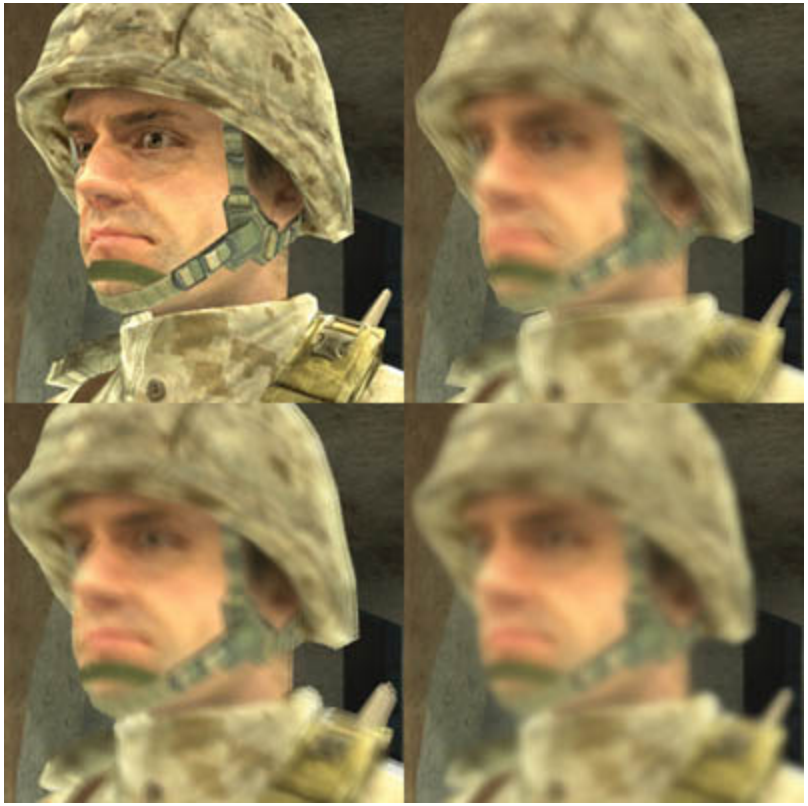
Depth of Field

- Not all objects should be in focus
- In-focus only at focal plane
- Objects in front or behind are blurred

Depth of Field – Post Process

- Render image and save depth map
- Blur regions with depths not at focal plane
- Problems at silhouettes, so…
  - Separate near/far layers
  - Process separately
  - Re-combine

Depth of Field



http://http.developer.nvidia.com/GPUGems3/gpugems3_ch28.html

Depth of Field (Skyrim)

Vignette

- Some cameras darken corners
- Easy to achieve as post process
  - Calculate distance of pixel from center
  - Darken based on distance

Vignette

Vignette

Vignette

Fog and Haze

- Particles in air scatter light
- Light from distance objects scattered more
- Far away objects are washed out
- One of earliest post processing effects
- Post processing:
  - Render image and save depth (z-buffer)
  - Blend with fog color based on depth
  - Blending factor is exponential with distance
- Other method: translucent smoke polygons

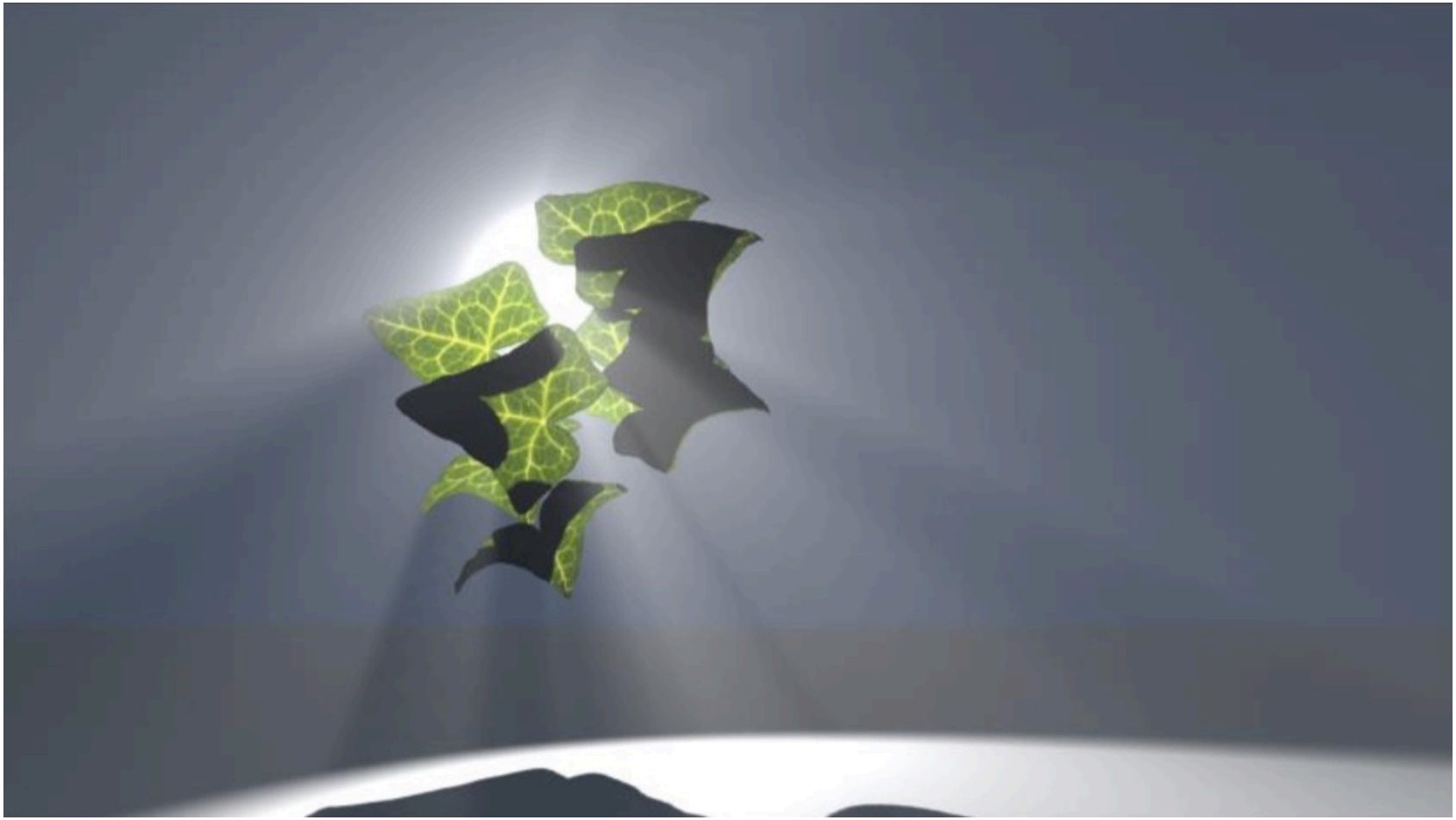
Vietcong - Purple Haze

Vietcong - Purple Haze

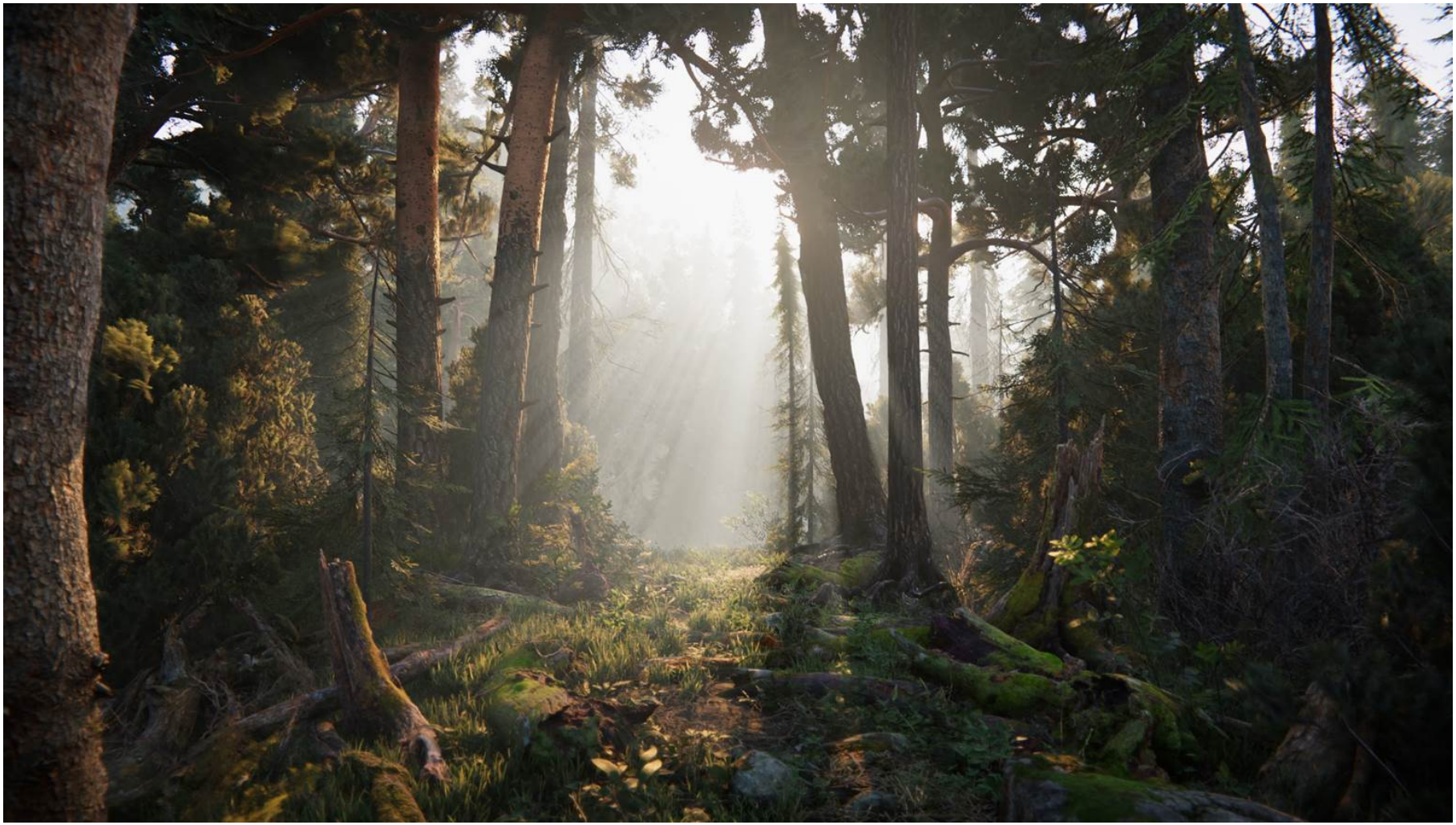
Light Shafts in Haze

- Known as *participating media*
- Add up all the light scattered along paths
- Use voxels to store fog/haze
- Step through voxels, summing light
- Computationally expensive
- Incorporated in Book of the Dead demo

Participating Media



Book of the Dead (demo)



Watch Video

Bloom

- Bright lights “glow” (e.g. neon)
- Color fills region around light
- Post-process:
  - Create image with just lights
  - Blur it like crazy
  - Blend this with original image

No Bloom - Doom 3

Bloom - Doom 3

Lots of Bloom - Doom 3

Lens Flare

- “Echos” of bright lights
- Due to light interacting with lens elements
- Often strands of flares, different sizes
- Can include color shifts (chromatic aberration)

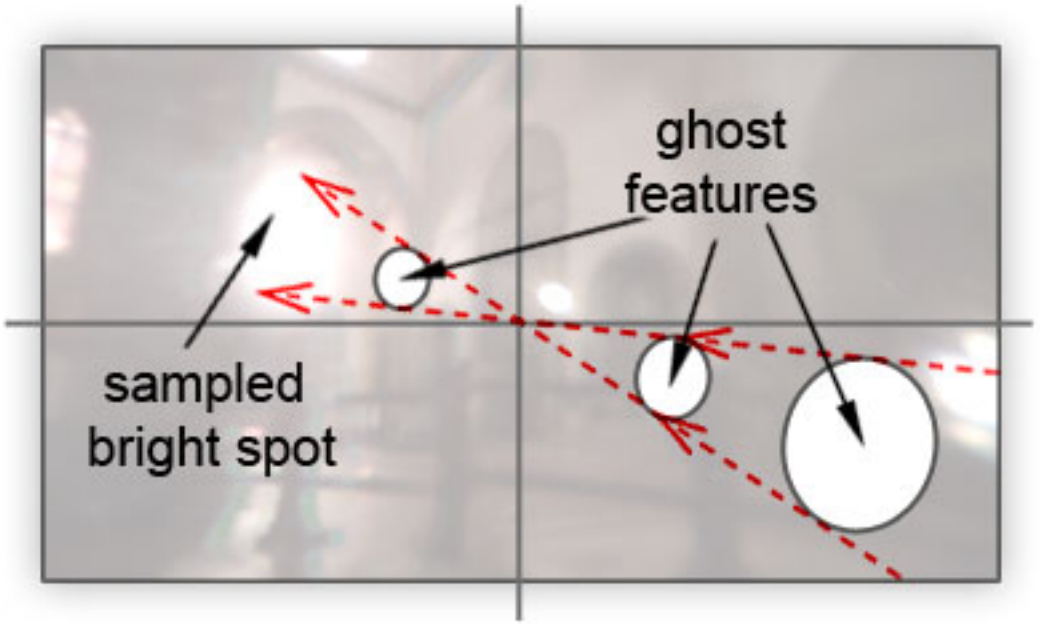
Lens Flare

Lens Flare as Post Process

- Identify bright pixels near screen center
- Create pixel copies radially
- Shift colors
- Blur
- Add back into original image

John Chapman Tutorial:
john-chapman-graphics.blogspot.co.uk/2013/02/pseudo-lens-flare.html

Lens Flare



Lens Flare

Lens Flare

Lens Flare

pre blur

post blur

Lens Flare (Battlefield 3)

Real-Time GPU Ray Tracing

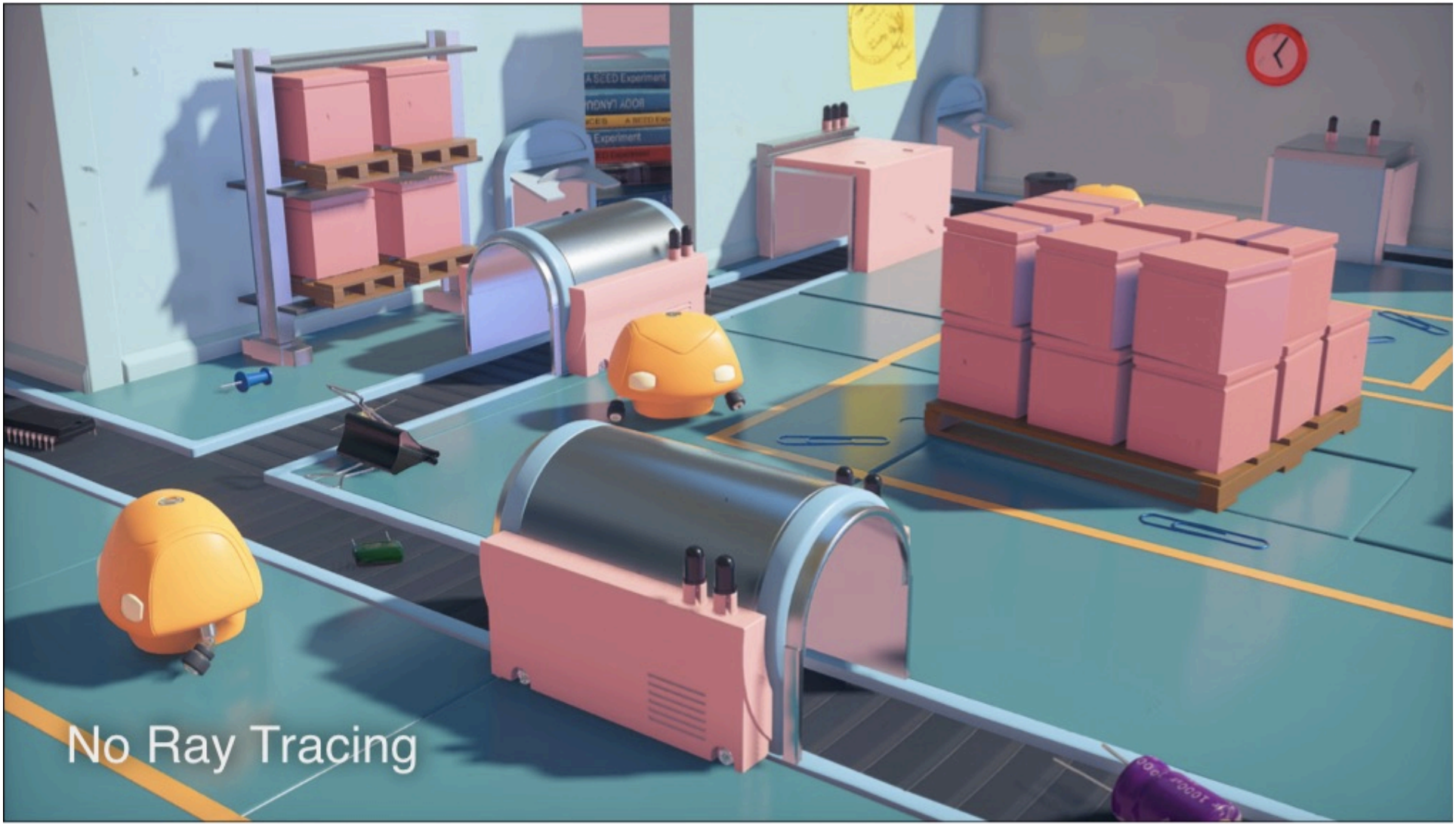
- New Nvidia chips (GeForce RTX 2060)
- Ray tracing built into hardware
- Incorporated into DirectX 12 API
- Starting to be used in games

Ray Tracing Effects

- Indirect illumination (multiple-bounce light)
- Correct reflections
- Better depth-of-field
- Faster ambient occlusion
- Still much slower than rasterization


Epic Games / ILMxLAB


EA SEED Project PICA PICA


No Ray Tracing


Real-Time Hybrid Ray Tracing

Watch Videos

More on Game Rendering

SIGGRAPH 2018 course on game rendering:

http://advances.realtimerendering.com/s2018/index.htm

End