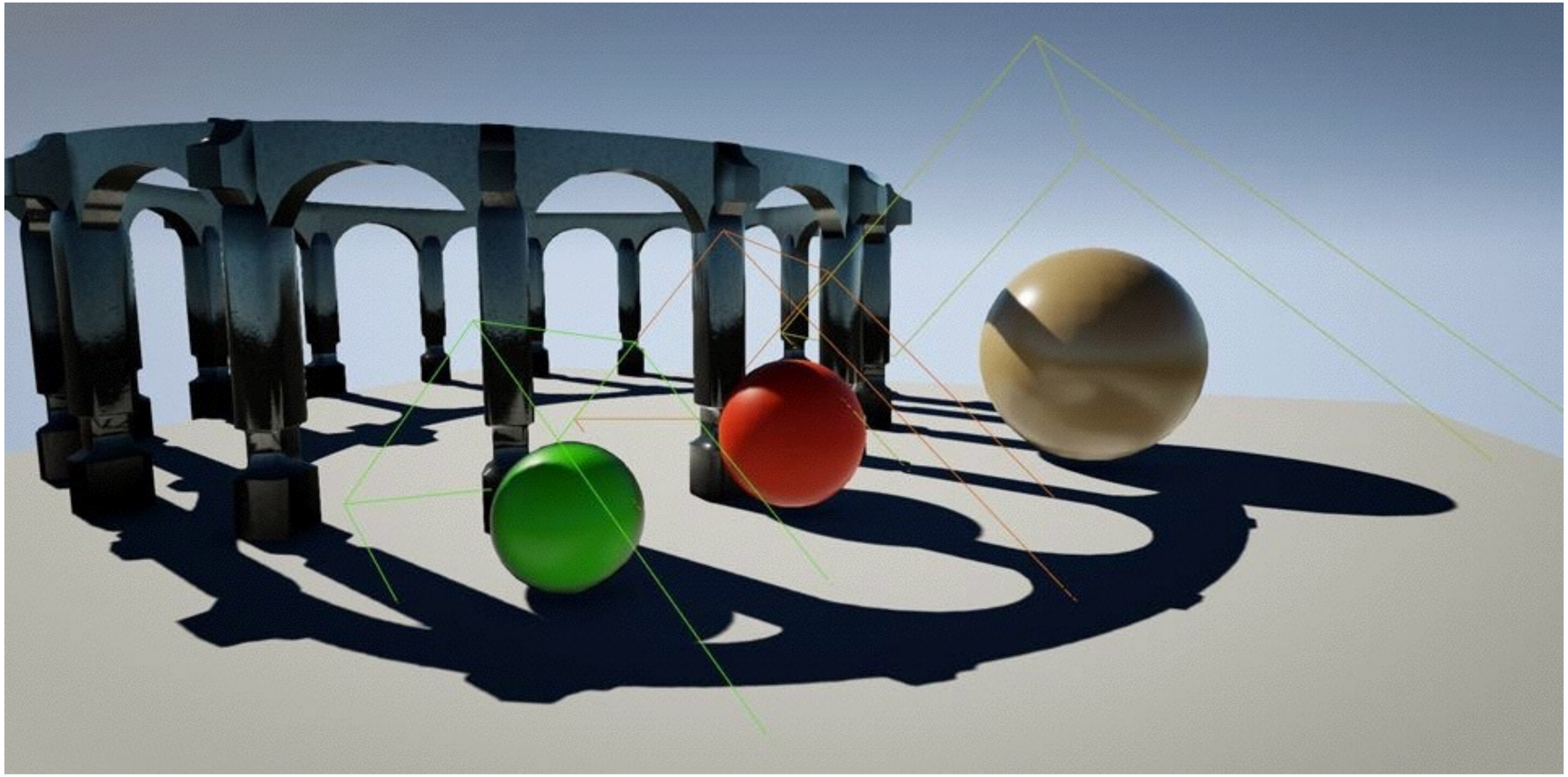
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Practical 9

Intro to Computer Graphics

Shadows

Shadows

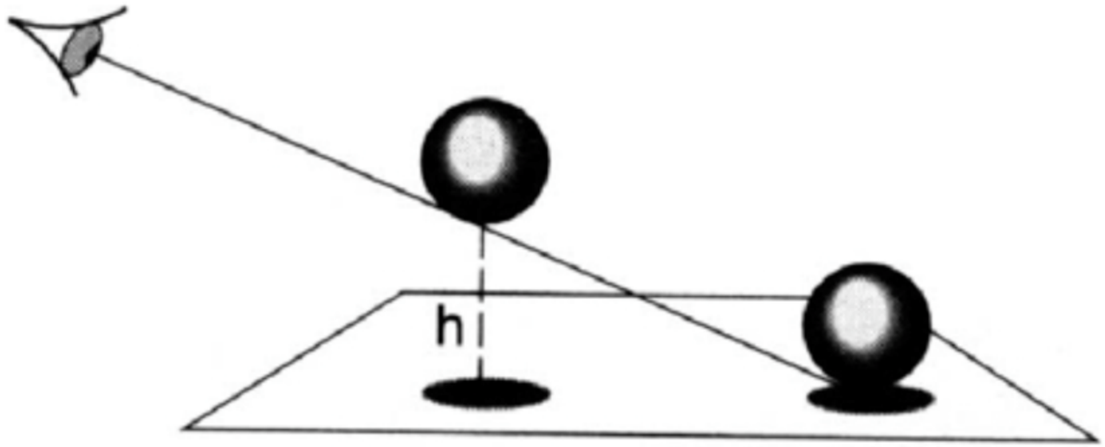


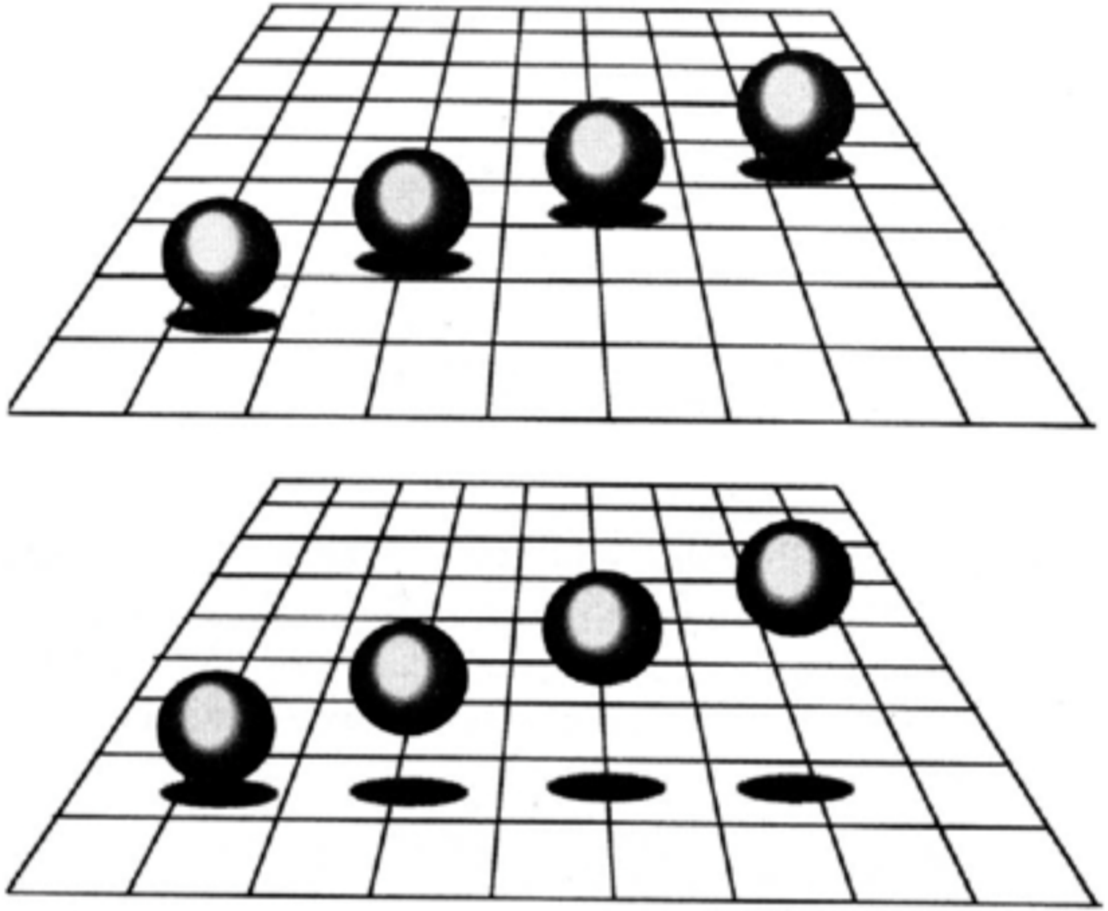
Shadows

- Why
  - Fancy/cool effect
  - Realism
  - Depth Perception

Depth Perception

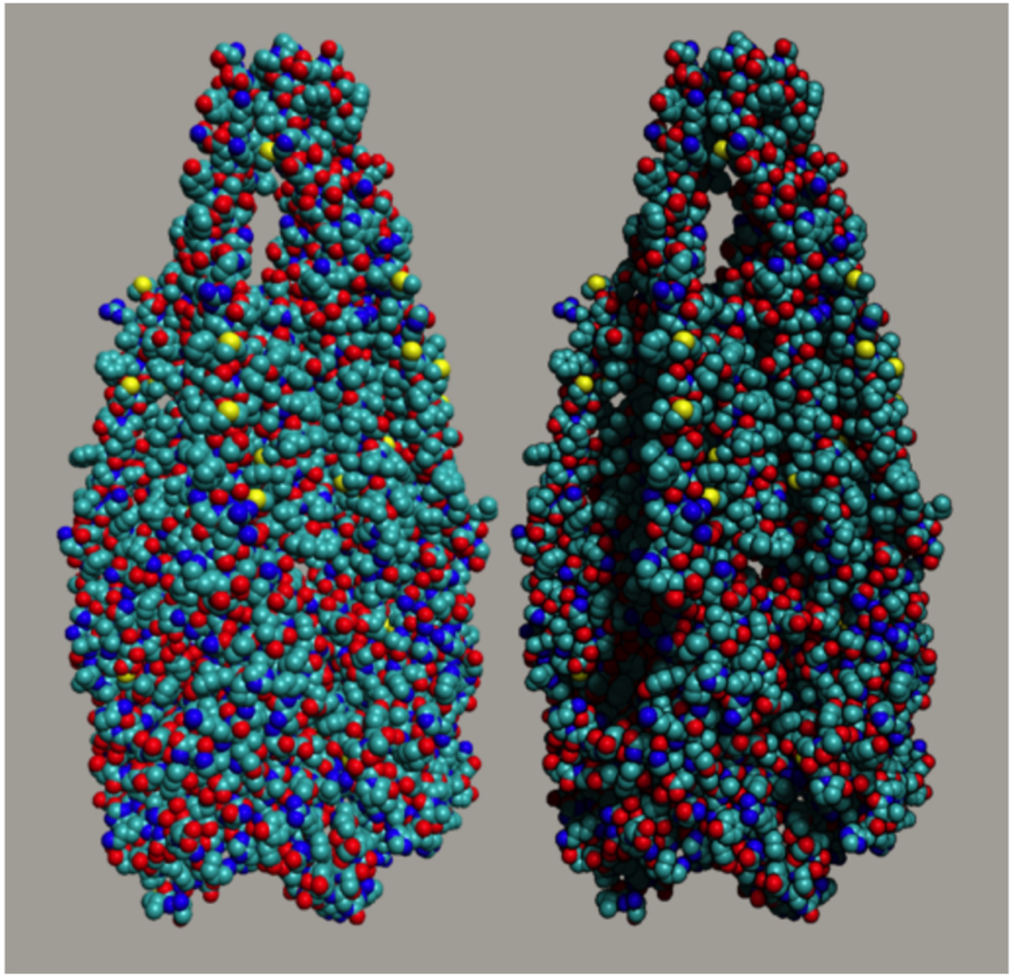
- Shadows are important for 3D depth perception





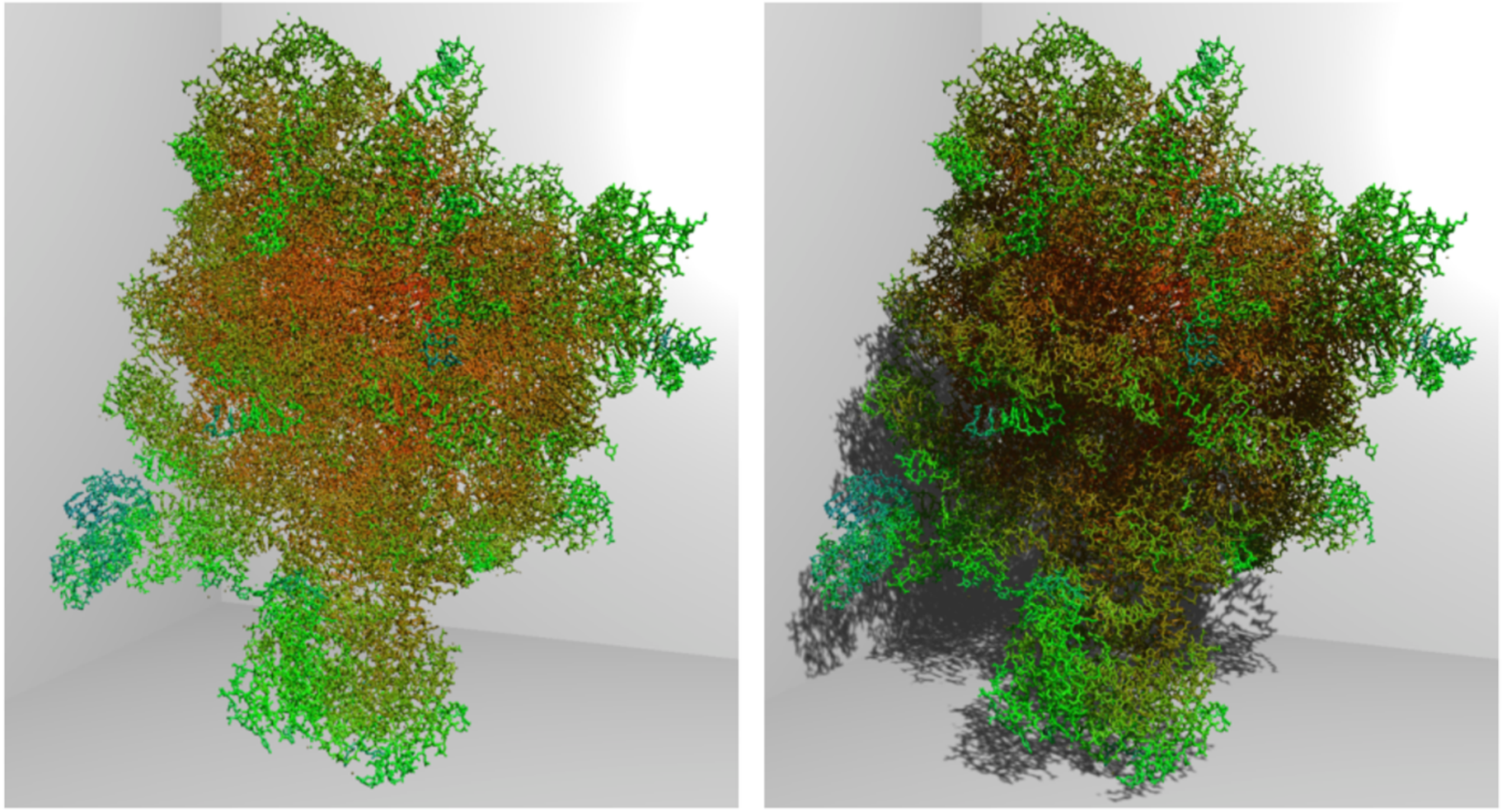
Depth Perception

- Shadows are important for 3D depth perception



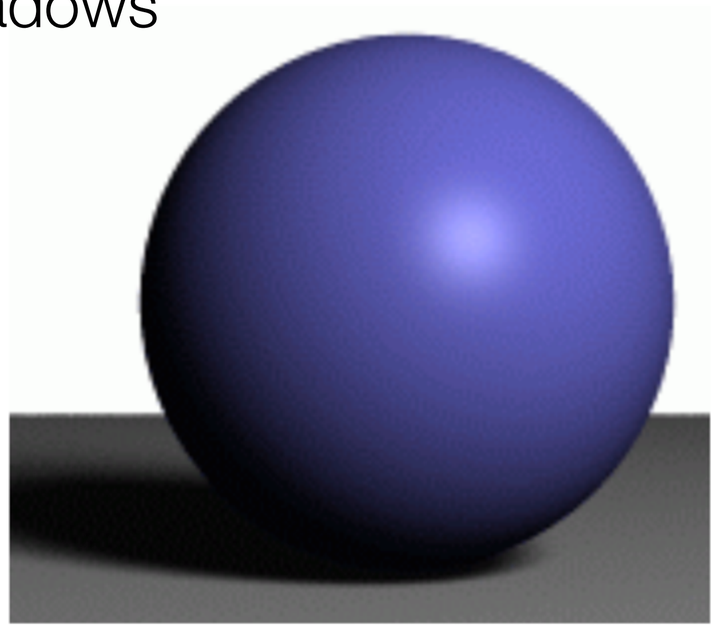
Depth Perception

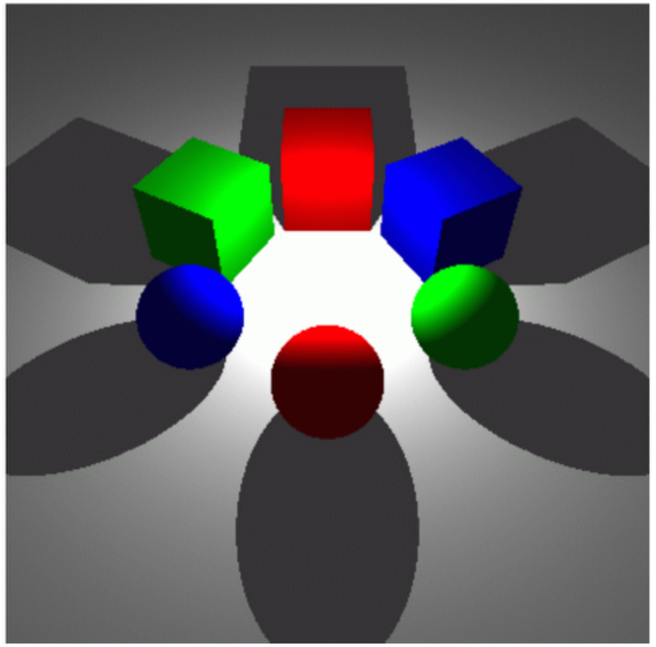
- Shadows are important for 3D depth perception



Shadows and Lighting

- Shadows are important for intuition about scene lighting
  - position of the light (e.g. sundial)
  - hard shadows vs. soft shadows (point light vs. area light)
  - directional vs. point light





Shadow Computation

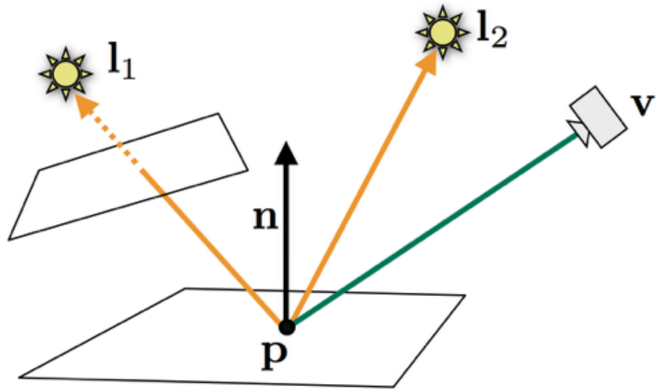
- Visibility:
  - Which objects can be seen from the viewpoint?
- Shadows:
  - Which objects can be seen from the light source?

Shadow Computation

- If a light source is occluded, simply skip its contribution to lighting computation

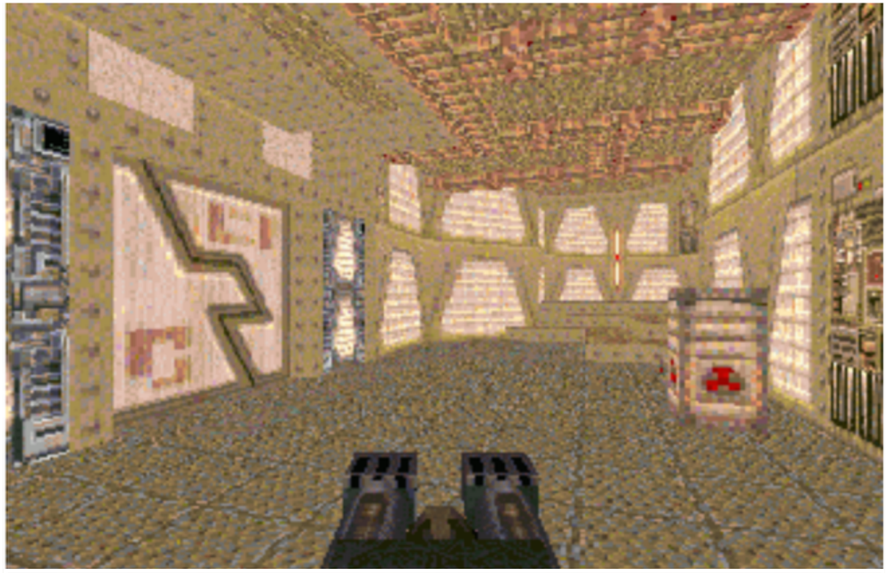
$$I(\mathbf{p},\mathbf{n},\mathbf{v}) = I_{\text{ambient}} + \sum_{\mathbf{l}_i} \text{shadow}(\mathbf{p},\mathbf{l}_i) \cdot [I_{\text{diffuse}}(\mathbf{p},\mathbf{n},\mathbf{l}_i) + I_{\text{specular}}(\mathbf{p},\mathbf{n},\mathbf{v},\mathbf{l}_i)]$$

$$\text{shadow}(\mathbf{p},\mathbf{l}_i) = \begin{cases} 1 & \mathbf{l}_i \text{ is visible from } \mathbf{p}, \\ 0 & \mathbf{l}_i \text{ is blocked from } \mathbf{p} \end{cases}$$

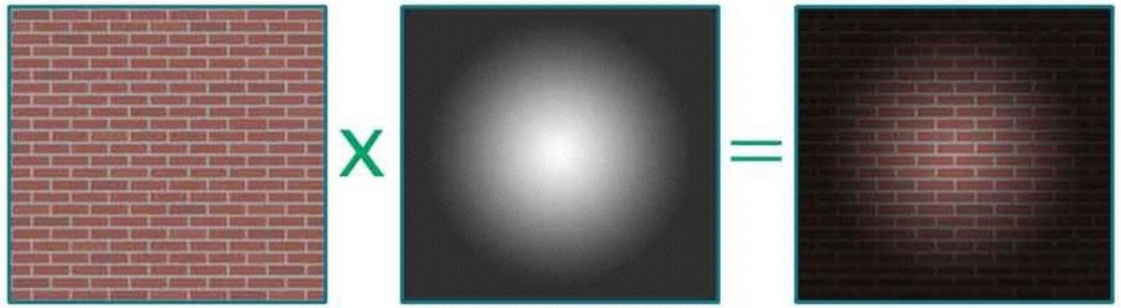


Light Maps

- Store precomputed lighting in textures
  - Precomputation can take shadows into account
  - Precomputation usually done by raytracing
  - Works for **static** scenes only
- Old school
  - but still in use



Light Maps



Dynamic Shadows

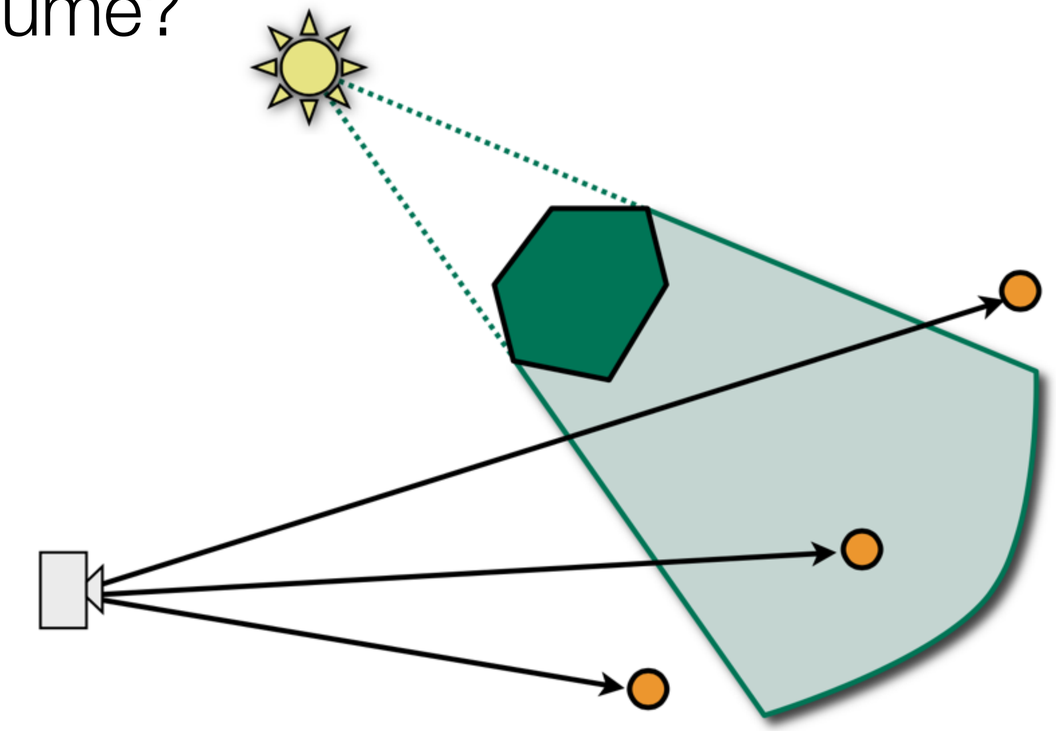
- Shadows are a global effect
  - Occluders block light from receiver
  - Occluders and lights can move/deform dynamically
- How to achieve **global** effects with **local** rendering pipeline?
  - Have to use multiple render passes

Dynamic Shadows

- Shadow Volumes
  - outdated
- Shadow Maps
  - state of the art

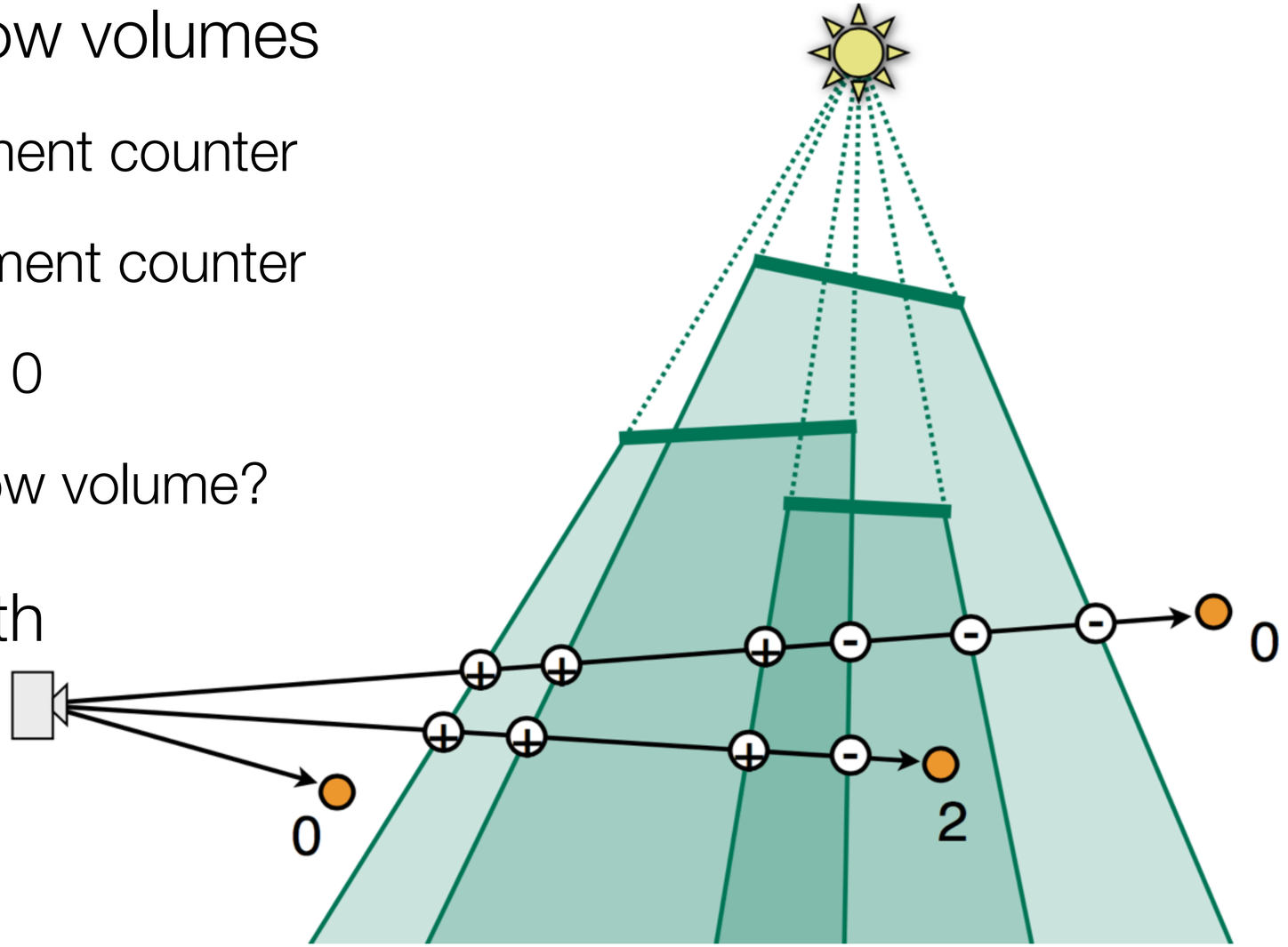
Shadow Volumes

- Polygonal representation of shadowed regions
  - Cone-like polyhedral volume
- Test: Point inside/outside shadow volume?
  - Count intersections of viewing ray



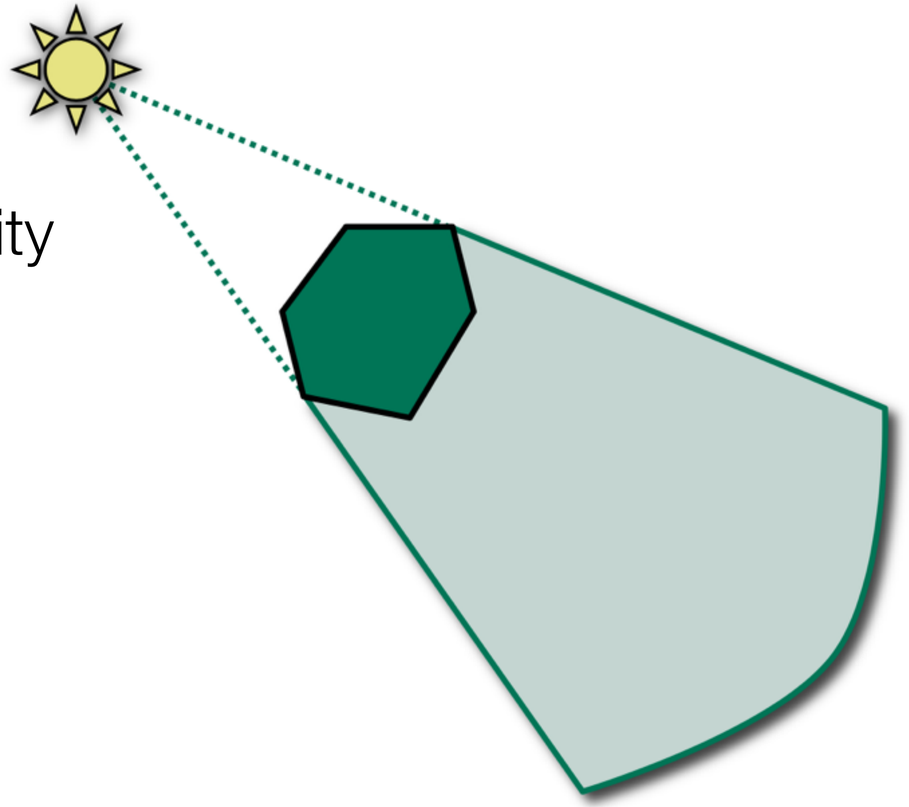
Shadow Volumes

- Several overlapping shadow volumes
  - Entering intersections increment counter
  - Leaving intersections decrement counter
  - Point in shadow if counter ≠ 0
  - What if camera inside shadow volume?
- Can be done efficiently with stencil buffer

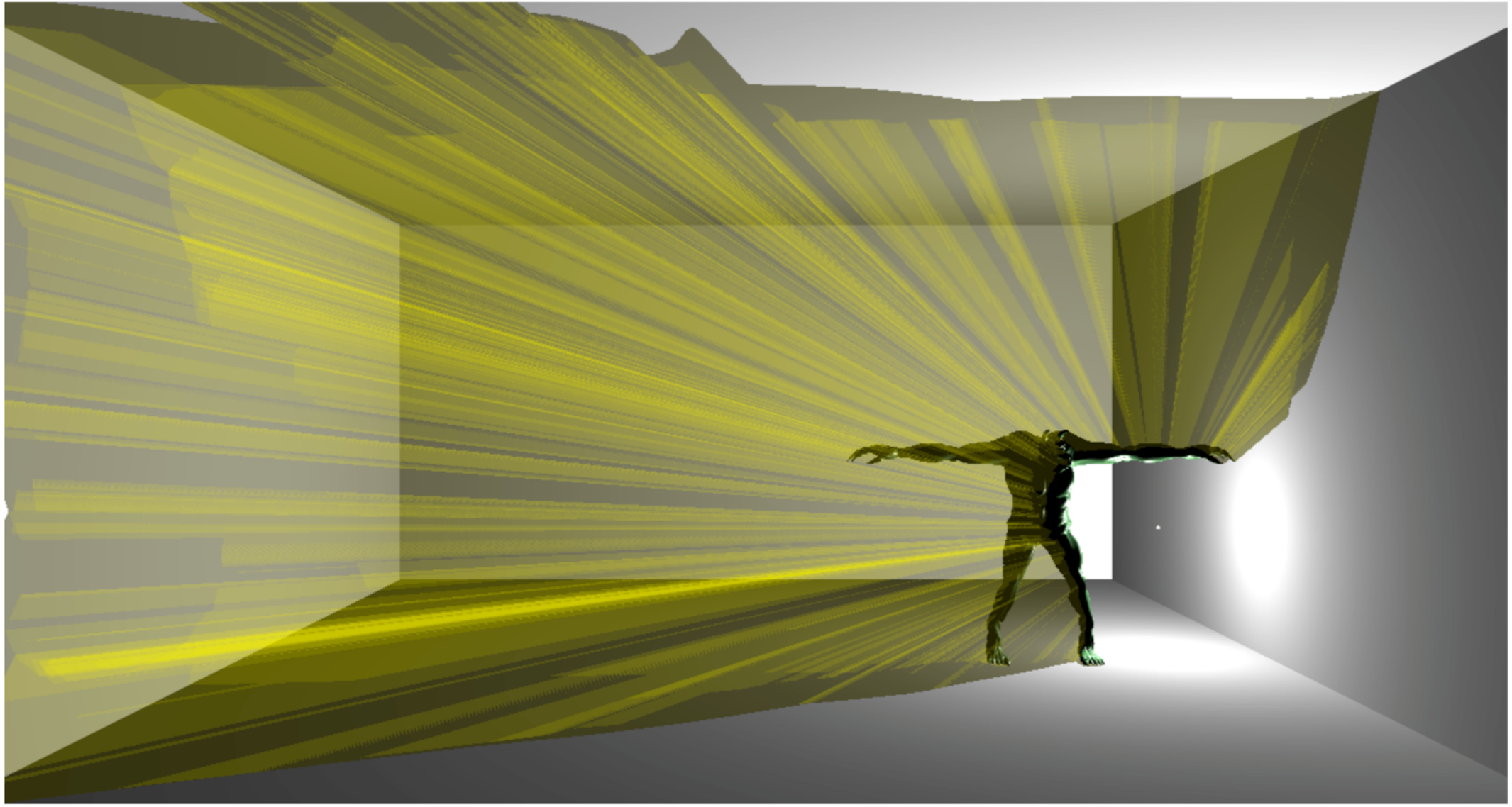


Shadow Volumes

- How to generate an object's shadow volume?
  - Each silhouette edge spans an infinite quad
  - Front-facing triangles build front-cap
  - Back-facing triangles extend to back-cap at infinity
- Silhouette edges
  - Incident to a front- and back-facing triangle



Shadow Volumes



Shadow Volumes

Dynamic Shadows

- Shadow Volumes
  - outdated
- **Shadow Maps**
  - state of the art

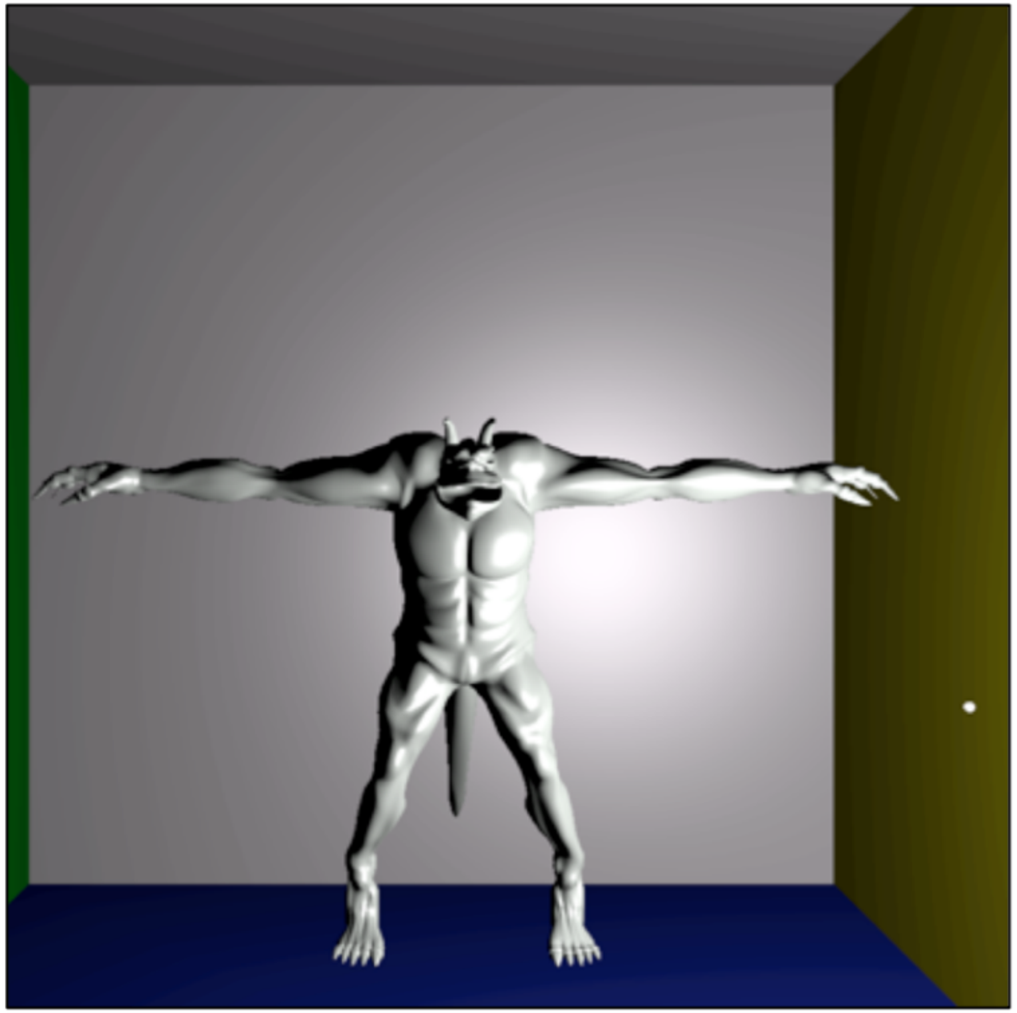
Shadow Computation

- Visibility:
  - Which objects can be seen from the viewpoint?
- Shadows:
  - Which objects can be seen from the light source?
- **Apply visibility techniques for shadows**
  - **z-buffer→shadow map**

Shadow Maps

- Render scene as seen from light source
  - Don’t render any colors
  - Store z-buffer as texture (holds distance to light)
  - Light’s z-buffer is called **shadow map**

Shadow Maps



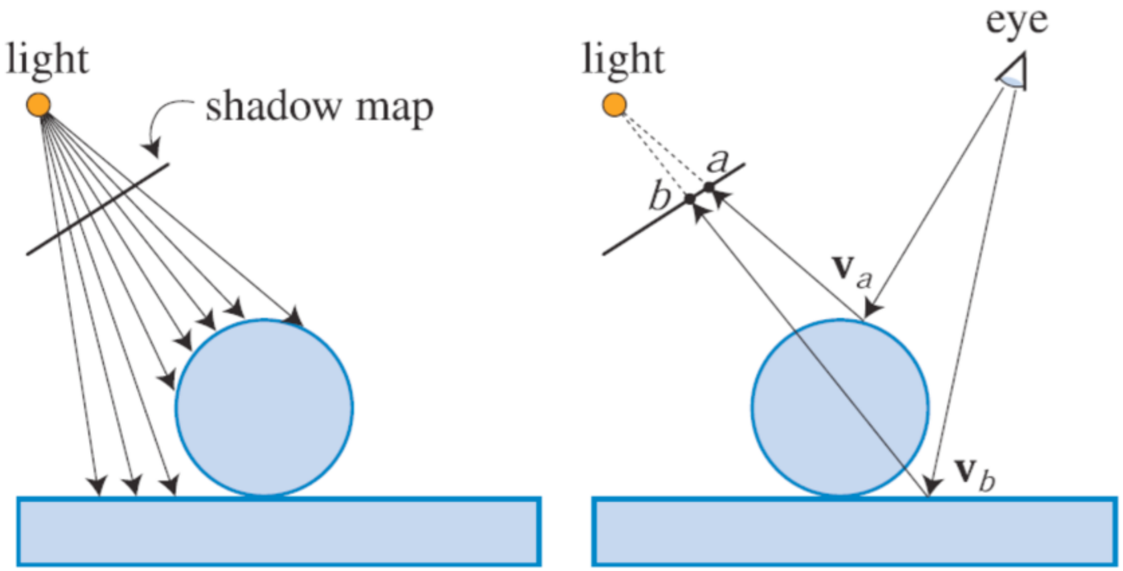
Scene rendered from eye point

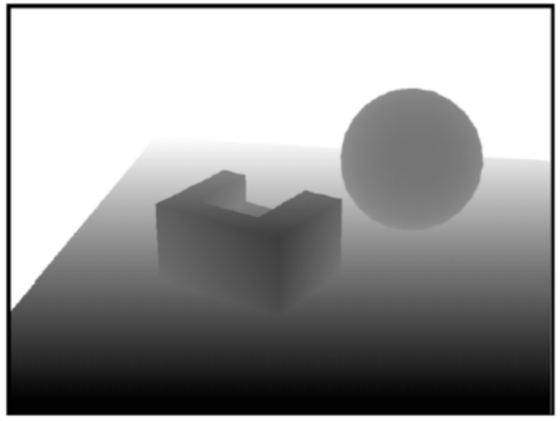
z-buffer from light source

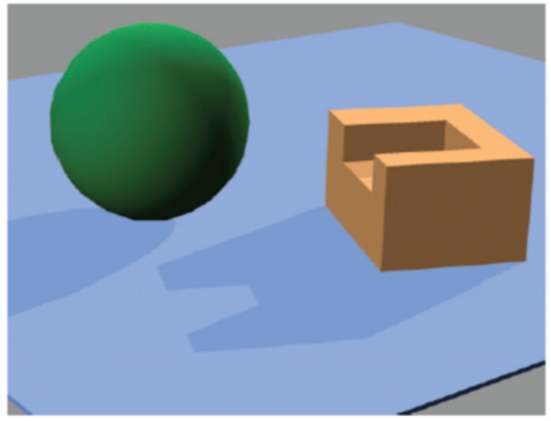
Shadow Maps

- Render scene as seen from light source
  - Store z-buffer (holds distance to light)
  - Light's z-buffer is called **shadow map**
- Render scene from eye point
  - Light a certain image plane pixel (x,y)?
  - Map it back into world coordinates: (x',y',z')
  - Project it into shadow map (aka transform it into light's perspective): (x", y")
  - Distance point-to-light > depth stored in map → Point is in shadow

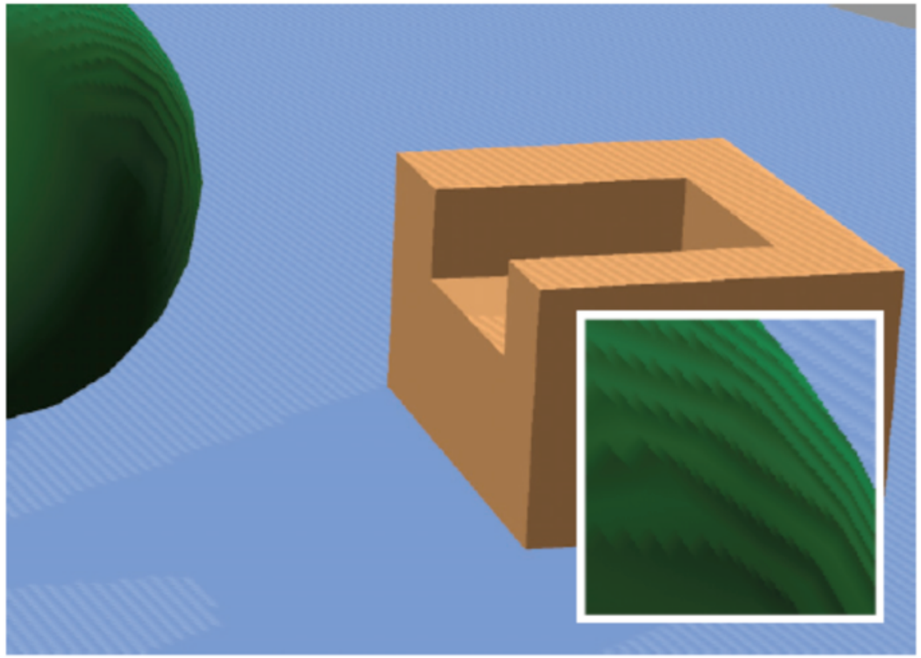
Shadow Maps

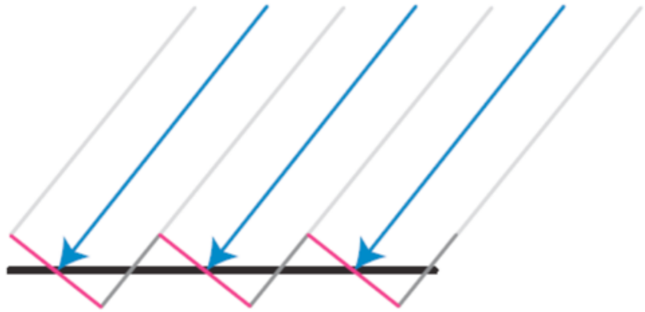




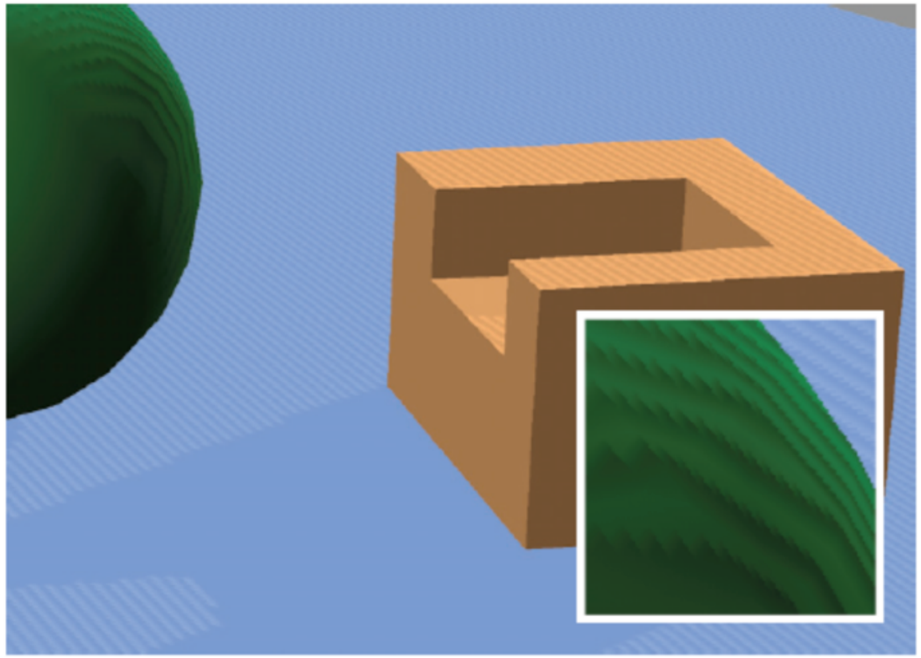


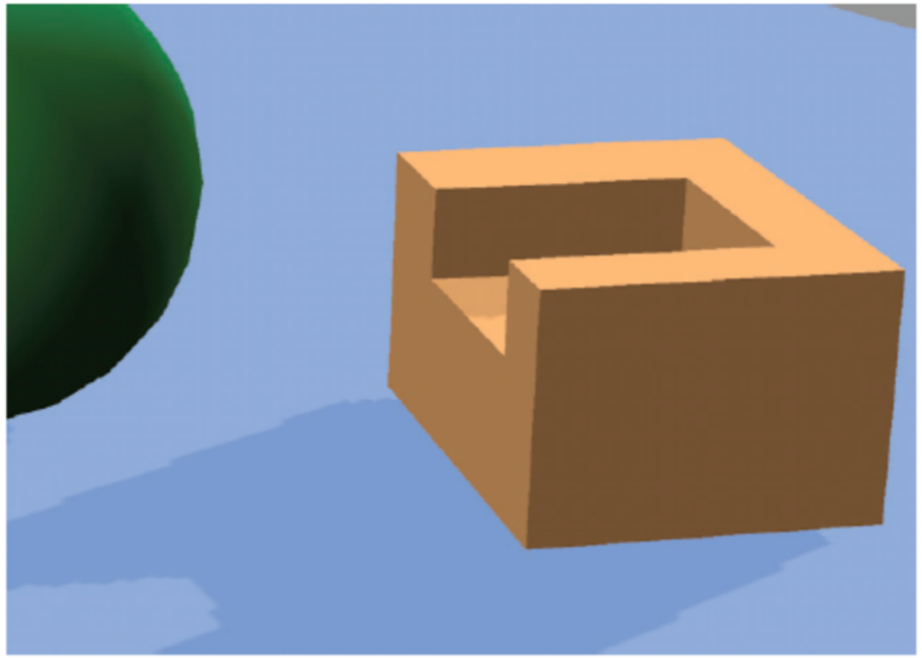
Shadow Acne

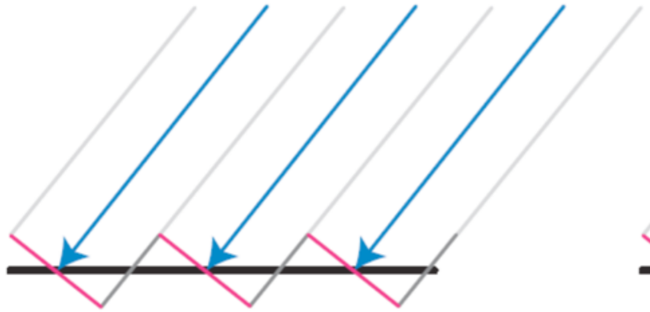


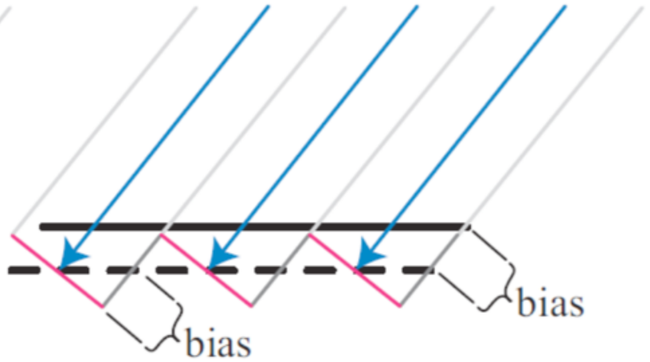


Shadow Acne

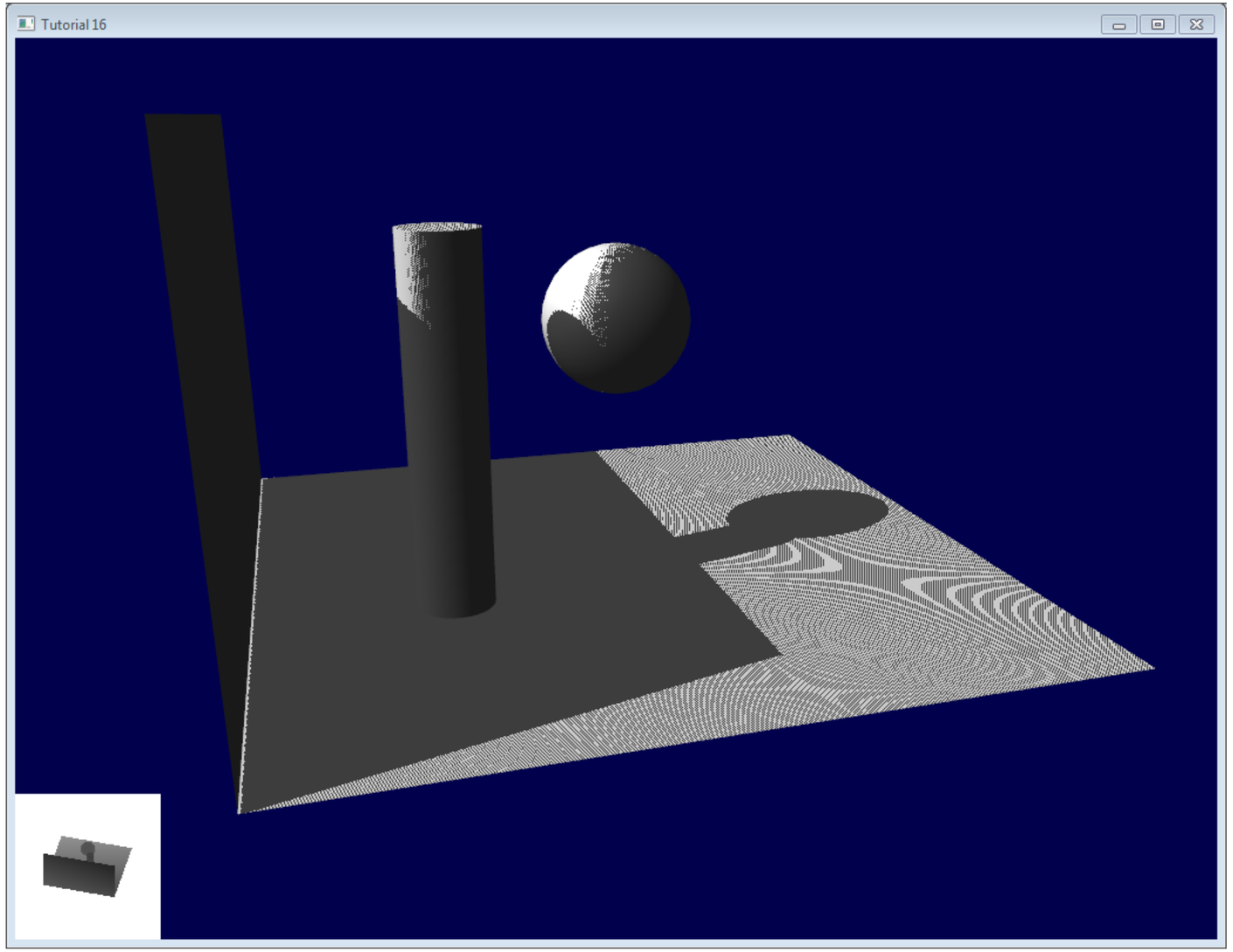




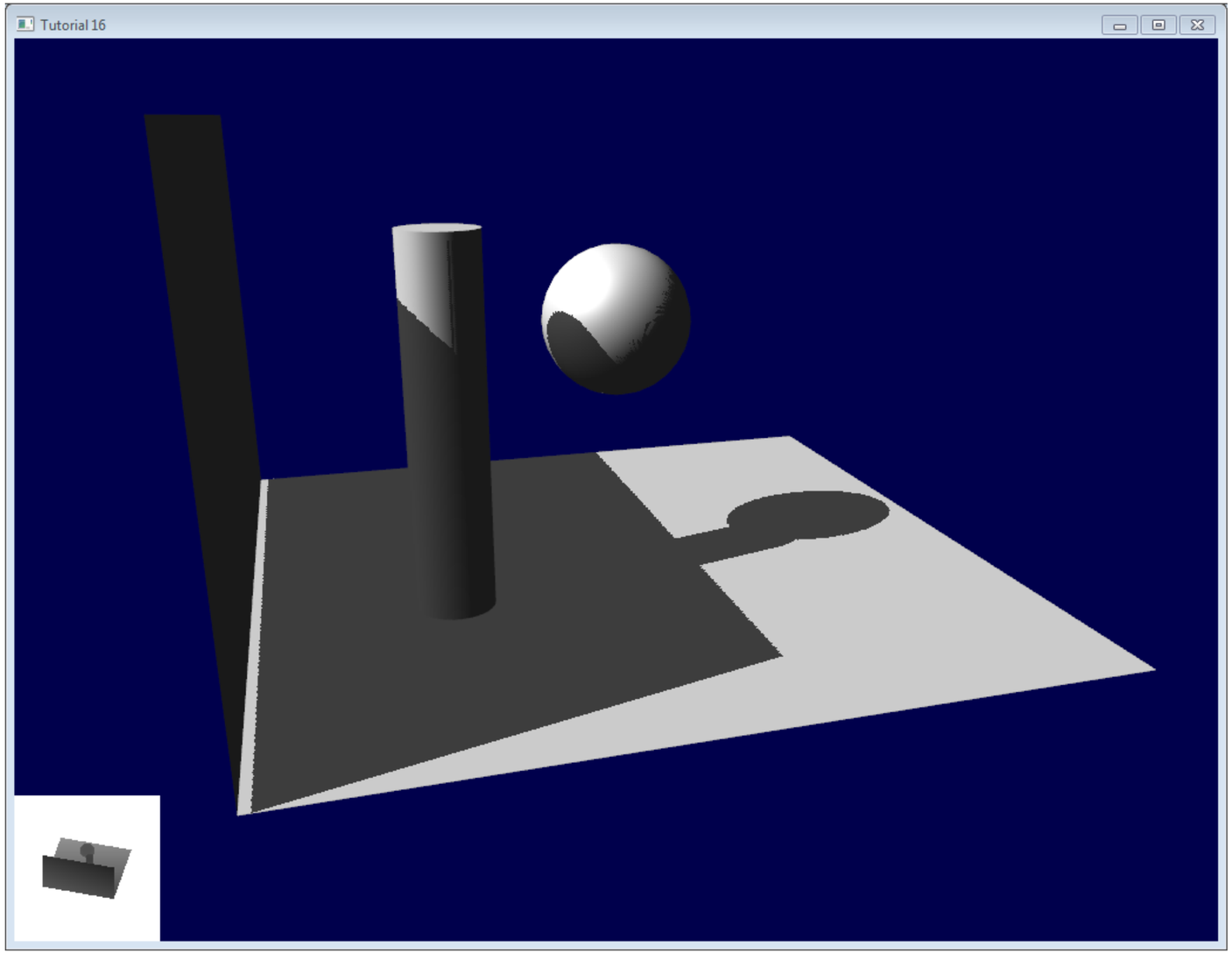




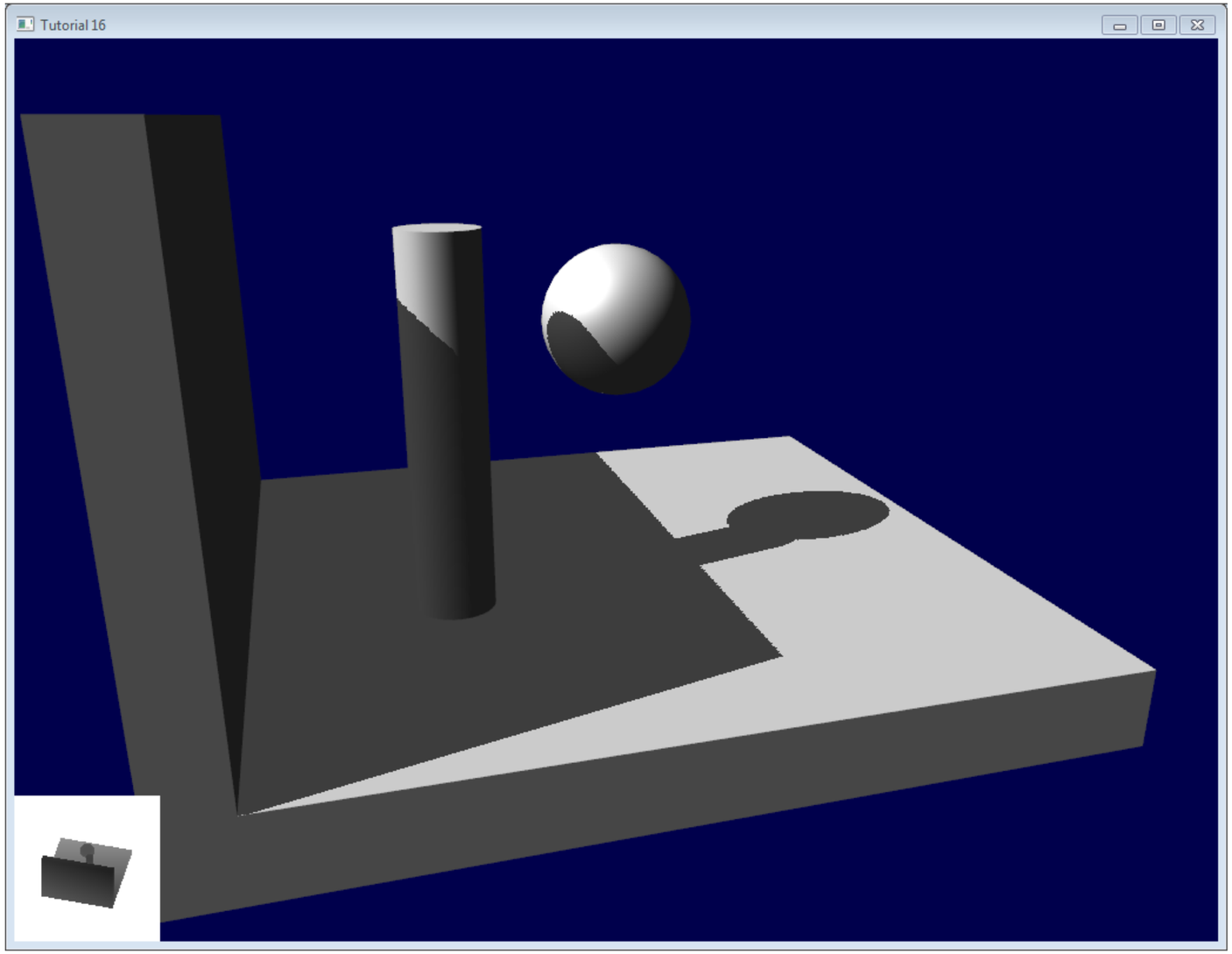
Peter Panning



Peter Panning



Peter Panning



OpenGL Implementation

- Shadow mapping tutorial, read it!
  http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/
- Several render passes
  - 1 for each light
  - Final accumulation and shading render pass

OpenGL Implementation

- Render scene w/o light contribution: for each light source
  - Render scene from light source
  - Store z-buffer in shadow map
- Render scene with light contribution (accumulation)
  - Shadow map lookup for each pixel
  - Pixels in shadow are discarded or darkened
    - Other pixels are lit and rendered

Details Shadow Map Render Pass

- For each light
  1. Generate and bind an FBO with only a **depth attachment**, no color attachments
  2. Disable writing colors (`glDrawBuffer(GL_NONE)`)
  3. Use texture as depth attachment (= shadow map)
  4. Set up a light projection matrix (ortho for directional, perspective for point lights)
  5. Set up light view matrix to transform vertices into light's coordinate system
  6. Set up shader program (vertex shaders does mvp transformation, fragment shader is empty)
  7. Render scene (and show map gets filled automatically)
  8. Unbind FBO

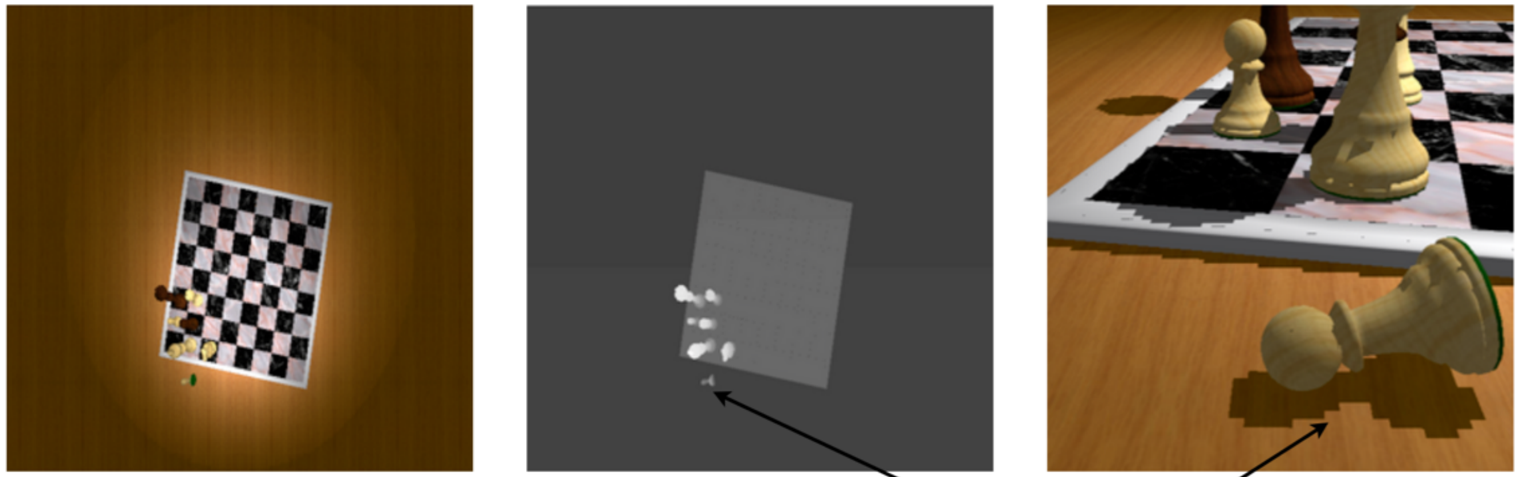
Details Accumulation Pass

1. Generate default shader
2. Pass shadow map(s) accessible in shader by using **`uniforms`**
3. Pass mvp matrices for both, camera and light to shader as **`uniforms`**
4. Render scene again
5. Vertex shader applies mvp for both, camera and lights (afterwards the coords are in NDC)
6. Transform light's NDC coords into [0, 1] range (NDC is in [-1, 1]) with $\begin{pmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
   x and y are the uv coords into the shadow map, z is the distance to the light
7. If the fragment shader check if pixels are in shadow and shade accordingly

```
if (texture(shadow_map, shadow_ndc.xy).r < (shadow_ndc.z - bias))
      shadow = true;
```

Shadow Map Resolution

- What if the shadow map has low resolution?
- What if the shadow map has to cover large scene? (e.g. outdoor landscape)

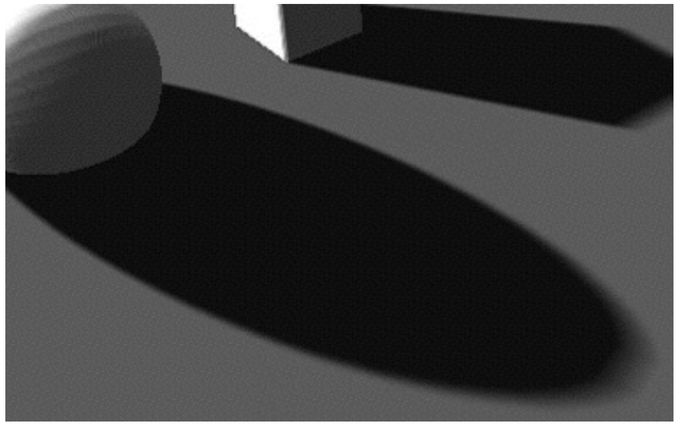


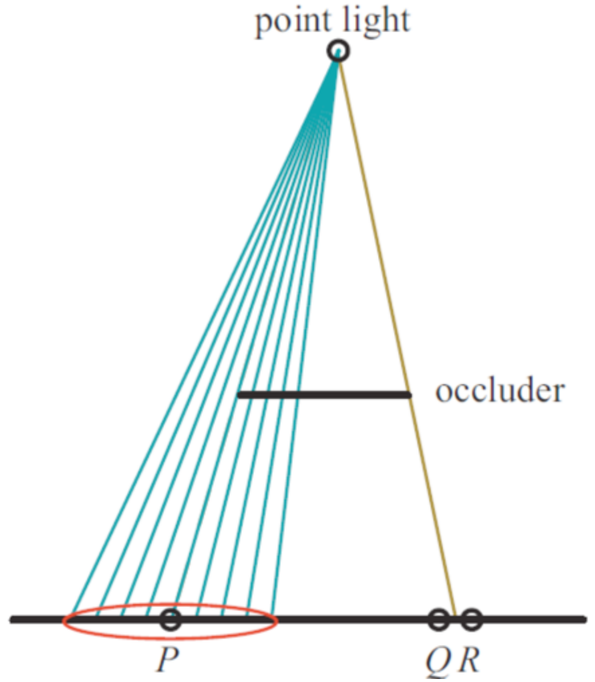
Light Frustum

- What is the ideal light frustum?
  - As tight as possible but must contain all objects that could cast shadows into the camera frustum

Soft Shadows

- Low-pass filter shadow map pixels?
  - No, corresponds to filtered geometry
- Filter boolean shadow map results
  - Sample in pixel vicinity
  - How many shadow map samples yield light/shadow?
  - *Percentage Closer Filtering* (PCF)
  - Sample positions around P,
    e.g. with Poisson disk sampling



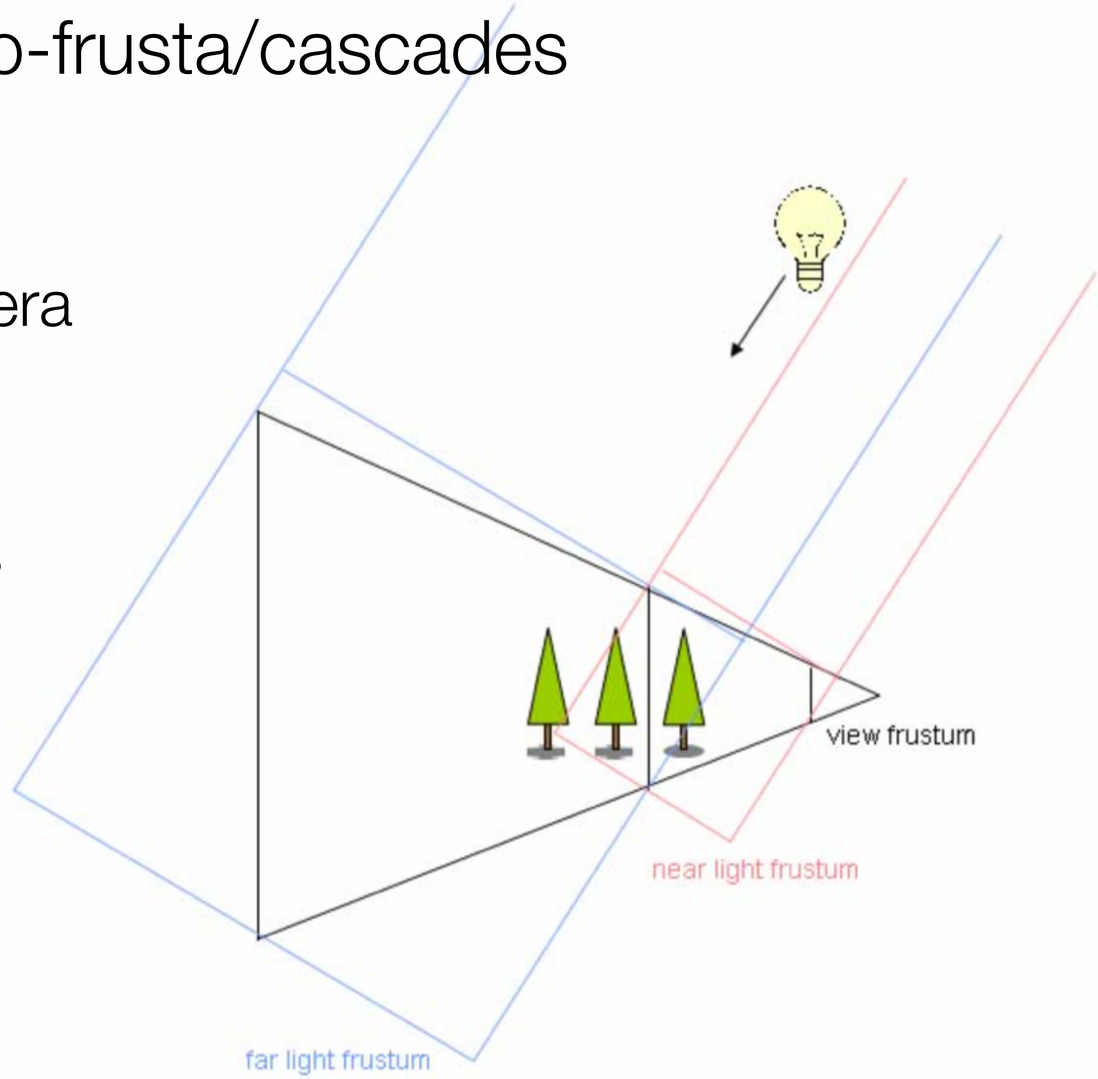


Cascaded Shadow Maps

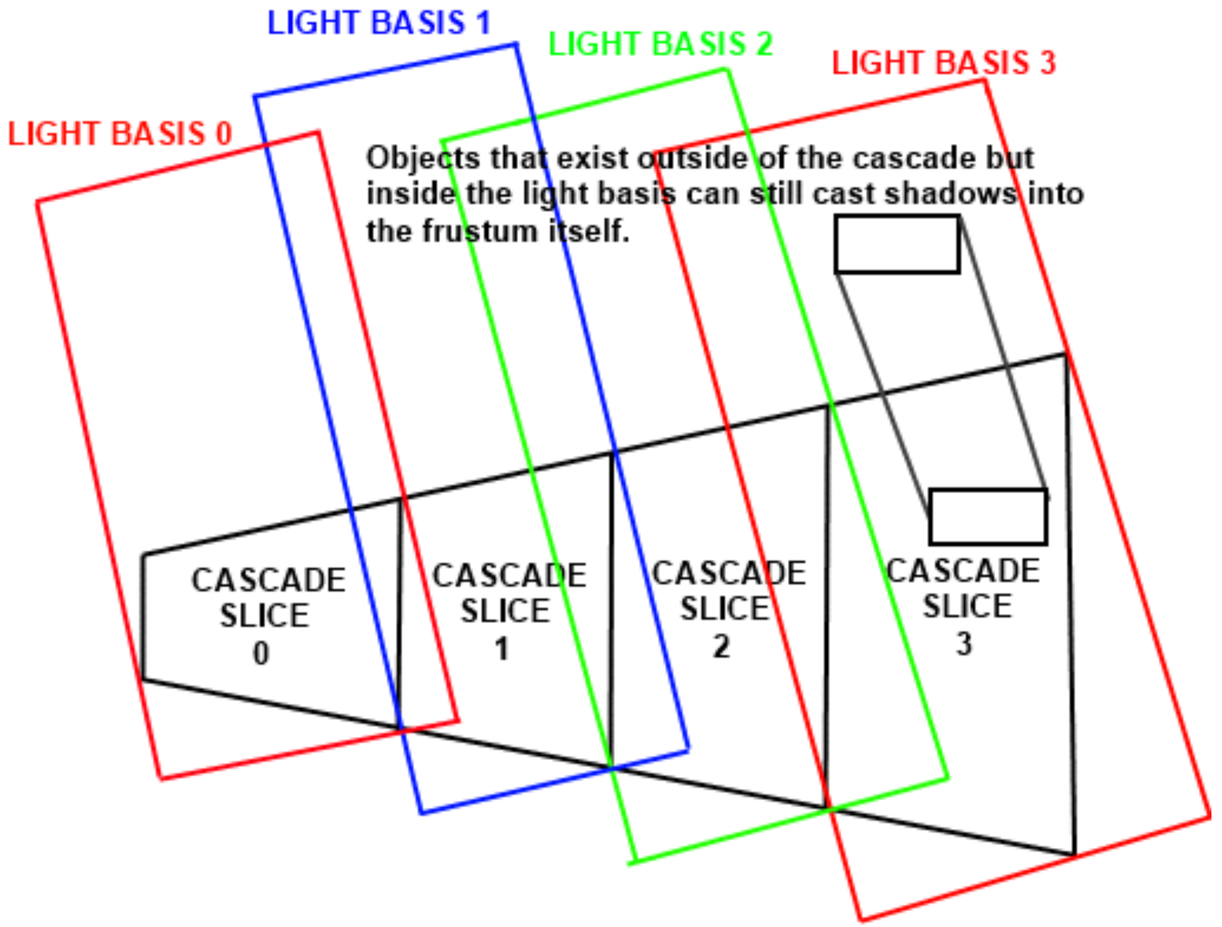
- Use a combination of several shadow maps
- Ideal for large outdoor scenes
- Rouslan Dimitrov, *Cascaded Shadow Maps*, 2007

Cascaded Shadow Maps

- Split camera frustum into several sub-frusta/cascades
  - Create shadow map for each cascade
  - Higher shadow resolution closer to camera
  - Lower shadow resolution further away
  - State of the-art games use ~4 cascades



- Extend light frusta if objects outside camera frustum cast shadows



Reminder

- Intermediate feedback session next week
  - mandatory
  - pass/fail
    - pass = you're there on Thursday and have done "something"
    - fail = you don't show up or you've done absolutely nothing so far, -0.5pt of project grade
    - send email to icg15@groupes.epfl.ch if you can't come Thursday to make an appointment

Reminder

- Talk by Pascal Mueller this Thursday 1pm about Esri/CityEngine