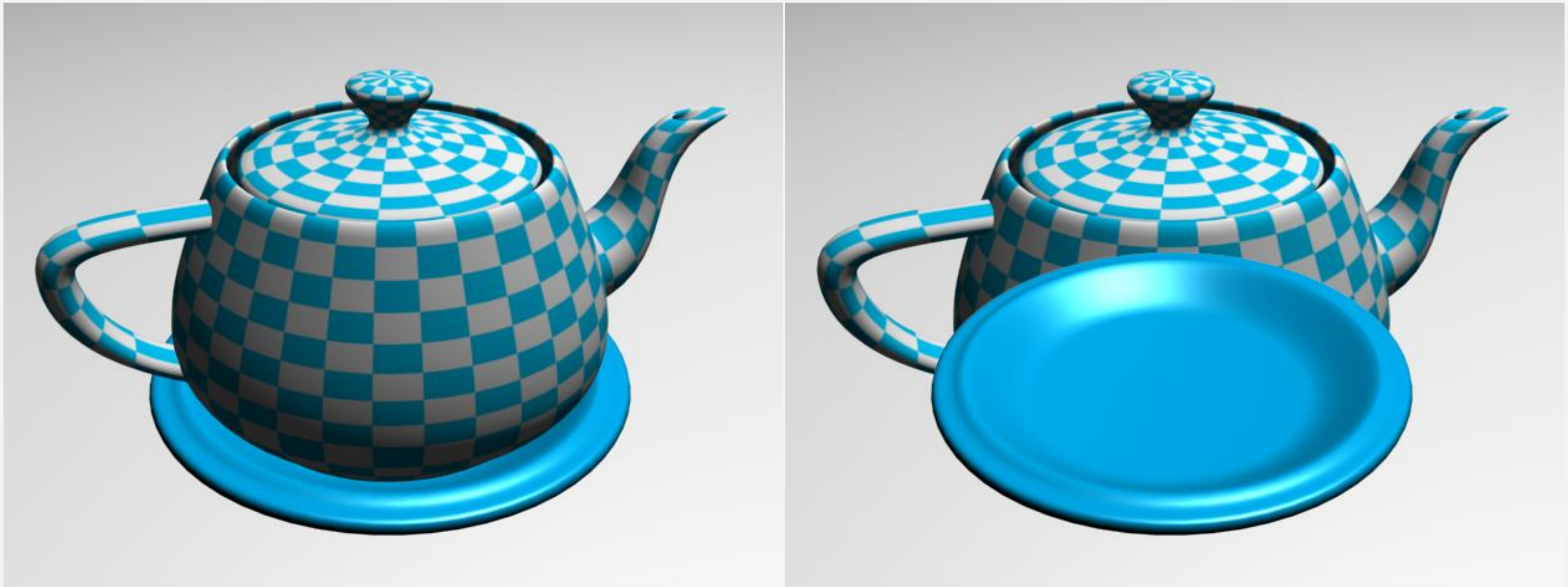
Visibility and shadows

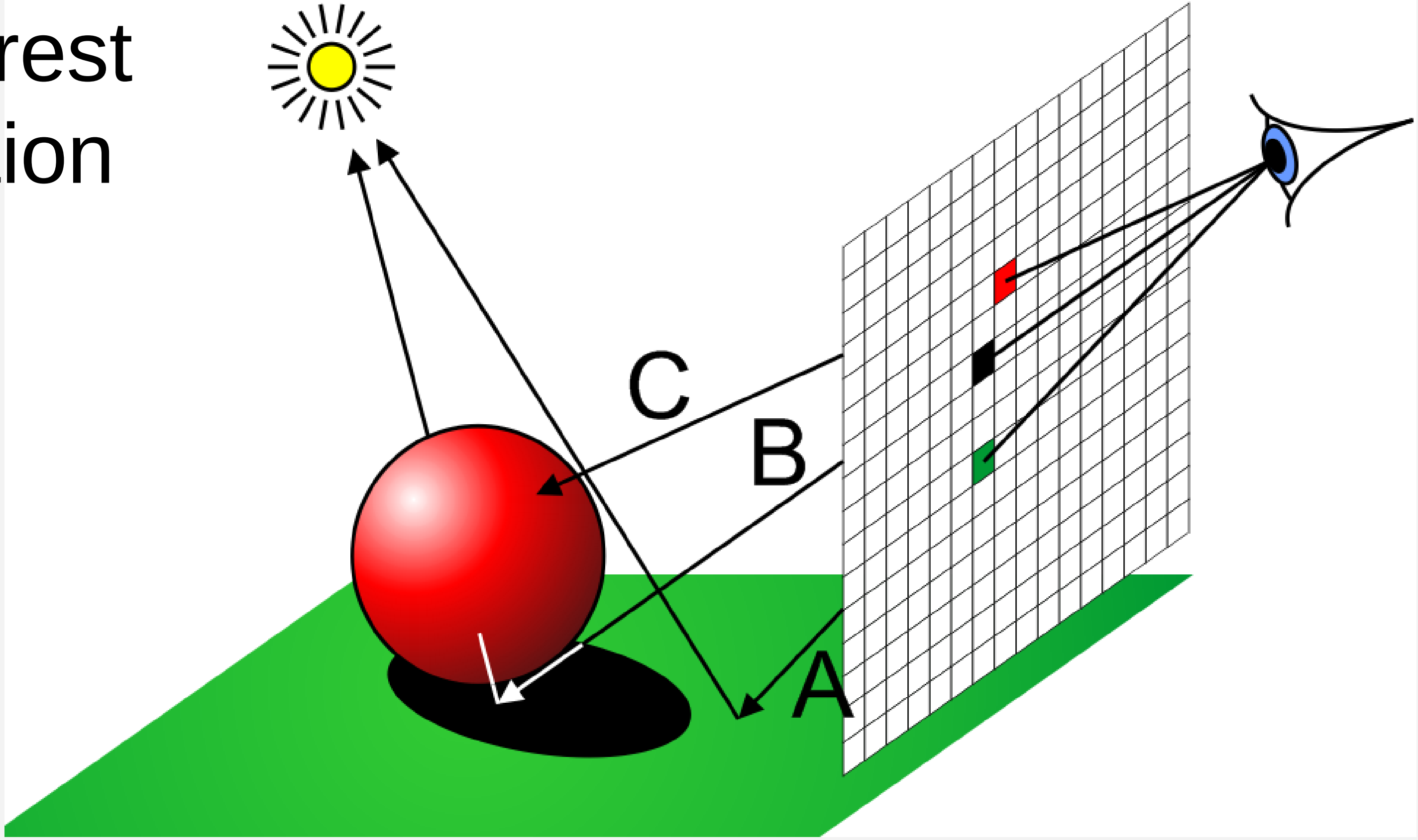
Why visibility?

- projecting 3D objects into 2D not enough
- depth has to be considered (the 3rd D)



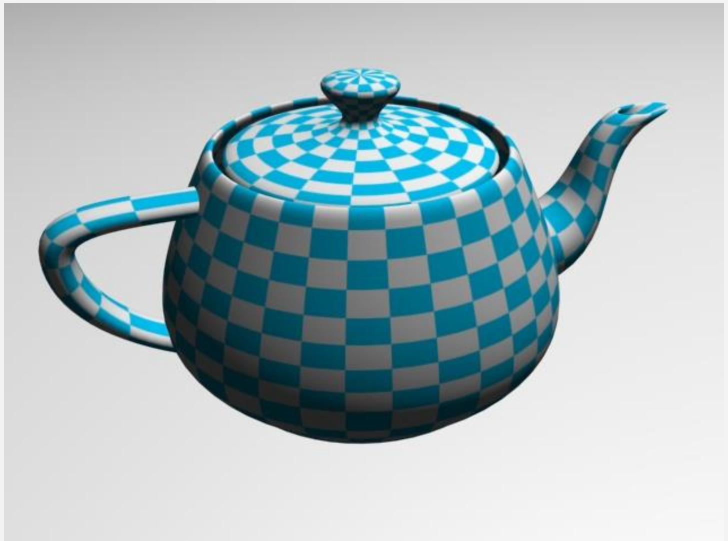
How do we solve visibility?

- Remember raytracing?
- All objects evaluated at the same time
- The nearest intersection counts



Visibility in other methods

- Remember local → world → camera
- Each object treated separately (locally)
  - i.e. we need to take care of the depth ourselves, unlike raytracing
  - moreover, one object has many separate faces!



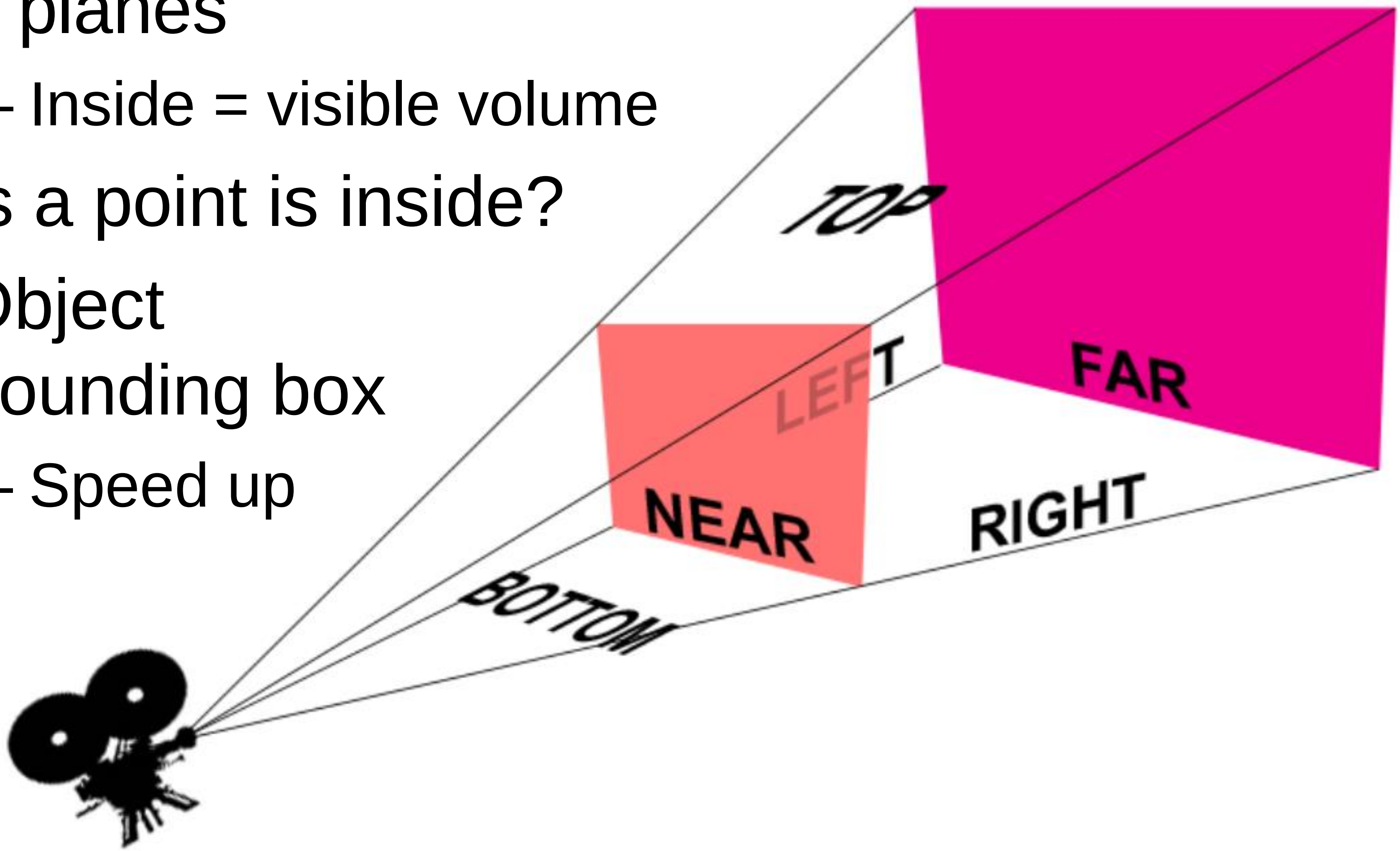
Optimizing visibility

- Get rid of objects that are surely not visible

- Frustum culling
- Backface culling
- Occlusion culling

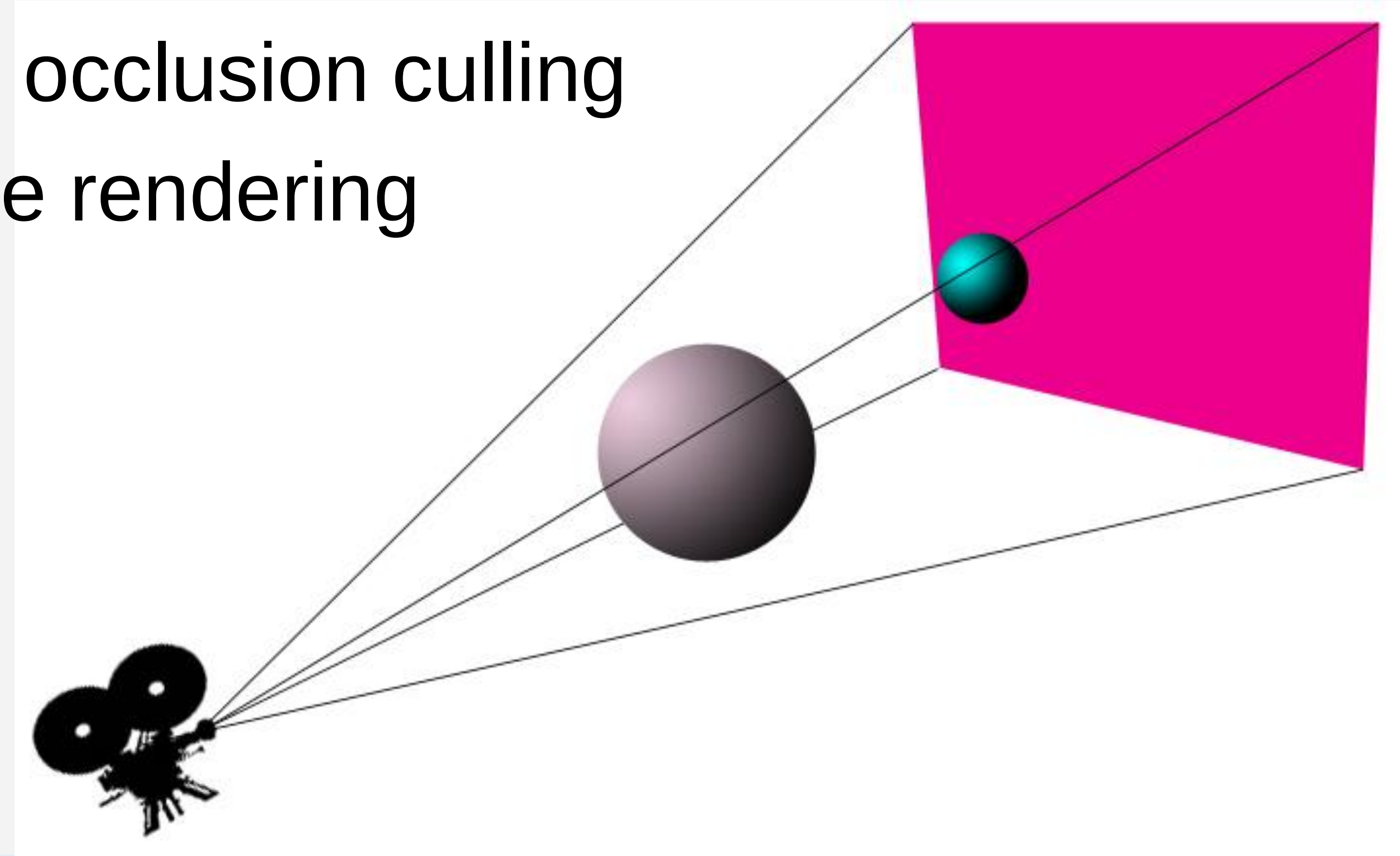
Frustum culling

- 6 planes
  - Inside = visible volume
- Is a point is inside?
- Object bounding box
  - Speed up



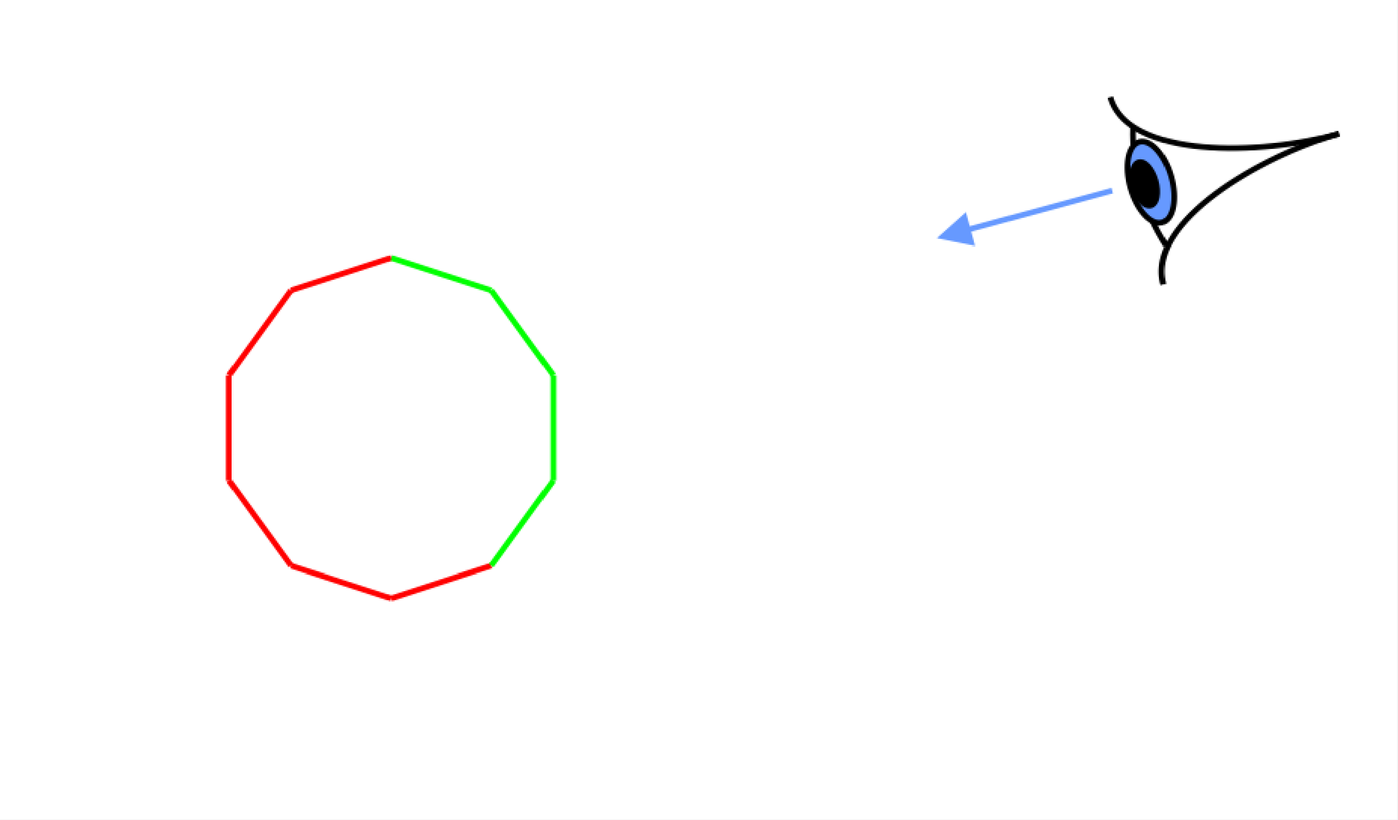
Occlusion culling

- Some objects are fully occluded by others
- Spatial relations between objects
- Portals, occlusion culling
- Realtime rendering



Backface culling

- Which object faces are visible?
- Remember normal vector (face orientation)

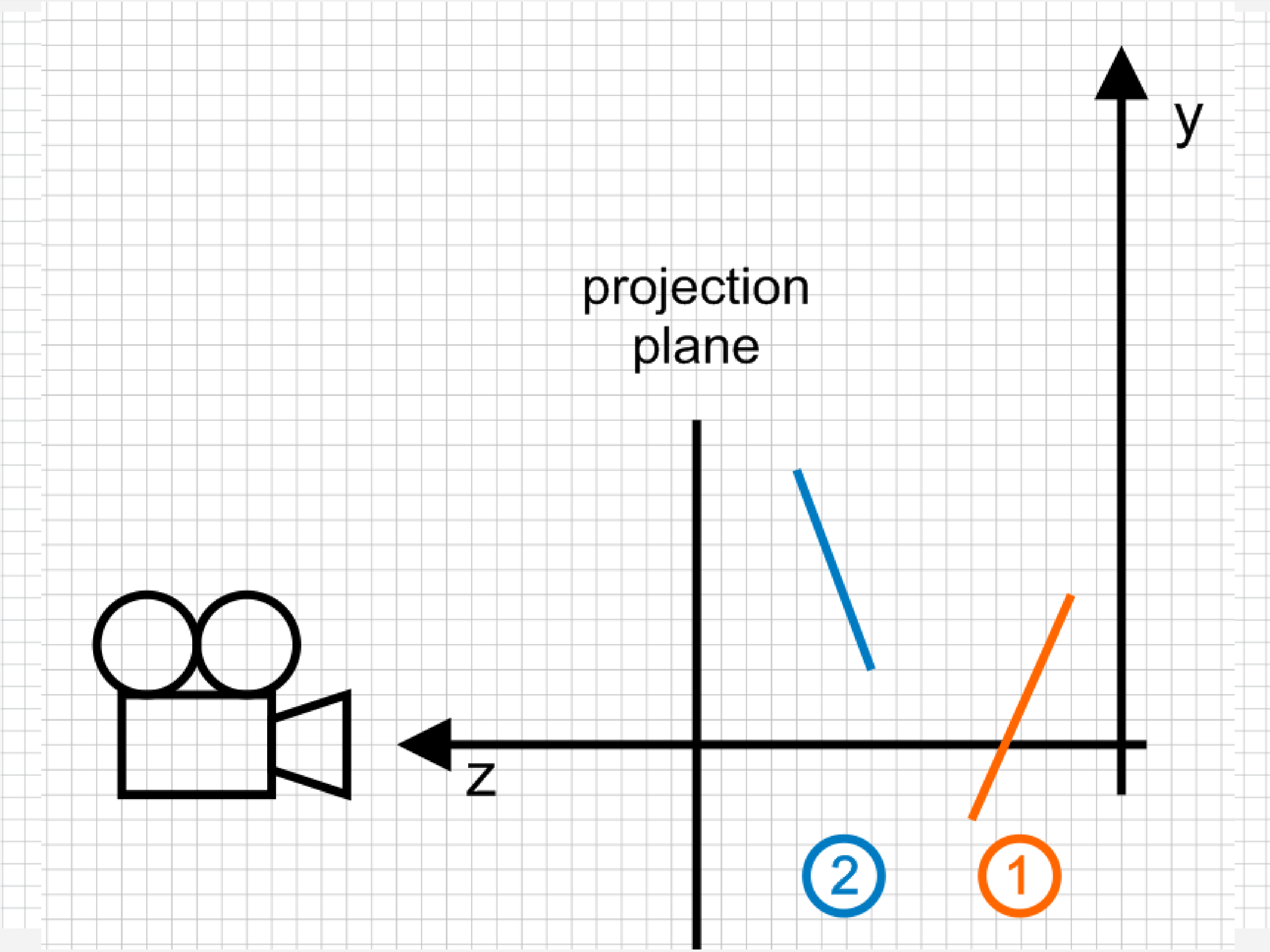


Multiple objects

- Let's consider polygonal objects => reduce the object visibility into faces visibility

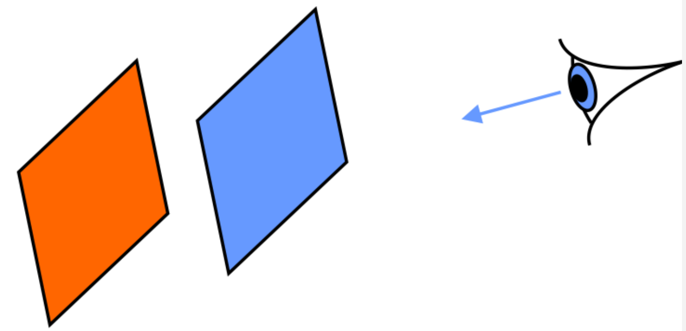
- Main concept – consider object depth (z)

Remember world → camera

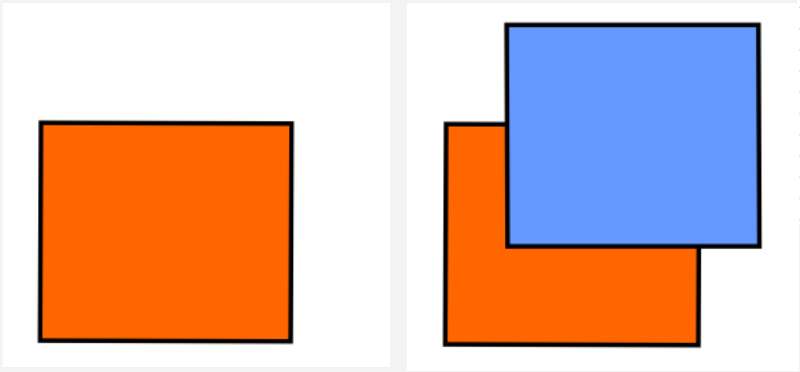


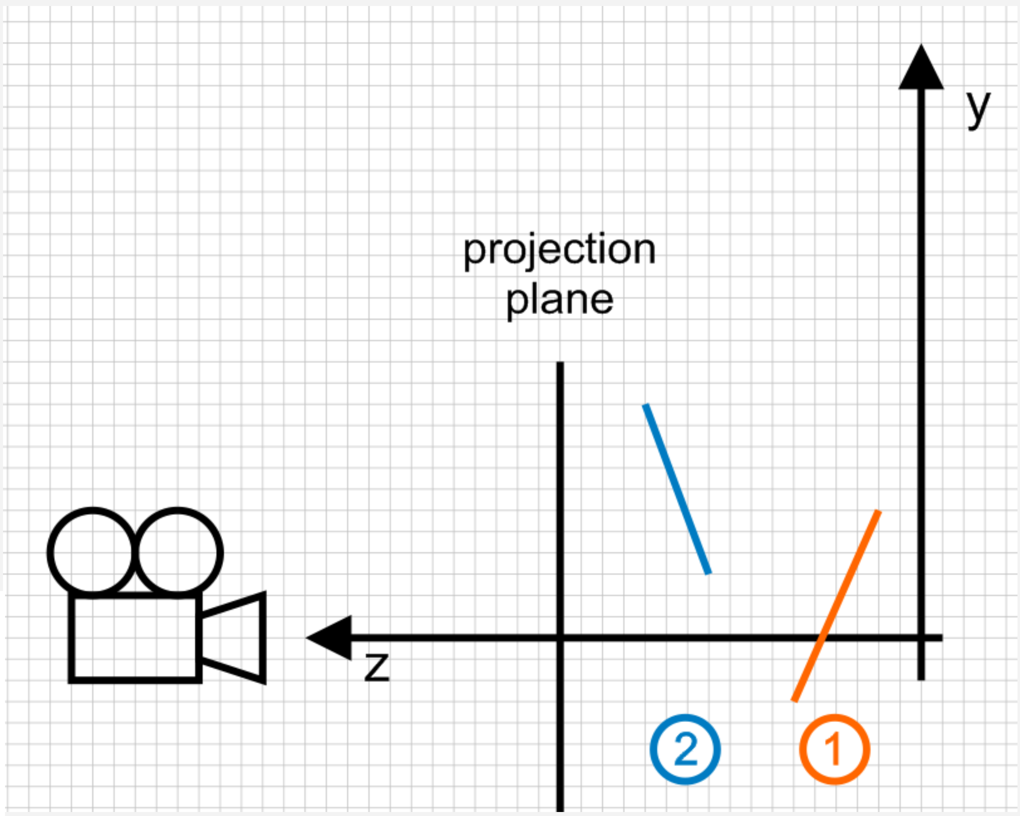
Painter's algorithm

- Sort faces in a back-to-front order, render



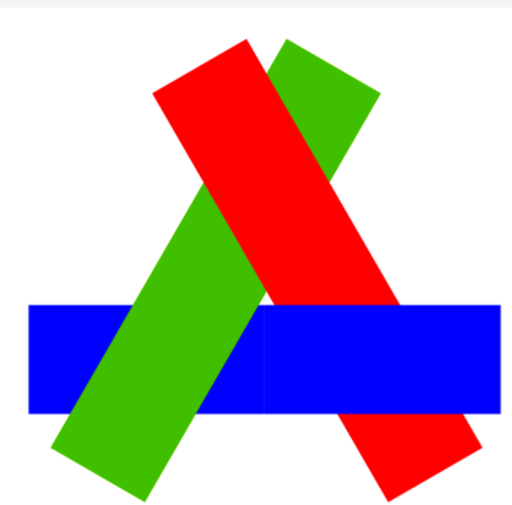
- New pixels over-write old pixels

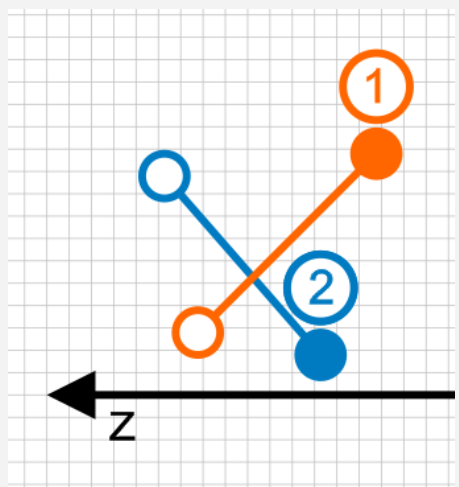


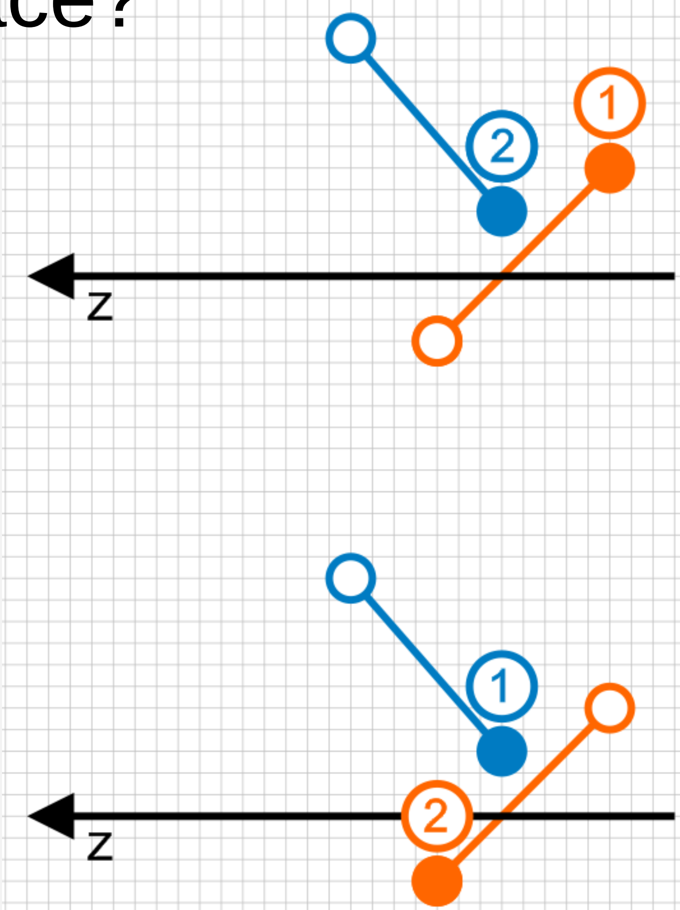


Painter's algorithm problems

- which vertex represents a face?
- intersecting faces
- cyclically overlapping faces







- redundant rendering

Other algorithms

- Warnock algorithm
  - subdivide screen into a quadtree until whole cell empty or whole cell inside polygons
- Reversed painter's algorithm
  - paint front-to-back and paint only empty areas
- Z-buffer
  - remember z-value for each pixel and only paint when new z is higher

Z-buffer

- works in screen space
- screen w×h ↔ z-buffer w×h
- `for each 0≤x≤w,0≤y≤h: z-buffer[x,y] ← z`$_{min}$

```
for each face:
    rasterize it into pixels {x,y,z}
        for each face's pixel (x,y,z):
            if z > z-buffer[x,y] then :
            z-buffer[x,y] ← z
            and screen[x,y] ← face color
```

Z-buffer pros and cons

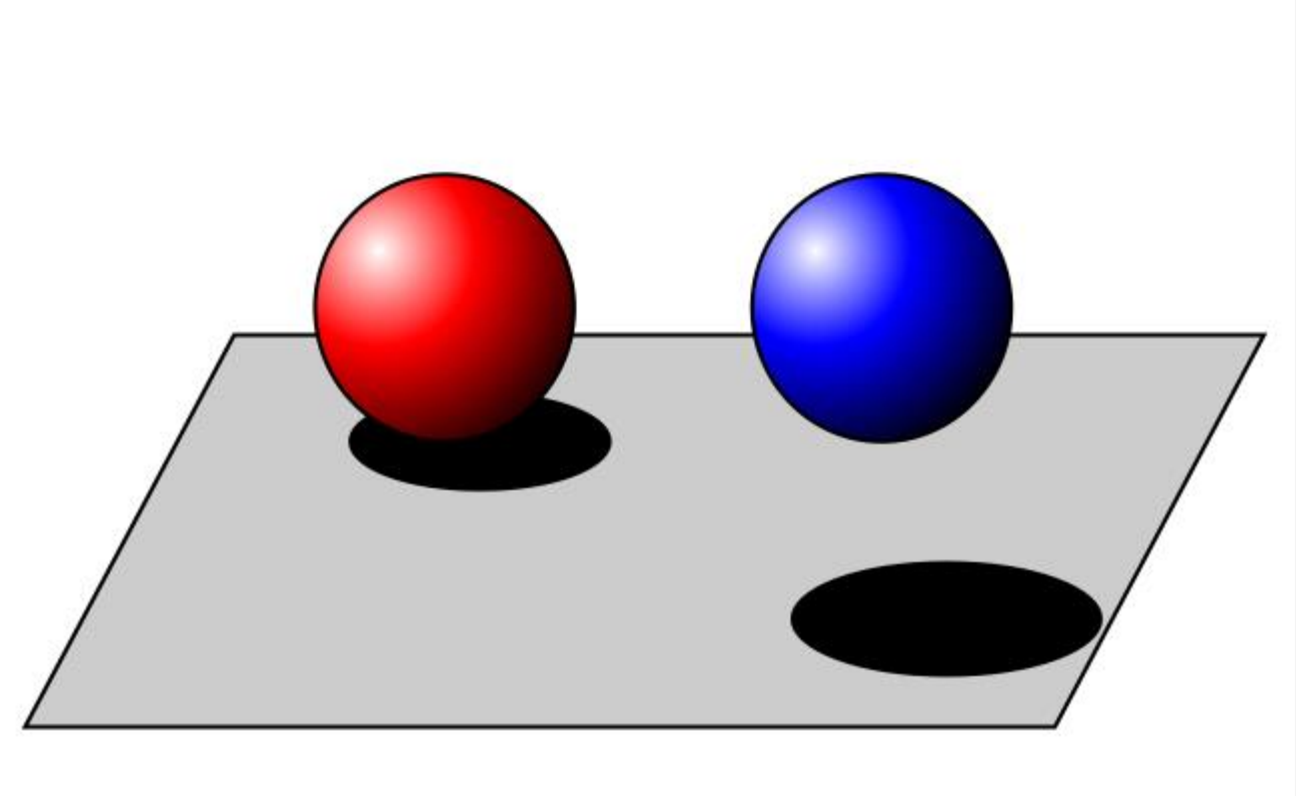
- GPU support
- precision issues might occur
- z-buffer test before per-pixel-lighting or pixel shading saves a lot of redundant work
- memory demands (width×height×precision)
  – can be reduced by scanline (width×1×precision)

Lights, visibility, texture...

What's missing is shadows

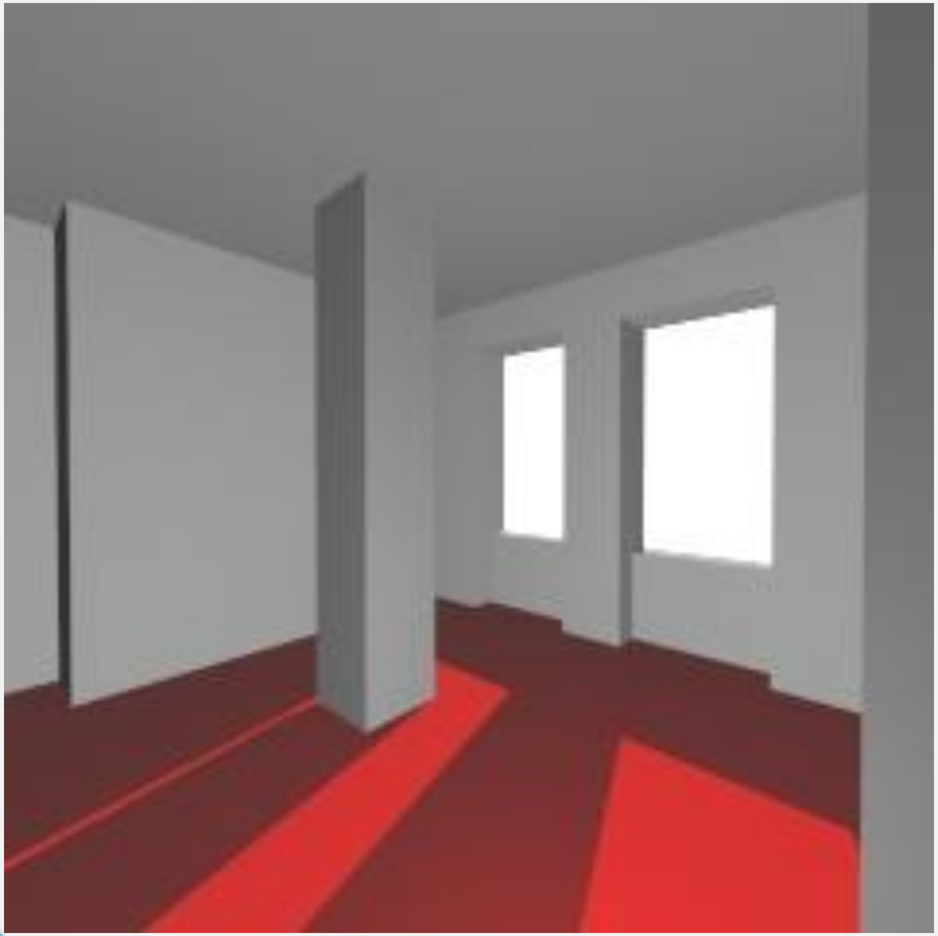
Shadows

Why shadows?



Shadows in global methods

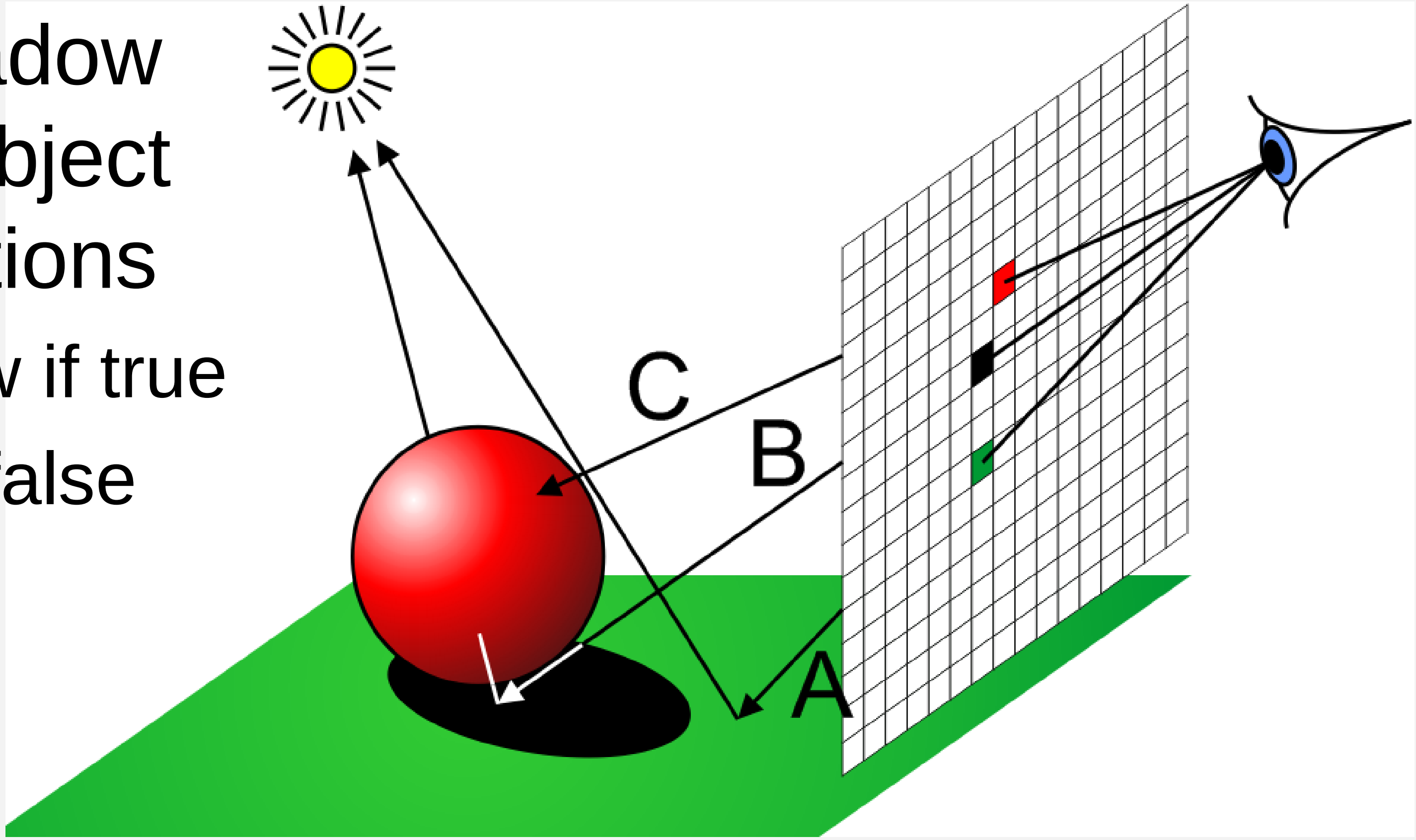
raytracing



radiosity

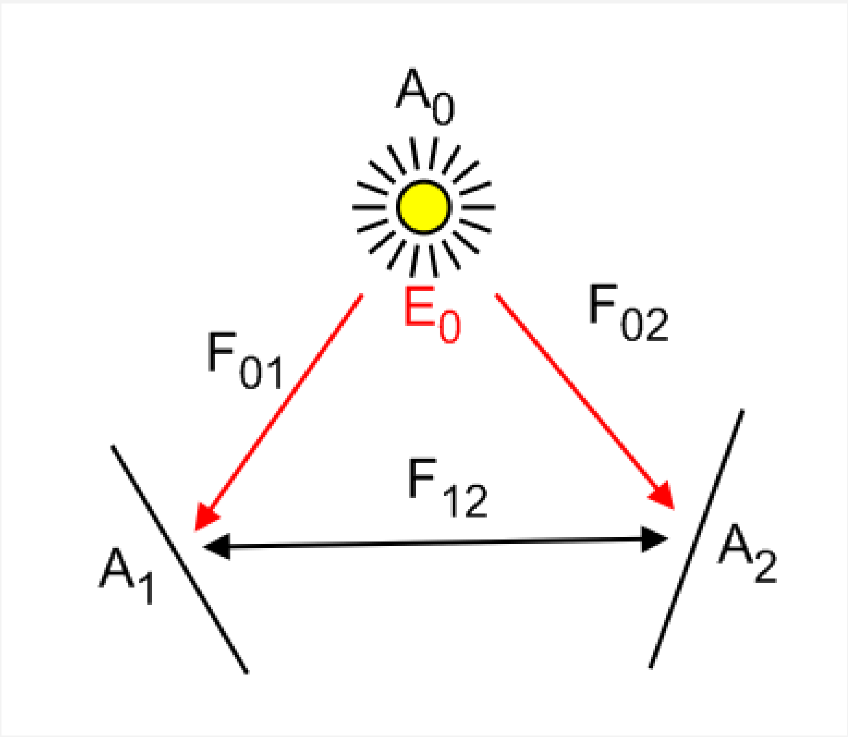
Raytraced shadows

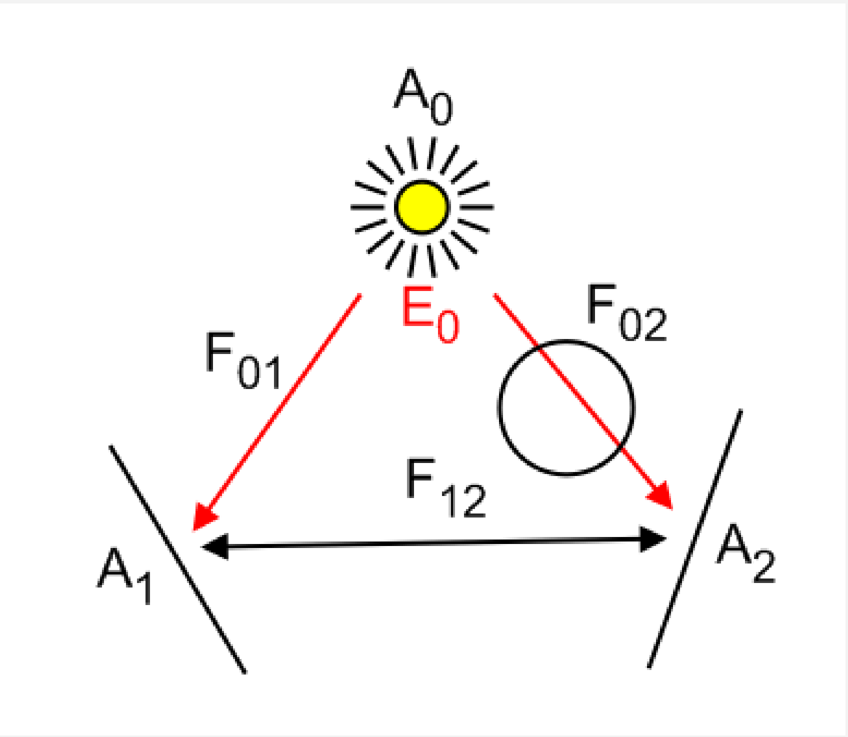
- Camera→pixel→ray→**intersection**←object
- Intersection→**shadow ray**→light(s)
- Test shadow ray for object intersections
  - shadow if true
  - light if false



Radiosity shadows

- Are created naturally without special effort
- Mutually occluded areas have low form factor





- e.g. $F_{02} = 0.7$ $F_{02} = 0.05$

Shadows in local methods

Shadow volumes

geometry space

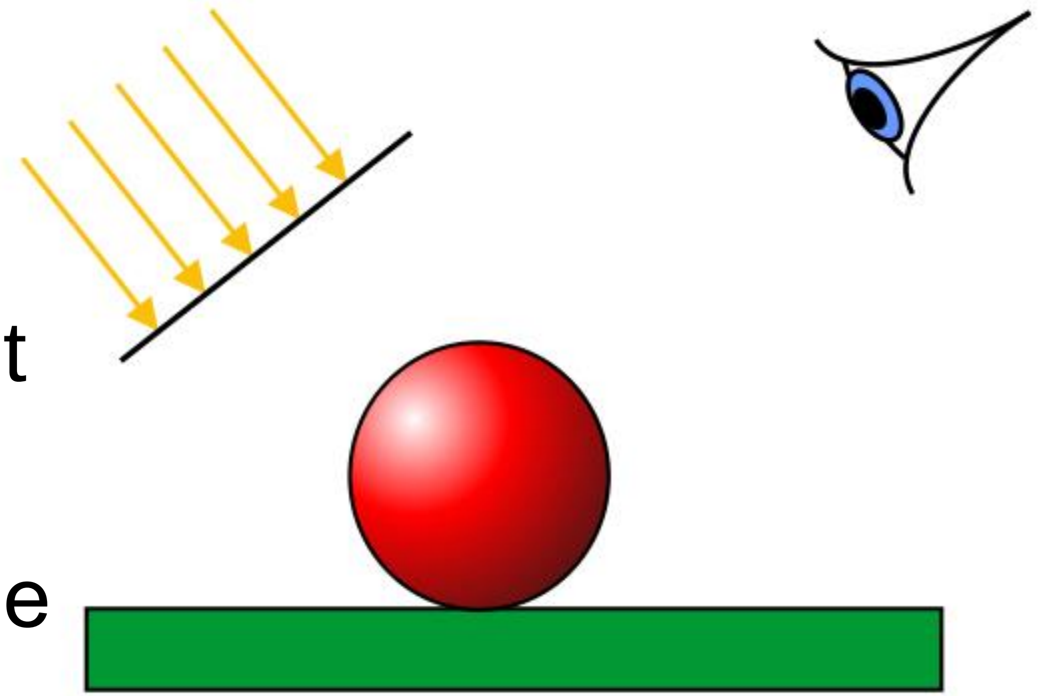
REALTIME

Shadow maps

screen space

Shadow maps

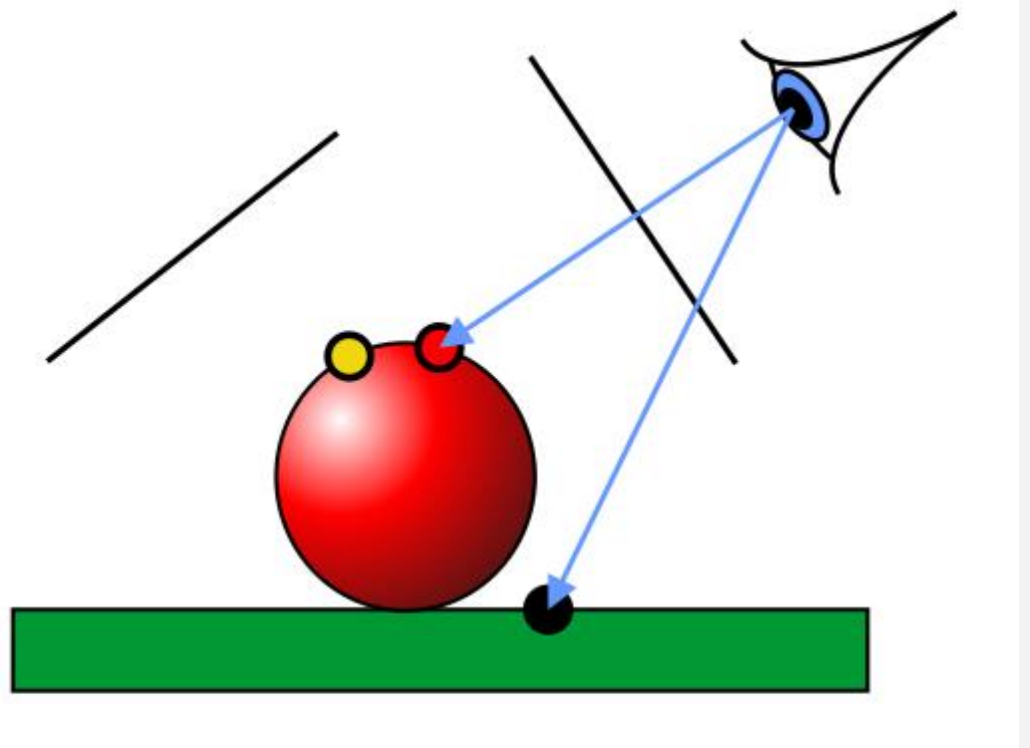
- z-buffer analogy
- look from the light
- “render” the scene and store depth information in a shadow map
  - 2D raster data
  - smallest distance between light and objects



Shadow maps

- For a polygon pixel to be rendered

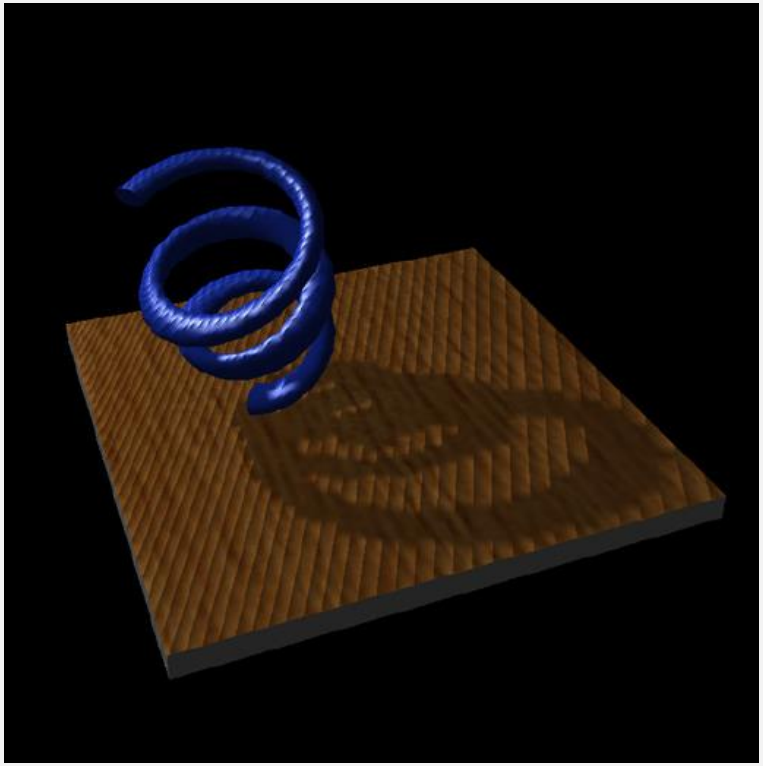
- Find its position in the light's projection plane → (x,y,z)



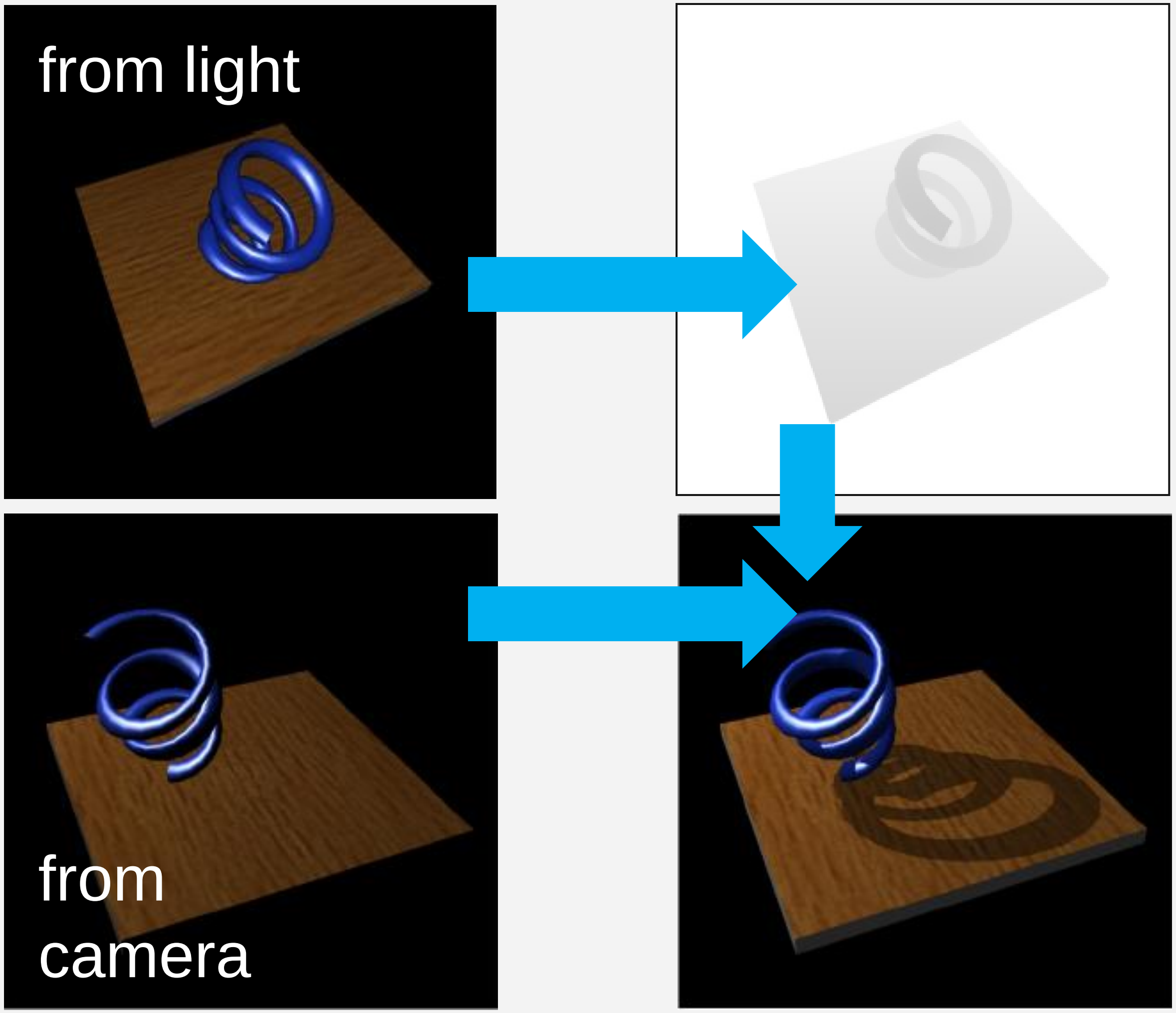
- If z > shadow map [x,y] then shadow

Shadow maps pros & cons

- memory consumption
  - 1 light = 1 shadow map
  - high resolution necessary
- aliasing
  - use high resolution
  - filtering necessary
- smooth (soft) shadows
  - when filtered
- imprecise due to z-buffer quantization
- light-specific transformation



Shadow mapping example



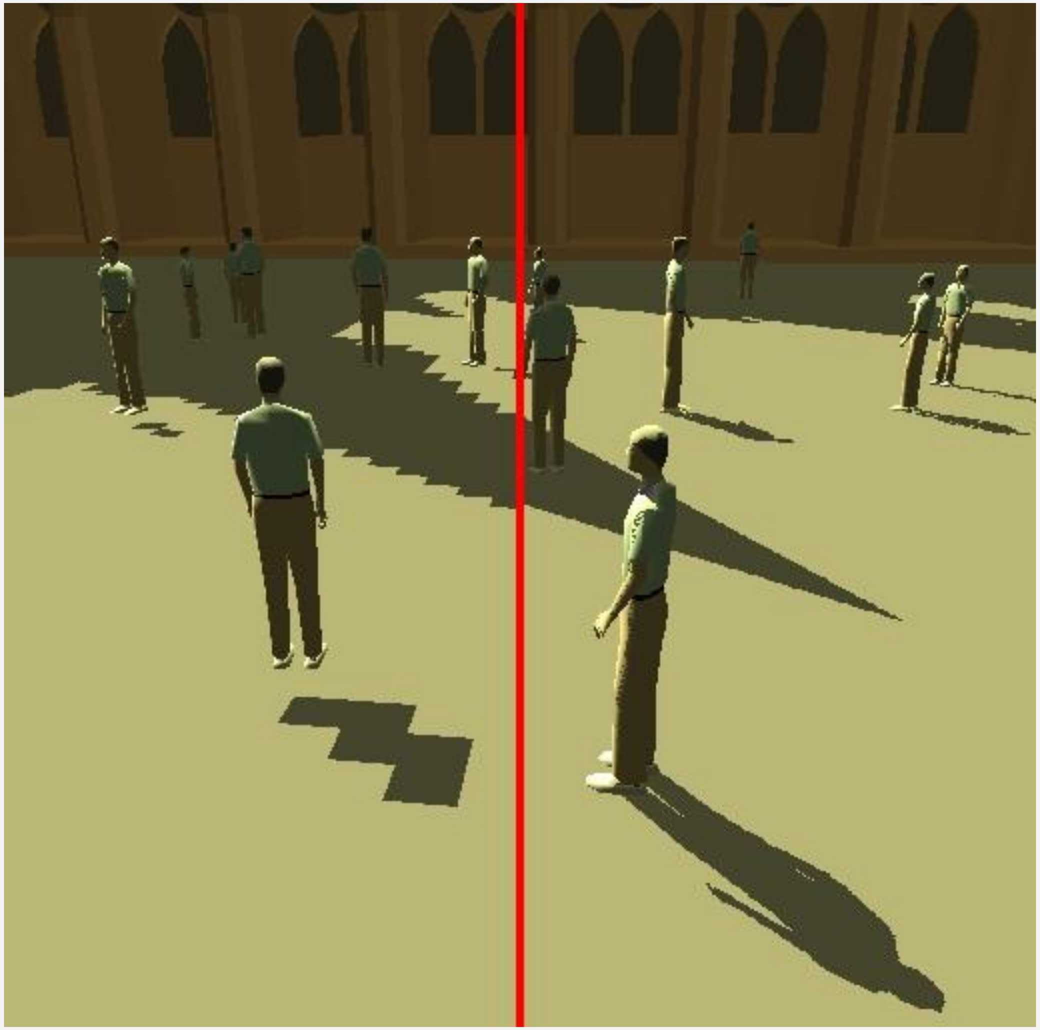
depth buffer from light's point of view

final image

http://www.nealen.net/projects/ibr/shadows.pdf

Shadow map resolution

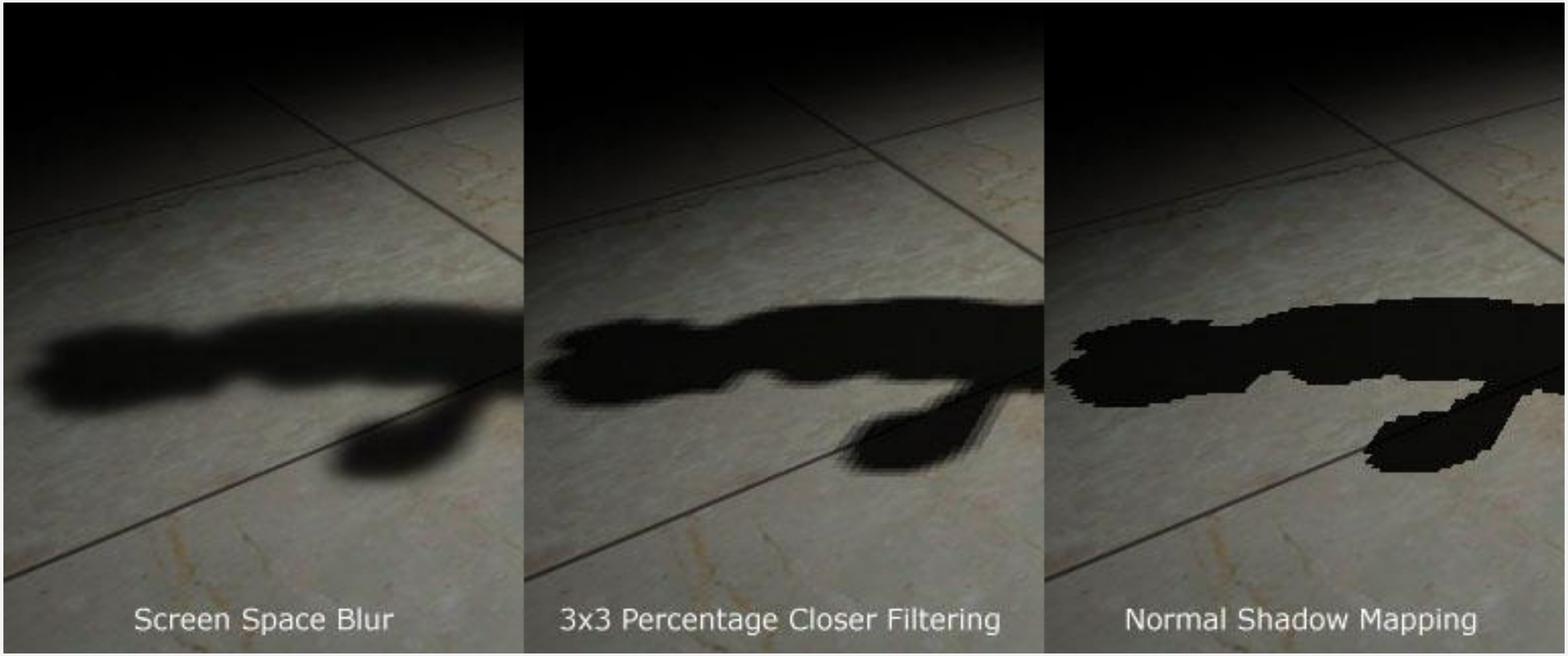
- How many points are stored in the 2D shadow map
- Low counts = shadow artifacts



Stamminger, Drettakis: Perspective Shadow Maps

Filtering and soft shadows

- Removes artifacts (jagged edges)
- Simulates soft shadows

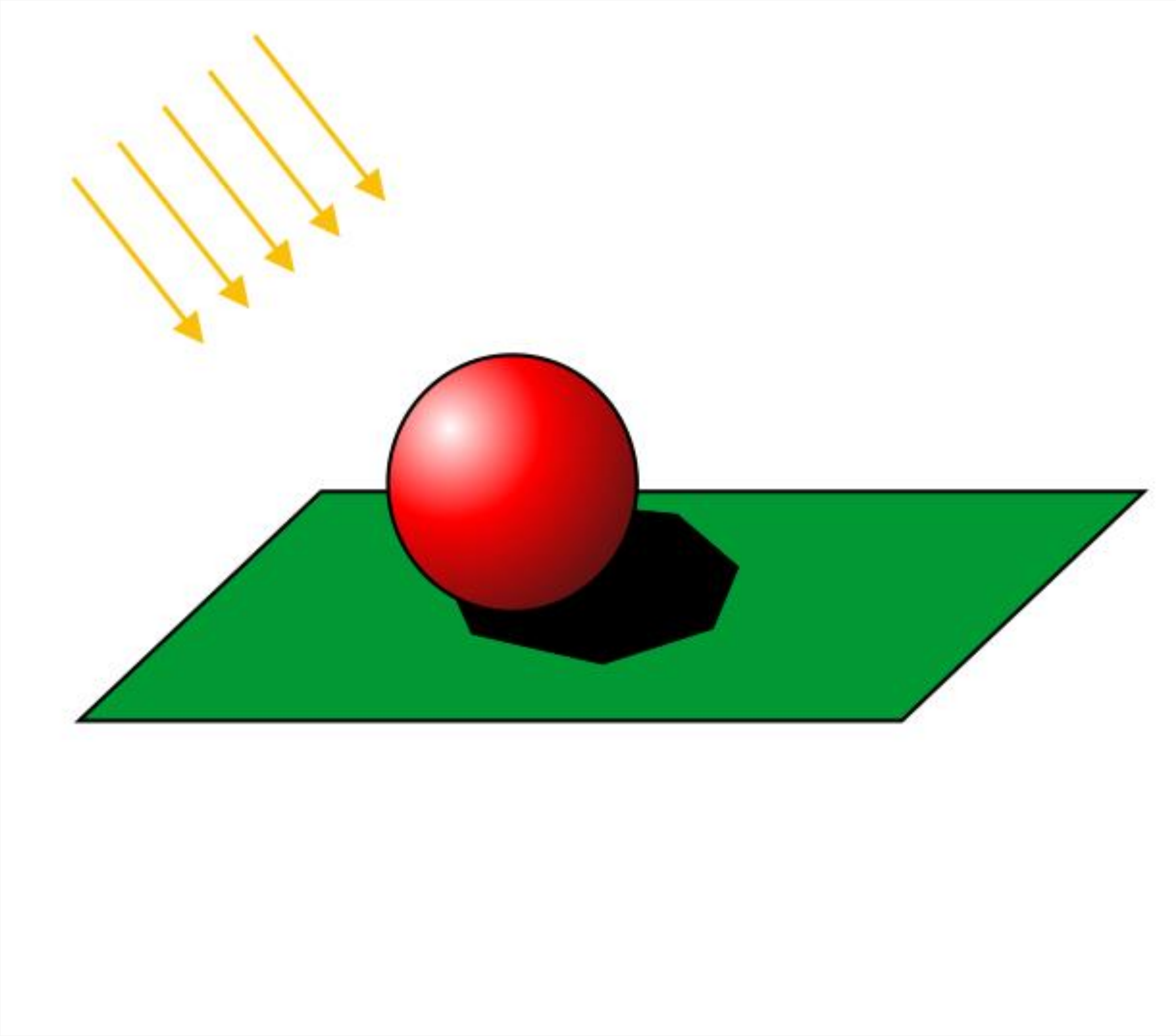


Soft-Edged Shadows, http://www.gamedev.net

Shadow volume

- create dummy geometry object extending each object in the direction of the light
  - shadow volume
- when displaying an object to a pixel (x,y,z), test if (x,y,z) is inside/outside the shadow volume

Shadow volume



Pseudo-code

1. Compute ambient light for whole scene and update z-buffer along with that

2. **Which screen areas are in shadow?**

3. For all areas outside the shadow:
    4. Compute diffuse and specular light components

5. Iterate for all lights

Shadow volume pros & cons

- hard shadows
  – modifications for soft shadows necessary
- GPU implementation using stencil buffer
- high complexity for high-polygon models
- what if camera is inside the shadow volume?
- shadow volumes expensive on CPU
  – now vertex shaders

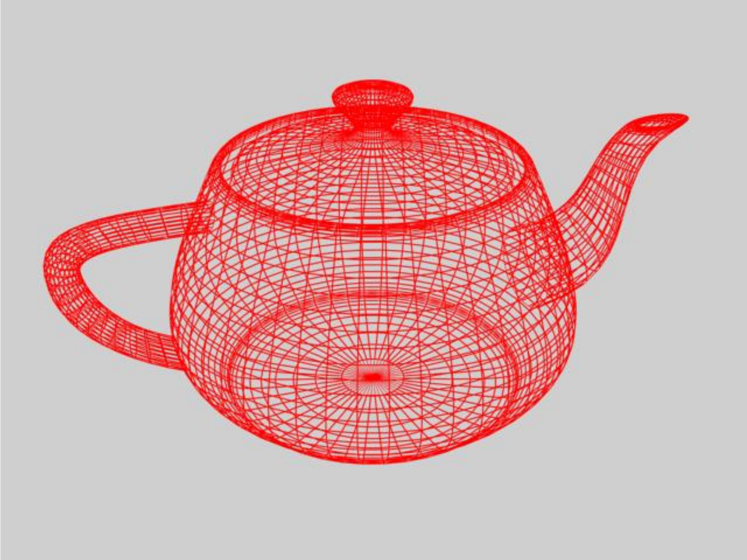
Shadows vs. light types

- Directional (parallel) light
  - easy shadow maps and shadow volumes
- Spot light
  - shadow map by perspective transformation
  - easy shadow volume
- Omni light
  - shadow map hard
  - easy shadow volume (same as spotlight)
- Area light
  - approximate by multiple lights

Summary

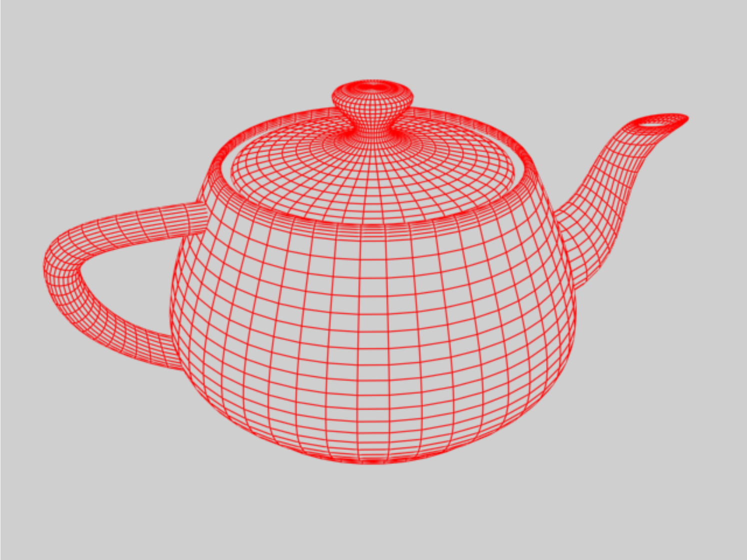
Camera, object, scene

- Local, world (global), camera coordinates
- Transformations (in 2D & 3D)
- Matrix operations
  - translate, rotate, scale
  - projections (orthogonal, perspective)
- Object representation
  - boundary, volume, polygonal parametric, implicit, F-rep



Rasterization, Visibility

- Line and polygon rasterization
- Linear interpolation
- Antialiasing



- Visible volume
- Backface culling
- Painter’s algorithm
- Z-Buffer

Texture

- Texture coordinates, texture mapping
- Texture filtering
  - Bilinear interpolation
  - Nearest neighbor

Lighting

- Light types
- Lighting models and illumination techniques
  - – local, global
  - – empiric, physical
- Shading models
  - – flat, Gouraud, Phong
- Raytracing, radiosity

Shadows

- Shadow generation in global illumination
- Shadow generation in local models
- Stencil shadows (shadow volume)
- Shadow maps
- Soft shadows