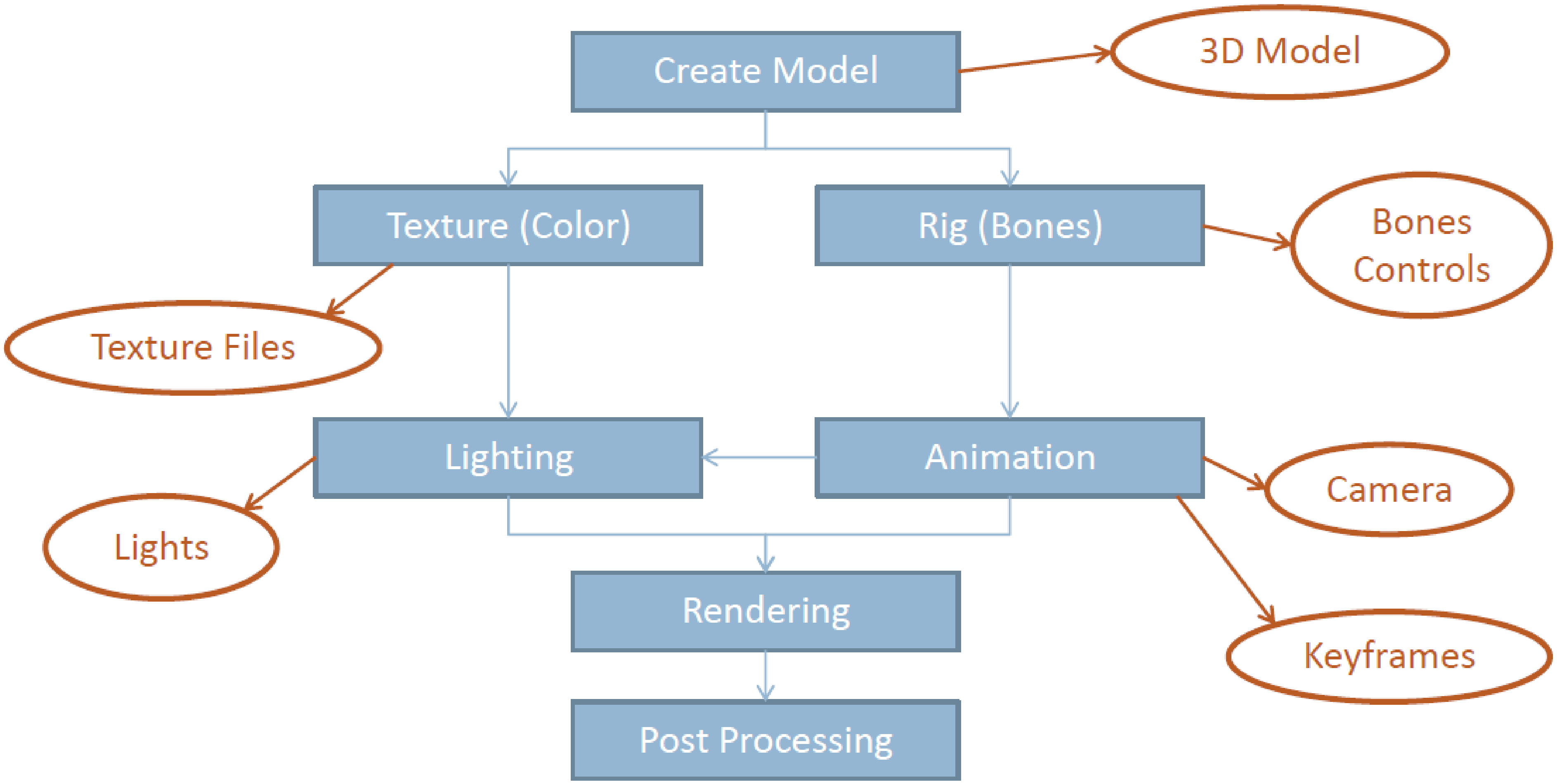
Introduction to Rendering Technics

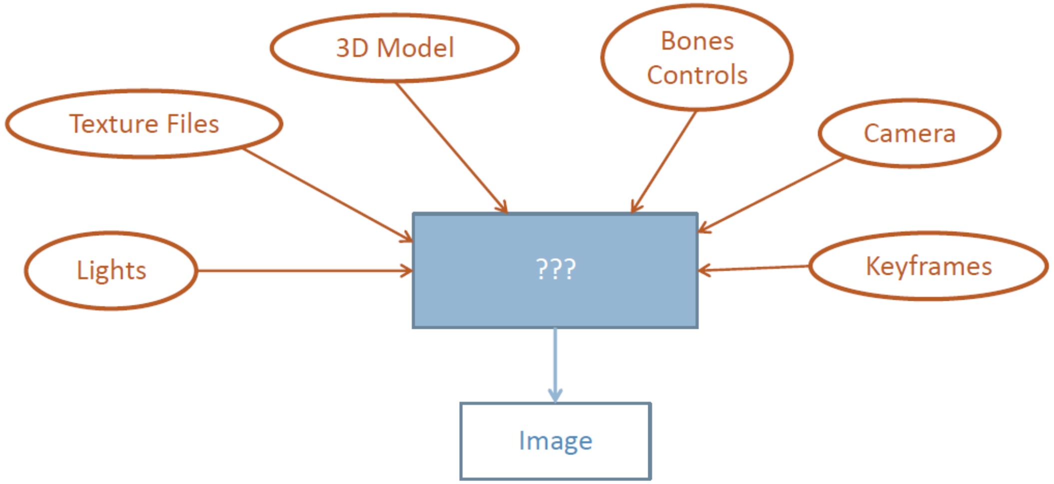
Ciência da Computação - PUCPR
Construção de Software Gráfico 3D

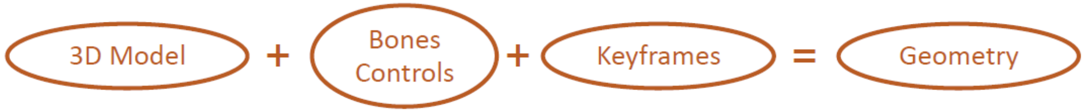
What is 3D Graphics

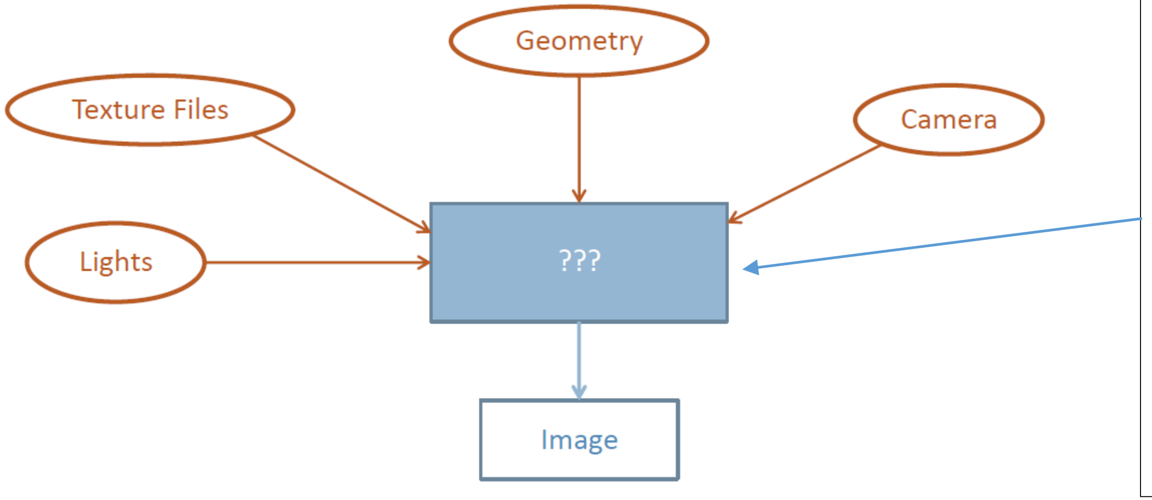
- Artists workflow – in a nutshell



What is Rendering







- Perspective
- Occlusion
- Color / Texture
- Lighting
- Shadows
- Reflections / Refractions
- Indirect illumination
- Sampling / Antialiasing

Rendering – two approaches

- Start from **geometry**
  - For each polygon / triangle:
    - Is it visible?
    - Where is it?
    - What color is it?

Rasterization

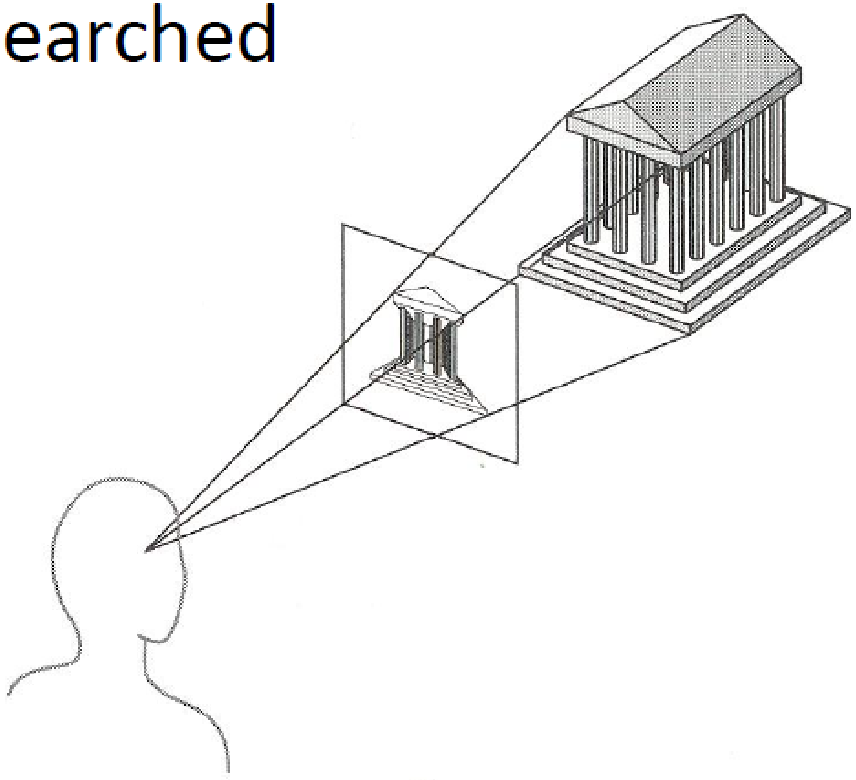
- Start from **pixels**
  - For each pixel in the final image:
    - Which object is visible at this pixel?
    - What color is it?

Ray Tracing

Rasterization

Rasterization

- Basic idea: Calculate projection of each triangle onto the 2D image space
- Extensively used and researched
- Optimized by GPU
- Strongly parallelized
- OpenGL
- DirectX

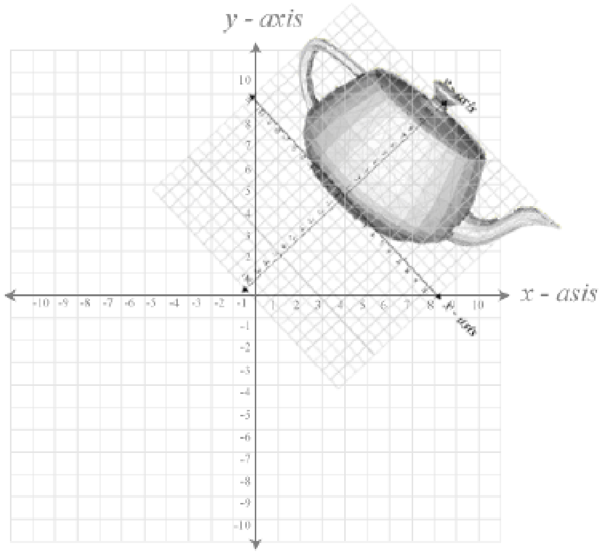


Rasterization – graphics pipeline

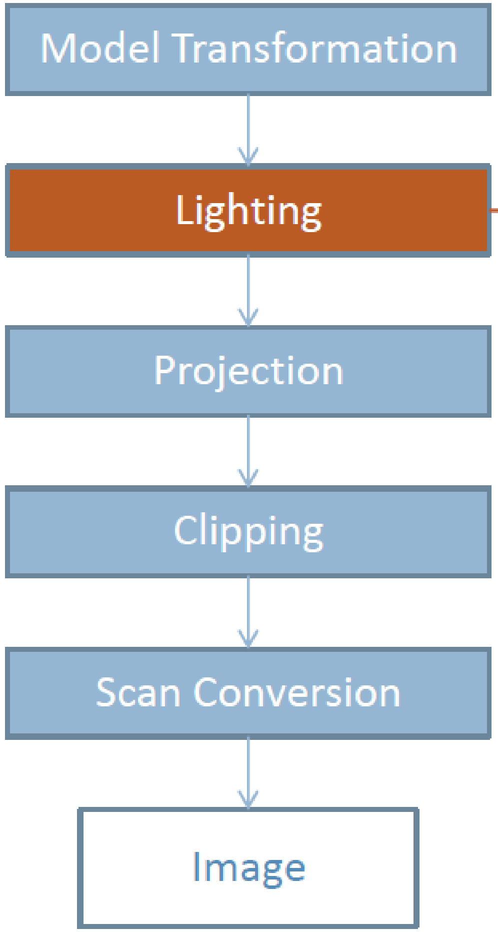
Model Transformation → Lighting → Projection → Clipping → Scan Conversion → Image

Model Transformation → Lighting → Projection → Clipping → Scan Conversion → Image

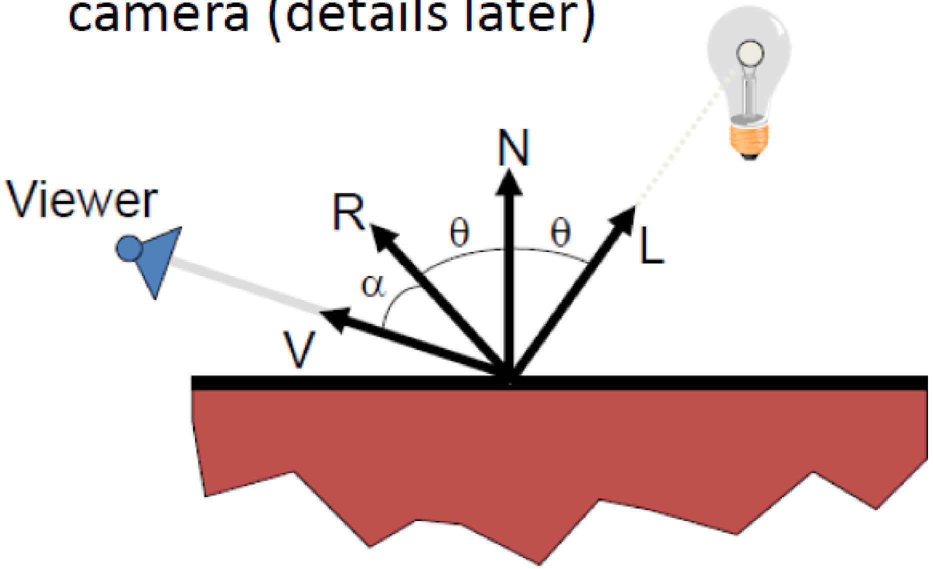
- Transform each triangle from **object space** to **world space**
- Local space -> Global space



Rasterization – graphics pipeline



- Computation is based on angles between light source, object and camera (details later)

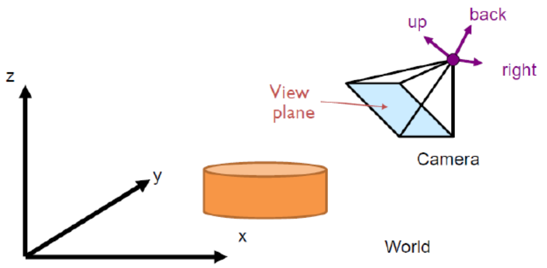


- Backface culling

Rasterization – graphics pipeline

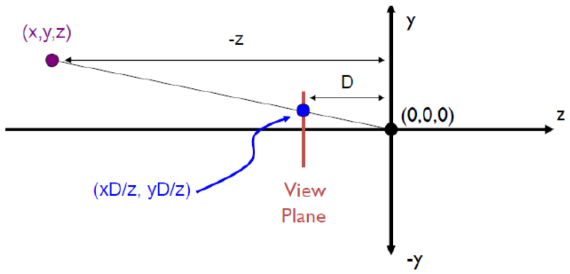
Model Transformation
↓
Lighting
↓
Projection
↓
Clipping
↓
Scan Conversion
↓
Image

- Step 1: Transform triangles from **world space** to **camera space** (orthogonal transformation)



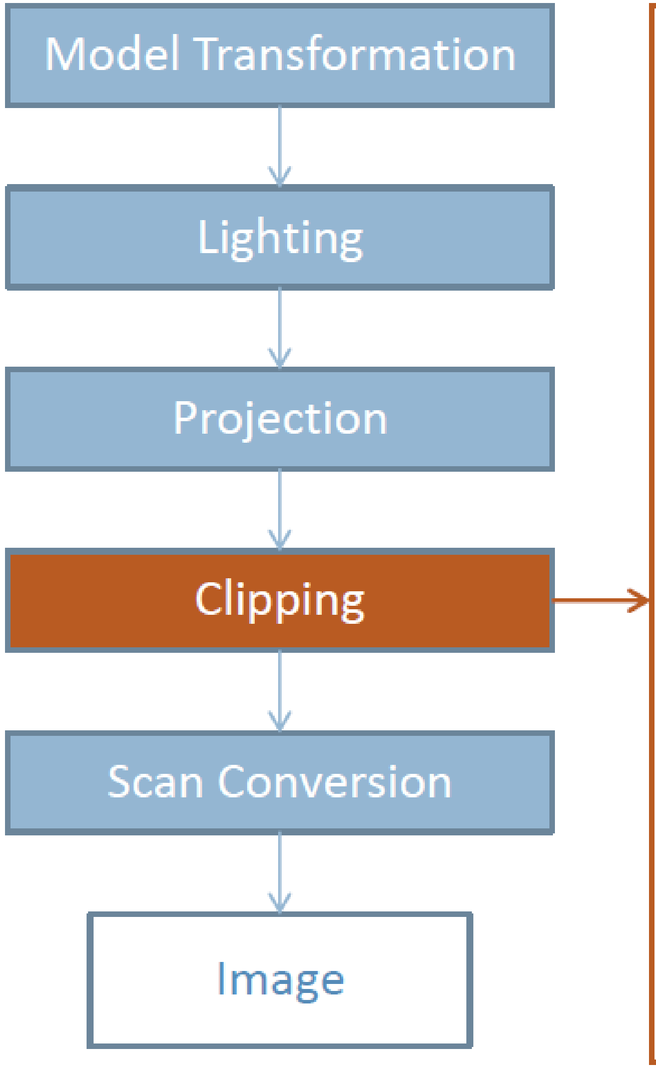
- Step 1: Transform triangles from **world space** to **camera space** (orthogonal transformation)
- Camera is at (0, 0, 0)
- X axis is right vector
- Y axis is up vector
- Z axis is “back vector” (away from camera)

- Step 2: Perspective Projection
- Depends on focal length (D)

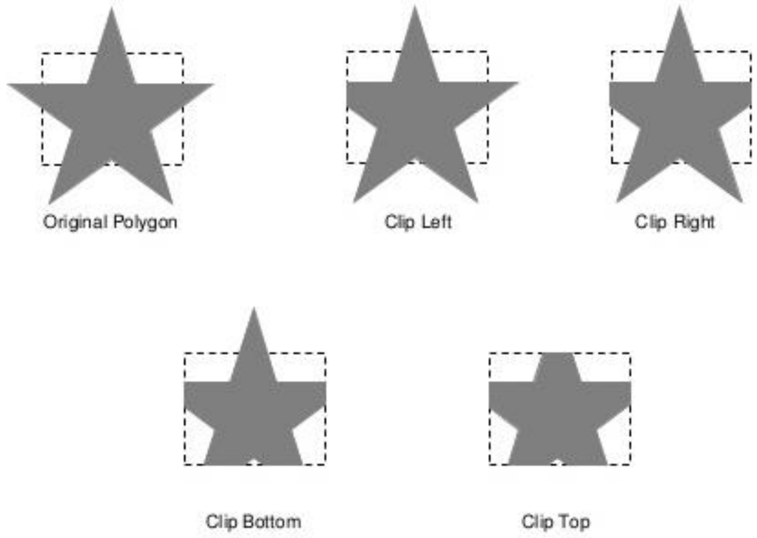


- Calculate Z-Buffer

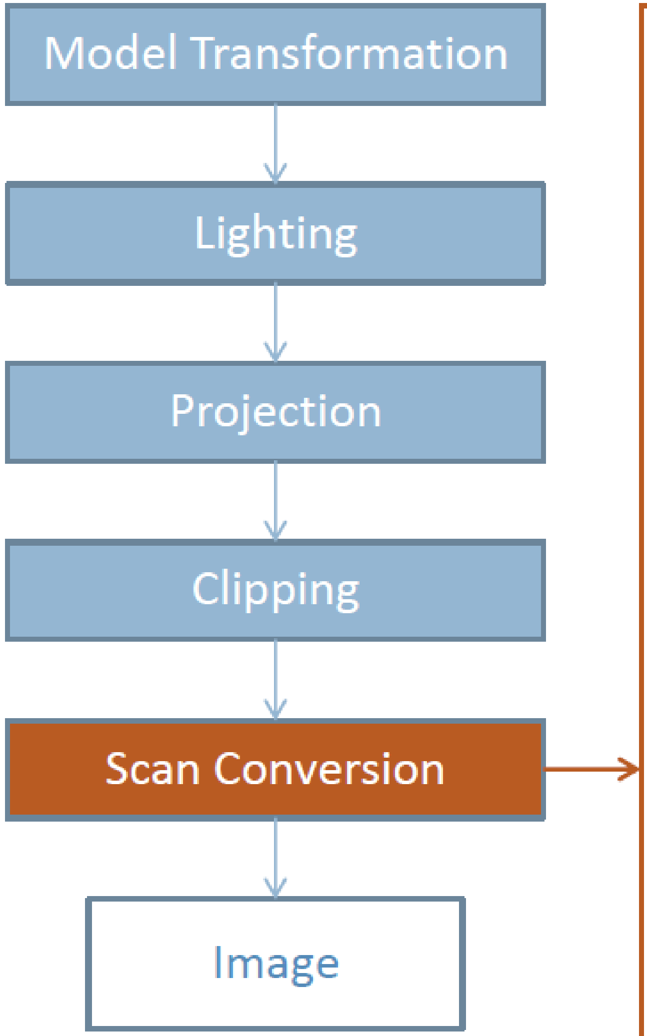
Rasterization – graphics pipeline



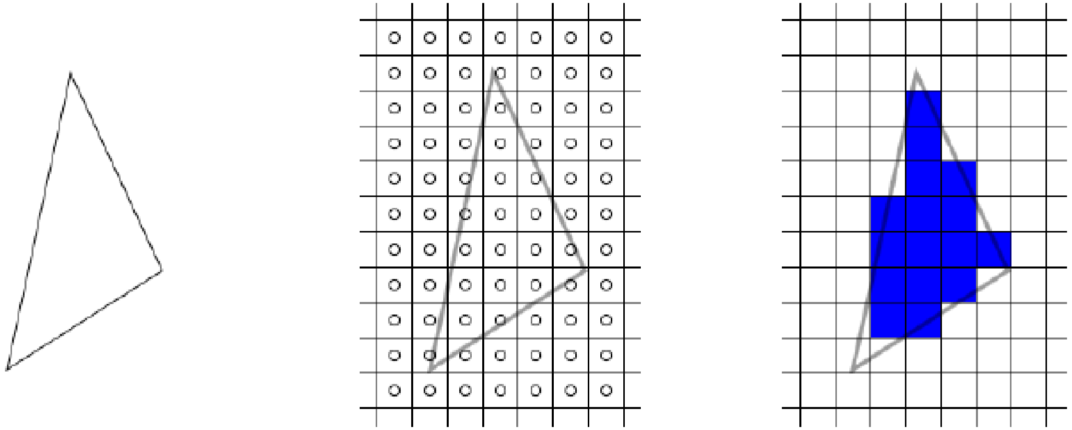
- Remove triangles that fall outside the clipping plane
- Determine boundaries of triangles partially within the clipping plane



Rasterization – graphics pipeline



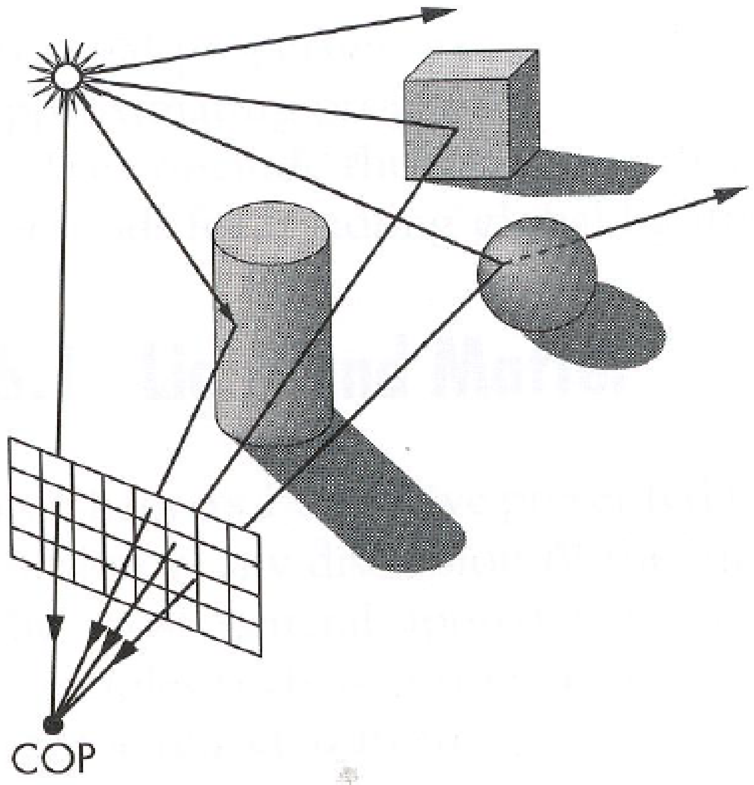
- Drawing the triangles in 2D
- Scanning horizontal scan lines for each triangle



Ray Tracing

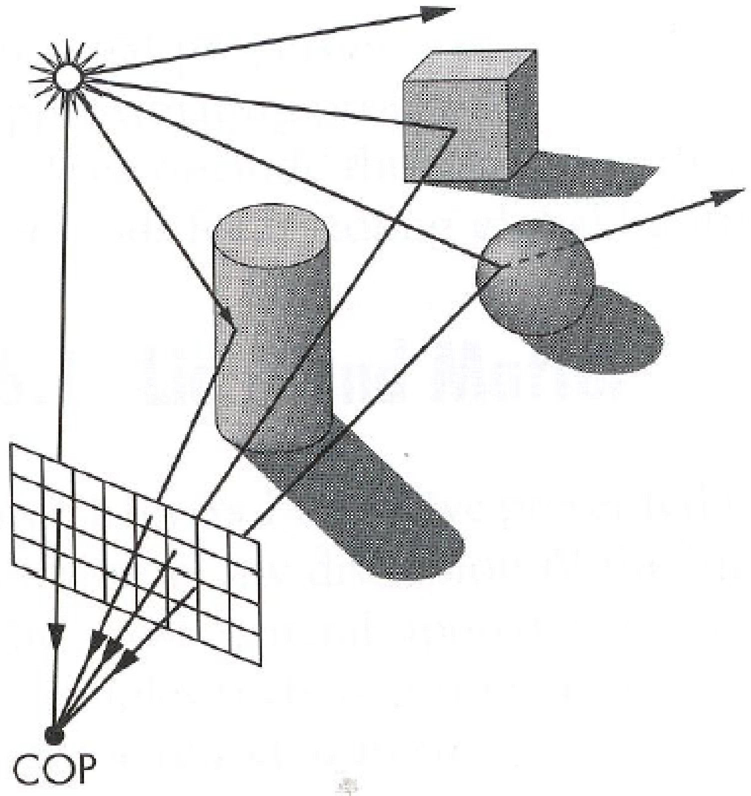
Ray Tracing

- Basic idea: Shoot a "visibility ray" from center of projection (camera) through each pixel in the image and find out where it hits
- This is actually backward tracing – instead of tracing rays **from** the light source, we trace the rays from the viewer back **to** the light source



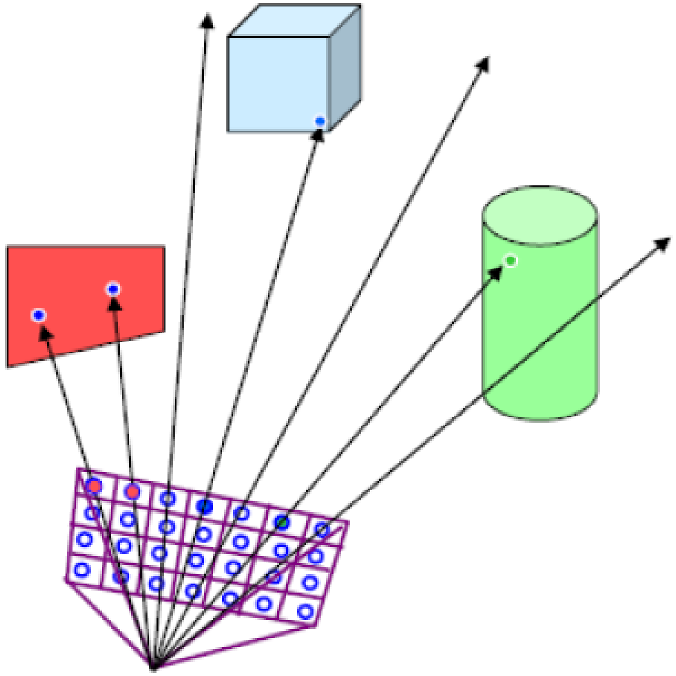
Ray Tracing

- Backward tracing is called **Ray Casting**
- Simple to implement
- For each ray find intersections with every polygon – slow…
- Easy to implement realistic lighting, shadows, reflections and refractions, and indirect illumination



Ray Tracing

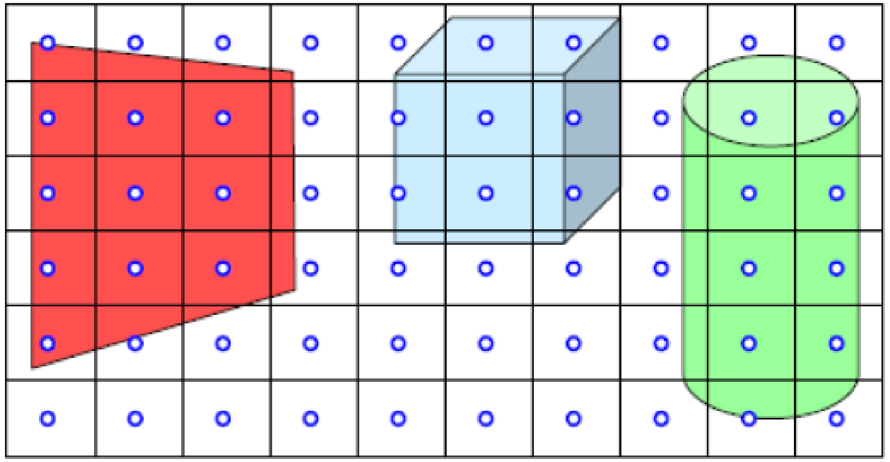
- For each sample (pixel or subpixel):
- Construct a **ray** from eye position through viewing plane



Subpixel rendering is a way to increase the apparent resolution of a computer's liquid crystal display (LCD) or organic light-emitting diode (OLED) display by rendering pixels to take into account the screen type's physical properties.

Ray Tracing

- For each sample (pixel or subpixel):
- Construct a **ray** from eye position through viewing plane
- Find first (closest) surface that intersects the ray



Ray Tracing

Primary rays (*view rays* on image) are emitted from the camera through each pixel of the screen, and then are checked on the intersection with scene geometry.

From each point of intersection, shadow rays are spawn towards each light source (*shadow rays totally dominate*).

If surface of hit point is reflective or/and refractive, secondary rays are spawn (they behave just like primary rays).

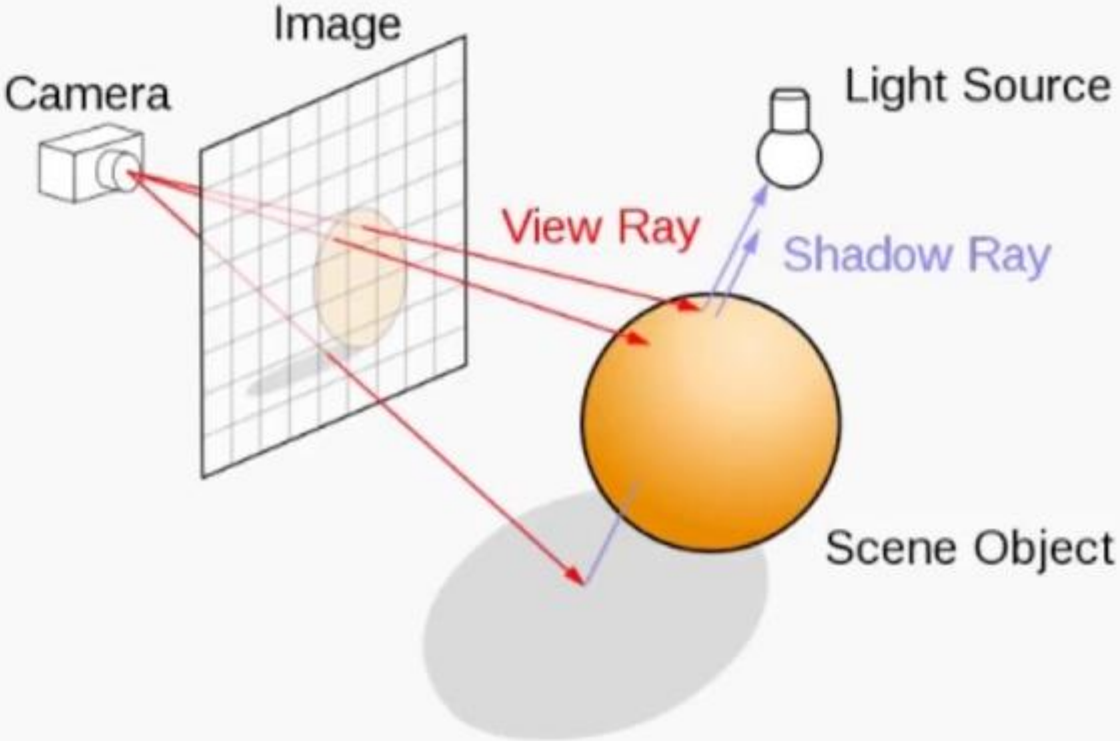
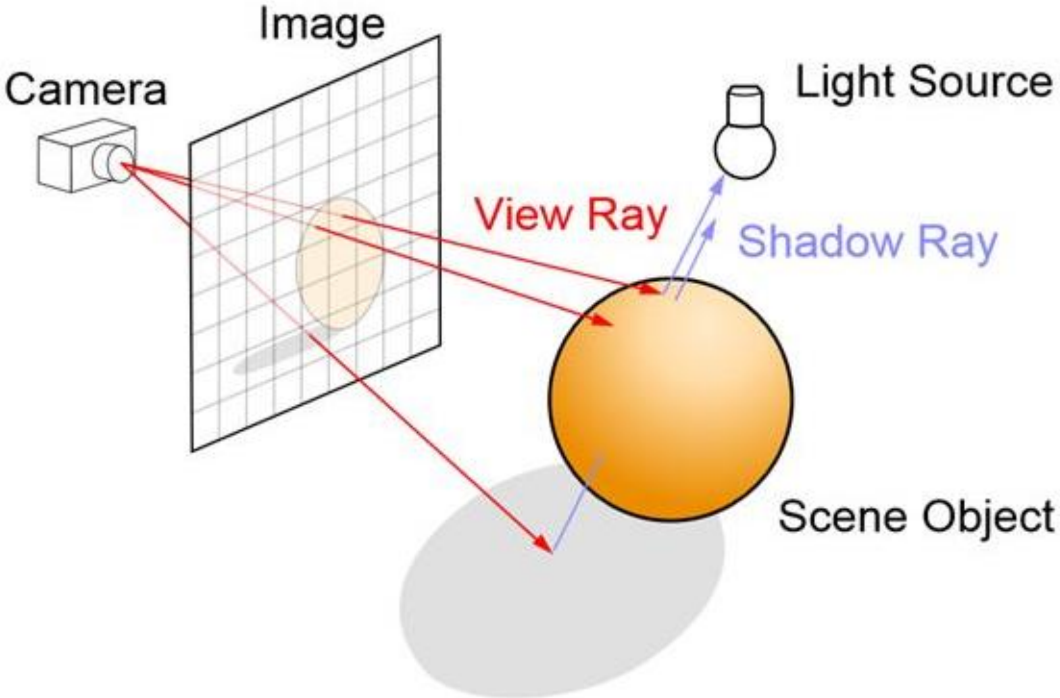


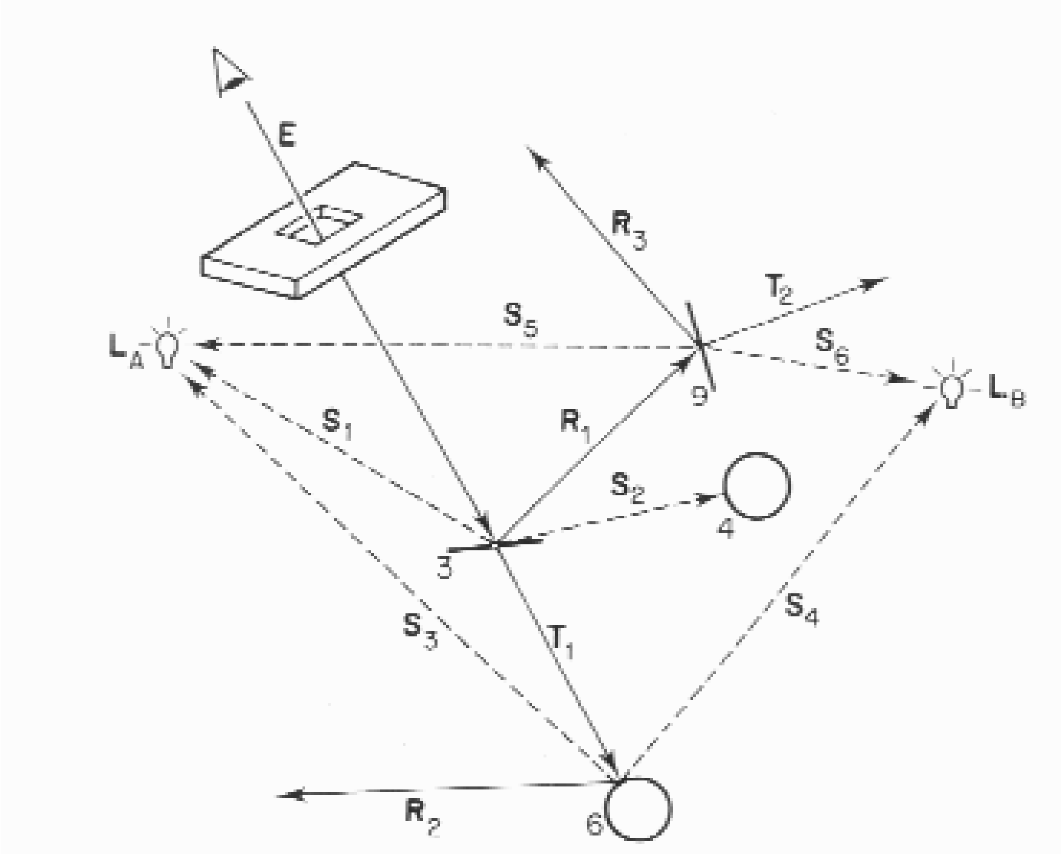
Image from Wikipedia.

Ray Tracing

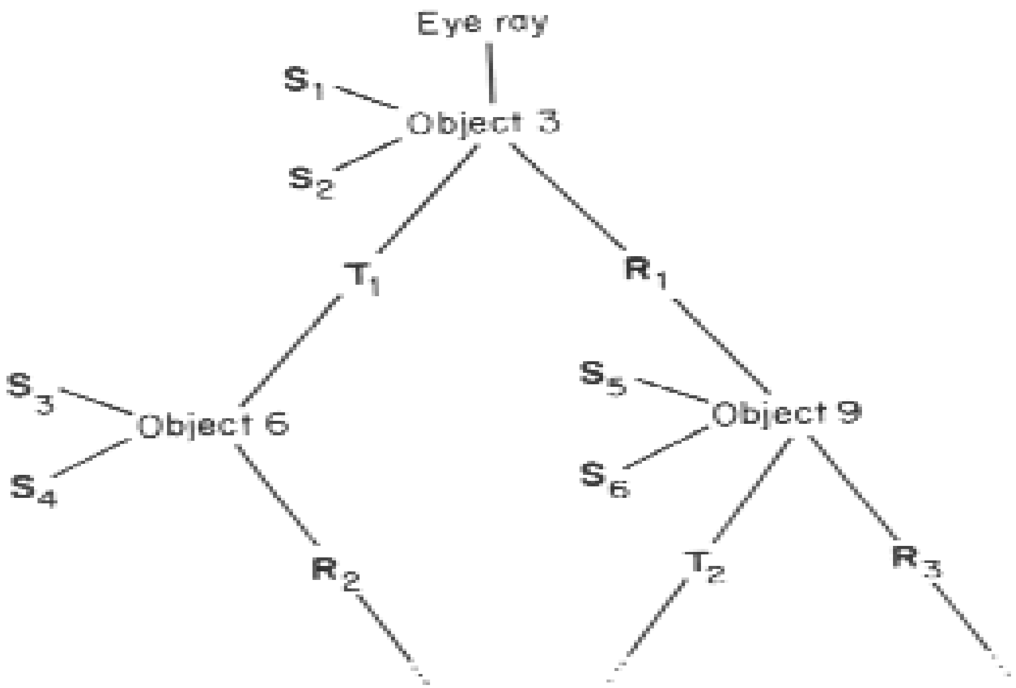
Once the nearest object has been identified, the algorithm will estimate the incoming light at the point of intersection, examine the material properties of the object, and combine this information to calculate the final color of the pixel.



Ray Tracing



Ri – reflected rays
Si – shadow rays



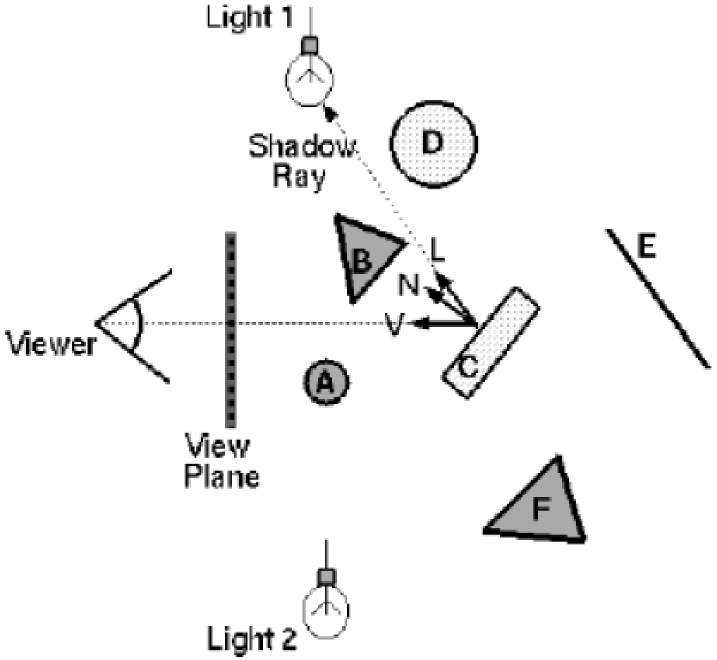
The ray tree in schematic form

Ray Tracing - intersections

- Finding intersections
  - Intersecting spheres
  - Intersecting triangles (polygons)
  - Intersecting other primitives
  - Finding the closest intersection in a group of objects / all scene

Ray Tracing – computing color and shadows

- Shadow term tell which light source are blocked
- $S_L = 0$ if ray is blocked,
  $S_L = 1$ otherwise
- Direct illumination is only calculated for unblocked lights



- Illumination formula:

$$\boxed{I = I_E + K_A I_A + \sum_L (K_D(N \bullet L) + K_S(V \bullet R)^n) S_L I_L}$$

Shadow term