Theoretical Backgrounds of Audio & Graphics

Graphics Rendering Basic Intro

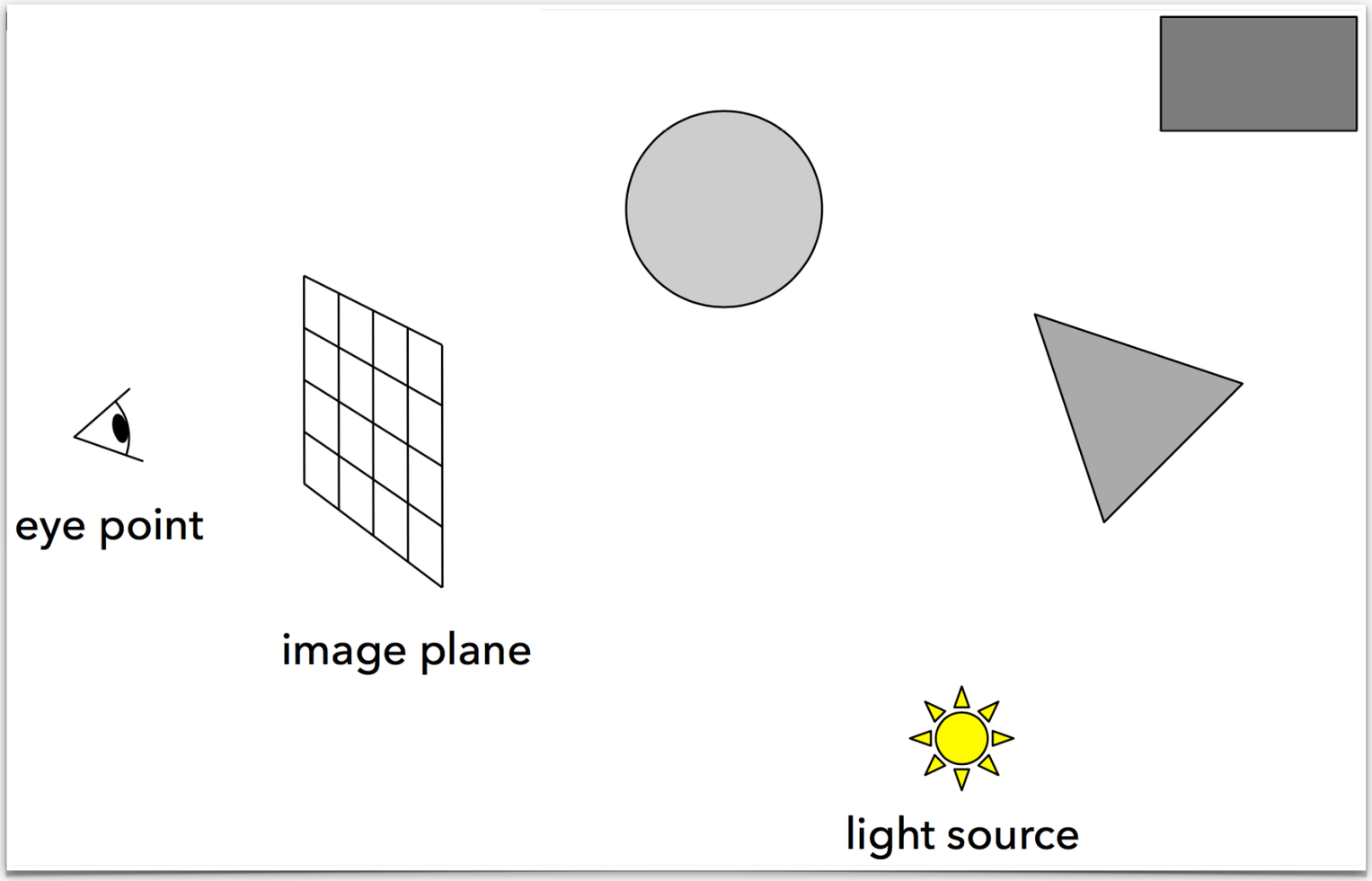
Angela Brennecke | Prof. Dr.-Ing.
Audio & Interactive Media Technologies

Filmuniversität Babelsberg
KONRAD WOLF

Winter term 2020/2021

Graphics Rendering

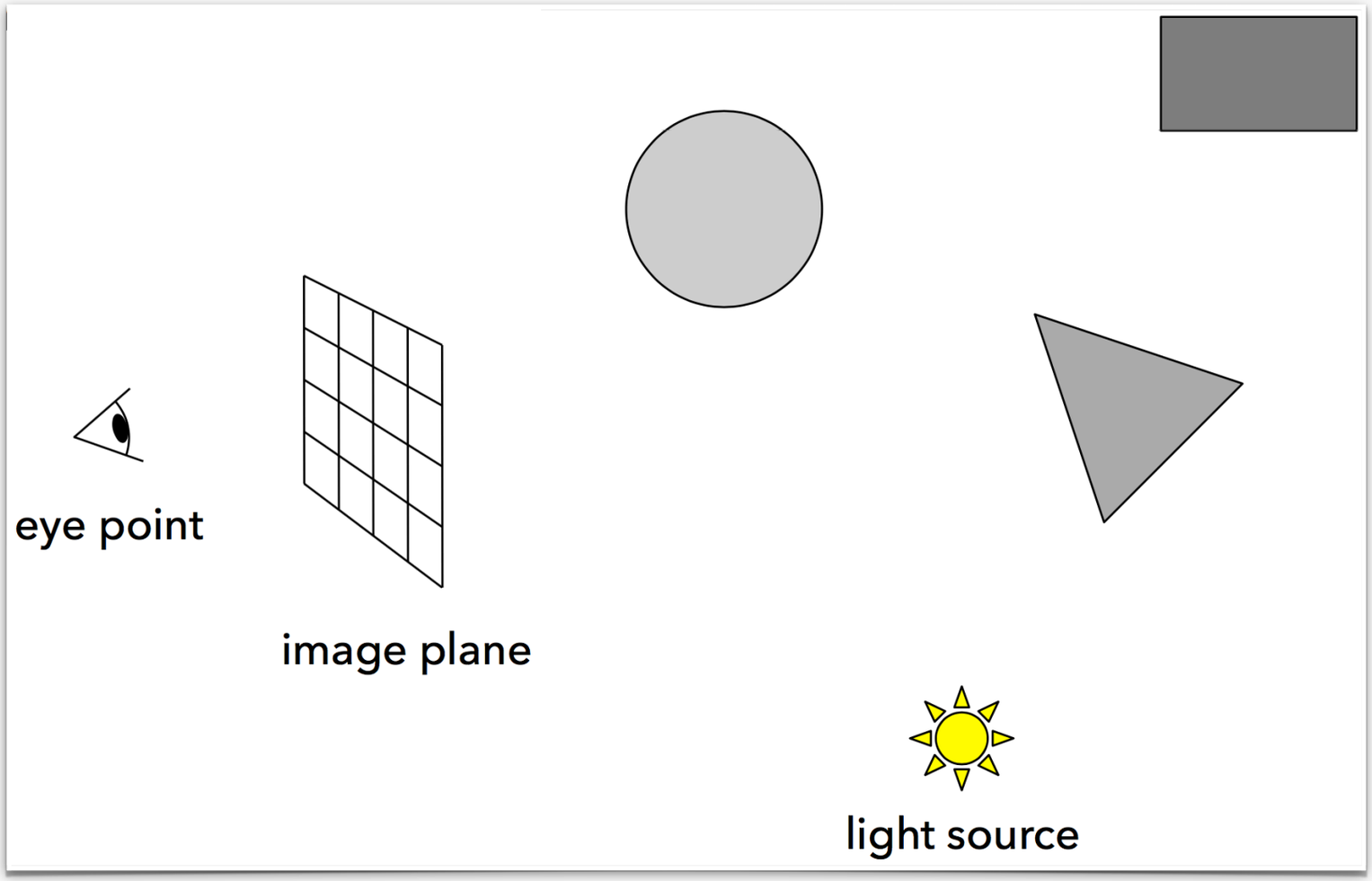
- Rendering describes the process of generating a 2D image from a given 3D geometric scene using a camera & illumination model



Jarosz, W., Computer Graphics, Fall 2016, Dartmouth College

Graphics Rendering

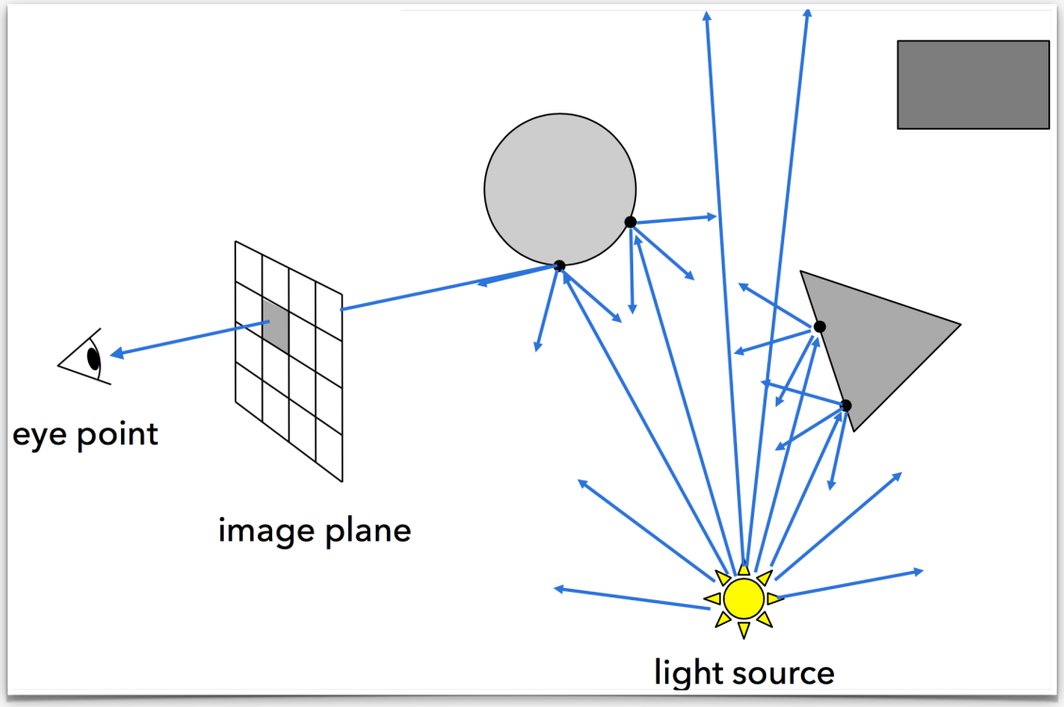
- Main goal is to simulate a realistic scene illumination and a realistic representation of objects regarding their visibility, textures or shadows



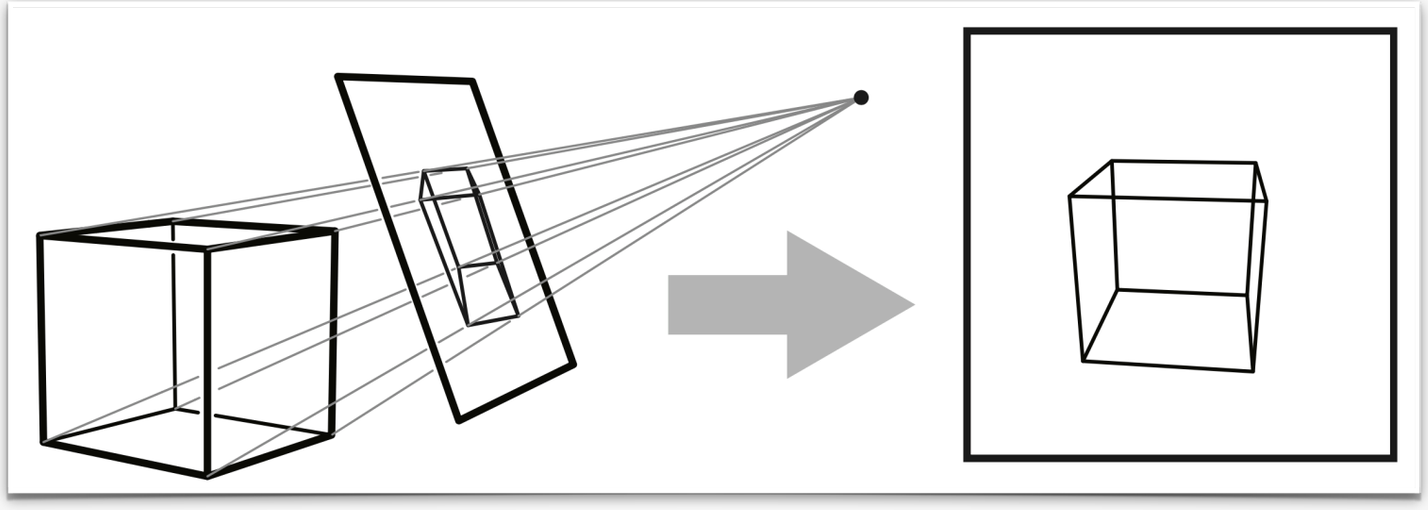
Jarosz, W., Computer Graphics, Fall 2016, Dartmouth College

Graphics Rendering

- Two central approaches for rendering:
  ray tracing and **rasterization**



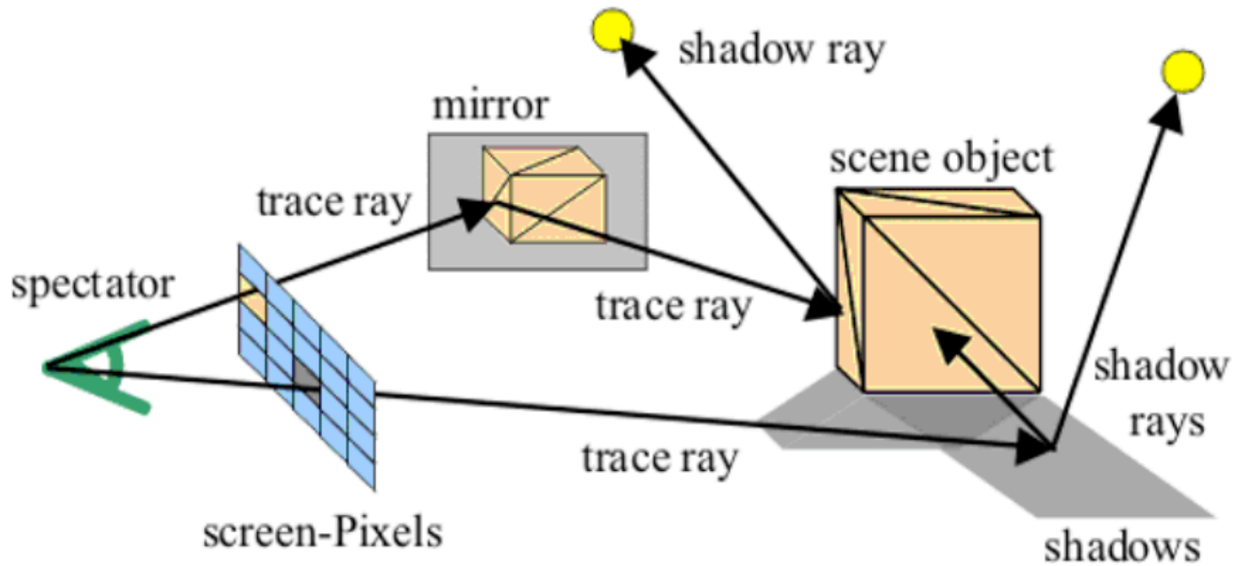
Jarosz, W., Computer Graphics, Fall 2016, Dartmouth College



Marschner S., Computer Graphics, Spring 2018, Cornell University

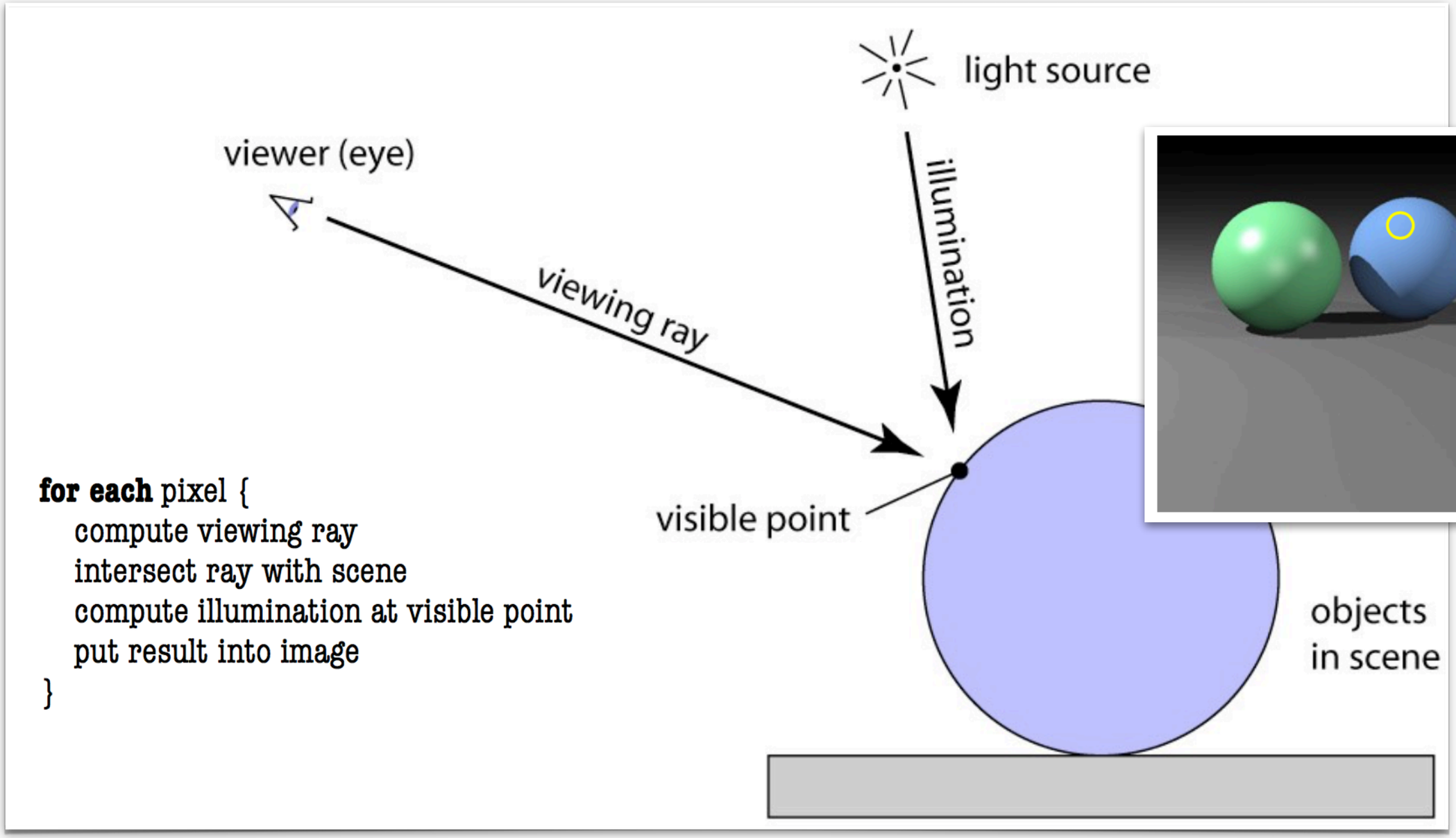
Ray Tracing

- For every point on the image plane,
  - cast a ray from the eye into the scene,
  - determine color and brightness of the light traveling to the eye, and
  - store it in the image.



Gieseke, L. Mathematics for Audio & Graphics, WS17/18, Film University

Ray Tracing



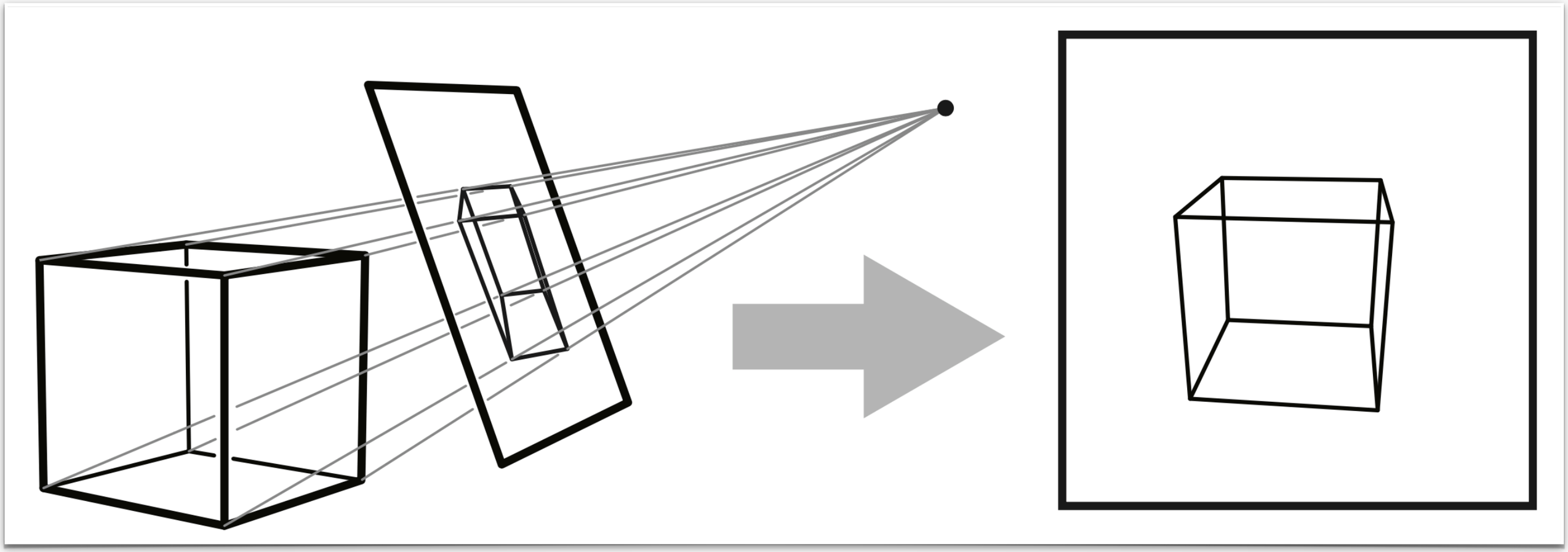
Marschner S., Computer Graphics, Spring 2018, Cornell University

Ray Tracing

- Pro
  - Can render everything that can be intersected with a ray
  - Supports parallel computation & recursion (every pixel can be computed independently, and needs to be written to only once)
- Con
  - Requires entire scene description in memory at once
  - Hard to implement in hardware (changing due to general purpose GPUs)
  - So far, still not supporting interactive applications well

After slides from Gieseke, L. Mathematics for Audio & Graphics, WS17/18, Film University & Jarosz, W., Computer Graphics, Fall 2016, Dartmouth College

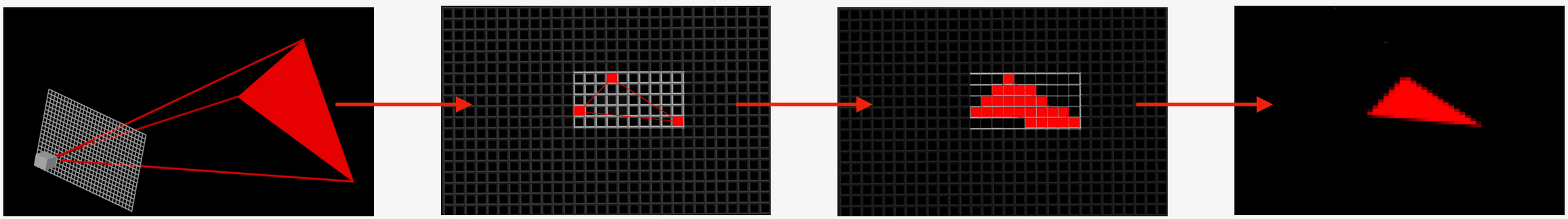
Rasterization



Marschner S., Computer Graphics, Spring 2018, Cornell University

Rasterization

- Rasterization iterates over all 3D objects (i.e., triangle meshes) and determines which pixel the individual triangles covers
- Instead of tracing a ray through the scene, 3D points are **projected** onto the image plane and their contribution is being evaluated and processed further

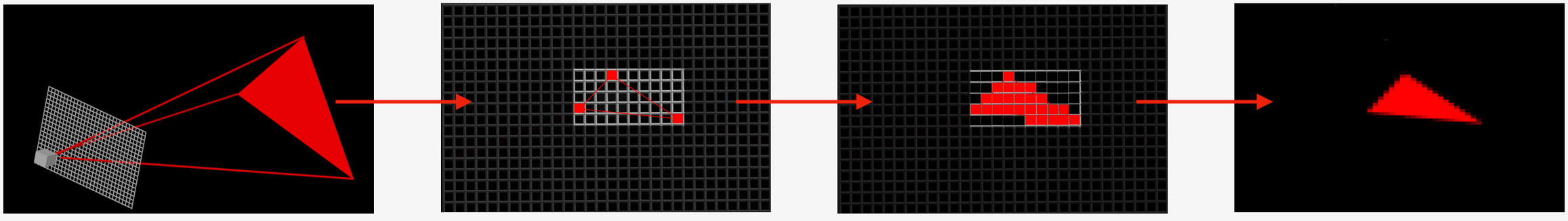


https://filmmakeriq.com/lessons/rasterization/

Rasterization

- In principle, all computations are ***local*** to the primitive being rendered
  - Almost every illumination effect (direct illumination, cast shadows, soft shadows, caustics, reflections, lens effects, motion blur, global illumination) needs its own solution
- Physical correctness is almost impossible to achieve

Gieseke, L. Mathematics for Audio & Graphics, WS17/18, Film University



https://filmmakeriq.com/lessons/rasterization/

Rasterization

- Pro
  - Well supported in hardware (GPU's do rasterization)
  - Rasterizer only needs one triangle at a time, *plus* the entire image and depth information per pixel
  - Supports interactive & real-time rendering
- Con
  - Restricted to triangle meshes (scan-convertible primitives)
  - Shading artifacts due to All computations are executed per objects, i.e., no unified solution for shadows, transparency, etc.

After slides from Gieseke, L. Mathematics for Audio & Graphics, WS17/18, Film University & Jarosz, W., Computer Graphics, Fall 2016, Dartmouth College

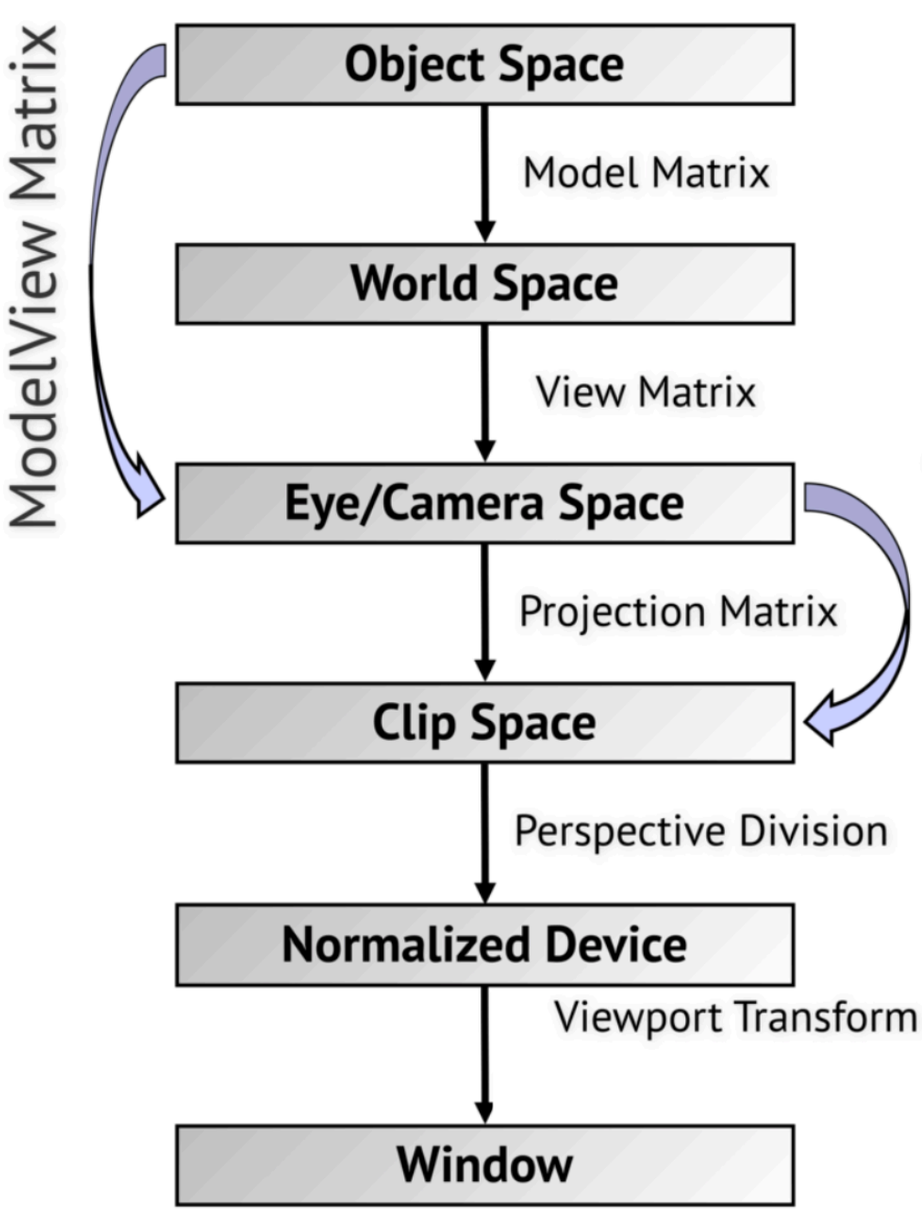
Ray Tracing and Rasterization

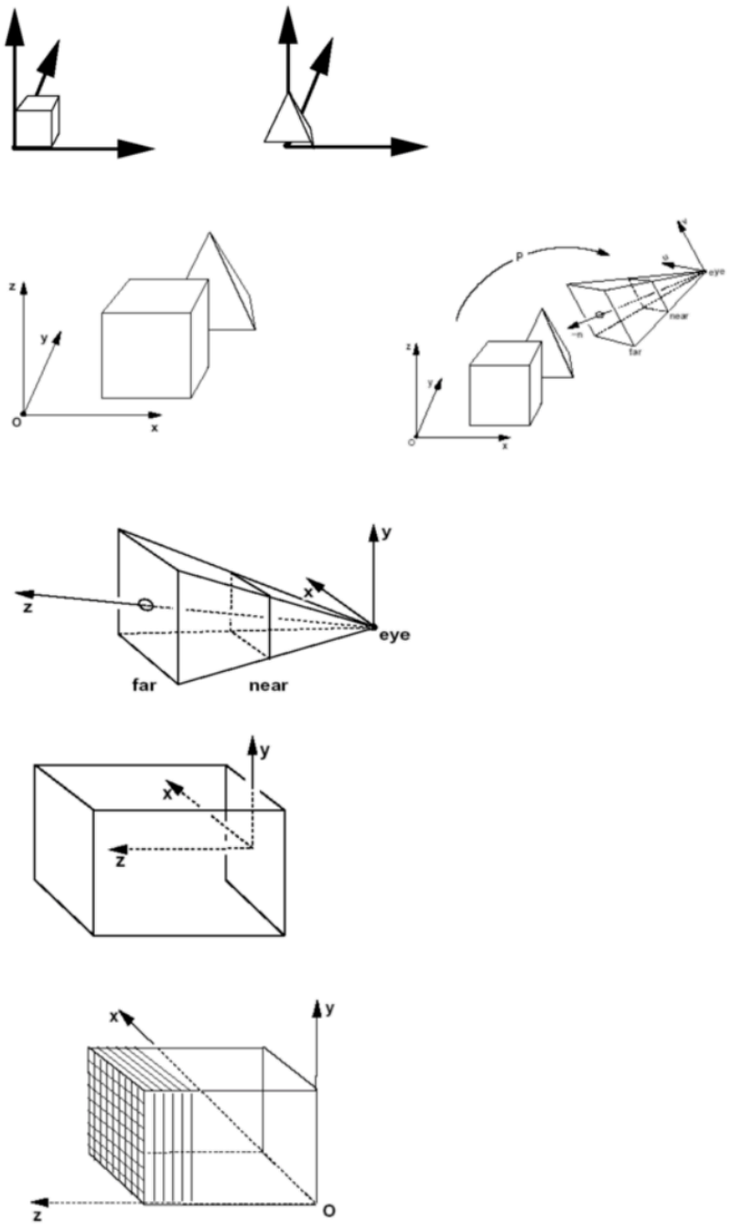
- Most common approaches to rendering are **image-order** (i.e., ray tracing) and **object-order** (i.e., rasterization) based

- Image-order
  - Iterate over the pixels and decide which image value (“color”, “brightness” etc.) it should have
- Object-order
  - Iterate over the scene objects primitives (usually, a collection of primitives such as triangles) and determine their contribution to the output image

Gieseke, L. Mathematics for Audio & Graphics, WS17/18, Film University

The Graphics Pipeline

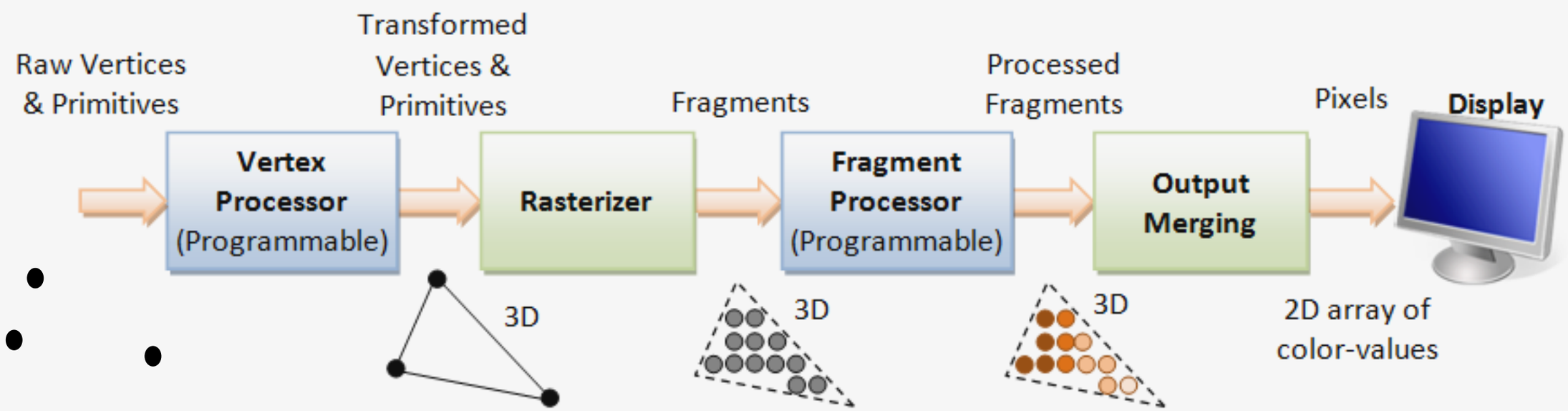




[Image source: [Image Synthesis Class 20xx]][Image Synthesis, M. Fuchs, 2016.]

Pipeline Stages

- The **graphics** or **rendering pipeline** is the standard approach to object-order **rasterization** and describes the **steps** or **stages** required to render a 3D scene into a 2D image



http://www.ntu.edu.sg/home/ehchua/programming/opengl/cg_basicstheory.html

References

- Buss, S. (2003): **3D Computer Graphics—A Mathematical introduction with OpenGL**. Cambridge University Press, New York, NY, USA.
- Shiffman, D. (2012): **Nature of Code.** https://natureofcode.com/book/

- Lecture slides
  - Gieseke, L. Mathematics for Audio & Graphics, 2017/18, Film University
  - Jarosz, W., Computer Graphics, Fall 2016, Dartmouth College
  - Marschner S., Computer Graphics, Spring 2018, Cornell University