Introduction to Computer Graphics (CS360A)

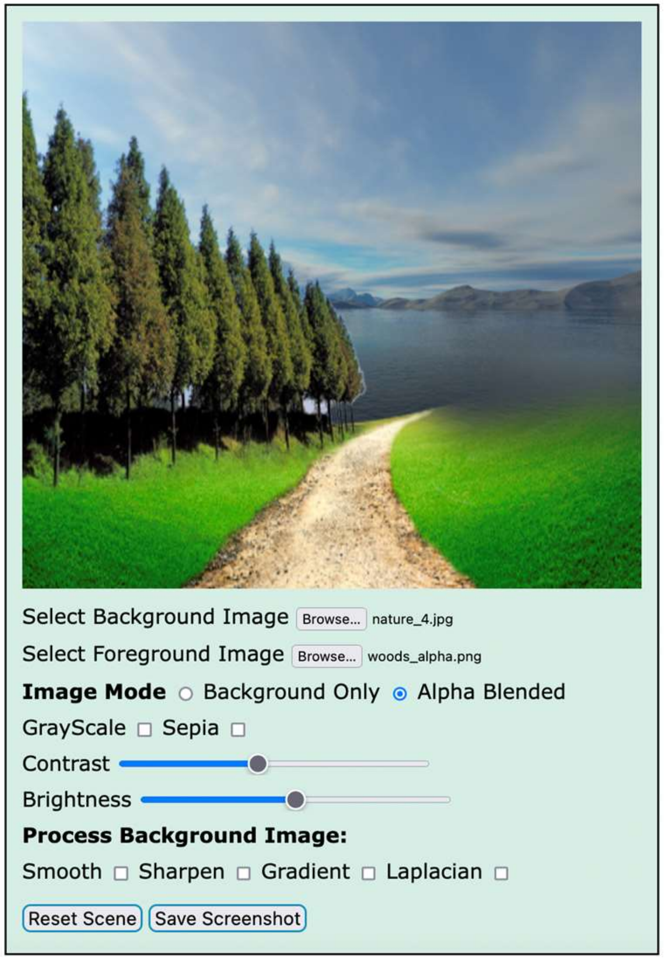
Instructor: Soumya Dutta

Department of Computer Science and Engineering

Indian Institute of Technology Kanpur (IITK)

email: soumyad@cse.iitk.ac.in

Assignment 4: Image Processing App in Shader



Select Background Image nature_4.jpg

Select Foreground Image woods_alpha.png

Image Mode ○ Background Only ● Alpha Blended

GrayScale □ Sepia □

Contrast

Brightness

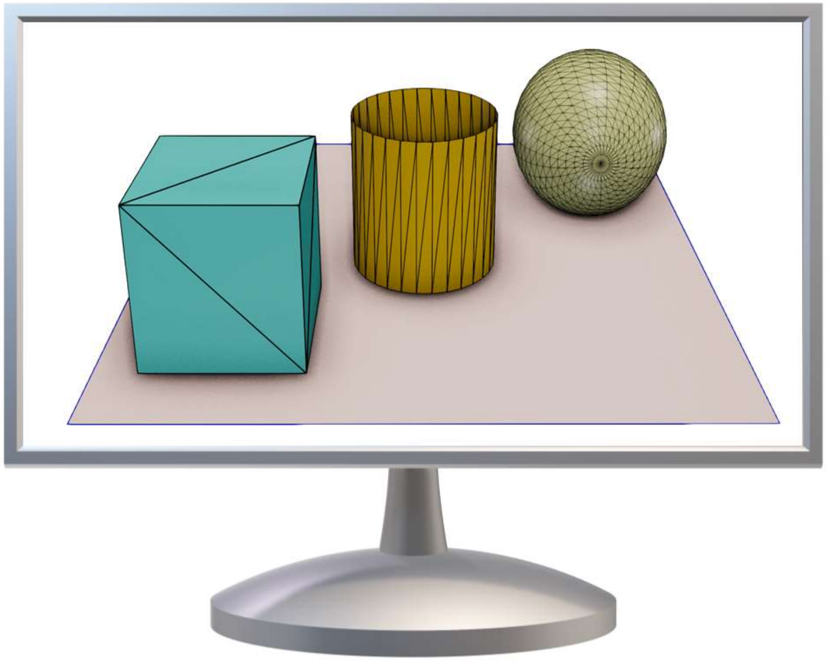
Process Background Image:

Smooth □ Sharpen □ Gradient □ Laplacian □

Due date: Oct 22, 11:59pm

No extension!

Rendering Algorithms



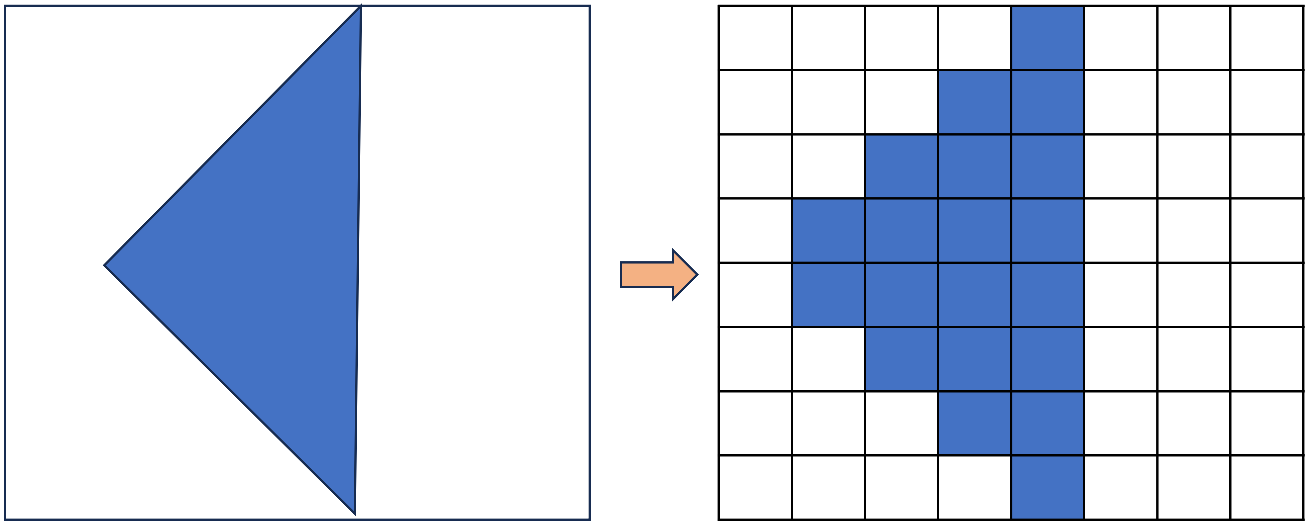
- We want to generate raster images on the computer screen
- Rendering algorithms are primarily concerned about which object fills up which pixel on the screen
  - Which triangle of the object covers which pixels
- Then we can do shading and illumination

Rendering Algorithms

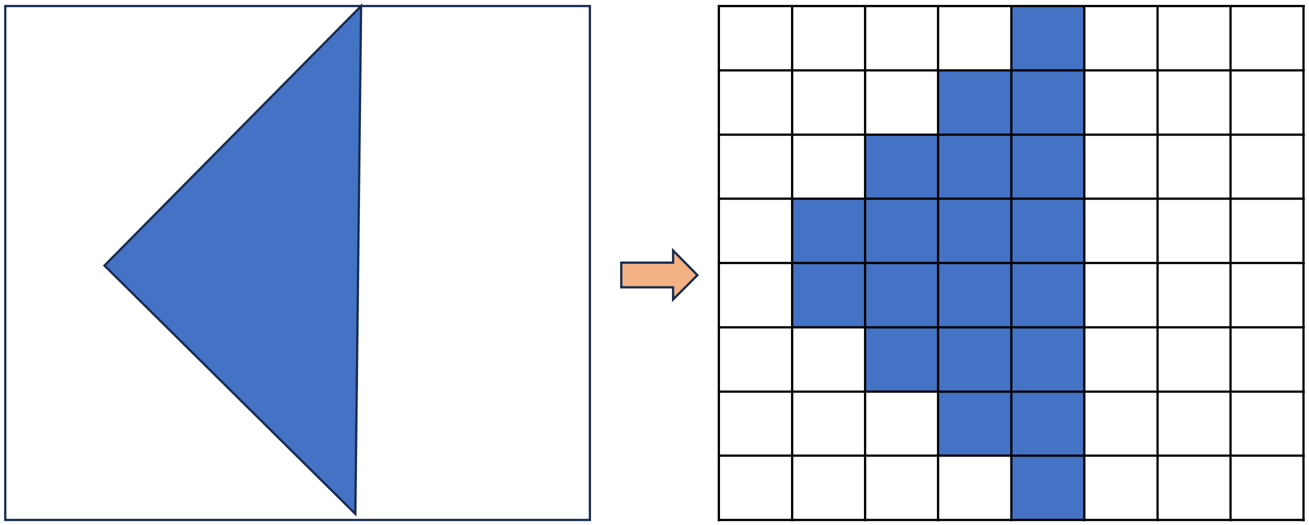
- Rasterization
  - Painter’s Algorithm
  - Z-buffer Algorithm
  - A-buffer algorithm
  - ….
- Ray Tracing
  - Ray casting
  - Path tracing
  - Photon mapping
  - …..

Rasterization

- We have been using rasterization so far in this course to generate images using 2D/3D graphics
- Rasterization is actually done by the GPU for us



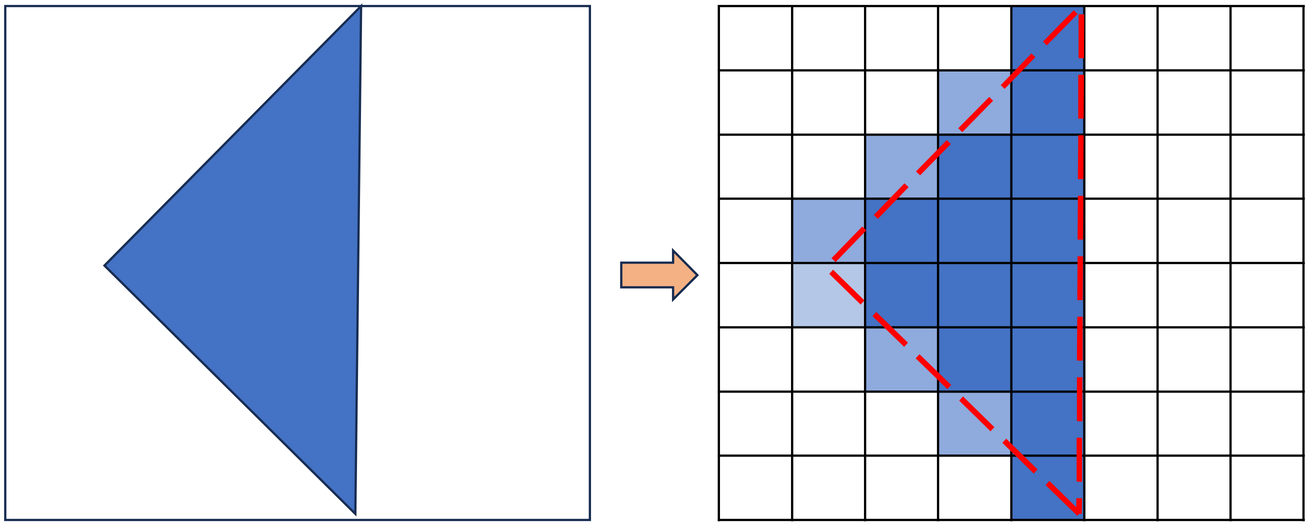
Rasterization



Canonical view volume space

Raster Image

Rasterization



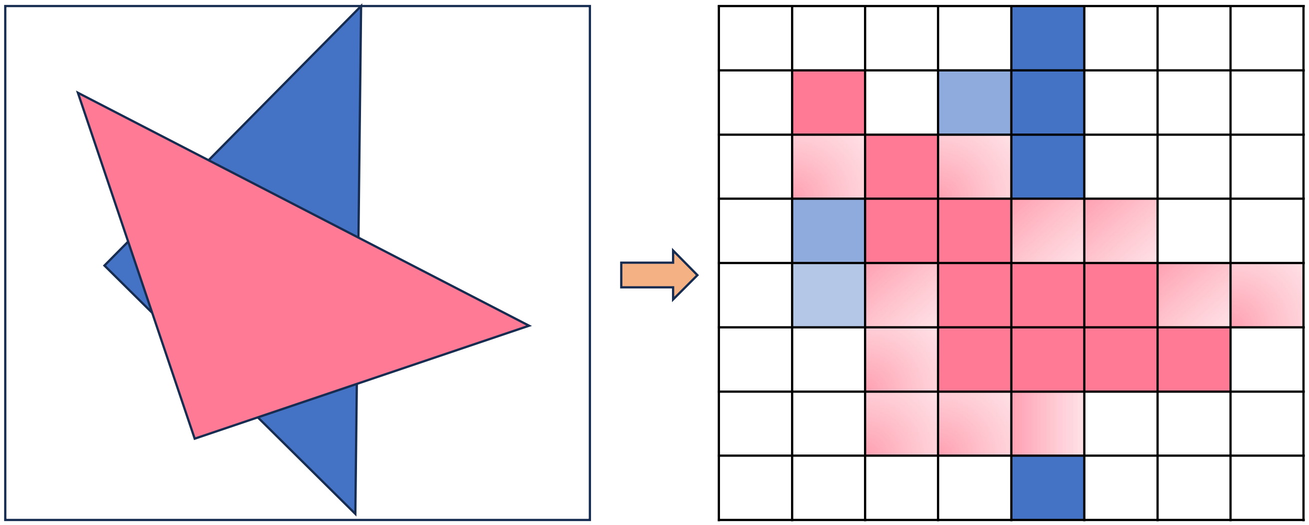
Canonical view volume space

Raster Image

(With anti-aliasing)

The key thing to note is that the process is trying to figure out what percentage of the triangle is covering the pixel

Rasterization



Canonical view volume space

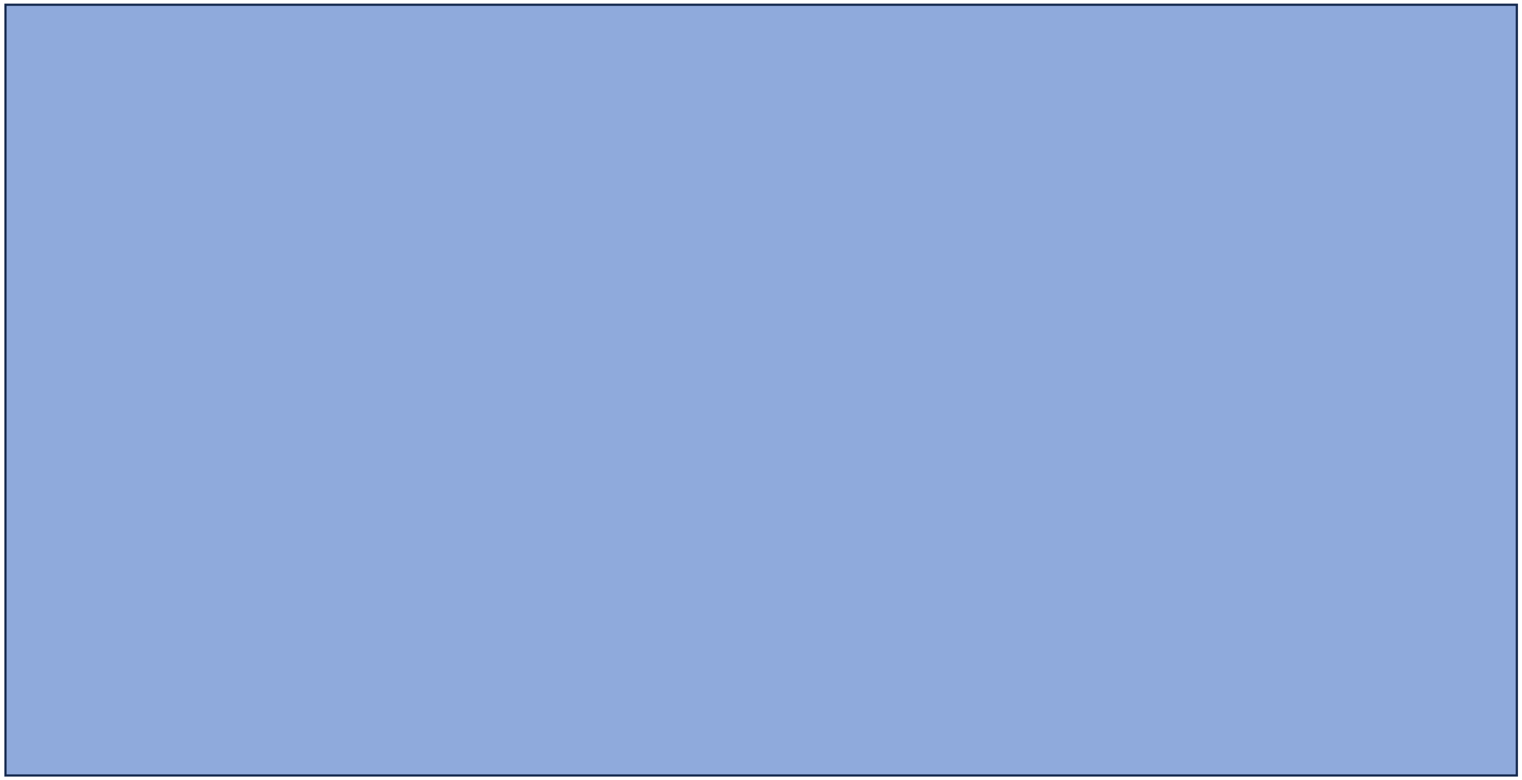
Raster Image

(With anti-aliasing)

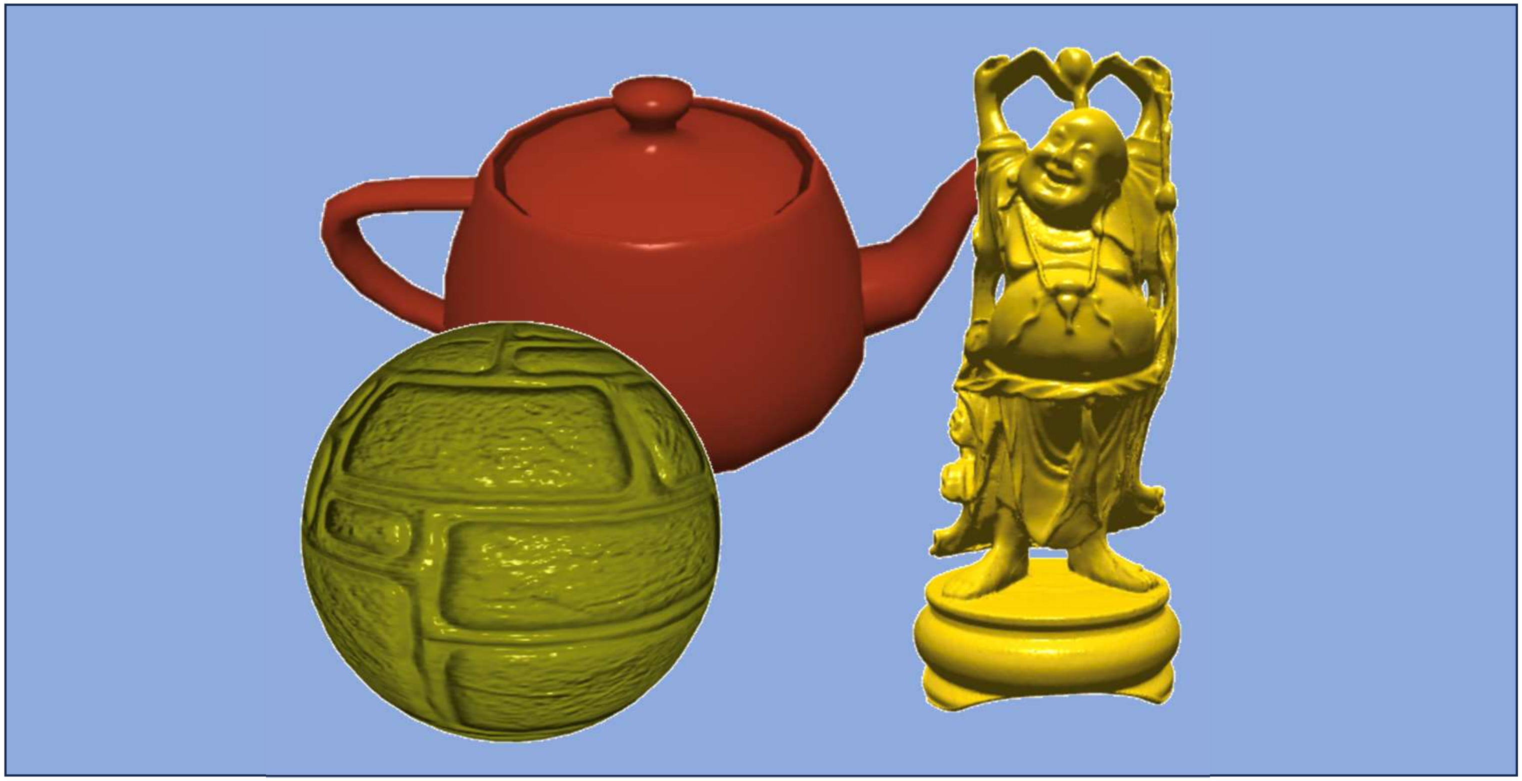
The key thing to note is that the process is trying to figure out what percentage of the triangle is covering the pixel

Painter’s Algorithm

Painter’s Algorithm

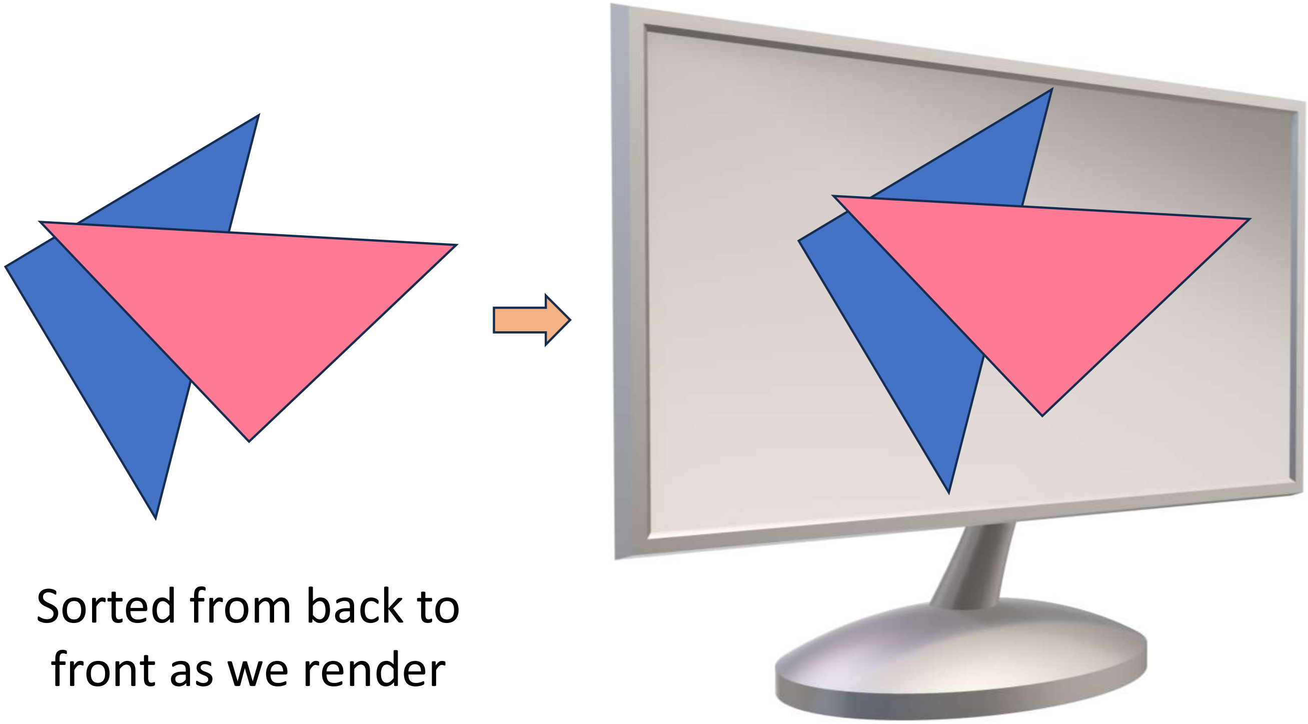


Painter’s Algorithm



Rendering back to front

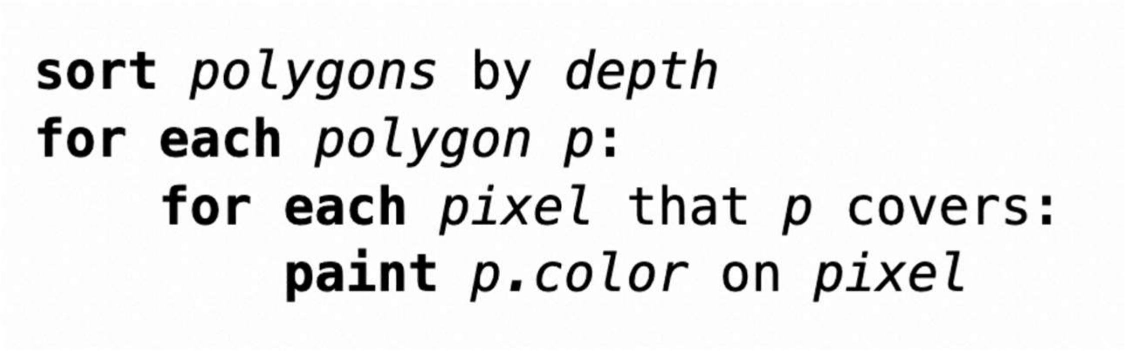
Painter’s Algorithm



Sorted from back to front as we render

Sorting is tricky with a computational cost

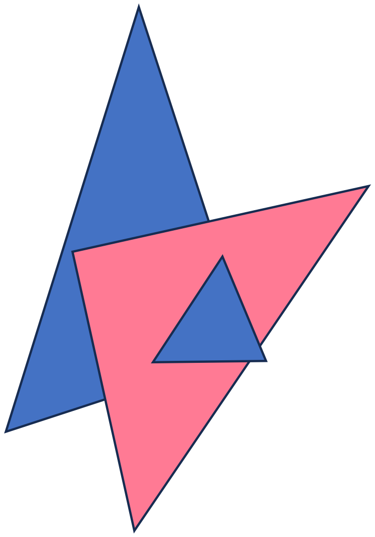
Painter’s Algorithm

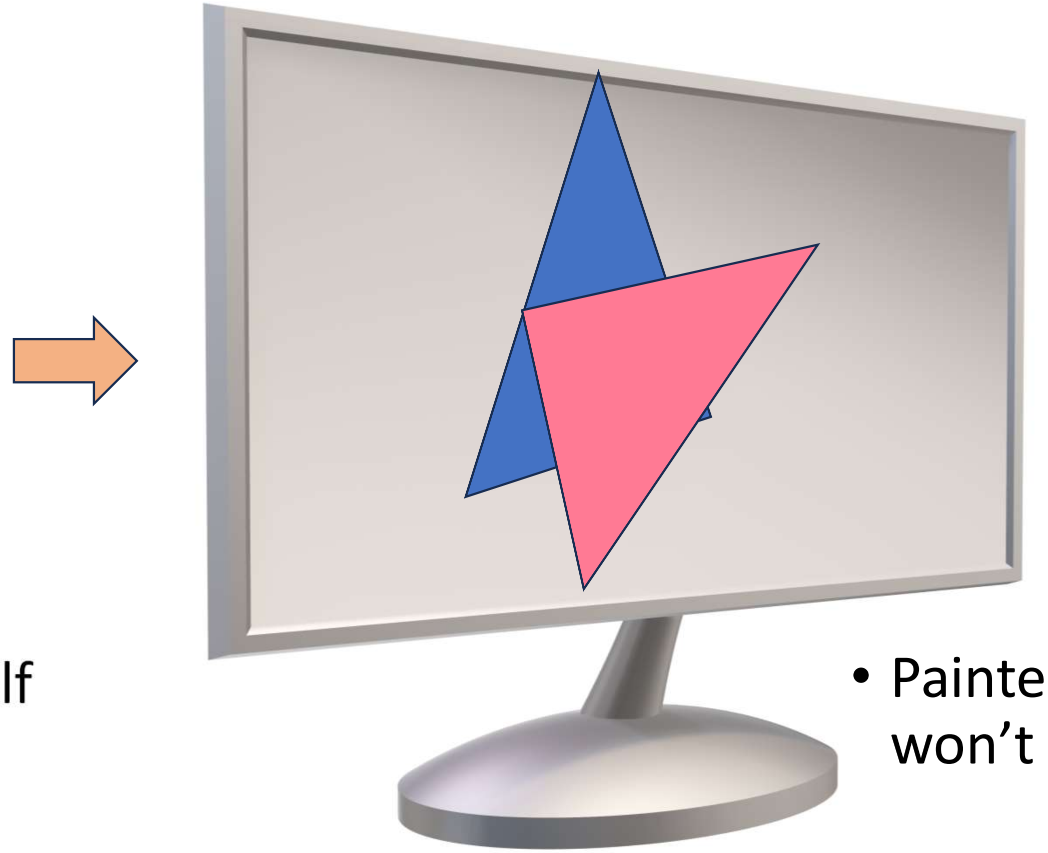


```
sort polygons by depth
for each polygon p:
    for each pixel that p covers:
        paint p.color on pixel
```

- Needs sorting mechanism of triangles from front to back or back to front
- Cannot handle intersecting geometry correctly and may produce incorrect/non-physical images on the screen

Painter’s Algorithm





- How about dealing with Self intersecting triangles?

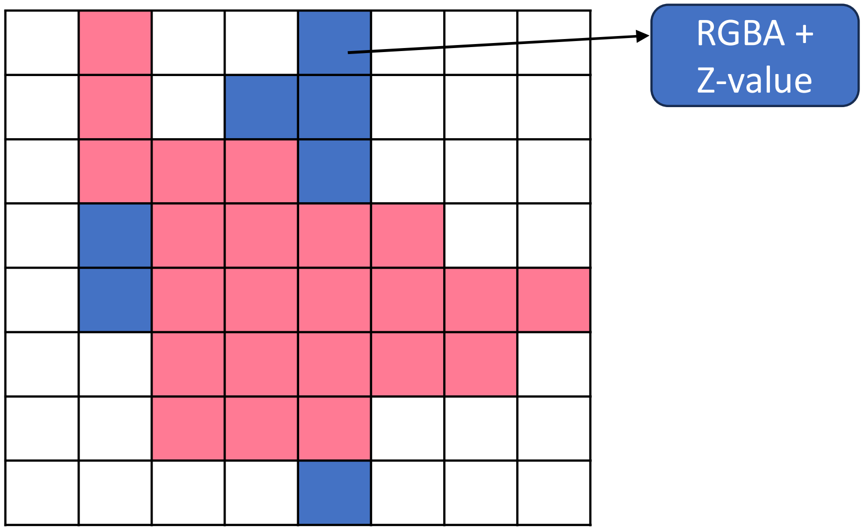
- Painter’s algorithm won’t work

Z-Buffer Algorithm

- The most popular and most frequently used rendering algorithm in computer graphics to produce raster images on screen
- All GPUs use this method
  - GPU rasterizer uses it after vertex shader
- Everything we see on the screen uses this method

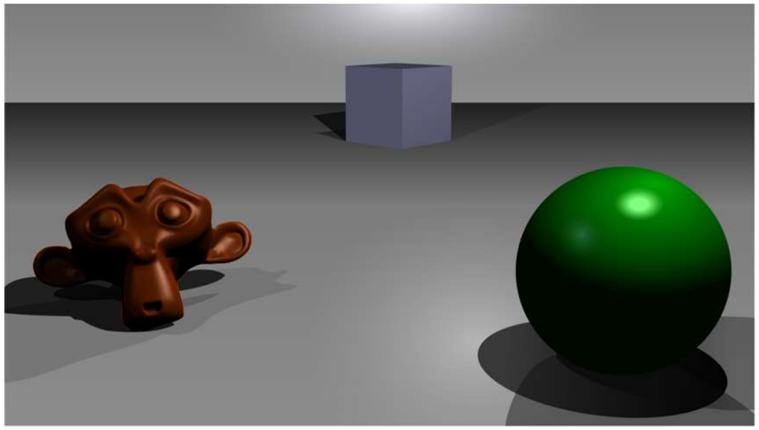
Z-Buffer Algorithm

- We will store the depth value at each pixel in depth buffer
  - Distance from the screen
- No sorting is required
- During rendering we can look up the depth value and then shade the pixel accordingly depending on who is the nearest object

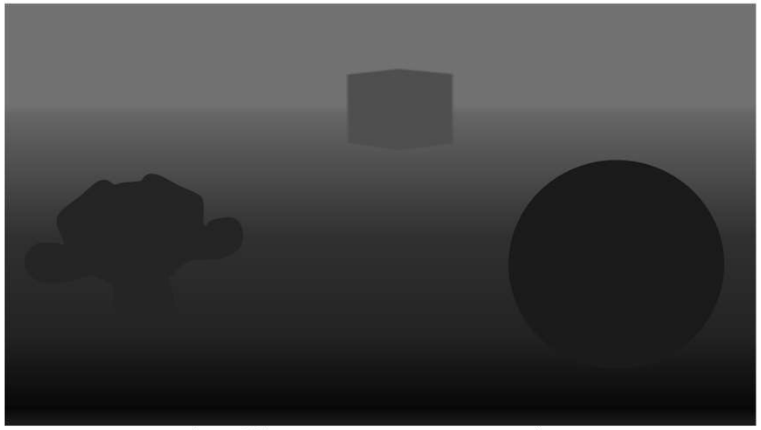


Z-Buffer Algorithm

- We will store the depth value at each pixel in depth buffer
    - Distance from the screen
- No sorting is required
- During rendering we can look up the depth value and then shade the pixel accordingly depending on who is the nearest object



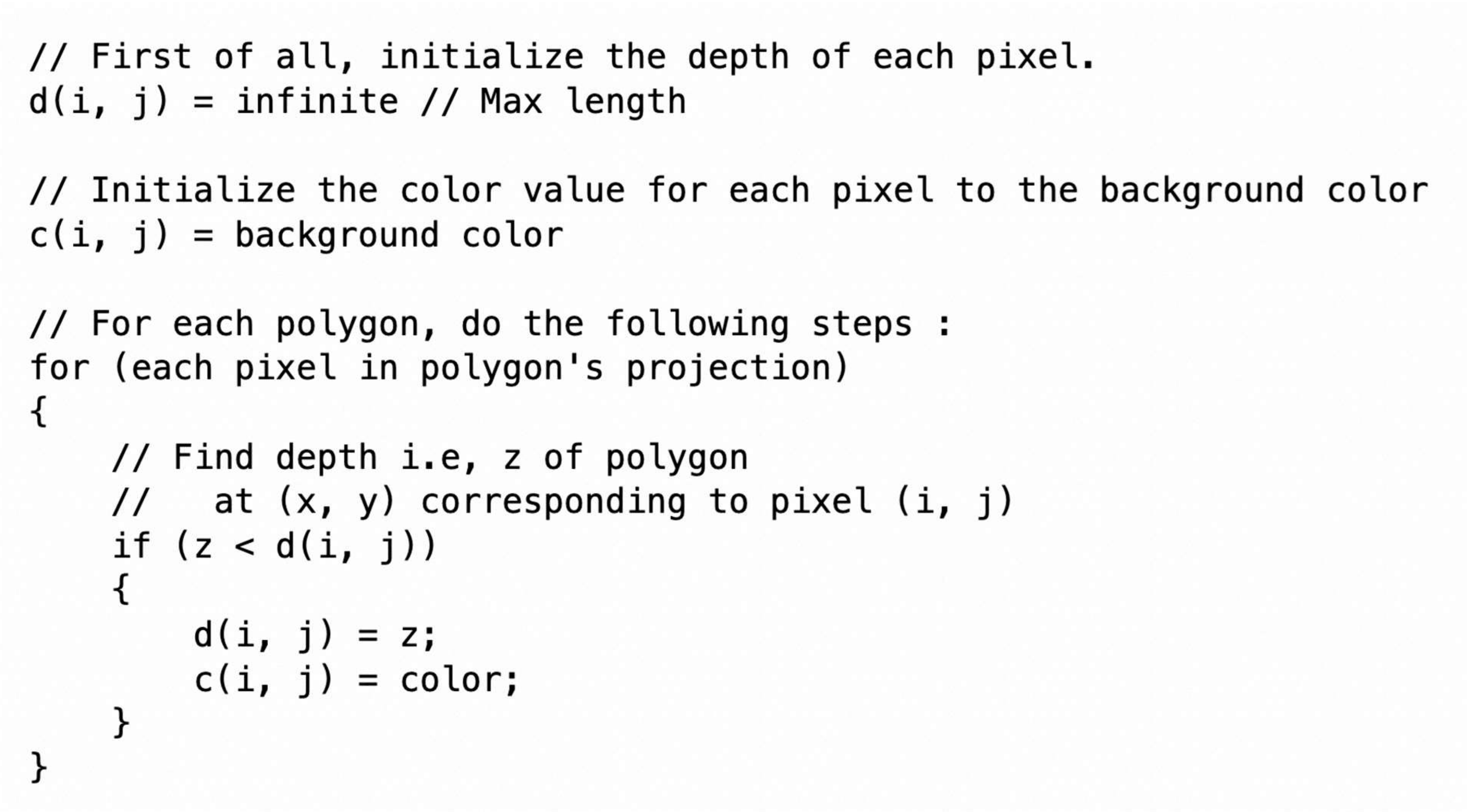
A simple three-dimensional scene



Z-buffer representation

Wikipedia

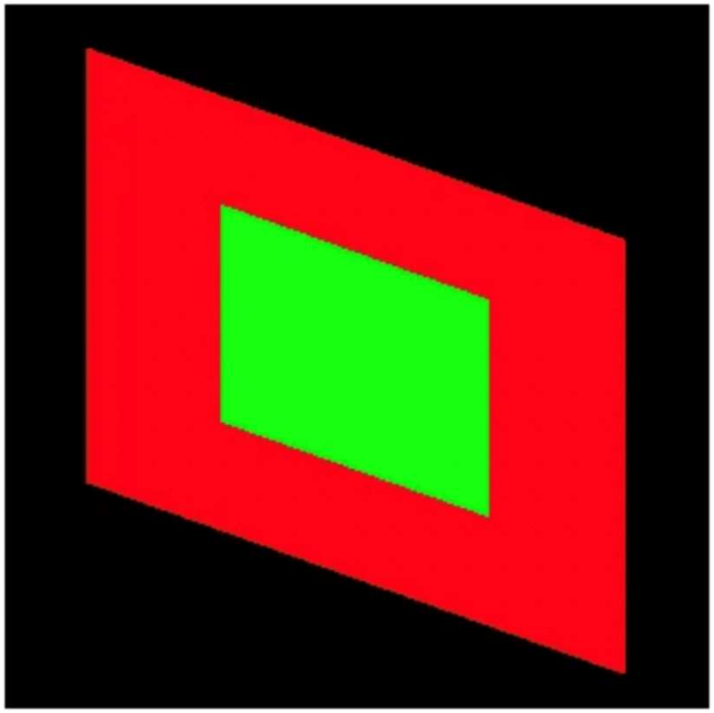
Z-Buffer Algorithm



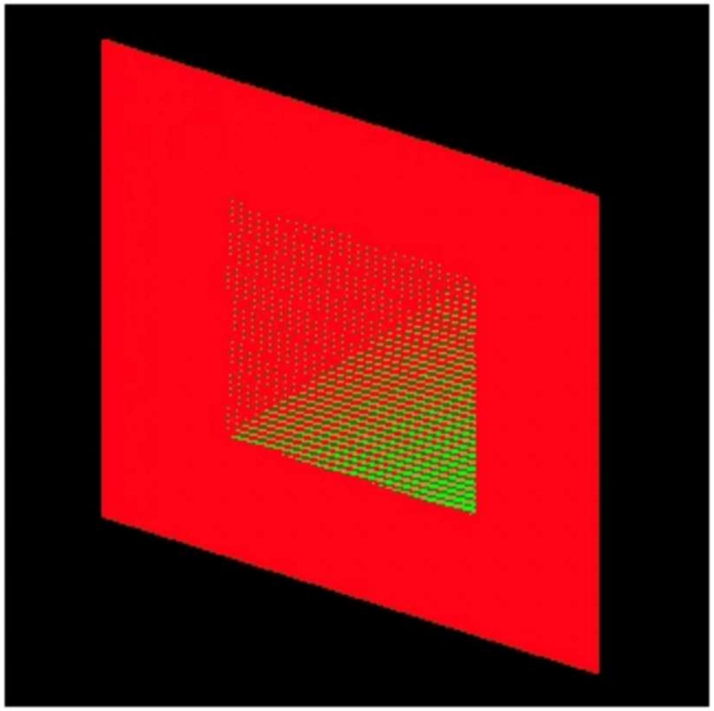
```
// First of all, initialize the depth of each pixel.
d(i, j) = infinite // Max length

// Initialize the color value for each pixel to the background color
c(i, j) = background color

// For each polygon, do the following steps :
for (each pixel in polygon's projection)
{
    // Find depth i.e, z of polygon
    //   at (x, y) corresponding to pixel (i, j)
    if (z < d(i, j))
    {
        d(i, j) = z;
        c(i, j) = color;
    }
}
```

Wikipedia

Z-Fighting



A simple scene



But sometimes we see this

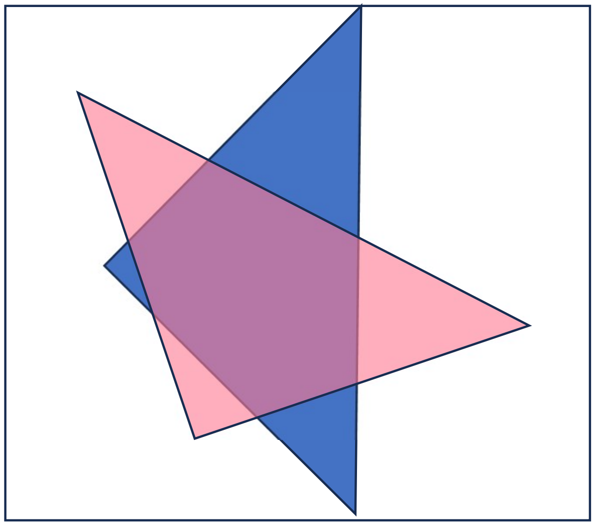
What is happening?

Z-Fighting

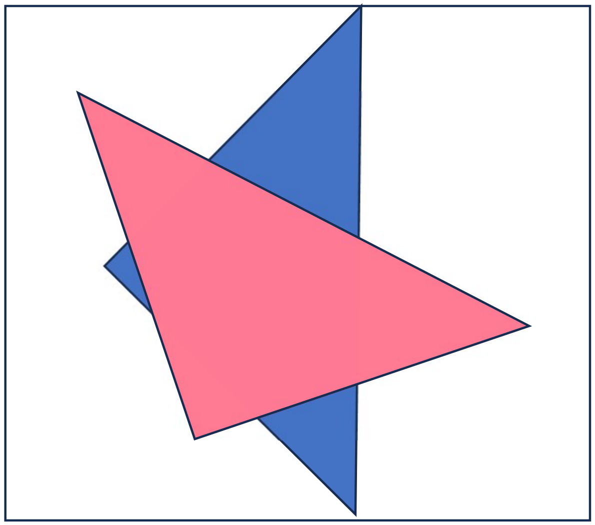
- When two or more primitives have almost same distance from the camera, their z values are also identical
- How do we select which primitive to use for a particular pixel?
- Prevalent for coplanar polygons
- Moving camera may result in flickering as some primitive wins/loses a pixel
- Resolution:
  - Use high-precession, high-resolution z-buffer
  - Apply a post-transformation screen space z-buffer offset

Z-Buffer Algorithm

- Can handle intersecting geometry
- Needs sorting for rendering semi-transparent objects



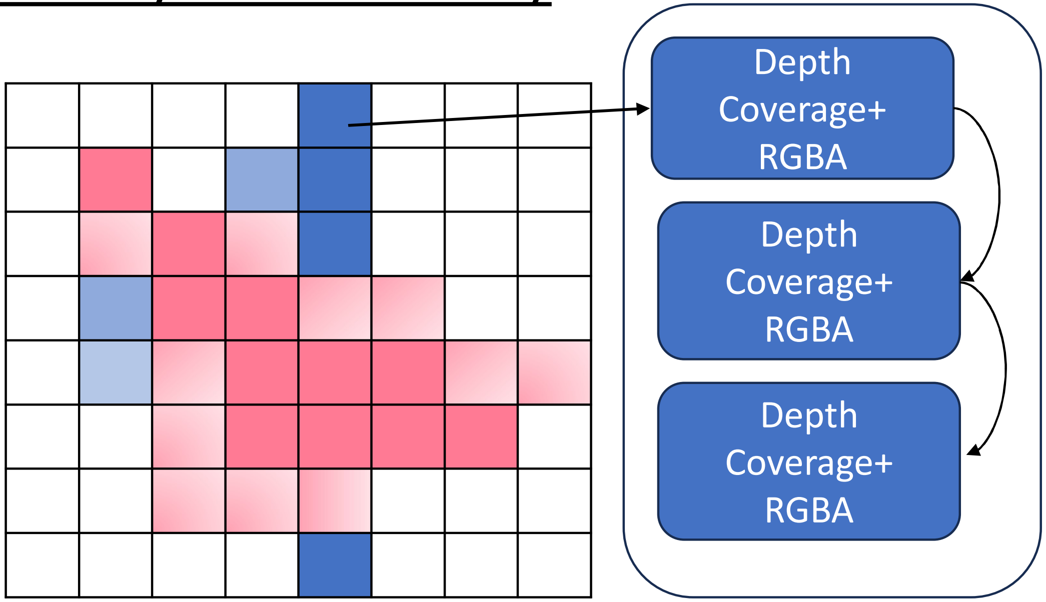
Back to front works fine



Front to back will have issues

A-Buffer Algorithm

- Can handle intersecting geometry
- Sorting independent rendering is possible
- Require additional dynamic memory



IITK CS360A

Ray Tracing

Rasterization vs Ray Tracing

- **Rasterization**

  For each primitive

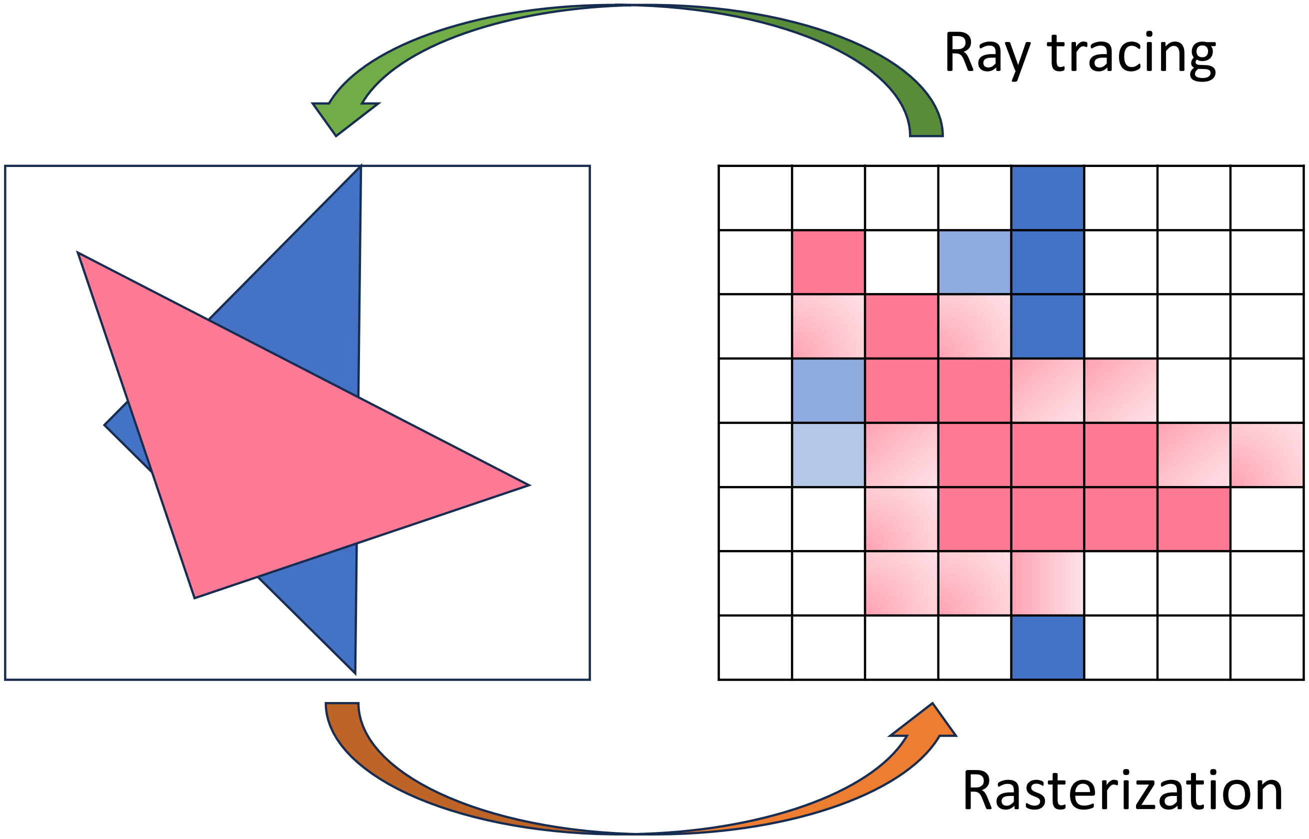
  Find pixels/fragments

- **Ray Tracing**

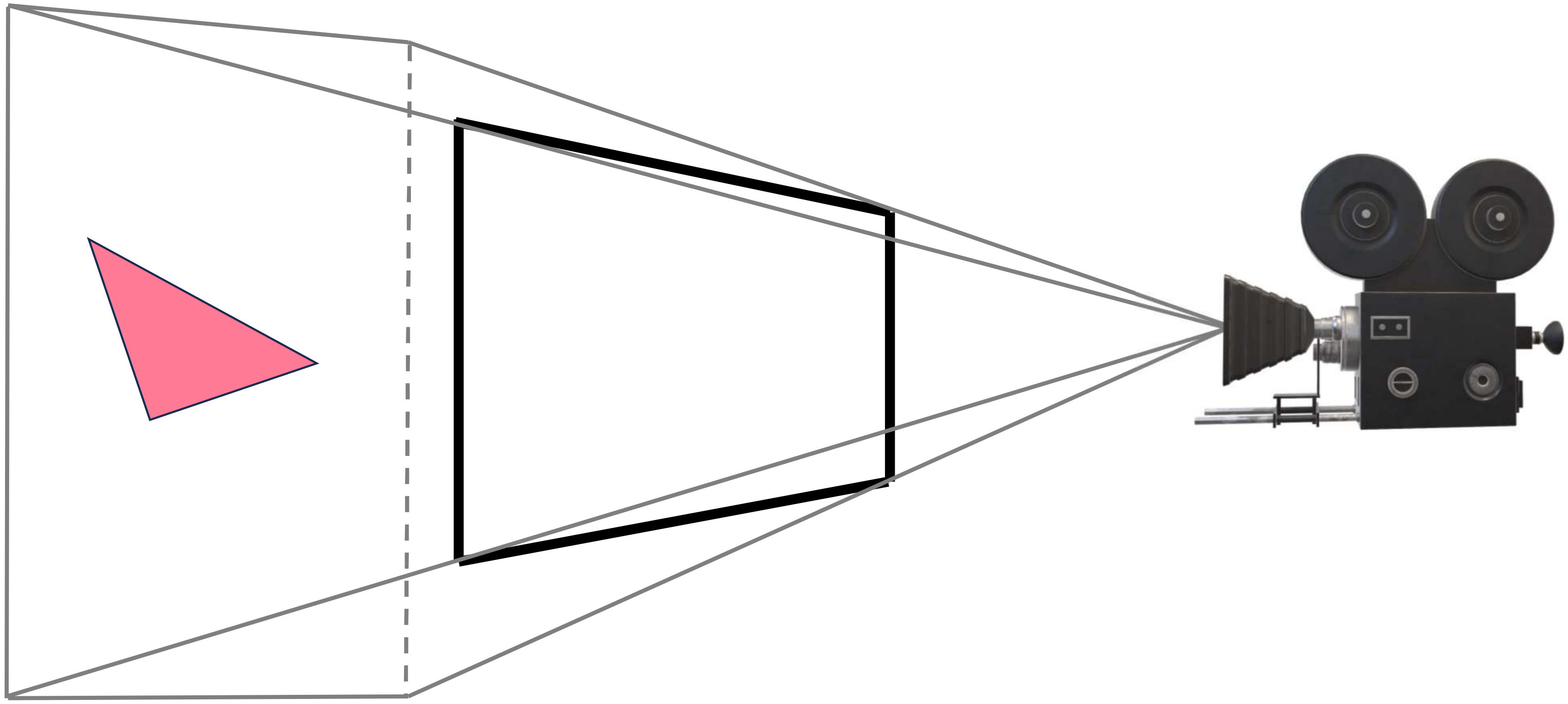
  For each pixel/fragments

  Find the closest primitive

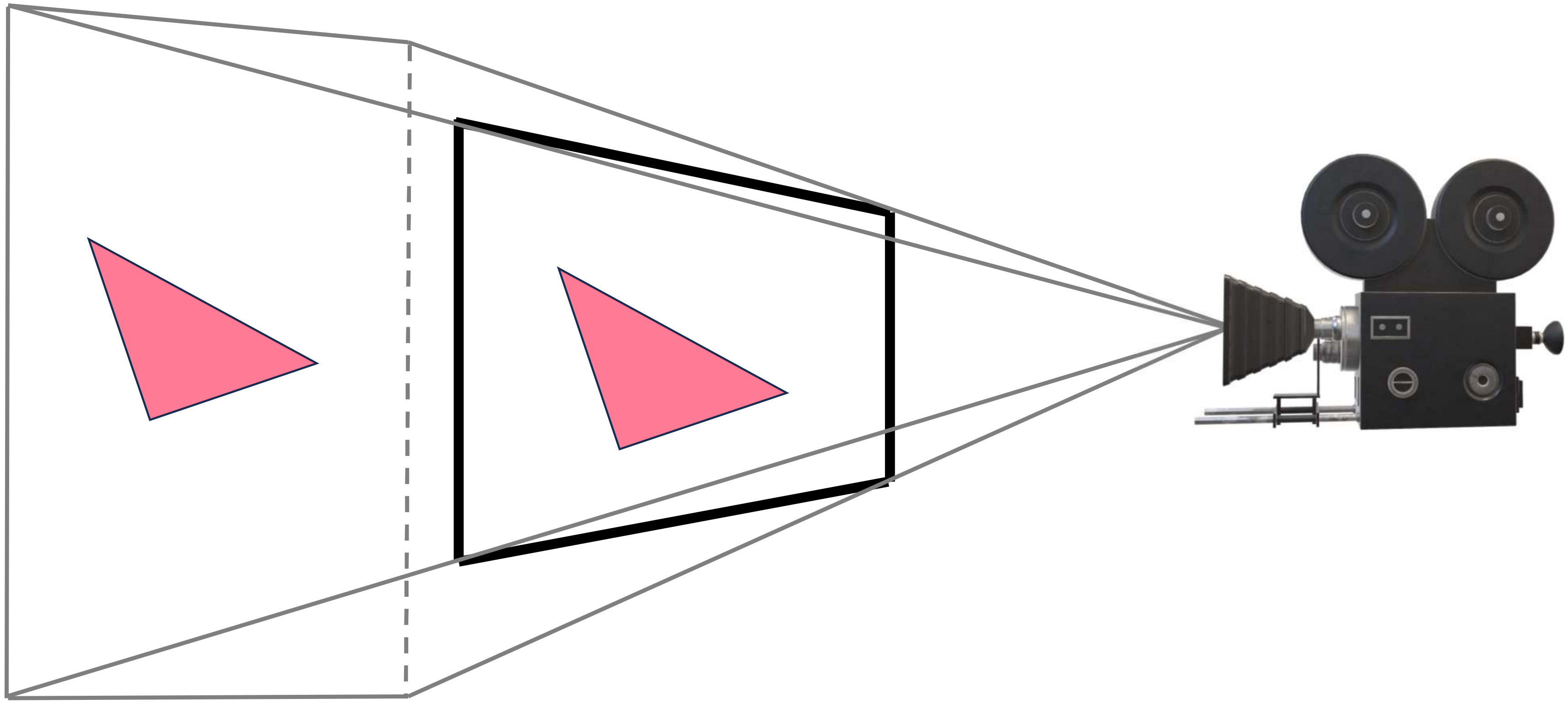
Rasterization vs Ray Tracing



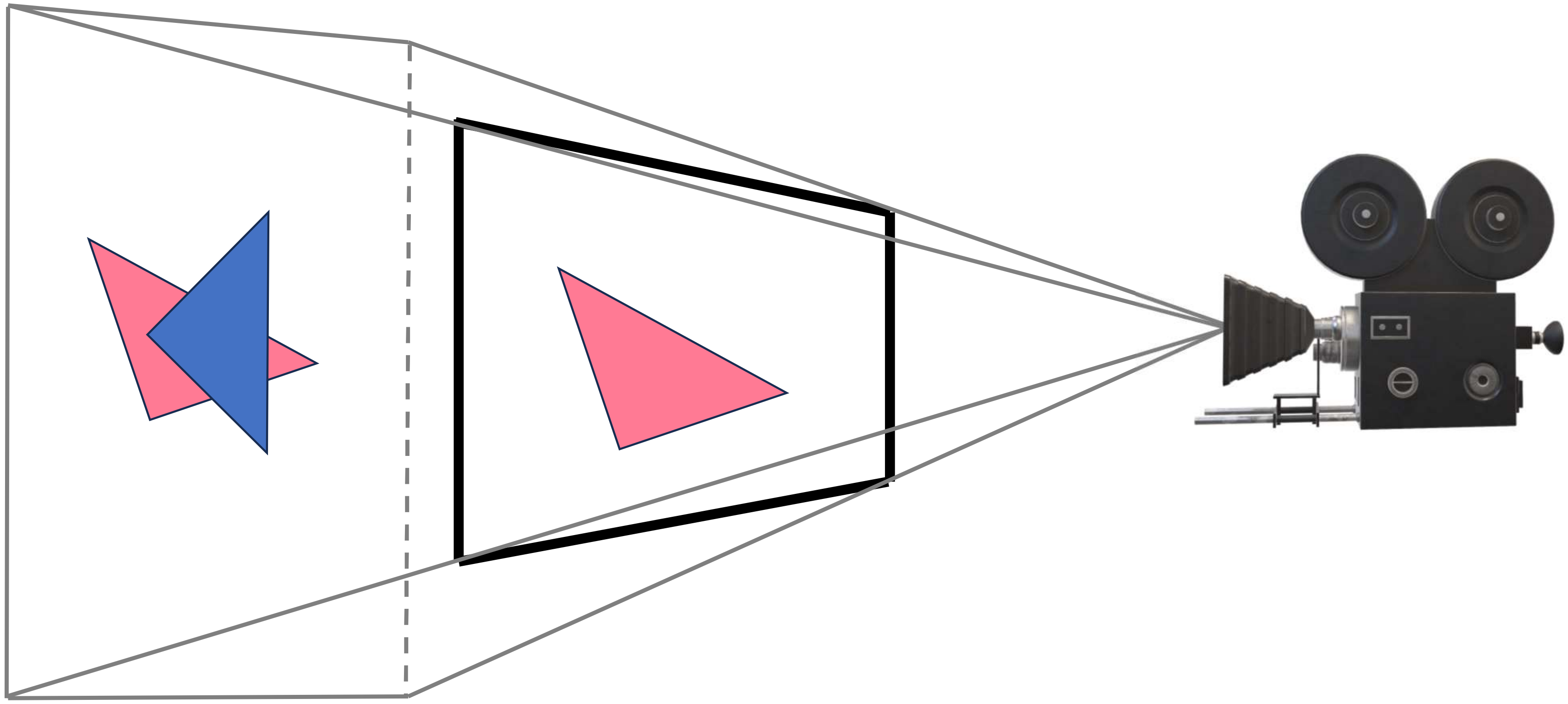
Rasterization



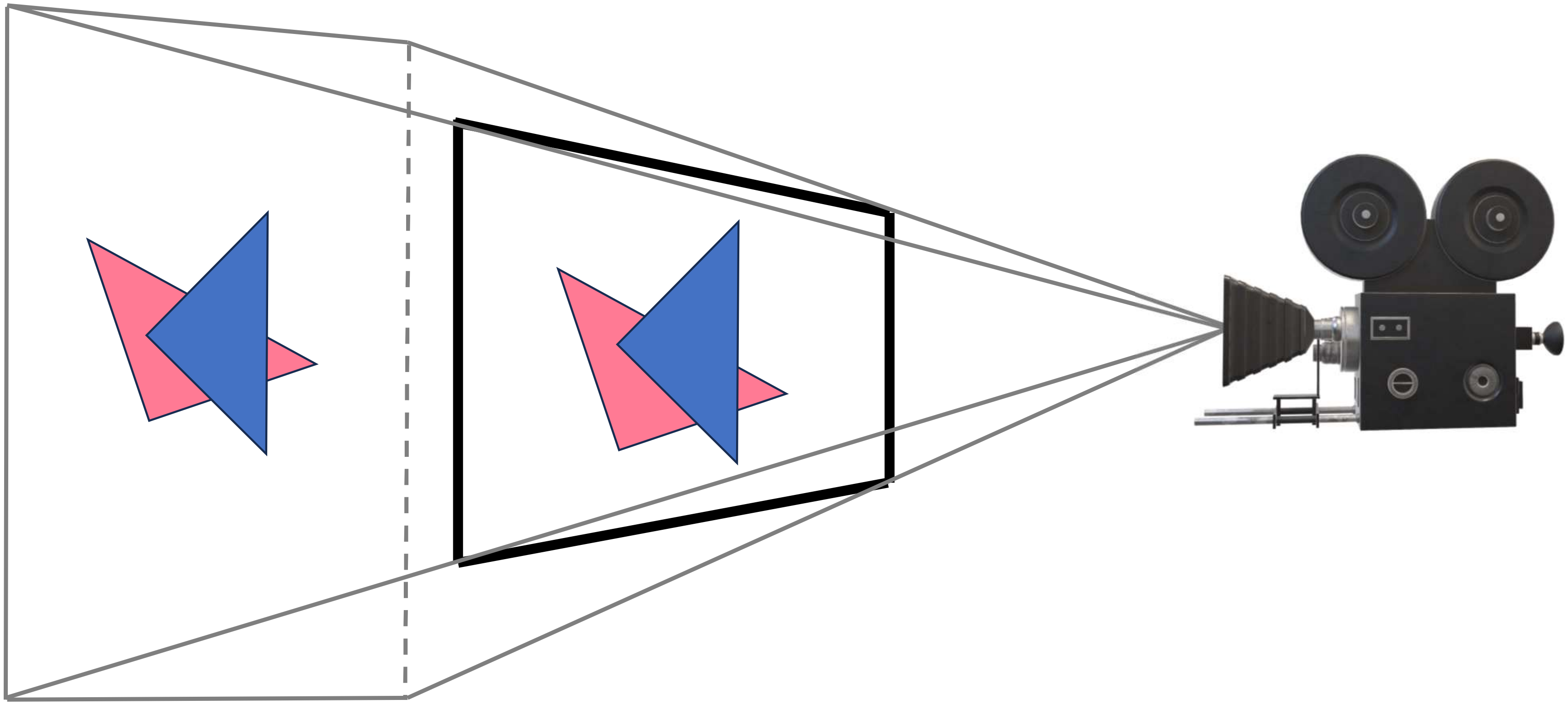
Rasterization



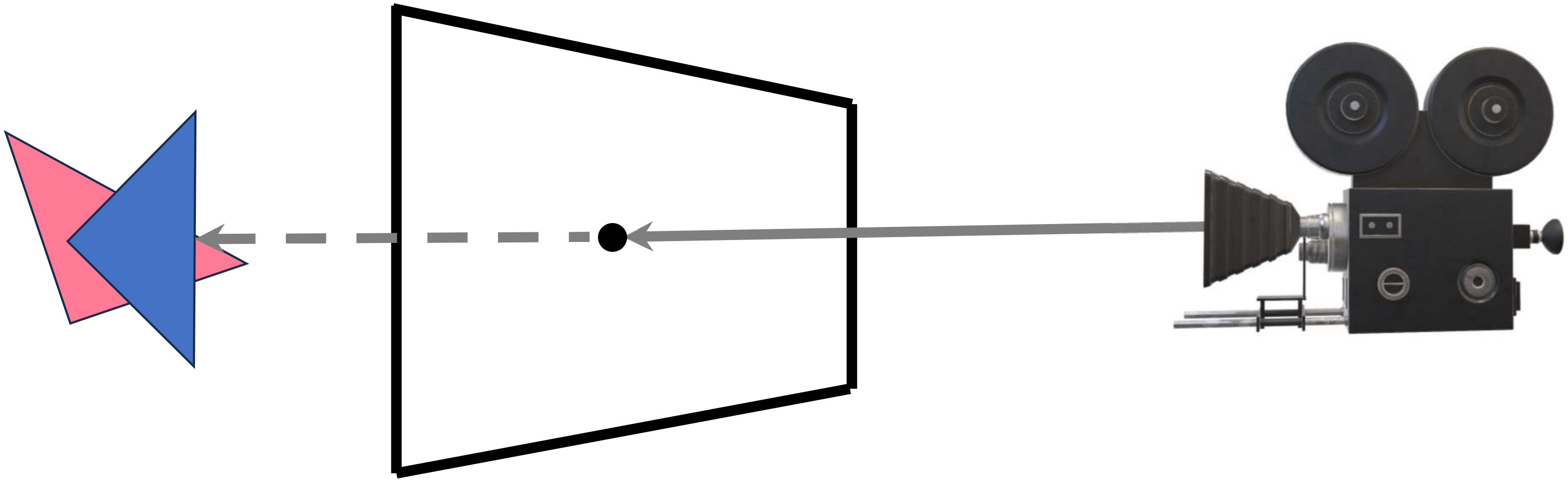
Rasterization



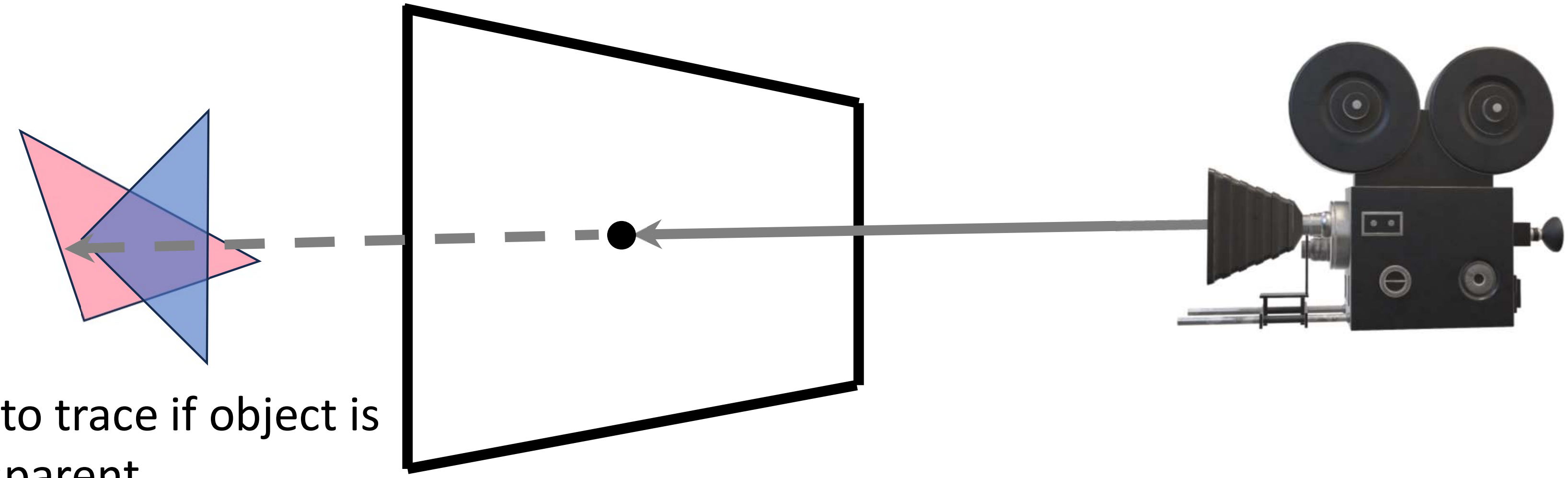
Rasterization



Ray Tracing

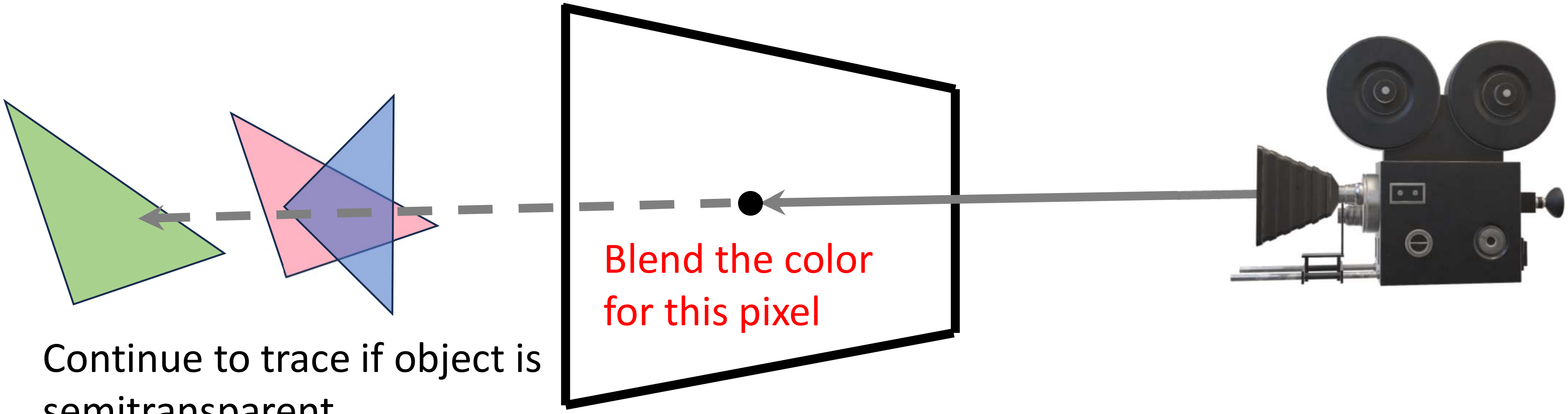


Ray Tracing



Continue to trace if object is semitransparent

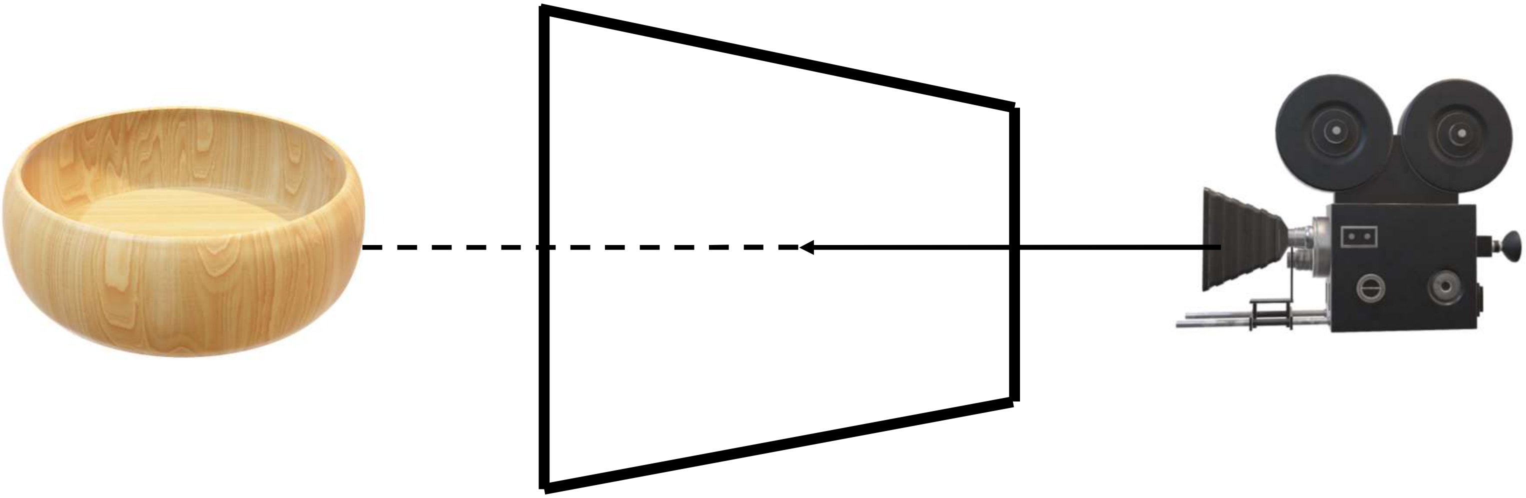
Ray Tracing



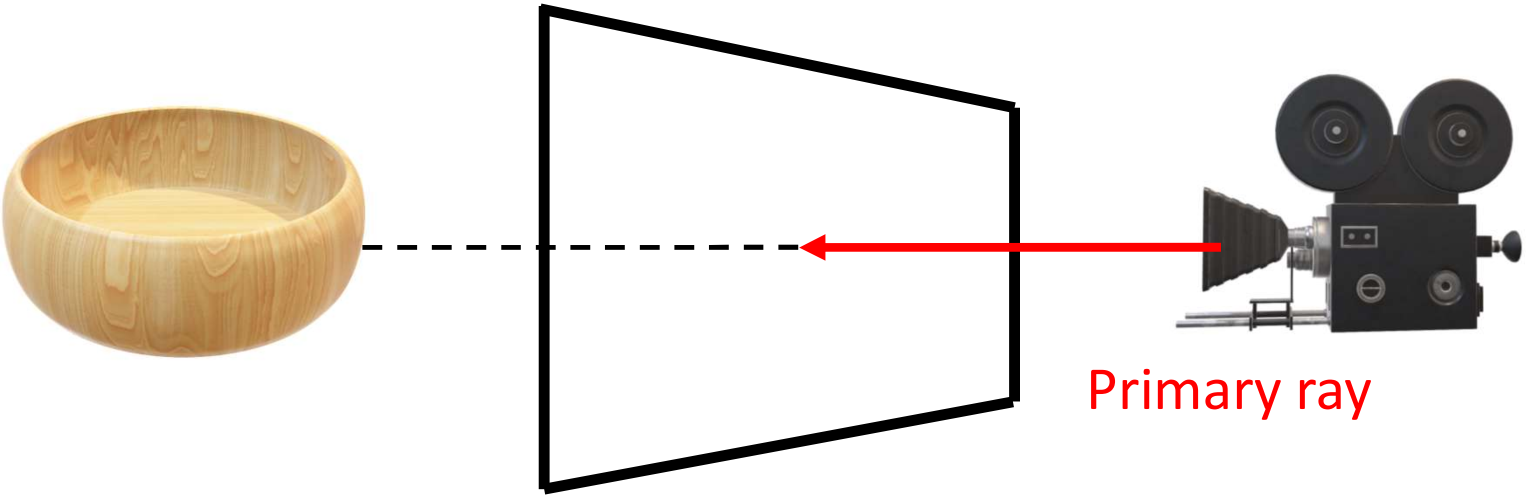
Blend the color for this pixel

Continue to trace if object is semitransparent

Various Types of Rays

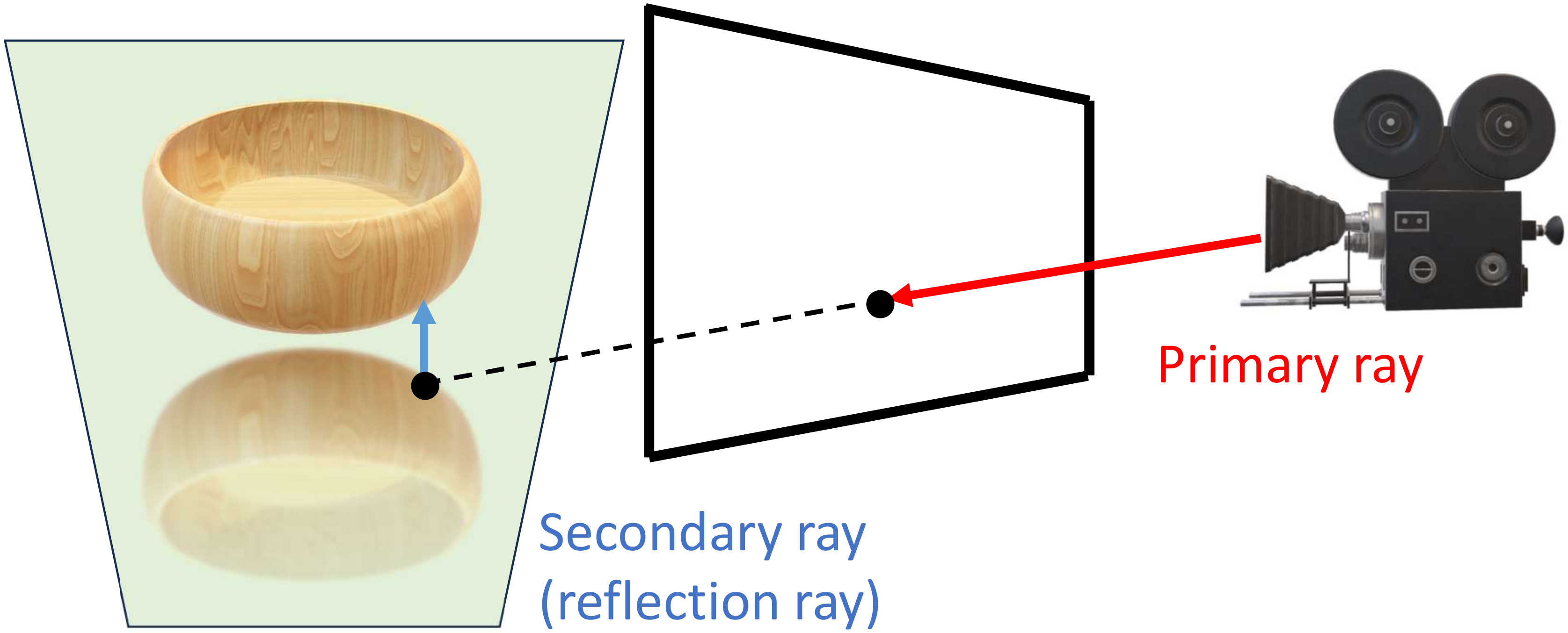


Various Types of Rays

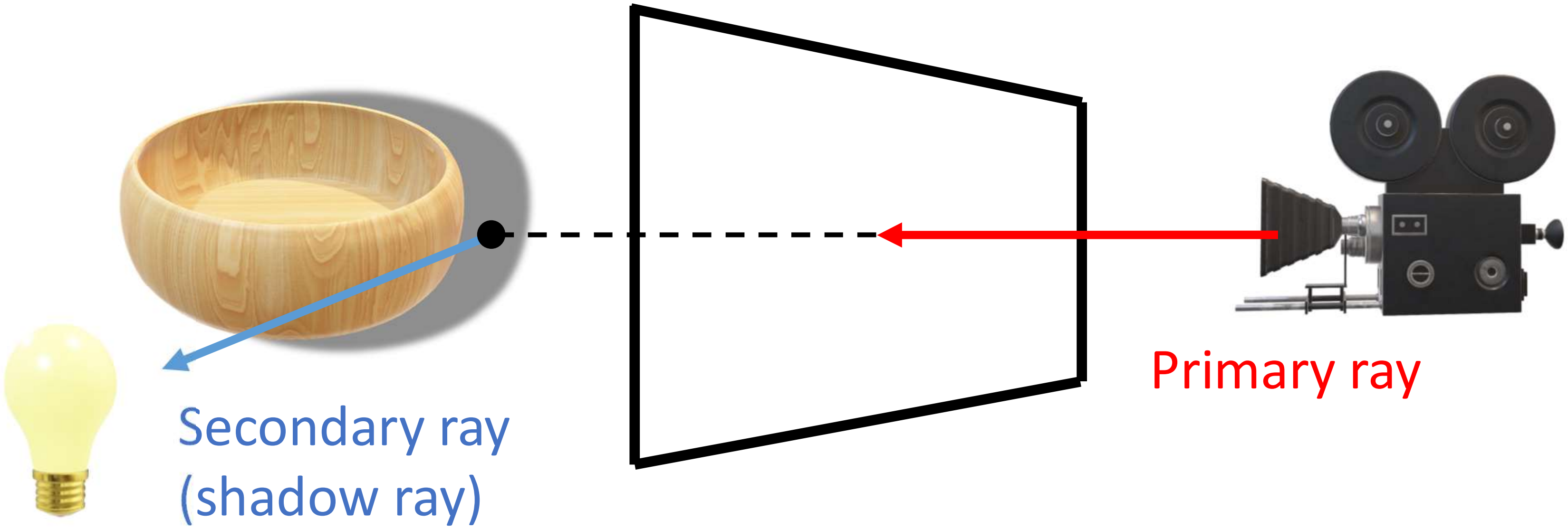


Primary ray

Various Types of Rays



Various Types of Rays



- Secondary Ray types
    - Reflection, shadow, refraction, realistic illumination, etc…

Rasterization vs Ray Tracing

- Rasterization is a fast operation that has GPU support
- Ray tracing on the other hand is expensive and can be slow
  - Need to trace many types of rays to generate realistic images
- Historically that is why computer games, animations, real-time graphics applications all use rasterization
- Ray tracing was thought as an off-line postproduction method to generate high quality realistic graphics!

Realistic Ray Tracing

Source: Nvidia, generated by: OctaneRender

Realistic Ray Tracing

™ & © 2016 Twentieth Century Fox Film Corporation. All rights reserved.

Source:artofvfx.com

Realistic Ray Tracing for VFX

IITK CS360A

Rasterization vs Ray Tracing

- **BUT** things are changing fast!
- Real time ray tracing is here!
- **Nvidia’s RTX**
  - Special hardware support for ray tracing operations

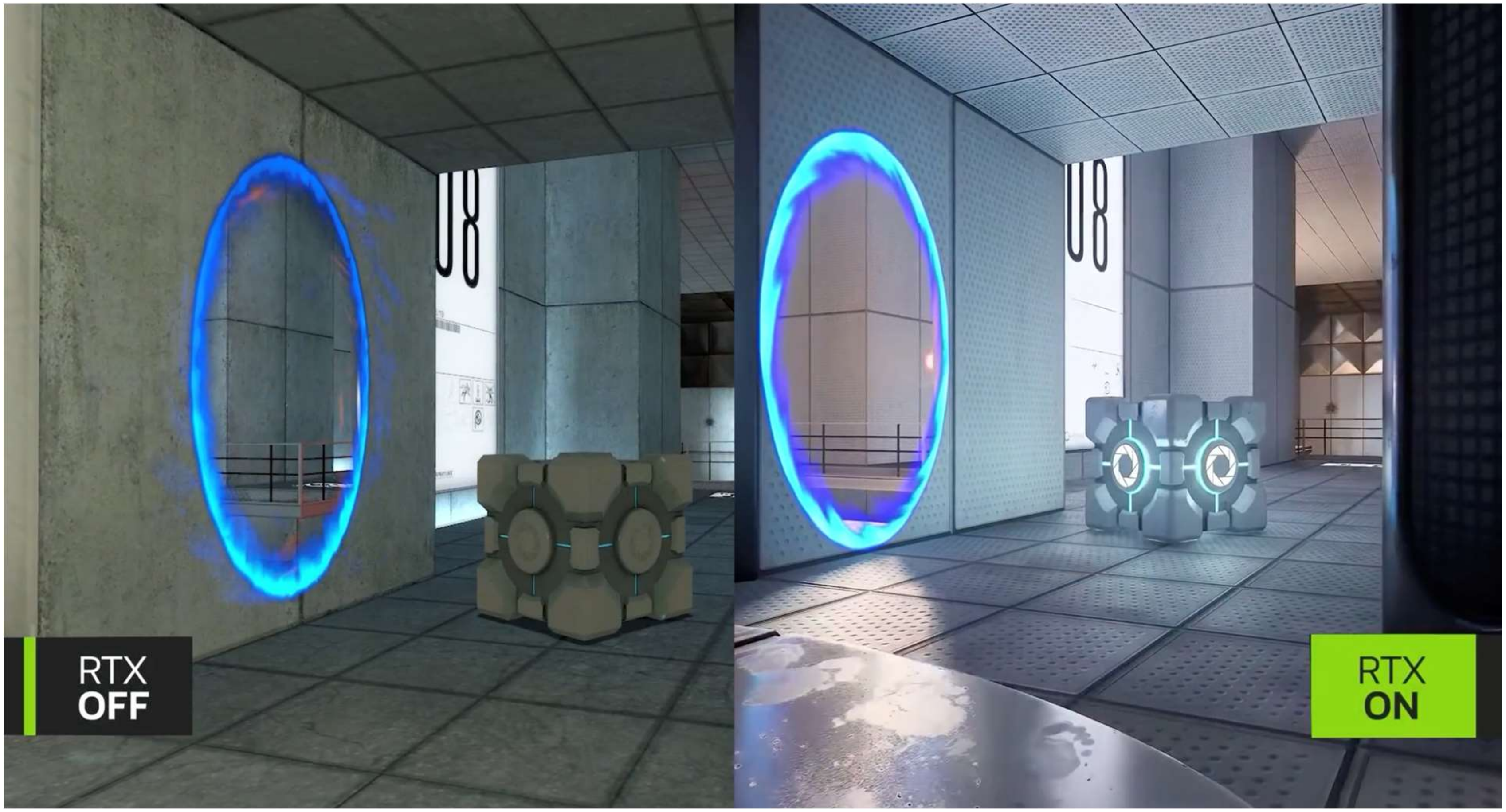
Ray Tracing

- **Software**
  - CPU-based
  - GPU-based
- **Hardware**
  - Ray tracing in specialized GPU hardware
  - Utah HWRT

Ray Tracing

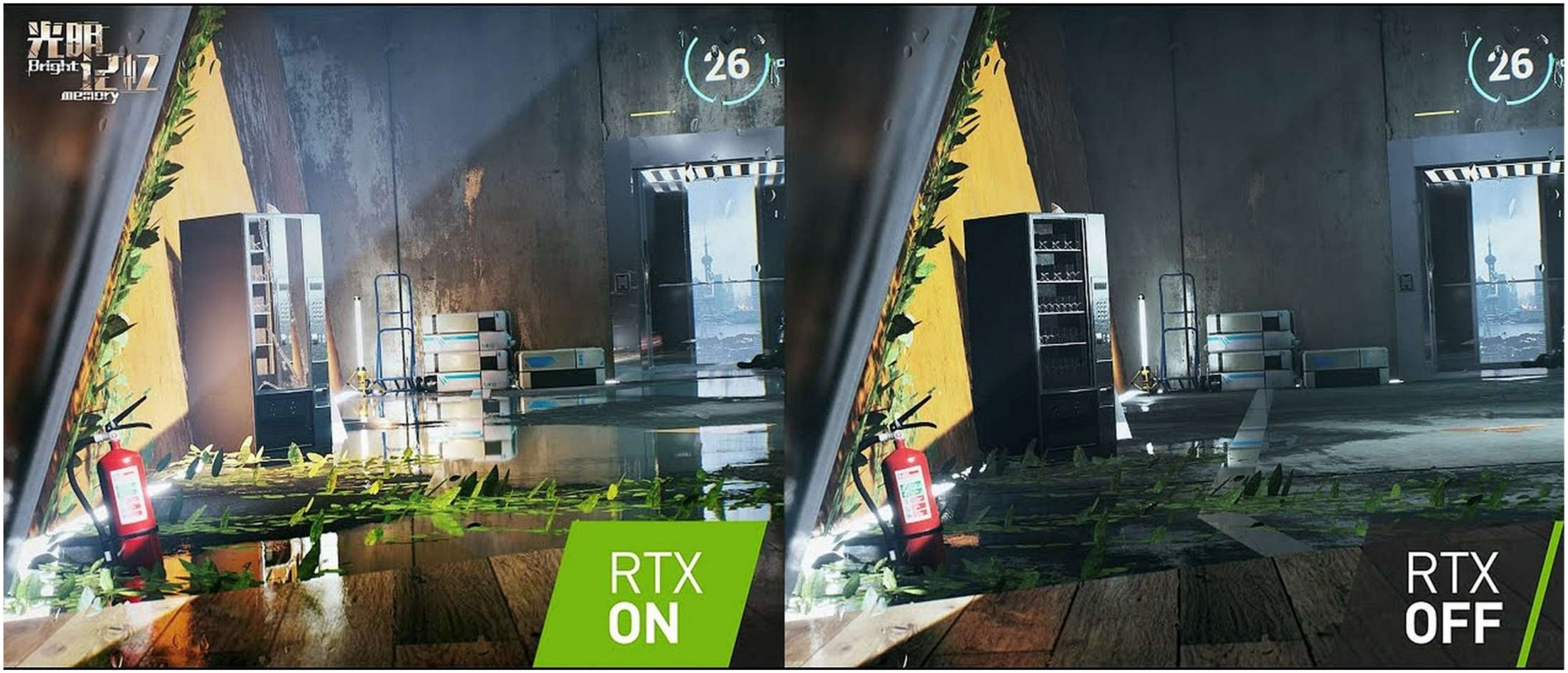
- **Software**
  - CPU-based
  - GPU-based (we will do this!)
- **Hardware**
  - Ray tracing in specialized GPU hardware
  - Utah HWRT

Nvidia RTX for Real Time Ray Tracing



https://www.ign.com/

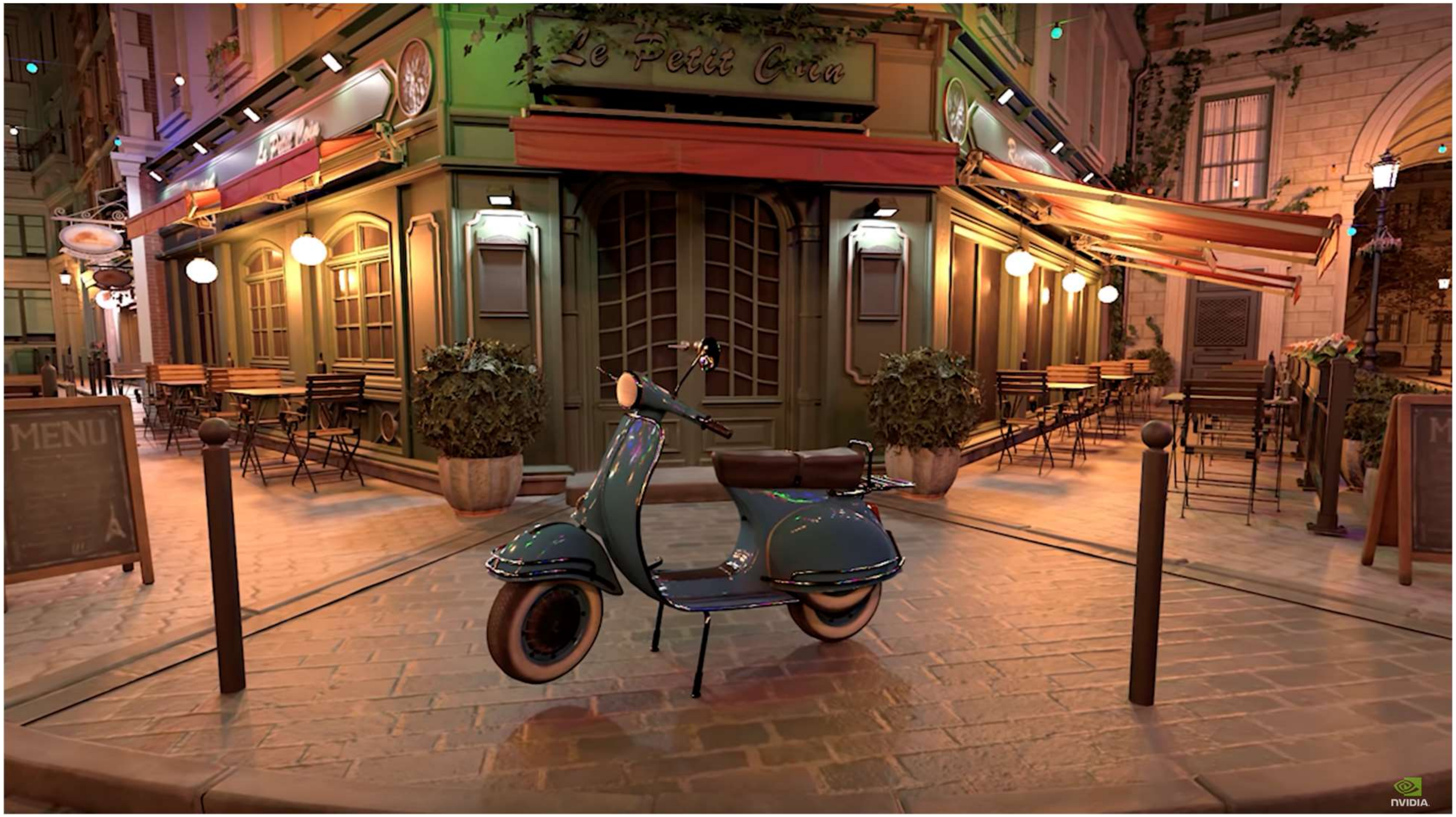
Nvidia RTX for Real Time Ray Tracing



Reality: Rasterization + Ray Tracing

- Use rasterization for primary scene generation
- Use ray tracing for special effects and realism
  - Reflection
  - Refraction
  - Shadow
  - Global illumination
  - Ambient occlusion
  - Many more…

Reality: Rasterization + Ray Tracing



Source: Nvidia

NVIDIA Marbles at Night | RTX Demo