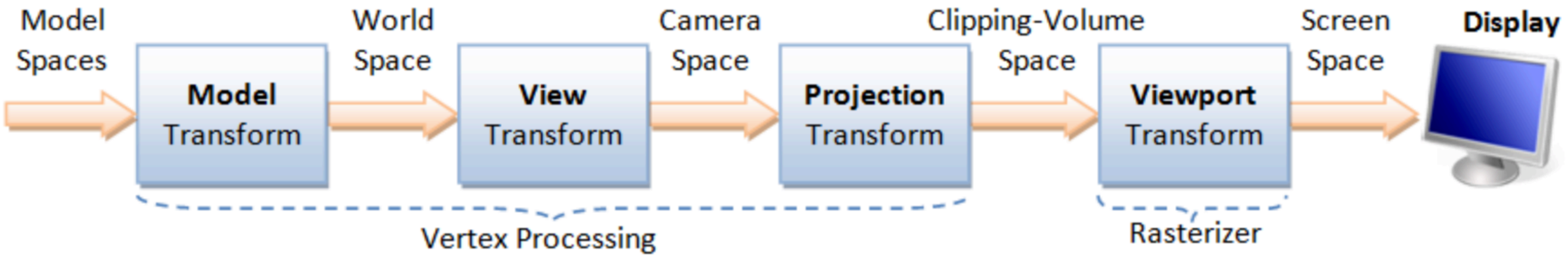
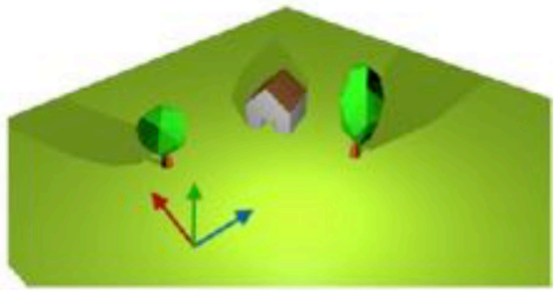
CS/PSY 484: Visual Rendering



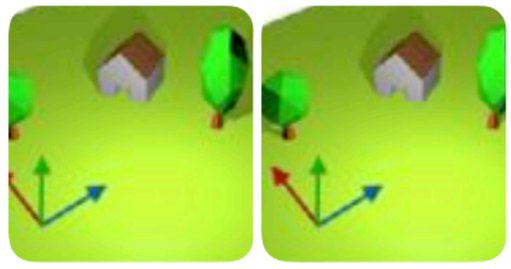
remember this from chapter 4?

Model-World-View Projection

model space

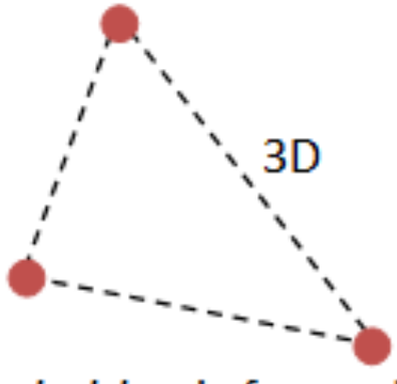


World space



Stereo view space

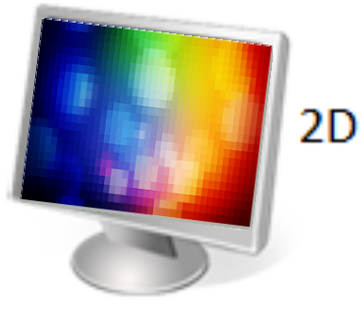
how is every pixel colored?



A *primitive* is formed by one or more *vertices*.

Vertices are not grid-aligned

?



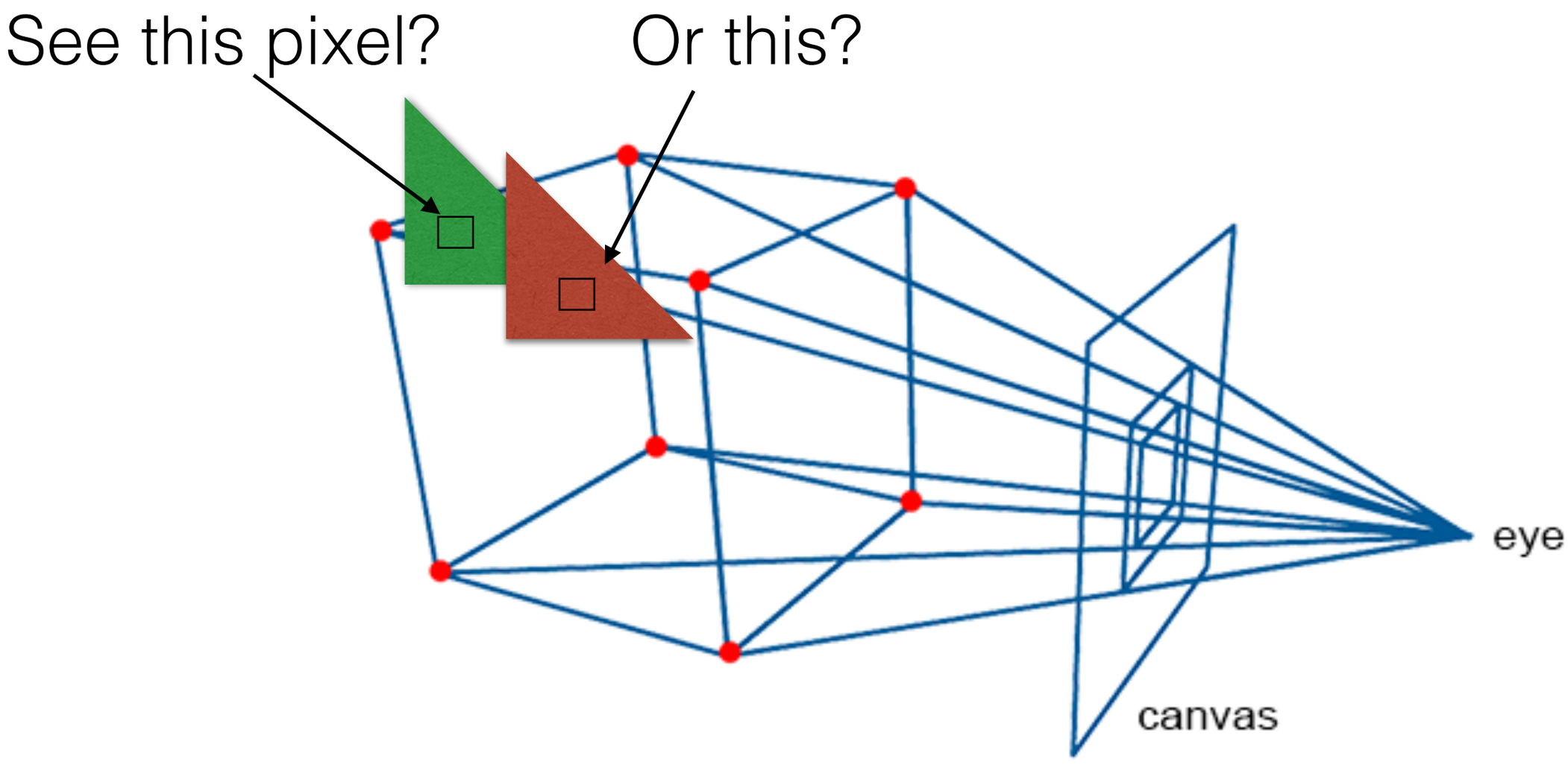
All primitives are merged to produce 2D *pixels* on the display

rendering process

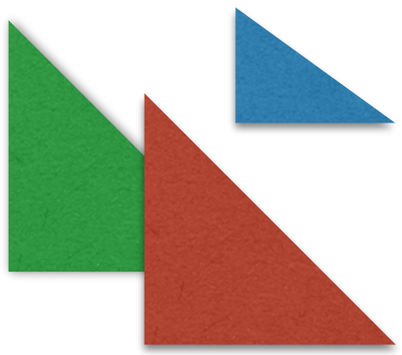
- Visibility problem (pixel visible)
- Shading (if so what color?)

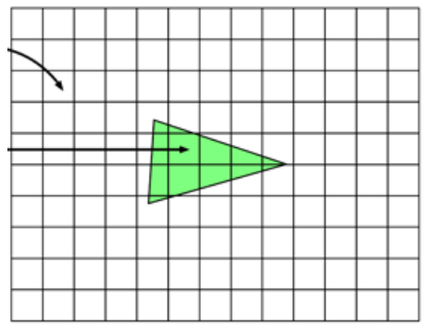
} combined

visibility problem



Rendering process





Two distinct rendering methods:

- Object order (iterate over triangles) complex ~ 1M triangles to
- Image order (iterate over pixels) 1080p ~ 2M pixels to consider
- Complicated for VR due to high refresh rates >100 Hz.

Raytracing

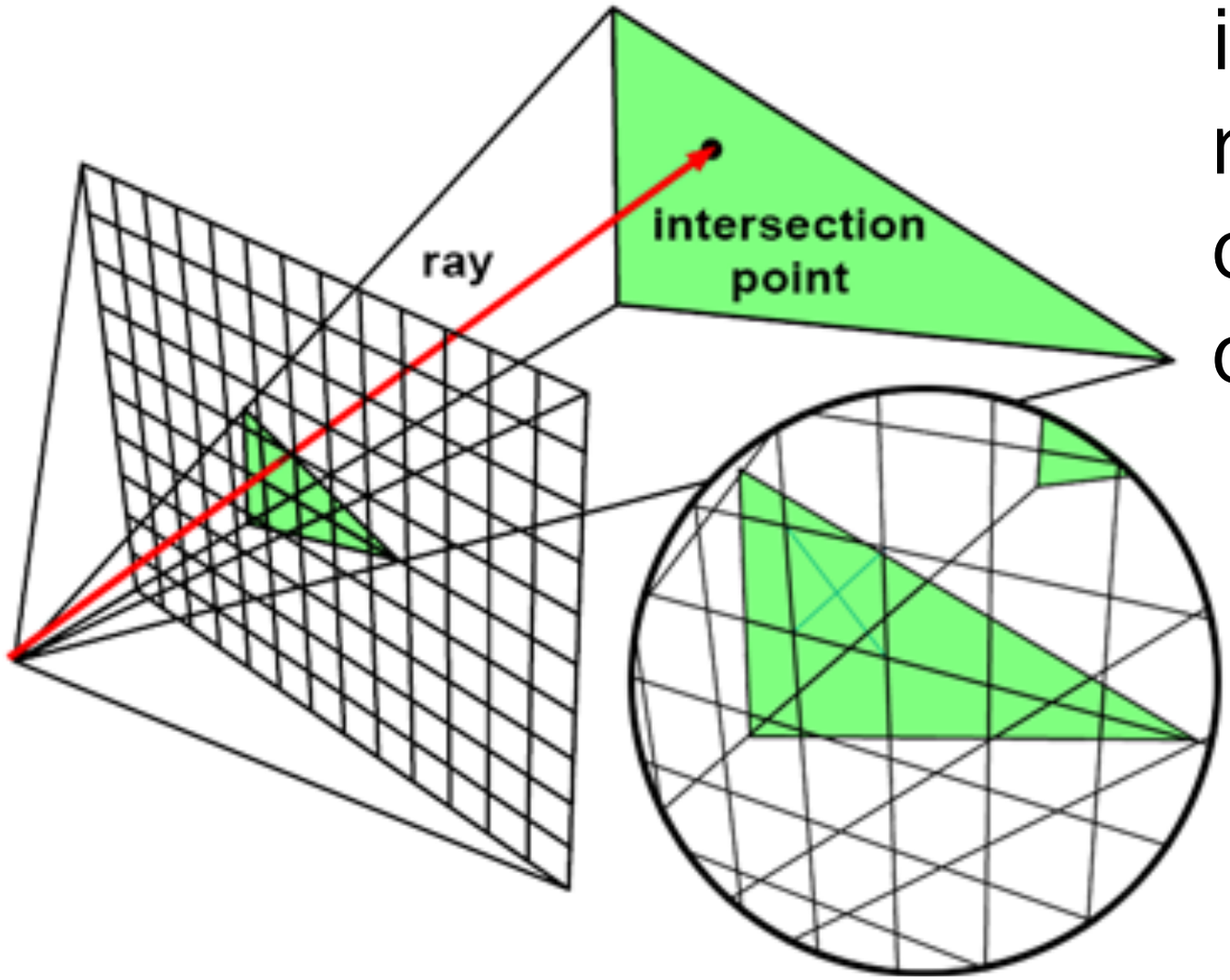
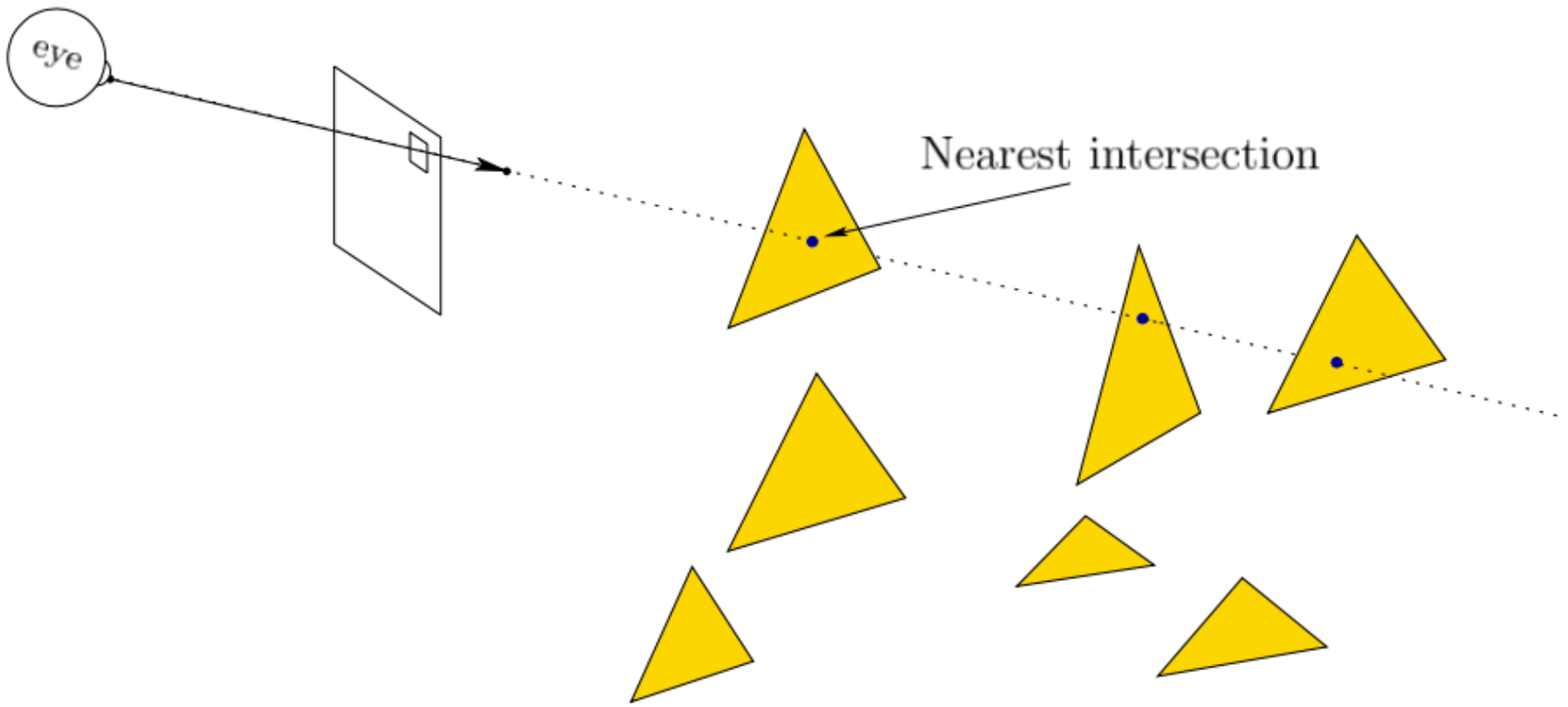


image order rendering: loops over pixels, then over triangles

Two steps: 1) ray casting and 2) shading

Ray casting

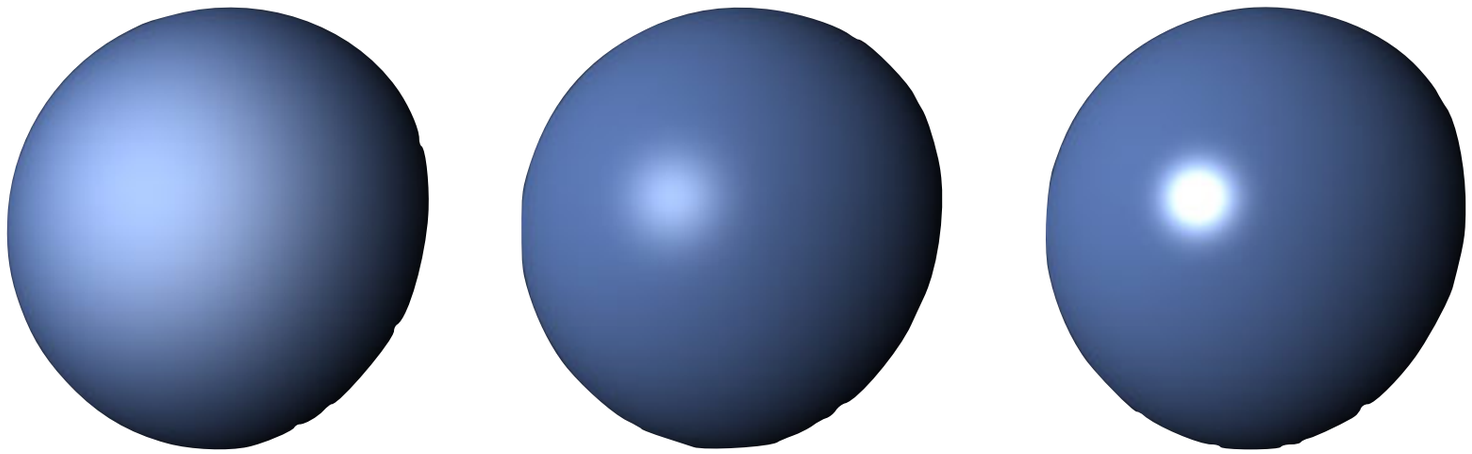


Using analytical geometry: take a pixel, cast ray from the eye to find the closest triangle.

Arrange triangles into a 3D data structure

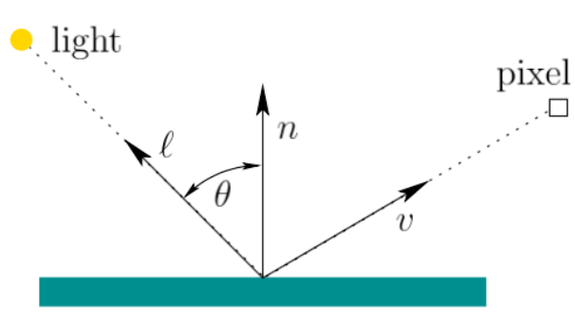
Shading

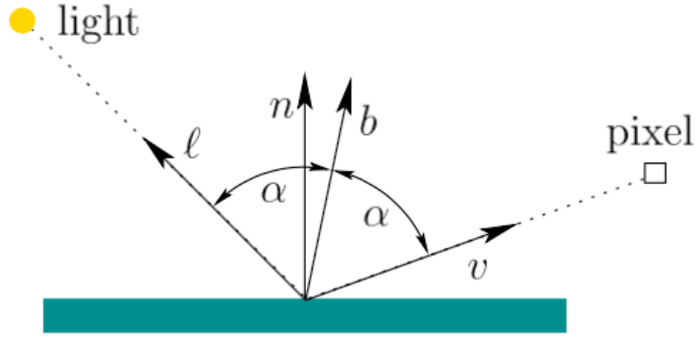
What color should each pixel have?



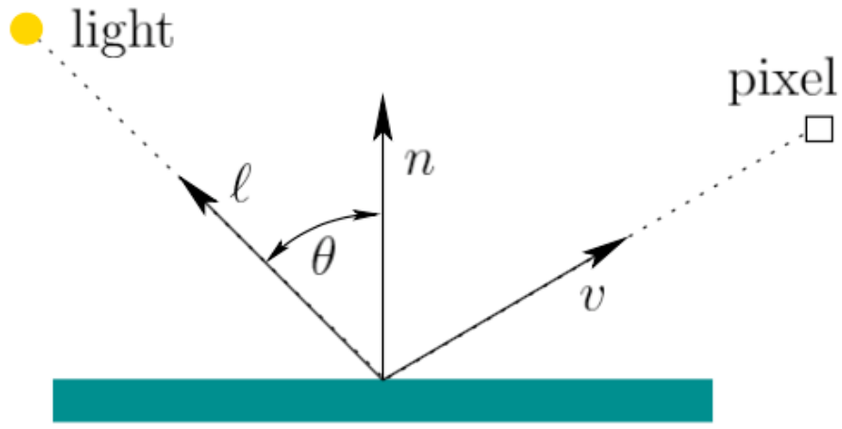
Lambert

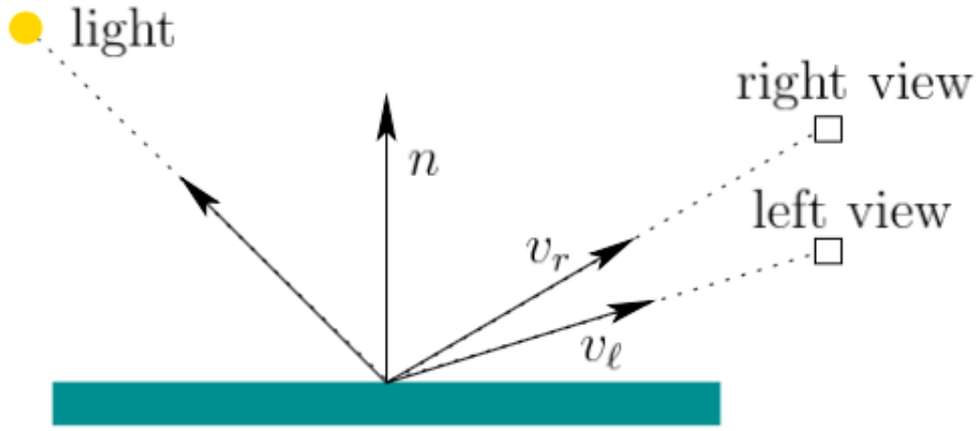
Blinn-Phong





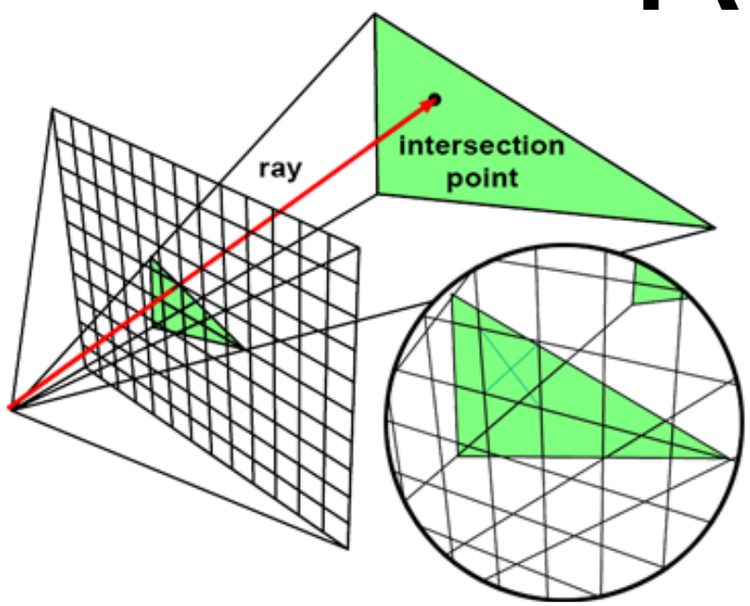
Shading for VR





challenging to simulate

Raytracing

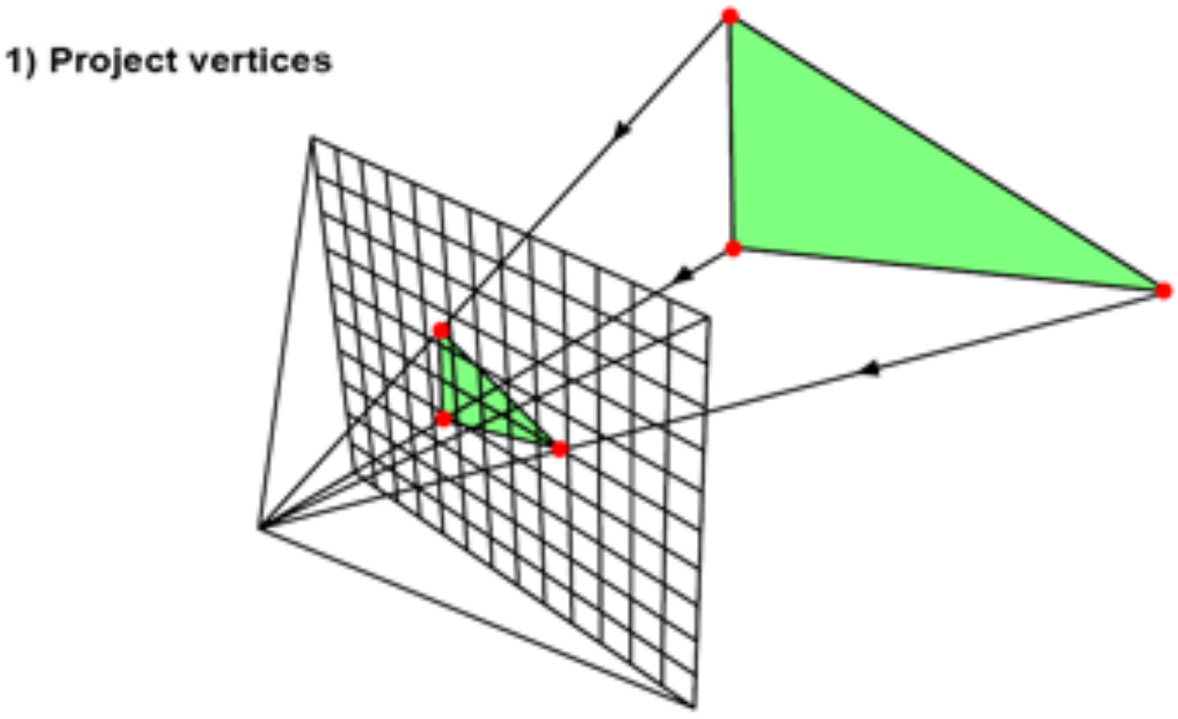


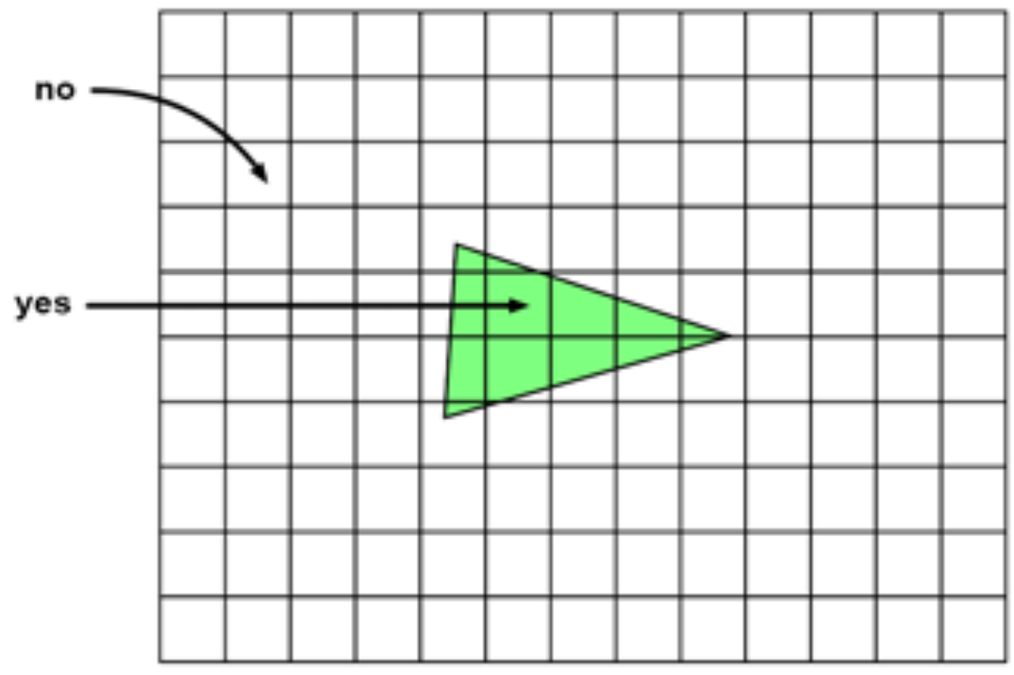
Doesn’t scale.
1080p image @90hz
Requires 180M raycasts per second

Real time Raytracing for
Games demonstrated
You need a pricey GPU

https://youtu.be/
J3ue35ago3Y

Rasterization





Object order rendering: loops over triangles, then over pixels

But how do we know which triangle is visible?

(visibility problem)

painters algorithm

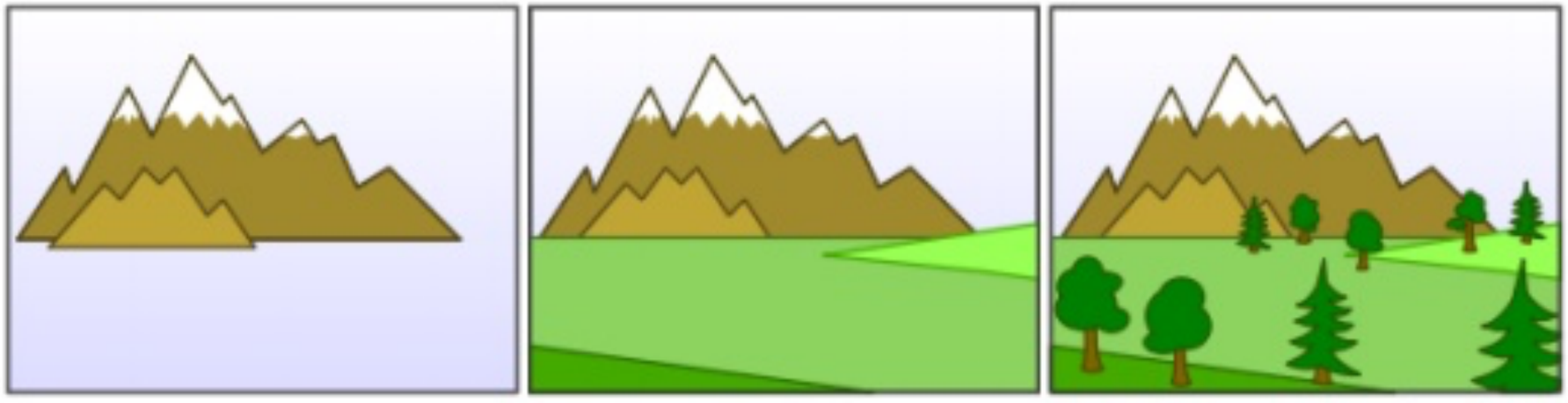
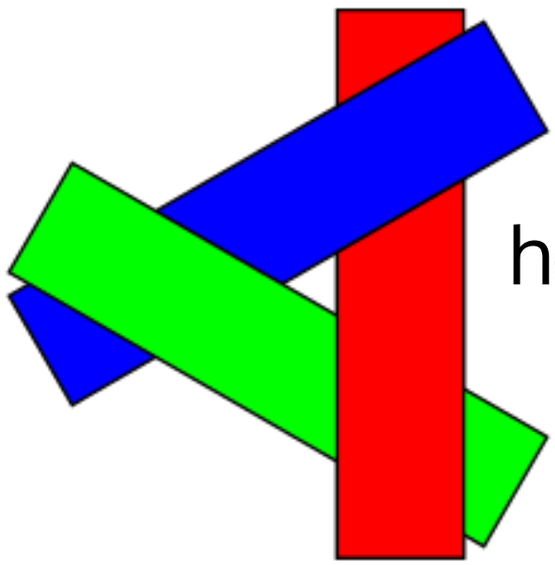


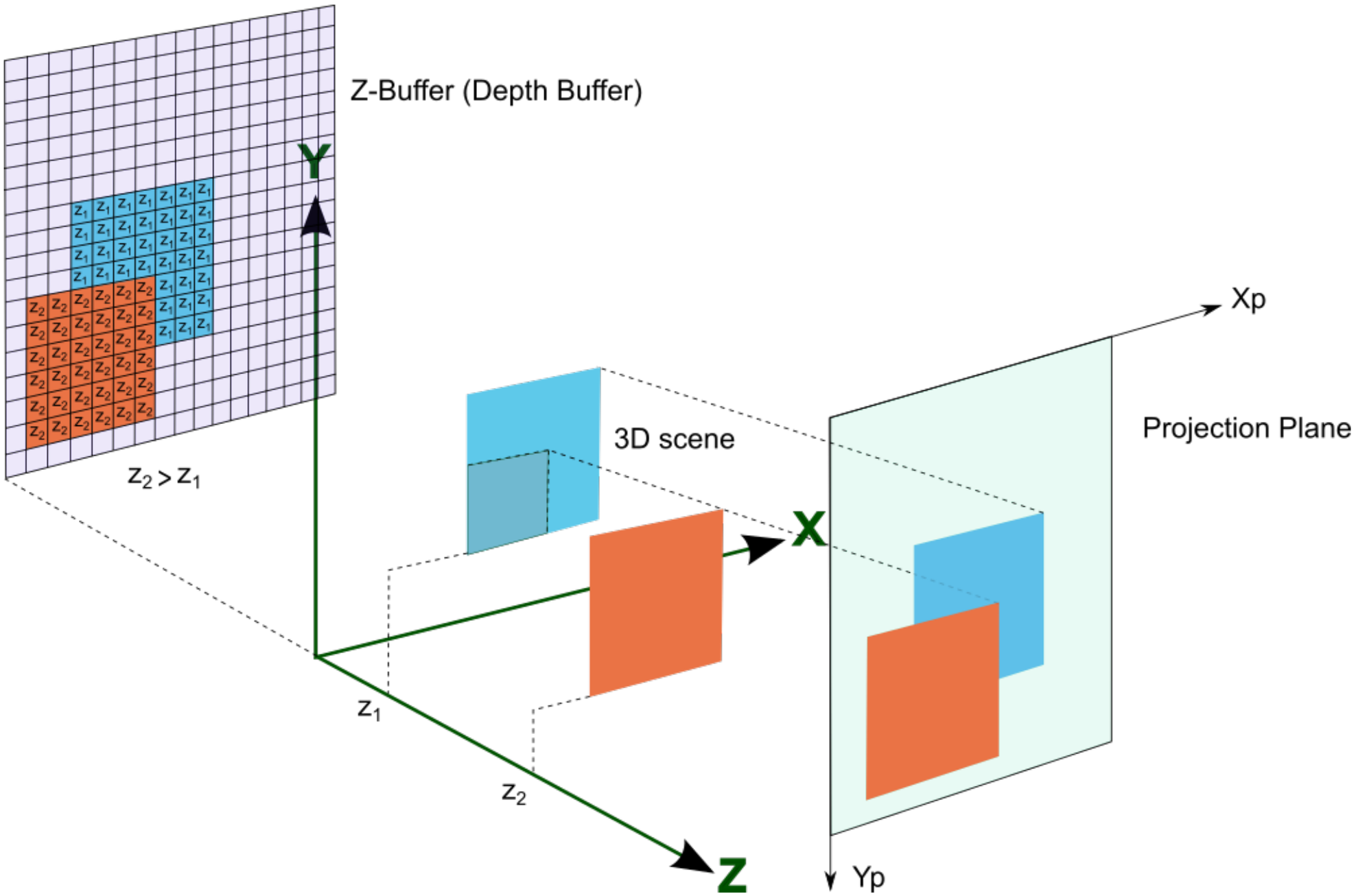
Fig.3.4. Painter's algorithm

inefficient for large numbers of objects (many won't be visible)



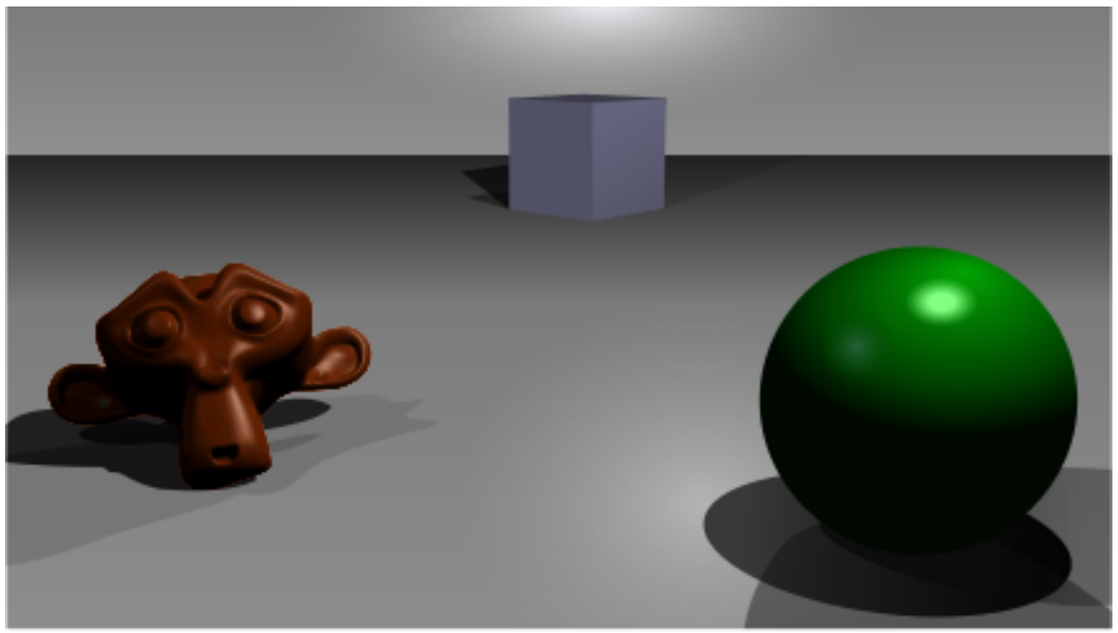
how to solve this?

Z-buffer

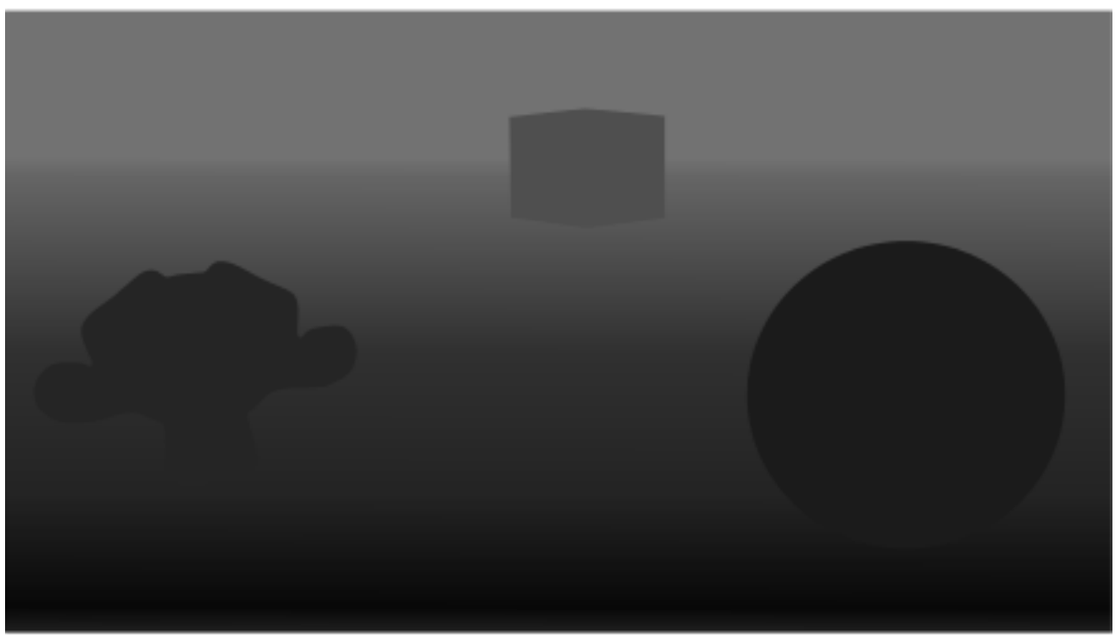


Keep track of the depth of every pixel that is visible

Z-buffer



A simple three-dimensional scene



Z-buffer representation

How It Works:

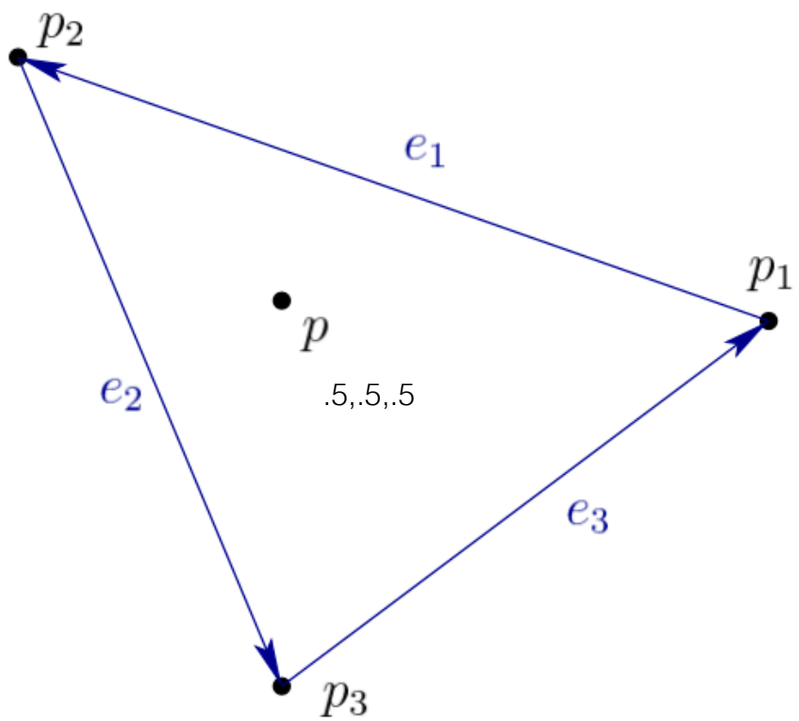
- Each pixel in the framebuffer has a corresponding depth value stored in the **Z-buffer**.
- When rendering a new pixel, its depth (Z-value) is compared to the existing depth in the buffer:
  - If the new pixel is **closer**, it replaces the old one.
  - If it's **farther**, it's discarded.

Benefits:

✅ **Efficient hidden surface removal** (only the closest surfaces are drawn).
✅ **Works well with modern GPUs** for real-time 3D rendering.
✅ **Supports complex scenes with overlapping objects.**

Barycentric coordinates

1,0,0



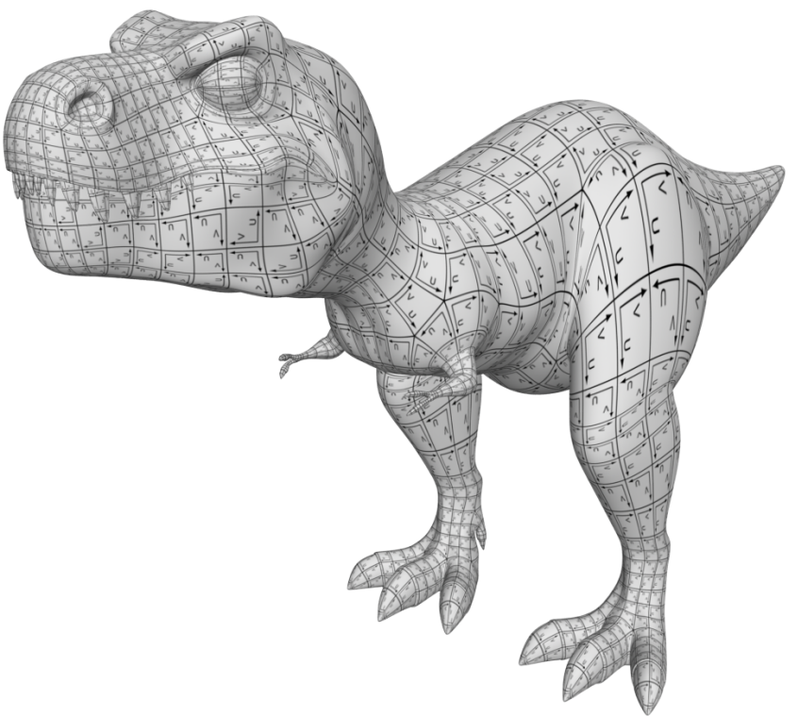
Barycentric coordinates specify the location of every point p in a triangle as a weighted average of its vertices p1, p2 and p3

Simplifies test if a point is inside or outside the triangle

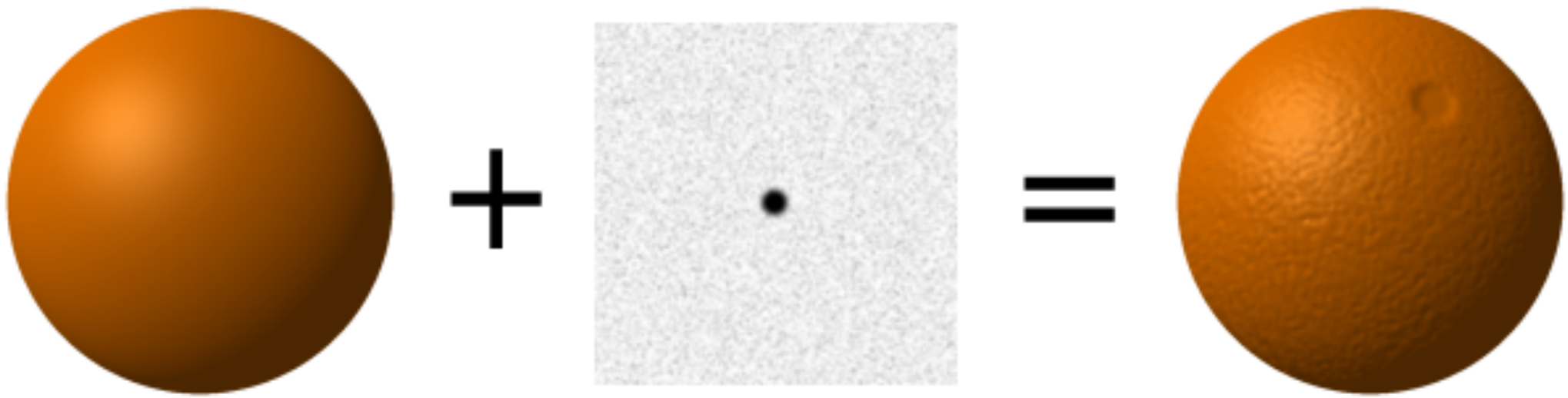
Comparison

Feature	Object-Order Rendering	Image-Order Rendering
Processing Approach	Iterates over objects	Iterates over pixels
Example Algorithms	Z-buffer, rasterization	Ray tracing, ray casting
Performance	Fast, hardware-accelerated	Slower, but high-quality lighting
Overdraw Issues	Yes (depth tests needed)	No (only visible pixels computed)
Real-time Suitability	Excellent (used in games)	Traditionally slow, but improving
Best for	Real-time graphics, games	High-quality offline rendering, movies

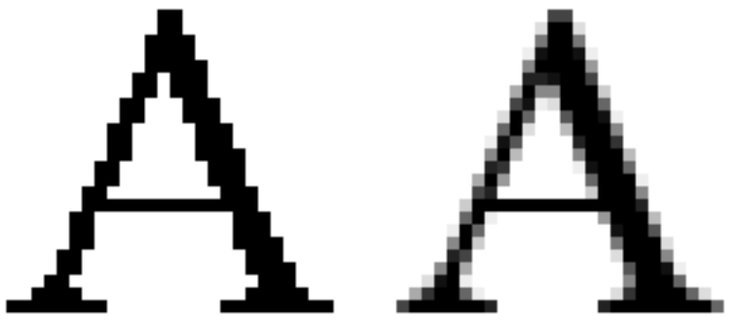
texture mapping



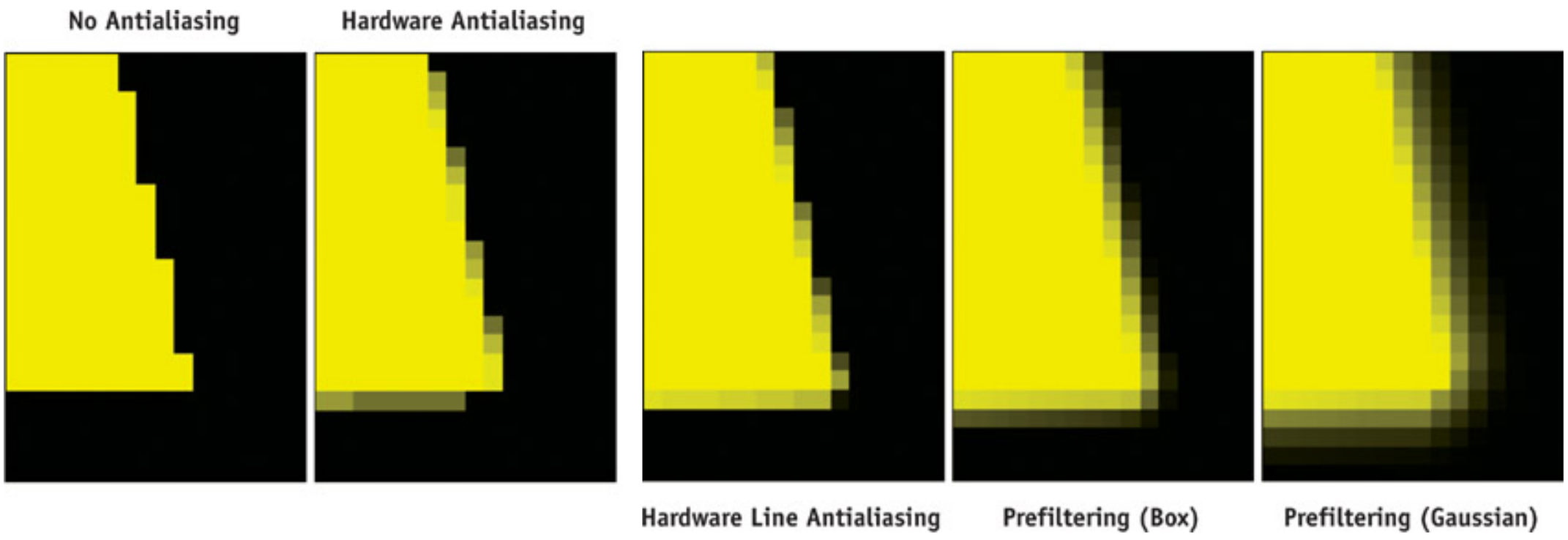
Bump mapping



alters the surface normal
Alters shading/appearance



Aliasing



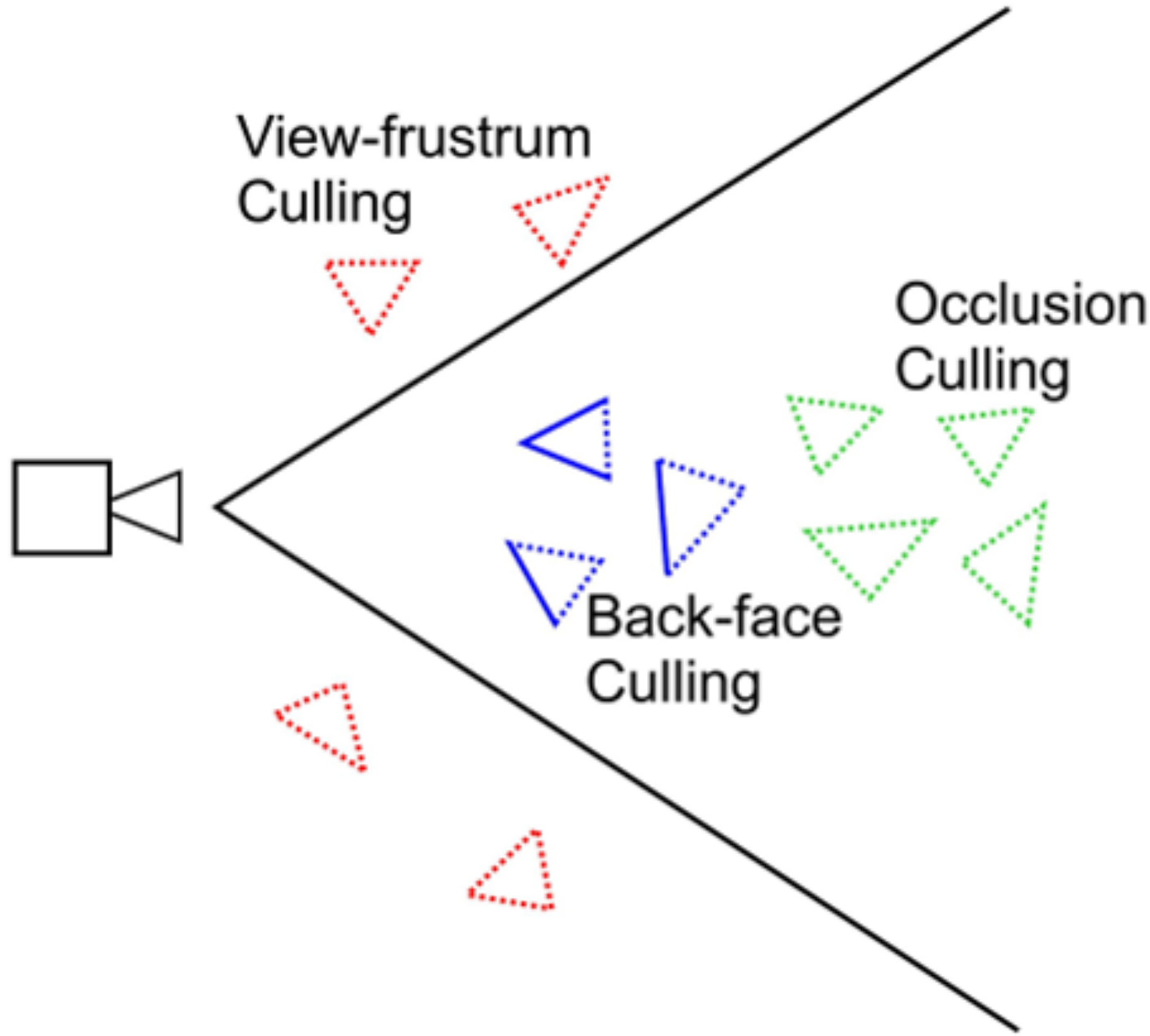
Artifacts appear due to discretization

not a huge problem for modern displays (>300 ppi)
but it is for VR (eyes close to screen)

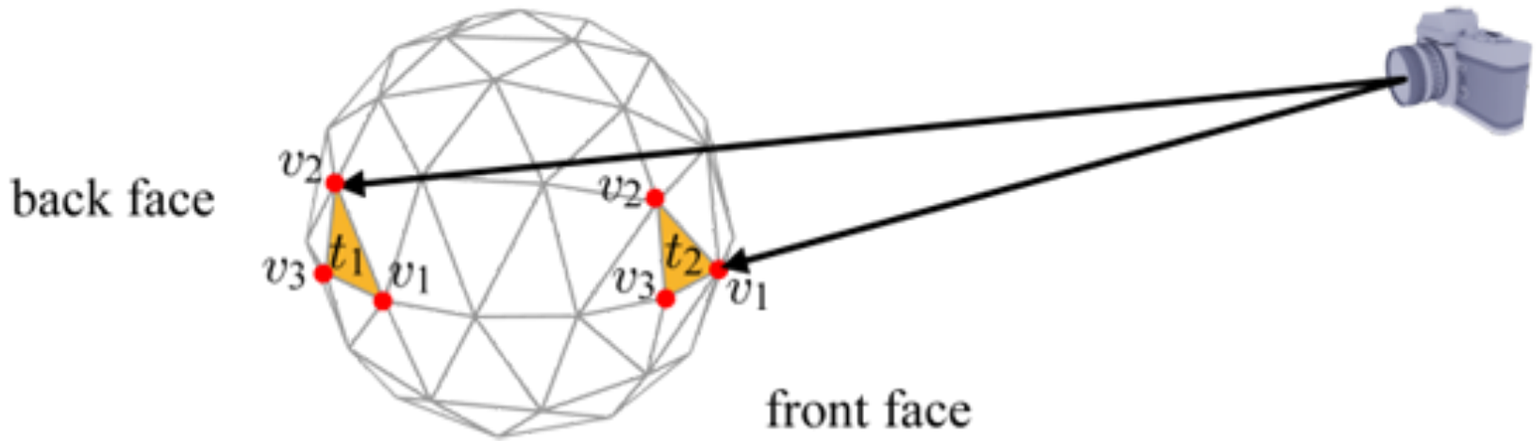
Mipmapping

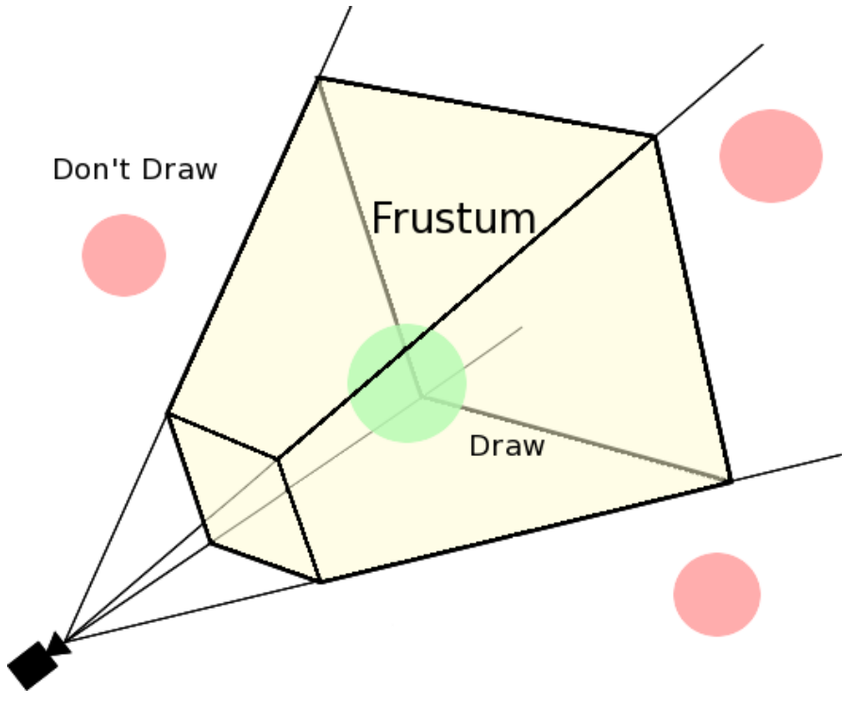
A **mipmap** is a precomputed sequence of progressively smaller versions of a texture.
When rendering, the system selects the most appropriate texture level based on the object's distance from the camera.
This reduces aliasing (unwanted visual artifacts) and improves performance by using lower-resolution textures for distant objects.

culling


View-frustrum
Culling
Occlusion
Culling
Back-face
Culling

back-face / frustum culling



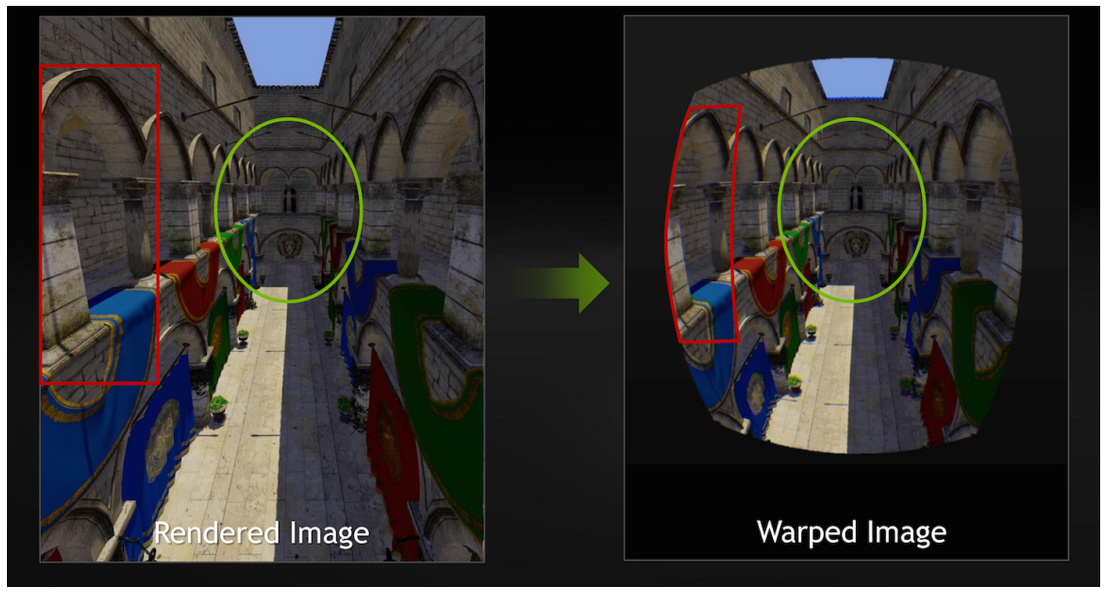


VR rasterization problems

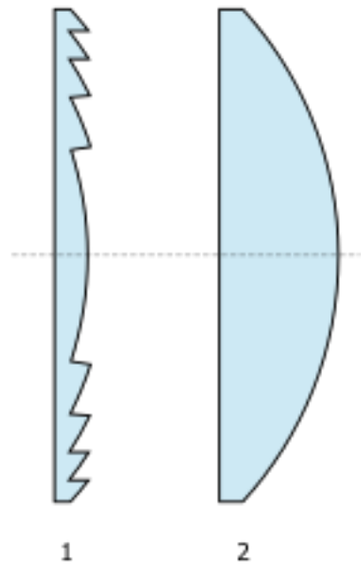
- Aliasing problem due to VR displays not having a high pixel density (pixels vs refresh rate).
- Provide moving mismatch (tradeoff refresh rate versus pixel density)
- Magnified by stereo perception
- Stereo vision; texture mapping may look fake

Screen door effect

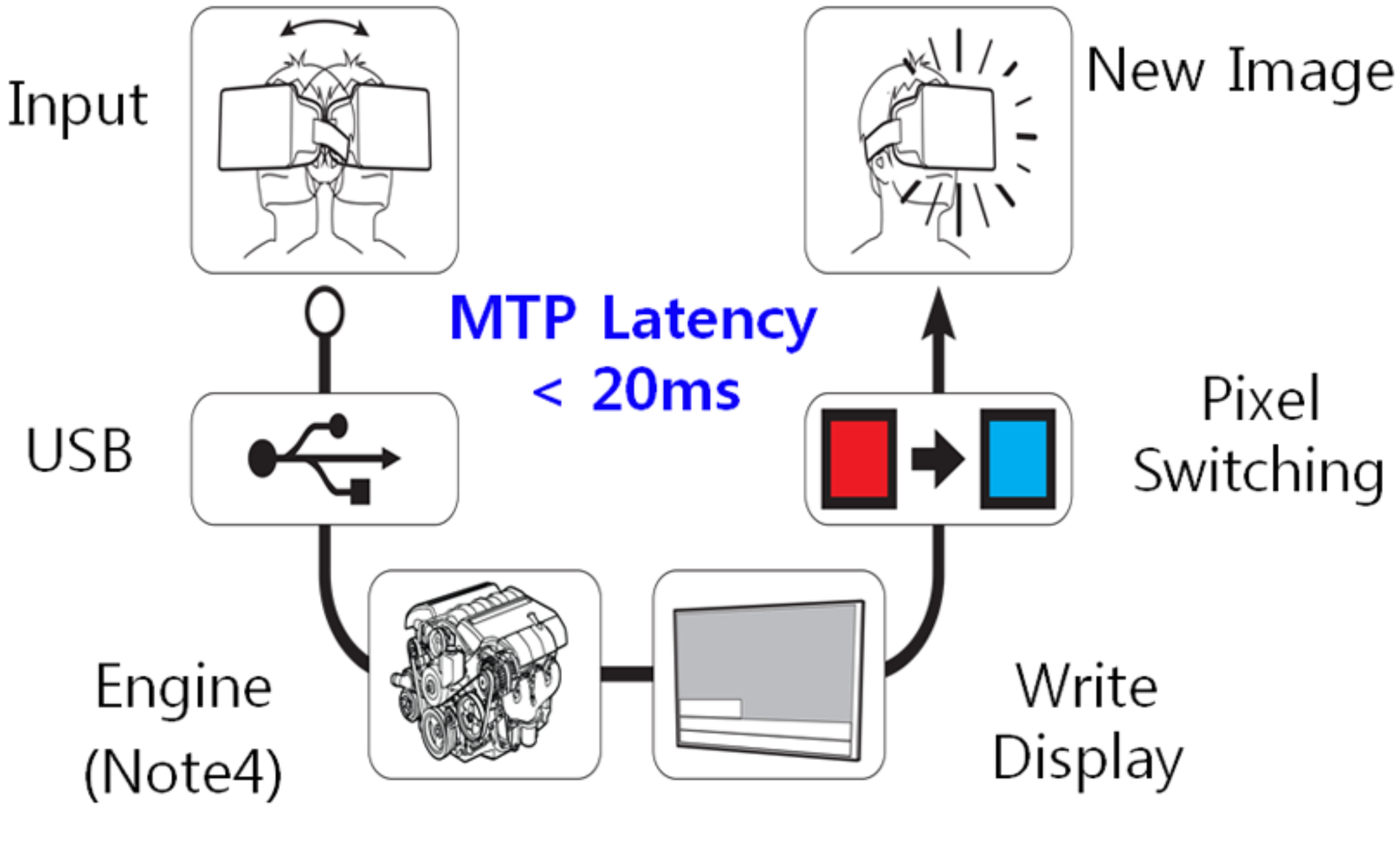
Correcting distortions



- barrel /pincushion distortion - because of a high FOV
- correct using fresnel lens
- More accurately bends light
- Also thinner
- But can have a bit of glare with light coming from behind



Latency


Input
New Image
MTP Latency
< 20ms
USB
Pixel
Switching
Engine
(Note4)
Write
Display

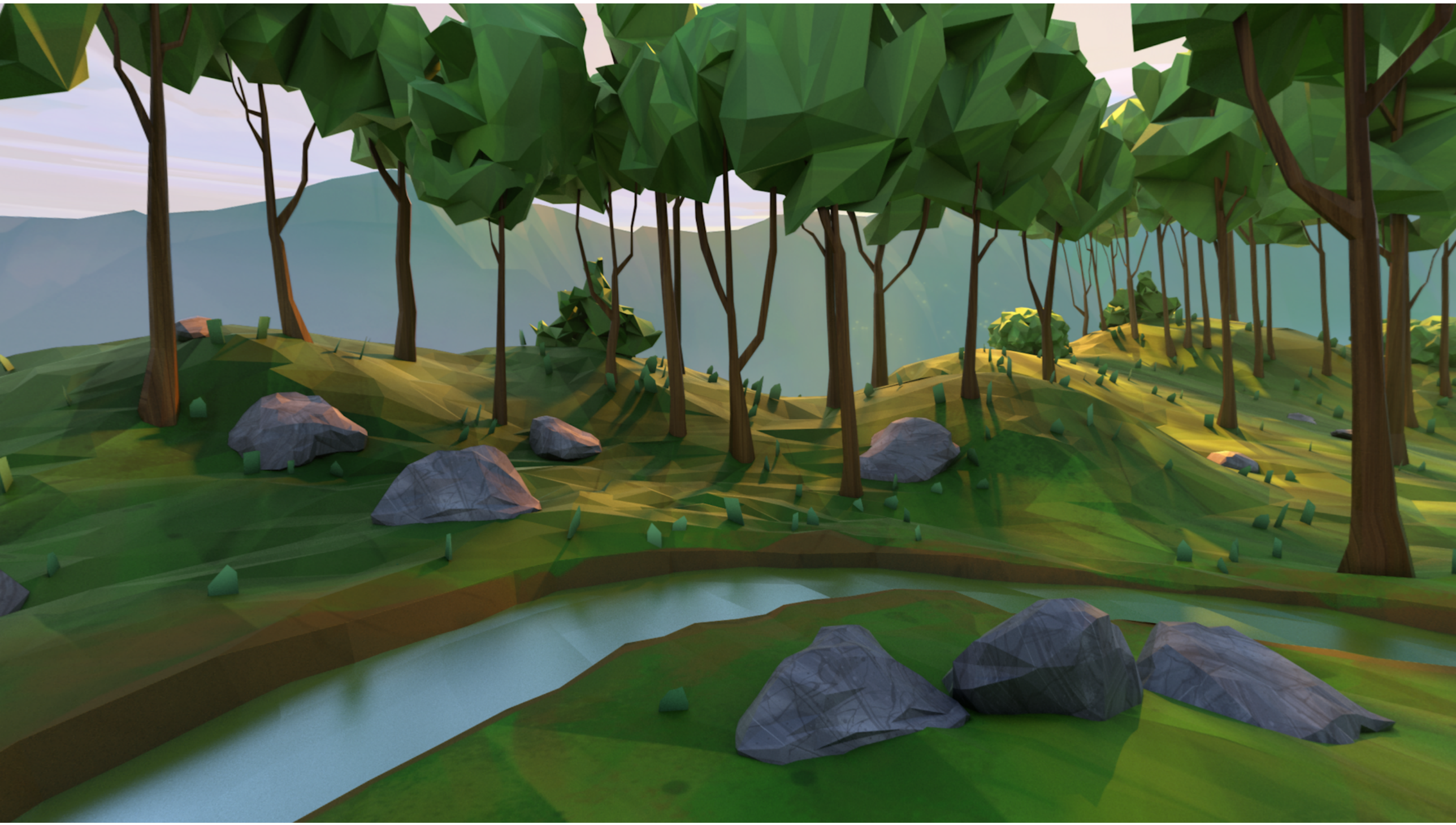
High latency

- Reduces presence
- causes VR sickness
- Inefficient interaction

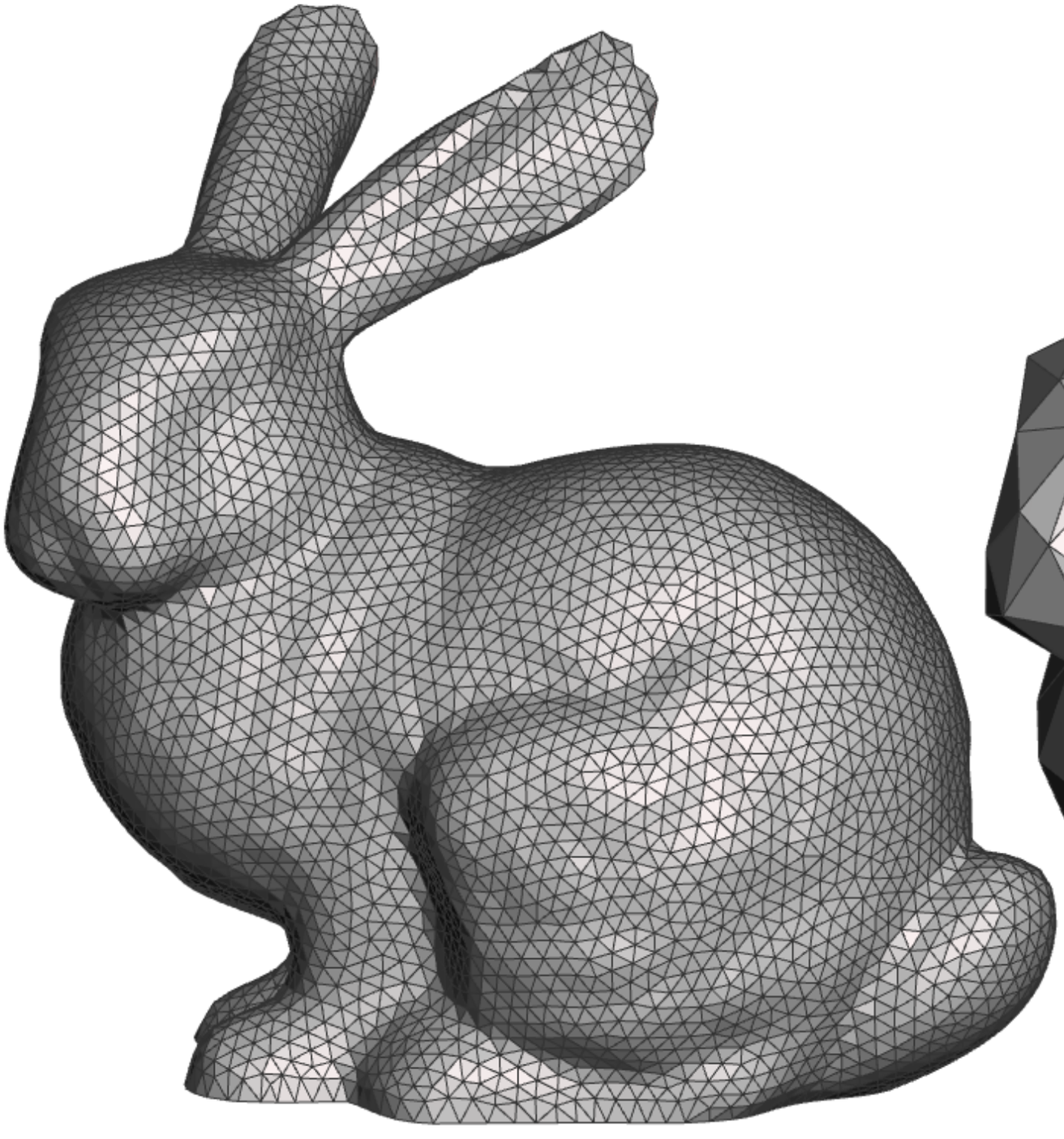
Remedies

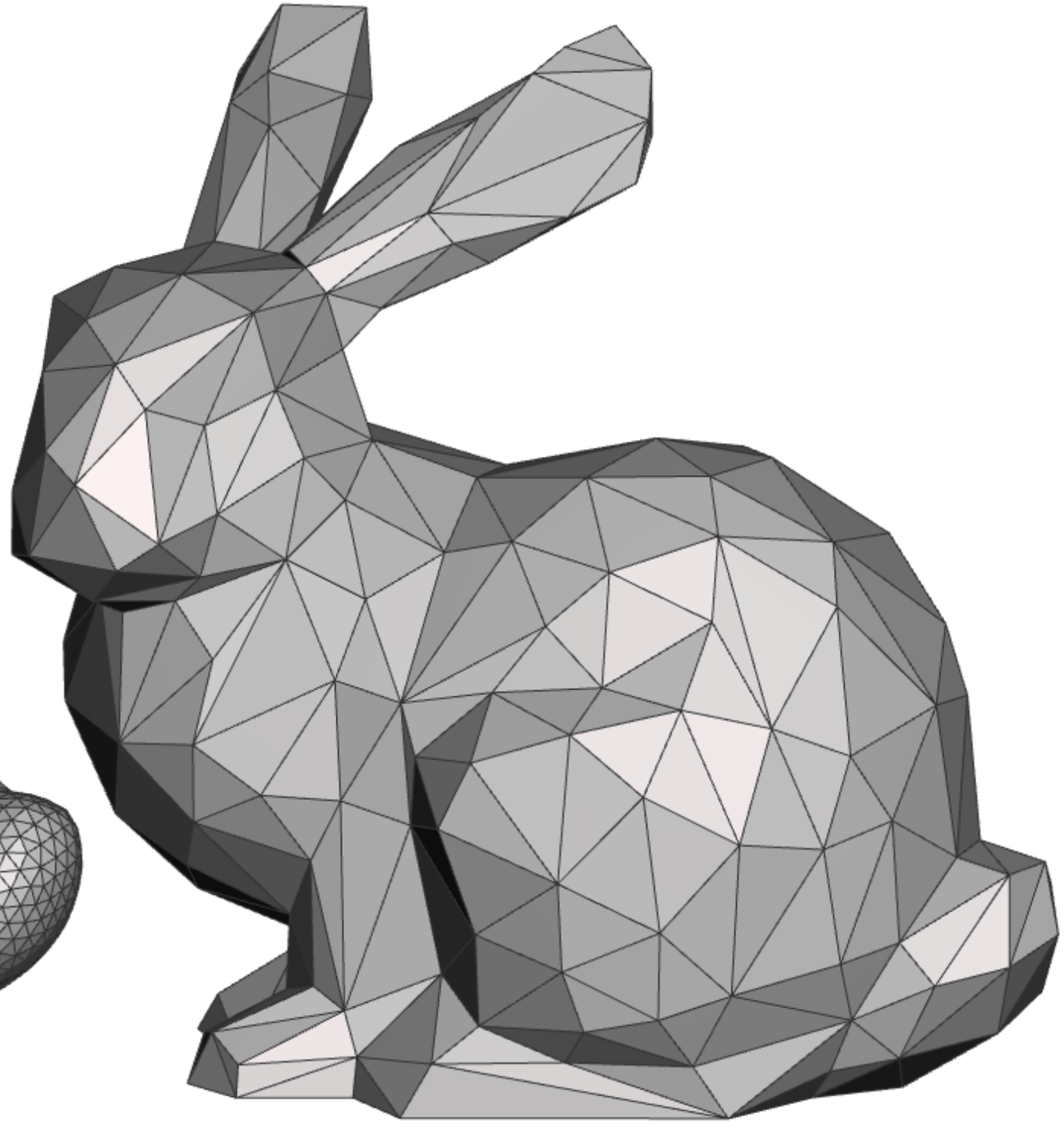
- Lower complexity
- Improve rendering pipeline performance
  - remove pipeline delays
  - user prediction to estimate future viewpoints
  - Shift/distort the rendered image to compensate for missing frames.

Simplifying



Simplifying





low poly

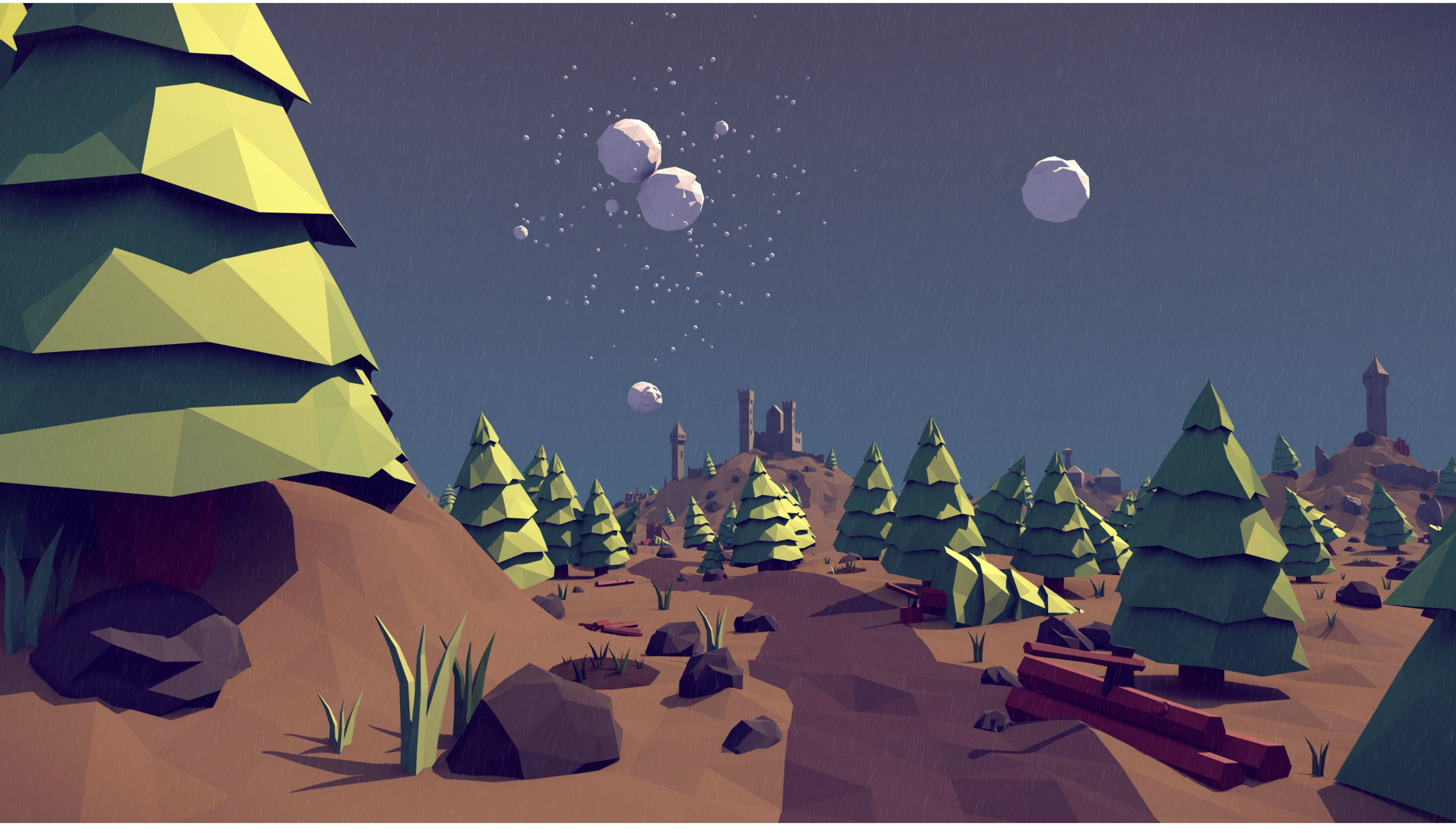
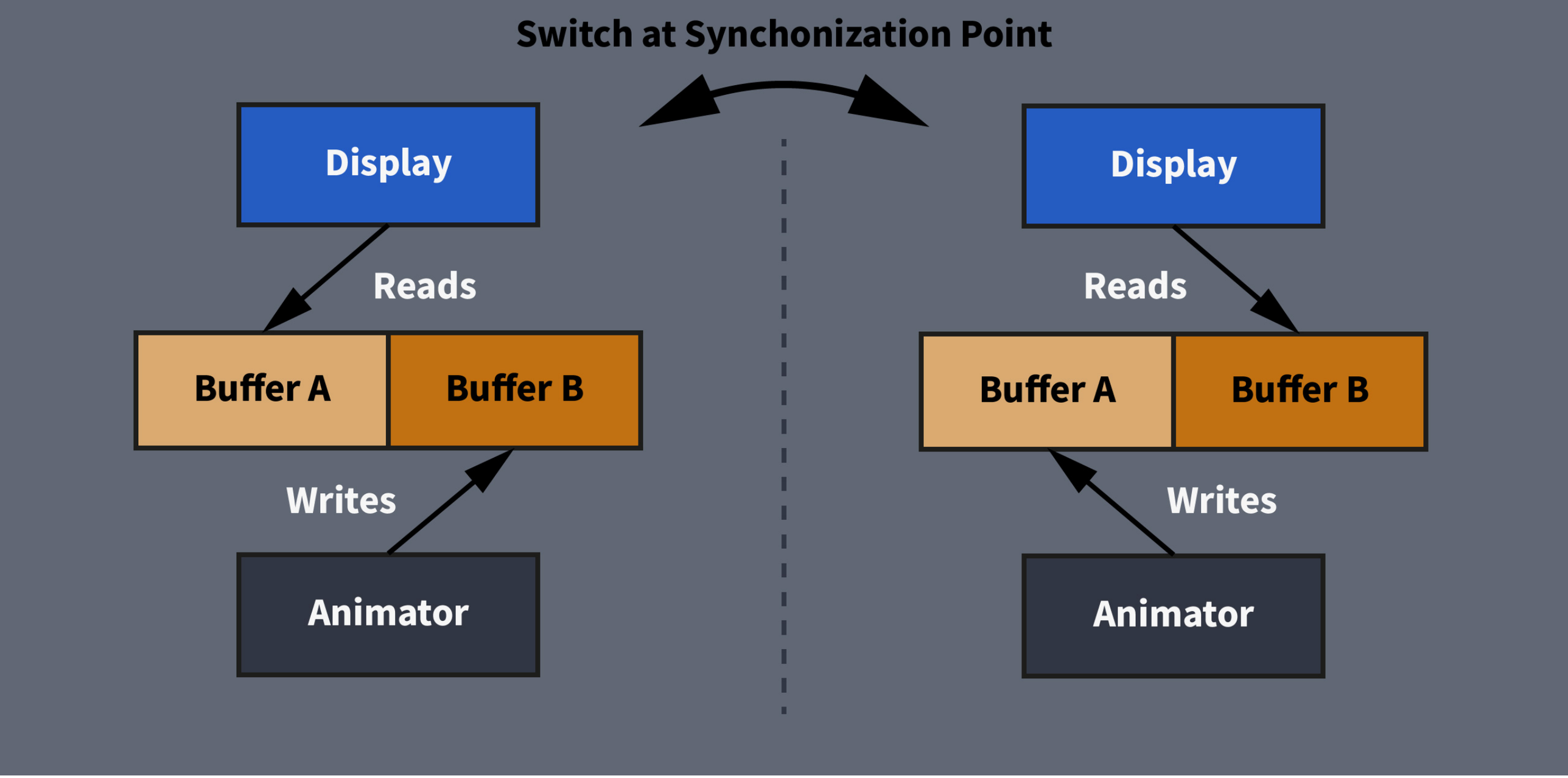


Image tearing

Double-Buffering



How It Works:

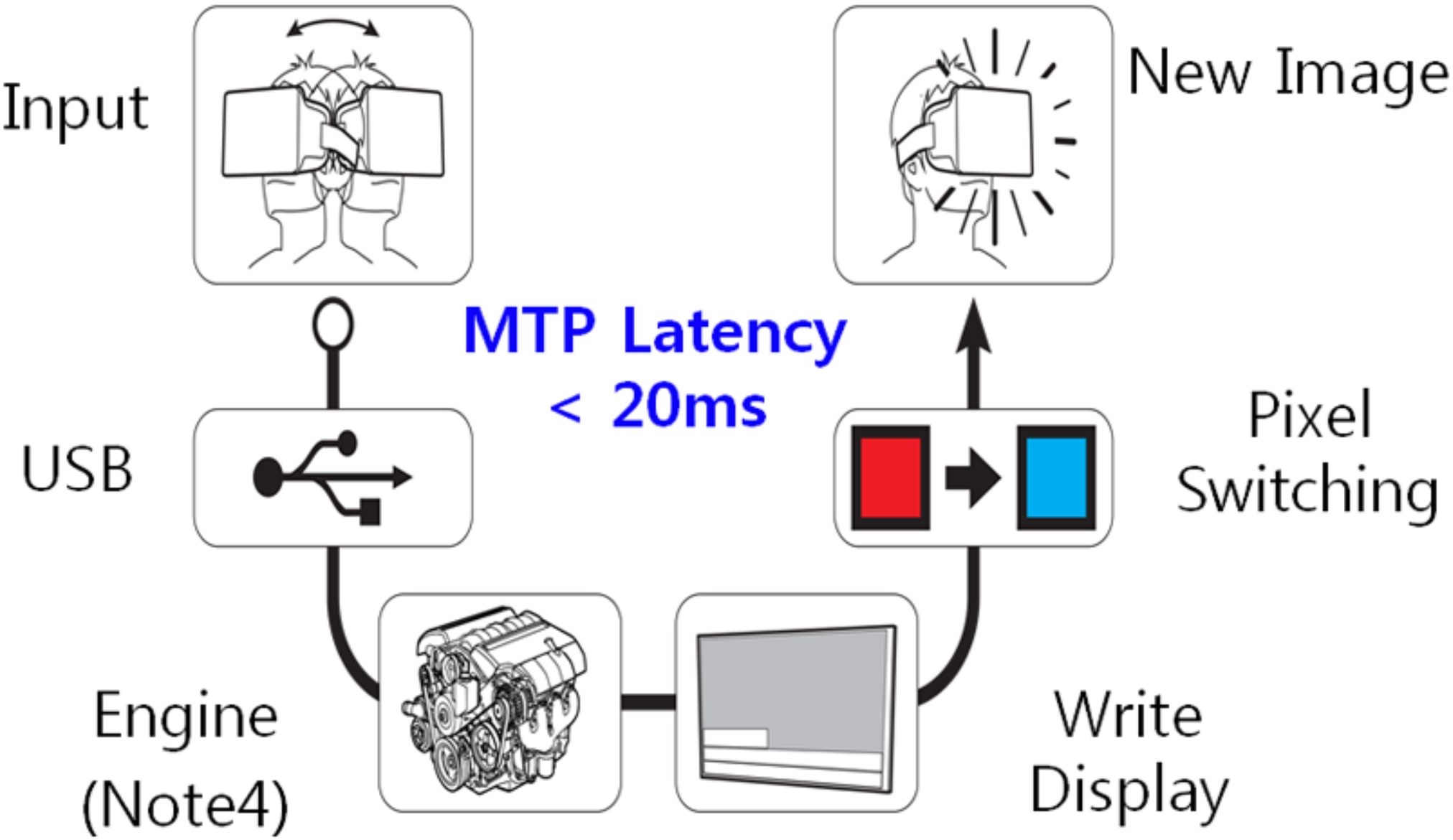
- There are two buffers:
  1. **Front buffer** – The buffer currently being displayed on the screen.
  2. **Back buffer** – The buffer where the next frame is drawn.
- Once the frame is fully rendered in the **back buffer**, the buffers **swap**, making the new frame visible.

Benefits:

✅ **Eliminates flickering** by ensuring only fully rendered frames are displayed.
✅ **Prevents tearing** by synchronizing frame updates with the display refresh rate.
✅ **Smooth animations** and improved rendering ❌ increases latency

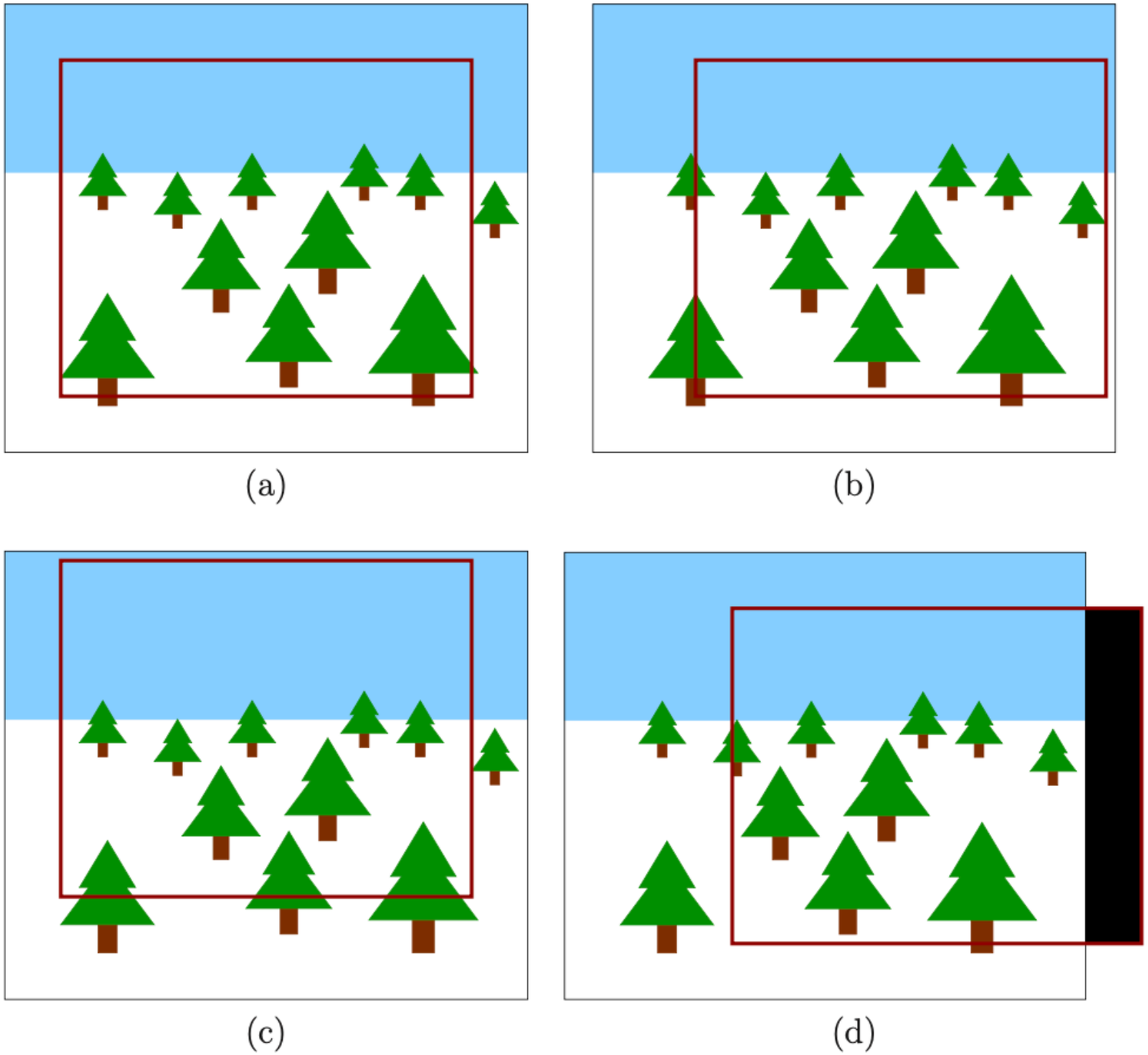
Prediction

Head has a lot of inertia



▲*Principles of the Gear VR performance*

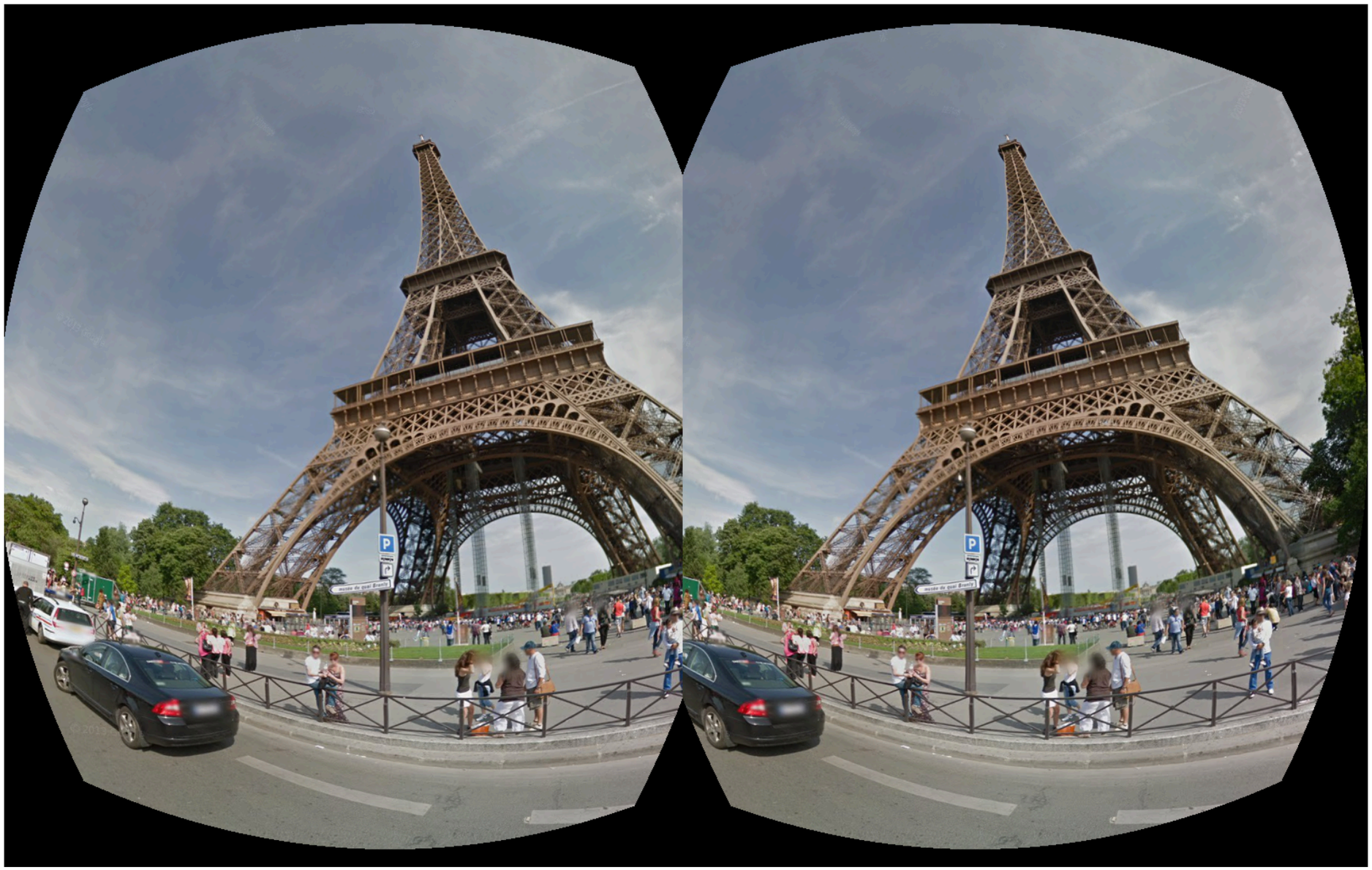
Post rendering Image warping



How It Works:

- The rendered image is **transformed** using geometric operations like **perspective correction, distortion removal, or viewpoint changes**.
- This is done **without re-rendering** the entire scene, making it computationally efficient.

Immersive photos/videos



VR streetview

Immersive videos

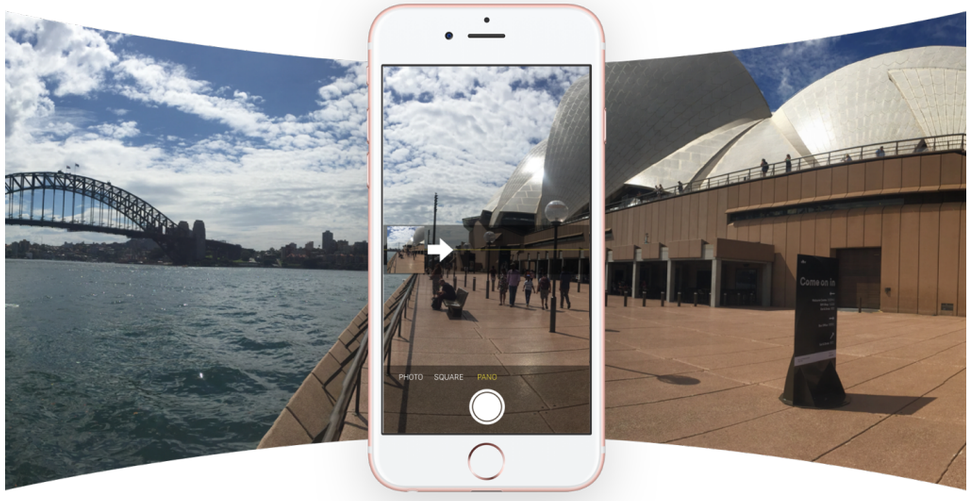
Hulu: Virtually Mike and Norah

Projection

- Video/photo on VR is form of texture mapping
- Mismatch frame rates (24fps versus 60-96hz)
- VR allows for simulating 3D movies
- Panoramic photo = photosphere
- Panoramic video = videosphere

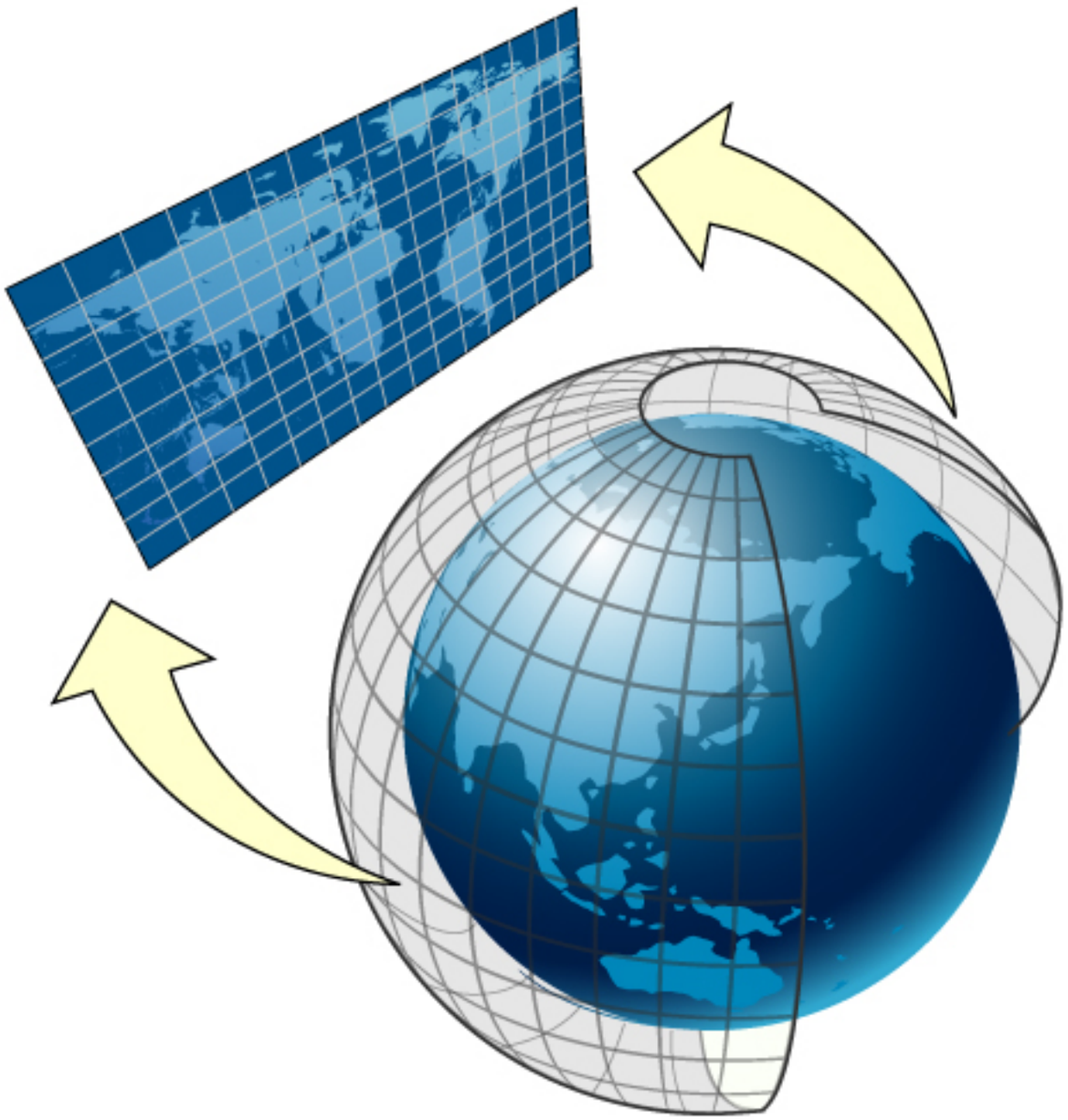
capturing a wider FOV

cameras have a limited FOV



Camera rigs

mapping



light field technology