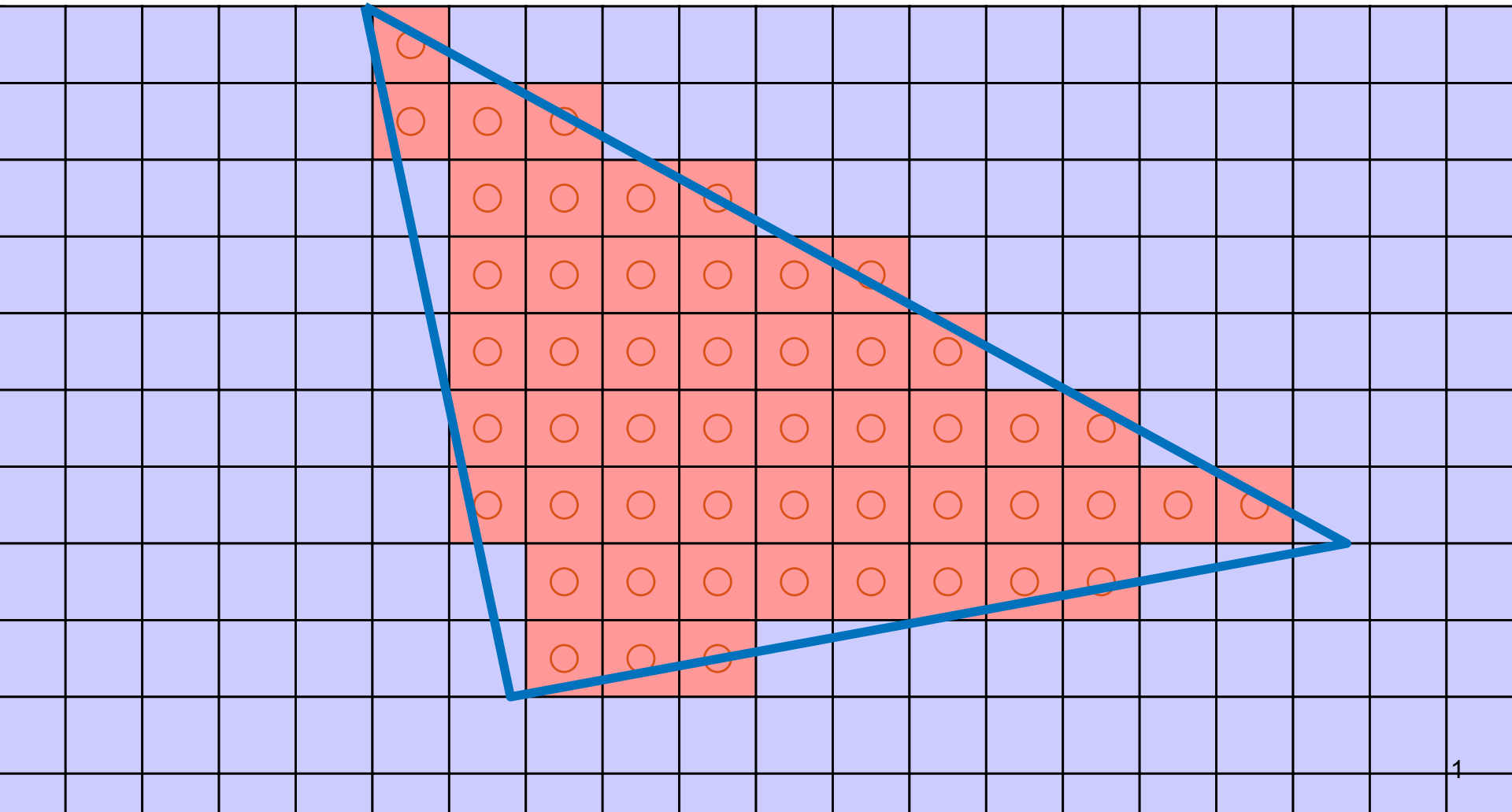
Rasterization & The Graphics Pipeline

15.1 Basics: Rasterization vs. Ray Tracing

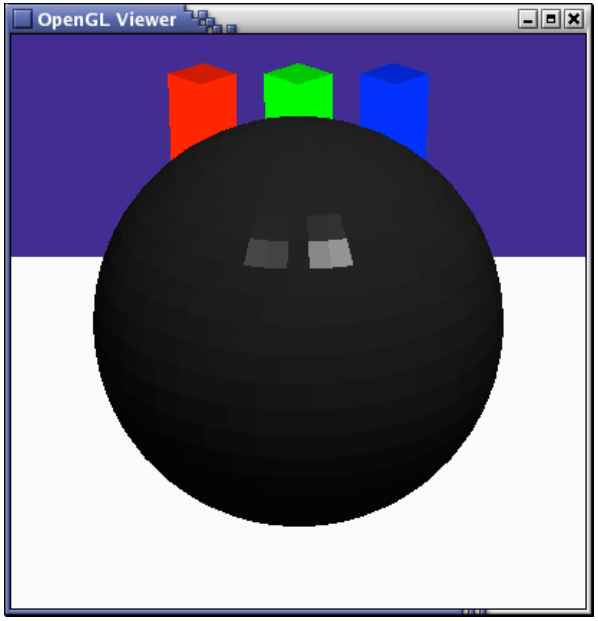


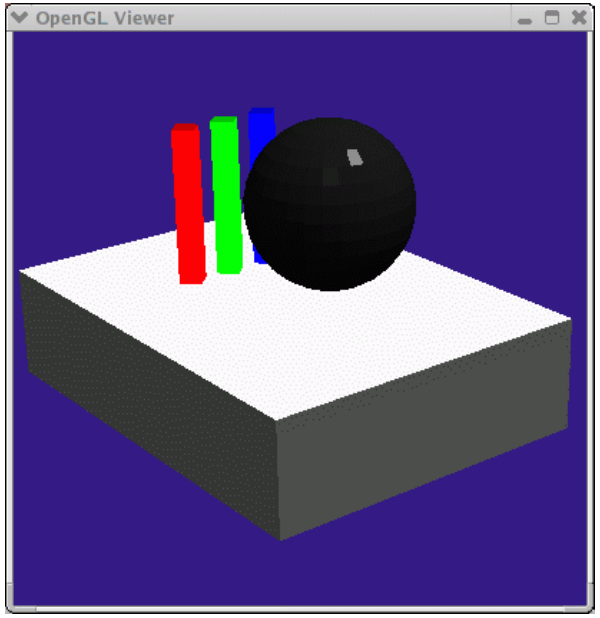
In This Video

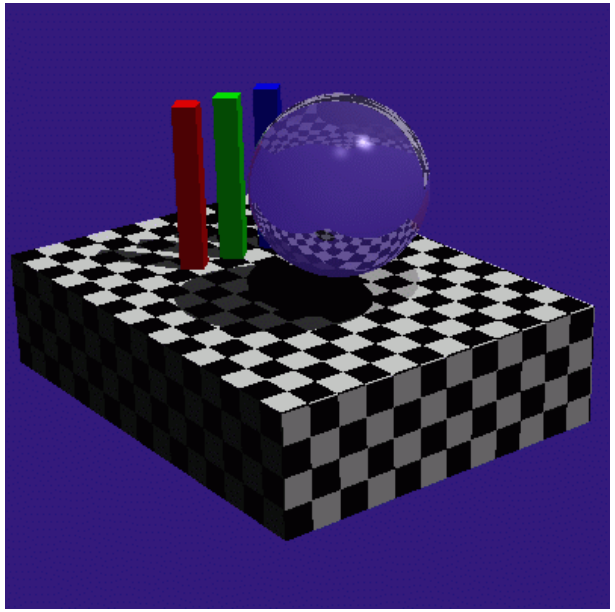
- What is real-time graphics mostly based on?
- Rasterization vs. ray tracing
  - Different ordering of pixel/primitive combinations
  - Working sets of the two algorithms
    - What is streamed over and then forgotten?
    - In contrast, what is kept in memory for random access?
  - Advantages and disadvantages

How Do We Render Interactively?

- Use graphics hardware, via OpenGL or DirectX
  - OpenGL is multi-platform, DirectX is MS only





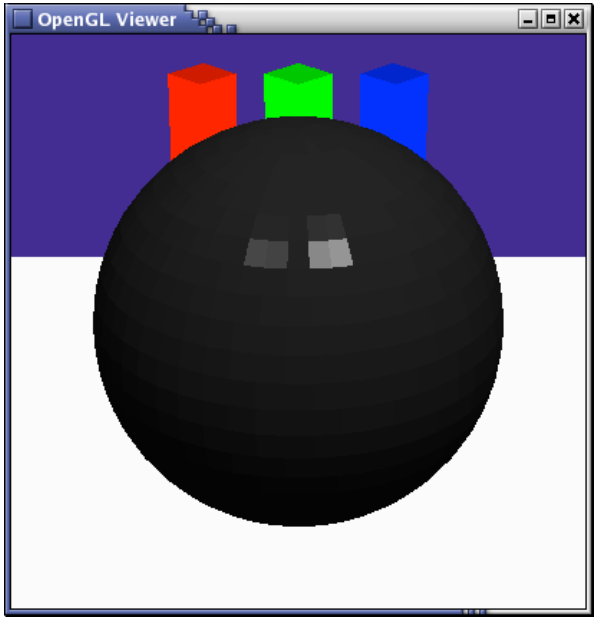


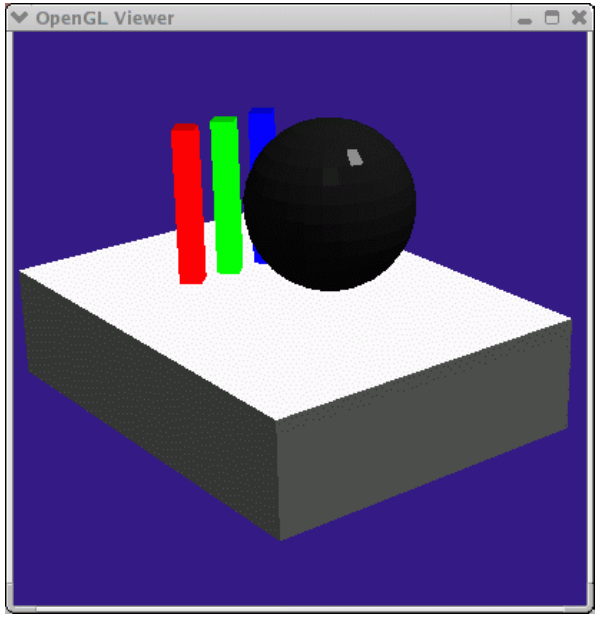
- *OpenGL rendering* Very basic!

Our ray tracer

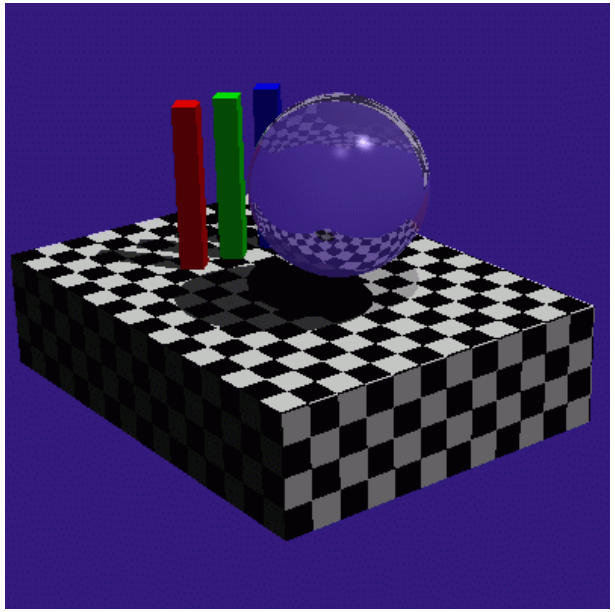
How Do We Render Interactively?

- Use graphics hardware, via OpenGL or DirectX
  - OpenGL is multi-platform, DirectX is MS only





OpenGL rendering



Our ray tracer

- Most global effects available in ray tracing will be sacrificed for speed, but some can be approximated

All Is Not Lost

- Example: “Screen space reflection” is possible..
- ... but MUCH more cumbersome than ray tracing

Uncharted 4

Quantum Break

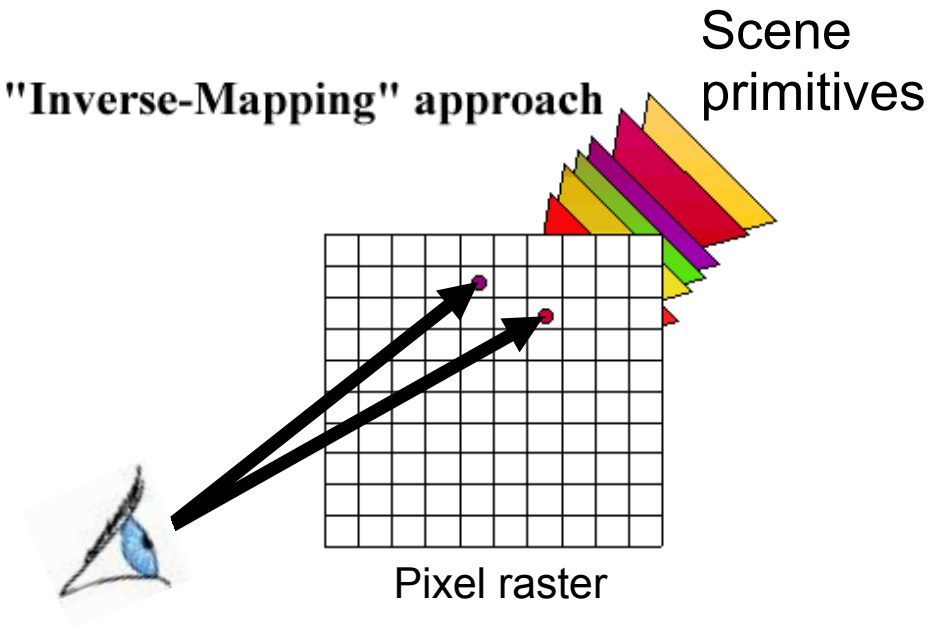
Battlefield 1 by DICE

Ray Casting vs. GPUs for Triangles

Ray Casting

```
For each pixel (ray)
  For each triangle
    Does ray hit triangle?
```

Keep closest hit



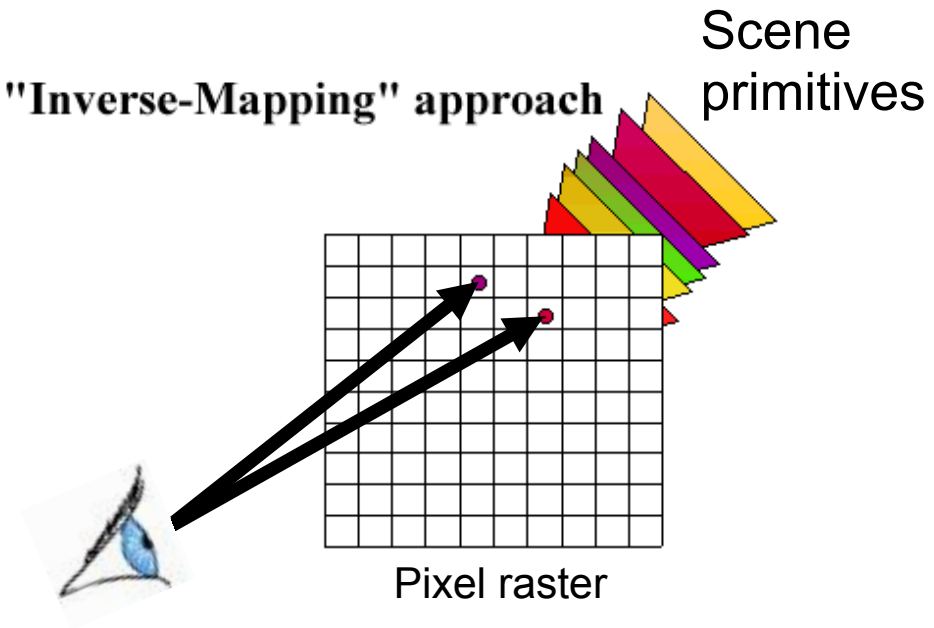
Ray Casting vs. GPUs for Triangles

Ray Casting

```
For each pixel (ray)
  For each triangle
    Does ray hit triangle?
```

Keep closest hit

Last time we replaced this with a traversal of a Bounding Volume Hierarchy (BVH)



Ray Casting vs. GPUs for Triangles

Ray Casting

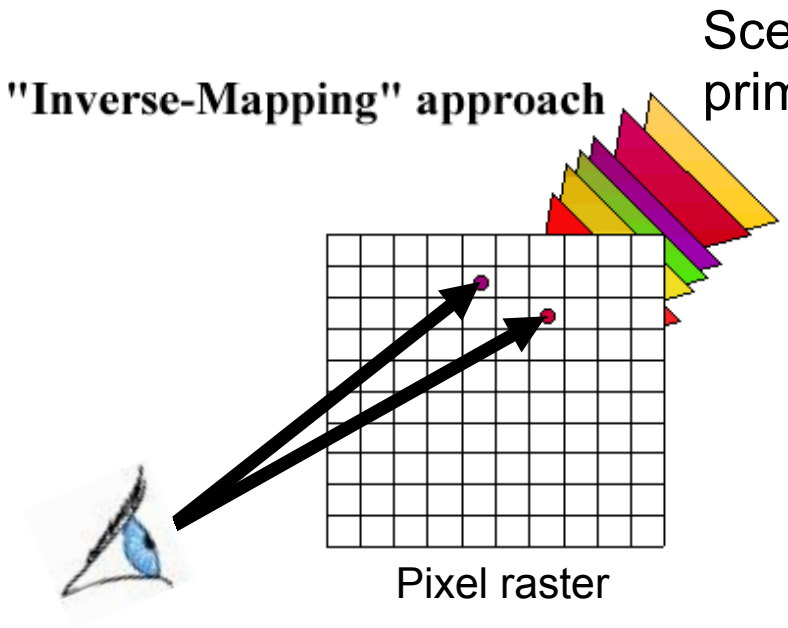
```
For each pixel (ray)
  For each triangle
    Does ray hit triangle?
```

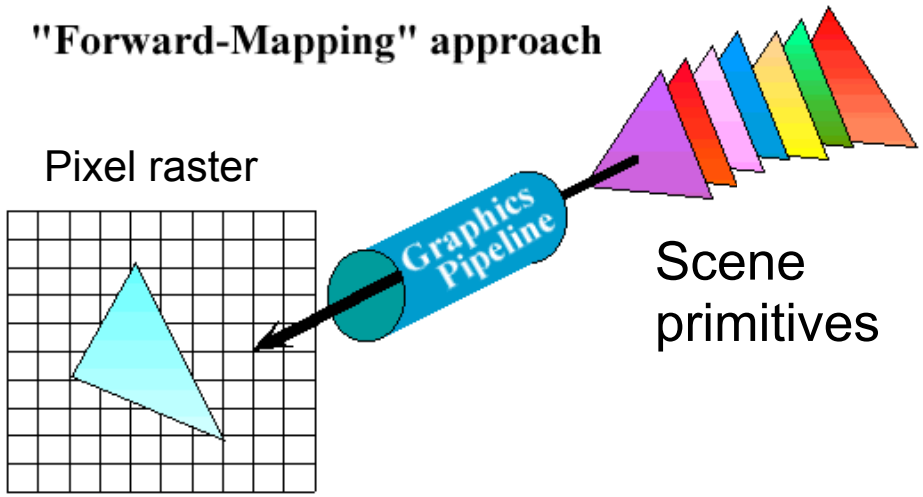
Keep closest hit

GPU

```
For each triangle
  For each pixel
    Does triangle cover pixel?
```

Keep closest hit



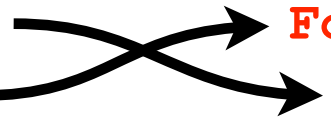


Ray Casting vs. GPUs for Triangles

Ray Casting

```
For each pixel (ray)
  For each triangle
    Does ray hit triangle?
```

Keep closest hit



GPU

```
For each triangle
  For each pixel
    Does triangle cover pixel?
```

Keep closest hit

It's just a different order of the loops!

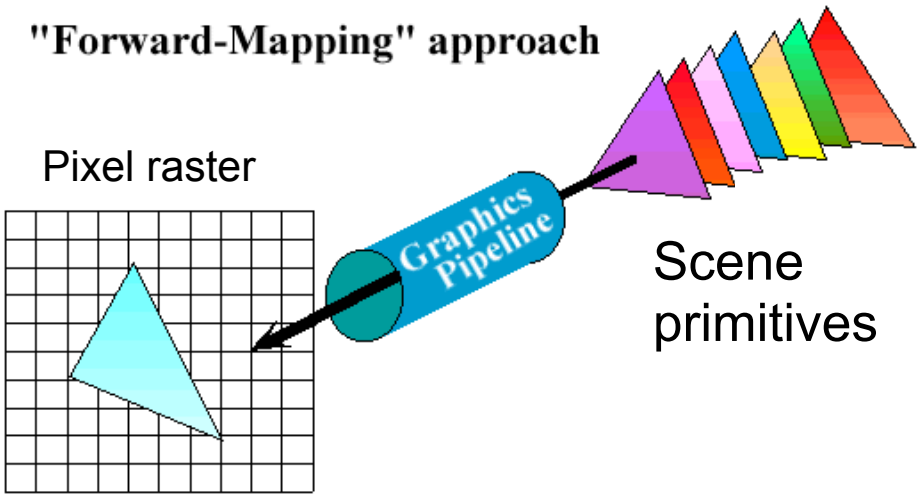
GPUs do Rasterization

- The process of taking a triangle and figuring out which pixels it covers is called **rasterization**

GPU

```
For each triangle
  For each pixel
    Does triangle cover pixel?
```

Keep closest hit



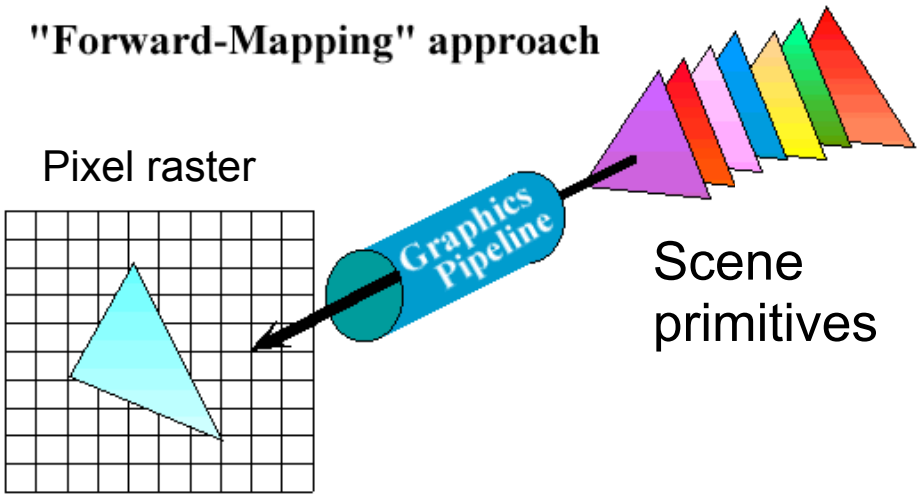
GPUs do Rasterization

- The process of taking a triangle and figuring out which pixels it covers is called **rasterization**
- We've seen acceleration structures for ray tracing; rasterization is not stupid either
  - We're not actually going to test *all* pixels for each triangle

GPU

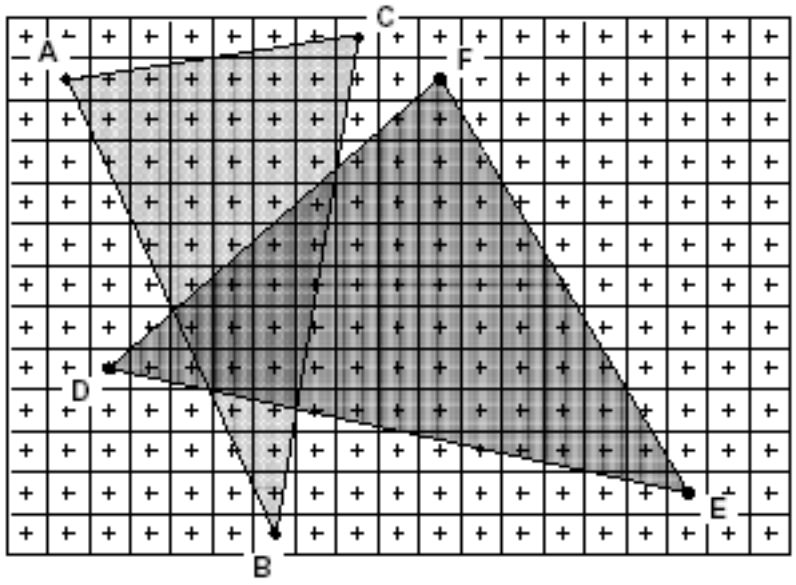
```
For each triangle
  For each pixel
    Does triangle cover pixel?
```

Keep closest hit

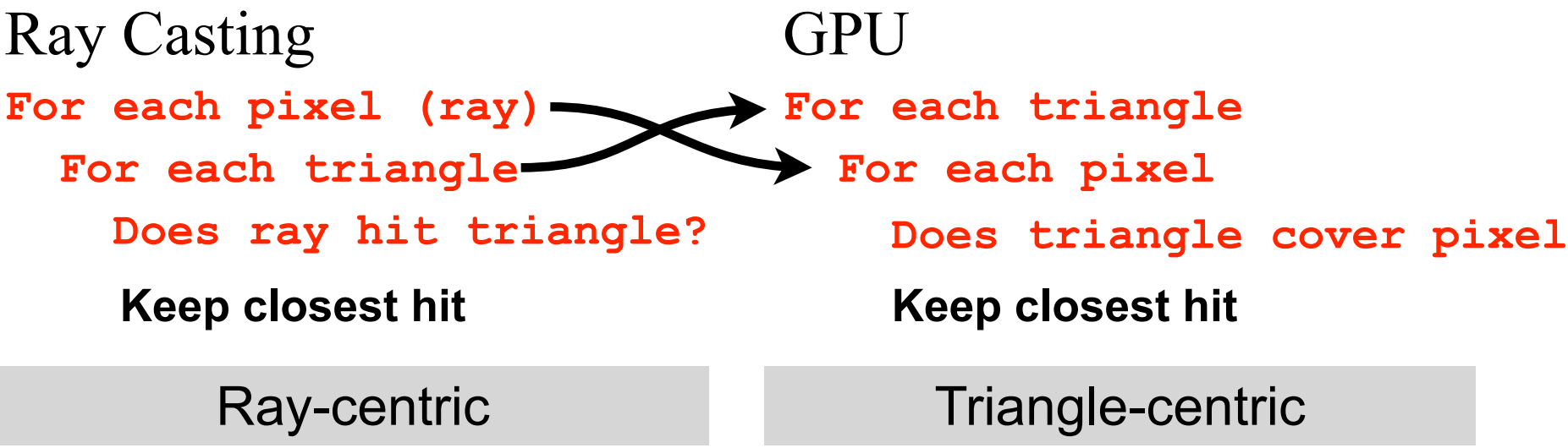


Rasterization ("Scan Conversion")

- Given a triangle's vertices & extra info for shading, figure out which pixels to "turn on" to render the primitive
- Compute illumination values to "fill in" the pixels within the primitive
- At each pixel, keep track of the closest primitive (z-buffer)
  - Only overwrite if triangle being drawn is closer than the previous triangle in that pixel

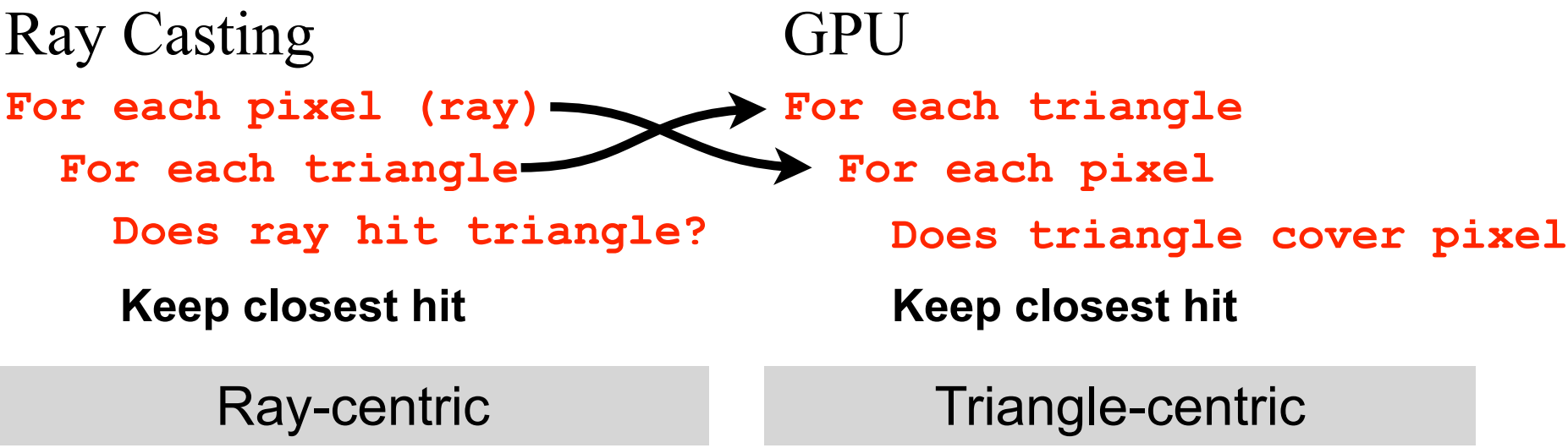


What are the Main Differences?



- What needs to be stored in memory in each case?

What are the Main Differences?



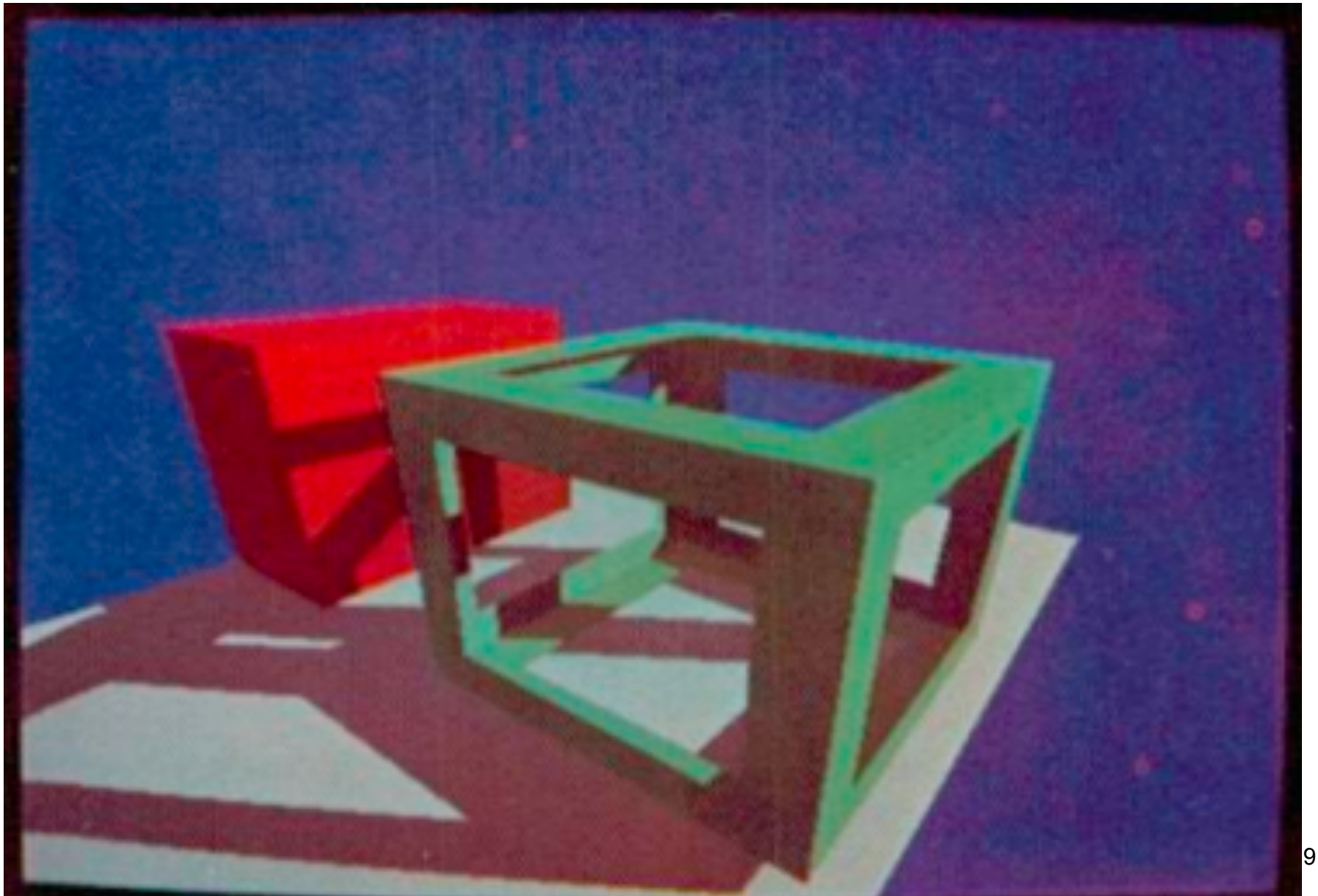
- In this basic form, ray tracing needs the entire scene description in memory at once
  - Then, can sample the image completely freely
- The rasterizer only needs one triangle at a time, *plus* the entire image and associated depth information for all pixels

Rasterization Advantages

- Modern scenes are more complicated than images
  - A 1920x1080 frame at 64-bit color and 32-bit depth per pixel is 24MB (not that much)
    - Of course, if we have more than one sample per pixel this gets larger, but e.g. 4x supersampling is still a relatively comfortable ~100MB
    - ***4K gets to hundreds of megabytes!***
  - Our scenes are routinely larger than this in bytes
    - This wasn't always true...

Rasterization Advantages

Weiler, Atherton 1977

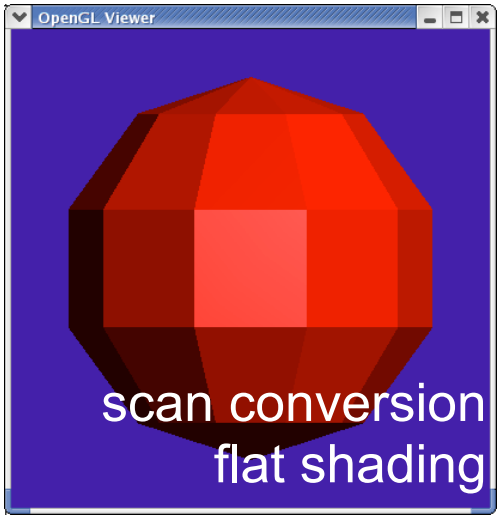


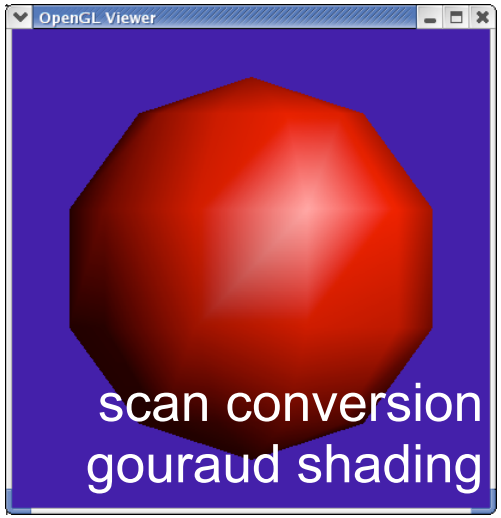
Rasterization Advantages

- Modern scenes are more complicated than images
  - A 1920x1080 frame (1080p) at 64-bit color and 32-bit depth per pixel is 24MB (not that much)
    - Of course, if we have more than one sample per pixel (later) this gets larger, but e.g. 4x supersampling is still a relatively comfortable ~100MB
  - Our scenes are routinely larger than this
    - This wasn't always true
- A rasterization-based renderer can *stream* over the triangles, no need to keep entire dataset around
  - Allows parallelism and optimization of memory systems

Rasterization Limitations

- Restricted to scan-convertible primitives
  - Pretty much: triangles
- Faceting, shading artifacts
  - *This has largely gone away with programmable per-pixel shading, though*
- No unified handling of shadows, reflection, transparency
- Potential problem of overdraw (high depth complexity)
  - Each pixel touched many times





Ray Casting / Tracing

- Advantages
  - Generality: can render anything that can be intersected with a ray
  - Easily allows recursion (shadows, reflections, etc.)

- Disadvantages
  - Harder to implement in hardware (less computation coherence, must fit entire scene in memory, worse memory behavior)
    - Not such a big point any more given general purpose GPUs
  - Was traditionally too slow for interactive applications
  - **These are not as true any more as they used to be**

Up next: the Graphics Pipeline

Battlefield 4 by DICE (Stockholm)