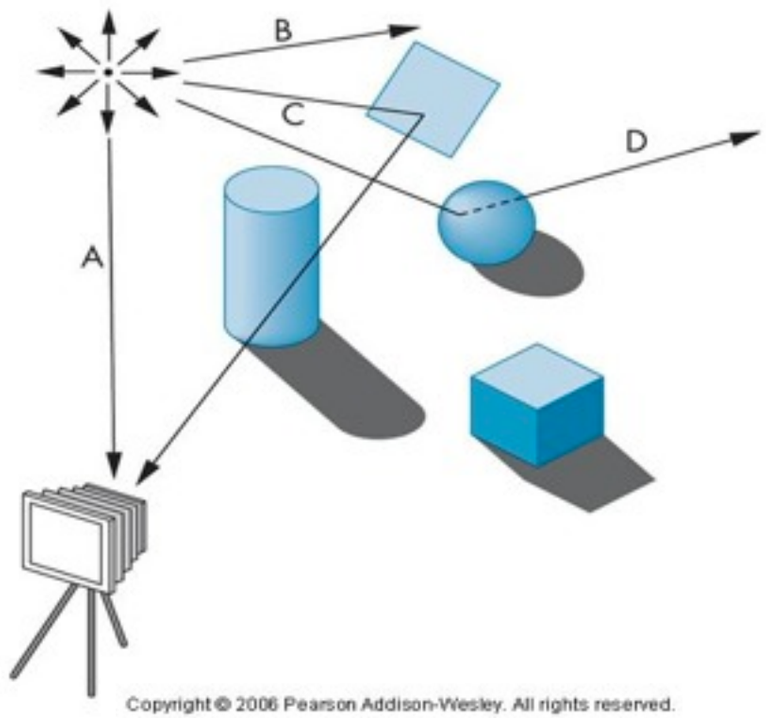
The Graphics Pipeline

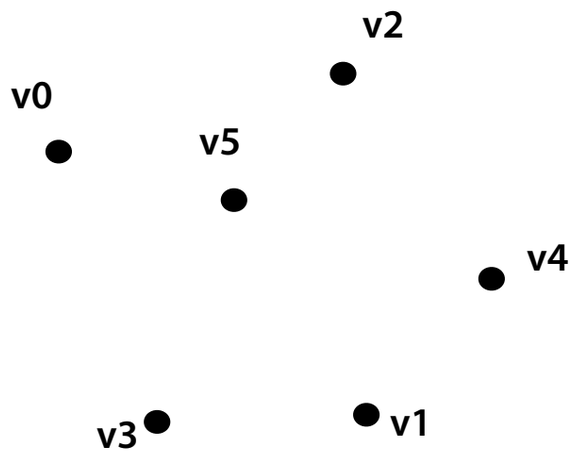
Prof. Vladlen Koltun
Computer Science Department
Stanford University

The synthetic camera model

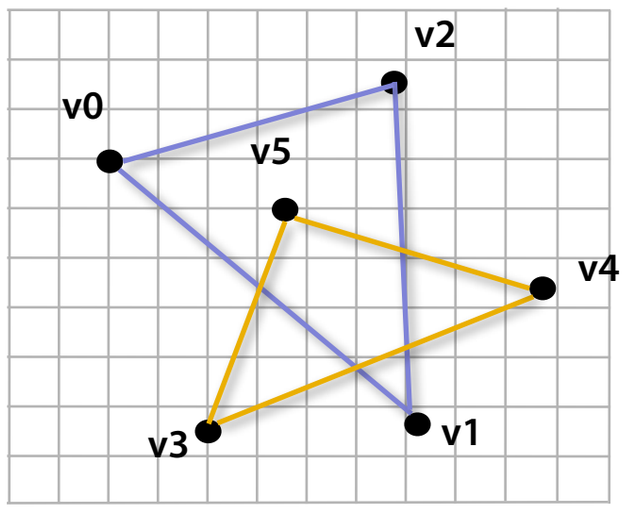


- Two components of viewing
  - Set of geometric objects that form the content of the scene
  - Viewer through which the scene is imaged

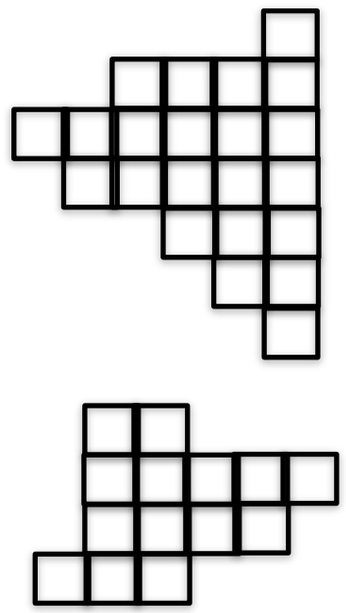
Pipeline entities



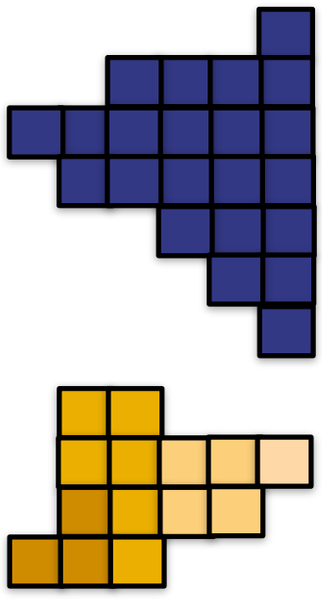
Vertices



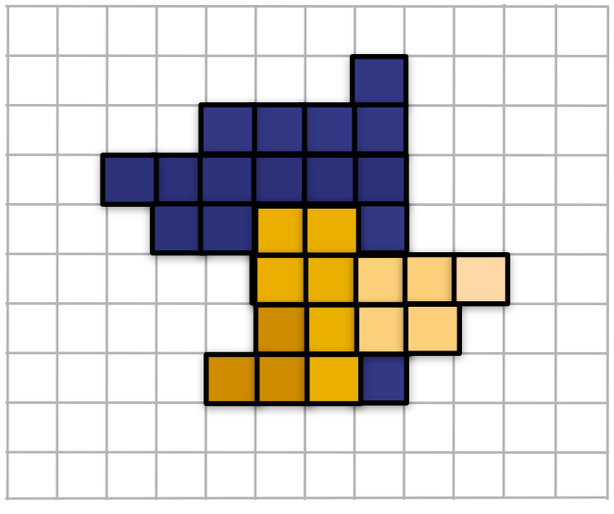
Primitives



Fragments



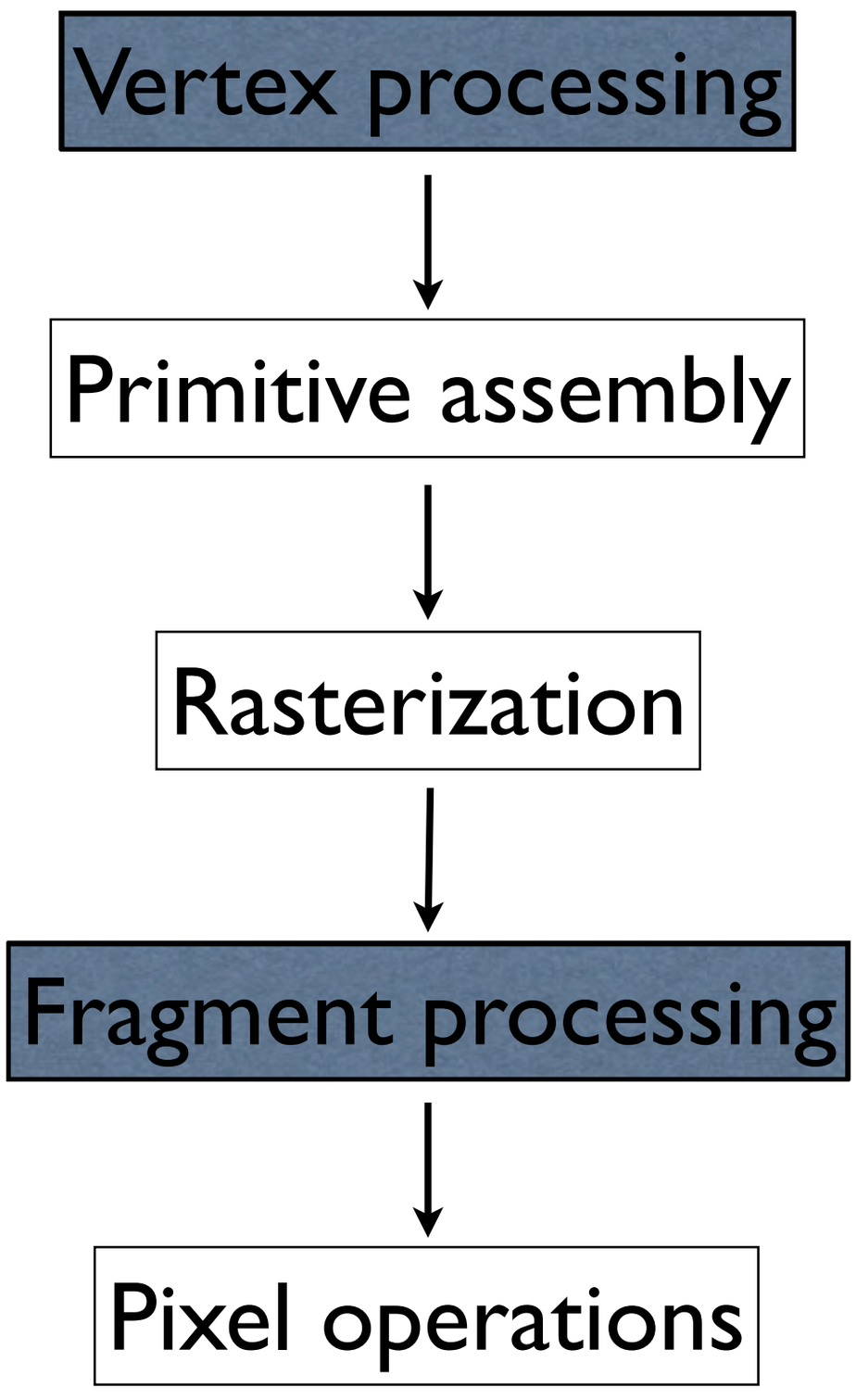
Fragments (shaded)



Pixels

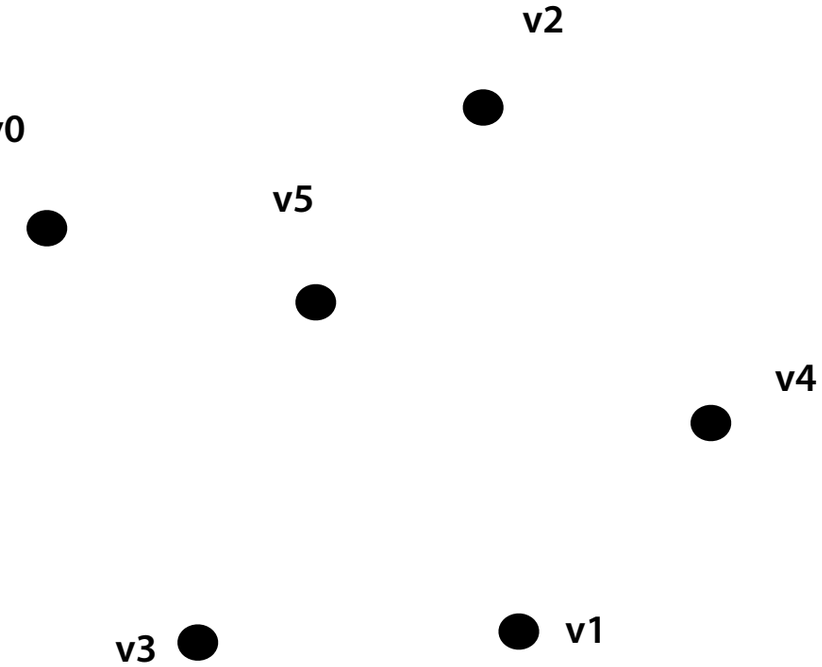
slide by Kayvon Fatahalian

The graphics pipeline



Vertex processing

Vertices are transformed into "screen space"

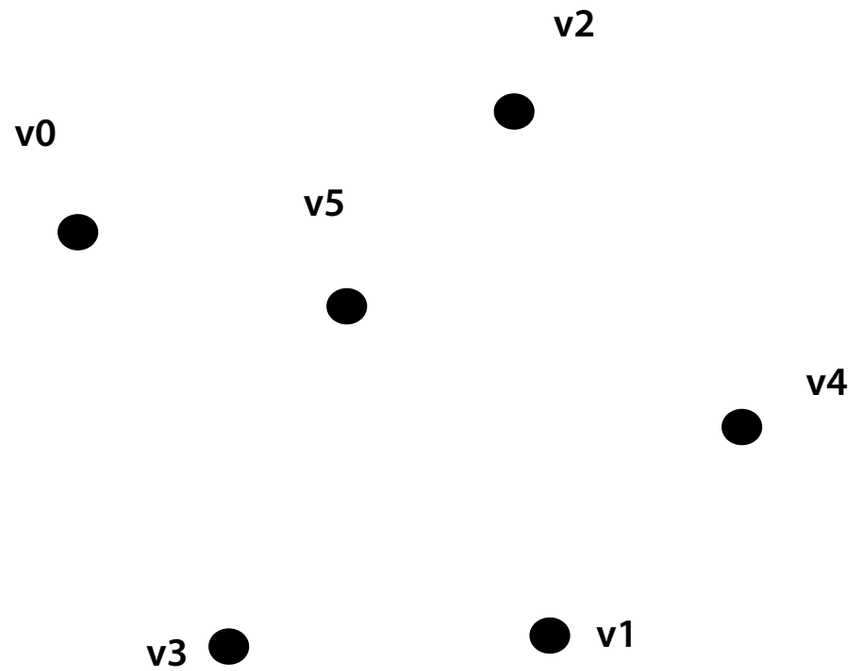


Vertices

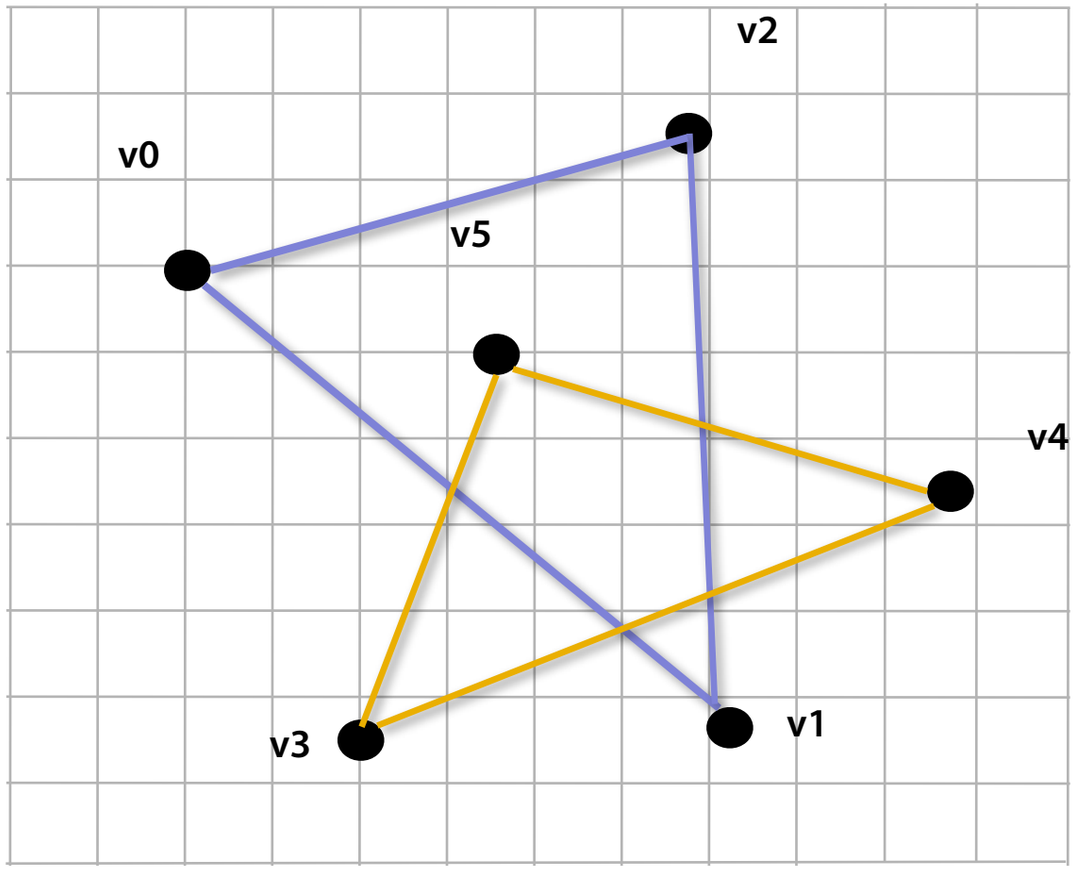
slide by Kayvon Fatahalian

Primitive processing

Then organized into primitives that are clipped and culled…



Vertices

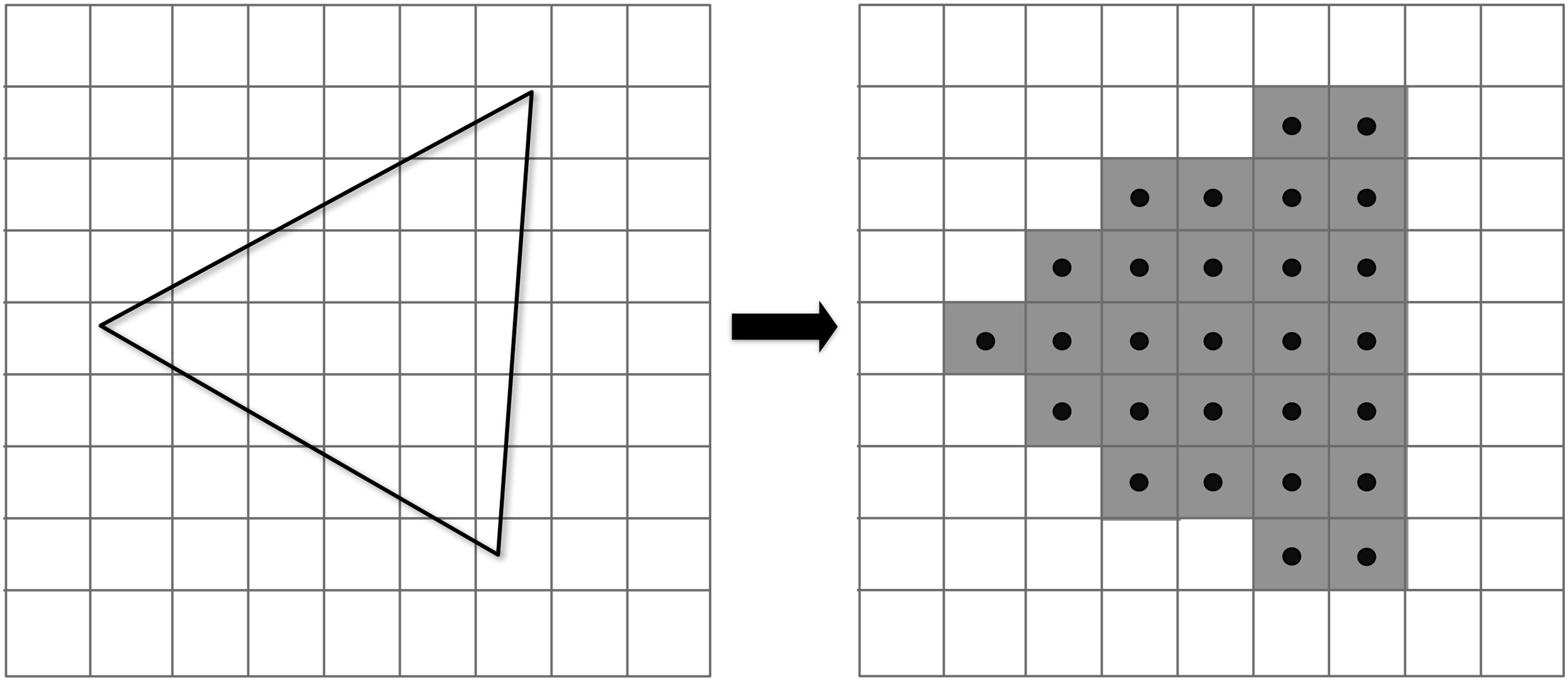


Primitives (triangles)

slide by Kayvon Fatahalian

Rasterization

Primitives are rasterized into "pixel fragments"

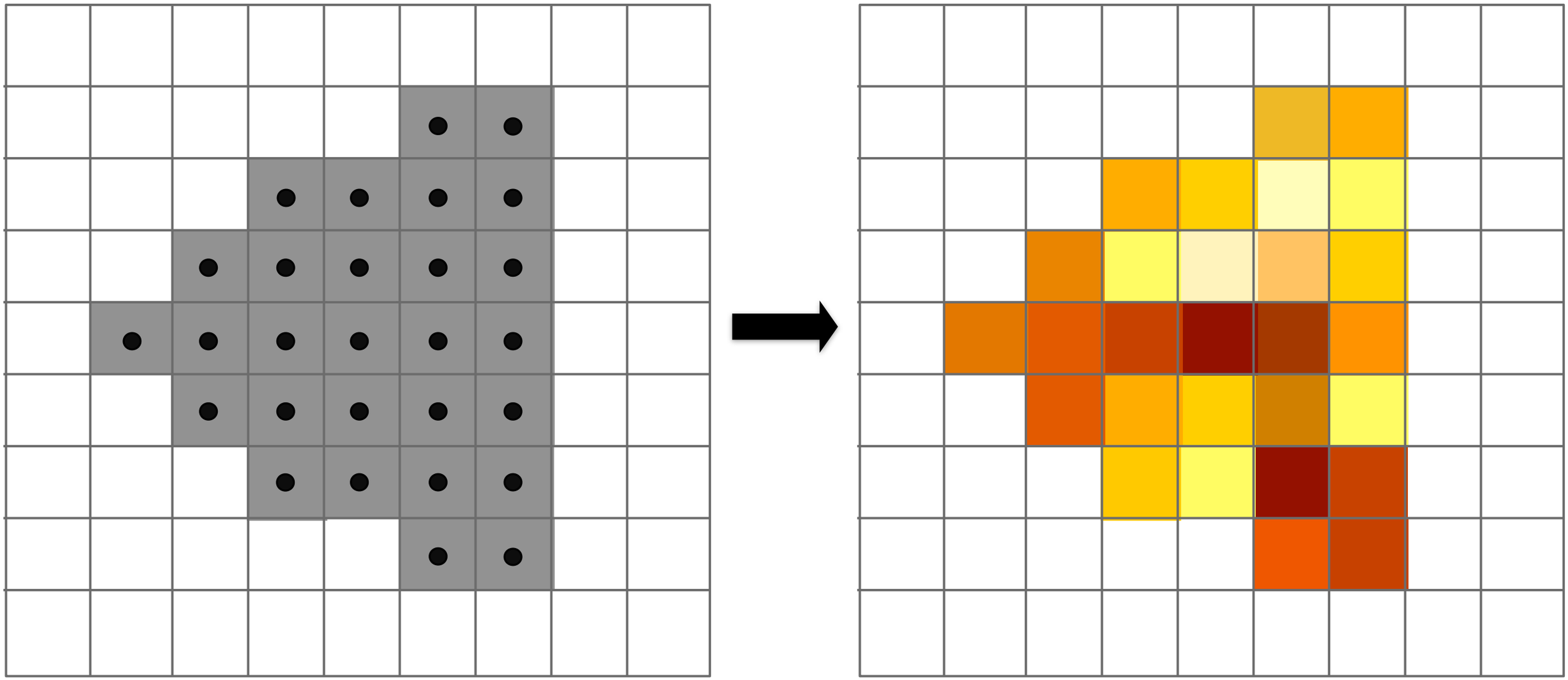


Fragments

slide by Kayvon Fatahalian

Fragment processing

Fragments are shaded to compute a color at each pixel

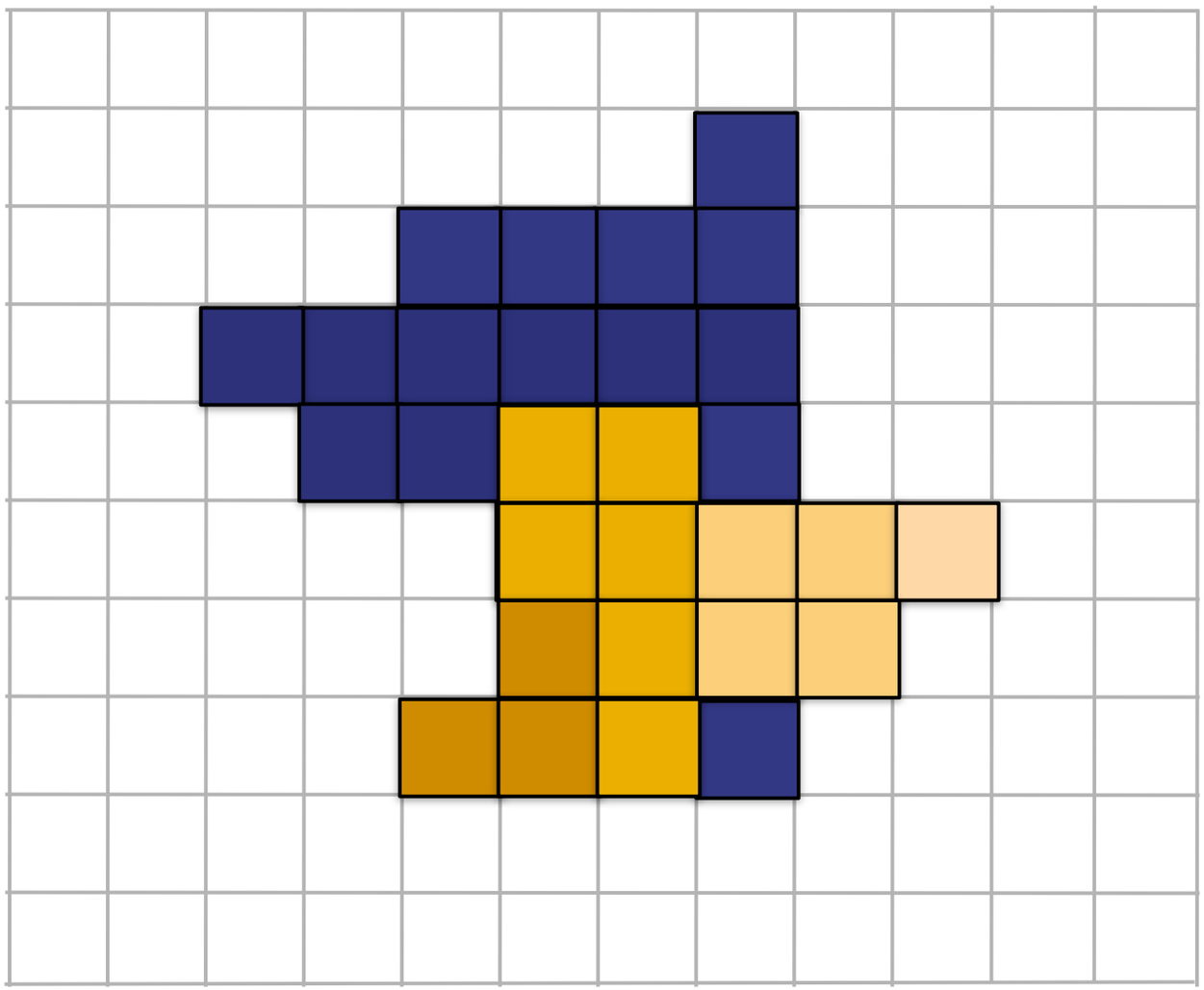


Shaded fragments

slide by Kayvon Fatahalian

Pixel operations

Fragments are blended into the frame buffer at their pixel locations (z-buffer determines visibility)

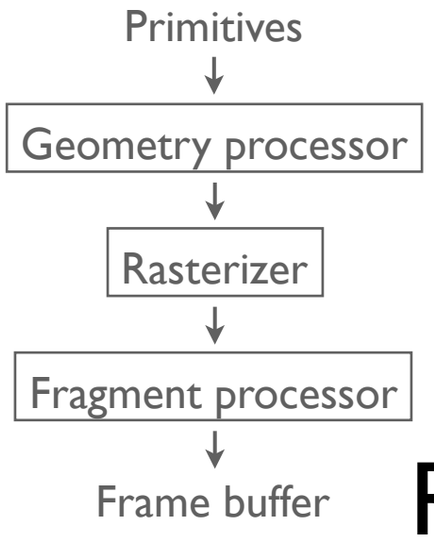


Pixels

slide by Kayvon Fatahalian

Frame buffer

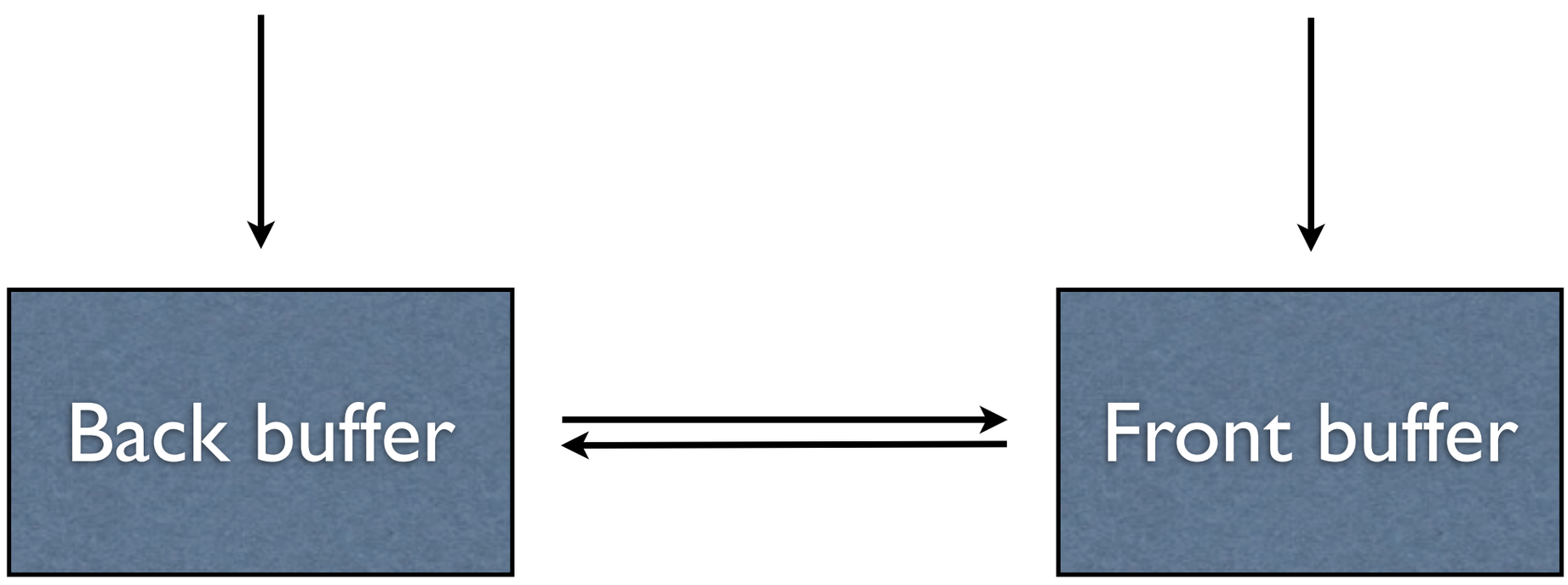
- Memory buffer used for the construction of the image.
- Not all data that passes through the frame buffer is displayed. It is like a sandbox in which the image is constructed.
- Used by the window system for display.



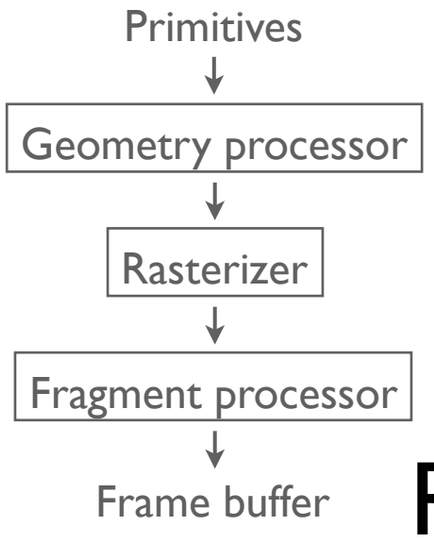
Double buffering

Rendering

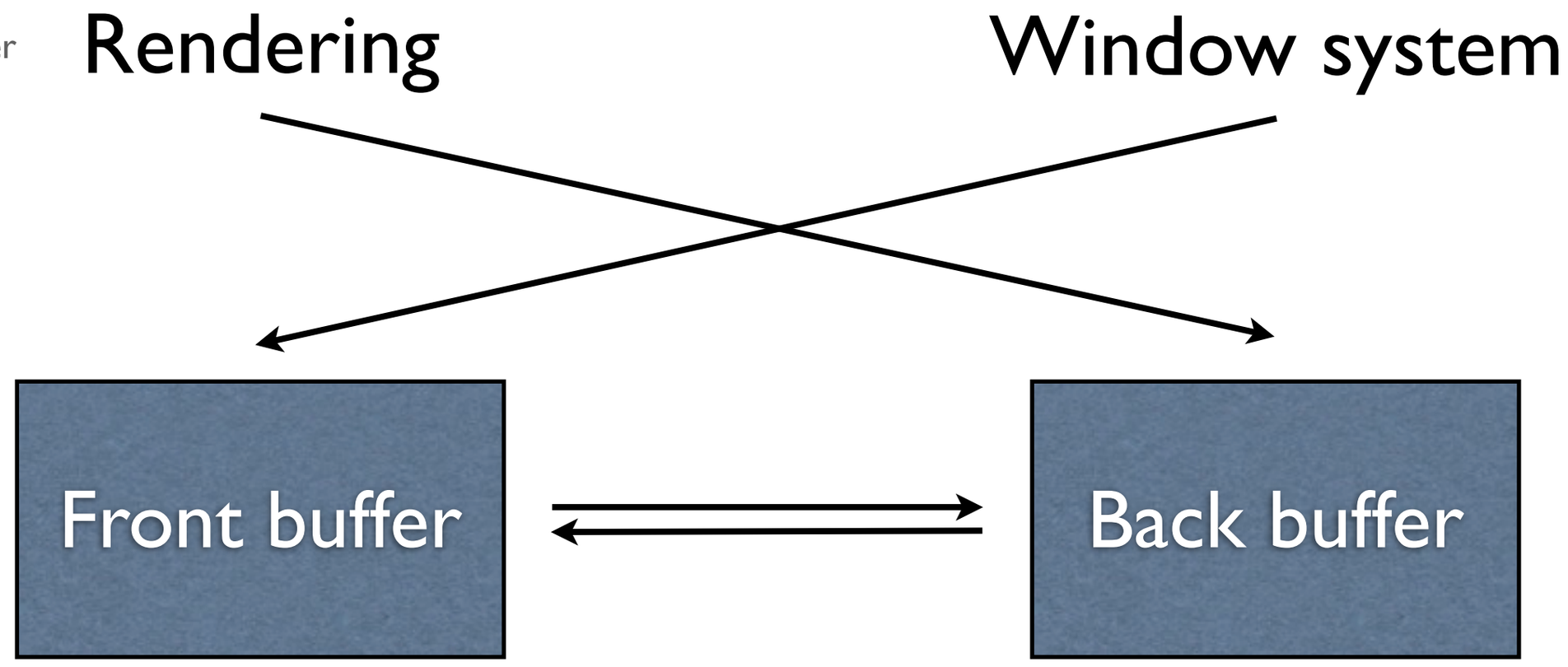
Window system



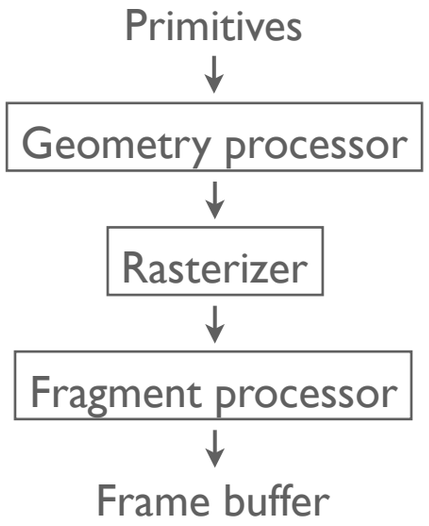
- Render into the back buffer while the window system points to the front buffer. When the next frame is assembled, swap.
- Avoids terrible visual artifacts



Double buffering



- Render into the back buffer while the window system points to the front buffer. When the next frame is assembled, swap.
- Avoids terrible visual artifacts



Advantages and disadvantages of pipeline model

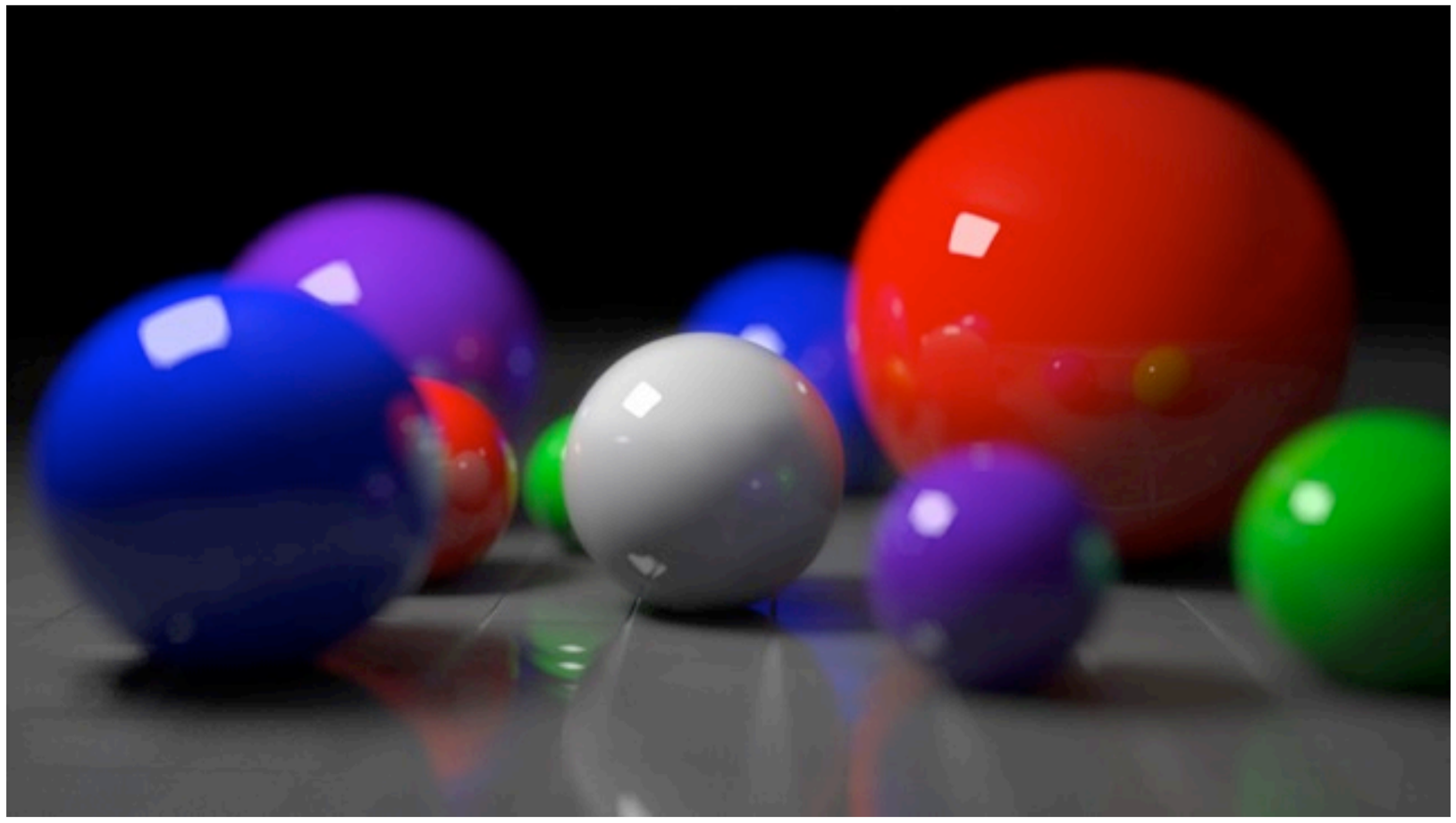
- Great for parallel processing
  - Vertices processed independently
  - Fragments processed independently
- Does not support interactions between multiple objects in the scene
  - Global illumination, shadows, reflection, refraction

Alternatives to the pipeline?

Global illumination

- Consider indirect illumination that is transmitted by means of other objects
- Primitives are no longer independent

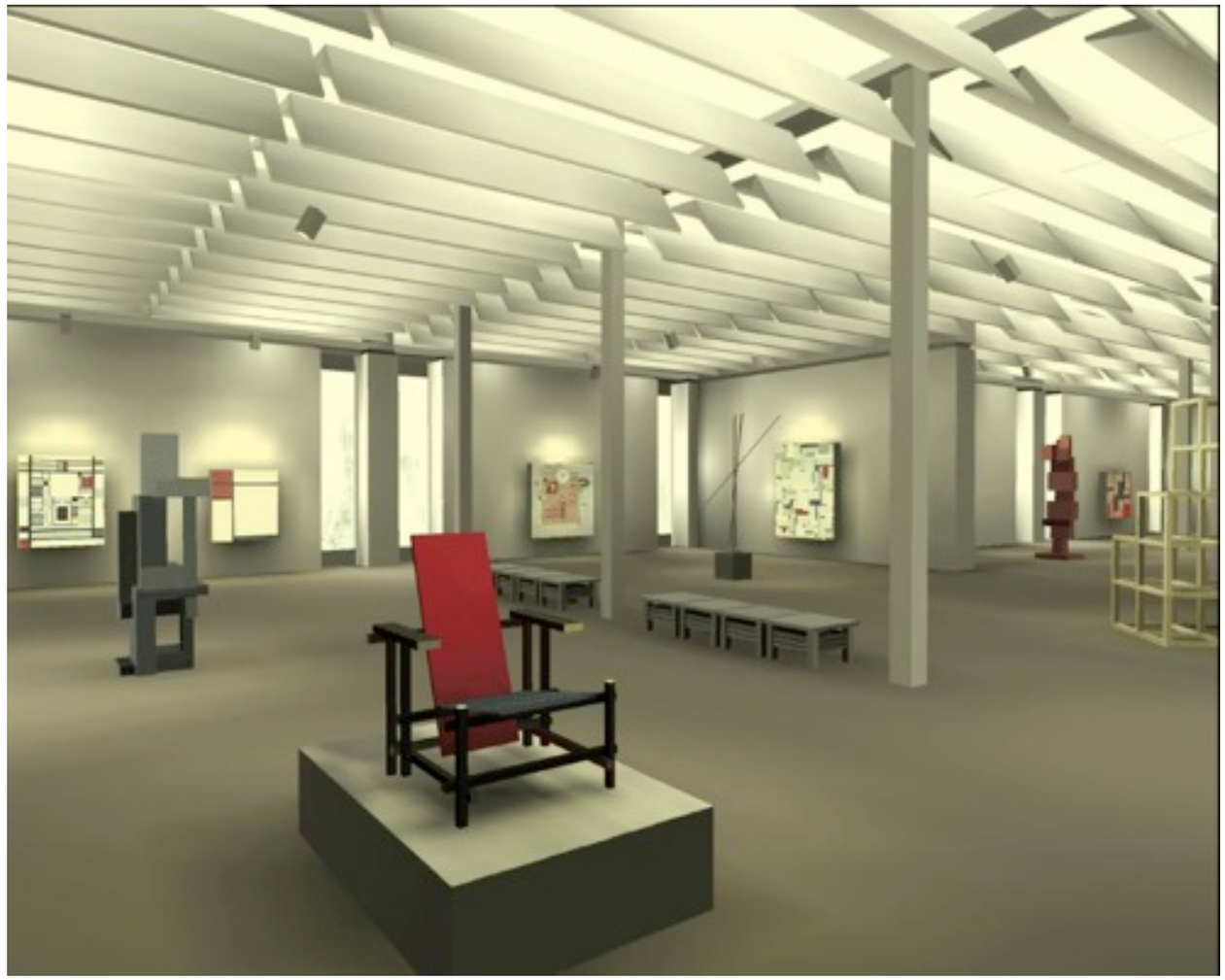
Ray tracing



- Rays are cast from the viewpoint and followed recursively through the scene.
- Whitted ray tracing: Compute direct illumination from light sources at every point hit by traced rays

Mimigu (Wikipedia)

Radiosity



- Discretize scene into patches. Compute strength of interaction between patches.
- Shoot light from source patches, deposit in other patches. Iterate until light is absorbed.

Photon mapping

- Stage 1: Trace photons from light sources and deposit onto *photon map* when photons interact with diffuse surfaces
- Stage 2: Cast rays from viewpoint and estimate radiance

OpenGL

Why a graphics API?

- Save developers from having to implement standard functionality
- Facilitate portability through abstraction

Design considerations for OpenGL

- Enable applications to run on a variety of platforms
- Describe rendering operations succinctly
- Accommodate extensions as graphics hardware evolves
- Maximize performance by closely modeling the capabilities and characteristics of graphics hardware

Agnostic to the window system

- OpenGL was designed to work with a variety of window systems
  - Interaction with the window system is handled by GLUT
- No facilities for obtaining user input
- Does not handle the actual display, merely assembles an image

A rendering API

- The OpenGL API supports the display of geometric primitives
  - Takes a specification of geometric objects
  - Computes illumination
  - Images the scene
- Modeling and animation largely left to the application