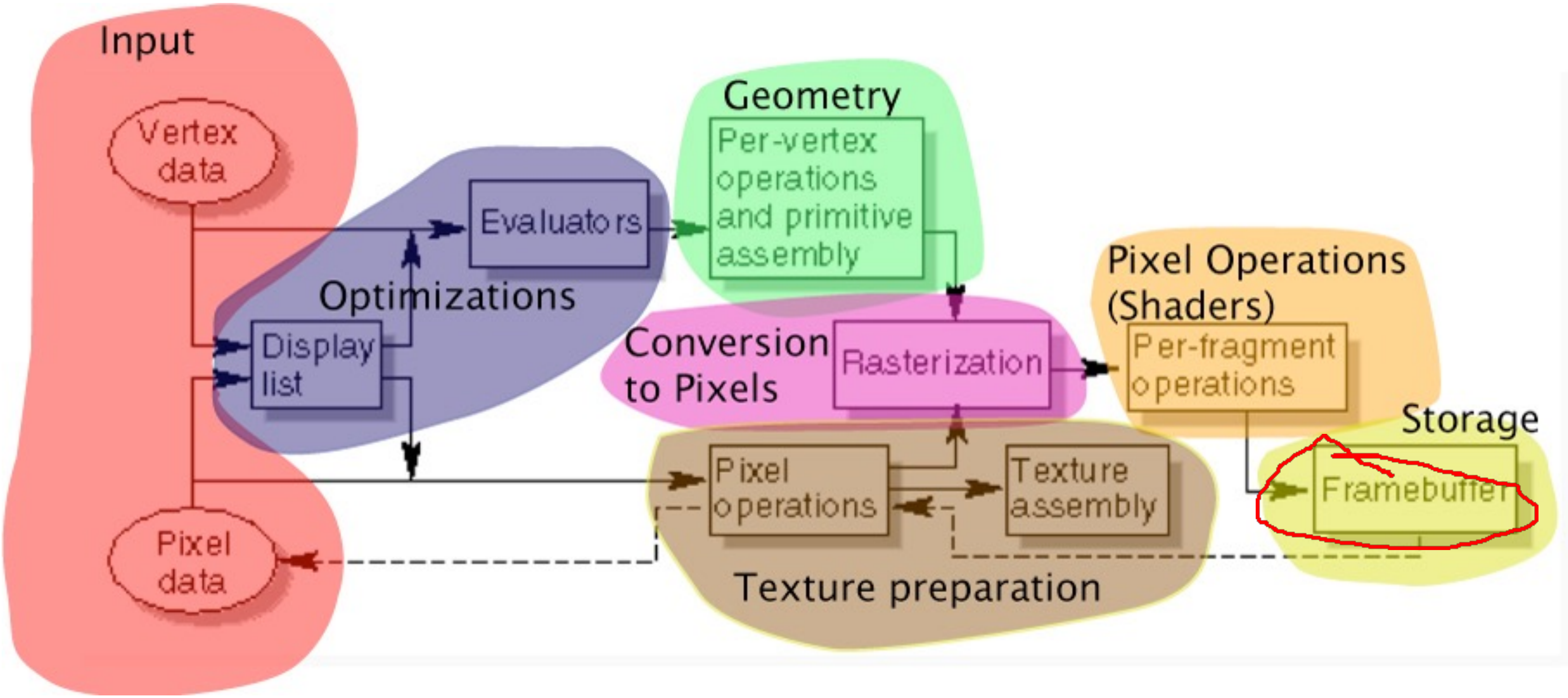
The Frame Buffer: Blending & Compositing

Definition

- Blending combines geometric objects
  - e.g. transparency
- Compositing combines entire images
  - multi-pass textures
  - accumulating results
- Both depend on details of frame-buffer

OpenGL Pipeline



The Frame Buffer

- An image is a rectangular array of data
- OpenGL computes 1 image per frame
  - Stores the image in the frame buffer
  - A special array on the video card
- Frame buffer has several components

Front & Back Buffers

- We have already seen double-buffering
  - drawing one image, displaying another
  - flip between them with glutSwapBuffers()
- The image displayed is in the front buffer
- The image being drawn is in the back buffer
- OpenGL can actually draw into either

Quad Buffering

- It's possible to draw stereo images
  - one image for each eye
  - so we have left and right buffers
- Can be combined with front / back:
  - for details, see the Red Book

Frame Buffer Components

- Frame buffer has:
  - colour buffer for RGBA
  - depth buffer for z-depth
  - stencil buffer
  - accumulation buffer

Stencil Buffer

- Used for masking:
  - covering up parts of frame buffer
  - like using masking tape for painting
- Each “pixel” is on or off
  - marks whether to draw there or not

Accumulation Buffer

- A spare copy of the frame
  - often with higher precision
  - used to composite images
    - multi-pass images
    - generally higher quality

Pixel Buffers

- Extra buffer(s) you can draw into
  - but never display on screen
- Typically used to create textures
- Sometimes specialized for this purpose
- We won't worry about them at this level

Setting the Buffer

- glDrawBuffer(): which to draw into
  - defaults to GL_FRONT_LEFT
  - you won't need this
- glReadBuffer(): which to read from
  - useful for screen captures
  - also used to create textures

Clearing Buffers

- glClear(): clears the buffers specified
  - very slow, so only do once / frame
  - uses colour specified by glClearColor()
  - sets every pixel to that colour
  - often has specialized hardware

Masking Buffers

- By default, all buffers can be changed
- We can turn this on and off:
  - glDepthMask(GL_TRUE)
  - glDepthMask(GL_FALSE)
- Also have glStencilMask(), glColorMask()

How it works

- Rasterization converts triangle to pixels
- OpenGL calls pixels fragments
  - fragments are processed in parallel
  - colour, lighting, &c. computed, then:
  - several tests performed on fragments

Fragment Operations

- For each fragment (pixel), OpenGL does:
  - scissor test
  - alpha test
  - stencil test
  - depth test
  - blending, dithering and logical operations

Scissor Test

- Scissoring is for rectangular regions
  - defined with glScissor(x,y,width,height)
  - fragments inside are kept
  - fragments outside are discarded

Alpha Test

- Compares alpha to a fixed target number
- Discards fragment if comparison fails
- Comparisons possible: $<, \leq, =, \geq, >, \neq$
- Set comparison with glAlphaFunc()

Stencil Test

- A stencil is a shape you paint through
- Fragment compared to stencil: glStencilFunc()
- glStencilOp() changes stencil (if desired)
- Set the stencil with glStencilOp(GL_KEEP)

Depth Test

- We’ve already used this
  - Fragment’s depth compared to buffer
    - failure means fragment is discarded
    - success means depth in buffer is reset
- Comparison set with glDepthFunc()
  - defaults to GL_LESS (keep closer value)

Depth Quantization

- Depth buffer has only a few bits - e.g. 16
- Fragments can only be at 216 distances
- Quantizes distance from near to far clipping planes
- Objects too close to each other render incorrectly
  - Usually see a mixture of pixels from each

Blending Operations

- Blending mixes old & new colours
  - usually based on alpha value
  - alpha usually means opacity
  - specifies how to mix colours
    - more in a minute on this

Dithering

- Older cards don't have many colours
- Approximate colours by dithering
  - mixing darker and lighter pixels
  - hardware does it
  - OpenGL only lets you turn it on / off

Logical Operations

- Again, mostly for older machines
  - bitwise boolean operations
  - restricted form of blending

Accumulation Buffer

- Combines multiple versions of a frame
  - Draw image in back buffer first
  - Call glAccum(GL_ACCUM,x) to add (x * image) to accumulation buffer
  - Repeat for each version of the frame
  - Copy back with glAccum(GL_RETURN,x)

Translucency

- We can see through transparent objects
- More accurately, translucent objects
  - allow some light through from behind
  - specify opacity with alpha component
  - takes alpha * new + (1-alpha) * old

Alpha Transparency

- Set alpha as part of material properties
- Call glEnable(GL_BLEND) to enable
- Use glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
  - takes alpha * source (new value)
  - adds (1-alpha) * destination (old value)

Rendering Order

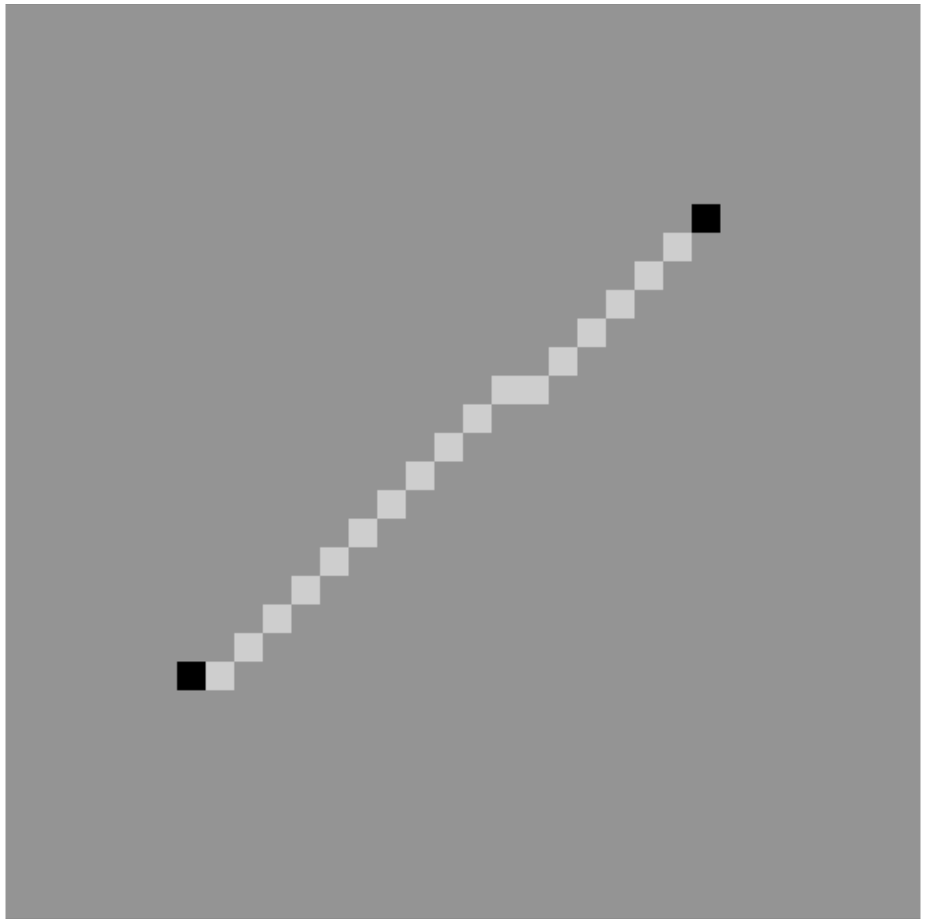
- Alpha blending has to come last
  - draw solid objects first
  - then translucent objects
- Back to painter's algorithm & sort order
- Use carefully

OpenGL Effects

- Antialiasing (spatial and temporal)
- Depth of focus, fog
- Polygon Offset
- Textual Information
- Selection & Picking
- Shadows

Aliasing

- Aka the jaggies
- Not enough pixels
- Eye isn't fooled
- Worse if moving



Use More Pixels

Why this Happens

- Retinal cells see small patches
  - integrate incoming light on that patch
- Pixels are not integrated
  - sampled at one point
- Solution: average several samples

Another Explanation

- Pixel-sized patches are larger than cells
- And retina is good at edge detection
- So we perceive the edge of the pixel
  - our eyes work against us
- Solution: blur the pixel

Technical Explanation

- Eye is reconstructing image internally
- The edge is a high-frequency object
  - hard to reconstruct
  - needs more samples (pixels)
- All of these boil down to this:
  - we need more samples

Averaging Samples

- Use the accumulation buffer
  - render the image several times
  - jitter the camera (move it slightly)
  - each image is slightly different
  - edge is no longer so abrupt

Red Book Example

Aliased

Anti-Aliased

Multisampling

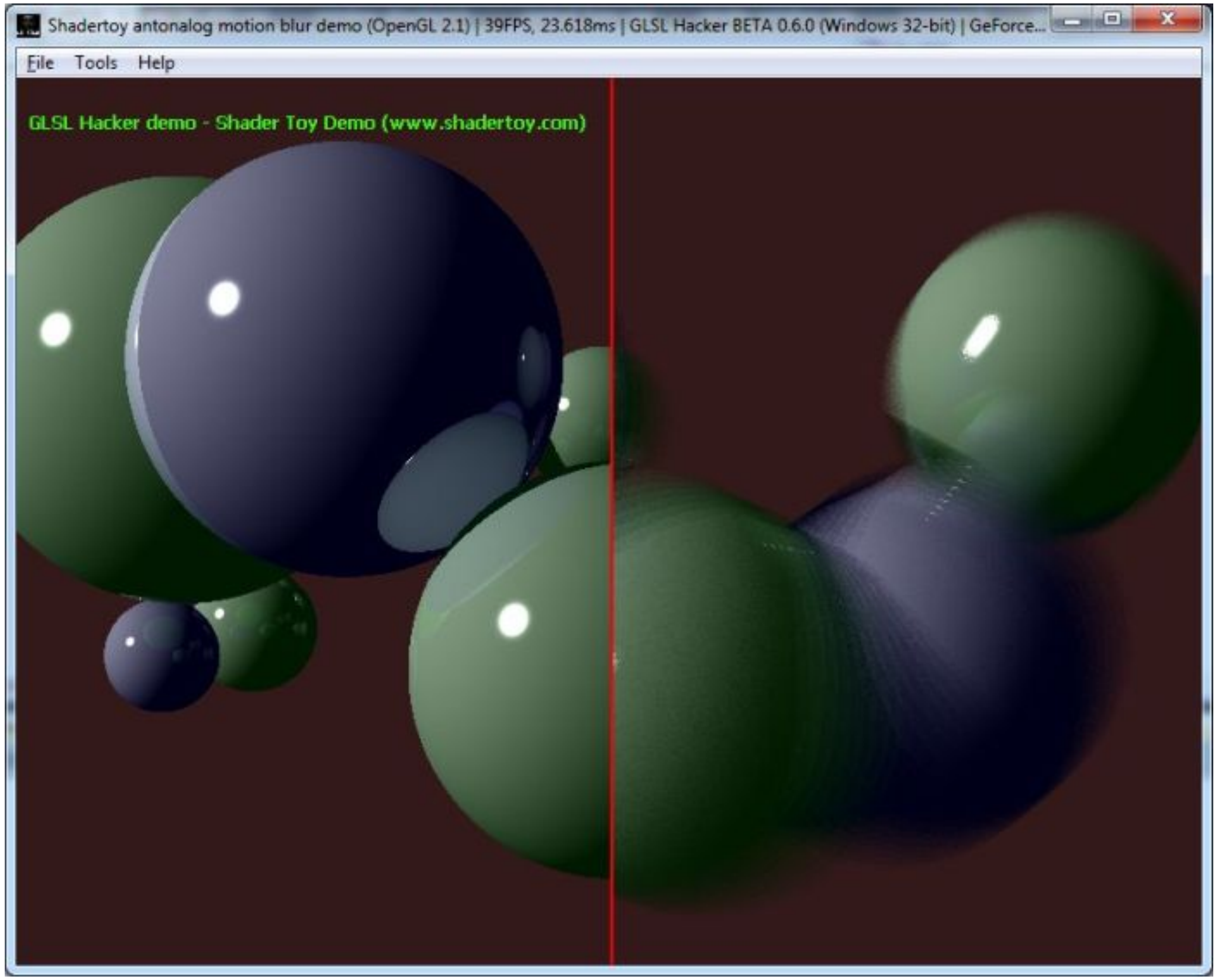
- We could sample more pixels than we show
  - E.g. sample 1600x1200, display 800x600
  - Each pixel shown is average of 2x2 samples
  - This is called multisampling
  - Also called full-screen anti-aliasing (FSAA)
  - How you jitter the pixels is important

Temporal Aliasing

- What happens to a fast moving object?
  - Our eyes integrate light over time
  - We see the sum of its positions
  - If it's too fast, it's blurred
- We do this in the accumulation buffer

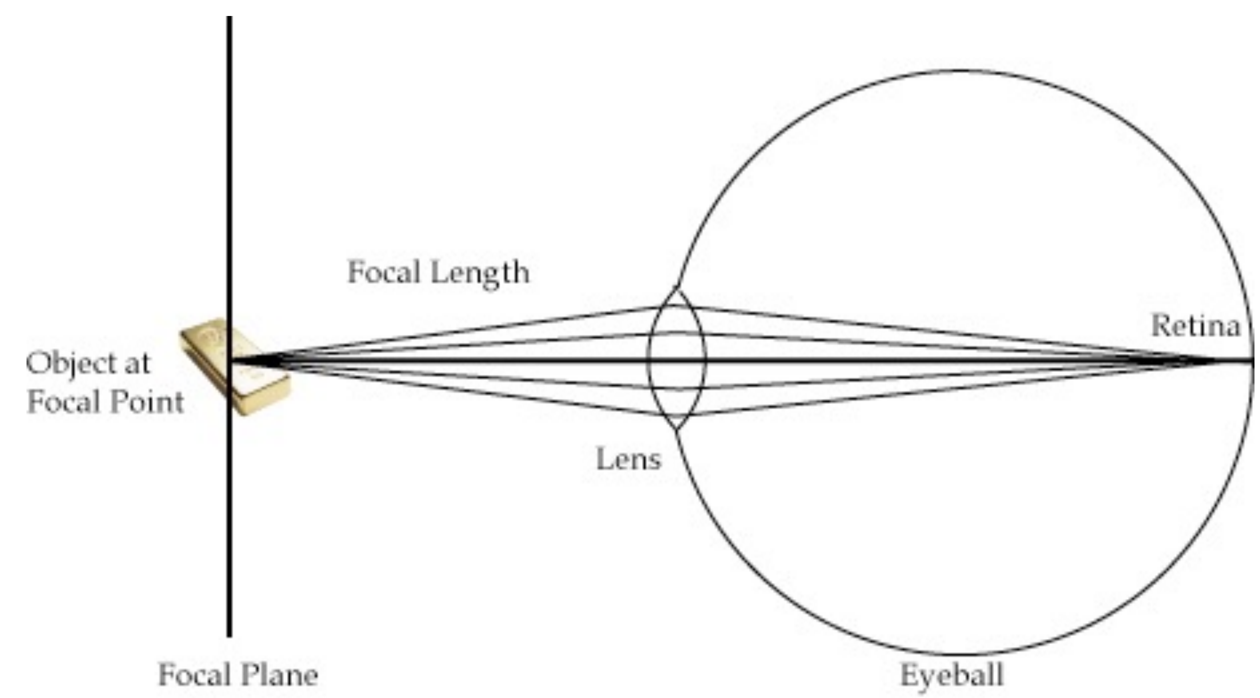
Motion Blurring

- Render several times
- Move object each time
- Accumulate frames
- Image from www.shadetoy.com



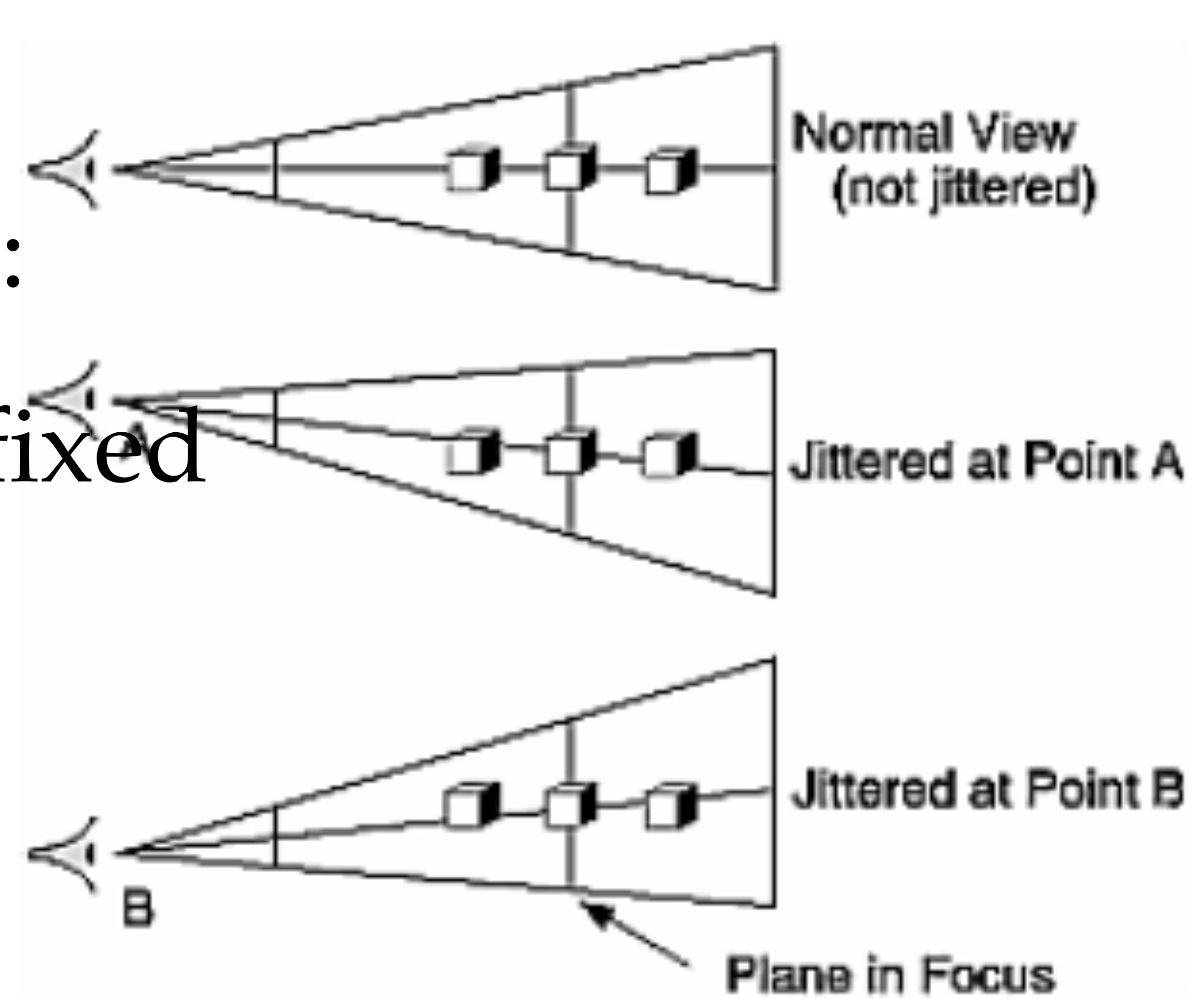
Depth of Focus

- Our eyes focus at a fixed z-distance
  - the focal distance or depth of focus
  - objects at the focal distance are sharp
  - objects at other distances are not
    - they're blurred

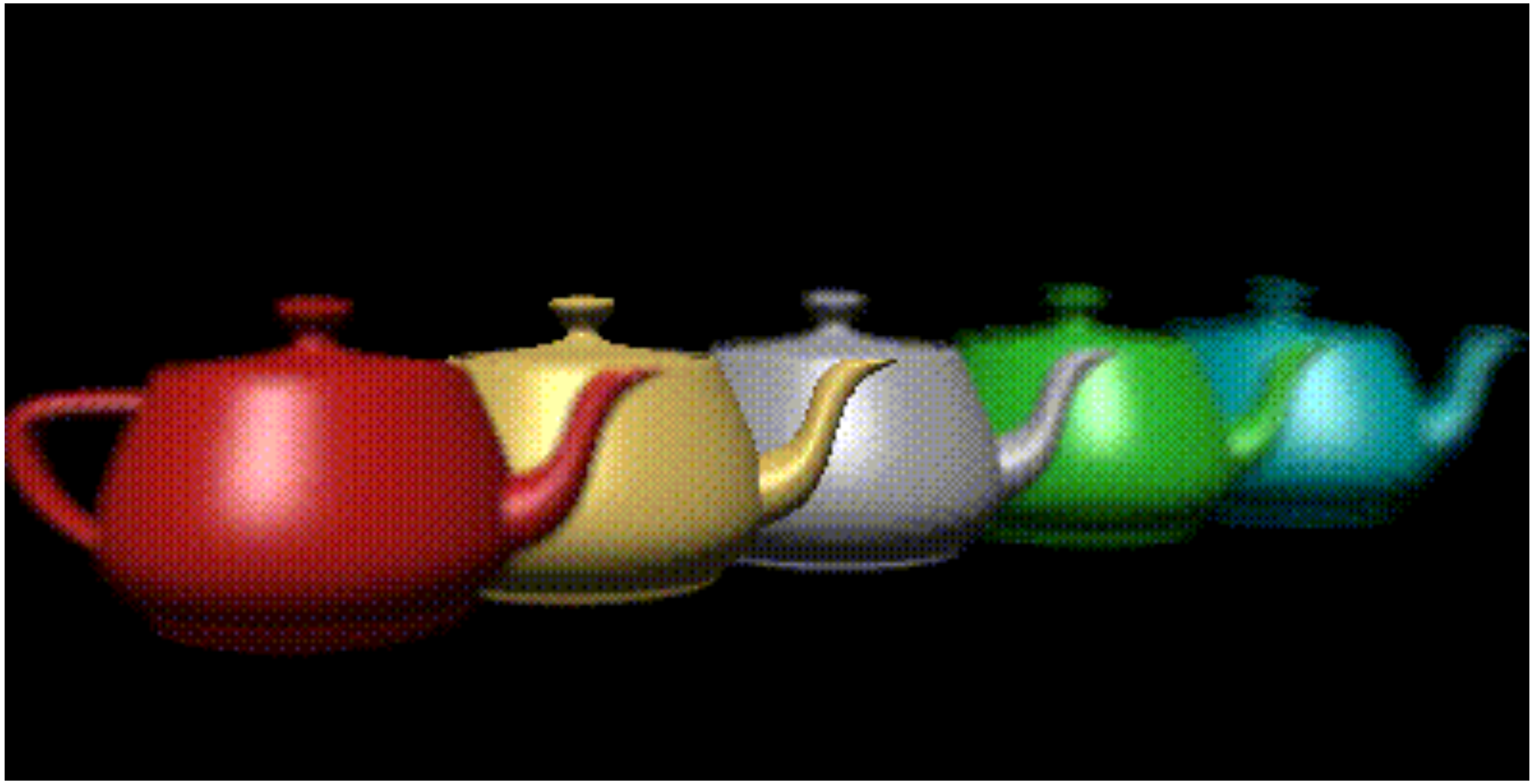


Accumulating Depth of Focus

- Jitter camera slightly
- Keep focal plane fixed:
  - objects in plane are fixed
  - other objects move



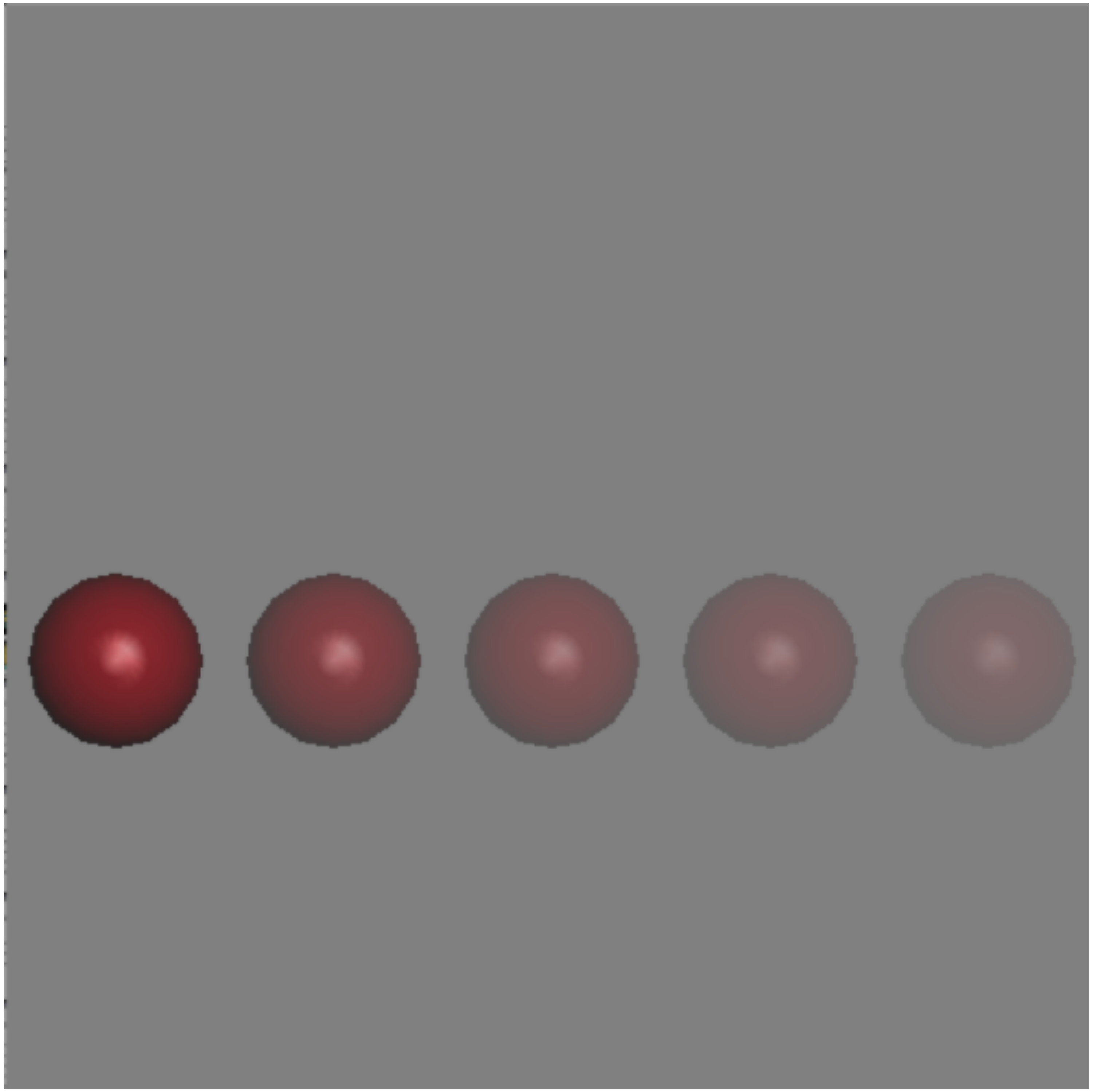
Accumulated Result



Fog / Haze

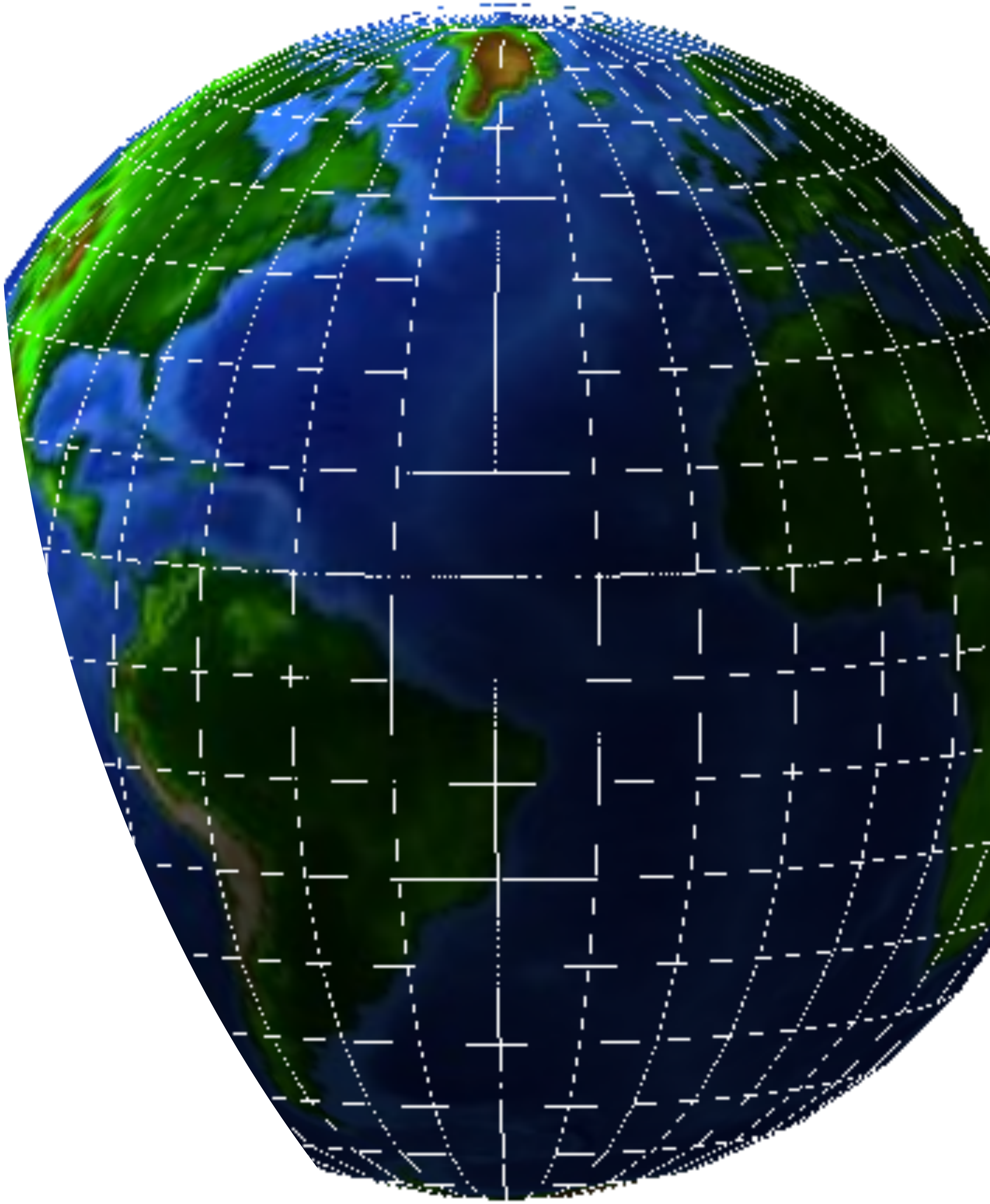
- Light in a ray can get scattered
  - reflected from microscopic particles
  - fog (water), smoke (soot), haze (dust)
- Objects therefore fade with distance
  - uses depth from the depth buffer
  - glEnable(GL_FOG), &c.

Example of Fog



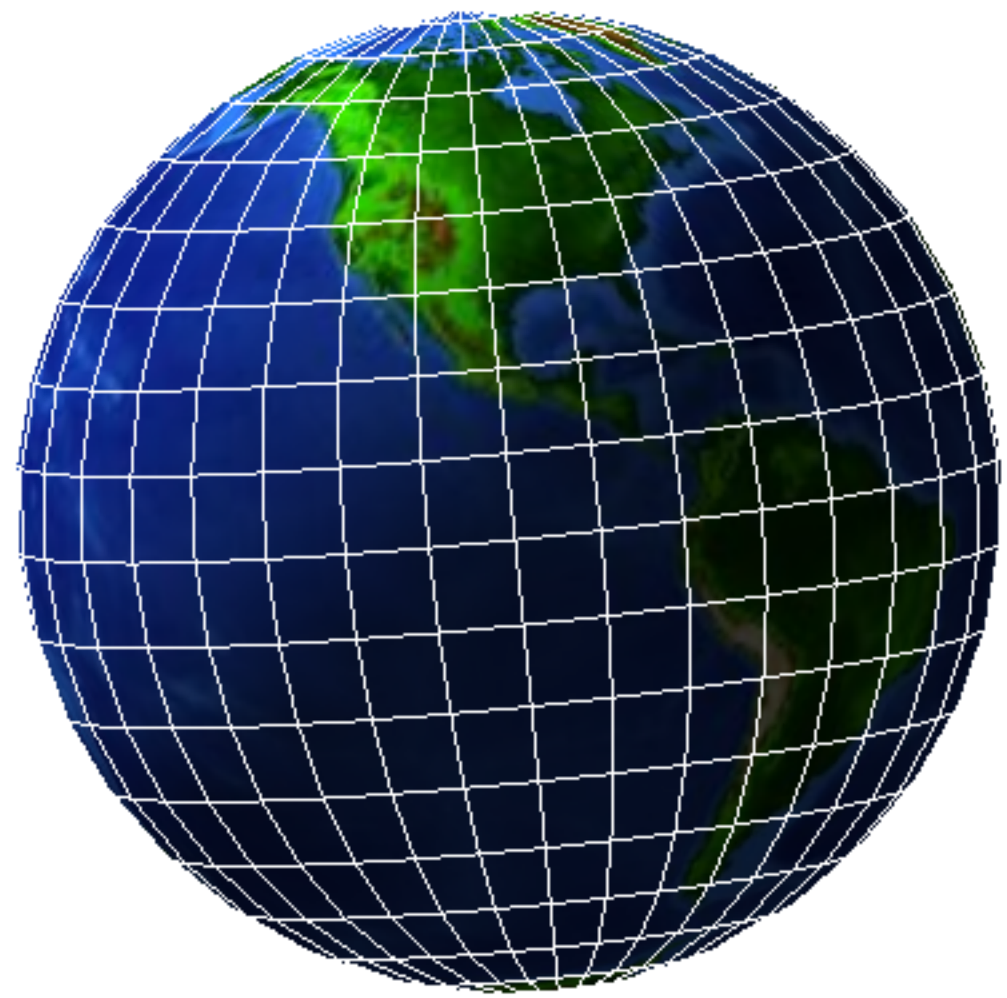
Polygon Offset

- What if we want to draw triangle edges?
- Draw once as a solid, once as wireframe
- But we run into depth buffer quantization



Bigger Wireframe

- Scale the wireframe
- Easy for sphere
- Doesn’t always work nicely
- Lines don’t quite match triangles



Polygon Offset

- Get OpenGL to take care of it
- glEnable(GL_POLYGON_OFFSET);
- glPolygonOffset() moves the polygon
    - slightly towards or away from eye
    - gets rid of the problem
- useful for decals & hidden lines

Frame Buffers allow for different rendering pipelines