Computer Graphics

by Ruen-Rone Lee

ICL/ITRI

Wrap up from last course

- **Texture mapping**
  - **Texture addressing**
  - **Texture filtering**
  - **Multi-texturing**
  - **Bump mapping**

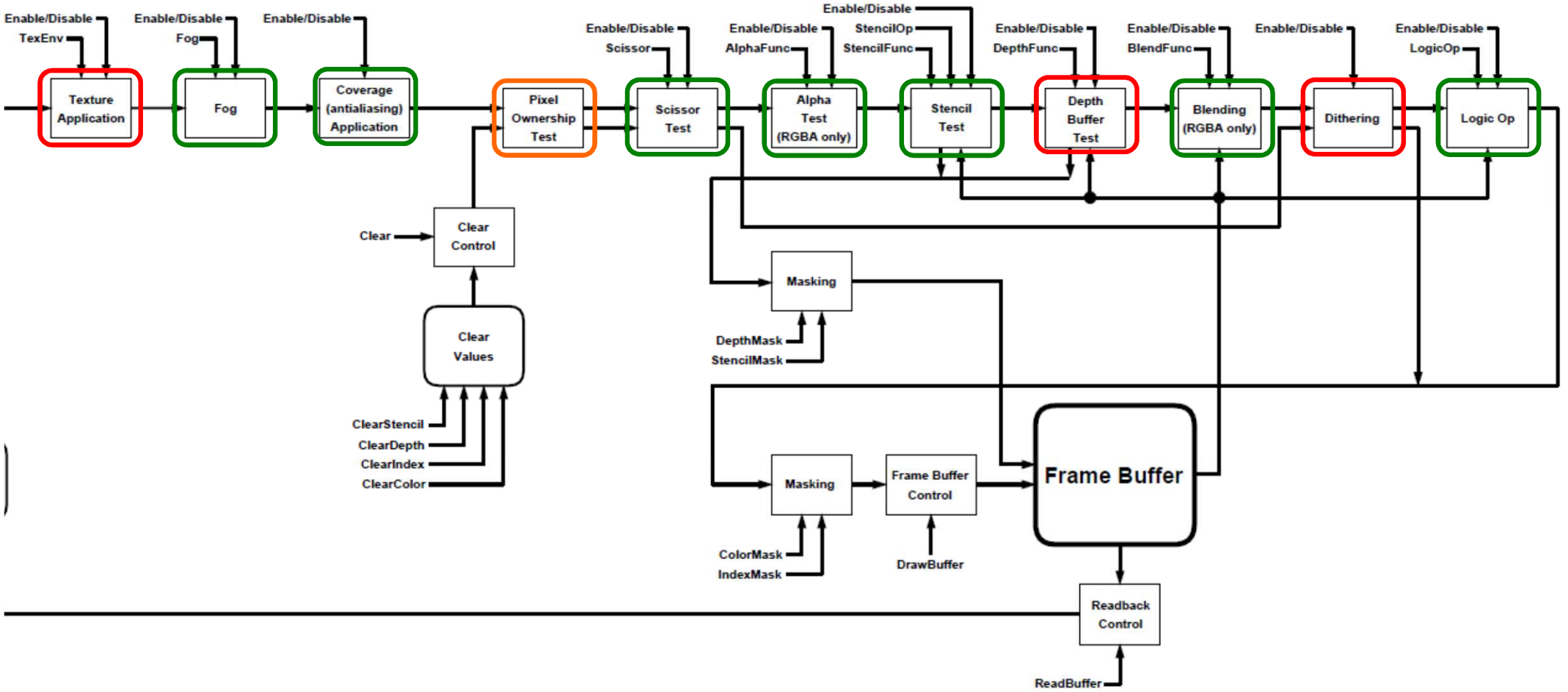
Fragment Operations

Fragment Tests

Blending

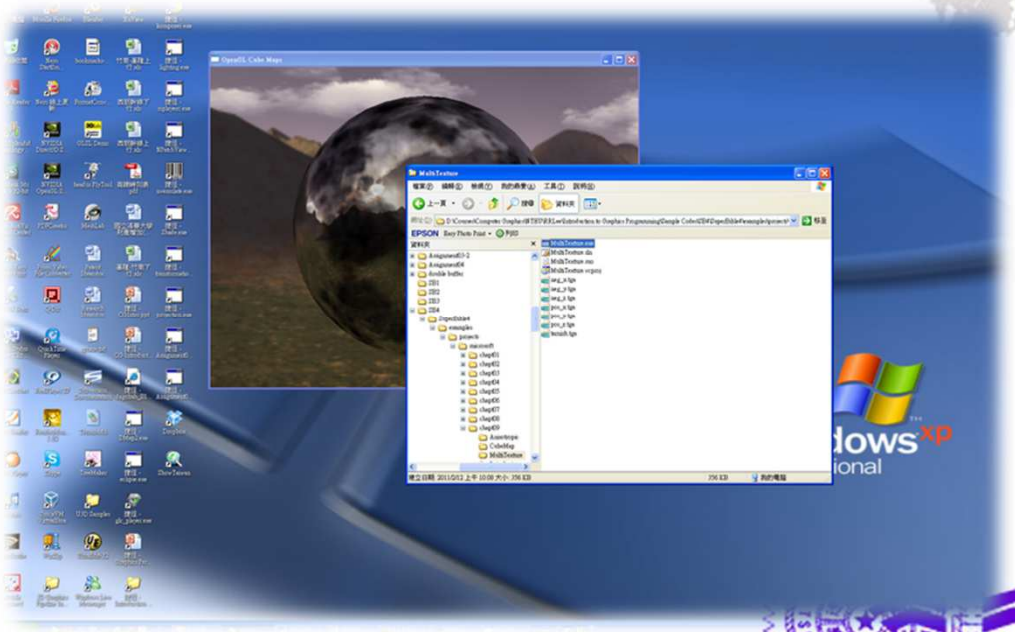
Fragment Operations

- OpenGL Pipeline (Fragment Processing)



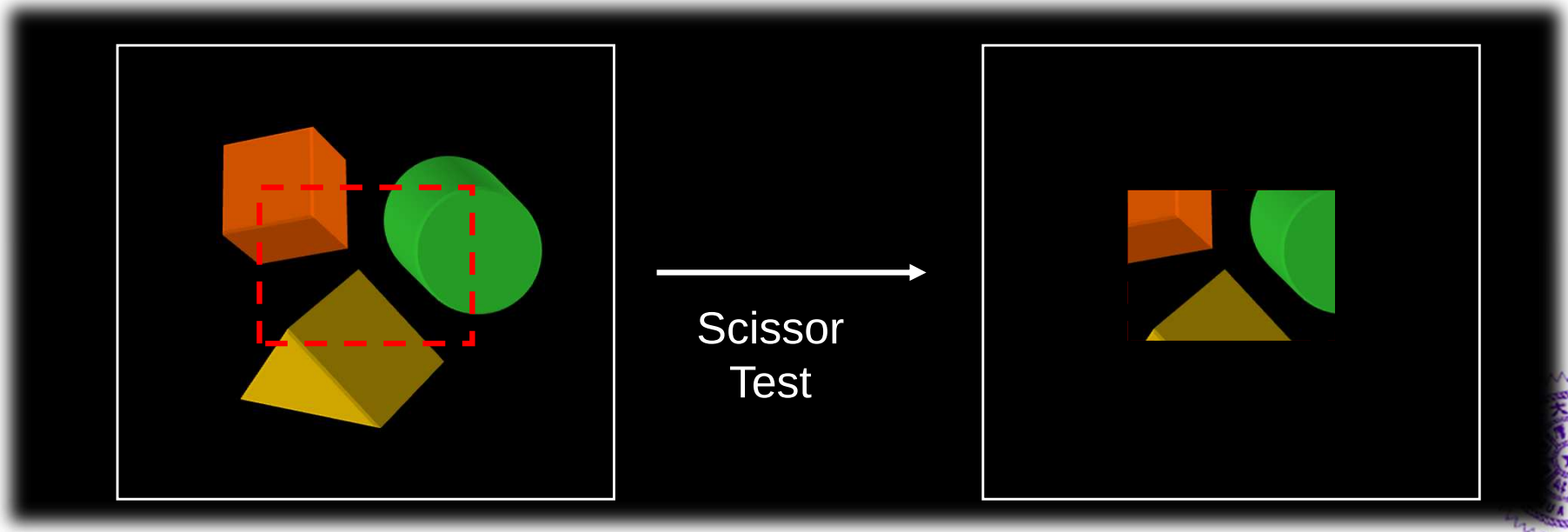
Pixel Ownership Test

- determine if the pixel at location (x_w, y_w) in the framebuffer is currently owned by the GL context
- This test allows the window system to control the GL's behavior, for instance, when a GL window is obscured



Scissor Test

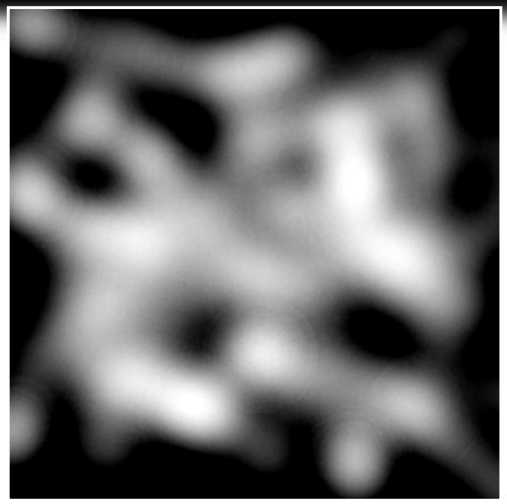
- Similar to clipping but performs on screen space during rasterization
- Given a rectangular region in screen space. Accept only to those pixels within the defined region



Alpha Test

- The fragment alpha is tested against a reference alpha to determine whether the fragment is accepted or not
- Alpha test functions
  - Never, Always, <, <=, =, >= >, !=

Alpha mask →



Application of Alpha Test

- **Remove transparent pixels in texture mapping**

Stencil Test

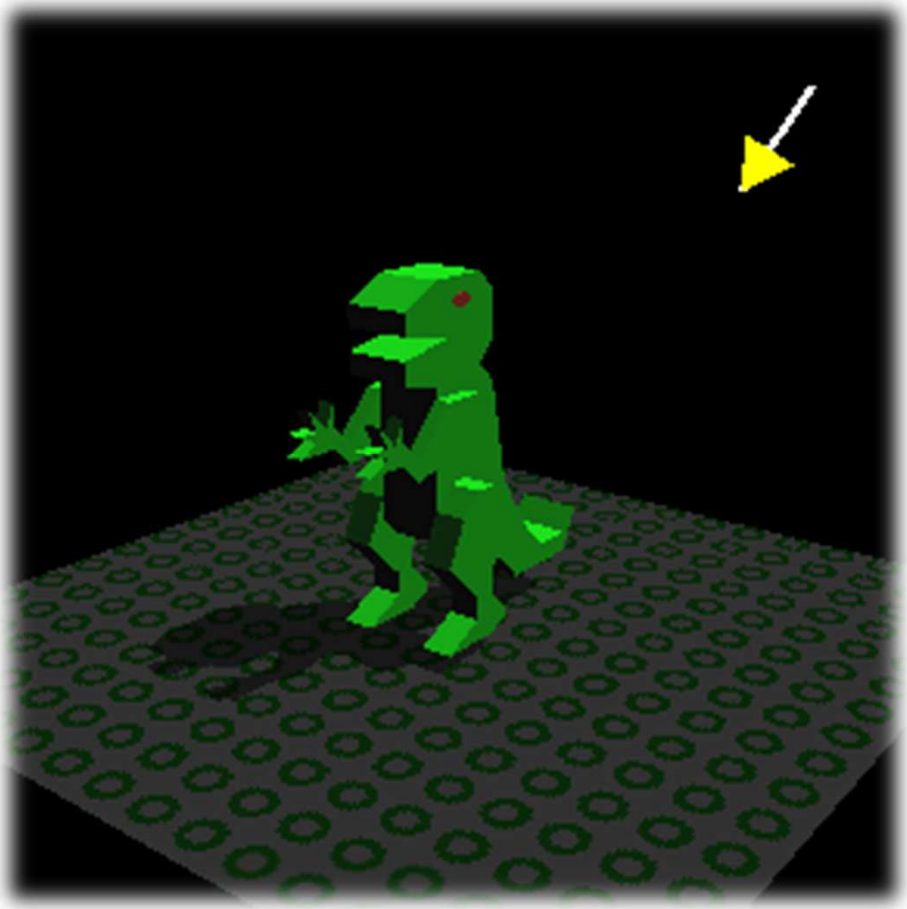
- A stencil buffer is required and is commonly stored together with depth buffer. Eg. A 32-bit Stencil-Z buffer value contains an 8-bit stencil value and a 24-bit depth value
- Stencil buffer is a buffer used to mask pixels in an image
- Stencil test with a reference value
  - Reference stencil value is compared to the corresponding value in stencil buffer
- Stencil buffer update by a specific stencil operation, such as increase or decrease

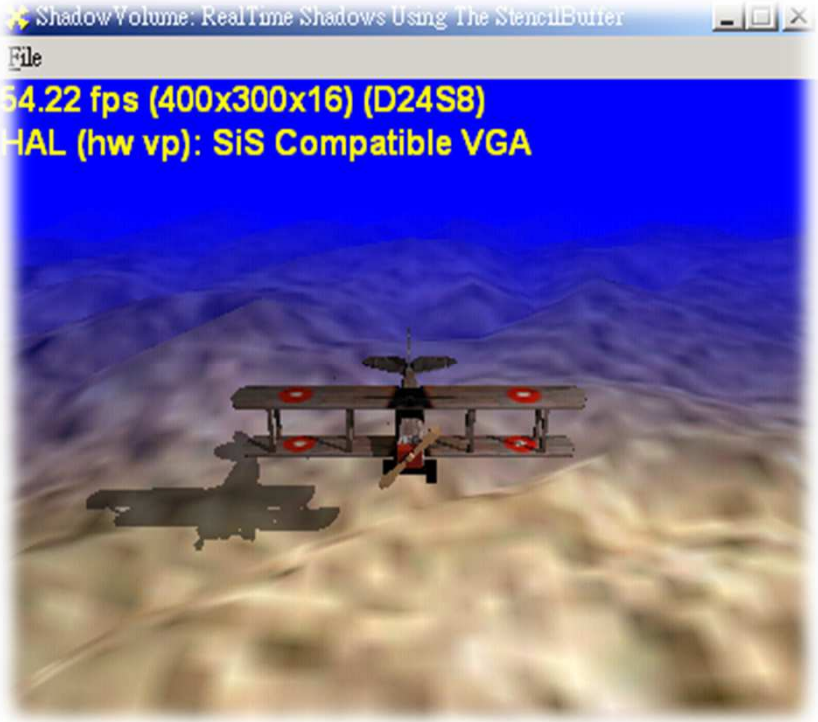
Application of using Stencil Buffer

- **Reflection**

Application of using Stencil Buffer

- **Shadow Volume**



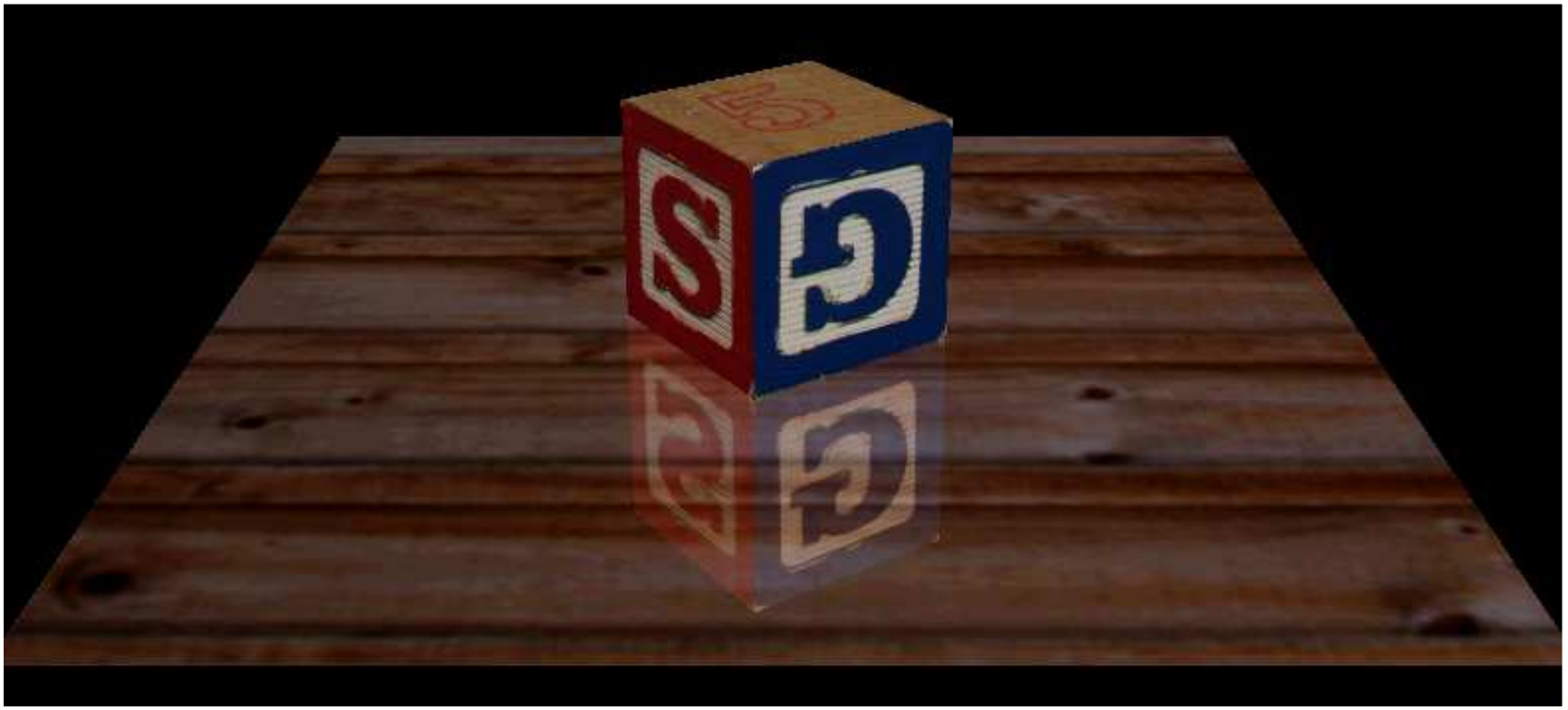


Depth Buffer Test

- Already introduced in previous topic
- For performance considering, depth buffer test can be performed as early as possible
- Early Z-test cannot be performed if the depth value is modified during the pipeline. Eg. shader code might change the output depth value
- If stencil test with stencil buffer is enabled, even the Z-test is fail, the fragment stencil buffer still need to modified based on the stencil buffer operation defined for Zfail

Blending

- Combine the incoming fragment color with the pixel color in frame buffer



General Blending Equation

- $Color = Src \times S_f + Dst \times D_f$

  Src: *Source color*

  Dst: *Destination color*

  S_f: *Source blending factor*

  D_f: *Destination blending factor*

  $Color = (R_s S_r + R_d D_r , G_s S_g + G_d D_g , B_s S_b + B_d D_b , A_s S_a + A_d D_a)$

Src and Dst Blending Factors

Constant	RGB Blend Factor	Alpha Blend Factor
GL_ZERO	(0, 0, 0)	0
GL_ONE	(1, 1, 1)	1
GL_SRC_COLOR	(R_s, G_s, B_s)	A_s
GL_ONE_MINUS_SRC_COLOR	$(1, 1, 1)-(R_s, G_s, B_s)$	$1 - A_s$
GL_DST_COLOR	(R_d, G_d, B_d)	A_d
GL_ONE_MINUS_DST_COLOR	$(1, 1, 1)-(R_d, G_d, B_d)$	$1 - A_d$
GL_SRC_ALPHA	(A_s, A_s, A_s)	A_s
GL_ONE_MINUS_SRC_ALPHA	$(1, 1, 1)-(A_s, A_s, A_s)$	$1 - A_s$
GL_DST_ALPHA	(A_d, A_d, A_d)	A_d
GL_ONE_MINUS_DST_ALPHA	$(1, 1, 1)-(A_d, A_d, A_d)$	$1 - A_d$
GL_CONSTANT_COLOR	(R_c, G_c, B_c)	A_c
GL_ONE_MINUS_CONSTANT_COLOR	$(1, 1, 1)-(R_c, G_c, B_c)$	$1 - A_c$
GL_CONSTANT_ALPHA	(A_c, A_c, A_c)	A_c
GL_ONE_MINUS_CONSTANT_ALPHA	$(1, 1, 1)-(A_c, A_c, A_c)$	$1 - A_c$
GL_SRC_ALPHA_SATURATE	(f, f, f); $f = \min(A_s, 1-A_d)$	1

Alpha Blending

- **Source blending factor is set to *source alpha* and destination blending factor is set to (1.0 – *source alpha*)**

  $$Color = Src \times S_a + Dst \times (1 - S_a)$$

- **Alpha value is used as the fraction of translucency**
- **Alpha value equal to 1.0 is opaque**
- **Alpha value equal to 0.0 is transparent**

Alpha Blending

- **Example**

Fog Blending

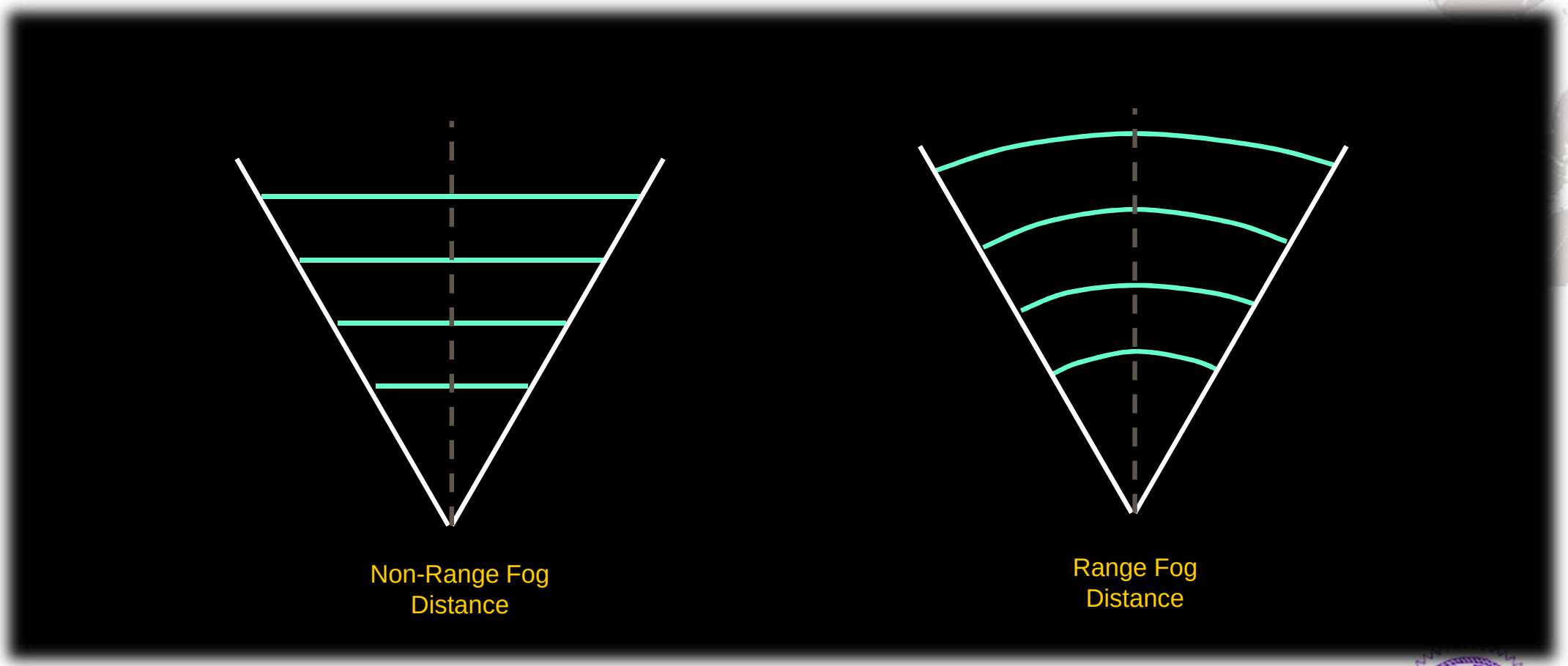
- **Fog Blending Equation**
  - $C = f \times C_i + (1 - f) \times C_f$
  - f: fog factor
  - C_i: incoming color
  - C_f : fog color

Fog Blending

- Vertex Fog
  - Derive fog factor for each vertex
  - Apply fog blending at each vertex
  - Interpolate the vertex color (with fog) during rasterization
- Pixel Fog
  - Per-pixel calculate fog factor using the depth of each pixel
  - Apply fog blending for each pixel

Fog Blending

- **Range Fog**



Fog Factor

- **Linear**

$$f = \frac{end - d}{end - start}$$

$start$: the distance at which fog effects begin

end : the distance at which fog effects no longer increase

d : depth; distance from the viewpoint

Fog Factor

- **Exponential**

$$f = \frac{1}{e^{d \times density}}$$

e : the base of natural logarithms

$density$: an arbitrary fog density between $[0.0, 1.0]$

d : depth; distance from the viewpoint

Fog Factor

- **Exponential-2**

$$f = \frac{1}{e^{(d \times density)^2}}$$

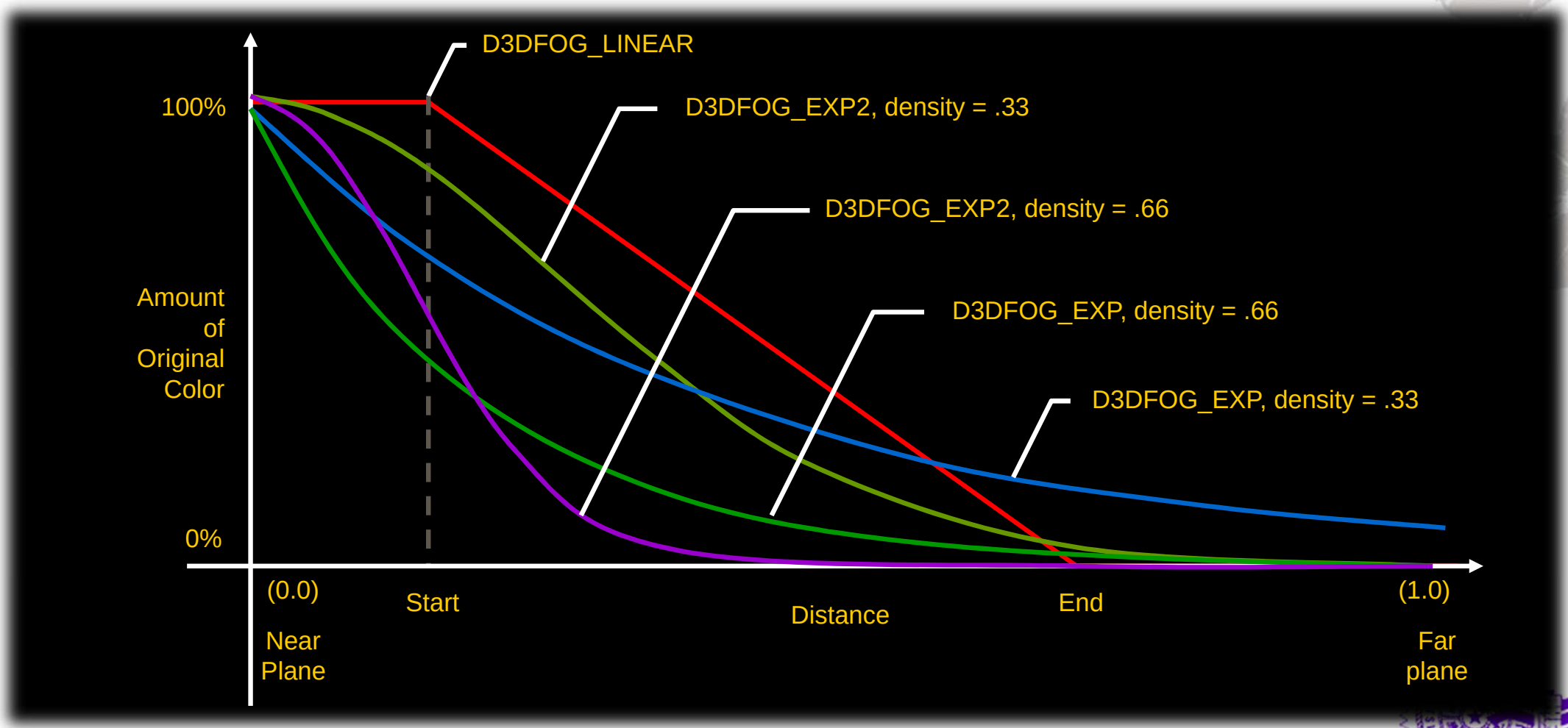
e : the base of natural logarithms

$density$: an arbitrary fog density between $[0.0, 1.0]$

d : depth; distance from the viewpoint

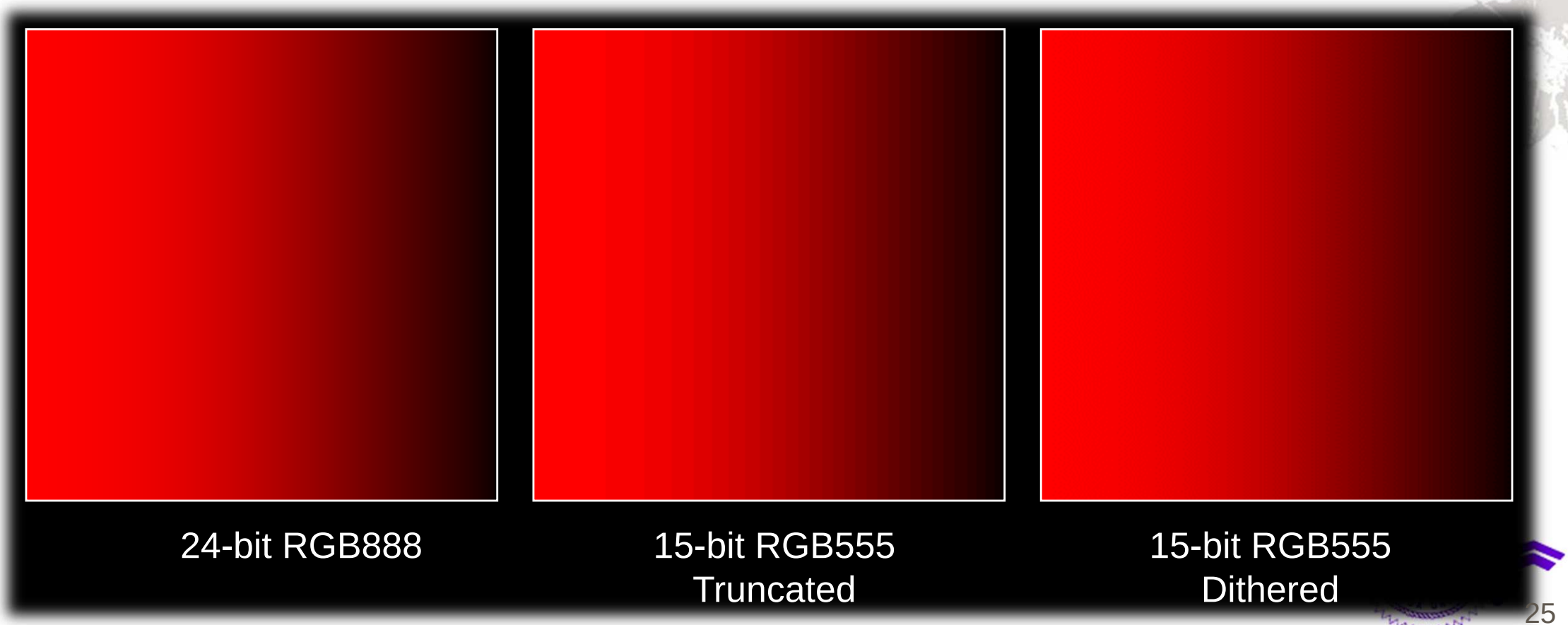
Fog Factor

- Comparison



Dithering

- **Dithering is a method to use less colors to represent the same image**
  - **Eg. 24-bit RGB888 to 15-bit RGB555**



Dithering

- **Implementation**
  - **Dither Pattern**

$$\begin{bmatrix} 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \\ 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \end{bmatrix}$$

8 bits to 4 bits

$$\begin{bmatrix} 1 & 5 & 0 & 4 \\ 7 & 3 & 6 & 2 \\ 0 & 4 & 1 & 5 \\ 6 & 2 & 7 & 3 \end{bmatrix}$$

8 bits to 5 bits

$$\begin{bmatrix} 0 & 2 & 0 & 2 \\ 3 & 1 & 3 & 1 \\ 0 & 2 & 0 & 2 \\ 3 & 1 & 3 & 1 \end{bmatrix}$$

8 bits to 6 bits

Q&A