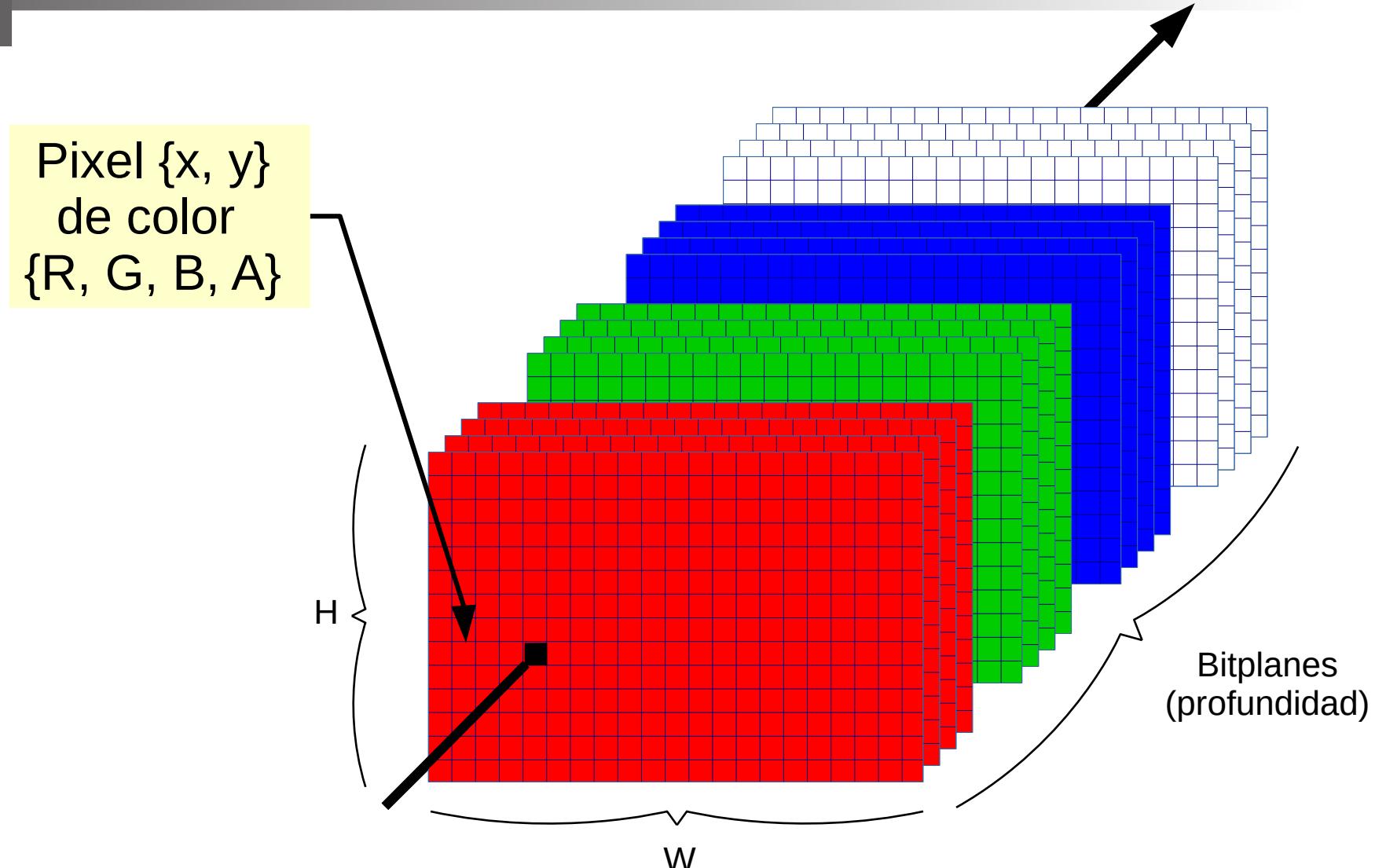


Unidad 7

Técnicas del Espacio de la Imagen

Buffer de Color

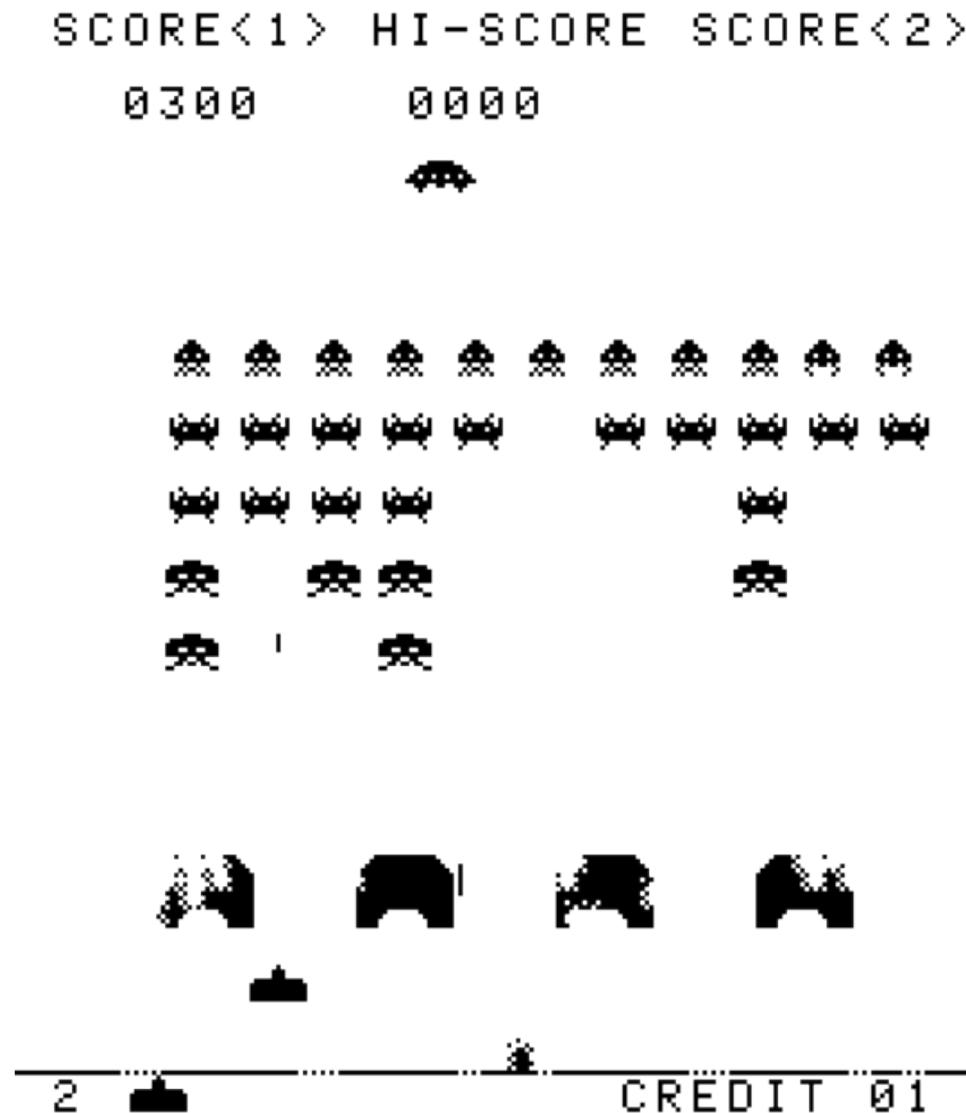


R0, G0, B0, A0, R1, G1, B1, A1, R2, G2, B2, A2, R3, G3, B3, A3...

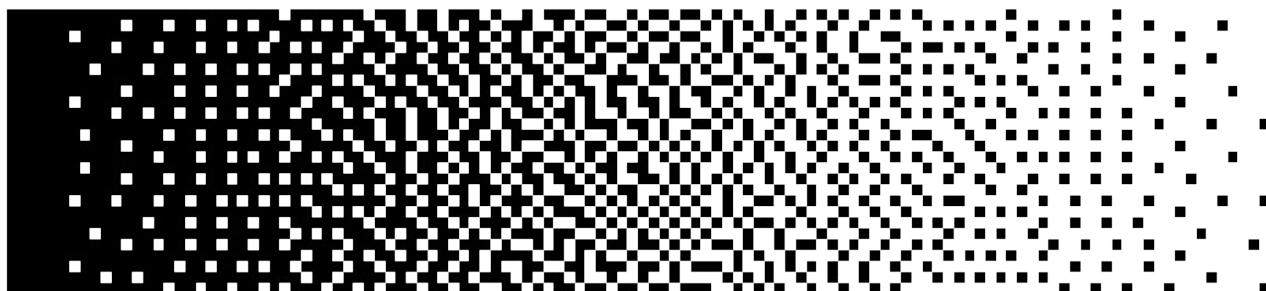
Profundidad de Color: Monocromo (1 bit)

1 bit (y poca resolución)

SPACE
INVADERS

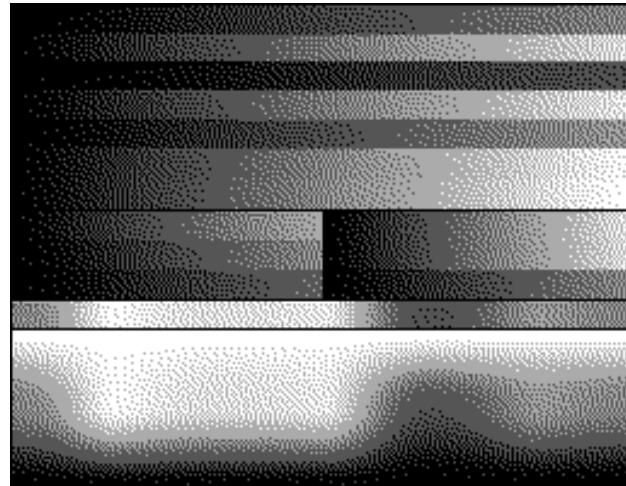
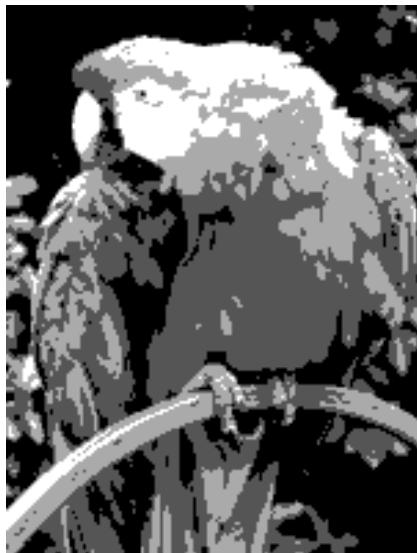


Profundidad de Color: Dithering

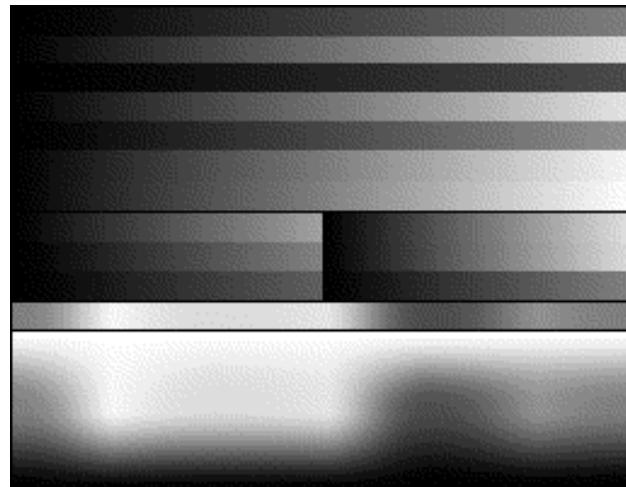


Profundidad de Color (2/4 bits)

2 bit



4 bit



Profundidad de Color: 256 Tonos/Colores (8bits)

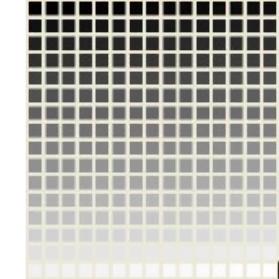
256 Colores



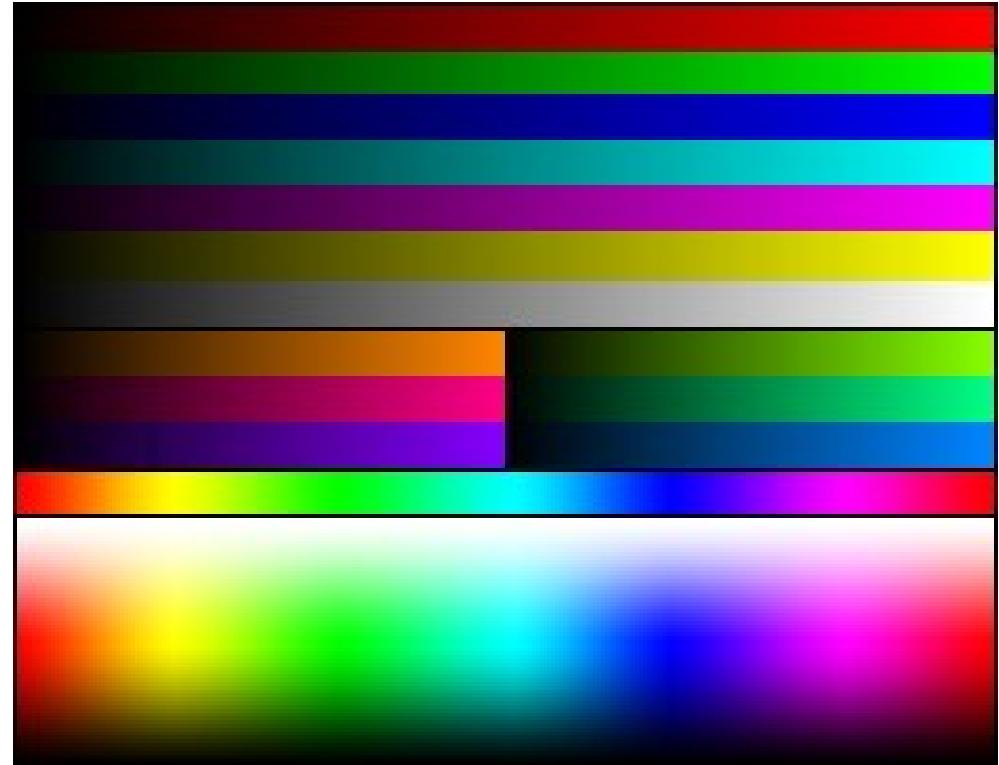
256 Tonos



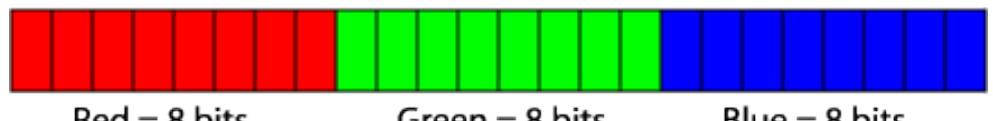
Colores indexados



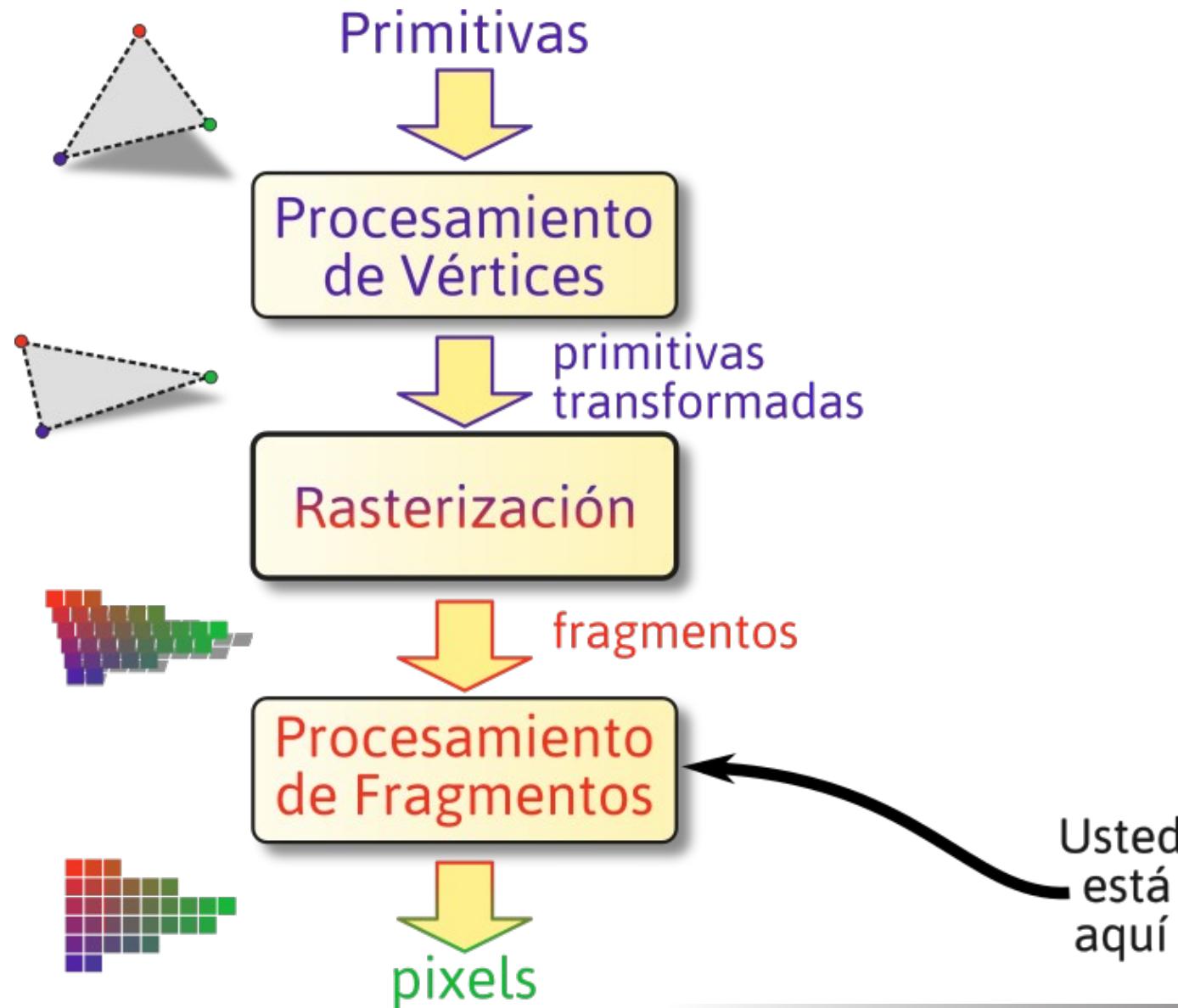
Profundidad de Color: True Color (24bits)



24 bit Color



Usted está aquí



Tests de Fragmentos

Rasterización

Ownership
Scissor
Alpha
Stencil
Depth

Actualización de buffers

`glEnable(GL_xxxxxx_TEST)`

`glDisable(GL_xxxxxx_TEST)`

`glXxxxxFunc(comp, [ref], [mask])`

***comp*:** posibles comparadores

- GL_NEVER
- GL_ALWAYS
- GL_LESS
- GL_EQUAL
- GL_GREATER
- GL_NOTEQUAL

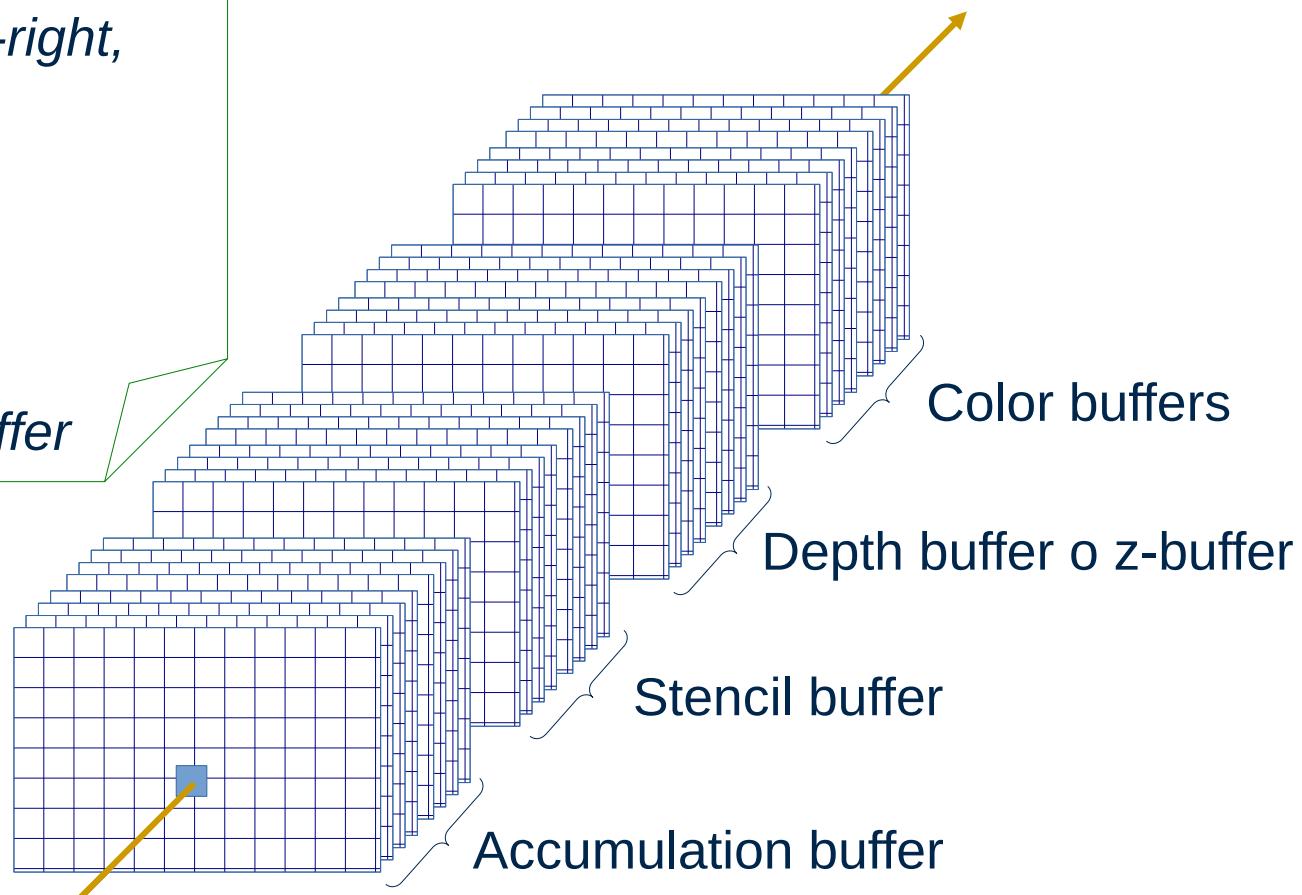
***ref*:** valor con el que se compara (alpha y stencil)

***mask*:** para enmascarar bits en la comparación (solo stencil)

Framebuffer

- *Color buffers:*
 - *front-left, front-right,*
 - *back-left, back-right,*
 - *Auxiliary[0-...]*
- *Depth buffer*
- *Stencil buffer*
- *Accumulation buffer*

```
glutInitDisplayMode(buffers "or"eados)  
glGetIntegerv(GL_DEPTH_BITS,&valor)
```



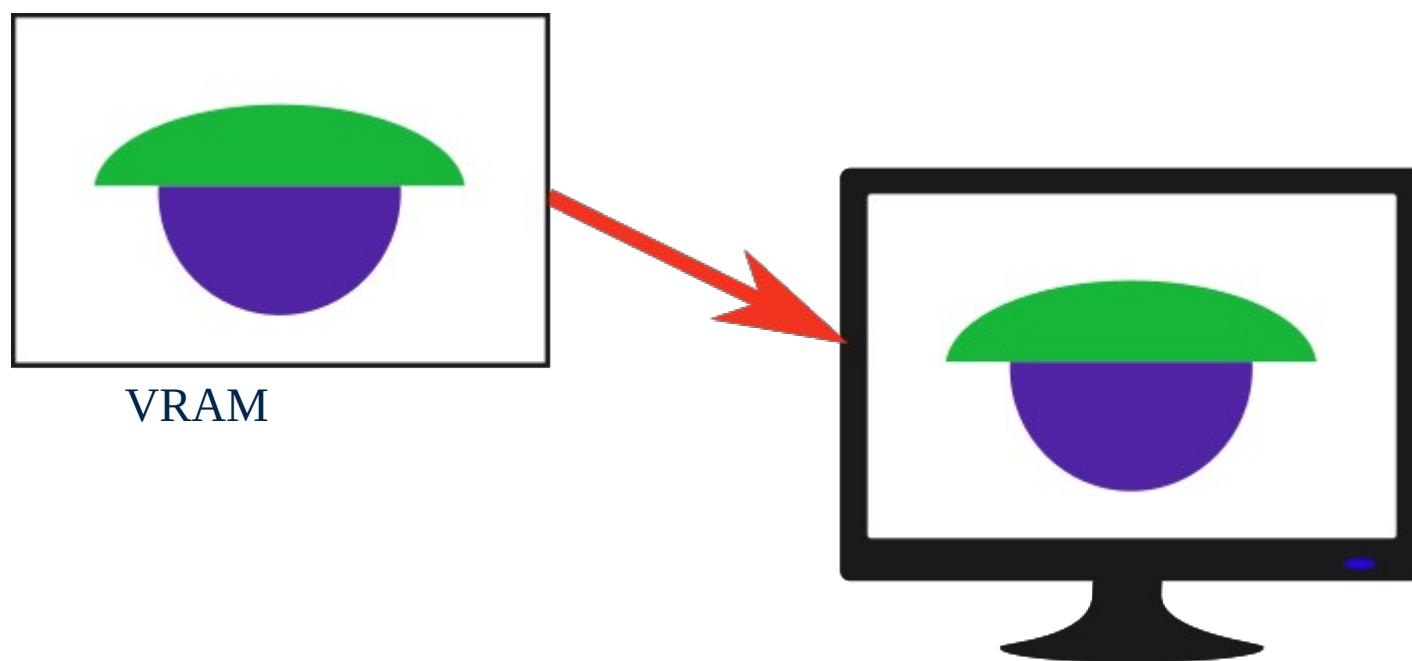
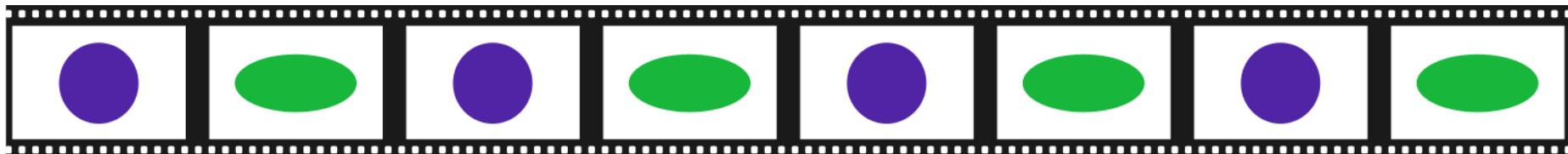
Framebuffer objects

en OpenGL 3.0

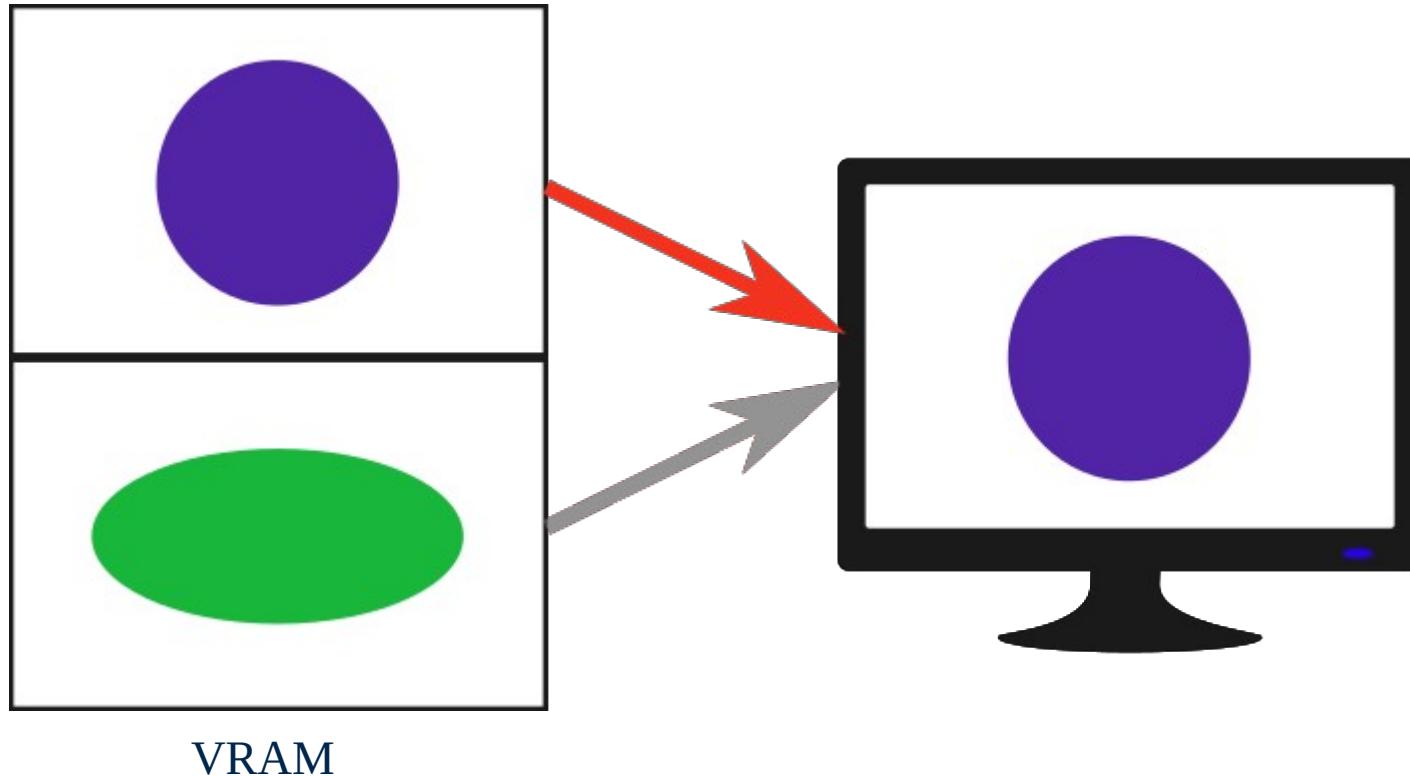
Render Targets Model

en Direct3D 11

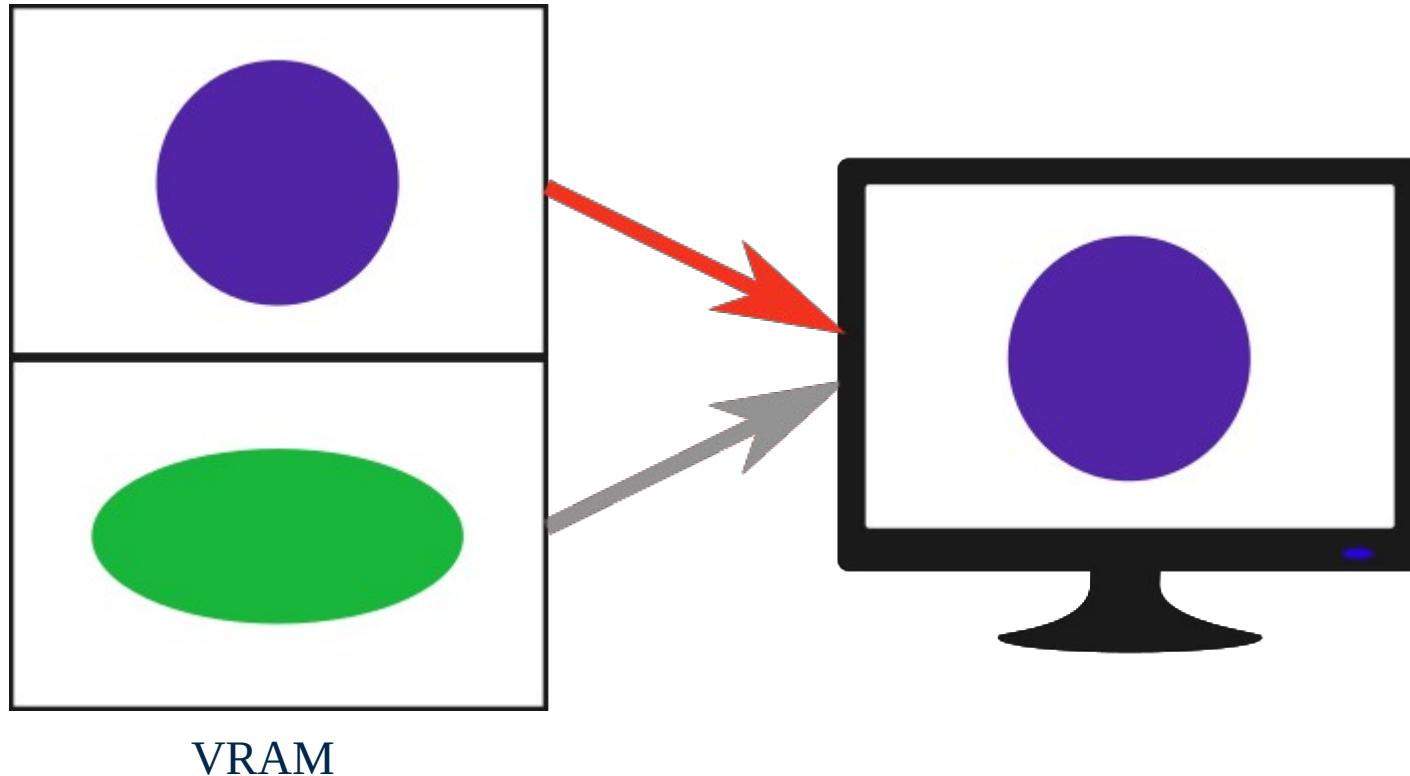
Simple Buffer - Tearing



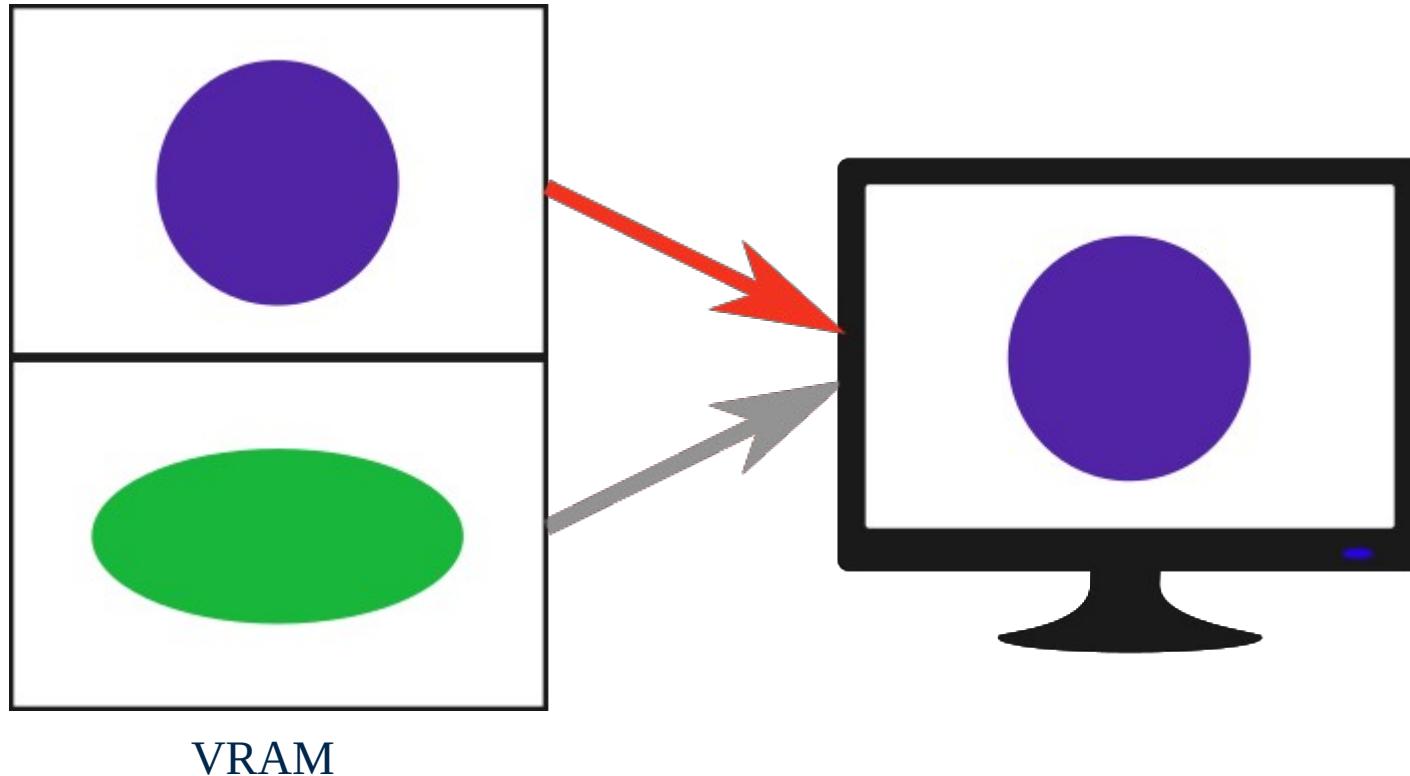
Double Buffer



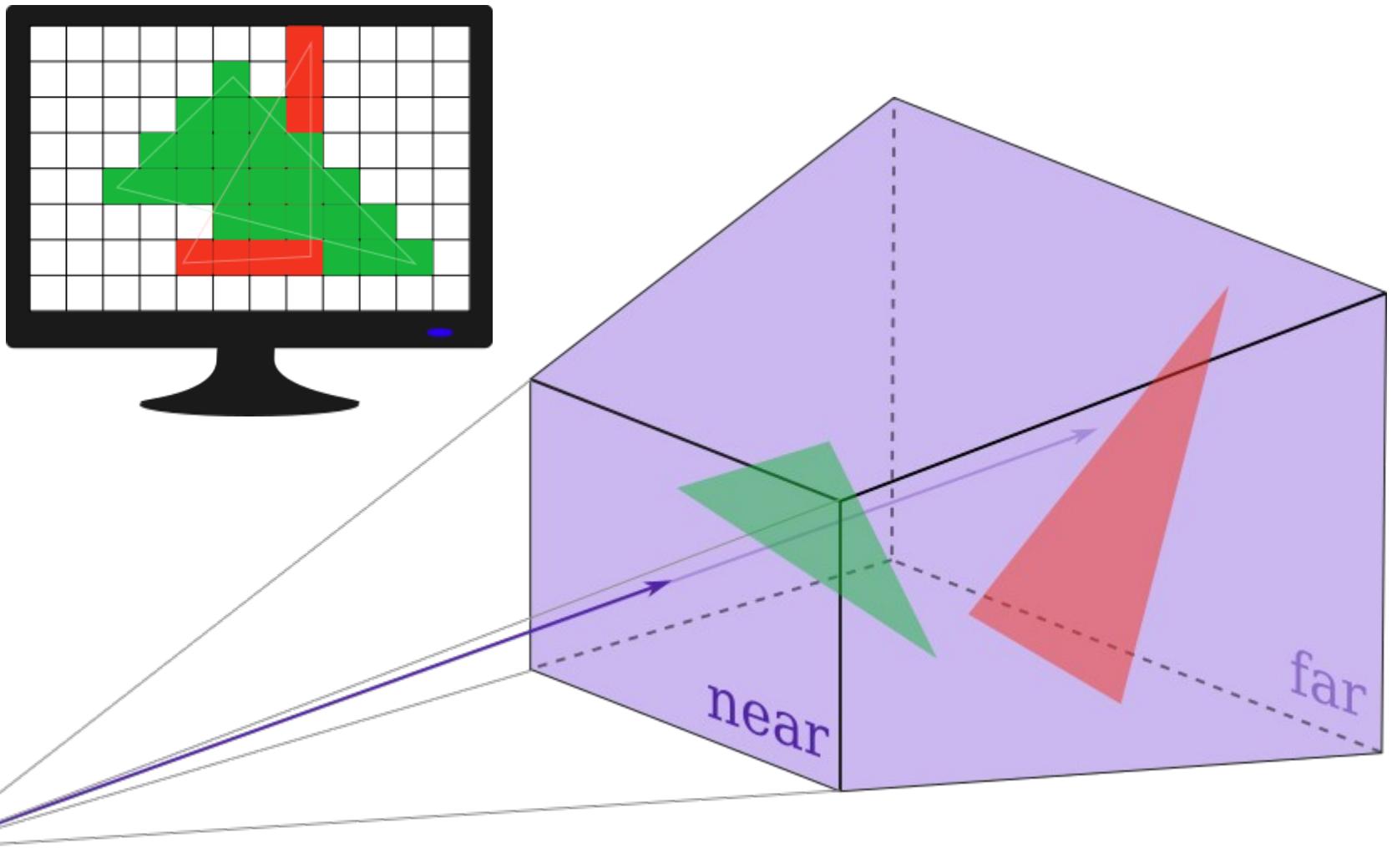
Double Buffer



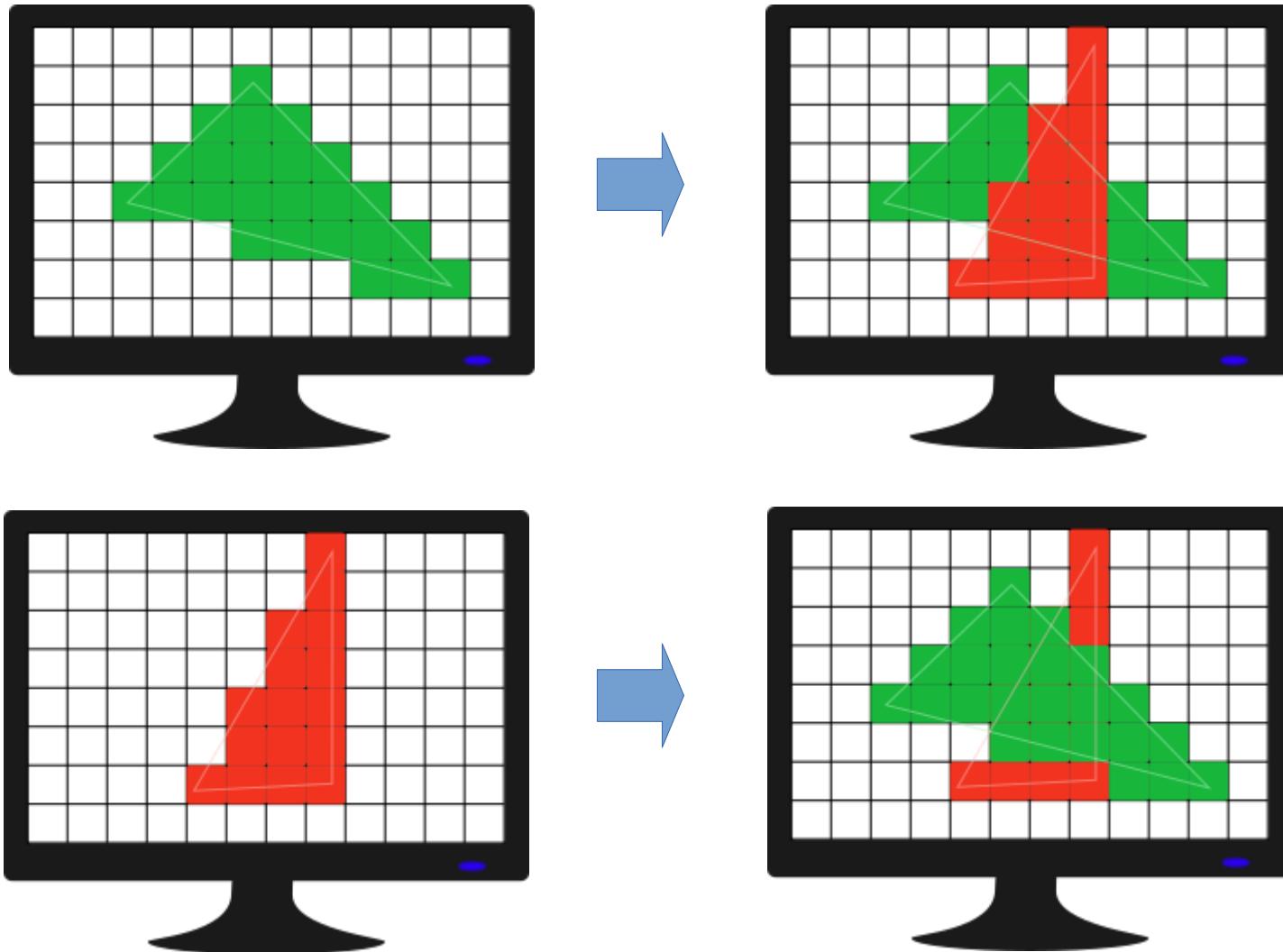
Double Buffer



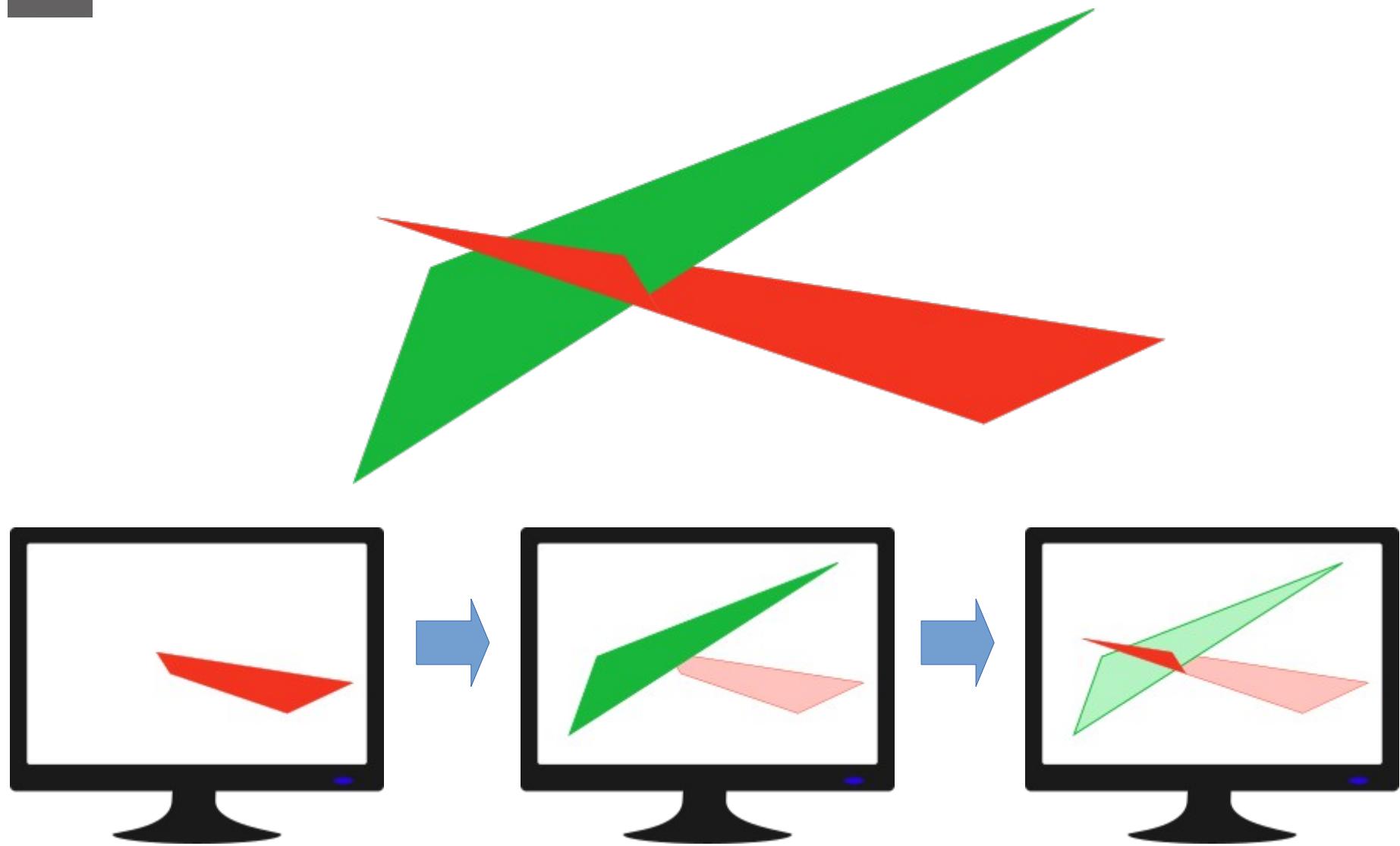
Visibilidad y Rasterización



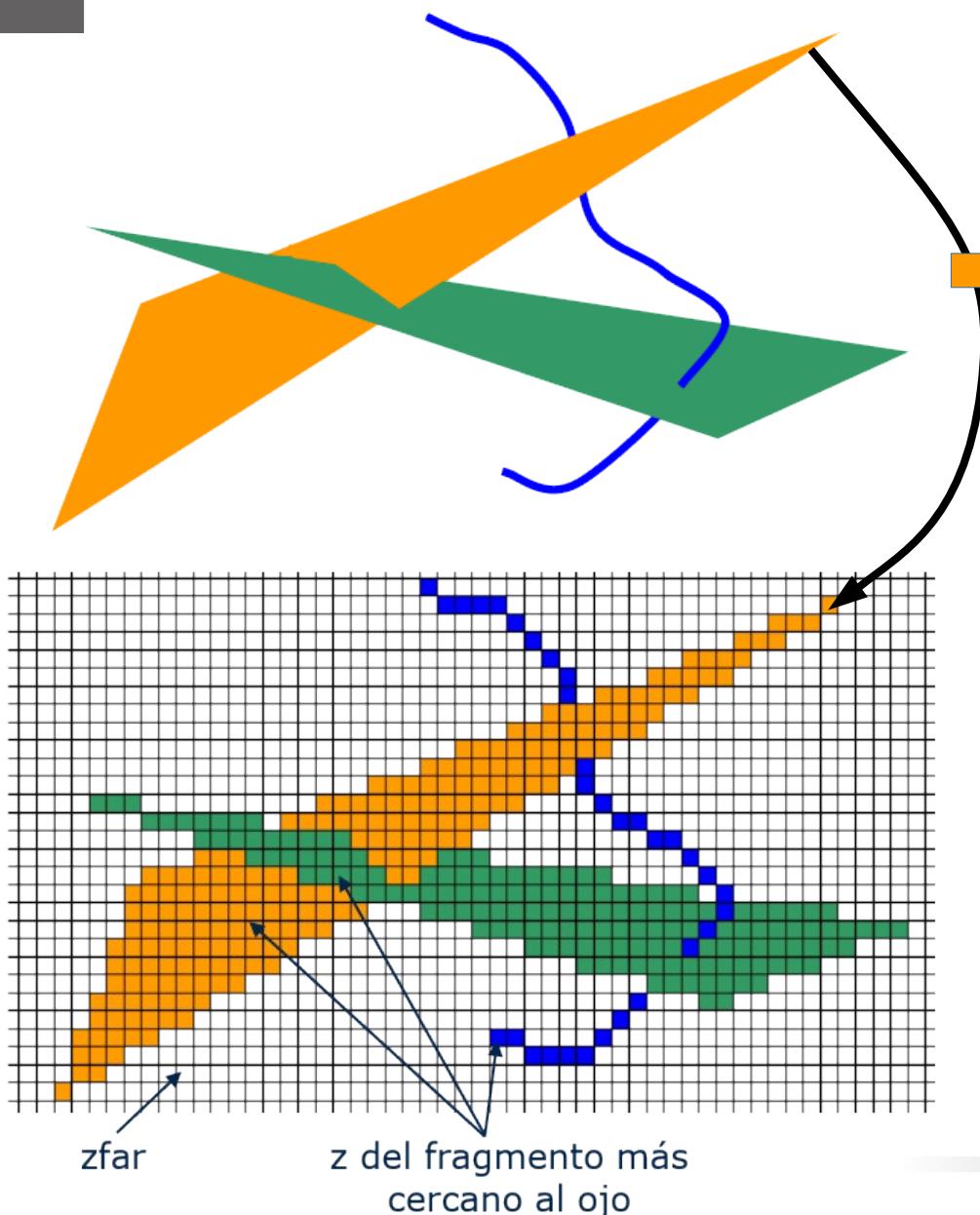
Visibilidad y Rasterización (sin z-buffer)



Visibilidad y Rasterización (sin z-buffer)



Depth Buffer



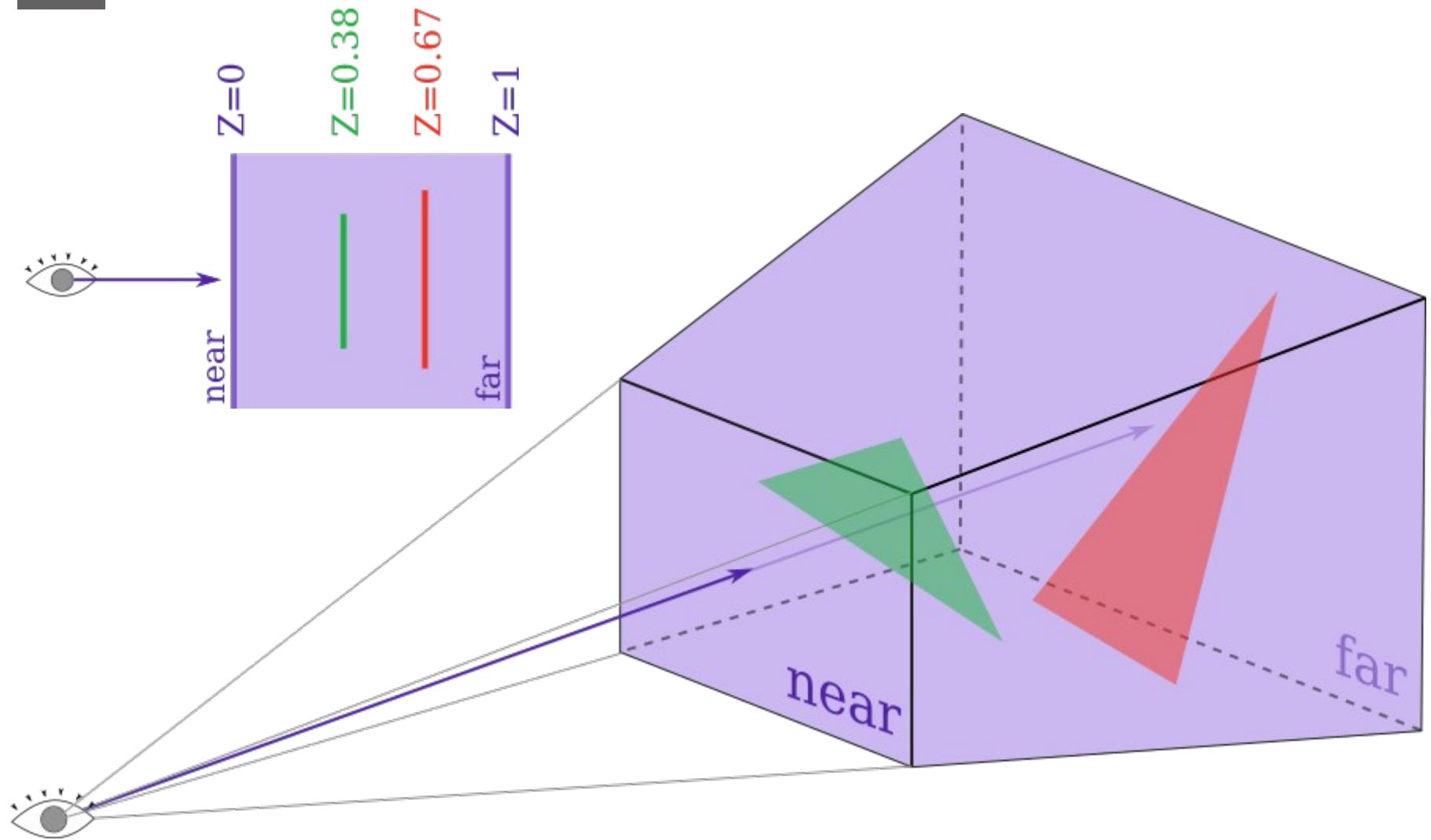
Para todo x, y
 $\text{depth}(x,y) = zfar$

Para cada Primitiva:

Por cada fragmento:

Si $z < \text{depth}(x,y)$
 $\text{color}(x,y) = \text{RGBA}$
 $\text{depth}(x,y) = z$

Algoritmo Z-Buffer



Algoritmo Z-Buffer

Rasterizando 1ro el Verde (z=.38), después el Rojo (z=.67)

Color Buffer

A large 10x10 grid of squares, consisting of 100 individual squares arranged in a single continuous pattern.

Depth Buffer

Color Buffer

A 10x10 grid containing a single, solid green shape. The shape is composed of 18 squares and has a complex, irregular form. It appears to be a collection of various geometric shapes like triangles, rectangles, and trapezoids joined together.

Depth Buffer

1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	.38	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	.38	.38	.38	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	.38	.38	.38	.38	.38	1.0	1.0	1.0	1.0
1.0	1.0	.38	.38	.38	.38	.38	.38	.38	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	.38	.38	.38	.38	.38	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.38	.38	.38	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.38	.38	.38	1.0

Color Buffer

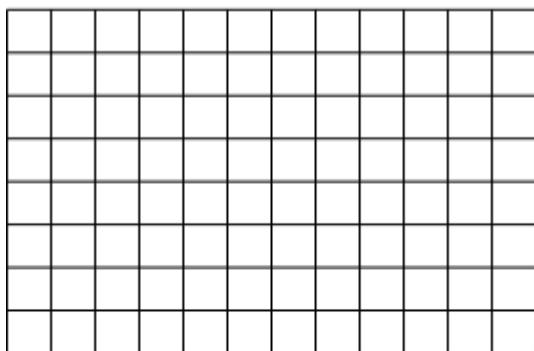
A 10x10 grid containing a pattern of green and red squares. The pattern consists of several L-shaped blocks. A large green L-shaped block is positioned in the center-left area. To its right, there is a vertical column of three red squares. Below the central green block, there is a horizontal row of four green squares. At the bottom, there is a horizontal row of five red squares. The remaining squares in the grid are white.

Depth Buffer

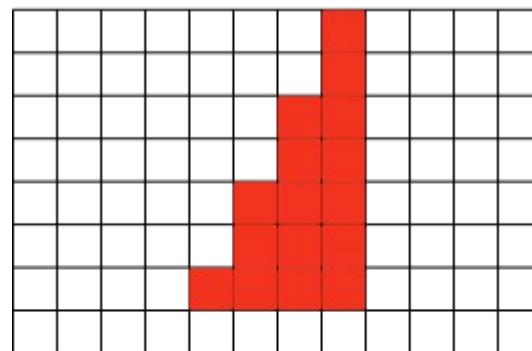
Algoritmo Z-Buffer

Rasterizando 1ro el Rojo ($z=.67$), después el Verde ($z=.38$)

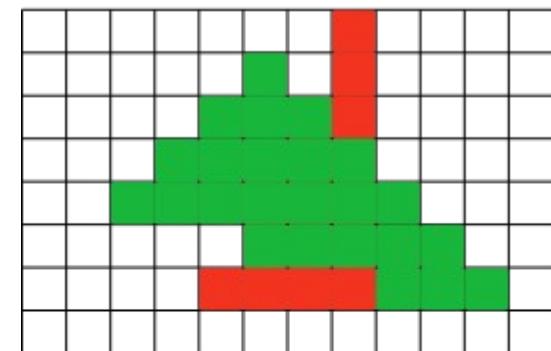
Color Buffer



Color Buffer



Color Buffer



Depth Buffer

1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0



Depth Buffer

1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.67	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.67	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.67	.67	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.67	.67	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.67	.67	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.67	.67	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.67	.67	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.67	.67	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.67	.67	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.67	.67	1.0	1.0	1.0



Depth Buffer

1.0	1.0	1.0	1.0	1.0	1.0	1.0	.67	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	.38	.67	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	.38	.38	.38	.67	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	.38	.38	.38	.38	.38	1.0	1.0	1.0
1.0	1.0	1.0	1.0	.38	.38	.38	.38	.38	.38	1.0	1.0	1.0
1.0	1.0	1.0	.38	.38	.38	.38	.38	.38	.38	1.0	1.0	1.0
1.0	1.0	.38	.38	.38	.38	.38	.38	.38	.38	1.0	1.0	1.0
1.0	.38	.38	.38	.38	.38	.38	.38	.38	.38	1.0	1.0	1.0
.38	.38	.38	.38	.38	.38	.38	.38	.38	.38	1.0	1.0	1.0
.38	.38	.38	.38	.38	.38	.38	.38	.38	.38	1.0	1.0	1.0

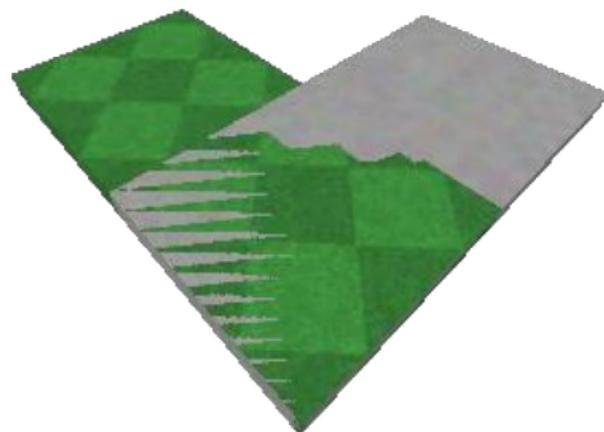
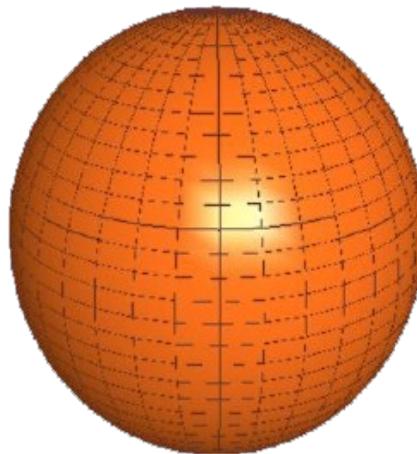
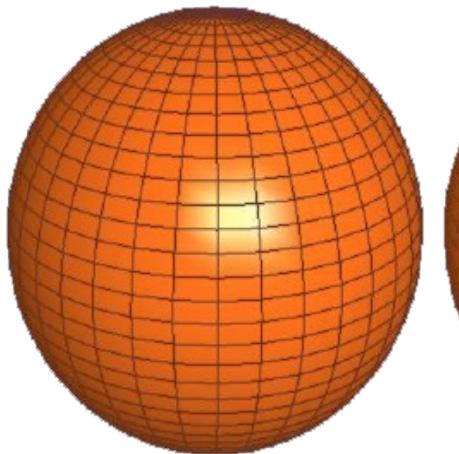
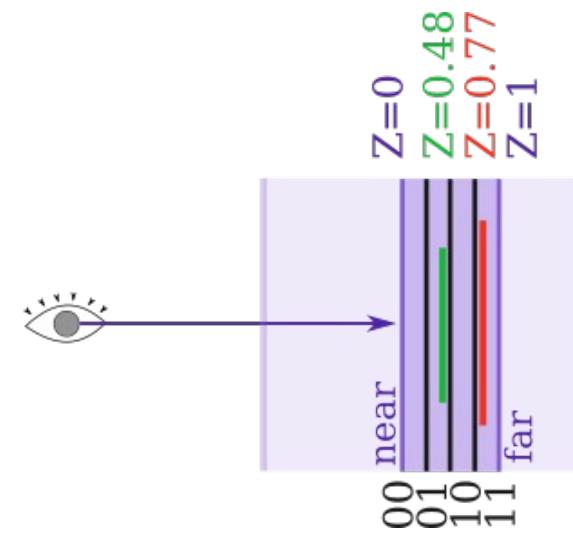
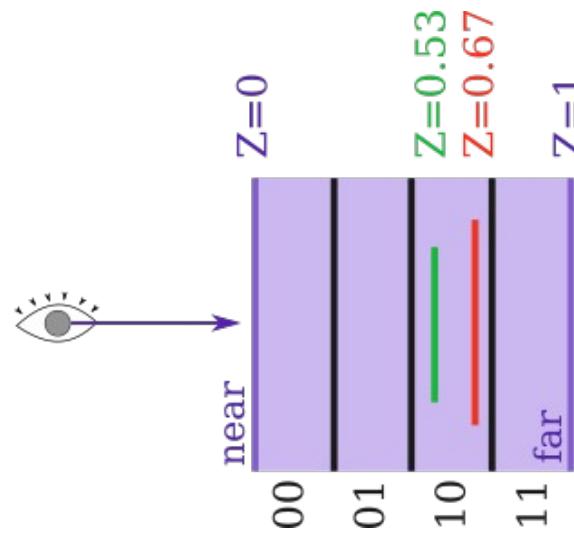
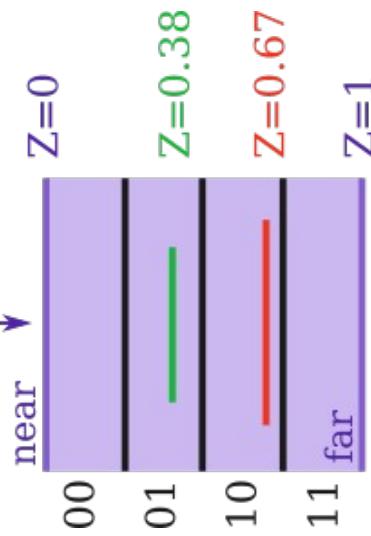
Depth Buffer

Color buffer



Depth buffer

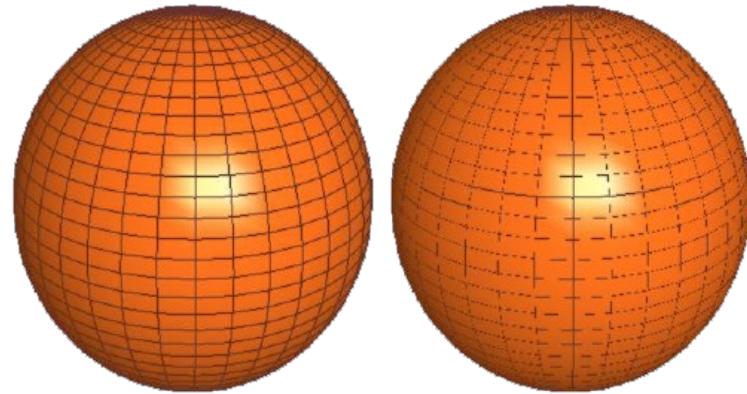
Depth Buffer: Z-Fighting



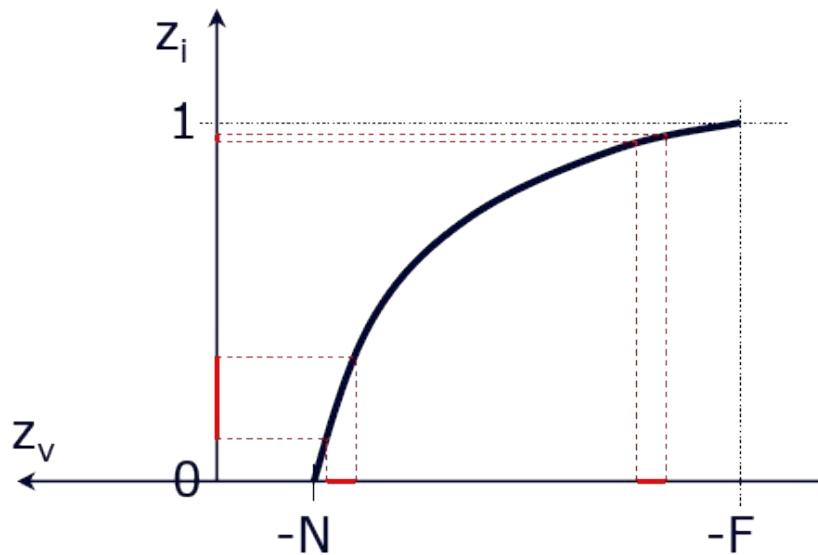
Depth Buffer: Z-Fighting

Modificar los valores de Z

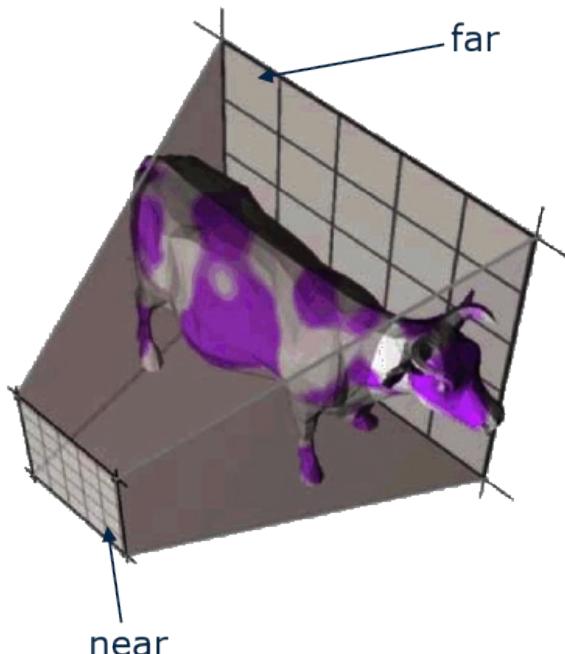
```
void glPolygonOffset(GLfloat factor,  
                     GLfloat units)
```



Ajustar los planos Near y Far



mayor precisión
cerca del plano near



Depth Buffer: Z Visual vs. Z de Imagen

Frustum:

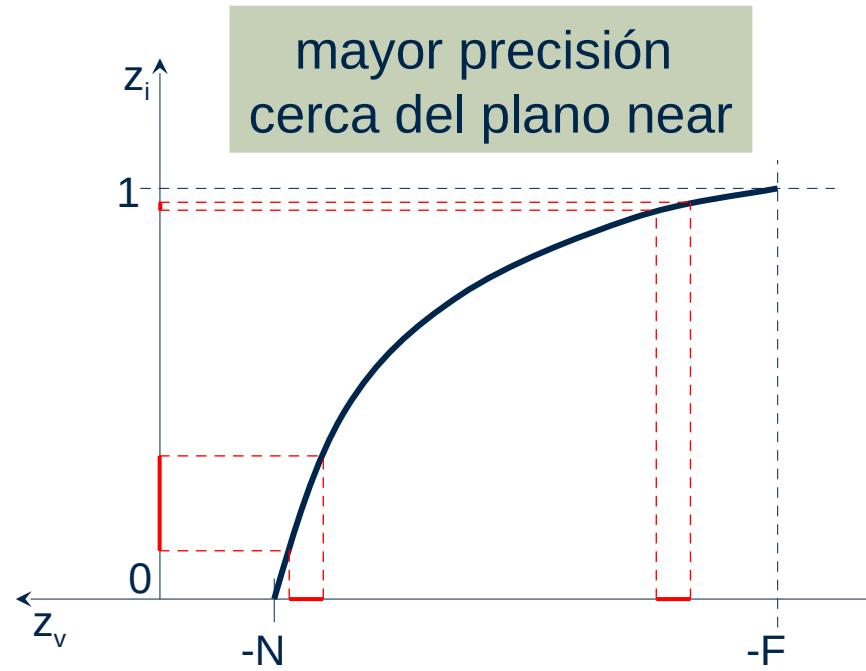
- Far
- Near
- Right
- Left
- Top
- Bottom

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$z_{\text{imagen}} = \frac{F+N}{F-N} + \frac{1}{z} \frac{2FN}{F-N} = a + \frac{b}{z}$$

$$\begin{pmatrix} \frac{2N}{R-L} & 0 & \frac{R+L}{R-L} & 0 \\ 0 & \frac{2N}{T-B} & \frac{T+B}{T-B} & 0 \\ 0 & 0 & -\frac{F+N}{F-N} & \frac{-2FN}{F-N} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} \dots \\ \dots \\ -\frac{F+N}{F-N}z - \frac{2FN}{F-N} \\ -z \end{pmatrix}$$

Matriz Proyectiva



Depth Buffer: Z-Fighting

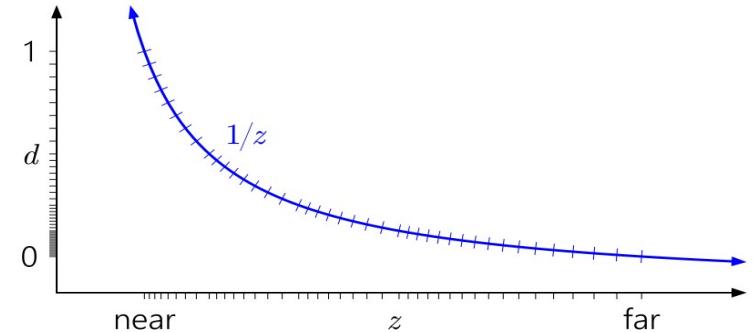
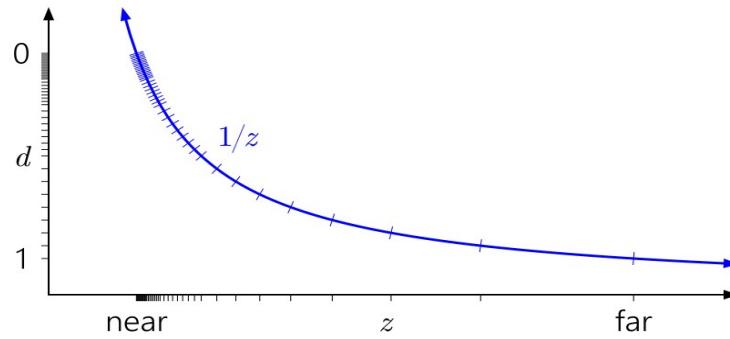
En la práctica hay muchos otros trucos y detalles que considerar...

- Ej: Profundidad logarítmica:

$$z_{\text{imagen}} = \frac{2 \log(z - N + 1)}{\log(F - N + 1)} - 1$$



- Ej: Z invertido



- Y más...

<http://www.reedbeta.com/blog/depth-precision-visualized/>

Algoritmo Z-Buffer y Transparencias

Rasterizando 1ro el Verde (z=.38), después el Rojo (z=.67)

Color Buffer

A blank 10x10 grid for drawing or plotting.

Depth Buffer

Color Buffer

A 10x10 grid of squares. The cells are shaded green in a specific pattern: the first column is entirely white, the second column has green cells from row 2 to 5, the third column has green cells from row 2 to 7, the fourth column has green cells from row 2 to 9, the fifth column has green cells from row 2 to 10, and the sixth through tenth columns are entirely white. This creates a shape resembling the uppercase letter 'E'.

Depth Buffer

1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	.38	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	.38	.38	.38	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	.38	.38	.38	.38	.38	1.0	1.0	1.0	1.0
1.0	.38	.38	.38	.38	.38	.38	.38	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	.38	.38	.38	.38	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.38	.38	.38	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.38	.38	.38	1.0

Color Buffer

A 10x10 grid containing a pattern of green and red squares. The pattern consists of a central column of green squares, flanked by two columns of red squares. The red squares form a U-shape around the green column. The entire pattern is centered within the grid.

Depth Buffer

Depth Buffer: Transparencias Múltiples

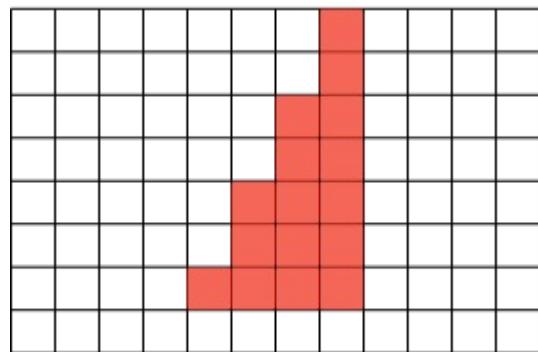
Pocos objetos transparentes → Ordenamiento espacial



<http://www.adriancourreges.com/blog/2015/11/02/gta-v-graphics-study/>

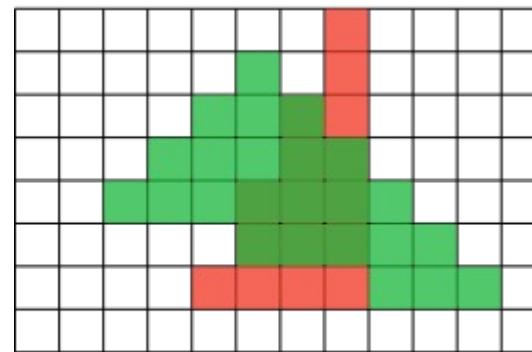
Transparencias Múltiples

Color Buffer



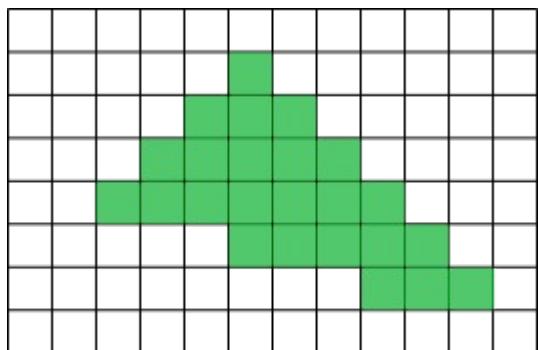
$$3/4 R + 1/4 B$$

Color Buffer

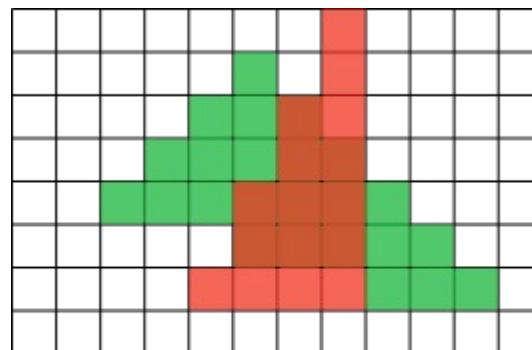


$$3/4 V + 1/4 (3/4 R + 1/4 B)$$

$$3/4 V + 3/16 R + 1/16 B$$



$$3/4 V + 1/4 B$$



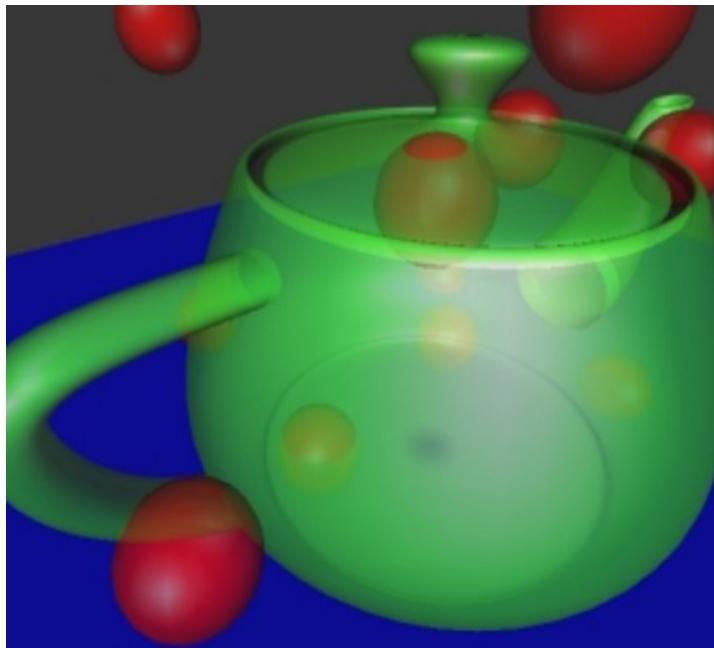
$$3/4 R + 1/4 (3/4 V + 1/4 B)$$

$$3/4 R + 1/16 V + 1/16 B$$

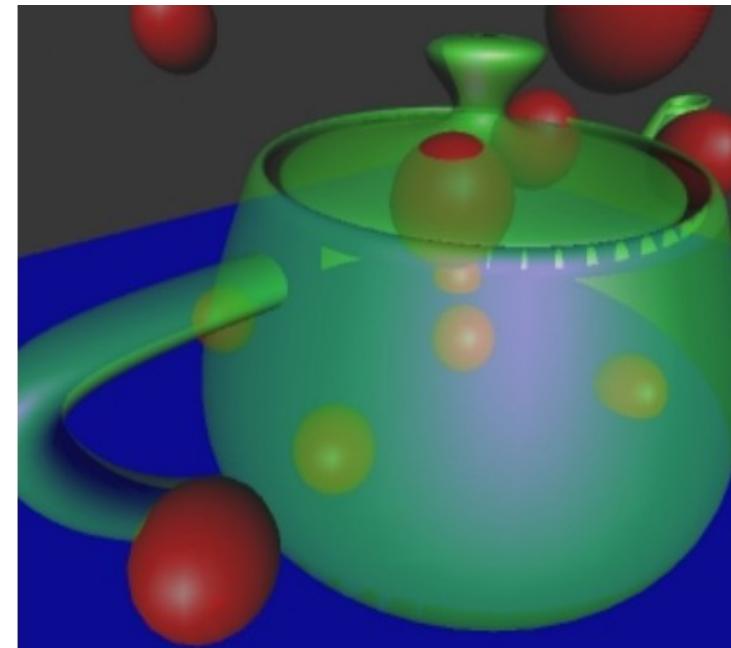
Depth Buffer: Transparencias Múltiples

Escenas más complejas → Técnicas especiales

Orden correcto



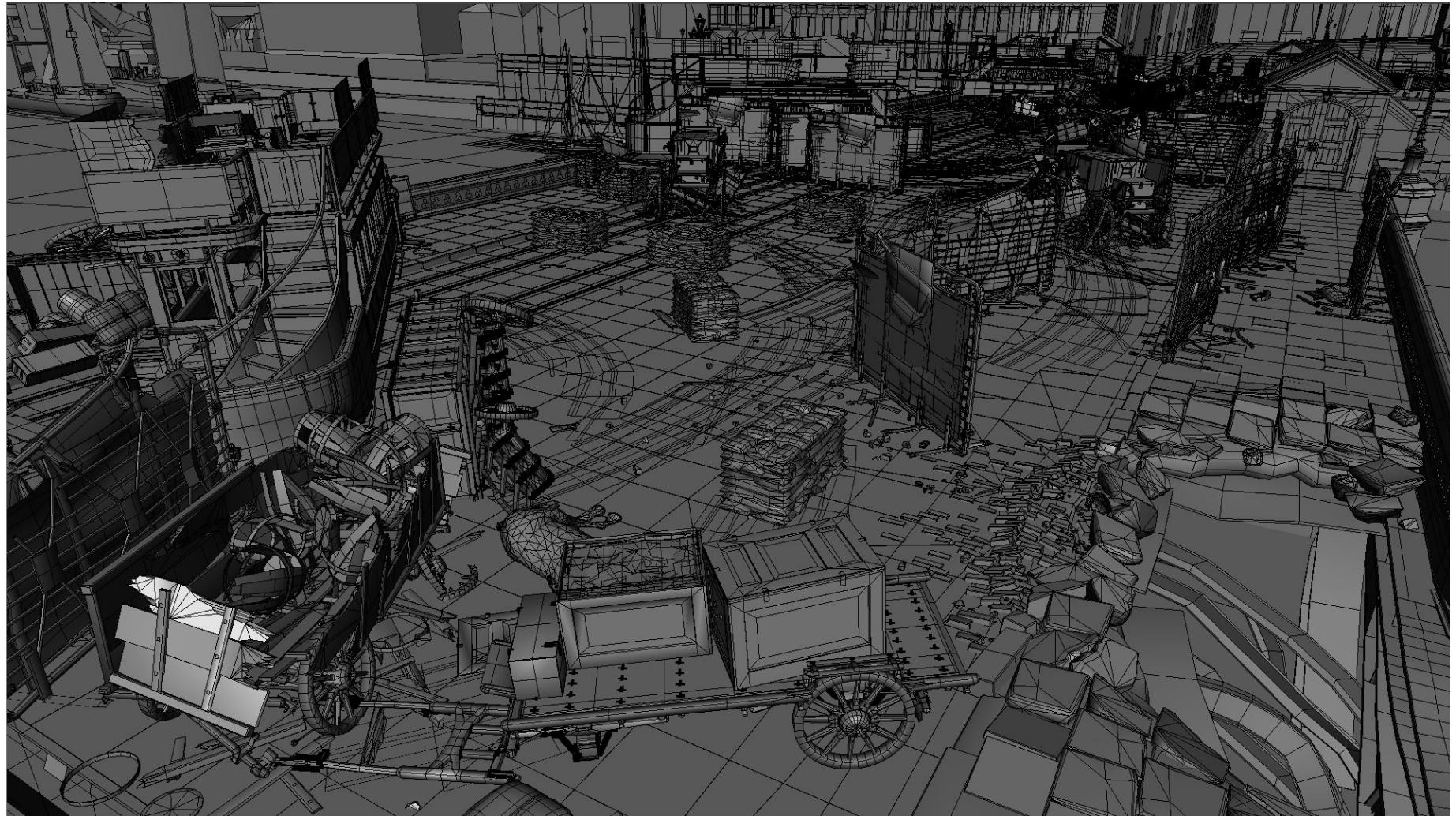
Orden incorrecto



Algunas:

- Depth peeling (doble z-buffer)
- Buffers con linked lists
- Order independent blending (más adelante)

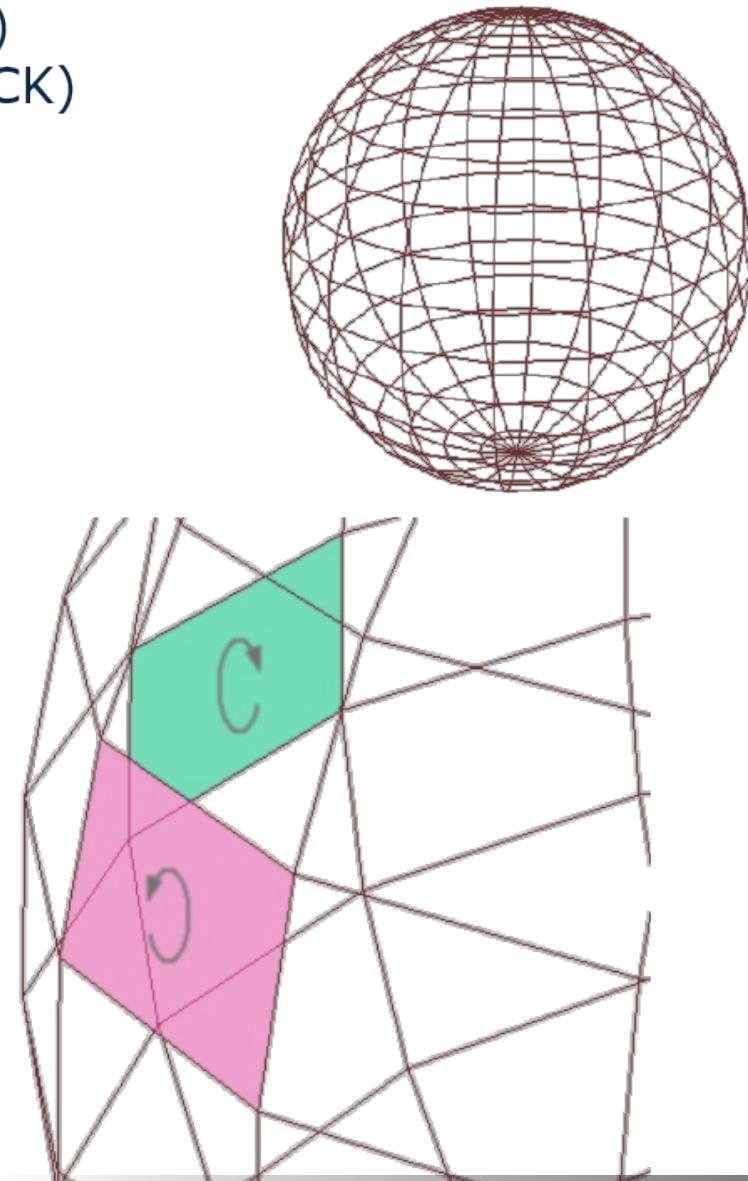
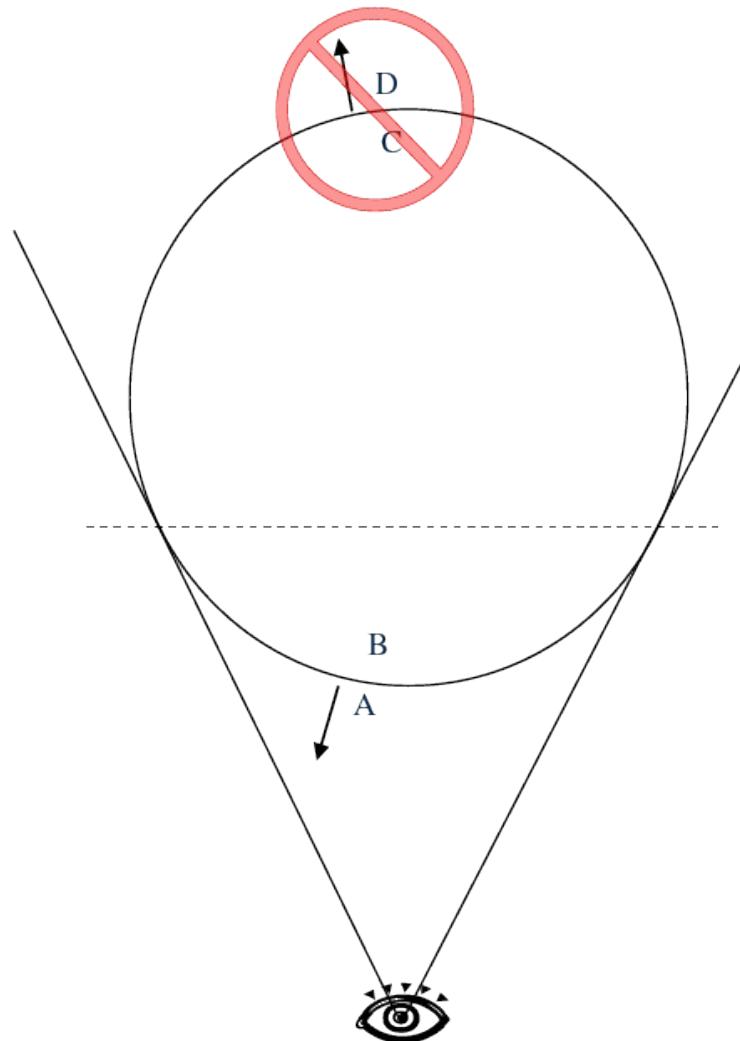
Depth Buffer: Escenas Complejas



Las escenas complejas requieren
rasterizar muchas primitivas

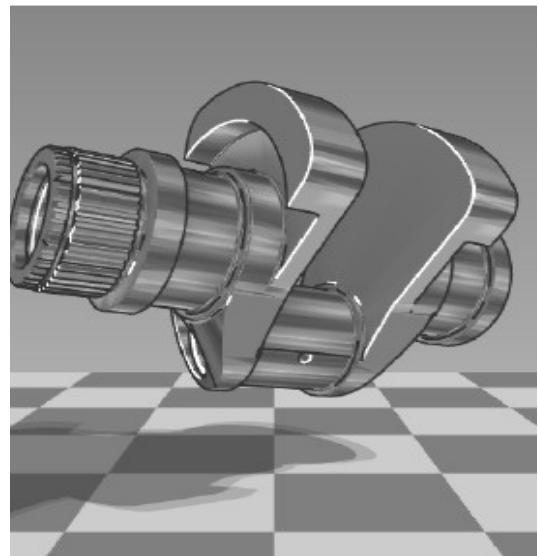
Backface Culling

`glFrontFace(GL_CW or GL_CCW)`
`glCullFace(GL_FRONT or GL_BACK)`

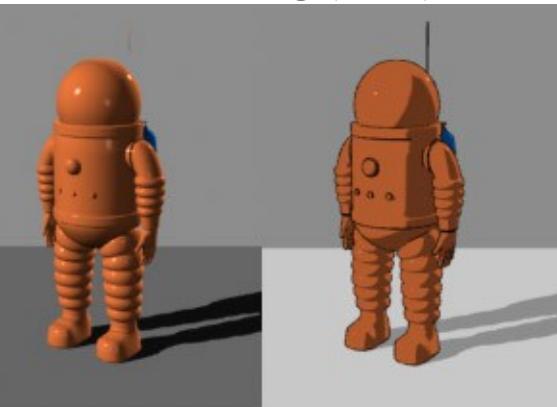


Depth Buffer: Otros Usos

Siluetas para mejorar la visualización



Screen Space Ambient Occlusion (SSAO)



plastic shader

toon shader

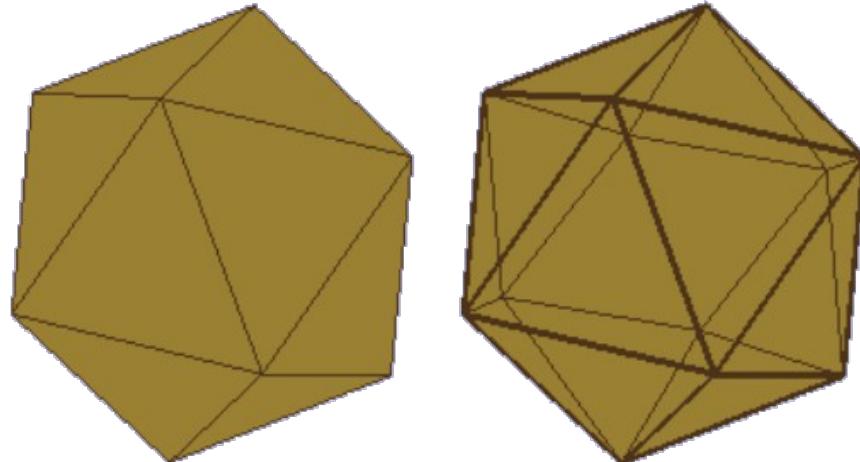
without ambient occlusion

with ambient occlusion

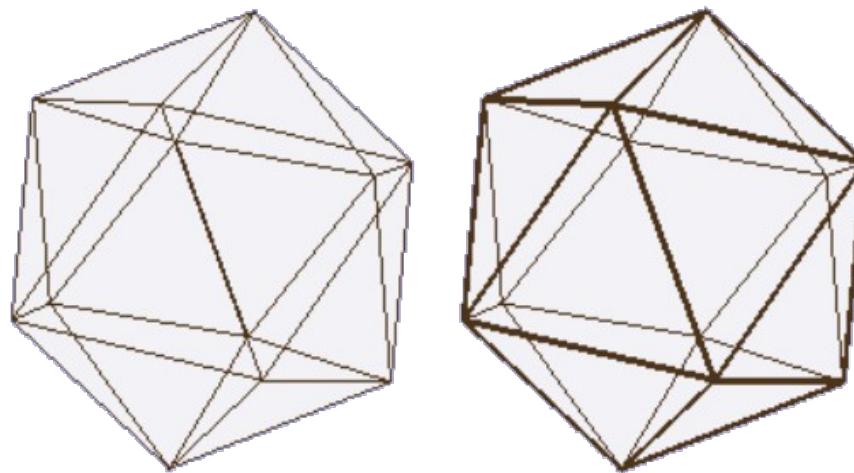
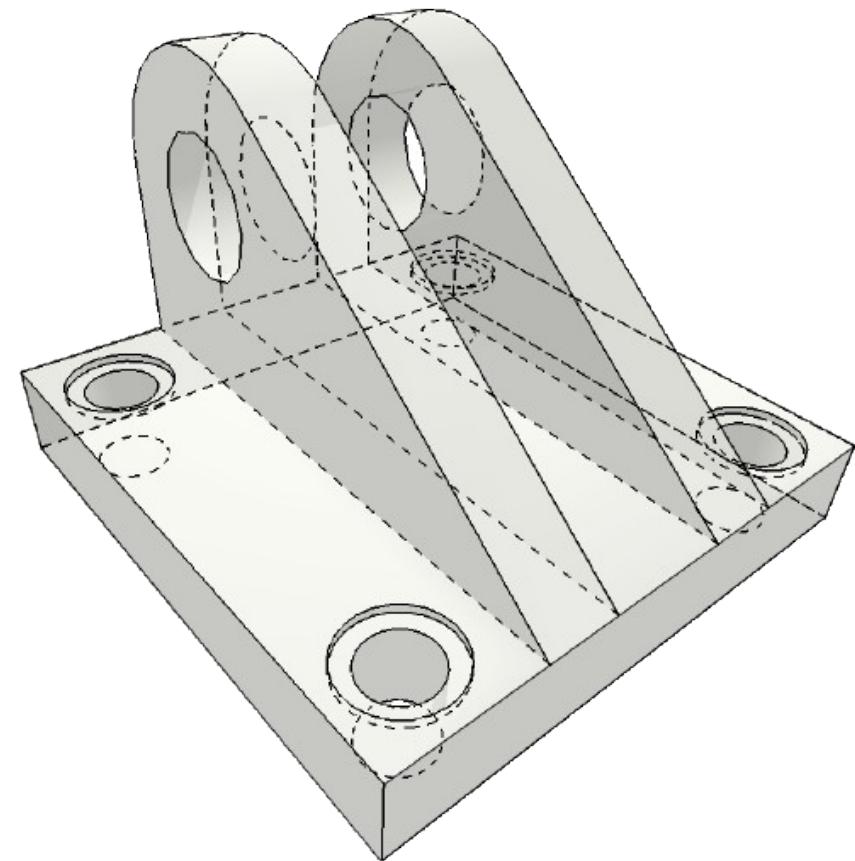
Espacio de la Imagen

Depth Buffer: Otros Usos

Lineas ocultas

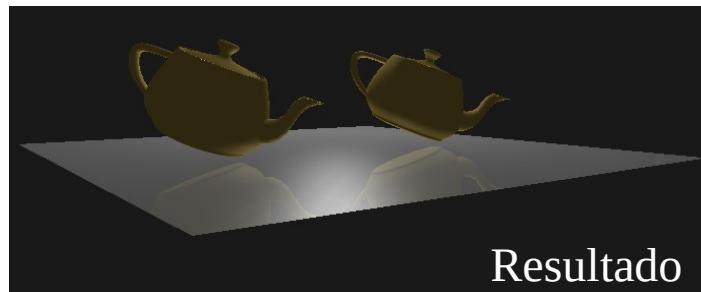
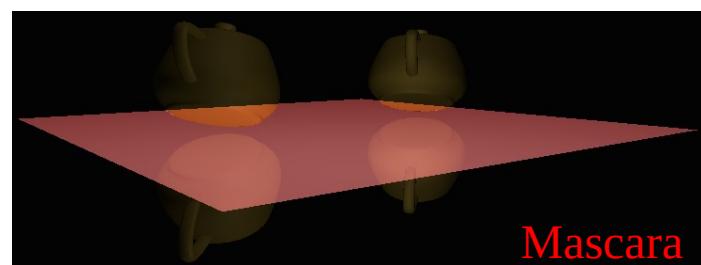
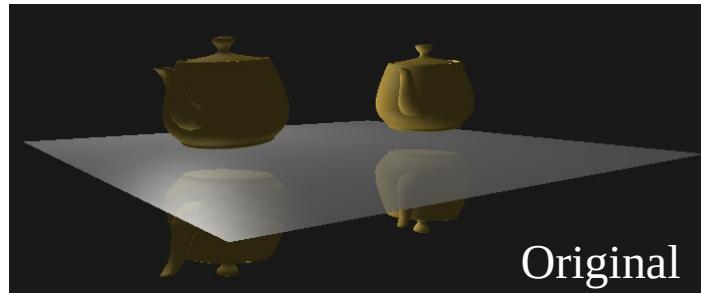
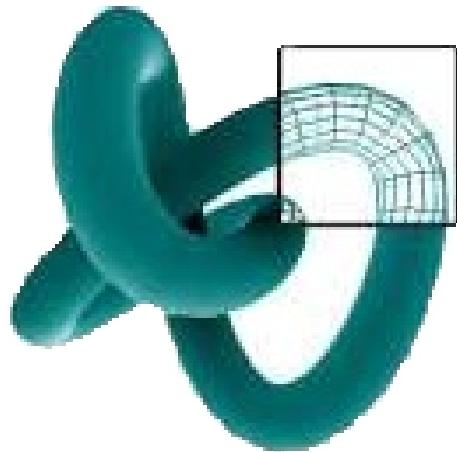


`glColorMask(R,G,B,A)`
`glDepthFunc(comparador)`

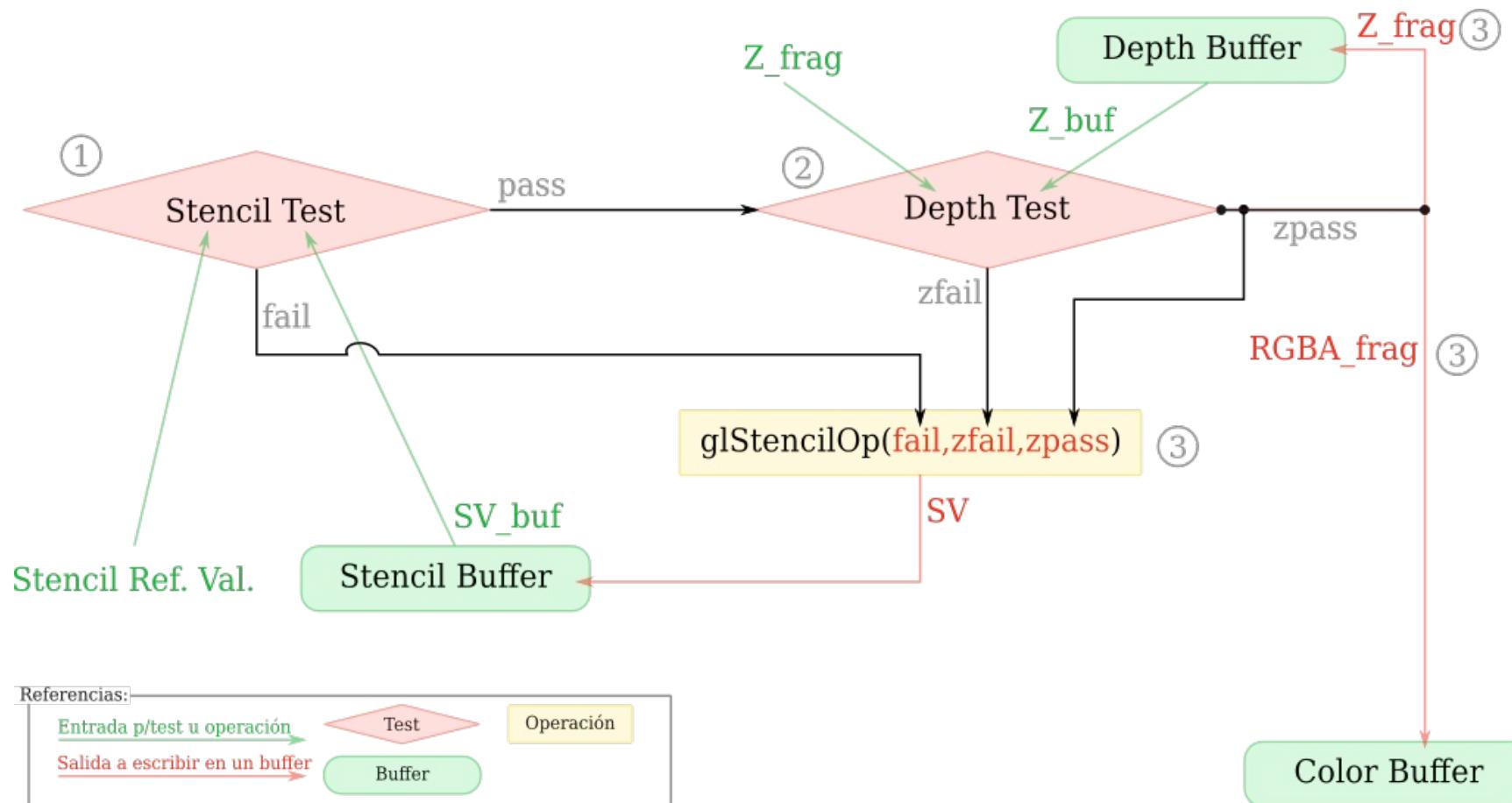


Stencil Buffer: Aplicaciones

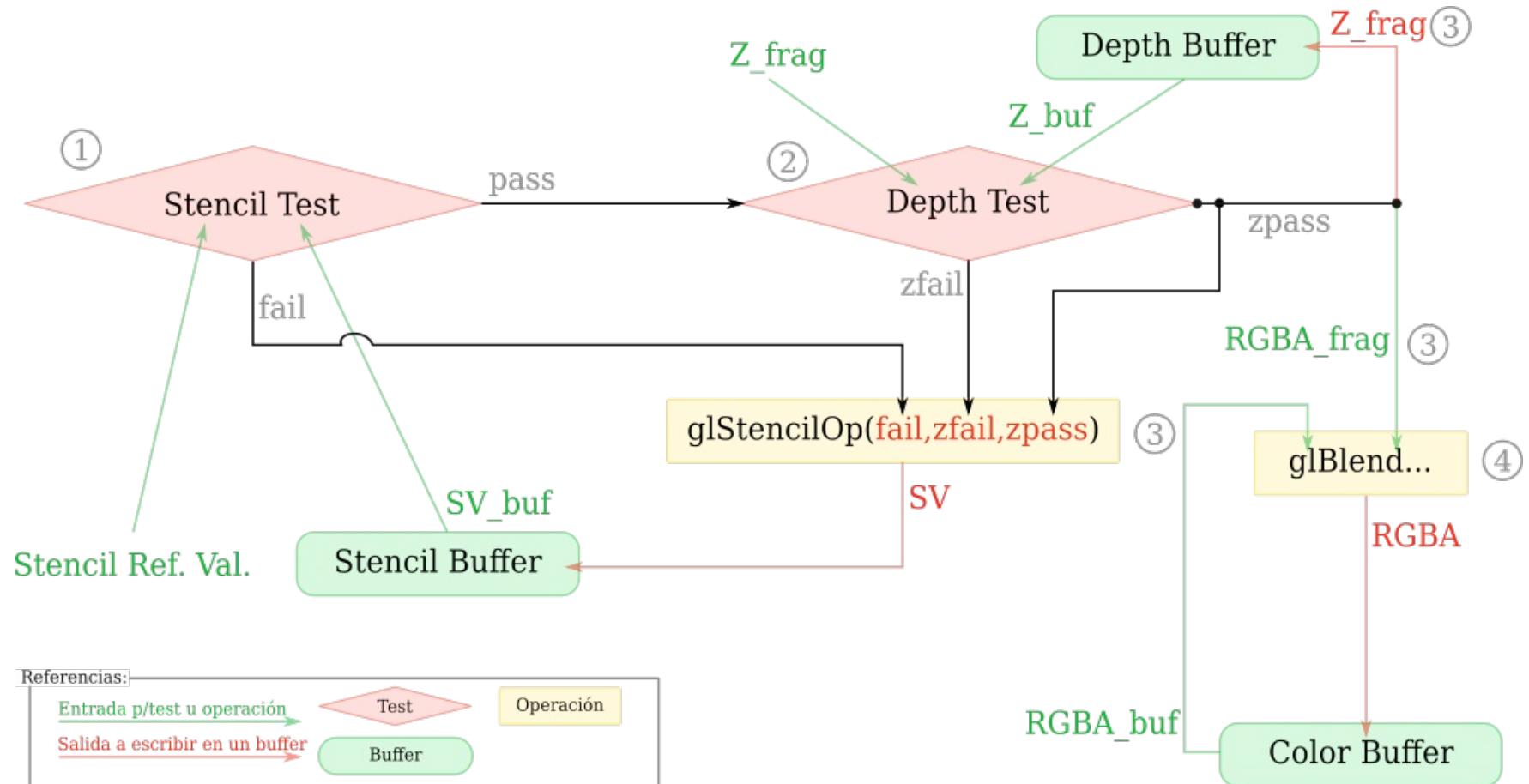
- Reflejos y Espejos
- Sombras
- Outlining y Siluetas
- Visualización
- Refracción del agua
- Aplicar efectos a ciertos pixels
- Optimización
- Y más



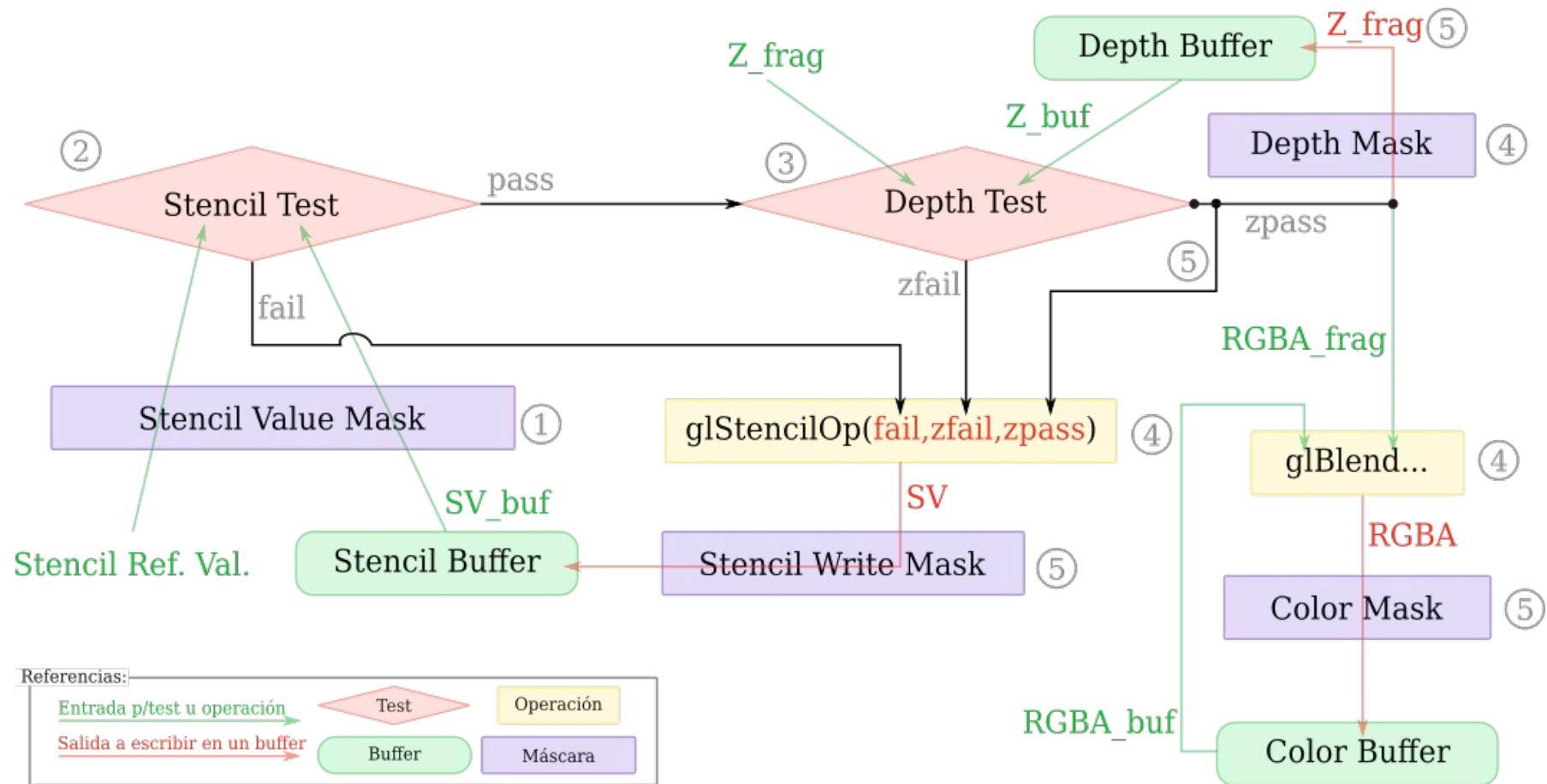
Stencil + Depth + Color



Stencil + Depth + Color

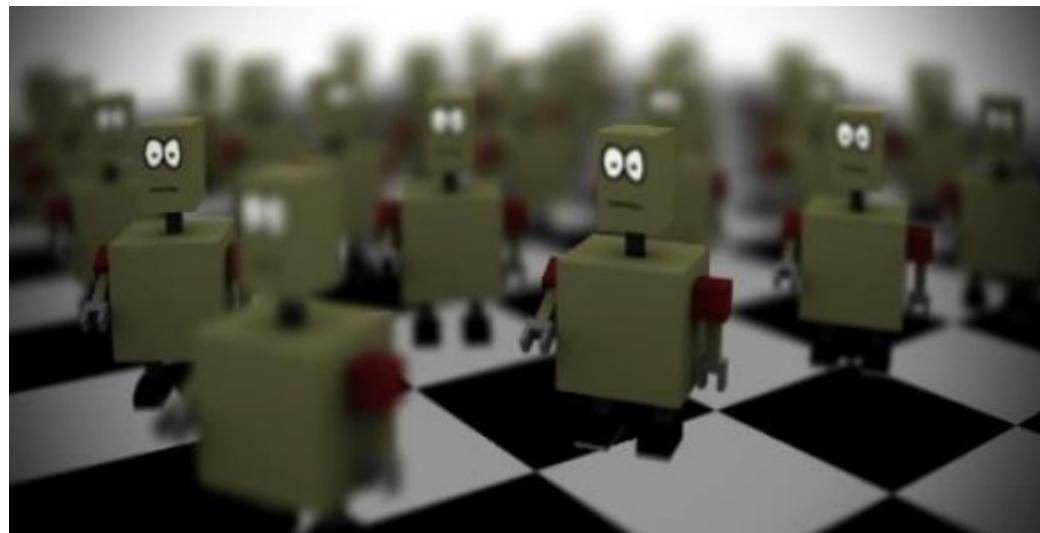


Stencil + Depth + Color



Accumulation Buffer

- Buffer de color “especial”
- 16 bits por color (48 bits)
- Reemplazado por FBOs
- Aplicaciones
 - Motion blur
 - Depth of field
 - Antialising
 - Soft shadows



Tests de Fragmentos

Rasterización

Ownership

Scissor

Alpha

Stencil

Depth

Actualización de
buffers

`glEnable(GL_xxxxxx_TEST)`

`glDisable(GL_xxxxxx_TEST)`

`glXxxxxxFunc(comp, [ref], [mask])`

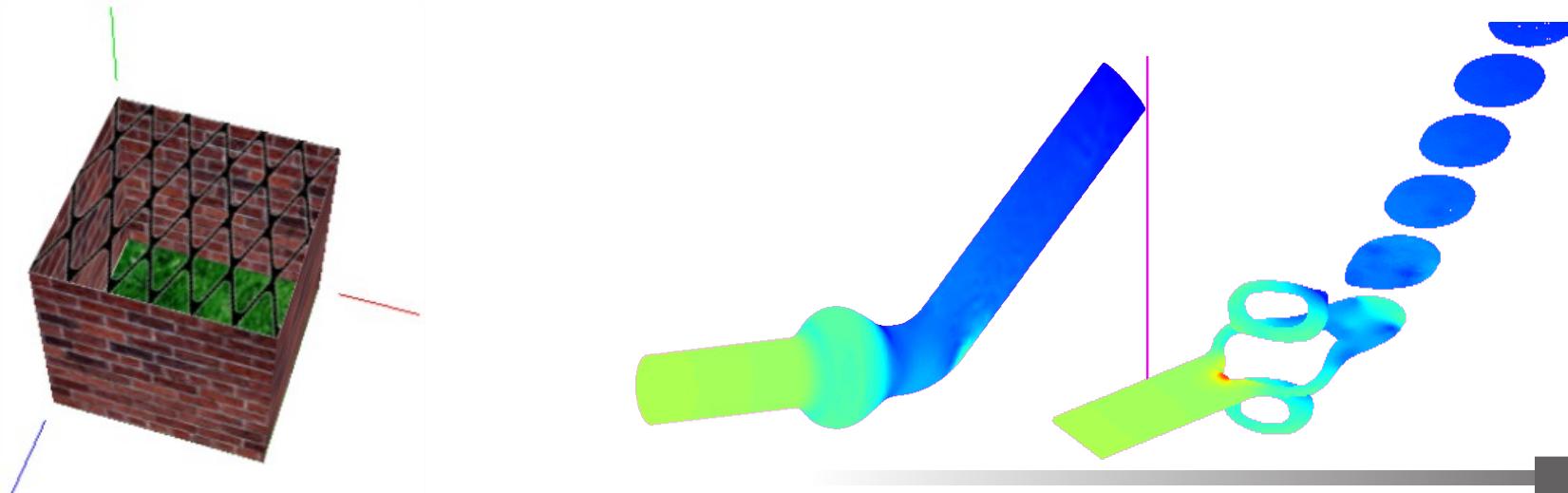
comp: posibles comparadores

- `GL_NEVER`
- `GL_ALWAYS`
- `GL_LESS`
- `GL_EQUAL`
- `GL_GREATER`
- `GL_NOTEQUAL`

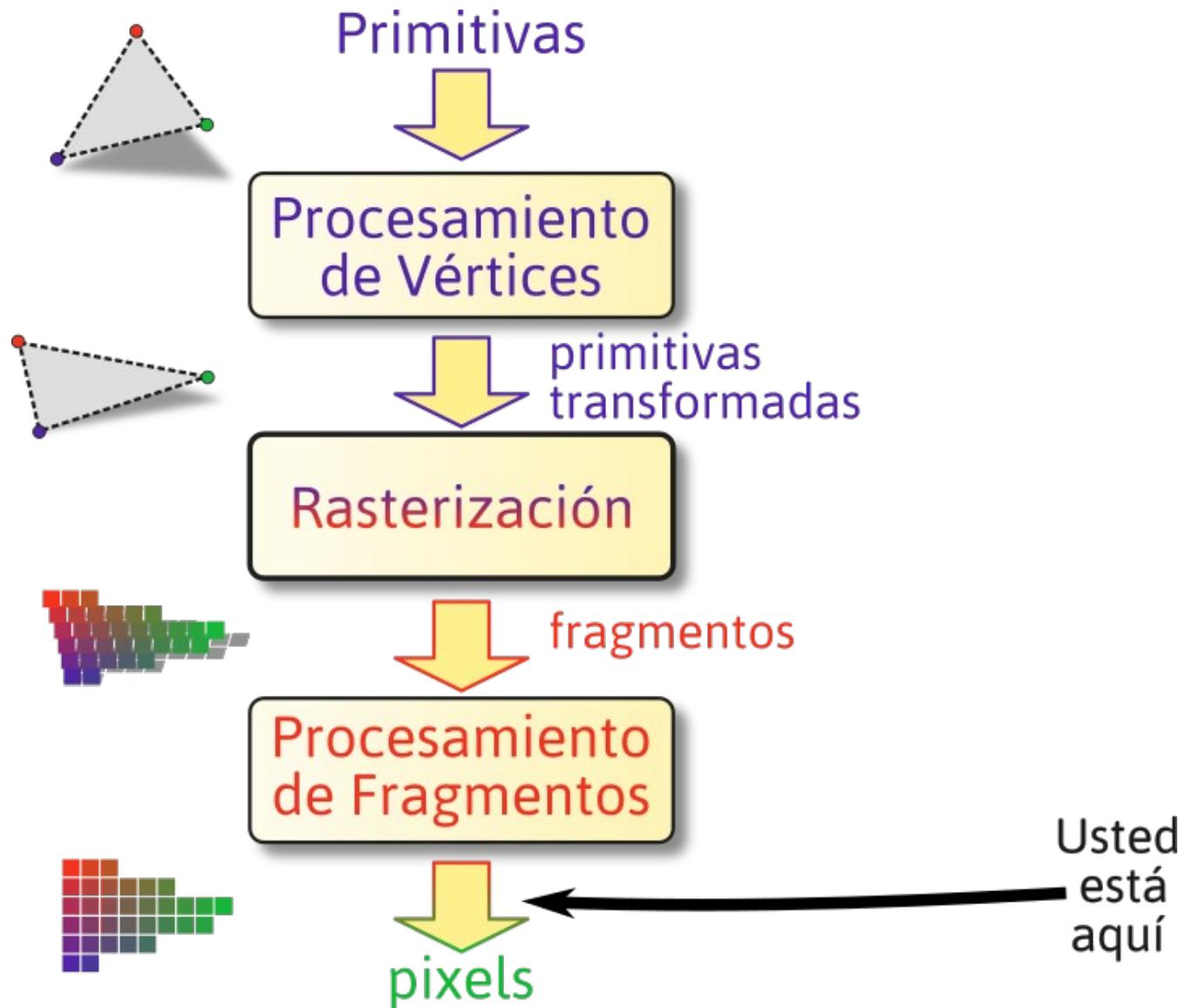
ref: valor con el que se compara
(alpha y stencil)

mask: para enmascarar bits en la
comparación (solo stencil)

Tests de Fragmentos: Alpha-Test



Usted Está Aquí



Alteración de Fragmentos: Blending

• <code>glBlendEquation(op)</code>	$D = F_s S \text{ op } F_d D$
<code>glBlendFunc(F_s, F_d)</code>	D: color almacenado en el buffer S: color del fragmento entrante
<code>GL_FUNC_ADD</code>	$D = F_s S + F_d D$
<code>GL_FUNC_SUBTRACT</code>	$D = F_s S - F_d D$
<code>GL_FUNC_REVERSE_SUBTRACT</code>	$D = F_d D - F_s S$
<code>GL_MIN</code>	$D = \min(F_s S, F_d D)$
<code>GL_MAX</code>	$D = \max(F_s S, F_d D)$

Alteración de Fragmentos: Blending

$$glBlendEquation(op) \quad glBlendFunc(F_s, F_d) \quad \left. \begin{array}{l} D = F_s S \text{ op } F_d D \\ D: \text{color almacenado en el buffer} \\ S: \text{color del fragmento entrante} \end{array} \right\}$$

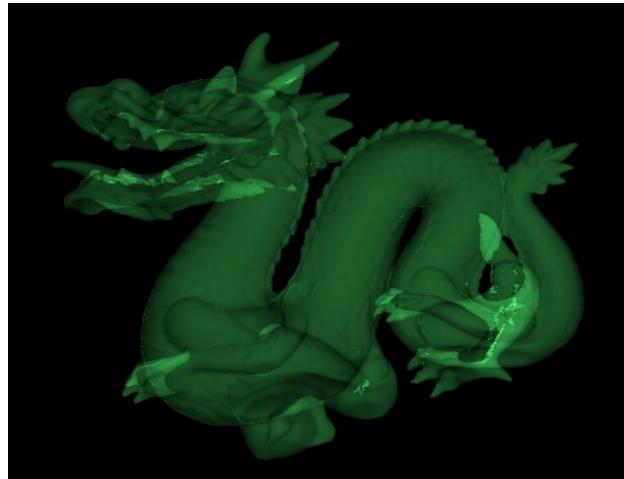
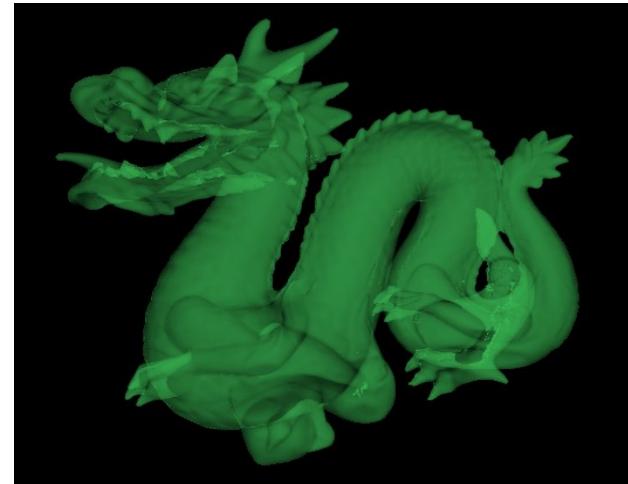
GL_ZERO	(0, 0, 0, 0)
GL_ONE	(1, 1, 1, 1)
GL_DST_COLOR	(R _d , G _d , B _d , A _d)
GL_SRC_COLOR	(R _s , G _s , B _s , A _s)
GL_ONE_MINUS_DST_COLOR	(1, 1, 1, 1) - (R _d , G _d , B _d , A _d)
GL_ONE_MINUS_SRC_COLOR	(1, 1, 1, 1) - (R _s , G _s , B _s , A _s)
GL_SRC_ALPHA	(A _s , A _s , A _s , A _s)
GL_ONE_MINUS_SRC_ALPHA	(1, 1, 1, 1) - (A _s , A _s , A _s , A _s)
GL_DST_ALPHA	(A _d , A _d , A _d , A _d)
GL_ONE_MINUS_DST_ALPHA	(1, 1, 1, 1) - (A _d , A _d , A _d , A _d)
GL_SRC_ALPHA_SATURATE	(f, f, f, 1); f=min(A _s , 1-A _d)

Alteración de Fragmentos: Blending

$$D = A_s S + (1 - A_s) D$$

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)
```

```
glBlendEquation(GL_FUNC_ADD)
```



$$D = S + D$$

```
glBlendFunc(GL_ONE, GL_ONE)  
glBlendEquation(GL_FUNC_ADD)
```



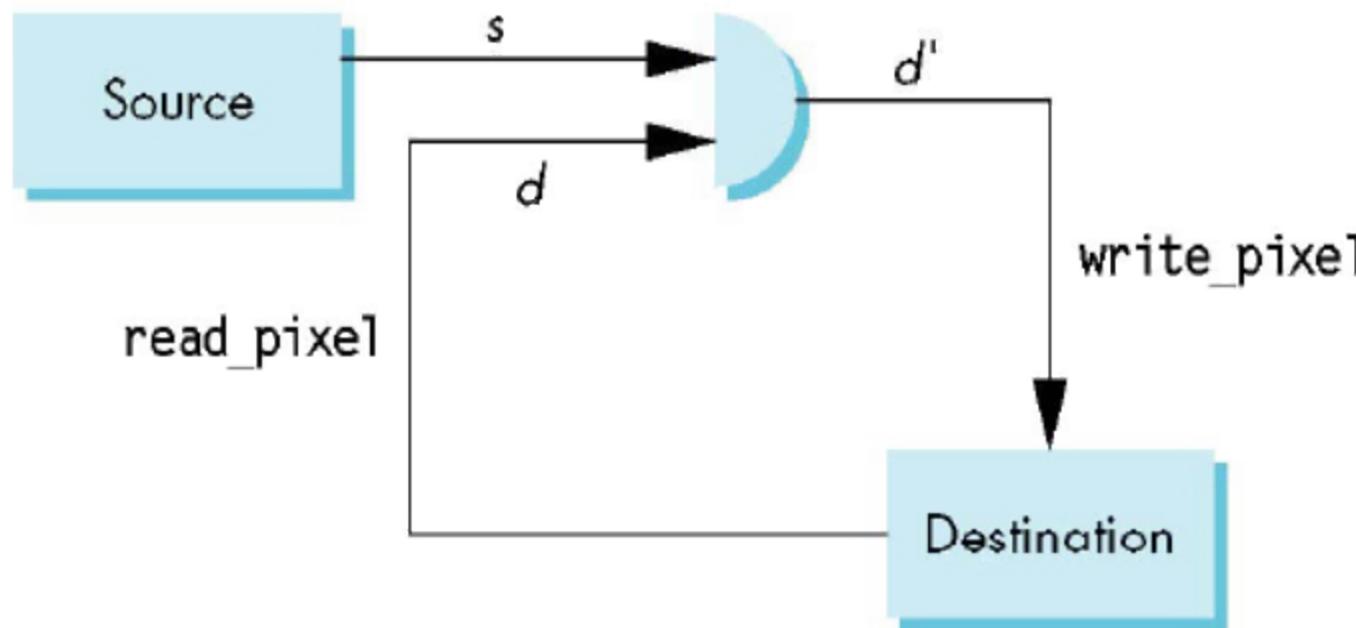
$$D = S \cdot D$$

```
glBlendFunc(GL_ZERO, GL_SRC_COLOR)
```

Alteración de Fragmentos: Operaciones Lógicas

`glLogicOp(op)` \rightarrow $D = S \text{ op } D$

D: color almacenado en el buffer
S: color del fragmento entrante



Alteración de Fragmentos: Operaciones Lógicas

glLogicOp(op) \rightarrow D = S op D

D: color almacenado en el buffer
S: color del fragmento entrante

Parameter	Operation	Parameter	Operation
GL_CLEAR	0	GL_AND	s&d
GL_COPY	s	GL_OR	s d
GL_NOOP	d	GL_NAND	$\sim(s \& d)$
GL_SET	1	GL_NOR	$\sim(s d)$
GL_COPY_INVERTED	$\sim s$	GL_XOR	$s \wedge d$
GL_INVERT	$\sim d$	GL_EQUIV	$\sim(s \wedge d)$
GL_AND_REVERSE	$s \& \sim d$	GL_AND_INVERTED	$\sim s \& d$
GL_OR_REVERSE	$s \sim d$	GL_OR_INVERTED	$\sim s d$

Deben saber

Image precision vs. Model precision

Depth buffer: ventajas, problemas y usos

Stencil buffer: funcionamiento, usos

Tests: usos, técnicas multipaso

Blending y operaciones lógicas

Bibliografía

Red-Book:

- 5: A Hidden-Surface Removal Survival Kit
- 6: Blending, Polygon Offset
- 8: Lo que quieran saber sobre imágenes
- 10: Framebuffer (todo)
- 14: Usos alternativos

El material extra es para curiosear, ver otras fuentes o una punta para aprender más, no es necesario estudiarlo.