# PROGRAMACIÓN 3D

Máster en Programación de Videojuegos

## TEMA 6: Mezclado de colores

Juan Mira Núñez

U-tad
CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

# Índice

- Introducción
- Transparencia
- Mezclado de colores
- Orden de pintado

# Mezclado de colores

# Introducción.

- Las texturas de OpenGL soportan un componente llamado **alpha** , que contiene información sobre la opacidad de texel (píxel de textura) a pintar.
- **0.0** significa totalmente transparente, **1.0** totalmente opaco.
- Con la configuración actual del motor, se ignora el canal alpha , así que independientemente de este valor, se pintan los píxeles totalmente opacos (a este modo de pintado se le suele llamar sólido o SOLID).
- Es posible definir modos de pintado que tienen en cuenta el valor de este canal alpha.

# Mezclado de colores

Vamos a hablar de *blending* o mezclado de colores.

El ***blending*** es una técnica que consiste en mezclar los colores del píxel que vamos a pintar con los colores del píxel existente en el backbuffer en esas coordenadas.

El píxel a pintar tiene valores **RGBA**.

El backbuffer únicamente tiene valores **RGB**(en la mayoría de los casos).

Según la forma en que realicemos la mezcla de estos colores, obtendremos distintos tipos de pintado.

# Mezclado de colores

**BLEND_SOLID (sólido):** Los píxeles a pintar reemplazan los píxeles del backbuffer ignorando el valor alpha.

color = rgb_origen

**BLEND_ALPHA (alpha):** Los píxeles a pintar se mezclan con los del backbuffer considerando el valor alpha de origen.

color = rgb_origen*alpha + rgb_destino*(1-alpha)

**BLEND_MUL (multiplicativo):** El color a pintar se multiplica por el color del backbuffer.

color = rgb_origen * rgb_destino

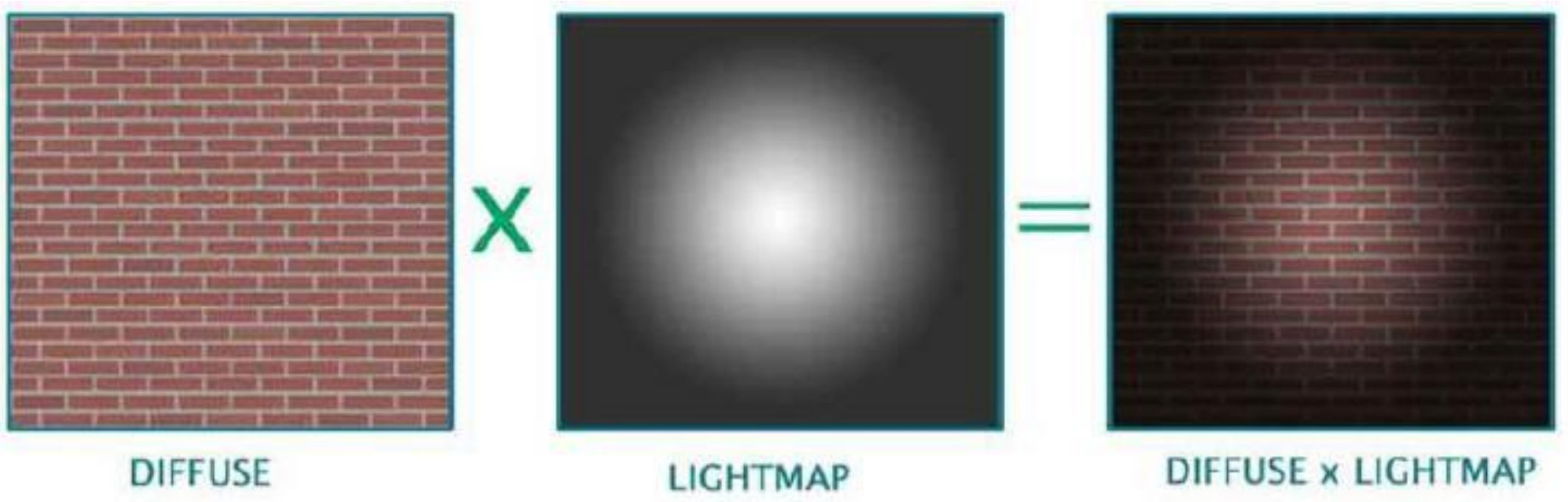
**BLEND_ADD (aditivo):** El color a pintar (normalmente multiplicado por alpha) se suma al color del backbuffer.

color = rgb_origen*alpha + rgb_destino

# Mezclado de colores

# Mezclado de colores

El modo multiplicativo se utiliza para simular iluminación mediante texturas:



# Mezclado de colores

El modo multiplicativo se utiliza para simular iluminación mediante texturas:

DIFFUSE

LIGHTMAP

DIFFUSE x LIGHTMAP

# Mezclado de colores (OpenGL)

En OpenGL, el estado GL_BLEND debe estar activado para poder hacer mezclado de colores.

El modo de pintado se establece con la función:

- **void glBlendFunc(GLenum src_factor , GLenum dst_factor)**

Ej (Modo de pintado para Alpha);

glEnable (GL_BLEND);

glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

# Mezclado de colores (OpenGL)

- **void glBlendFunc(GLenum src_factor , GLenum dst_factor)**

Establece el tipo de mezclado de colores a realizar . La fórmula para calcular el color resultante es:

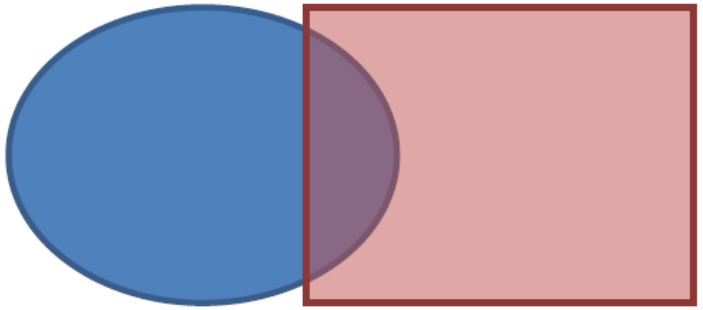
**Color = src _factor*src_color + dst_factor*dst_color**

- **Src _factor:** Cómo afecta el alpha al color de origen
- **Src_color:** El color de salida del fragment shader.
- **Dst_factor**: Cómo afecta el alpha al color buffer.
- **Dst_color:** El color del color buffer .

# Mezclado de colores (OpenGL)

- **void glBlendFunc(GLenum src_factor , GLenum dst_factor)**

Tenemos un círculo azul en el color buffer y queremos pintar un cuadrado rojo semitransparente encima:

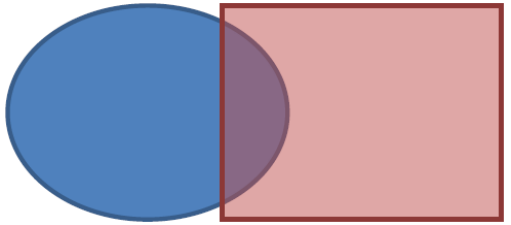


- **RGBA:** Círculo (Dst ) (0, 0, 1, 1) - Cuadrado (Src) (1, 0, 0, 0.5)
- **Color** = (Src Color (1, 0, 0) * Src Factor (0,5)) + (Dst Color (0, 0, 1) * Dst Factor (1-0,5))
- **Color Resultante** = 50% azul y 50% Rojo
- glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

Source Factor

Destination Factor

# Mezclado de colores (OpenGL)



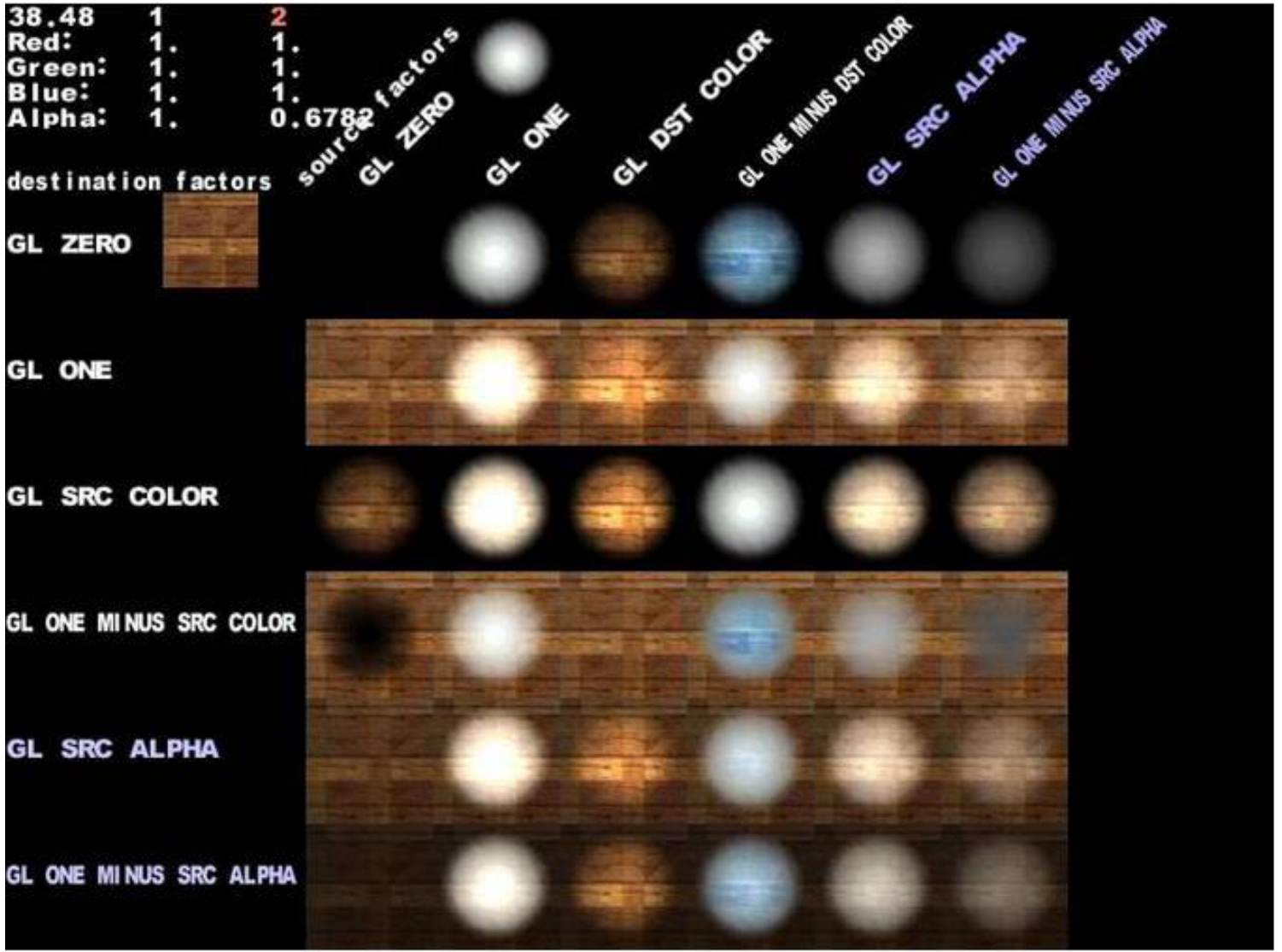
- **void glBlendFunc(GLenum src_factor , GLenum dst_factor)**

Algunos valores para src_factor y dst_factor:

- **GL_ZERO** (0)
- **GL_ONE** (1)
- **GL_SRC_COLOR** (Factor igual al color de origen)
- **GL_DST_COLOR** (Factor igual al color del destino)
- **GL_ONE_MINUS_SRC_COLOR** (Factor igual a 1 - color origen)
- **GL_ONE_MINUS_DST_COLOR** (Factor igual a 1 - color destino)
- **GL_SRC_ALPHA** (Valor de Alpha en origen)
- **GL_ONE_MINUS_SRC_ALPHA** (1 - Valor de alpha en origen)

# Mezclado de colores (OpenGL)

- **void glBlendFunc(GLenum src_factor , GLenum dst_factor)**



# Orden de pintado (usando Blending)

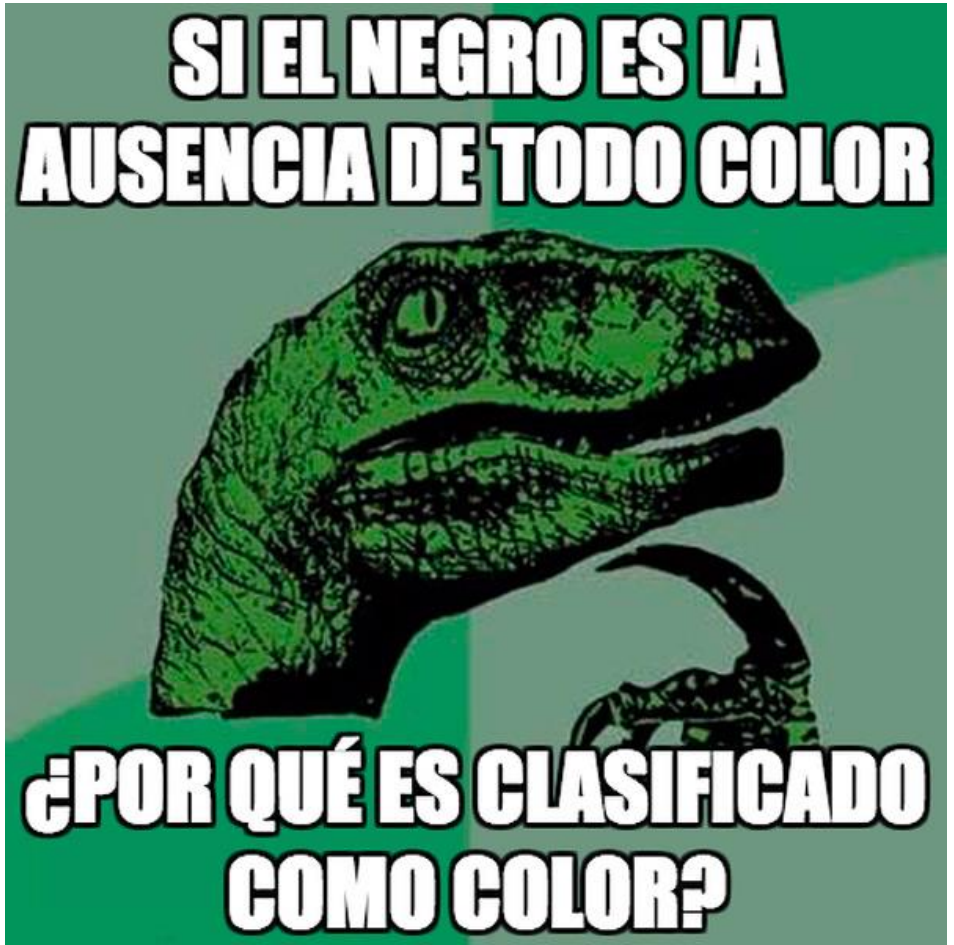
Los elementos transparentes se pintan después de los que tienen elementos sólidos

- **Proceso Básico**
  1. Pintamos todos los objetos opacos usando el depthbuffer
  2. Ordenamos los objetos transparentes por profundidad (distancia a la cámara, por ejemplo)
  3. Dibujamos objetos transparentes en orden

Si se pintan elementos con transparencia, hay que distinguir entre aquellos objetos en los que consideramos que todos sus píxeles son parcial o totalmente transparentes, y los que tienen píxeles totalmente opacos.

- Ventajas:
  - Permite pintar objetos semitransparentes
  - Es bastante eficiente
- Desventajas:
  - No cubre todos los casos.

# ¿Dudas?


SI EL NEGRO ES LA AUSENCIA DE TODO COLOR
¿POR QUÉ ES CLASIFICADO COMO COLOR?