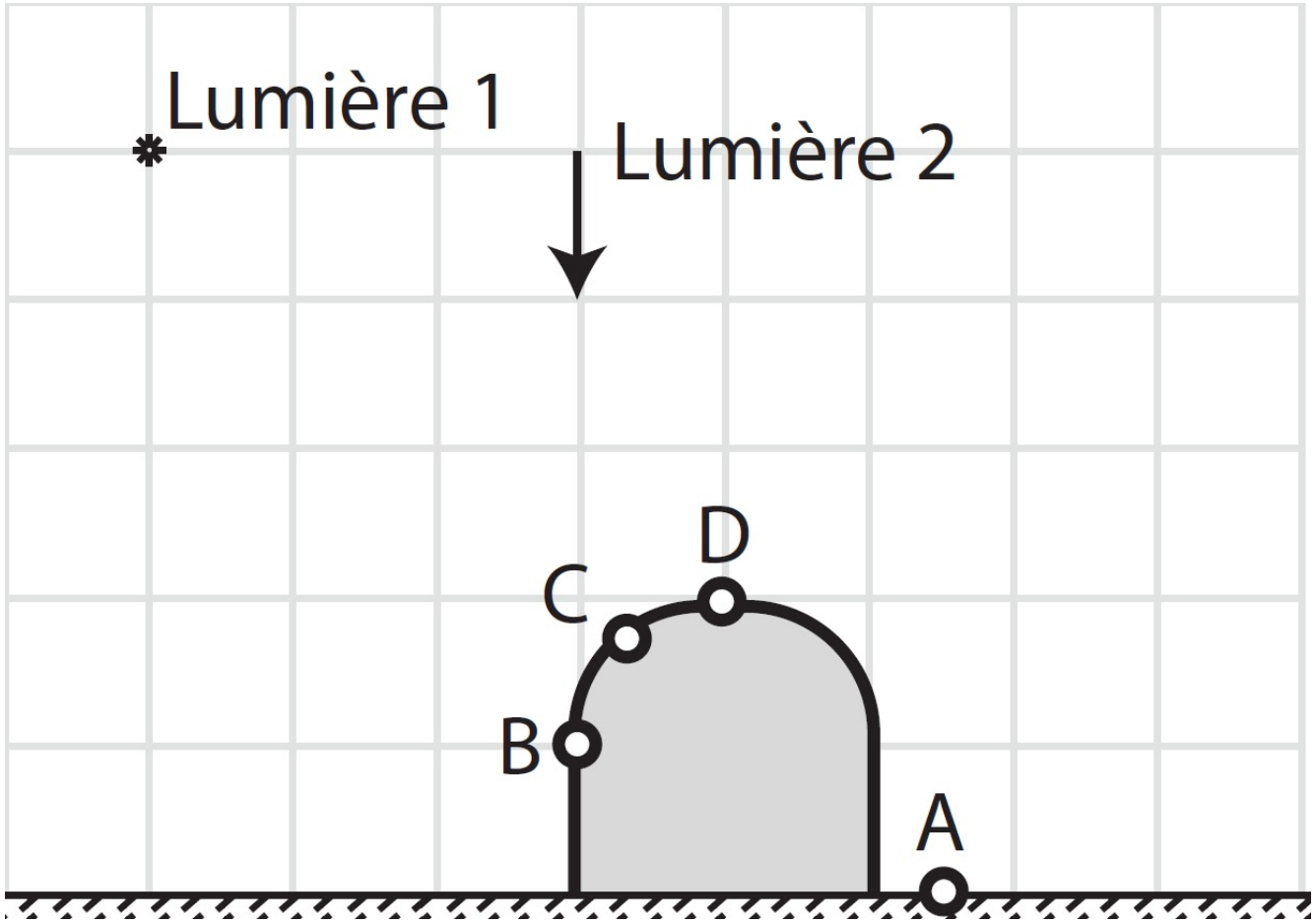
$$I_{L1} = (1, 0, 0.5)$$
$$I_{L2} = (0, 1, 0.5)$$
$$k_d(surface) = (1,1,1)$$
$$k_d(B, C, D) = (0,0,1)$$



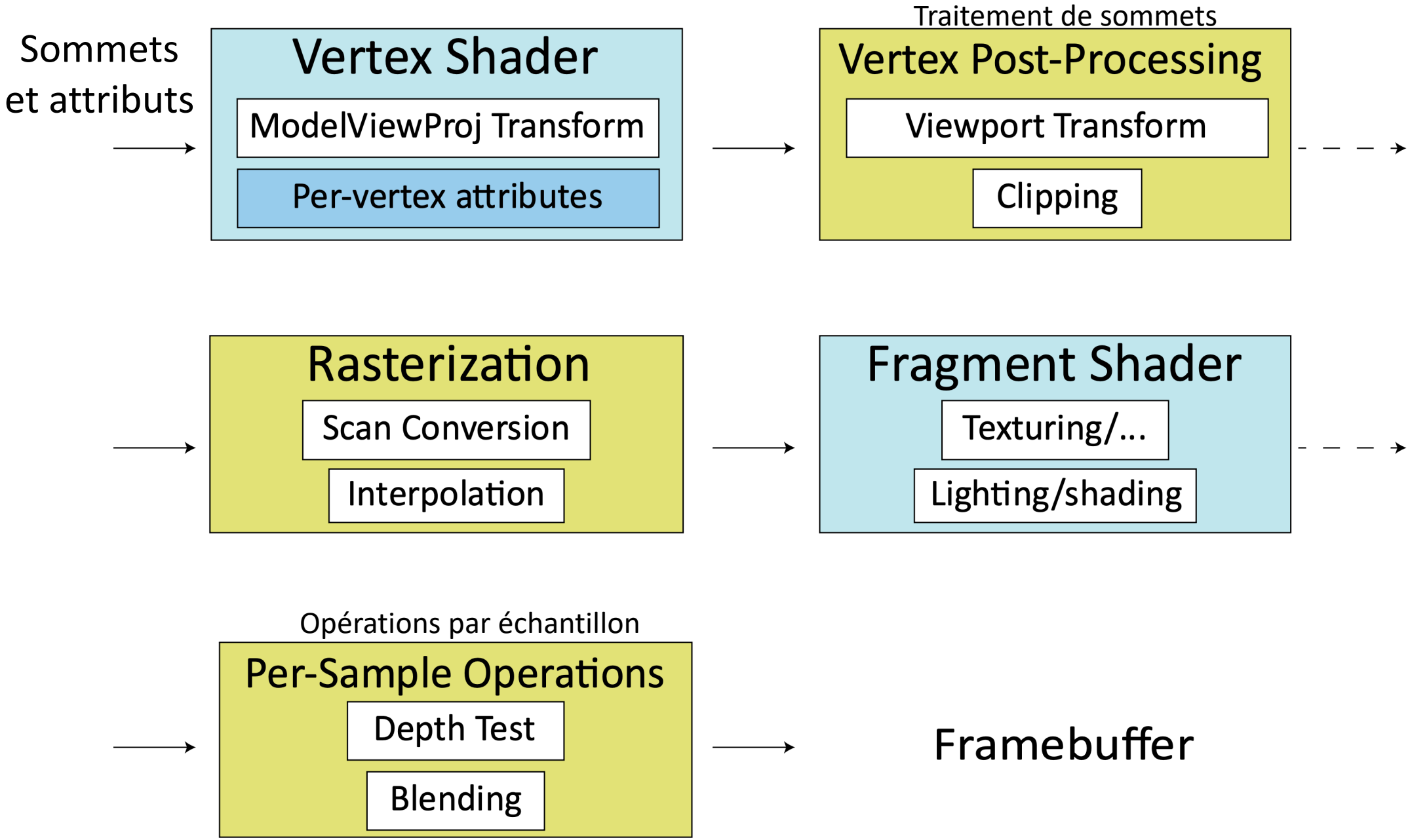
IFT 3355: INFOGRAPHIE
BLENDING
(MÉLANGE)

Livre de référence: G: 16, S: 4

http://tiny.cc/ift3355

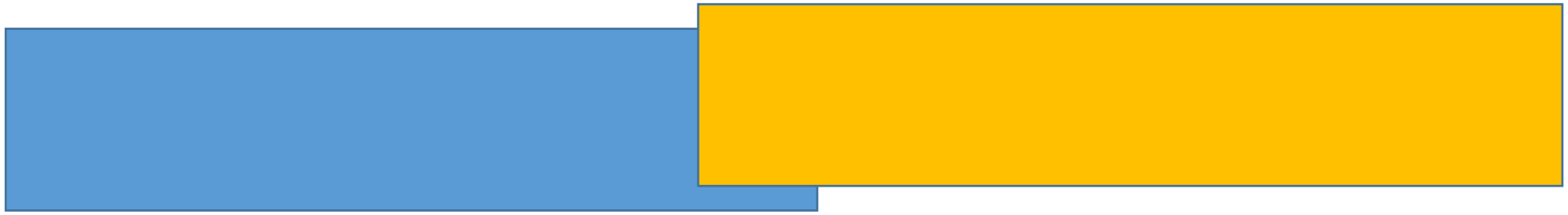
Mikhail Bessmeltsev

PIPELINE: PLUS DE DÉTAILS



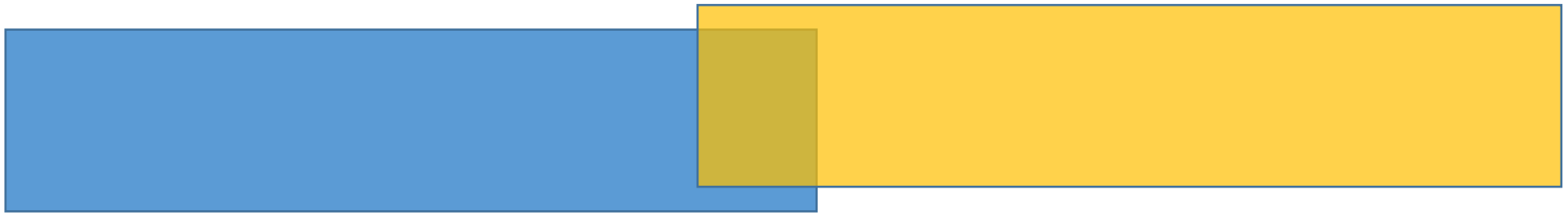
OPAQUE VS. TRANSPARENT

- Si tous les objets sont opaques ou complètement transparents, il est inutile d'utiliser *blending*
- Pour les objets opaques: comme avant, tampon de profondeur
- Dans ce cas, on peut utiliser le test de profondeur **avant** Fragment Shader! ("*Early depth test*")
  - *(si, bien sûr, le fragment shader ne modifie pas le z)*



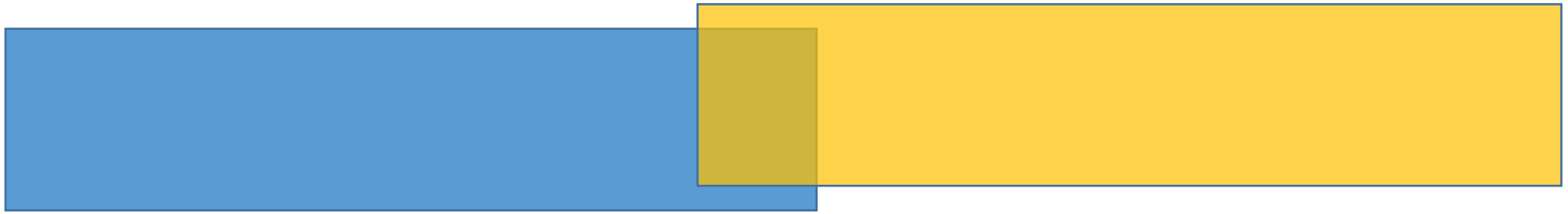
OPAQUE VS. TRANSPARENT

- Pour les objets transparents, chaque fois que l'on écrit dans le *framebuffer*, il faut considérer ce qui est déjà là



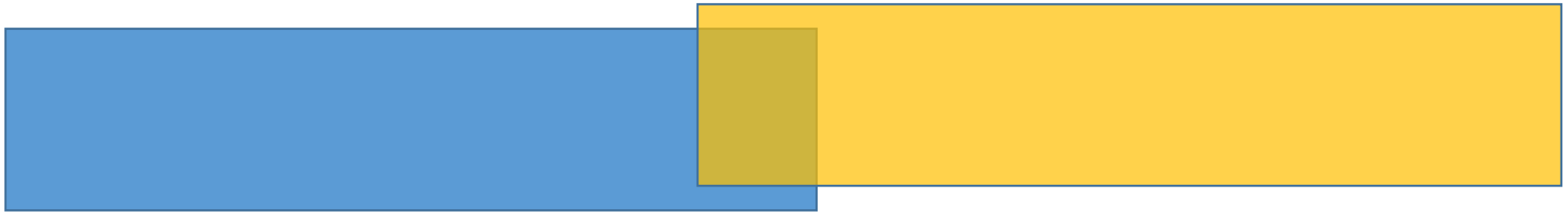
OPAQUE VS. TRANSPARENT

- Pour les objets transparents, chaque fois que l'on écrit dans le *framebuffer*, il faut considérer ce qui est déjà là
- Par fragment:
  - La couleur du fragment: la couleur de **source**
  - Ce que contient le *framebuffer*: la couleur de **destination**



OPAQUE VS. TRANSPARENT

- Pour les objets transparents, chaque fois que l'on écrit dans le *framebuffer*, il faut considérer ce qui est déjà là
- Par fragment:
  - La couleur du fragment: la couleur de **source**
  - Ce que contient le *framebuffer*: la couleur de **destination**
- La même idée que les calques dans Photoshop/Illustrator



Comment combiner les deux couleurs en une seule?

BLENDING: IL Y A PLUSIEURS FAÇONS

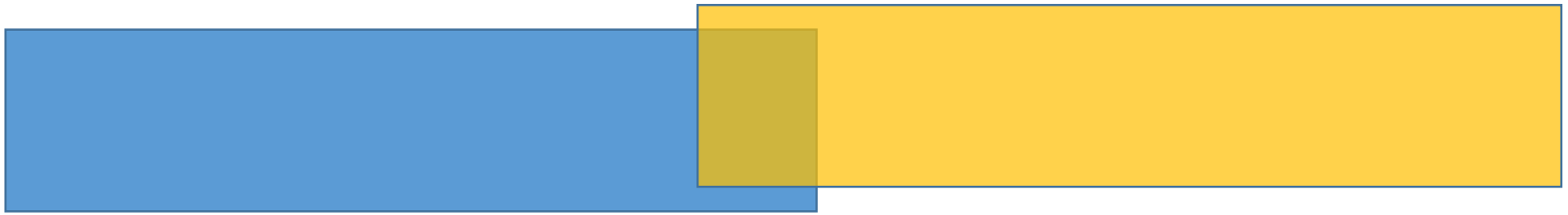
- (démo)
- http://threejs.org/examples/webgl_materials_blending.html

LES ÉQUATIONS DU MÉLANGE

- $D = (r, g, b, \alpha)_D$ - la couleur de destination (ce que contient le *framebuffer*)
- $S = (r, g, b, \alpha)_S$- la couleur source (le fragment actuel)
- $Out = (r, g, b, \alpha)_{Out}$– la couleur de sortie (le mélange)

Les équations du mélange:

$$Out.rgb = f_1(D.rgb, S.rgb)$$
$$Out.\alpha = f_2(D.\alpha, S.\alpha)$$



$\boldsymbol{\alpha = 1.0}$ (opaque)

$\boldsymbol{\alpha = 0.7}$ (transparent)

LES ÉQUATIONS DU MÉLANGE

Les équations du mélange:

$$Out.rgb = f_1(D.rgb, S.rgb)$$
$$Out.\alpha = f_2(D.\alpha, S.\alpha)$$

On peut choisir les fonctions f_1 et f_2. Les options standards sont:

$$f(D,S) = d \cdot D + s \cdot S$$
$$f(D,S) = d \cdot D - s \cdot S$$
$$f(D,S) = s \cdot S - d \cdot D$$
$$f(D,S) = \min(D,S)$$
$$f(D,S) = \max(D,S)$$

d, s – quelques paramètres

D (S) = $D.rgb$ $(S.rgb)$ ou $D.\alpha$ $(S.\alpha)$

LES ÉQUATIONS DE MÉLANGE

On peut choisir les fonctions f_1 et f_2. Les options standards sont

$$f(D,S) = d \cdot D + s \cdot S$$
$$f(D,S) = d \cdot D - s \cdot S$$
$$f(D,S) = s \cdot S - d \cdot D$$
$$f(D,S) = \min(D,S)$$
$$f(D,S) = \max(D,S)$$

d, s – quelque paramètres

$D\ (S)$ = $D.rgb\ (S.rgb)$ ou $D.\alpha\ (S.\alpha)$

Les options standards pour d, s sont:

$$d, s \in \{D.rgb, 1 - D.rgb, S.rgb, 1 - S.rgb, D.\alpha, 1 - D.\alpha, S.\alpha, 1 - S.\alpha$$

constant}

QUELS MÉLANGES PEUT-ON FAIRE?

- La transparence simple (“*over operator*”):
  - $f_1 = ADD, f_2 = ADD$
  - $d_1 = 1 - S.\alpha$
  - $s_1 = S.\alpha$
  - $d_2 = 0$
  - $s_2 = 1$

$$Out.rgb = (1 - S.\alpha) \cdot D.rgb + S.\alpha \cdot S.rgb$$
$$Out.\alpha = 0 \cdot D.\alpha + 1 \cdot S.\alpha$$

$\boldsymbol{\alpha = 1.0}$ (opaque)

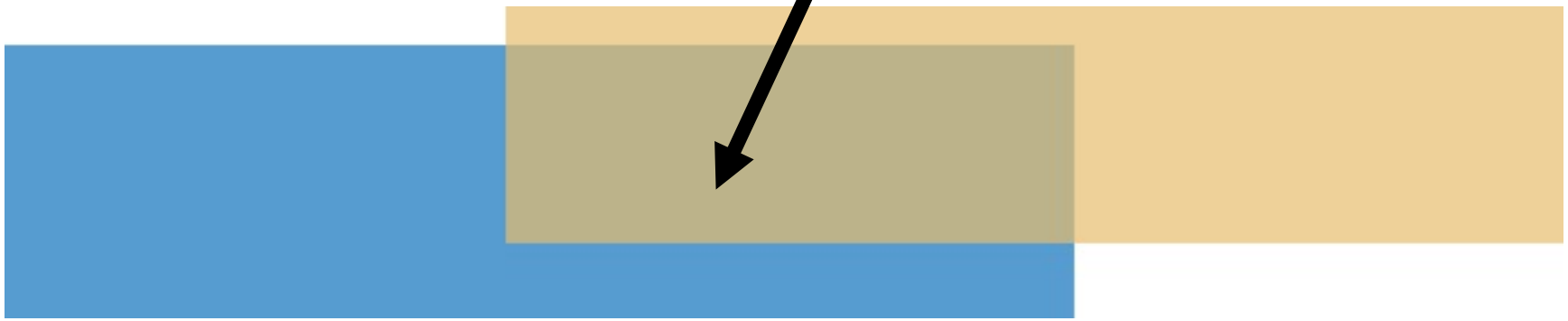
$\boldsymbol{\alpha = 0.7}$ (transparent)

QUELS MÉLANGES PEUT-ON FAIRE?

- La transparence simple ("*over operator*"):
  - $f_1 = ADD, f_2 = ADD$
  - $d_1 = 1 - S.\alpha$
  - $s_1 = S.\alpha$
  - $d_2 = 0$
  - $s_2 = 1$

rgb: $(1 - 0.7) \cdot (\mathbf{0, 0, 1}) + 0.7 \cdot (\mathbf{1, 1, 0})$
= (0,0,0.3) + (0.7, 0.7,0)=**(0.7,0.7,0.3)**

$$Out.rgb = (1 - S.\alpha) \cdot D.rgb + S.\alpha \cdot S.rgb$$
$$Out.\alpha = 0 \cdot D.\alpha + 1 \cdot S.\alpha$$



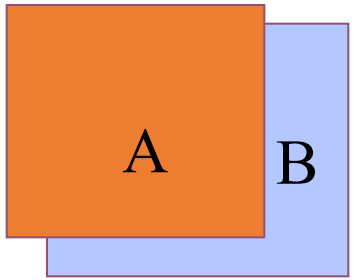
$\boldsymbol{\alpha = 1.0}$ (opaque)

$\boldsymbol{\alpha = 0.7}$ (transparent)

OVER OPERATOR

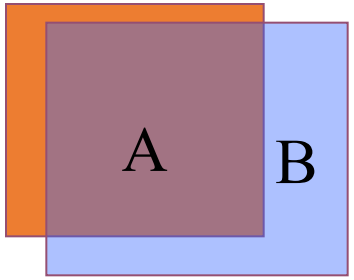
$$Out.rgb = (1 - S.\alpha) \cdot D.rgb + S.\alpha \cdot S.rgb$$

- Exemples: $A.\alpha = 1, B.\alpha = 0.4$



A over B:

$$Out.rgb = (1) \cdot A.rgb + (1 - 1) \cdot B.rgb$$



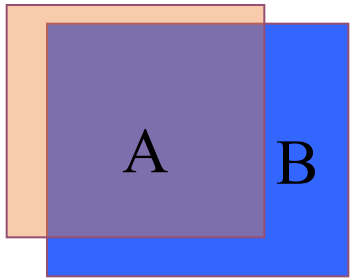
B over A:

$$Out.rgb = (0.4) \cdot A.rgb + (1 - 0.4) \cdot B.rgb$$

OVER OPERATOR

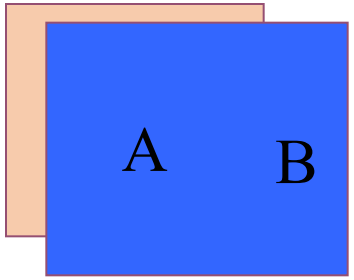
$$Out.rgb = (1 - S.\alpha) \cdot D.rgb + S.\alpha \cdot S.rgb$$

- Exemples: $A.\alpha = 0.4, B.\alpha = 1$



A over B:

$$Out.rgb = (1 - 0.4) \cdot A.rgb + (0.4) \cdot B.rgb$$



B over A:

$$Out.rgb = (0) \cdot A.rgb + (1) \cdot B.rgb$$

QUELS MÉLANGES PEUT-ON FAIRE?

- "Multiplier"
  - $f_1 = ADD, f_2 = ADD$
  - $\boldsymbol{d_1 = S.rgb}$
  - $\boldsymbol{s_1 = 0}$
  - $d_2 = 0$
  - $s_2 = 1$

$$Out.rgb = S.rgb \cdot D.rgb$$
$$Out.\alpha = 0 \cdot D.\alpha + 1 \cdot S.\alpha$$

$\boldsymbol{\alpha = 1.0}$ (opaque)

$\boldsymbol{\alpha = 0.7}$ (transparent)

QUELS MÉLANGES PEUT-ON FAIRE?

- “Assombrir”
  - $f_1 = MIN, f_2 = ADD$
  - $d_2 = 0$
  - $s_2 = 1$

$$Out.rgb = \min(S.rgb, D.rgb)$$
$$Out.\alpha = S.\alpha$$

$\boldsymbol{\alpha = 1.0}$ (opaque)

$\boldsymbol{\alpha = 1.0}$ (opaque)

ALPHA BLENDING EN OPENGL

- *Alpha blending* est une opération dépendante de l'ordre!
  - Il importe quel objet est affiché en premier
  - Et quelle surface se trouve devant

- Pour les scènes 3D, il faut garder une trace de l'ordre de rendu: ***« Transparency Sorting »***
  - Afficher d'abord la surface arrière
  - Les surfaces opaques vont en premier
  - L'algorithme du peintre!
  - **On utilise le test de profondeur**
    - **e.g. pour afficher un objet opaque avant les objets translucides**

Rappel

ALGORITHME DU PEINTRE: LES PROBLÈMES

- Les polygones qui s'intersectent posent un problème
- Même les polygones sans intersections peuvent poser problème aussi:

ALPHA BLENDING EN OPENGL

- *Alpha blending* est une opération dépendante de l'ordre!
  - Il importe quel objet est affiché en premier
  - Et quelle surface se trouve devant

- Pour les scènes 3D, il faut garder une trace de l'ordre de rendu: ***« Transparency Sorting »***
  - Afficher d'abord la surface arrière
  - Les surfaces opaques vont en premier
  - L'algorithme du peintre!
  - **On utilise le test de profondeur**
    - **e.g. pour afficher un objet opaque avant les objets translucides**
  - **Souvent, il est nécessaire de diviser les polygones**

SI $\alpha = 0$ OU 1 SEULEMENT

- *Blending* n'est pas nécessaire
- On peut utiliser le test de profondeur
- Mais… qu'est-ce qui se passe après un fragment avec $\alpha = 0$?

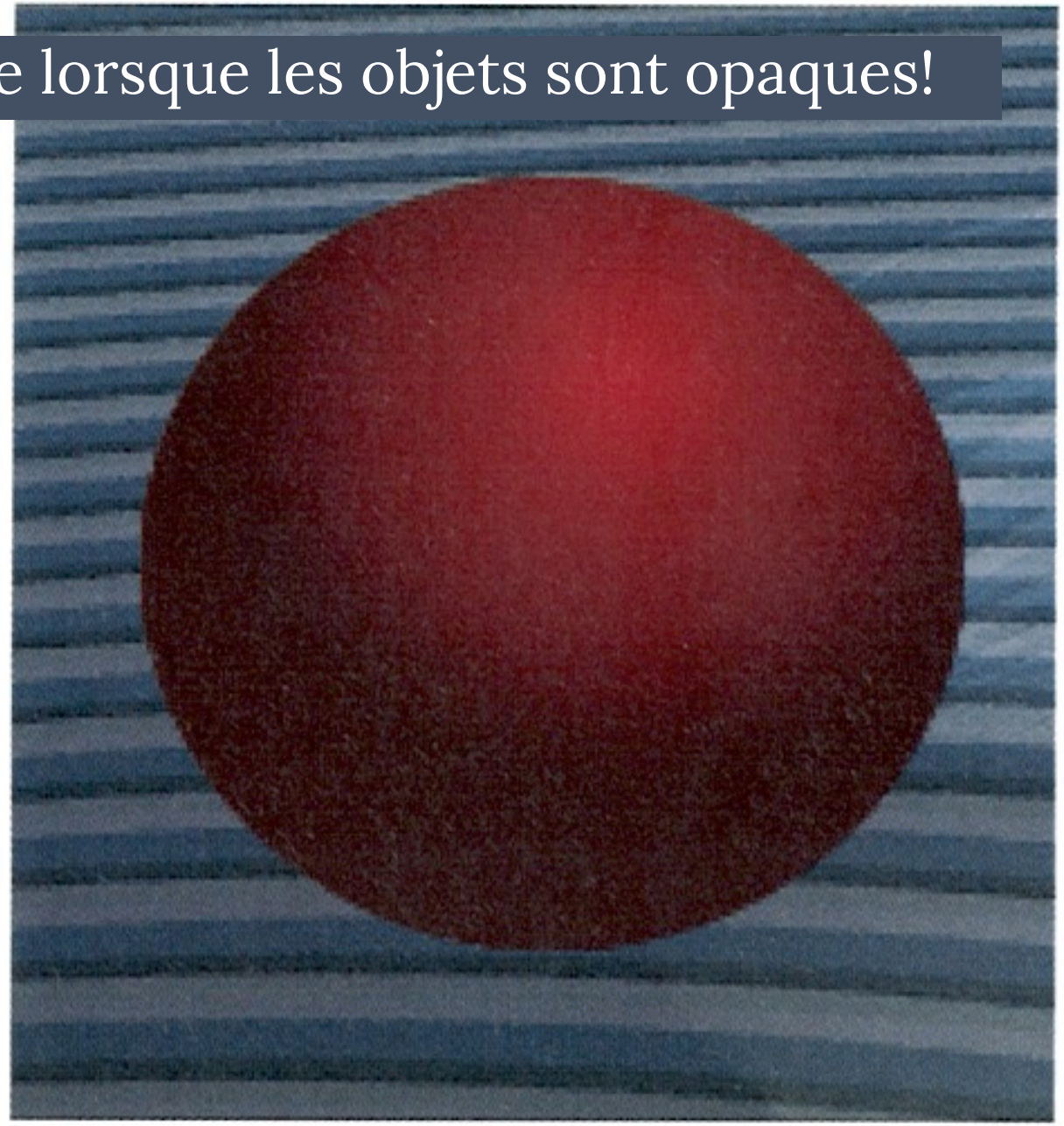
SI $\alpha = 0$ OU 1 SEULEMENT

- *Blending* n'est pas nécessaire
- On peut utiliser le test de profondeur

```
#version 330
in vec2 texCoord;
out vec4 outColor;
uniform sampler2D theTexture;
void main()
{
  vec4 texel = texture(theTexture, texCoord);
  if(texel.a < 0.5)
    discard;
  outColor = texel;
}
```

UN AUTRE EXEMPLE

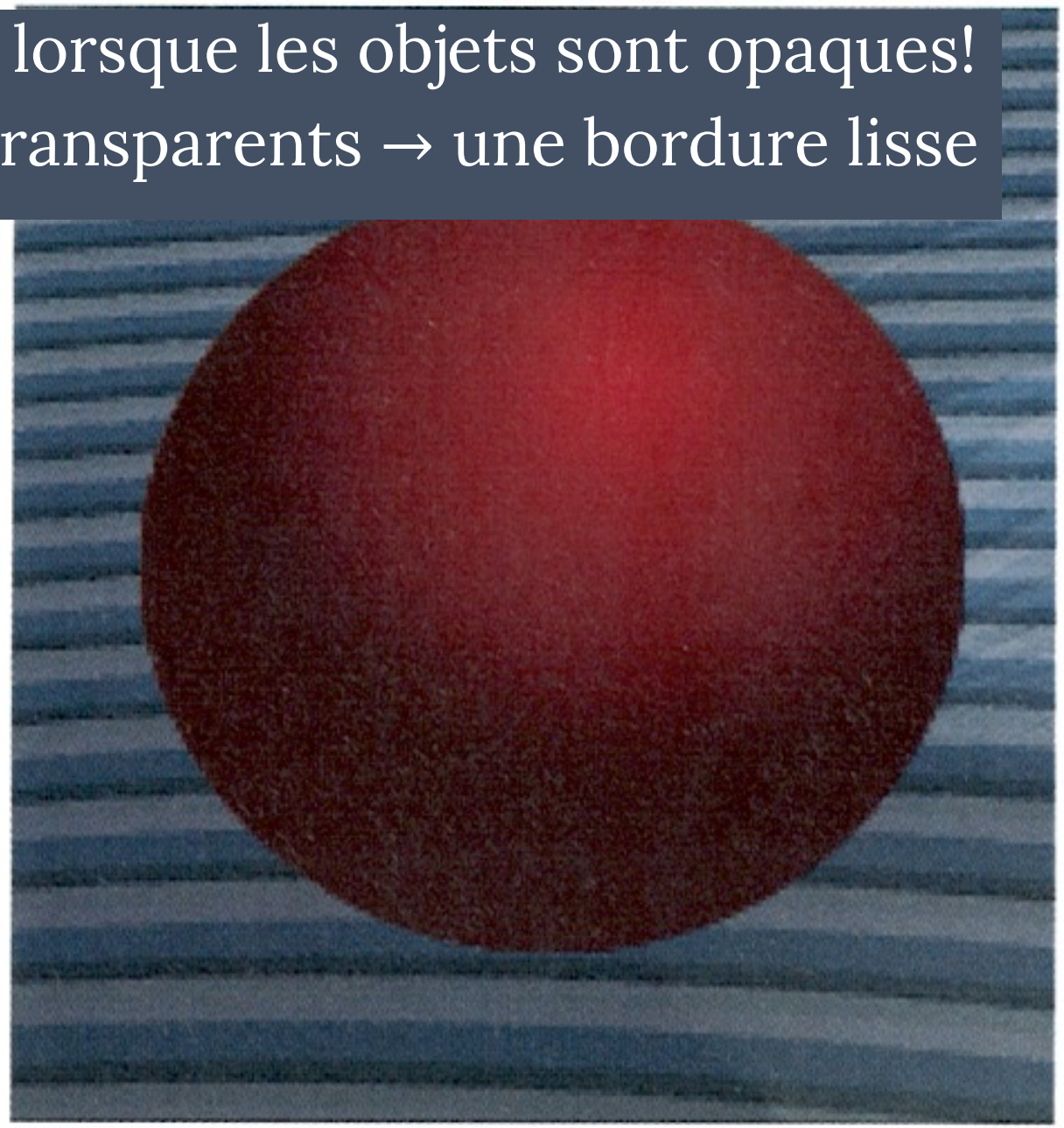
La même idée peut être utilisée lorsque les objets sont opaques!



BLENDING EXAMPLE

La même idée peut être utilisée lorsque les objets sont opaques!

Les pixels sur la frontière sont transparents → une bordure lisse



TRANSPARENCE INDÉPENDANTE DE L'ORDRE

- “Depth Peeling”, “Dual Depth Peeling” et les autres
- http://developer.download.nvidia.com/SDK/10/opengl/src/dual_depth_peeling/doc/DualDepthPeeling.pdf