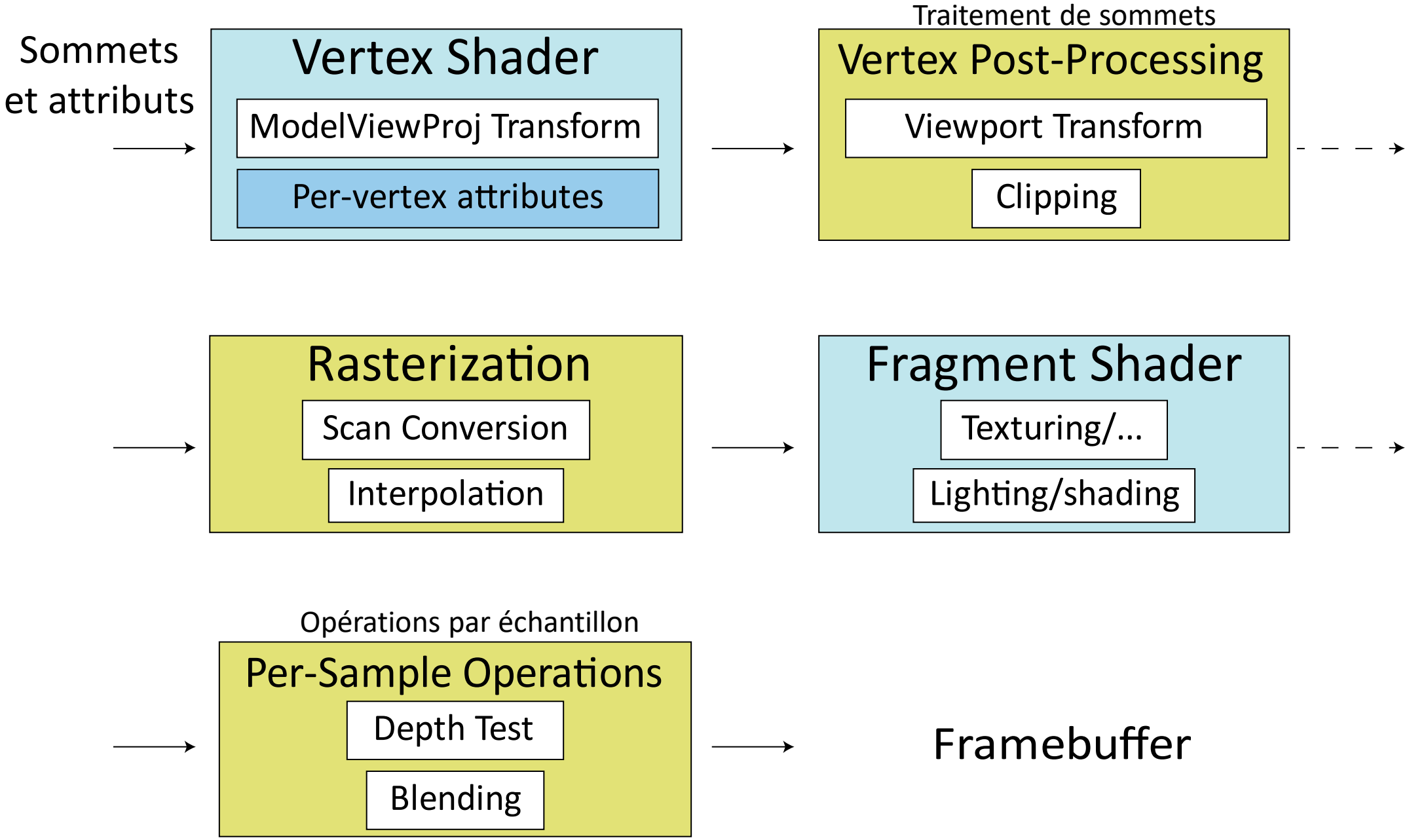
IFT 3355: INFOGRAPHIE

REVUE

FİN

PIPELINE: PLUS DE DÉTAILS



LA MATRICE AUGMENTÉE

Translation

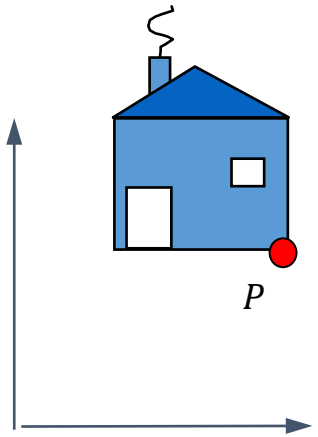
La transformation linéaire

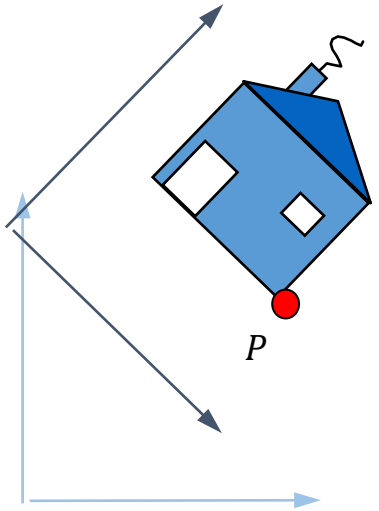
$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & b_x \\ m_{21} & m_{22} & m_{23} & b_y \\ m_{31} & m_{32} & m_{33} & b_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

TRANSFORMATION DE COORDONNÉES

Les colonnes sont les nouveaux vecteurs de base (et la nouvelle origine)!

$$\begin{pmatrix} \cos\theta & -\sin\theta & p_x \cdot (1-\cos\theta) + p_y \cdot \sin\theta \\ \sin\theta & \cos\theta & p_y \cdot (1-\cos\theta) + p_x \cdot \sin\theta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$





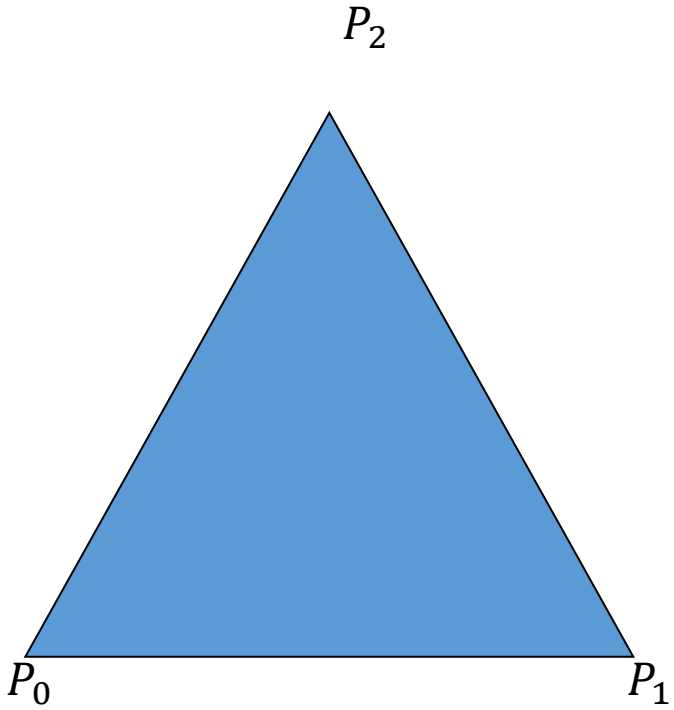
LE TRIANGLE

- La normale

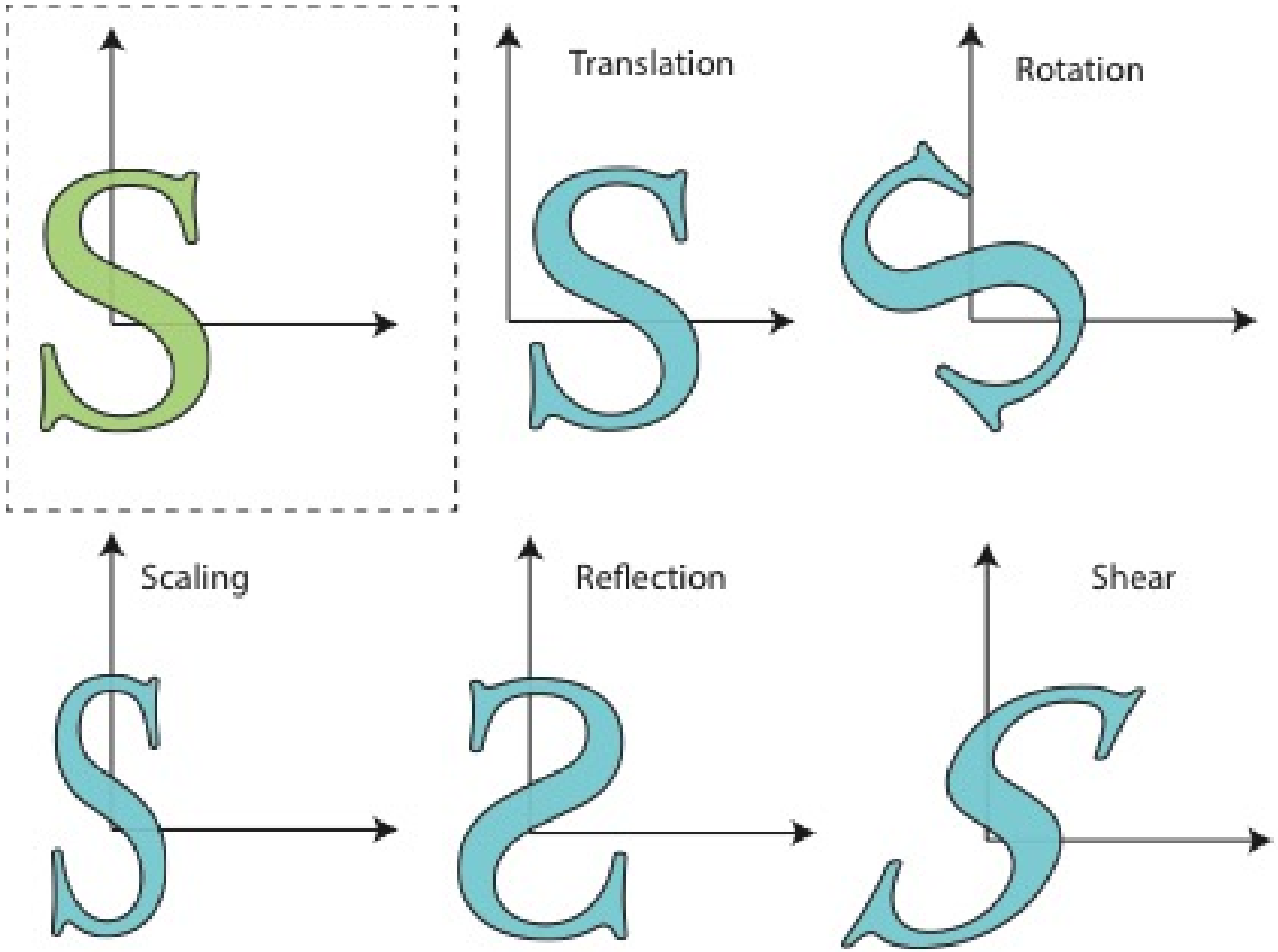
$$n = \frac{(P_1 - P_0) \times (P_2 - P_0)}{\|(P_1 - P_0) \times (P_2 - P_0)\|}$$

- L'aire

$$A = \frac{1}{2} \left\| \overrightarrow{P_0 P_1} \times \overrightarrow{P_0 P_2} \right\|$$



LES TRANSFORMATIONS AFFINES



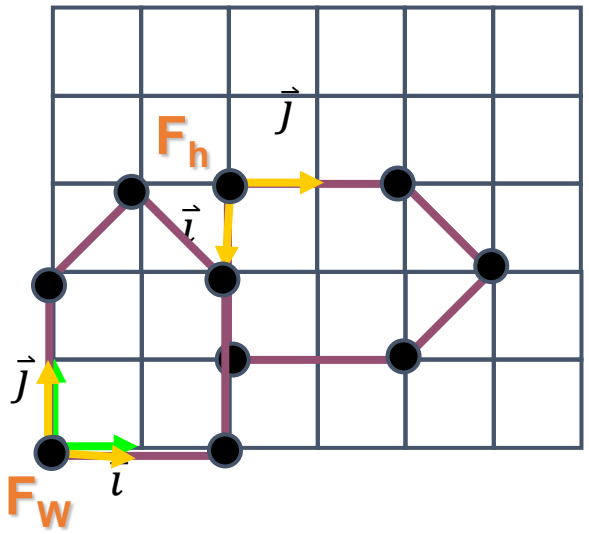
LES COORDONNÉES HOMOGÈNES

Les coordonnées homogènes $\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix}$ Les coordonnées euclidiennes

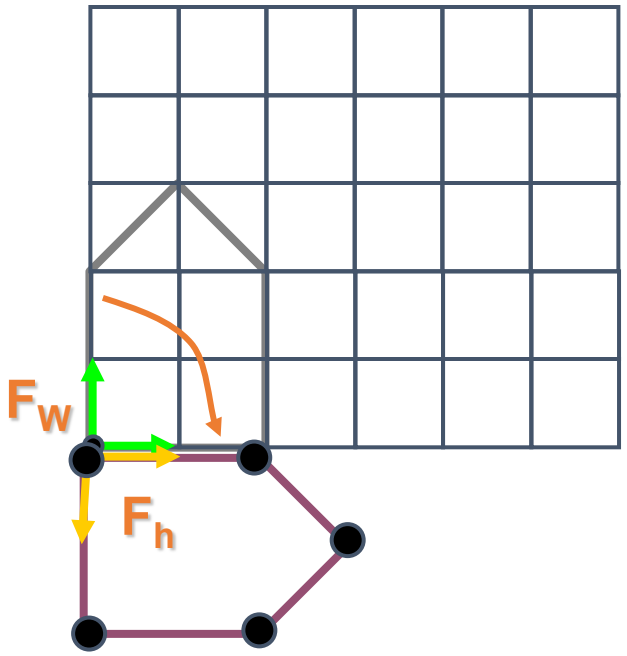
Pas 1-à-1!

LES TRANSFORMATIONS COMPOSÉES

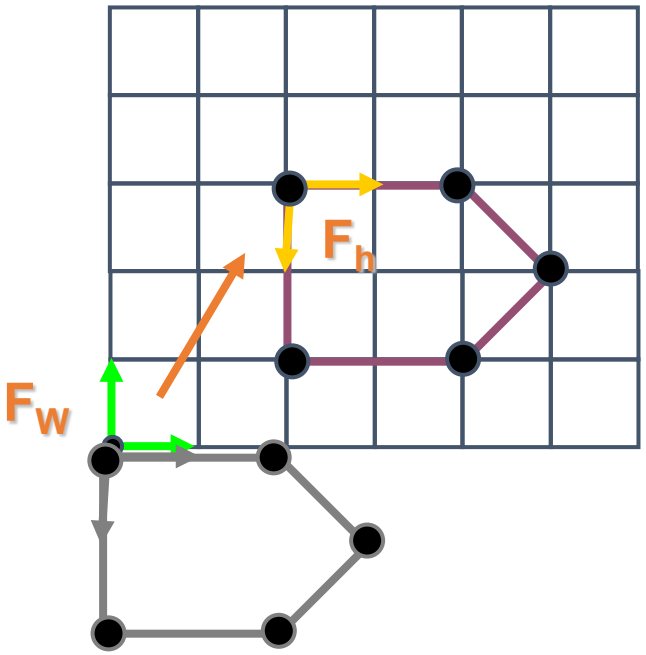
Si on veut



Rotate(z,-90)



Translate(2,3,0)



$$P_A = Rot(z,-90)P_h$$

$$P_W = Trans(2,3,0)P_A$$

$$P_W = Trans(2,3,0)Rot(z,-90)P_h$$

LES TRANSFORMATIONS COMPOSÉES

$$P_W = Trans(2,3,0)Rot(z,-90)P_h$$

- **Lire de droite à gauche** dans le système de coordonnées fixé
  - Déplacer l'objet
- **Lire de gauche à droite**: toutes les opérations sont dans le système de coordonnées local
  - Changer le système des coordonnées

$$M_{MV} = Trans(2,3,0) \cdot M_{MV}$$
$$M_{MV} = Rot(z,-90)M_{MV}$$

LES COMPOSITIONS SIMPLES

$$Tr(x_1, y_1, z_1) \cdot Tr(x_2, y_2, z_2) = Tr(x_1 + x_2, y_1 + y_2, z_1 + z_2)$$
$$Tr(x_2, y_2, z_2) \cdot Tr(x_1, y_1, z_1) = Tr(x_2, y_2, z_2) \cdot Tr(x_1, y_1, z_1)$$

$$Scale(a, b, c) \cdot Scale(d, e, f) = Scale(ad, be, cf)$$
$$Scale(a, b, c) \cdot Scale(d, e, f) = Scale(d, e, f) \cdot Scale(a, b, c)$$

$$Rot(\alpha, 0,0,1) \cdot Rot(\beta, 0,0,1) = Rot(\alpha + \beta, 0,0,1)$$
$$Rot(\alpha, 0,0,1) \cdot Rot(\beta, 0,0,1) = Rot(\beta, 0,0,1) \cdot Rot(\alpha, 0,0,1)$$

LES COMPOSITIONS PLUS COMPLEXES

$$Tr(x,y,z) \cdot Scale(a,b,c) \neq Scale(a,b,c) \cdot Tr(x,y,z)$$
$$Tr(x,y,z) \cdot Scale(a,b,c) = Scale(a,b,c) \cdot Tr\left(\frac{x}{a},\frac{y}{b},\frac{z}{c}\right)$$

$$Tr(x,y,z) \cdot Rot(\alpha,0,0,1) \neq Rot(\alpha,0,0,1) \cdot Tr(x,y,z)$$
$$Rot(\alpha,0,0,1) \cdot Rot(\beta,0,1,0) \neq Rot(\beta,0,1,0) \cdot Rot(\alpha,0,0,1)$$
$$Scale(a,a,a) \cdot Rot(\beta,0,0,1) = Rot(\beta,0,0,1) \cdot Scale(a,a,a)$$

$$Scale(a,b,c) \cdot Rot(\beta,0,0,1) \neq Rot(\beta,0,0,1) \cdot Scale(a,b,c)$$

LE CHANGEMENT D'ÉCHELLE NON UNIFORME **AVANT** LA ROTATION

- Pas quelque chose qu'on s'attend!
- $M = Scale(a, b, c) \cdot Rot(\beta, 0,0,1) =$

$$\begin{pmatrix} \mathrm{a} \cdot \cos(\beta) & -a \cdot \sin(\beta) & 0 & 0 \\ \mathrm{b} \cdot \sin(\beta) & b \cdot \cos(\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Les vecteurs de la base ne sont pas orthogonaux!
- (démo)

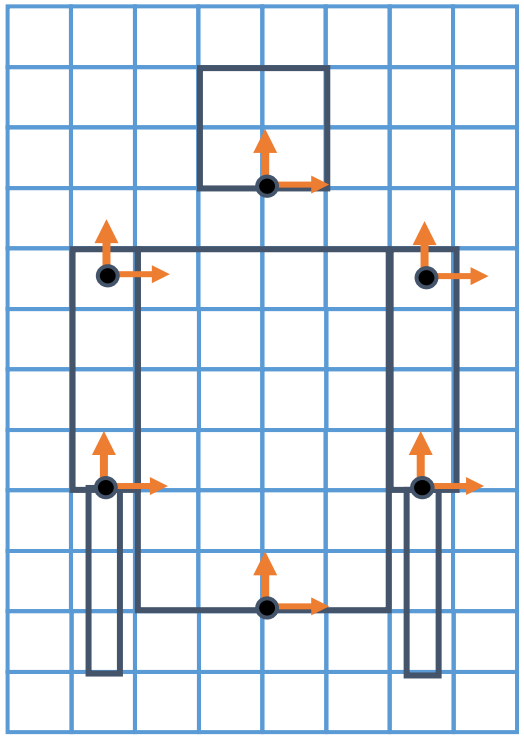
LES TRANSFORMATIONS INVERSES

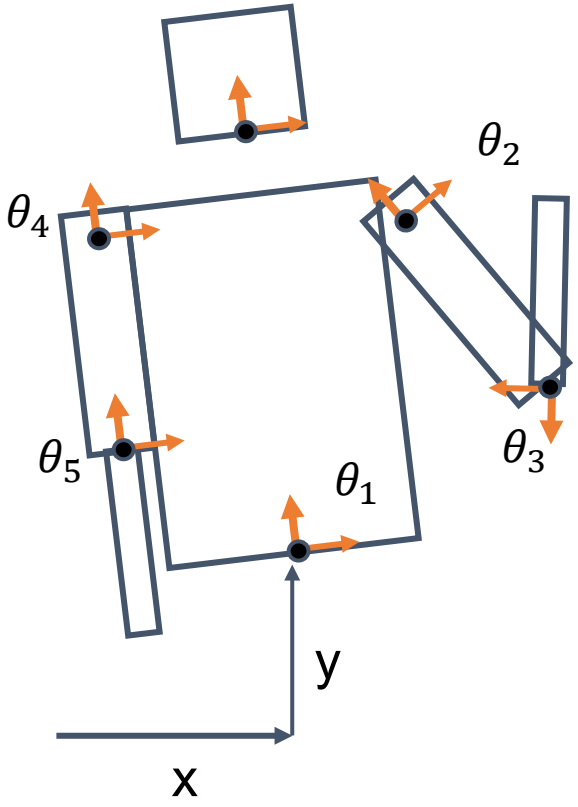
$$Tr(x,y,z)^{-1} = Tr(-x,-y,-z)$$

$$Rot(\alpha,0,0,1)^{-1} = Rot(-\alpha,0,0,1) = Rot(\alpha,0,0,1)^{T}$$ (orthogonal!)

$$Scale(a,b,c)^{-1} = Scale\left(\frac{1}{a},\frac{1}{b},\frac{1}{c}\right)$$

LA HIÉRARCHIE DE TRANSFORMATIONS





$$M_1 = Tr_{(x,y)} \cdot Rot\ \theta_1$$

$$M_2 = M_1 \cdot Tr_{(2.5,5.5)} \cdot Rot\ \theta_2$$

$$M_3 = M_2 \cdot Tr_{(0,-3.5)} \cdot Rot\ \theta_3$$

LE RENDU

- Qu’est ce que le rendu?
- Quels sont les données d’entrée et de sortie?
- Décrire chaque étape de rendu
- Comment faire le rendu en temps réel?
- Comment faire le rendu réaliste?

LES COORDONNÉES HOMOGÈNES

- Pourquoi on en a besoin?
- Comment convertir?
  - 1-1?
- Où dans le pipeline on travaille avec CH/CE?
- Comment distinguer un vecteur d'un point?

LES MATRICES DE TRANSFORMATIONS

- Qu'est-ce qu'une transformation affine? Linéaire?
- Peut-on tous les représenter comme des opérations matricielles?
- Qu'est-ce qu'une structure de matrice de transformation?

PIPELINE

- Quelles sont les transformations dans le pipeline?
- Quels sont les systèmes de coordonnées?
- Pourquoi on utilise la division perspective?
- Pourquoi on fait *clipping* avant de diviser perspective?
- Pourquoi avons-nous besoin d'une transformation de la fenêtre?

LES MATHS

- Quelles sont les équations implicites, explicites et paramétriques pour définir la géométrie?
  - Quelles sont leurs limites?

- Comment intersecter deux objets s'ils sont
  - définis implicitement
  - définis explicitement

- De combien de paramètres avons-nous besoin pour représenter des objets de façon paramétrique?

LES MATHS

- Comment calculer une normale à une surface / courbe implicite?
- Comment calculer un plan tangent?
- Comment approximer la surface d'une forme 2D?
- Comment intersecter un rayon avec un polygone en 2D? En 3D?

PROJECTIONS

- Quel est le but des projections?
- Quelle est la différence entre les projections ortho- et perspective?
- Qui choisit la projection à utiliser?
- Peut-on obtenir une projection presque orthographique tout en utilisant une matrice de projection perspective?
- Que se passe-t-il avec
  - z dans la projection perspective?
  - volume de vue?

CLIPPING

- Que passe-t-il avec points pendant *clipping*? Triangles?
- Quelles sont les équations des plans du volume de vue?
- Comment tester si un triangle doit être decoupé?

RASTERIZATION

- Qu'est-ce que la rasterization?
- Comment rasterizer un polygone?
- Pourquoi on interpole?
- Quelles sont les valeurs que nous interpolons généralement?
- Comment?
- Comment se fait-il dans le tracer de rayons / chemins?

ÉCLAIRAGE

- Qu'est-ce qu'un ombrage Gouraud?
- Que sont les matériaux Lambert / Phong?
- Si la scène est éclairée uniquement avec de la lumière ambiante, que verrons-nous?
  - Seulement diffus / spéculaire?
- Comment contrôler la taille de la surbrillance spéculaire?
- Comment faire *shading* dans le lancer de rayons? dans le tracer des chemins?
- En tracer des chemins, comment simuler des matériaux plus complexes?

LES TEXTURES

- Si une texture contient une seule brique, quelles devraient être les coordonnées de texture pour les coins du mur?
- Pourquoi on utilise des mipmaps?
- De combien de mémoire avons-nous besoin pour eux?
- Comment générer des mipmaps?
- Où obtient-on les coordonnées de texture?
- Comment les interpolons-nous?

BUMP AND NORMAL MAPPING

- Why?
- Which mapping would you use to add scales to a fish?
- Bullets on the walls?
- Fur on an animal?
- How do we apply bump mapping?

ENVIRONMENT MAPS

- Why do we need them?
- What are the types?
- How do we generate them?
- How do we apply them?
- When do we re-generate them?

SHADOW MAPS

- Why do we need them?
- How does it fit into pipeline?
- What's the algorithm?

LES MODÈLES ET ALGORITHMES D'ÉCLAIRAGE

L'éclairage local - rapide

- Plus loin de la physique
- Approximer l'apparence
- L'interaction de chaque objet avec la lumière directe

L'éclairage global – lent

- Plus proche de la physique
- Les interactions entre les objets

LANCER DE RAYONS: L'ALGORITHME DE BASE

```
RayTrace(r,scene)
obj = FirstIntersection(r,scene)

if (no obj) return BackgroundColor;
else {
    if (Reflect(obj))
       reflect_color = RayTrace(ReflectRay(r,obj));
    else
       reflect_color = Black;

    if (Transparent(obj))
       refract_color = RayTrace(RefractRay(r,obj));
    else
       refract_color = Black;

   return Shade(reflect_color, refract_color, obj);
}
```

QUAND ARRÊTER?

- Cet algorithme ne se termine pas

- Les critères de terminaison
  - Aucune intersection
  - La contribution des rayons secondaires est inférieure à un seuil
  - La profondeur maximale / nombre de générations est atteinte

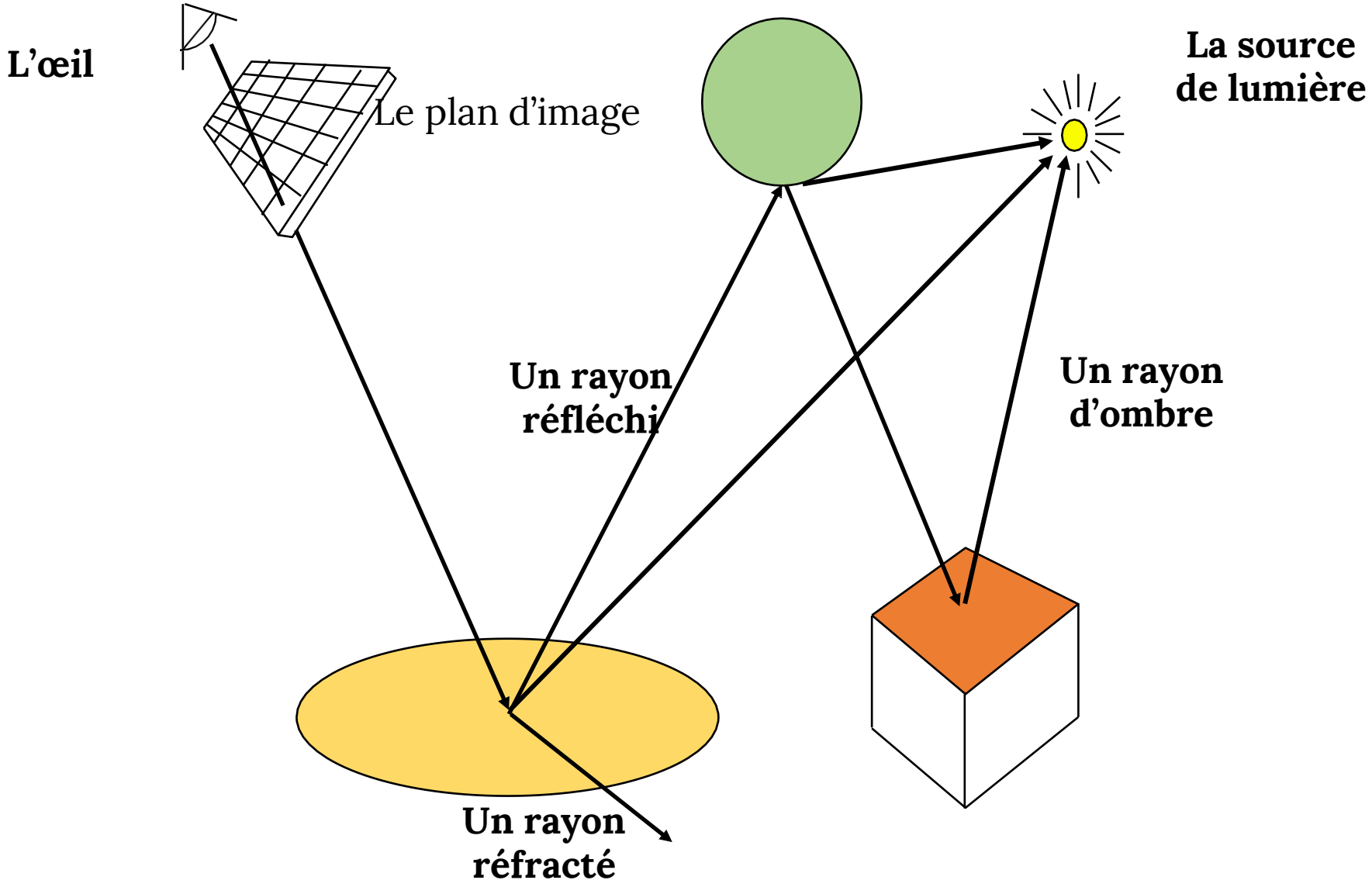
LES OMBRES

- Tracer un rayon de l'objet à la source de lumière
  - Si le rayon intersecte quelque chose ⇒ le point est dans l'ombre

```
shadow = RayTrace(LightRay(obj,r,light));


return Shade(shadow,reflect_color,refract_color,obj);
```

L'IDÉE DU LANCER DE RAYONS



LANCER DE RAYONS: L'ÉCLAIRAGE DIRECT

- L'information de la surface locale (la normale…)
  - Pour les surfaces implicites, $F(x,y,z) = 0$:
    la normale $n(x,y,z)$ est le gradient de F:

$$n(x,y,z) = \nabla F(x,y,z) = \begin{pmatrix} \partial F(x,y,z)/\partial x \\ \partial F(x,y,z)/\partial y \\ \partial F(x,y,z)/\partial z \end{pmatrix}$$

  - Exemple:

$$F(x,y,z) = x^2 + y^2 + z^2 - r^2$$

$$\mathbf{n}(x,y,z) = \begin{pmatrix} 2x \\ 2y \\ 2z \end{pmatrix}$$

Elle doit être normalisée

LES OMBRES DOUCES: LES LUMIÈRES SURFACIQUES

- Jusqu'ici:
  - Toutes les lumières étaient ponctuelles ou directionnelles
    - Pour le pipeline OpenGL et lancer de rayons
  - Donc, à chaque point, on avait besoin de calculer l'éclairage pour **UNE** direction par lumière
- En réalité:
  - Toutes les lumières ont une aire finie
  - Maintenant on a besoin d'intégrer sur toutes les directions vers la lumière

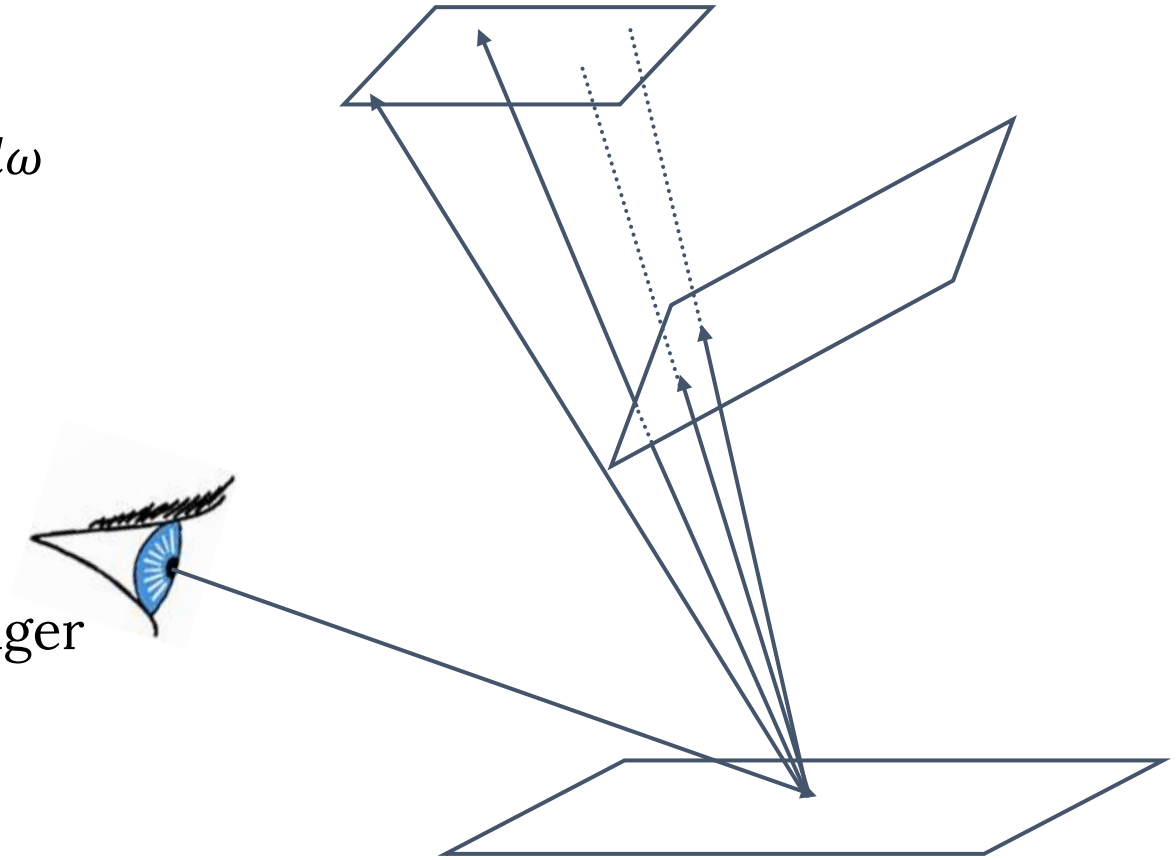
LES LUMIÈRES SURFACIQUES

- Les lumières surfaciques:
  - Nombre infini de rayons lumineux
  - Il faut intégrer:

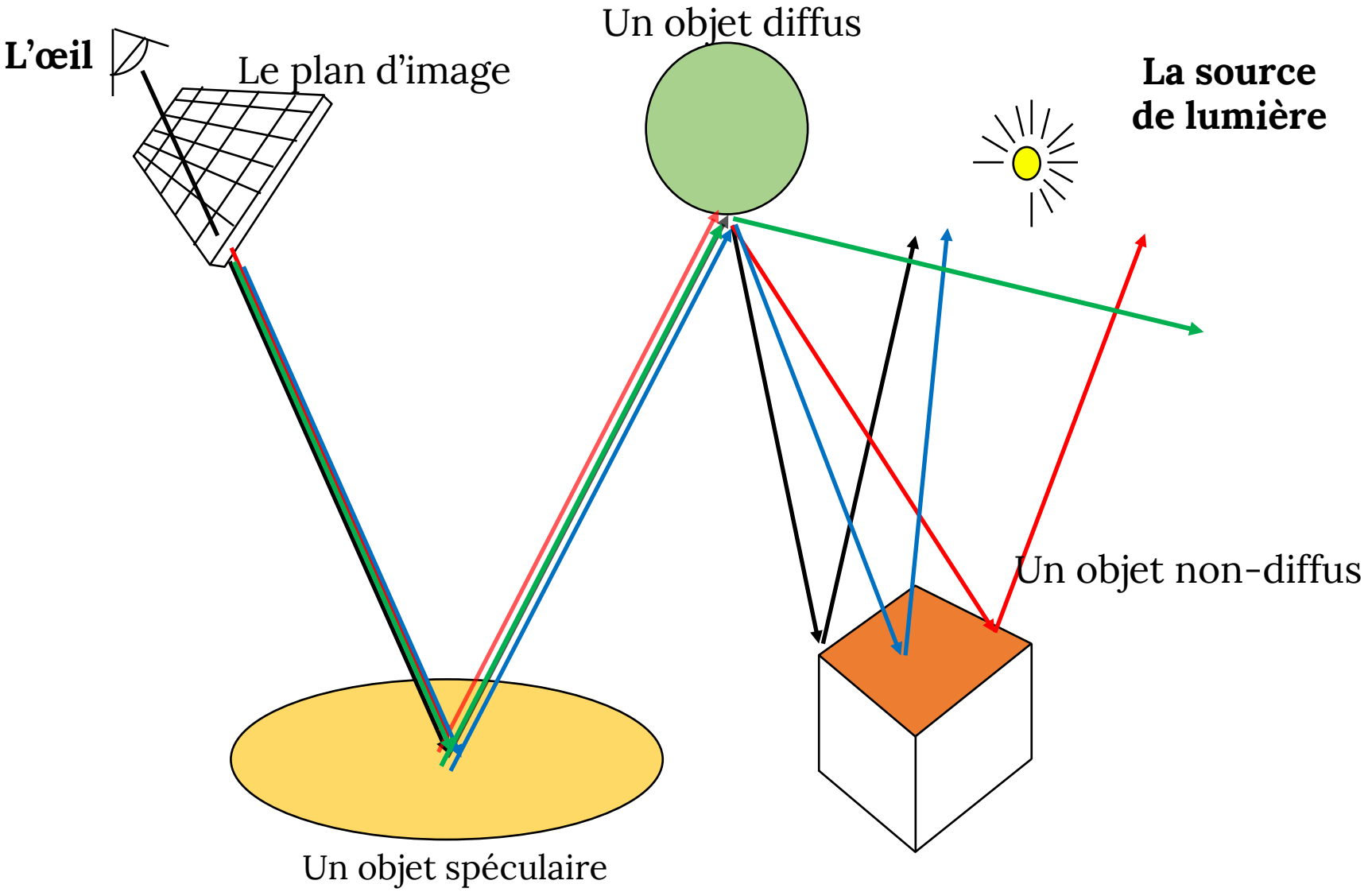
$$I_{reflected} = \int_{\substack{light \\ directions}} \rho(\omega, \boldsymbol{n}) V(\omega) I_{light}(\omega) d\omega$$

- La visibilité et l'intensité peuvent changer pour différents points

Les lumières surfaciques



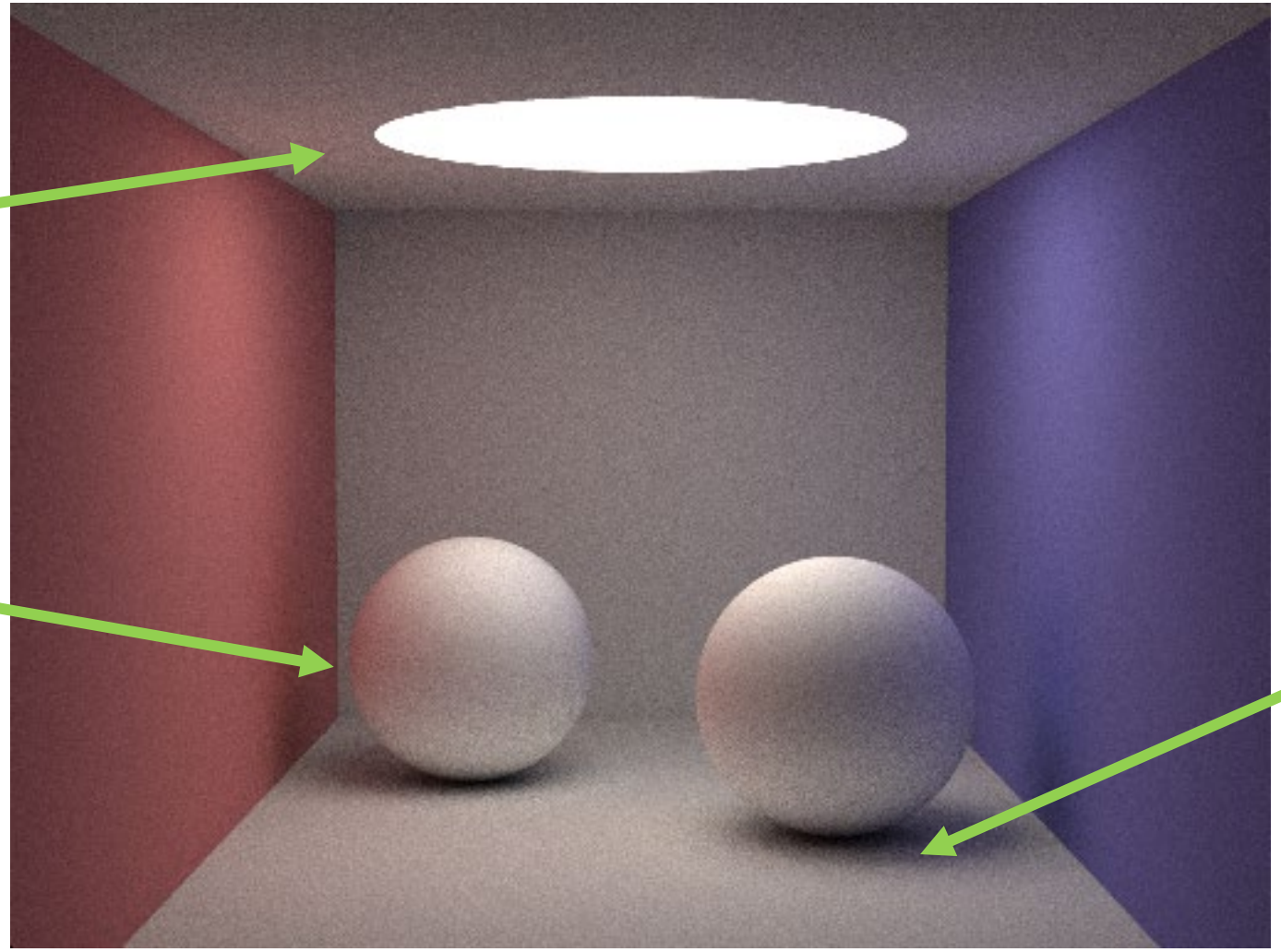
TRACER DE CHEMINS



TRACER DE CHEMINS: LA VERSION LA PLUS SIMPLE

```
PathTrace(Ray r) {
P = closestIntersection(r);
if (random(emit, reflect) == emit)
return EmissionColor;
else {
Ray v = {intersectionPt,
randomDirectionInHemisphere(r.normalWhereObjWasHit)};
double cos_theta = dot(v.direction, r.normalWhereObjWasHit);
return PathTrace(v)*cos_theta*reflectance;
}
}
```

ON OBTIENT

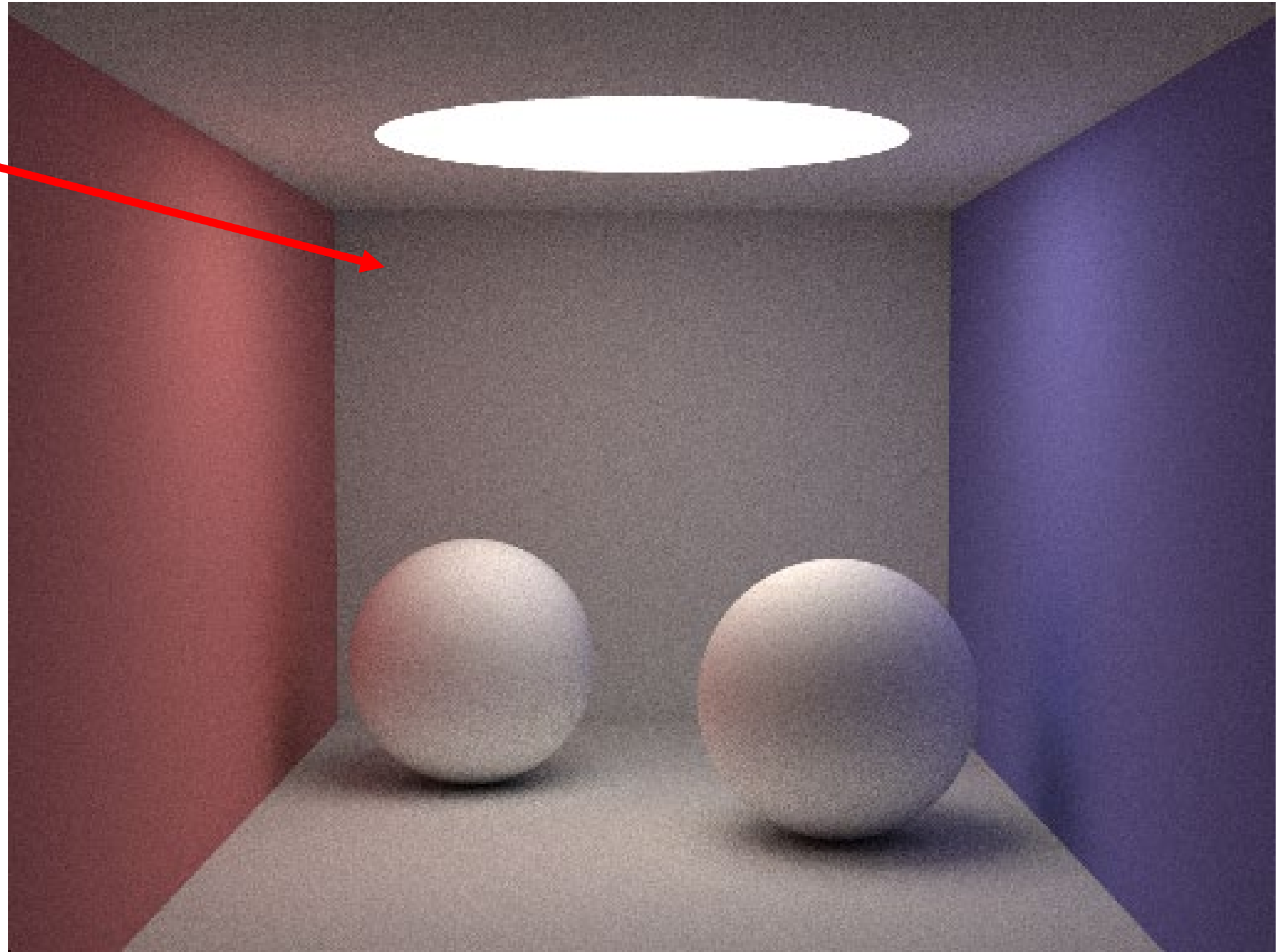


La lumière surfacique

Les réflexions sur les objets diffus ‘*color bleeding*’

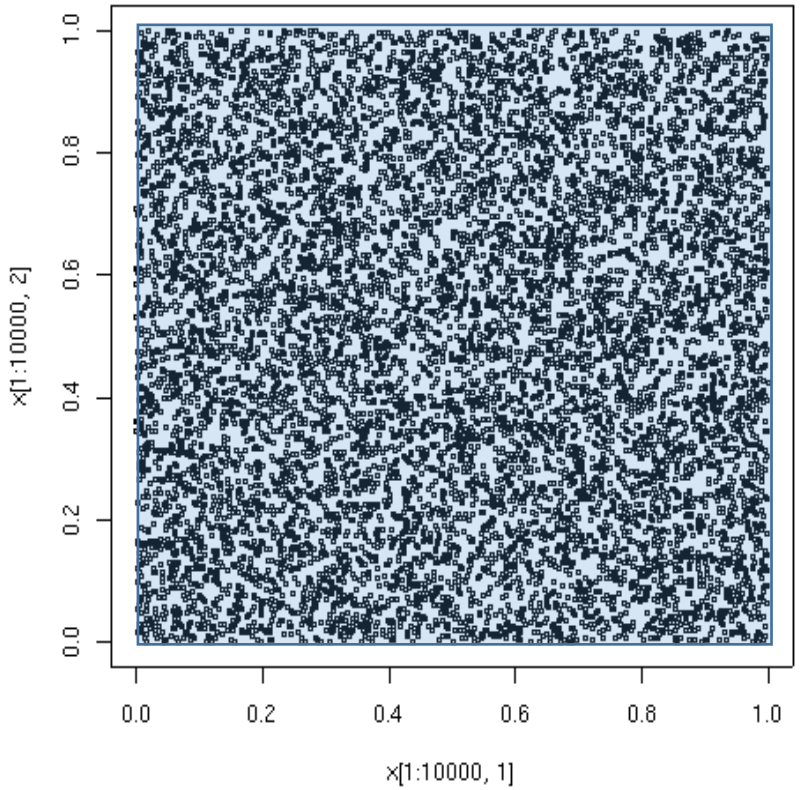
Les ombres douces

Le bruit



GÉNÉRER DES POINTS ALÉATOIRES

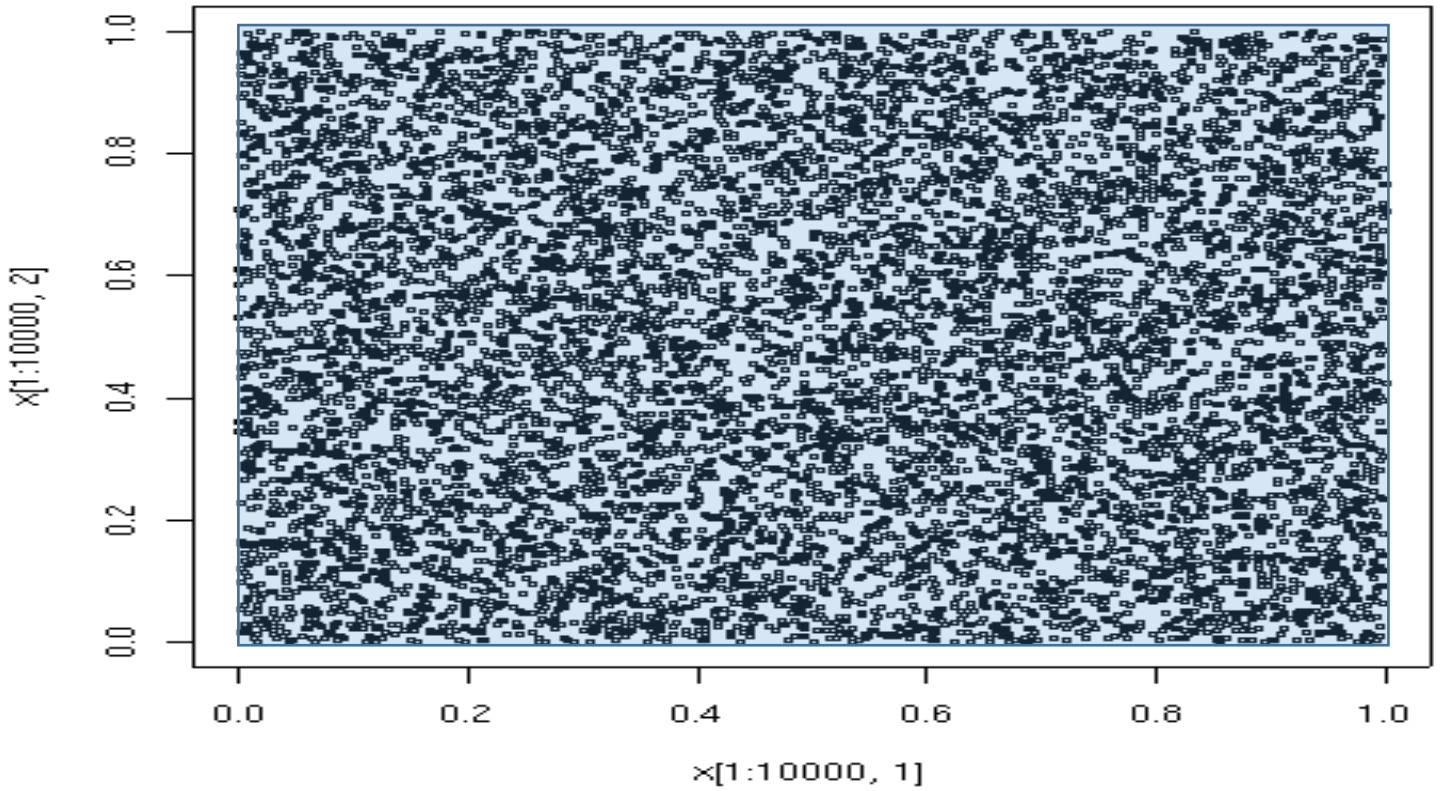
- Comment générer un point à l'intérieur d'un carré?




```
for (i=0…N)
{
    x = rand();
    y = rand();
}
```

GÉNÉRER DES POINTS ALÉATOIRES

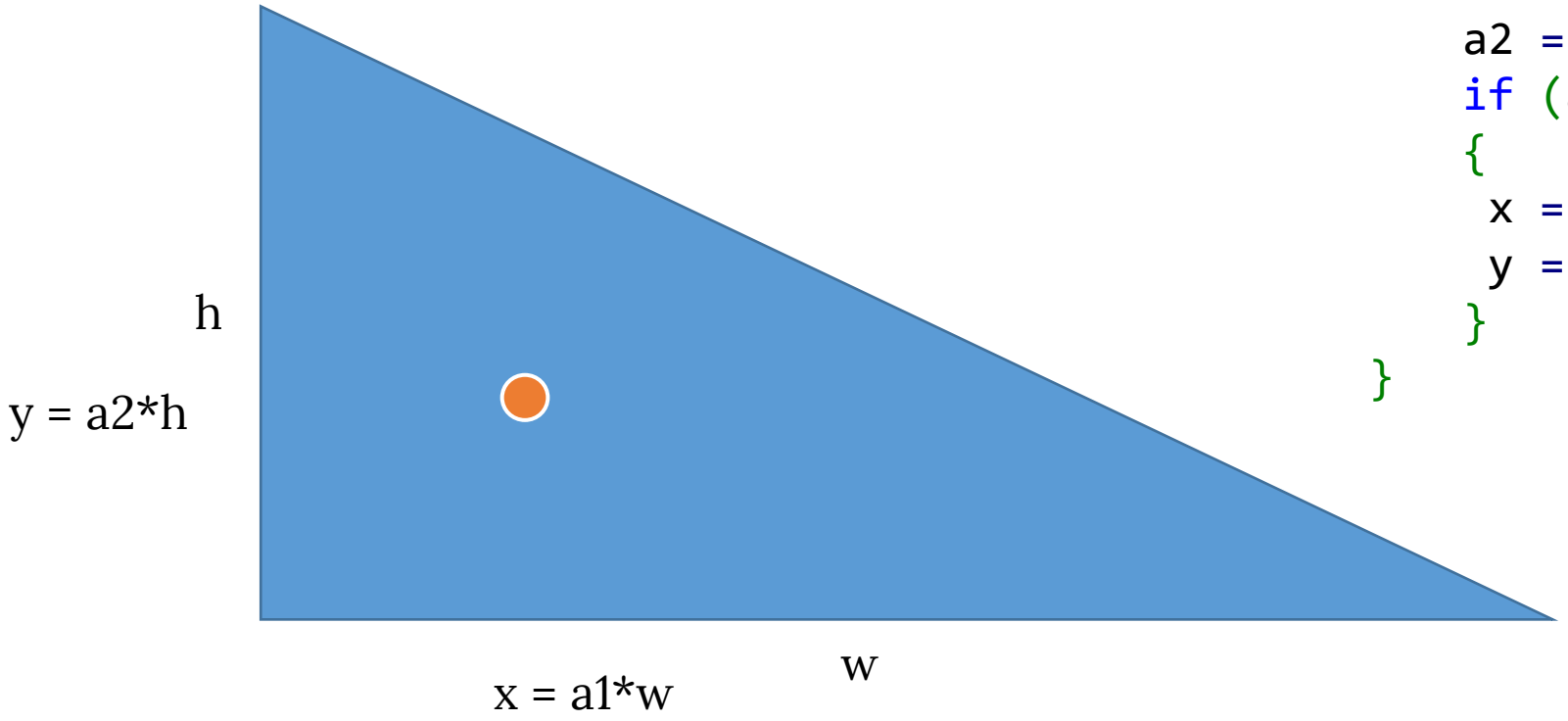
- Comment générer un point à l'intérieur d'un rectangle?




```
for (i=0…N)
{
    x = w*rand();
    y = h*rand();
}
```

GÉNÉRER DES POINTS ALÉATOIRES

- À l'intérieur d'un triangle rectangle?

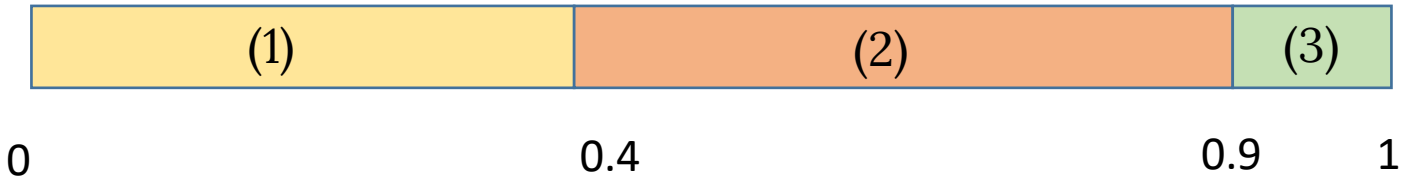



```
for (i=0…N)
{
    a1 = rand();
    a2 = rand();
    if (a1+a2<1)
    {
     x = a1*w;
     y = a2*h;
    }
}
```

COMMENT CHOISIR PARMI 3 ACTIONS

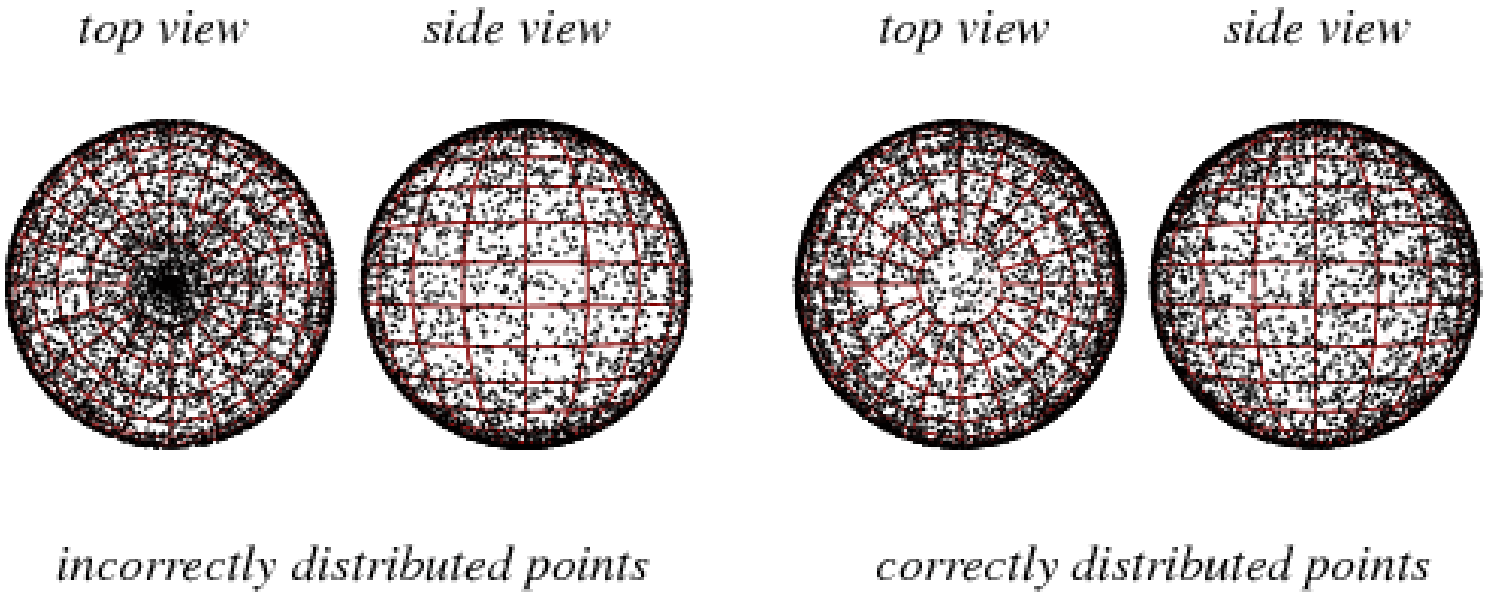
1. “Aller écouter IFT 3355”, $P = 0.4$
2. “Déjeuner”, $P = 0.5$
3. “Dormir tard”, $P = 0.1$

```
x = random();
if (x < 0.4)
 return 1;
else if (x < 0.9)
 return 2;
else
 return 3;
```



COMMENT GÉNÉRER UNE DIRECTION SUR UN HEMISPHÈRE ALÉATOIREMENT?

- Uniformément?
- ~~Option 1:~~
  - ~~$\phi = random(1.0);$~~
  - ~~$\theta = random(1.0);$~~
  - ~~$dir = \begin{pmatrix} \cos(\theta)\sin(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{pmatrix}$~~

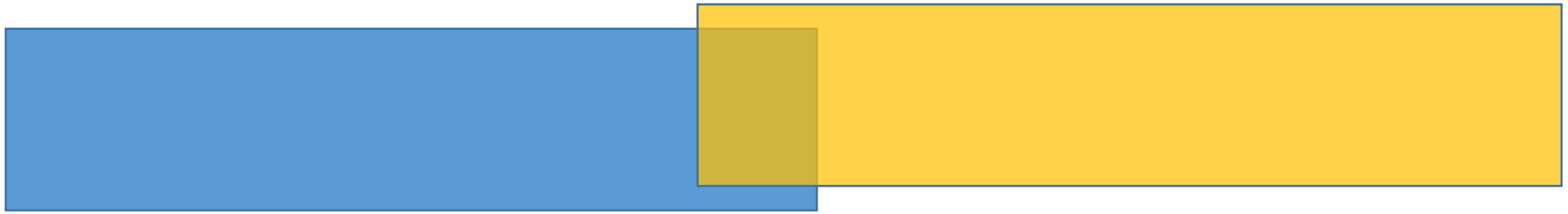


incorrectly distributed points *correctly distributed points*

- Option 2:
  - Générer z uniformément
  - Générer θ uniformément
  - $dir = \begin{pmatrix} \sqrt{1-z^2} \cdot \cos(\theta) \\ \sqrt{1-z^2} \cdot \sin(\theta) \\ z \end{pmatrix}$

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

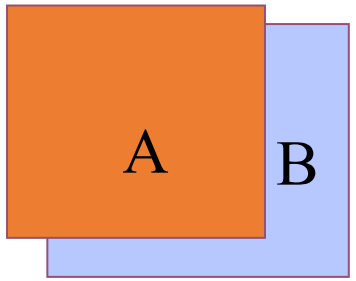


Comment combiner les deux couleurs en une seule?

OVER OPERATOR

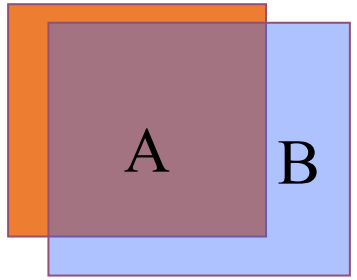
$$Out.rgb = (1 - S.\alpha) \cdot D.rgb + S.\alpha \cdot S.rgb$$

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



A over B:

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



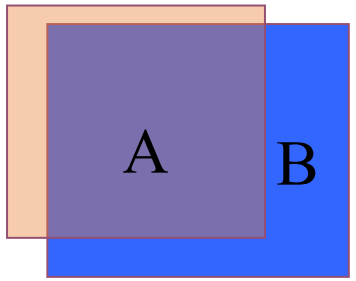
B over A:

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

OVER OPERATOR

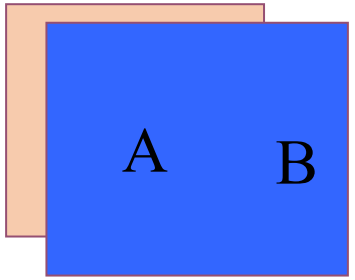
$$Out.rgb = (1 - S.\alpha) \cdot D.rgb + S.\alpha \cdot S.rgb$$

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



A over B:

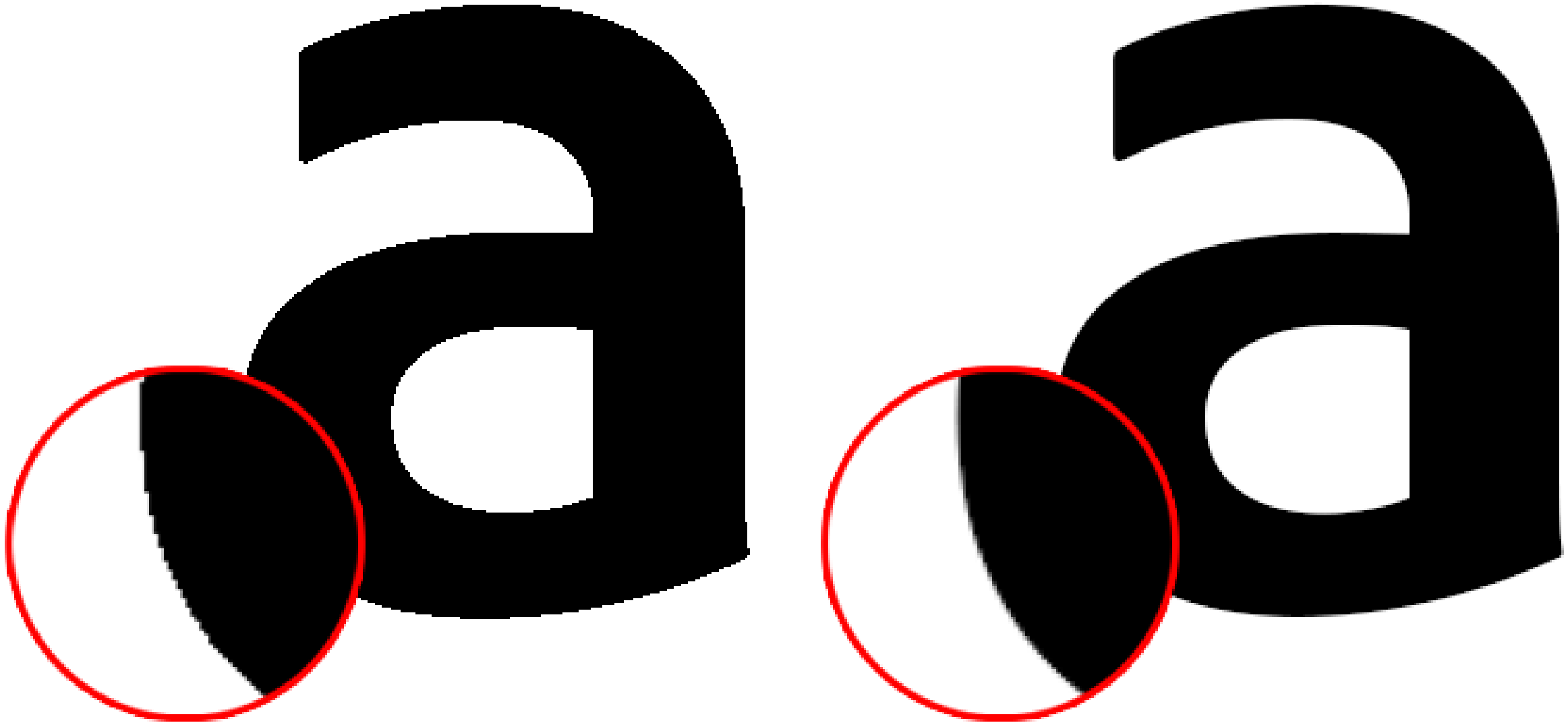
$$Out.rgb = (1 - 0.4) \cdot A.rgb + (0.4) \cdot B.rgb$$



B over A:

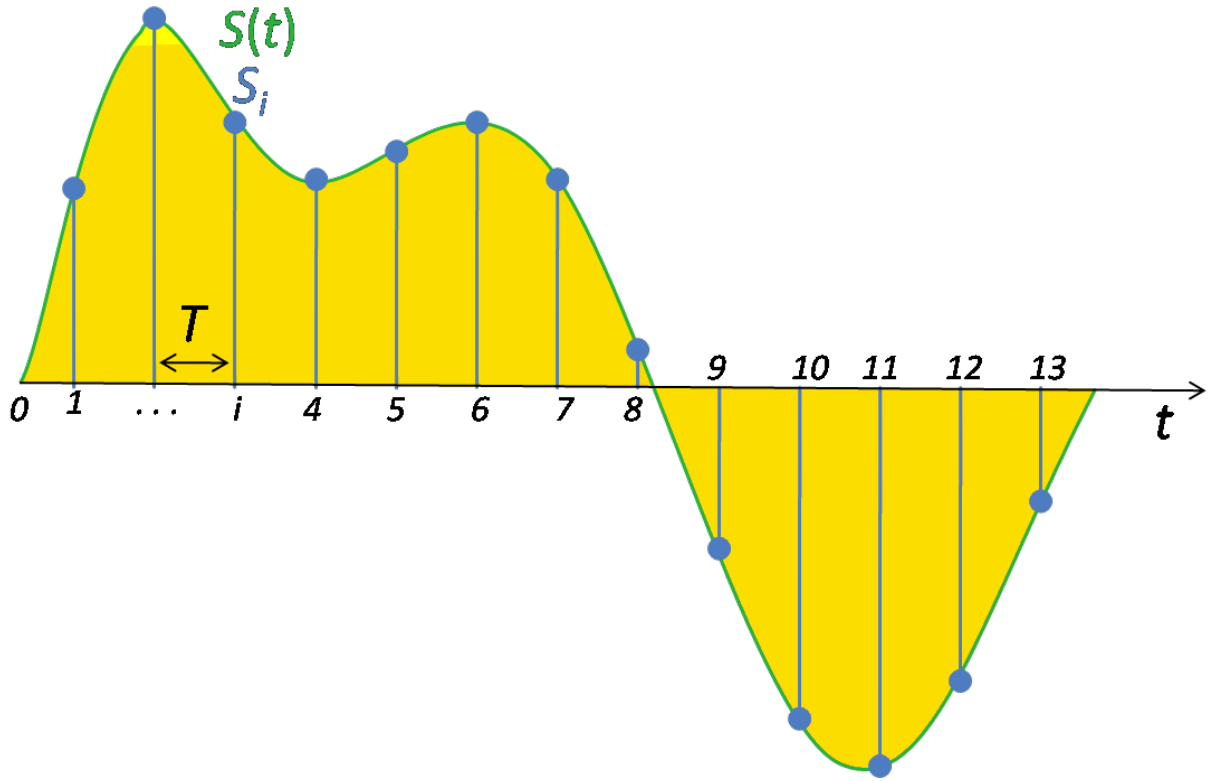
$$Out.rgb = (0) \cdot A.rgb + (1) \cdot B.rgb$$

ALIASSAGE & ANTI-ALIASSAGE



CONTINU VS DISCRET

- Continu → Discret: ***Échantillonnage***
- Discret → Continu: ***Reconstruction/Interpolation***



SUPER-SAMPLING

- Si les position d'échantillonnage sont sur grille régulière, c'est *super-sampling*.
- On peut utiliser les autres modèles des positions
  - Les modèles moins réguliers peuvent éviter les erreurs systématiques (biais)

MULTI-SAMPLING

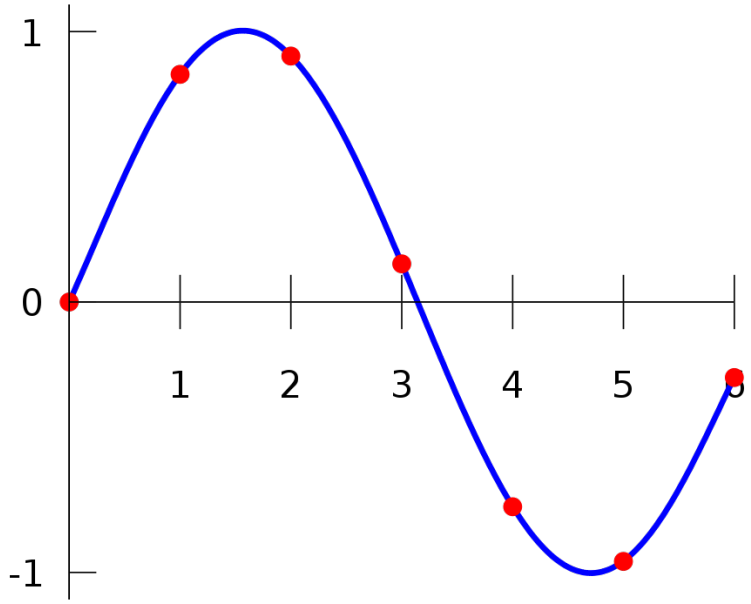
- Pourquoi c'est efficace?

- Les couleurs souvent changent lentement à l'intérieur de chaque triangle
  ⇒ on ne doit pas calculer en haute résolution
  Pas correct pour les textures

- Pour les textures: prétraiter la texture elle-même
  - Mipmap!

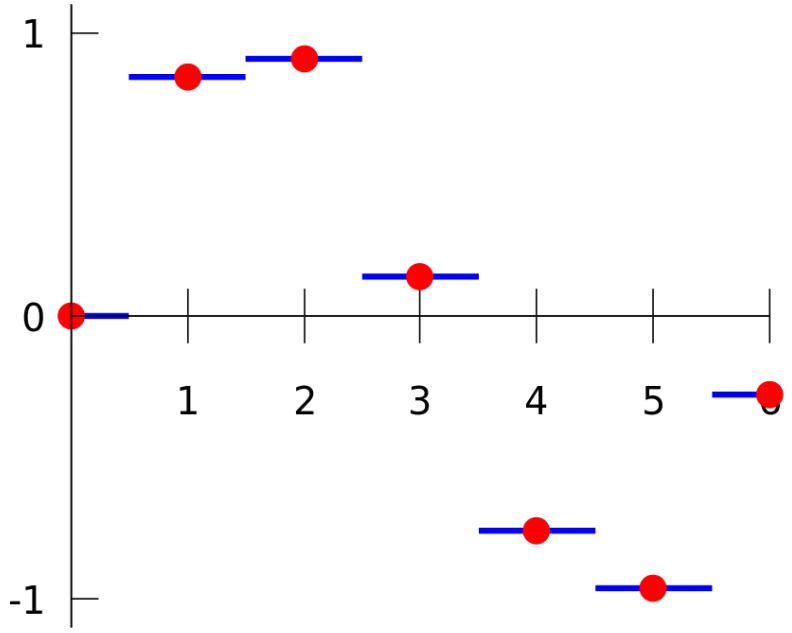
QU'EST-CE QUE L'INTERPOLATION?

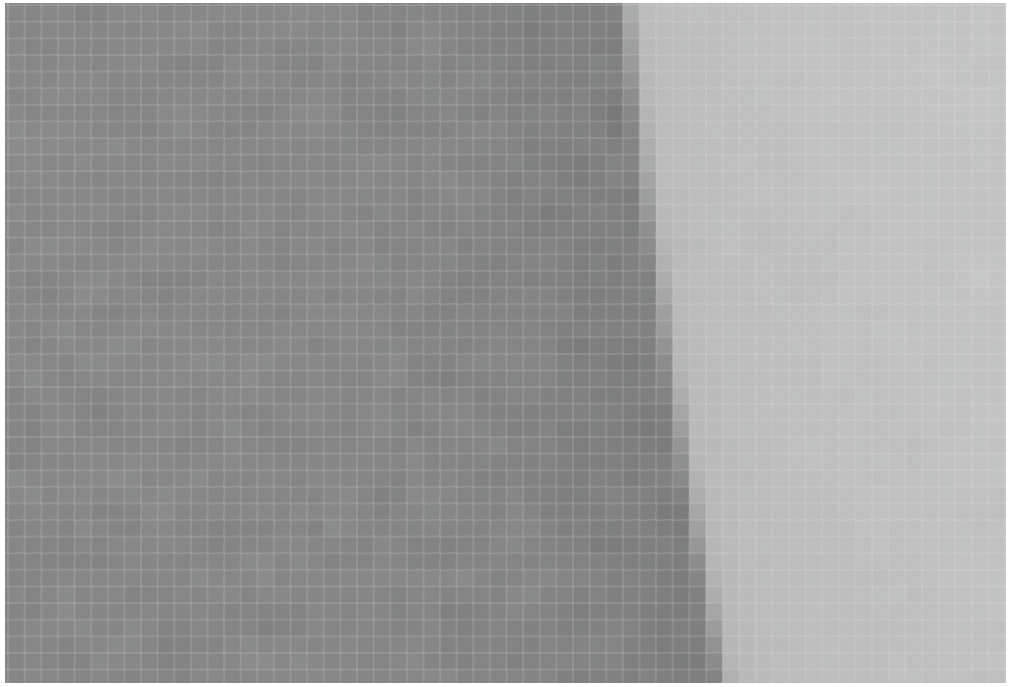
- Intuitivement: remplir les valeurs entres celles qu'on connait
- Plus formellement:
  - Soit $\{x_i, y_i\}, i = 1 \dots N$
  - Trouver une fonction $f(x)$, telle que $\boldsymbol{f(x_i) = y_i}, i = 1 \dots N$



L'INTERPOLATION CONSTANTE

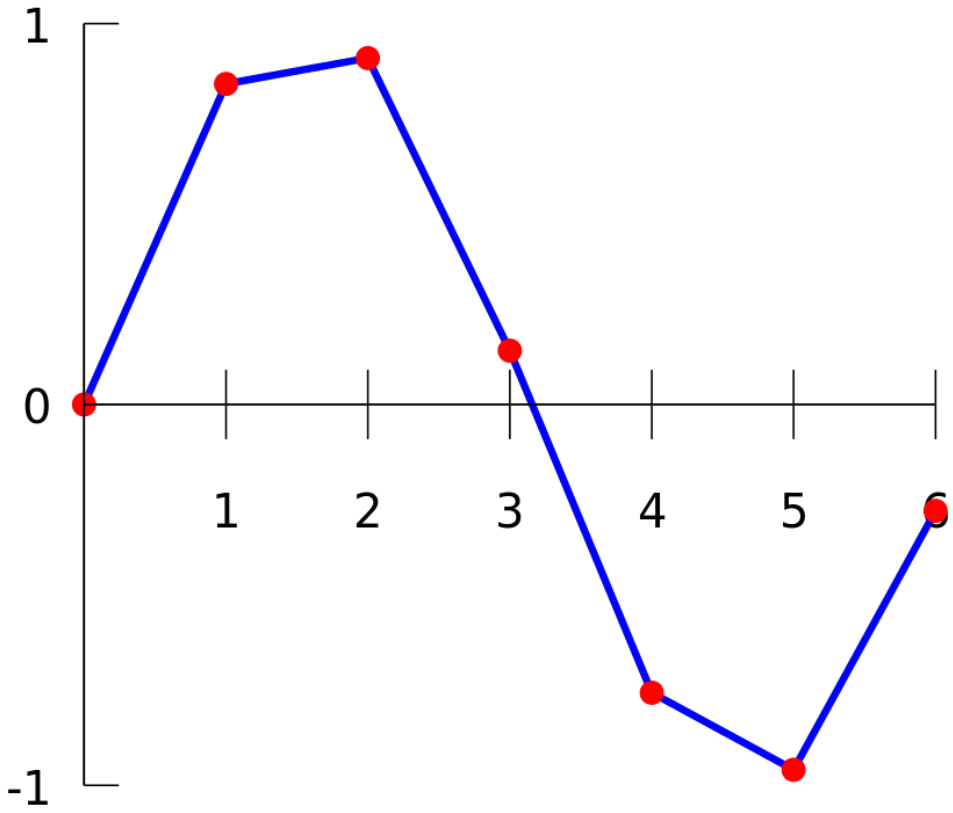
- Aussi appelée « le plus proche voisin»
- Très simple
- Mais discontinue





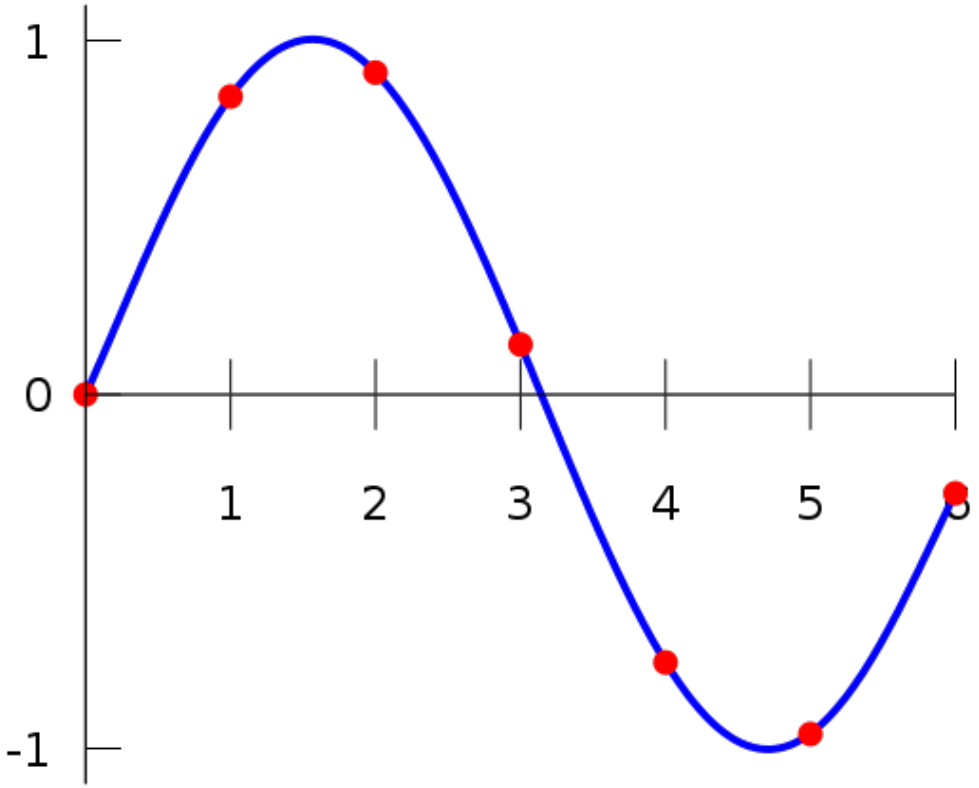
L'INTERPOLATION LINÉAIRE

- Dessiner une ligne entre chaque paire de points consécutifs



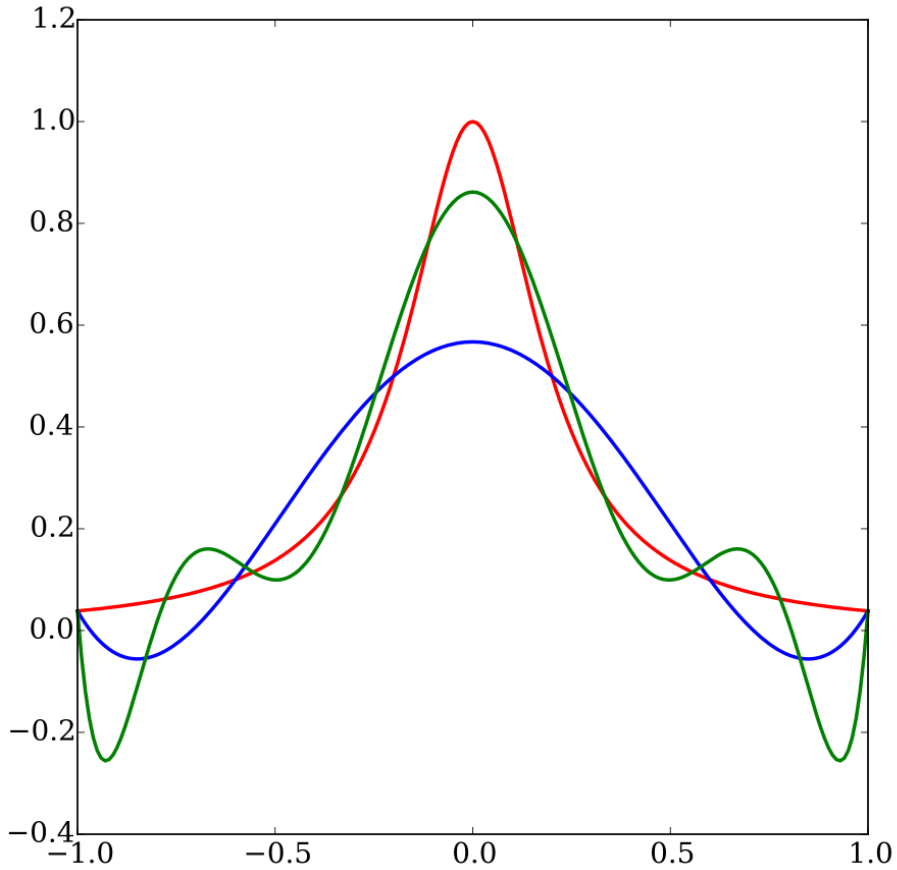
L'INTERPOLATION POLYNOMIALE

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$



PHÉNOMÈNE DE RUNGE

Si le degré du polynôme est plus élevé, la fonction devient ondulée



La fonction qu'on interpole

Le polynôme de degré 5

Le polynôme de degré 9

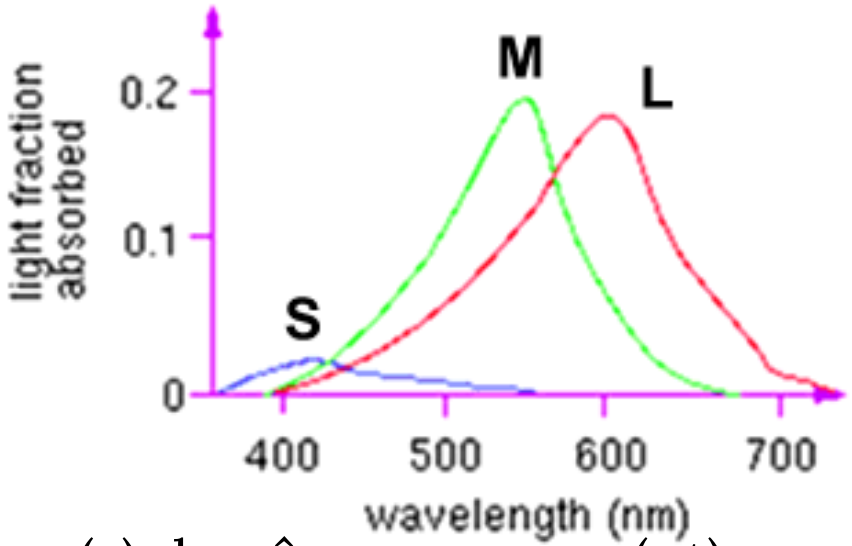
LES SPLINES

- On limite le degré du polynôme à quelque chose petit, e.g. 3
- Utiliser les fonctions par morceaux
- On peut rendre la fonction lisse

$$fx(x) = \begin{cases} \frac{1}{4}(x+2)^3 & -2 \leq x \leq -1 \\ \frac{1}{4}\left(3|x|^3 - 6x^2 + 4\right) & -1 \leq x \leq 1 \\ \frac{1}{4}(2-x)^3 & 1 \leq x \leq 2 \end{cases}$$

LA TRICHROMIE

- Trois types de cônes
  - L ou R, plus sensible à la lumière rouge (610 nm)
  - M ou G, plus sensible à la lumière verte (560 nm)
  - S ou B, plus sensible à la lumière bleue (430 nm)



  - Le daltonisme = type(s) de cônes manque(nt)

L'ESPACE DE COULEUR CIE

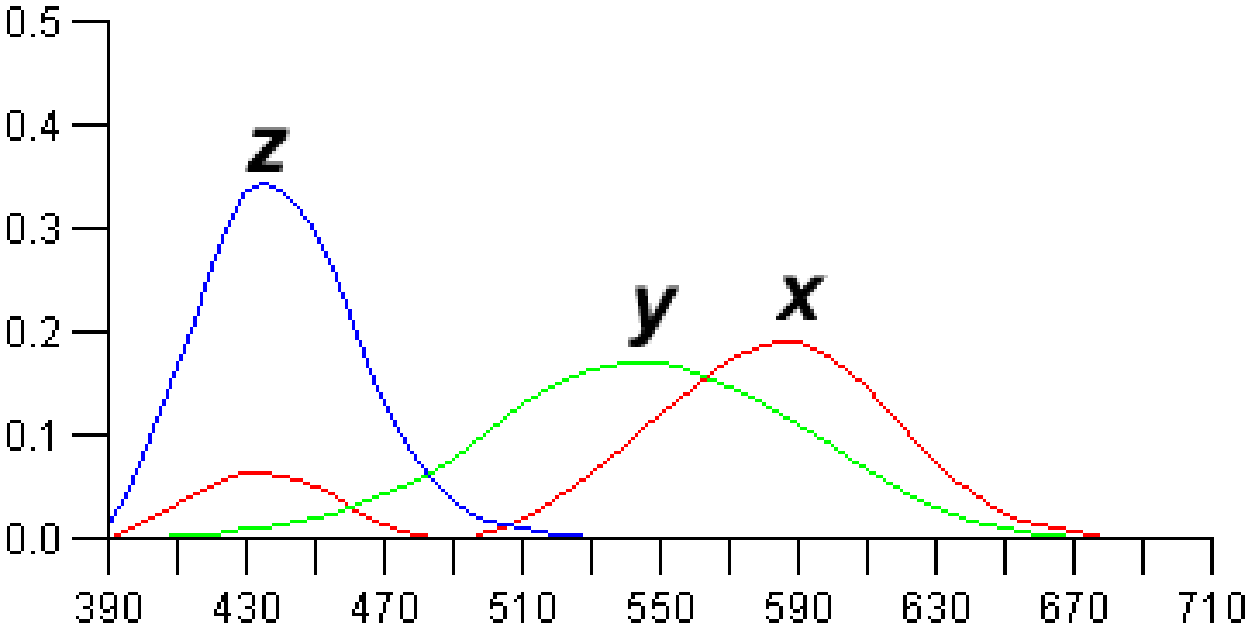
- CIE a défini 3 lumières X, Y, Z imaginaires
  - Toutes les longueurs d'onde λ peuvent être construites par une combinaison positive
  - Les fonctions de base!

NB:

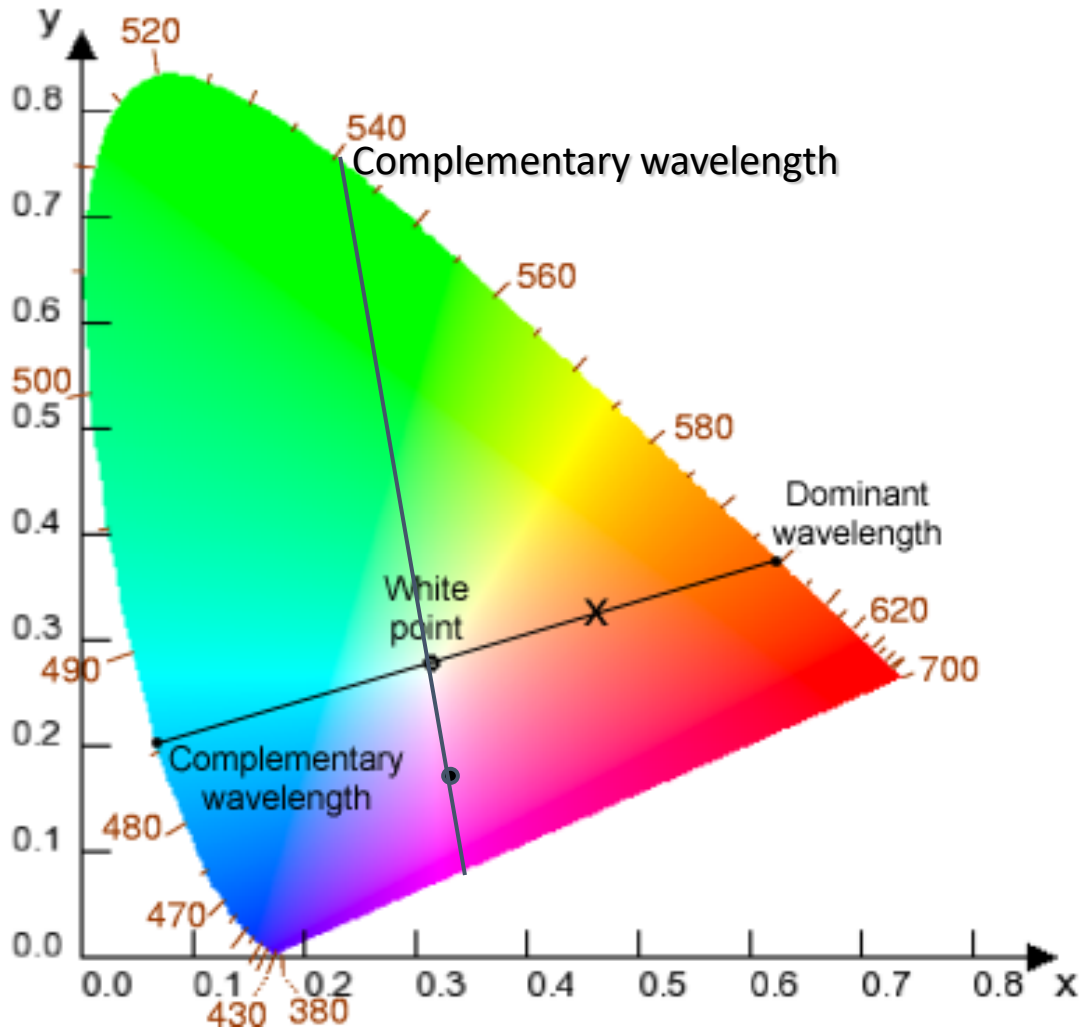
$X \sim R$

$Y \sim G$

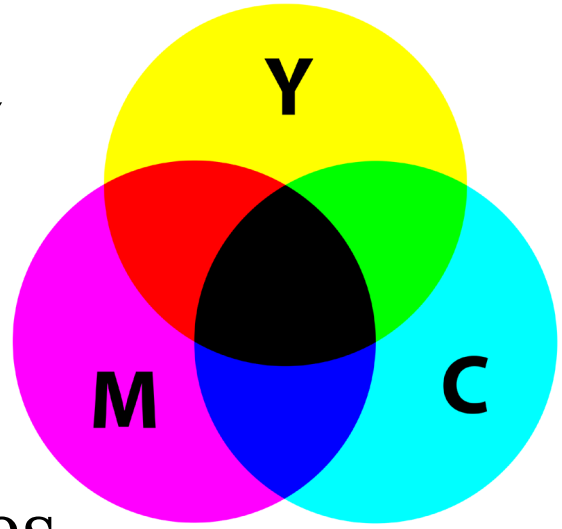
$Z \sim B$



L'INTERPOLATION
LES COULEURS COMPLÉMENTAIRES
LA LONGUEUR D'ONDE DOMINANTE



LE MODÈLE DE COULER CMY

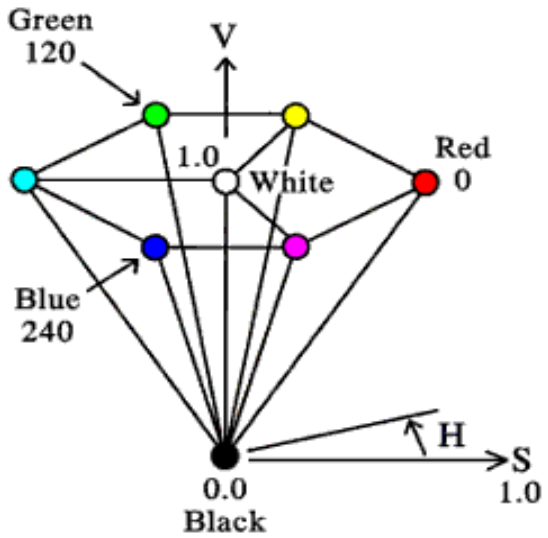


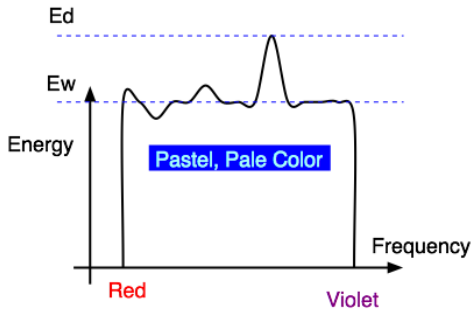
- Utilisé dans l'impression
  - La lumière est absorbée par les colorants
- Cyan (C), magenta (M) et jaune (Y), les primaires, sont les compléments de rouge, vert et bleu
- Les couleurs primaires sont soustraites du papier blanc
  - Rouge = Blanc-Cyan = Blanc-Vert-Bleu (0,1,1)
  - Vert = Blanc-Magenta = Blanc-Rouge-Bleu (1,0,1)
  - Bleu = Blanc-Jaune = Blanc-Rouge-Vert (1,1,0)
  - $(r, v, b) = (1 - c, 1 - m, 1 - j)$

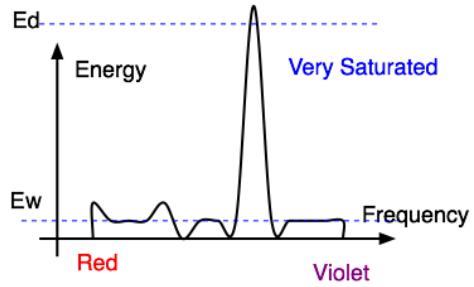
L'ESPACE DE COULEUR HSV

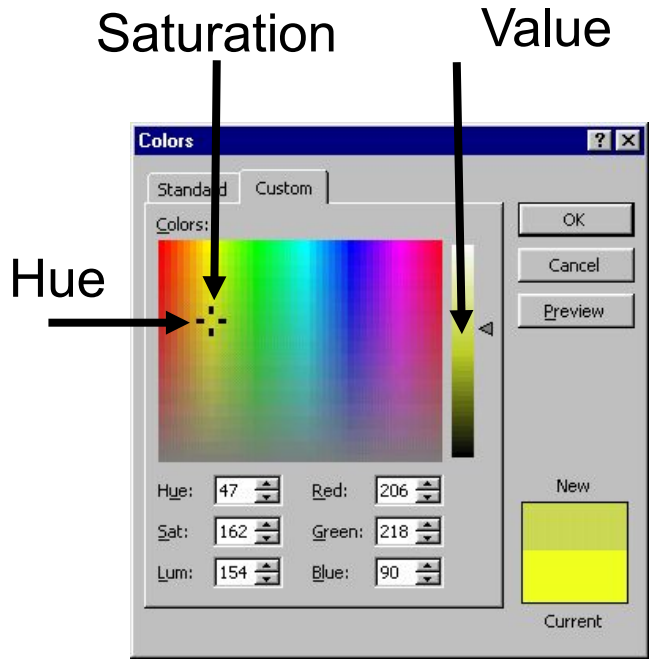
Un espace plus intuitif pour les gens

- *H* = *Hue* (la teinte)
  - La longueur d'onde dominante, "**la couleur**"
- S = *Saturation*
  - La distance au blanc/gris
- V = *Value* (la valeur)
  - La distance au noir
  - aussi: *brightness B*, *intensity I*, *lightness L*









MERCI ET AU REVOIR!

FIN