

<http://tiny.cc/ift3355>

# IFT 3355: INFOGRAPHIE

# PIPELINE DE RENDU II

Livre de référence: G:I.1

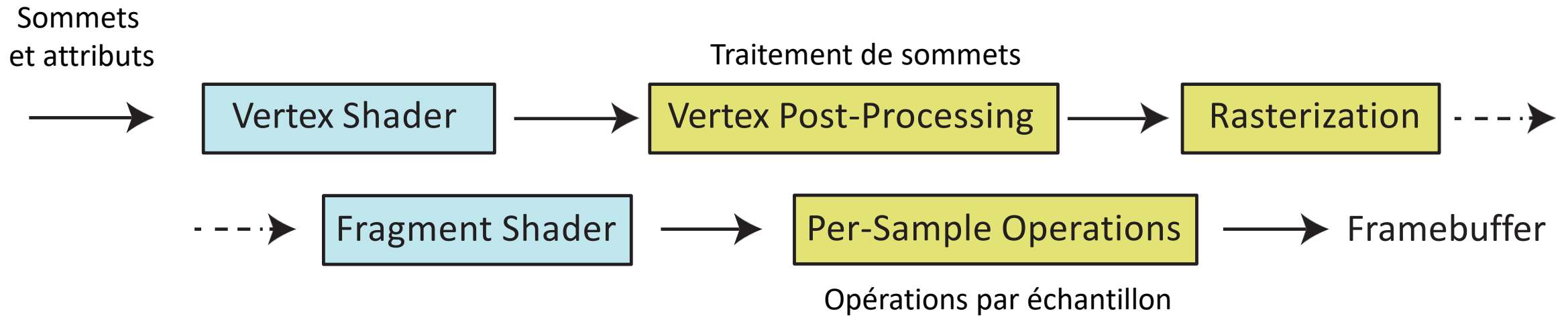


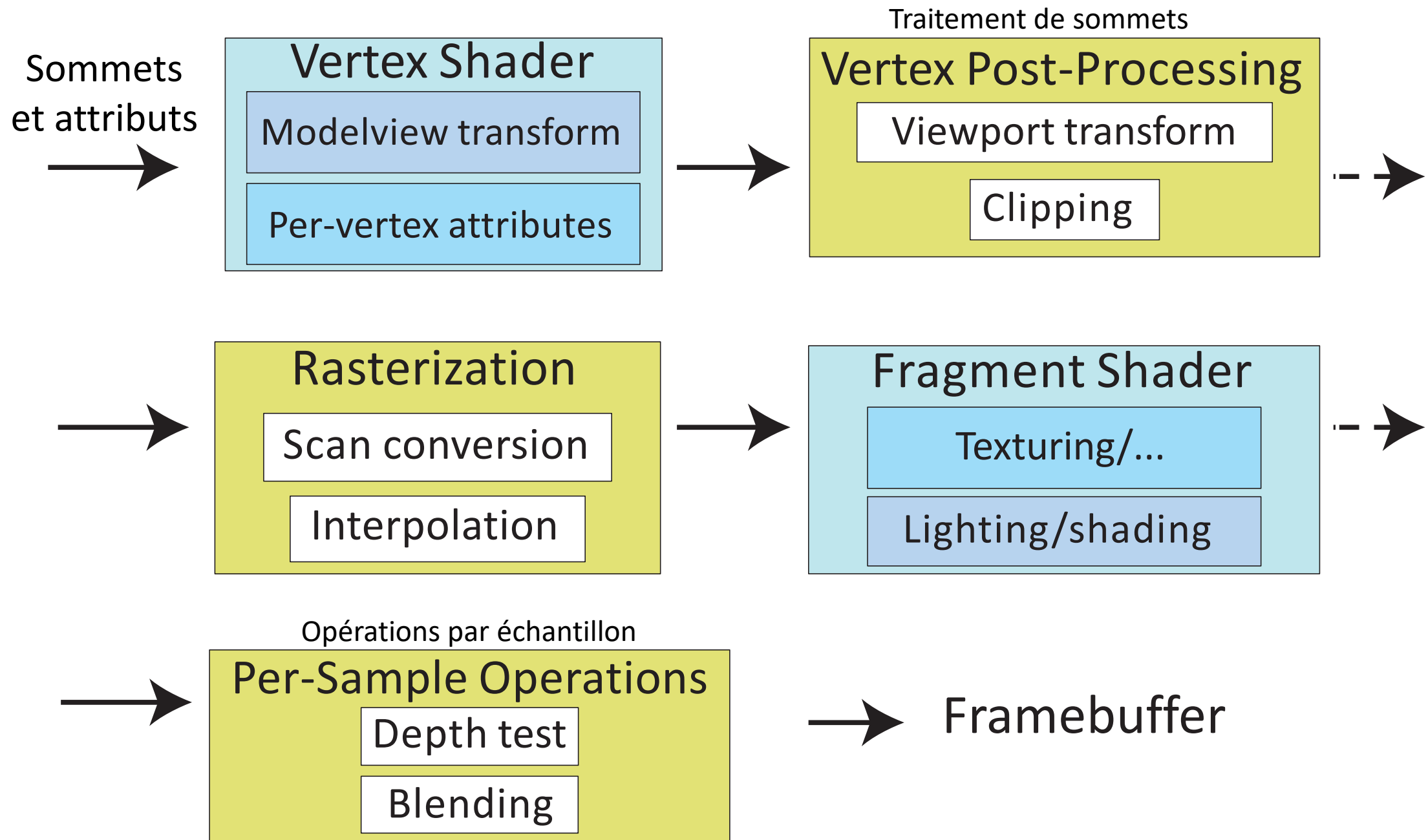
Mikhail Bessmeltsev

# D1

- Comment ça se passe?
- Le contrôleur fait ses premiers pas?
  
- Venez aux démos
- Apprenez comment déboguer

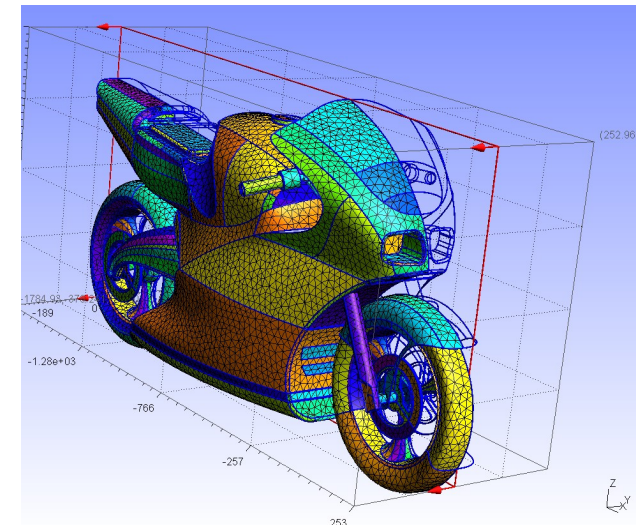
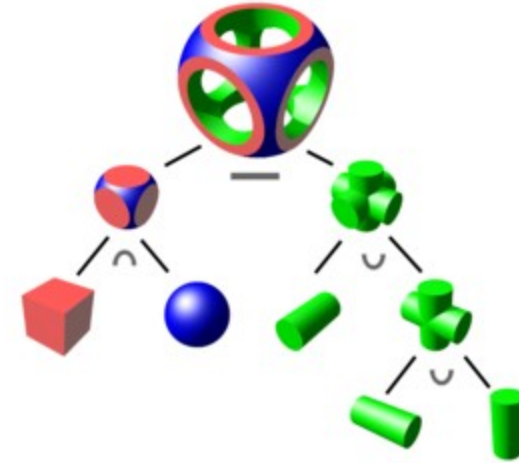
# OPENGL PIPELINE DE RENDU





# COMMENT REPRÉSENTER LES FORMES?

- Volumétrieque – l'algèbre de Bool avec les primitives volumétriques
  - Les sphères, cônes, cubes, ...
- Représentation frontière (*Boundary representation*) – l'union de patches de surface
  - Une seul primitive – triangle/quad
  - Une surface d'ordre élevé



# LES FORMES – LES COURBES/SURFACES

- Les représentations mathématiques:
  - Les fonctions explicites
  - Les fonctions paramétriques
  - Les fonctions implicites

# LES FORMES: LES FONCTIONS EXPLICITES

- Les courbes:

- $y$  est une fonction de  $x$ :
- Pas toutes le courbes

$$y := \sin(x)$$

- Les surfaces:

- $z$  est une fonction de  $x$  et  $y$ :
- Pas toutes les surfaces

$$z := \sin(x) + \cos(y)$$

# LES FORMES: LES FONCTIONS PARAMÉTRIQUE

- Les courbes:
  - 2D:  $x$  et  $y$  sont les fonctions de la valeur du paramètre  $t$
  - 3D:  $x$ ,  $y$  et  $z$  sont les fonctions de la valeur du paramètre  $t$

$$C(t) := \begin{pmatrix} \cos(t) \\ \sin(t) \\ t \end{pmatrix}$$



# LES FORMES: LES FONCTIONS PARAMÉTRIQUE

- Les surfaces:
  - Une surface  $S$  est définie en fonction des valeurs des paramètres  $s, t$
  - Les noms des paramètres peuvent être différents:

$$S(\phi, \theta) := \begin{pmatrix} \cos(\phi) \cos(\theta) \\ \sin(\phi) \cos(\theta) \\ \sin(\theta) \end{pmatrix}$$

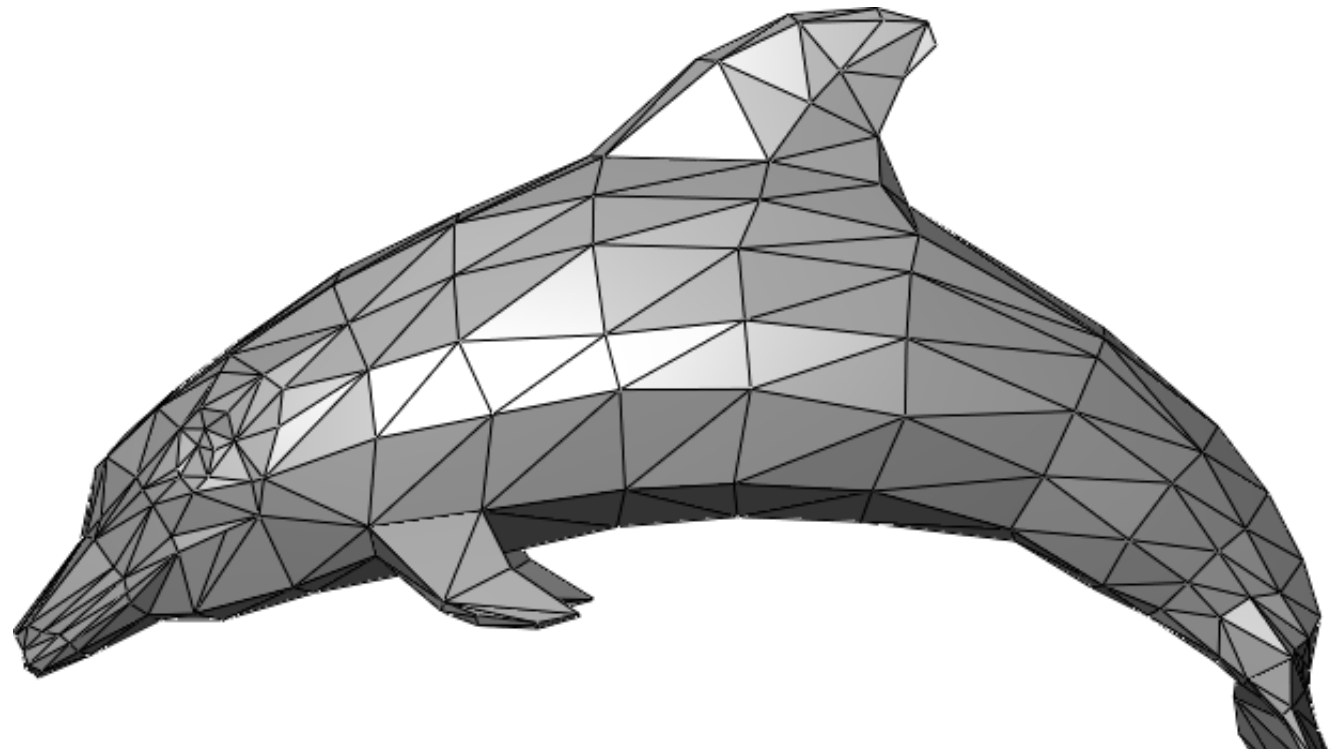
# LES FORMES: LES FONCTIONS IMPLICITES

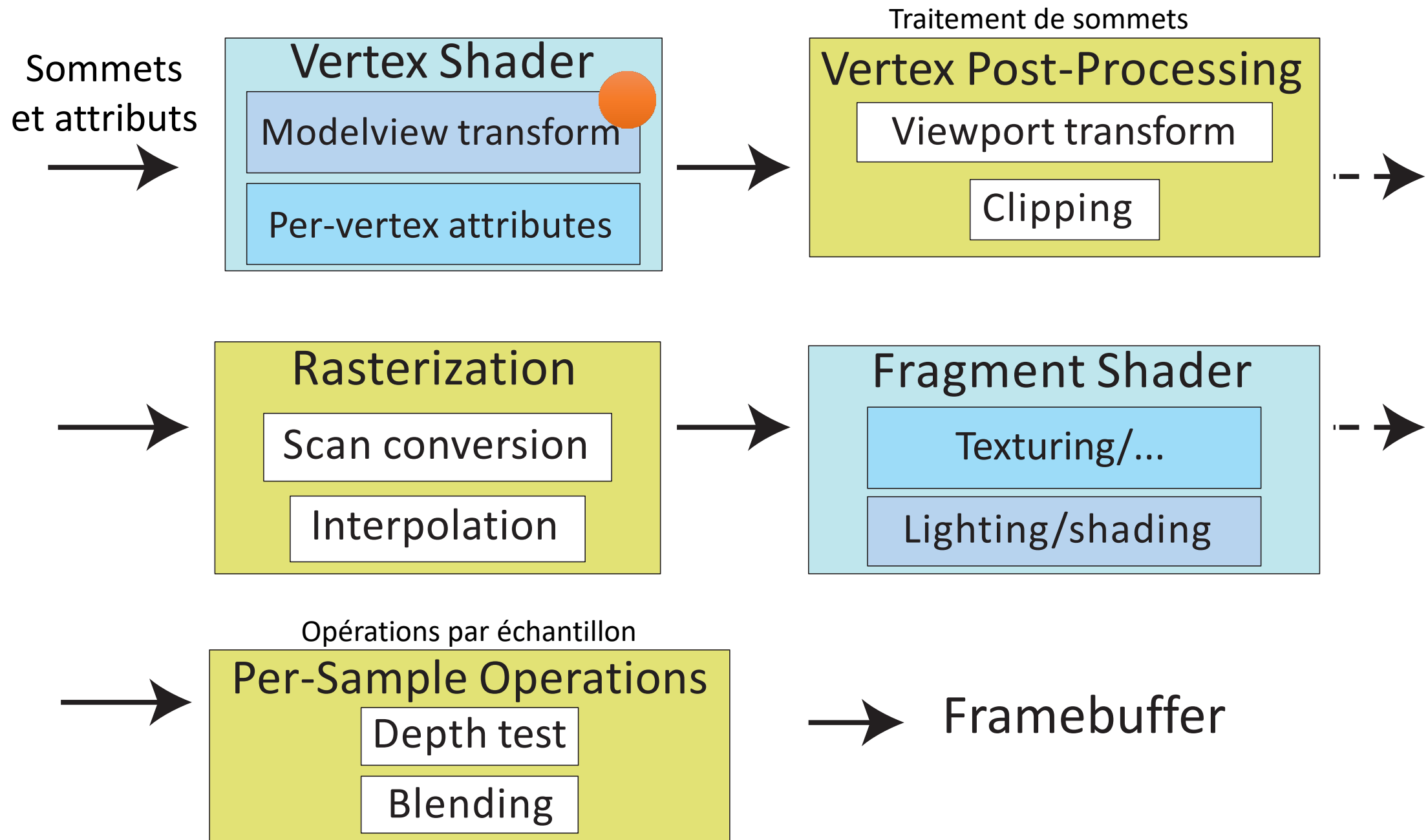
- La surface (3D) ou la courbe (2D) est défini par les racines de la fonction
  - E.g.:

$$S(x, y, z): x^2 + y^2 + z^2 - 1 = 0$$

# LES FORMES: LES MAILLAGES DE TRIANGLES

- La liste de sommets
- Les triangles sont définis comme  $\{\text{vertex\_index1}, \text{vertex\_index2}, \text{vertex\_index3}\}$



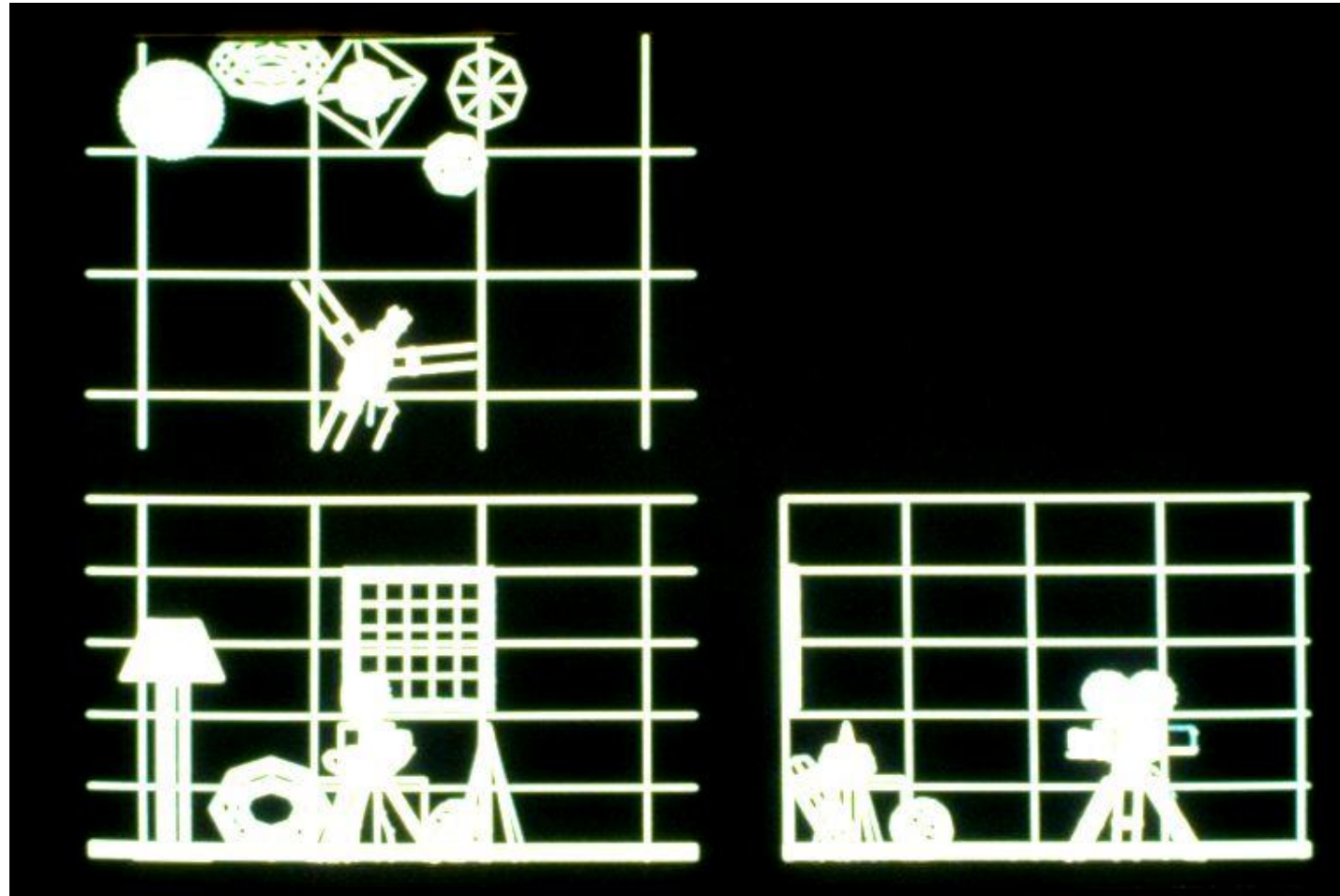


# LES TRANSFORMATIONS DE MODÈLES ET DE VUE

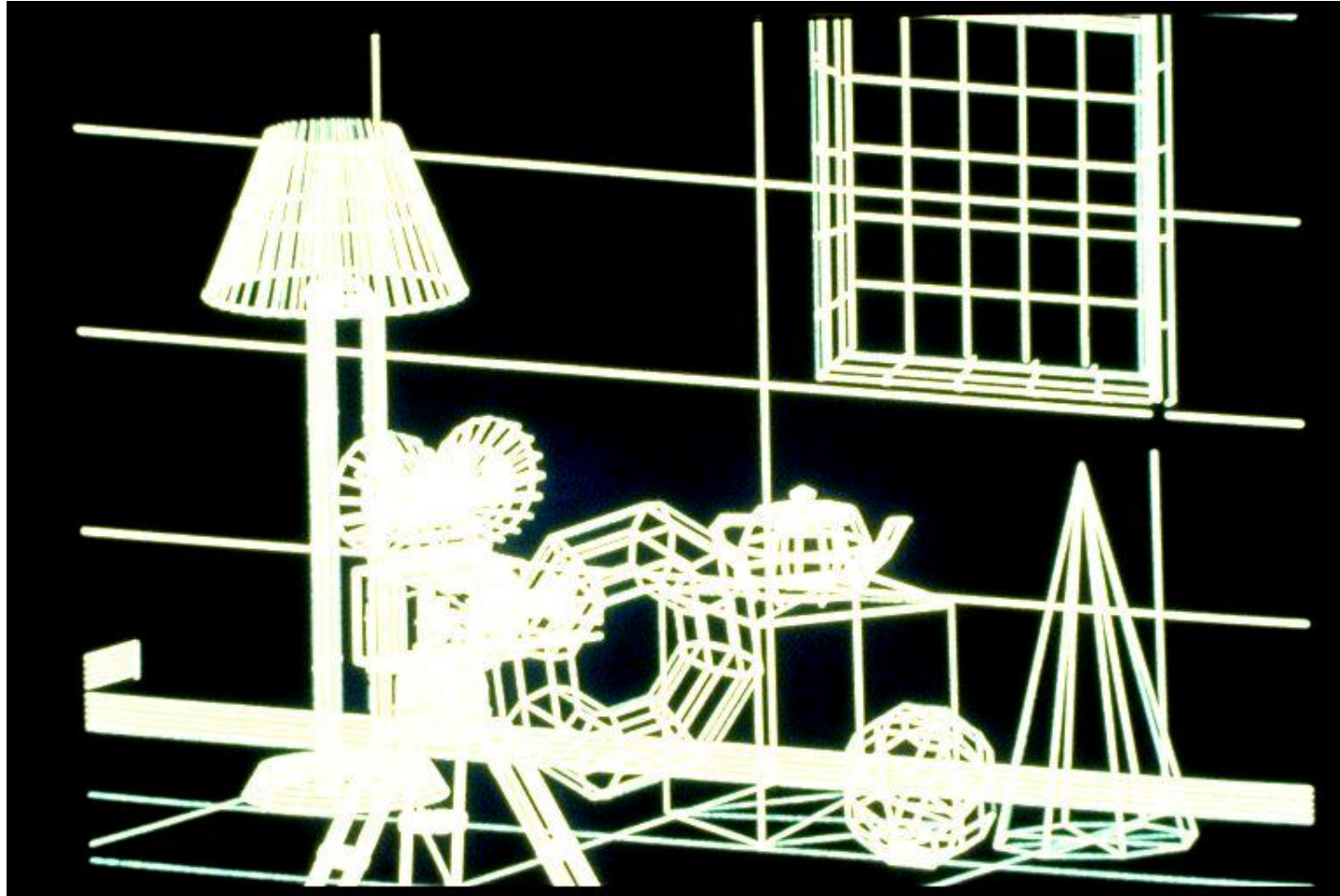
- Placer des objets – Les transformations de modèles
  - Mapper des points du système de coordonnées de l'objet au système de coordonnées du monde
- Regarder depuis la caméra – La transformation de la vue
  - Mapper des points du système de coordonnées du monde au système de coordonnées de vue (l'oeil)



# PLACEMENT D'OBJETS LES TRANSFORMATIONS DE MODÈLES

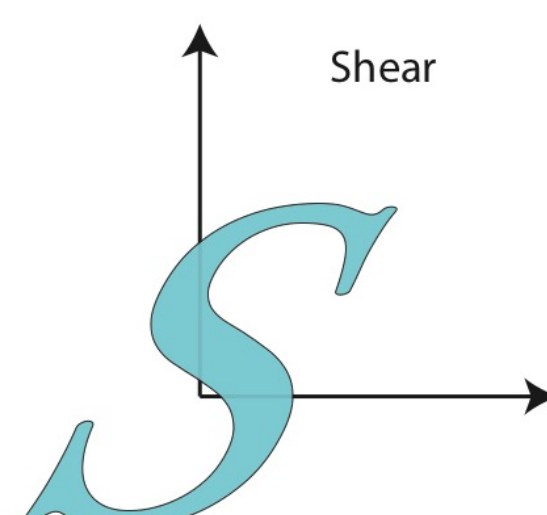
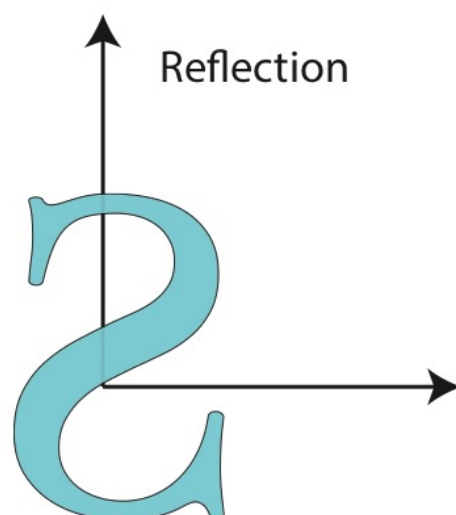
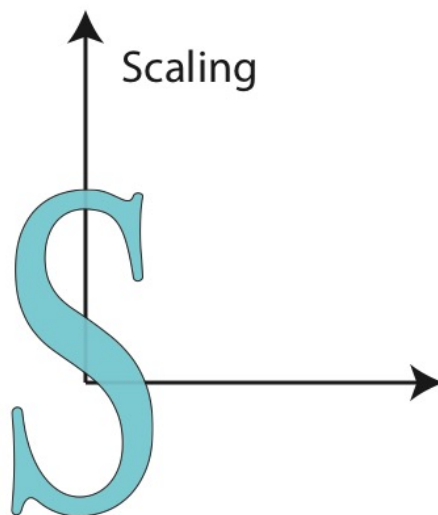
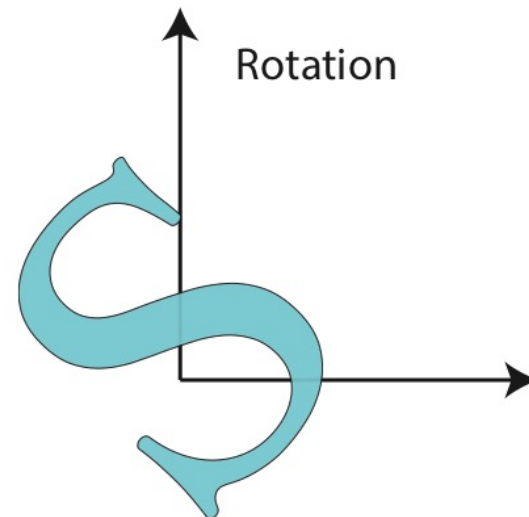
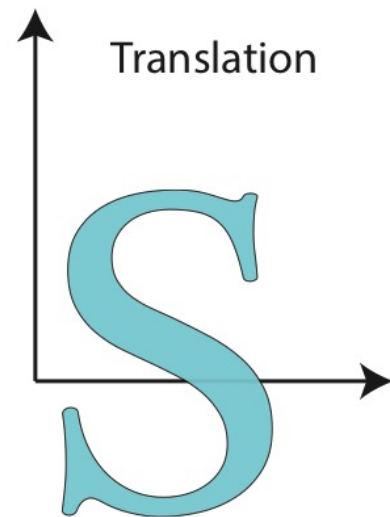
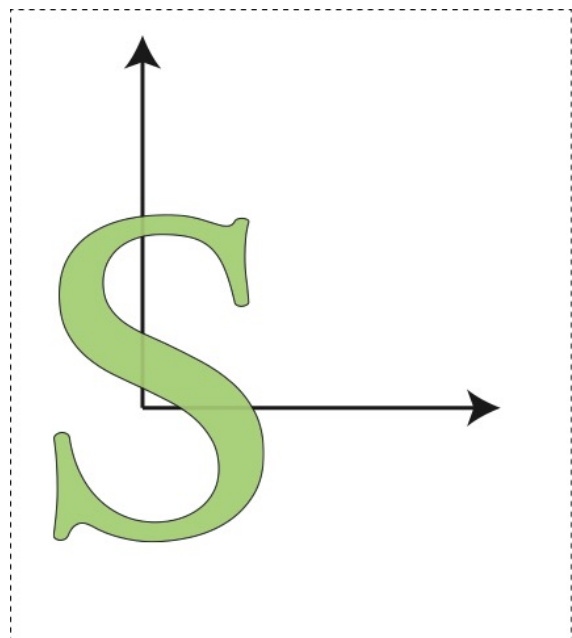


# REGARDER DEPUIS LA CAMÉRA – LA TRANSFORMATION DE VUE

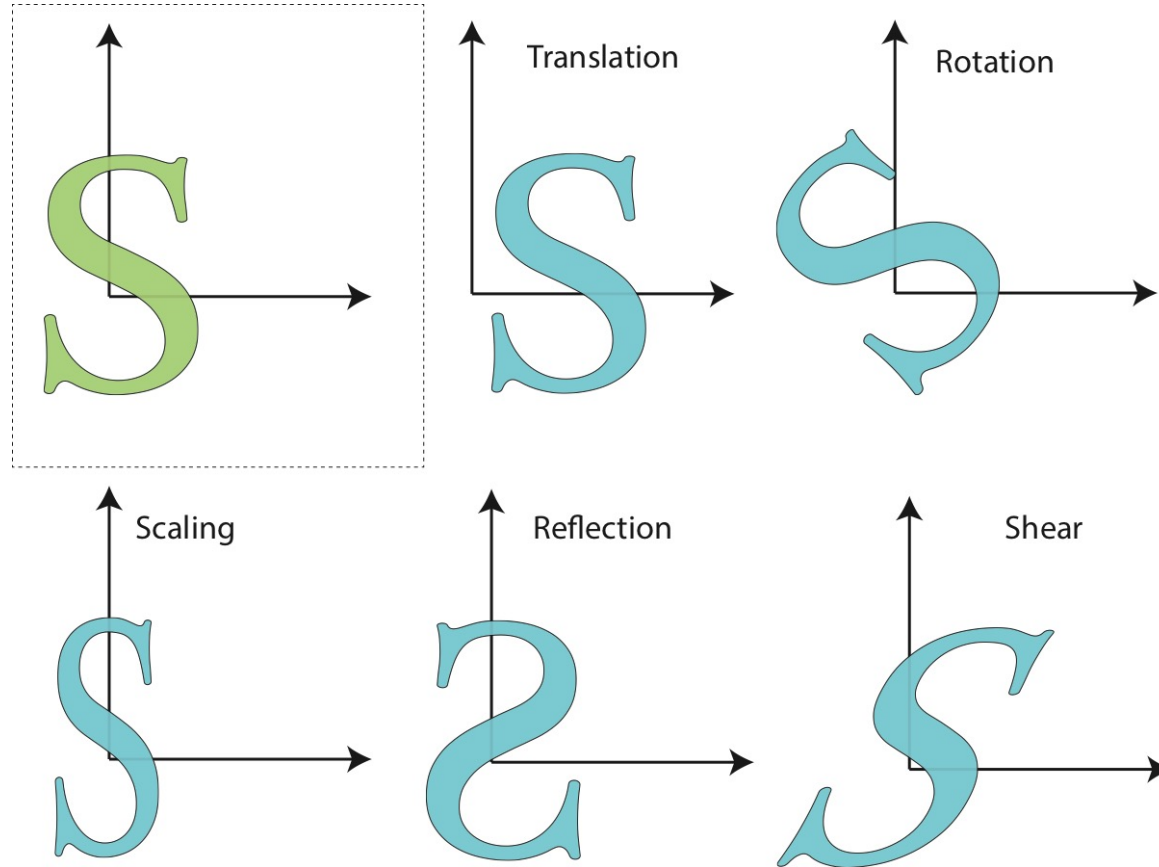




# LES TRANSFORMATIONS



# LES TRANSFORMATIONS



- Autres transformations (pas gérées par le pipeline de rendu):
  - Déformation de forme libre
    - Shaders peuvent faire ça



# LES TRANSFORMATIONS DE MODÈLES ET DE VUE

- Les transformations linéaires
  - Les rotations, changements d'échelle, cisaillement
  - Peuvent être exprimées par une matrice 3x3
  - E.g. changement d'échelle (non uniforme):

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

# LES TRANSFORMATIONS DE MODÈLES ET DE VUE

- Les transformations affines
  - Les transformations linéaires + translations
  - Peuvent être exprimées par une matrice 3x3 + un vecteur 3D
  - E.g. changement d'échelle + translation:

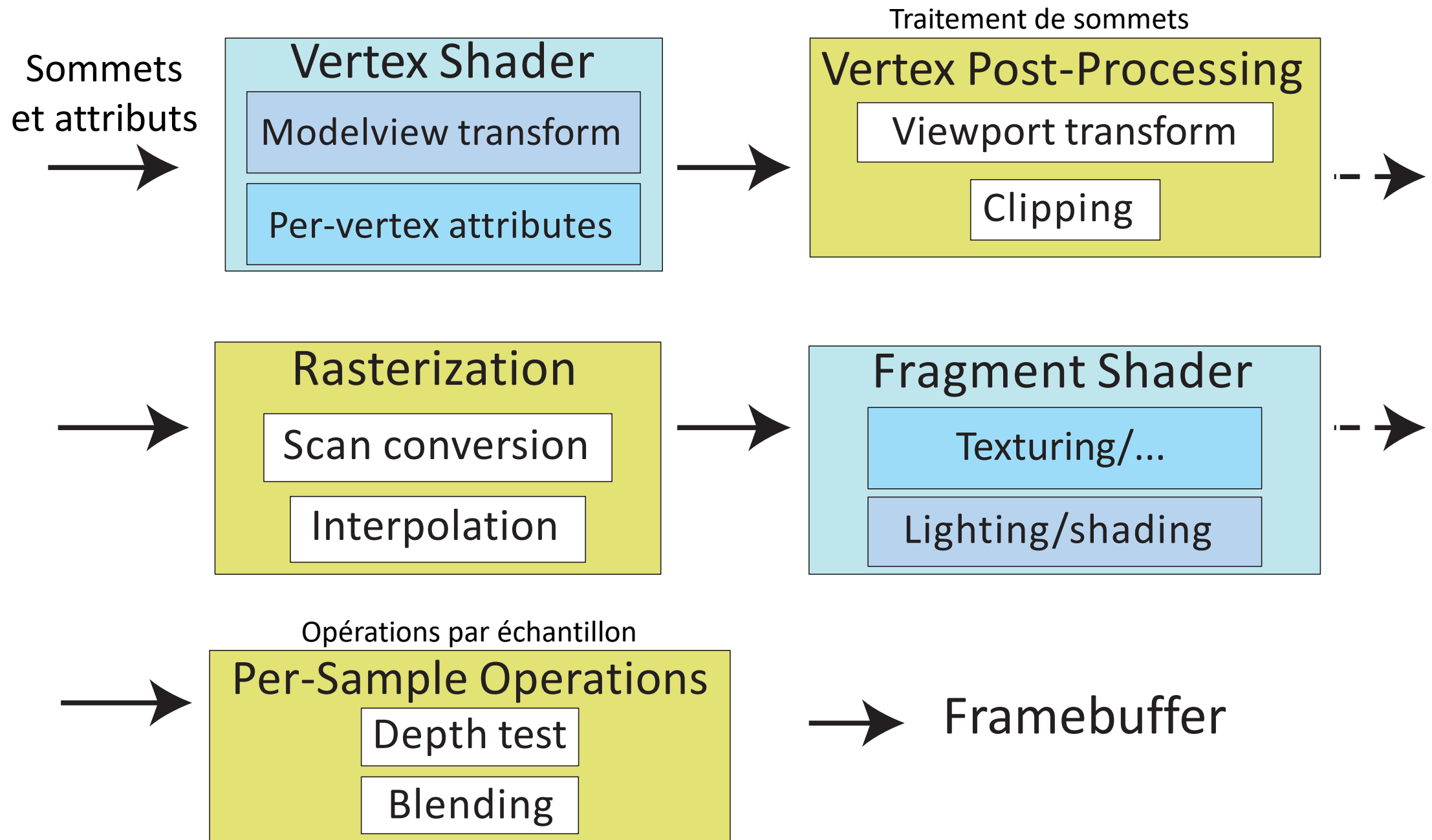
$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

- Une autre représentation: matrice 4x4 en coordonnées homogènes

# LES MATRICES

- Les coordonnées de l'objet → les coordonnées du monde
  - **Matrice du modèle**
  - Une par objet
- Les coordonnées du monde → les coordonnées de la caméra
  - **Matrice de la vue**
  - Une par caméra

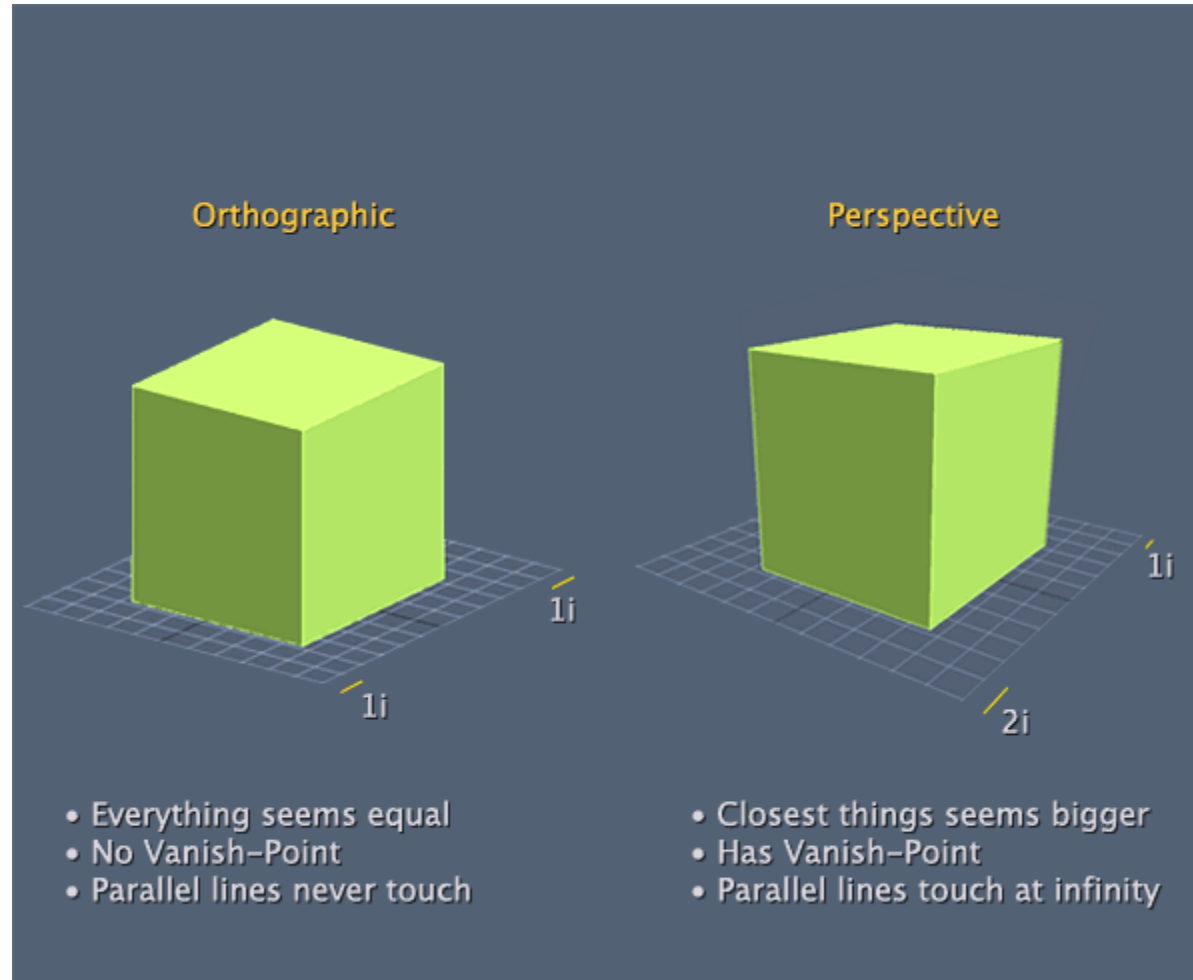
# PIPELINE: PLUS DE DÉTAILS



# PROJECTION

- Projeter la géométrie 3D sur le plan 2D
- Simuler une caméra
- Un modèle de la caméra:
  - Sténopé (*pinhole*)
  - Il y a des modèles plus complexes, mais c'est rare en CG

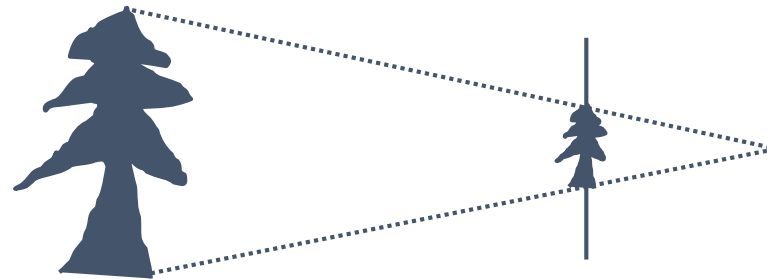
# PROJECTION PERSPECTIVE





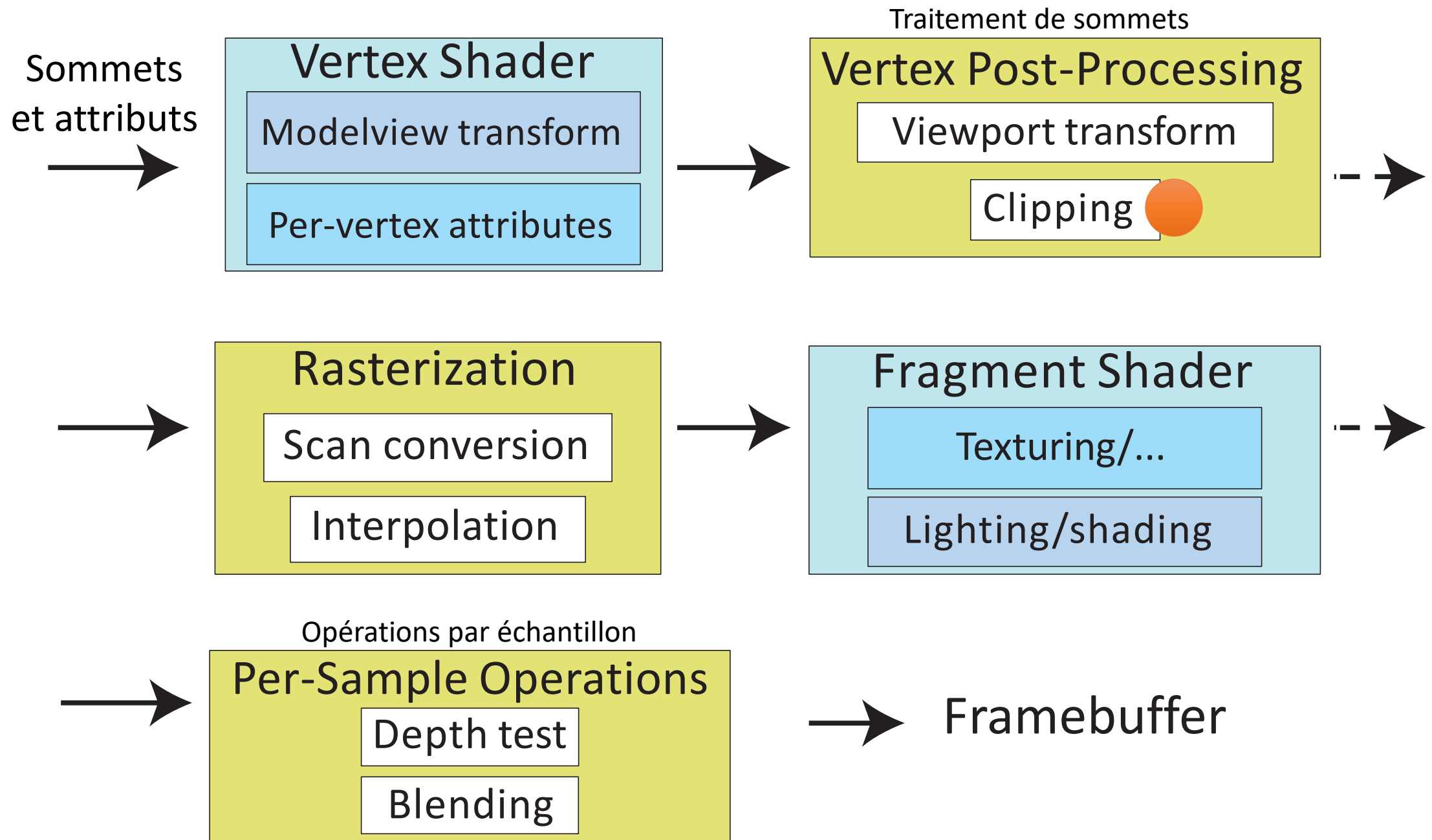
# PROJECTION PERSPECTIVE

- Le plan image est devant le centre de projection



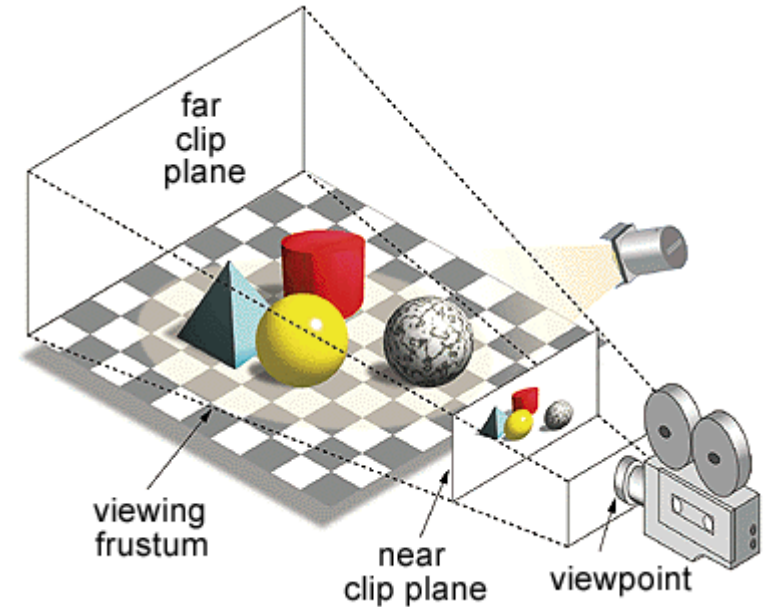
- La transformation perspective est **l'une** des transformations de projection
- Les transformations linéaires et affines appartiennent également à cette classe
- Toutes les transformations projectives peuvent être exprimées comme une multiplication par matrice  $4 \times 4$

# PIPELINE: PLUS DE DÉTAILS

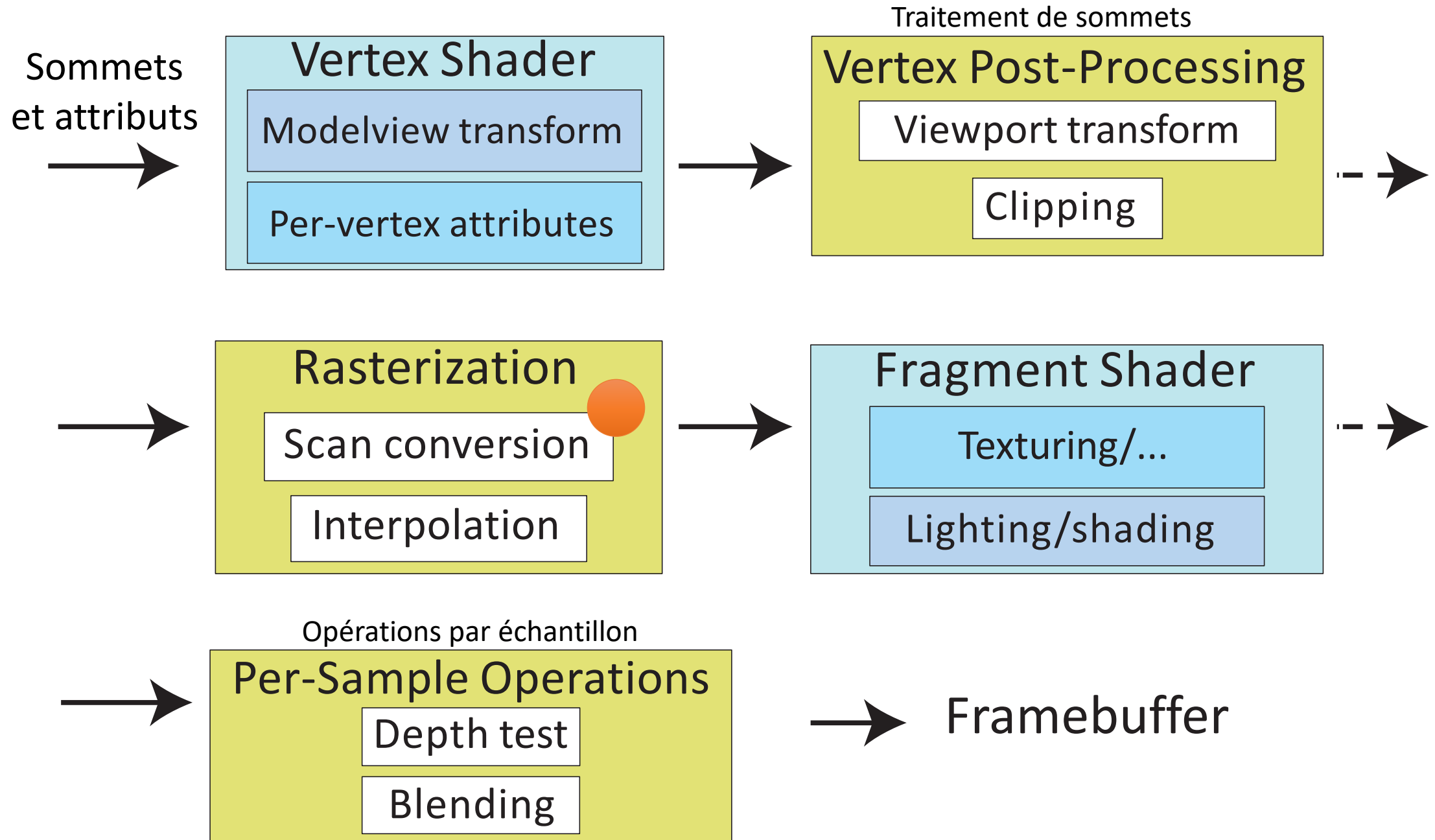


# CLIPPING

- Supprimer la géométrie invisible
  - Géométrie en dehors du volume de vue (*viewing frustum*)
    - Y compris si l'objet est trop près ou trop loin
  
- Optimisation

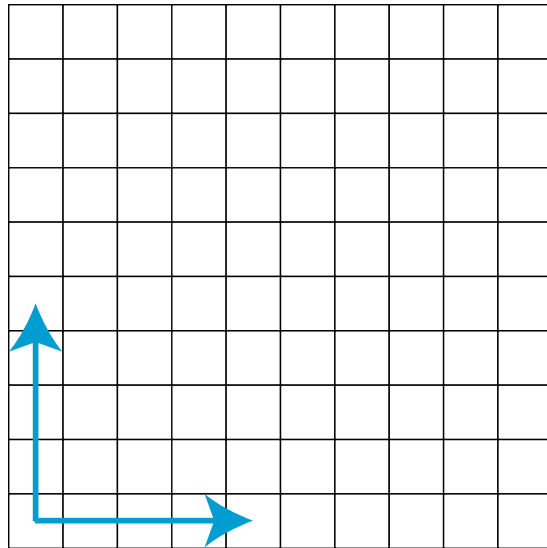


# PIPELINE: MORE DETAILS

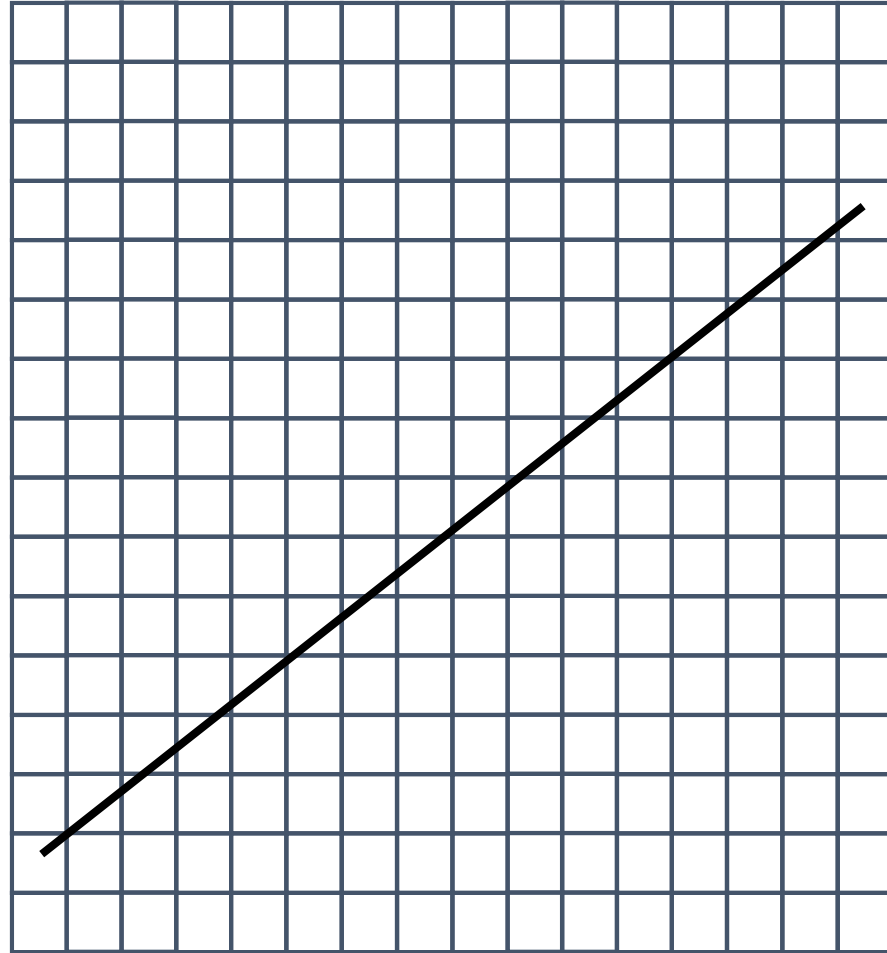


# SCAN CONVERSION/RASTERIZATION

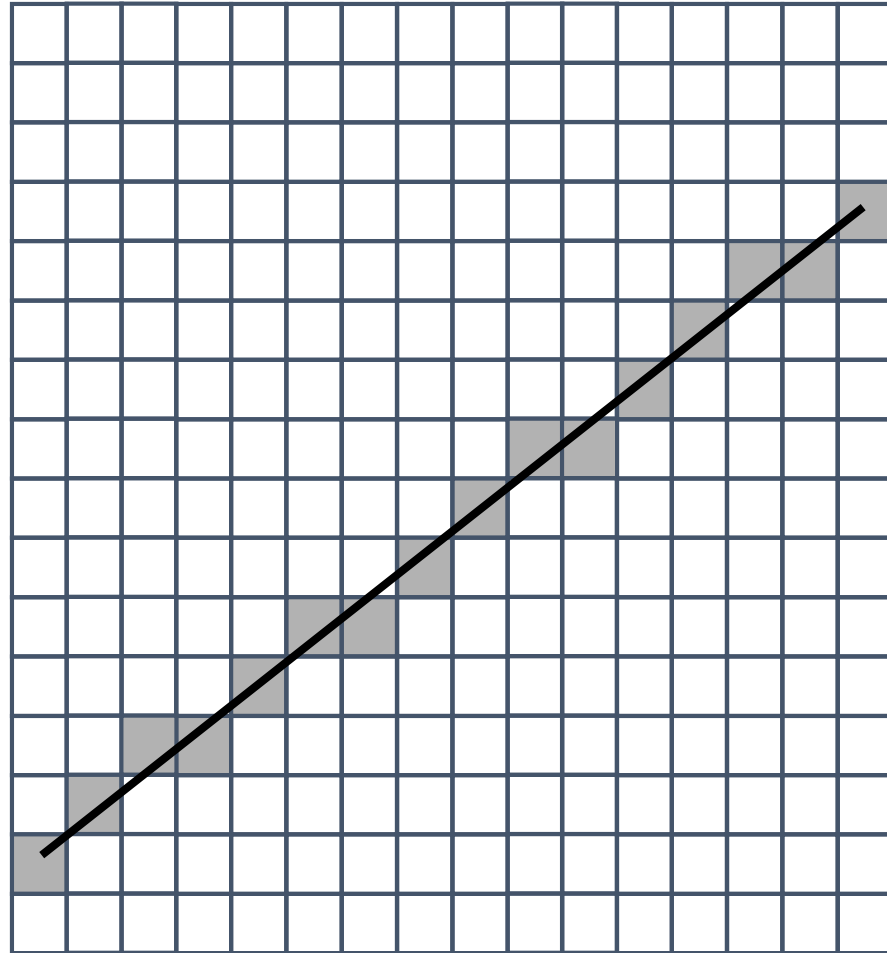
- Convertir la géométrie continue en représentation discrète
- L'affichage raster est une grille discrète des éléments
- La terminologie:
  - **Espace écran:** Le système discret 2D de l'affichage (pixels)



# SCAN CONVERSION



# SCAN CONVERSION

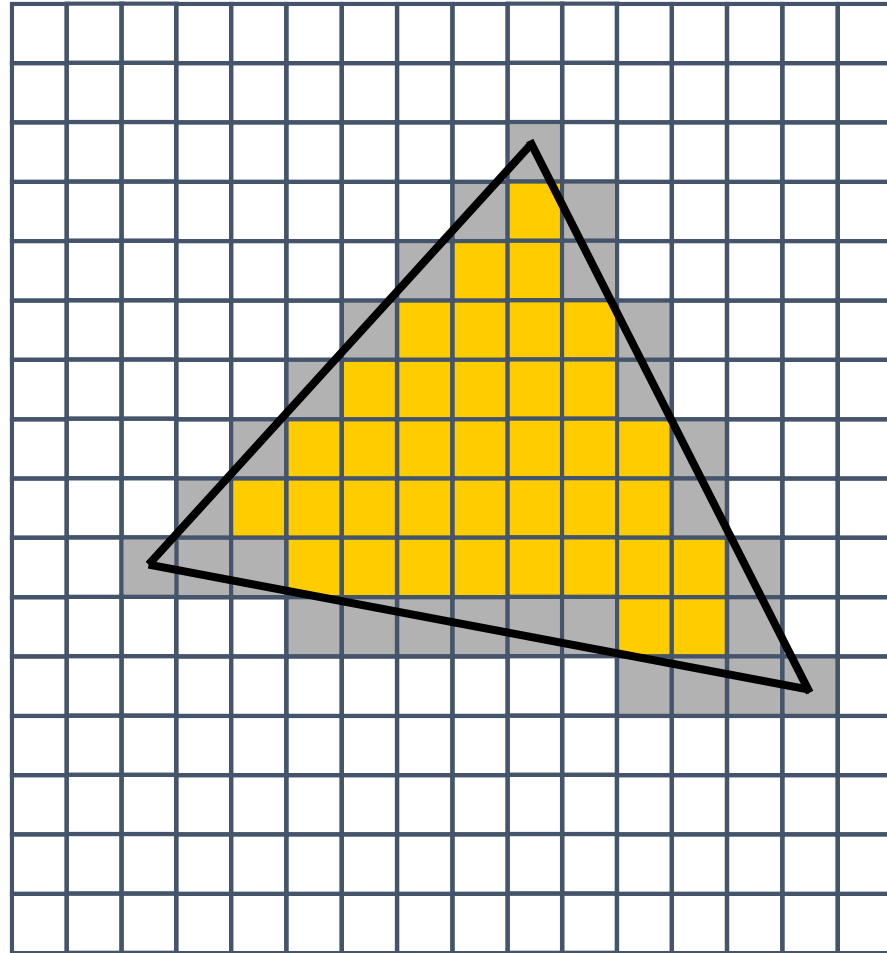


# SCAN CONVERSION

- Le problème:
  - La ligne est infiniment mince, mais l'image a une résolution
  - Le processus peut produire
    - Marches d'escalier
    - Aliassage
  - L'un des problèmes fondamentaux en infographie

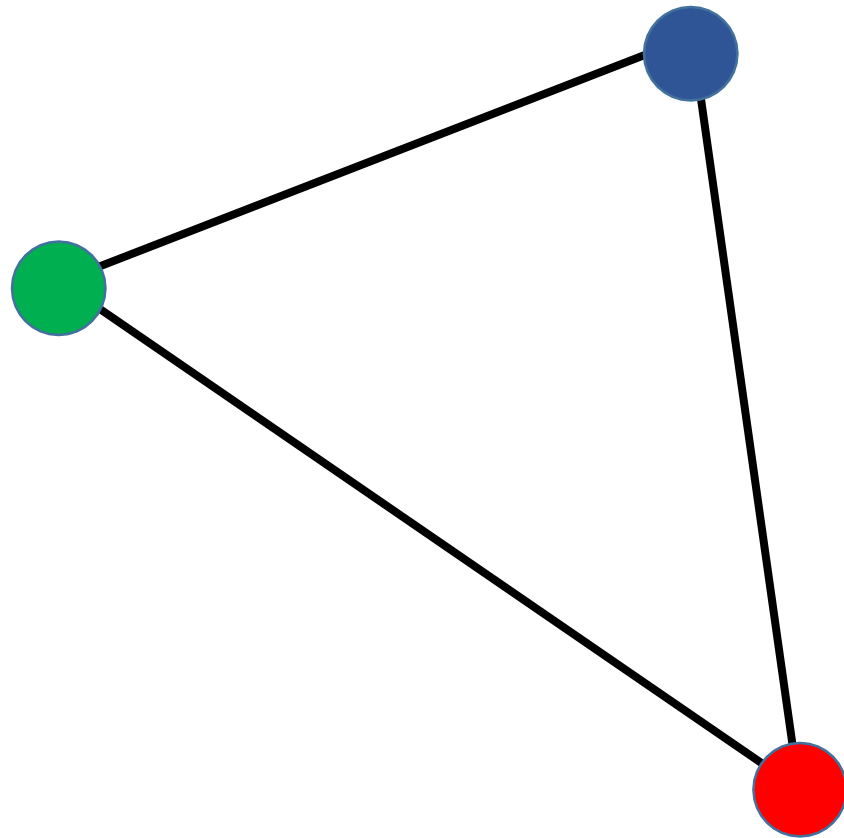


# SCAN CONVERSION



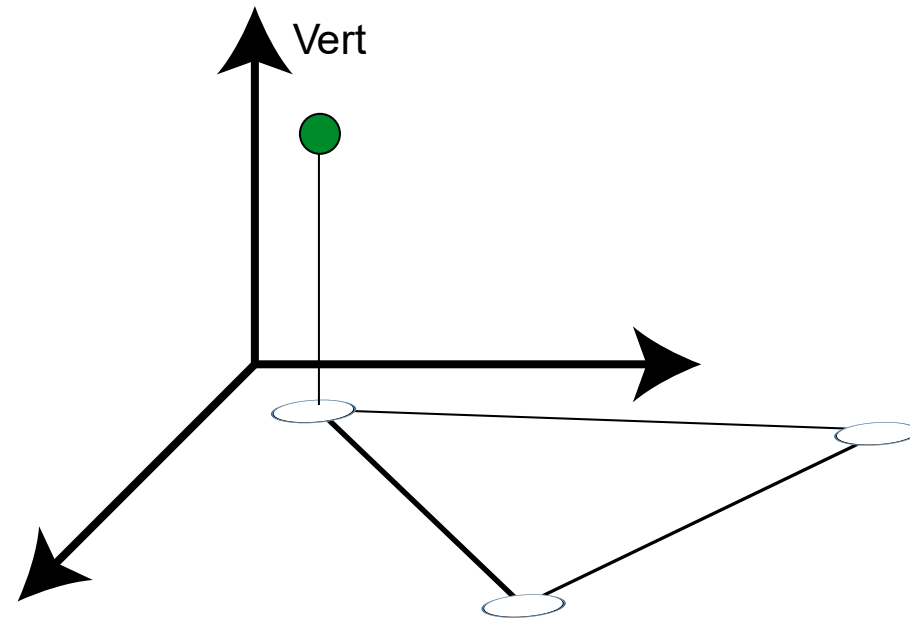
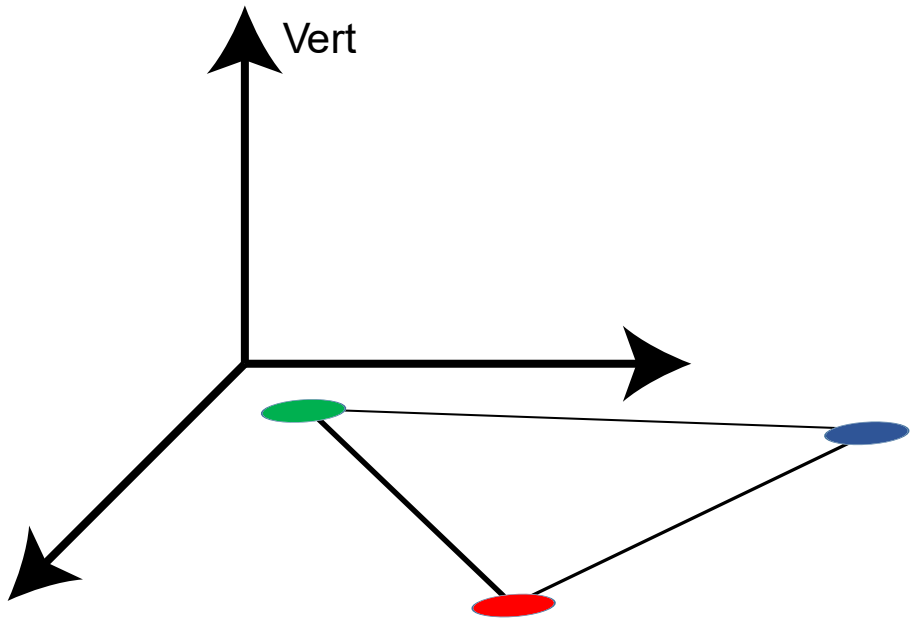
# L'INTERPOLATION DE COULEURS

On assume une couleur par sommet de chaque triangle  
Comment remplir les couleurs entre elles?



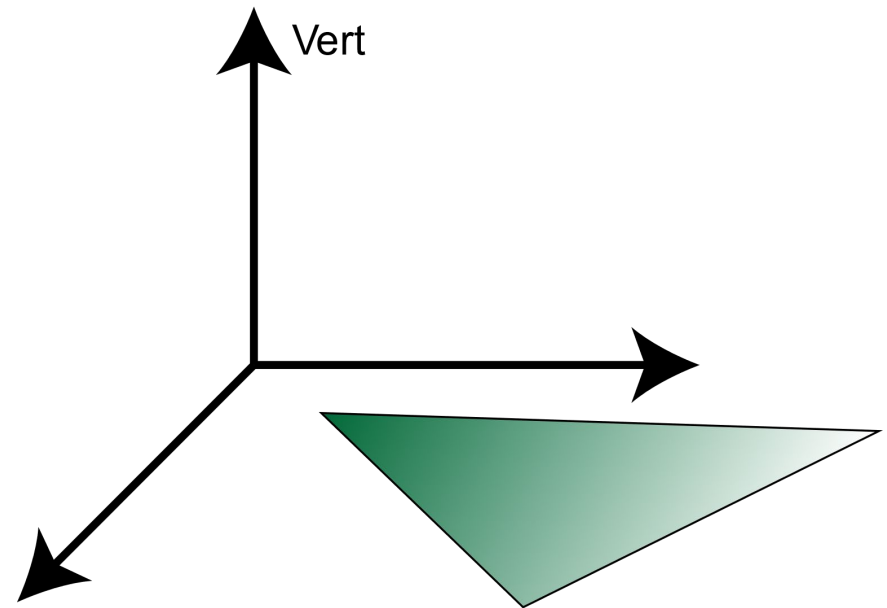
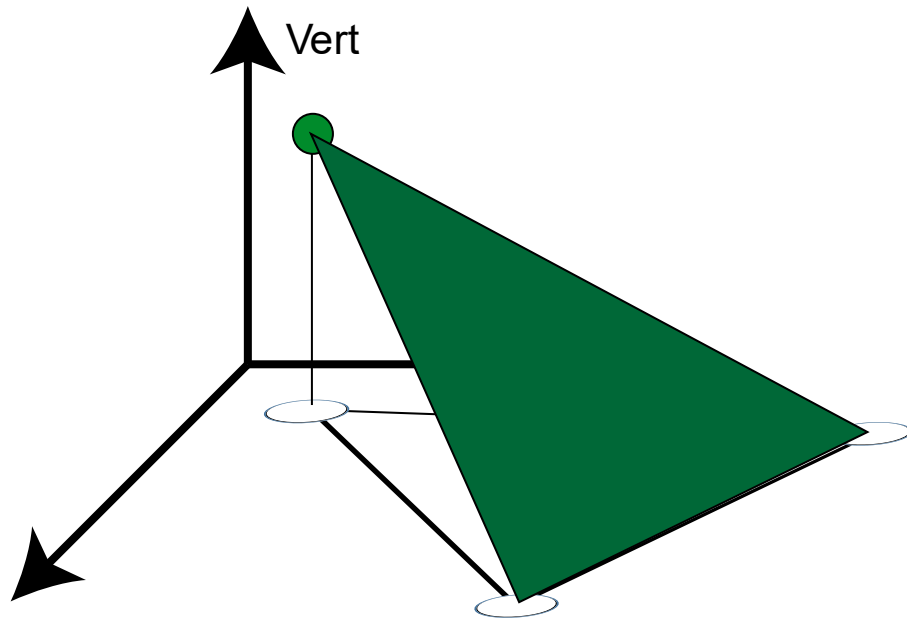
# L'INTERPOLATION DE COULEURS

On assume une couleur par sommet de chaque triangle  
Comment remplir les couleurs entre elles?



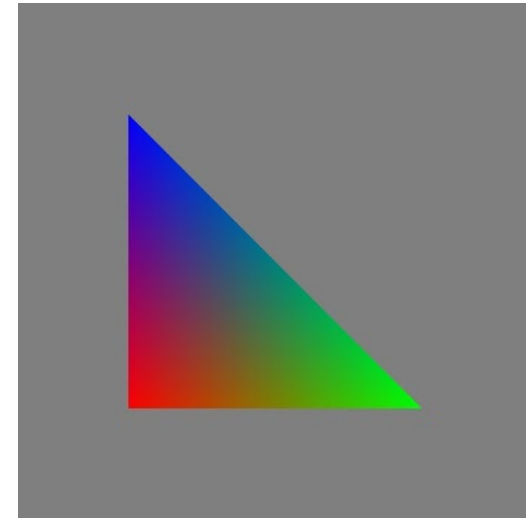
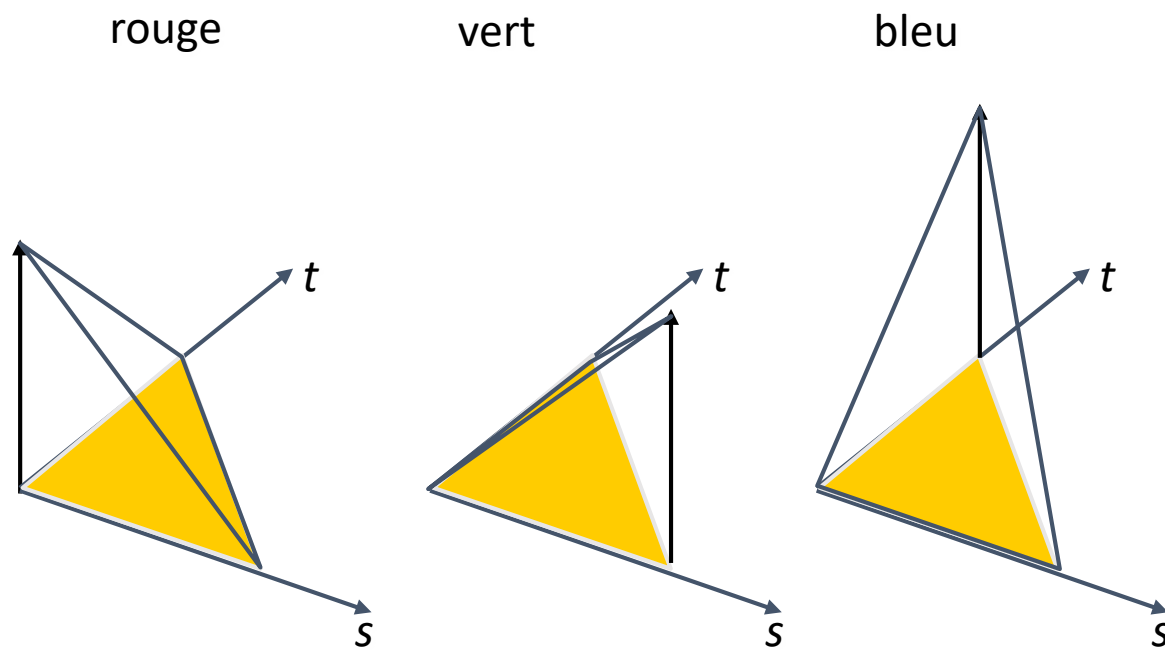
# L'INTERPOLATION DE COULEURS

L'intepolation linéaire

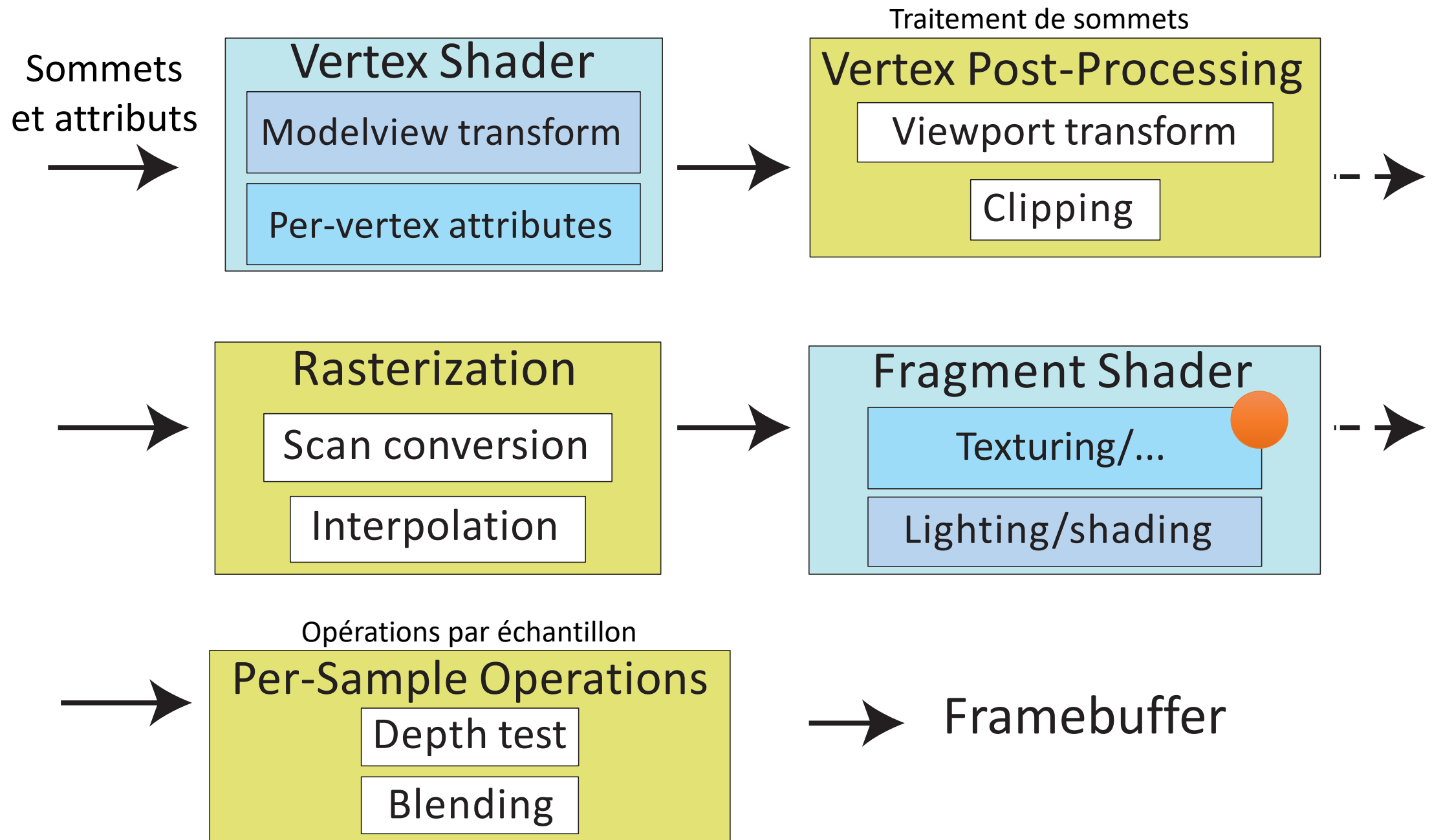


# L'INTERPOLATION DE COULEURS

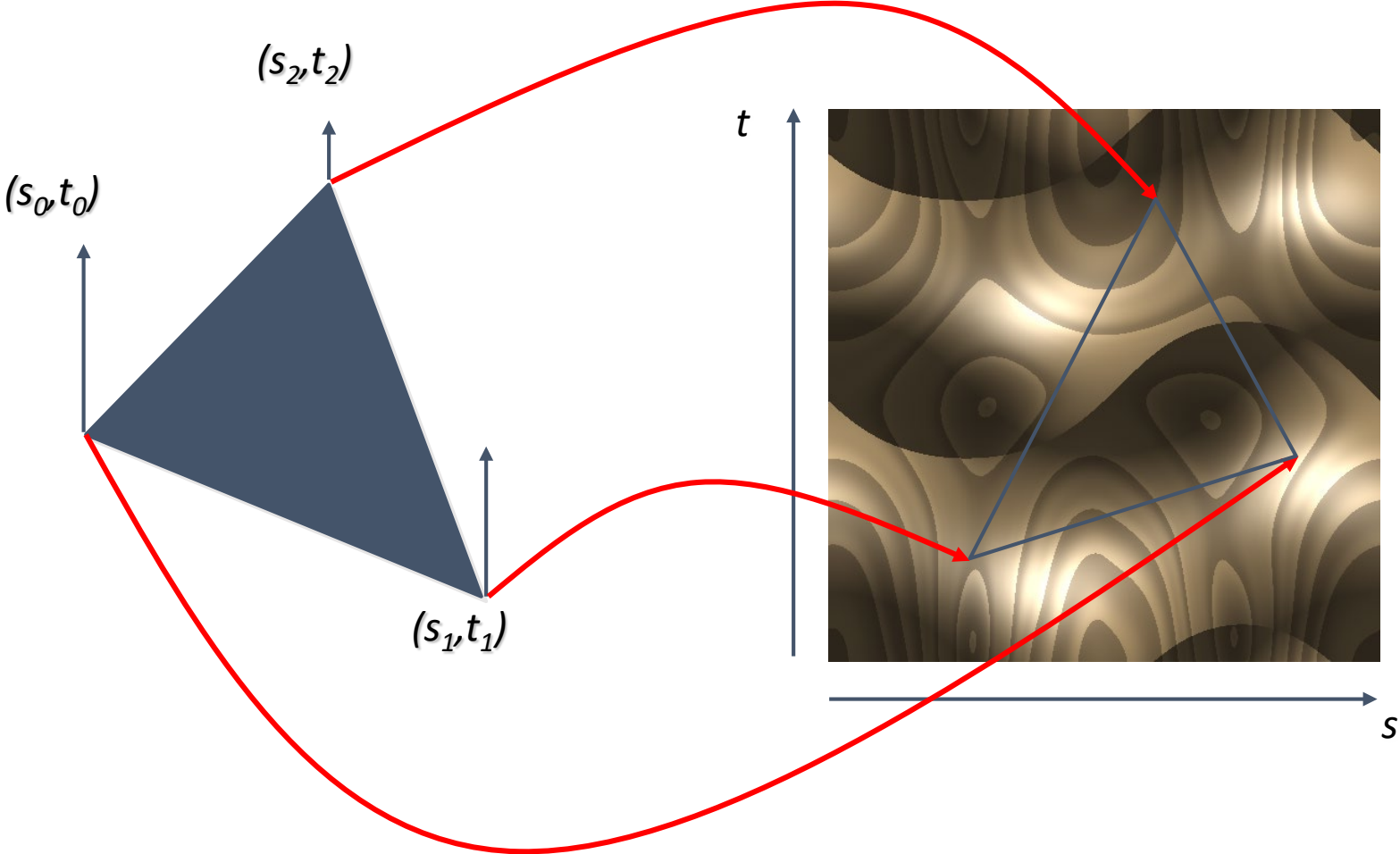
- Exemple:



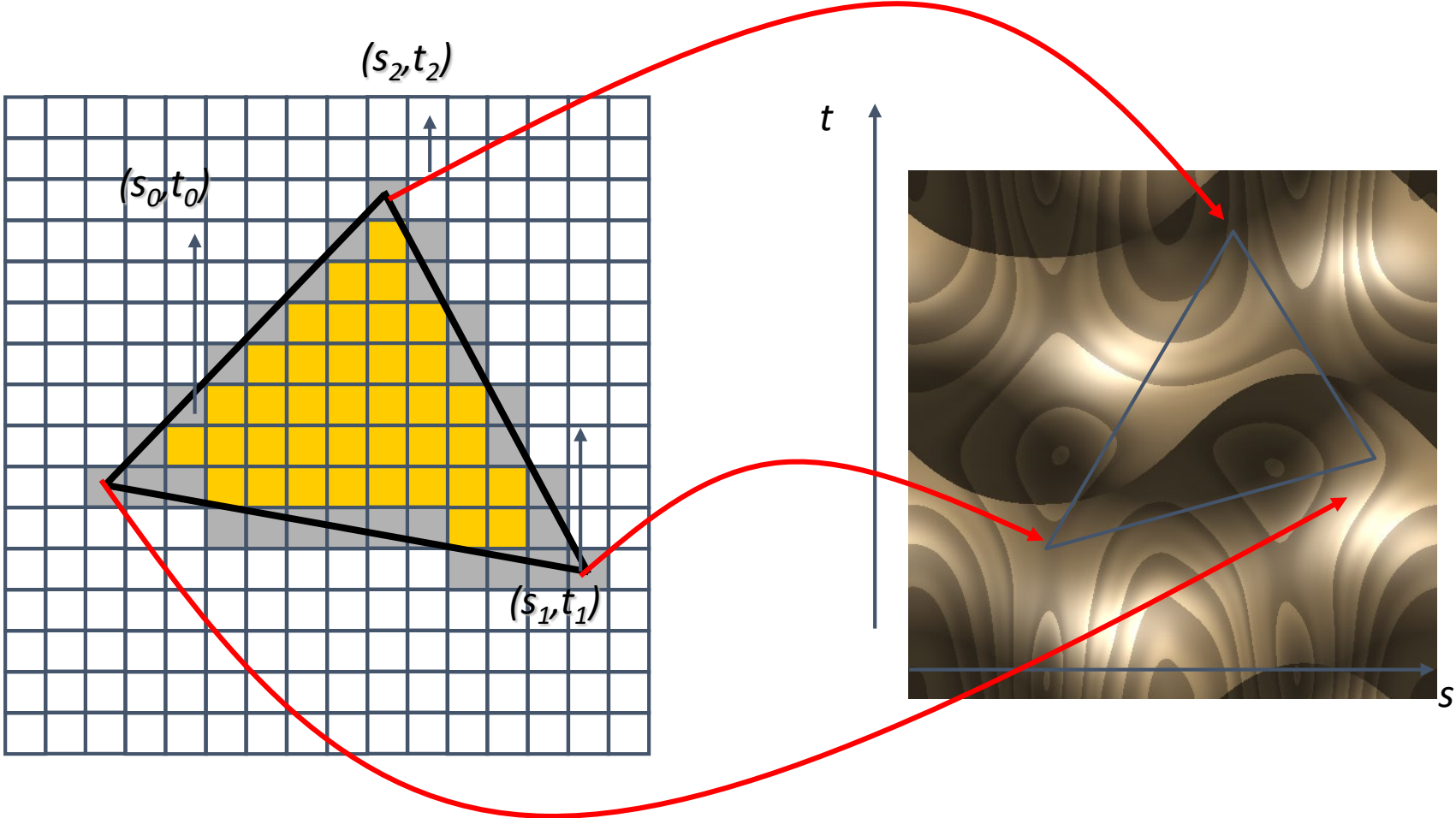
# PIPELINE: PLUS DE DÉTAILS



# TEXTURES



# TEXTURES





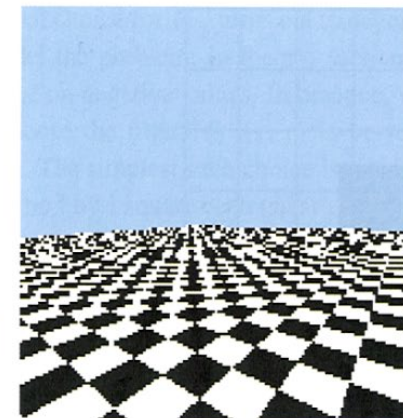
# TEXTURES



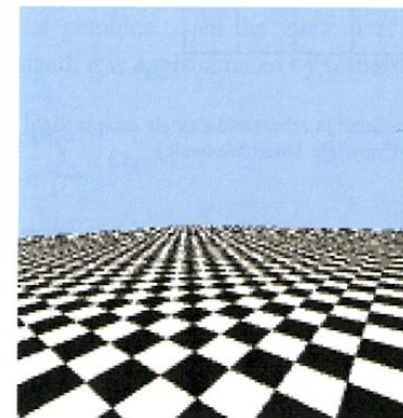
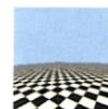
# TEXTURES

- Les problèmes:
  - Calculer une fonction de mapping 1-1 entre le maillage et la texture
  - Comment mapper un pixel de texture (*texel*) à un pixel d'écran?
    - Une texture peut apparaître très déformée
    - Magnification/minification des textures
  - Filtrage de textures
  - Réduire l'aliassage (*anti-aliasing*)

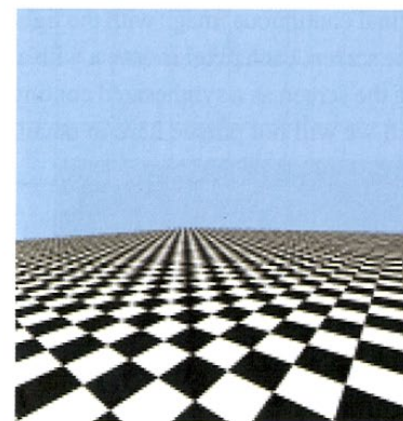
Aliasing



Anti-aliasing  
(multi-sampling)



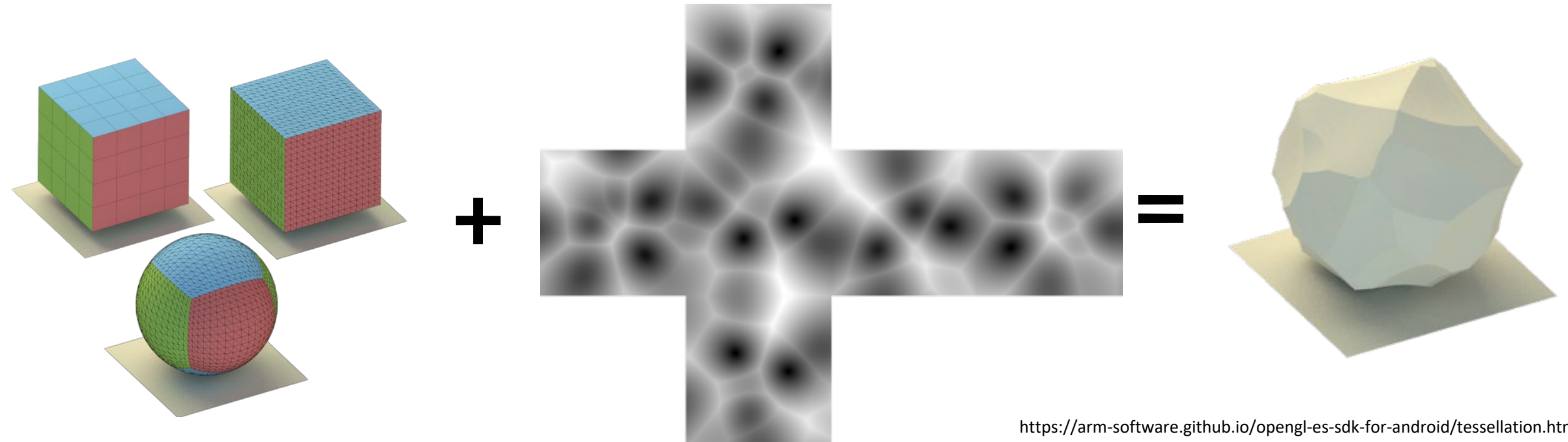
Anti-aliasing  
(super-sampling)



# MAPPING DÉPLACEMENT

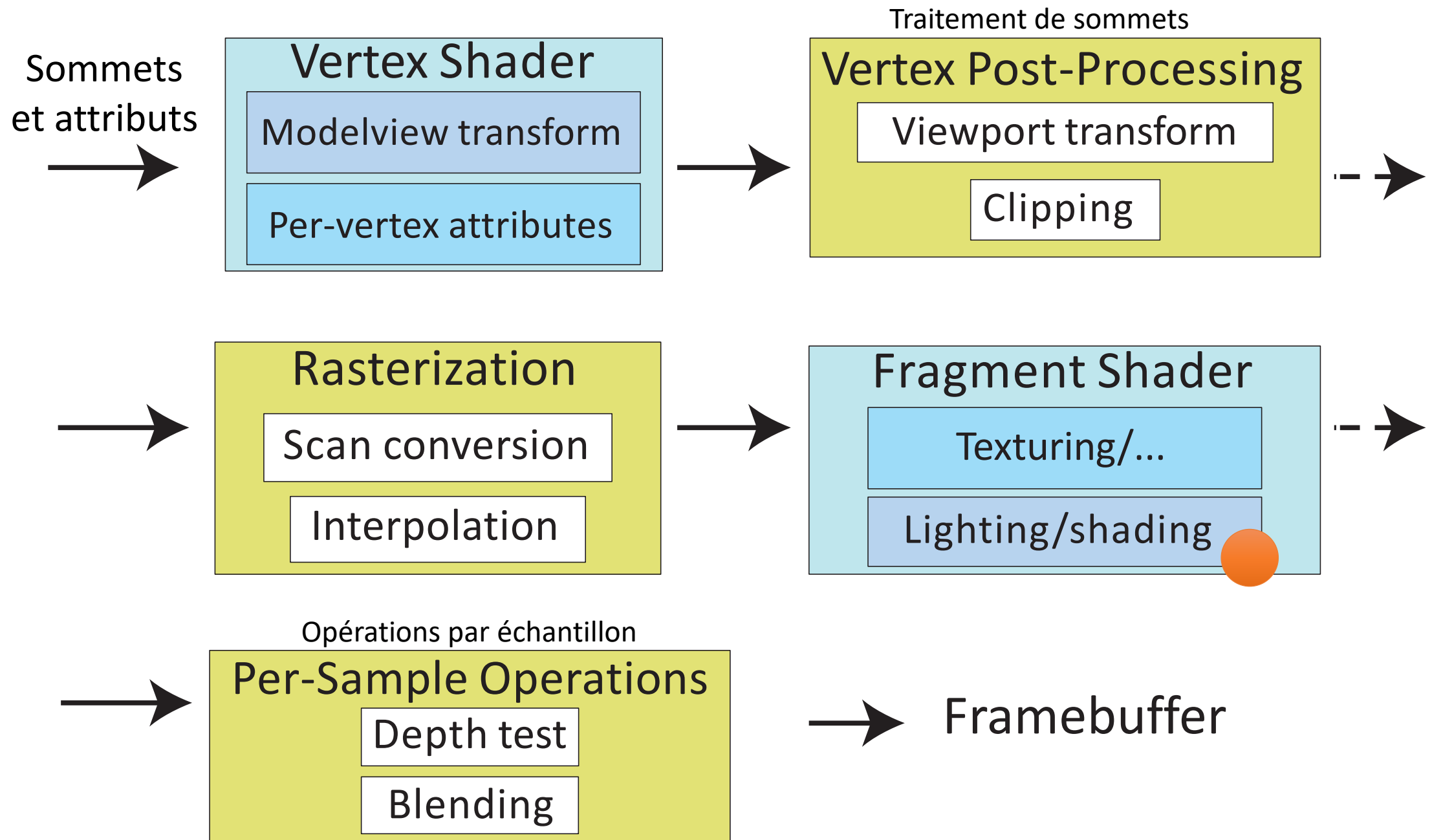


# MAPPING DÉPLACEMENT

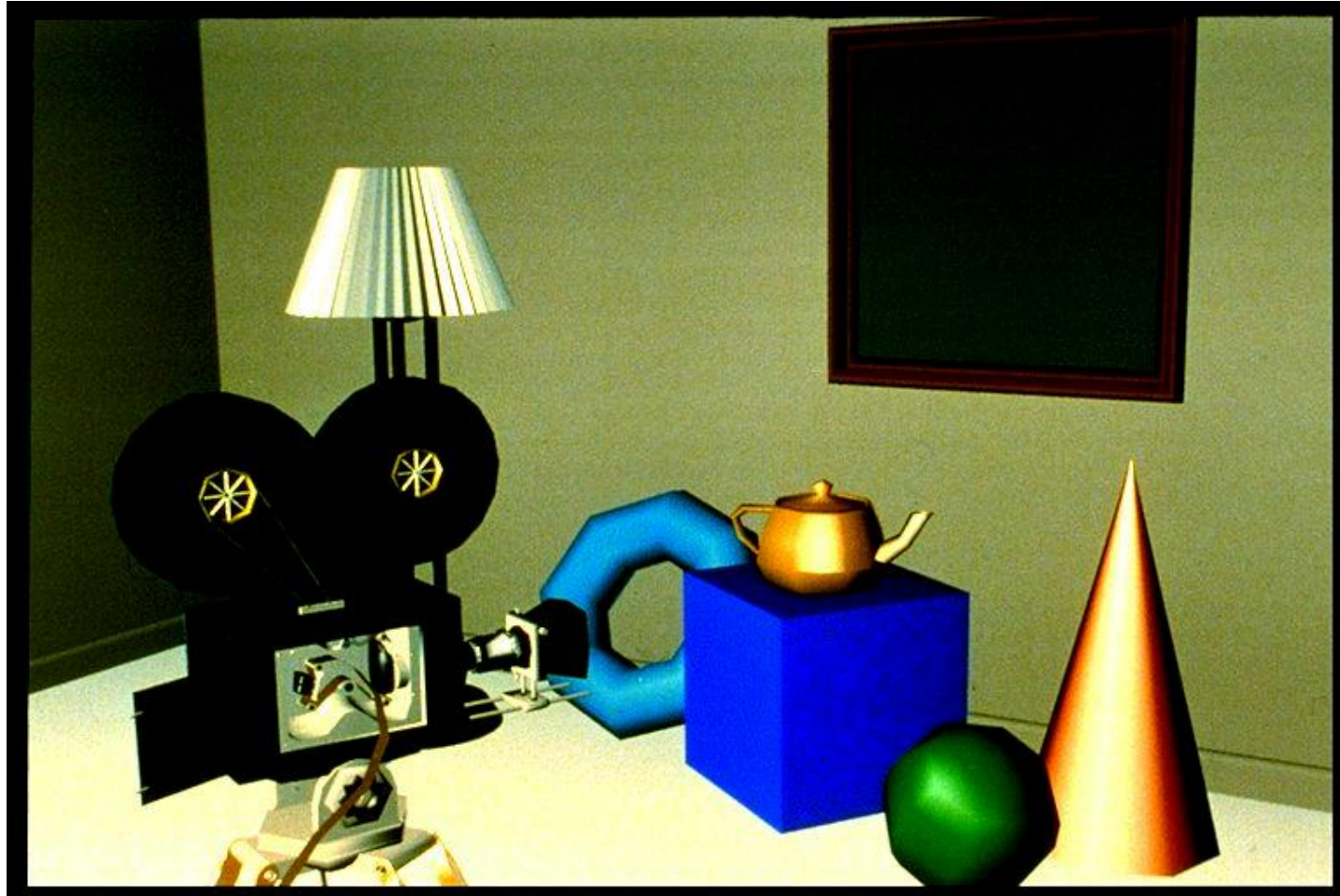


Normalement fait dans un *vertex shader*

# PIPELINE: PLUS DE DÉTAILS



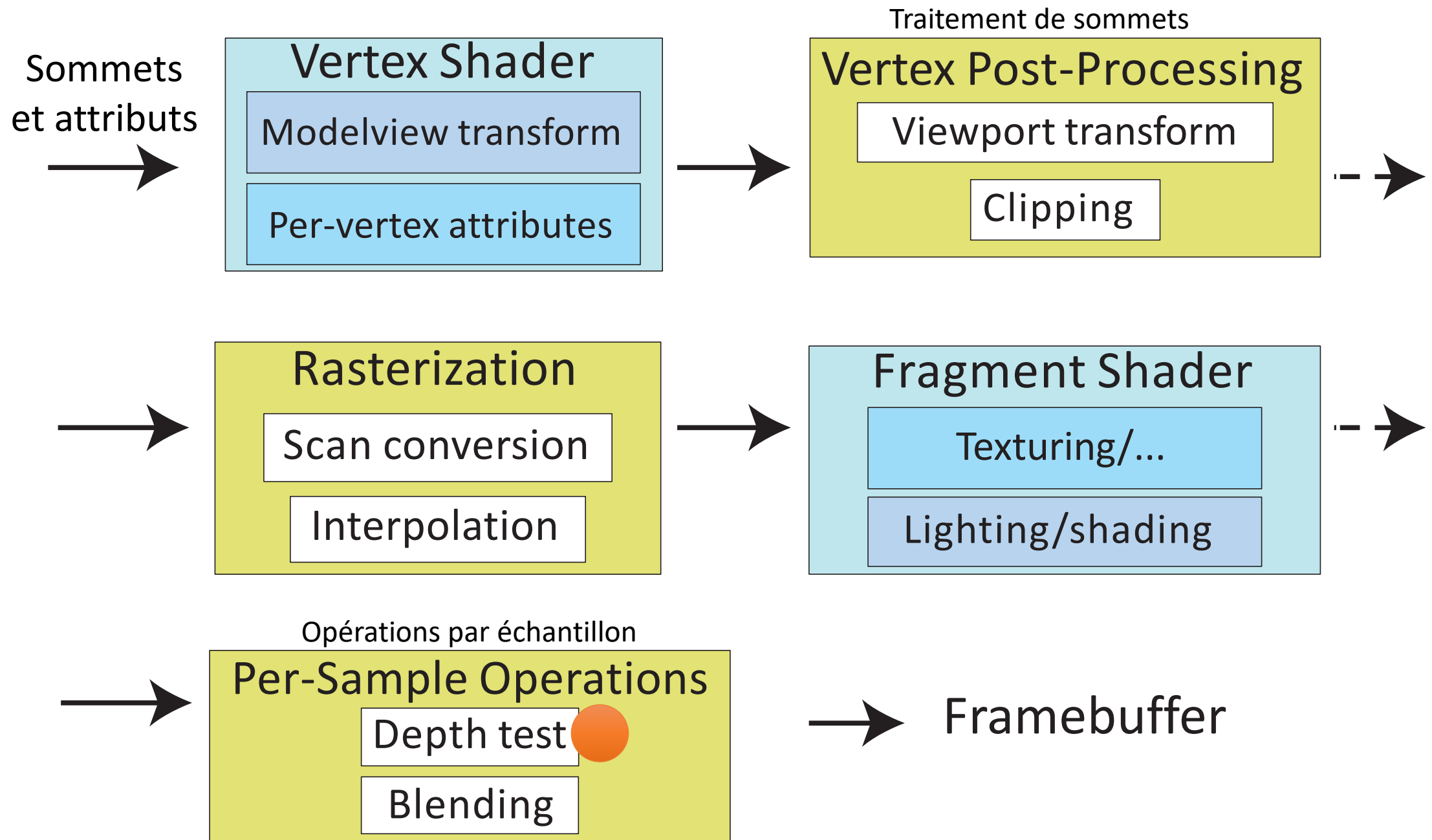
# ÉCLAIRAGE



# ÉCLAIRAGE COMPLEXE

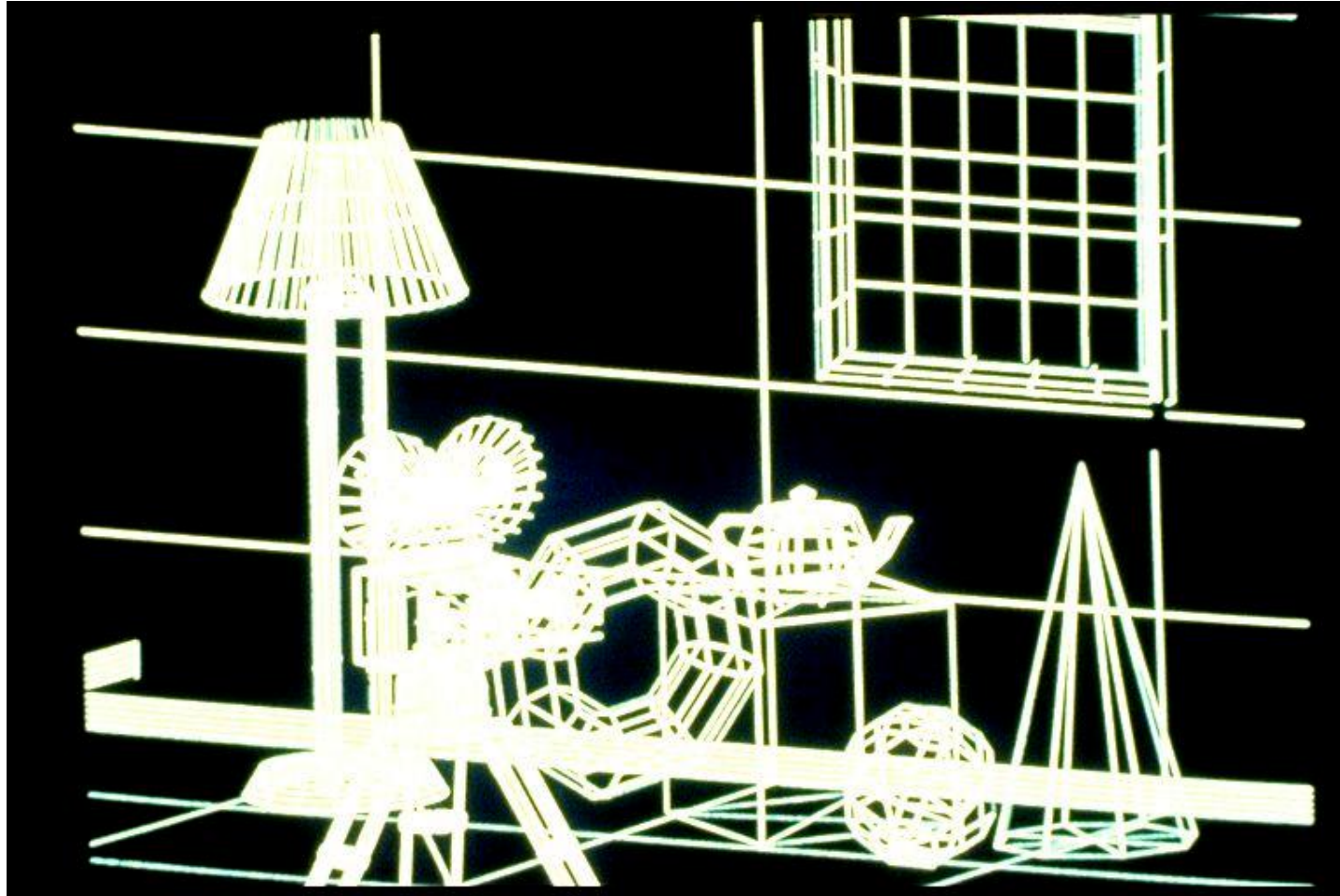


# PIPELINE: PLUS DE DÉTAILS

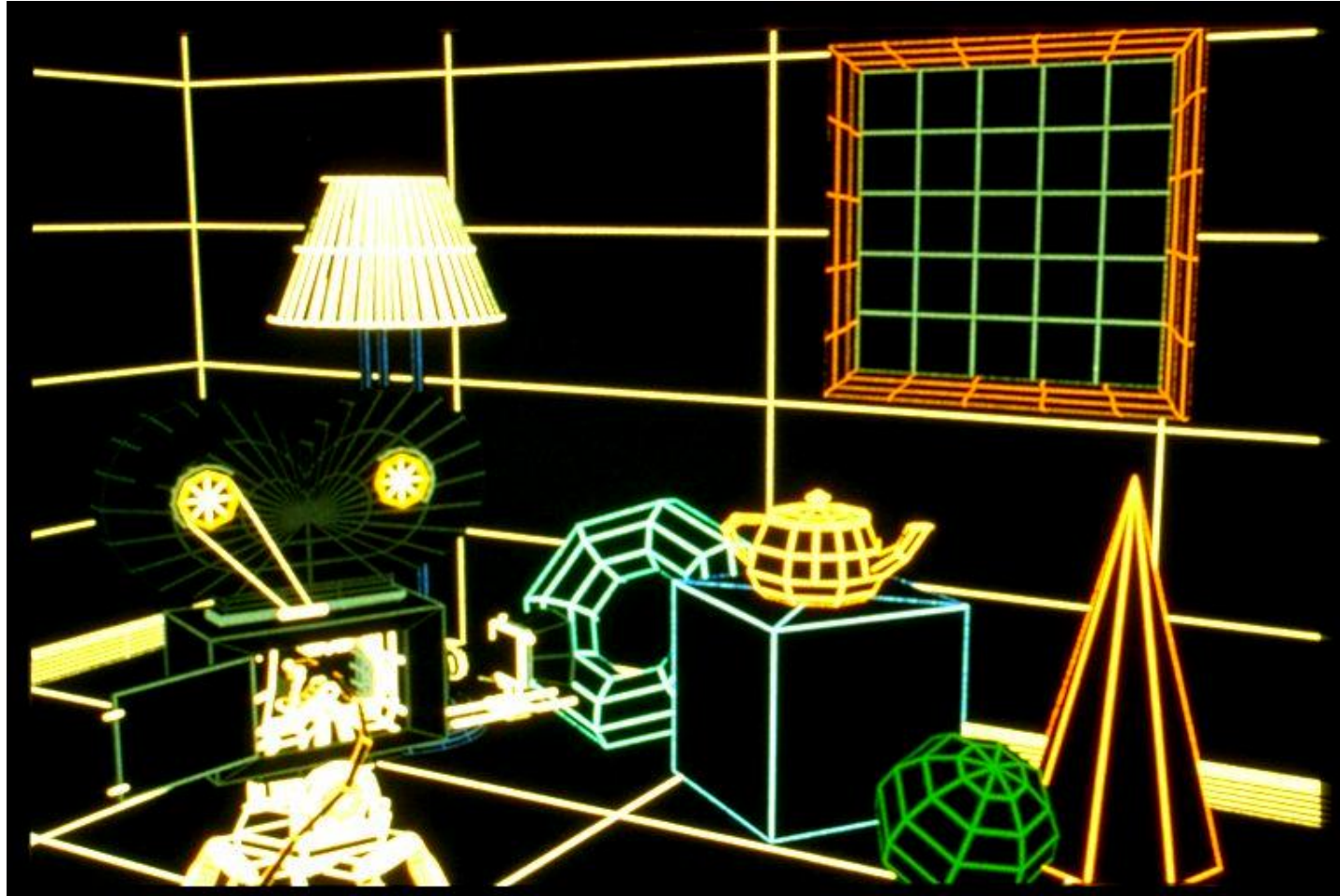




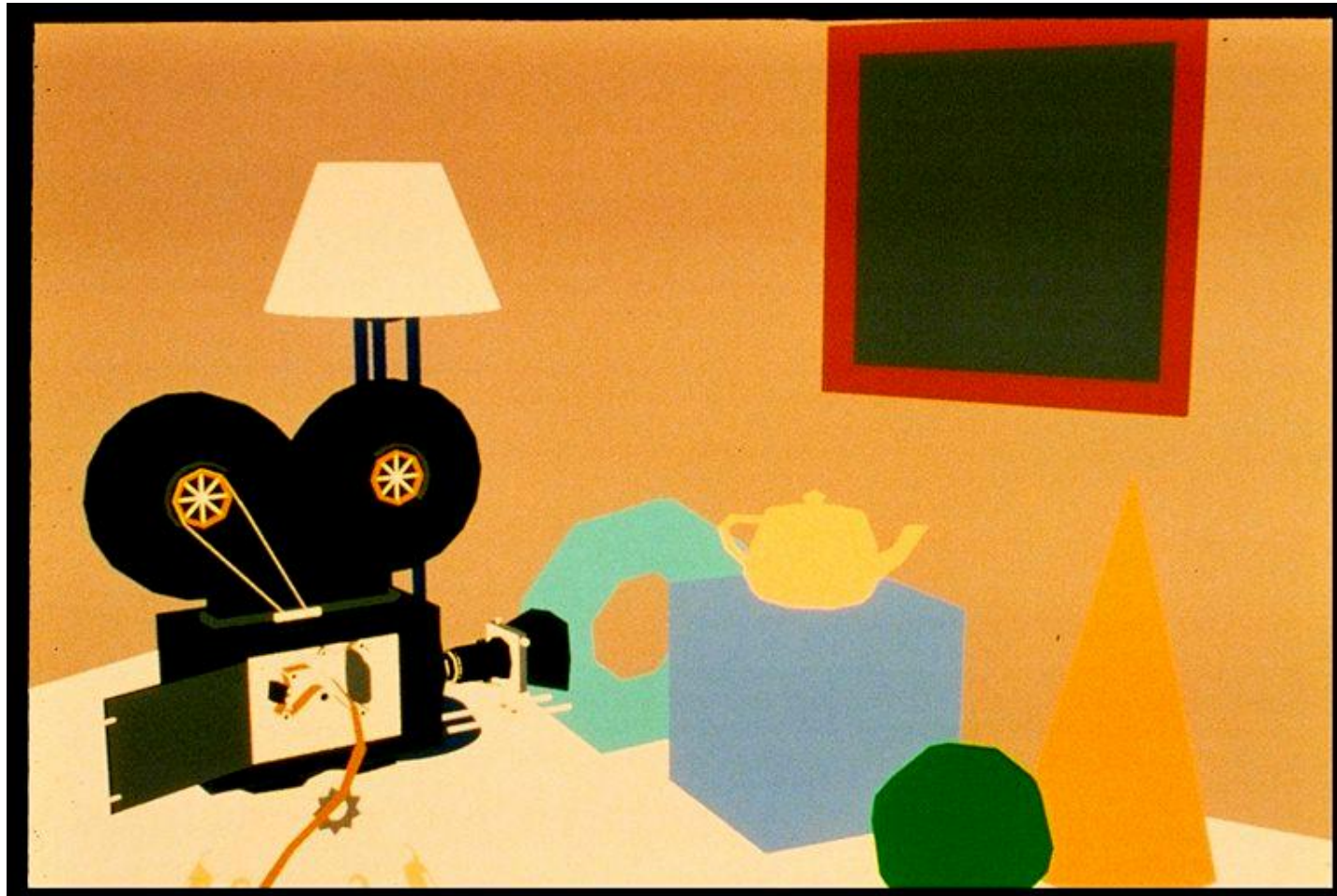
# SANS ÉLIMINATION DES SURFACES CACHÉES



# AVEC ÉLIMINATION DES SURFACES CACHÉES



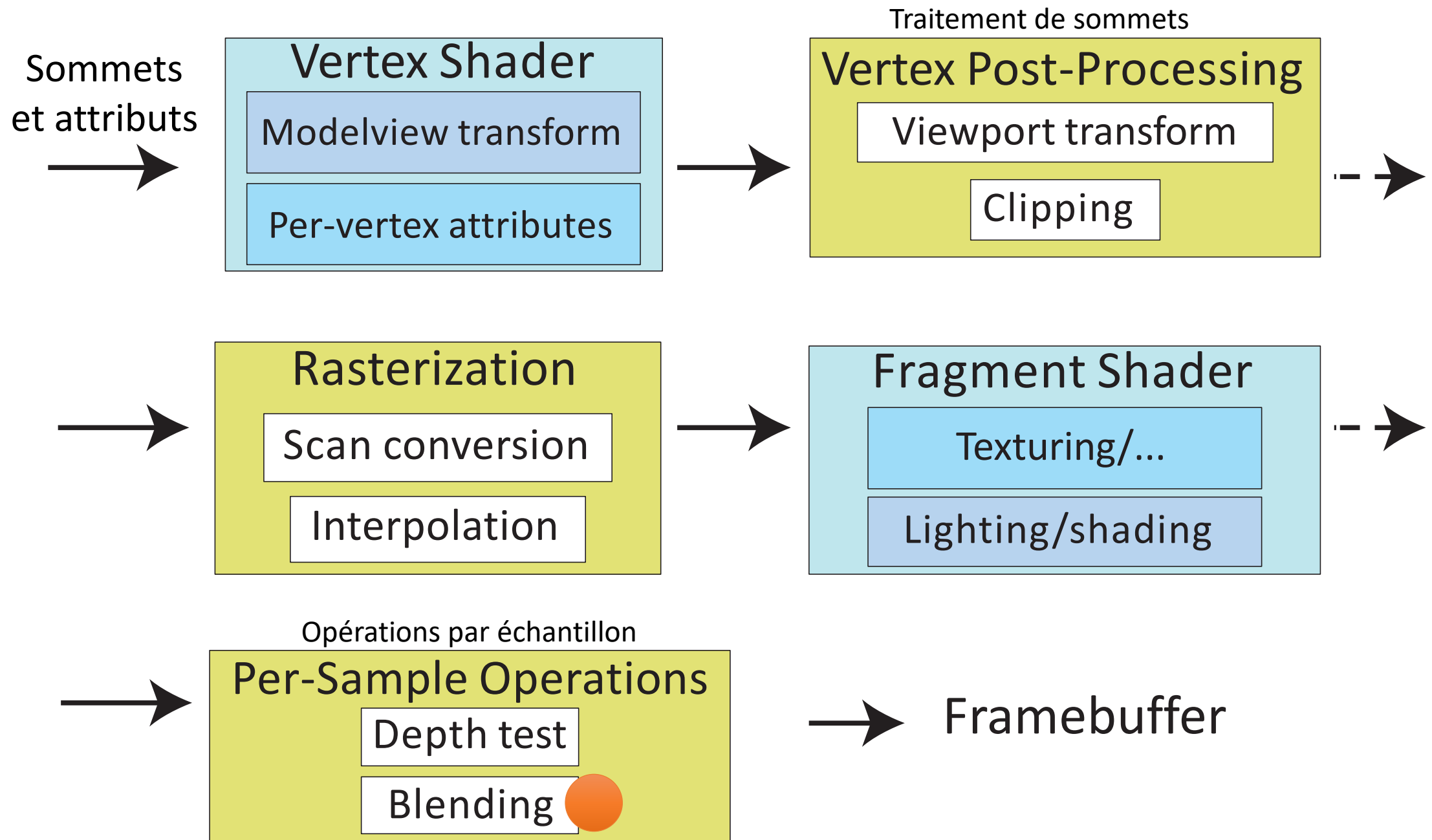
# AVEC ÉLIMINATION DES SURFACES CACHÉES



# ÉLIMINATION DES SURFACES CACHÉES

- Déterminer les surfaces visibles ou éliminer les surfaces cachées
- Deux classes d'algorithmes
  - En espace image
    - Tampon de profondeur (z-buffer)
  - En espace objet
    - E.g. ordre des intersections du rayon

# PIPELINE: PLUS DE DÉTAILS



# LE MÉLANGE DES COULEURS (BLENDING)

- Blending:
  - Fragment → pixels
  - Dessiner du plus éloigné au plus rapproché
  - Sans blending → remplacer la dernière couleur
  - Avec blending: combiner les anciennes & les nouvelles valeurs avec certaines opérations arithmétiques
- Frame Buffer : Une partie de mémoire sur le GPU qu'on voit sur l'écran

# RÉFLEXIONS/OMBRES



</PIPELINE>

- Questions?