

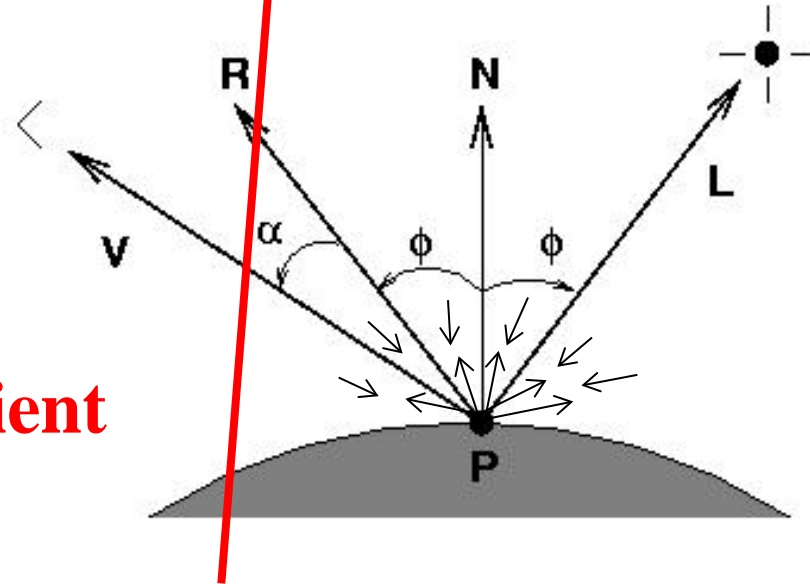
Realisme (II)

- Realisme: Il·luminació (2)
 - **Il·luminació en OpenGL 3.3 (1)**
 - Càlcul de color en vèrtexs (en el Vertex Shader)
 - Shading de polígons
 - Suavitzat d'arestes
 - Il·luminació en OpenGL 3.3 (2)
 - Càlcul de color en fragments

Recordem: Càlcul color en un punt

$$I_{\lambda}(P) = I_{a\lambda}k_{a\lambda} + \sum_i (I_{fi\lambda} k_{d\lambda} \cos(\Phi_i)) + \sum_i (I_{fi\lambda} k_{s\lambda} \cos^n(\alpha_i))$$

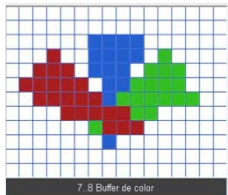
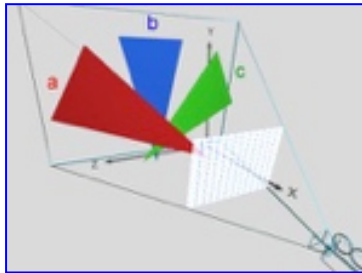
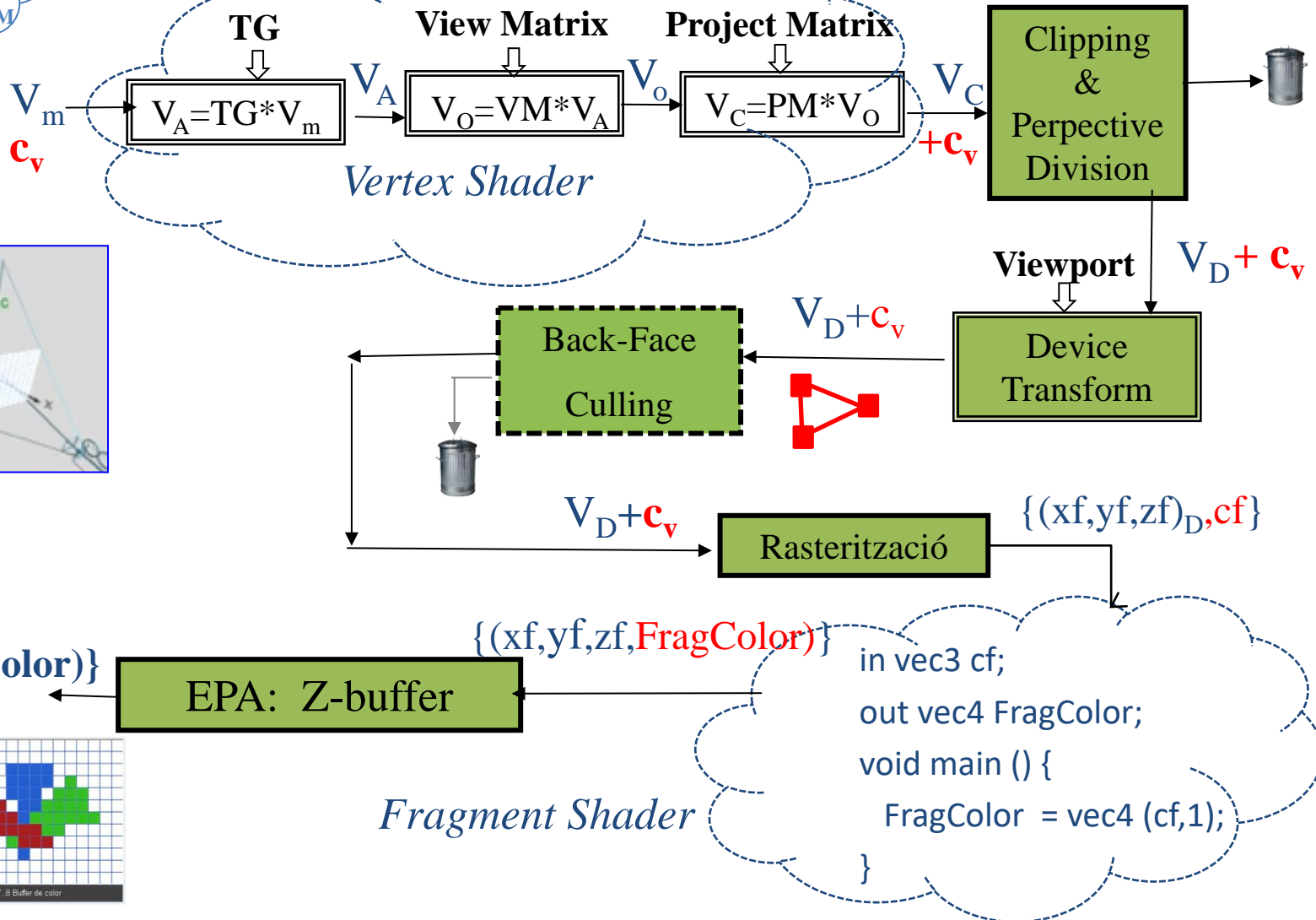
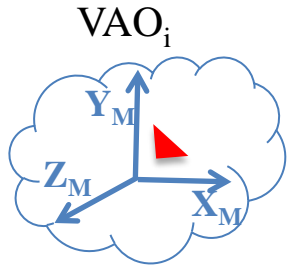
Model Ambient



Model Difús (Lambert) **Model Especular (Phong)**

Procés de visualització

Com calcular el color aplicant fórmula de models empírics?



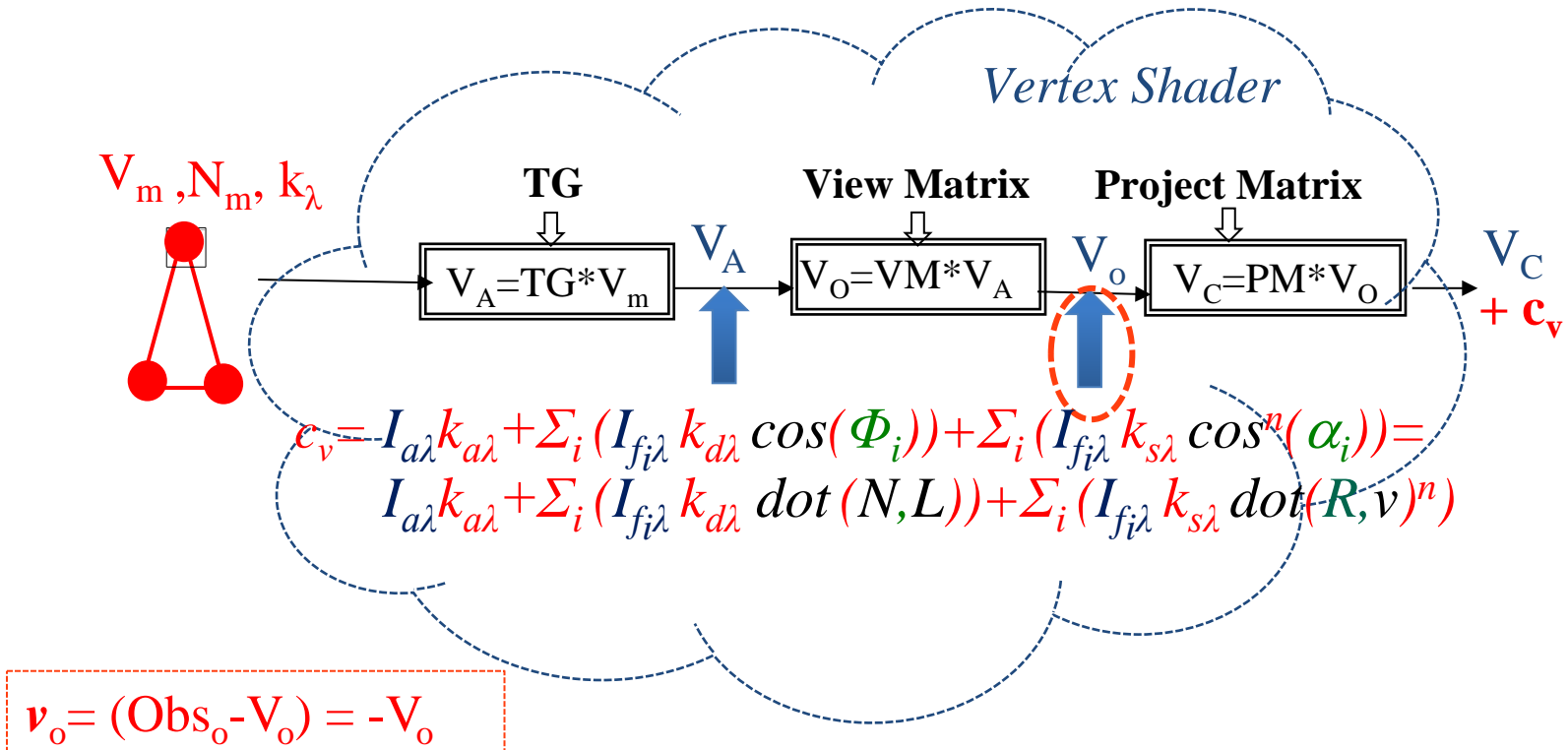
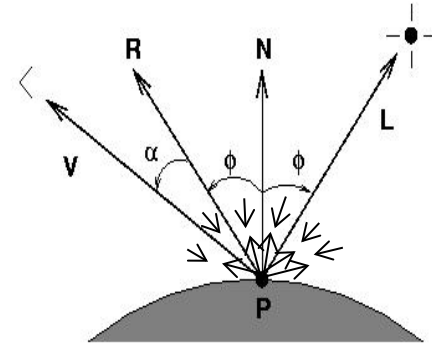
Càlcul del color per vèrtex en el Vèrtex Shader (1)

Atributs per cada model:

- Coordenades (V), normal (N) i constants de material (k_λ) per vèrtex en VBOs del seu VAO

Uniforms:

- Fonts de llum actives => color, posició
- Llum ambient



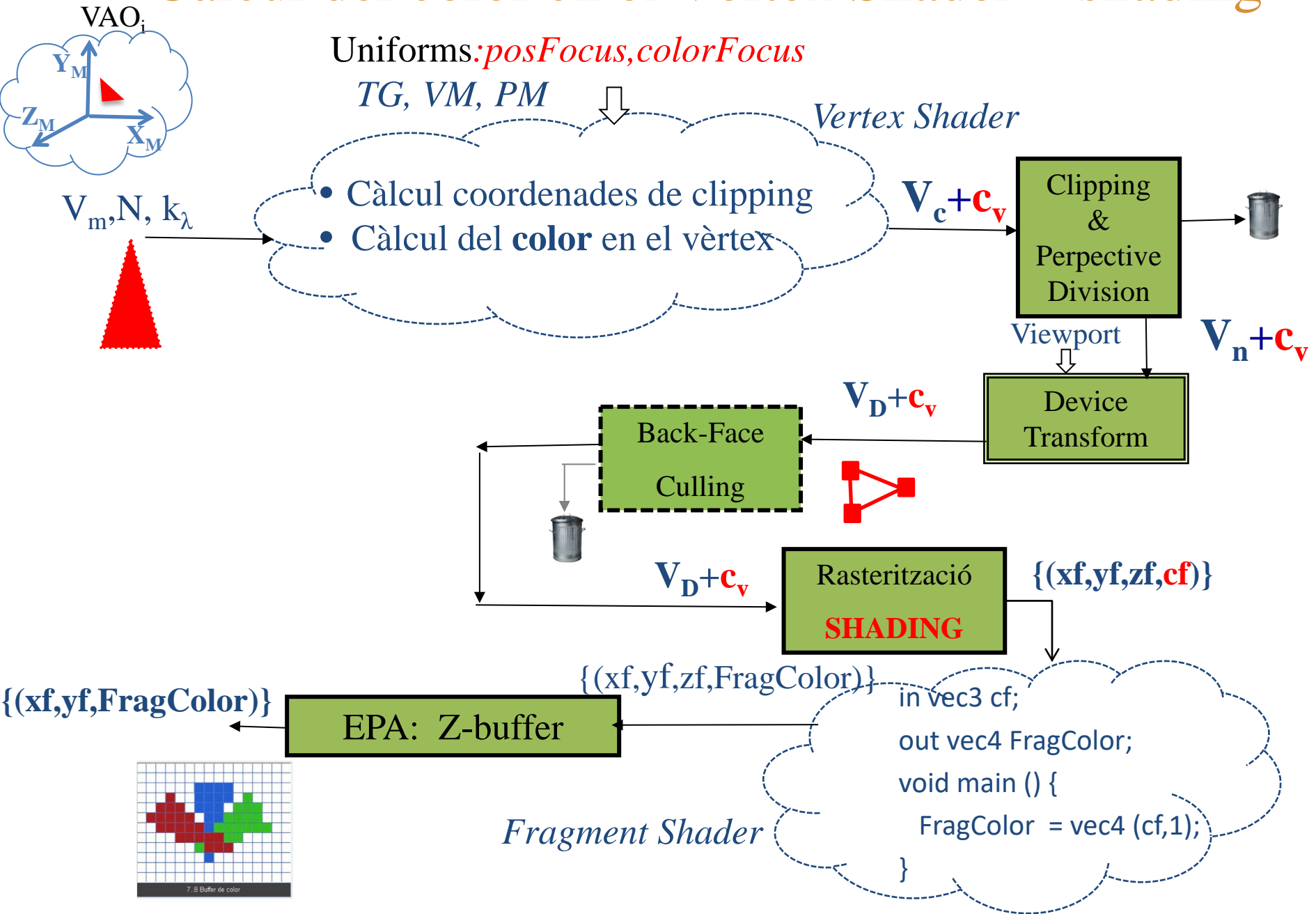
Càlcul del color per vèrtex en el Vèrtex Shader (2)

El càlcul el farem per cada vèrtex (al Vertex Shader)

I el farem en SCO, per tant:

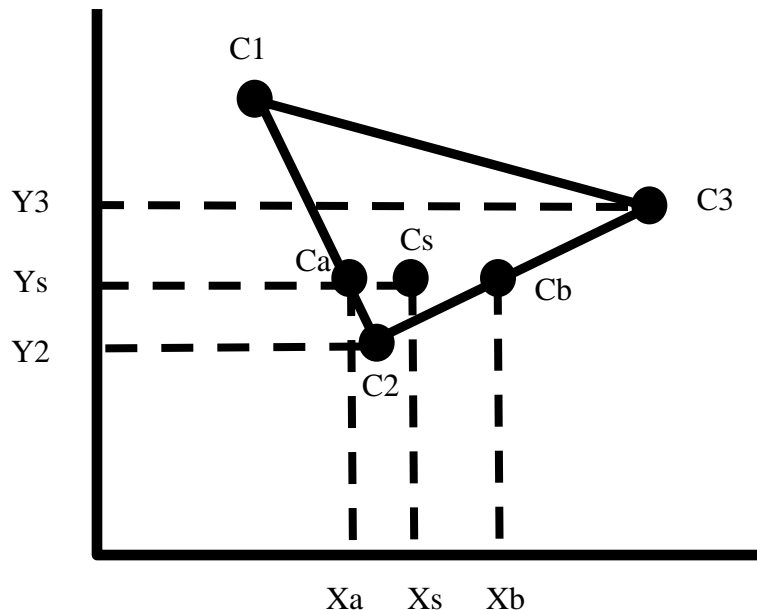
- Cal passar la posició del vèrtex a SCO
 - multiplicat per (**view * TG**)
- Cal passar el vector normal a SCO
 - multiplicat per la matriu **inversa** de la **transposada** de (**view * TG**), -li direm **NormalMatrix**-
*mat3 NormalMatrix = inverse (transpose (mat3 (view * TG)))*
- La posició del focus de llum també ha d'estar en SCO
 - Multiplicat per **view** (si no la tenim directament en SCO)

Càlcul del color en el Vèrtex Shader + shading



Recordem: Shading (colorat) de polígons

- Colorat Constant \equiv **Flat shading** $\rightarrow C_f = C_l$
color uniforme per tot el polígon (funció del color calculat en un vèrtex); cada cara pot tenir diferent color.
- Colorat de Gouraud \equiv **Gouraud shading** \equiv **Smooth shading**

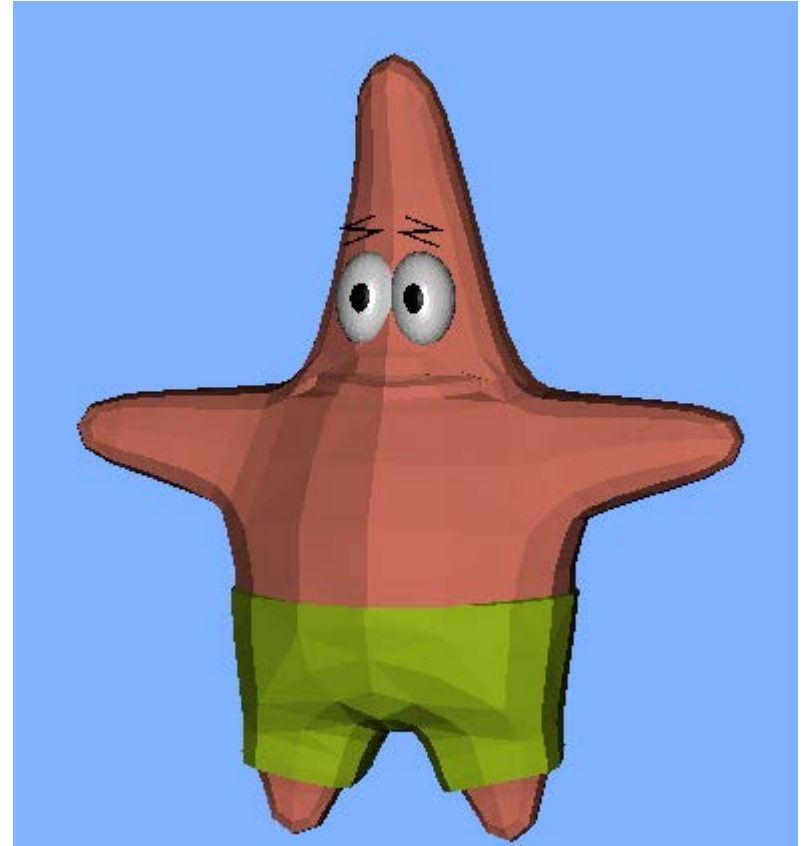
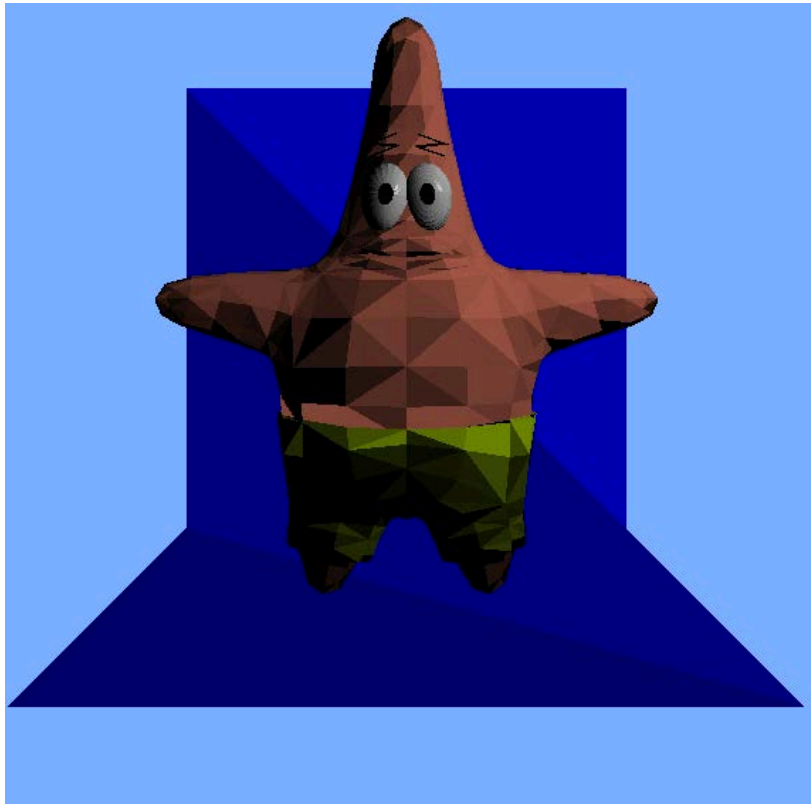


$$C_a = \frac{1}{Y_1 - Y_2} (C_1(Y_s - Y_2) + C_2(Y_1 - Y_s))$$

$$C_b = \frac{1}{Y_3 - Y_2} (C_2(Y_3 - Y_s) + C_3(Y_s - Y_2))$$

$$C_s = \frac{1}{X_b - X_a} (C_a(X_b - X_s) + C_b(X_s - X_a))$$

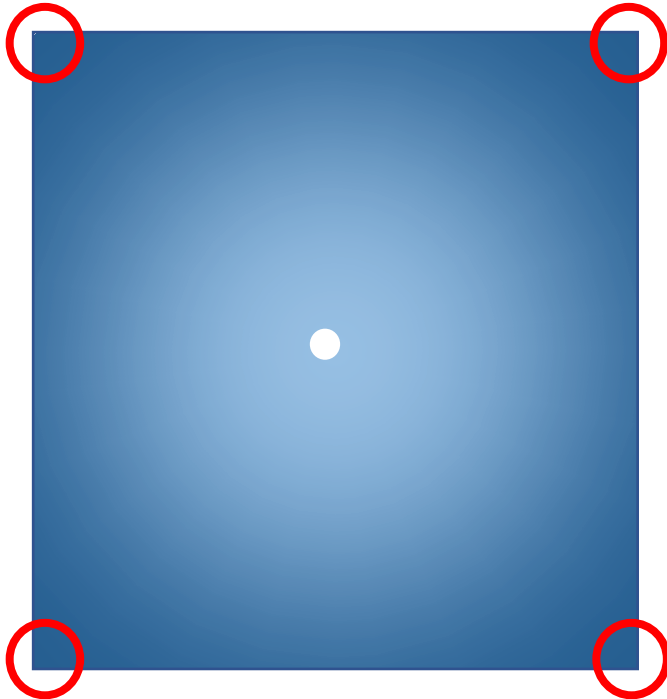
Flat versus Gouraud/smooth Shading



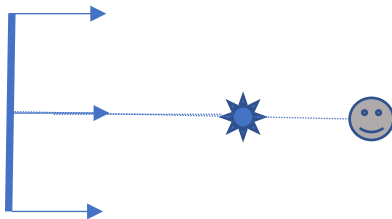
Càlcul color en VS: limitacions

Efecte del càlcul de Lambert i Phong en Vertex Shader:

Com s'hauria de veure



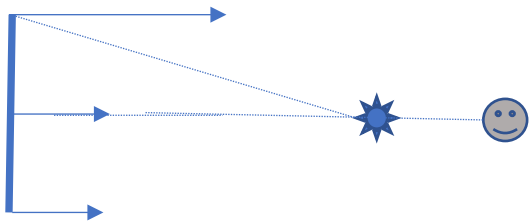
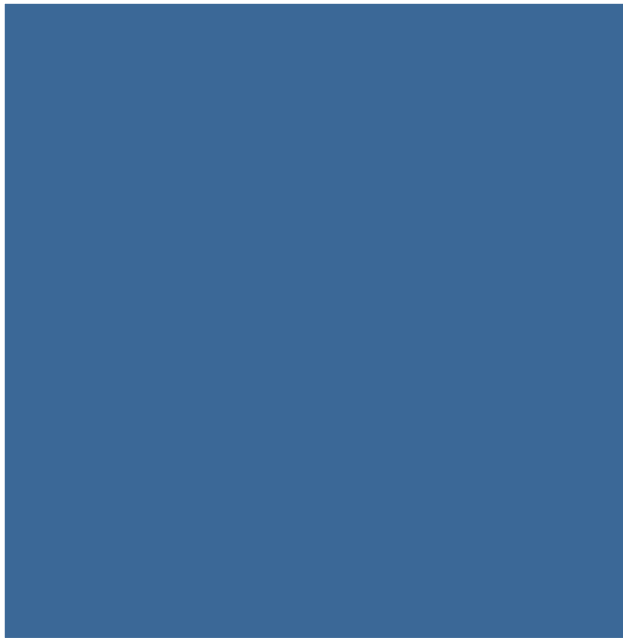
Com es veu



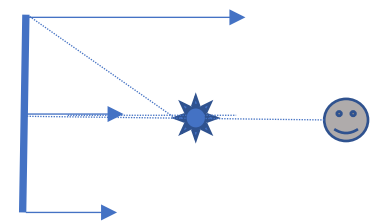
Possible solució: **Discretitzar més**

Càlcul color en VS: limitacions

Què passa si apropem el focus de llum al quadrat?



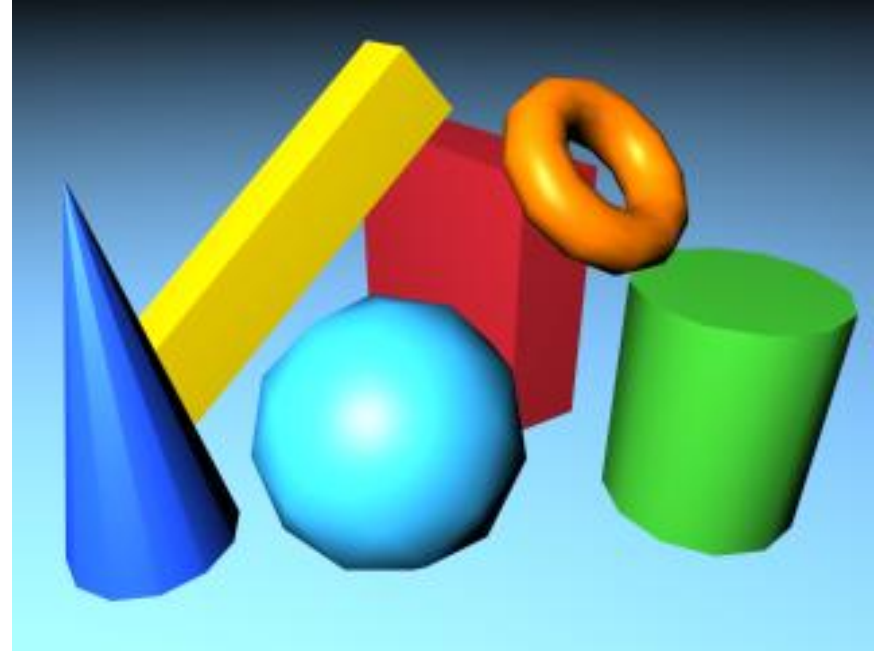
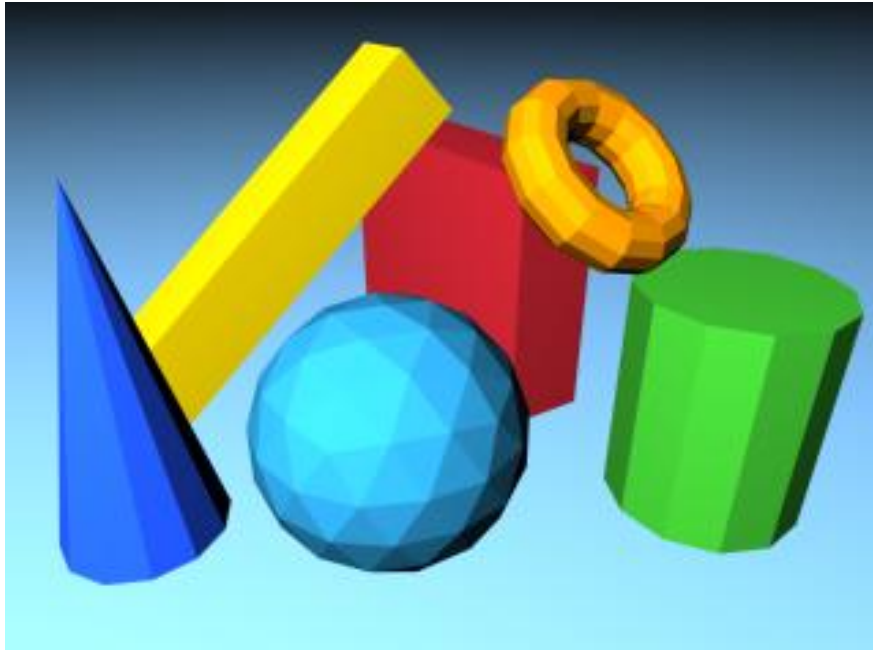
angle ϕ creix



Realisme (II)

- Realisme: Il·luminació (2)
 - Il·luminació en OpenGL 3.3 (1)
 - Càlcul de color en vèrtexs (en el Vertex Shader)
 - Shading de polígons
 - **Suavitzat d'arestes**
 - Il·luminació en OpenGL 3.3 (2)
 - Càlcul de color en fragments

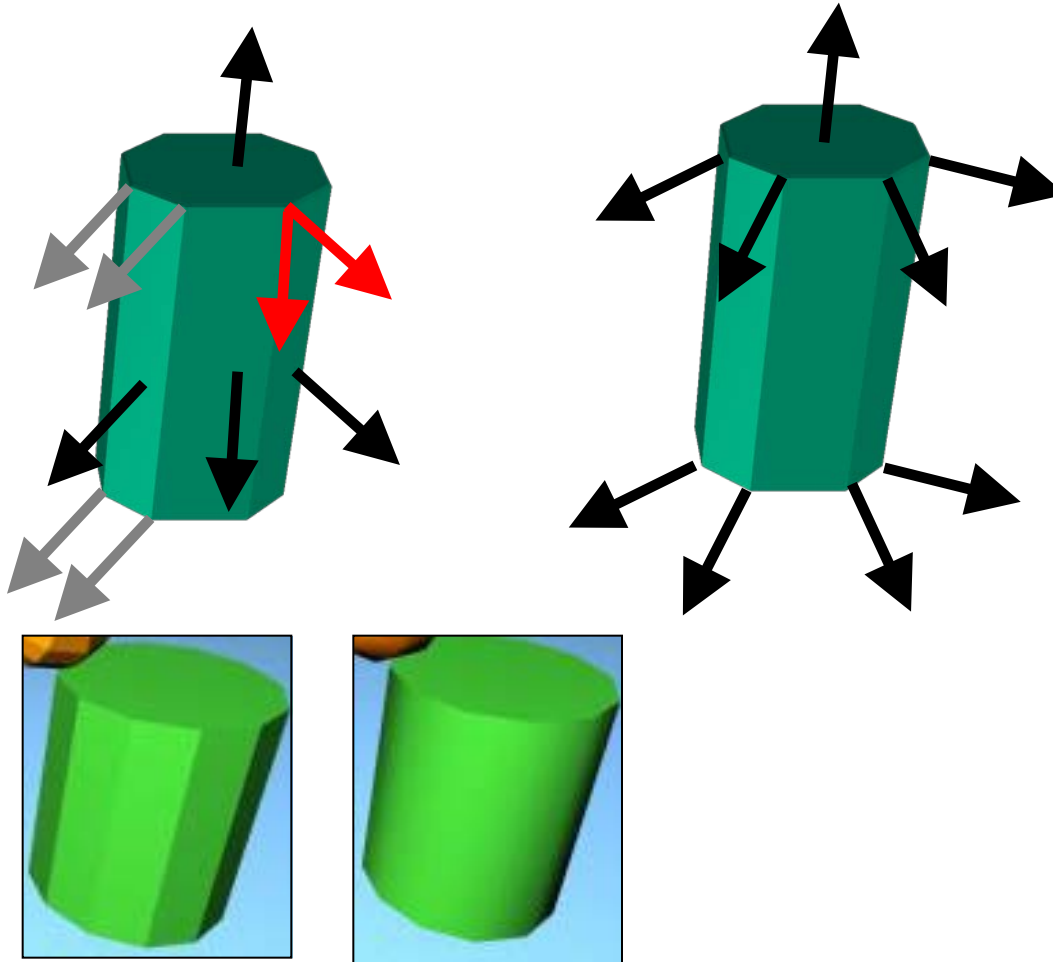
Suavitzat d'arestes (1)



Quin model d'il·luminació i shading s'utilitza?
Per què no es veuen les arestes?
Noteu la forma de les siluetes

Suavitzat d'arestes (2)

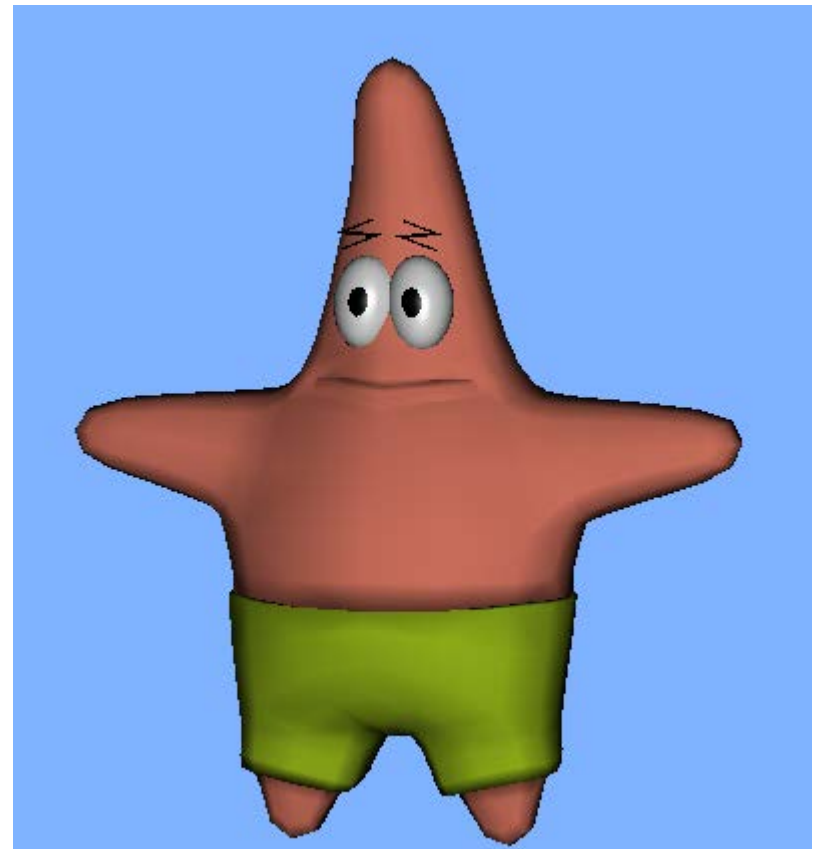
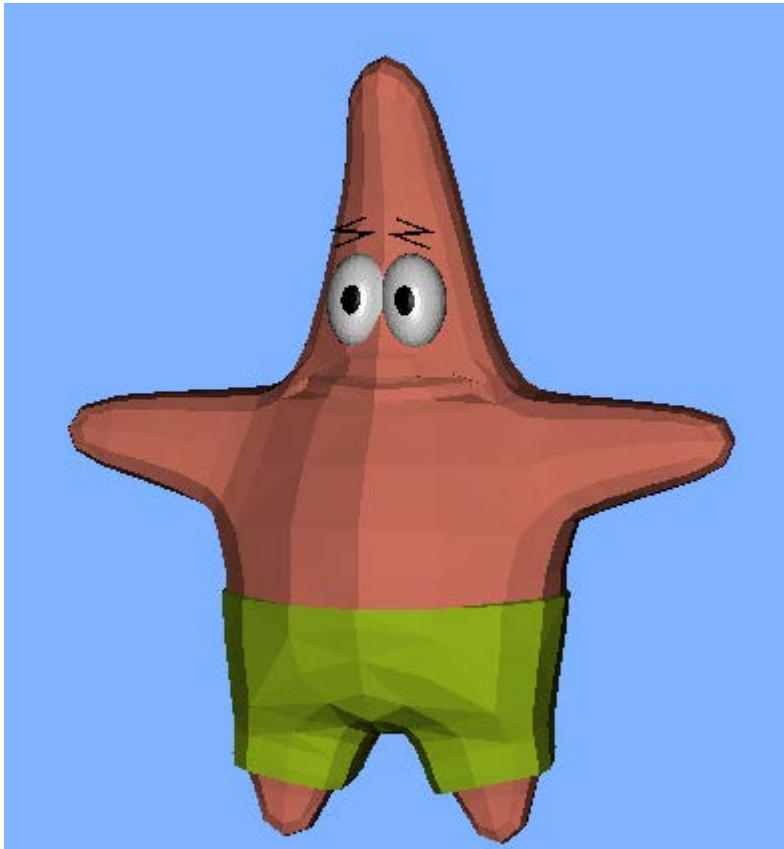
- Normal per cara vs normal per vèrtex



$$\vec{N}_v = \frac{\sum_i \vec{N}_i}{\left\| \sum_i \vec{N}_i \right\|}$$

Suavitzat d'arestes: exemple

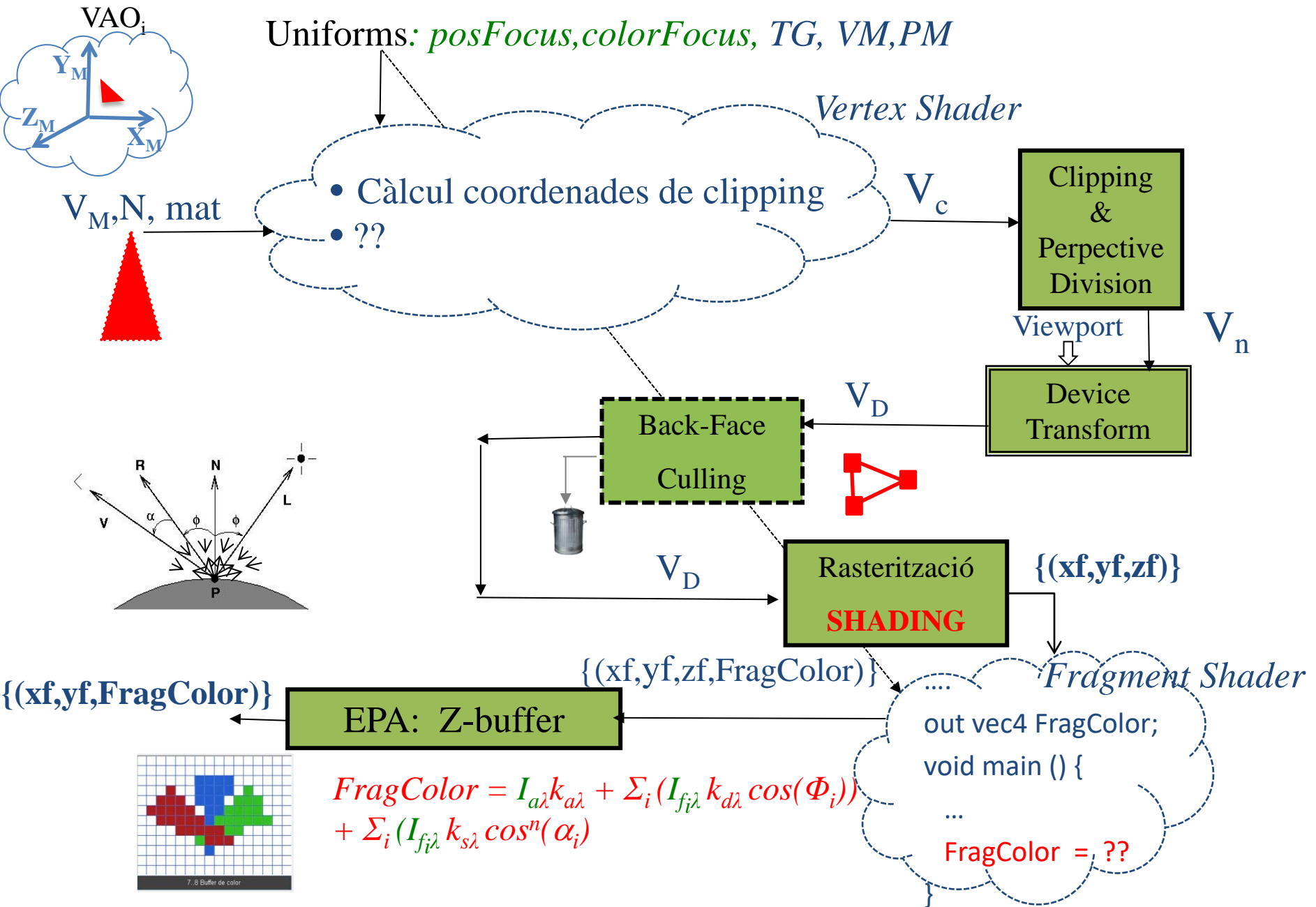
- Normal per cara vs normal per vèrtex



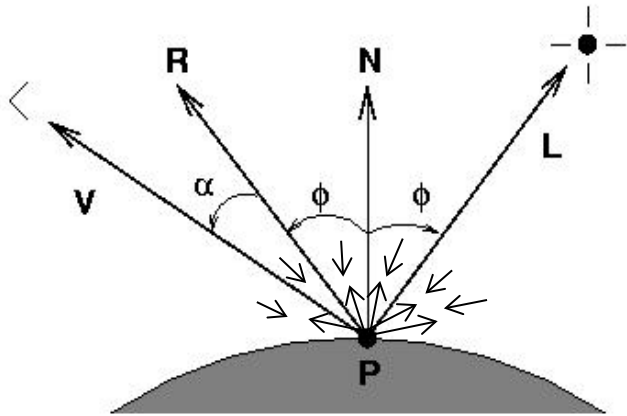
Realisme (II)

- Realisme: Il·luminació (2)
 - Il·luminació en OpenGL 3.3 (1)
 - Càlcul de color en vèrtexs (en el Vertex Shader)
 - Shading de polígons
 - Suavitzat d'arestes
 - **Il·luminació en OpenGL 3.3 (2)**
 - Càlcul de color en fragments

Càlcul del color per fragment en el Fragment Shader (1)



Càlcul del color per fragment en el Fragment Shader (2)



$$\text{FragColor } I_{a\lambda} k_{a\lambda} + \sum_i (I_{fi\lambda} k_{d\lambda} \cos(\Phi_i)) + \sum_i (I_{fi\lambda} k_{s\lambda} \cos^n(\alpha_i))$$

$\cos(\Phi) \Rightarrow \text{dot}(L, N)$ en SCO

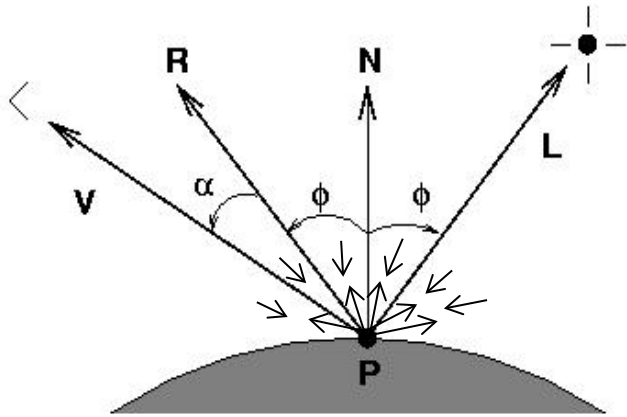
$\cos(\alpha) \Rightarrow \text{dot}(R, v)$ en SCO

Idea 1: Per cada píxel (fragment)

- ✓ Tenim la informació de les llums, són *uniforms*
- Requereix el vèrtex, altres vectors en SCO o SCA i les constants material
 - ✓ Tenim el fragment en SCD i matrius (uniforms) \Rightarrow *podríem calcular les coordenades del punt que es projecte en SCO o SCA aplicant inversa de les matrius.*
 - **Procés costós**
 - **No tenim accés a la Normal ni a les constants empíriques del material (eren atributs/in en VS)**

ALTRE IDEA?

Càlcul del color per fragment en el Fragment Shader (3)



$$\text{FragColor} = I_{a\lambda}k_{a\lambda} + \sum_i (I_{fi\lambda}k_{d\lambda} \cos(\Phi_i)) + \sum_i (I_{fi\lambda}k_{s\lambda} \cos^n(\alpha_i))$$

$$\cos(\Phi) \Rightarrow \text{dot}(L, N) \text{ en } SCO$$

$$\cos(\alpha) \Rightarrow \text{dot}(R, v) \text{ en } SCO$$

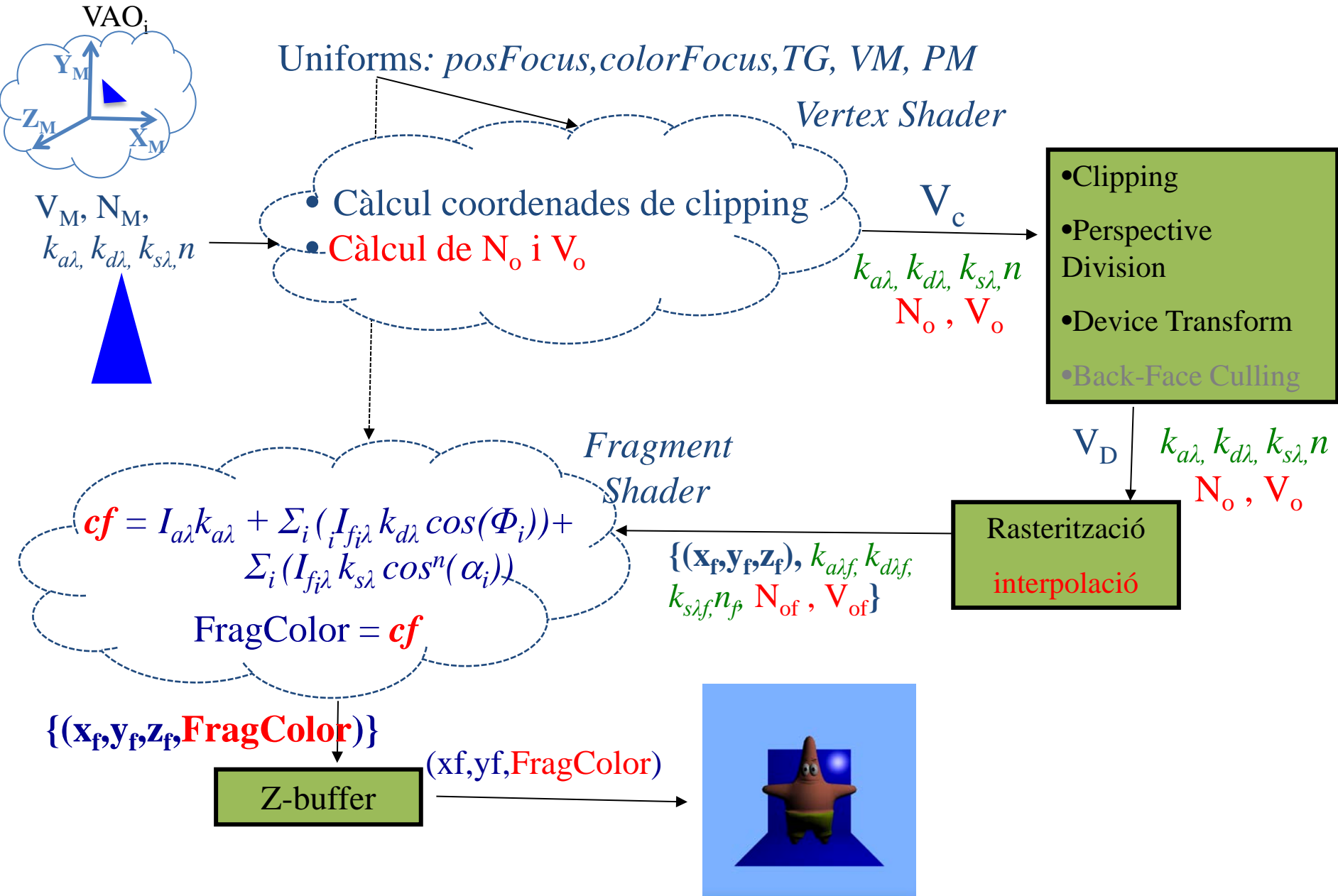
Proposta de solució:

- ✓ Tenim la informació de les llums, són *uniforms*
- ✓ Podem fer “out” en el VS dels atributs associats al vèrtex: N i V *en SCO*, i de les constants empíriques de material. La rasterització calcularà el seu valor pel fragment interpolant la informació dels vèrtexs del triangle ➔ tindrem els seus valors en el FS com variables “in” 😊

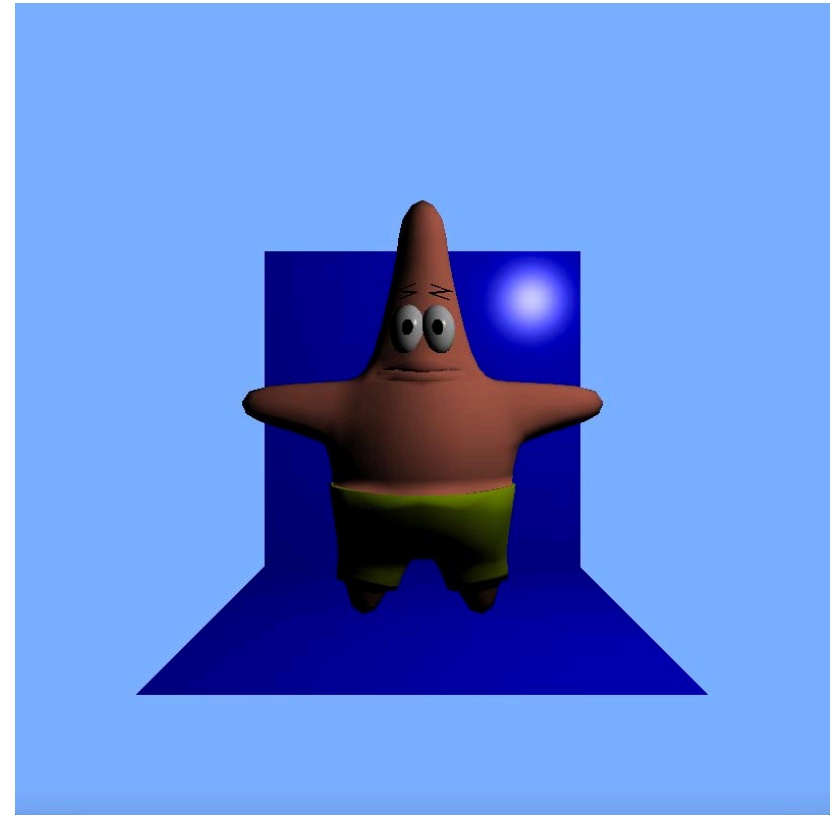
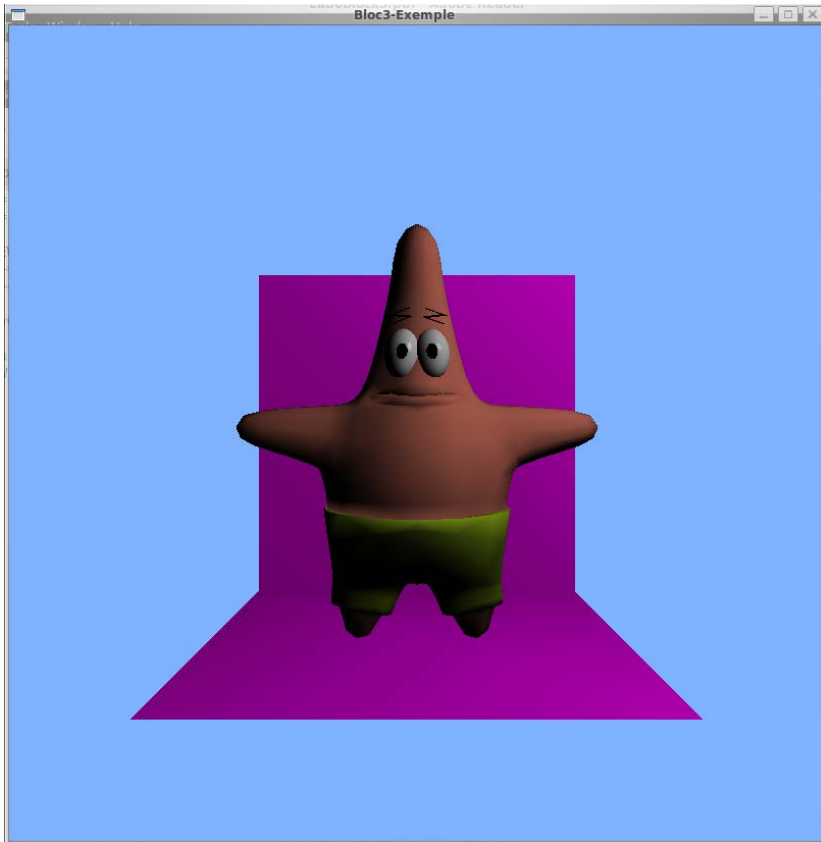
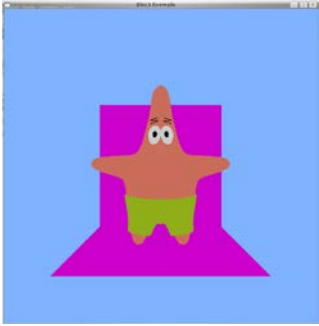
Observació:

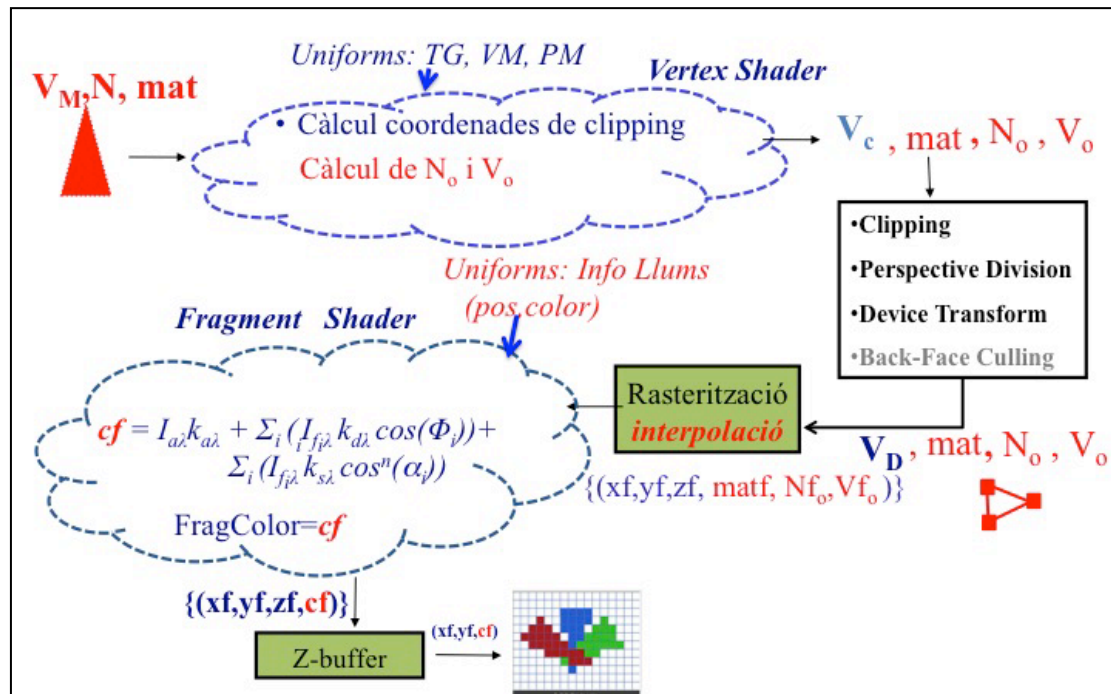
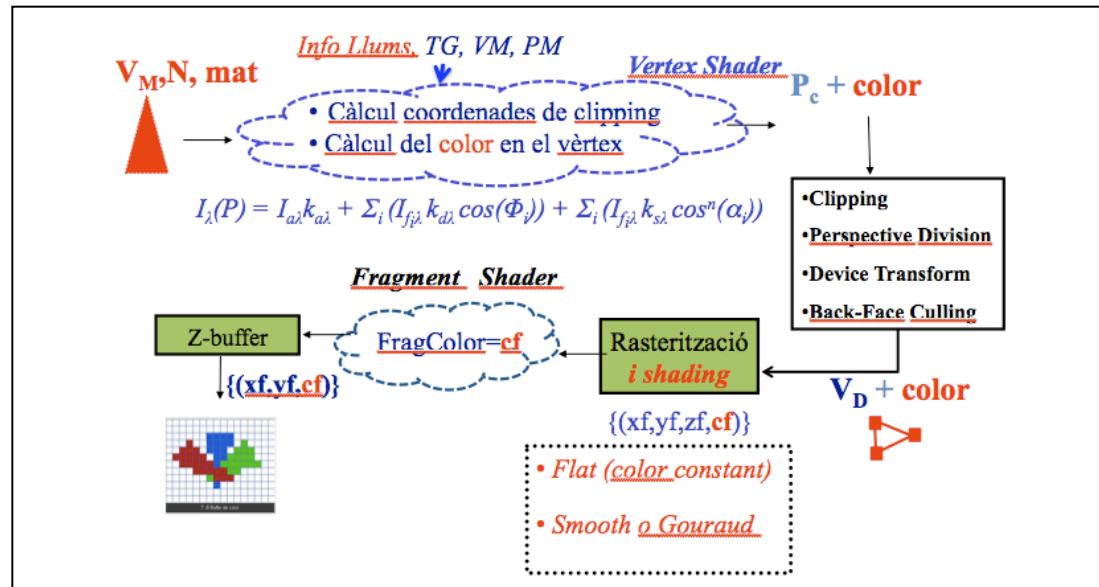
- La normal del fragment és una aproximació de la normal del punt del triangle que es projecta en el fragment. Aquesta interpolació de la normal es coneix com “*shading de phong*”

Càlcul del color per fragment en el Fragment Shader (4)



Exemple final: Sense il·luminació i Il·luminació en el VS versus FS





Conceptes i preguntes

- Càlcul del color en el vèrtex shader. Quants VBOs requereix per VAO d'un model? Quina informació requereix? Cal fer algun càlcul en el fragment shader? La geometria en quin sistema de coordenades de referència ha d'estar?
- Limitacions del càlcul del color en el vèrtex shader.
- Suavitzat d'arestes. Quan cal fer-lo? Quina informació requereix? Degut a quin procés del procés de visualització es pot realitzar? Requereix algun càlcul extra en el vèrtex i/o en el fragment shader?
- Càlcul del color en el fragment shader. Quina informació requereix? Com accedeix a aquesta informació?
- Perquè el càlcul de la il·luminació d'una escena és més costós si es realitza en el FS que en el VS (la fórmula és la mateixa)?
- Quines limitacions en el realisme d'una escena es resolen calculant la il·luminació en el FS enlloc del VS? Quin experiment faries per assegurar-te executant un programa si calcula la il·luminació en el VS o en FS?