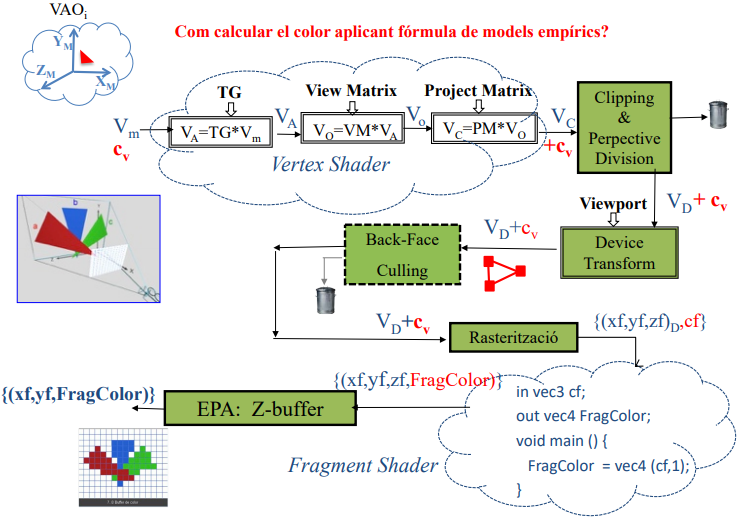
**Realisme 2**

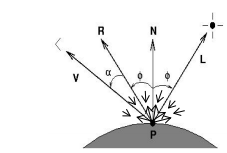
**Realisme: Il·luminació (2)**

**1. Il·luminació en OpenGL 3.3 (1)**

****

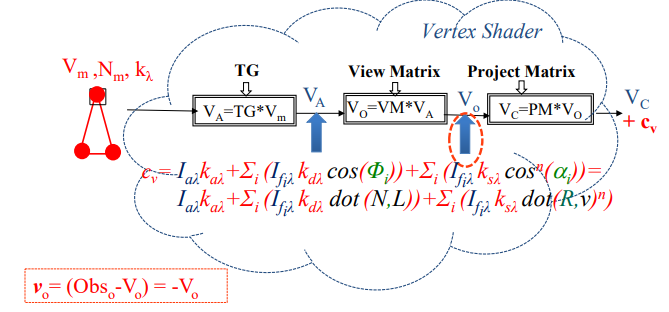
Càlcul del color per vèrtex en el **Vèrtex Shader** (1)

Atributs per cada model:

* Coordenades (V), normal (N) i constants de material (kλ) per vèrtex en VBOs del seu VAO.

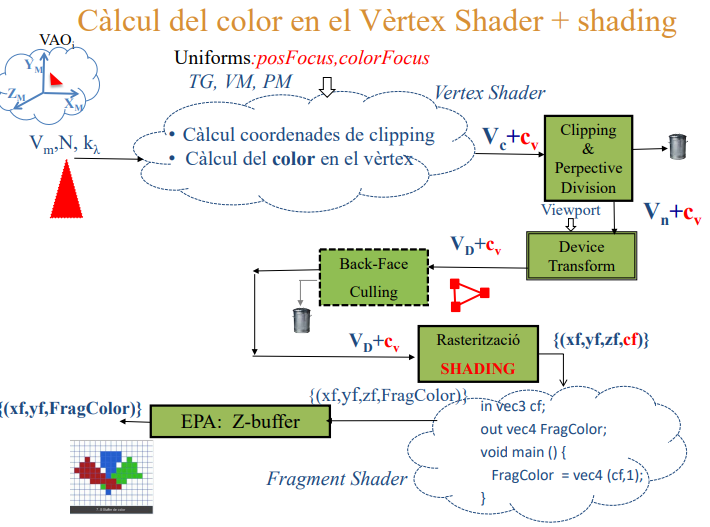
Uniforms:

* Fonts de llum actives => color, posició.
* Llum ambient.

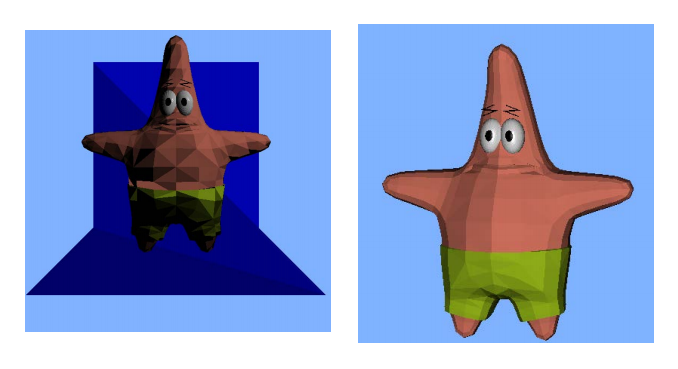


El càlcul el farem per cada vèrtex (al Vertex Shader) i el farem en SCO, per tant:

* Cal passar la posició del vèrtex a SCO:
  + multiplicant per (view \* TG).
* Cal passar el vector normal a SCO:
  + multiplicant 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).

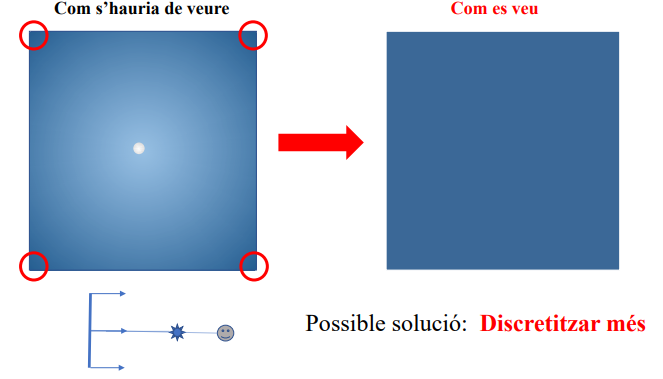


Flat versus Gouraud/smooth Shading:

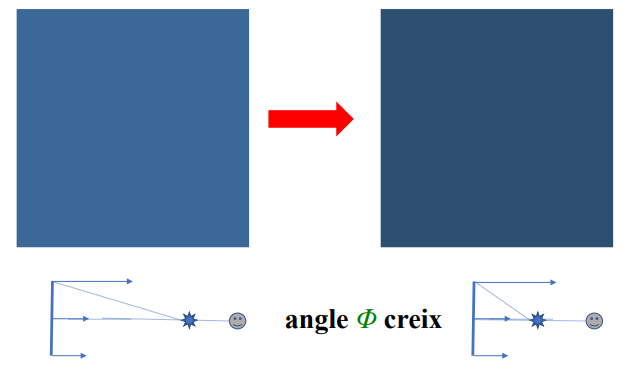


Càlcul color en VS: limitacions

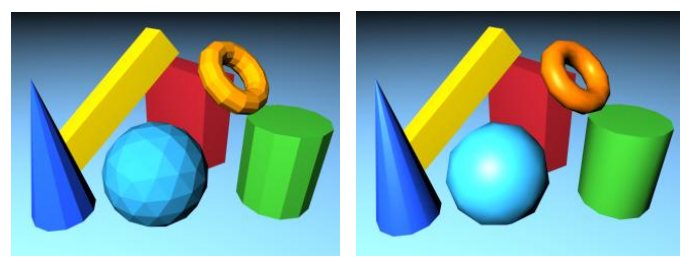
* Efecte del càlcul de Lambert i Phong en Vertex Shader (no s’aprecien taques):



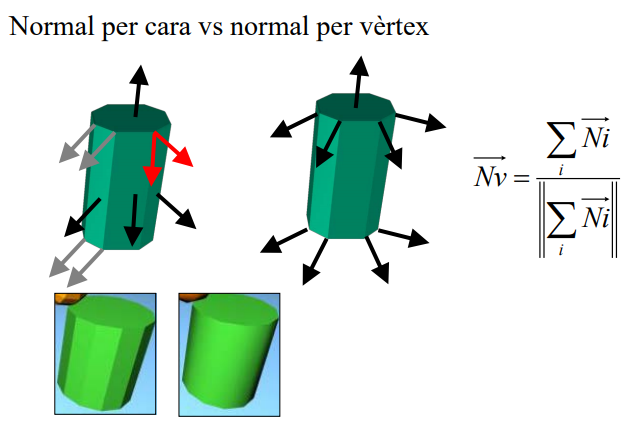
* Què passa si apropem el focus de llum al quadrat?
  + Si creix l’angle, llavors el color és més fosc.



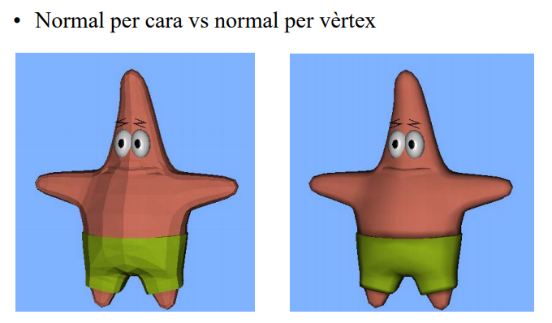
**2. Suavitzat d’arestes**

****

Mateix model, però amb suavitzat d’arestes diferents. El de la dreta té un discretització més gran que el de l’esquerra (hi ha més triangles).

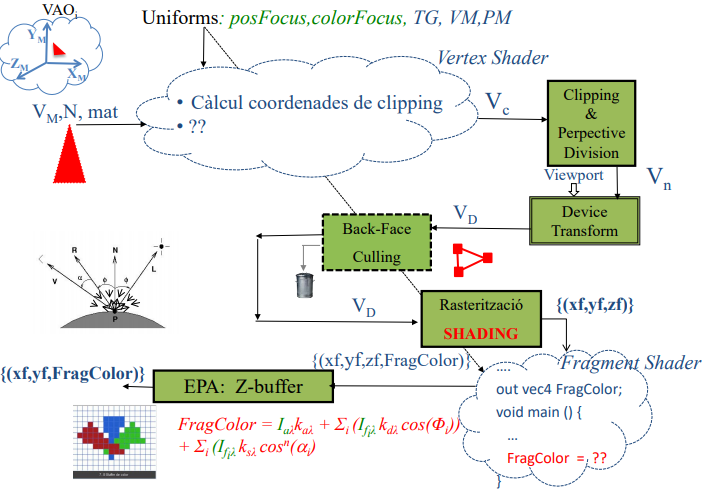


Si es té una normal per cada vèrtex, en comptes que per cada cara, la imatge obtinguda és més aproximat al real.



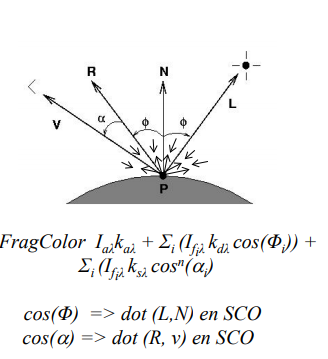
**3. Il·luminació en OpenGL 3.3 (2)**

Càlcul de color en **fragments (Fragment Shader):**

****

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)=> podríem calcular les coordenades del punt que es projecte en SCO o SCA aplicant inversa de les matrius.
  + És un procés costós respecte a fer-ho al Vertex Shader.
  + No tenim accés a la Normal ni a les constants empíriques del material (eren atributs/in en VS).

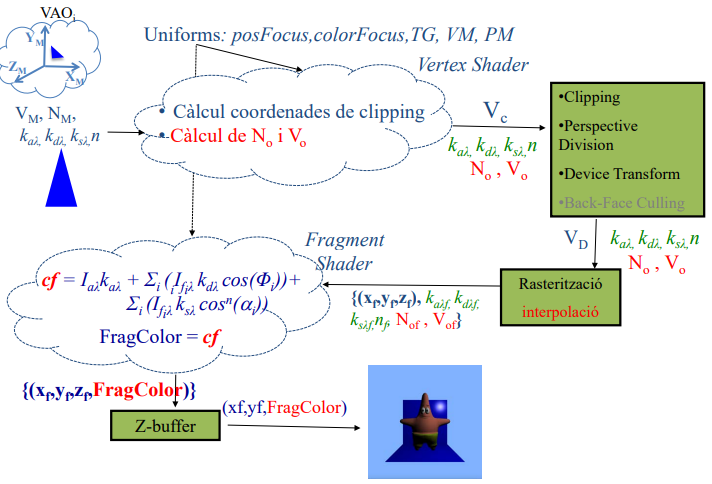
****

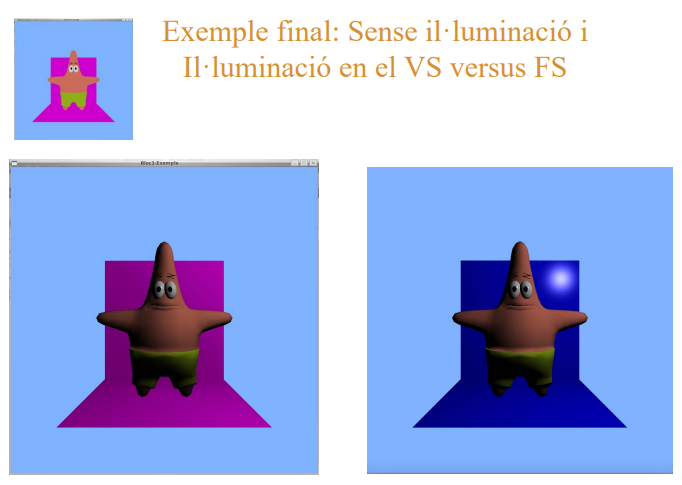
Proposta de solució:

* 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”.

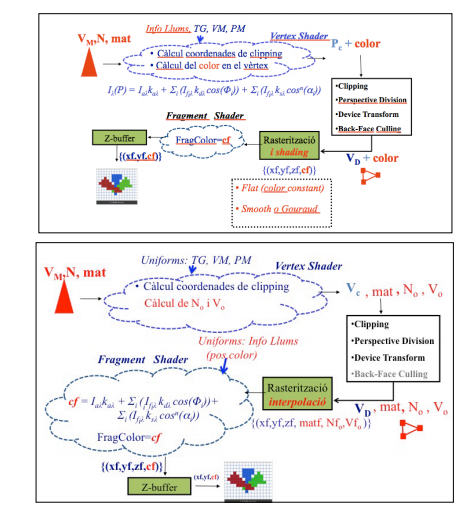
Observació:

* 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”.

****

****

**Resum:**

****