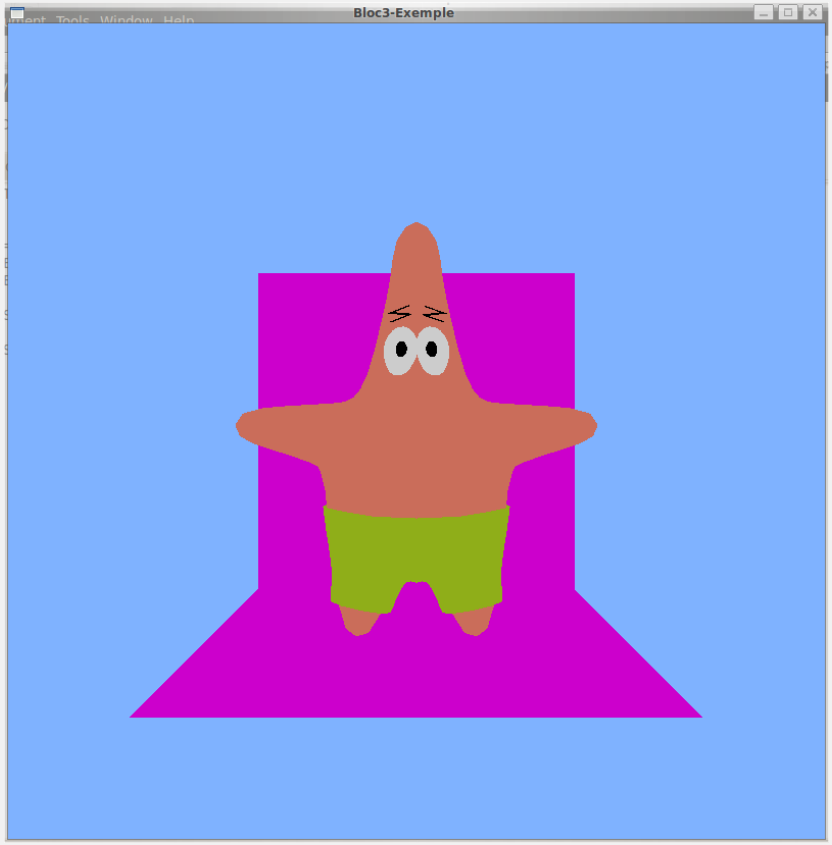
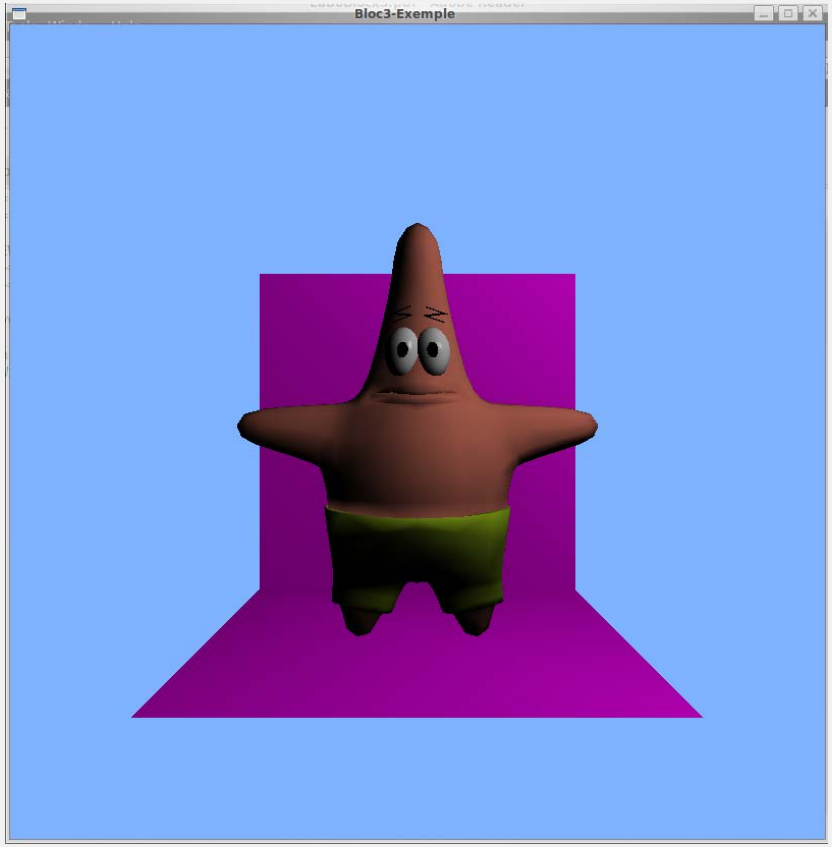
Laboratori OpenGL – Sessió 3.1
Bloc 3

Realisme - Il·luminació:





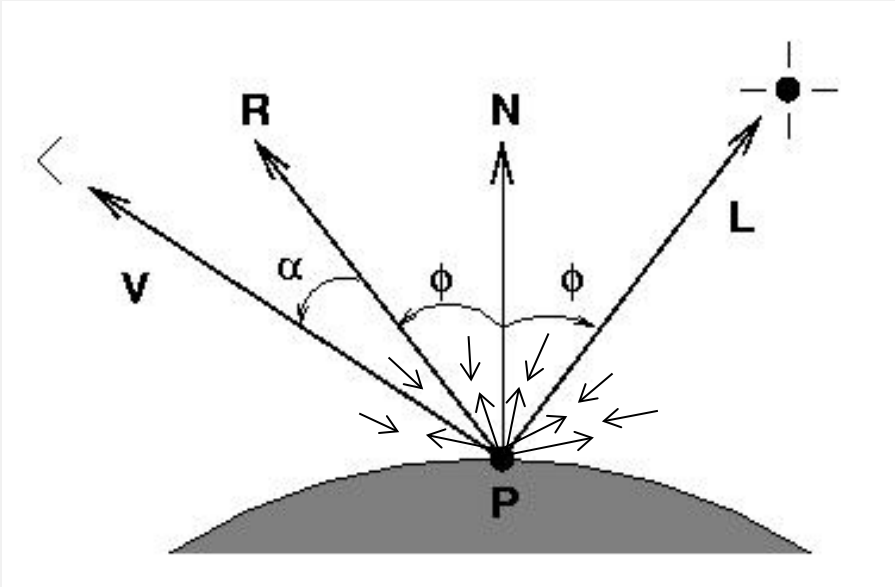
Càlcul color en un punt: models empírics

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

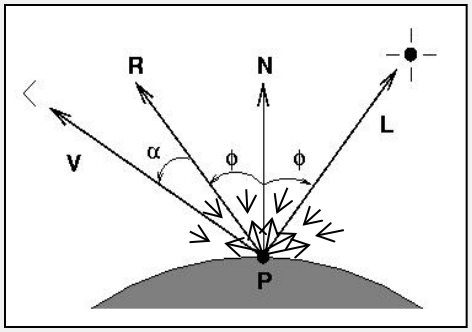
$cos(\Phi)$ => dot (L,N)

$cos(\alpha)$ => dot (R, v)

L, N, R i v normalitzats



Càlcul color en un punt: models empírics



Què necessitem?

- Propietats del material
- Vector normal

} Per cada vertex (punt)

- Color de llum ambient
- Posició del focus de llum
- Color del focus de llum

} Per cada focus de llum

- Posició observador – en SCO sabem que és (0,0,0) -

Càlcul color en un punt: models empírics

Primer farem el càlcul 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** -
- La posició del focus de llum també ha d'estar en SCO
  - Multiplicat per **view** **(si no la tenim ja directament en SCO)**

Càlcul color en un punt: models empírics

Calcular matriu inversa de la trasposada de view * TG

- Al vertex shader (en GLSL):

```
mat3 NormalMatrix = inverse (transpose (mat3 (view * TG)));
```

  - ➢ es fa el càlcul de la matriu per a cada vèrtex

- Al programa (amb glm):

```
#include “glm/gtc/matrix_inverse.hpp”
glm::mat3 NormalMatrix = glm::inverseTranspose(glm::mat3(View*TG));
```

  - ➢ cal tenir les matrius View i TG com a atributs de la classe
  - ➢ i cal passar la NormalMatrix com a uniform al VS per cada objecte

Anàlisi del codi de l'esquelet

- Analitzar quins mètodes implementats.
- Analitzar implementació dels mètodes.
  - Quina càmera tenim?
  - Quina escena?
  - Quina interacció?
- Analitzar els shaders
  - Atributs, uniforms, funcions

Exercici 1

Càlcul color usant model Lambert:

```
vec3 Lambert (vec3 NormSCO, vec3 L)
{
    // Aquesta funció calcula la il·luminació amb Lambert assumint que els vectors
    // que rep com a paràmetres estan normalitzats

    vec3 colRes = llumAmbient * matamb; // Inicialitzem color a component ambient
    // Afegim component difusa, si n'hi ha
    if (dot (L, NormSCO) > 0)
        colRes = colRes + colFocus * matdiff * dot (L, NormSCO);
    return (colRes);
}
```

Cal calcular en *main*: L en SCO, Normal en SCO,
normalitzar vectors i cridar a Lambert

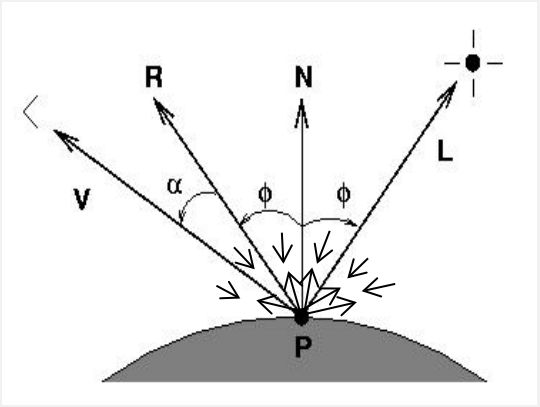
Càlcul del color per vèrtex en el Vèrtex Shader (2)

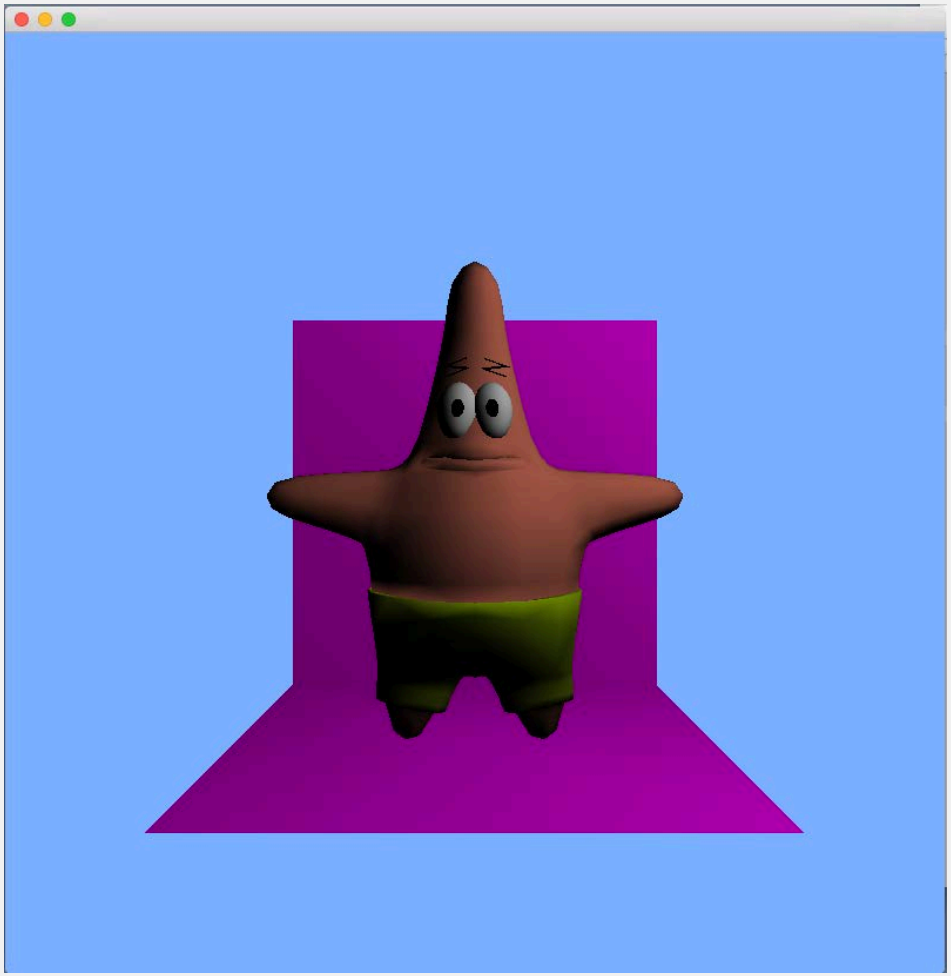
(transparència extreta de teoria)

Pseudocodi de Vertex Shader per calcular color en SCO:

$$c_v = I_{a\lambda} k_{a\lambda} + \Sigma_i (I_{fi\lambda}\, k_{d\lambda} dot(N_o, L_o)) + \Sigma_i (I_{fi\lambda}\, k_{s\lambda}\, dot(R_o, v_o)^n)$$

```
in vec3 vertex, N;
in vec3 matamb, matdiff, matspec;
in float matshin;
uniform mat4 proj, view, TG;
uniform vec3 posFocus, Ia, If;
out vec3 fcolor;
...
void main()
{
   mat3 NormalMatrix= inverse (transpose (mat3 (view*TG)))
   vec3 NormSCO= normalize(NormalMatrix*N);
   vec4 vertexSCO= view*TG* vec4(vertex,1.0);
   vec4 focusSCO = view* vec4 (posFocus, 1.0);
   vec3  LSCO=normalize (focusSCO.xyz-vertexSCO.xyz);
   fcolor= color_vertex(NormSCO, LSCO, vertexSCO);
   gl_Position = proj * view * TG * vec4 (vertex, 1.0);
}
```



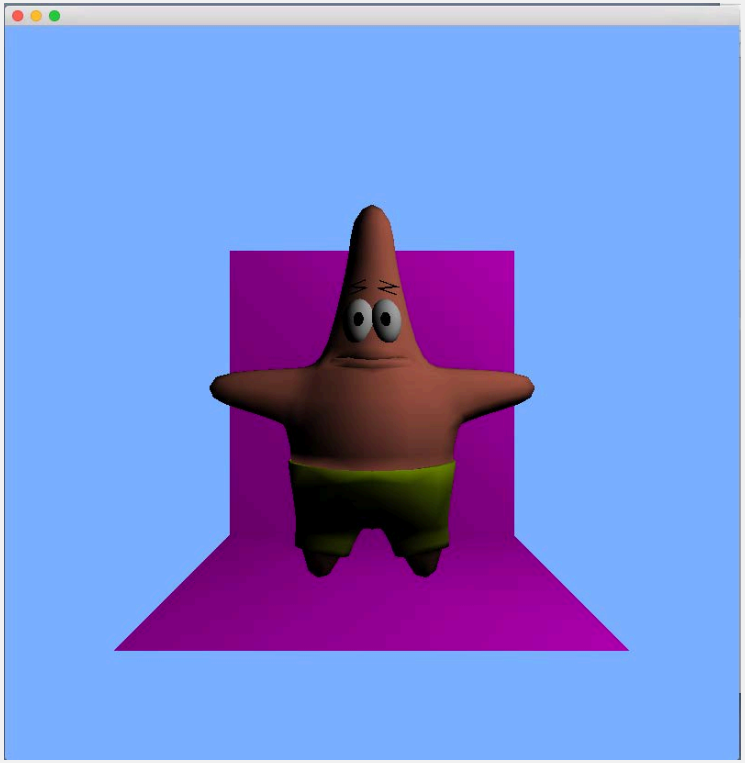


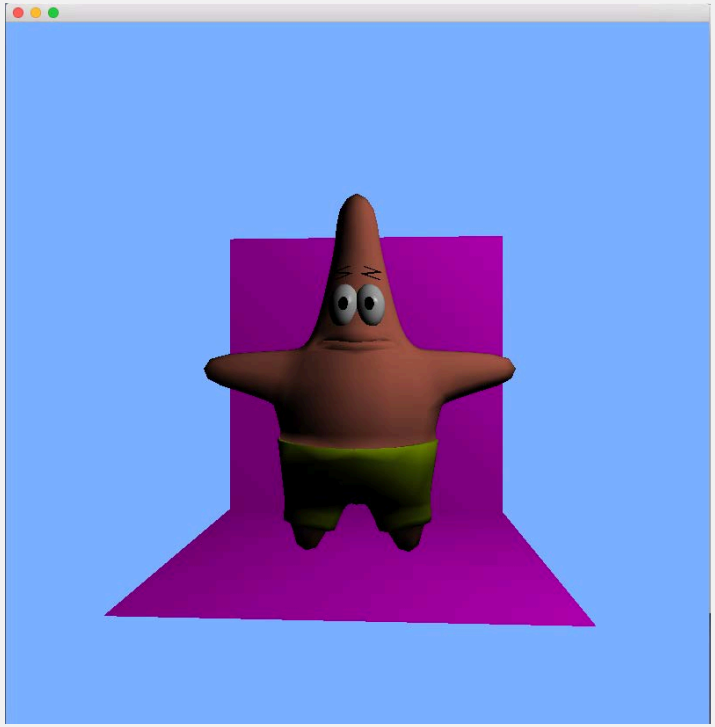
Proveu a moure càmera (si voleu poseu tb rotació en X) i veureu que cares il·luminades no varien. **Llum d’escena**.

Exercici 2

Càlcul color usant model Phong:

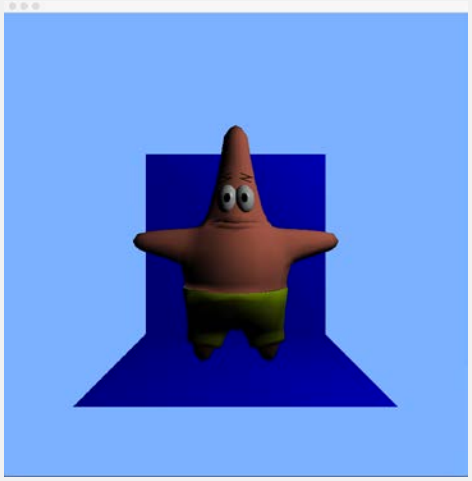
```
vec3 Phong (vec3 NormSCO, vec3 L, vec4 vertSCO)
{
    // Els vectors rebuts com a paràmetres (NormSCO i L) estan normalitzats
    vec3 colRes = Lambert (NormSCO, L); // Inicialitzem color a Lambert
    // Calculem R i V
    if (dot (NormSCO, L) < 0)
        return colRes;  // no afecta la component especular
    vec3 R = reflect (-L, NormSCO); // equival a:: 2.0 * dot (NormSCO, L) * NormSCO - L;
    vec3 V = normalize (-vertSCO.xyz);
    if ((dot (R, V) < 0) || (matshin == 0))
        return colRes; // no afecta la component especular
    // Afegim la component especular
    float shine = pow (dot (R, V), matshin);
    return colRes + matspec * colFocus * shine;
}
```





Proveu a moure càmera (si voleu poseu tb rotació en X) i veureu que cares il·luminades no varien; però taca especular sí (en ulls).
Llum d'escena.
El terra pot tenir taca especular?

Exercicis 3 i 4



3) Canvi material terra+paret

- Ha de ser de plàstic blau

4) Canvi posició focus de llum

- Ha de ser la posició (1, 0, 1) en SCA

Exercici 5

Pas a uniforms de la posició i el color del focus de llum:

- Convertir la posició i el color en uniforms en el VS
- Inicialitzar aquests uniforms al MyGLWidget
- Fixem-nos que ara podríem passar el uniform de posició directament ja en SCO

Podem també passar a uniform el color de la llum ambient

Exercici 6

Fer que la posició del focus de llum es mogui amb les tecles K i L:

- K → mou el focus cap a les X-
- L → mou el focus cap a les X+