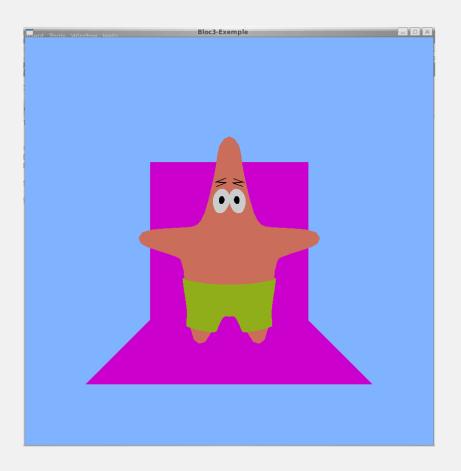
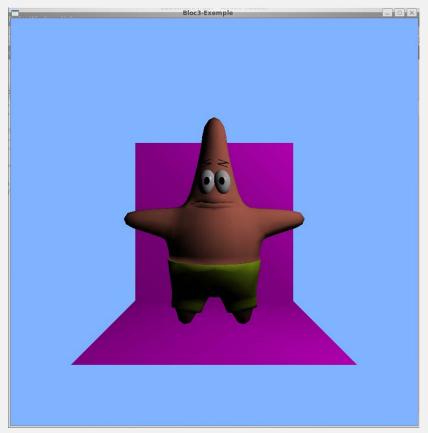
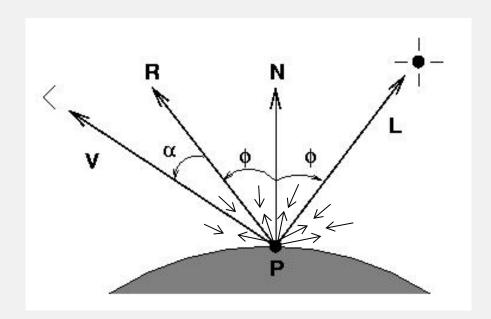
Laboratori OpenGL – Sessió 3.1 Bloc 3

Realisme - Il·luminació:





$$I_{\lambda}(P) = I_{a\lambda}k_{a\lambda} + \sum_{i} (I_{fi\lambda}k_{d\lambda}\cos(\Phi_{i})) + \sum_{i} (I_{fi\lambda}k_{s\lambda}\cos^{n}(\alpha_{i}))$$



Què necessitem?

- Propietats del material
- Vector normal
- Color de llum ambient
- Posició del focus de llum
- Color del focus de llum

Per cada focus de llum

V a b b

Per cada vertex (punt)

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

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)

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):

```
glm::mat3 NormalMatrix = glm::inverseTranspose(glm::mat3(View*TG));
```

- ➤ cal tenir les matrius View i TG com a atributs de la classe #include "glm/gtc/matrix_inverse.hpp"
- i cal passar la matriu com a uniform al VS per cada objecte

Anàlisi del codi de l'esquelet

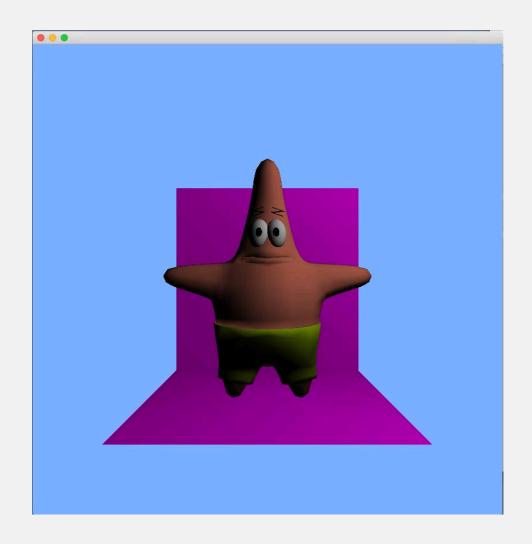
- Analitzar quins mètodes implementats.
- Analitzar implementació dels mètodes.
 - Quina càmera tenim?
 - Quina escena?
 - Quina interacció?
- Analitzar els shaders
 - Atributs, uniforms, mètodes

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);
}
```

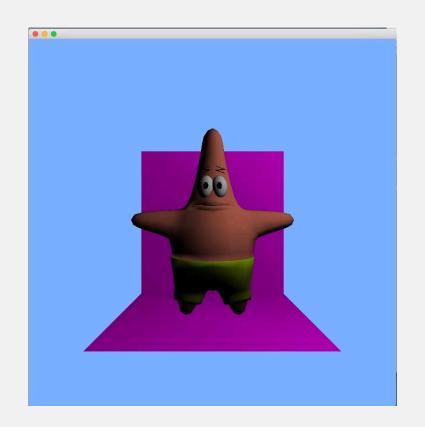
Cal calcular en *main*: L en SCO, Normal en SCO, normalitzar vectors i cridar a Lambert



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

Càlcul color usant model Phong:

```
vec3 Phong (vec3 NormSCO, vec3 L, vec4 vertSCO)
  // Els vectors rebuts com a paràmetres 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:: normalize (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;
```



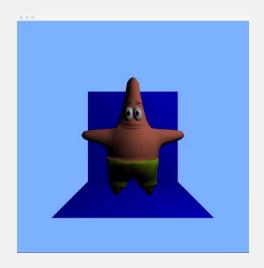


Proveu a moure càmera (si voleu poseu tb rotació en X) i veureu que cares il·luminades no varien; però taca especular sí (en ulls). **Llum d'escena**.

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, 0) en SCA



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

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

- \triangleright K \rightarrow mou el focus cap a les X-
- \triangleright L \rightarrow mou el focus cap a les X+