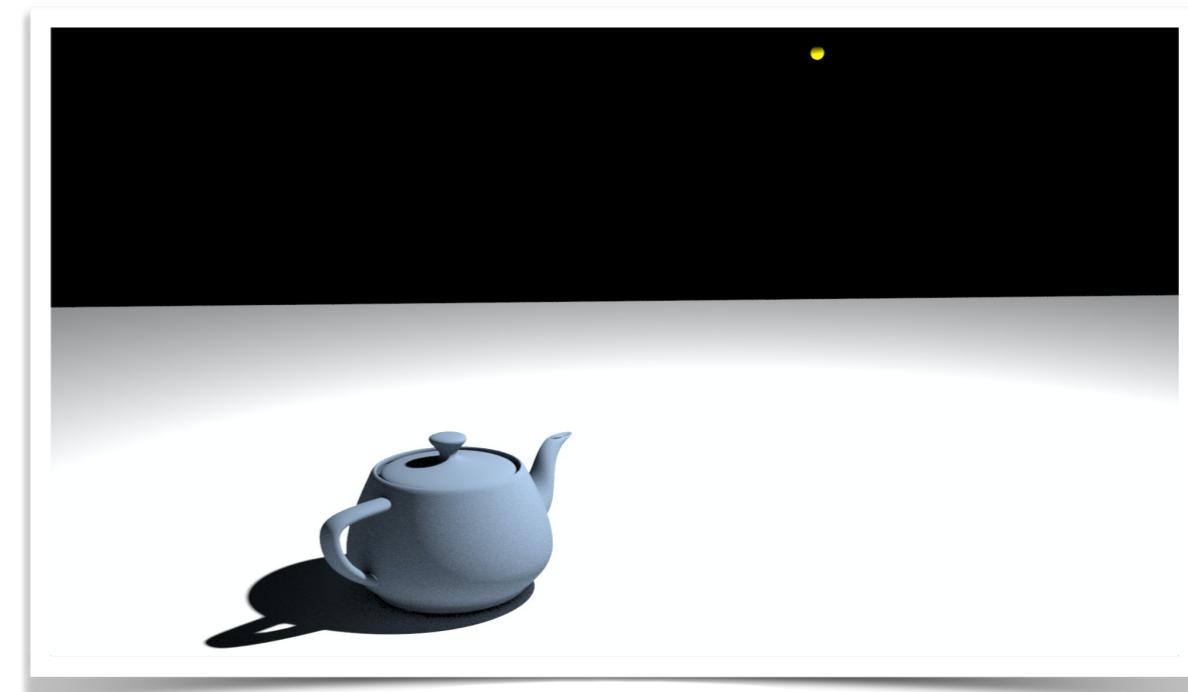


Sessió Setmana 5

GiVD 2022-23

Guió de la sessió

1. Planificació de la setmana 5
 2. Estructures optimització
 3. Model d'il·luminació: Phong, Blinn-Phong i Cell shading
 4. Ombres
 5. Kahoot!
 6. Intro a Reflexions i Transparències
-



1. Planificació

2023 March



Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	
13	14	15	16	17	18	19	
Tema 2	Pràctica 1						Set. 5
20	21	22	23	24	25	26	
Tema 2	Pràctica 1					Problemes 1	Set. 6
27	28	29	30	31	01	02	
Examens parciais				Dia del parcial 15:00-18:00. Prova/Entrevista Pràctica 1			Pràctica 1

Tasca 1: 2 preguntes amb 4 possibles respostes, només una és certa.

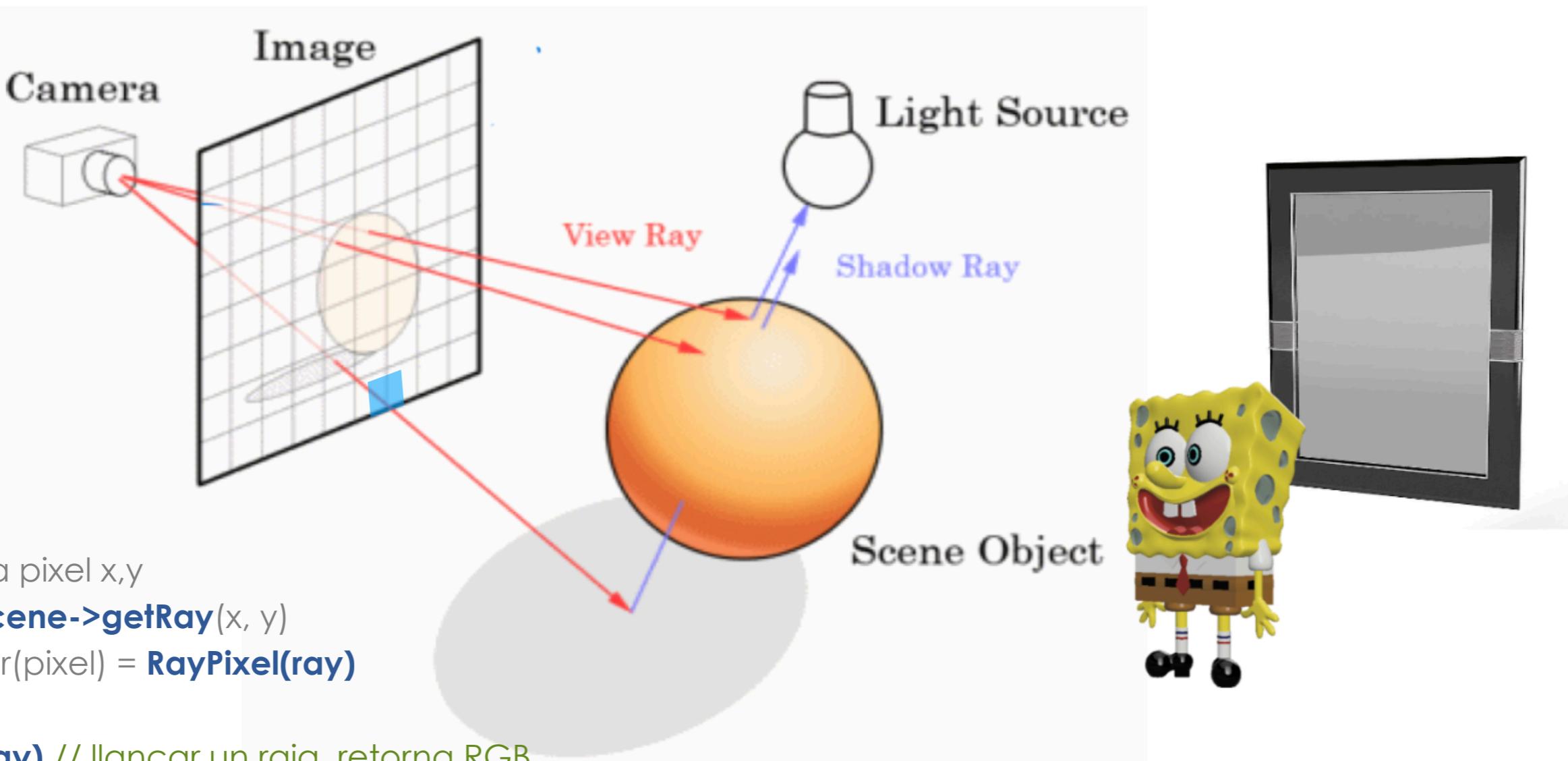
Questionari del campus com exemple de preguntes a lliurar



Data límit:

**Diumenge 26 de Març de 2023
23:59h**

<https://campusvirtual.ub.edu/mod/assign/view.php?id=4225669>

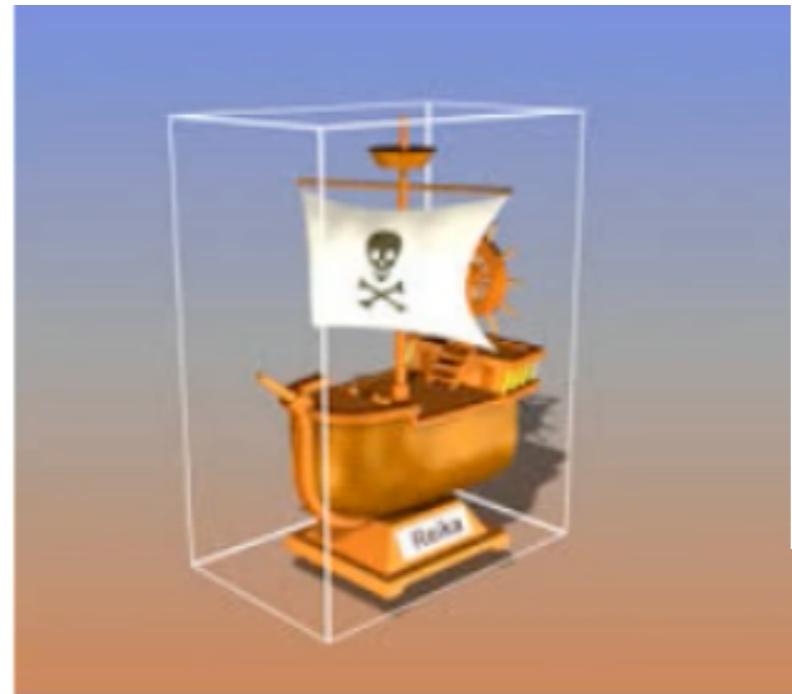


- **run()**
 - per cada pixel x, y
 - $r = \text{scene-} \rightarrow \text{getRay}(x, y)$
 - $\text{color(pixel)} = \text{RayPixel(ray)}$
- **RayPixel(ray)** // llançar un raig, retorna RGB
 - $\text{hitInfo} = \text{hit(ray)}$ // hit(ray, tmin, tmax, hitinfo)
 - si object_point retorna **Shade(hitInfo, ray)**
 - sino retorna Background_Color // o Intensitat ambient global
- **hit(ray)**
 - per cada objecte en l'escena
 - hit(ray, surface)** // hit(ray, tmin, tmax, hitinfo)
 - retorna el punt més proper a l'observador (+normal, +material)
- **Shade(hitInfo, ray)** // retorna la il·luminació en el point (strategy)
 - retorna color

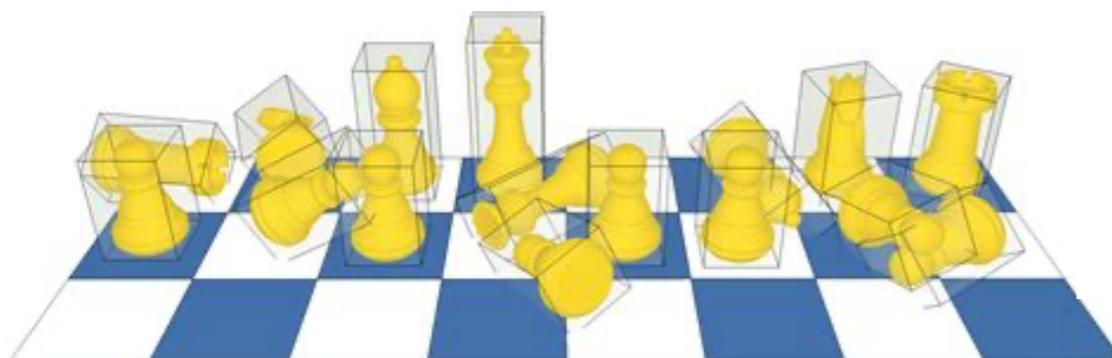
Acceleració de les interseccions raig-objecte



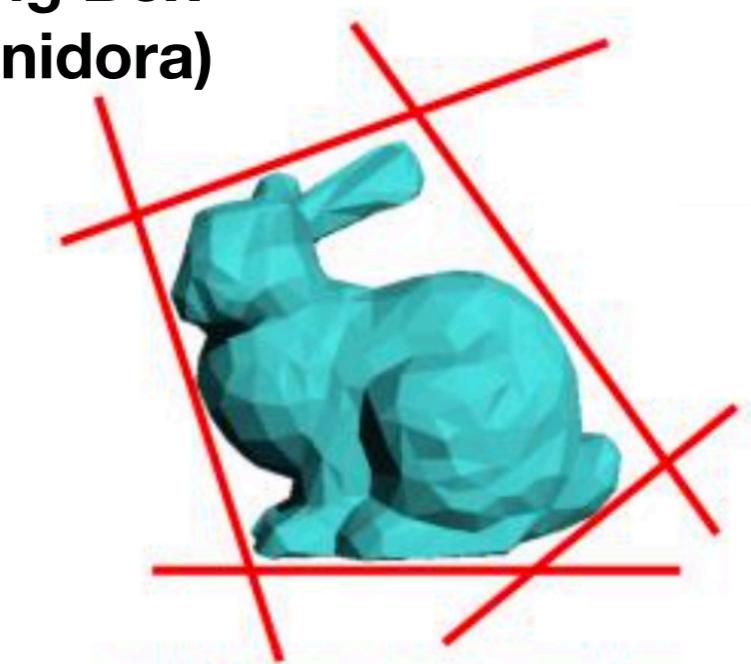
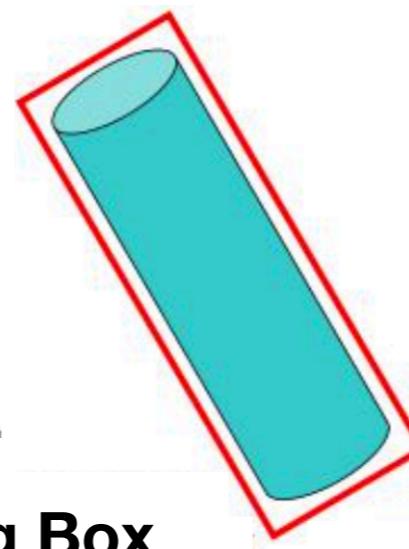
Bounding sphere
(Esfera mínima contenidora)



Axis-Aligned Bounding Box
(Capsa mínima contenidora)



Object-Oriented Bounding Box



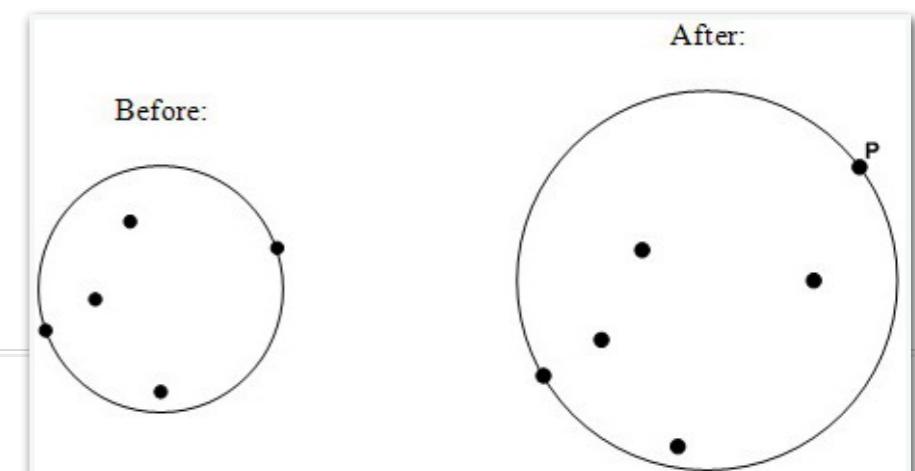
Bounding Half-Spaces
(plans arbitraris)

Esfera mínima contenidora

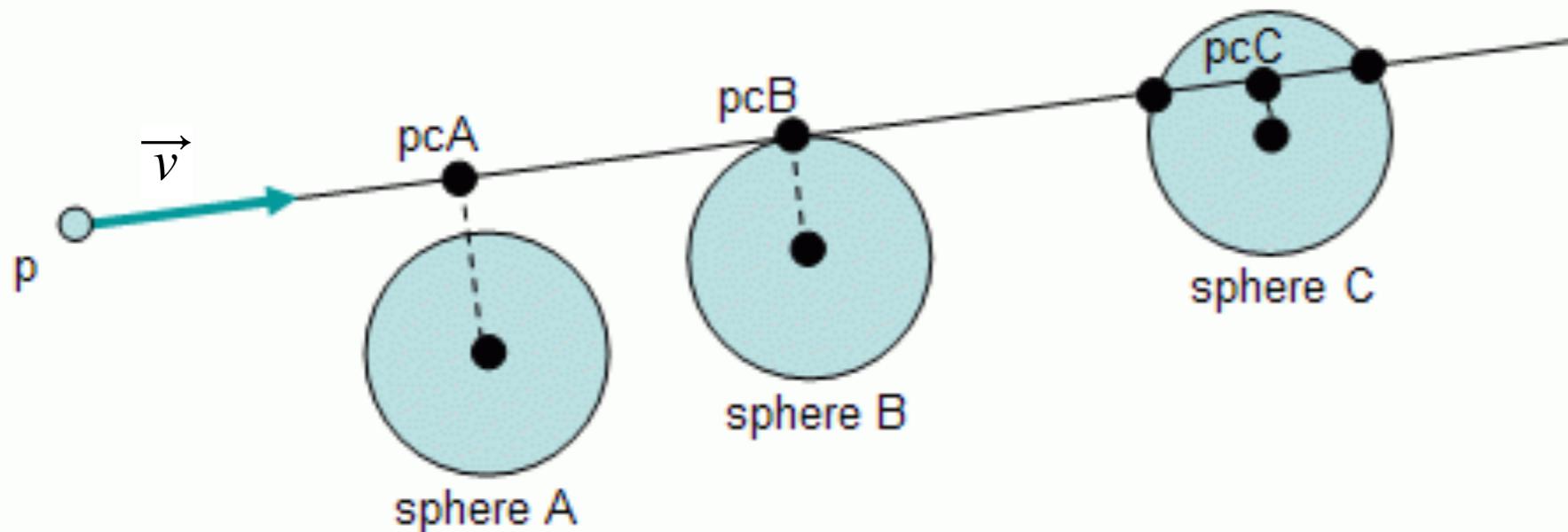
Com calcular-la en una mesh de n vèrtexs?



- de forma exacta
 - en temps $O(n^2)$
 - en temps quasi polinomial (mètode de Welzl)
<https://citeseerx.ist.psu.edu/doc/10.1.1.46.1450>
- de forma aproximada (mètode de Ritter) $O(3*n)$
https://www.researchgate.net/profile/Jack-Ritter/publication/242453691_An_Efficient_Bounding_Sphere/links/56e9d24e08ae95bddc2a2358/An-Efficient-Bounding-Sphere

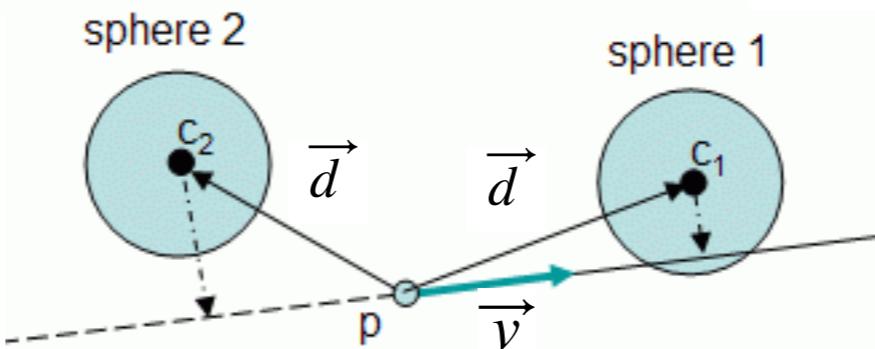


Esfera mínima contenidora

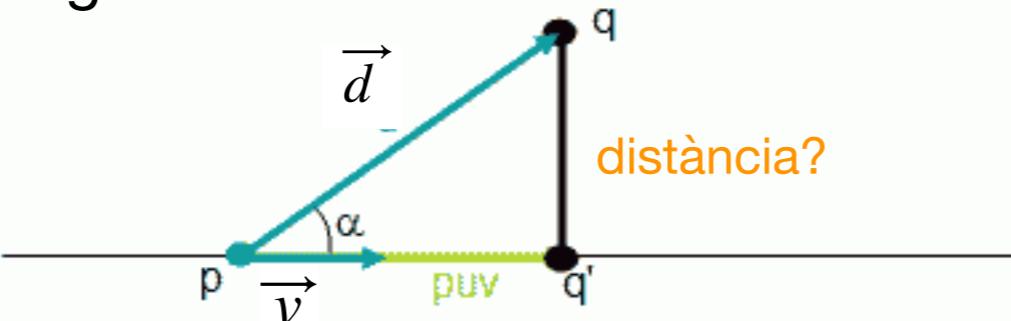


PAS 1: Descartar esferes que estan “darrera” del raig:

$$\vec{v} \cdot \vec{d} \leq 0$$



PAS 2: Càcul de la distància del centre de l'esfera al raig



$$projeccio(\vec{d}, \vec{v}) = \frac{\vec{d} \cdot \vec{v}}{|\vec{v}|} * \vec{v}$$

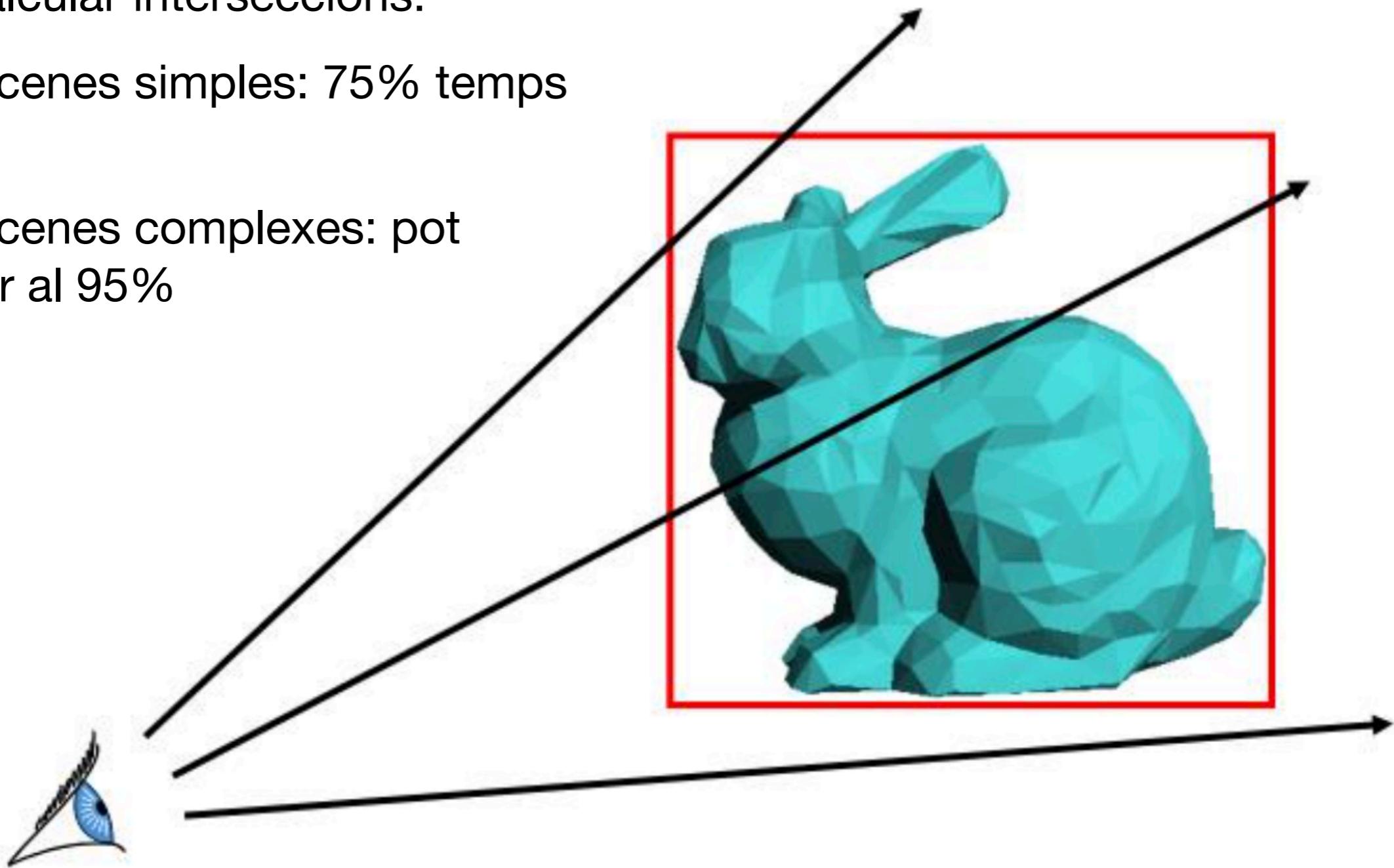
$$q' = p + projeccio(\vec{d}, \vec{v})$$

$$distancia = |q - q'|$$

2. Estructures optimització

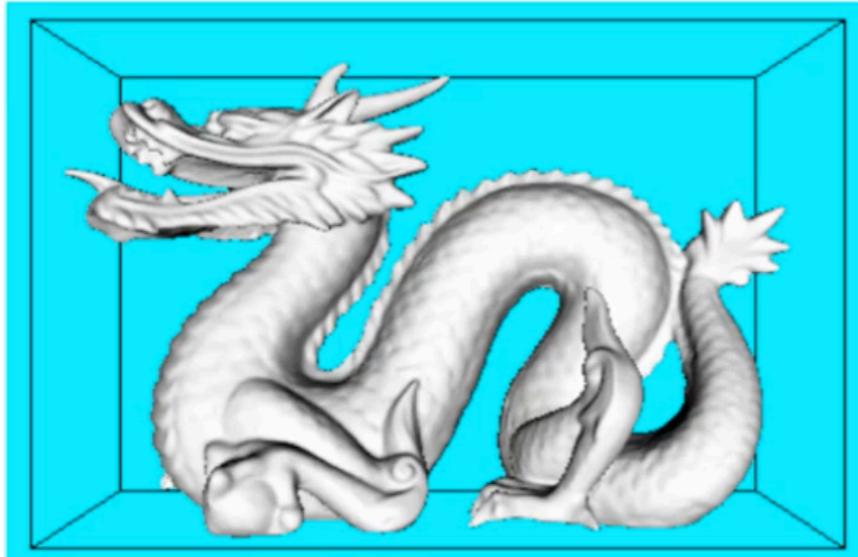
Per a calcular interseccions:

- En escenes simples: 75% temps total
- En escenes complexes: pot arribar al 95%



Costos Bounding Volumes

Bounding Volumes



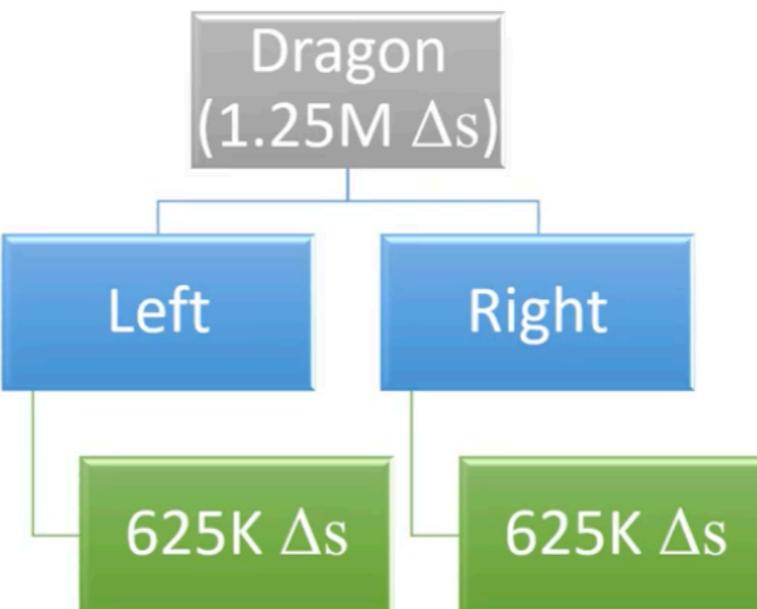
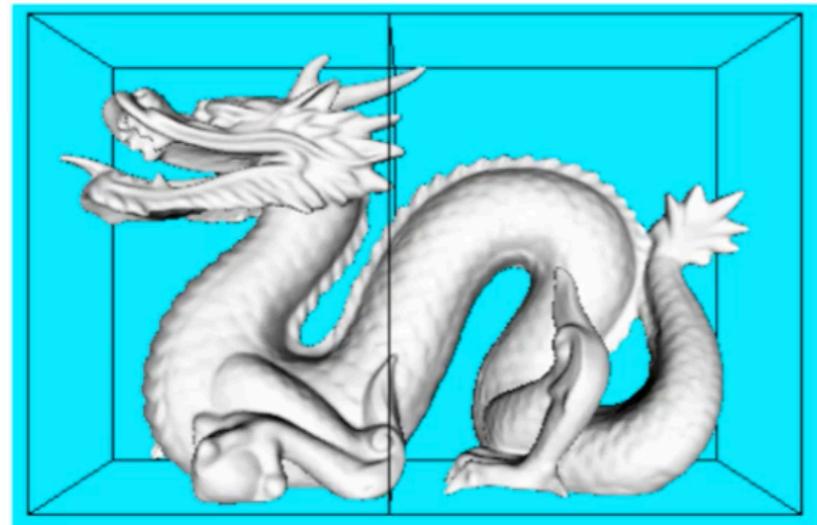
1,000x1000 pixels

Assume 5% rays miss bounding volume
95% rays hit volume and must be tested
against 1.25 million triangles

50,000 rays need 1 AABB test

950,000 rays need 1 AABB test and
1.25M ray-triangle tests

We improved the speed slightly (around
4-5%)



1,000x1000 pixels

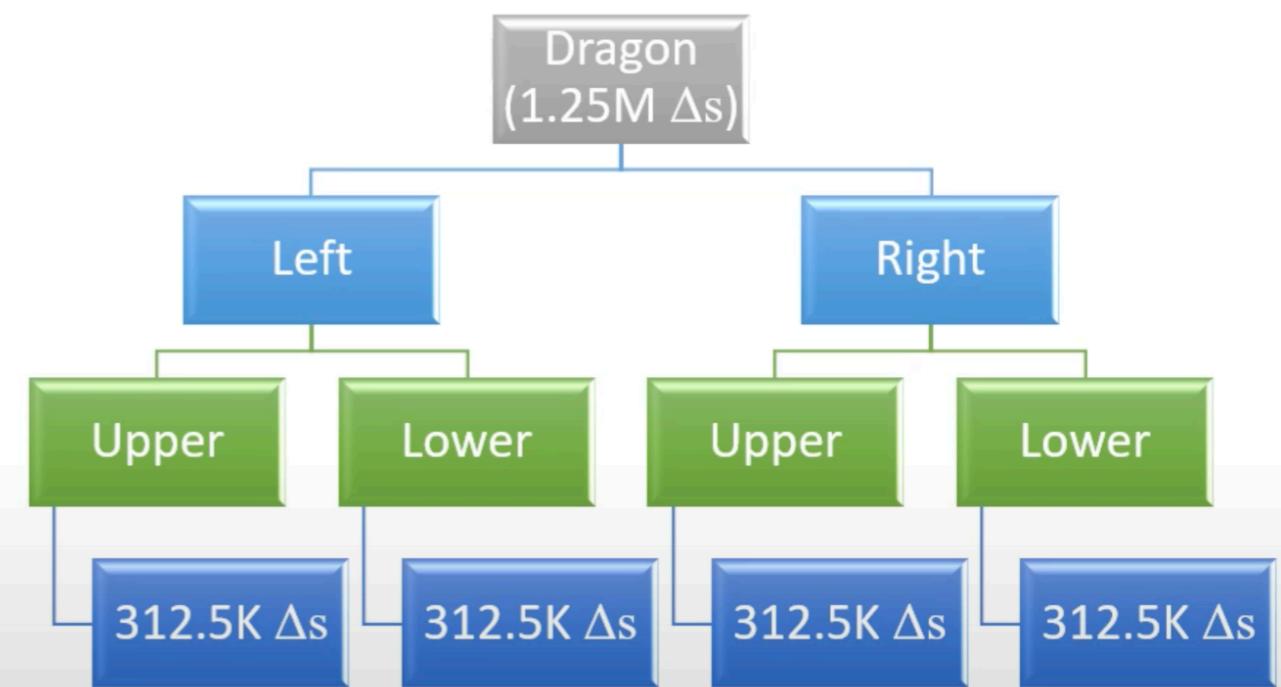
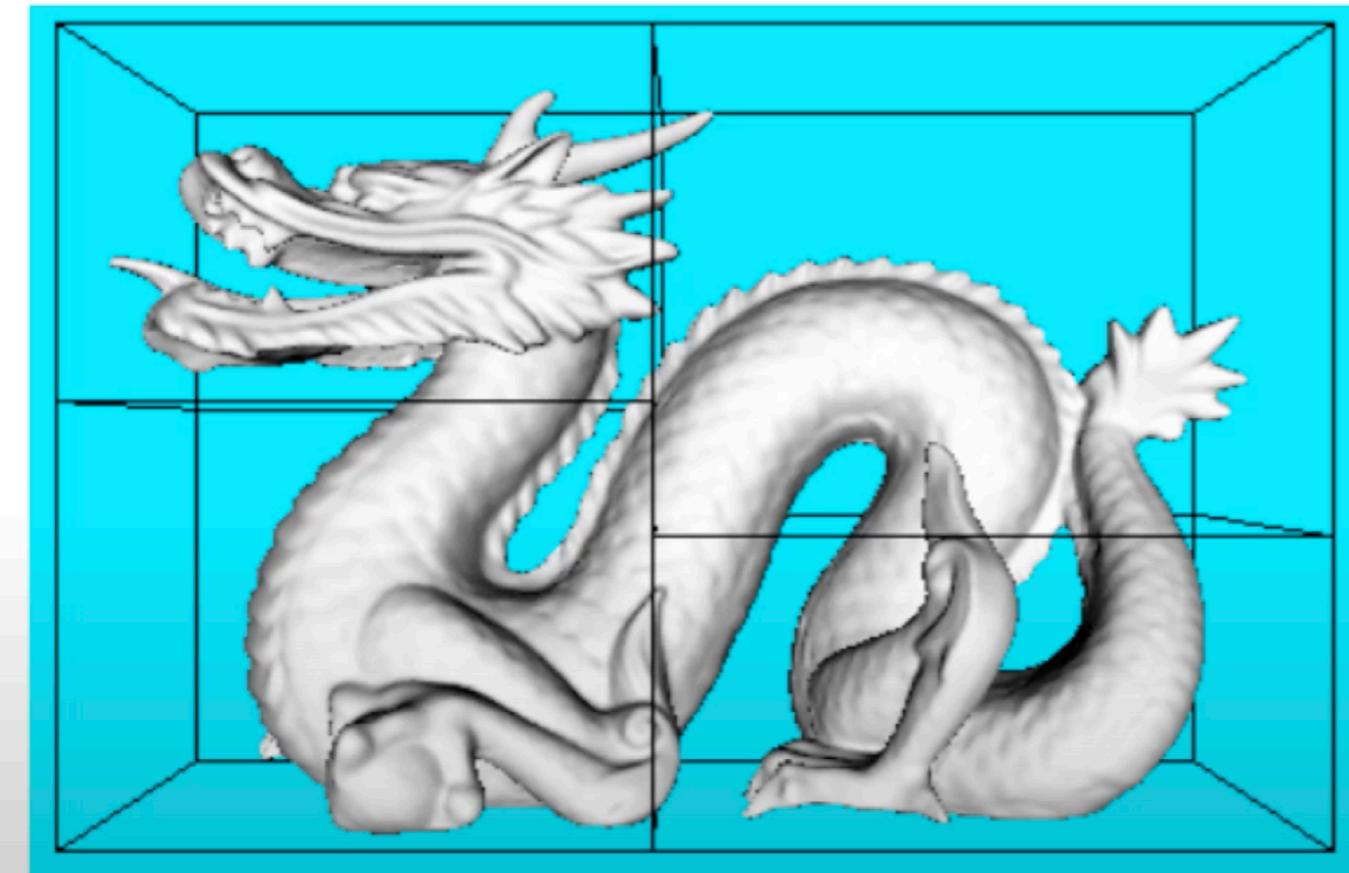
Assume 5% rays miss bounding volume
95% rays hit volume and must be tested
against 2 bounding volumes

47.5% rays hit left and must be tested
against 625K triangles.
47.5% hit right.

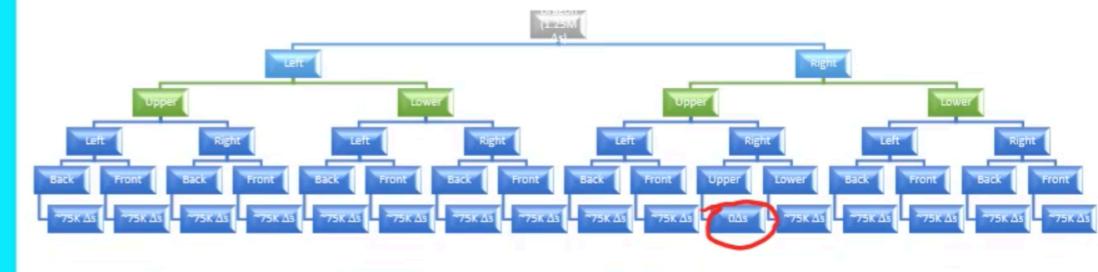
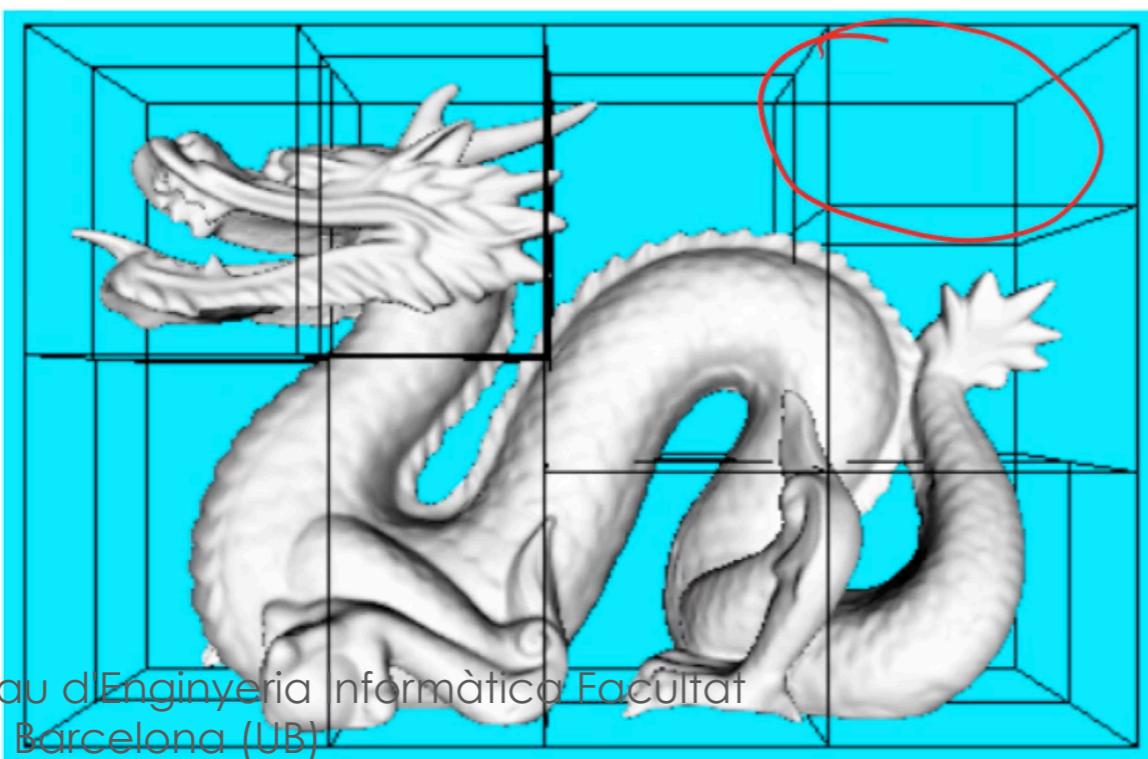
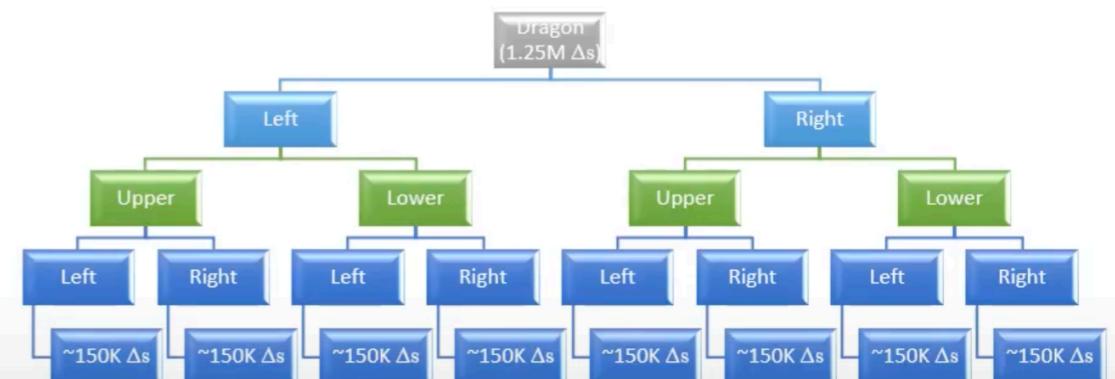
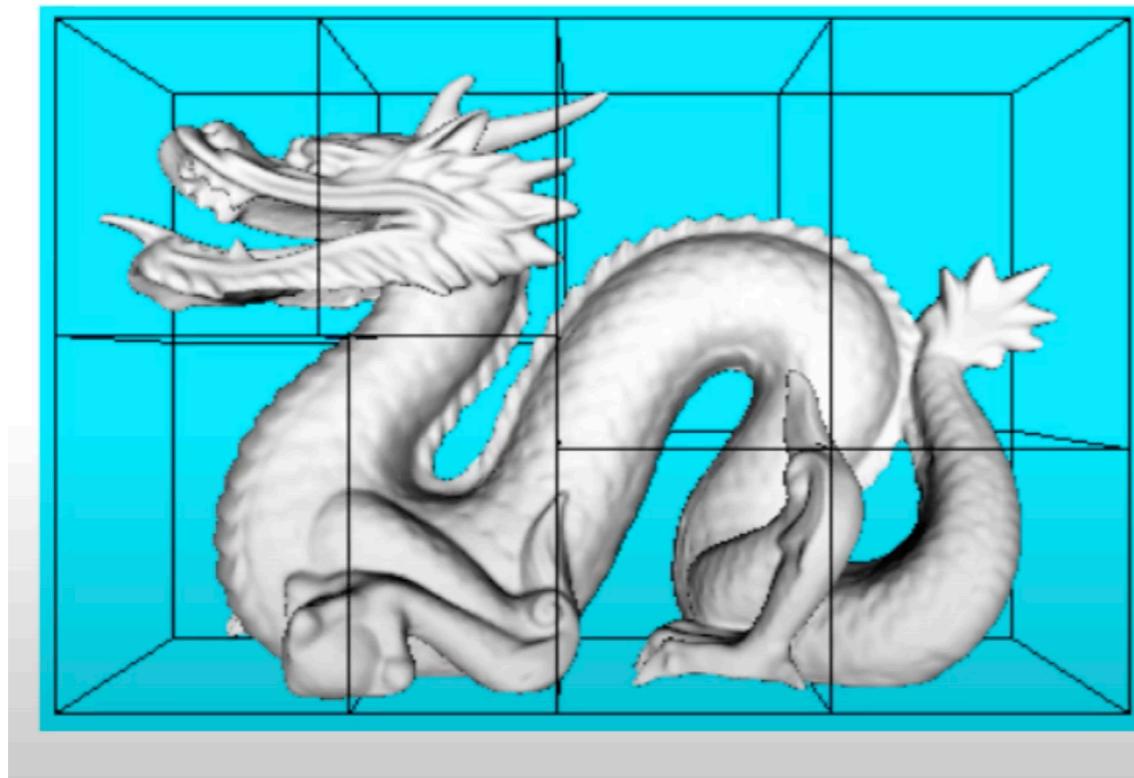
50,000 rays need 1 AABB test
950,000 rays need 3 AABB tests and
625K ray-triangle tests

We improved the speed significantly
(around 50%)

Costos Bounding Volumes



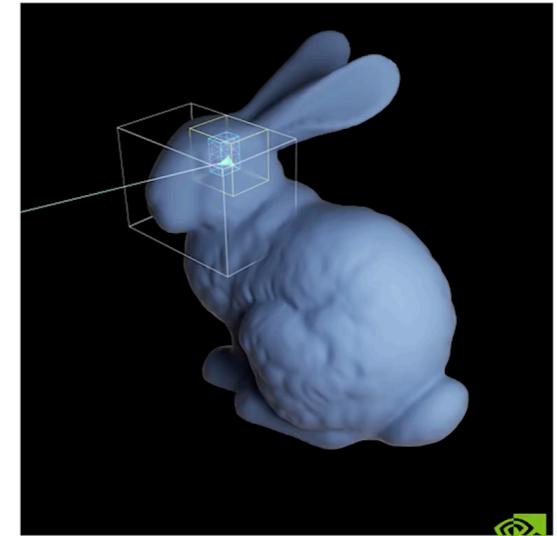
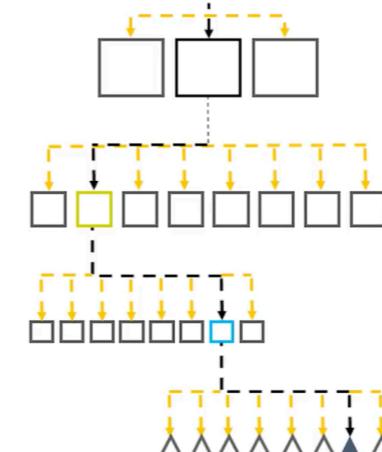
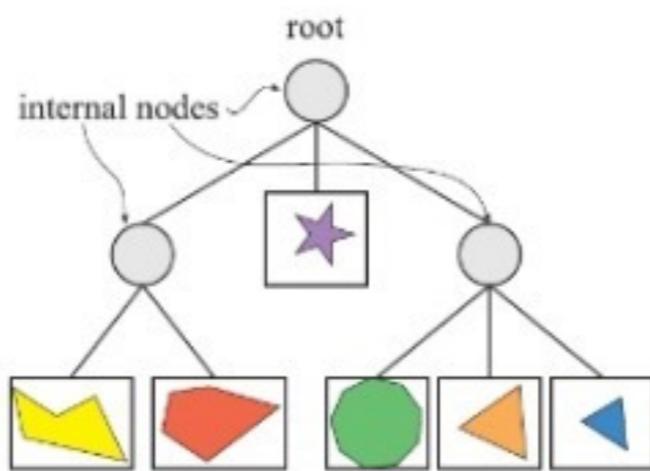
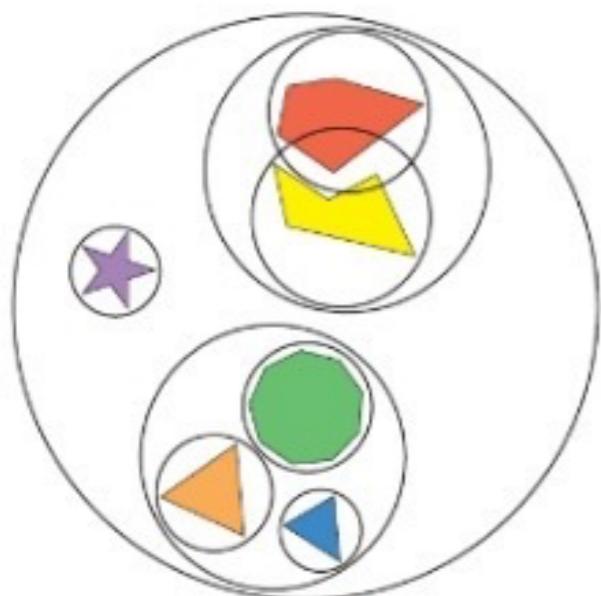
Costos Bounding Volumes



Cap triangle

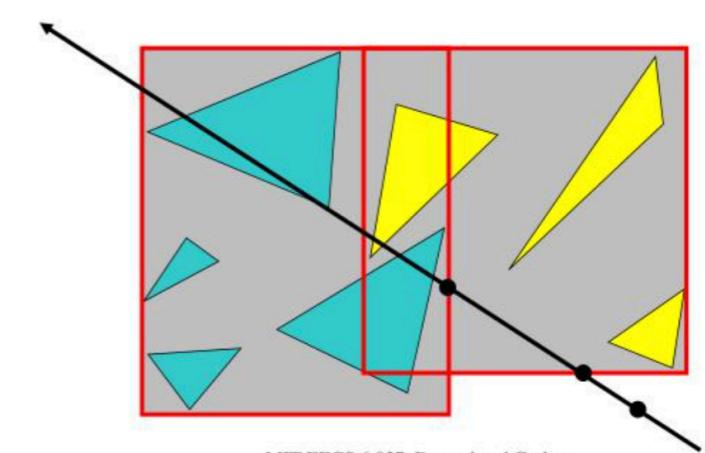
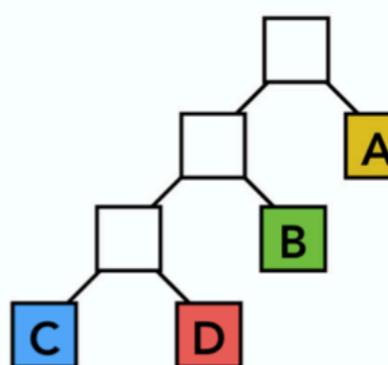
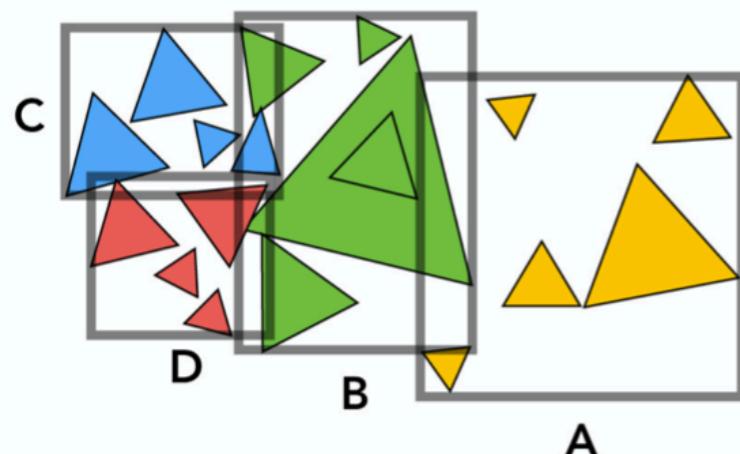
2. Estructures per accelerar (Escenes)

- Bounding Volumes Hierarchies:



imatge "Real-time Rendering"

Recórrer una BVH acostuma a tenir un cost de $O(\log N)$

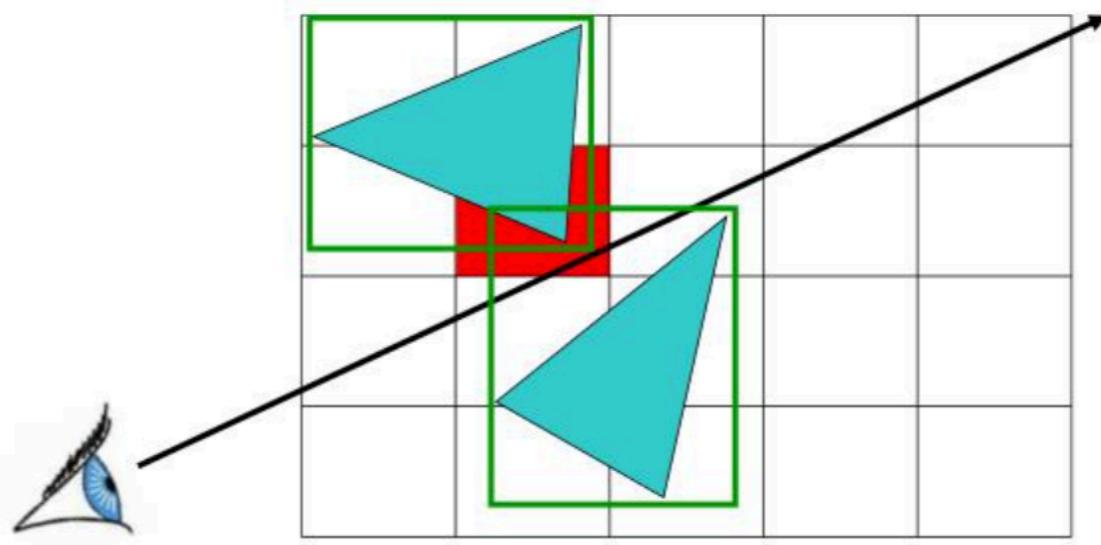
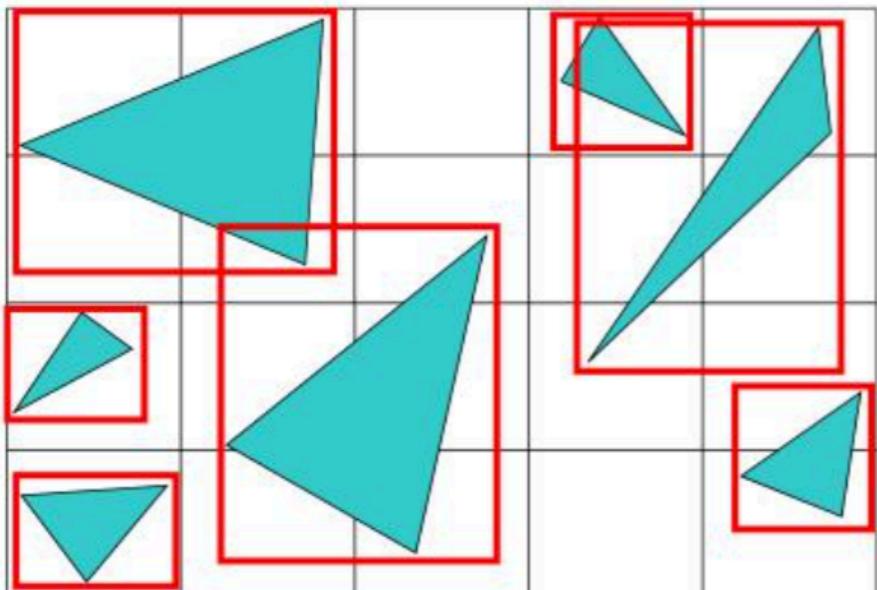
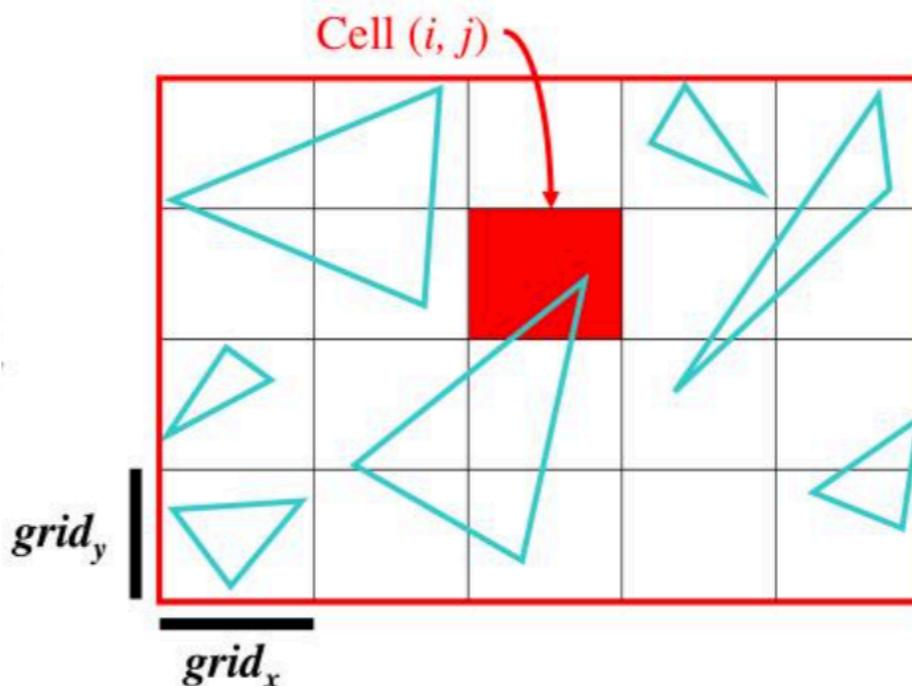


2. Estructures optimització

- **Regular Grid:**

- resolució

$(grid_x, grid_y, grid_z)$

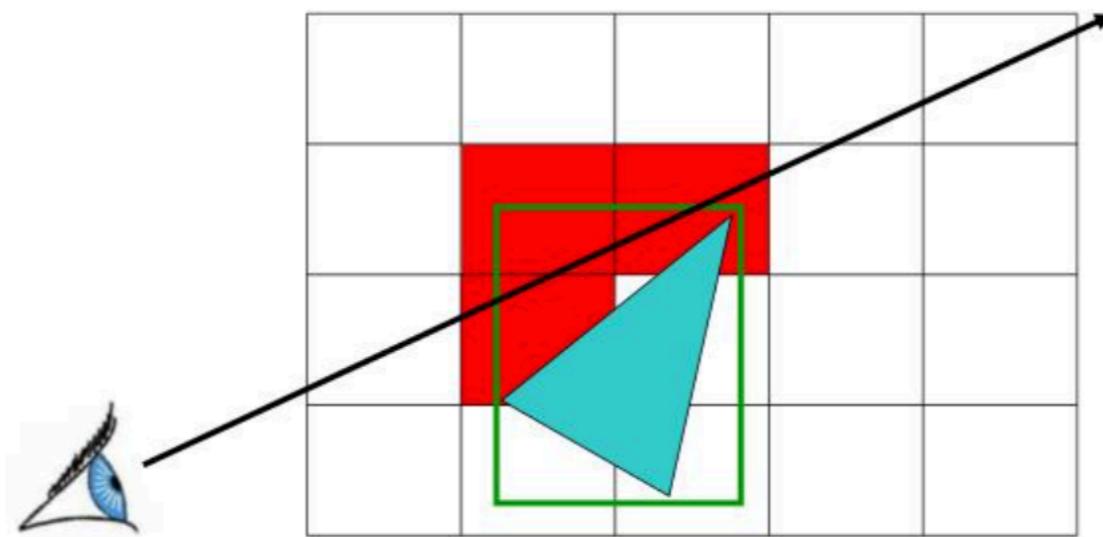
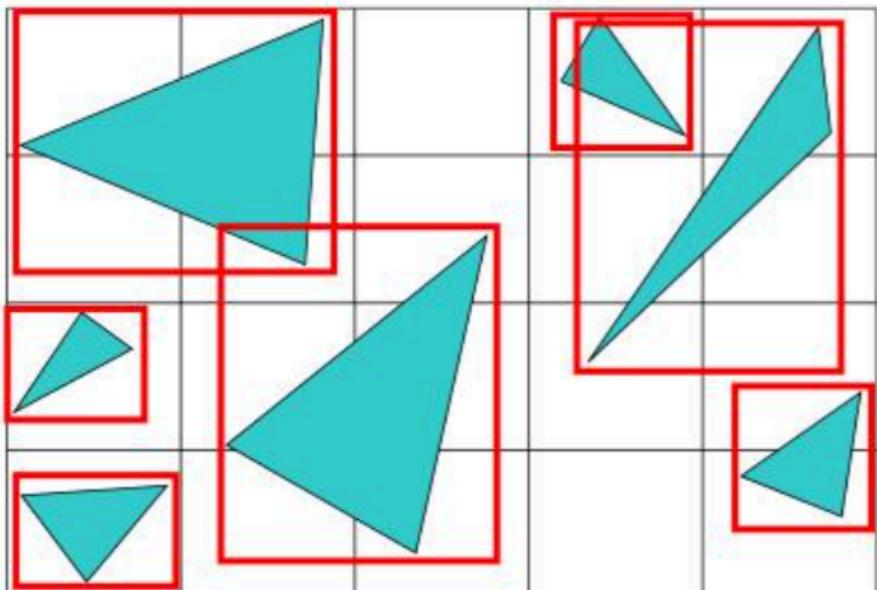
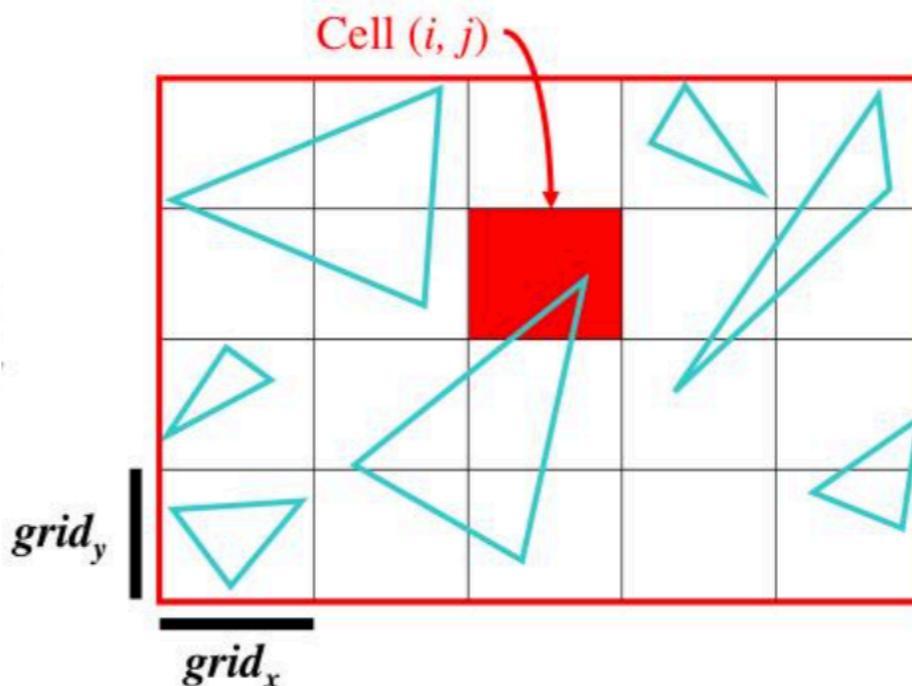


2. Estructures optimització

- **Regular Grid:**

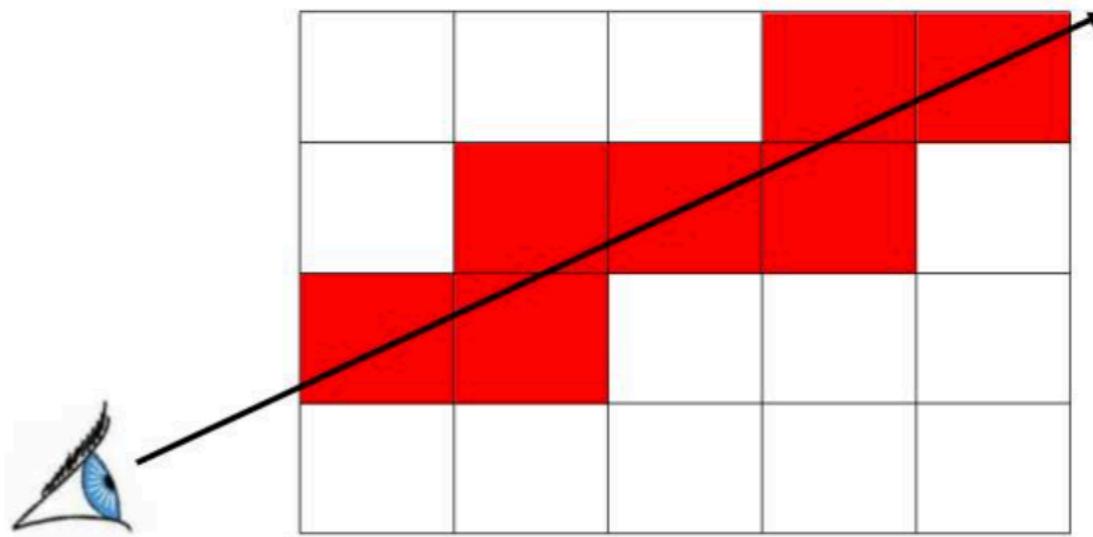
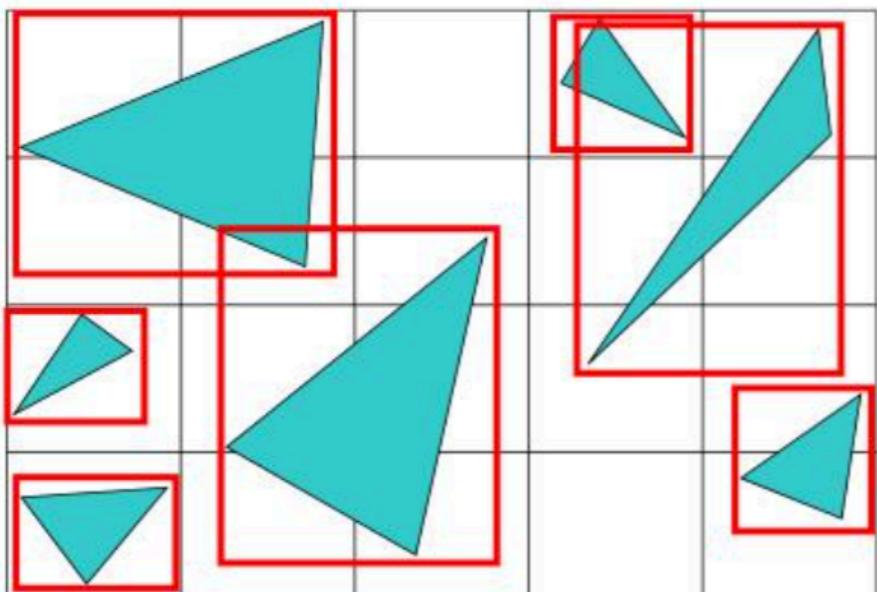
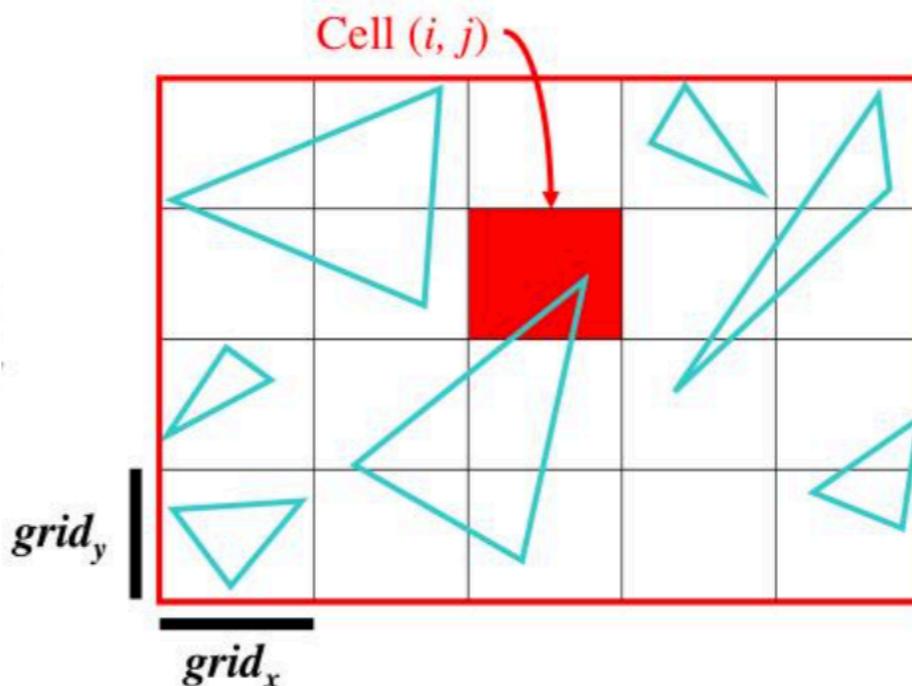
- resolució

$(grid_x, grid_y, grid_z)$



2. Estructures optimització

- **Regular Grid:**
 - resolució
 $(grid_x, grid_y, grid_z)$

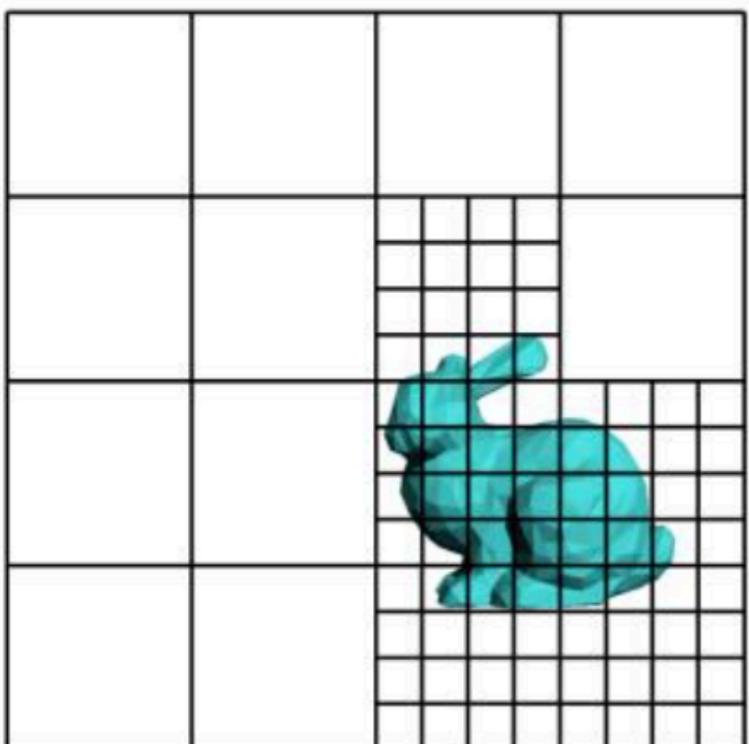


3DDDA - 3D Digital Difference Analyzer

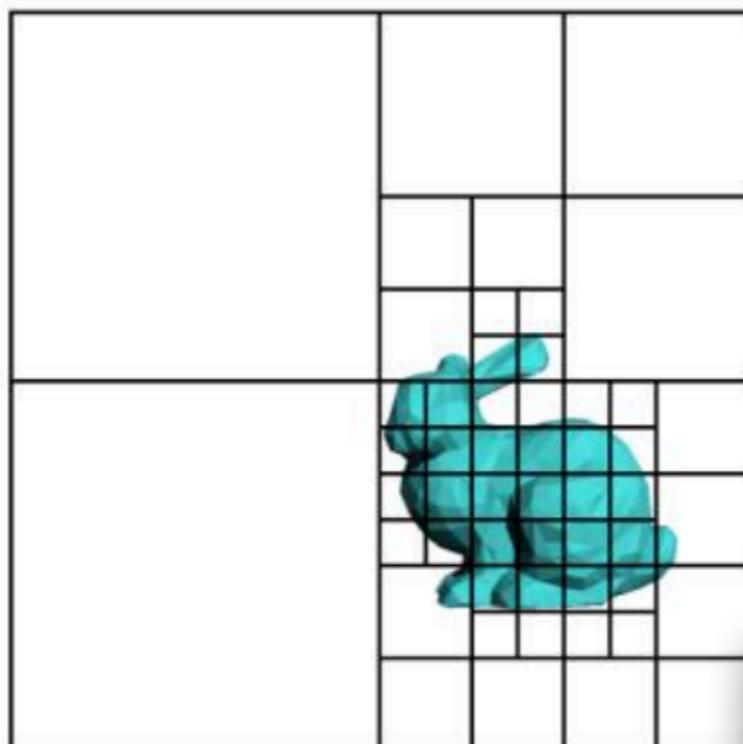
2. Estructures optimització

- **Adaptive Grids:**

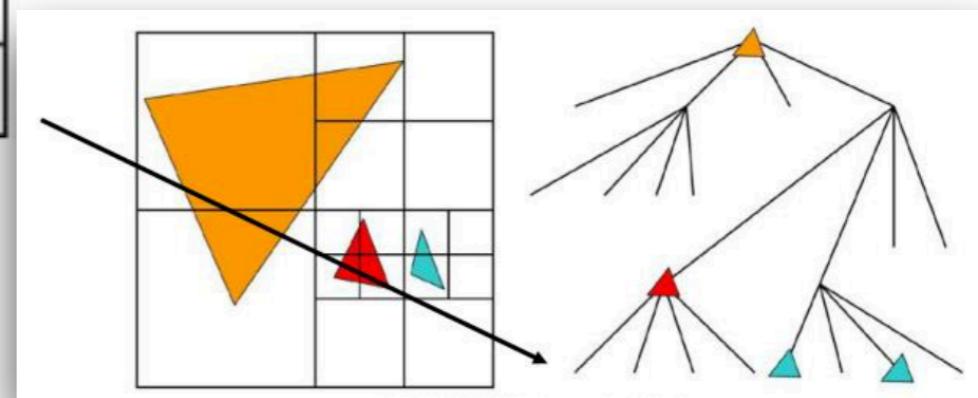
- subdivisió fins a que cada cel·la no conté més de n elements o s'arriba a una màxima profunditat



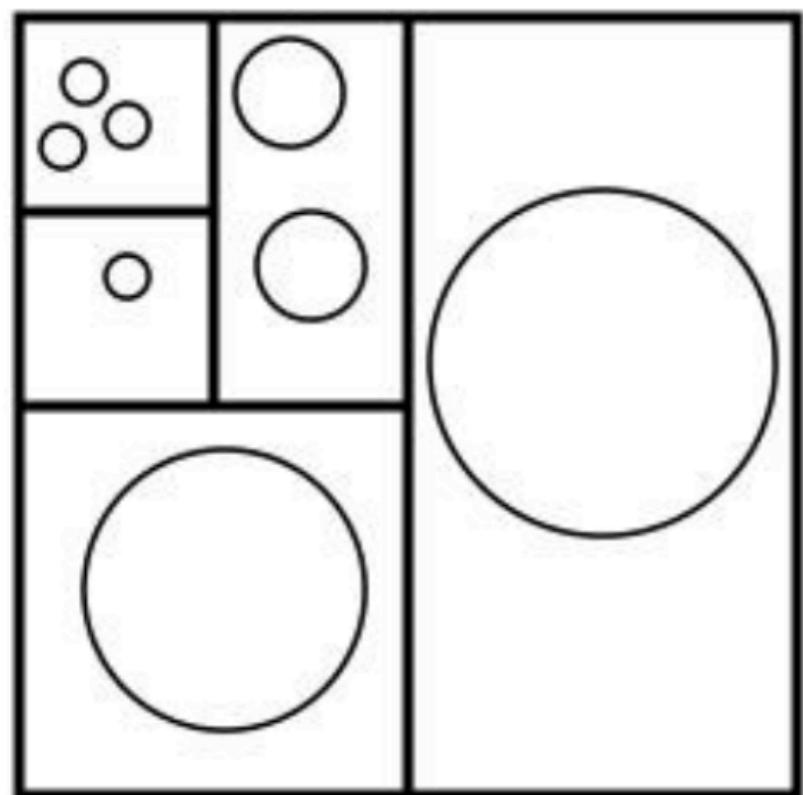
Nested Grids



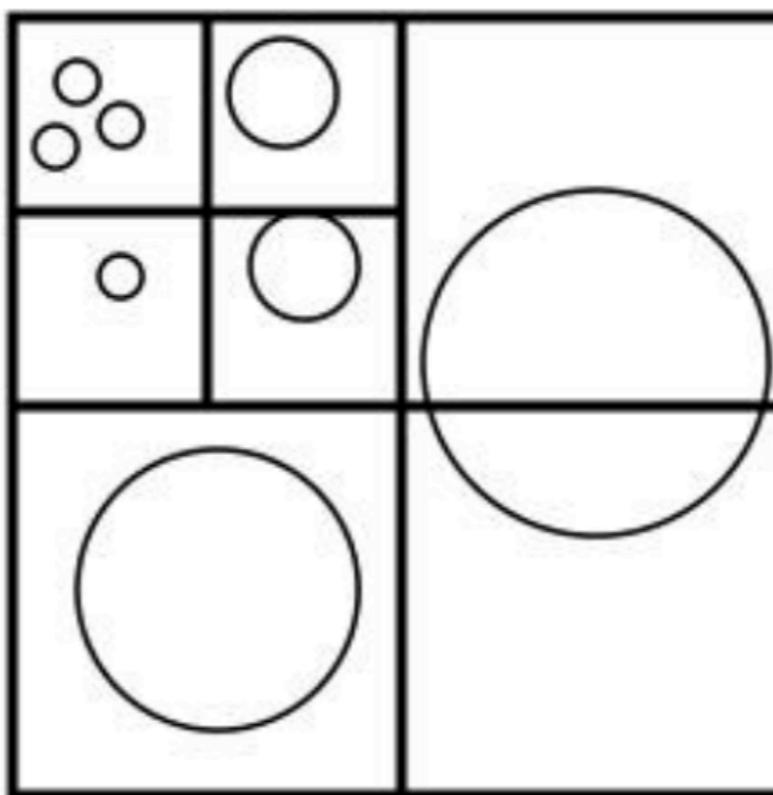
Octree/(Quadtree)



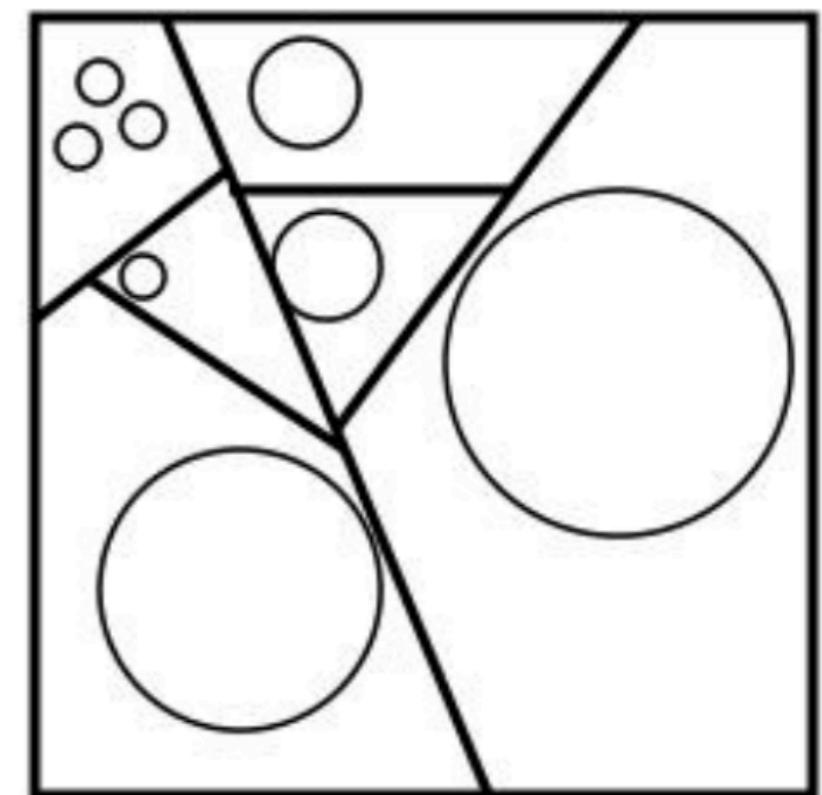
Bounding Volume Hierarchy



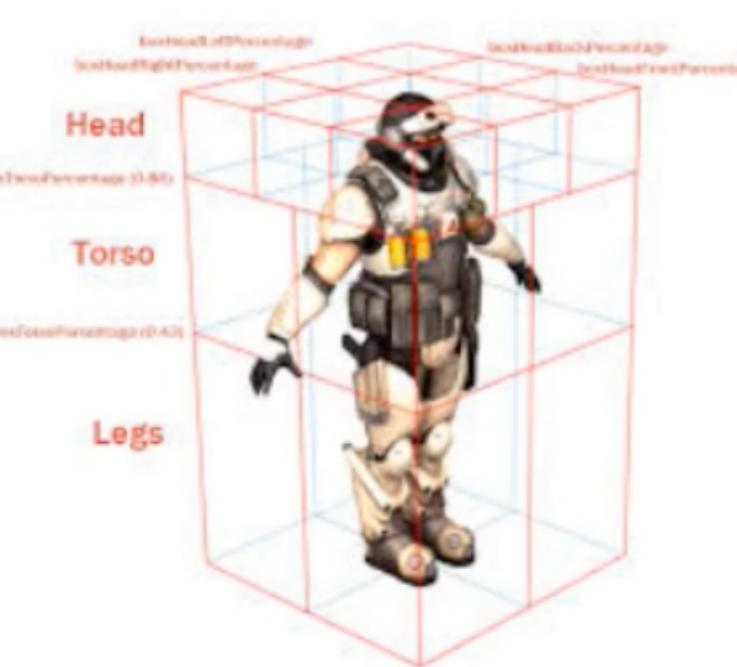
KD tree



octtree

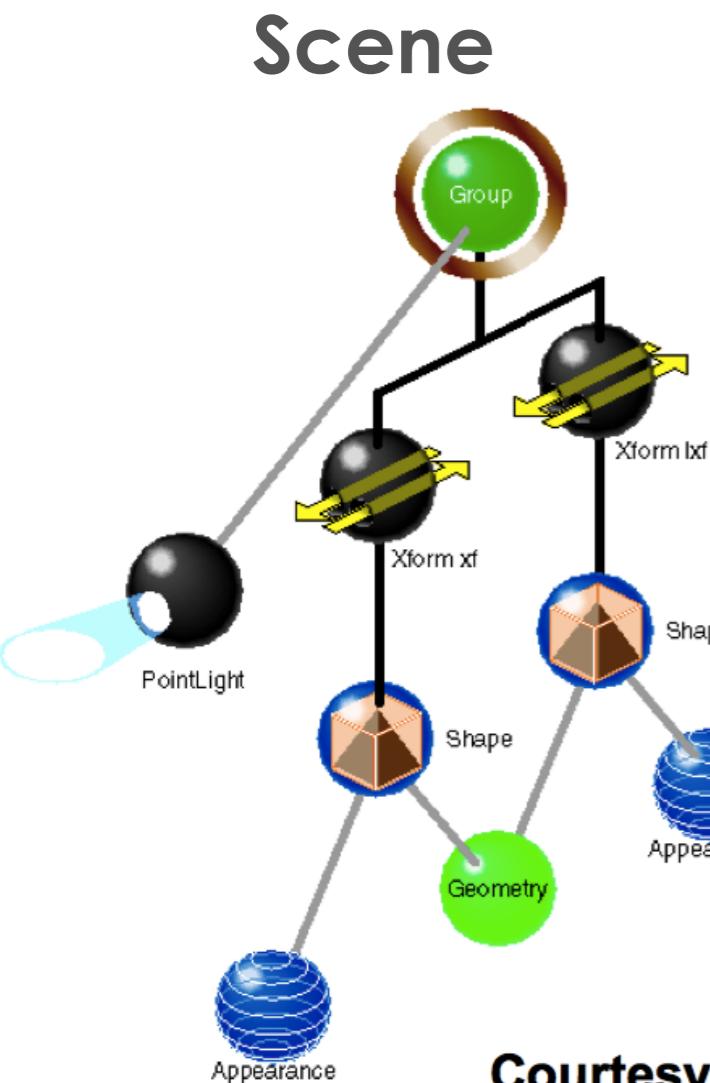


BSP tree

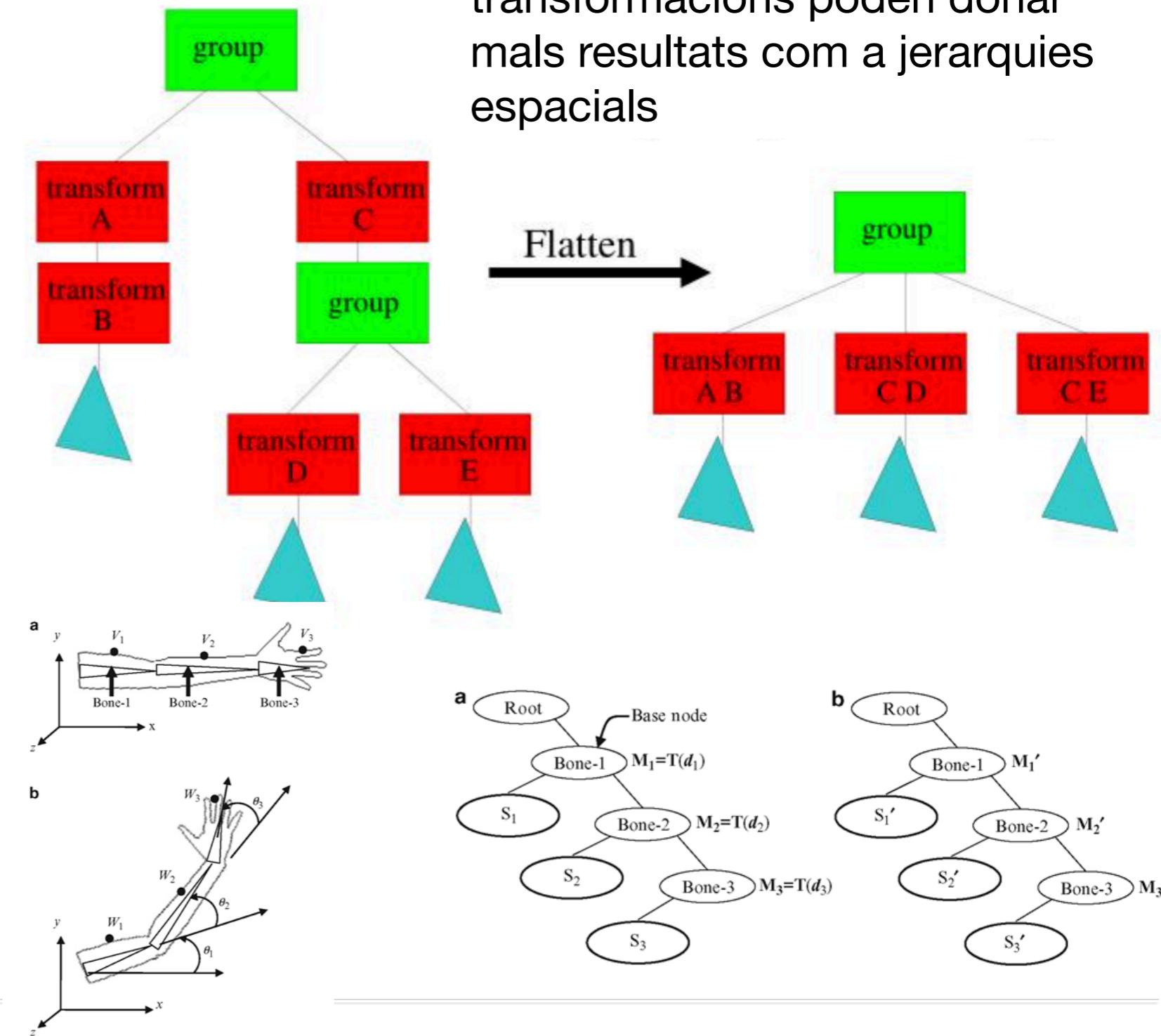


Com estructurar les escenes?

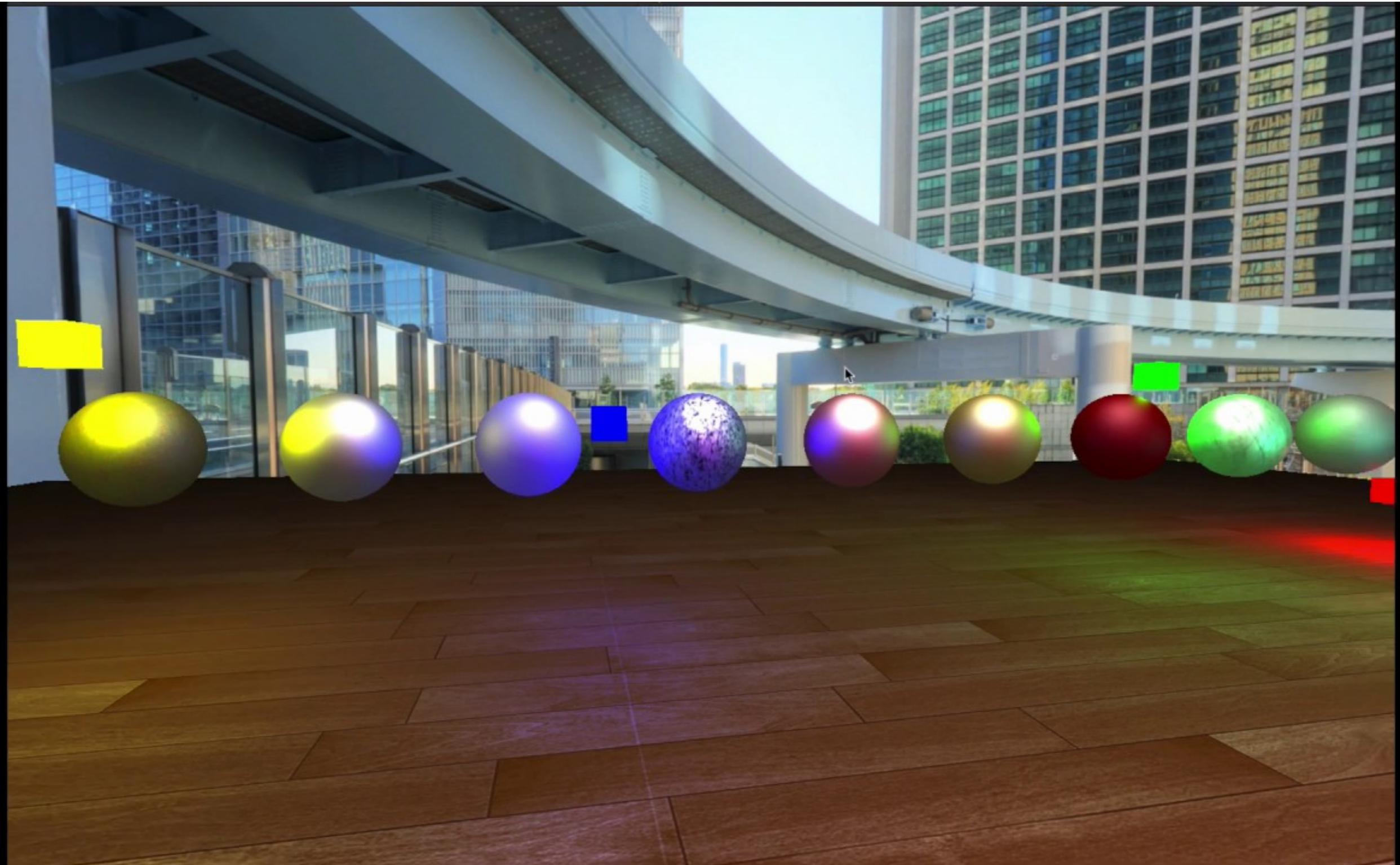
Jerarquies basades en transformacions poden donar mals resultats com a jerarquies espacials



Courtesy of SGI



3. Models d'il.luminació



3. Models d'il·luminació

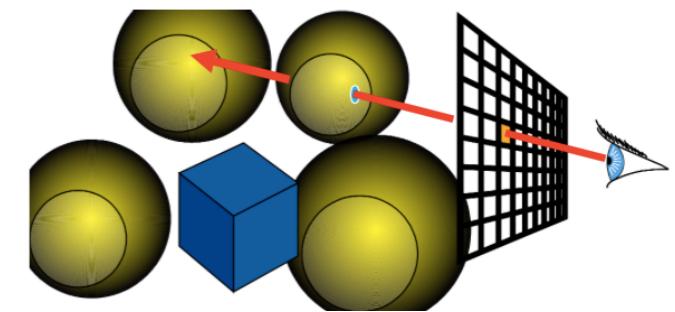
- **Raycast() // RayTracer.cpp**

per cada pixel x,y

 raig = camera->getRay(x, y)

 color(pixel) = **RayPixel (raig) // (computeColor(raig))**

 posaColorPixel(x, y, color)



- **RayPixel(ray) { // RayTracer.cpp**

 intersectInfo= **hit(ray) // hit(ray, tmin, tmax, hitInfo)**

 si intersecta retorna **shading(scene, hitInfo, lookFrom)**

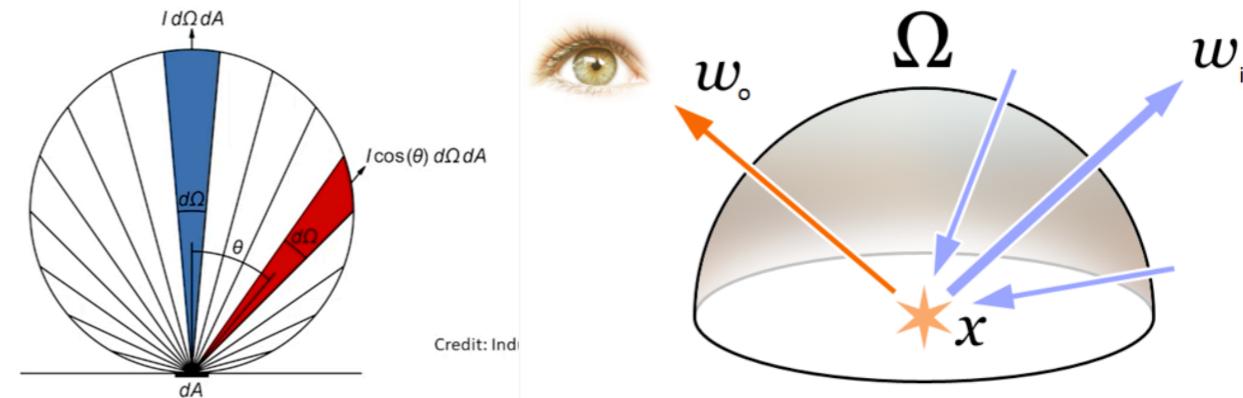
 sino retorna Background_Color // o Intensitat ambient global

}

- **Shade(scene, hitInfo, lookFrom) // Strategy Shading:** retorna el color en el punt intersecció

 retorna color usant fórmula de **Blinn-Phong o segons Phong o Color o Cell Shading**

L'equació del rendering

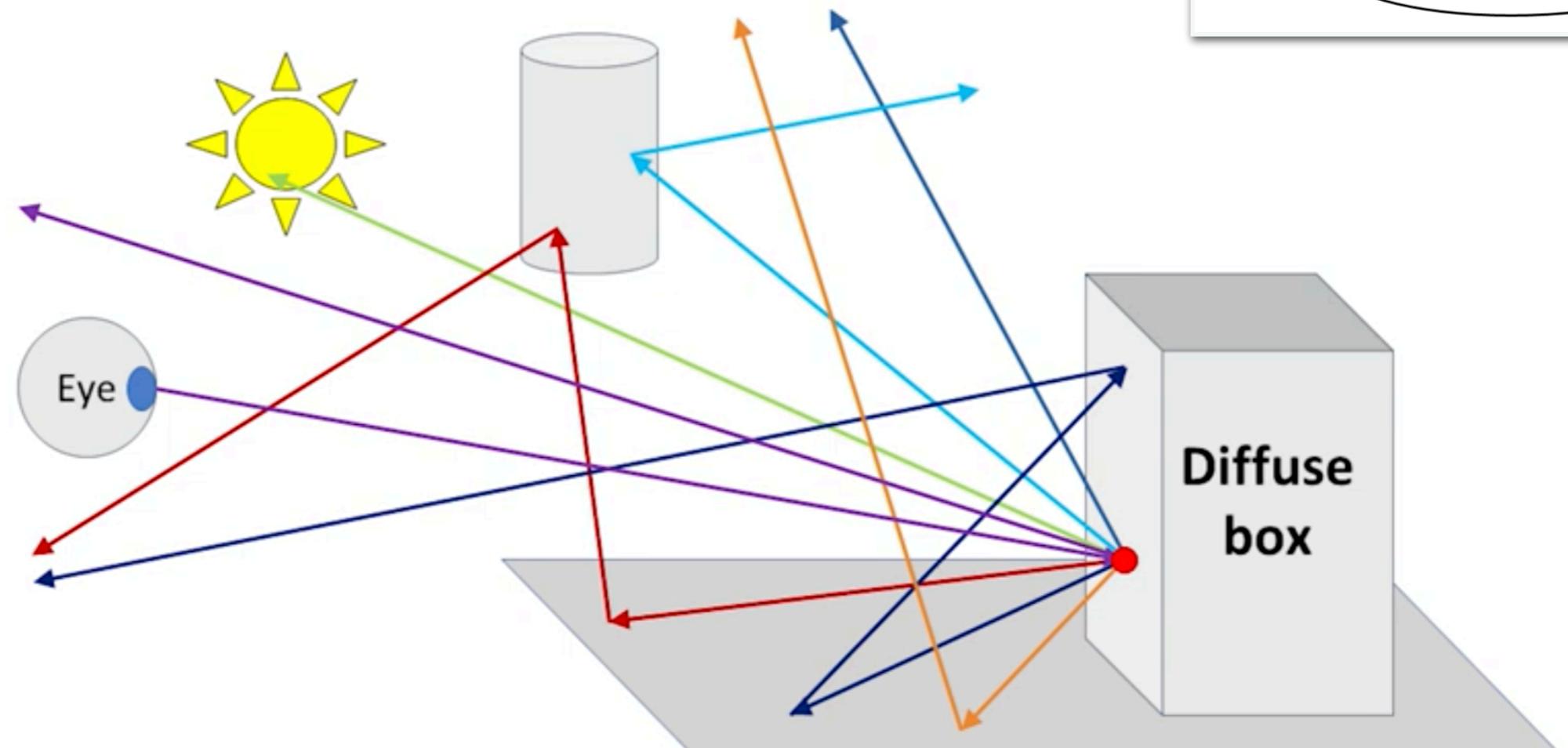
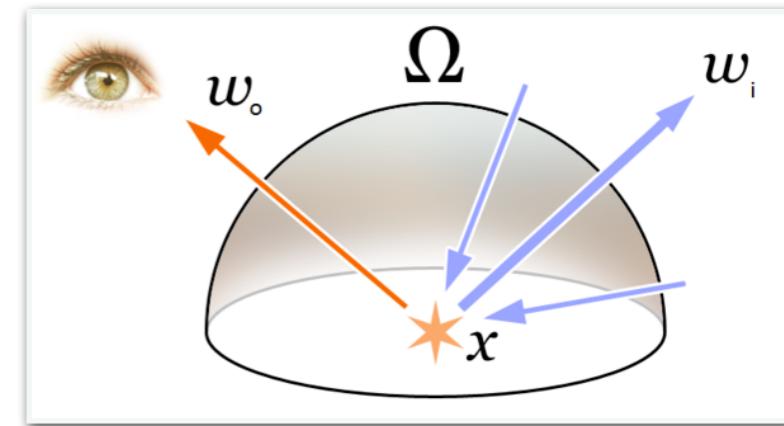


<p>Outgoing direction</p> $L_o(X, \hat{\omega}_o) = L_e(X, \hat{\omega}_o) + \int_{\mathbf{S}^2} L_i(X, \hat{\omega}_i) f_X(\hat{\omega}_i, \hat{\omega}_o) \hat{\omega}_i \cdot \hat{n} d\hat{\omega}_i$ <p>A point in the scene</p> <p>All incoming directions (a sphere)</p>	<p>Incoming direction</p> <p>Surface normal</p>
---	---

$$L_o(X, \hat{\omega}_o) = L_e(X, \hat{\omega}_o) + \int_{\mathbf{S}^2} L_i(X, \hat{\omega}_i) f_X(\hat{\omega}_i, \hat{\omega}_o) |\hat{\omega}_i \cdot \hat{n}| d\hat{\omega}_i$$

Outgoing light	Emitted light	Incoming light	Material	Lambert
----------------	---------------	----------------	----------	---------

L'equació del rendering



$$L_o(X, \hat{\omega}_o) = L_e(X, \hat{\omega}_o) + \int_{\mathbf{S}^2} L_i(X, \hat{\omega}_i) f_X(\hat{\omega}_i, \hat{\omega}_o) |\hat{\omega}_i \cdot \hat{n}| d\hat{\omega}_i$$

Outgoing light

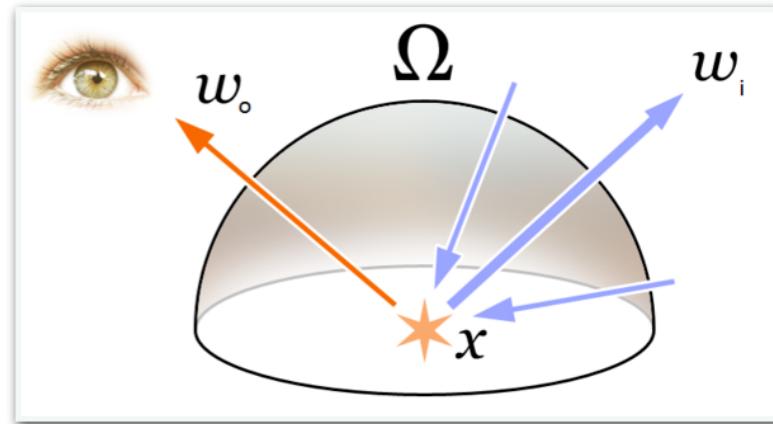
Emitted light

Incoming light

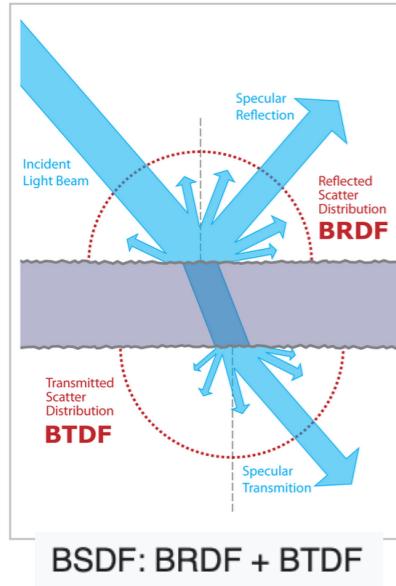
Material

Lambert

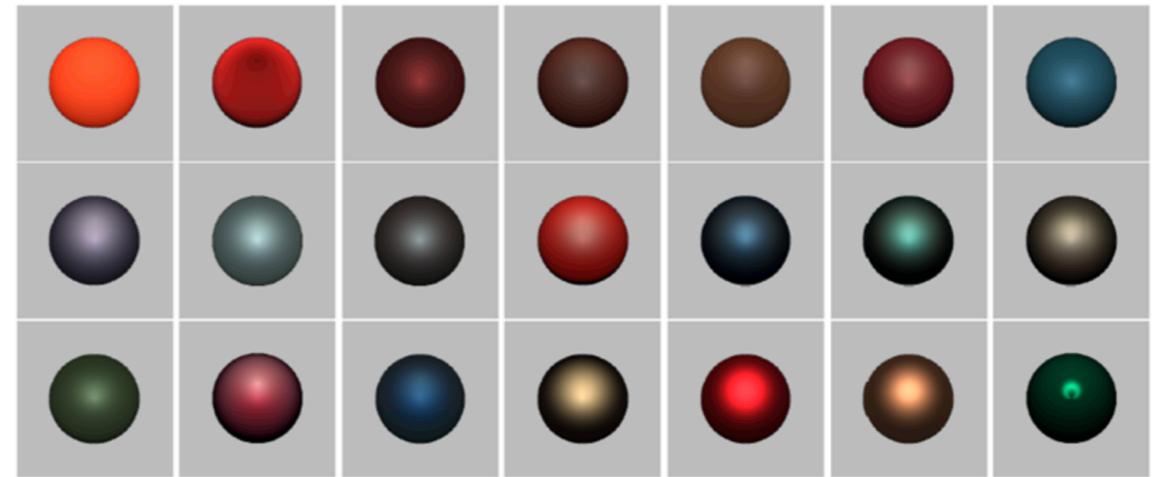
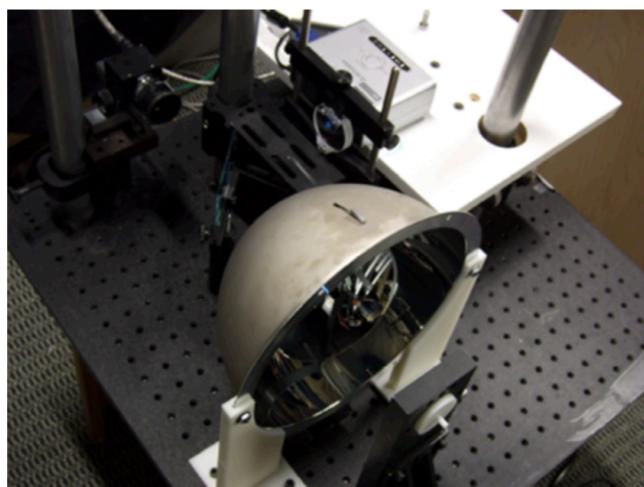
L'equació del rendering



$$L_o(X, \hat{\omega}_o) = L_e(X, \hat{\omega}_o) + \int_{\mathbf{S}^2} L_i(X, \hat{\omega}_i) f_X(\hat{\omega}_i, \hat{\omega}_o) |\hat{\omega}_i \cdot \hat{n}| d\hat{\omega}_i$$



Vary sampling direction depending on bidirectional scattering distribution function (BSDF) and angle



3. Models d'il·luminació

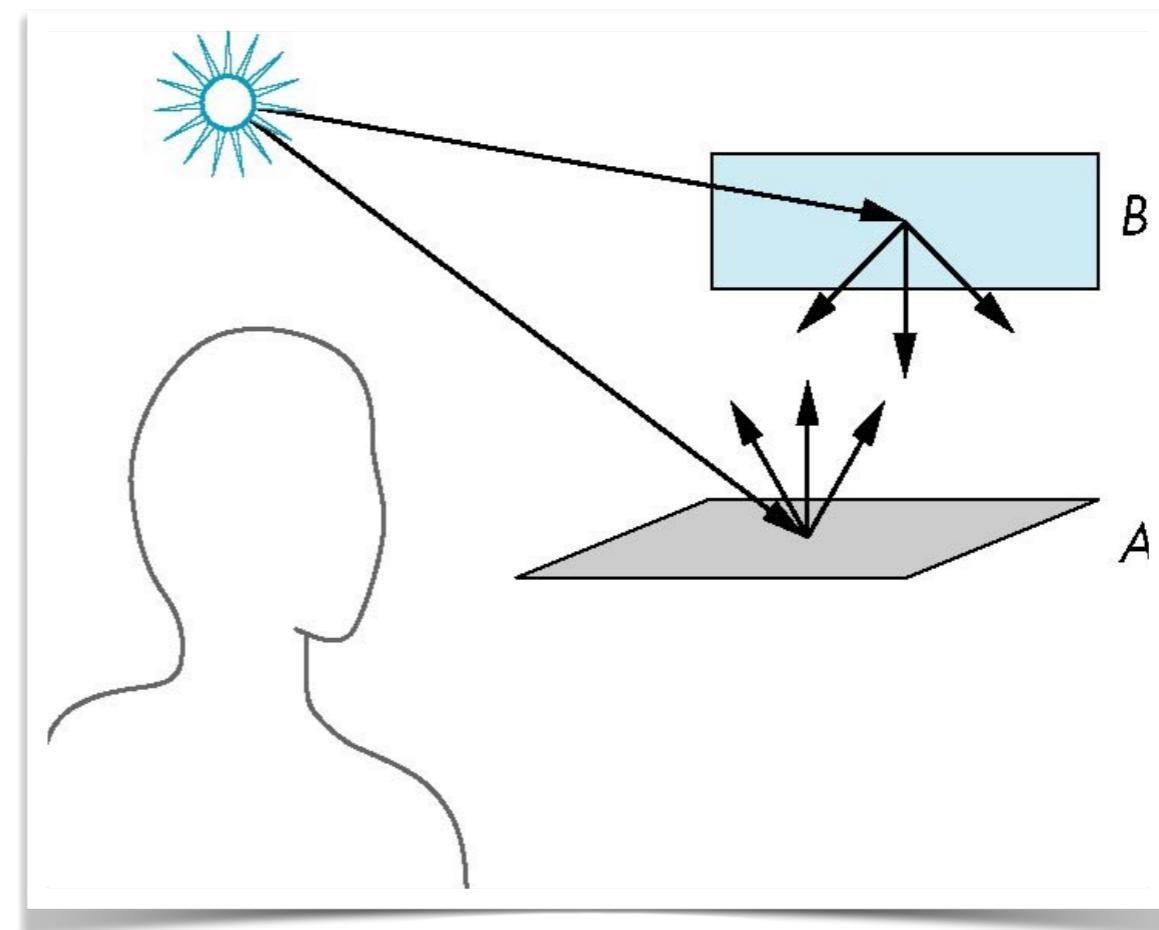
- Mecanismes de comportament de la llum

absorció

emissió

reflexió

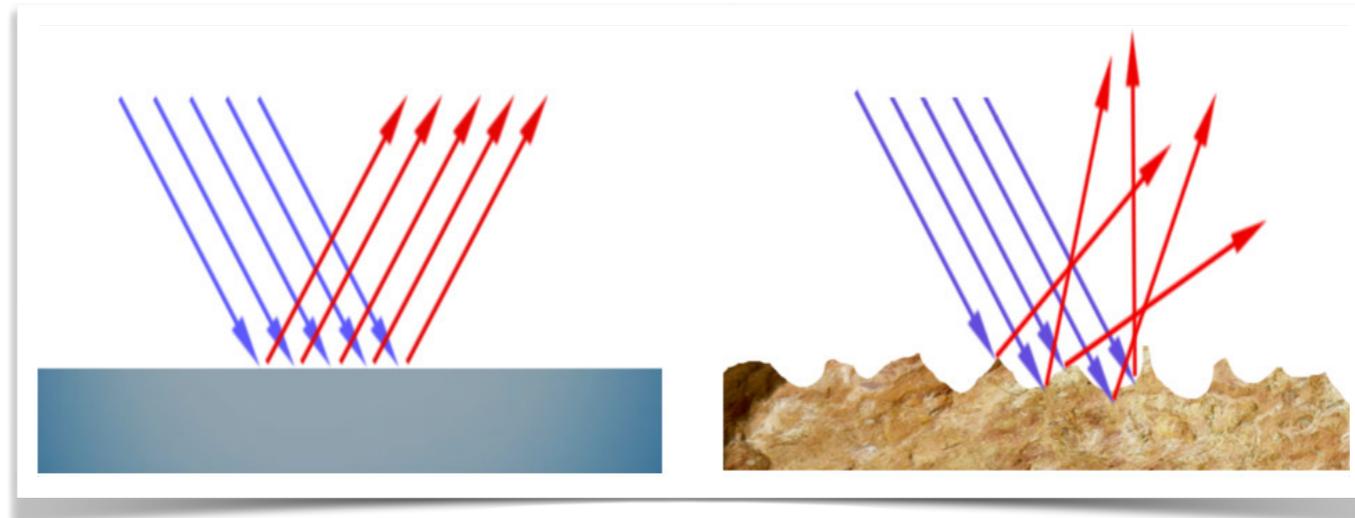
transmissió



3. Models d'il·luminació

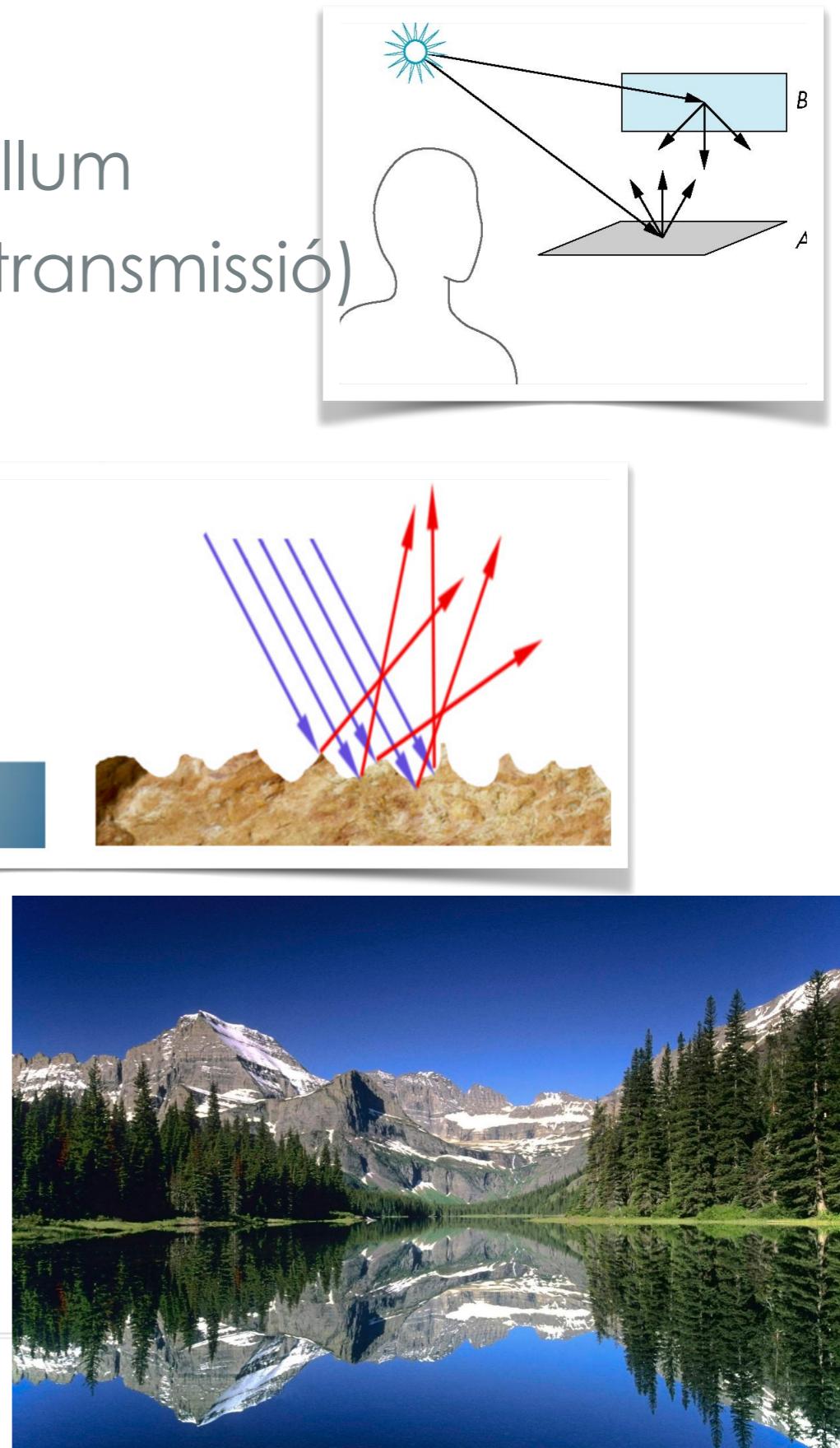
- Mecanismes de comportament de la llum
(absorció, emissió, reflexió (scatter), transmissió)
- Tipus de Reflexions:

- difuses
- especulars



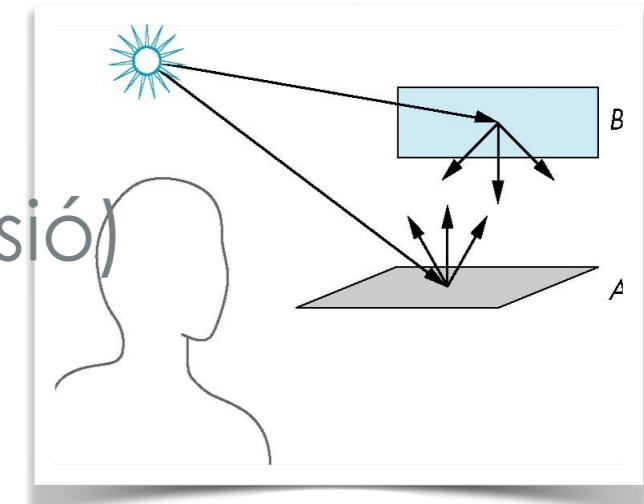
Demostrador de rugositats:

<https://www.shadertoy.com/view/4sSfzK>

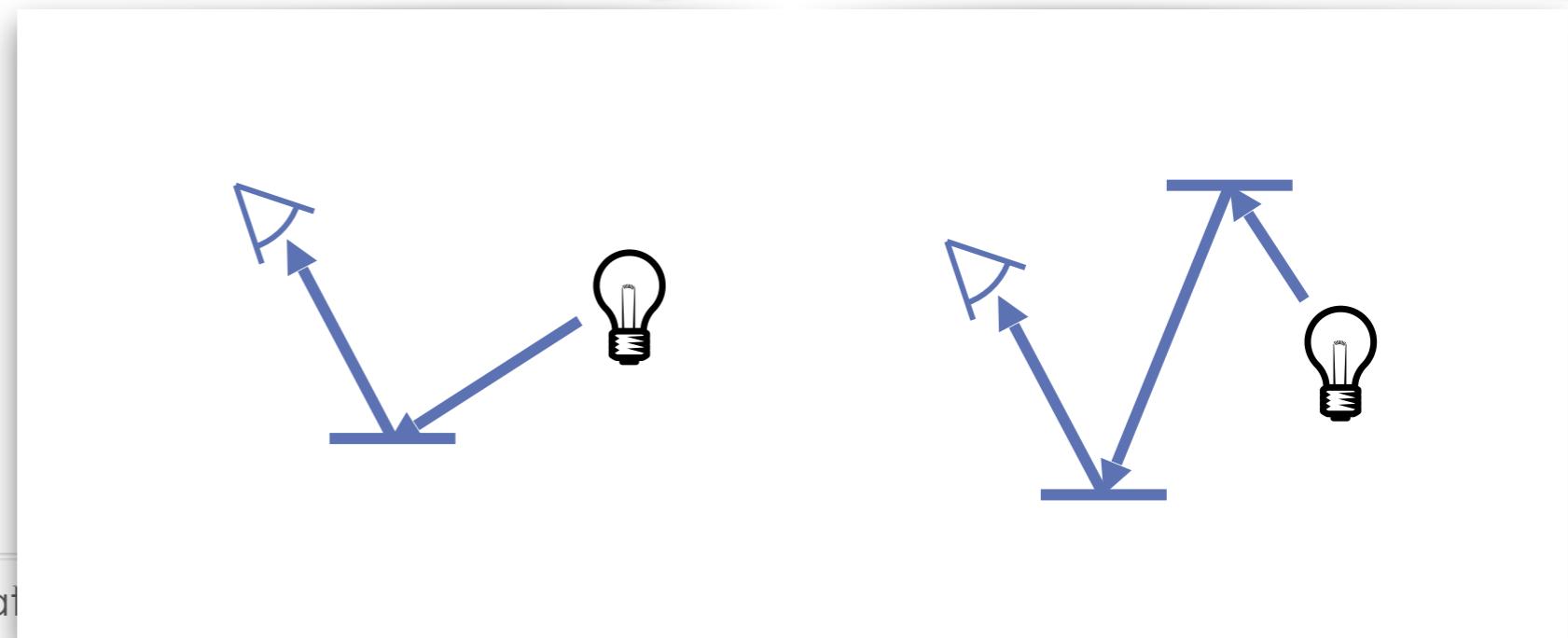
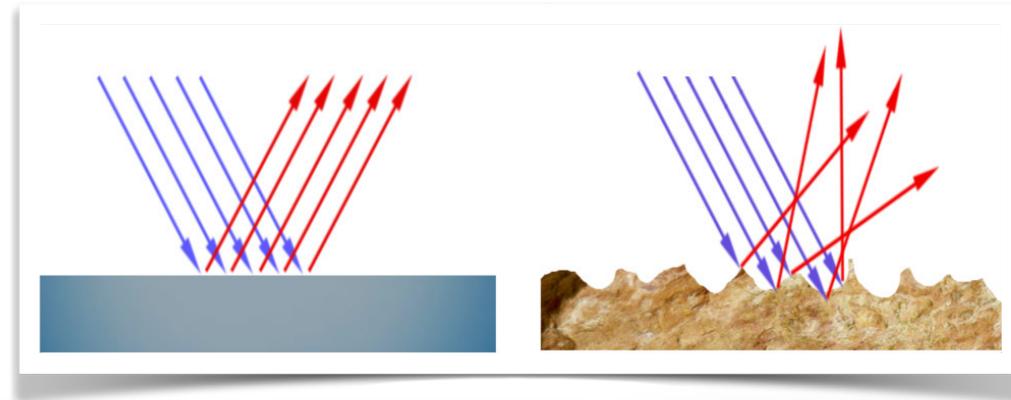


3. Models d'il·luminació

- Mecanismes de comportament de la llum
(absorció, emissió, reflexió (scatter), transmissió)
- Tipus de Reflexions: difuses/especulars

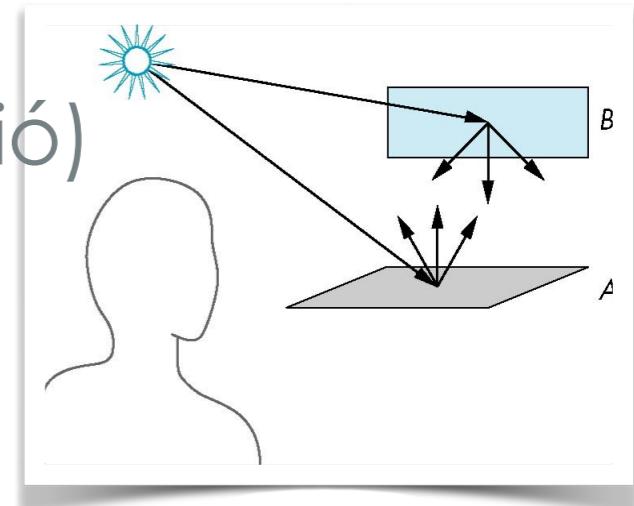


- Llum
 - directa
 - indirecta



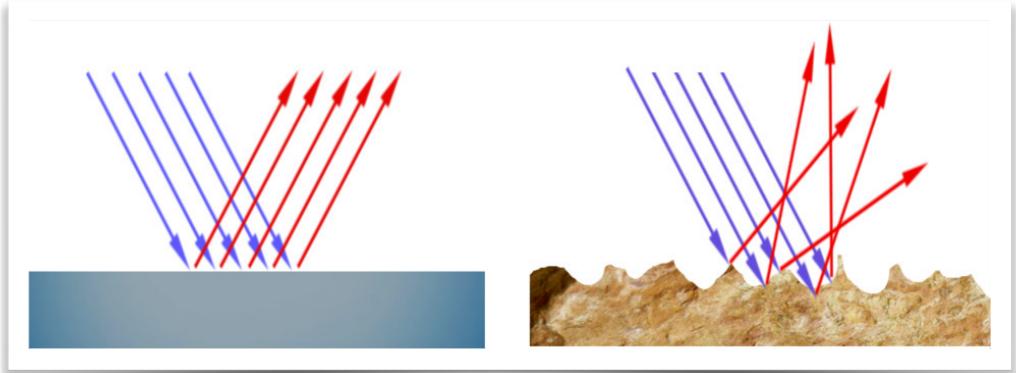
3. Models d'il·luminació

- Mecanismes de comportament de la llum
(absorció, emissió, reflexió (scatter), transmissió)



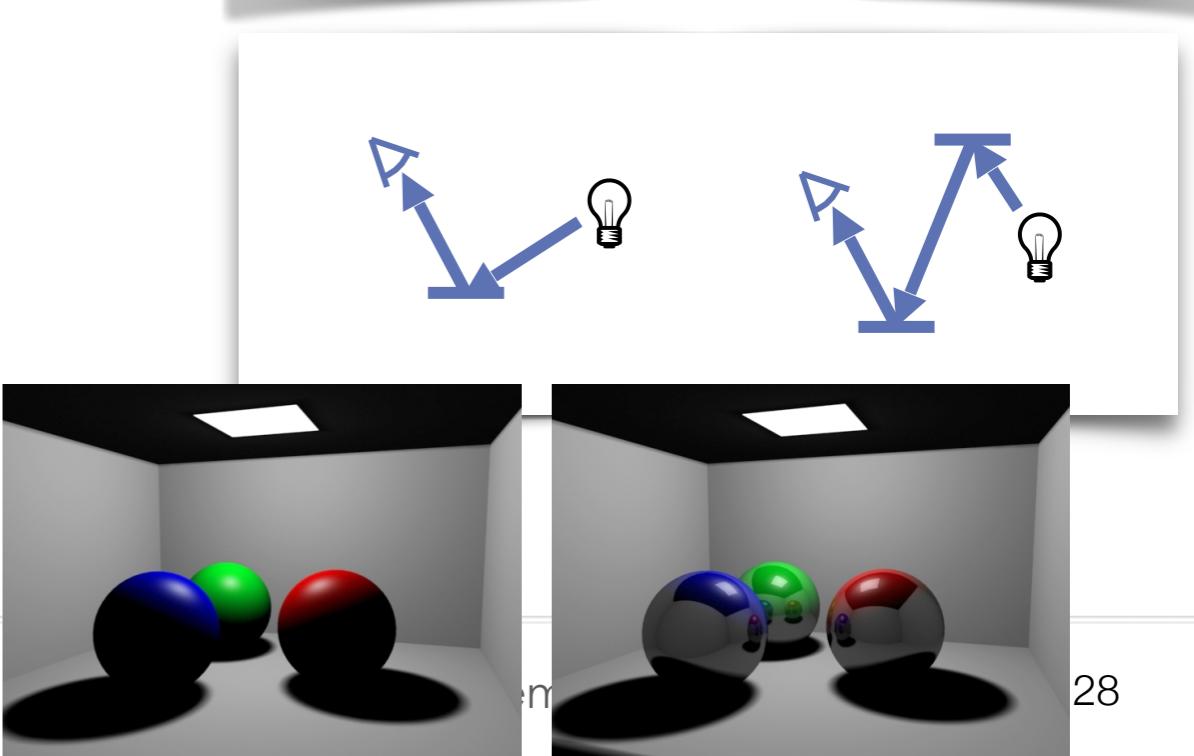
- Tipus de Reflexions: difuses/especulars

- Llum directa/indirecta



- Càlculs

- locals
- globals



3. Models d'il·luminació

- Llums: I_a , I_d , I_r (R , G , B)

Tipus	Exemples / Imatges
Ambient global	
Puntual	
Direccional	
Spot light	 Classe Light.h

- Propietats que defineixen un material



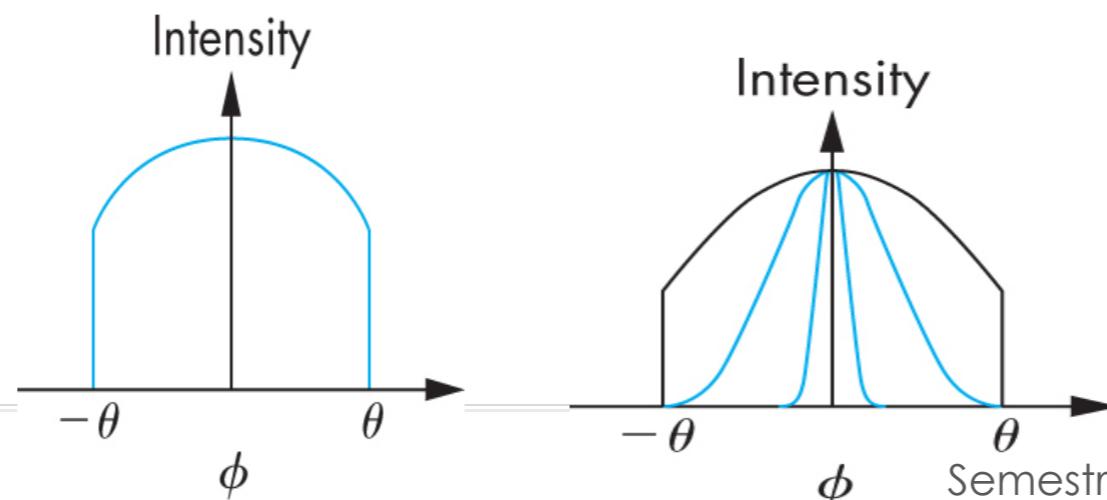
- Component **difusa** K_d (R , G , B)
- Component **ambient** K_a (R , G , B)
- Component **especular** K_s (R , G , B)
- Exponent de **reflexió especular** (shininess) $a_s = 1 \dots 500$

Classe Material.h

Llums i atenuació

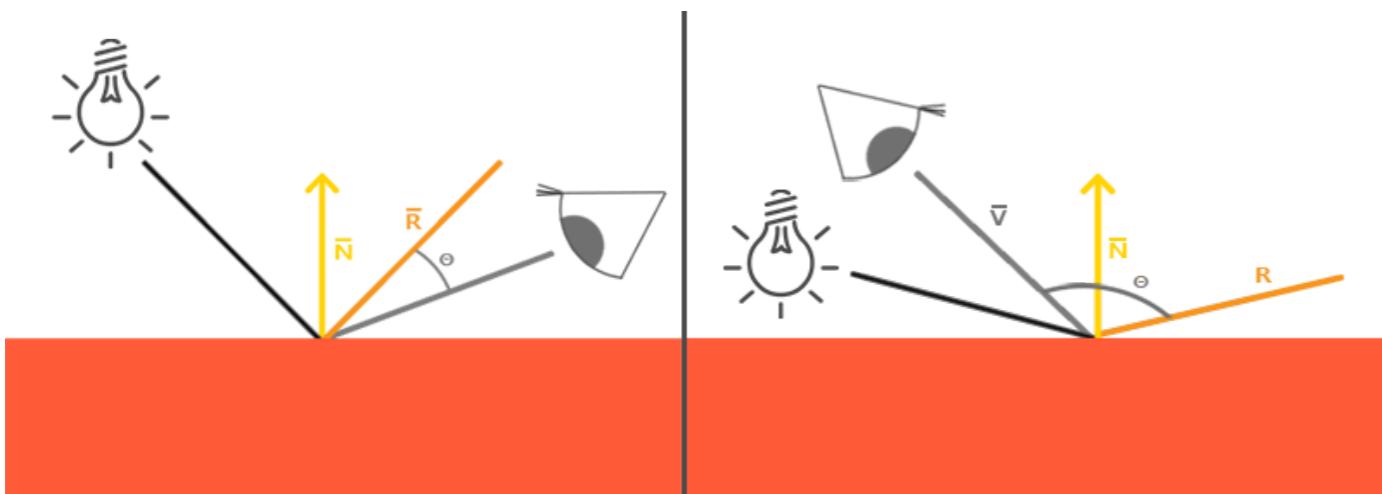
Atenuació en profunditat:

- La quantitat de llum rebuda per una superfície és inversament proporcional a la distància de la superfície al focus de llum $1/d^2$
- Es pot aplicar un factor de $1/(a + bd + cd^2)$ a les components de la llum, tan difusa com especular, que arriben a la superfície, on a, b, c són paràmetres que es determinen segons l'escena.
- en spotlights l'atenuació pot ser concentrada al centre del con, exponencial



3. Phong vs. Blinn-Phong

Model de Phong



$$\vec{R} = 2\vec{N}(\vec{N} \cdot \vec{L}) - \vec{L}$$

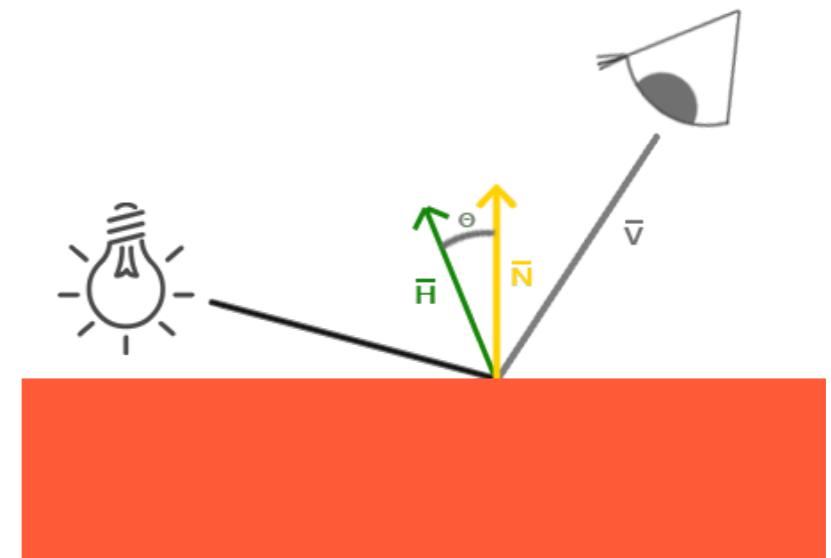
$$I \sim k_s * I_s * \cos^\alpha \phi$$

$\cos \phi = \vec{R} \cdot \vec{V}$ (amb els vectors normalitzats)

<https://www.desmos.com/calculator/5k43xjrgd8>

Pere Dolcet, 2022

Model de Blinn-Phong



$$\vec{H} = (\vec{L} + \vec{V}) / |\vec{L} + \vec{V}|$$

$$I \sim k_s * I_s * \cos^\beta \theta$$

$\cos \theta = \vec{N} \cdot \vec{H}$ (amb els vectors normalitzats)

<https://www.desmos.com/calculator/sct0asktyy>

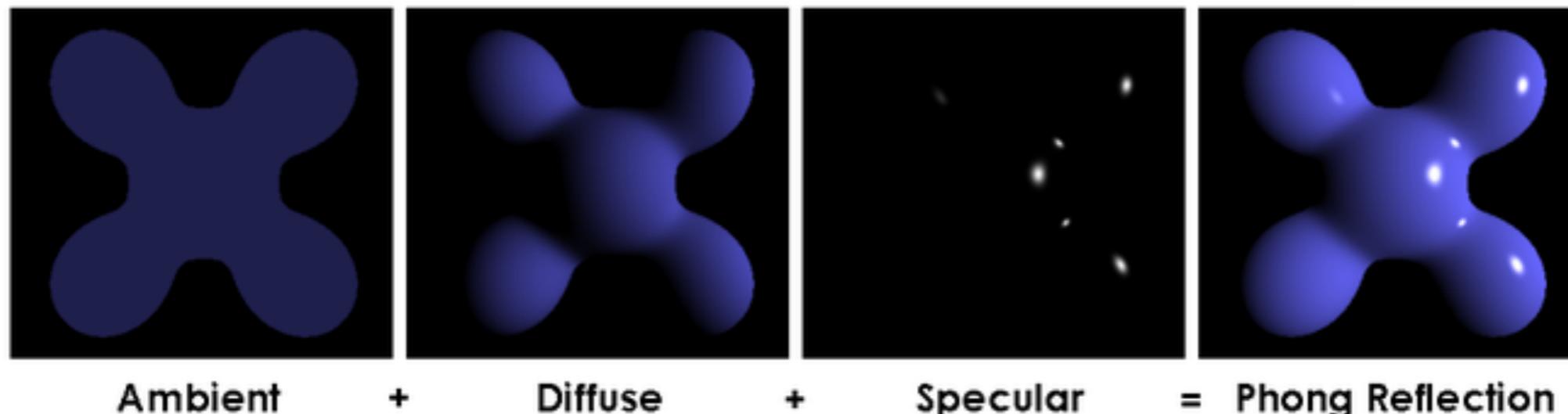
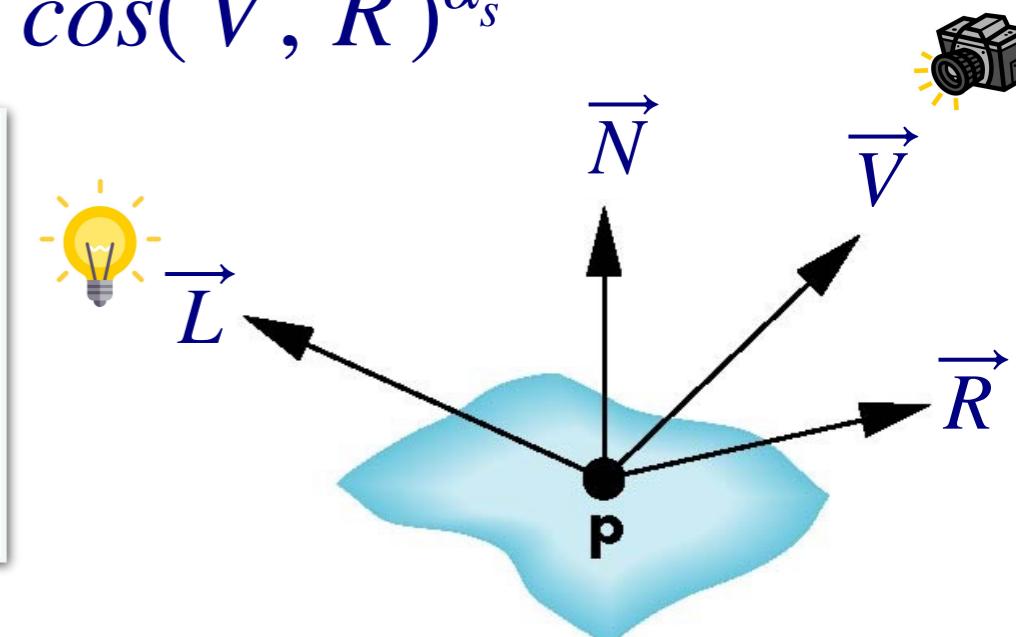
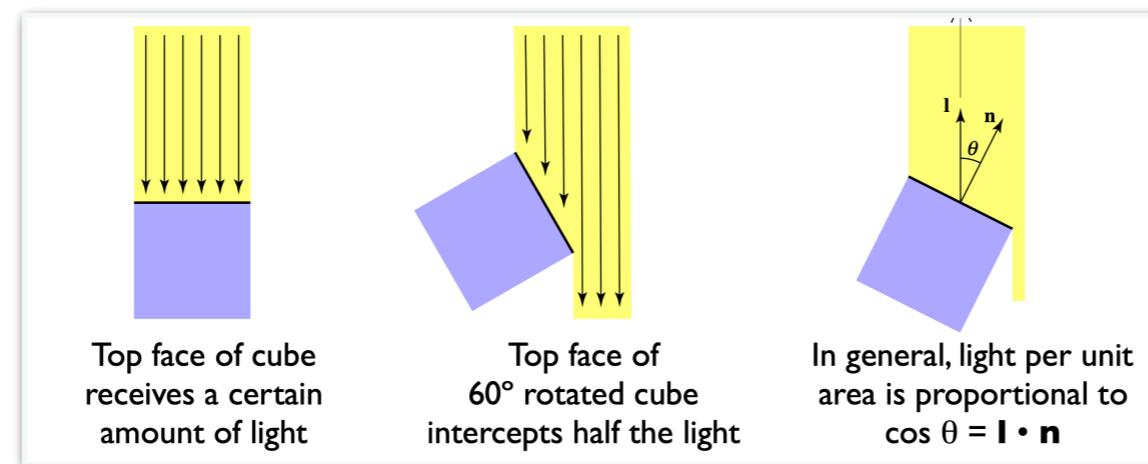
Pere Dolcet, 2022

$$I_{total} = I_{a_{global}} K_a + \sum_{i=1}^{numLlums} \frac{1.0}{a_i + b_i d_i + c_i d_i^2} (I_{d_i} K_d \max(L_i \cdot N, 0.0) + I_{s_i} K_s \max((N \cdot H_i), 0.0)^\beta) + I_{a_i} K_a$$

Model de Phong

Per a una llum, el model de Phong es pot escriure com:

$$I = k_a I_a + k_d I_d \cos(\vec{L}, \vec{N}) + k_s I_s \cos(\vec{V}, \vec{R})^{\alpha_s}$$



<http://multivis.net/lecture/phong.html>

$$\mathbf{R} = 2 (\mathbf{L} \cdot \mathbf{N}) \mathbf{N} - \mathbf{L}$$

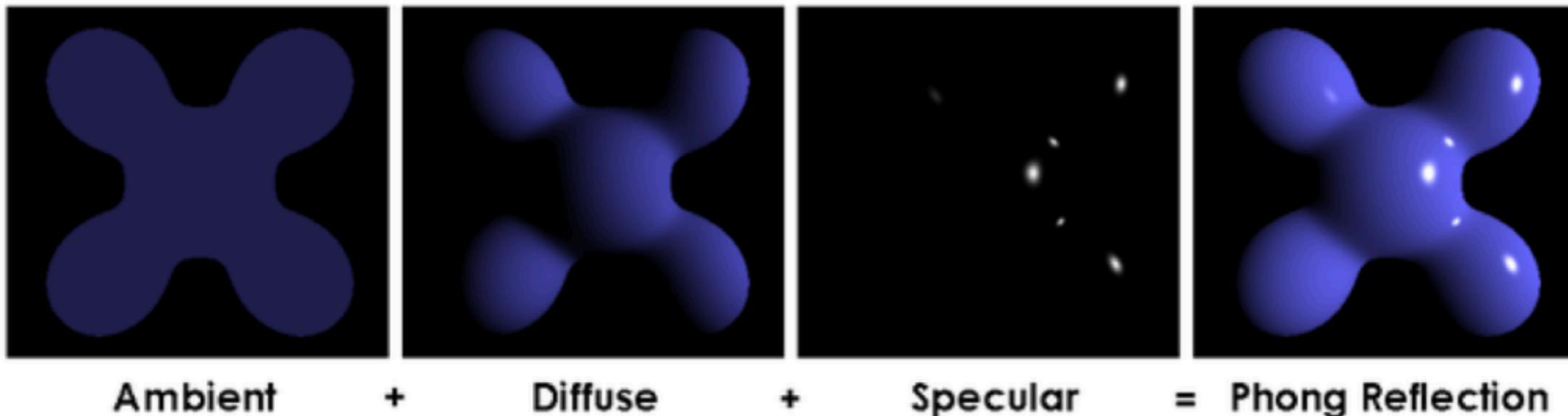
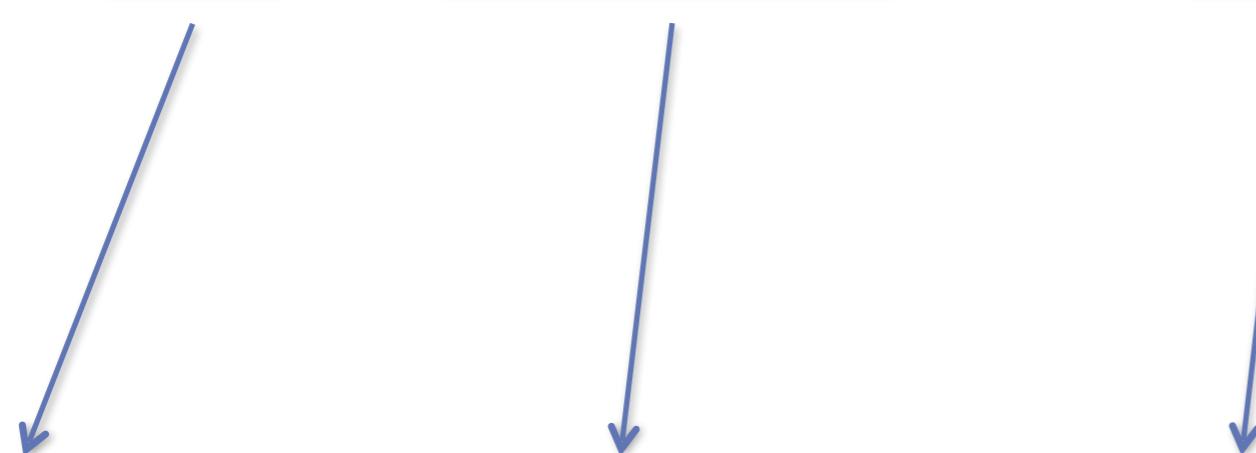
<https://www.desmos.com/calculator/5k43xjrgd8>

Pere Dolcet, 2022

Model de Blinn-Phong

Model de **Blinn-Phong**:

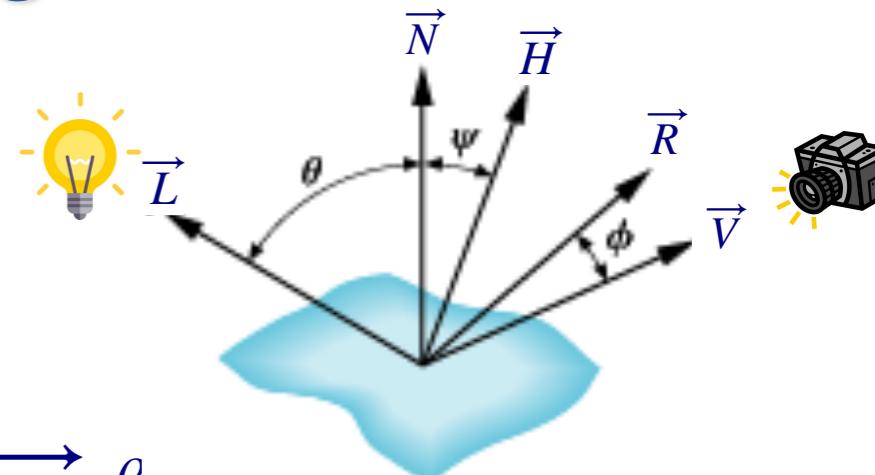
$$I = \underbrace{k_a I_a}_{\text{Ambient}} + \underbrace{k_d I_d \cos(\vec{L}, \vec{N})}_{\text{Diffuse}} + \underbrace{k_s I_s \cos(\vec{N}, \vec{H})^\beta}_{\text{Specular}}$$



<https://www.desmos.com/calculator/sct0asktyy>
Pere Dolcet, 2022

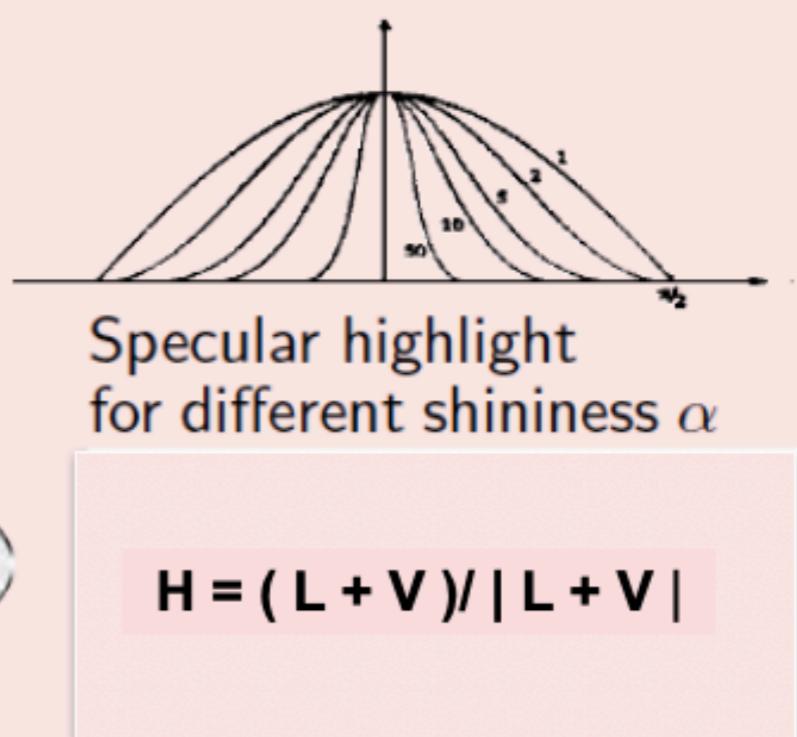
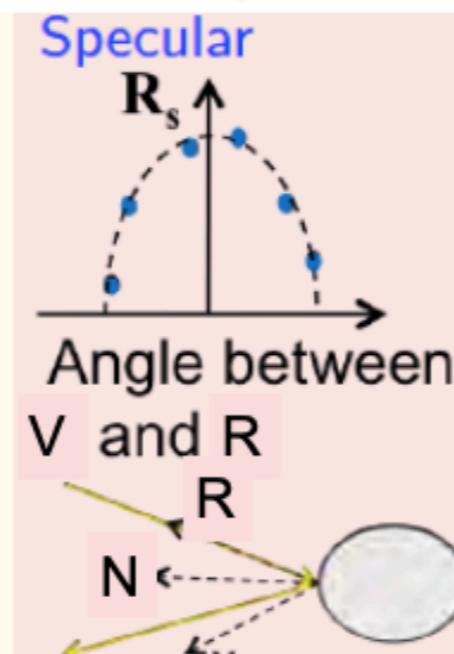
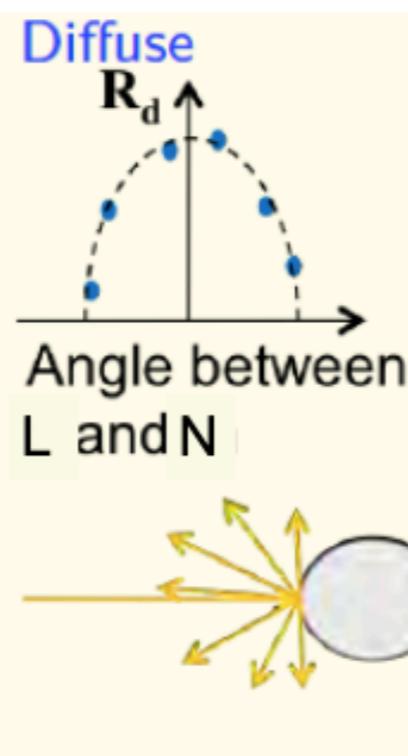
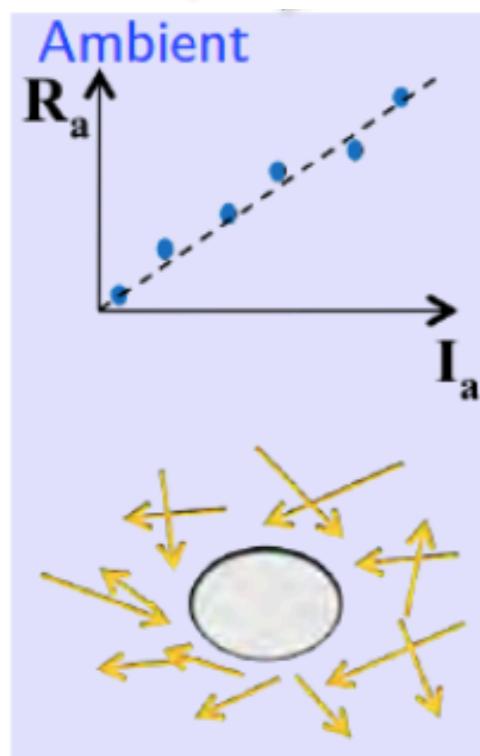
$$\vec{H} = (\vec{L} + \vec{V}) / |\vec{L} + \vec{V}|$$

Model de Blinn-Phong



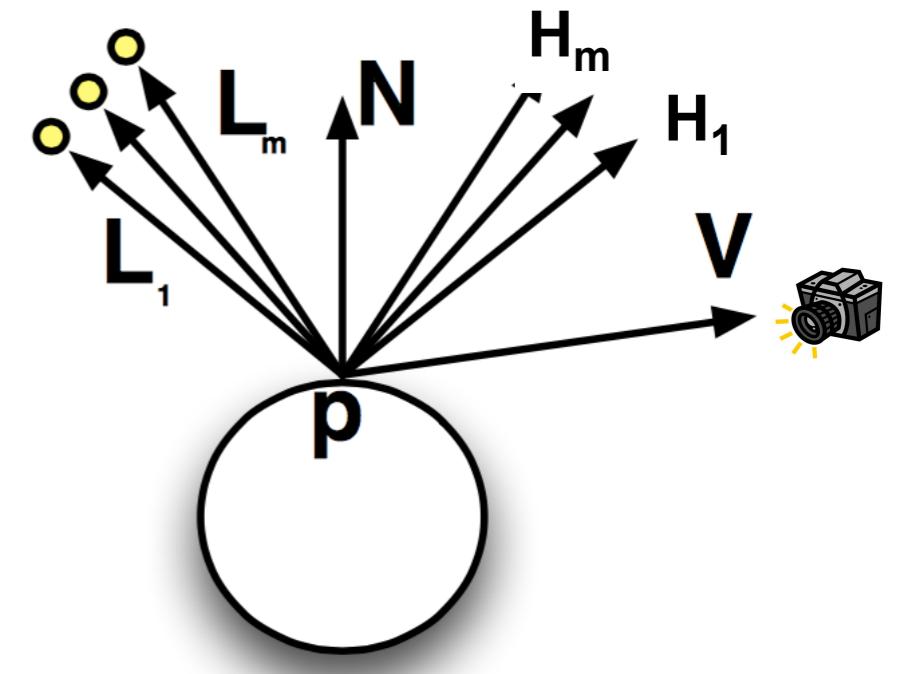
Model de **Blinn-Phong**:

$$I = \underbrace{k_a I_a}_{\text{Ambient}} + \underbrace{k_d I_d \cos(\vec{L}, \vec{N})}_{\text{Diffuse}} + \underbrace{k_s I_s \cos(\vec{N}, \vec{H})^\beta}_{\text{Specular}}$$



Model de Blinn-Phong

La intensitat total en un punt es calcula com la contribució de **totes** les llums de l'escena en el punt més la intensitat ambient global de l'escena.



$$I_{total} = I_{a_{global}}K_a + \sum_{i=1}^{numLlums} \frac{1.0}{a_i + b_id_i + c_id_i^2} (I_{d_i}K_d \max(L_i \cdot N, 0.0) + I_{s_i}K_s \max((N \cdot H_i), 0.0)^\beta) + I_{a_i}K_a$$

On,

L és el vector normalitzat entre el punt i la llum,

N és la normal al punt,

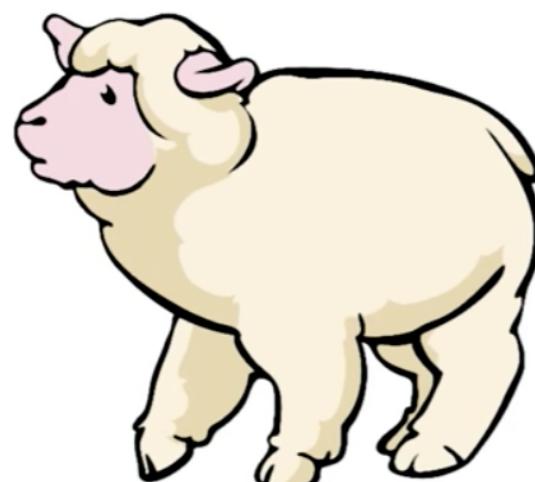
V és el vector normalitzat entre el punt i l'observador

H es calculen a partir dels vectors anteriors

di és la distància del punt a la llum

Cel o Toon Shading

- Tècnica no-fotorealista
- Usa **pocs colors** (tons) en funció de l'angle entre la **llum** i la **normal** de l'objecte
- Els tons s'escullen en funció del *cosinus* de l'angle entre la *llum* i la *normal* a la superfície.
- Si la **normal** és propera a la **direcció de la llum**, s'escull un to més clar. Si és llunyana, s'escull un to més fosc.



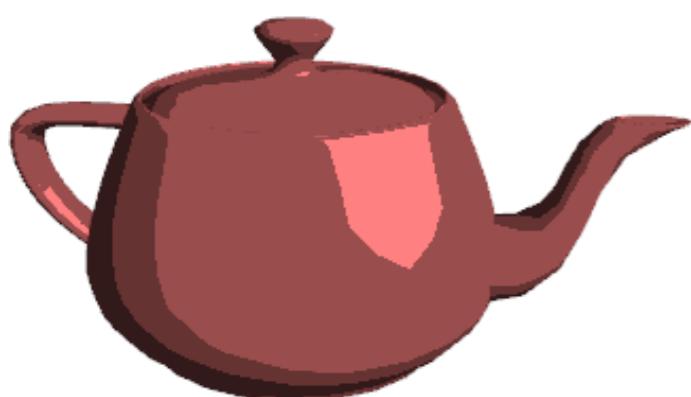
<http://www.floored.com/blog/2014sketch-rendering/>

Cel o Toon Shading

Toon shading:

```
intensity = dot(lightDir, vnormal);

if (intensity > 0.95)
    color = vec4(1.0, 0.5, 0.5, 1.0);
else if (intensity > 0.5)
    color = vec4(0.6, 0.3, 0.3, 1.0);
else if (intensity > 0.25)
    color = vec4(0.4, 0.2, 0.2, 1.0);
else
    color = vec4(0.2, 0.1, 0.1, 1.0);
```



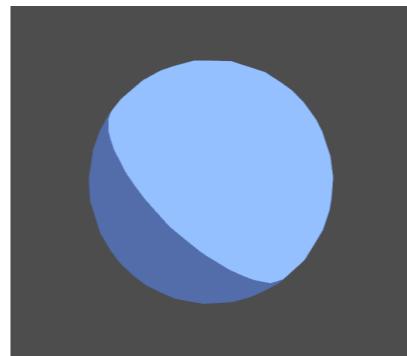
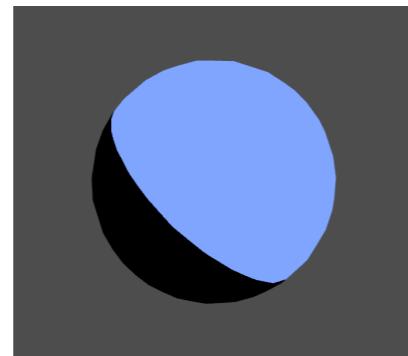
- Com modificaràs el material per tenir una gradació de tons?



Cel o Toon Shading

Toon shading extensions:

- Es pot afectar el color calcular amb la intensitat AMBIENT de la llum sumant-la al color obtingut.



$$+k_a I_a$$

- Es pot calcular la component especular com Blinn-Phong on k_{d_0} és color més clar del rang de colors
- Es poden calcular siluetes (*Rim shading*) en funció de la normal i el vector de visió, sumant la component

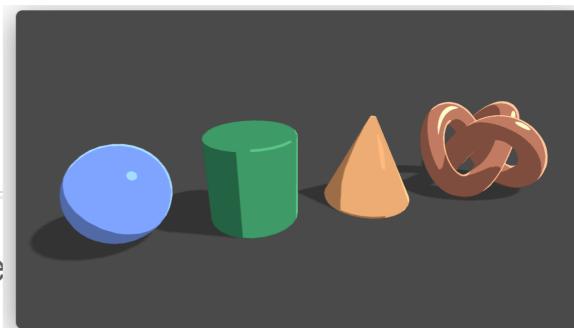
$$\text{rimDot} = (1 - \cos(\vec{V}, \vec{N})) > 0,75 ? 1 : 0$$

I es pot graduar només amb la direcció de la llum

$$\text{rimDot} * \cos(\vec{N}, \vec{L})$$

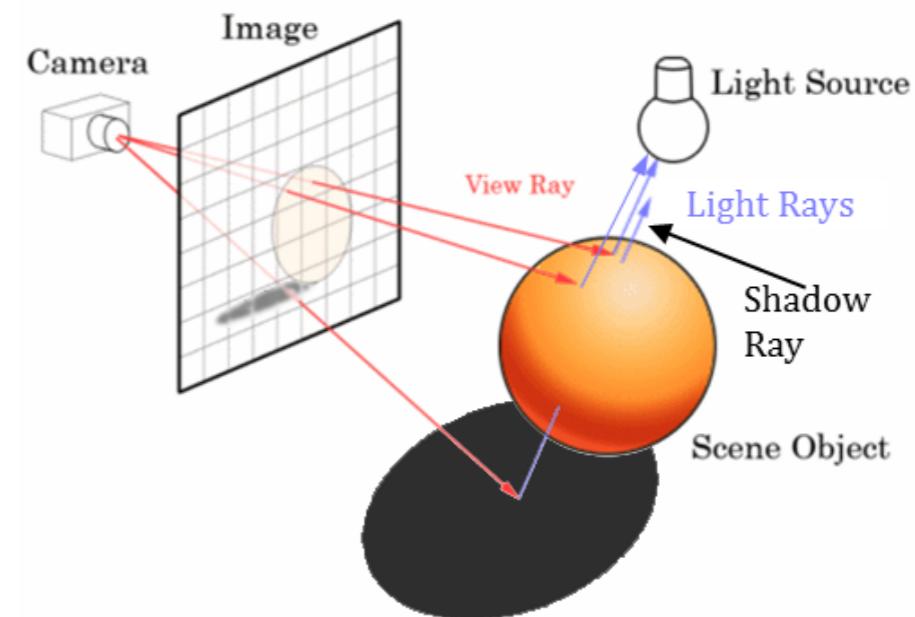
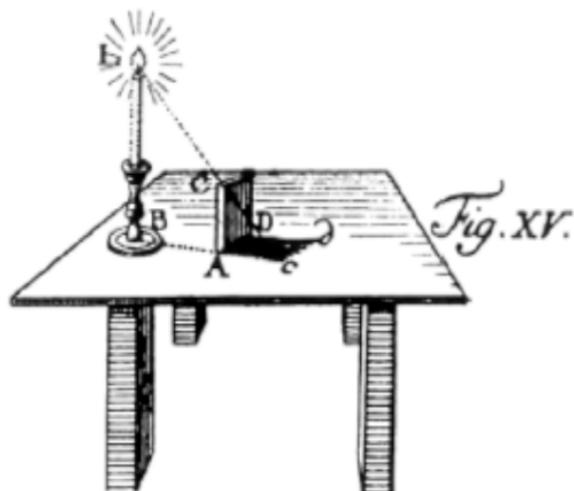
$$+k_{d_0} I_s \cos(\vec{N}, \vec{H})^\beta$$

$$+k_{d_0} k_s \text{ rimDot}$$



4. Ombres

- **Rajos d'ombres:** rajos des del punt d'intersecció a les llums



- Com s'integra en el RayTracing?

$$I_{total} = I_{a_{global}} K_a + \sum_{i=1}^{numLlums} \frac{1.0}{a_i + b_i d_i + c_i d_i^2} (I_{d_i} K_d \max(L_i \cdot N, 0.0) + I_{s_i} K_s \max((N \cdot H_i), 0.0)^\beta) + I_{a_i} K_a$$

4. Ombres

blinnPhongShadow (escena, llums, point, normal, material)

retorna color

// retorna la il·luminació en el punt tenint en compte ombres

c = lambientGlobal * ka

per a cada llum

rL = càcul raig de llum entre **point** i la **llum**

factorOmbra = calculOmbra(rL); // A la classe ShadingStrategy.cpp

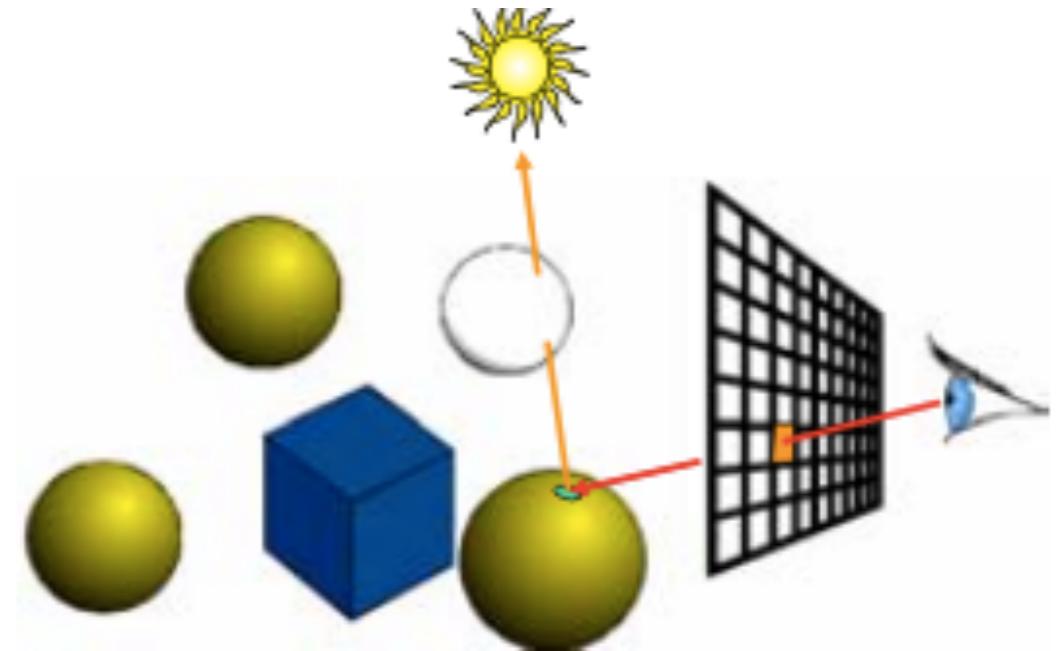
c += ambient + factorOmbra * (difús + espectral);

fper

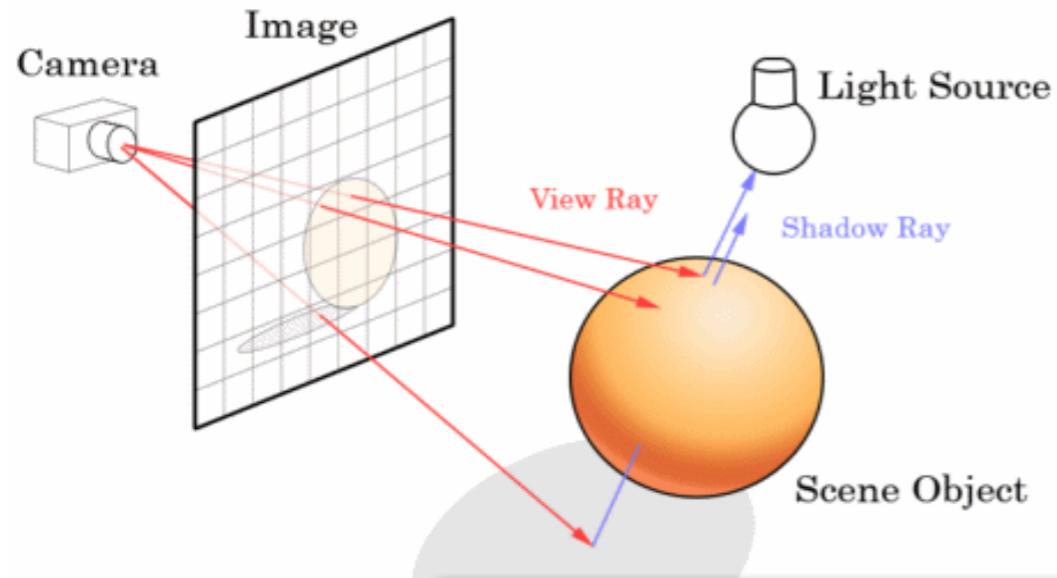
retorna c

factorOmbra és 0.0 si la llum està totalment
ocluïda per un objecte opac

factorOmbra és 1.0 si no hi ha cap objecte
entre la llum i el punt p



4. Ombres: shadow rays



```
class ShadingStrategy {
public:
    // TODO: Fase 2: Canviar el mètode per passar les llums per calcular el shading
    virtual vec3 shading(shared_ptr<Scene> scene, HitInfo& info, vec3 lookFrom, std::vector<shared_ptr<Light>>, vec3 ambientLight) {
        return vec3(0.0, 0.0, 0.0);
    };

    // FASE 2: Calcula si el punt "point" és a l'ombra segons si el flag està activat o no
    float computeShadow(shared_ptr<Light> light, vec3 point, shared_ptr<Scene> scene);

    virtual ~ShadingStrategy() {};
};
```

factorOmbra

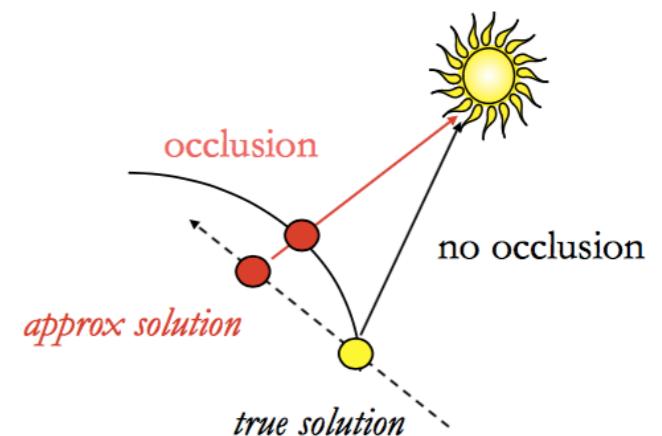
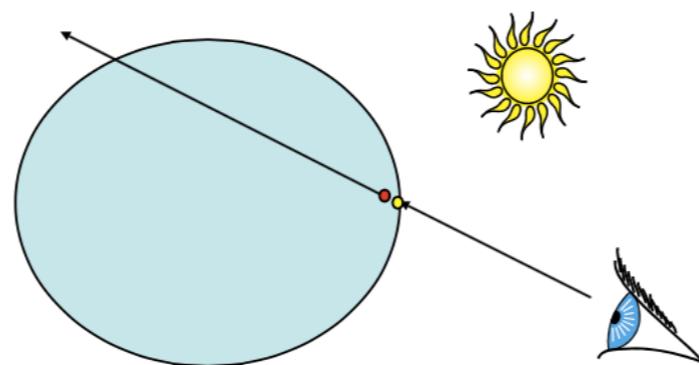
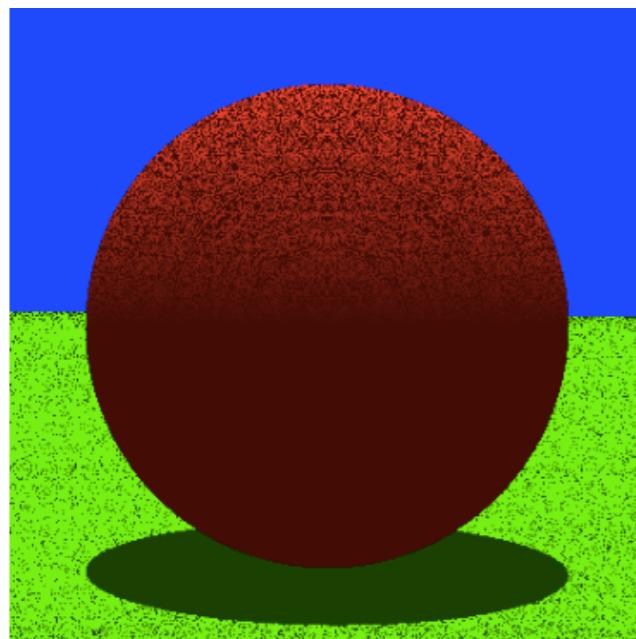
```
void RayTracer::init() {
    auto s = setup->getShadingStrategy();
    auto s_out = ShadingFactory::getInstance().switchShading(s, setup->getShadows());
    if (s_out!=nullptr) setup->setShadingStrategy(s_out);
}
```

```
class ColorShading: public ShadingStrategy
{
public:
    ColorShading() {};
    vec3 shading(shared_ptr<Scene> scene, HitInfo& info, vec3 lookFrom, std::vector<shared_ptr<Light>>, vec3 ambientLight) override;
    ~ColorShading() {};
};
```

```
class ColorShadow : public ShadingStrategy
{
public:
    ColorShadow() {};
    vec3 shading(shared_ptr<Scene> scene, HitInfo& info, vec3 lookFrom, std::vector<shared_ptr<Light>>, vec3 ambientLight) override;
    ~ColorShadow() {};
};
```

4. Problemes amb ombres

- **Problema de les auto-interseccions (self-shadowing):**
Problema de precisió (shadow acne)

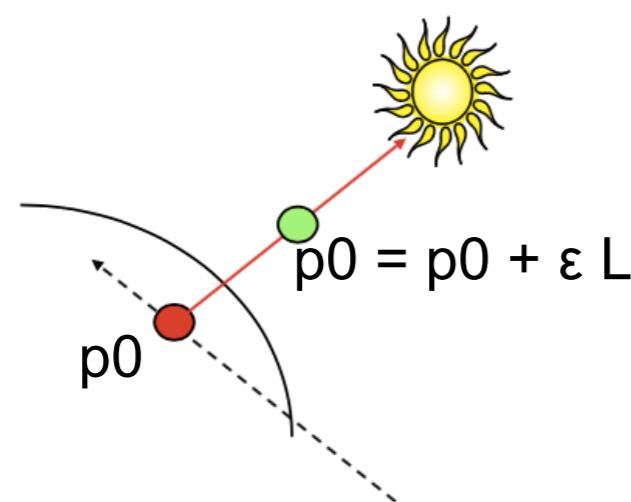


Rraig cap a la llum es defineix com:

$$p = p_0 + \lambda * L$$

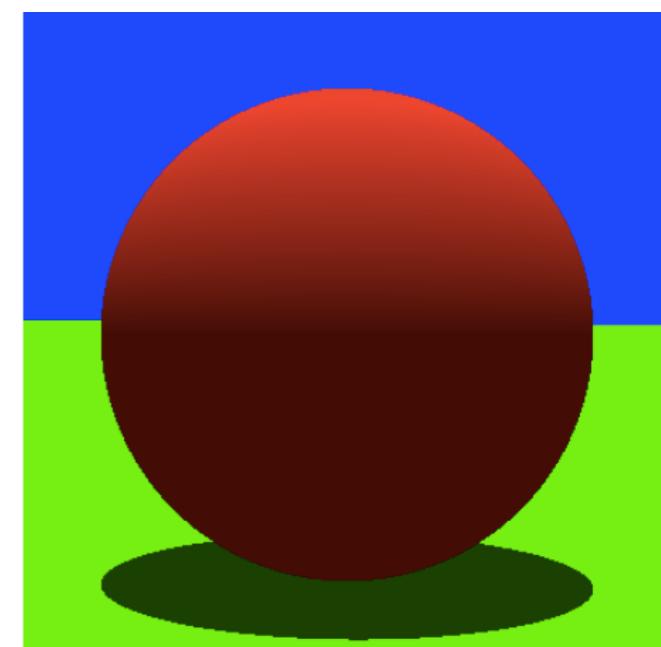
On,

p_0 és el punt d'intersecció,
 L és el vector de p_0 a la
posició de la llum



Es canvia p_0 per:

$$p_0 = p_0 + \epsilon * L, \text{ amb } \epsilon = 0.01$$



5. Kahoot!



**2on kahoot GiVD 2022-23
(Blinn-Phong)**

Crèdits

Aquí trobaràs els mètodes d'interseccar raig amb d'altres objectes

<http://www.realtimerendering.com/intersections.html#IV24>

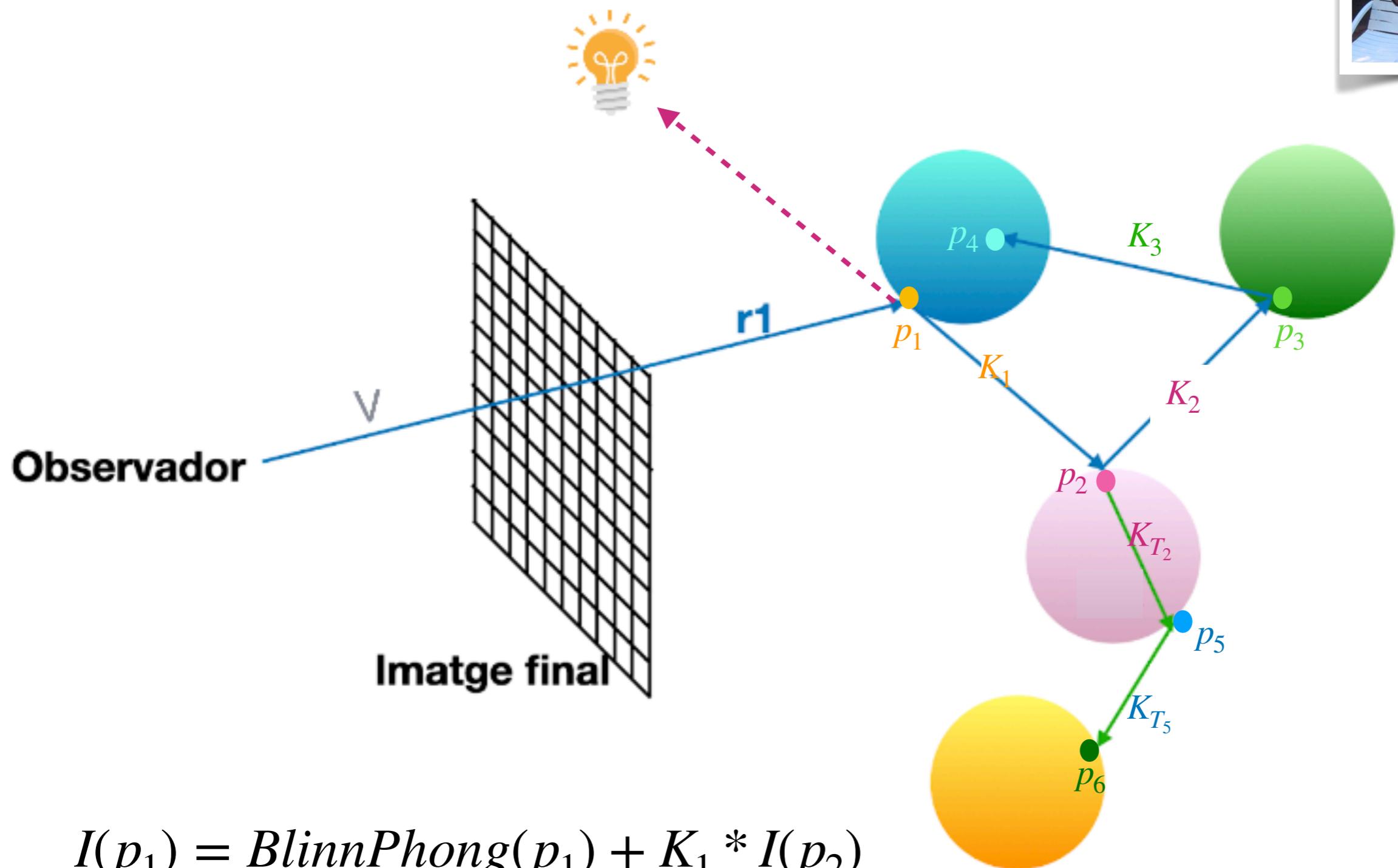
Aquí trobaràs els mètodes d'interseccar raig amb d'altres objectes (codi)

<https://www.iquilezles.org/www/articles/intersectors/intersectors.htm>



The rendering equation: **https://www.youtube.com/watch?v=AODo_RjJoUA**

6. Rajos secundaris



$$I(p_1) = \text{BlinnPhong}(p_1) + K_1 * I(p_2)$$

$$I(p_2) = \text{BlinnPhong}(p_2) + K_2 * I(p_3) + K_{T_2} * I(p_5)$$

6. Rajos secundaris

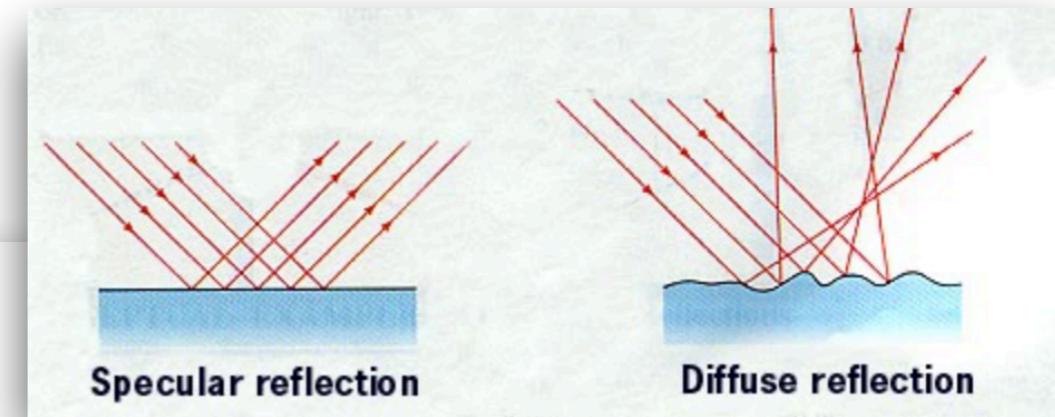
Rajos secundaris:

```
class Material: public Serializable
{
public:
    Material();
    Material(vec3 d);
    Material(vec3 a, vec3 d, vec3 s, float shininess);
    Material(vec3 a, vec3 d, vec3 s, float shininess, float opacity);
    ~Material();

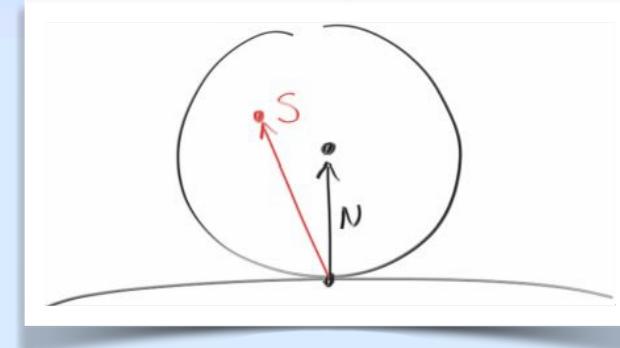
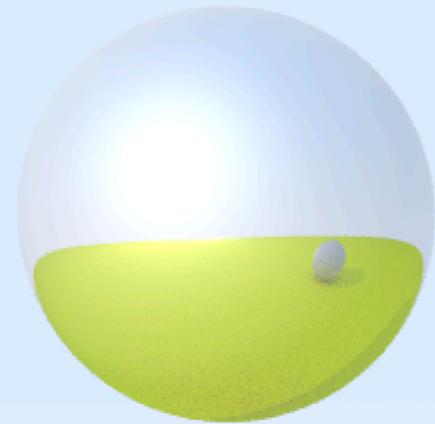
    virtual bool scatter(const Ray& r_in, const HitInfo& rec, vec3& color, Ray & r_out) const = 0;
    virtual vec3 getDiffuse(vec2 point) const;

    vec3 Ka;
    vec3 Kd;
    vec3 Ks;
    vec3 Kt;
    float nu_t;

    float shininess;
    float opacity; // opacity es la fracció de 0..1 (0 és totalment transparent, 1 és totalment opac)
```



6. Rajos secundaris



```
bool Lambertian::scatter(const Ray& r_in, const HitInfo& rec, vec3& color, Ray & r_out) const {  
    vec3 target = rec.p + rec.normal + Hitable::RandomInSphere();  
    r_out = Ray(rec.p, target-rec.p);  
    color = Kd;  
    return true;  
}
```

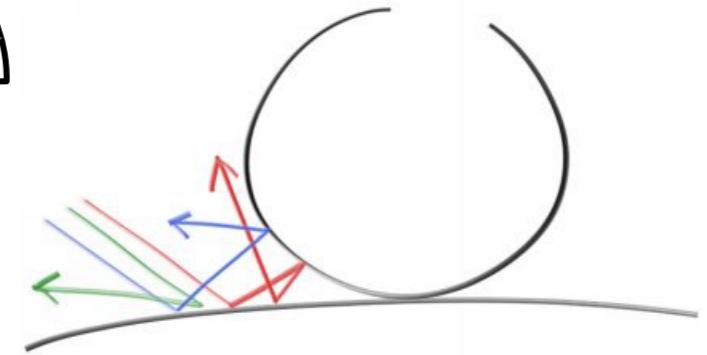
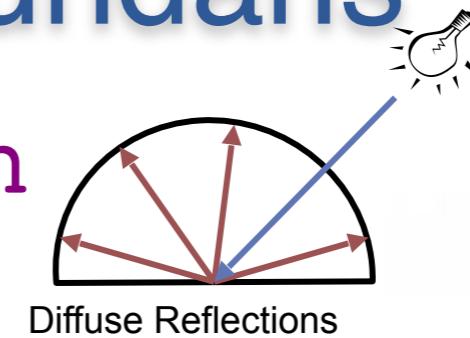
6. Rajos secundaris

Materials difosos: **Lambertian**

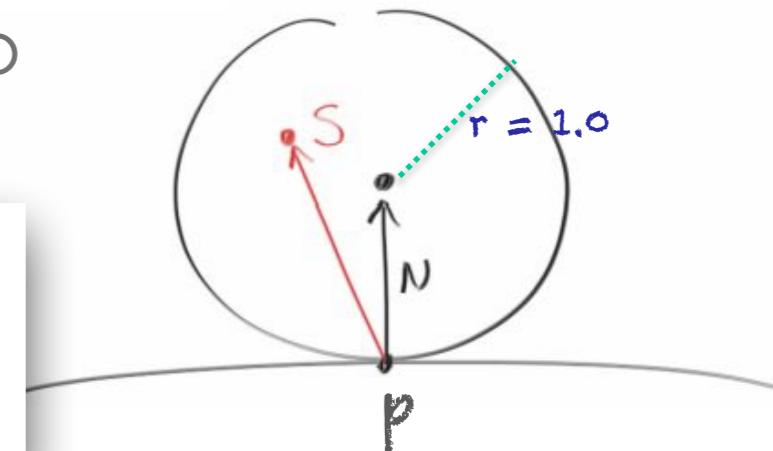
- Com es calcula el raig scattered
 - Raig rebotat en qualsevol direcció:

$p_0 = p$ (punt d'intersecció)

$v = (S - p) = (p + N + \text{randomInSphere}()) - p$



```
bool Lambertian::scatter(const Ray& r_in, const HitInfo& rec, vec3& color, Ray & r_out) const {  
    vec3 target = rec.p + rec.normal + Hitable::RandomInSphere();  
    r_out = Ray(rec.p, target - rec.p);  
    color = Kd;  
    return true;  
}
```



- Com es calcula K ?

- Quan més fosc és el material, més llum és absorbida (per això és més fosc)

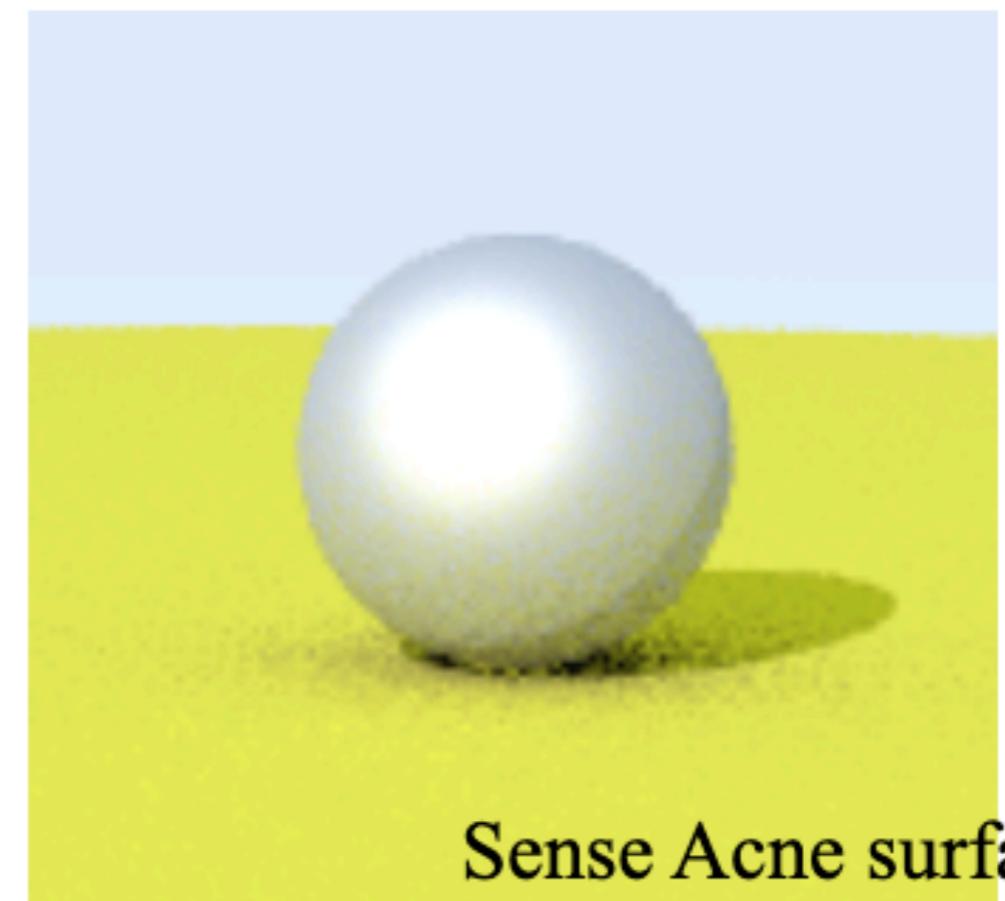
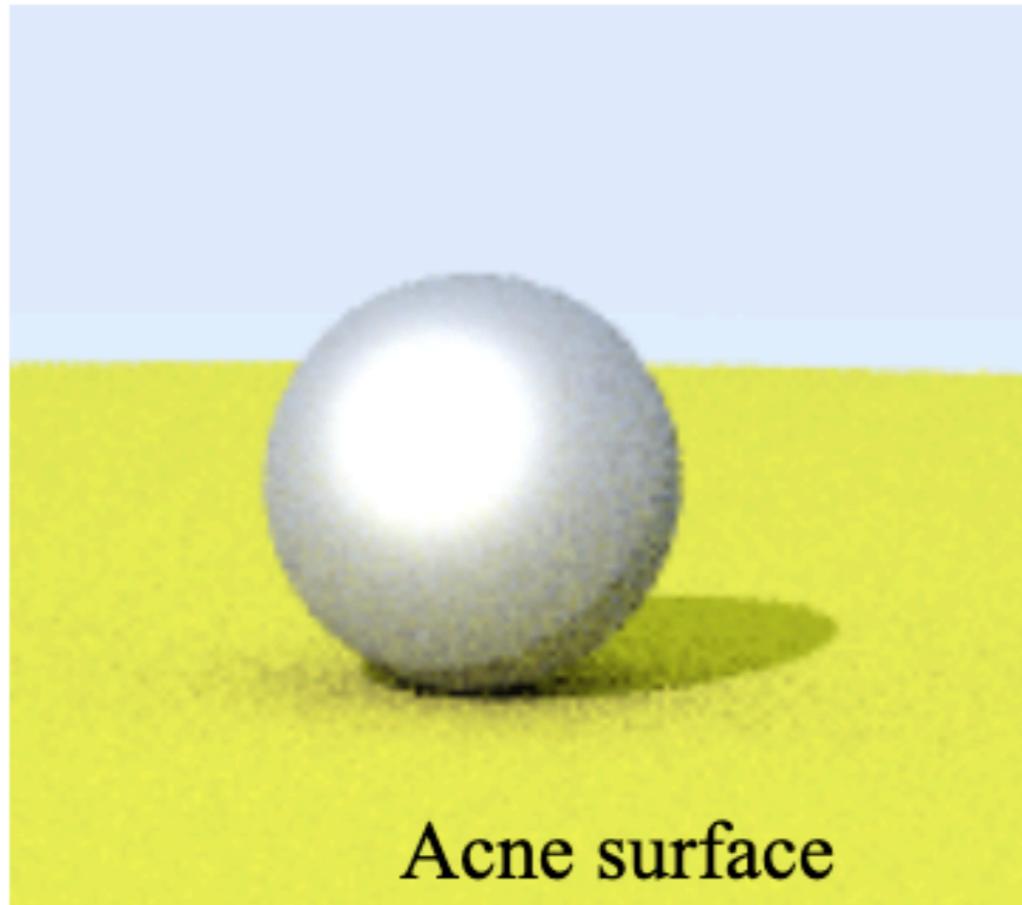
$K = K_d$ (color difós del material)

```
| color = Kd;
```

6. Rajos secundaris

- ## Efectes de surface acne en el raig:

Per construir el raig reflectit cap també tenir en compte els problemes de precisió del càlcul del punt d'intersecció. Cal moure'l en direcció del raig de sortida o scattered.



epsilon aconsellat: 0.01

6. Com integrar un nou material a la pràctica

1. Construir la nova classe que hereti de material i implementar el mètode:

```
virtual bool scatter(const Ray& r_in, const HitInfo& rec, vec3& color, Ray & r_out) const = 0;
```

2. Si calen més atributs al material, cal implementar els mètodes de lectura dels fitxers json, cridant primer a la classe mare.

```
virtual void read (const QJsonObject &json) = 0;  
virtual void write (QJsonObject &json) const = 0;  
virtual void print (int indentation) const = 0;
```

3. Aquests mètodes es cridaran automàticament des de la classe Object

6. Com integrar un nou material a la pràctica

meshExemple.json

3. Aquests mètodes es cridaran automàticament des de la classe Object

```
void Object::read (const QJsonObject &json)
{
    if (json.contains("material") && json["material"].isObject()) {
        QJsonObject auxMat = json["material"].toObject();
        if (auxMat.contains("type") && auxMat["type"].isString()) {
            QString tipus = auxMat["type"].toString().toUpper();
            MaterialFactory::MATERIAL_TYPES t = MaterialFactory::getInstance().getMaterialType(tipus);
            material = MaterialFactory::getInstance().createMaterial(t);
            material->read(auxMat);
        }
    }
}
```

6. Rajos secundaris



Materials especulars (Metal):

- Com es calcula la direcció del raig reflectit?

Recordant la fòrmula de raig reflectit del Tema 2b, cal canviar els signes, donat que el raig incident va en direcció al punt.

$$r_{out} = r_{in} - 2(r_{in} \cdot N) \cdot N$$

glm::reflect(..)

on r és el Raig incident

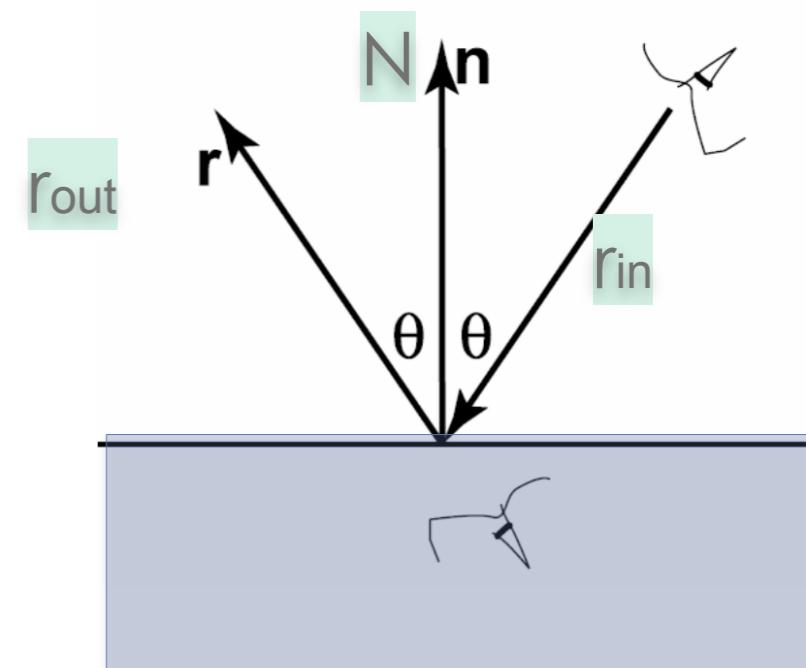
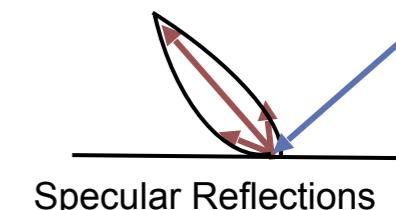
N la normal al punt d'intersecció

r_{out} és el Raig reflectit (scattered)

Rraig reflectit:

$p_0 = p$ (punt d'intersecció)

$v = r_{out}$



- Com es calcula K ? Es a dir, com atenua el color?

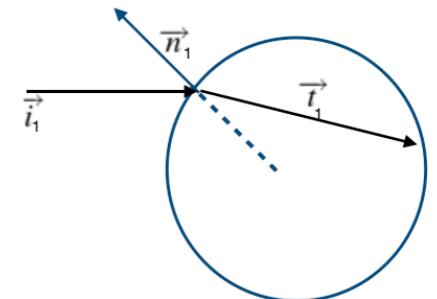
Mitjançant el coeficient especular K_s del material

6. Rajos secundaris

Materials especulars (**Transparent**):

- Com es calcula la direcció del raig transmès?

Recordant la fòrmula de raig transmès segons la llei de Snell



$$\frac{\sin \theta_t}{\sin \theta_i} = \frac{\mu_i}{\mu_t} = \mu_{it}$$

`t = glm::refract(...)`

Sustancia	Índice de refracción (línea sodio D)
Azúcar	1.56
Diamante	2.417
Mica	1.56-1.60
Benceno	1.504
Glicerina	1.47
Agua	1.333
Alcohol etílico	1.362
Aceite de oliva	1.46

suposem tots els objectes situats en el buit



- Com es calcula K? Es a dir, com atenua el color?

Mitjançant el coeficient especular K_t del material(o 1-opacitat del material)



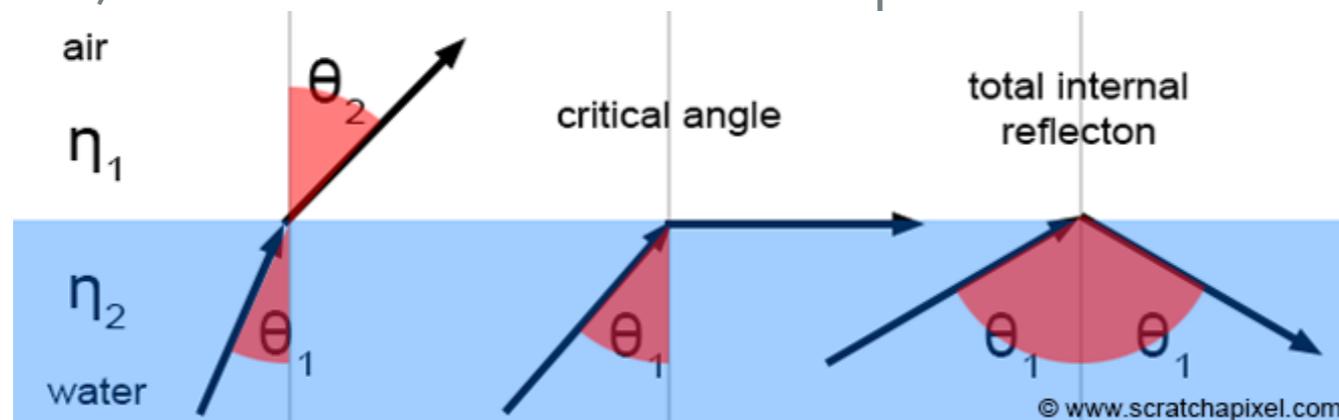
6. Rajos secundaris

Cas crític:

$$\frac{\sin \theta_t}{\sin \theta_i} = \frac{\mu_i}{\mu_t} = \mu_{it}$$

REFLEXIÓ TOTAL INTERNA (angle refrectat surt de la superfície).

Per tant, només es té en compte el vector reflectit.

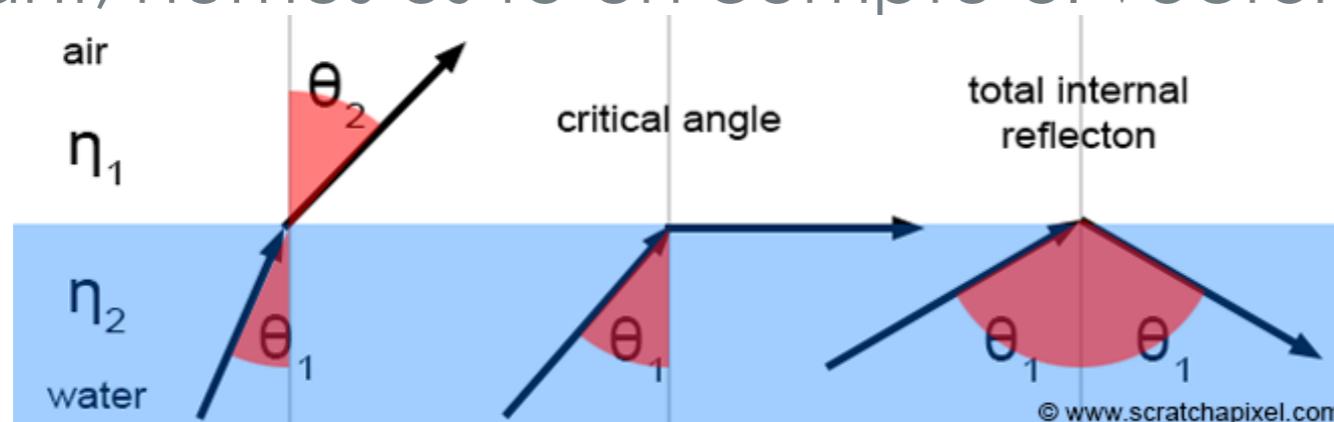


- Com es calcula la direcció del raig transmès?
Com el raig reflectit
- Com es calcula K? Es a dir, com atenua el color?
Mitjançant el coeficient especular K_s del material

6. Rajos secundaris

Cas crític:

REFLEXIÓ TOTAL INTERNA (angle refrectat surt de la superfície).
Per tant, només es té en compte el vector reflectit.



- Com es calcula la direcció del raig transmès?

Com el raig reflectit

`glm::reflect(...)`

`t = glm::refract(...)`

`glm::length(t) < epsilon`

- Com es calcula K? Es a dir, com atenua el color?

Mitjançant el coeficient especular K_s del material