

Sessió Setmana 6

GiVD 2022-23

Guió de la sessió

1. Planificació de la setmana 6
 2. Recap il·luminació local,
reflexions, transparències
 3. Recap a textures
 4. Extensions
 5. Dubtes de la pràctica
-



1. Planificació



Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
20	21	22	23	24	25	26
Tema 2	Pràctica 1					Problemes 1 Set. 6



Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
27	28	29	30	31	01	02
		Examens parciais		Dia del parcial 15:00-18:00. Prova/Entrevista Pràctica 1		Practica 1
03	04	05	06	07	08	09
		Setmana Santa				
10	11	12	13	14	15	16
Problemes 2	Pràctica 2					Set. 7

1. Instruccions per lliurar la pràctica

- Lliurar la pràctica
 - al **campus**: 2 fitxers
 - .zip corresponent al github (Seguir el patró de README del projecte)
 - pdf de la memòria de la pràctica (és el README)
 - al **github** marcar la darrera versió “Pràctica 1 final”

Les vostres aportacions

- Cada grup ha de pujar com a mínim una imatge per cada equip a la web
- Animeu-vos a pujar-ne més!

Padlet 22-23

anna3143 • 17d

Terrific renderings 2022-23

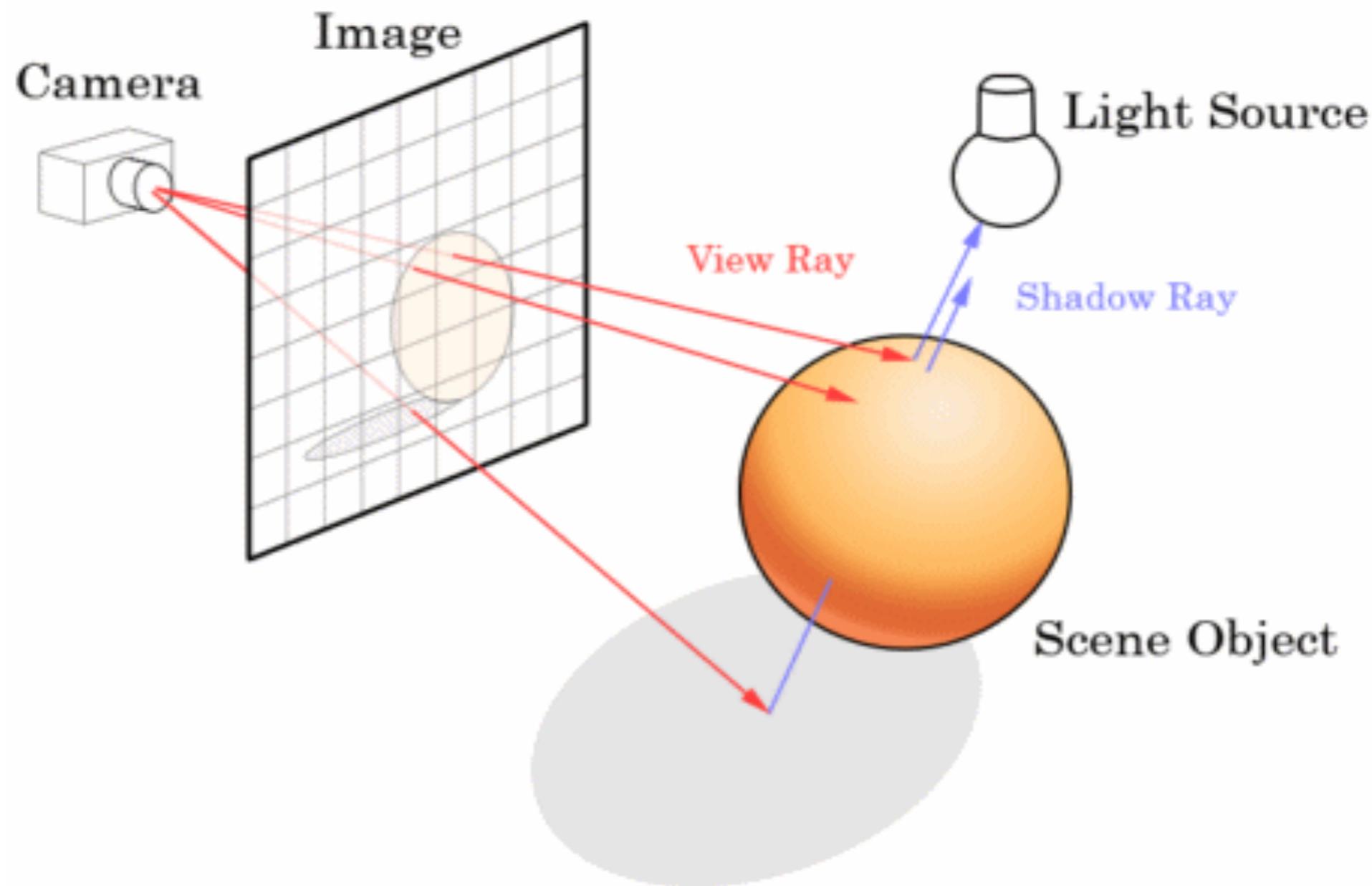
GiVD teams

1. Instruccions per la prova

- Prova: 31.03.2023 15h (Aules IB, IC, ID i IF)
 - Es presencial, individual i obligatòria per presentar la pràctica 1
 - prova de modificació del codi de la pràctica (uns 50 min)
 - valuació d'un pràctica d'un altre equip (uns 50 min)
 - valuació personal i del vostre equip (uns 5 min)



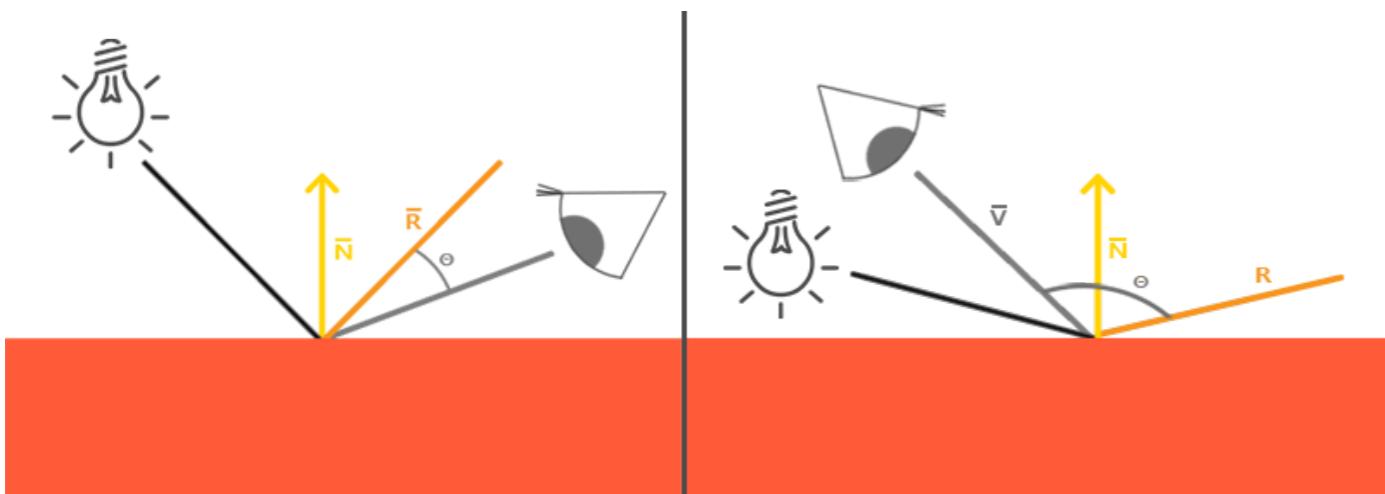
2. Recap il·luminació local: Blinn-Phong



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

2. Recap il·luminació local: Blinn-Phong

Model de Phong



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

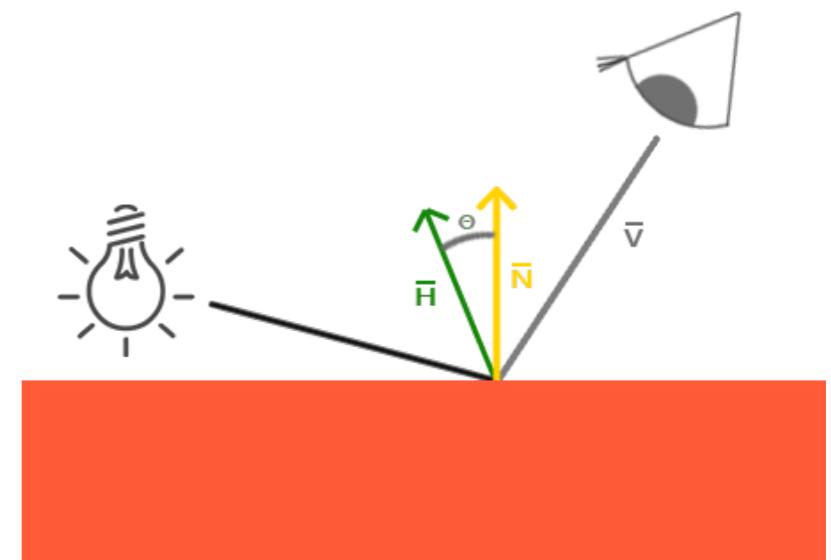
$$I \sim k_s * I_s * \cos^\alpha \phi$$

$\cos \phi = \vec{R} \cdot \vec{V}$ (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)

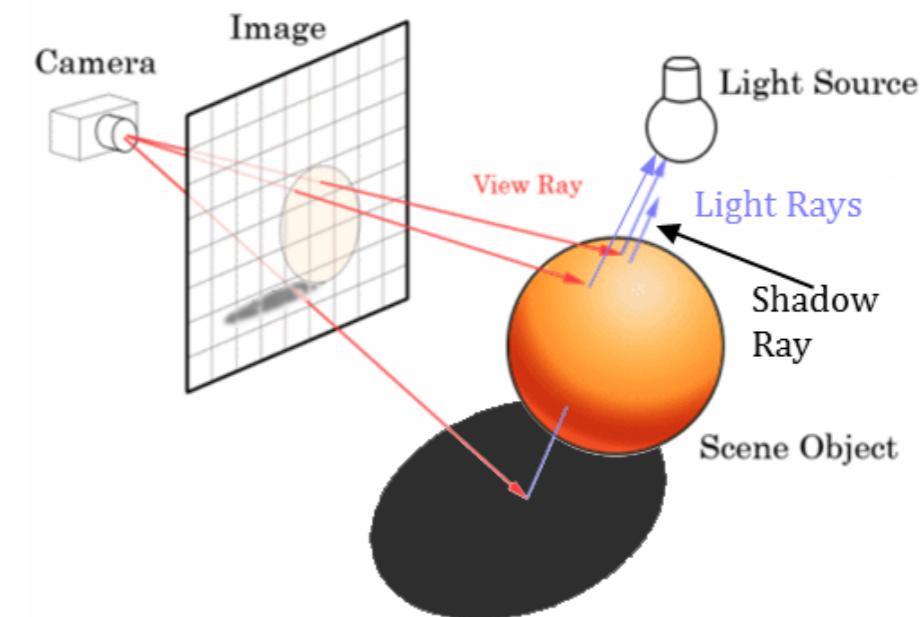
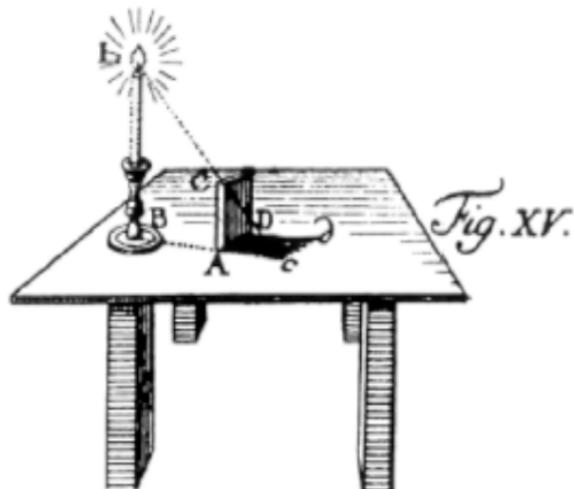
<https://www.desmos.com/calculator/sct0asktyy>

Pere Dolcet, 2022

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

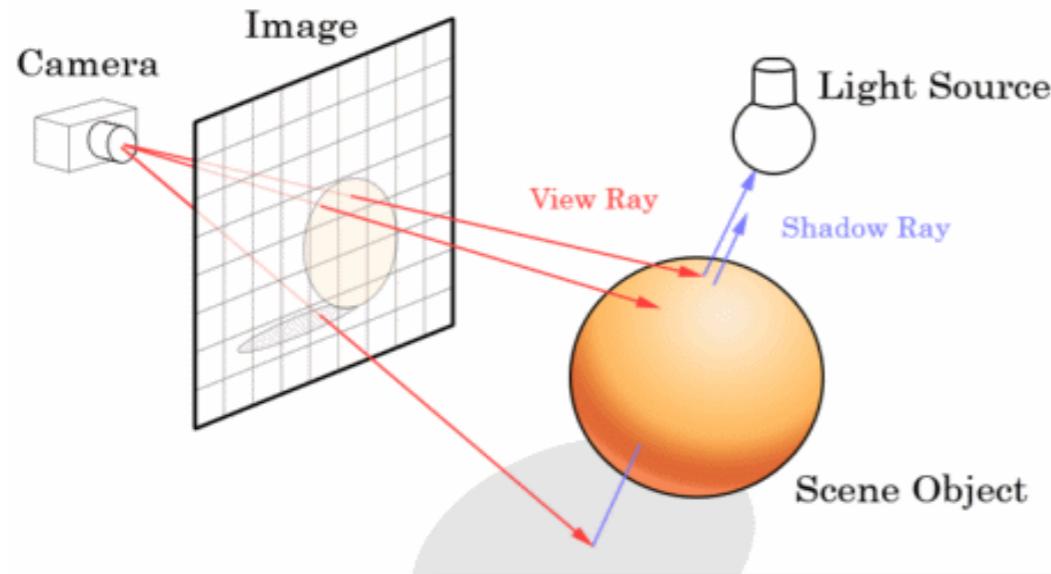
2. Recap il·luminació local:ombres

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



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

2. Recap il·luminació local:ombres



```
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() {};
};
```

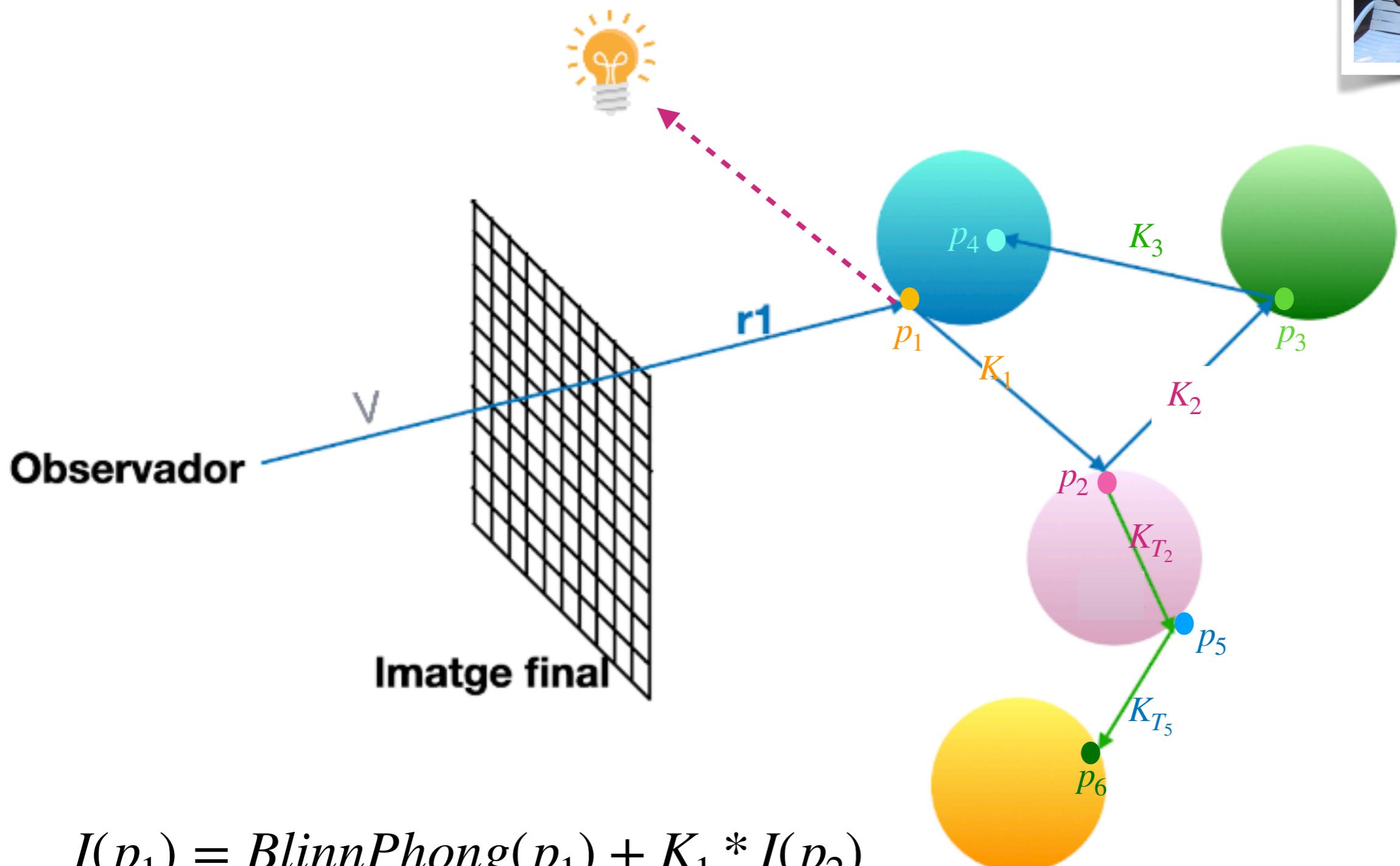
factorOmbra

```
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() {};
};
```

2. Recap: Rajos secundaris



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

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

2. Recap: Rajos secundaris

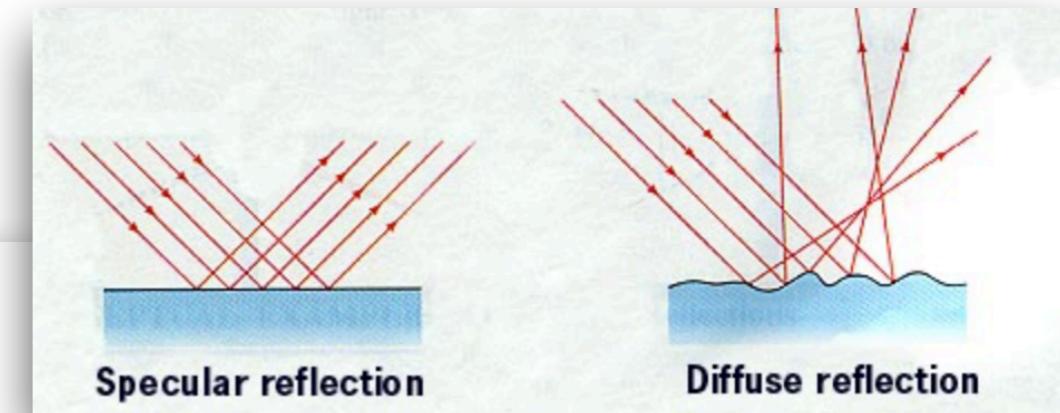
Rajos secundaris:

```
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();

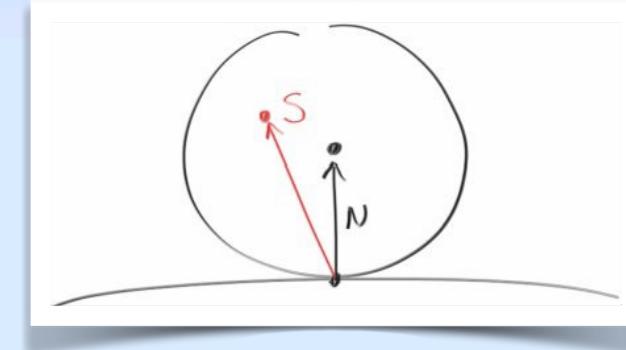
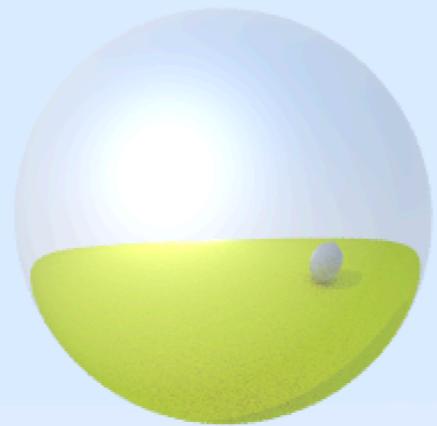
    virtual bool scatter(const Ray& r_in, const HitInfo& rec, vec3& color, Ray & r_out) const = 0;
    virtual vec3 getDiffuse(vec2 point) const;

    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)
```



2. Recap: 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;  
}
```

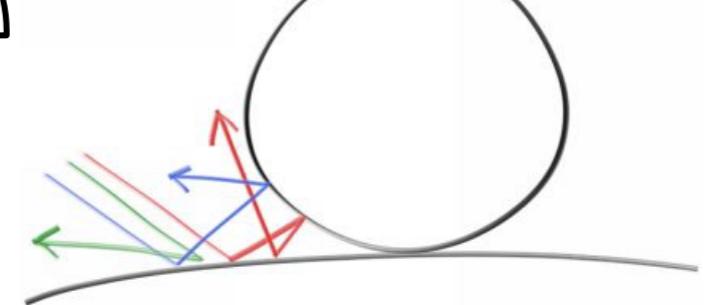
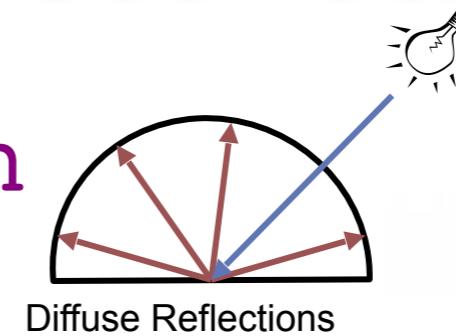
2. Recap: Rajos secundaris

Materials difosos: **Lambertian**

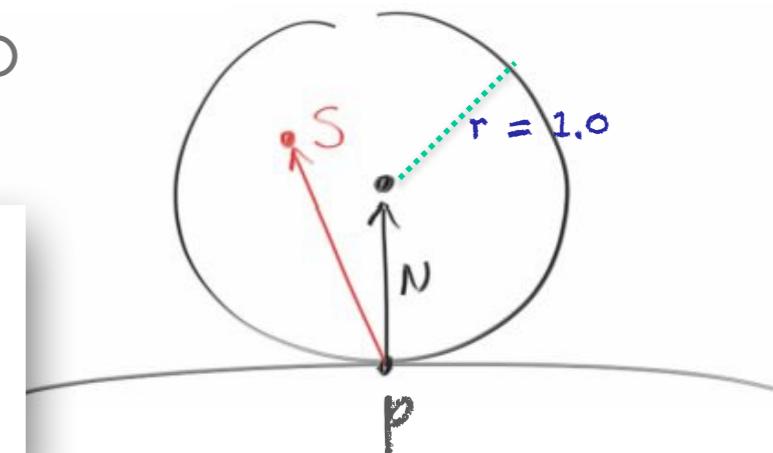
- Com es calcula el raig scattered
 - Raig rebotat en qualsevol direcció:

$p_0 = p$ (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)

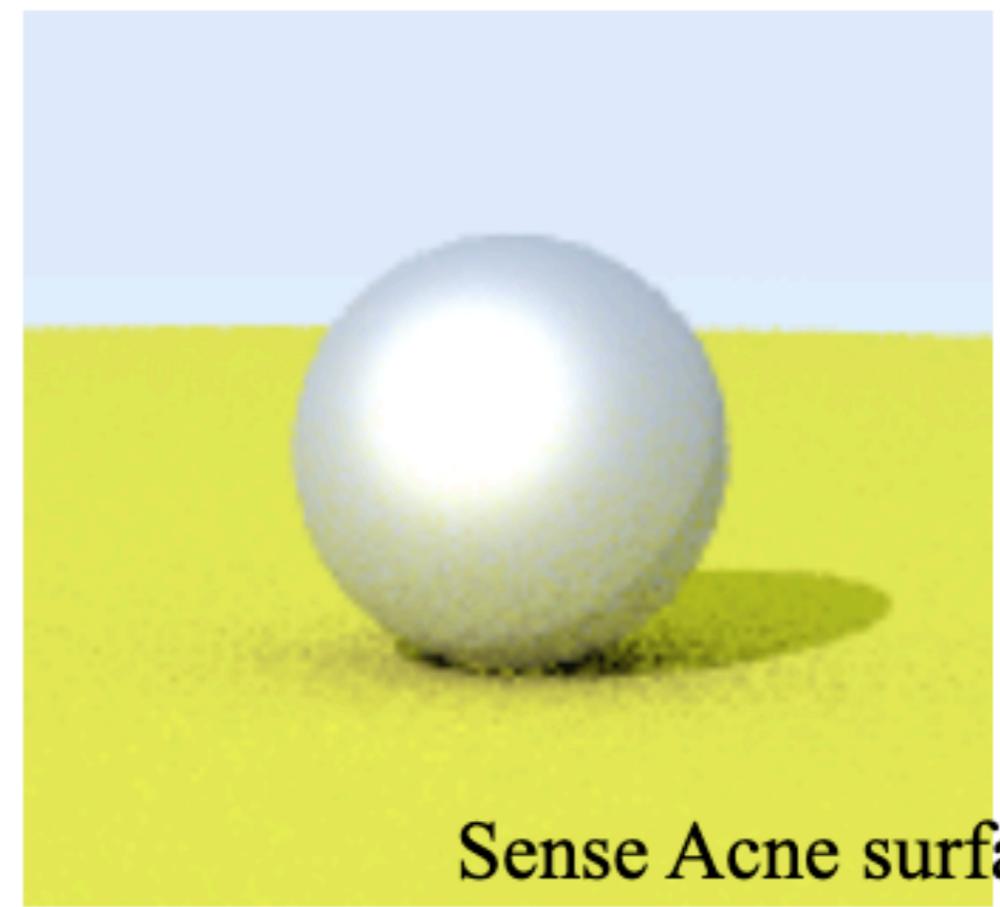
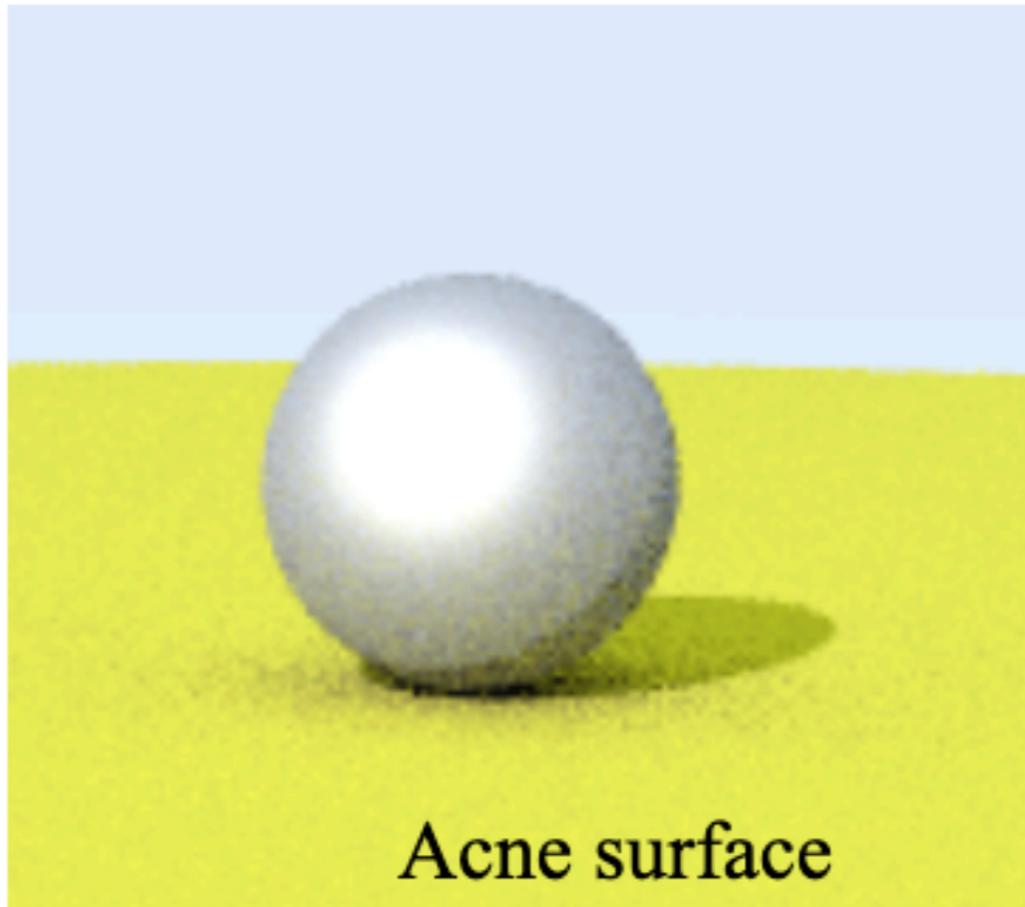
$K = K_d$ (color difós del material)

```
|   color = Kd;
```

2. Recap: 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

2. 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

2. Com integrar un nou material a la pràctica

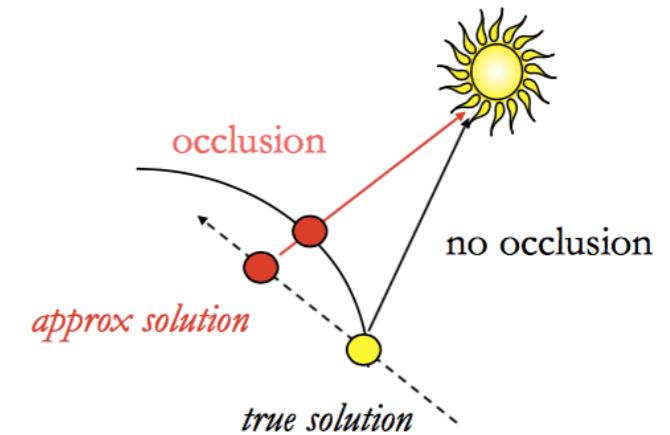
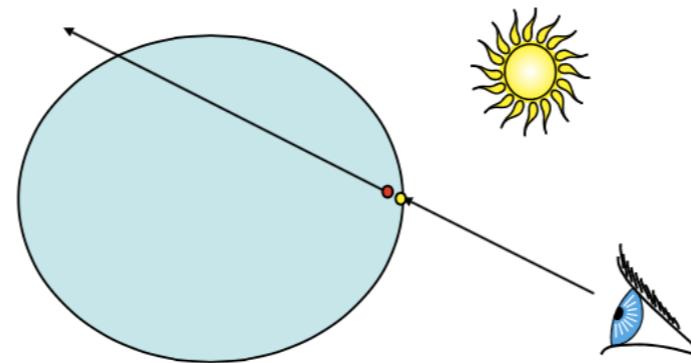
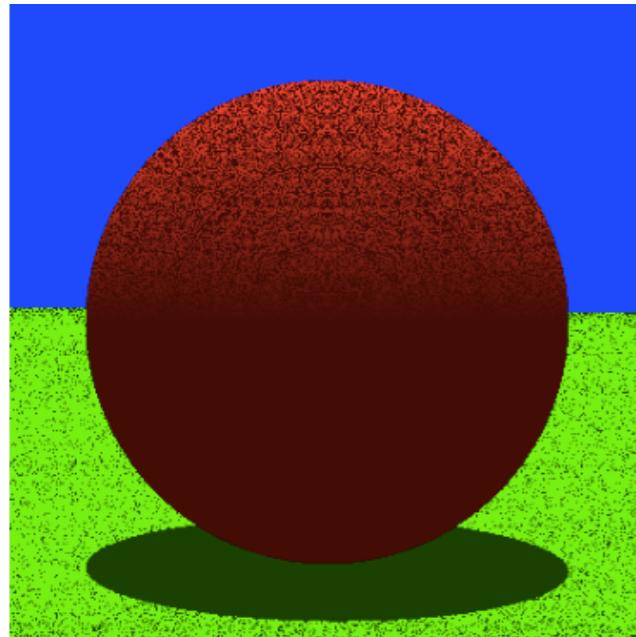
meshExemple.json

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);
        }
    }
}
```

2. Recap : shadow and shading acné

- Problema de les auto-interseccions (acné):

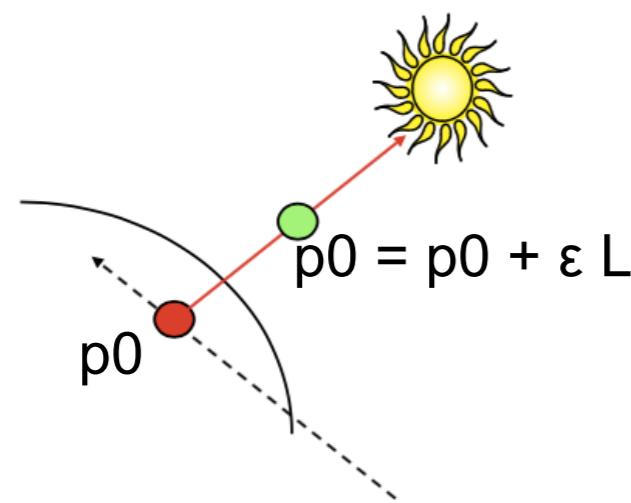


Rraig cap a la llum es defineix com:

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

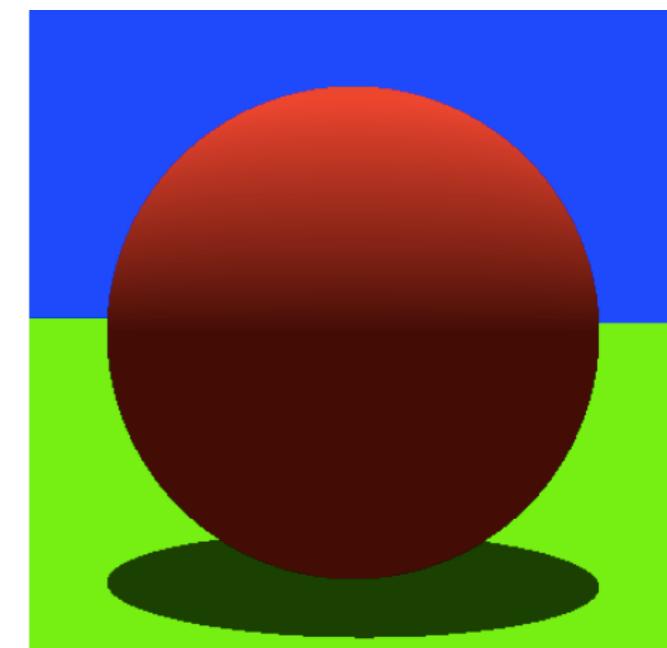
On,

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:

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



2. Recap: 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 d és el Raig incident

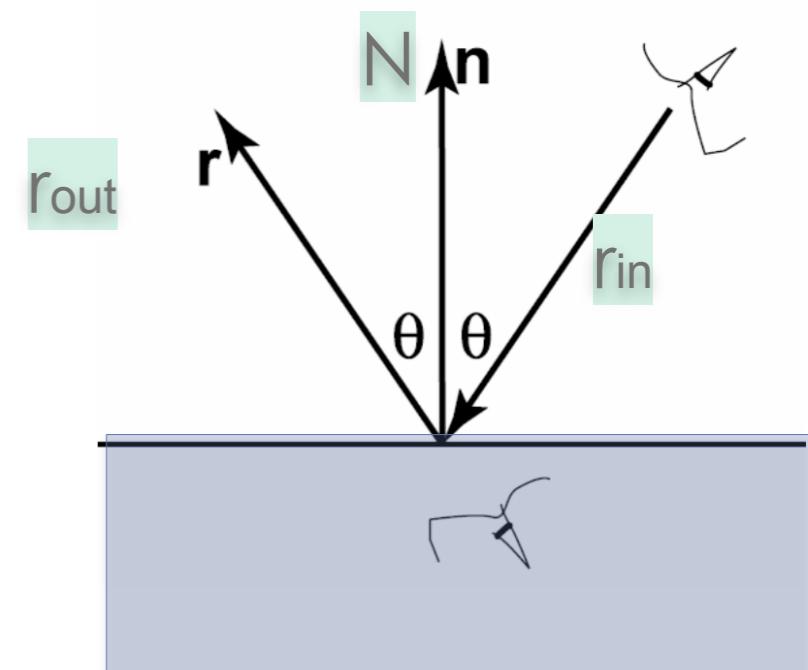
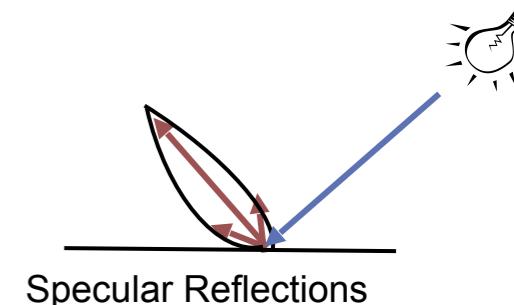
N la normal al punt d'intersecció

rout 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?

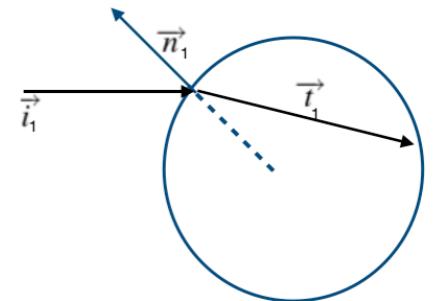
Mitjançant el coeficient especular K_s del material

2. Recap: 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 **Kt** del material(o 1-opacitat del material)



2. Recap: Rajos secundaris

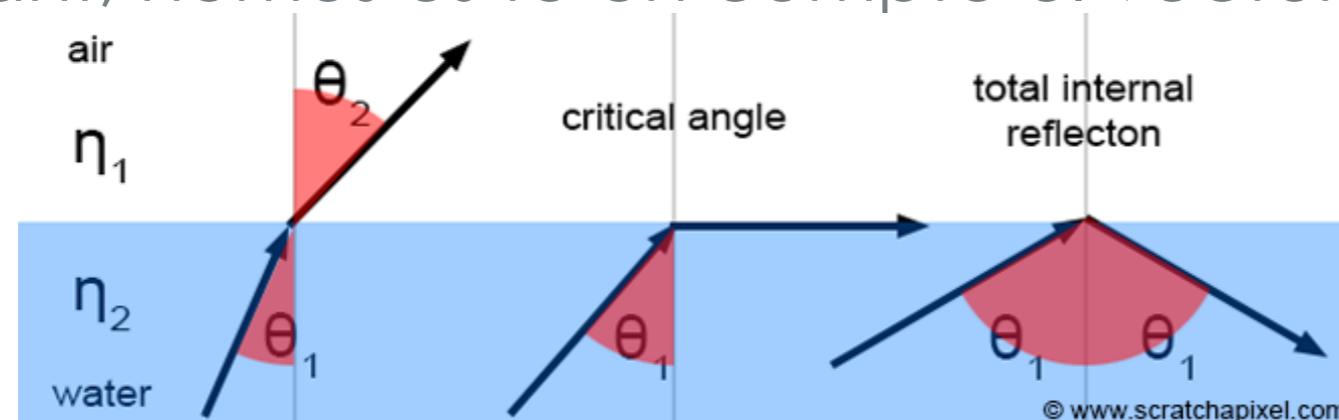
Cas crític:

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

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

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

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

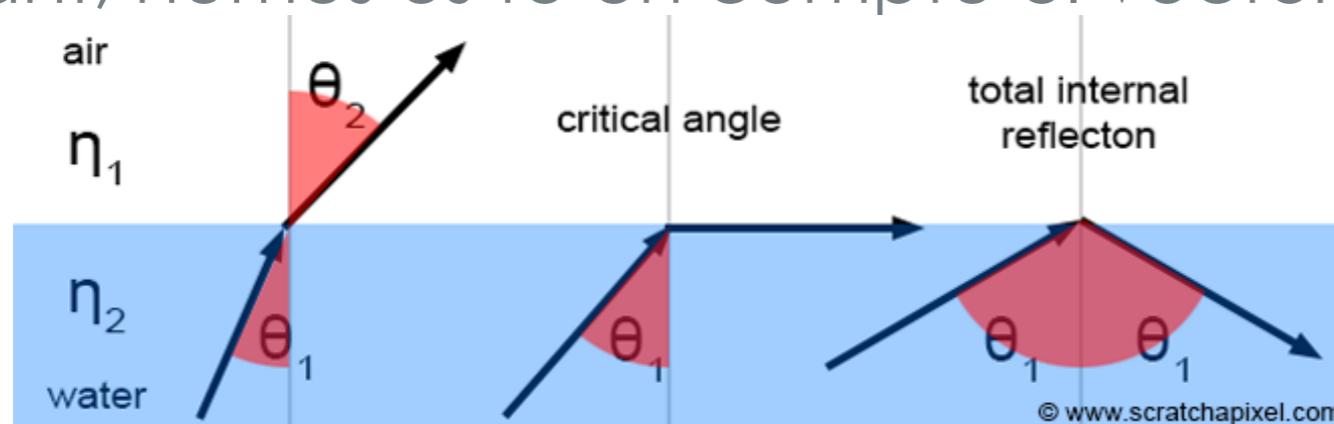


- Com es calcula la direcció del raig transmès?
Com el raig reflectit
- Com es calcula K? Es a dir, com atenua el color?
Mitjançant el coeficient especular Ks del material

2. Recap: 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?

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

Com el raig reflectit

```
glm::reflect(...)
```

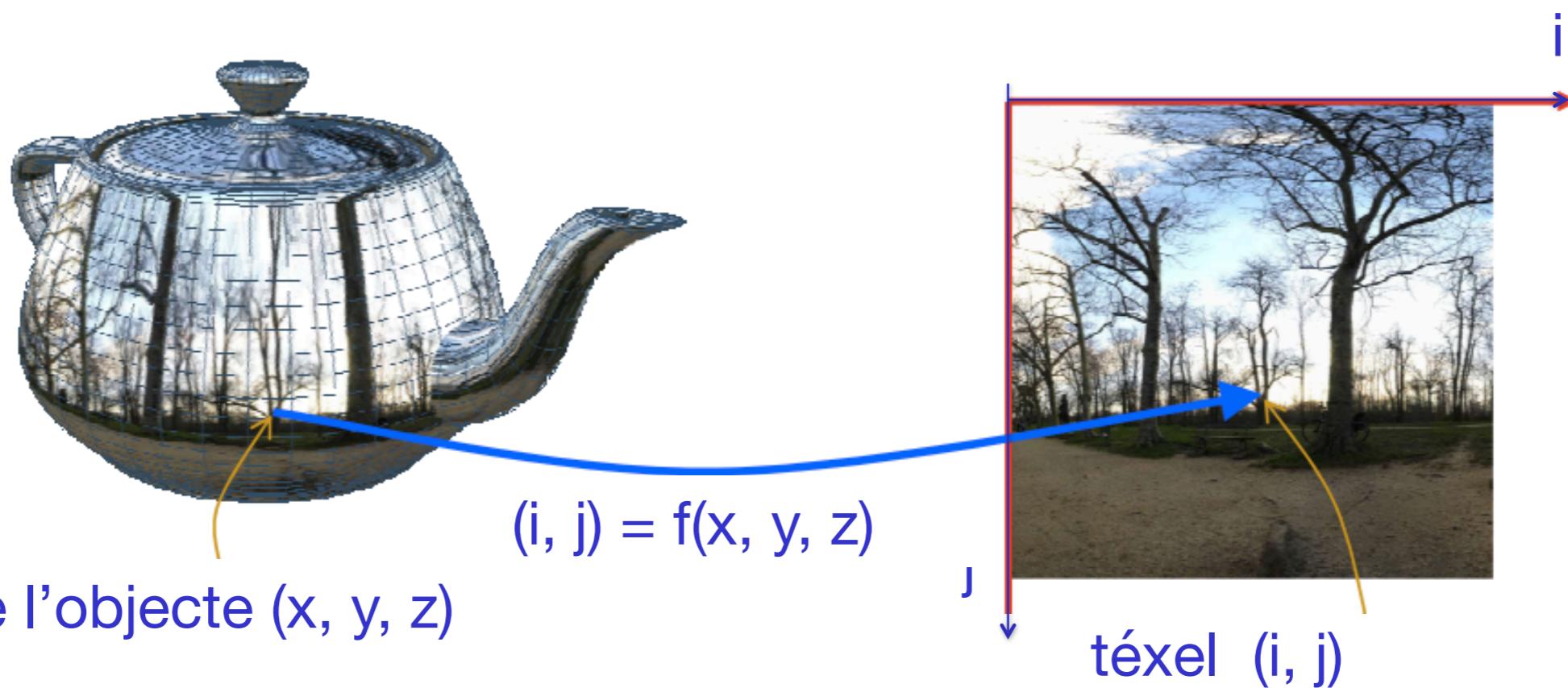
glm::length(t) < epsilon

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

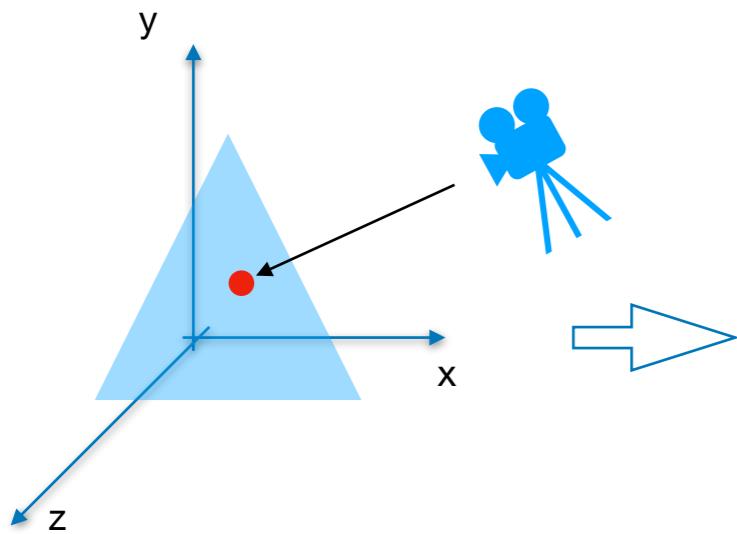
Mitjançant el coeficient especular K_s del material

3. Recap raytracing i textures

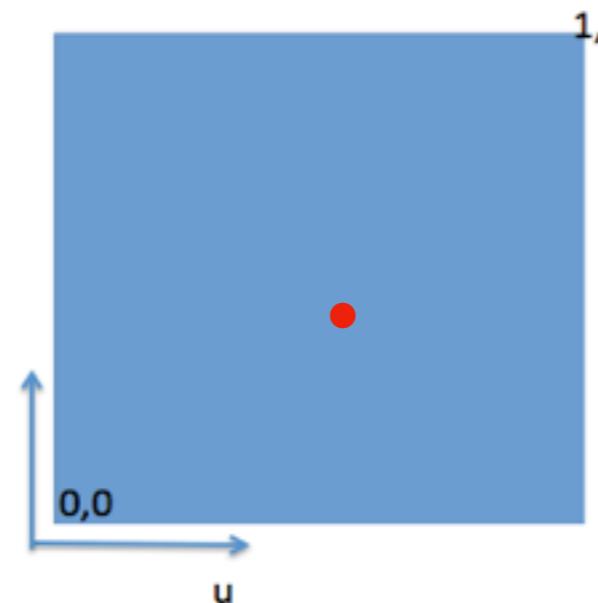
- 'Enganxar' o **mapear** (fer mapping) una textura sobre un objecte consisteix en definir una funció unívoca (**funció de mapping**) que a cada punt d'un objecte (x, y, z) li fa corresponder un punt d'una imatge (i, j).
- El mapping pot ser **directa** o en dues fases



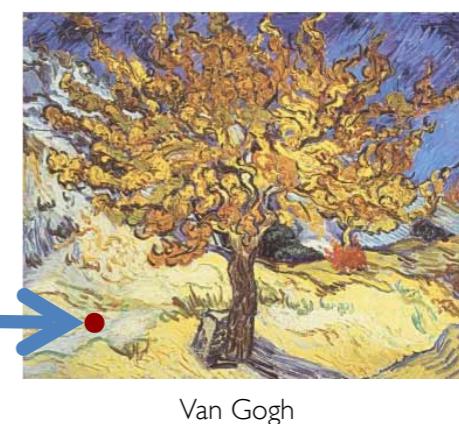
3. Recap raytracing i textures



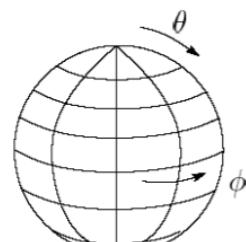
Funció de projecció



Funció de correspondència

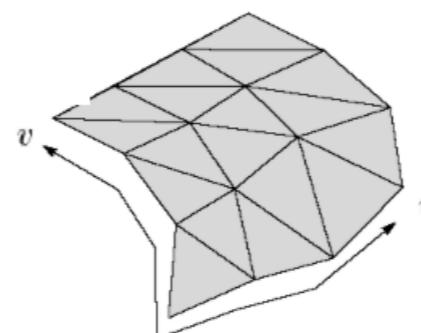


$$\begin{array}{l} u = x/w \\ v = y/h \end{array}$$



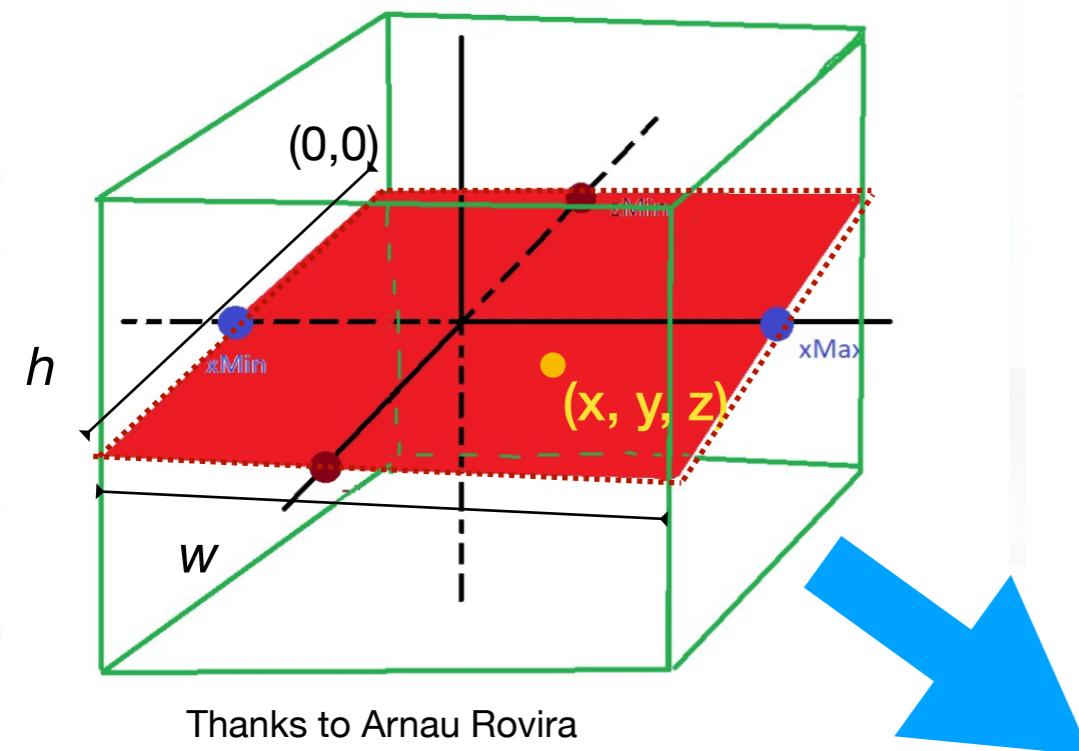
hitInfo

$$\begin{array}{l} u = \phi / 2\pi \\ v = y/h \end{array}$$



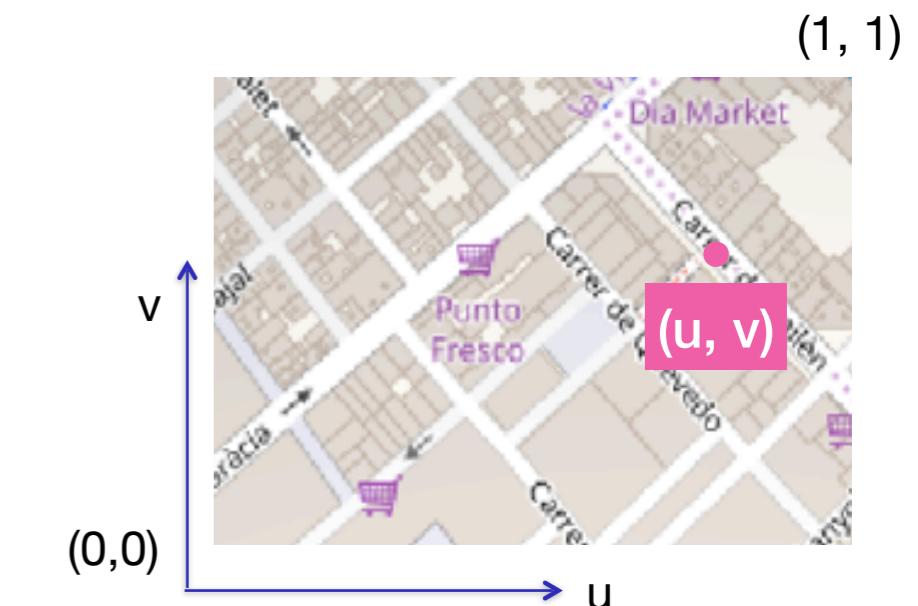
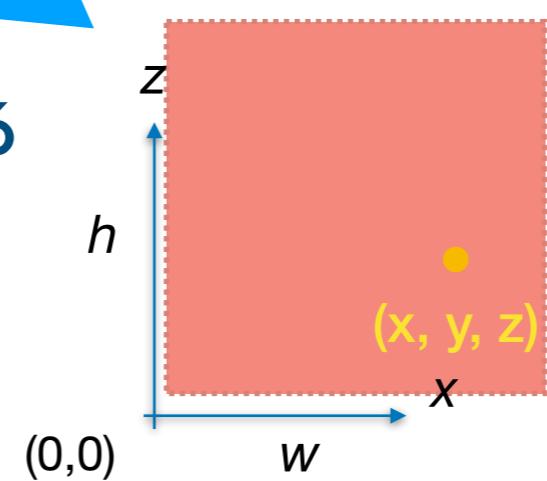
3. Recap raytracing i textures

- **Pla:** $(x, y, z) \rightarrow (u, v)$



Thanks to Arnau Rovira

Funció de projecció

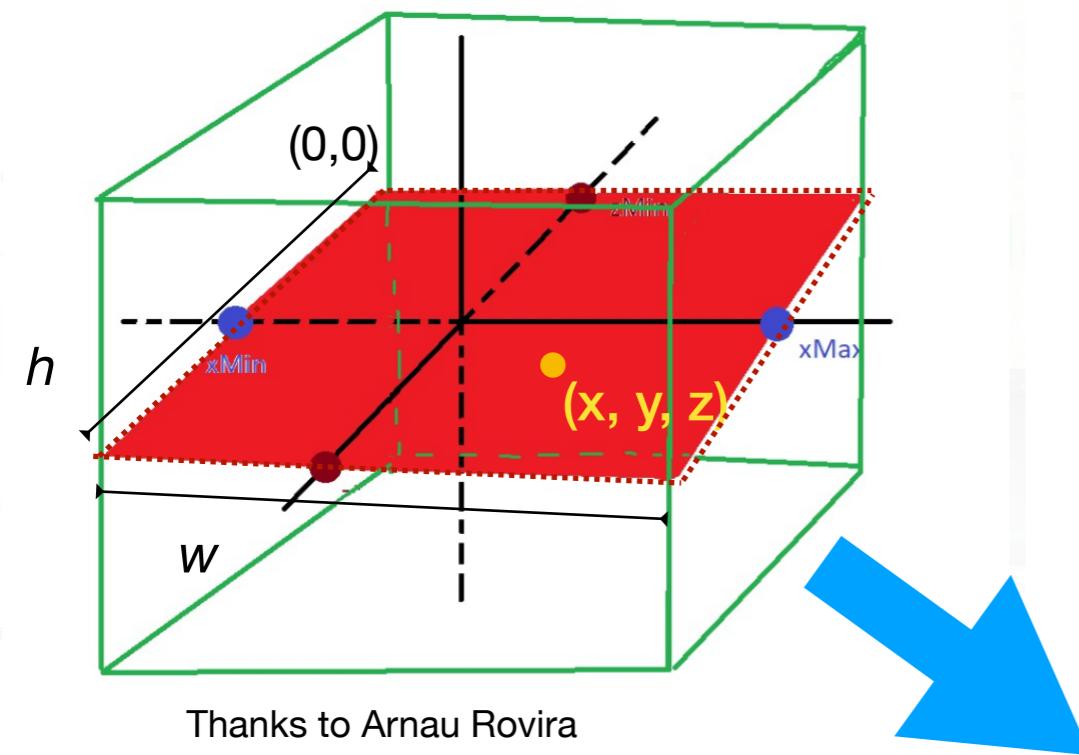


$$u = x/w$$
$$v = z/h$$

Funció de correspondència

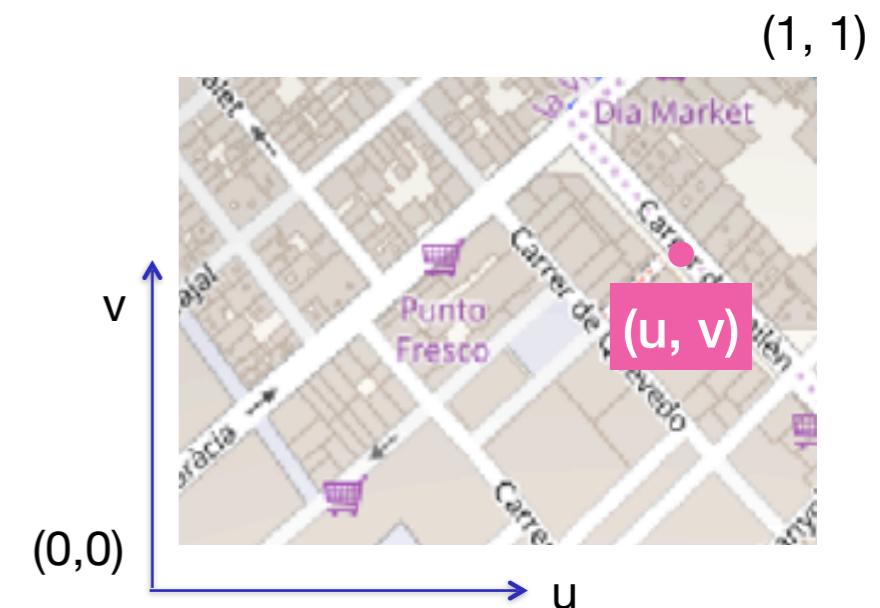
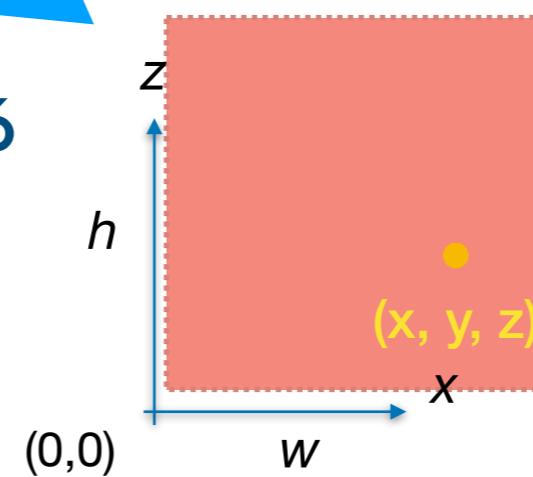
3. Recap raytracing i textures

- Pla: $(x, y, z) \rightarrow (u, v)$



Thanks to Arnau Rovira

Funció de projecció



$$u = x/w$$
$$v = z/h$$

```
#include "Texture.h"

Texture::Texture(QString nomfitxer)
{
    bool success = image.load(nomfitxer);
    if (!success) std::cout << "Imatge no trobada" << endl;
}

Texture::~Texture() {}

vec3 Texture::getColorPixel(vec2 uv) {
    QColor c(image.pixel((uv.x)*image.width(), (uv.y)*image.height()));
    return vec3(c.redF(), c.greenF(), c.blueF());
}
```

3. Recap raytracing i textures

Funció de projecció: càlcul del uv a hitInfo

VisualMapping.hh

REAL data

maxPoint

```
"Real": {  
    "minPoint": [-5, -5],  
    "maxPoint": [5, 5]  
},
```

minPoint

```
"Virtual": {  
    "minPoint": [-5, 0, -5],  
    "maxPoint": [5, 2.0, 5]  
},
```

minPoint

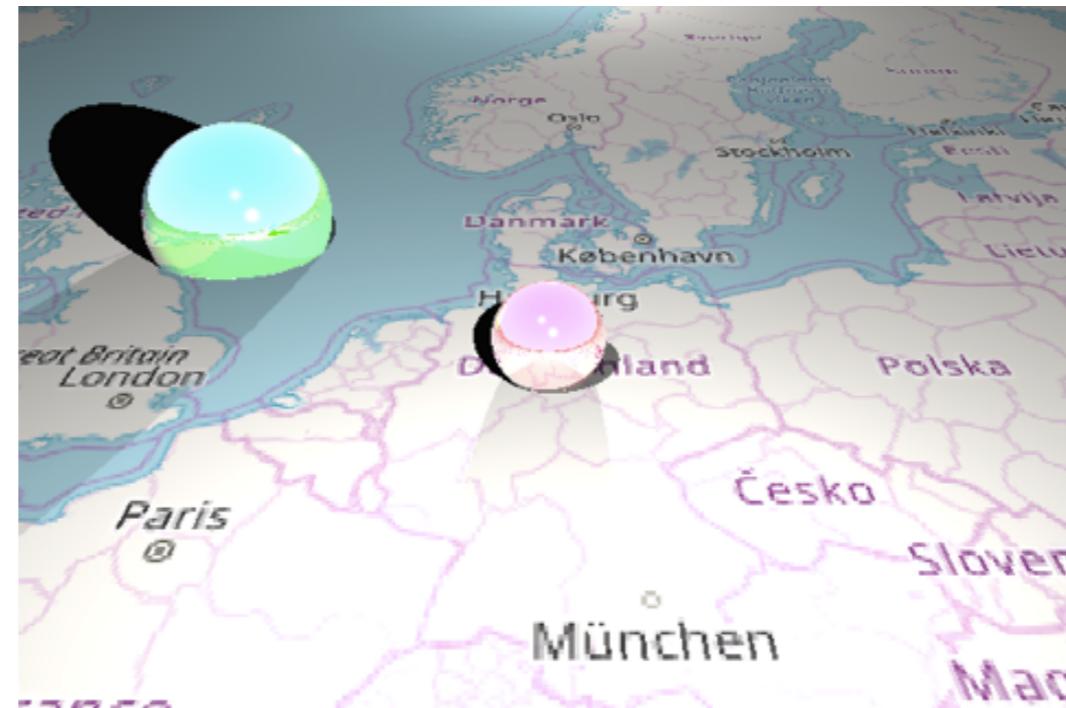
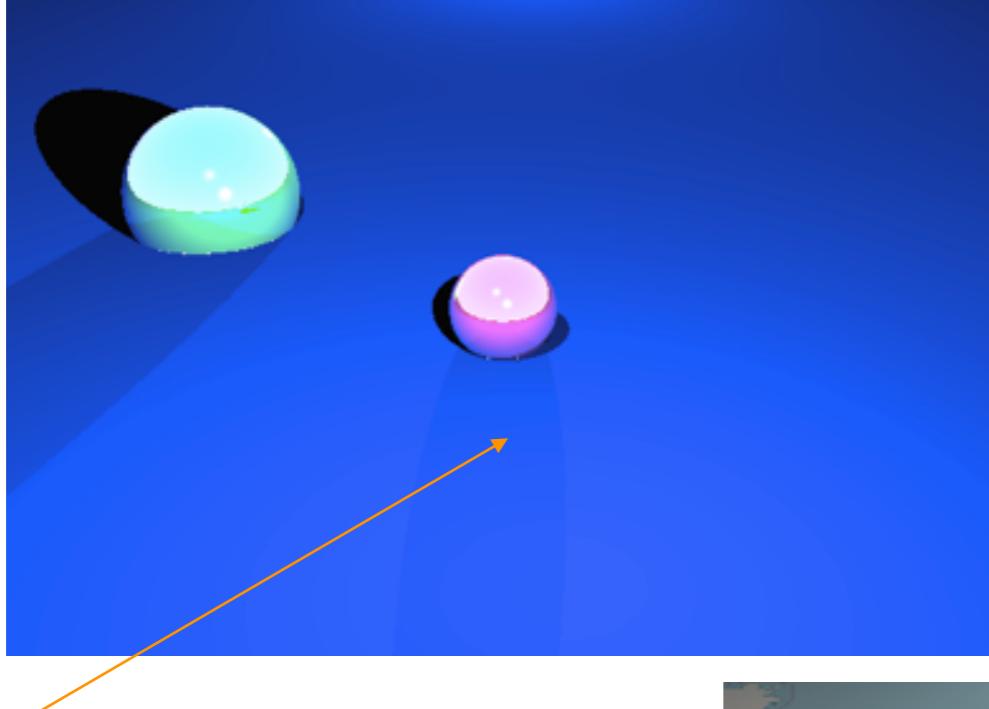
FITTEDPLANE

maxPoint

vector<AttributeMapping *> attributeMapping

```
"name": "temperatura",  
"gyzmo": "sphere",  
"minValue": 0,  
"maxValue": 2,  
"colorMap": "COLOR_MAP_TYPE_INFERNO",  
"material": {  
    "type": "lambertian",  
    "ka": [0.2, 0.2, 0.2],  
    "kd": [0.7, 0.6, 0.5],  
    "ks": [0.7, 0.7, 0.7],  
    "shininess": 10.0  
},
```

3. Recap raytracing i textures



MaterialTextura:
té un mètode
`getDiffuse(u, v)` que
retorna el color del pixel de la
textura corresponent al valor
(u,v). El seu mètode `scatter`
no retorna cap raig de sortida

3. Recap raytracing i textures

- Aplicació del valor de textura a la il·luminació

Funció de transformació del valor

Modificar valor en l'equació d'il·luminació

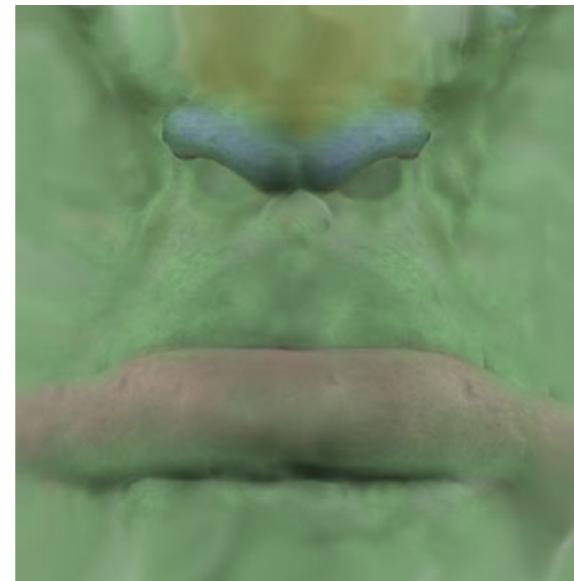
- Quins components de l'equació de Blinn-Phong es poden modificar amb el valor de la textura?

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

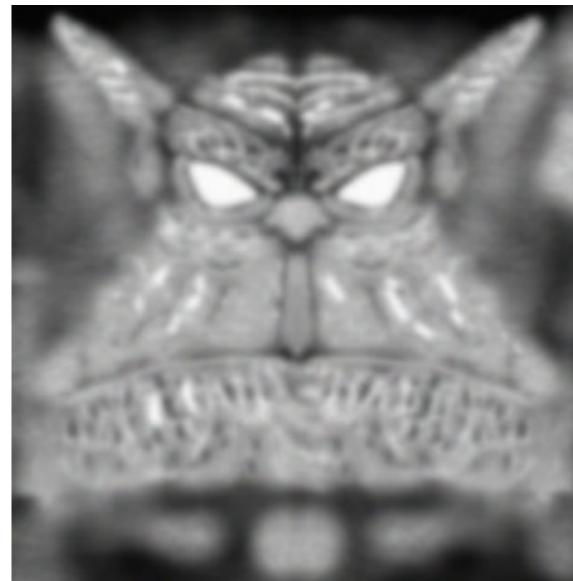
- La component difusa K_d (color)
- El vector normal (tècnica de bump-mapping)
- La component specular K_s

3. Recap raytracing i textures

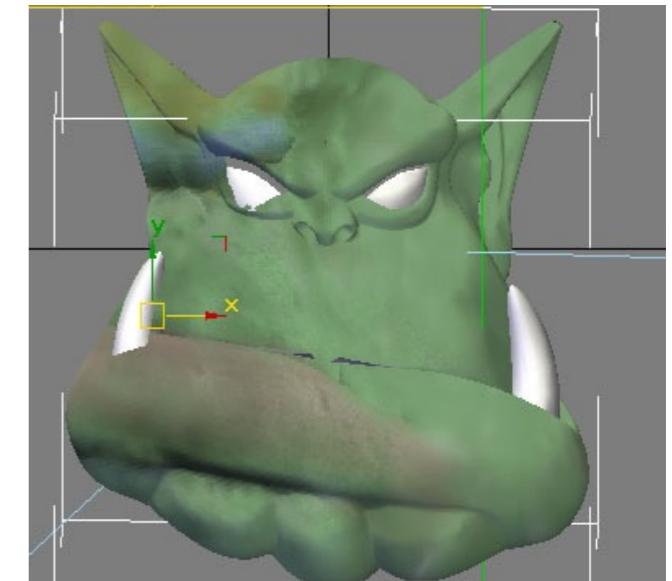
Amb el RGB de la textura es poden variar diferents propietats òptiques



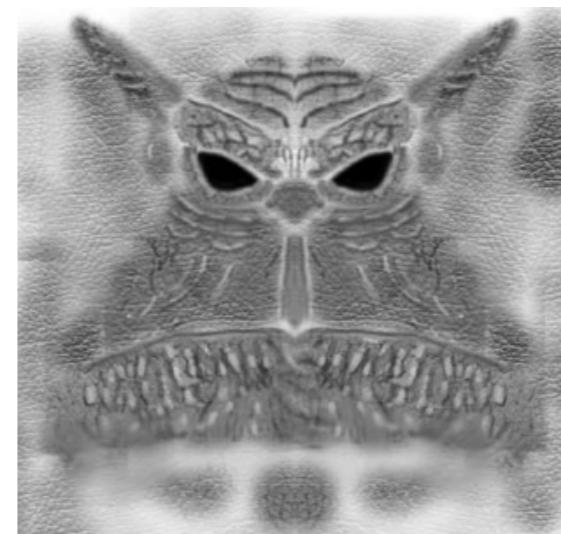
Color



Especularitat



Resultat



Normals



Resultat final

Semestre Primavera 2022-23 ● 30

3. Recap raytracing i textures

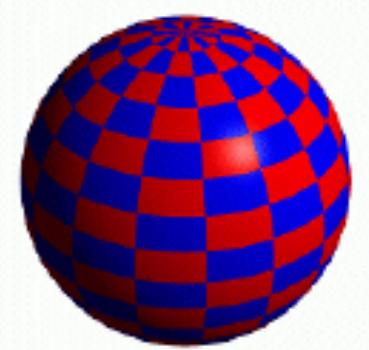
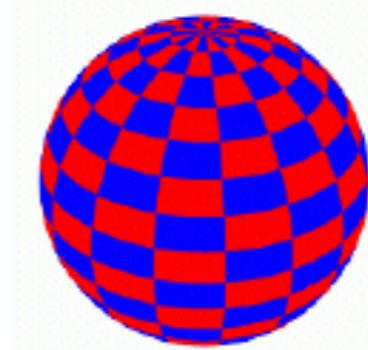
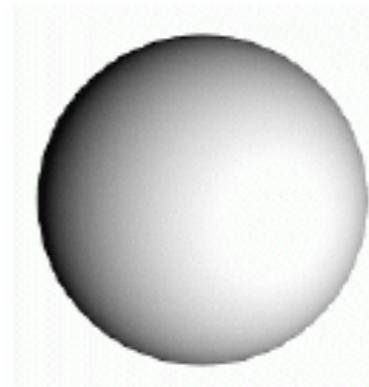
- Aplicació del valor de textura a la il·luminació

Funció de transformació del valor

Modificar valor en l'equació d'il·luminació

- Funcions de transformació

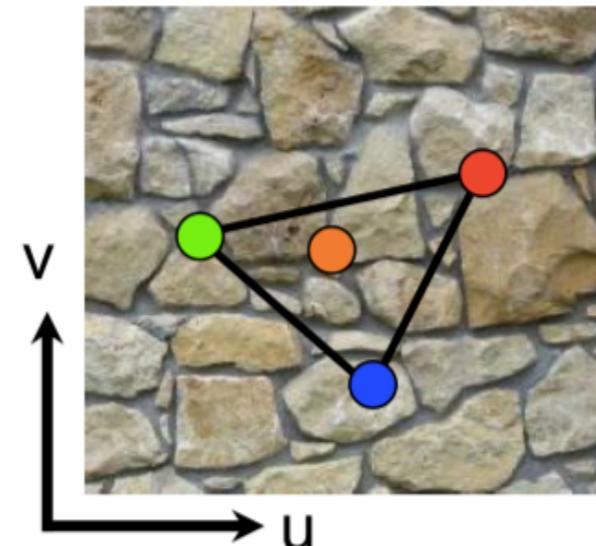
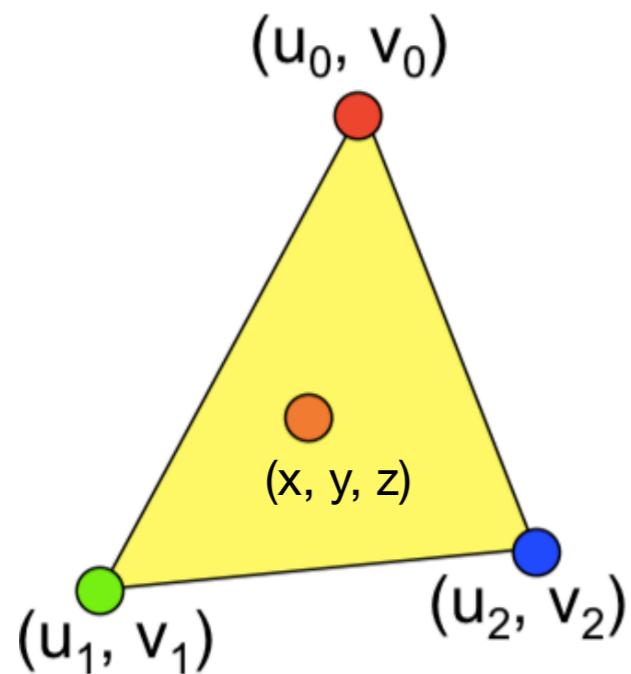
- Reemplaçar
- Modular o blending
- (AND, OR, ...)



- Serveixen per fer la composició de diferents textures amb diferents materials o propietats

3. Recap raytracing i textures

- **Triangle:** $(x, y, z) \rightarrow (u, v)$
- Per a cada vèrtex del triangle es tenen definides les coordenades (u, v)
- A cada punt interior del triangle s'interpolen les seves coordenades (u, v) segons les seves coordenades baricèntriques



<https://www.geogebra.org/m/sUuDYe85>

3. Recap raytracing i textures

```
void Mesh::load (QString fileName) {  
    QFile file(fileName);  
    if(file.exists()) {  
        if(file.open(QFile::ReadOnly | QFile::Text)) {  
            while(!file.atEnd()) {  
                QString line = file.readLine().trimmed();  
                QStringList lineParts = line.split(QRegularExpression("\\s+"));  
                if(lineParts.count() > 0) {  
                    // if it's a comment  
                    if(lineParts.at(0).compare("#", Qt::CaseInsensitive) == 0)  
                    {  
                        // qDebug() << line.remove(0, 1).trimmed();  
                    }  
  
                    // if it's a vertex position (v)  
                    else if(lineParts.at(0).compare("v", Qt::CaseInsensitive) == 0)  
                    {  
                        vertexs.push_back(vec4(lineParts.at(1).toFloat(),  
                                              lineParts.at(2).toFloat(),  
                                              lineParts.at(3).toFloat(), 1.0f));  
                    }  
  
                    // if it's a normal (vn)  
                    else if(lineParts.at(0).compare("vn", Qt::CaseInsensitive) == 0)  
                    {  
                        //  
                    }  
                }  
            }  
        }  
    }  
}
```

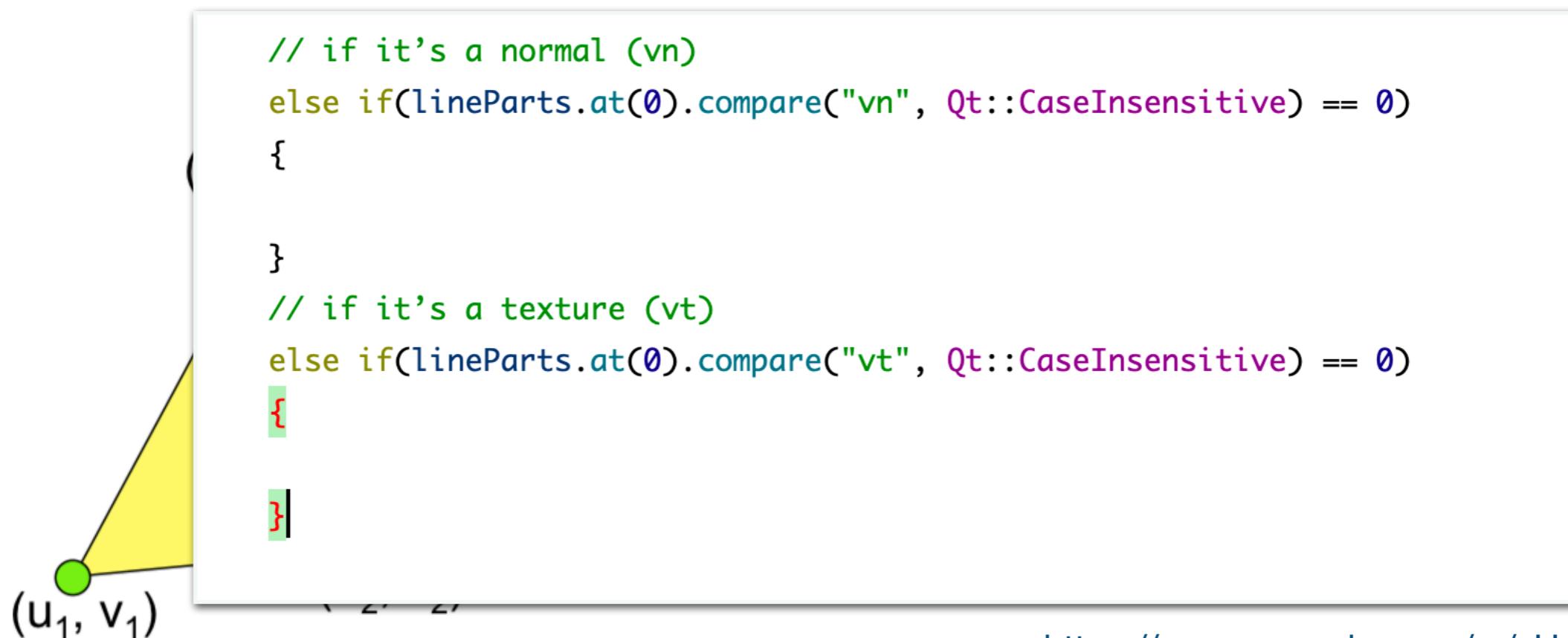
v)

)

sUuDYe85

3. Recap raytracing i textures

- **Triangle:** $(x, y, z) \rightarrow (u, v)$
- Per a cada vèrtex del triangle es tenen definides les coordenades (u, v)
- A cada punt interior del triangle s'intercala la seva coordenada (u, v) segons les seves coordenades baricèntriques

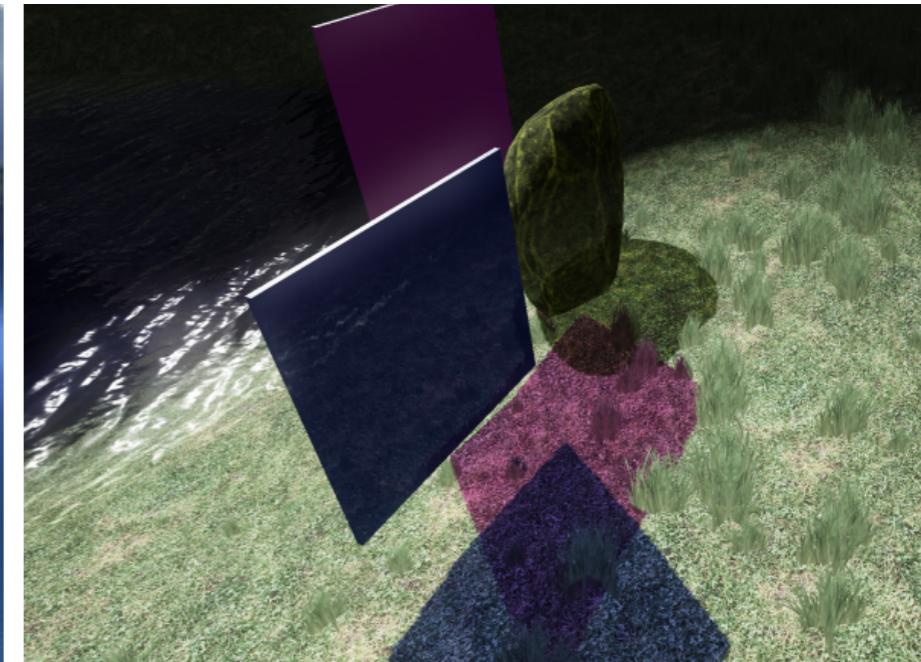


<https://www.geogebra.org/m/sUuDYe85>

4. Extensions

Efectes d'il·luminació

- Ambient Occlusion
- Defocus blur
- Environmental mapping
- Filtres de llums

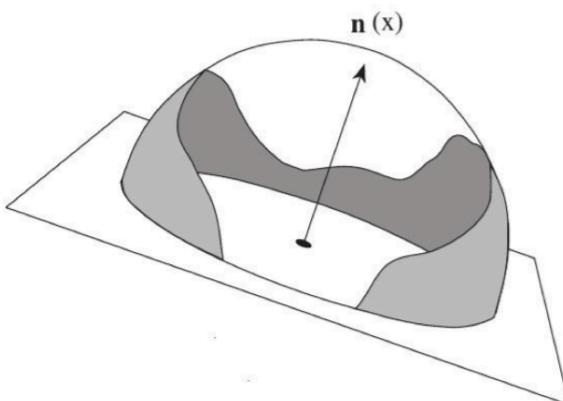


4. Extensions

- Ambient occlusion
 - Ombres provocades per oclusió de la llum ambient en llocs on arriba menys llum

Afecta als termes de la llum ambient: es calcula un factor sobre els termes ambient:

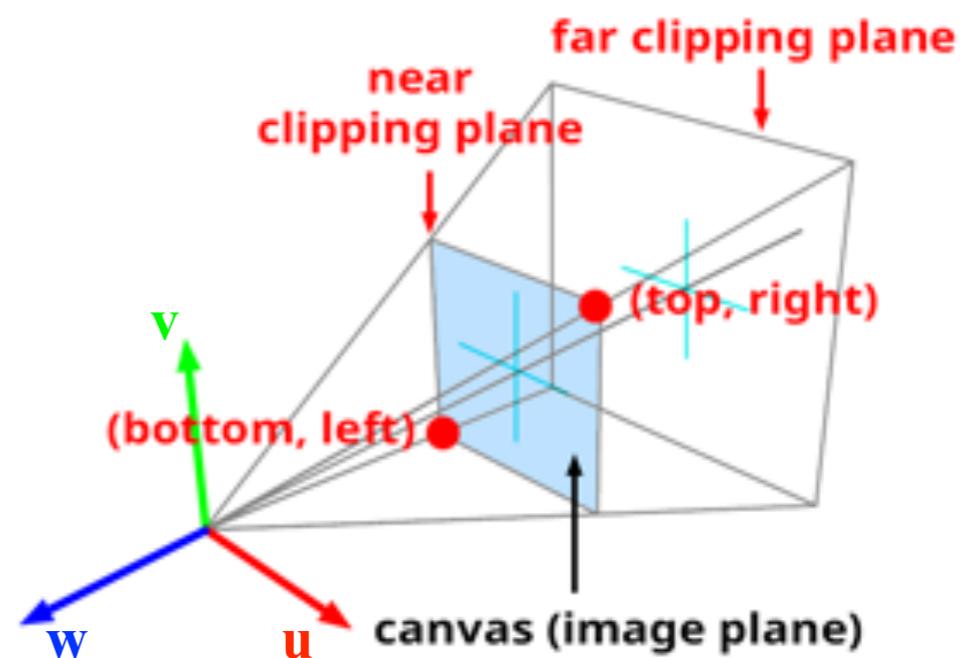
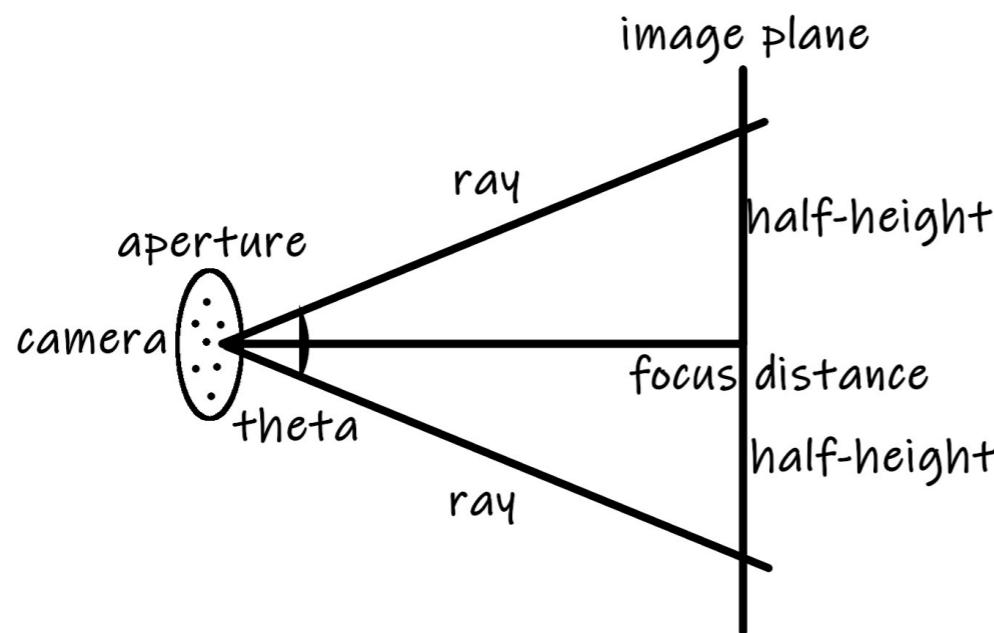
- ambient global (**escenes outdoor**): Es llencen n rajos aleatoris des del punt d'intersecció cap al “cel” i es calcula un factor per a ponderar el terme ambient
$$\text{numeroRajosCel} / \text{numeroRajosTotals}$$
- ambient de cada llum (**escenes indoor**): es calcula un terme a cada punt. Es llancen rajos per veure si tenen occlusors apropiats. Es el mateix per totes les llums.
$$1.0 - \text{numeroRajosIntersectenObjs} / \text{numeroRajosTotal}$$



4. Extensions

- Defocus blur:

Per simular l'apertura només cal canviar lleugerament l'origen del raig primari en el moment de calcular el raig
(`Camera.cpp`)

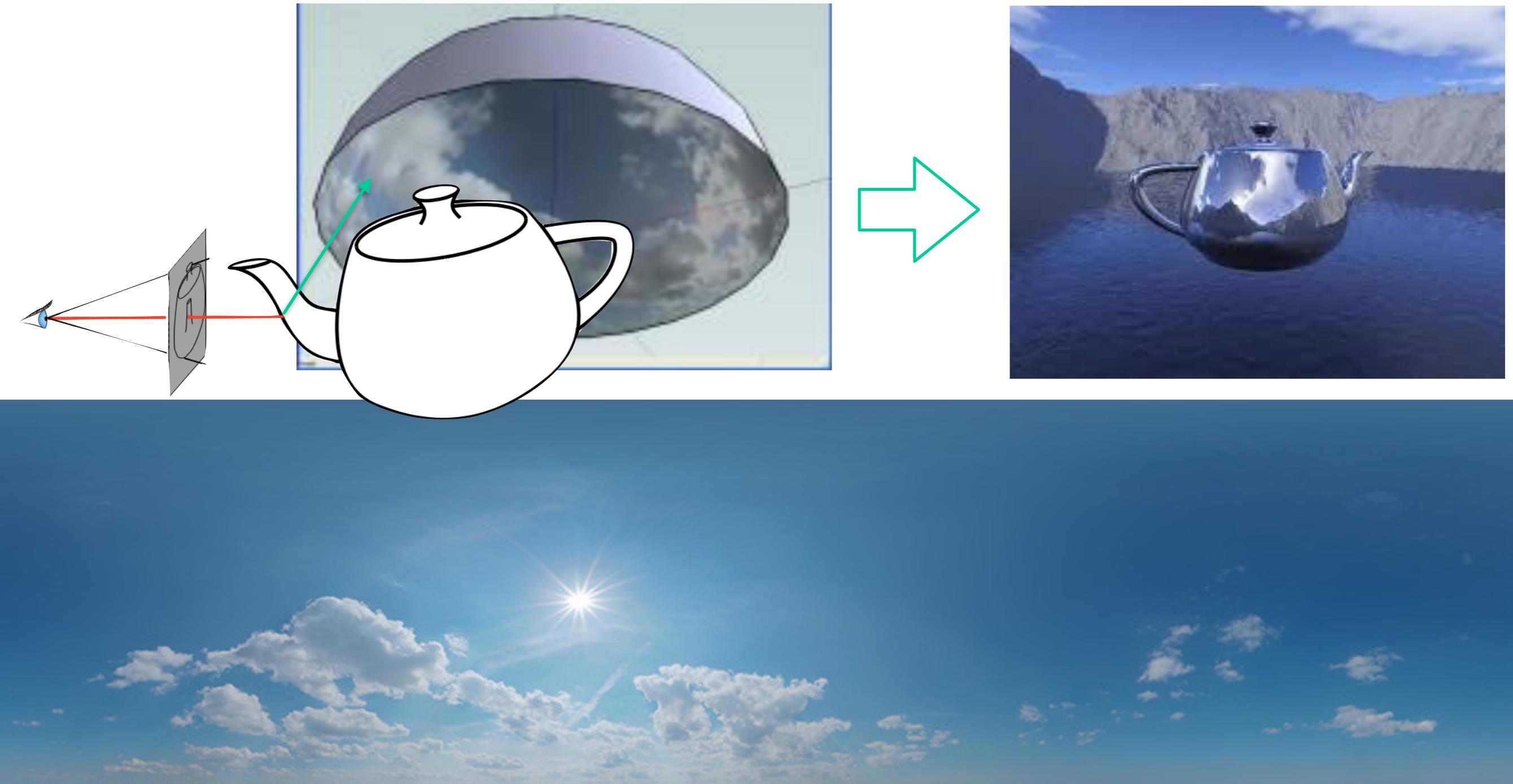


© www.scratchapixel.com

Raig:

```
punt_origen = lookFrom + offset;  
offset = u *random.x + v * random.y;  
random = lens_radius * random_in_unit_disk();
```

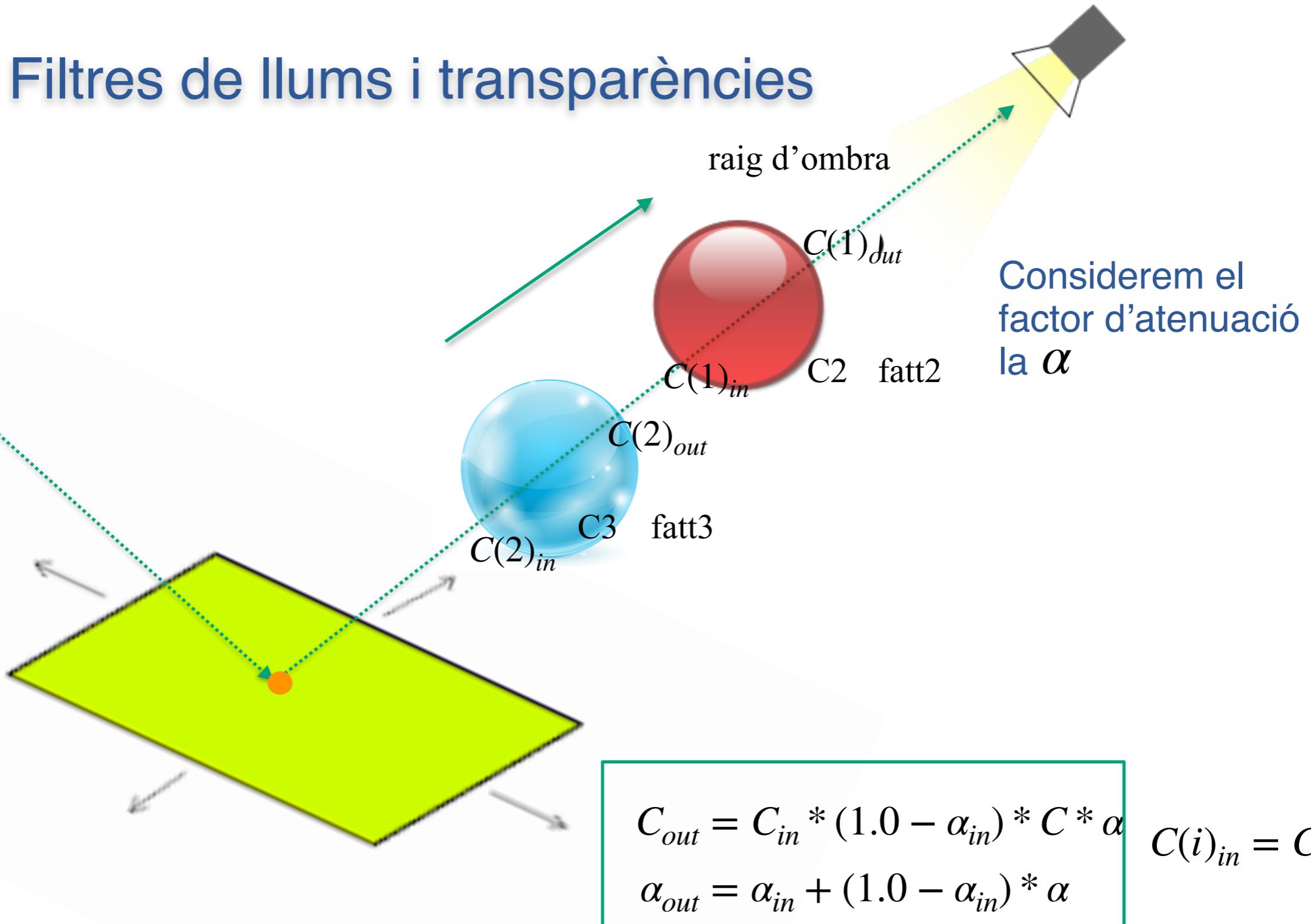
4. Extensions



<https://photostockeditor.com/premium/218725729/full-360-degree-seamless-panorama-in-equirectangular-spherical-equidistant-projection-panorama-view>

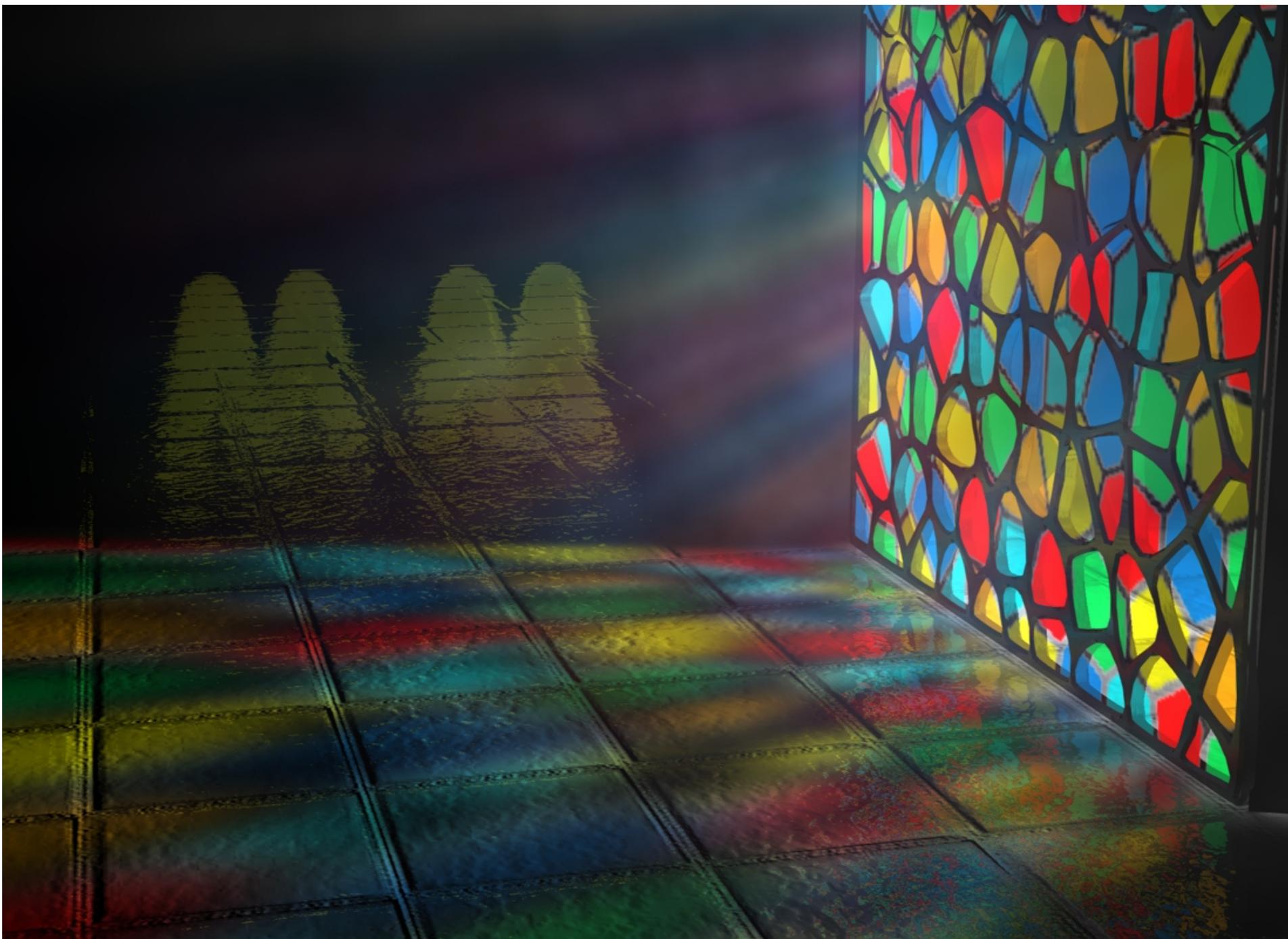
4. Extensions

- Filtres de llums i transparències



4. Extensions

- Filtres de llums i transparències amb textures



4. Extensions

- Visualization mapping
 - ús de paletes diferents
 - definició de n propietats (localització i gizmo)
 - animacions de les dades
 - mapeig de les dades en una esfera

