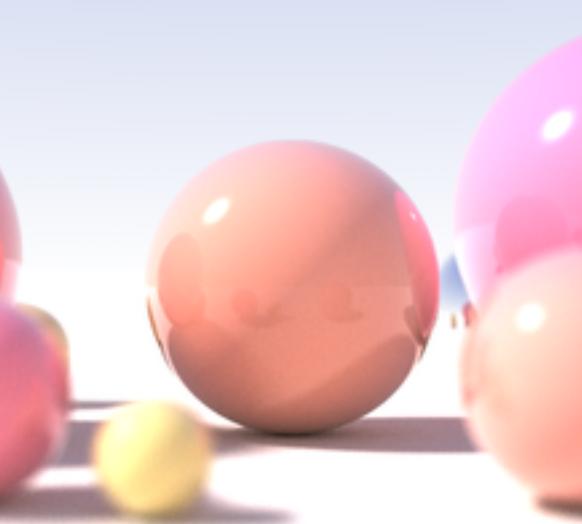


Gràfics i Visualització de Dades



T2a: RayTracing

Anna Puig

Índex

2.1. Introducció: algorisme principal

2.2. Càlcul de Raig Primari

2.3. Càlcul del color:

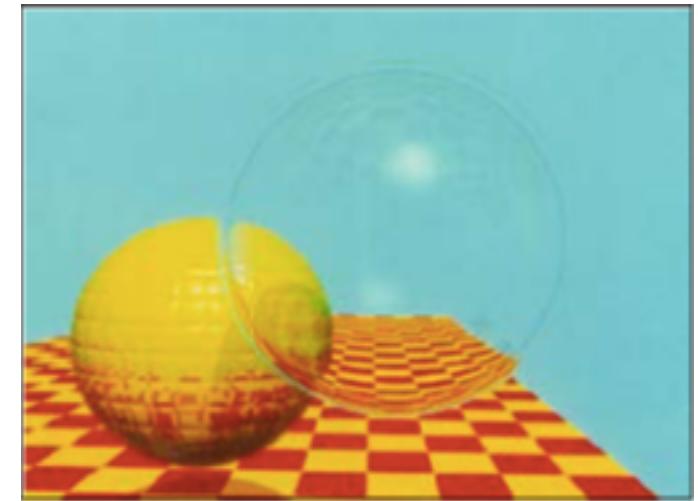
 2.3.1. Interseccions amb objectes

 2.3.2. Materials i Llums

2.4. Ombres

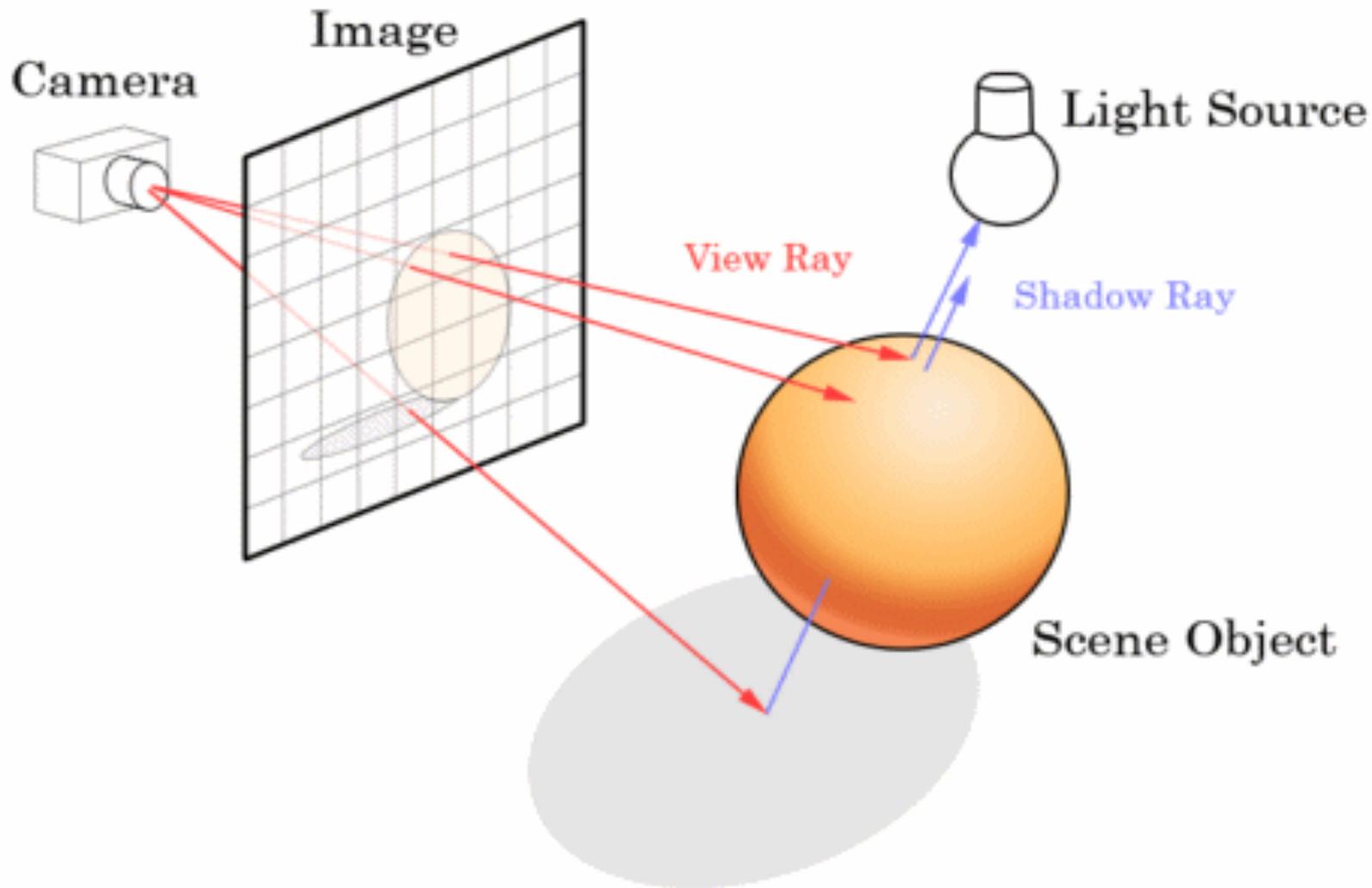
2.5. Reflexions i transparències

2.6. RayTracing en models de volum



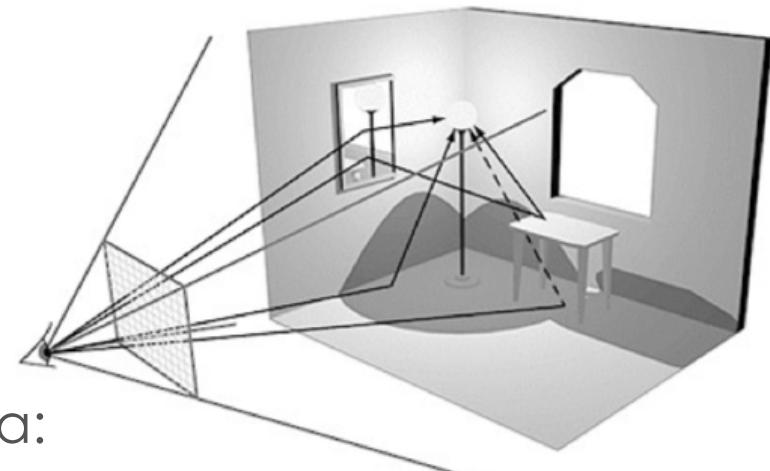
Whitted, 1980

2.1. Introducció



2.1. Introducció

Òptica geomètrica (la llum es considera ona i partícula)

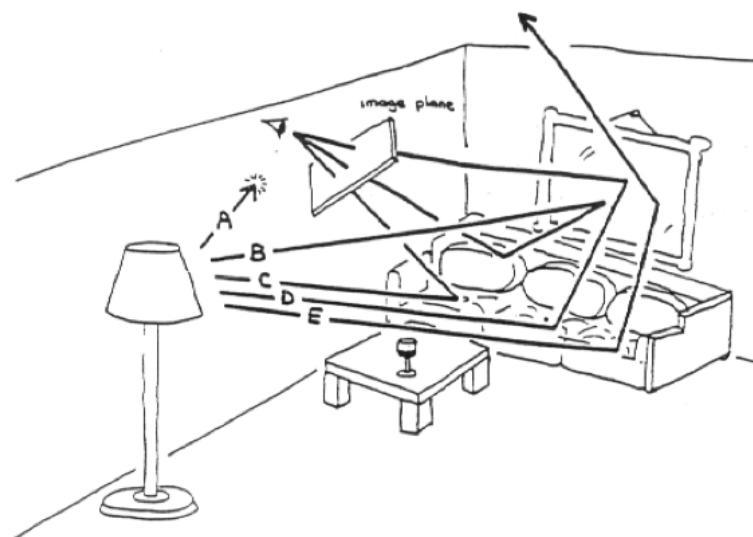


Algunes regles de l'òptica geomètrica:

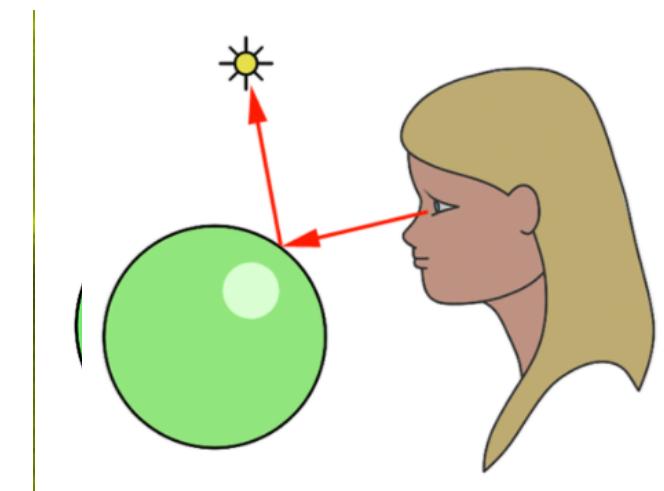
- La llum és un flux de fotons amb longitud d'ona que anomenarem “**rajos de llum**”
- Els rajos de llum viatgen en **trajectòries rectilínies**
- Els rajos de llum no interfereixen entre ells quan es creuen
- Els rajos de llum obeyeixen les **lleis de reflexió i refracció**
- Els rajos de llum viatgen des de les fonts de llum fins els objectes i la física és invariant en sentit invers (**principi de reciprocitat**)

2.1. Introducció

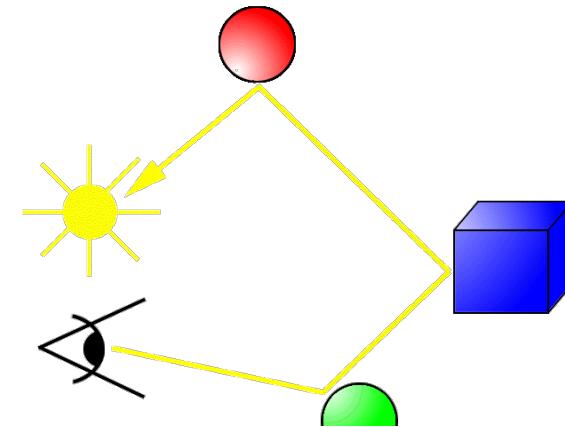
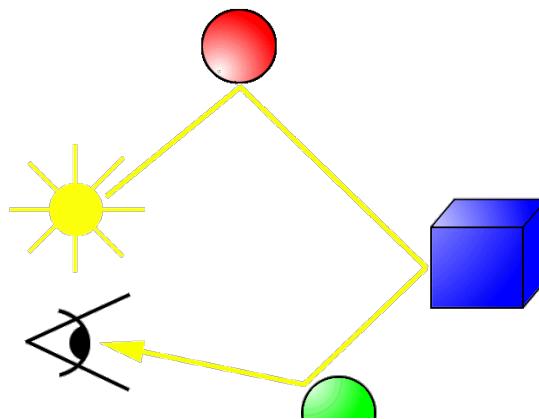
- **Problema a solucionar:** càlcul dels fotons que impacten a la imatge.



Forward RayTracing (photon tracing)



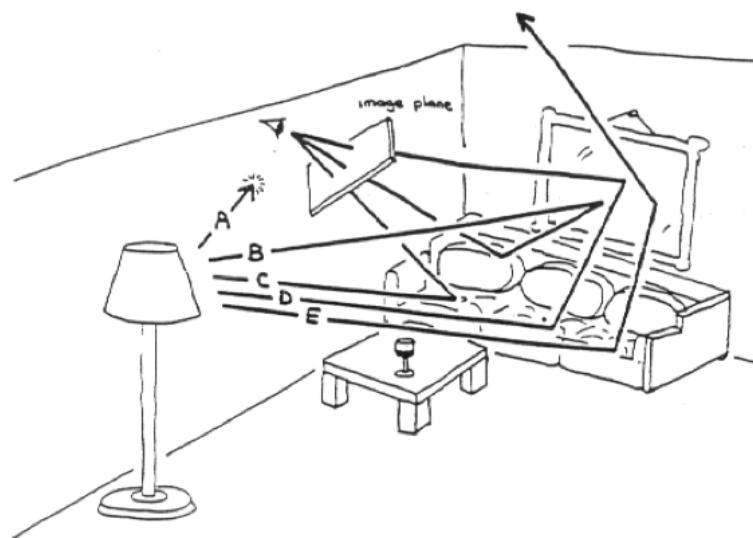
Backward RayTracing



<https://cs.stanford.edu/people/eroberts/courses/soco/projects/1997-98/ray-tracing/types.html>

2.1. Introducció

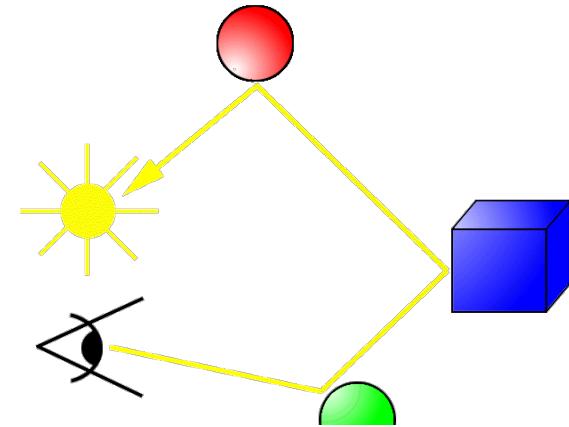
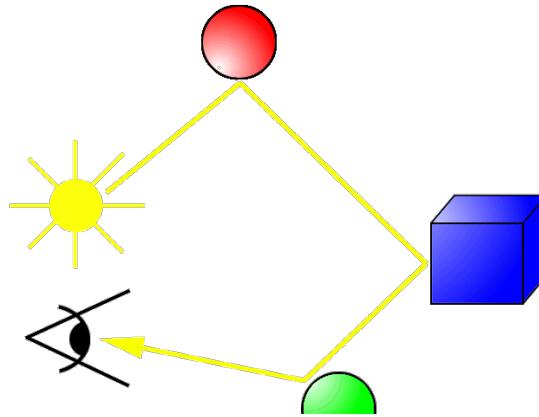
- **Problema a solucionar:** càlcul dels fotons que impacten a la imatge.



Forward RayTracing (photon tracing)



Backward RayTracing

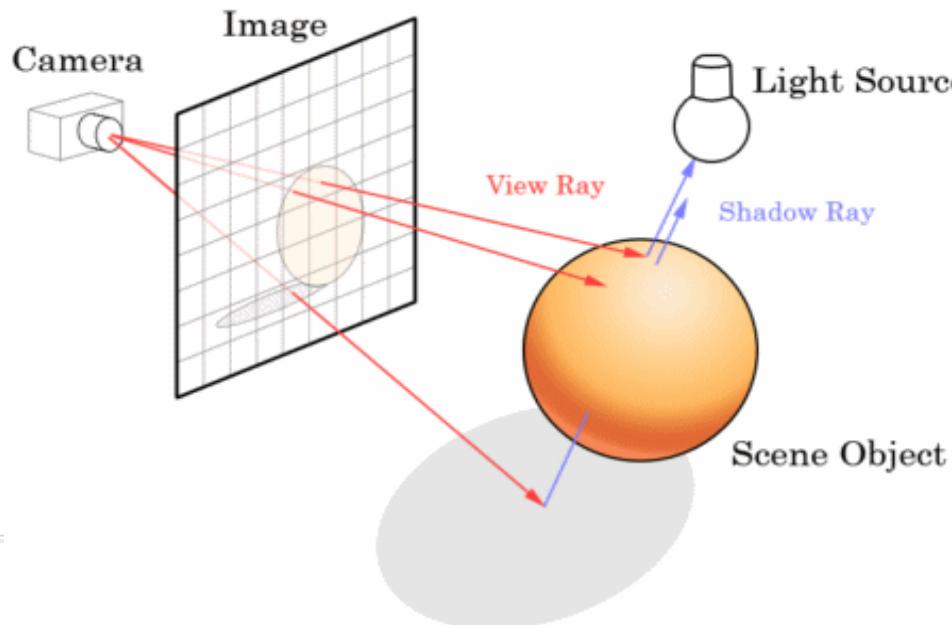
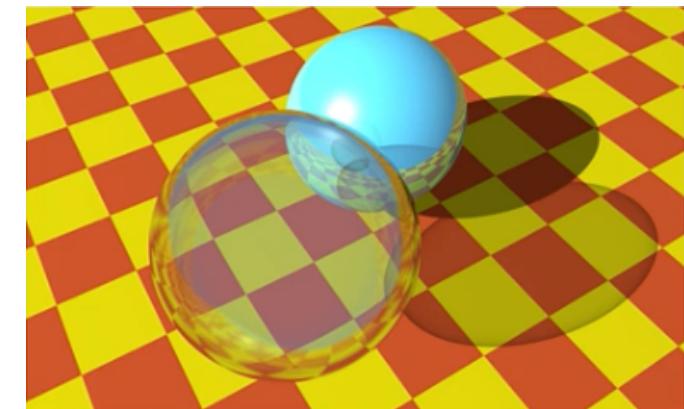


<https://cs.stanford.edu/people/eroberts/courses/soco/projects/1997-98/ray-tracing/types.html>

2.1. Introducció

Appel'68, Whitted'80, Heckbert'90, Veach'95

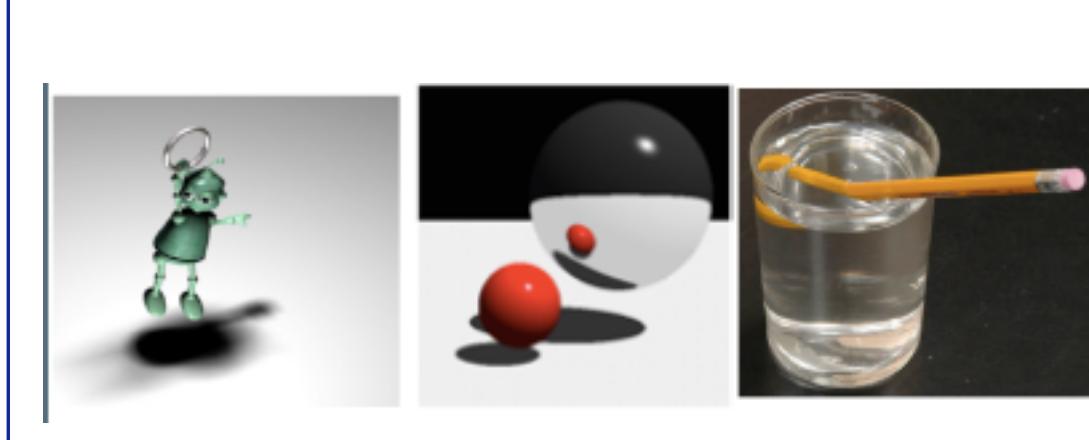
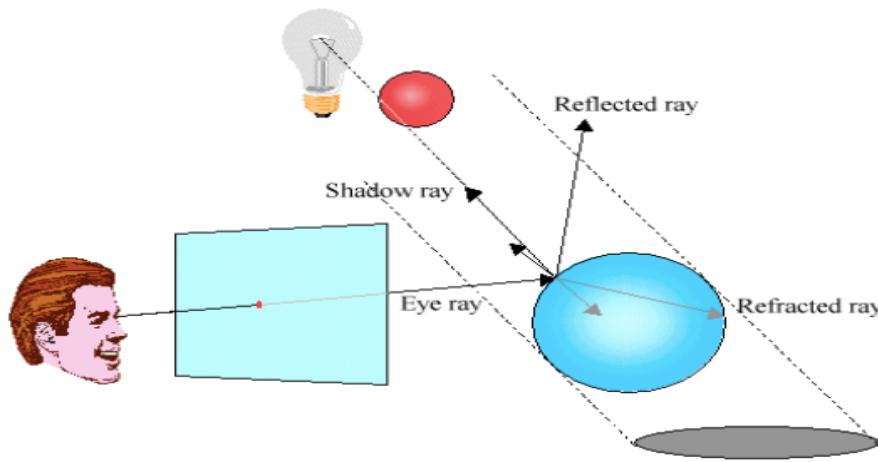
- **Backward raytracing:** Un dels mètodes més utilitzats en Gràfics per Computador per a visualitzar una imatge
- Alternativa al z-buffer
- Adient per a simular reflexions especulars, produirombres, transparències, etc.
- Usat amb altres tècniques d'il·luminació global.



2.1. Introducció

Algorisme recursiu: (computeRayColor)

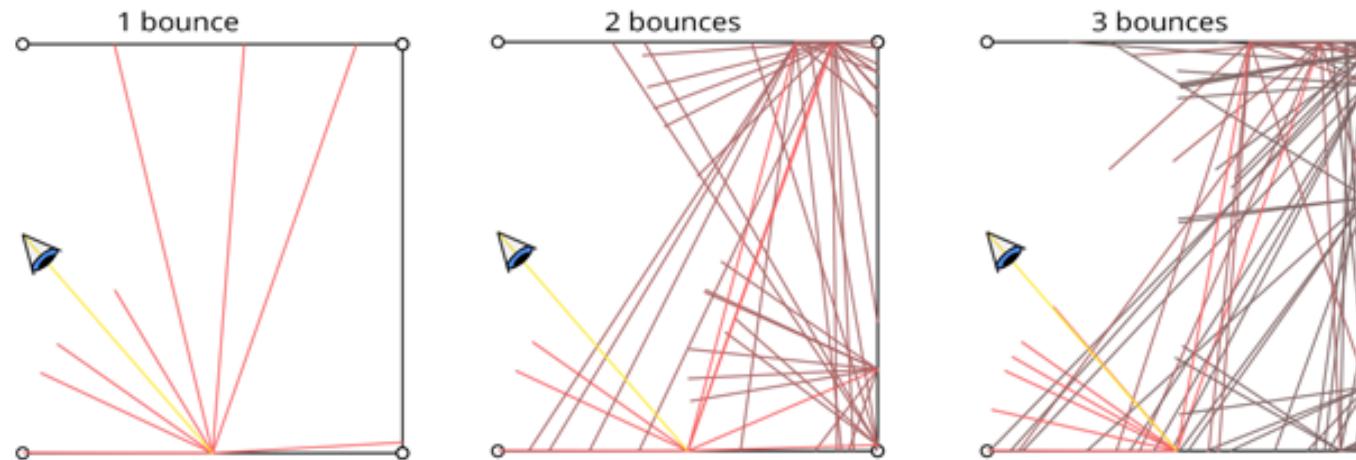
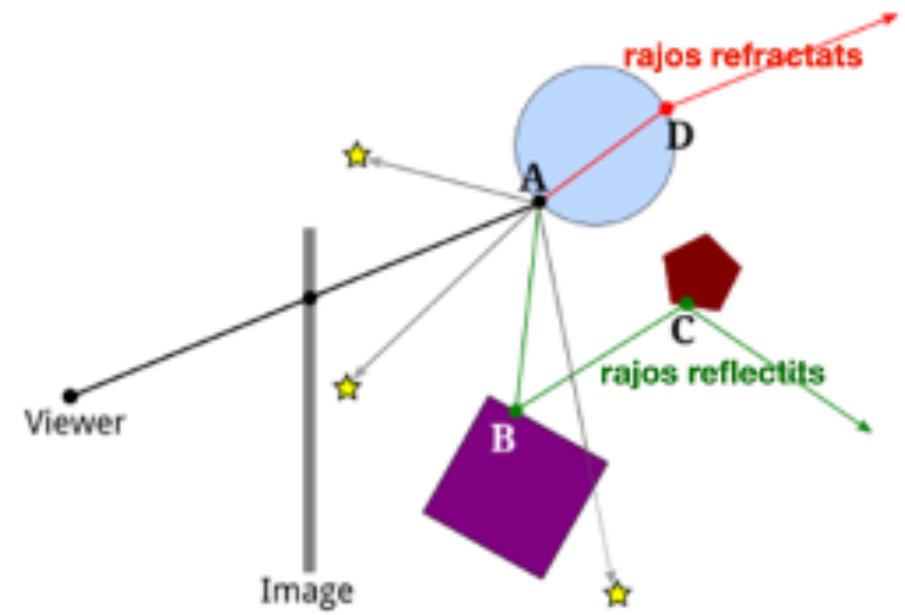
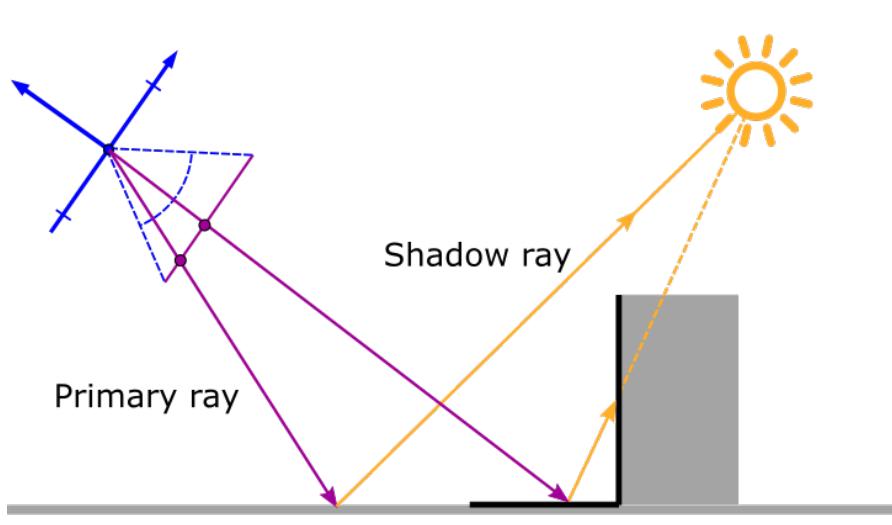
1. **Rajos primaris (eye ray):** des de l'observador a l'escena
2. **Rajos d'ombra (shadow ray):** des del punt de l'objecte a les llums
3. **Rajos secundaris:**
 - 3.1. **Rajos de reflexió (reflected ray):** des del punt de l'objecte en direcció de mirall
 - 3.2. **Rajos de transparència (refracted ray):** des del punt de l'objecte en direcció de refracció



2.1. Introducció

- **Raycast()** // generar una imatge per cada pixel x,y
 $r = \text{ComputeRay}(x, y)$
 color(pixel) = **RayPixel(ray)**
 fper
- **RayPixel(ray)** // llançar un raig, retorna RGB
 hitInfo= **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 o en l'escena
 o->**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
 retorna color

2.1. Introducció



© www.scratchapixel.com

Crèdits

Eric Haines, NVIDIA's expert,

<https://news.developer.nvidia.com/ray-tracing-essentials-part-1-basics-of-ray-tracing/>

RayTracing Gems,

<https://www.realtimerendering.com/raytracinggems/>

Vídeo Star Wars:

<https://www.youtube.com/watch?v=1do53NjBrTw>

Índex

2.1. Introducció: algorisme principal

2.2. Càlcul de Raig Primari

2.3. Càlcul del color:

 2.3.1. Interseccions amb objectes

 2.3.2. Materials i Llums

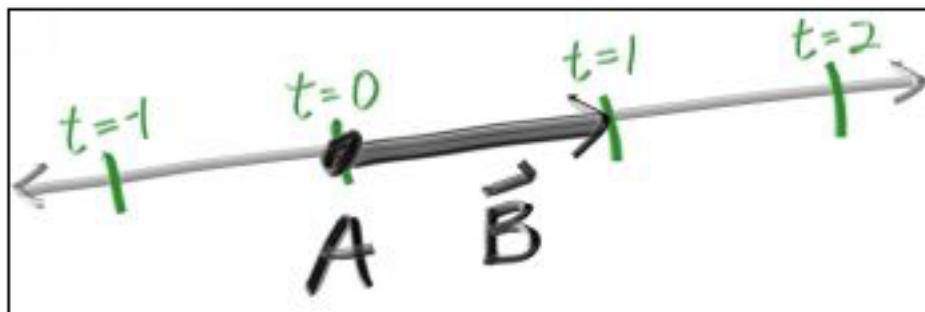
2.4. Ombres

2.5. Reflexions i transparències

2.6. RayTracing en models de volum

2.1. Introducció

- **ComputeRay:** Com es defineix un raig? Segons la seva recta paramètrica

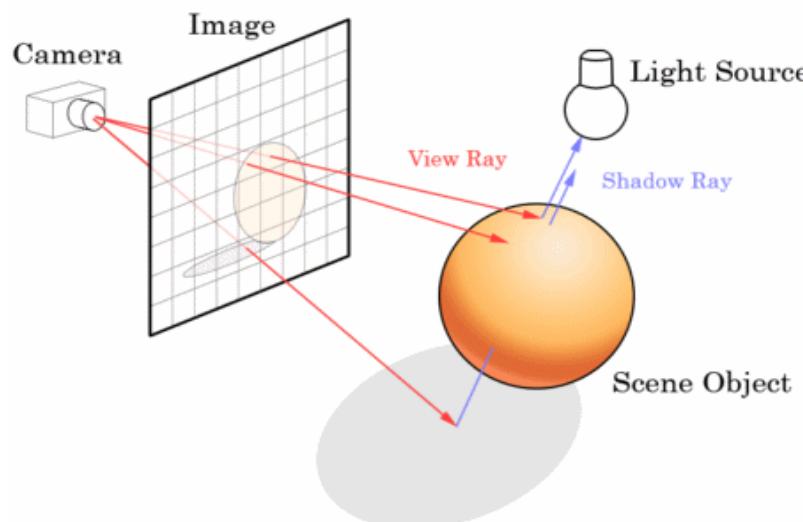


$$p = A + \vec{t} * \vec{B}$$

A punt inicial

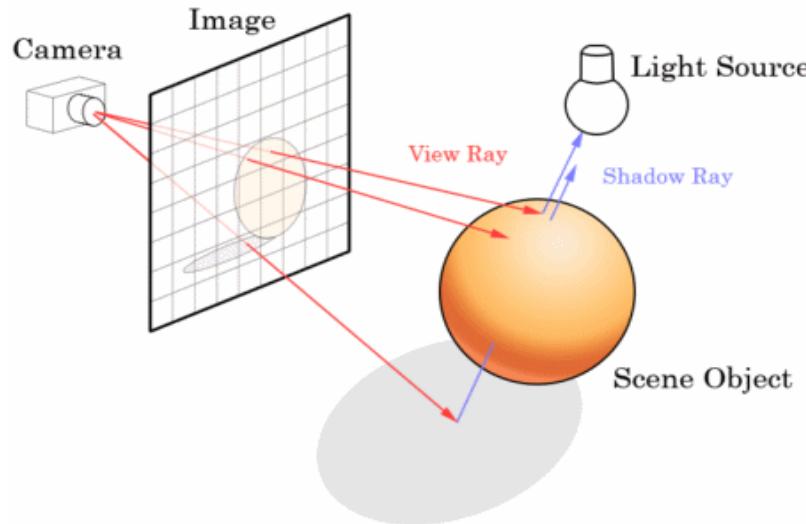
\vec{B} vector director

t paràmetre de la recta

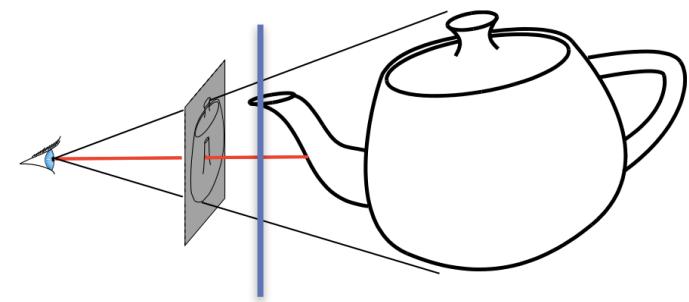


2.2. Rajos primaris

- **Rraig Primari:** Raig des de l'observador fins l'escena



Rraig des l'observador que passa pel píxel (i , j)

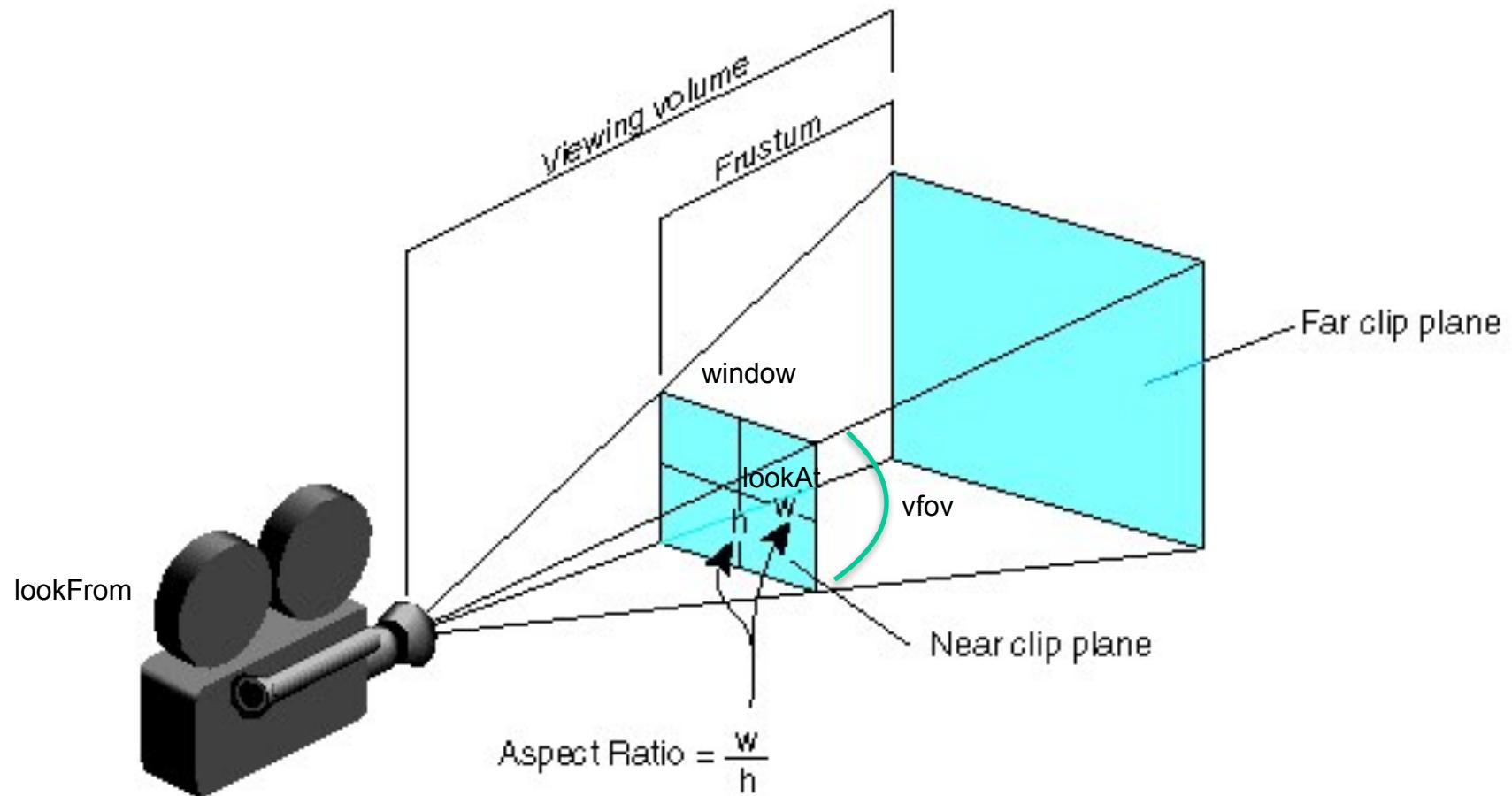


Rraig definit amb dos punts en coordenades de món (3D): p_0 i p_1 .

On $A = p_0$ i $B = p_1 - p_0$ en l'equació de la recta $p = A + tB$

Per tant es necessita saber on està l'observador i el tros de món 3D que es veurà des de la càmera.

2.2. Càmera: atributs



<https://cs.wellesley.edu/~cs307/threejs/demos/Camera/camera-api.shtml>

2.2. Càmera: atributs

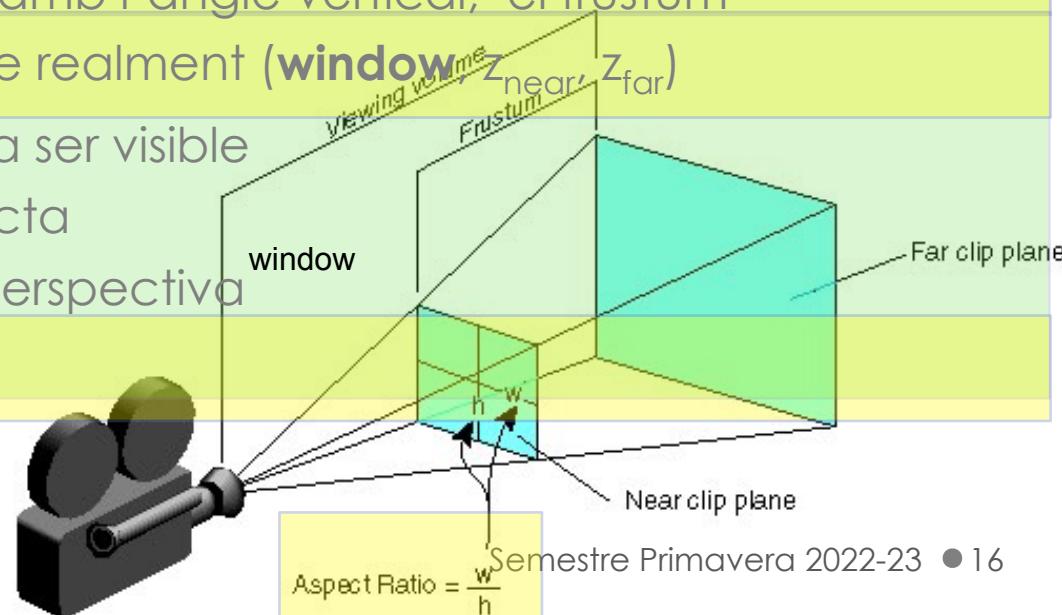
Posició i orientació

- Components de la càmera:

- **Posició de l'observador:** punt 3D en coordenades de món on es situa l'observador o el viewpoint. (lookFrom)
 - Pot ser donada per tres angles i una distància
- **VRP:** View Reference Point: punt 3D en coordenades de món on mira l'observador (lookAt)
- **VUP:** vector de verticalitat: vector 3D que defineix la direcció vertical de l'observador
- **D:** distància de l'observador al VRP

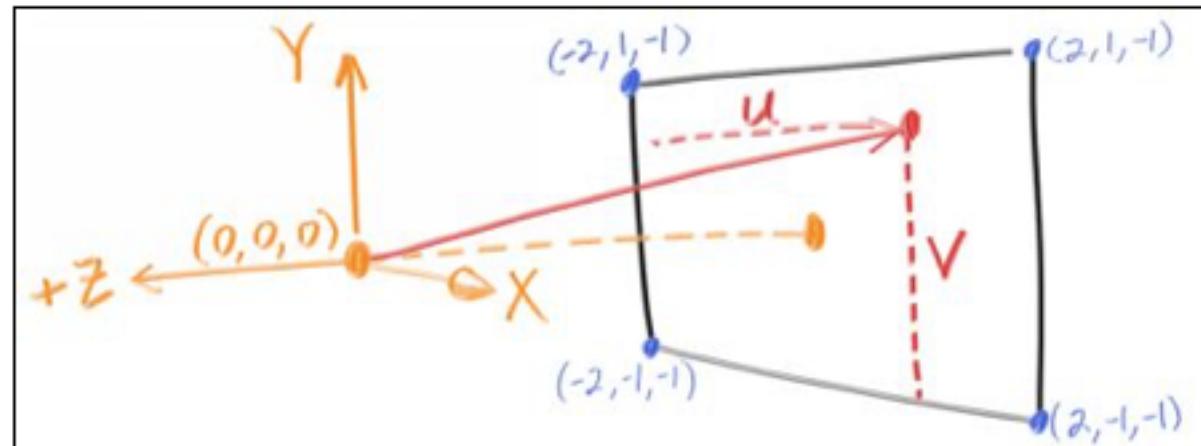
Projecció i límits de visió

- **Obertura de la càmera:** defineix amb l'angle vertical, el frustum
- **Frustum:** volum de l'escena visible realment (**window**, z_{near} , z_{far})
- **Volum de visió:** volum candidat a ser visible
- **Pla de projecció:** pla on es projecta
- **Tipus de projecció:** paral·lela o perspectiva
- **Viewport:** imatge de píxels



2.2. Rajos primaris

- **Rraig Primari:** Amb versió simple de la càmera



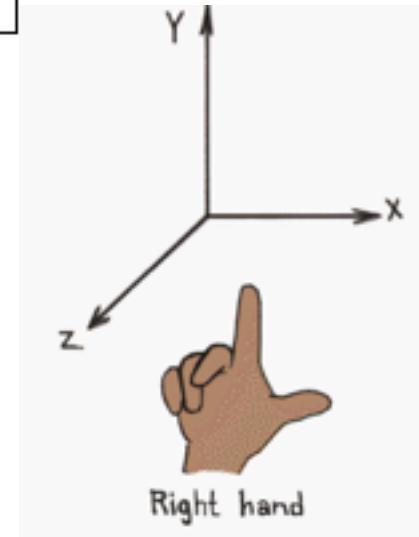
- **Convenció:**

- el món 3D sempre està a l'espai de les Z's negatives

- **Simplificació:**

- imatge de 200×100 píxels origen $(0, 0)$ (viewport)
- observador en el $(0, 0, 0)$ del món 3D
- distància focal = 1 (pla de projecció a $Z = -1$)
- $\text{vfov} = 90^\circ$ (vertical field-of-view)

- aspect ratio = 2 , per tant window $(-2, -1)$ a $(2, 1)$



2.2. Rajos primaris

- **Rraig Primari:** Amb versió simple de la càmera

```
per j=nPixelsY-1; j>=0; j - -  
per i =0 ; i < nPixelsX; i++
```

```
float u' = i / nPixelsX;
```

```
float v ' = (nPixelsY - j )/ nPixelsY;
```

```
raig = calculaRaig( punt_origen, vector_director) ???
```

```
color(pixel) = RayPixel (raig)
```

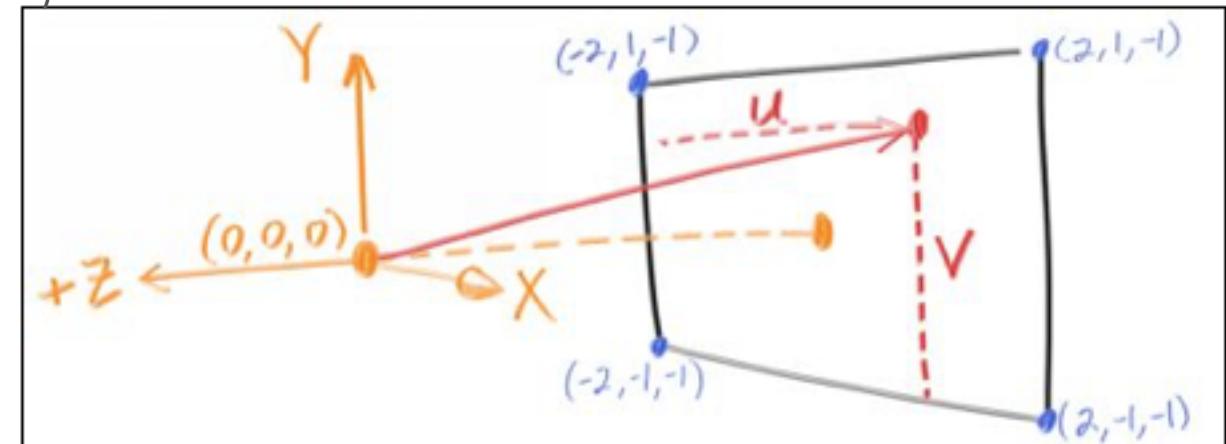
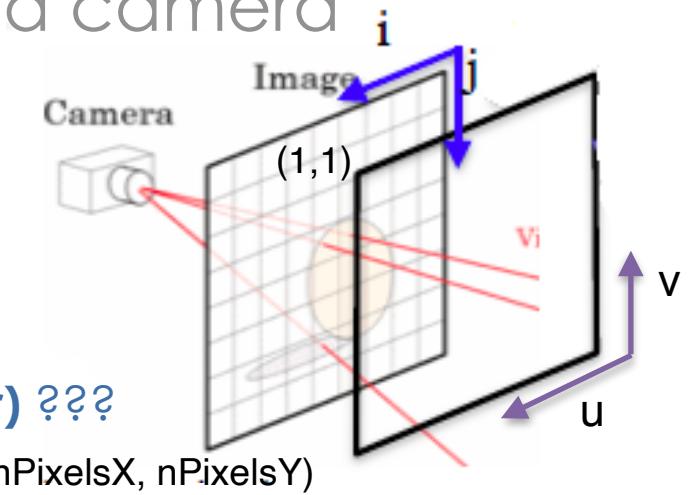
```
posaColorPixel(i, j, color)
```

```
fper
```

```
fper
```

```
punt_origen = (0,0,0)
```

```
vector_director =  
(lower_left_corner +  
horitzontal * u +  
vertical * v - punt_origen)
```



```
horitzontal = 4*(1,0,0); vertical = 2*(0,1,0);
```

Índex

2.1. Introducció: algorisme principal

2.2. Càlcul de Raig Primari

2.3. Càlcul del color:

2.3.1. Interseccions amb objectes

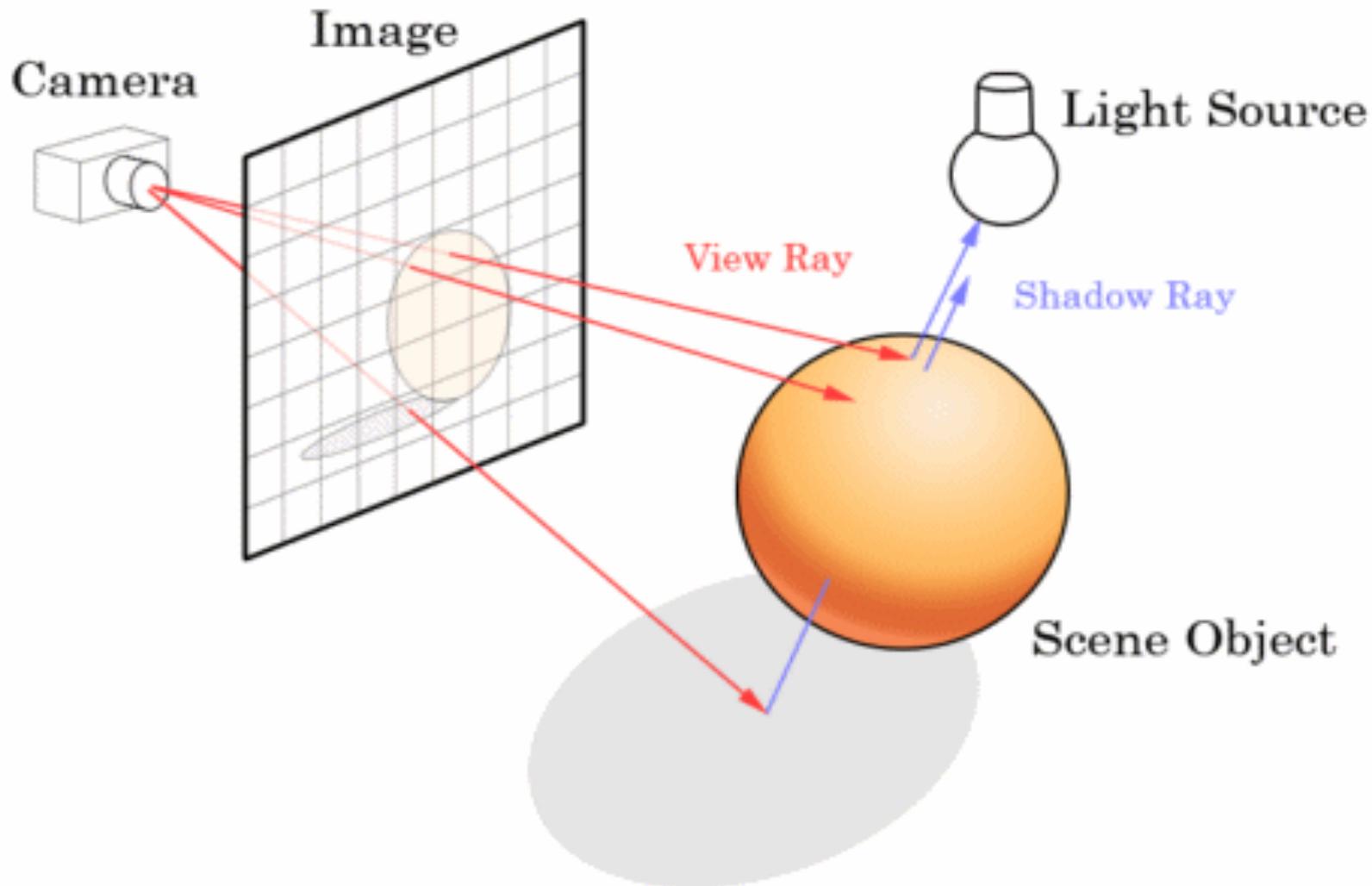
2.3.2. Materials i Llums

2.4. Ombres

2.5. Reflexions i transparències

2.6. RayTracing en models de volum

2.1. Introducció



2.1. Introducció

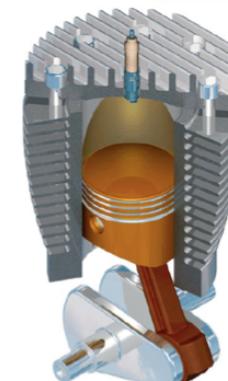
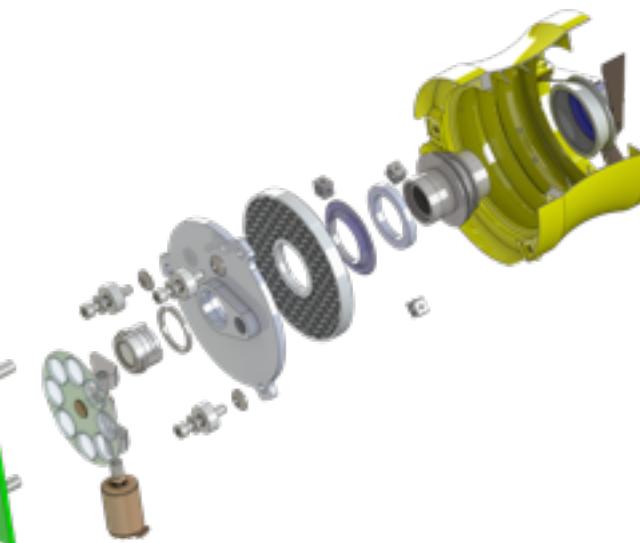
- **Raycast()** // generar una imatge
 - per cada pixel x,y
 - r = **ComputeRay**(x, y)
 - color(pixel) = **RayPixel(ray)**
- **RayPixel(ray)** // llançar un raig, retorna RGB
 - intersectInfo= **hit(ray)**
 - // intersection (ray, tmin, tmax, hitinfo)
 - si object_point retorna **Shade(hitInfo, ray)**
 - sino retorna Background_Color // o Intensitat ambient global
- **hi(ray)**
 - per cada objecte en l'escena
 - hit**(ray, surface) //intersection (ray, tmin, tmax, hitinfo)
 - retorna el punt més proper a l'observador (+normal, +material)
- **Shade(hitInfo, ray)** // retorna la il·luminació en el point
 - retorna color

2.3.1. Interseccions amb l'escena

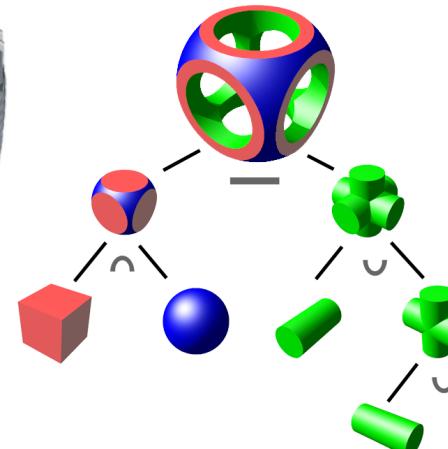
Una **escena** està formada per un conjunt d'objectes

- Com s'estructura?

Llista d'objectes

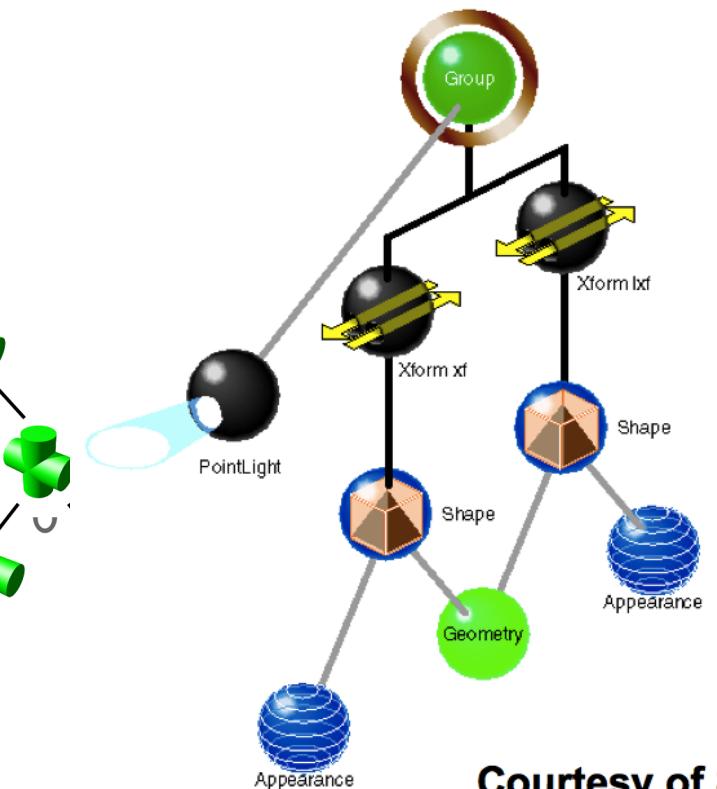


Arbres CSG



- Intersecció amb el raig?
 - Recorregut per tots els objectes i es troba la intersecció més propera amb $t > 0$, si l'escena només està formada per objectes superficials

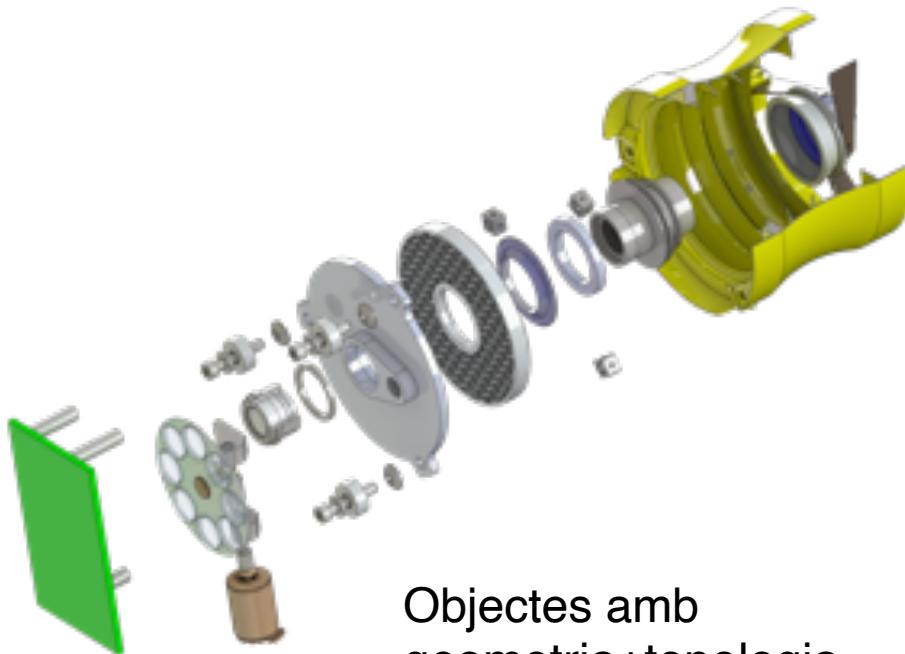
Scene Graph



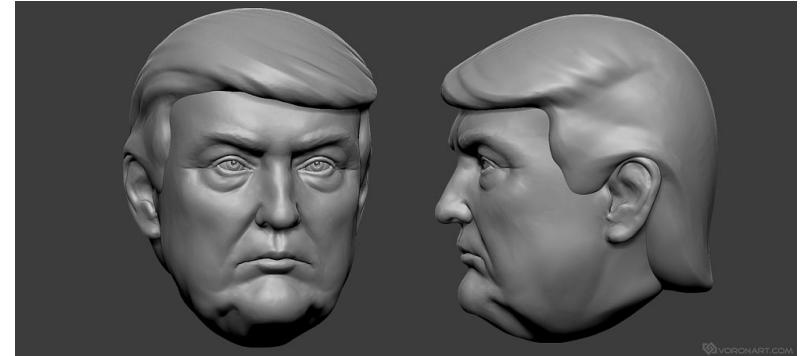
Courtesy of SGI

2.3.1. Interseccions amb l'escena

Llista d'objectes



Objectes amb
geometria+topologia
Materials



Wavefront objects .obj

The "v" identifies this element as a vertex

The "f" identifies this element as a face

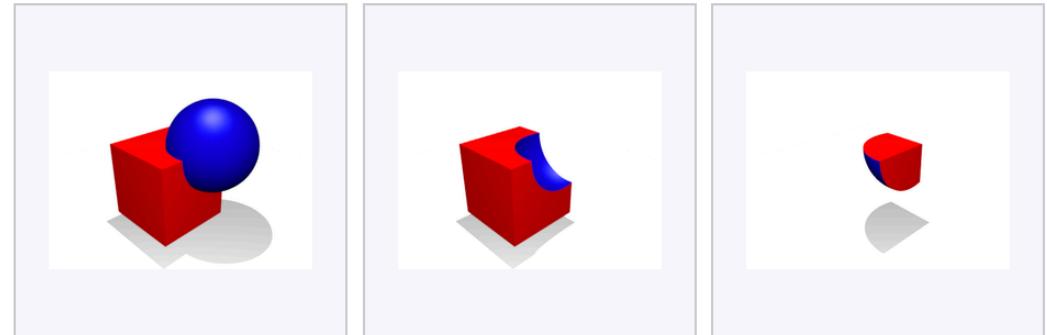
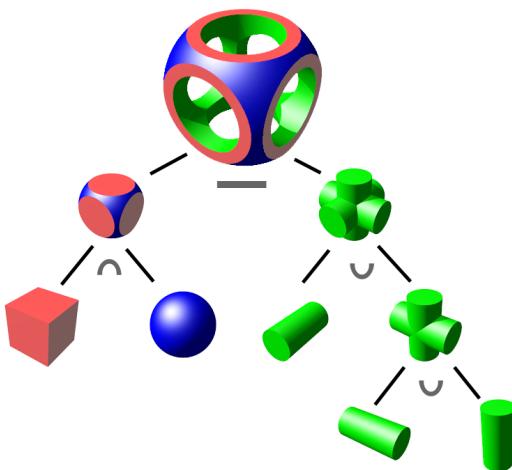
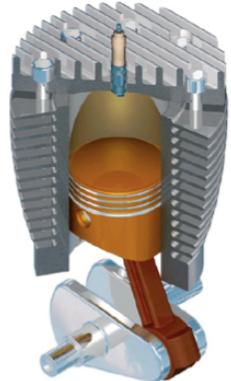
```
v 1.000000 0.000000 0.000000
v 0.000000 1.000000 0.000000
v 0.000000 0.000000 1.000000
v -1.000000 0.000000 0.000000
v 0.000000 -1.000000 0.000000
v 0.000000 0.000000 -1.000000
f 1 2 3
f 2 4 3
f 4 5 3
f 5 1 3
f 2 1 6
f 4 2 6
f 5 4 6
f 1 5 6
```

Vertex coordinates (x,y,z)

Definition of faces from vertex indices (1st vertex is 1)

2.3.1. Interseccions amb l'escena

Arbres CSG



Union

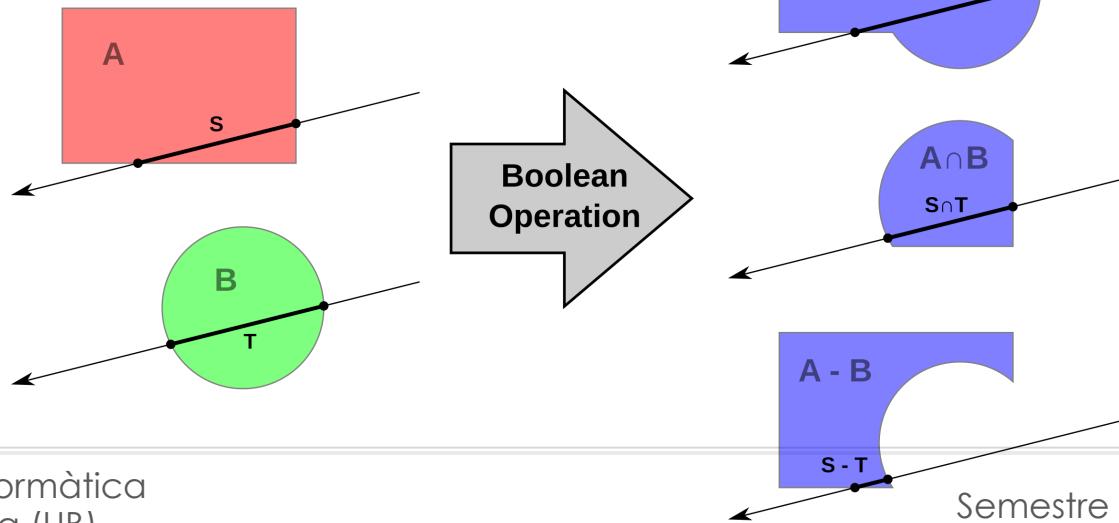
Merger of two objects into one

Difference

Subtraction of one object from another

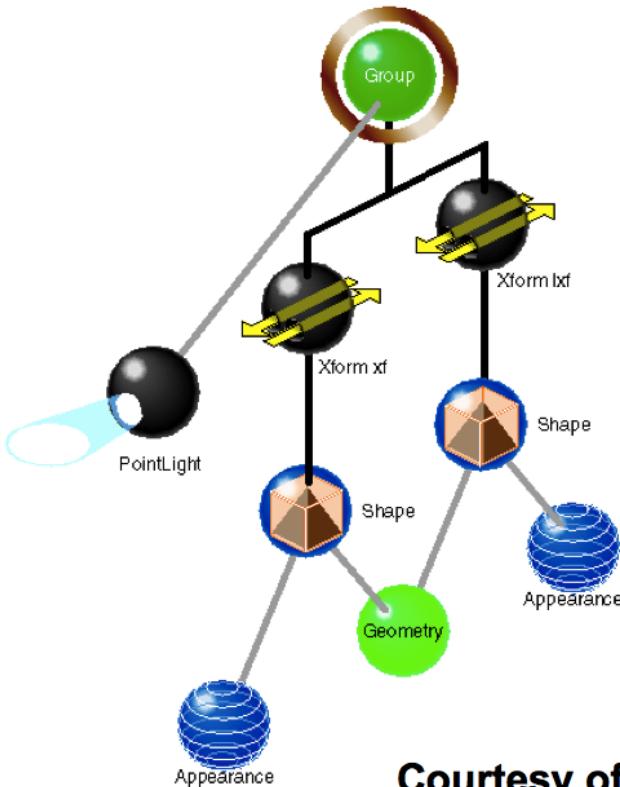
Intersection

Portion common to both objects

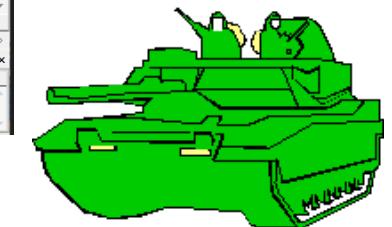
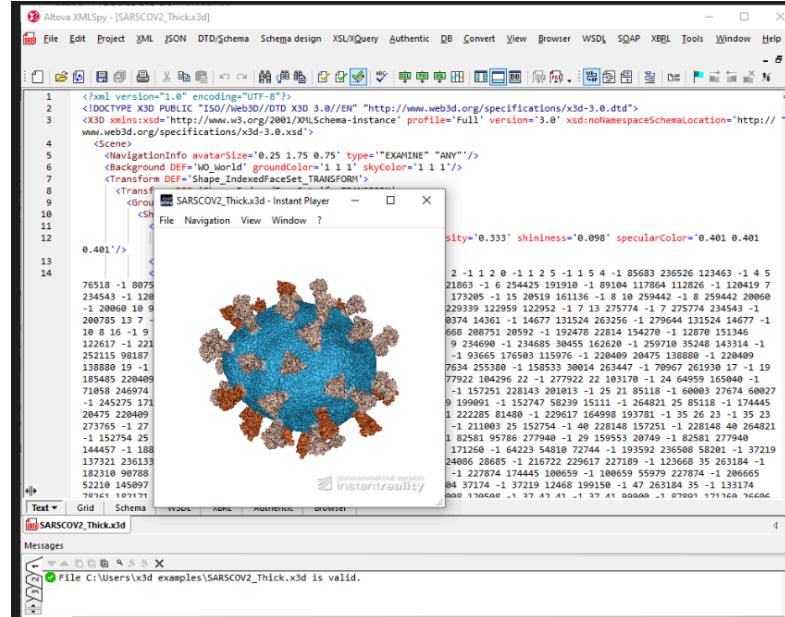


2.3.1. Interseccions amb l'escena

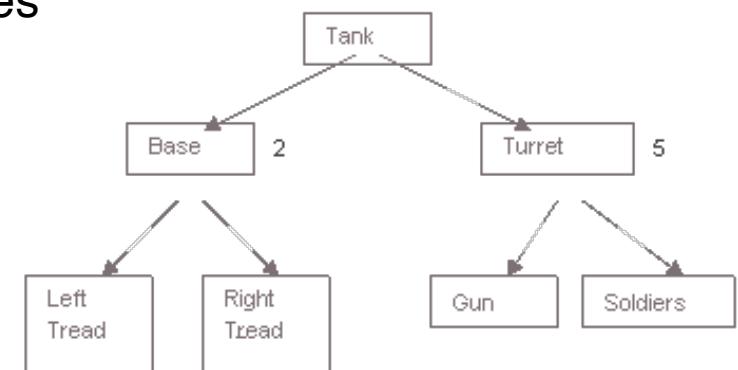
Scene Graph



Courtesy of SGI



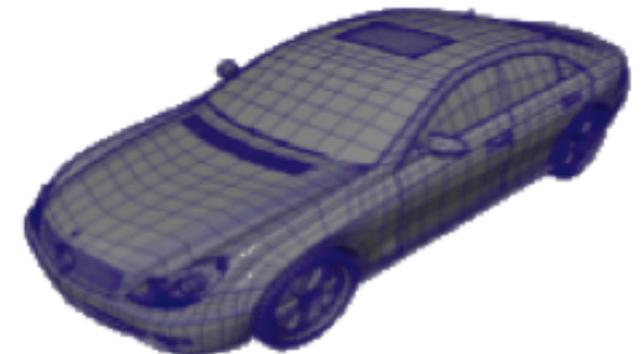
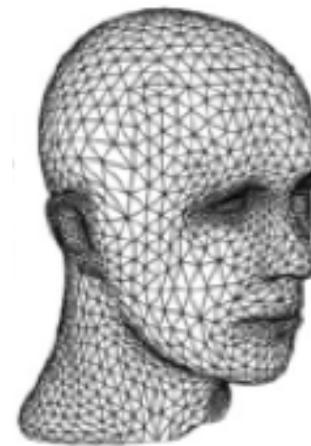
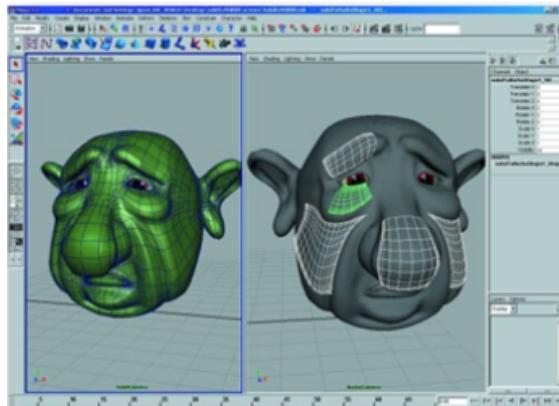
Multiples objectes
TGs
Il·luminació
Shadings
Culling



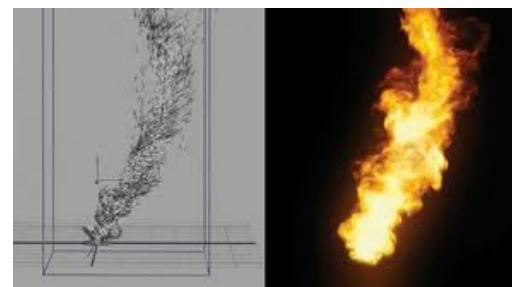
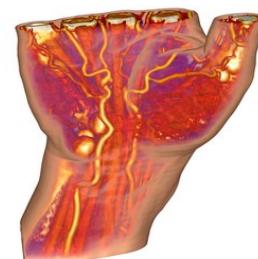
2.3.1. Interseccions amb l'escena

Tipus d'objectes:

- **Objectes superficials:** només representen la frontera de l'objecte amb l'exterior.



- **Objectes volumètrics:** es representa el volum intern dels objectes o característiques de l'espai com temperatura, densitat, etc.)

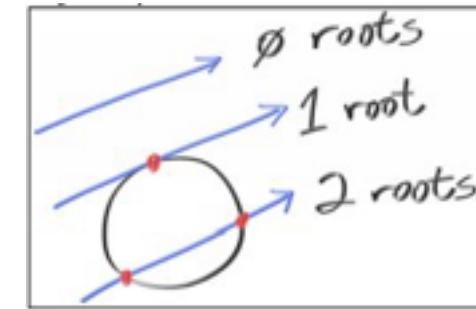


2.3.1. Objectes: com es modelen?

- **Objectes paramètrics:** un objecte en el espai R^3 està representat per un conjunt d'equacions paramètriques

- **Esfera:** definida per el centre C (c_x, c_y, c_z) i el seu radi R

$$(p_x - c_x)^2 + (p_y - c_y)^2 + (p_z - c_z)^2 = R^2$$

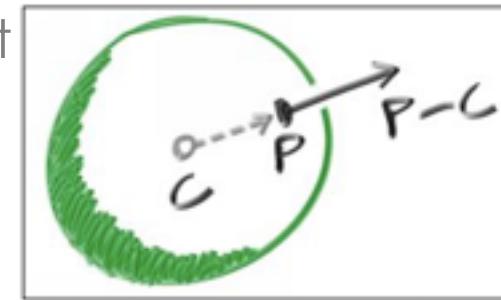


- **Pla:** definit per la seva normal (n_x, n_y, n_z) i un punt de pas que determina d

$$n_x p_x + n_y p_y + n_z p_z + d = 0$$

- Com es calcula la **intersecció** amb el Raig? I la **normal**? Substituïnt en la equació de l'objecte el punt p definit per l'equació paramètrica del Raig

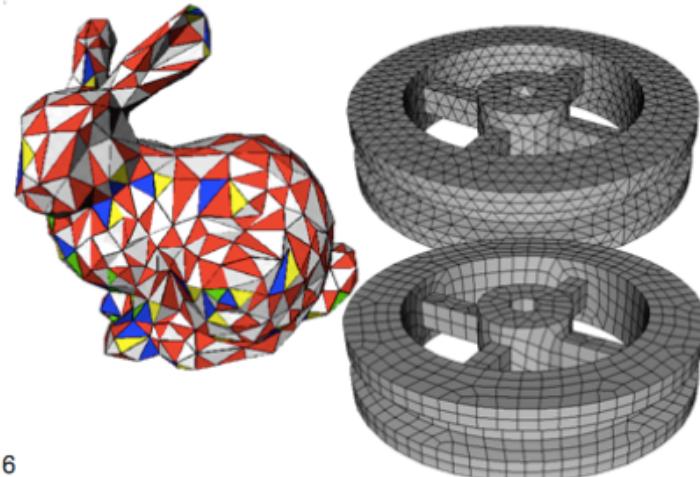
$$p = A + t * \vec{B}$$



2.3.1. Objectes: com es modelen?

- **Malles poligonals**: un objecte en el espai R^3 (poliedre) que està representat per un conjunt (tancat) de polígons
 - Un **polígon** és l'interior d'un conjunt planar tancat i connectat d'arestes lineals.
 - Una **aresta** es defineix pels seus dos punts extrems o vèrtexs
 - Un **vèrtex** és un punt representat per les coordenades x, y, z.

Les malles més utilitzades són malles o *mesh* de triangles o de quadrilàters

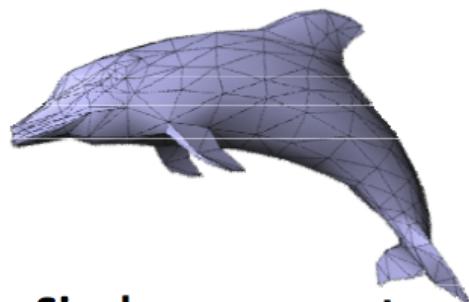


[1] Veure secció 2.4.4 del llibre [Angel2011]

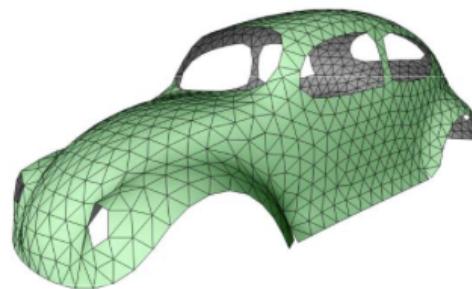
6

2.3.1. Objectes: com es modelen?

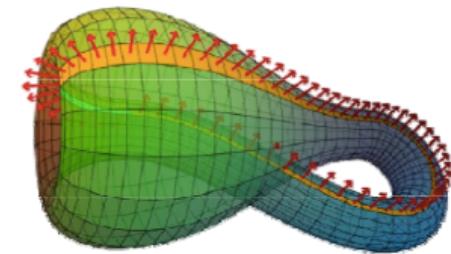
Diferents tipus de malles



**Single component,
closed, triangular,
orientable manifold**

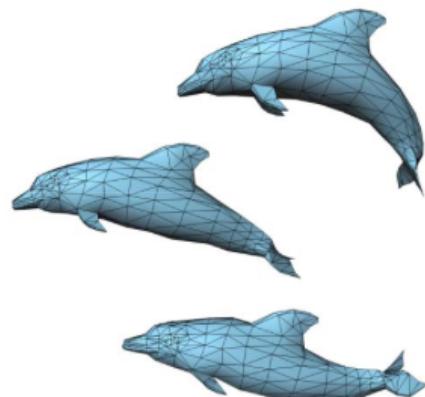


With boundaries

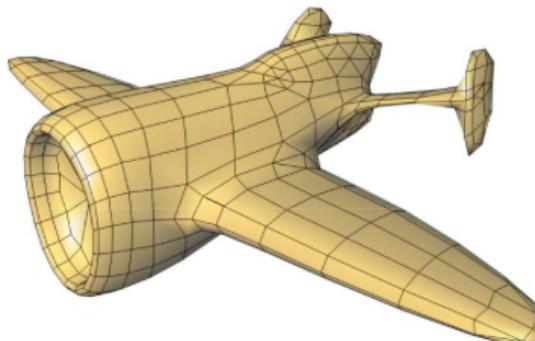


Not orientable

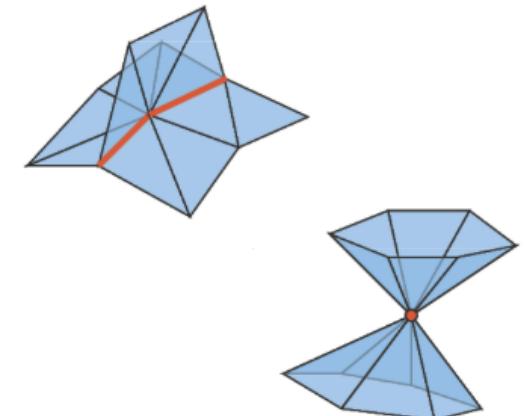
<http://www.kleinbottle.com/>



Multiple components



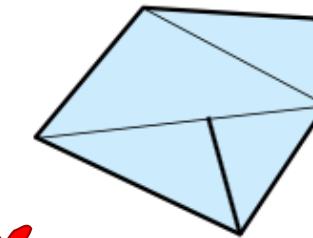
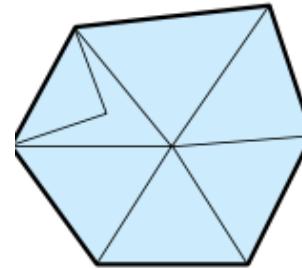
Not only triangles



Non manifold

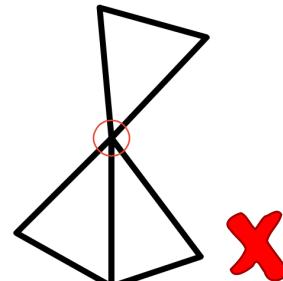
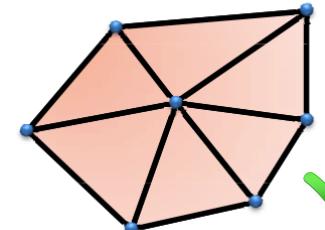
2.3.1. Objectes: com es modelen?

- Una malla poligonal ha de satisfer que:
 - 1) La intersecció de dos polígons qualsevols de la malla ha de ser buida o una aresta comú o un vèrtex comú.

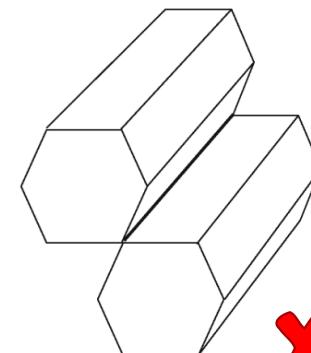
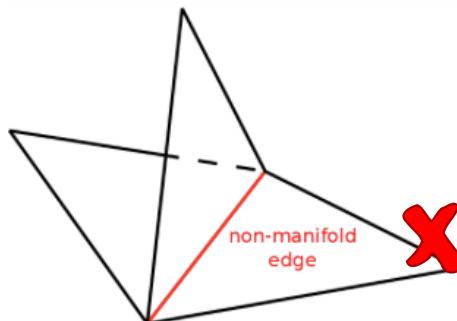


✗

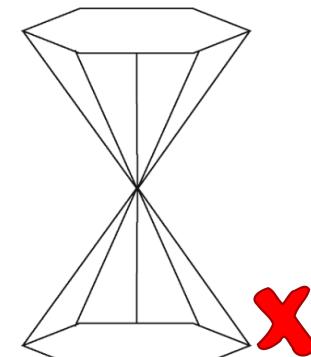
- 2) Només situacions **2-manifold**: cada aresta només es compartida per dues cares, excepte en lesarestes de la frontera. Tots els vèrtexs tenen un conjunt continu de cares al seu voltant (disc continu)



✗



✗



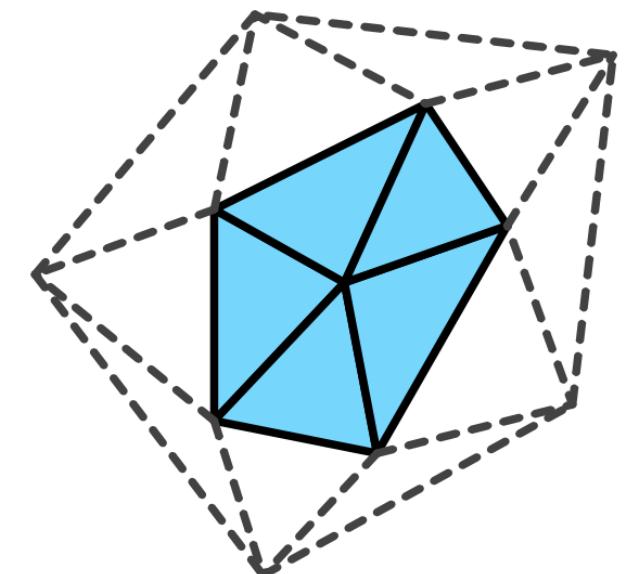
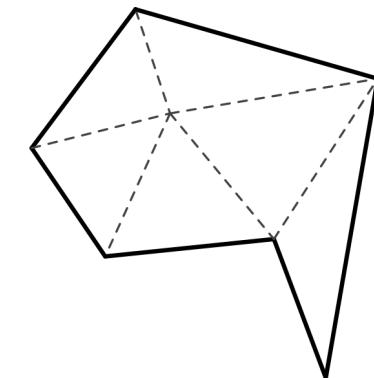
✗

2.3.1. Objectes: com es modelen?

- **Mesh de triangles:** tot polígon es pot subdividir en un conjunt de triangles.

NOTA: Un triangle es defineix amb **tres** vèrtexs i garanteix que només un únic pla pot passar per aquests tres punts.

- Cal que:
 - Totes les arestes pertanyen a 2 triangles
 - Tots els vèrtexs tenen un conjunt continu de triangles al seu voltant



[1] Veure secció 2.4.4 del llibre [Angel2011]

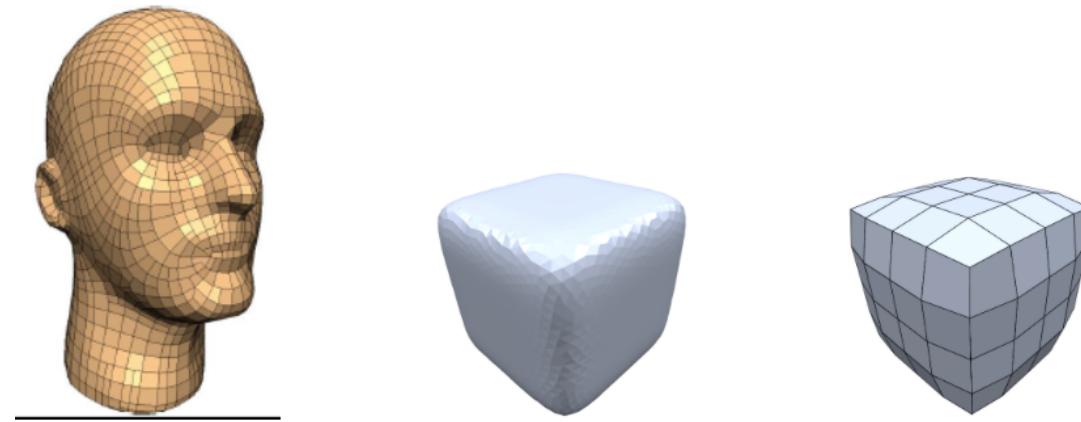
2.3.1. Objectes: com es modelen?

- **Mesh de quadrilàters: els polígons són quadrilàters**

NOTA: Un quadrilàter té el problema que els seus 4 punts no sempre estan en un únic pla

Avantatges:

- Més fàcil d'alignar les arestes a la curvatura
- Més fàcils per aplicar textures
- Més fàcil per a mapejar superfícies paramètriques (Bézier, etc.)

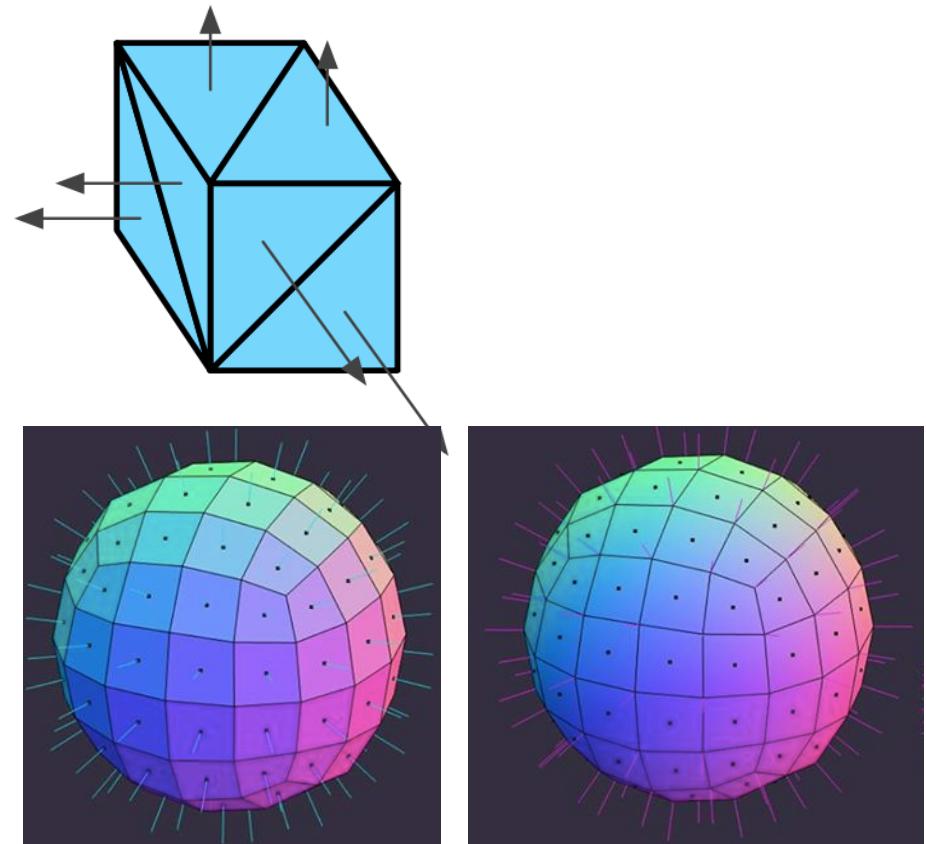
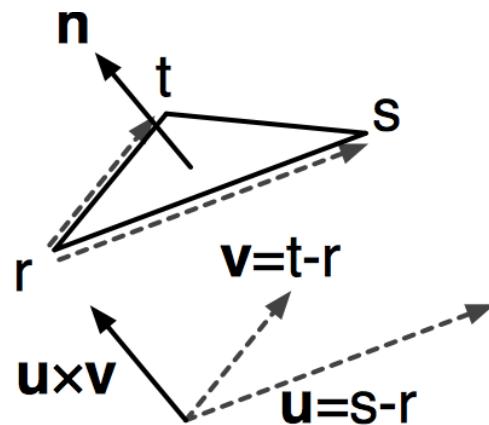


2.3.1. Objectes: normals?

Com es calcula la normal a cada polígon?

Els vèrtexs de cada polígon es guarden ordenats segons sentit anti-horari de forma que la normal obtinguda segons l'ordre definit apunti sempre cap a l'exterior del poliedre.

$$\mathbf{n} = (\mathbf{s} - \mathbf{r}) \times (\mathbf{t} - \mathbf{r})$$



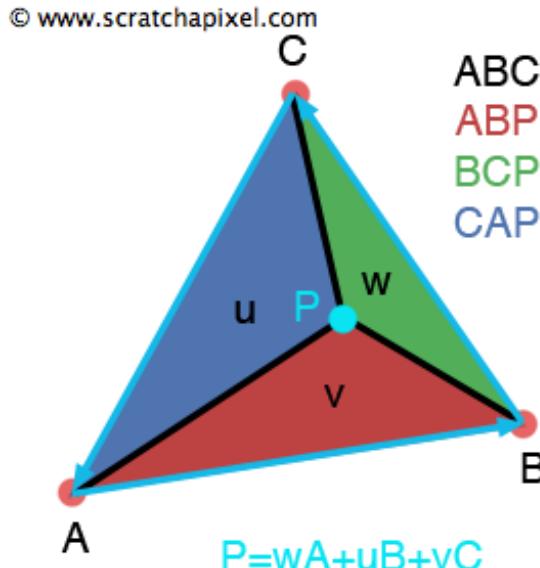
2.3.1. Triangle: intersecció?

Com es calcula la intersecció del raig amb cada triangle?

1. Càlcul de la intersecció amb el pla del triangle:

- Pla amb normal n
- Pla que passa per un vèrtex del triangle

2. Comprovació si el punt és interior al triangle: usant



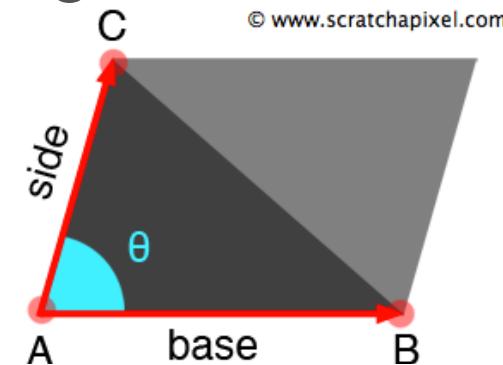
$$u = \frac{\text{TriangleCAP Area}}{\text{TriangleABC Area}}$$

$$v = \frac{\text{TriangleABP Area}}{\text{TriangleABC Area}}$$

$$w = \frac{\text{TriangleBCP Area}}{\text{TriangleABC Area}}$$

$$\boxed{P = wA + uB + vC}$$

on $u+v+w = 1$ i $0 < u, v, w < 1$



$$\text{Triangle}_{area} = \frac{\|(B - A)\| * \|(C - A)\| \sin(\theta)}{2}$$

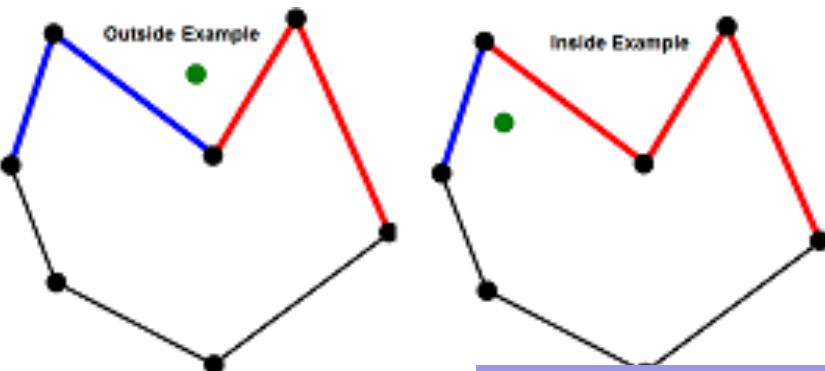
2.3.1. Objectes: intersecció?

Com es calcula la intersecció del raig amb cada polígon?

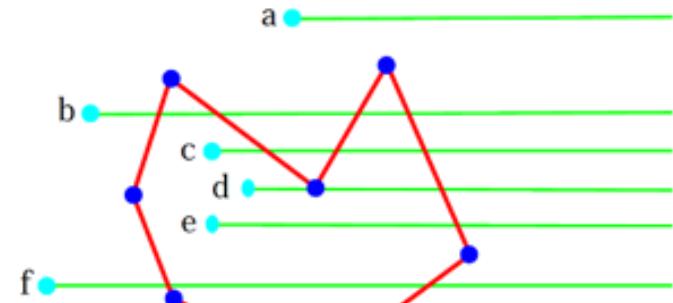
1. Càlcul de la intersecció amb el pla del polígon:

- Pla amb normal n
- Pla que passa per un vèrtex del polígon

2. Comprovació si el punt és interior al polígon



Número imparell d'interseccions implica punt interior
"Si la intersecció és un vèrtex del polígon només es conta com intersecció si l'altre vèrtex de l'aresta està per sota del raig"



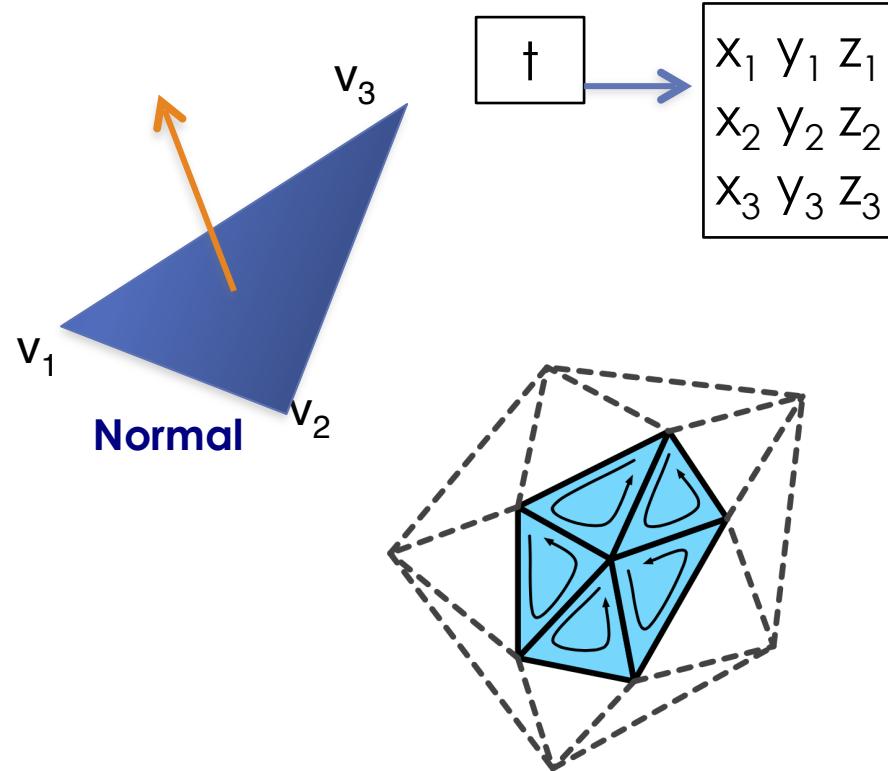
2.3.1. Objectes: com es representen?

Representacions de mallas poligonals:

1. explícita
2. vèrtex indexats
3. adjacència de cares
4. winged edge

2.3.1. Objectes: com es representen?

- Representació de mesh de polígons: **Representació Explícita** (fitxers **STL**)
 - Llista de vèrtexs per a cada polígon*



Per n triangles: $3*3*n$ floats

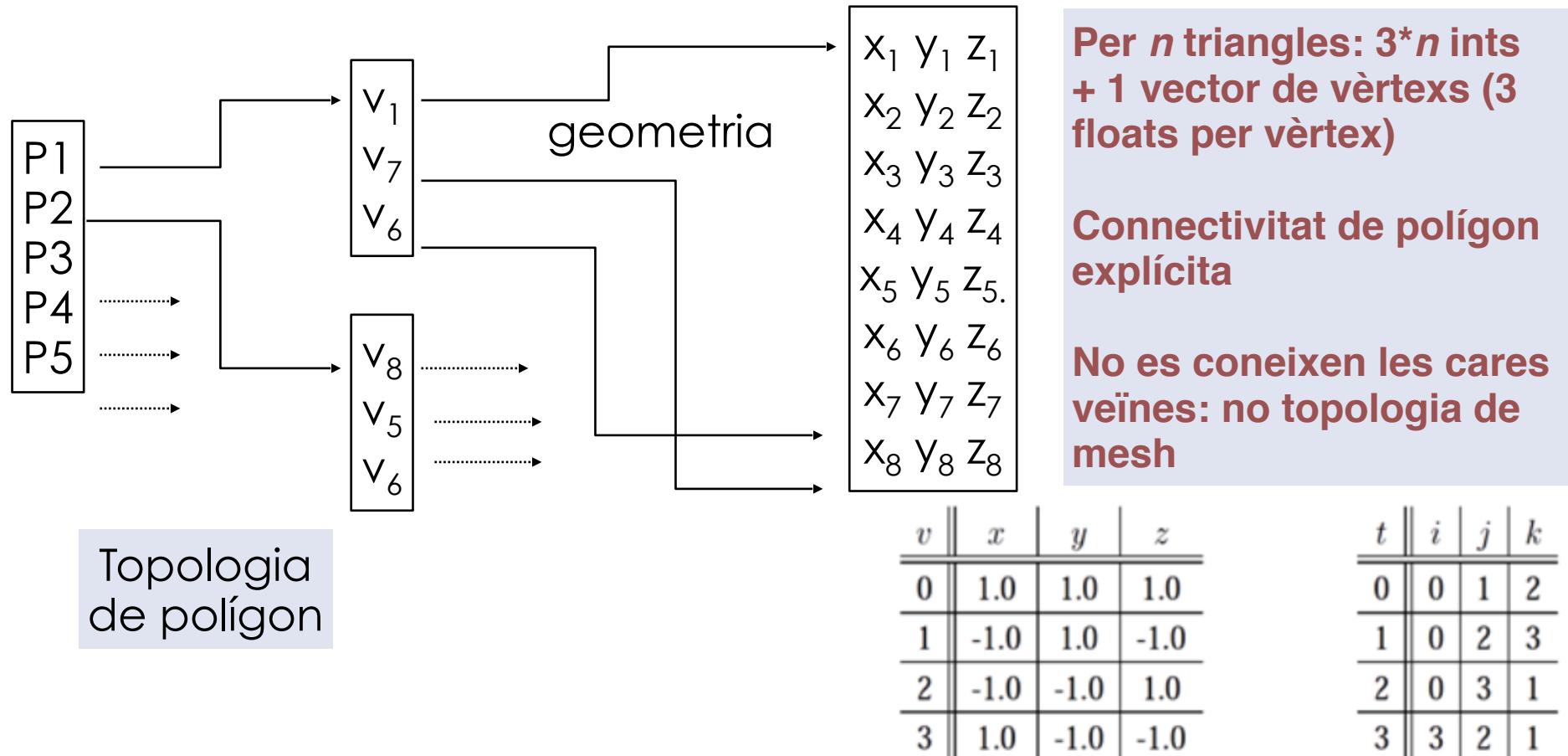
No hi ha connectivitat guardada explícitament

v	x	y	z
0	1.0	1.0	1.0
1	-1.0	1.0	-1.0
2	-1.0	-1.0	-1.0
3	1.0	1.0	1.0
4	-1.0	-1.0	-1.0
5	1.0	-1.0	-1.0

Ordenats en sentit anti-horari 3 a 3

2.3.1. Objectes: com es representen?

- Representació de mesh de polígons: **Llistes de polígons + vèrtexs: Vèrtexs indexats** (fitxers **OBJ**, **OFF**)



2.3.1. Objectes: com es representen?

- Representació de mesh de polígons: **adjacència de cares**:
 - Llista de vèrtexs:**
 - les seves coordenades
 - Index a una de les cares a la que pertany
 - Llista de cares:**
 - Indexes als punts
 - Indexes a les cares adjacents

Per n triangles:
 $3*n_vertexs$ floats + 1
 $*n_vertexs$ enter + $6*n$ enters

Connectivitat de mesh i de polígon explícites

v	x	y	z	l
0	1.0	1.0	1.0	0
1	-1.0	1.0	-1.0	0
2	-1.0	-1.0	1.0	0
3	1.0	-1.0	-1.0	1

t	i	j	k	n_0	n_1	n_2
0	0	1	2	3	1	2
1	0	2	3	3	2	0
2	0	3	1	3	0	1
3	3	2	1	0	2	1

Topologia de malla

2.3.1. Objectes: com es representen?

- Representació de mesh de polígons: Llistes de **cares+arestes+vèrtexs**: **model Winged-edge** (optimització)

<https://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/model/winged-e.html>

- Llista d'arestes:**

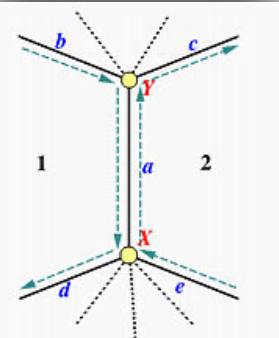
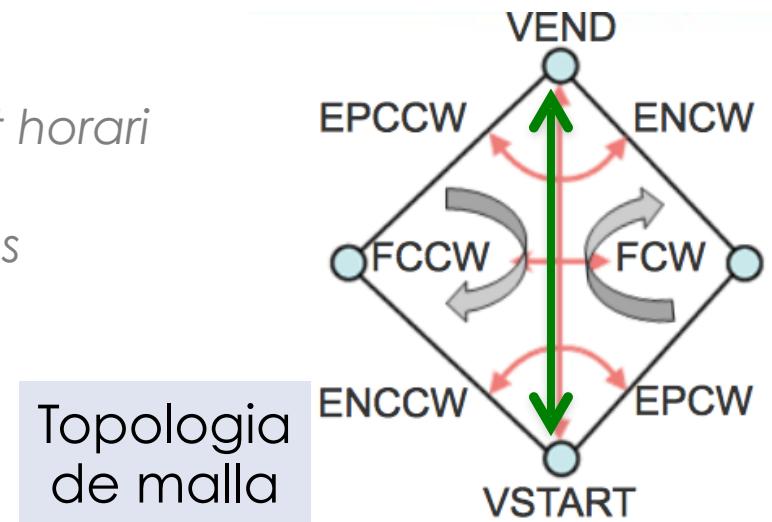
- Vèrtex inicial (VSTART) i final (VEND) - sentit horari
- Les dues cares adjacents (FCW i FCCW)
- Arestes anterior i posterior de les dues cares (EPCW, ENCW, EPCCW, ENCCW)

- Llista de vèrtexs:**

- Posició
- Una aresta adjacent

- Llista de cares:**

- Una aresta adjacent



Edge	Vertices	Faces	Left Traverse	Right Traverse				
Name	Start	End	Left	Right	Pred	Succ	Pred	Succ
a	X	Y	1	2	b	d	e	c

