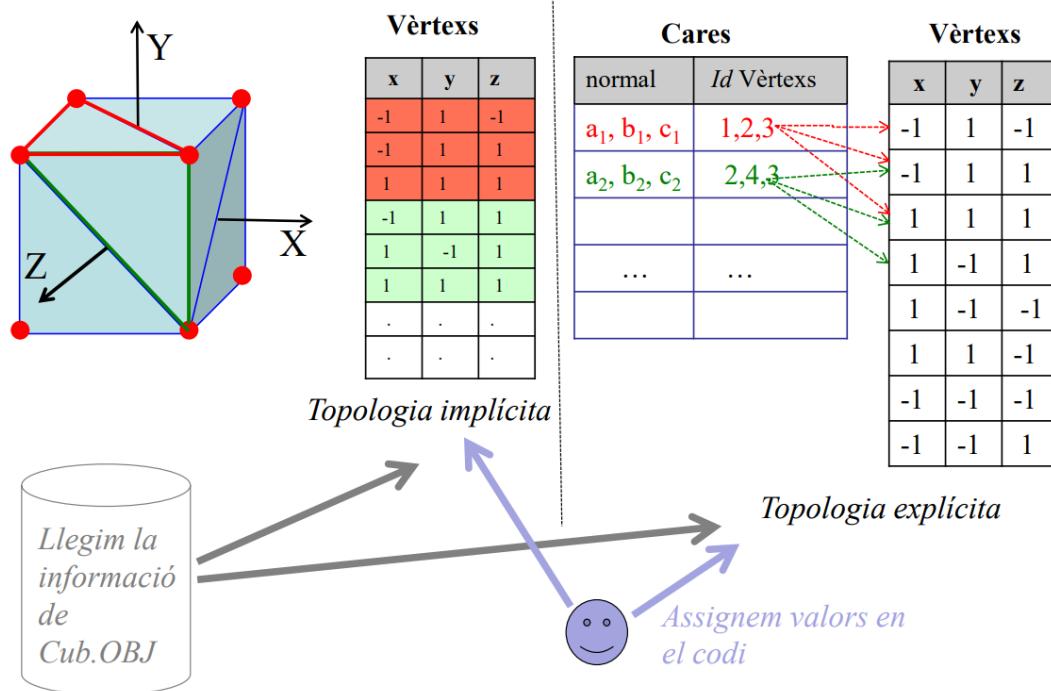


# Model Fronteres: conjunt de triangles

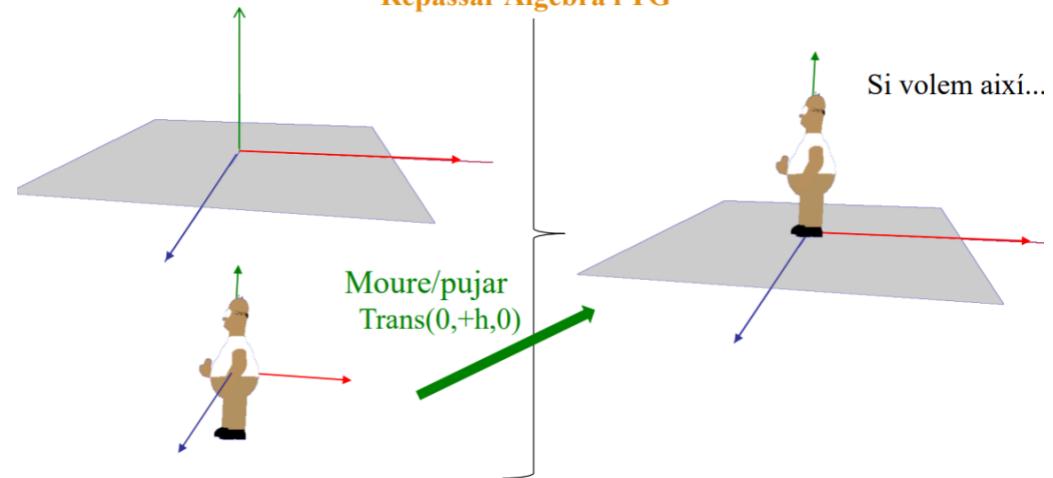


## Pintar en OpenGL 3.3: “core” mode

1. Crear en GPU/OpenGL un *VAO* que encapsularà dades del model. Crear *VBO* que guardarà les coordenades dels vèrtexs (i si cal, normal, color,...)
2. Guardar la llista de vèrtexs en el *VBO* (i si cal, color i normal en els seus *VBO*)
3. Cada cop que es requereix pintar, indicar el *VAO* a pintar i dir que es pinta: `glDrawArrays(...)`. Acció **pinta\_model()** a teoria.



## Repassar Àlgebra i TG



- Cal aplicar TG que modifica coordenades vèrtexs
- TG queda definida per matriu 4x4:

$$\mathbf{V}_A = \text{TG } \mathbf{V}_m = T(0, +h, 0) \mathbf{V}_m$$

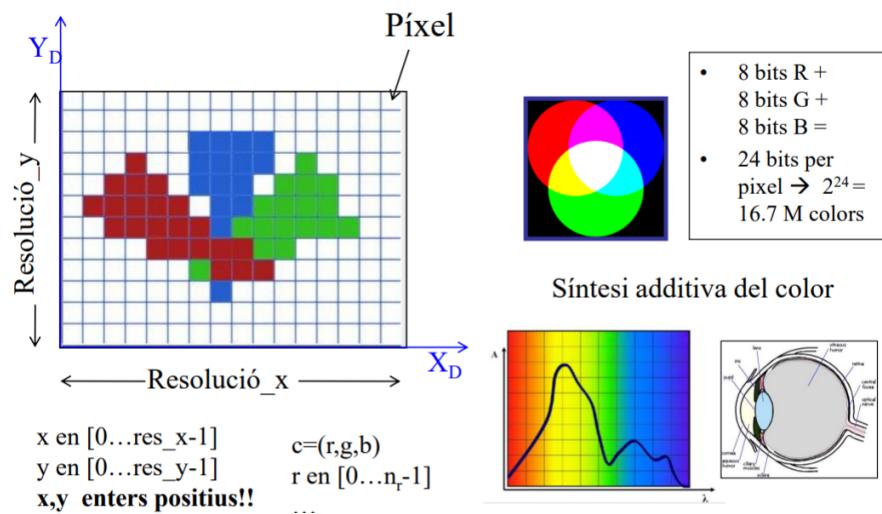
$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Sobre processament de vèrtexs:

- Important diferenciar SCA (sistema coordenades aplicació / món / escena) de SCO (sistemes de coordenades de l'observador).<sup>1</sup>
- La posició de l'OBS no depèn del tipus d'òptica (obvi).
- Visualització: posició + orientació, òptica, fer la foto, emmarcar.
- El viewport és tota la finestra d'OpenGL.

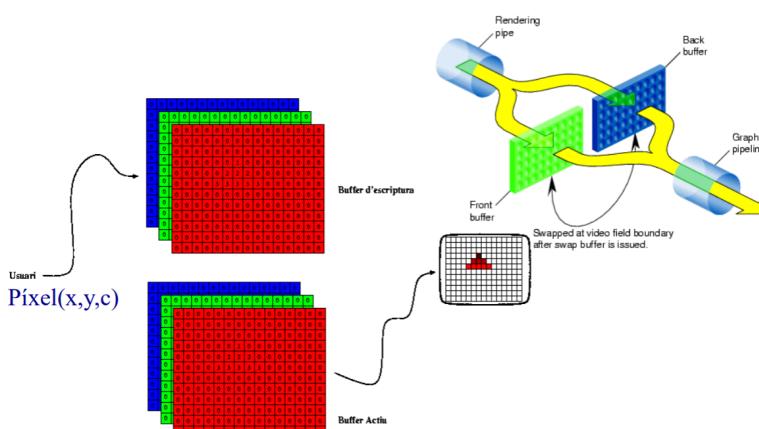
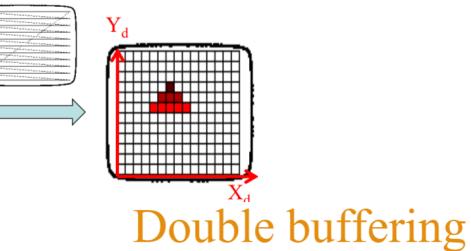
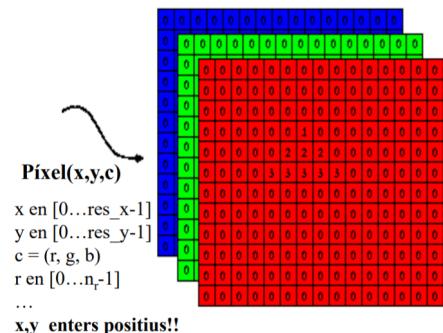
<sup>1</sup> Un Patricio centrat a l'escena està a la posició (0, 0, 0) en SCA (perquè *està* centrat en l'escena); però, si mirem el Patricio des de davant, segurament el centre del Patricio serà a la coordenada (0, 0, -2R), perquè, des de la càmera (observador), ens hem de desplaçar 2 vegades el radi de l'escena per aplegar al punt mitjà del Patricio.

# Pantalles d'escombrat/raster

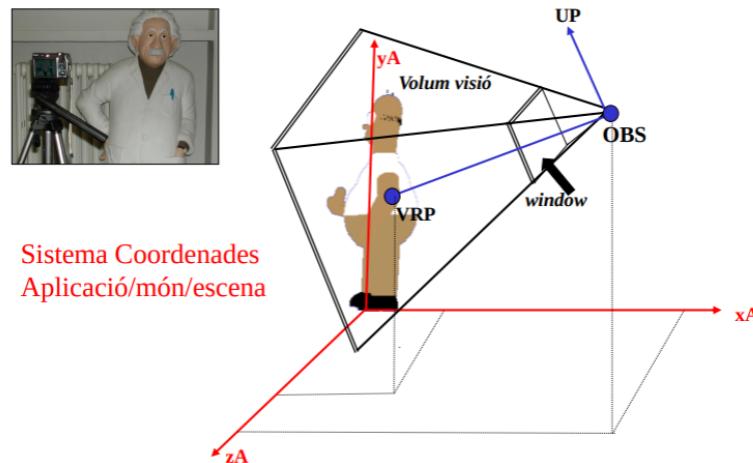


## Frame buffer

fb és taula  $[\text{res\_x}][\text{res\_y}]$  de color  
 $fb[x][y] = c$

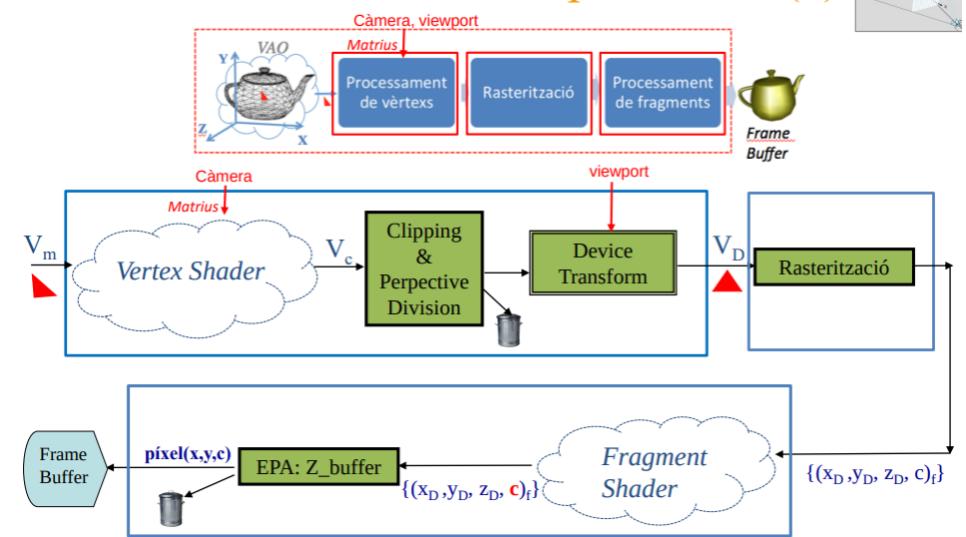


# Com indicar la càmera?



1. Ubicació respecte SCA: obs, vrp, up
  2. Definir el Volum de Visió: òptica (window, zNear, zFar)
- Per visualitzar:
    - Fem un cop: per cada model, crear i omplir el seu VAO.
    - Cada cop que refresquem finestra: per cada model m, calculem la seva TG (transformacions geomètriques), indiquem a OpenGL que la volem usar (modelMatrix(TG)) i pintar\_model().
    - Cura, perquè apliquem la TG *abans* de tot!! Per tant, donat un vèrtex de model V, el passem a coordenades de clipping així<sup>2</sup>:  $VC = PM VM TG V$ .

## Pintar/visualitzar en OpenGL 3.3 (2)

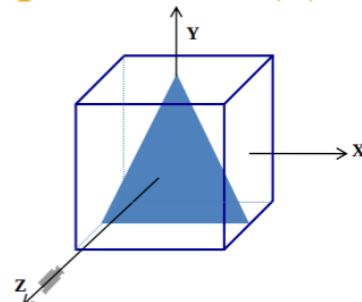


## Pintar/visualitzar en OpenGL 3.3 (3)

### Vertex Shader

```
#version 330 core
in vec3 vertex;

void main() {
    gl_Position = vec4 (vertex, 1.0);
}
```

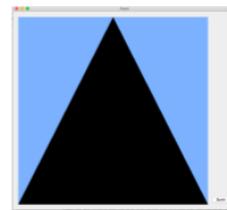


Volum de Visió cub de (-1,-1,-1) a (1,1,1)

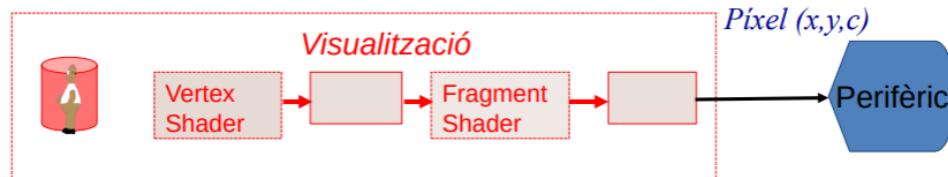
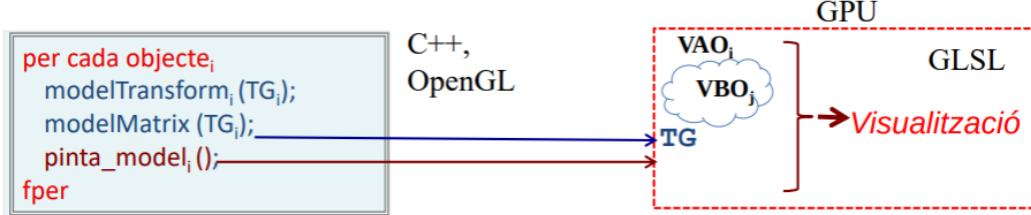
### Fragment Shader

```
#version 330 core
out vec4 FragColor;

void main() {
    FragColor = vec4(0, 0, 0, 1);
}
```



## Pintar en OpenGL 3.3: “core” mode

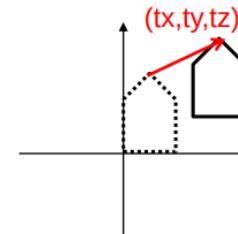


```
#version 330 core
in vec3 vertex;
uniform mat4 TG;

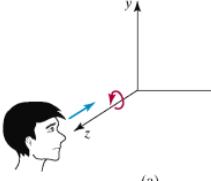
void main() {
    gl_Position = TG * vec4(vertex, 1.0);
}
```

## Transformacions Geomètriques

Transformació geomètrica



$$x' = x+tx; y'=y+ty; z'=z+tz$$



$T(tx,ty,tz)$

$$\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$R_z(\text{angle})$

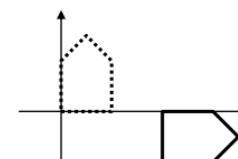
$$\begin{bmatrix} \cos \alpha & -\sin \alpha & 0 & 0 \\ \sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$S(s_x,s_y,s_z)$

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## Composició de Transformacions

- Imaginem que volem



No es pot fer amb cap de les matrius anteriors

- Cal compondre/efectuar dues transformacions

$$\mathbf{P}' = R_z(-90^\circ) \cdot \mathbf{P}$$

$$\mathbf{P}'' = T(3,0,0) \cdot \mathbf{P}'$$

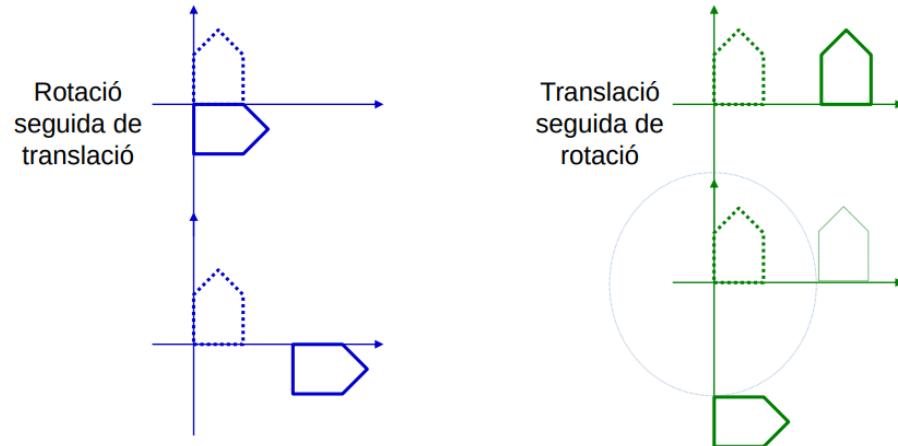
$$\mathbf{P}'' = T(3,0,0) \cdot (R_z(-90^\circ) \cdot \mathbf{P}) = (T(3,0,0) \cdot R_z(-90^\circ)) \cdot \mathbf{P} = \mathbf{M} \cdot \mathbf{P}$$

## Composició de Transformacions

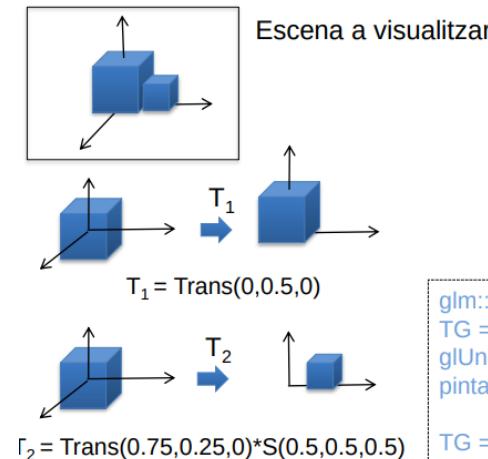
$$\mathbf{T}(3,0) \cdot \mathbf{R}(-90^\circ) \neq \mathbf{R}(-90^\circ) \cdot \mathbf{T}(3,0)$$

②      ①      ②      ①

- Multiplicació de matrius no és commutativa



### Exemple simple de TG (1)



Exemple codi per pintar en CPU

Pseudo-codi

```

TG = Translate (0, 0.5, 0);
modelMatrix (TG);
pinta_cub ();
TG = Translate (0.75, 0.25, 0);
TG = TG*Scale (0.5, 0.5, 0.5);
modelMatrix (TG);
pinta_cub();

```

```

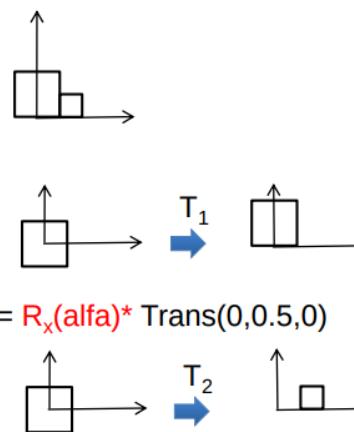
glm::mat4 TG = glm::mat4(1.f);
TG = glm::translate (TG, glm::vec3(0,0,0.5));
glUniformMatrix4fv (transLoc, 1, GL_FALSE, &TG[0][0]);
pinta_cub ();

TG = glm::mat4(1.f);
TG = glm::translate (TG, glm::vec3(0.75,0.25,0));
TG = glm::scale(TG, glm::vec3(0.5,0.5,0.5));
glUniformMatrix4fv (transLoc, 1, GL_FALSE, &TG[0][0]);
pinta_cub();

```

Com faríeu per girar tota l'escena alfa graus respecte l'eix x?

### Exemple simple (2): Girar escena repetit eix X alfa graus



$$T_1 = R_x(\alpha) * \text{Trans}(0,0,0.5)$$

$$T_2 = R_x(\alpha) * \text{Trans}(0.75,0.25,0) * S(0.5,0.5,0.5)$$

```

AUX = Rotate (alfa, (1,0,0));
TG = AUX * Translate(0, 0.5, 0);
modelMatrix (TG);
pinta_cub ();

```

```

TG = AUX * Translate(0.75, 0.25, 0);
TG = TG * Scale(0.5,0.5,0.5);
modelMatrix (TG);
pinta_cub();

```

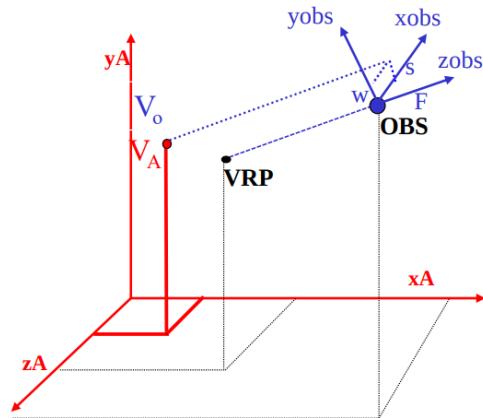
Sobre els tres tipus de transformacions:

- Translació:  $\text{Translate}(tx, ty, tz)$ .
- Escalat:  $\text{Scale}(sx, sy, sz)$ . Normalment  $sy = sz$  per no deformar.
- Rotació: a teoria:  $\text{Rotatex}(\varphi_x)$ ,  $\text{Rotatey}(\varphi_y)$ ,  $\text{Rotatez}(\varphi_z)$ , tot i que també es fa servir  $\text{Rotate}(\varphi, (x, y, z))$ , on  $(x, y, z) = (0, 0, 0)$  excepte un dels tres que és 1.
- Si volem fer translació + rotació, la matriu a aplicar és  $R \cdot T$ . Importa l'ordre!

Sobre rotacions i escalats:

- Un cop estem en l'origen (hem fet Translació( $-t$ ), on  $t$  és el centre/base de la capsella mínima contenidora), podem aplicar rotació + escalat o escalat + rotació, no importa!

## Pas a SCO amb VRP, OBS i Up: Càcul View Matrix



```
VM = lookAt(OBS, VRP, Up);
viewMatrix(VM);
```

$$VM = \begin{bmatrix} s.x & s.y & s.z & 0 \\ w.x & w.y & w.z & 0 \\ F.x & F.y & F.z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & -OBS.x \\ 0 & 1 & 0 & -OBS.y \\ 0 & 0 & 1 & -OBS.z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$F = OBS - VRP = (F.x, F.y, F.z) \quad F = F / \|F\|$$

$$s = Up \times F \quad s = s / \|s\|$$

$$w = F \times s$$

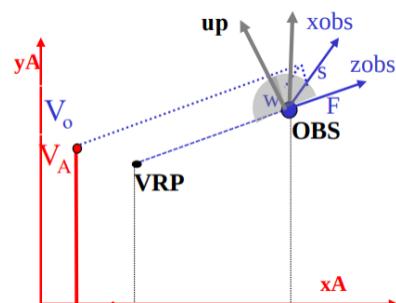
viewMatrix

$$\downarrow$$

View Transform

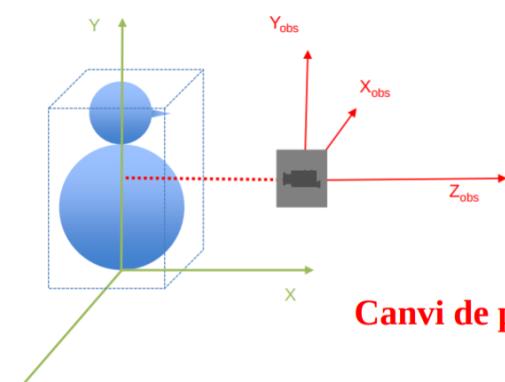
$$V_o = VM * V_A$$

$$V_o = (x_o, y_o, z_o, 1)$$



**OBS** = Observador  
**VRP** = View Reference Point  
**up** = View Up Vector

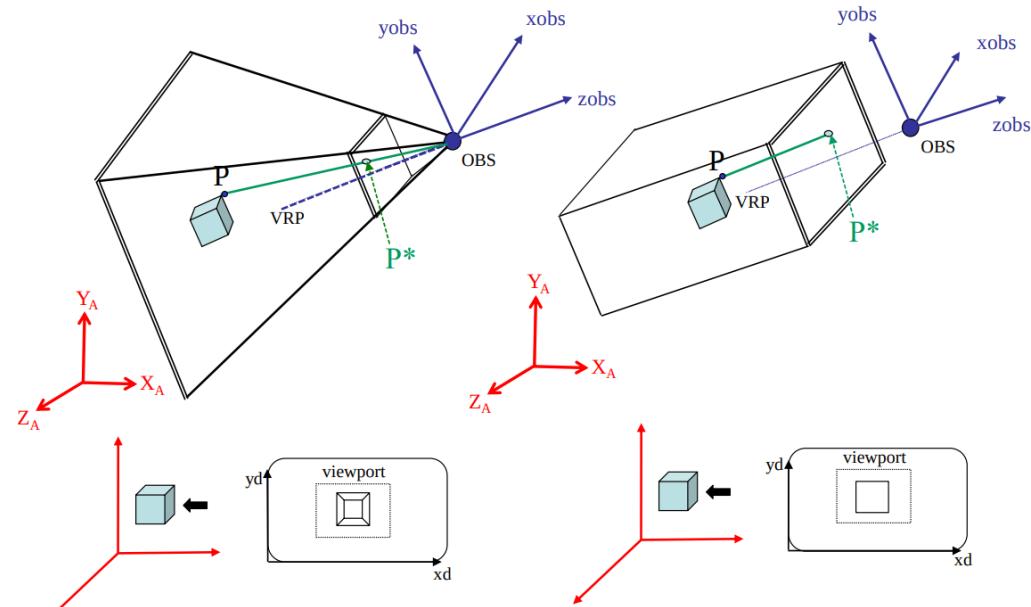
**up** "indica" la direcció de l'eix vertical de la Càmera



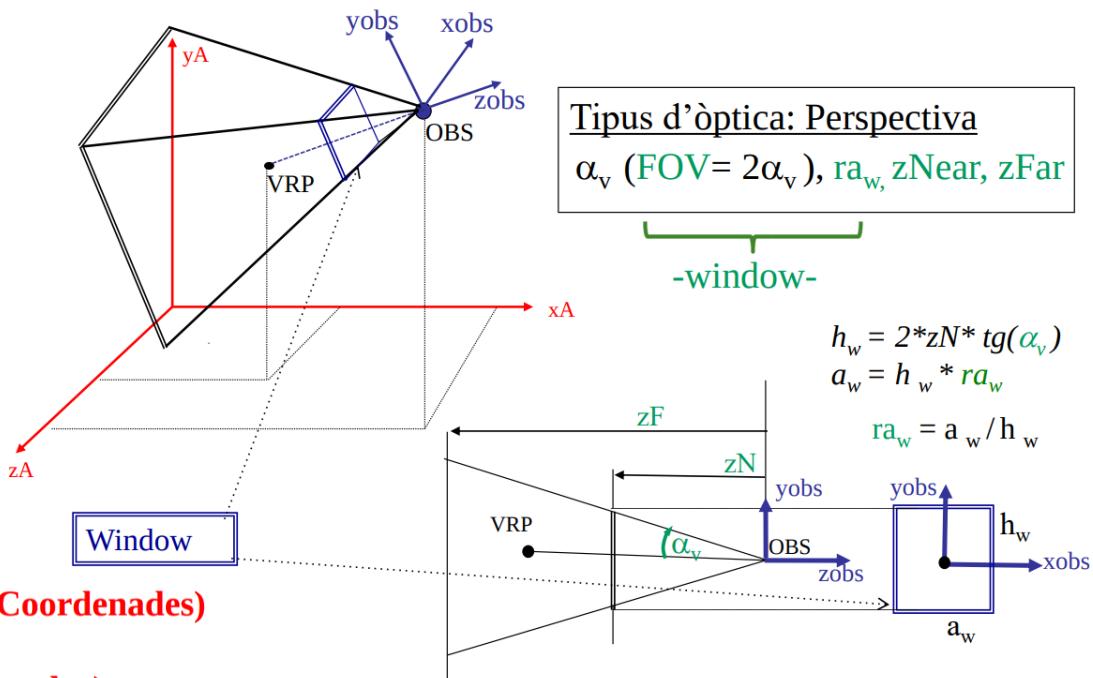
Canvi de punt de vista (canvi de Sistema de Coordenades)

Pas de SCA (Aplicació) a SCO (Observador)

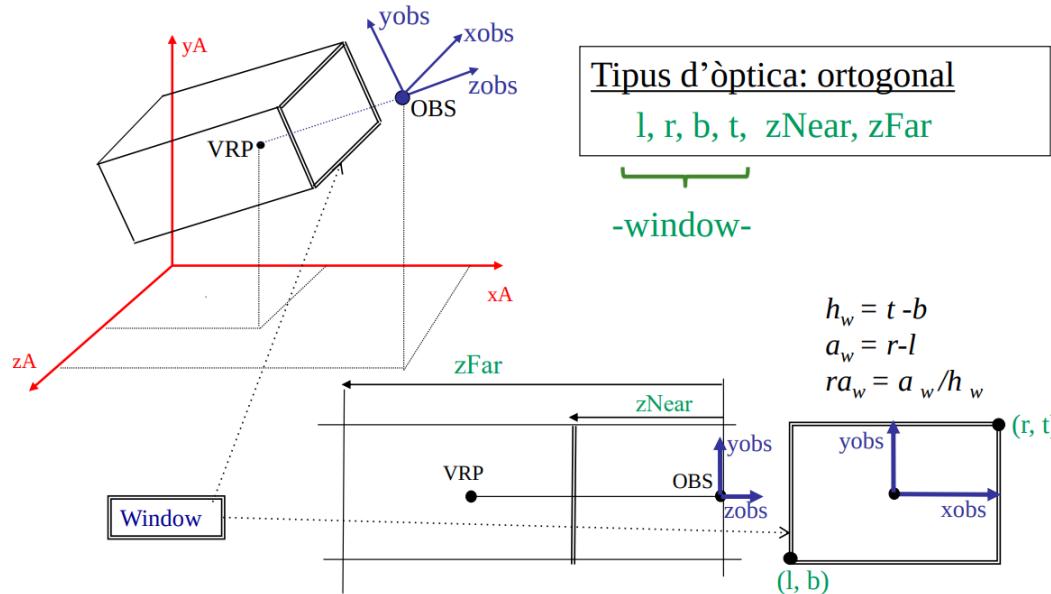
## Projecció: Perspectiva o Ortogonal



### Optica perspectiva: paràmetres



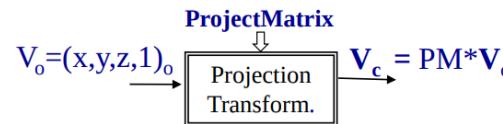
# Òptica ortogonal: paràmetres



## Càcul matriu Project Matrix

$$PM = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

a=2/(r-l) b=2/(t-b)  
c=2/(zf-zn)  
d=(zn+zf)/(zf-zn)



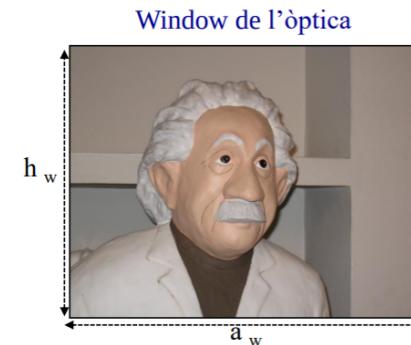
### Òptica Ortogonal

$V_c = (x_c, y_c, z_c, w_c)$  on  $w_c = 1$   
 $PM = \text{ortho}(l, r, b, t, zN, ZF);$   
 $\text{projectMatrix}(PM);$

### Òptica Perspectiva

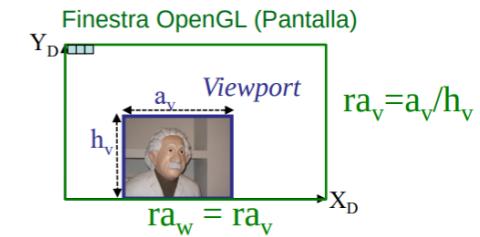
$V_c = (x_c, y_c, z_c, w_c)$  on  $w_c = -z_0$   
 $PM = \text{perspective}(FOV, ra, zN, ZF);$   
 $\text{projectMatrix}(PM);$

# Sobre la relació d'aspecte del window i del viewport

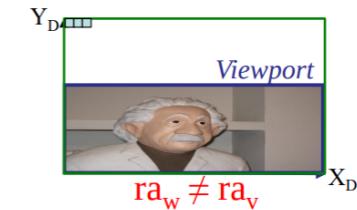


Per a no tenir deformació en la imatge

$$ra_w = ra_v$$

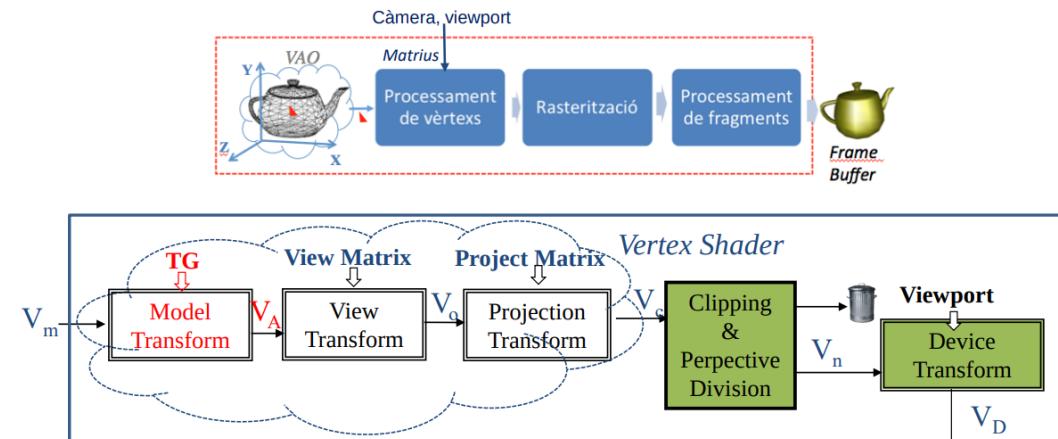


glViewport (ox, oy, av, hv)



$$ra_w \neq ra_v$$

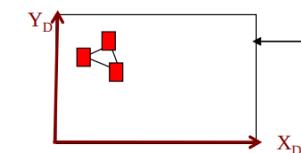
## Vertex Shader – afegim TG



$$V_A = \text{TG} * V_m$$

$$V_O = VM * V_A = VM * \text{TG} * V_m$$

$$V_C = PM * V_O = PM * VM * \text{TG} * V_m$$



# Programació bàsica del vertex shader

#version 330 core

```
in vec3 vertex;
uniform mat4 PM;
uniform mat4 VM;
uniform mat4 TG;

void main() {
    gl_Position = PM*VM*TG*vec4 (vertex, 1.0);
}
```

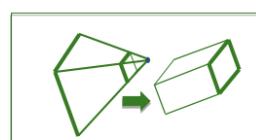
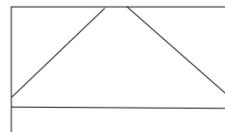
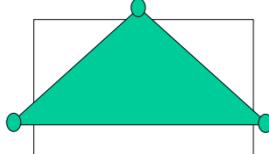
## Vertex Shader

## Clipping

Condició per a que un Vèrtex sigui interior al volum de visió:

$$\begin{aligned} -w_c &\leq x_c \leq w_c \\ -w_c &\leq y_c \leq w_c \\ -w_c &\leq z_c \leq w_c \end{aligned}$$

$V_c = (x_c, y_c, z_c, w_c)$  on  $w_c=1$  en ortogonal  
 $V_c = (x_c, y_c, z_c, w_c)$  on  $w_c=-z_o$  en perspectiva



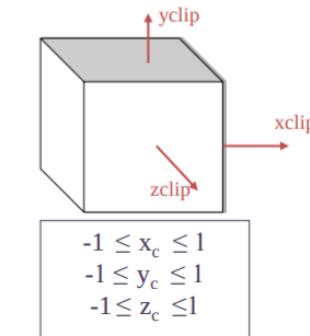
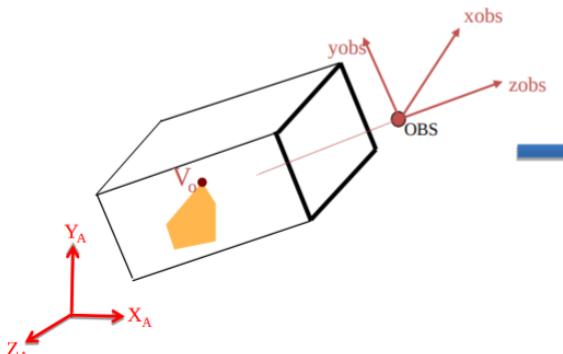
$$V^* = V_c / w_c \rightarrow V_n$$

$$\begin{aligned} -1 \leq x_n &\leq 1 \\ -1 \leq y_n &\leq 1 \\ -1 \leq z_n &\leq 1 \end{aligned}$$

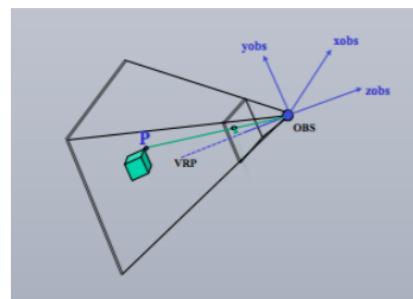
## Projecció: Ortogonal

$$PM = \begin{pmatrix} a & 0 & 0 & e \\ 0 & b & 0 & f \\ 0 & 0 & c & d \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{aligned} a &= 2/(r-l) & e &= (r+l)/(r-l) \\ b &= 2/(t-b) & f &= (t+b)/(t-b) \\ c &= 2/(zf-zn) & d &= (zn+zf)/(zf-zn) \end{aligned}$$

$$V_c = (x_c, y_c, z_c, w_c) \text{ on } w_c=1$$



## Projecció: Perspectiva



$$\begin{aligned} \text{Vèrtex projectat:} \\ V^* &= V_c / w_c = -V_d / z_o \\ x^* &= -x_c / z_o \quad y^* = -y_c / z_o \quad z^* = -z_c / z_o \end{aligned}$$

Inversament proporcional a distància a observador 😊



$$PM = \begin{pmatrix} 1/r_a * a & 0 & 0 & 0 \\ 0 & 1/a & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad \begin{aligned} a &= \tan(\text{FOV}/2) \\ c &= (zf+zn)/(zn-zf) \\ d &= 2*zn * zf / (zn-zf) \end{aligned}$$

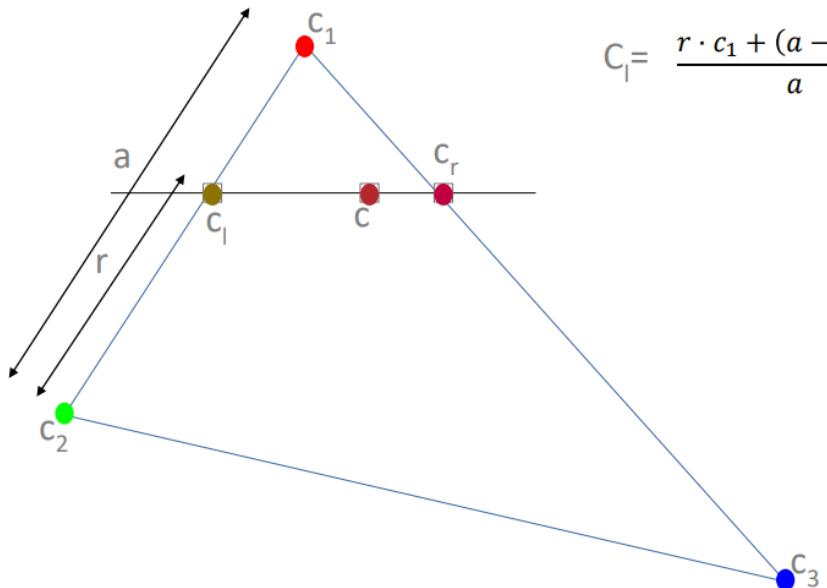
$$V_c = (x_c, y_c, z_c, w_c) \text{ on } w_c=-z_o$$

$$\begin{aligned} -w_c &\leq x_c \leq w_c \\ -w_c &\leq y_c \leq w_c \\ -w_c &\leq z_c \leq w_c \end{aligned}$$

# Rasterització: interpolació



$$C_l = \frac{r \cdot c_1 + (a - r) \cdot c_2}{a}$$



Sobre càmera perspectiva:

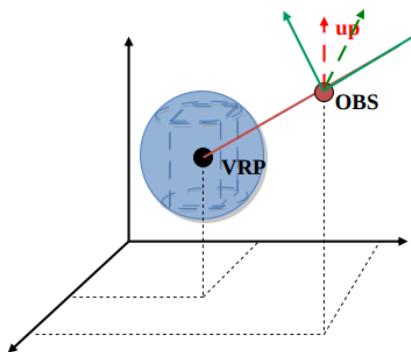
- El volum de visió sempre és respecte l'OBS!!!
- Recordem que, en perspectiva, tenim l'angle d'obertura  $\alpha$  (la meitat del FOV),  $zN$ ,  $zF$  i relació d'aspecte del window ( $raw$ ).
- El càcul de l'altura del window és  $aw = 2 zN \tan(\alpha)$ , ja que  $\tan(a) = (hw/2) / zN$ .
- L'amplada del window és  $raw \cdot aw$ .
- Per no tenir deformació cal  $raw = rav$  (window és la meva imatge, viewport és el lloc de la pantalla que tinc per aquesta imatge).
- Perspective( $FOV, raw, zN, zF$ ); projectMatrix( $PM$ );
- Quan multipliquem per la project matrix, tenim vèrtexs en coordenades de clipping. Per què això simplifica els càlculs? Perquè només hem de mirar, donat un  $VC = (xc, yc, zc, wc)$  que  $xc$  estigui entre  $-wc$  i  $+wc$ ,  $yc$  entre  $-wc$  i  $+wc$ , etc. Notem que  $wc$  és l'oposat a la coordenada  $z$  del vèrtex en coordenades d'observador (just abans de multiplicar per la Project Matrix!).
- Després d'aquest pas, es divideix el vector per  $wc$ , de manera que, un cop descartat els vèrtexs fora del rang de visió, tenim  $Vn = (xc/wc, yc/wc, zc/wc, 1)$ . Això fa més grans els vèrtexs a prop de l'OBS i més petits si estan més allunyats: efecte de les vies del tren.
- El següent pas és el Device transform: simplement és la regla de tres de transformació del meu window al meu viewport: la cantonada de dalt a l'esquerra es correspon amb la cantonada de dalt a l'esquerra, etc. (sol ser en el rang  $[-1, 1]$  a  $[0, 1]$ ).

## Processament de fragments: El fragment Shader

### Fragment Shader

```
#version 330 core
in ...
out vec4 FragColor;
void main() {
    FragColor = vec4(0, 0, 0, 1);
}
```

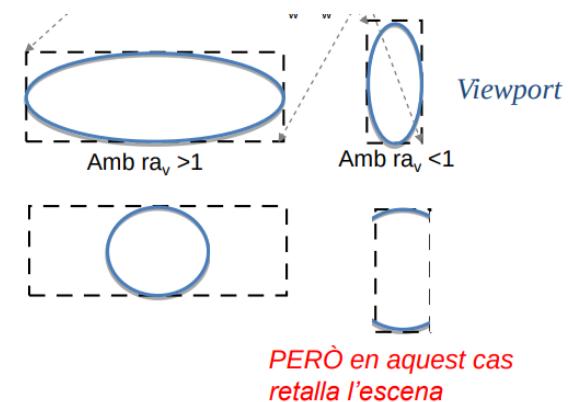
## Càmera 3a persona (1): Inicialització posicionament amb OBS, VRP, up



- Centrat => **VRP=CentreEscena**
- Per assegurar que l'escena es veu sense retallar des d'una posició arbitrària CAL que **OBS** sempre fora capsà mínima contenidora; per assegurar-ho CAL que **OBS** fora de l'esfera englobant de la capsà => distància "d" de l'**OBS** a **VRP** superior a R esfera.
  - CapsaMinCont=(xmin,ymin,zmin,xmax,ymax,zmax)
  - CentreEscena=Centre(CapsaMinCont) =  $((xmax+xmin)/2,(ymax+ymin)/2,(zmax+zmin)/2)$
  - $R=\text{dist}((xmin,ymin,zmin),(xmax,ymax,zmax))/2$
  - $d>R$ ; per exemple  $d=2R$
  - **OBS=VRP+ d\*v;**  $v$  normalitzat en qualsevol direcció;  
per exemple  $v=(1,1,1)/\|(1,1,1)\|$
  - **up** qualsevol que no sigui paral·lel a  $v$ ; si volem escena vertical (eix Y es vegi vertical)  $up=(0,1,0)$

$$\begin{aligned} ZN &= d-R; \quad ZF=d+R \\ \alpha_v &= \arcsin(R/d) \\ \Rightarrow \text{FOV} &= 2\alpha_v \\ ra_w &= a_w/h_w = 1 \end{aligned}$$

Per a què **no hi hagi deformació**, cal modificar  $ra_w$  i forçar una

$$ra_w^* = ra_v$$


- Si  $ra_v > 1$  ( $>$  que la  $ra_w$  mínima requerida que és 1) => No es retalla l'escena al fer  $ra_w^* = ra_v$  no cal modificar  $\alpha_v$  (FOV)

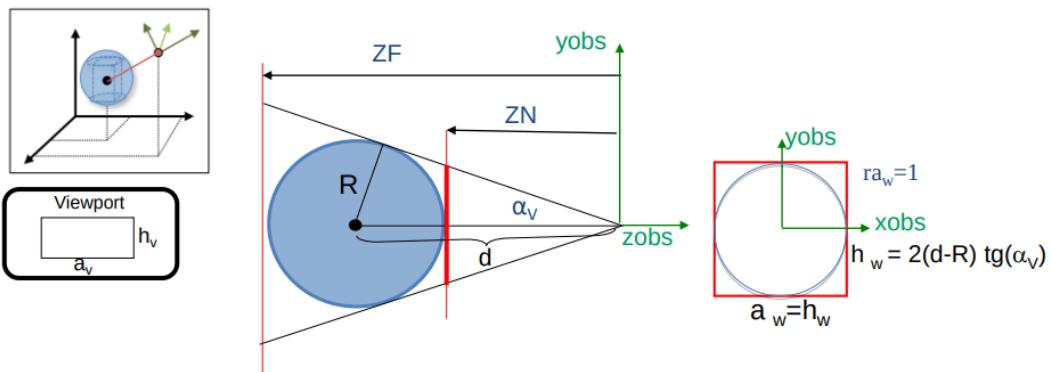


- Si  $ra_v < 1$  ( $<$  que la  $ra_w$  mínima requerida que és 1) => al fer  $ra_w^* = ra_v$  es retalla escena



Cal incrementar l'angle d'obertura  
 $\text{FOV}=2\alpha_v^*$  on  
 $\alpha_v^* = \arctg(\tan(\alpha_v)/ra_v)$

## Càmera en 3a persona (2): tota l'escena, sense deformar i òptica perspectiva



- Si tota l'esfera englobant està dins la profunditat del camp de visió, no retallem l'escena.

Per tant,  $ZN \in [0, d-R]$     $ZF \in [d+R, \dots]$

$ZN = d-R; \quad ZF = d+R$  per aprofitar la precisió en profunditat

- Per a aprofitar al màxim la pantalla, el viewport, el window de la càmera s'ha d'ajustar per veure tota l'escena; una aproximació és ajustar el window per veure l'esfera englobant.

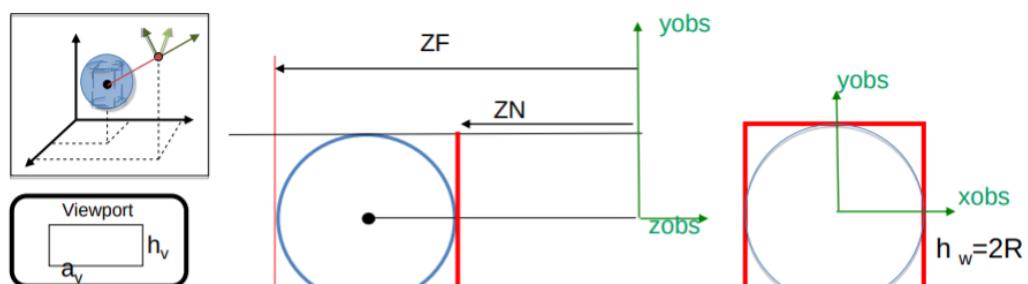
-  $R = d \sin(\alpha_v)$  ;  $\alpha_v = \arcsin(R/d) \Rightarrow \text{FOV}=2\alpha_v$

- com el window està situat en ZN,  $\alpha_v$  determina que la seva alçada sigui:

$$h_w = 2(d-R) \tan(\alpha_v)$$

- $ra_w = a_w/h_w = 1$  ( $\alpha_H$  hauria de ser igual a  $\alpha_v$  per assegurar que esfera no resulta retallada)

## Càmera 3a persona (5): tota l'escena, sense deformar i òptica ortogonal



- **ZN i ZF** mateix raonament que en càmera perspectiva.

- **Window mínim requerit (centrat)= (-R,R,-R,R)** => una  $ra_w = 1$  (per què?)

- Si  $ra_w \neq ra_v \Rightarrow$  deformació (per què?)

- Si  $ra_v > 1 \Rightarrow$  cal incrementar la  $ra_w \Rightarrow$  modificar window com  $ra_w = a_w/h_w \Rightarrow$  podem incrementar  $a_w$  o decrementar  $h_w$  (és retallaria esfera!!)

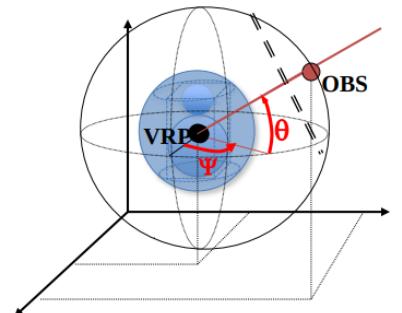
Per tant, modifiquem  $a_w$ :

$$a_w^* = ra_v * h_w = ra_v * 2R$$

$$\text{window} = (-R \ ra_v, R \ ra_v, -R, R)$$

- Raonament similar per recalcular window quan  $ra_v < 1$

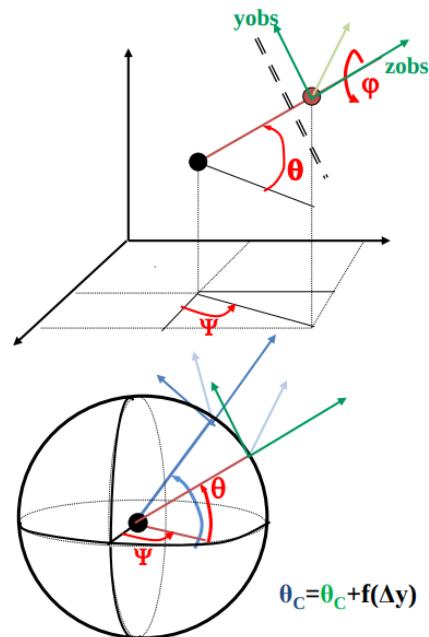
## Moure la Càmera per inspeccionar l'escena



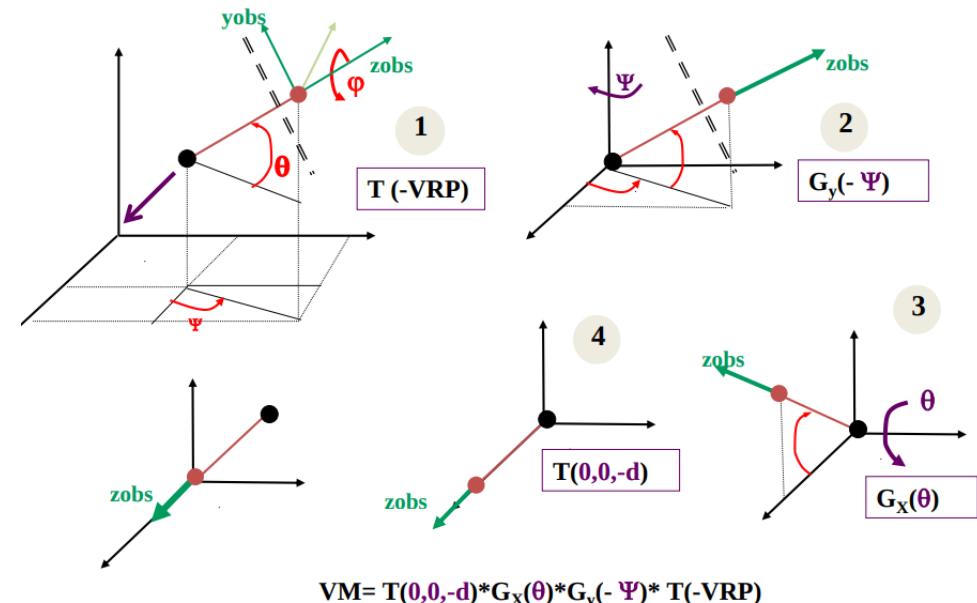
- Els angles (d'Euler) determinen la posició d'un punt en l'esfera
- Des de la interfície d'usuari desplaçem el cursor dreta/esquerra ( $\Psi$ ) i pujar/baixar ( $\theta$ ); per moure OBS sobre l'esfera
- No cal canviar l'òptica

### Com calculem OBS, VRP, up?

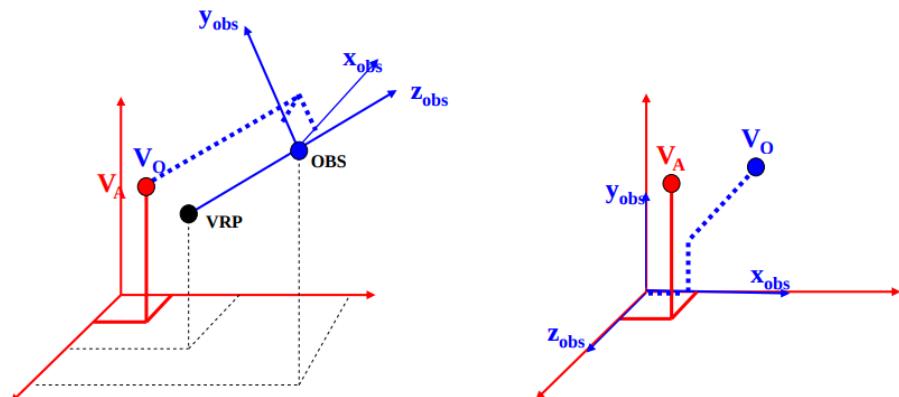
$VM = \text{lookAt}(\text{OBS}, \text{VRP}, \text{up});$   
 $\text{viewMatrix}(VM);$



## Càcul VM directe a partir d'angles Euler, VRP i d (3)



## Càcul viewMatrix directe a partir d'angles Euler, VRP i d (1)



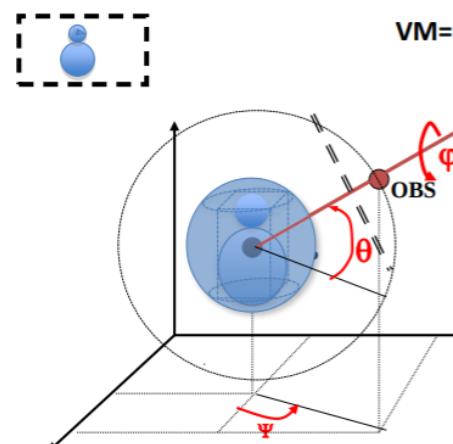
RECORDEU: La  $\text{viewMatrix}$  serveix per tenir la posició dels vèrtexs respecte del SCO

$$V_o = VM * V_A$$

Es pot calcular la VM:

- Pensant que movem la càmera (OBS, VRP, Up)
- Pensant que tenim una càmera fixa al SCA i ubiquem tota l'escena respecte d'ella → realitzar una mateixa  $TG_{VM}$  a tots els objectes. Si vèrtexs queden respecte a la càmera en la mateixa posició →  $TG_{VM}$  serà igual a la VM calculada amb OBS, VRP, Up

## Exemple: Posicionament amb angles Euler (4)



$$VM = T(0,0,-d) * G_z(-\phi) * G_x(\theta) * G_y(-\Psi) * T(-VRP)$$

$VM = \text{Translate}(0,0,-d)$   
 $VM = VM * \text{Rotate}(-\phi, 0, 0, 1)$   
 $VM = VM * \text{Rotate}(\theta, 1, 0, 0)$   
 $VM = VM * \text{Rotate}(-\Psi, 0, 1, 0)$   
 $VM = VM * \text{Translate}(-VRP.x, -VRP.y, -VRP.z)$   
 $\text{viewMatrix}(VM)$

Compta amb signes:

- Si s'ha calculat  $\psi$  positiu quan càmera gira cap a la dreta, serà un gir anti-horari respecte eix Y de la càmera, per tant, matemàticament positiu; com girem els objectes en sentit contrari, cal posar  $-\psi$  en el codi.
- Si s'ha calculat  $\theta$  positiu quan pugem la càmera, serà un gir horari; per tant, matemàticament un gir negatiu; com objecte girarà en sentit contrari (anti-horari), ja és correcte deixar signe positiu.

Sobre càmera en 3a persona + Euler:

- Tenim un OBS fora de la capsà mínima, un VRP definit al centre de la capsà i un up adient (normalment  $(0, 1, 0)$ ).
- Per rotar l'escena, pensem els moviments a fer amb la capsà per tenir la visió "per defecte" (OBS a l'origen mirant cap a Z negatives).
- Assumim angles positius cap a la dreta i cap a dalt.
- Els passos són: Translació(centre capsà), RotacióY( $-\psi$ ), RotacióX( $\theta$ ), Translació $(0, 0, -d)$ . Normalment  $d = 2 R$ .<sup>3</sup>

Càmera perspectiva sense retallar:

- Si  $rav > 1$ , no es retalla, no cal modificar el FOV. Només  $raw^* = rav$ .
- Si  $rav < 1$ , fem  $raw = rav$  i incrementem l'angle d'obertura amb  $FOV = 2 \alpha_V^*$ , on  $\alpha_V^* = \text{arctg}(\text{tg}(\alpha_V) / rav)$ .

Sobre òptica ortogonal:

- Paràmetres:  $(l, r, b, t), zN, zF$ .
- Única diferència respecte perspectiva: el  $wc$  és sempre 1 (si el punt en coordenades de clipping està a Z més pròximes o més llunyanes a l'observador no importa).

Càmera ortogonal sense retallar:

- Si  $rav > 1$  (massa ample), passem de  $(-R, R, -R, R)$  a  $(-rav \cdot R, rav \cdot R, -R, R)$ .
- Si  $rav < 1$  (massa alt), passem de  $(-R, R, -R, R)$  a  $(-R, R, -rav \cdot R, rav \cdot R)$ .

Model CIE:



- Basat en estímuls R, G i B als cons i bastons.
- És una funció  $f(R, G, B)$ ; s'acota en el pla  $X + Y + Z = 1$

### 3. Color

Diferenciem:

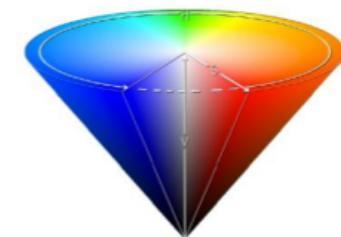
	Síntesi additiva	Síntesi subtractiva
Mitjà	Llum	Paper
Color base	Negre	Blanc
Esquema de colors	RGB	CMY(K)
Esquema visual		
	(R, G, B)	(C, M, Y)
Blanc	(1, 1, 1)	(0, 0, 0)
Negre	(0, 0, 0)	(1, 1, 1)
Red	(1, 0, 0)	(0, 1, 1)
Green	(0, 1, 0)	(1, 0, 1)
Blue	(0, 0, 1)	(1, 1, 0)
Cyan	(0, 1, 1)	(1, 0, 0)
Magenta	(1, 0, 1)	(0, 1, 0)
Yellow	(1, 1, 0)	(0, 0, 1)
Conversió	$(A, B, C) \leftrightarrow (1 - A, 1 - B, 1 - C)$	

Sobre CMY(K):

- La component K (blacK) cal perquè les tintes C, M i Y solen tenir impureses i, juntes, no fan un negre pur a la vida real.

Model HSV:

- Hue (matiz) + Saturació + Valor.
- Hue és el color (roig / blau / ...).
- La saturació indica la quantitat de gris del color.
- El valor (o intensitat).



Model CIE:



## 4. Usabilitat i més coses

### Introducció a la HCI

HCI:

- Human-Computer Interface: com interaccionen humans i computadors.
- Va d'entendre tecnologies interactives i pràctiques dels humans.
- La HCI és principalment (però no només!!) usabilitat.

User Interfaces:

- User Interfaces: eines i mètodes que es fan servir per comunicar humans i sistemes.

Usabilitat:

- La usabilitat és (def. ISO): "habilitat en què un producte es pot fer servir per un grup específic d'usuaris (!!!) per dur a terme tasques específiques (!!!) de manera efectiva, eficient, i amb satisfacció en un entorn d'ús específic (!!!)".
- La usabilitat sempre es refereix a un grup concret d'usuaris i a un usuari concret d'una aplicació!!!!
- Per tant: usabilitat = eficàcia (es fa allò que volem) + eficiència (bon ús de recursos) + satisfacció.

User experience:

- La UX és quant a dispositiu: volem crear una bona sensació al fer servir el nostre dispositiu (!!!).

Interaction design:

- Com interactuem amb els dispositius (per exemple: mida de pantalles dels mòbils, teclat físic / virtual en pantalla, etc.).
- Regla del polze (imatge).



Més d'HCI:

- En escriptoris: pantalles grans, espai per tot, ratoli, teclat, resolució gran.
- En mòbils: pantalla menuda, pensar com ajustar contingut a l'espai reduït. Problemes amb les notificacions. Interacció amb dit/stylus. Sense teclat. Poca resolució i limitacions de software.
- En tablets: mida més gran, però hem de pensar encara com fer cabre les coses. Semblant als mòbils.

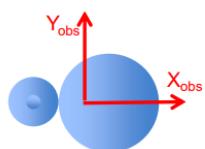
Sobre aplicacions:

- Diferenciem web apps i apps natives.
- Web apps: develop once & deploy everywhere, fàcil d'actualitzar, llenguatges fàcils; però, no és tan rica en UI, protocols ineficients i insegurs. Solen estar dissenyades per a grans pantalles amb ratolins.
- Aplicacions natives: UI més rica, molts controls, accessos a recursos locals fàcils i segurs; menys varietat de llenguatges i eines de programació, dissenyades per pantalles petites i amb control de touch. Però, no hi ha accés universal (Android vs iOS), difícil d'actualitzar, menys general que el development d'escriptori.

## Exemple 1 bis:

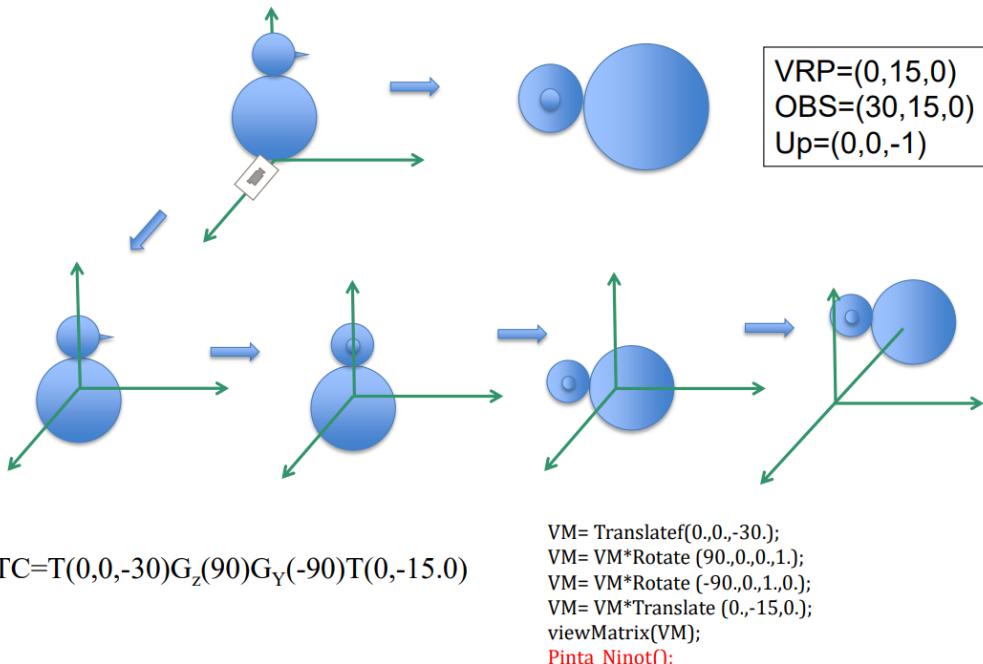


VRP=(0,15,0); OBS=(30,15,0), up=(0,1,0)

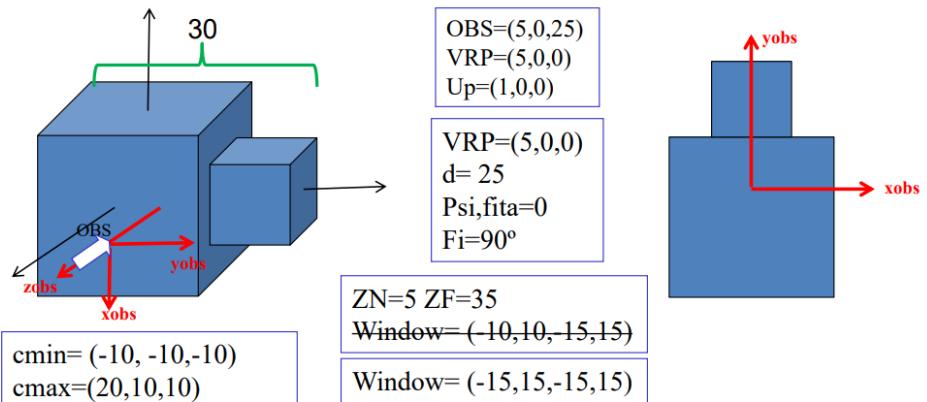


*Quins paràmetres si volem que quedí així?  
amb lookAt() i amb Euler*

## Solució Exemple 1 bis.

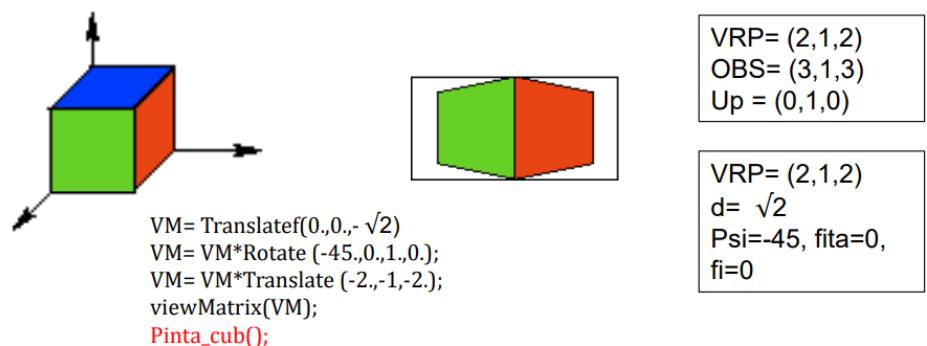


**Solució exemple 7:** Una escena està formada per dos cubes, un de costat 20 centrat al punt (0,0,0), i l'altre de costat 10 centrat al punt (15,0,0). Indiqueu TOTS els paràmetres d'una càmera **ortogonal/perspectiva** que permeti veure a la vista dos quadrats, un damunt de l'altre (el més gran a sota), de manera que ocupin el màxim de la vista (*viewport*). Cal que indiqueu la posició i orientació de la càmera especificant; a) VRP, OBS i up b) **angles Euler**. El *viewport* és quadrat.



**Solució exemple 2:** Tenim una escena amb un cub de costat 2 orientat amb els eixos i de manera que el seu vèrtex mínim està situat a l'origen de coordenades. La cara del cub que queda sobre el pla  $x=2$  és de color vermell, la cara que queda sobre el pla  $z=2$  és de color verd i la resta de cares són blaves.

Indica TOTS els paràmetres d'una càmera perspectiva que permeti veure completes a la vista només les cares vermella i verda. La relació d'aspecte del *viewport* (vista) és 2. Fes un dibuix indicant la imatge final que s'obtindria. **Posiciona la càmera amb angles d'Euler.**



- Usability is always referred to a **concrete user group** and a **concrete user application**

- **Efficacy** is the ability of correctly and completely achieving a certain goal.
- **Efficiency** is the relation of used resources and the completeness and correctness of achieved goals.
- **Satisfaction** is the comfort and acceptation of a system by the users and other people that are affected by its use.

## Principis de disseny

Basat en Bruce Tognazzini:

- Interfícies efectives: donar als usuaris sensació de control / ocultar treball intern del sistema.
- Aplicacions efectives: màxim aprofitament de l'esforç dels usuaris.

## Estètica:

- *Fashion should never TRUMP usability.*
- En qualsevol cas, tests d'usability.
- Important molt de contrast text / fons.

## Anticipació:

- Anticipar-se a les accions de l'usuari.

## Autonomia:

- Donem una mica de control (customization) a l'usuari.
- Els usuaris poden fer les seves decisions.
- Però informem l'usuari de la informació necessària per mantenir el control: estat actual, tasques...
- Ús de progress indicators, spinners...

## Color:

- Tindre cura dels discapacitats quant al color.

## Consistència:

- Molts nivells de consistència: plataforma / gamma de productes / in-app / estructures visibles / estructures invisibles / comportament d'usuari.
- Consistència de plataforma: guies de consistència (UX Android, iOS) i dins de la mateixa companyia.
- Tenim un *look & feel* comú entre productes i serveis.
- Consistència entre gammes de productes (Microsoft Office, Google apps).
- Estructures visibles: icones, símbols, posicionament similar...
- Estructures invisibles (per exemple, scroll bars invisibles en Macintosh).
- Comportament d'usuari: mai canviïs el significat d'una acció habitual!!! Canviar un gest après és molt frustrant!
- Inconsistència induïda: fem els objectes diferents i actuen de manera diferent.
- Forcem canvis visuals si les accions que trigueren aquests objectes són diferents.
- Continuitat induïda: cal esforçar-se per la continuïtat, no per la consistència.
- Les noves versions han de canviar grans àrees (noves característiques). Han de ser bastant diferents de la versió prèvia.
- Consistència amb user expectations: no forcis l'usuari a aprendre una nova manera de fer alguna cosa.

## Valors per defecte:

### Discoverability:

- Generar la il·lusió de simplicitat no simplifica les coses.
- Si l'usuari no ho troba, no existeix!
- Fer servir active-discovery: oferir característiques no testades a usuaris experts.
- Tots els controls han de ser visibles (excepció: teclats en smartphones...).

### Eficiència:

- Mirar la productivitat de l'usuari, no de l'ordinador!
- Els errors han d'ajudar. Explica què està malament, digues a l'usuari què fer.

## Interfícies explorables:

- L'usuari ha de sentir-se lliure explorant les interfícies.
  - Fer accions reversibles
  - Sempre habilitar un *undo*.

## Informar l'usuari:

- Cursor de ratolí animat si triguem  $\frac{1}{2}$  – 2 segons. Més de 2 segons, digues a l'usuari quant trigarà. Si > 5 segons, fer servir un progress. Si > 10 segons, informar l'usuari i entretenir-lo. Si > 15 segons, afegir indicador visual per saber quan hem acabat.

# Principis universals i lleis de percepció en disseny

## Regla 80/20:

- El 80% dels efectes generats en un sistema gran estan causats pel 20% de variables en aquest sistema (*80% of application usage on only 20% of its features*).

## Test d'usabilitat-estètica:

- És important l'estètica. Estètica bona = més fàcil d'usar.

## Alineació correcta:

- Coherència amb l'alignació.

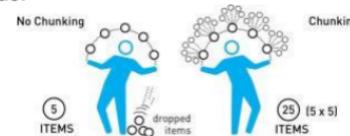
## Chunking:

- Un chunk és una unitat d'informació en la memòria a curt termini.
- Chunking vol dir que ubiquem la informació de manera que s'acomoda als límits dels humans a processar bits d'informació.
- Millor chunks petits que grans.
- Nombre màgic:  $7 \pm 2$ .

### ➤ Smaller chunks are easier to remember than larger lists

Most people can remember a list of 5 words for 30 seconds, but few can remember a list of 10 words for 30 seconds.

654563465  
654 - 56 - 34 - 65



### ➤ Magical number: $7+/-2$ (contemporary estimation $4+/-1 \rightarrow 5$ )

- It refers to elements that must be memorize:

➤ *Menu items, telephone numbers...*

## Color:

- Reforça el layout de disseny.
  - Com a molt, 5 colors.
  - Fer servir un color principal i alternatiu.
  - Simbolisme: diferent signi cat de colors en cultures.
- Saturation: Attracts attention
    - When performance and efficiency are important, the use of desaturated colors may help, perceived as more professional
    - Saturated colors attract attention and are perceived as more exciting and dynamic (but may increase eye fatigue)

## ► *LATCH principle*. Information is organized according to:

- **Location:** Information comes from different places (cities in a map, medicine: location of the body).
- **Alphabet:** Usually for large amounts of data (words in dictionary...)
- **Time:** Events with fixed durations. (meeting schedules).
- **Category:** To classify goods/elements of similar importance. Suitable for shops...
- **Hierarchy:** By magnitude, order of importance

## ► *Garbage-in garbage-out (GIGO):*

Computer scientists have long known that inadequate input information often generates bad results

- **Type error:** The input is provided in an incorrect type (*mistakes*).  
If undetected, it may generate large amounts of garbage.  
Ex.: Numerical fields filled with a phone number or credit card number...  
Type checks, input formatting, default values, example of inputs
- **Quality error:** The input has the correct type but has some defects (*slips*).  
Ex.: Amounts of money in a money transfer.  
May be alleviated with confirmations and previews.

## ► *Iconic representation:* Images try to represent objects or actions.

### Four types:

- **Similarity:** The icon is visually similar to the action/object to be represented. Adequate for simple objects (turn right) ↗
- **Example:** Elements can be related to the image (plane for airport) ✈
- **Symbolic:** High level of abstraction (unlock icon) 🔒
- **Arbitrary:** No relationship with element or action (nuclear symbol) ☢

## Lleis de Gestalt:

1. Llei de Pragnanz: llei de la bona figura, simplicitat. Formes simples.
2. Llei de tancament: 
3. Llei de similitud: agrupar elements en entitats col·lectives o totalitàries. 
4. Llei de proximitat: diferència espai per veure grups 
5. Llei de simetria: [ ] ( ).
6. Llei de continuïtat: els elements en una ruta cinètica es percepren relacionats.
7. Llei de common fate: els elements amb la mateixa direcció o moviment es percepren com a una unitat col·lectiva.

## Orientation sensitivity:

- Les orientacions vertical/horizontal estan bé. Però les obliquies costen.
- Dos efectes:
  - **Efecte d'obliquitat:** ens costa, a les neurones, interpretar dissenys amb orientació obliqua.
  - **Pop-out effect:** destaquen els elements amb bona separació de 30°.

Superioritat de les imatges en front dels textos.

Rule of thirds (imatge en 9 parts).

## ► **Signal to noise ratio:**

- A ratio higher than 1:1 indicates more signal than noise.
- *The goal of communication is maximizing signal and minimizing noise.*

Keep de design simple => *enhance perception*

We can *enhance information* by using redundant coding and highlighting.

Remove noise by eliminating unnecessary elements.

## Color perception problems

### ► Color blindness:

- Inability to distinguish the colors the same way than non-color impaired people
  - 5-10% of men
  - 1-2% of women

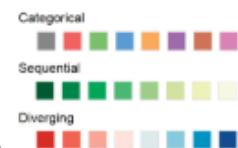
### ► Most common types of colour blindness are:

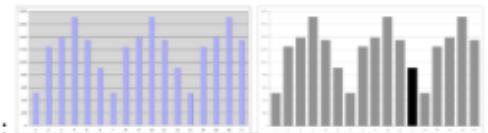
- **Deuteranopia (M cones):** Reduced sensitivity to green light (common).
- **Protanopia (L cones):** Reduced sensitivity to red light (rare).
- **Tritanopia (S cones):** Reduced sensitivity to blue light (very rare).
- **Achromatopsia:** Cannot see any colour at all. Also not very common.

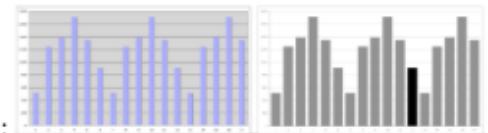
- Es veuen amb un test d'Ishihara: 

Consells:

- Forcem contrast fons / color de text.
- Evitem colors adjacents o de lluminositat similar.
- El fons ha de ser consistent.
- Contrast colors light amb colors dark.
- Fer servir colors per a alguna cosa, no malbaratar-los!
- Usem colors softs per a la majoria d'informació i colors brillants/dark per a informació que requereix més atenció.

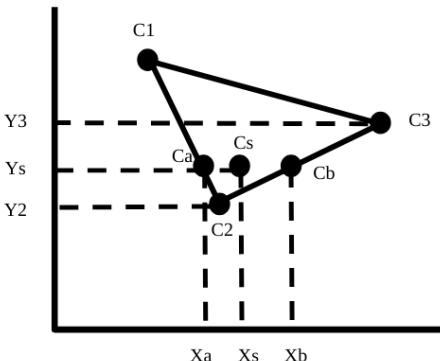


- Sense ordre / ordre sense valor neutral / amb valor neutral: 

- Evitar soroll als gràfics (de-emphasizing): 
- Evitem combinació verd + roig al mateix esquema (solució roig + gris).
- Evitar 3D.
- Fer servir colors opositius, anàlegs (taronja / verd / groc...), triades (porpra / taronja / blau), quatríades (porpra / taronja / verd / blau, és el més estrany).

# Shading (colorat) de polígons

- Colorat Constant  $\equiv$  Flat shading  $\rightarrow C_f = C1$   
color uniforme per tot el polígon (funció del color calculat en un vèrtex); cada cara pot tenir diferent color.
- Colorat de Gouraud  $\equiv$  Gouraud shading  $\equiv$  Smooth shading

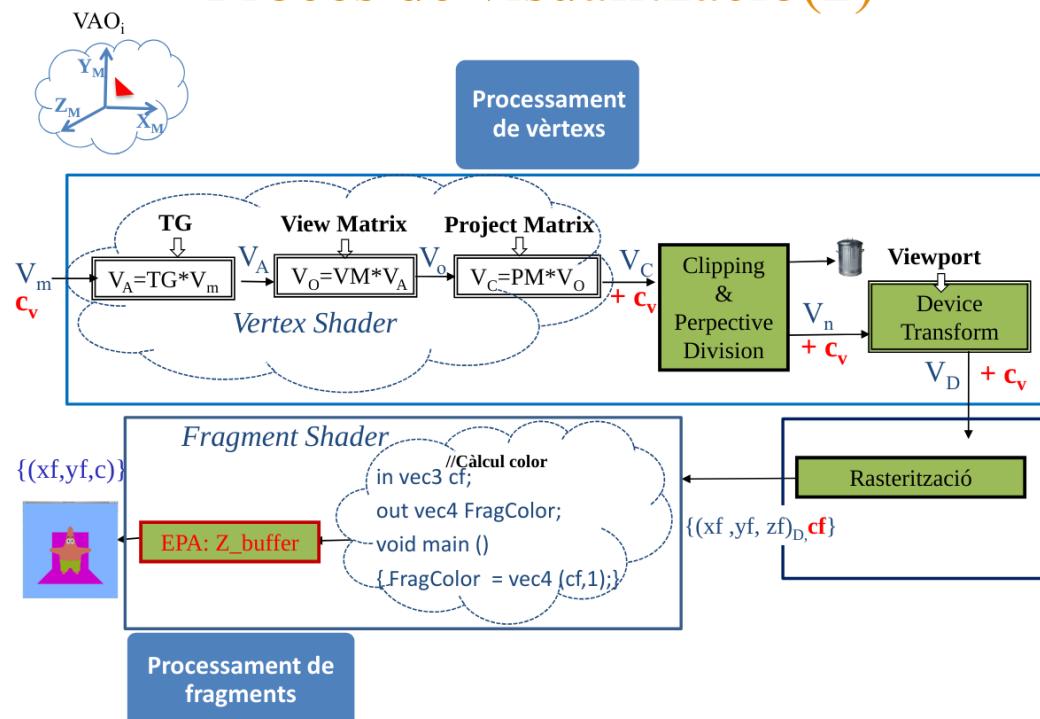


$$Ca = \frac{1}{Y_1 - Y_2} (C_1(Y_s - Y_2) + C_2(Y_1 - Y_s))$$

$$Cb = \frac{1}{Y_3 - Y_2} (C_2(Y_3 - Y_s) + C_3(Y_s - Y_2))$$

$$Cs = \frac{1}{X_b - X_a} (Ca(X_b - X_a) + Cb(X_s - X_a))$$

## Procés de visualització(2)



# Depth Buffer

- Mètode EPA en espai imatge (a nivell de píxel/fragment)
- Després de la **rasterització i del Fragment Shader**
- Requereix conèixer per a cada píxel, un valor (depth) que sigui proporcional a la distància a l'observador a la que es troba el polígon que es projecta en el píxel.
- No importa ordre en que s'enviïn a pintar els triangles (ordre en què estiguin en VBO)
- No requereix tenir el Back-face culling activat

## Depth Buffer (z-buffer)

Dos buffers de la mateixa resolució que la pantalla

Buffer color (frame\_buffer)

$(r, g, b) \in [0, 2^n - 1]$

Buffer profunditats (depth\_buffer)

$z \in [0, 2^{nz} - 1]$

- Inicialitzar al color de fons

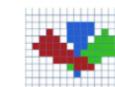


`glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)`

- Inicialitzar al més lluny possible



- Per a cada fragment



$\{(x_f, y_f, cf)\}$



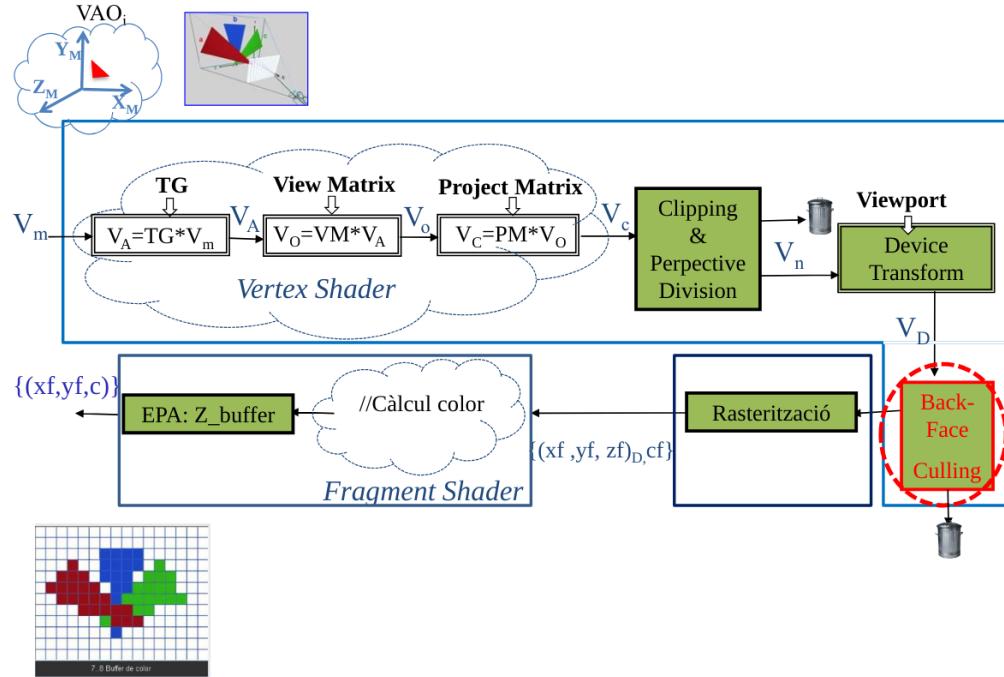
$\{(x_f, y_f, zf, cf)\}$

```

if (zf < depth_buffer[xf,yf]) {
    depth_buffer [xf,yf] = zf;
    color_buffer [xf,yf] = cf;
}

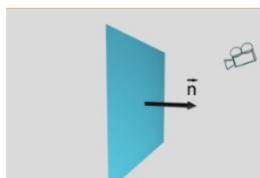
```

# Procés de visualització: EPA (2)

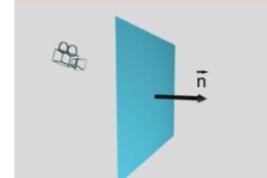


## Back-face Culling

- Mètode EPA en espai *objecte* (a nivell de triangle)
- Requereix cares orientades, opaques, objectes tancats
- Considera escena formada només per la *cara* i l'*observador*
- És conservatiu (determina les cares que “segur” no són visibles)



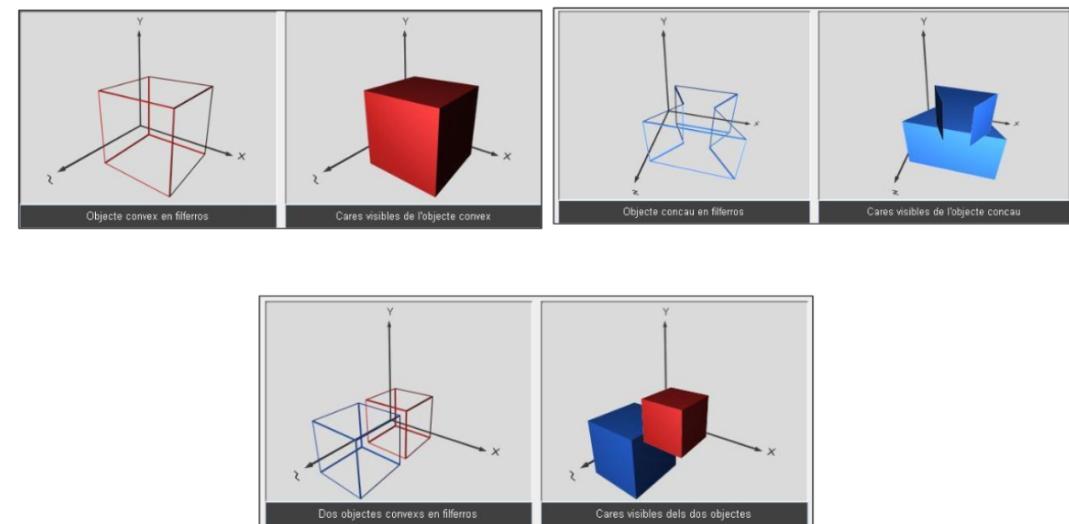
visible



no visible

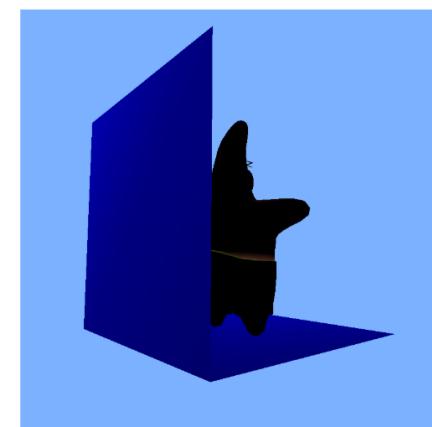
## Back-face Culling

- Culling com a EPA només si l'escena conté un únic objecte convex.

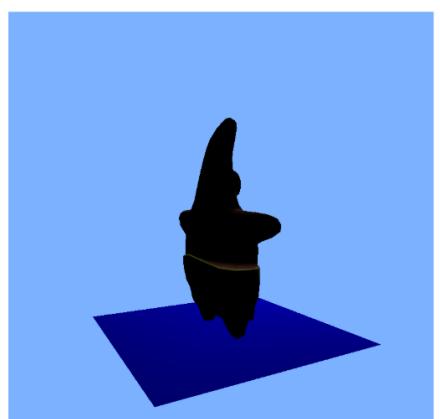


Exemple que podreu comprovar al laboratori

Sense back-face culling



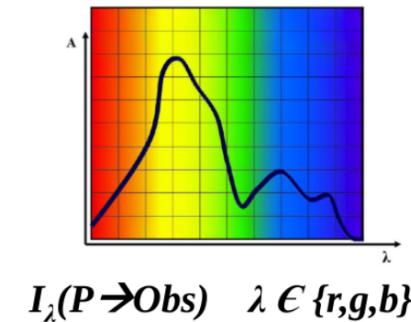
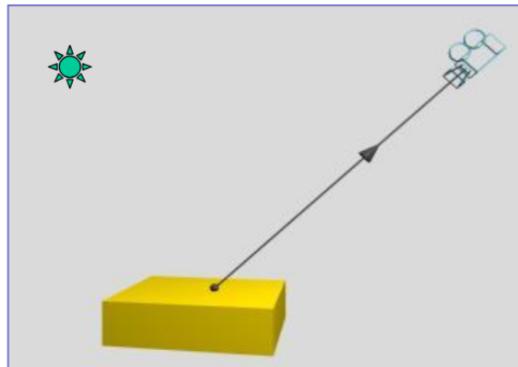
Amb back-face culling



`glEnable(GL_CULL_FACE);`

# Color d'un punt

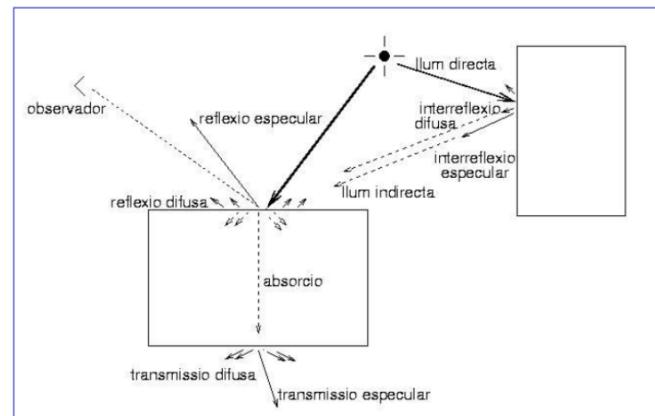
El color amb el que un Observador veu un punt P de l'escena és el color de la llum que arriba a l'Obs procedent de P:  $I_\lambda(P \rightarrow Obs)$



## Elements que intervenen

El color que arriba a l'Obs procedent de P,  $I_\lambda(P \rightarrow Obs)$ , funció de:

- Fonts de llum
- Materials
- Altres objectes
- Posició de l'observador
- Medi pel que es propaga



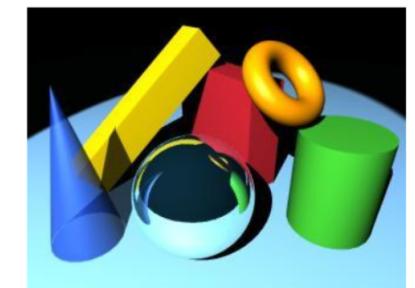
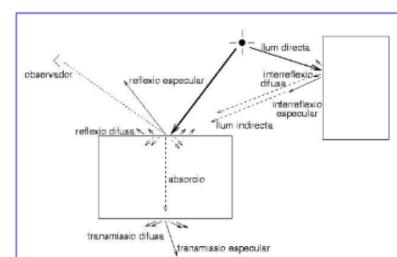
## Models locals o empírics

- Només consideren per al càlcul del color: el punt P en què es calcula, els focus de llum (sempre puntuals) i la posició de l'observador.
- No consideren altres objectes de l'escena (noombres, no miralls, no transparències).
- Aproximen la transmissió de la llum per fòrmules empíriques i les propietats de reflexió dels materials per constants.



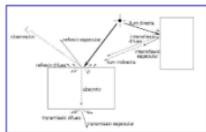
## Models de traçat de raig

- Els models d'il·luminació de traçat de raig consideren:
  - Focus de llum puntuals
  - Altres objectes existents en l'escena però només transmissions especulars
- Permeten simularombres, transparències i miralls.
- Són més costosos en càlcul.



# Models de radiositat

- Consideren els focus de llum com un objecte qualsevol de l'escena.
- Els objectes només poden produir reflexions difuses pures.
- Com que totes les reflexions són difuses, la radiositat no considera la posició de l'observador.
- Poden modelar ombres i penombres, però no miralls ni transparències.
- Són els més costosos i es basen en l'anàlisi de l'intercanvi d'energia entre tots els objectes de l'escena.



## Model empíric ambient

- No es consideren els focus de llum de l'escena.
  - La llum ambient és deguda a reflexions difuses de llum entre objectes, per tant es considera que no prové de cap focus específic i no té cap direcció concreta.
  - Tots els punts de l'escena reben la mateixa aportació de llum.
  - S'observarà el mateix color en tots els punts d'un mateix objecte.
- Equació:  $I_a(P) = I_{a\lambda} k_{a\lambda}$
- $I_{a\lambda}$ : color de la llum ambient
  - $k_{a\lambda}$ : coef. de reflexió ambient



Exemple amb una esfera amb:

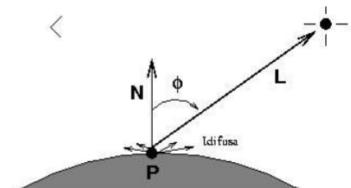
$$I_a = (1,1,1)$$

$K_a$  també blanca amb intensitat variant entre 0.2, 0.4, 0.6, 0.8 i 1



## Model empíric difús (Lambert)

- Focus puntuals. Objectes només tenen reflexió difusa pura.
- Podem imaginar que el punt  $P$  irradia la mateixa llum en totes direccions i per tant el seu color no depèn de la direcció de visió.



$$I_a(P) = I_{f\lambda} k_{d\lambda} \cos(\Phi) = I_{f\lambda} k_{d\lambda} \text{dot}(N, L)$$

*si  $|\Phi| < 90^\circ$*

- $I_{f\lambda}$ : color ( $r,g,b$ ) de la llum del focus puntual  $f$
- $k_{d\lambda}$ : coef. de reflexió difusa del material
- $\cos(\Phi)$ : cosinus de l'angle entre la llum incident i la normal a la superfície en el punt  $P$ .

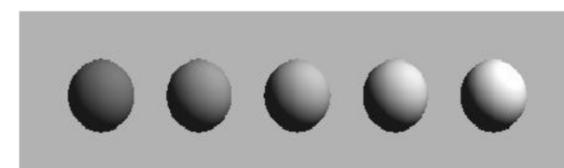
*Pot calcular-se com el producte escalar entre  $N$  i  $L$  si estan normalitzats.*



Exemple amb una esfera amb:

$$I_f = (1,1,1)$$

$K_d$  també blanca amb intensitat variant entre 0.4, 0.55, 0.7, 0.85 i 1



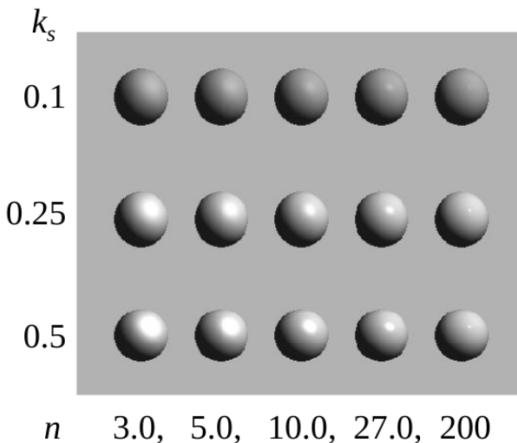
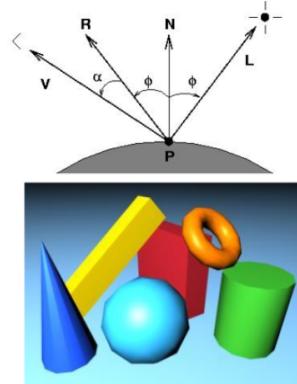
# Model empíric especular (Phong)

- Focus de llum puntuals i objectes només reflexió especular.
- L'observador només podrà observar la reflexió especular en un punt si es troba en la direcció de la reflexió especular.
- La direcció d'especularitat és la simètrica de  $\mathbf{L}$  respecte  $\mathbf{N}$  i es pot calcular com:  $\mathbf{R} = 2\mathbf{N}(\mathbf{N} \cdot \mathbf{L}) - \mathbf{L}$  si tots els vectors són normalitzats.

$$I_\lambda(P) = I_{f\lambda} k_{s\lambda} \cos^n(\alpha) = I_{f\lambda} k_{s\lambda} \operatorname{dot}(R, v)^n$$

si  $|\Phi| < 90^\circ$

- $I_{f\lambda}$ : color (r,g,b) del focus puntual f
- $k_{s\lambda}$ : coef. de reflexió especular (x,x,x)
- n : exponent de reflexió especular
- v és vector normalitzat que uneix punt amb Obs



Exemple d'una esfera amb:

$$I_f = (1,1,1)$$

$k_d$  blanca amb intensitat 0.5

$k_s$  blanca amb 0.1, 0.25 i 0.5

n : 3.0, 5.0, 10.0, 27.0, 200

# Resum

Color d'un punt degut a...	Depèn de la normal?	Depèn de l'observador?	Exemple
Model ambient	No	No	
Model difús	Sí	No	
Model especular	Sí	Sí	

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

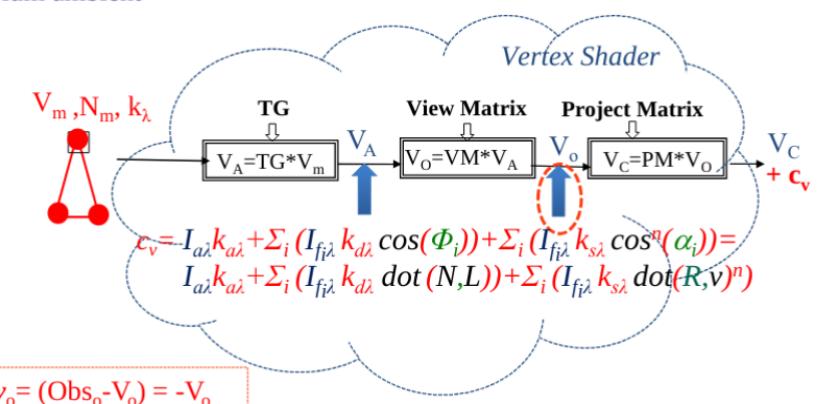
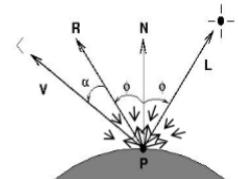
## Càlcul del color per vèrtex en el Vèrtex Shader (1)

Atributs per cada model:

- Coordenades (V), normal (N) i constants de material ( $k_i$ ) per vèrtex en VBOs del seu VAO

Uniforms:

- Fonts de llum actives => color, posició
- Llum ambient



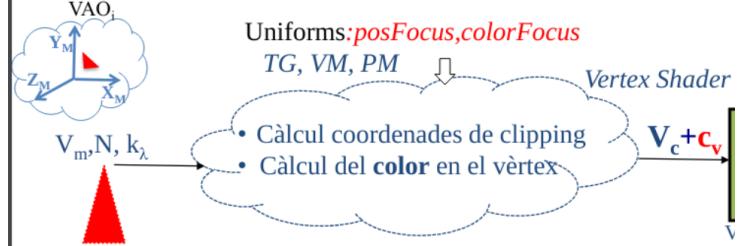
## Càcul del color per vèrtex en el Vèrtex Shader (2)

El càlcul el farem per cada vèrtex (al Vertex Shader)

I el farem en SCO, per tant:

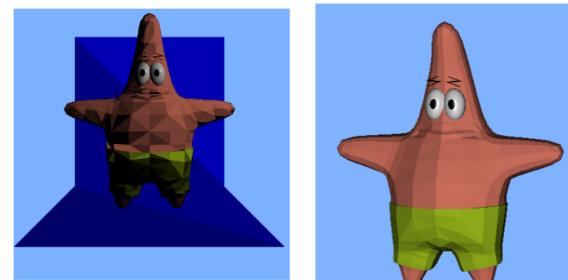
- Cal passar la posició del vèrtex a SCO
  - multiplicant per (**view \* TG**)
- Cal passar el vector normal a SCO
  - multiplicant per la matriu **inversa de la transposta de (view \* TG)**, -li direm **NormalMatrix**-  
`mat3 NormalMatrix = inverse (transpose (mat3 (view * TG)))`
- La posició del focus de llum també ha d'estar en SCO
  - Multiplicat per **view** (si no la tenim directament en SCO)

### Càcul del color en el Vèrtex Shader



- Colorat Constant  $\equiv$  **Flat shading**  $\rightarrow C_f = C_1$   
color uniforme per tot el polígon (funció del color calculat en un vèrtex); cada cara pot tenir diferent color.
- Colorat de Gouraud  $\equiv$  **Gouraud shading**  $\equiv$  **Smooth shading**

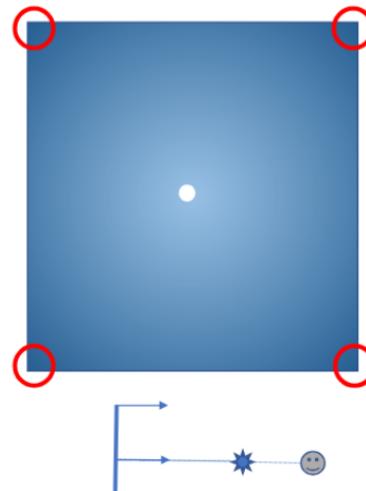
Flat versus Gouraud/smooth Shading



## Càcul color en VS: limitacions

Efecte del càlcul de Lambert i Phong en Vertex Shader:

Com s'hauria de veure



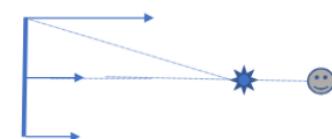
Com es veu



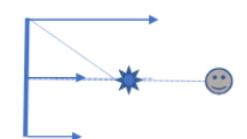
Possible solució: **Discretitzar més**

## Càcul color en VS: limitacions

Què passa si apropiem el focus de llum al quadrat?

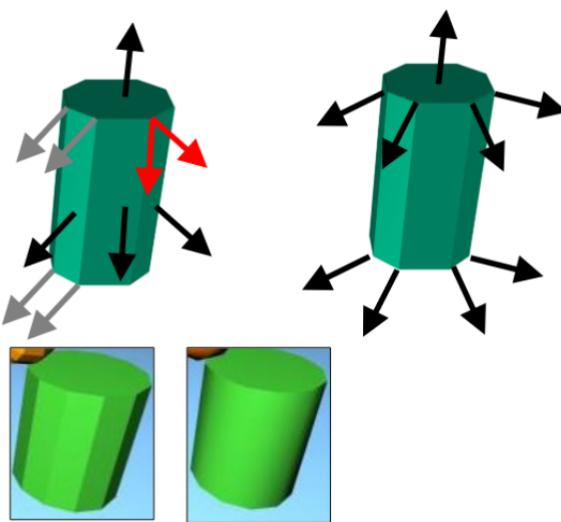


angle  **$\Phi$**  creix



## Suavitzat d'arestes (2)

- Normal per cara vs normal per vèrtex

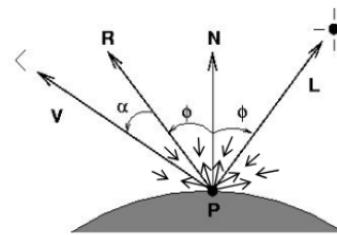


$$\vec{N}_v = \frac{\sum_i \vec{N}_i}{\left\| \sum_i \vec{N}_i \right\|}$$

## Càcul del color per fragment en el Fragment Shader (3)

### Proposta de solució:

- ✓ Tenim la informació de les llums, són *uniforms*
- ✓ Podem fer “out” en el VS dels atributs associats al vèrtex: N i V en SCO, i de les constants empíriques de material. La rasterització calcularà el seu valor pel fragment interpolant la informació dels vèrtexs del triangle → tindrem els seus valors en el FS com variables “in” 😊



$$FragColor = I_{a\lambda} k_{a\lambda} + \sum_i (I_{f\lambda} k_{d\lambda} \cos(\Phi_i)) + \sum_i (I_{f\lambda} k_{s\lambda} \cos^n(\alpha_i))$$

$$\cos(\Phi) \Rightarrow \text{dot}(L, N) \text{ en SCO}$$

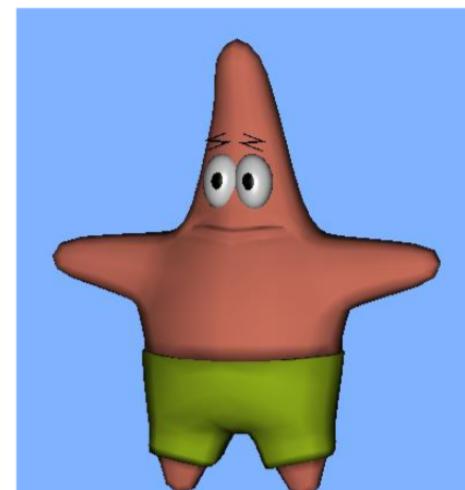
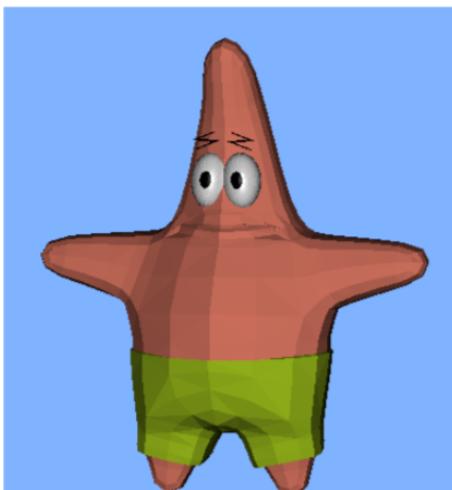
$$\cos(\alpha) \Rightarrow \text{dot}(R, v) \text{ en SCO}$$

### Observació:

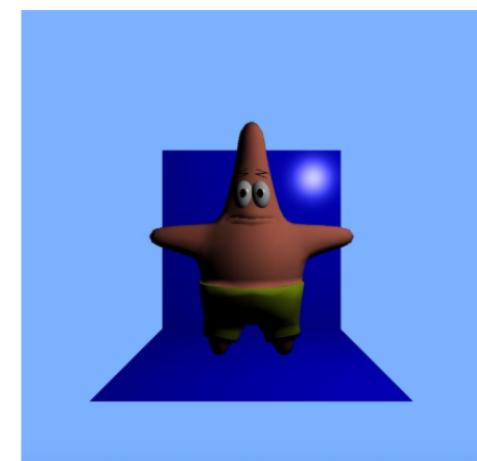
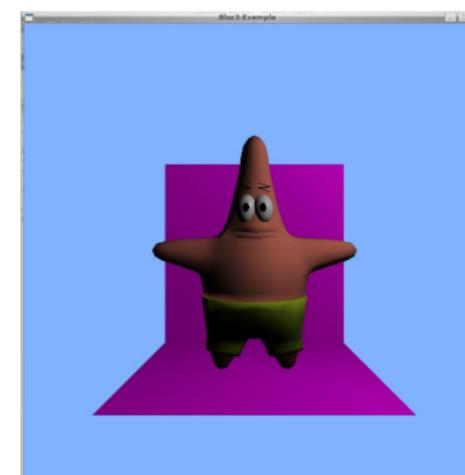
- La normal del fragment és una aproximació de la normal del punt del triangle que es projecta en el fragment. Aquesta interpolació de la normal es coneix com “shading de phong”

## Suavitzat d'arestes: exemple

- Normal per cara vs normal per vèrtex



Exemple final: Sense il·luminació i Il·luminació en el VS versus FS



## Interacció de disseny i avaluació

- Entropia de Shannon: la quantitat d'informació per tramestre un missatge.
- Font + transmitter + canal + receiver.
- Amb {A, B, C} tenim  $\log_2(3)$ .
- Amb A1, A2, B1, B2, C1, C2 tenim suma d'incerteses = producte de continguts dels logaritmes =  $\log_2(6)$ .
- La probabilitat és P. Per tant  $\log_2(M) = \log_2(1/P) = -\log_2(P)$  (es diu la "sorpresa").
- El sumatori de les  $p_i$  és 1.

$$-\sum_{i=1}^N p_i \log_2 p_i$$

- La entropia és, en definitiva  $H = -\sum_{i=1}^N p_i \log_2 p_i$ , on N és el nombre de caràcters.
- Interferència: la quantitat mitjana d'informació rebuda és  $R = H(x) - H_y(x)$ , on  $H_y(x)$  és l'equivocació o entropia condicional de x quan y és coneuguda.

## Llei de Hick-Hyman:

- Descriu el temps de decisió en funció de la informació.
- Costa més respondre a un estímul quan aquest pertany a un gran conjunt, a una diferència de quan hi ha menys estímuls.
- Per fer una decisió, el reaction time és  $RT = a + b H_t$ , on a i b són constants i  $H_t$  és la informació tramesa.
- Per tant,  $RT = a + b \log_2(n + 1)$ . Aquest +1 és per la incertesa de si respondre o no.

$$RT = a + b \log_2(n + 1)$$

## Llei de Fitts:

- Relació linear entre la dificultat d'una tasca i el temps de moviment.
- Task difficulty = ID =  $\log_2(2A / W)$ , on A és l'amplitud del moviment i W és l'amplada del target.
- Movement time MT =  $a + b \cdot ID$ . Paràmetres a i b.
- No funciona molt bé per a targets petits.
- És vàlida per a ratolins, joysticks, dit, stylus... però falla en altres.

$$ID = \log_2\left(\frac{2A}{W}\right)$$

## Llei de Fitts. Variant de MacKenzie:

$$\bullet \quad MT = a + b \cdot ID = a + b \cdot \log_2\left(\frac{D}{W} + 1\right)$$

### Implicacions de la llei de Fitts:

- Les cantonades tenen *width* infinit, per tant, són fàcils d'aplegar.
- Mantenir les coses relacionades a prop.
- Els filtres han d'estar pròxims a la barra de cerca.
- Els elements oposats separats.
- Menús de pop-up: redueix distància de travel, només usats per experts!
- Incrementem percepció si agrupem les coses amb buit pel mig.
- Com més gran és un botó, més usable és.
- Fer les coses destructives / delicades més difícils: fem servir slides en comptes de buttons per apagar el nostre mòbil.

$$MT = a + b \log_2\left(\frac{D}{W} + 1\right)$$

- D is the distance of movement
- W is the width of the target

## Accelerar targets:



- Bubble targets: pot distraure ( ).

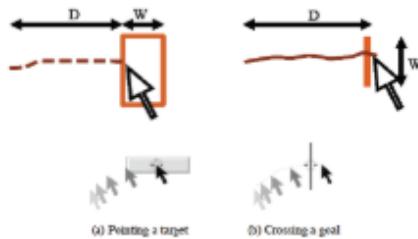


- Augmentar mida amb coses properes al cursor
- Bubble cursor (fa servir diagrames de Voronoi).
- Attract pointer: en 3D.

## Control-display ratio:

- Hi ha un mapeig moviment mà a la vida real i moviment cursor (parlem de 3D). Com fem aquest mapeig? Constant, depèn velocitat mouse / cursor.

## Llei de crossing:



- 
- Pot ser continu o discret (soltem el cursor), ortogonal o colineari.
- Crossing i pointing són bastant semblants. Menys error en crossing, però.

$$T = a + b \log_2 \left( \frac{D}{W} + 1 \right)$$

- T is the average moving time between passing the two goals.
- D is the distance between the two goals
- W is the width of each goal
- a and b are constants to be determined

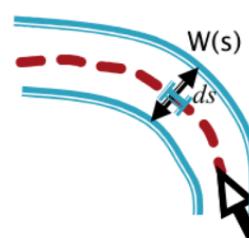
## Llei de Steering:

- Navegació per menús / navegació 3D / drag and drop / dibuix.

### ▶ Movement time across the path $T_s$ :

$$T_s = a + bID_s \quad T_s = a + b \int_C \frac{ds}{W(s)}$$

- C is the length of the path
- W(s) is the path width at point s



## Dispositius de control directe:

- Problemes: estratègia land-on: seleccionar a un clicking point, feedback més ràpid, propens a errors.
- Estratègia lift-off: el click inicial crea el cursor; en arrossegar tenim precisió.
- Són durables.

### ► Direct-control devices. Issues:

- Imprecision in pointing. Many factors:

- *Quality of the screen:*  
Capacitive screens less precise than resistive
- *Size of the pointer*  
*Fat and not-so-fat fingers*



#### ◦ Land-on strategy:

- Select on clicking point
- Faster feedback
- Prone to errors

#### ◦ Lift-off strategy:

- Initial click creates “cursor”, dragging used for precision pointing, lift-off selects
- More time consuming



### ► Indirect-control devices:

- Examples:

- Mouse, trackball, joystick, graphics tablets...

#### ◦ Issues:

- Alleviate hand fatigue
- Eliminate screen occlusion
- Mouse is the clear king
  - Cost-effective
  - Precise
  - Hand has a surface to rest on
  - Buttons easy to press
  - Long movements require to pick up mouse and replace
  - May be improved using accelerated moves



## Teclats

### QWERTY:

- Les tecles més usades estan lluny unes de les altres.
- Està dissenyat per l'anglès.

### AZERTY:

- Dissenyat pel francès.

### Disseny de teclats:

- Balancejar mà dreta i esquerra.
- Maximize the load on the home row.
- Maximitzar la freqüència d'alternança entre seqüències de mans.
- Minimitzar la freqüència de fer servir el mateix dit.
- El temps per moure d'una tecla a una altra depèn de la distància de la tecla i l'amplada de les tecles.  $MT = a + b \log_2 (D / W + 1)$ .
- Per anar més ràpids, podem canviar la mida del target, canviar la distància o canviar la velocitat del cursor.

### Problemes comuns:

- Autocorrecció. Millor no fer-la servir.
- Majúscula automàtica (per exemple, en e-mails).
- Suport de diferents idiomes.

$$MT_{ij} = a + b \log_2 \left( \frac{D_{ij}}{W_{ij}} + 1 \right)$$

- $D_{ij}$  is the distance between keys  $i$  and  $j$ ,
- $W_{ij}$  is the width of each key
- Bi et al. also use the effective width

### DVORAK:

- Vocals en un costat

### Teclats partits:

- No proporcionen cap millora de velocitat.

## Formal usability tests. Tasks and roles

### ► Usability test roles:

- A: Test administrator
- B: Briefing
- CO: Camera Operator
- DR: Data Recorder
- HD: Help Desk Operator
- PE: Product Expert
- S: Statistician

### ► Preparation (A):

- Product understanding: Purpose of the product, parts ready to test, type of users...: A, PE
- Test purpose: Product comparison, within/between subjects...: A, S
- Measures/Goals: Number of iterations, counting mistakes/errors, timings...: A, S

### ► Product understanding (A + PE):

1. Understand the purpose of the product
2. Parts of the product are ready for testing
3. Types of people who will use the product
4. Determine the use given to the product
5. Conditions of usage of the product



## ▶ Preparation:

- Product understanding: Purpose of the product, parts ready to test, type of users...: A, PE
- Test purpose: Product comparison, within/between subjects...: A, S
- **Measures/Goals:** Number of iterations, counting mistakes/errors, timings...: A, S



## ▶ Implementation:

- Participants' selection: A
- Task scenarios: initial conditions, steps: A
- Pilot test: Members of the team: A, B, CO, DR, HD
- Testing, A, B, CO, DR, HD

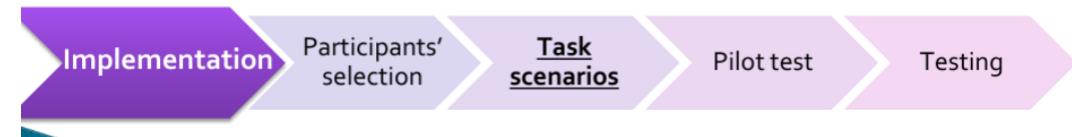
## ▶ Participants' selection

- It's complicated.
- Should be representative
  - People that could be real users
    - E.g. no other managers!!!
  - Specialized recruiting agencies are a possibility
- No-show rates above 10%
- For statistical significance: 10–12 participants
- Less formal: 4–5 participants per user group
- Ensure recruiting criteria reflects user characteristics
  - E.g. for a website, ensure participants have prior experience browsing



## ▶ Test task & scenarios:

- Tasks must be representative
  - Core tasks: Features that everybody uses (write a text)
  - Peripheral tasks: Features used less often (table insertion)
- Scenarios must be determined
  - Define initial conditions
  - Description of the scenario: what to do and why
    - Should not provide step-by-step instructions but should include details
    - Some action must be taken on finish
  - Not all users must be provided with the same scenarios (may depend on the user profile)



## ▶ Testing (A, B, CO, DR, [HD]):

- Brief participants: B
- Initial questionnaire: B
- Develop tasks: B, CO, DR, [HD], A
- Debrief: B
- Final questionnaires: B



## ► Reporting (whole team):

- Data Analysis & Evaluation: A, S
- Issues/Measures & **Recommendations**: A, S, team
- Report: A, S, team
  - Describe & prioritize the usability problems
  - Present quantitative measurements

## ► Data analysis & evaluation:

- Frequency: Number of users that find a problem divided by the number of users testing the app or web
  - Easy (objective) to evaluate
- Severity: Importance of the problem
  - Might be completely catastrophic or simply cosmetic
  - Difficult (more subjective) to evaluate

## ► Usability problems:

- Should indicate the importance: severity
- Can be classified:
  - Mistakes: Errors due to incorrect intention
  - Slips: Errors due to appropriate intention but incorrect action
- Expertise does not affect on the number of errors
  - But affects how fast they are handled

## ► Problem evaluation. Dumas and Redish:

- **Level 1**: Prevents Task Completion
- **Level 2**: Creates significant delay and frustration
- **Level 3**: Problems have a minor effect on usability
- **Level 4**: Subtle and possible enhancements/suggestions



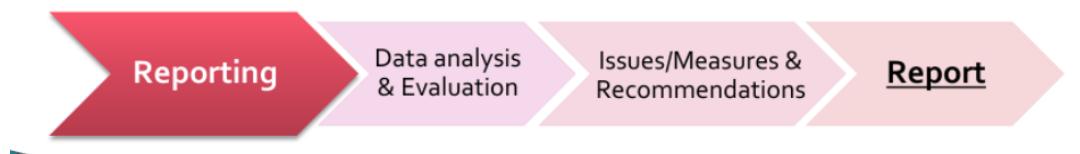
## ► Recommendations:

- Create a problem grid: frequency/impact
- Global changes (prevent task completion) first
  - A *missing help* may be a global problem or something related with a concrete UI
- Try to give at least one recommendation for each problem
  - Present the different trade-offs clearly



## ► Problem evaluation. Conclusions

- Do not use a large number of categories
  - Do not get obsessed by the number of categories either
- Different evaluators may disagree on some problems' severity
- Treat frequency separately from severity
- Do not forget to point out positive findings



## ► Guerrilla usability testing

- Take someone in a coffee or public space and ask her to use a website for a couple of minutes
- Observe users
  - Ask open-ended questions such as “What would you do here?”
- Get to know them a bit
  - Offer coffee or bagels
- Analyse captured data
  - Considering your audience

## ► “Usability testing on 10 cents a day”

- Prepare some tasks to evaluate
- Grab somebody from the company as user
- Gather stakeholders in an observing room
- Let the user do a set of tasks
- Capture gestures, mouse, record...
- Discuss over lunch (order pizza for everybody)
- Report

## ► Remote testing

- Like traditional tests but participant and facilitator are in different physical locations
  - Participants can do the test at home
  - Facilitator watches remotely

## ► Remote testing. Two types:

- Unmoderated:
  - Users do the task completely alone
- Moderated:
  - Users have access to a facilitator

## ► Unmoderated Remote testing

- Users don’t have real-time support
- Don’t get any clue on how the session went
- No opportunity to ask detailed questions
  - Sometimes the software allows to have some of them predefined
- Preferable to work only on a few specific elements than a broad view of a product
- Good for tight timeframes

## ► Moderated Remote testing

- Facilitator can change or reorder tasks as needed
- Facilitator can ask follow-up questions or clarifications
- Participant is less likely to spend time on tasks not related to the test
- Test sessions can be longer (usually about an hour)
- Can perform more in-depth tests
- The team can watch the test and discuss afterwards

## ► Heuristic evaluation:

- 3–5 usability experts evaluate an app or UI

### ► Heuristic evaluation. Process:

- Collect the UI
- Understand the business and users’ needs
- Understand user motivations and tasks to accomplish
- Define the heuristics
- Use a minimum of 3 experts
- Set up a consistent evaluation system
- Highlight problem(s) and its rating
- Compare and analyse the results of multiple experts

## Realitat virtual

De nició:

- La realitat virtual és una simulació interactiva per computador des del punt de vista del participant, en la qual se substitueix o s'augmenta la informació sensorial que rep

Elements:

- **Simulació interactiva:** Reproduir un món que només existeix a l'ordinador.
- **Immersió sensorial:** Desconnectar els sentits del món real per tal de portar-los cap al món virtual. "Visual immersion": Els objectes existeixen independentment del dispositiu de visualització. Utilitza visió estereoscòpica per reproduir el món.
- **Interacció implícita:** El sistema decideix què vol l'usuari a partir dels seus moviments.

## Virtual Reality Systems

‣ Immersive



‣ Semi-Immersive



## VR Interaction & 3D Selection

### ‣ Hand extension techniques or 3D point cursors

- A 3D point in space is represented as a mapping of the user's hand position.

### ‣ Ray-based techniques

- Use the hand position and some element to indicate orientation
- A ray is generated in space and is used as a pointer
  - Also called aperture-based selection techniques or ray cursors



### ‣ Hand extension:

- May require ample movements due to the direct mapping with 3D world
- Sometimes elements are difficult to reach
- May be more intuitive if virtual world represents some real world

### ‣ Ray-based techniques:

- Hand position + wrist orientation
- Head position and hand direction

## Realitat augmentada

Introducció:

- Realitat augmentada: és la combinació de l'escena real i la virtual, tenim accés al món virtual i real al mateix temps.
- Objectiu: millorar el rendiment i la percepció del món però mantenint la diferència entre el real i el virtual.
- A diferència de la RV, la RA no té una immersió total.
- Es molt important registrar els objectes de la realitat per tal d'evitar problemes en aplicar la RA.

### Conceptes:

- **Video see-through:** combina vídeo en temps real amb càmeres frontals que posen imatges virtuals.
- **Optical see-through:** l'usuari veu el món real directament, un exemple són les *Google Glasses* (requereix un calibratge constant, no es pot posar efectes oclusius).
- **AR projection:** les imatges són projectades directament sobre els objectes físics. No és necessari cap aparell visual per veure'ls, però cal el calibratge dels projectors per les diferents distàncies a més que només es poden utilitzar en interiors per la seva brillantor tan baixa.