

Gràfics i Visualització de Dades

Tema 4: Dels vèrtexs als fragments: projecció, clipping i mètodes de rasterització

Anna Puig

NOTA: Aquestes transparències es corresponen a les explicacions del tema 4 del llibre de referència bàsica

[Angel2011] Edward Angel, Dave Shreiner, **Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL, 6/E**, ISBN-10: 0132545233. ISBN-13: 9780132545235, Addison-Wesley, 2011

Objectius

- En el tema 4 s'estudiaran:
 - els **tipus de projeccions** implicades en el pipeline de visualització,
 - com es defineixen els **plans de clipping**,
 - s'introdueixen els **algorismes d'eliminació de parts amagades**
 - i les bases dels algorismes de **rasterització**.
- En aquesta primera part del tema:
 - s'analitzaran les projeccions paral·leles i perspectives utilitzades a la visualització clàssica i els seus avantatges i inconvenients
 - es definirà el volum de visió que determina els plans de *clipping* (o de retallat)
 - S'estudiarà com es simplifiquen en la implementació en **OpenGL**, definint la matriu ***projection***
 - Es detalla com s'implementen tots els passos en la **GPU**.

Índex

4.1. Introducció

4.2. Viewing: Projeccions

4.3. Volum de visió: definició del clipping

4.4. Normalització

4.5. Implementació en OpenGL

4.6. Implementació en GPUs

4.7. Clipping: mètodes

4.8. Rasterització

4.9. Eliminació de parts amagades

4.10. Implementació en OpenGL i en GPUs



Índex

4.1. Introducció

4.2. Viewing: Projeccions

4.3. Volum de visió: definició del clipping

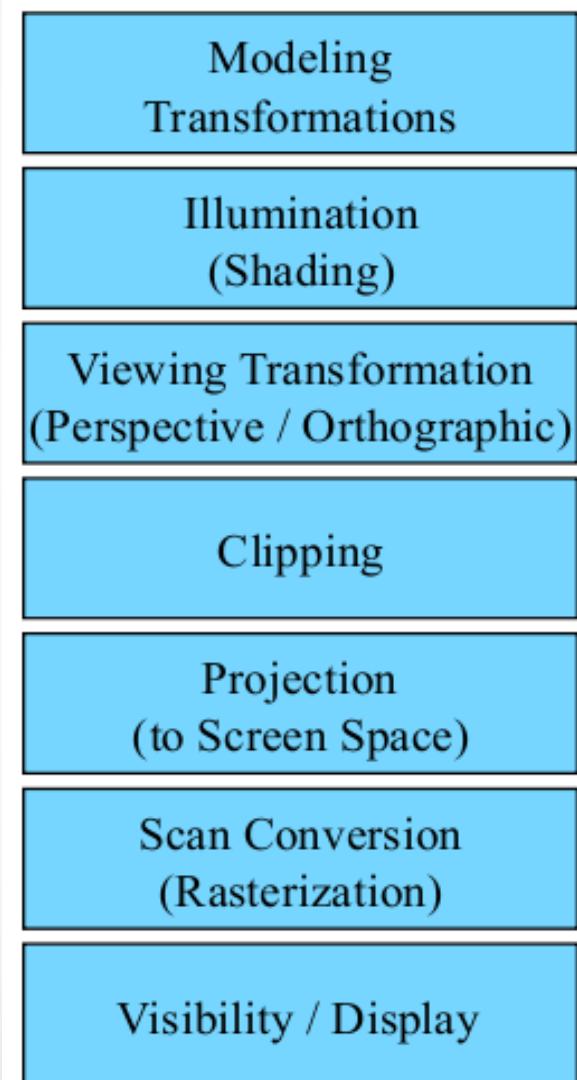
4.4. Normalització

4.5. Implementació en OpenGL

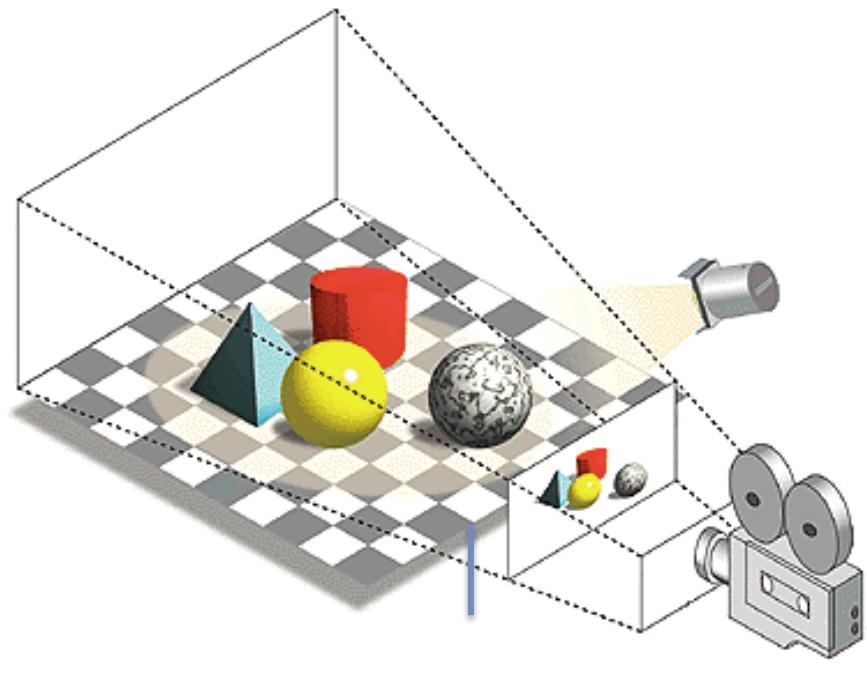
4.6. Implementació en GPUs



Pipeline de visualització



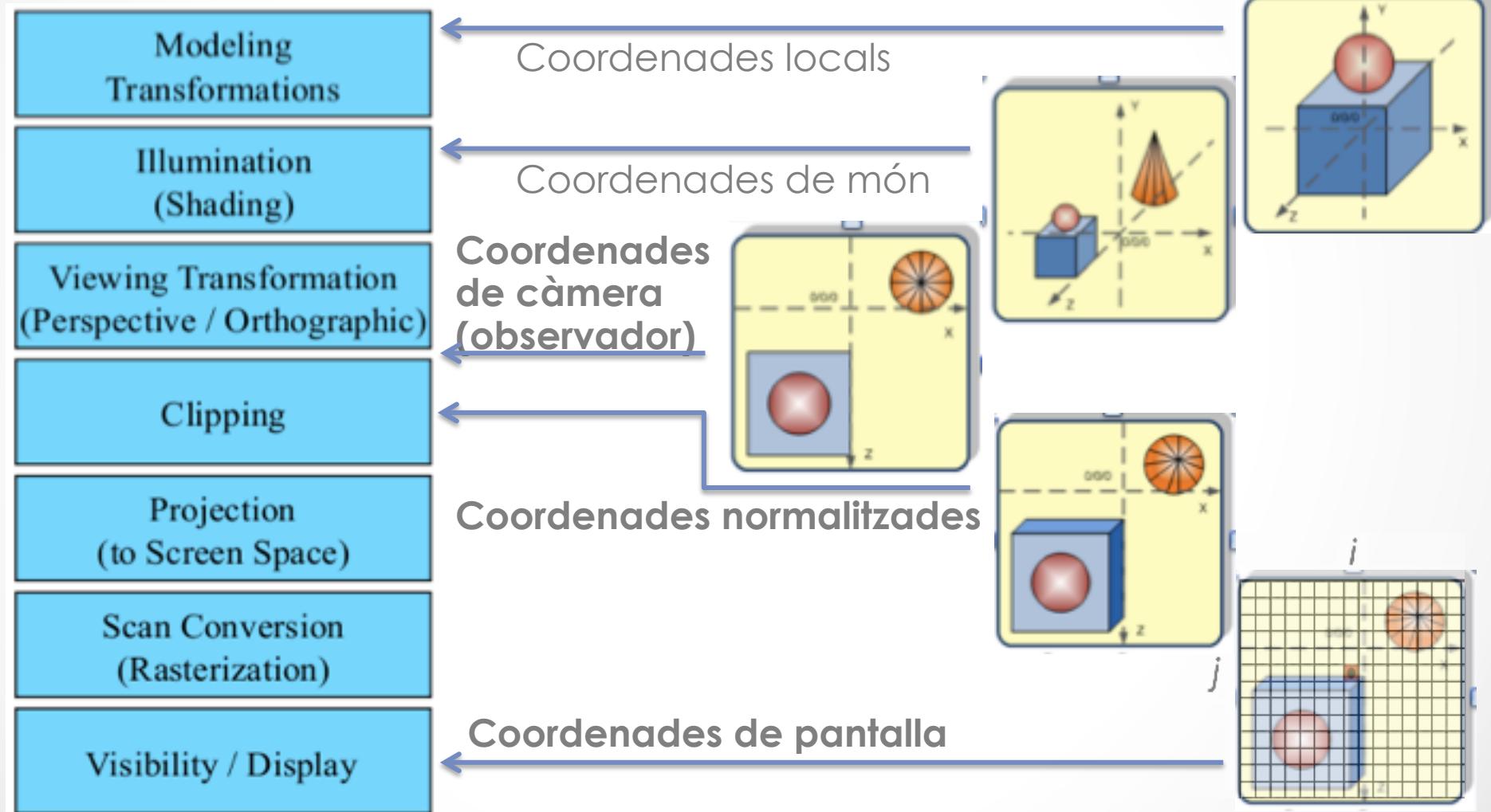
- **Input:** Objectes, Llum, Càmera i viewport



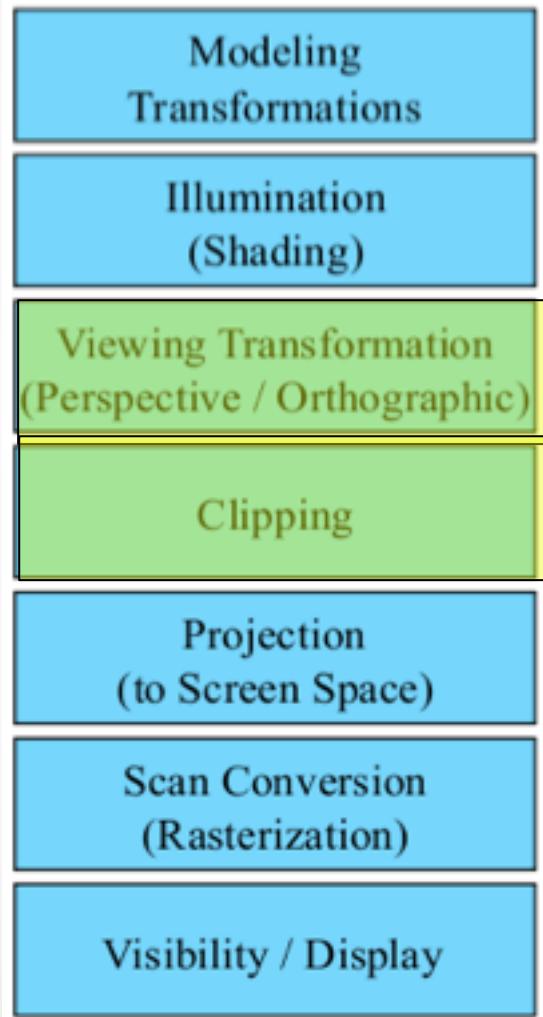
- **Output:** Frame buffer
(Colors/Intensitats (ex. 24-bit RGB a cada píxel))

Pipeline de visualització

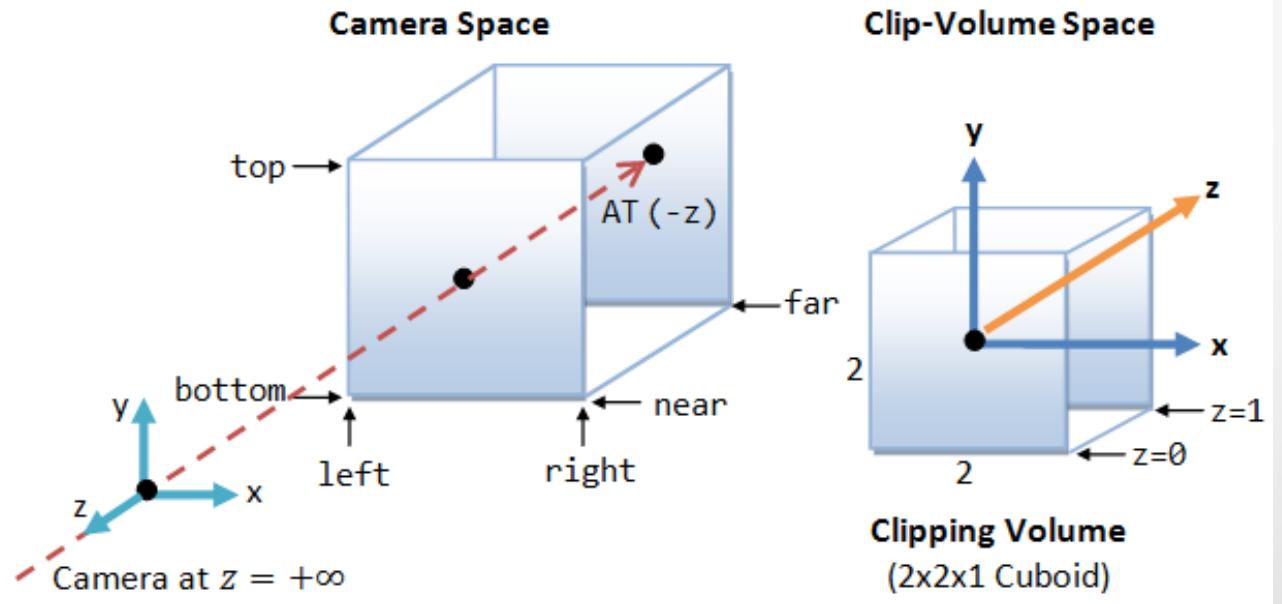
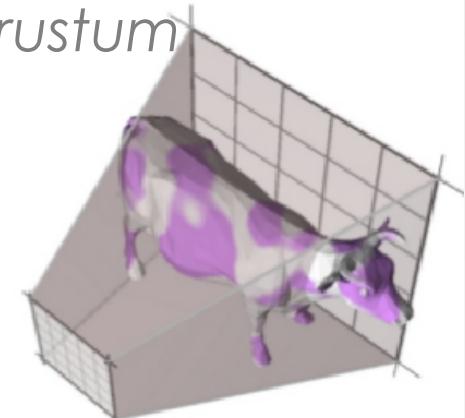
Diferents sistemes de coordenades a diferents etapes:



Pipeline de visualització

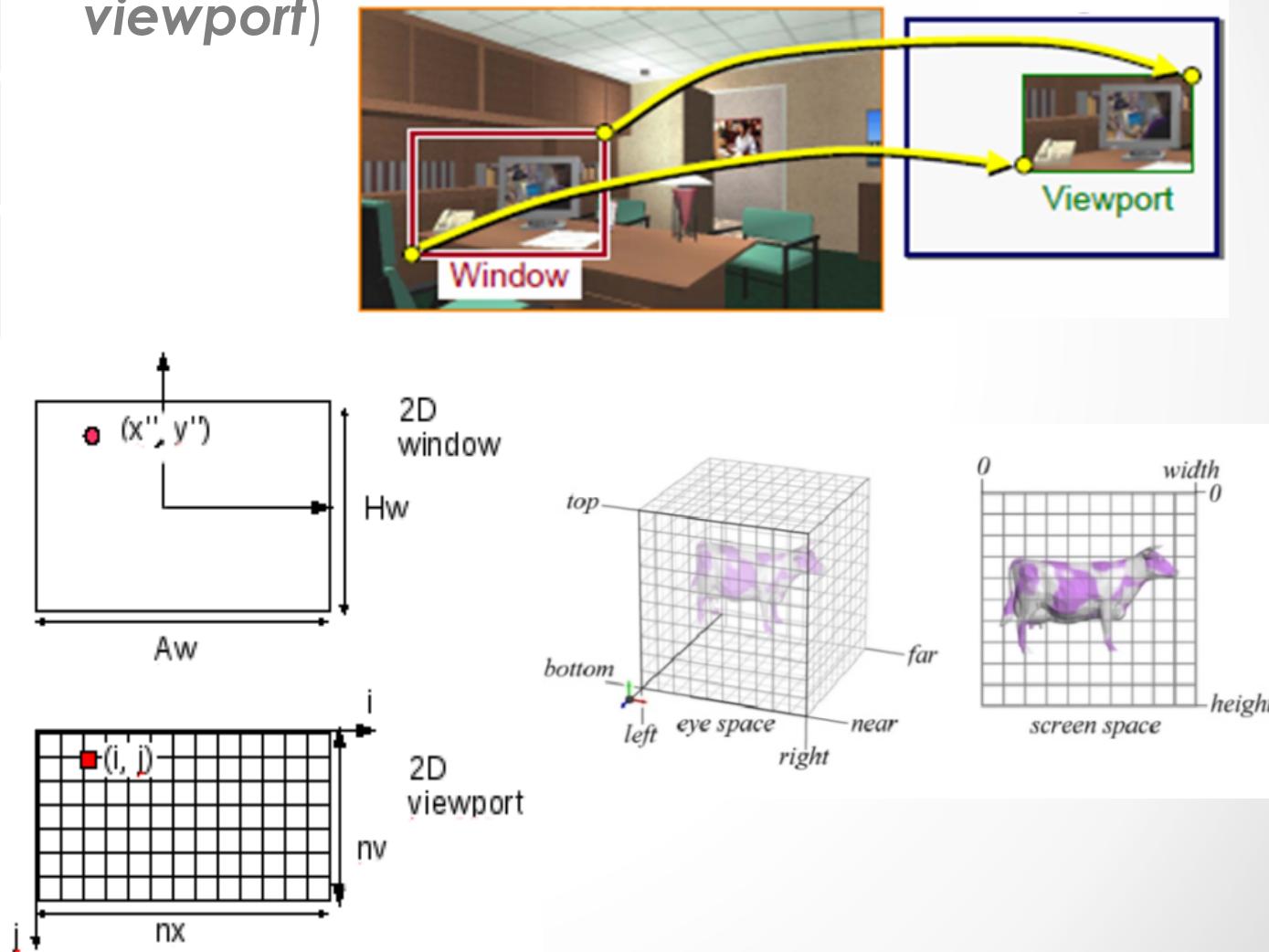
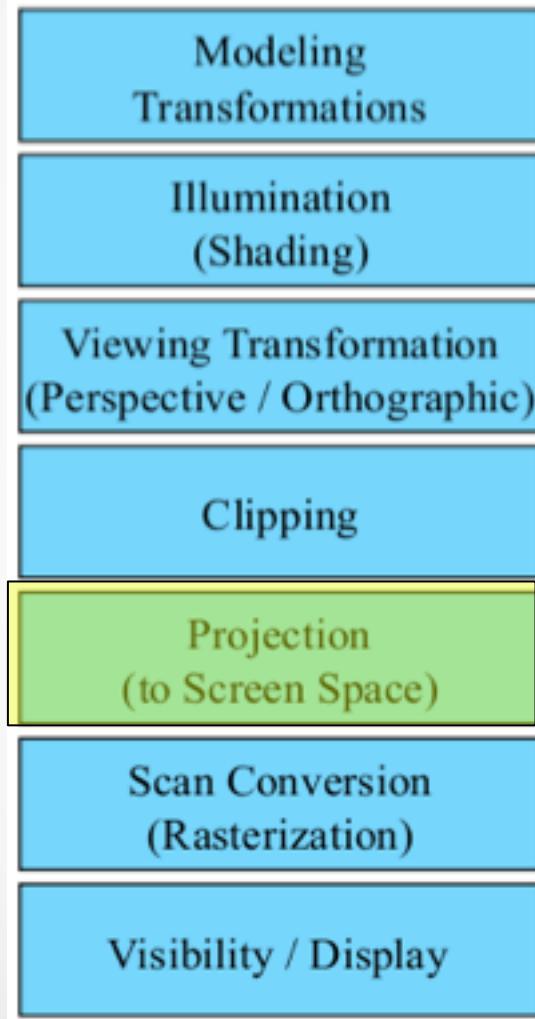


- Es **normalitzen** les coordenades dels vèrtexs
- Tots els objectes de fora del *frustum* s'eliminen per clipping

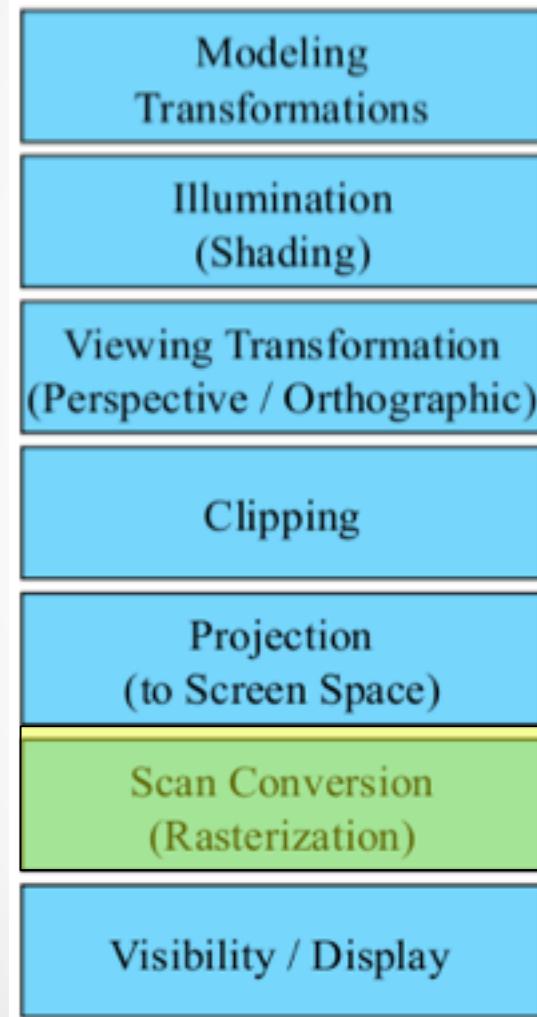


Pipeline de visualització

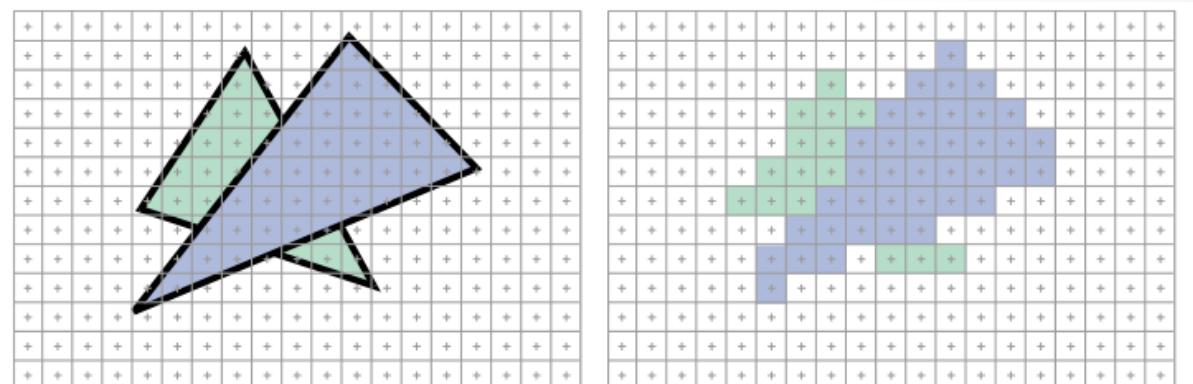
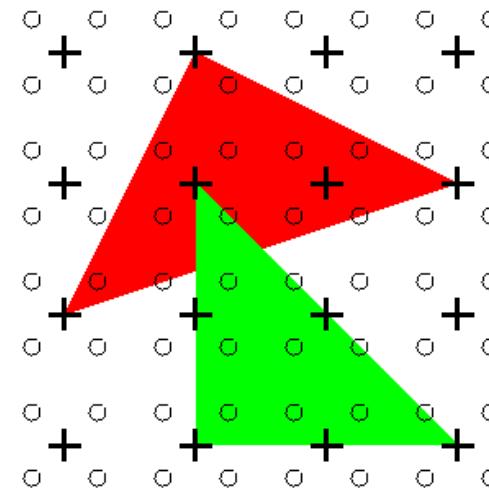
- Es projecta cada vèrtex de l'espai 2D continu (**window**) al frame buffer (o **viewport**)



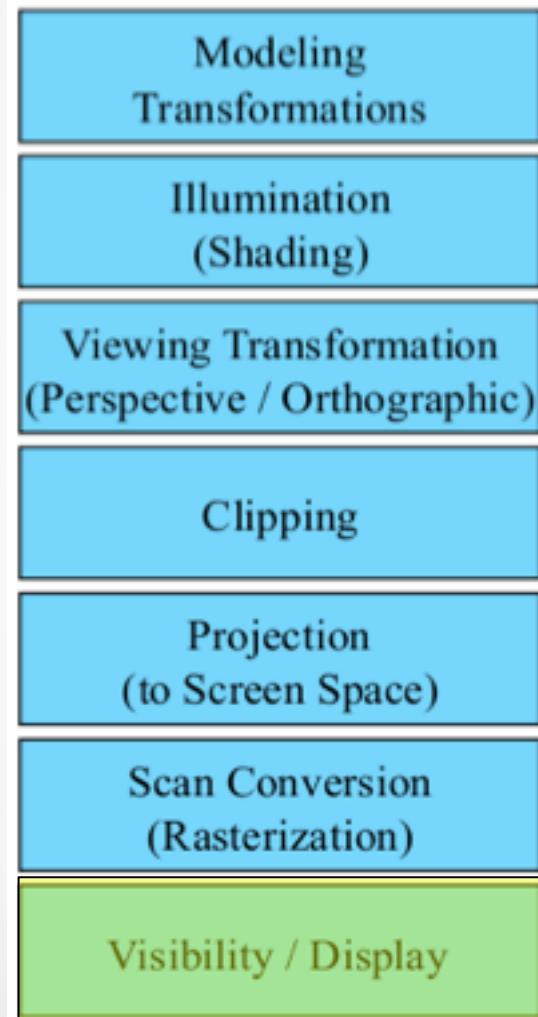
Pipeline de visualització



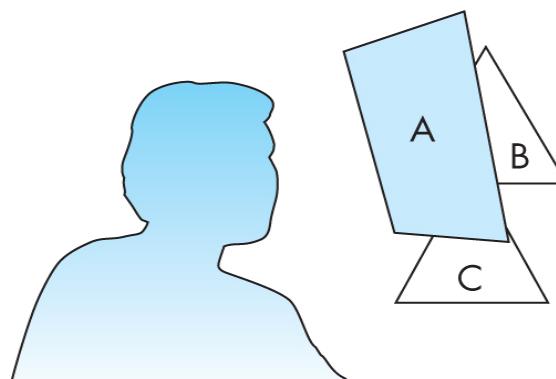
- Es rasteritzen les àrees definides pels punts projectats (**scan conversion**)



Pipeline de visualització



- De tots els objectes projectats en el mateix píxel, es mostra només el que està més proper a l'observador (**depth-buffer**: s'anomena al buffer que guarda de la profunditat més propera a l'observador). És el mètode de visibilitat o **d'eliminació de parts amagades**.



Índex

4.1. Introducció

4.2. Viewing: Projeccions

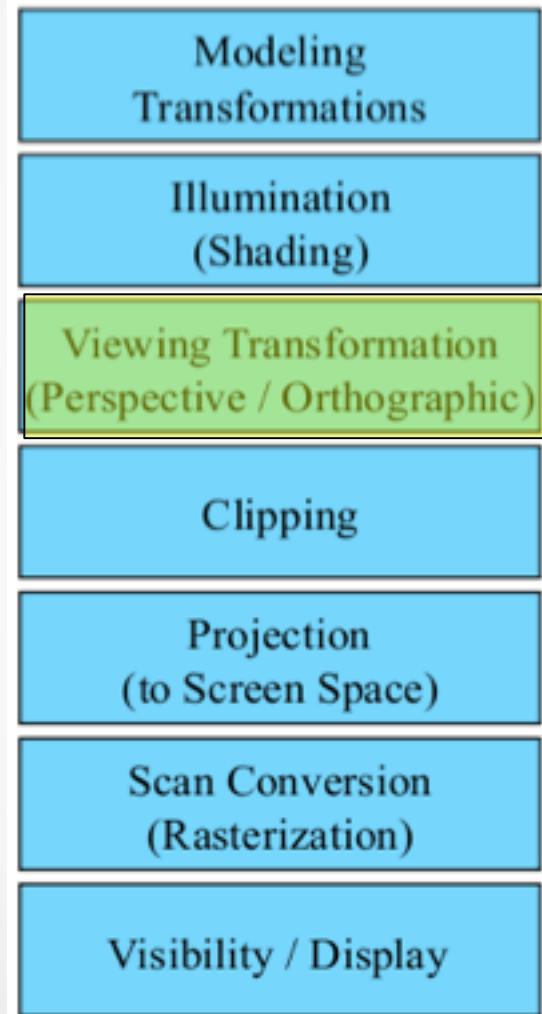
4.3. Clipping

4.4. Normalització

4.5. Implementació en OpenGL

4.6. Implementació en GPUs

4.2. Viewing



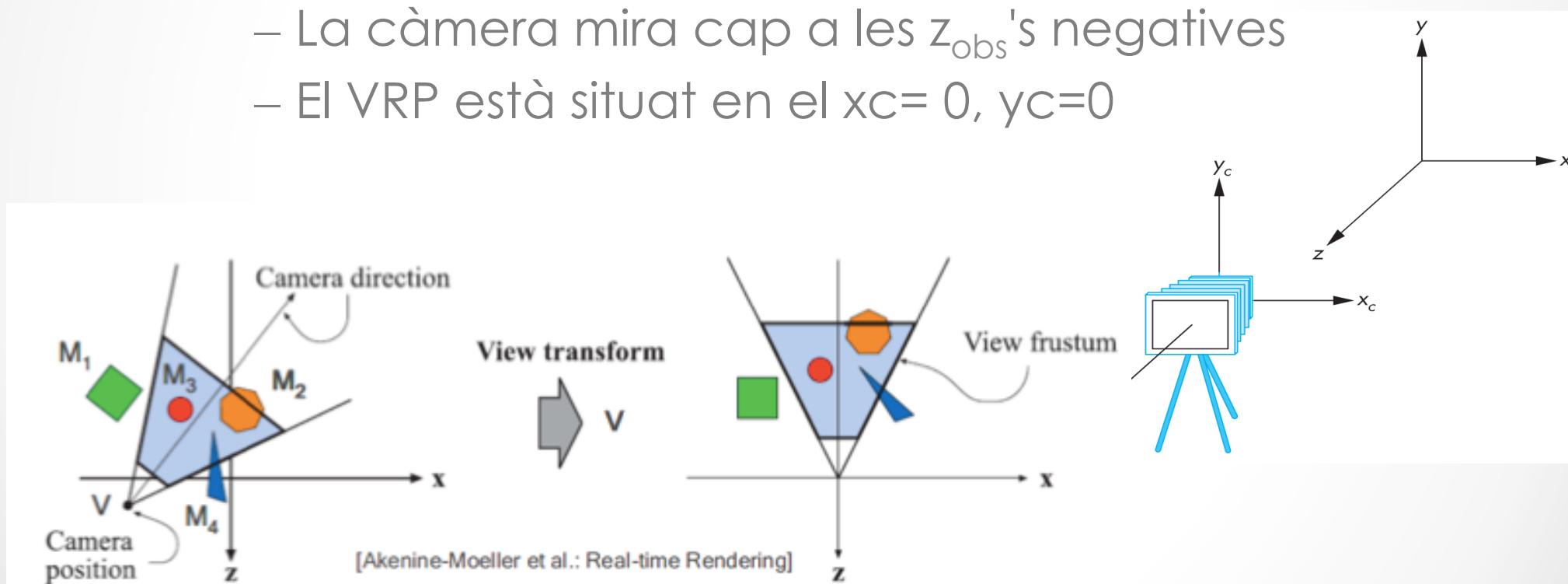
La transformació de visualització es composa de dos passos:

- 1 - el càlcul de la TG per a passar de coordenades de món a coordenades de càmera (**matriu model-view**)
- 2 - el càlcul de la TG per a realitzar la projecció de les coordenades 3D a coordenades 2D (**matriu projection**)

4.2. Viewing

1. Una vegada s'han definit els paràmetres de la càmera i s'ha realitzat la transformació amb la matriu **model-view**, s'ha de complir que:

- La càmera mira cap a les z_{obs} 's negatives
- El VRP està situat en el $x_c=0$, $y_c=0$



4.2. Viewing: Projeccions

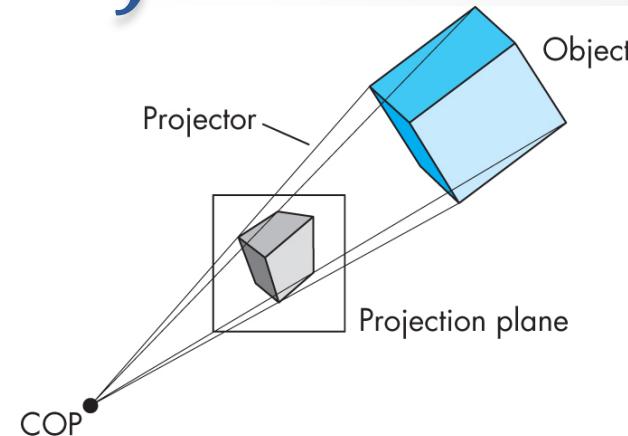
2. El càlcul de la TG per a realitzar la projecció de les coordenades 3D a coordenades 2D (**matriu de projecció**) es defineix com:

- Aquest càlcul es fa definint la piràmide de visió
- La piràmide de visió vé definida per:
 - El tipus de projecció
 - El volum de visió (*clipping volume*)
 - El pla de projecció
- Aquestes especificacions permeten definir la matriu de **projecció**
- Aquesta matriu es concatenarà per l'esquerra a la matriu **model view**

4.2. Viewing: projeccions

Una **projecció** es defineix per:

- un o més objectes
- un observador amb una superfície de projecció
- una funció de projecció (**o projector**) que va des de l'objecte(s) fins la superfície de projecció



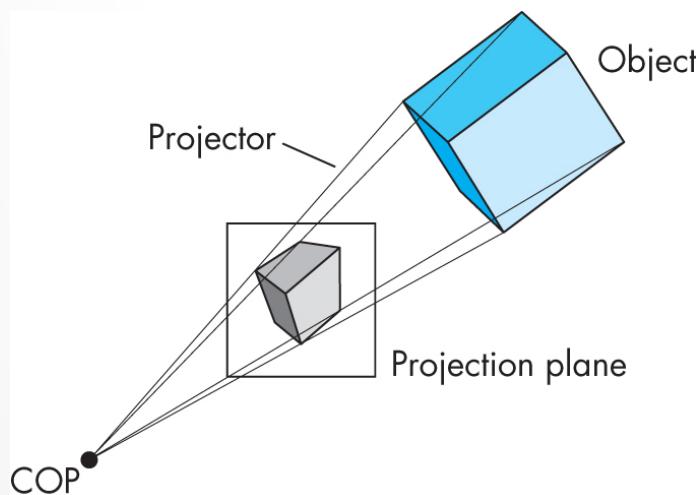
Les vistes clàssiques utilitzades en arquitectura, disseny, etc. estan basades en les relacions entre aquests elements

Es suposa que cada objecte es construeix a partir de les seves cares principals (edificis, poliedres, objectes manufacturats, etc.)

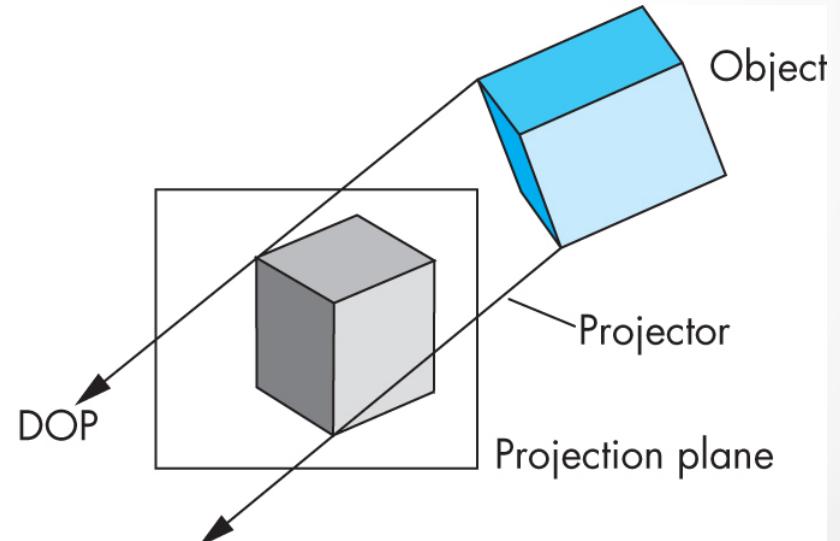
4.2. Viewing: projeccions

Les projeccions estàndard projecten en un pla: són les **projeccions geomètriques planars**.

- Els projectors són **rectes**
- Són projeccions que preserven les rectes però no necessàriament els angles

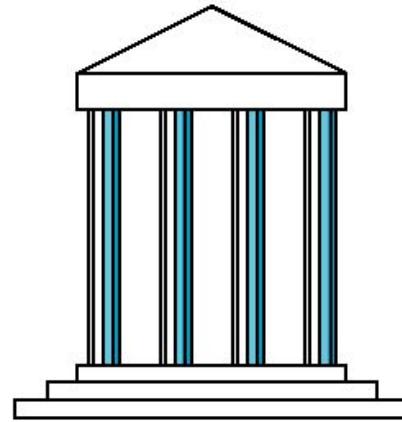


COP: Centre de projecció

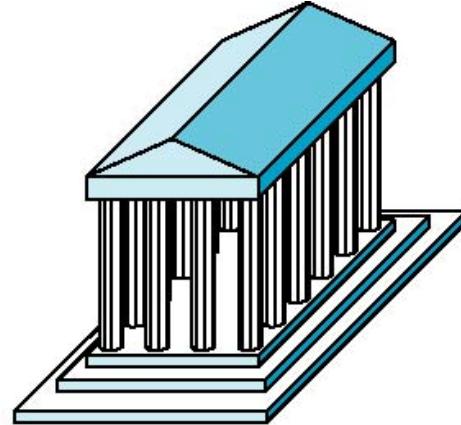


DOP: Direcció de projecció

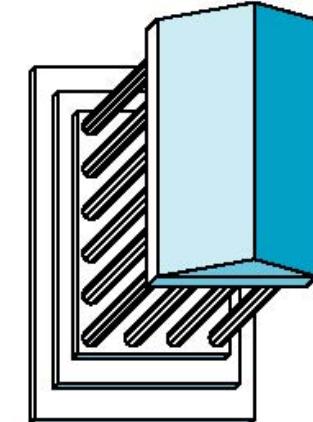
4.2. Projeccions clàssiques



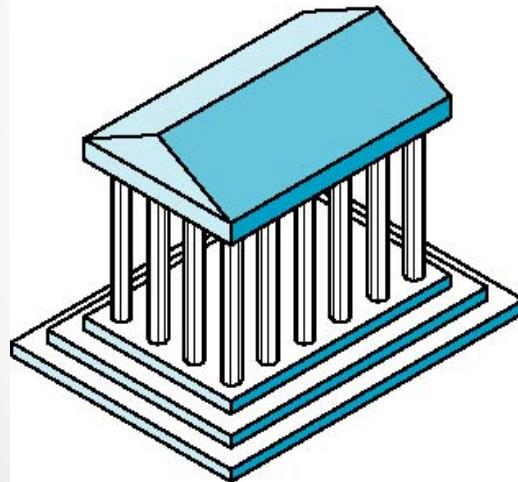
Front elevation



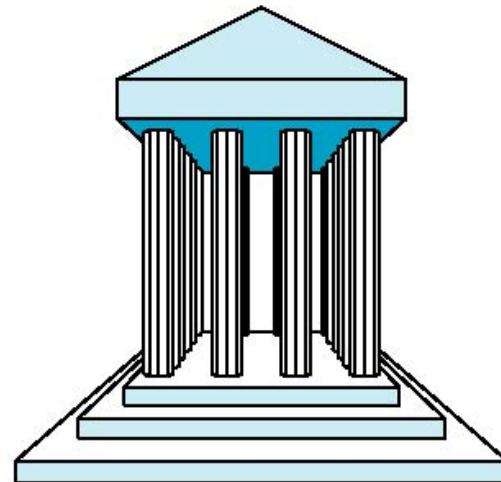
Elevation oblique



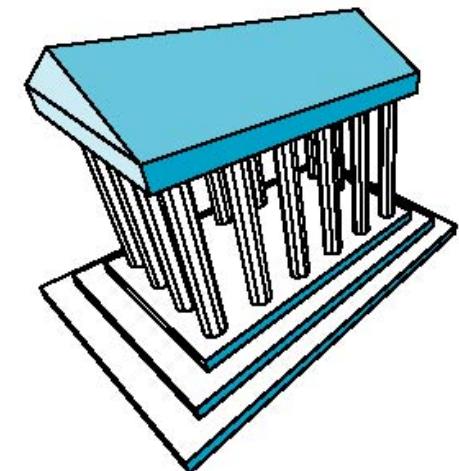
Plan oblique



Isometric



One-point perspective

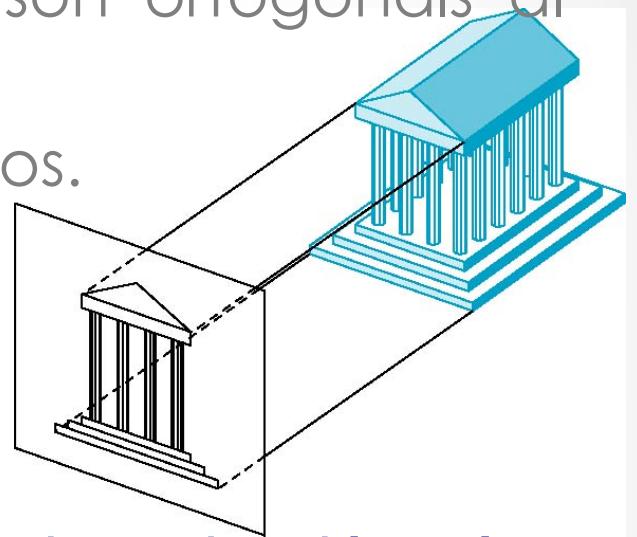
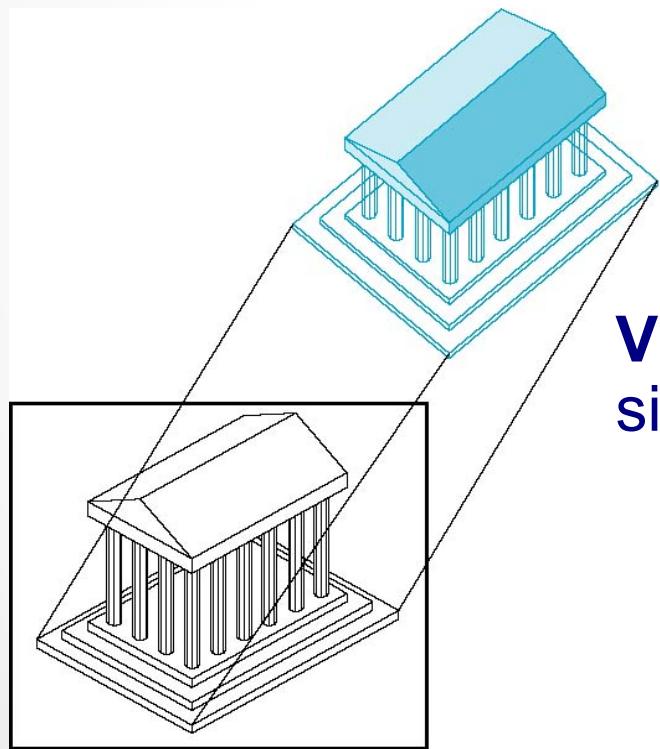


Three-point perspective

4.2. Projeccions paral·leles ortogonals

Projeccions ortogonals: els projectors són ortogonals al pla de projecció.

- **Projeccions ortogràfiques:** Segons els eixos.

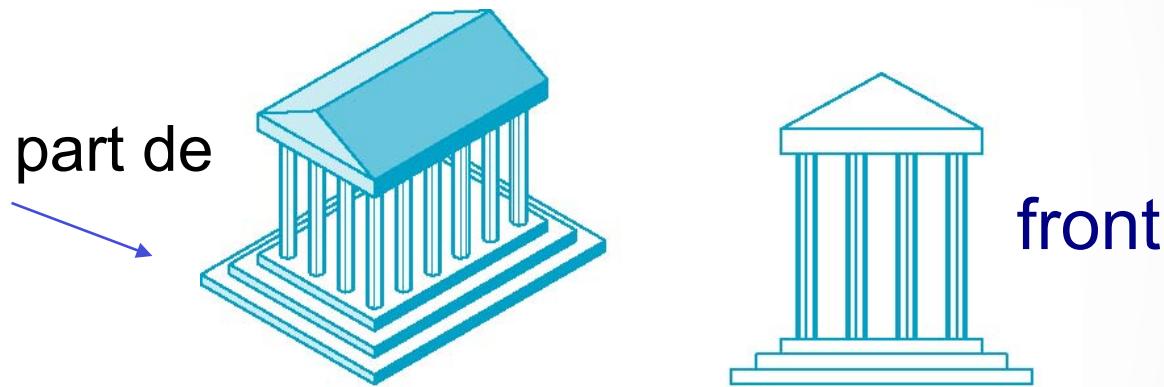


Vista isomètrica: el pla de projecció està situat simètricament en relació als eixos.

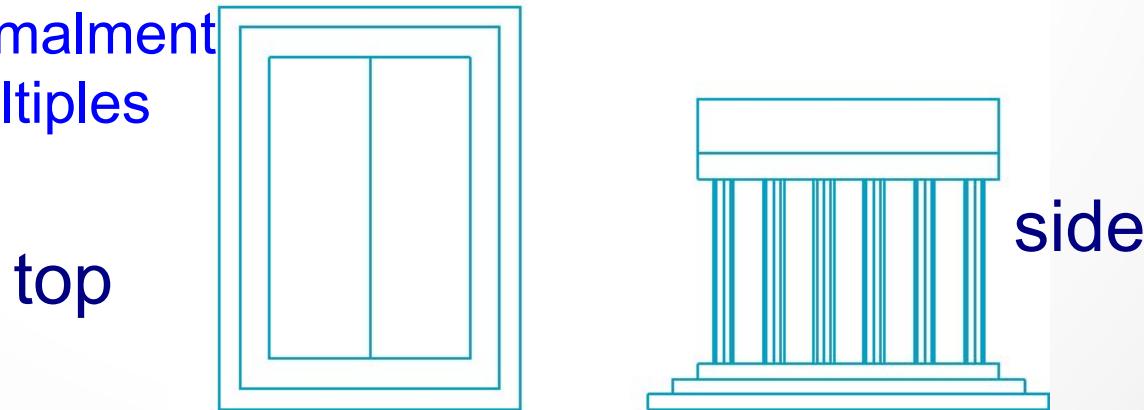
4.2. Projeccions paral·leles ortogonals

Projecció ortogràfica amb vistes múltiples: Pla de projecció paral·lel a la cara principal (front, top, side)

isomètrica (no forma part de les vistes múltiples)



En aplicacions CAD, normalment es mostren les vistes múltiples més la isomètrica



4.2. Projeccions paral·leles

Projecció ortogràfica: matriu projecció

- Es projecta al pla $z = 0$, simplement posant $z=0$.
- Equivalent a la transformació en coordenades homogènies següent

$$\mathbf{M}_{\text{orth}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

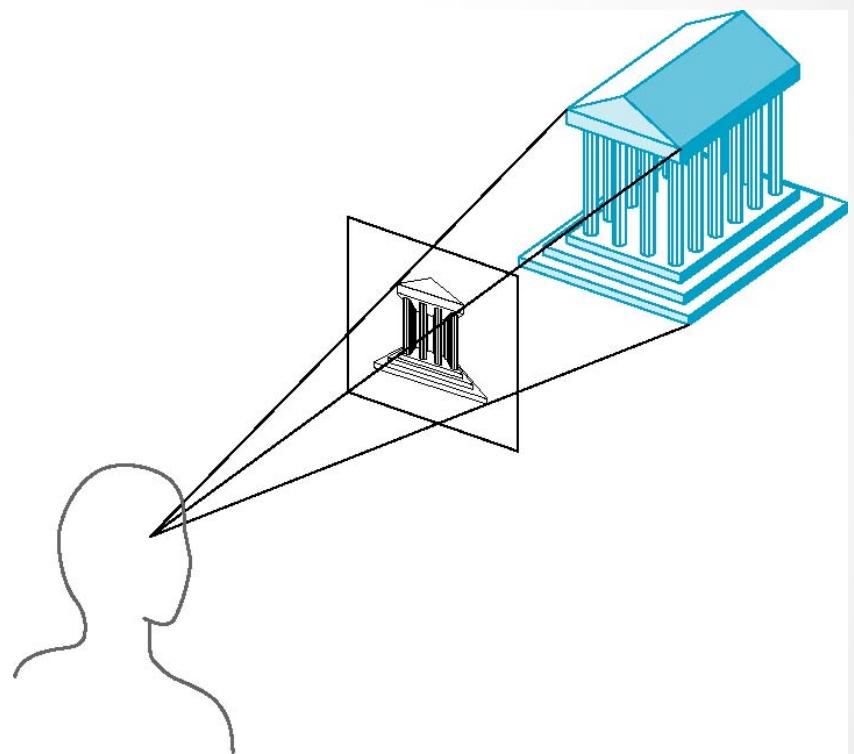
- La projecció ortogràfica directa és:

$$\mathbf{P} = \mathbf{M}_{\text{orth}}$$

4.2. Projeccions perspectives

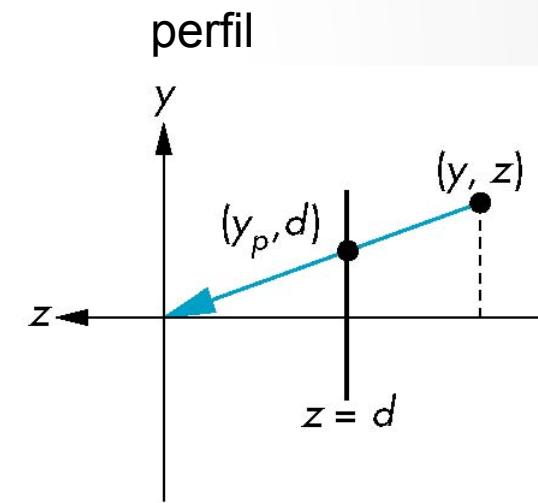
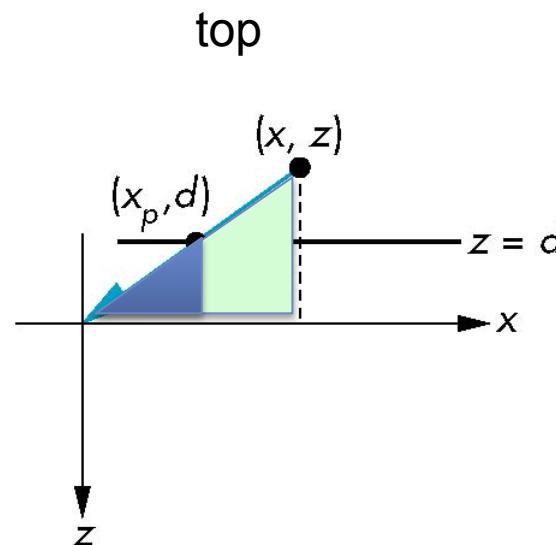
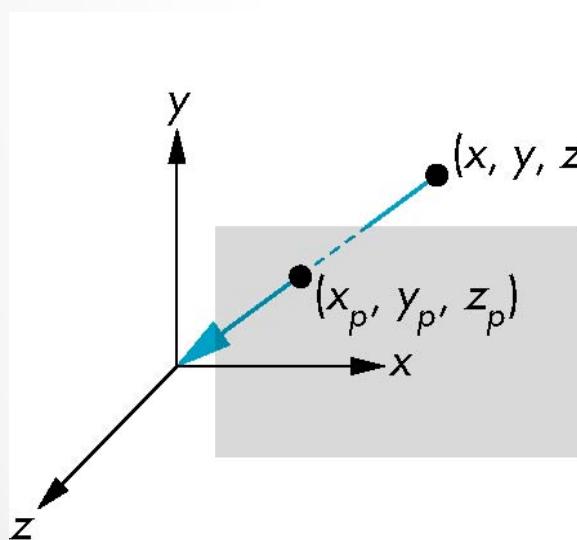
Projeccions perspectives: els projectors convergeixen al centre de projecció

- Els objectes més llunyans a l'observador es projecten més petits
- Realisme
- Distàncies iguals entre els objectes no es projecten en distàncies iguals
- Només es preserven els angles paral·lels al pla de projecció
- Són més costoses de construir que les projeccions paral·leles



4.2. Projeccions perspectives

Definició de la matriu projecció: Per una projecció **perspectiva**, si tenim el centre de projecció a l'origen i tenim el pla de projecció a distància **d** de la càmera ($z=d$, $d<0$), la projecció perspectiva d'un punt es calcularia com:



$$x_p = \frac{x}{z/d}$$

$$y_p = \frac{y}{z/d}$$

$$z_p = d$$

4.2. Projeccions perspectives

Aquest càlcul, posat en forma de matrius, és:

$$\text{Si } \mathbf{p} = \mathbf{M}\mathbf{q}, \text{ on } \mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \Rightarrow \mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

Per obtenir coordenades homogènies de la forma $(x,y,z,1)$ cal que fem la divisió de cada un dels components del punt per z/d

$$x_p = \frac{x}{z/d} \quad y_p = \frac{y}{z/d} \quad z_p = d$$

Índex

4.1. Introducció

4.2. Viewing: Projeccions

4.3. Volum de visió: definició del *clipping*

4.4. Normalització

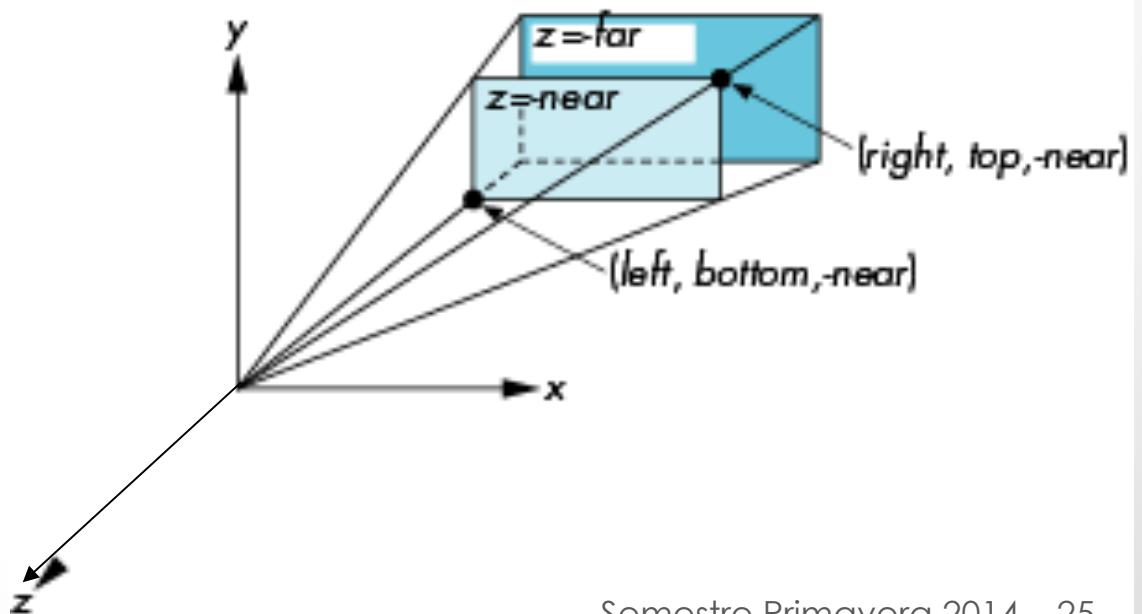
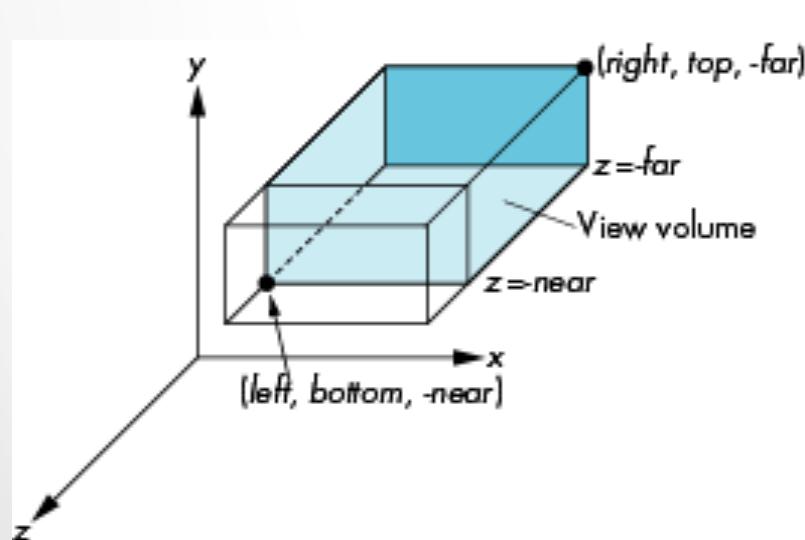
4.5. Implementació en OpenGL

4.6. Implementació en GPUs

4.3. Volum de visió

Els **plans de retallat** o de *clipping* defineixen el volum visible de l'escena a visualitzar:

- el volum visible en projeccions **paral·leles** és un paral·lepípede: un ortoedre en projeccions ortogràfiques.
- el volum visible en projeccions **perspectives** és una piràmide truncada.



4.3. Volum de visió

Es defineixen dos plans addicionals paral·lels al pla de la càmera:

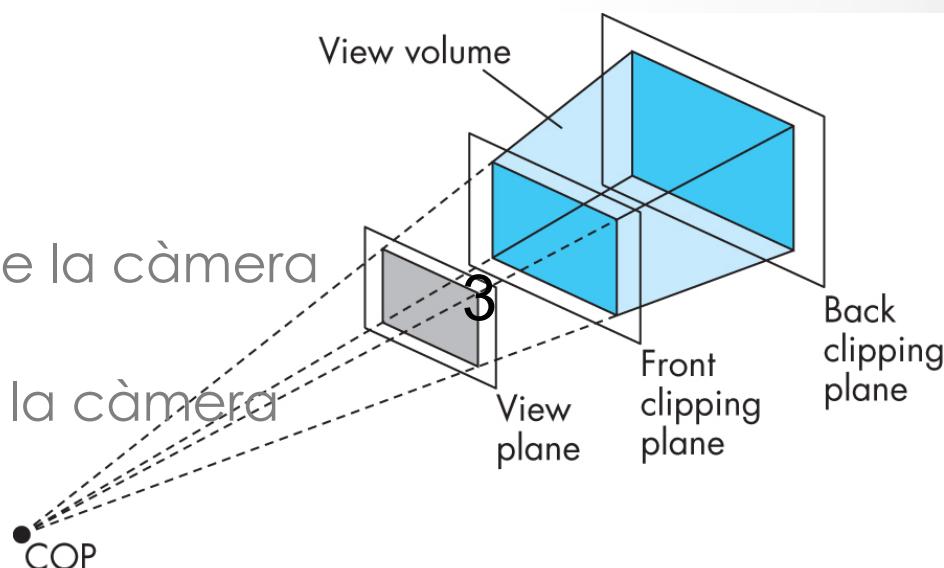
- el pla de retallat anterior (z_{near})
- el pla de retallat posterior (z_{far})

z_{near} i z_{far} es mesuren com la distància (amb signe) dels plans al pla de la càmera.

Exemple:

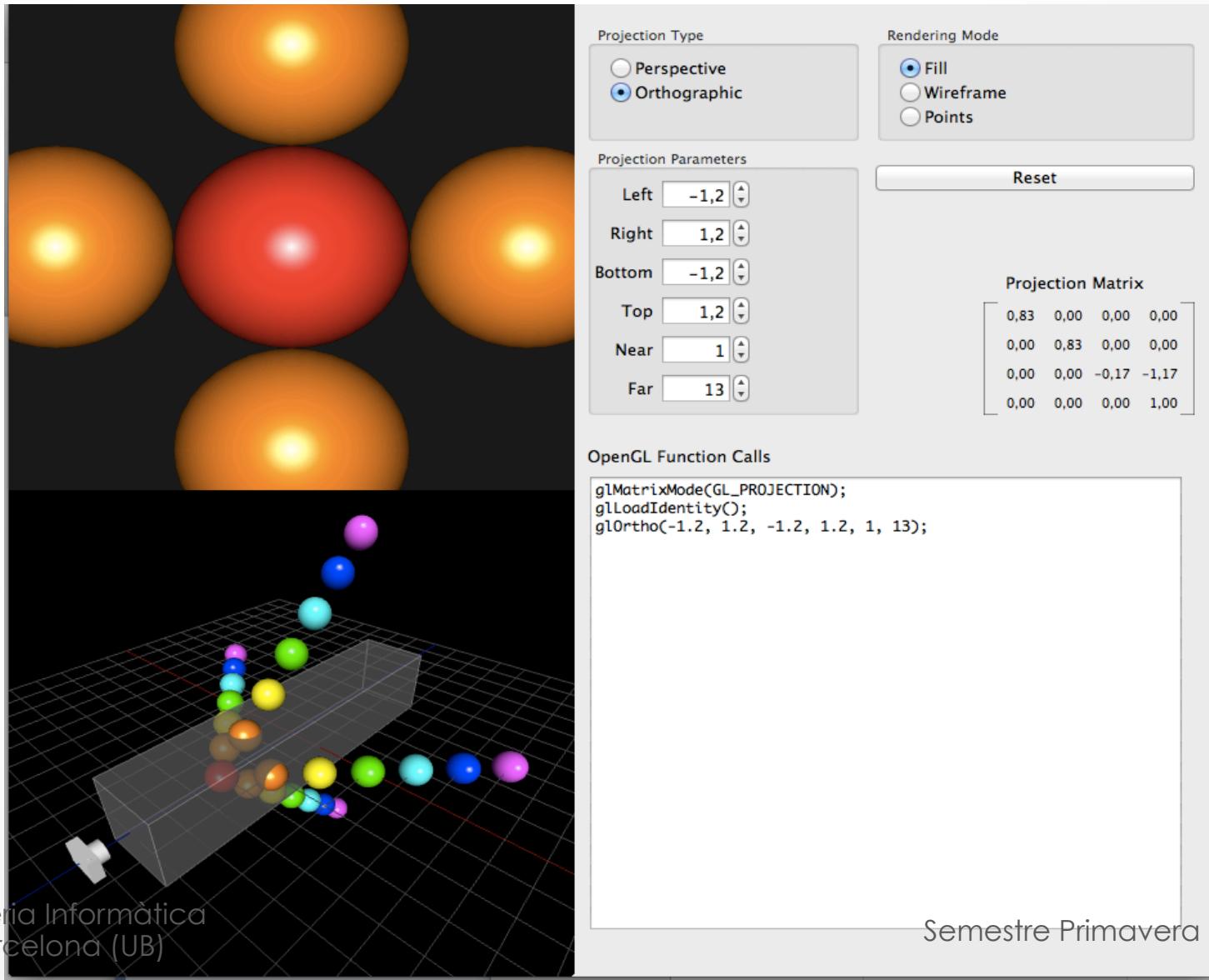
$z_{\text{near}} = 1$ pla anterior està davant de la càmera

$z_{\text{far}} = 2$ pla posterior està davant de la càmera



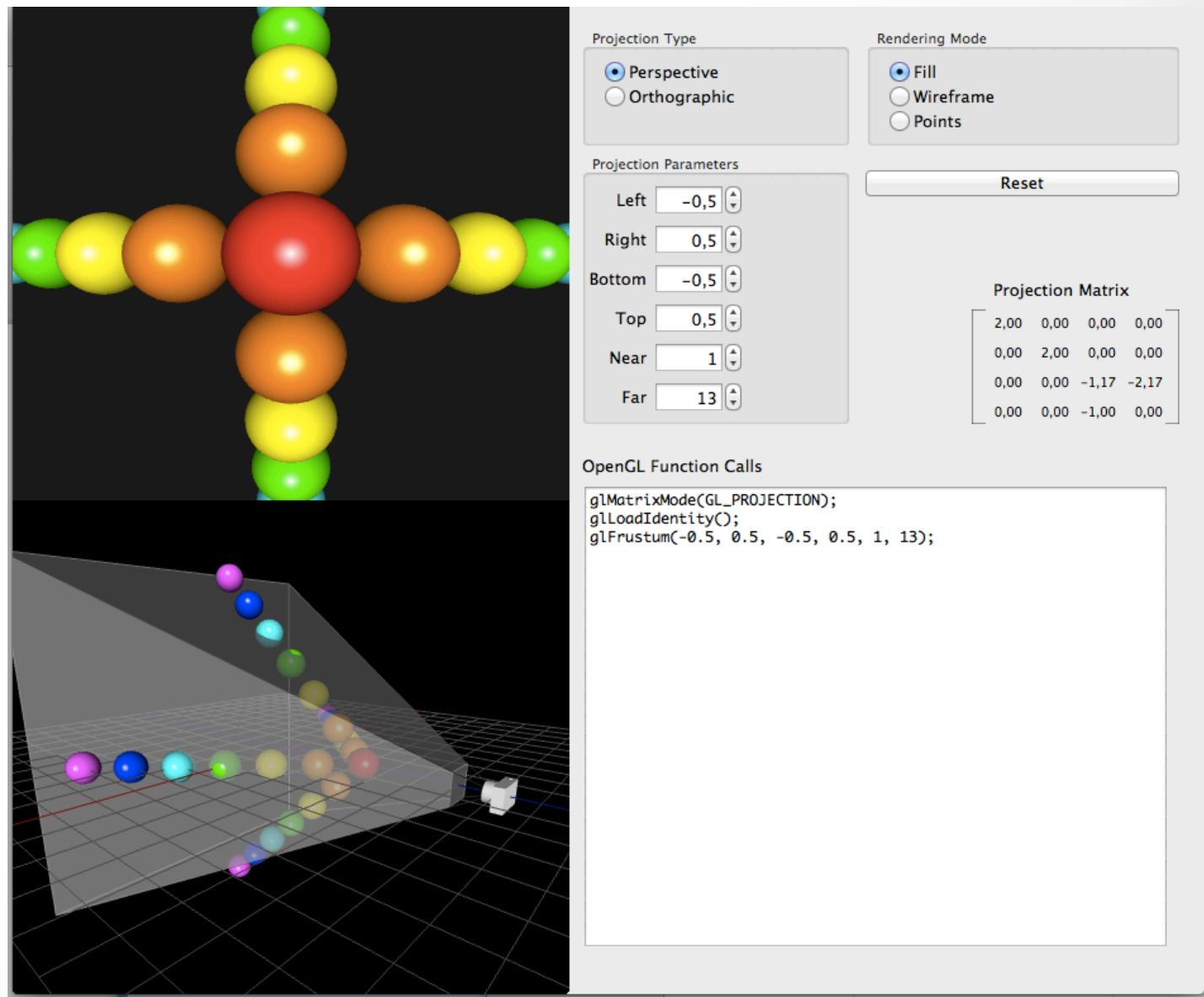
4.3. Volum de visió

Projeccions ortogonals:



4.3. Volum de visió

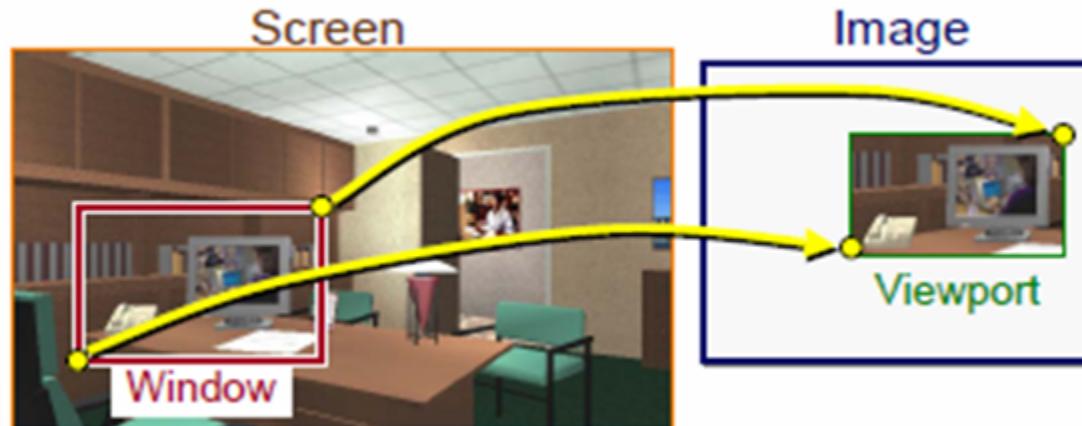
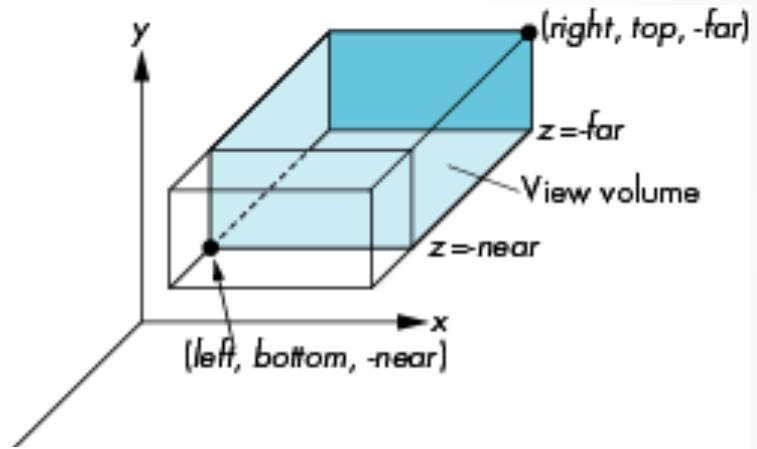
Projeccions
perspectives:



4.3. Volum de visió

Com es defineixen els límits del volum de visió?

concepte de **window**: àrea 2D en el pla de projecció de la zona visible centrada en el 0,0.



Índex

4.1. Introducció

4.2. Viewing: Projeccions

4.3. Volum de visió: definició del *clipping*

4.4. Normalització

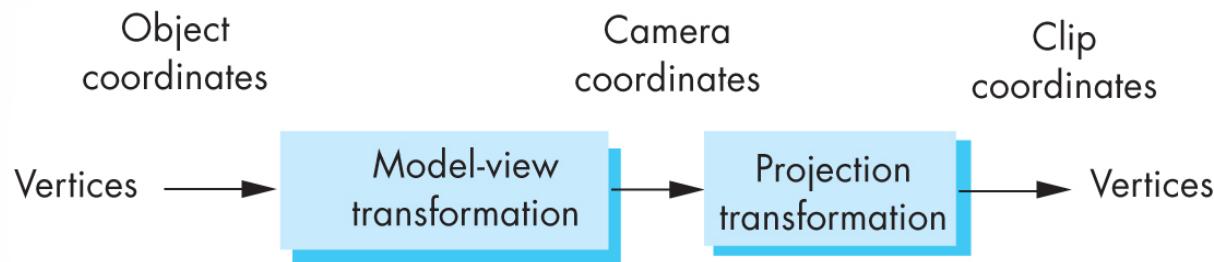
4.5. Implementació en OpenGL

4.6. Implementació en GPUs

4.4. Normalització

Els vèrtexs es defineixen en coordenades homogènies i es transformen amb les matrius **model view** i **projection**.

Per defecte són les matrius identitat (que definirien una vista ortogonal)



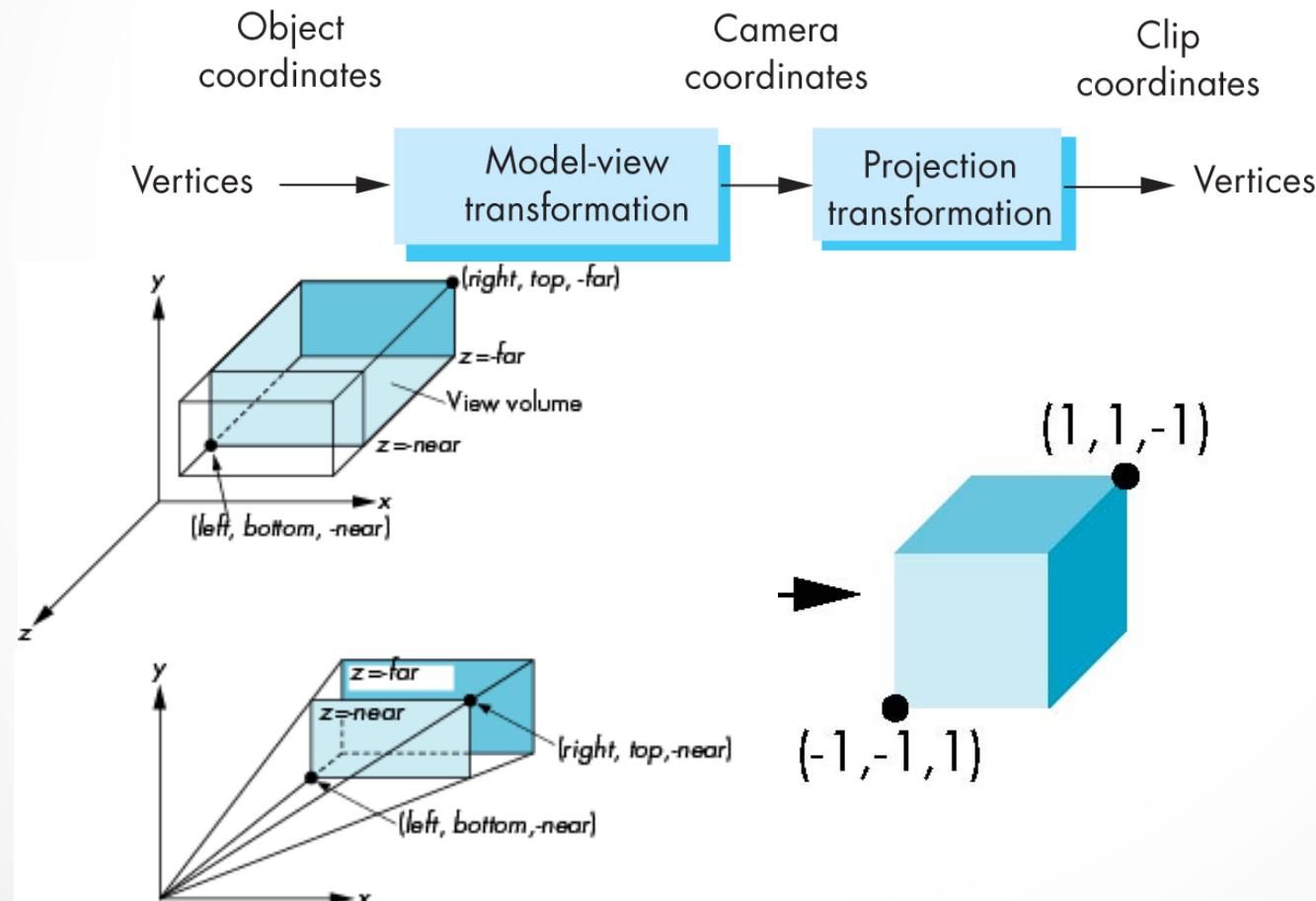
Per a calcular totes els tipus de projeccions dins del mateix *pipeline*, es fa un procés de **normalització** que:

- fa el clipping contra un cub simple, independentment del tipus de projecció
- permet realitzar la projecció final a coordenades 2D al final del *pipeline*

Aquest fet és important ja que per l'eliminació de parts amagades és necessari conservar la informació de profunditat tant com es pugui.

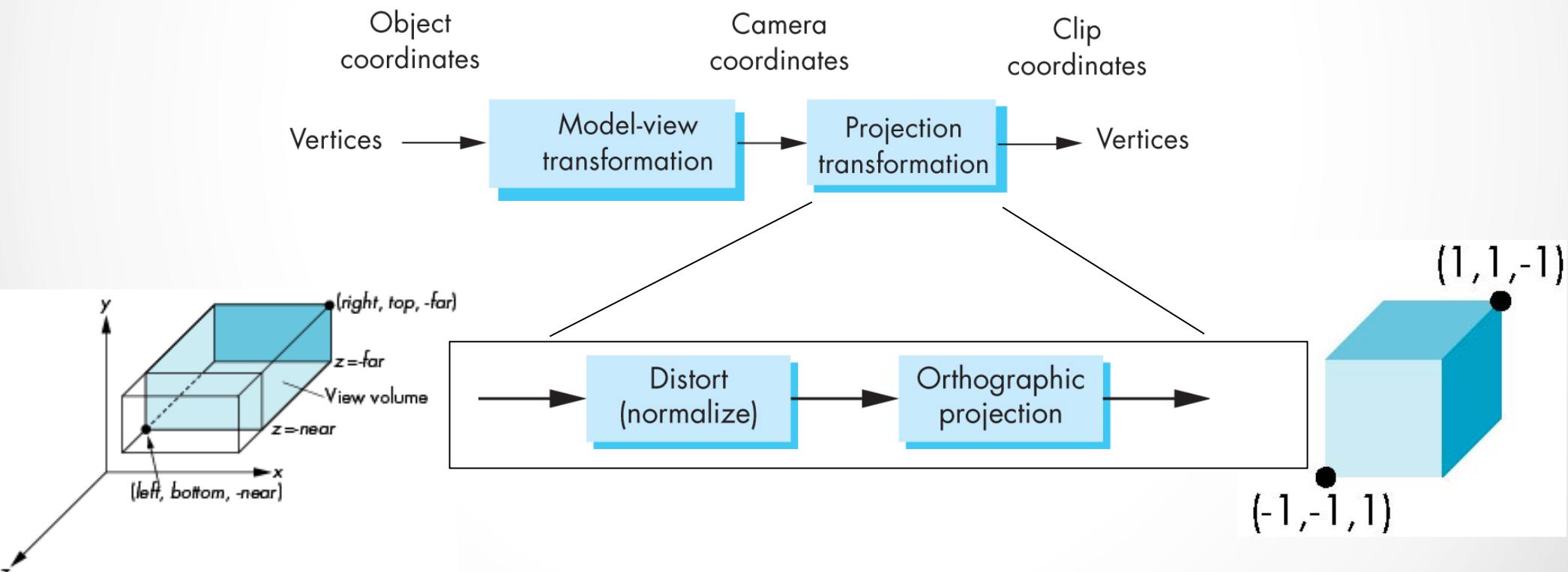
4.4. Normalització

La **normalització** és una transformació que permet convertir el volum de visió en un cub centrat a l'origen i d'aresta 2



4.4. Normalització

La **normalització** és una transformació que permet convertir el volum de visió en un cub centrat a l'origen i d'aresta 2

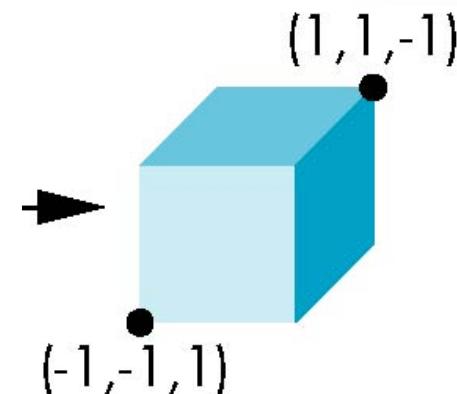


▶ <http://cs.brown.edu/courses/cs123/demos/camera/>

4.4. Normalització

El volum normalitzat en projeccions paral·leles:

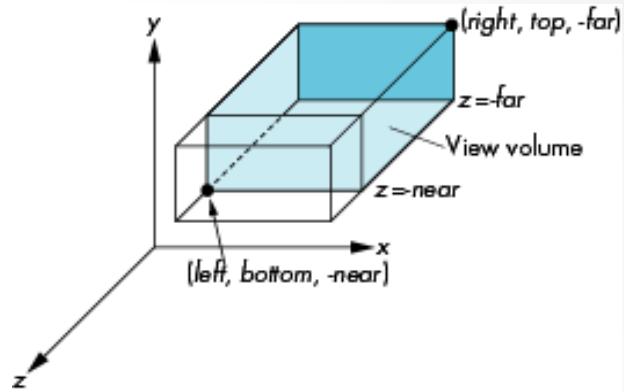
- Centre del pla anterior de clipping $(0, 0, 1)$
- La càmera mira cap a l'eix de les z's negatives:
 - $\text{Look} = (0,0,-1)$
- VUP en vertical cap a les y's positives
 - $\text{VUP} = (0,1,0)$
- Window normalitzada a :
 - -1 a 1 en direcció \mathbf{x} i \mathbf{y}
- Plans de clipping antero-posteriors:
 - Pla anterior de clipping $z = 1$
 - Pla posterior de clipping $z = -1$
- Després d'aplicar la normalització tots els vèrtexs de l'escena que cauen fora dels límits dels plans $x = (-1, 1)$, $y = (-1, 1)$, i $z = (-1, 1)$ s'eliminen.



4.4. Normalització

Projecció paral·lela:

1. - Es mou el centre a l'origen:



Translate $(-(\text{left}+\text{right})/2, -(\text{bottom}+\text{top})/2, (\text{near}+\text{far})/2)$)

2. - S'escala a un cub d'aresta 2

Scale $(2/(\text{left}-\text{right}), 2/(\text{top}-\text{bottom}), 2/(\text{near}-\text{far}))$

$$\mathbf{P} = \mathbf{ST} = \begin{bmatrix} \frac{2}{\text{right} - \text{left}} & 0 & 0 & \frac{\text{right} + \text{left}}{\text{right} - \text{left}} \\ 0 & \frac{2}{\text{top} - \text{bottom}} & 0 & \frac{\text{top} + \text{bottom}}{\text{top} - \text{bottom}} \\ 0 & 0 & \frac{2}{\text{near} - \text{far}} & \frac{\text{far} + \text{near}}{\text{far} - \text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.4. Normalització

Projecció ortogràfica: Projecció final

- Es projecta a $z = 0$, simplement posant $z=0$.
- Equivalent a la transformació en coordenades homogènies a:

$$\mathbf{M}_{\text{orth}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La projecció ortogràfica completa és:

$$\mathbf{P} = \mathbf{M}_{\text{orth}} \mathbf{ST}$$

4.4. Normalització

Projeccions paral·lela o ortogràfiques: (mat.h)

Ortho(left,right,bottom,top,near,far)

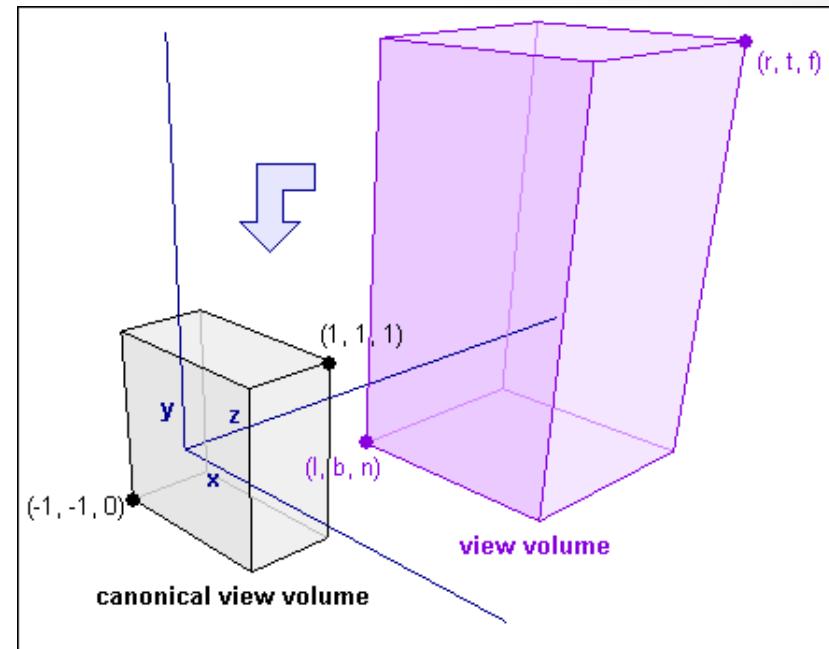
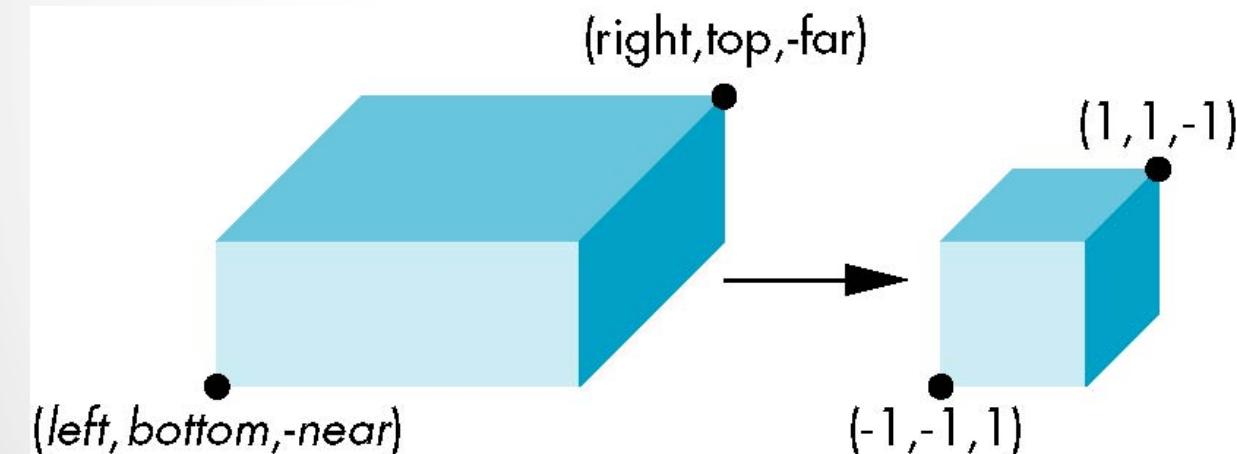
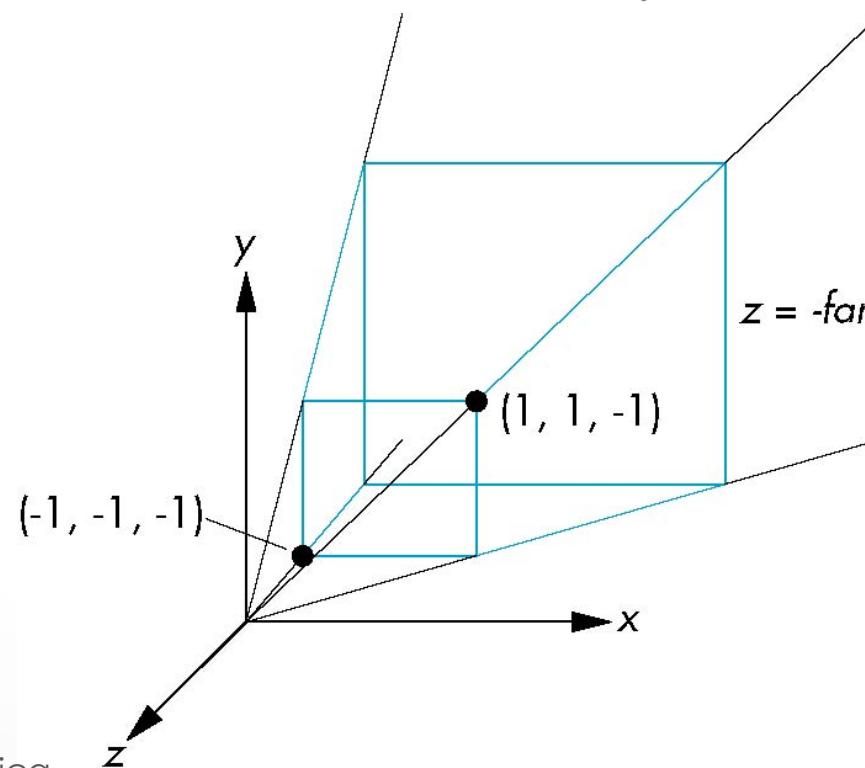


Image credit: Direct 3D
[http://www.codeguru.com/cpp/misc/mis...](http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123_2/)

4.4. Normalització

Projeccions perspectives: Cas simple

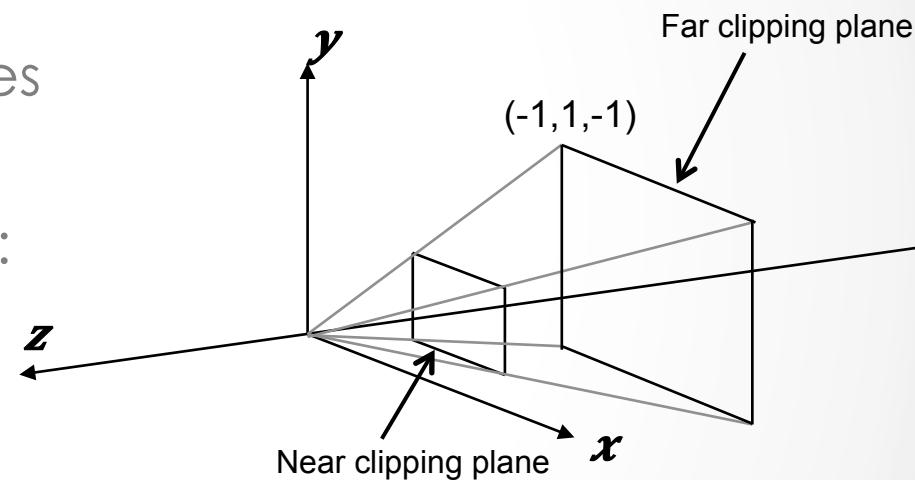
Considerem la projecció perspectiva següent amb el COP a l'origen, el pla de projecció és a $z = -1$ i un angle d'obertura de la càmera de 90° i simètric, determinat pels plans $x = \pm z$, $y = \pm z$



4.4. Normalització

El volum normalitzat en projeccions perspectives:

- El centre de pla anterior és al punt $(0,0,1)$
- La càmera mira cap a l'eix de les z's negatives:
 - $\text{Look} = (0,0,-1)$
- VUP en vertical cap a les y's positives
 - $\text{VUP} = (0,1,0)$
- Window al pla de clipping posterior:
 - -1 a 1 en direcció x i y
- Plans de clipping antero-posteriors:
 - Pla anterior de clipping $z = 1$
 - Pla posterior de clipping $z = -1$
- Després d'aplicar la normalització tots els vèrtexs de l'escena que cauen fora dels límits dels plans $x = (-1, 1)$, $y = (-1, 1)$, i $z = (-1, 1)$ s'eliminen.



4.4. Normalització

Projecció perspectiva: Cas simple amb z_{near} i z_{far}

$$\mathbf{N} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Aplicada al punt $(x, y, z, 1)$:

$$x' = x/z$$

$$y' = y/z$$

$$z' = -(\alpha + \beta/z)$$

$$\alpha = \frac{\text{near} + \text{far}}{\text{far} - \text{near}}$$

$$\beta = \frac{2\text{near} * \text{far}}{\text{near} - \text{far}}$$

El pla anterior ($z=\text{near}$) serà $z' = 1$
El pla posterior ($z=\text{far}$) serà $z' = -1$
Els plans laterals, $x' = \pm 1, y' = \pm 1$

4.4. Normalització

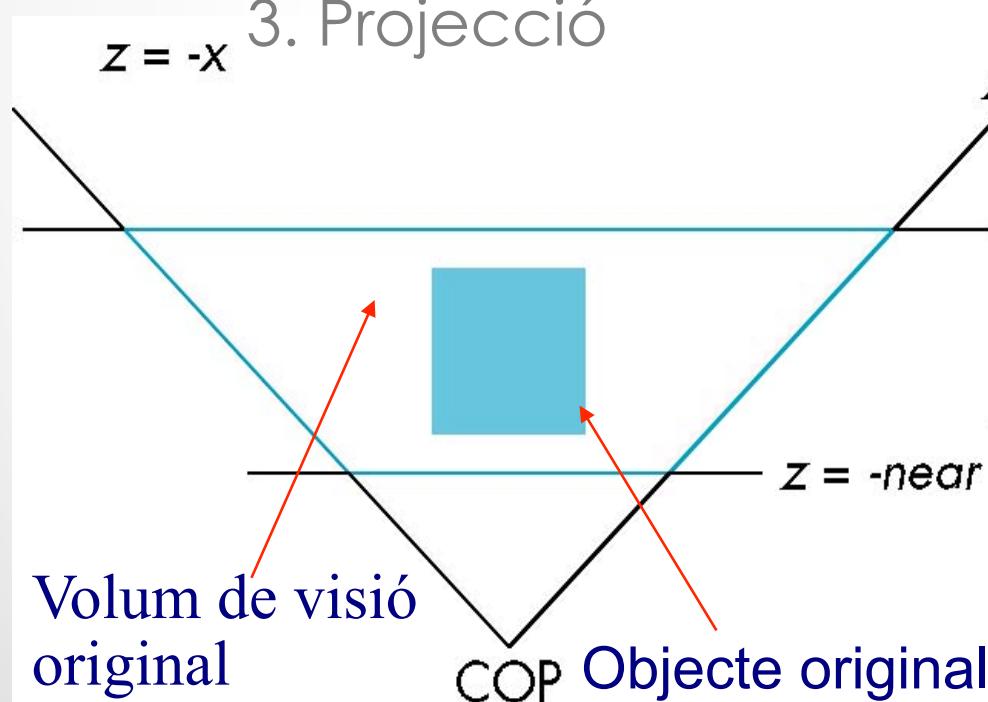
Projecció perspectiva: En el cas general, on els angles d'obertura no són simètrics ni de 90° :

1. Conversió a volum simètric i de 90° : Shear

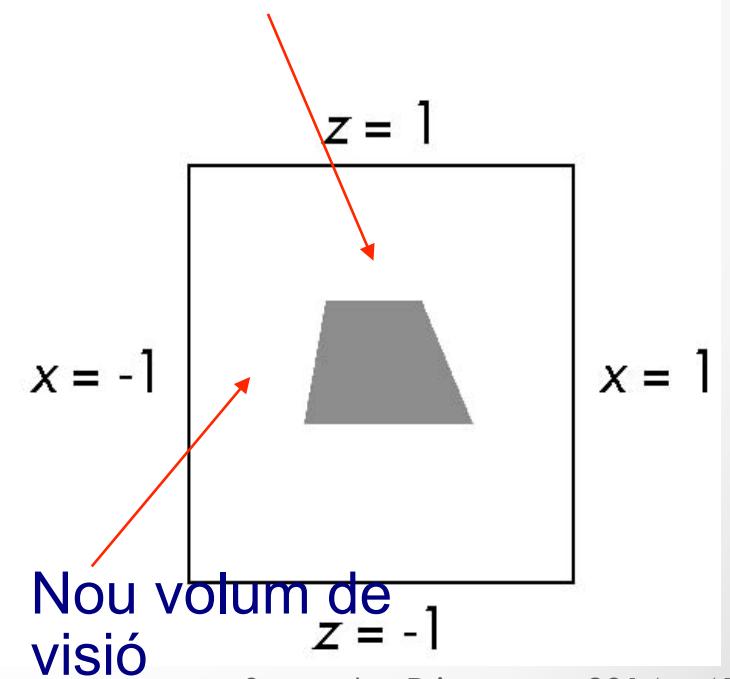
Shear: $((\text{left} + \text{right})/2, (\text{top} + \text{bottom})/2, -\text{near})$ a $(0, 0, -\text{near})$

2. Normalització

3. Projecció



Objecte projectat amb distorsió



4.4. Normalització

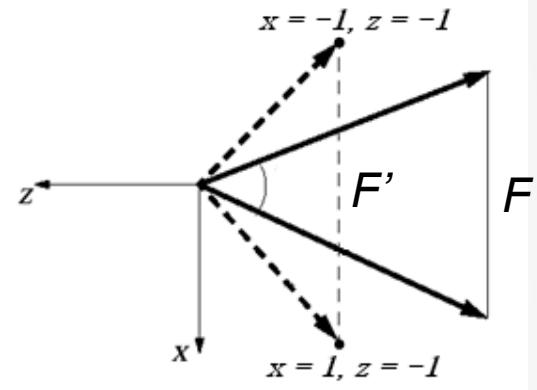
Projecció perspectiva: En el cas general, on els angles d'obertura no són simètrics ni de 90° , es fa primer la conversió a un volum simètric i de 90° (amb un shear, H) + un escalat (normalització, N) + perspectiva(P)

$$\text{Persp} = \text{PNH}$$

Matriu perspectiva (definida en la transparència 24)

shear Normalització

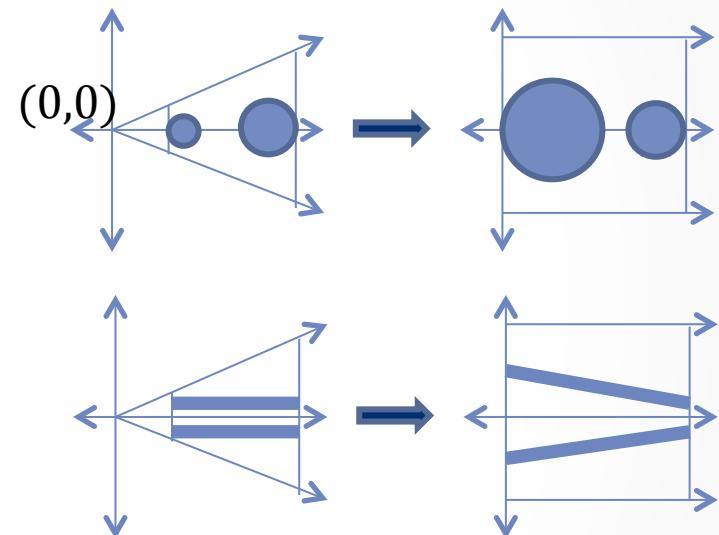
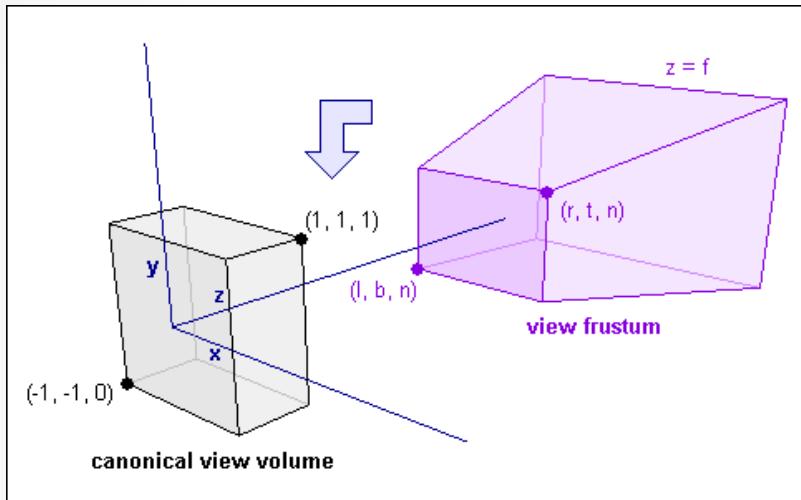
$$\text{Persp} = \text{PNH} = \begin{pmatrix} \frac{2 \cdot \text{near}}{\text{left} - \text{right}} & 0 & \frac{\text{left} + \text{right}}{\text{left} - \text{right}} & 0 \\ 0 & \frac{2 \cdot \text{near}}{\text{top} - \text{bottom}} & \frac{\text{top} + \text{bottom}}{\text{top} - \text{bottom}} & 0 \\ 0 & 0 & -\frac{\text{far} + \text{near}}{\text{far} - \text{near}} & -\frac{2 \cdot \text{far near}}{\text{far} - \text{near}} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$



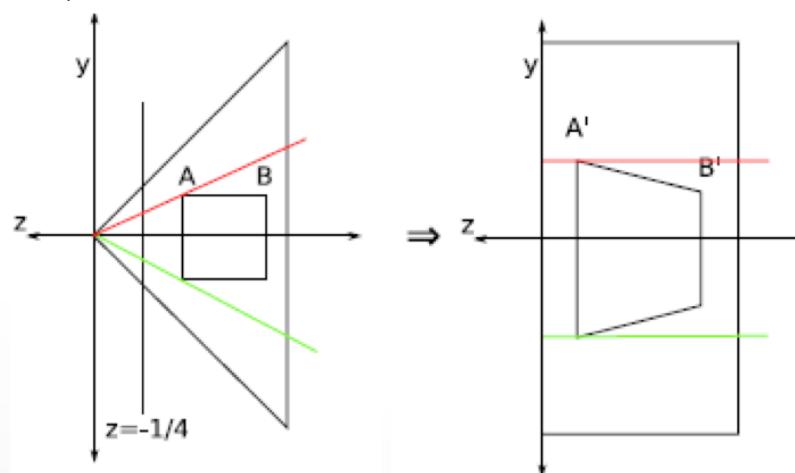
4.4. Normalització

Projeccions perspectives: (mat.h)

Frustum(left, right, bottom, top, near, far)



DirectX: <http://www.codeguru.com/cpp/misc/misc/graphics/article.php/c10123/Deriving-Projection-Matrices.htm#page-3>



Índex

4.1. Introducció

4.2. Viewing: Projeccions

4.3. Volum de visió: definició del *clipping*

4.4. Normalització

4.5. Implementació en OpenGL

4.6. Implementació en GPUs

4.5. OpenGL

En resum, hi ha tres punts en el *pipeline* a determinar:

- **Càmera:**

Definició de la matriu model-view

- **Projectar:**

Definició de la matriu de projecció

- **Retallar (clipping):**

Definició del volum de visió

4.5. OpenGL

Situació inicial a OpenGL:

- Els sistemes de coordenades dels objectes i de la càmera coincideixen.
La matriu model-view és la identitat
- La càmera està a l'origen.
- OpenGL especifica el volum de visió (o visible) com un cub centrat a l'origen i de mida d'aresta 2 (projecció ortogràfica segons eix z).
La matriu de projecció és la identitat

4.5. OpenGL

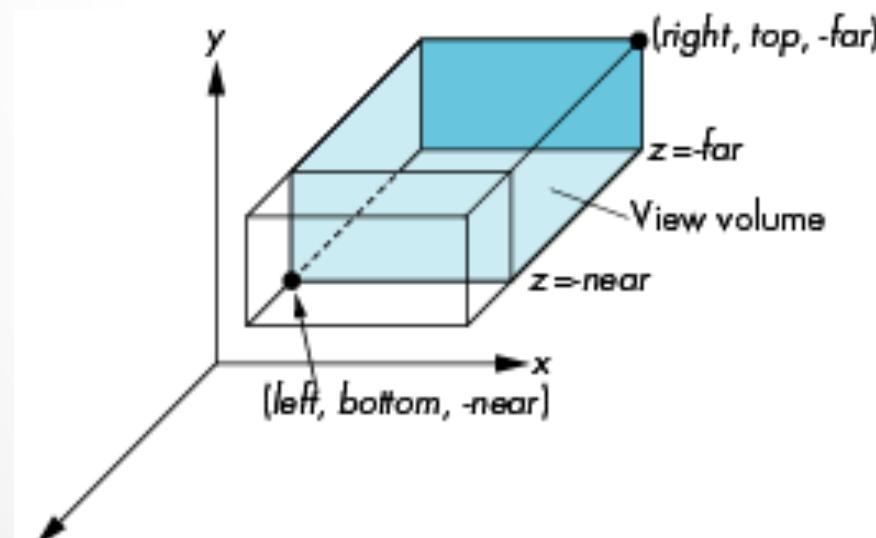
Definició de la matriu projection: Per una projecció **ortogonal** qualsevol, s'usa la crida:

- En versions d'OpenGL on el pipeline és fixe:

void glOrtho(left,right,bottom,top,near,far);

- En versions d'OpenGL on es programa en la GPU, usarem:

mat4 Ortho(left,right,bottom,top,near,far)

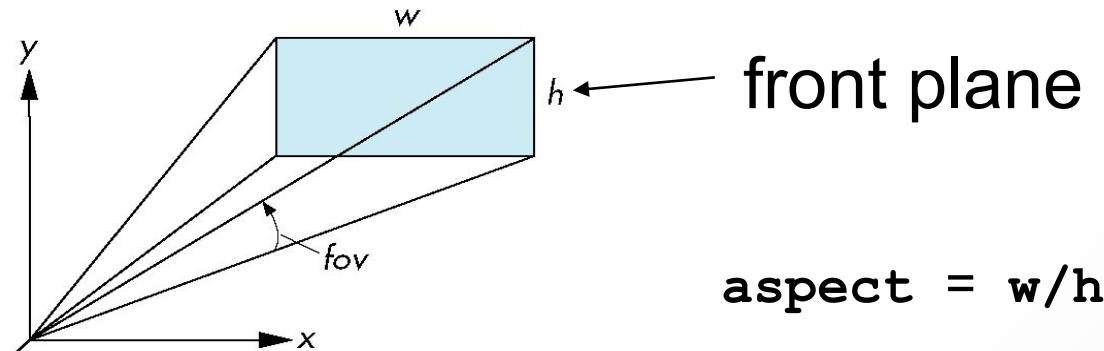


Near i **far** estan mesurats des de la càmera

4.5. OpenGL

Definició de la matriu projection: Per una projecció **perspectiva** qualsevol, s'usa la crida:

- En versions d'OpenGL on el pipeline és fix, s'usa la crida:
void gluPerspective(fovy, aspect, zNear, zFar);
- En versions d'OpenGL on es programa en la GPU, usarem:
mat4 Perspective (fovy, aspect, near, far)

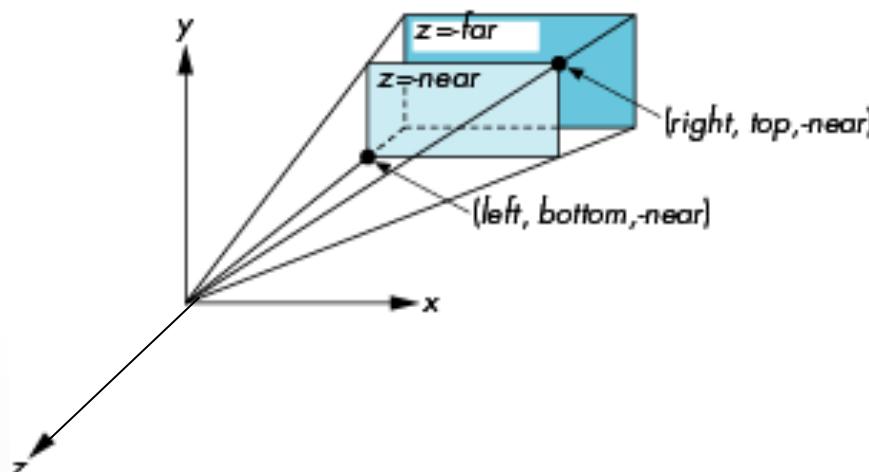


4.5. OpenGL

Definició de la matriu projection: Per una projecció **perspectiva** qualsevol, s'usa la crida:

- En versions d'OpenGL on es programa en la GPU, també es pot usar la crida (permét projeccions perspectives no simètriques, a diferència de la crida Perspective):

mat4 Frustum(left, right, bottom, top, near, far)

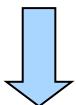


Índex

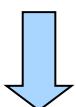
- 4.1. Introducció
- 4.2. Viewing: Projeccions
- 4.3. Clipping
- 4.4. Normalització
- 4.5. Implementació en OpenGL
- 4.6. Implementació en GPUs**

4.6. GPUs

PAS 1. En l'aplicació, es creen les matrius model-view i projection



PAS 2. Es passen aquestes matrius al vertex shader



PAS 3. En el vertex shader es multiplica el punt per les matrius en l'ordre adient i es normalitzen els vèrtexs

4.6. GPUs

- **Pas 1:** Des de l'aplicació, es creen les matrius model-view (mv) i projection (p)
 - Per a crear matrius i manipular-les teniu una classe **Common** i les classes **mat4** i **vec4**.
- **Pas 2:** Es passen al vertex shader les matrius des de la CPU.
 1. S'obté el lloc de memòria que correspon a la GPU, fent la comanda:

```
GLuint model_view, projection;  
model_view = program->uniformLocation("model_view");  
projection = program->uniformLocation("projection");
```

2. Es lliguen les variables:

```
glUniformMatrix4fv( model_view, 1, GL_TRUE, mv );  
glUniformMatrix4fv( projection, 1, GL_TRUE, p );
```

TGs en el vertex shader

- **Pas 3:** En el vertex shader es multipliquen els punts per les matrius en l'ordre adient i es normalitzen els vèrtexs

El codi del vertex shader, seria:

```
in vec4 vPosition;  
in vec4 vColor;  
out vec4 color;  
uniform mat4 model_view;  
uniform mat4 projection;
```

```
void main()  
{  
    gl_Position = aplicar les matrius a vPosition
```

gl_Position = passar a coordenades homogènies
color = vColor;

```
}
```

Conclusions

- En aquestes transparències hem après a situar la càmera, els tipus de **projeccions** que es poden realitzar, els **volums de visió** que defineixen els **plans de retallat (clipping)**, el procés de **normalització**
- Hem après a definir la matriu **model-view** i la matriu **projection**.
- S'ha vist com s'implementen en el pipeline fixe d'OpenGL
- S'ha vist com es fan els càlculs de les matrius des de l'aplicació i com es passen al **vertex shader** de la GPU.