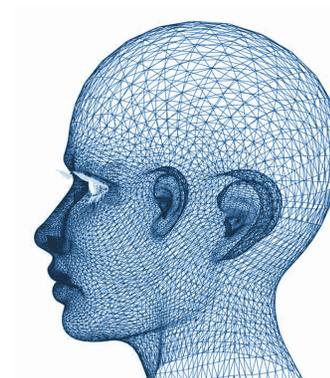


Appunti di Computer Graphics



Ing. Luca Gardelli

lgardelli@deis.unibo.it

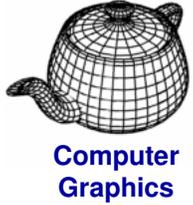
lgardelli@unibo.it

DEIS - Dipartimento di Elettronica Informatica e Sistemistica

Università degli Studi di Bologna sede di Cesena

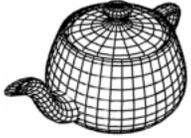
via Venezia 52, 47023, Cesena (FC) Italy

Ph: +39 0547 339244 Fax: +39 0547 339208

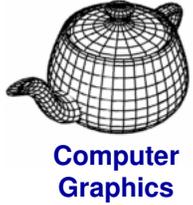


Sommario

- Introduzione
- Trasformazioni
- Modello di Cinepresa
- Rendering Pipeline

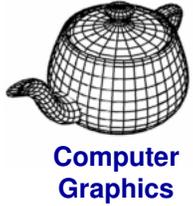


Introduzione



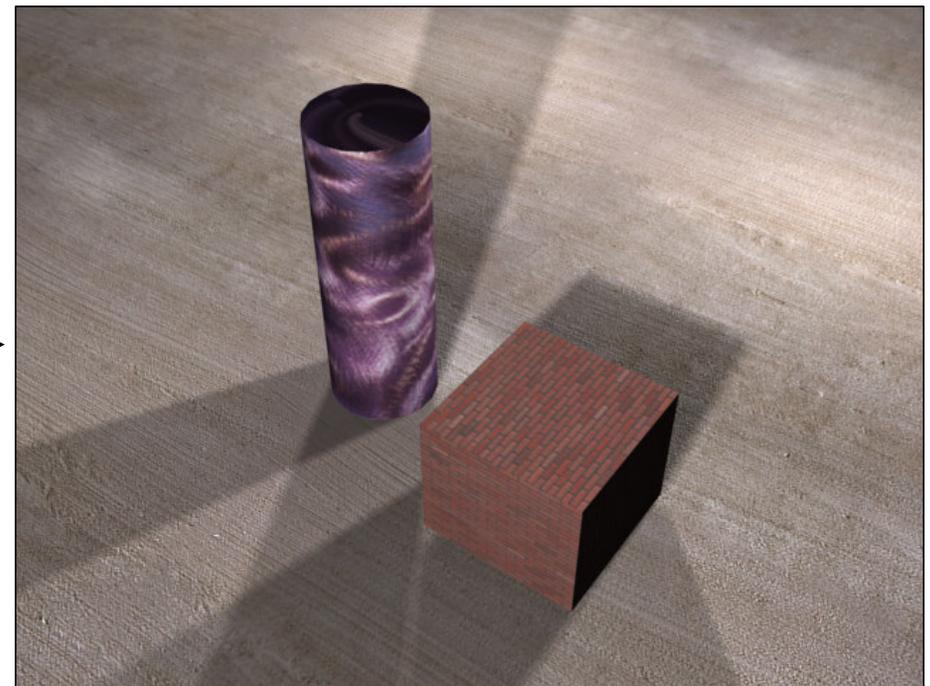
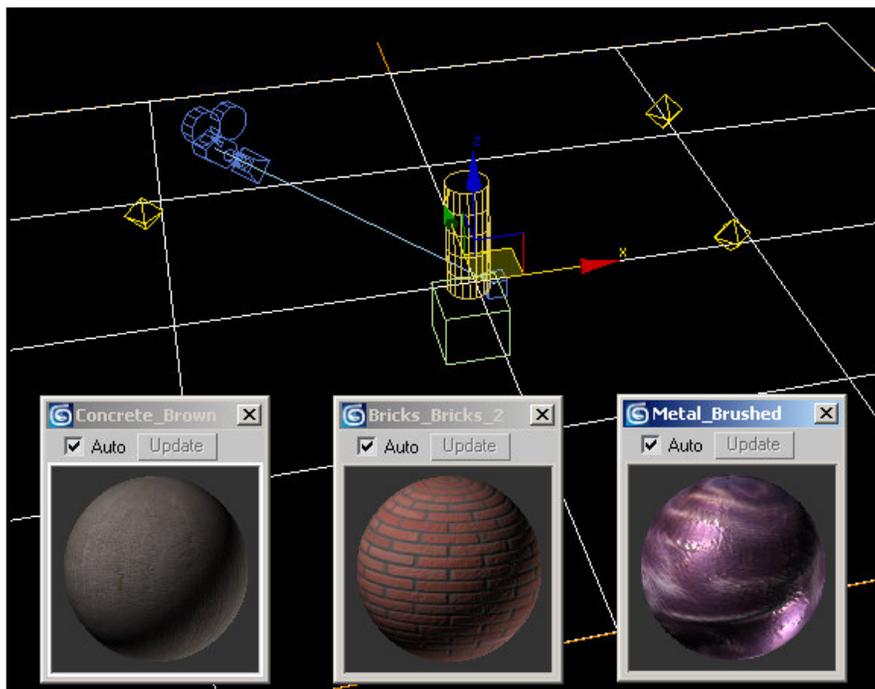
Grafica 3D

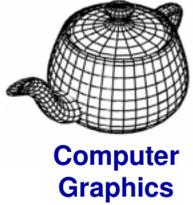
- Quali sono gli elementi necessari alla grafica 3D?
 1. Un **osservatore**
 - Posizione
 - Angolo di visuale..
 2. Una **scena**
 - Primitive geometriche
 - Luci e effetti atmosferici
 - Informazioni per l'animazione
 - Materiali e Texture..



Rendering

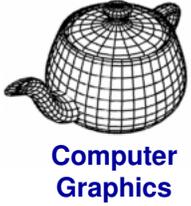
- Si dice **rendering** il processo di trasformazione – in base alle caratteristiche dell'osservatore – di una scena tridimensionale in un'immagine bidimensionale





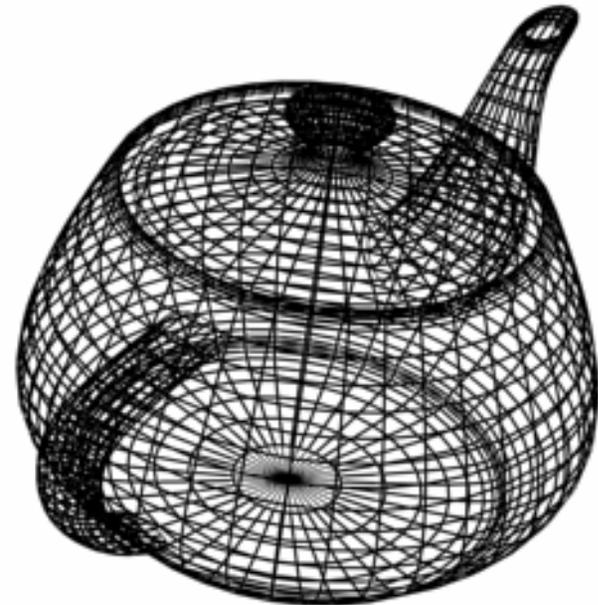
Stadi del rendering

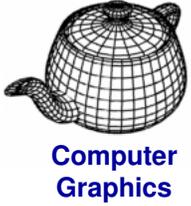
- Concettualmente il rendering è composto da 3 fasi
 1. **Application stage** – i modelli 3d vengono trasformati in primitive
 2. **Geometry stage** – si posizionano gli oggetti e la cinepresa (camera), si eseguono le proiezioni di vista
 3. **Rasterizer stage** – si determina il colore dei pixel in base alle texture e posizione relativa degli oggetti



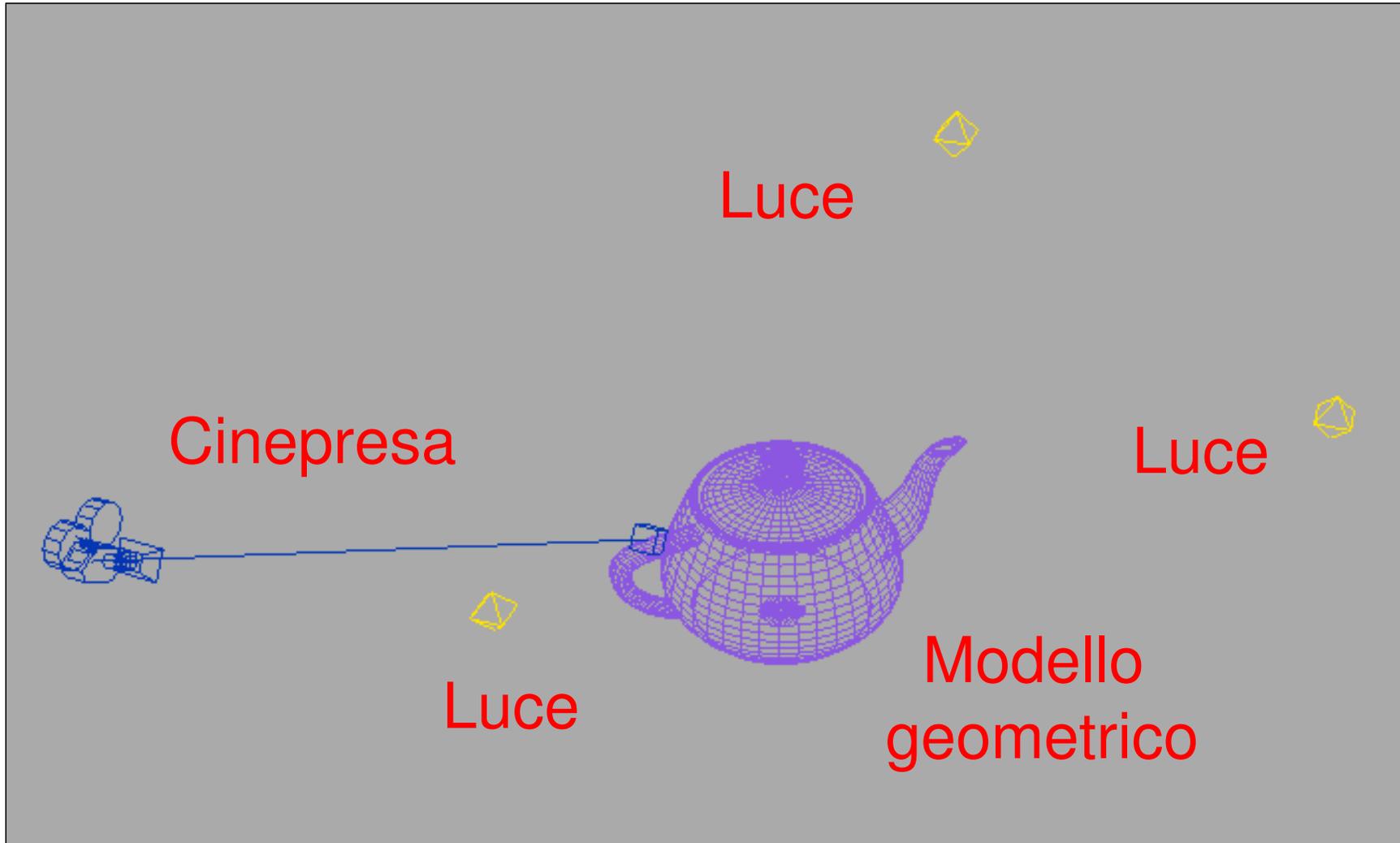
Application stage

“teapot(10 , 25.0)”



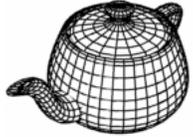


Geometry stage

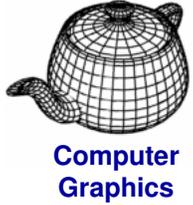


Rasterizing stage





Trasformazioni

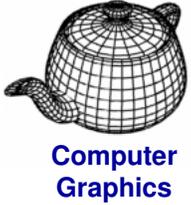


Quaternioni

- L'algebra dei quaternioni è stata descritta da **Sir William Rowan Hamilton** alla fine del XIX secolo e applicati principalmente alla meccanica
- Sostanzialmente si riprende l'algebra di base e la si estende alla quarta dimensione
- In questo seminario non andremo nei dettagli dell'algebra dei quaternioni
- In pratica ci interessano perchè ci permettono di rappresentare la traslazione come una matrice 4x4: questo non è possibile con una matrice 3x3!

$$q = (x, y, z, w)$$

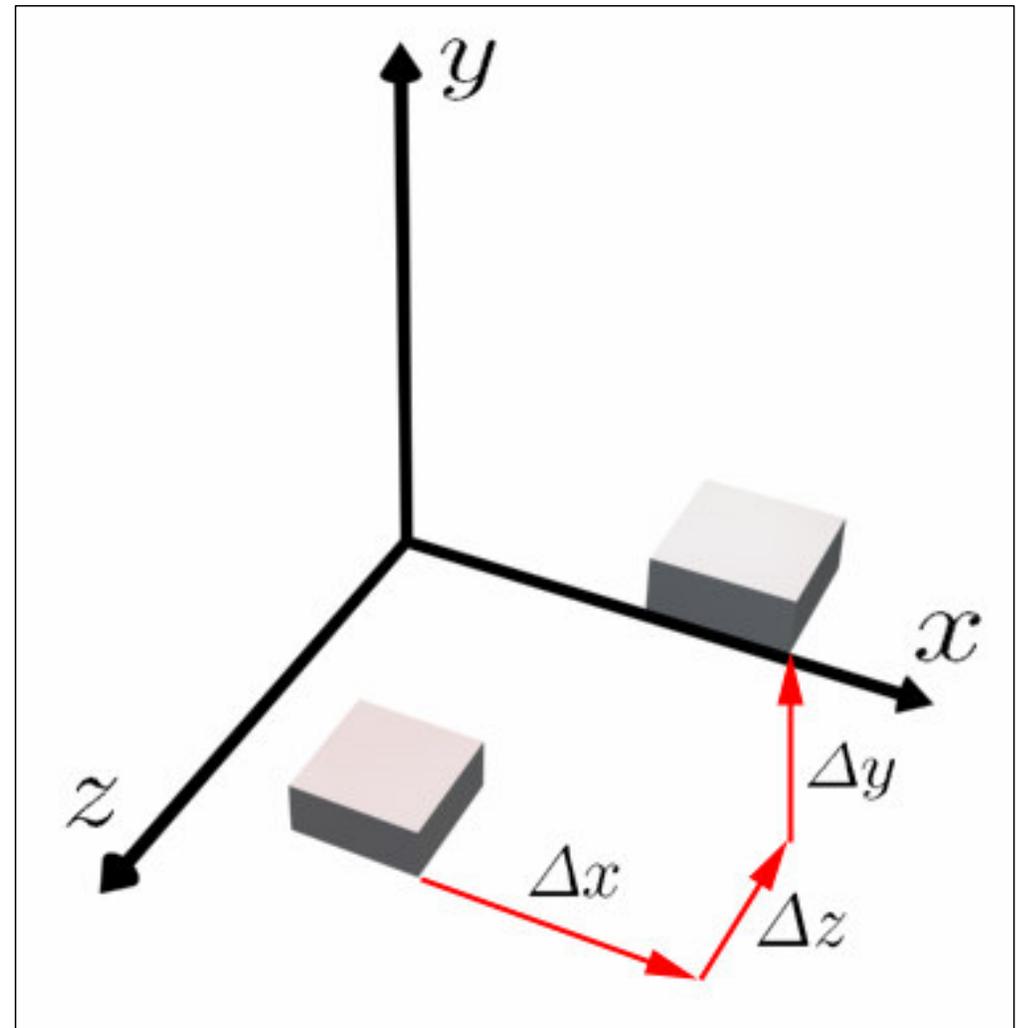
dove la n-pla (x,y,z) è un punto nello spazio 3D ed indica un punto nello **spazio di coordinate omogeneo** se lo scalare $w=1$ o un vettore se lo scalare $w=0$

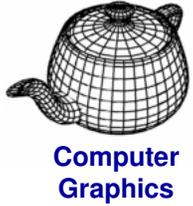


Traslazione

- La traslazione non modifica un oggetto, ma ne cambia la posizione nello spazio
- Può essere rappresentata da una matrice nello spazio omogeneo (in basso)

$$T = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

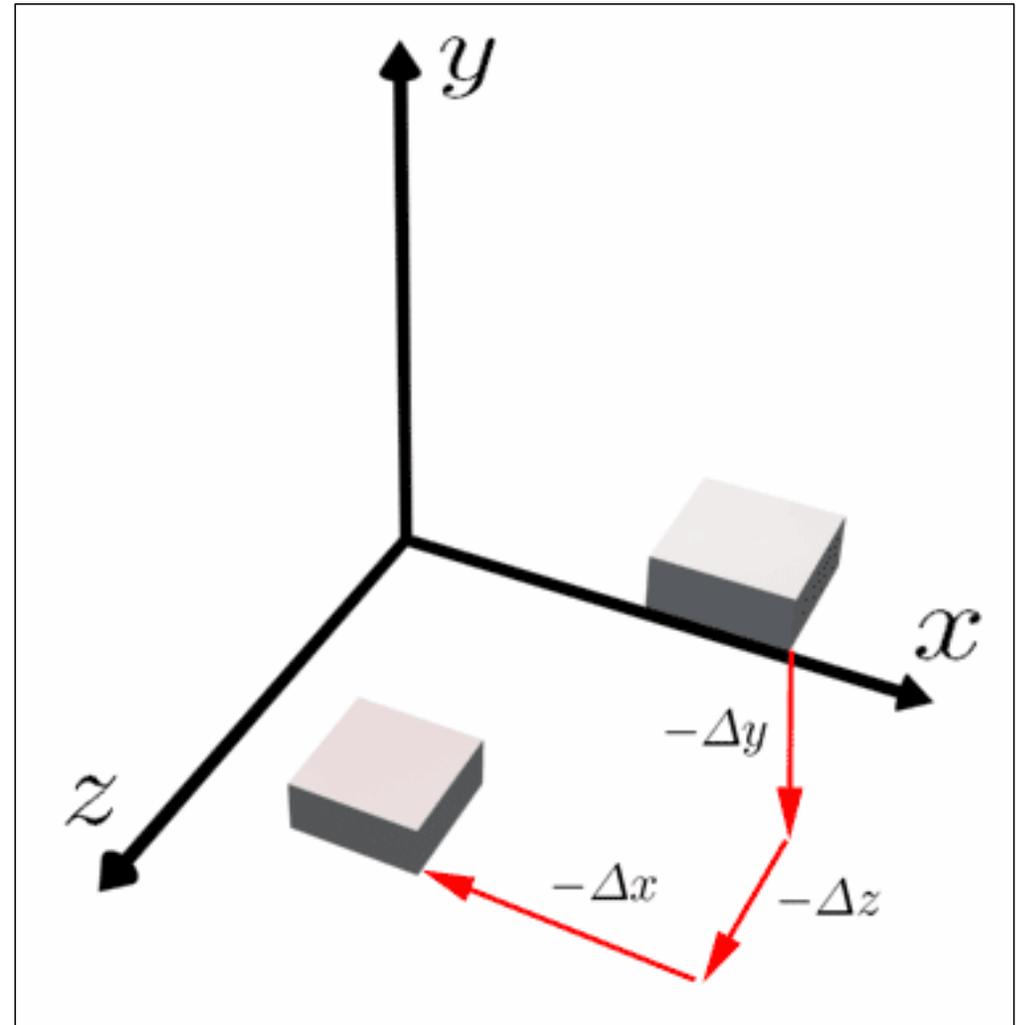


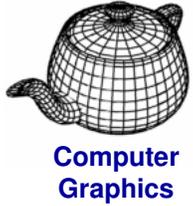


Traslazione inversa

- La trasformazione inversa della traslazione equivale ad una traslazione in negativo
- Può essere rappresentata da una matrice nello spazio omogeneo (in basso)

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -\Delta x \\ 0 & 1 & 0 & -\Delta y \\ 0 & 0 & 1 & -\Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

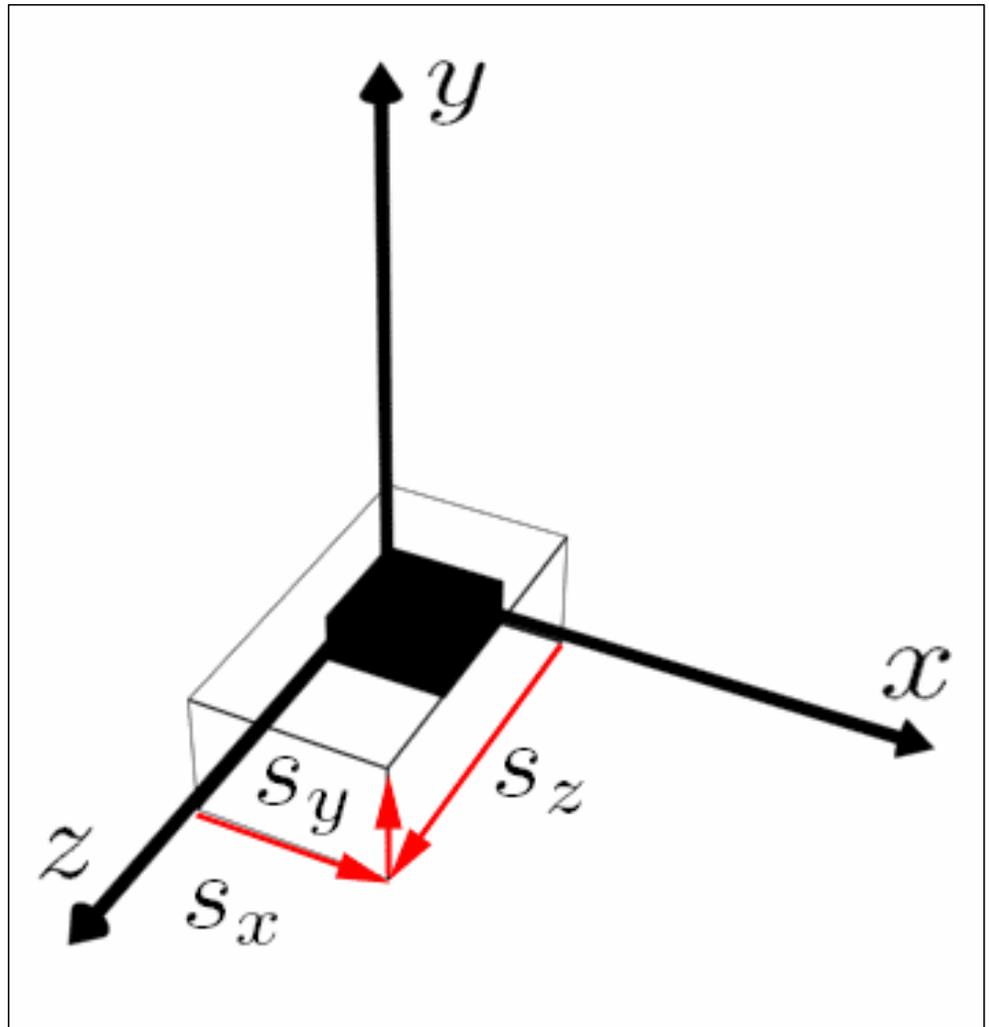




Scalatura

- La scalatura modifica un oggetto
- Può essere rappresentata da una matrice nello spazio omogeneo (in basso)

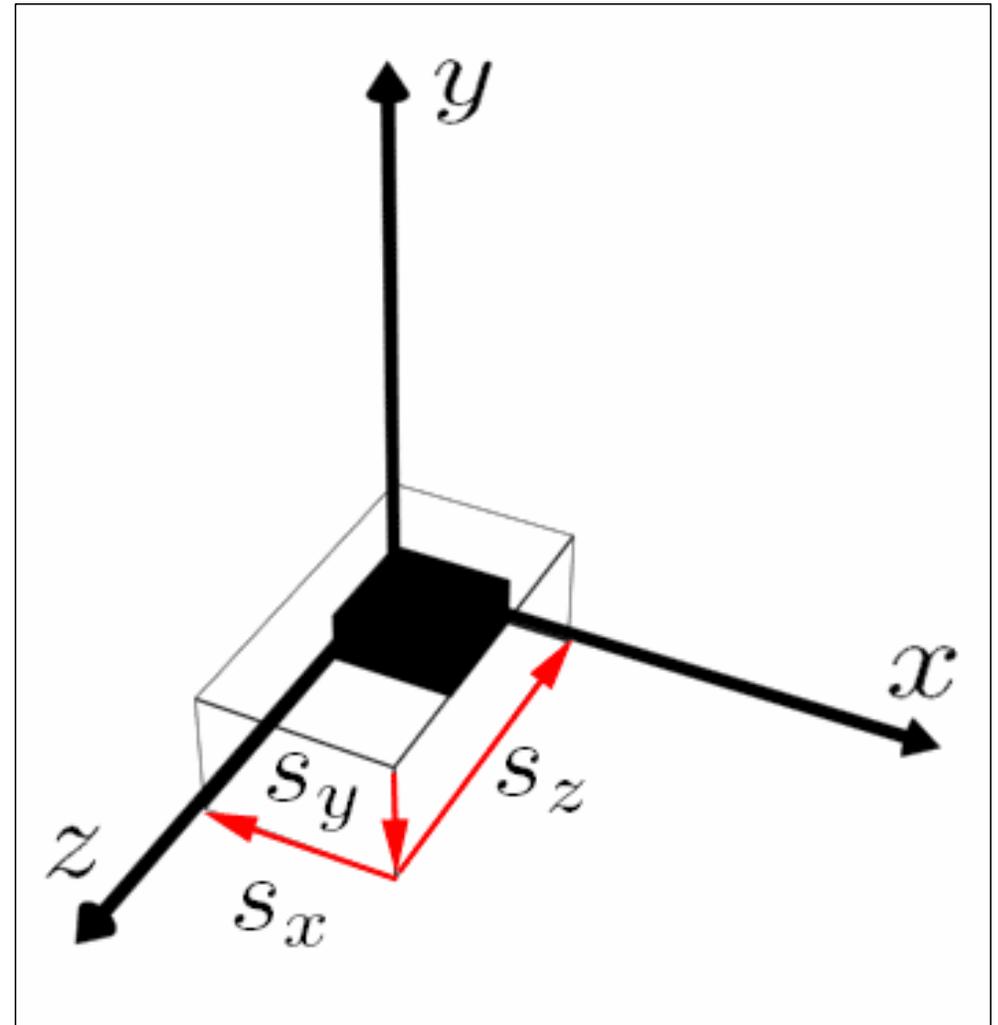
$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

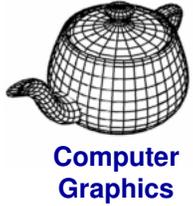


Scalatura inversa

- La scalatura inversa equivale ai reciproci dei fattori di scala
- Può essere rappresentata da una matrice nello spazio omogeneo (in basso)

$$S^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 & 0 \\ 0 & 0 & \frac{1}{s_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

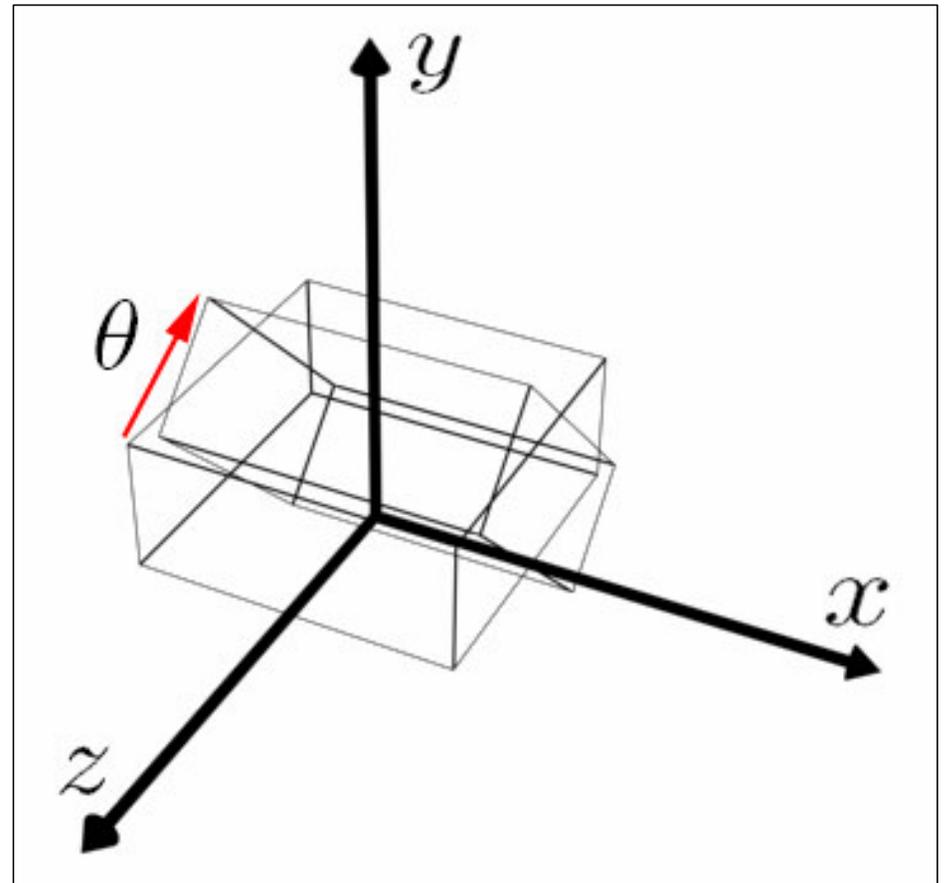




Rotazione lungo asse x

- La rotazione non modifica un oggetto, ma ne cambia l'orientamento nello spazio.
- Può essere rappresentata da una matrice nello spazio omogeneo (in basso)

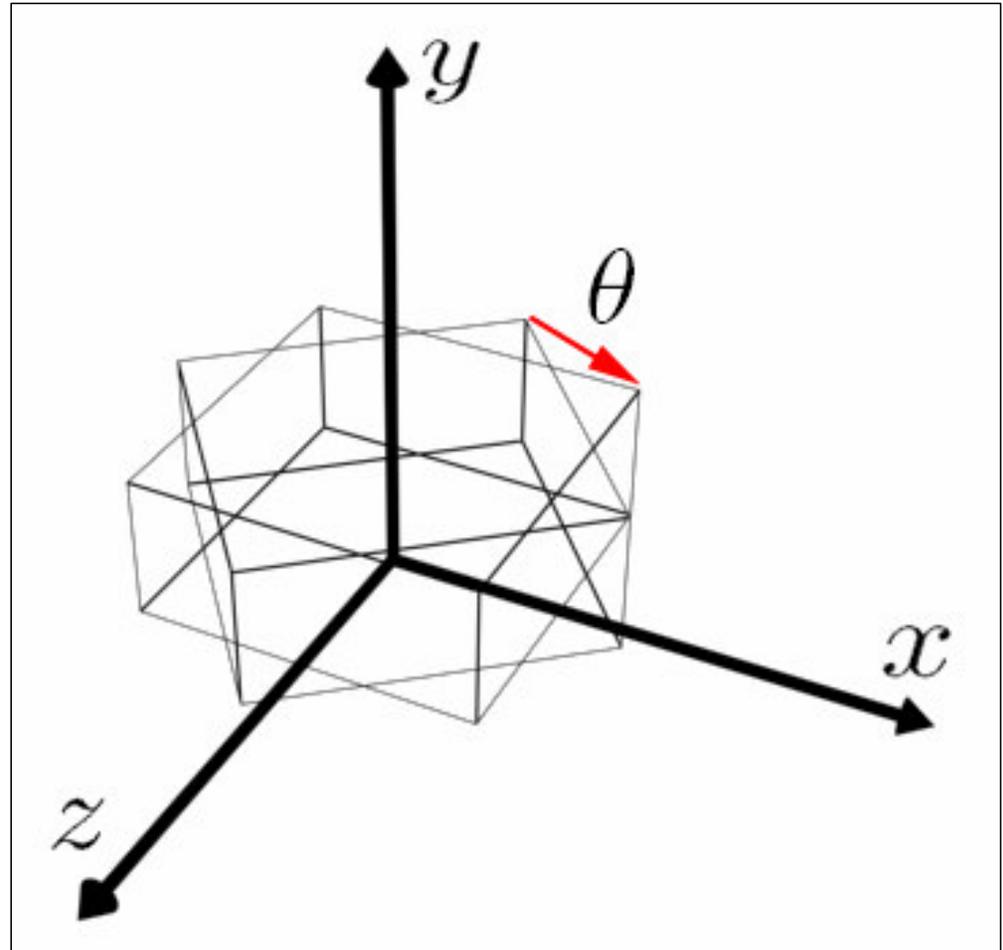
$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

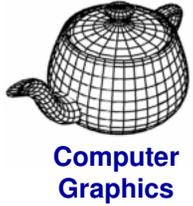


Rotazione lungo asse y

- La rotazione non modifica un oggetto, ma ne cambia l'orientamento nello spazio.
- Può essere rappresentata da una matrice nello spazio omogeneo (in basso)

$$R_y = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

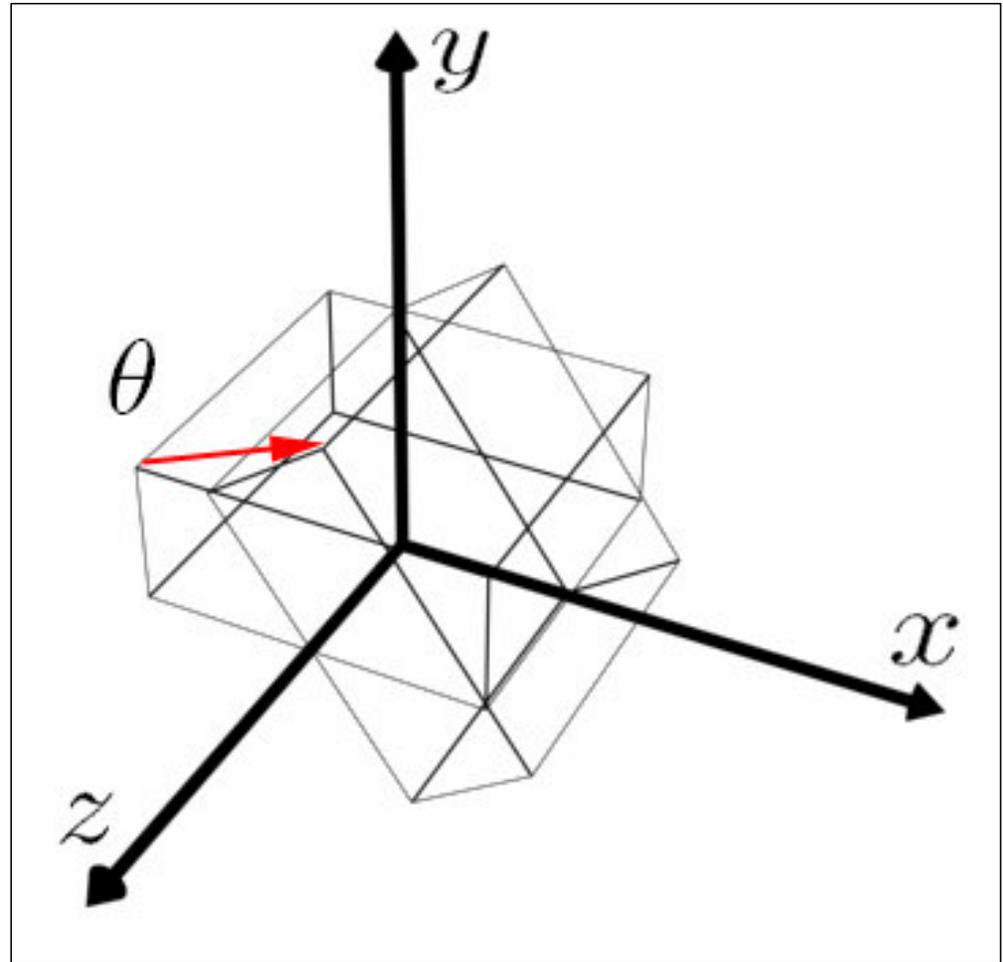




Rotazione lungo asse z

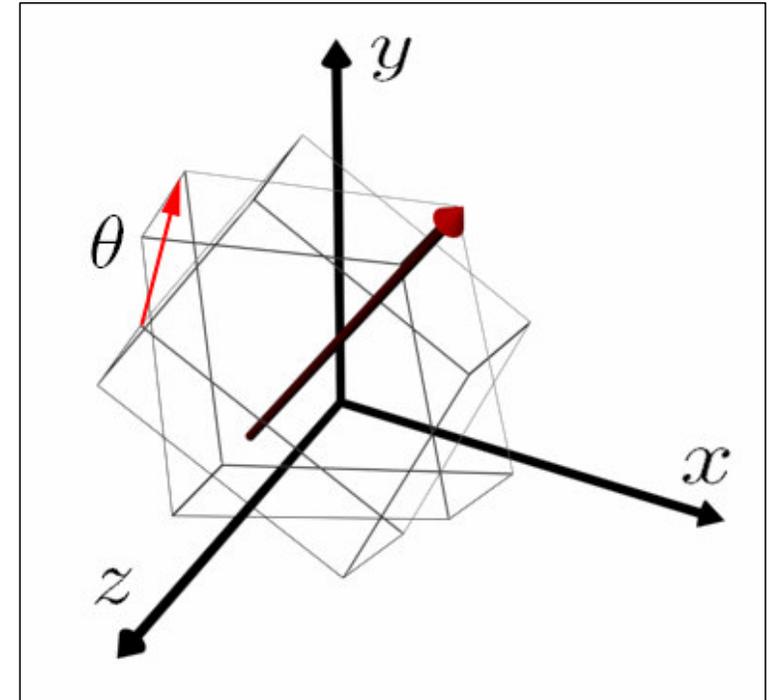
- La rotazione non modifica un oggetto, ma ne cambia l'orientamento nello spazio.
- Può essere rappresentata da una matrice nello spazio omogeneo (in basso)

$$R_z = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

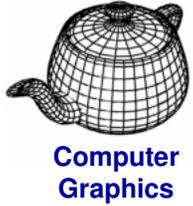


Rotazione lungo un asse

- La rotazione non modifica un oggetto, ma ne cambia l'orientamento nello spazio.
- Può essere rappresentata da una matrice nello spazio omogeneo (in basso), dove $v = (x,y,z)$ è l'asse di rotazione unitario

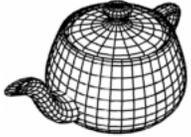


$$R_{axis} = \begin{bmatrix} \cos \theta + (1 - \cos \theta)x^2 & (1 - \cos \theta)xy - \sin \theta z & (1 - \cos \theta)xz + \sin \theta y & 0 \\ (1 - \cos \theta)yx + \sin \theta z & \cos \theta + (1 - \cos \theta)y^2 & (1 - \cos \theta)yz - \sin \theta x & 0 \\ (1 - \cos \theta)zx & (1 - \cos \theta)zy + \sin \theta x & \cos \theta + (1 - \cos \theta)z^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

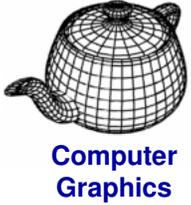


..ed ora...?

- Abbiamo già indicato le fasi del processo di rendering
- L'osservatore svolge un ruolo fondamentale in questo processo
- Ora siamo anche in grado di eseguire delle trasformazioni su punti e vettori
- Proseguiamo proponendo un modello di osservatore
- Utilizzeremo la metafora della cinepresa virtuale



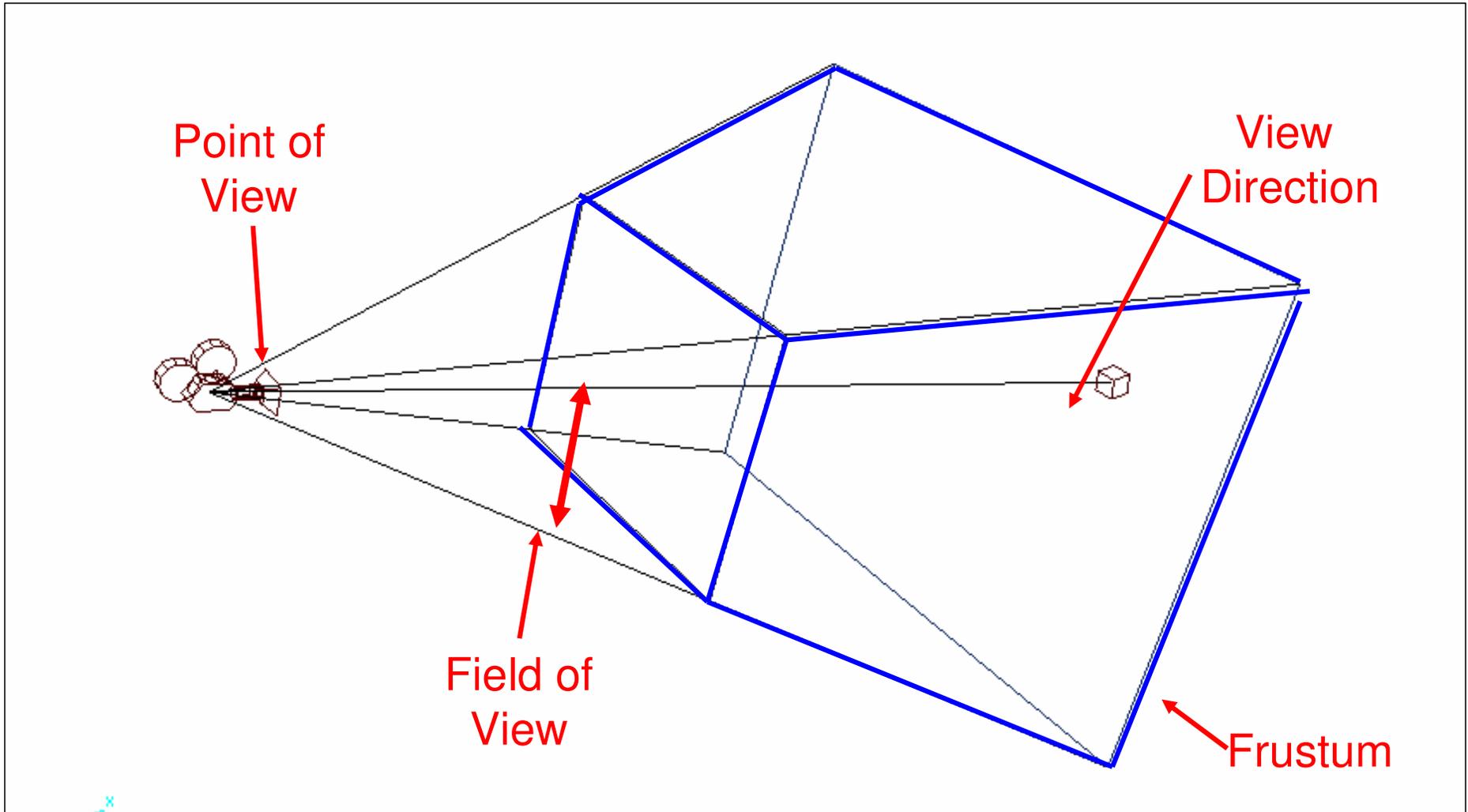
Modello di Cinepresa



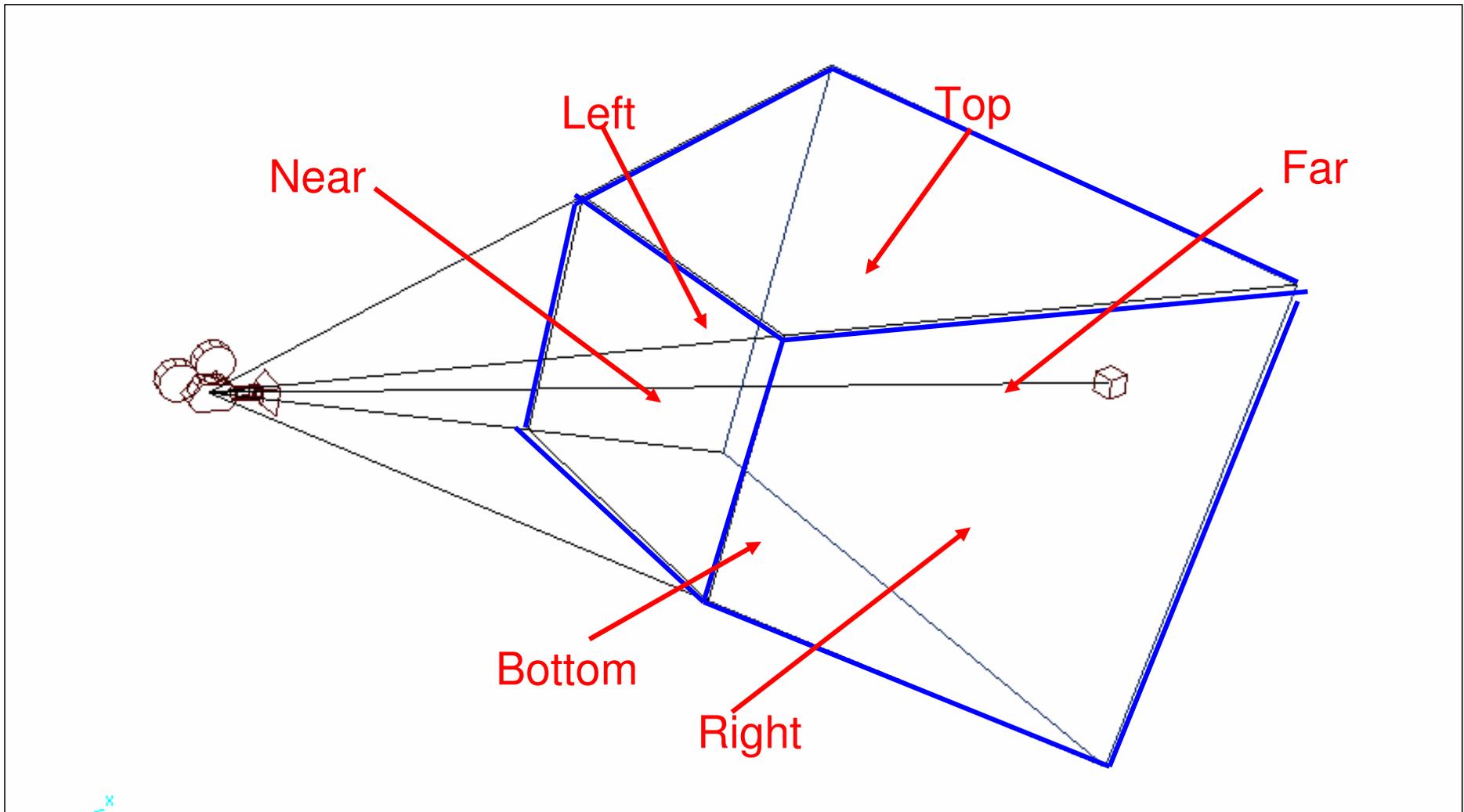
Camera Model

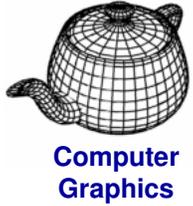
- E' definito da
 - **Point of view**: la posizione dell'osservatore
 - **View direction**: la direzione in cui l'osservatore sta guardando
 - **Field of view (fov)** : l'apertura del cono di vista
 - **clipping planes**: limitano ulteriormente la porzione di scena visibile
- **Tipo di proiezione**: in generale si assume che la proiezione sia di tipo prospettico a 3 punti di fuga, ma potrebbe anche essere una proiezione parallela

Camera Model



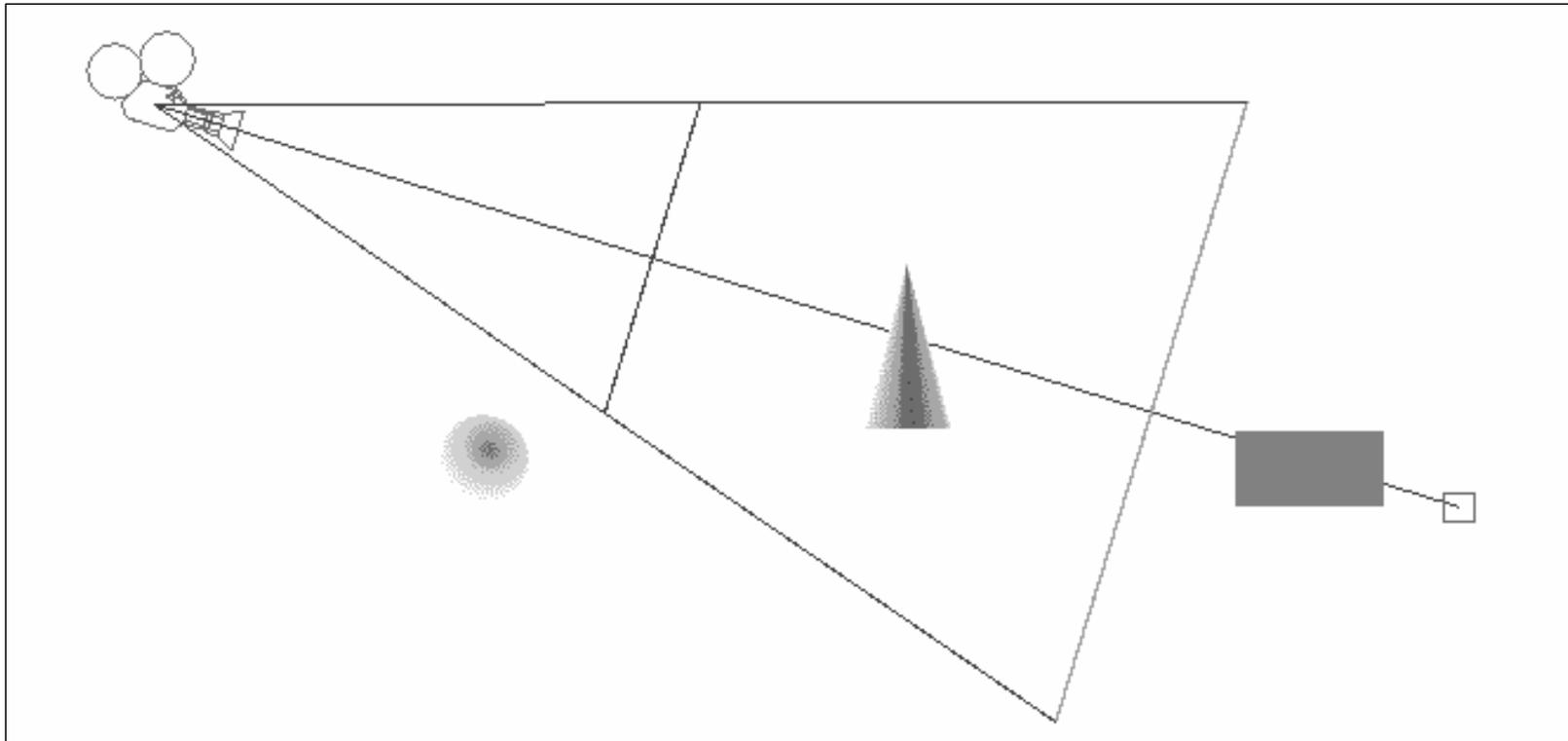
Camera Model – Clipping Planes

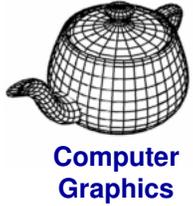




Camera Model

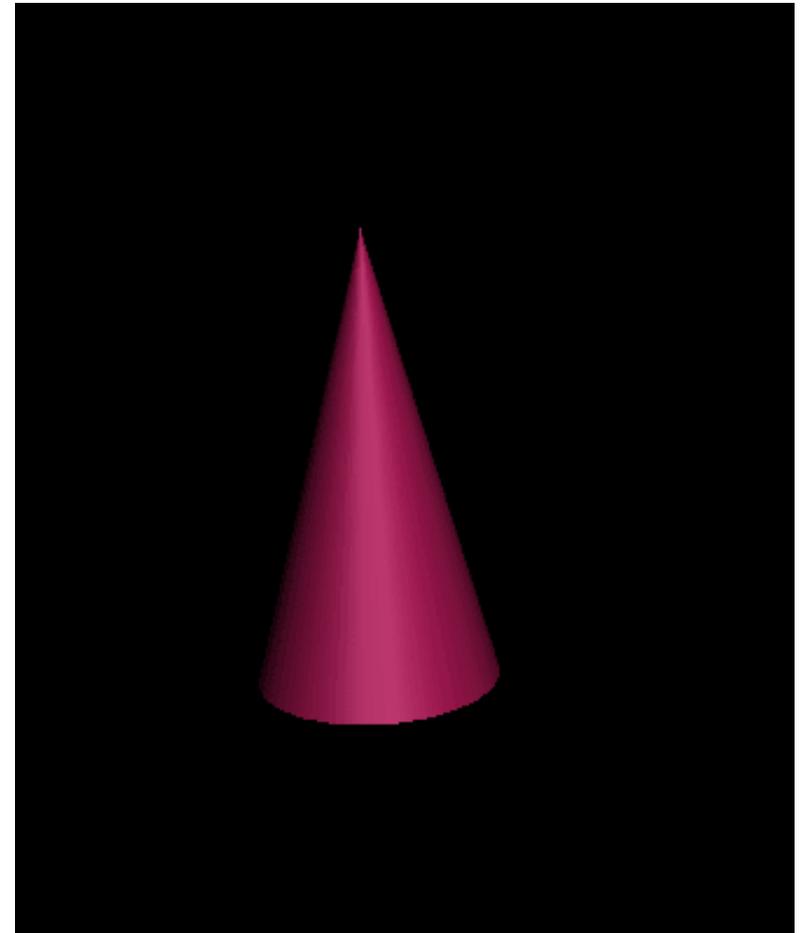
- Se si renderizza la scena attraverso la cinepresa cosa si osserva?

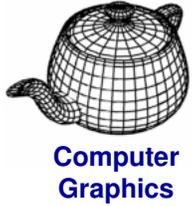




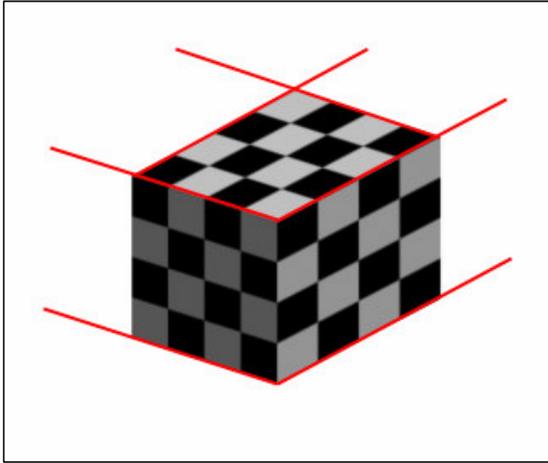
Camera Model

- Viene renderizzato solo ciò che è all'interno del frustum, cioè il cono
 - il parallelepipedo è oltre il far clipping plane
 - la sfera è fuori dal campo della visuale

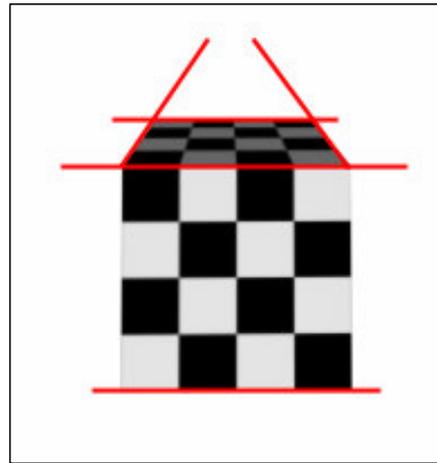




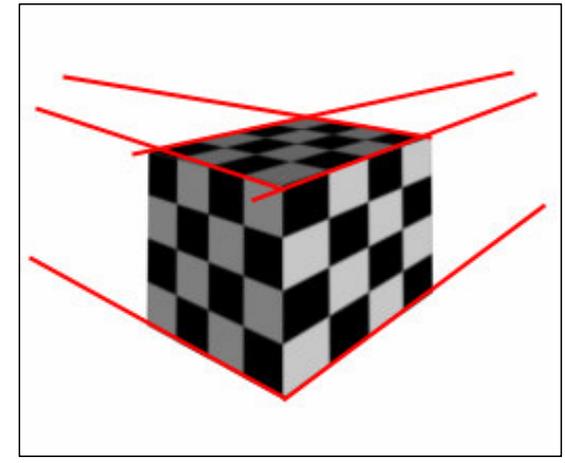
Tipi di proiezione



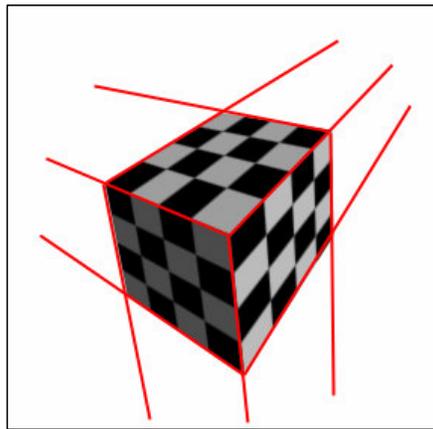
Parallela



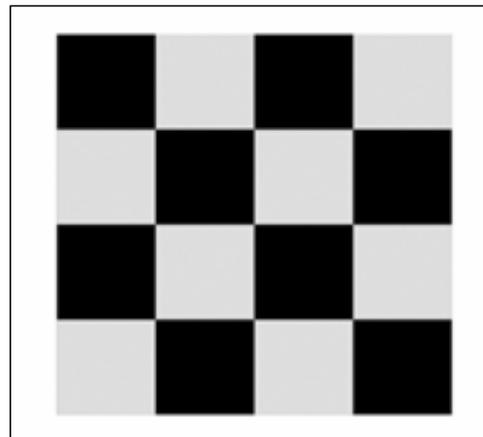
Prospettiva 1 punto



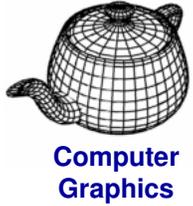
Prospettiva 2 punti



Prospettiva 3 punti

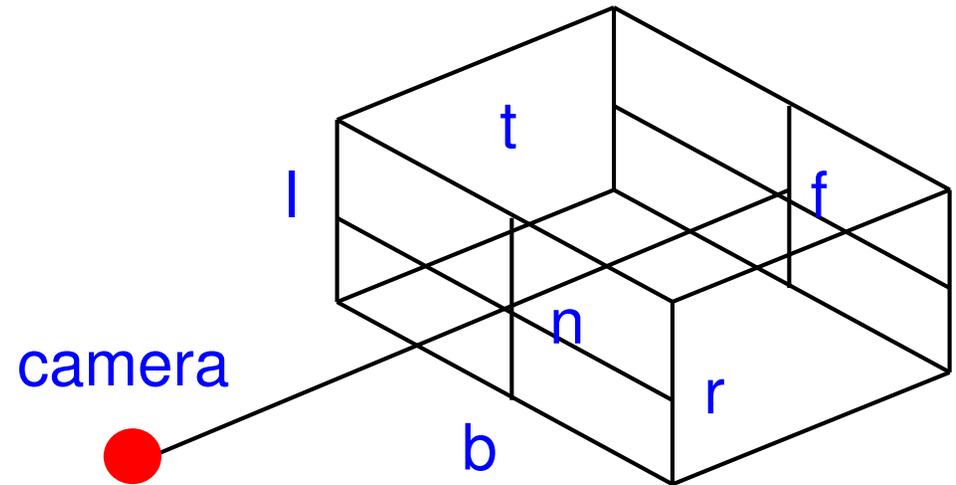


Ortogonale



Proiezione parallela

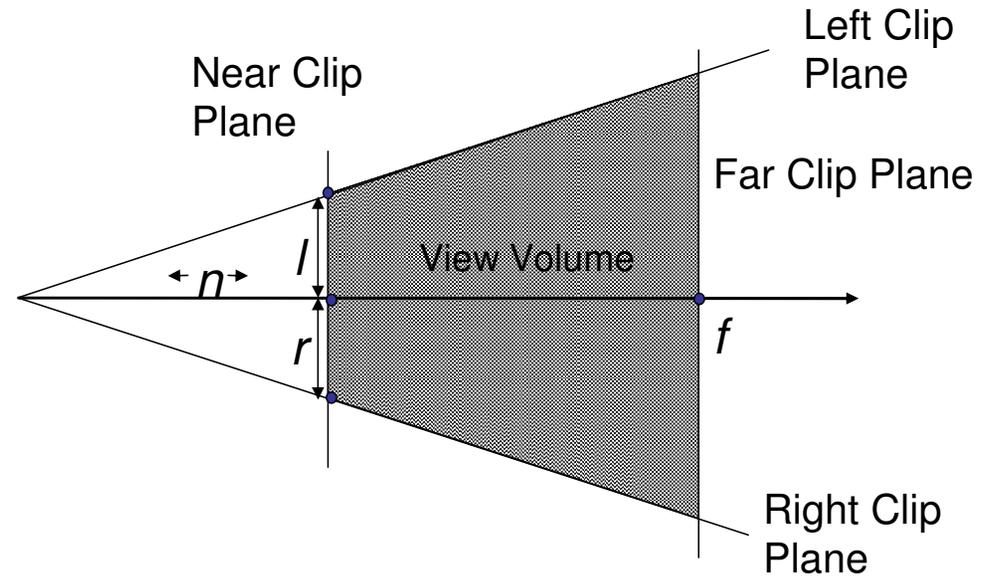
- Occorre ricondursi alla configurazione canonica, cubo nello spazio $-1, +1$



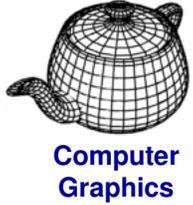
$$P_{orto} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Deformazione prospettica

- Occorre ricondursi alla configurazione canonica, cubo nello spazio $-1, +1$
- Occorre deformare gli oggetti in modo da simulare la prospettiva

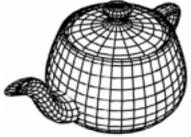


$$P_{persp} = \begin{bmatrix} \frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & -\frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{(n+f)}{n-f} & \frac{-2nf}{n-f} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

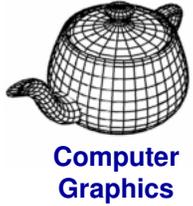


..ed ora...?

- Abbiamo già indicato le fasi del processo di rendering
- Abbiamo proposto un modello di cinepresa
- Siamo in grado di eseguire trasformazioni su punti e vettori
- Proseguiamo vedendo nel dettaglio quali sono le operazioni coinvolte nella **pipeline di rendering**



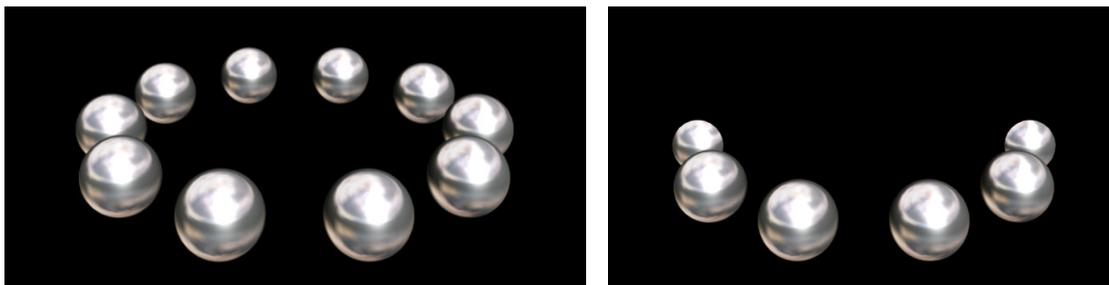
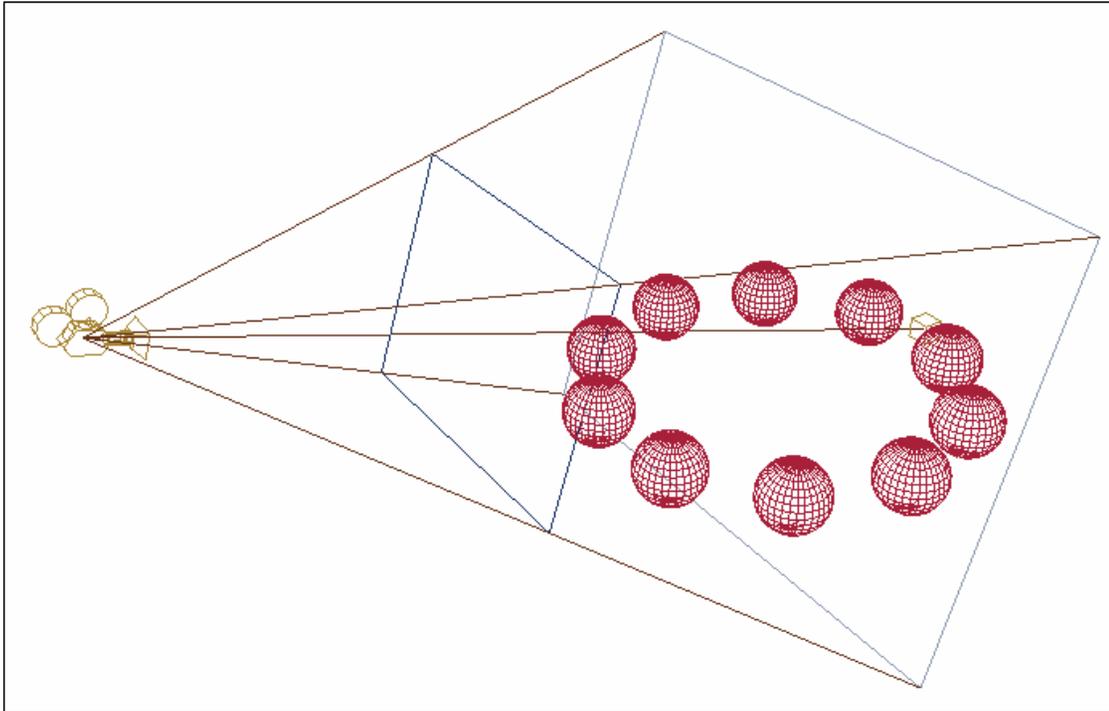
Rendering Pipeline



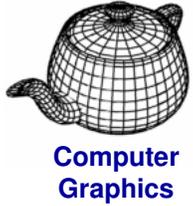
Rendering Pipeline

- Come avviene la rendering?
 1. si elimina tutto ciò che è fuori dal frustum
- *clipping*
 2. si convertono le coordinate da spazio globale a spazio locale della cinepresa
 3. si esegue la proiezione prospettica (in uno cubo unitario)
 4. si eliminano le parti di un oggetto che non sono visibili dall'osservatore - *back-face culling*
 5. si eliminano le parti di un oggetto che sono occluse da un altro oggetto – *hidden surface removal*
 6. si applica il modello di illuminazione scelto

Clipping



- Elimina le parti di scena che non giacciono all'interno del frustum
- A sinistra la scena
- In basso a sinistra il rendering con *far clipping disabilitato*
- In basso a destra il rendering con *far clipping abilitato*

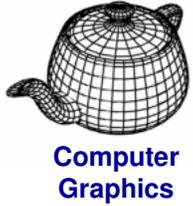


Conversione di coordinate

- E' necessario riportare le coordinate globali delle primitive da renderizzare allo spazio di coordinate locale alla cinepresa

$$T_{view} = T^{-1} S^{-1} R^{-1}$$

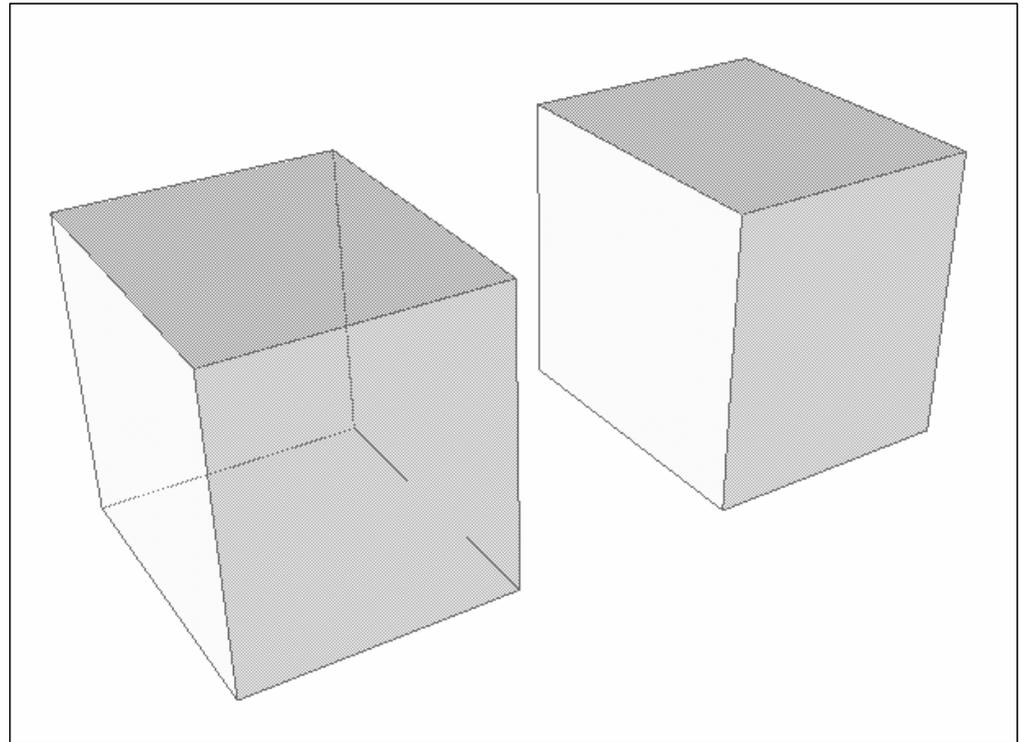
- Dove T, S, R sono rispettivamente le matrici di traslazione, rotazione e scalatura

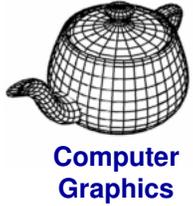


Back-Face Culling

- A sinistra il back-face culling è *disattivato*
- A destra il back-face culling è *attivato*
- Questo meccanismo si basa sul confronto fra la normale dei poligoni con la direzione di vista dell'osservatore

$$visibility = n_p \cdot n_v > 0$$

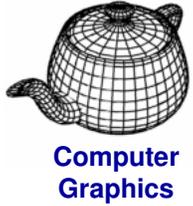




Hidden Surface Removal

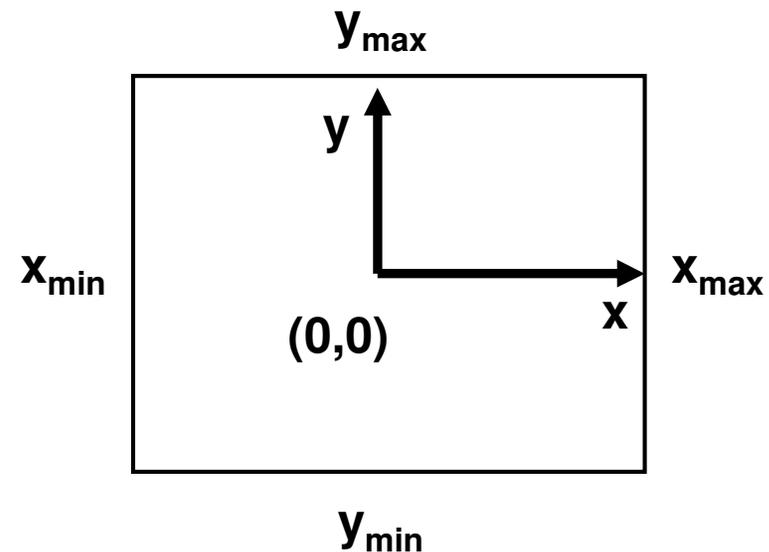
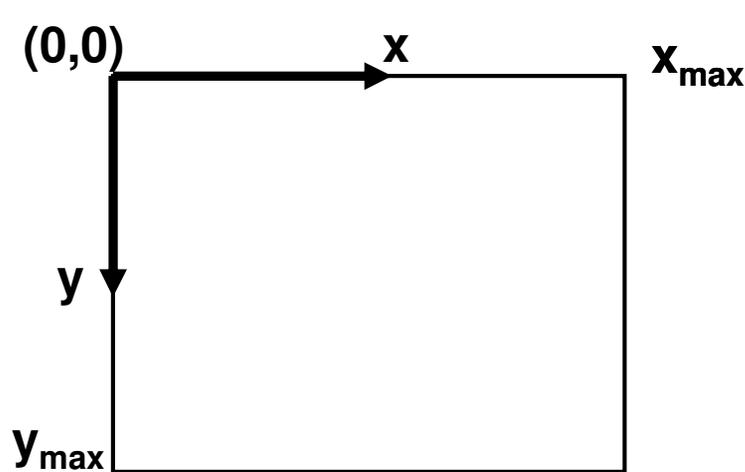
- Hidden surface removal elimina le parti degli oggetti occlusi
- E più complesso del culling, non si può eliminare completamente una faccia
- Solitamente si basa sul depth-buffer





Conversione Screen \Leftrightarrow Viewport

Assumiamo che la viewport abbia l'origine al centro e le stesse dimensioni della window!



$$x_s = x_v + \text{width}/2$$
$$y_s = \text{height}/2 - y_v$$

$$x_v = x_s - \text{width}/2$$
$$y_v = \text{height}/2 - y_s$$