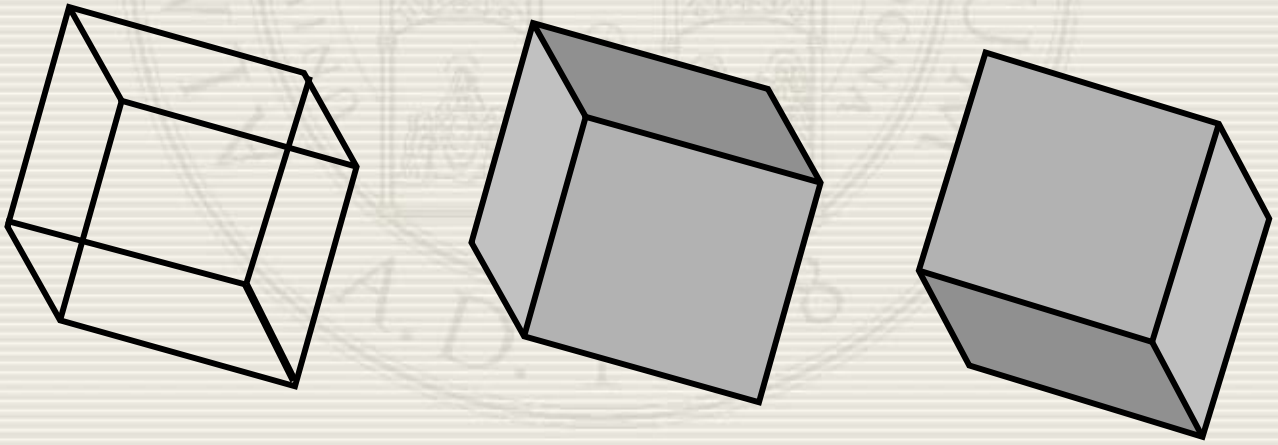
Algoritmi di Real-Time Rendering

Wire Frame

Il problema delle linee e superfici nascoste, insieme ad una visualizzazione real-time, è uno dei problemi più interessanti della Computer Graphics.

Gli algoritmi relativi, cercano di determinare quali lati, superfici (facce triangolari) o volumi (tetraedri) siano visibili da un determinato Vp in un tempo utile per una visualizzazione real-time.



Ambiguità del wire frame

Classificazione e un po' di storia

Eliminazione Parti Nascoste

- Hidden Line Removal (HL)
- Hidden Surface Removal (HS)

Eliminazione Parti Nascoste

Tutti gli algoritmi che si basano sull'eliminazione delle parti nascoste implicano un ordinamento.

L'ordinamento principale è basato sulla distanza tra un lato (Edge) o una superficie (Face) e il punto di osservazione (Vp).

Un oggetto vicino all'osservatore avrà più probabilità di essere visibile di un oggetto lontano; dopo aver determinato la distanza si procede all'ordinamento in senso locale per determinare se l'oggetto è nascosto da quelli più vicini.

L'efficienza di questi algoritmi dipende quindi dall'efficienza delle tecniche di ordinamento.

Per aumentare questa efficienza si usa il concetto di coerenza, cioè la tendenza di parti della scena a rimanere costanti nello spazio.

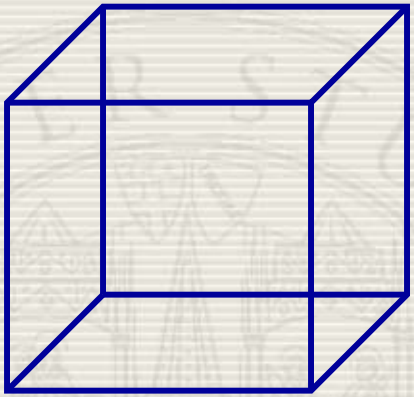
Hidden Line/Surface Removal

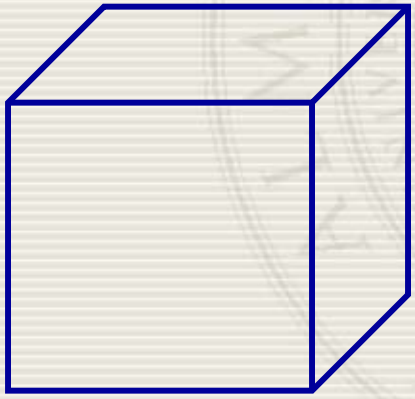
Classificazione: Sutherland, Sproull, Schumaker (1974):

- **Object Space**
  - –Calcoli geometrici su poligoni nello spazio 3D
  - –Precisione Floating point
  - –Si deve processare la scena nell'ordine degli oggetti
- **Image Space**
  - –Visibilità al pixel
  - –Precisione Intera
  - –Si deve processare la scena nell'ordine delle immagini

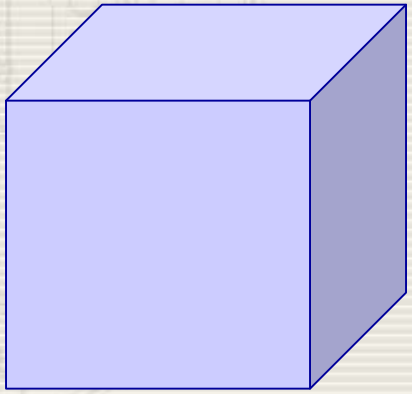
Hidden Line/Surface Removal

WireFrame





Hidden Line



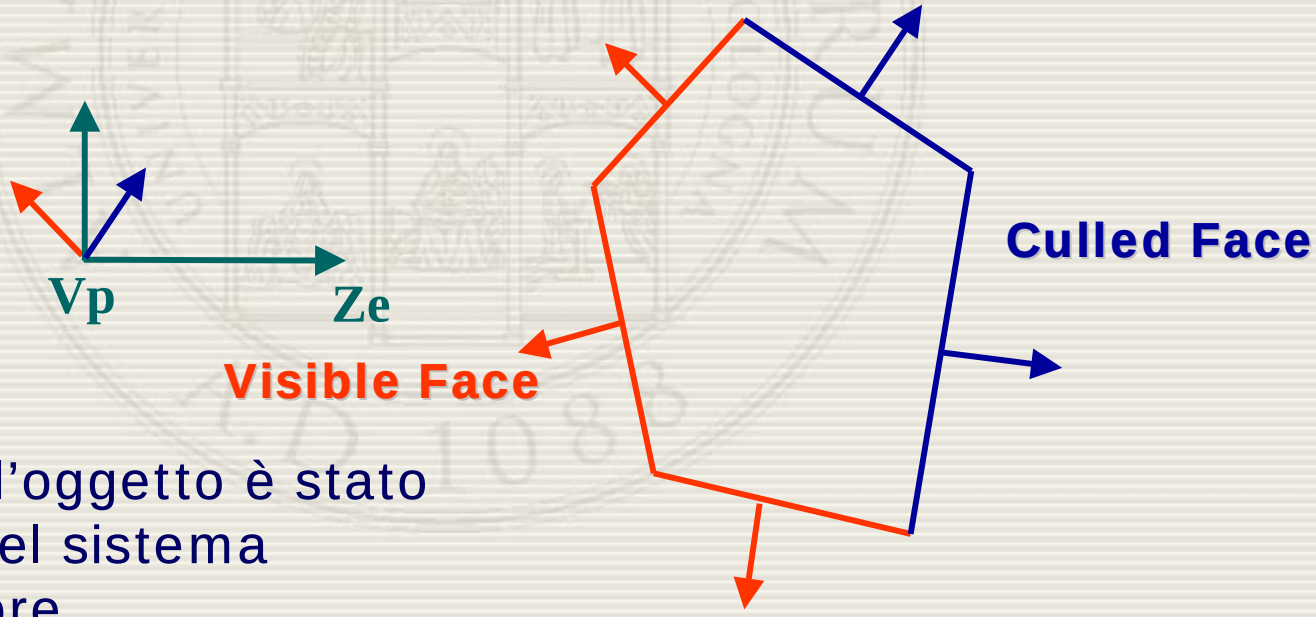
Hidden Surface

Algoritmo Back Face Culling (HS)

Si applica ad oggetti (Mesh 3D) convessi:

•I Vertici delle Facce siano dati in senso antiorario rispetto a chi guarda dall'esterno; la normale calcolata punta allora verso l'esterno;

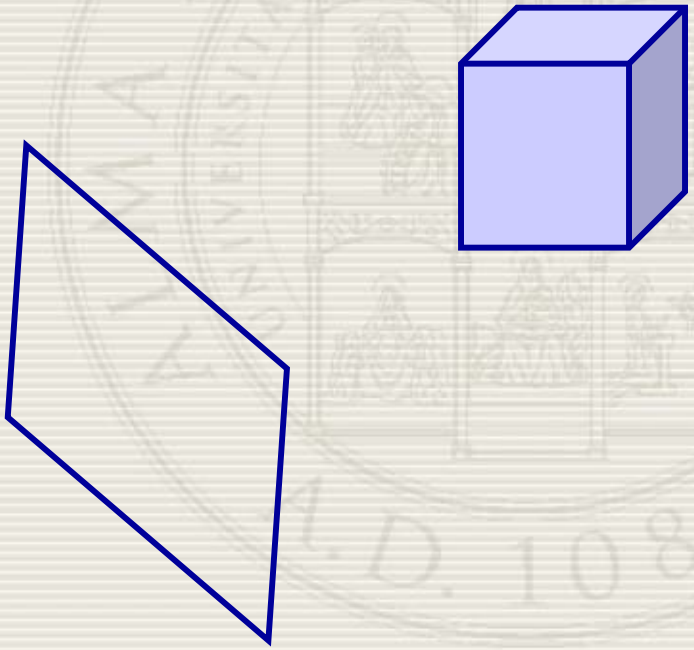
•Si testi la componente Z della normale di ogni faccia;
se negativa la faccia non dovrà essere disegnata perché non visibile dall'Osservatore;



Esempio 2D; l'oggetto è stato trasformato nel sistema dell'Osservatore.

Backface Culling

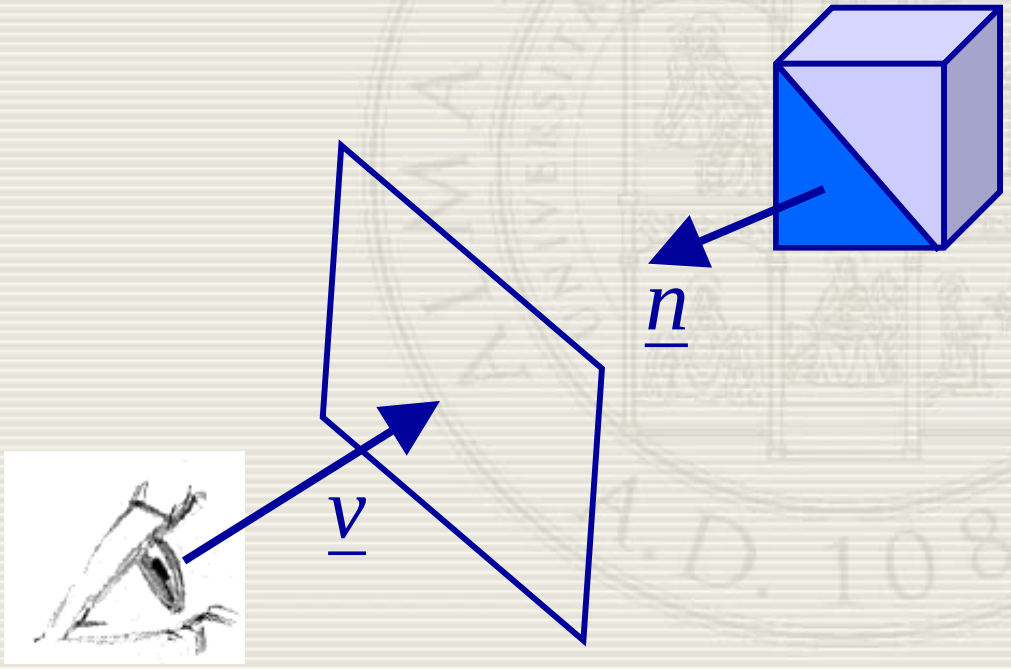
Più in generale:



Backface Culling

Più in generale:

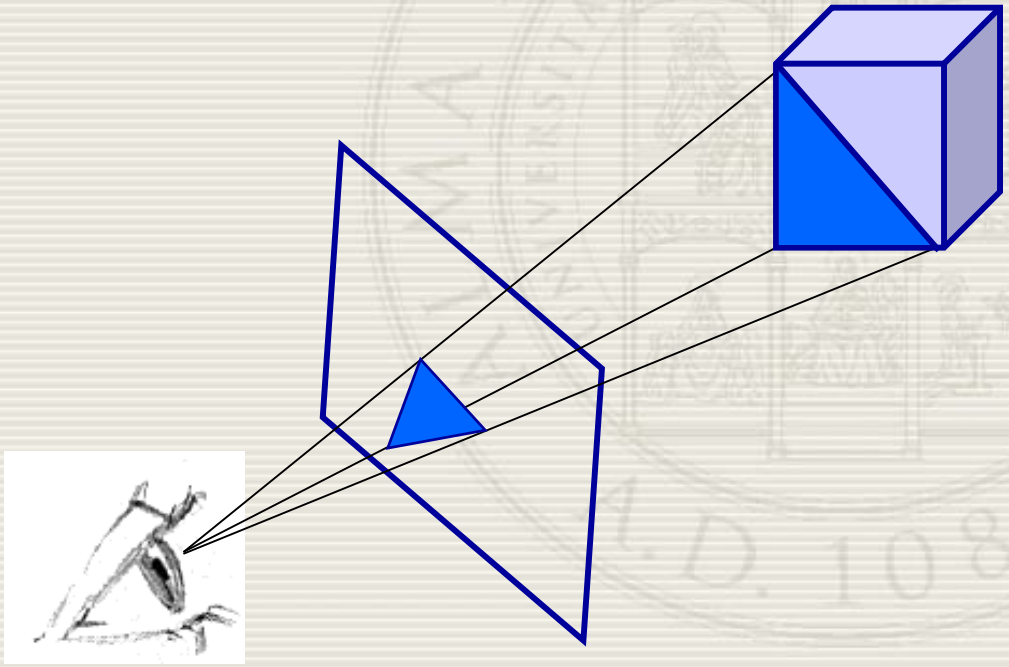
$\underline{n} \cdot \underline{v} < 0$, allora processa faccia



Backface Culling

Più in generale:

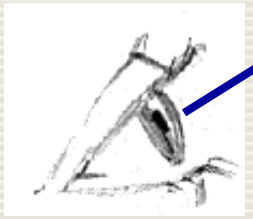
... proiezione e disegno

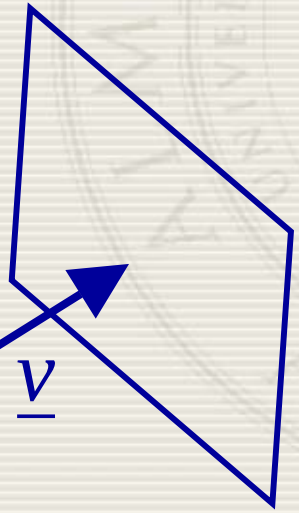


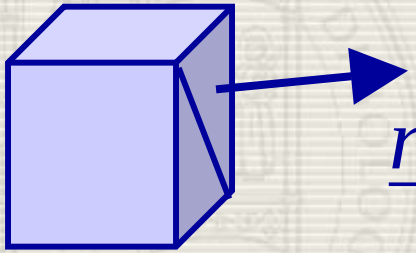
Backface Culling

Più in generale:

$\underline{n} \cdot \underline{v} \geq 0$, allora elimina faccia





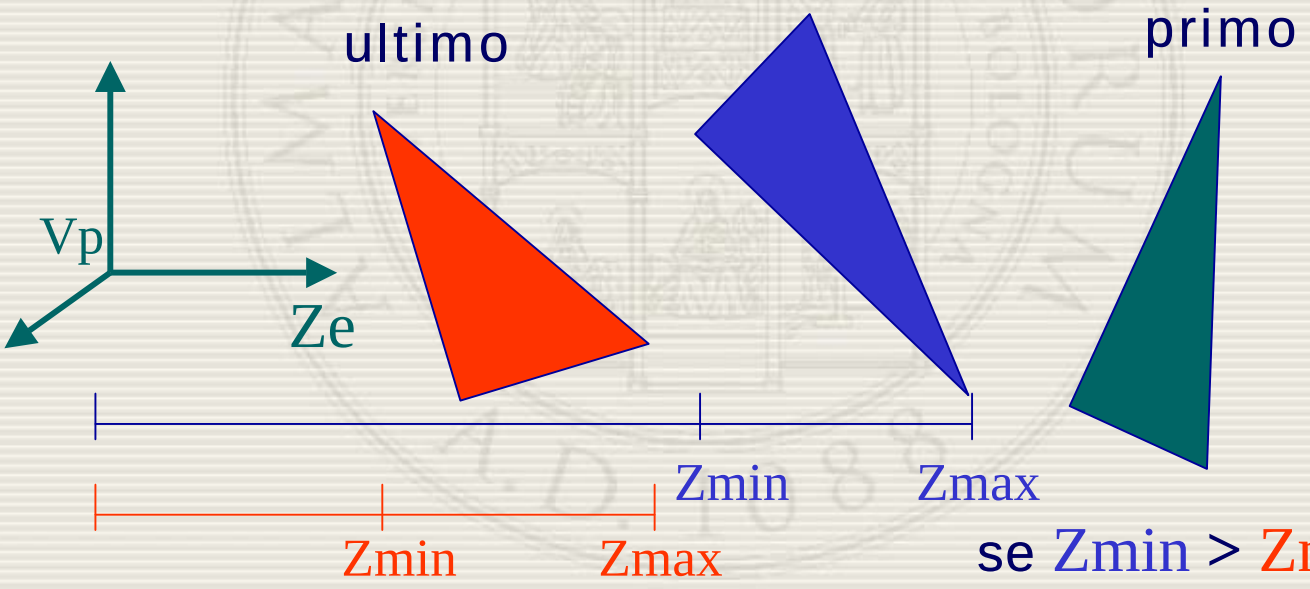


Backface Culling

- Vantaggi
  - Aumenta la velocità di rendering rimuovendo circa la metà della facce, che quindi non verranno processate (proiezione e disegno/scan conversion)
  - Non serve l'ordinamento delle facce
- Svantaggi
  - Funziona solo per superfici chiuse, convesse e senza buchi;
  - Non può essere considerato un vero algoritmo di Hidden Surfaces

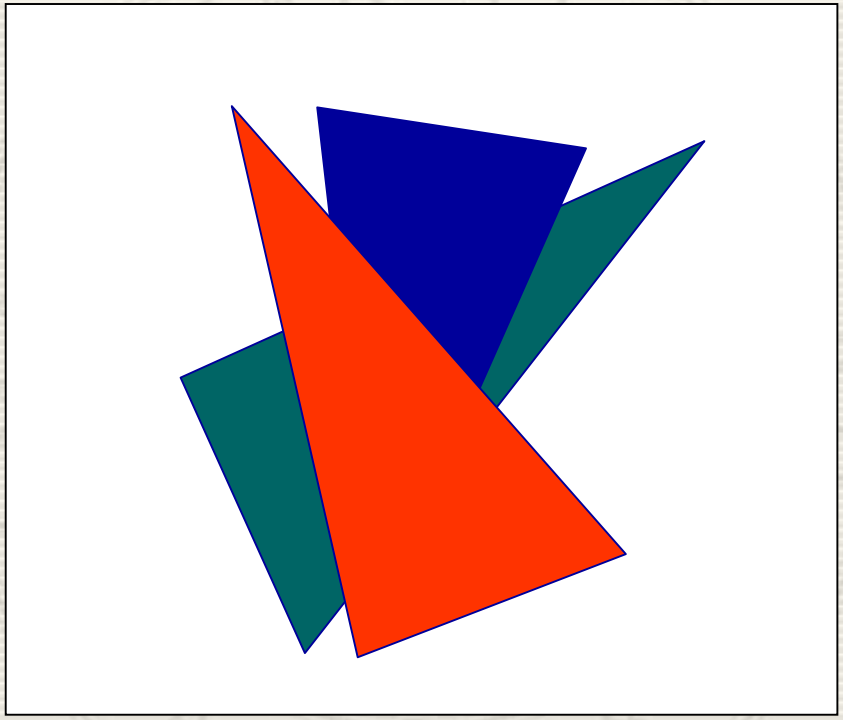
Algoritmo del Pittore

- Si ordinano i triangoli dal più lontano al più vicino rispetto all'Osservatore (usando la coordinata Ze nel sistema dell'Osservatore)
- Si rasterizzano (scan conversion) i triangoli secondo l'ordinamento così determinato



se Zmin > Zmax allora l'ordine è blu e rosso

Algoritmo del Pittore



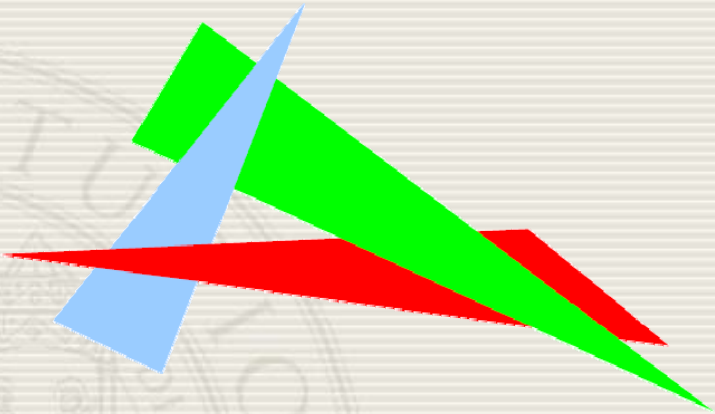
Fase di disegno

Nota: vengono disegnati tutti pixel di tutte le facce

Algoritmo del Pittore

Casi in cui non funziona:

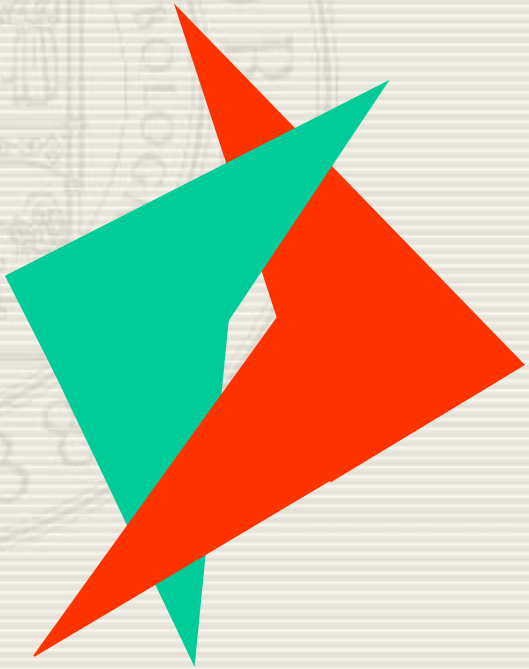
- Intersezioni
- Cicli



Si risolvono suddividendo i triangoli, ma questo è difficile e costoso;

Complessità:

- Ordinamento

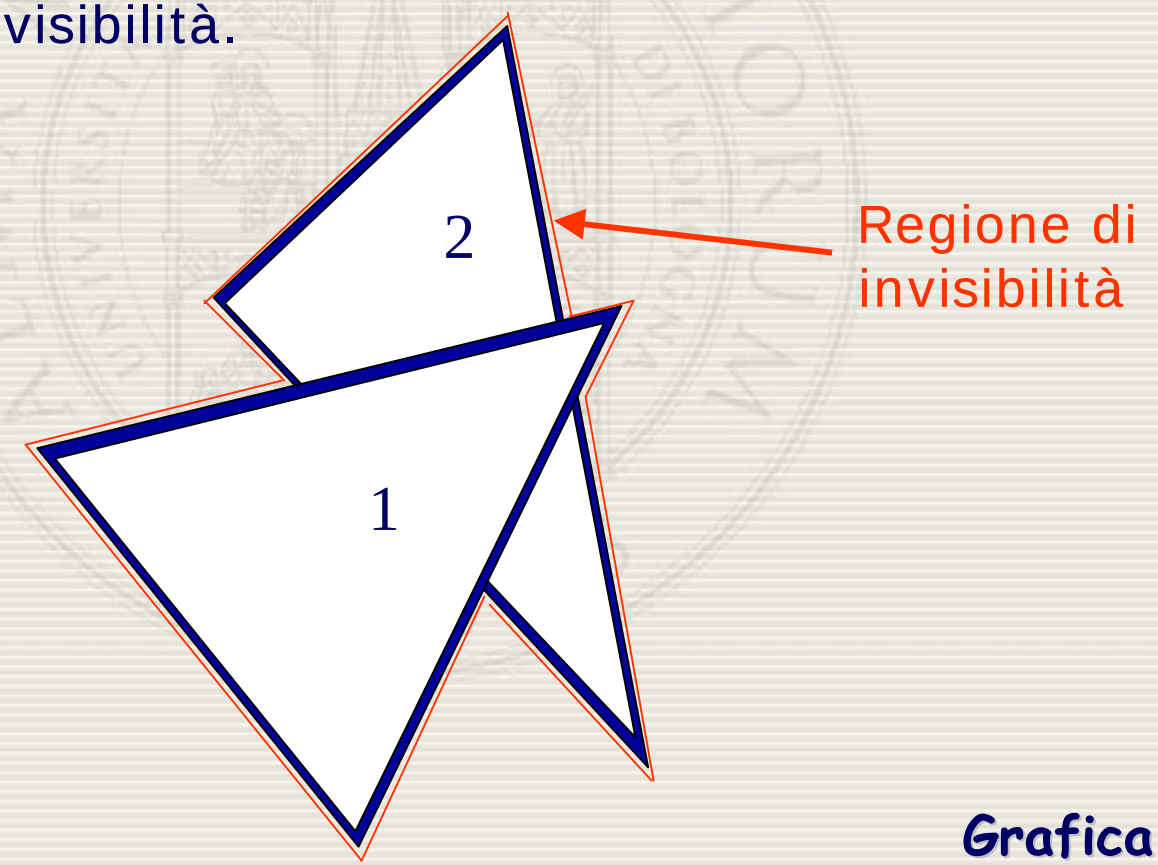


Algoritmo del Pittore

- Vantaggi
  - Si basa su un semplice agoritmo di ordinamento di poligoni
- Svantaggi
  - E' difficile definire un criterio di ordinamento
  - Ridisegna certi pixel molte volte
  - L'ordinamento può esere costoso

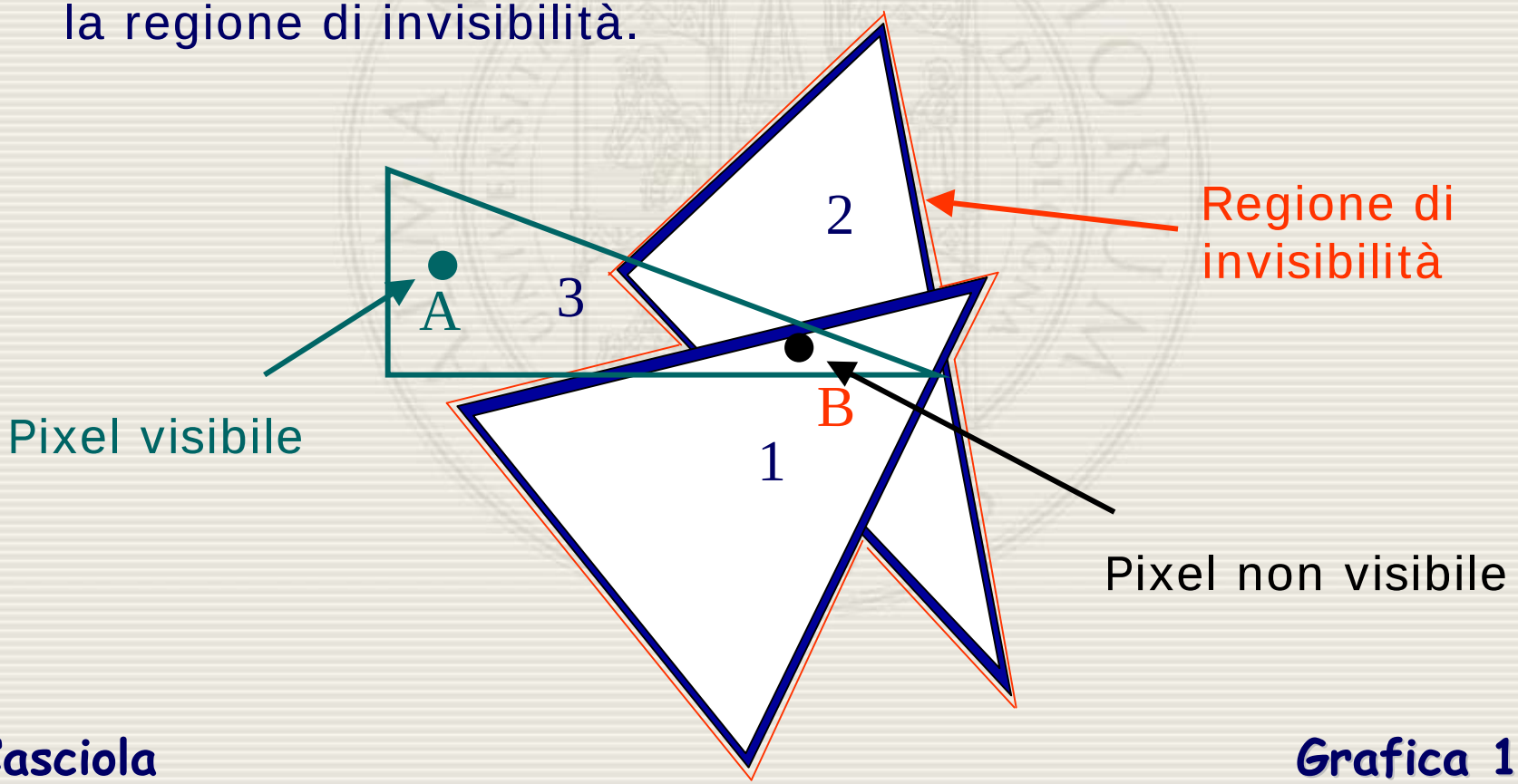
Visibility Buffer

Si ordinano i triangoli dal più vicino al più lontano rispetto all'Osservatore (usando la coordinata Ze nel sistema dell'Osservatore)
Si rasterizzano (scan conversion) i triangoli secondo l'ordinamento così determinato pixel per pixel, testando la regione di invisibilità.



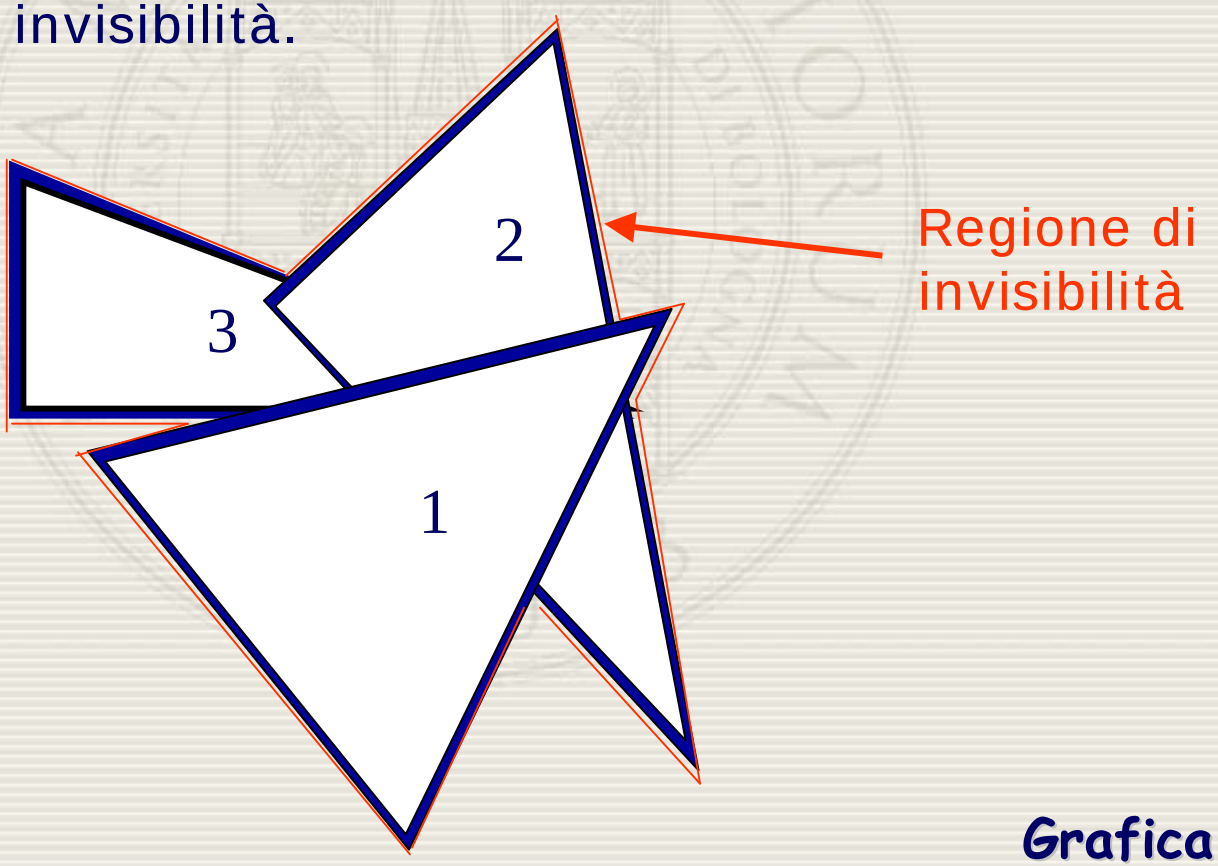
Visibility Buffer

Si ordinano i triangoli dal più vicino al più lontano rispetto all'Osservatore (usando la coordinata Ze nel sistema dell'Osservatore)
Si rasterizzano (scan conversion) i triangoli secondo l'ordinamento così determinato pixel per pixel, testando la regione di invisibilità.



Visibility Buffer

Si ordinano i triangoli dal più vicino al più lontano rispetto all'Osservatore (usando la coordinata Ze nel sistema dell'Osservatore)
Si rasterizzano (scan conversion) i triangoli secondo l'ordinamento così determinato pixel per pixel, testando la regione di invisibilità.



Visibility Buffer

- Vantaggi
  - Si basa su un semplice algoritmo di ordinamento di poligoni
  - Ogni pixel viene disegnato una sola volta
- Svantaggi
  - E' difficile definire un criterio di ordinamento
  - L'ordinamento può esere costoso
  - Si deve memorizzare la regione di invisibilità

Algoritmo Z-buffer

E' l'algoritmo di eliminazione di superfici nascoste implementato sulle GPU.
Fu proposto originariamente da Catmull nel 1975 ed è un algoritmo Image Space.

E' una semplice estensione del concetto di Frame Buffer;

- Viene usato un frame buffer (di int) per memorizzare le intensità/colori di ogni pixel;
- Viene usato un buffer, detto Z-buffer, (di float) per memorizzare le coordinate Z o profondità di ogni pixel visibile nello spazio immagine;

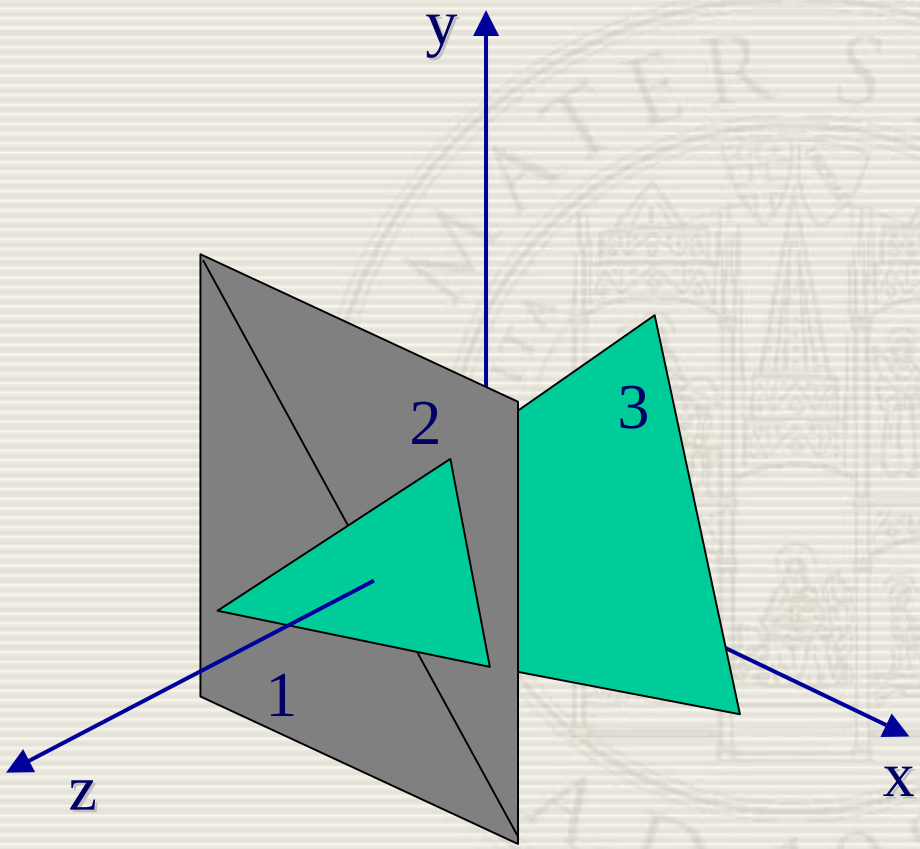
Algoritmo Z-buffer

- Si considera una faccia (triangolo);
- Si applica la trasformazione di vista per ottenere la sua immagine sulla viewport;
- Si rasterizza la faccia e per ogni pixel considerato, si determina la profondità Z del punto 3D che il pixel rappresenta, in coordinate dell'osservatore;
- Si confronta tale profondità con quella memorizzata nello Z-buffer in corrispondenza di quel pixel;
- Se dal confronto risulta che il nuovo pixel (punto) ha profondità minore, allora si memorizza la sua profondità nello Z-buffer e la sua intensità/colore nel frame buffer.

Algoritmo Z-buffer

```
void zbuffer() {

for (y=0; y<Vymax; y++)
  for(x=0; x<Vxmax; x++) {
    FrameBuffer(x,y)=BACK_Col;;
    ZBuffer(x,y,INF_Val);
 }

 for (every triangle)
   for (all pixel (px,py) of the projected triangle) {

      pZ = computeZ(px,py);
      if (pZ < ZBuffer(px,py))  //pixel is visible
      {
         FrameBuffer(px,py) = computeColor(px,py);
         ZBuffer(px,py) = pZ;
      }

   }

}
```

Esempio



Oggetti 3D

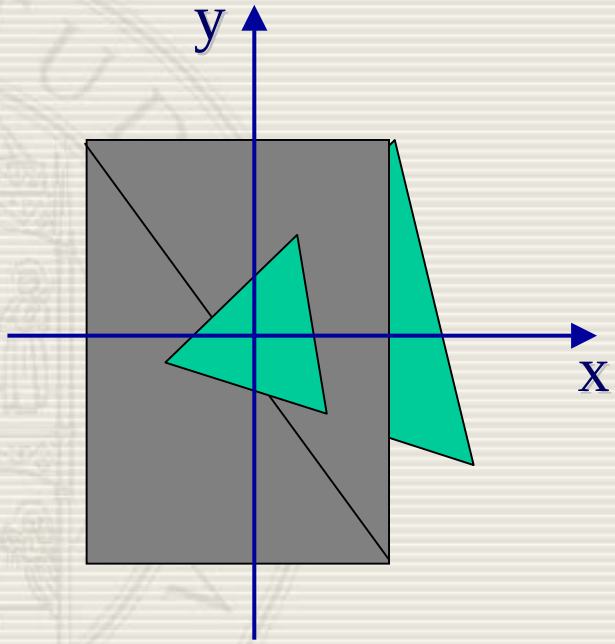
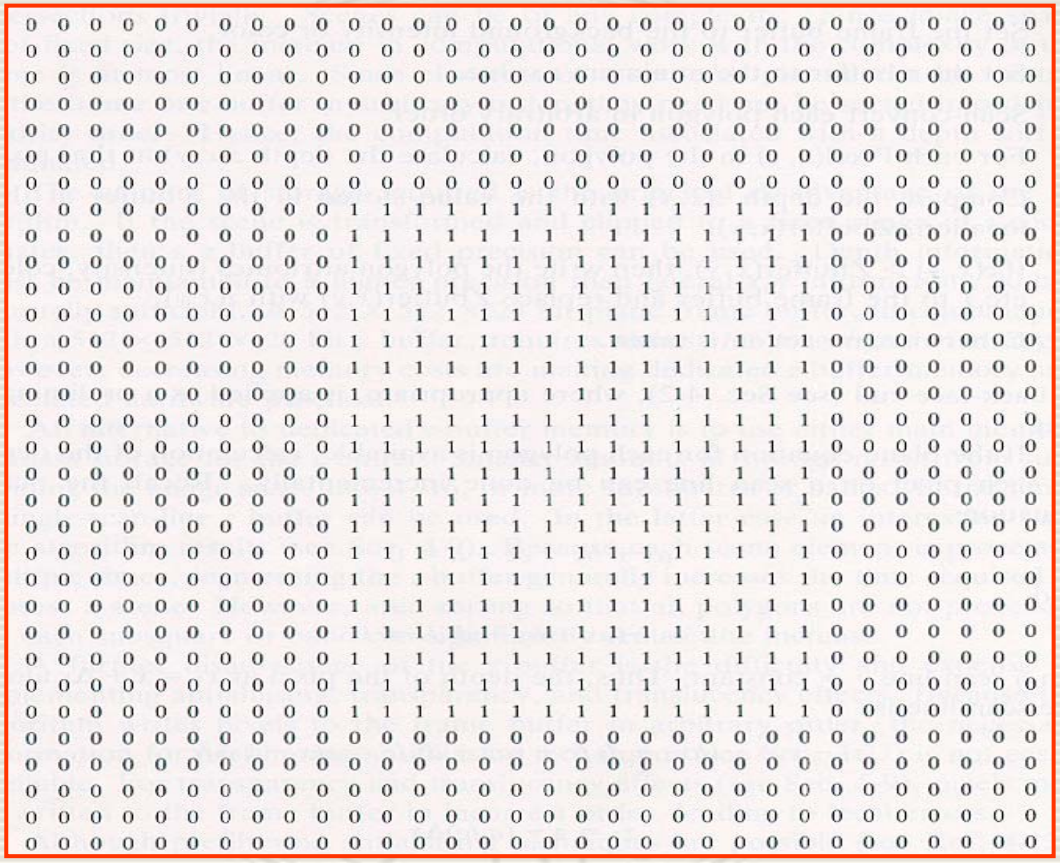


Immagine schermo

Z-Buffer dopo triangoli 1 e 2

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

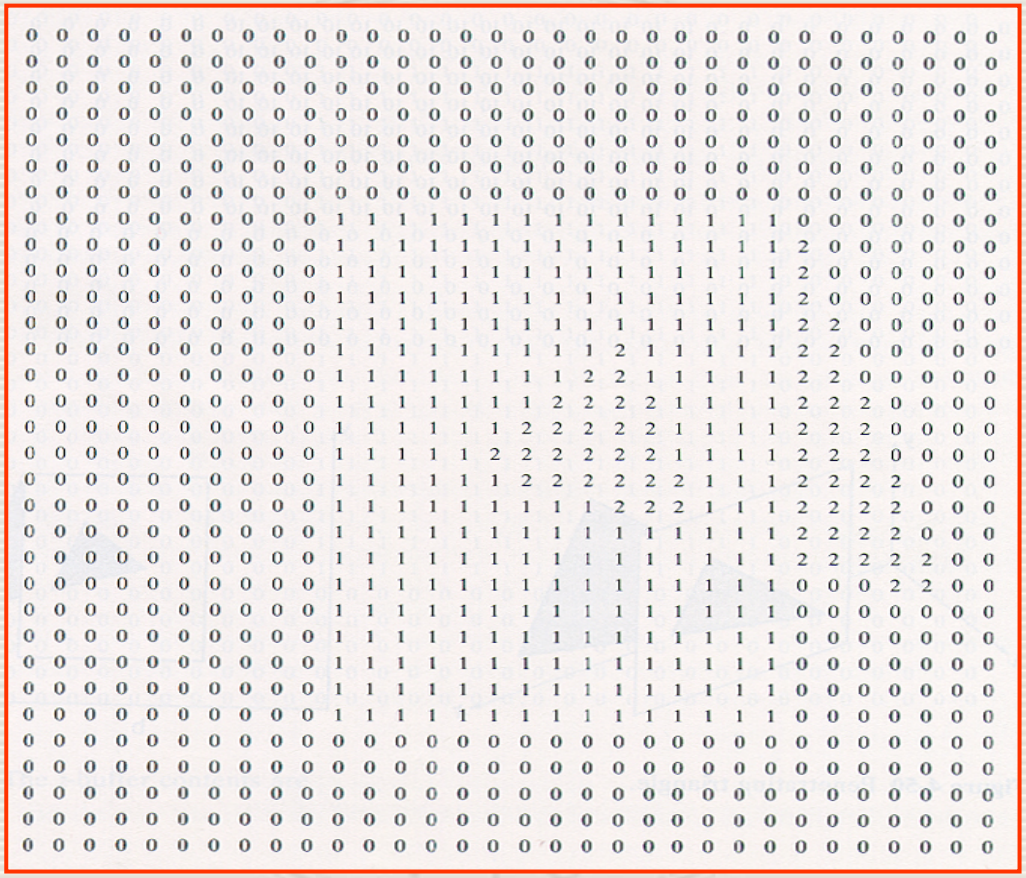
Frame Buffer dopo triangoli 1 e 2



Z-Buffer dopo il triangolo 3

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	5	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	5	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	6	5	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	11	10	10	10	10	10	6	5	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	12	11	10	10	10	10	10	6	6	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	13	12	11	10	10	10	10	10	7	6	5	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	14	13	12	11	11	10	10	10	10	7	6	5	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	15	14	13	12	12	11	10	10	10	10	7	6	6	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	14	13	13	12	11	10	10	10	10	7	7	6	5	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	12	11	11	10	10	10	8	7	6	5	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	8	7	6	6	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	8	7	7	6	5	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	6	5	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Frame Buffer dopo il triangolo 3



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	1	1	1	1	1	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	1	1	1	1	2	2	2	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	1	1	1	1	2	2	2	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	2	1	1	1	1	2	2	2	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	1	1	1	2	2	2	2	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2	1	1	1	2	2	2	2	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	2	2	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Algoritmo Z-buffer

- Vantaggi
  - Semplice da implementare e non prevede alcun ordinamento
  - Permette uno “streaming approach” per il disegno dei poligoni
- Svantaggi
  - Richiede spazio di memoria aggiuntivo
  - C’è ancora una sorta di ridisegno

Algoritmo Z-buffer

Indice argomenti che affronteremo:

- Introduzione alle coordinate baricentriche;
- Profondità di un pixel;
- Pipeline Grafica per Z-buffer
- Rasterizzazione o Scan Conversion
  - con colore
  - con texture
- Clipping di punti, segmenti e poligoni
- Modelli di illuminazione
- Algoritmo di shading (sfumatura colore)