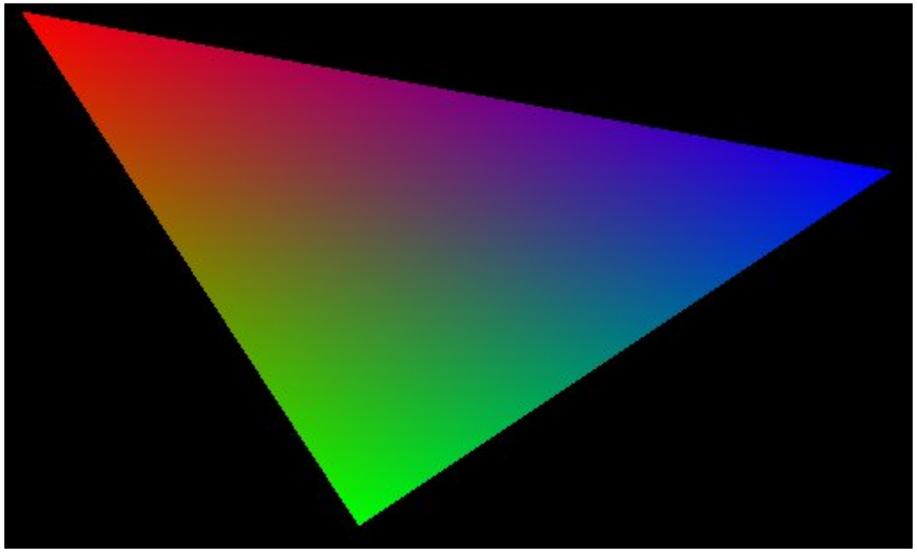
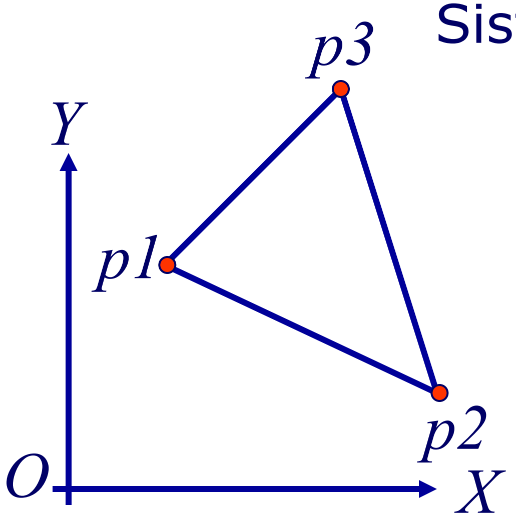
Algoritmi di Rasterizzazione



Sistema di Riferimento 2D

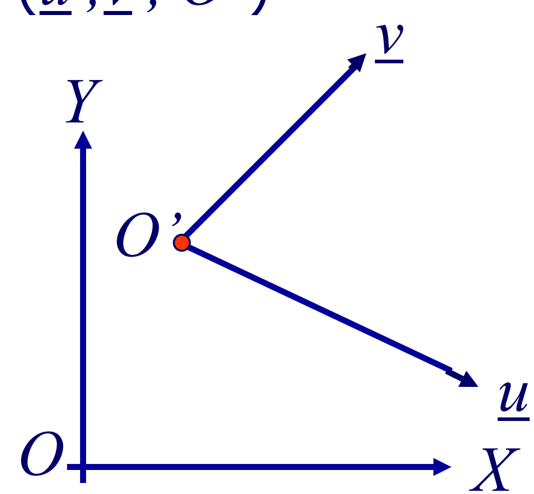
Riprendiamo i concetti di Spazio Affine e Sistema di Riferimento. Per esempio, nel caso 2D, si ha che dato un triangolo si può definire un sistema di riferimento a lui associato.

Il triangolo sia dato mediante tre vertici non allineati $p1$, $p2$ e $p3$ (dati in senso antiorario) in un sistema di coordinate cartesiane XYO.



Allora possiamo definire il seguente Sistema di Riferimento: $(\underline{u}, \underline{v}, O')$

$$O' = p1$$
$$\underline{u} = p2 - p1$$
$$\underline{v} = p3 - p1$$



Sistema Baricentrico

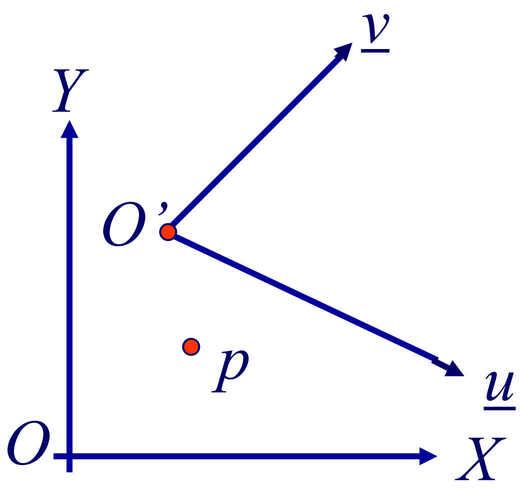
I punti p di coordinate cartesiane $[x,y]$, possono allora essere rappresentati nel nuovo sistema di riferimento non ortogonale/cartesiano, ma affine, come:

$$p = p1 + \beta\,(p2\text{-}p1) + \gamma\,(p3\text{-}p1)$$

e le sue coordinate saranno: $p = [\beta\,,\ \gamma,\ 1]$.

A partire dalla rappresentazione per p possiamo scrivere:

$$\begin{aligned} p &= p1 + \beta\,(p2\text{-}p1) + \gamma\,(p3\text{-}p1) \\ &= (1\text{-}\ \beta\ \text{-}\ \gamma)p1 + \beta\,p2 + \gamma\,p3 \\ &= \alpha\,p1 + \beta\,p2 + \gamma\,p3 \end{aligned}$$



che porta a

$$p(\alpha,\ \beta\,,\ \gamma) = \alpha\,p1 + \beta\,p2 + \gamma\,p3$$

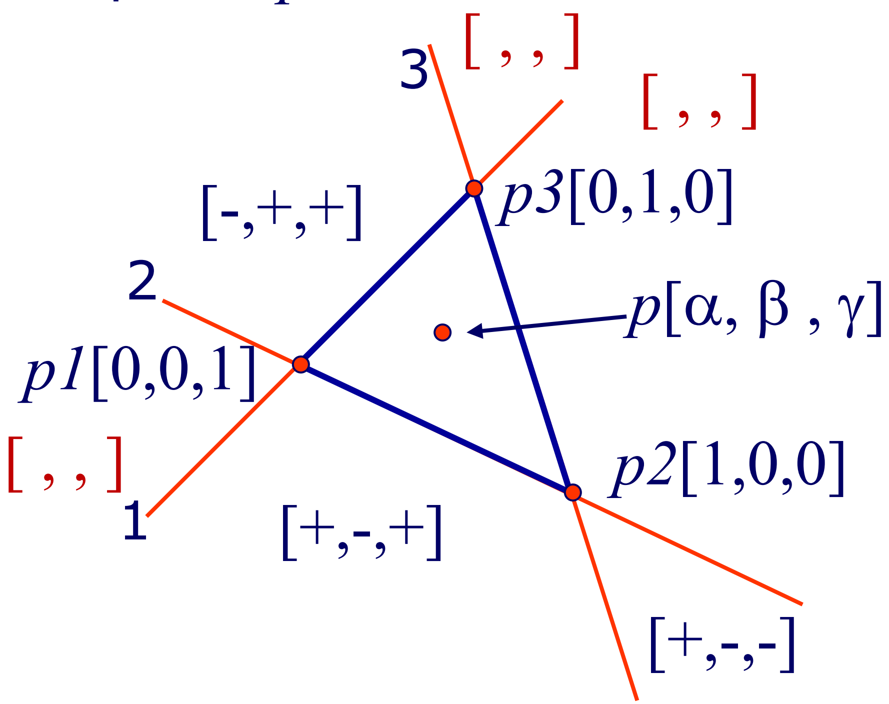
con

$$\alpha + \beta + \gamma = 1$$

Siamo allora in un sistema di riferimento baricentrico e $[\alpha,\ \beta\,,\ \gamma]$ sono dette coordinate baricentriche di p.

Sistema Baricentrico

Osservazione: le coordinate baricentriche descrivono un punto p come combinazione affine dei vertici del triangolo.



- Per ogni punto p interno al triangolo di vertici $p1$, $p2$ e $p3$ si ha:

$$0 < \alpha < 1$$
$$0 < \beta < 1$$
$$0 < \gamma < 1$$

- Punti su un lato del triangolo hanno una coordinata nulla
- Le coordinate di un vertice del triangolo sono due 0 ed un 1

Esercizio: si osservino le zone del piano con coordinate negative; completare le mancanti.

Coordinate Baricentriche

Osservazione: dati due punti $p1$ e $p2$ resta definito un segmento; possiamo definire il sistema baricentrico $(t,O')=(p2-p1,p1)$ così che ogni punto p sulla retta per $p1$ e $p2$ potrà essere espresso in coordinate baricentriche come:

$$p = \alpha\, p1 + \beta\, p2 \quad \text{con} \quad \alpha + \beta = 1.$$

Ma questo lo abbiamo già visto in precedenza e altro non è che la rappresentazione in forma parametrica di un segmento:

$$p(t) = (1-t)\, p1 + t\, p2 \quad \text{con} \quad t \in [0, 1].$$

Problema: dato il punto $p=[p_x,p_y]$ sulla retta per $p1$ e $p2$ quali sono le sue coordinate baricentriche?

Dall'espressione in forma parametrica ricaviamo il valore di t da una delle due componenti; la prima sarà:

$$p_x = (1-t)\, p1_x + t\, p2_x = t\, p1_x + t\,(p2_x - p1_x)$$

da cui $$t = \frac{p_x - p1_x}{p2_x - p1_x}$$

Coordinate Baricentriche

Problema: dato il punto $p = [p_x, p_y]$ sul piano definito dai tre punti non allineati $p1$, $p2$ e $p3$ quali sono le sue coordinate baricentriche?

Dall'espressione per componenti si ha:

$$\begin{cases} p_x = \alpha\, p1_x + \beta\, p2_x + \gamma\, p3_x \\ p_y = \alpha\, p1_y + \beta\, p2_y + \gamma\, p3_y \\ \alpha + \beta + \gamma = 1 \end{cases}$$

Si tratta di un sistema lineare, non singolare, nelle incognite α, β e γ

$$\begin{pmatrix} p1_x & p2_x & p3_x \\ p1_y & p2_y & p3_y \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix}$$

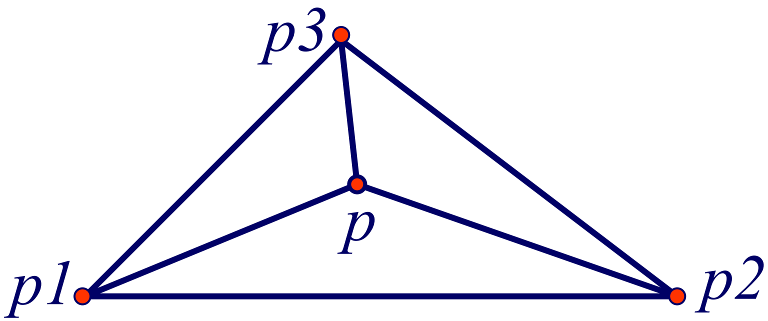
Coordinate Baricentriche

Risolvendo con Cramer (forma esplicita) si ha:

$$\alpha = \frac{< p, p2, p3 >}{< p1, p2, p3 >} \quad \beta = \frac{< p1, p, p3 >}{< p1, p2, p3 >} \quad \gamma = \frac{< p1, p2, p >}{< p1, p2, p3 >}$$

con $<A,B,C>$=Area del triangolo di vertici A, B e C dati in senso antiorario, ossia:

$$< A, B, C > = \frac{1}{2} \det \begin{pmatrix} A_x & B_x & C_x \\ A_y & B_y & C_y \\ 1 & 1 & 1 \end{pmatrix}$$



Nota: α, β, e γ sono ottenute come il rapporto delle aree dei triangoli individuati da p, $p1$, $p2$ e $p3$.

Coordinate Baricentriche

Le espressioni viste

$$p(\alpha, \beta) = \alpha\, p1 + \beta\, p2 \qquad \alpha,\beta \ in\ [0,1],\ \ \alpha + \beta = 1$$
$$p(\alpha, \beta, \gamma) = \alpha\, p1 + \beta\, p2 + \gamma\, p3 \qquad \alpha,\beta,\gamma \ in\ [0,1],\ \alpha + \beta + \gamma = 1$$

rappresentano il segmento di retta ed il triangolo piano, in forma parametrica, rispettivamente di estremi i due punti $p1$ e $p2$ e i tre punti non allineati $p1$, $p2$ e $p3$.

Si osservi che i punti possono essere in R^n (cioè una qualunque dimensione) e le espressioni continueranno a valere.

Noi siamo interessati ad usarle sia in R^2 che in R^3.

Coord. Baricentrice e rasterizzazione

Dato un triangolo in coordinate schermo, la sua rasterizzazione consiste nel disegnare con un colore tutti i pixel del triangolo (i pixel interni e/o sui lati del triangolo).

per y=v_{ymin},…,v_{ymax}

per x=v_{xmin},…,v_{xmax}

$p=[x,y]$

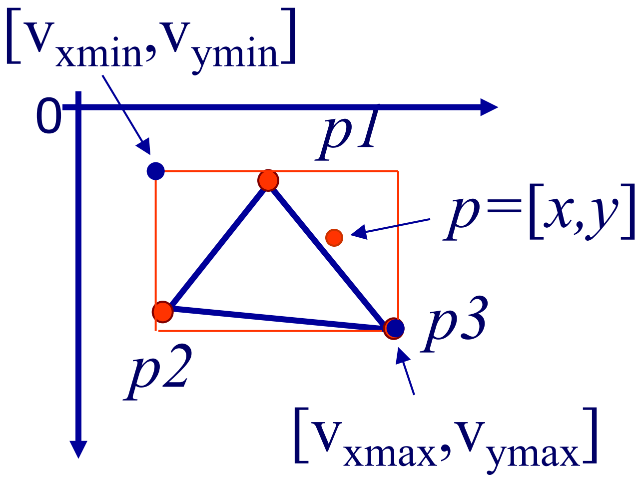
$$\alpha = \frac{< p, p2, p3 >}{< p1, p2, p3 >}$$

$$\beta = \frac{< p1, p, p3 >}{< p1, p2, p3 >}$$

$$\gamma = \frac{< p1, p2, p >}{< p1, p2, p3 >}$$

se $(\alpha >= 0 \;\&\&\; \beta >= 0 \;\&\&\; \gamma >= 0)$

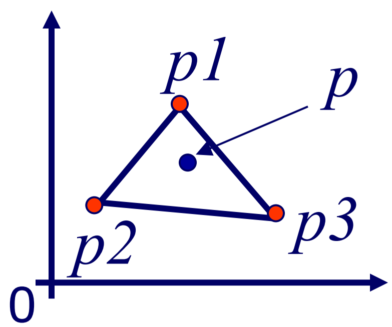
draw(p,c)

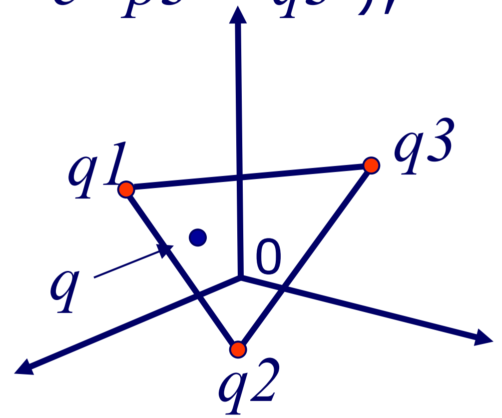


E' costoso? Ogni pixel può essere elaborato in parallelo!
E' perfetto per le GPU.

Coord. Baricentriche e Intepolazione

Osservazione: dati un triangolo 2D ed un triangolo 3D è semplice, usando le coordinate baricentriche, definire un mapping lineare che trasformi i punti del primo in punti del secondo (tale mapping viene definito univocamente chiedendo che $p1 \rightarrow q1$, $p2 \rightarrow q2$ e $p3 \rightarrow q3$);





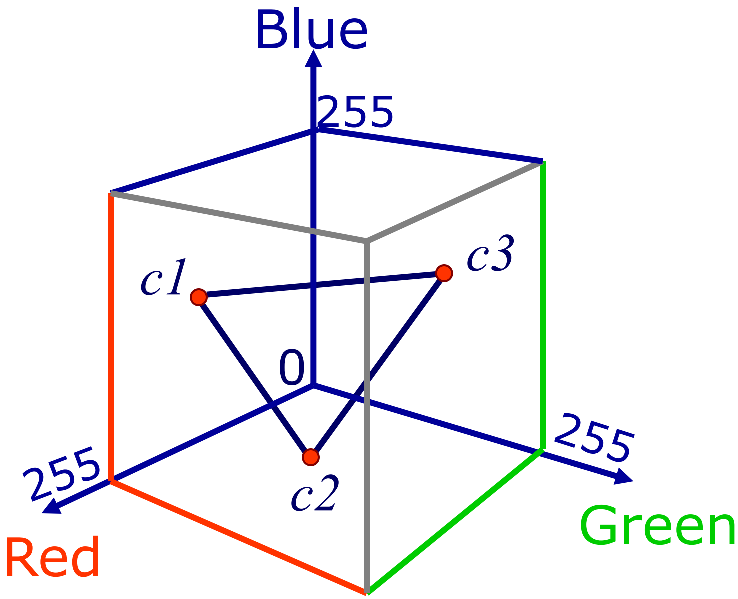
Siano $[\alpha, \beta, \gamma]$ le coordinate baricentriche di p nel triangolo $p1,p2,p3$, allora il suo corrispondente q in $q1,q2,q3$ sarà dato da

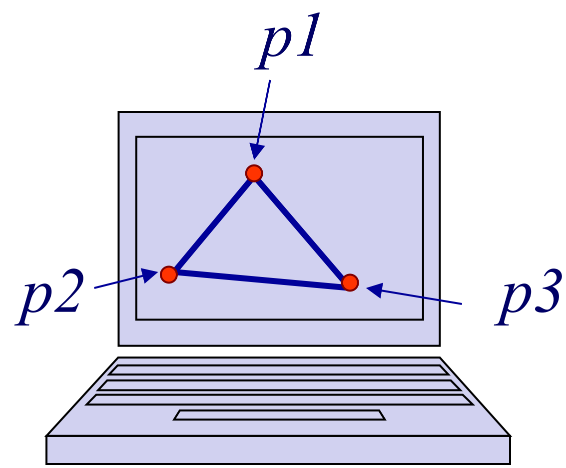
$$q = \alpha\, q1 + \beta\, q2 + \gamma\, q3.$$

Nota: p e q avranno le stesse cordinate $[\alpha, \beta, \gamma]$ nei due triangoli.

Interpolazione Colori

Dati tre colori $c1$, $c2$ e $c3$ nello spazio RGB dei colori, si possono usare le coordinate baricentriche per ottenere l'interpolazione colore.



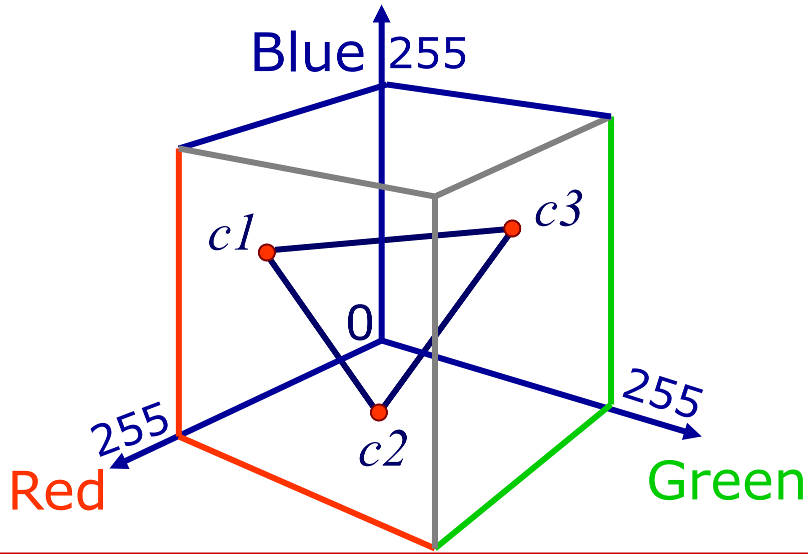


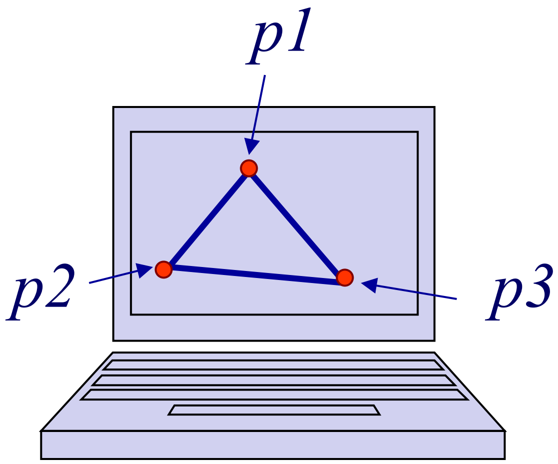
Siano $p1=[x1,y1]$, $p2=[x2,y2]$ e $p3=[x3,y3]$ tre punti in coordinate schermo, $c1=[r1,g1,b1]$, $c2=[r2,g2,b2]$ e $c3=[r3,g3,b3]$ tre colori, nello spazio 3D dei colori, …

Interpolazione Colori

… allora per colorare o rasterizzare un triangolo a schermo, mediante l'interpolazione colore, si può seguire la seguente procedura:

-per ogni pixel $p=[x,y]$ del triangolo schermo;
- -determinare le coordinate baricentriche $p(\alpha, \beta, \gamma)$ rispetto a $p1$, $p2$ e $p3$;
- -calcolare $c = \alpha\, c1 + \beta\, c2 + \gamma\, c3$;
- -disegnare il pixel p con il colore c: draw(p,c)





Interpolazione Colori

per y=v_{ymin},…,v_{ymax}

per x=v_{xmin},…,v_{xmax}

$p=[x,y]$

$$\alpha = \frac{< p, p2, p3 >}{< p1, p2, p3 >}$$

$$\beta = \frac{< p1, p, p3 >}{< p1, p2, p3 >}$$

$$\gamma = \frac{< p1, p2, p >}{< p1, p2, p3 >}$$

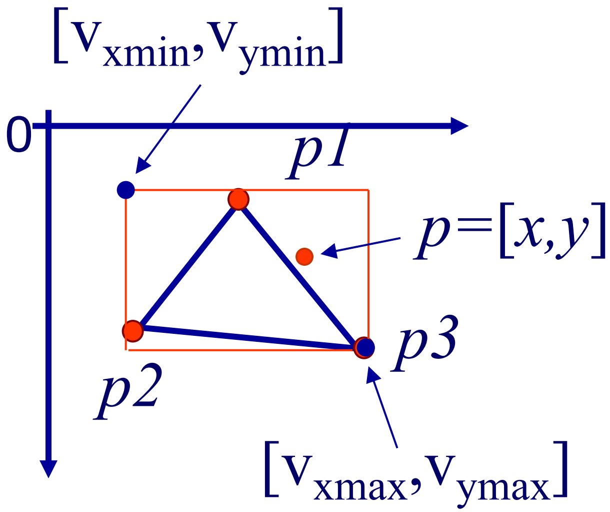
se $(\alpha >=0 \ \&\& \ \beta >=0 \ \&\& \ \gamma >=0)$

$c_x = \alpha \ r1 + \beta \ r2 + \gamma \ r3$

$c_y = \alpha \ g1 + \beta \ g2 + \gamma \ g3$

$c_z = \alpha \ b1 + \beta \ b2 + \gamma \ b3$

draw(p,c)



Esempio

Cartella HTML5_2d_2, codice:

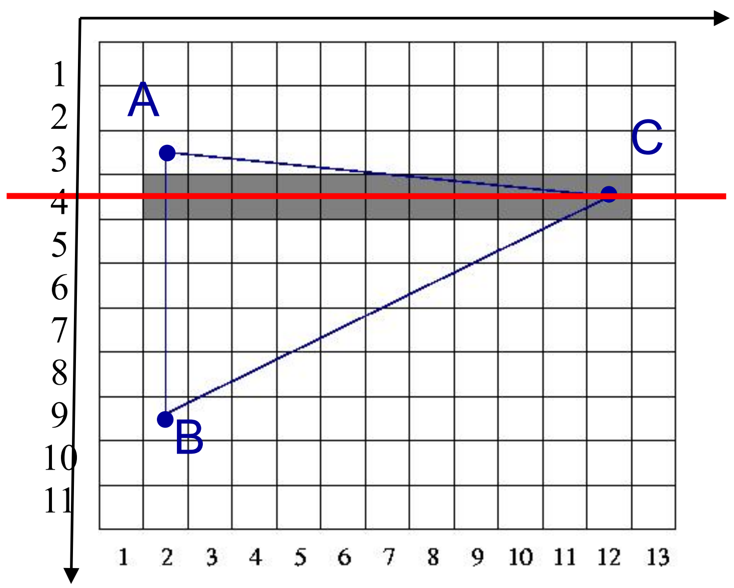
raster_draw_color.html e .js

Rasterizzazione e Scan Conversion

Si vuole esaminare un algoritmo di scan conversion per rasterizzare triangoli. L'idea base dell'algoritmo consiste nel determinare le sequenze orizzontali di pixel (scan-line) del triangolo

Per ogni scan-line che ha a che fare con il triangolo:

- Trovare l'intersezione della scan-line con i due lati del triangolo
- Accendere i pixel della scan-line fra le due intersezioni trovate

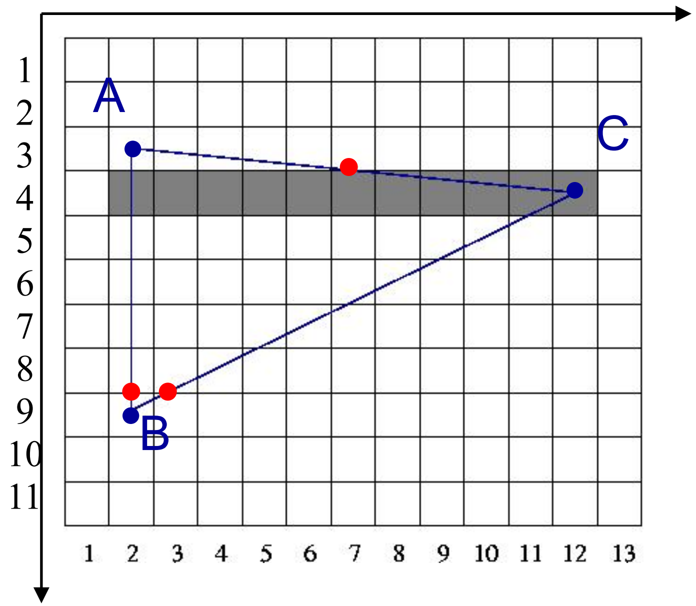


Per determinare le intersezioni di una scan-line con i lati del triangolo si utilizza l'algoritmo di linea incrementale con incremento unitario in Y

Un algoritmo di Rasterizzazione

Osservazioni sulla fase di intersezione:

- i lati orizzontali del triangolo vengono scartati perché non producono intersezioni;
- Tutti i lati vengono accorciati per garantire che ogni scan-line intersecata con il triangolo fornisca solo due intersezioni;
- utilizzo dell'algoritmo di linea incrementale per determinare le intersezioni in modo semplice.



I lati abbiano estremi (x0,y0) e (x1,y1) e (x0,y0) sia sempre l'estremo con ordinata minore, allora:

n=y1-y0 (numero di scan-line – 1)

m=n/(x1-x0) (pendenza)

dx=1/m (inverso della pendenza)

dy=1 (incremento scan-line)

$$x_{i+1} = x_i + dx$$

$$y_{i+1} = y_i + dy$$

Algoritmo di Linea Incrementale

Sia P0 = (8 , 1) e P1 = (1 , 4)

applichiamo l'algoritmo di linea incrementale:

sarà:

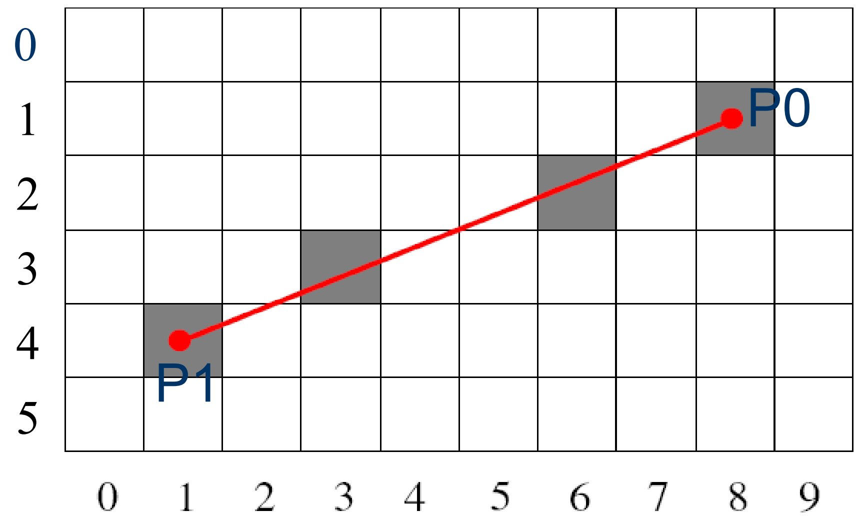
n=3, m=-3/7, dx=1/m=-7/3 ≅ -2.333, dy=1

--> (8,1)

(5.666,2) --> (6,2)

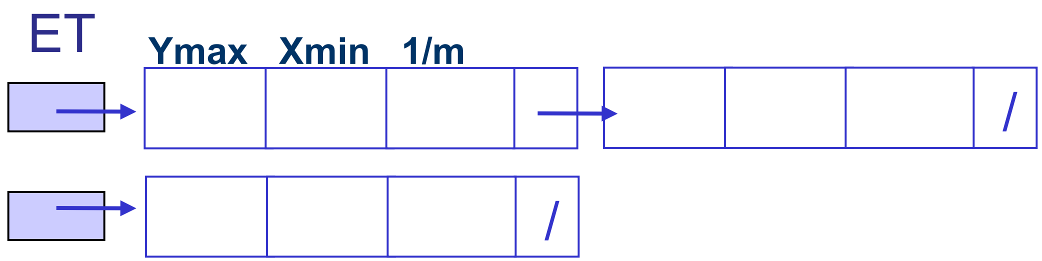
(3.333,3) --> (3,3)

(1.000,4) --> (1,4)

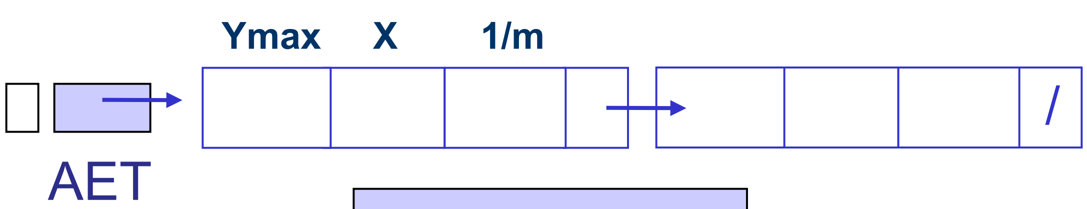


Rasterizzazione di Triangoli

L'algoritmo può essere implementato utilizzando due strutture: Edge Table (ET) e Active Edge Table (AET); l'ET avrà solo due entry e tre record, mentre l'AET avrà solo due record:

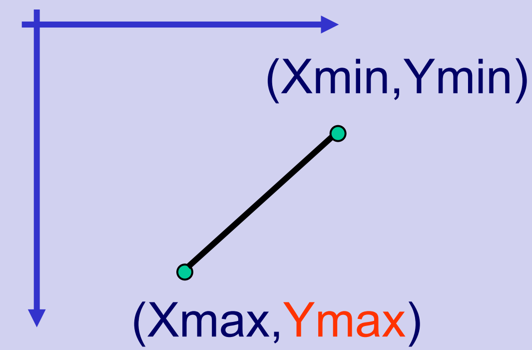


$$\frac{1}{m} = \frac{Xmax-Xmin}{Ymax-Ymin}$$




```
X + = 1/m;
Y ++;
```

Nota: con Xmax si intende la coord. x associata al punto di ordinata maggiore



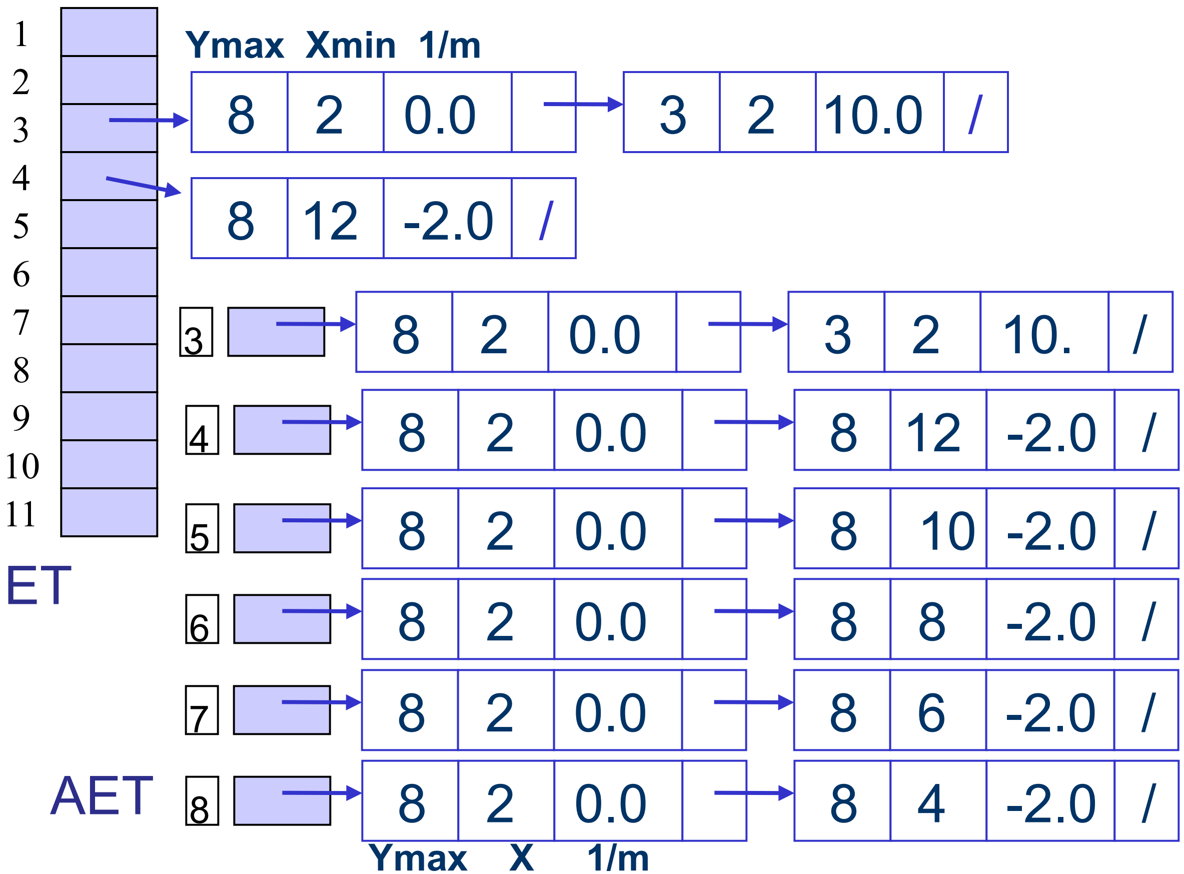
Un algoritmo di Rasterizzazione

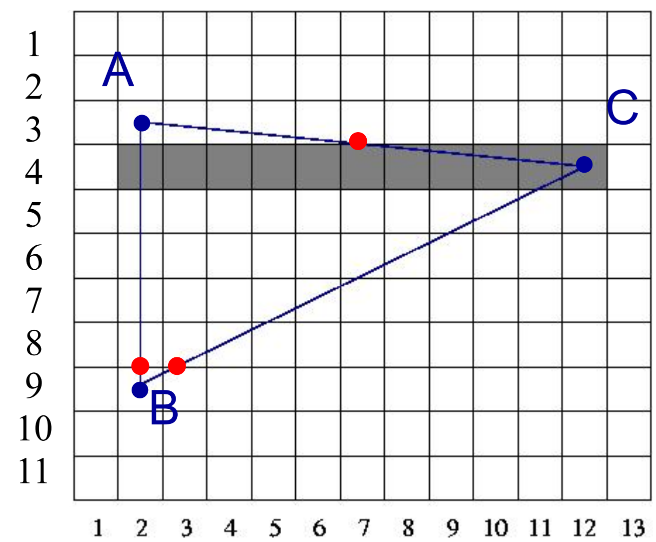
Una volta costruito l'ET, i passi dell'algoritmo sono:

1. Porre Y alla prrima ordinata dell'ET;
2. Inizializzare l'AET a vuoto;
3. Ripetere fino a che l'AET o l'ET sono vuoti:
   - 3.1 muovere l'informazione relativa a Y dall'ET all'AET, mantenendo l'AET ordinato sulle X;
   - 3.2 disegnare sulla scan-line Y i pixel utilizzando coppie di ascisse dall'AET;
   - 3.3 rimuovere dall'AET quelle informazioni per cui Y=Ymax;
   - 3.4 per ciascuna informazione rimasta nell'AET, aggiornare X con X+1/m;
   - 3.5 ordinare l'AET in base alle ascisse;
   - 3.6 Y=Y+1;

Rasterizzazione di Triangoli

Simulazione dell'algoritmo di rasterizzazione.





Esempio

Cartella HTML5_2d_2, codice:

scan_conv_triang.html e .js

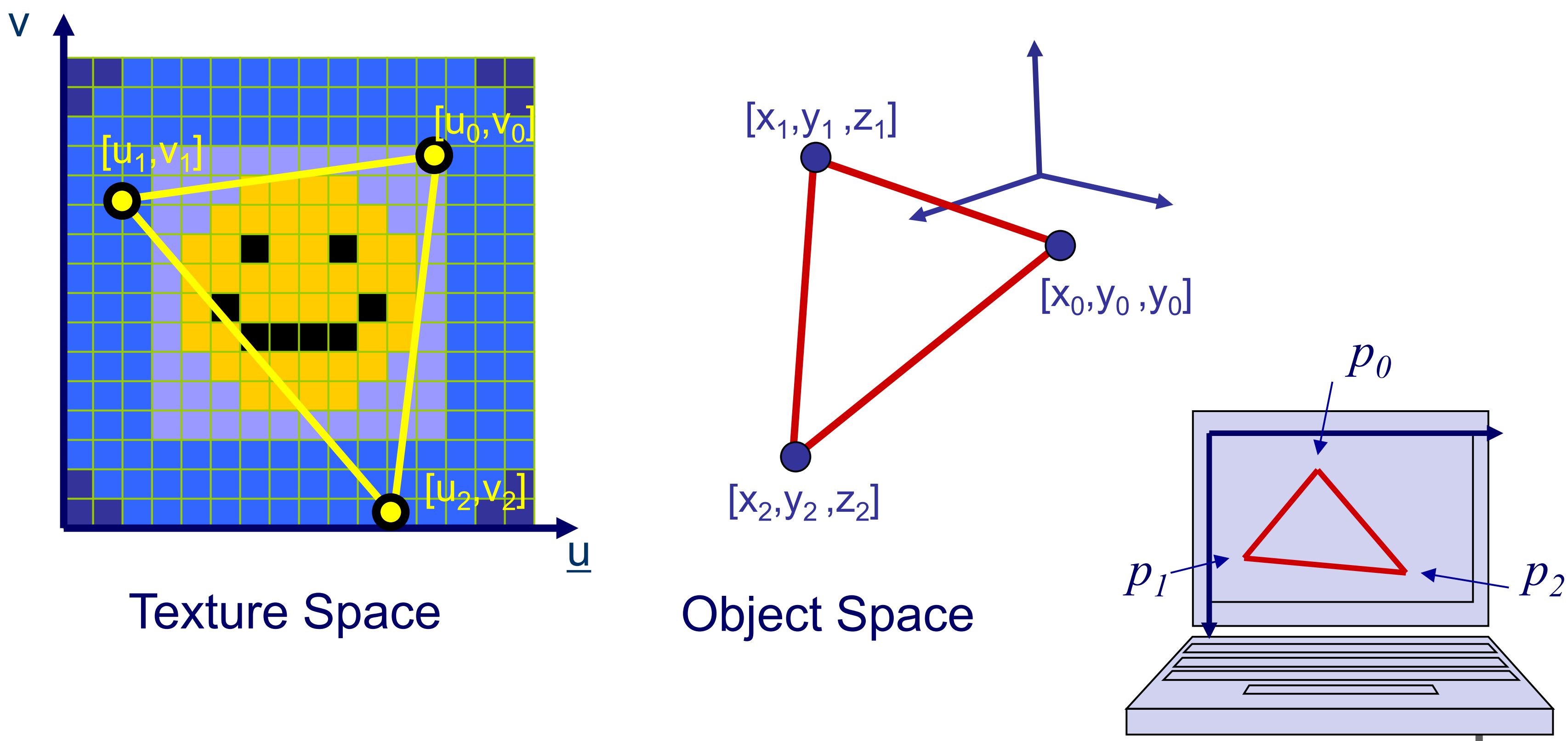
che utilizza:

rasterize_triang.js

Texture Mapping

Con Texture Mapping ci si riferisce al processo di applicazione di un'immagine sulla superficie di un modello 3D.

Ai vertici (di ogni triangolo 3D) si associano delle coordinate [*u*,*v*] nello spazio texture/immagine



Rasterizzazione con Texture

Una volta che ai vertici di un triangolo 3D sono state associate delle coordinate texture (coordinate [u,v] di una immagine), il problema si riduce a rasterizzare il triangolo pixel per pixel recuperando il colore corretto per ogni pixel dall'immagine texture.

Questo può essere fatto utilizzando le coordinate baricentrice date dall'associazione vertici triangolo da rasterizzare e vertici immagine texture; quindi a partire da un pixel del triangolo che si sta rasterizzando si determina un punto (non un pixel) del piano immagine.

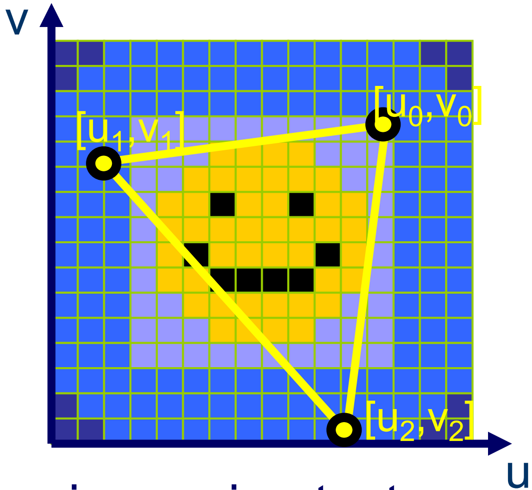
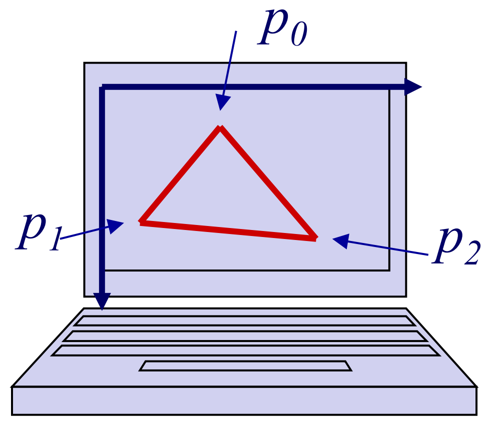


immagine texture

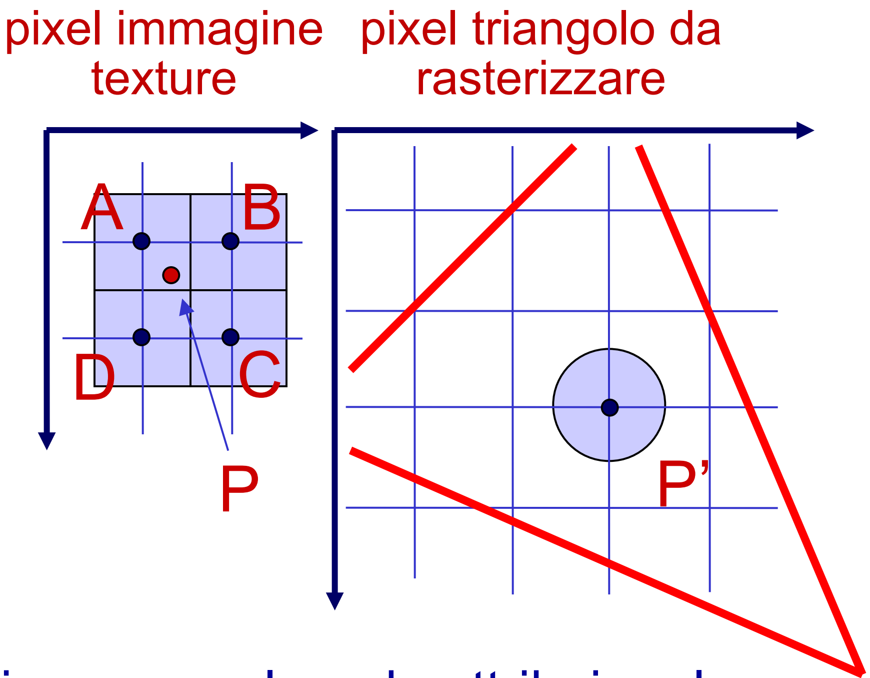


triangolo da rasterizzare

Immagini Texture

1.Si consideri un pixel P’ di un triangolo schermo (proiezione di un triangolo 3D) e gli si applichi la trasformazione baricentrica per portarlo nello spazio dell’ immagine (punto P);

2.si procede alla determinazione dei pixel (A, B, C, D) dell’immagine più vicini a P.



Ci sono più tecniche per determinare un colore da attribuire al pixel del triangolo che si deve rasterizzare a partire dal o dai pixel più prossimi dell’immagine texture; richiamiamone alcuni:

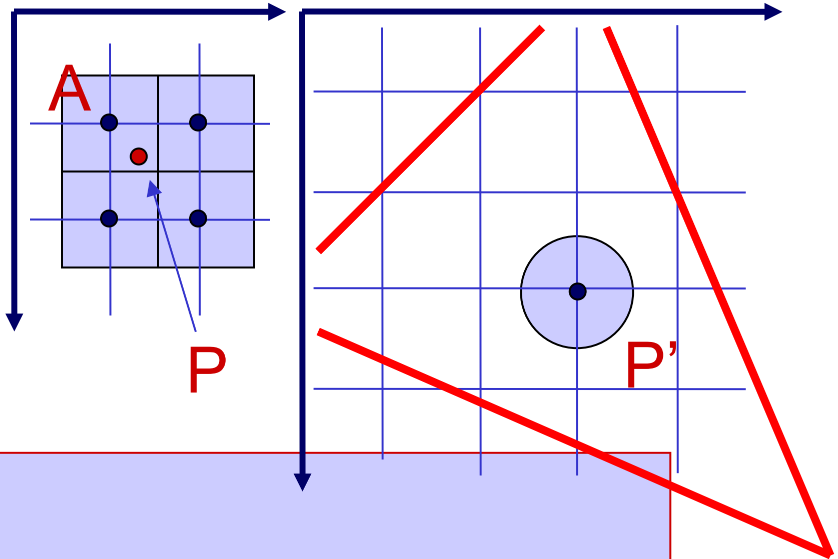
- Nearest Neighbor
- Bilinear Interpolation
- Bicubic Interpolation

Nearest Neighbor

Come dice il nome, consiste nel considerare il pixel dell'immagine texture più vicino a P.

Nel caso di figura, sarà il pixel A; si tratta dello schema più semplice possibile.

Dalle coordinate floating point (px,py) di P, vengono determinate le coordinate intere (ax,ay) di A come:

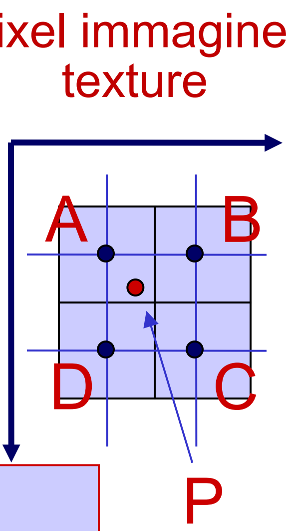


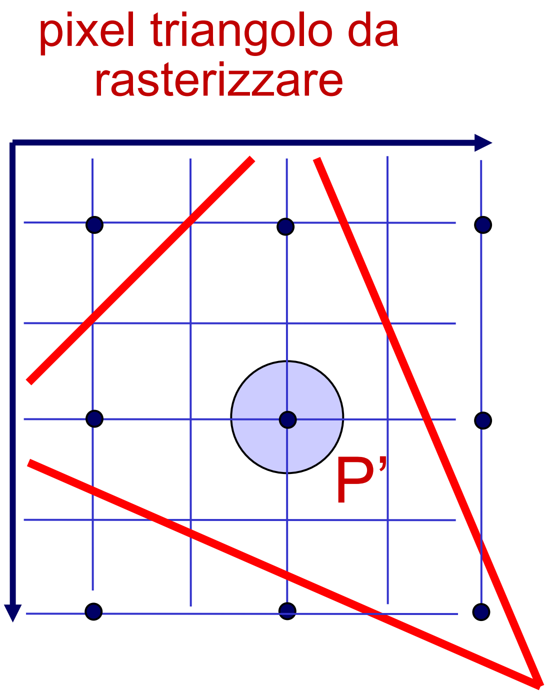

```
float px,py;
int ipx,ipy,ax,ay;
   inv_trasf(ipx, ipy, &px, &py, …);
   ax = (int) px;
   ay = (int) py;
   pixv=getPixelImage(image,ax,ay);
   setPixel(imageData,px,py,pixv[0],pixv[1],pixv[2],255);
```

Bilinear Interpolation

Questa tecnica considera i 4 pixel, dell'immagine texture, più vicini a P;

le loro coordinate intere vengono determinate dalle coordinate floating point (px,py) di P, come:





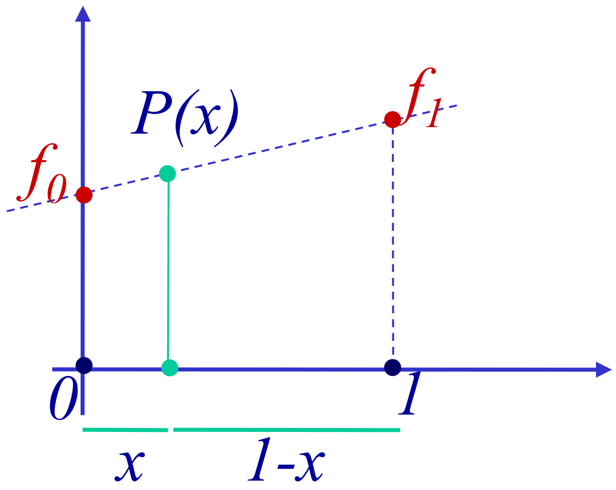

```
float px,py;
int ipx,ipy,ax,ay,bx,by,cx,cy,dx,dy;

   inv_trasf(ipx, ipy, &px, &py, …);
   ax = dx = (int) px;
   ay = by = (int) py;
   bx = cx = ax + 1;
   dy = cy = ay +1;
```

Le componenti colore dei 4 pixel più vicini vengono poi combinate in modo pesato (interpolazione) per arrivare alle componenti colore da attribuire a P'.

Linear Interpolation

Si considera l'interpolazione polinomiale di grado 1 di due valori scalari f_0 ed f_1 assegnati in corrispondenza delle ascisse *0* e *1*.

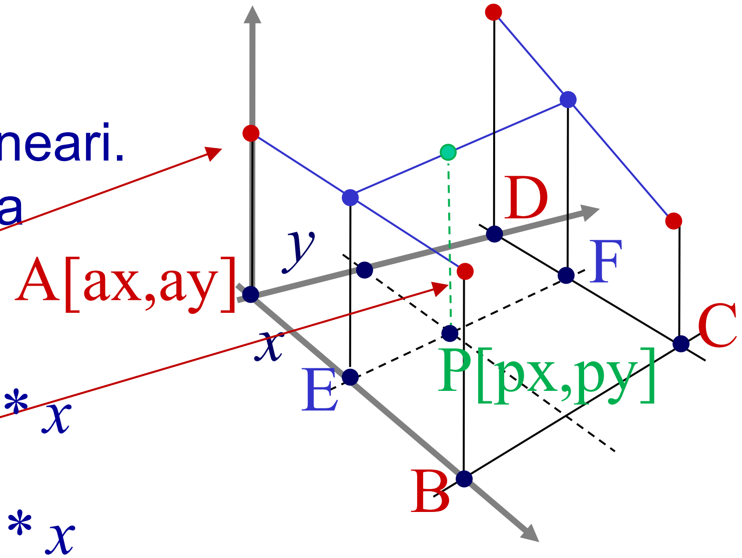


Si consideri la base polinomiale *(1-x)*, *x* per lo spazio dei polinomi lineari che permette di scrivere l'interpolante lineare come:

$$P(x) = f_0 * (1-x) + f_1 * x$$

Bilinear Interpolation

Linterpolazione bilineare si costruisce mediante una sequenza di interpolazioni lineari. Sia $x=px-ax$ e $y=py-ay$, allora



$$f\,E = f\,A * (1-x) + f\,B * x$$

$$f\,F = f\,D * (1-x) + f\,C * x$$

$$f\,P = f\,E * (1-y) + f\,F * y$$

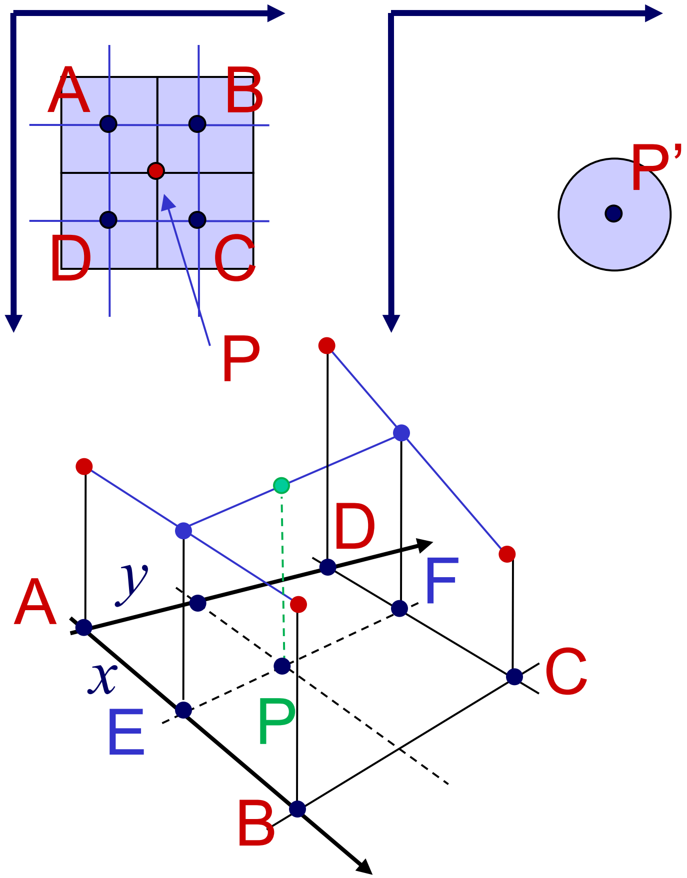
od anche, sostituendo:

$$f\,P = f\,A*(1-x)*(1-y) + f\,B*x*(1-y) + f\,D*(1-x)*y + f\,C*x*y$$

Caso particolare di Bilinear Interpolation

Sia data un'immagine le cui dimensioni siano potenze di 2 e la si voglia scalare di ½.

In questa situazione ogni pixel dell'immagine texture, nella trasformazione inversa si troverà esattamente al centro di quattro pixel originali e il bilinear interpolation si ridurrà ad una media aritmetica dei quattro pixel:



Cioè per $x=y=1/2$:

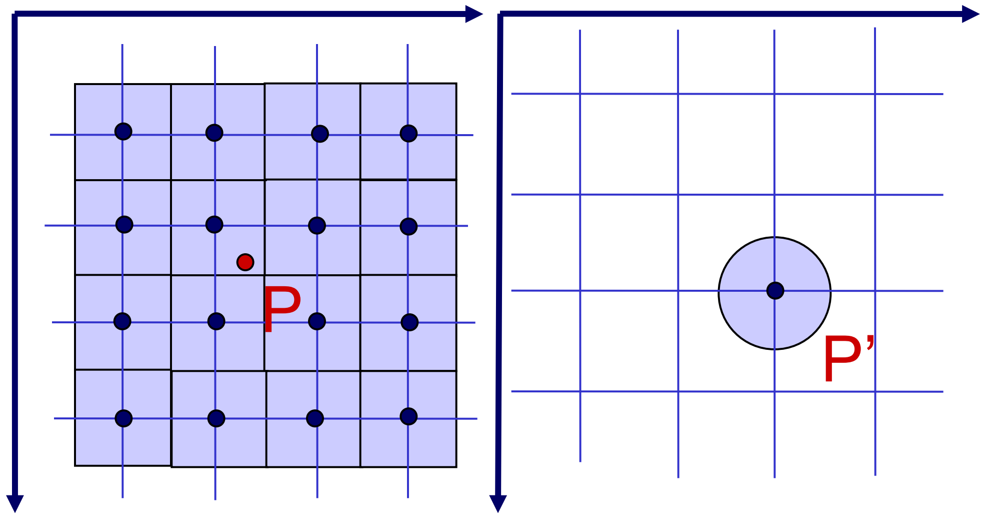
$$f\,P = f\,A*(1-x)*(1-y) + f\,B*x*(1-y) + f\,D*(1-x)*y + f\,C*x*y$$

$$= (f\,A + f\,B + f\,C + f\,D)/4$$ (media alla base del MipMapping)

Bicubic Interpolation

Questa tecnica considera i 4x4 pixel, dell'immagine texture, più vicini a P;

le loro coordinate vengono determinate dalle coordinate floating point (px,py) di P:



Le componenti colore di questi 4x4 pixel vengono combinate in modo pesato (interpolazione) per arrivare alle componenti colore da attribuire a P'.

Questo metodo produce immagini migliori rispetto ai due metodi precedenti ed è forse la combinazione ideale fra tempo di calcolo e qualità. Per questo è il metodo standard in molti programmi di editing di immagini come Adobe Photoshop.

Esempio

vedi: cartella HTML5_2d_2

raster_draw_image.html e .js

Server Web locale

Esistono due modi diversi per visualizzare gli esempi .html e .js
È possibile aprire direttamente il file .html in un browser oppure è possibile installare un Server Web Locale.

Il primo modo funzionerà per la maggior parte degli esempi di base, ma quando iniziamo a caricare risorse esterne come modelli o immagini, aprire il file HTML non funziona.

In questo caso è necessario un Server Web Locale affinché le risorse esterne siano caricate correttamente.

Vedremo un paio di modi diversi in cui è possibile configurare un semplice Server Web Locale.

Server Web locale

La configurazione di un server Web locale è semplice, ma dipende da ciò che è già stato installato sulla macchina.

L'approccio basato su Python dovrebbe funzionare sulla maggior parte dei sistemi Unix / Mac, perché hanno già installato Python. Su tali sistemi che hanno Python dare il seguente comando da shell:

```
>python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

bisogna farlo nella cartella in cui si ha il codice HTML5 e JavaScript; verrà avviato un server Web locale sulla porta 8000.
Ora nel browser (chrome) digitare come indirizzo

```
localhost:8000
```

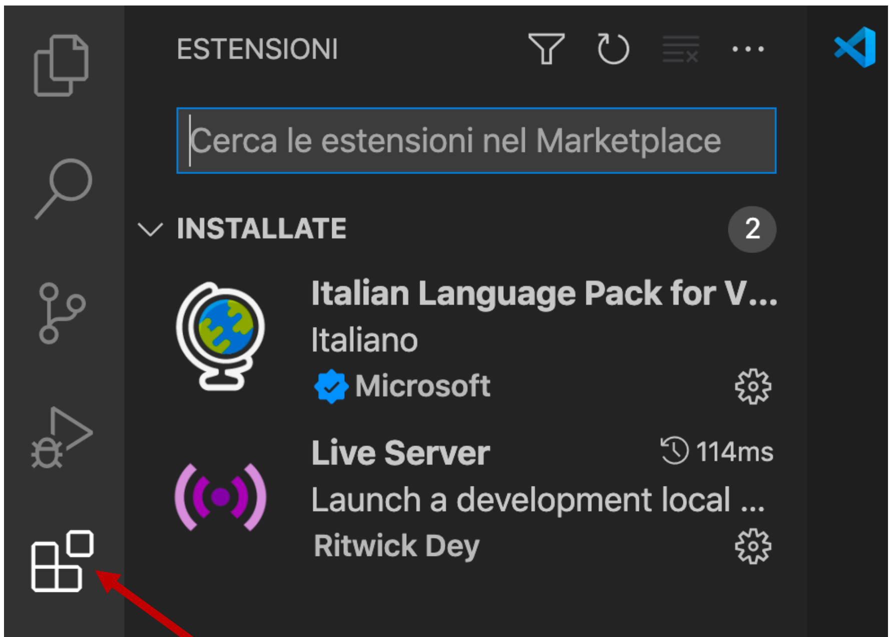
e apparirà la lista dei file nella cartella

Nota: l'equivalente di SimpleHTTPServer in phyton3 è http.server:

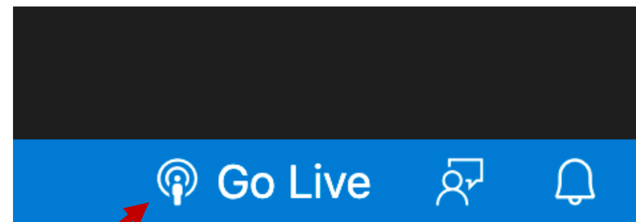
```
>python –m http.server 8000
```

Server Web locale

In alternativa, se stiamo usando come editor Visual Studio Code (vscode), possiamo installare l'estensione Live Server una tantum ed eseguire ogni codice html attivando velocemente un Server Web locale.

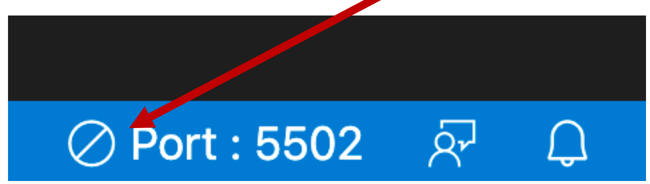


Per cercare una estensione e installarla



Per eseguire il codice visualizzato con un Web Server locale attivo

Per spegnere il Server Web locale



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Giulio Casciola
Dip. di Matematica
giulio.casciola at unibo.it