Sistemi Multimediali

Trasformazione di immagini

Marco Gribaudo
marcog@di.unito.it,
gribaudo@elet.polimi.it

Trasformazione di immagini

Come accennato in precedenza, in un'operazione di composizione un'immagine puo' essere trasformata prima di essere combinata con lo sfondo.

In particolare vengono solitamente applicate alle immagini tutte le tecniche viste per trasformare le primitive presentate in SVG.

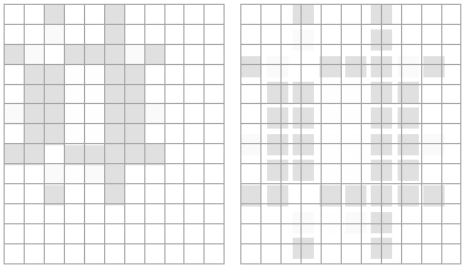
Trasformazione di immagini

Ogni operazione di ingrandimento, riduzione o rotazione (di un angolo non multiplo di 90º) causa una perdita di qualita'.

Il problema e' dovuto al fatto che si perde una corrispondenza 1:1 tra i pixel dell'immagine sorgente, e quelli della destinazione.

Trasformazione di immagini

Non essendoci corrispondenza diretta, i colori dei pixel devono essere "calcolati" con un algoritmo, causando una inevitabile perdita di qualita'.



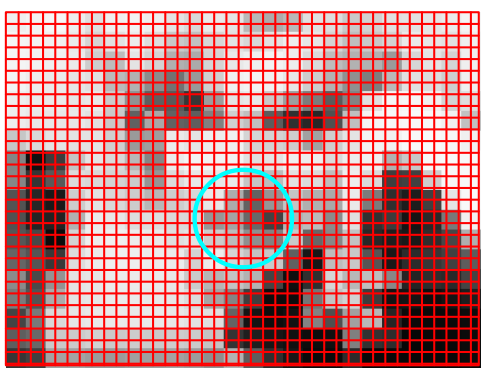
Trasformazione di immagini

A seconda del tipo di trasformazione, si verificano problemi distinti, e si adottano soluzioni diverse per combatterli. In particolare vedremo come:

Ingrandire
Rimpicciolire
Ruotare

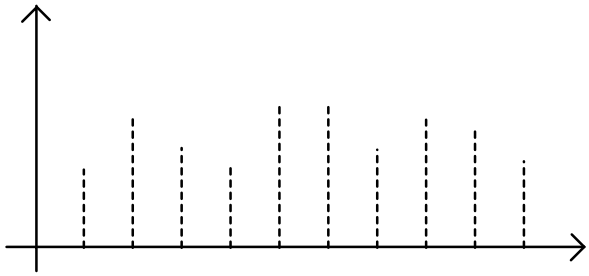
Ingrandimento

Nel caso dell'ingrandimento, si avra' un numero di pixel *superiore* a quello dell'immagine originale.



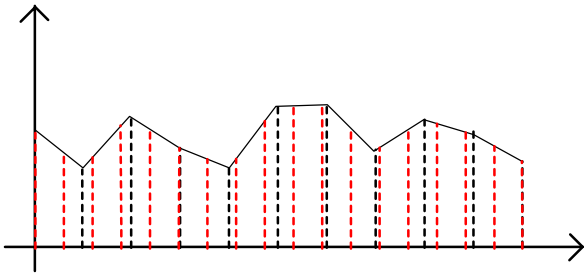
Ingrandimento

In questo caso si interpreta l'immagine come un segnale.



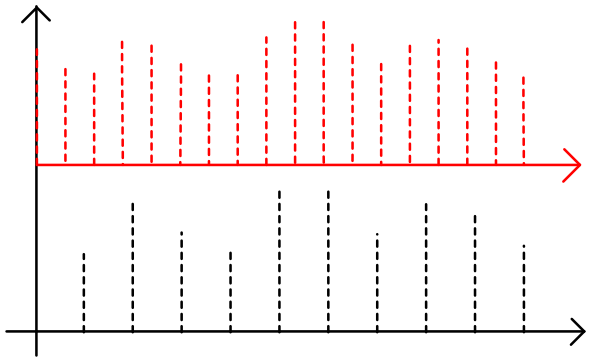
Ingrandimento

L'algoritmo prevede di ricostruire il segnale originale, e di campionarlo nuovamente ad una frequenza piu' alta.



Ingrandimento

Per questo si parla di *ricampionamento* (o *resampling*)



Ingrandimento

Vi sono diversi modi con cui ricostruire il segnale originale per ricampionarlo. I piu' diffusi sono:

Pixel piu' vicino

Bilineare

Bicubico

Ingrandimento

Per ogni pixel di coordinate *(x',y')* dell'immagine ingrandita, tutti gli algoritmi calcolano la corrispondente posizione *(x,y)* nell'immagine originale. Se l'ingrandimento e' di un fattore β, allora semplicemente si avra':

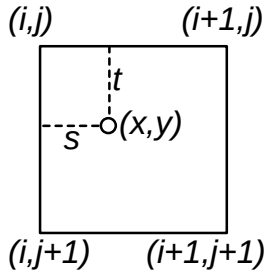
$$x = x' / \beta$$

$$y = y' / \beta$$

Ingrandimento

Inoltre, per determinare il colore del pixel, gli algoritmi separano la parte intera dalla parte decimale delle coordinate *(x,y)* :

$$i = \lfloor x \rfloor \qquad s = x - i$$
$$j = \lfloor y \rfloor \qquad t = y - j$$



Ingrandimento

Indicheremo con $p_{i,j}$ il pixel alle coordinate *(i,j)* dell'immagine originale, e con *p'* il pixel che si sta calcolando per l'immagine ingrandita.

Il calcolo del colore deve essere ripetuto per tutti i canali dell'immagine (es. rosso, verde, blu).

Il calcolo e' pero' indipendente dal canale, per cui verra' riportato una volta sola.

Ingrandimento

Il meccanismo *Pixel piu' vicino*, non esegue alcuna interpolazione del segnale ricostruito, ed assegna semplicemente il colore del pixel piu' prossimo alla coordinata *(x,y)* nell'immagine originale.

$$p' = \begin{cases} & s < 0.5 & s \geq 0.5 \\ t < 0.5 & p_{i,j} & p_{i+1,j} \\ t \geq 0.5 & p_{i,j+1} & p_{i+1,j+1} \end{cases}$$

Ingrandimento

E' il metodo piu' semplice, ma e' anche quello che fornisce i risultati meno soddisfacienti, in quanto le immagini create sono "scalettate".

Ingrandimento

Nel meccanismo *blineare*, viene eseguita una doppia interpolazione lineare tra i 4 (2x2) pixel attorno alla posizione in cui si trova il pixel ricostruito.

$$\begin{aligned} p' &= (1-s)\,(1-t)\,p_{i,j} + s\,(1-t)\,p_{i+1,j} + \\ &\quad +(1-s)\,t\,p_{i,j+1} + s\,t\,p_{i+1,j+1} \end{aligned}$$

Ingrandimento

La tecnica bilineare ha un ottimo rapporto tra qualita' dell'immagine prodotta ed efficienza.

Ingrandimento

La tecnica bicubica, utilizza invece un'iterpolazione con un polinomio di terzo grado del segnale. Essa miscela i 16 (4x4) colori dei pixel vicini alla posizione del punto da considerare.

$$p' = \sum_{k=-1}^{2} \sum_{l=-1}^{2} p_{i+k,j+l} \phi_k(s) \phi_l(t)$$

$$\phi_{-1}(h) = \frac{-h^3 + 3h^2 - 2h}{6} \quad \phi_1(h) = \frac{-h^3 + h^2 + 2h}{2}$$

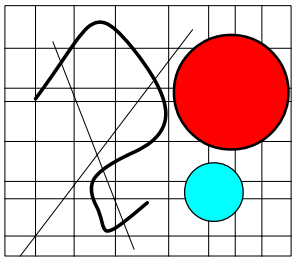
$$\phi_0(h) = \frac{h^3 - 2h^2 - h + 2}{2} \quad \phi_2(h) = \frac{h^3 - h}{6}$$

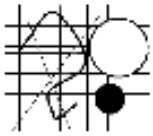
Ingrandimento

La tecnica bicubica e' quella capace di fornire i risultati migliori, al prezzo di un piu' lungo tempo di elaborazione.

Riduzione

Anche nel caso della riduzione di un'immagine, la tecnica del pixel piu' vicino e' applicabile, ed e' in grado di fornire risultati di modesta qualita', ma in modo piuttosto efficiente.





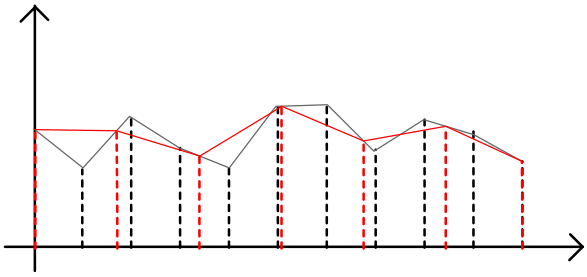
Riduzione

Per aumentare la qualita' della riduzione, non e' possibile utilizzare tecniche di ricampionamento.

Nella riduzione infatti il problema e' dovuto alla perdita di dettagli che possono conseguire all'utilizzo di un piu' ristretto numero di pixel.

Riduzione

I dettagli corrispondono infatti ad alte frequenze. Riducendo la frequenza di campionamento, per il teorema di Nyquist, si potranno considerare solamente un numero inferiore di dettagli.



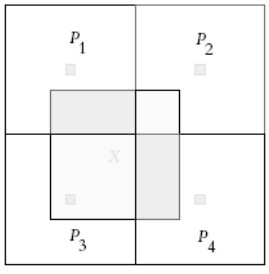
Riduzione

Per ottenere una migliore qualita', si cerca di riprodurre il processo fisico di campionamento dei punti di un'immagine.

La lente posta davanti ad ogni sensore tende infatti ad integrare i colori che ricadono al suo interno, producendo una tonalita' media.

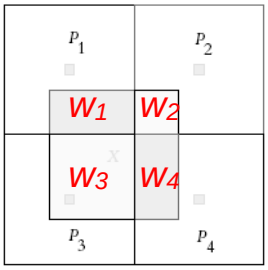
Riduzione

Si cerca quindi di mediare i colori di tutti i pixel dell'immagine sorgente che cadono all'interno di un pixel dell'immagine ridotta...



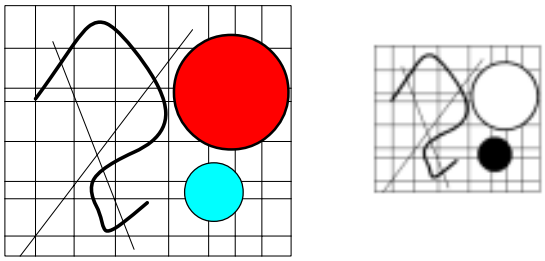
Riduzione

Pesando ogni colore per un valore corrispondente all'intersezione delle aree.



Riduzione

In questo modo dettagli molto pronunciati, pur riducendosi a colori di intensita' inferiore, rimangono comunque visibili.



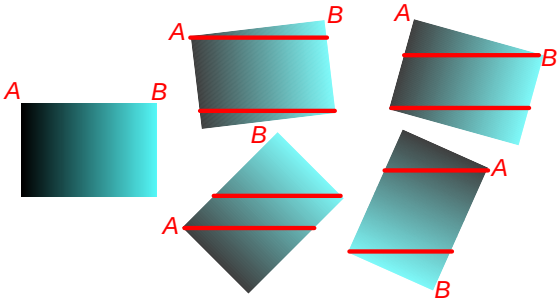
Altre trasformazioni

Rotazioni ed altre trasformazioni (quali ad esempio lo *shear*), sono invece molto piu' complesse da realizzare.

Le difficolta' nascono principalmente dal gran numero di configurazioni che si possono verificare.

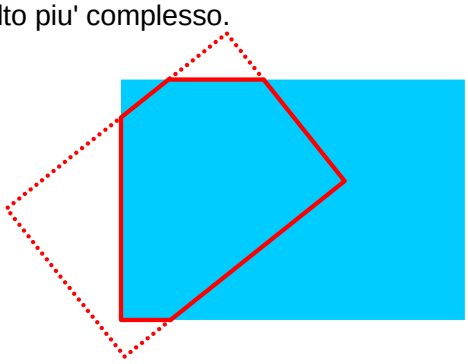
Altre trasformazioni

Considerando ad esempio una rotazione, vi sono tantissimi modi in cui i vertici possono disporsi nello spazio.



Altre trasformazioni

Inoltre, alcune parti dell'immagine possono uscire dal quadro, trasformando un quadrilatero in un poligono molto piu' complesso.



Altre trasformazioni

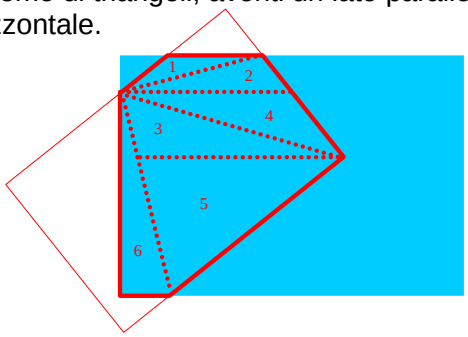
Per calcolare correttamente le coordinate, viene fatto un uso abbastanza massiccio dell'*interpolazione lineare*. Se una funzione e' tale per cui $f(x_0)=y_0$ ed $f(x_1)=y_1$, allora si approssima $f(x)$ per $x_0 < x < y_0$ con:

$$f(x) = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0}$$

Chiameremo in questo caso l'operazione di interpolazione $I(x|...)$, vale a dire $f(x) = I(x|x_0,y_0,x_1,y_1)$

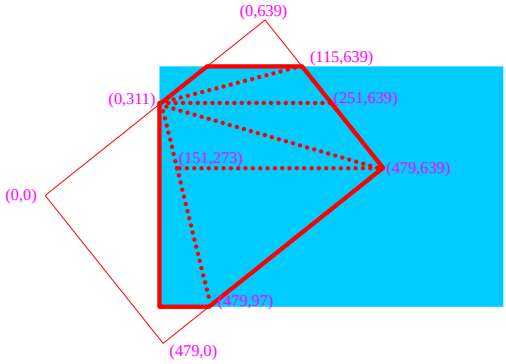
Altre trasformazioni

Prendendo spunto dalle tecniche utilizzate in grafica 3D, ogni figura trasformata viene disegnata come un insieme di triangoli, aventi un lato parallelo all'asse orizzontale.



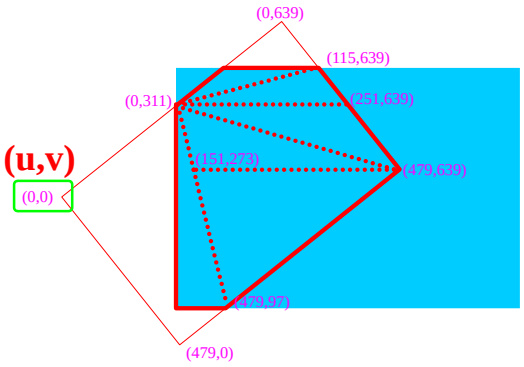
Altre trasformazioni

Ogni vertice di un triangolo ha associato una coordinata dell'immagine originale.



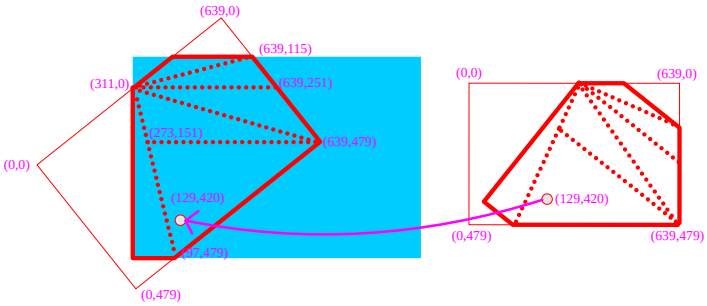
Altre trasformazioni

Per non creare confusione, le coordinate dell'immagine verranno indicate con le lettere *u* e *v*.



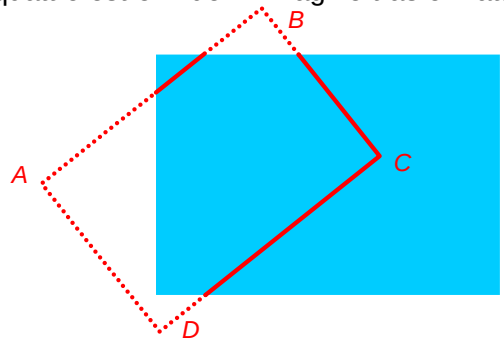
Altre trasformazioni

Durante il disegno dei pixel del triangolo, vengono utilizzate le coordinate *u v* per selezionare i punti dell'immagine originale da cui copiare il colore.



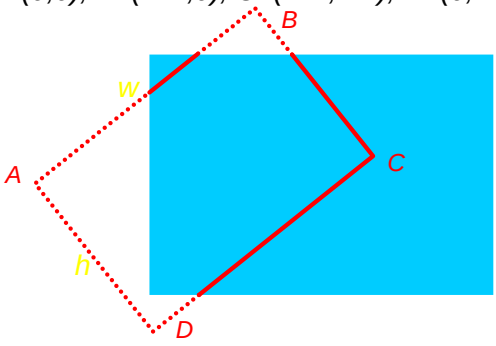
Altre trasformazioni

La prima fase determina quindi i punti (nello spazio dell'immagine originale) in cui andranno a finire i quattro estremi dell'immagine trasformata.



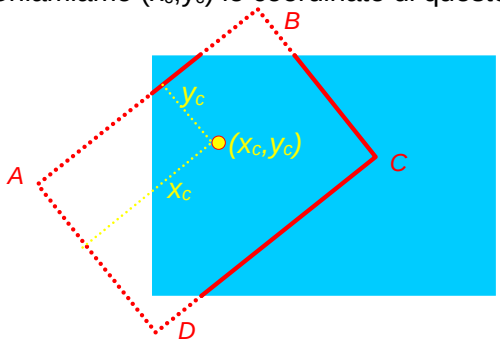
Altre trasformazioni

Supponiamo che l'immagine abbia dimensioni *w, h*.
Le coordinate di partenza dei quattro vertici saranno:
A=(0,0); B=(w-1,0); C=(w-1,h-1); D=(0,h-1).



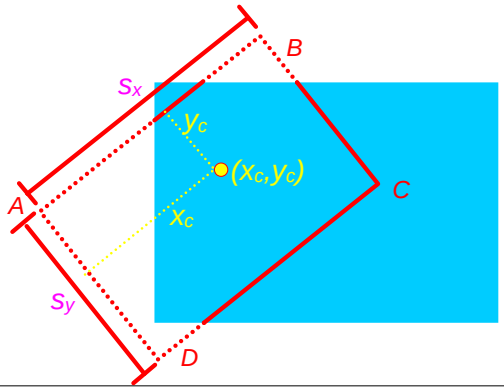
Altre trasformazioni

Ogni immagine ha un centro, rispetto al quale si eseguono operazioni di rotazione e di scala. Chiamiamo (x_c,y_c) le coordinate di questo centro



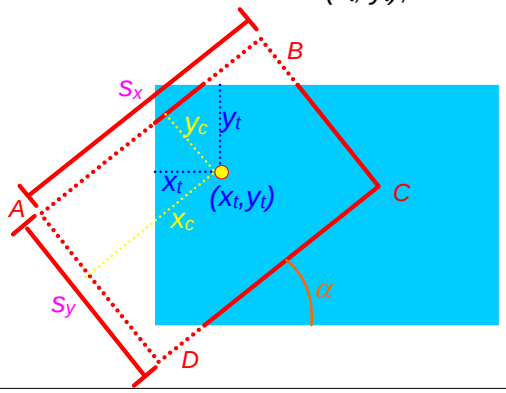
Altre trasformazioni

Se variamo la scala orizzontale e verticale rispettivamente di un fattore s_x e s_y, ...



Altre trasformazioni

... ruotiamo la figura di un agnolo α, e posizioniamo il suo centro alle coordinate (x_t, y_t),...



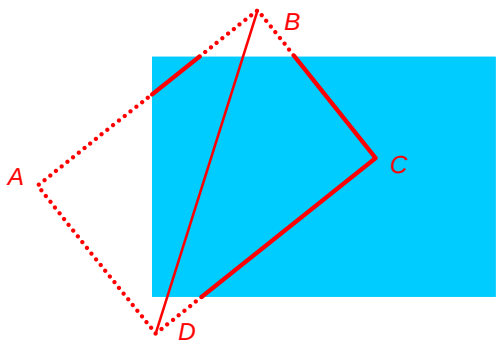
Altre trasformazioni

Possiamo determinare le coordinate finali (x,y) di un punto $P=(x_p,y_p)$, con le seguenti formule:

$$
\begin{aligned}
x &= x_t + s_x(x_p - x_c)\cos\alpha - s_y(y_p - y_c)\sin\alpha \\
y &= y_t + s_x(x_p - x_c)\sin\alpha + s_y(y_p - y_c)\cos\alpha
\end{aligned}
$$

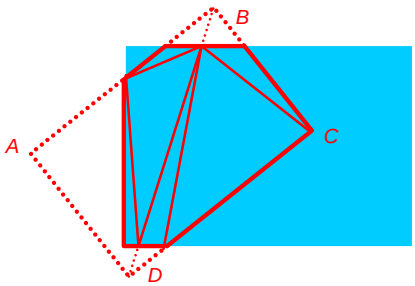
Altre trasformazioni

Il rettangolo viene quindi suddiviso in due triangoli adiacenti: *ABC* e *BCD*.



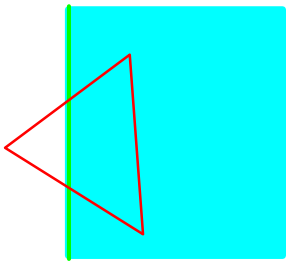
Clipping

La fase successiva spezza quindi i due triangoli originali in un insieme (eventualmente maggiore) di triangoli interamente contenuti all'interno dell'immagine di destinazione.



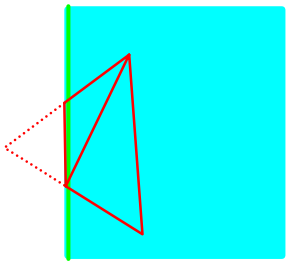
Clipping

Questo processo prende il nome di *Clipping*, e la sua versione piu' semplice considera ad uno ad uno tutti i trinagoli, confrontandoli con i quattro estremi dell'immagine.



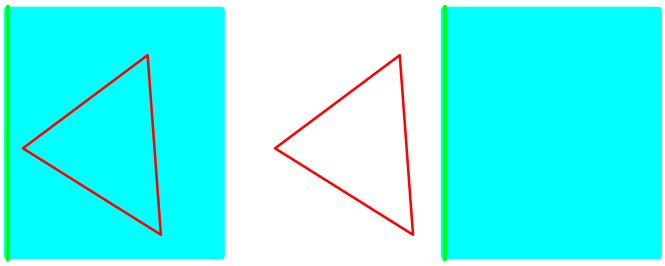
Clipping

Ogni confronto si occupa di suddividere il triangolo in modo che sia interamente contenuto in un semipiano. Dato quindi un triangolo ed un semipiano, vi sono 4 possibbilita'.



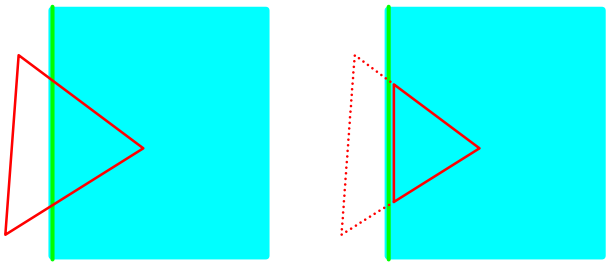
Clipping

Il triangolo e' interamente contenuto (o non contenuto) dentro il semipiano. In questo caso il triangolo e' accettato cosi' com'e' (o rimosso poiche' non visibile).



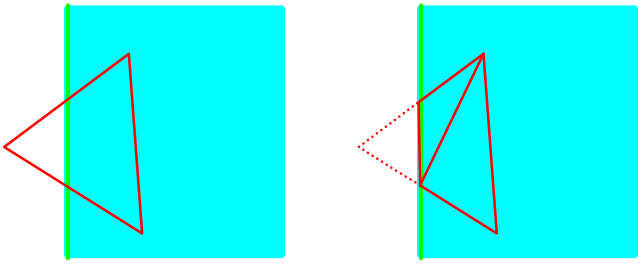
Clipping

Se un vertice cade nel semipiano, e due fuori, allora il triangolo viene ridotto, intersecando i due spigoli col confine del semipiano.



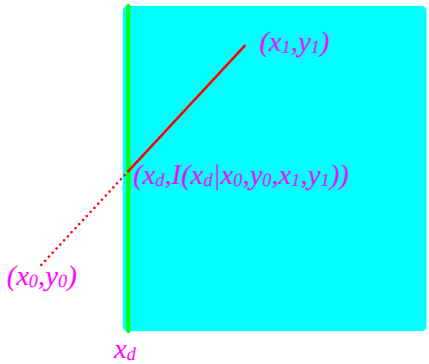
Clipping

Se due vertici cadono dentro il semipiano, ed uno fuori, il triangolo viene spezzato in due triangoli, avendo come vertici le intersezioni col semipiano.



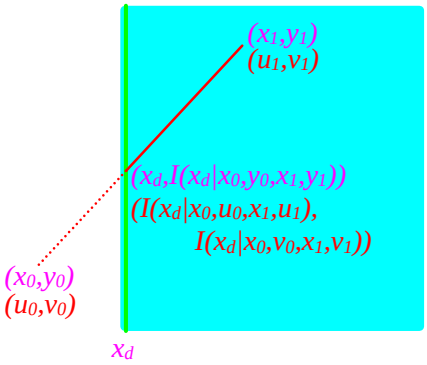
Clipping

In entrambe i casi, le coordinate dei punti di intersezione si ottengono per interpolazione:



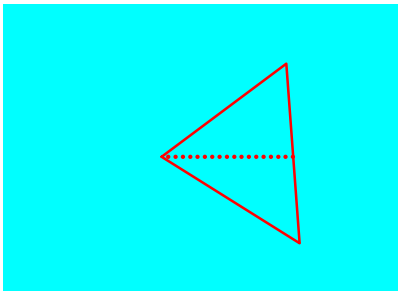
Clipping

Ai nuovi vertici vengono associate nuove coordinate *u v*, ottenute anche esse per interpolazione.



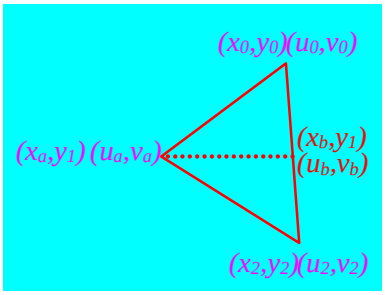
Rasterizzazione

Il procedimento con cui viene disegnato un triangolo prende il nome di *rasterizzazione*. Esso inizia suddividendo il triangolo in due triangoli con uno spigolo parallelo all'asse delle x.



Rasterizzazione

Coordinate *x*, *u* e *v* del vertice che divide il triangolo in due vengono calcolate per interpolazione.



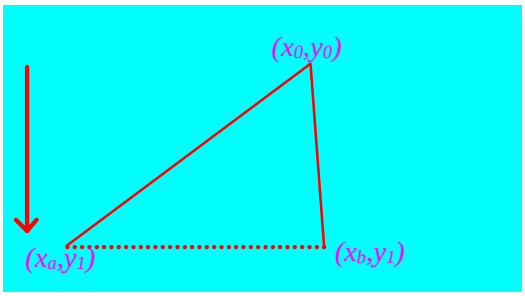
$$x_b = I(y_1|y_0,x_0,y_2,x_2)$$

$$u_b = I(y_1|y_0,u_0,y_2,u_2)$$

$$v_b = I(y_1|y_0,v_0,y_2,v_2)$$

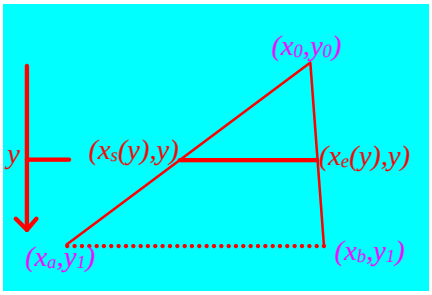
Rasterizzazione

Partendo quindi dal vertice non appartenente allo spigolo parallelo all'asse x, cicla su tutte le righe di pixel che corrispondono al triangolo. Chiamiamo y_0 ed y_1 le righe iniziali e final del triangolo.



Rasterizzazione

Per ogni riga *y*, interpolando le coordinate degli estremi, determina la coordinata $x_s(y)$ del pixel iniziale e quella $x_e(y)$ del pixel finale.

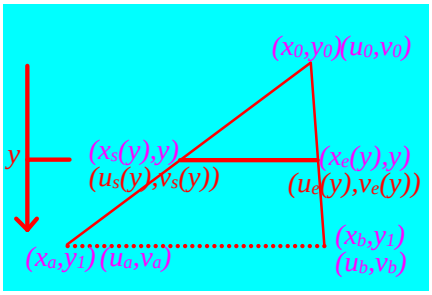


$$x_s(y) = I(y|y_0,x_0,y_1,x_a)$$

$$x_e(y) = I(y|y_0,x_0,y_1,x_b)$$

Rasterizzazione

Inoltre, sempre per interpolazione, determina le coordinate $uv_s(y)$ e $uv_e(y)$ associate ai punti $x_s(y)$ ed $x_e(y)$.



$$u_s(y) = I(y|y_0,u_0,y_1,u_a)$$

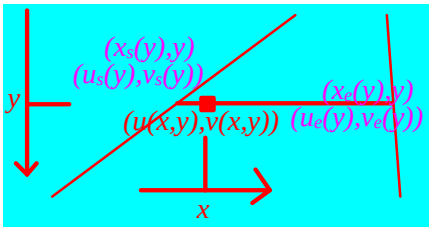
$$u_e(y) = I(y|y_0,u_0,y_1,u_b)$$

$$v_s(y) = I(y|y_0,v_0,y_1,v_a)$$

$$v_e(y) = I(y|y_0,v_0,y_1,v_b)$$

Rasterizzazione

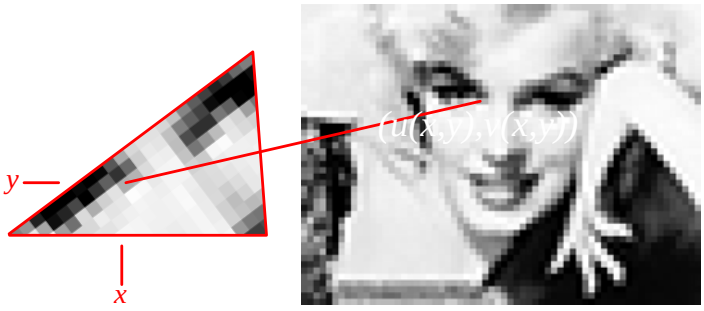
L'algoritmo quindi disegna tutti i pixel da $x_s(y)$ a $x_e(y)$. Per questi punti calcola le coordinate $uv(x,y)$ dell'immagine originale, sempre per interpolazione.



$$u(x,y) = I(x|x_s(y),u_s(y),x_e(y),u_e(y))$$
$$v(x,y) = I(x|x_s(y),v_s(y),x_e(y),v_e(y))$$

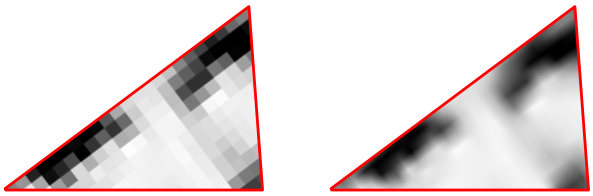
Rasterizzazione

Il colore dell'immagine trasformato viene quindi letto dal punto alle coordinate *uv* dell'immagine originale.



Rasterizzazione

Anche in questo caso si utilizza l'interpolazione bilineare per ridurre la scalettatura della texture.



Mip-Maps

Una trasformazione generica puo' sia ridurre che ingrandire l'immagine originale.

In questo caso risulta pero' troppo complesso applicare la media dei colori come visto per la riduzione semplice.

Si adopera allora una semplificazione chiamata *Mip-Mapping*.

Mip-Maps

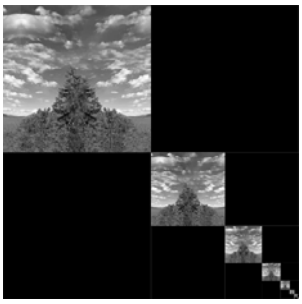
Una **MIPmap** (*multum in parvo*) e' una sequenza di immagini rappresentanti la stessa figura, ognuna di dimensione dimezzata rispetto alla precedente.

Mip-Maps

Ogni copia dimezzata viene effettuata mediando il colore di quattro pixel adiacenti.

Mip-Maps

Una MIPmap occupa il 33% in piu' rispetto all'immagine semplice (per dimostrarlo basta immaginare di separare i canali dei colori rosso, verde e blu).



Mip-Maps

Le copie rimpicciolite, vengono create automaticamente nella fase di caricamento dell'immagine originale.

Durante la copia della sorgente sullo schermo, i pixel vengono presi dall'immagine con le dimensioni piu' vicine a quella dell'immagine di destinazione.