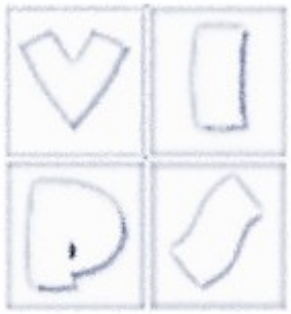


Grafica al calcolatore
-
Computer Graphics

8 – Tecniche di Mapping

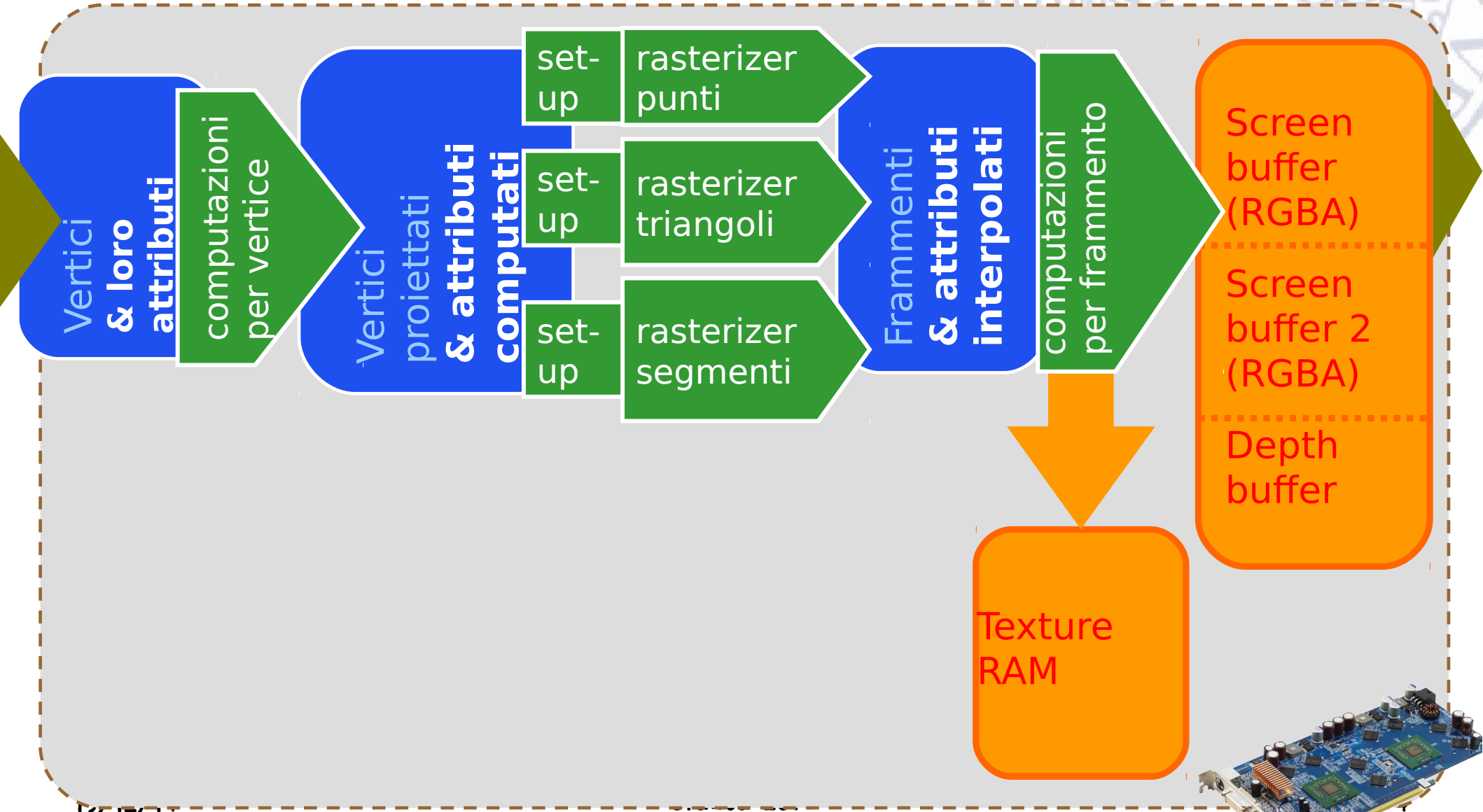
Introduzione

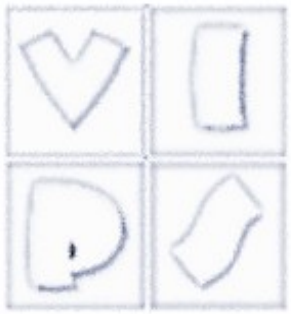
- Il modello di illuminazione di Phong è abbastanza versatile: con una scelta opportuna dei vari parametri si possono imitare diversi materiali in modo abbastanza realistico (per esempio la plastica)
- E' comunque limitato: non si possono simulare i dettagli di un materiale, a meno di non introdurli nella geometria (ovvero aumentare il numero dei poligoni).
- Invece di incrementare la complessità del modello, si aggiungono particolari come parte del processo di rendering.
- Le tecniche di mappatura (texture mapping) interagiscono con lo shading usando una mappa bidimensionale o texture (tessitura) come tabella di lookup, in modo da aggiungere dettagli alla superficie.
- Si possono ottenere risultati ottimi dal punto di vista del fotorealismo con un limitato carico computazionale.
- Il texture mapping è ampiamente supportato dalle schede grafiche.

Texture mapping

- Il texture mappping, nella sua forma più semplice (color mapping) consiste nell'applicare una immagine su una superficie, come una decalcomania.
- Esempi: una etichetta su una lattina, una foto su un cartellone pubblicitario, oppure tessiture regolari come legno o marmo su una superficie.
- Una texture map una matrice bidimensionale di dati indirizzati da due coordinate s e t comprese tra 0 ed 1. Il dato contenuto è tipicamente un colore (la mappa è una immagine), ma potrebbe essere qualcos'altro.
- Più in generale una texture map può contenere qualunque tipo di informazione che incide sull'apparenza di una superficie: la texture map è una tabella ed il texture mapping consiste nel recuperare dalla tabella (lookup) l'informazione che serve per effettuare il rendering di un certo punto.

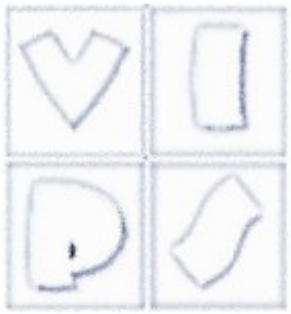
Memoria RAM nelle schede grafiche





Texture Mapping

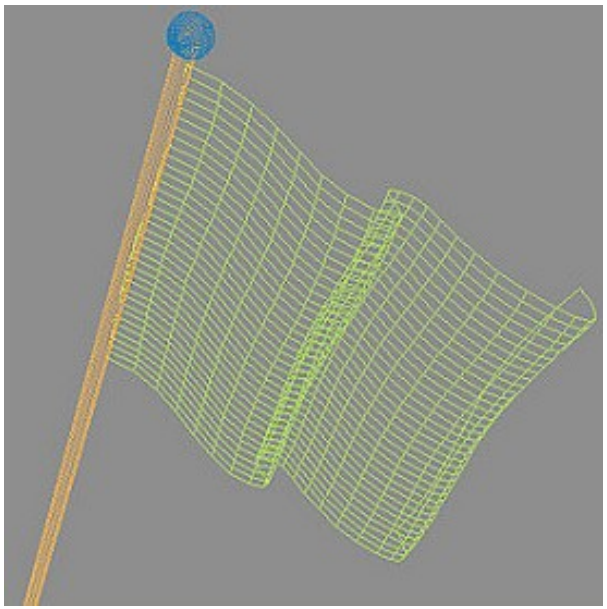
- Nelle operazioni per frammento si può accedere ad una RAM apposita: la Texture RAM strutturata in un insieme di Textures (“tessiture”)
- Ogni tessitura è un array 1D, 2D o 3D di Texels (campioni di tessitura) dello stesso tipo



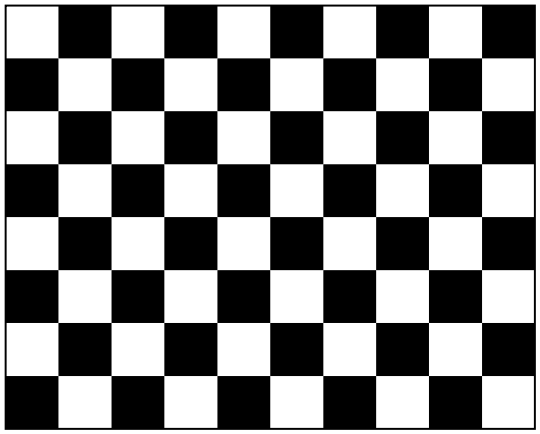
Texels

- Sono esempi di texels:
  - Ogni texel un colore (componenti: R-G-B, o R-G-B-A): la tessitura è una “color-map”
  - Ogni texel una componente alpha: la tessitura è una “alpha-map”
  - Ogni texel una normale (componenti: X-Y-Z): la tessitura è una “normal-map” o “bump-map”
  - Ogni texel contiene un valore di specularità: la tessitura è una “shininess-map”

Rimappare immagini sulla geometria



+

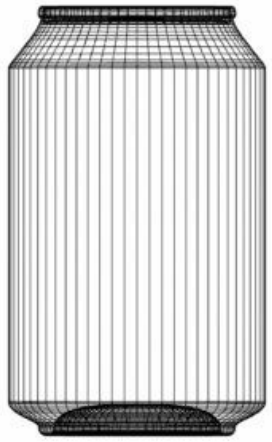


=

geometria 3D
(mesh di quadrilateri)

RGB texture 2D
(color-map)

Rimappare immagini sulla geometria



Rimappare immagini sulla geometria

+

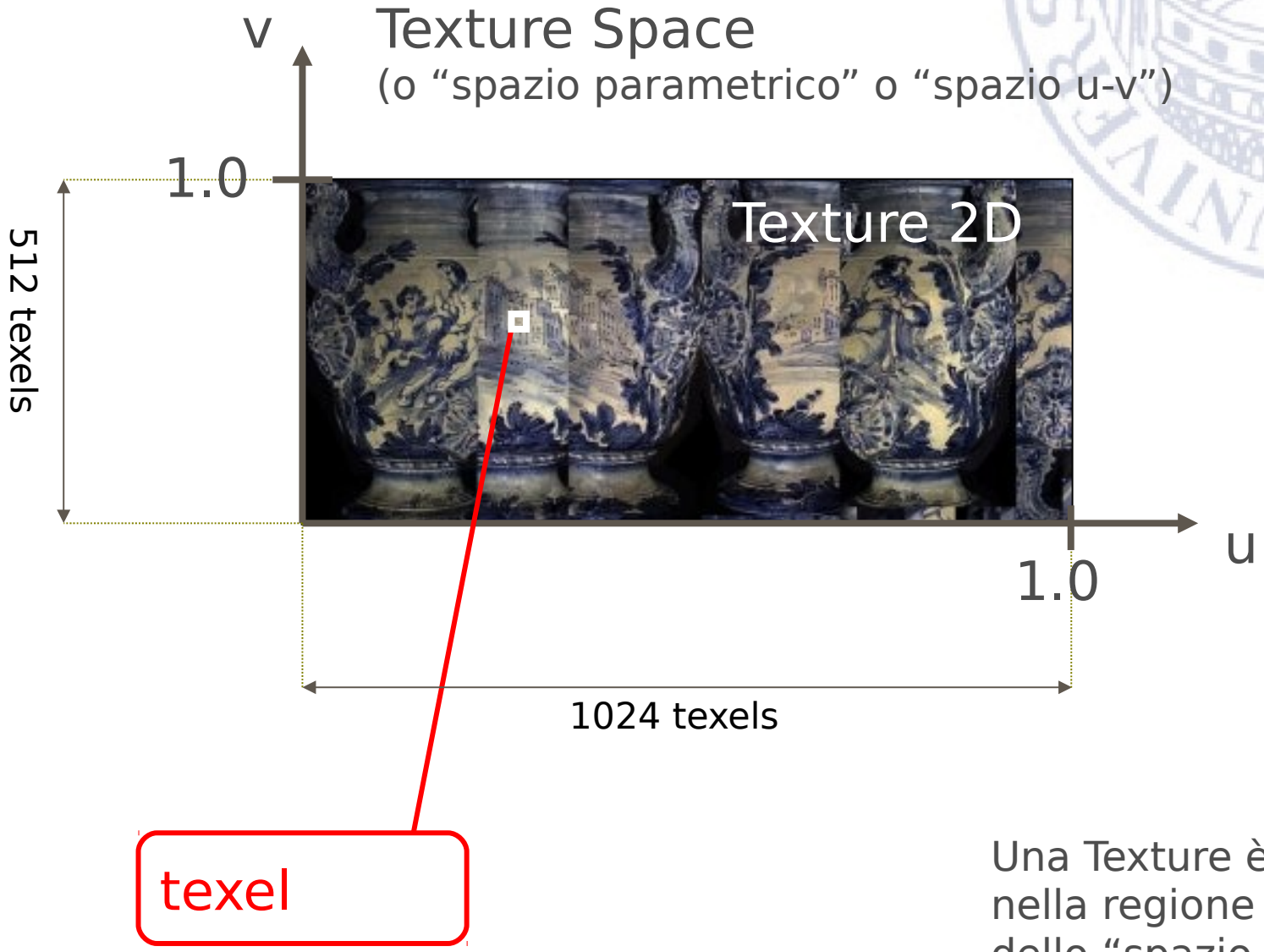
=

Texture Mapping: storia

- 1974 introdotto da Ed Catmull
  - nella sua Phd Thesis
- Solo nel 1992 (!) si ha texture mapping in hardware
  - Silicon Graphics RealityEngine
- Dal '92 a oggi ha avuto aumento rapidissimo della diffusione
  - strada intrapresa soprattutto da low end graphic boards
- Oggi è una delle più fondamentali tecniche di rendering
  - Il re indiscusso delle tecniche image based

Ed Catmull

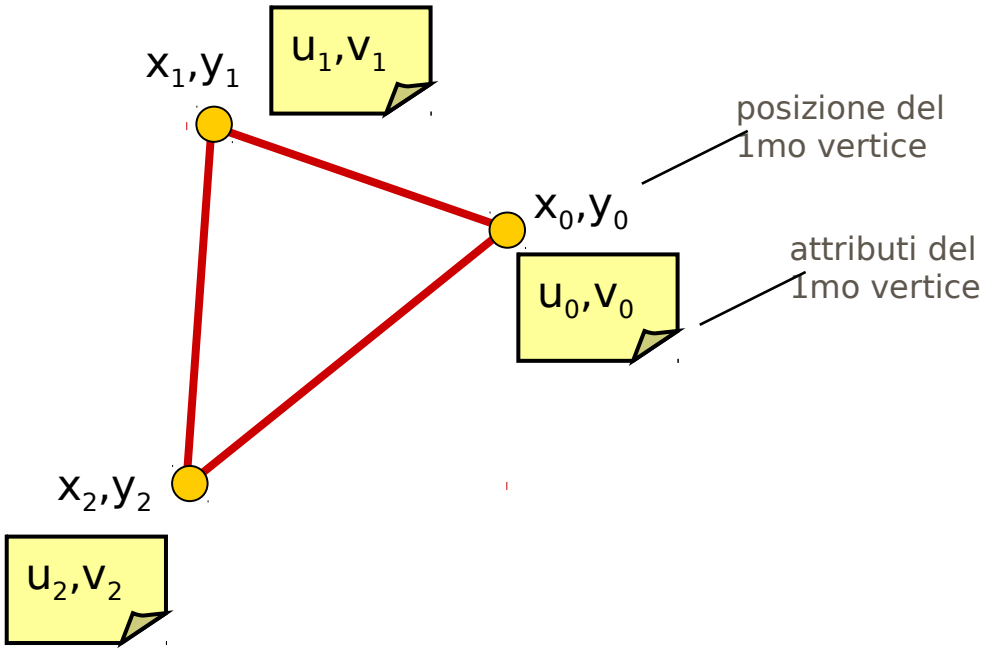
Notazione



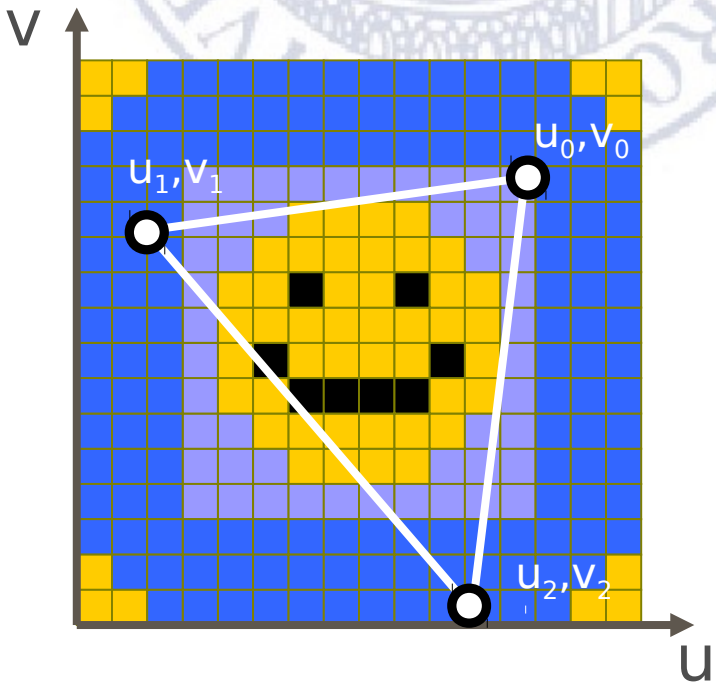
Una Texture è definita nella regione [0,1] x [0,1] dello “spazio parametrico”

Texture Mapping

- Ad ogni vertice (di ogni triangolo) assegno le sue coordinate u,v nello spazio tessitura



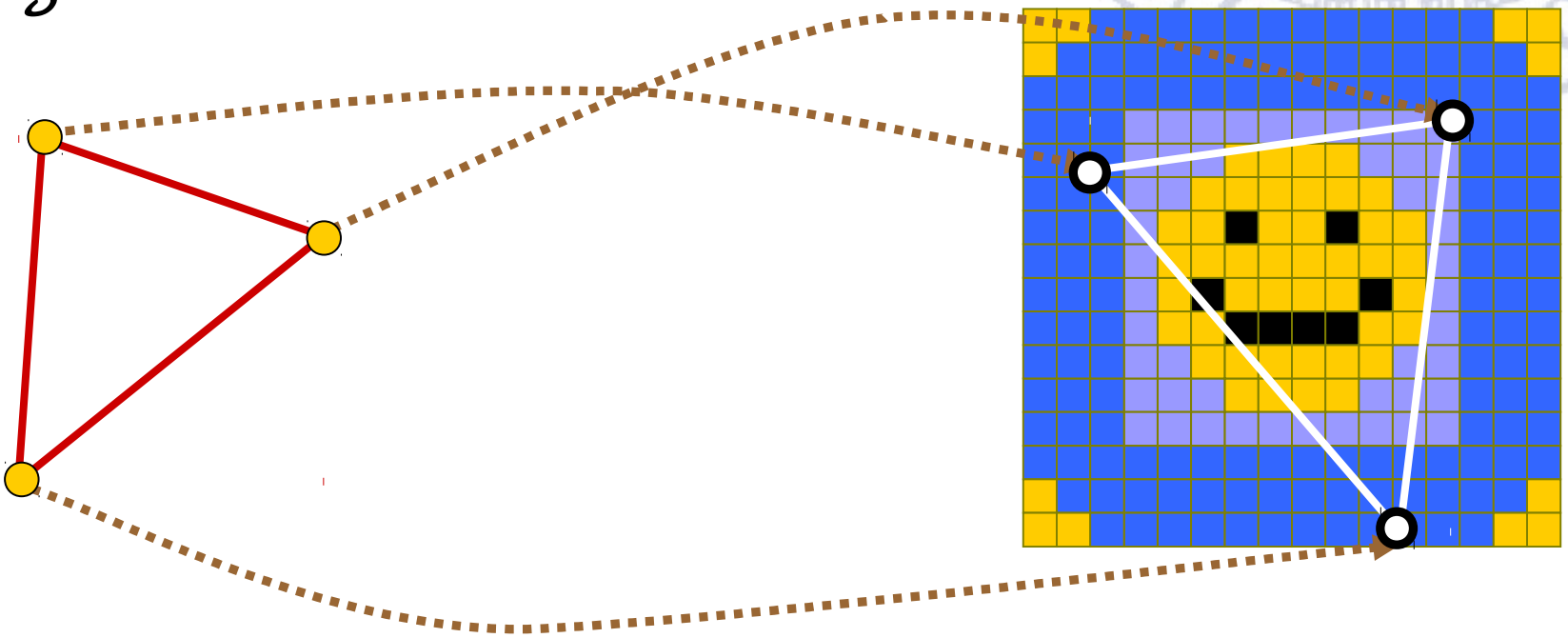
Screen Space



Texture Space

Texture Mapping

- Così in pratica definisco un mapping fra il triangolo e un triangolo di tessitura

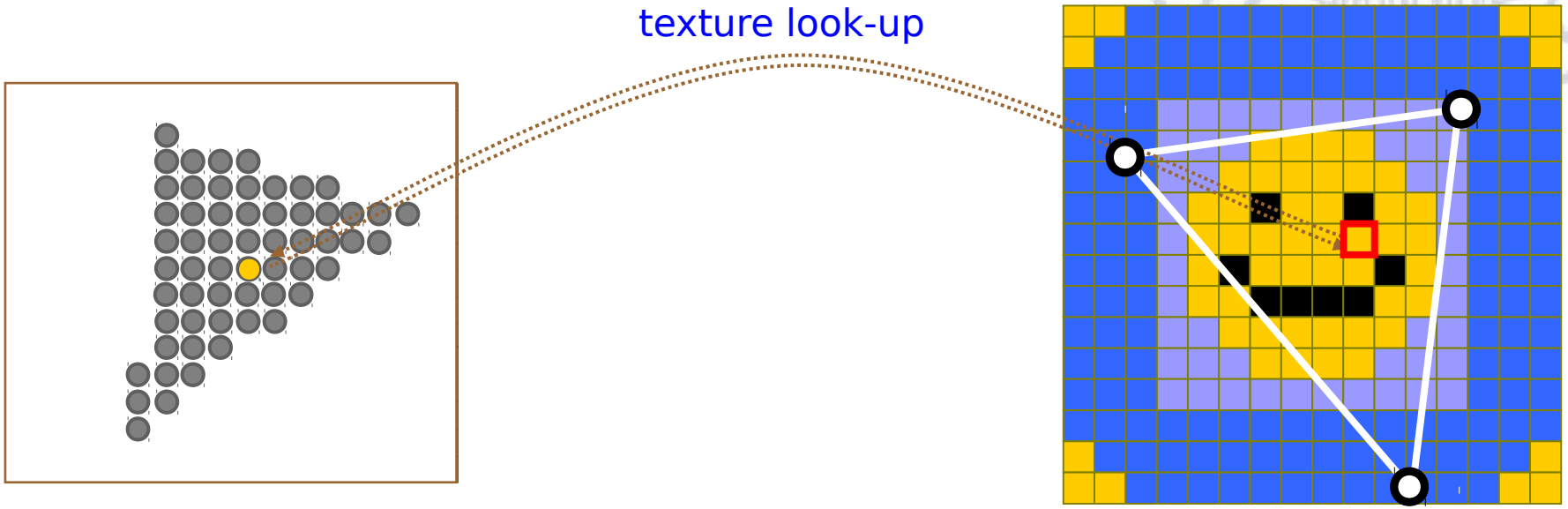


Screen Space

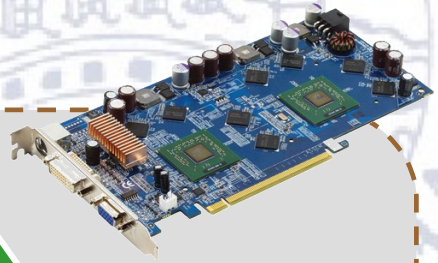
Texture Space

Texture Mapping

- Ogni vertice (di ogni triangolo) ha le sue coordinate u,v nello spazio tessitura



Texture Mapping



Vertici **& loro attributi**

computazioni per vertice

Vertici poriettati **& attributi computati**

set-up

set-up

set-up

rasterizer punti

rasterizer triangoli

rasterizer segmenti

Frammenti **& attributi interpolati**

computazioni per frammento

Screen buffer

texture look-up

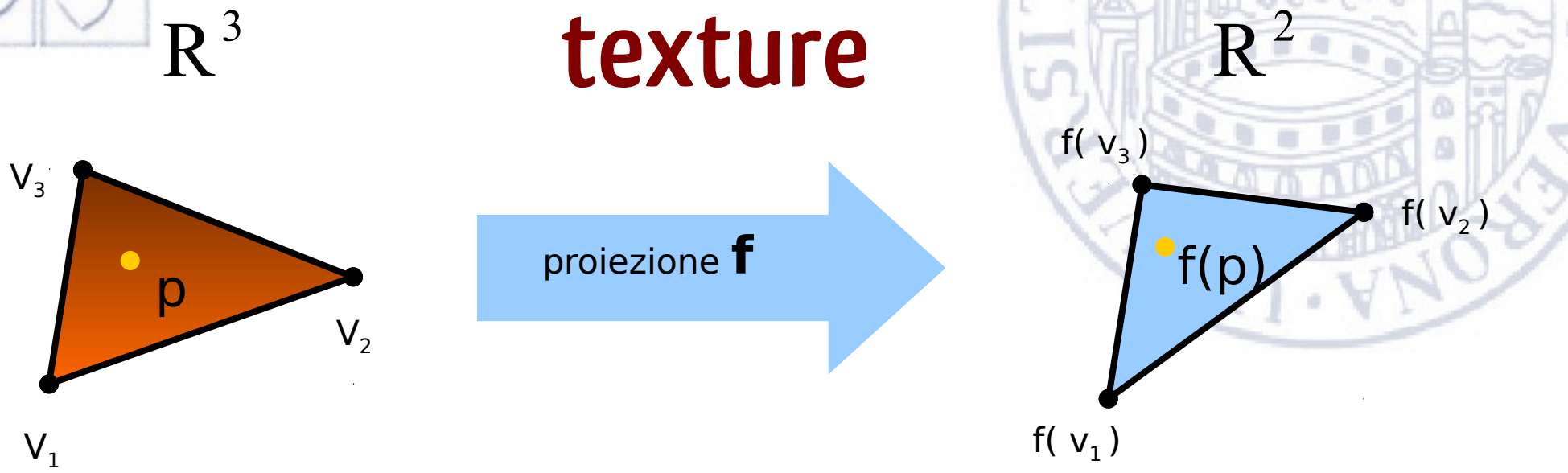
compreso: coordinate texture (per vertice!)

interpolo coordinate texture

coordinate texture interpolate

Texture RAM

Interpolazione delle coordinate texture



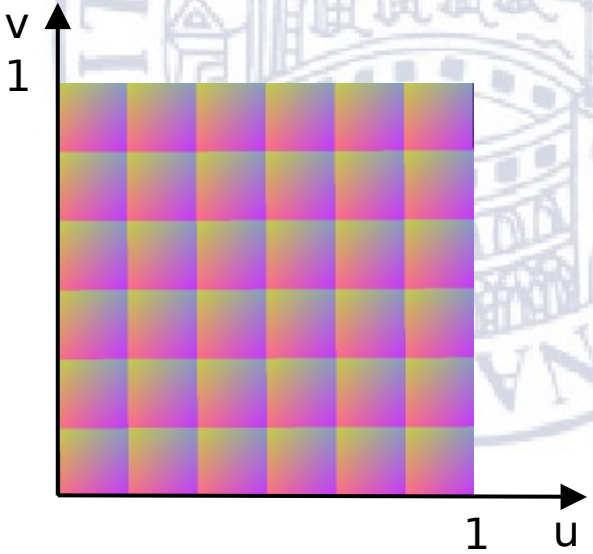
p ha coordinate baricentriche a,b,c nel triangolo v_1 v_2 v_3

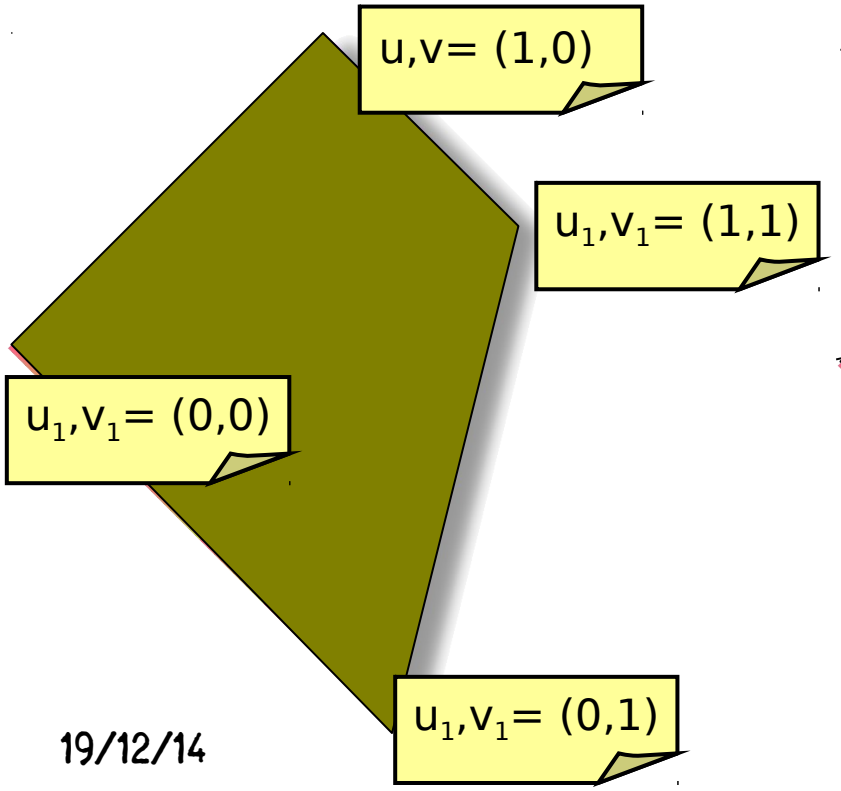
f(p) ha coordinate baricentriche a,b,c nel triangolo $f(v_1)$ $f(v_2)$ $f(v_3)$

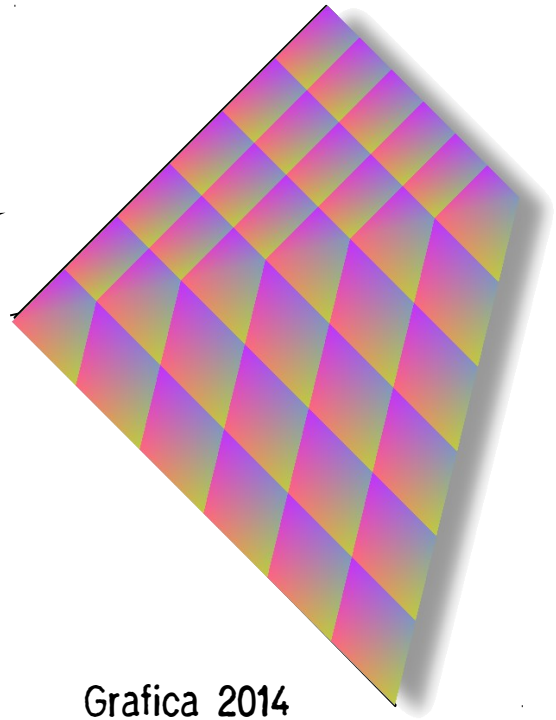
- Non vale per la proiezione prospettica poiché è solo una approssimazione che è utile per colori e normali ma non funziona quando interpoliamo coordinate texture...

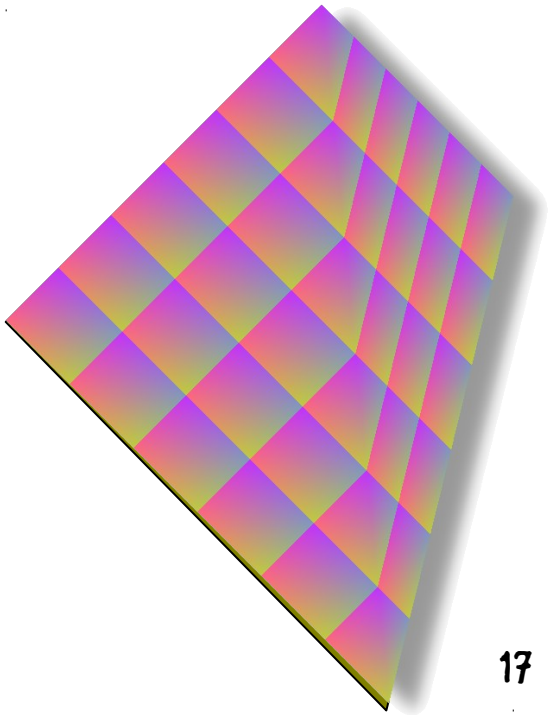
Interpolazione delle coordinate texture

- Esempio:





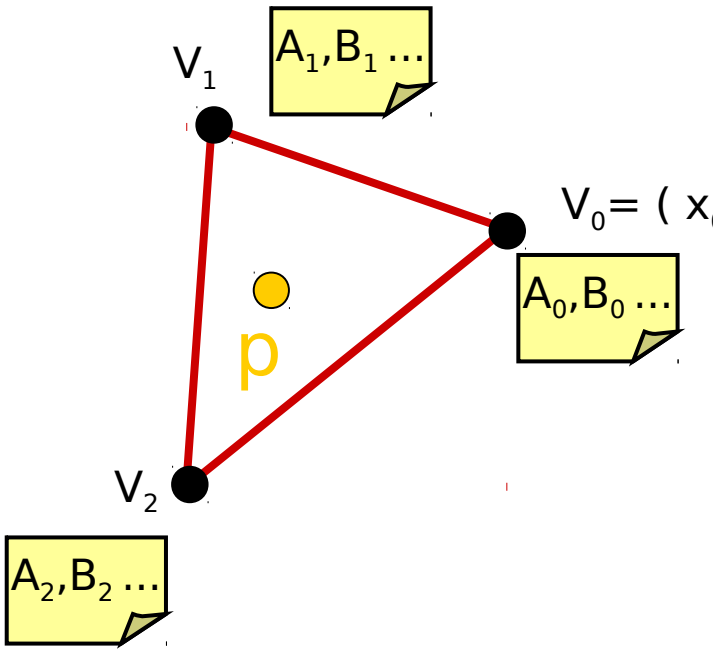




Correzione Prospettica

- p ha coordinate baricentriche $c_0\ c_1\ c_2$

$$p = c_0 v_0 + c_1 v_1 + c_2 v_2$$



attributi di p:
(senza considerare
la correzione prospettica)

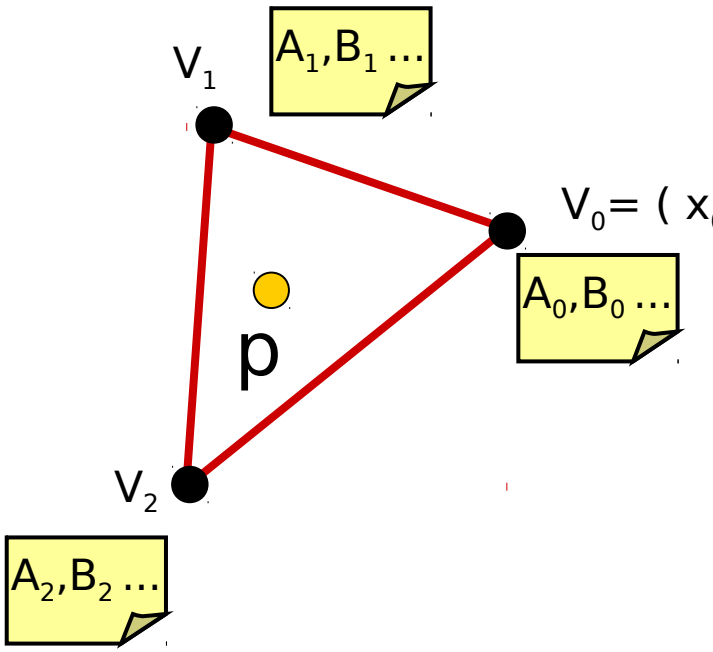
$$A_p = c_0 A_0 + c_1 A_1 + c_2 A_2$$

$$B_p = c_0 B_0 + c_1 B_1 + c_2 B_2$$

Correzione Prospettica

- **p** ha coordinate baricentriche $c_0\ c_1\ c_2$

$$p = c_0 v_0 + c_1 v_1 + c_2 v_2$$

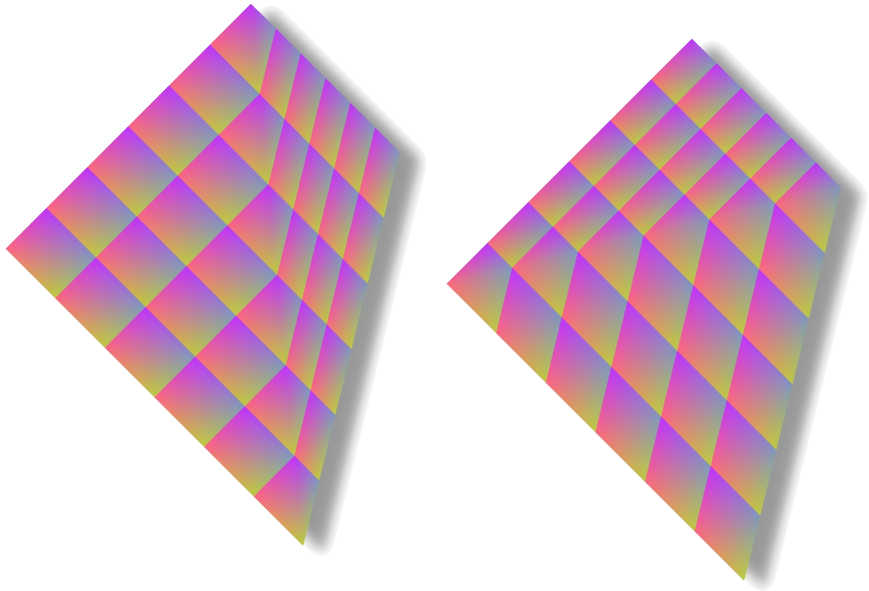


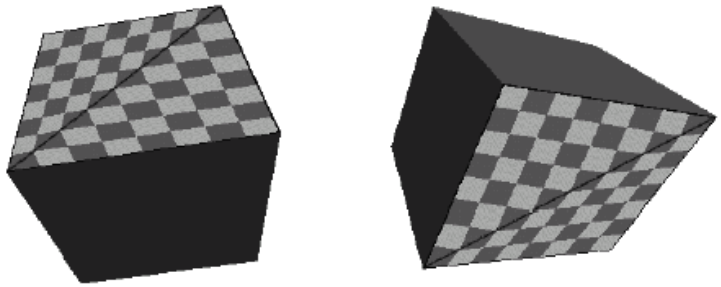
attributi di p:
(con la correzione prospettica)

$$A_p = \frac{c_0 \frac{A_0}{w_0} + c_1 \frac{A_1}{w_1} + c_2 \frac{A_2}{w_2}}{c_0 \frac{1}{w_0} + c_1 \frac{1}{w_1} + c_2 \frac{1}{w_2}}$$

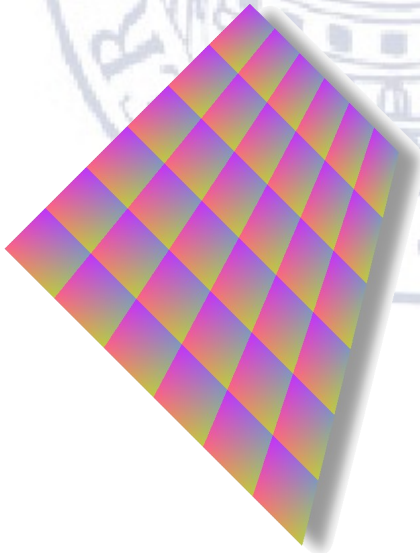
Correzione Prospettica

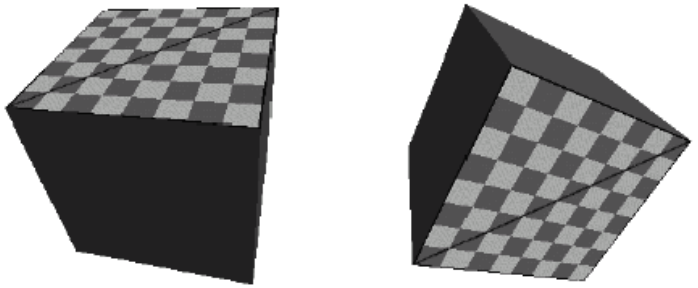
- Senza



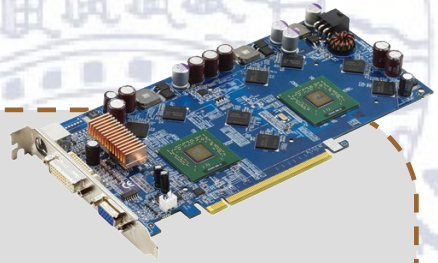


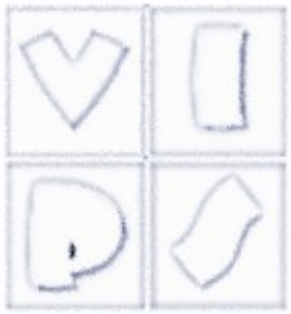
- Con





Nota: la tessitura va caricata





Nota: la tessitura va caricata

- Da disco a memoria RAM main (sulla scheda madre)
- Da memoria RAM a memoria dell'HW grafico
- Entrambe le operazioni sono piuttosto lente e sono impossibili da fare una volta per frame quindi nel progetto dell'applicazione si devono utilizzare strategie per la gestione delle texture

Assegnazione delle coordinate texture

- Due classi di soluzioni:
  - Calcolare le coordinate textures on-the-fly durante il rendering…
  - Precalcolarla (e salvarle insieme alla mesh)
- Non esiste una soluzione ideale, dipende dall'applicazione che stiamo progettando
- Modelli con una sola texture l'avranno precalcolata, per altri che variano dinamicamente l'assegneremo in rendering

Texture mapping

- Da Angel

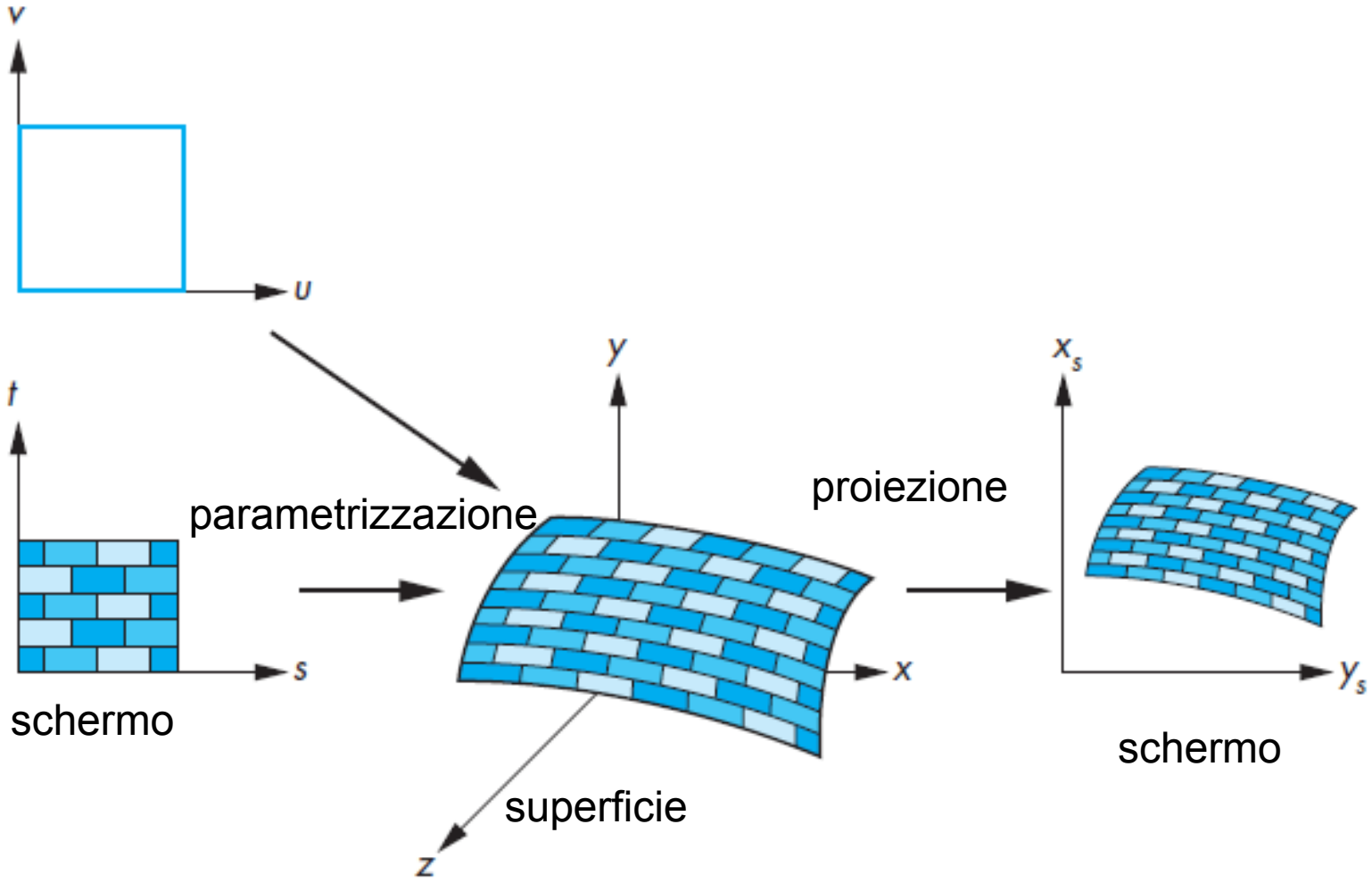
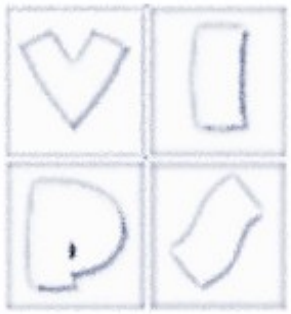


FIGURE 7.9 Texture maps for a parametric surface.

Texture mapping

- La texture può essere applicata dopo il calcolo della illuminazione con Phong (per modificare attributi come colore, luminosità o trasparenza) oppure può modificare i parametri (come le normali) che entrano nel modello di Phong.
- Un passaggio chiave è stabilire una corrispondenza univoca tra superficie dell'oggetto e texture.
- Occorre definire la funzione di parametrizzazione W() che associa un punto (s; t) della texture ad un punto P della superficie dell'oggetto 3D (è una funzione che "spalma " la texture sulla superficie).
- Il punto P viene poi mappato dalla proiezione in un punto (xs; ys) dello schermo. Il rendering della texture si occupa poi di stabilitre il valore di texture da associare a ciascun pixel.



Parametrizzazione

- Nella mappatura in un passo si definisce (in forma analitica) la funzione W che definisce la corrispondenza tra i pixel della texture ed i punti della superficie.
- Questo si può fare quando si ha la descrizione parametrica della superficie soggiacente alla maglia poligonale.
- Altrimenti si specifica tabularmente la corrispondenza W^{-1} tra vertici della maglia e punti della texture.
- Se la superficie è data in forma parametrica, ad ogni suo punto P sono associate due coordinate (parametri) $(u; v)$.
- Per ottenere W basta specificare la mappa che va da $(u; v)$ sulla superficie a $(s; t)$ nella texture.
- E' opportuno che W sia invertibile. Spesso è l'identità (con qualche fattore di normalizzazione).

Parametrizzazione

- Ad esempio si consideri un cilindro di altezza h. per il quale si definisce la funzione

$$W : (\theta, z) \rightarrow (s, t) = (\frac{m}{2\pi}\theta, \frac{n}{h}z)$$

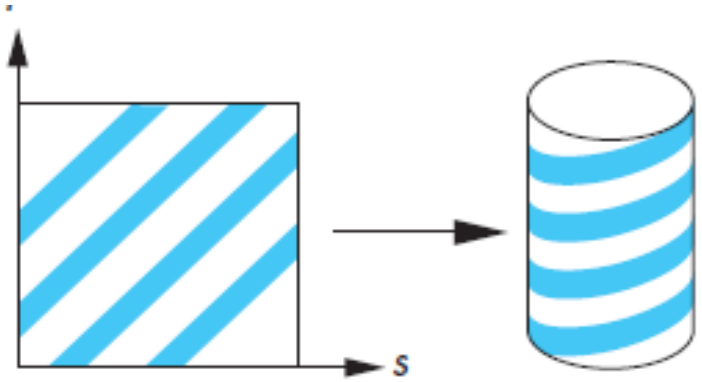
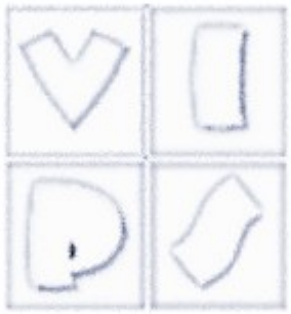


FIGURE 7.13 Texture mapping with a cylinder.

- dove (m; n) sono le dimensioni della texture map.



S/O - Mapping

- Una tecnica più generale, che si può usare senza conoscere l'equazione parametrica della superficie, è la mappatura in due passi
- Si mappa la texture su una superficie intermedia semplice, in modo che la parametrizzazione (corrispondenza punti-superficie con pixel-texture) sia immediata; questa prende il nome di S-mapping
- Quindi si mappa ogni punto della superficie intermedia in un punto della superficie in esame; questa prende il nome di O-mapping
- La concatenazione dei due mapping genera la corrispondenza W tra i pixel della texture ed i punti dell'oggetto

- Il primo passaggio (S-mapping) è in genere semplice; basta scegliere superfici facili da parametrizzare
- Ad esempio si può prendere come superficie intermedia un cilindro (vedi slide precedente)
- Oltre al cilindro è facile fare l'S-mapping con cubi, piani e sfere.
- In genere si considera la superficie intermedia come esterna all'oggetto da tessiturare.

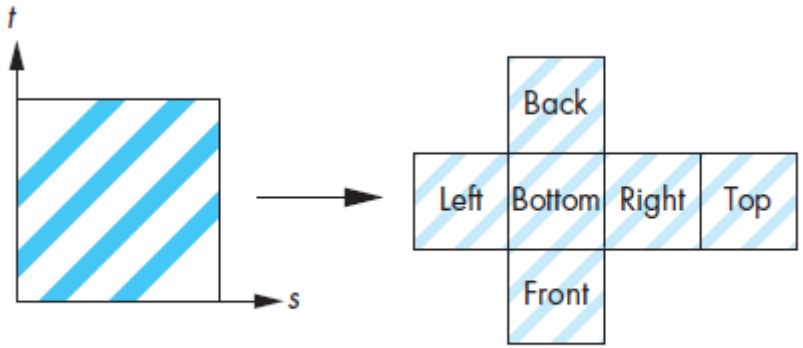
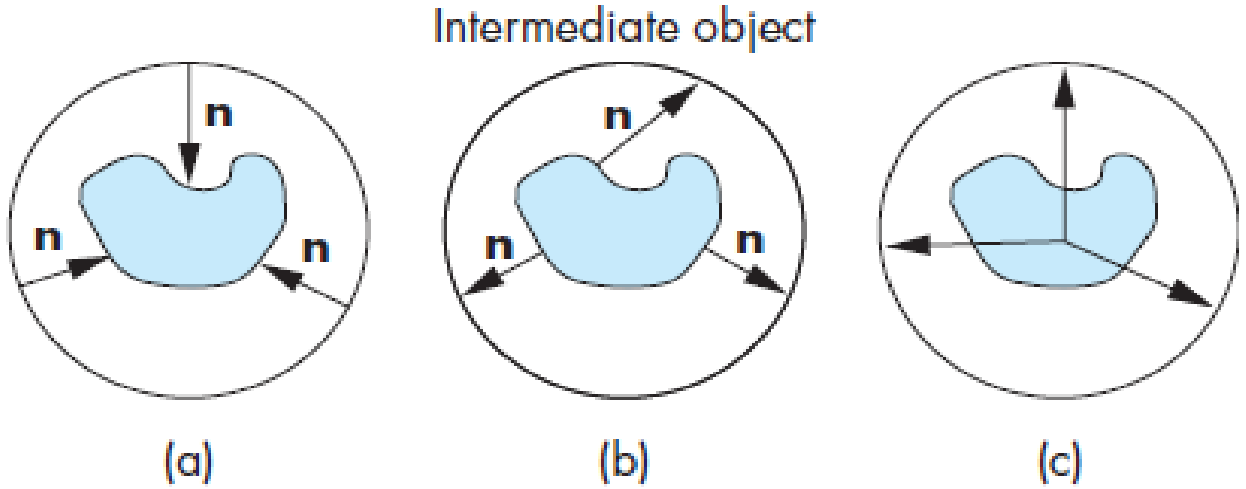


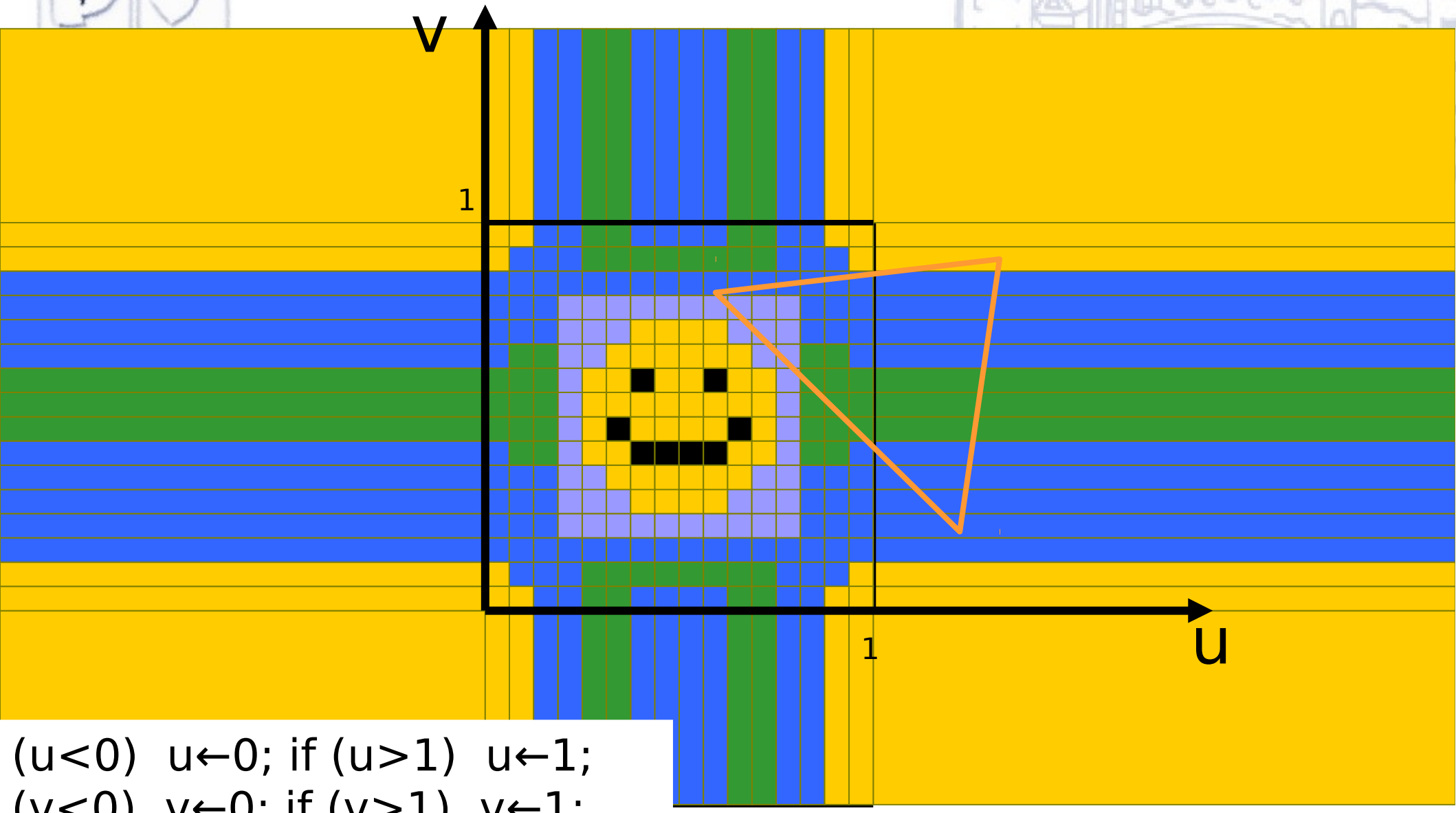
FIGURE 7.14 Texture mapping with a box.

- Per l'O-mapping ci sono varie scelte
  - Si considera la normale uscente da un punto dell'oggetto; il raggio che passa per tale punto e con direzione tale normale intersecherà la superficie intermedia in un punto, stabilendo così l'O-mapping
  - Anziché usare la normale si può usare la retta che congiunge il centroide dell'oggetto con il punto considerato
  - Si può considerare la normale in un punto della superficie intermedia e la retta che passa per tale punto e con direzione questa normale, intersecherà la superficie dell'oggetto in un punto, stabilendo un altro possibile O-mapping

Esempi di O-mapping.
(a) Usando la normale alla superficie intermedia.
(b) Usando la normale dalla superficie dell'oggetto.
(c) Usando i raggi dal centro dell'oggetto.
(©Angel)

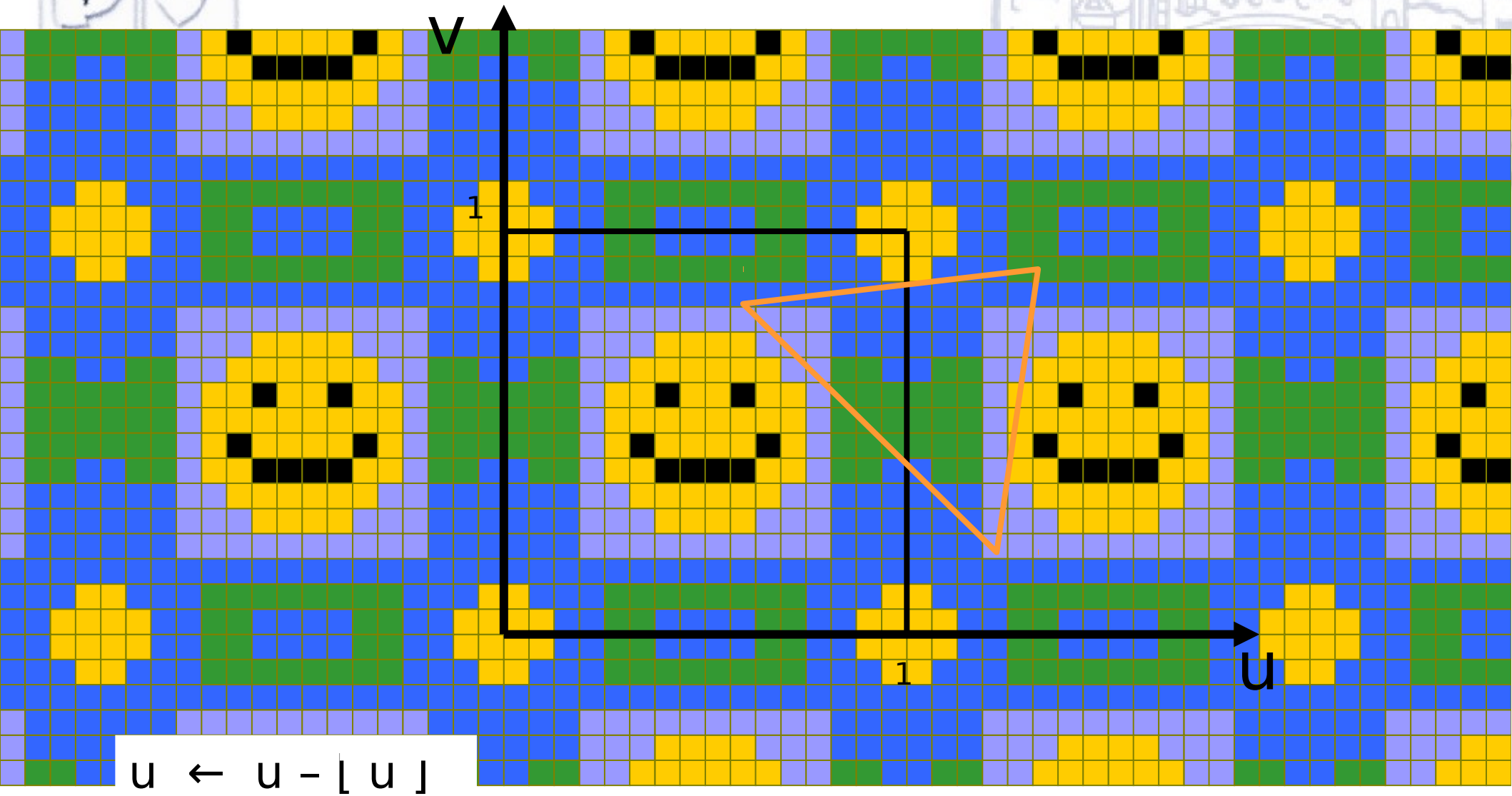


Texture fuori dai bordi: modo *clamp*



(u<0) u←0; if (u>1) u←1;
(v<0) v←0; if (v>1) v←1;

Texture fuori dai bordi: modo *repeat*



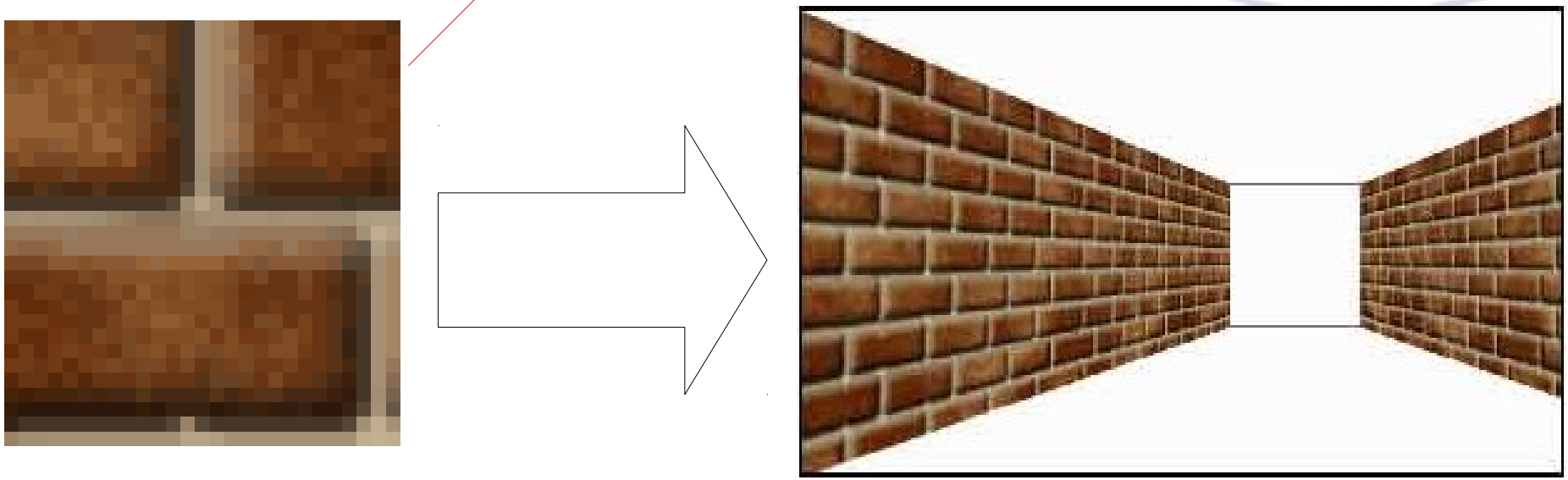
$u \leftarrow u - \lfloor u \rfloor$

$v \leftarrow v - \lfloor v \rfloor$

Tessiture ripetute

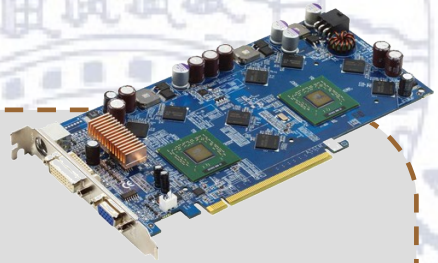
- Tipico utilizzo:

Nota: deve essere TILABLE



Molto efficiente in spazio: una sola texture mappa su molti triangoli

Texture Mapping



Vertici & loro attributi

computazioni per vertice

Vertici poriettati & attributi computati

set-up

set-up

set-up

rasterizer punti

rasterizer

rasterizer segmenti

Frammenti & attributi interpolati

computazioni per frammento

Screen buffer

texture look-up

compreso: coordinate texture (per vertice!)

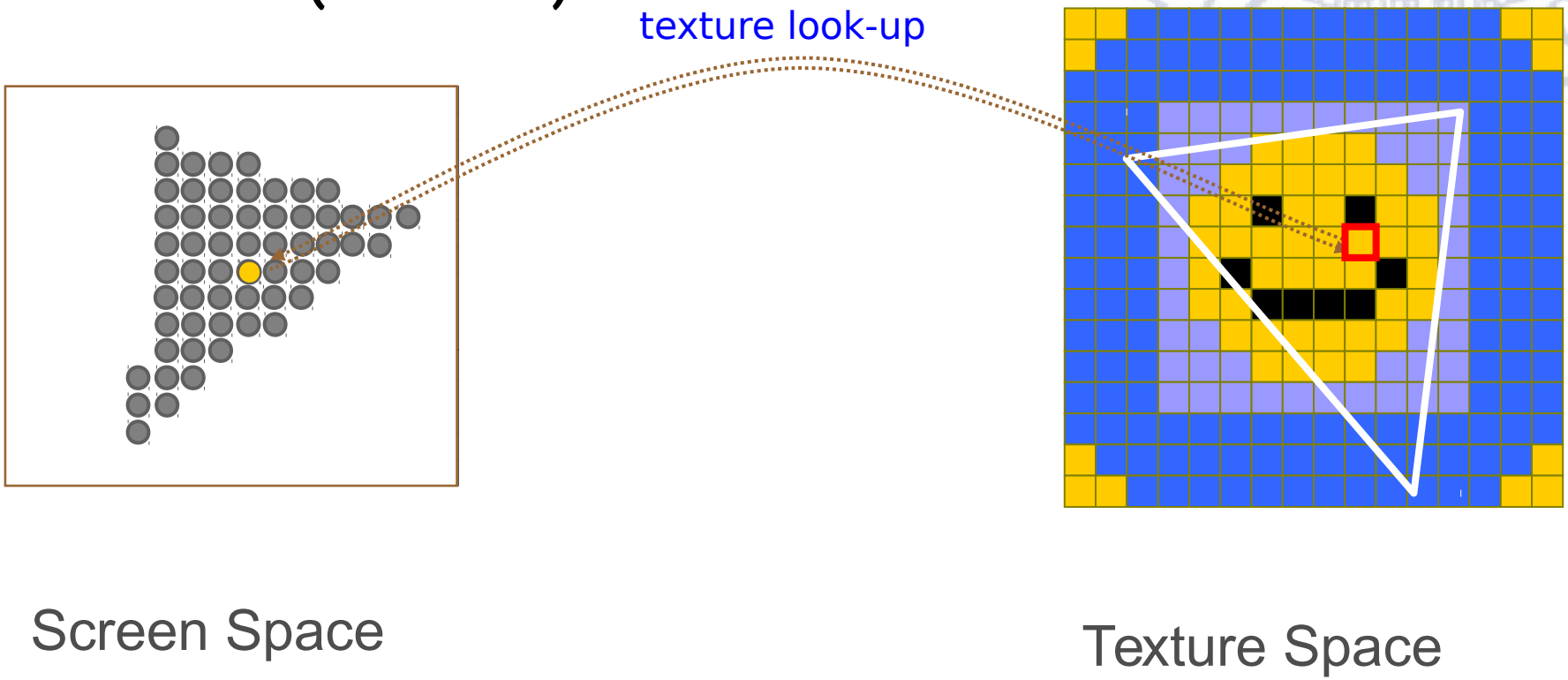
interpolo coordinate texture

coordinate texture interpolate

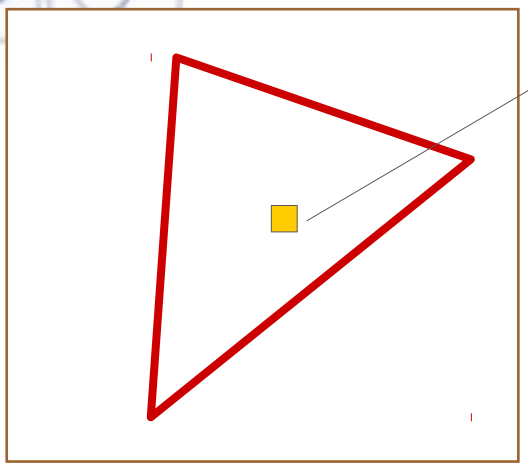
Texture RAM

Texture Look-up

- Un frammento ha coordinate non intere (in texels)

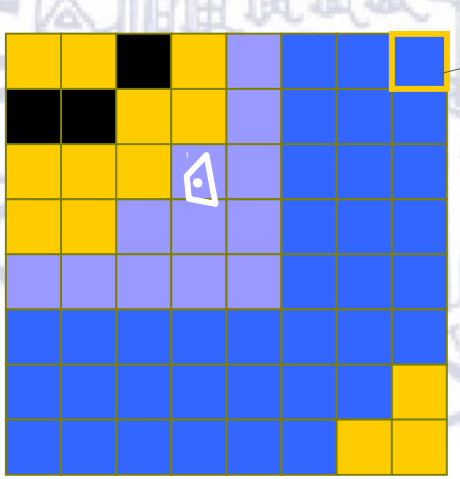


Texture Look-up



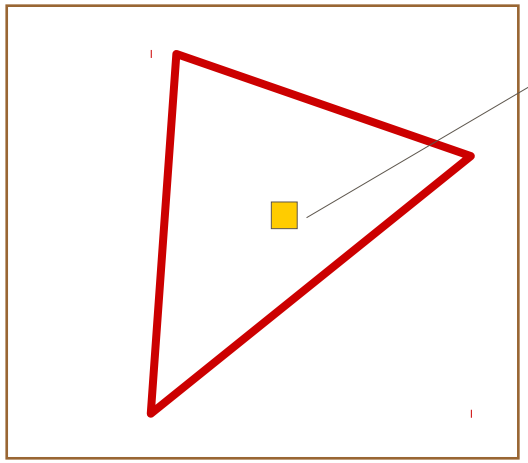
un pixel = meno di un texel

magnification



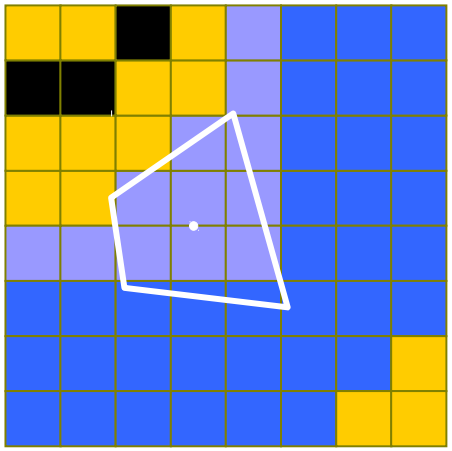
Screen Space

Texture Space



un pixel = più di un texel

minification

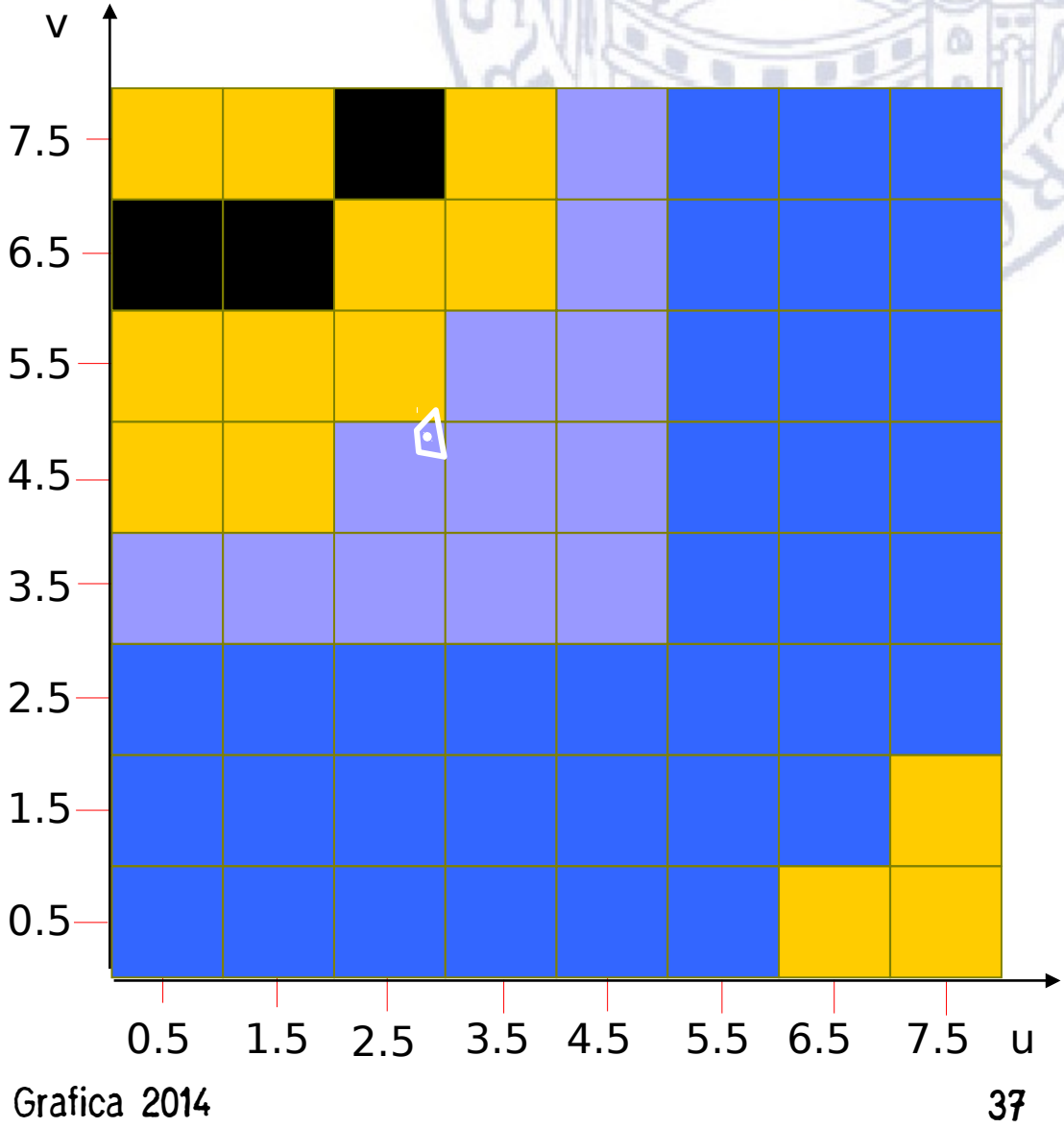


Caso Magnification

Soluzione 1:
prendo il texel in cui sono

(equivale a prendere
il texel più vicino)

equivale ad arrotondare
alle coordinate texel
intere



"Nearest Filtering"

Caso Magnification

Nearest Filtering: risultato visivo

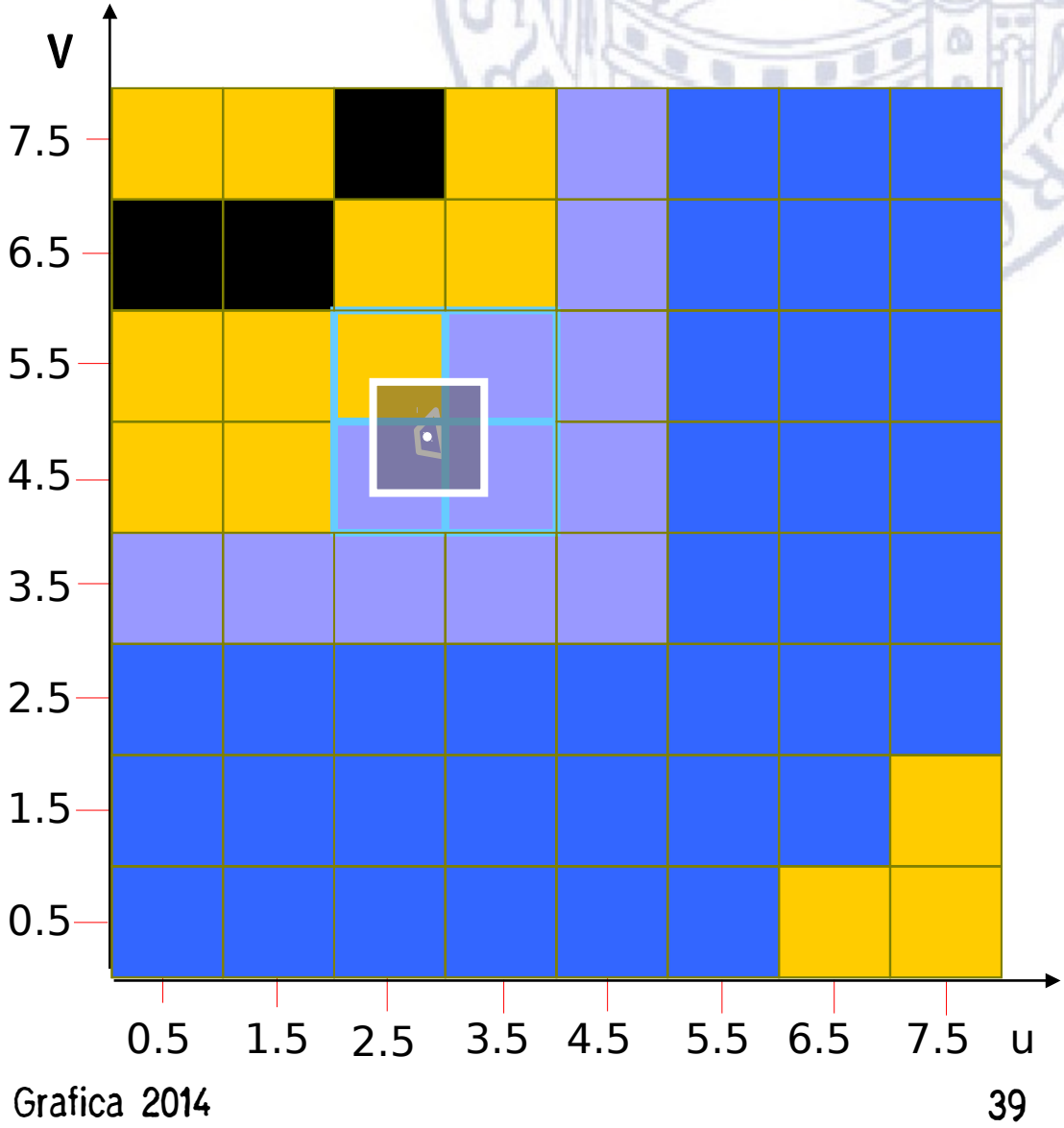
texture 128x128

"si vedono i texel !"

Caso Magnification

Soluzione 2:
Medio il valore dei quattro texel più vicini

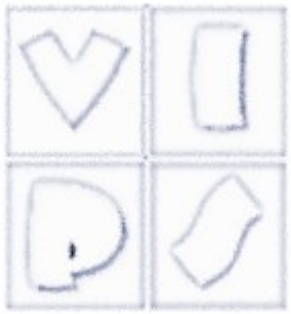
Interpolazione Bilineare



Caso Magnification

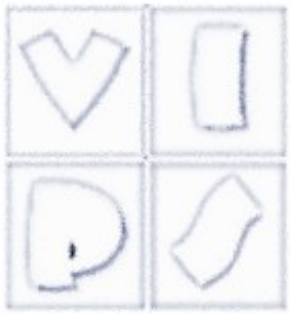
Bilinear Interpolation: risultato visivo

texture 128x128

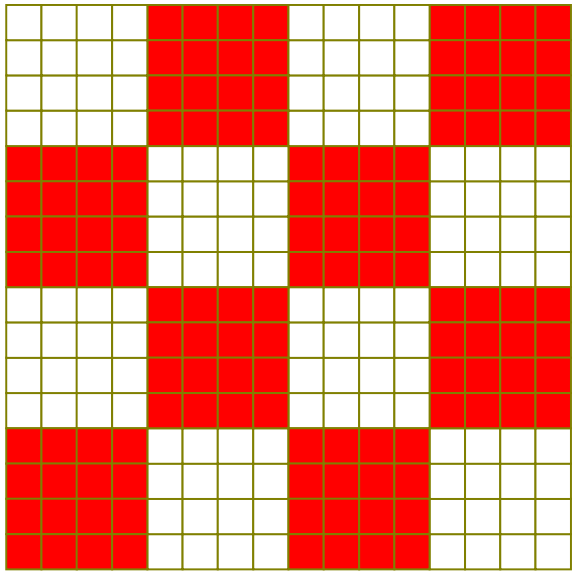


Caso Magnification

- Modo Nearest:
  - si vedono i texel
  - va bene se i bordi fra i texel sono utili
  - più veloce
- Modo Interpolazione Bilineare
  - di solito qualità migliore
  - può essere più lento
  - rischia di avere un effetto "sfuocato"



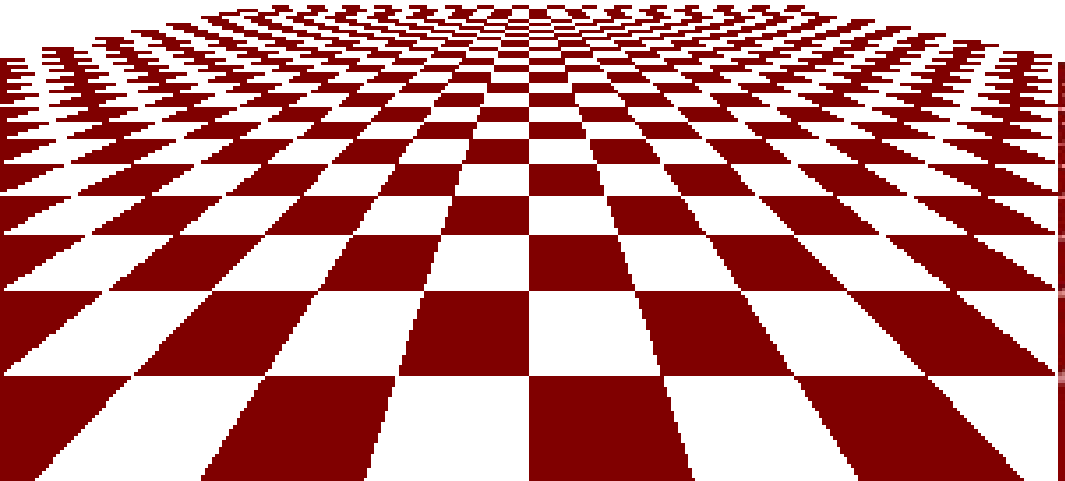
Caso Minification

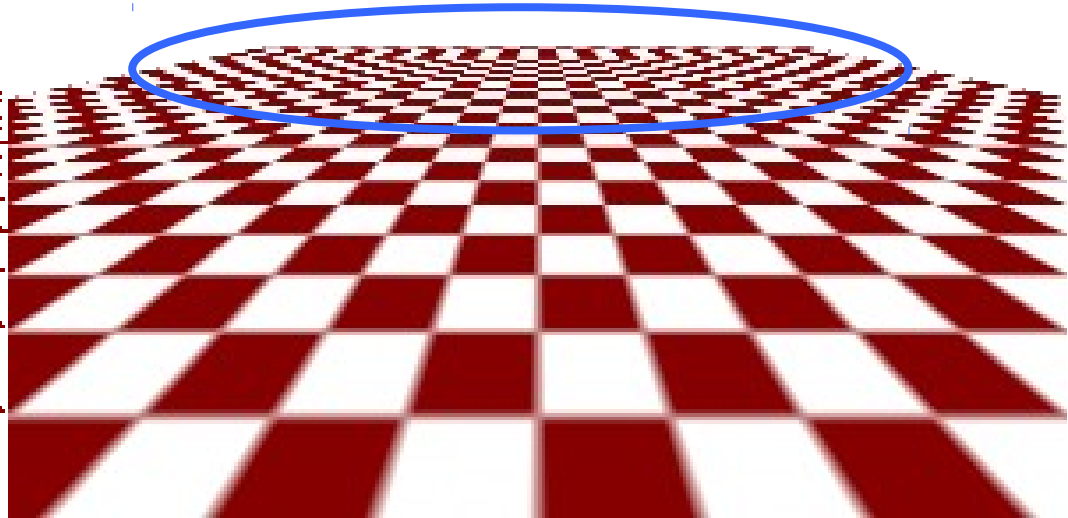


Nearest Filtering

Bilinear interpolation

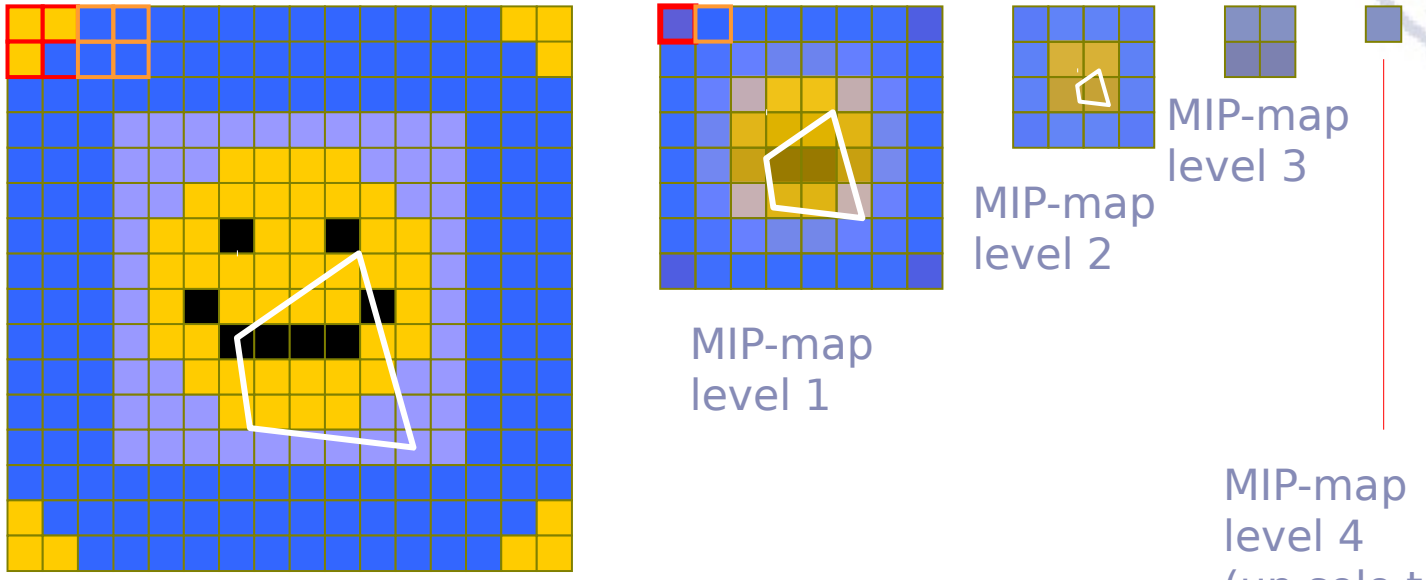
non risolve il problema

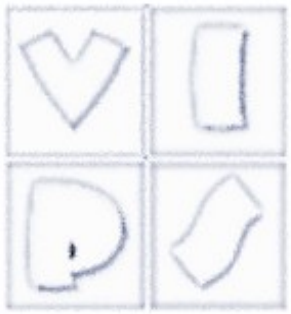




Caso Minification: MIP-mapping

MIP-mapping: "Multum In Parvo"

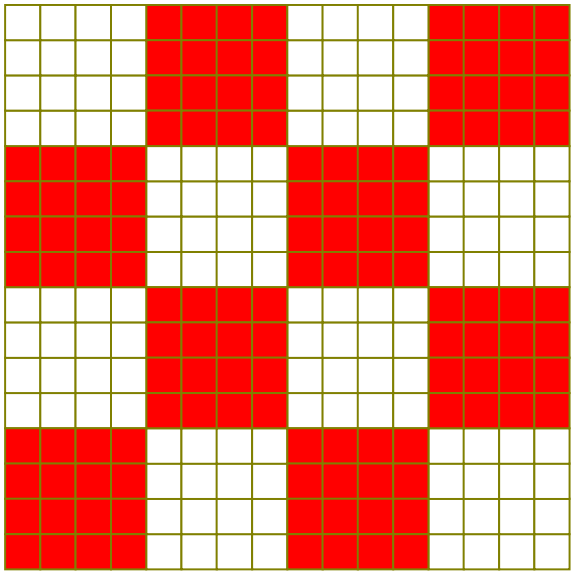


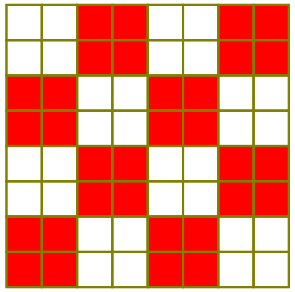


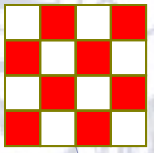
Mipmap Math

- Definiamo un fattore di scala, ρ=texels/pixel come valore massimo fra ρ_x e ρ_y, che può variare sullo stesso triangolo, può essere derivato dalle matrici di trasformazione ed è calcolato nei vertici, interpolato nei frammenti
- Il livello di mipmap da utilizzare è: $\log_2 \rho$ dove il livello 0 indica la massima risoluzione
- Il livello non è necessariamente un numero intero e può quindi essere arrotondato

Caso Minification: MIP-mapping



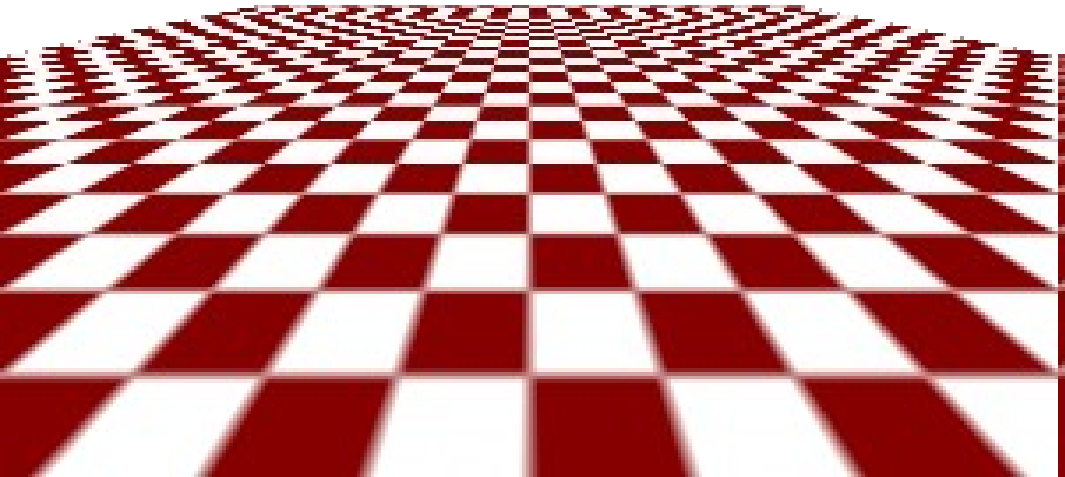


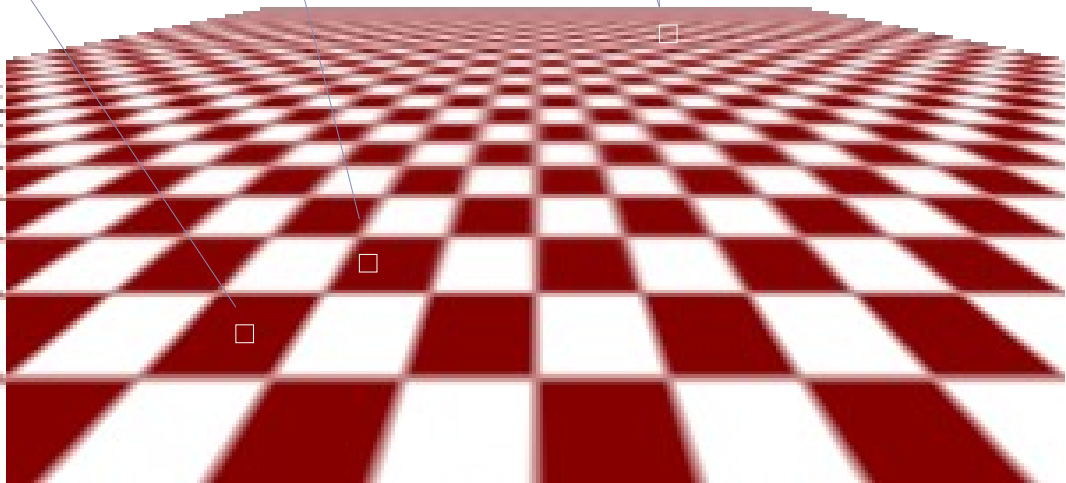


Bilinear interpolation

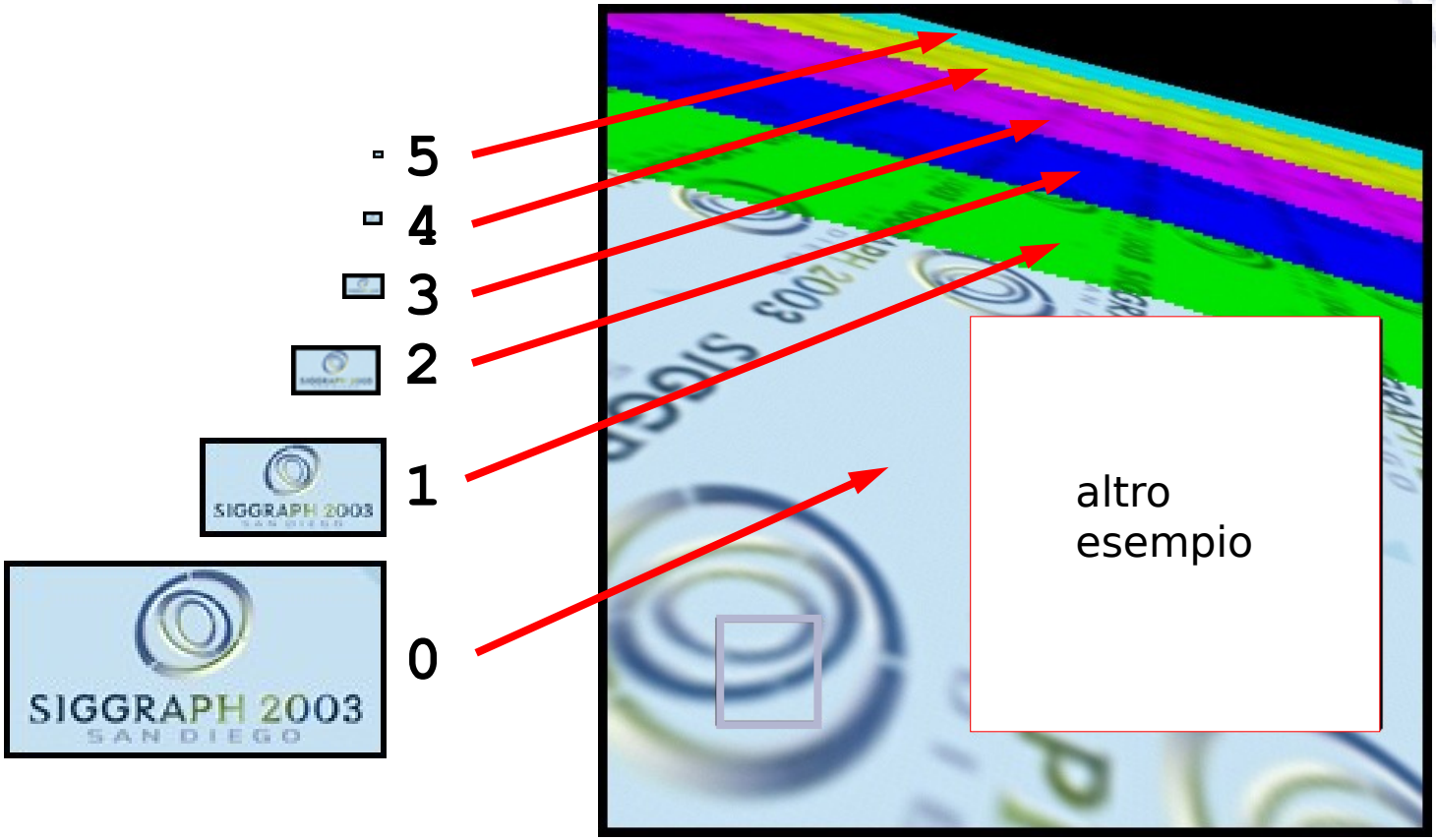
non risolve il problema

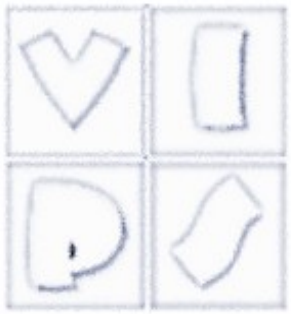
MIP-mapping





Caso Minification: MIP-mapping

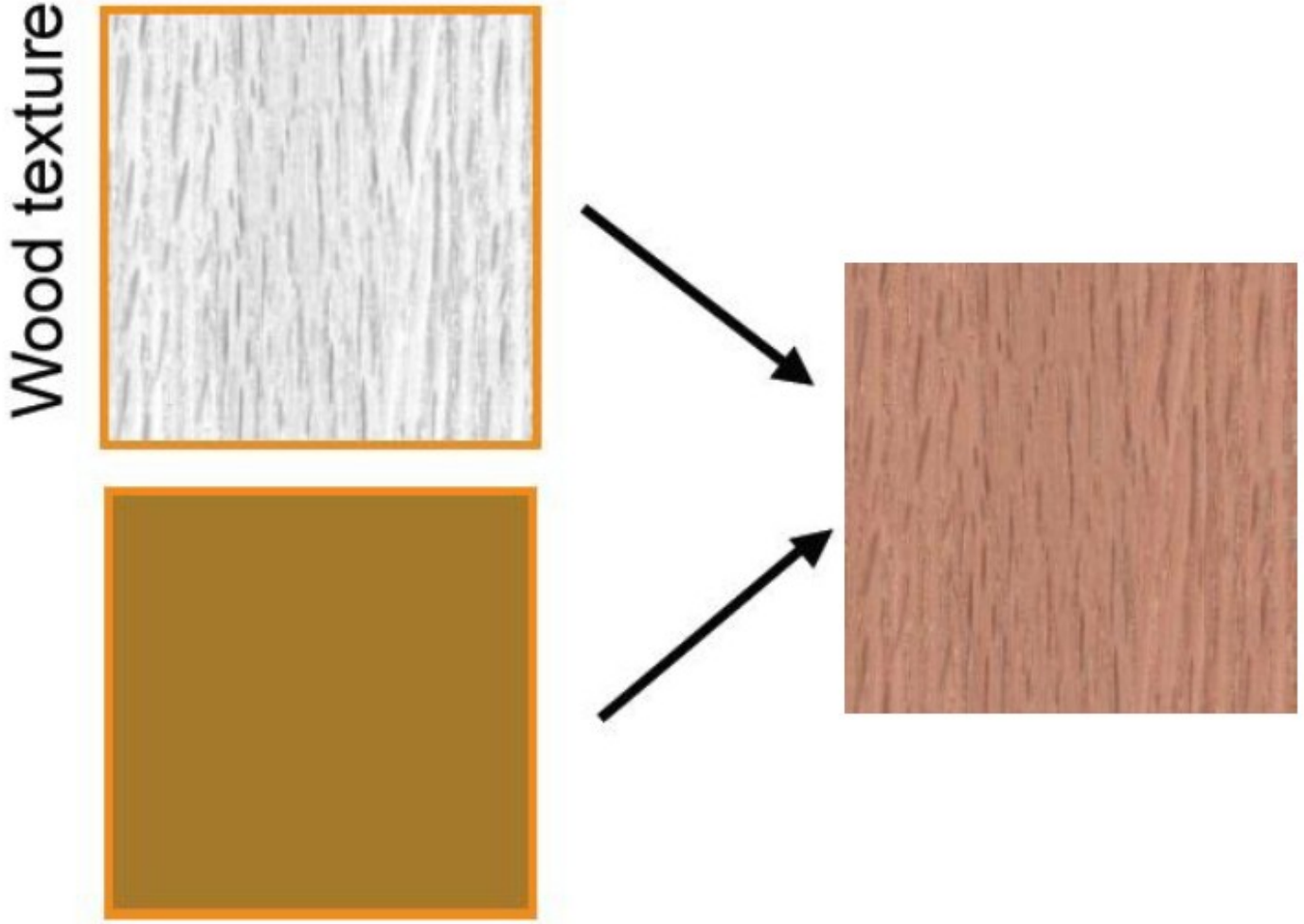




Uso delle texture

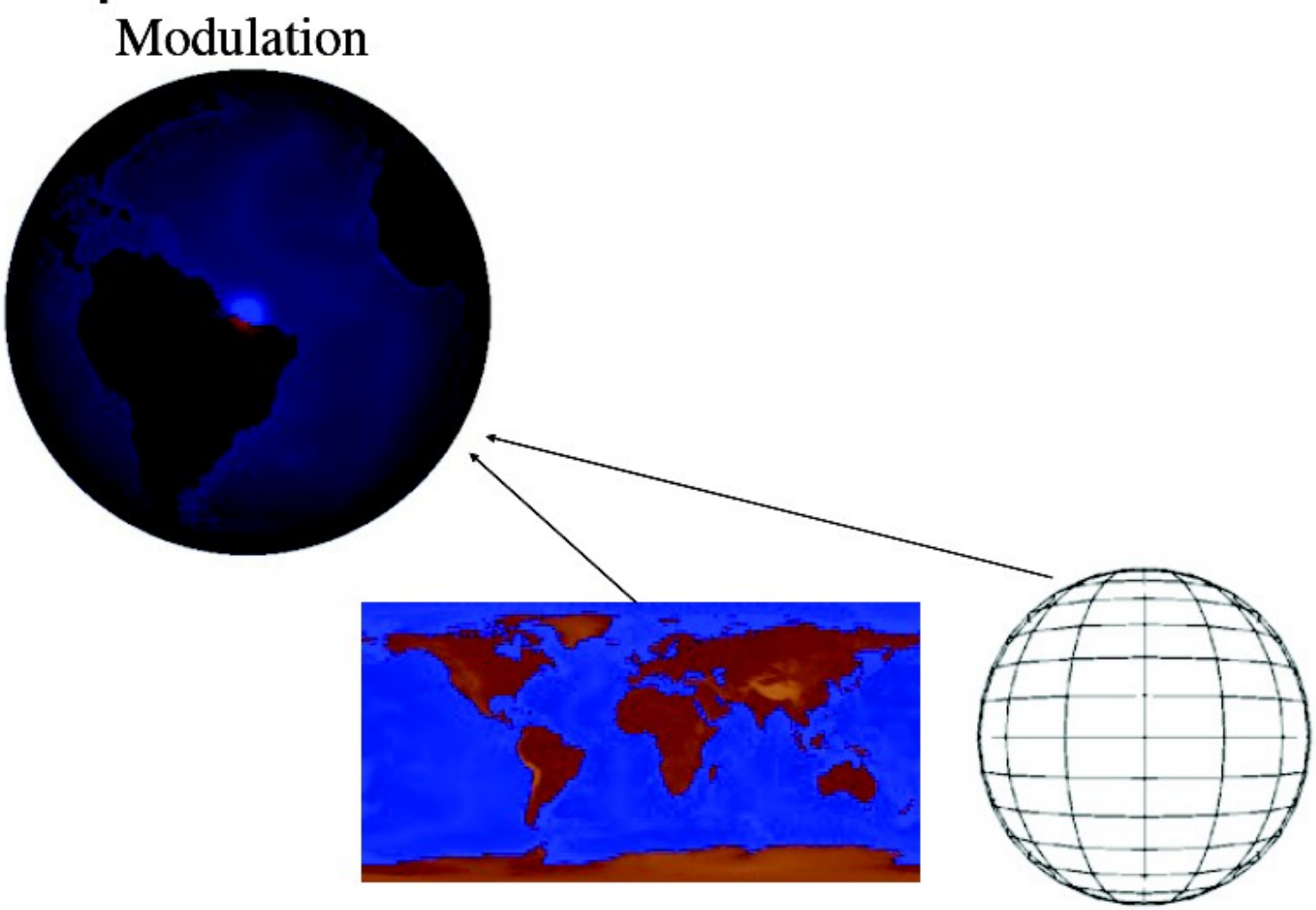
- La “color mapping” che si ottiene nel modo descritto applicherebbe la texture come un adesivo (decalcomania)
- Si usa però tipicamente modulare il colore della texture moltiplicandolo per il valore risultante dallo shading.
  - In questo caso è tipico assegnare colore bianco (o grigio) alle componenti ambientale, diffusa e speculare modello di Phong
  - Si ottiene di modulare il colore della texture con l’intensità di illuminaizone della superficie calcolata dal modello di Phong.
  - Problema: si perdono gli highlights, che assumono il colore della texture. Soluzione: modulare con la texture separatamente solo le componenti diffusa, ambientale ed emissiva e sommare separatamente alla fine la componente speculare.

Modulazione



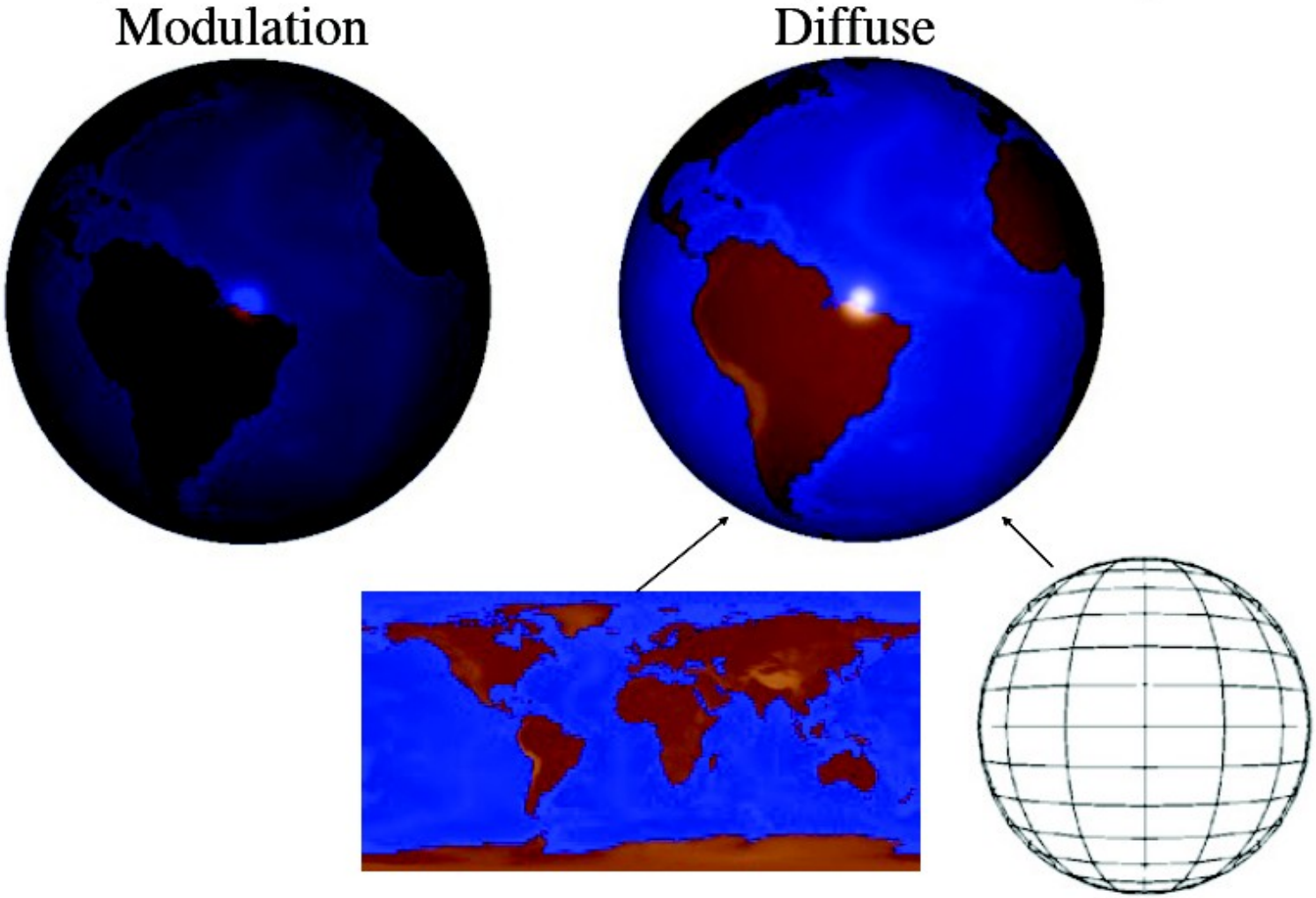
$$I = T(s,t)(I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L + K_T I_T + K_S I_S)$$

Modulazione



$$I = T(s,t)(I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L + K_T I_T + K_S I_S)$$

Modulazione

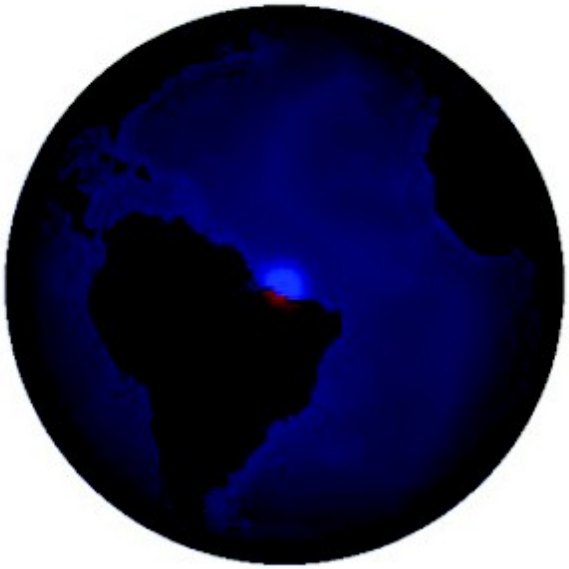


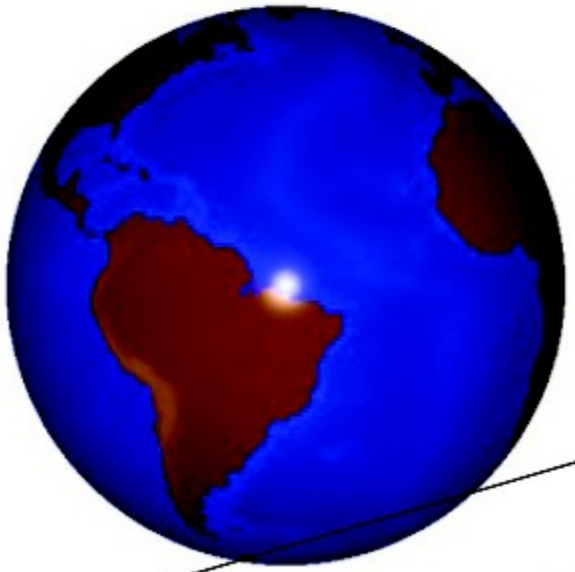
$$I = I_E + K_A I_A + \sum_L \left(T(s,t) \left((N \bullet L) + K_S (V \bullet R)^n \right) S_L I_L + K_T I_T + K_S I_S \right)$$

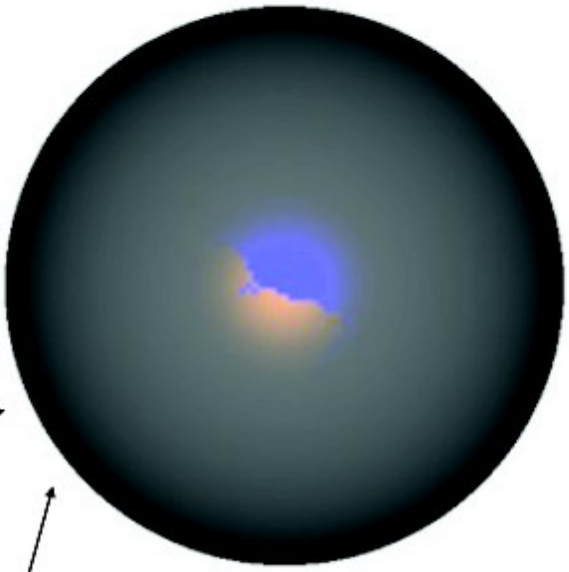
Modulation

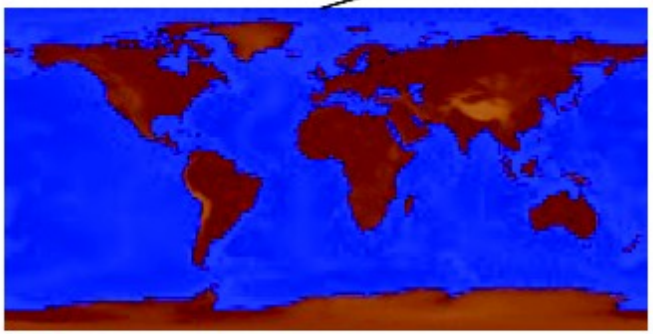
Diffuse

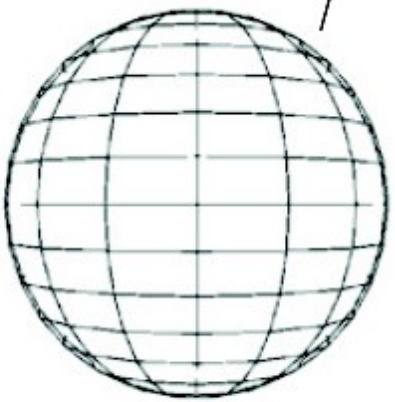
Specular











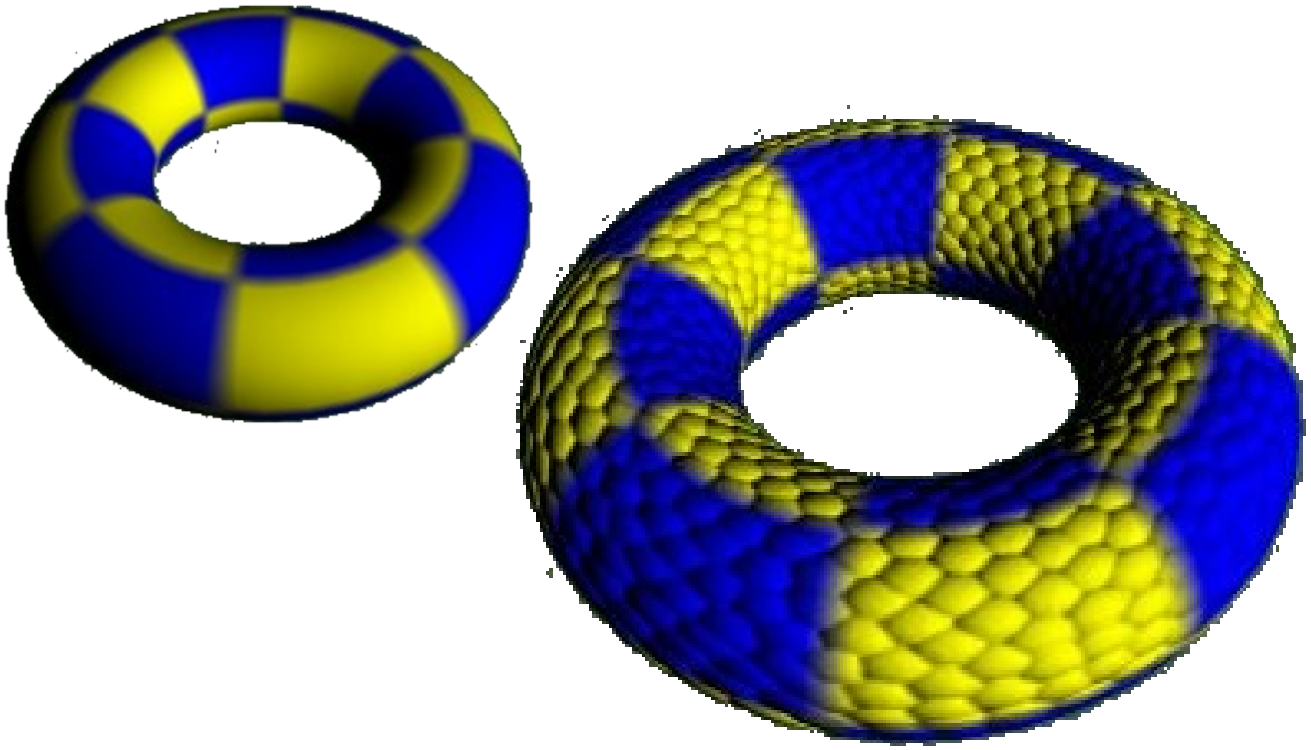
$$I = I_E + K_A I_A + \sum_L \left(K_D (N \bullet L) + T(s,t) (V \bullet R)^n \right) S_L I_L + K_T I_T + K_S I_S$$

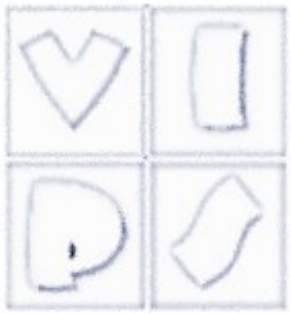
Uso delle texture

- Vi sono molte altre tecniche basate sull'uso di textures.
  - Bump map: questa tecnica perturba la normale in un punto con il valore corrispondente nella texture; altera lo shading della superficie senza modificare la geometria
  - Mappa delle normali: estensione della precedente, specifica la normale in ogni punto.
  - Mappa di riflessione: detta anche environment map usa una texture per dare l'impressione che l'oggetto rifletta l'ambiente circostante.
  - Mappa della luce (light map) contiene il risultato del calcolo dell' illuminazione fatto off-line (modello globale, view-independent).
  - Mappa di trasparenza: si usa per modulare l'opacità dell'oggetto; in tal modo alcune parti possono essere rese trasparenti, altre opache. Spesso (OpenGL) tale informazione viene accorpata al colore (RGBA)
  - Mappe di emissione, specularità

Bump mapping

- Un'ulteriore modifica all'apparenza del rendering può essere effettuata usando il bump mapping





Bump mapping

- Il metodo prevede di variare la normale alla superficie pixel per pixel utilizzando la formula:

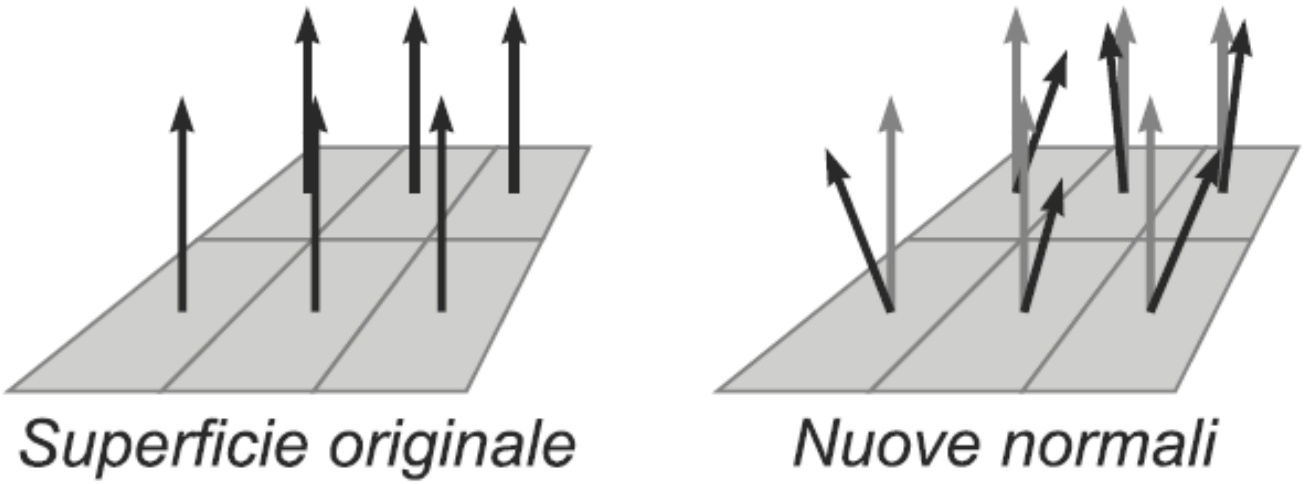
$$\vec{\mathbf{N}}_{new} = \vec{\mathbf{N}}_{old} + \vec{\mathbf{D}};$$

- I texel in questo caso sono utilizzati ad uno stadio diverso rispetto ai color texel, prima del calcolo dell'equazione di illuminazione

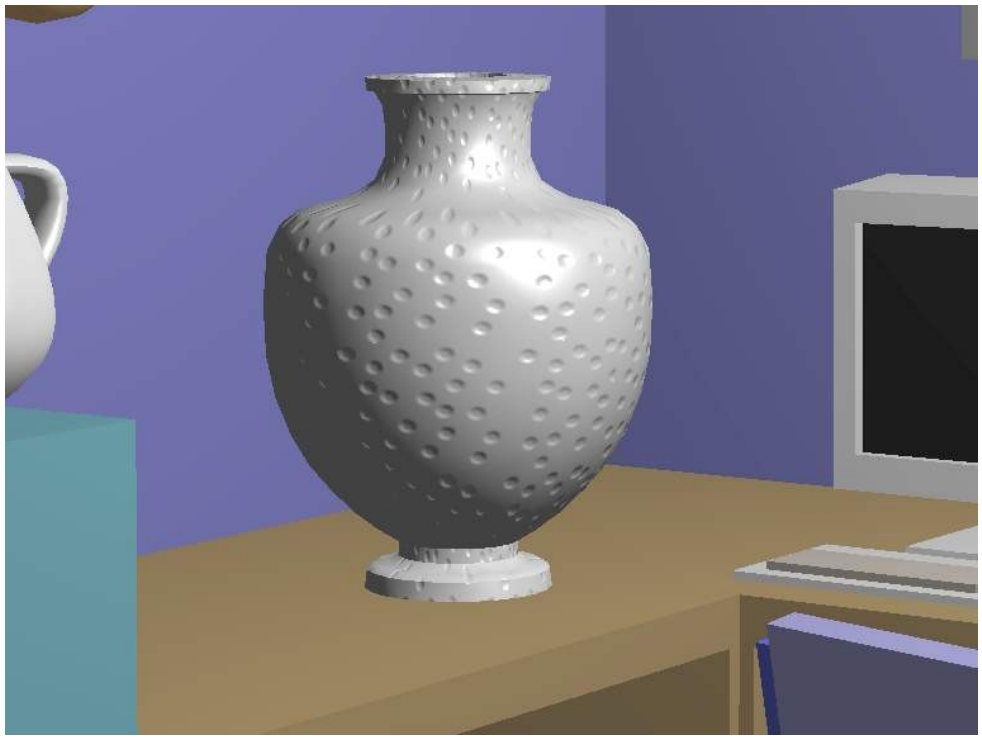
$$\vec{\mathbf{D}} = (\Delta x, \Delta y, \Delta z)$$

Bump mapping

- L'effetto che si ottiene è una perturbazione del valore delle normali che altera il rendering senza modificare la geometria



Esempio



Analisi geometrica

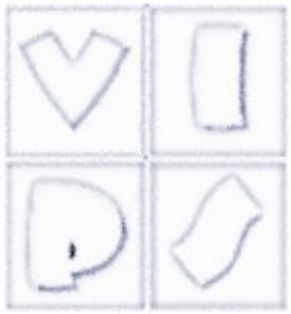
- Sia $P(u, v)$ un generico punto della superficie (parametrizzata) da perturbare.
- Sia data la mappa $B(u, v)$ che specifica la perturbazione (virtuale) da applicare alla superficie, spostando il punto P lungo la sua normale della quantità $B(u, v)$ (assumiamo per semplicità s = u e t = v). La normale in P è dunque, a meno di normalizzazione

$$\mathbf{n} = P_u \times P_v$$

  - dove P_u e P_v sono le derivate parziali rispetto ai due parametri

- Se spostassimo P lungo $\mathbf{n}$ di un valore $B(u, v)$ si otterrebbe

$$P'(u, v) = P(u, v) + B(u, v)\mathbf{n}$$



Analisi

- Per calcolare la nuova normale $\mathbf{n}$ devo derivare P' rispetto a u e v:

$$P'_u = P_u + B_u\,\mathbf{n} + B(\partial n/\partial u)$$

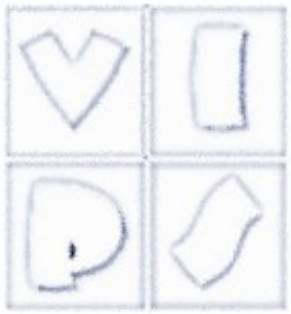
$$P'_v = P_v + B_v\,\mathbf{n} + B(\partial n/\partial u)$$

- Supponendo che $B(u, v)$ sia sufficientemente piccola e la superficie sufficientemente regolare da poter trascurare l'ultimo termine si ottiene

$$\mathbf{n}' = P'_u \times P'_v = \mathbf{n} + B_u\,\mathbf{n} \times P_v - B_v\,\mathbf{n} \times P_u$$

Bump vs Displacement mapping

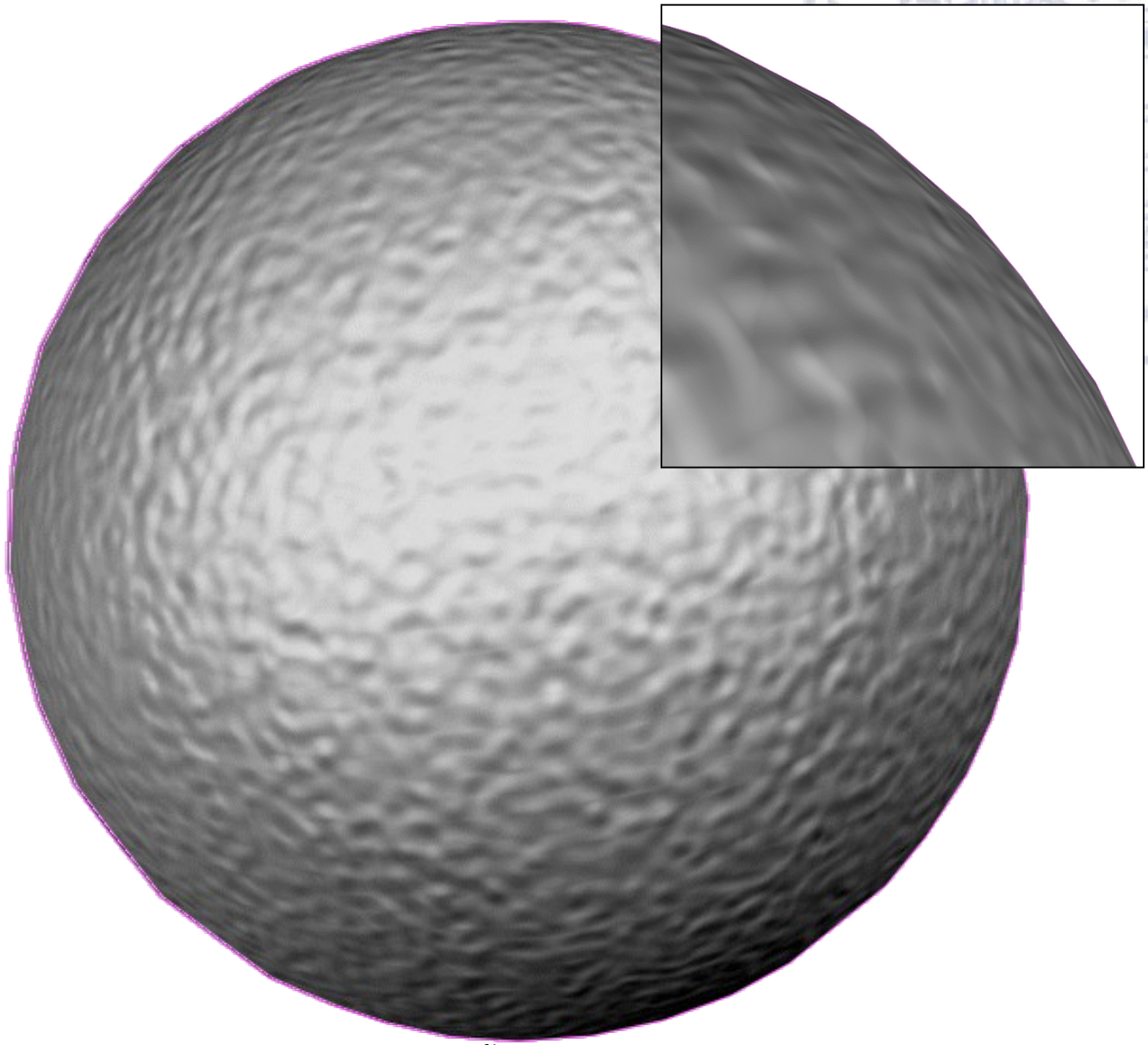
- Se applicando il modello di illuminazione a P si usa $\mathbf{n}'$ al posto di $\mathbf{n}$ si ottiene l'impressione che il punto sia stato perturbato.
- Si noti che il valore $B(u, v)$ non viene usato se non per calcolarne le derivate parziali. Si puosono memorizzare queste ultime nella bump map, risparmiando tempo di calcolo on-line.
- Alternativa: perturbare geometricamente il punto P , non solo la normale; in tal caso si parla di displacement map. Più oneroso computazionalmente, anche se i risultati sono realistici se visti da vicino, cosa non vera per il bump-mapping

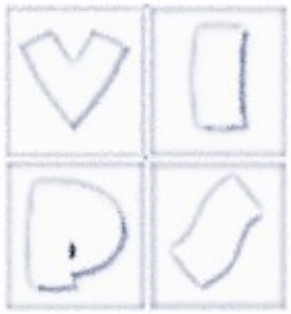


Analisi

- Il Bump mapping, modificando solo le normali della superficie, non cambia le proprietà geometriche di questa; se è liscia prima del bump-mapping, lo rimane anche dopo (si vede guardando la silhouette).
- La differenza tra bump-mapping e displacement-mapping è comunque piccola per la maggior parte delle situazioni (telecamera lontana da oggetto, superficie non vista di taglio).

Bump mapping





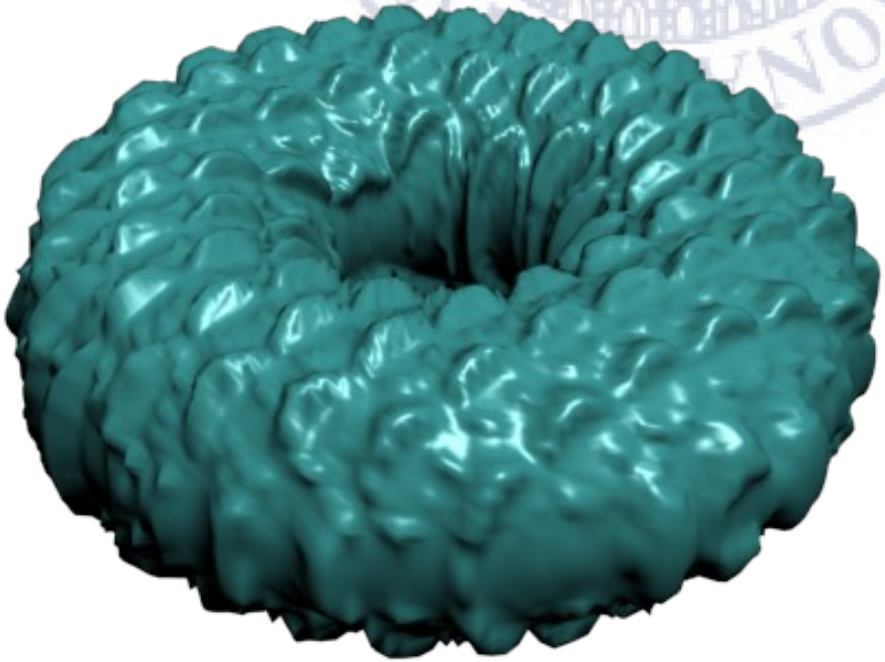
Displacement mapping

- Nel displacement mapping si modifica quindi effettivamente la geometria dell'oggetto spostando i punti della superficie:

$$P_{new} = P_{old} + h \cdot \vec{\mathbf{N}}$$

- Il displacement mapping è eseguito in fase di rendering e non modifica stabilmente la geometria della scena
- Rispetto al bump mapping anche la silhouette del modello mostra le corrette deformazioni

Displacement mapping



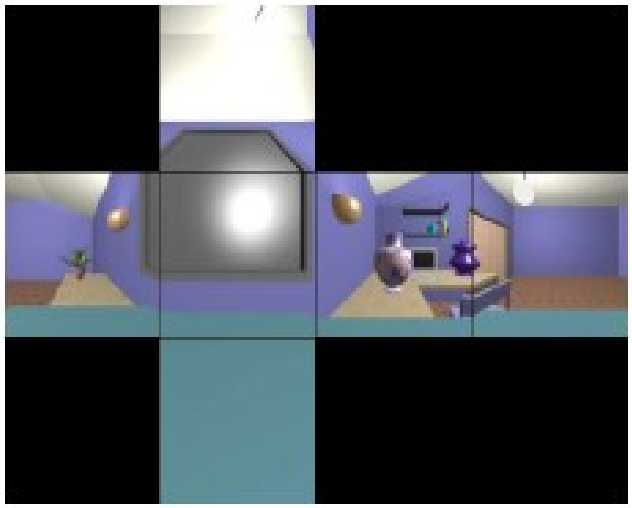
Environment map

- Dato un oggetto compatto (e relativamente piccolo) con una superficie lucida riflettente (come la teiera metallica), lo si racchiude in un cubo ideale e si ottengono sei immagini corrispondenti a sei telecamere poste nel centro dell'oggetto e con piano immagine coincidente con le facce dei cubi
- Le immagini così ottenute si compongono in una texture map che prende il nome di environment map
- Il rendering viene effettuato con O-mapping della environment map cubica usando il vettore di riflessione $\mathbf{r}_v$.

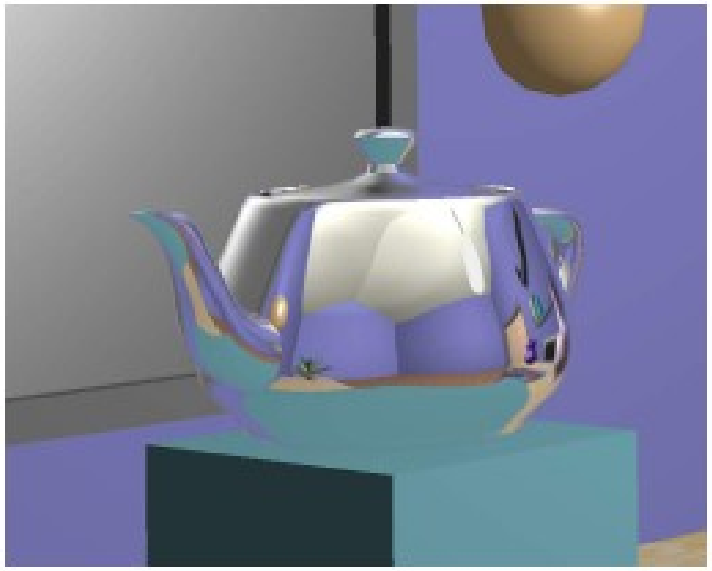
- $\mathbf{r}_v$ è il versore della direzione di vista v riflesso rispetto alla normale, e rappresenta la direzione con cui deve incidere un raggio di luce sulla superficie per essere riflesso specularmente lungo la direzione di vista:

$$\mathbf{r}_v = 2\mathbf{n}(\mathbf{n} \cdot \mathbf{v}) - \mathbf{v}$$

- Per assegnare la coordinata texture ad un punto della superficie si prosegue nella direzione di $\mathbf{r}_v$ fino ad incontrare un punto del cubo.
- L'oggetto sembrerà riflettere l'ambiente circostante



(1) Envmap



(2) Immagine

- La environment map può essere generata togliendo l'oggetto e prendendo sei viste della scena, oppure può consistere di fotografie di una scena reale. In tal caso serve ad immergere realisticamente un oggetto sintetico riflettente in una scena reale.

(3) Envmap

(4) Immagine

© Alan Watt

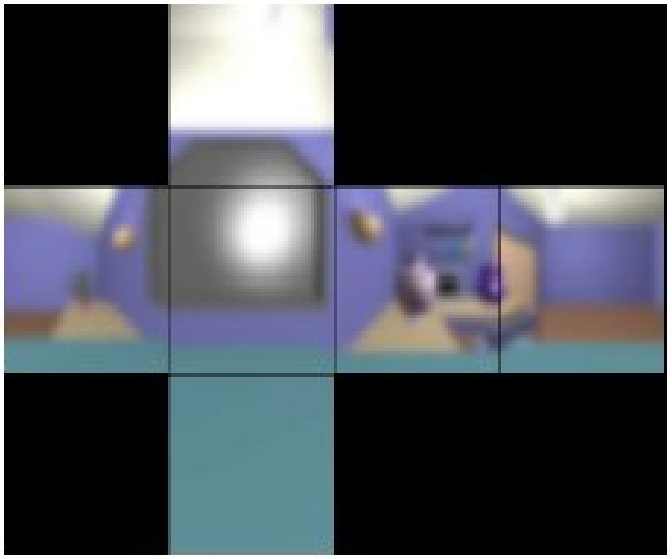
- Da notare che comunque questo tipo di tecnica è un trucco; per alcune tipologie di oggetti o situazioni particolari ci si può facilmente accorgere che la riflessione dell'oggetto non è realistica (es. non ci sono auto-riflessioni per oggetti non convessi).

- La tecnica qui delineata (mappatura cubica) è una delle possibili; un'altra fa uso di una mappatura sferica, ovvero l'oggetto viene racchiuso in una sfera e la mappa è una immagine di forma circolare che contiene una veduta deformata dell'ambiente.
- E' la stessa immagine che si ottiene dalla proiezione ortografica di una sfera perfettamente riflettente

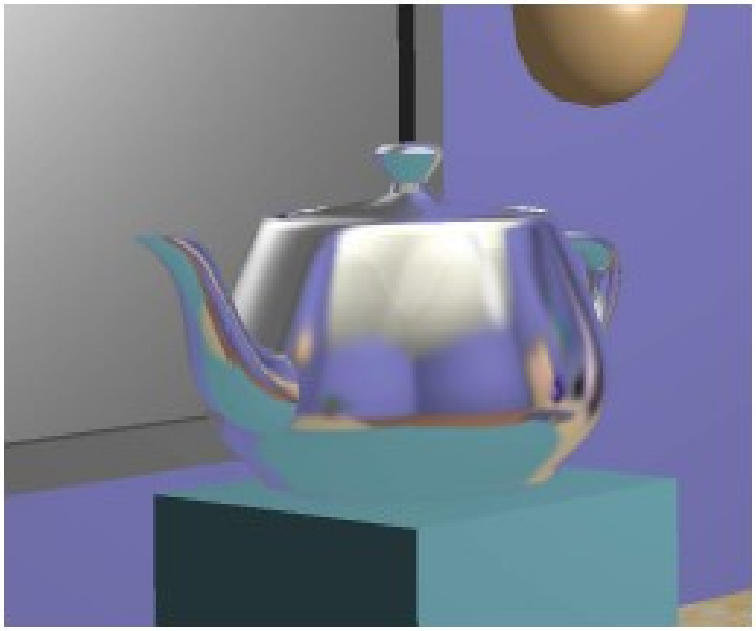
(5) Mappa sferica

(6) Immagine

- Si può creare un effetto lucido non perfettamente speculare sfocando (blurring) l'environment map.
- Accenniamo infine alla tecnica del chrome mapping, che prevede di usare una environment map fatta di chiazze di luce molto sfocate (che non c'entra nulla con la scena) per creare l'effetto di una superficie cromata.



(7) Blurred Envmap



(8) Immagine

Image-based lighting

- Si tratta di una estensione del reflection (environment) mapping.
- Serve ad illuminare in modo realistico un oggetto sintetico.
- Si memorizza in una immagine, chiamata light probe, il valore di illuminazione (radianza) lungo ogni direzione attorno ad un punto.
- Il light probe viene acquisito dal vero, tramite fotografie.
- Si usa poi questa nella soluzione della equazione della radianza. In particolare, si elimina la ricorsione nelle valutazione dell'integrale, poiché i valori di radianza lungo ogni direzione incidente sono predefiniti.

Image-based lightning

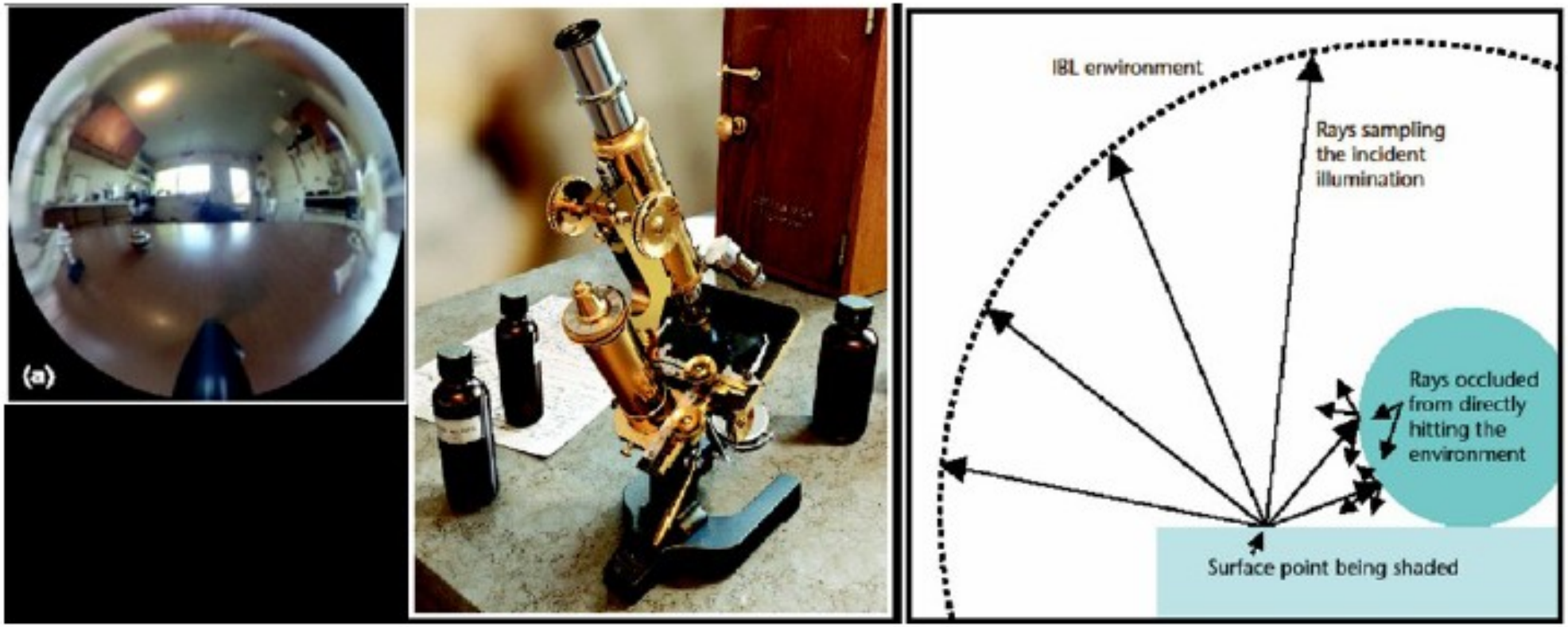
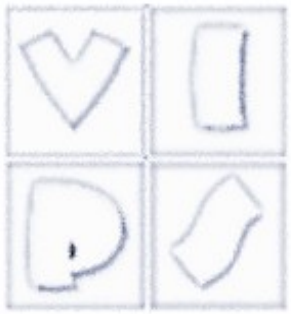


Figura 1: Light probe acquisito in una cucina, modello sintetico di un microscopio illuminato con il light probe, schema del IBL (da Debevec (2002))

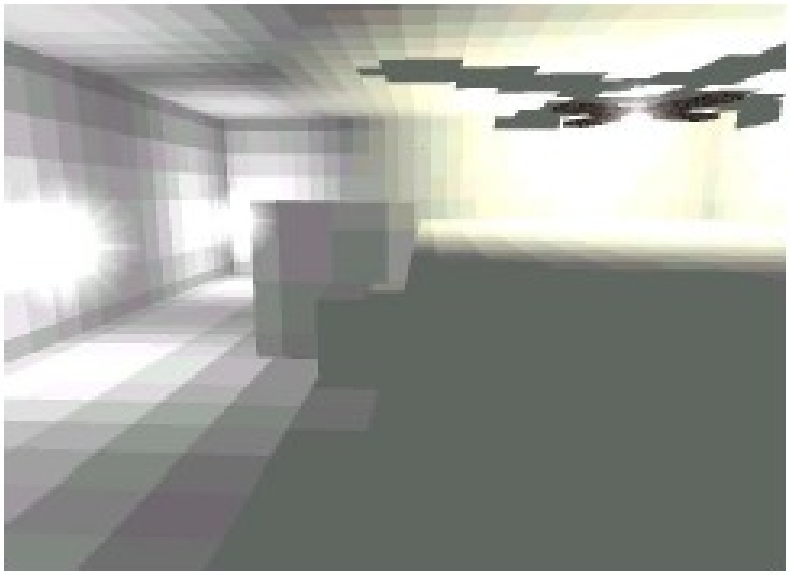
Light map

- Si tratta, essenzialmente, di calcolare, off-line l'illuminazione in ogni punto delle superfici che compongono la scena (senza texture).
- Si usa solitamente una soluzione view independent della equazione della radianza, come radiosity.
- I valori di illuminazione vengono salvati in una immagine chiamata light map (tipicamente a livelli di grigio).
- In fase di rendering (on line) si aggiunge la texture con modulazione, ovvero moltiplicando lightmap e texture.



Light map

- Il vantaggio è che questa tecnica produce uno shading di migliore qualità rispetto a Gouraud ed è più veloce.
- Però funziona solo per elementi statici; elementi in movimento vanno trattati con altri metodi e le due cose vanno integrate in modo opportuno.
- La light map può avere anche una risoluzione molto inferiore a quella dell'immagine, poi viene filtrata prima dell'applicazione.
- La tecnica è largamente usata (specialmente nell'industria dei videogiochi).

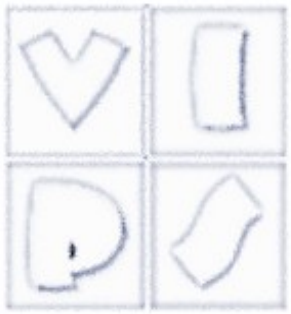


(1) Solo Light map

(2) Con Light map filtrata

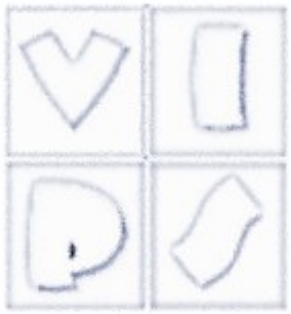
(3) Solo texture

(4) Finale



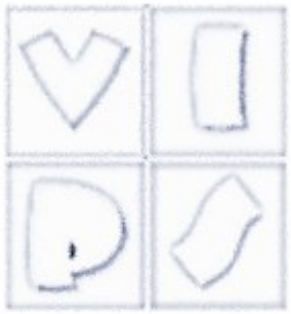
Impostori planari

- Il texture mapping viene usato anche per simulare un numero elevato di oggetti geometricamente complicati devono essere inclusi nella scena.
- Esempio: un terreno anche molto realistico, perde completamente di credibilità se non si vedono alberi (ad esempio). Ma un albero è un oggetto piuttosto complesso da un punto di vista geometrico
- Se il terreno è visto da molto lontano allora è sufficiente applicare la giusta texture al terreno per dare l'impressione che questo sia ricoperto da foreste



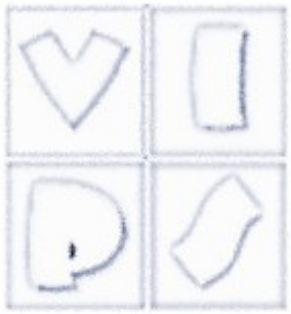
Impostori planari (sprite)

- Il problema è complesso in una situazione intermedia, in cui siamo abbastanza lontani dal terreno da poter abbracciare con lo sguardo una porzione rilevante di foresta (magari centinaia o migliaia di alberi), ma troppo vicini perché una semplice texture sul terreno basti (si vede che è piatto)
- Altro esempio: in una esplosione vi sono svariate decine di pezzi che volano da tutte le parti (tipicamente in fiamme)
- Se l'applicazione deve introdurre una loro descrizione geometrica, il carico di calcolo sarebbe proibitivo, sopratutto se vi è la possibilità di più esplosioni in sequenza



Impostori planari (sprite)

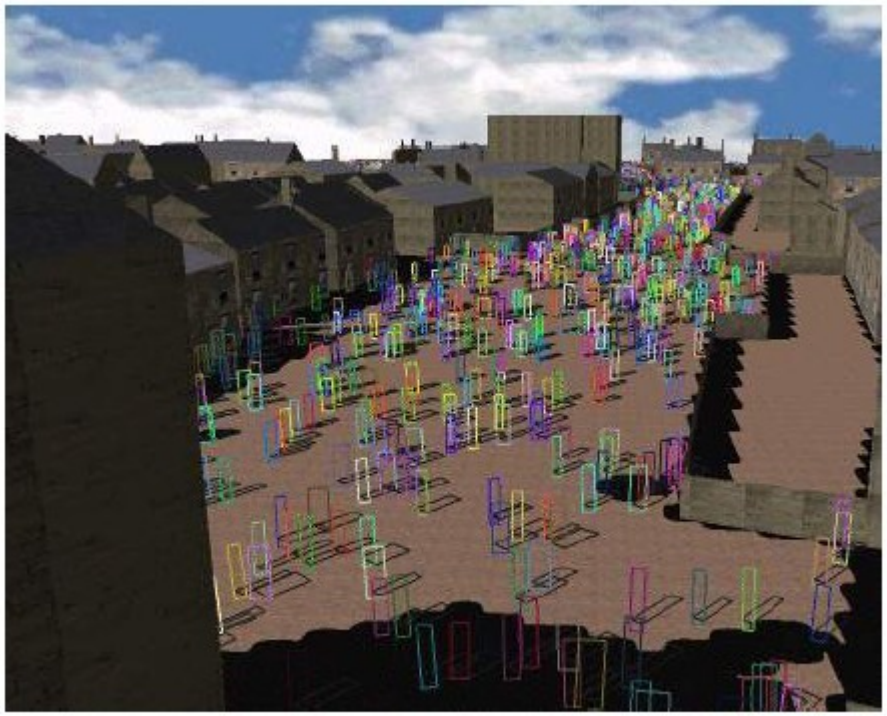
- Altro esempio: si supponga di voler fare il rendering di una folla di persone. Il problema è del tutto analogo a quello degli alberi discusso sopra, con la differenza che le persone si possono anche muovere. In queste situazioni si introducono i cosiddetti impostori planari o sprite
- Essenzialmente si tratta di rappresentare l'oggetto da inserire nella scena come una immagine
- Tale immagine viene usata come texture per un poligono (in genere un rettangolo)



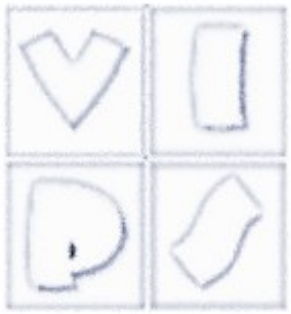
Impostori planari (sprite)

- In genere l'immagine ha un canale α di trasparenza tale che solo l'oggetto che si vuole rappresentare risulti visibile, ovvero non si vede il bordo della texture
- Se si posiziona tale poligono in modo opportuno, l'oggetto sembrerà inserito geometricamente nella scena, non semplicemente una immagine 2D
- Per ogni istanza basta aggiungere, da un punto di vista della geometria, un semplice poligono. Si possono così aggiungere centinaia di elementi senza sovraccaricare il motore di rendering grafico.
- Come va posizionato il poligono? Dipende dall'applicazione.

Esempio



© Slater et al.



Billboarding

- In genere, se la telecamera si può spostare liberamente nella scena, bisogna far si che il poligono non venga mai visto di taglio (altrimenti risulta palese la sua natura bidimensionale)
- La tecnica più nota va sotto il nome di billboarding
- Il poligono viene ruotato intorno ad un suo asse per far si che la faccia del poligono punti sempre in direzione della camera
- Se $\mathbf{n}$ è la normale al poligono e $\mathbf{v}$ è il vettore di vista, ovvero il vettore che specifica la direzione dalla camera al billboard (entrambi i vettori si suppongono normalizzati), allora bisognerà fare in modo che $\mathbf{n} \cdot \mathbf{v} = -1$

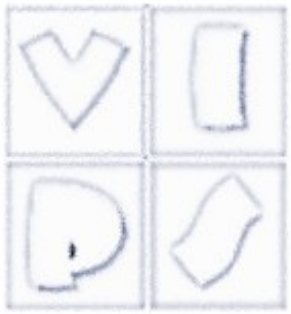
Billboarding

- Se l'impostore possiede, grosso modo, una simmetria sferica, allora sarà possibile ruotarlo per soddisfare la precedente condizione
- Se l'oggetto da simulare ha simmetrie più restrittive (ad es. cilindrica nel caso di un albero) allora la tecnica ha senso solo se la telecamera è vincolata a stare su un piano perpendicolare all'impostore, o comunque molto vicino a tale piano
- Inoltre muovendo la telecamera si può notare come l'oggetto mostri sempre la stessa faccia all'osservatore, il che può risultare strano. In presenza di molti impostori ed in scene viste molto velocemente comunque non vi si fa caso

Billboarding

- Una variante prevede di cambiare immagine a seconda dell'angolo di vista (tipico il caso di 8 immagini)
- L'esempio degli alberi è la più nota applicazione del billboard. Se la telecamera è vincolata a rimanere vicina al terreno e a puntare in direzione parallela a questo, allora gli alberi possono rappresentarsi efficacemente tramite billboard
- Una variante, meno efficace, è quella di costruire l'albero come due poligoni intersecantesi posti perpendicolari e proiettare l'immagine su entrambi i poligoni; in questo caso, per ottenere l'effetto 3D non c'è bisogno di ruotare l'impostore
- Un altro esempio tipico di billboard è la simulazione delle lens flare, ovvero le riflessioni interne tipiche di alcuni obbiettivi di telecamere o gli effetti di rifrazione (molto di moda, ma poco significativi)

Esempio



Billboarding

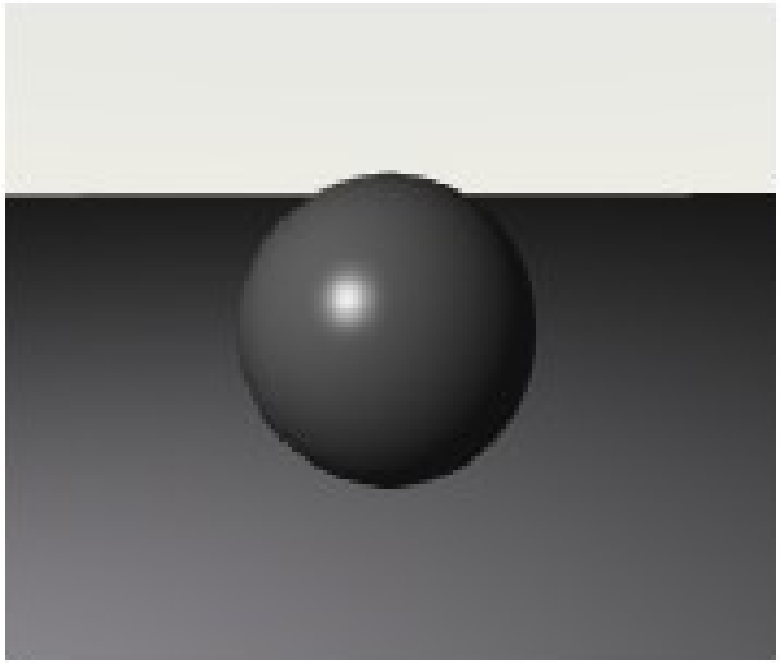
- Un altro uso tipico dei billboard è la simulazione di esplosioni e in generale, per effetti che coinvolgano numerose particelle
- Ciascuna particella viene rappresentata da un poligono con texture opportuna e viene spostata nella scena avendo sempre cura che la faccia del poligono sia rivolta verso l'osservatore.
- Si possono usare sistemi di particelle simili per simulare, ad esempio, effetti atmosferici (neve, pioggia, etc)

Ombre geometriche

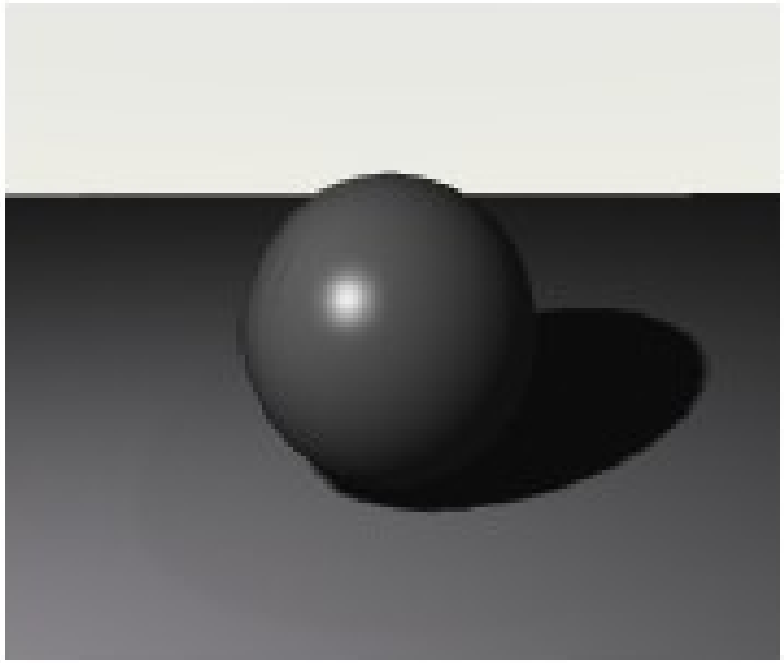
- Un particolare che si nota immediatamente nelle immagini ottenute con modelli di illuminazione locale, è l'assenza di ombre proiettate, che sono fondamentali per dare profondità e realismo all'immagine.
- Non confondere il chiaroscuro (o shading), che può essere ottenuto con algoritmi locali, con l'ombra proiettata (cast shadow), che non è locale di natura.
- Varie tecniche introducono le ombre in algoritmi locali
- Ne vediamo brevemente due
  - Queste generano ombre geometriche, perché calcolano la forma dell'ombra ma non l'illuminazione al suo interno

Ombre geometriche

- Differenza tra shading e ombra



(5) Solo chiarosuro

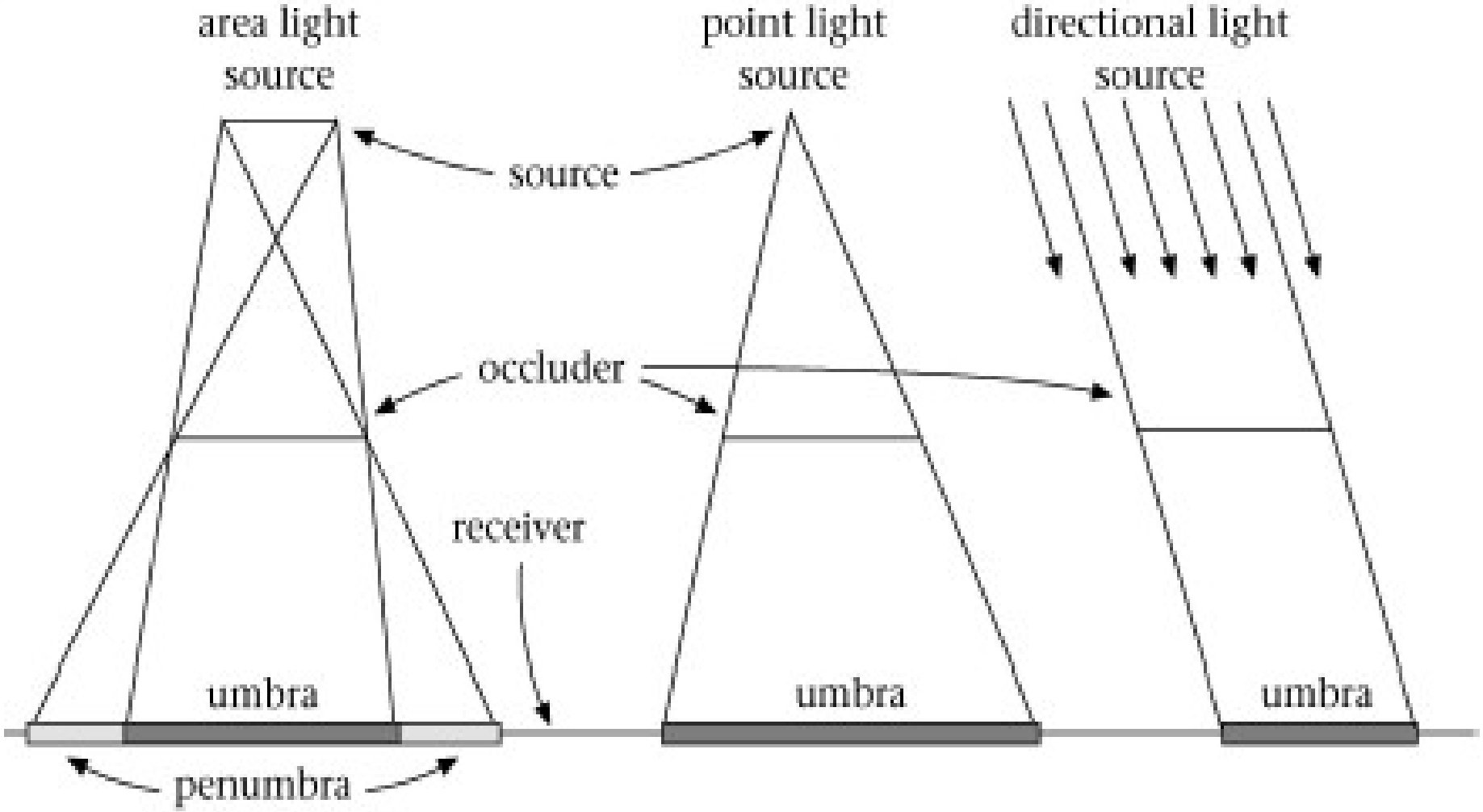


(6) Con ombra

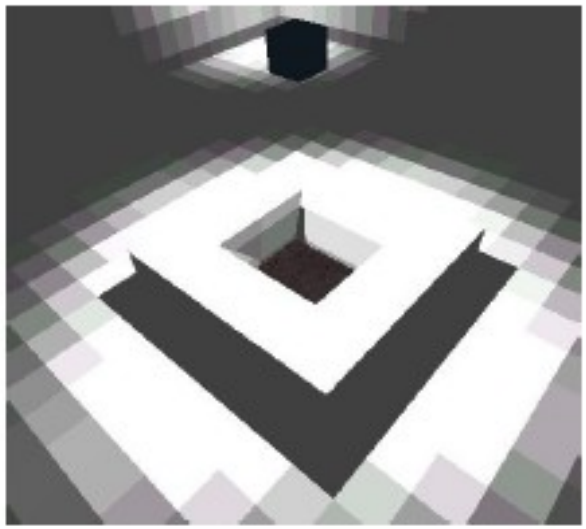
Proprietà delle ombre proiettate

- L'ombra che il poligono A getta sul poligono B a causa di una sorgente luminosa puntiforme si può calcolare proiettando il poligono A sul piano che contiene il poligono B con centro di proiezione fissato in coincidenza della sorgente.
- Non si vede alcuna ombra se la sorgente luminosa coincide con il punto di vista. Ovvero, le ombre sono zone nascoste alla luce. Questo implica che si possono usare tecniche (modificate) di rimozione di superfici nascoste per calcolare le ombre.
- Per scene statiche le ombre sono fisse. Non dipendono dalla posizione dell'osservatore
- Se la sorgente (o le sorgenti) è puntiforme l'ombra ha un bordo netto, ovvero non c'è penombra (come nello spazio).

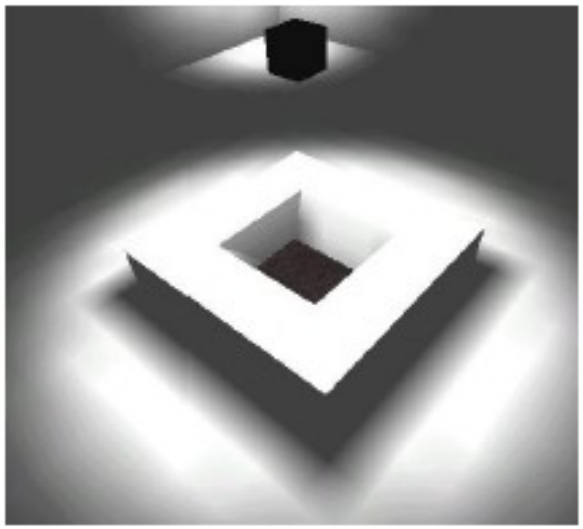
Proprietà delle ombre proiettate



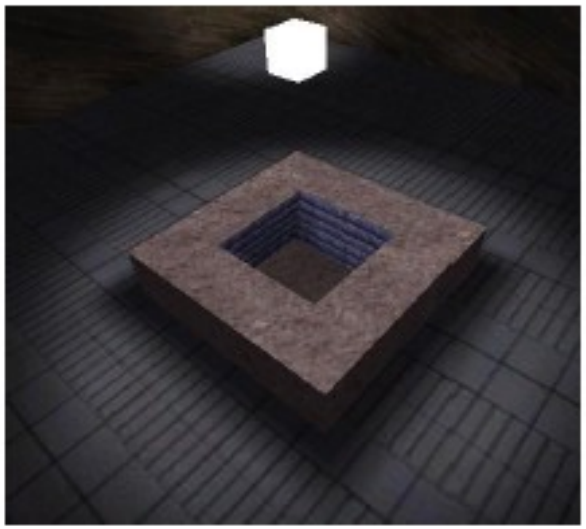
- Le ombre (geometriche) possono essere precalcolate per oggetti statici, ed aggiunte alla lightmap (molto usato nei videogiochi).



(7) Lightmap + geomeric shadows



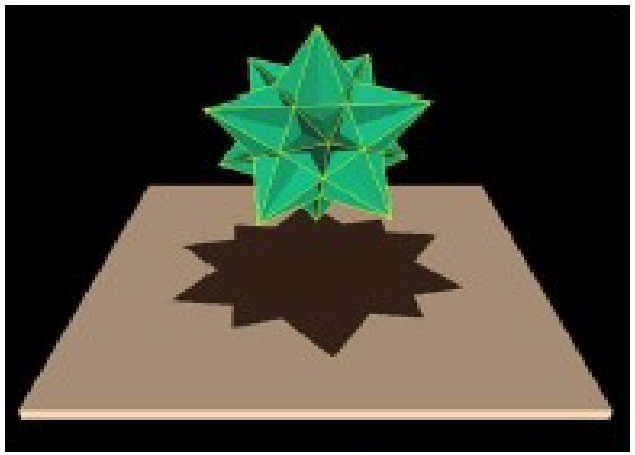
(8) Filtrata



(9) Applicata all'oggetto con texture

Ombra sul piano (fake shadows)

- Si tratta di una tecnica per calcolare l'ombra portata da un oggetto su un piano π (di solito il terreno) a causa di una luce posizionata in $\mathbf{P}_l$.
- L'idea è di disegnare l'ombra come un oggetto piatto –(tipicamente nero) sul piano.
- Per disegnare l'ombra si effettua il rendering della scena, con colore nero, con una telecamera centrata in $\mathbf{P}_l$ e che ha il piano π come piano immagine.
- Se la luce è direzionale (distant light) si una una matrice di proiezione ortogonale
- La tecnica è semplice ed efficiente: si tratta semplicemente di fare il rendering della scena due volte.



© Bert Schoenwaelder

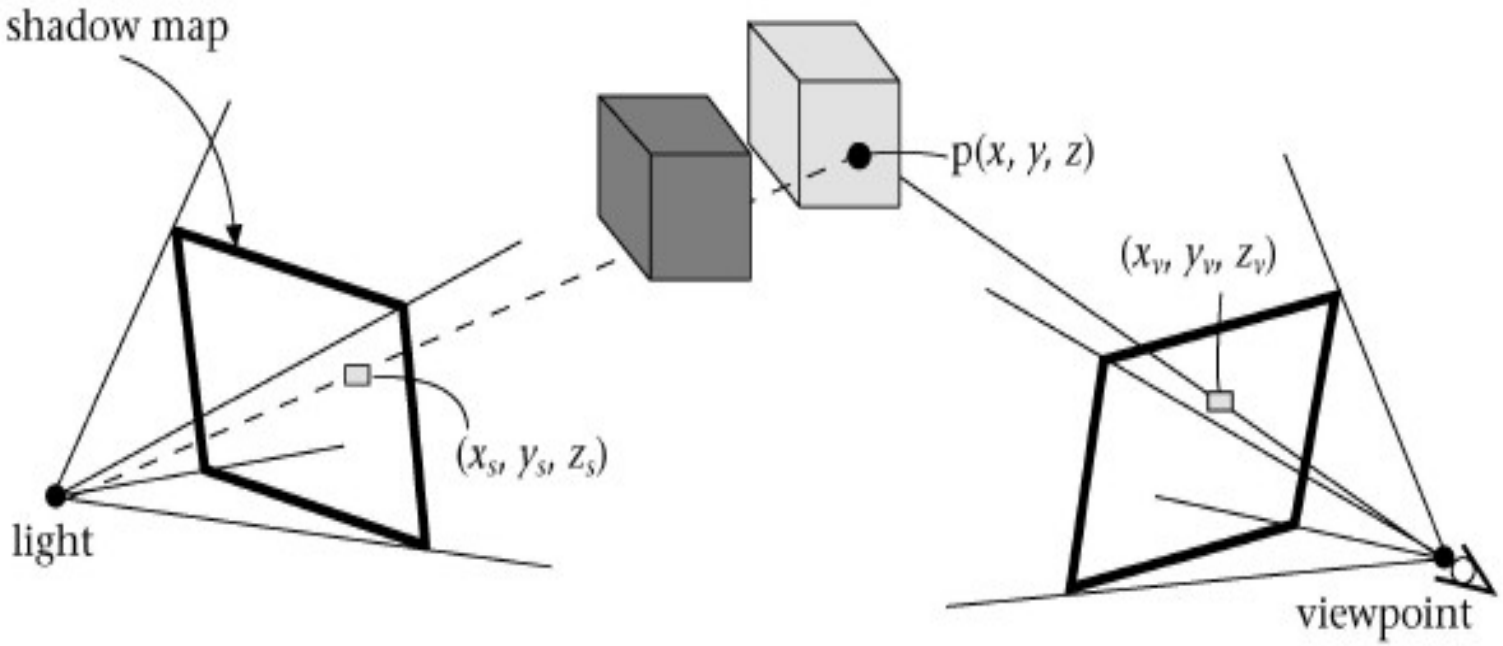
Ombra sul piano (fake shadows)

- Il metodo gestisce oggetti arbitrariamente complicati, ma la scena è vincolata ad essere molto semplice: un solo oggetto oppure più oggetti sufficientemente distanti da non farsi ombra.
- Calcolare l'ombra di un oggetto su un altro oggetto con questo metodo è fattibile, ma è più complicato
- I triangoli d'ombra giacciono sullo stesso piano del piano π visto sopra; ci possono essere problemi di risoluzione dello z-buffer per cui nel rendering alcuni di questi triangoli (o una loro parte) possono finire sotto il piano (z-fighting) .

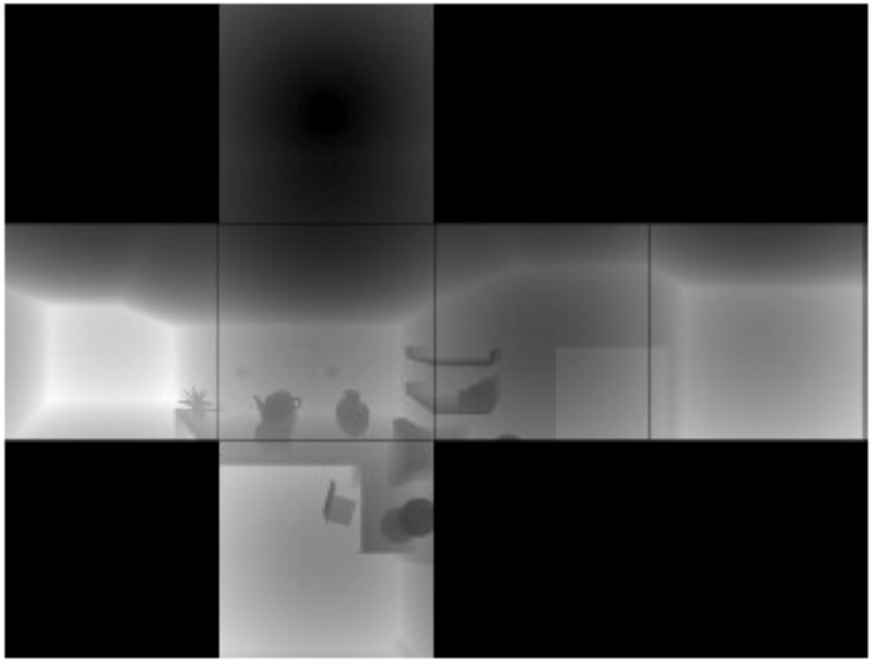
Shadow buffer

- Questa tecnica si rifà alla seconda osservazione sulle proprietà delle ombre, ovvero si basa su un algoritmo di rimozione delle superfici nascoste.
- Si calcola uno z-buffer dal punto di vista della luce (detto anche shadow buffer o shadow map):
  - Se la luce è spot-light allora basta calcolare uno z-buffer singolo
  - Se la luce è una point-light, la si immagina racchiusa in un cubo e si calcolano sei z-buffer, uno per ogni faccia del cubo.
- In fase di rendering, se un punto di un poligono deve essere disegnato, lo si trasforma nel sistema di riferimento della luce e si trova il valore z_l nello shadow buffer

- Se lo z del punto (nel riferimento della luce, non della camera) è più grande di z_l, significa che vi è un oggetto che blocca la luce per quel punto, quindi è in ombra e lo si può colorare di conseguenza. Altrimenti è colpito dalla luce e si opera lo shading normalmente
- Se si hanno più luci bisognerà avere uno shadow buffer per ognuna.
- La tecnica eredita le inefficienze dello z-buffer: richiede molta memoria e compie calcoli che poi possono venire sovrascritti.



Shadow buffer



(10) Shadow buffers

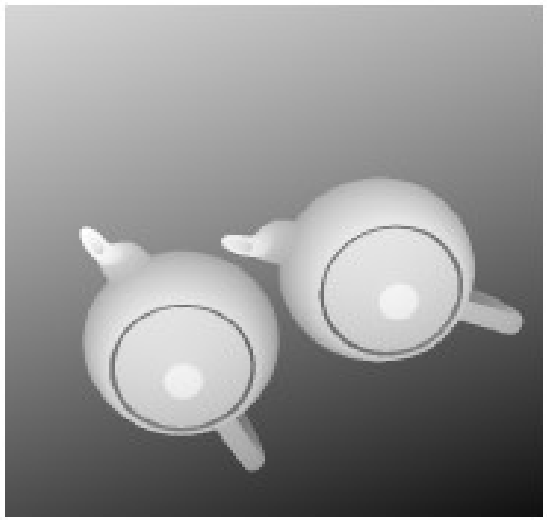
(11) Image

Esempio: environment map e shadow buffer

(12) Scena prima

(13) Scena dopo

(14) Envmap

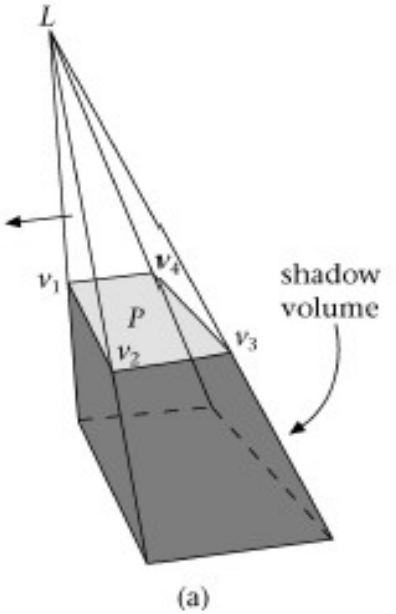


(15) Shadow buffer

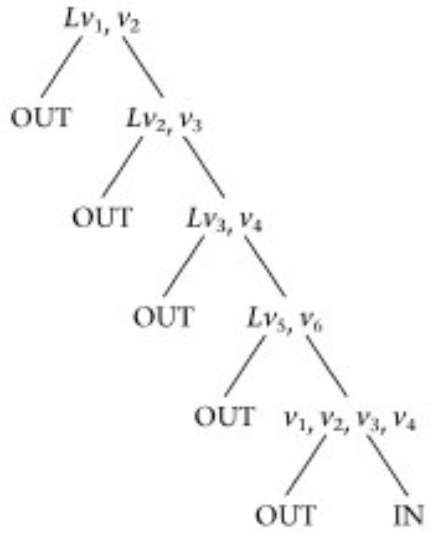
Shadow volume

- Metodo introdotto da Crow (1977) e Bergeron (1986).
- Il volume d'ombra (shadow volume) di un poligono rispetto ad una sorgente luminosa puntiforme è la parte di spazio che il poligono occlude alla vista della luce

- E' un tronco di piramide semi-infinita (senza base) che ha il vertice nella sorgente luminosa ed è limitata da una parte dal poligono stesso.
- Le facce della piramide prendono il nome di shadow planes. Vengono rappresentate in modo che il "fronte" sia verso la luce ed il "retro" verso l'ombra.

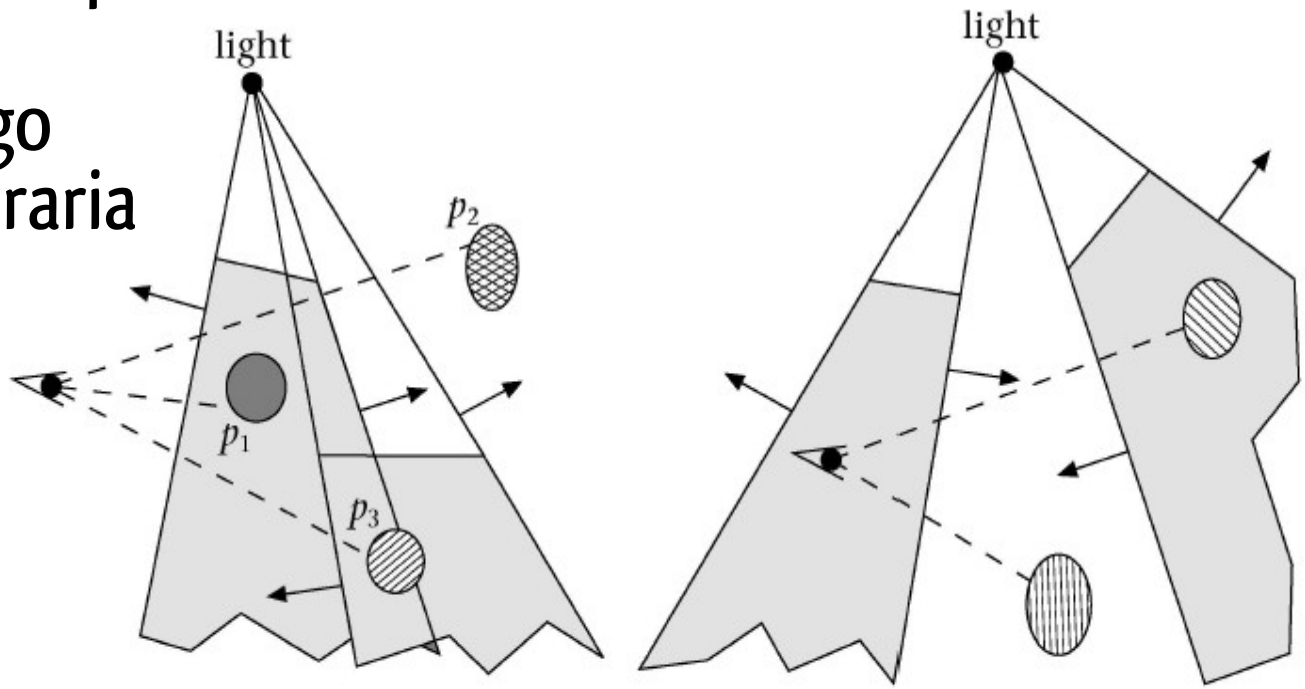


(a)



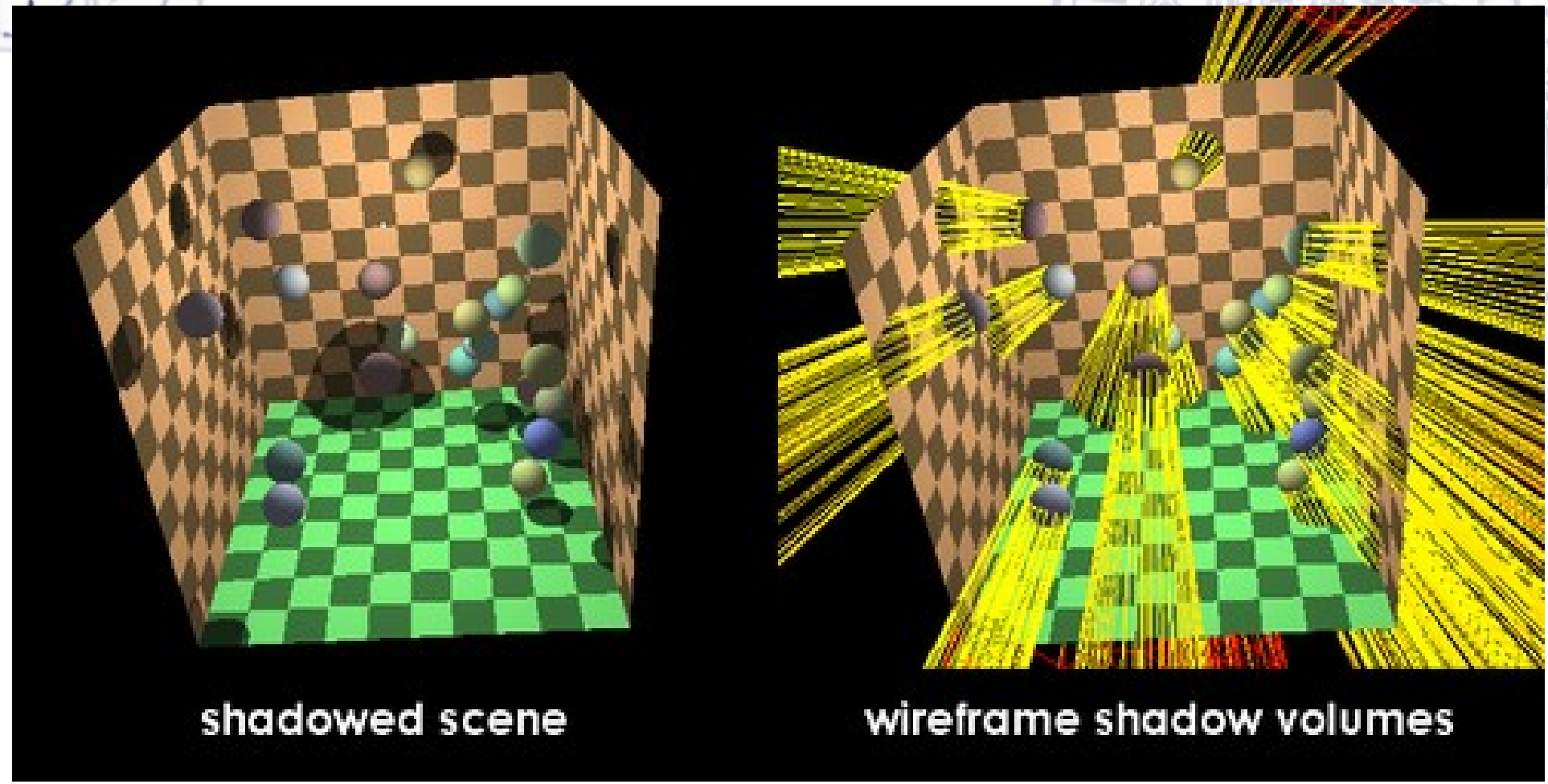
(b)

- Dato un punto di vista non in ombra, e dato un punto P della scena, tracciamo il segmento di retta che congiunge il punto di vista e P e contiamo gli attraversamenti degli shadow planes, incrementando un contatore quando l'attraversamento è in entrata (dal fronte) e decrementandolo quando è in uscita (dal retro). Se la differenza è zero il punto è nella luce, altrimenti è nell'ombra (di uno o più poligoni).
- Se il punto di vista è nell'ombra bisogna partire dal numero di shadow volumes nei quali è contenuto. Si calcola contando le intersezioni con gli shadow planes lungo una semiretta arbitraria con origine nel punto di vista.

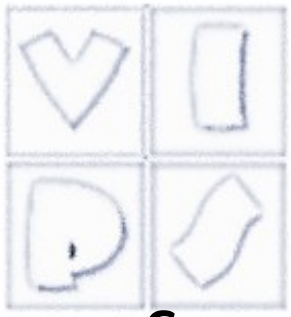


Shadow volume

- In pratica: la generazione degli shadow planes avviene off-line, vengono aggiunti alla scena come poligoni (vengono tagliati ad una certa distanza dalla luce) e vengono processati come tutti gli altri poligoni, eccetto che sono invisibili. Durante la scan conversion (con un algoritmo scan-line), quando viene incontrato un tale poligono invece che colorare il pixel corrispondente, si incrementa/decrementa un contatore per pixel. Il risultato è una mappa che per ogni pixel dice se è in luce (= 0) oppure in ombra (> 0)
- In OpenGL si fa in tre passate, con un trucco che usa z-buffer e stencil buffer.
- Varianti:
  - calcolare le silhouette delgli oggetti ed usarle per gli shadow volume (invece dei singoli poligoni).
  - creare gli shadow volumes solo degli oggetti “importanti”.



From wikipedia



Riferimenti

- Scateni et al. Cap 6.4-6.5
- Angel cap. 7.4

Domande di verifica

- Perché il texture mapping si fa tipicamente con mappatura inversa?
- Che differenza c'è tra gli artefatti nella mappatura delle tessiture che possono verificarsi nel caso si usi la proiezione prospettica o quella ortografica per creare la scena?
- Si indichino differenti applicazioni del texture mapping
- Qual è la differenza tra bump mapping e displacement mapping?