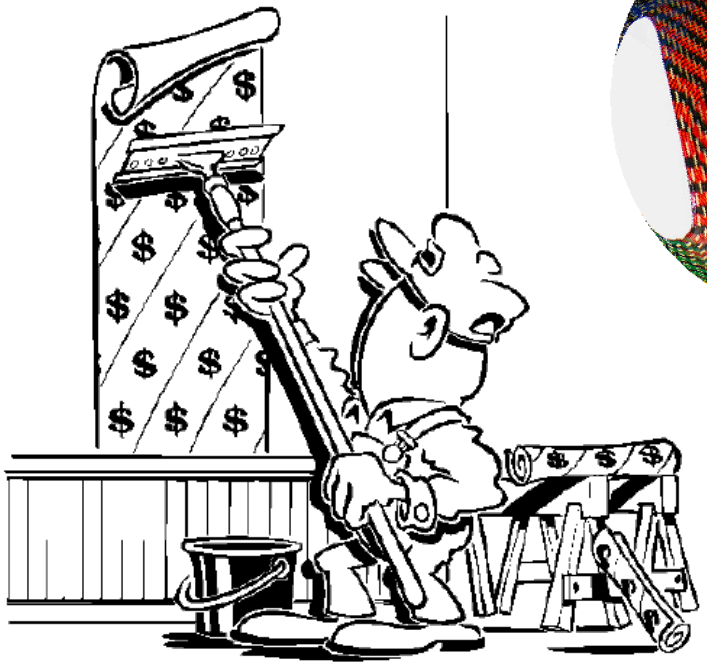
Texture nella Computer Graphics



Texture Mapping

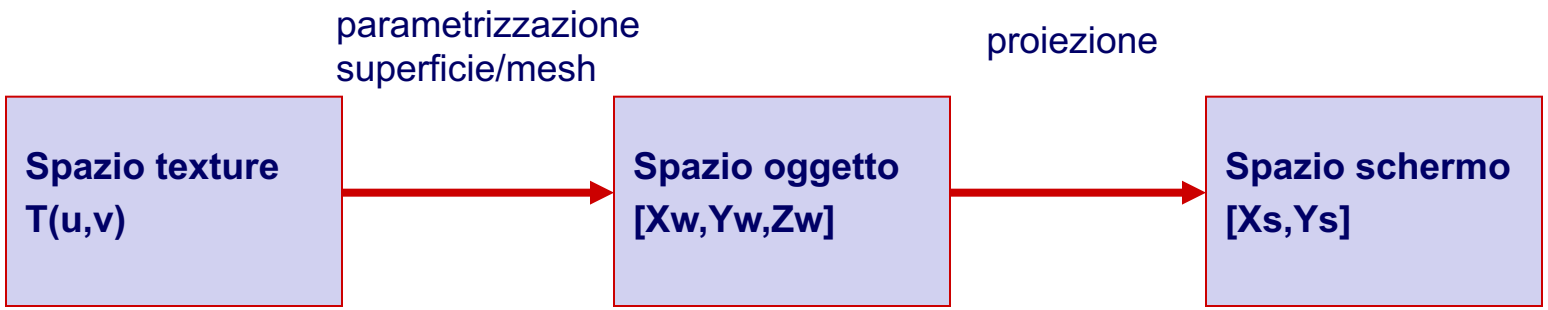
Con Texture ci si riferisce ad un'immagine di qualsiasi tipo utilizzata per "colorare" la superficie di un oggetto o modello 3D; la superficie del modello apparirà molto più complessa di quello che è, ma soprattutto più realistica.

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

Texture Mapping 2D

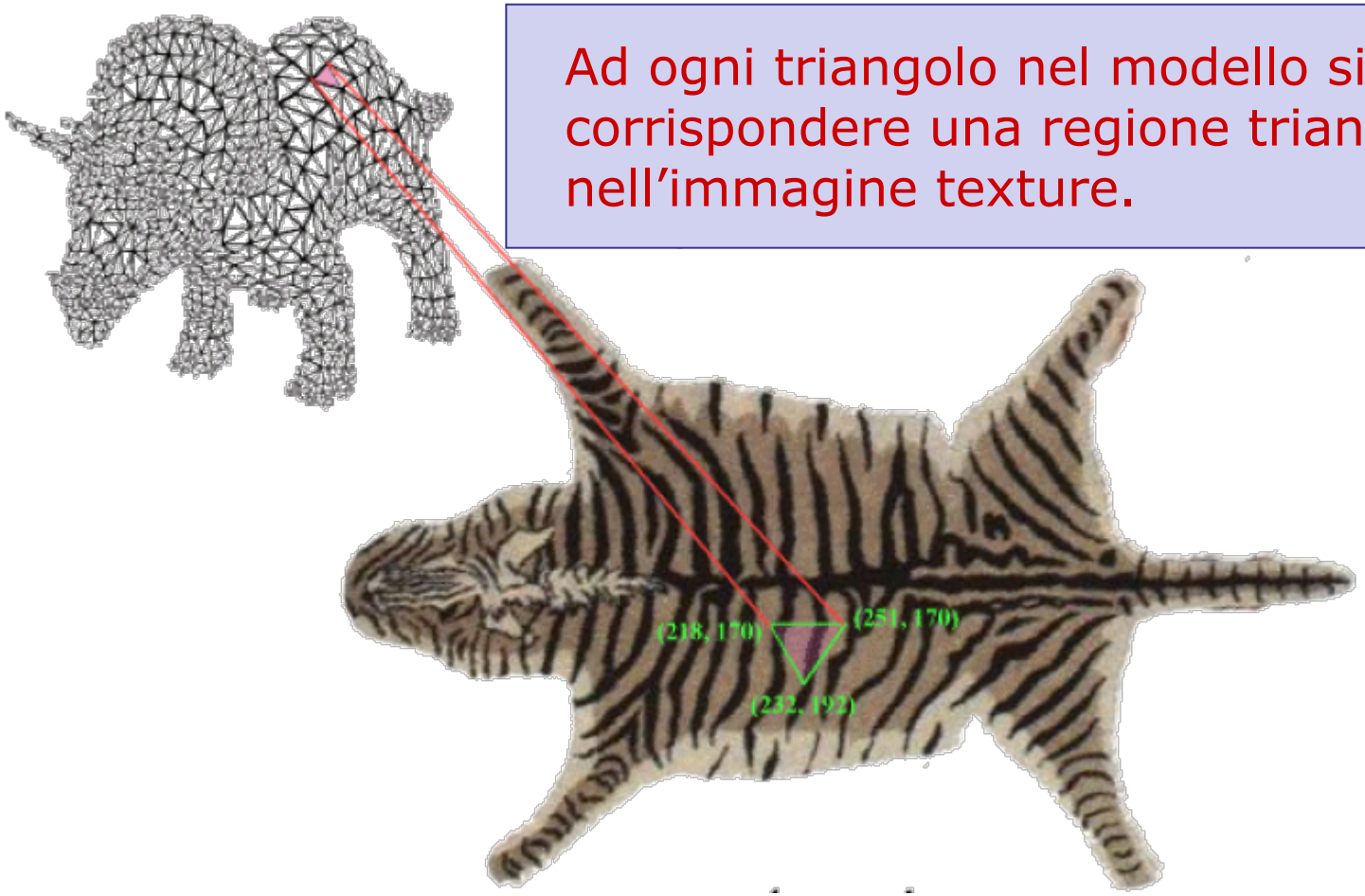
Come si deve procedere ?

Una texture viene definita in un dominio (o spazio texture), mentre l'oggetto è nello spazio oggetto; si deve definire una trasformazione fra questi: parametrizzazione

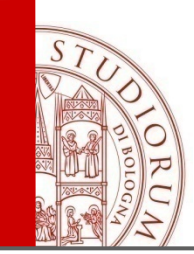


Texture Mapping 2D

Ad ogni triangolo nel modello si fa corrispondere una regione triangolare nell'immagine texture.



Esempio di mapping fra spazio texure e mesh 3D. Problema difficile, sia che sia fatto a mano sia in modo automatico



Texture *Mapping* 2D

Texture Mapping 2D in un contesto Z-Buffer.

Lo Z-Buffer implica un processo pixel per pixel per ogni faccia/triangolo; questo significa che si deve individuare un singolo valore texture (texel) per ogni pixel, per essere inserito nello schema shading.

Per fare questo si fa uso di un Inverse Mapping; si trova la preimmagine del pixel corrente nel dominio texture.

Il texture mapping richiede un trattamento speciale di Antialiasing poiché tende a produrre effetti di Aliasing.

Texture Mapping 2D: esempio

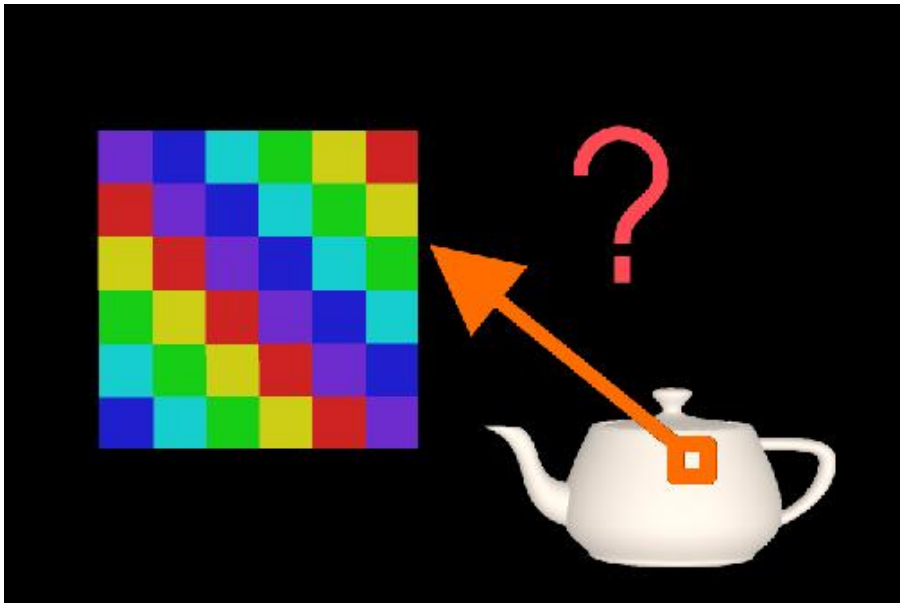
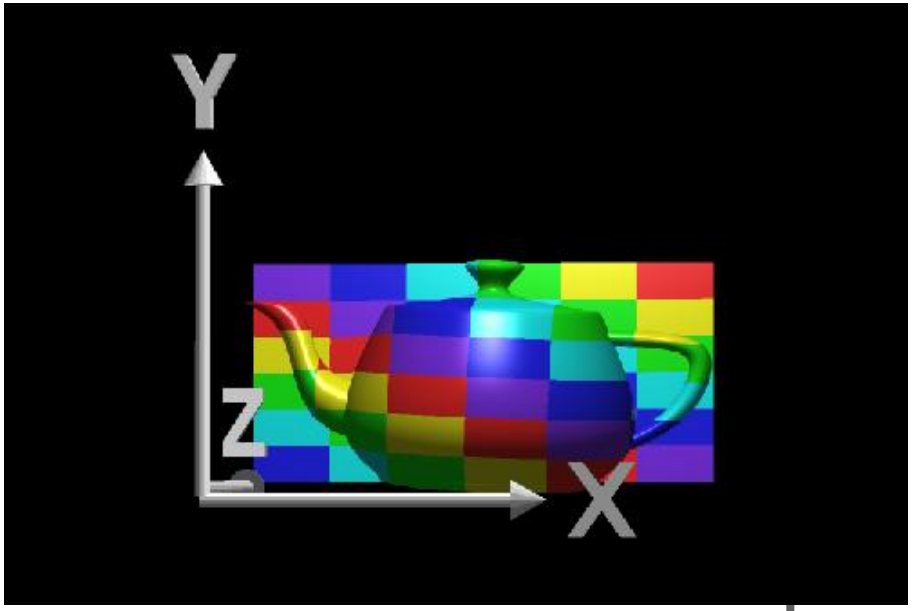


Image Mapping

Parametrizzazione
[u,v]=[x,y]→[x,y,z]



Texture Mapping 2D: esempio

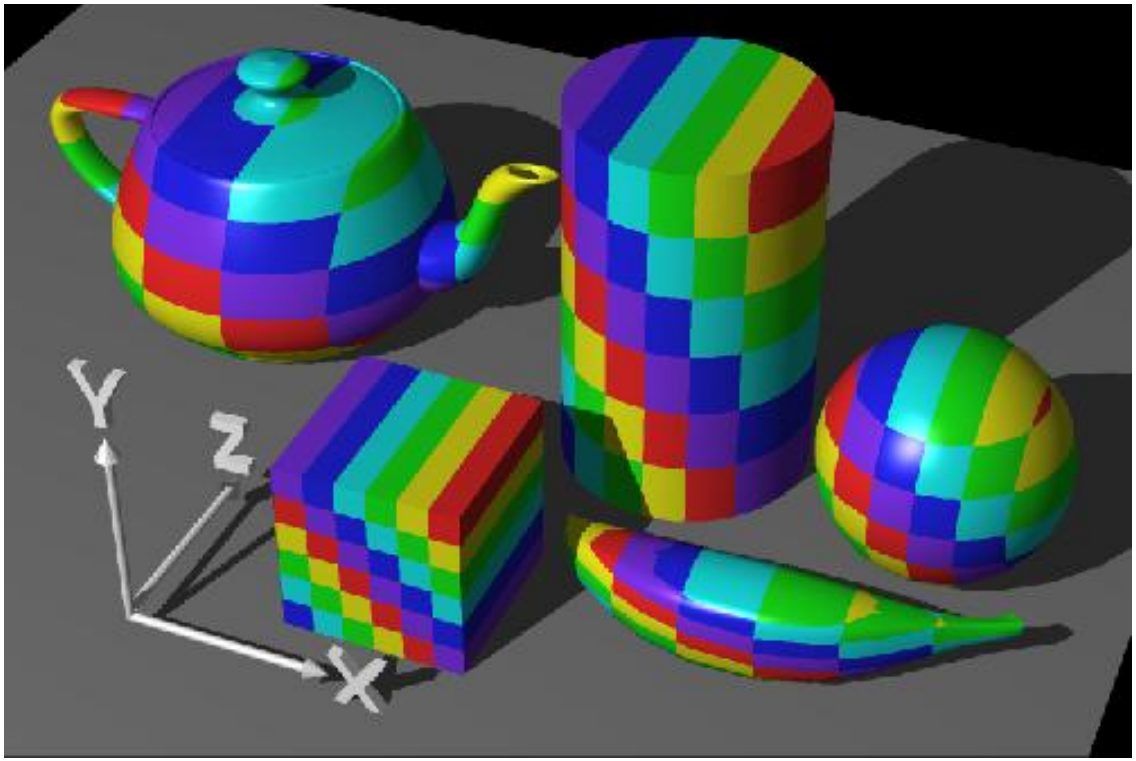
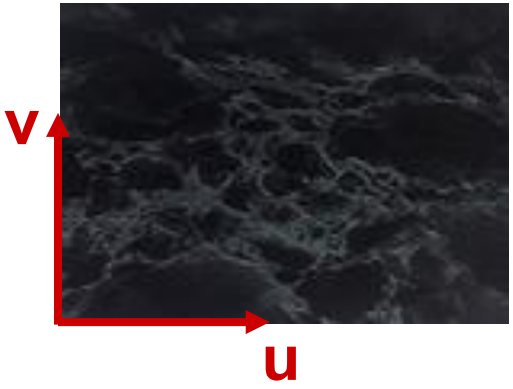


Image Mapping secondo parametrizzazione
[u,v]=[x,y]→[x,y,z]

Texture Mapping 2D: esempio

parametrizzazione [u,v]=[x,y]→[x,y,z]

Texture Mapping 2D: esempio



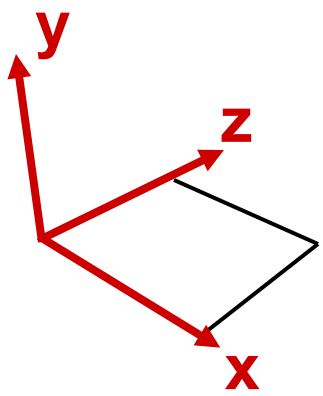
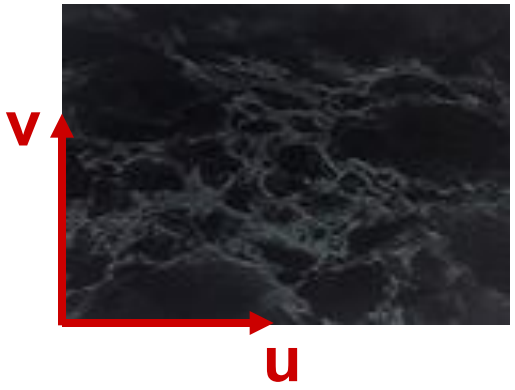
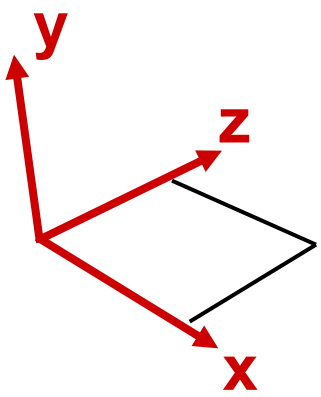
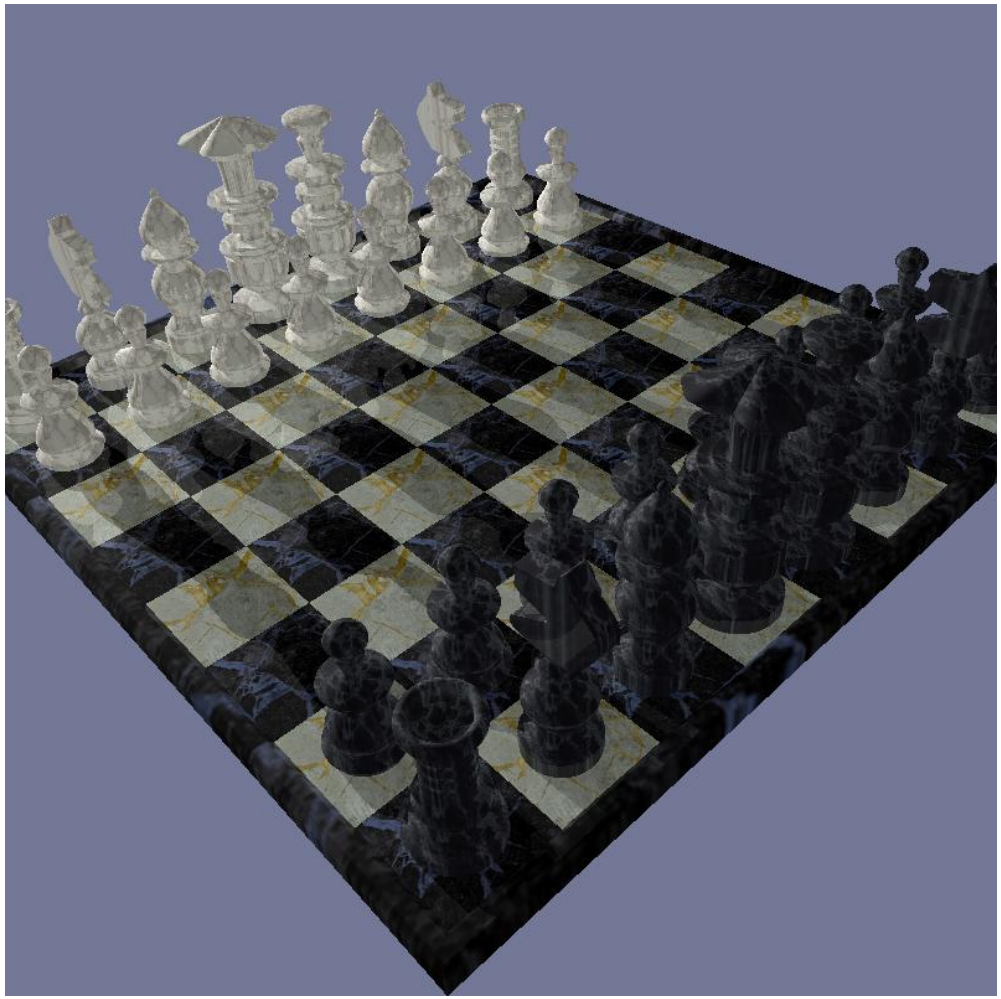


Image Mapping,
parametrizzazione [u,v]=[x,z]→[x,y,z];
lo spazio texture è stato posizionato alla base dell'oggetto

Texture Mapping 2D: esempio





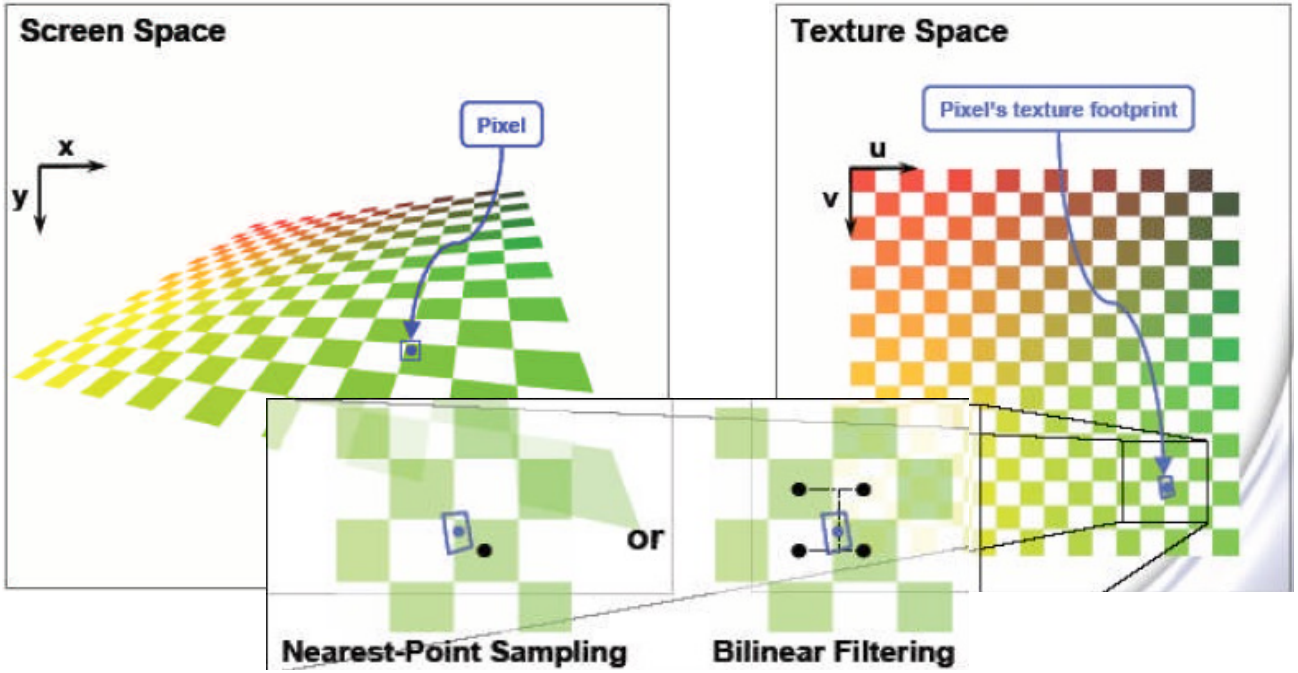


Texture Mapping 2D: esempio

Magnification e Minification

Pixel → Inverse Mapping → Texel

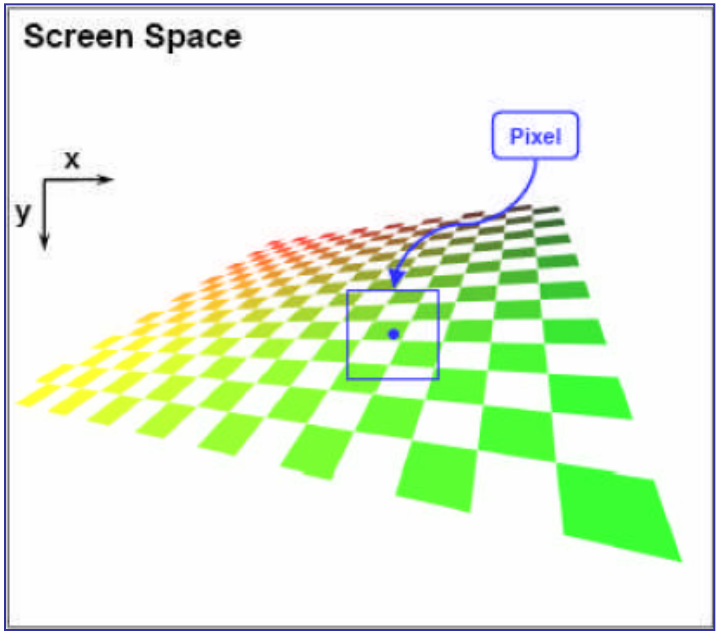
Cosa accade se un texel è più grande di un pixel?
Minification: la texture deve essere rimpicciolita.

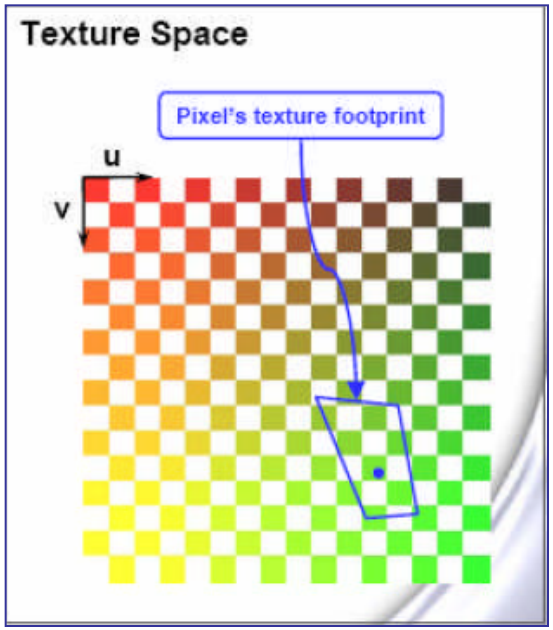


Magnification e Minification

Pixel → Inverse Mapping → Texel

Cosa accade se un texel è più piccolo di un pixel?
Magnification: la texture deve essere ingrandita.





Magnification e Minification

Tecniche di Antialiasing (o Filtering) più usate:

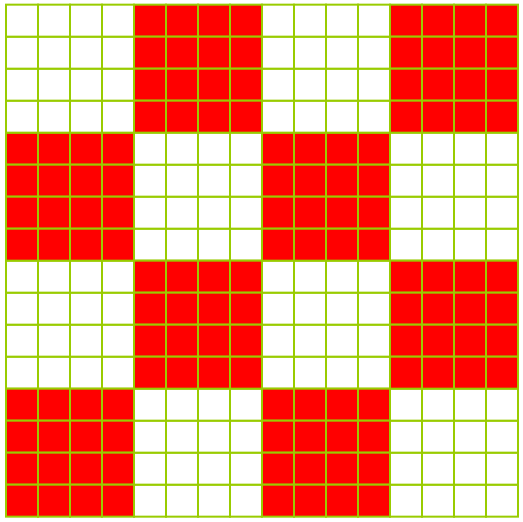
Nearest Neighbor: ad ogni pixel viene dato il colore del texel più vicino; più pixel avranno stesso colore

Effetto Pixelation

Effetto Blur

Bilinear Interpolation: per ogni pixel vengono presi i 4 texel più vicini e interpolati linearmente nelle due dimensioni per avere un valore medio per il pixel

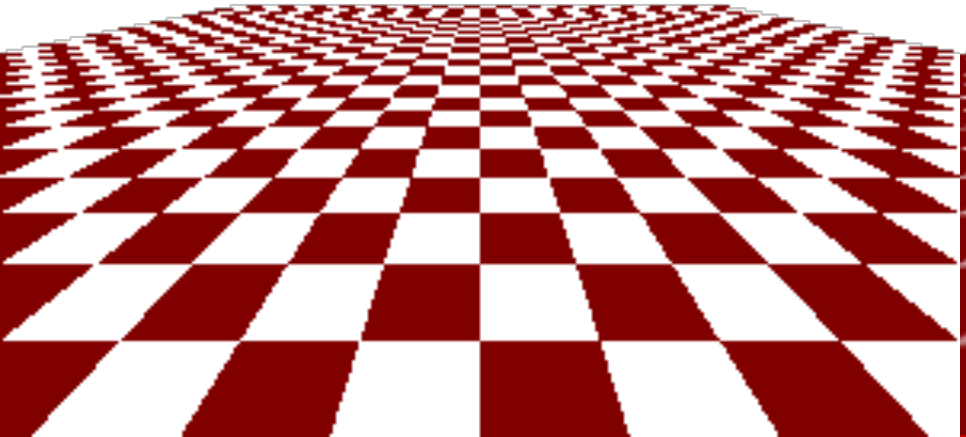
Esempio di Minification

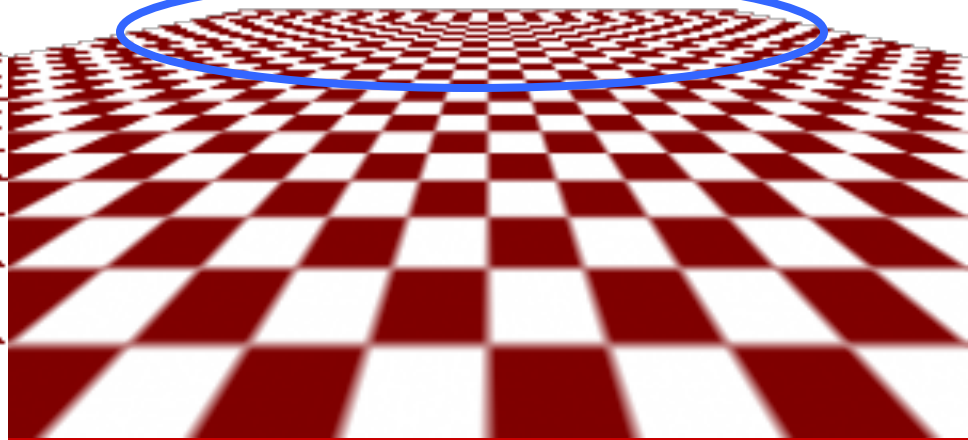


Nearest Neighbor

Bilinear Interpolation

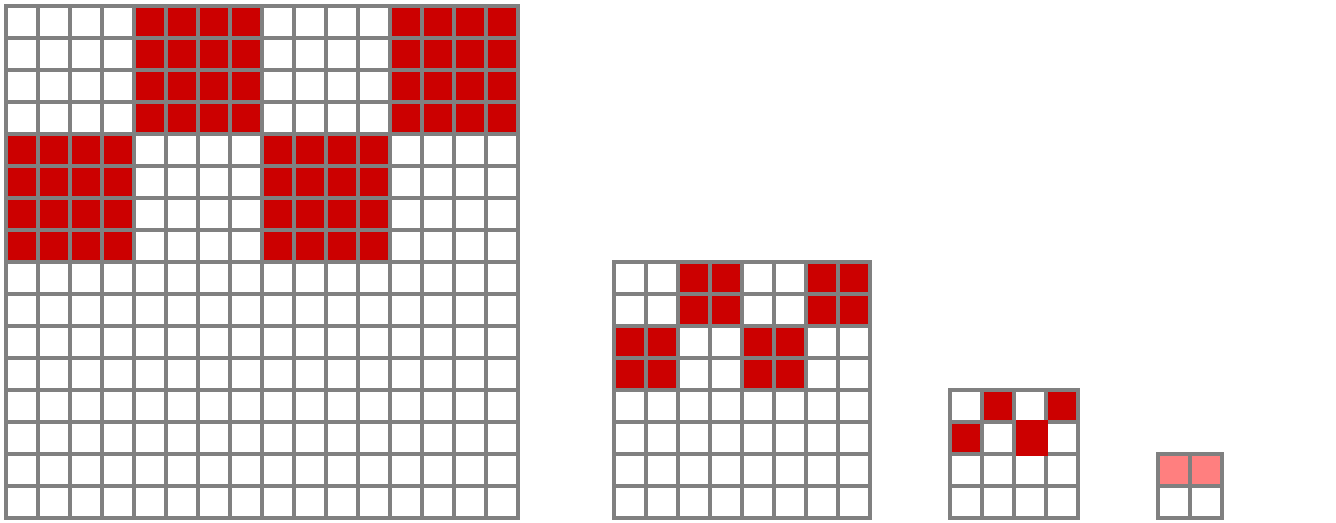
non risolve il problema





MIP Mapping

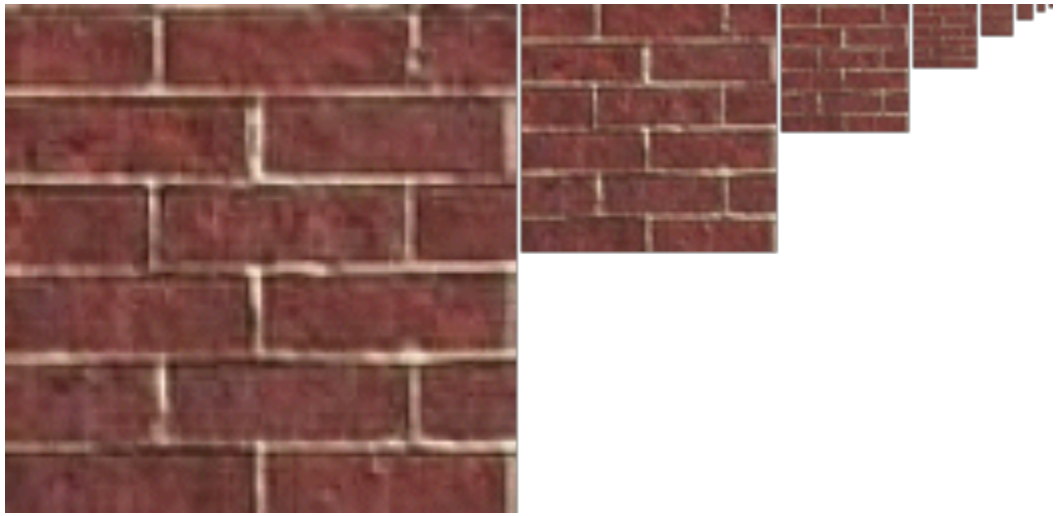
Multum In Parvo: molte cose in poco spazio



Texture originale *versioni a bassa risoluzione*

Per una texture di $2^n \times 2^n$ texel, si calcolano $n-1$ texture, ciascuna a metà risoluzione della precedente.

MIP Mapping

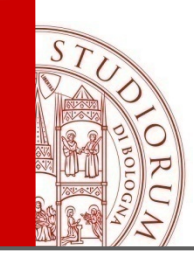


Liv.0 *Liv.1* *Liv.2 …*

Come si genera un MIP-map da una texture?

Per ogni texel al Livello i, si fa la media dei valori dei 4 corrispondenti texel al livello i-1.

Ogni livello del MIP-map rappresenta una versione pre-blurred di texel multiple

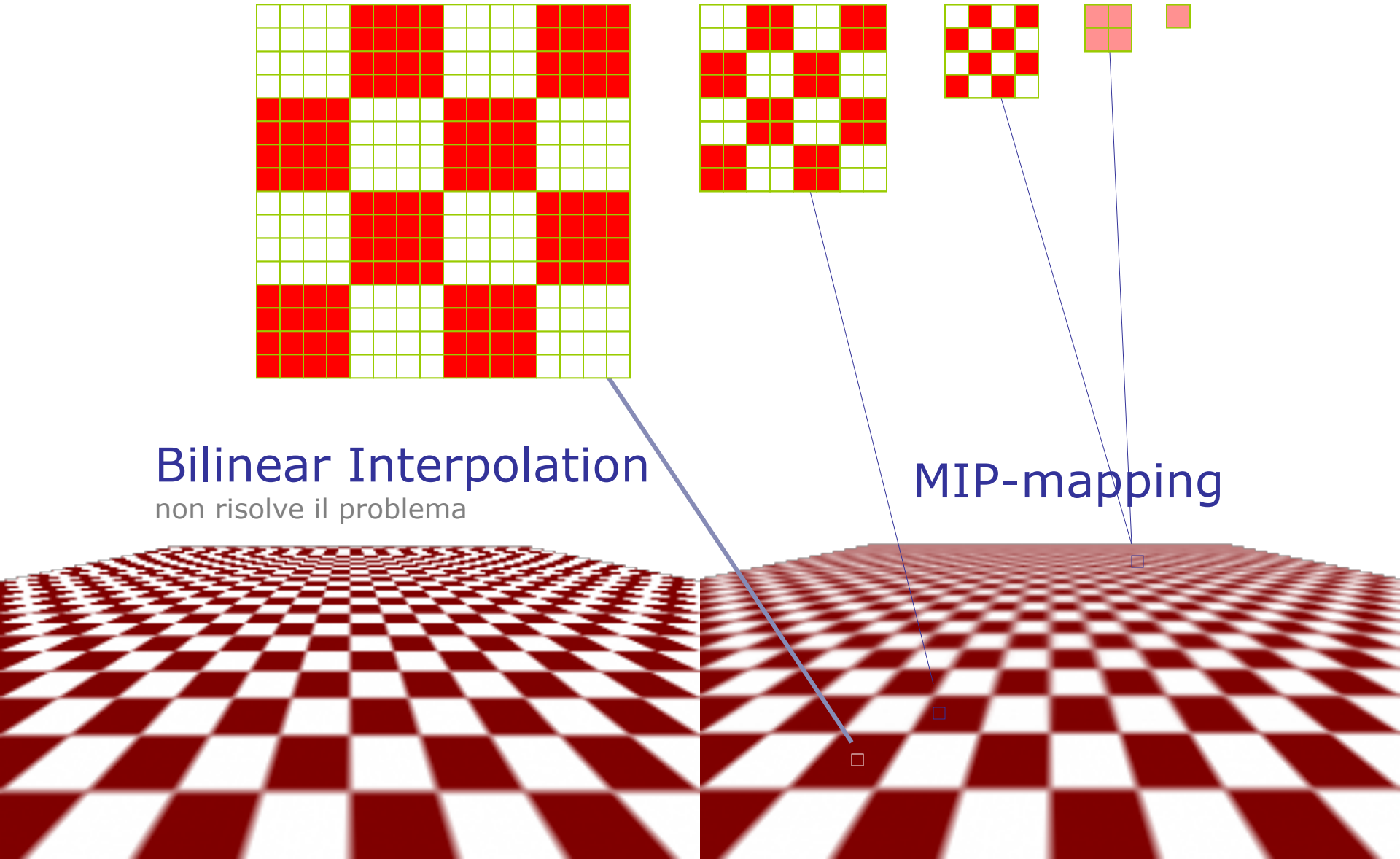


Mip Mapping

Quando e come si fa il rendering?

- Immaginiamo la texture ricoperta di pixel (cioè la dimensione dei pixel in texel nella texture originale)
- Si determini il livello del MIP map in cui i texel sono approssimativamente in corrispondenza 1 ad 1 con i pixel
- Si interpolino i valori dei 4 texel più vicini

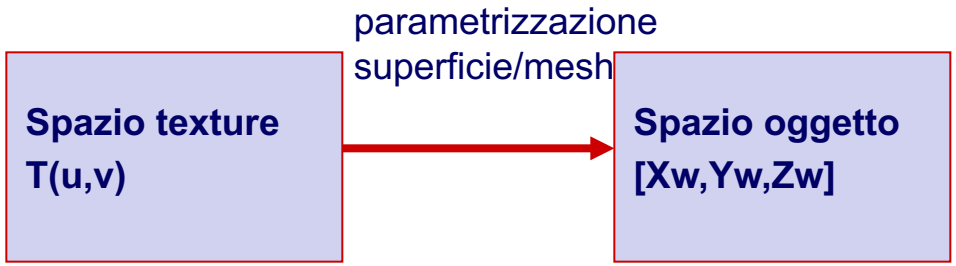
MIP Mapping: esempio



Texture Mapping 2D

Two-Part Mapping

1. **S-mapping**
2. **O-mapping**



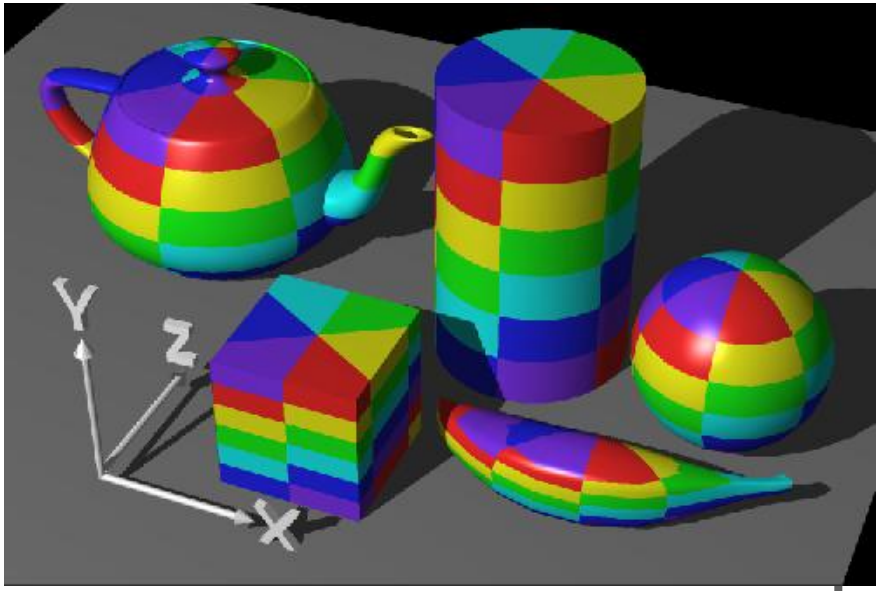
S-mapping **O-mapping**

Texture Mapping 2D

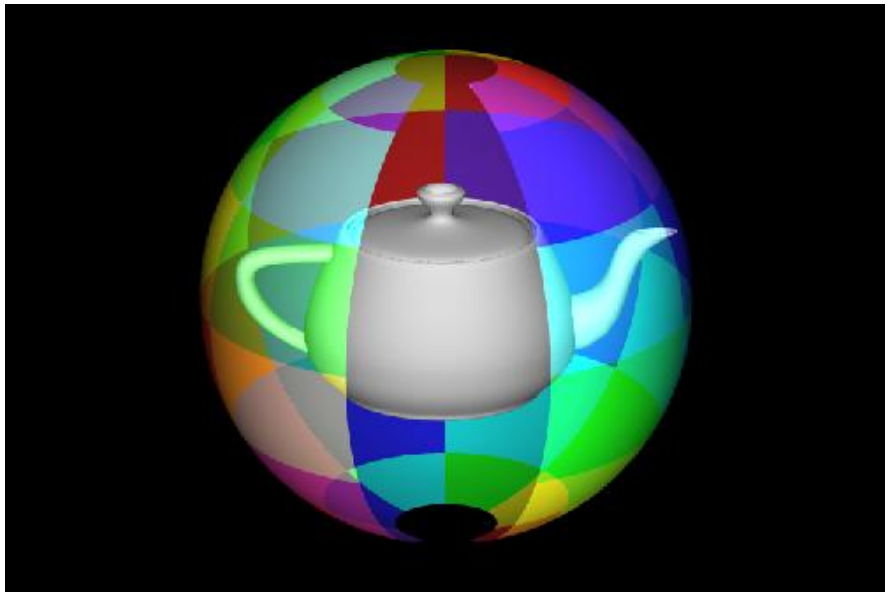
Two-Part Mapping

1. S-mapping
2. O-mapping

Superficie S intermedia:
Cilindro

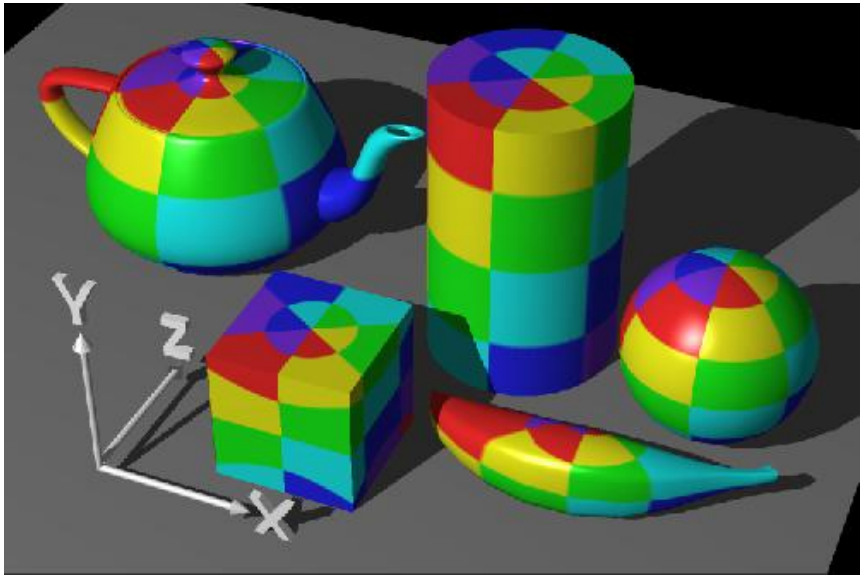


Texture Mapping 2D



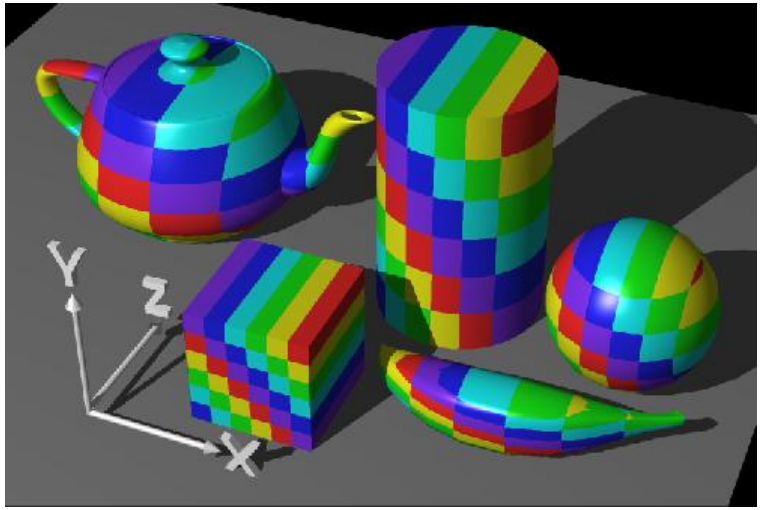
Two Part Mapping

1. S-mapping
2. O-mapping

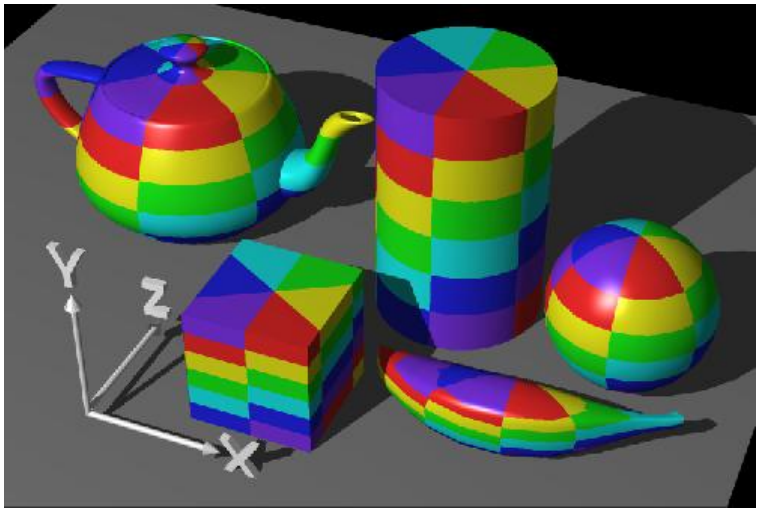


Superficie S intermedia:
Sfera

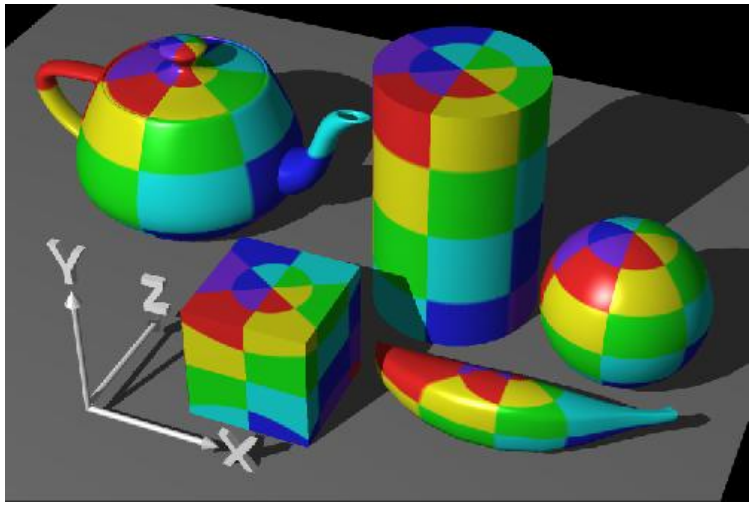
Texture Mapping 2D: confronto



Piano XY



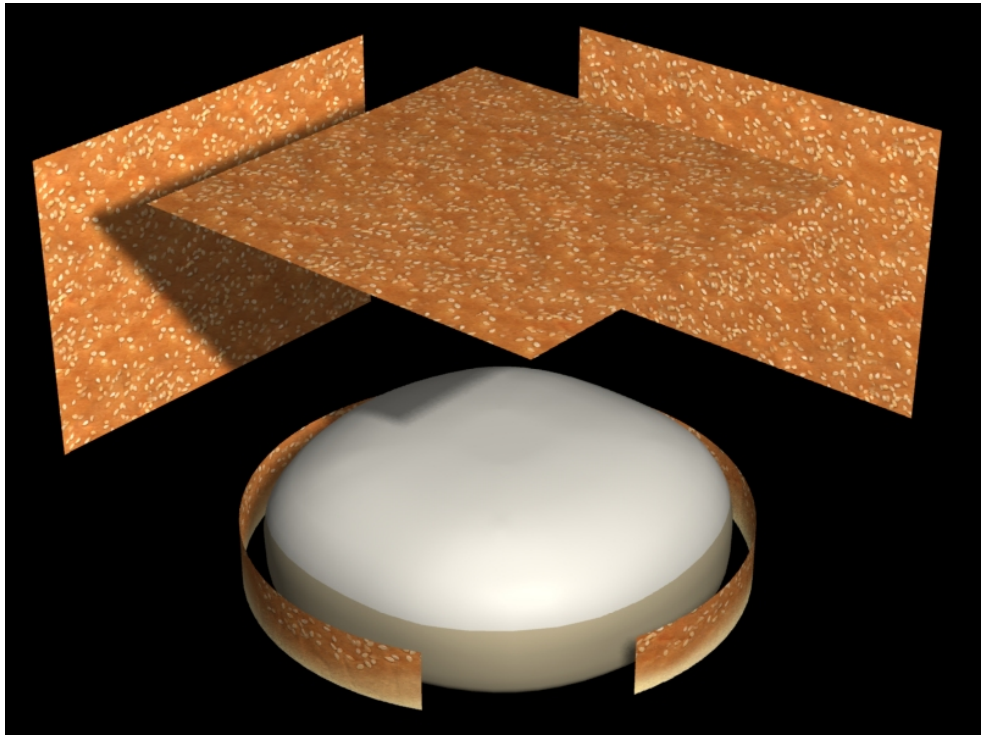
Cilindro



Sfera

Texture Mapping 2D: esempio

Two Part Mapping



Texture Mapping 2D: esempio

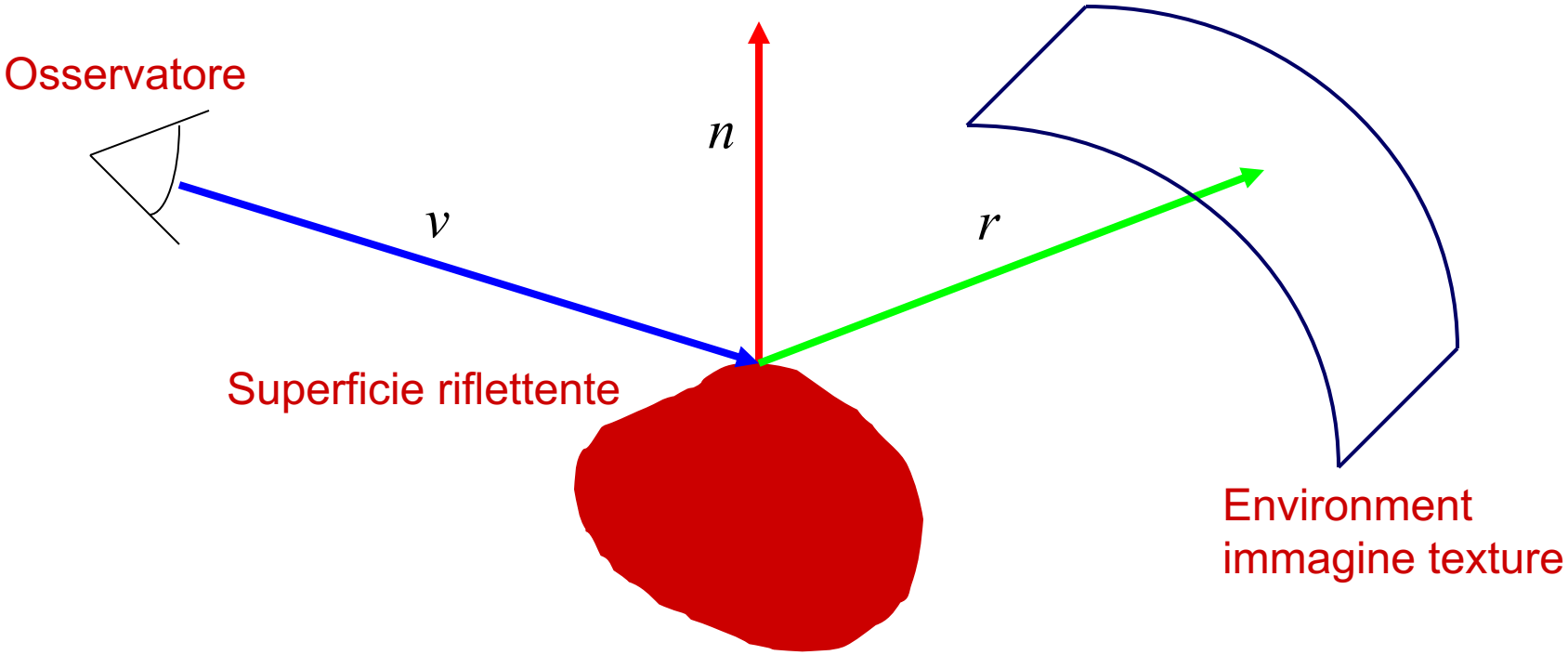
Texture Mapping 2D: esempio

Two Part Mapping:

cilindro

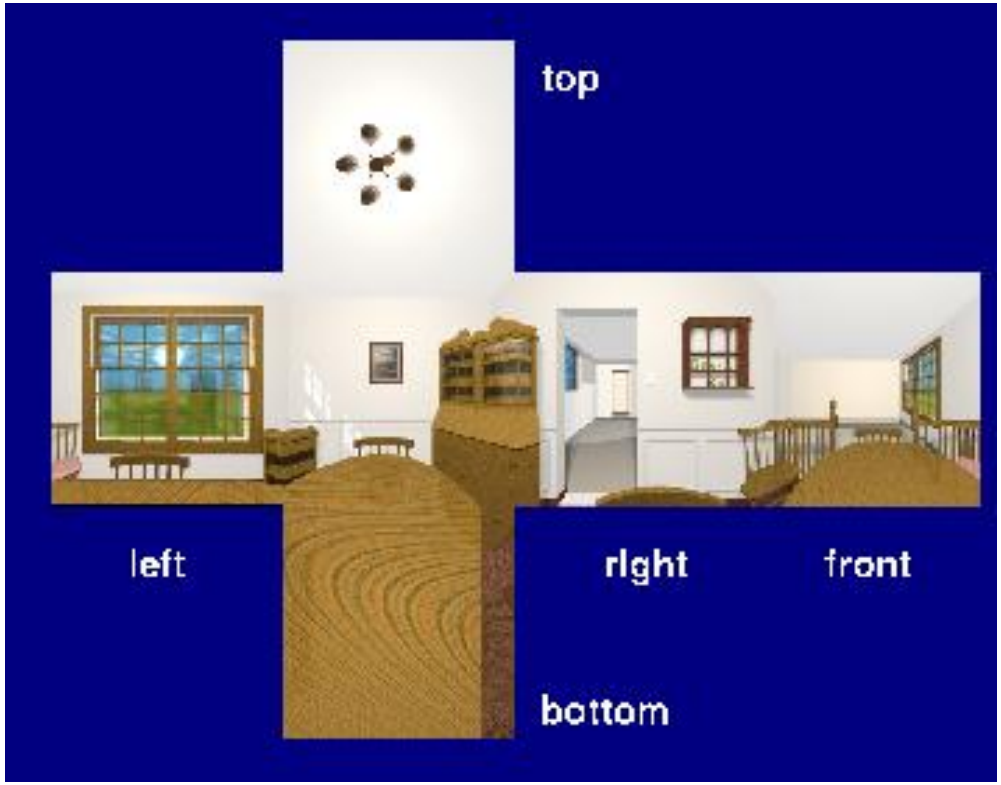
Environment Mapping

Raggio riflesso: $r = 2(n \cdot v)n - v$



La texture viene applicata nella direzione del raggio riflesso *r*, dall'environment immagine texture sull'oggetto

Cube Environment Mapping

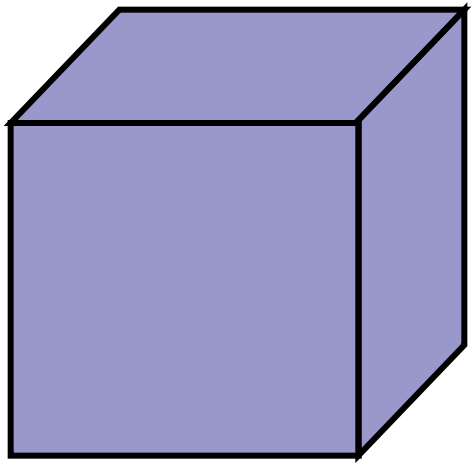


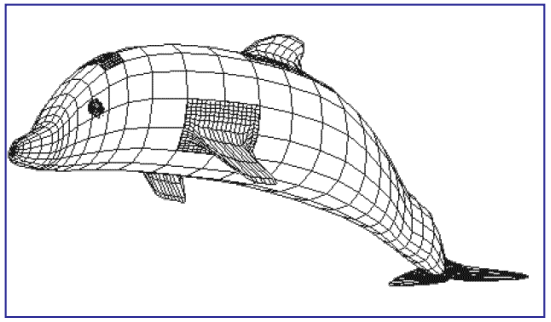
Solid Texturing o 3D Texture

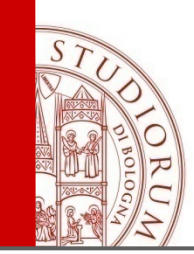
(anche Texture Mapping 3D o Texture Procedurale)

Texture a 3 dimensioni che viene applicata ad un modello 3D

L'oggetto 3D viene "immerso" nello spazio texture 3D e il colore di ogni punto dell'oggetto viene definito dalla sua posizione nello spazio texture.







Solid Texturing

Vantaggi:

- Non è più necessario applicare texture 2D a modelli 3D
- Possibilità di simulare in maniera realistica materiali naturali (uso di texture procedurali 3D)
- Nuove possibilità di applicazioni

Svantaggi

- Alta occupazione di memoria delle texture (una texture 64x64x64 RGBA occupa 1MB, una texture 256x256x256 RGBA occupa 64MB)
- Per texture procedurali overhead computazionale che ne limita l'uso in applicazioni real-time (a meno che non siano state pre-calcolate)
- Difficoltà d'uso per l'applicazione all'oggetto di immagini (texture come immagine e non come procedural texture, es: volto di una persona)

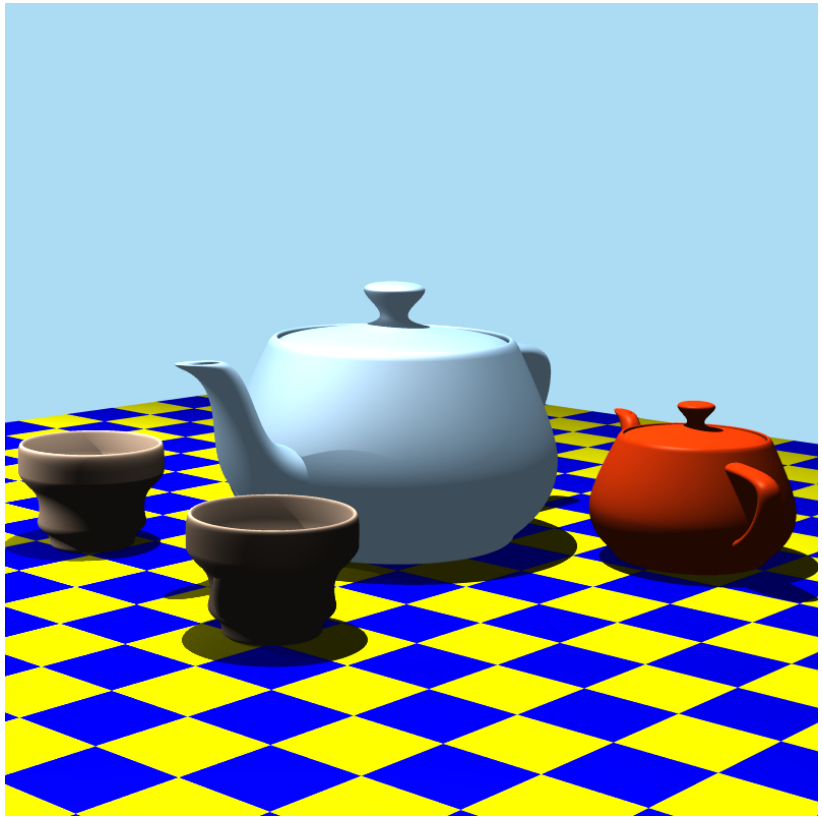
Solid Texturing: esempio

wood texture

Lo spazio 3D texture consiste in cilindri concentrici e alterni in colore; come un tronco d'albero.

Solid Texturing: esempio

cube texture



Lo spazio 3D texture consiste in una scacchiera 3D

Texture Procedurali

L'approccio procedurale non nasce con le texture, …

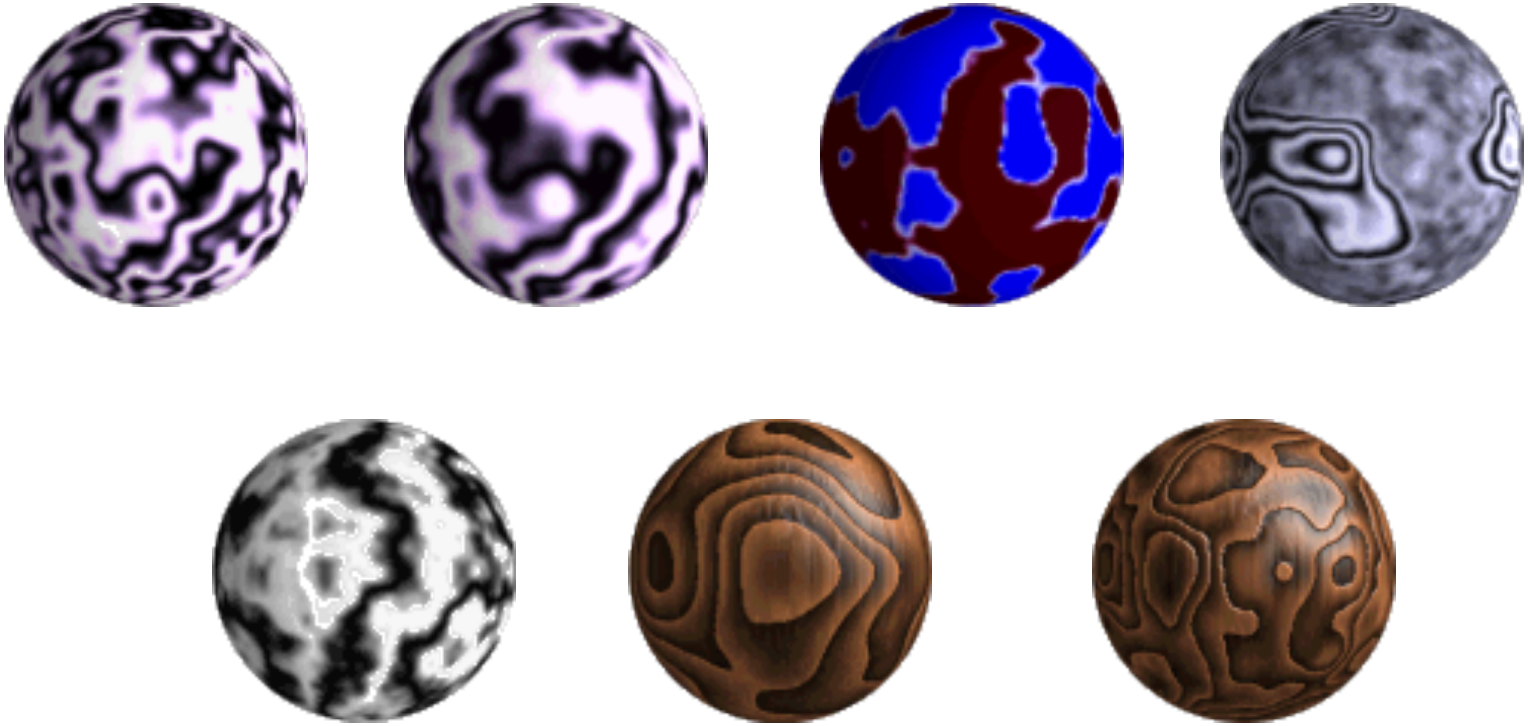
…ma trova il suo massimo sviluppo con le texture 3D.

E' possibile costruire texture procedurali anche 2D o 4D (dove le 4ª dimensione può essere il tempo – utile per creare effetti fisici realistiche, come le nuvole, ecc.).

Nota: con metodo procedurale si intende un metodo matematico, tipicamente pseudo-casuale o di tipo frattale (sono anche detti solid noise method).

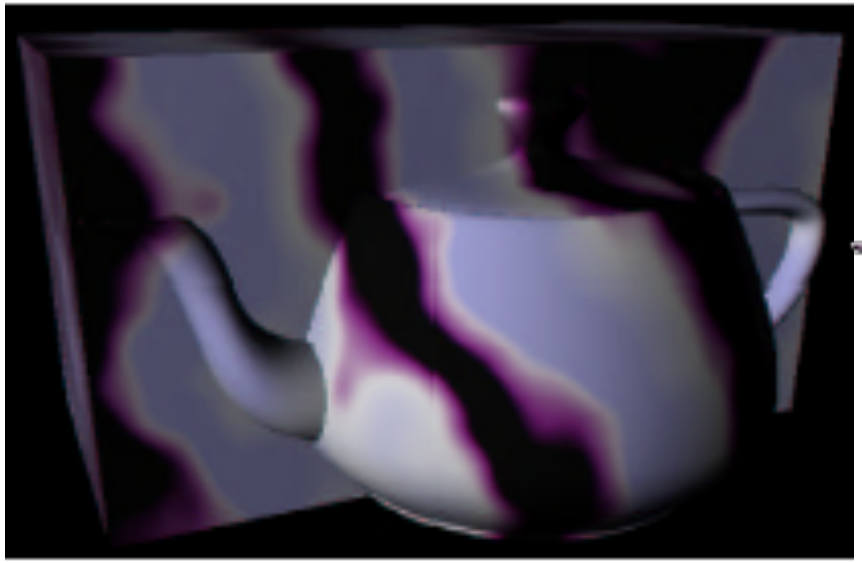
Questi metodi sono utili per la costruzione di texture che rappresentano materiali naturali come il legno, il marmo, la roccia, ecc.

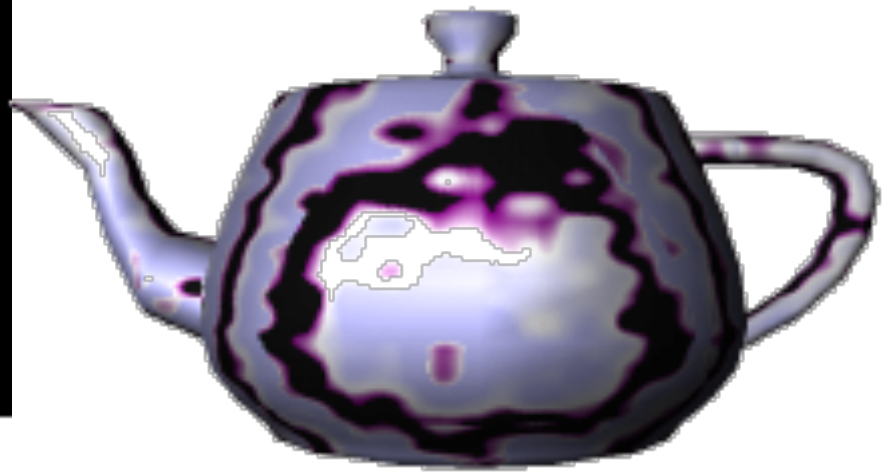
Texture Procedurali



3D Perlin Noise

Texture Procedurali o Solid Texturing: esempio



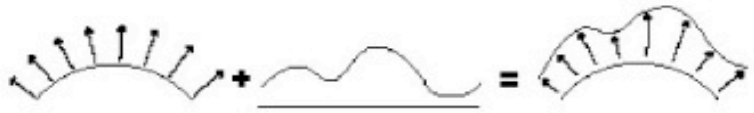


Bump/Normal Mapping

- Si sostituiscano i colori R,G,B con coordinate X,Y,Z
- Si interpretano i pixel come vettori normali
- L'immagine prodotta fa apparire la geometria molto più complessa di quello che è.

Bump Mapping

Offset surface position



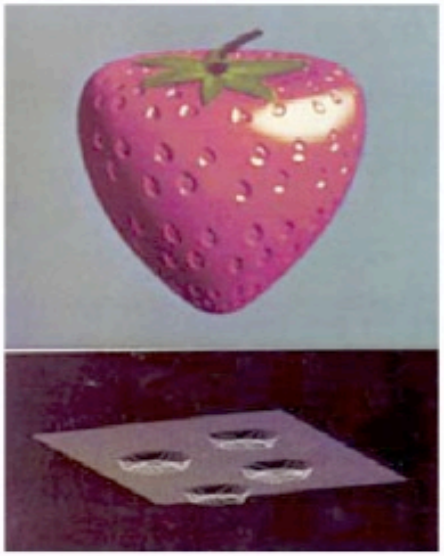
Displacement

$$\mathbf{N}(u,v) = \frac{\mathbf{P}_u(u,v) \times \mathbf{P}_v(u,v)}{\left|\mathbf{P}_u(u,v) \times \mathbf{P}_v(u,v)\right|}$$

$$\mathbf{P}'(u,v) = \mathbf{P}(u,v) + h(u,v)\mathbf{N}(u,v)$$

Perturb normal

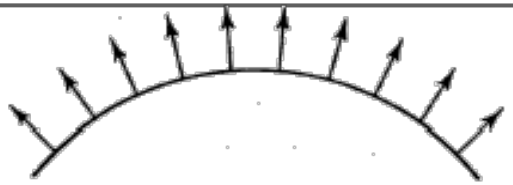
$$\mathbf{N}'(u,v) = \frac{\mathbf{P}'_u(u,v) \times \mathbf{P}'_v(u,v)}{\left|\mathbf{P}'_u(u,v) \times \mathbf{P}'_v(u,v)\right|}$$



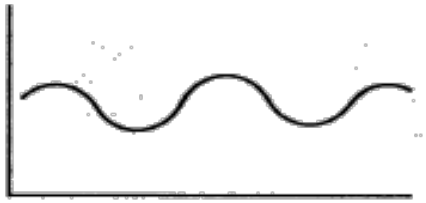
From Blinn 1978

Formulazione originale di Blinn (1978)

Bump Mapping

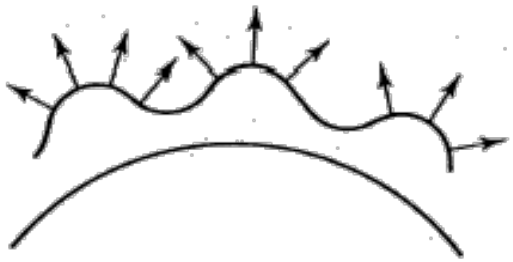


$O(u)$
Original surface

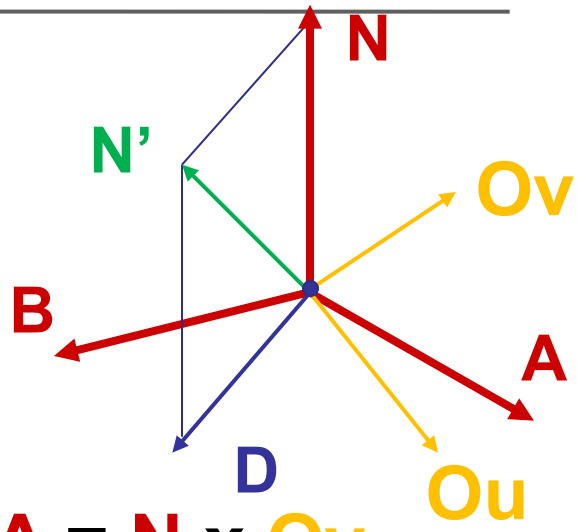


$B(u)$
A bump map

$O'(u)$
Lengthening or shortening $O(u)$ using $B(u)$



$N'(u)$
The vectors to the 'new' surface



$A = N \times Ov$

$B = N \times Ou$

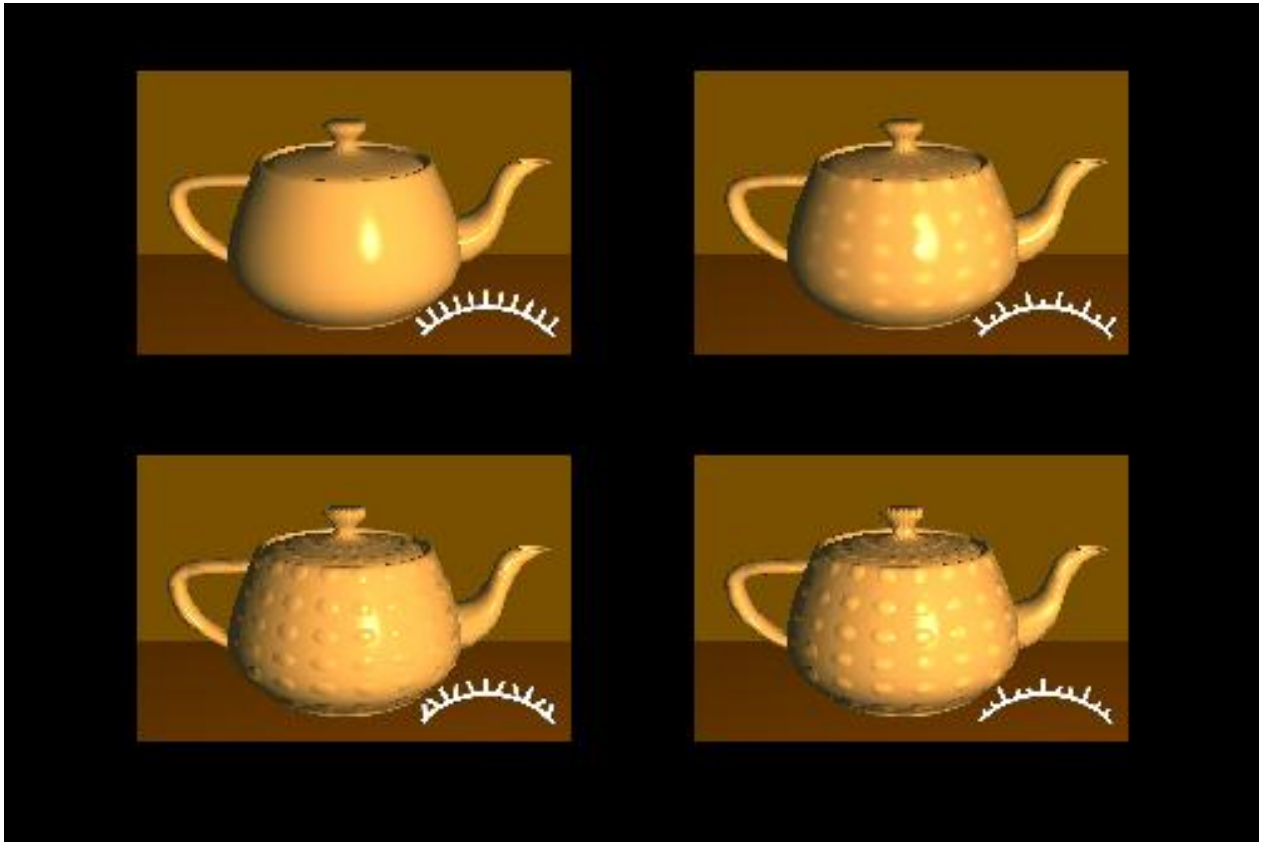
$N' = N + D$

dove $D = Bu\ A - Bv\ B$

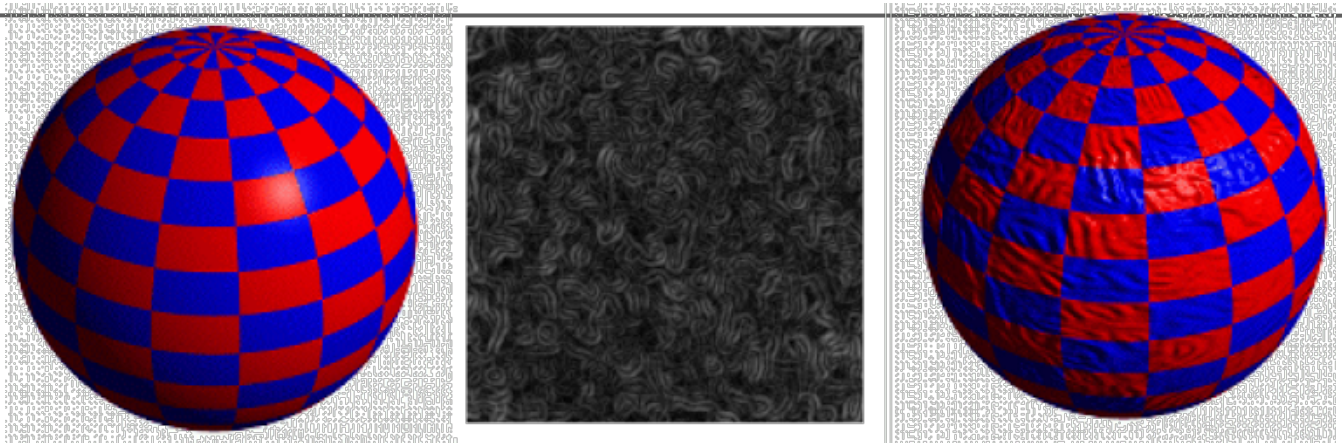
$Bu = dB(u,v)/du$

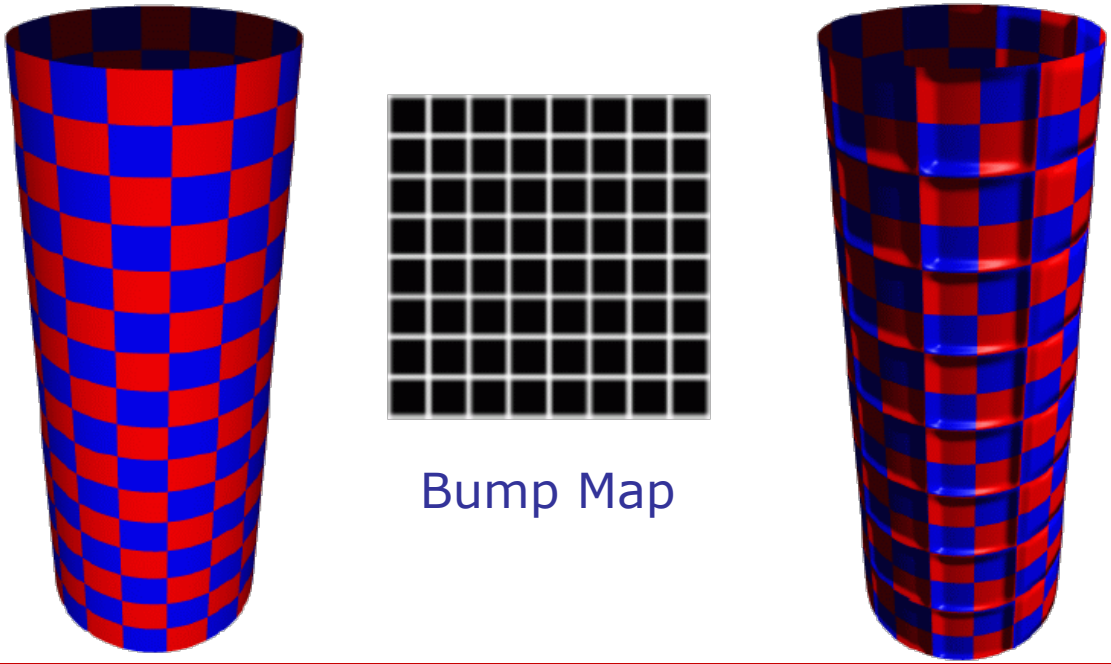
$Bv = dB(u,v)/dv$

Bump Mapping: esempi



Bump Mapping: esempi



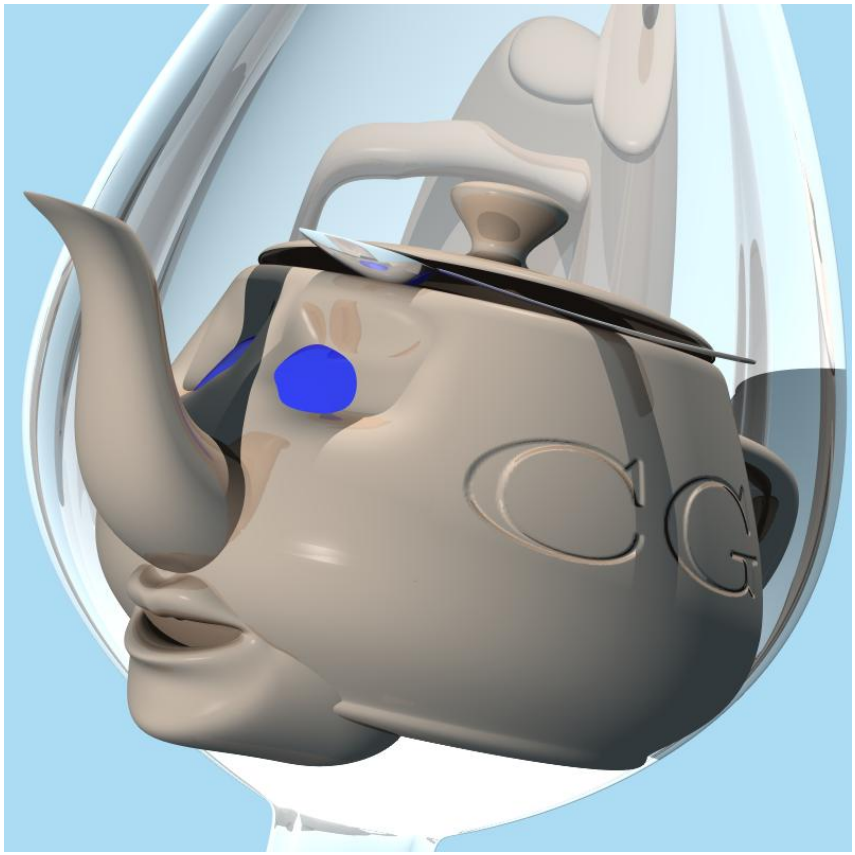


Bump Map

Bump Mapping: esempio

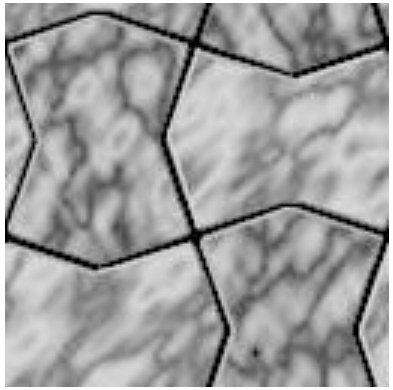
Bump Mapping: esempio

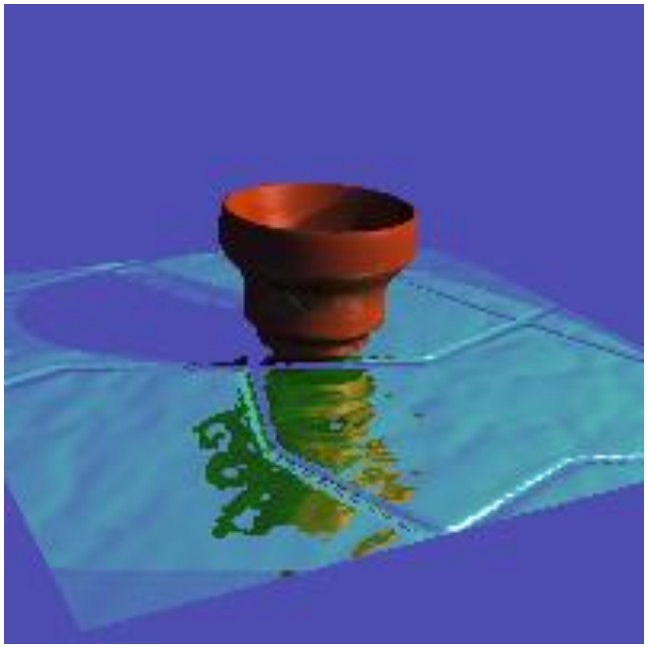
CG



Logo Corso Grafica A.A. ….

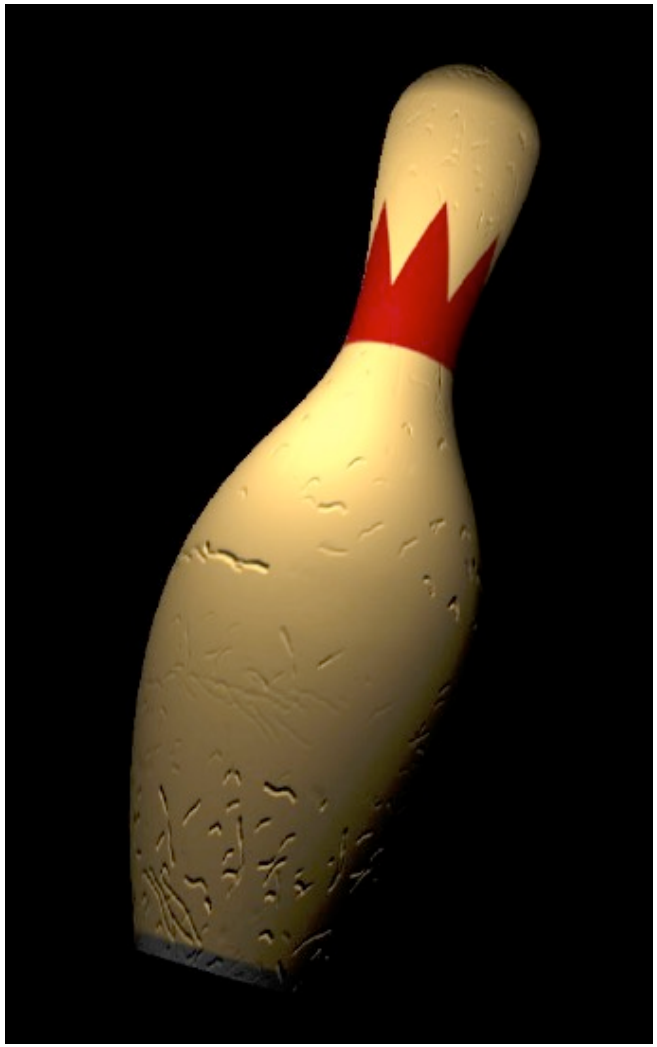
Bump Mapping: esempio



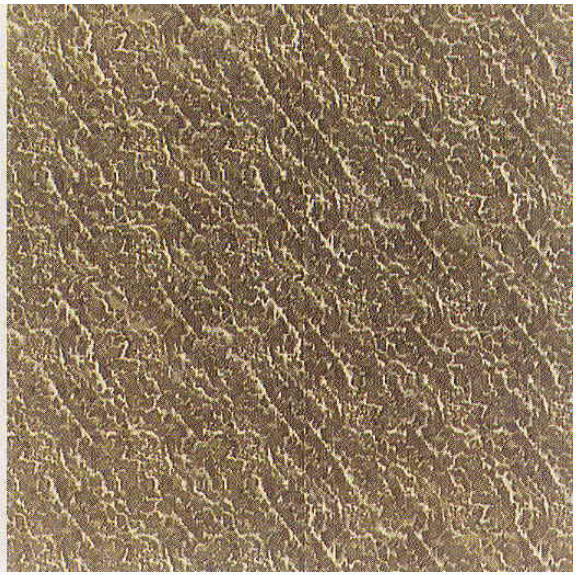


Bump Mapping: esempio

Bump Mapping: esempio



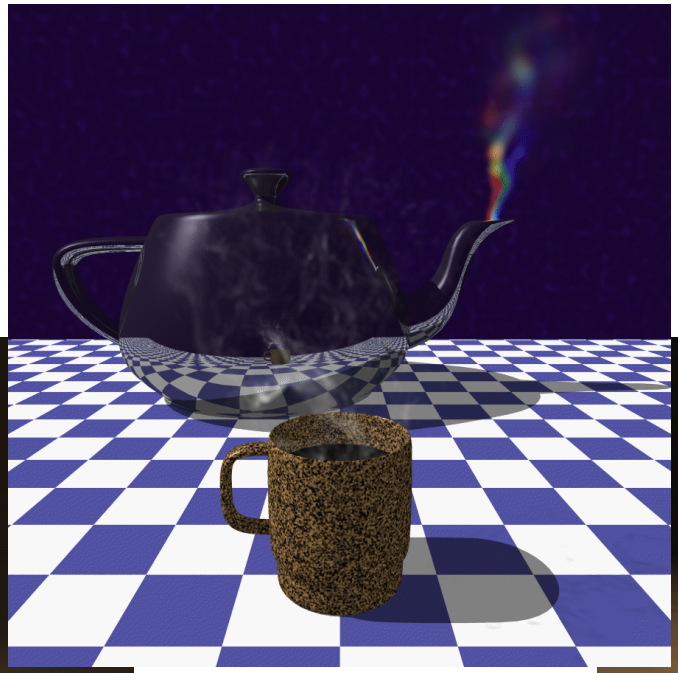
Bump Mapping

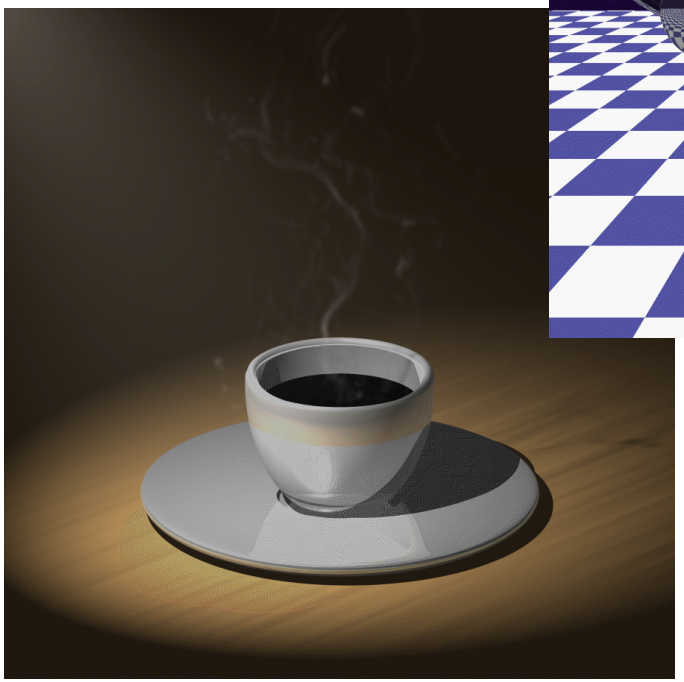


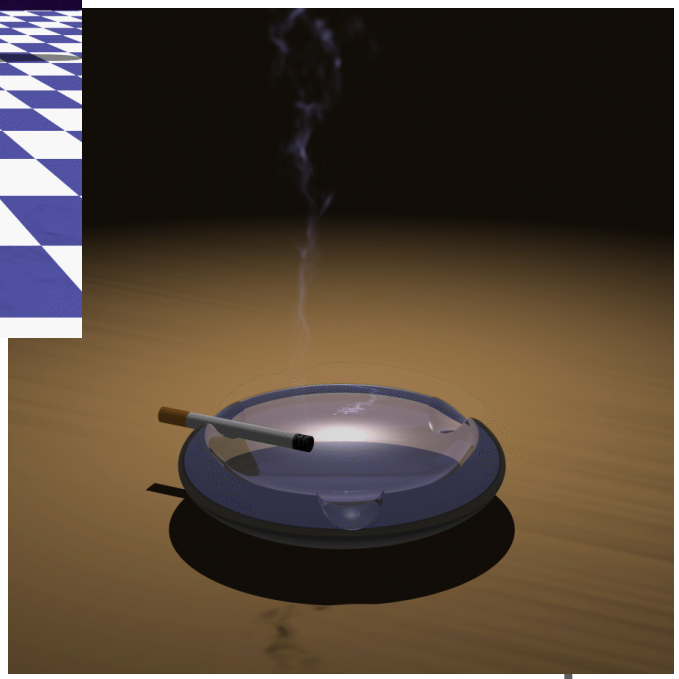
Multitexturing

Texture Mapping 2D + Bump Mapping

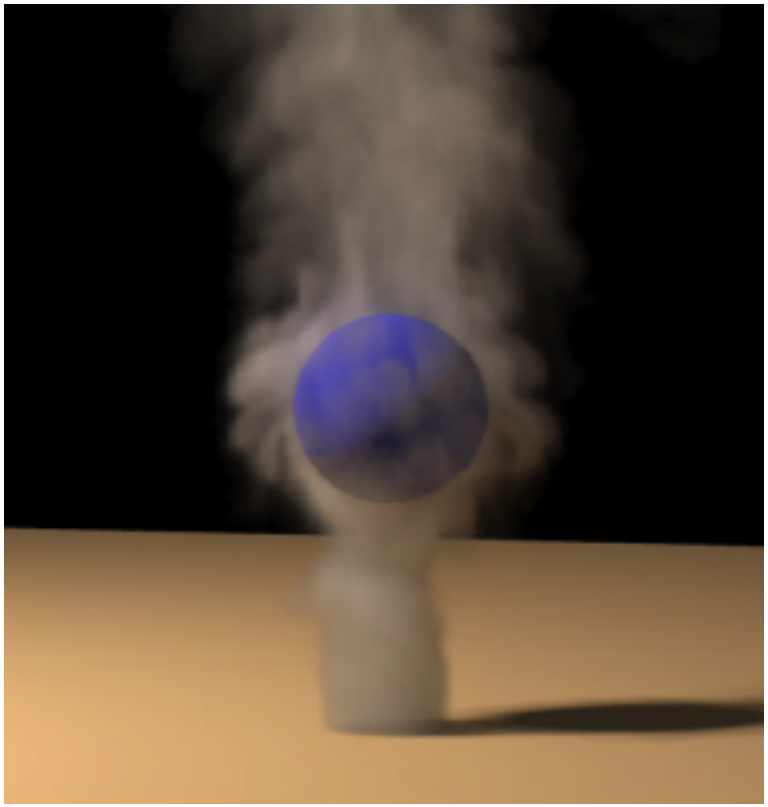
Smoke Simulation: esempio



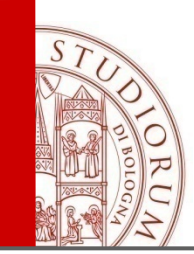




Smoke Simulation: esempio



Si noti come il fumo si muove correttamente intorno alla sfera

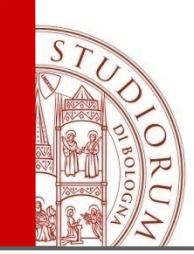


Texture con WebGL

1. Associare le coordinate texture 2D ai vertici 3D della mesh

2. Specificare la texture
   - Leggere o generare l'immagine
   - Assegnare l'immagine alla texture RAM

3. Specificare i parametri della texture
   - Filtering, Wrapping, …



Creare un texture object e specificare una Texture per quell'oggetto

Un texture object memorizza texture data (a livello di GPU) e li rende disponibili per un loro utilizzo/riutilizzo

1. (init( )) Sia image l'id di un'immagine RGBA (array di texel);

2. (init( )) Crea un texture object
   texName = gl.CreateTexture( );

3. (init( )) Associa un texture object ad un tipo di texture
   gl.bindTexture(target, texName);

target: gl.TEXTURE_2D, gl.TEXTURE_CUBE_MAP,
gl.TEXTURE_CUBE_MAP_POSITIVE_X,
gl.TEXTURE_CUBE_MAP_NEGATIVE_X, ecc.

Creare un texture object e specificare una Texture per quell'oggetto (continua)

4. (init( )) copia una texture image 2D da un array di texel in texture RAM della GPU:

```
gl.texImage2D( target, level, internalformat,
        w, h, border, format, type, image );
```

- target: gl.TEXTURE_2D, gl.TEXTURE_CUBE_MAP, gl.TEXTURE_CUBE_MAP_POSITIVE_X, ecc.
- level: livello del mipmap, 0 se non si utilizza mipmapping;
- internalformat: componenti colore (RGBA) considerate;
- w,h,border = width, height e bordo in pixel

Es.: gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 512, 512, 0
gl.RGBA, gl.UNSIGNED_BYTE, image);

Anche:

```
gl.texImage2D( target, level, internalformat,
        format, type, image );
```

Creare un texture object e specificare una Texture per quell'oggetto (continua)

Le coordinate texture (coordinate 2D) sono attributi dei vertici della mesh e devono essere passati alla GPU come le coordinate dei vertici;

```
var texcoordBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, texcoordBuffer);

gl.bufferData(gl.ARRAY_BUFFER, new
            Float32Array(texcoord),gl.STATIC_DRAW);
```

Dove texcoord è un array JavaScript contenente le coordinate texture 2D.

MIP Mapping

WebGL mette a disposizione il metodo:

```
gl.generateMipmap(target);
```

target: gl.TEXTURE_2D, gl.TEXTURE_CUBE_MAP

Un mipmap, come detto, è una collezione di immagini/ /texture progressivamente più piccole, ciascuna di dimensioni 1/4 della precedente.

La parte fissa della pipeline, una volta generate le texture, se richiesto, sceglie quella che si adatta meglio alla risoluzione delle facce da texturare.

MIP Mapping e Bind

Attenzione: se le dimensioni delle immagini/texture non sono una potenza di due non si può richiedere a WebGL 1.0 di operare un MipMap.

E' bene procedere prima ad una scala dell'immagine/texture per portarla a dimensioni che siano potenze di 2

bind del texture object ogni volta che si deve rendere l'oggetto al quale la texture è associata

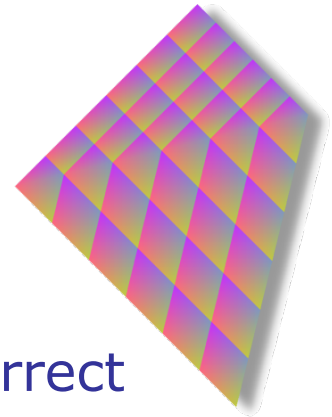
```
gl.bindTexture(gl.TEXTURE_2D, texName[i]);
```

Interpolazione valori Texture

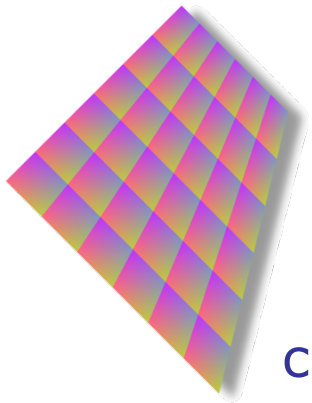
WebGL usa l'interpolazione bilineare per determinare le coordinate texture dei pixel delle facce triangolari così come per determinare i colori, dai colori dei vertici.

Per default l'interpolazione è con correzione prospettica; a questo proposito si vedano i due codici nella cartella HTML5_webgl_2:

cube_texture_perspective_non_correct.html
cube_texture_perspective_correct.html



non correct



correct (default)

Modalità di applicazione delle Texture

Si può chiedere a WebGL come applicare queste texture settando per ognuna i parametri di filtering e di wrap, mediante:

gl.texParameteri/f(target , pname , …);

- target: gl.TEXTURE_2D, gl.TEXTURE_CUBE_MAP
- pname: gl.TEXTURE_{MIN,MAG}_FILTER
  gl.TEXTURE_WRAP_{S,T},

Modalità di filtri

- minification o magnification
- speciali filtri di minification mipmap

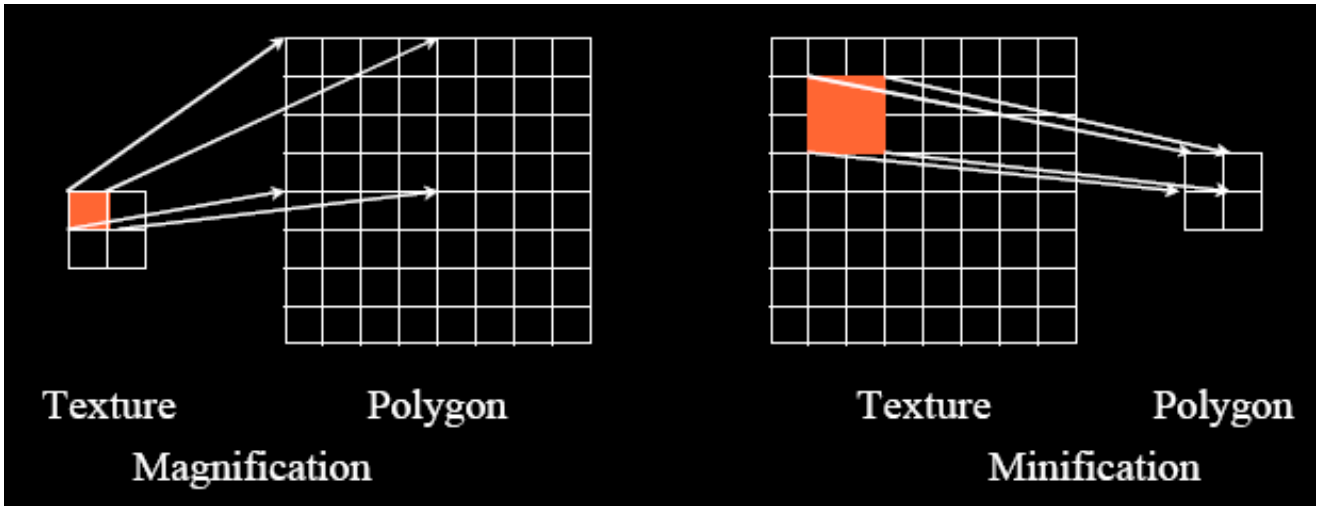
Modalità Wrap

- clamping o repeating

Modalità di filtri: Anti-Aliasing

gl.texParameteri(target, gl.TEXTURE_{MIN,MAG}_FILTER, param)

param: gl.NEAREST, gl.LINEAR,
gl.NEAREST_MIPMAP_NEAREST, gl.LINEAR_MIPMAP_NEAREST,
gl.NEAREST_MIPMAP_LINEAR , gl.LINEAR_MIPMAP_LINEAR



Magnification:
il texel è più piccolo di un pixel
Soluzione: interpolazione

Minification:
il texel è più grande di un pixel
Soluzione: media

Modalità di filtri: Anti-Aliasing

gl.texParameteri(target, gl.TEXTURE_{MIN,MAG}_FILTER, param)

param: gl.NEAREST, gl.LINEAR,
gl.NEAREST_MIPMAP_NEAREST, gl.LINEAR_MIPMAP_NEAREST,
gl.NEAREST_MIPMAP_LINEAR , gl.LINEAR_MIPMAP_LINEAR

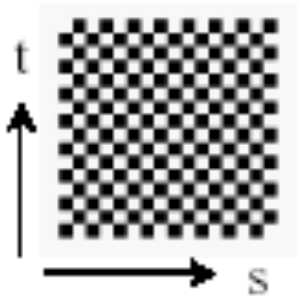
- NEAREST = choose 1 pixel from the biggest mip
- LINEAR = choose 4 pixels from the biggest mip and blend them

- NEAREST_MIPMAP_NEAREST = choose the best mip, then pick one pixel from that mip
- LINEAR_MIPMAP_NEAREST = choose the best mip, then blend 4 pixels from that mip
- NEAREST_MIPMAP_LINEAR = choose the best 2 mips, choose 1 pixel from each, blend them
- LINEAR_MIPMAP_LINEAR = choose the best 2 mips. choose 4 pixels from each, blend them

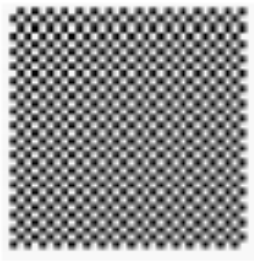
Modalità di Wrap

gl.texParameteri(target, gl.TEXTURE_WRAP_{S,T} , param)

param: gl.REPEAT, gl.CLAMP_TO_EDGE, gl.MIRRORED_REPEAT



texture

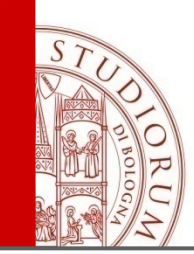


gl.REPEAT

> [0,1]

gl.CLAMP_TO_EDGE

Valori >1.0 set 1.0
Valori < 0.0 set 0.0



Environment Mapping

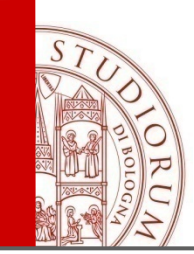
Come modalità automatica di applicazione delle texture, in WebGL esiste solo il cube environment mapping, cioè il mapping tella texture sulla geometria specificata mediante riflessione sulle pareti di un cubo che contiene la geometria stessa.

Questo si ottiene semplicemente specificando come target nelle varie chiamate

gl.TEXTURE_CUBE_MAP

Nelle applicazioni è utile poter avere le coordinate texture in modo semplice/automatico per proiezione da un piano nello spazio, o mediante una superficie (two-part mapping), o altro, ma WebGL non mette a disposizione nulla di tutto ciò.

Sarà compito dell'utente programmare tali metodi, se necessari, o cercare librerie realizzate da altri e utili a questo scopo.



Texture e Shader (GLSL)

Come si è detto, ad ogni vertice viene associata una coordinata texture e ogni pixel/fragment interno ad un triangolo ha un colore dato dall'interpolazione del colore dei vertici.

Le coordinate texture vengono lette insieme ai vertici dal Vertex Shader che le passa come varying al Fragment Shader che nella corrispondente variabile varying ottiene le coordinate interpolate (l'interpolazione viene effettuata dalla parte fissa della pipeline).

Quindi il Fragment Shader effettua il campionamento della texture

```
precision mediump float;
varying vec2 v_texcoord; // Passed in from the vertex shader
uniform sampler2D u_texture; // The texture
void main() {
    gl_FragColor = texture2D(u_texture, v_texcoord);
}
```

Esercizi

In HTML5_webgl_2 analizzare e provare i seguenti codici:

cube_textures_atlas.html e .js
cube_texture_skybox.html e .js
cube_texture_cubemap.html e .js
cube_texture_environment_map.html e .js

e analizzare i seguenti:

cube_texture_perspective_correct.html
cube_texture_perspective_non_correct.html
cube_texture_skybox_environment_map.html

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Giulio Casciola

Dip. di Matematica

giulio.casciola at unibo.it