Corso di *Tecniche Avanzate per la Grafica*

Tecniche di Shadowing

Docente:

Massimiliano Corsini

Laurea Specialistica in Informatica

Facoltà di Scienze MM. FF. NN.
Università di Ferrara

Overview

- Shadow Mapping
- Soft Shadow (PCF)

Ombre

- Conferiscono realismo ad una scena
- Le ombre ci aiutono a collocare spazialmente gli oggetti
- Senza le ombre gli oggetti ci appaiono “fluttuare” nel vuoto
- Ricordiamoci che sono un effetto visivo ***globale*** e che quindi dipendono da tutta la scena
- Sono state sviluppate molte tecniche per il rendering delle ombre in real-time

Ombre

Image from ARTIS research group (INRIA)

Disegnare le ombre...

- Cosa significa disegnare le ombre su una scena?
- Significa calcolare, per ogni pixel della scena, se questo riceve luce oppure è occluso (messo in ombra) da qualche altro oggetto della scena
- Un pixel può anche ricevere “poca” luce e non essere completamente oscurato

Shadow Mapping

in parte tratto da
“Hardware Shadow Mapping”
NVidia Corporation

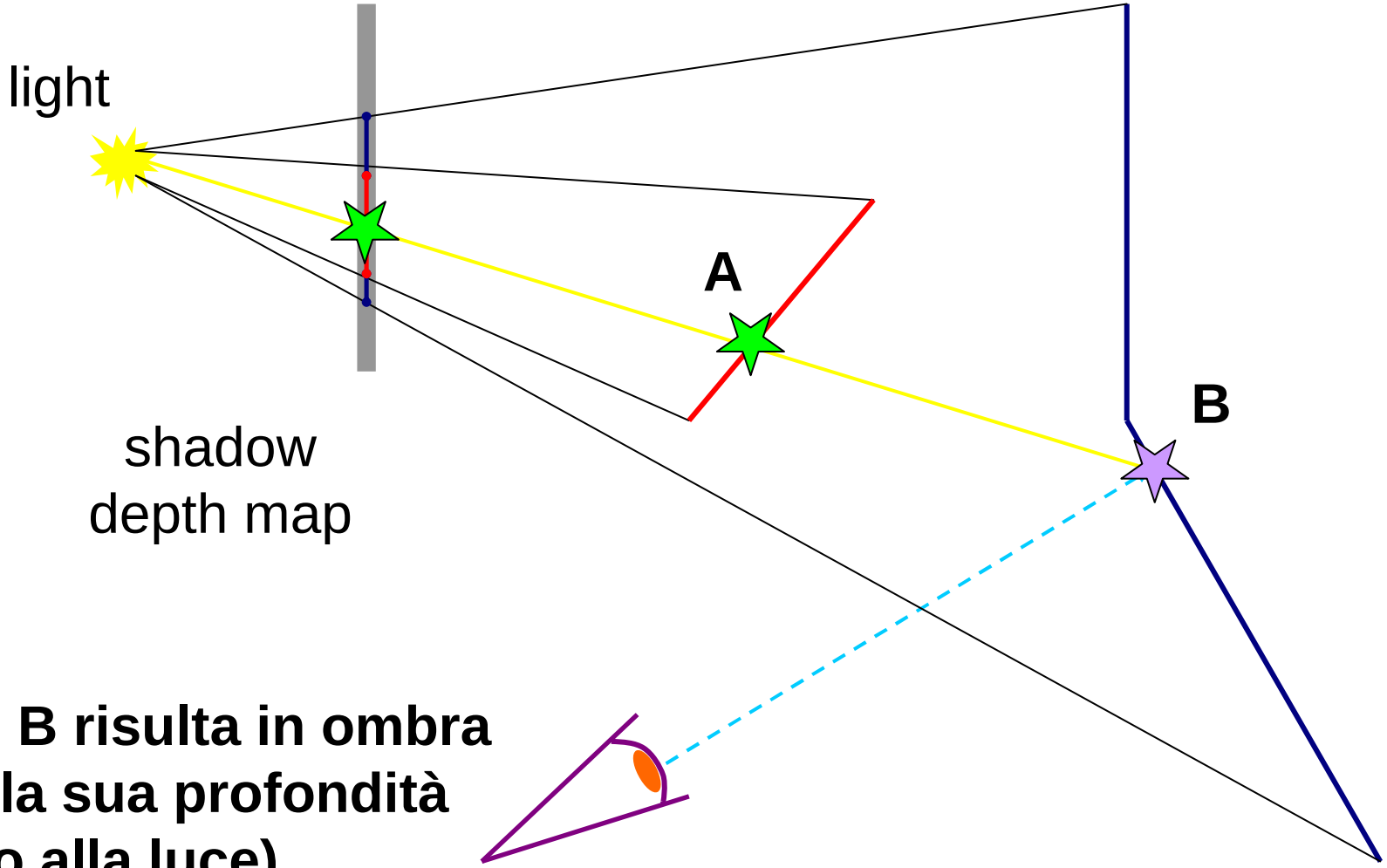
Shadow Mapping

- Shadow mapping è una tecnica image-based sviluppata da Lance Williams nel 1978.
- È particolarmente adatta ad essere implementata sull'hardware della scheda grafica (utilizza il depth buffer ed il texturing)
- In più è una tecnica molto semplice da implementare ed a differenza di altre tecniche (shadow volumes) non richiede di elaborare la geometria degli oggetti.

Idea base

- L'idea base consiste nel renderizzare la scena dal punto di vista della luce (si assume di avere una luce *spotlight*)
- A questo punto l'hardware grafico nel depth buffer ha memorizzato la visibilità risolta dal punto di vista della luce, ossia i pixel in luce e la relativa profondità
- Si utilizza poi il depth buffer così ottenuto (shadow map) in una passata di rendering successiva per determinare i pixel in ombra

Idea base



Il punto B risulta in ombra perchè la sua profondità (rispetto alla luce) è maggiore del punto A.

Idea base

Eye’s Viewpoint

Light’s Viewpoint

Shadow Map

Algoritmo

- Passi dell'algoritmo di shadow mapping:
  1. Si effettua il rendering (su texture) della scena dal punto di vista della luce (***light space***)
  2. Si utilizza il depth buffer così ottenuto (shadow map) come texture corrente
  3. Si proietta la shadow map sulla scena
  4. Nel fragment shader si effettua il test di shadow

Accorgimenti

- L’algoritmo appena mostrato implementato senza nessun tipo di accorgimento non può funzionare (funzionerebbe solo se avessimo precisione infinita e risoluzione infinita)
- Pertanto conviene aggiungere un offset alla shadow map per evitare artifatti di self-shadowing
- Per far questo si può usare la glPolygonOffset(...)

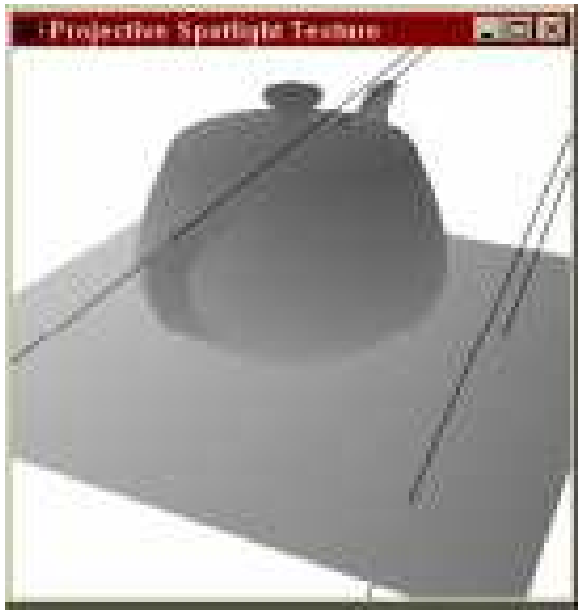
Proiettare la shadow map

- Si deve utilizzare la **`glTexGen(..., EYE_LINEAR)`** → ovviamente il piano da settare è quello relativo alla shadow map generata dal punto di vista della luce
- Ci ritroviamo nel fragment shader le coordinate *r,s,t,q* della texture proiettata → *r/q* è la distanza del pixel corrente dalla luce, *s/q* e *t/q* indicizzano la shadow map
- Il pixel riceve luce se

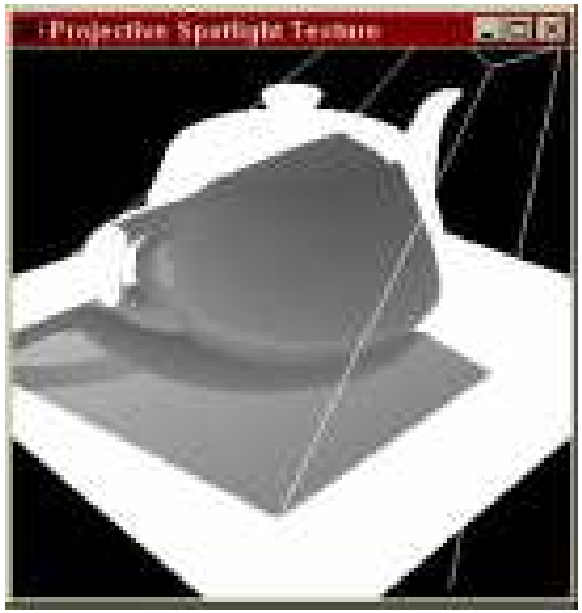
  r/q < shadowmap(s/q, t/q) (shadow test)

Shadow Mapping

Scena con shadow mapping



Distanza Luce (per-pixel)

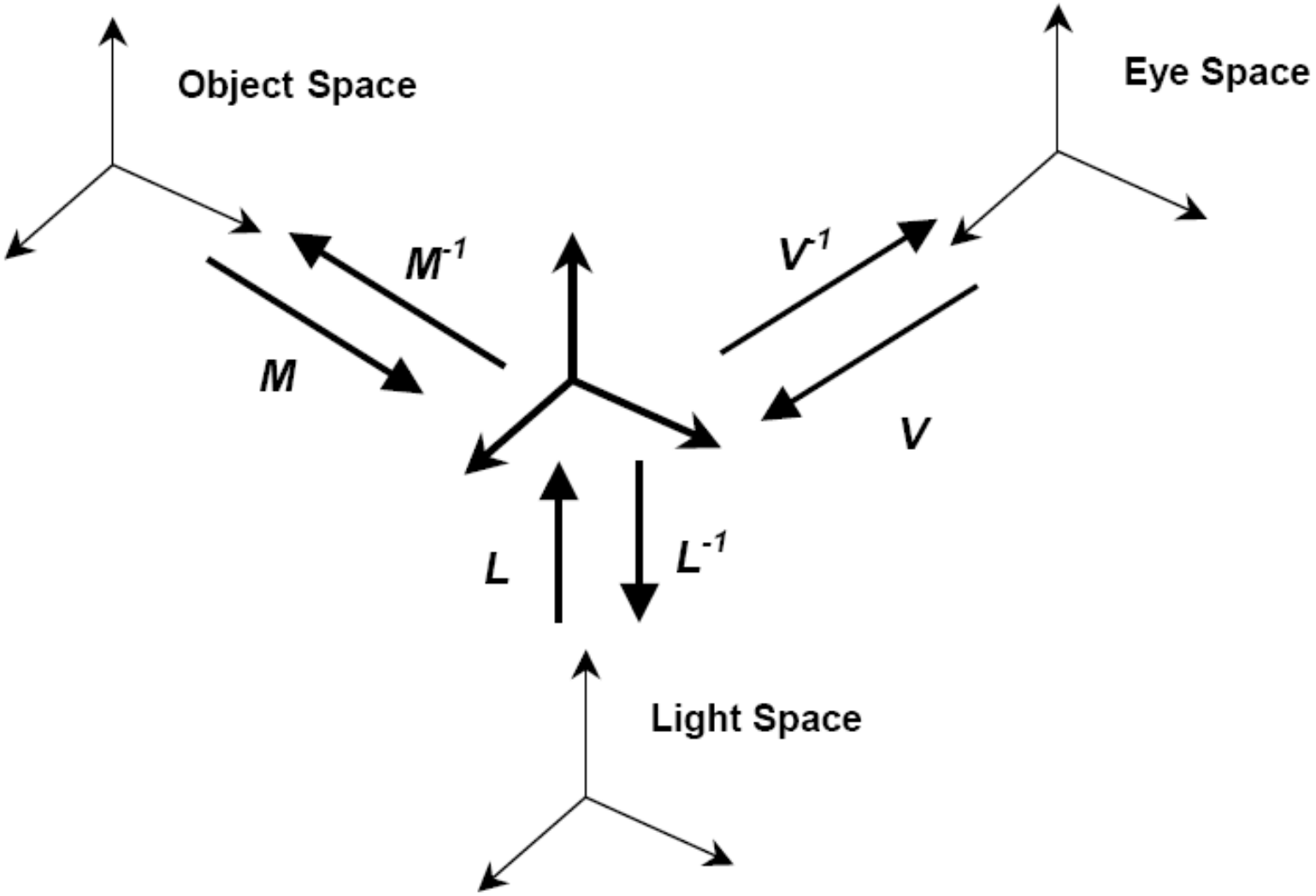


Shadow map proiettata

Vantaggi e Limitazioni

- Non richiede geometry processing aggiuntivo
- Buono solo per spotlight... E le luci puntiformi?
- La qualità della shadow map dipende dal campionamento del depth buffer in light's space rispetto a quello in eye's space
- Problema della "proiezione negativa"
- Problema del polygon offset...

Trasformazioni in gioco



Implementazione

- Per ulteriori dettagli implementativi vi rimando al sito del corso:
  - Tutorial NVidia sull'hardware shadow mapping
  - Link ad un tutorial semplice (con codice sorgente)

Soft Shadow

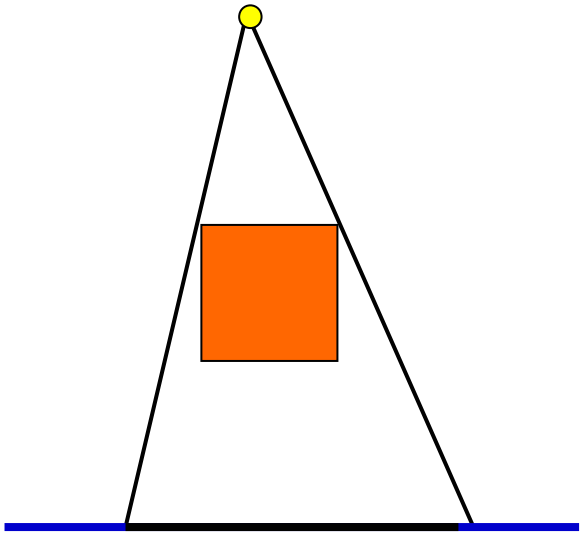
in parte tratto da
NVidia SDK White Paper
“Soft Shadow”

Soft Shadows and Hard Shadows

- Fino a quando assumiamo la luce puntiforme otteniamo delle ombre cosiddette ***hard***, ossia caratterizzate da una discontinuità netta.
- Nella realtà tutte le luce sono caratterizzate da un'area e questo produce l'effetto della ***penombra*** e conseguentemente l'ammorbidimento dei contorni delle ombre.

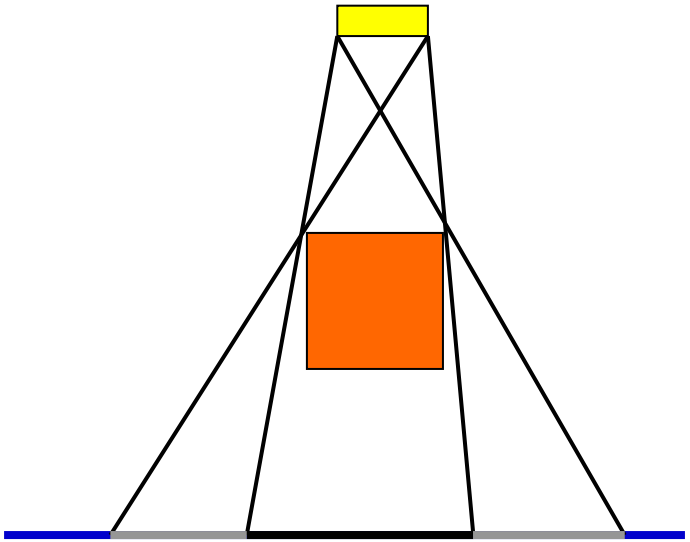
Soft Shadows and Hard Shadows

Luce puntiforme



ombra

Area-light



ombra

penombra

penombra

Hard vs Soft

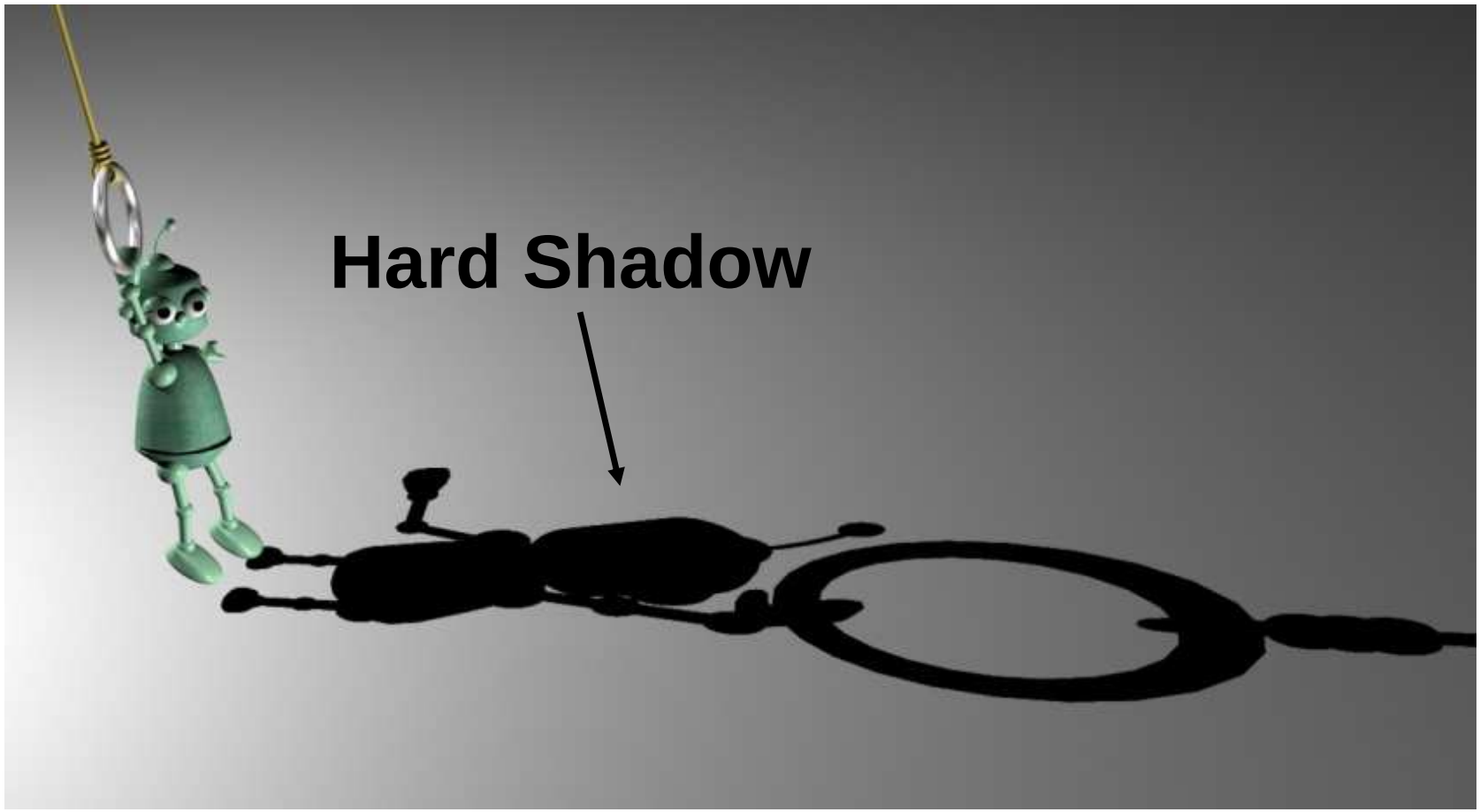


Image from ARTIS research group (INRIA)

Hard vs Soft

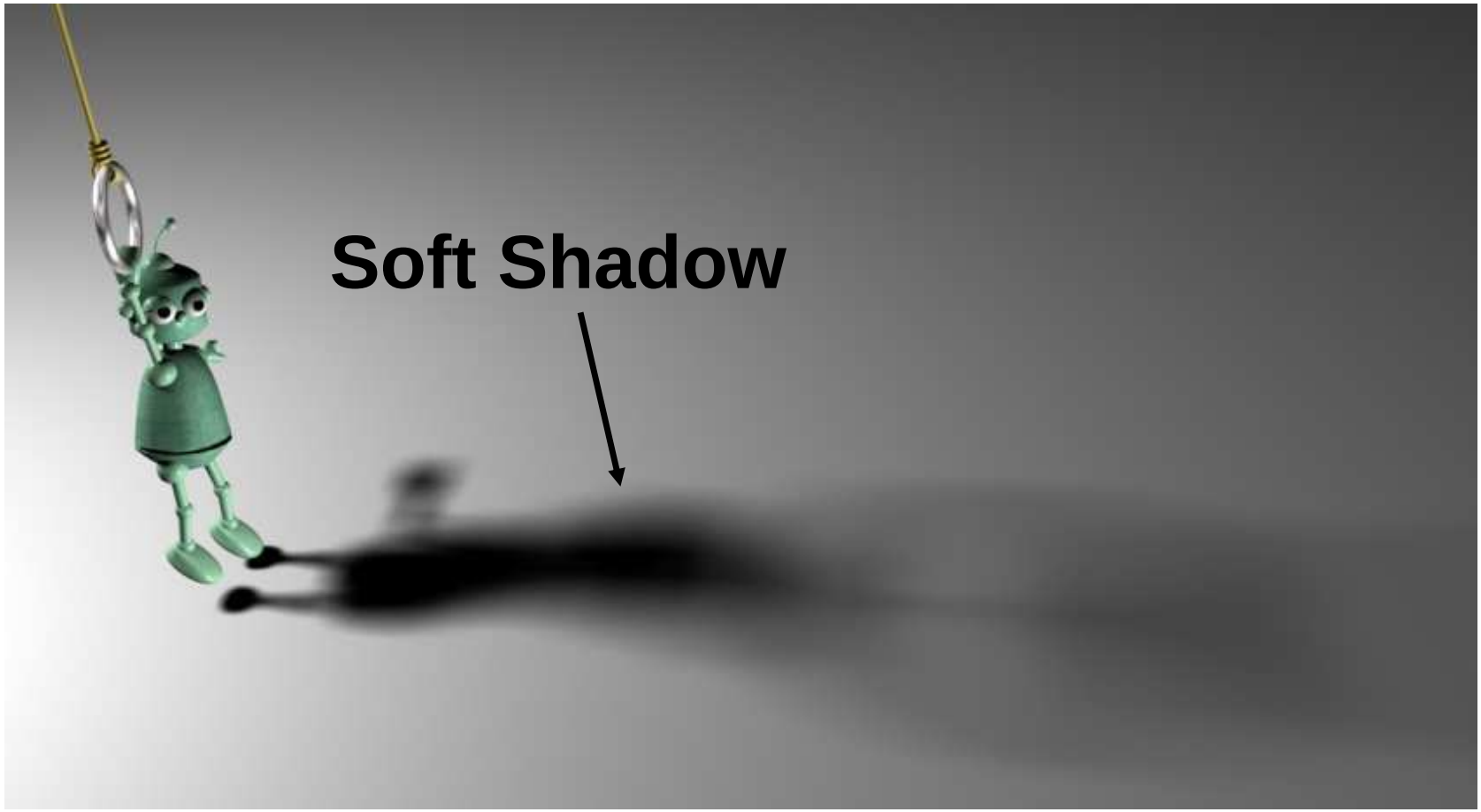


Image from ARTIS research group (INRIA)

Percentage-Closer Filtering

- Un’area-light può essere simulata con molte point-light
- Ma in questo modo avremmo bisogno di più shadow maps (una per point-light) per realizzare la penombra
- Guardando il problema da un altro punto di vista → anzichè più shadow maps utilizzo più campioni della stessa shadow maps per simulare la penombra
- Si ricorre ad un’idea tratta dal cosiddetto ***Percentage-Closer Filtering***

Percentage-Closer Filtering

- Alla Pixar fu sviluppata la tecnica del ***percentage-closer filtering (PCF)*** per mitigare gli effetti di aliasing nei bordi delle shadow map.
- Il PCF consiste nel confrontare la profondità del pixel corrente con la corrispondente posizione nella shadow map, effettuando il test di shadow anche con i pixel immediatamente vicini nella shadow map. L'intensità dell'ombra è proporzionale al numero di shadow test positivi.

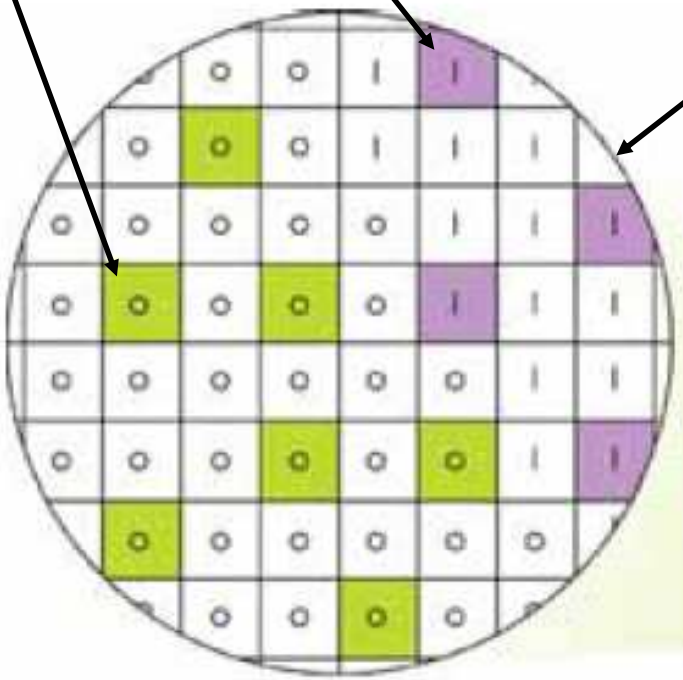
PCF

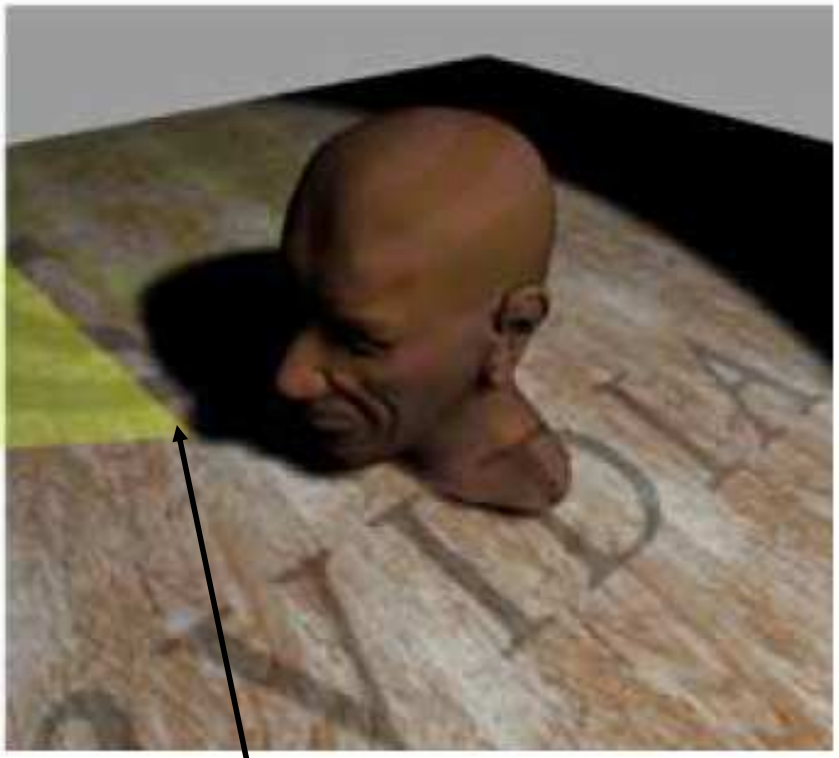
- Esempio, di 9 test effettuati 4 risultano in ombra, allora il pixel è in ombra al 50% circa, di 9 test 1 solo risulta in ombra, il pixel è in ombra al 10% circa.
- L'idea della tecnica soft-shadow basata su PCF è agire come nel PCF, ma anzichè considerare tutti i pixel immediatamente vicini al pixel considerato sulla shadow map, i test di shadow si effettuano su *alcuni pixel selezionati in un intorno più ampio*. Questo permette di aumentare l'area della penombra.

Percentage-Closer Filtering

pixels selezionati

intorno considerato sulla shadow map





pixel in esame

Campionando localmente su un'area della shadow map si può ottenere delle soft shadow

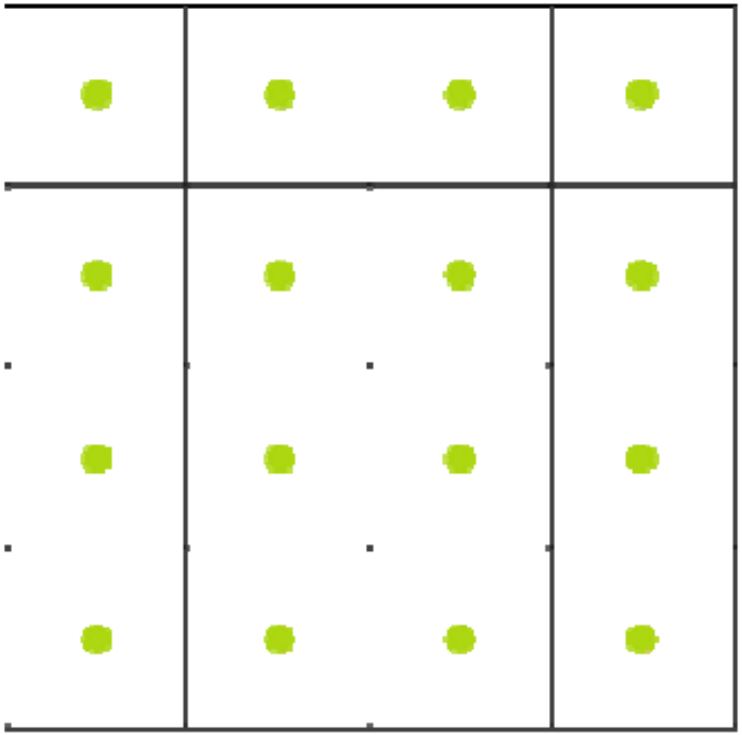
Considerazioni

- Più grande è l'intorno considerato sulla shadow map e più evidente è l'effetto di blurring sull'ombra → l'area della penombra aumenta
- Bisogna stare attenti a come si selezionano i pixel nell'intorno, ossia a come si campiona → un campionamento uniforme può portare a fastidiosi artifatti visivi (banding)

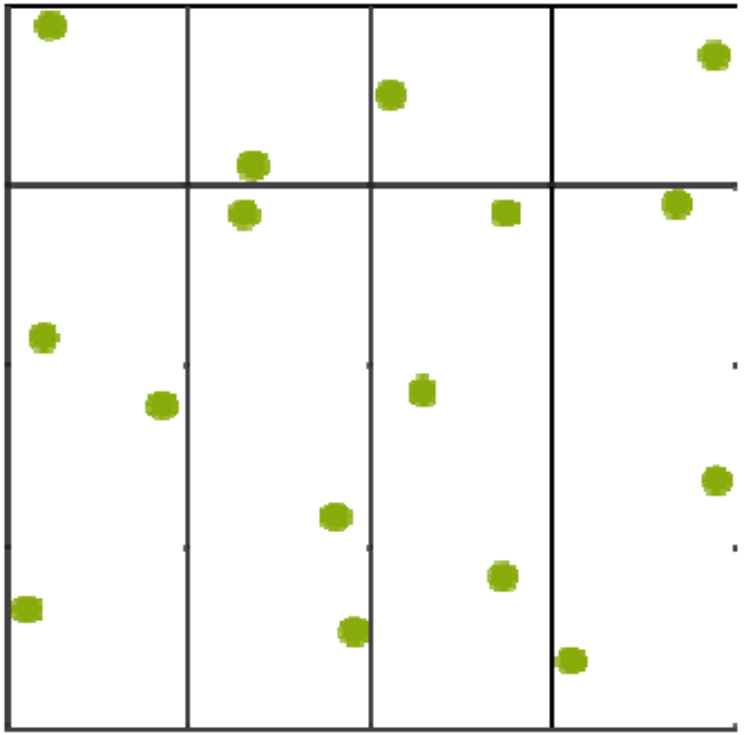
Jittering

- ***Jittering*** è un metodo di campionamento che ci permette "di nascondere il banding con rumore ad alta frequenza
- Il sistema visivo umano "reagisce meglio" ad un rumore uniforme che al banding
- Il Jittering consiste nel considerare inizialmente i campioni uniformi (su una griglia) e perturbare causalmente le posizioni dei campioni.
- La griglia dei campioni *jittered* è calcolata inizialmente ed utilizzata per tutti i pixel.
- Poichè vogliamo avere un intorno a *simmetria circolare*, le coordinate [0,1]x[0,1] della griglia vengono rimappate su un disco utilizzando le coordinate polari.

Jittering



Uniform Samples



Jittered Samples

Implementazione

- Conviene precomputare varie matrici di jittering ed immagazzinare tutti gli offset in una texture 3D da utilizzare nel fragment shader
- La variabile predefinita gl_FragCoord (nel fragment shader) rimanda le coordinate del frammento in fase di rasterizzazione
- Possiamo usare tali coordinate per indicizzare la texture 3D dei samples (in ***r*** memorizzo la sequenza dei samples, in ***s*** e ***t*** faccio un tiling dello screen space)

Ottimizzazione

- Con un precampionamento della shadow map si può determinare se un pixel è completamente in ombra oppure completamente illuminato → in questo modo si può ridurre il numero di pixel su cui valutare il contributo di soft shadow (***branching***)
- Si deve aver cura di non calcolare il contributo di soft-shadow sui pixel che appartengono ad una faccia che non riceve luce (back-face)

Ottimizzazione

Senza ottimizzazione

Con ottimizzazione

I pixel rossi sono i pixels sui quali viene sovracampionata la shadow map per ottenere la soft shadow

Conclusioni

- Abbiamo visto una tecnica semplice ed efficace per realizzare soft shadows in modo approssimato
- La tecnica può essere implementata sull'hardware grafico grazie al *branching* negli shaders
- Il principio utilizzato da questa tecnica può essere utilizzato anche per altri effetti come il motion blur e il depth-of-field.

Domande?