

**Ombrage**

# Ombre et lumière

- ▶ Les ombres augmentent le réalisme



Zaxxon (1982)

Cry Engine



# Ombre et lumière

- ▶ Les ombres augmentent le réalisme
- ▶ Les ombres aident à percevoir
  - des objets cachés



© 2003 - Artis



© 2003 - Artis

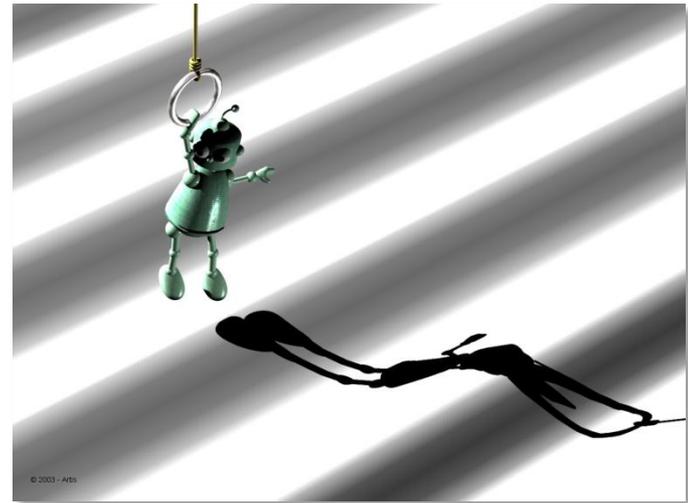
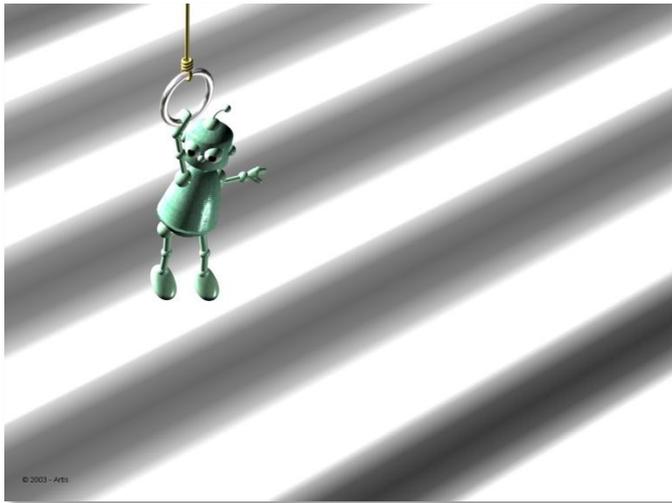
# Ombre et lumière

- ▶ Les ombres augmentent le réalisme
- ▶ Les ombres aident à percevoir
  - des objets cachés
  - le positionnement relatif des objets



# Ombre et lumière

- ▶ Les ombres augmentent le réalisme
- ▶ Les ombres aident à percevoir
  - des objets cachés
  - le positionnement relatif des objets
  - la forme des objets



# Ombre et lumière

- ▶ Contraintes pour les ombres en temps-réel
  - Lampes Dynamiques
  - *Shadow Casters* Dynamiques
  - *Shadow Receivers* Dynamiques

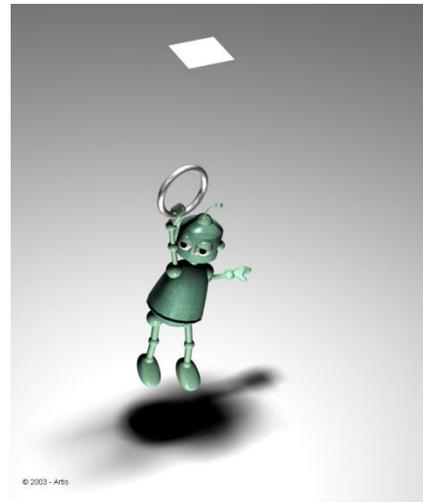
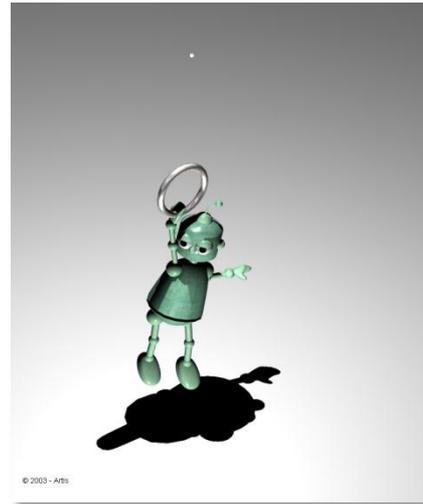


Doom 3 (2004)

# Ombre et lumière

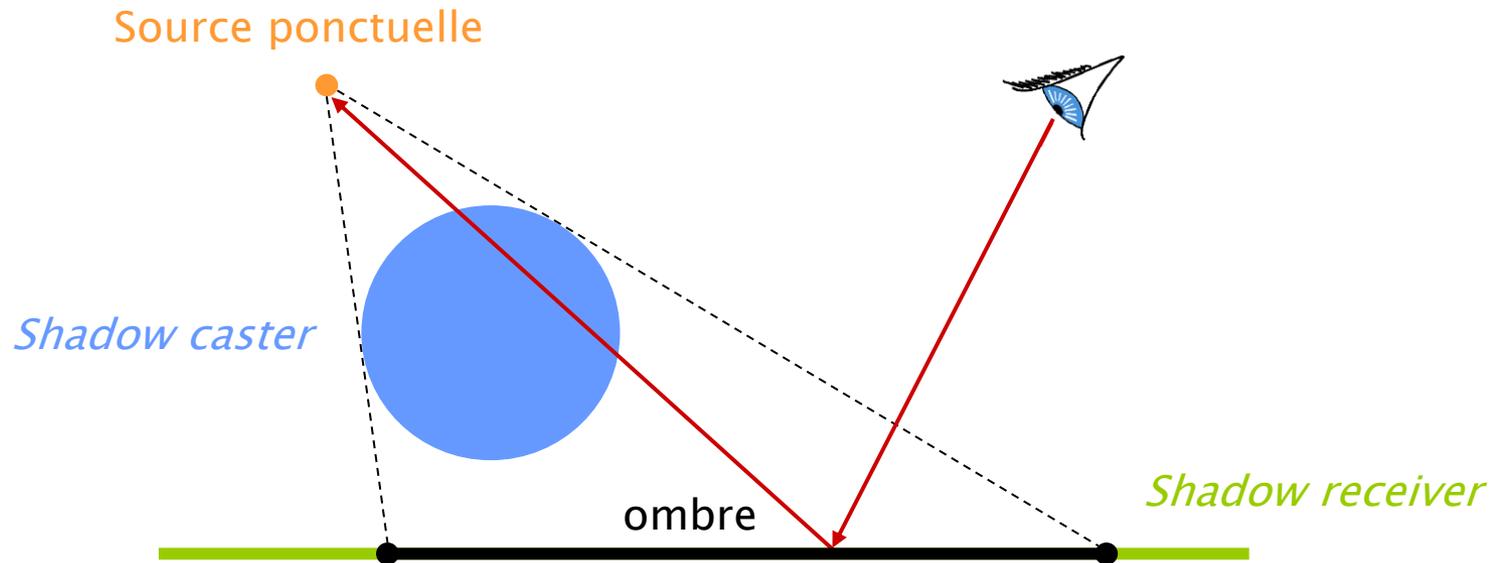
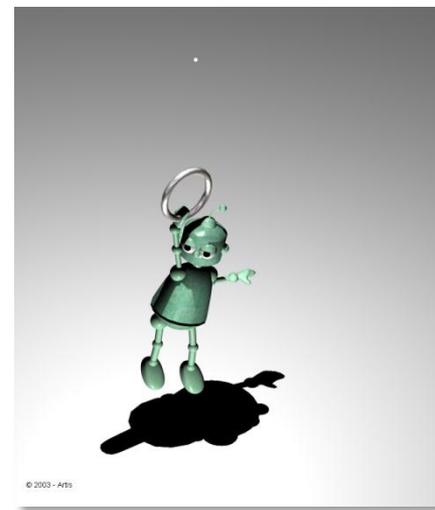
## ▶ Type d'ombres

- **Ombres dures**
  - Hard shadows
  - Source ponctuelle
  
- **Ombres douces**
  - Soft shadows
  - Source étendue



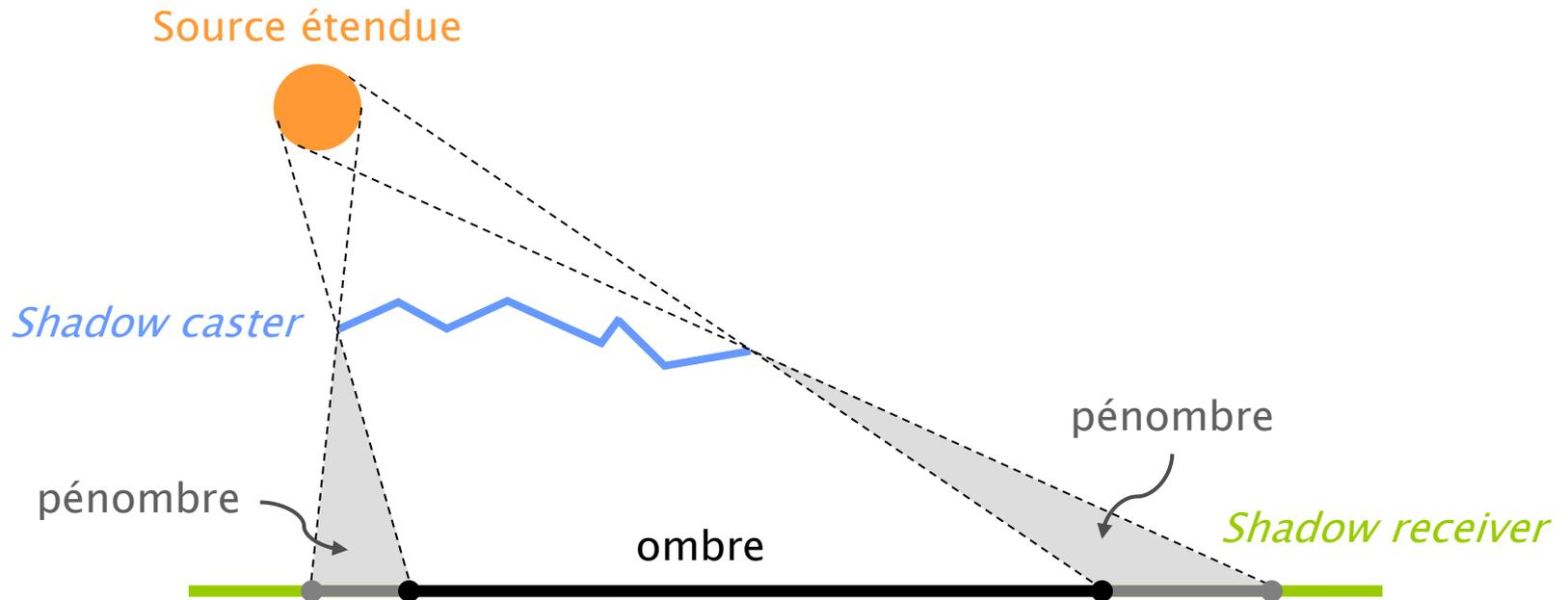
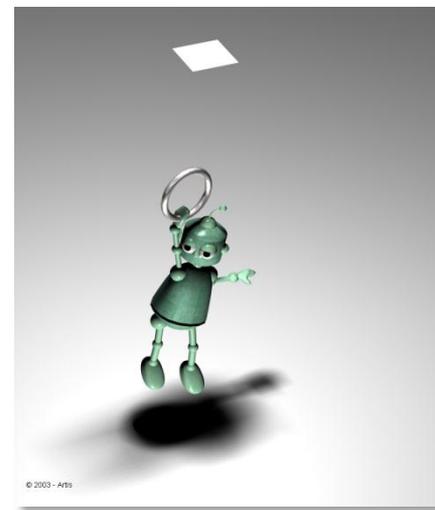
# Ombre dure

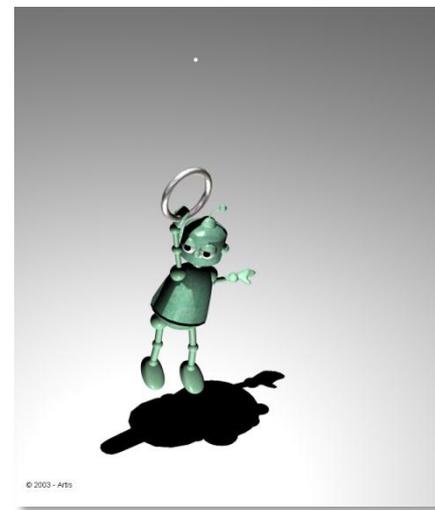
- ▶ Quand la source est ponctuelle
- ▶ Un point est dans l'ombre s'il ne voit pas la source



# Ombre douce

- ▶ 3 zones :
  - Ombre : source totalement cachée
  - Pénombre : source partiellement cachée
  - Eclairée : source totalement visible

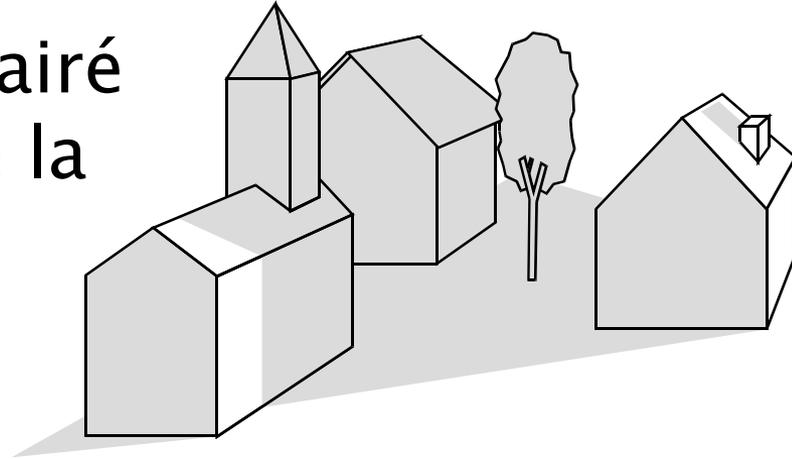




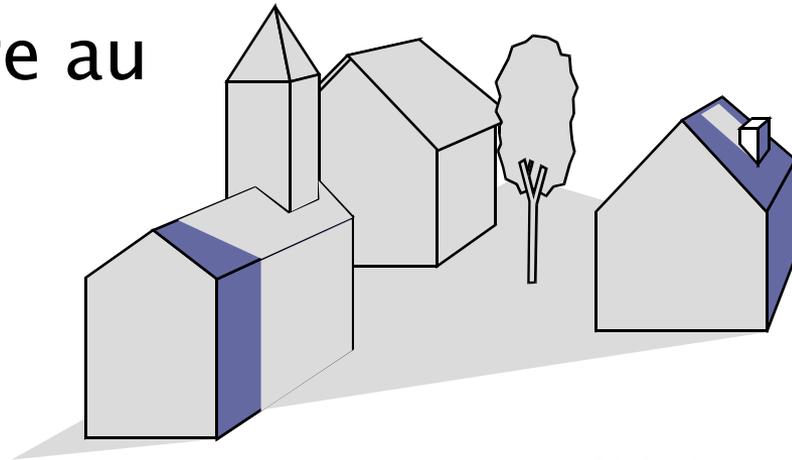
# Calcul des ombres dures

# Dualité ombre/vue

- ▶ Un point est éclairé s'il est visible de la source

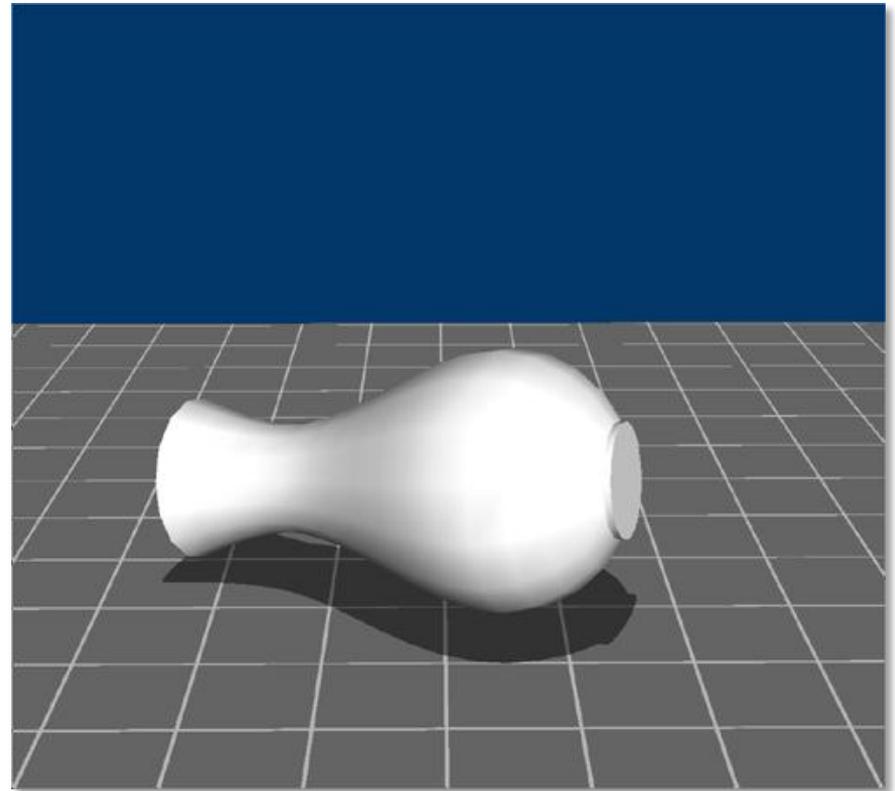
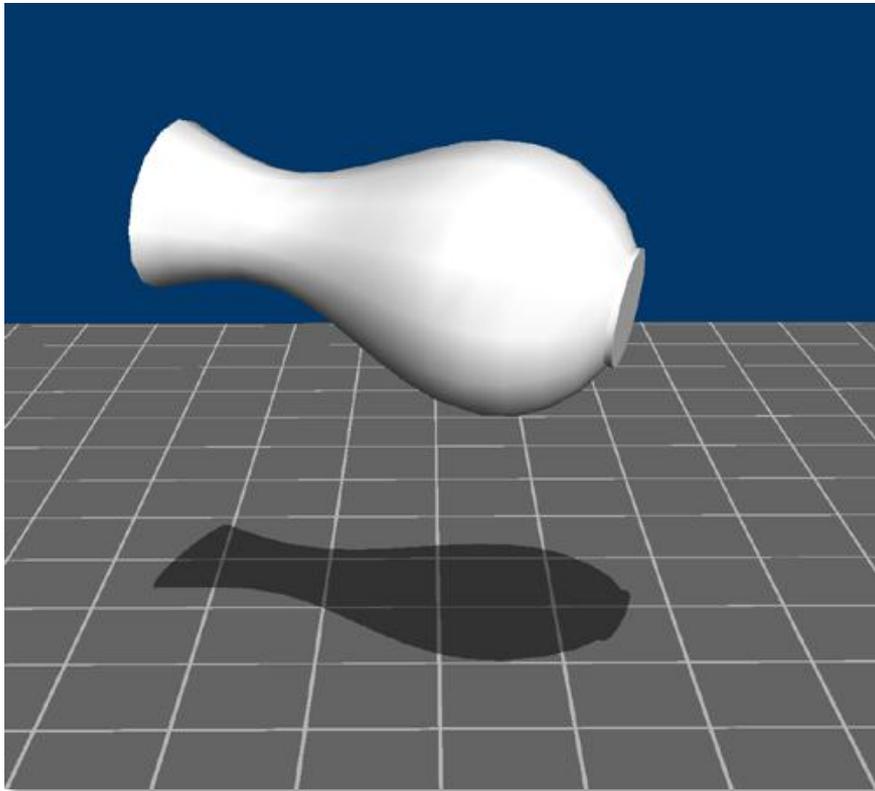


- ▶ Le calcul des ombres est donc similaire au calcul d'une vue



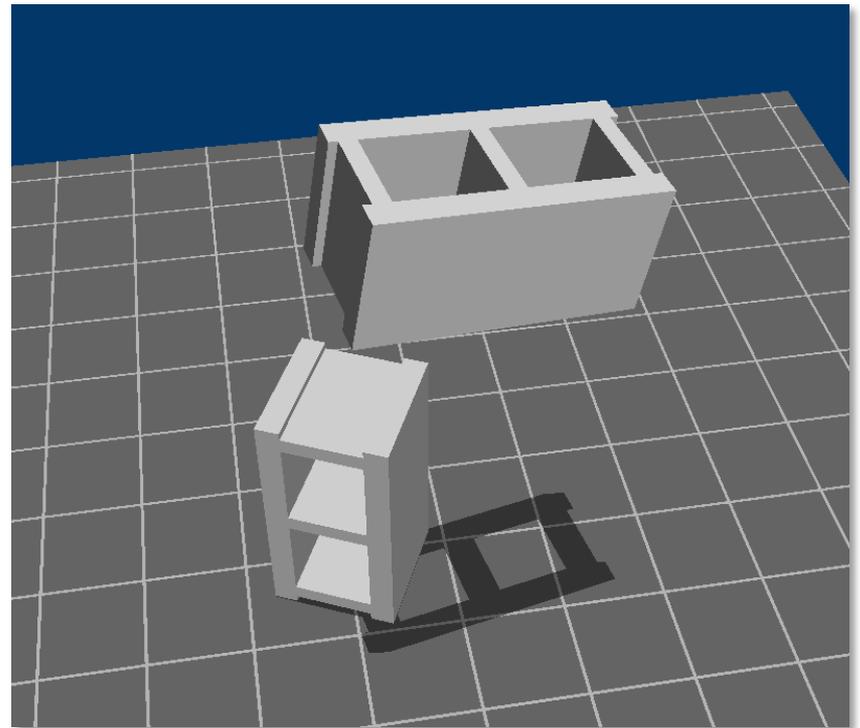
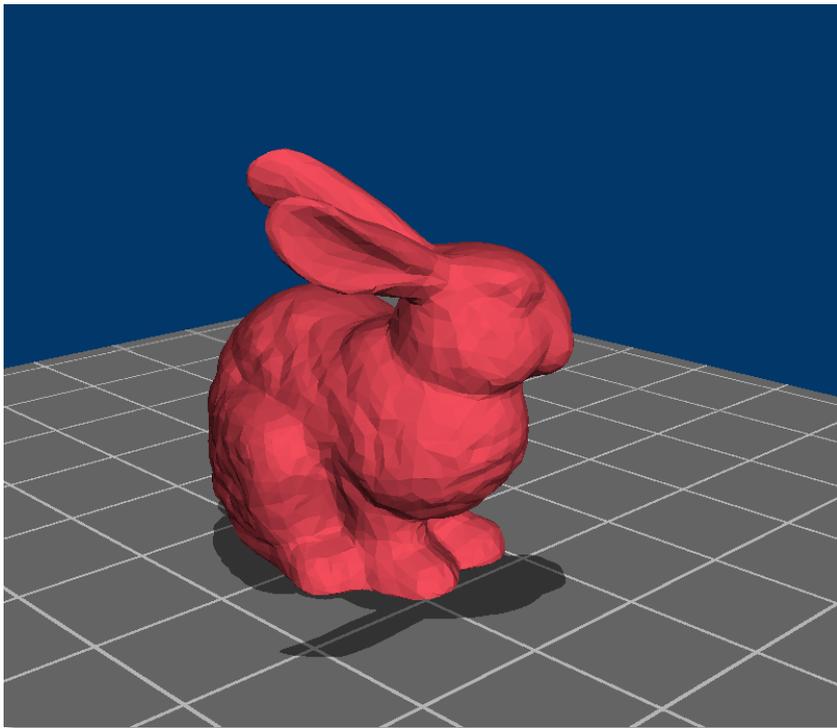
# Ombres planes

- ▶ Dessiner les primitives une seconde fois projetées sur le sol



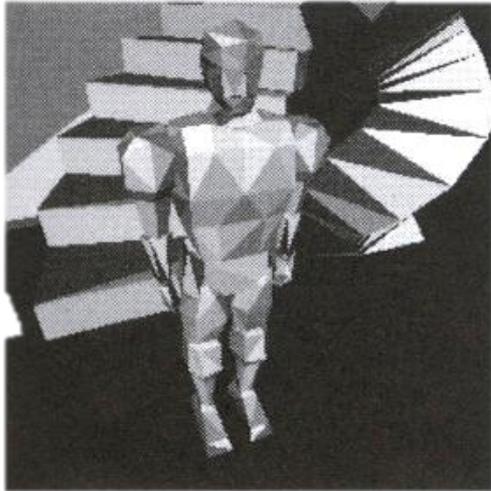
# Ombres planes +/-

- + Simple et efficace
- Pas d'auto ombrage, pas d'ombres sur des surfaces courbes, sur d'autres objets

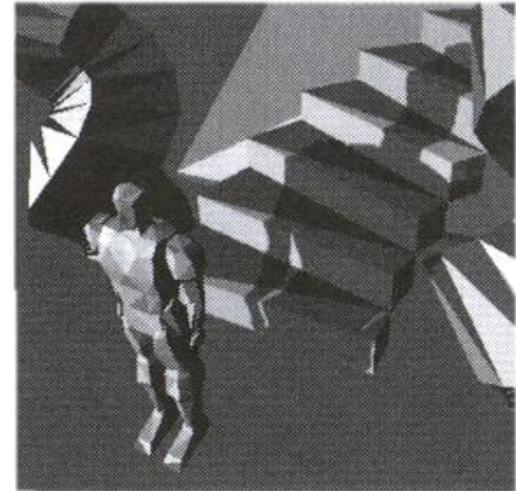


# Utilisation des textures

- ▶ Séparer obstacle et récepteur
- ▶ Calculer une image de l'obstacle vu de la source
- ▶ L'utiliser comme texture sur le récepteur



Vues de la source



Vue de l'œil

# Méthodes actuelles



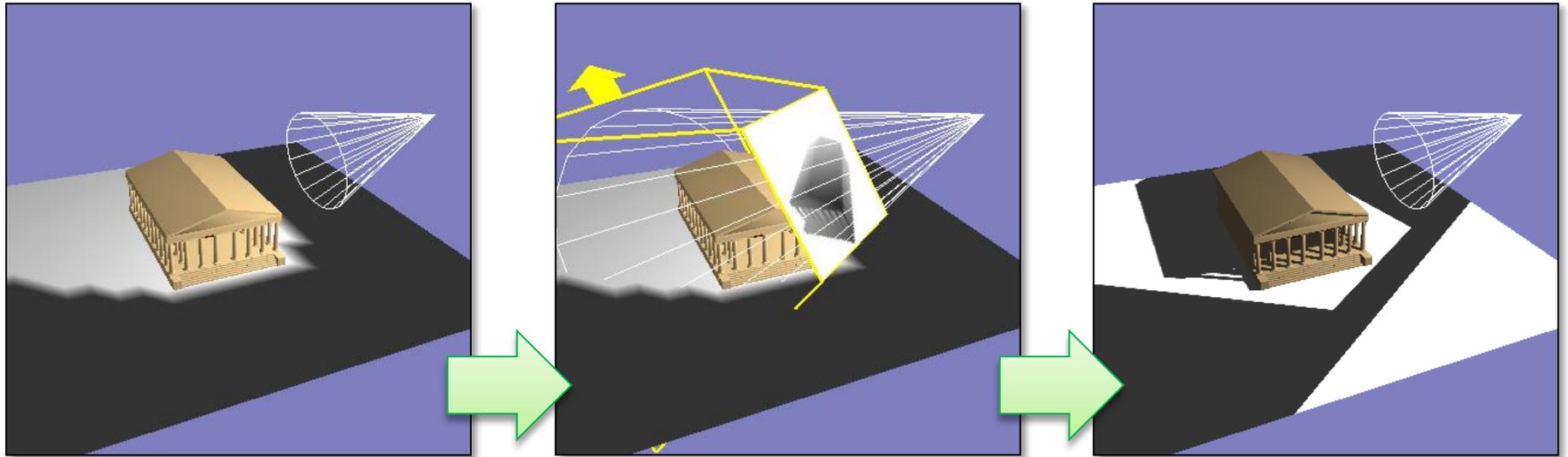
## ▶ Shadow Maps

- Approche “image”

## ▶ Shadow Volumes

- Approche “objet”

# Shadow maps



1. Rendu *offscreen* **depuis la lampe**
  - conserver le z-buffer dans un texture
2. Rendu **depuis la caméra**
  - Comparer le z courant avec la profondeur dans la texture (*projective texture lookup*)
  - Moduler l'éclairage (diffus et spéculaire) en fonction du test binaire de visibilité

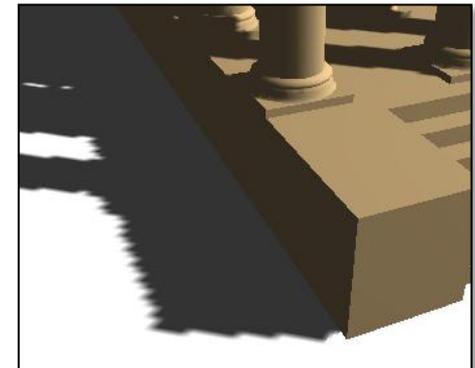
# Shadow maps : bilan

## ▶ Avantages

- Simple à implémenter
- Fonctionne pour des scènes quelconques
- Coût indépendant de la complexité de la scène

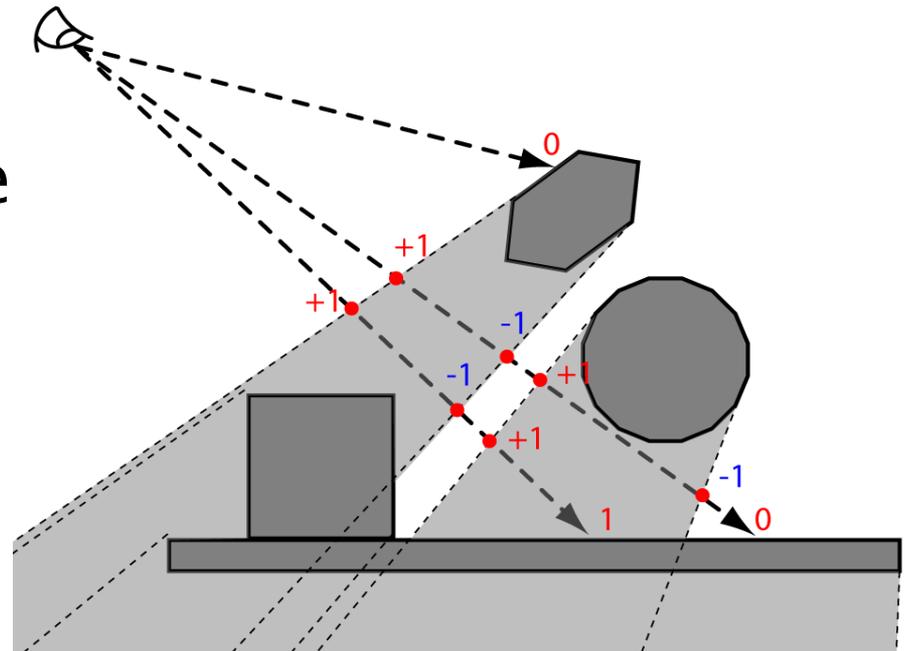
## ▶ Inconvénients

- Plusieurs ( $\geq 2$ ) rendus de la scène
- Gestion des sources omni-directionnelles
- Problème d'aliasage
  - augmenter la résolution de la shadow map ne suffit pas toujours (objet vu sous un angle rasant)



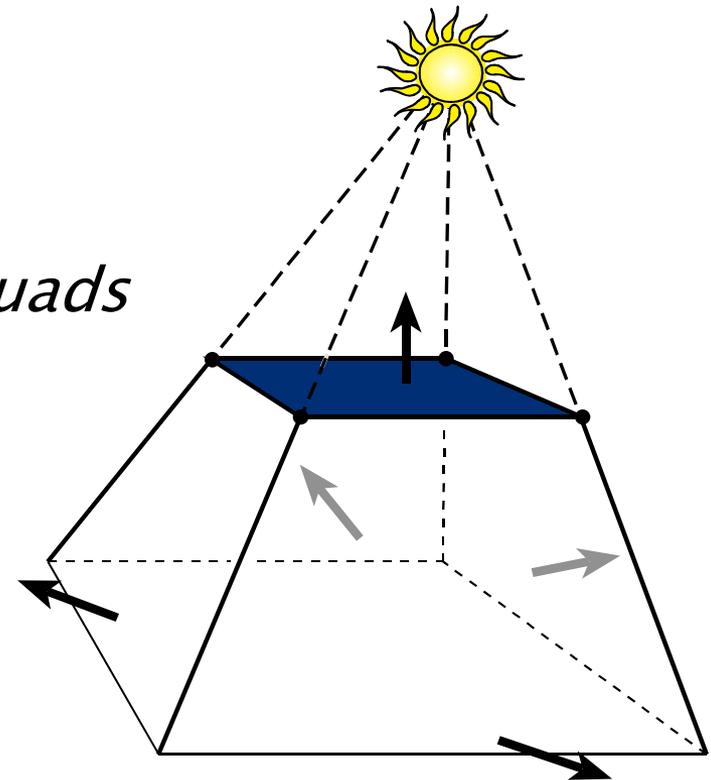
# Shadow Volumes : principe

1. Pour chaque *shadow casters*, construire un **volume d'ombre**
2. Pour chaque fragment dessiné, **compter** combien de fois on entre/sort d'un volume
  - > 0 : dans l'ombre
  - = 0 : dans la lumière



# Shadow Volumes : comment ?

- ▶ Construire les volumes d'ombres
  - trouver la silhouette des objets vus depuis la source
  - construire des *quads* infinis s'appuyant
    - sur la source
    - sur chaque arête de silhouette
- ▶ Compter les entrées/sorties
  - utiliser le *stencil buffer*
  - Traiter les différents *shadow quads* selon leur orientation



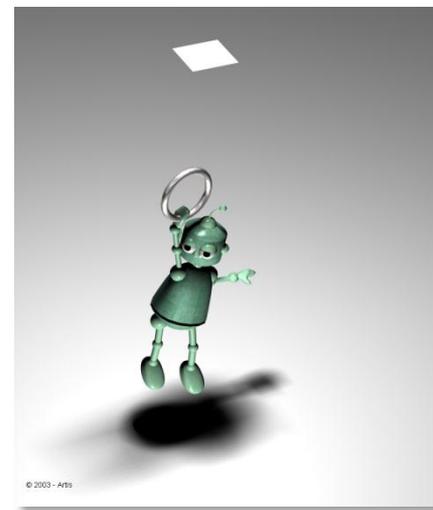
# Shadow volumes: bilan

## ▶ **Avantages :**

- ombres précises
- positions quelconques source/caméra
- robuste si bien programmé

## ▶ **Inconvénients :**

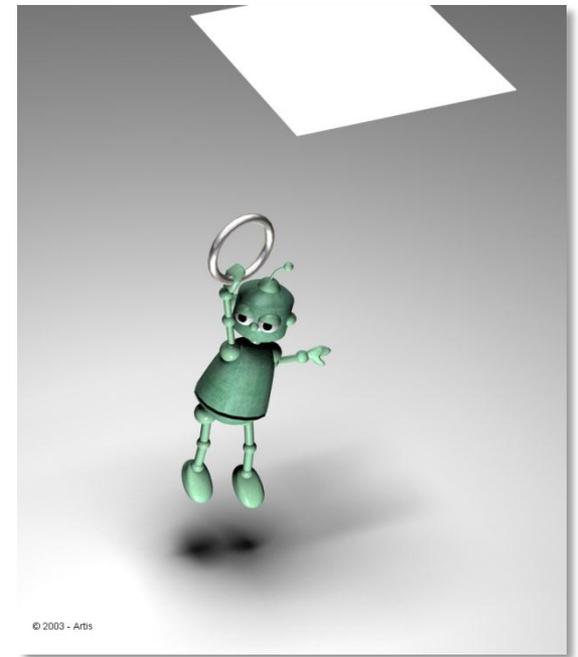
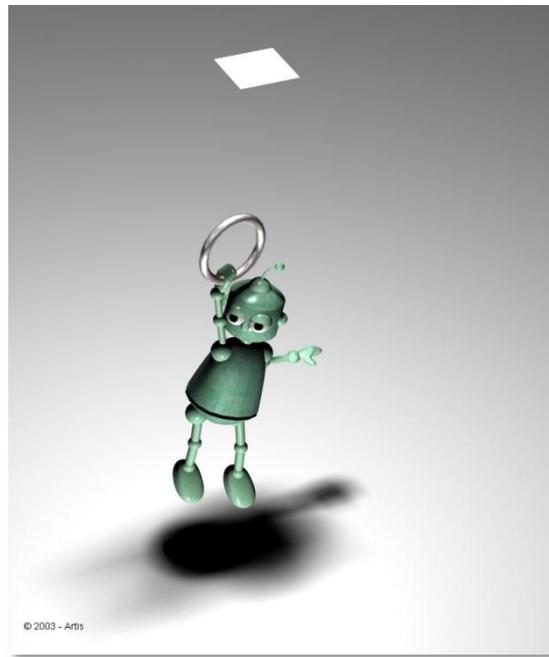
- calcul de la silhouette (sur CPU/GPU)
- scènes bien modélisées préférables
- 2 rendus de la scène + rendu des volumes



# Calcul des ombres douces

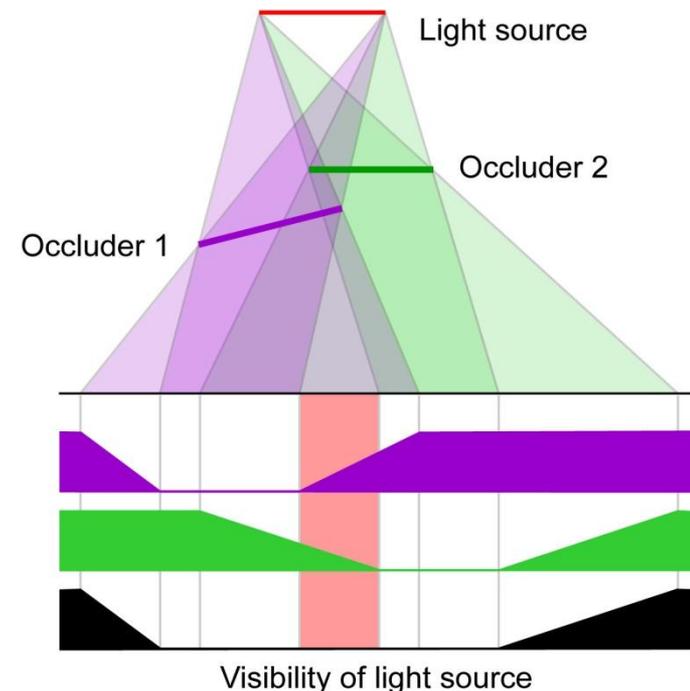
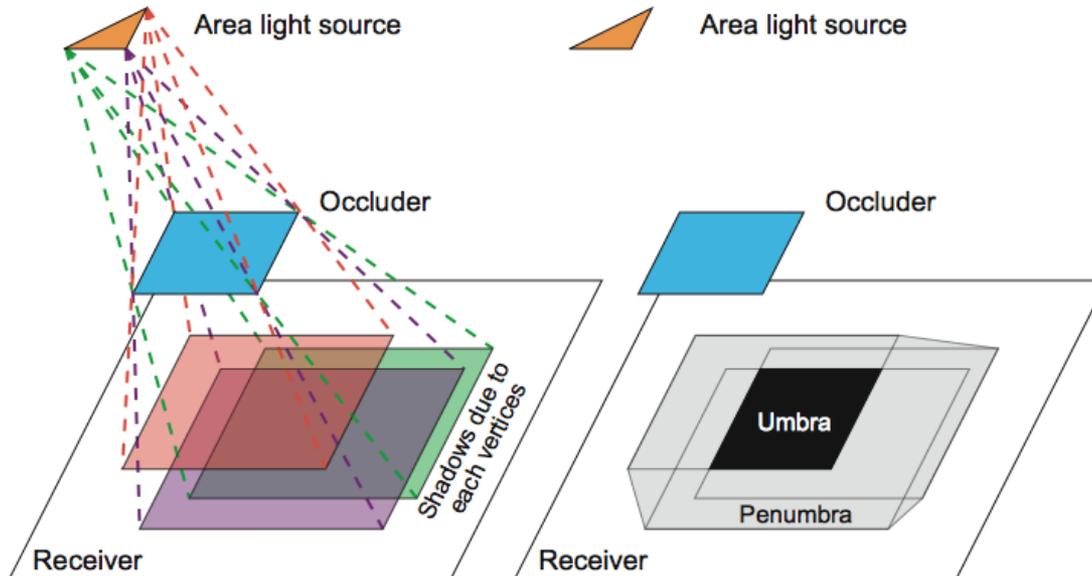
# Ombres douces

- ▶ Algorithmiquement plus compliqué
  - problème de visibilité **point-surface continue**
    - au lieu de point-point discret
    - silhouette ?



# Ombres douces

- ▶ Algorithmiquement plus compliqué
  - problème de visibilité **point-surface continue**
    - au lieu de point-point discret
    - silhouette ?
  - ombre de la somme  $\neq$  somme des ombres
    - Si A cache 50% et B cache 50%, A+B ne cache pas 100%



# Ombres douces

- ▶ Algorithmiquement plus compliqué
  - problème de visibilité **point-surface continue**
    - au lieu de point-point discret
    - silhouette ?
  - ombre de la somme  $\neq$  somme des ombres
    - Si A cache 50% et B cache 50%, A+B ne cache pas 100%
- ▶ Il existe de nombreuses méthodes
  - Plus ou moins **exactes**
    - Approximation géométrique des *shadow casters*
    - Pré-calculs (*Precomputed Radiance Transfert*)
    - Sélection manuelle des *shadow casters* / *receivers*
  - plus ou moins **rapides**

# Ombres douces par *sampling*

- ▶ Accumulation d'ombres :
  - calculer plusieurs ombres ponctuelles
  - additionner et moyenner les résultats
  - *accumulation buffer*
  - nombre d'échantillons élevés
    - temps de calcul multiplié par # échantillons



4 échantillons



1024 échantillons

# Extension du shadow volume

- ▶ Pour chaque arête de silhouette :
  - calculer volume englobant la pénombre
  - pour chaque pixel dans ce volume
    - calculer coefficient d'atténuation
- ▶ Beau, réaliste mais très couteux

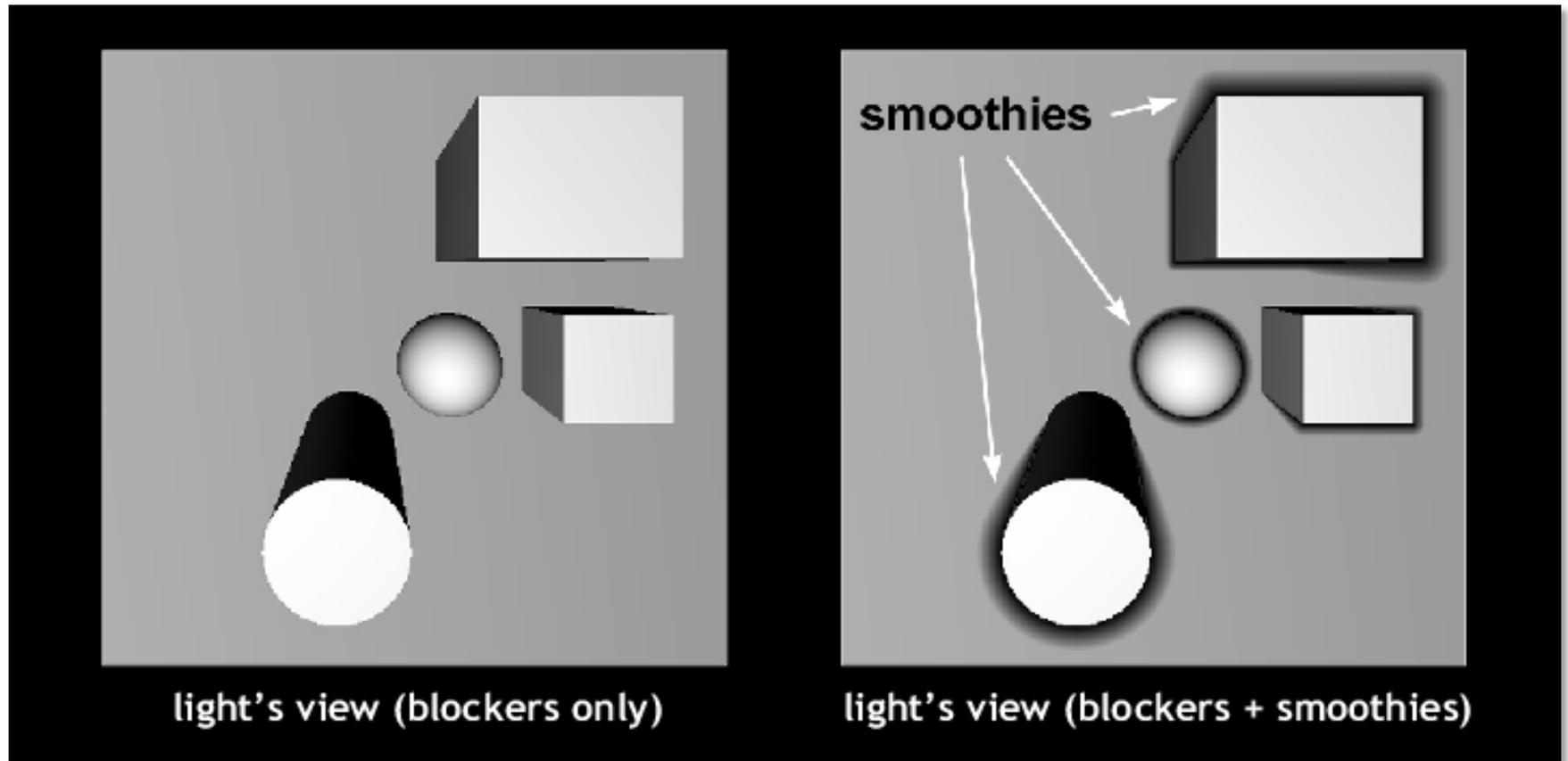


Penumbra wedges [Sig03] *U. Assarson, T. Möller*

# Mix objet / image

Rendering Fake Soft Shadows with Smoothies [SoR03]

*E. Chan, F. Durand*



# Extension des shadow maps

- ▶ *Percentage Close Filtering* (PCF)
  - Échantillonner et filtrer la *shadow map*
  - Partiellement accéléré par le GPU (noyau 2x2)



1 échantillon



noyau 9x9



noyau 17x17

# Extension des shadow maps

## ▶ *Percentage Closer Soft Shadows* (PCSS)

[Fernando 05]

- Calcul de la taille du noyau par échantillonnage ponctuelle de la source lumineuse
- Filtrage par PCF (ou l'une de ses extensions)



# Extension des shadow maps

- ▶ *Percentage Closer Soft Shadows* (PCSS)  
[Fernando 05]



PCSS



PCF

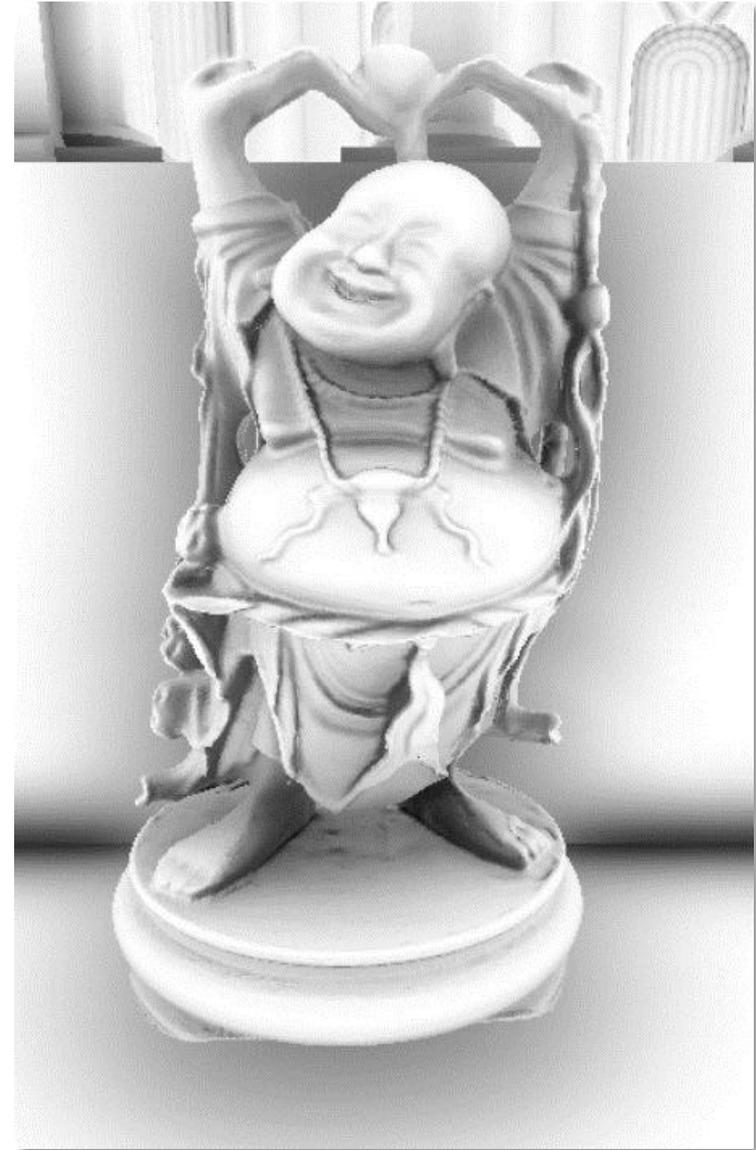
Hellgate: London (2007)



# Ambiant occlusion

# Motivation

- ▶ Approximation de l'occlusion sous un éclairage extérieur
  - ou {
    - terme ambiant « intelligent » tenant compte de la visibilité
    - terme diffus avec une distribution complexe de lampes
- ▶ Indices perceptuels de la profondeur, courbure et proximité spatiale

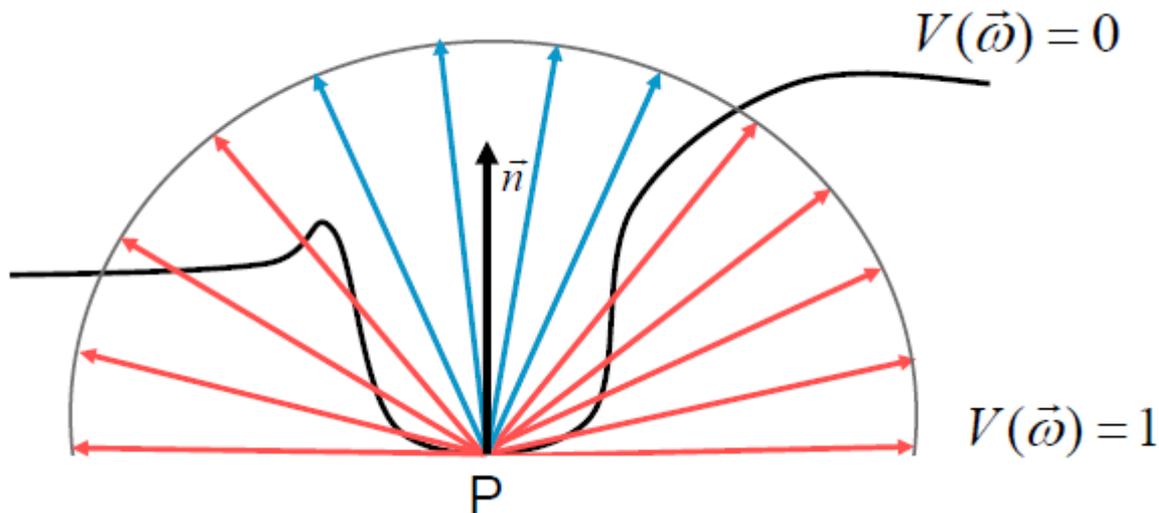


# Définition

- ▶ Intégrale de la visibilité sur l'hémisphère  $\Omega$  de rayon R :

$$A_P(\vec{n}) = \frac{1}{\pi} \int_{\Omega} V_P(\vec{\omega})(\vec{n} \cdot \vec{\omega}) d\omega$$

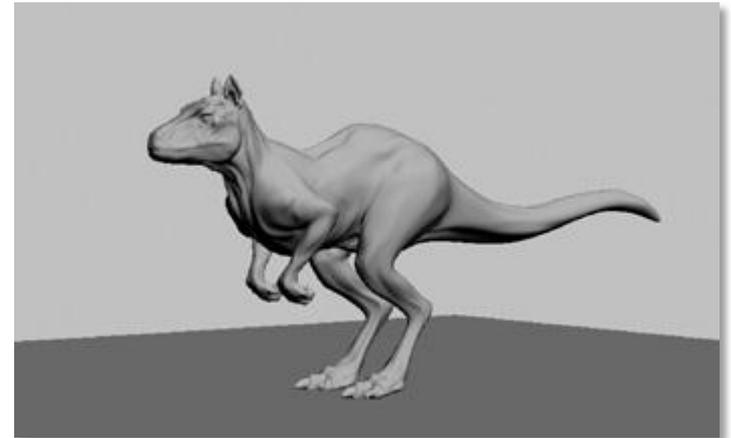
- Pondération par le cosinus  $\Rightarrow$  éclairage diffus
- Généralement, atténuation selon la distance à P



# Calcul de l'intégral

GPU Gems, chap 17

- ▶ Échantillonnage
  - Pré-calcul par *ray-casting*
  - Stockage dans une texture
- + Pas de surcoût lors du rendu
- Pré-calcul lent
- Scène statique



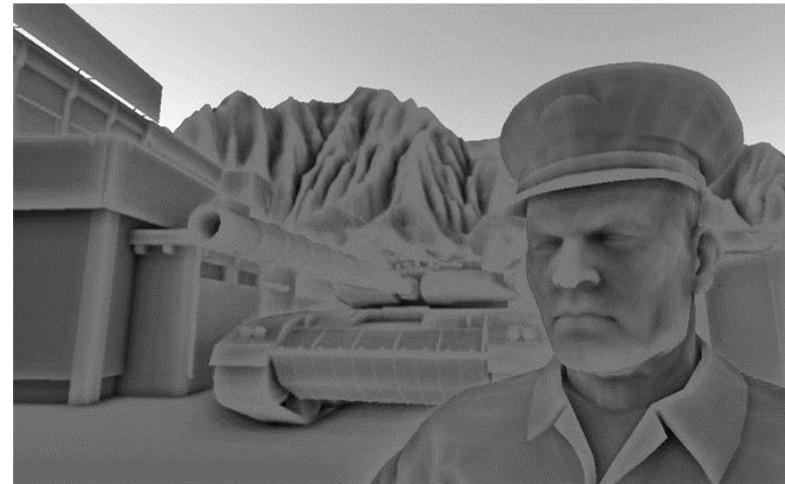
Diffus



Diffus + AO

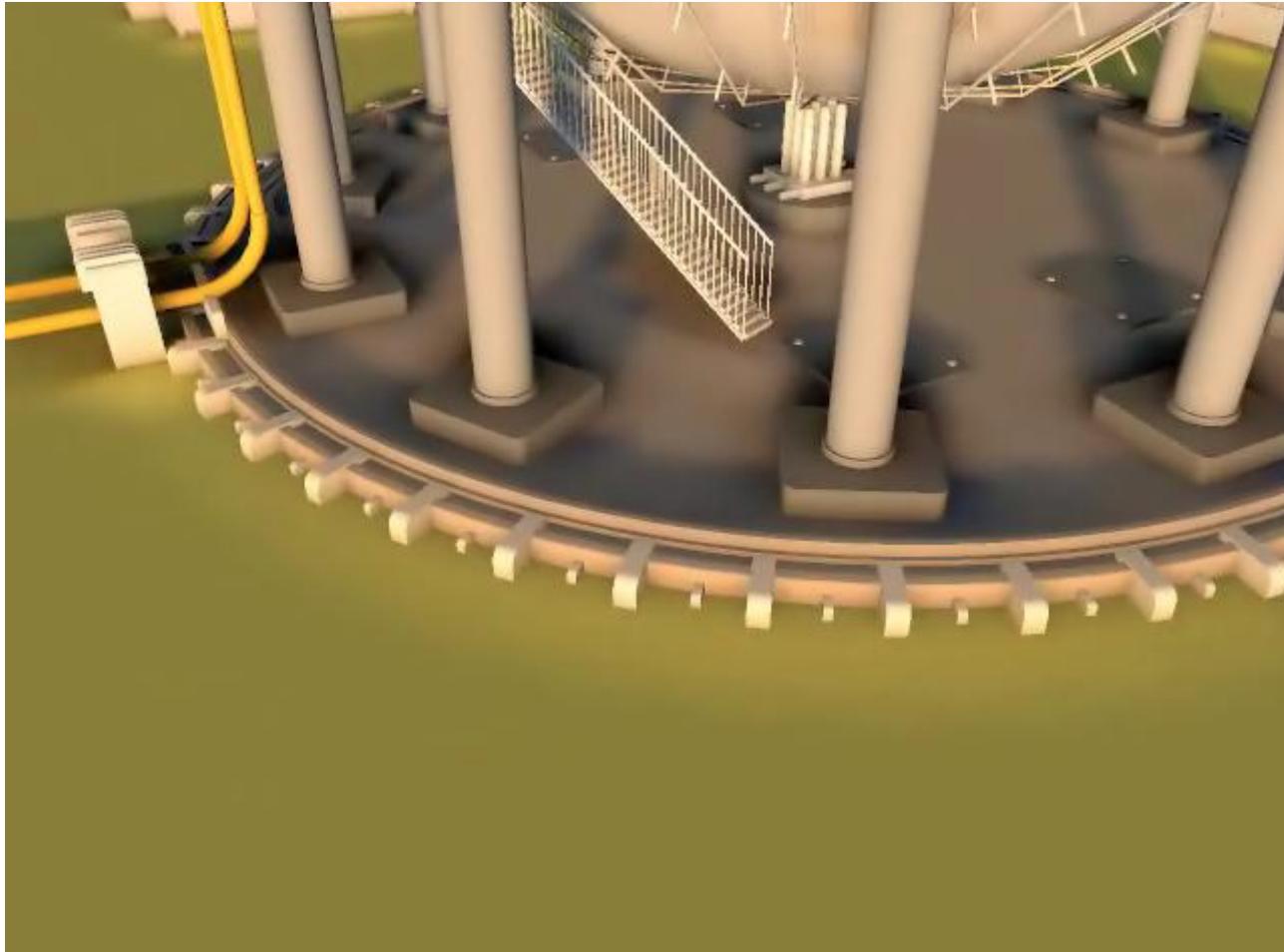
# Calcul de l'intégral

- ▶ *Screen-Space Ambient Occlusion (SSAO)*
  - Utilisation du *depth buffer* comme approximation de la scène (carte de hauteur)
  - Échantillonnage aléatoire grossier sur GPU de l'hémisphère pour chaque pixel
  - Filtrage pour réduire le bruit
- + Indépendant de la complexité de la scène
- + Pas de pré-calcul
- + Scène dynamique
- Surcoût lors du rendu



Cry Engine 2

# Au-delà du SSAO



Approximating Dynamic Global Illumination in Image Space  
*Ritschel et al. 2009*