IFT3355: Infographie
Sujet 5: `shading` 1
(illumination *locale* 1)

Derek Nowrouzezahrai
Département d'informatique et de recherche opérationelle
Université de Montréal

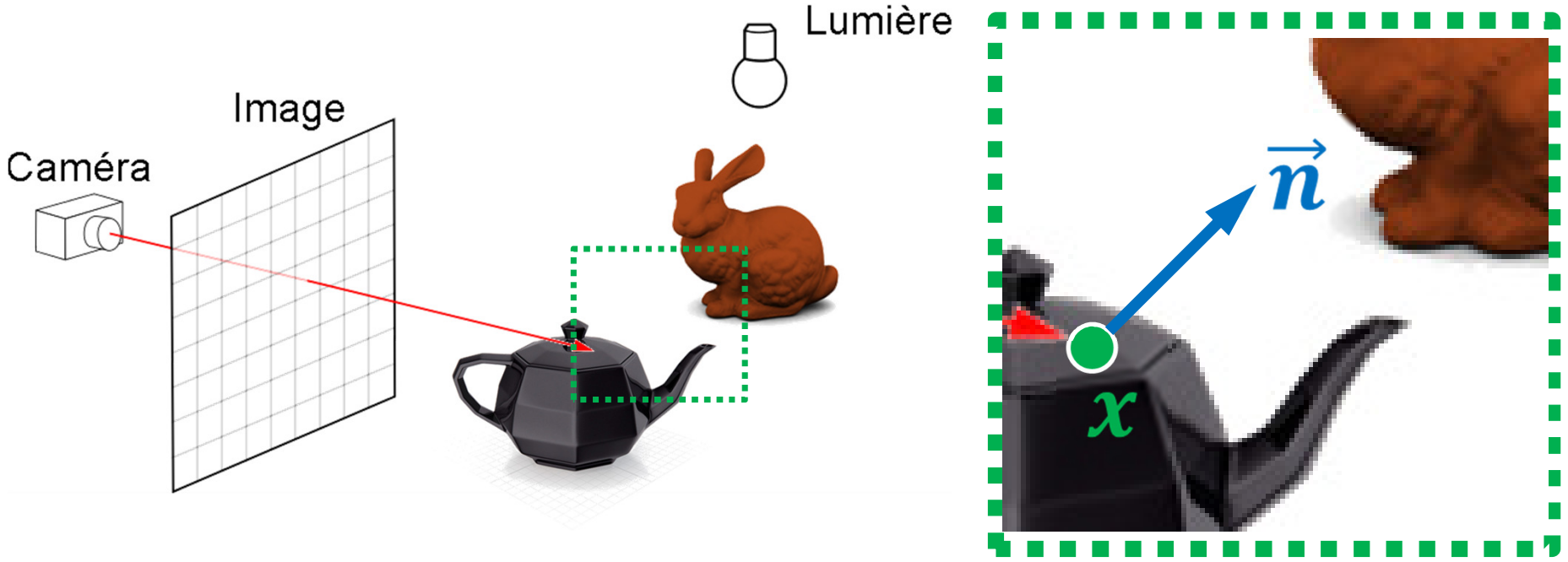
Survol: illumination locale

- On vient de voir comment déterminer la visibilité vers l'oeil en utilisant la **rastérisation** ou le **lancer de rayons**
  - Maintenant, on doit calculer la *couleur* à chaque pixel

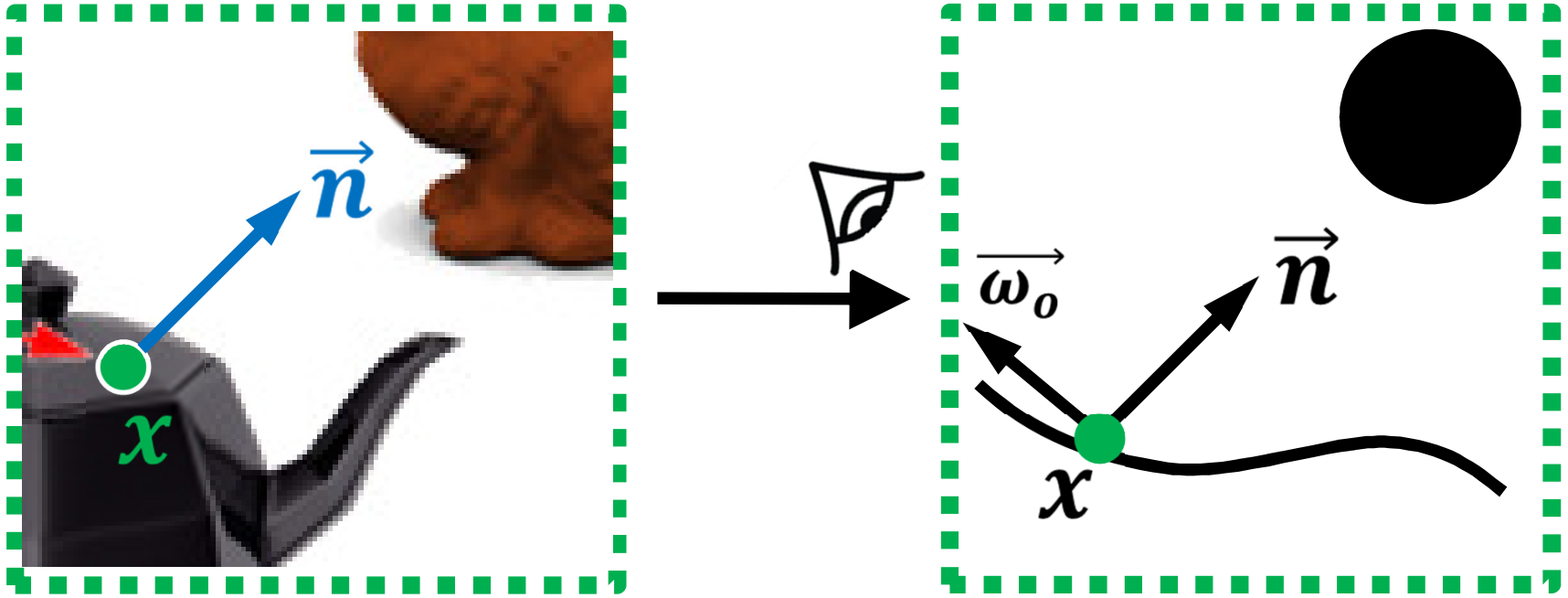
1. Illumination locale 1: Après quelques définitions et motivations, nous présenterons un modèle d'illumination locale pseudo-réaliste qui peut être appliqué dans un contexte de **lancer de rayons** (*ou de rastérisation*)
2. Illumination locale 2: Nous discuterons comment utiliser la méthode de **lancer de rayons** pour simuler les ombres ponctuelles, la réflexion et la réfraction
3. Illumination locale 3: Nous retournerons brièvement pour discuter le calcul du shading et quelques techniques pour simuler les ombres ponctuelles dans un système de **rastérisation**
4. Illumination globale: ces méthodes s'appuieront presque exclusivement dans un contexte de **lancer de rayons**

Configuration Canonique

- Pour nos discussions de shading nous supposerons:
  - qu'on considère chaque pixel d'une manière indépendante
  - que nous avons déjà déterminé la visibilité du caméra
  - nous avons tous les propriétés de l'objet le plus proche
  - nous avons tous les propriétés des lumières dans la scène

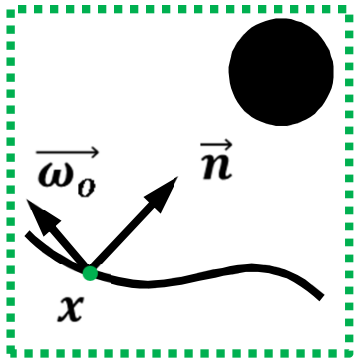


Configuration Canonique



- Pour plus facilement considérer plusieurs modèles de shading on peut simplifier/remplacer ce diagramme avec:
  - la surface avec le point x que nous voulons “shader”
  - la normale à x
  - le direction de vue
  - des abstractions des autres objets

Illumination



Si on considère des modèles d'illumination *réalistes* ou *pseudo-réalistes*, le shading à $\boldsymbol{x}$ dépend sur:

- les propriétés de la surface de l'objet à $\boldsymbol{x}$
  - géométrie, réflectance/matériaux, etc.
- les propriétés des lumières dans la scène
- et potentiellement les propriétés (et le shading!) sur tous les autres objets

Illumination locale

Illumination globale

Définitions

- Illumination
  - définit le transport de la lumière dans la scène
- *Shading*
  - considère des valeurs de l'illumination en des points sur une surface et interpole ces valeurs pour des points intermédiaires d'après le modèle de *shading*
  - *flat, Gouraud, Phong*
  - Typiquement en lancer de rayons, le *shading* en un point correspond à son illumination

Locale vs. Globale: autres interpretations

- Illumination locale
  - ne considère que la contribution directe des sources de lumière
- Illumination globale
  - considère la contribution directe des sources de lumière ainsi que la lumière interréfléchie entre les surfaces de la scène

Lumière

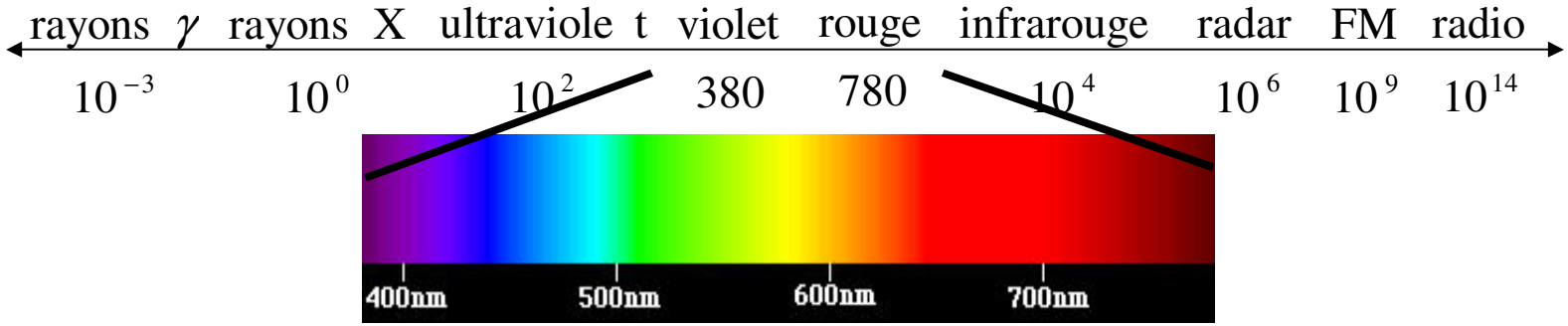
- La lumière est émise par les sources de lumière et interagit dans la scène
- Nature duale
  - onde
  - particule

Photon

- Un photon transporte une certaine énergie à une longueur d'onde donnée (couleur)
- A chaque interaction (réflexion, réfraction, absorption), un photon peut changer sa direction et/ou sa couleur (changement spatial et spectral)
- Si un photon possède une longueur d'onde entre

$$380 \leq \lambda \leq 770\ \text{nm}$$

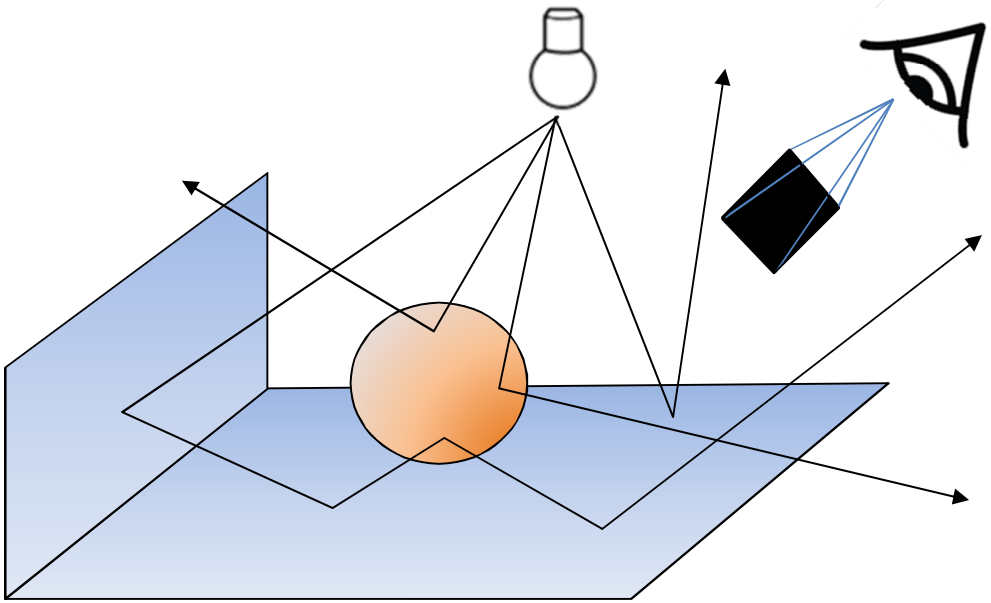
il est alors visible par l'oeil humain



Photon

- Si un photon passe par la position de l'oeil tout en intersectant la fenêtre graphique, sa couleur contribue au pixel qu'il traverse
- Chaque photon sera donc traité par l'optique géométrique, sauf si mentionné autrement

Simuler l'illumination



- L'image parfaite demandera de suivre ainsi une infinité de photons
- Il faut donc simplifier le problème

Modèle simplifié de réflexion locale

- Nous commencerons avec un modèle d'illumination pseudo-réaliste basé sur trois types d'effets:
  - réflexion ambiante
  - réflexion diffuse
  - réflexion spéculaire

Ambiante Diffuse Spéculaire Combinée

Réflexion Ambiante

- Sans interréflexion de la lumière, tout ce qui est dans l'ombre (*umbra*) est **noir**
- Calculer l'interréflexion entre les surfaces est un problème (d'illumination globale) extrêmement complexe
- On simplifie (à outrance?) son résultat en parlant d'une lumière ambiante qui est partout la même, pour n'importe quelle direction
- Toute surface éclairée seulement par une lumière ambiante aura un éclairage uniforme. Cette surface apparaît sans profondeur.

La lumière ambiante

L_a Propriété de scène
Intensité de la lumière ambiante

k_a Propriété de surface
Une surface réfléchit une proportion k_a de la lumière ambiante

$S(\lambda)$ “Couleur” de la surface

$$L = L_a k_a S(\lambda)$$

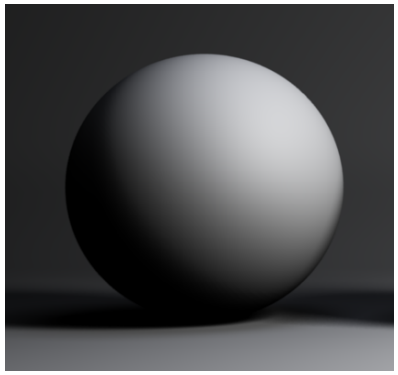
Variations sur la lumière ambiante

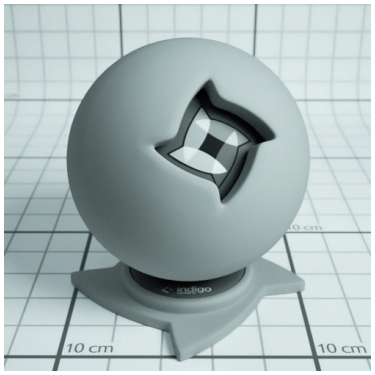
- Ajouter une source de lumière à la position de l'oeil (sans notion de distance)
- Utiliser plusieurs sources de lumière pour remplacer l'illumination du ciel, mais coûteux
- Utiliser la valeur absolue du terme diffus $\left|\vec{N} \cdot \vec{L}\right|$, au lieu de la valeur positive, ce qui correspond au *two-sided lighting*

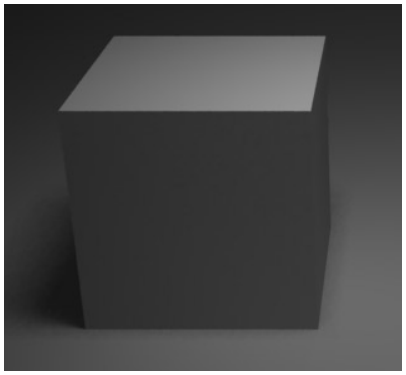
Réflexion diffuse

- Normalement, l'intensité d'un point sur une surface varie dépendant de sa distance à la lumière et son orientation relative à la lumière
- Réflexion égale en intensité

  dans toutes les directions
- Ex : peinture matte, papier, bois sablé

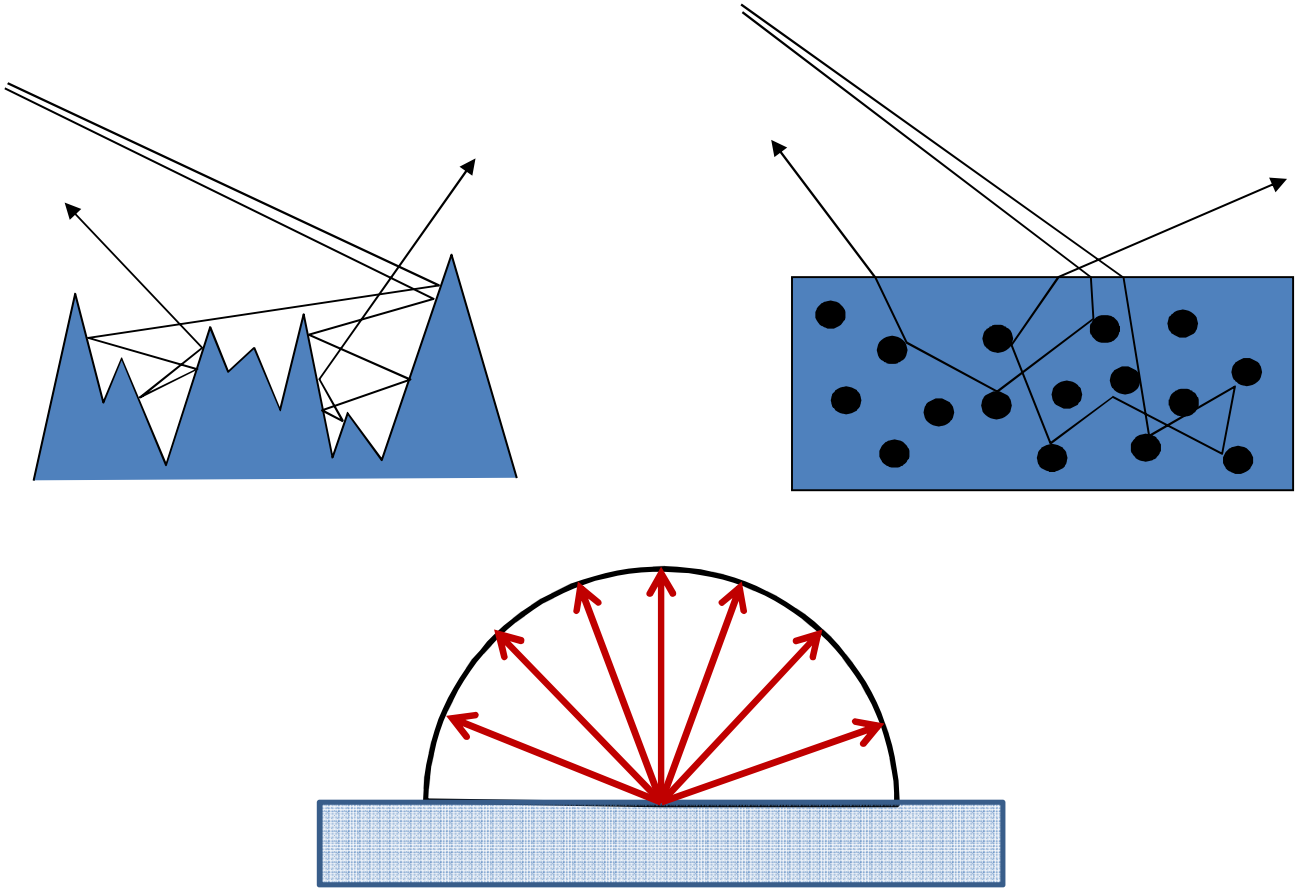






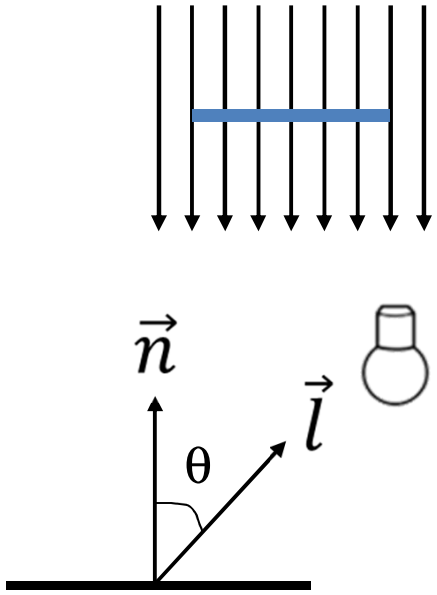
Réflexion diffuse

- Deux modèles pour expliquer ce type de réflexion



La réflexion diffuse

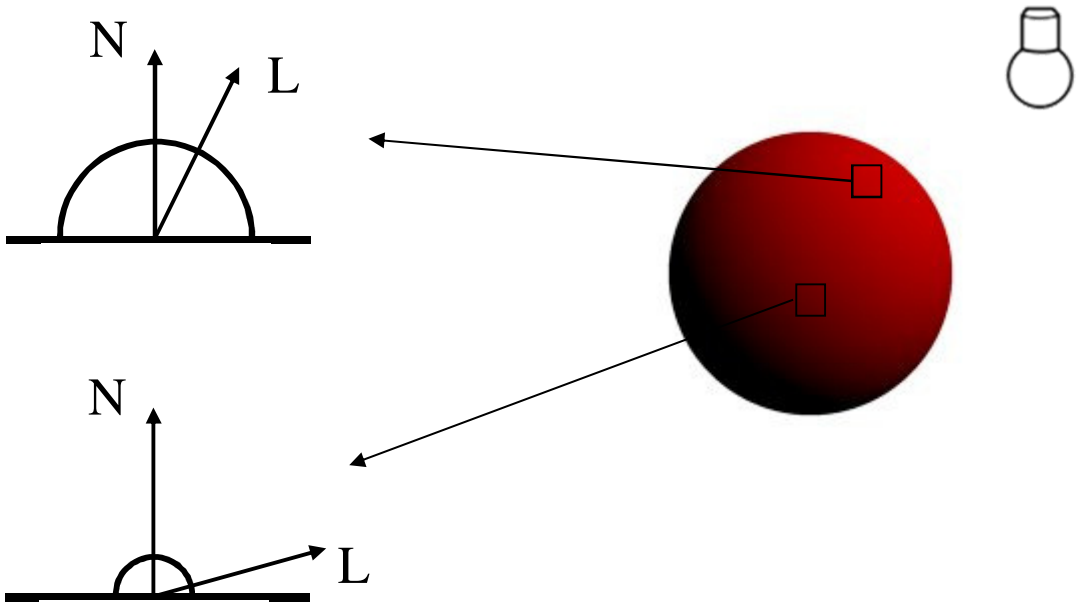
- La réflexion diffuse ou lambertienne est associée à la perception de la *forme* des objets de type *mat*
- Plus la surface fait face à la lumière, plus elle en reçoit



$$\cos\theta = \vec{n} \cdot \vec{l}$$

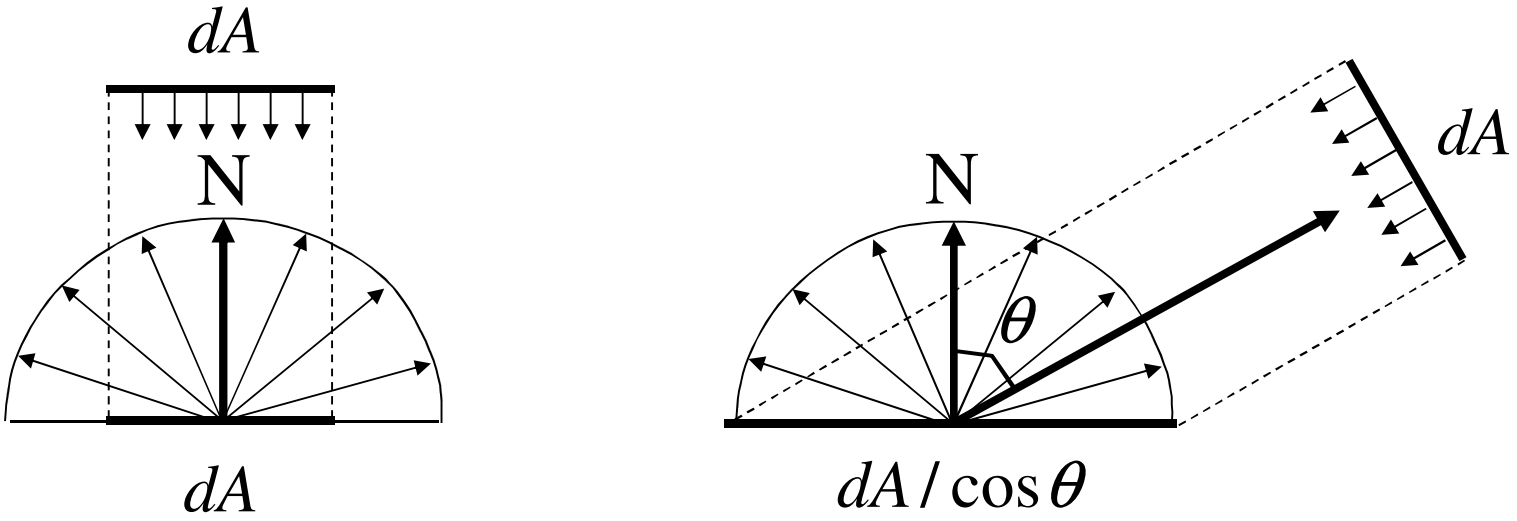
La réflexion diffuse

- La réflexion diffuse redistribue les photons également (en intensité) au-dessus de la surface



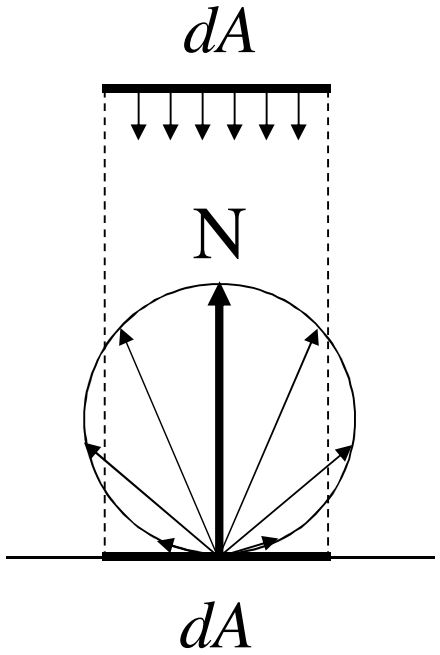
Réflexion lambertienne

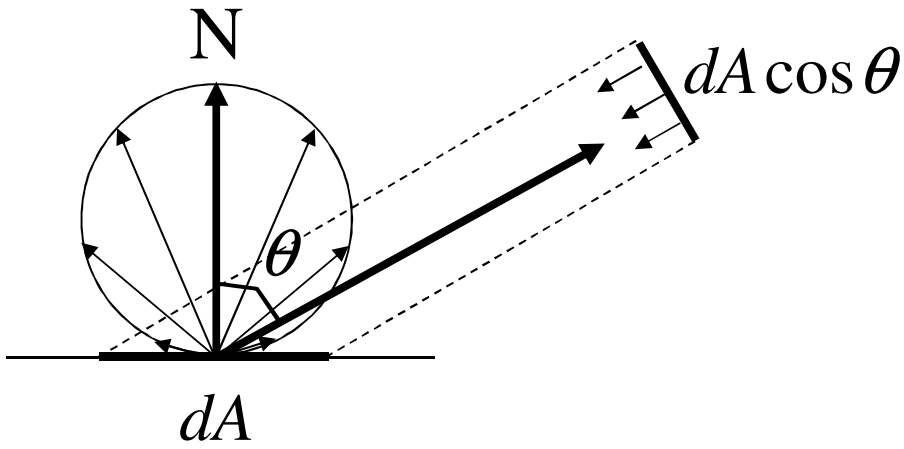
- Une surface est dite lambertienne si la surface apparaît également brillante de n'importe quelle direction au-dessus de cette surface
- La surface illuminée s'accroît comme $dA / \cos\theta$



Réflexion lambertienne

- Ceci est équivalent à dire que la lumière est réfléchie également en intensité dans toutes les directions au-dessus de cette surface





Réflexion lambertienne

L_l intensité de la lumière l

k_d Propriété de surface
Une surface réfléchit une proportion k_d
de la lumière de façon diffuse

$\vec{n} \cdot \vec{l}$ $\cos\theta_L$ où $\vec{n}$ et $\vec{l}$ sont des vecteurs normalisés
on devrait spécifier pour toute orientation de surface $\max(\vec{n} \cdot \vec{l}, 0)$ mais on allège habituellement la formulation à $\vec{n} \cdot \vec{l}$

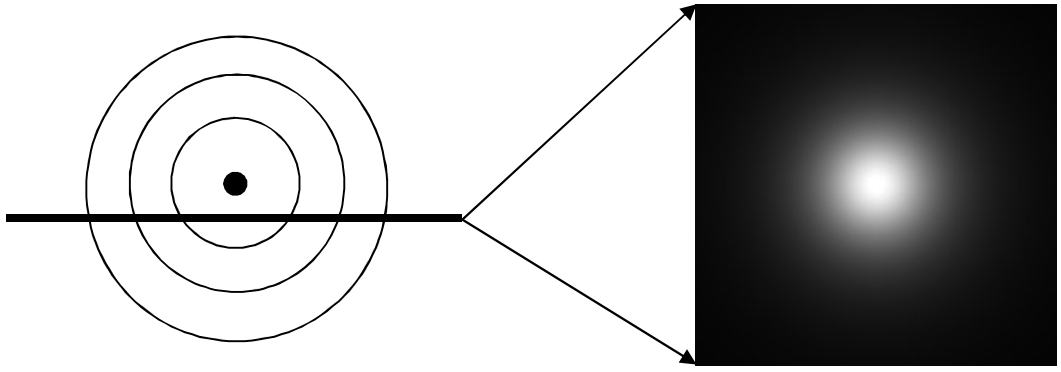
$$L = k_d L_l (\vec{n} \cdot \vec{l}) S(\lambda)$$

Lumière directionnelle

- Pour une lumière directionnelle (i.e. à l'infini), l'intensité de la lumière qui atteint une surface n'est pas dépendante de la distance à la lumière.
- Tous les rayons issus de la lumière sont considérés parallèles entre eux
- Deux surfaces identiques de même orientation mais à des distances différentes auront le même shading dans ce cas

Lumière ponctuelle

- L'intensité varie comme $1/r^2$ où *r* est la distance d'un point à la lumière
- L'intensité émise par un point de lumière est égale sur chaque *dA* à une distance fixe du point (surface d'une sphère de $4\pi r^2$)
- Un élément sur cette surface recevra $dA/(4\pi r^2)$ de l'intensité du point de lumière



Le modèle de réflexion (ou d'illumination) ambiant-diffus

- Pour un point de lumière, le modèle de réflexion devient:

$$L_p = S(\lambda)\left(L_a k_a + \frac{L_l k_d \max(\vec{n} \cdot \vec{l}, 0)}{r^2}\right)$$

  - Attention: les modèles de réflexion dépendent des distances et des orientations des surfaces. Ils doivent donc être évalués en espace monde, et non pas en espace projectif...

Couleur - RGB

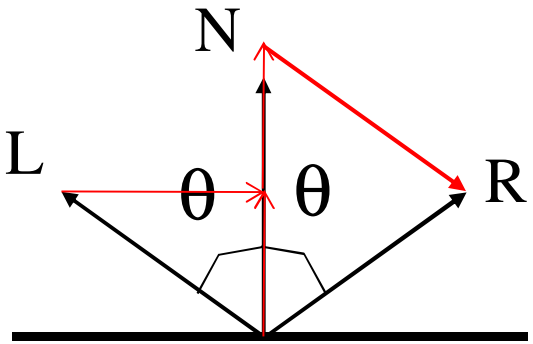
- Il faut pouvoir transférer les intensités en couleurs
- Les formules précédentes se généralisent si on suppose que l'on peut traiter les longueurs d'onde indépendamment
- Chaque coefficient détermine la contribution (intensité) à chaque longueur d'onde
- Le modèle ambient-diffus pour une lumière de couleur (L_R, L_G, L_B) sur une surface de couleur (S_R, S_G, S_B):

$$L_{p\lambda} = L_{a\lambda} k_{a\lambda} S_\lambda + \frac{L_{l\lambda} k_{d\lambda} S_\lambda \max(\vec{n} \cdot \vec{l}, 0)}{r^2} \quad \lambda \in \{R, G, B\}$$

La réflexion spéculaire (Miroir)

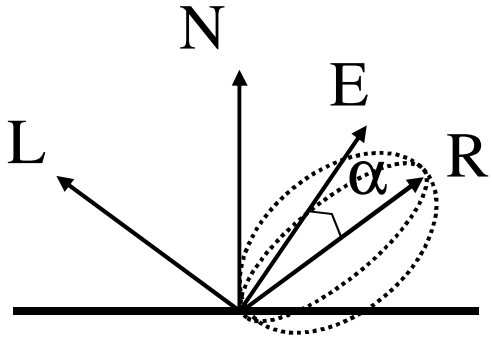
- La réflexion spéculaire apparaît sur des surfaces brillantes sous la fome d'une région de haute intensité (pour chaque lumière) que l'on appelle le *highlight*
- Ce *highlight* se déplace alors qu'on change le point de vue (soleil sur voiture, plancher ciré)
- Pour une surface parfaitement lisse (miroir), la direction de réflexion spéculaire est unique et correspond à

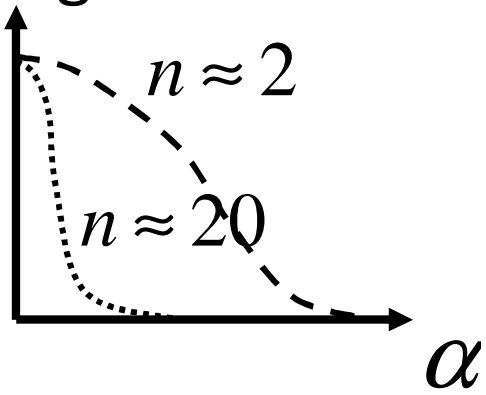
$$\vec{R} = 2(\vec{N} \cdot \vec{L})\vec{N} - \vec{L}$$



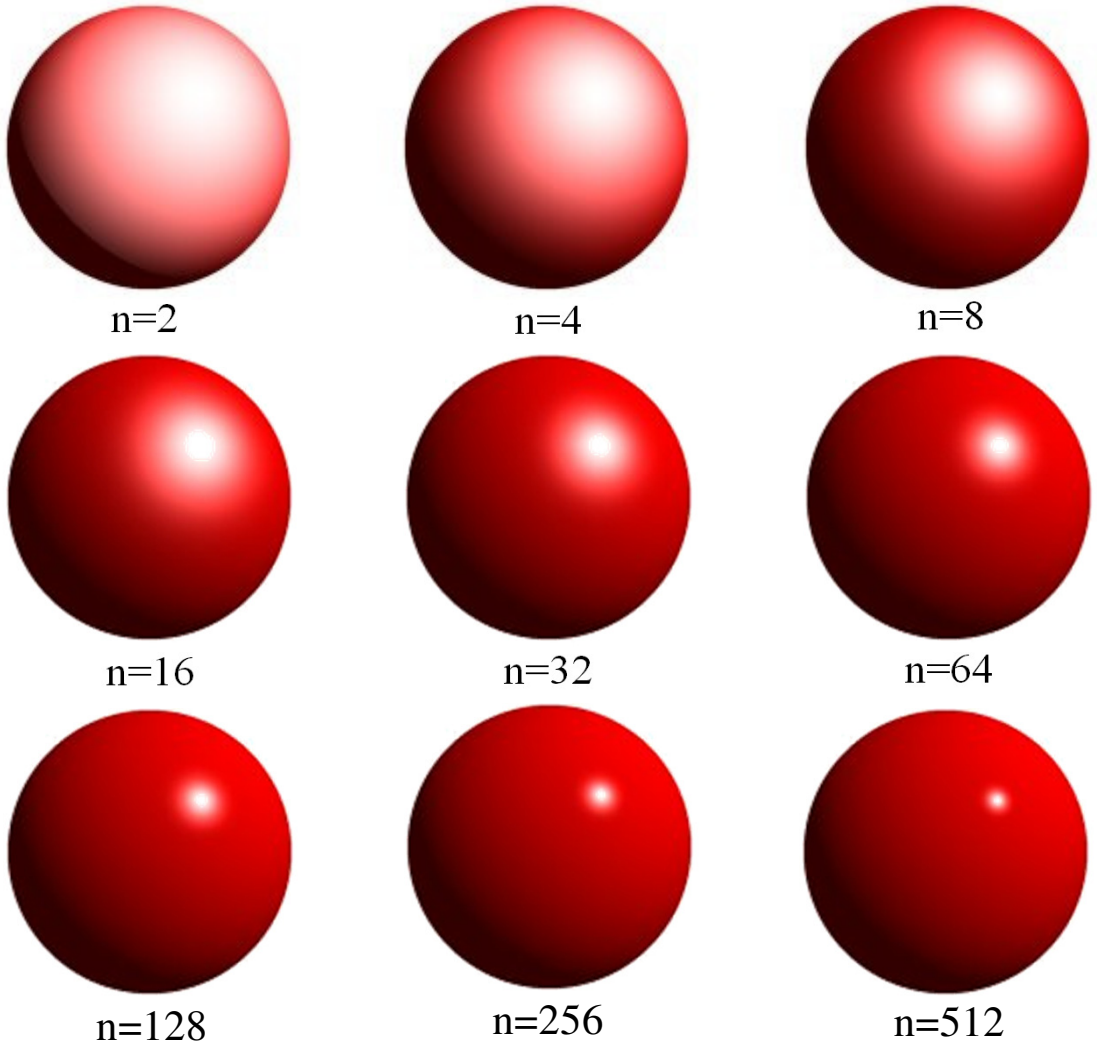
La réflexion spéculaire (Phong)

- Si une surface est un peu rugueuse, un peu de lumière sera réfléchie spéculairement autour de $\vec{R}$
- Le modèle spéculaire de *Phong* fait décroître l'intensité de cette réflexion selon $\cos^n \alpha$ entre les directions $\vec{R}$ et $\vec{E}$ $(= \overrightarrow{\omega_o})$
- *n* contrôle la rugosité de la surface telle que pour un miroir $n \to \infty$ et une surface très rugueuse $n \to 1$



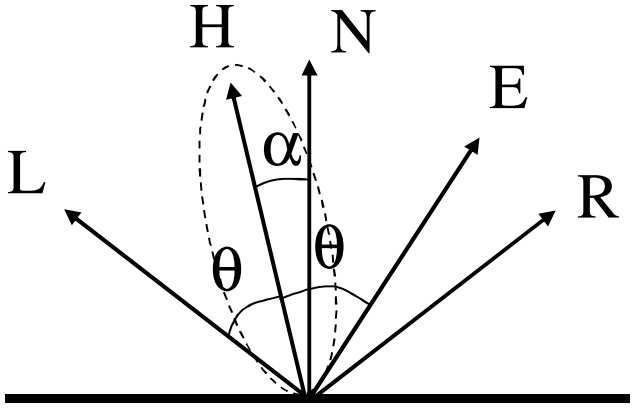


La réflexion spéculaire (Phong): la rugosité *n*



La réflexion spéculaire (Blinn)

- Le modèle spéculaire de *Blinn* fait lui aussi décroître l'intensité de cette réflexion selon $\cos^n \alpha$ mais cette fois entre les directions $\vec{N}$ et $\vec{H}$
- $\vec{H}$ est le vecteur bisecteur entre $\vec{L}$ et $\vec{E}$



$$\vec{H} = \frac{\vec{L} + \vec{E}}{\left\| \vec{L} + \vec{E} \right\|}$$

La réflexion spéculaire (Blinn)

- Ce modèle est basé sur une surface constituée d'une distribution de petits miroirs disposés en V
- Le modèle tient compte de la distribution, du masquage entre facettes et du coefficient de Fresnel

+ Physiquement plus plausible
+ $\left(\vec{N} \cdot \vec{H}\right)^n$ ne peut pas être négatif si $\left(\vec{N} \cdot \vec{L} > 0\right) \text{et } \left(\vec{N} \cdot \vec{E} > 0\right)$
+ Pour une lumière directionnelle et une projection orthographique, $\vec{H}$ est constant sur toute la scène

Réflexion plastique vs. métallique

- Un métal a tendance à réfléchir spéculairement selon la couleur de sa surface
- Un plastique (diélectrique) a tendance à réfléchir spéculairement selon la couleur de la lumière
- Soit *m* la proportion de réflexion spéculaire de la surface, le modèle peut s'exprimer comme

$$L_{p\lambda} = [mS_{\lambda} + (1-m)]\frac{k_{s\lambda}L_{l\lambda}\max\left(\vec{N}\cdot\vec{H},0\right)^{n}}{r^{2}}$$

Sources de lumière multiples: la linearité d'illumination

- Chaque lumière contribue à l'intensité finale
- Si on utilise plusieurs sources de lumière, on n'a qu'à sommer la contribution de chacune
- La contribution ambiante n'est considérée qu'une seule fois
- Le modèle final correspond donc à

$$L_{p\lambda} = L_{a\lambda} k_{a\lambda} S_\lambda + \sum_i \frac{L_{l\lambda i}}{r_i^2} \left[k_{d\lambda} S_\lambda \left\lfloor \vec{N} \cdot \vec{L}_i \right\rfloor + k_{s\lambda} \left[m S_\lambda + (1-m) \right] \left\lfloor (\vec{R}_i \cdot \vec{E})^n \right\rfloor \right]$$

Sources de lumière multiples

+ simple et intuitif
- aucune conservation de l'énergie
- Contraintes parfois appliquées à ce modèle

$$k_{d\lambda} + k_{s\lambda} = 1$$
$$k_{*\lambda} \in [0,1]$$
$$L_{*\lambda} \in [0,1]$$
$$S_{\lambda} \in [0,1]$$

  - mais puisque le modèle n'est pas entièrement basé sur une théorie précise, ces contraintes sont souvent inutiles, pour ne pas dire nuisibles
- Attention au *clamping* des valeurs supérieures à 1.0 (ou 255)

IFT3355: Infographie
Sujet 5: `shading` 2
(illumination *locale* 2)

Derek Nowrouzezahrai
Département d'informatique et de recherche opérationelle
Université de Montréal

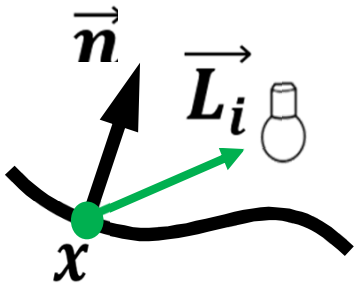
Survol: lancer de rayons secondaires

- Le premier rayon identifie la surface visible de l'œil où on peut effectuer notre calcul de shading
  - Pour les objets **opaques** et **non réfléchissant** ce valeur peut être obtenu avec ex: le modèle ambiant-diffus présenté plus tôt
- Dans ces cas, d'autres rayons peuvent être tracés pour déterminer, par exemple, la visibilité des lumières et calculer leurs contributions
- Des rayons secondaires peuvent aussi être utiliser pour déterminer le shading des objets (semi-)**transparentes** et/ou **réfléchissants**

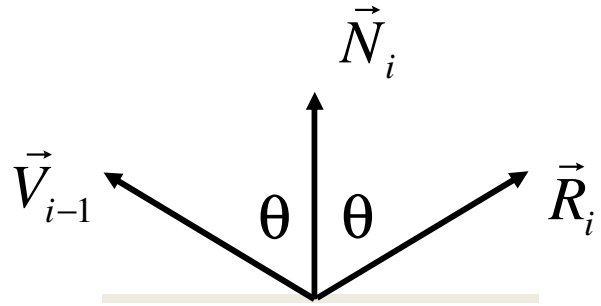
- On doit s'assurer de ne pas ré-intersecter l'objet au nouveau point de départ (*surface acné*)
  - ajouter une distance epsilon au rayon
  - ne pas intersecter la surface elle-même
    (plus limité: surface convexe et test par rapport à la normale)

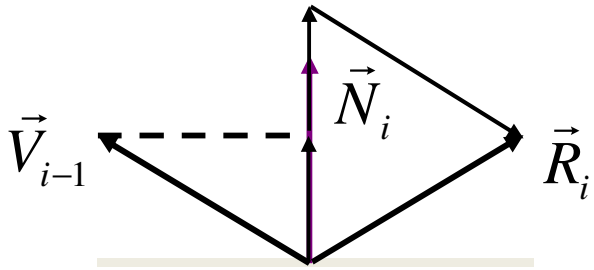
Lancer de rayons: types de rayons

- Rayons d'ombre: $\vec{L}_i = \dfrac{(L - P_i)}{\|(L - P_i)\|}$



- Rayons réfléchis: $\vec{R}_i = 2(\vec{V}_{i-1} \cdot \vec{N}_i)\vec{N}_i - \vec{V}_{i-1}$

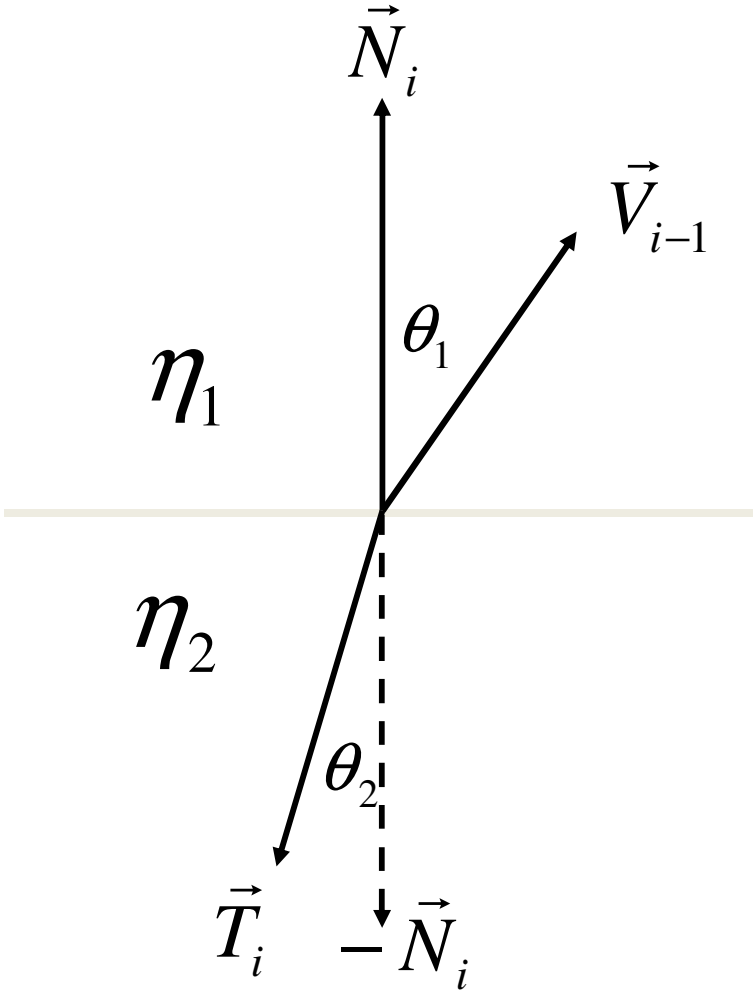




Lancer de rayons: types de rayons

- Rayons transmis / réfractés:

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{\eta_2}{\eta_1}$$

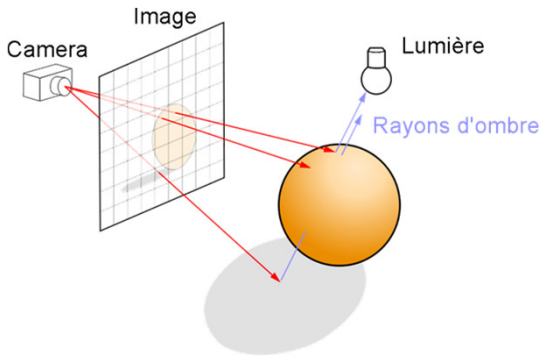


Ombres

- Déterminer si un objet est dans l'ombre pour une lumière consiste à tester la visibilité de la lumière à partir de l'objet
- Cette visibilité est simplefiée: il faut seulement identifier **si** un objet bloque la lumière, et non pas l'objet le plus proche

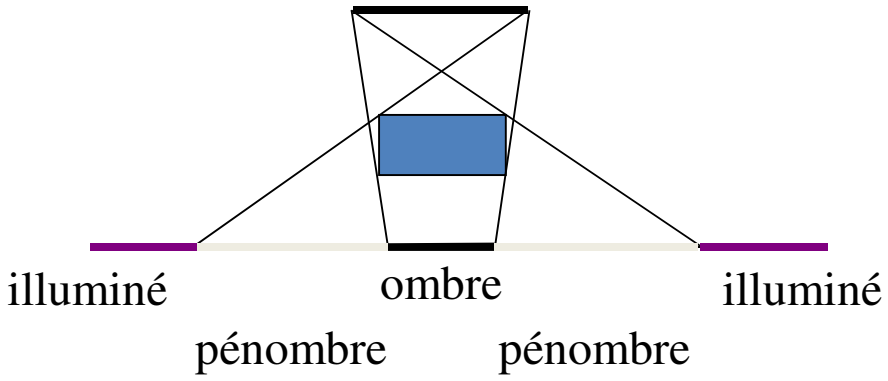
```
for( chaque pixel p )
  Générer rayon r de l'oeil à p
  float dist_min = infinité
  intersection h = null
  for( chaque objet o )
    if( trace( r, o ) et
        hit_dist <= dist_min )
      dist_min = hit_dist
      h.objet = o
      h.dist = hit_dist
p.couleur = shade(h)
```

```
bool dans_ombre = false;
for( chaque objet o )
  if( trace( r, o ) )
    dans_ombre = true;
    break;
```



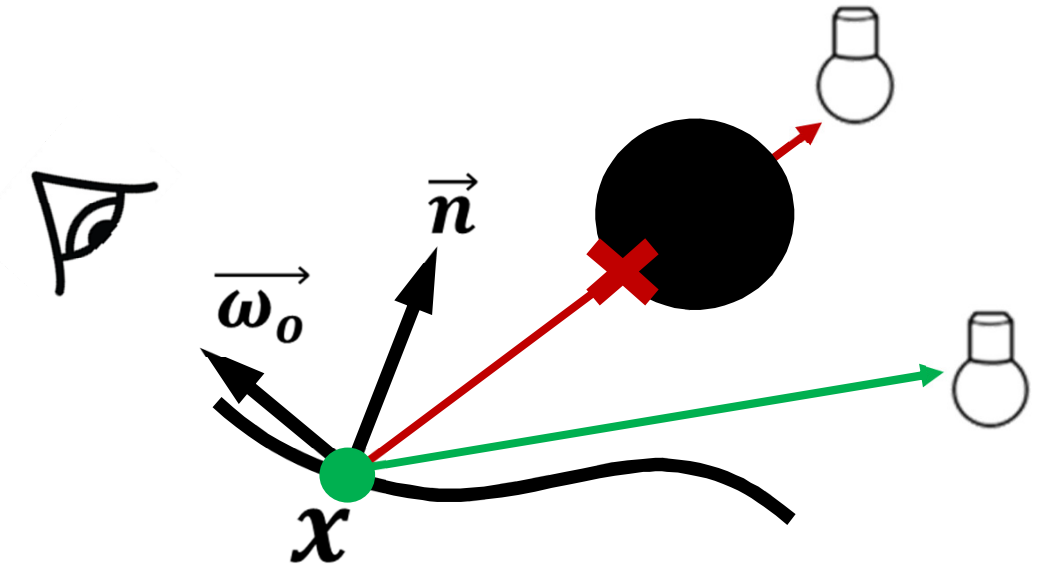
Ombres

- Pour une lumière directionnelle et ponctuelle, la décision est binaire. Un point est illuminé ou est dans l'ombre.
- Pour une lumière étendue (*e.g.* linéaire ou polygonale), on doit calculer la proportion de la lumière visible du point. Un point peut alors être dans l'ombre, complètement illuminé, ou dans la pénombre.

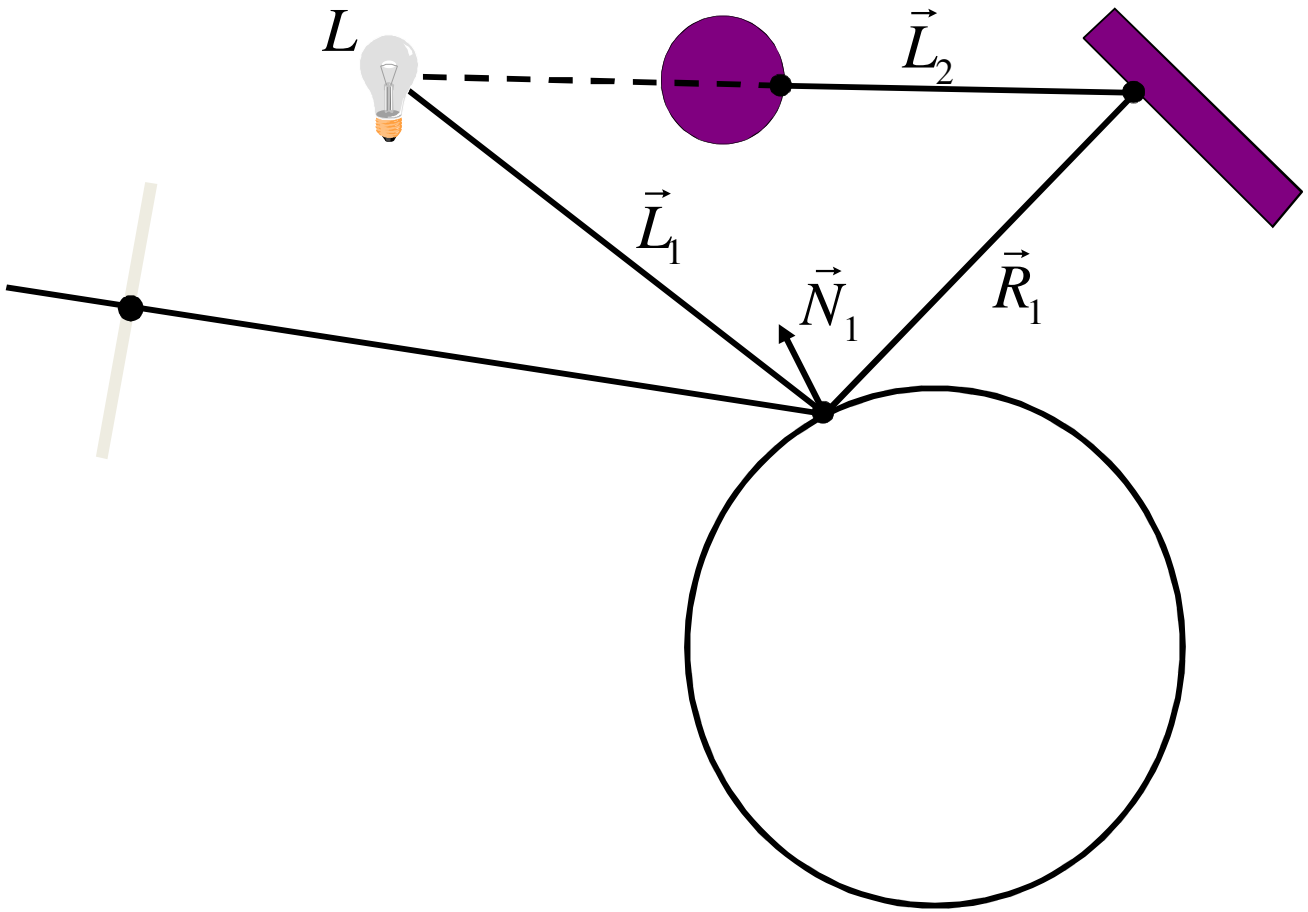


Calcul des ombres avec les rayons (pour lumière directionnelle et ponctuelle)

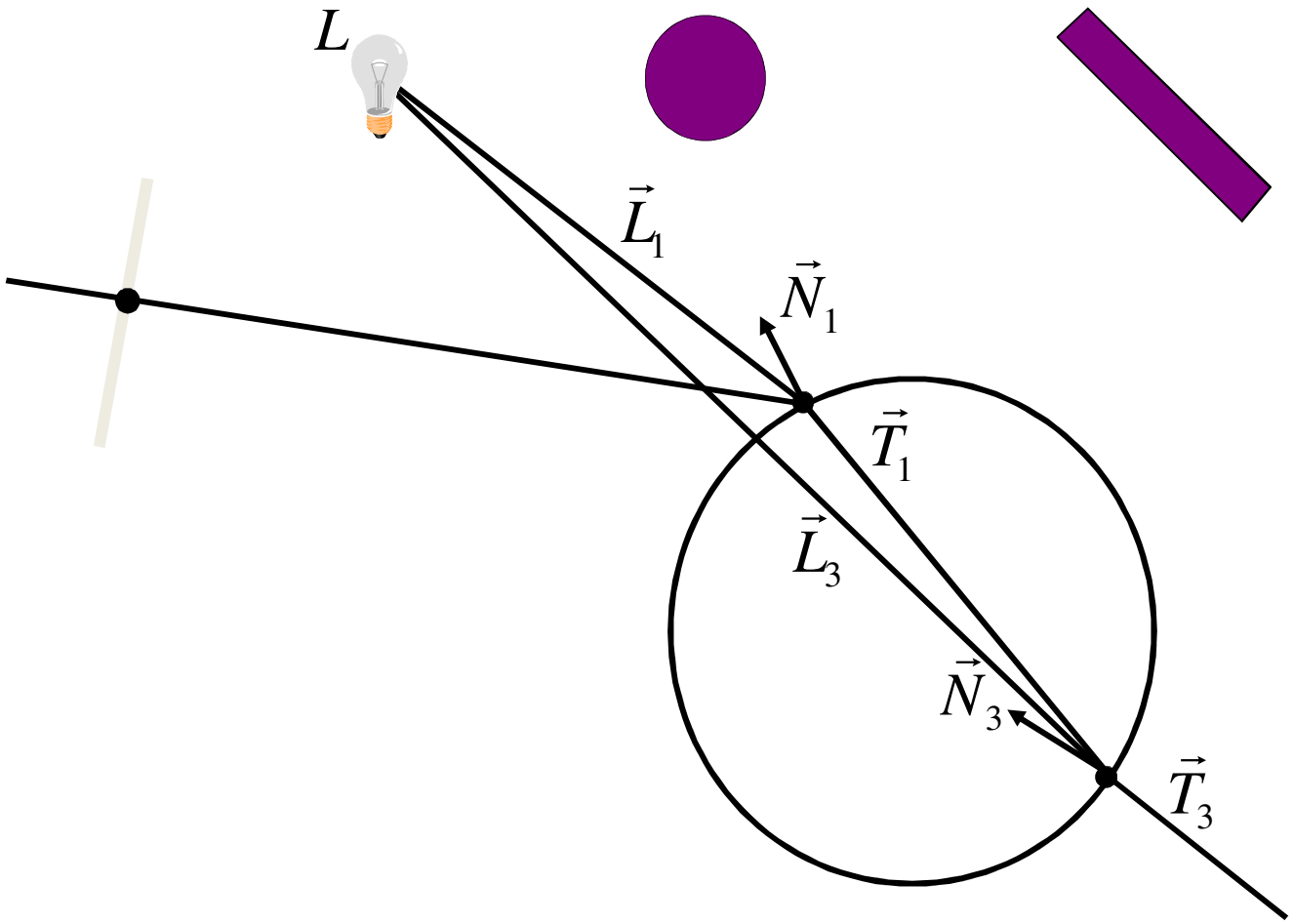
- Trace un rayon d'ombre du point jusqu'à la lumière
- S'il n'y a aucune intersection sur ce segment, le point est illuminé



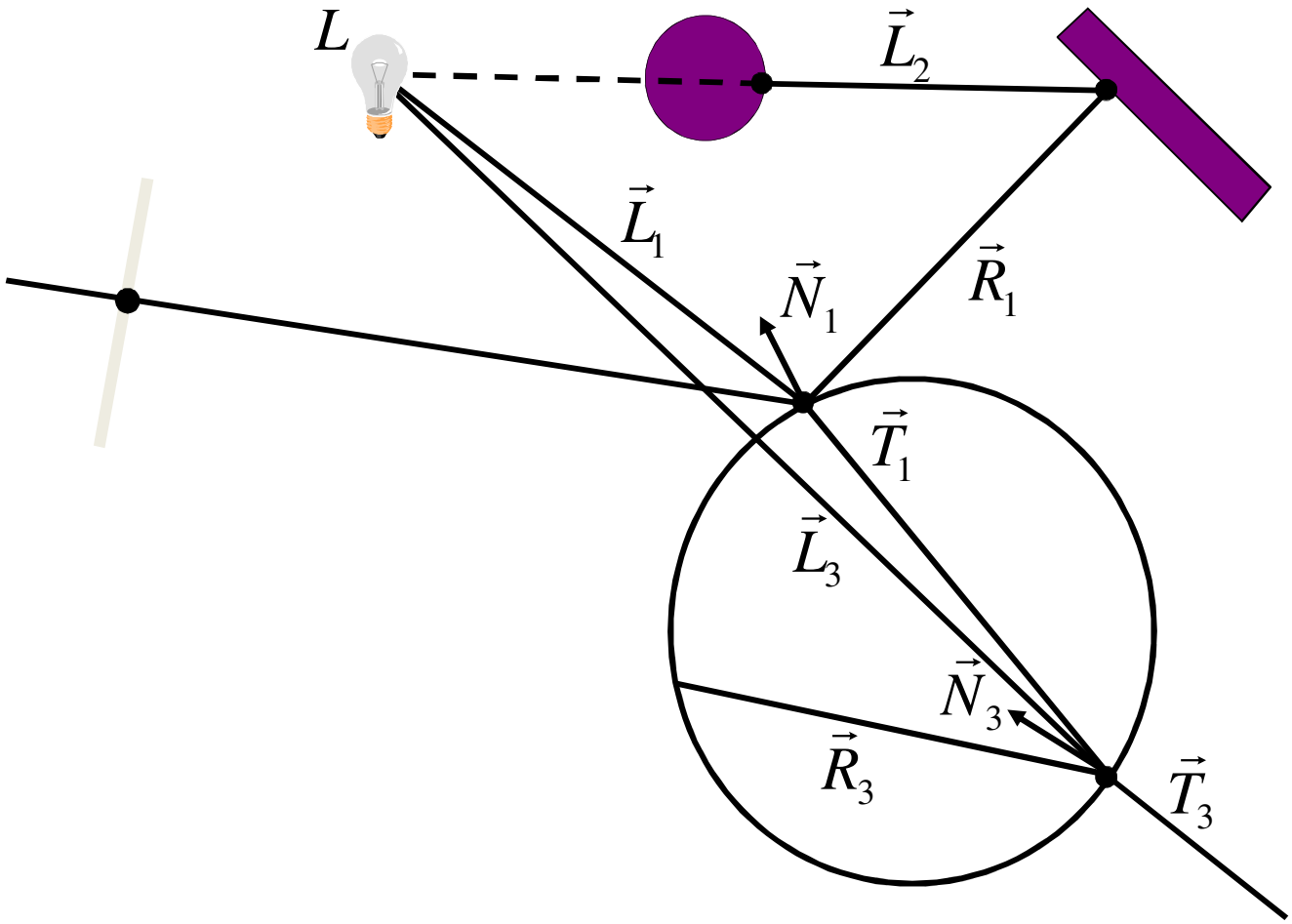
Lancer de rayons récursif: réflexion



Lancer de rayons récursif: transmis



Lancer de rayons récursif: réflexion + transmission



Lancer de rayons: réfraction

$$\vec{T}_i = \vec{M}\sin\theta_2 - \vec{N}_i\cos\theta_2$$

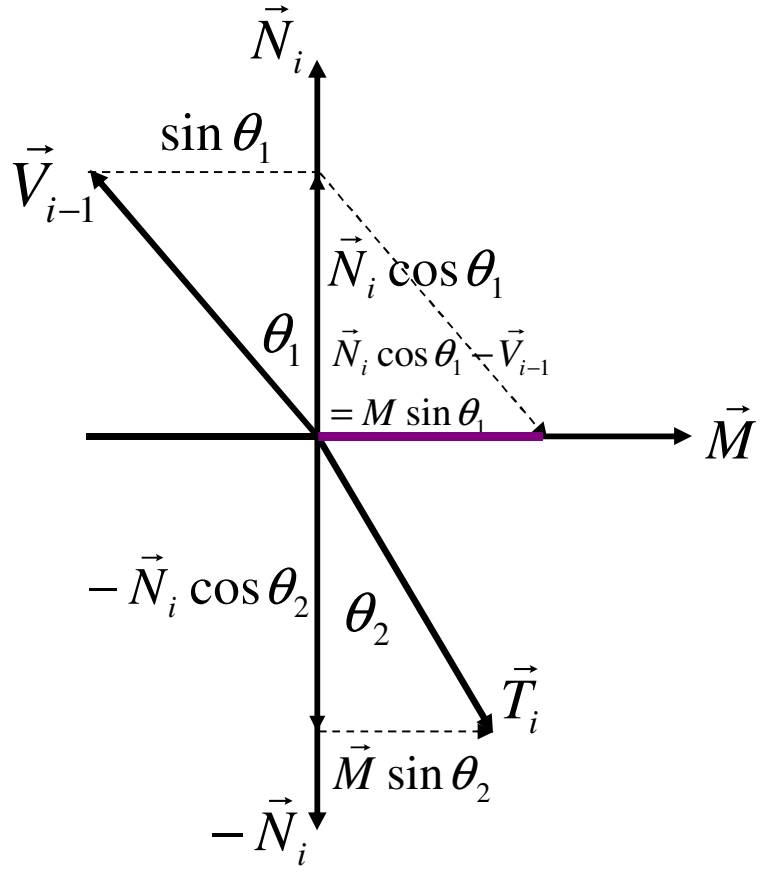
$$\vec{M} = \frac{(\vec{N}_i\cos\theta_1 - \vec{V}_{i-1})}{\sin\theta_1}$$

$$\vec{T}_i = \frac{(\vec{N}_i\cos\theta_1 - \vec{V}_{i-1})}{\sin\theta_1}\sin\theta_2 - \vec{N}_i\cos\theta_2$$

$$\vec{T}_i = \frac{\eta_1}{\eta_2}(\vec{N}_i\cos\theta_1 - \vec{V}_{i-1}) - \vec{N}_i\cos\theta_2$$

$$\eta = \eta_1/\eta_2$$

$$\vec{T}_i = \vec{N}_i(\eta\cos\theta_1 - \cos\theta_2) - \eta\vec{V}_{i-1}$$



$$\sqrt{1-\sin^2\theta_2} = \sqrt{1-\eta^2\sin^2\theta_1} = \sqrt{1-\eta^2(1-\cos^2\theta_1)}$$

$$\vec{T}_i = \vec{N}_i\left(\eta(\vec{N}_i\cdot\vec{V}_{i-1}) - \sqrt{1-\eta^2(1-(\vec{N}_i\cdot\vec{V}_{i-1})^2)}\right) - \eta\vec{V}_{i-1}$$

Lancer de rayons distribués: autres effets

- Anti-aliassage
  - échantillonnage régulier
  - échantillonnage stratifié (*jitter*)
- Pénombres
  - échantillonner la lumière
  - échantillonner l'angle solide formé par la lumière
- Profondeur de champ
  - échantillonner les lentilles (simulées) d'une caméra
- Réflexion *glossy*
  - échantillonner le cone de réflexion
- Flou de mouvement
  - échantillonner (le mouvement dans) le temps

IFT3355: Infographie
Sujet 5: `shading` 3
(illumination *locale* 3)

Derek Nowrouzezahrai
Département d'informatique et de recherche opérationelle
Université de Montréal

Shadings

(à ne pas confondre avec *Shading*)

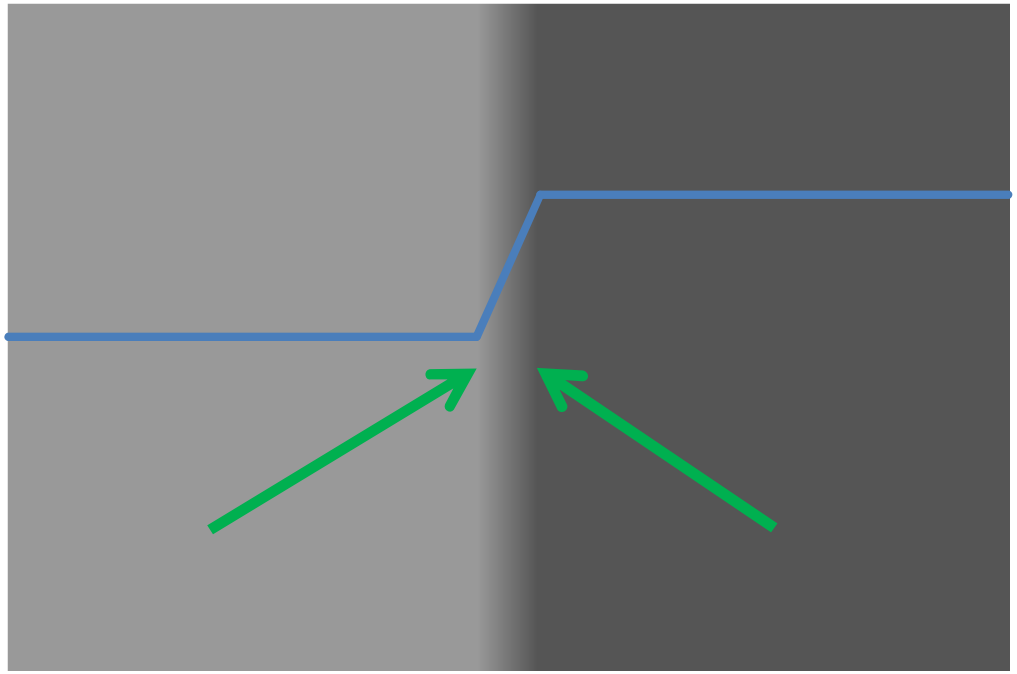
- On peut calculer l'illumination à chaque point d'une scène, mais il est possible d'en réduire le coût en utilisant des approximations pour l'illumination sur un polygone
- Plusieurs techniques:
  - *Flat shading*
  - *Gouraud shading*
  - *Phong shading*
    - (à ne pas confondre avec le modèle d'illumination *Phong*)

Flat Shading

- calcule l'illumination pour un point du polygone (centre ou un sommet)
- suppose que cette illumination est la même sur tout le polygone
- couleur uniforme et *Mach bands*
- valide si
  - $\left(\vec{N} \cdot \vec{L}\right)$ est constant : lumière directionnelle
  - $\left(\vec{N} \cdot \vec{E}\right)$ est constant : projection parallèle

Mach bands

- Une discontinuité de valeur ou de pente d'intensité est exagérée perceptuellement par l'oeil dû à un facteur d'inhibition de récepteurs adjacents dans l'oeil



Calcul de normales des polygones

- Calcul analytique si on connaît la surface que le polygone approxime (ex: sphère)
- Moyenne des normales des polygones dont le sommet fait partie (ou moyenne pondérée par l'angle formé par ce polygone au sommet)

$$\vec{N} = \frac{\sum \vec{N}_i}{\left\| \sum \vec{N}_i \right\|}$$

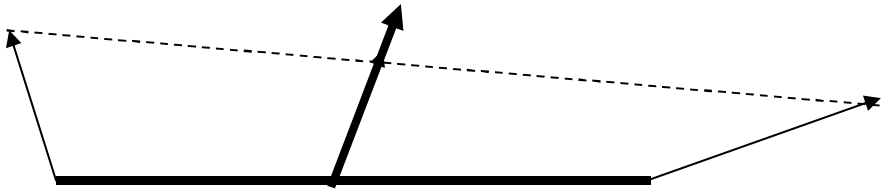
- Si une arête existe vraiment, un sommet a *i* normales dont une seule est choisie dépendant du polygone à rendre

Shading de Gouraud

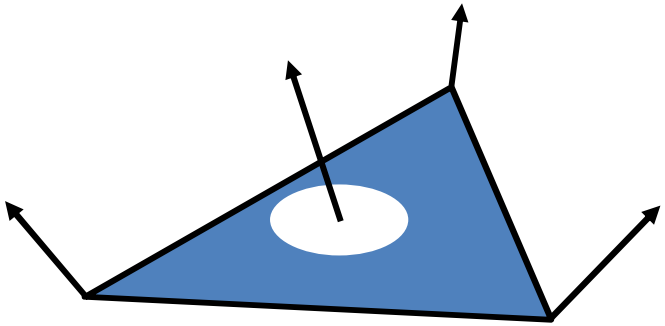
- calcule l'illumination à chaque sommet du polygone (normale + illumination → couleur)
- interpolation bilinéaire des couleurs tout comme on le faisait pour la profondeur
- rapide mais peut rater des *highlights,* il faut alors un maillage plus fin des polygones
- élimine les discontinuités d'intensité mais pas celles de pente d'intensité, donc les *Mach bands* sont réduites mais elles sont toujours présentes

Shading de Phong

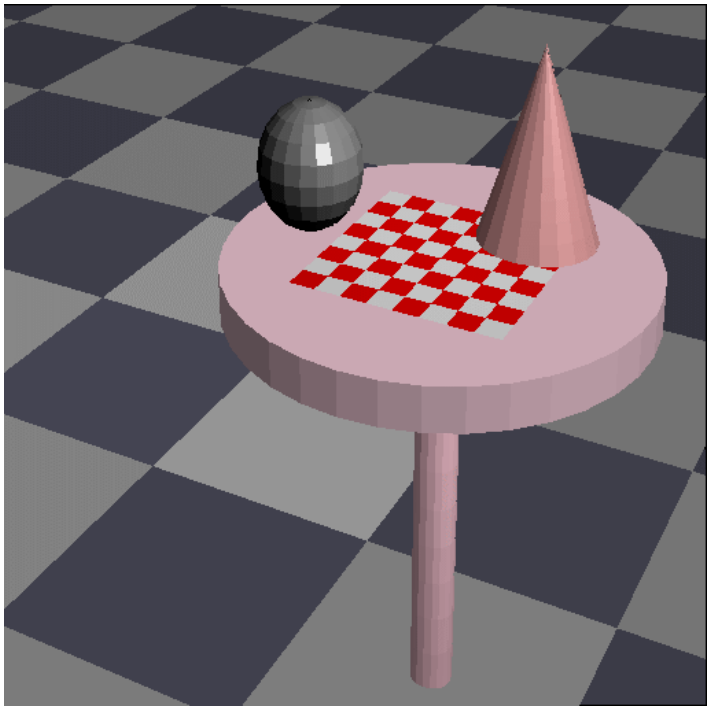
- interpolation bilinéaire des normales pour tout point dans le polygone
- calcule l'illumination pour chaque normale interpolée



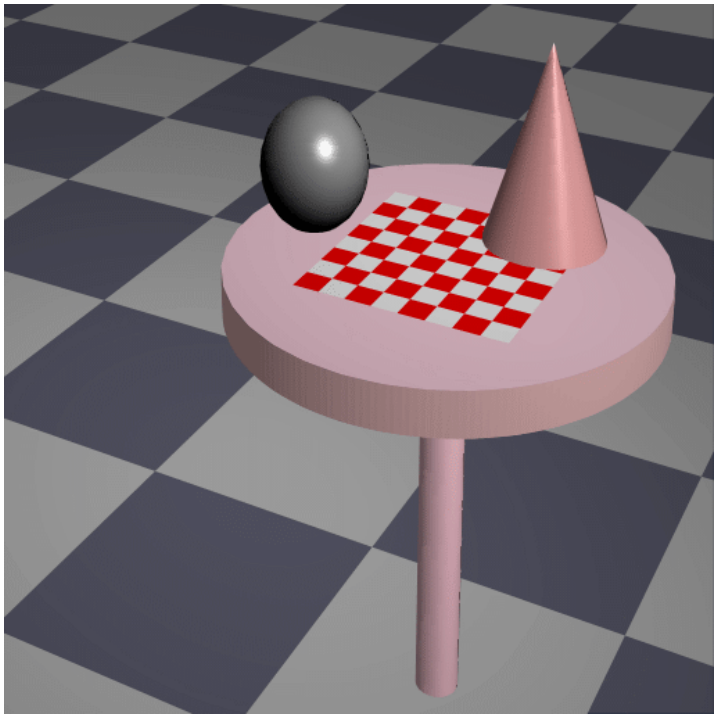
- Plus coûteux à calculer mais approxime beaucoup mieux les *highlights*



Flat shading *vs.* Gouraud shading

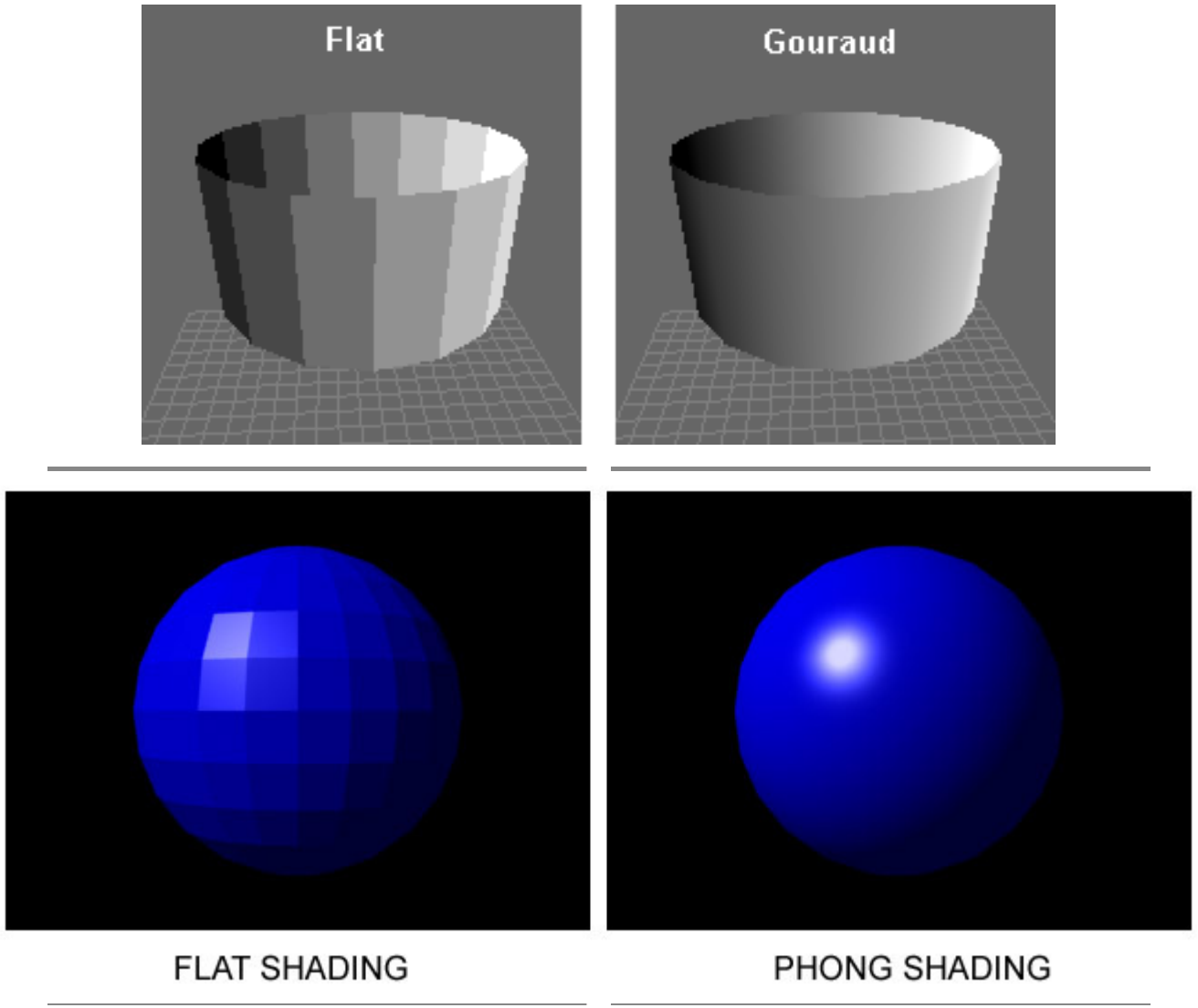


flat shading



Phong shading

Flat vs. Gouraud vs. Phong



Survol: les ombres (ponctuelles)… encore

- Rappelons qu'on a déjà étudié comment effectuer le calcul des ombres causés par des lumières ponctuelle/directionnelles dans un système de **lancer de rayons**
- Avant étudier des effets plus compliqués d'illumination globale (plus ou moins dans un contexte de lancer de rayons), nous revisiterons le calcul des ombres ponctuelle (ombre dur) dans un système de **rastérisation**

- Nous étudierons deux techniques:
  - Un qui utilise un tampon de profondeur appelé un *shadow map*
  - Et un qui génère de la géométrie supplémentaires qui représentes des volume d'ombre – des *shadow volumes*

Tampon de profondeur d'ombre - *Shadow map*

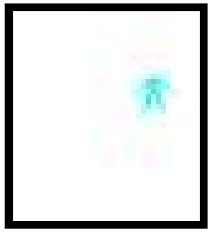
- Construit un tampon de profondeur à partir de la lumière *L* (aucun tampon image nécessaire)
- Pour déterminer si un point *P* est dans l'ombre

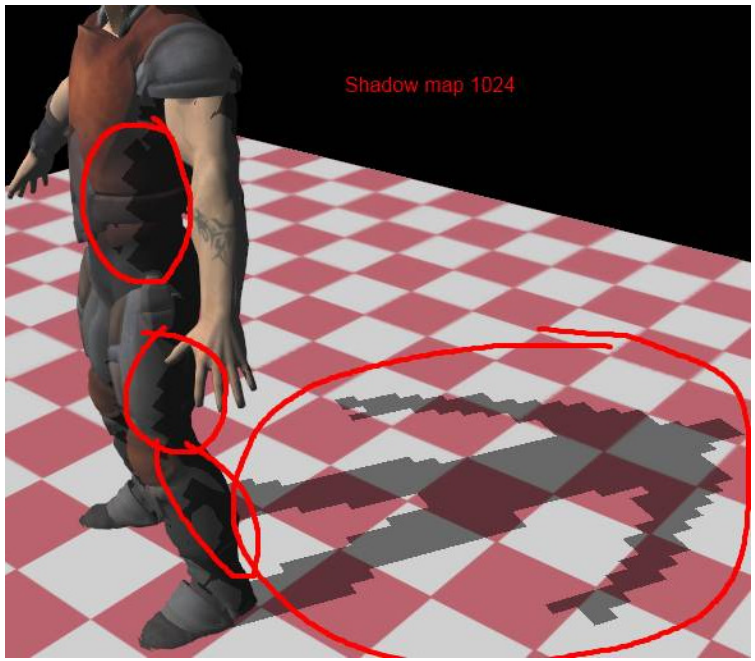
```
bool dans_ombre = true;
for( chaque pixel P sur l'écran )
  point P_l = mul(L_projection, P.position)
    if( distance( L – P ) < P_l.z )
      dans_ombre = false;
```

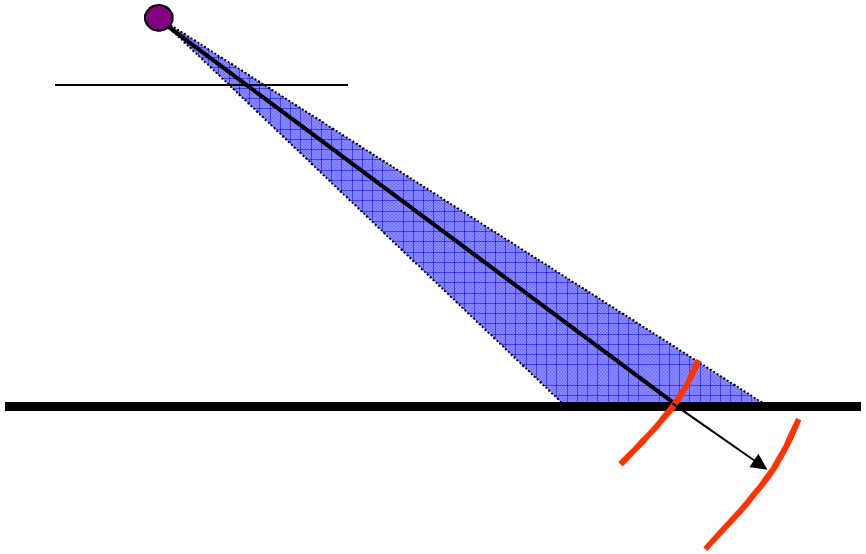
+ facile à implanter et disponible en hardware
- plusieurs *shadow maps* pour omni-directionnel
- résolution vs. mémoire
- aliassage

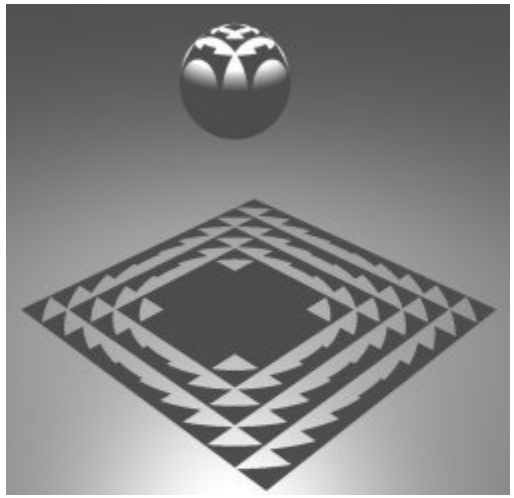
Shadow map - artéfacts

- effet d'escaliers aux silhouettes des ombres
- *acné de surface* (ajoute epsilon à la distance du point)



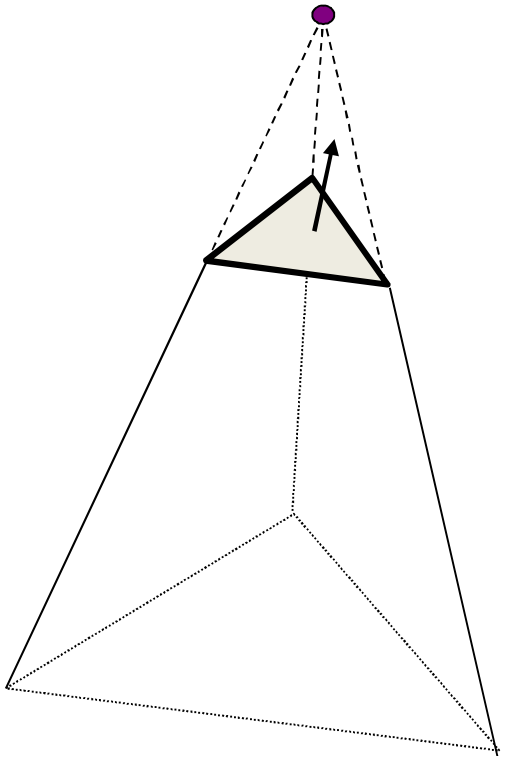


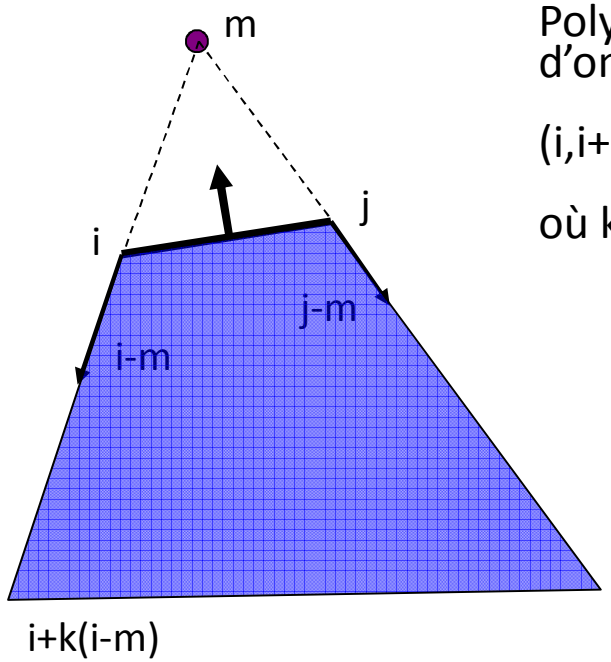




Volumes d'ombre (de Crow)

- Pour chaque arête d'un polygone faisant face à la lumière, on construit un quadrilatère reposant sur le plan arête-lumière.





Polygone (quadrilatère) d'ombre:

(i,i+k(i-m),j+k(j-m),j)

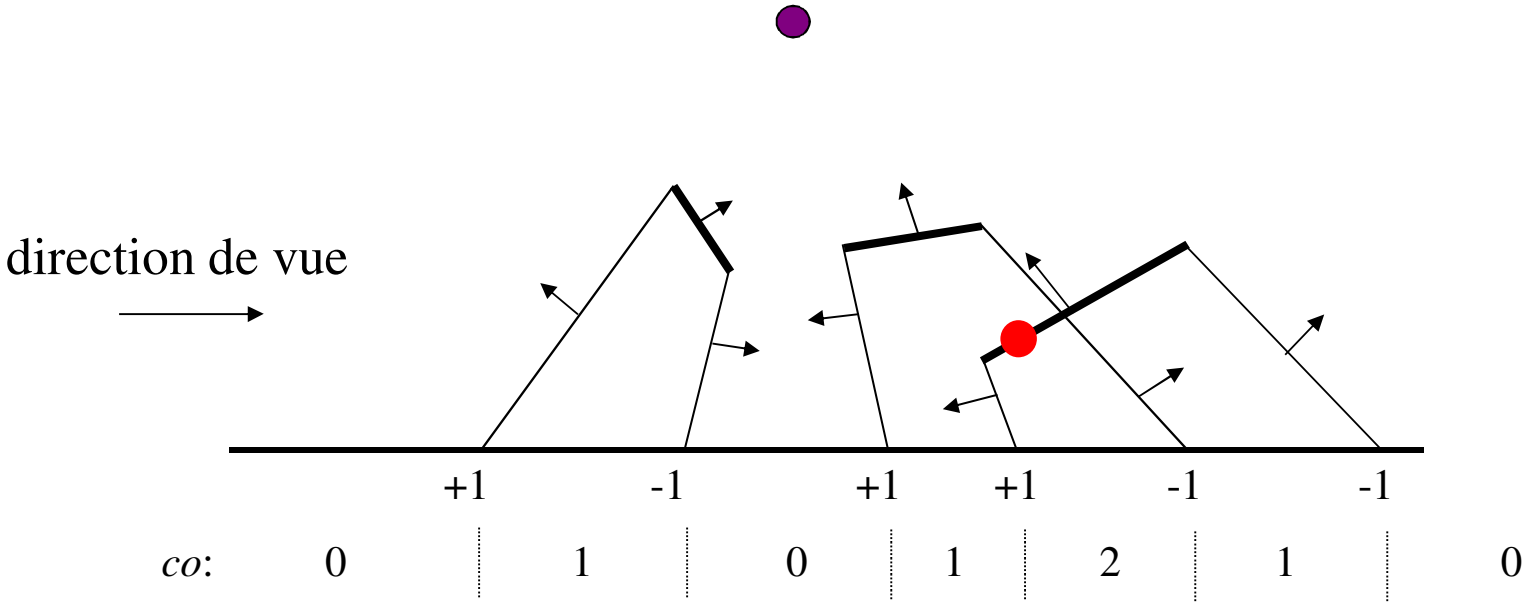
où k est très grand

Volumes d'ombre

- Un point sera dans l'ombre s'il est à l'intérieur d'un polyèdre d'ombre défini par chaque polygone, sa projection de la lumière (*cap*) et ses polygones d'ombre
- On fait d'abord un rendu pour remplir le tampon de profondeur (*depth buffer*), puis un rendu des polygones d'ombre
- Lors de ce rendu, on conserve un compteur d'ombre *co*

```
if (N · E < 0)     // polygone d'ombre fait face à l'oeil
    co = co + 1    // front - facing
else
    co = co − 1    // back - facing
```

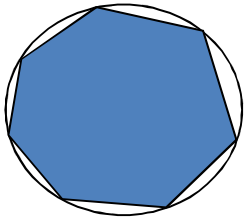
Volumes d'ombre



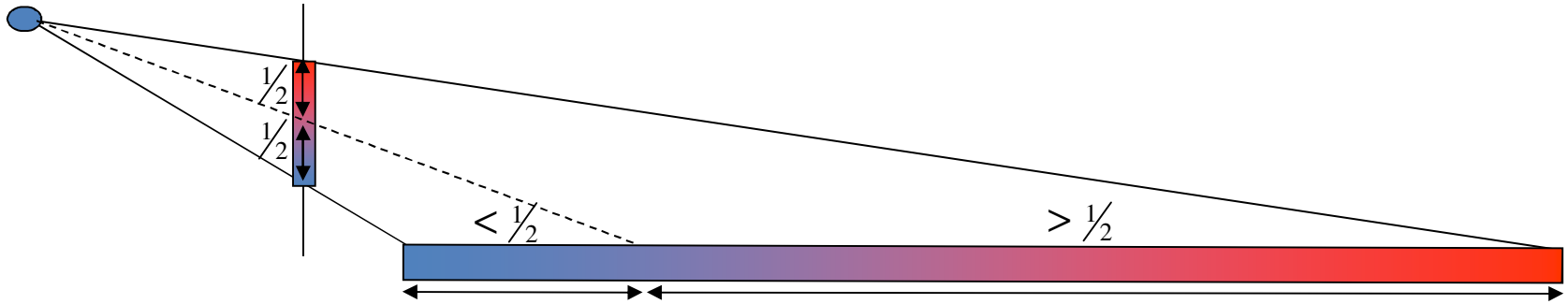
+ espace objet et omni-directionel
- seulement pour des polygones
- limites (min-max) sur les valeurs du compteur (*stencil planes*)
- initialisation de *co* pour le point de vue avec problèmes si les *front ou back clipping* intersectent un polygone d'ombre

Problèmes reliés à l'interpolation du *shading*

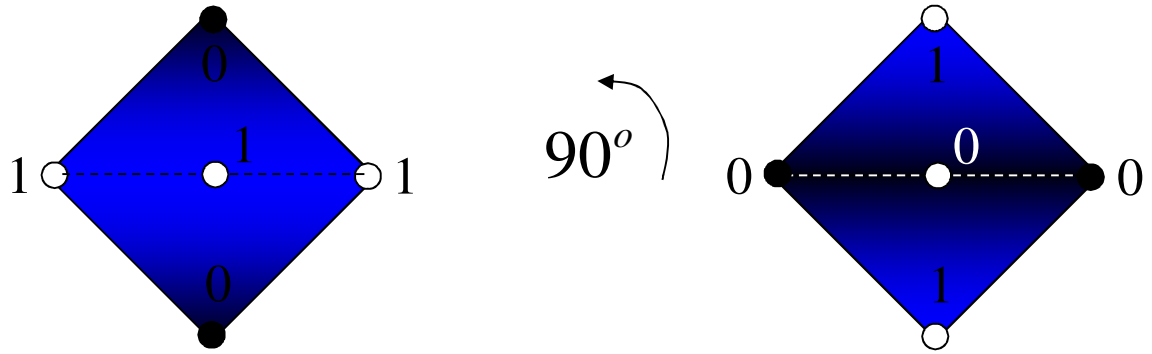
- Silhouette polygonale



- Distorsion due à la projection en perspective



- Dépendance d'orientation



Problèmes reliés à l'interpolation du *shading*

- Normales comme une mauvaise représentation