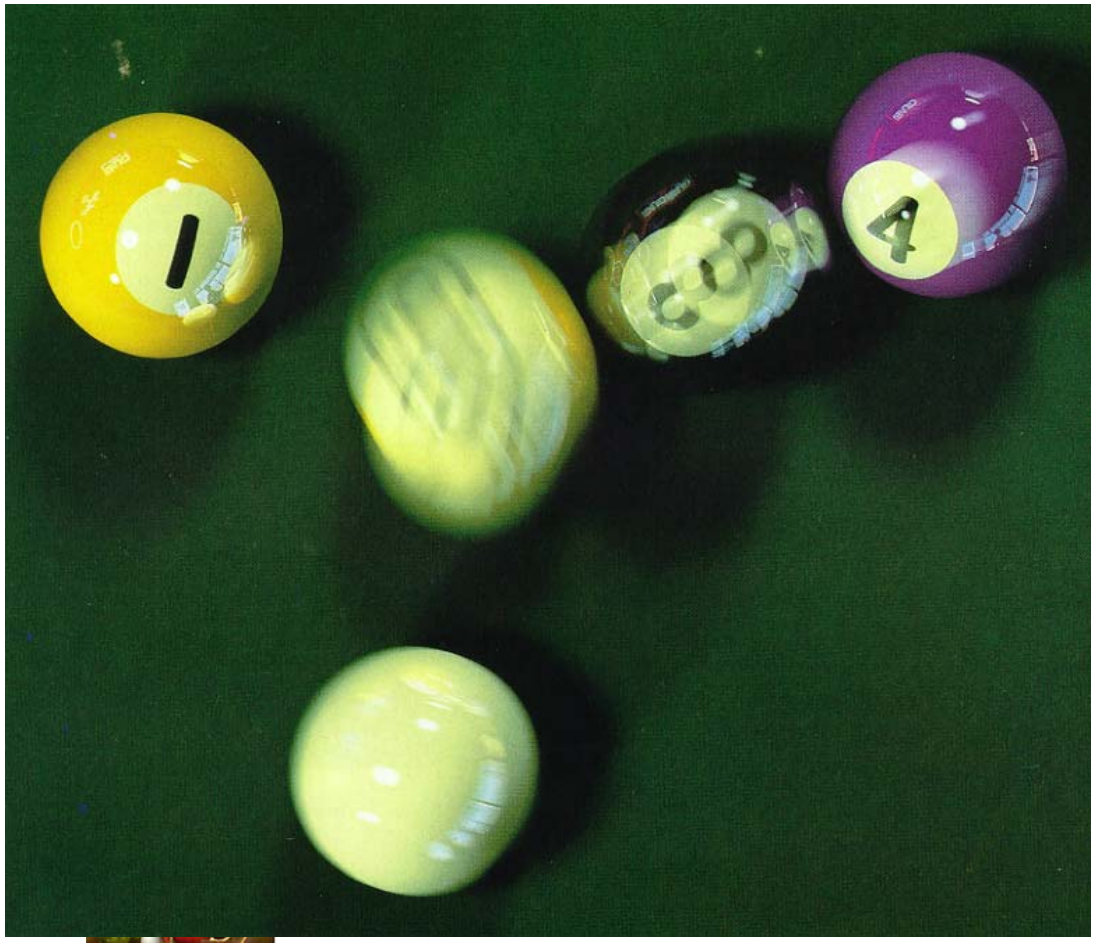
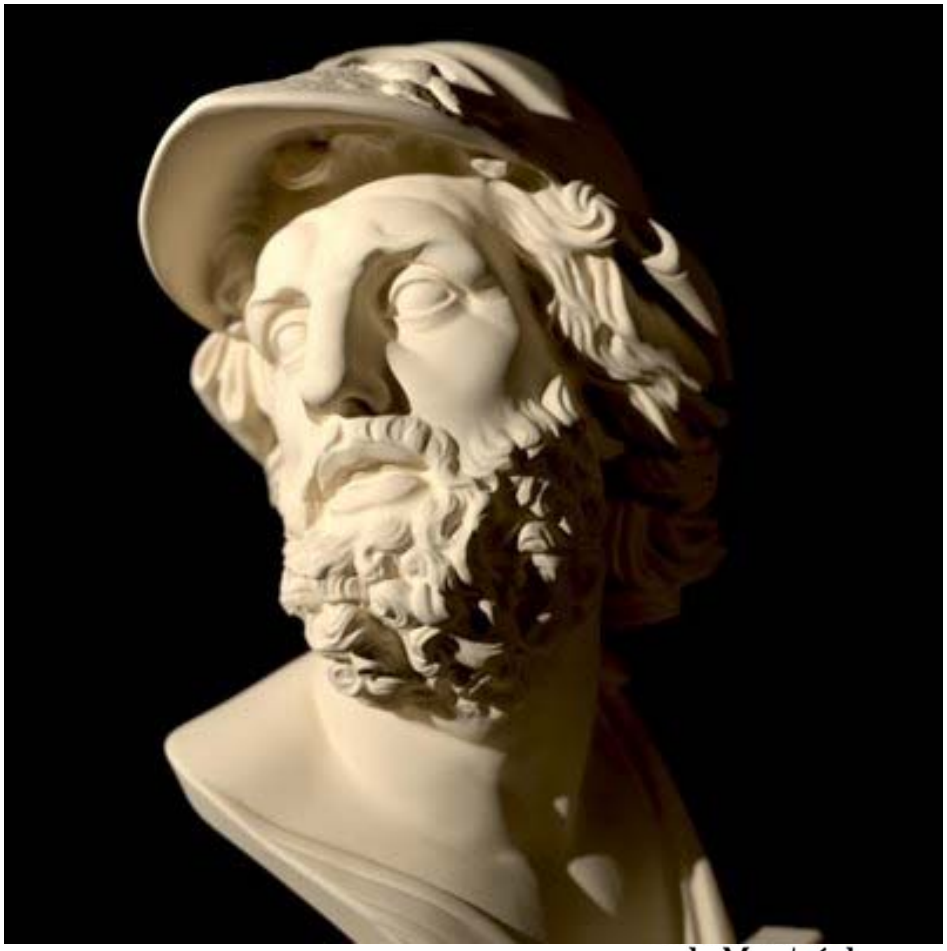
IFT3355: Infographie
Sujet 4: Visibilité 3
(lancer de rayons 101)

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

Université de Montréal

Motivation





LIGUM

de Montréal

Lancer de rayons
(*ray tracing/ray casting*)

- Les techniques précédentes en espace image utilise un combinaison de *projection* et *scan-conversion*
  - Cette famille de méthode de détermination de visibilité est appelée **rastérisation**
- D'un autre côté, les lanceurs de rayons (ou *ray casting* pour ne traiter que la visibilité) consiste à lancer
  - des rayons à travers de la fenêtre pour identifier la première surface visible de la scène 3D (i.e. pour traiter la visibilité)
  - *O(n p)* pour *p* pixels et *n* objets (peut le réduire à *O(log(n) p)*)
- Traitement pixel par pixel plutôt que objet par objet
- Utilisé aussi pour le *picking*

Université de Montréal

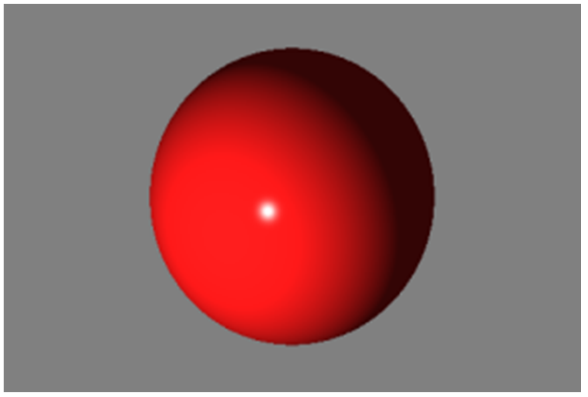
Lancer de rayons: idée générale

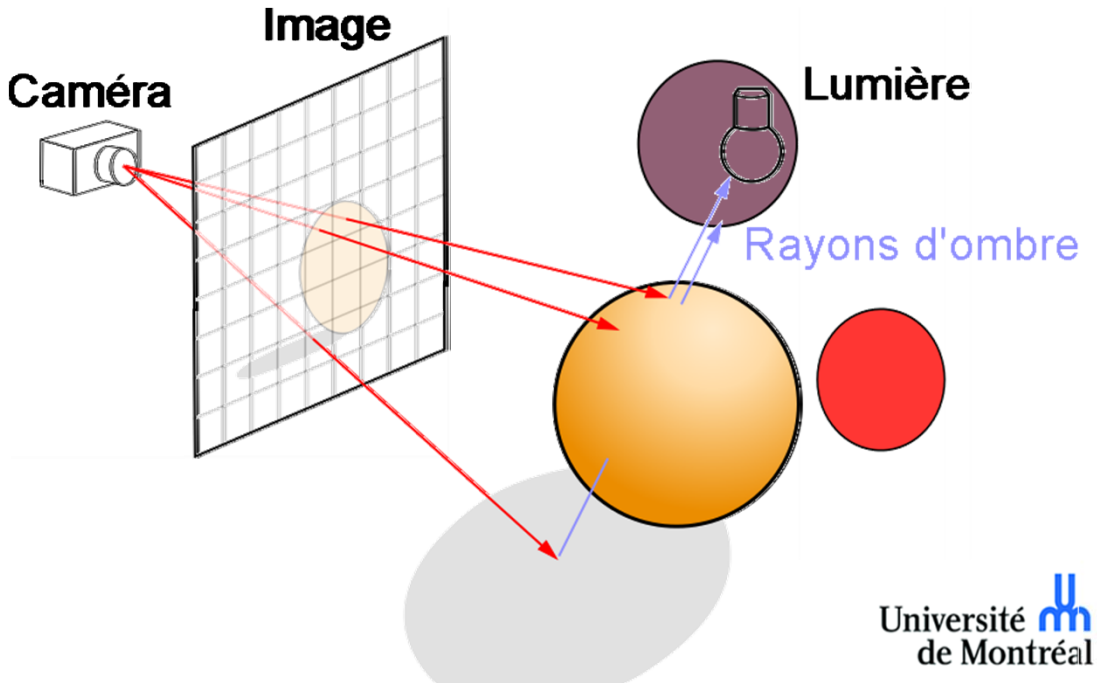
```
for( chaque pixel p )
    Générer un rayon r de l'oeil à p

    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 et h.dist = hit_dist

    p.couleur = shade(h)
```





Université de Montréal

Lancer de rayons

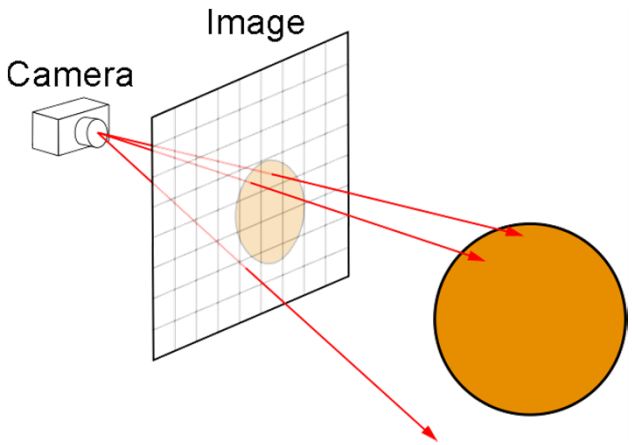
Visibilité:

1. Générer les rayons de l'oeil
2. Déterminé la fonctionnalité de **`trace`** pour
   - des plans
   - des sphères
3. Trouver l'intersection la plus proche

Illumination (locale, globale):

4. Qu'est-ce qu'on peut accomplir avec **`shade`**?

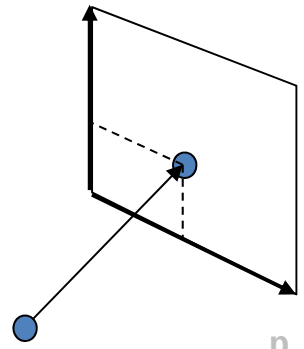
```
for( chaque pixel p )
  Générer rayon r de l'oeil à p
  float dist_min = infinité
  intersection h = null
  for( chaque objet o )
    if( trace( r, o ) et
        hit_dist <= dist_min )
      dist_min = hit_dist
      h.objet = o
      h.dist = hit_dist
p.couleur = shade(h)
```



Université de Montréal

La forme générale d'un rayon

```
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)
```



- Un rayon est défini par
  - une origine (x_0, y_0, z_0)
  - une direction $(\Delta x, \Delta y, \Delta z)$
- Pour les rayons de l'oeil:
  - défini par la position du caméra et un point sur la fenêtre (x_1, y_1, z_1)
- Sous forme paramétrique un rayon s'exprime comme

$$P(t) = P_0 + t(P_1 - P_0)$$

$$x = x_0 + t\Delta x \quad y = y_0 + t\Delta y \quad z = z_0 + t\Delta z$$

$$\Delta x = x_1 - x_0 \quad \Delta y = y_1 - y_0 \quad \Delta z = z_1 - z_0$$

Université de Montréal

Intersection avec un plan

- L'intersection d'un rayon avec un plan (*A,B,C,D*) consiste à exprimer *les points sur le plan* en fonction des points sur le rayon qui nous intéressent

$$Ax + By + Cz + D = 0$$

$$A(x_0 + t\Delta x) + B(y_0 + t\Delta y) + C(z_0 + t\Delta z) + D = 0$$

$$t = \frac{-(Ax_0 + By_0 + Cz_0 + D)}{A\Delta x + B\Delta y + C\Delta z}$$

- Pour l'intersection avec un polygone, il faudra déterminer si le point d'intersection est à l'intérieur du polygone

```
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)
```

Université de Montréal

Intersection avec une sphère

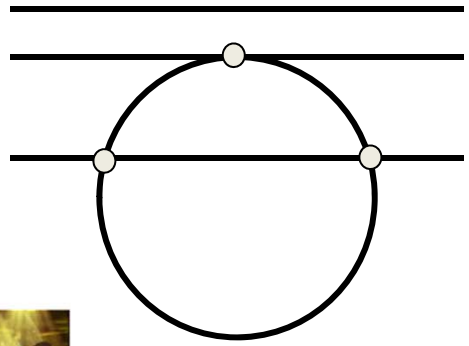
- Similairement pour l'intersection d'un rayon avec une sphère

$$x^2 + y^2 + z^2 = 1$$

$$(x_0 + t\Delta x)^2 + (y_0 + t\Delta y)^2 + (z_0 + t\Delta z)^2 = 1$$

$$(\Delta x^2 + \Delta y^2 + \Delta z^2)t^2 + 2(x_0\Delta x + y_0\Delta y + z_0\Delta z)t + (x_0^2 + y_0^2 + z_0^2 - 1) = 0$$

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



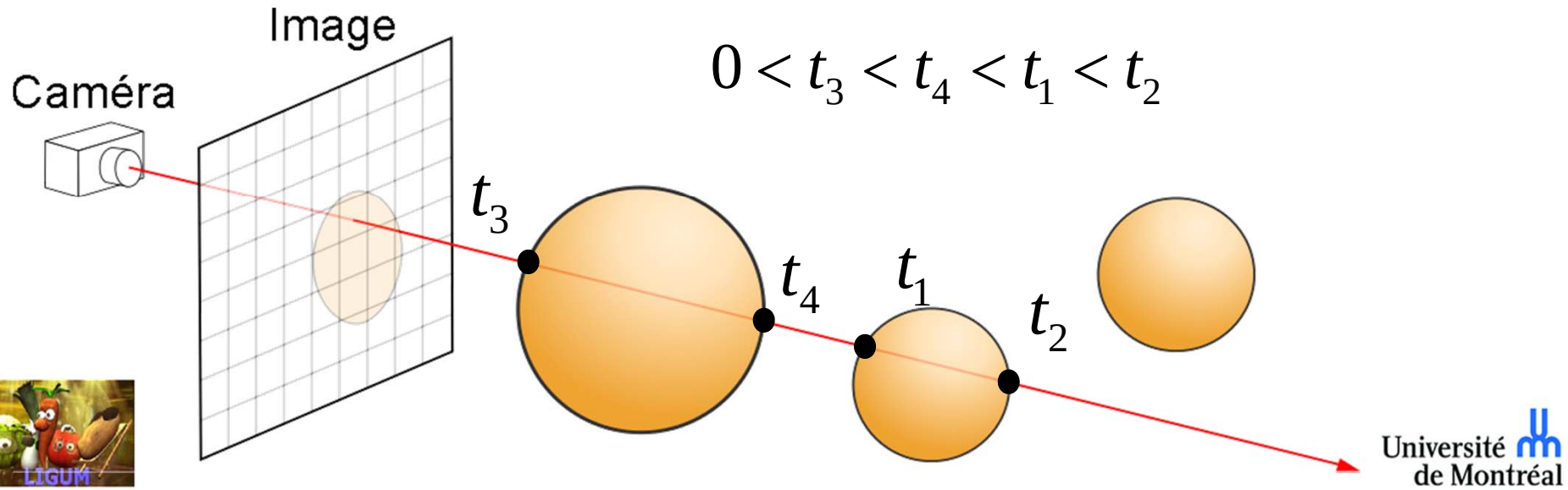
0, 1 ou 2 intersections réelles

Université de Montréal

Visibilité au pixel

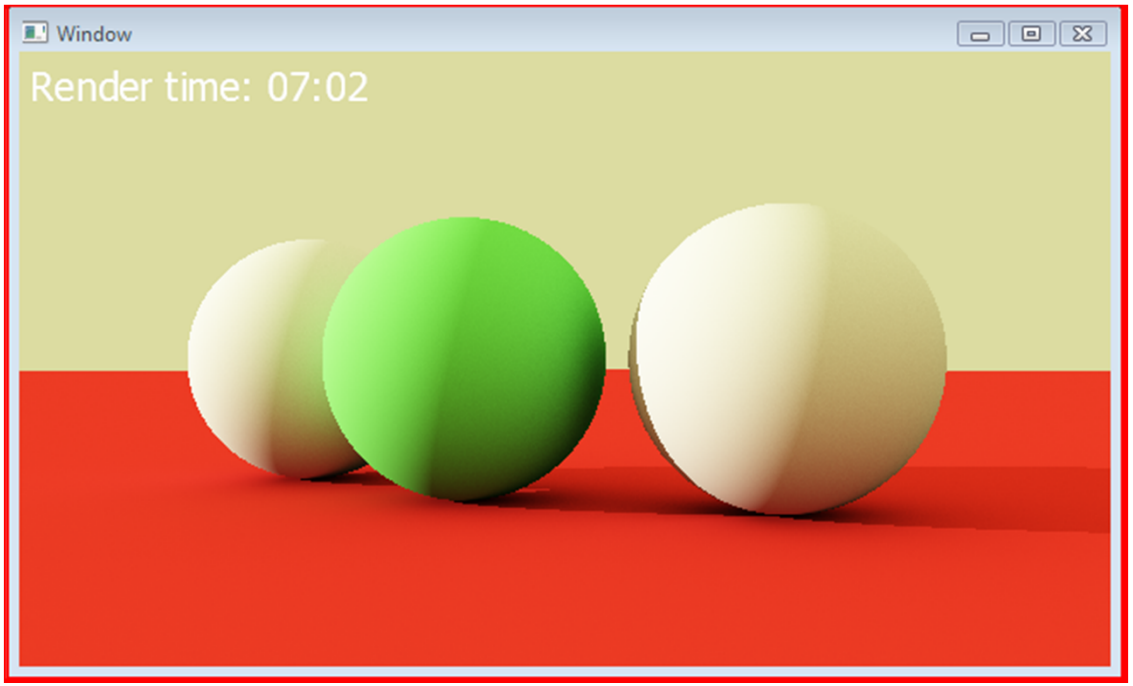
- Pour identifier la surface visible du pixel/caméra, il suffit d'intersecter *tous* les objets et de ne conserver que l'intersection ayant la plus petite valeur positive de *t*

```
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)
```



Université de Montréal

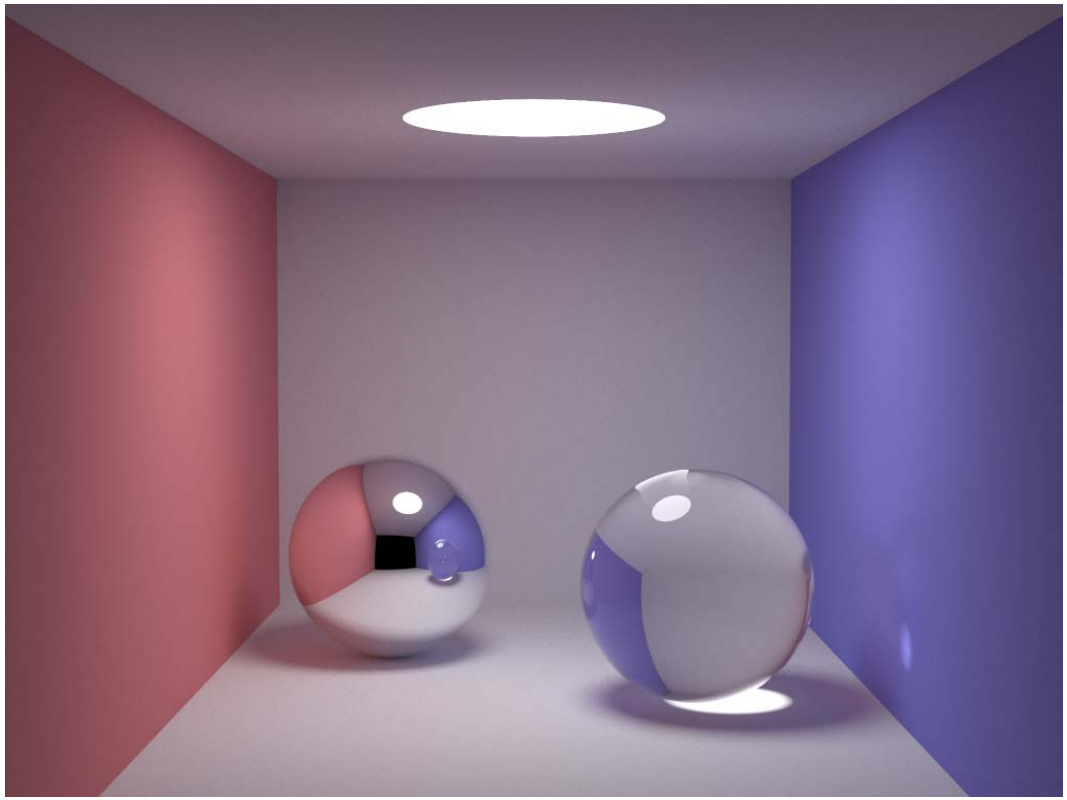
Visibilité au pixel




```
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)
```

Université de Montréal

On n’peut rien faire avec des sphères/plans…

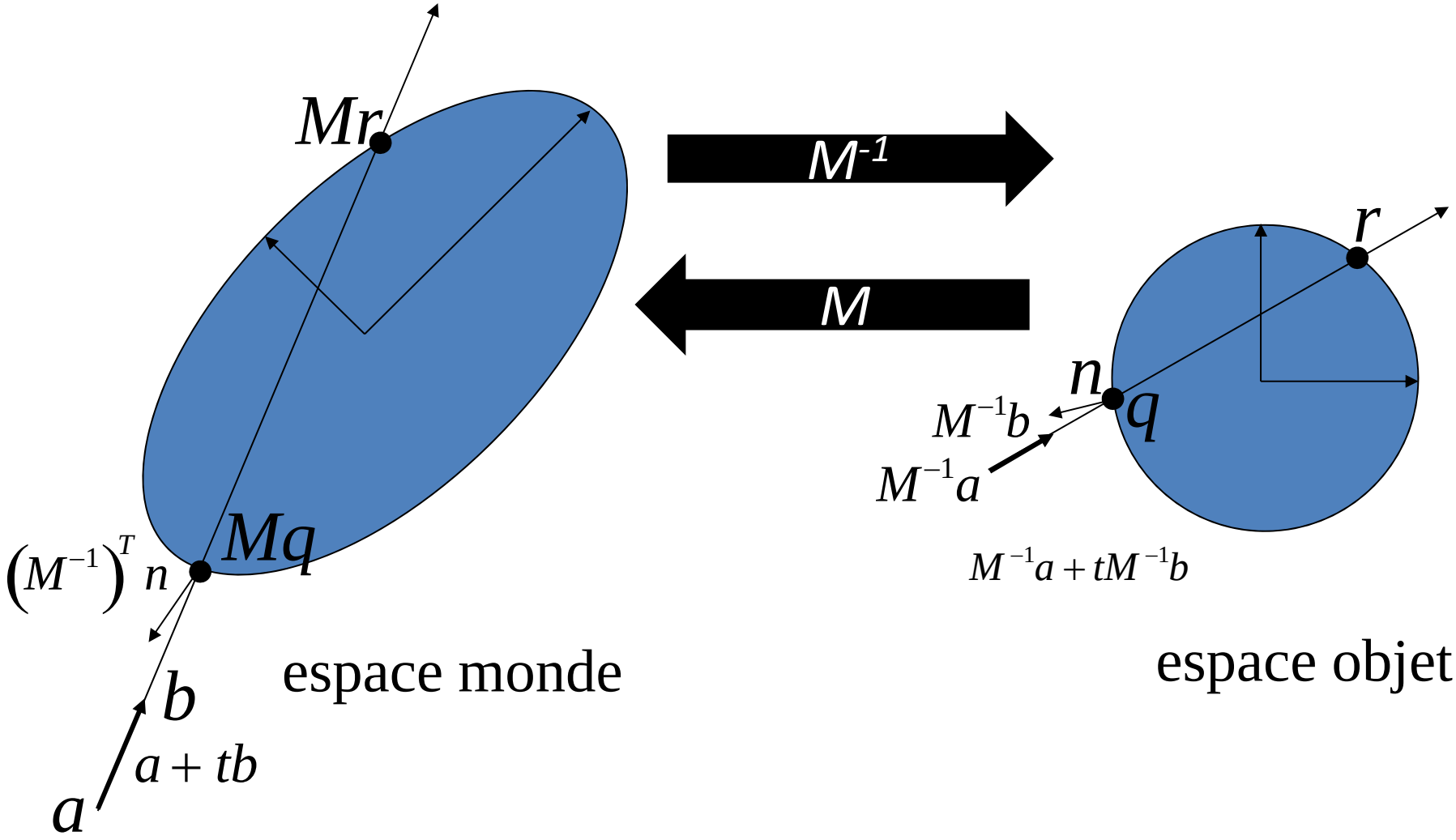


Générer avec seulement 99 lignes de code!

http://kevinbeason.com/smallpt/

Université de Montréal

Intersection en espace objet



Université de Montréal

Accélération d'un lanceur de rayons

Rayon - objet

algorithmes d'intersection
- volumes englobants

Moins d'intersections par rayon

- subdivisions d'espace
techniques directionnelles
- hiérarchies de volumes englobants

Moins de rayons

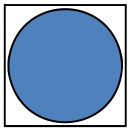
contrôle de profondeur
statistiques

Rayons généralisés

faisceau
conique
paraxial

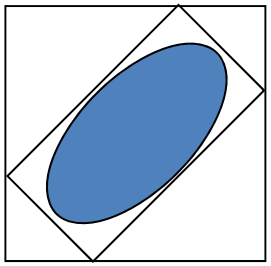
Université de Montréal

Volumes englobants

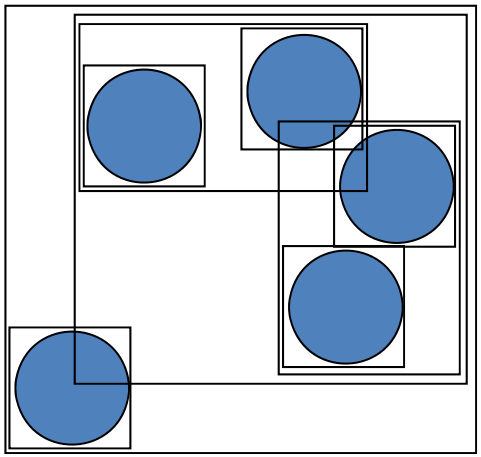


espace objet

boîte orientée
(*OBB*)



boîte alignée sur les axes
(*AABB*)



hiérarchie de boîtes englobantes
(*BVH*)