Chapitre 1

Le lancé de rayon

Contenu du chapitre

Le lancé de rayon (ou ray-tracing) est une méthode de rendu qui utilise les lois de l'optique géométrique. Elle consiste à suivre le trajet inverse des rayons lumineux afin de calculer les propriétés géométriques et lumineuses de la scène.

1 L'algorithme de base : le raycasting

1.1 Principe

La méthode de base utilise le fait que dans un milieu homogène, le déplacement des rayons lumineux se fait en ligne droite.

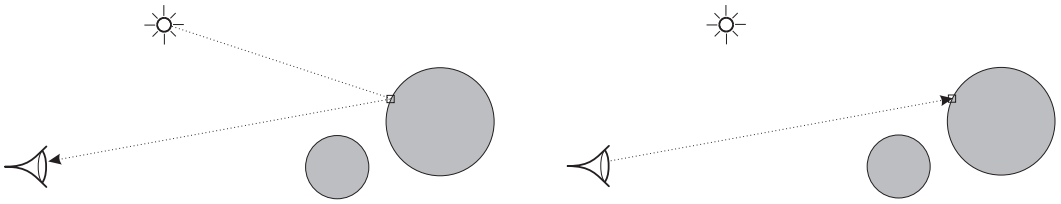
Pour suivre le trajet d'un rayon lumineux, il suffit donc de se déplacer sur une droite. Le rayon présenté à la figure 1.1(a) suit une trajectoire **LDE** (on suppose dans l'exemple que toutes les réflexions dans la scène sont diffuses). L'objet est perçu par l'oeil à cet endroit parce qu'un rayon lumineux direct va en ligne droite depuis sa surface jusqu'à l'oeil.

Une droite partant de l'oeil et allant exactement dans la direction dont provient le rayon intersecte l'objet à l'endroit exact où celui-ci est vu.

1.2 Définition d'un lanceur de rayon

Le suivi inverse d'un rayon nous permet de connaître ce qui est vu dans une direction donnée (la direction de ce rayon) à partir depuis une position d'observation (l'origine du rayon).

Reprenons maintenant la définition de la caméra que nous avions donnés à la section **??** (on reprend les notations de cette section). L'observateur est placé en un point Ω. Ce qu'il voit est l'image qui se forme par projection de la scène mise en perspective sur le plan $\mathcal{P}$. Trivialement, si on lance un rayon R depuis Ω dans la direction $\mathbf{u}$, alors



(a) Les objets que nous percevons directement sont vus car un rayon lumineux va droit de la surface de l'objet à notre oeil.

(b) Si un rayon est envoyé depuis l'oeil dans la direction exacte d'où venait le rayon perçu, celui-ci intersecte l'objet.

FIG. 1.1 – Principe du raycasting

au point d'intersection du rayon R et du plan $\mathcal{P}$, on observe exactement ce qui est "vu" par le rayon.

Inversement, on considère maintenant la grille discrète de l'écran sur le plan de projection (chaque carré de cette grille représente un pixel de notre écran). Si on lance un rayon depuis Ω passant par le centre d'un de ces carrés, alors ce que "voit" ce rayon est ce qu'il faut[1] afficher dans le pixel correspondant sur l'écran. En conséquence, il nous faut lancer un rayon par pixel (voir figure 1.2).

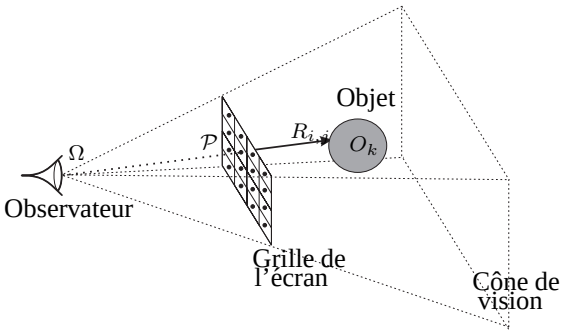


FIG. 1.2 – Principe du raycasting

En reprenant le modèle de caméra que nous avions défini, le rayon $R_{i,j}(t)$ associé au pixel $P_{i,j}$ de l'écran se définit comme :

$$R_{i,j}(t) = \Omega + t.\mathbf{u}_{i,j} \qquad \text{où } t > 0$$

où $\mathbf{u}_{i,j}$ est la direction dans laquelle on lance le rayon pour qu'il passe par le centre du carré de coordonnées (i, j) dans la grille, à savoir :

$$\mathbf{u}_{i,j} = d.\mathbf{U} + i.\delta.\mathbf{W} + j.\delta.\mathbf{V}$$

où $(\mathbf{U}, \mathbf{V}, \mathbf{W})$ est la base construite à partir de la direction $\mathbf{U}$ du regard de l'observateur et de la direction $\mathbf{V}$ du haut, d est la distance focale (calculée à partir de la largeur du champ de vision φ), et δ est la finesse de la grille qui dépend de la résolution de l'image à calculer.

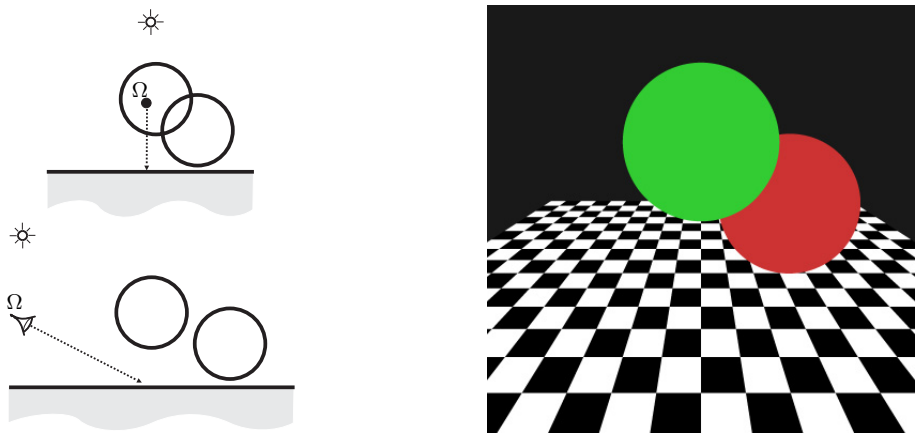
[1] Oui, je mens effrontément. Encore une fois, on ne peut pas déduire du comportement en un seul point du carré (le centre du pixel) ce qui se passe dans tout le carré. On négligera cette question dans un premier temps. On y apportera une réponse dans le chapitre sur l'aliasing.

1.3 Algorithme

La détermination des objets visibles dans une scène par la méthode du raycasting (intégrant le lanceur de rayon) est donc la suivante :

$\{I_{i,j}\}$ = RAYCASTING OBSERVATEUR $(\Omega, \mathbf{U}, \mathbf{V}, \varphi)$ OBJETS $\{O_k\}_{k=1...n}$
pour tout pixel (i, j) de l'image
  calculer l'équation $R_{i,j}(t) = \Omega + t.\mathbf{u}_{i,j}$ du rayon associé.
  $\{(t_p, O_p)\} = \{(t_p, O_p)$ tels que $R_{i,j}(t_p)$ soit la première intersection du rayon avec l'objet $O_p\}$
  si l'ensemble des intersections est vide
  alors $I_{i,j}$ = couleur du fond.
  sinon (t^*, O^*) = parmi tous les couples $\{(t_p, O_p)\}$, celui qui a le plus petit t.
    $P = R_{i,j}(t^*)$ la position du point d'intersection entre le rayon et l'objet O^*.
    $I_{i,j}$ = couleur de l'objet O^* au point P.

Algorithme 1: Algorithme du raycasting



(a) Exemple de scène (b) Résultat

FIG. 1.3 – Exemple de raycasting :

1.4 Commentaires

On notera à propos de cet algorithme que :

– On remonte **seulement** les rayons lumineux diffus directs entre les objets et l'oeil. A noter que lors d'une réflexion diffuse sur une surface visible par l'observateur, **un seul** rayon de la réflexion diffuse arrive jusqu'à l'oeil (celui qui est exactement dans la direction de l'oeil). Dans le cas du raycasting, aucun rendu

n'est effectué : les objets sont vus à lumière constante (*i.e.* la seule source lumineuse est la lumière ambiante, et celle-ci est constante partout).
- Si tous les rayons partent d'un point d'observation 0 et passent à travers une grille située sur un plan de projection, alors on recrée naturellement une projection à un point de fuite. Il n'est donc pas nécessaire de faire une mise en perspective ; elle est naturellement incluse dans la méthode du lancé de rayon.
- L'algorithme ne demande pas qu'un objet soit décomposé en facettes pour fonctionner. Pour la plupart des objets canoniques, les solutions exactes de calcul d'intersection entre l'objet et une demi-droite sont connues explicitement (voir la section 3). La représentation de l'objet est exacte.
- Il paraît évident que l'algorithme passe la plus grande partie de son temps à calculer des intersections. Plusieurs types d'optimisation peuvent être proposées : dans le calcul même des intersections (voir section 3), et par des stratégies visant à limiter le nombre de ces calculs (voir section 4).
- Cet algorithme peut être adapté pour effectuer le rendu réaliste d'une scène en déterminant le trajet des rayons lumineux présents dans une scène. Il constitue l'une des solutions déterministes au problème de l'illumination globale. On lui donne alors le nom de *ray-tracing*.

2 Ray-tracing et rendu

2.1 Justification

Le principe du raycasting peut être étendu au calcul du rendu sur une scène. Dans l'algorithme présenté en première partie, les seuls rayons considérés étaient les rayons (issus des réflexions diffuses avec un éclairage parfaitement uniforme) directs entre l'oeil et les objets de la scène. Or la connaissance des lois de l'optique nous permet de remonter plus loin dans l'origine des rayons qui parviennent à l'oeil. On peut ainsi déterminer l'origine des rayons qui éclairent la surface observée. Notamment :
- le principe de réciprocité permet ce suivi inverse des rayons lumineux. Selon ce principe, l'inversion du sens d'un rayon lumineux (par inversion de la position de l'émetteur et du récepteur) ne change pas l'énergie reçue pas le récepteur.
- le déplacement en ligne droite des rayons lumineux dans tous les milieux transparents (air, verre, eau) se fait en ligne droite. Le déplacement le long d'un rayon lumineux est donc simple.
- l'utilisation des lois de Snell/Descartes nous permettent de calculer les directions de réflexion/réfraction lorsqu'un rayon lumineux touche une surface. Il devient alors possible de gérer les effets de reflet et de transparence.

– Un point est éclairé par une source de lumière s'il existe un rayon lumineux direct entre la source et ce point. Savoir si une source éclaire directement un point revient donc à regarder si le rayon du point à la source entre en intersection avec un autre objet.

Par conséquent, il va devenir possible d'évaluer l'approximation de l'équation d'illumination globale que nous donnions dans le chapitre **??** en lançant des rayons. On se référera à ce chapitre pour la justification de cette équation. Elle se formule comme suit :

$$L = L_e + L_r^a + \sum_{j=1}^{p} \left[s^j . \left(L_{r,d}^j + L_{r,s}^j \right) + s_t^j . \left(L_{t,d}^j + L_{t,s}^j \right) \right] + L_r^O + L_t^O$$

et est basé sur les approximations suivantes :

1. pour une réflexion diffuse, on ne considère que les rayons issus des éclairages directs.
2. pour une réflexion (resp. une réfraction) spéculaire, on ne considère que le rayon dans la direction principale de réflexion (resp. de réfraction) car c'est dans cette direction que va la plus grande part de l'énergie.
3. un terme d'atténuation diminue l'intensité du rayon lumineux à chaque rebond (absorption).
4. un terme d'intensité lumineuse ambiante modélise la somme des contributions des rayons négligés dans les deux points précédents. Il est constant sur toute la scène.

2.2 Évaluation de l'équation d'illumination globale

Nous abordons maintenant la façon d'évaluer chacun des termes de cette équation en un point P d'un objet O_k. On note $\mathbf{N}$ la normale à la surface au point P (orientée vers l'extérieur de la surface), et $\mathbf{V}$ la direction d'où cette surface est observée (*i.e.* la direction d'où l'on a envoyé le rayon inverse qui intersecte la surface au point P). On construit le vecteur normal $\mathbf{N}'$ à la surface vu pour le rayon inverse : $\mathbf{N}' = \mathbf{N}$ si $\mathbf{N}$ et $\mathbf{V}$ sont du même côté de la surface (*i.e.* $\mathbf{N}.\mathbf{V} > 0$), et $\mathbf{N}' = -\mathbf{N}$ sinon. Le détail est donné dans le cadre du modèle de Hall (voir section **??**). La version simplifiée du modèle de Hall (en gros, en remplaçant les fonctions de Fresnel ρ et ρ' par 1) s'appelle le modèle de Whitted.

termes constants ne dépendant que des caractéristiques de l'objet et de constantes (notamment, la lumière ambiante).

L_e la luminance intrinsèque de l'objet au point P (0 si l'objet O_k n'est pas une source lumineuse).

L_r^a la luminance ambiante au point P dépend de la couleur de l'objet O_k en ce point et de la luminance ambiante.

pour la $j^{\text{ème}}$ source de lumière , on note $\mathbf{L}$ la direction de cette source de lumière vue depuis P, et L_j sa localisation (donc $\mathbf{L} = \mathbf{PL_j}/|PL_j|$).

$s^j = 1$ si la source est observée par réflexion sur la surface. $s^j = 1$ correspond à deux conditions :

$\mathbf{N}'.\mathbf{L} > 0$: $\mathbf{L}$ est du même côté de la surface que l'observateur.

la source L_j n'est pas occultée : le rayon lumineux partant P et allant dans la direction $\mathbf{L}$ n'intersecte aucun objet **opaque** sur son trajet de P à L_j.

$L_{r,d}^j$ **et** $L_{r,s}^j$: les luminances diffuses et spéculaires réfléchies pour cette source. Elles ne sont évaluées que si $s^j = 1$. Elles dépendent exclusivement de la géométrie locale ainsi que des caractéristiques de la source lumineuse et de l'objet.

$s_t^j = 1$ si la source est observée par réfraction à travers la surface. $s_t^j = 1$ si et seulement si :

$\mathbf{N}'.\mathbf{L} < 0$: $\mathbf{L}$ est de l'autre côté de la surface par rapport à l'observateur.

$k_t \neq 0$ la surface est transparente. Cela se traduit par un indice de réfraction k_t non nul.

la source L_j n'est pas occultée : le rayon lumineux partant de P et allant dans la direction $\mathbf{L}$ n'intersecte aucun objet **opaque** sur son trajet de P à L_j.

L'aspect exclusif de s^j et s_t^j apparaît donc ici clairement.

$L_{t,d}^j$ **et** $L_{t,s}^j$: les luminances diffuses et spéculaires réfractées pour cette source. Elles ne sont évaluées que si $s_t^j = 1$. Elles dépendent exclusivement de la géométrie locale ainsi que des caractéristiques de la source lumineuse et de l'objet.

pour les objets : L'évaluation de la luminance en provenance des objets va nécessiter l'évaluation de la luminance en d'autres points de la scène, et oblige à une évaluation récursive.

L_r^O luminance réfléchie dans la direction d'arrivée du rayon $\mathbf{V}$ par un objet. Cette luminance s'évalue comme suit si le coefficient de transmitivité par réflexion n'est pas nul :

– calculer le vecteur réfléchi $\mathbf{R_V}$ de $\mathbf{V}$.

- envoyer un rayon depuis P dans la direction $\mathbf{R_V}$ afin de calculer la luminance dans cette direction.
- évaluer L_r^O à partir de cette luminance, de la distance à l'objet qui a renvoyé cette luminance, et des caractéristiques de la surface.

L_t^O luminance transmise au point P vers la direction d'arrivée du rayon $\mathbf{V}$ par un objet de l'autre côté de la surface. Cette luminance s'évalue comme suit si la surface est transparente et que le coefficient de transmitivité par réfraction n'est pas nul :

- calculer le vecteur transmis de $\mathbf{T_V}$ de $\mathbf{V}$.
- envoyer un rayon depuis P dans la direction $\mathbf{T_V}$ afin de calculer la luminance dans cette direction.
- évaluer L_t^O à partir de cette luminance, de la distance à l'objet qui a renvoyé cette luminance, ainsi que des caractéristiques de la surface et des milieux traversés.

2.3 L'algorithme du ray-tracing

L'algorithme du ray-tracing est une fusion entre le lanceur de rayon du raycasting (voir section 1) et la méthode d'évaluation de l'équation d'illumination globale exposée dans la section précédente. Il est composé de deux parties :

un lanceur de rayon : pour chaque pixel $I_{i,j}$ de l'écran, envoyer le rayon $R_{i,j}$ correspondant. La luminance associée au pixel $I_{i,j}$ est la luminance observée dans la direction du rayon $R_{i,j}$ depuis la position Ω de l'observateur.

une fonction de calcul de luminance : qui évalue la luminance reçue depuis la direction $\mathbf{u}$ par un point P. Cette fonction est récursive car elle peut avoir besoin d'aller chercher la luminance en d'autres points de la scène pour évaluer la luminance au point P. On fixe alors un nombre r d'appels récursifs maximum afin d'éviter d'éventuelles explosions algorithmiques (reflet d'un miroir dans un miroir à l'infini). On utilise également une pile de milieu m afin de gérer simplement les changements de milieu que le rayon lumineux doit subir tout au long de son trajet (nous repréciserons ce point). Voir l'algorithme 2 pour le principe de base.

Nous présentons à la figure 1.4 des résultats de raytracing dans un cas avec et sans objet transparent. On y observe l'effet du nombre d'appels récursifs maximum sur le résultat du rendu.

Nous complétons cet algorithme par quelques remarques :

1. Comme il est aisé de savoir quelle est la distance parcourue par le rayon lumineux entre les différents rebonds, il est assez courant de pondérer les lumi-

CALCULLUMINANCE(RAYON $(A, \mathbf{u})$, OBJETS $\{\mathcal{O}_k\}_{k=1..p}$, LUMIÈRES $\{L_i\}_{i=1..n}$, PILE m, ENTIER r)
P = INTERSECTION(RAYON , OBJETS)
si il n'y a pas d'intersection
alors renvoyer $L =$ couleur du fond
sinon *(* évaluation de la luminance au point d'intersection *)*

$\mathcal{O}^*$ = objet sur lequel se trouve le point P.
$\mathbf{N}$ = normale à la surface de l'objet $\mathcal{O}^*$ au point P
$\mathbf{V} = \mathbf{PA}$ *(* direction d'arrivée du rayon *)*
si $\mathbf{V}.\mathbf{N} < 0$ **alors** $\mathbf{N}' = -\mathbf{N}$
sinon $\mathbf{N}' = \mathbf{N}$
$L = L_e + L_r^a$
pour chaque source de lumière L^j

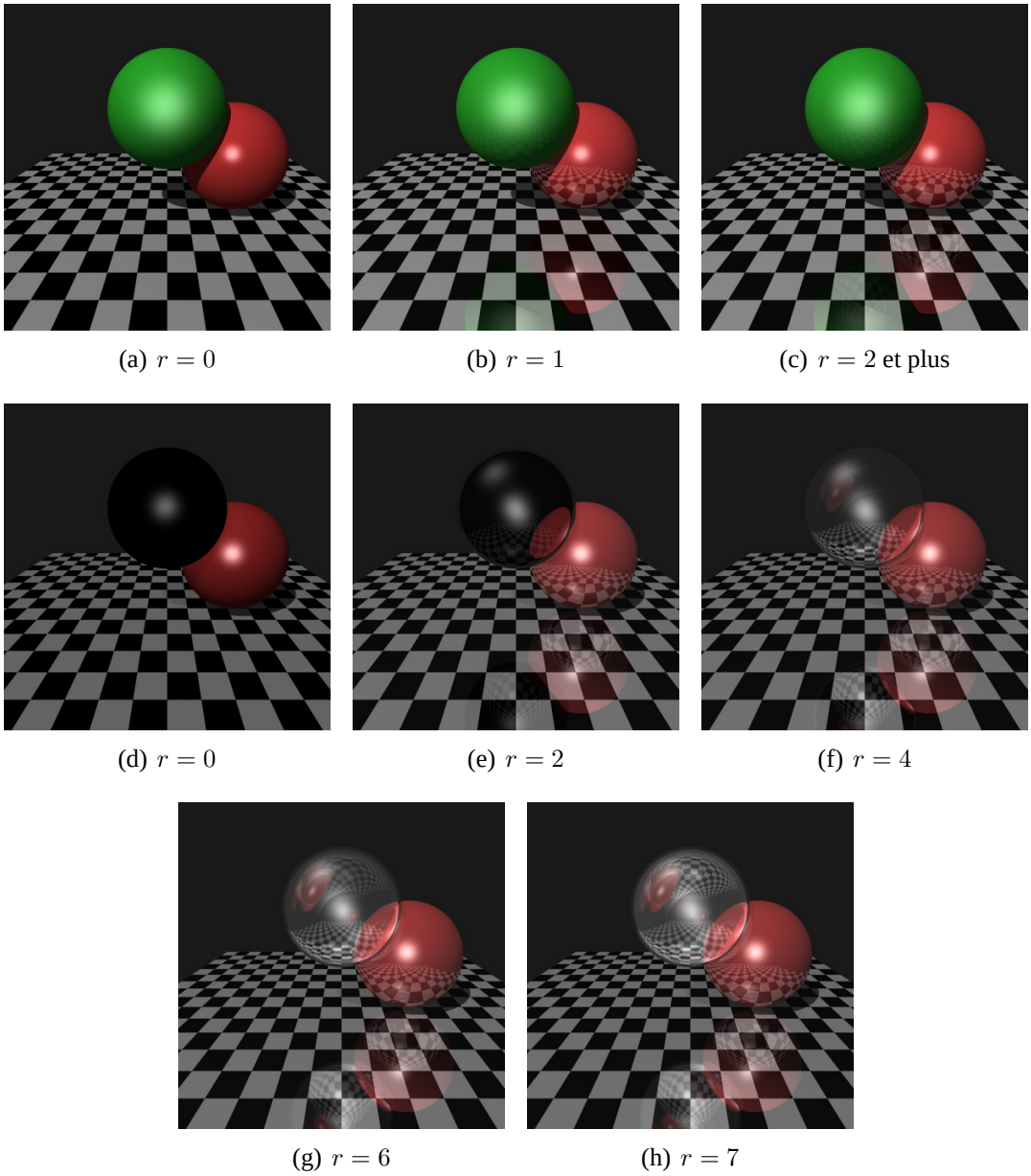
$\mathbf{L_j} = \mathbf{PL_j}$ *(* direction de la source de lumière *)*
si $\mathbf{L_j}.\mathbf{N}' > 0$
alors *(* éclairage par réflexion *)*
s^j = RAYONOMBRE(RAYON $(P, \mathbf{L_j})$, POINT L_j, OBJETS)
si $s^j = 1$ **alors** $L = L + (L_{r,d}^j + L_{r,s}^j)$
sinon *(* éclairage par transparence *)*
si $\mathcal{O}^*$ est transparent ou translucide
alors s_t^j = RAYONOMBRE(RAYON $(P, \mathbf{L_j})$, POINT L_j, OBJETS)
si $s_t^j = 1$ **alors** $L = L + (L_{t,d}^j + L_{t,s}^j)$

si $r \geq 1$ **alors** *(* contribution des objets *)*

si $\mathcal{O}^*$ est réfléchissant
alors $\mathbf{R_V}$ = direction de réflexion pour $\mathbf{V}$
CALCULLUMINANCE(RAYON$(P + \epsilon.\mathbf{R_V}, \mathbf{R_V})$, OBJETS, LUMIÈRES, m , $r-1$)
$L = L + L_r^O$
si $\mathcal{O}^*$ est transparent
alors $\mathbf{T_V}$ = direction de réfraction pour $\mathbf{V}$
si $\mathbf{N} = \mathbf{N}'$ **alors** EMPILER le milieu de $\mathcal{O}^*$ dans m
sinon DÉPILER m
CALCULLUMINANCE(RAYON$(P + \epsilon.\mathbf{T_V}, \mathbf{T_V})$, OBJETS, LUMIÈRES, m , $r-1$)
$L = L + L_t^O$

renvoyer L

Algorithme 2: Fonction récursive de calcul de luminance du raytracing



(a) $r = 0$ (b) $r = 1$ (c) $r = 2$ et plus

(d) $r = 0$ (e) $r = 2$ (f) $r = 4$

(g) $r = 6$ (h) $r = 7$

FIG. 1.4 – Exemple d'images obtenues par la méthode du raytracing en fonction du nombre de récursions autorisées. (a)-(c) sans objets transparents. (d)-(h) avec des objets transparents. La sphère transparente est faite d'une mince couche de verre.

nances obtenues par des fonctions d'atténuation (voir **??**) afin de tenir compte des atténuation atmosphériques. Ces atténuations peuvent intervenir en pondérant les luminances par une fonction dépendant :

– de la distance parcourue par le rayon. On atténue ainsi la luminance d'une source en fonction de sa distance au point éclairé, ou la luminance totale renvoyée par les objets (ce dernier est inclus dans le modèle de Hall à travers le coefficient de transmitivité).
– du nombre d'objets transparents traversés dans le cas des sources lumineuses.

Pour le calcul les différentes intensités, différents modèles sont présentés à la section **??**. Les fonctions d'atténuation des rayons lumineux sont souvent omises des modèles pour simplifier les expressions. Penser à les réintégrer lors du calcul des intensités.

2. Les fonctions supplémentaires définies dans l'algorithme ont le rôle suivant :

   INTERSECTION(RAYON,OBJETS) calcule l'intersection entre le rayon et tous les objets de la scène, et renvoie, si elle existe le point d'intersection le plus proche de l'origine du rayon.

   RAYONOMBRE(RAYON,POINT,OBJETS) renvoie la valeur 1 si le rayon n'intersecte aucun objet **opaque** sur son trajet en l'origine du rayon et le point considéré. Il est préférable d'écrire une fonction différente de la fonction d'intersection puisque celle-ci peut s'arrêter dès qu'une intersection est trouvée.

   On peut écrire des versions évoluées de cette fonction afin de savoir comment modifier la luminance de la source en fonction des transmittances spectrales des objets transparents traversés (au plus simple, leurs produits ; voir aussi la remarque sur les atténuations). Par exemple, une source lumineuse blanche observée à travers un rubis semblera être de couleur rouge (voir dans page ? ? le fonctionnement du terme $s^j(\lambda)$ dans le modèle de Hall).

   DÉPILER et EMPILER permettent de gérer les transitions entre les différents environnements. Au moment du lancement, on place dans la pile le milieu dans lequel l'origine du rayon se trouve. Lorsqu'on pénètre dans une surface, on empile son milieu. Lorsqu'on en sort, on la dépile. Sur l'exemple de la figure 1.5(a), entre b et c la pile de milieux contient $m = \{m_3, m_2, m_1\}$; on se trouve dans le milieu m_3 (placé au sommet de la pile). Lorsqu'on sort de la sphère intérieure en c, on dépile m. m devient $\{m_2, m_1\}$. On se trouve donc dans le milieu m_2. Ce type de gestion ne reste cohérent que si, soit il n'y a pas d'intersection entre les objets (inclusion : cas du (a)), soit les objets qui s'intersectent ont le même milieu (intersection : le résultat du (b)

est correct si $m_1 = m_2$, sinon le milieu de l'intersection change en fonction de l'objet qui est intersecté en premier).

On notera que dans les versions évoluées du modèle de Hall, un milieu est défini par deux valeurs : l'indice de réfraction $\dot{\eta} = \dot{\eta}_t/\dot{\eta}_i$ et de transmitivité du milieu T. Comme spécifié dans l'algorithme du raytracing, il faut déterminer si le rayon est entrant ($\mathbf{N} = \mathbf{N}'$) ou sortant de l'objet ($\mathbf{N} = -\mathbf{N}'$) pour déterminer les calculs à effectuer. On note $\texttt{Sommet}(m)$ le sommet de la pile m et $\texttt{Second}(m)$ le suivant du sommet :

si le rayon est entrant :

$\dot{\eta} = \dot{\eta}_{\text{objet}}/\dot{\eta}_{\texttt{Sommet}(m)}$

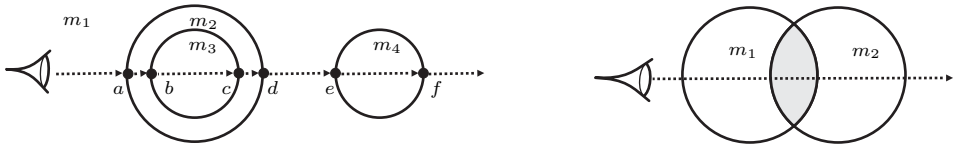
Pour les rayons réfléchis ($\mathbf{R_V}$ ou $\mathbf{L_j}$), utiliser la transmitivité $T_{\texttt{Sommet}(m)}$.
Pour les rayons réfractés ($\mathbf{T_V}$ ou $\mathbf{L_j}$), utiliser la transmitivité T_{objet} et empiler le couple $(T_{\text{objet}}, \dot{\eta}_{\text{objet}})$ avant de relancer le rayon.

si le rayon est sortant :

$\dot{\eta} = \dot{\eta}_{\texttt{Second(m)}}/\dot{\eta}_{\texttt{Sommet(m)}}$

Pour les rayons réfléchis ($\mathbf{R_V}$ ou $\mathbf{L_j}$), utiliser la transmitivité $T_{\texttt{Sommet}(m)}$.
Pour les rayons réfractés ($\mathbf{T_V}$ ou $\mathbf{L_j}$), utiliser la transmitivité $T_{\texttt{Second}(m)}$ et dépiler m avant de relancer le rayon.

Attention, dans le cas où l'observateur de la scène est placé à l'intérieur d'un objet, il faut initialiser correctement la pile sinon la pile de milieu risque de se vider complètement. Par exemple, sur la figure 1.5, à l'extérieur des objets, la pile est initialisée à $\{m_1\}$. A l'intérieur de la double sphère, la pile doit être initialisée à $\{m_3, m_2, m_1\}$.

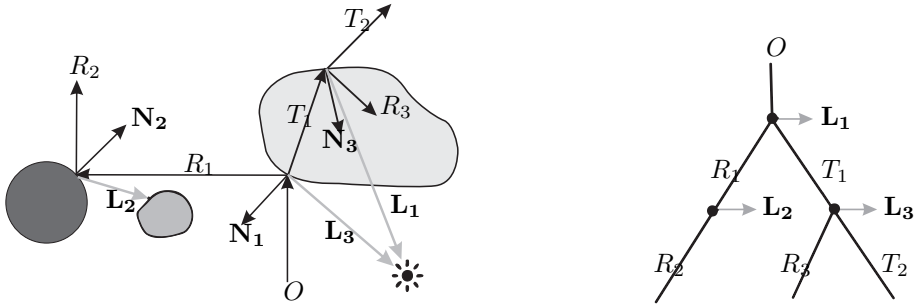


(a) Exemple de succession de changement de milieu.

(b) Ce qu'il ne faut pas faire si on utilise une pile.

FIG. 1.5 – Gestion de la pile de milieux.

3. Lors du rendu dans une scène contenant beaucoup de réflexions/réfractions, le nombre de rayons relancés à partir du rayon initial peut aller jusqu'à $(n+2).2^r$ (où r est le niveau maximal de récursion autorisé et n le nombre de sources de lumière), sans compter les rayons d'ombre. La gestion des réflexions/réfractions multiples a donc un coût important (cf figure 1.6). En pratique, pour les scènes comportant beaucoup de réflexions, on ne va pas au delà de $r = 16$.



(a) Ensemble des rayons lancés pour $r = 2$. (b) Arbre de dépendance des rayons lancés.

FIG. 1.6 – Exemple de rayon lancé avec trois objets dont un transparent et une source de lumière

(a) Le problème : l'erreur numérique dans la position du point peut provoquer des intersections indésirables. (b) La solution : on déplace légèrement l'origine du rayon.

FIG. 1.7 – Problème d'erreurs numériques.

4. A cause des problèmes inhérents aux calculs numériques, la position de l'intersection trouvée **n'est pas exactement** l'intersection sur la surface. Aussi, il est possible lorsqu'on renvoie un rayon depuis la surface d'intersecter à nouveau la surface quasiment au même point (voir figure 1.7). Un symptome de ce phénomène est souvent la présence de points noirs sur une surface. Solution : plutôt que de relancer le rayon $(A, \mathbf{u})$, on relance le rayon $(A', \mathbf{u})$ avec $A' = A + \epsilon.\mathbf{u}$ avec ϵ petit ($\epsilon = 10^{-5}$ suffit). Le déplacement de A à A' permet d'être sûr que le rayon est bien lancé depuis l'autre côté de la surface.

5. Un phénomène d'**aliasing** apparaît lors de l'utilisation de cet algorithme à cause de l'écart entre chaque rayon et du fait qu'un objet très brillant est toujours visible même s'il est très loin. Donc, si la projection d'un tel objet est de taille inférieure à un pixel, sa contribution à l'intensité du pixel est importante, et il doit être visible. Or, si tel est le cas, les rayons lancés ont toutes les chances de manquer leurs cibles.

   Une solution à ce problème consiste à munir chaque objet d'un flag et d'une boîte englobante artificiellement augmentée de façon à ce qu'un rayon intersecte nécessairement cette boîte (petit exercice de géométrie utilisant les paramètres de la caméra : si D est la distance à l'observateur, le rayon R de la sphère englobante à choisir est $R = \sqrt{2}.\delta.D/d$). Ainsi, lors de l'algorithme du raytracing :
   - pendant le calcul des intersections, si l'on intersecte le boîte englobante sans intersecter l'objet, on active le flag.
   - si un flag a été activé, on relance des rayons jusqu'à le toucher et on moyenne les luminances trouvées sur tous les rayons lancés (voir le chapitre **??** sur l'aliasing et les stratégies à adopter dans ce cas pour relancer les rayons et pour calculer la luminance).

6. Les sources de lumière de taille inférieure à la taille d'un pixel (donc les sources ponctuelles) doivent être traitée spécifiquement, sinon les trajets **LE** ne seront pas visible sur l'image. Par conséquent, pour chaque source "ponctuelle" dans le champs de vision de l'observateur :
   - déterminer dans quel pixel cette source apparaît.
   - lancer un rayon d'ombre pour savoir si cette source est visible.
   - si elle l'est, évaluer la luminance associée à ce rayon.
   - accumuler les valeurs obtenues à celle déjà existante dans le pixel de façon similaire à ce qui est fait pour le suréchantillonnage (voir le chapitre sur l'aliasing).

   A noter qu'on ne peut pas utiliser telle quelle la même méthode que dans la remarque précédente sur les objets brillants car il n'y a aucune chance de toucher un objet ponctuel en renvoyant des rayons au hasard. Un "truc" consiste alors à

entourer chaque source ponctuelle par une petite sphère (ce qui n'est pas sans conséquence, car la source devient alors non ponctuelle).

2.4 Forces et limites de la méthode à travers un exemple

En suivant la notation de Heckbert, les trajectoires considérées par ce modèle sont les suivantes :

LDE : vue d'un objet par éclairage direct d'une source lumineuse.
LDSE : reflet d'un objet sur un objet brillant.
LSE : reflet d'une source de lumière sur une surface brillante.
LDSSE : reflet d'un objet à travers deux surfaces brillantes consécutives ou vue d'un objet à travers un objet transparent.

Si ***** représente un nombre quelconque de répétitions (y compris aucune) du caractère précédent, alors l'algorithme du ray-tracing modélise les trajectoires **LDS*E** et **LS*E** (ou **L(D|S)S*E**)[2].

En pratique, et en particulier à cause de l'atténuation des rayons, on fixe le nombre maximum r de rebonds considérés et le ***** de la notation précédente a donc pour sens : "quelques uns".

Nous analysons les résultats obtenus par la méthode du ray-tracing sur un exemple à travers le suivi de 7 rayons lumineux dans une scène. La scène est constituée de 3 objets :

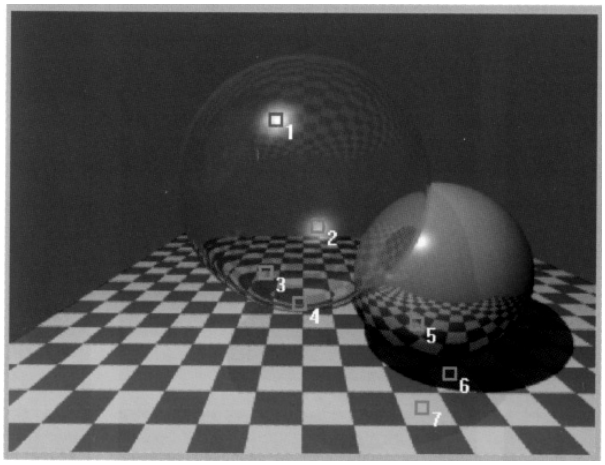
- un damier mat bicolore.
- une sphère opaque brillante.
- une sphère de verre parfaitement transparente, d'épaisseur assez faible. L'intérieur est constitué d'air.

Nous détaillons les phénomènes locaux mis en jeu ainsi que quelques aberrations associées à cette méthode :

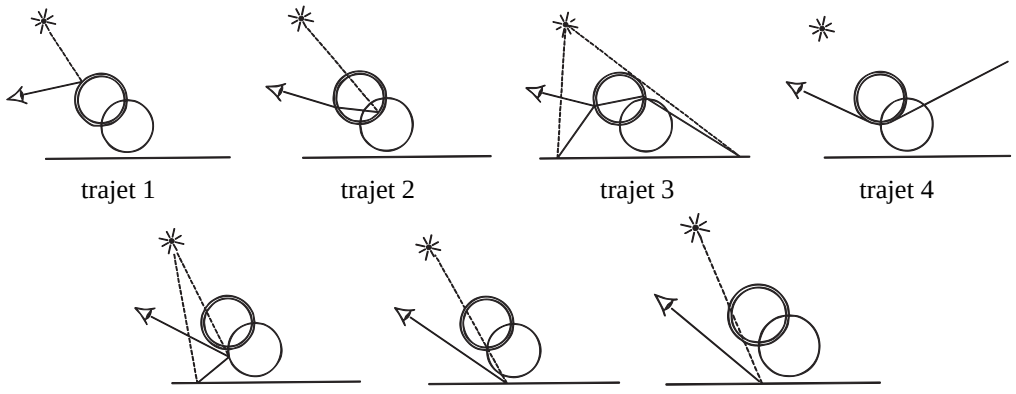
Trajet 1 : Reflet de la source lumineuse sur la sphère transparente.
La sphère étant parfaitement transparente, sa spécularité est très grande, pourtant une tache lumineuse large apparaît. Ceci est lié à la différence des paramètres des modèles entre les réflexions spéculaires locales et celles calculées récursivement. Cette différence est volontaire et destinée à augmenter l'impression de brillance des objets en utilisant des taches larges.

Trajet 2 : Reflet de la source lumineuse provenant de l'intérieur de la source transparente.
Le rayon d'ombre lancé à travers la sphère transparente va en ligne droite à la

[2] mais pas toutes les trajectoires de ce type, voir l'exemple **LSSE** en fin de section suivante



(a) Image : 7 points sont notés sur l'image



(b) Trajet des rayons lumineux en chacun de ces 7 points.

FIG. 1.8 – Exemple de Ray-tracing.

source et ne subit pas de réfraction (comme spécifié dans l'algorithme). La position de la tache lumineuse à l'intérieur de la sphère est donc fausse. Cette erreur est généralement acceptée pour 2 raisons :

- Le modèle ne gèrerait pas ce type de réflexion car le rayon lancé n'a aucune chance de rencontrer la source ponctuelle dans la direction principale de réflexion avec les appels récursifs successifs.
- L'erreur n'est pas visible, car on n'a aucune idée intuitive de l'endroit où cette tache devrait être.

Trajet 3 : Mélange de deux contributions :

- la contribution majoritaire du damier derrière la sphère vu par transparence. La damier subit les distorsions attendues.
- la réflexion du damier sur la sphère. Elle est faiblement perceptible mais visible.

Trajet 4 : Le rayon entre dans la sphère mais reste seulement dans l'épaisseur de verre, et suffisament longtemps pour obtenir une réfraction importante. Le rayon touche le fond de la scène.

Trajet 5 : La couleur obtenue à la surface est le mélange de deux couleurs :

- celle obtenue localement sur la sphère opaque.
- celle résultant du reflet du damier sur la sphère (obtenu par récurrence).

Trajet 6 : Ce point est placé sur l'échiquier dans l'ombre de la sphère opaque (détecté par intersection entre le rayon d'ombre relancé en ce point vers la source lumineuse et la sphère opaque). Seule l'intensité ambiante apporte de la lumière à ce point. On remarque que le bord de l'ombre est très net (ce qui est théoriquement correct pour une source ponctuelle), mais légèrement choquant à l'oeil.

Trajet 7 : Cas identique au précédent excepté que l'objet qui fait de l'ombre est transparent. Le rayon est donc seulement atténué. Mais encore une fois, on ne prend pas en compte la réfraction lors de la traversée de la sphère. Par conséquent, la position et la forme de l'ombre associée à cette sphère transparente sont incorrectes.

Un problème supplémentaire non géré est le cas où une intensité lumineuse importante est issue d'une réflexion spéculaire, et que celle-ci provient d'une direction qui n'est pas la direction principale de réflexion. Par exemple, le reflet sur une surface d'une lampe réfléchie par un miroir ne fait pas partie des trajets gérés par la méthode originale du ray-tracing bien qu'il s'agisse d'un trajet **LSSE** (voir figure 1.9).

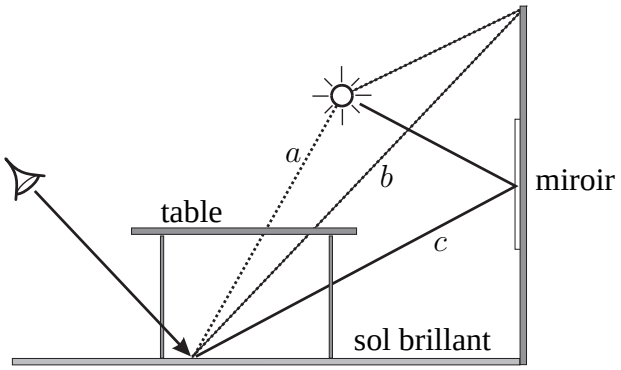


FIG. 1.9 – Exemple de rayon non géré par la méthode de Whitted. Les rayons directs (a) ainsi que le rayon dans la direction principale de réflexion (b) sont bloqués par la table. Pourtant, une tache lumineuse apparaît sous la table en suivant le trajet (c).

2.5 Conclusion

Cet algorithme est la première solution satisfaisante proposée pour la résolution de l'équation d'illumination globale. Il est capable de gérer des effets spéculaires complexes et donne d'excellents résultats pour la synthèse d'objets ayant de fortes composantes spéculaires ou transparentes malgré quelques aberrations assez peu visibles, et l'absence de gestion des effets spéculaires par les sources indirectes. Ce dernier problème sera réglé dans la partie consacrées aux aspects avancés du raytracing.

A l'inverse, lorsque la scène contient principalement des réflexions diffuses, le résultat est peu satisfaisant puisque globalement constant. Ceci est dû à la faiblesse de la modélisation des rayonnements diffus indirects : ils ne sont représentés que par un seul terme : l'intensité ambiante.

3 Calcul des intersections

Cette section est entièrement dédiée au calcul des intersections. Nous avons déjà donné à la section **??** une méthode générale pour calculer l'intersection entre une surface dont l'équation sous forme implicite est connue, et une demi-droite sous forme paramétrique. Nous rappellons tout d'abord cette méthode de calcul classique sur un plan et une sphère. Puis, pour chacun des objets canoniques, nous donnons la méthode de calcul optimisée, ainsi que le calcul de la normale au point d'intersection. Pour le cas d'objets contraints par des plans (hémisphère, ...), on prendra comme base de réflexion la méthode proposée pour les cylindres à la section 3.4.3.

Dans toute cette section, on note A l'origine du rayon lancé, et $\mathbf{u}$ sa direction. Sous

forme paramétrique, ce rayon s'écrit :

$$R(t) = A + t.\mathbf{u} \text{ avec } t > 0$$

Pour toute surface $\mathcal{S}$, on cherche la plus petite valeur de t_0 **positive** telle que $R(t_0)$ soit l'intersection entre $R(t)$ et $\mathcal{S}$ la plus proche de A. On notera que la plupart des méthodes optimisées exigent que le vecteur $\mathbf{u}$ soit **normalisé**. On notera que si $\mathbf{u}$ est normalisé, alors t représente la distance à laquelle on se trouve du point A dans la direction $\mathbf{u}$.

3.1 Choix du repère

Assez naturellement, la construction de la scène implique que l'objet $\mathcal{O}$ avec lequel on doit intersecter le rayon $R(t)$ n'est pas connu directement : on connaît l'objet tel que nous l'avons défini (par exemple, un objet géométrique élémentaire), et la matrice de transformation $\mathtt{M}$ qui permet de transformer et placer cet objet dans le repère global.

Si tel est le cas, il est alors beaucoup plus facile de **calculer l'intersection dans le repère local de l'objet** ; ceci peut être réalisé en passant le rayon dans le repère de l'objet.

On rappelle que la matrice $\mathtt{M}$ en coordonnées homogènes de la transformation affine qui passe du repère local au repère global s'écrit sous la forme :

$$\mathtt{M} = \begin{bmatrix} \mathtt{S} & \mathtt{T} \\ 0 & 1 \end{bmatrix}$$

où $\mathtt{S}$ est la matrice d'une transformation linéaire et $\mathbf{T}$ le vecteur de translation. Alors, le rayon dans le repère local s'écrit $R'(t) = A' + t.\mathbf{u}'$ avec :

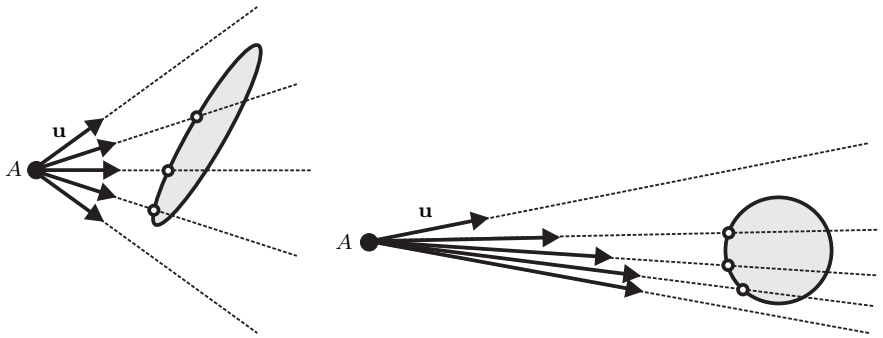
$$\begin{array}{rcl} \mathtt{A}' & = & \mathtt{S}^{-1}.(\mathtt{A} - \mathtt{T}) \\ \mathtt{u}' & = & \mathtt{S}^{-1}.\mathtt{u} \end{array} \tag{1.1}$$

Par conséquent, le calcul de l'intersection entre le rayon $R'(t)$ et l'objet $\mathcal{O}$ dans le repère local donne directement la valeur du paramètre t correspondant à l'intersection recherchée dans le repère global[3].

En conséquence, le calcul d'intersection peut se mener de la façon suivante :

– Passer le rayon $R(t)$ du repère global au repère local : $R'(t) = A' + t.\mathtt{u}'$ (voir équation 1.1).

[3]Soit P' la solution calculée dans le repère local, et t la valeur du paramètre associée. Soit P l'intersection dans le repère global. On a $\mathbf{A'P'} = t.\mathbf{u}'$. En transformant chaque membre par $\mathtt{S}$, par linéarité, on a $\mathbf{AP} = t.\mathbf{u}$. **Par conséquent, la valeur du paramètre est le même**. Attention, ceci n'est valide que parce que la transformation est affine.



(a) Intersections dans le repère global.

(b) Les mêmes intersections calculées dans le repère local. On a conservé le changement d'échelle des vecteurs induit par la transformation inverse.

FIG. 1.10 – Le calcul d'intersection dans le repère local devient le calcul d'intersection entre un rayon élémentaire et le repère global

.

- Calculer le paramètre t_0 correspondant à l'intersection entre le rayon $R'(t)$ et l'objet $\mathcal{O}$ dans son repère local.
- L'intersection dans le repère global est $R(t_0)$.

Calculer l'intersection dans les repères locaux exige de changer de repère à chaque rayon pour chaque objet. Sans précaution, cette méthode peut se révéler coûteuse. Mais, on notera que l'on a tout intérêt à disposer pour chaque objet de sa matrice de transformation inverse $\mathtt{S}^{-1}$. Elle peut être calculée une fois pour toutes (elle ne dépend pas du rayon). On peut également profiter de la cohérence des rayons. Plus précisément, comme indiqué dans la section **??**, le rayon envoyé correspondant aux coordonnées (i, j) s'écrit $R_{i,j}(t) = A + t.(\mathbf{U} + i.\delta.\mathbf{V} + j.\delta.\mathbf{W})$. Le rayon transformé se construit donc pour chaque objet comme celui ayant pour point d'origine le point A' (constant chaque objet) et pour direction une combinaison linéaire des vecteurs $\mathbf{U}'$, $\mathbf{V}'$ et $\mathbf{W}'$ (eux aussi constants pour chaque objets). Cet approche est facilitée par le fait que les méthodes de calcul dans les repères locaux n'ont généralement pas besoin que le vecteur de direction du rayon soit normalisé.

Pour ramener la normale au point d'intersection depuis le repère local vers le repère global, on consultera la section **??** du chapitre **??**.

3.2 Intersection rayon-plan

On rappelle brièvement l'expression du calcul d'intersection entre un rayon et un plan $\mathcal{P} = (C, \mathbf{N})$ (passant par le point C et de normale $\mathbf{N}$).

$$(R(t) - C).\mathbf{N} = 0 \Leftrightarrow (A - C).\mathbf{N} + t.\mathbf{u}.\mathbf{N} = 0$$

Cette dernière équation a deux dégénérescences possibles :

1. si $(A - C).\mathbf{N} = 0$, alors le point A est dans le plan $\mathcal{P}$.
2. si $\mathbf{u}.\mathbf{N} = 0$, alors le rayon est parallèle au plan.

Dans les cas non dégénérescents, on trouve le point d'intersection P :

$$t = \frac{(C - A).\mathbf{N}}{\mathbf{u}.\mathbf{N}} \Leftarrow P = A + t.\mathbf{u}$$

On en déduit l'algorithme d'intersection. On notera que si le vecteur $\mathbf{N}$ est normalisé, alors $(A - C).\mathbf{N}$ est la distance de A au plan $\mathcal{P}$ (notée d sur la figure 1.11.

On traite les cas dégénérescents (ou proches de la dégénérescence) par l'utilisation[4] d'un seuil ϵ. Si le cosinus θ de l'angle entre $\mathbf{N}$ et $\mathbf{u}$ est inférieur à ce seuil, alors le cas est dégénéré. Même si le calcul de ce cosinus n'est pas zéro, l'intersection que l'on pourrait trouver avec un angle aussi faible serait de toute façon déraisonnable. Si les vecteurs $\mathbf{N}$ et $\mathbf{u}$ ne sont pas normalisés (ou de norme proche de 1), $m = |\mathbf{u}|.|\mathbf{N}| \cos\theta$, le test à considérer n'est pas $m < \epsilon$ mais $m^2/(|\mathbf{u}|^2.|\mathbf{N}|^2) < \epsilon$, afin de garder la cohérence de ce test. Par ailleurs, dans le cas où $m < \epsilon$, il nous est parfaitement égal de savoir si le rayon n'intersecte pas le plan ou s'il y est inclus, car dans ce dernier cas ($d = 0$), un rayon ne peut pas se déplacer : il est donc rasant.

```
INTERSECTION RAYON(A, u) PLAN(C, N)
m = N.u
si (|m| < ε)
alors pas d'intersection
sinon   L = A − C
        d = N.L
        t = −d/m
        si t > 0
        alors intersection = A + t.u
        sinon pas d'intersection
```

Algorithme 3: Intersection rayon-plan

Normale au point d'intersection : $\mathbf{N}$ (évidemment, puisque la normale est constante !).

[4] En flottant, prendre $\epsilon = 10^{-5}$

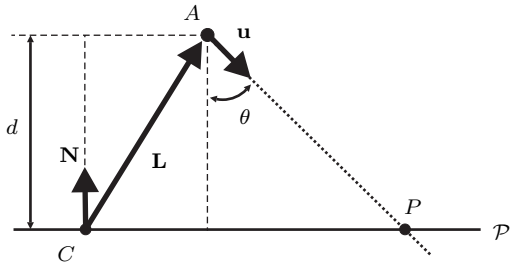


FIG. 1.11 – Géométrie du calcul d'intersection entre un rayon et un plan (cas où $|\mathbf{N}| = 1$).

3.3 Intersection rayon-sphère

3.3.1 Méthode classique

Soit une sphère centrée en O et de rayon r. Un point P appartient à la surface de cette sphère si et seulement si (équation de la sphère sous forme implicite) :

$$|OP| = r \Leftrightarrow \mathbf{OP}.\mathbf{OP} = r^2 \Leftrightarrow P.P = r^2$$

La demi-droite intersecte donc la sphère s'il existe une valeur de $t > 0$ telle que : $R(t).R(t) = r^2$ qui est une équation avec une seule inconnue en t. En remplaçant par l'expression du rayon :

$$\begin{aligned} & (A + t.\mathbf{u}).(A + t.\mathbf{u}) = r^2 \\ \Leftrightarrow \quad & |A|^2 + 2.(\mathbf{u}.A).t + t^2.|\mathbf{u}|^2 = r^2 \\ \Leftrightarrow \quad & |\mathbf{u}|^2.t^2 + 2.(\mathbf{u}.A).t + |A^2| - r^2 = 0 \end{aligned}$$

C'est une équation du second degré en t donc la solution réduite est :

$$\begin{cases} \Delta & = (\mathbf{u}.A)^2 - |\mathbf{u}|^2.(|A^2| - r^2) \leq 0 \\ t_\pm & = \dfrac{-\mathbf{u}.A \pm \sqrt{\Delta}}{|\mathbf{u}|^2} \end{cases}$$

où $t_{1/2}$ correspondent aux deux intersections possibles entre le rayon et la sphère (une seule dans le cas dégénéré où le rayon est tangent à la sphère).

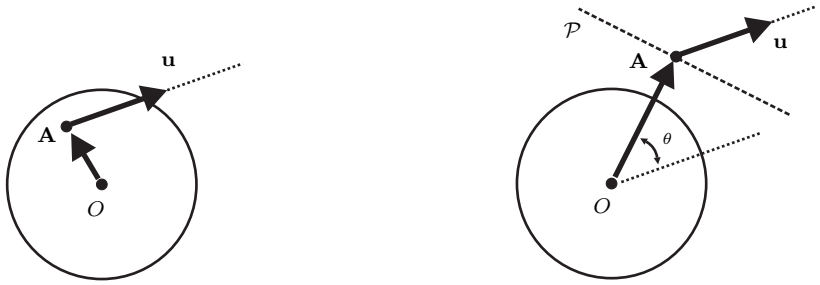
On détermine la solution à choisir entre t_1 et t_2 à partir de la géométrie locale du problème. Pour simplifier, on note $\mathbf{A}$ le vecteur $\mathbf{OA}$. On a alors plusieurs cas :

– si le point est dans la sphère ($a^2 < r^2$), alors les deux solutions sont telles que $t_- < 0 < t_+$. La solution recherchée est $t = t_+$ (le point dans la direction $\mathbf{u}$, voir figure 1.12(a)).

INTERSECTION RAYON$(A, \mathbf{u})$ SPHÈRE(O, r)
$a^2 = \mathbf{A}.\mathbf{A}$
$u^2 = \mathbf{u}.\mathbf{u}$
$d = \mathbf{u}.\mathbf{A}$
$\Delta = d^2 - u^2.(a^2 - r^2)$
si $\Delta < 0$ **alors** *pas d'intersection*
si $a^2 < r^2$
alors $t = (-d + \sqrt{\Delta})/u^2$
sinon **si** $d \geq 0$
alors *pas d'intersection*
sinon $t = (-d - \sqrt{\Delta})/u^2$
intersection $= A + t.\mathbf{u}$

Algorithme 4: Intersection rayon-sphère

– si le point est à l'extérieur de la sphère, alors le signe du produit scalaire $\mathbf{A}.\mathbf{u}$ nous donne la solution. S'il est positif, alors la demi-droite n'intersecte pas la sphère (les deux solutions sont "derrière" A). S'il est négatif, alors la demi-droite intersecte la sphère, et la solution est la plus proche des deux solutions trouvées : $t = t_-$ (voir figure 1.12(b))



(a) Le point est à l'intérieur. Il y a deux solutions, une dans la direction $\mathbf{u}$ et une dans la direction $-\mathbf{u}$. La solution recherchée est la solution positive.

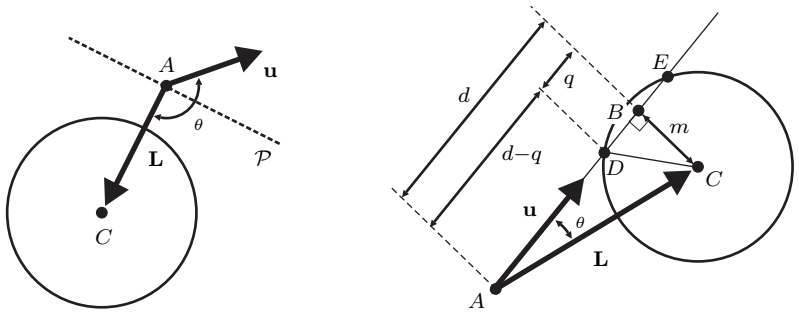
(b) Le point est à l'extérieur. Si $\mathbf{A}$ et $\mathbf{u}$ sont dans la même direction, il n'y a pas de solution. Dans le cas contraire, il y a deux solutions.

FIG. 1.12 – Géométrie locale pour le choix de la solution

3.3.2 Méthode optimisée

Les algorithmes optimisés proposent tous une approche géométrique du problème de l'intersection (contrairement à la résolution analytique présentée dans la section précédente). On considère toujours le problème de l'intersection mais on suppose

maintenant que le centre de la sphère est quelconque (on note le centre C et le rayon r). On suppose que le vecteur **u** est **normalisé**.



(a) sens du premier test : cas où $\mathbf{L}.\mathbf{u} < 0$ et A est à l'extérieur de la sphère ($l^2 > r^2$)

(b) Géomètrie du problème.

FIG. 1.13 – Méthode optimisée de calcul d'intersection entre un rayon et une sphère

INTERSECTION RAYON$(A, \mathbf{u})$ SPHÈRE(C, r)
$\mathbf{L} = C - A$
$d = \mathbf{L}.\mathbf{u}$
$l^2 = \mathbf{L}.\mathbf{L}$
si ($d < 0$ **et** $l^2 > r^2$) **alors** *pas d'intersection*
$m^2 = l^2 - d^2$
si $(m^2 > r^2)$ **alors** *pas d'intersection*
$q = \sqrt{r^2 - m^2}$
si $(l^2 > r^2)$ **alors** $t = d - q$
sinon $t = d + q$
intersection $= A + t.\mathbf{u}$

Algorithme 5: Intersection rayon-sphère

Nous présentons l'algorithme complet (voir Algorithme 5), et expliquons ses principaux aspects (voir aussi figure 1.13). Les deux premiers tests ont pour but d'éliminer tous les cas de non intersection :

premier test : soit $\mathcal{P}$ le plan passant par le point A et de vecteur normal $\mathbf{L}$ (voir figure 1.13(a)). Si la sphère n'intersecte pas le plan $\mathcal{P}$ (vérifié par le test $l^2 > r^2$) et si l'angle entre $\mathbf{L}$ et $\mathbf{u}$ est supérieur à $\pi/2$, alors le rayon ne peut pas intersecter la sphère. Donc, si $\mathbf{L}.\mathbf{u} = \cos\theta < 0$, le plan sépare alors l'espace en deux : d'un côté la demi-droite et de l'autre côté la sphère.

second test : considérons le triangle ABC (voir figure 1.13(b) pour les notations). On note les distances dans ce triangle $l = AC$, $d = AB$ et $m = BC$. $\mathbf{u}$ étant un

vecteur unitaire, on a :

$$\mathbf{u}.\mathbf{L} = |\mathbf{u}|.|\mathbf{L}|.\cos\theta = 1.l.\cos\theta = d$$

où $l = |\mathbf{L}|$. Ce triangle étant rectangle, on a $d^2 + m^2 = l^2$. Comme m représente la distance orthogonale du centre C de la sphère au rayon, le rayon n'intersecte pas la sphère si $m > r$.

valeur de t : si les deux tests précédents ont échoués, il y a nécessairement une intersection. Nous avons alors deux cas, suivant que le rayon part de l'intérieur de la sphère ($l < r$) ou pas :
- si l'origine du rayon part de l'extérieur, alors $t = AB - BD = d - q$ (première intersection avec la sphère en D)
- sinon, $t = AB + BE = d + q$ (l'intersection ne peut être que la seconde intersection en E).

Ces deux valeurs de t correspondent respectivement à chacune des intersections du rayon avec la sphère (*i.e.* $t_{1/2} = d \mp q$).

Comme dans la version précédente, une racine carrée est calculée (pour la valeur de m), mais elle ne l'est **que** lorsqu'il y a une intersection. Par ailleurs, l'obligation de normaliser le vecteur $\mathbf{u}$ conduit à un calcul de racine carrée supplémentaire, mais cette normalisation n'a lieu qu'une seule fois par rayon, avant le calcul de toutes les intersections.

3.3.3 Normale N au point d'intersection

Trivialement, si P est le point d'intersection, la normale au point P se définit à partir du centre de la sphère comme : $\mathbf{N} = \mathbf{CP}/r$.

3.4 Intersection rayon-cylindre

On considère dans cette partie le calcul d'intersection avec un cylindre infini, puis contraint par des plans.

3.4.1 Méthode classique d'intersection avec un cylindre infini

Dans cette section, le cylindre est de centre O, d'axe Oz, et de rayon r. Son équation cartésienne est dans ce repère local $x^2 + y^2 = r^2$. En notant $A = (a_x, a_y, a_z)$ et $\mathbf{u} = (\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z)$, il faut donc résoudre en t l'équation suivante :

$$(a_x + t.u_x)^2 + (a_y + t.u_y)^2 = r^2$$

On obtient après développement :

$$({u_x}^2 + {u_y}^2).t^2 + 2.(a_x.u_x + a_y.u_y).t + ({a_x}^2 + {a_y}^2) = r^2$$

On note respectivement $A_z = (a_x, a_y, 0)$ (resp. $\mathbf{u}_z$) la projection de A (resp. $\mathbf{u}$) dans le plan Oxy. Dans ce plan, la projection du cylindre est un cercle ; il est alors clair que le rayon intersecte le cylindre si le rayon projeté intersecte ce cercle. L'équation du cylindre se réécrit donc sous la forme :

$$t^2.{\mathbf{u}_z}^2 + 2.t.A_z.\mathbf{u}_z + {A_z}^2 - r^2 = 0$$

Le déterminant réduit s'écrit par conséquent :

$$\begin{aligned}\Delta &= (A_z.\mathbf{u}_z)^2 - {\mathbf{u}_z}^2.({A_z}^2 - r^2) \\ &= r^2.{\mathbf{u}_z}^2 - (A_z.\mathbf{u}_z^\perp)^2 \\ &= {\mathbf{u}_z}^2.\left(r^2 - (A_z.\frac{\mathbf{u}_z^\perp}{|\mathbf{u}_z^\perp|})^2\right)\end{aligned}$$

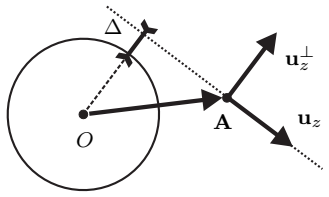


FIG. 1.14 – Sens du déterminant.

où $\mathbf{u}_z^\perp = (u_y, -u_x, 0)$ est le vecteur orthogonal à $\mathbf{u}_z$ dans le plan Oxy (donc $|\mathbf{u}_z^\perp| = |\mathbf{u}_z|$. Il est alors possible de donner directement une interprétation géométrique au déterminant (voir figure ci-contre) : il représente une mesure de la distance à laquelle le rayon passe au plus près du cylindre (à la constante multiplicatrice $|\mathbf{u}_z|$ près). Si Δ est négatif, alors le rayon passe trop loin. Sinon, il intersecte. Les solutions sont alors classiquement :

$$t_\pm = \frac{-A_z.\mathbf{u}_z \pm \sqrt{\Delta}}{{\mathbf{u}_z}^2}$$

Afin d'implémenter ce cas efficacement, on utilise le produit scalaire $A_z.\mathbf{u_z}$ pour déterminer la position des solutions (même idée que pour la sphère, voir figure 1.12). Voir l'algorithme 6. On notera qu'aucun calcul de racine carrée n'est nécessaire avant d'être sur de l'existence d'une solution. On ne détermine pas laquelle des deux intersections doit être retenue car nous avons besoin des deux pour pouvoir traiter le cas du cylindre tronqué (les cylindres infinis sont plutôt rares...). On consultera à ce sujet la section 3.4.3.

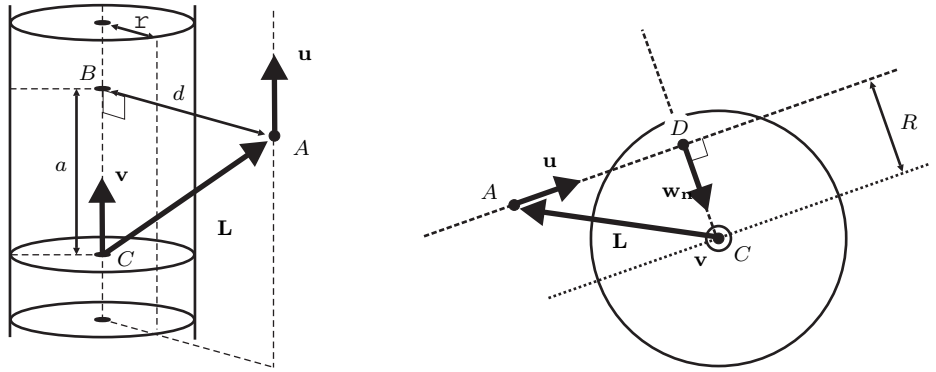
3.4.2 Méthode optimisée d'intersection avec un cylindre infini

On considère dans cette section un cylindre infini de centre C, d'axe $\mathbf{v}$, et de rayon r. On suppose également que les vecteurs $\mathbf{u}$ et $\mathbf{v}$ sont **normalisés**.

Nous ne détaillons de cet algorithme que la partie permettant de détecter si une intersection a oui ou non lieu. Pour le complément (calcul effectif de l'intersection), nous renvoyons aux *Graphics Gems vol. IV* d'où cet algorithme est extrait.

INTERSECTION RAYON$(A, \mathbf{u})$ CYLINDRE(O, r, Oz)
$u_z{}^2 = \mathbf{u}_z.\mathbf{u}_z$.
$d = \mathbf{u}_{\bar{z}}^{\perp}.A_z$
$\Delta = r^2.u_z{}^2 - d^2$
si $\Delta < 0$ **alors** *pas d'intersection*
$l = \mathbf{u}_z.A_z$
$a^2 = A.A$
si $l > 0$ **et** $a^2 > r^2$ **alors** *pas d'intersection*
$t_{\pm} = (-l \pm \sqrt{\Delta})/u_z{}^2$
intersections = $A + t_{\pm}.\mathbf{u}$

Algorithme 6: Intersection rayon-cylindre infini



(a) cas où $w = 0$: $\mathbf{u}$ et $\mathbf{v}$ sont colinéaires. (b) Test d'intersection entre le rayon et la sphère.

FIG. 1.15 – Méthode optimisée de calcul d'intersection entre un rayon et un cylindre

Premier test on calcule le produit vectoriel entre la direction du rayon $\mathbf{u}$ et la direction de l'axe du cylindre $\mathbf{v}$. S'il est nul, alors les deux vecteurs sont colinéaires. Dans ce cas, on calcule la distance orthogonale d de l'origine du rayon à l'axe du cylindre (voir figure 1.15(a)). Elle nous permet alors de déterminer la position du rayon par rapport au cylindre.

Second test soit $\mathcal{P}$ le plan passant contenant le rayon et parallèle à l'axe du cylindre (= plan passant par A et de vecteur normal $\mathbf{w} = \mathbf{u} \wedge \mathbf{v}$). On note $\mathbf{w_n}$ le vecteur $\mathbf{w}$ normalisé. Soit D l'intersection du plan $\mathcal{P}$ est de la droite $(C, \mathbf{w})$ (voir figure 1.15(b)). Par construction, $\mathbf{w}$ étant normal au plan, D est donc la projection orthogonale de C sur $\mathcal{P}$. Comme le plan $\mathcal{P}$ contient le rayon, $R = CD$ est la distance à laquelle le rayon se rapproche le plus de l'axe central du cylindre. Cette distance R est obtenue en calculant le produit scalaire : $\mathbf{CA}.\mathbf{w_n} = \pm R$.

Choix de l'intersection L'intersection à choisir dépend de la position de l'origine du rayon A par rapport au cylindre :

INTERSECTION RAYON$(A, \mathbf{u})$ CYLINDRE$(C, r, \mathbf{v})$
$\mathbf{L} = A - C$
$\mathbf{w} = \mathbf{u} \wedge \mathbf{v}$
$w^2 = \mathbf{w}.\mathbf{w}$
si $(w^2 = 0)$
alors *(* cas rayon parallèle à l'axe du cylindre *)*
$a = \mathbf{L}.\mathbf{v}$
$\mathbf{D} = \mathbf{L} - a.\mathbf{v}$
$d^2 = \mathbf{D}.\mathbf{D}$
si $d^2 > r^2$ **alors** *pas d'intersection*
sinon intersection(s)= $\pm\infty$ (dans le cylindre)
sinon $\mathbf{w_n} = \mathbf{w}/w$
$R = |\mathbf{L}.\mathbf{w_n}|$
si $R > r$
alors *pas d'intersection*
sinon $\mathbf{E} = \mathbf{L} \wedge \mathbf{v}$
$t = -\mathbf{E}.\mathbf{w_n}/w$
$\mathbf{F} = \mathbf{w_n} \wedge \mathbf{v}$
$\mathbf{F_n} = \mathbf{F}/|\mathbf{F}|$
$s = \sqrt{r^2 - R^2}/|\mathbf{u}.\mathbf{F_n}|$
intersection(s) = $A + (t \mp s).\mathbf{u}$

Algorithme 7: Intersection optimisée rayon-cylindre

- si A est dans le cylindre (*i.e.* $\mathbf{L}.\mathbf{v} < r$), alors l'intersection à choisir est l'intersection sortante : $t_2 = t + s$.
- sinon (A est à l'extérieur du cylindre), et l'intersection à choisir est l'intersection entrante $t_1 = t - s$.

Notons que cette version optimisée n'est plus rapide que si l'axe du cylindre n'est pas parallèle à l'un des axes du repère global.

3.4.3 Troncature du cylindre

L'algorithme, tel qu'il est écrit, renvoie les deux intersections (entrantes et sortantes) du cylindre, alors qu'habituellement, seule l'intersection la plus proche de l'origine du rayon nous intéresse. Ceci permet d'adapter facilement cet algorithme à l'intersection entre un rayon et un cylindre fermé par des plans.

Chaque plan est traité de la façon suivante. Considérons le plan de normale $\mathbf{N}$ et passant par le point B. On prend soin d'orienter la normale N vers l'extérieur (voir figure 1.17). On se place dans le cas où le rayon intersecte effectivement le cylindre infini.On a alors 3 cas :

$\mathbf{u}.\mathbf{N} = 0$ (voir figure 1.17(a)), le rayon est parallèle au plan. Tout dépend alors de la position du rayon par rapport au plan :

– s'il est au dessus du plan, il n'y a pas d'intersection.
– s'il est en dessous du plan, l'intersection est t_1.

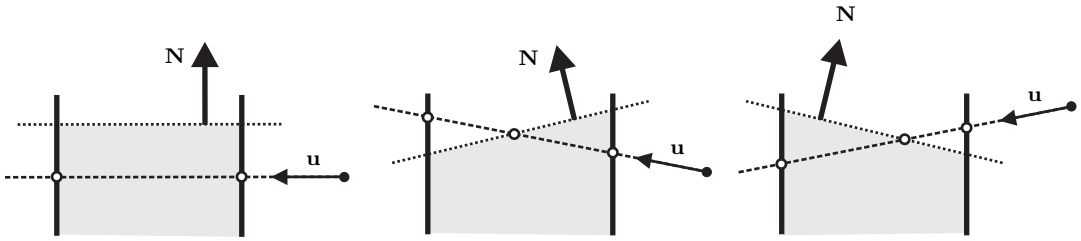
Il suffit par conséquent de tester le signe du produit scalaire : $(A - B).\mathbf{N}$.

$\mathbf{u}.\mathbf{N} > 0$ (voir figure 1.17(b)), l'intersection entre le rayon et le plan est une intersection sortante. Par conséquent, le rayon intersecte d'abord le cylindre avant d'intersecter le plan.

$\mathbf{u}.\mathbf{N} < 0$ (voir figure 1.17(c)), l'intersection entre le rayon et le plan est une intersection entrante. Par conséquent, le rayon intersecte d'abord le plan.

Notons que dans la méthode proposée, rien n'oblige la normale au plan à être colinéaire à l'axe du cylindre (on exclut quand même le cas où la normale au plan est orthogonale à l'axe du cylindre *i.e.* $\mathbf{v}.\mathbf{N} = 0$). On donne ci-dessous le code correspondant.

```
(t_in, t_out) = les deux intersections avec le cylindre
D_c = u.N
D_w = (A - B).N
si Dc = 0
alors (* rayon parallele au plan *)
        si D_w > 0 alors pas d'intersection
sinon   t = -D_w / D_c
        si D_c ≥ 0
        alors   si (t > t_in) et (t < t_out) alors t_out = t
                si (t < t_in) alors pas d'intersection
        sinon   si (t > t_in) et (t < t_out) alors t_in = t
                si (t > t_out) alors pas d'intersection
(* répéter le code ci-dessus autant de fois qu'il y a de plan *)
si t_in > t_out alors pas d'intersection
intersection = t_in
```



(a) cas 1 : $\mathbf{u}.\mathbf{N} = 0$ (b) cas 2 : $\mathbf{u}.\mathbf{N} > 0$ (c) cas 3 : $\mathbf{u}.\mathbf{N} < 0$

FIG. 1.16 – Différents cas pour les cylindres limités par des plans.

Notons que l'on peut rapidement éliminer les rayons dont l'origine est au dessus du plan ($A.\mathbf{N} > 0$) et dont la direction est la même que celle de la normale ($\mathbf{v}.\mathbf{N} > 0$).

Suivant les cas, il peut être avantageux de faire ce test plus tôt.

Lorsque le calcul d'intersection sur l'objet à tronquer est fait dans le repère local, on aura tout intérêt à effectuer également la troncature dans ce même repère, surtout si les plans sont alignés avec les axes.

3.4.4 Normale N au point d'intersection

Si P est le point d'intersection et Q la projection orthogonale de P sur l'axe du cylindre, alors le vecteur $\mathbf{QP}$ est un vecteur normal à la surface. On construit donc la normale comme suit :

$CQ = \mathbf{CP}.\mathbf{v}$
$\mathbf{QP} = \mathbf{CP} - CQ.\mathbf{v}$
$\mathbf{N} = \mathbf{QP}/r$

Attention, penser dans le cas du cylindre contraint à adapter le calcul de la normale en fonction de la position du point d'intersection (sur le plan ou sur le cylindre).

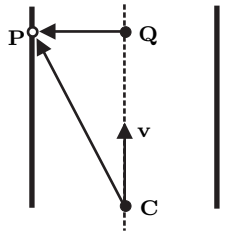


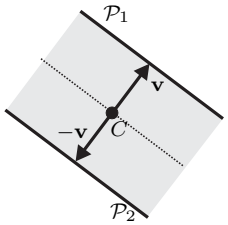
FIG. 1.17 – Calcul de la normale.

3.5 Intersection rayon-slabs

3.5.1 Définition

Un slab est un couple de plans parallèles $(\mathcal{P}_1, \mathcal{P}_2)$ définis par un centre C, un vecteur normalisé $\mathbf{v}$ et une distance δ de la façon suivante :

$$\begin{aligned} \mathcal{P}_1 &= (C + \delta.\mathbf{v}, \mathbf{v}) \\ \mathcal{P}_2 &= (C - \delta.\mathbf{v}, -\mathbf{v}) \end{aligned}$$



On les utilise généralement pour définir des objets à faces opposées parallèles (parallélépipède, décaèdre, ...) ou des boîtes englobantes convexes.

3.5.2 Intersection rayon-union de slabs

On considère maintenant l'objet défini par l'intersection de n slabs $\{(C_i, \mathbf{v}_i)\}_{i=1...n}$, et on recherche l'intersection entre un rayon et cet objet. Pour un rayon, on note $(t_i^{\min}, t_i^{\max})$ les deux intersections (cas non dégénéré, une par plan) entre le rayon et le slab $(C_i, \mathbf{v}_i)$. On définit alors :

$$\begin{aligned} t^{\min} &= \max_i t_i^{\min} \\ t^{\max} &= \min_i t_i^{\max} \end{aligned}$$

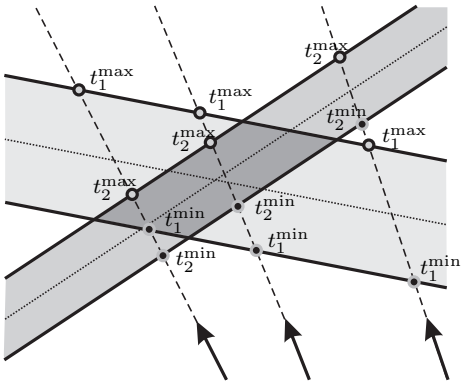


FIG. 1.18 – Trois exemples d'intersection entre un rayon et un slabs.

respectivement la plus éloignée des intersections avec le plan le plus proche de chaque slab, et la plus proche des intersections avec le plan le plus éloigné de chaque slab. Alors, on peut montrer (voir aussi figure 1.18) qu'un critère nécessaire et suffisant d'intersection est $t^{\min} \leq t^{\max}$.

En utilisant ce critère et le calcul d'intersection entre un rayon et un plan, on en déduit immédiatement l'algorithme d'intersection entre le rayon et le slabs (voir algorithme 8). On notera que si tous les centres des slabs sont confondus, alors le calcul de p peut être extrait de la boucle.

3.6 Intersection rayon-facette

On considère maintenant une facette à trois sommets v_1, v_2 et v_3. On sait que tout point du triangle peut s'écrire sous la forme :

$$T(\lambda_1, \lambda_2) = v_1 + \lambda_1.\mathbf{v_1v_2} + \lambda_2.\mathbf{v_1v_3}$$

où $\lambda_1 \geq 0$, $\lambda_2 \geq 0$ et $\lambda_1 + \lambda_2 \leq 1$.

INTERSECTION RAYON $(A, \mathbf{u})$ SLABS $\{(C_i, \delta_i, \mathbf{v}_i)\}_{i=1...n}$
$t^{\min} = -\infty$
$t^{\max} = +\infty$
pour i **allant de** 1 **à** n
    $\mathbf{L} = C_i - A$
    $d = \mathbf{v}_i.\mathbf{L}$
    $m = \mathbf{v}_i.\mathbf{u}$
    si $(|m| < \epsilon)$
    alors **si** $|d| > \delta_i$
        alors *pas d'intersection*
    sinon $t_1 = (d - \delta_i)/m$
        $t_2 = (d + \delta_i)/m$
        si $(t_1 > t_2)$ **alors** SWAP(t_1, t_2)
        si $(t_1 > t^{\min})$ **alors** $t^{\min} = t_1$
        si $(t_2 < t^{\max})$ **alors** $t^{\max} = t_2$
        si $(t^{\min} > t^{\max})$ **ou** $(t^{\max} < 0)$ **alors** *pas d'intersection*
si $(t^{\min} > 0)$ **alors** $t = t^{\min}$
    sinon $t = t^{\max}$
intersection = $A + t.\mathbf{u}$

Algorithme 8: Intersection rayon-slabs

3.6.1 Calcul de l'intersection

Notre problème peut donc se reformuler : trouver le triplet[5] $(t, \lambda_1, \lambda_2)$ tel que $R(t) = T(\lambda_1, \lambda_2)$, ce qui s'écrit :

$$v_1 + \lambda_1.\mathbf{v_1v_2} + \lambda_2.\mathbf{v_1v_3} = A + t.\mathbf{u}$$

$$\Leftrightarrow \quad \begin{bmatrix} -\mathbf{u} & \mathbf{v_1v_2} & \mathbf{v_1v_3} \end{bmatrix} . \begin{bmatrix} t \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = A - v_1$$

Il s'agit d'un système de 3 équations à 3 inconnues. En notant $\mathbf{L} = A - v_1$, et en utilisant la règle de Cramer pour résoudre ce système, on obtient :

$$\begin{bmatrix} t \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \frac{1}{\begin{vmatrix} -\mathbf{u} & \mathbf{v_1v_2} & \mathbf{v_1v_3} \end{vmatrix}} . \begin{bmatrix} |\mathbf{L} & \mathbf{v_1v_2} & \mathbf{v_1v_3}| \\ |-\mathbf{u} & \mathbf{L} & \mathbf{v_1v_3}| \\ |-\mathbf{u} & \mathbf{v_1v_2} & \mathbf{L}| \end{bmatrix}$$

Or, le calcul du déterminant $\begin{bmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \end{bmatrix}$ de trois vecteurs $\mathbf{a}$, $\mathbf{b}$ et $\mathbf{c}$ peut se réécrire comme : $\begin{vmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \end{vmatrix} = -(\mathbf{a} \wedge \mathbf{c}).\mathbf{b} = -(\mathbf{c} \wedge \mathbf{b}).\mathbf{a}$. En conséquence, la solution peut se

[5] On rappelle que le cas où le rayon est inclus dans le plan de la facette ne nous intéresse pas (dans le cas d'un objet facettisé, parce qu'il existe une autre facette qui intersecte elle aussi ce rayon). On cherche donc **une solution au plus**.

réécrire comme :

$$\begin{bmatrix} t \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \frac{1}{(\mathbf{u} \wedge \mathbf{v_1 v_3}).\mathbf{v_1 v_2}} . \begin{bmatrix} (\mathbf{L} \wedge \mathbf{v_1 v_2}).\mathbf{v_1 v_3} \\ (\mathbf{u} \wedge \mathbf{v_1 v_3}).\mathbf{L} \\ (\mathbf{L} \wedge \mathbf{v_1 v_2}).\mathbf{u} \end{bmatrix}$$

On en déduit l'algorithme 9. Le premier test vérifie si le rayon et le plan sont parallèles (en comparant la valeur du déterminant à 0). Encore une fois, on utilise un ϵ pour stabiliser l'algorithme et éviter des solutions déraisonnables. Puis, on calcule λ_1 et λ_2 en prenant soin de vérifier leurs domaines de validité dès leur obtention.

INTERSECTION RAYON $(A, \mathbf{u})$ FACETTE (v_1, v_2, v_3)
$\mathbf{v_1 v_2} = v_2 - v_1$
$\mathbf{v_1 v_3} = v_3 - v_1$
$\mathbf{p} = \mathbf{u} \wedge \mathbf{v_1 v_3}$
$m = \mathbf{v_1 v_2}.\mathbf{p}$
si $(m > -\epsilon)$ **et** $(m < \epsilon)$ **alors** *pas d'intersection*
$\mathbf{L} = A - v_1$
$\lambda_1 = (\mathbf{L}.\mathbf{p})/m$
si $(\lambda_1 < 0)$ **ou** $(\lambda_1 > 1)$ **alors** *pas d'intersection*
$\mathbf{q} = \mathbf{L} \wedge \mathbf{v_1 v_2}$
$\lambda_2 = (\mathbf{u} \wedge \mathbf{q})/m$
si $(\lambda_2 < 0)$ **ou** $(\lambda_2 > 1)$ **ou** $(\lambda_1 + \lambda_2 > 1)$ **alors** *pas d'intersection*
$t = (\mathbf{v_1 v_3}.\mathbf{q})/m$
intersection = $A + t.\mathbf{u}$

Algorithme 9: Intersection rayon-facette

3.6.2 Calcul de la normale à la facette

Dans le cas où l'on souhaite faire de l'ombrage de Phong sur la facette, les coordonnées (λ_1, λ_2) de l'intersection sur la facette permettent d'obtenir directement l'expression de la normale interpolée. En notant $\mathbf{N_1}$, $\mathbf{N_2}$ et $\mathbf{N_3}$ les normales respectives aux sommets v_1, v_2 et v_3, alors la normale au point d'intersection est :

$$\mathbf{N}(\lambda_1, \lambda_2) = (1 - \lambda_1 - \lambda_2).\mathbf{N_1} + \lambda_1.\mathbf{N_2} + \lambda_2.\mathbf{N_3}$$

3.7 Exercices

Nous laissons les cas suivants à la réflexion des lecteurs assidus.

1. **Cas du cône**
   Un cône possède beaucoup de similarité avec le cas du cylindre. On pourra

utiliser les tests d'intersections avec le cylindre englobant le cône et ses plans contraints pour filtrer rapidement les rayons. Proposer alors un algorithme d'intersection calculant au plus une racine carrée.

2. **Cas du cube unité**
   Utiliser trois slabs utilisant **tous** le même point centre, et optimiser en tenant compte du fait que les normales sont alignées avec les axes.
3. **Cas de l'objet convexe**
   Mathématiquement, un objet convexe se définit comme l'espace délimité par une union (éventuellement infinie) de plans. Considérons alors un ensemble de n plans $\{\mathcal{P}_i\}_{i=1...n}$ où chaque plan $\mathcal{P}_i$ est défini par son point d'origine P_i et sa normale $\mathbf{N}_i$. Proposer un algorithme permettant de calculer l'intersection entre un rayon et un tel objet.

4 Accélération du ray-tracing

L'algorithme original du raytracing passe 95% du temps à calculer des intersections entre les rayons et les objets. Nous présentons dans cette partie des optimisations travaillant sur 3 aspects du ray-tracing :

- la limitation du nombre de lancements récursifs.
- un test rapide (mais approximatif) d'intersection.
- la restriction du nombre d'objets avec lesquels il faut calculer une intersection.

Les deux dernières méthodes sont fondées sur le principe qui consiste, **hors contexte** (*i.e.* sans savoir où se trouve l'observateur), à tenter :

- soit de précalculer des quantités fréquemment utilisées.
- soit de construire des structures de données en vue d'accélérer les calculs une fois que le contexte est connu (cas pour les deux méthodes proposées).

4.1 Contrôle de la profondeur

Un algorithme de ray-tracing naïf n'arrête de lancer récursivement des rayons que dans 3 cas :

- Le rayon lancé ne touche rien.
- L'objet touché par le rayon est mat et opaque.
- La profondeur maximale r est atteinte.

Or en général, dans une scène, seule une petite partie des objets ont des caractéristiques de transparence ou de spécularité et des lieux géométriques qui nécessitent de relancer les rayons. L'adaptation de la profondeur de récursion aux types de surface rencontrées

sur le trajet d'un rayon lumineux permettrait d'élaguer l'arbre des récursions et de ne le limiter qu'aux rayons lumineux qui produisent des résultats visibles.

Cette idée peut être mise en place assez simplement en utilisant les atténuations le long du chemin du rayon. Lorsqu'un rayon spéculaire/réfracté est relancé, l'intensité du rayon renvoyé est multiplié par un terme de transmitivité plus petit que 1 (atténuation).

Voici un exemple. Dans une scène ne comprenant que des objets brillants mais non transparents, supposons que nous avons la suite récursive d'évaluation suivante :

$$\begin{array}{ccccccccc} O & \longrightarrow & P_1 & \longrightarrow & P_2 & \longrightarrow & P_3 & \longrightarrow & P_4 \\ & \xleftarrow{k_0\times} & I_1 & \xleftarrow{+k_1\times} & I_2 & \xleftarrow{+k_2\times} & I_3 & \xleftarrow{+k_3\times} & I_4 \end{array}$$

Le premier rayon est envoyé depuis l'observateur O et renvoyé de P_i à P_{i+1} par une suite de réflexions spéculaires. La valeur I_i correspond à l'évaluation **locale seulement** de l'intensité au point P_i (le terme $I_a + \sum_{j=1}^{p}(I_j^d + I_j^s + I_j^t)$), et k_i à l'atténuation qui sera appliquée à l'intensité du rayon relancé. L'intensité au point O s'évalue par :

$$\begin{aligned} I &= k_0 \times (I_1 + k_1 \times (I_2 + k_2 \times (I_3 + k_3 \times (I_4)))) \\ &= k_0 \times I_1 + k_0.k_1 \times I_2 + k_0.k_1.k_2 \times I_3 + k_0.k_1.k_2.k_3 \times I_4 \end{aligned}$$

Par conséquent, pour un rayon qui rebondit p fois, l'intensité locale I_p trouvée au $p^{\text{ième}}$ rebond subit l'ensemble des atténuations du chemin. Le produit π_p des atténuations :

$$\pi_p = \prod_{i=0}^{p-1} k_i$$

indique l'atténuation réelle et minimale que subira le rayon dans ses rebonds suivants. Ainsi, si ce produit est faible, la contribution des rebonds suivants a toutes les chances d'être négligeable. On définit donc un seuil ε en dessous duquel l'atténuation est considérée comme trop forte pour continuer la récursion. Le code est modifié de la façon suivante avant de relancer un rayon :

(calcul de l'atténuation d'un rayon relancé *)*
$\pi_p = k_{p-1} \times \pi_{p-1}$
(test si récursion nécessaire *)*
si $\pi_p < \varepsilon$
alors $I_o = 0$
sinon *(* évaluer I_o en relançant un rayon *)*

Le facteur d'atténuation π_p est transmis lors de l'appel récursif pour les évaluations suivantes. Il faut, de plus, gérer deux facteurs d'atténuation différents (autant que de rayons relancés) : un pour la réflexion spéculaire et un pour la réfraction.

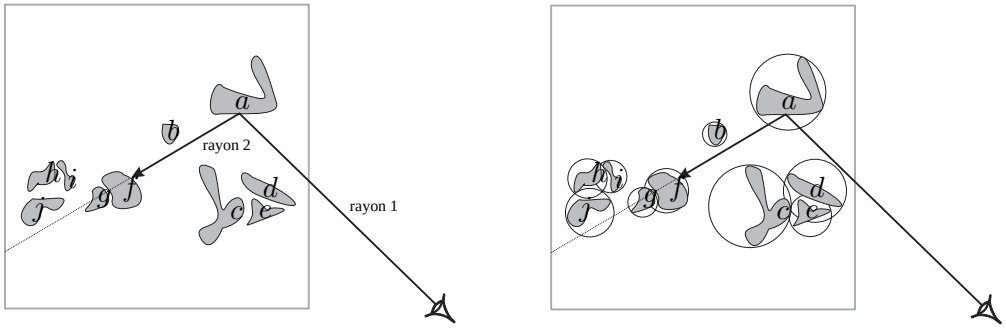
Des études ont montré que dans une scène comportant de forts effets spéculaires et avec $r = 15$, la profondeur moyenne de récursion est de l'ordre de 1.71 lorsque cette méthode est utilisée. Cette méthode permet donc d'élaguer considérablement l'arbre des appels récursifs. Elle a pour seul inconvénient d'élaguer également les quelques rares cas où une intensité très forte située plus loin dans la récursion contrebalance le coefficient d'atténuation.

4.2 Boîtes englobantes

Dans cette partie, on cherche à réduire le coût d'une intersection entre un rayon et un objet. Le principe des boîtes englobantes est le suivant :

- définir une boîte englobant l'objet pour laquelle le coût du test d'intersection avec un rayon est sensiblement inférieur à celui entre l'objet et le rayon.
- lorsqu'un rayon est lancé, on applique la méthode suivante :

  calculer l'intersection entre la boîte englobante et le rayon.
  si il y a une intersection
  alors calculer l'intersection entre l'objet et le rayon.



(a) Exemple de scène (b) Avec boite englobante

intersection avec	boîtes	objets
sans boîtes (rayon 1 et 2)	×	10
avec boîtes (rayon 1)	10	1
avec boîtes (rayon 2)	10	4

FIG. 1.19 – Utilisation des boites englobante.

Cette approche n'est pas nécessairement avantageuse, car il est possible que le temps gagné à éliminer rapidement un objet grâce à sa boîte englobante soit perdu

lors du calcul de la seconde intersection qu'il est nécessaire de calculer avec tous les objets qui n'auront pas été écarté. Plus formellement, si ε est le pourcentage de rayons nécessitant un double calcul d'intersections, que C_b et C_o sont les coûts respectifs du calcul d'intersection pour la boîte et l'objet, cette condition s'écrit :

$$C_o > \underbrace{(1-\varepsilon).C_b}_{\text{intersections simples}} + \underbrace{\varepsilon.(C_b+C_o)}_{\text{intersections doubles}} = C_b + \varepsilon.C_o$$

Donc, si la première intersection écarte la moitié des rayons, le calcul de l'intersection avec la boîte doit être au moins deux fois plus rapide que le calcul de l'intersection avec l'objet. Si cette condition est évidemment vérifiée lorsque le calcul d'intersection est fait systématiquement avec tous les objets, elle ne l'est pas nécessairement si cette méthode est utilisée en conjonction avec une méthode de partition d'espace.

L'efficacité de cette méthode dépend également de la fidélité avec laquelle la boîte englobante colle à l'objet. Plus l'espace entre la boîte et l'objet est important, plus le deuxième test a des chances de conclure à une absence d'intersection. Ce facteur agit directement sur le nombre de rayons pour lesquels deux intersections seront calculées à la place d'une seule. Il est donc important de choisir une boîte englobante adaptée à l'objet. Les boîtes englobantes les plus couramment utilisées sont les sphères, les parallélépipèdes rectangles et les cylindres.

Enfin, des hiérarchies de boîtes englobantes peuvent être utilisées :

- soit de façon ascendante : on groupe plusieurs boîtes englobantes d'objets dans une boîte englobante plus grande, et on réitère le procédé. La règle de regroupement est la proximité géométrique, toujours de façon à limiter la taille de la boîte englobante. Par exemple, pour un volume défini à partir de facettes.
- soit de façon descendante, l'objet est découpé en sous-parties qui sont à leur tour munies d'une boîte englobante. Par exemple, pour un objet non convexe mal représenté par sa boîte englobante.

L'arbre est alors traité de la façon suivante : s'il y a une intersection entre le rayon et la boîte, on relance récursivement un calcul d'intersection sur tous les fils de la boîte. S'il n'y a pas d'intersection, on ne calcule aucune autre intersection dans l'arbre. Le traitement renvoie la liste des intersections trouvées dans l'arbre.

4.3 Cohérence spatiale

On profite de la cohérence spatiale des objets. Le principe est le suivant : l'espace occupé par la scène est découpé en régions disjointes. Pour chacune de ces régions, on connaît l'ensemble des objets qui y est contenu. Lorsqu'un rayon est lancé, son trajet traverse une suite de régions. On ne teste alors que le sous-ensemble d'objets

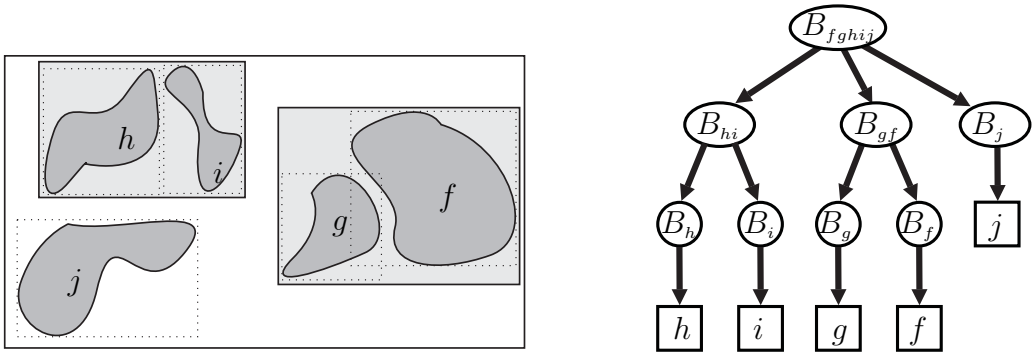


FIG. 1.20 – Hiérarchie de boîtes englobantes. On note B_g (resp. B_{gf}) la boîte englobante de g (resp. g et f).

contenus dans ces régions **et** dans l'ordre où ces régions sont traversées. Quand une région est traversée, si le rayon y intersecte au moins un objet, alors l'objet recherché est le premier qui est touché par le rayon lors de la traversée de la région. Il est alors inutile de poursuivre la recherche.

Cette méthode présente donc les avantages suivants :

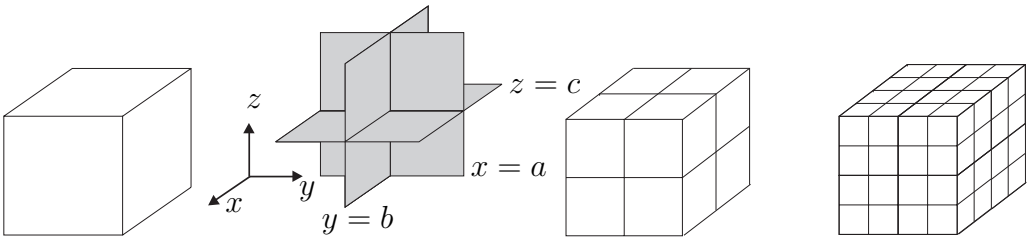
- Le découpage de l'espace implique que les intersections ne soient testées qu'avec les objets situés à proximité du trajet du rayon.
- Le fait de ne tester les intersections que dans les régions traversées et dans l'ordre où elles sont traversées permet d'arrêter les recherches dès la première région contenant une intersection.

et permet d'effectuer un ray-tracing en temps constant, presque indépendamment de la complexité de la scène, avec un temps de rendu par rayon relativement constant.

Nous présentons deux méthodes de partitionnement d'espace. Ces partitionnements sont, pour une scène, réalisés une fois pour toutes, et sont indépendants de la phase du rendu.

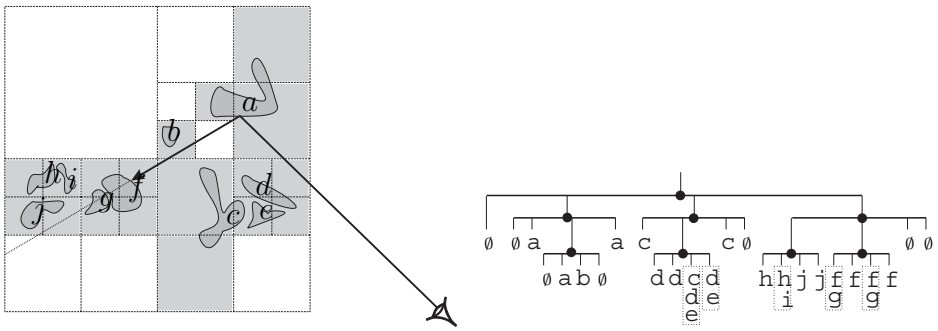
4.3.1 Les octrees

Un octree est un arbre qui représente une subdivision d'espace et dans lequel chaque noeud a 8 fils. La descente d'un niveau dans l'arbre correspond au raffinement d'une portion d'espace en 8 sous-espaces. La subdivision est faite à l'aide de triplets de plans généralement orthogonaux, comme présenté à la figure 1.21. La figure 1.22 présente un exemple (en 2D) de cette méthode.



(a) Espace original. (b) Triplet de plans utilisés pour la subdivision. (c) Après une subdivision. (d) Après deux subdivisions uniformes.

FIG. 1.21 – Subdivision par octree.



(a) Exemple en 2D. (b) Arbre associé.

FIG. 1.22 – Octree en 2D (quadtree).

4.3.2 Les arbres binaires

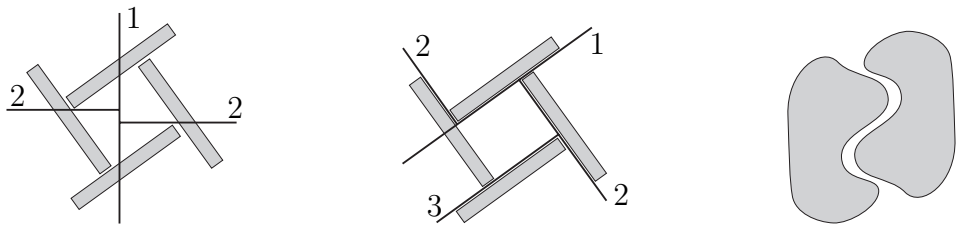
Le principe est exactement le même qu'avec les octrees, excepté qu'un seul plan est utilisé. La région est donc subdivisée en 2 sous-régions à chaque noeud. Il conduit à moins de subdivision, et s'adapte mieux. Il est donc préférable d'utiliser ce découpage. Nous traitons donc cet exemple plus en détail.

Simplicité de la subdivision :

En choisissant des plans alignés avec les axes, on réduit considérablement le coût d'un calcul d'intersection avec un rayon $P(t) = A + t.\mathbf{u}$:

– pour $x = a$, alors $t = (a - A_x)/u_x$.
– pour un plan général $\mathbf{N}.P = d$, alors $t = (d - \mathbf{N}.A)/(\mathbf{N}.\mathbf{u})$.

Le nombre d'opération est de $(1+, 1\times)$ dans le premier cas, et de $(5+, 7\times)$ dans le second. Simplifier l'expression de la surface de subdivision rend moins facile le découpage de l'espace (voir 1.23). En régle générale, les plans quelconques permettent une bonne adaptation pour les objets convexes ou les objets non imbriqués.



(a) Cas complexe pour des plans alignés avec les axes.

(b) Gestion du même cas avec des plans quelconques.

(c) Un cas mal géré.

FIG. 1.23 – Découpage d'espace : simplicité contre adaptation.

Choix de la subdivision :

La construction d'un subdivision optimale est un problème NP-complet. Tout le problème est de minimiser le découpage des objets, tout en équilibrant de l'arbre. Ces deux contraintes sont en général incompatibles.

Exemple d'heuristique favorisant le découpage, et défavorisant l'équilibre :

– s'il y a plus de deux objets dans une région, la région est subdivisée.
– choisir la position du plan de façon à ce qu'il divise par deux le nombre d'ob-

jets dans chaque sous-espace[6].
– on définit une profondeur limite pour l'arbre. Au-delà, une région n'est plus subdivisée.

Structure de données pour la description de l'arbre

un nœud subdivise la portion d'espace courante en deux. On stocke dans un nœud :

la coordonnée i à tester (on notera $P(i)$ la coordonnée en x de P).

la position a du plan. L'équation du plan de subdivision est $P(i) = a$.

deux liens : un lien f_g correspondant à la portion d'espace vérifiant $P(i) \leq a$, et un lien f_d correspondant à $P(i) > a$.

Ces liens font références, soit à un nœud si la portion d'espace associée continue d'être subdivisée, soit à une feuille sinon.

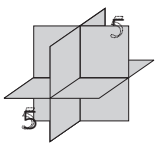
une feuille contient l'ensemble des objets inclus ou intersectants cette partie d'espace. Sa structure de données contient donc uniquement une liste de liens vers les objets correspondants. Un même objet peut par conséquent être référencé dans plusieurs feuilles.

Parcours de l'arbre de subdivision d'espace

Le parcours de l'arbre de subdivision permet de déterminer les zones de l'espace traversées sur le chemin du rayon.

Lors de l'exécution d'une recherche du premier objet intersectant, il est important de marquer les objets avec lesquels un calcul d'intersection a déjà été effectué. En effet, si un objet se partage entre plusieurs subdivisions de l'espace, il y a un fort risque d'être amené à calculer de l'intersection plusieurs fois avec cet objet si on ne prend pas de précautions. Un système de marquage temporel est le plus adapté dans ce cas : avant chaque parcours, on initialise une variable τ contenant l'heure système précise (à la microseconde) codée sous forme d'un entier. Lors du parcours, avant tout calcul d'intersection, on teste si l'objet n'est pas déjà marqué ; et si cela n'est pas le cas, on calcule l'intesection et on le marque avec τ. Le parcours suivant de l'arbre se faisant avec une autre marque τ', il m'est alors pas nécessaire de réinitialiser les marquage des objets.

[6] Ceci n'implique pas (dans le cas de l'octree) que le nombre d'objets dans chaque région soit le même. Le cas suivant aboutit à une subdivision avec 6 régions complètement vides (le numéro est le nombre d'objets de la région, les régions non numérotés ne contiennent aucun objet).



```
(Objet, Point) INTERSECTION( Rayon(A, u). ArbreA)
    τ = μtime()
    Rayon = (A, u)
    retourner BROWSETREE(Rayon,A, τ, 0, ∞)

(Objet, Point) BROWSETREE(Rayon(A, u) ArbreA, entier τ, réel min, réel max)
    si A est vide alors retourner pas d'intersection
    si A est une feuille alors
        pour tous les objets O_i dans A
            si O_i n'est pas marqué par τ
            alors (t_i, P_i) = FIRSTINTESECT(O_i,Rayon)
                 stocker l'intersection (dans l'objet)
                 marquer l'objet O avec τ.
        trouver l'intersection (t, P) la plus proche parmi les (t_i, P_i) des objets marqués τ
        si min ≤ t < max
        alors renvoyer (t, P, O)
        sinon renvoyer pas d'intersection
    si A est un nœud alors
        t = (a − P(i))/u(i)
        p = A(i) + min.u(i)   (* coordonnée i de l'origine du rayon sur sa partie courante *)
        si p ≤ a
        alors near = f_g   (* fils associé à la portion du rayon la plus proche *)
              far = f_d   (* fils associé à la portion du rayon la plus éloignée *)
        sinon near = f_d
              far = f_g
        si (t < 0) ou (t > max)   (* intervalle complet du côté near *)
        alors retourner BROWSETREE(Rayon, near , τ, min, max)
        sinon si (t < min)   (* intervalle complet du côté far *)
              alors retourner BROWSETREE(Rayon, far , τ, min, max)
              sinon hit = BROWSETREE(Rayon, near , τ, min, t)
                    si hit = pas d'intersection
                    alors retourner BROWSETREE(Rayon, far , τ, t, max)
                    sinon retourner hit
```

Algorithme 10: Parcours de l'arbre de subdivision

4.3.3 Conseils généraux

Nous terminons cette partie par quelques conseils pratiques supplementaires sur la mise en place d'une méthode de subdivision d'espace :

- l'utilisation des boîtes englobantes permet de faciliter la détermination des plans lors de la construction de l'arbre de subdivision.
- pour le calcul d'intersection avec un rayon, une fois la région déterminée, calculer d'abord l'intersection avec la boîte englobante de l'objet (ou des objets) qui s'y trouve(nt).

– dans le cas des arbres binaires, lorsqu'il ne reste qu'un seul objet dans la région, on peut continuer une subdivision dans un mode englobant : les plans choisis sont ceux qui créent les plus grands volumes vides sans intersecter l'objet. On intègre ainsi une boîte englobante (de type slabs) dans les derniers niveaux de subdivision.

(a) Boîte définie par 4 plans binaires orthogonaux. (b) Boîte définie par 6 plans binaires adaptés.

FIG. 1.24 – Utilisation de plan binaire pour définir des boîtes englobantes.

– Lorsque le domaine contient de grandes zones vides, il peut être avantageux de les faire apparaître dans la subdivision. En effet, si un rayon traverse une telle zone, il n'y a pas d'intersection à calculer, et la progression sur le rayon est rapide.
– Lorsque le nombre d'objets est faible, la gestion du l'arbre de subdivision peut se réveler plus lourde que le temps gagné à l'utiliser.

4.4 Light buffer

Lors de l'exécution de l'algorithme du ray-tracing, avec n sources de lumière, alors, en tout point où l'intensité est évaluée :

– 2 rayons sont relancés dans les directions principales de réflexion et de réfraction.
– n rayons sont lancés vers les sources de lumière.

et requièrent donc $n + 2$ calculs d'intersection de ces rayons avec la scène. Le calcul d'intersection pour les rayons d'ombres devient donc rapidement l'activité principale du lanceur de rayon dès que le nombre de sources lumineuses augmente.

Or, dans la plupart des cas, les sources lumineuses sont immobiles, et il devient possible de profiter de la cohérence spatiale des rayons d'ombres lancés en direction des sources. Pour chaque source lumineuse, tous les rayons d'ombres convergent vers les sources. On reprend alors un peu l'idée du light buffer utilisé avec le z-buffer pour le calcul des ombres, et on l'adapte au découpage de l'espace autour de la source lumineuse :

- Un light buffer est associé à chaque source lumineuse.
- Le light buffer est un cube englobant la source, et dont chaque face est découpée en cellules (tableau bidimensionnel).
- A chaque cellule est associée à une zone de l'espace autour de la source (voir figure 1.25(a)).
- Pour chaque cellule, on dispose de la liste des objets contenus dans la zone de l'espace associée à la cellule ou l'intersectant (voir figure 1.25(b)). Cette liste est triée par distance à la source croissante.
- Cette construction est réalisée en projetant la scène sur chaque face du cube où le centre de projection est la source lumineuse. Puis, les objets se projetant dans chaque cellule sont répertoriés, et ordonnés en fonction de leurs distances à la source.

Pour déterminer si un point est éclairé par une source particulière, on utilise alors la méthode suivante :

- Déterminer la distance d entre la source et le point.
- Déterminer sur quelle face du cube, et dans cette face, sur quelle cellule le point est projeté.
- Tester dans cette cellule l'ensemble des objets qui sont à une distance inférieure ou égale à d de la source.

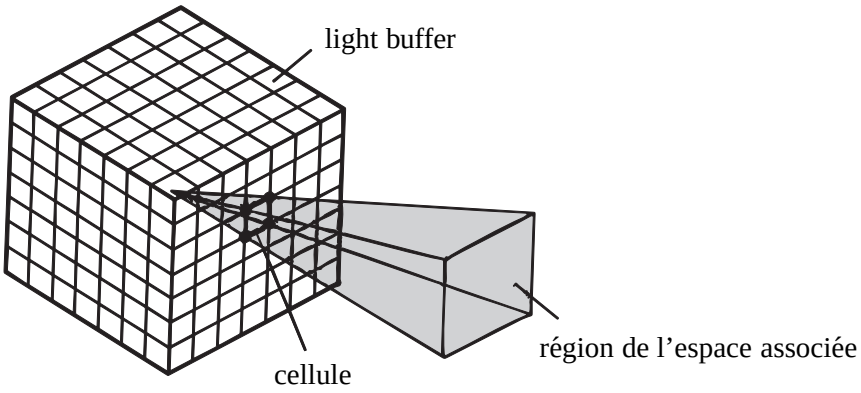
On ne teste ainsi que les objets susceptibles d'occulter la source lumineuse. Nous terminons par quelques remarques sur cette méthode.

- Comparé aux méthodes de subdivision d'espace, le light-buffer revient à spécialiser la subdivision de l'espace vis-à-vis de la source.
- Plus on s'éloigne de la source lumineuse, plus le cône s'élargit. Ainsi, lorsqu'on se trouve loin de la source, le croisement des informations du light buffer et de celles en provenance du partitionnement de l'espace présenté à la section précédente peut limiter le nombre d'objets à tester.
- Il peut être intéressant, dans chaque cellule, de conjuguer la liste des objets par une structure de données permettant de tester d'abord les intersections avec les objets les plus gros (*i.e.* ceux qui ont le plus de chance d'intersecter le rayon).
- Le coût de stockage associé à cette méthode est **très** important. Il dépend évidemment du nombre de sources, de la résolution du light buffer (*i.e.* du nombre de cellule sur chaque face du cube), et du nombre d'objets.

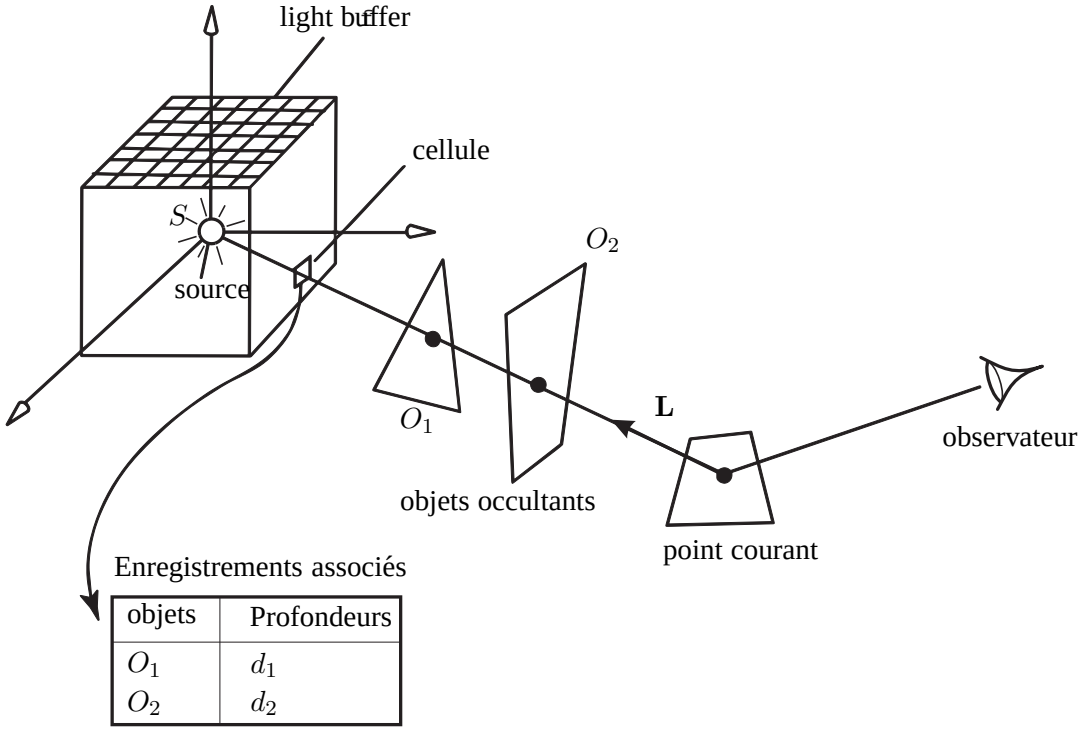
5 Conclusion

Ce chapitre présente la version de base du lancé de rayon. On pourra consulter :

- Le chapitre **??** pour les expressions des différentes luminances.



(a) Découpage de l'espace pour chaque cellule du light-buffer. La source de lumière est placée au centre du cube.



objets	Profondeurs
O_1	d_1
O_2	d_2

(b) Construction de la structure de données associée à chaque cellule. On note O_i les objets associés à la cellule, et d_i la distance entre l'objet O_i est la position de la source de lumière S.

FIG. 1.25 – Principe du light buffer.

– Les chapitres **??** et la section **??** pour la gestion des sources non ponctuelles.
– Le chapitre **??** pour les techniques avancées de rendu à base de lancé de rayons.

Annexe A : Sphere-tracing

Le sphere-tracing est une methode proposée par Hart (1994) qui permet d'effectuer le ray-tracing de surfaces implicites dans laquelle l'intersection n'est plus calculée analytiquement, mais itérativement. **Cet algorithme est particulièrement bien adapté pour le rendu de surfaces implicites**.

L'idée est particulièrement simple : pour effectuer le ray-tracing d'une surface, il suffit de savoir calculer la distance d (ou une minoration de celle-ci) d'un point P quelconque de l'espace à la surface. Si on considère la sphère de centre P et de rayon r alors : au mieux, elle est tangente à la surface ; au pire, elle n'intersecte pas la surface.

Si le point P se déplace sur un rayon, d me donne alors la distance que je peux parcourir depuis P sur ce rayon sans traverser la surface (d'où le nom de sphère tracing). La connaissance de cette distance permet ainsi de se déplacer itérativement le long d'un rayon jusqu'à l'intersection avec la surface (s'il en existe une) (voir figure 1.29).

A.1 Algorithme du sphere-tracing

Prérequis

L'algorithme du sphère-tracing demande de connaître pour la surface dont on veut effectuer le rendu :

– **une fonction de distance** $d(x, S)$ qui renvoie la distance entre le point x et le point le plus proche de x situé à la surface de S **ou une minoration de celle-ci**. Plus cette fonction de distance est proche de la distance réelle à la surface, et plus la convergence de l'algorithme sera rapide. La section A.3 donne des méthodes pour évaluer cette fonction.
– **une distance de traversement** D qui donne, à partir de la position initiale du rayon et de l'objet, la distance au-delà de laquelle plus aucune intersection ne sera trouvée (ou une majoration de celle-ci).

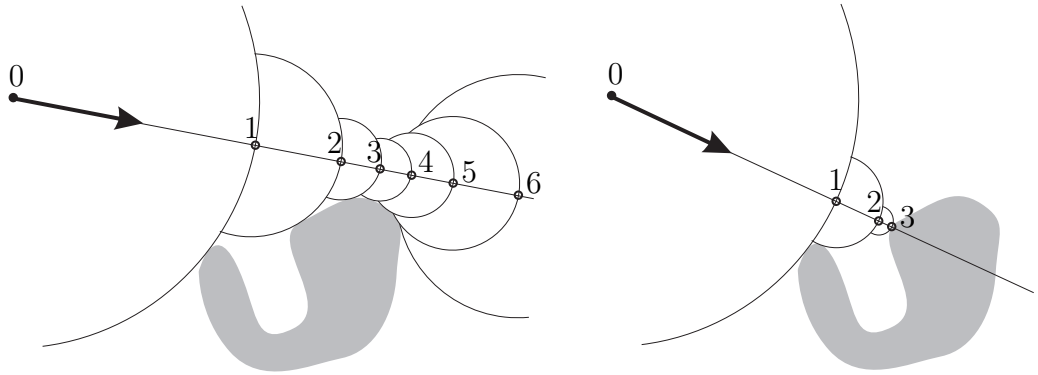
On fixe aussi une distance ϵ en dessous de laquelle on considère avoir intersecté la surface (tolérance sur la position de l'intersection).

Algorithme

On veut calculer l'intersection entre un rayon $(\Omega, \mathbf{u})$ et une surface S pour laquelle les fonctions de calcul de distance et de distance de traversement sont connus. On suppose que le vecteur $\mathbf{u}$ est normalisé (**absolument nécessaire**).

// initialisations
$t = 0$
D = distance maximale de traversement depuis Ω
dans la direction $\mathbf{u}$.
// algorithme
tant que $t < D$
$P = \Omega + t.\mathbf{u}$
$d = d(P, S)$
si $d < \epsilon$
alors P est le point d'intersection.
retourner intersection
$t = t + |d|$
retourner pas d'intersection

Algorithme 11: Algorithme du sphère tracing.



(a) cas sans intersection. (b) cas avec intersection.

FIG. 1.26 – Sphere tracing : progression itérative sur le rayon depuis son origine en utilisant la distance la plus proche à l'objet.

L'algorithme se déroule comme précisé à la figure 1.29. On avance progressivement sur le rayon en utilisant la distance maximale à l'objet. La distance de traversement est utilisée afin de stopper l'algorithme dès que possible.

Pour le calcul de la normale au point d'intersection, comne cet algorithme est généralement utilisé sur des surfaces implicites, on rappelle que la normale au point (x, y, z) d'une surface implicite $S(x, y, z) = 0$ est obtenue avec $\mathbf{N} = \nabla S(x, y, z) = (\partial S/\partial x, \partial S/\partial y, \partial S/\partial z)(x, y, z)$ (voir chapitre 1).

A.2 Distance à quelques surfaces classiques

Nous donnons dans cette section les surfaces les plus classiques pouvant être rendues avec cette méthode. On consultera l'article de Hart (1994) pour d'autres types de surface plus complexes.

Dans toute cette section, on **supposera être placé dans le repère local de l'objet** (centré à l'origine). L'équation du rayon devra donc être passée dans ce repère où les formules de distance devront être adaptées. On notera $P = (x, y, z)$ la position du point sur le rayon à l'itération courante. On rappelle que $(\Omega, \mathbf{u})$ est le rayon avec lequel on recherche l'intersection.

Sphère : centrée en O et de rayon r.

distance : $d(P, S) = \sqrt{x^2 + y^2 + z^2} - r$
distance de traversement : $D = \sqrt{\Omega_x^2 + \Omega_y^2 + \Omega_z^2} + 2.r$.
normale au point d'intersection : $\mathbf{N} = P/r$ (normalisé).

Cylindre infini : d'axe Oz et de rayon r.

distance : $d(P, S) = \sqrt{x^2 + y^2} - r$
distance de traversement : $D = (\sqrt{\Omega_x^2 + \Omega_y^2} + 2.r)/\sqrt{\mathbf{u}_x^2 + \mathbf{u}_y^2}$.
normale au point d'intersection : $\mathbf{N} = (x, y, 0)$ (à normaliser).

Cône infini : d'axe Oz et d'angle d'ouverture θ.

distance : $d(P, S) = \sqrt{x^2 + y^2}.\cos\theta - |y|.\sin\theta$
distance de traversement : (à compléter).
normale au point d'intersection : utiliser l'équation implicite du cône : $x^2 + y^2 - z^2.\tan^2\theta = 0$.

Tore : d'axe Oz, de grand rayon R et de petit rayon r.

distance : $d(P,S) = \sqrt{(\sqrt{x^2+y^2} - R)^2 + z^2} - r$

distance de traversement : (à compléter).

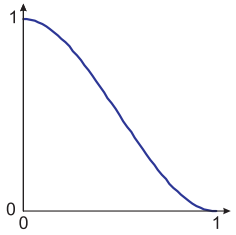
normale au point d'intersection : utiliser l'équation implicite du tore : $(x^2+y^2+z^2-(R^2+r^2))^2 - 4R^2(r^2-z^2) = 0$.

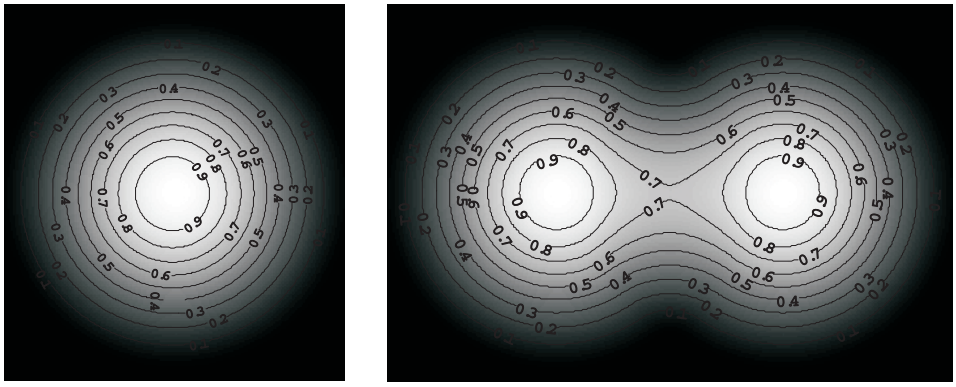
Metaballs : une metaball est un type particulier de surface équipotentielle définie à partir d'un ensemble de points P_i auxquels sont placés des potentiels R_i. Une fonction $C_R(x)$ permet de contrôler la décroissance du potentiel en fonction de son potentiel. La metaball d'équipotentiel T est alors la surface implicite définie par :

$$S(P) = T - \sum_i C_{R_i}(|P - P_i|)$$

où l'expression de la fonction de décroissance $C_R(x)$ est (par exemple) :

$$C_R(r) = \begin{cases} 2.\frac{r^3}{R^3} - 3.\frac{r^2}{R^2} + 1 & \text{si } r < R \\ 0 & \text{sinon.} \end{cases}$$





(a) Avec un point de potentiel. (b) Avec deux points de potentiel.

FIG. 1.27 – Metaballs : la fonction de potentiel avec des exemples de courbes équipotentielles.

Les caractéristiques des metaballs pour être utilisées avec l'algorithme de sphere-tracing sont :

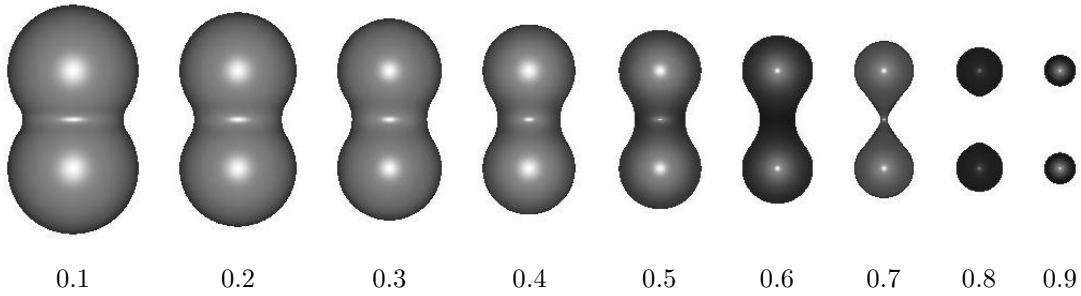


FIG. 1.28 – Metaballs : variation de la surface obtenue en fonction de la valeur de T.

distance :

$$d(P,S) = \frac{2}{3}.\left(T - \sum_i C_{R_i}(|P - P_i|)\right).\left(\sum_i R_i\right)$$

Cette expression est obtenue en utilisant les propriétés des fonctions Lipschitziennes (voir section A.4).

distance de traversement : *a priori*, $D = \max_i(|P - P_i| + R_i)$.

normale au point d'intersection :

- si on considère chaque surface équipotentielle indépendamment (P_i par P_i), la fonction $C_R(r)$ étant isotrope, la normale (non normalisée) $\mathbf{N}_i$ au point P est $P - P_i$.
- la surface finale est obtenue par composition linéaire de chaque surface équipotentielle avec pour chaque P_i un poids $C_{R_i}(|P - P_i|)$. Par conséquent, la normale au point d'intersection est :

$$\mathbf{N} = \sum_i C_{R_i}(|\Omega - P_i|).\mathbf{N}_i$$

Une variation classique des metaballs consiste à utiliser des fonctions de potentiel non isotropes (cube, tore, ...), où à utiliser des poids négatifs pour "creuser" la surface (en pratique : remplacer $C_{R_i}(r)$ par $\varepsilon_i.C_{R_i}(r)$ où $\varepsilon_i = \pm 1$).

A.3 Optimisations

Diverses optimisations peuvent être utilisées pour accélerer l'algorithme du sphere-tracing :

les objets englobants : comme pour le ray-tracing, on peut tester l'intersection du

rayon avec l'objet englobant avant d'utiliser le sphere-tracing.

l'inégalité triangulaire : le sphère tracing étant un algorithme se déplaçant itérativement sur le rayon, et le but étant de trouver l'intersection avec l'objet le plus proche, il est par conséquent plus rapide de traiter en même temps tous les objets nécessitant le sphere-tracing dans une seule boucle en stoppant le parcours du rayon dès qu'une intersection est trouvée. On note $\{O_i\}$ l'ensemble des objets, d_i la distance qu'il reste à parcourir avant d´avoir à calculer une nouvelle intersection avec l'objet O_i, D_i la distance maximale de traversement pour O_i. De cette

// initialisations
$d_{\text{last}} = 0$
$d_{\text{min}} = +\infty$
$t = 0$
$D = max_i D_i$
pour tout O_i, $d_i = 0$
// itérations
répéter jusqu'à ce que ($d_{\text{min}} < \epsilon$) **ou** ($t > D$)
    $P = \Omega + t.\mathbf{u}$
    pour tous O_i
        si $d_i - d_{\text{last}} > d_{\text{min}}$
        alors $d_i = d_i - d_{\text{last}}$
        sinon
            $d_i = d(P, O_i)$
            $d_{\text{min}} = \min(d_{\text{min}}, d_i)$
        $d_{\text{last}} = d_{\text{min}}$
    $t = t + d_{\text{min}}$

Algorithme 12: Optimisation du sphère tracing (inégalité triangulaire) : avancée progressive sur le rayon pour tous les objets en même temps.

façon, on progresse sur le rayon avec tous les objets à la fois, et on s'arrête à la première intersection trouvée sur l'ensemble des objets. Le parcours du rayon ne va ainsi pas plus loin que nécessaire.

la convexité : deux propriétés peuvent être utilisées pour accélérer les calculs si une surface est convexe :

le rejet trivial : si on se trouve à la position P sur le rayon $(\Omega, \mathbf{u}$, alors le rayon ne peut plus intersecter la surface S au delà de P si :

$$\nabla S(P).\mathbf{u} \geq 0$$

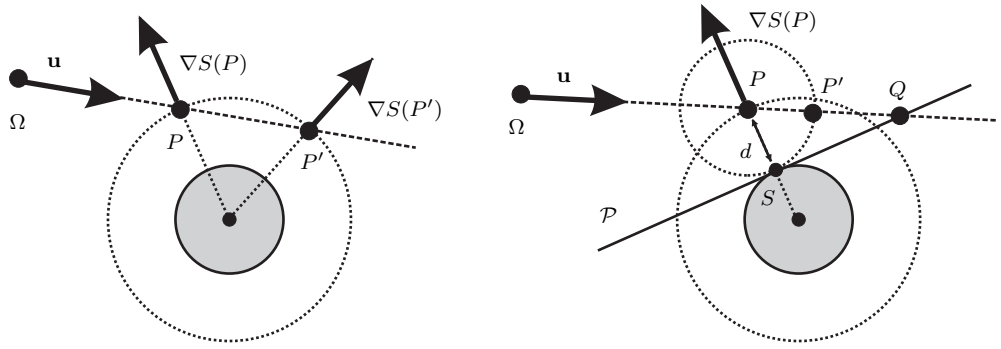
A noter que $\nabla S(P)$ est la tangente à la surface implicite $S(P) = d$ où $d = d(P, S)$. Voir aussi la figure 1.29.

Si la fonction de distance $d(x, S)$ est la **vraie fonction de distance**, cette condition d'arrêt peut se remplacer alternativement par : la distance à la surface ne décroit plus. En effet, pour une surface convexe, la variation de la distance donne des informations précieuses. Si la distance devient constante, on ne pourra pas se rapprocher plus de la surface. Si la distance augmente, alors on ne peut que continuer à s'éloigner.

le plan tangent : la surface étant convexe, la surface n'intersecte aucun de ses plans tangents. Par conséquent, si on se trouve à une distance d de la surface, et si on avance sur un rayon dont la direction $\mathbf{u}$ n'est pas celle de l'opposé de la normale, on peut avancer sur ce rayon de plus de d sans intersecter la surface. Plus précisément, la distance d' que l'on peut parcourir est de :

$$d' = \frac{d}{-\nabla S(P).\mathbf{u}}$$

Si on considère le triangle rectangle SPQ, on a $\cos \widehat{SPQ} = -\nabla S(P).\mathbf{u}$, par conséquent $PQ = d/\cos \widehat{SPQ}$. L'algorithme du sphere-tracing permet de se déplacer jusqu'en $R = P + d.\mathbf{u}$. La convexité de S nous permet de dire que l'on peut avancer de d' sans intersecter la surface (donc jusqu'en $Q = P + d'.\mathbf{u}$) puisque la surface ne peut intersecter un de ses plans tangents (voir aussi figure 1.29).



(a) Rejet trivial. (b) Plan tangent.

FIG. 1.29 – Sphere-tracing : propriétés géométriques supplémentaires pouvant être utilisée lorsque l'objet est convexe.

A.4 Distances et fonctions Lipschitziennes

Une fonction $S : \mathbb{R}^3 \to \mathbb{R}$ est dite Lipschitzienne sur un domaine D si et seulement si pour tout $x, y \in D$, il existe une constante positive finie λ telle que :

$$|S(x) - S(y)| \leq \lambda.|x - y| \tag{1.2}$$

La constante de Lipschitz, notée $\texttt{Lip}(S)$ est la plus petite constante λ vérifiant l'équation 1.2.

On peut montrer que si S est une fonction Lipschitzienne et λ est une constante inférieure ou égale à $\texttt{Lip}(S)$, alors S/λ est une distance signée à sa surface implicite. En effet, si on reprend l'équation 1.2 avec pour y, un point sur la surface S, alors $S(y) = 0$. Ceci est vrai pour tout y sur la surface, donc également pour le y le plus proche de x. Par conséquent, pour ce dernier, $|x - y| = d(x, S) = \inf_{y \in S} |x - S(y)|$. Par conséquent, pour tout x, 1.2 se réécrit :

$$\frac{1}{\lambda}|S(x)| \leq d(x, S) \tag{1.3}$$

Par conséquent, $S(x)/\lambda$ est une minoration de la distance signée à la surface (autrement dit, on ne se trouve pas plus près que $S(x)/\lambda$.

Evaluation de la constante de Lipschitz

Dans le cas où $S(x)$ est une fonction à une variable, la réécriture de l'équation 1.2 dans le cas où $|x - y|$ tend vers 0 conduit[7] à :

$$\lim_{|x-y| \to 0} \frac{|S(x) - S(y)|}{|x - y|} \leq \lambda$$

Par conséquent,

$$\lambda \geq \max_{x \in R} |S'(x)|$$

Donc, on peut choisir comme constante de Lipschitz la majoration de la plus grande dérivée (donc variation) de S dans l'intervalle réel R.

De la même façon, dans le cas où $S(x)$ est une fonction 3D (*i.e.* x est un point 3D dans la région de l'espace R), cette écriture devient :

$$\lambda \geq \max_{x \in R} |\nabla S(x)|$$

et on peut choisir comme constante de Lipschitz le maximum de la norme du gradient de S dans la région R.

[7] $|x - y| \to 0$ est bien le cas le pire, puisque $|S(x) - S(y)|/|x - y| = a$ peut être réalisé à pente constante avec $|S'| = a$.

Autres résultats d'intérêt :

- **Renormalisation** : la fonction $f/\texttt{Lip}(f)$ a pour constante de Lipschitz 1 :

$$\texttt{Lip}\left(f/\texttt{Lip}(f)\right) = 1$$

  Une renormalisation permet de faire un sorte que $|f(x)|$ représente directement la distance à la surface implicite.
- **Somme :** la constante de Lipschitz d'une somme est majorée par la somme des constantes de Lipschitz. Autrement dit :

$$\texttt{Lip}\left(\sum_i S_i\right) \leq \sum_i \texttt{Lip}(S_i)$$

- **Composition :** la constante de Lipschitz de composition de deux fonctions $f \circ g$ est le produit de leurs constantes de Lipschitz :

$$\texttt{Lip}(f \circ g) = \texttt{Lip}(f) \times \texttt{Lip}(g)$$

A.5 Fonctions CSG sur des surfaces implicites

On utilise les expressions suivantes pour générer les surfaces CSG associées à la composition des deux surfaces représentées par les fonctions implicites f et g :

opérations CSG

union : $(f \cup g)(x) = \max\left(f(x), g(x)\right)$

intersection : $(f \cap g)(x) = \min\left(f(x), g(x)\right)$

privé de : $(f \cap g)(x) = \min\left(f(x), -g(x)\right)$

opérations CSG avec fonction de blending f_b

union : $(f \cup g)(x) = \max\left(f(x), g(x)\right) + f_b(|g(x) - f(x)|)$

intersection : $(f \cup g)(x) = \min\left(f(x), g(x)\right) - f_b(|g(x) - f(x)|)$

privé de : $(f \cup g)(x) = \min\left(f(x), -g(x)\right) - f_b(|g(x) + f(x)|)$

Index