Chapitre IV

Définitions

Mots clés

Graphes, classification, lancer de rayon, parallélisme.

Résumé

Ce chapitre présente les éléments théoriques de base nécessaires à l'exposé de notre méthode d'optimisation. Ces éléments sont un ensemble d'entités algorithmiques, de relations entre ces entités ainsi que quatre graphes importants. Cette approche permet également de proposer une nouvelle classification des méthodes de lancer de rayon.

Summary

In this chapter, basic theorical elements are described : algorithmic elements, relationships between them and four important graphes. They introduce the presentation of our ray-tracing speedup technic and a new classification for classical methods.

Introduction

La classification des méthodes d'accélération en lancer de rayon présentée dans le chapitre précédent n'est pas totalement suffisante pour plusieurs raisons :

- elle ne fournit pas vraiment un ensemble de critères précis qui permettraient de classer parfaitement une optimisation dans une catégorie donnée,
- certaines méthodes se retrouvent dans différentes branches de cette classification,
- elle est basée sur des techniques existantes et peut se révéler incomplète pour d'autres à venir.

Dans ce chapitre, nous proposons une formalisation du lancer de rayon qui d'une part est utile pour la présentation générale de notre méthode et d'autre part peut permettre de classer plus rigoureusement les différentes optimisations. L'idée centrale présentée dans ce mémoire s'appuie sur ces définitions théoriques importantes qui sont ensuite associées à l'environnement informatique disponible.

Cette séparation de l'aspect théorique et de la réalité physique est fondamentale par rapport à l'hypothèse de portabilité maximale de la méthode d'optimisation proposée.

I) Définitions du lancer de rayon selon la théorie des graphes

I.1) Définition de trois entités algorithmiques pour le lancer de rayon

I.1.a) Les objets réels

Les objets réels se définissent comme les éléments géométriques de base qui composent la scène et qui sont associés à une procédure de calcul d'intersection élémentaire. Les objets réels de base sont généralement connus par l'équation de leur surface. À ces objets est associée de façon univoque une description photométrique. Ils peuvent être simples comme des facettes, des volumes quadriques, appartenir à une hiérarchie comme les feuilles d'un arbre CSG, ou les boîtes englobantes d'une hiérarchie de boîtes englobantes. Les boîtes englobantes dont on est amené à calculer l'intersection avec un rayon sont donc considérées comme des objets réels. L'espace pourrait être vu comme un objet réel d'une géométrie fixée a priori, et dont la couleur serait la couleur ambiante, qui serait transparent et qui contiendrait tous les autres objets.

On notera E_r l'ensemble des objets réels.

On distinguera $E_r = (O = \{O_1, O_2, \dots O_n\} + B = \{B_1, B_2, \dots B_m\})$ où $m \neq n$ en général, B est l'ensemble des boîtes englobantes associées à l'ensemble O des objets de la scène qui ne sont pas des boîtes.

Ces objets peuvent être regroupés de différentes façons. Par exemple :

1) Considérons un découpage en n voxels de l'espace. Le graphe des objets réels est composé de n sous-graphes regroupant les objets appartenant à un même voxel. Si les voxels sont traités indépendamment par le programme, alors ces n sous-graphes sont déconnectés les uns des autres. Dans le cas contraire, chaque sous-graphe pourra être assimilé à un sommet, et certains sommets seront connectés entre eux (pour marquer des propriétés de connexité spatiale par exemple).

2) Considérons une scène organisée selon une hiérarchie de boîtes englobantes. Le graphe des objets réels sera là aussi composé d'un graphe principal qui est

l'arborescence d'englobement des boîtes et pour chacune d'elles, d'un sous-graphe qui sera l'ensemble des objets contenus dans un volume englobant.

De manière générale, on considère donc un graphe dont chaque sommet est un sous-ensemble d'objets E'_r de E_r. Ses arêtes définissent une relation entre deux sommets (par exemple la contiguïté entre deux voxels, l'appartenance à un volume englobant, …). Il sera appelé graphe des données réelles et noté G_r. On peut déjà constater que plus G_r est structuré au lancement du programme, plus le logiciel de rendu dispose d'informations au début de ses calculs.

I.1.b) Les objets virtuels

Ces objets sont ceux qui servent à calculer les informations lumineuses selon le modèle utilisé. Dans le cas le plus simple, ce sont les rayons (primaires, secondaires - réfléchis ou transmis - ou d'ombres). Un rayon est généralement connu par un point de départ et un vecteur directeur mais il faut également ajouter à cette liste des objets virtuels, les rayons généralisés utilisés en tracer de cône, faisceau ou de crayon (pencil tracing), ainsi que les rayons secondaires créés stochastiquement ou statistiquement.

On notera E_v l'ensemble des objets virtuels.

Si l'on regroupe des rayons (par exemple selon les voxels qu'ils traversent), on définit le graphe des données virtuelles comme le graphe dont chaque sommet est un sous-ensemble E'_v de E_v et dont les arêtes définissent une relation entre deux sommets (comme la contiguïté entre deux voxels voisins). Il sera noté G_v.

Pour G_v également, on dispose de plus ou moins d'informations préalables. Par exemple, dans un cas défavorable, on peut considérer qu'il n'y a aucune relation entre les rayons, mais on peut tenir compte du fait qu'un rayon réfléchi ou transmis possède un père, ou que deux rayons passant par deux pixels voisins sont eux aussi voisins, etc… Là aussi, il existe un gisement d'information préalable qui peut être exploité pendant les calculs.

NB : Chaque rayon primaire passant par un pixel de l'écran, l'ensemble de ces pixels est donc totalement déduit d'un sous-ensemble de G_v. Les pixels ne sont donc pas des entités algorithmiques indépendantes.

I.1.c) Les processus

Un processus (on utilisera de façon synonyme le mot tâche) est un élément algorithmique qui regroupe un ensemble de calculs ou de données. Un processus peut ne contenir que les éléments de déclaration d'un objet par exemple ou inversement ne contenir aucune donnée de façon permanente. Ainsi, on peut réduire une tâche à un objet ou encore à une fonction d'intersection.

On notera P l'ensemble des processus.

On définit le graphe des processus comme le graphe dont les sommets sont les tâches et dont les arêtes symbolisent une communication bidirectionnelle entre deux sommets. Il sera noté G_p.

Soit p P. Pour tout p_i tel (p,p_i) est un arc de G_p, on a :

- si (p,p_i) existe alors (p_i,p) existe aussi (liaison bidirectionnelle, on parlera d'arête),

- p_i sera appelé voisin de p,

- l'ensemble des voisins de p sera noté V(p),

- les arcs a^+ et a^- associés à une arête sont respectivement associés à une étiquette d^+ et d^- (étiquette que l'on nommera aussi direction topologique : on dira alors que l'on va de p à p_i par la direction d^+ par exemple). On donnera dans la suite des dénominations plus explicites pour ces directions : Nord, Est, Ouest, Sud, …

- on note L l'ensemble des directions du graphe. En topologie carrée (4-connexe), L={Nord, Est, Ouest, Sud} ou en abrégé, L={N, E, O, S}.

- on notera L(p) l'ensemble des directions associé au sommet p. On appellera aussi topologie de p cet ensemble L(p).

G_p sera étiqueté de la façon suivante :

- si D est le plus grand degré des sommets du graphe, alors on dispose d'au plus D étiquettes pour colorier ce graphe,

- pour notre programme, on étiquettera les arcs du graphe de façon à ne jamais créer de cycle d'une étiquette donnée et sans que deux arcs sortants (resp. entrants) aient la même étiquette. Cette hypothèse est nécessaire car des messages vont être envoyés selon ces directions et ils ne doivent pas être émis plusieurs fois voire indéfiniment transmis.

- on impose donc qu'il existe au plus un seul arc entrant (resp. sortant) par sommet ayant une étiquette donnée (pas deux sorties Nord par exemple).

On définit G'_p comme le graphe déduit de G_p en ne gardant que les sommets de degré D. Si G'_p n'est pas le graphe vide, alors on dira que G_p a une topologie régulière. Si G'_p est connexe, alors on dira que G_p a une topologie totalement régulière. La diffusion d'un message quelconque sera optimale sur un graphe dont la topologie est totalement régulière.

Le bord de G_p est défini comme l'ensemble des sommets de G_p n'appartenant pas à G'_p.

I.2) Associations de ces trois entités

Nous avons donc caractérisé le lancer de rayon par une approche de la théorie des graphes :

1) Les données concernant les objets de la scène sont décrites par G_r.

2) Les données concernant les calculs d'optique géométrique (y compris la visualisation des pixels) sont décrites par G_v.

3) Les autres données et la dynamique des calculs est décrite par G_p.

Le résultat final (l'image) est issu de l'association de ces trois graphes (pour l'exécution algorithmique, on considère également un graphe des machines qui est décrit plus bas). Pour exposer clairement cette association, nous avons besoin de définir un certain nombre de relations.

I.2.a) (processus, objet réel)

Cette relation booléenne sera définie comme vraie si le processus considéré possède dans ses données l'objet dont il est question. Le cas symétrique n'a à ce jour pas de sens.

Cette relation résulte de la projection de G_r sur G_p.

I.2.b) (processus, objet virtuel)

Cette relation booléenne sera définie comme vraie si le processus considéré possède dans ses données l'objet dont il est question. Le cas symétrique n'a à ce jour pas de sens.

Cette relation résulte de la projection de G_v sur G_p.

I.2.c) (objet virtuel, objet réel)

Cette relation booléenne définit l'association d'un objet virtuel (i.e. un rayon) avec un objet réel (par exemple une facette). Le but du lancer de rayon est de trouver pour un rayon donné, la bonne association avec l'objet réel puis d'effectuer des calculs photométriques et géométriques. Ce calcul sera noté dans la suite inter(objet virtuel, objet réel) est sera défini comme vrai si cette intersection a une solution. Le symétrique n'a pas de sens en lancer de rayon mais en a en radiosité par exemple (si l'on considère que l'objet virtuel est l'énergie reçue par une facette).

Cette relation résulte de l'association de G_v et de G_r. Le lancer de rayon découvre complètement cette association à la fin de ses calculs. Chaque processus réalise l'association : $f : E_v \rightarrow E_r$ (voire de $E'_v \rightarrow E'_r$)

ou $E_r \rightarrow E_v$ (voire de $E'_r \rightarrow E'_v$)

telle que f soit injective et associée un objet à un vecteur pour calculer la couleur du pixel.

I.2.d) (processus, processus)

Cette relation indique qu'il y a une communication entre les deux processus. On symbolisera le fait qu'un message est échangé par la notation (processus,processus)[message].

I.2.e) (objet virtuel, objet virtuel)

Cette relation symbolise les relations qu'il peut y avoir entre deux rayons (par exemple cohérence, notion de rayon généralisé).

I.2.f) (objet réel, objet réel)

Cette relation n'est rien d'autre qu'une récriture des arbres CSG par exemple (union(A,B), différence(C,D), etc…). Elle peut également exprimer des propriétés topologiques de G_r.

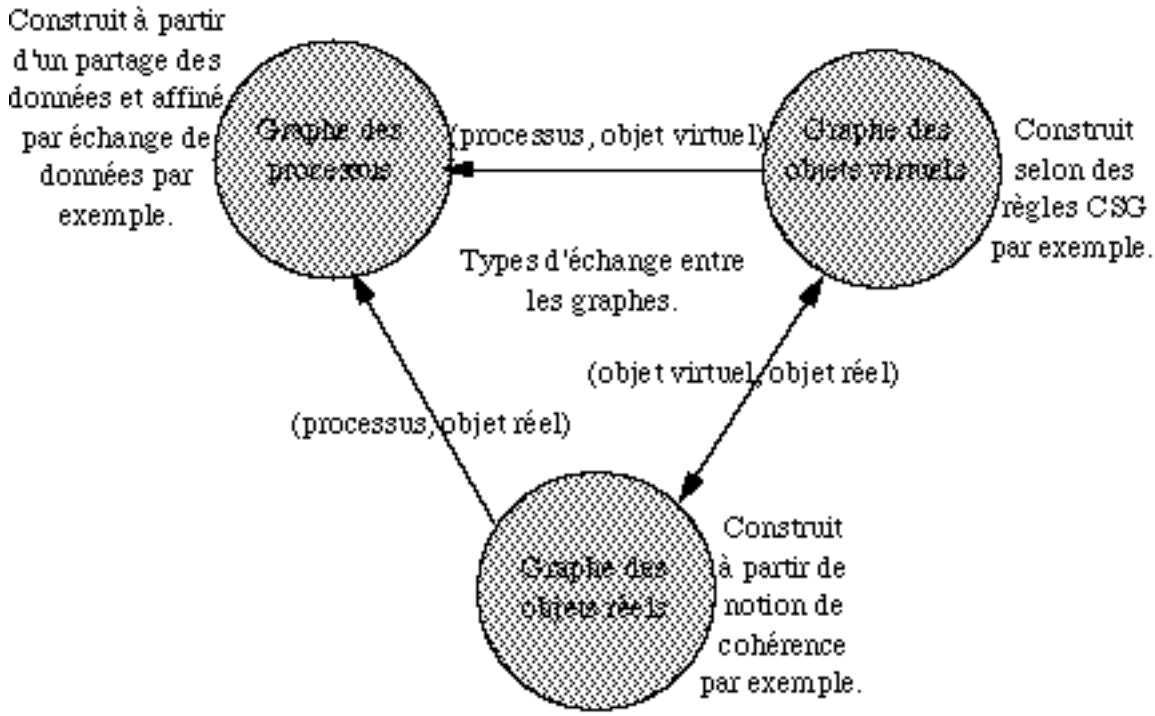


Figure IV-1 : relation entre les trois graphes

I.3) Opérateurs sur B

Dans notre approche du lancer de rayon, les boîtes englobantes sont des pavés alignés sur les axes du repère orthonormé direct Oxyz. On définit les six opérateurs suivants qui extraient les extrema en x,y,z de d'une boîte B_i :

a) en x : gauche(B_i), droite(B_i)

b) en y : pres(B_i), loin (B_i)

c) en z : bas(B_i), haut(B_i)

I.4) Nature des communications

Soient P_1 un processus, et P_x un voisin de P_1. Il est possible de caractériser quatre types de communications dans le cas général :

1) (P_1, P_x)[objet réel] : communication dans le cas des bases de données distribuées. Ces communications servent à construire ou modifier G_r.

2) (P_1, P_x)[objet virtuel] : communication utilisée dans le cas des algorithmes utilisant une définition spatiale. Elles servent à construire ou modifier G_v.

3) (P_1, P_x)[objet calcul] : communication dans le cas d'un processus ayant localement toutes les données nécessaires à un calcul et demandant à un autre processus spécialisé de réaliser ce calcul (on peut imaginer un processus spécialisé dans le calcul des sphères par exemple). Cela peut aussi caractériser un rééquilibrage dynamique de charge (c'est alors une façon de modifier G_p).

4) (P_1, P_x)[information] : communication exprimant des cas plus divers comme la terminaison d'un calcul, la synchronisation ou, dans notre cas, l'échange d'information de nature topologique.

I.5) Exemples

Si l'on retrouve ces trois graphes dans tous les programmes de lancer de rayon, leur niveau de complexité et cependant très différent. Plus il y aura d'information au niveau du graphe et plus les possibilités d'optimisation seront grandes. Par conséquent, un programme se caractérise par son niveau d'information initial et des méthodes mises en œuvre pour affiner ces informations :

- relation entre les noeuds du graphe : est-ce que l'on a réussi à mettre en évidence des sous-groupes de données ayant des propriétés définies entre elles et entre les autres sous-groupes ?

- taille et informations sur le contenu de leurs éléments : les sommets sont-ils des petits ou des grands ensembles ? Pourra-t-on les modifier ultérieurement ?

- le type et le nombre d'associations réalisables en cours de calculs : tous les sommets d'un graphe devront-ils être testés avec tous les sommets d'un autre, ou dispose-t-on d'informations permettant de réduire le nombre d'associations ?

- en cours de calcul, quels sont les mécanismes mis en œuvre pour ajouter de l'information aux différents graphes ?

Le résultat final peut également être caractérisé par les trois graphes qui auront, ou pas, été transformés et améliorés. Il est même envisageable de pouvoir quantifier la qualité des graphes de départ si l'on est capable d'établir des relations entre la structure des graphes (nombre de nœuds, d'arêtes, de sous-graphes connexes, etc…) et des critères d'efficacité (granularité, distribution, cohérence,…).

I.5.a) Exemple 1 : lancer de rayon de base

Dans le cas du lancer de rayon le plus simple par exemple, chaque graphe est réduit à un ensemble totalement déconnecté d'éléments (i.e. objets réels, rayons…) et il n'y a qu'un processus : les objets réels forment un ensemble non ordonné d'éléments et les rayons sont utilisés sans aucun lien entre eux (les rayons primaires sont lancés indépendamment et les autres rayons sont traités au fur et à mesure de

leur génération puis perdus). Calculer une image revient à faire, dans ce cas, une recherche combinatoire de la solution.

I.5.b) Exemple 2 : cas des bases de données réparties et de la division de l'écran par regroupement de rayons primaires

Hypothèses : G_p fixé, G_r estimé (par une méthode à décrire) lors de la distribution de la base de données, G_v estimé (on a les rayons primaires donc on peut sur-estimer tout l'arbre des rayons) et associé à G_p (chaque processus a en charge un certain nombre de rayons primaires et calculera tout l'arbre de calcul).

Algorithme type :

```
Pour tout rayon tel que (P1, rayon) Faire
            Pour tout objet réel Faire
                  Si non (P1, objet réel)
                  Alors (P1, Px)[objet réel]    /* on modifie Gr */
                  Fsi
                  Si inter(rayon, objet réel)  /*  association  Gv
Gr */
                  Alors garder la solution si c'est la meilleure
                  Fsi
            Fait
            calcul rayon secondaire            /* on construit Gv
*/
            lancer récursivement le programme (rayon secondaire)
            calcul couleur
Fait
```

Ici, on voit que le graphe des processus est au départ complètement connu, que le graphe des objets réel a été estimé (par un pré-traitement par exemple) et associé à G_p puis effectivement construit par échange de messages en cours de calculs [BBP90] et que le graphe des objets virtuels et construit au fur et à mesure après avoir rejeté les rayons primaires se perdant dans le vide et calculé les rayons secondaires ou d'ombre.

I.5.c) Exemple 3 : cas de la division spatiale volumique (voxels)

Hypothèses : G_p connu, G_r surestimé lors de la distribution de la base de données par voxel et donc associé à G_p, G_v inconnu.

Algorithme type :

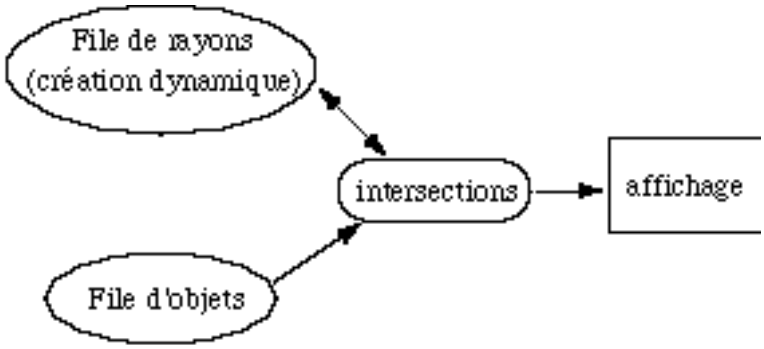
```
Tant que non ((P1, Px)[fin du programme]) Faire
            (P1, Px)[rayon entrant]            /* construction de
Gv*/
            Pour tout objet réel tel que (P1, objet réel) Faire
                  Si inter(rayon entrant, objet réel) /*associe Gv
Gr*/
                  Alors garder la solution si c'est la meilleure
                  Fsi
            Fait
            calcul des rayons sortants
            Pour tout objet réel tel que (P1, objet réel) Faire
                  Si inter(rayon sortant, objet réel) /*associe Gv
Gr*/
                  Alors garder la solution si c'est la meilleure
                  Fsi
            Fait
            Si on a pas trouvé de solution pour le rayon sortant
            Alors (P1, Px)[rayon sortant]
            Fsi
Fait
```

Ce type de programme [CWBV86] se caractérise par un graphe des objets virtuels totalement inconnus au lancement de l'algorithme. On remarquera également l'existence du test de terminaison (qui n'est pas un problème simple en parallélisme).

I.5.d) Exemple 4 : méthodes vectorielles

On peut avec cette classification, caractériser des méthodes vectorielles comme [PB85].

Le graphe des processus G_p est connu (pipeline) et figé schématiquement comme :



Algorithme type :

```
Envoyer les rayons primaires dans le pipeline
Tant que (Pipeline, rayon) Faire
            Pour tous les objets Faire
                  Si inter(rayon, objet) Alors
                  Alors garder la solution si c'est la meilleure
                        Envoyer  les  rayons  secondaires  dans  le
pipeline
                  Fsi
            Fait
Fait
```

G_r est estimé à partir de l'arbre CSG mais il n'est jamais réellement construit.

G_v est réduit à un sommet (les rayons forment un ensemble non ordonné : on peut donc avec cette classification expliquer directement les difficultés rencontrées par cette approche vectorielle puisque les calculs se font à la chaîne sans possibilité d'optimisation autre que le pipelinage des calculs).

I.6) Relations avec la classification classique des méthodes parallèles

Pour optimiser le lancer de rayon par des méthodes parallèles, deux grandes approches sont classiquement distinguées [Lu92] :

1) Parallélisation par regroupement de rayons primaires (G_p fixé, G_r estimé lors de la distribution de la base de données, G_v est sous-estimé car on ne connaît que les rayons primaires et il est associé à G_p. La scène projetée sur l'écran est découpée en zones 2D de pixels [CDHMS89]. A chaque zone, on associe un processus de calcul qui prend en charge la totalité de l'arbre de calcul pour chaque pixel de la zone (zone primaire). Deux stratégies sont envisageables pour l'accès à la base de données : (1) on la duplique autant de fois qu'il y a de processus, ou (2) on la distribue sur tous les processus et l'on introduit un protocole d'échange de données entre les processus

pour les cas où l'un d'entre eux aurait besoin d'informations ne se trouvant pas sur la base qu'il possède [BP90].

2) Partage en voxels de l'espace 3D (G_p connu, G_r estimé - en rajoutant parfois des doublons - lors de la distribution de la base de données par voxel et donc associé à G_p, G_v inconnu). Chaque processus est associé à un voxel. La part de base de données détenue par chaque processus contient les objets englobés en tout ou partie par son voxel. Durant les calculs, les communications servent essentiellement à échanger des rayons ((P_1,P_x)[objet virtuel]) entre voxels [DS84], [CWBV86] et donc à construire G_v. Pour une répartition de la charge de travail homogène, un pré-échantillonnage peut être effectué [BBP89]. Cette approche conduit à créer des voxels de différentes tailles. D'autres partitions peuvent être déduites des optimisations séquentielles utilisant les volumes englobants (hiérarchisés ou non) [Le92] ou une division récursive de l'espace [Gl84], [CW88], [AK89]. Ces méthodes font une évaluation de temps ou de densité d'objets (donc des différents graphes), et leur efficacité dépend grandement de celle-ci. L'intérêt de cette seconde approche est de pouvoir distribuer la base de données sur tous les processus. Ainsi, on réduit le nombre de calculs d'intersections et la mémoire nécessaire sur chaque site. Elle est cependant gourmande en nombre de messages (chaque rayon fait l'objet d'envoi de messages lors de sa transmission et pour le rapatriement des couleurs des intersections locales) et crée parfois des problèmes de charge de communication, voire de terminaison [Pr89].

I.7) Notion de "bon graphe"

On pourra dire que G_r est un bon graphe s'il exprime des propriétés (topologiques voire géométriques) importantes et exploitables (par exemple, un pavage carré et régulier de cubes peut être totalement défini par la connaissance de la géométrie d'un cube et d'un graphe qui sera un maillage carré ; ici G_r exprime complètement la structure de la scène modélisée).

Le "meilleur" graphe G_v, est plus facile à définir : c'est l'arbre des rayons qui donnent effectivement lieu à une intersection utile (calcul qui renvoie une information photométrique - on reviendra sur cette notion d'utile au chapitre consacré aux tests). Un algorithme utilisant comme donnée de départ le meilleur G_v n'existe pas à notre connaissance. On pourrait tout de même imaginer une application particulière qui après avoir calculé une fois pour tout G_v sur une scène donnée permettrait de recalculer une nouvelle scène en ne changeant que les paramètres photométriques ambiant, de diffusion lambertienne et spéculaires. La

géométrie ne changeant pas, il serait alors très efficace de réutiliser le graphe des objets réels et virtuels.

Il peut également envisager de tenir compte d'informations géométriques comme la zone d'espace traversée, etc…

Enfin, un bon graphe G_p doit avoir des propriétés classiques en parallélisme (extensibilité, granularité adaptée, routage simple, etc…).

I.8) Conclusion sur cette classification

Selon notre classification, l'algorithme parfait est celui qui connaît avant de lancer le programme les graphes terminaux G_p, G_v, G_r avec des sommets réduits à un élément) et qui n'a alors plus qu'à calculer inter(objet virtuel, objet réel). Ce qui est d'ailleurs trivial si l'on dispose aussi de f. Ce cas évident étant de fait difficilement réalisable (il pourrait exister sous la forme d'un algorithme travaillant sur une scène "formelle", c'est à dire un lancer de rayon pré-calculé avec des variables non instanciées, f étant la fonction qui réalise l'affectation des valeurs aux variables), on dispose en général au mieux d'un graphe complètement connu (par exemple G_p), d'une estimation des deux autres et cette connaissance va être complétée en cours de calculs en suivant, dans le meilleur des cas, une heuristique pour effectuer le moins de calculs possibles. Les méthodes d'optimisation se caractérisent donc par la connaissance dont on dispose au départ, et de la méthode et l'ordre employés pour affiner l'association de G_p, G_v, G_r . Il existe de nombreuses combinaisons possibles selon que l'on connaît ou que l'on estime les graphes au départ de l'algorithme. Cependant on peut s'apercevoir ici que, même après vingt ans de recherches, toutes les possibilités n'ont pas été implantées (Cf. Figure IV-3).

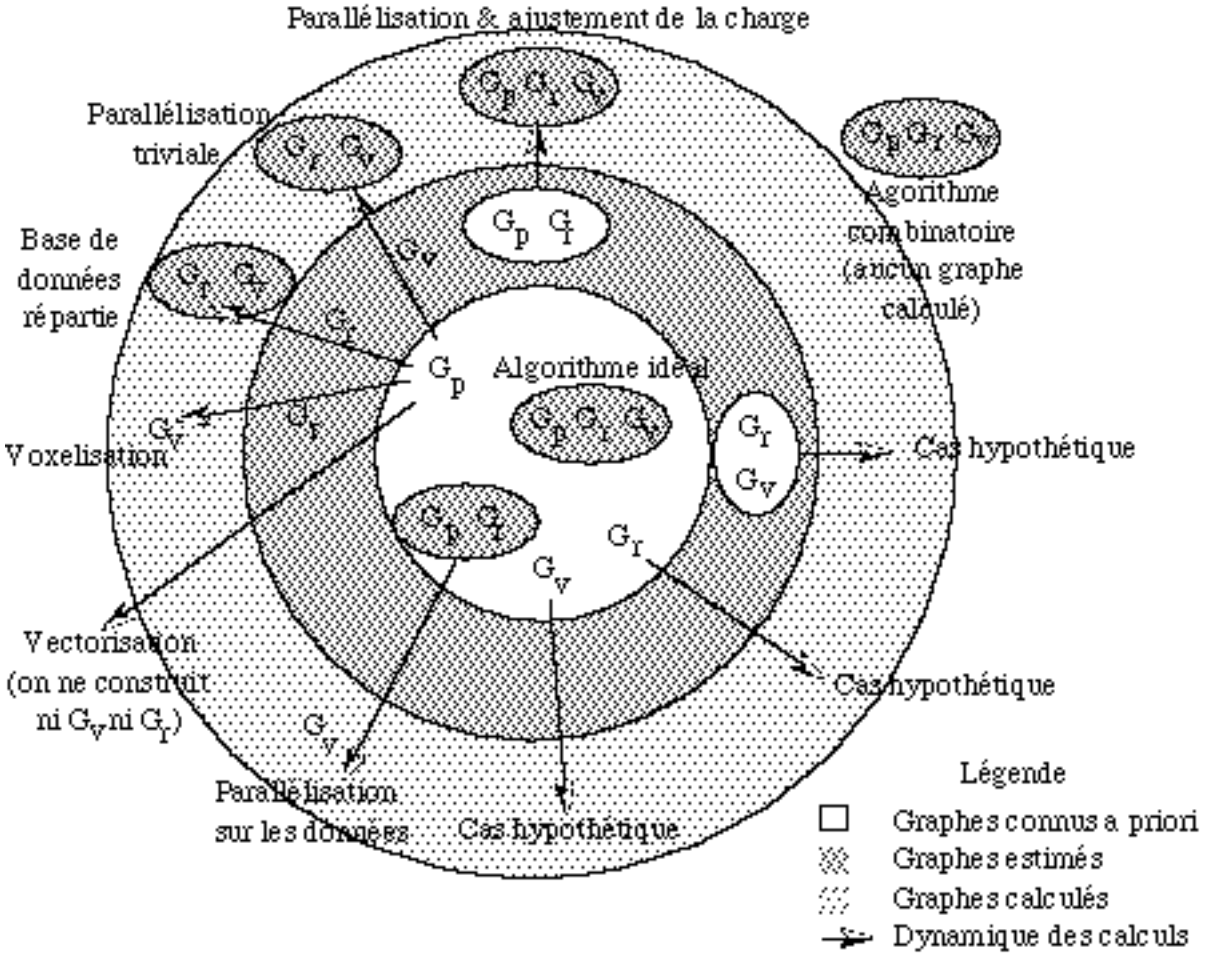


Figure IV-3 : nouvelle classification

La méthode que nous présentons dans la suite fixe G_p, estime G_v par un ordre sur les calculs, pourrait estimer G_r par rapport aux projections des boîtes englobantes et calcule G_v et G_r.

II) Caractérisation importante de certaines propriétés topologiques simples

Toujours en utilisant le formalisme précédent, nous pouvons donner un certain nombre de règles qui expriment des propriétés topologiques et qui peuvent servir à optimiser les calculs et la construction des graphes. Ces règles ont été appliquées dans notre programme. Elles sont simples et une partie a probablement été utilisée dans d'autres travaux. Cependant, il n'existe pas à notre connaissance un répertoire explicite de ces règles dans la littérature classique.

Toutes les règles (notées A_n) décrites ci-dessus ont été définies à partir de la question suivante :

Soit un point de l'espace ou une zone de l'espace traversée par un rayon à un instant donné du calcul. Peut-on déduire des événements en amont, des informations sur l'espace environnant permettant d'optimiser les calculs futurs ?

Répondre à cette question permet d'injecter de l'information servant à améliorer les graphes présentés ci-dessus.

Hypothèses :

1) $E_r = \{O_1, O_2, \dots, O_n\} + \{B_1, B_2, \dots, B_n\}$ (n=m)

2) B_i est la boîte englobante de O_i.

3) Il existe une liste L_1 des objets triés selon la profondeur des boîtes englobantes de l'œil vers le fond de la scène, et une autre L_2 du fond de la scène vers l'œil.

4) On définit le cône d'ombre circonscrit associé à un objet O et une source S comme suit :
Soit C le centre de la boîte englobante de O et P_i un sommet de cette boîte. Alpha est le plus grand angle décrit par (SC, SP_i) (Cf. Figure IV-4).

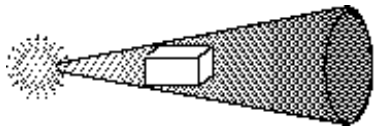


Figure IV-4

5) On définit le cône d'ombre inscrit associé à un objet O et une source S comme suit :

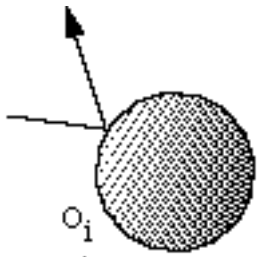
Soit C le centre de la boîte englobante de O et P un point de l'espace. Alpha est le plus grand angle tel que (SC, SP)<Alpha implique que P est dans l'ombre de O (Cf. Figure IV-5) si la boîte de O est entre S et P.

Figure IV-5

$\underline{A_1}$: Topologie des réflexions

Soit R le rayon réfléchi par l'objet O_i à l'extérieur de O_i.

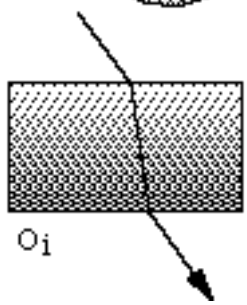
Alors inter(R, O_i) n'a pas de solution.



$\underline{A_2}$: Topologie des transmissions

Soit R le rayon transmis par O_i à l'extérieur de O_i.

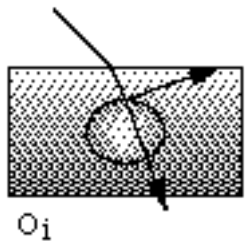
Alors inter(R, O_i) n'a pas de solution.



$\underline{A_3}$: Topologie des rayons à l'intérieur d'un objet

Soit R un rayon interne à O_i (transmission ou réflexion interne de O_i ou d'un autre objet dans O_i).

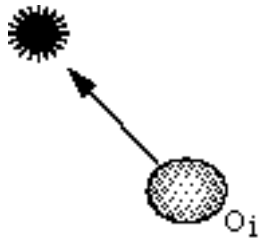
Alors inter(R, O_i) n'a de solution que pour des objets tels que $B_i \leftrightarrow B_z \neq \Delta$.



$\underline{A_1 : \text{Topologie des rayons d'ombrage}}$

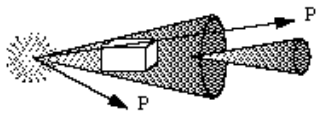
A_{4a}) Soit R le rayon d'ombre lancé à l'extérieur de O_i.

Alors inter(R, O_i) n'a pas de solution.



Soit R le rayon d'ombre lancé d'une source lumineuse en direction d'un point P situé après l'objet selon la direction Source/Objet.

A_{4b}) Si P n'est pas dans le cône d'ombre circonscrit de l'objet O_i par rapport à la source considérée alors inter (R, O_i) n'a pas de solution.



A_{4c}) Si P est dans le cône inscrit de l'objet O_i par rapport à la source considérée alors inter(R, O_i) a toujours une solution.

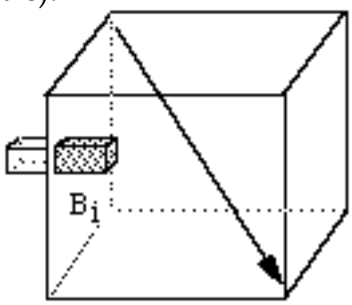
NB : les règles A_{4b} et A_{4c} pourraient être appliquées aux autres types de rayon mais leur intérêt algorithmique est plus faible pour différentes raisons :

- Cette optimisation appliquée aux ombres se fait toujours par rapport à un point fixe (lumière). Un certain nombre de calculs peut donc être stocké une fois pour toutes et par conséquent le coût de cette optimisation est faible dans ce cas.
- Les rayons primaires et secondaires doivent aboutir à un résultat complet et pas seulement à savoir si un rayon touche un objet ou pas.

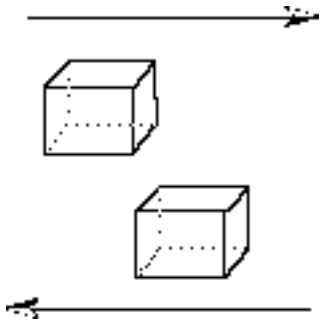
Ces critères d'optimisation rappellent dans une certaine mesure le théorème de cohérence des rayons de [OM87] déjà présenté dans l'état de l'art. L'optimisation des rayons d'ombre est ici facilitée puisqu'il est facile de stocker un certain nombre de paramètres qui ne changent pas (angle, vecteur source/centre).

$\underline{A_5 : \text{Topologie des rayons d'ombrage et secondaires}}$

Soit R un rayon d'ombre ou secondaire partant d'une zone z_a et arrivant à une zone z_b. Soit B_z la boîte définie par z_a, z_b.
Si $B_i \leftrightarrow B_z = \Delta$ alors inter(R, O_i) n'a pas de solution.



A_6 : Topologie des rayons sur un espace trié en profondeur



Soit un rayon R. Si inter(R, O_i) est la meilleure solution déjà retenue et que pour tout j>i tel que:

1) R s'éloigne de l'œil et on parcourt B selon L_1 et près(B_j)>loin(B_i) ou

2) R se rapproche de l'œil et on parcourt B selon L_2 et loin(B_j)<près(B_i)

Alors inter(R, O_i) est la solution recherchée.

Les relations présentées expriment la connaissance associée à l'espace traversé par un rayon ou en un point donné. Cependant, ces règles nous semblent pouvoir entrer dans le cadre des notions de cohérence dont [FDFH93] p. 657 donne une liste :

1) Définition : on exploite la cohérence lorsque l'on réutilise des calculs faits pour une partie de l'environnement, ou l'image, ou encore des morceaux proches, soit sans aucun changement ou avec des changements incrémentaux plus efficaces qu'un recalcul total de la solution.

2) Cohérence des objets : si un objet est entièrement séparé d'un autre, les comparaisons peuvent n'avoir besoin d'être faites qu'entre ces objets, et non entre leurs faces ou leurs arêtes (on retrouve une idée de boîte englobante).

3) Cohérence des faces : les propriétés des surfaces varient régulièrement sur celle-ci (modification incrémentale tout au long de la face).

4) Cohérence des arêtes : la visibilité d'une arête change si et seulement si elle passe derrière une arête visible ou si elle traverse une face.

5) Cohérence d'arêtes d'intersection : si deux faces planaires se coupent, l'arête d'intersection peut-être déterminée avec deux points d'intersection.

6) Cohérence des zones : un groupe adjacent de pixels affiche souvent la même surface.

7) Cohérence de la profondeur : les morceaux adjacents d'une même surface sont " bornés" en profondeur. Des surfaces différentes sont généralement séparées en profondeur.

8) Cohérence temporelle : deux images successives dans le temps sont généralement peu différentes.

Nous avons donc donné une expression logique de certains cas simples de cohérence. Dans la suite de ce mémoire, cette liste de cas sera complétée par des règles plus élaborées.

Conclusion

Dans ce chapitre, le problème du lancer de rayon a été analysé et décomposé en terme de graphes.

Cette approche permet de mieux classer les méthodes existantes mais aussi de montrer que certaines voies n'ont pas été explorées (lancer de rayon formel, recalcul partiel après changement de certains caractères photométriques, etc…).

Elle est également importante car elle introduit la méthode d'optimisation exposée au chapitre suivant et qui complète la première série de règles d'optimisations que nous venons de présenter.

Bibliographie

[AK89] : ARVO (J.), KIRK (D.). -"A Survey Of Ray Tracing Acceleration Techniques" p. 201-262 In An Introduction To Ray Tracing / Ed. A. S. Glassner. London : Academic Press Limited, 1989. - 327 p.

[BBP89] : BADOUEL (D.), BODIN (F.), PRIOL (T.). "Lancer De Rayon : Approches Parallèles". Rennes IRISA, Publication Interne n°452, Jan. 1989, 15 p.

[BP90] : BADOUEL (D.), PRIOL (T.) - "Vm_pRay An Efficient Ray Tracing Algorithm On A Distributed Memory Parallel Computer". INRIA Rennes, Rapport De Recherche n°1198.

[CDHMS89] : CROW (F.C.), DEMOS (G.), HARDY (J.), McLAUGHLIN (J.), SIMS (K.) - "3D image synthesis on the connection machine" in Parallel processing for computer vision and display / sous la direction de DEW, EARNSHAW, HEYWODD : Addison Wesley, 1989.

[CW88] : CLEARY (J.G.), WYVILL (M.). - "Analysis Of An Algorithm For Fast Ray Tracing Using Uniform Space Subdivision". The Visual Computer, 4(2), July 1988.

[CWBV86] : CLEARY (J.G.), WYVILL (B.), BIRTHWISTLE (G.M.), VATTI (R.) - "Multiprocessor ray tracing", Computer graphics forum, 5(1), march 1986, p. 3-12

[DS84] : DIPPE (M.), SWENSEN (J.).- "An Adaptive Subdivision Algorithm And Parallel Architecture For Realistic Image Synthesis". SIGGRAPH'84, 18(3), 1984, p. 149-158.

[FDFH93] : FOLEY (J.), van DAM (A.), FEINER (S.), HUGHES (J.). - “Computer Graphics : principles and practice”, Addison Wesley, 1993, 1175 p.

[Gl84] : GLASSNER (A.S.).- "Space subdivision for fast ray tracing". IEEE CG&A, 4 (10), 1984.

[Le92] : LEFER (W.). - "Parallélisation du lancer de rayon : une solution au problème de l'équilibre de la charge". Actes de GROPLAN 93, Nantes, 1992, p. 163-170.

[Lu92] : Lucas (M.). - "La Parallélisation De La Synthèse D'images Réalistes". Revue Internationale De C.F.A.O. Et D'infographie, 7(1), 1992, p.41-50.

[OM87] : OHTA (M.), MAEKAWA (M.). - “Ray coherence theorem and constant time ray tracing algorithm”. Computer Grapics 1987 (Proc. Of CG International’87) , ed. T.L Kunni, 1987, p. 303-314.

[PB85] : PLUNKETT (D.J.), BAILEY (M.J.). - “The vectorization of a ray tracing algorithm for improved execution speed.”. - IEEE computer graphics application, 5(8), Août 1985, p. 52-60.

[Pr89] : PRIOL (T.). -"Lancer De Rayon Sur Des Architectures Parallèles : Étude Et Mise En Œuvre". Thèse Soutenue À L'université Rennes I, 1989, 136 p.