Chapitre VI

Résultats numériques

Mots clés

Lancer de rayon, tests de performance

Résumé

Ce chapitre regroupe un ensemble de tests dont le but est de valider les optimisations présentées, d'étudier et de comparer les performances de notre algorithme de lancer de rayon.

Summary

This chapter is composed with a set of tests. The goal is to prove the rightness of our optimizations, to study and compare performances of our ray-tracer.

Introduction

Après avoir présenté en détail l'ensemble des optimisations que nous avons mises au point pour le lancer de rayon, et en nous appuyant sur le protocole de tests proposé au chapitre précédent, il est maintenant possible de réaliser une batterie de tests qui auront quatre buts :

- montrer que les règles d'optimisations (A_n et B_n) sont efficaces en séquentiel et en parallèle,
- montrer que ces optimisations ne faussent pas la justesse du programme (on calcule toujours la même image),
- évaluer et comparer les performances de cet algorithme avec une autre méthode sensiblement différente,
- montrer la portabilité et la robustesse du programme.

I) Calibrage de l'algorithme

Pour vérifier l'action des différentes optimisations décrites dans ce mémoire, nous avons effectué une série de tests qui consistent à activer les unes après les autres ces optimisations et ce sur trois scènes différentes (les calculs se font sur un écran 1024*1024) :

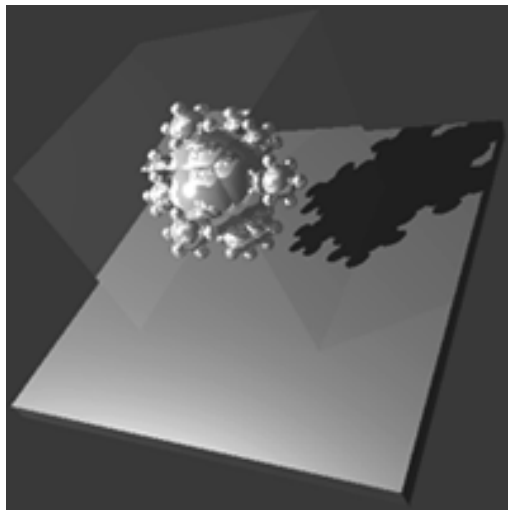


Figure VI-1 : Presse-papiers

- La scène "presse-papiers" repré-sente un cube transparent posé sur un autre cube et englobant un flocon de sphères réfléchissantes. Elle est en tout composée de 40 objets dont une lumière (Cf. Figure VI-1).

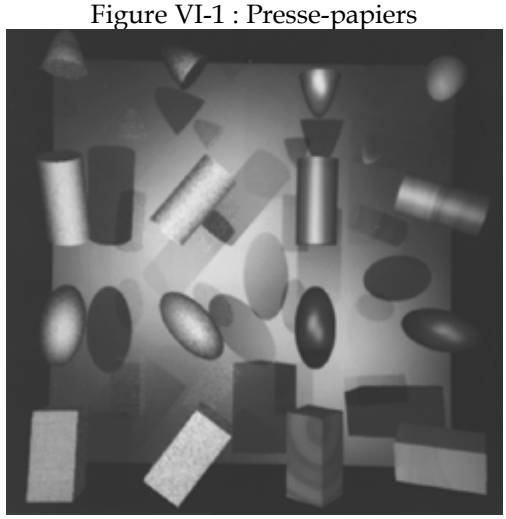


Figure VI-2 : Objets

- La scène "objets" représente un miroir devant lequel sont alignés les objets de base (texturés) de notre algorithme (sauf les facettes triangulaires). Elle est composée de 20 objets dont 3 lumières (Cf. Figure VI-2).

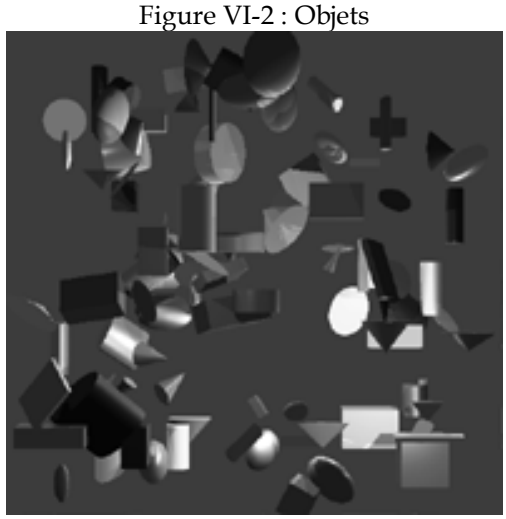


Figure VI-3 : Scène aléatoire

- La scène "aléatoire" est un ensemble de 100 objets dont la géométrie et la photométrie ont été déterminées aléatoirement (Cf. Figure VI-3).

T_1) Pour le premier test, aucune optimisation n'est activée et un seul processus est lancé.

Pour les tests suivants, 1024 processus sont lancés en parallélisme simulé sur une RS6000-320 avec 16 Mo de mémoire.

T_2) Le second test ajoute les cônes d'ombre (règles A_4).

T_3) Le troisième test active en plus le tri en profondeur des objets, et utilise les projections min/max des objets sur l'écran (règle A_6).

T_4) Le quatrième test active en plus les déductions topologiques au niveau du processus (règles B_1, B_2 et B_4).

T_5) Le cinquième test ajoute les déductions au niveau de la zone de processus (règles B_{0a} et B_{0b}).

T_6) Le sixième ajoute les déductions liées aux boîtes englobantes (règle A_5).

T_7) Le test numéro sept introduit en plus les règles pour les petits objets (règles B_3).

T_8) Le huitième ajoute les déductions internes à un processus (vu et non vu : règles B_5).

T_9) Le test neuf permet l'ajustement des boîtes (règle B_6).

T_{10}) Le dixième ajoute les déductions sur les frontières des processeurs (cas particulier de la règle B_2).

T_{11}) Le test onze active toutes les déductions des ombres (boîtes englobantes, activation des tris en profondeur, règle A_7).

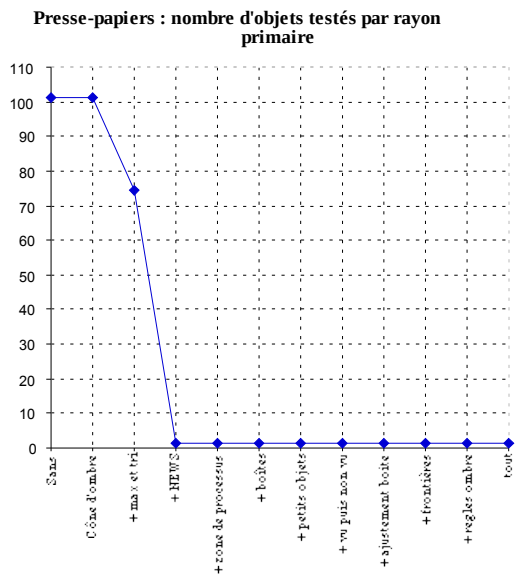
T_{12}) Enfin, le dernier test active toutes les déductions sur les rayons secondaires (boîtes englobantes, activation des tris en profondeur).

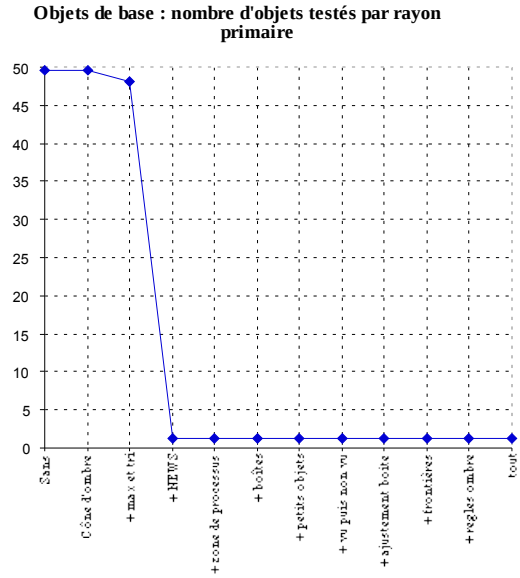
Il faut noter que toutes les optimisations ne sont pas totalement indépendantes (par exemples, les boîtes sont utilisées pour les rayons d'ombre et pour les rayons

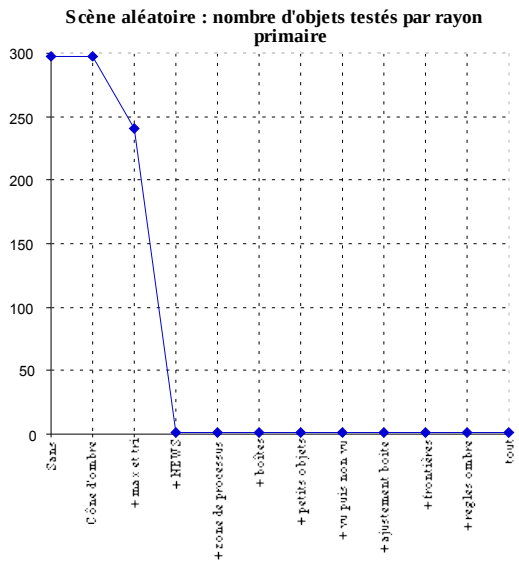
secondaires mais sont ajustées grâce au traitement des rayons primaires) et que certaines optimisations peuvent être activées par morceau. C'est pourquoi, nous ne pouvons pas toujours présenter ces tests en spécifiant que la règle A_x ou B_z a été ajoutée. Cependant, toutes ces règles sont finalement passées en revue.

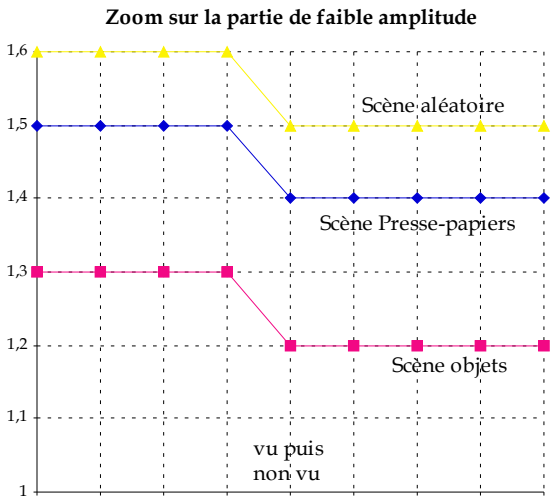
Enfin, certaines des règles activées n'étant efficaces qu'en parallélisme vrai, nous pouvons déjà savoir qu'elles n'apporteront pas de gain pour ces tests. Cependant, il est tout de même intéressant de savoir si elles sont coûteuses ou pas lorsqu'elles sont inefficaces.

I.1) Calibrage des rayons primaires









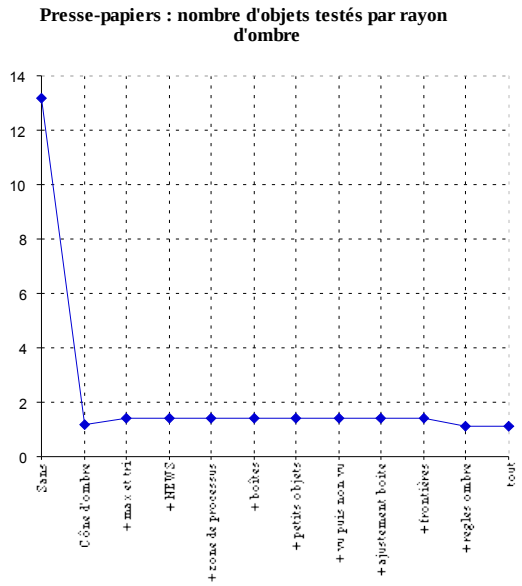
Figures VI-4a, VI-4b, VI-4c : Résultats de l'activation successive des optimisations pour les rayons primaires

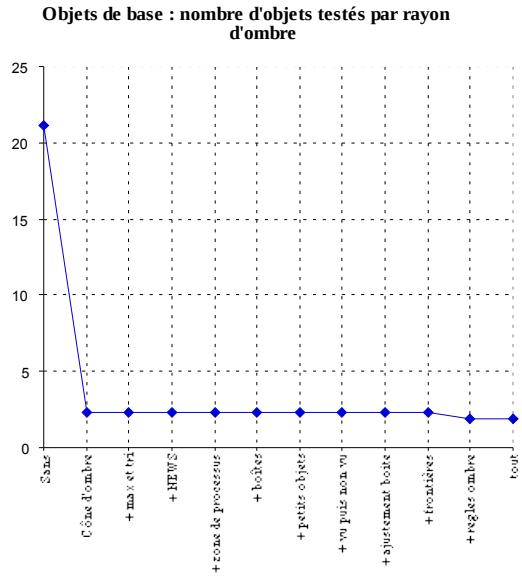
On constate un comportement similaire sur les trois courbes. Deux règles sont efficaces pour les rayons primaires :

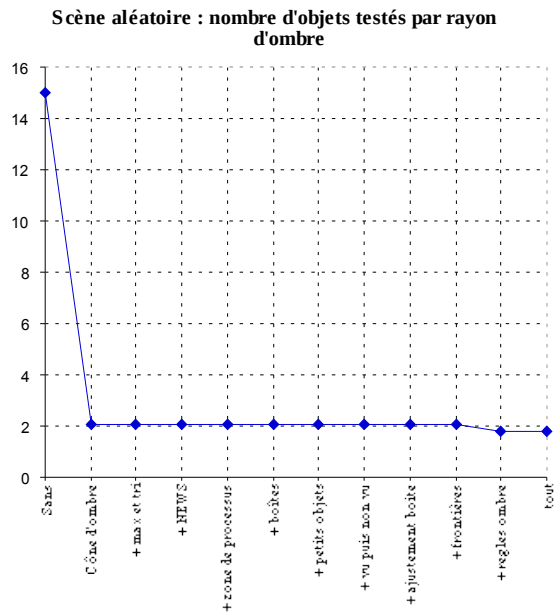
- le tri des objets,
- les déductions topologiques liées à un processus (ces règles donnent d'ailleurs le plus gros gain).

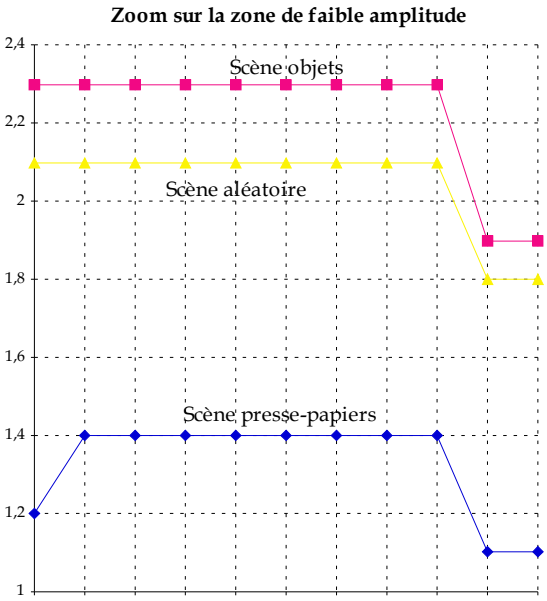
Les trois scènes n'étant pas closes, il est normal que l'on semble tester parfois plus d'objets par rayon qu'il n'y en a dans la base de données.

I.2) Calibrage des rayons d'ombrage









Figures VI-5a, VI-5b, VI-5c : Résultats de l'activation successive des optimisations pour les rayons d'ombrage

Deux règles améliorent les performances de l'algorithme pour les rayons d'ombres :

- l'utilisation des cônes d'ombre,
- l'activation des règles sur les boîtes englobantes.

Cependant, contrairement aux apparences, la deuxième règle n'est pas nécessairement beaucoup moins efficace que la première : en effet, après l'application de la règle des cônes, peu de gains peuvent encore être réalisés c'est pourquoi on ne gagne pas grand chose. Si l'on avait inversé l'ordre d'activation des règles on aurait sans doute vu plus de gain apparaître pour la seconde.

On notera également pour la scène presse-papiers que la remonté du nombre d'objets testés lors de l'introduction des tris, n'est qu'un artefact : la réorganisation des données fait par hasard disparaître un cas favorable (on détermine qu'il y a une ombre dès que l'on a intersecté un objet : l'ordre de stockage des objets n'est donc pas indifférent).

I.3) Calibrage des rayons réfléchis et transmis

Il n'y a qu'un test qui en parallélisme simulé permet de diminuer le nombre d'objets testés par rayon réfléchi et transmis (T_{12} : activation des tris et des boîtes englobantes). Lorsque l'on applique ces optimisations, on passe de :

* 51,4 objets testés à 5 pour la scène presse-papiers,
* 169,5 à 25,2 pour la scène des objets de base,
* 194,3 à 8,1 pour la scène aléatoire.

I.4) Résultats globaux du calibrage

La figure VI-6 montre le pourcentage moyen d'objets testés pour tous les rayons confondus et par rapport au nombre total d'objets de la base de données. On notera l'allure caractéristique des courbes que l'on va retrouver ci-dessous. On remarquera que sur ces trois scènes tests, on finit par tester en moyenne peu d'objets (7,2% pour le presse-papiers, 14,3% pour objets, 2,2% pour la scène aléatoire).

Les optimisations semblent d'ailleurs avoir plus d'impact sur la scène

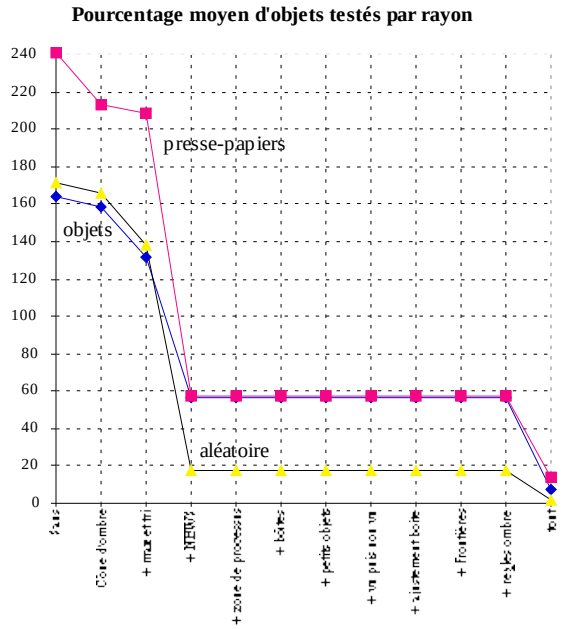


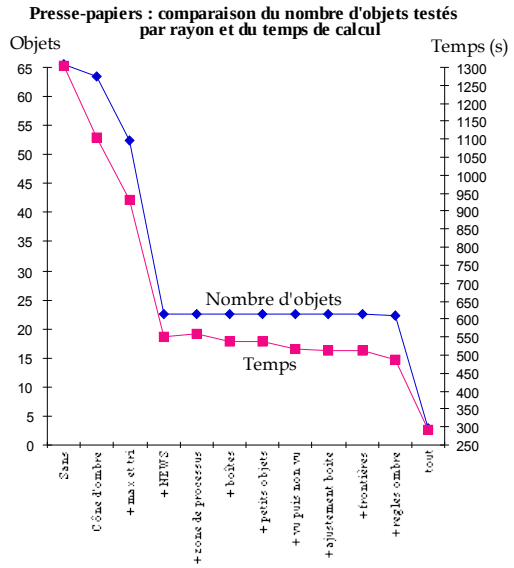
Figure VI-6 : pourcentage d'objets testés

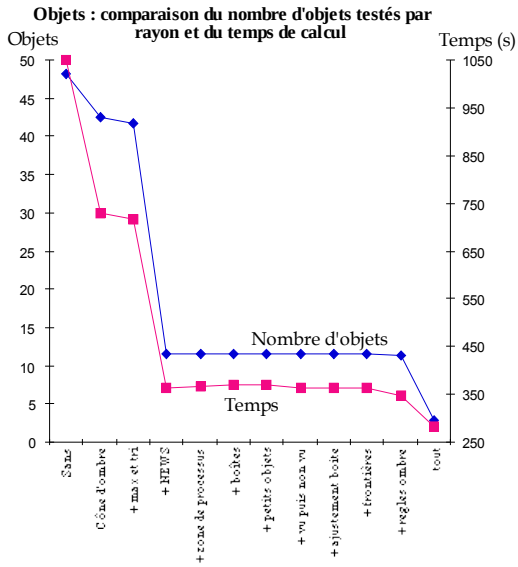
aléatoire mais cela est sans doute dû au fait que cette scène comporte plus d'objets, donc les résultats sont plus visibles.

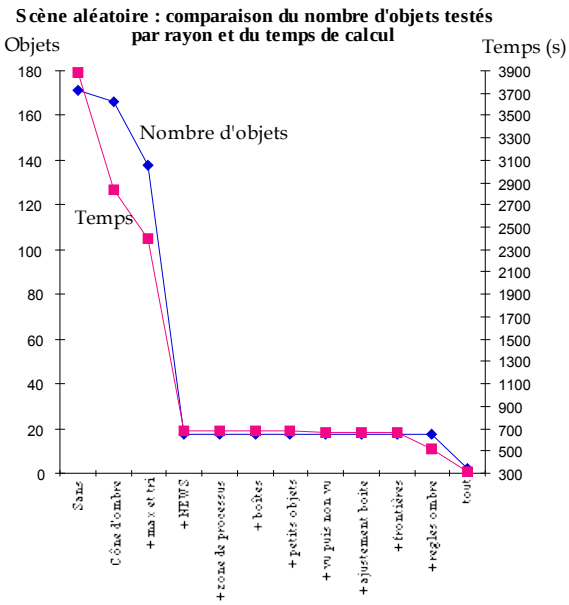
Enfin, si au départ des tests on teste plus de 100 % d'objets, c'est simplement parce que les trois scènes ne sont pas fermées.

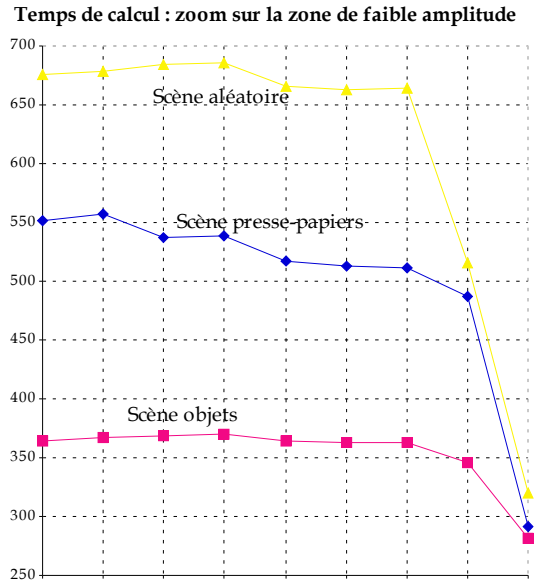
Tous les résultats des paragraphes I.1) à I.3) prenaient en compte le nombre d'objets testés par type de rayon. Comme nous l'avons déjà dit, ce paramètre est important mais doit être comparé au temps de calcul pour que l'on soit sûr qu'il exprime effectivement un gain. De plus des gains purement algorithmiques pourraient échapper à ce décompte d'objets.

Les trois courbes suivantes présentent sur le même graphique le nombre d'objets testés pour tous les rayons confondus et le temps de calcul total.









Figures VI-7a, VI-7b, VI-7c : Résultats de l'activation successive des optimisations tous rayons confondus

On notera tout d'abord que les courbes de temps et d'objets testés ont des allures qui s'accordent fortement. En fait, seule l'amplitude des variations est différente.

En particulier, la règle de déduction locale à un processus (<j'ai vu un objet puis je ne le vois plus>) ainsi que l'ajustement des boîtes englobantes et l'activation de toutes les règles sur les ombres font apparaître un gain en terme de temps qui a du mal à être montré en terme de nombre d'objets (l'examen des chiffres montre qu'en fait, ce gain touche la première décimale des résultats).

Enfin, on notera que les règles qui ne réussissent pas à générer un gain sur notre jeu de scènes sont très peu pénalisantes en temps et que les optimisations peuvent être plus ou moins efficaces selon le type de scène.

I.5) Meilleurs résultats

Enfin, pour conclure cette première phase, le programme a été lancé sur chaque scène avec 4096 processus de façon à obtenir le meilleur résultat possible (comme il y a relativement peu d'objets, la gestion des messages n'est pas trop pénalisante).

I.5.a) Presse-papiers

- temps de calcul : 85,4 s
- nombre d'objets testés tous rayons confondus : 2,6
- pourcentage moyen de la base de données testé par rayon : 6,5 %

I.5.b) Objets de base

- temps de calcul : 81,6 s
- nombre d'objets testés tous rayons confondus : 2,7
- pourcentage moyen de la base de données testé par rayon : 13,4%

I.5.c) Aléatoire

- temps de calcul : 106,3 s
- nombre d'objets testés tous rayons confondus : 2,1
- pourcentage moyen de la base de données testé par rayon : 2,1 %

I.6) Justesse de l'algorithme

Il a été dit que l'on pouvait en partie vérifier la justesse de l'algorithme optimisé grâce à la notion de rayon utile en prenant comme référence les résultats de l'agorithme non optimisé et en faisant la comparaison avec les résultats optenus avec toutes les optimisations.

Nous avons vérifié que la scène "presse-papiers" comprenait **1 008 889** rayons utiles et que ce chiffre était constant pour tous les tests.

Nous avons constaté une variation entre les deux tests, de 9 rayons pour la scène "objets". Le nombre de rayons utiles passe de **658 386** à **658 395** (la variation sur les d'ombres).

Enfin, nous avons constaté une variation de 114 rayons pour la scène "aléatoire", le nombre de rayons utiles passant de **703 049** à **703 163** (la variation se faisant sur les rayons d'ombre).

Nous n'avons pas pu déterminer avec certitude l'origine de ces écarts très faibles (moins de 0,05 % ce qui montre que notre protocole de tests permet de

mesurer finement la qualité d'un logiciel). Ils peuvent être dus à trois causes différentes :

- erreur de programmation sur les versions non optimisées ou sur les versions optimisées du programme,
- erreur de comptage des rayons (il a été dit que le comptage des intersections utiles pour les ombres était plus compliqué à cause des transparences),
- erreur d'aliassage (des calculs ou structurel) du programme.

La différence pixel à pixel des deux images donne les résultats suivants :

3574 pixels sur 1048576 sont différents soit 0,34%

Différence moyenne par composante	Maximum de différence
R : 24/255 soit 9,5% (écart type 16,3)	R : 38,43%
V : 3/255 soit 1,0% (écart type 2,8)	V : 13,33%
B : 31/255 soit 12,3% (écart type 19,7)	B : 39,22%

Visuellement, "l'image différence" fait apparaître quelques taches sombres dans des tons rouges/bleus mais il est extrêment difficile de différencier les deux résultats sans cette "image différence" à cause de la faible amplitude moyenne de l'écart et du peu de pixels affectés (moins de 1%).

Malgré ces faibles variations, nous pouvons conclure à la validité des règles d'optimisation utilisées : cette variation est totalement imperceptible à l'œil et il n'est même pas sûr qu'elle soit quantifiable sur la couleur finale de chaque pixel (on cherche à calculer trois valeurs entre 0 et 255). Il serait d'ailleurs intéressant de se poser la question de savoir jusqu'à quel niveau de perte d'information on peut aller sans que cela soit détectable. En effet, il ne sert à rien de calculer plus d'information si l'image finale ne peut pas les faire apparaître et nous avons sans doute là une source d'optimisation possible.

II) Test séquentiel

II.1) Configuration

Tous les calculs d'images qui suivent sont effectués sur un écran de 512*512 pixels avec un algorithme de lancer de rayon qui accepte jusqu'à 10 niveaux de récursivité.
La scène test de cette seconde série est composée de 156 objets dont 3 lumières (Cf. Figure VI-8).

Figure VI-8 : Scène salle à manger

II.2) Résultats de référence

Résultats de référence pris sur une station RS6000-320, 16 Mo de RAM, avec un seul processus, aucune optimisation (boîte englobante, règles, …) :

Nombre d'objets testés par rayon primaire :	**155**
Nombre d'objets testés par rayon secondaire :	**165,4**
(la scène n'est pas totalement close à cause de la fenêtre)	
Nombre d'objets testés par rayon d'ombre :	**99,0**
Nombre d'objets testés par rayon :	**130,6**
Temps de calcul :	**2419,8 s**

III) Tests parallèles

III.1) Parallélisme simulé

Nous avons fait varier le nombre de processus en parallélisme simulé (de 1 à 4096 processus) ; la scène calculée est celle de la Figure VI-8.

III.1.a) Évolution du nombre d'objets testés par rayon et du temps de calcul

On constate sur la Figure VI-9 que les deux paramètres diminuent toujours ensemble jusqu'à 4096 processus. Notre station commençant à "swapper", le temps de calcul augmente alors que le nombre d'objets testés par rayon continue de descendre. De plus, comme cela est montré au paragraphe suivant, la gestion des messages n'est plus alors négligeable. Pour 4096 processus, on ne teste que 3% de la base de données par rayon.

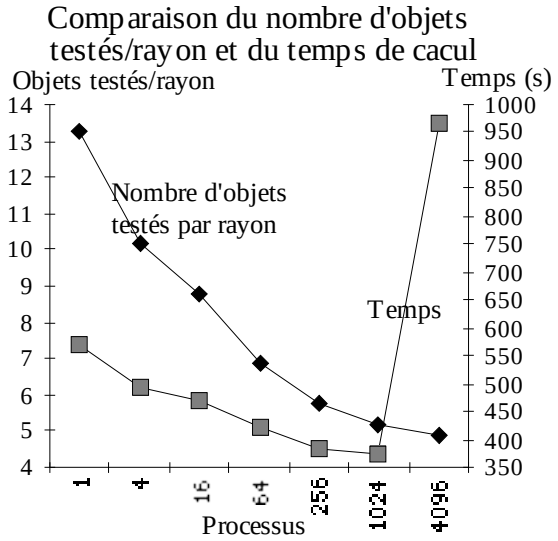


Figure VI-9

On notera que le meilleur résultat est 6,4 fois plus rapide que le programme de référence (sans optimisation) alors que l'on teste 22,8 fois moins d'objets par rayon en moyenne.

III.1.b) Évolution du nombre de messages et de leur temps de gestion

On constate sur la Figure VI-10 une augmentation exponentielle du nombre de messages générés (et naturellement du temps de gestion de ces messages). Si l'augmentation du nombre de processus est bien bénéfique, il est cependant impossible d'en avoir un trop grand nombre sur un même site. Le parallélisme vrai est nécessaire si l'on veut gérer plus de 1024 processus sur une telle scène (voire moins si l'on a plus d'objets).

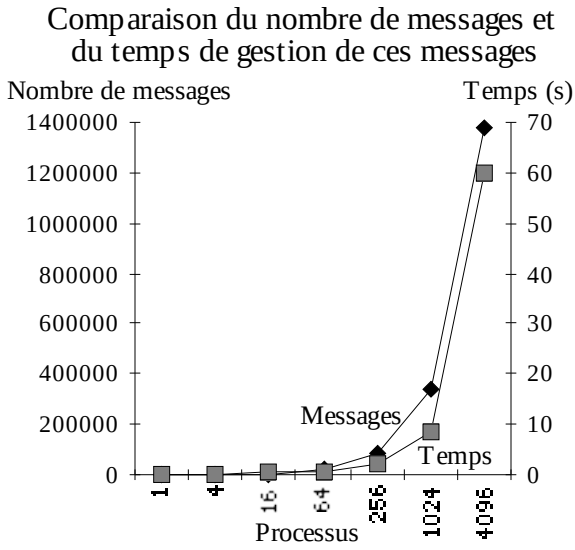


Figure VI-10

III.1.c) Évolution en fonction du nombre d'objets

Le programme qui a servi à générer la scène aléatoire a été repris de façon à créer des scènes comprenant entre 1 et 2250 objets (par pas de 250). La probabilité de

générer une source lumineuse est de 1/16 (avec 10 sources maximum), un objet réfléchissant de 1/16, un objet transparent de 1/16, réfléchissant et transparent 1/16, un objet mat 3/4 (ces probabilités ont été choisies arbitrairement).

256 processus calculent une image de 512*512 pixels (en parallélisme simulé sur la RS6000-320). Les résultats auraient pu être meilleurs mais nous avons préféré avoir moins de processus pour tester le plus d'objets possible sans que l'ordinateur ne "swappe". De plus, la scène étant générée aléatoirement, c'est bien l'ordre de grandeur de ces résultats et l'évolution générale qui sont intéressants, pas les chiffres à la décimale près.

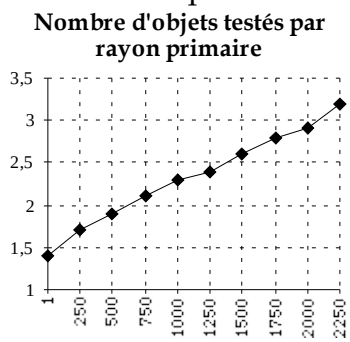


Figure VI-11a

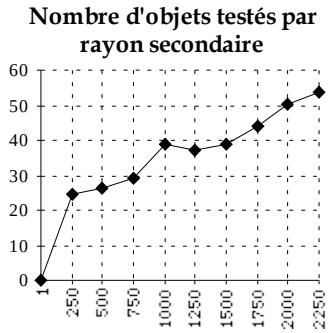


Figure VI-11b

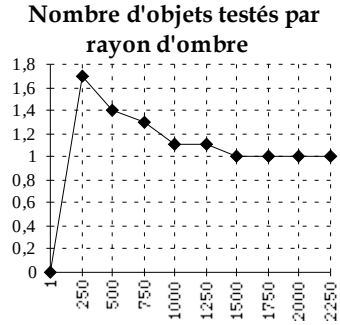


Figure VI-11c

Le comportement du nombre d'objets testés par rayon primaire (Figure VI-11a) semble linéaire sur cet échantillon de scènes : on teste environ 1,4+0,001*objets contenus dans la base de données par rayon primaire. Pour des scènes modélisées avec des objets complexes, ce résultat est très bon car on a alors rarement plus de 1000 objets. Si la scène a été créée avec des facettes le résultat est toujours intéressant même si l'on a un million de facettes.

Le nombre d'objets testés par rayon réfléchi et transmis (Figure VI-11b) semble plus irrégulier mais suit très grossièrement une progression linéaire de 20+0,01*objets contenus dans la base de données.

La courbe des ombres est très particulière (Figure VI-11c). En effet, les résultats semblent montrer que plus il y a d'objets, meilleur est le comportement du lancer de rayon. En fait, telle que la scène est définie, plus il y a d'objets et plus il y a d'ombres sur les objets. Or notre algorithme a une très grande efficacité lorsqu'il a déjà découvert une ombre. Donc, plus les surfaces ombrées sont importantes, meilleurs sont les résultats. Cependant, nous ne pouvons pas tirer de règles générales pour une scène quelconque : l'efficacité des cônes d'ombre est par exemple totale sur une scène qui ne serait modélisée qu'à l'aide de sphères mais moins bonnes sur des objets fortement asymétriques. La seule conclusion est que les résultats sont meilleurs que sur les tests des rayons réfléchis et transmis, et parfois meilleurs que sur les rayons primaires.

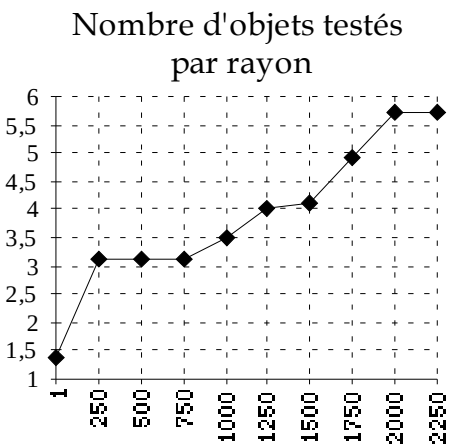


Figure VI-11d

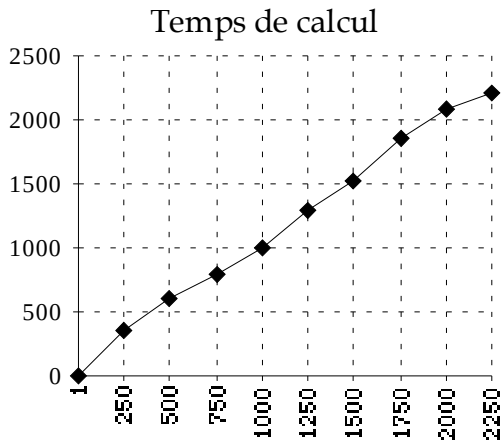
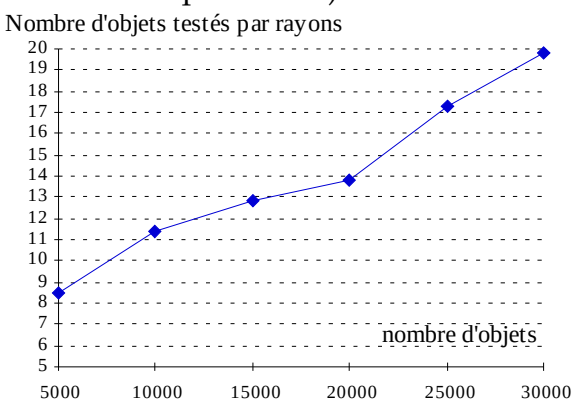


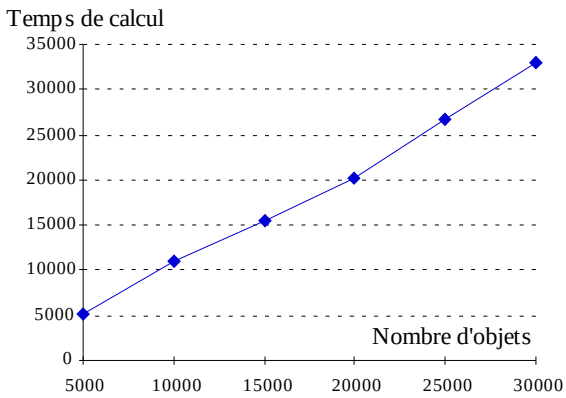
Figure VI-11e

La courbe de résultats concernant tous les rayons confondus (Figure VI-11d) n'est pas surprenante après l'analyse qui a été faite sur les trois types de rayon : on retrouve très grossièrement une progression linéaire (plus irrégulière lorsqu'il y a peu d'objets) du type 1,5+0,02*nombre d'objets dans la base de données.

Enfin, les temps de calculs (Cf. Figure VI-11e) semblent clairement linéaires sur ce jeu de scènes (les irrégularités constatées sur les autres courbes sont fortement gommées). Grossièrement, on peut dire que le temps de calcul est d'une seconde par objet de la base de données (sur RS6000-320). Pour mieux mettre en évidence ce phénomène, une dernière série de tests a été effectuée sur une machine possédant plus de mémoire (IBM 3AT avec 128 Mo de mémoire vive) et une partition écran de 256 processus (les optimisations sont donc un peu moins bonnes mais le besoin mémoire est plus faible).



Nombre d'objets testés par rayon en fonction du nombre d'objets



Temps de calculs en fonction du nombre d'objets

Figures VI-12 : Confirmation du comportement quasi-linéaire de l'algorithme

III.2) Parallélisme vrai

III.2.a) Configuration

Pour ce test, nous avons distribué les calculs sur un réseau hétérogène de machines en lançant le programme d'abord sans messages entre machines puis en le relançant en ajoutant ces messages entre stations (les messages à l'intérieur des processeurs étaient activés dans les deux cas). Nous exprimons les résultats en terme d'objets testés par rayon ainsi qu'en temps de calcul.

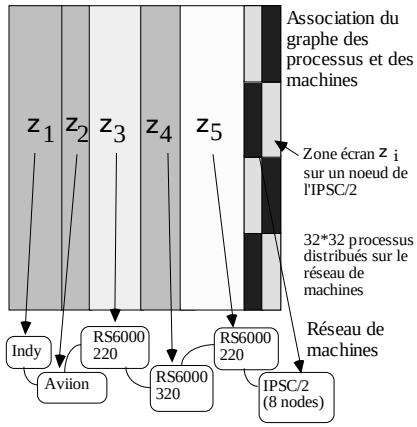


Figure VI-13

Un ensemble de 1024 processus (32x32 en topologie carrée) est distribué sur un réseau de 6 machines (Cf. Figure VI-13). Une de ces machines est un IPSC/2 à huit processeurs : nous testons donc deux niveaux de parallélisme vrai ici. L'écran est de 1024*1024 pixels.

La scène calculée est celle de la Figure VI-14 (201 objets + 4 lumières).

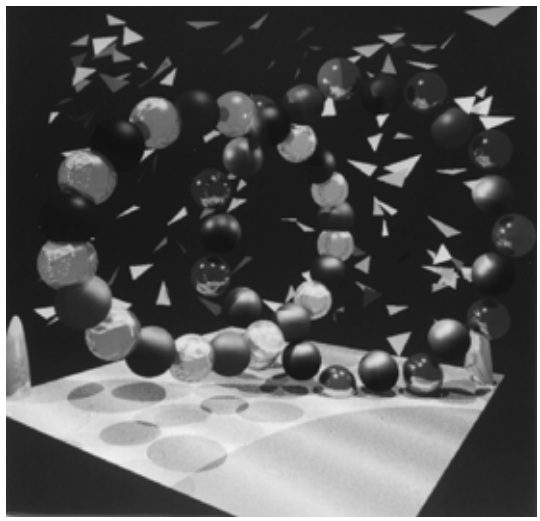
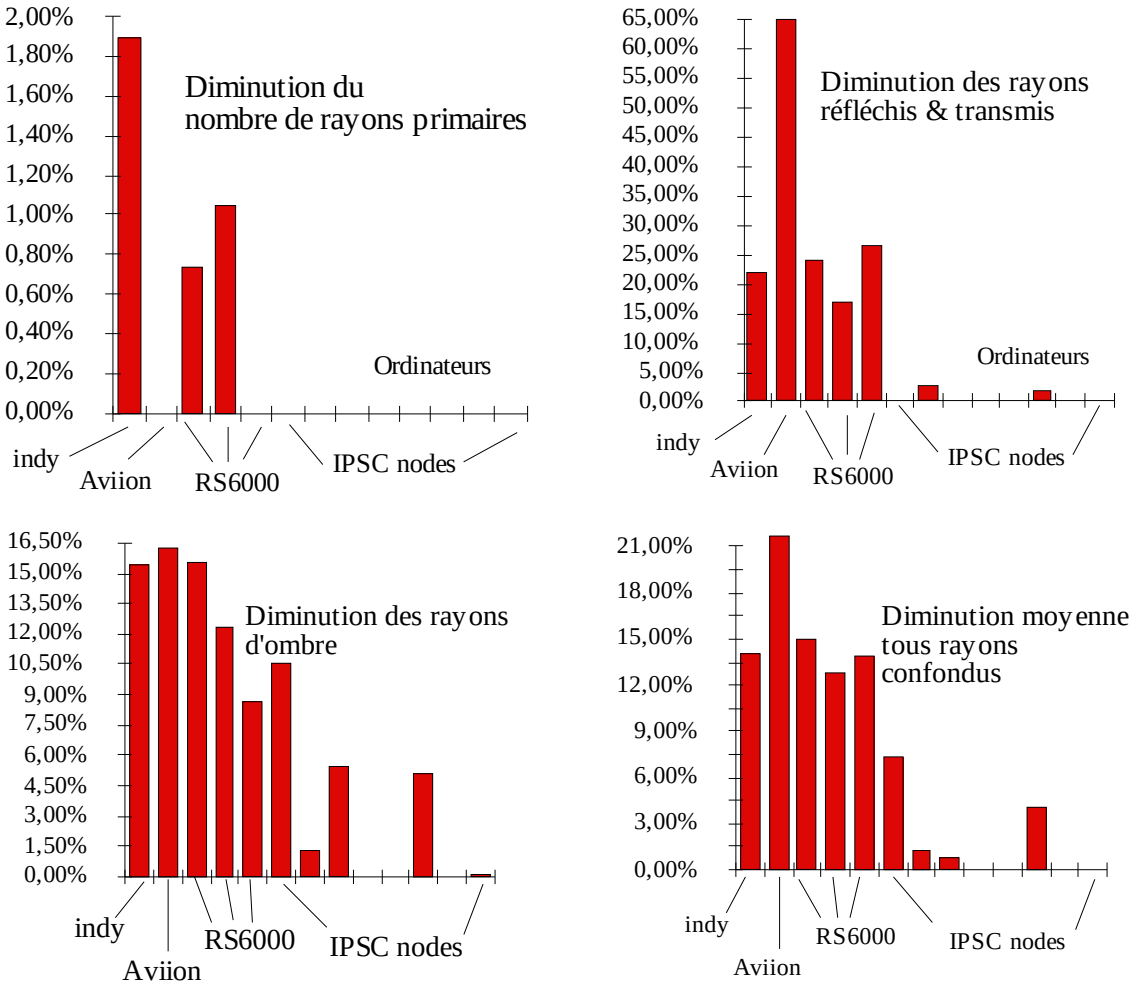


Figure VI-14

III.2.b) Nombre d'objets testés par type de rayon

Dans cette série de tests, on a compté le nombre de rayons (primaire, secondaire, d'ombre) lancés par le programme sans message, puis avec messages échangés (la scène n'étant pas close, on lance plus de rayons qu'il n'y a de rayons utiles). Les communications entre machines se font par Ethernet alors que celles entre les noeuds de l'IPSC/2 utilisent la librairie proposée par INTEL. Naturellement, une diminution du nombre de rayons lancés est un bon indice de performance.

Les courbes suivantes montrent l'effet de l'activation des messages entre machine :



Figures VI-15a, b,c,d : Gains dus aux messages en parallélisme vrai

Ces résultats montrent clairement que les messages échangés (par paquets de 1 à 13 messages) entre machines ont effectivement une action sur les performances des optimisations. Le bénéfice engendré peut varier selon le contenu de la zone écran traitée par un processeur, la puissance de calcul de ce processeur ainsi que la rapidité des communications (les messages ne doivent pas arriver trop tard).

III.2.c) Comparaison en terme de temps de calcul

Prendre les temps en parallélisme vrai n'est pas simple. Deux options pouvaient être prises :

1) prendre le temps sur chaque machine et considérer que le temps d'exécution était le maximum de ces temps,

2) prendre le temps de calcul de la machine qui est chargée de l'affichage de l'image finale augmenté du temps d'attente pour avoir le dernier morceau d'image.

La deuxième option a été choisie même si elle majore le temps réel de calcul (avec un protocole de communication adéquat, le transfert et l'affichage de l'image par exemple peuvent se faire quasiment en parallèle au calcul, ce que nous ne faisons pas ici).

Résultats	Avec messages internes mais pas externes	Avec messages internes et externes
calculs+attente	577.6 + 68.75 = **646,35 s**	559.4 + 11.07 = **570.47 s**
Messages entre machines	0	720

Les améliorations constatées au paragraphe précédent se répercutent ici en terme de temps (-13,3%). Ce résultat n'était pas a priori évident car les communications entre machines sont beaucoup plus coûteuses que les communications en parallélisme simulé (notre protocole de communication est basé sur des commandes systèmes par "rcp", donc simples mais pas nécessairement très performantes).

III.2.d) Distribution de la scène de la Figure VI-8 sur un réseau de stations

Enfin, un dernier test parallèle a été lancé. La scène de la salle à manger a été distribuée sur un réseau de 14 machines localisées sur deux sites différents :

* site de Marne-La-Vallée :
- 1 RS6000-320
- 1 RS6000-250
- 1 RS6000 40P
- 1 RS6000 3AT
- 1 Data General Aviion-300
- 1 Cray YMP-EL
- 1 Silicon Graphics Onyx
- 6 Silicon Graphics Indy

*site de Besançon
- 1 RS6000-220

Toutes les machines sont reliées par Ethernet (les débits entre Besançon et Marne-la-Vallée étant faibles...). Une partie du réseau de Marne-La-Vallée est composé de fibre optique.

L'intérêt de ce test est de montrer que notre programme fonctionne sur un réseau très hétérogène de machines, malgré les difficultés de liaison et de distance des différentes stations. Ce test a un autre intérêt (pas totalement voulu) : pour des raisons que nous ignorons, l'Onyx a des problèmes de communications avec certaines machines du réseau : en effet, dans certains cas, elle n'est pas en mesure d'envoyer des messages. Or l'image est quand même calculée, ce qui prouve, par

l'exemple, que le programme est bien tolérant aux pannes de communication (le temps de calcul sur cette machine étant simplement dégradé).

Temps total de calcul de la scène jusqu'à l'affichage : **101,9 s**
Temps de calcul de la scène pour la machine chargée de l'affichage : **72,6 s**
Plus court temps de calcul pour le calcul d'un bout de l'image : **39 s**

Les temps sont indicatifs et il n'est pas possible d'en déduire un comportement général car le gain de la parallélisation va dépendre du nombre de processeurs, du nombre et de la répartition des processus, du débit du réseau, du taux d'occupation CPU des stations utilisées (nous n'étions pas seul sur le réseau pour ce test) et naturellement de la scène. Enfin, le réseau étant totalement hétérogène, un calcul de l'accélération n'aurait pas eu de sens.

IV) Comparaison avec d'autres algorithmes

Les graphiques suivants illustrent l'intérêt des tests de comparaisons. Deux algorithmes de lancer de rayon sont comparés :

1) un algorithme basé sur une méthode de Roth améliorée [Pi94] :

- discrétisation de l'écran permettant par projection de créer un sous arbre optimisé pour une portion d'écran donnée (boîtes englobantes 2D),
- optimisation du parcours de cet arbre (fils d'un noeud associés à une boîte 3D),
- rééquilibrage éventuel de l'arbre avant les calculs,
- voxelisation régulière de l'espace,
- adaptation des règles B_2 et B_5 (objet vu en ligne/colonne 1, objet vu en ligne/colonne n puis non vu).

2) une version de notre algorithme [RA94].

Une série de scènes sert de support aux différents tests :

Scène I : calibrage de l'algorithme. La scène 1 est constituée d'un cube opaque dont la projection recouvre totalement l'écran (Cf. Figure VI-16).

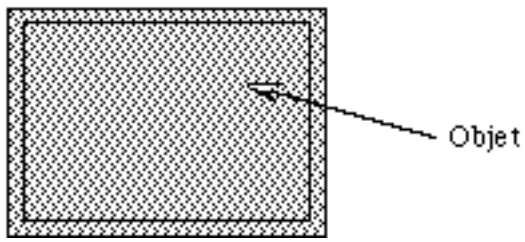


Figure VI-16

Scène II : Scène composée de 100 cubes opaques de même taille. Les cubes forment un mur de "briques" dont la projection recouvre l'écran (Cf. Figure VI-17).

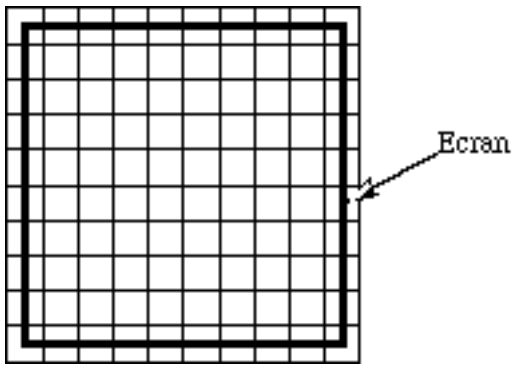


Figure VI-17

Scène III : test de l'illumination. Il y 9 sources invisibles qui éclairent 91 cubes (tous visibles) répartis au hasard.

Scènes IV.1, IV.2 : tests des réflexions. L'écran est recouvert par la projection d'un cube A réfléchissant. À la sous-scène 1 (Cf. Figure VI-18a), on voit un cube B par reflet sur A. Pour la scène 2, on reprend la 1 avec une discrétisation de B en 99 cubes disjoints (Cf. Figure VII-17b).

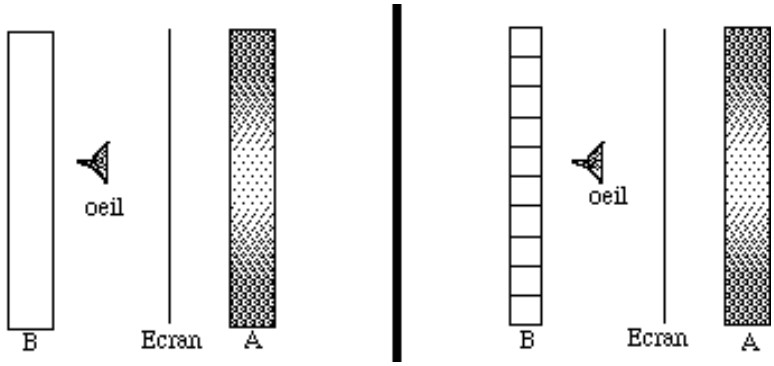


Figure VI-18a Figure VI-18b

Scène V : test des transmissions. L'écran est recouvert par un cube transparent (Indice de réfraction : 1,0). Derrière lui se trouvent 99 petits cubes.

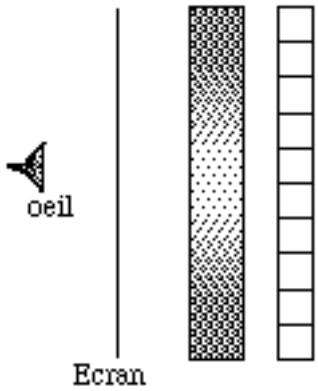


Figure VI-19

Scènes VI : test de complexité. Devant l'écran, on place un pavage de cubes en faisant varier leur nombre : 1, 2, 4, 16, 64, 512.

IV.1) Vérification de la justesse des méthodes

Le nombre de rayons utiles entre 1) et 2) est comparé. La différence étant sur ce jeux d'exemples toujours inférieure à 0,8%, nous en concluons que le protocole est valide et que les erreurs d'aliassage suffisent à expliquer cette différence. Les scènes où il y a beaucoup de réflexions ou de transmissions sont les scènes les plus sujettes aux erreurs d'aliassage.

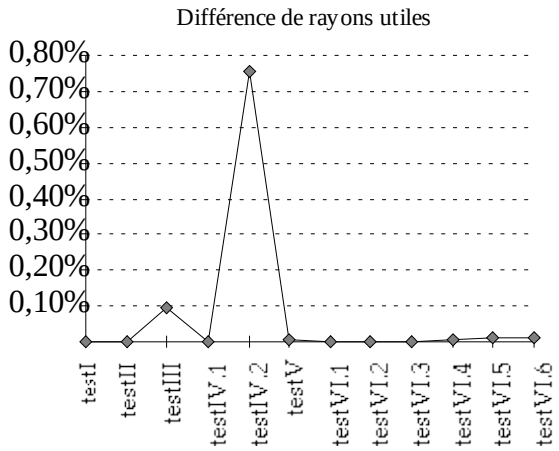


Figure VI-20

IV.2) Objets testés par rayon primaire

Sur les deux courbes de la Figure VI-21 les performances des deux algorithmes sont comparées (par rapport au nombre d'objets testés par rayon primaire). Le programme présenté dans ce mémoire est meilleur que la version améliorée de Roth et les résultats sont la plupart du temps proche de 1, meilleur résultat possible.

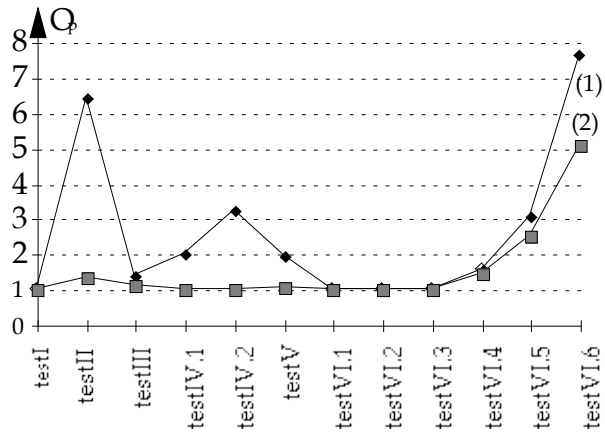


Figure VI-21

IV.3) Pourcentage d'intersections utiles pour les rayons primaires

La Figure VI-22 montre le pourcentage d'intersections utiles pour les deux algorithmes. Ce pour-centage est un bon indice de performance si le coût d'élimination des calculs inutiles n'est pas trop important.

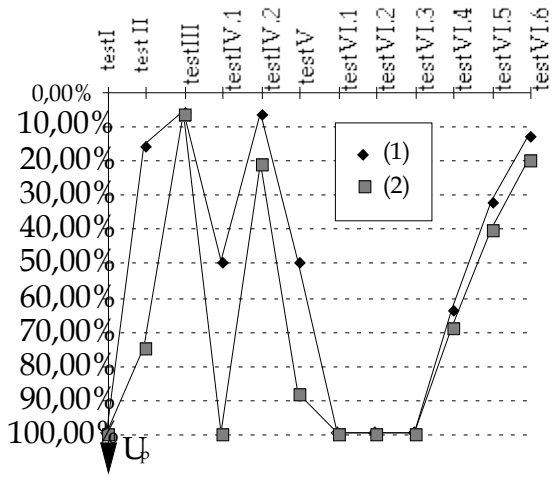


Figure VI-22

IV.4) Nombre moyen d'objets testés tous rayons confondus

On retrouve la hiérarchie des tests précédents lorsque l'on considère tous les rayons. On notera que certains cas sont plus problématiques (lorsque les objets sont derrière la caméra par exemple).

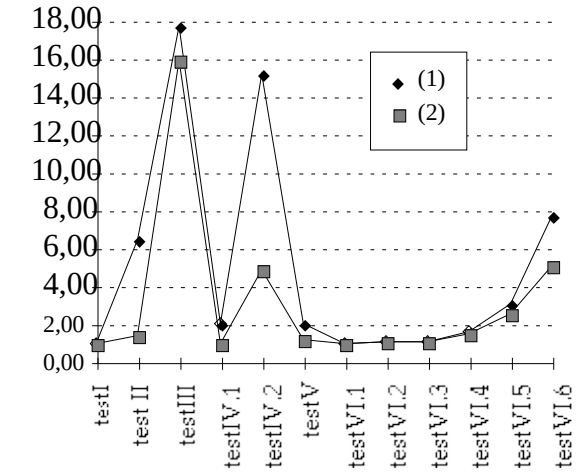


Figure VI-23

IV.5) Temps de calculs

Les temps de calculs reflètent globalement les résultats exprimés par le nombre d'objets testés par rayon (Cf. Figure VI-24). Cependant, des divergences pourront apparaître si le coût des optimisations est très différent ou si l'on tombe dans un cas très favorable pour un algorithme (test III).

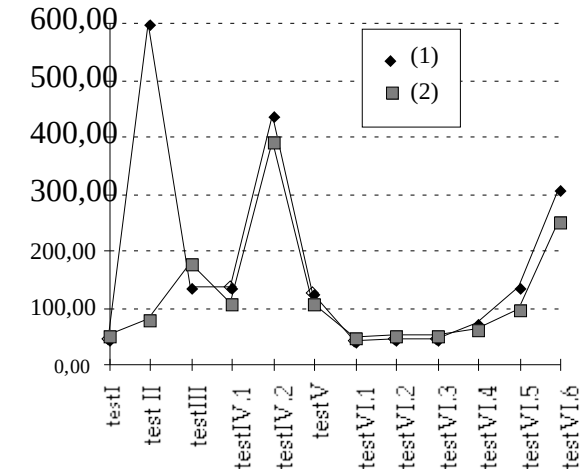


Figure VI-24

V) Conclusion

Dans ce chapitre, un large ensemble de tests a été exposé. Ces tests permettent de conclurent que les idées d'optimisation introduites dans cette thèse sont effectivement efficaces, qu'elles ne faussent pas l'algorithme et que le programme associé obtient de meilleures performances que la version améliorée de l'algorithme de Roth présentée. Ce point n'était pas acquis d'avance car nous étions partis des hypothèses de la généralité du principe et de la portabilité qui auraient pu nuire aux performances du programme. Nous avons même constaté un phénomène intéressant : à plusieurs reprises lors du développement du logiciel, il est apparu que certaines procédures annexes du lancer de rayon (comme le tri de la base de données, la gestion des tableaux booléens, …) devenaient plus coûteuses que les

calculs d'intersections (pour mémoire [Wh80] donnait 87% du temps passé à calculer les intersections et les ombrages). Ces procédures ont toujours pu être améliorées mais le phénomène est symptomatique de la réduction du problème du lancer de rayon lui-même.

Il est également intéressant de constater que cet algorithme fonctionne sur des machines ayant une configuration mémoire et une puissance modeste (Aviion avec 12 Mo de mémoire) aussi bien que sur des machines plus performantes (RS6000 à base de Power2 avec 128 Mo de mémoire ou Cray bi-processeur) voire des machines massivement parallèle (l'hypercube a été cité dans ce mémoire, mais le programme a également été testé sur le KSR1 de l'université de Manchester) ce qui est important toujours par rapport à l'exigence d'un algorithme qui soit le plus largement utilisable possible.

Bibliographie

[Pi94] PIRANDA (B.) - "Optimisation de l'algorithme du lancer de rayon dans le traitement de scènes définies par des arbres CSG". Rapport de DEA, Besançon, 1994, 129 p.

[RA94] RIS (Ph.), ARQUÈS (D.) - "Parallel ray tracing based upon a multilevel topological knowledge acquisition of the scene ", Computer Graphics Forum, Eurographics'94, Oslo, Septembre 1994, p. 221-232.

[Wh80] : WHITTED (T.). - "An improved illumination model for shaded display". - Communication of ACM, 23 (6), 1980, p.343-349.