UNIVERSITE DE LIMOGES
ECOLE DOCTORALE Science – Technologie – Santé

Faculté des Sciences et Techniques
Laboratoire XLIM UMR CNRS 6172

Thèse № [ ]

Thèse
pour obtenir le grade de
Docteur de l'Université de Limoges
Discipline : Informatique
présentée et soutenue par
Nancy EL EZ EDDINE EL DANDACHY
le 20 novembre 2007

Techniques alternatives de visualisation pour la prise de connaissance de scenes tridimensionnelles

Thèse dirigée par
Dimitri PLEMENOS
et co-encadrée par
Bachar EL HASSAN

JURY :
President

Rapporteurs
Monsieur Michel MÉRIAUX, professeur à l'Université de Poitiers
Monsieur Yves DUTHEN, professeur à l'Université de Toulouse
Examinateurs
Monsieur Djamchid GHAZANFARPOUR, professeur à l'Université de Limoges
Monsieur Dimitri PLEMENOS, professeur à l'Université de Limoges
Monsieur Bachar EL HASSAN, maître de conférences à l'Université Libanaise

A Safwan,

A mes parents et mes beaux parents,

Résumé

Le développement rapide du domaine de la synthèse d'image, la diffusion de son domaine dans de nombreuses applications et puis grâce au développement des matériels en vitesse et capacité en mémoire permettant ainsi la visualisation de scènes de hauts niveaux, le problème de la compréhension et de la prise des connaissances des scènes tridimensionnelles devient de plus en plus pertinent et compliqué.

Depuis le milieu des années 70, pratiquement aucune technique de base nouvelle de visualisation n'a vu le jour. Tous les efforts des chercheurs se sont portés sur les possibilités d'amélioration des techniques déjà existantes soit en réduisant les temps de calcul, soit en réduisant l'encombrement de la mémoire, soit encore en inventant des modèles photométriques plus sophistiqués permettant une meilleure qualité des images obtenues. D'autres chercheurs ont concentré leurs études sur la recherche des méthodes de calcul automatiques de bons points de vue ou à faire une animation tout autour de la scène suivant un chemin qui respecte des règles heuristiques évitant les brusques changements déconcertants l'observateur.

Or ces techniques ne sont pas suffisantes pour répondre à toutes les scènes qui peuvent être crées par les matériels actuels. Nous allons donc proposer dans ce mémoire des techniques alternatives basées sur la combinaison des techniques existantes de visualisation afin d'améliorer la compréhension de certaines scènes complexes.

Nous allons tout d'abord étudier le cas des scènes tridimensionnelles complexes qui comportent beaucoup de lumières, des miroirs et des objets transparents, produisant ainsi des effets réalistes qui peuvent créer des illusions dues à la présence des ombres, réflexions et réfractions. La présence de ces effets réalistes peut confondre l'utilisateur et l'empêcher de distinguer entre les objets réels de la scène et tout ce qui est illusoire. Pour améliorer la compréhension de ce type des scènes, nous avons proposé une nouvelle méthode qui combine la technique de visualisation réaliste de lancer de rayon avec l'algorithme économique du raffinement sélectif et la méthode de suivi de contour par le code de direction afin de mettre en évidence les objets réels de la scène en détectant leurs contours apparents dans le but de leurs distinguer de leurs réflexions et réfractions.

Un autre type de scènes sera introduit dans ce mémoire, celui des scènes qui comportent des objets englobant d'autres objets. Trois nouvelles techniques alternatives seront décrites dans ce mémoire afin d'améliorer la visualisation et la prise de connaissance de ce type de scène.

La première tend à visualiser l'objet englobant en mode filiaire alors que l'objet intérieur sera visualisé en mode plein. L'élimination des parties cachées sera réglée par la combinaison de la méthode du z-buffer avec le back facing culling.

La deuxième approche tend à créer un trou sur les faces de l'objet englobant faisant ainsi apparaître l'objet intérieur. Deux méthodes sont proposées afin de réaliser ce but. La première est appliquée seulement aux scènes dans lesquelles l'objet englobant en modélisé par des facettes polygonales et elle tend à éliminer les facettes de l'objet englobant qui cachent l'objet intérieur. La deuxième peut être appliquée à n'importe quelle modélisation de scène et elle tend tout d'abord à visualiser les deux objets: l'objet englobant et l'objet intérieur, puis d'assombrir les pixels proportionnels à la silhouette de l'objet intérieur et orthogonaux dirigés vers l'extérieur de l'objet intérieur.

Mots clés: synthèse d'image, visualisation 3D, prise de connaissance, lancer de rayon et raffinement sélectif, détection de contour.

Abstract

The fast development of the image synthesis domain, the spread of this domain in lot of applications and then because of the development of PCs in speed and memory capacities, the problem of scene understanding and extracting knowledge is becoming more and more pertinent and complicated.

Since the half of the seventies, practically no new basic techniques of visualization were invented. All the researchers' efforts were focused on the possibility of the enhancement of existent techniques whether by reducing the time of computations, or by inventing photometric models more sophisticated allowing the obtaining of better image quality. Other researchers have turned their attention to search for methods that compute automatically a good point of view position or do an automatic animation around the scene following a path that respect heuristic rules in order to avoid brusque changes that might disconcert the observer.

However, these techniques aren't sufficient to resolve the problem of the visualization of all type of scenes created by the PCs so developed nowadays. We are going to propose in this thesis alternative techniques which are based on the combination of existent visualization techniques in order to enhance the understanding of complex scenes.

We are going first to study the case of three-dimensional complex scene that contain lot of lights, mirrors and transparent objects which produce realistic effects that might create illusions due of the presence of shadows, reflections and refractions. The presence of these realistic effects might confuse the observer and prevent him to distinguish between real objects of the scene and illusions. In order to enhance the understanding of this type of scenes, we have proposed a new method that combine between the ray tracing realistic technique of visualization with the selective refinement improvement algorithm and the followed contour technique by the code direction method, in order to underline the real objects of the scene by detecting their apparent contours so that we will be able to distinguish them from their reflexions and refractions.

Another type of scenes will be introduced in this thesis, scenes which contain objects that include other objects. Three new alternative techniques will be described in order to enhance the visualization end the taking knowledge of this type of scene.

The first one leads to visualize the exterior object on wireframe mode while the interior one will be visualized in fill mode. The elimination of hidden surfaces will be regulated by the combination of the z-buffer method with the back facing culling technique.

The second approach leads to create a hole on the surface of the exterior object in order to show the interior one. Two methods will be proposed in order to achieve this project. The first one is applied only for scenes where exterior objects are modeled by a polygonal mesh and leads to eliminate the exterior faces which hide the interior object. The second method can be applied to any scene model and leads first to visualize both objects: the exterior and the interior one, and then make darken the pixels which are proportional and orthogonal to the silhouette of the interior object oriented to the outside of the interior object.

Key words: Image synthesis, 3D visualization, knowledge extraction, raytracing and selective refinement, contour detection.

Remerciements

Cette thèse n'aurait vu le jour sans la confiance, la patience et la générosité de mon directeur de recherche, Monsieur Dimitri PLEMENOS, Professeur à l'Université de Limoges, que je veux vivement remercier. La pleine confiance qu'il m'a accordée et sa croyance dès l'admission en mes capacités et mes pouvoirs, m'ont beaucoup encouragée et permis d'élaborer un plan de thèse personnel et propre à mes aspirations. Je voudrais aussi le remercier pour le temps et la patience qu'il m'a accordés tout au long de ces années, d'avoir été toujours disponible pour d'intenses et rationnelles discussions. De plus, les conseils qu'il m'a divulgué tout au long de la rédaction, ont toujours été clairs et succincts, me facilitant grandement la tâche et me permettant d'aboutir à la production de cette thèse. Je suis en plus reconnaissante pour son soutien et son encouragement sur le plan personnel et professionnel pendant mon séjour en France.

Mes plus sincères remerciements vont également à Dr Bachar EL HASSAN, Maître de conférence à la faculté de génie de l'Université Libanaise Branche I, qui en agissant à titre de co-directeur a fortement enrichi ma formation. Ses conseils et ses commentaires auront été fort utiles.

Merci à M. Djamchid GHAZANFARPOUR, Professeur à l'Université de Limoges et directeur du groupe de recherche SIR dont je fais partie, d'avoir accepté de participer au jury de cette thèse, et aux Professeurs Michel MÉRIAUX de l'Université de Poitiers et Yves DUTHEN de l'Université de Toulouse d'avoir accepté d'être les rapporteurs de ce manuscrit. Je leurs remercie également pour la rapidité avec laquelle ils ont lu mon manuscrit et l'intérêt qu'ils ont porté à mon travail. Leurs remarques et suggestions lors de la lecture de mon rapport m'ont permis d'apporter des améliorations à la qualité de ce dernier.

Je dois un grand merci à mes chers parents et beaux parents. Je ne pourrais jamais oublier leur soutien et leur aide moral qu'ils m'ont offert pendant tous mes études.

Mes plus chaleureux remerciements s'adressent à mon mari Safwan à qui je réserve une reconnaissance particulière pour la patience qu'il m'a accordée pendant la période de la préparation de cette thèse. Je voudrais également souligner sa participation à la discussion entourant les travaux de cette thèse qu'il m'a fournis tout au long de la réalisation de ces travaux et lui remercier pour sa participation à la lecture de cette thèse.

Table des matières

Chapitre 1
Introduction

Ce chapitre comporte tout d'abord une introduction sur l'importance du domaine de la synthèse d'image et de sa diffusion dans plusieurs applications artistiques aussi bien que scientifiques. Nous passons ensuite en revue, les problèmes qui empêchent d'avoir une bonne compréhension des scènes 3D. Nous parlons des solutions qui ont été mises en oeuvre à travers les années afin de résoudre ces problèmes. Finalement nous décrivons notre but de ce travail ainsi que son organisation dans ce mémoire.

I. Introduction

Avec la parole, l'image constitue l'un des moyens les plus importants qu'utilise l'homme pour communiquer avec autrui. C'est un moyen de communication universel dont la richesse du contenu permet aux êtres humains de tout âge et de toute culture de se comprendre. Chacun essaie alors d'analyser et comprendre l'image à sa manière, pour en dégager une impression et d'en extraire des informations précises.

Pour cela, nous remarquons que dès les années 60, les médias, les systèmes de communication et les supports de stockage sont passés de l'analogique au numérique à tous les niveaux de la chaîne de traitement. Et on peut aussi affirmer, sans grand risque de se tromper, que le XXI^{ème} siècle sera, entre autres, celui des images virtuelles. Pour s'en convaincre, il suffit de regarder à quel point notre quotidien a changé ces dernières années.

La part consacrée aux images de synthèse ne fait que croître. Prenons par exemple les bulletins météorologiques télévisés qui sont présentés à l'aide d'animations 3D, les films utilisant massivement des effets spéciaux informatiques, les publicités, … Bien sûr, toutes les industries sont aussi très friandes d'images de synthèse. Et demain, peut-être nous déplacerons nous dans des mondes virtuels pour faire nos achats, aller au travail ou au cinéma …

D'autre part, en plus de la représentation des modèles 3D dans les jeux vidéo et au cinéma, l'emploi de l'infographie pour l'amélioration de la compréhension ou du transfert des connaissances a été également généralisé dans la plupart des disciplines scientifiques et techniques.

Les physiciens, les chimistes ou les biologistes ont souvent des masses très importantes de données à visualiser, manipuler puis analyser afin d'en extraire des informations. L'architecture et l'urbanisme sont bien évidemment concernés mais avec un besoin plus important de réalisme du rendu final.

L'industrie pétrolière s'appuie également sur des techniques de modélisation et de visualisation sismique afin d'optimiser les forages. Le problème de visualisation est là bien spécifique dans la mesure où il ne s'agit pas seulement de percevoir des surfaces mais aussi des volumes.

D'autres secteurs comme l'industrie automobile ou l'aéronautique ont recours à la conception assistée par ordinateur (maquettes virtuelles pour la conception d'objets industriels). Ces outils permettent d'établir différentes versions pour un même projet et ainsi représentent des aides à la prise de décision. Il ne faut pas oublier aussi les travaux qui sont en progression maintenant dans la réalité virtuelle et la réalité augmentée pour l'amélioration de la prise de connaissance et surtout l'aquisition et la transmission des données.

Ce mémoire s'inscrit dans le cadre de l'amélioration de la visualisation et présente nos recherches dans le domaine de l'amélioration de la compréhension des scènes tridimensionnelles qui peuvent être produites à partir des applications citées ci-dessus. Avant de présenter la démarche suivie, il est nécessaire d'aborder certains problèmes, relatifs à la synthèse d'images qui seront le fil directeur de nos travaux.

II. Problématique, raison et finalité du travail

Il est bien clair donc après tout ce que nous venons de citer, que les images de synthèse sont promises à un bel avenir. Mais bien évidemment, il ne faut pas oublier que la diffusion rapide de la synthèse d'image dans tous ces domaines, crée des exigences différentes selon le domaine et les besoins. Ainsi avoir de bonne qualité et de bons résultats qui soient les plus proches possibles de la réalité a certainement un prix. En plus de talent de modélisation dont doit faire preuve le concepteur de ces images, il faut ajouter le temps et la mémoire nécessaires pour les calculer, et surtout l'éventail des techniques de visualisation à offrir par le moteur de rendu afin d'améliorer la compréhension de l'environnement virtuel.

Mais évidemment aussi, plus la qualité, et donc la taille, des images virtuelles croissent vite, plus le recours à la création de nouvelles techniques de visualisation plus performantes devient indispensable afin de répondre aux besoins de l'utilisateur et satisfaire les yeux du public exigeant qui s'est largement familiarisé par cette technologie et demande de plus en plus de facilité, de rapidité et certainement du réalisme.

D'autre part, plus les techniques de visualisation s'améliorent dans leur capacité de visualiser des scènes de plus en plus complexes, plus le problème de la compréhension des scènes devient pertinent et ceci est dû à plusieurs raisons.

Au début, l'un des problèmes les plus répandus était le choix d'une mauvaise position du point de vue ce qui pourrait faire manquer à l'observateur des détails importants ou des informations nécéssaires à la compréhension des scènes tridimensionnelles. Plusieurs techniques [KK88, CC88, PB96, PDB99 et DG01] ont été mises en oeuvre dans ce cadre et les recherches continuent jusqu'à maintenant afin de donner de meilleurs résultats avec le moindre coût possible en temps et mémoire [VFSH02, SP05 et SPT06].

Mais ces techniques n'ont pas pu résoudre le problème des scènes complexes. Même le calcul de plusieurs bons points de vue n'était pas suffisant parce qu'il ne garantit pas le passage d'un point à un autre sans faire des changements brusques qui peuvent confondre l'utilisateur. Plusieurs auteurs ont proposé alors des méthodes d'exploration globale et locale à travers une animation qui traversent des bons points de vue selon un chemin qui suit des règles heuristiques évitant les changements brusques de la caméra. [PDB99, PDB00, DG01, VS03, VP03, JPGT05, JTP06, SP05 et SPT06].

Cependant, ces techniques ne sont pas capables seules de répondre à tous les problèmes qui empêchent la prise de connaissance des scènes 3D utilisées dans différents domaines et applications. Ce qui a amené les chercheurs à découvrir de nouvelles techniques alternatives de visualisation qui aident à améliorer la compréhension de la scène chacune, selon les exigences et les besoins de l'application. Ceci a été dans le but d'étendre l'usage de la visualisation numérique dans divers domaines dans lesquels le rendu photoréaliste est incapable de répondre à leurs besoins. Très récemment, il y a une grande évolution qui tend à utiliser le rendu non photoréaliste (NPR) comme une alternative au rendu photoréaliste afin d'améliorer la visualisation et la compréhension des scènes 3D.

Loin des techniques du rendu NPR qui sont aussi intéressantes et qui tendent à imiter le rendu artistique et produire des images impressionnantes proche de la peinture ou des dessins animés et même aussi de la mosaïque, il y a recourt au rendu NPR dans le domaine de l'image médicale afin d'extraire des relations de corrélations dans les donnés [SOMBK87, HB96, MLM95, MAN96].

Le rendu NPR est aussi utilisé en parallèle avec le rendu photoréaliste pour régler le problème d'antialiassage des images obtenues en rendu réaliste par le lancer de rayon [NG00]. Il est aussi utilisé dans [SLM02] pour l'amélioration et l'accélération de la visualisation des volumes de données multidimensionnelles.

D'autres techniques ont été menées pour répondre aux besoins des gens qui ont des problèmes dans la distinction des couleurs (Daltoniens) par la conversion des images en niveau de gris puis l'attribution d'autres couleurs tout en préservant la qualité visuelle des détails [RGW05].

Aussi, parmi les raisons qui aboutissent à une mauvaise compréhension de la scène est le réalisme lui même dans le but d'avoir une image de la scène la plus proche possible à la réalité. Bien que ceci puisse apparaître paradoxal, trop de réalisme n'est pas toujours souhaitable et ne permet pas d'avoir toujours une bonne compréhension de la réalité. En effet, dans le cas des scènes qui comportent des lumières, miroirs et objets transparents, la présence des ombres, réflexions et réfractions peuvent donner à l'utilisateur l'illusion de la visualisation des objets non existants en réalité provenant des objets réfléchis et réfractés, ce qui peut alors confondre l'utilisateur et l'empêcher de pouvoir distinguer entre ce qui est réel et ce qui est illusoire.

Un autre type de scènes, nous parait être aussi intéressant et est aussi traité et étudié dans ce mémoire. C'est le cas des scènes qui comportent des objets englobant d'autres objets. Aucune des techniques de visualisation existantes ni les tecniques d'explorations globales et locales ne sont capables d'offrir une idée suffisante du contenu de la scène et bien évidement du contenu des objets puisque ces techniques sont basées sur la visualisation des objets les plus proches de la position du point de vue. Comme les objets qui se trouvent à l'intérieur d'autres objets sont toujours masqués par les objets englobant par rapport au point de vue, ils ne seront jamais visibles. Pour cela, nous allons proposer dans ce mémoire de nouvelles techniques de visualisation qui aident à améliorer la compréhension et la prise de connaissance de ce type de scènes.

III. Objectif de ce travail

Le but du travail entrepris dans cette thèse est tout d'abord d'étudier les différentes techniques existantes qui aident à l'amélioration de la compréhension des scènes tridimensionnelles, discuter et évaluer leurs performances, puis mettre en œuvre de nouvelles techniques alternatives de visualisation pour l'amélioration de la prise de connaissance des scènes 3D complexes qui sont difficiles à comprendre à partir des techniques de visualisation existantes.

Notre objectif est de régler le problème de la compréhension de deux types particuliers de scènes complexes. Le premier type est celui de scènes comportant beaucoup d'effets réalistes provenant de la présence de miroirs, lumières et objets transparents. Ces derniers confondent l'observateur et l'empêchent de distinguer entre les objets réels de la scène et les objets créés par les réflexions et les réfractions. Nous proposons de résoudre ce problème en utilisant une nouvelle méthode qui utilise le rendu non photoréaliste avec le rendu photoréaliste afin de mettre en valeur les objets réels de la scène.

Le deuxième type de scènes complexes auquel nous sommes intéressés, est celui des scènes comportant des objets englobants d'autres objets, empêchant ainsi la visibilité des objets intérieurs et la possibilité d'avoir une idée suffisante du contenu de la scène.

IV. Organisation du mémoire

Afin d'atteindre notre objectif qui entre de le cadre de l'amélioration de la compréhension des scènes complexes par des techniques alternatives de visualisation, nous proposons quatres méthodes. Chacune est appliquée à une application bien spécifique.

Avant de les détailler nous allons présenter tout d'abord dans le chapitre 2, une description brève des techniques de visualisation existantes qui peuvent être classées entre méthodes classiques et méthodes réalistes.

Nous allons ensuite parler dans le chapitre 3, des techniques qui aident à avoir une meilleure prise de connaissance à travers une exploration automatique tout autour de la scène, soit en calculant un bon point de vue, soit en faisant une animation tout autour de la scène par une caméra virtuelle.

Nous abordons ensuite dans le chapitre 4, le cas des scènes complexes comportant des miroirs, beaucoup de lumières et d'objets transparents créant ainsi des illusions empêchant l'utilisateur d'avoir une bonne compréhension de la scène. Notre contribution dans l'amélioration de la compréhension de ce type de scène sera décrite dans ce chapitre. Elle est basée sur la détection des contours apparents des objets réels de la scène par la méthode de lancer de rayon avec raffinement sélectif combiné avec le suivi des contours par la méthode de code de direction. Comme nous allons utiliser une technique de détection des contours, une représentation rapide des

techniques existantes dans ce domaine sera décrite au début de ce chapitre.

Nous décrivons dans le chapitre 5, notre contribution dans la création de nouvelles techniques de visualisation basées sur la combinaison des techniques existantes dans le but de régler le problème de la visualisation des scènes comportant des objets englobant d'autres objets. Il s'agit en effet d'une nouvelle technique alternative qui tend à visualiser les contours apparents de l'objet englobant puis l'objet intérieur en mode plein tout en éliminant les parties cachées par la méthode du z-buffer combinée par la technique du «back facing culling».

Afin d'avoir une idée globale de la scène ainsi qu'être capable d'explorer l'intérieur des objets englobants, il est possible de créer un trou sur l'objet englobant faisant apparaître son intérieur. Deux nouvelles techniques aussi décrites dans le chapitre 5 et proposées afin d'accomplir ce but. La première consiste à éliminer les facettes de l'objet englobant qui cachent l'objet se trouvant à l'intérieur, alors que la deuxième tend à créer un trou proportionnel et orthogonal à la silhouette de l'objet caché selon une profondeur convenable.

Finalement, une présentation critique du travail ainsi qu'une conclusion et perspective seront élaborées dans le chapitre 6 avant de présenter à la fin de ce mémoire la bibliographie sur laquelle a été basée notre recherche et la liste des figures utilisées dans ce rapport.

Chapitre 2
Techniques de visualisation

Ce chapitre comportera une description générale des techniques de visualisation classique qui sont basées sur le mode en fil de fer et le mode plein avec l'élimination des parties cachées. Nous allons tout particulièrement, traiter en détail le problème de l'élimination des parties cachées vu son importance dans l'amélioration de la compréhension et de la prise de connaissance des scènes 3D. Nous allons ensuite donner une description rapide et brève sur la visualisation réaliste qui est basée sur le lancer de rayons. Finalement, les problèmes et les limites de ces techniques seront abordés et discutés à la fin de ce chapitre afin de proposer des améliorations et des solutions aux cas où les techniques existantes sont incapables de donner une bonne prise de connaissance.

I. Introduction

Durant les premières années du développement de la synthèse d'images, l'effort a été porté essentiellement sur les algorithmes de visualisation. Ainsi la plupart des techniques connues et utilisées actuellement ont leurs origines dans les algorithmes des années 60 et 70.

Depuis le milieu des années 70, à part de la radiosité, pratiquement aucune technique de base nouvelle n'a vu le jour et les efforts des chercheurs se sont portés sur les possibilités d'amélioration des techniques déjà existantes soit en réduisant les temps de calcul, soit en réduisant l'encombrement de la mémoire, soit encore en inventant des modèles photométriques plus sophistiqués permettant une meilleure qualité des images obtenues.

Il n'est pas dans nos intentions de faire une présentation exhaustive de tous les travaux existants dans le domaine de la visualisation, mais plutôt d'en présenter les concepts de base avant d'introduire dans les prochains chapitres de nouvelles techniques alternatives de visualisation qui aident à avoir une meilleure prise de connaissance de certaines scènes difficiles à comprendre avec les techniques existantes.

II. Visualisation Classique

La visualisation classique peut être en mode fil de fer dans laquelle, seulement les contours des objets ou des facettes sont affichés dans la scène *(Fig. 2.1(a))*, ou en mode plein avec une élimination des parties cachées dans laquelle il s'agit de la visualisation de tous les objets ou les facettes jugés les plus proches à la position d'un point de vue donné *(Fig. 2.1 (b))*.

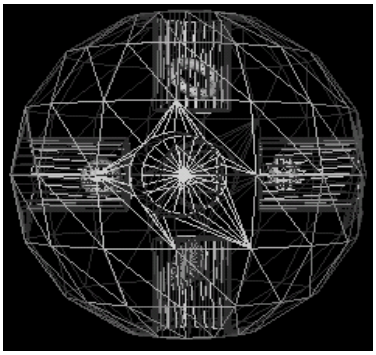


Fig. 2.1(a): Visualisation en mode fil de fer.

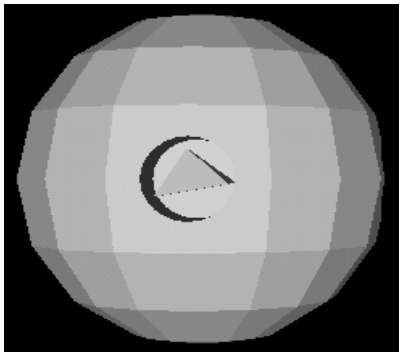


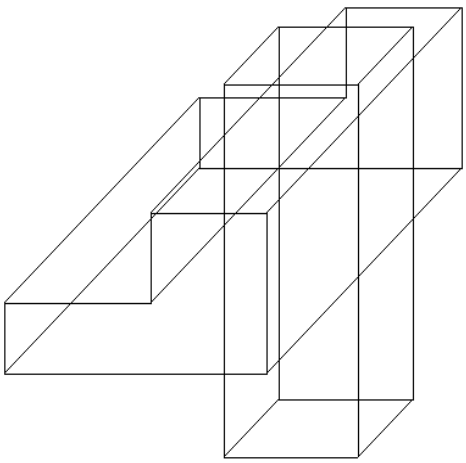
Fig. 2.1(b)Visualisation en mode plein avec une élimination des parties cachées.

II.1 Élimination des parties cachées

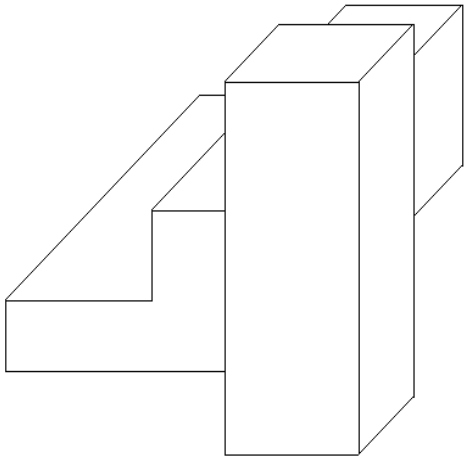
II.1.1. Introduction

Une scène 3D est généralement composée de plusieurs objets dont certains peuvent être en partie ou complètement cachés par d'autres. Aucun objet n'est vu dans sa totalité à cause de l'existence de faces arrières qui sont invisibles par l'oeil de l'observateur. Il faut donc, afin d'obtenir une scène réaliste, ne pas les afficher à l'écran.

Ajoutons aussi le fait que c'est en tenant compte de l'opacité des objets et de leurs positions relatives les uns par rapport aux autres que l'on peut imiter la notion de profondeur *(Fig. 2.2)*. Il ne faut pas oublier surtout l'ambiguïté qui peut être levé en supprimant les lignes ou les surfaces qui ne sont pas visibles de l'un ou l'autre point de vue *(Fig. 2.3)*.

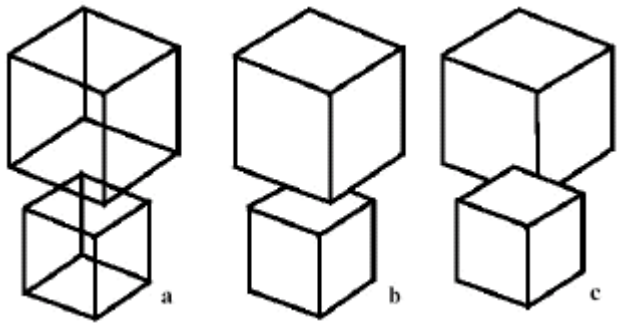


(a)Visualisation de tous les contours.



(b) Visualisation avec une élimination des parties cachées.

Fig. 2.2: Illustration du besoin d'éliminer des parties cachées pour avoir une notion de profondeur.



Les figures b et c peuvent tous deux être représentés par a.

Fig. 2.3: Ambiguïté pour interpréter la figure a.

II.1.2. Les approches « Objet » et les approches « Image »

Le problème de la suppression des parties cachées est un problème fondamental en synthèse d'images et considéré comme étant l'un des plus difficiles de l'infographie.

Les premiers algorithmes ont été développés dès les années 60. Ce traitement est appelé indifféremment élimination des parties cachées ou détermination des surfaces visibles. Le but de ces algorithmes est de déterminer les lignes, les côtés, les surfaces ou les volumes qui sont visibles pour un observateur situé en un point donné de l'espace, ou qui lui sont visibles. Les principales caractéristiques qui serviront à évaluer les différents algorithmes sont leur rapidité d'exécution, leur gourmandise en mémoire et leur indépendance par rapport au matériel. Ajoutons aussi la facilité d'utilisation et la qualité des résultats obtenus.

La complexité de ce problème a suscité l'éclosion d'un grand nombre de solutions diverses et qui peuvent être classées selon divers critères. Très grossièrement, nous pouvons les répartir en deux classes [SIR74]: les approches à précision objet et les approches à précision image. Dans la première, nous déterminons pour chaque objet, l'ensemble de ses éléments qui ne sont pas occultés par des éléments du même objet ou d'un autre objet. Dans la seconde, nous regardons pour chaque pixel, l'objet auquel il appartient et qui est le plus proche du centre de projection tout en étant dans le volume de vue de l'image.

Ces deux approches peuvent être représentées par le pseudo code suivant:

Pour (chaque objet de la scène)
{
Déterminer les éléments de l'objet non occultés par d'autres éléments du même objet ou d'un autre objet

Afficher ces éléments avec la couleur appropriée
}

Pour (chaque pixel de l'image)
{
Déterminer parmi les objets traversés par le rayon visuel relatif à ce pixel, l'objet le plus proche de l'observateur

Afficher le pixel avec la couleur appropriée
}

Les algorithmes de l'espace objet sont implémentés dans le système des coordonnées physique dans lequel les objets sont décrits. Théoriquement, lorsque le nombre n des primitives à traiter n'est pas très grand, ils nécessitent moins de calcul que les algorithmes de l'espace image. En effet, la complexité en temps de calcul pour les approches à précision objet est en $O(n^2)$, alors que celles de la précision de l'écran est en $O(n.p)$ où p le nombre total de pixels qui est généralement très grand. Parcontre lorsque n devient très grand (n>p), le travail dans l'espace objet devient beaucoup plus coûteux.

Mais pratiquement, les algorithmes et les structures de données associées aux algorithmes de l'espace objet sont souvent loin d'être simples contrairement aux algorithmes de l'image qui sont implantés dans le système des coordonnées de l'écran sur lequel les objets sont visualisés. Ceci implique que tout pixel de l'image a une vue « globale » de la scène, ce qui permet donc de traiter plus facilement les phénomènes de transparence et d'aliassage.

Les points faibles des algorithmes de l'image résident dans la nécessite de connaître toutes les primitives avant de pouvoir calculer un pixel et de devoir traiter tous les pixels de l'image, y compris ceux qui ne participent à la représentation d'aucune primitive. D'autre part, cette approche dépend en partie du matériel puisqu'elle dépend du nombre de pixels qui vont pouvoir être affichés. Enfin, la visibilité doit être recalculée à chaque changement de résolution.

Malgré tous les points faibles que nous avons cités sur les algorithmes de l'espace images, en pratique, ils sont plus efficaces, car il est plus facile d'utiliser la cohérence dans l'implantation en balayage de trame d'un algorithme de l'espace image.

Parmi les algorithmes orientés objets les plus connus il est important de citer l'algorithmes de ROBERTS qui a été la première solution connue au problème de la suppression des parties cachées [ROB63, TIP64 et PM77]. Il y a aussi les algorithmes de tri comme l'algorithme du peintre et celui de NEWELL [NNS72], les algorithmes du BSP trees et l'algorithme de tri par profondeur de WEILER- ATHERTON [WA77] qui est basés sur le tri des surfaces des polygones .

Dans le cadre des algorithmes orientés image, il existe trois grandes familles d'algorithmes de visualisation avec élimination des parties cachées [BPP86]:

- Les algorithmes de visualisation par **balayage ligne par ligne** ou de **scanline**, où la scène est découpée en des plans parallèles équidistants et le problème de la visibilité est résolu en deux dimensions pour chaque plan. L'algorithme de WATKINS [WAT70] a été le plus célèbre opérant sur des scènes composées de facettes. En 1980, MAHL a appliqué cet algorithme pour des scènes composés de quadriques [MAH72]. Une variante de cette algorithme a été faite afin de l'appliquer à des scènes composées de surfaces paramétriques [LC79 et LCWB80] .

- Les algorithmes de visualisation par **subdivision** de l'espace image en des sous espaces jusqu'à l'obtention d'une situation simple permettant de résoudre facilement le problème de la visibilité. L'algorithme de WARNOCK [WAR69] a été le premier algorithme utilisant une méthode de subdivision récursive de l'espace image. Une autre variante de cet algorithme a été appliquée dans [CE74, CE75, GRI75 et GRI78] qui subdivise chaque surface pour l'affichage de surfaces gauches.

- Les algorithmes **à liste de priorités ou par classement en profondeur** dont l'objectif est d'effectuer un tri afin d'obtenir une liste définitive des éléments de la scène par ordre de priorité de leur distance, ou profondeur comptée à partir du point de vue. L'algorithme du Z-BUFFER a été considéré comme étant le plus simple et le plus utilisé parmi les algorithmes à priorité. Il a été initialement proposé par CATMULL dans [CE74].

Il y a certainement d'autres algorithmes qui sont aussi intéressants et peuvent donner de bons résultats. Certains auteurs ajoutent aussi une troisième catégorie d'algorithmes, à savoir des algorithmes hybrides qui réalisent un prétraitement dans l'espace objet puis travaillent dans l'espace image. Plus de détails sont décrits dans [PLE98, ROG97, FVH90, FVHP94, et SIR74].

II.1.3. Le « Backfacing culling »

La plupart des algorithmes de l'élimination des parties cachées nécessitent des tris [SIR74], ce qui les rend généralement lents, et ont besoins souvent de plusieurs minutes, voire plusieurs heures de calculs. L'idée principale de ces tris est basée sur le calcul de la distance géométrique entre un volume, une surface, un côté ou un point, et le point de vue. Plus un objet est éloigné du point de vue, plus il y a des chances qu'il soit totalement ou partiellement occulté par un autre objet plus rapproché du point de vue.

Pour optimiser le temps et la mémoire, il existe la technique du test de « backfacing culling » qui est généralement utilisée comme une étape préliminaire à un certain nombre d'algorithmes d'élimination des parties cachées d'une scène. Elle permet de déterminer pour chaque objet les faces qui seront visibles par l'observateur et d'éliminer celles qui ne le seront pas.

Très souvent cette technique peut être applicable seule pour des scènes définies par des faces polygonales et elle est généralement suffisante pour éliminer les parties cachées si:

1. l'objet est seul dans la scène (il ne faut pas qu'un objet puisse en masquer un autre).

2. l'objet est convexe (dans un objet concave, des faces avant peuvent être masquées par d'autres).

Cette technique est rapide et donne de bons résultats dans le cas de scène comportant des objets convexes bien orientables, ce qui est le cas des scènes avec lesquelles nous travaillons. Nous allons essayer dans le chapitre 5 d'étendre cette méthode et la combiner avec d'autres algorithmes de visualisation afin de régler le problème de l'élimination des parties cachées des certaines scènes particulières avec lesquelles les techniques existantes sont incapable seules de donner une bonne visualisation.

En 3D, une facette (un polygone) possède deux « faces »: une face directe et une face indirecte. L'orientation est fixée par l'ordre des sommets de la facette.

- La face directe correspond à la succession des sommets dans le sens direct (trigonométrique).

- La face indirecte correspond à la succession des sommets dans le sens indirect.

Par convention, en regardant un objet de l'extérieur, les polygones dont les sommets sont, par rapport à un certain point de vue, orientés dans le sens contraire des aiguilles d'une montre (CCW) (sens directe), sont dits faces frontales (front facing). Tous les polygones dont les sommets sont orientés dans le sens d'une aiguille d'une montre (CW) (sens indirecte) sont alors dits faces arrières (back facing) (*Fig.2.4*).

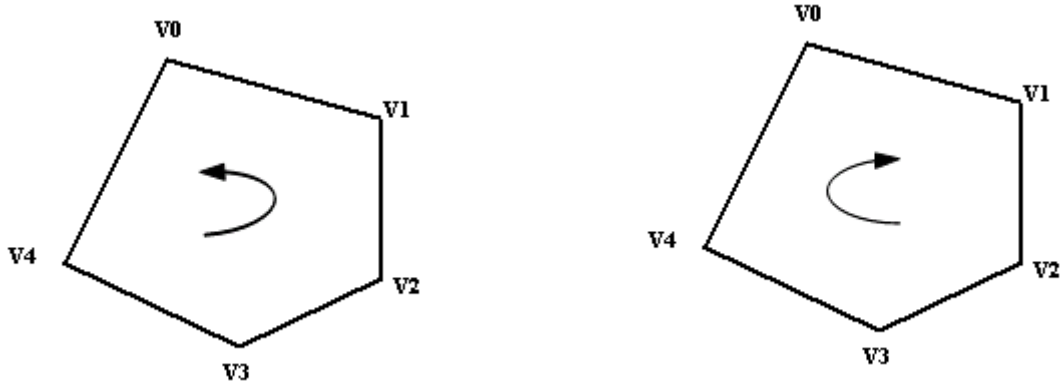


Fig. 2.4: (a) Orientation du polygone dans le sens direct: Face frontale. (b) Orientation du polygone dans le sens indirect: Face arrière..

Les contours de l'objet englobant qui doivent être visibles sont ceux des polygones frontaux par rapport à la position du point de vue. Autrement dit, ce sont les contours des polygones dont la normale pointe dans la direction opposée au point d'observation. Ces parties, vues du point d'observation, sont en effet occultées par l'objet lui-même. (*Fig. 2.5*).

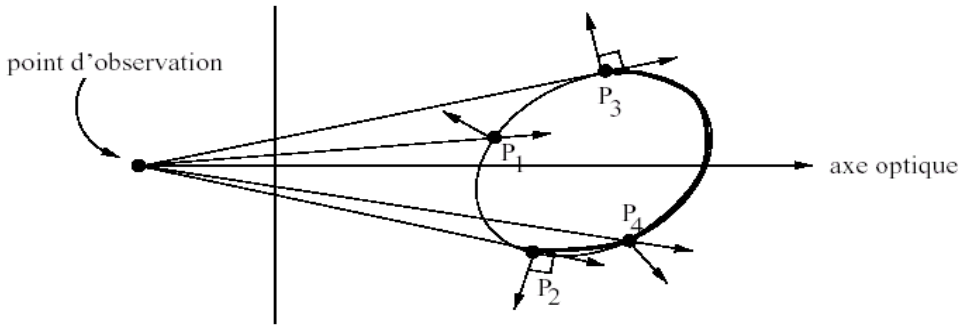


Fig. 2.5: P1 visible, P2 et P3 sont à la limite de visibilité, P4 n'est pas visible..

Ce traitement se fonde sur une observation simple : si nous connaissons la normale d'une face *n*, et la direction de vue *v* de l'observateur, il est possible de déterminer si cette face sera vue. En effet, toute face dont la normale pointe dans une direction « opposée » à la direction de vue sera occultée par l'objet lui-même. De manière plus mathématique, si la normale à une face fait un angle de moins de 90° avec la direction de vue alors la face est dissimulée. Pour détecter une telle configuration, il suffit de réaliser le produit scalaire entre *n* et *v* et de regarder son signe.

En effet, $n.v = ||n||.||v||.cos(\theta)$ où θ mesure l'angle entre *n* et *v*.

Le signe de cos(θ) détermine le signe du produit scalaire. Si cet angle est supérieur ou égal a 90° alors cos(θ) est négatif ou nul, sinon il est positif. Un produit scalaire négatif ou nul signifie que nous voyons la face, tandis qu'un produit scalaire positif indique qu'elle est occultée.

Appliquer cet algorithme implique que nous avons défini une orientation cohérente sur les faces d'un objet qui permet de définir clairement intérieur et extérieur : la normale à une face doit pointer vers l'extérieur de l'objet ! La normale à une face peut alors être calculée à l'aide du produit vectoriel de 2 arêtes consécutives de la face.

Voyons un peu le cas d'objets définis à l'aide de facettes triangulaires. Les notations sont celles de la figure *Fig. 2.6*. Le vecteur normal est obtenu par le produit vectoriel:

$$\mathbf{N} = \mathbf{P_1P_2} \wedge \mathbf{P_1P_3}$$

Le produit scalaire dont le signe va déterminer si la face est visible ou non est le produit suivant :

$$V_1 \cdot N = V_1 \cdot (P_1P_2 \wedge P_1P_3)$$

Nous pouvons aussi l'exprimer à l'aide des vecteurs V1, V2 et V3 :

$$V_1 \cdot N = V_1 \cdot ((V_2 - V_1) \wedge (V_3 - V_1))$$

D'où:

$$V_1 \cdot N = V_1 \cdot (V_2 \wedge V_3)$$

La face sera alors visible si et seulement si le produit mixte précédent est négatif ou nul.

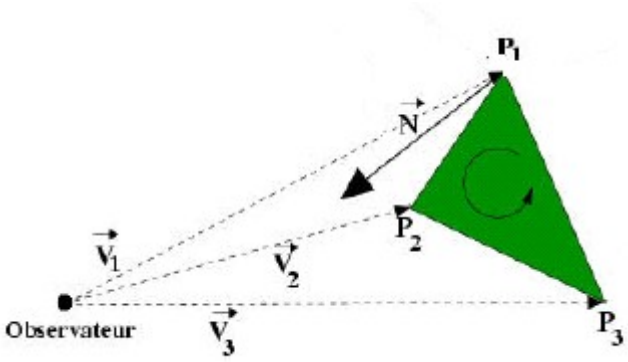


Fig. 2.6: Calcul de la normale d'une facette

La notion de frontale/arrière peut aussi être appliquée à des objets non polygonaux en associant cette notion à chaque point d'un objet. En effet, dans le cas d'une description polygonale, un point *P* est dit frontal s'il est un élément d'une facette frontale c'est à dire si le produit scalaire $N.V < 0$ où *N* est la normale à la facette au point *P* et *V* est le vecteur de vue (*V=P-O*), (ou bien $N.V > 0$ si nous considérons la direction de vue dans le sens contraire en prenant *V=O-P)*.

Pour une description non polygonale, il suffit donc de voir si chaque point est frontal ou arrière en testant le signe de *N.V* (*Fig. 2.7* et *Fig. 2.8*).

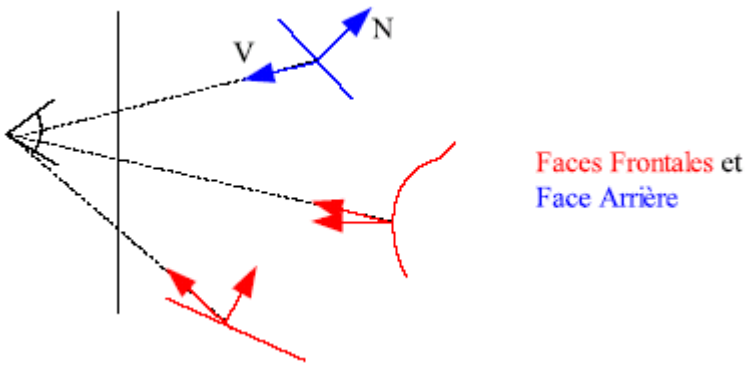


Fig. 2.7:Des points frontaux et des points arrières.

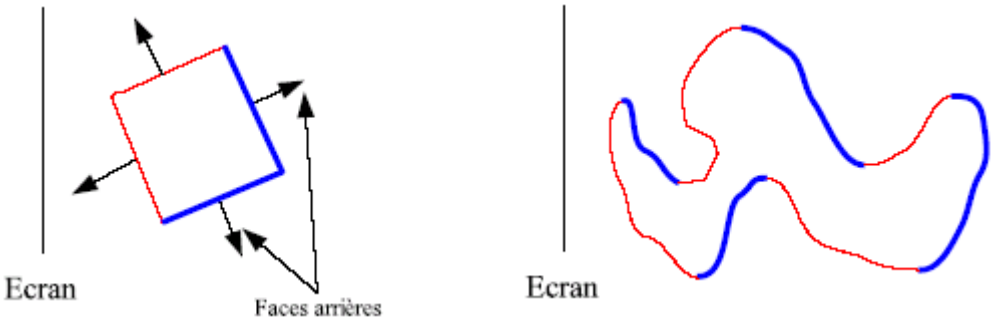


Fig. 2.8: Des parties frontales et des parties arrières.

Pratiquement il est plus facile de travailler dans le système des coordonnées fenêtres que de travailler en coordonné universel. Et comme l'orientation d'un polygone ne change pas si nous le projetons sur l'espace fenêtre, il suffit d'étudier l'orientation des projections des polygones sur l'espace fenêtre.

Pour une facette convexe de sommet (V1,V2,V3, ...), dans OpenGL [WNDS99], la décision si la face est frontale ou arrière dépend du signe de la valeur *A* du polygone calculé en coordonnées fenêtres par la formule suivante:

$$A = \sum_{i=0}^{n-1} x_i y_{i\oplus 1} - x_{i\oplus 1} y_i$$

où:

- $i \oplus 1$ est (i+1) mod n
- n est le nombre de sommets dans le polygone
- x_i et y_i sont les coordonnées fenêtres du sommet indice i du polygone

Pour un objet vu de l'extérieur, la valeur $A>0$ correspond aux polygones qui sont front facing alors que $A<0$ correspond aux polygones back facing. En effet, soit le tripler (u,v,n), le repère de coordonnées lié à l'espace fenêtre, et soit un polygone projeté sur l'espace image de sommets (P_1, P_2, P_3) de coordonnées respectifs (x_1, y_1, z), (x_2, y_2, z) et (x_3, y_3, z) où z est la distance entre l'oeil et le plan de projection $(z>0)$ (*Fig. 2.9*).

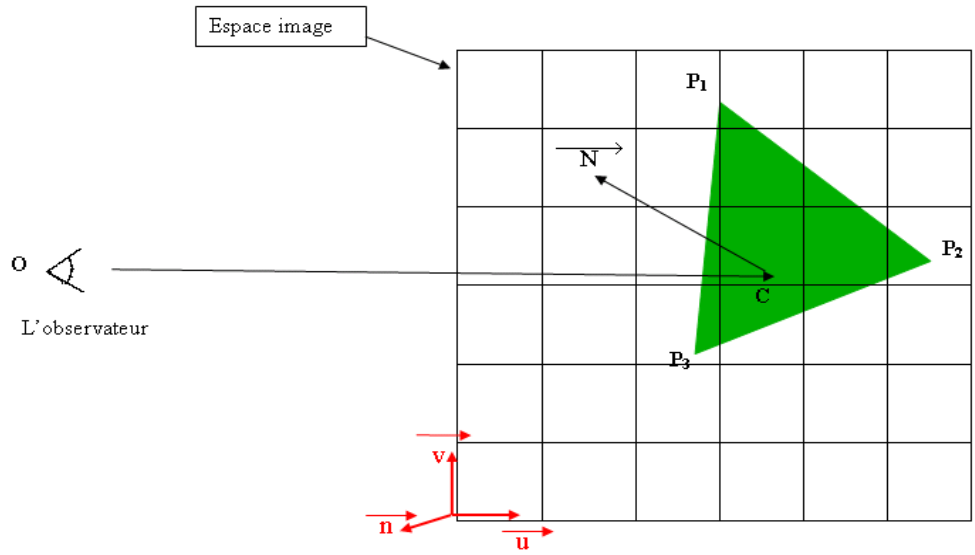


Fig. 2.9: Le polygone $P_1P_2P_3$ projeté sur l'écran et de normale N.

La normale à la facette est la même en tout point du plan de projection, en particulier en *C* qui est le point d'intersection entre l'axe optique et le plan de projection.

L'expression du vecteur *OC* dans le repère (u,v,n) n'est autre que :

$$OC = (-z) \cdot n$$

N est obtenu par la formule suivante:

$$N = P_1P_2 \wedge P_1P_3$$

Il est facile de vérifier que :

$$N = \left(\sum_{i=1}^{3} x_i y_{i\oplus 1} - x_{i\oplus 1} y_i \right) . n$$

La face est frontale si et seulement si *N.OC <0* c'est à dire

$$-z . \left(\sum_{i=1}^{3} x_i y_{i\oplus 1} - x_{i\oplus 1} y_i \right) < 0$$

Ce qui explique le fait que la face est frontale si et seulement si :

$$A = \left(\sum_{i=1}^{3} x_i y_{i\oplus 1} - x_{i\oplus 1} y_i \right) > 0$$

III. Visualisation réaliste

La visualisation réaliste des scènes tridimensionnelles comporte non seulement une visualisation en mode plein avec une élimination des parties cachées mais elle prend aussi en considération des effets de lumières afin de délivrer des solutions très proches de la réalité en introduisant des ombres, des effets de miroirs ou de transparences et de textures, des réflexions et des réfractions. Ces algorithmes sont plus lents et demandent souvent plusieurs minutes, même parfois plusieurs heures de calcul.

Les algorithmes du lancer de rayon , le lancer de photon et de la radiosité sont basées sur la visualisation réaliste. Nous n'allons pas aborder dans ce chapitre les principes de la radiosité, ni ceux du lancer de photon parce qu'ils n'entrent pas dans les objectifs de ce travail.

III.1. Le lancer de rayons

L'algorithme de lancer de rayon, si populaire aujourd'hui, date de 1968 et il a été initialement suggéré par Appel [APP68]. Sa première mise en œuvre remonte à 1971, dans le logiciel de visualisation tridimensionnelle MAGI. Il permet de calculer la visibilité des objets en même temps que leur illumination. Il est capable de gérer les ombres, les transparences, les plaquages de textures et les interactions entre les objets. En outre, il est adapté à n'importe quel type de primitives graphiques. Cette amélioration a bien sûr un coût : les temps de calculs sont bien plus importants que pour les algorithmes vus précédemment.

Le principe de l'algorithme est le suivant:

- On considère un faisceau de rayons imaginaires reliant l'œil de l'observateur au centre de chaque carreau élémentaire (pixel) de l'espace image.

- Pour chaque rayon, ses intersections avec toutes les surfaces de la scène sont calculées, afin de déterminer le point d'intersection le plus proche de l'observateur.

- L'intensité lumineuse de ce point d'intersection est affectée au pixel correspondant.

Une fois que le point visible par l'observateur est déterminé, il faut traiter le problème des ombres portées en lançant un rayon du point visible vers la source de lumière. Si ce rayon coupe une surface avant le point visible, le point visible sera alors considéré dans l'ombre .

KAY et GREENBERG [KAG79] ont proposé une extension de l'algorithme de lancer de rayons permettant la prise en compte de la réfraction du rayon lorsqu'il traverse des surfaces transparentes.

WHITTED [WHI80] a proposé la décomposition de l'intensité lumineuse d'un point en une composante de réflection spéculaire S et une composante de transmission T. Cette décomposition donne un arbre binaire que l'algorithme doit parcourir pour chaque rayon lancé (*Fig. 2.10*). Le suivi des rayons visuels s'effectue dans le sens inverse de la lumière, de l'oeil vers la scène, d'ou le nom de "backward raytracing".

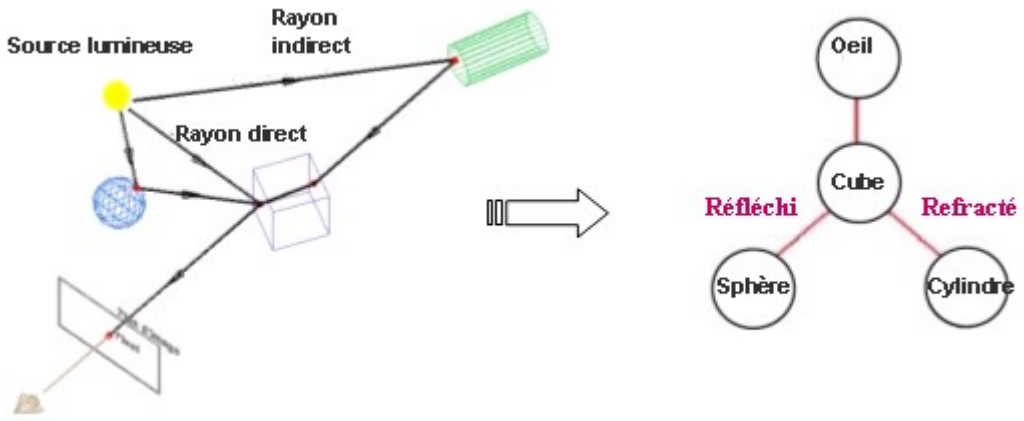


Fig. 2.10 : Trajectoire des rayons dans l'algorithme du lancer du rayon.

D'autres améliorations ont été portés à l'algorithme du lancer de rayon dont le but est de réduire le nombre ou la longueure de rayons lancés et le temps de calcul tout en préservant une qualité acceptable des images obtenues:

- Les techniques du volume englobant qui reviennent à regrouper de façon cohérente les objets de la scène puis à ne lancer que les rayons qui coupent les boîtes englobantes [WHI80, RW80, MULL86 et KK86]. ROTH en 1982 [RO82] a utilisé l'optimisation par des boîtes englobantes pour les arbres CSG (Constructive Solid Geometry) (*Fig. 2.11*).

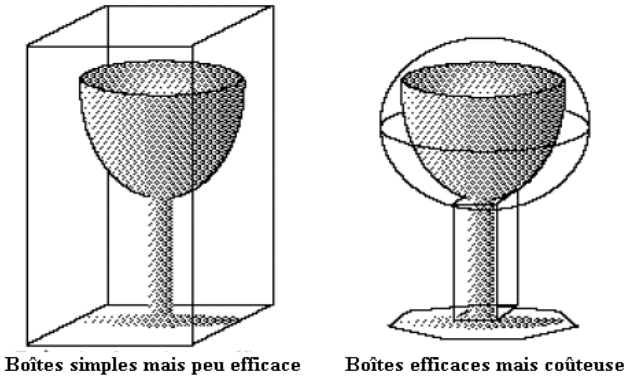


Fig. 2.11 : Exemples de boîtes englobantes.

- Les techniques de division spatiales dont l'idée générale est de considérer l'espace de la scène dans sa globalité puis de le découper en sous-espaces. Cette division de l'espace a donné naissance au concept de voxel 3D. Les données seront alors organisées en Octrees [YMS83] ou en BSP tree en effectuant des divisions binaires récursives de l'espace (Binary Space Partition) [Ka85] (*Fig. 2.12*)

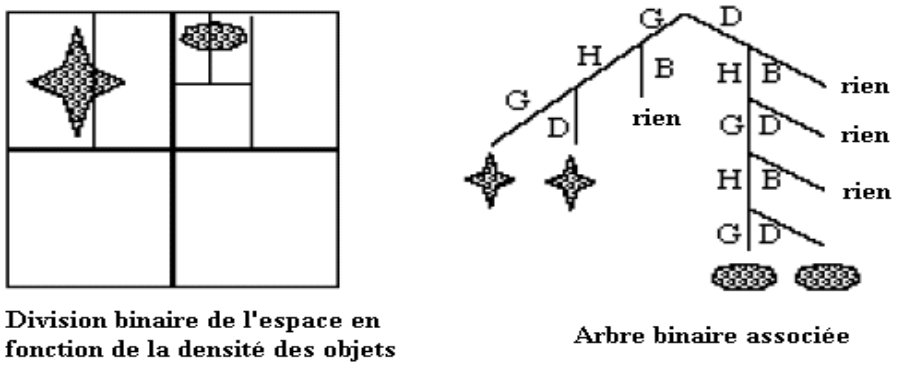


Fig. 2.12 : Partition binaire de l'espace.

- La méthode des intervalleles englobant considère des lignes de balayages et calcule, pour chaque ligne des intervalleles englobant afin de ne lancer que les rayons qui correspondent aux intervalleles englobant de la ligne de balayage courante.

- Les méthodes qui optimisent le temps de calcul en faisant une classification des rayons lancés [AK87, SDB85, MULL86 et STN87] ou en lançant des rayons volumiques appelés faisceaux [HH84] ayant la forme d'un cône [AMAN84] ou d'un pyramide de section polygonale quelconque [GHAZ92, GH98a et GH98b] (*Fig. 2.13*) .

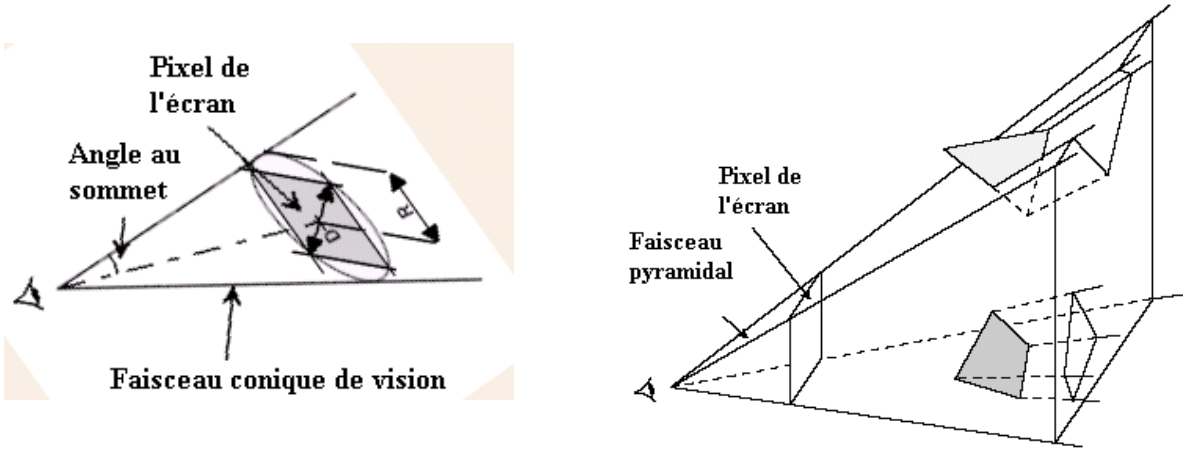


Fig. 2.13 : Optimisation par lancé de faisceaux de rayons.

- Les techniques qui utilisent des machines parallèles afin d'économiser le temps et qui sont basées sur une parallélisation à l'aide d'un processeur par pixel ou groupe de pixels, par voxel ou par objet. Pour avoir plus d'information, il suffit de se rapporter aux travaux [GS85], [CPC84], [GP89], [PR89], [KH95], [LUC91] et [PIT93].

Un tableau relativement complet qui résume le comportement des méthodes d'accélération de la technique de lancer de rayon figure dans [EXC88]. Voir aussi [AMAN84], [AK87], [STN87], [KK86], [GLA84], [GH98a], [GH98b], [HAZ98], [RIS96] et dans [CHH02] et [CRR06] les techniques les plus récentes qui tendent à accélérer les calculs d'intersections rayon-triangle à l'aide de vertex et de pixel shaders en utilisant des processeurs graphiques (GPUs).

En plus de ces techniques d'accélération de l'algorithme de lancer de rayon, il existe deux techniques assez importantes dont le but est d'obtenir une réduction significative du temps de calcul, tout en préservant une qualité acceptable des images obtenues: Le raffinement sélectif et l'expansion sélective [PLE87]. Nous allons étudier en détail l'algorithme du raffinement sélectif que nous allons utilisé dans notre méthode d'amélioration de la compréhension de visibilité.

III.1.1. Le raffinement sélectif et lancer de rayon

Le raffinement sélectif est l'une des approches les plus élégantes qui tend à diminuer le temps de calcul de l'algorithme de lancer de rayon en limitant le nombre de rayons lancés. Elle combine les méthodes classiques d'élimination des parties cachées avec des techniques de subdivision. L'objectif principal de cet algorithme est la réduction de la complexité algorithmique du problème de la visibilité

Cette technique a été initialement utilisé par WARNOCK en 1969 dans [WAR69] dans son algorithme d'élimination des parties cachées par des subdivisions récursives de l'espace image, puis par CATMULL en 1974 dans [CE74] pour la visualisation des surfaces gauches par subdivisions des surfaces de la scène jusqu'à l'obtention des situations simples où le problème de la visibilité pouvait être résolu facilement. JANSEN et VAN WIJK [JAW84] et PLEMENOS [PLE87] ont proposé des méthodes similaires appliquées à l'élimination des parties cachées par l'algorithme de lancer de rayon.

L'idée de base de la méthode détaillée dans [PLE87] est la suivante:

> *Pour une région de l'espace image, un nombre limité de rayons est lancé, l'intensité lumineuse des points correspondants est calculée et les surfaces visibles respectives sont déterminées. Si, à partir de ces renseignements, il s'avère qu'il y a une forte probabilité pour que la même surface visible corresponde à tous les points de la région, la surface visible en question est affectée à la région. Dans le cas contraire, la région est subdivisée et le processus recommence. La subdivision maximale n'est atteinte que dans le voisinage des contours et des intersections des surfaces.*

Principe

Cet algorithme possède deux versions. Une idée commune consiste à diviser l'espace image en des macros pixels de $2^n x 2^n$ pixels chacun.

Première version

1. Pour chaque macro pixel, un rayon est lancé de l'observateur vers son pixel haut gauche HG (*Fig. 2.14*).

2. Si la surface visible à partir de ce pixel est différente de celles vues par les pixels HG des macros pixels voisins, le macro pixel courant est subdivisé en quatre sous macros pixels et le processus recommence pour chacun des quatre nouveaux macros pixels ainsi créés.

3. Sinon, il y a une grande probabilité que tous les pixels du macro pixel courant voient la même surface visible et alors une intensité lumineuse approchée leur est attribuée et qui peut être obtenue en faisant une interpolation linéaire entre les intensités lumineuses des pixels HG du macro pixel courant et ceux des macros pixels voisins.

4. Le processus de subdivision s'arrête dès qu'un seuil de subdivision, défini par l'utilisateur, est atteint.

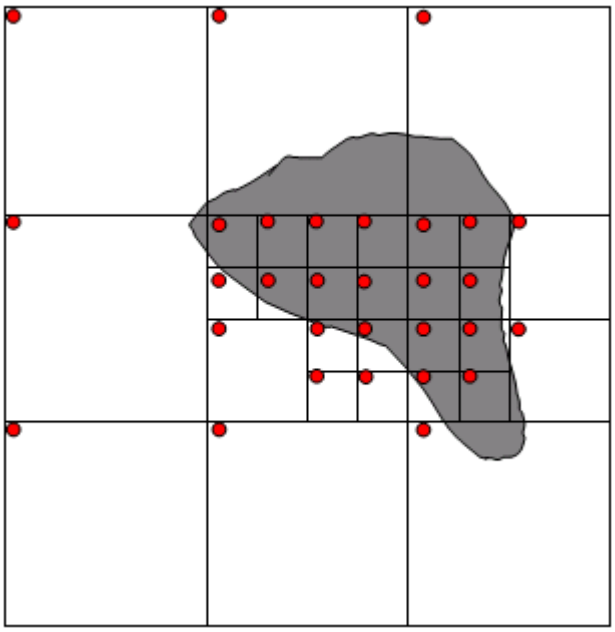


Fig. 2.14 : Raffinement sélectif en version 1.

Deuxième version

1. Dans chaque macro pixel, des rayons sont lancés de l'observateur vers des pixels appelé « pixels-guides » dont le nombre et la position sont fixés à l'avance par l'utilisateur. Pour obtenir des résultats probants, le nombre de pixels-guides doit être au moins égale à 3 (*Fig. 2.15*).

2. Si les surfaces visibles à partir de ces pixels guides sont différentes, le macro pixel courant est subdivisé en quatre sous macros pixels et le processus recommence pour chacun des quatre nouveaux macros pixels ainsi créés.

3. Sinon, on considère que la même face est visible par tous les pixels du macro pixel courant et une intensité lumineuse approchée leur est attribuée.

4. Le processus de subdivision s'arrête dès qu'un seuil de subdivision, défini par l'utilisateur, est atteint.

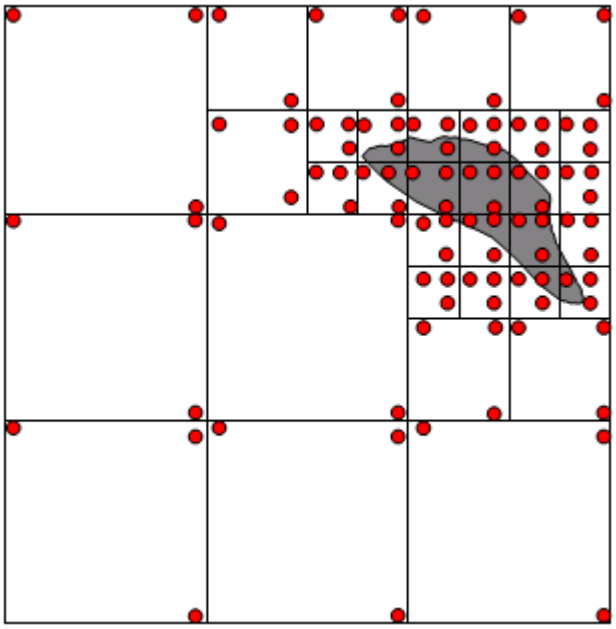


Fig. 2.15 : Raffinement sélectif en version 2 avec 3 pixels-guides Haut-Gauche, Haut-Droit et Bas-Droit.

En plus du raffinement sélectif, il existe aussi les techniques de l'expansion sélective appliquées au lancer de rayon dans le but de réduire le nombre de rayons lancés afin d'éviter les rayons qui ne coupent pas la scène et donc de réduire le temps de calcul des parties visibles de la scène. Cependant cette technique dépend d'un échantillonnage préalable, déstiné à déterminer la liste initiale de macros pixels utiles. Ainsi un échantillonnage insuffisant risque parfois de supprimer certains petits objets. Pour avoir plus de détails sur l'algorithme de cette technique, il suffit de jeter un coup d'oeil dans [PLE91].

IV. Avantages et inconvénients

La plupart des algorithmes de visualisation classique sont relativement rapides et, avec l'augmentation de la puissance des ordinateurs, ils permettent d'obtenir rapidement des images simples. Cependant, ces techniques présentent des inconvénients dont les principaux sont les suivants:

- il s'agit en général d'algorithmes complexes dont la mise en œuvre est souvent difficile.

- Le traitement global appliqué aux scènes par ces algorithmes, s'il permet de calculer correctement l'intensité lumineuse directe à chaque point d'un élément de la scène, est assez mal adapté au calcul des ombres portées, de la transparence et de certaines composantes de l'intensité lumineuses dues à l'éclairement indirect.

Le lancer de rayon a par contre l'avantage d'être simple, facile pour être mis en oeuvre et exige une faible occupation en mémoire. Il donne une bonne qualité d'image qui est la plus proche de la réalité. Il facilite le traitement des effets de lumières tout particulièrement celui des ombres portées ainsi que le traitement de la semi transparence. Bien évidement aussi, le règlement des réflexions et des réfractions.

La visualisation réaliste par la méthode de lancer de rayon a aussi l'avantage d'être capable de s'appliquer à des scènes mixtes (composées de surfaces et des facettes planes) car seul le module des intersections change selon la nature de la scène.

Cependant, en plus du problème de l'aliassage, cette technique de visualisation nécessite un temps de calcul très élevé qui est de l'ordre O(nb_Ligne*nb_Colonne*nb_Surface). Ainsi pour un espace image de 512 * 512 pixels avec une scène de 10 surfaces, le module de calcul d'intersections sera appelé 2 621 440 fois.

D'où l'importance de la combinaison de l'algorithme de lancer de rayon avec le raffinement sélectif qui réduit le temps moyen de calcul obtenu par le lancer de rayon « pur » en le divisant par six.

Dans les chapitres suivants, nous allons présenter une nouvelle technique basée tout d'abord sur le lancer de rayon avec le raffinement sélectif qui est d'une part utile pour la création des effets hyper réaliste et l'optimisation du temps et du calcul, et qui est, d'autre part, combiné avec la technique de suivi de contour, nécessaire pour la détection des contours apparents des objets réels de la scène afin d'améliorer la prise de connaissance des scènes complexes qui sont difficiles d'être comprises visuellement à cause de la présence des effets réalistes qui peuvent confondre l'utilisateur et l'empêcher de distinguer entre ce qui est réel et ce qui est illusoire.

Nous allons ensuite décrire trois nouvelles techniques de visualisation qui font une combinaison entre les modes de visualisation classiques existantes afin d'améliorer la compréhension des scènes qui comportes des objets englobant d'autres objets.

Mais avant d'en parler, nous allons présenter brièvement dans le chapitre 3, les techniques existantes de prise de connaissances qui sont basées sur l'exploration automatique autour de la scène par une caméra virtuelle.

Chapitre 3 Prise de connaissance par exploration automatique

Nous allons présenter dans ce chapitre les principales techniques existantes qui permettent d'avoir une meilleure prise de connaissance des scènes tridimensionnelles. Dans le cas de scènes simples, il est parfois suffisant d'avoir une bonne compréhension par le calcul automatique d'un bon point de vue. Mais lorsque la scène à traiter devient plus complexe, il est peut être nécessaire d'effectuer une animation autour de la scène à travers de bons points de vue décrivant un chemin qui suit des règles heuristiques évitant ainsi les brusques changements de directions qui peuvent distraire l'observateur.

I. Introduction

Le problème de la compréhension des scènes 3D est devenu de nos jours de plus en plus pertinent à cause du développement des applications aux sites web et de la possibilité de l'utilisateur de découvrir de nouvelles scènes sur l'Internet dont il n'a aucune idée préalable du contenu. Ses scènes sont parfois difficiles à comprendre à cause de plusieurs raisons.

L'une des raisons les plus simples est le choix d'une mauvaise position du point de vue ce qui pourrait faire manquer à l'observateur des détails importants ou des informations nécessaires à la prise de connaissance. Le calcul d'un bon point de vue est nécessaire dans ce cas là afin d'améliorer la visualisation de la scène.

Mais parfois dans le cas où il s'agit de comprendre un monde virtuel complexe, les informations que donne un seul point de vue deviennent insuffisantes à la compréhension de la scène. Même le calcul de plus d'un bon point de vue n'est pas généralement dans la plupart des cas, une solution satisfaisante parce que la transition d'un point à un autre pourrait être brusque ce qui peut déconcerter et confondre l'utilisateur, surtout lorsque le nouveau point de vue est loin du point courant.

La meilleure solution dans le cas d'un monde complexe est d'offrir une exploration automatique à l'aide d'une caméra qui se déplace autour de la scène suivant des bons points de vue et selon un chemin spécifique qui suit certaines règles afin d'éviter les changements brusques des directions de vue.

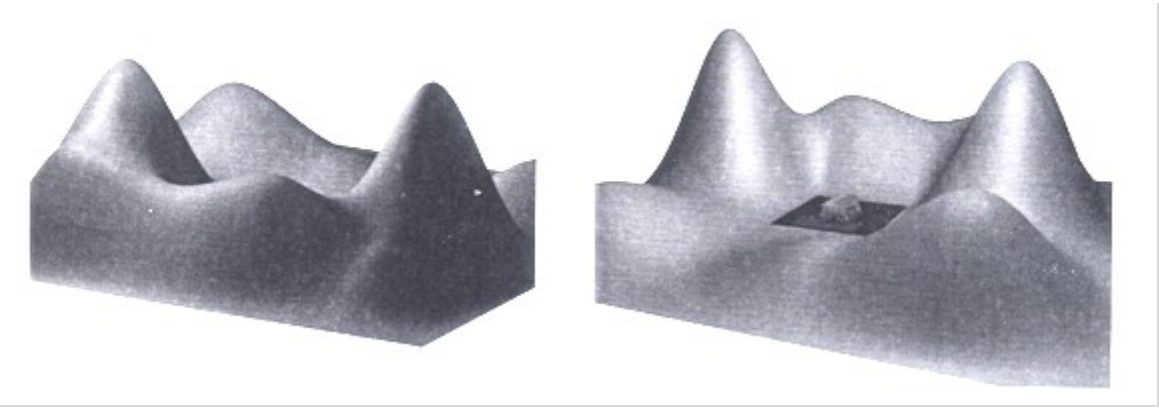
II. Techniques principales d'exploration d'une scène

Les premiers travaux dans le domaine de la compréhension du monde virtuel ont été publiés à la fin des années 80 et le début des années 90. Il y avait très peu de travaux parce que la communauté de l'informatique graphique n'était pas convaincue que ce domaine est important pour l'infographie.

Le but de ces travaux était d'aider l'utilisateur à comprendre le monde virtuel par le calcul d'un bon point de vue permettant ainsi de voir le plus de détail possible dans la scène.

II.1. Calcul d'un bon point de vue

Parfois, la position du point de vue peut aider ou non à avoir une bonne prise de connaissance. Par exemple, *Fig. 3.1* présente deux positions différentes du point de vue pour une même scène. La partie (a) de la figure présente une mauvaise prise de connaissance due à une mauvaise position du point de vue. Un simple changement de la position du point de vue a donné l'image de la partie (b) qui est certainement meilleure que la première puisqu'on a pu avoir une idée plus globale et détaillée sur le contenu de la scène.



(a) *(b)*

Fig. 3.1: (a) vue maladroite d'un décor. (b) vue typique d'un décor.

Pour cela, il est important d'être capable de proposer une technique automatique permettant le calcul d'une bonne position d'un point de vue. Plusieurs méthodes ont été proposées pour détecter un bon point de vue.

Kamada et Kawai considèrent dans [KK88] qu'une direction est considérée comme un bon point de vue si elle minimise le nombre des images dégénérées obtenues en projetant orthogonalement la scène. Une image est dite dégénérée si plusieurs arêtes sont superposées sur une même ligne. (*Fig. 3.2*)

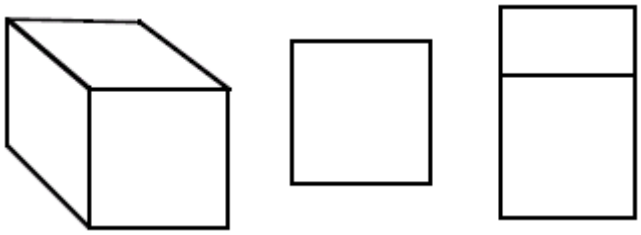


Fig. 3.2: (a) Vue non dégénérée. (b) et (c) Vues dégénérées.

Ils ont proposé de minimiser l'angle maximal de déviation entre une direction de vue et la normale au plan d'une face considérée. Cette technique est intéressante pour une visualisation en fil de fer. Cependant, elle n'est pas efficace pour une visualisation réaliste puisqu'elle ne prend pas en considération la visibilité des objets et un grand élément de la scène peut cacher tous les autres à la fin du rendu.

Plemenos et Beneyada ont proposé dans [PB96] une méthode itérative pour un calcul automatique d'un bon point de vue et ils ont créé une heuristique qui étend la définition donnée par Kamada et Kawai. Un point est considéré comme étant un bon point de vue s'il donne, en plus de la minimisation de la déviation entre la direction de vue et les normales des facettes, le plus grand nombre de détails. La qualité du point de vue est basée sur deux critères géométriques: le nombre de polygones visibles et l'aire des parties visibles projeté. Ceci est calculé en utilisant la formule suivante:

$$I(v) = \frac{\sum_{i=1}^{n} \left[\frac{P_i(v)}{P_i(v)+1} \right]}{n} + \frac{\sum_{i=1}^{n} P_i(v)}{r} \qquad (1)$$

Où:

- $p_i(v)$:la surface projetée visible du polygone numéro i obtenu du point de vue v.
- r : la surface totale projetée.
- n : le nombre total de polygones dans la scène.
- [a] : le plus petit entier ≥ a

En pratique ces mesures sont calculées tout simplement dans [PDB99] et [DG01] en utilisant OpenGL. Pour chercher le nombre de polygones visible et la surface projetée visible de chaque polygone on utilise OpenGL, en calculant le rendu de la scène à l'aide d'un Z-buffer en associant une couleur différente à chaque face sans tenir compte d'aucun éclairage, ni faire d'anti-aliassage, ni de tramage.(*Fig. 3.3*)

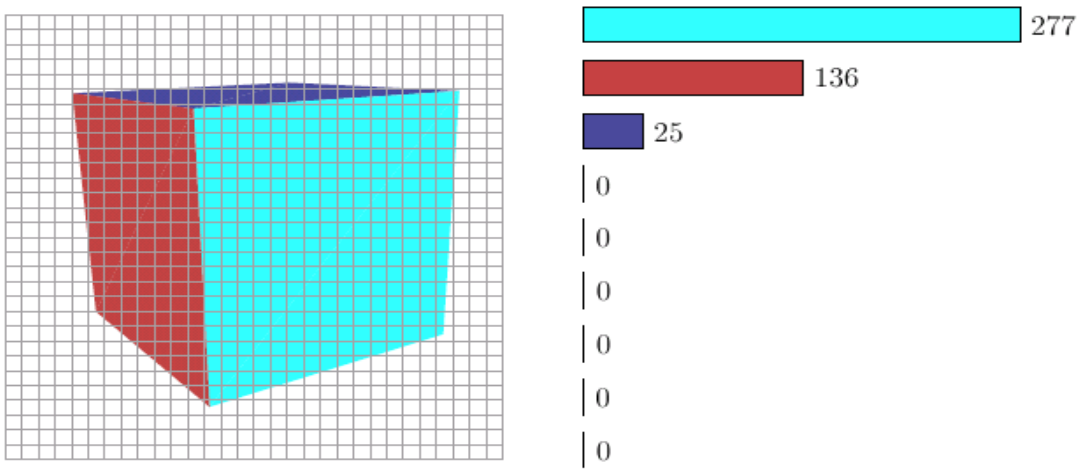


Fig. 3.3: un calcul rapide du nombre des surfaces visibles.

Le processus pour déterminer la position d'un bon point de vue se fait comme suit:

- La scène est placée au centre de la sphère unité dont la surface représente tous les points de vue possibles.

- La sphère est divisée par ces 3 axes de coordonnées x, y et z, en 8 triangles sphériques (*Fig. 3.4*).

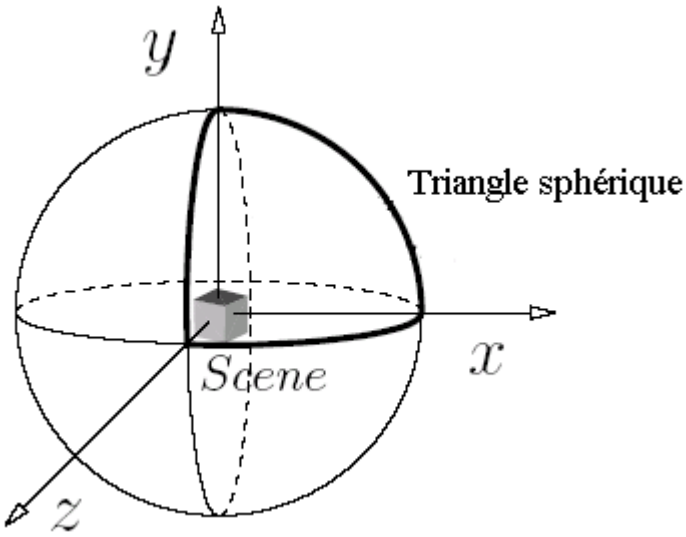


Fig. 3.4: La sphère des points de vue est subdivisées en 8 triangles sphériques.

- Le meilleur triangle sphérique choisi est celui dont les sommets représentent la meilleure qualité obtenue par la formule (1)

- Finalement, le meilleur point de vue est choisi dans le triangle sphérique à travers des subdivisions successives. (*Fig. 3.5*)

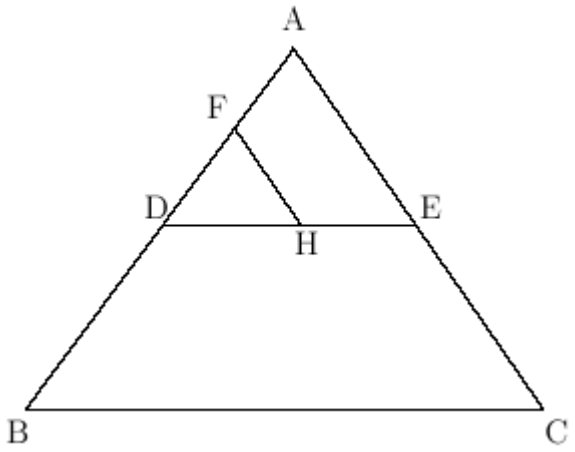


Fig. 3.5: Division récursive d'un triangle sphérique.

Cette méthode donne généralement des résultats intéressants. Cependant, le critère de nombre de polygones visibles peut produire quelques inconvénients.

Afin de résoudre ce problème, Sokolov et al. dans [SP05] et [SPT06], ont proposé de remplacer le critère du nombre de polygone par celui de la courbure ou l'inflexion totale de la scène qui est donnée par l'équation suivante:

$$I(p) = \sum_{v \in V(p)} \left| 2\pi - \sum_{\alpha_i \in \alpha(v)} \alpha_i \right| \cdot \sum_{f \in F(p)} P(f) \qquad (2)$$

Où

- F(p): l'ensemble des polygones visibles d'un point de vue p
- P(f) : l'aire de la surface projeté du polygone f
- V(p): le nombre de sommets visibles de la scène d'un point de vue p
- α(v) : l'ensemble des angles adjacents au sommet v

La formule (2) utilise la courbure en un sommet qui est la somme de tous les angles adjacents à un sommet v retranché de 2π. (*Fig. 3.6*)

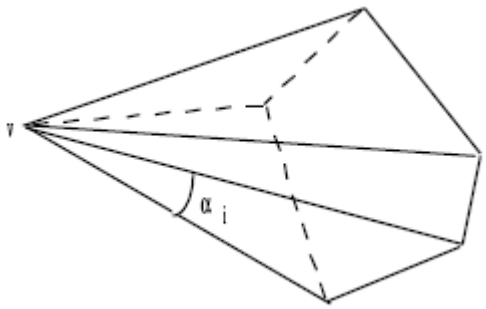


Fig. 3.6: les courbures d'un sommet

Le meilleur point de vue est calculé en utilisant une donnée de structure appelée graphe de visibilité (visibility-graph), qui permet d'associer, à chaque point de vue potentiel discret de la surface de la sphère, une pertinence visuelle de vue.

Colin a proposé dans [CC88] une méthode qui calcule le bon point de vue pour des modèles d'octrees. La direction de vue est considérée bonne si elle représente une grande quantité de voxels. La méthode consiste à calculer tout d'abord le meilleur triangle sphérique parmi les 8 triangles obtenus par les 3 axes de coordonnées. Il calcule ensuite dans ce triangle sphérique, la meilleure direction de vue en utilisant une interpolation linéaire entre les 3 directions du triangle sphérique.

Vasquez et Sbert ont créé dans [VSFH02, VS03] un nouveau critère pour évaluer la quantité d'information capturée d'un point de vue. Ce critère est appelé l'entropie d'un point de vue. L'entropie d'un point de vue est donnée par la formule suivante :

$$I(p) = \sum_{i=0}^{N_f} \frac{A_i}{A_t} \cdot \log_2 \frac{A_t}{A_i}, \qquad (3)$$

Où:

- N_f : le nombre de faces dans la scène
- A_i : la surface de la face i projeté sur la sphère des directions centrées au point de vue.
- A_t : la surface totale projetée tout autour de la sphère.

L'entropie est maximale lorsque le point de vue peut voir toutes les faces avec la même surface relative projetée A_i / A_t. Un bon point de vue est celui qui correspond à l'entropie maximale choisie parmi un ensemble de points de vue placés sur la surface de la sphère qui entoure la scène.

D'autres techniques de calcul d'un bon point de vue sont détaillées dans [PSF04, SP05, SPT06]

II.2. Animation autour de la scène

Beaucoup de techniques ont été développées afin d'obliger la caméra virtuelle à faire une animation (autour de la scène ou à l'intérieur) tout en suivant des chemins bien définis et respectant certaines lois et règles qui empêchent la confusion et le retour rapide de la caméra vers la position de départ.

Deux types d'explorations ont été proposés afin d'assurer l'exploration du monde virtuel d'une manière dynamique.

- **L'exploration globale :** la caméra se déplace tout en restant à l'extérieur du monde à explorer. Ces techniques permettent d'avoir une vue globale de la scène mais elles peuvent manquer certains détails inaccessibles à partir de l'éxterieur.

- **L'exploration locale :** la caméra se déplace à l'intérieur de la scène. Elle devient comme une partie d'elle.

Les techniques de l'exploration locales permettent d'atteindre tous les détails visibles possibles du monde virtuel mais elles ne donnent pas une idée globale de la scène. Elles peuvent être utiles même parfois nécessaires dans certains cas mais l'exploration globale est la seule méthode qui est capable de fournir une connaissance globale de la scène.

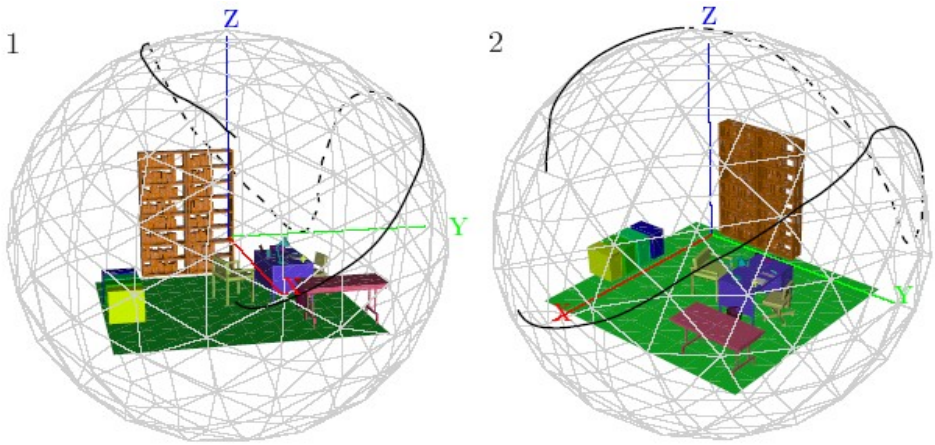


Fig 3.7: exemple d'une exploration globale.

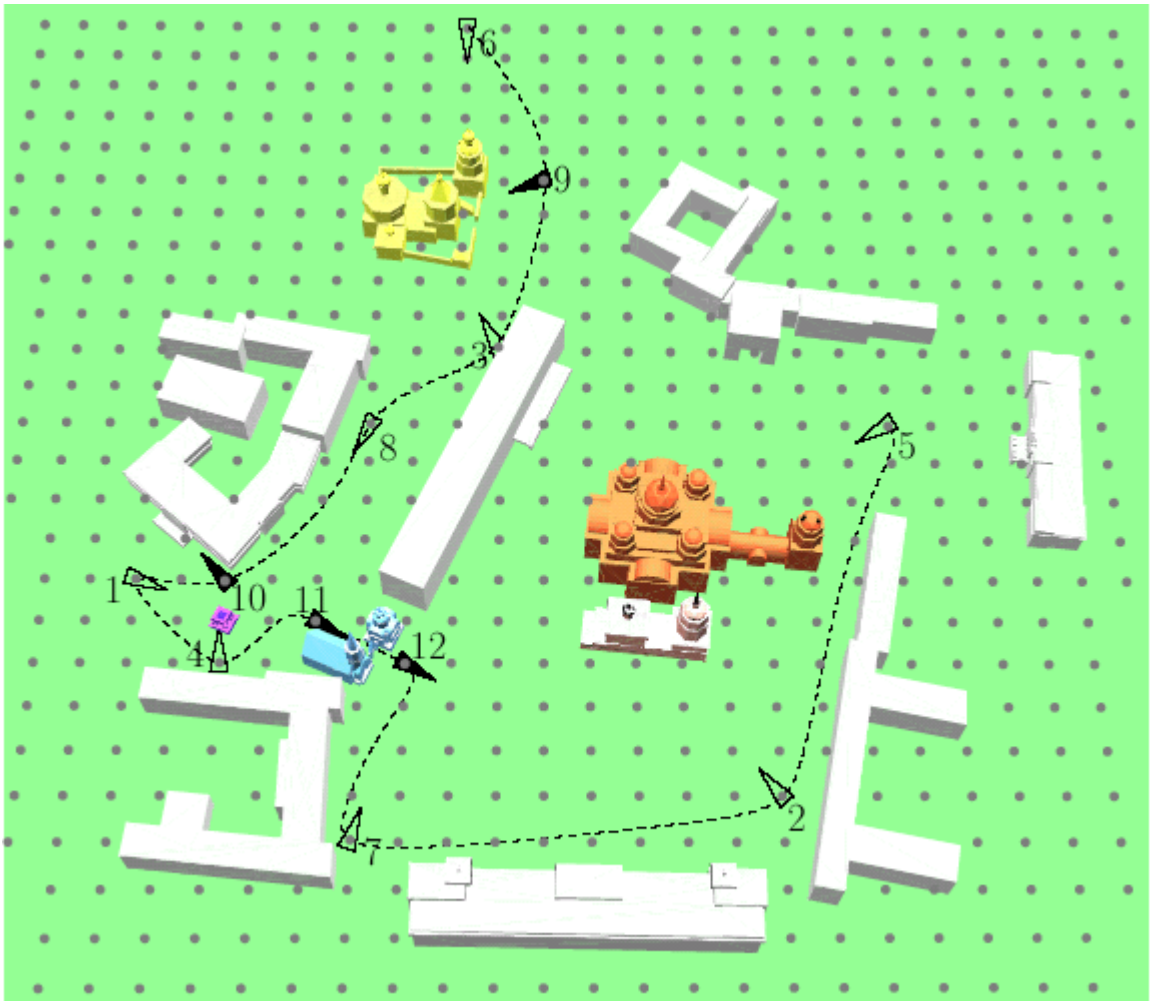


Fig 3.8: exemple d'une exploration locale.

L'exploration globale

Plemenos et al. et Dorme [PDB99], [PDB00], [DG01], ont proposé une méthode dans laquelle une caméra virtuelle se déplace en temps réel sur la surface d'une sphère unité entourant le monde virtuel.

La scène a été examinée d'une façon incrémentale durant l'observation. Tous les polygones du monde virtuel sont pris en considération à chaque étape de l'exploration. La méthode est basée sur de variantes de formule heuristique suivante:

$$w_c = \frac{n_c}{2}\left(1+\frac{d_c}{p_c}\right) \qquad (4)$$

Où :

1. w_c est le point de la position courante de la caméra

2. n_c est la complexité du point de vue à partir de la position courante de la caméra

3. p_c est la longueur du chemin tracé par la caméra du point de départ jusqu'à la position courante.

4. d_c : la distance entre la position de départ et la position courante de la caméra.

Afin d'éviter les retours rapides vers la position de départ, l'importance d'un point de vue est choisie inversement proportionnelle au chemin décrit par la caméra du point de départ vers la position courante. En plus, pour avoir un mouvement souple de la caméra, seulement trois nouveaux points de vue sont pris en considération pendant le calcul de la position suivante. (*Fig. 3.7*)

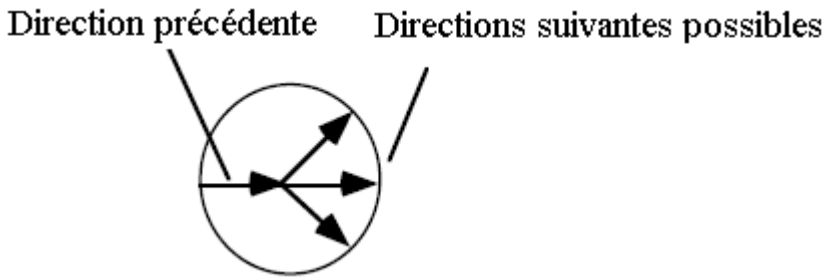


Fig. 3.9 : Seulement trois directions sont considérées afin d'assurer un chemin souple de la caméra

Plusieurs variantes de cette technique ont été proposées et appliquées. Vasquez et al. [VS03], [VP03] utilisent une méthode similaire pour les explorations intérieures (indoor) et extérieure (outside) au monde virtuel. Ils ont utilisé le critère de l'entropie pour le calcul de la pertinence d'un point de vue.

Sokolov et al. [SP05], [SPT06], ont aussi proposé une méthode qui utilise la structure de visibilité des graphes (the visibility graph structure) permettant de voir tous les sommets de la scène et qui est calculée d'une façon incrémentale. L'idée principale est d'avancer la caméra vers des endroits non explorés encore.

Ainsi, connaissant la trajectoire du point de départ vers la position courante de la caméra, la caméra est poussée vers des points de vue pertinents permettant la vue d'un maximum de détails et de régions qui ne sont pas encore vues de la caméra. Pour ce faire, à chaque étape, un poids est associé à chaque point de la sphère discrète ainsi qu'à la position courante de la caméra.

La valeur du poids associé à un point de vue est choisie selon la nouvelle information apportée par le point de vue (la formule (2)). La position de la caméra est alors soumise à la loi gravitationnelle de Newton. La superposition des forces gravitationnelles de la position courante de la caméra est un vecteur de mouvement.

Une autre méthode pour le calcul d'un ensemble minimal de points de vue a été proposée par Jaubert et al. [JPGT05][JPT06]. Dans cette méthode, un nombre suffisant de points de vue est calculé au début, puis l'ensemble minimal de bons points de vue est crée par la suppression successive de ceux qui ne permettent pas de voir plus de détails que les autres restants.

L'exploration locale

Dans certain cas nous avons besoin d'explorer ou se déplacer à l'intérieur d'un monde virtuel comme par exemple dans le cas des jeux vidéo (avatars), l'exploration d'un musé, etc...

Dans ce type d'exploration, il faut mette en évidence trois conditions essentielles:

- éviter les obstacles: la caméra ne doit pas passer à travers les objets
- la caméra doit explorer la partie importante de la scène
- la caméra doit le plus possible montrent de bonnes vues.

Les obstacles peuvent être statiques (des murs, des objets ou des surfaces inaccessibles) ou dynamiques (des obstacles qui peuvent changer le formes ou de positions à travers le temps).

Yunfang et al [YZW03] ont présenté un algorithme basé sur une vision synthétique utilisée pour déterminer dynamiquement le chemin parcouru par un avatar dans un monde d'Intelligence Virtuelle. Ils ont utilisé la vision synthétique et la scène d'Octree pour simuler le sens de la vision d'un avatar de l'environnement et de la mémoire de la scène respectivement.

Vazquez and Sbert [VS03] proposent une méthode automatique d'exploration locale d'une scène avec un nombre limité de degrés de liberté afin de simuler la façon dont un être humain marche. La méthode est basée sur le critère de l'entropie pour le calcul d'un point de vue. Le processus d'exploration s'arrête au moment où la caméra est incapable de découvrir de nouvelles informations.

Dans [MC00, VFSH02, JPGT05], une technique basée image est utilisée pour contrôler le mouvement de la caméra dans un monde virtuel qui peut changer à travers le temps. Le problème de cette technique est l'adaptation de la caméra face au changement du monde.

D'autres techniques sont aussi détaillées et proposées dans [SOK06] pour l'exploration locale dans un monde virtuel.

III. Conclusion

Nous avons brièvement présenté dans ce chapitre les techniques existantes qui aident à avoir une meilleure prise de connaissance, soit à travers le calcul d'un bon point de vue, soit à travers une animation globale ou locale autour du monde virtuel.

Ces techniques ont beaucoup d'avantages dans l'amélioration de la compréhension de la scène. Surtout lorsque la caméra virtuelle est placée en une bonne position de point de vue permettant de voir le plus possible d'informations et de détails.

Ainsi les techniques d'exploration autour de la scène, que ce soit globale ou locale, sont efficaces pour donner une idée globale de la scène et permettre de visiter des endroits de la scène qui sont difficiles à découvrir par une simple image obtenue d'un bon point de vue.

Cependant, ces techniques sont dans la plupart des cas, lentes et coûteuses en mémoires, surtout lorsqu'il s'agit de scènes complexes, il serait difficile de les appliquer en temps réel.

D'autre part, ces techniques utilisées seules ne sont pas suffisantes dans le cas des scènes complexes comportant des miroirs, des lumières et des objets transparents (*Fig 3.10*). Les effets réalistes qui en résultent, pourraient être gênants et empêcher l'observateur d'avoir une bonne compréhension du monde virtuel.

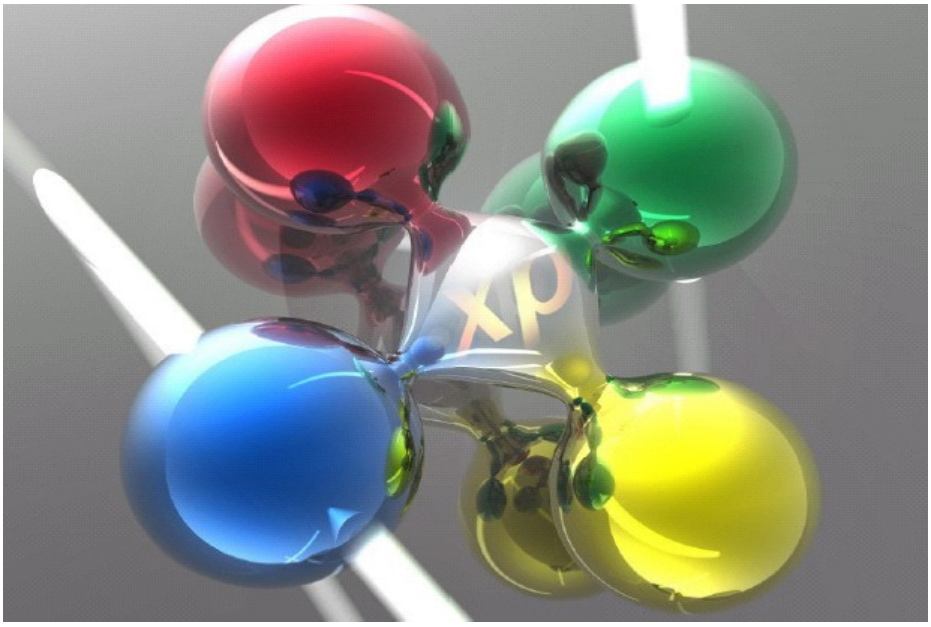


Fig.3.10(a):Exemple d'une scène réaliste

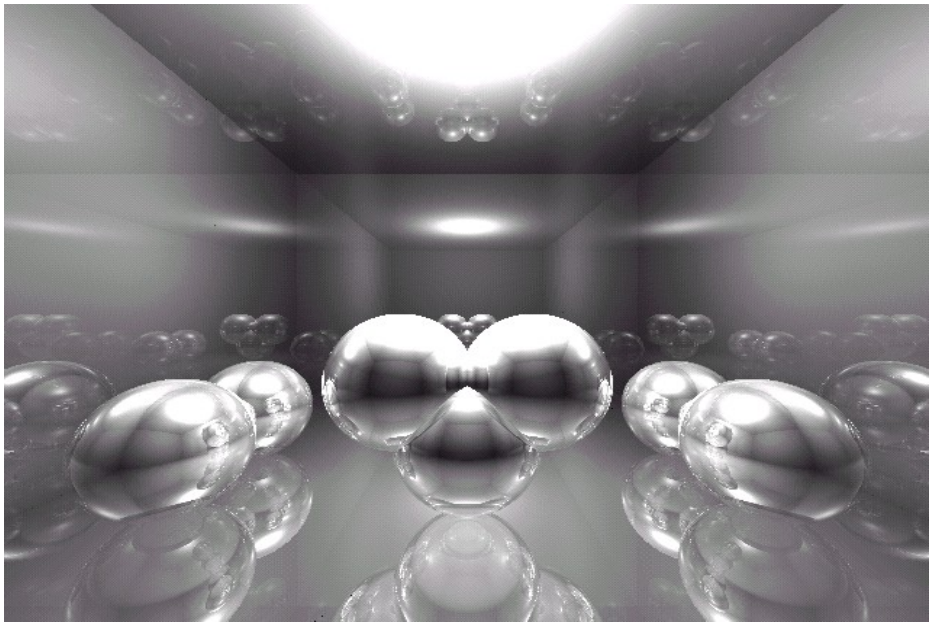


Fig 3.10(b): Exemples de scènes complexes comportant beaucoup d'effets réalistes.

La présence des illusions dues aux ombres et aux objets réfléchis et réfractés peut confondre l'observateur parce qu'il sera difficile de distinguer entre ce qui est réel et ce qui est illusoire. Même si l'utilisateur choisi une bon point de vue et/ou fait une exploration globale ou locale autour de la scène, les illusions vont toujours distraire et confondre l'utilisateur et l'empêcher d'avoir une bonne prise de connaissance.

Pour cela, nous sommes bien convaincu qu'il faut trouver de nouvelles techniques autres que l'exploration automatique des scènes qui soient capables d'améliorer la compréhension à travers une nouvelle technique de visualisation qui met en évidence les objets réels de la scène afin de les distinguer de ce qui est illusoire.

En effet, entourer les objets réels de la scène par une certaine couleur pourrait à priori aider l'observateur à avoir une meilleur prise de connaissance, même s'il y a beaucoup de lumières, de réflexions et de transparences dans la scène parce que le contour ainsi ajouté tend à attirer l'attention vers l'objet réel et à le distinguer de son image par réflexion ou à mieux le mettre en évidence lorsqu'il y a beaucoup de lumières.

Pour ce faire, nous allons tout d'abord étudier brièvement dans le chapitre suivant les techniques de base de détection de contours. Une nouvelle approche appliquée aux scènes complexes avec un excès de réalisme sera ensuite présentée dans le but d'étudier sa performance dans l'amélioration de la compréhension des scènes complexes.

Chapitre 4 Prise de connaissance par extraction des contours apparents

---

Nous allons tout d'abord présenter dans ce chapitre les techniques existantes de détection des contours et qui sont catégorisées en 3 groupes d'algorithmes: les algorithmes de l'espace image, les algorithmes hybrides et les algorithmes de l'espace objet.

Une nouvelle technique sera ensuite présentée, permettant une meilleure prise de connaissance des scènes difficiles à comprendre à partir d'une image obtenue par rendu réaliste à cause de la présence des illusions dues aux effets de réflections et de réfractions. La technique est basée sur l'extraction des contours apparents des objets réels en utilisant l'algorithme de lancer de rayon avec l'approche du raffinement sélectif combinée par la méthode de suivi de contour par code de direction.

I. Introduction

Le terme de rendu non photoréaliste (RNP), a été depuis longtemps appliqué seulement pour des illustrations artistiques comme celles de « plume et encre » ou l'aquarelle. Récemment, plusieurs chercheurs en infographie ont découvert que le RNP peut être utilisé comme étant une méthode alternative au rendu réaliste.

En effet, les techniques de RNP sont capables d'extraire, transporter et examiner plusieurs caractéristiques et propriétés du matériel ainsi qu'ajouter certains détails ou éliminer des informations qui sont erronés ou supplémentaires indésirables afin d'aider l'observateur à avoir une meilleure prise de connaissance d'une scène tridimensionnelle complexe.

Pour cela, l'extraction des contours a pu jouer un un rôle central dans beaucoup d'applications. Elle n'est pas seulement utilisé dans le rendu non photoréaliste pour faire des dessins animés ou un style artistique, mais elle est aussi utilisée dans des illustrations techniques, en architecture, en atlas médicaux (médical atlases) [HZ00], en robotique médicale [OZ06], et dans l'amélioration du rendu photoréaliste comme un moyen efficace pour le calcul des ombres (shadow volumes) [HZ00] ainsi que pour la création rapide et le rendu souple des ombres sur un plan [HE01].

L'extraction des contours a aussi été utilisée pour faciliter le rendu haptique (Haptic rendering) [JC01]. Quelques auteurs, [CPSC98, JRP02] ont décrit l'usage des silhouettes dans des applications CAD/CAM. Des systèmes ont aussi été construits en utilisant la notion des silhouettes afin d'aider en modélisation et pour les taches de capture du mouvement [BL01, FPT99, et LGM00]. D'autres applications et de techniques basées sur la détection des contours sont décrites dans le cours 7 de SIGGRAPH 05 [RUS05].

Nous allons donc essayer de nous inspirer de ces techniques par l'utilisation de l'extraction des contours apparents des objets présents dans une scène complexe tridimensionnelle visualisée par le lancer de rayon et contenant beaucoup de lumières, des miroirs, et des objets transparents, afin d'affronter le problème de l'amélioration de la prise des connaissance.

L'approche proposée est basée sur la technique de lancer de rayons avec raffinement sélectif combinée par la technique de suivi de contour par le code de direction. Cette technique sera décrite en détail dans la partie III de ce chapitre mais

une présentation brève et rapide des techniques d'extractions de contours existantes sera proposée avant, dans la partie II.

II. Techniques existantes d'extraction des contours apparents

Avant de parler des techniques de détection des contours, il est bien nécessaire de définir en quelques mots les différents types de contours (*Fig. 4.1*):

- La silhouette de l'objet (silhouette edge): elle représente des surfaces ou volumes de la scène dont la projection sur l'écran a donné des arêtes. Elle est donc composée d'une suite de courbes ou d'arêtes adjacentes aux polygones visibles qui sont en face de la caméra (front-facing) et aux polygones invisibles qui sont derrière la caméra (back-facing).

- Les bordures (border edge): ce sont les vrai arêtes qui appartiennent exactement à un seul polygone.

- Les arêtes d'intersections (Crease edge): ce sont les arêtes ou les courbes d'intersections entre deux polygones front-facing (ou back-facing respectivement).

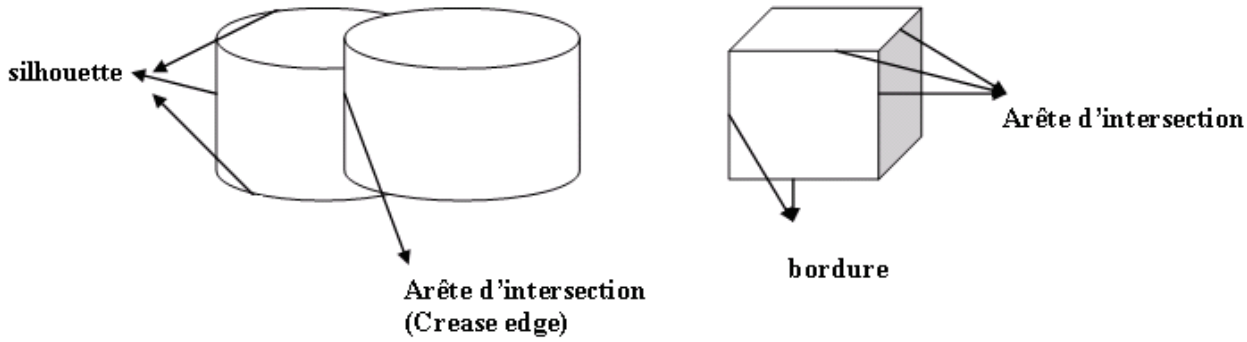


Fig. 4.1 : Les différents types de contours.

Isenberg distingue dans son article [IS03] entre les algorithmes de détection des contours dans l'espace image qui opèrent sur les tampons de l'image, les algorithmes hybrides dans lesquels les opérations se fond dans l'espace objet mais la silhouette est produite dans l'espace image, et finalement les algorithmes de l'espace objet où toutes

les opérations sont faites dans l'espace objet et où la silhouette résultante est représentée par une description analytique.

II.1. Les algorithmes de l'espace image

Dans l'espace image, la méthode la plus facile pour la recherche des contours est celle utilisée dans les techniques de l'analyse d'image qui tendent à détecter les contours en appliquant des filtres de détection des discontinuités dans les tampons d'images. Parmi ces filtres, on peut citer, Sobel [RK82], Canny, Dirich [CJ86], transformation de Hough etc… chacun est représenté sous la forme de matrices de convolutions.

La convolution de ces matrices avec par exemple la matrice des points au voisinage d'un pixel permet de calculer la variation de couleur autour de ce pixel au travers d'une fonction de variation. Il suffit donc de choisir ensuite un seuil convenable à partir duquel une variation locale est considérée comme la marque d'un point contour.

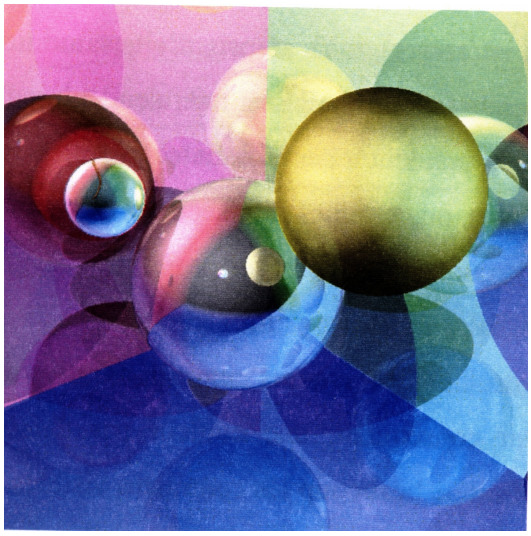


Fig. 4.2(a) Une scène comportant des effets réalistes

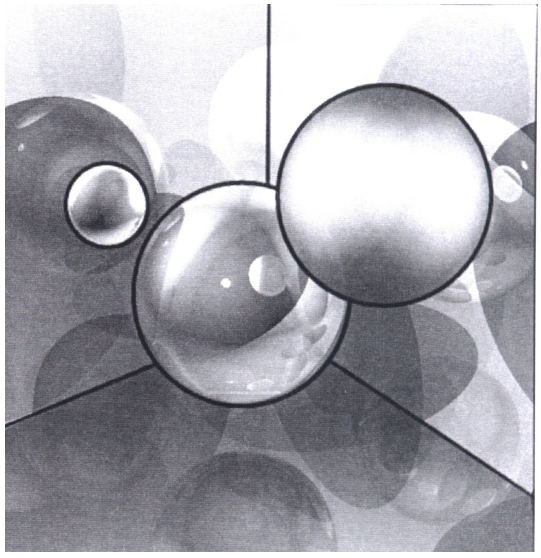


Fig. 4.2(b)Détection des contours par le filtre de sobel

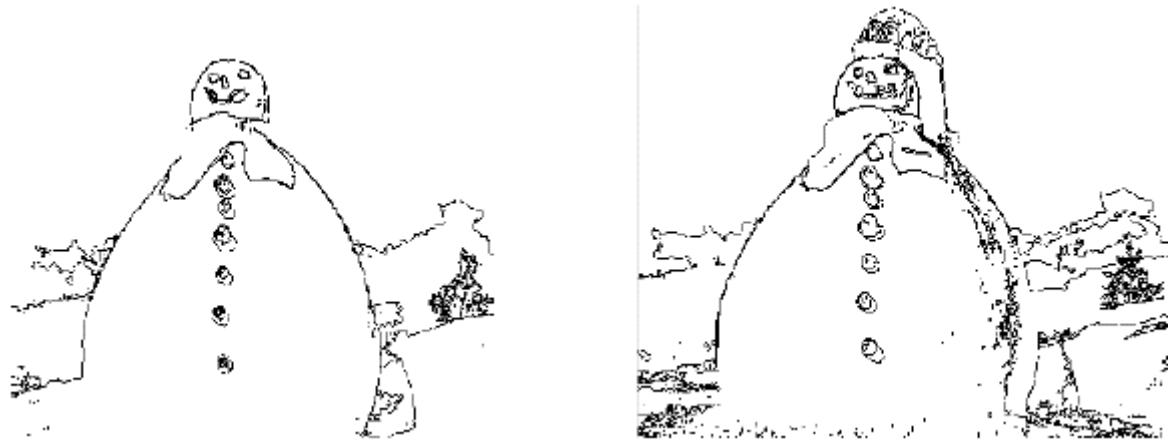
Saito et Takahachi [ST90], Decaudin [DEC96], Hertzmann [HA99], Deussen et Strothotte [DS00], puis Nienhaus et Dellner [ND03] ont étendu la méthode de filtrage pour l'appliquer sur des tampons géométriques autres que le tampon d'image appelés

les G-buffers qui sont formés du tampon de profondeur (z-buffer), du tampon des normales (normal-buffer) et du tampon d'identité (ID-buffer).

En détectant les discontinuités d'ordre zéro dans la carte de profondeur, nous obtiendrons la silhouette ou le profil des objets, puis en détectant les discontinuités d'ordre 1 de la carte des normales, nous obtiendrons les arêtes d'intersection. En joignant les deux contours obtenus, nous obtiendrons les contours apparents des objets d'une scène (*Fig. 4.4*).

Récemment, des techniques statistiques opérant sur l'espace image, détectent les contours en faisant une segmentation des couleurs des objets par utilisation des méthodes d'apprentissage des réseaux de neurones artificiels RNA [CNH03, CNH05, WLS00, APA02, LHS03]. Par exemple dans *Fig. 4.3*, il s'agit de la détection des contours de la scène Frosty par une méthode statistique basée sur l'utilisation d'un réseau de neurones artificiels auto organisateur SOM décrit dans [CNH03 et CNH05].

Les algorithmes des l'espace image ont l'avantage d'être relativement simples à implémenter puisqu'ils opèrent sur des tampons qui peuvent être générés facilement avec les cartes graphiques existants actuellement. Ils peuvent être appliqués à n'importe quelle scène indépendamment de la façon dont elle est modélisée et ils donnent de bons résultats dans les cas de scènes simples.



(a): détection des contours en utilisant 2 neurones

(b):détection des contours en utilisant 4 neurones

Fig. 4.3 : Détection des contours de la scène Frosty par une méthode statistique.

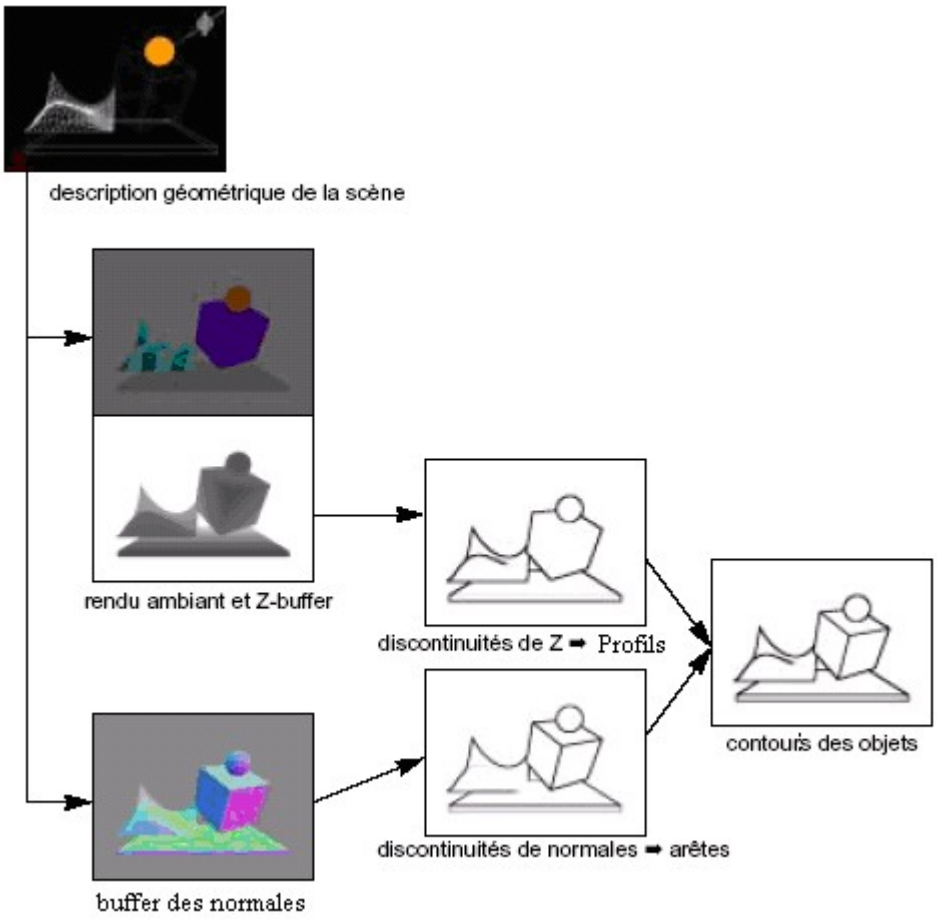


Fig. 4.4: Détection des contours par la détection des discontinuités de la scène

Cependant, ces techniques ont des limites. L'inconvénient principal est qu'elles dépendent, d'une certaine valeur seuil qui doit être ajustée par l'utilisateur à chaque fois qu'une nouvelle scène est introduite. Un autre inconvénient, avec des scènes complexes et en présence d' ombres, de textures et d'effets réalistes comme les réflexions et les réfractions, ces techniques deviennent incapables de détecter correctement les silhouettes. Des contours peuvent être ignorés et/ou d'autres ajoutés à cause des illusions dues aux ombres, et à la présence des objets réfléchis et réfractés.

II.2. Les algorithmes hybrides

Rustagi [RP89], Rossignac et Emmerik [RE92] utilisent des algorithmes hybrides caractérisés par des opérations sur l'espace objet (des translations) suivis à chaque fois par le rendu des polygones modifiés utilisant le tampon de profondeur. Raskar et Cohen [RC99] (*Fig 4.5*), Gooch et al. [GB99] puis Raskar [RR01], ont généralisé cette approche dans leurs travaux en utilisant le traditionnel tampon de profondeur avec l'élimination des faces arrières (back facing culling) ou des faces frontales (front facing culling) respectivement.

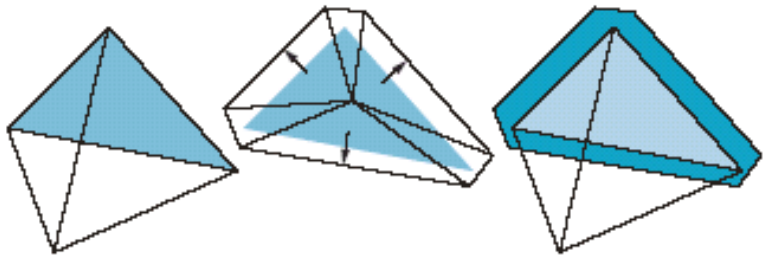


Fig. 4.5: Élargissement des polygones arrières afin de former des ligne larges des silhouettes.

Avec les algorithmes hybrides, l'apparence des images générées tend à être plutôt stylistique que réaliste.

Comme les algorithmes travaillant dans l'espace image, les algorithmes hybrides peuvent tomber dans des problèmes numériques dus à la limitation en résolution du tampon de profondeur.

II.3. Les algorithmes de l'espace objet

Le calcul des silhouettes dans ce groupe d'algorithmes, comme son nom le dit, prend place entièrement dans l'espace objet. L'algorithme trivial de l'espace objet est basé sur la définition de la silhouette comme étant exactement la ligne de séparation entre une face frontale et une face arrière.

L'algorithme se fait en deux étapes. Il s'agit tout d'abord de classifier tous les polygones comme frontaux ou arrières par rapport à un point de vue, l'algorithme examine ensuite dans la seconde étape toutes les lignes et sélectionne seulement celles qui partagent exactement un polygone frontal et un polygone arrière.

Afin de réaliser ce processus, Buchanan et Sousa [BS00] suggèrent d'utiliser une structure de base de données appelée tampon des bordures (edge buffer) dans laquelle ils réservent en mémoire, pour chaque ligne deux bits supplémentaires qui peuvent contenir F ou A pour des polygones frontaux ou arrières (*Fig. 4.6*) .

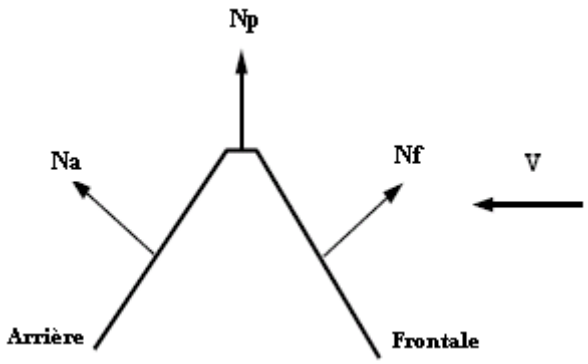


Fig. 4.6 : Une face jugée arrière ou frontale selon le sens de N.V

Cet algorithme avec ou sans l'usage de la structure du tampon de bordure garantit la recherche de tous les contours des objets de la scène. Il est facile d'être implémenté et convenable pour les applications qui travaillent sur de petits modèles.

Cependant, il est coûteux en temps de calcul et en mémoire parce qu'il est obligé de déterminer pour chaque face une orientation afin de préciser s'il s'agit d'une face frontale ou arrière. En effet il s'agit d'un produit scalaire calculé pour chaque face. Ainsi pour des projections perspectives, l'algorithme est obligé de recalculer le vecteur de la direction de vue pour chaque face et surtout à chaque fois qu'il étudie une ligne. C'est une méthode linéaire en terme de nombre de lignes et de faces mais ceci est très lent pour un calcul de rendu itératif d'un model de dimension assez grandes.

Afin d'accélérer cet algorithme, Card et Mitchell [CM02] ont suggéré de faire les calculs sur la carte graphique (GPU) au lieu d'utiliser l'unité centrale de processus (CPU).

Hertzmann [HA99] puis Hertzmann ET Zorin [HZ00] ont introduit une technique qui ne traite que le cas des mailles constituées de triangles (tout polygone peut être décomposé en triangles). La recherche des contours consiste donc à calculer, pour chaque sommet, le produit scalaire entre la normale et le vecteur de visée. (Puis à le normaliser)

$$d_i = \frac{\mathbf{n}_i \cdot (\mathbf{x}_i - \mathbf{C})}{||\mathbf{n}_i||||\mathbf{x}_i - \mathbf{C}||}$$

Le but est alors de trouver les points où $d_i = 0$. Ces points se situent entre deux sommets où le signe de d_i change et peuvent être calculés par interpolation linéaire (*Fig. 4.7(a)*) (Cela est correct, car on considère que l'objet est lisse). Il suffit ensuite joindre les parties des silhouettes obtenues dans chaque triangle afin de former une approximation de la silhouette totale. (*Fig. 4.7(b)*)

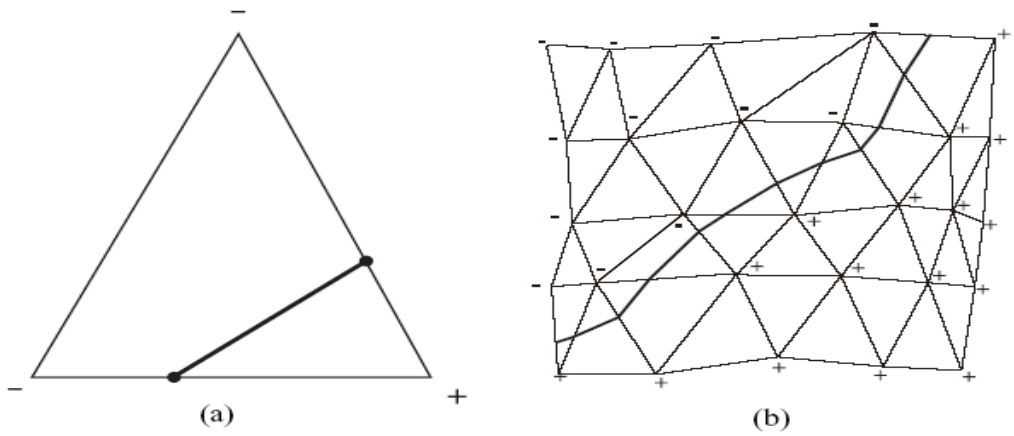


Fig. 4.7: Calcul de la silhouette par la méthode de Hertzmann.

La silhouette obtenue par cette méthode est plus proche de la silhouette réelle que celle obtenue par la méthode traditionnelle. Pour avoir plus de précision, on peut utiliser une subdivision adaptative en subdivisant les triangles qui contiennent un morceau de la silhouette et de réitérer l'algorithme. Pour la modélisation par surface implicite, on peut citer l'article de David J. Bremer [BH98].

Une fois les contours détectés, il faut en éliminer les parties qui ne sont pas visibles. Ce problème classique en infographie est traité dans de nombreux ouvrages, et les algorithmes se comptent par dizaines. Mais citons encore une fois l'article de Hertzmann [HA99] qui y consacre un paragraphe. La méthode présentée, consiste à couper les contours (courbes) au niveau des changements de visibilité potentiels. On obtient donc un nouveau jeu de courbes. Il suffit ensuite de déterminer celles qui doivent effectivement être affichées avec un test par rayon.

On distingue trois situations où la visibilité d'une surface courbe peut changer (*Fig. 4.8*):

a) La surface passe derrière une silhouette, un bord ou un pli.

b) La surface croise une silhouette ou un pli.

c) c'est à la fois une silhouette ou un bord et une discontinuité de la surface.

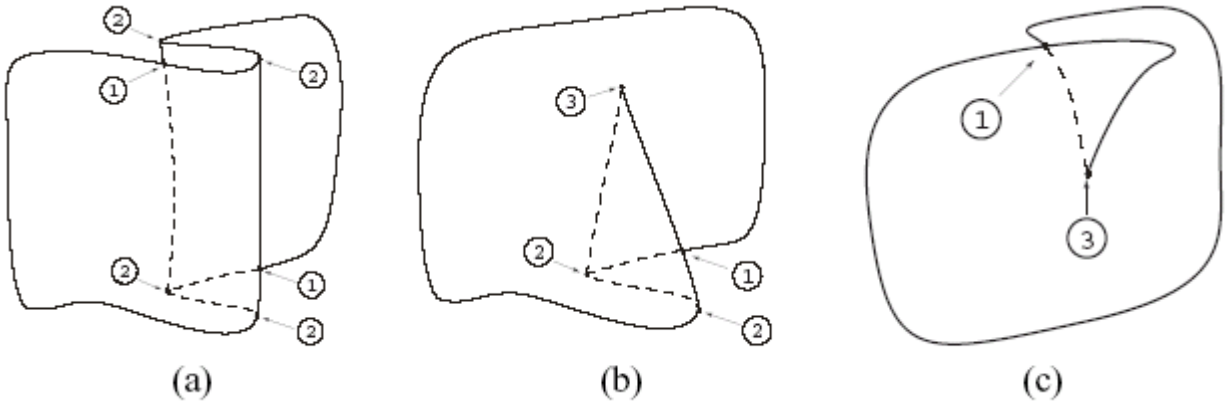


Fig. 4.8: 3 situations où la visibilité d'une surface courbe peut changer

Loin des algorithmes basés sur le calcul du Z-buffer, Nehab et Gatass ont proposé dans [NG00] une méthode complètement différente de détection des contours en utilisant la méthode de lancer des rayons pour la visualisation et en effectuant une segmentation des chemins des rayons en les divisant en des classes équivalentes.

Cette catégorisation des chemins des rayons (ray path catégorisation) est générée durant le processus du rendu et utilisée pour la détermination des contours dans l'image résultat.

Chaque pixel est représenté par son pixel bas gauche. Afin de déterminer les pixels qui forment les contours, chaque pixel est comparé à ces 3 voisins (les voisins a, b et c pour le pixel p dans *Fig. 4.9*) et il est classé comme un *pixel contour* s'il appartient à une classe différente de celle de ces voisins.

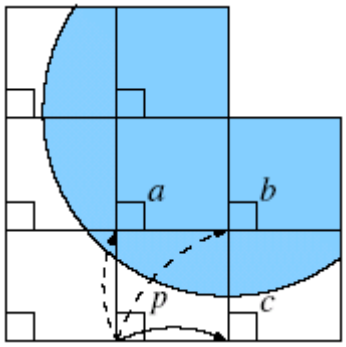
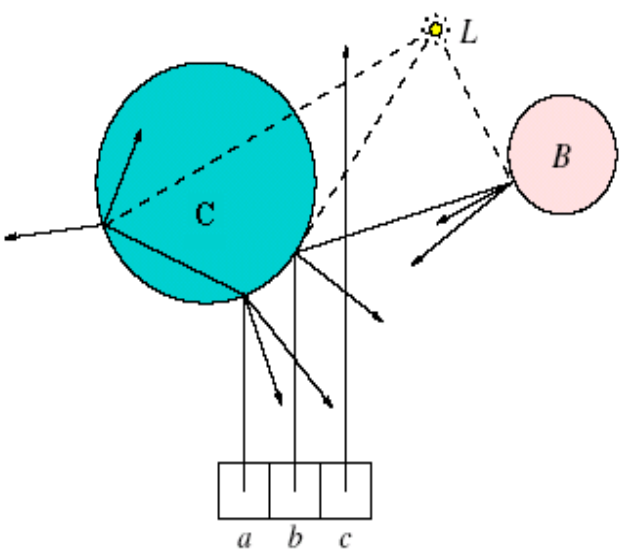


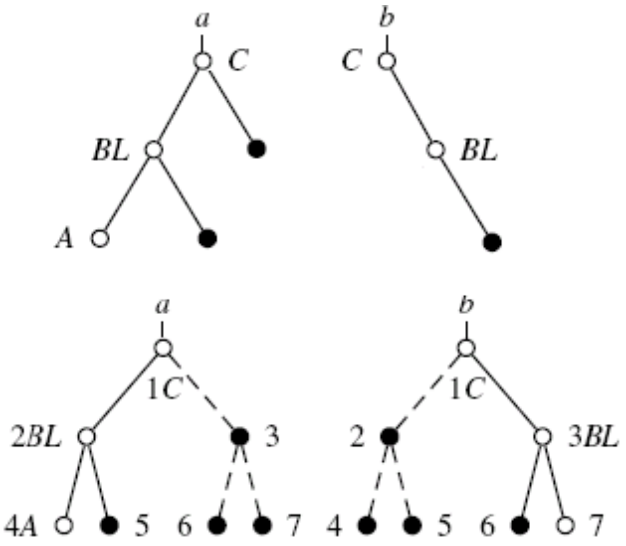
Fig. 4.9: Un pixel est jugé de contour lorsque sa catégorie est différente de celle de ses voisins.

Le chemin suivi par un rayon est représenté par une arbre (*Fig. 4.10(b)*)et peut être décrit comme suit (*Fig. 4.10(a)*):

- le rayon qui coupe l'objet
- les rayons lancés par la source lumineuse et visibles par l'objet coupé
- les rayons réfléchis et réfractés (branches enfants droits et gauches respectivement)



(a) Chemin suivi par un rayon



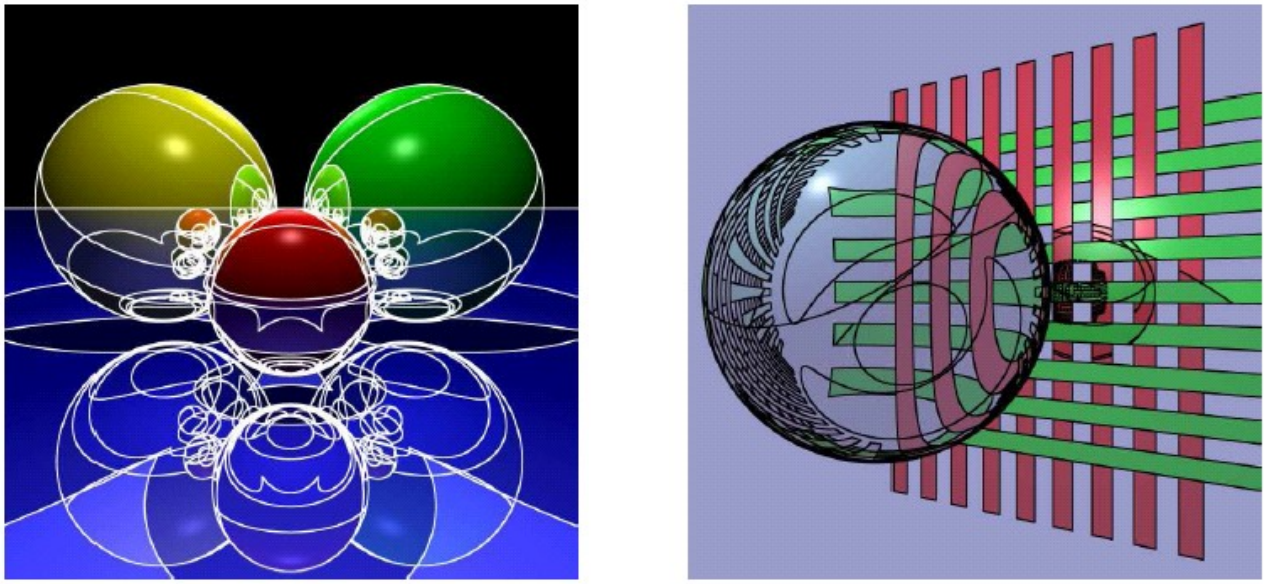
(b)Représentation d'un rayon par un arbre

Fig. 410: Description du chemin décrit par un rayon

Deux pixels appartiennent à la même catégorie si leurs arbres associés aux chemins des rayons sont égaux. Deux arbres sont égaux s'ils ont le même mot obtenu en joignant les caractères suivis des niveaux de profondeurs associés à chaque noeud. Par exemple, les deux mots obtenus par les arbres dans *Fig. 4.10(b)* sont les suivants: « a1C2BL4A » et « b1C3BL ».

Les résultats obtenus sont convaincants. Malheureusement, cette méthode donne de bons résultats, seulement, pour des cas de scènes simples et à partir de fréquences moyennes dans l'antialiassage. Les hautes fréquences présentes dans la scènes sont mal apparues et les basses fréquences apparaissent gênantes. Pour améliorer la qualité de l'image il est peut être possible de faire un sur-échantillonnage (oversampling) de toute l'image, ce qui est certainement très coûteux. Ajoutons aussi le coût qu'il faut dépenser pour la comparaison des mots.

Avec des scènes complexes, la méthode peut échouer surtout lorsque la scène représente beaucoup de réflexions et de réfractions (*Fig. 4.11*). Les contours des objets réfléchis et réfractés sont aussi détectés par cette méthode, ce qui nuit à la compréhension de la scène.



(a) détection des contours des objets réfléchis. *(b) détection des contours des parties réfractées*

Fig. 4.11: Méthode de catégorisation des rayons.

II.4. Avantages et inconvénients

Les techniques d'extractions des contours que nous avons discutées ci-dessus ont leurs avantages et leurs inconvénients. En effet :

Les algorithmes de l'espace image ont l'avantage d'être relativement faciles à implémenter, et donnent de bons résultats dans les cas de scènes simples puisqu'ils opèrent sur des tampons qui peuvent être générés facilement avec les matériels graphiques existants actuels. En plus, ils peuvent être appliqués à n'importe quelle scène indépendamment de la façon dont elle est modélisée.

Cependant, ces techniques sont limitées et insuffisantes parce que, d'une part, elles dépendent de certaines valeurs qui demandent des ajustements de la part de l'utilisateur, et d'autre part imprécises puisqu'elles ne détectent pas parfois tous les contours ou ajoutent dans certain cas des détails dus aux ombres, réflexion et réfraction ce qui est gênant à la compréhension des scènes.

Les méthodes qui opèrent sur la modélisation 3D de la scène sont acceptables mais elles souffrent tout d'abord d'être coûteuses en temps de calcul et en mémoire. Certaines d'entre elles présentent aussi des problèmes dans l'antialiasing en donnant de faux résultats avec des hautes ou basses fréquences. Mais le problème le plus important et qui existe aussi chez les algorithmes de l'espace image est qu'elles détectent les contours des objets réfléchis et réfractés, ce qui constitue notre problème majeur qui empêche la compréhension et la prise de connaissance des scènes réalistes complexes.

Nous ne sommes pas intéressés aux techniques hybrides citées ci-dessus puisqu'elles donnent des résultats plutôt stylistiques et artistiques que réalistes. Cependant nous allons dans le paragraphe suivant proposer une nouvelle technique hybride permettant de donner de solutions satisfaisantes à certains problèmes de prise de connaissance des scènes 3D.

III. Nouvelle approche de détection des contours

Nous définissons une scène complexe comme étant une scène qui comporte beaucoup de lumières, miroirs, et objets transparents (*Fig. 4.12*). De telles scènes sont généralement difficiles à comprendre par un rendu réaliste à cause de la présence des ombres, réflexions et réfractions qui peuvent donner des illusions et confondre l'observateur.

Afin de résoudre ce problème, nous proposons d'entourer les objet réels par leur contours apparents pour aider l'observateur ou l'utilisateur de cette scène à pouvoir distinguer entre ce qui est réel et ce qui est illusoire provenant des effets réalistes comme la réflexion et la réfraction.

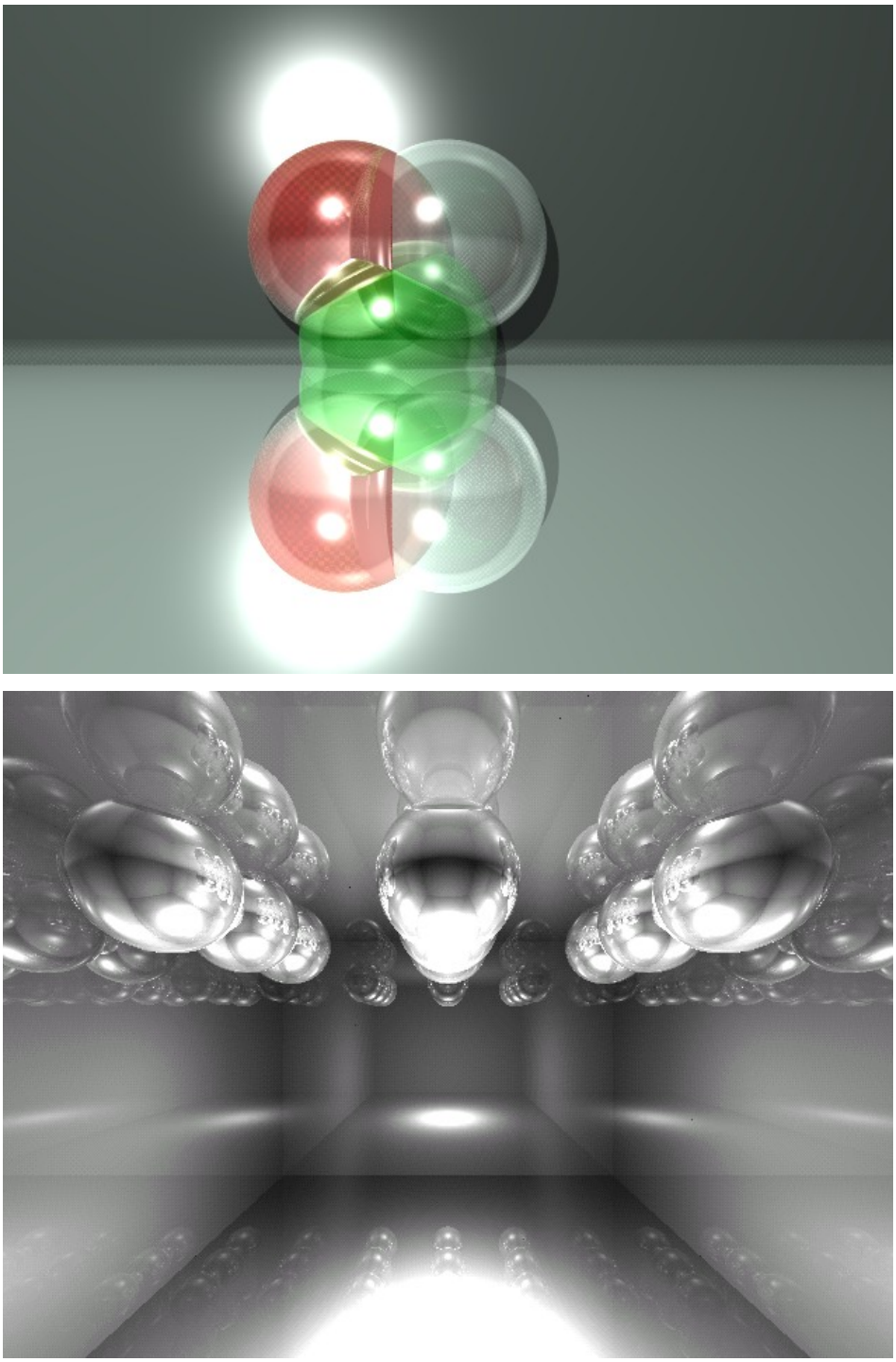


Fig. 4.12 : Exemples de scènes 3D complexes.

Notre approche de détection des contours apparents entre dans la zone des algorithmes hybrides et elle est décrite en détail dans [DPB07]. Elle tend à chercher dans l'espace image, les pixels qui définissent les contours de chaque objet réel présent dans la scène alors que les manipulations et les calculs se fond tous dans l'espace objet. L'algorithme est formé de deux parties:

1. La technique du raffinement sélectif [PLE91] qui tend à chercher dans l'espace image et pour chaque objet réel de la scène, un pixel initial appartenant au contour (*Fig. 4.13*).

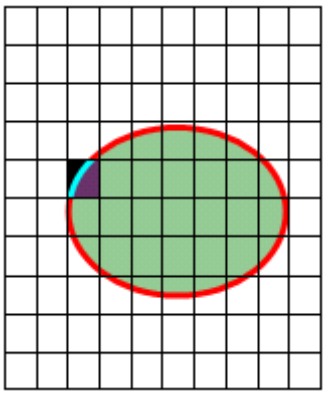


Fig. 4.13: Recherche d'un pixel contour initial pour chaque objet.

2. La technique de suivi du contour par la méthode de code de direction qui cherche pour chaque objet réel de la scène, le reste des *pixels contours* (*Fig. 4.14.*).

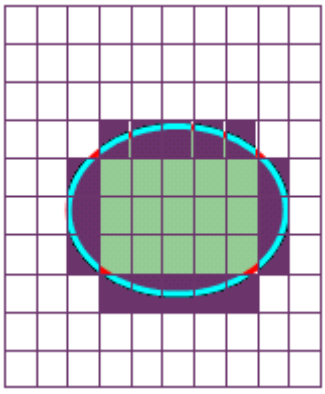


Fig. 4.14: Recherche du reste des pixels du contour.

III.1. Raffinement sélectif

Notre but dans cette partie est de chercher pour chaque objet réel présent dans la scène, d'un *pixel contour* initial qui va être utilisé comme un point de départ dans la partie suivante.

Tout d'abord, nous divisons l'espace image en des macro pixels de $2^n x 2^n$ pixels. Dans la pratique, l'expérience a montré que pour la majorité des scènes, des macros pixels de $2^3 x 2^3$ pixels permettent d'obtenir des résultats tout à fait satisfaisants et évite la disparition des petits objets.

L'étape suivante consiste à choisir les pixels guides que nous devrons fixer pour chaque macro pixels. Comme le nombre des pixels guides devrait être, d'après [PLE91], supérieur à 3 afin d'éviter la disparition des morceaux des objets, nous avons choisi pour chaque macro pixel, 4 pixels guides: les pixels haut-gauche (HG), haut-droit (HD), bas-gauche (BG) et bas-droit (BD). Nous lançons ensuite à chaque macro pixel, un rayon vers ces 4 pixels guides (*Fig. 4.15*). Nous récupérons alors pour chaque rayon lancé, le numéro de l'objet le plus proche visible par le point de vue.

Le numéro de l'objet obtenu sera ensuite associé au pixel correspondant comme un identificateur (Id) du pixel. Si le rayon ne coupe aucun objet, l'identificateur associé au pixel est -1.

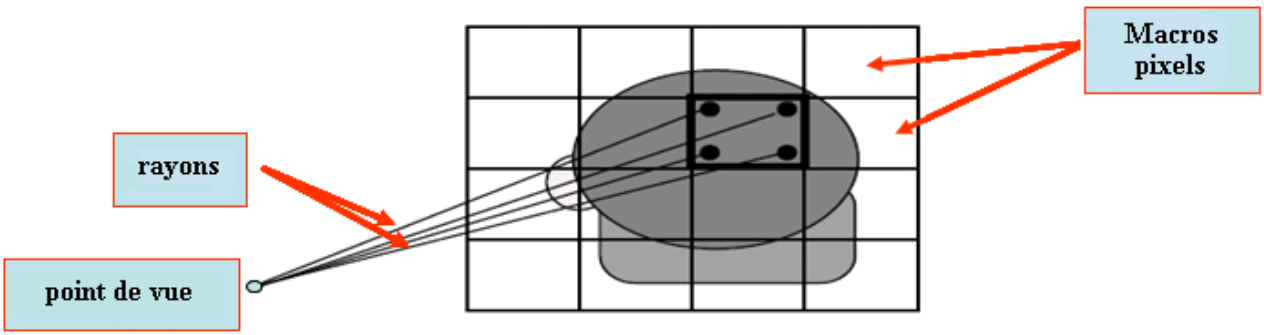


Fig. 4.15 : Lancer un rayon vers chacun des pixels HG, HD, BG, BD de chaque macro pixel.

Les macros pixels qui représentent différentes intersections doivent contenir un *pixel contour*. Ce sont alors nos macros pixels utiles qui seront subdivisés en 4 sous macros pixels (*Fig.4.16*). Le même processus s'applique pour chaque macro pixel jusqu'à l'obtention de blocs de 2x2 pixels chacun.

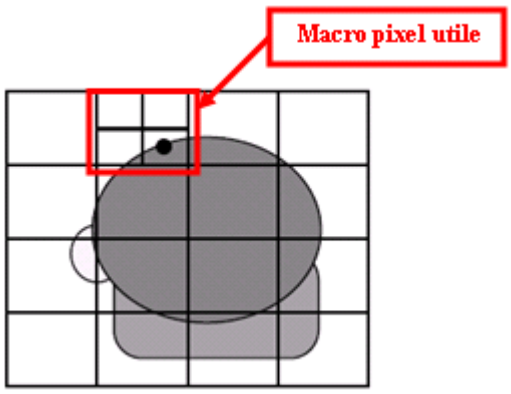
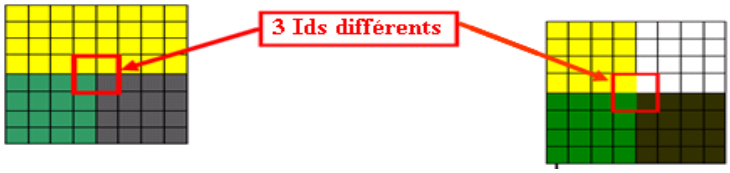


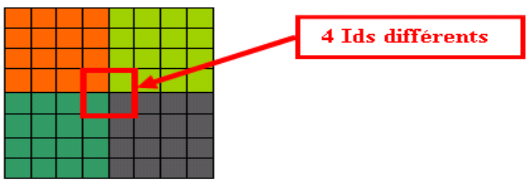
Fig. 4.16: subdivision des macro pixels utiles en 4 sous macros pixels.

Le bloc de 2x2 pixels dont les pixels admettent des intersections avec des objets différents doit essentiellement contenir un *pixel contour* au moins. Généralement, nous pouvons avoir 2, 3 ou 4 pixels différents dans un bloc (*Fig. 4.17)*. Plus nous avons des différentes intersections au niveau d'un seul bloc, plus il y a de *pixels contours* initiaux.

(a) 2 pixels contours initiaux



(b) 3 pixels contours initiaux



(c) 4 pixels contours initiaux

Fig. 4.17: Bloc de 2x2 pixels contenant des pixels contours initiaux.

Nous devrions avoir autant de *pixels contours* initiaux que d'objets dans la scène. Pour ne pas avoir plus qu'un *pixel contour* initial appartenant à un même objet, nous avons associé à chaque objet un tampon de visite qui est initialisé à zéro et sera mis à 1 une fois un *pixel contour* initial est trouvé pour l'objet courant. Une fois le tampon de visite d'un objet est rempli à 1, il suffit alors, de négliger tous les pixels qui admettent comme Id le numéro de l'objet courant.

III.2. Suivi du contour par code de direction

L'algorithme de suivi de contour commence au début par un point contour initial puis suit à chaque fois une certaine direction qui nous conduit vers le *pixel contour* suivant. Le même processus se répète jusqu'à l'obtention du contour complet d'un objet (*Fig. 4.18*). Ceci est applicable pour chaque *pixel contour* initial afin d'obtenir les contours de tous les objets.

Autrement dit, l'algorithme peut être appliqué en suivant les 3 étapes suivantes:

1. Choisir un point de départ
2. Choisir la direction initiale qui nous conduit au second point contour
3. Choisir la direction suivante qui nous conduit au point contour suivant

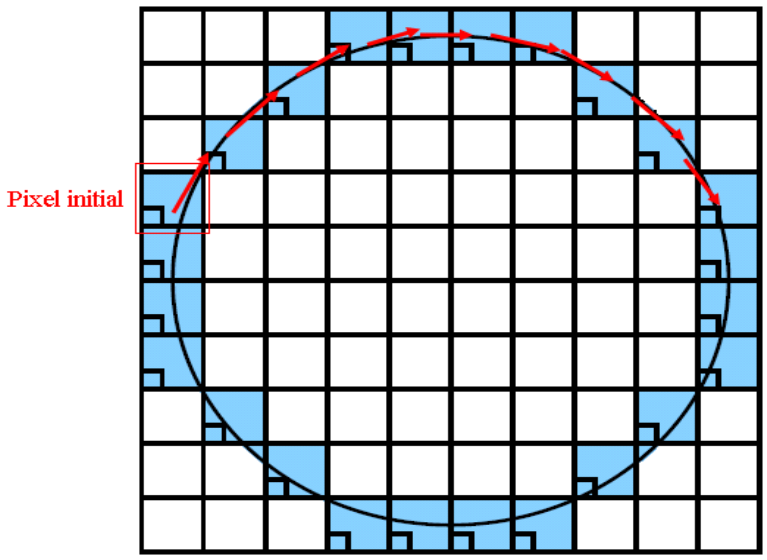


Fig. 4.18: Méthode de code de direction.

Avant de détailler les étapes de l'algorithme, nous allons définir pour chaque pixel ces 8 voisins indexés de 0 à 7 comme le montre *Fig. 4.19*. Nous définissons ensuite dans le voisinage d'un pixel courant et pour chaque pixel voisin, son pixel suivant et son pixel précédent respectant l'ordre indexé de 0 à 7. Par exemple, pour un pixel courant de coordonnées (x,y), considérons son voisin d'indice 1 et de coordonnées (x-1,y+1). Ce voisin admet comme précédent le pixel d'indice 0 de coordonnées (x,y+1) et le pixel d'indice 2 et de coordonnées (x-1,y) comme pixel suivant (*Fig. 4.20*).

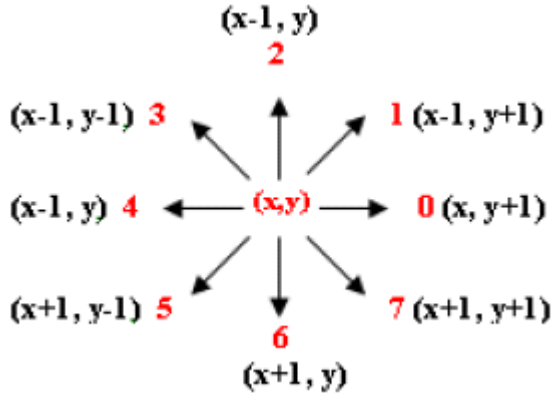


Fig. 4.19: Définition des voisins d'un pixel.

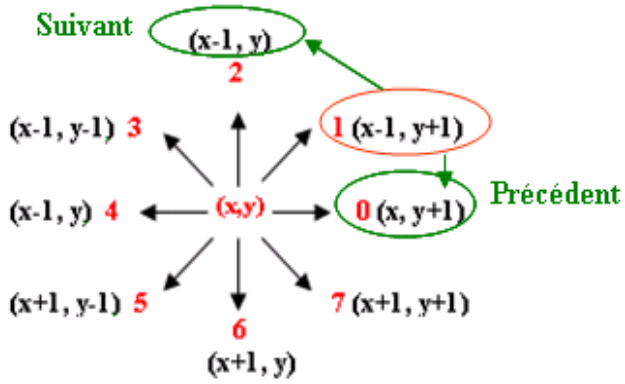


Fig. 4.20: Définition du pixel suivant et du pixel précédent d'un voisin

Étape 1: Choisir le point de départ

L'algorithme commence initialement par les *pixels contours* initiaux récupérés par la méthode de raffinement sélectif comme étant des points de départ pour cette étape. Plus il y a de *pixels contours* initiaux obtenus par le raffinement sélectif, plus il y a des contours qu'il faut détecter.
Avant de passer à la deuxième étape nous allons donner quelques définitions.

Premièrement, il est évident que tous les pixels définissant le contour d'un objet doivent avoir le même identificateur que celui du point de départ. Nous considérons alors qu'un pixel est un point de contour s'il admet tout d'abord le même identificateur (Id) que celui du pixel de départ.

D'autre part, puisque le point contour devrait être un point de séparation entre deux zones ou objets différents, il faut que son pixel suivant et son pixel précédent admettent des identificateurs différents. Sinon, le pixel courant est nécessairement à l'intérieur d'un objet.

Étape 2: Choisir la direction initiale qui nous conduit au second point contour

Le second *pixel contour* devrait être l'un des 8 voisins du pixel de départ. Nous commençons alors à lancer un rayon à chaque voisin. Le premier voisin qui admet le même identificateur (Id) que celui du point de départ et tel que son suivant et son précédent dans le voisinage du pixel de départ, admettent de différents Ids, sera notre second pixel de contour.

Une fois trouvé, le second pixel de contour sera notre pixel courant de l'étape suivante pour la recherche du pixel de contour suivant. Si tous les pixels du voisinage de pixel de départ sont testés et aucun parmi eux n'était un pixel de contour, nous arrêtons la recherche.

Étape 3: Choisir la direction suivante qui nous conduit au point contour suivant

Pour la recherche du *pixel contour* suivant, le même processus est appliqué mais cette fois-ci en considérant le *pixel contour* trouvé dans l'étape 2 comme étant un point de départ. À chaque fois qu'un nouveau *pixel contour* est trouvé, le pixel de départ sera mis à jour le et nous répétons le même processus.

Il se peut que pendant la recherche du *pixel contour* suivant, l'algorithme tombe sur un ancien *pixel contour* déjà visité pendant une ancienne itération. Pour éviter ceci, il suffit d'associer à chaque pixel un tampon de visite initialisé au départ à zéro. Dès qu'un pixel testé est jugé comme pixel de contour, le tampon de visite sera mis à 1.

Le test du pixel voisin et suivant du pixel courant n'est alors appliqué qu'aux pixels qui n'ont pas encore été visités. Ce qui peut servir d'éviter le retour rapide vers le pixel de départ tout en économisant le temps de calcul et la mémoire.

L'algorithme arrête la recherche lorsqu'il tombe dans l'un de ces deux cas:

- retour au point de départ initial. Dans ce cas là, le contour de l'objet trouvé est un contour fermé

- Aucun des voisins d'un pixel courant n'est un pixel de contour et alors dans ce cas là le contour détecté est un contour ouvert.

III.3. Résultats

La méthode décrite dans ce qui précède a été implémentée et a donné des résultats permettant de conclure qu'il est possible d'améliorer la visualisation d'une scène afin de mieux comprendre des scènes complexes qui contiennent des réflexions, des réfractions et des ombres en entournant les objets réels par leurs contours apparents.

Dans nos test, nous avons utilisés 8 scènes, numérotées de 1 à 8. Les images rendues des scènes 7 et 8 sont de dimensions 248x248 pixels tandis que les autres sont de dimensions 640x480 pixels. Les contours apparents sont tracés en rouge afin de distinguer les objets réels présents dans la scène. Les résultats obtenus sont présentés dans les figures qui suivent et elles représentent différents cas.

Les scènes 1 et 2 représentent le même objet avec, puis sans réflexions et réfractions. Nous avons obtenu pour tous les deux le même contour et avec le même coût de temps. Ceci est dû au fait que notre algorithme calcule le contour directement de la scène et non pas de son image indépendamment de la façon dont la scène est rendue.

D'autre part, la scène 1 n'est pas visuellement bien comprise et peut confondre l'observateur puisqu'il n'est pas clair s'il s'agit au sol d'un objet sombre ou de l'ombre d'un objet. En traçant les contours des objets réels de la scène, nous avons pu effacer le doute en séparant les ombres sur le sol des objets réels de la scène de la réflexion de l'ombre sur la sphère.

Les scènes 3 et 4 montrent la performance de notre algorithme en cas d'intersection, ainsi que sa capacité de détecter les silhouettes, les bordures ainsi que les lignes d'intersections visibles malgré la présence dans Scène 3 et 4 de toutes les illusions aux murs dues aux réflexions et réfractions.

Finalement les figures *Fig. 4.29* et *Fig. 4.31*, montrent la performance de notre algorithme dans la séparation entre les objets réels et les objets réfléchis et réfractés malgré la présence énorme dans les scènes 5 et 6 des effets miroirs, des réflexions et des réfractions qui sont très gênants et distrayants pour l'observateur.

Les coûts en temps d'exécution de l'algorithme pour le calcul du rendu des scènes ainsi que pour le calcul des contours sont présentés ci dessous par le tableau 1*. Les tests ont été fait sur un processeur standard de type Intel Pentium III ayant une capacité de 866 MHZ avec une mémoire RAM de 256 MB.

Scène	**Temps pour détecter le contour apparent des objets réels/Temps pour le calcul du rendu de la scène**	**Temps pour détecter la silhouette des objets réels /Temps pour le calcul du rendu de la scène**
Scène 1	0.93%	0.93%
Scène 2	0.98%	0.98%
Scène 3	1.74%	1.74%
Scène 4	0.56%	0.56%
Scène 5	2.18%	1.65%
Scène 6	4.63%	3.27%
Scène 7	69.14%	41.71%
Scène 8	57.47%	68.72%

Tableau 1: le temps nécessaire pour le rendu de la scène et pour la détection des contours et des silhouettes des objets réels de la scène.

* « mn » signifie le temps en minute et « s » signifie le temps en second

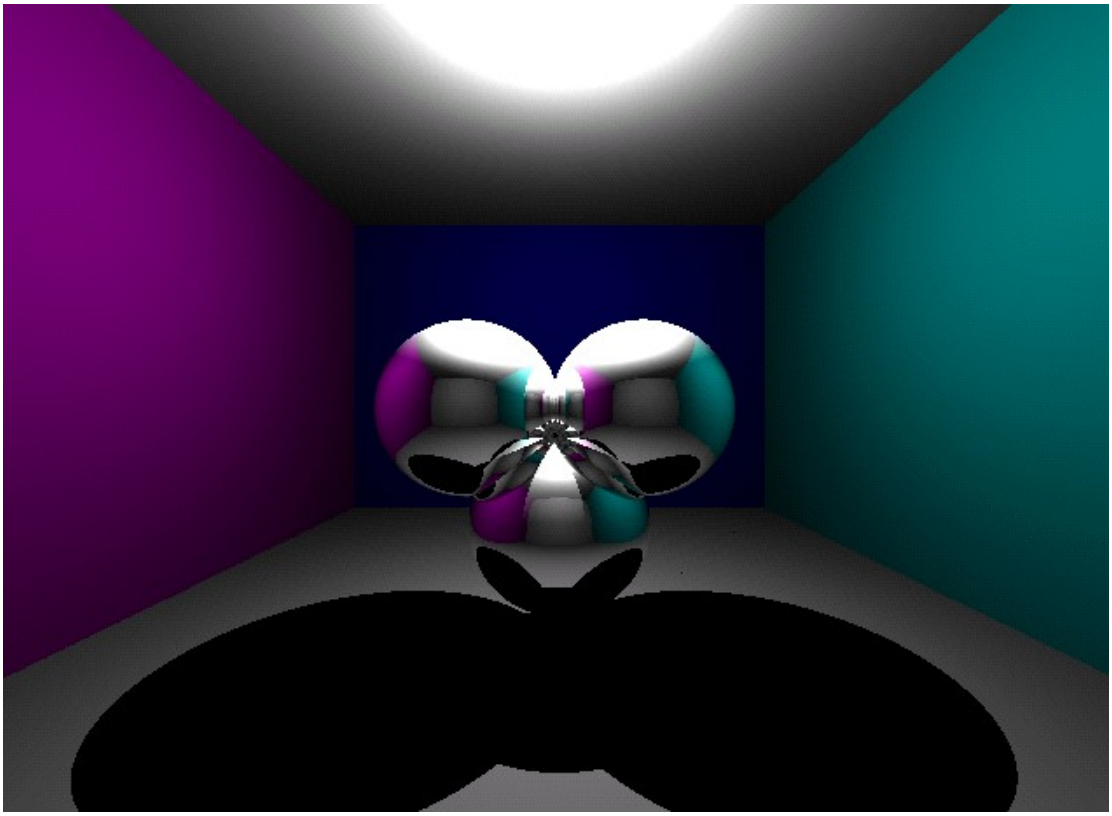


Fig.4.21:Scène 1.

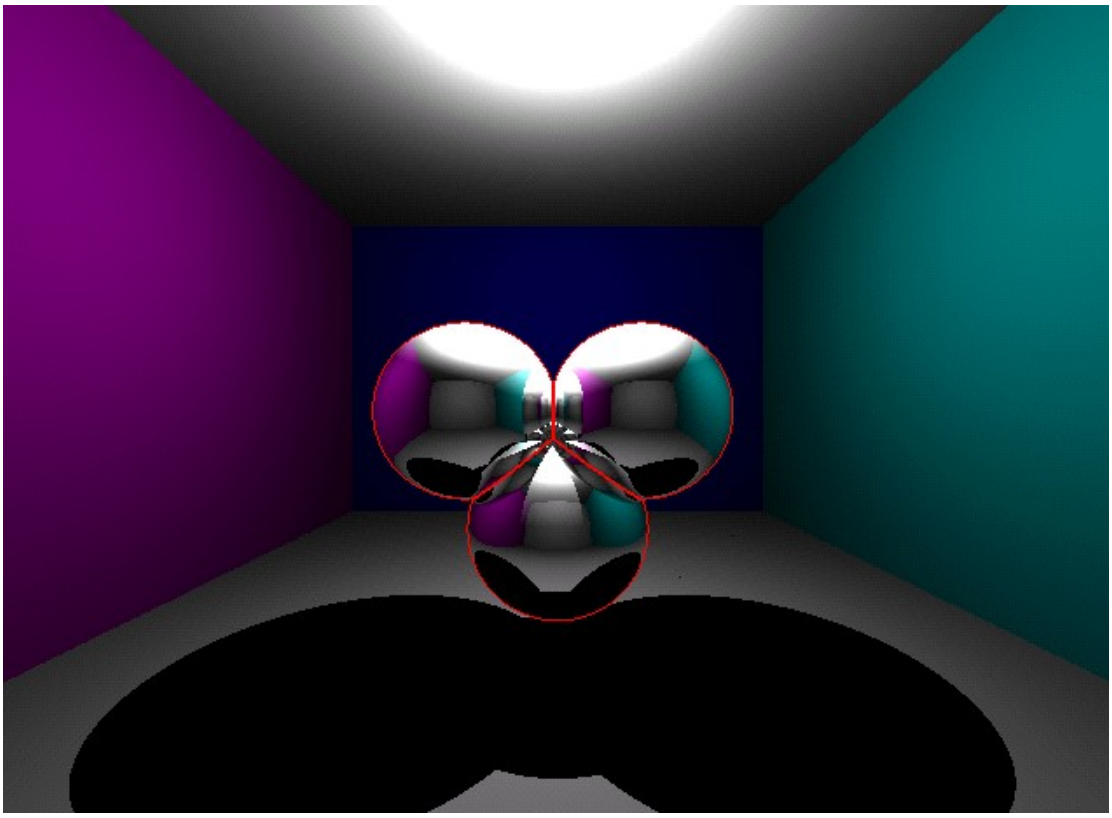


Fig. 4.22: Contour de Scène 1.

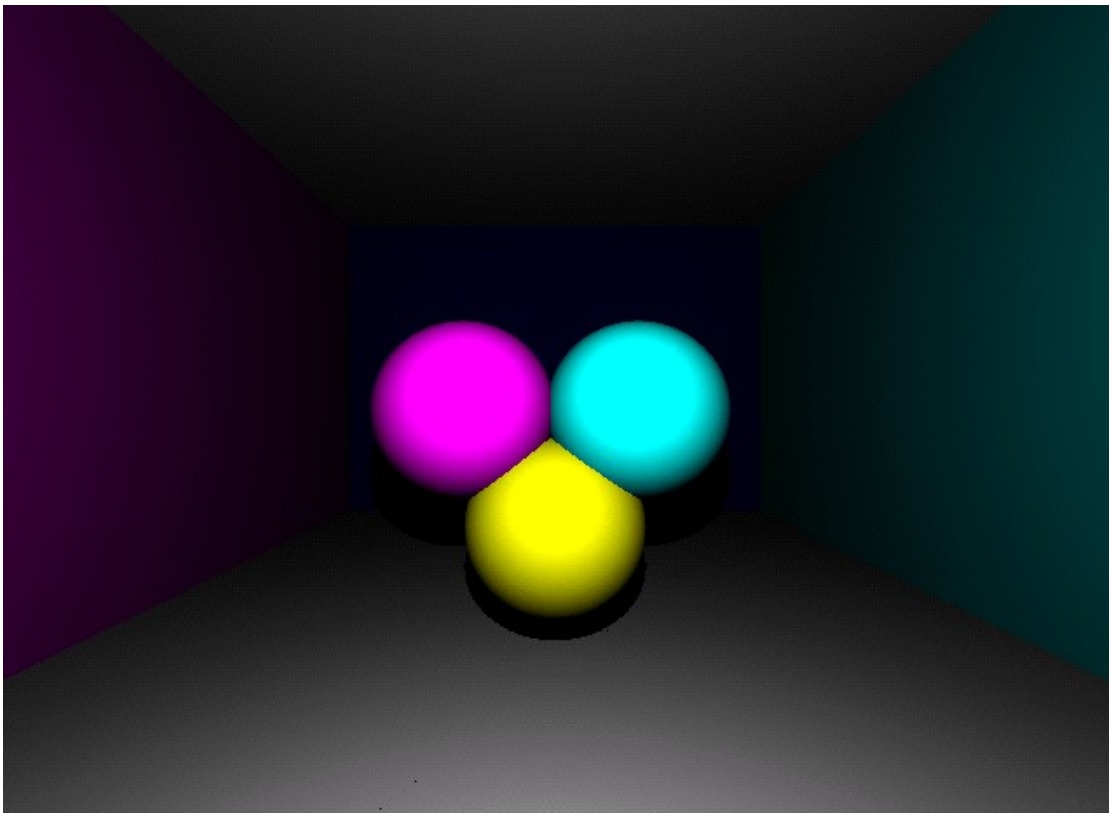


Fig. 4.23: Scène 2.

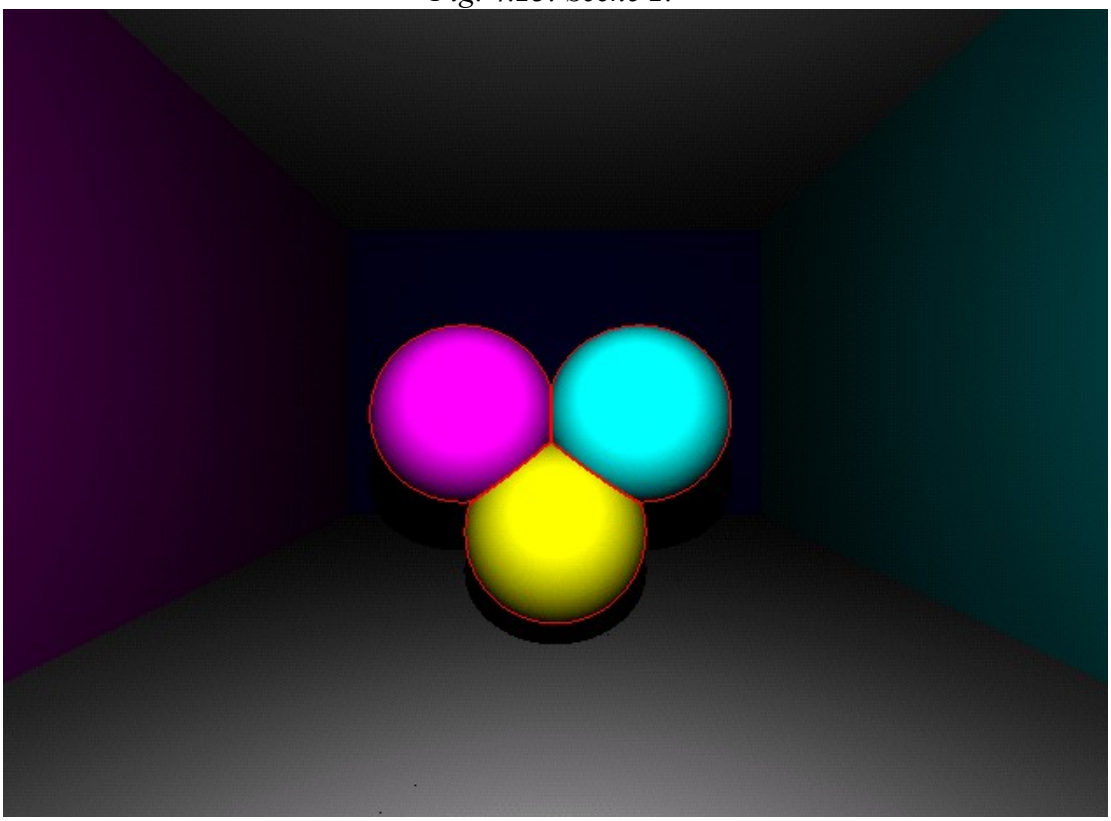


Fig. 4.24: Contour de Scène 2.

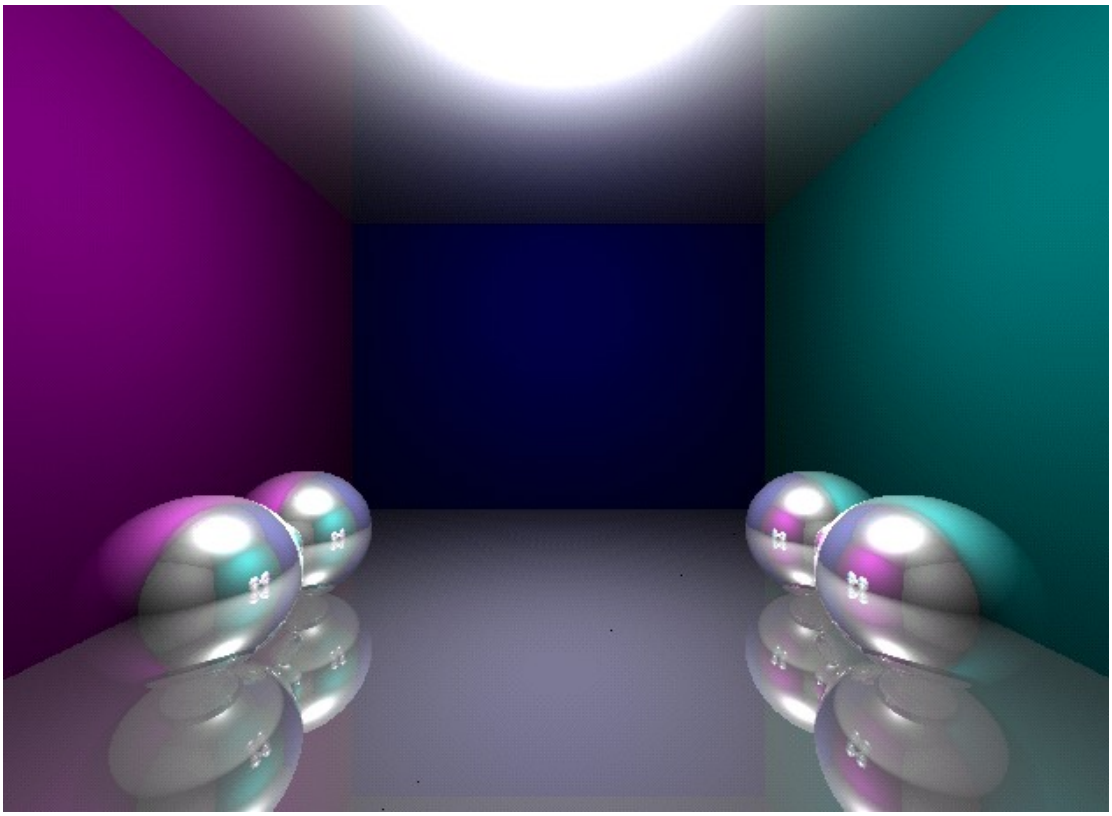


Fig. 4.25: Scène 3.

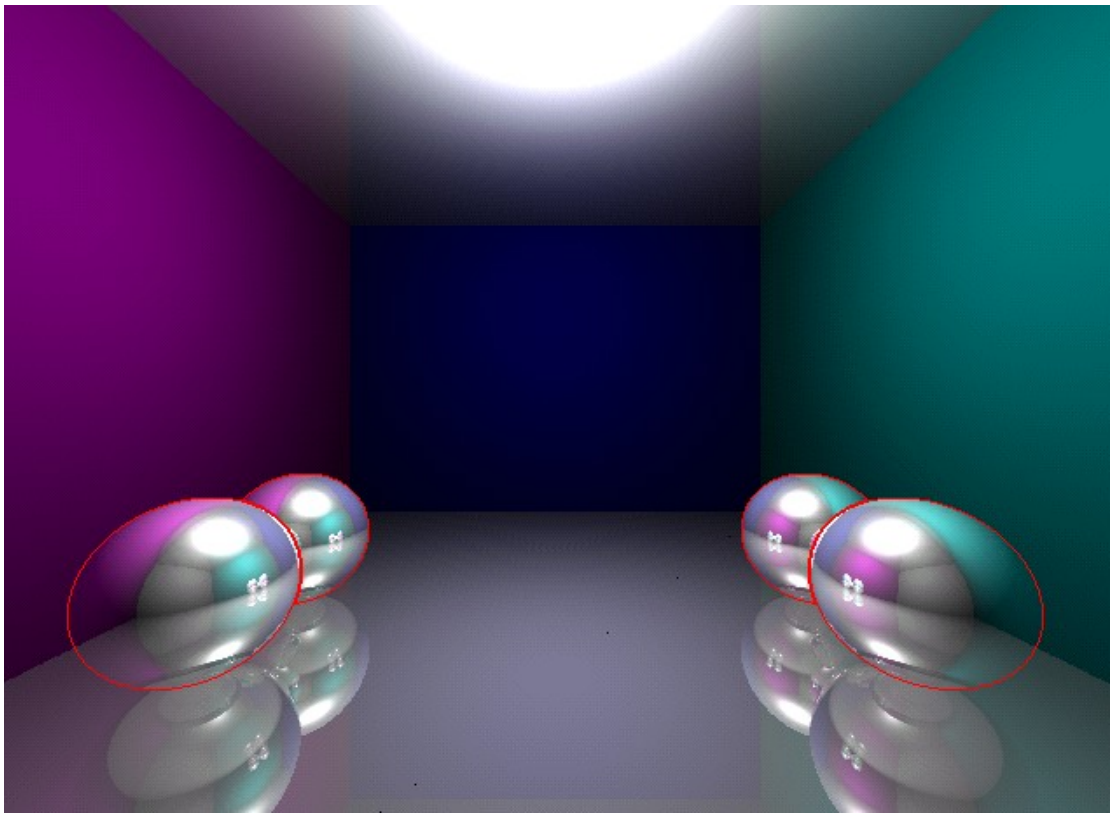


Fig. 4.26: Contour Scène 3.

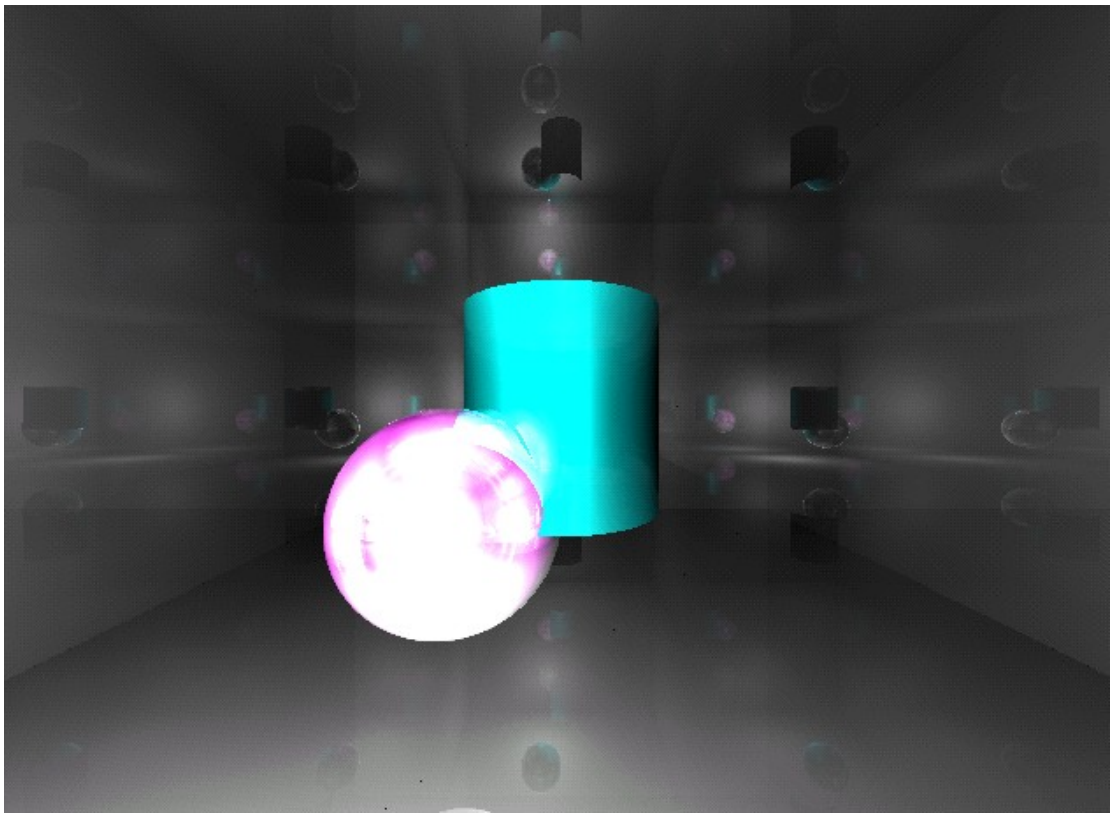


Fig. 4.27: Scène 4.

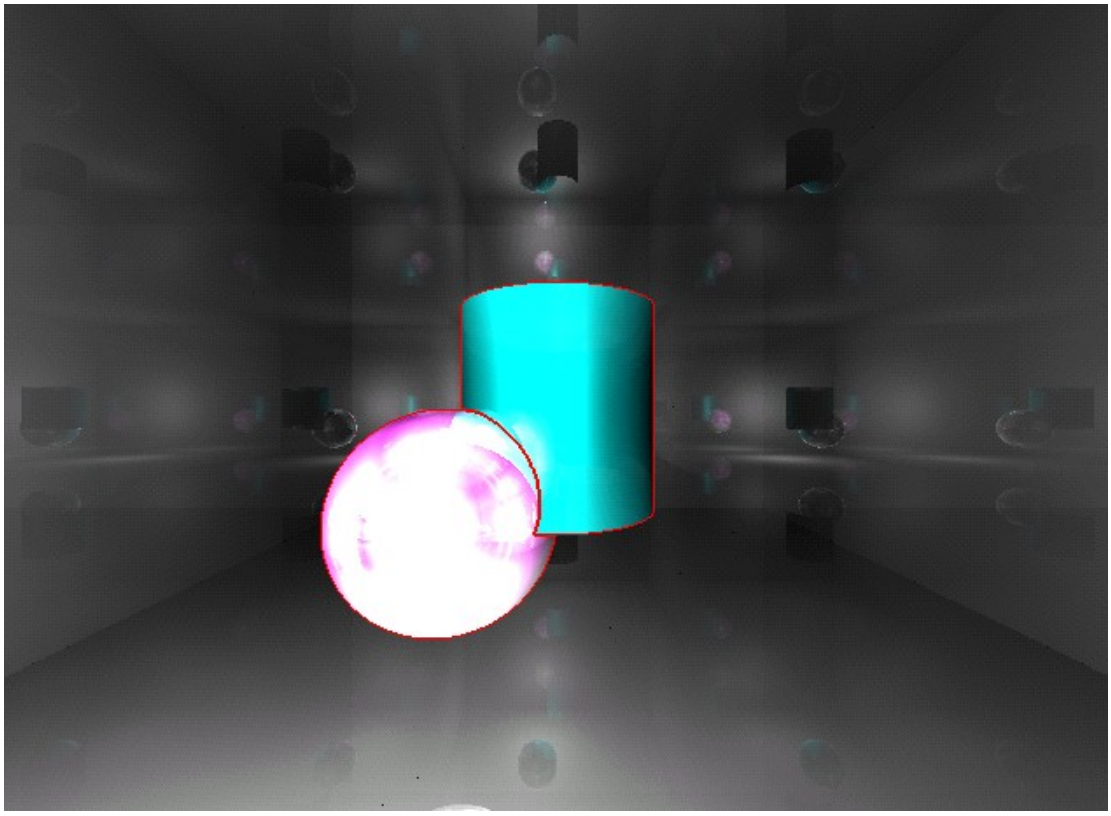


Fig. 4.28: Contour Scène 4.

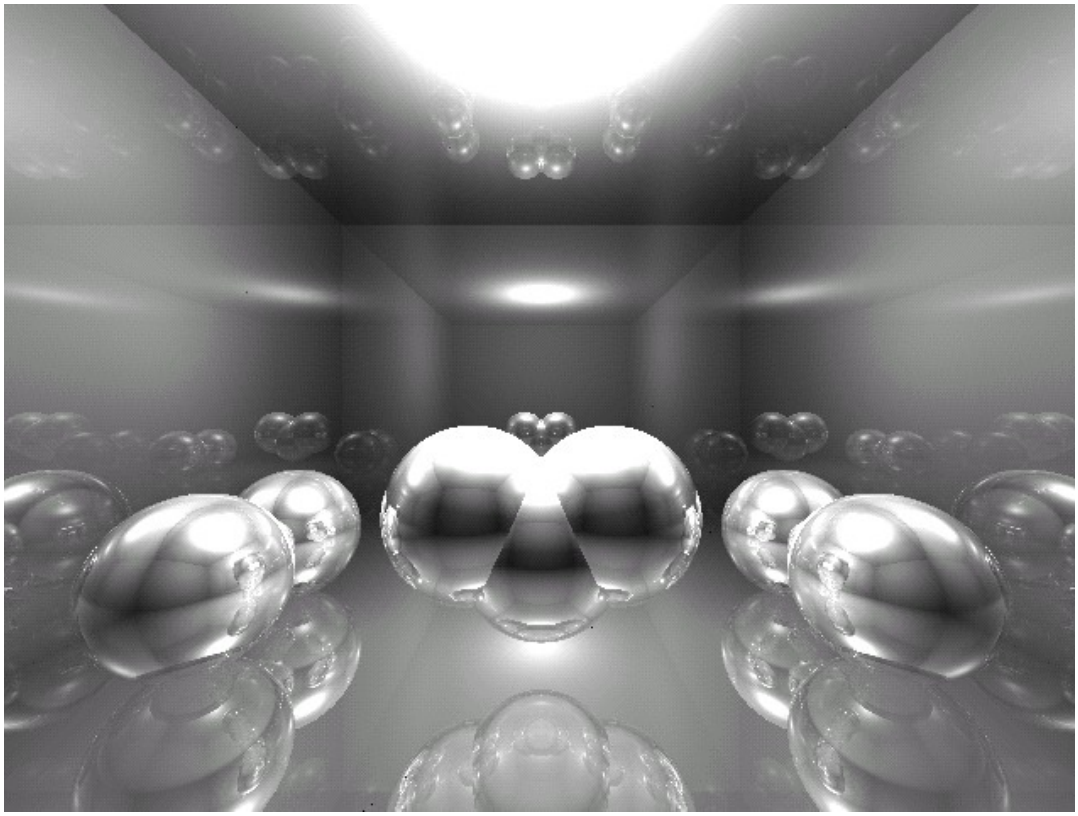


Fig. 4.29 Scène 5

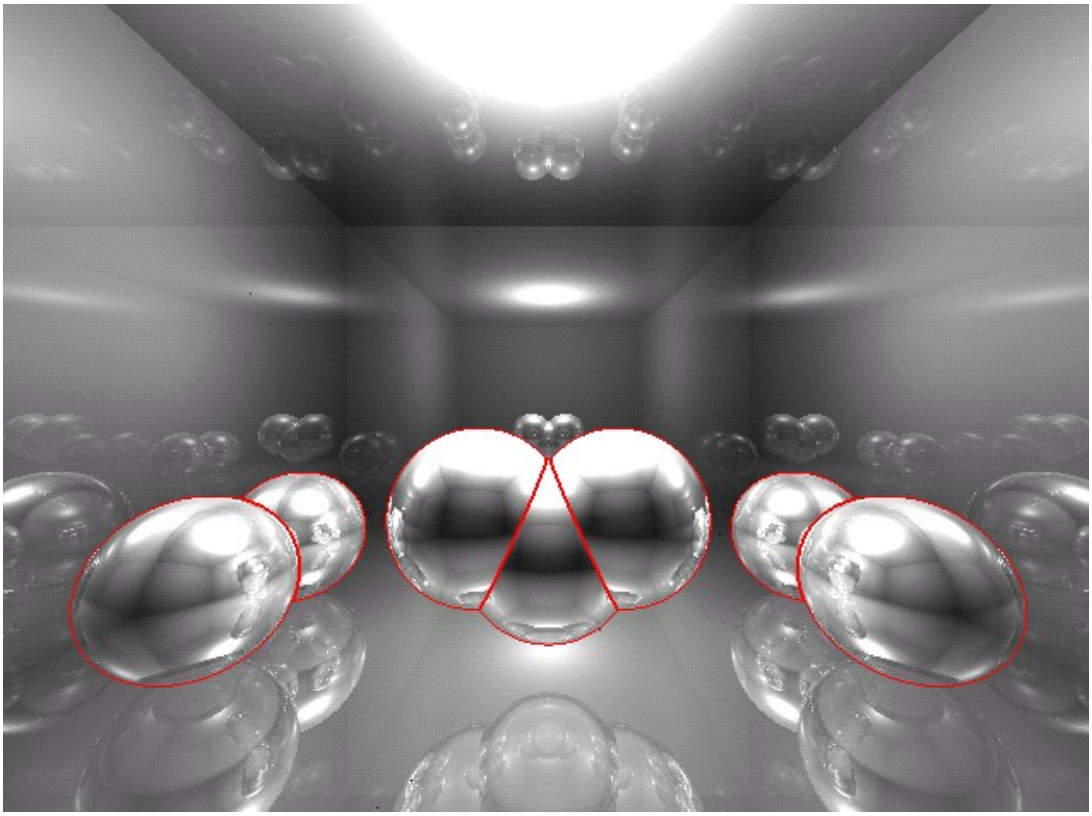


Fig. 4.30 Contour Scène 5

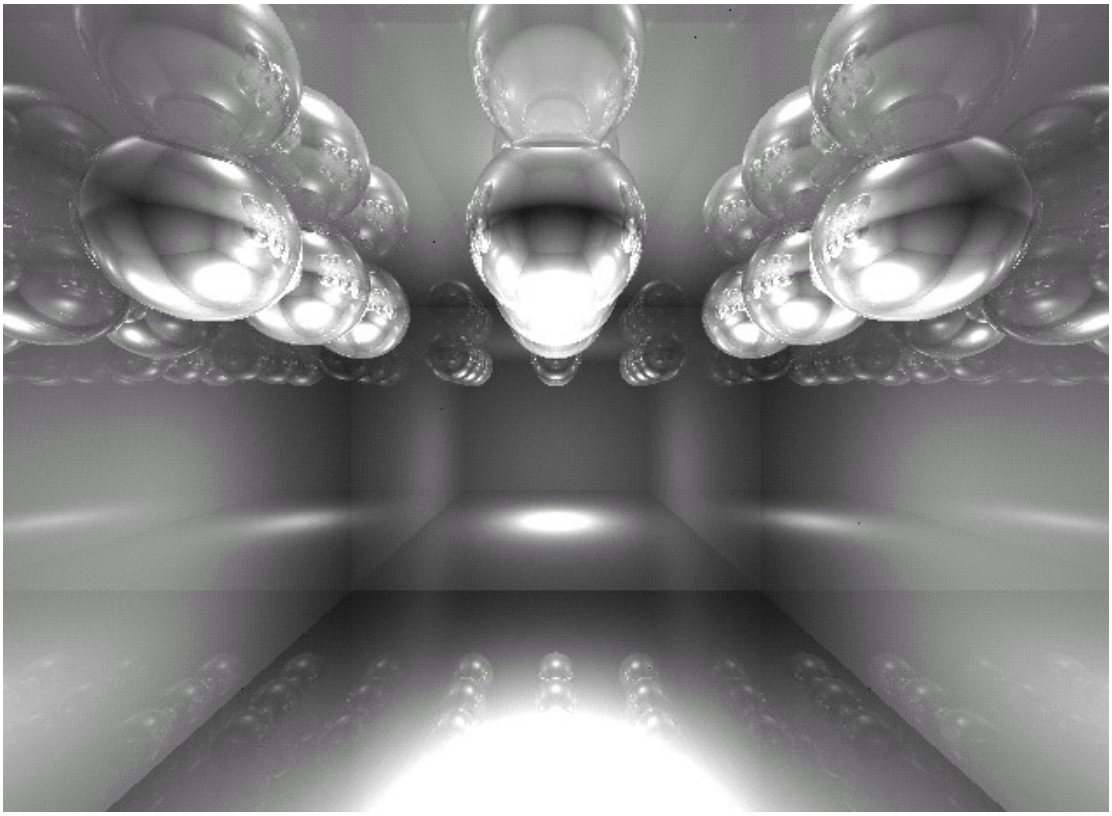


Fig. 4.31 Scène 6

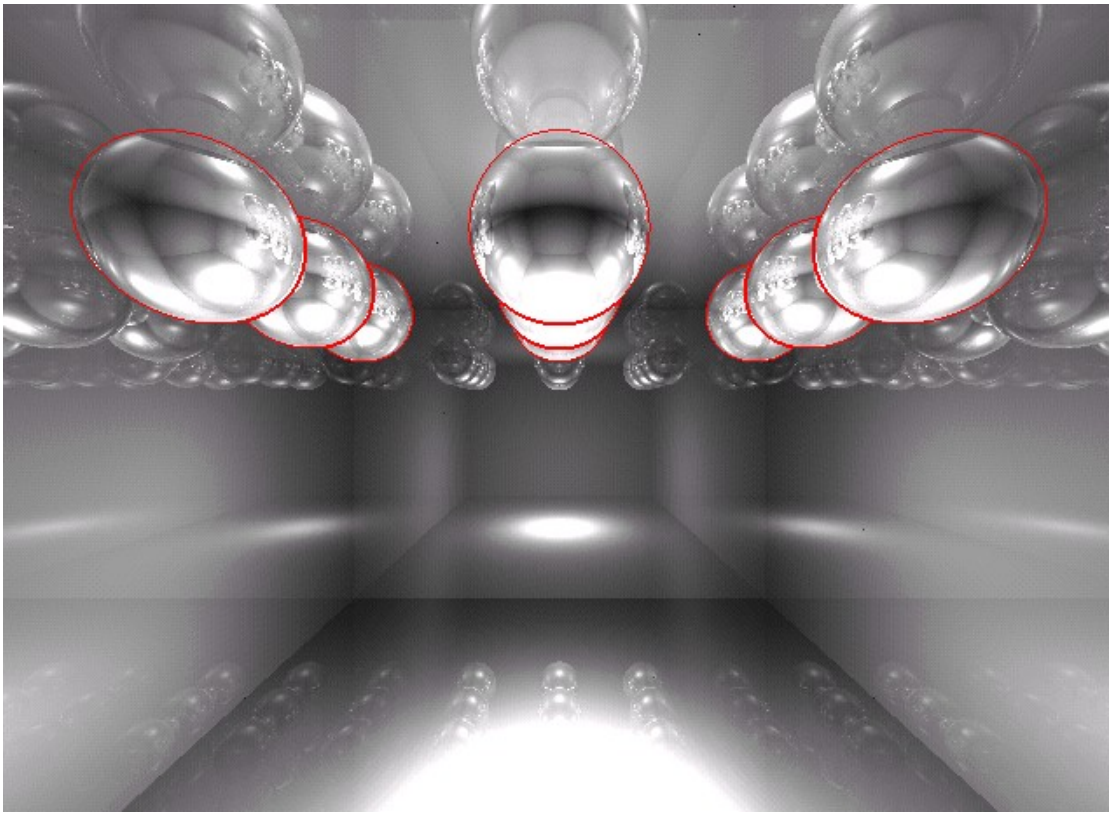


Fig. 4.32 Contour Scène 6

Cette méthode peut être appliquée à n'importe quel type de scènes et à n'importe quelle modélisation, que ce soit une scène modélisée par des polygones, définie par une description analytique et même pour une scène mixte. Elle peut servir aussi à régler le problème de l'anti-aliassage puisque le contour obtenu est de l'ordre d'un seul pixel (*Fig. 4.33*).

Fig. 4.33: le contour obtenu règle le problème de l'anti-aliassage.

Comme il est présenté dans *Fig. 4.34*, la scène 7 est modélisée par des facettes polygonales et elle est formée de 1056 sommets et 2088 polygones. Il s'agit d'une sphère contenant en son centre un autre objet qui a la forme d'un petit cube.

La présence d'une lumière proche de l'objet et orthogonale au centre a empêché de visualiser le cube d'une part, et d'avoir une idée suffisante sur le volume de la sphère à cause de la présence des ombres cachant la frontière de la sphère. La détection des contours de la scène a donc amélioré la compréhension de la scène et a permis de voir plus de détails qui ont été omis à cause d'un éclairage mal placé.

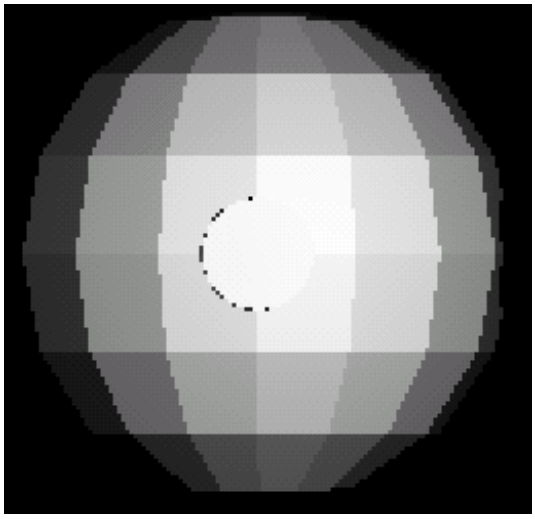


Fig. 4.34:Scène 7

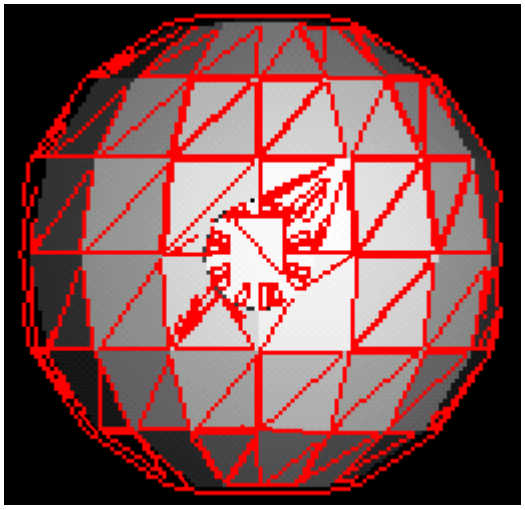


Fig. 4.35: Contour Scène 7.

Cependant, lorsque le nombre de polygones augmente, la visualisation des contours de tous les polygones peut devenir gênante et aboutir à une mauvaise prise de connaissance comme le montre l'exemple de la scène 8 qui est formé de 4052 sommets et 8100 polygones (*Fig. 4.36*). La taille des polygones est très petite et ils sont éloignés par rapport au point de vue ce qui a donné comme résultat un mauvais contour. Afin d'obtenir une solution meilleure que celle dans *Fig. 4.37,* il faut rapprocher la position du point de vue, ainsi que la position du plan de projection, et augmenter la taille de l'espace image. Ce qui est certainement coûteux en mémoire et temps de calcul.

D'autre part, chaque polygone est traité par notre algorithme comme étant un objet de la scène. Ce qui oblige l'algorithme à détecter les contours des polygones et non pas ceux de l'objet formé par ces polygones, ce qui n'est pas toujours efficace pour la compréhension de la scène.

La meilleure solution dans ce cas là est de détecter juste la silhouette de l'objet en éliminant tous les autres contours intérieurs qui sont inutiles et empêchent d'avoir une bonne prise de connaissance de la scène. La silhouette des objets n'est autre que la ligne qui joint les pixels séparant deux objets différents ou un objet du fond de l'image.

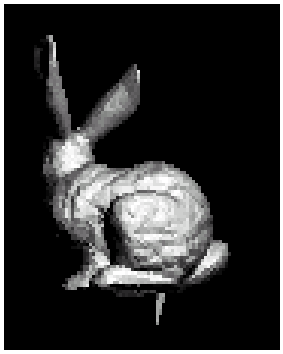


Fig. 4.36: Scène 8

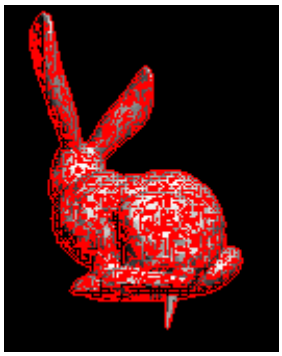


Fig. 4.37: Contour Scène 8

Pour détecter la silhouette séparant les objets du fond, il suffit de prendre les *pixels contours* dont l'un des deux pixels précédent et suivant appartient au fond (Id = -1) et l'autre appartient à l'objet (admet l'Id de l'objet). Alors que pour détecter les contours d'intersections entre deux objets différents, chacun des deux pixels, suivant et précédent, doivt avoir un Id différent. L'un admet l'Id du premier objet et l'autre admet l'Id du deuxième (*Fig. 4.38*).

Fig. 4.38: Silhouette de Scène 8.

Il est bien clair que la visualisation de la silhouette de l'objet est une meilleure solution pour ce type de scène puisqu'il y aura une mise en valeur de l'objet définie par la modélisation polygonale permettant ainsi une élimination de tout ce qui peut être gênant à la compréhension de la scène.

Mais, pour une scène modélisée par des facettes polygonales représentant plusieurs objets qui sont incapables d'être définis séparément, la détection de la silhouette de chaque objet à part devient impossible.

Comme par exemple avec la scène 7, notre algorithme échoue pour détecter le contour apparent de chaque objet séparément parce que nous ne sommes pas capables de savoir quels polygones appartiennent à chacun des deux objets (la sphère et le cube).

Le cube fait partie de la sphère, il est donc impossible de savoir quels sont les polygones qui le constituent. Il est alors juste possible de détecter la silhouette séparant l'objet du fond de l'image. Ce qui n'est pas tout à fait suffisant pour ce type de scène (*Fig. 4.39*).

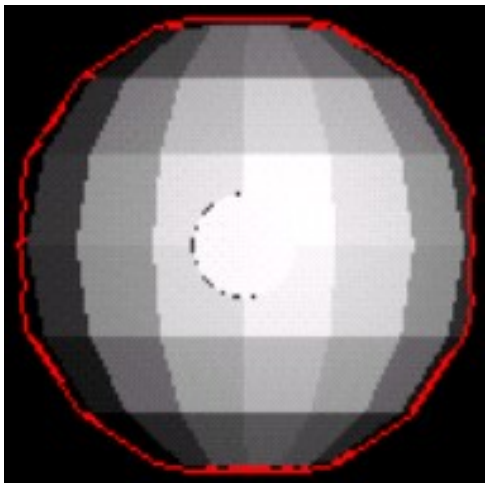


Fig. 4.39: Silhouette de Scène 7.

Pour une scène mixte (*Fig. 4.40*), définie par des objets modélisés par des facettes polygonales et d'autres objets représentés par leurs descriptions analytiques, l'algorithme réussit à détecter tous les contours apparents de chacun des deux objets ainsi que les contours d'intersections (*Fig. 4.41*). Il est capable aussi de détecter la silhouette de chacun à part sans avoir de problème puisque chaque objet est défini séparément. (*Fig. 4.43*)

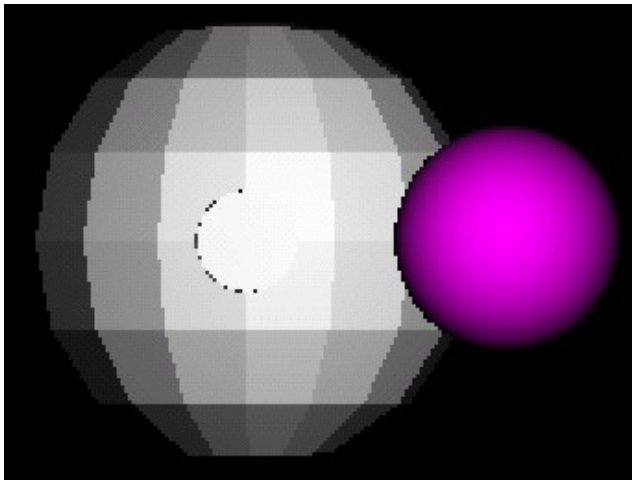


Fig. 4.40: Exemple d'une scène mixte.

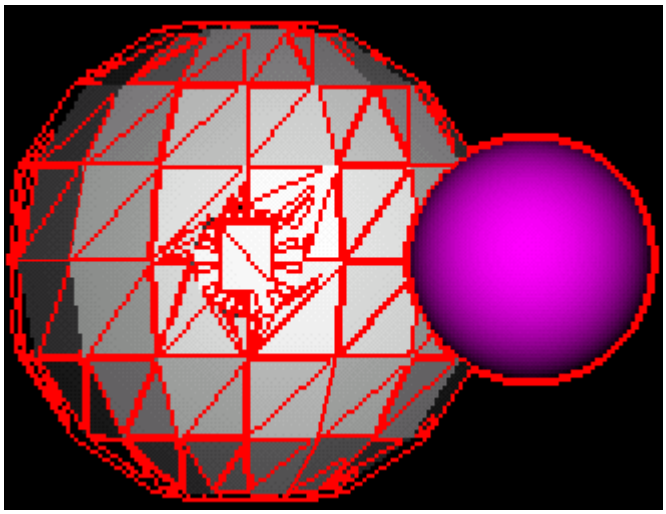


Fig. 4.41: Détection des contours apparents de la scène mixte .

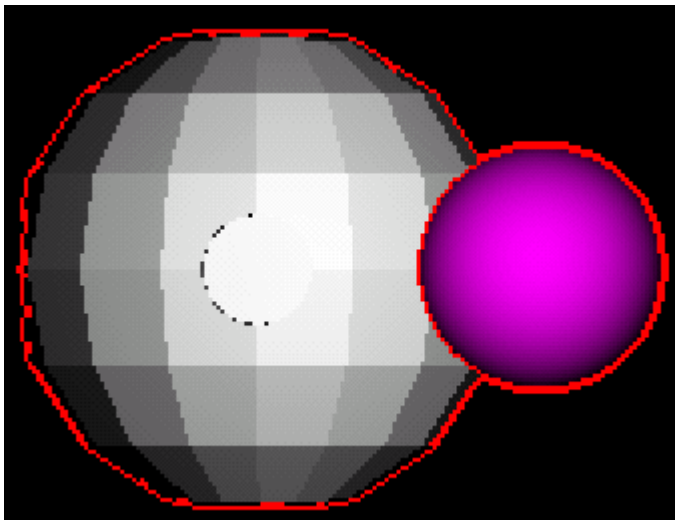


Fig. 4.42: Détection de la silhouette des objets de la scène mixte.

IV. Conclusion

Dans ce chapitre, après avoir présenté les techniques principales de détection des contours, nous avons présenté une méthode permettant visuellement de comprendre des scènes complexes par extraction des contours apparents des objets réels de la scène. Ce type de méthode peut servir à améliorer les techniques de différents domaines comme en jeux vidéos, architecture, robotique médicale et chirurgie virtuelle, réalité virtuelle et augmentée.

Notre méthode combine le lancer de rayon avec le raffinement sélectif et permet l'extraction des contours apparents des objets réels de la scène. Ces contours, ajoutés au rendu réaliste de la scène permettent à l'utilisateur de mieux comprendre la scène et de distinguer entre les différentes parties de l'image qui correspondent à ceux des objets réels ou ceux des réflexions, réfractions et ombrages. Les résultats obtenus apparaissent convaincants et la méthode peut être précédée par des techniques de calcul d'un bon point de vue afin d'améliorer plus précisément la compréhension de la scène.

Chapitre 5 Prise de connaissance par combinaison de modes de visualisation

---

Nous allons parler dans ce chapitre de deux nouvelles approches qui tendent à combiner des modes de visualisation existant afin d'améliorer la compréhension de scènes comportant des objets englobant d'autres objets:

1. La première approche tend à visualiser l'objet englobant en fil de fer et l'objet intérieur en mode plein avec élimination des parties cachées en combinant le z-buffer avec le « back facing culling ».

2. La deuxième approche tend à créer un trou sur l'objet englobant, faisant apparaître son intérieur en:

    a) Éliminant les facettes de l'objet englobant qui cachent l'objet intérieur.

    b) Assombrissant les pixels orthogonaux à la silhouette de l'objet intérieur.

I. Introduction

Avec l'évolution rapide du matériel et l'optimisation des techniques de visualisation, la synthèse d'image a maintenant atteint un niveau de perfection lui permettant de l'utiliser dans divers domaines et pour plusieurs objectifs.

Mais, malgré le progrès des techniques existantes de visualisation, il existe toujours des limites qui empêche d'avoir une bonne compréhension. Considérons le cas des scènes dont on veut connaître à la fois l'intérieur et l'extérieur. Comme par exemple en architecture, le cas d'un bâtiment que l'on veut visualiser avec une partie de son intérieur, ou dans le domaine de la robotique médicale pour le développement des modèles 3D ou dans l'animation des systèmes biologiques, le cas de l'exploration d'un corps humain et d'une partie de l'intérieur d'un organe.

Ce sont des cas qui sont difficiles d'explorer par les méthodes de visualisation existantes puisqu'il s'agit de la visualisation d'une scène qui comporte un objet englobant un autre objet. Ainsi, les techniques d'explorations basées sur le choix d'un bon point de vue ou d'une animation globale tout autour de la scène sont inefficaces. En effet, l'objet englobé reste toujours caché à cause des modes de visualisation utilisés qui tendent à ne visualiser que l'objet le plus proche à l'oeil. Ce qui fait que l'objet qui se trouve à l'intérieur étant toujours plus loin par rapport l'oeil, il ne sera jamais visualisé.

D'autre part, les techniques basées sur l'exploration locale peuvent nous aider à explorer l'intérieur de l'objet englobant mais elles sont aussi insuffisantes parce qu'elles ne donnent que des informations locales. Nous avons besoin aussi d'avoir une idée globale de la scène. En combinant ces techniques avec celles basées sur le calcul de bon point de vue et l'animation globale tout autour de la scène, nous allons peut être pouvoir obtenir une exploration locale ainsi qu'une idée globale de la scène. Mais il ne faut pas oublier le fait que ces techniques sont très coûteuses en mémoire et difficiles d'être appliquées en temps réel.

D'où la nécessité de créer de nouvelles techniques qui résolvent ce problème. Pour ce faire, nous avons pensé à changer les modes de visualisation connus (le fil de fer et le mode plein avec une élimination des parties cachées) et utiliser des nouveaux modes alternatifs qui combinent les modes de visualisation existants afin de permettre d'avoir en même temps une idée globale sur la scène ainsi qu'une possibilité d'explorer les objets qui se trouvent à l'intérieur, tout en économisant le temps et la mémoire.

II. Techniques alternatives de visualisation

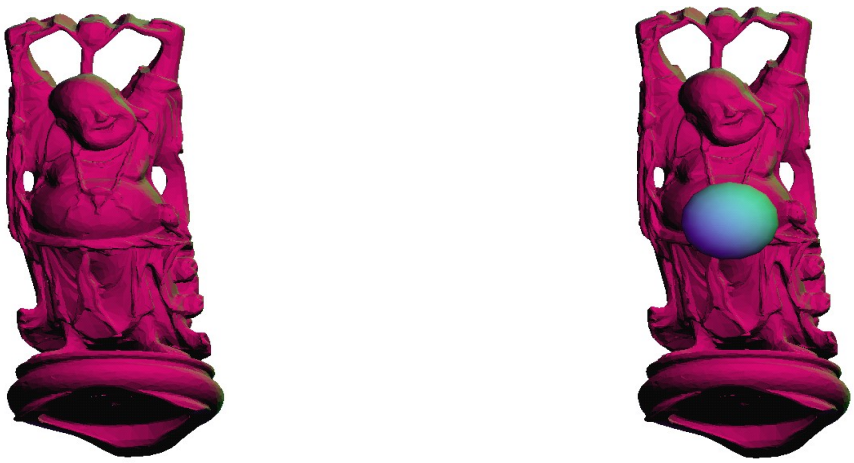
Avant de décrire les techniques alternatives que nous proposons afin d'améliorer la visualisation des scènes composées d'objets englobant d'autres objets, nous devons tout d'abors, être capable de savoir quels sont les objets intérieurs de la scène que nous voulons rendre visible.

Les techniques existantes de visualisation sont d'après le chapitre 3 de ce mémoire, la visualisation en fil de fer dans laquelle il s'agit de la visualisation de tous les contours des objets de la scène, et la visualisation en mode plein avec une élimination des parties cachées qui tend à visualiser l'objet le plus proche à la position du point de vue.

Dans le cas des scènes comportant des objets englobant d'autres objets, la visualisation des objets en mode plein avec une élimination des parties cachées nous empêche de voir l'intérieur d'un objet puisque l'objet intérieur est toujours masqué par l'objet englobant jugé le plus proche à l'observateur (*Fig. 5.1(a)*). En désactivant, l'élimination des parties cachées, la visualisation des objets sera faite selon leurs ordres de l'appel dans l'algorithme.

Si l'objet englobant était appelé après l'objet intérieur, il va certainement le cacher puisqu'il est plus volumineux. D'autre part, cette technique de visualisation est loin du réalisme et elle nous empêche de suggérer l'idée principale et la plus importante de la scène: un objet se trouve à l'intérieur d'un autre objet.

Comme par exemple dans *Fig. 5.1(b),* bien que la sphère doit être selon la définition de sa position, à l'intérieur du Budha, en désactivant l'élimination des parties cachées en appelant la sphère après le Budha, nous avons pu visualiser les deux objets. Mais, un observateur qui n'a aucune idée sur la description de la scène ne va jamais comprendre qu'il s'agit d'une sphère qui se trouve à l'intérieur du Budha.

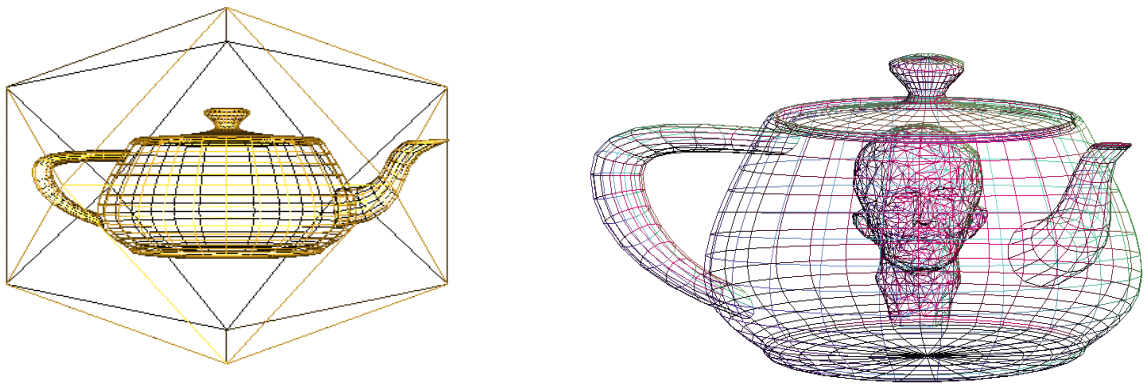


(a) (b)

Fig. 5.1: (a) Avec élimination des parties cachées. (b) Sans élimination des parties cachées.

Si nous passons vers la visualisation en mode fil de fer, la visualisation juste des contours de l'objet englobant va nous permettre de voir son intérieur tout en restant capable d'avoir une idée globale sur la scène.

Voici ci-dessous dans *Fig. 5.2 (a) et Fig. 5.2 (b),* des exemples d'objets englobant d'autres objets, visualisés en mode filiaire. Il s'agit, dans *Fig. 5.2 (a),* d'un Isocahédron solide contenant une théitère à l'intérieur, alors que dans *Fig. 5.2 (b)*, d'une théière solide contenant la Tête d'une statue. Il est bien clair qu'avec une visualisation en mode fil de fer, il sera plus facile d'avoir une meilleure compréhension de la scène puisque la visualisation juste des contours des polygones de l'objet englobant, nous permet de voir son contenu et par suite d'avoir une idée sur les objets qui se trouvent à l'intérieur.



(a) (b)

Fig. 5.2: (a) Visualisation en mode fil de fer du Isocahédron contenant une Théitère à l'intérieur. (b) Visualisation en fil de fer du Théitère contenant l'objet Statue à l'intérieur.

Or, dans le cas des scènes qui sont modélisées par un grand nombre de polygones, la visualisation en mode filiaire de tous les objets va rendre l'image obtenue gênante et ambiguë par rapport à l'observateur puisqu'il sera difficile de voire les détails des objets et de distinguer entre les différents objets présents dans la scène.

En effet, dans *Fig. 5.3 (a)* il s'agit de la visualisation en mode fil de fer d'une scène comportant une sphère se trouvant à l'intérieur d'un lapin composé de 4052 sommets et de 8100 polygones. Avec ce nombre de polygones, il est à peine possible de voir ou distinguer qu'il s'agit d'une boule à l'intérieur. Alors qu'avec les 13295 sommets et 26999 polygones du Budha qui se trouve dans la scène de *Fig. 5.3 (b)*, il est impossible de distinguer la boule.

Ceci à cause du grand nombre de polygones présents dans la scène. Plus le nombre des polygones et celui des sommets augmentent dans la scène plus la compréhension de la scène sera difficile ou même parfois impossible.

(a) *(b)*

Fig. 5.3: (a) Visualisation en mode filiaire du Lapin et de la boule qui est à peine visible. (b) visualisation en mode filiaire de Budha et de la boule qui demeure non visible.

Nous constatons alors, que lorsqu'il s'agit d'une scène dont les objets sont modélisés par des polygones, et dont l'un englobe l'autre, la visualisation des deux objets dans le même mode de visualisation (fil de fer ou mode plein) est gênante et n'est pas efficace pour avoir une bonne compréhension de la scène puisque la visualisation en mode plein va masquer l'objet intérieur *(Fig. 5.1 (a))* et avec la visualisation en fil de fer, nous allons avoir du mal à distinguer entre les contours des deux objets *(Fig. 5.3(b))*.

Comment faire donc pour pouvoir visualiser les deux objets, l'objet englobant et l'objet qui se trouve à l'intérieur tout en respectant l'élimination des parties cachées, sans perdre l'idée globale de la scène, sans avoir une ambiguïté sur la compréhension de la nature et la description de l'objet intérieur et finalement sans perdre l'inspiration qu'il s'agit d'un objet à l'intérieur d'un autre objet?

Nous avons pensé qu'une combinaison des modes de visualisation, la visualisation des contours apparents de l'objet englobant et la visualisation en mode plein avec l'élimination des parties cachés des objets intérieurs**,** pourra nous aider à avoir une meilleure prise de connaissance des scènes qui comportent des objets englobant d'autres objets. En effet, la visualisation de l'objet englobant en mode filiaire nous permet de voir son contenu alors que la visualisation de l'objet à l'intérieur en mode plein pourra nous aider à avoir plus d'information et de détails sur la nature ou la description de l'objet.

II.1. La visualisation avec élimination des parties cachées de l'objet englobant en mode filiaire et de l'objet intérieur en mode plein

Cette méthode comporte tout d'abord la visualisation en mode fil de fer de l'objet englobant et la visualisation en mode plein avec une élimination des parties cachées par la méthode de z-buffer de l'objet intérieur (*Fig. 5.4(a)* et *Fig. 5.5(b)*). Ceci est possible avec la librairie graphique d'OpenGL en choisissant le mode de visualisation des polygones de l'objet courant avant de le visualiser (mode FILL ou mode LINE). Comme le z-buffer est incapable d'éliminer les contours inutiles et invisibles par le point de vue, nous allons étudier ce problème et le régler séparément dans ce qui suit.

(a) *(b)*

Fig. 5.4: (a) Visualisation du Lapin et de la Boule en mode filiaire. (b) Visualisation du Lapin en mode filiaire et la Boule en mode plein

(a) *(b)*

Fig. 5.5: (a) Visualisation du Budha et de la Boule en mode filiaire. (b) Visualisation du Budha en mode filiaire et la Boule en mode plein

Pour l'élimination des contours cachés de l'objet englobant, nous allons utiliser la méthode de « back facing culling » (BFC).

Comme nous avons déjà vu dans le chapitre 3, pour éliminer les parties cachées, il suffit d'associer une orientation aux polygones frontaux et l'orientation inverse au polygones arrière afin d'éliminer tous les polygones qui admettent le critère qui définie l'orientation des polygones arrières.

Le travail sur l'espace image étant plus simple, nous allons travailler sur l'orientation des projetés des polygones et se baser sur le critère défini par OpenGL [WNDS 99] qui étudie le signe de la valeur A du polygone calculé en coordonnées fenêtres et qui est défini par la formule suivante:

$$A = \sum_{i=0}^{n-1} x_i y_{i\oplus 1} - x_{i\oplus 1} y_i$$

Où:

- $i \oplus 1$ est (i+1) mod n
- n est le nombre de sommets dans le polygone
- x_i et y_i sont les coordonnées fenêtres du sommet indice i du polygone

Pour un objet vu de l'extérieur, la valeur *A>0* correspond aux polygones qui sont front facing alors que *A<0* correspond aux polygones back facing. Afin donc de ne visualiser que les contours apparents de l'objet englobant, il suffit d'activer le l'élimination des polygones ayant la valeur *A<0* puisqu'ils correspondent pour un objet vue de l'extérieur aux polygones arrières.

Mais pour un objet qui se trouve à l'intérieur, l'orientation des polygones frontaux et arrières change s'ils étaient vus de l'extérieur, et prennent le sens inverse. Les polygones frontaux sont dans le sens inverse des aiguilles d'une montre (CW) par rapport à la position de l'oeil, et qui correspondent alors à la valeur *A<0*. Les polygones arrières sont alors dans le sens des aiguilles d'une montre (CCW) et correspondent à la valeur *A>0*.

Si nous laissons le test de l'élimination activé et nous appliquons le même processus à l'objet intérieur en le visualisant en mode plein, nous allons éliminer les polygones ayant *A<0* et qui correspondent dans ce cas là aux polygones frontaux qui doivent être visibles. Il y aura dans ce cas là, la visualisation des polygones arrières par rapport au point de vue.

Voici ci dessous dans *Fig. 5.6*, un exemple dans lesquels nous avons activé le test de l'élimination des polygones (polygones arrières pour l'objet englobant) ayant A<0 sans le désactiver pour l'objet intérieur. Nous avons pu éliminer les contours invisibles de la sphère englobante mais le Teapot qui se trouve à l'intérieur est visualisé dans l'ombre.

Nous avons en fait mis dans cette scène une seule lumière située dans la même région de l'espace qui contient le point de vue. Il est évident alors que la partie arrière par rapport au point de vue soit dans l'ombre puisque les faces frontales empêchent la lumière de les voir.

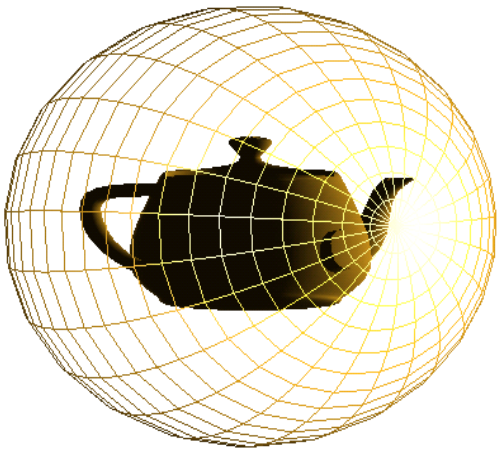
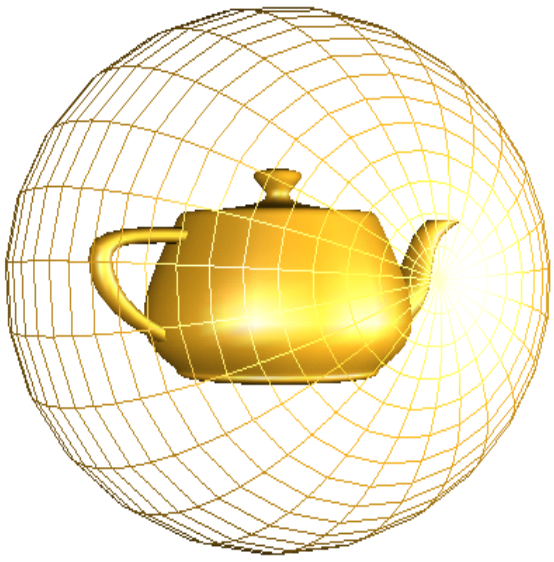
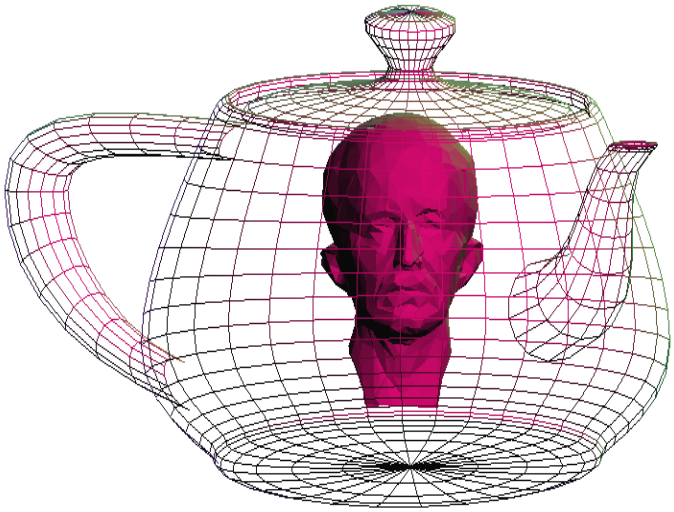


Fig. 5.6:Activation du culling des polygones ayant A<0 sans l'annuler pour la visualisation du Teapot se trouvant à l'intérieur et en présence d'une seule lumière dans la région du point de vue.

Pour éviter ceci il suffit d'éliminer le test de l'élimination pour l'objet qui se trouve à l'intérieur et laisser au tampon de z-buffer le travail de l'élimination des polygones arrières. Nous obtiendrons alors les résultats présents dans *Fig. 5.7*:



(a)



(b)

(c)

(d)

Fig. 5.7: Élimination des parties arrières par le backfacing culling et le test de z buffer

Lorsque la scène comporte plus qu'un objet intérieur, la méthode fonctionne correctement tous en éliminant les parties cachées comme il faut (*Fig. 5.8*). Mais lorsque le nombre d'objets englobant augmente, le back facing combiné avec le test de z-buffer seront incapables d'éliminer les contours arrières dans la zone où il y a l'intersection des contours des objets englobant.

Ceci est dû au fait qu'il est difficile d'associer deux orientations différentes aux facettes de l'intersection, dont l'une associée au premier objet englobant et l'autre associée au deuxième (*Fig. 5.9*).

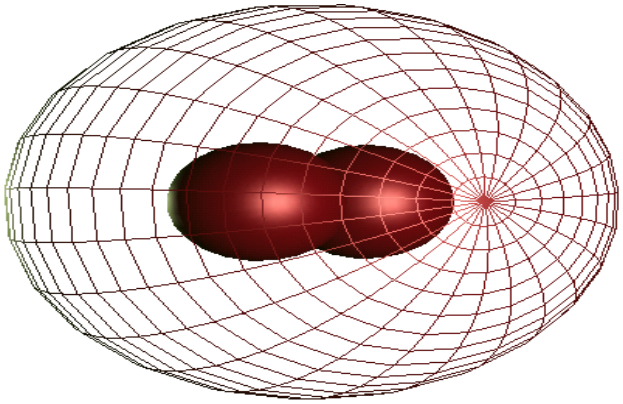


Fig. 5.8 : Deux objets intérieurs.

Par contre, ceci n'influe pas sur la compréhension de la scène dans le cas où le nombre de polygones est raisonnable. En outre, la présence de tous les contours des facettes de l'intersection des deux objets englobants, ajoute plus de réalisme à la scène et aide à avoir une meilleure compréhension de la scène.

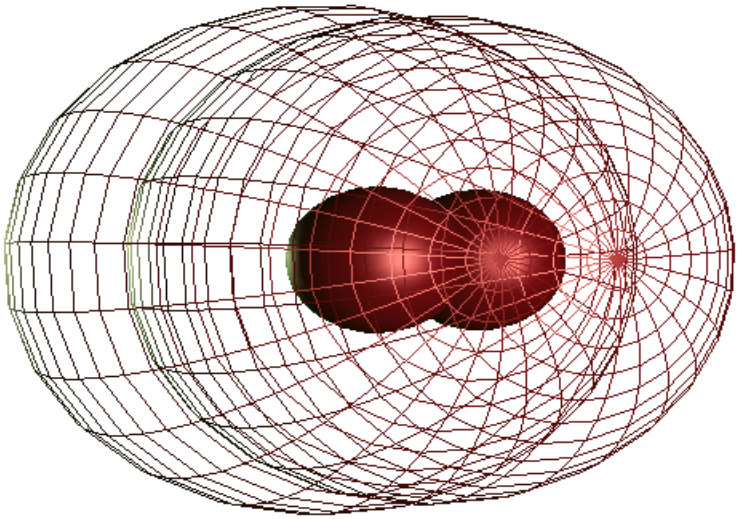


Fig. 5.9 : Deux objets intérieurs et deux objets englobant.

Le problème devient compliqué lorsque le nombre de polygones des objets englobant devient énorme. La visualisation des contours apparents peut devenir gênante et nous empêcher à comprendre l'objet qui se trouve à l'intérieur.

Ajoutons à cela le fait que, la visualisation juste des contours de l'objet englobant peut contribuer à une perte d'information surtout si nous avons intérêt à voir une grande partie de l'extérieur de l'objet englobant ainsi qu'une partie de l'intérieur sans que les contours de l'objet englobant apparaissent. Comme par exemple, dans le cas de l'avion de la figure *(Fig. 5.10)*.

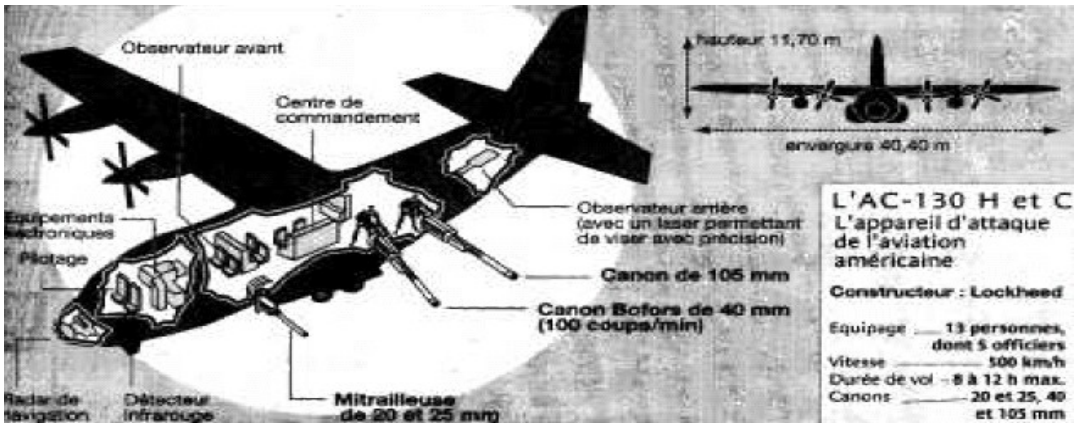


Fig. 5.10: L'image d'un avion avec des trous laissant voir son intérieur.

Pour pouvoir aboutir à un tel résultat, nous avons pensé à une technique qui tend à visualiser l'objet englobant en introduisant la notion d'un trou qui soit juste proportionnel à la taille des objets intérieurs et qui leur rend suffisamment visibles.

II.2. Faire un trou proportionnel à la taille des objets intérieurs

Pour créer un trou qui soit proportionnel à la taille des objets intérieurs, nous avons proposé deux méthodes :

1. Éliminer les facettes de l'objet englobant qui cachent l'objet intérieur.

2. Assombrir à une certaine profondeur, les pixels qui sont orthogonaux à la silhouette des objets intérieurs.

II.2.1. Élimination des facettes de l'objet englobant qui cachent l'objet intérieur

Tout d'abord, comme il s'agit d'une méthode qui crée un trou par une élimination de certaines facettes de l'objet englobant, il est important donc, pour qu'elle soit applicable, que les objets englobant de la scène soient définis par une modélisation polygonale.

En plus, il est nécessaire de savoir quelle est la partie intérieure que nous voulons rendre visible. Cette partie peut être définie par n'importe quelle modélisation que ce soit une représentation polygonale ou représentation analytique.

Notre technique de visualisation réaliste traite le problème de visibilité en même temps que le calcul de l'illumination des objets. Notre intervention réside dans la partie qui consiste en la détermination des points ou parties visibles de la scène. Une fois un point visible est déterminé, le calcul de sa luminosité se fait de la même façon qu'avec le lancer de rayon ordinaire.

La méthode traditionnelle de visualisation par lancer de rayon avec une élimination des parties cachées tend à régler le problème de la visibilité en cherchant l'objet coupé (ou le polygone) le plus proche à l'oeil de l'observateur. Le point d'intersection entre le rayon et l'objet croisé sera alors calculé et la luminosité en ce point sera calculée et affectée au pixel correspondant. L'algorithme décrivant cette méthode est le suivant:

```
minDist=10^10
objetPlusProche=-1;
pour Ifac allant de 0 au Total_Facettes-1  faire
      S i (coupe(Rayon, Facette[Ifac])=vrai) alors
            Si (intersectDist<minDist) alors
                  minDist=intersectDist
                  objetPlusProche=Facette[Ifac]
            FinSi
     FinSi
FinPour
Si (objetPlusProche ≠ -1)
            calculePointIntersection(rayon,  objetPlusProche, pointIntersection)
            calculeCouleurPixel(Pixel, lumière, pointIntersection, r, v, b)
FinSi
```

Cependant, cette technique n'est pas efficace pour régler le problème de la visibilité de l'objet intérieur. Pour ce faire, nous avons proposé d'éliminer les facettes de l'objet englobant que nous considérons inutiles et qui cachent l'objet intérieur. L'algorithme de visibilité doit alors dépasser ces facettes inutiles et associer aux rayons qui les croisent, l'objet intérieur comme étant le plus proche à l'observateur afin d'être visualisé. Le point d'intersection le plus proche entre le rayon et l'objet intérieur est alors calculé, et la couleur en ce point est associée au pixel correspondant.

Afin d'obtenir un trou suffisant qui entoure l'objet intérieur sans trop l'enserrer, on doit éliminer :

1. Toutes les facettes de l'objet englobant qui sont avant et parfois derrière l'objet intérieur.

2. Toutes les facettes qui sont dans le voisinage de l'objet intérieur et qui sont derrière une facette avant à éliminer.

La tache la plus importante de l'algorithme est d'être capable d'identifier ces facettes inutiles afin de les éliminer. Pour ce faire, un paramètre d'élimination initialisé au début à zéro est associé à chaque facette de l'objet englobant.

Un rayon est alors lancé vers chaque pixel de l'image de projection. Les rayons qui coupent en même temps une facette de l'objet englobant et l'objet intérieur déterminent les facettes avant à éliminer pour lesquelles le paramètre d'élimination est mis à un.

L'ajustement du paramètre d'élimination devrait être faite pour les facettes croisées qui sont avant et derrière l'objet intérieur par rapport à la position du point de vue de l'observateur. Dans les deux cas, c'est l'objet intérieur qui doit être visualisé. En effet, si la facette croisée était avant l'objet intérieur, il s'agit alors d'une facette inutile, qui empêche l'apparition de l'objet intérieur et qui devrait être supprimée afin de rendre l'objet intérieur visible.

Mais si la facette croisée était derrière l'objet intérieur, ce dernier étant plus proche à l'observateur, devrait être automatiquement visualisé. Cependant, l'ajustement du paramètre d'élimination à un, permet d'éviter de visualiser une partie de la facette croisée par un rayon qui la coupe sans croiser l'objet intérieur tout en la jugeant la plus proche à l'observateur (*Fig. 5.11*).

Dans un cas pareil, la couleur associée au pixel correspondant sera celle du point d'intersection du rayon avec la facette, ce qui permet alors la visualisation d'une partie de la facette, empêchant ainsi la création d'un trou bien clair qui entoure l'objet intérieur tout entier (*Fig. 5.12*).

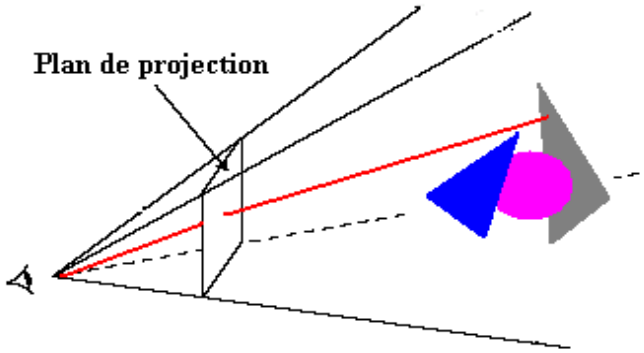


Fig. 5.11: exemple d'une facette derrière l'objet intérieur vue par le rayon lancé comme étant la plus proche à l'observateur.

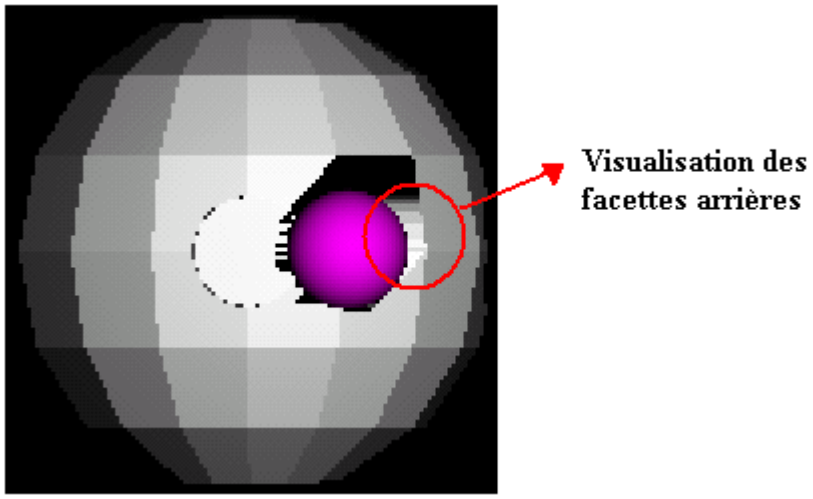


Fig. 5.12: Visualisation des facettes arrières dont le paramètre d'élimination n'est pas ajusté à un.

Les facettes qui sont derrière les facettes avant à éliminer et qui sont évidement dans le voisinage de l'objet intérieur, doivent être aussi éliminées afin de ne pas avoir des facettes visualisées proches à l'objet intérieur (*Fig. 5.13*). Ces facettes ne peuvent pas être éliminées à travers le paramètre d'élimination puisque, aucun rayon qui les coupe, ne croise l'objet intérieur. Et par suite le critère d'ajustement du notre paramètre n'est pas assurée. Afin de régler ce problème, il suffit d'ajuster la distance minimale de vue à chaque fois qu'un rayon coupe une facette avant à éliminer.

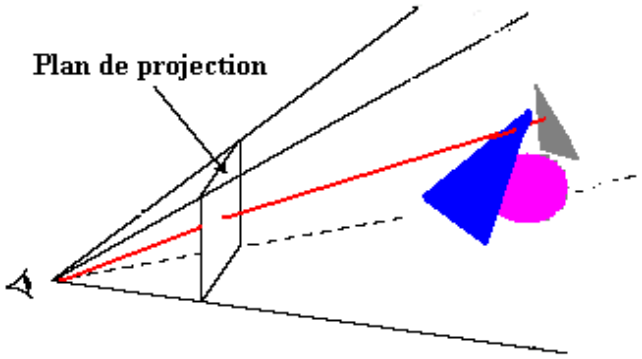


Fig. 5.13: Exemple d'une facette arrière à éliminée proche de l'objet intérieur.

Voici ci dessous dans la figure *Fig. 5.14*, les résultats obtenus par cette technique:

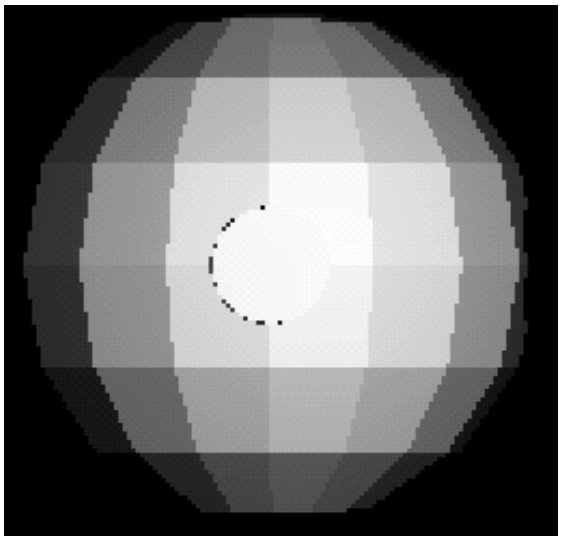


Fig. 5.14(a) :Visualisation en mode plein d'une scène formée de deux sphères l'une dans l'autre

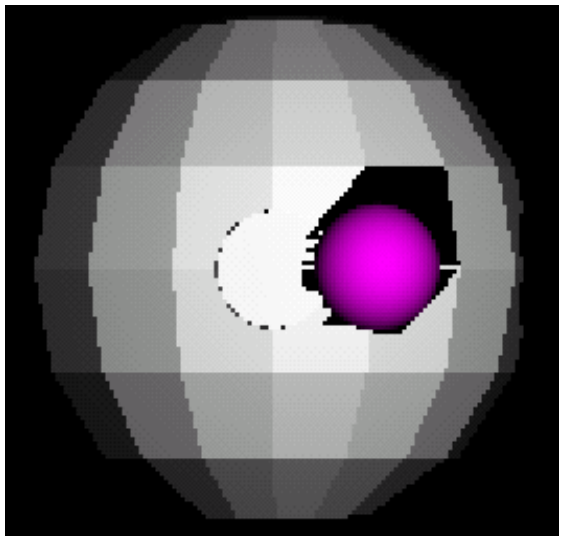


Fig. 5.14(b):Visualisation avec élimination des facettes cachant la sphère qui se trouve à l'intérieur.

L'algorithme de visibilité décrivant ce qui précède est le suivant:

```
Pour Ifac allant de 0 au Total_Facettes_ObjEng-1  faire
éliminer [objEng_facette[Ifac]]=0
finPour

Pour chaque Rayon faire
  minDist=10000000000
  objetPlusProche=-1;
        Pour i allant de 0 à Total_Facettes_ObjEng-1 faire
            Si (coupe(Rayon,objEng_facette[i])=vrai)  alors
                Si (coupe(Rayon, ObjInt)=vrai) alors
                                Si (intersectDist<minDist)
                                        minDist=intersectDist
                                        objetPlusProche=objetInt
                                        éliminer[objEng_facette[i ]]=1
                                FinSi
                        Sinon
                                Si (!éliminer[objEng_facette[i]]) alors
                                        si (intersectDist<minDist) alors
                                                minDist=intersectDist
                                                objetPlusProche=objEng_facette[i]
                                        FinSi
                                Sinon
                                      Si (intersectDist<minDist) alors
                                                minDist=intersectDist
                                                objetPlusProche = -1
                                      FinSi
                                FinSi
                        FinSi
                  FinPour
                FinSi
        FinPour

        Si (objetPlusProche!=-1) alors
                calculePointIntersection(rayon,  objetPlusProche, pointIntersection)
                calculeCouleurPixel(Pixel, lumière, pointIntersection, r, g, b)
        FinSi
FinPour
```

Le résultat de *Fig. 5.14* est obtenu pour des rayons lancés de bas en haut et de gauche à droite de l'espace de projection et pour des intersections avec des facettes de l'objet englobant parcourues de 0 vers nombre_Facette-1. En inversant l'ordre de parcours des facettes, nous obtiendrons le résultat de *Fig. 5.15*:

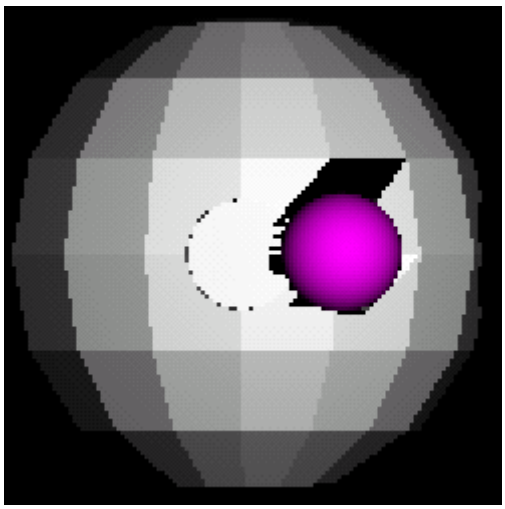


Fig. 5.15: Une partie des facettes avant à éliminer, non éliminée.

Ceci est dû au fait que, cet algorithme dépend de l'ordre de l'appel des facettes ainsi que de l'ordre des rayons lancés. En effet, considérons l'exemple de la figure suivante:

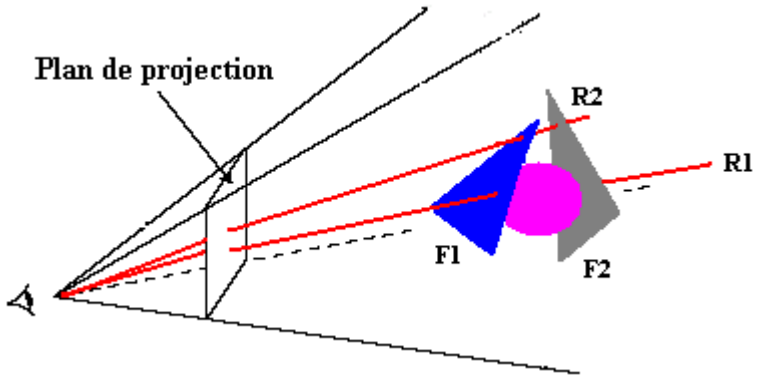


Fig. 5.16: Un exemple qui dépend de l'ordre de rayons lancés et du parcours des facettes.

F1 et F2 sont deux facettes avant et derrière l'objet intérieur. R1 est un rayon qui coupe les deux facettes ainsi que l'objet intérieur alors que le rayon R2 les coupe sans croiser l'objet intérieur.

Si R1 était appelé avant R2, F1 serait considérée comme une facette avant à éliminer et la distance minimale à l'observateur serait celle vers F1. par suite, en passant vers l'étude de R2, comme F1 a été éliminée par R1 et étant plus proche à l'observateur, aucun point d'intersection sera calculé et alors la couleur associé au pixel correspondant sera celle du fond.

Par contre, si R2 était appelé avant, F1 ne serait pas classifiée en tant qu'une facette avant à éliminer parce que le rayon ne croise pas l'objet intérieur, et par suite comme elle est la plus proche à l'observateur, le point d'intersection du rayon avec la facette F1 ainsi que la couleur correspondante seraient alors calculés et associés au pixel correspondant.

D'où la classification des facettes avant à éliminer devrait être faite à part avant de commencer le processus de visualisation. Pour ce faire, nous avons pensé d'ajouter une étape préliminaire, qui tend à lancer un rayon vers les pixels de l'image de projection et met à jour le paramètre d'élimination à chaque fois qu'un rayon coupe une facette de l'objet englobant et l'objet intérieur.

Cependant, le test de tous les pixels de l'espace image est une perte de temps et de mémoire parce que les pixels utiles sont ceux qui sont dans la projection de la boîte englobant l'objet intérieur sur l'espace image. Il suffit donc de calculer ces pixels en projetant la boîte englobant l'objet intérieur sur l'espace de projection afin d'obtenir un sous espace de projection dans lequel les rayons lancés vont certainement rencontrer les facettes avant à éliminer.

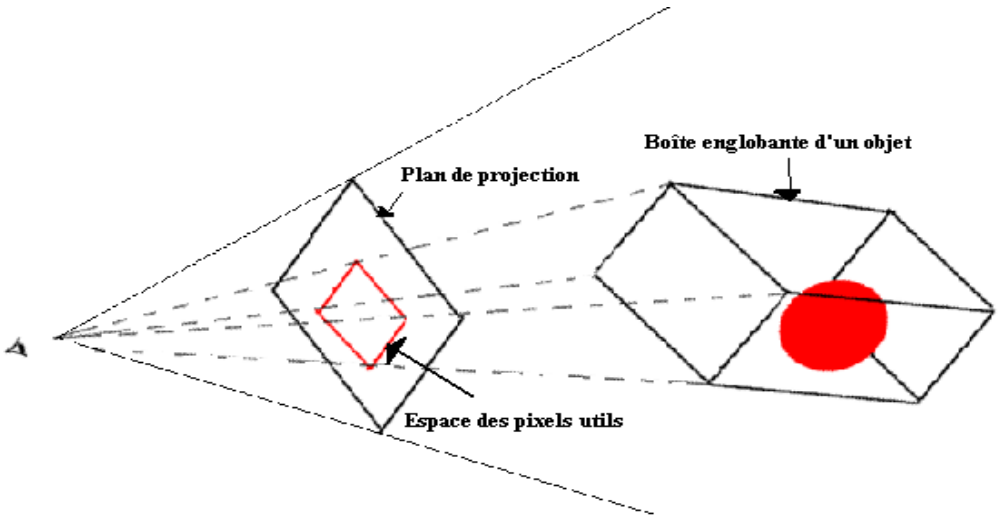


Fig. 5.17: Projection de la boîte englobante d'un objet dans l'espace image.

Une fois les pixels utiles calculés, les rayons correspondants qui croisent en même temps une facette de l'objet englobant et l'objet intérieur déterminent les facettes avant à éliminer de la scène selon l'algorithme suivant:

```
Pour chaque Rayon lancé vers les pixels utiles faire
        Pour i allant de 0 à Total_Facettes_ObjEng-1 faire
            Si (coupe(Rayon,objEng_facette[i]=vrai ) et (coupe(Rayon, ObjInt)=vrai )) alors
                                éliminer[objEng_facette[i ]]=1
            FinSi
        FinPour
Fin Pour
```

L'algorithme de visibilité décrit précédemment sera maintenant appliqué sans aucun souci provenant de l'ordre des facettes ou des rayons lancé parce que nos facettes sont maintenant bien classifiées avant de commencer le test de visibilité.

En cherchant, avant de commencer l'algorithme de visibilité, les facettes avant à éliminer, cettes dernières ont pu être éliminées tout entières permettant ainsi d'avoir un trou bien clair entourant l'objet englobant sans trop l'enserrer. Le résultat obtenu est le suivant:

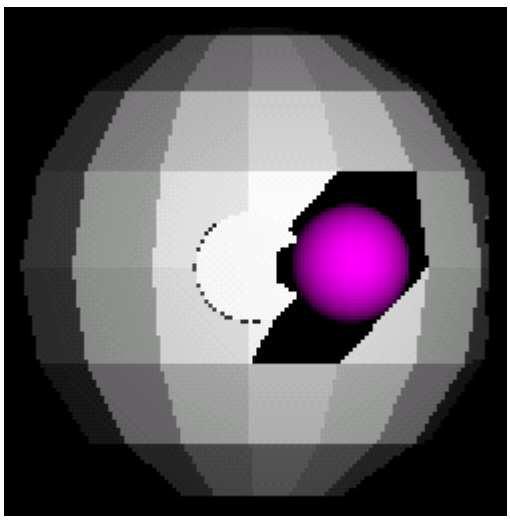


Fig. 5.18: Le résultat obtenu en faisant la recherche des facettes avant à éliminer, avant de commencer l'algorithme de visibilité.

Mais parfois, pour des scènes modélisées par des polygones de grandes ou petites tailles, la création d'un trou par une élimination de facettes inutiles n'est pas efficace. En effet, dans le cas de polygones de grandes tailles, nous risquons d'enlever une grande partie de l'objet englobant ce qui pourrait le rendre ambigu et difficile à comprendre (*Fig. 5.19*). Alors que dans le cas de polygones de très petites tailles, le trou créé risque à ne pas apparaître parce qu'il va enserrer l'objet intérieur ce qui va aboutir à perdre l'impression qu'il s'agit d'un objet à l'intérieur d'un autre.

Il ne faut pas oublier aussi le cas des scènes qui sont représentées analytiquement par des courbes, surfaces ou volumes qui peuvent être définis par des équations implicites ou explicites. Dans ce cas là, il est évident que la méthode de l'élimination des facettes polygonales ne peut pas être appliquée puisqu'il ne s'agit pas de polygones.

Nous avons donc besoin de penser à une méthode qui crée un trou indépendamment de la façon dont la scène est modélisée. D'où l'idée de créer un trou qui soit proportionnel à la silhouette de l'objet caché et suivant une certaine épaisseur de façon à ce que l'objet caché soit visible tout en donnant l'impression qu'il est à l'intérieur d'un autre objet.

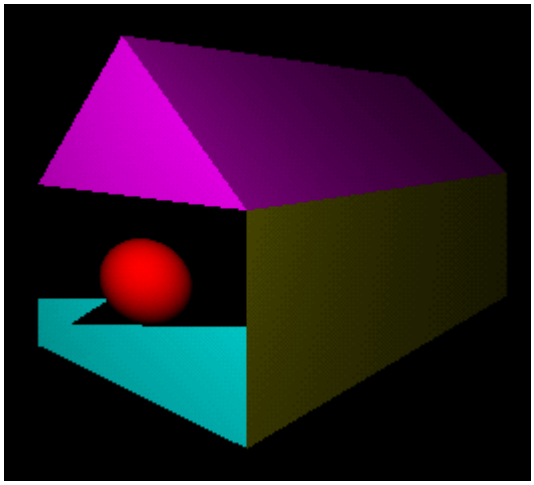


Fig. 5.19: Cas d'une scène modélisée par des polygones de grandes tailles

II.2.2. Assombrir les pixels orthogonaux à la silhouette de l'objet intérieur

Nous allons tout d'abord supposer que l'algorithme est capable d'identifier les objets ou les parties intérieures à voir indépendamment des objets qui se trouvent à l'extérieur. Pour faciliter les choses, nous désignons dans la suite de cette partie, les objets extérieurs par les objet englobants, et les objets ou les parties intérieurs que nous voulons rendre visible par les objets intérieurs.

Cette méthode est divisée en deux étapes.

1. Visualiser en même temps les objets englobants et les objets intérieurs.

2. Assombrir les pixels orthogonaux à la silhouette des objets intérieurs et qui sont dirigés vers l'extérieur des objets intérieurs.

II.2.2.1. Visualisation des deux objets : l'objet englobant et l'objet intérieur

Cette étape s'occupe de la visualisation des deux listes d'objets, les objets englobants et les objets intérieurs. Lorsqu'il existe plusieurs objets englobant et/ou plusieurs objets intérieurs, la technique de visualisation doit respecter la visualisation de l'objet le plus proche par rapport à la position du point de vue entre les objets englobants d'une part et les objets intérieurs d'une autre part.

Nous avons ensuite associé aux objets de la scène une certaine priorité définie comme suit:

Lorsqu'un rayon lancé coupe l'un des objets englobants et ne coupe aucun des objets intérieurs, l'algorithme associe la priorité aux objets englobant de la scène qui seront les seuls objets testés pour la recherche de l'objet le plus proche à l'observateur afin de le visualiser. Tandis que le degré de priorité sera associé aux objets intérieurs dans le cas où le rayon lancé coupe l'un des objets englobants puis l'un des objets intérieurs. L'algorithme cherche dans ce cas là parmi les objets intérieurs celui qui est le plus proche à l'observateur afin de le visualiser.

Pour la recherche de l'objet le plus proche parmi l'une ou l'autre liste des objets, il s'agit de l'algorithme ordinaire connu de lancer de rayon qui calcule la distance de l'oeil à chaque objet de la liste ayant la priorité, et la compare avec la distance

minimale obtenue. Cette dernière sera mise à jour ainsi que l'objet le plus proche à chaque fois que la nouvelle distance à un objet donné est plus petite que celle de l'objet courant.

L'algorithme de visualisation est le suivant:

```
Pour chaque Rayon faire
  minDist=10000000000
  ObjetPlusProche=-1;
  Pour I allant de 0 au Nb_ObjEng-1  faire
        Pour J allant de 0 au Nb_ObjInt-1  faire
              Si (coupe(Rayon, ObjetInt[J]) alors
                    Si (dist(oeil, ObjetInt[J])<minDist) alors
                          minDist=dist(Oeil, ObjetInt[J])
                          objetPlusProche=ObjetInt[J]
                    FinSi
              Sinon
                    Si (dist(oeil, ObjetEng[I])<minDist) alors
                          minDist=dist(Oeil, ObjetEng[I])
                          objetPlusProche=ObjetEng[J]
                    FinSi
              FinSi
        FinPour
  FinPour
FinPour
```

II.2.2.2. Assombrir les pixels orthogonaux à la silhouette de l'objet intérieur

Cette étape consiste tout d'abord à détecter dans l'espace fenêtre, les pixels qui appartiennent à la silhouette des objets intérieurs, puis à assombrir à partir d'une certaine profondeur, tous les pixels qui sont orthogonaux à la silhouette et dirigés vers l'extérieur.

La détection de la silhouette des objets intérieurs est certainement possible, indépendamment de la façon dont la scène est modélisée par l'approche décrite dans le chapitre 4 qui utilise le raffinement sélectif muni du suivi du contour par la méthode du code de direction.

La recherche des pixels orthogonaux à la silhouette se fait dans la deuxième partie de la méthode de détection des contours apparents, celle de code de direction, pendant la recherche du *pixel contour* suivant.

La méthode de code de direction tend à suivre le contour en partant d'un pixel nommé pixel courant afin de chercher le *pixel contour* suivant qui sera utilisé pour la prochaine étape en tant que le pixel courant.

À chaque fois que nous trouvons le *pixel contour* suivant, nous cherchons dans l'espace image 2D, le vecteur tangent à la silhouette de l'objet intérieur au point courant et qui est calculé en faisant la différence entre le point contour suivant et le point contour courant. Le vecteur tangent est calculé par la formule suivante et sera mis à jour à chaque fois qu'un nouveau *pixel contour* est trouvé (*Fig. 5.20*).

Vect_tangent (pixel_courant) = pixel_contour_suivant – pixel_courant

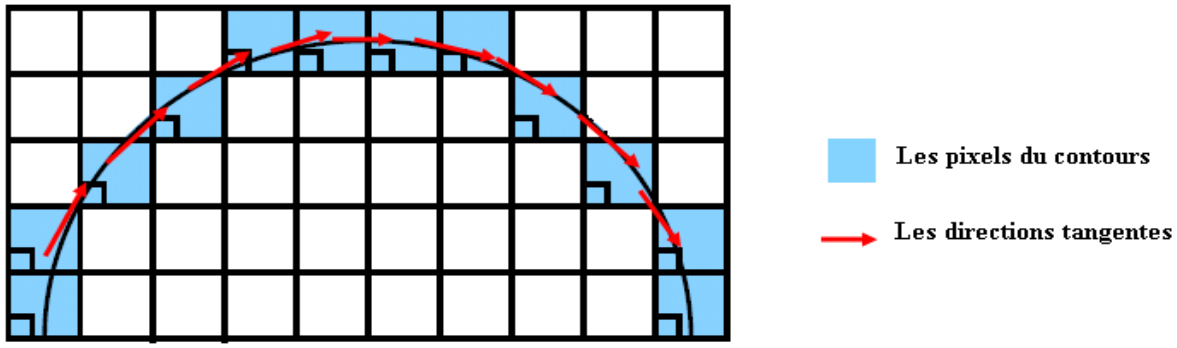


Fig. 5.20: les directions tangentes calculées tout au long de la silhouette de l'objet .

Nous cherchons ensuite pour chaque vecteur tangent, et toujours dans le plan de projection 2D, le vecteur normal sortant dirigé vers l'extérieur de l'objet intérieur.

Pour chaque vecteur tangent, deux vecteurs normaux sont possibles d'être atteints vers deux points parmi les 8 voisins du point courant.

Voici ci dessous dans *Fig. 5.21*, l'exemple d'une direction tangente dont le vecteur est de coordonnées (1,1) dans un espace 2D,et qui est dirigé vers le point Haut-Droite (HD) du voisinage du pixel courant (C). Les vecteurs normaux possibles dans ce cas là, sont alors dirigés vers l'un des deux points Haut-Gauche (HG) et Bas-droite (BD). Le vecteur normal qui est dirigé vers le point HG est de coordonnées (-1,1) alors que le vecteur normal dirigé vers le point BD est de coordonnées (1,-1).

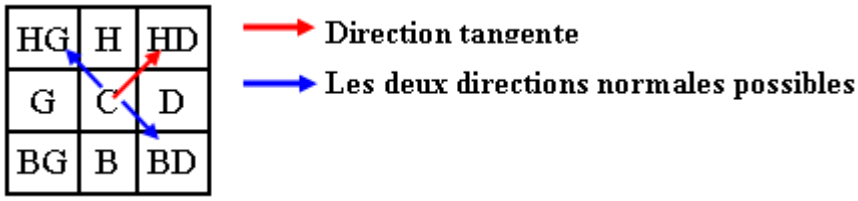


Fig. 5.21: Un exemple d'un vecteur tangent et ces deux vecteurs normaux possibles.

Le problème revient maintenant à déterminer lequel de ces deux vecteurs normaux possibles est le vecteur sortant vers l'extérieur de l'objet. Généralement, comme l'origine de chacun de ces deux vecteurs normaux est un point du contour, le vecteur normal sortant doit avoir comme extrémité un point qui se trouve à l'extérieur de l'objet alors que le vecteur normal entrant doit avoir comme extrémité un point de l'intérieur de l'objet.

D'autre part, l'intérieur et l'extérieur d'un objet sont deux zones qui doivent avoir des couleurs différentes alors que les pixels voisins d'un même objet doivent avoir des couleurs voisines (*Fig. 5.22*). Pour cela, en testant la couleur des deux pixels qui sont à l'extrémité des deux vecteurs normaux, nous obtiendrons le pixel qui correspond à l'extrémité du vecteur normal sortant en rejetant celui qui possède, à partir d'un certain seuil ξ, une couleur voisine à celle de l'objet, éliminant ainsi le pixel qui correspond alors à l'extrémité du vecteur normal entrant.

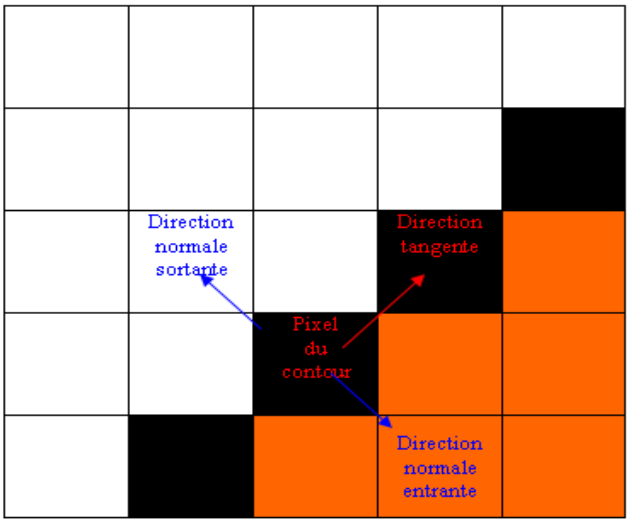


Fig. 5.22: Le pixel de la direction normale entrante possède une couleur voisine à celle de l'objet contrairement au pixel de la direction normale sortante.

Un pixel est donc un point qui appartient à l'intérieur d'un objet si et seulement si :

$$|\text{couleur(pixel)} - \text{couleur(objet)}| < \xi$$

Dans la figure suivante *Fig. 5.23*, nous avons appliqué ce processus à une scène composée d'une sphère blanche qui contient à l'intérieur une petite sphère de couleur magenta. Dans *Fig. 5.23(a),* nous avons juste visualisé les deux sphères en même temps tout en associant aux pixels du contour de la sphère magenta une couleur grise de coordonnées en mode RGB (25,25,25). Dans *Fig. 5.23(b)*, nous avons détecté et assombris tous les pixels extrémités des vecteurs normaux dirigés vers l'extérieur de la sphère magenta.

(a) *(b)*

Fig. 5.23: (a) La visualisation en noir du contour de la sphère intérieure. (b) Un trou avec une profondeur d'un seul pixel.

La profondeur d'un seul pixel n'est pas suffisante pour pouvoir comprendre qu'il s'agit d'un trou. En effet, le trou obtenu est très proche à l'objet et il est difficile de le voir ou le distinguer. Pour cela, afin de définir une profondeur au trou, nous avons pensé à assombrir, à partir d'un certain niveau k, tous les pixels qui sont sur la direction normale sortante obtenu en multipliant le vecteur normal sortant par k.

Comme le vecteur normal est un vecteur unitaire, le pixel à assombrir d'indice k sur la direction normale peut être obtenu en ajoutant aux coordonnées du *pixel contour*, qui est l'origine du vecteur normal, les coordonnées du vecteur normal multipliés par la valeur k. D'où la formule suivante:

$$P_k = P_0 + k \times N$$

Où :

- P_k est le pixel indice k sur la direction normale
- P_0 est l'origine du vecteur normal
- N est le vecteur normal
- k est la profondeur du trou

Il suffit alors d'assombrir tous les P_k pour obtenir le résultat de la figure *Fig. 5.24*:

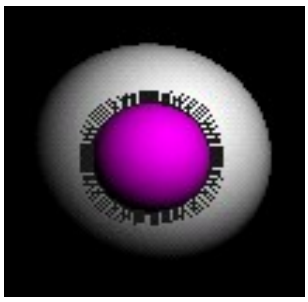


Fig. 5.24: Un trou avec une profondeur de 4 pixels..

Il est bien clair que le trou obtenu n'est pas assez plein et il apparaît comme s'il était discontinu ce qui fait manquer le résultat de réalisme. Pour remplir ce trou, une fois le pixel définissant la direction normale sortante à l'objet est trouvé, nous cherchons dans le voisinage de l'origine de la direction normale, son pixel suivant et son pixel précédent définissant ainsi les directions suivantes et précédentes à la direction normale afin de les assombrir.

Le pixel suivant et le pixel précédent sont obtenus dans le voisinage du pixel courant orienté de 0 à 7 et défini de la même façon qu'au chapitre précédent (*Fig. 5.25*).

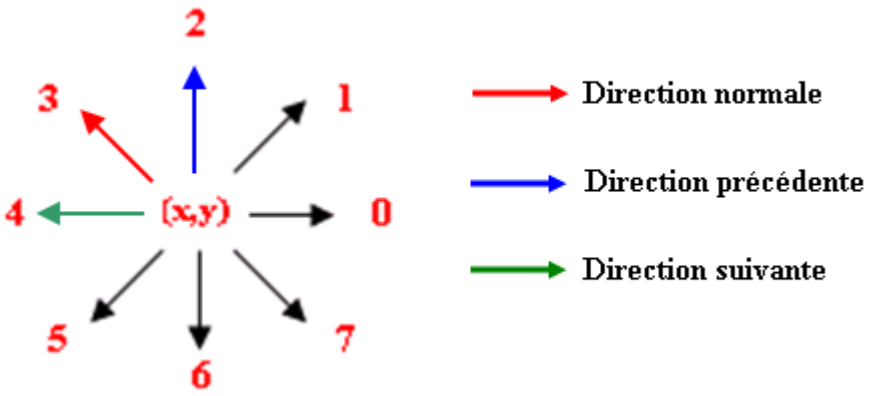


Fig. 5.25: Une direction normale (de couleur *rouge) et sa direction suivante (de couleur verte) et précédente (de couleur bleue) dans le voisinage numéroté de 0 à 7.*

Une fois la directions suivantes et précédentes sont définies, nous assombrissons à un niveau k tous les pixels de la direction normale N_k, les pixels Ns_k de la direction suivant la direction normale et les pixels NP_k de la direction précédent la direction normale.

Considérons par exemple dans la figure *Fig. 5.26*, un exemple qui trace un trou à une profondeur de 2 pixels. Les pixels de la direction normale sont N1 et N2

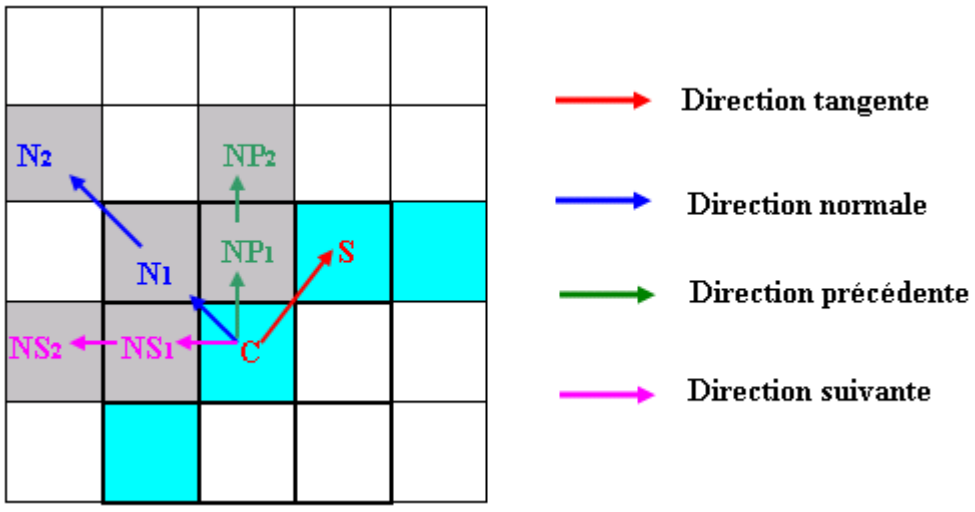


Fig. 5.26: Un exemple d'une profondeur d'un trou de deux pixels.

Généralement, une profondeur k de 4 pixels est suffisante et peut donner de bons résultats. Les résultats obtenus pour la scène des deux sphères (*Fig. 5.27*) et la scène comportant une maison et une sphère dans son intérieur (*Fig. 5.28*) ont été convaincants.

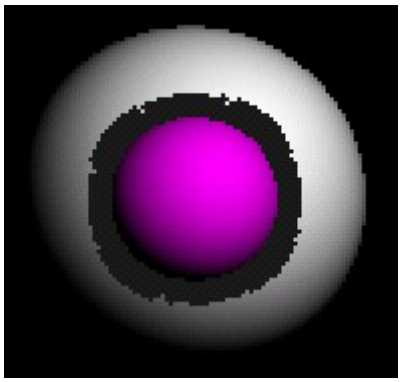


Fig. 5.27: Création d'un trou de deux sphères l'une dans l'autre.

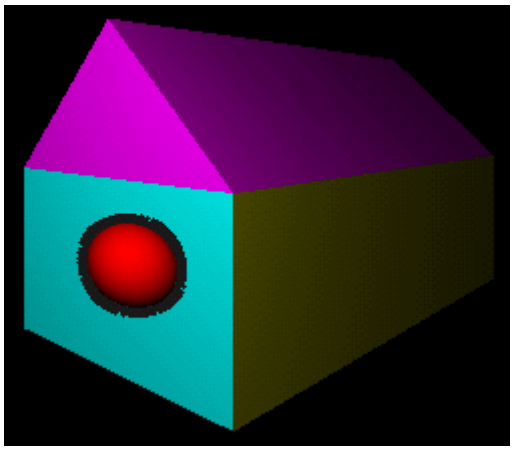


Fig. 5.28: Création d'un trou d'une maison contenant une sphère à l'intérieur.

Plus des pixels de la normale sont assombris, plus le trou sera large. Mais il faut faire attention au fait que le trou ne doit pas être assez grand par rapport au volume de l'objet englobant afin de ne pas en masquer une grande partie, ce qui peut alors aboutir à une mauvaise compréhension de la scène.

Nous ne nous sommes pas intéressés aux cas où l'objet intérieur est très proche à la frontière de l'objet englobant, parce qu'il sera inutile de créer un trou qui n'est pas capable d'avoir une profondeur suffisante sans dépasser le volume de l'objet englobant.

Voici par exemple dans la figure *Fig. 5.29*, le cas de deux sphères tangentes intérieurement.

(a) *(b)*

Fig. 5.29: (a) Deux sphères tangentes intérieurement. (b) Un trou qui masque une grande partie de l'objet englobant

Pour régler ce problème, il faut faire un certain test qui prend en considération la taille de la projection de l'objet englobant sur l'écran 2D et la profondeur du trou. Pour ce faire il suffit de calculer pour l'objet englobant son rectangle englobant obtenu en projetant sur l'écran 2D son volume englobant. Et alors la profondeur ne doit pas dépasser la moitié de la distance minimale entre le dernier pixel à assombrir et les quatre cotés du rectangle englobant.

Afin d'éviter l'obtention de l'image présentée dans *Fig. 5.29(a)*, il est peut être possible d'utiliser l'une des techniques de calcul d'un bon point de vue décrites dans le chapitre 3 de ce mémoire, avant de lancer la méthode de création d'un trou.

III. Conclusion

Nous avons dans ce qui précède, présenté trois techniques qui aident à améliorer la visualisation des scènes tridimensionnelles contenant un objet englobant un autre objet. Ces méthodes sont faciles à implémenter, rapides et ne sont pas coûteuses en mémoire. Les résultats obtenus par chacune de ces trois méthodes sont convaincants et répondent à notre but: l'amélioration de la visualisation et de la prise de connaisance des scènes complexes.

Chacune méthode admet des avantages mais aussi des invonvenients qui lui empêche d'être applicable à tout type de scènes comportant des objets englobant d'autres objets.

La première technique tend à combiner deux techniques de visualisation existantes: la technique de z-buffer avec le back facing culling. Le but de cette combinaison est de visualiser les contours apparents de l'objet englobant afin de permettre à l'observateur de voir son intérieur et ainsi d'identifier l'objet intérieur. Cependent, dans le cas où le nombre de facettes de l'objet englobant devient grand la visualisation des contours apparents des objets englobants peut nuire à l'observateur et l'empêcher de voir son intérieur et ainsi d'avoir une bonne compréhension de la scène.

La création d'un trou autour des objets intérieurs est une idée plus efficace que le première parce qu'elle garantit à l'observateur d'être capable de voir nettement les objets englobants ainsi que leurs intérieurs. Nous avons proposé deux méthodes qui permettent la création d'un trou. La première technique est basée sur la recherche des facettes des objets englobants qui empêchent la visualisation des objets intérieurs afin de les éliminer. Ce qui empêche cette technique d'être appliquée à n'importe quelle modélisation.

La deuxième technique peut être appliquée à n'importe quelle type de modèles puisqu'elle est basée sur la détection des *pixels contours* des objets intérieurs et ensuite sur la recherche des pixels orthogonaux à la silhouette des objets intérieurs afin de les assombrir.

Pratiquement, elle ne présente aucun problème sauf le cas où un objet intérieur est très proche de la frontière d'un objet englobant ce qui nous rend incapable de créer un trou autour de l'objet englobant sans dépasser sa frontière. Afin de régler ce problème, il suffit de débuter par un bon point de vue calculé par l'une des techniques présentée dans le chapitre 3 de ce mémoire.

Chapitre 6 Conclusion générale et perspectives

Nous présentons tout d'abord dans la conclusion générale, une évaluation et une étude critique sur les résultats obtenus ainsi que les améliorations apportées par le travail effectué dans ce mémoire. De nouvelles idées, des possibilités d'amélioration et d'autres axes de recherches sont proposées dans la perspective.

I. Conclusion Générale

Le but du travail entrepris dans cette thèse était la mise en œuvre de techniques alternatives pour l'amélioration de la prise de connaissance de scènes tridimensionnelles complexes qui sont difficiles à comprendre à partir des techniques de visualisation existantes.

Notre objectif est de régler le problème de la compréhension de deux types de scènes complexes. Le premier type est celui des scènes comportant beaucoup d'effets réalistes qui confondent l'observateur et l'empêchent de distinguer entre les objets réels de la scène et les objets illusoires. Le deuxième type de scènes complexes est celui des scènes comportant des objets englobant d'autres objets empêchant ainsi la visibilité des objets intérieurs.

II. Améliorations apportées

Dans le cadre de ce travail nous avons apporté les améliorations suivantes.

1. **Dans le chapitre 4**, nous avons tout d'abord pu, proposer une nouvelle approche qui tend à améliorer la compréhension des scènes complexes difficiles à comprendre à cause de la présence des effets réalistes. La méthode combine:

   - la visualisation réaliste de la scène par la méthode de lancer de rayon muni du raffinement sélectif

   - la visualisation non réaliste par la méthode de suivi de contour muni de la technique de code de direction pour la détection des contours apparents des objets réels de la scène.

- Les résultats obtenus par cette méthode sont convaincants et répondent à notre objectif décrit ci dessus. Ils permettent aussi de conclure qu'il est possible d'améliorer la visualisation des scènes réalistes non seulement par les techniques réalistes mais aussi par la combinaison des techniques non réalistes. Ceci paraîtrait paradoxal mais la détection des contours apparents des objets réels de la scène a permis de les mettre en valeur et de les distinguer des objets illusoires dus aux ombres, réflexions et réfractions.

- La méthode présentée peut être considéré comme une technique principale de détection des contours et entre dans la catégorie des méthodes hybrides. Elle permet de détecter la silhouette ainsi que les arêtes d'intersections et les bordures des objets réels de la scène sans prendre de l'espace en mémoire et de temps de calculs.

- Elle apporte une amélioration considérable par rapport aux techniques existantes puisqu'elle permet de

  - détecter les contours réels de la scène et non pas une forme artistique ou approximative.

  - détecter seulement les contours des objets réels de la scène sans ceux des ombres et des objets réflectés et réfractés.

  - régler le problème de l'anti-aliassage des scènes puisque les contours obtenus sont de l'ordre d'un pixel.

- L'algorithme est capable de détecter les contours des objets indépendamment de la façon dont la scène est modélisée. En effet, il est efficace pour des scènes définies par une modélisation polygonales ainsi que des scènes présentées par une description analytique. Cependant, cette technique ne peut être appliquée que pour des scènes dont la modélisation de chaque objet est définie séparément dans la scène.

2. **Dans le chapitre 5**, nous avons traité le deuxième type de scènes complexes. Celui des scènes qui comportent un objet englobant un autre objet. Pour améliorer la prise de connaissance, nous avons proposée deux approches qui permettent la visibilité des deux objets ensemble: l'objet englobant et l'objet intérieur tout en respectant l'ordre de profondeur des objets.

   - La première tend à faire une combinaison des modes de visualisation existant. La visualisation de l'objet englobant est en mode fil de fer et de l'objet intérieur en mode plein. Nous avons réglé le problème de l'élimination des parties cachées par la combinaison de la méthode de z-buffer avec le « back-facing culling ».

- La méthode est rapide, n'est pas gourmande en mémoire ni en temps de calculs et donne de bons résultats. Cependant, elle ne peut être appliqués que pour des scènes dont l'objet englobant est modélisé par une description polygonale. D'autre part, une fois le nombre de polygones de l'objet englobant devient grand, la technique ne sera pas efficace parce que les contours de tous les polygones de l'objet englobant empêche la visibilité de l'objet intérieur.

- La deuxième approche tend à créer un trou sur les parois de l'objet englobant et proportionnel à l'objet intérieur permettant ainsi la visibilité de ce dernier. Afin de réaliser cet objectif, nous avons proposé deux méthodes.

  - La première méthode tend à éliminer les facettes de l'objet englobant qui empêche la visibilité de l'objet intérieur.

    - La méthode est efficace, facile à implémenter, rapide et donne de bons résultats

    - Cependant,

      - elle donne un très grand trou lorsque les polygones modélisant l'objet englobant sont de grand taille et un trou trop proche de l'objet lorsque les polygones sont de très petite taille

      - elle ne peut pas être applicable pour des objets englobants définis par une description non polygonale

  - La deuxième méthode tend tout d'abord à visualiser les deux objets: l'objet englobant et l'objet intérieur, puis à détecter les pixels de l'espace image qui sont orthogonaux à la silhouettes et dirigés vers l'extérieur de l'objet intérieur afin de les assombrir donnant ainsi l'impression d'un trou.

    - La méthode est efficace, facile à implémenter, rapide et donne de bons résultats

    - Elle peut être appliquée à n'importe quelle modélisation de scènes.

III. Perspectives

Les résultats obtenus permettent de répondre aux objectifs fixés au départ. Ils permettent même d'aller au delà dans la mesure où les techniques proposées, de par leur efficacité, peuvent être utilisées non seulement dans le cadre de l'amélioration de la prise de connaissance des scènes 3D mais aussi dans d'autre applications dans le mondes virtuels relativement complexes.

La méthode de détection de contours présentée dans le chapitre 4, étant une méthode hybride qui travaille sur l'espace objet et donne le résultat sur l'espace image (2D) la rend capable d'être utilisée dans le domaine de la réalité augmentée pour la résolution des problèmes d'occultation des objets ajoutés à une scène en testant l'occultation d'un point du contour au lieu de tester tous les points de l'objet.

En effet, notre méthode de détection des contours détecte sur l'espace image les *pixels contours* de l'objet ajouté sans être obligé de le visualiser tout entier. Elle donne les *pixels contours* exacts de l'objet et non pas sous forme artistique ce qui permet de déduire s'il y a occultation en utilisant tout simplement ces pixels et voir si le rayon qui les traverse coupe la scène ou non.

Il est possible aussi pour des scènes complexes de grandes tailles ou pour un espace image de grandes dimension, d'optimiser les calculs de la luminosité des pixels de l'espace image. En utilisant la méthode de détection des contours par la méthode du raffinement sélectif combiné par le suivi du contour, une fois les *pixels contours* trouvés, le calcul de la luminosité de ces pixels par le lancer de rayon est suffisant pour déduire celles de tous les autres pixels.

En effet, chaque couple de *pixels contours* appartenant à une même ligne de l'espace image définit un intervalle à l'intérieur duquel la couleur des pixels est calculée par une interpolation linéaire. Si la longeur de l'intervalle est assez grande, il suffit de calculer la couleur réelle du pixel milieu de l'intervallele afin d'obtenir deux intervalles. Le processus de subdivision d'intervalle et le calcul des couleurs par interpolation se répètent jusqu'à l'obtention de toutes les couleurs des pixels de l'espace image. Afin d'obtenir les couleurs des objets réflectés et réfractés, il suffit de calculer la couleur des *pixels contours* réflectés et réfractés, puis d'appliquer le processus d'interpolation à l'interieur des intervalles pour calculer la couleurs de tous les objets.

Les résultats obtenus par les méthodes proposées pour l'amélioration de la compréhension des scènes comportant des objets englobant d'autres objets sont prometteurs mais peuvent être complétés par d'autres méthodes.

Il est possible par exemple après avoir créé le trou autour de l'objet intérieur, d'effectuer une animation afin d'extraire plus de détails et d'information sur le contenu de la scène

Afin de régler le problème de l'élimination des parties cachées de ce type de scènes, nous avons utilisé la combinaison de la méthode du z-buffer avec le back-facing culling. Mais il est possible d'imaginer d'autres propositions, comme par exemple de faire la combinaison du tampon de stencil avec celui du z-buffer ou aussi le lancer de rayon avec le back facing culling.

Il est possible aussi d'obtenir de meilleurs résultats dans les précédentes méthodes que nous avons proposées dans ce mémoire par une méthode de calcul d'un bon point de vue. En fait, nous avons utilisé une technique rapide qui tend à calculer la boîte englobante de la scène et prendre un point de vue sur l'axe orthogonal en son centre. Cette méthode n'est pas toujours efficace surtout dans le cas de la création d'un trou d'un objet intérieur qui est proche de la frontière de l'objet englobant. Afin d'éviter une telle situation nous avons besoin d'une technique de calcul d'un bon point de vue capable de calculer une pertinence visuelle comme par exemple celle proposée par Sokolov et al. dans [SP05] et [SPT06] qui est basée sur l'étude de la courbure ou de l'inflexion de la scène.

Bibiographie

[AMAN 84] AMANATIDES J.,
"Ray tracing with cones",
Computer Graphics (Proceedings of SIGGRAPH'84), Vol. 18 No. 3, pp. 129-135,
July 1984.

[AK 87] ARVO (J.), KIRK D.,
"Fast ray tracing by ray classification".
Computer Graphics, Vol. 21 No 4, pp. 55-64,
July 1987.

[APA 02] ATSALAKIS A., PAPAMARKOS N., ANDREADIS I.,
"On estimation of the number of image principal colors and color reduction through self-organized neural networks",
Wiley Periodicals
2002.

[APP 68] APPEL A.,
"Some techniques for shading machine renderings of solids",
Proc. AFIPS Spring Joint Computer Conference,
1968.

[BH 98] BREMER D. J., HUGHES J. F.
"Rapid appriximate silhouette rendering of implicit surface".
Brown University,
1998.

[BL 01] BOTTINO A. and LAURENTINI A.,
"Experimenting with non instructive motion capture in a virtual environment".
The visual Computer, Volume 17, Number 1, pp. 14-29,ISSN 0178-2789.
2001.

[BPP 86] BOUZEFRANE R., PLEMENOS D., PUEYO X.
"Les grande familles d'algorithmes de visualisation 3D"
Revue internationale de CFAO et d'Infographie, Volume1, Number 2,
1986

[BS 00] BUCHANAN J.W. and SOUSA M.C.,
"The edge buffer: A data structure for easy silhouette rendering,"
Proceedings 1st Int'l Symp. Non-Photorealistic Animation and Rendering, ACM Press, pp. 39-42.
2000

[CE 74] CATMULL, EDWIN,
"A subdivision algorithm for computer display of curved surfaces", Ph.D Thesis, University of Utah, Also UTEC.CSC-74-133, and NTIS A004 968,
December 1974.

[CE 75] CATMULL, EDWIN,
"computer display of curved surfaces",
Proc. IEEE Conf. Computer Graphics Pattern Recognition Data Struct.,p.11,
May 1975

[CC 88] COLIN.C,
"A system for exploring the universe of polyhedral shapes",
Eurographics'88, Nice (France),
September 1988.

[CHH 02] CARR A. N., HALL J. D., and HART J. C.,
"The Ray engine",
Department of Computer Science, University of Illinois, Technical Report
UIUCDCS-R-2002-2269,
2002.

[CJ 86] CANNY J.,
"A Computational approach to edge detection",
IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6,
November 1986.

[CM 02] CARD D. and MITCHELL J.L.,
"Non-Photorealistic rendering with pixel and vertex shaders,"
Vertex and Pixel Shaders Tips and Tricks, W. Engel, ed., Wordware,
2002.

[CNH 03] EL-CHAAR W., NASR C., HAMAD D.,
"Recognition of the outlines of an image by ANN",
Proceedings of the Artificial Neural Networks in Engineering Conference,
p651-656, St. Louis, USA,
Nov 2003

[CNH 05] EL CHAAR W. , NASR C., HAMAD D.,
"Automatic SOM based color segmentation algorithm".
Proceeding of the Artificial Neural Networks in Engineering Conference, St. Louis, USA,
November 2005.

[CPSC 98] CHUNG Y. C., PARK J. W., SHIN H., and CHOI B.K.,
"Modeling the surface swept by generalized cutter for NC verification".
Computer-aided Design, Volume 30, Number 8, pp. 587-594.
July 1998

[CPC 84] COOK R.L., PORTER T., CARPENTER L.,
"Distributed ray tracing",
Computer Graphics (Siggraph'84 proceedings), Vol. 18 No 3, pp. 137-145,
June 1984.

[CRR 06] CASSAGNABÈRE C., ROUSSELLE F., RENAUD C.,
" CPU-GPU multithreaded programming podel: application to the path tracing with next event estimation algorithm".
ISVC Vol. 2.pp. 265-275,
2006.

[DEC 96] DECAUDIN P.,
"Cartoon looking rendering of 3D scenes",
Research Report INRIA 2919,
June 1996.

[DG 01] DORME G.,
"Etude et réalisation de techniques de prise de connaissance de scènes tridimensionnelles ".
Thèse de Doctorat, Université de Limoges (France).
Juin 2001

[DPB 07] DANDACHY N., PLEMENOS D., EL HASSAN B.,
"Scene understanding by apparent contour extraction",
3IA Internationnal Conference, Athens (Greece),
March 2007.

[DS 00] DEUSSEN O. and STROTHOTTE T.,
"Computer-generated Pen and Ink illustration of trees",
Proceedingss Siggraph 2000, Computer Graphics, (Proceedings Ann. Conf. Series),
Volume 34, ACM Press, 2000, pp. 13-18.
2000.

[EXC 88] EXCOFFIER T.,
"Construction géométrique de solides et accélérations des algorithmes de lancer de rayons",
Thèse de Doctorat, Lyon,
1988.

[FPT 99] FUA P., PLANKERS R., and THALMANN D.,
"From synthesis to analysis: Fitting human animation models to image data".
In computer Graphics internationnal' 99, pp.4, IEEE CS Press. ISBN 0-7695-0185-0.
June 1999

[FVH 90] FOLEY, VAN DAM, FEINER, HUGUES,
"Computer Graphics : Principle and Practice"
Addison Wesley,
1990

[FVHP 94] FOLEY, VAN DAM, FEINER, HUGUES, PHILIPS,
"Introduction to computer graphics",
Addison-Wesley,
1994

[GB 99] GOOCH B. et al.,
"Interactive technical illustration,"
Proceedings 1999 ACM Symp. Interactive 3D Graphics, ACM Press, 1999,pp. 31-38.
1999

[GHAZ 92] GHAZANFARPOUR D.,
"Visualisation réaliste par lancer de pyramides et subdivision adaptative",
acte du MICAD 92, pp.167-181, PARIS,
Février 1992.

[GH 98a] GHAZANFARPOUR D., HASENFRATZ J.M.,
"A beam tracing with precise antialiasing for polyhedral scenes",
Computer & Graphics, Vol. 22 no 1,
January 1998.

[GH 98b] GHAZANFARPOUR D., HASENFRATZ J.M.,
"Une synthèse des variantes du lancer de rayons et du lancer de faisceaux",
Revue internationale de CFAO et d'informatique graphique, Vol.13, No 3, pp.235-264
Septembre 1998.

[GLA 84] GLASSNER A.S.,
"Space subdivision for fast ray tracing",
IEEE CG&A
October 1984.

[GP 89] GREEN S.A., PADDON D.J.,
"Exploiting coherence for multiprocessor ray tracing".
IEEE computer graphics and application, Vol. 9 No 11, pp. 12-26,
November 1989.

[GRI 75] GRIFFITHS J. S.,
"A data structure for the elimination of hidden surfaces by patches subdivision",
CAD, Vol 7, No 3,
July 1975.

[GRI 78] GRIFFITHS J. S.,
"A surface display algorithm",
CAD, Vol 10, No 1,
January 1978.

[GS 85] GOLDSMITH J., SALMON J.,
"A ray tracing system for the hypercube",
Caltech Concurrent Computing Project Memorandum HM154,

California Institude of Technology,
1985.

[HA 99] HERTZMANN A.,
“Introduction to 3D non-photorealistic rendering: silhouettes and outlines”,
Non-Photorealistic rendering(Siggraph 99 Course Notes),S.Green, ed., ACM Press
1999.

[HAZ 98] HASENFRATZ J.M.,
"Lancer de faisceaux en synthèse d'images",
Thèse de Doctorat, Université de Limoge,
Janvier 1998.

[HB 96] HARIKUMAR G., BRESLER Y.,
"Feature extraction for exploratory visualization of vector valued imagery",
IEEE Transactions on Image Processing, Vol.5, pp. 1324-1334,
September 1996.

[HE 01] HAINES E.,
“Soft planar shadows using plateaus”.
Journal of graphics Tools, Volume 6, Issue 1, pp. 19-27.
2001

[HH 84] HECKBERT P. S., HANRAHAN P.,
“Beam tracing polygonal objects”,
Computer Graphics (SIGGRAPH’84 Proceedings), Vol. 18 No 3, pp. 119-127,
July 1984.

[HZ 00] HERTZMANN A. and ZORIN D.,
“Illustrating smooth surfaces,”
Proceedings Siggraph 00, Computer Graphics (Proceedings Ann. Conf. Series),
S.N. Spencer, ed., ACM Press, pp. 517-526.
2000

[IS 03] ISENBERG T. et al.,
"A Developper’s guide to silhouette algorithms for polygonal models",

IEEE Computer Graphics and Applications, Volume 23, Number 4, pp. 28-37,
July/August 2003.

[JAW 84] JANSEN F.W., VAN WIKJ J.J.,
"Previewing techniques in raster graphics",
EUROGRAPHICS'84,
September 1984

[JC 01] JOHNSON D. E., COHEN E.,
“Spatialized normal cone hierarchies”,
Symposium on interactive 3D Graphics, pp.129-134.
In 2001 ACM, ISBN 1-58113-292-1
March 01

[JPT 06] JAUBERT B., PLEMENOS D., TAMINE K.,
“Techniques for off-line exploration using a virtual camera”,
International Conference 3IA’2006, Limoges (France),
May 23 – 24, 2006.

[JPGT 05] JAUBERT B., PLEMENOS D., GRASSET G., and TAMINE K.,
"Intelligent visibility-based 3D scene processing techniques for computer games",
In GraphiCon’05, Novosibirsk (Russia),
June 2005.

[JRP 02] JENSEN C. G., RED W. E., and PI J.,
“Tool selection for five axis curvature matched machining”.
Computer-aided Design, Volume 34, Number 3, pp. 251-266, ISSN 0010-4485.
March 2002

[KA 85] KAPLAN M.R.,
“Space tracing, a constant time ray tracer. State of the art in image synthesis”,
SIGGRAPH’85 Course notes, Vol. 11,
July 1985

[KAG 79] KAY D. S., GREENBERG D.,
“Transparency for computer syntesized images“
CACM. 4,

1979

[KH 95] KEATES M., HUBBOLD R.J.,
“Interactive ray tracing on a virtual shared-memory”,
Computer graphics forum, Vol. 14 No. 4, pp. 189-202,
Oct. 1995.

[KK 86] KAY T. L., KAJIYA J.
“Ray tracing complex scenes”.
Computer Graphics, Vol. 20 No 4, pp. 269-278,
Août 1986.

[KK 88] KAMADA.T, KAWAI.S,
"A simple method for computing general position in displaying three-dimensional objects",
Computer Vision, Graphics and Image processing, Vol. 41,
1988.

[LC 79] LANE J.M., CARPENTER L. C.,
"A generalized scane line algorithm for the computer display of parametrically defined surfaces",
Computer Graphics and Image Processing CGIP(11), No. 3, pp. 290-297,
November 1979.

[LCWB 80] LANE J.M., CARPENTER L. C., WHITTED T., BLINN J. F.,
"Scan line methods for displaying parametrically defined surfaces",
CACM, Vol. 23, no 1,
January 1980.

[LGM 00] LEE W., GU J., and MAGNENAT-THALMANN N.,
“Generating animable 3D virtual humans from photographs »
Computer Graphics Forum,Vol. 19, No. 3, ISSN 1067-7055.
August 2000

[LHS 03] Li B., Hu H., Spacek L.,
“An adaptive color segmentation algorithm for sony legged robots”,
21st IASTED International Conference on Applied Informatics, Innsbruck, Austria, Feb 2003

[LUC 91] LUCAS M.,
"Parallélisme et synthèse d'image",

TSI, Vol. 10, No. 3,
Juillet 1991.

[MAH 72] MAHL R.,
"Visible surface algorithms for quadric patches",
IEEE Transactions on Computers, Vol. c-21, No 1,
January 1972.

[MAN 96] MANDUCA A.,
"Multispectral image visualization with non linear projections",
IEEE Transactions on Image Processing, Vol. 5, pp. 1486-1490,
October 1996.

[MC 00] MARCHAND E. and COURTY N.,
"Image-based virtual camera motion strategies",
In S. Fels and P. Poulin, editors, Graphics Interface Conference, GI'00, pp. 69–76, Montreal, Quebec, Morgan Kaufmann,
May 2000.

[MLM 95] MITRA S. K., LI H., MANJUNATH B. S.,
" Multisensor image fusion using the wavelet transform",
Computer Vision, Graphics, and Image Processing, Vol. 57, No. 3, pp. 627-640,
1995.

[MULL 86] MULLER H.,
"Image generation by space sweep",
Computer Graphics Forum,Vol 5, pp. 189-196,
1986.

[ND 03] NIENHAUS M. and DOELLNER J.,
"Edge enhancement- An algorithm for real time non photorealistic rendering" Journal of WSCG 2003, Plzen Czech Republic,Vol. 11, No. 1, ISSN 1213-6972.
2003

[NEW 74] NEWELL M. E.,
"The utilization of procedure models in digital image synthesis",
Ph.D Thesis, University of Utah. Also UTEC-CSc-76-218 and NTIS AD/A 039 008/LL,
1974.

[NG 00] NEHAB D. and GATTAS M.,
“Ray path categorization”,
Proceedingss of the Brazilian Symposium on Computer Graphics and Image Processing -SIBGRAPI, Gramado, Brazil, pp. 227-234,
2000.

[NNS 72] NEWELL M. E., NEWELL R.G., SANCHA T. L.,
"A new approach to the shaded picture problem",
Proceeding ACM National Conference,
1972.

[OZ 06] OLSON M. and ZHANG H.,
“Silhouette extraction in hough space”,
Computer Graphics Forum (special issue on Eurographics 2006), Vol. 25, N0. 3, pp.273-282.
2006

[PDB 00] BARRAL P., DORME G. PLEMENOS D.,
“Scene understanding techniques using a virtual camera”,
Short paper, Eurographics’2000, Interlagen (Switzerland),
August 20 - 25, 2000.

[PDB 99] BARRAL P., DORME G., PLEMENOS D.,
“Visual understanding of a scene by automatic movement of a camera”
GraphiCon’99, Moscow (Russia),
August 26 - September 3, 1999.

[PIT 93] PITOT P.,
"The Voxar Project (Parallel Ray-Tracing),"
IEEE Computer Graphics and Applications, vol.13, no.1, pp. 27-33, Jan/Feb,1993

[PLE 87] PLEMENOS D.,
"Techniques de raffinement séléctif pour la visualisation réaliste de scènes tridimentionnelles",
Revue Intrernationnale de CFAO et d'Infographie, Vol. 1, No 4,
1987.

[PLE 91] PLEMENOS D.,
“Contribution à l'étude et au développement des techniques de modélisation, génération et visualisation de scènes”. Le projet

Multiformes
Thèse de Doctorat d'état, Université de Nantes, Nantes (France),
November 1991.

[PLE 93] PLEMENOS D.,
"Exploring virtual worlds: Current techniques and future issues",
In International Conference GraphiCon’2003, Moscow (Russia),
September 2003.

[PB 96] PLEMENOS D., BENAYADA M. ,
"Intelligent display in Scene Modeling. New techniques to automatically compute good views",
Graphicon'96,Saint Petersburg,
July 1996

[PLE 98] PLEMENOS D.,
"Images et outils graphiques pour la CAO", support de cours
Université de Limoges,
Octobre 1998.

[PM 77] PETTY J.S. and MACH K.D.,
"Contouring and hidden-line algorithms for vector graphic displays",
Air Force Applied Physics Lab. Rep., AFAPL-TR-77-3, ADA 040530,
January 1977

[PR 89] Priol T.,
“Lancer de rayon sur des architectures parallèles : étude et mise en oeuvre”. Thèse Soutenue À L'université Rennes I, pp.136,
1989.

[PSF 04] PLEMENOS D., SBERT M., and FEIXAS M..
"On viewpoint complexity of 3D scenes".
In International Conference GraphiCon’2004, Moscow (Russia),
September 2004.

[RC 99] RASKAR R.and COHEN M.,
“Image precision silhouette edges,”
Proceedings 1999 ACM Symp. Interactive 3D Graphics, S.N. Spencer, ed., ACM Press, 1999, pp. 135-140.11.
1999.

[RE 92] ROSSIGNAC J.R. and VAN EMMERIK M.,
"Hidden contours on a frame-buffer,"
Proceedings 7th Eurographics Workshop Computer Graphics Hardware, Eurographics, pp. 188-204.
1992

[RGW 05] RASCHE K., GEIST R., WESTALL J.,
"Re-coloring images for gamuts of lower dimension",
Eurographics 2005, Vol. 24, No. 3,
2005

[RIS 96] RIS M. P.,
"Parallélisation du lancer de rayon par évaluatuion dynamique de la topologie de la scène",
Thèse de doctorat à l'Université de Franche-Comté,
Septembre 1996.

[RK 82] Rosenfeld A . and Kak A .C .
"Digital picture processing",
Academic Press, Inc, Second edition, Vol. 2, New York,
1982 .

[RO 82] ROTH S.D.,
"Ray casting for modeling solids",
Computer graphics and image processing, Vol. 18, pp. 109-144,
1982.

[ROB 63] ROBERTS L.G.,
"Machine perception of three dimentional solids",
MIT Lincolin Lab. Rep., TR 315,
May 1963.

[ROG 97] ROGERS D. F.,
"Algorithmes pour l'infographie",
Ediscience Internationnal,ISBN 2-84074-039-7,
1997

[RP 89] RUSTAGI P.,
"Silhouette line display from shaded models",
Iris Universe, p. 42-44.

Fall 1989

[RR 01] RASKAR R.,
“Hardware support for non photorealistic rendering,”
Proceedings 2001 Siggraph/Eurographics Workshop on Graphics Hardware, ACM Press, pp. 41-46
2001

[RW 80] RUBIN S., WHITTED T.,
“A three dimensional representation for fast rendering of complex scenes”.
Computer graphics, Vol. 14 No 3, pp. 110-116,
July 1980.

[RUS 05] RUSINKIEWICZ S. et al.,
“Line drawings from 3D models”,
International Conference on Computer Graphics and Interactive Techniques, ACM Siggraph 2005 Course 7, Los Angelos, California, Number 1,
July 2005

[STN 87] SHINYA M., TAKAHASHI T., NAITO S.,
“Principles and applications of pencil tracing”,
SIGGRAPH’87, Vol. 21 No 4, pp. 45-54,
July 1987.

[SIR 74] SUTHERLAND, IVAN E., SPROULL, ROBERT F., AND SCHUMACKER R.A.,
"A characterization of ten hidden surface algorithms",
Computing Survey, Vol. 6, pp. 1-55,
1974

[SLM 02] STOMPEL A.,LUM E. B., and MA K.L.,
"Feature-enhanced visualization of multidimensional, multivariate volume data using non photorealistic rendering techniques",
Proceedings of Pacific Graphics 2002 Conference,
2002.

[SOK 06] SOKOLOV D.,
"Exploration différée de mondes virtuels sur internet",
Thèse de doctorat de l'Université de Limoges,

12 Décembre 2006.

[SOMBK87] SCHMIEDL U., ORTHENDAHL D. A., MARK A. S., BERRY I., KAUFMAN L.,
"The utility of principal component analysis for the image display of brain lesions: A priliminary, comparative study",
Magnetic Resonance In Medecine,Vol. 4 pp. 471-486,
1987.

[SP 05] SOKOLOV D., PLEMENOS D.,
“Viewpoint quality and scene understanding”,
VAST 2005 Eurographics Symposium Proceedingss, pp. 67-73, Pisa, Italy
2005

[SPT 06] SOKOLOV D., PLEMENOS D., TAMINE K.,
“Methods and data structures for virtual world exploration”,
The Visual Computer,
2006.

[SDB 85] SPEER L. R., DEROSE T. D.,BARSKY A.,
“A theoretical and empirical analysis of coherent ray tracing”,
Computer Generated Images: The State of the Art,
1985.

[ST 90] SAITO T. and TAKAHASHI.T.
“Comprehensible rendering of 3D shapes”,
Computer Graphics (SIGGRAPH ’90 Proceedings), Vol. 24, pp. 197–206,
August 1990.

[TIP 64] TIPPET J.T. et al. (eds.),
"Optical and electo-optical information processing",
MIT Press, Cambridge, pp. 159-197,
1964.

[VSFH 02] VAZQUEZ P. P., SBERT M., FEIXAS M. and HEIDRICH W.,
"Image-based modeling using viewpoint entropy",
In Computer Graphics International, pp. 267–279,
2002.

[VP 03] VAZQUEZ P.-P.,
"On the selection of good views and its application to computer graphics".
PhD Thesis, Barcelona (Spain),
May 26, 2003.

[VS 03] VASQUEZ .P.P, SBERT . M,
"Automatic indoor scene exploration",
3IA Computer graphics and Artificial intelligence Conference 2003, Limoges (France), pp. 13-24,
May 2003.

[WA 77] WEILER K. and ATHERTON P.,
"Hidden surface removal using polygon area sorting",
Computer Graphics (Proc. SIGGRAPH), Vol. 11, pp. 214-222,
1977

[WAR 69] WARNOCK C.S.,
"A hidden surface algorithm for computer generated half-tone pictures"
University of Utah, Tra 15, CS Departement,
1969

[WAT 70] WATKINS G.S.,
"A real-time visible surface algorithm",
University of Utah, Computer Science Department, UTEC-CSC-70-101, NTIS AD 762 004
June 1970

[WHI 80] WHITTED T.,
"An improved illumination model for shaded display",
Communications of the ACM, Vol. 23 No 6, pp. 343-349,
June 1980.

[WLS 00] WU Y., LIU Q., SUANG T.,
"An adaptive self-organizing color segmentation algorithm with application to robust real-time human and localization",
Asian Conference on Computer Vision, Taiwan
2000

[WNDS 99] WOO M., NEIDER J., DAVIS T., SHREINER D.,
"OpenGL programming guide",

Third Edition, Addison-Wesley, ISBN 0-201-60458-2,
1999.

[YMS 83] YAU , MANN-MAY , SRIHARI S.N.,
“A hierarchical data structure for multi-dimensional digital images”
Communication of ACM, Vol. 26 No 7, p. 504-515,
July 1983.

[YZW 03] YUNFANG G., ZHIGENG P., and WEIWEI X.,
"Vision-based path planning in intelligent virtual environment",
In International conference 3IA’03, pp. 25–33, Limoges (France),
May 2003.

Liste des Figures