UNIVERSITE DE LIMOGES

ECOLE DOCTORALE « Sciences, Technologie, Santé »
FACULTE DES SCIENCES ET TECHNIQUES

Thèse n°XX-XXXX

THESE

Pour obtenir le grade de
DOCTEUR DE L'UNIVERSITE DE LIMOGES
Discipline / Spécialité : Informatique

Présentée et soutenue par
Benoît JAUBERT

Juillet 2008

Outils pour les jeux sur ordinateur : Prise de connaissance de scènes 3D

Thèse dirigée par Dimitri PLEMENOS

JURY

Rapporteurs :

M. Marc DANIEL : Professeur de l'Université de la Méditerranée

M. Yves DUTHEN : Professeur de l'Université de Toulouse

Examinateurs :

M. Djamchid GHAZANFARPOUR : Professeur de l'Université de Limoges

M. Georges MIAOULIS : Professeur du TEI d'Athènes

M. Dimitri PLEMENOS : Professeur Emérite de l'Université de Limoges

Remerciements

Je tiens tout d'abord à remercier le directeur de cette thèse, M. Dimitri Plemenos pour m'avoir guidé, encouragé, conseillé tout au long de ces années de thèse et pour m'avoir poussé à terminer ce mémoire. Je lui souhaite une bonne retraite et des jours paisibles.

Je tiens aussi à remercier l'encadrement du laboratoire pour leurs enseignements et pour leur écoute tout au long de cette période. Dans cette optique je veux principalement citer M^{me} S. Simonet, et Messieurs M. Gauthier, O.Terraz, P. Barral qui ont été mon environnement proche et mon conseil pour ma vie au laboratoire.

Je passe une dédicace spéciale à toutes les jeunes personnes que j'ai eu le plaisir de côtoyer durant ces quelques années, étudiants et/ou doctorants, P. Rousseau, A. Ahmad, G. Guimberteau mais aussi les autres Y. Coulais, S. Desroches, etc. Merci aux étudiants que j'ai pu croiser tout au long des enseignements à qui j'espère avoir transmis des connaissances solides.

Je remercie grandement Messieurs Yves Duthen et Marc Daniel d'avoir accepté d'être rapporteurs et Messieurs Djamchid Ghazanfarpour, Georges Miaoulis et Dimitri Plemenos pour leur rôle d'examinateurs. Merci de me faire l'honneur de participer à cette thèse.

Je veux aussi remercier mes collègues qui m'ont permis de terminer ce document en faisant en sorte de m'éviter, autant que possible, les heures de nuit, les éventuelles astreintes et en me permettant de passer des journées agréables au travail, S. Le Luyer, E. Gasnier, A. Bechade, A. Poulier, R. Leprince, J. B. Lecluse, T. Saffon puis maintenant B. Champion, D. Harper, L. Penot et S. Tharaud.

Je remercie mes parents pour l'éducation qu'ils m'ont donnée et pour leur soutien, ainsi que ma famille et mes amis qui m'ont permis de décompresser et d'oublier les mauvaises périodes lorsque le besoin s'en faisait sentir.

Enfin, je tiens à conclure en remerciant mes proches et tout particulière M^{elle} Barbot d'Hauteclaire Sophie, sans qui cette thèse ne serait jamais arrivée à terme et qui de part ses encouragements répétés ne m'a pas laissé la possibilité de lâcher prise. Je saurai être aussi insistant lorsque les rôles seront inversés.

Résumé

L'évolution des jeux vidéo est telle que la plupart d'entre eux se retrouvent maintenant face au problème de la gestion de points de vue et de trajectoires. Avec le développement des jeux en trois dimensions et des cinématiques utilisant le moteur graphique du jeu, il arrive fréquemment que les angles de vue choisis ne soient pas les meilleurs possibles. Ainsi, si les moyens mis en oeuvre pour améliorer les vitesses d'affichage et de calcul, la qualité des graphismes, la fluidité et la maniabilité du jeu sont arrivés à une qualité qui ne souffre que peu de critiques, le travail à faire sur les placements de caméra et de points de vue est encore loin d'être terminé.

Nous répertorions dans un premier temps les travaux existants, portant sur le calcul de bons points de vue et la création de trajectoire, dans le domaine de l'informatique et de la robotique. Dans le second chapitre, nous présentons les méthodes et techniques que nous proposons afin de déterminer et d'évaluer de bons points de vue pour un monde ou des objets en trois dimensions en se basant sur diverses notions telles que les poylgones, les objets ou les matériaux. Le troisième chapitre introduit des techniques afin de découper des scènes en objets et des objets en sous-objets permettant d'intégrer cette notion lorsque celle ci n'est pas présente au sein d'une scène. Le quatrième et dernier chapitre décrit la création de trajectoires sur sphère englobante, ou en monde ouvert, et fournit des pistes pour l'utilisation de ces méthodes en temps réel.

Mots-clés : point de vue, trajectoire, monde virtuel, scène, 3D, exploration, position de caméra.

Abstract

The evolution of video games is such that most of them are now facing the problem of managing views and trajectories. With the development of games in three dimensions and cinematic using the graphics engine of the game, it frequently happens that the angles are not chosen the as well as possible. Thus, if the means implemented to improve the speed display and calculation, quality graphics, fluidity and handling of the game came to a quality that suffers little criticism, the work to be done on camera placements and points of view is still far from over.

We first describe the previous works, dealing with viewpoints computing and trajectory creation. In the second chapter, we present methods and technics to determine and evaluate goods points of view for virtual worlds ou three dimensionnal objects, basing on as various notions as polygons, objects or materials. The third chapter introduces techniques to cut scene into objects and objects into parts allows this notion to be integrate in scene inwhich it not be present. The fourth and last chapter describes the creation of trajectory on surrounding sphere and in open worlds and gives indices to use theses methods in real time environment.

Keywords : viewpoint, trajectory, virtual world, scene, 3D, exploration, camera position.

Table des matières

Table des figures

Introduction

Introduction

Les jeux vidéo apparurent dans les bars. Le premier d'entre eux, « Space War », fût créé en 1961 mais reste néanmoins expérimental. « Pong », sorti en 1973 par Atari, est considéré comme le premier « vrai » jeu vidéo. Vinrent ensuite « Space Invaders » en 1978 et le célèbre « Pacman » en 1980 à une période qui marque aussi les débuts des consoles de jeu.

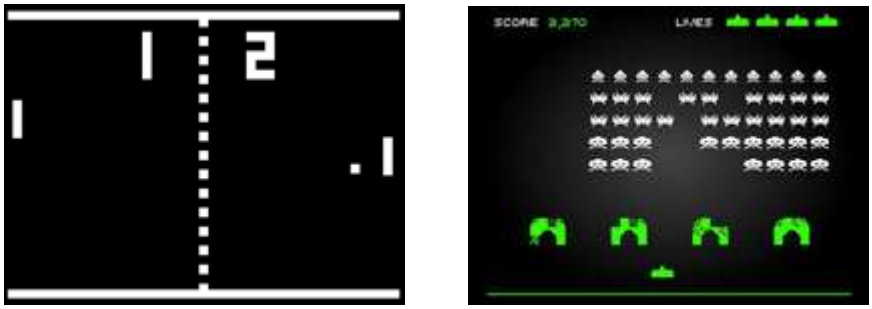


Figure 1 : Les premiers jeux vidéo (Pong et Space Invaders)

Dès lors, les programmes comme les machines n'ont cessés de se développer. Avec des ordinateurs de plus en plus puissants apparaissent des jeux de plus en plus recherchés et étudiés. D'abord en deux dimensions, les rendus deviennent peu à peu en trois dimensions et les problèmes posés par la conception de ces jeux vont croissant.

Figure 2 : Des jeux récents (Halo 2 et Crysis – captures éditeur)

Avec les améliorations des machines et des programmes, il devient difficile et coûteux de développer un jeu vidéo et il faut une équipe complète pour y arriver, là ou auparavant un seule personne suffisait. Les problèmes d'affichage commencent eux aussi à être plus nombreux et le développement d'outils tels que les moteurs 3D devient primordial.

A. Fond du sujet et motivations

L'évolution des jeux vidéo est telle que la plupart d'entre eux se retrouvent maintenant face au problème de la gestion de points de vue et de trajectoires. Avec le développement des jeux en trois dimensions et des cinématiques utilisant le moteur graphique du jeu, il arrive fréquemment que les angles de vue choisis ne soient pas les meilleurs possibles. Ainsi, si les moyens mis en oeuvre pour améliorer les vitesses d'affichage et de calcul, la qualité des graphismes, la fluidité et la maniabilité du jeu sont arrivés à une qualité qui ne souffre que peu de critiques, le travail à faire sur les placements de caméra et de points de vue est encore loin d'être terminé.

Figure 3 : Une capture de cinématique (Neverwinter Nights 2)

Il est en effet encore fréquent d'apercevoir au détour d'une cinématique des objets masquant la zone intéressante ou des personnages peu visibles, ce qui est facilement imputable à de mauvais choix faits sur le placement des caméras.

Il devient donc nécessaire de se pencher sur ces questions et d'avoir des techniques pertinentes qui permettront d'améliorer ce qui est un élément important d'un jeu vidéo.

De plus, la course à l'esthétisme sur un bon nombre de jeux fait qu'il est maintenant nécessaire d'optimiser la modélisation des personnages et des objets. Le principe est simple et peut-être énoncé de la manière suivante : un élément éloigné n'a pas besoin d'autant de détails que ce même élément au premier plan. Des méthodes ont ainsi été développées pour faire des simplifications sur les objets. Certaines d'entre elles nécessitent notamment des choix de points de vue pour choisir les zones plus aisément simplifiables en modifiant au minimum le rendu.

Le calcul de bons points de vue sur un objet devient alors une part importante du processus de simplification car le choix de ceux-ci peut clairement faire la différence. Ils peuvent ensuite servir de points de comparaison entre l'objet initial et l'objet simplifié pour savoir si les deux sont suffisamment proches l'un de l'autre pour que la simplification puisse être qualifié de pertinente et satisfaisante.

Un autre problème est celui de la modélisation et du rendu basés images. Il faut choisir un nombre minimal de bons points de vue pour modéliser une scène.

B. Problématique du sujet

L'exploration (action de découvrir une région ou un site particulier et d'en observer ses caractéristiques) n'est pas si facile à appréhender que ce que nous pourrions initialement croire. Cette notion est très variable en fonction des domaines qui l'utilisent et les critères qui la dirigent peuvent varier dans des proportions importantes. Un botaniste ne sera pas intéressé par les mêmes points de vue qu'un paysagiste, par exemple, et ces deux protagonistes s'intéressent pourtant aux plantes.

Il convient donc de bien spécifier les différents facteurs et les différentes notions que nous allons prendre en compte et utiliser, sans oublier dans quelles circonstances nous nous plaçons. Certains domaines peuvent d'ailleurs partager des problématiques se rapprochant ou se confondant même par moment. Nous citerons pour exemple la robotique, la synthèse d'images et la conception assistée par ordinateur.

1. Définition d’un bon point de vue

La définition d'un bon point de vue est une chose assez subjective et est fortement dépendante du besoin inhérent de l'utilisateur. En effet le regard porté sur une scène ne sera pas le même en fonction de la personne concernée.

Prenons l'exemple d'un objet simple : une fleur dans un vase. Un botaniste préfèrera se focaliser sur la plante, sans se préoccuper du reste alors qu'un artisan potier s'intéressera plus facilement au travail du contenant. Le bon point de vue n'est donc pas universel et c'est pourquoi nous nous attacherons par la suite à spécifier dans quels cas nous nous plaçons pour dire que telle ou telle chose est meilleure que telle autre.

Dans le cadre de jeux vidéo, il est important de voir à tout moment l’objet ou les objets qui nous intéressent. Nos méthodes essaieront donc de prendre en compte cet élément important pour calculer nos points de vue.

2. Notion de caméra virtuelle

La notion de caméra virtuelle est importante car c'est elle qui va décider de notre format de point de vue.

Le premier avantage d'une caméra virtuelle par rapport à une caméra réelle est qu'il est facile de repérer son centre optique au sein de l'espace car ce n'est qu'un point. Le second avantage est que la caméra virtuelle peut faire fi de toutes les contraintes de poids, support, etc. qui peuvent empêcher le placement réel.

Nous ne sommes donc soumis à aucune autre contrainte que la visibilité dans cette recherche du bon placement. La première information nécessaire est la position de ce centre optique que nous nommerons position de la caméra. Une fois une caméra placée à un endroit spécifique, il reste encore à choisir vers quel endroit l'orienter. De la même façon que pour sa position, son regard sera défini par un point de l'espace. Ce point n'est bien sur pas unique et indique la direction de la caméra plutôt qu'un point précis.

Enfin, une fois que nous avons placé la caméra et choisi vers quel endroit elle devait regarder, il nous reste à choisir où situer le haut de celle ci. En effet, contrairement à notre monde où la gravité nous sert de repère, un objet ou même un monde virtuel ne possède pas de sens. De la même manière que pour la direction de vision, la direction du vecteur "haut" se fait en indiquant un point de référence. Ce point de référence sera donc variable en fonction de la situation et du type de jeu dans lequel nous nous trouvons.

```
Caméra  ( PositionX, PositionY,  PositionZ,
          RegardX  , RegardY ,  RegardZ  ,
          HautX    , HautY   ,  HautZ )
```

3. Pertinence d'une trajectoire de caméra

Juger la pertinence d'une trajectoire de caméra n'est pas aussi simple que nous pourrions le penser au premier abord. Plusieurs contraintes et besoins peuvent entrer en compte et il est nécessaire d'optimiser les uns par rapport aux autres. Nous pouvons séparer deux types de contraintes sur le trajectoire : les contraintes fixes (il faut ou il ne faut pas) et les contraintes variables (il serait préférable de). Ce second type de contrainte est bien entendu celui qui va nous permettre d'adapter la trajectoire à nos besoins.

Un premier critère que nous pouvons retenir sera dépendant de la distance parcourue. L'idéal étant un trajet minimisant la distance entre les différents points de vue. Ce critère entre dans la catégorie des variables.

Un second critère important est celui de la gestion d'obstacle. Nous considèrerons qu'une trajectoire optimisée devra éviter de passer à travers les obstacles. Il s'agit donc d'un type de contraintes fixe.

Nous de détaillerons pas ici les différents critères utilisables que nous verrons plus tard mais nous devons bien comprendre que la complexité de création d'une trajectoire va augmenter significativement avec le nombre de critères utilisés.

Il va donc être nécessaire encore une fois de bien désigner les tenants et les aboutissants pour savoir quels critères il est important d'utiliser à quels moments. Une trajectoire lors d'une cinématique, par exemple, aura des besoins différents suivant le fait que les personnages se déplacent ou non et si leur zone de mobilité est réduite.

C. Présentation du mémoire

Dans un premier temps, nous allons présenter l'existant et nous attacher à décrire les différentes méthodes existantes dans le domaine. Nous verrons à cette occasion ce qui a été fait dans la recherche et le calcul de bons points de vue ainsi que la génération de trajectoire de caméra.

Une fois ce tour d'horizon terminé, nous présenterons nos méthodes permettant de définir et de choisir un ou plusieurs points de vue pour un monde virtuel. Nous évoquerons à ce moment différentes possibilités qui s'offrent à l'utilisateur en fonction des besoins de celui-ci.

Nous verrons ensuite des techniques permettant de découper un monde virtuel en sous parties, plus simples à traiter et permettant d'avoir des points de vue plus proches des entités et plus spécifiques à des régions données de l'espace.

Enfin, nous terminerons par la création de trajectoires optimisées dont le but sera d'offrir à l'utilisateur une vision globale de la scène. Nous verrons lors de cette partie comment générer une trajectoire, soit de manière incrémentale, soit en faisant des précalculs, en minimisant les temps de calculs.

Chapitre I
Etat de l’Art

I. Etat de l'Art

Le nombre de travaux sur le sujet qui nous intéresse reste encore aujourd'hui peu élevé. Les causes semblent évidentes. D'une part, le sujet est relativement récent : les scènes non complexes n'ont pas besoin de techniques d'exploration évoluées. D'autre part, ces méthodes ne sont devenues nécessaires que suite à la création d'Internet et au développement des mondes virtuels. Le nombre grandissant de jeux vidéo nécessitant des représentations de zones en trois dimensions a permis de s'intéresser de plus près à l'exploration et aux techniques connexes.

Les scènes sont standardisées et représentées sous la forme de polygones (nous trouverons principalement des triangles ou des carrés). A l'évidence, plus la scène est détaillée et plus elle contient de polygones. Nous ne développerons pas les différents effets de rendus au niveau des textures et des lumières puisqu'ils ne nous sont d'aucun intérêt pour l'étude de la scène et pour les différentes méthodes qui vous seront présentées par la suite.

Les premiers travaux de compréhension visuelle d'une scène furent publiés à la fin des années 1980 et le domaine ne cesse depuis de se développer. De plus, des éléments provenant d'autres domaines pourront également nous aider afin de compléter notre recherche. Différentes approches se présentent donc à nous.

Nous verrons tout d'abord un premier ensemble de techniques qui permettent le calcul d'un bon point de vue pour des scènes. Il s'agit d'offrir une vue satisfaisante qui permet de prendre en compte un maximum d'éléments tout en éliminant le plus grand nombre possible de problèmes qui pourraient occasionner une mauvaise compréhension de la scène.

Nous nous intéresserons ensuite aux techniques qui traitent des trajectoires et nous verrons à cette occasion des recherches faites sur les déplacements.

A. Le calcul de points de vue de qualité

Le calcul d'un bon point de vue est un sujet qui a donné lieu à très peu de travaux. Pourtant, le problème se révèle vraiment important dans toute visualisation et ce quel que soit

le type de support utilisé. Un point de vue correct va, en effet, nous permettre de bien analyser et de bien comprendre la structure d'une scène.

L'évolution du matériel ainsi que des représentations ont permis tout naturellement l'amélioration des techniques de calcul. Nous allons donc vous présenter certaines de ces méthodes.

1. Position de la caméra par les barycentres [Blinn88]

C'est en 1988 que Blinn nous présente une technique basée sur les barycentres. Son étude porte sur les calculs nécessaires au positionnement d'une caméra de sorte que les barycentres d'un vaisseau spatial et d'une planète soient projetés à des positions données de l'écran, et que la caméra soit à une distance fixe du vaisseau. L'utilisateur choisit certains paramètres tels que la distance de la caméra ou encore la position des objets à l'écran. La vue est ensuite calculée de sorte que les différents objets de la scène (en nombre restreint) soient projetés aux positions données. Ainsi le choix de la représentation de la scène tel qu'il l'envisage revient au départ à l'utilisateur.

Ces travaux seront ensuite repris puis étendus par ceux de Languénou [LBGC98], en appliquant une approche déclarative. L'utilisateur n'a pas besoin de donner de contraintes précises ni même de coordonnées écran précises mais seulement des régions dans lesquelles il souhaite que les objets se projettent. Ainsi la résolution des contraintes due à ce type de description peut soit aboutir à plusieurs solutions, soit ne pas aboutir dans le cas où les requêtes de l'utilisateur ne sont pas possibles.

2. Réduction des arrêtes [KK88]

Kamada et Kawai proposent en 1988 une autre approche en automatisant les calculs de bons points de vue, et ce, sans intervention de l'utilisateur. Pour ce faire, ces derniers utilisent une méthode qui minimise le nombre d'arrêtes superposées d'un objet car celles-ci nuisent à la bonne compréhension de la scène, principalement lors des rendus en fils de fer.

3. Approximation pyramidale [Col88]

Colin propose une méthode qui avait été développée pour des scènes modélisées par des arbres octaux. Le but de cette méthode était de calculer un bon point de vue pour une position de la scène représentée par un arbre octal. La méthode utilise un principe de calcul de direction par approximation afin de calculer une bonne direction de vision. Ce principe peut se décrire comme suit :

- choisir les trois meilleures directions de vision parmi les six directions correspondant aux trois axes de coordonnées qui passent à travers le centre de la scène.
- calculer une bonne direction dans la pyramide définie par les trois directions choisies, en prenant en compte l'importance de chacune de ces directions.

L'axe de vue est considéré comme meilleur qu'un autre si celui-ci permet de voir plus de détails qu'un autre. D'autre part, la méthode choisit un point de vue qui montre le plus grand nombre de pixels.

4. Surface visible [Col90]

Colin, en 1990, propose une méthode qui permet de choisir automatiquement une bonne vue afin d'observer une scène modélisée par un arbre octal. Cette méthode retient la direction qui permet de voir le plus grand nombre de voxels. Elle s'appuie sur des valeurs exactes ou bien sur des valeurs approchées.

5. Recherche adaptative d'une vue [Ple91][PB96]

En 1991, Plemenos s'intéresse aux scènes composées de polygones. Il présente une méthode basée sur la recherche adaptative d'une vue en fonction du nombre de polygones visibles et de la surface de projection maximale. Dans le cas où des propriétés de certains éléments sont connues, Plemenos présente également une méthode de mise en valeur. Cette méthode procède par découpage de l'espace en différentes régions dans lesquelles il est

possible de placer une caméra. Ces régions sont associées de manière générique à chaque propriété pour leur capacité à les mettre en évidence. Un objet est, par exemple, mieux perçu de côté si l'on cherche à montrer sa hauteur plutôt que si l'on choisit une vue de dessus.

Dans ses travaux, Plemenos [Ple91, PB96] propose une méthode itérative permettant le calcul automatique d'un bon point de vue. Pour cela, la scène est placée au centre d'une sphère et la surface de cette dernière représente l'ensemble des points de vue possibles. Puis, la sphère est divisée en huit triangles sphériques dont les sommets sont les intersections des axes avec la sphère. Chacun des six sommets est évalué (la fonction d'évaluation utilise un z-buffer et cherche à maximiser le nombre de polygones visibles) et chacun des triangles obtient la somme des valeurs de ses sommets respectifs. Le triangle désigné comme étant le plus intéressant est celui qui obtient la valeur la plus importante. Ensuite, le triangle sphérique qui est sélectionné est subdivisé récursivement. Ses trois sommets sont alors évalués et seul le sommet de plus grande valeur est conservé. La fonction est alors rappelée sur le sommet conservé et les milieux des segments du triangle issus de ce sommet.

Ex : si l'on désigne A comme étant le meilleur sommet, la fonction sera rappelée sur le triangle AB'C' :

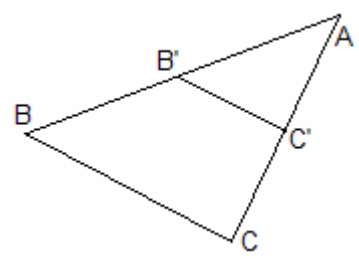


Figure 4 : Méthode des triangles sphériques

Un certain nombre de critères, comme le nombre d'itérations ou encore la surface du triangle permettent d'arrêter cette récursivité.

Le fait de choisir un bon point de vue permet de comprendre facilement une scène simple. Par contre, dans le cadre de scènes complexes, un seul point de vue n'est en principe pas suffisant. En effet, même si l'utilisateur a la possibilité d'observer la scène depuis plusieurs bons points de vue, cela ne suffit pas à assurer la compréhension de scènes complexes parce que les fréquents changements occasionnés par le passage d'un point de vue à un autre peuvent entraîner des confusions chez l'utilisateur. La meilleure solution pour bien comprendre des scènes complexes reste l'exploration automatique et intelligente à l'aide d'une caméra virtuelle. L'exploration doit éviter les changements brusques afin d'assurer à l'utilisateur une découverte confortable de la scène.

Au moment même où l'utilisateur découvre une scène à l'écran, il est possible que l'angle de vue choisi ne soit pas optimal afin de bien comprendre la scène. Deux possibilités d'offrent alors à lui.

Soit il procède à une exploration en temps réel. La scène est alors visitée pour la première fois. Le chemin suivi par la caméra est déterminé de façon incrémentale. Pour ce type d'exploration, il est important d'employer des techniques rapides d'exploration de sorte que l'utilisateur puisse comprendre la scène visitée en temps réel.

Soit il choisit une exploration différée. La scène est d'abord visitée par un programme d'exploration avant même d'être vue par l'utilisateur. Le programme analyse la scène tout en guidant le mouvement de la caméra de façon à sélectionner des points de vue intéressants. L'utilisateur pourra visiter la scène plus tard en suivant un chemin déjà déterminé. Pour ce type d'exploration, il n'est pas nécessaire d'avoir recours à des techniques rapides pour déterminer le chemin suivi par la caméra.

6. Vue générique [Free93] et [Free94]

Au cours des années 1993 et 1994, Freeman propose une autre technique qui permet de calculer un bon point de vue pour un objet composé de polygones. En regardant sous quelle forme l'objet a le plus de chances d'être observé, il suggère de choisir une vue typique qui

présentera le plus de stabilité. Ainsi, un cube se verra sous la forme de trois faces équilibrées plutôt que comme une seule et simple face.

Dans le domaine de la visualisation par ordinateur, Weinshall et Werman [WW97] présentent en 1997 une méthode permettant de calculer le degré de stabilité d'une vue et d'obtenir une vue typique en faisant une équivalence entre généricité et stabilité de vue. Ils définissent deux unités de mesure pour les vues. L'une estime la probabilité qu'une certaine vue d'un objet en 3D donné soit observée, cela peut s'utiliser pour identifier des vues typiques ou caractéristiques. L'autre mesure la faible différence d'images lorsque le point de vue est légèrement décalé, ce qui se révèle utile pour identifier des vues génériques.

7. Théorie de l'information

Sbert et al. [SFR+02, Vaz03, Feixas99, Feixas02, VFSH01] introduisent une approche basée sur une théorie de l'information : ils proposent de maximiser la fonction suivante appelée aussi entropie d'un point de vue :

$$I(p) = \sum_{i=0}^{N_f} \frac{A_i}{A_t} \times \log_2 \frac{A_t}{A_i}$$

Avec :

p est le point de vue,
Nf est le nombre de faces de la scène,
Ai est l'aire projetée de la face i,
A0 est l'aire projetée de l'arrière plan dans les scènes ouvertes,
At est l'aire totale de la projection.

La fonction utilisée est celle de l'entropie de Shannon dans laquelle les aires projetées des faces sont prises en tant qu'une variable aléatoire discrète. L'entropie maximale est obtenue lorsqu'un certain point peut voir toutes les faces avec la même aire projetée relative Ai/At. S'il est impossible de voir l'arrière plan depuis le point de vue, cela signifie que l'une des faces a une entropie maximale de log2 Nf. Sbert essaie, tout en optimisant la valeur de

l'entropie dans les images, de récupérer le plus grand nombre de faces sous la meilleure orientation possible.

Sbert [SPFG05] propose un algorithme basé sur la distance de Kullback-Leibler qui a pour but de trouver l'ensemble minimal et représentatif des vues pour un objet ou pour une scène donnée.

8. Visibilité de mailles

Chang Ha Lee [LVJ05] introduit l'idée de visibilité de mailles comme mesure d'importance régionale locale pour des maillages graphiques. La visibilité de mailles y est définie indépendamment de l'échelle en utilisant un opérateur d'environnement sur des courbes de poids Gaussiens. La mesure de l'importance inspirée de la perception humaine, calculée par le système de visibilité de mailles offre de meilleurs résultats en comparaison à des mesures purement géométriques des formes, telles que les courbures.

9. Etude de préférences utilisateurs

Blanz [BTB99] mène des études auprès d'utilisateurs afin de déterminer les facteurs influençant les vues préférées pour les objets en trois dimensions. La conclusion est que la sélection d'une vue est le résultat d'interactions complexes entre tâche, géométrie d'objets et familiarité avec ces objets. Les recherches démontrent que la visibilité (et l'occlusion) de parties saillantes d'un objet est l'un des facteurs déterminants dans le choix de la vue.

C'est au début des années 2000 que Gooch [GRM01] met au point un système qui utilise des principes inspirés des arts ainsi que plusieurs des facteurs suggérés par Blanz [BTB99] afin de calculer automatiquement des points de vue initiaux pour des objets en trois dimensions. De tels systèmes peuvent grandement bénéficier de modèles purement géométriques.

10. Utilisation de l'éclairage [MC00]

Concernant une vue de la scène observée qui dispose d'un éclairage optimisé selon une appréciation photographique, Marchand et Courty [MC00] proposent une méthode incrémentale qui permet le déplacement de la caméra jusqu'à ce que celle-ci se situe à la fois dans l'alignement de la scène et d'une lumière.

11. Degré de courbure [Sok06]

Sokolov [SP05][SPT06][Sok06] et [SP08], propose une nouvelle approche d'évaluation d'un point de vue en générant un graphe analytique de visibilité. Ce dernier n'est plus la simple évaluation d'un ensemble de point de vue mais la projection des faces visibles sur un plan et permet de ne pas avoir à discrétiser les zones pouvant être considérées comme intéressantes. De plus, un nouveau critère pour estimer la qualité d'un point de vue nous est présenté, considérant la courbure totale d'une surface visible comme étant une quantité d'information pour un point de vue.

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

F(p) est l'ensemble des faces visibles à partir du point de vue p
P(f) est l'aire projetée de la face f
V(p) est l'ensemble des points visibles de la scène à partir de p
α(v) est l'ensemble des angles adjacents au point v

L'avantage de cette méthode est son invariance à n'importe quelle subdivision de la scène qui maintienne la topologie. En effet, si une face est subdivisée en polygones, toutes les arrêtes et sommets internes seront ignorés du fait des angles nuls.

12. Récapitulatif

Les différentes approches que nous venons de voir ont des atouts variables en fonction du domaine d'application qui va les utiliser.

Le calcul de surface visible, le calcul de vue générique, la visibilité de maille et la préférence utilisateurs sont plutôt des méthodes orientées pour la Conception Assistée par Ordinateur ou pour le design.

L'utilisation du degré de courbure bien que pertinente pour la CAO prend une dimension encore supérieure lors de l'exploration de scènes non animées, comme par exemple, pour des visites virtuelles.

Dans le cadre des jeux vidéo, les méthodes les plus intéressantes à utiliser sont celles de la recherche adaptative et de la théorie de l'information. Ces deux méthodes peuvent être appliquées à des scènes et quelques légères modifications peuvent permettre de les étendre aux scènes animées.

B. Calcul de trajectoire(s) de caméra

Le calcul de trajectoire(s) est un sujet qui est la source de beaucoup de réflexion et de travaux surtout dans le domaine de la robotique. Ce qui nous intéresse tout particulièrement en matière de robotique consiste à trouver un chemin de robot ou de caméra dans une scène jusqu'à une position finale connue.

Les chercheurs travaillent depuis longtemps sur le calcul d'un chemin optimal dans un graphe d'états. Pour cela, l'algorithme de Dijkstra et l'algorithme A* sont les plus utilisés. Attention, notons au passage que pour un espace de dimension 2 ou 3, ces méthodes nécessitent quelques modifications avant d'être appliquées.

Le domaine de la robotique soulève également le problème du suivi d'un robot mobile. Dans ce cas de figure, une caméra ou bien un autre robot se déplace afin d'assurer toujours une bonne visibilité du robot qui est mobile.

Toujours en matière de robotique, la méthode proposée par [MC00] permet de suivre le robot mobile à l'aide d'un certain nombre de stratégies tels que l'évitement d'obstacles, le maintien d'une distance raisonnable avec le mobile ou bien encore un contrôle de la visibilité à partir d'une approche orientée image. La méthode peut fonctionner en temps réel dans les jeux vidéo ainsi que dans tout autre environnement réactif.

Des produits commerciaux tels que Quick Time VR [Che95] offrent des panoramas. Il s'agit d'images de 360° prises à partir d'une seule position. Par ailleurs, l'utilisateur est restreint aux mouvements de rotation et de zoom de la caméra, il est impossible de naviguer librement et de plus, les informations présentées dépendent des photos initiales. De plus, l'utilisateur doit sauter d'un point de vue à un autre et il est possible d'être induit en erreur, de perdre la notion de structure de la scène après plusieurs sauts.

Ces différents travaux nous permettent d'élucider un bon nombre de problèmes qui se posent à nous. Par exemple : comment placer une caméra pour obtenir la vision la plus intéressante de la scène ?

L'utilisateur doit toutefois renseigner le programme sur les éléments qu'il souhaite voir dans la scène comme c'est le cas dans [MC00].

Ces méthodes peuvent facilement s'appliquer dans le cas où l'utilisateur est déjà bien renseigné sur une partie de la scène. Dans le cas où l'utilisateur n'a aucune connaissance ni aucun détail sur la présentation de la scène, il est alors nécessaire de lui fournir un maximum de renseignements de manière externe qui lui permettront de choisir les parties de la scène qu'il souhaite observer dans le détail.

Le fait que la scène soit partitionnée de façon distincte est impératif dans la plupart des méthodes qui nous sont proposées. Or, il nous est impossible d'avoir recours à ces méthodes puisque les informations nécessaires ne sont pas présentes sur les scènes utilisées.

Bien que toutes ces méthodes ne nous soient d'aucune utilité si nous devions les manipuler en l'état, nous pouvons néanmoins avoir recours à certaines des approches qu'elles utilisent.

1. Une approche heuristique [Don87]

L'espace des positions est discrétisé et remplacé par une grille régulière dans laquelle les chemins envisagés sont des séquences de points adjacents sur cette même grille. La recherche de la trajectoire est en fait guidée par un champ de potentiel (une fonction définie dans l'espace) qui renvoie une valeur minimale à la position d'arrivée.

[Kha86] propose de construire cette fonction à l'aide de sommes de champs attractifs (arrivée, points prioritaires) qui possèdent une valeur minimale à la position de destination et la valeur augmente avec la distance de ce point et de champs répulsifs (obstacles, points à éviter) qui eux possèdent une valeur nulle en toute position de l'espace suffisamment loin des obstacles de la scène et tend vers l'infini à leur proximité auxquels s'ajoute une valeur fonction de la distance par rapport à l'arrivée.

C'est en 1996 que Plemenos et Benayada [PB96] proposent une méthode heuristique qui étend la définition donnée par Kamada et Kawai. Cette méthode considère qu'un point de vue est bon s'il permet d'observer le plus grand nombre de détails possibles tout en minimisant la déviation. Plemenos et Benayada montrent dans leur étude que si nous considérons seulement la minimisation, les points de vue résultants peuvent cacher des informations importantes de la scène. A la fonction de maximisation, Plemenos et Benayada ajoutent un autre paramètre qui compte les détails observés. Cet autre paramètre est le nombre de faces visibles à partir d'un point de vue. D'après [PB96], la qualité d'un point de vue peut se calculer grâce à la formule suivante :

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

Avec :

C(p) est la qualité du point de vue p,
Pi(p) est le nombre de pixels correspondant au polygone i dans l'image obtenue à partir du point de vue p,

r est le nombre total de pixels dans la scène,
n est le nombre total de polygones dans la scène.
[x] est le plus petit entier supérieur ou égal à x.

2. Une approche probabiliste [BKL+96]

Dans le cadre d'une approche probabiliste proposée par [BKL+96], des positions sont tirées au hasard au sein même de l'espace de recherche. Pour chaque paire de position, l'algorithme recherche s'il est possible ou non de les joindre par un chemin direct. L'ensemble des positions ainsi que l'ensemble des chemins directs possibles constituent alors une carte routière à partir de laquelle il est possible d'effectuer des recherches. Partant du fait que les points de départ et d'arrivée sont connus à l'avance, le chemin est alors calculé de manière incrémentale jusqu'à ce que les deux points finissent par se rejoindre.

3. Une approche déclarative [Moun98]

L'approche déclarative permet, à partir d'un calcul automatique d'une trajectoire de caméra de procéder à la visite d'une scène. Mounier [Moun98] a recours pour cela à un algorithme génétique afin de construire une trajectoire qui traverse la scène. L'utilisateur doit partitionner cette scène en cellules et renseigner le programme sur l'intérêt qu'il porte à chacune d'entre elles. L'algorithme génétique calcule alors une trajectoire tout en tenant compte des intérêts précédemment établis par l'utilisateur.

L'application de Marchand et Courty [MC00] permet elle aussi la visite d'un musée par exemple. Il est possible de demander à ce que la caméra nous montre une série d'objets ou encore de tableaux. Notons au passage que la méthode de [MC00] permet aussi à l'utilisateur de décrire ses intentions d'après ce qu'il observe au cours de l'animation.

4. Création incrémentale de la trajectoire [Dor00]

Cette méthode repose sur deux étapes distinctes. Il s'agit tout d'abord de trouver un point de vue de départ (en utilisant la méthode des triangles sphériques vue dans [Ple91, PB96] qui permet le calcul automatique de bons points de vue dans le cadre d'exploration de scènes en temps réel). Une fois que nous avons déterminé un bon point de vue, la création de la trajectoire peut alors débuter.

Plemenos et al. et Dorme [BDP99, BDP00b, Dor01, BDP00a] ont proposé une méthode dans laquelle une caméra virtuelle se déplace en temps réel sur la surface de la sphère englobante du monde virtuel. L'exploration est en temps réel, la scène est examinée de façon incrémentale pendant l'observation. Tous les polygones du monde virtuel sont pris en compte à chaque étape de l'exploration. La méthode repose sur des techniques heuristiques dont voici la formule :

$$w(c) = \frac{v_c}{2} \times \left(1 + \frac{d_c}{p_c} \right)$$

Avec :

wc est le poids de la position courante
vc est la complexité du point de vue pour la scène, basée sur la surface projetée à l'écran et le nombre de polygones visibles
pc est le chemin parcouru par la caméra à partir du point de départ jusqu'à la position actuelle
dc est la distance du point de départ jusqu'à la position actuelle.

Dans le but d'éviter des retours rapides de la caméra au point de départ, l'importance d'un point de vue est inversement proportionnelle au chemin de la caméra.

Aussi, pour une direction du mouvement de la caméra, seuls trois nouveaux points de vue sont considérés lors du calcul de la position suivante.

- choix d'une direction de départ :

Pour son premier mouvement, la caméra peut choisir parmi les huit axes directionnels principaux : Nord, Nord-est, Est, Sud-est, Sud, Sud-ouest, Ouest, Nord-ouest. Le pas entre deux positions consécutives de la trajectoire est constant et défini arbitrairement.

Après évaluation de chacun de ces nouveaux points de vue, seul le meilleur d'entre eux est conservé et constitue la prochaine position de la caméra. Ce point nous permet de définir la direction que va suivre la caméra. Cette dernière va permettre de limiter les déplacements tout en assurant la fluidité de la trajectoire.

- choix d'une direction suivante :

Nous pouvons utiliser la même méthode pour choisir, soit la direction suivante, soit la direction de départ, à la différence près qu'il faut en plus tenir compte de la direction courante pour ne pas occasionner de changements brutaux dans la trajectoire.

- facteurs supplémentaires :

Il peut arriver que la trajectoire de la caméra devienne cyclique dans l'hypothèse où un point de vue fortement évalué se trouverait trop proche de la caméra.

Lorsque la caméra opère systématiquement le même changement de direction (c'est-à-dire toujours vers la droite ou toujours vers la gauche), on obtient un cycle court entraînant forcément une boucle puisque la caméra revient à un point déjà visité. Ceci est aussi valable pour des cycles longs qui peuvent apparaître de la même manière. Des solutions proposées par la recherche en profondeur ou encore l'heuristique permettent de limiter ces effets en retardant le retour au point de départ.

La recherche en profondeur utilise les techniques d'intelligence artificielle. Chaque déplacement est évalué à l'avance ce qui nous permet de prévoir la trajectoire de la caméra et de laisser de côté des déplacements qui auraient moins d'intérêt.

L'heuristique, quant à elle, modifie la fonction d'évaluation en prenant en compte la distance entre le point testé et le point de départ ainsi que la distance parcourue. Cette méthode garantit que la trajectoire ne reste pas autour du point d'origine mais ne protège pas d'un blocage ultérieur sur un autre maximum local.

C'est au cours de l'année 2003 que Sbert et Vazquez [Vaz03] mettent au point une méthode d'exploration très proche de celle que nous venons d'étudier précédemment. Comme

pour les recherches proposées par Barral et al. [BDP00b], les mouvements consécutifs sont choisis entre trois nouvelles directions possibles, en fonction du dernier mouvement, l'objectif étant d'assurer une trajectoire lisse. La différence est qu'un point de vue est choisi en fonction de l'entropie et du nombre de faces non encore visibles. Evaluer les qualités des trois positions suivantes possibles revient à multiplier l'entropie de chaque point par le nombre de nouvelles faces qui apparaissent en regard à un ensemble de faces déjà vues. Si aucun des trois nouveaux points de vue possibles ne montre de nouvelles faces, le choix devra alors se porter sur le point le plus éloigné de la position initiale.

L'exploration en temps réel n'est, dans la plupart des cas, pas nécessaire puisque l'utilisateur a assez de temps et peut attendre un pré calcul pour les points de vue et/ou les trajectoires intéressantes.

Jaubert [Jau04, JTP06] propose une méthode d'exploration en temps différé. Cette dernière est basée sur un pré calcul d'un ensemble minimal de bons points de vue. Cet ensemble est trié par ordre d'importance et stocké avec le monde virtuel. Cet ensemble trié est utilisé à chaque fois qu'une exploration est nécessaire.

Les mondes virtuels sont plutôt faits pour une exploration interne : mondes peuplés d'avatars tels que dans les jeux vidéos, simulations telles que les visites virtuelles de bâtiments. Ce type d'exploration peut laisser apparaître trois problèmes principaux. Tout d'abord, la collision avec les obstacles, la caméra ne doit pas passer à travers les objets. Ensuite, la caméra doit explorer des parties importantes de la scène. Enfin, la caméra doit montrer autant de bonnes vues que cela lui est possible.

Les obstacles peuvent être fixes (murs, objets en général, endroits inaccessibles) ou dynamiques (des obstacles peuvent changer de position ou encore de forme).

5. Déplacement d'un avatar

Yunfang [YZW03] présente un algorithme basé sue une vision synthétique qui est utilisée pour déterminer dynamiquement le trajet d'un avatar dans un environnement virtuel

intelligent. Il utilise une vision synthétique et une scène sous forme d'arbre octal pour simuler le sens de vision de l'avatar et la mémoire de la scène.

Vazquez et Sbert [VS03] proposent une méthode automatique d'exploration d'intérieurs qui limite des degrés de liberté (dans le but de simuler un déplacement humain). Cette méthode est basée sur l'entropie du point de vue. Le processus d'exploration s'arrête quand la caméra est incapable de découvrir une nouvelle information.

Dans [MC00, VFSH02, PGJT05] des techniques basées image sont utilisées pour contrôler les mouvements de caméras dans un monde virtuel en mouvement. Le problème présenté sur le papier est l'adaptation de la caméra face aux changements du monde.

6. Récapitulatif

Les différentes approches que nous venons de voir peuvent convenir pour le domaine des jeux vidéo. En effet, quelle que soit l'approche utilisée, il peut y avoir un intérêt à utiliser chacune de ces méthodes. Cependant, les créations incrémentales de trajectoire, ainsi que les approches heuristiques semblent plus appropriées à ce type d'utilisation.

Les autres approches (déclaratives, probabilistes, etc.) sont, quand à elles, plus orientées vers la Conception Assistée par Ordinateur, le design ou les visites virtuelles. Cependant, certaines particularités de ces modèles peuvent être incorporables dans une méthode purement dédiée aux jeux vidéos, comme par exemple, l'inclusion de renseignements à des zones de la scène (approche déclarative).

C. Conclusion

Dans ce chapitre nous avons brièvement passé en revue les techniques d'exploration automatiques de scènes, telles que la sélection de bons points de vue et la création de films. Malheureusement, il y a peu d'études qui prennent ce problème du côté graphique et environnement virtuel alors que plusieurs publications prennent le problème du côté de la vision artificielle robotique.

Comme nous l'avons mentionné précédemment, le but de l'exploration dans le domaine des ordinateurs est complètement différent des objectifs des techniques utilisées en robotique. Dans le premier domaine, le but du programme qui guide une caméra virtuelle est d'autoriser un comportement humain, pour comprendre un nouveau monde en utilisant un chemin calculé automatiquement, dépendant de la nature du monde. La principale interaction est entre la caméra et l'utilisateur, l'un virtuel, l'autre humain et non entre deux agents virtuels ou un agent virtuel et son environnement. Les méthodes existantes en robotique ont, du coup, beaucoup d'inconvénients pour les parties qui nous intéressent.

Nous pouvons conclure cette partie avec l'idée que les différents travaux que nous venons de passer en revue sont très utiles. Ceux ci ont posé des bases solides qui servent encore de référence pour évaluer la qualité des nouvelles méthodes. Ils nous renseignent en effet sur des points importants en nous permettant, par exemple, de savoir comment nous pouvons faire afin de déplacer une caméra jusqu'à un point stratégique et donc intéressant de la scène.

Chapitre II
Calcul de Bons Points de Vue

II. Calcul de bons points de vue

Nous souhaitons fournir des techniques et des formules permettant de générer un ensemble fini de points de vue, estimés intéressants, pour une scène modélisée en trois dimensions. Pour ce faire, nous partons d'un ensemble initial de points de vue potentiellement intéressants auquel nous faisons subir diverses évaluations pour finalement obtenir un ensemble final contenant les meilleurs éléments.

Le calcul de points de vue est la méthode la plus modulable et la plus sujette aux avis contraires. Pour répondre à cette subjectivité de l'évaluation, nous avons voulu la plus grande adaptabilité de la technique d'estimation avec l'utilisation d'un grand nombre de points de pondération que nous appliquons sur les différents facteurs qui composent les formules. Nous ne donnerons pas de valeurs spécifiques vu que chaque partie des formules est explicitée et qu'il est alors facile de choisir les valeurs pour d'obtenir le résultat désiré.

Nous allons donc, tout d'abord, aborder le sujet de la récupération des informations à partir de la scène et présenter les deux méthodes employées. Nous verrons, en second lieu, les différents critères utilisés pour estimer la qualité d'un point de vue. Nous aborderons ensuite la nécessité de commencer avec un ensemble de départ qui donnera l'ensemble des points de vue que nous évaluerons. Enfin, nous terminerons avec la création d'un ensemble de points de vue final résultant de l'application de nos méthodes d'évaluation sur notre ensemble initial.

A. Récupération des informations

Afin de pouvoir évaluer la vision de la scène à partir d'un point de vue spécifique, il est nécessaire de mettre en place un moyen de récupérer des informations. Ces méthodes, pour être pertinentes, doivent pouvoir fournir des indications de visibilité liées aux notions de quantité et de qualité.

Nous avons choisi d'utiliser deux méthodes conjointes pour cela, celles-ci étant complémentaires l'une de l'autre et permettant de pallier ou de renforcer, respectivement, les défauts et les qualités de l'autre.

1. Le lancer de rayon

Le lancer de rayon est une des deux méthodes utilisées pour récupérer certaines informations importantes de la scène. Il s'agit d'une méthode nécessitant un important temps de calcul mais autorisant une précision importante. Le fonctionnement de la méthode est le suivant :

Pour chaque polygone, nous créons des rayons (segments) partant du point de vue et finissant sur une partie aléatoire de celui-ci. Le nombre de rayons créés par polygone est une variable définie par l'utilisateur, ce qui nous permet de moduler la précision du résultat et son exactitude.

Le temps de calcul pour le lancer de rayon est directement lié à deux paramètres que nous allons expliciter en décomposant l'algorithme simplifié non optimisé dans le cadre d'une évaluation de la visibilité des polygones composant la scène :

```
Pour chacun des P polygones
|    Pour R rayons (nombre de rayons créés)
|    |    Créer un rayon entre le point de vue et le polygone
|    |    Pour chacun des P-1 polygones (tous sauf celui de création du rayon)
|    |    |    Tester si ce polygone coupe le rayon
|    |    Fin
|    Fin
Fin
```

Nous voyons immédiatement que le nombre d'itérations de la fonction du lancer de rayon est de **« P x R x (P – 1) »** et ce pour un seul point de vue évalué. En prenant une scène de mille polygones et en créant cinq rayons par polygone, nous arrivons déjà à quasiment cinq millions d'itérations.

Il est donc logique que cette méthode soit très coûteuse en temps de calcul et ce malgré un certain nombre d'optimisations apportées (principalement le fait de ne tester que les polygones (objets) dont les boites englobantes sont traversées par le rayon et se trouvent dans la même zone que le rayon).

Un exemple d'utilisation est le calcul de la visibilité d'un polygone. Pour ce faire, nous vérifions si chacun de ces rayons est intersecté par un des autres polygones composant la

scène (polygones autres que celui servant de base au rayon). Ainsi, en générant un assez grand nombre de segments, nous pouvons savoir si le polygone est visible ou pas du point de vue (c'est-à-dire si au moins un des rayons atteint le polygone sans être bloqué par un autre).

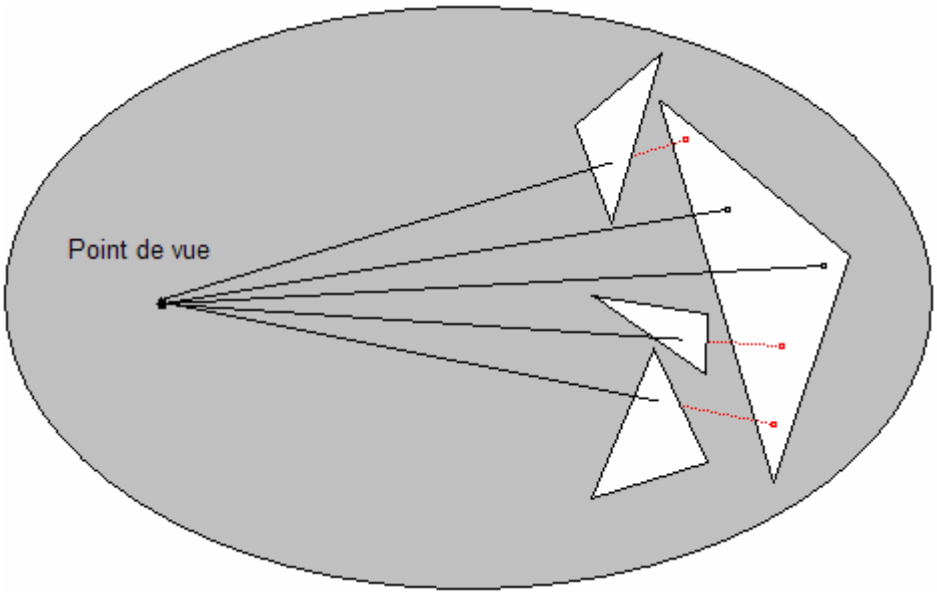


Figure 5 : Visibilité d'un polygone

La figure 5 présente une méthode de calcul de visibilité d'une scène utilisant un lancer de rayon.

2. Utilisation du Z-Buffer

Il était nécessaire dans la contrainte éventuelle du temps réel de disposer d'une méthode permettant une récupération rapide d'une partie des informations. Cette méthode, bien que moins précise que le lancer de rayon, permet une récupération et une analyse très rapide.

Nous faisons un rendu, différent en fonction du ou des critères recherchés, de la scène à partir du point de vue à tester. Une fois ce rendu terminé, l'image générée est interprétée en fonction du codage couleur de celle-ci. Bien entendu, différents effets de rendu (éclairage, anti-alliassage, etc.) peuvent être nécessaires ou contre indiqués en fonction des résultats souhaités. La figure 6 illustre cette méthode.

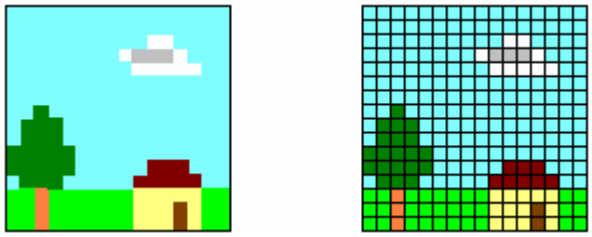


Figure 6 : Le Z-Buffer

Par exemple, pour connaître le nombre de polygones visibles à partir du point de vue, la technique la plus souvent utilisée consiste à attribuer une couleur distincte à chacun des polygones de la scène. Un tampon image est alors généré et chaque pixel du rendu est analysé. Nous comptons alors chaque occurrence des couleurs présentes qui correspondent respectivement à un polygone. Si la couleur est présente, le polygone est visible et dans le cas contraire, le polygone est masqué.

Nous pouvons en jouant de ces couleurs et de ces rendus avoir un certain nombre d'informations différentes de la scène.

Le temps de calcul est ici amoindri par rapport au lancer de rayon. Voici l'algorithme simplifié et non optimisé de cette méthode lorsqu'il est utilisé pour connaître la liste des polygones visibles :

```
Générer le rendu
Pour chaque N pixel du rendu
|     Pour chaque P polygone
|     |     Tester si la couleur du pixel correspond à celle attribuée au polygone
|     Fin
Fin
```

Le nombre d'itérations de la méthode est donc de **« N x P »** pour l'évaluation d'un point de vue, ce qui est un gain de temps très important par rapport au lancer de rayon. Par contre, le problème de cette méthode est dû à la taille du rendu généré. En effet, lorsque celui-ci est créé, il correspond à une image d'un nombre fini de pixels.

Imaginons un cas extrême avec une image générée de trois pixels sur trois. Un pixel n'ayant qu'une seule couleur, il n'est possible de représenter que neuf couleurs différentes et

donc neuf polygones. Si, à partir d'un point de vue que nous voulons évaluer, il est possible de voir un nombre plus important que ces neuf faces, l'échantillonnage n'est pas suffisant et cette méthode n'est, par conséquent, pas pertinente.

3. Choix de la méthode

Le lancer de rayons et le z-buffer, que nous venons de découvrir, seront les deux méthodes utilisées pour extraire des informations de la scène afin d'évaluer la qualité des différents points de vue. Celles-ci sont complémentaires et permettent soit une meilleure qualité, soit une plus grande vitesse de calcul. En fonction des besoins, et principalement si nous nous plaçons soit en temps réel, soit en temps différé, il sera plus pertinent d'utiliser l'une ou l'autre méthode.

Cependant, si la plupart des critères peuvent être obtenus à l'aide des deux techniques proposées, certains d'entre eux ne sont l'apanage que de l'une d'entre elles.

Ainsi, si l'utilisateur veut utiliser tous les critères d'évaluation, il est nécessaire de recourir aux méthodes conjointes et il n'est alors plus question de compter sur une exploration se déroulant en temps réel.

B. Les différents critères utilisés

Pouvoir dire clairement et facilement qu'un point de vue est bon, mauvais, suffisant, … n'est pas chose aisée. Ces affirmations subjectives sont fortement dépendantes de la scène, du type d'exploration voulu et de l'utilisateur. Cependant pour pouvoir créer une méthode qui choisisse les points de vue à la place de l'utilisateur, il est nécessaire de faire des choix et de fixer une ou plusieurs fonctions d'évaluation qui donneront une note à chacun des points de vue possibles.

Le fait de donner plusieurs variations sur la fonction d'évaluation en permettant de la pondérer partiellement va autoriser une adaptabilité de celle-ci aux différents types d'explorations et aux différents besoins de l'utilisateur.

Nous allons donc voir les différents critères apparaissant dans notre formule du bon point de vue. L'utilisation, ou non, de chacun de ces critères est à la libre appréciation de l'utilisateur et aux types d'informations qu'il souhaite mettre en avant.

1. Notion de face

La notion de face (polygone) est la base de nombreuses méthodes d'étude et de recherche de points de vue. Elle est incontournable dès qu'il est question d'exploration de scènes ou de mondes virtuels vu qu'à l'heure actuelle, toute modélisation en trois dimensions est obligatoirement basée sur des polygones.

Visibilité d'une face

Le premier critère que nous garderons pour les polygones de la scène sera leur éventuelle visibilité à partir du point de vue. Pour chacune des faces de la scène, nous avons alors le choix entre deux méthodes : soit nous prenons uniquement la visibilité « ***vf0*** » de celle-ci (booléen à 0 (non visible) ou 1 (visible)), soit nous choisissons de nuancer cette visibilité comme nous allons le voir.

Pour récupérer cette information à l'aide du z-buffer, nous attribuons à chacun des polygones une couleur distincte. Nous générons alors le tampon image, sans aucun effet de rendu, et nous regardons si la couleur apparaît sur l'image. Si c'est le cas, le polygone est visible, sinon il ne l'est pas.

Avec le lancer de rayon, nous créons un certain nombre de segments reliant le point de vue avec un point du polygone. Pour chacun de ces segments nous regardons si celui-ci est intersecté par un autre polygone. Si ce n'est pas le cas (un des segments n'est coupé par

aucun polygone), alors le polygone est visible. Dans le cas contraire (tous les segments sont intersectés par au moins un des polygones), le polygone n'est pas visible.

Pourcentage visible

Pour compléter la visibilité d'une face, il peut être pertinent de récupérer, plutôt qu'une information binaire, la quantité visible (le pourcentage) de celle-ci. Pouvons nous considérer que voir une faible partie d'un polygone suffit à en comprendre son utilité dans la scène ?

C'est pour pouvoir intégrer cette nuance que ce critère a été mis en place. La visibilité devient donc un réel compris entre 0 (non visible) et 1 (totalement visible). Nous la noterons « ***pf()*** ».

Afin de pouvoir calculer le pourcentage visible, il faut générer le même rendu que précédemment et compter le nombre de pixels visibles de la face. Ensuite un autre rendu est généré en affichant que la face concernée puis nous comptons le total de pixels pour ce polygone. Nous divisons alors le nombre visible par le nombre total pour obtenir un réel correspondant au pourcentage visible.

En utilisant le lancer de rayon, la technique se rapproche beaucoup de celle utilisée pour la visibilité d'une face. Nous créons toujours un certain nombre de segments et nous vérifions pour chacun d'eux s'il est intersecté par un polygone. Nous regardons alors le rapport de rayons intersectés et non intersectés. Le pourcentage visible est égal, au nombre de segments qui ne sont pas interrompus, divisé par le nombre de segments créés.

Inclinaison d'une face

Pour clore notre batterie de critères sur les polygones de la scène, nous ajoutons une notion d'inclinaison de celle-ci par rapport au point de vue. Nous considérons qu'il est plus intéressant de voir une face dont la normale est parallèle à la ligne de vision plutôt qu'une autre. Nous appellerons α l'angle entre le rayon issu du point de vue et allant au polygone et la normale de ce dernier (figure 7).

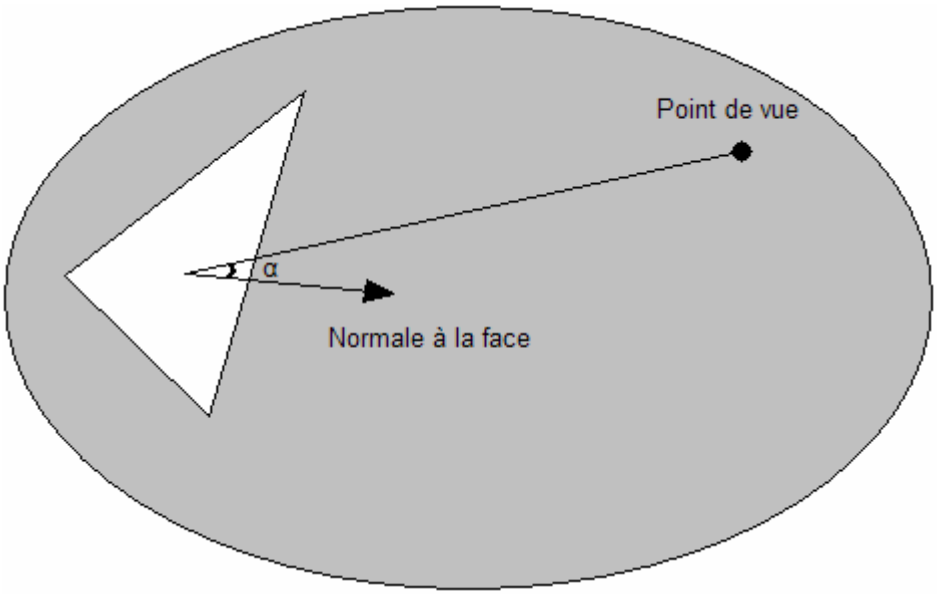


Figure 7 : Inclinaison d'une face et angle de vue

Le calcul de cet angle ne nécessite ni l'utilisation d'un z-buffer, ni celui d'un lancer de rayon. Il suffit pour se faire de définir deux vecteurs : un premier vecteur qui est le vecteur directeur de la droite issue du point de vue et passant par le centre de gravité du polygone et un second étant le vecteur normal au polygone. L'angle formé par ces deux vecteurs est l'angle d'inclinaison de la face.

Faces déjà vues

Nous avons besoin lors de notre recherche de noter les faces ayant été vues par des points de vue déjà sélectionnés. Ces faces verront alors leur note diminuée (divisée par 2 ou 3) ou annulée (mise à 0). Cette revalorisation des polygones est extrêmement importante, notamment dans le cadre des animations et des cinématiques. Dans tous les cas, cette action garantit que la fonction d'évaluation passe outre la majorité des redondances d'informations.

Estimation de la qualité d'une face

Voici la formule obtenue à l'aide de tous les critères précédemment énumérés :

$$EvalPoly(P) = \frac{a_p.vf(P) + b_p.pf(P) + c_p.\cos(\alpha)}{a_p + b_p + c_p}$$

Les variables « $\mathbf{a_p}$ », « $\mathbf{b_p}$ » et « $\mathbf{c_p}$ » sont des réels servant à pondérer notre formule pour l'adapter aux circonstances. Ainsi dans le cas général nous prendrons ces trois variables égales à 1.

Nous pouvons, de plus, constater que cette formule donne un résultat compris entre 0 et 1 permettant d'avoir un ordre d'idée très rapidement lors des différents essais effectués mais aussi une valeur normée.

2. Notion d'objet

La recherche de positions de caméra, de bons points de vue et l'exploration d'une scène ne peuvent cependant pas uniquement se baser sur les polygones (techniques de bas niveau) mais doivent tenter de hausser le débat dans des strates plus évoluées de la compréhension de la scène. Pour se faire il est nécessaire de sortir des sommets et des polygones pour arriver à cerner d'autres éléments qui peuvent renseigner sur les spécificités de certaines entités.

Nous avons donc rajouté un élément de hiérarchisation entre la scène globale et le polygone : l'objet. Nous obtenons donc la hiérarchie présentée sur la figure 8.

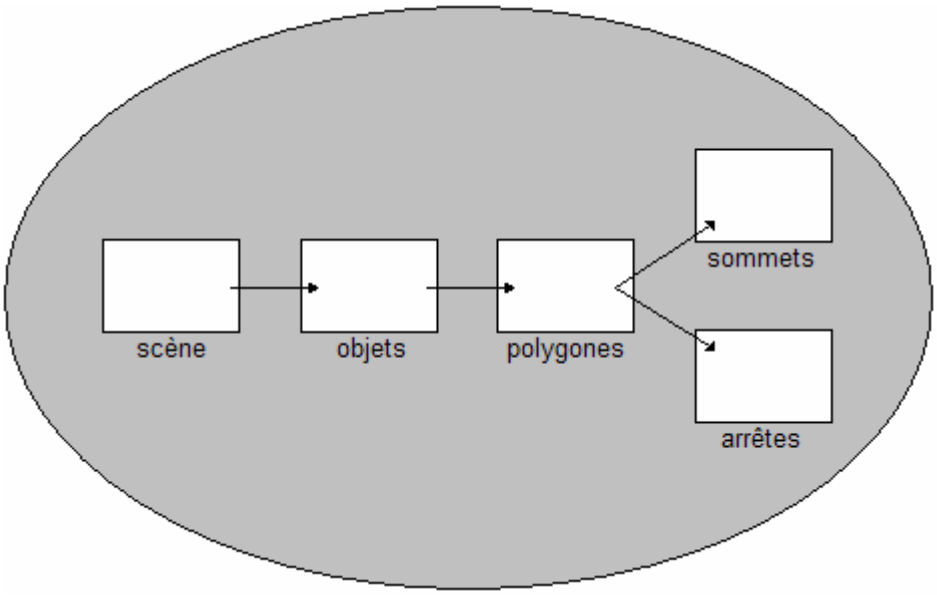


Figure 8 : Hiérarchisation basique d'une scène

La scène est composée d'objets, eux-même composés de polygones et de sommets. Ce schéma peut facilement être découpé récursivement dans le but d'ajouter autant de niveaux de hiérarchie que désiré. Chaque objet est alors découpé en sous-objets qui eux-même peuvent être découpés en sous-sous-objets et ainsi de suite jusqu'au niveau de détail souhaité.

Pour ce qui nous concerne, nous considérons que chaque entité à partir du polygone est distincte, c'est-à-dire que chaque polygone ne peut appartenir qu'à un seul sous objet qui n'appartient qu'à un seul et unique objet. Nous ne détaillerons les méthodes qu'avec un seul niveau (pas de sous-sous-objets) mais celles-ci sont facilement adaptables à plusieurs niveaux (figure 9).

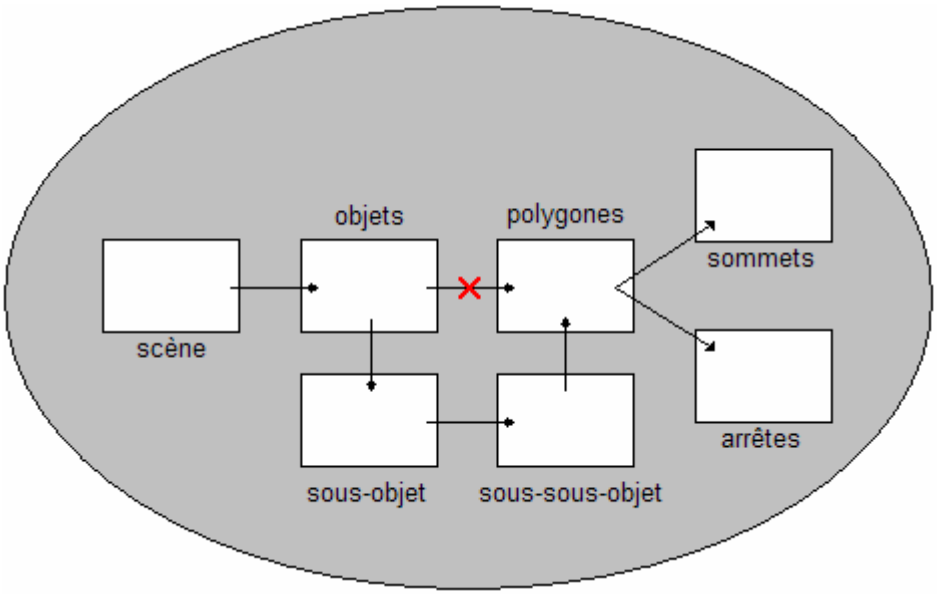


Figure 9 : Hiérarchisation évolutive d'une scène – premier et second niveau

Nous allons donc voir, à l'instar du polygone, les différents critères possibles sur les objets.

Visibilité d'un objet

Définir la visibilité d'un objet n'est pas aussi facile que de déterminer celle d'une facette. En effet, du point de vue commun, un objet peut être considéré visible différemment en fonction du contexte. Le cas le plus simple étant donc de considérer un objet visible à partir du moment où l'on voit un des polygones qui le composent. Cette méthode est très discutable car pour un objet de plusieurs milliers de polygones, une face ne représente pas beaucoup de choses. Nous le noterons « ***vo()*** ».

Pourcentage de visibilité

Comme pour les faces, nous définissons pour l'objet un pourcentage de visibilité « **po()** » qui sera le nombre de polygones visibles divisé par le nombre de polygones qui le composent. Cette information est alors plus intéressante que la précédente mais reste incomplète en cas de grandes différences au niveau de la modélisation des différentes parties

de l'objet. Cependant, sur une scène dont les degrés de définition des différents éléments sont uniformes, ce critère est très pertinent.

Pourcentage de surface visible

Ce dernier critère comble les lacunes des deux précédents. Nous divisons la surface des polygones visibles de l'objet par la surface des polygones qui le composent. Ainsi si une zone est plus détaillée qu'une autre, celle-ci ne sera pas mieux notée qu'une autre de même surface. Nous le noterons « **so() »**.

Objets déjà vus

Dans nos explorations nous considèrerons les objets déjà vus (par un précédent point de vue choisi par exemple) comme d'un moindre intérêt par rapport à un objet qui ne l'aurait pas été. Ainsi un tel objet verra sa note grandement diminuée ou même mise à 0 en fonction des cas. L'intérêt de moduler ainsi les valeurs des différents objets est d'encourager la découverte de nouveaux objets plutôt que de risquer de stagner sur des objets dont nous avons déjà pris connaissance. Bien entendu, ceci n'est valable qu'une fois les premiers points de vue choisis car cela nécessite des références.

Estimation de la qualité d'un objet

Voici la formule obtenue à l'aide de tous les critères précédemment énumérés :

$$EvalObjet(O) = \frac{a_o.vo(O) + b_o.po(O) + c_o.so(O)}{a_o + b_o + c_o}$$

Les variables « $\mathbf{a_o}$ », « $\mathbf{b_o}$ » et « $\mathbf{c_o}$ » sont des réels servant à pondérer notre formule pour l'adapter aux circonstances. Ainsi dans le cas général chacune de ces variables sera égale à 1.

Comme dans la formule d'évaluation des polygones, la valeur résultante de cette estimation sera comprise entre 0 et 1.

3. Notion de variation d'objet

La notion de variation d'objet a été introduite pour les scènes animées. Que se passe t'il dans le cadre d'une scène animée ? La position de certains polygones et donc de certains objets peut changer au court du temps. Ainsi des polygones peuvent disparaître, apparaître ou se modifier ce qui influe alors sur les objets dont ils sont les composants. La scène variant en fonction du temps, nous ne pouvons plus nous baser sur une exploration basique.

Nous rajoutons donc un axe temporel t en plus des x, y et z habituels. La scène à l'origine est en « $\mathbf{t_o}$ ». Nous considérons ensuite des états de la scène à intervalles de temps réguliers. A tous ces états, nous pouvons séparer les objets de la scène en deux parties : les objets ayant bougés et ceux qui sont restés fixes. Ainsi en « $\mathbf{t_1}$ », nous avons une partie des objets sur lesquels s'appliquent les règles normales (les objets fixes entre « $\mathbf{t_o}$ » et « $\mathbf{t_1}$ ») et le reste soumis à de nouvelles règles. Ces derniers ne sont alors plus considérés comme des objets déjà vus (du fait de leur mouvement, ils restent un centre d'intérêt privilégié). Bien entendu, cela ne reste valable que le temps de leur déplacement.

Ainsi, au fur et à mesure du temps, les seuls objets qui resteront intéressants seront ceux en mouvement. Ce qui permettra assez rapidement de focaliser la caméra sur le ou les lieux de l'action et donc sur les zones intéressantes.

4. Notion d'éclairage

L'éclairage est une chose peu prise en compte dans l'évaluation d'un point de vue. Nous parlons ici du cas où les sources de lumières sont positionnées dans la scène et font partie intégrante de celle-ci, comme par exemple, des torches sur un mur.

Nous avons choisi d'utiliser un Z-buffer pour le coupler avec les autres critères. Un rendu est effectué en niveau de gris avec l'éclairage. La couleur de chaque pixel « **lum(p) »** est alors comprise entre 0 et 1, correspondant au spectre allant respectivement du noir (0) au blanc (1). De cette manière nous gardons la cohérence instaurée (tous les critères sont notés entre 0 et 1, avec le 1 qui est considéré comme la valeur haute). Bien entendu si la scène que nous considérons ne comporte pas d'éclairage, ce critère n'est pas pris en compte.

La note pour une capture (et donc pour un point de vue) est calculée comme suit :

$$evallumiere(PdV) = \frac{\sum_{p=p_0}^{p_n} lum(p)}{nombre\,de\,pixels}$$

Si les lumières sont mobiles, il faut se placer dans le même cas de figure que précédemment (voir la notion de variation d'objet).

5. Evaluation du point de vue

Nous venons de voir un certain nombre de critères qui peuvent être utilisées pour définir si un point de vue est plus intéressant qu'un autre.

En réunissant tous les critères et systèmes de notation détaillés, nous pouvons alors donner une note d'évaluation d'un point de vue. Cette note prend donc en compte un large choix de critères que nous pouvons à volonté pondérer pour obtenir des variantes en fonction des besoins de l'utilisateur.

$$note(PdV) = \frac{a_e.evallumiere(PdV) + b_e.\frac{\sum_{o=o_0}^{o_n} evalobjet(o)}{nb\ objets} + c_e.\frac{\sum_{p=p_0}^{p_n} evalpoly(p)}{nb\ \ faces}}{a_e + b_e + c_e}$$

Les variables « $\mathbf{a_e}$ », « $\mathbf{b_e}$ » et « $\mathbf{c_e}$ » servent à pondérer la formule. « **nb objets** » est le nombre d'objets que comporte la scène et « **nb faces** » le nombre de polygones. Les fonctions sont celles vues précédemment.

C. Choix d'un ensemble de départ

Le choix d'un ensemble de départ est primordial pour la qualité des points de vue et le temps de traitement. En effet, il est impossible de prendre en compte tous les points de vue possibles et imaginables car il faudrait évaluer une infinité de cas, ce qui prendrait donc un temps infini. Il est donc nécessaire de faire un tri sur l'ensemble des points de vue possibles.

Une grosse partie de ce travail peut être faite en réduisant l'espace de travail à un espace fini. En effet, une scène est toujours un espace limité fini : les objets qui la composent étant dénombrables, nous pouvons déterminer une boite englobante contenant l'ensemble de celle-ci. De ce fait, placer un point de vue à une grande distance de cette boite englobante ne sert à rien, bien au contraire. Il existe donc une distance maximale pour voir correctement la zone ce qui limite considérablement le volume à traiter (figure 10).

Figure 10 : Eloignement du point de vue

Inversement, une trop grande proximité peut gêner la vision. En se plaçant contre un objet, il peut devenir impossible pour une personne de savoir ce que représente ce qu'il est en train de regarder. Il est donc important de ne pas trop se rapprocher d'un objet pour ne pas perdre le recul nécessaire à l'interprétation de celui-ci (figure 11).

Figure 11 : Proximité du point de vue

Enfin, dernier point logique, il faut regarder vers la scène pour que les points de vue soient pertinents car, dans le cas contraire, nous ne voyons rien.

Nous venons de réduire énormément les zones pour placer des points de vue potentiellement intéressants mais il reste toujours une infinité de possibilités. Il nous faut donc transformer un ensemble infini en nombre fini de possibilités.

Nous remarquons que deux points de vue proches sont extrêmement redondants. En effet en supposant que ceux-ci soient quasiment confondus, l'aperçu de la scène sera, à peu de choses près, identique. Il est donc inutile de tester des points de vue proches d'un point de vue déjà testé ou qui le sera. Nous pouvons donc répartir les points de vue à évaluer à une certaine distance les uns des autres et ainsi discrétiser l'espace de travail. Nous obtenons donc un nombre fini de points de vue qui constituera l'ensemble de départ. Cet ensemble contient tous les points de vue susceptibles d'être intéressants.

Bien entendu, en fonction des différents types de scène ou d'exploration, nous pouvons optimiser différemment les ensembles et les modifier pour qu'ils collent au mieux à ce que nous voulons en faire comme nous allons bientôt le voir.

1. Exploration externe d'un objet

L'exploration externe d'un objet est le cas le plus simple d'exploration. Seule l'enveloppe extérieure de l'objet est prise en compte.

Nous avons mis en place deux systèmes de répartition pour un ensemble initial. Le premier utilise une unique sphère englobante comme nous le verrons dans un premier temps alors que le second discrétise l'espace entre deux sphères, ce que nous expliciterons ensuite.

Nous plaçons donc les points de vue sur une sphère englobante de la scène dont le rayon est calculé avec la formule suivante :

$$p = \frac{3}{2} \times \sqrt{2 \times \max(dX^2, dY^2, dZ^2)} \text{ avec } dU^2 = (U\max - U\min)^2$$

La sphère est, bien entendu, discrétisée à des maillages plus ou moins importants en fonction de ce que nous voulons obtenir et de la précision recherchée dans le résultat (figure 12).

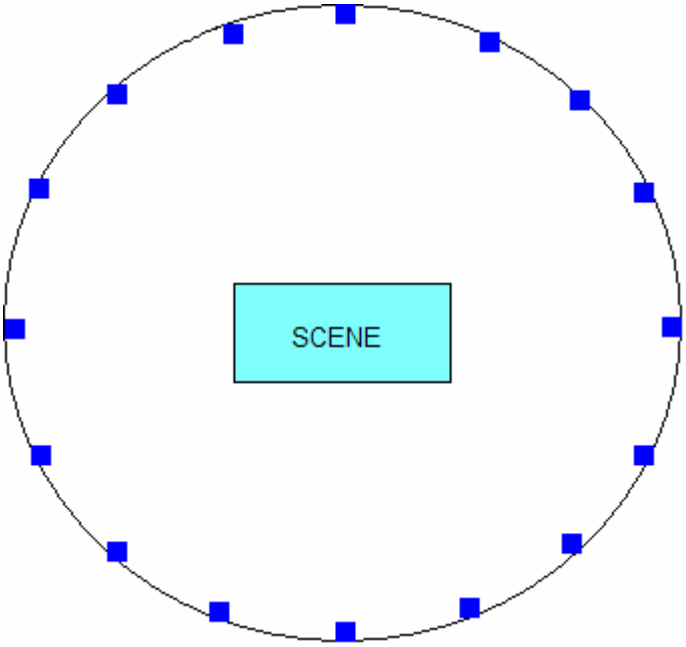


Figure 12 : Points de vue répartis sur la surface de la sphère englobante

Les points de vue possibles sont donc placés sur cette sphère et regardent le centre géométrique de l'objet.

Nous utilisons les coordonnées sphériques (figure 13) et nous faisons varier Θ et ϕ dans une double boucle pour placer les points de vue.

Pour Θ allant de 0 à 2Π
| Pour ϕ allant de 0 à Π
| | Placer le point de vue en (ρ , ϕ , Θ) par rapport au centre de l'objet
| Fin
Fin

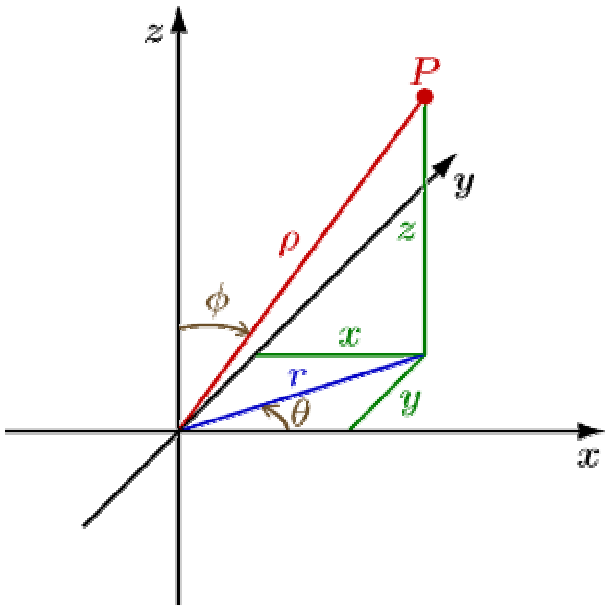


Figure 13 : Coordonnées sphériques

Le seul problème de cette méthode est que les points de vue obtenus ne sont pas répartis de manière homogène sur la sphère.

Pour pallier ce problème et avoir une méthode qui se rapproche de celle de notre création de trajectoire (voir le chapitre sur la création de trajectoire), nous ajoutons une seconde sphère englobante ayant pour rayon soixante quinze pourcent de celui de la première sphère.

$$p = \frac{3}{2} \times \sqrt{2 \times \max(dX^2, dY^2, dZ^2)} \text{ et } p_2 = \frac{9}{8} \times \sqrt{2 \times \max(dX^2, dY^2, dZ^2)}$$

avec $dU^2 = (U\max - U\min)^2$

Nous plaçons alors les points de vue selon un certain pas en x, y et z dans l'espace entre les deux sphères englobantes (figure 14).

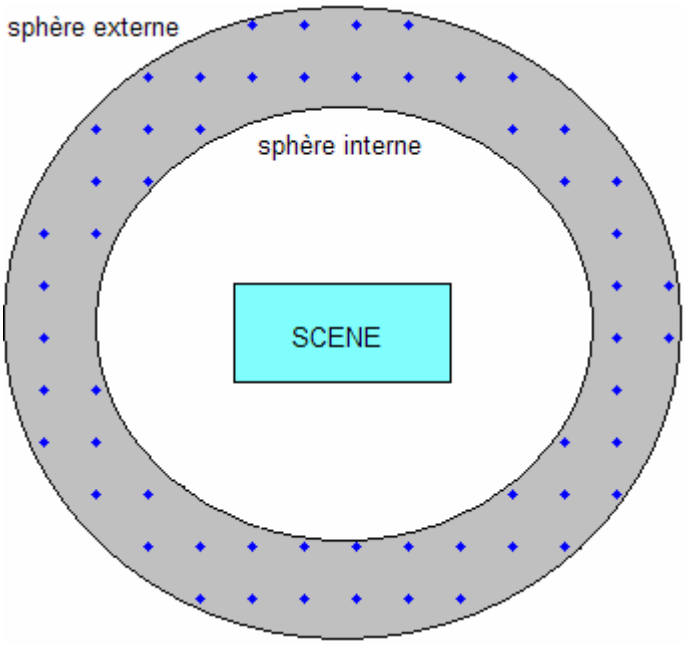


Figure 14 : Utilisation de deux sphères englobantes

```
Pour x allant de xmin à xmax
|    Pour y allant de ymin à ymax
|    |    Pour z allant de zmin à zmax
|    |    |    Si le point (x, y, z) est entre les deux sphères
|    |    |    |    Placer un point de vue en (x, y, z)
|    |    |    Fin
|    |    Fin
|    Fin
Fin
```

Cette seconde méthode permet d'avoir une répartition plus uniforme des points de vue de départ et a le mérite de nous éviter les passages de coordonnées sphériques à cartésiennes et inversement.

2. Exploration externe d'une scène de plusieurs objets

Dans le cas d'une scène contenant plusieurs objets, il est fort probable que ceux-ci soient de tailles très différentes. Dans ce cas, un ensemble basé uniquement sur la sphère englobante de la scène complète ne peut pas être pertinent.

Nous avons donc choisi de prendre, pour un ensemble de n objets, un ensemble de n+1 groupes de points de vue. Le premier d’entre eux est le même que dans le cas précédent, c'est-à-dire un ensemble de points appartenant à la sphère englobante du tout (nous allons considérer que les deux méthodes –avec une ou deux sphères englobantes- vues précédemment fonctionnent sur le même principe pour ne pas détailler la suite en double). Les n autres groupes sont des points de vue appartenant aux sphères englobantes de chacun des objets. Ainsi nous avons des points de vue pour voir l'ensemble et d'autres permettant de se rapprocher des objets en particulier. La figure 15 illustre cette proposition.

Il est à noter que cette façon de faire rejoint la précédente lorsque qu'il n'y a qu'un seul objet car les deux ensembles générés sont alors identiques.

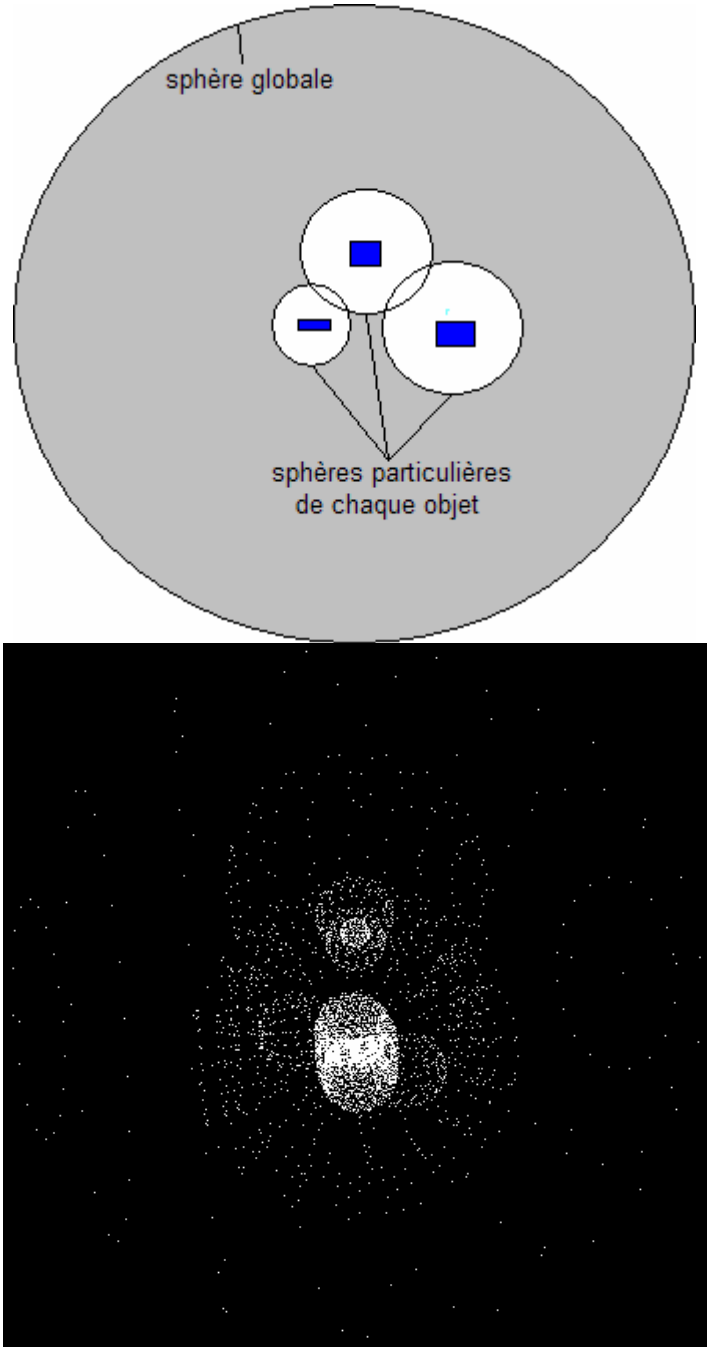


Figure 15 : Sphères englobantes et objets multiples

L'avantage de ce procédé est que nous garantissons d'avoir, à la fois, des points de vue globaux (sur la sphère englobante de toute la scène), et des points de vue plus particuliers (sur chacun des objets).

De plus, en se rapprochant ainsi de chacun des objets, les détails qui peuvent nous échapper dans une exploration classique, apparaissent. Ainsi des endroits qui n'étaient pas

visibles par la sphère externe peuvent maintenant être évalués et estimés, notamment en ce qui concerne les petits détails ou les objets contenant un grand nombre de protubérances.

La figure 16 présente des points de vue obtenus lorsque l'on quitte la sphère englobante de la scène et que l'on se rapproche des objets.

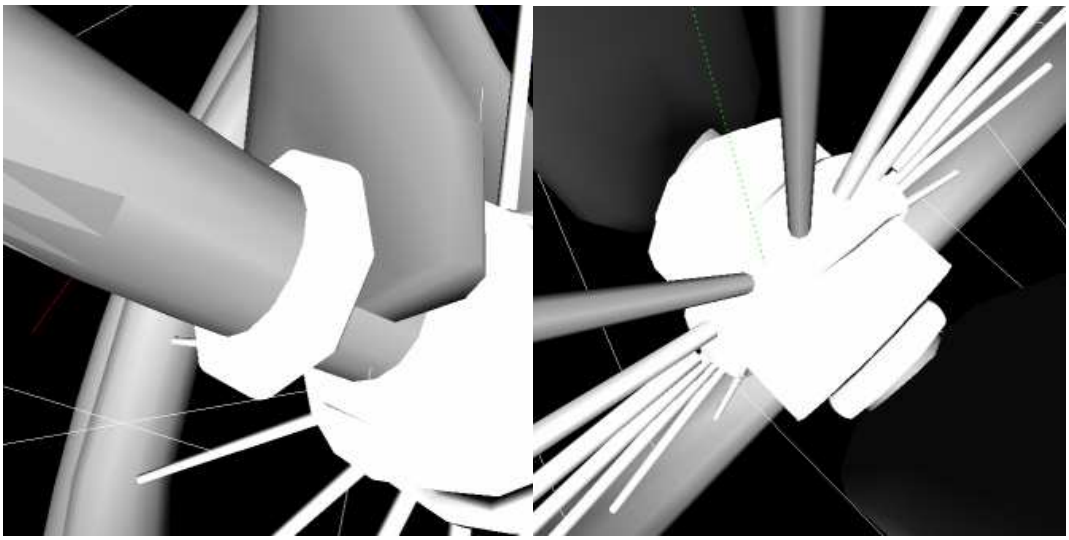


Figure 16 : Points de vue à proximité des objets

3. Visite d'une ville

Pour une ville, même si l'exploration précédente peut donner des résultats convenables, il peut être bienvenu de se rapprocher d'une exploration plus humaine. Pour coller alors au mode de déplacement, l'ensemble est choisi dans un plan à hauteur fixée par rapport au sol. Par contre, pour chaque position, plusieurs points existent avec des directions de vision différentes ce qui correspond au fait que l'être humain oriente sa tête et pose son regard sur des objets qui ne sont pas forcément juste devant lui. Nous pouvons donc représenter l'ensemble des endroits vers lesquels il regarde comme une demi-sphère centrée sur sa position ou une sphère si l'on considère que regarder le sol peut apporter quelque chose d'intéressant. Cependant, les points de vue orientés vers le sol ne seront généralement pas aussi intéressants que les autres et il n'est donc pas nécessaire de les considérer dans la plupart des cas. Les figures 17 et 18 nous présentent les positionnements de ces points de vue.

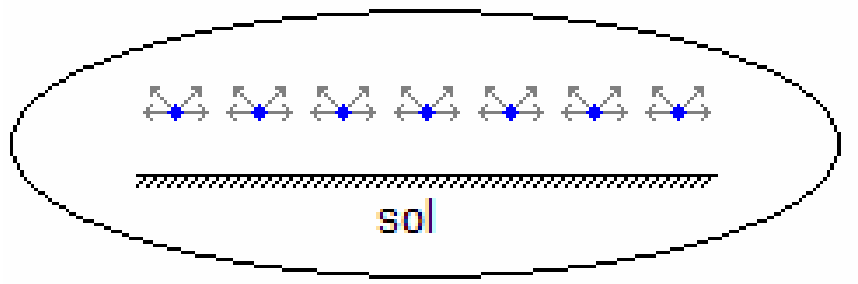


Figure 17 : Position des points de vue pour l'exploration d'une ville

Nous répartissons donc les positions à intervalle régulier sur un plan parallèle au sol à une hauteur indiquée. Pour chacune de ces positions, nous créons plusieurs points de vue orientés vers plusieurs directions.

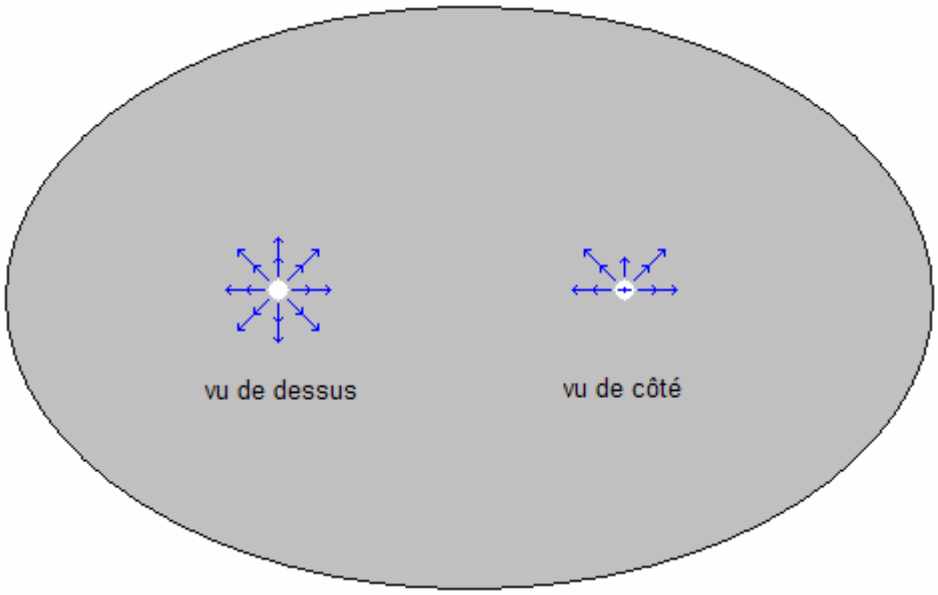


Figure 18 : Aperçu d'une position et des points de vue associés

Nous représentons ainsi une exploration humaine avec les possibilités d’inclinaisons normales du regard.

4. Cas généraux et spécifiques

Dans tous les cas, il est donc possible de choisir et/ou de créer un ensemble précis de points de vue pour une scène spécifique. L'utilisateur devra alors choisir la répartition, le type d'ensemble ou définir les points de vue un à un.

Ce choix d'un ensemble de départ est très important car il représente la première optimisation possible sur l'ensemble des calculs. Si l'ensemble est mal choisi et ne contient aucun point de vue intéressant, alors aucun des points de vue sélectionnés ne le sera. Si le nombre d'entités est trop important, le temps de calcul global va s'en ressentir.

Ainsi, si l'ensemble des points de vue initiaux est assez réduit et pertinent, il y a fort à parier que le choix des points de vue sera finalement très pertinent et intéressant.

D. Extraction de l'ensemble final

Une fois que nous avons défini correctement un ensemble de départ qui contient les points de vue potentiellement intéressants, il convient de sélectionner un nombre de points de vue qui seront finalement choisis et deviendront l'ensemble final de points de vue.

Pour ce faire, nous allons utiliser la fonction d'évaluation, que nous avons vu dans la première partie de ce chapitre, sur chacun des points de vue de l'ensemble de départ.

1. Contraintes sur l'ensemble final

L'ensemble final se doit de satisfaire un certain nombre de contraintes pour que nous puissions le considérer de qualité. Tout d'abord la répartition des points de vue sur la scène doit être intelligente. En effet si un point de vue est désigné comme très bon par la fonction d'évaluation, il y a de très fortes chances pour que ses voisins proches le soient aussi. Il n'est cependant pas pertinent qu'ils apparaissent tous dans l'ensemble final.

Autre point, l'ensemble final ne doit pas être trop important ou en tout cas bien moindre que l'ensemble initial. En effet, si nous prenons un ensemble final égal à l'ensemble initial, il n'est pas nécessaire d'évaluer les points de vue.

Il faut donc que cet ensemble final évite au maximum les redondances d'informations (objets identiques vus, positions proches, ...) et minimise le nombre de points de vue.

2. De l'intérêt de stocker des informations

Dans le but d'éviter la redondance entre les points de vue, stocker des informations est capital et permet de garantir un bon ensemble final. Cela permet de connaître et de repérer les objets ou les parties d'objets qui ont été vus lors des sélections précédentes. Ainsi il est beaucoup plus facile d'éviter les informations déjà acquises sur des objets et de préférer soit d'autres points de vue de l'objet soit des points de vue sur de nouveaux objets.

3. Présentation de la méthode

Nous allons expliciter ici la méthode de sélection de l'ensemble final de points de vue.

Voici l'algorithme d'extraction de l'ensemble final :

```
Tant que critère d'arrêt non vrai
|    Pour chaque point de vue de l'ensemble initial
|    |    Evaluer le point de vue
|    Fin
|    Pour le meilleur point de vue
|    |    Ajouter aux marqués les polygones et objets vus
|    |    Ajouter ce point de vue à l'ensemble final
|    |    Retirer ce point de vue de l'ensemble initial
|    Fin
Fin
```

Le critère d'arrêt peut être de plusieurs formes.

Tout d’abord, cela peut être une contrainte sur la cardinalité de l’ensemble final de points de vue. Lorsque le nombre de points de vue demandé est atteint, la sélection s’achève.

Ensuite, il est possible d’arrêter lorsque l’évaluation des points de vue tombe sous un seuil ou devient nulle. Le seuil peut, par exemple, être donné par un pourcentage de la plus haute évaluation possible (le meilleur point de vue), ou lorsque qu’aucun nouvel élément n’est visible lors d’une boucle d’évaluation.

La figure 19 illustre le résultat de cette sélection de points de vue appliquée à une scène composée de plusieurs objets. Chacun des objets est indiqué par un trait blanc. L’extrémité de ce trait est un des points de vue choisis.

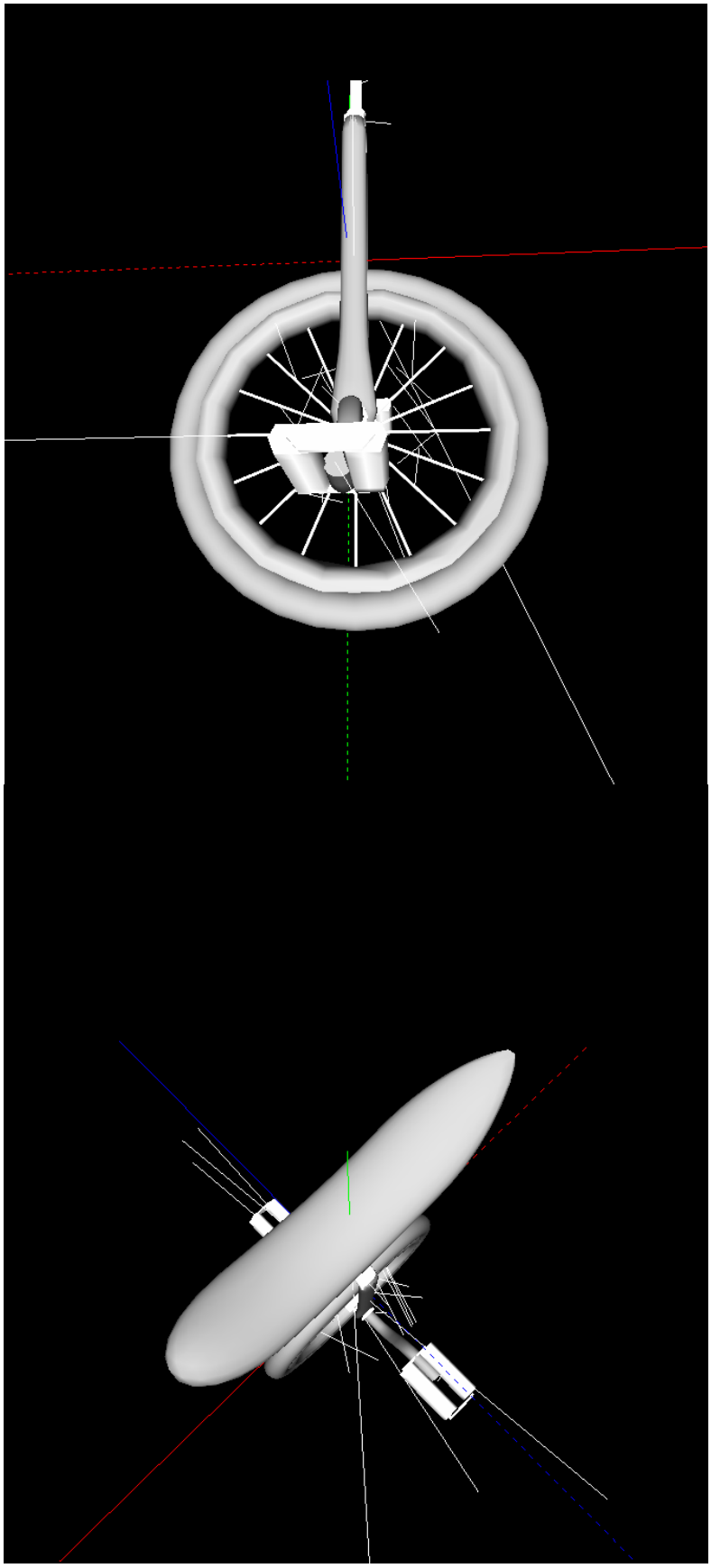


Figure 19 : Points de vue choisis et objets multiples

E. Application aux jeux vidéo

Nous allons voir comment appliquer ces méthodes au monde des jeux vidéo. Il convient tout d'abord de bien comprendre que les formules livrées sont génériques et qu'il convient de les pondérer correctement pour obtenir le résultat souhaité. De plus, les calculs nécessitent une importante connaissance de la scène et il est obligatoire de passer du temps pour définir tous les éléments utiles.

1. Choix de l'ensemble de départ

Comme nous l'avons vu, le choix de l'ensemble de départ est la première optimisation à effectuer dans le but de réduire les temps de calcul. Dans la plupart des jeux, une cardinalité limitée à une cinquantaine de points sera largement suffisante. Une caméra placée au dessus du joueur avec un angle de vision de 45° vers le sol fera des merveilles. Il suffit donc de tester un plan comme pour l'exploration humaine, mais à une distance plus importante du sol, avec des points de vue orientés vers le sol et/ou le personnage et ses environs.

2. Comment calculer en temps réel ?

Le calcul en temps réel peut se poser dans trois cas de figures : soit les données ne peuvent pas être conservées, soit le pré-calcul n'est pas possible, soit le monde est trop vaste pour être discrétisé en un nombre réduit de points de vue.

Que pouvons nous faire dans ce cas pour simplifier les calculs ?

Il convient alors, dans une optique de temps réel d'éliminer les critères prenant du temps et des ressources pour ne garder que ceux qui peuvent être générés rapidement (principalement ceux utilisant le z-buffer). De plus, au lieu de travailler sur les objets et polygones, il convient de ne plus travailler que sur les boites englobantes de ces objets. Chaque objet est donc remplacé par un parallélépipède adapté en taille et composé de douze polygones (six faces de deux polygones chacune). Ensuite, pour tout type de traitement, il

suffit alors de travailler avec la scène composée des boîtes englobantes sans prendre en compte la scène principale.

Cette transformation doit bien sûr se faire en pré-traitement mais il n'est ensuite plus nécessaire de le refaire.

F. Conclusion

L'estimation d'un point de vue est une chose subjective et sujette à controverse. C'est avec cette idée que nous avons voulu fournir une méthode d'estimation entièrement paramétrable. Dans cette optique, les différents points de pondération permettent d'utiliser notre méthode dans les situations les plus diverses et permettent de prendre, ou de ne pas prendre, en compte chacun des critères d'estimation proposés.

Le second point de contrôle -le choix de l'ensemble de départ- doit être correctement défini par l'utilisateur. En effet, les cas génériques ne peuvent pas correspondre, dans la majorité des cas, à un ensemble de départ optimisé même s'ils offrent une première approche intéressante et peuvent orienter les paramétrages ultérieurs. Un mauvais choix de l'ensemble de départ peut provoquer de très mauvais résultats sur l'ensemble final obtenu car celui-ci n'est qu'un sous ensemble du premier.

Quoi qu'il en soit, les paramétrages adaptés sont spécifiques à chaque scène et doivent faire l'objet d'ajustements de la part de l'utilisateur. Il est donc nécessaire de bien connaître, et la scène, et le résultat attendu, pour obtenir les meilleurs résultats possibles.

Chapitre III
Décomposition et Regroupement

III. Décomposition et regroupement

Nous venons de voir que nos méthodes peuvent utiliser une certaine hiérarchisation de la scène afin d'augmenter la qualité de l'évaluation. Or, cette organisation des données n'existe pas forcément dans toutes les scènes, et, principalement dans celles qui n'ont pas été créées par l'utilisateur.

Comment, dès lors, utiliser ces techniques sur des scènes qui ne possèdent pas ce découpage ? Cette partie va donc présenter des méthodes qui, à partir de scènes ne possédant pas d'informations hiérarchiques, vont introduire le niveau d'objets dans la décomposition d'un monde virtuel.

Nous avons choisi d'appréhender le problème sous deux approches différentes. La première idée est de partir de l'ensemble de la scène pour la découper en sous parties (objets). La seconde approche fonctionne à l'inverse et part des sommets et des polygones pour les regrouper en différents objets. Nous verrons enfin comment regrouper ces deux approches afin d'en faire une méthode générale et pertinente.

A. Décomposition de la scène en objets

1. De l'existence des informations

Nous partons du principe que la scène contient des informations « cachées » à l'utilisateur qui sont pourtant susceptibles d'être intéressantes. Bien que dans de rares cas, ces données ne soient pas présentes, la majorité des scènes comportent deux types de renseignements qui vont nous permettrent de distinguer les différentes parties qui la composent et de les noter comme objets différents.

2. Utilisation de la modélisation de la scène

Il est rare qu'un monde en trois dimensions soit modélisé d'une seule traite par son créateur. En effet, dans la plupart des cas, chaque objet est fabriqué indépendamment du reste

pour être ensuite placé ou dupliqué dans la scène. Cela peut s'expliquer par la complexité et le coût de la modélisation et la nécessité, dès lors, de rentabiliser chaque objet en le réutilisant dans différentes scènes.

Une scène se retrouve donc être un ensemble d'objets rapportés, obtenus de sources parfois différentes et n'ayant pas forcément de rapport direct entre eux.

Ces informations de modélisation peuvent alors apparaître sous deux formes :

- Chacun des fichiers (pour les objets) est indépendant des autres et un fichier pour la scène indique le placement de chacun en leur appliquant homothéties et rotations.
- Un seul fichier existe pour les objets et le monde. L'ensemble des informations est contenu dans celui-ci.

Dans ce premier cas, il n'y a aucun problème du fait que les objets sont directement définis par l'utilisateur ou le créateur de la scène. Il suffit alors de les utiliser tels qu'ils se présentent.

Dans le second cas, nous pouvons rencontrer trois types de fichiers :

- Les informations sont clairement notées et les différents objets apparaissent clairement, nous revenons dans le premier cas de figure.
- Les informations ne sont pas clairement notées et il est nécessaire d'analyser le fichier pour pouvoir obtenir les informations que l'on désire sur la modélisation de la scène. C'est le cas que nous allons détailler.
- Les informations ne sont pas présentes et il est impossible de tirer des conclusions à partir du fichier source. Nous ne pouvons alors pas utiliser le fichier en lui-même pour obtenir des informations.

Le diagramme ci-dessous (figure 20) résume les différents cas que nous venons de voir. Nous allons donc nous attacher à détailler le cas où un seul fichier est présent et où les informations sont présentes mais pas forcément facilement identifiables.

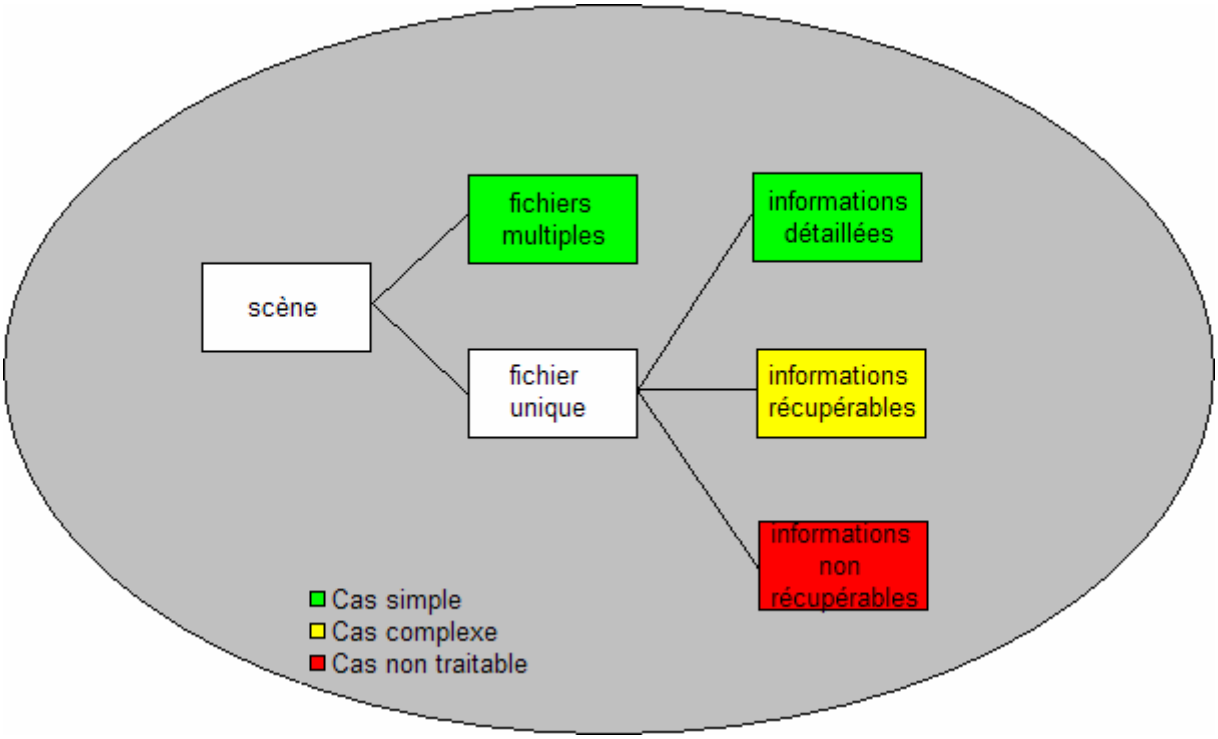


Figure 20 : Différents cas de figure en fonction du (des) fichier(s)

La problématique qui se pose maintenant à nous est la récupération d'informations non évidentes sur la structure de la scène. Nous allons donc voir un ensemble de possibilités, non exhaustif, permettant de discerner et de séparer les différents objets au sein d'une même scène.

(1) Ordre des champs dans le fichier

Afin d'expliciter cette possibilité, nous allons nous placer dans un cas simple avec uniquement des sommets et des polygones. La scène contient uniquement trois triangles.

Voici deux possibilités de fichier :

Cas 1	Cas 2
Sommet 1 (X1, Y1, Z1)	Sommet 1 (X1, Y1, Z1)
Sommet 2 (X2, Y2, Z2)	Sommet 2 (X2, Y2, Z2)
Sommet 3 (X3, Y3, Z3)	Sommet 3 (X3, Y3, Z3)
Sommet 4 (X4, Y4, Z4)	Polygone (1, 2, 3)
Sommet 5 (X5, Y5, Z5)	Sommet 4 (X4, Y4, Z4)
Sommet 6 (X6, Y6, Z6)	Sommet 5 (X5, Y5, Z5)
Sommet 7 (X7, Y7, Z7)	Sommet 6 (X6, Y6, Z6)
Sommet 8 (X8, Y8, Z8)	Sommet 7 (X7, Y7, Z7)
Sommet 9 (X9, Y9, Z9)	Sommet 8 (X8, Y8, Z8)
Polygone (1, 2, 3)	Sommet 9 (X9, Y9, Z9)
Polygone (4, 5, 6)	Polygone (4, 5, 6)
Polygone (7, 8, 9)	Polygone (7, 8, 9)

Comme nous pouvons le voir, la scène est identique pour les deux solutions, mais dans le second cas, nous pouvons parfaitement distinguer les trois polygones en tant que deux objets distincts sans aucun calcul, le premier polygone étant entièrement défini avant que les sommets du second ne soient indiqués.

Cette méthode est applicable dès le chargement du fichier, et ce, sans aucun calcul.

(2) Utilisation des composantes connexes

L’ensemble de la scène, dans sa version la plus épurée, est un ensemble de points (sommets) et d’arêtes (côtés des polygones) que nous pouvons assimiler à un graphe non orienté. Nous pouvons alors par les techniques dédiées calculer les différentes composantes connexes et les considérer chacune comme un objet différent.

Voici l'algorithme qui permet cette extraction des composantes connexes de la scène :

```
Tant que tous les polygones n'appartiennent pas à un objet
|   Fini ← faux
|   Prendre un polygone non rattaché à un objet
|   Tant que non fini
|   |   Fini ← vrai
|   |   Pour chaque polygone de la scène
|   |   |   Si  ( polygone non rattaché )
|   |   |   |   Si le polygone a un sommet en commun avec un polygone rattaché
|   |   |   |   |   Rattacher le polygone à cet objet
|   |   |   |   |   Fini ← faux
|   |   |   |   Fin si
|   |   |   Fin si
|   |   Fin pour
|   Fin tant que
Fin tant que
```

(3) Les texture, matériaux et couleurs

Lorsqu'un monde virtuel est chargé, celui-ci peut contenir des textures, des informations sur les textures à appliquer (une texture est une image représentant une surface et permettant de simuler l'apparence de celle-ci lorsqu'elle est appliquée sur un polygone), des matériau, ou des couleurs.

Il existe certaines propriétés des matériaux utilisées lors des modélisations de mondes virtuels, principalement lorsque des sources de lumière sont présentes, que nous pouvons utiliser pour déterminer les différentes entités présentes au sein d'une scène. Il s'agit des propriétés d'absorption et de réflexion.

Figure 21 : Texture de bois

Figure 22 : Matériaux possédant différentes propriétés

La figure 21 présente une texture de bois qui peut être appliquée à un élément d'une scène. La figure 22 présente deux objets composés de matériaux aux propriétés différentes. Lorsque ce genre d'objets est modélisé, il contient alors de telles informations.

Nous pouvons alors décomposer la scène en fonction des textures appliquées sur les différentes parties de celle-ci. Nous obtenons une décomposition très proche du visuel, avec des objets pertinents. Nous utilisons la même méthode que précédemment en rajoutant en plus une clause pour la couleur ou la texture.

```
Tant que tous les polygones n'appartiennent pas à un objet
|   Fini ← faux
|   Prendre un polygone non rattaché à un objet
|   Tant que non fini
|   |   Fini ← vrai
|   |   Pour chaque polygone de la scène
|   |   |   Si  ( polygone non rattaché )
|   |   |   |   Si le polygone a un sommet en commun avec un polygone rattaché
|   |   |   |   et qu'une couleur (et/ou texture et/ou matériau) est en commun
|   |   |   |   |   Rattacher le polygone à cet objet
|   |   |   |   |   Fini ← faux
|   |   |   |   Fin si
|   |   |   Fin si
|   |   Fin pour
|   Fin tant que
Fin tant que
```

Par exemple pour le format Wavefront Obj :

```
v       0       2       2
v       0       0       2
v       2       0       2
v       2       2       2
v       0       2       0
v       0       0       0
v       2       0       0
v       2       2       0
usemtl red
f       1       2       3       4
f       8       7       6       5
f       4       3       7       8
usemtl blue
f       5       1       4       8
f       5       6       2       1
f       2       6       7       3
```

Ce code correspond à un cube dont les trois premières faces sont en rouge et les trois suivantes sont en bleu. Chaque triplet de face peut donc être considéré comme un objet ou comme sous objet du cube.

3. Avantages et inconvénients

Nous venons de voir plusieurs techniques servant à la décomposition de la scène en entités distinctes. Bien entendu, toutes ne sont pas applicables en permanence et dépendent de la scène. Par exemple, si aucune couleur ou aucune texture n'est définie, il est inutile d'essayer de les utiliser.

Les avantages sont assez nombreux. Tout d'abord nous avons une méthode fiable, quasiment confondue à l'idée humaine d'objet, vu qu'elle s'appuie sur les informations laissées par le ou les créateurs du monde virtuel. La pertinence de ce découpage ne peut donc pas être remis en cause. De plus, le traitement nécessaire est simple à mettre en place et peut être effectué lors du chargement de la scène. Il peut donc s'apparenter à un précalcul peu coûteux dont l'utilisation est transparente pour l'utilisateur (combinée avec le chargement). Enfin, cette méthode ne nécessite aucun traitement ultérieur, le découpage étant fait une fois pour toutes pour une scène donnée. Il est donc possible de le stocker et de l'intégrer ensuite directement dans le fichier du monde pour éviter tout retraitement.

Le principal problème de cette technique vient de la source même de son efficacité : il s'appuie sur des données cachées et/ou inutilisées mais néanmoins présentes dans la scène. Si les données en question ne sont pas présentes, la méthode ne fonctionne pas. Le second point noir concerne l'adaptabilité de la méthode : il est impossible de spécifier le nombre d'objet que l'on veut obtenir et il est impossible de prévoir à l'avance ce que l'on obtiendra. Cela peut être considéré comme génant si nous nous retrouvons avec un trop grand nombre d'objets.

Pour pallier ces deux inconvénients, nous avons alors besoin d'une méthode qui, même si elle s'avère moins pertinente, doit être utilisable en permanence et permettre de spécifier le nombre d'objets que l'on veut obtenir.

Nous pouvons voir sur la figure 23 l'application de cette technique sur une scène (chaque couleur représente un objet différent) uniquement composée de sommets et de polygones. Les composantes connexes sont alors utilisées pour obtenir ce résultat.

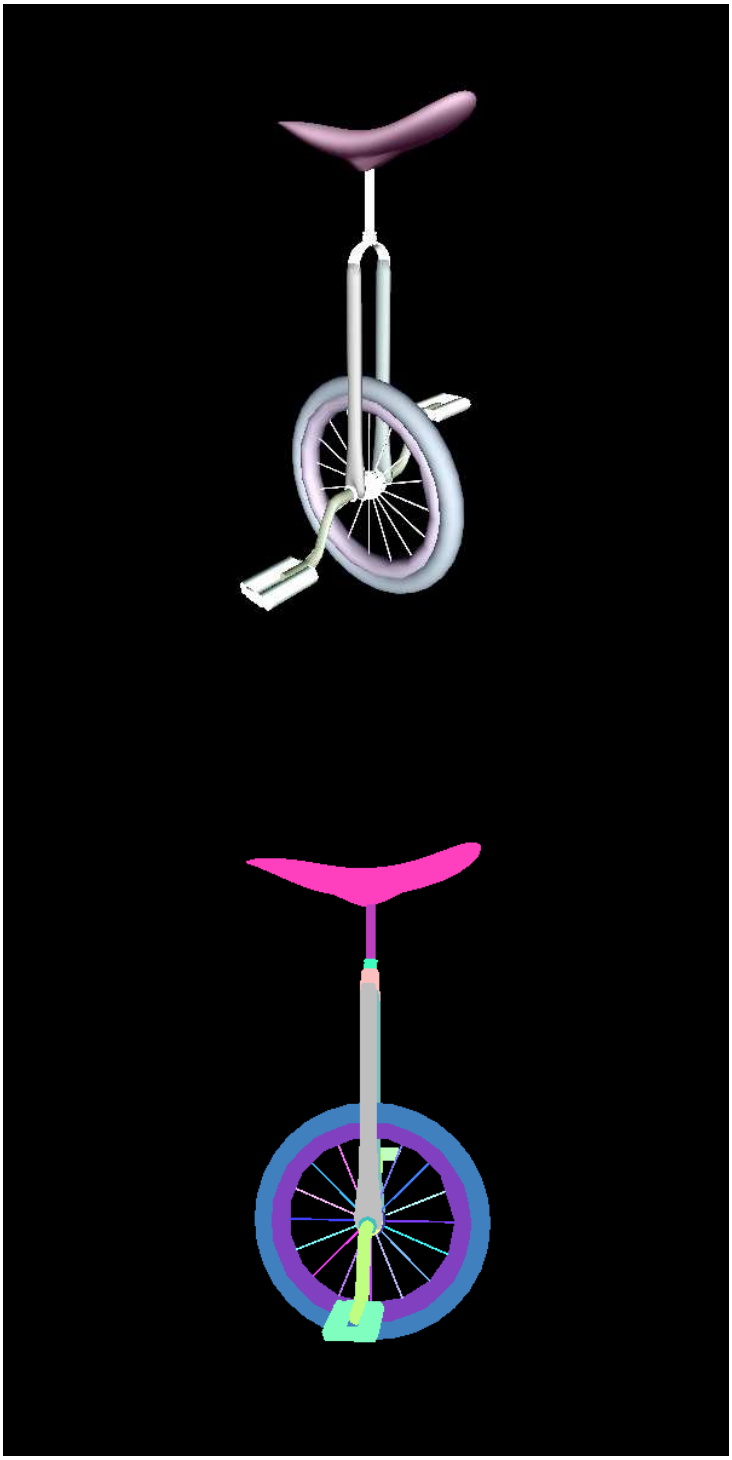


Figure 23 : Décomposition utilisant la modélisation de la scène

B. Méthode de regroupement

Comment faire lorsque la scène ne contient aucune donnée exploitable pour la séparer en différents objets ? Nous allons maintenant voir une méthode, déclinée en deux variantes, qui permet, à partir des points et des polygones composants le monde en trois dimensions, de créer le nombre voulu d'objets.

1. Principe du regroupement

Le principe de la méthode est de partir des polygones qui composent la scène et de les considérer comme des objets séparés (figure 24). Nous les fusionnons alors deux à deux jusqu'à obtenir le nombre voulu d'entités. Pourquoi donner un nombre voulu d'entité ? Parce la méthode se basant juste sur des critères entre polygones ou objets, celle-ci n'a pas d'autres moyens d'arrêter le traitement.

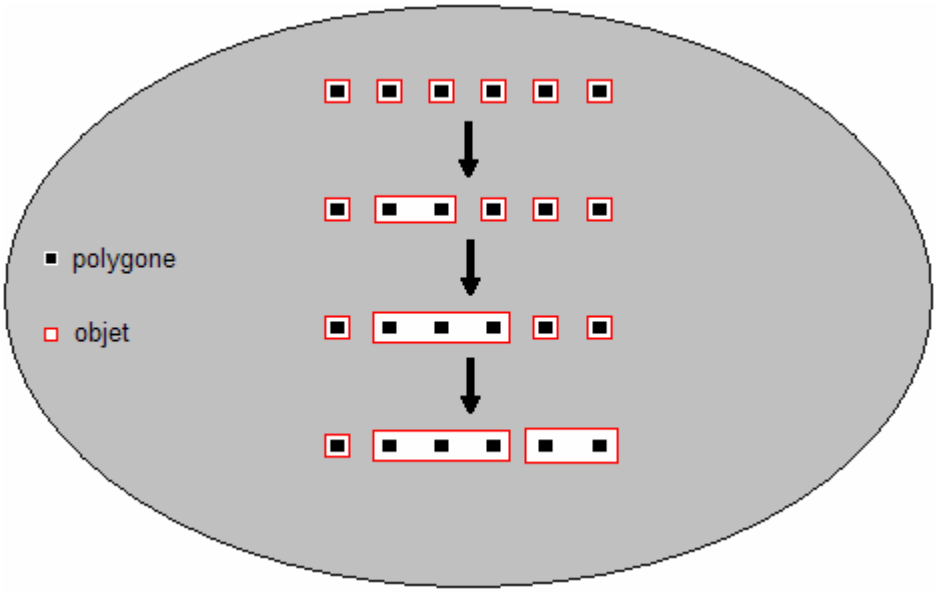


Figure 24 : Regroupement des polygones en objets

Voici l'algorithme principal de fonctionnement :

Scinder la scène en objets (1 polygone = 1 objet)
Prétraitement
Tant que le nombre d'objets est supérieur au nombre désiré
| Sélectionner deux objets O_i et O_j
| Fusionner O_i et O_j en un seul et même objet
Fin

La sélection des deux objets se fait soit par la méthode de proximité, soit par celle du centre de gravité comme nous allons le voir. Il est nécessaire d'avoir un nombre d'objet désiré en critère d'arrêt car cette méthode ne peut s'arrêter d'elle-même.

La fusion est simplement nominative et ne modifie en rien la géométrie de la scène : l'appartenance à un objet est juste une annotation sur chaque polygone.

2. Regroupement par proximité

Le rapprochement par proximité est la première variante de la méthode de regroupement.

Nous commençons par calculer le centre de gravité des polygones. Pour se faire, nous prenons la moyenne des positions des sommets. Nous calculons ensuite les distances euclidiennes entre chaque centre de gravité des différents polygones et nous les ordonnons. Nous obtenons alors la liste ordonnée des proximités entre les différents polygones. Cette liste est alors utilisée pour définir les objets à fusionner.

La méthode fonctionne en utilisant cet algorithme :

Calculer les centres de gravité des polygones
Attribuer chaque polygone à un objet
Pour chaque polygone P_i du premier P_0 au dernier P_n
| Pour chaque polygone P_j de P_{i+1} au dernier P_n
| | Ajouter dans la « liste des distances » le triplet (P_i, P_j, distance(P_i, P_j))
| Fin
Fin
Trier la « liste des distances » par ordre croissant.
Tant que le nombre d'objets est supérieur au nombre désiré
| Prendre le premier triplet (P_i, P_j, distance(P_i, P_j)) de la liste ordonnée
| Si P_i et P_j n'appartiennent pas au même objet
| | Fusionner les objets respectifs de P_i et P_j
| Fin
| Supprimer le triplet (P_i, P_j, distance(P_i, P_j)) de la liste ordonnée
Fin

Ce type de regroupement représente certains avantages. Les calculs se font dans un ordre bien défini et restent valables jusqu'à la fin du traitement. De plus, chaque étape est bien distincte des autres. L'intérêt premier de cette méthode est qu'elle utilise une notion de proximité entre les polygones et donne une séparation en objets très intéressante lorsque les sous parties sont proches les unes des autres.

L'inconvénient majeur est la possibilité d'obtenir des objets disproportionnés les uns par rapport aux autres. Cela provient du fait qu'aucune vérification et qu'aucun calcul n'est fait par rapport aux objets définis précédemment. La liste servant de guide est générée au tout début et n'est plus modifiée par la suite que pour supprimer les éléments traités. Elle n'est pas recalculée et triée suivant de nouveaux paramètres. La figure 25 présente les résultats de cette méthode.

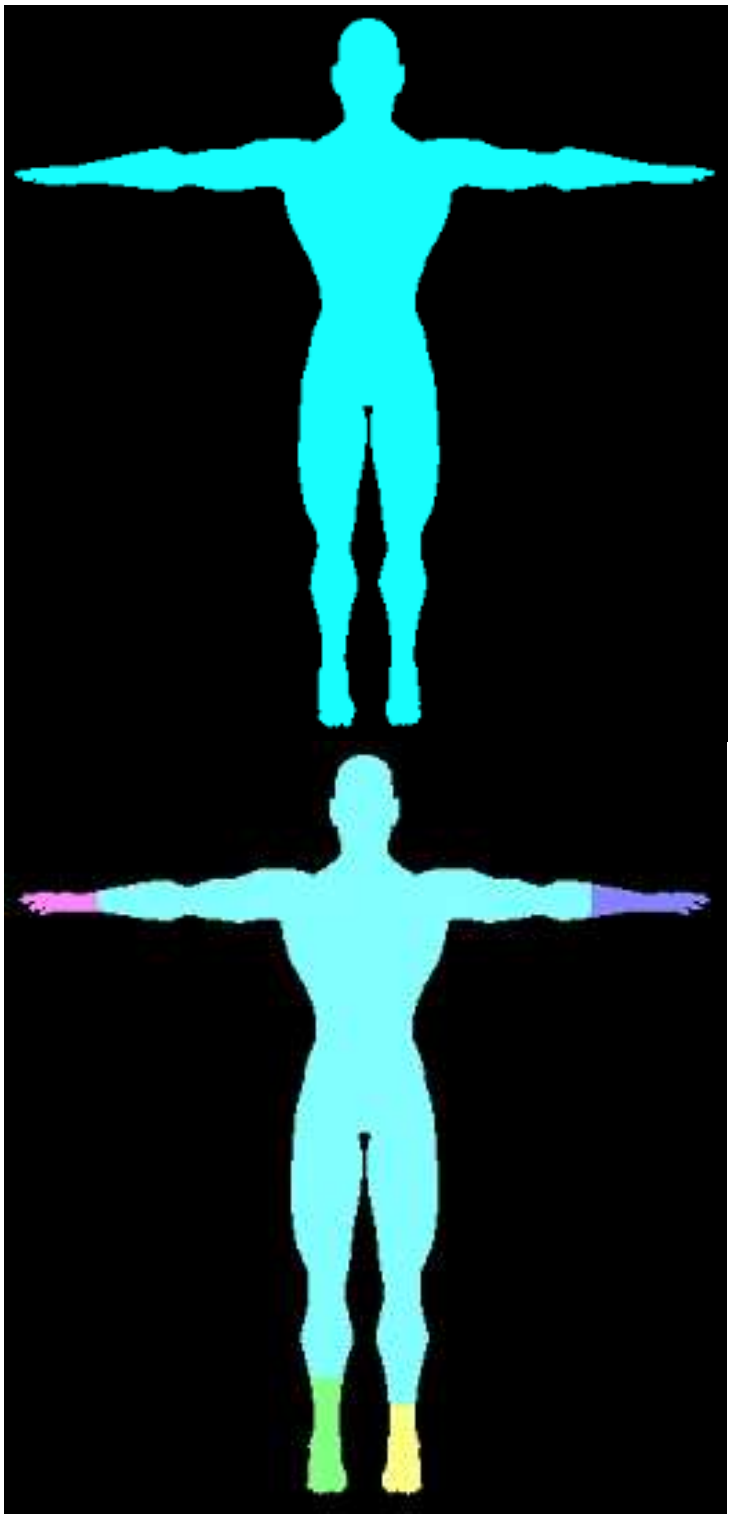


Figure 25 : Décomposition d'un objet - méthode de regroupement par proximité

3. Regroupement par centre de gravité

Le regroupement par centre de gravité corrige ce problème. La liste ordonnée est recalculée lors de chaque itération et ne se base plus sur le centre de gravité des polygones mais sur celui des objets (qui est le centre de gravité des différents centres de gravité des polygones qui le composent). La figure 26 illustre la décomposition par centre de gravité.

La méthode fonctionne en utilisant cet algorithme :

Attribuer chaque polygone à un objet
Calculer les centres de gravité des objets
Pour chaque objet O_i du premier O_0 au dernier O_n
| Pour chaque objet O_j de O_{i+1} au dernier O_n
| | Ajouter dans la « liste des distances » le triplet (O_i, O_j, distance(O_i, O_j))
| Fin
Fin
Trier la « liste des distances » par ordre croissant
Tant que le nombre d'objets est supérieur au nombre désiré
| Prendre le premier triplet (O_i, O_j, distance(O_i, O_j)) de la liste ordonnée
| Fusionner les objets O_i et O_j en un seul objet O_i *
| Recalculer le centre de gravité de O_i
| Supprimer les triplets contenant O_i ou O_j de la « liste des distances »
| Pour chaque objet O_j du premier O_0 au dernier O_n
| | Si (O_j différent de O_i)
| | | Ajouter le triplet (O_i, O_j, distance(O_i, O_j)) à la « liste des distances»
| | Fin
| Fin
| Trier la « liste des distances » par ordre croissant
Fin

* L'objet O_j n'existe plus en tant qu'objet à ce moment là. L'objet O_i l'absorbe.

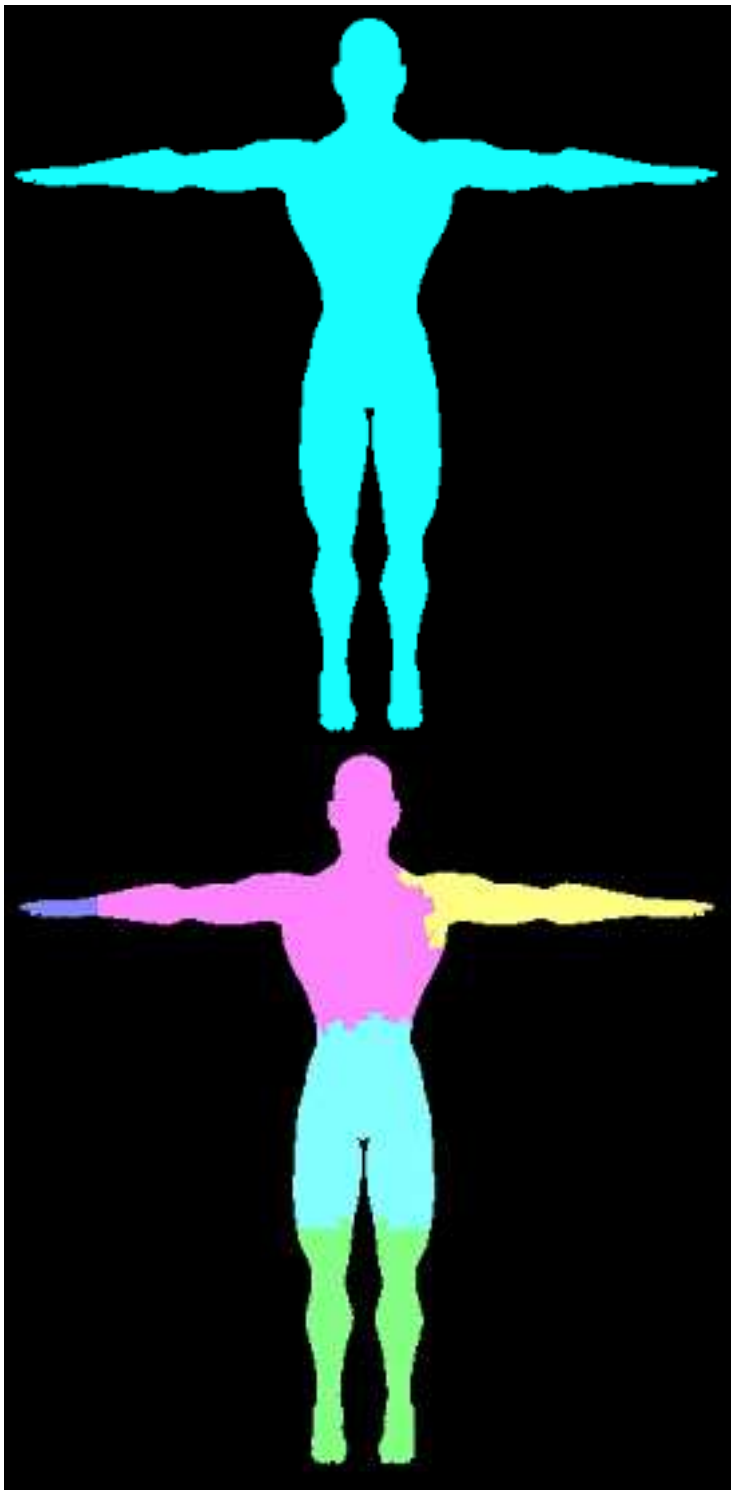


Figure 26 : Décomposition d'un objet - méthode de regroupement par centre de gravité

4. Avantages et inconvénients

Le principal avantage de cette technique est que celle-ci fonctionne dans tous les cas possibles. En se basant sur les sommets et les polygones, elle assure d'être utilisable quelle que soit la scène sur laquelle elle va être appliquée.

Par contre, cette méthode ne peut pas se vanter d'obtenir des résultats aussi convaincants que ceux obtenus par la méthode de décomposition.

Nous obtenons donc ici une technique très intéressante qui permet de pallier les défauts de cette dernière tout en gardant des résultats satisfaisants pour le calcul de bons points de vue.

Le choix entre les deux solutions offertes (proximité ou gravité) et le résultat obtenu sera variable en fonction des scènes même si la méthode de gravité donne, en général, de meilleurs résultats. En effet, la plupart du temps, les objets obtenus sont de tailles plus équilibrées que par le regroupement par proximité.

Nous pouvons voir l'application de ces deux possibilités sur une même scène (figures 27 et 28).

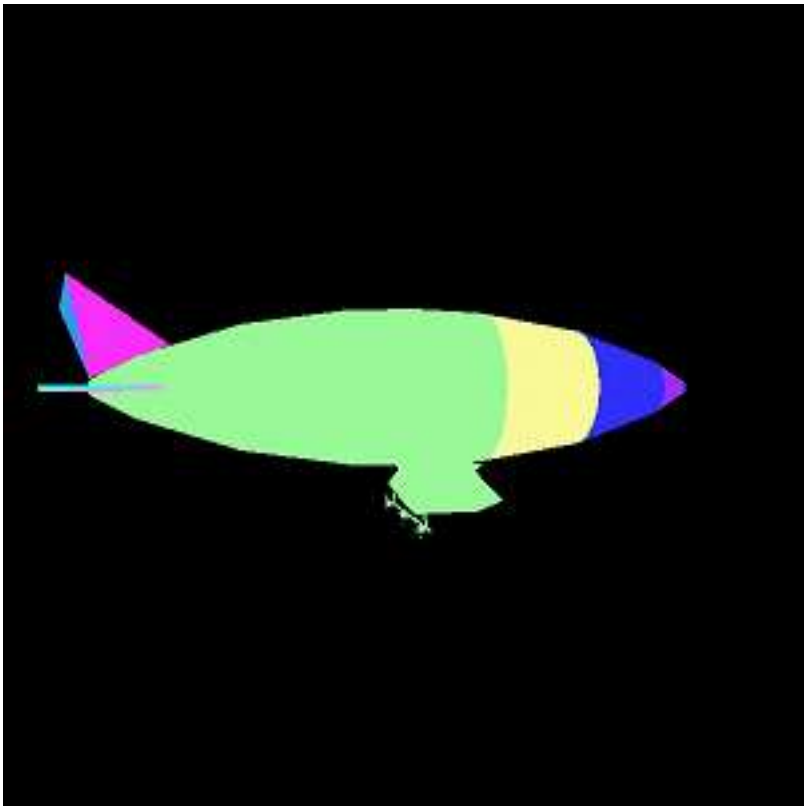


Figure 27 : Regroupement par proximité (10 objets demandés)

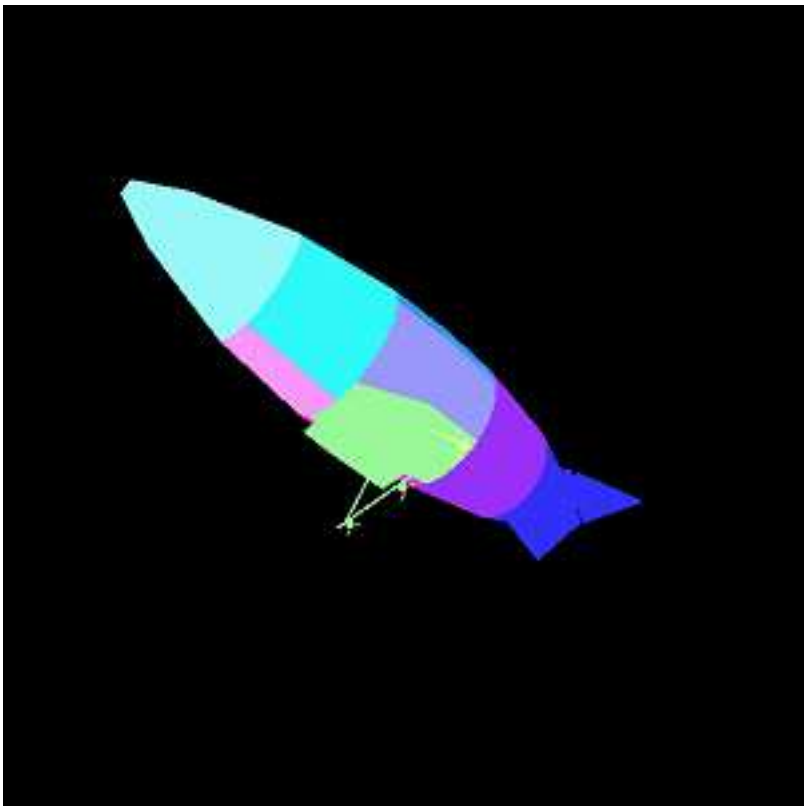


Figure 28 : Regroupement par centre de gravité (10 objets demandés)

C. Méthodes combinées

Nous venons donc de définir deux méthodes utilisables en fonction des différentes caractéristiques de la scène à découper. Le problème qui se pose est donc de choisir quelle méthode utiliser, ou de les essayer toutes les deux, afin de sélectionner celle qui donne les meilleurs résultats.

Or, comme nous venons de le voir, les deux solutions sont complémentaires et utilisent des approches différentes. Pourquoi ne pas fusionner les deux fonctionnements afin de réunir tous les avantages et tenter d'éliminer la plupart des défauts ?

Nous allons donc voir comment fonctionne la méthode issue du regroupement des deux premières techniques.

1. Initialisation

Lors de l'initialisation de la méthode, l'utilisateur va avoir à définir plusieurs paramètres afin de scinder la scène en objets de la façon qu'il juge la plus pertinente.

Il devra tout d'abord définir les critères qui vont servir dans la première partie puis ensuite choisir le nombre d'objets qu'il désire (ou alors un intervalle pour ce nombre) qui sera utilisé dans la seconde partie.

2. Première partie : utilisation de la méthode de décomposition

La première partie de cette méthode combinée est tout simplement la même que la méthode de décomposition de la scène en objets. L'utilisateur définit, lors de l'initialisation, les critères qui vont être utilisés pour tenter de créer différents objets à partir de la scène et la méthode est alors utilisée sur la scène.

A la fin de cette première partie, nous obtenons un certain nombre d'objets (le même bien entendu que si nous avions utilisé la méthode de décomposition seule) qui va conditionner la suite du traitement.

Si le nombre d'objets obtenus correspond au nombre d'objets voulus par l'utilisateur, le résultat obtenu est satisfaisant et la méthode s'arrête là. Dans le cas contraire, un second traitement est lancé en fonction du nombre d'objets courants par rapport au nombre voulu.

3. Seconde partie : utilisation des méthodes de regroupement

La seconde partie de la méthode conjointe utilise le principe de la méthode de regroupement, ou tout du moins une variante de celle-ci, laissée au choix de l'utilisateur (nous utilisons le regroupement par centre de gravité par défaut). La variation utilisée va dépendre du fait que l'on ait besoin d'un plus grand nombre, ou d'un plus petit nombre, d'objets que ce que la première partie de la méthode a obtenu.

Dans le cas où le nombre d'objets est trop important et que nous devons donc le réduire, nous utilisons la méthode de réduction. A l'inverse, lorsque le nombre d'objets est trop faible, nous utilisons le traitement d'augmentation.

Ces deux méthodes vont progressivement augmenter ou diminuer le nombre d'objets de la scène (un seul et unique objet par itération). Le temps de calcul nécessaire peut donc être très important.

a) Traitement de Réduction

La fonction de réduction ne s'applique que si le nombre d'objets de la scène temporaire est trop grand pour répondre à la demande de l'utilisateur. Au lancement de cette méthode, nous avons donc une scène, prédécoupée en objets (lors de la première partie), dont nous devons réduire le nombre d'objets.

Cette partie est très proche de la seconde méthode employée seule. Nous allons ici fusionner deux objets pour obtenir une scène composée d'un objet de moins que celle que nous avions initialement. Bien entendu, le principe varie légèrement en fonction de la méthode de regroupement utilisée. Nous allons donc décrire succinctement les deux approches.

(1) Par proximité

Nous calculons tout d'abord le centre de gravité de chacun des polygones de la scène. Dans un second temps, pour chaque paire d'objets, nous calculons la plus petite distance entre les polygones qui les composent. Les deux objets étant alors les plus proches sont fusionnés en un seul et même objet. Nous obtenons alors une scène avec un objet de moins.

La méthode fonctionne en utilisant cet algorithme :

Calculer les centres de gravité des polygones
Pour chaque objet O_i du premier O_0 au dernier O_n
| Pour chaque objet O_j de O_{i+1} au dernier O_n
| | distance_ O_i_ O_j ← ∞
| | | Pour chaque polygone P_i de O_i
| | | | Pour chaque polygone P_j de O_j
| | | | | Si distance(P_i, P_j) < distance_ O_i_ O_j
| | | | | | distance_ O_i_ O_j ← distance(P_i, P_j)
| | | | | Fin
| | | | Fin
| | | Fin
| | Fin
| Fin
Fin
Fusionner les deux objets O_1 et O_2 tels que distance_ O_1_ O_2 soit minimale

En cas d'égalité sur les distances minimales entre les objets nous sélectionnons ceux dont la somme des volumes des boîtes englobantes est le plus faible. Cela permet d'obtenir des objets ayant des disparités de volumes moins importants lorsque le cas se présente.

(2) Par centre de gravité

Nous commençons par calculer le centre de gravité de chacun des objets de la scène. Nous calculons ensuite les distances entre ces points servant de références pour chaque paire possible d'objets. Les deux objets étant alors les plus proches sont fusionnés en un seul et même objet. Nous obtenons alors une scène avec un objet de moins.

La méthode fonctionne en utilisant cet algorithme :

Calculer les centres de gravité des objets
Pour chaque objet O_i du premier O_0 à l'avant dernier O_{n-1}
| Pour chaque objet O_j de O_{i+1} au dernier O_n
| | Calculer la distance_ O_i_ O_j
| Fin
Fin
Fusionner les deux objets O_1 et O_2 tels que distance_ O_1_ O_2 soit minimale

Comme dans la variante précédente, en cas d'égalité sur les distances minimales entre les objets nous sélectionnons ceux dont la somme des volumes des boîtes englobantes est le plus faible. De la même façon, cela permet d'obtenir des objets ayant des disparités de volumes moins importants lorsque le cas se présente.

b) Traitement d'Augmentation

La fonction d'augmentation s'applique lorsque le nombre d'objet de la scène temporaire est trop faible pour répondre à la demande de l'utilisateur. Nous avons donc, en entrée de cette méthode, une scène prédécoupée en objets par l'utilisation de la première partie, dont nous devons augmenter le nombre d'objets.

Cette partie est relativement proche de la méthode de regroupement utilisée seule mais varie néanmoins sur certains points importants. Nous allons ici découper un objet en deux parties qui seront alors considérées chacune comme un objet indépendant. L'objet initial en tant que tel n'existera alors plus.

Nous ne détaillerons pas ici la méthode de décomposition que nous avons vue précédemment, celle-ci est appliquée telle que décrite pour ses deux variantes. Cependant nous allons détailler comment définir sur quoi l'appliquer et de quelle manière.

La méthode fonctionne sur le principe que nous allons maintenant décrire. Nous prenons l'ensemble des objets de la scène et nous calculons le volume des boites englobantes de chacun de ces objets. Nous sélectionnons alors l'objet qui a la boite la plus volumineuse pour lui appliquer la méthode de regroupement (avec l'une ou l'autre des variantes) et regrouper les polygones qui le composent en deux objets distincts.

La méthode fonctionne en utilisant cet algorithme :

- Calculer les boites englobantes des objets qui composent la scène
- Sélectionner O : l'objet ayant le plus grand volume de boite englobante
- Appliquer la méthode de regroupement sur l'objet O comme s'il s'agissait d'une scène indépendante.

En cas d'égalité lors du calcul des volumes des boites englobantes de objets, le premier objet correspondant est sélectionné.

4. Récapitulatif du fonctionnement de la méthode

La figure 29 présente le schéma récapitulatif du fonctionnement des méthodes combinées.

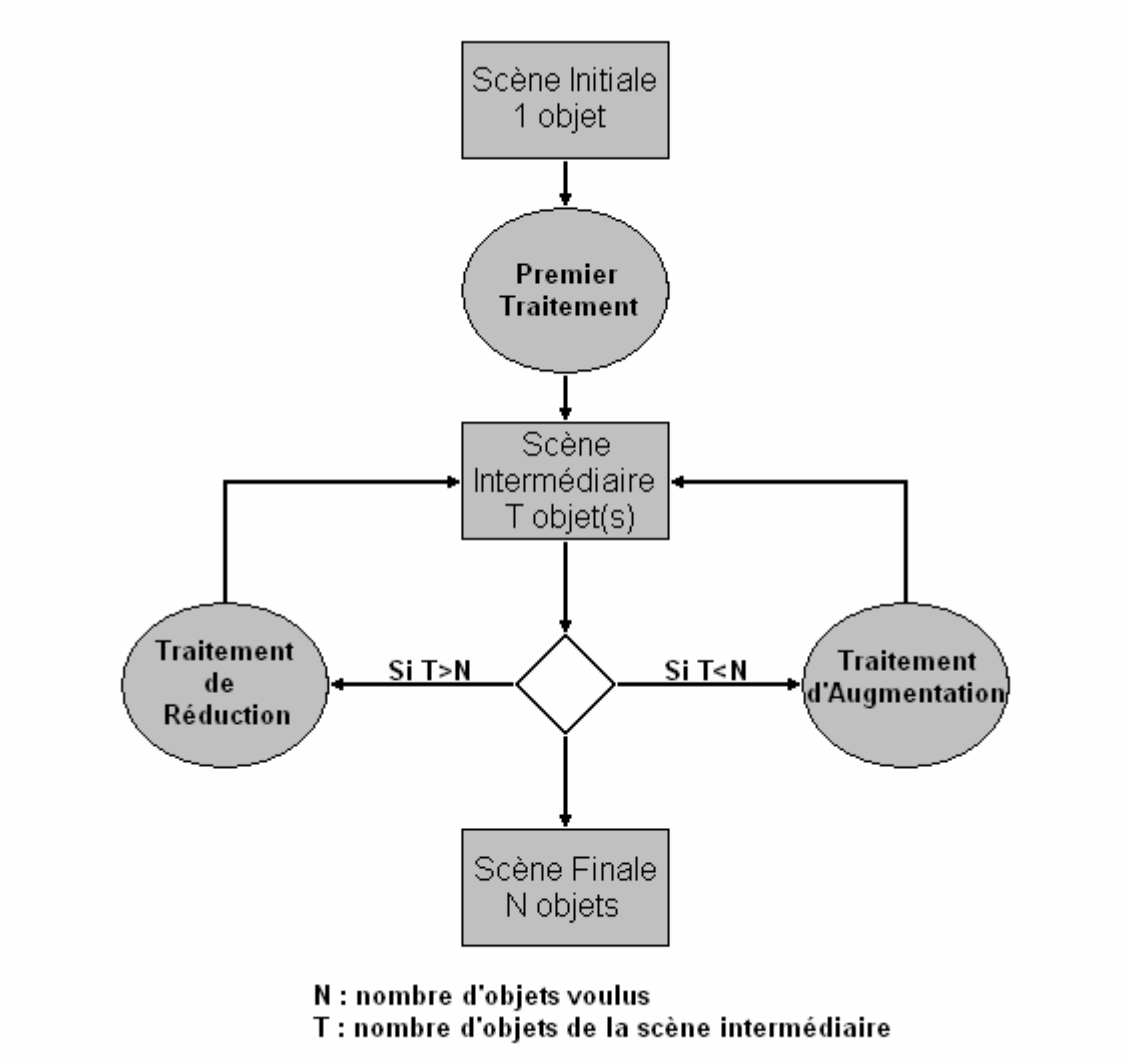


Figure 29 : Diagramme des méthodes combinées

Voici quelques résultats des différentes méthodes que nous venons de voir sur une même scène. La figure 30 présente la décomposition d'un objet en utilisant les composantes connexes de la scène.

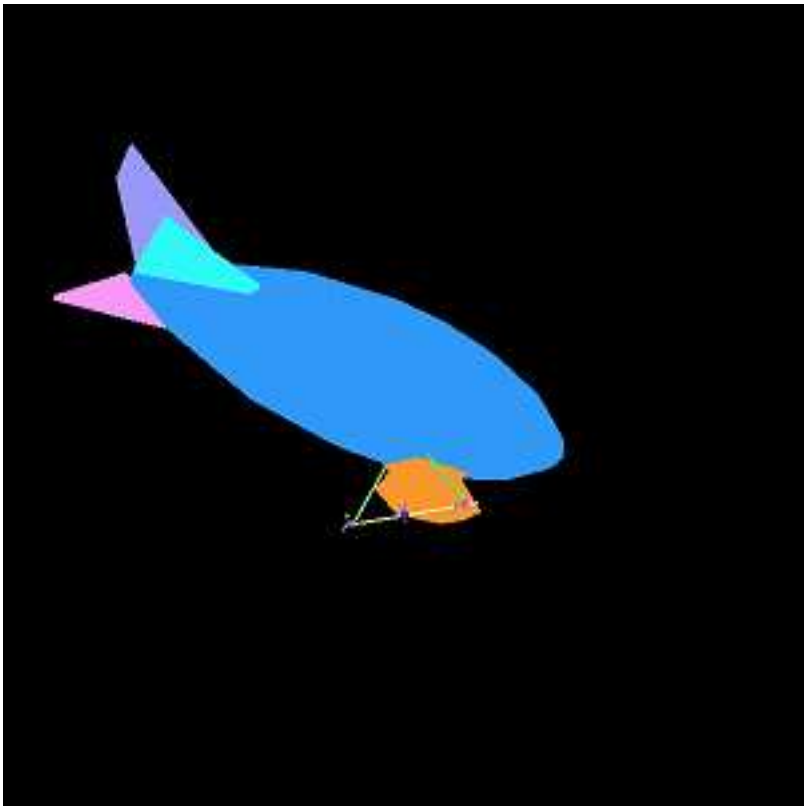


Figure 30 : Décomposition par utilisation des composantes connexes

La figure 31 présente l'utilisation de la méthode de regroupement en utilisant le centre de gravité sur ce même objet.

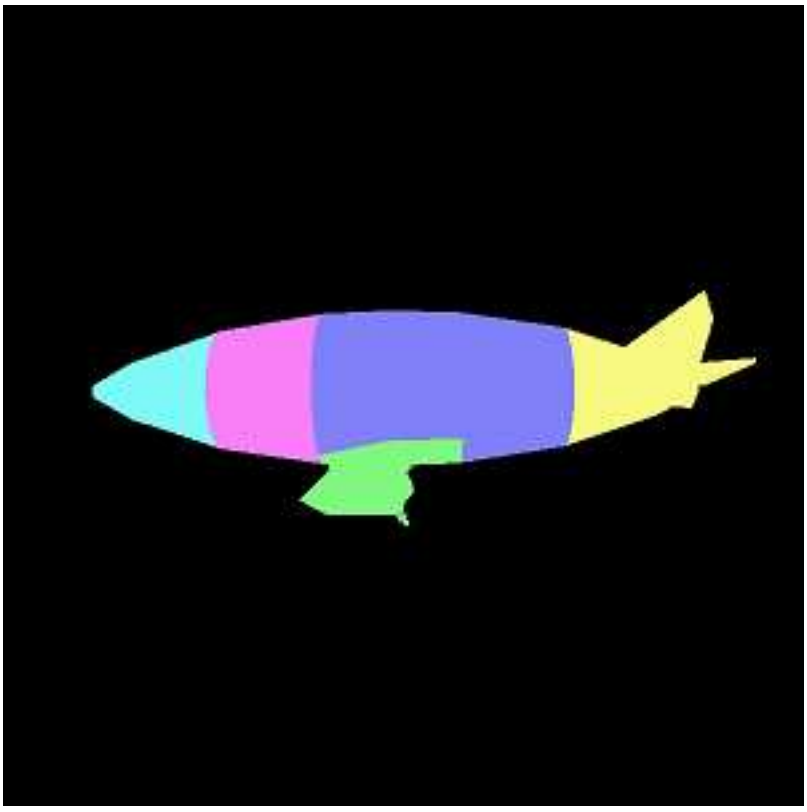


Figure 31 : Regroupement avec la méthode du centre de gravité

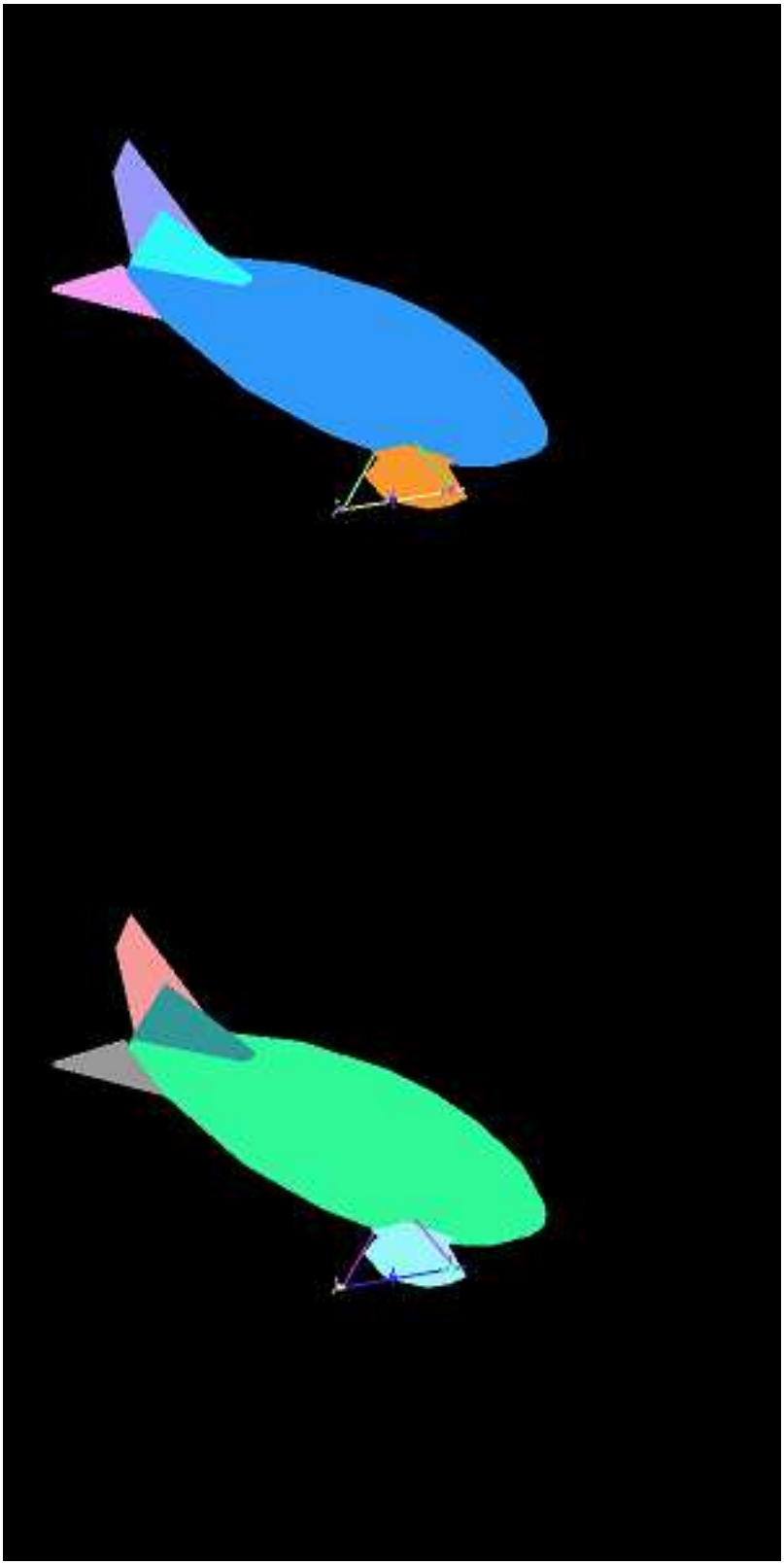


Figure 32 : Utilisation des méthodes combinées (avec différents nombres d'objets)

L’utilisation des méthodes combinées présentée sur la figure 32 montre les résultats avec un nombre d’objets demandés différent.

D. Conclusion

Les techniques de décomposition et de regroupement que nous venons de voir permettent d'inclure, dans une scène qui ne la possèderait pas, la notion d'objet. La solution retenue (méthodes conjointes), grâce à la gestion de différents cas de figure, est utilisable quelle que soit la scène sur laquelle nous l'appliquons.

Dans une bonne partie des scènes, les résultats obtenus sont très bons, et, lorsque celles-ci contiennent des informations utilisables, nous obtenons un découpage en objets confondu avec celui de la perception humaine. Dans les autres cas, notre méthode offre des résultats satisfaisants, suffisants pour un calcul de points de vue et une exploration plus pertinents que sur une scène globale.

Ces méthodes, utilisables principalement lorsque les objets ne sont pas définis dans la scène, prennent toutes leur dimension lorsque l'utilisateur n'est pas le concepteur de la scène et préfère une méthode automatiquement à un traitement humain, plus long et plus complexe.

Chapitre IV
Exploration et Trajectoire

IV. Exploration et trajectoire

Nous venons de présenter des méthodes permettant de calculer un ensemble de bons points de vue pour un monde virtuel en trois dimensions. Cependant, si pour un objet simple un ensemble de points de vue peut être satisfaisant pour avoir une connaissance correcte de la scène, il n'en est pas de même pour les scènes complexes. Des points de vue, sur différentes parties du monde, se trouvant parfois à une grande distance les uns des autres, peuvent désorienter l'utilisateur facilement et finalement ne rien apporter en terme de connaissance réelle.

Il est donc nécessaire d'étudier la possibilité de générer une trajectoire automatiquement dans le but de relier ces points de vue et de permettre à l'utilisateur d'appréhender correctement l'environnement qu'il veut explorer.

Nous allons donc nous attacher à décrire des méthodes de création de trajectoires, répondant à différents besoins en fonction des différents types d'exploration que l'utilisateur désire. Dans un premier temps, nous présenterons certaines contraintes inhérentes aux trajectoires et à leur création. Nous étudierons ensuite l'exploration « externe » des objets et nous verrons à quel moment les utiliser. Enfin, nous traiterons les explorations « internes » lorsque la trajectoire passe à l'intérieur de la scène.

A. Propriétés et contraintes sur les trajectoires

Une trajectoire se doit d'avoir un certain nombre de propriétés sans lesquelles celle-ci perd de sa qualité et/ou de son intérêt. Une trajectoire peut être assimilée à une fonction du temps qui fait correspondre une position (avec tout ce que celle-ci implique) à un instant donné. La réunion de ces positions et de ces instants permet d'obtenir et de définir la trajectoire.

Nous pouvons séparer les propriétés et les contraintes sur les trajectoires en deux types distincts. Tout d'abord nous traiterons des propriétés qui sont obligatoires pour qu'une trajectoire soit jugée « correcte ». Dans un second temps, nous parlerons des contraintes plus

particulières qui ne s'appliquent qu'à certains types d'exploration et donc à certaines trajectoires.

1. Propriétés et contraintes générales

Les contraintes sur les différentes trajectoires sont fortement dépendantes de ce que voudra obtenir l'utilisateur. En fonction des différentes utilisations possibles, il peut être nécessaire de laisser certaines d'entre elles de côté ou au contraire d'en forcer d'autres. Voici les différentes contraintes que nous avons retenues sur les trajectoires en général.

a) Longueur d'une trajectoire

Il est toujours plus intéressant de tenter de minimiser la trajectoire dans une certaine mesure. Une trajectoire trop longue est moins pertinente et entraîne forcément une redondance de l'information qui n'est pas toujours souhaitable. Bien entendu, il peut être nécessaire de s'attarder sur un point important mais le reste de la trajectoire se doit de minimiser les détours, les boucles et les superpositions de parties de trajectoires.

b) Fluidité et continuité

Pour ne pas trop gêner l'utilisateur, la trajectoire doit avoir une certaine homogénéité. Il faut au maximum éviter les à-coups et les changements brusques de direction. De plus le déplacement doit se faire à intervalles réguliers et sans saut d'une position à une autre (sauf si c'est volontaire bien évidemment mais dans ce cas il est plus prudent de parler de plusieurs trajectoires distinctes).

c) Gestion des obstacles

Les obstacles peuvent être un problème dans la création des trajectoires. Tant que nous restons sur la sphère englobante de la scène il n'y a aucun problème mais plus nous nous approchons de la scène et plus le risque de heurter des éléments de celle-ci devient important. Il faut donc une méthode de création de trajectoire qui est capable de prendre en compte ces obstacles ou qui évite les possibilités de collision avec les éléments.

2. Propriétés et contraintes particulières

Certains types de contraintes sur les trajectoires peuvent être qualifiés de contraintes particulières car n'étant pertinents que sur des créations de trajectoires spécifiques. Ces cas spéciaux sont cependant problématiques et vont obliger la modification des méthodes et des calculs plus importants. Nous allons maintenant décrire quelques uns des cas qui peuvent se présenter à nous lors de la création de trajectoire.

a) Ordre des points de passage

Au début de la création de trajectoire, nous avons un ensemble de points de vue à relier entre eux. Dans le cas général, pour optimiser la longueur de la trajectoire, nous essayons de réorganiser et de trier cet ensemble de points de vue. Cependant, il peut être nécessaire d'avoir un ordre précis pour les différents points de passage. C'est le cas notamment lors d'une cinématique ou d'une visite guidée programmée où l'ordre des différents points de vue ne laisse pas place au hasard. Nous allons voir deux tris qui permettent d'organiser l'ensemble de points de vue dans le but d'optimiser la taille de la trajectoire.

(1) Tri par proximité

Nous classons l'ensemble des points de vue en prenant en compte la distance entre le point de vue actuel et le suivant. Le premier point de vue choisi est celui qui possède la meilleure évaluation. Nous prendrons alors comme point de vue suivant celui qui est le plus proche de lui. Le troisième point de vue sera celui qui est le plus proche du second et ainsi de suite. Bien entendu un point de vue classé n'est plus pris en compte.

Cette méthode permet de classer les points rapidement et de manière correcte. Les trajectoires sont en général assez intéressantes car elles ne reviennent que rarement sur elles mêmes. Il peut cependant arriver, lorsque de nombreux points sont regroupés, que ce type de classement ne soit pas satisfaisant car la trajectoire a alors tendance à faire des boucles.

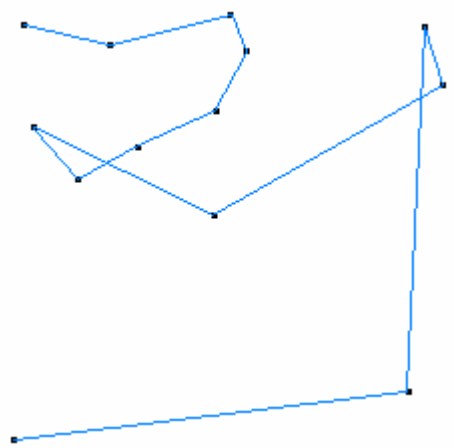


Figure 33 : Tri par proximité

La figure 33 présente l'ordre choisi pour les différents points en fonction du critère (ici par proximité).

(2) Tri par chemin minimal

Nous classons les points de vue en faisant en sorte que le chemin reliant l'ensemble des points soit le plus petit possible en utilisant des algorithmes tels que ceux utilisés dans la théorie des graphes. Chaque point de vue est considéré comme un nœud du graphe et un arc relie chacun des points à chacun des autres. Les arcs ainsi créés sont à double sens (graphe non orienté) et leur valeur est la distance euclidienne entre les points qui lui servent de base.

Cet ordonnancement des points est le plus satisfaisant au niveau des résultats mais nécessite des temps de calculs plus importants, à plus forte raison lorsque le nombre de points de vue augmente.

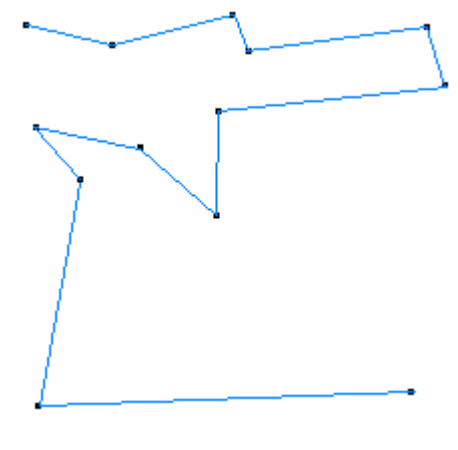


Figure 34 : Tri par chemin minimal

La figure 34 présente l'ordre choisi pour les différents points en fonction du critère (ici par chemin minimal).

b) Distance par rapport aux éléments

Lorsque la trajectoire ne se situe pas sur une sphère englobante et que celle-ci peut se rapprocher des objets, la simple gestion de collision avec les obstacles n'est pas toujours suffisante. En effet, le fait d'être trop proche, ou trop éloigné mais dans une moindre mesure, d'une entité peut nuire à la visibilité. Si dans la majorité des cas, cela n'est pas un problème car les passages de trajectoires à proximité d'objets sont rares, il peut s'avérer nécessaire d'essayer de créer une trajectoire qui passe à une distance « respectable » des différents éléments de la scène.

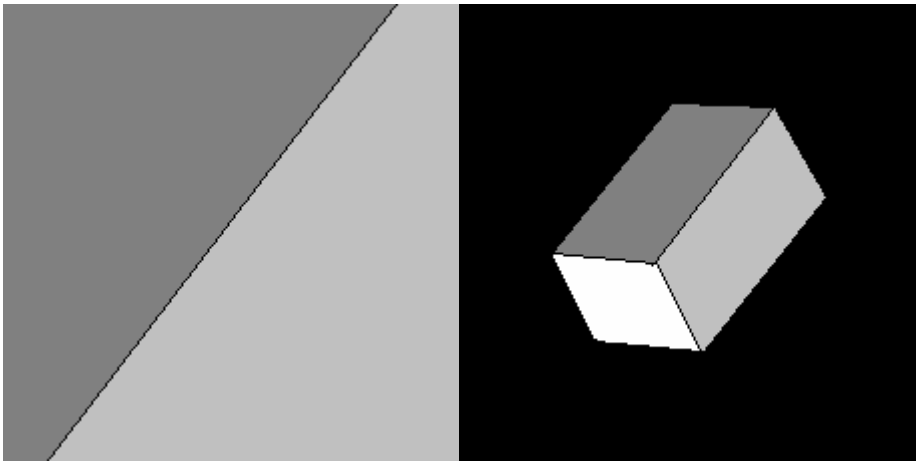


Figure 35 : Point de vue trop proche d'un objet et point de vue correct

La figure 35 montre le problème de visibilité que peut entraîner la trop grande proximité d'un point de vue lors de l'exploration.

c) Timing et synchronisation

La synchronisation est une contrainte vraiment particulière qui ne va apparaître que lors de certaines scènes animées. Chaque point de vue que nous devons relier par une trajectoire est choisi spécifiquement et valable pour un unique moment. Il est alors nécessaire que la trajectoire passe par ce point ci, ou par un point assez proche, à l'instant idoine.

Il y a alors deux paramètres qui peuvent varier pour assurer la synchronisation. Le premier est la longueur de la trajectoire entre deux points : en allongeant la trajectoire il est possible de retarder l'arrivée de la caméra au point synchronisé. Il n'est cependant pas « possible » de diminuer la taille d'une trajectoire en dessous de son chemin minimal (la ligne droite entre deux points) et jouer sur la longueur ne peut donc permettre de retarder l'arrivée au point de vue. Le deuxième paramètre est la vitesse de déplacement entre un point à un autre. En faisant varier ce dernier, nous pouvons différer dans les deux sens l'arrivée à un point de passage, soit en accélérant le déplacement, soit en le ralentissant. Le problème est que nous n'avons alors aucune certitude sur la continuité de la trajectoire et la question du confort de l'utilisateur peut se poser. En effet, de brusques changements sur la vitesse comme sur la direction peuvent incommoder et il est donc nécessaire de se prémunir. De plus, assurer

l’arrivée au bon endroit et au bon moment d’une trajectoire complique les calculs et devient problématique lors des calculs en temps réel.

Lors de nos travaux, nous n’avons pas implémenté de méthode répondant à cette contrainte et nous sommes donc resté sur la partie théorique de celle-ci.

B. Création de trajectoire sans gestion d’obstacles

La création de trajectoire sans gestion d’obstacles permet de générer en temps réel une trajectoire satisfaisante avec très peu de calculs. Nous allons présenter quelques types de trajectoires simples ici. Les collisions ne sont pas traitées dans cette partie.

1. Création de trajectoire sur une sphère englobante

Le but de cette méthode est d’éviter les changements brusques de direction dans la trajectoire afin que celle-ci soit aussi fluide que possible. Pour cela le principe est le suivant. Le point courant possède un vecteur de direction « v » fonction de sa trajectoire actuelle. Le vecteur directeur référence est le vecteur normalisé « U_{CB} » où C est le point courant et B le point à atteindre. Nous allons décrire la technique en coordonnées sphériques. La méthode est la même pour les coordonnées cartésiennes sauf que le système est étendu en trois dimensions puis ramené en deux, comme sur la méthode précédente.

Nous notons donc ici C le point courant, U_C son vecteur directeur (qui indique sa direction), A le point à atteindre et U_{AC} le vecteur direction indiquant la direction du point A à partir du point C. Chacune de ces variables est fonction de deux paramètres φ et θ.

Nous avons :

$$U_{AC}\varphi = \frac{A\varphi - C\varphi}{\sqrt{(A\varphi - C\varphi)^2 + (A\theta - C\theta)^2}}$$
$$U_{AC}\theta = \frac{A\theta - C\theta}{\sqrt{(A\varphi - C\varphi)^2 + (A\theta - C\theta)^2}}$$

Ces deux valeurs normalisées correspondent donc à deux valeurs comprises entre 0 et 1 telles que leur somme au carré est égale à 1. Elles appartiennent donc au cercle trigonométrique de rayon 1 et nous pouvons donc leur faire correspondre une valeur α telle que $\cos(\alpha) = U_{AC}\varphi$ et $\sin(\alpha) = U_{AC}\theta$.

Nous pouvons bien sûr procéder de même avec le vecteur U_C et trouver un angle β qui vérifie $\cos(\beta) = U_C\varphi$ et $\sin(\beta) = U_C\theta$.

Une simple condition sur la différence entre les deux angles obtenus suffit pour savoir s'il faut incrémenter ou décrémenter β pour qu'il tende vers α. Une fois la valeur de β modifiée nous avons les nouvelles valeurs $U_C\varphi$ et $U_C\theta$ qui lui correspondent et donc la nouvelle orientation de la trajectoire.

Ce procédé est illustré par les figure 36 : la trajectoire est crée de manière incrémentale en modifiant légèrement la direction de celle-ci jusqu'à l'orienter vers le point de vue suivant. La position courante est au centre du cercle trigonométrique dont les axes sont les composantes φ et θ des coordonnées sphériques. L'angle α est celui qui indique la direction du prochain point à atteindre. L'angle β est l'angle qui indique la direction courante de la trajectoire. Nous augmentons alors cet angle pour orienter la trajectoire vers le prochain point à atteindre (ou le diminuons suivant la différence entre les deux angles comprise entre $]-\pi,\pi]$ modulo 2π).

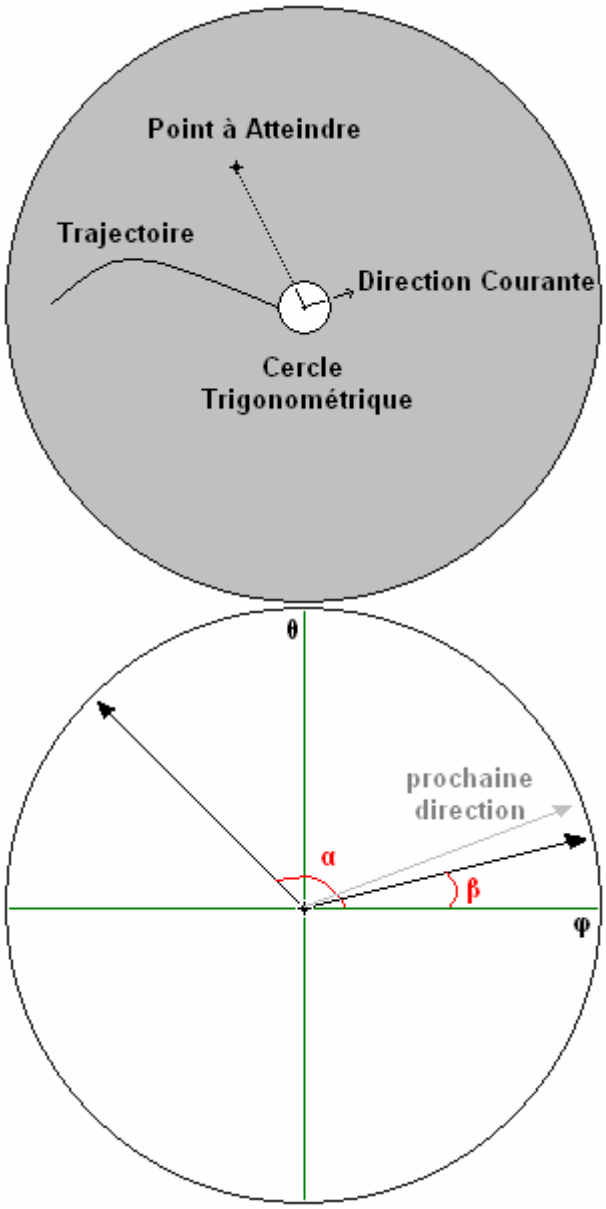


Figure 36 : Création de trajectoire sur sphère englobante

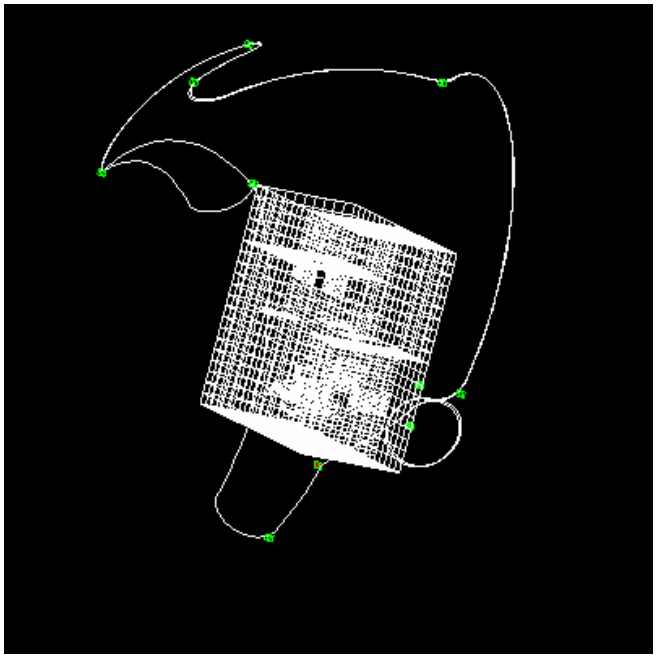


Figure 37 : Création de trajectoire sur sphère englobante (1)

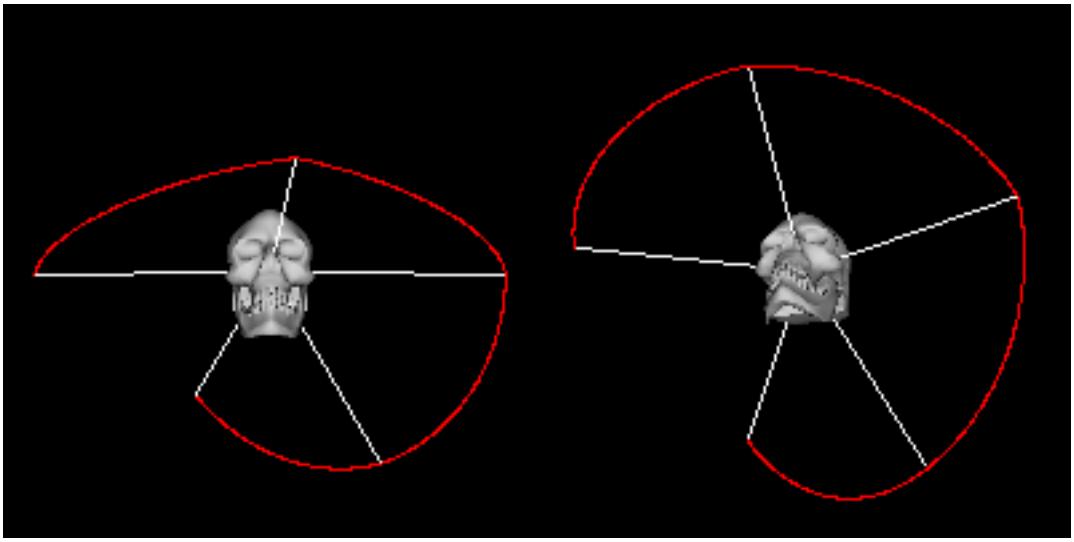


Figure 38 : Création de trajectoire sur sphère englobante (2)

Les figures 37 et 38 présentent des exemples de notre méthode de création de trajectoires sur une sphère englobante.

2. Création de trajectoire simple

Nous allons décrire une méthode d'exploration simple et très peu coûteuse en temps de calcul mais qui ne gère pas les obstacles. Nous nous basons donc uniquement sur l'ensemble de points de vue pour générer la trajectoire de manière incrémentale.

Nous partons du principe que l'ensemble des points de vue est correctement spécifié. Pour rappel, un point de vue est un vecteur de dimension neuf donc les trois premières indiquent la position, les trois suivantes le point vers lequel il regarde et enfin les trois dernières indiquent le haut de la caméra. Nous allons donc définir trois vecteurs par point de vue pour décrire la méthode. Nous les nommerons ainsi :

Point de vue :
- Position : Xp, Yp, Zp
- Orientation : Xo, Yo, Zo
- Haut : Xh, Yh, Zh

Afin de créer simplement les trajectoires, nous allons prendre les trois parties (position, orientation et haut) comme des entités distinctes.

a) Principe de base

Pour expliquer le fonctionnement de la création de ce type de trajectoire, nous allons prendre deux points distincts et consécutifs P_1 et P_2. Nous définissons un vecteur qui va servir de déplacement pour chacune des trois parties (d_p , d_o , d_h). Nous allons prendre un seul des trois vecteurs (position) pour l'exemple mais le fonctionnement est le même pour les deux autres.

La distance entre P_1 et P_2 peut se définir comme un vecteur D_p :

$$D_p (P_1 , P_2) = (X_{p2} – X_{p1} , Y_{p2} – Y_{p1} , Z_{p2} – Z_{p1}).$$

Nous pouvons maintenant définir un vecteur unitaire qui va être le point de référence de déplacement. Voici sa composante en X :

$$Ux_p(P_1, P_2) = \frac{X_{p2} - X_{p1}}{\left\| D_p(P_1, P_2) \right\|}$$

Ce vecteur est un vecteur unitaire de déplacement pour la trajectoire entre les points P_1 et P_2. Nous allons ensuite créer un vecteur plus petit qui va servir de « pas » entre deux positions de la trajectoire. Ce pas va être très variable et différent suivant la fluidité que l'on veut et la distance entre les deux points P_1 et P_2.

Le vecteur déplacement va donc être égal au vecteur unitaire multiplié par une constante α qui est définie comme la distance entre les deux points divisée par le nombre de déplacements voulus qui vont composer la trajectoire.

Pour la composante en x du vecteur cela donne :

$$dx_p(P_1, P_2) = \frac{X_{p2} - X_{p1}}{\left\| D_p(P_1, P_2) \right\|} \times \alpha$$

En appliquant cette formule aux autres composantes (en Y et en Z), nous obtenons notre vecteur de déplacement dédié à la position d_p (figure 39).

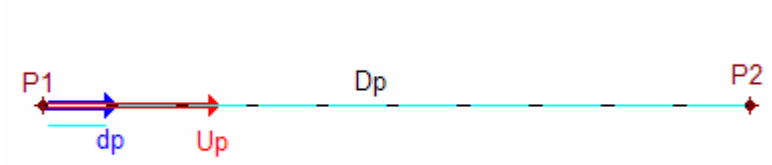


Figure 39 : Principe de création de trajectoire simple

Nous utilisons alors les mêmes calculs pour les deux autres vecteurs de déplacement d_o et d_h et nous obtenons un pas de même dimension qu'un point de vue (9 dimensions) qui va servir de « pas » pour la trajectoire entre deux points définis.

Si nous utilisons cette méthode telle quelle sur des scènes composées de multiples objets (notamment avec une ou plusieurs sphères englobantes), cette création n'est la plupart du temps pas pertinente. En effet une trajectoire va se retrouver à passer très près d'un objet et ce même si les deux points de vue consécutifs de la trajectoire appartiennent à la même sphère. Pour pallier ce problème, nous réalisons une projection du point de vue de façon à ce que sa distance au point regardé soit égale, soit au rayon de la sphère englobante si nous sommes sur une unique sphère, soit à une interpolation entre les rayons des deux sphères références (du point de vue précédent et suivant). Cet ajustement permet de plus d'arrondir la trajectoire (cf. figure 40).

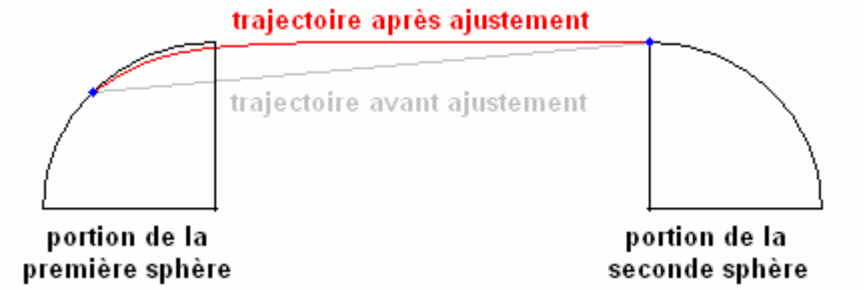


Figure 40 : Ajustement de la trajectoire

Nous voyons que le principe de base de cette méthode permet de gérer facilement la création de trajectoire entre deux points de vue. Cependant, la trajectoire est rectiligne et en cas de multiples points de vue, cette dernière va subir des changements brusques de direction et ne sera alors pas confortable pour l'utilisateur.

b) Evolution vers une trajectoire sans changement brusque

Maintenant que nous avons vu le principe de la méthode, nous allons voir comment modifier le principe de création afin d'avoir une trajectoire plus uniforme et sans à-coups.

Nous allons ajouter une notion d'accélération qui sera définie par deux valeurs : une variable « a » qui va être la référence d'accélération et un « pas d'accélération » que nous

allons noter « d_a ». Cette accélération va être appliquée aux différents vecteurs de déplacement des points de vue que nous avons vu précédemment. Une fois l'accélération égale au déplacement de base, celle-ci n'est plus augmentée dans la version standard. Nous utilisons ensuite le principe inverse à proximité du point d'arrivée.

Prenons l'exemple avec un dx_p que nous supposerons à 1 (chaque point de la trajectoire est à une unité de l'autre en X avec le principe de base). L'accélération est initialisée à 0 et le « pas » à une valeur choisie que nous prendrons ici à 0,2. Le point P1 est en X=0 et le point P2 en X=10.

Pour calculer la position suivante, nous prenons la position courante à laquelle nous ajoutons un déplacement. Ce déplacement est calculé comme le déplacement normal multiplié par l'accélération, puis nous incrémentons cette dernière du pas d'accélération.

Les graphiques 41 et 42 présentent les composantes en X des positions de la caméra en fonction du temps, et ce pour différentes accélérations ou contraintes.

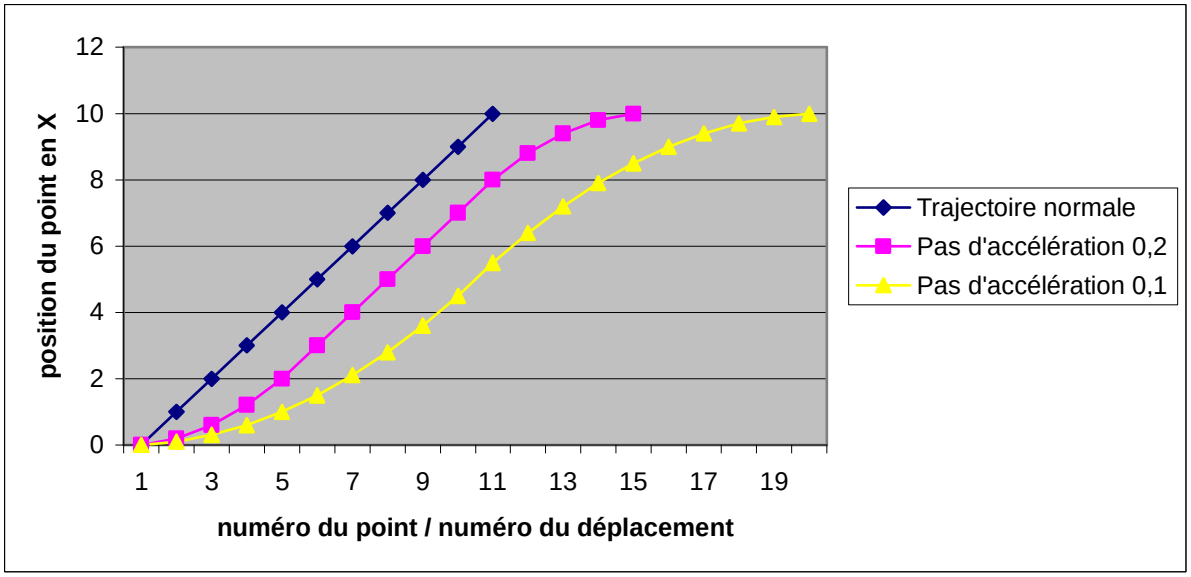


Figure 41 : Trajectoires et pas d'accélération

Nous voyons que le nombre de positions qui composent la trajectoire est inversement proportionnel au pas de la trajectoire (si nous mettons un seuil à l'accélération). Le graphique

ci-dessous montre le pas 0,2 lorsque nous ne bloquons pas le déplacement maximum, pour aller d'un point en X=0 à un point en X=20.

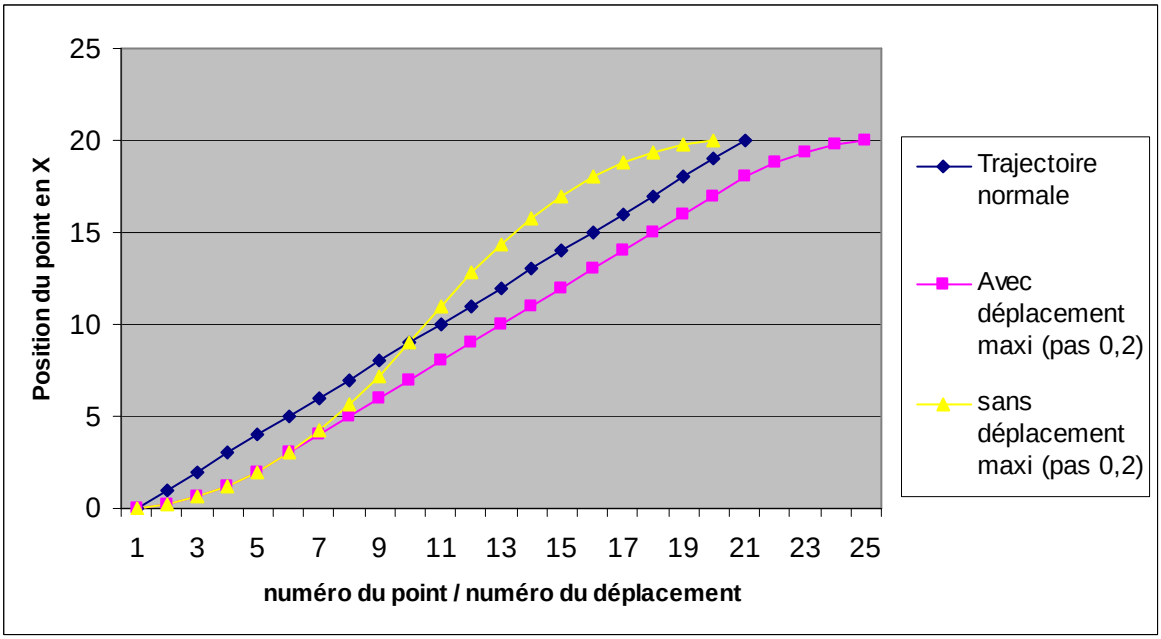


Figure 42 : Trajectoire et déplacement maximum

En laissant le point de vue une seconde avant de laisser la trajectoire redémarrer vers le point de vue suivant, nous obtenons une trajectoire fluide dont les changements de direction sont masqués par la diminution de la vitesse à l'approche des points de vue et l'arrêt temporaire sur ceux-ci.

Cette méthode de création de trajectoire est très peu coûteuse en temps de calcul et peut donc être utilisée lorsque l'on a peu de ressources à attribuer à ce type de calculs. De plus, si la trajectoire ne risque pas de rencontrer des obstacles (ensemble de points de vue au dessus de l'espace contenant les objets comme dans les mondes virtuels ouverts), le principal défaut de ce type de génération est évincé. Nous pouvons voir des exemples de ce type de création de trajectoire sur les figures 43, 44 et 45.

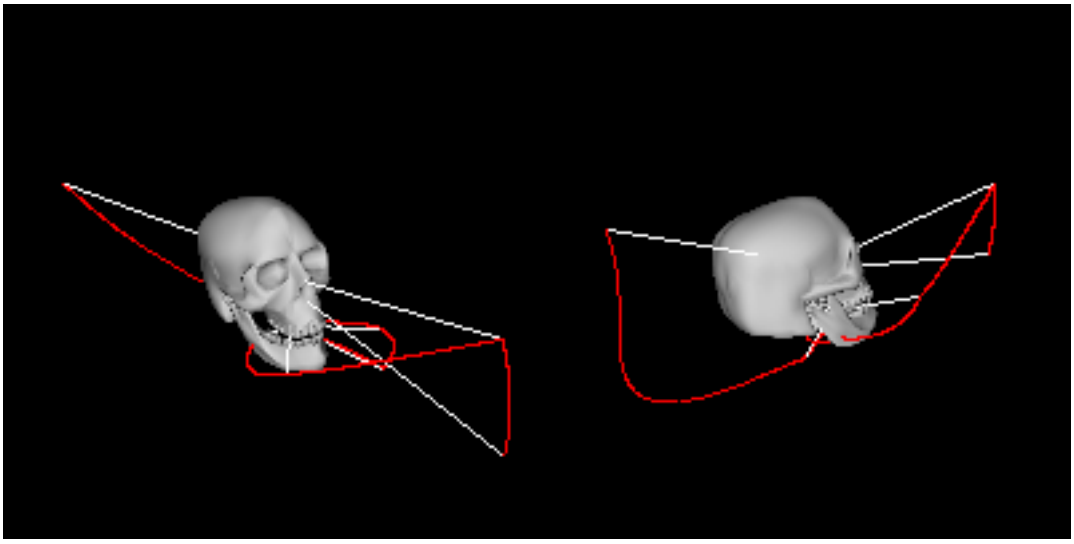


Figure 43 : Création de trajectoire sans gestion des obstacles(1)

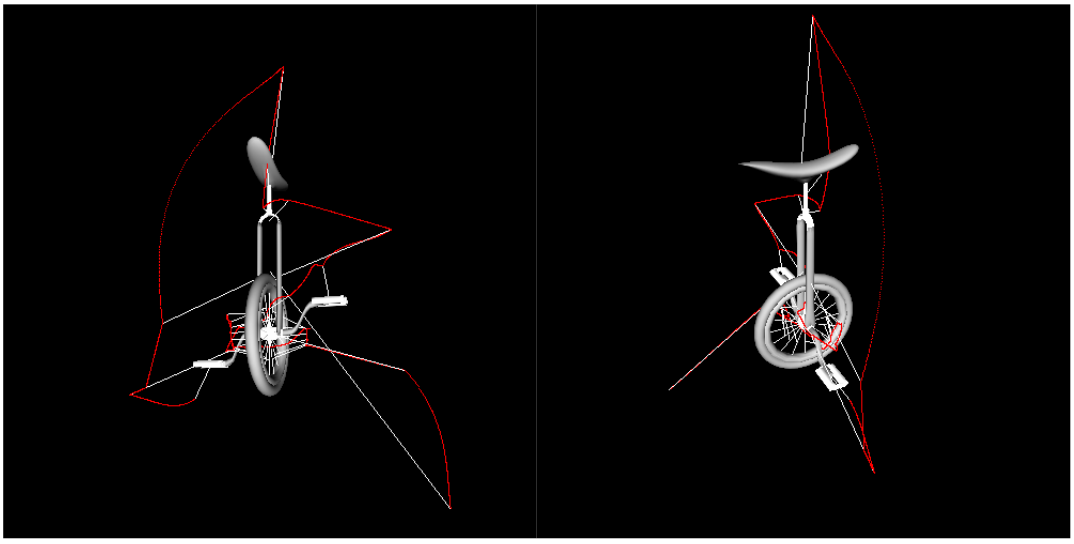


Figure 44 : Création de trajectoire sans gestion des obstacles (2)

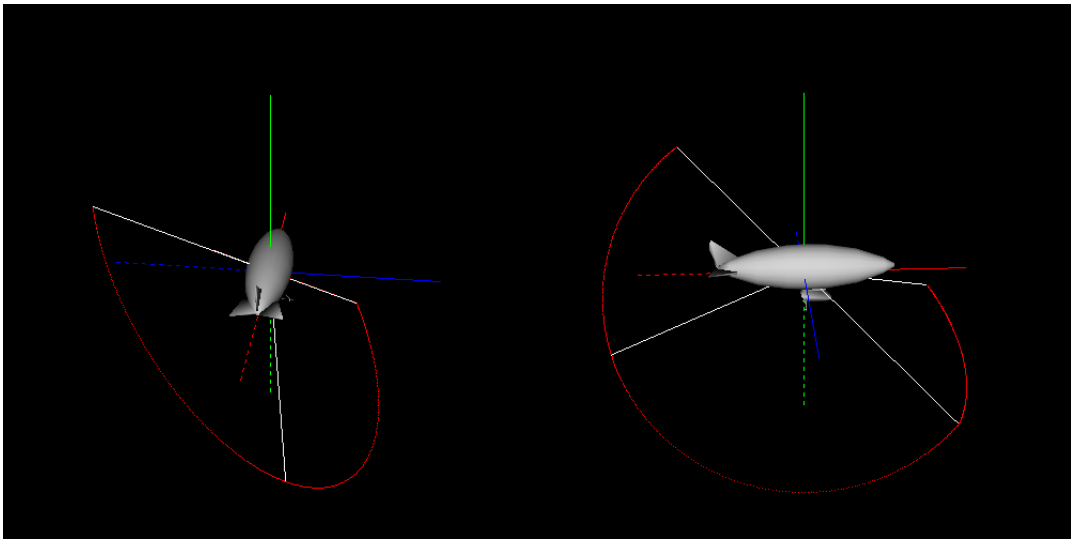


Figure 45 : Création de trajectoire sans gestion des obstacles (3)

C. Création de trajectoire avec gestion des obstacles

Les méthodes que nous venons de voir sont pratiques car les trajectoires sont créées « à la volée ». Il est cependant nécessaire d'avoir la possibilité de créer des trajectoires plus évoluées qui évitent les obstacles tels que les éléments de la scène.

Cette partie n'a été implémentée qu'en deux dimensions et seule la partie de calcul d'intersection entre la scène et le maillage a été développée en trois dimensions. Nos résultats de trajectoire n'apparaîtront donc que sur des rendus 2D mais nous expliciterons cependant les méthodes d'évolution vers une scène en trois dimensions.

1. Principe de fonctionnement

Nous allons discrétiser l'espace en une grille en trois dimensions. Nous obtenons un parallélépipède rectangle composé d'un ensemble de cubes dont les sommets vont représenter des positions possibles de passage de la trajectoire et les arrêtes les chemins éventuels entre ces points. Le pas de discrétisation est calculé en fonction de la taille de la scène et est égal à la plus grande dimension de la boîte englobante de la scène divisée par un entier représentant le nombre de mailles.

Une fois que ce maillage est mis en place, nous allons calculer quels sont les chemins (les arrêtes du maillage) qui sont intersectés par des polygones ou des objets de la scène. Ces liaisons sont alors détruites et nous ne conservons donc que les liaisons qui n'ont aucun contact avec la scène ou les objets du monde virtuel.

Nous éliminons alors l'ensemble des points du maillage qui ne sont plus accessibles depuis l'extérieur.

Nous obtenons alors l'ensemble des zones qui sont accessibles sans avoir à traverser les objets qui composent la scène.

Pour la compréhension, nous allons présenter l'ensemble de la méthode en deux dimensions en décomposant le processus étape par étape.

Nous créons donc notre grille. Chaque point est placé et des liaisons sont créées afin de générer le maillage.

Nous calculons quelles liaisons ont des intersections avec les faces qui constituent la scène et nous les supprimons.

Nous calculons alors la composante connexe en partant de l'extérieur et nous supprimons les points qui ne sont pas atteignables (et les liaisons auxquelles ils appartiennent).

Nous définissons alors les points de passage en faisant une corrélation entre les points de vue et les points de notre grille. Un point de passage est le point appartenant à la grille qui est le plus proche d'un des points de vue.

Nos points directeurs de trajectoire sont alors calculés et le squelette de la trajectoire est alors visible. Pour se faire, nous calculons le plus court chemin pour passer par l'ensemble des points de passage via les liaisons. En cas d'égalité, nous choisissons le chemin ayant le moins de changements de direction.

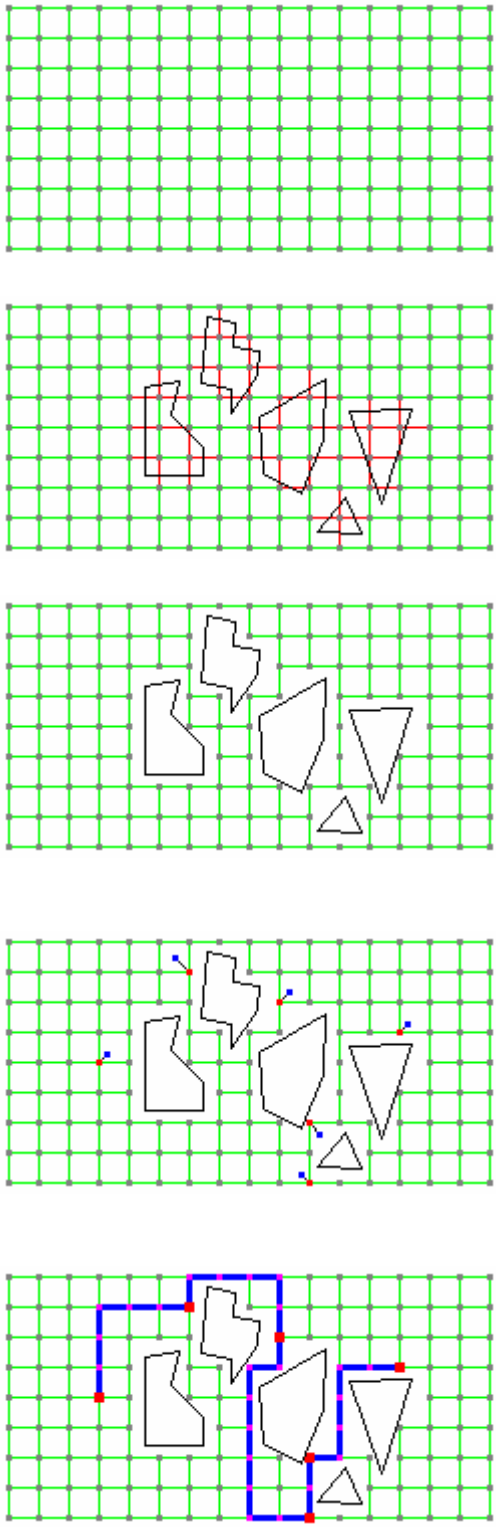


Figure 46 : Etapes de la création d'une trajectoire

Il suffit maintenant d'atténuer les angles de la trajectoire en évitant les changements brusques de direction.

Nous avons, en trois dimensions, vingt quatre possibilités de changements de directions qui sont symétriques deux à deux (figure 47). Cela nous donne donc douze variations différentes que nous calculons à l'aide d'un cercle de rayon égal à un dixième du maillage.

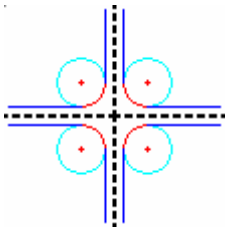


Figure 47 : Atténuation des angles de la trajectoire

Nous obtenons alors notre trajectoire finale (figure 48).

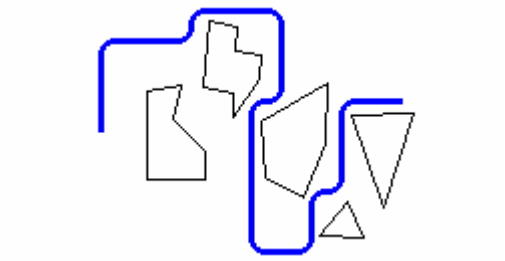


Figure 48 : Création de trajectoires complexes

2. Optimisations

Nous venons de voir le principe de la création de trajectoires complexes. Cependant, en l'état, cette méthode n'est pas optimisée et nous allons donc y apporter les modifications nécessaires.

a) Réduction du domaine

Il est inutile lors de la discrétisation de créer des mailles trop petites en découpant des zones où aucun obstacle n'est présent. En effet, non seulement le résultat ne sera pas pertinent

(nous savons immédiatement que dans une zone vide il n'y a pas de liaison intersectée) mais le temps de calcul sera énormément allongé.

Nous allons donc réduire la taille des zones discrétisées jusqu'à ne contenir que les boites englobantes des objets. Pour ce faire, nous n'appliquons le maillage qu'aux boites englobantes des objets ou des sous-objets.

Les autres liaisons seront considérées comme valides sans aucun calcul. Nous réduisons donc les calculs aux zones pertinentes (figure 49).

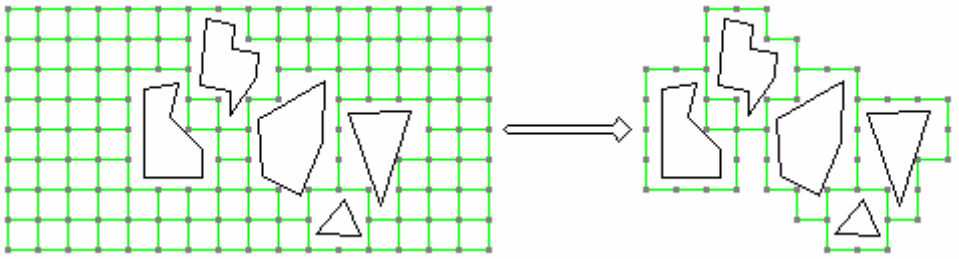
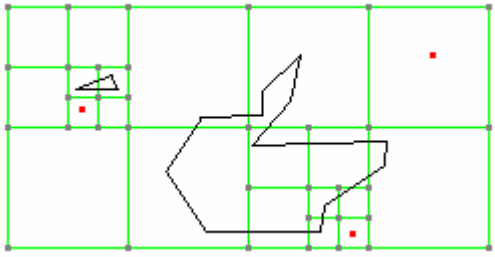


Figure 49 : Réduction du domaine aux boîtes englobantes

b) Découpe récursive de la zone de recherche

Un problème peut se présenter à nous lors du calcul des intersections. En effet, du fait que nous positionnons le point de vue sur un des points proches, rien ne nous garantit qu'un petit objet n'est pas intercalé. Tant qu'un élément est dans la même zone qu'un point de vue, nous découpons alors récursivement la zone.



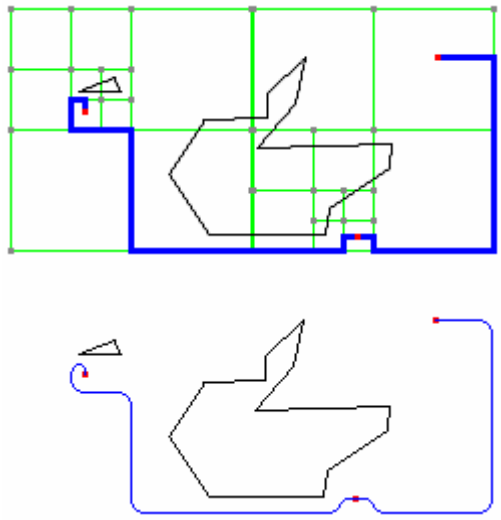


Figure 50 : Découpe récursive de la zone et trajectoire obtenue

La figure 50 présente la trajectoire obtenue suivant ce principe. Nous partons du point de vue le plus à gauche qui est le premier point de la trajectoire. La petite spirale est provoquée par le fait que la trajectoire cherche en premier lieu à rejoindre le bord le plus proche afin de coller au maillage. Une fois ceci fait, la trajectoire essaie de joindre par le chemin le plus court un des sommets du carré occupé par le point de vue suivant et ainsi de suite.

Ce principe nous garantit qu'aucun obstacle n'est présent dans la même zone qu'un des points de vue et nous pouvons du coup garder ces derniers tels quels sans avoir à les déplacer sur un des sommets de la zone.

De plus, cette propriété permet d'adoucir la trajectoire dans ces zones mais aussi de passer la trajectoire non plus sur les liaisons mais aussi à l'intérieur de la zone si le besoin s'en fait sentir (arrivée au niveau du point de vue).

Le temps de calcul, bien qu'un peu augmenté, peut être maintenu car lors de la première itération, nous pouvons garder certaines informations utiles, notamment quelles faces ont causé le découpage récursif. Il suffit alors de ne tester que celles-ci lors de l'itération suivante.

Nous pouvons voir sur les figures 51 et 52 un exemple de discrétisation de la zone en trois dimensions. En rouge nous voyons les points qui ne sont pas complets (un point complet

possède encore l’ensemble de ses six liaisons). En vert nous voyons les points accessibles et en blanc les points non accessibles.

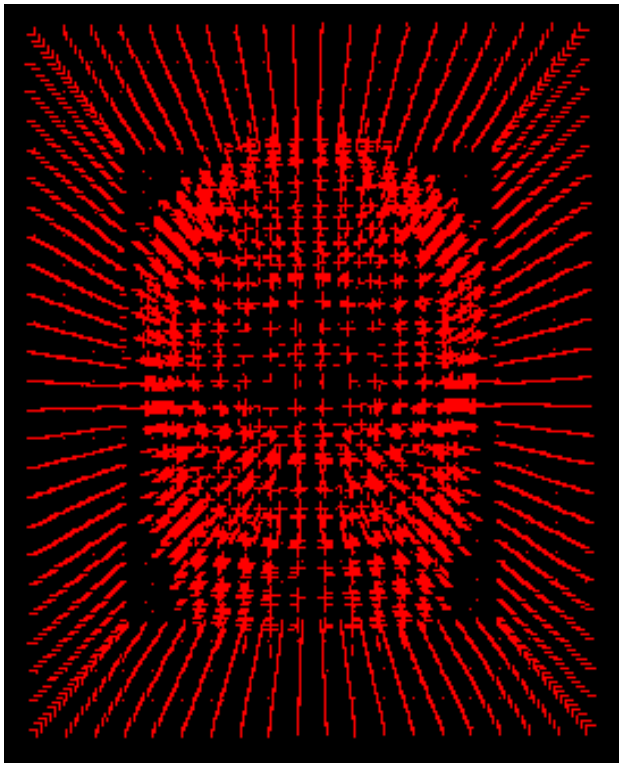


Figure 51 : Maillage et liaisons coupées

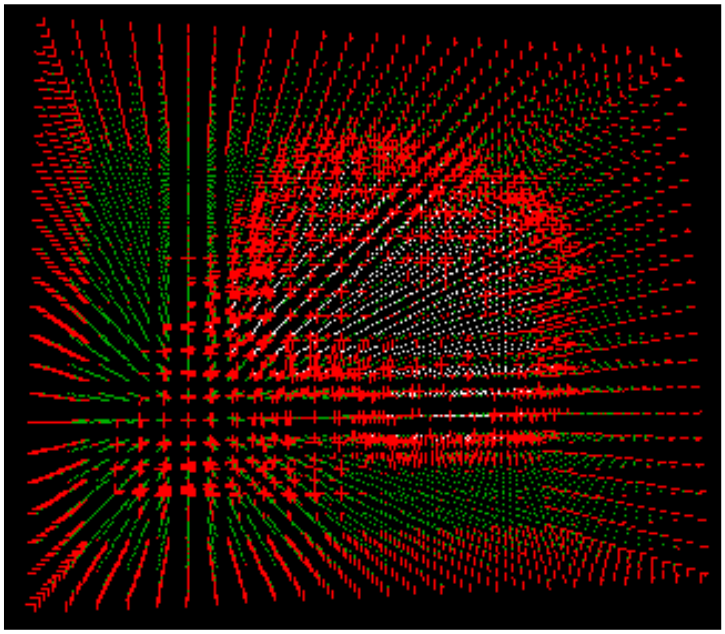


Figure 52 : Maillage et points accessibles ou bloqués

D. Conclusion

Nous venons de voir un certain nombre de propositions destinées à permettre la création de trajectoires dans un environnement en trois dimensions. Ces techniques s'appuient sur un ensemble de points de vue qui est utilisé comme squelette pour la future trajectoire.

Trois approches ont été présentées. La première présente une exploration externe de la scène avec des positions placées sur une sphère englobante de la scène, ne nécessitant, de fait, aucune gestion d'obstacles. La seconde étend les possibilités de la première en permettant de s'affranchir de l'extérieur de la scène et de s'approcher des différents objets qui la composent. La troisième approche avance une piste de génération de trajectoire en utilisant une grille (une zone discrétisée) et en calant la trajectoire sur celle-ci. Cette dernière approche, seule, permet de gérer les obstacles dans la création de trajectoire.

Conclusion

Conclusion et travaux futurs

Nous avons présenté, tout au long de ces travaux, des méthodes et des techniques pour l'exploration au sein des mondes virtuels et des jeux vidéo. L'ensemble des modèles est, de ce fait, susceptible d'être utilisé dans cette optique. De plus, afin de permettre une réutilisation et des modifications plus simples de ces travaux, nous avons voulu une approche modulaire de l'ensemble qui fait que chaque partie est indépendante et peut-être utilisée seule ou conjointement aux autres. Il est alors plus facile d'extraire et de modifier la partie qui nous intéresse et de la faire évoluer dans le sens recherché.

Dans un premier temps, nous avons présenté une méthode d'estimation de la qualité de points de vue et de calculs d'un ensemble pertinent de ceux-ci pour une scène donnée. Les critères permettant de juger un point de vue sont basés sur la notion de faces, d'objets, et d'éclairage et sont entièrement paramétrables par des systèmes de pondération des formules. Tout ceci permet un réel ajustement de l'évaluation à l'utilisateur qui l'utilise mais nécessite aussi différents tests qui peuvent s'avérer assez fastidieux afin de configurer correctement l'ensemble. Ce point pourrait être amélioré en utilisant des techniques existantes dans la modélisation déclarative. De plus, il est possible d'ajouter d'autres critères d'estimation, tels que ceux présentés par M. Sbert ou D. Sokolov, qui pourraient enrichir ceux déjà existants et qui permettraient de compléter la méthode générique. Nous avons, de plus, vu que nos méthodes se fondent sur un ensemble de départ dans lequel les points de vue seront sélectionnés. Cet ensemble est, soit choisi par l'utilisateur, soit défini par défaut, et peut, à lui seul, mettre en péril la pertinence des résultats. Peut-être pourrions nous définir un moyen de déterminer automatiquement quel genre d'ensemble de départ conviendrait le mieux à telle ou telle scène ?

Dans un second temps, et afin de pouvoir utiliser le critère des objets dans l'estimation d'un point de vue, nous avons présenté une méthode qui permet d'intégrer dans une scène la notion d'objets. Cette méthode, utilisable sur n'importe quelle scène, permet en fonction de l'existence ou non d'informations dans cette dernière de la découper en différents objets avec des résultats pertinents dans la plupart des cas. Le principal problème vient du fait que certaines techniques (méthodes de regroupement) ont besoin d'une intervention extérieure pour fonctionner (un critère d'arrêt). Peut-être est il possible, une fois encore, d'intégrer des

notions utilisées en modélisation déclarative afin de fournir un résultat sans intervention extérieure, basé sur une évaluation de la scène et de permettre ensuite à l'utilisateur d'orienter les méthodes vers les résultats souhaités.

Dans la dernière partie, nous avons exploré la création de trajectoires avec plusieurs méthodes qui, à partir d'un ensemble de points de vue, vont permettre de générer différents modèles de trajectoires en fonction des besoins de l'utilisateur. Ces méthodes couvrent un panel intéressant de possibilités et le système de création se veut assez simple à mettre en œuvre. Cependant, nos travaux se sont limités à la création de trajectoires en deux dimensions dans le cas de trajectoires en monde ouvert avec gestion de collision avec la scène. Bien que l'extension en dimension supérieure ne semble pas poser de problèmes théoriques, nous espérons pouvoir vérifier cette méthode dans les mois qui viennent. De plus nous pensons qu'il est possible d'étendre l système de maillage en incluant des poids différents dans les liens entre les mailles dépendant de l'évaluation des poins de vue proches. Cela permettrait de rentabiliser les déplacements en les faisant passer de points intermédiaires à points de vue à part entière.

Dans de futurs travaux, nous espérons pouvoir mettre en place le système de gestion déclarative que nous avons évoqué sur l'ensemble des composantes de nos travaux. Il est important de ne pas laisser le devoir à l'utilisateur de paramétrer les traitements sans avoir une connaissance minimale de la scène et des résultats éventuels qui en découlent.

Bibliographie

Bibliographie

[And04] C. Andujar, P. P. Vazquez et M. Fairen. Way-Finder : guided tours through complex walthrough models, Computer Graphics Forum (Eurographics 2004), 2004.

[BDP99] P. Barral, G. Dorme et D. Plemenos. Visual understanding of a scene by automatic movement of a camera. International Conference GraphiCon'99, Moscou (Russie), 26 Août -3 Septembre 1999.

[BDP00a] P. Barral, G. Dorme et D. Plemenos. Intelligent scene exploration with a camera. International Conference 3IA'2000. Limoges (France), 3-4 Mai 2000.

[BDP00b] P. Barral, G. Dorme et D. Plemenos. Scene understanding techniques using a virtual camera. Eurographics 2000, Interlagen (Suisse), 20-25 Août 2000. Short papers proceedings.

[BKL+96] J. Barraquand, L. E. Kavraki, J. C. Latombe, T. Y. Li, R. Motwani et P. Raghavan. A random sampling framework for path planning in large dimensional configuration spaces, Int. J. og Robotics Research, 1996.

[BL97] W. H. Bares et J. C. Lester. Realtime generation of customized 3d animated explanations for knowledge-based learning environments. Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97), pages 347-354. AAAI press, 1997.

[Bre65] J. E. Bresenham. Algorithm for computer control of a digital plotter. IBM Systems Journal, 4(1): 25-30, 1965.

[BTB99] V. Blanz, M. J. Tarr et H. H. Bulthoff. What object attributes determine canonical views ? Perception, 28 : 575-599, 1999.

[Bur78] J. Burg. Modern Spectral Analysis. Annual meeting international society exploration geophysics (1978), Childers D., (Ed.), IEEE, pages 34-41. 3, 5, 10.

[BZRL98] W. H. Bares, L. S. Zettlemoyer, D. W. Rodriguez et J. C. Lester. Task-sensitive cinematography interfaces for interactive 3d learning environments. Proceedings of the 1998 International Conference on Intelligent User Interfaces, pages 81-88. Tasks and Usage, 1998.

[Cal86] C. R. Calladine. Gaussian curvature and shell structures. The Mathematics of Surfaces, pages 179-196, Oxford (Angleterre), 1986. Clarendon Press.

[CDST95] B. Chazelle, D. P. Dobkin, N. Shouraboura et A. Tal. Strategies for polyhedral surface decomposition : an experimental study. SCG'95 : Proceedings of the eleventh annual symposium on computational geometry, pages 297-305, New York, USA, 1995. ACM Press.

[Cha91] B. Chazelle. Triangulating a simple polygon in linear time. Discrete Comput. Geom., 6(5) : 485-524, 1991.

[Che95] S. C. Chen. Quick Time VR : an image-based approach to virtual environment navigation. In SIGGRAPH'95 : Proceedings of the 22nd annual conference on computer graphics and interactive techniques, pages 29-38, New York, USA, 1995. ACM Press.

[Chr76] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

[Coh03] D. Cohen-Or, Y. Chrysanthou, C. Silva et F. Durand. A survey of visibility for walkthrough applications. IEEE Transactions on Visualization and Computer Graphics 2003.

[Col88] C. Colin. A system for exploring the universe of polyhedral shapes. Eurographics'88, Nice (France), Septembre 1988.

[Col90] C. Colin. Automatic computing of good views of a scene, MICAD'90, Paris (France), Février 1990.

[Cow94] C. K. Cowan. Automatic camera and light-source placement using CAD models. IEEE Workshop on Directions in Automated CAD-Based Vision, page 22, Maui (Hawaï), 2-3 Juin 1991.

[DON87] B. R. Donald. A search algorithm for motion planning with six degrees of freedom, Artificial Intelligence, 31 : 295-353, 1987.

[Dor00] G. Dorme. Etude et réalisation de techniques de prise de connaissance de scènes tridimensionnelles, Limoges (France), Juin 2001.

[DP2000] G. Dorme et D. Plemenos. Méthodes de prise de connaissance de scènes tridimensionnelles, journées AFIG, Grenoble (France), Novembre 2000.

[Dur00] F. Durand. A multidisciplinary survey of visibility. ACM Siggraph course notes Visibility, Problems, Techniques, and Applications 2000.

[Dur02] F. Durand, G. Drettakis et C. Puech. The 3D visibility complex. ACM Trans. Graph. 2002, 21, 176-206.

[FCL99] S. Fleishman, D. Cohen-Or et D. Lischinski. Automatic camera placement for image-based modelling. PG'99 : Proceedings of the 7th Pacific Conference on Computer Graphics and Applications, page 12, Washington, DC, USA, 1999. IEEE Computer Society.

[Fei98] U. Feige. A threshold of ln N for approximating set cover. J. ACM, 45(4) : 634-652, 1998.

[Feixas99] M. Feixas, E. Acebo, P. Bekaert et M. Sbert. An information theory framework for the analysis of scene complexity, Eurograpics'99.

[Feixas02] M. Feixas. An Information Theory Framework for the Study of the Complexity of Visibility and Radiosity in a Scene. PhD thesis, Technical University of Catalonia (Espagne), 2002.

[FGRT92] N. Foo, B. J. Garner, A. Rao et E. Tsui. Semantic distance in conceptual graphs, pages 149-154, 1992.

[Free93] W. T. Freeman. Exploiting the generic view assumption to estimate scene parameters. Proccedings of the 4th International Conference on Computer Vision, pages 347-3556, Berlin (Allemagne), 1993. IEEE, Ashington, DC.

[Free94] W. T. Freeman. The generic viewpoint assumption in a framework for visual perception, Nature, 368 (6471) : 542-545, 7 Avril 1994.

[GJ79] M. R. Garey et D. S. Johnson. Computers and Intractability – A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York, 1979.

[Gra02] J. Grasset. Techniques d'amélioration de la visualisation : graphes et boîtes. Thèse de doctorat, Limoges (France), 8 Juillet 2002.

[Gra05] J. Grasset et D. Plemenos. Visibility-based Simplification of Objects in 3D Scenes. Proceedings of WSCG 2005 short papers.

[GRMS01] B. Gooch, E. Reinhard, C. Mouldinf et P. Shirley. Artistic composition for image creation. Proceedings of the 12th Eurographics Workshop on Rendering Techniques, pages 83-88, Londres (Angleterre), 2001. Springer-Verlag.

[GS84] S. Gull et J. Skilling. Maximum entropy method in image processing. IEE proceeding 131, Part F (1984), IEE, pages 646-659. 3, 5, 10.

[Gum02] S. Gumhold. Maximum entropy light source placement. Visualization 2002 International Conference, Octobre 2002.
[Ham93] B. Hamann. Curvature approximation for triangulated surfaces. Pages 139-153, 1993.

[HM03] M. Halle et J. Meng. Lightkit : A lighting system for effective visualization. IEEE Visualization, 2003.

[Jau04] B. Jaubert. Exploration automatique différée de mondes virtuels. Mémoire de Master, Université de Limoges (France), Juillet 2004.

[JPP02] V. Jolivet, D. Plemenos et P. Poulingeas. Inverse direct lighting with a Monte Carlo method and declarative modelling. Lecture Notes in Computer Science, 2002.

[JTP06] B. Jaubert, K. Tamine et D. Plemenos. Techniques for off-line scene exploration using a virtual camera. International Conference 3IA'06, Limoges (France), Mai 2006.

[Kha86] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research, 5(1):90–98, 1986.

[Kha96] M Khatib. Sensor-based motion control for mobile robots. PhD thesis, LAASCNRS December, 1996, 1996.

[KK88] T. Kamada et S. Kawai. A simple method for computing general position in Graphics and Image Processing, 41 (1) : 43-56, 1988.

[KMH01] R. Kosara, S. Miksch et H. Hauser. Semantic deph of Field, Proc. IEEE Symp. Information visualization, pages 97-104, 2001.

[KPC93] J. K. Kawai, J. S. Painter et M. F. Cohen. Radioptimization-goal based rendering. Proc. Of SIGGRAPH-93 : Computer Graphics, pages 147-154, Anaheim, CA, 1993.

[KTMB02] K. W. khawaja, D. Tretter, A. A. Maciejewski et C. A. Bouman. Automated Visual Assembly Inspection, Expert Systems : The Technology of Knowledge Management and Decision Making for the 21st Century, C. T. Leondes, (ed.), Academic Press, Vol. 3, 202, pages 661-700.

[LBGC98] E. Languenou, F. Benhamou, F. Goualard et M. Christie. The virtual cameraman : an interval based approach, Proc. d' ECAI'98 Workshop on Constraint techniques for artistic applications. aout 1998.

[Lue95] D. Luebke et C. Georges. Portals and mirrors simple, fast evaluation of potentially visible sets. Proceedings of the 1995 symposium of Interactive 3D graphics, 1995, 105-ff.

[LVJ05] C. H. Lee, A. Varshney et D. W. Jacobs. Mesh saliency. ACM Trans. Graph., 24(3) : 659-666, 2005.

[MAB+97] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims et S. Shieber. Design galleries : A general approach to setting parameters for computer graphics and animation. International Conference SIGGRAPH97.

[MC00] E. Marchand et N. Courty. Image-based virtual camera motion strategies. S. Fels et P. Poulin, editors, Graphics Interface Conference, GI00, pages 69-76, Montréal, (Québec), Mai 2000. Morgan Kaufmann.

[Moun98] J. P. Mounier. The use of genetic algorithm for planning in the field of declarative modelling, 3IA'98 International Conference, Limoges (France), Avril 1998.

[Nir02] S. Nirenstein, E. Blake et J. Gain. Exact from-region visibility culling. Proceedings of the 13th Eurographics workshop on Rendering, 2002, pages 191-202.

[NRTT95] H. Noser, O. Renault, D. Thalmann et N. Magnenat Thalmann. Navigation for digital actors based on synthetic vision, memory, and learning. Computers and Graphics, 19(1) : 7-19, 1995.

[PB96] D. Plemenos et M. Benayada. Intelligent Display Techniques in Scene Modelling. New Techniques to Automatically Compute Good Views. International Conference GraphiCon'96, Saint Petersburg (Russie), 1-5 Juillet 1996.

[PF92] P. Poulin et A. Fournier. Lights from highlights and Shadows. Computer Graphics, 25(2) : 31-38, Mars 1992.

[PGJT05] D. Plemenos, J. Grasset, B. Jaubert et K. Tamine. Intelligent visibility-based 3D scene processing techniques for computer games. GraphiCon'05, Novosibirsk (Russie), Juin 2005.

[Ph75] B. T. Phong. Illumination for computer generated Pictures, Communication of the ACM, Volume 18, Numéro 6, pages 311-317, Juin 1975.

[Ple87] D. Plemenos. Techniques de raffinement sélectif pour la visualisation réaliste de scènes tridimensionnelles. Revue internationale de CAO et d'Infographie, volume 2, numéro 4, 1987.

[Ple91] D. Plemenos. Contribution à l'étude et au développemnt des techniques de modélisation, génération et visualisation de scènes. Le projet MultiFormes. Thèse d'Etat, Nantes (France), Novembre 1991.

[Ple95] D. Plemenos. Declarative modelling by hierarchical decomposition. The actual state of the MultiFormes project. GraphiCon'95, Saint Petersburg (Russie), Juillet 1995.

[Ple03] D. Plemenos. Exploring virtual worlds : Current techniques and future issues. International Conference GraphiCon'2003, Moscou (Russie), Septembre 2003.

[PPV01] P. P. Vazquez, M. Feixas, M. Sbert et W. Heidrich. Viewpoint Selection Using Viewpoint Entropy. Vision, Modelling and Visualization 2001, Stuttgart, (Allemagne), pages 273-280, 2001.

[PPV02] P. P. Vazquez, M. Feixas, M. Sbert et A. Llobet. Viewpoint Entropy : A New Tool for Obtaining Good Views for Molecules. VisSym'02 (Eurographics-IEEE TCVG Symposium on Visualization), Barcelone (Espagne), 2002.

[PPV03] P. P. Vazquez. PhD thesis. On the Selection of Good Views and its Application to Computer Graphics. Technical University of Catalonia, 2003.

[PPV03b] P. P. Vazquez et M. Sbert. Fast adaptative selection of best views. Lectures Notes in Computer Science, 2003 (Proc. of ICCSA'2003).

[PPV03c] P. P. Vazquez et M. Sbert. Perception-based illumination information measurement and light source placement. Lectures Notes in Computer Science, 2003 (Proc. of ICCSA'2003).

[PPV03d] P. P. Vazquez, M. Feixas, M. Sbert et H. Heidrich. Automatic View Selection Using Viewpoint Entropy and its Application to Image-Based Modelling. Computer Graphics Forum, Décembre 2003.

[PPV03e] P. P. Vazquez et M. Sbert. Automatic indoor scene exploration. International Conference on Artificial Intelligence and Computer graphics, 3IA'2003, Limoges (France), Mai 2003.

[PRJ97] P. Poulin, K. Ratib et M. Jacques. Sketching shadows and highlights to position lights. Proceedings of Computer Graphics International 97, pages 56-63. IEEE Computer Society, Juin 1997.

[PSF04] D. Plemenos, M. Sbert et M. Feixas. On viewpoint complexity of 3D scenes. International Conference GraphiCon'2004, Moscou (Russie), Septembre 2004.

[Rigau00] J. Rigau, M. Feixas et M. Sbert. Information Theory Point Measures in a Scene. IIIA-00-08-RR, Institut de Informatica i Aplicaciones, Universitad de Girona, Girone (Espagne), 2000.

[Rigau02a] J. Rigau, M. Feixas et M. Sbert. New Contrast Measures for Pixel Supersampling. Advances in Modelling, Animation and Rendering. Proceedings of GGI'02, Bradford (UK), pages 439-451, 2002. Springer-Verlag London Limited, Londres (Angleterre).

[Rigau02b] J. Rigau, M. Feixas et M. Sbert. Entropy-Based Adaptative Sampling. Graphics Interface 2003, Halifax (Canada), Juin 2003.

[Sbert02] M. Sbert, J. Rigau, F. Castro et P. P. Vazquez. Applications of Information Theory to Computer Graphics. Proceedings of the 5th International Conference on Computer Graphics and Artificial Intelligence, 3IA'2002, Limoges (France), pages 21-36, Mai 2002.

[SFR+02] M. Sbert, M. Feixas, J. Rigau, F. Castro et P. P. Vazquez. Applications of the information theory to computer graphics. International Conference 3IA'2002, Limoges (France), Mai 2002.

[Sho92] K. Shoemake. Arcball : a user interface for specifying three-dimensional orientation using a mouse. Proceedings of Graphics Interface'92, pages 151-156, 1992.

[Sok06] D. Sokolov. Exploration différée de mondes virtuels sur Internet, Thèse, Limoges (France), Décembre 2006.

[SP05] D. Sokolov et D. Plemenos. Viewpoint quality and scene understanding. In Mark Mudge, Nick Ryan, and Roberto Scopigno, editors, VAST 2005 : Eurographics Symposium Proceedings, pages 67-73, ISTI-CNR Pisa, Italie, Novembre 2005. Eurographics Association.

[SP08] D. Sokolov et D. Plemenos. Virtual world explorations by using topological and semantic knowledge. Visual Comput, 2008.

[SPFG05] M. Sbert, D. Plemenos, M. Feixas et F. Gonzalez. Viewpoint quality : Measures and applications. Computational Aesthetics in Graphics, Visualization and Imaging, 2005.

[SPT06a] D. Sokolov, D. Plemenos et K. Tamine. Viewpoint quality and global scene exploration strategies. International Conference on Computer Graphics Theory and Applications (GRAPP2006), Sétubal (Portugal), Février 2006.

[SPT06b] D. Sokolov, D. Plemenos et K. Tamine. Methods and data structures for virtual world exploration. Visual Comput, 2006.

[SR61] S. Sesku et M. B. Reed. Linear Graphs and Electrical Networks. Addison Wesley, Londres (Angleterre), 1961.

[SS02] S. L. Stoev et W. Straber. A case study on automatic camera placement and motion for visualizing historical data. VIS'02 : Proceedings of the conference on Visualizaton'02, pages 545-548, Washington, DC, USA, 2002. IEEE Computer Society.

[TBBSG07] Tabareau N, Bennequin D, Berthoz A, Slotine JJ, Girard B. (2007) **:** Geometry of the superior colliculus mapping and efficient oculomotor computation. *Biol Cybern* ., 97 (4) : 279-292.

[Tell91] S. J. Teller et C. H. Sequin. Visibility preprocessing for interactive walkthroughs. Proceedings of the 18th annual conference on Computer graphics and interactive techniques 1991, 61-70.

[TM04] M. Tory et T. Möller. Human Factors In Visualization Research , IEEE Transactions on Visualization and Computer Graphics, vol. 10, no. 1, Janvier-Février 2004, pages 72-84.

[Vaz03] P. P. Vazquez. On the selection of good views and its applications to computer graphics. PhD thesis, Barcelone (Espagne), Mai 2003.

[VFSH01] P. P. Vazquez, M. Feixas, M. Sbert et W. Heidrich. Viewpoint selection using viewpoint entropy. In VMV'01 : Proceedings of the Vision Modelling and Visualization Conference 2001, pages 272-280. Aka GmbH, 2001.

[VFSH02] P. P. Vazquez, M. Feixas, M. Sbert et W. Heidrich. Image-based modelling using viewpoint entropy. Computer Graphics International, pages 267-279, 2002.

[VFSI02] P. P. Vazquez, M. Feixas, M. Sbert et A. Llobet. Viewpoint entropy : a new tool for obtaining good views of molecules. VISSYM'02 : Proceedings of the symposium on Data Visualization 2002, pages 183-188, Aire-la-ville (Suisse), 2002. Eurographics Association.

[VS03] P. P. Vasquez and Mateu Sbert, Automatic indoor scene exporation. In International Conference 3IA'2003, Limoges (France), May 2003.

[WW97] D. Weinshall et M. Werman. On view likehood and stability. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(2) : 97-108, 1997.

[YZW03] G. Yunfang, P. Zhigeng et X. Weiwei. Vision-based path planning in intelligent virtual environment. International conference 3IA'03, pages 25-33, Limoges (France), Mai 2003.

[ZTS02] E. Zuckerberger, A. Tal et S. Shlafman. Polyhedral surface decomposition with applications. Computers and Graphics, 26(5) : 733-743, 2002.

[ZZLY02] J. Zhong, H. Zhu, J. Li et Y. Yu. Conceptual graph matching for semantic search. ICCS'02 : proceedings of the 10th International Conference on Conceptual Structures, pages 92-196, Londres (Angleterre), 2002. Springer-Verlag.