Chapitre 6

Bilan et perspectives

Dans ce chapitre nous passons d'abord en revue les algorithmes que nous avons affectivement programmés. Nous examinons ensuite les liens et les différences qui existent entre les algorithmes de calcul de graphe de visibilité de scènes polygonales et de scènes composées d'objets courbes.

Enfin, nous faisons le point sur l'utilisation du complexe de visibilité pour résoudre des problèmes de visibilité et proposons quelques directions de recherche pour améliorer la résolution de ces problèmes.

— *Et maintenant en route vers de nouvelles aventures !*
Le commissaire Bougret et son fidèle adjoint l'inspecteur Charolles.

6.1 Réalisations

Comme le chapitre 5 l'a notamment montré, nous ne nous sommes pas arrêtés à l'aspect théorique des problèmes, mais nous avons aussi étudié l'aspect pratique des algorithmes et nous les avons notamment programmés. Nous détaillons dans cette section quelques aspects de la programmation des algorithmes et de leur réutilisation dans d'autres programmes.

Les programmes ont été écrits en C++, ce qui a permis d'utiliser la programmation orientée objet pour écrire les programmes. La programmation orientée objet permet de regrouper les différents éléments des structures de données en classes, ce qui les rend plus facile à manipuler et à intégrer dans les programmes. De plus l'utilisation de l'héritage permet de mieux gérer des éléments ayant des propriétés de bases communes. Nous avons utilisé le langage C++ tel qu'il est décrit dans la (future) norme ANSI et nos programmes ont été compilés avec le compilateur GNU `g++`.

6.1.1 Programmation du complexe de visibilité

Nous avons tout d'abord programmé l'algorithme de balayage optimal du graphe de visibilité d'une scène polygonale. Nous avons détaillé les structures de données utilisées pour représenter un arbre d'horizon dans le chapitre 2, lors de l'étude de la complexité de l'algorithme.

Pour profiter des relations de symétrie entre les deux arbres d'horizon et éviter de recopier inutilement des fonctions similaires, nous avons représenté l'arbre d'horizon inférieur par l'arbre d'horizon supérieur de la scène pivotée de 180°. Les deux arbres utilisent les mêmes fonctions définies dans une classe de base et les quelques fonctions différentes utilisées lors de l'initialisation ont été définies dans les deux classes dérivées correspondant aux deux types d'arbres. La figure 6.1 montre les deux arbres d'horizon initiaux d'une scène et fait apparaître ces liens entre les deux arbres.

Pour que le programme de balayage soit facilement réutilisable par un utilisateur, nous avons mis à la disposition de ce dernier des fonctions de rappel (callback). Lorsque le programme passe une arête e du graphe de visibilité, il appelle la fonction `passEdge(e)` qui peut être redéfinie par l'utilisateur. L'utilisateur adapte alors cette fonction selon ses besoins ; elle peut servir à construire effectivement le graphe de visibilité, ou simplement servir à afficher le graphe comme dans la figure 6.2 lorsqu'elle est redéfinie comme suit par exemple :

```
window w;
void passEdge(const edge* e)
{
  w.MoveTo(e->coord()->x(),e->coord()->y());
  w.LineTo(e->vis()->coord()->x(),e->vis()->coord()->y());
}
```

Nous avons ensuite utilisé le programme de balayage de graphe pour programmer la construction du complexe de visibilité, comme indiqué à la fin du chapitre 2. Les structures de données utilisées par le complexe de visibilité sont simples : chaque élément du complexe – sommet, arête et face – possède un pointeur vers les éléments

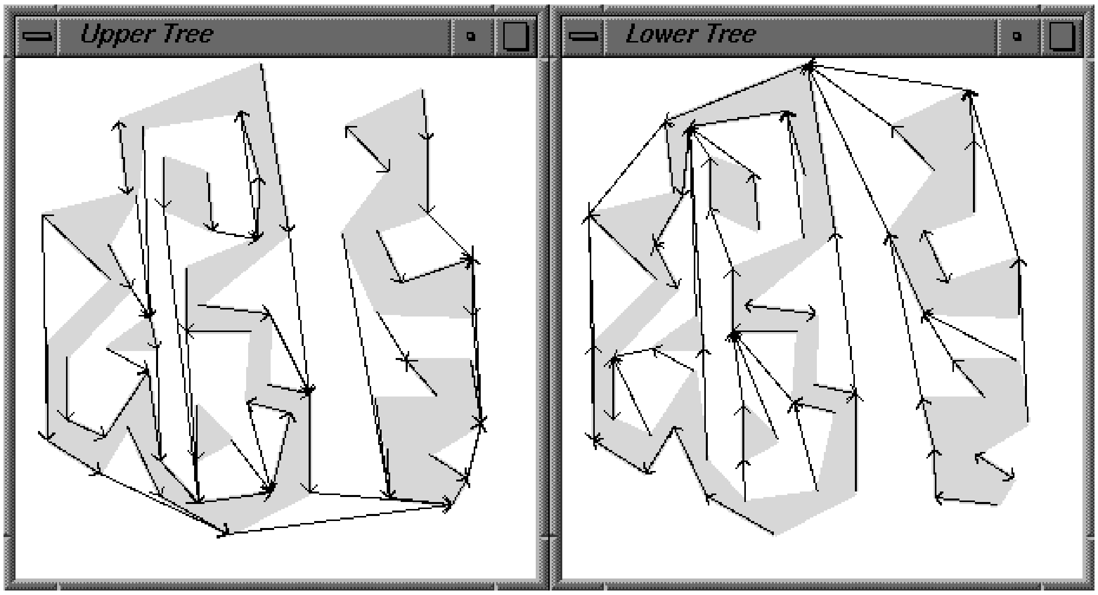

FIG. 6.1 – *Programme de calcul de graphe de visibilité : arbres d'horizon initiaux d'une scène.*

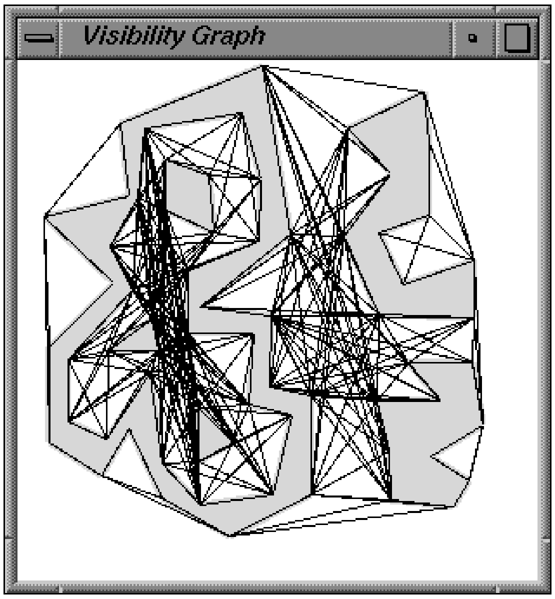

FIG. 6.2 – *Programme de calcul de graphe de visibilité : affichage du graphe.*

qui lui sont incidents : un sommet possède quatre pointeurs vers ses arêtes incidentes, une arête possède deux pointeurs vers ses sommets extrémités et des pointeurs vers ses faces incidentes, et une face possède deux pointeurs vers ses sommets extrêmes. Les six différents types d'arêtes sont dérivés à partir d'une classe de base qui fournit l'interface commune à toutes les arêtes.

Pour que le complexe puisse être utilisé ensuite dans des calculs de visibilité, il doit permettre l'accès à ses éléments. À cet effet, notre structure de données permet d'accéder aux deux faces semi-infinies du complexe et permet d'accéder pour chaque point de la scène à l'arête de la coupe initiale du complexe associée à ce point. Ces points d'accès permettent ensuite de faire des calculs de vue par exemple.

Pour les utilisateurs qui font des calculs globaux de visibilité et qui ont besoin de parcourir tous les éléments du complexe, nous avons programmé un itérateur qui permet d'effectuer un balayage topologique du complexe déjà construit : cet itérateur permet d'accéder successivement à tous les sommets du complexe de façon cohérente.

Bien que le complexe calculé sur un intervalle de directions de longueur π contienne toutes les informations nécessaires pour faire les calculs de visibilité, il n'est pas très pratique à utiliser : ses éléments sont coupés en deux aux bornes de l'intervalle. Nous avons alors calculé le complexe sur un intervalle de longueur 2π et « recollé » les morceaux coupés aux bornes de l'intervalle. Ainsi les calculs se font de façon continue dans $\mathbb{R}/2\pi$ sans problèmes de continuité aux limites.

6.1.2 Programmation des algorithmes de calculs de visibilité

La première utilisation du complexe que nous avons programmée est le calcul d'une vue autour d'un point. Pour rendre l'utilisation du programme flexible, nous avons défini la classe `view_iterator` qui calcule la vue selon le modèle d'un itérateur qui permet d'accéder successivement aux arêtes du complexe coupées par la courbe duale du point de vue. L'utilisateur décide ensuite de ce qu'il veut faire des arêtes fournies par l'itérateur.

Pour afficher par exemple la vue comme sur la figure 6.3, il suffit d'écrire le morceau de programme suivant :

```
window w;
complex c(nv,np,tabp);                //construction complexe
..........
point p(x,y);                         //point de vue
view_iterator it(p,c);                //iterateur de calcul de vue
while (it)
{
  w.MoveTo(x,y);
  w.LineTo((*it).x(),(*it).y());
  ++it;
}
```

Nous avons ensuite programmé les deux algorithmes de maintien dynamique d'une vue autour d'un point. Les programmes réutilisent le programme de calcul de vue pour calculer la liste initiale des arêtes du complexe coupées par la courbe duale du point de vue ; ils maintiennent ensuite cette liste quand le point de vue se déplace. L'algorithme de maintien lors de petits déplacements et l'algorithme de maintien le long d'une trajectoire ont été respectivement mis en œuvre par les classes `dynamic_view` et `traject_view`. Ces deux classes ont une fonction membre `moveTo(x,y)` qui permet de

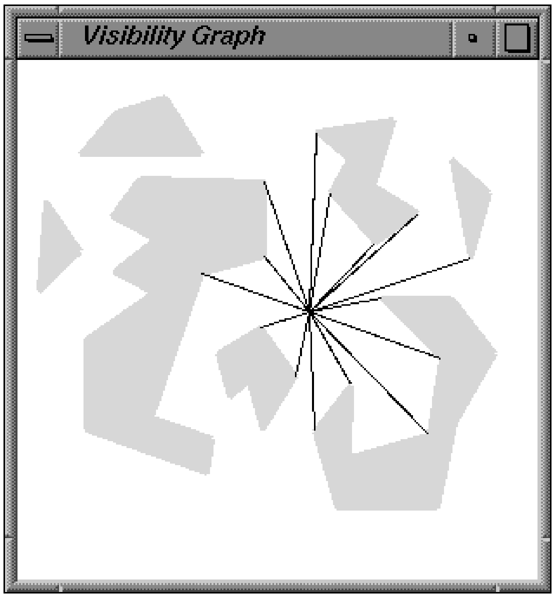

FIG. 6.3 – *Programme de calcul de vue : affichage de la vue.*

déplacer le point de vue.

Elles fournissent à l'utilisateur des itérateurs pour parcourir à un instant donné les sommets de la vue courante. Comme le but de l'algorithme de maintien le long d'une trajectoire est de signaler les changements de visibilité le long de cette trajectoire, la méthode `moveTo(x,y)` de la classe `traject_view` appelle une fonction de rappel à chaque changement de visibilité élémentaire. Cette fonction de rappel peut être redéfinie par l'utilisateur selon ses besoins.

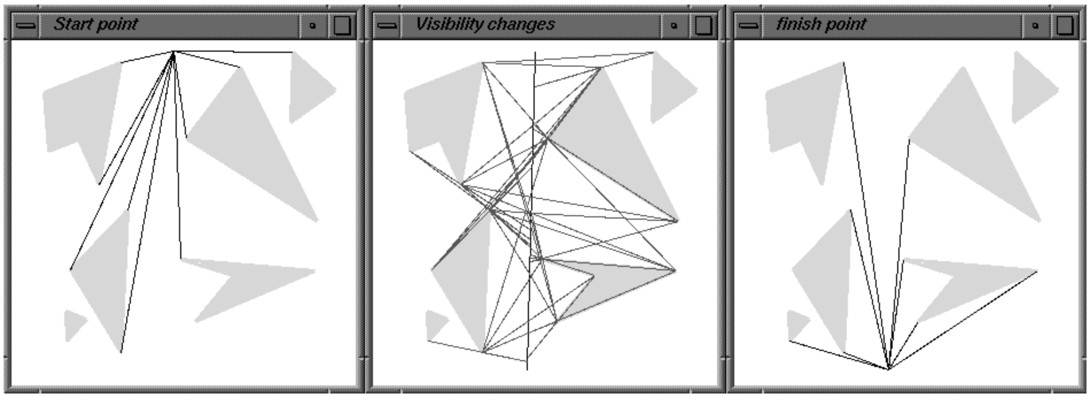

FIG. 6.4 – *Programme de maintien dynamique de vue le long d'une trajectoire.*

La figure 6.4 illustre l'exécution de l'algorithme de maintien de vue le long d'une trajectoire. Dans la première (resp. dernière) fenêtre, nous avons affiché la vue de départ (resp. d'arrivée) du point de vue, et dans la fenêtre du milieu, nous avons indiqué les limites de changement de visibilité traversées par le point de vue lorsqu'il se déplace en ligne droite depuis le point de départ jusqu'au point d'arrivée.

6.1.3 Utilisation du complexe de visibilité dans le calcul de radiosité

Les programmes que nous avons développés, notamment le programme de calcul du complexe de visibilité, ont été utilisés en synthèse d'images pour calculer l'éclairage d'une scène par la méthode de radiosité.

La méthode de radiosité est une méthode de calcul d'illumination globale de la scène. Dans cette méthode, tous les éléments d'une scène sont considérés comme émettant de l'énergie lumineuse : ils peuvent émettre une énergie lumineuse propre (source de lumière) et surtout ils renvoient une partie de l'énergie lumineuse qu'ils reçoivent des autres éléments de la scène.

Pour effectuer les calculs, la scène est divisée en éléments simples P_i. L'hypothèse est faite que l'énergie lumineuse totale émise en chaque point de la surface d'un élément est constante. Ce flux d'énergie – appelé radiosité – est noté B_i. L'énergie propre émise par un élément est notée E_i (figure 6.5a). Un élément est aussi considéré comme étant une surface diffuse, renvoyant la lumière de façon uniforme dans l'espace. Les éléments P_i sont alors caractérisés par le coefficient ρ_i de réflectance, coefficient indiquant la fraction de l'énergie renvoyée par la surface (figure 6.5b).

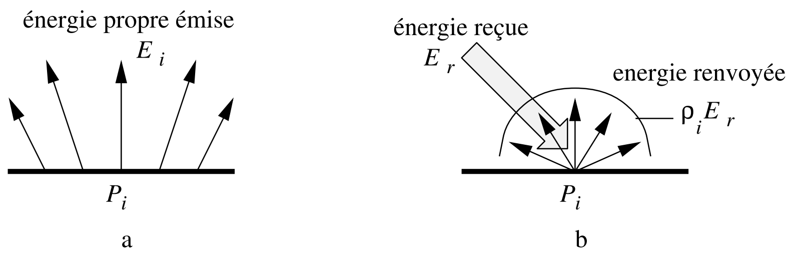

FIG. 6.5 – *Calcul de radiosité.* **a.** *Énergie propre émise par un élément.* **b.** *Énergie renvoyée par une surface diffuse.*

La méthode de radiosité cherche à calculer les valeurs de radiosité B_i de chaque élément. Ces valeurs vérifient l'équation de radiosité

$$B_i = E_i + \rho_i \sum_j F_{ij} B_j,$$

où F_{ij} est le facteur de forme entre les éléments P_j et P_j défini par

$$F_{ij} = \frac{1}{A_i} \int_{x \in P_i} \int_{y \in P_j} \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y)\, dy\, dx \ ,$$

où A_i désigne la surface de l'élément P_i, r est la distance $r = \|x, y\|$, θ_i (resp. θ_j) désigne l'angle entre le vecteur $\overrightarrow{xy}$ et la normale à P_i (resp. P_j) (figure 6.6a) et $V(x,y)$ est la fonction de visibilité entre les points : $V(x,y) = 1$ si x et y sont mutuellement visibles, $V(x,y) = 0$ si x et y sont cachés l'un de l'autre (figure 6.6b).

Le facteur de forme est une quantité purement géométrique et représente en quelque sorte l'ensemble des rayons allant de P_i à P_j. Une fois les facteurs de forme calculés,

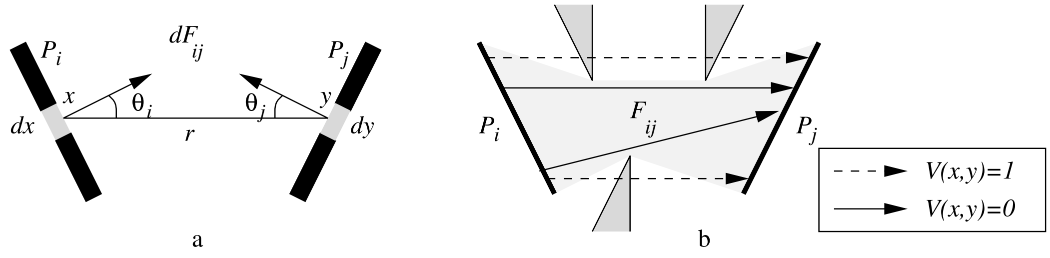

FIG. 6.6 – *Calcul de facteur de forme.* **a.** *Facteur de forme élémentaire entre deux éléments de surface sur* P_i *et* P_j. **b.** *Fonction de visibilité.*

il ne reste plus qu'à résoudre un système d'équations linéaires pour obtenir les valeurs de radiosité.

Le coût le plus important des algorithmes de calcul de radiosité est celui du calcul des facteurs de forme, ou plus précisément du calcul de la fonction de visibilité lors du calcul des facteurs de forme : il faut tester si les éléments d'intégration sont mutuellement visibles, et ce test demande des calculs de visibilité coûteux. Or le complexe de visibilité contient déjà ces calculs de visibilité. En particulier, les rayons allant de P_i à P_j à prendre en compte dans le calcul du facteur de forme F_{ij} sont ceux de la face du complexe d'étiquette (P_i, P_j). Le calcul des facteurs de forme se fait alors en considérant les faces du complexe de visibilité. Orti [ORDP96] a étudié comment calculer efficacement les facteurs de forme en utilisant le complexe de visibilité et a réutilisé nos programmes pour développer un environnement de calcul de radiosité de scènes polygonales 2D.

6.2 Polygones et objets courbes

Pocchiola et Vegter ont créé et étudié le complexe de visibilité dans le cadre de scènes composées d'objets courbes convexes. Nous examinons ici les rapports et les différences qui existent entre polygones et objets courbes dans les calculs de visibilité, notamment au niveau du calcul du graphe de visibilité.

Un polygone convexe peut être considéré comme un objet courbe convexe un peu particulier : l'objet n'est pas défini par une courbe mais par plusieurs morceaux de courbe. Dans certains travaux par exemple, les polygones sont considérés comme des approximations de courbes. En effet, les calculs de visibilité à effectuer sur les objets courbes, notamment les calculs de tangente, peuvent être délicats à mener en pratique : il faut résoudre des équations algébriques de degré élevé. Les calculs sont alors effectués sur des suites de polygones approximant de plus en plus les objets courbes : le calcul d'une tangente à un polygone convexe prend au plus un temps logarithmique en fonction du nombre de sommets du polygone.

Le parti que nous avons pris dans nos travaux est donc différent de ces approches. L'objet de base n'est pas le polygone mais le côté de polygone. Cela permet si nécessaire d'associer des propriétés différentes (couleur par exemple) aux côtés d'un même polygone. Un polygone ne peut plus alors être considéré comme *un* objet courbe. Par exemple, étant donné une direction θ il y a deux tangentes à un objet courbe, tangentes qui délimitent deux changements de visibilité (figure 6.7a). En revanche, pour un polygone à n sommets il y a n droites délimitant un changement de visibilité : ce sont les droites passant par les sommets du polygone (figure 6.7b).

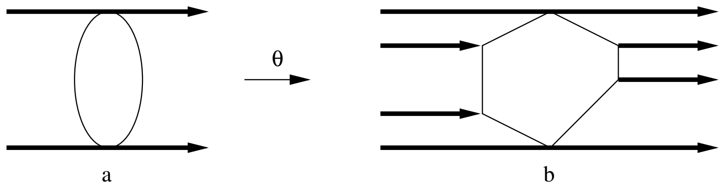

FIG. 6.7 – *Changements de visibilité selon une direction θ.* **a.** *Objet courbe : deux tangentes.* **b.** *Polygone : n droites passant par ses n sommets.*

Étant donné ces différences entre objets courbes et polygones, peut-on quand même réutiliser des techniques développées dans un cadre pour les appliquer dans l'autre cadre ? Nous discutons ici de la transposition des arbres d'horizon dans le cadre d'objets courbes et de la transposition des pseudo-triangulations dans le cadre de polygones.

De la même façon que nous associons à chaque segment de droite $[p, q]$ deux courbes duales p^* et q^* dans l'espace dual (figure 6.8a), il est possible d'associer deux courbes duales à un objet courbe convexe.

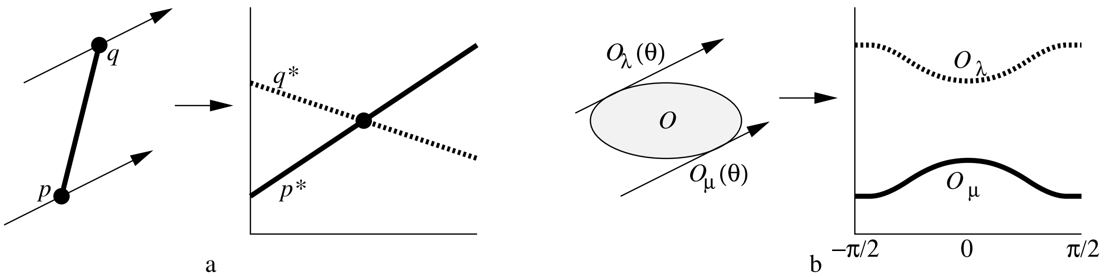

FIG. 6.8 – *Courbes duales.* **a.** *D'un segment de droite.* **b.** *D'un objet courbe convexe.*

Les courbes duales p^* et q^* représentent dans la scène l'ensemble des droites passant par p et q, c'est-à-dire l'ensemble des tangentes au segment $[p, q]$. Pour un objet courbe convexe O, il existe pour chaque direction θ deux tangentes $O_\mu(\theta)$ et $O_\lambda(\theta)$ à O de direction θ. La tangente $O_\mu(\theta)$ (resp. $O_\lambda(\theta)$) est telle que l'objet est au-dessus (resp. en dessous) de la tangente et est dite de type μ (resp. λ). Dans l'espace dual, quand θ varie les deux points duaux associés aux deux tangentes $O_\mu(\theta)$ et $O_\lambda(\theta)$ décrivent deux courbes duales O_μ et O_λ (figure 6.8b).

La dualité associe alors à une scène d'objets courbes un arrangement dual et Pocchiola [Poc90] s'en est servi par exemple pour faire des calculs de visibilité de façon similaire à ceux effectués avec l'arrangement dual d'une scène de polygones. Les courbes duales d'objets courbes convexes sont aussi les courbes support des arêtes du complexe de visibilité de la scène d'objets courbes.

Malgré ces similitudes, les arrangements duaux d'objets courbes ont une nature différente des arrangements duaux de polygones. L'arrangement dual d'une scène de polygones (pour un intervalle de directions de longueur π) est un arrangement de droites, ou de pseudo-droites, c'est-à-dire de courbes ayant deux à deux une seule intersection. L'arrangement dual d'une scène d'objets courbes convexes (pour un intervalle de directions de longueur 2π) est en revanche un arrangement de pseudo-cercles, c'est-à-dire de courbes ayant deux à deux deux points d'intersection, qui ne peut être ramené à un arrangement de pseudo-droites : l'intervalle séparant les deux points d'intersections de deux courbes est de longueur quelconque.

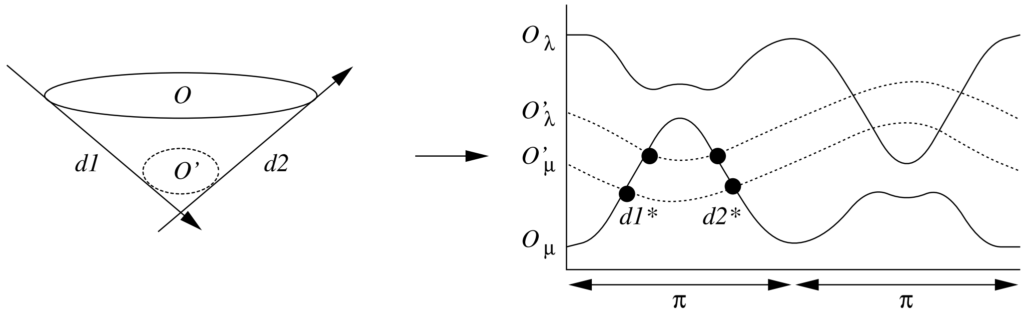

FIG. 6.9 – *Arrangement de courbes duales d'objets convexes.*

La figure 6.9 montre par exemple l'arrangement d'une scène de deux objets O et O' où les courbes O_μ et O'_μ ont leur deux points d'intersection dans l'intervalle $[-\pi/4, \pi/4]$.

Cette différence de nature entre les deux types d'arrangements modifie les propriétés des arbres d'horizon lorsque nous les transposons dans le cadre d'objets courbes convexes. Pour construire l'arbre d'horizon supérieur à partir d'une direction θ_0, nous étendons pour chaque objet O sa tangente $O_\mu(\theta_0)$ (resp. $O_\lambda(\theta_0)$) et la faisons tourner autour de O (dans le sens trigonométrique direct) jusqu'à ce qu'elle devienne tangente à un autre objet situé devant O. Chaque arête de l'arbre d'horizon est donc une bitangente, et la bitangente parent d'une bitangente de type O_μ (resp. O_λ) arrivant en O est la bitangente de type O_μ (resp. O_λ) partant de O.

Le graphe obtenu n'est malheureusement plus un arbre. La figure 6.10 montre l'arbre d'horizon supérieur d'une petite scène d'objets et la représentation combinatoire (figure 6.10b) montre que l'« arbre » d'horizon n'est plus connexe et comporte des cycles.

Les propriétés géométriques sont elles aussi modifiées. Par exemple, en voulant passer la bitangente $\lambda_1\mu_2$ de l'arbre d'horizon de la figure 6.10(a), nous voyons que les arêtes de l'enveloppe convexe inférieure des objets 3,4 et 5 ne font pas toutes partie de l'arbre d'horizon.

Des problèmes interviennent aussi à cause d'une autre différence entre les deux courbes duales d'un segment de droite et celles d'un objet courbe : la figure 6.8 nous montre que les courbes duales p^* et q^* des extrémités du segment se croisent alors que les courbes duales O_μ et O_λ de l'objet courbe restent séparées.

Les problèmes apparaissent quand nous devons passer une arête qui fait partie de l'enveloppe convexe de la scène. Dans le cas de polygones, l'arête suivante de cette arête

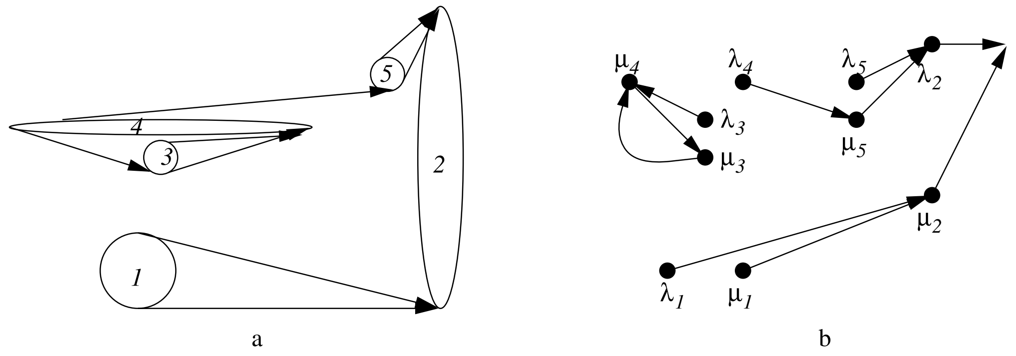

FIG. 6.10 – *« Arbre d'horizon » pour une scène d'objets courbes.* **a.** *Représentation géométrique.* **b.** *Représentation combinatoire.*

(p, q) est l'arête précédente de l'enveloppe convexe partant de p (figure 6.11a), alors que dans le cadre d'objets courbes, l'arête suivante ne fait pas partie de l'enveloppe convexe (figure 6.11b) : dans le premier cas, l'arête « pivote » autour de p alors que dans le deuxième cas, l'arête « glisse » autour de O.

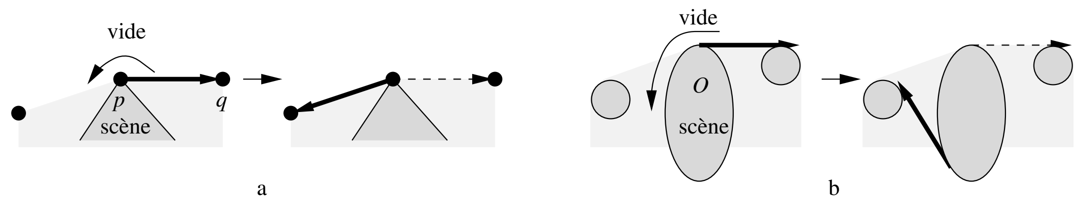

FIG. 6.11 – *Passage d'une arête sur l'enveloppe convexe.* **a.** *Scène de polygones.* **b.** *Scène d'objets courbes convexes.*

Pour transformer l'arrangement dual d'une scène d'objets courbes en arrangement de pseudo-droites, Pocchiola et Vegter [PV94] considèrent de nouveaux objets, les pseudo-triangles :

un pseudo-triangle est un ensemble de $\mathbb{R}^2$ borné et simplement connexe tel que :

- *sa frontière est composée d'une suite de trois courbes convexes tangentes en leurs extrémités ;*
- *l'ensemble est contenu dans le triangle formé par les trois extrémités de ces courbes.*

À chaque pseudo-triangle est associé une courbe duale, et l'ensemble des courbes duales des pseudo-triangles forme alors un arrangement de pseudo-droites facilement balayable topologiquement. Le lien entre les pseudo-triangles et la scène est donné par le concept de pseudo-triangulation :

une pseudo-triangulation d'un ensemble d'objets courbes convexes est un ensemble maximal de bitangentes libres non sécantes. (figure 6.12)

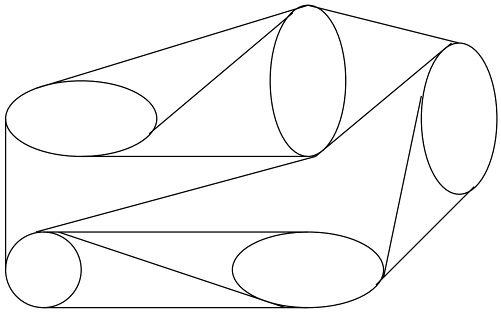

FIG. 6.12 – *Pseudo-triangulation d'une scène d'objets convexes.*

Les pseudo-triangulations découpent l'espace libre de la scène en pseudo-triangles et le balayage topologique de l'arrangement dual d'une pseudo-triangulation permet de balayer toutes les bitangentes de la scène.

Pocchiola et Vegter [PV96a] utilisent ensuite les pseudo-triangulations pour balayer le complexe, et donc le graphe, de visibilité de la scène en temps optimal. Deux pseudo-triangles adjacents forment un pseudo-quadrangle dont la bitangente commune est une diagonale du pseudo-quadrangle ; en basculant la bitangente dans l'autre diagonale du pseudo-quadrangle on obtient deux autres pseudo-triangles (figure 6.13). Si les deux pseudo-triangles considérés font partie d'une pseudo-triangulation, on obtient alors après la bascule de la bitangente une nouvelle pseudo-triangulation.

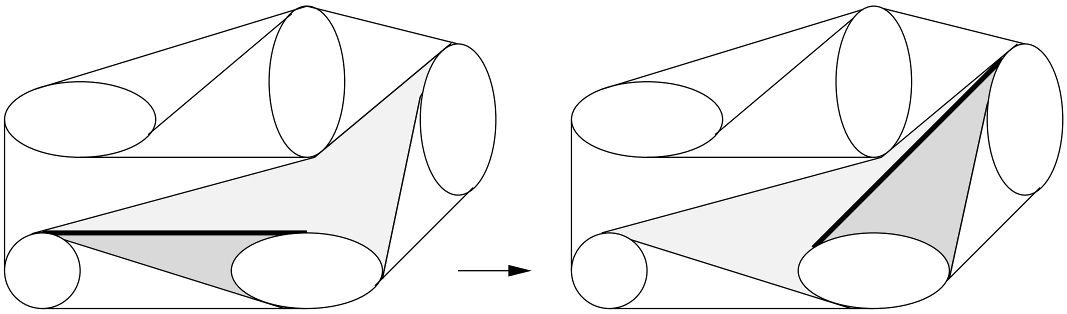

FIG. 6.13 – *Bascule d'une bitangente incidente à deux pseudo-triangles.*

L'algorithme de balayage de graphe de visibilité consiste à passer d'une pseudo-triangulation à une autre en basculant une bitangente : l'algorithme guide le choix de la bitangente à basculer et permet ainsi de basculer une à une toutes les bitangentes du graphe de visibilité de la scène. Cet algorithme peut être vu comme un algorithme de parcours des sommets du complexe de visibilité. Ce parcours est cependant effectué d'une façon différente de celui effectué par notre algorithme : quand un sommet v du complexe est passé, le sommet suivant considéré est le sommet extrême droit de la face dont v est le sommet extrême gauche.

Les différences entre objets courbes et polygones se font aussi sentir quand nous voulons transposer les pseudo-triangulations dans des scènes polygonales : une pseudo-triangulation devient alors une « vraie » triangulation avec des propriétés différentes.

Une pseudo-triangulation a moins de bitangentes qu'une triangulation : une pseudo-triangulation de n objets courbes contient $3n-3$ bitangentes alors qu'une triangulation de n_p polygones ayant au total n sommets contient $2n-3+3n_p-n_{conv}$ arêtes, où n_{conv} est le nombre d'arêtes de l'enveloppe convexe.

Ces différences sont visibles dans la figure 6.14 où nous avons affiché une pseudo-triangulation d'une scène de trois ellipses et la triangulation « correspondante » quand les ellipses sont remplacées par des segments de droite.

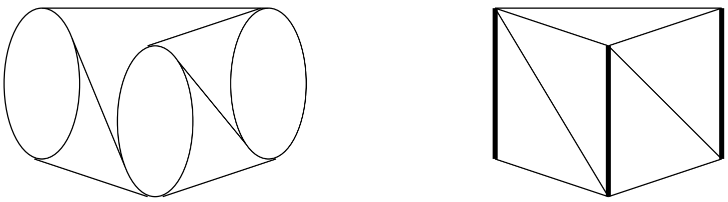

FIG. 6.14 – *Différences entre* **a.** *Une pseudo-triangulation, et* **b.** *Une triangulation.*

Ces différences sont dues au fait que toutes les arêtes ayant un point p pour extrémité sont non sécantes (figure 6.15a) alors que des tangentes à d'un objet O et de même type (μ ou λ) peuvent être sécantes (figure 6.15b).

FIG. 6.15 – *Tangentes* **a.** *Passant par un même point et non sécantes.* **b.** *À un même objet courbe et sécantes .*

Si nous appliquons quand même l'algorithme de bascule des diagonales des quadrangles à une triangulation, nous ne parcourons pas toutes les arêtes du graphe de visibilité. En effet, une arête (p, q) du graphe de visibilité peut, dans toutes les triangulations possibles de la scène, séparer deux triangles dont l'union forme un quadrangle concave. La seconde diagonale du quadrangle est alors située à l'extérieur du quadrangle et soit coupe un obstacle de la scène (figure 6.16a), soit coupe une arête de la triangulation (figure 6.16b), soit est déjà une arête de la triangulation (figure 6.16c).

L'arête (p, q) ne peut alors jamais être basculée et n'est donc pas visitée par l'algorithme.

Les ensembles de tangentes à un objet courbe convexe et les ensembles de « tangentes » aux côtés d'un polygone, c'est-à-dire de droites passant par les sommets du polygone, ont des propriétés fondamentalement différentes (même si le polygone est réduit à un segment de droite). Ces différences fondamentales viennent notamment du

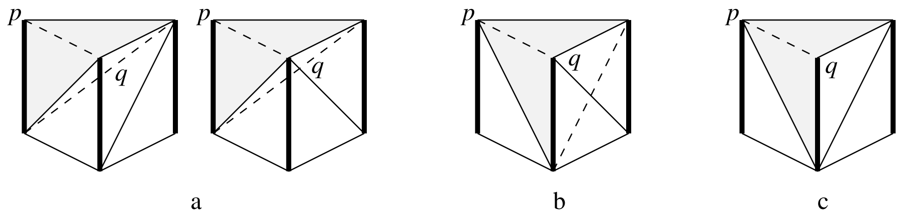

FIG. 6.16 – *Problèmes avec la diagonale extérieure d'un quadrangle.* **a.** *Diagonale coupant un obstacle.* **b.** *Diagonale coupant une arête de la triangulation.* **c.** *Diagonale déjà présente dans la triangulation.*

fait que les tangentes à un objet courbe « glissent » autour de l'objet alors que les tangentes d'un segment passent par une extrémité fixe du segment.

Les structures de données créées dans le cadre d'objets courbes (resp. polygones) dont les propriétés permettent de résoudre des problèmes de visibilité perdent alors leur propriétés lorsqu'elles sont transposées dans le cadre de polygones (resp. d'objets courbes) et y deviennent inutilisables.

6.3 Complexe de visibilité : bilan et perspectives

Nous faisons dans cette partie le point sur l'utilisation du complexe de visibilité pour effectuer des calculs de visibilité plus rapidement et évoquons des points qui restent à améliorer.

6.3.1 Utilité du complexe de visibilité

Le complexe de visibilité est un outil puissant pour résoudre certains problèmes de visibilité, en particulier des problèmes de mise à jour de visibilité. Le complexe est une structure de données relativement simple : il ne dépend que de la géométrie de la scène et ne contient pas de structures combinatoires annexes (excepté si nécessaire des arbres binaires de recherche). C'est notamment pour cela qu'il ne permet pas d'améliorer directement les temps de calcul de requêtes discrètes telles que le lancer de rayons. Il est d'ailleurs moins pratique à utiliser dans ce type de requêtes, car il prend une place mémoire proportionnelle à la taille du graphe de visibilité de la scène alors que les structures classiques utilisées pour faire du lancer de rayon n'occupent qu'une place linéaire en fonction du nombre de segments de la scène.

Le complexe de visibilité est en revanche très bien adapté aux calculs continus de visibilité, car il permet d'exploiter la cohérence spatiale de la scène ou la cohérence temporelle des déplacements dans la scène. Il permet d'avoir en plus une description de la visibilité dans l'espace objet (la scène) de façon indépendante de toute visualisation ultérieure.

Le complexe est aussi très utile dans des calculs faisant intervenir toutes les relations de visibilité entre objets de la scène car il contient déjà toutes ces relations. Nous avons par exemple évoqué son utilisation dans des calculs de radiosité.

Malgré tous ses avantages, le complexe de visibilité doit être employé avec précaution, notamment à cause de sa taille. Nous avons déjà évoqué le problème de la taille du complexe de visibilité : sa taille est proportionnelle à la taille du graphe de visibilité, soit une taille en $\Omega(n)$ et $O(n^2)$. L'utilisation du complexe de visibilité sur de grandes scènes peut se révéler problématique. Si les scènes sont relativement denses et que la majorité des objets sont cachés les uns des autres alors le complexe est très utile : il prend un espace mémoire raisonnable et permet d'effectuer des calculs de visibilité de façon vraiment plus rapide que s'il fallait dans ces calculs considérer à chaque fois tous les objets de la scène.

En revanche si les objets sont espacés et se voient pratiquement tous les uns les autres, le complexe de visibilité occupe une place mémoire quadratique et la performance gagnée sur les calculs de visibilité est moindre.

6.3.2 Complexe de visibilité hiérarchique

Tous les calculs de visibilité effectués avec le complexe de visibilité sont exacts, c'est-à-dire que tous les changements de visibilité sont pris en compte. Or un calcul exact peut être trop précis. Dans un calcul de vue par exemple, deux changements de visibilité intervenant dans les directions θ et $\theta + \varepsilon$ peuvent devenir indistinguables et être confondus en un seul changement de visibilité. Ce phénomène est du à l'existence d'une limite de résolution (ici angulaire) en dessous de laquelle les choses sont confondues. Cette limite de résolution est aussi bien matérielle (résolution en pixels d'un écran), physique (résolution d'une pellicule photo) que biologique (résolution de l'oeil). Il devient alors inutile de calculer des détails en deçà de cette résolution, détails qui ne seront pas perçus.

Pour éviter de faire des calculs « visuellement inutiles », il ne faut plus tout calculer, mais seulement ce qui est pertinent. Il ne faut cependant pas perdre des informations qui se révéleraient indispensables par la suite. Pour cela, il ne faut pas faire d'approximations dans le calcul même mais plutôt dans la modélisation de la scène : pour avoir des calculs rapides à un niveau de détails fixé, les calculs sont toujours menés de façon précise, mais ils sont effectués sur une autre scène moins détaillée que la scène originale (figure 6.17). Cependant il doit toujours être possible à partir d'une scène « d'approximation » de remonter à la scène détaillée originale.

La création de scènes à différents niveaux de détails s'effectue grâce à un regroupement hiérarchique des objets. Un groupe d'objets est assimilé à un seul *méta-objet* au niveau de hiérarchie supérieur, les méta-objets peuvent à leur tour être regroupés en un seul objet au niveau supérieur... En haut de la hiérarchie se trouve la version la plus simplifiée de la scène, et tout en bas de la hiérarchie se trouve la scène la plus détaillée, c'est-à-dire la scène originale. Le niveau de hiérarchie à choisir pour un groupe d'objets donné lors d'un calcul de visibilité dépend du niveau de détail adopté pour ce calcul.

Nous exposons des idées sur le calcul et l'utilisation du complexe de visibilité de scènes hiérarchiques.

Comme nous manipulons des polygones, nous pouvons regrouper un ensemble $(O_{i_j})_j$ de polygones en un polygone O_i dont les sommets ont été pris parmi les sommets des

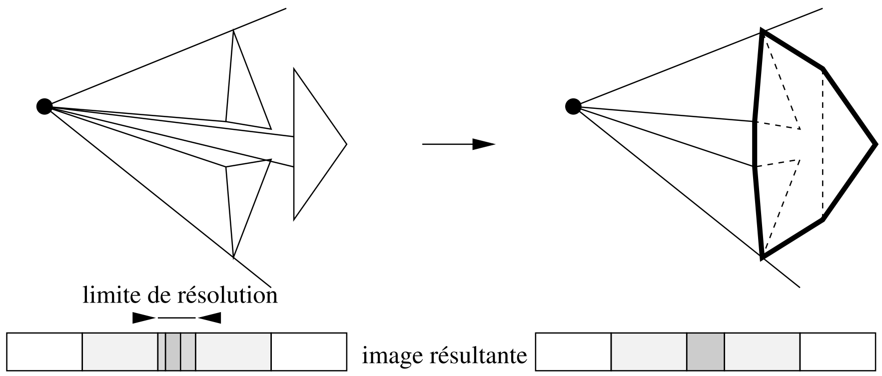

FIG. 6.17 – *Limite de résolution et calcul de vue à différents niveaux de détails.*

polygones O_{i_j}. Considérons maintenant une scène formée d'un ensemble de polygones $(O_i)_i$ où chaque polygone O_i représente un groupe de polygones $(O_{i_j})_j$. Nous calculons alors le complexe de visibilité de la scène composée des polygones O_i (figure 6.18a) et nous calculons le complexe de visibilité de chaque groupe de sous-polygones $(O_{i_j})_j$ (figure 6.18b).

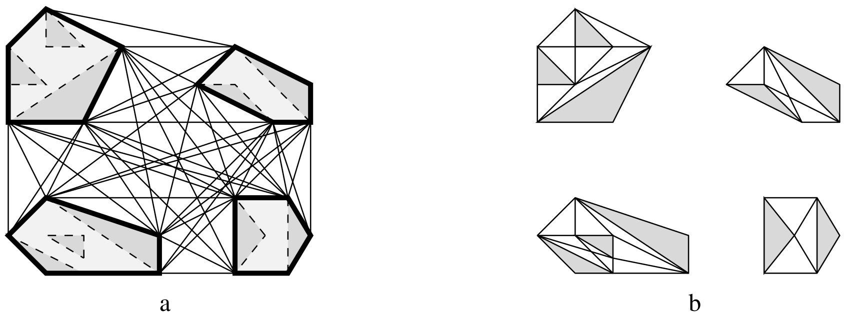

FIG. 6.18 – *Complexe hiérarchique.* **a.** *Calcul du complexe sur les méta-polygones.* **b.** *Calcul du complexe poour chacun des groupes de polygones.*

Comme les sommets des polygones sont des sommets de sous-polygones, les arêtes du complexe de visibilité de la scène sont portées par des courbes duales qui portent aussi les arêtes des sous-complexes. Ces courbes duales permettent alors de relier le complexe aux sous-complexes. Une arête e du complexe de la scène correspondant à un sommet de O_i est découpée dans le sous-complexe des polygones $(O_{i_j})_j$ en sous-arêtes. Nous associons alors à e la liste des sous-arêtes la décomposant.

Cela permet de faire par exemple un calcul de vue hiérarchique. La vue est effectuée en utilisant le complexe de la scène, et la liste des arêtes du complexe correspondant à la vue est gardée en mémoire. Si la visibilité entre deux points consécutifs p et q de la vue situés sur un même polygone doit être calculée avec plus de détails, alors grâce à liste de sous-arêtes des arêtes e_p et e_q associées à p et q, nous calculons les arêtes associées à p et q dans le sous-complexe du polygone, et calculons la vue entre ces deux arêtes (figure 6.19).

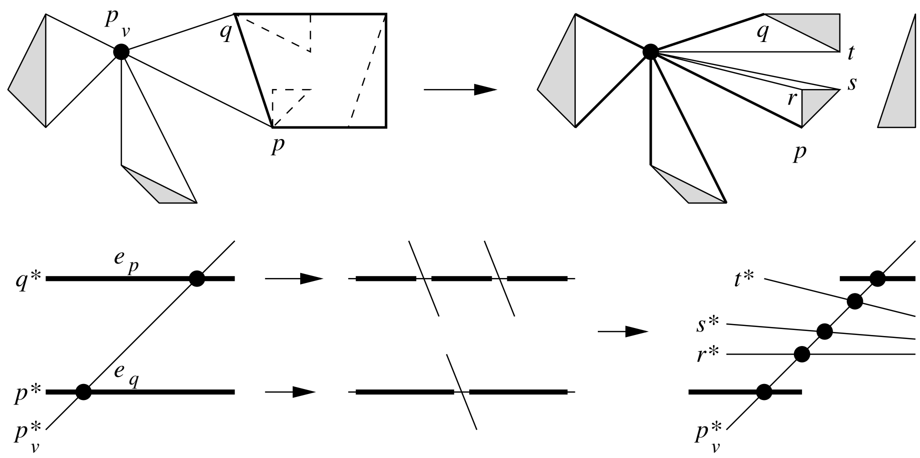

FIG. 6.19 – *Calcul de vue hiérarchique dans le complexe de visibilité hiérarchique.*

Nous envisageons la hiérarchisation du complexe de visibilité non seulement comme un moyen pour effectuer des calculs plus rapides selon le niveau de détails demandé, mais aussi comme un moyen permettant de rendre le complexe plus utilisable en pratique sur de grandes scènes.

En effet, nous avons déjà fait remarqué que la place mémoire occupée par le complexe pouvait être élevée. La hiérarchisation, telle que nous l'envisageons, permet de réduire cette place mémoire sans pénaliser lourdement les calculs de visibilité. Au lieu de calculer le complexe de tous les sous-polygones, nous calculons seulement le complexe des méta-polygones et les complexes des sous-ensembles de polygones de ces méta-polygones, ce qui prend au total une place mémoire nettement moins élevée. Nous n'avons pas alors dans le complexe hiérarchique les calculs de visibilité entre sous-polygones de méta-polygones différents. Si les méta-polygones sont relativement éloignés les uns des autres, alors ces calculs sont en général trop précis et la visibilité entre méta-polygones suffit comme approximation. Sinon, ces calculs peuvent être effectués rapidement grâce aux informations déjà présentes dans le complexe hiérarchique.

Il faudra aussi, pour tirer pleinement profit du complexe hiérarchique, développer des critères et des tests permettant de définir à chaque instant des calculs de visibilité à quel niveau de détails ces calculs doivent être effectués.

6.3.3 Maintien topologique du complexe de visibilité

Nous avons exposé un algorithme de balayage topologique du graphe de visibilité qui permet de calculer ce dernier en temps optimal. Le but d'un balayage topologique est de faire un balayage cohérent, sans être obligé de passer à chaque fois l'élément minimal global. Le balayage topologique est plus rapide : il permet d'éviter l'utilisation d'une queue de priorité et le coût logarithmique supplémentaire qui en découle. Il est aussi plus souple : plusieurs éléments peuvent être passés à chaque étape et même tous ces éléments peuvent être passés simultanément en parallèle.

Les algorithmes de maintien de visibilité dynamique – notamment le maintien dynamique du complexe de visibilité – que nous avons exposés utilisent toujours un balayage droit, c'est-à-dire que les événement sont traités dans l'ordre de leur date d'apparition. Or, quand nous cherchons à maintenir le complexe de visibilité lorsque les objets se déplacent, certains changements élémentaires de visibilité sont indépendants : l'ordre de mise à jour de deux changements de visibilité se produisant dans deux endroits mutuellement cachés de la scène est indifférent. Il serait alors intéressant d'avoir des structures de données qui permettent de déterminer quels sont tous les changements de visibilité qui peuvent être traités simultanément à un instant donné. Cela permettraient alors de ne plus utiliser de queue de priorité.

6.3.4 Le mot de la fin

Le complexe de visibilité n'est pas qu'une structure théorique : il existe vraiment, nous l'avons programmé :-) Bien que ce soit un outil puissant pour faire des calculs de visibilité, comme tout outil il a ses limites d'application et doit être employé à propos. Dans le cas de grandes scènes complexes, le complexe peut paradoxalement se révéler être trop puissant : contenant toutes les relations de visibilité entre les objets de la scène, il peut alors être trop grand et trop précis. L'utilisation du complexe hiérarchique, ou d'autres méthodes permettant de ne calculer complètement que les complexes de visibilité de parties de scènes et de faire rapidement des calculs de visibilité inter-scènes à un niveau de détails suffisant, peuvent alors concilier les avantages et les inconvénients du complexe de visibilité.