Université de Montréal

Simplification et morphisme en temps réel de modèles articulés

par

Jocelyn Houle

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la faculté des études supérieures

en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en informatique

Août 2000

Université de Montréal

Faculté des études supérieures

Ce mémoire de maîtrise intitulé

Simplification et morphisme en temps réel de modèles articulés

présenté par

Jocelyn Houle

a été évalué par un jury composé des personnes suivantes :

Président :	Sébastien Roy
Directeur de recherche :	Pierre Poulin
Membre :	Jean Meunier

Sommaire

Ce mémoire décrit une technique de simplification de maillages articulés. Cette technique est combinée à un morphisme permettant de réduire les artefacts de transition entre les différents niveaux de détails créés lors de la simplification.

Ce travail a été effectué en collaboration avec Electronic Arts Canada, un joueur majeur dans l'industrie des jeux vidéo. Notre recherche a donc été façonnée par leurs différents besoins, particulièrement en ce qui a trait à la performance. Nous devions donc développer une technique de simplification dont les résultats sont applicables en temps réel sur une machine courante.

Notre technique arrive non seulement à être applicable en temps réel, mais réussit en plus à réduire le problème de surgissement qui infestait les transitions.

Mots-clefs :

modèle articulé, modèle squelettique, simplification de maillages, morphisme, temps réel, jeux vidéo

Table des matières

Table des figures

Remerciements

Le tout premier remerciement revient à Pierre Poulin, mon directeur de recherche. Pierre m'a accueilli dans son laboratoire, m'a supporté dans mon cheminement et m'a aussi offert son amitié... Je ne pourrais pas demander mieux pour ma maîtrise. C'est grâce à lui si j'ai pu apprendre l'infographie : ses techniques, ses concepts, ses anecdotes et ses Coons (et Siggraph) Awards. Merci Pierre, pour m'avoir permis tout cela... C'est aussi grâce à lui si j'ai passé trois mois à Vancouver chez Electronic Arts. J'ai maintenant deux années de précieux souvenirs que je chérirai à jamais... Dans quelques années, genre, lorsque mes vœux pieux de rester proche auront fait place à la cruelle réalité, tu me manqueras... Profite bien du restant de ta sabbatique... et de tes fins de semaines !

Je remercie également Electronics Arts Canada pour leur ouverture d'esprit et pour m'avoir permis de travailler dans un contexte privilégié. Les modèles de joueur ainsi que les séquences d'animation associées sont leur gracieuseté. Un merci particulier revient à John Buchanan, pour ses efforts et ses initiatives dans l'intégration de la recherche infographique au sein d'une compagnie d'avant-plan. Merci au nom des trois internes, pour une opportunité en or. Merci à Mark et Claude pour tant de parties de foosball. Merci à Matt et Darren, de l'équipe Fifa, pour le temps et les informations fournies. Merci aussi à Paul Lalonde pour m'avoir montré "l'aigle" du PS2. Merci à Warren, Becky, Vivian, de même que Bernadette (du YWCA) et les autres pour les souvenirs qui me restent de mon séjour à Vancouver.

Incidemment, je remercie tous les gens du laboratoire IMAGER à U.B.C. pour autant de souvenirs. À Alain Fournier, décédé depuis, pour m'avoir montré la passion de la recherche et de la vérité... À Kelly Booth, pour m'avoir fait comprendre qu'une caméra *pinpoint* n'avait pas de *lookAt*. À Michiel van de Panne pour les discussions et les démos. Merci aussi à tout le lab IMAGER (Lisa, Rob, Rob, Cedric, Dave, Jason et les autres) pour avoir enduré ma présentation de même que ma présence (parfois importune) devant un PC à apprendre MFC en attendant une license de compilateur.

Mais de tous les labs du monde, il y en a un qui me tient particulièrement à cœur. Au LIGUM, et à tous ses membres, je dis merci !

Merci à Luc, pour tant de discussions éclairantes. Pour m'avoir aussi démontré qu'avec un dévouement et une discipline exemplaires (avec, bien sûr, l'intelligence et le talent appropriés), tout est possible... "Y'a sûrement moyen de le faire...", n'est-ce pas ? Merci aussi pour ton amitié... et ton humour !

Merci à Laurent, pour la constance et la régularité. Si j'étais une cigale, tu serais la fourmi. Merci pour avoir parti le bal et nous avoir initié à tant de choses : jongler, le cube Rubik, le diabolo... Merci aussi pour les opportunités passées, présentes et à venir. J'aurais juste une question : why no knots ?

Merci à Philippe, pour m'avoir montré la passion, l'ambition et la raison, enchevêtrées dans un labyrinthe sans mur... Merci pour m'avoir montré que la seule vraie façon d'être est d'être... vrai ! Après tant de discussions, je ne lirai plus jamais Kundera de la même façon...

À Sébastien et Éric, vous pourrez enfin travailler en paix. Je ne me qualifie pas pour un doctorat. C'est connu : il faut être marié et avoir un enfant lors de la permière année. J'ai juste trop peur de couler le test... J'ai en plus haute estime votre génie et votre folie ; un habile mélange.

À Sébastien, roi du Rubik, et maître de la répartie, je dis : 'i' et 'r' se répètent dans "roi" et "Rubik", alors que 'aeirt' sont les lettres communes de "maître" et "répartie". On sait jamais... Ça peut être utile pour les symétries !

À Éric, qui sacre toujours autant pendant des games de Quake3 : continue ! Tu serais trop *parfait* autrement...

Aux anciens, merci pour m'avoir laissé un lab joyeux (MC), rebelle (Éric), unique (Mathieu), bien configuré (Martin), mais surtout, plein d'héritage et de diversité (Christiane, Cyriaque, Patrick, Karim, Arash, Richard, Normand, Yi-Jen, et aussi Christian et Marie-Jeanne).

Merci aussi à tous les Français que le lab a aimé accueillir : Alexis, Ludovic, Nicolas, Laure et Alex. Vous nous manquez...

Aux nouveaux étudiants (Mathieu, François et Sydney), je souhaite bonne chance ! Sachez que je vous envie : les machines ont maintenant du *texture mapping* en *hardware* ! Sachez aussi qu'on retire du lab ce qu'on y apporte...

Merci aussi aux étudiants de l'université, avec qui j'ai partagé études, dîners, discussions, escalade, tarot et beignes...

Je tiens aussi à remercier mes proches qui ont su me supporter dans les moments difficiles. À mes parents, je lègue mes vingt ans d'étude et de rêves qui culminent en cette maîtrise. L'enfant velcro quittera bientôt pour son propre ciel, et je vous remercie du fond du cœur pour le soutien de tant d'années, pour les efforts et les sacrifices à mon égard... Pour ce que je suis devenu... Merci à ma sœur, pour avoir pavé le chemin, pour avoir brisé les murs que je n'aurais pas pu surmonter... Merci aussi à ma copine pour sa patience, sa complicité et sa compréhension. Merci aussi pour son amour qui demeurera, j'espère, pour des siècles encore... Merci à mes deux grands chums, Franco et Steve, pour tant d'années passées, et tant d'années à venir... Merci aussi à tous les autres copains pour... le fun !

Je remercie finalement le comité évaluateur pour avoir approuvé de leur sceau la qualité de cet humble document. La fierté accumulée de ces deux années de labeur fait maintenant place à la satisfaction d'un travail accompli. Les fruits de mes efforts peuvent maintenant accumuler la poussière au fond des tablettes...

À tous ceux que j'oublie, je demande pardon...

Fiat lux...

Fiat luxo...

Fiat knickknack !

Chapitre 1

Introduction

L'informatique graphique est un vaste domaine constamment en évolution. Du cinéma aux jeux vidéo, de la conception assistée par ordinateur aux systèmes de visualisation scientifique, en passant par les annonces publicitaires et l'enseignement, son ubiquité transparaît dans la vie quotidienne.

Notre recherche s'inscrit dans un cadre 3D en temps réel dans lequel le modèle géométrique 3D constitue la composante de base. Il existe plusieurs façons de représenter un modèle. Certaines primitives de base, telles la sphère, le cône et le cube, peuvent être utilisées. Elles sont parfois inappropriées pour représenter certains objets. Pour des modèles comme des êtres humains, par exemple, on utilise traditionnellement des surfaces paramétriques, notamment parce qu'elles sont courbes et lisses. Leur description concise (quelques points de contrôle) permet également une édition plus simple et plus intuitive lors de la modélisation. Afin de bénéficier de la puissance des cartes accélératrices 3D, nous devons toutefois utiliser une primitive élémentaire servant de dénominateur commun : le triangle.

Puisque les triangles sont restreints dans leur nature, on doit souvent en avoir un très grand nombre afin de représenter adéquatement les objets. Les surfaces paramétriques peuvent être discrétisées au niveau désiré, c'est-à-dire qu'elles peuvent être triangulées à souhait, jusqu'à la précision de la machine. Les numérisateurs 3D (*scanners*) sont de plus en plus précis (de l'ordre du micromètre). Il est donc facile, de nos jours, d'obtenir un très grand nombre de triangles. Ce qui est plus difficile, c'est de ne pas en obtenir un trop grand nombre...

La simplification de maillage est un procédé automatique par lequel la complexité polygonale d'un modèle peut être réduite. Cette réduction s'opère typiquement par une série de modifications locales permettant d'enlever des faces au modèle. C'est grâce à une *métrique*

que l'on évalue l'impact d'une opération de simplification sur un modèle polygonal. L'impact tolérable d'une simplification dépend de l'application du modèle résultant. Idéalement, un modèle simplifié est indiscernable du modèle original et offre des gains en vitesse (affichage, détection de collisions, etc.) et même en mémoire. Le chapitre 2 passe en revue l'essentiel des techniques de simplification de maillage.

La présente recherche s'applique sur des modèles animés utilisés dans des jeux vidéo. Développée en considérant les besoins particuliers d'*Electronic Arts* (une des plus grandes compagnies actuelles de jeux vidéo), notre technique s'inscrit dans un cadre strict que nous décrivons au chapitre 3. Le chapitre 4 décrit notre technique de simplification pouvant être appliquée sur des modèles animés. Par la suite, au chapitre 5, nous décrivons comment facilement effectuer des transitions de haute qualité entre les différents niveaux de simplification. Nous discutons ensuite au chapitre 6 des différentes extensions dont notre technique pourrait bénéficier. Finalement, nous concluons le présent travail en résumant les points saillants de notre recherche.

Chapitre 2

Travaux antérieurs

Si tu ne sais plus où tu vas, regarde d'où tu viens...

2.1 La conception

La simplification de modèles s'esquisse relativement tôt en infographie. Déjà, en 1976, Clark [Cla76] énonce quelques simples faits reposant sur le bon sens, mais qui établiront les bases du domaine de la simplification en infographie.

Il remarque tout d'abord qu'un modèle peut contenir des détails superflus qui n'ont que très peu d'impact sur l'image rendue. En effet, si un modèle n'occupe que quelques pixels à l'écran, il n'a certainement pas besoin d'une qualité fine. Ainsi, un être humain pourrait être représenté plus convenablement en n'utilisant qu'une simple boîte avec des membres grossiers, voire même une simple boîte, en autant, bien sûr, que le rendu de ce modèle soit suffisamment grossier. De la même façon, un modèle se déplaçant rapidement pourrait être considéré moins important en termes de qualité de rendu, et pourrait ainsi être simplifié proportionnellement à sa vitesse à l'écran.

Pour atteindre ses buts, Clark développe une arborescence multi-résolution dans laquelle chaque nœud contient une représentation différente pour le même objet. Puisque l'on veut qu'un modèle simple offre un avantage en termes d'efficacité, l'arborescence est construite de façon à ce que les niveaux supérieurs correspondent aux représentations les plus grossières. Si la représentation à un nœud donné est suffisante, alors elle sera utilisée directement. Autrement, la représentation est récursivement remplacée par celles des enfants, lesquels correspondent à

des versions plus détaillées des représentations parentes.

La technique de Clark, quoique relativement simple du point de vue conceptuel, s'est toutefois avérée relativement compliquée pour l'époque, puisqu'elle nécessitait de construire chaque représentation manuellement, faute de technique automatique. Bien qu'il fallut attendre quinze ans afin d'obtenir de telles méthodes, il n'en demeure pas moins que l'idée derrière l'article de Clark reste toujours d'actualité : améliorer la performance avec des pertes de qualité modestes, voire même imperceptibles.

Quelques années plus tard, Crow [Cro82] décrit une technique de génération d'images pour laquelle les objets, afin d'être rendus plus efficacement, peuvent être composés de plusieurs représentations qui sont sélectionnées selon leur nécessité. Contrairement à Clark, Crow ne décompose pas le modèle en sous-parties : chacune des versions simplifiées représente le modèle original. Cet article a cependant eu peu d'impact puisqu'une grande partie de l'article était dédiée à la description d'une syntaxe de scène. Cette technique a néanmoins le mérite de constituer l'une des premières applications concrètes des modèles multi-résolution.

Mais compte tenu de l'engouement grandissant pour les surfaces paramétriques popularisées par diverses formalisations [Béz66, Coo74, CC78, Bar83, LLM+90], ces germes d'idées sont demeurés marginaux pendant plusieurs années. En fait, puisque les surfaces paramétriques peuvent être hautement raffinées (infiniment, en théorie), les modèles sont plutôt devenus de plus en plus complexes. C'est le besoin pressant de simplification qui dépoussiéra le domaine.

2.2 La naissance

Bien que les idées de Clark soient simples et claires, elles n'ont été appliquées que dans quelques simulateurs de vol de pointe, qui étaient les seuls à pouvoir se permettre la construction et la gestion de modèles multi-résolution. Les conditions mémoires plus restreintes ainsi que les implications de devoir construire les diverses résolutions manuellement expliquent en grande partie cette situation. Les besoins grandissants de techniques automatiques de génération de modèles simplifiés entraînent qu'en 1992, une poussée de contributions marque le véritable commencement de la simplification de modèles polygonaux.

2.2.1 Rossignac & Borrel

Le premier ouvrage traitant de simplification automatique provient du domaine du design assisté par ordinateur (CAD). Rossignac et Borrel ont développé chez *IBM* ([RB92], puis publié

dans [RB93]) une technique rapide et robuste destinée aux modèles CAD, peu réputés pour leur simplicité. Leur technique se devait d'être rapide, robuste et applicable à des variétés 3D (modèles dits *non-manifolds*). Elle comporte six étapes :

1. **Gradation (*grading*)** La gradation consiste à associer, pour chaque sommet du modèle, un poids correspondant à l'importance visuelle relative de ce point dans le modèle. Ainsi, par exemple, les sommets compris dans une zone de haute courbure locale se verront offrir un poids élevé comparé aux zones de faible courbure. Également, les sommets associés à des triangles de grande superficie (aire) obtiendront un poids plus élevé que ceux associés à de plus petits triangles.

2. **Triangulation** Chaque polygone de plus de trois sommets est triangulé de façon à obtenir, en bout de ligne, un maillage triangulaire.

3. **Agrégation (*clustering*)** L'agrégation consiste à regrouper les sommets selon leur proximité, en les englobant dans une grille régulière.

4. **Synthèse** Pour chaque élément de grille (agrégat, ou *cluster*), la synthèse calcule un sommet moyen en tenant compte des poids calculés lors de la gradation. Tous les sommets compris dans cet élément de grille sont ensuite remplacés par ce seul et unique représentant.

5. **Élimination** Si plus d'un sommet d'un triangle se trouve dans un même agrégat, alors le triangle sera dégénéré en ligne, ou peut-être même en point, puisqu'il utilise plus d'une fois le même sommet. La phase d'élimination consiste tout simplement à retirer ces triangles inutiles de la structure.

6. **Rajustement de normales** Dans cette dernière étape, des normales plus adéquates sont calculées afin d'obtenir des résultats plus agréables à l'œil.

Bien que robuste et fort rapide, cette technique comporte toutefois des inconvénients de taille. Tout d'abord, la qualité du modèle simplifié dépend beaucoup de la résolution de la grille. Un mauvais choix de grille peut facilement amplifier un triangle originalement petit, et produire des artefacts loin d'être négligeables (voir figure 2.1). Incidemment, le passage entre deux niveaux de détails (créés chacun avec leur propre grille) souffre de surgissement (*popping*) puisqu'il n'y a pas de cohérence entre les grilles utilisées.

Malgré tout, cette technique demeure extrêmement rapide, et sert de base pour toutes les techniques de simplification par agrégation.

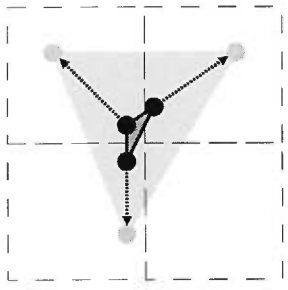


FIG. 2.1: L'utilisation d'une grille régulière (en pointillé) entraîne parfois, lors de l'étape de la synthèse, que les sommets moyens calculés (représentés en pâle) étirent les polygones.

2.2.2 Turk

Bien que fonctionnelle, la méthode par agrégation ne tient pas compte de la nature du modèle simplifié et offre donc des résultats mitigés. C'est en voulant profiter des particularités des objets lisses que Turk développe sa technique [Tur92]. L'idée de base est analogue à la mue des animaux.

On commence par échantillonner aléatoirement la surface : ces échantillons constitueront les sommets du modèle simplifié. Afin de distribuer les sommets uniformément sur la surface, Turk simule des forces de répulsion entre les sommets voisins. Au besoin, les sommets trop rapprochés l'un de l'autre sont déplacés sur les faces adjacentes, mais en demeurant toujours sur la surface du modèle. Une fois positionnés, ces sommets sont ajoutés au modèle original par une technique appelée "tesselation mutuelle". Les sommets originaux sont ensuite enlevés un à un. Afin de conserver la continuité de la surface, les ouvertures ainsi créées sont remplies de nouveaux triangles (étape de *re-tiling*). On se retrouve finalement à remplacer les faces et les sommets originaux par de nouvelles valeurs : le modèle a mué.

Le résultat final est fort approprié pour les objets lisses, mais a malheureusement tôt fait d'adoucir les fortes discontinuités, telles les pointes et les coins. Néanmoins, la technique s'applique avec succès aux modèles lisses et a l'avantage de permettre de spécifier en termes de sommets la taille finale du modèle.

2.2.3 Schroeder *et al.*

Au même moment, Schroeder *et al.* développent leur technique de décimation [SZL92]. Pour la résumer sommairement, il suffit d'itérer sur les sommets et d'enlever tous ceux dont

la distance au plan de support moyen des faces adjacentes est en deçà d'une certaine distance spécifiée. Le trou créé est ensuite bouché par une technique nommée *loop splitting* qui s'efforce de construire des triangles les plus équilatéraux possible. Cette technique de simplification offre de bons résultats pour des modèles créés par la technique du *marching cube* [LC87] pour laquelle elle a été conçue. Malgré tout, elle demeure fortement limitée par le fait que le modèle original est constitué d'un sous-ensemble des sommets originaux.

Ces travaux sont en bonne partie responsables de l'essor qu'a alors subi le domaine, mais ils ne suffisent toutefois pas à expliquer l'intensité de l'engouement qui se créa. La source de la frénésie trouve plutôt ses racines dans un article qui, à première vue, semble n'avoir aucun rapport avec la simplification. En ne partant que d'échantillons spatiaux (un simple nuage de points), Hoppe *et al.* reconstruisent la surface d'un modèle [HDD+92]. Ces travaux sont à l'origine de la simplification par fusion (*edge collapse simplification*), laquelle constitue la base de la grande majorité des techniques actuelles (y compris la présente).

2.3 L'adolescence

Les premiers pas ayant été effectués, il s'agissait maintenant d'améliorer les résultats. Intrigués par l'idée de trouver une façon de représenter des modèles avec moins de triangles, Hoppe *et al.* se sont attaqué à l'optimisation de maillage [HDD+93].

2.3.1 La puberté

Ces travaux reposent sur trois opérateurs locaux : *edge collapse*, *edge split* et *edge swap* (figure 2.2). Le premier consiste à raccourcir une arête jusqu'à néant, enlevant par le fait même les deux faces qui la partagent (ou une seule dans le cas d'une arête de contour). Le second opérateur, l'*edge split*, consiste à scinder les deux faces voisines de façon à relier les sommets originalement non connectés. Le dernier opérateur, l'*edge swap*, remplace quant à lui l'arête partagée par deux sommets opposés par l'arête sécante, reliant ainsi plutôt les deux autres sommets. En prenant bien soin de respecter la topologie, l'algorithme applique itérativement ces opérateurs en tentant de minimiser une énergie. Cette énergie est calculée en trois termes.

Le premier terme, l'énergie de distance, est tout simplement la distance au carré d'un sommet à sa position de départ. Il sert à inciter les sommets à demeurer près de leur position origi-

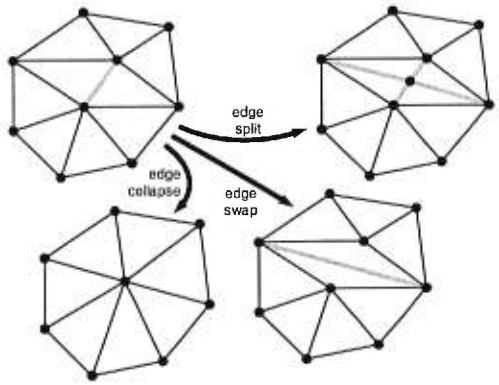


FIG. 2.2: Les opérateurs locaux d'optimisation de maillage.

nale. Le second terme, appelé l'énergie de représentation, sert quant à lui à réduire le nombre de sommets dans le modèle final. Ainsi, en associant une valeur proportionnelle au nombre total de sommets dans la représentation courante, on peut arriver à enlever des sommets quand bien même une telle action est forcée d'accroître l'énergie de distance. Mais ces deux termes ne sont toutefois pas suffisants, puisqu'il n'existe peut-être pas de minimum à un tel système. Ceci se manifeste parfois par le positionnement optimal exagéré d'un sommet, lequel entraîne l'apparition d'artefacts visuels importants. Pour pallier à ce problème, les auteurs ont recours à un troisième terme : une énergie de ressort. En plaçant initialement sur chaque arête un ressort au repos, une énergie proportionnelle à la distorsion des faces est ainsi ajoutée, ce qui assure, aux dires des auteurs, l'existence d'un minimum.

Ayant précédemment développé une technique de reconstruction de surface, les auteurs ont validé leur technique de la façon suivante : partant d'un modèle quelconque, ils échantillonnent la surface de façon à n'obtenir que des points. Ensuite, ils reconstruisent la surface et la simplifient grâce à leur nouvelle technique. Les résultats, quoique coûteux à calculer, demeurent toutefois fort impressionnants.

2.3.2 La crise d'adolescence

Quelques années plus tard, Hoppe démontre que seul l'opérateur d'*edge collapse* est suffisant afin obtenir une simplification de qualité [Hop96]. L'idée clef de la technique consiste à effectuer une série de fusions de sommets (*ecol*) tout en conservant la transformation inverse, la fission (*vsplit*). Ainsi, Hoppe réussit à simplifier un modèle jusqu'à un modèle extrêmement simple (quelques triangles seulement), mais conserve également, dans une liste, les informa-

tions nécessaires pour reconstituer le modèle original. Le résultat final offre autant de modèles intermédiaires qu'il y a de structures de fusion, puisqu'on peut aisément appliquer autant de *vsplits* que l'on désire, et revenir comme bon nous semble. Il baptise cette structure "maillage progressif" (*progressive mesh*), et lui trouve ensuite une panoplie de fonctionnalités intéressantes.

Tout d'abord, Hoppe introduit la simplification d'attributs, permettant ainsi de conserver les discontinuités de couleurs, de normales ou de textures. Cet ajout, quoique coûteux, offre toutefois de bien meilleurs résultats pour les modèles utilisant de tels attributs, plutôt que de conserver intactes les valeurs lors de la simplification.

Puisque le maillage progressif consiste en un modèle simplifié (appelé M_0), associé à une liste de *vsplits* (lesquels appliqués dans l'ordre offriront le modèle original), Hoppe développe le principe de transmission progressive. Cette technique consiste tout simplement à envoyer rapidement le modèle simplifié M_0, et ensuite envoyer progressivement les *vsplits* que l'on applique au fur et à mesure. De cette façon, on peut rapidement avoir une idée grossière du modèle, pour ensuite le raffiner avec le temps. Cette technique est très alléchante dans un contexte de bande passante réduite.

Hoppe réussit même à n'appliquer qu'un sous-ensemble des structures de fusion, et permettre alors un raffinement sélectif (*selective refinement*). En n'appliquant que les fusions qui affectent les composantes à l'intérieur du volume de vue (*frustum*), on peut simplifier un modèle là où il n'est pas visible. De la même façon, on peut ajouter les fusions qui n'affectent que l'arrière du modèle, ou même enlever les faces qui n'ont que quelques pixels d'impact à l'écran. On obtient donc une technique dépendante du point de vue (voir section 2.5.4).

La technique de simplification peut également offrir des taux de compression sans perte, en utilisant un encodage judicieux des structures de fusion. Cette compression est toutefois effectuée au profit de l'efficacité, et offre des rendements relativement faibles.

Finalement, Hoppe introduit le principe de géomorphisme. Prenant deux représentations, l'une de haute résolution et l'autre de faible résolution, Hoppe réussit à déterminer, pour tout sommet du modèle de haute résolution, vers quel sommet unique il correspond dans le modèle de faible résolution. Connaissant cette relation, Hoppe effectue une simple interpolation linéaire entre les positions et crée ainsi un modèle intermédiaire pouvant être paramétré dans le temps. La création de tels modèles, appelés "géomorphes", réduit de façon drastique le phénomène de surgissement (*popping*) apparent lors de la transition entre deux niveaux de détails. Mais nous parlerons plus en détail des géormorphes au chapitre 5.

2.4 L'âge adulte

Les maillages progressifs offrent certes d'excellents résultats, mais ils sont cependant sujets à un précalcul relativement long et à une consommation mémoire tout aussi importante. La métrique par quadriques d'erreur s'attaque simultanément à ces deux limitations.

2.4.1 La maturation

La métrique des quadriques d'erreur tire son origine de Ronfard et Rossignac [RR96]. Dans leur article, une simplification par fusion est effectuée d'une façon inspirée des maillages progressifs. Mais contrairement à Hoppe, les auteurs tentent d'optimiser la position finale d'une fusion en utilisant les plans de support des faces adjacentes aux deux sommets qui fusionnent. Les auteurs font remarquer qu'un sommet peut être vu comme siégeant au point d'intersection exact de ces plans de support. Considérant ceci, une position optimale pour deux sommets fusionnant ensemble pourrait tout aussi bien tenter de minimiser la distance maximale à chacun de ces plans (la distance au carré est utilisée, question de simplifier les calculs). Afin de tenir compte de la progression de la simplification, l'ensemble des plans est conservé avec le sommet fusionné. Puisque cet historique grandit avec la simplification, le calcul d'optimisation devient de plus en plus complexe au fur et à mesure que la simplification progresse.

2.4.2 Le nouveau système métrique

La simplification utilisant la métrique des quadriques d'erreur (*quadric error metric*) tient cependant son origine des travaux de Garland et Heckbert [GH97, GH98].

Dans leur premier article [GH97], Garland et Heckbert partent du fait qu'une surface quadrique (voir section 4.3.2) peut être représentée par une matrice symétrique 4×4 (10 valeurs). Plutôt que de conserver un historique de plans, ils décident de sommer les quadriques ensemble, ce qui insère quelque peu d'imprécision (puisque quelques plans pourraient être sommés plus d'une fois), mais a cependant l'avantage de ne pas gonfler indûment les ressources (mémoire et temps de calcul). En associant une quadrique à chaque sommet, et en évaluant la somme de quadriques de chacun des sommets que l'on fusionne, on peut effectuer une simplification à la manière de Hoppe [Hop96]. Afin de trouver la position optimale du sommet fusionné, on n'a qu'à inverser la matrice de quadrique, ou prendre la meilleure évaluation sur le segment si la matrice ne s'inverse pas. Les résultats finaux sont de très grande qualité et ne prennent qu'une fraction du temps en comparaison des maillages progressifs pour une qualité comparable.

Ils proposent également de permettre de fusionner des sommets géométriquement proches, sans pour autant qu'ils soient logiquement connectés (c'est-à-dire, partageant une ou des faces). Ceci permet de joindre des parties distinctes, mais rapprochées du modèle. Cette extension exige cependant de définir les transformations de fusion de façon à supporter les modèles n'étant plus des variétés 3D, ce qui constitue une autre de leurs améliorations.

Leur technique peut aussi tenir compte des attributs aux sommets en étendant la matrice dans des dimensions supérieures [GH98]. Cela leur permet de considérer la couleur aux sommets, ainsi que les coordonnées de texture lors de la simplification, question de conserver les discontinuités d'attributs le plus longtemps possible.

L'avantage principal de la technique de simplification par quadriques d'erreur repose principalement dans sa rapidité, laquelle ne s'effectue pas au profit de la qualité. Compte tenu que la technique ne prend qu'une fraction du temps de celle de Hoppe [Hop96], mais offre une qualité tout aussi respectable, il n'est pas surprenant que Hoppe lui-même ait eu son mot à dire.

Dans sa réplique [Hop99], Hoppe réussit à simplifier l'extension dans un espace supérieur (ce que Garland et Heckbert faisaient) en projetant plutôt les attributs et en analysant leur gradient. La matrice résultante étant creuse (*sparse*), elle peut donc être évaluée et inversée beaucoup plus rapidement, accélérant par le fait même l'algorithme en entier (du moins, lorsque l'on considère les attributs). Hoppe évalue également l'impact d'un cadre amnésique (*memoryless simplification*, voir [LT98, LT99]) ainsi que de la préservation du volume, et conclut qu'ils ont un impact décroissant en importance (mais il offre peu de justifications plus formelles, et l'on doit se fier à son jugement).

Compte tenu des avantages évidents qu'offrent les quadriques d'erreur, nous avons choisi de les utiliser dans le cadre de notre recherche. Nous les expliquerons donc plus en détail dans la section 4.3.2.

2.5 Album de famille

Par souci de complétude, nous nous devons de mentionner plusieurs autres travaux. Considérant qu'une classification chronologique serait trop fastidieuse et trop décousue, nous avons opté pour une classification axée plutôt selon le type.

2.5.1 Les agrégats

La construction d'agrégats est un problème trouvant des utilités dans de multiples domaines en informatique. Dans un cadre infographique de simplification de maillage, les agrégats servent tout simplement à regrouper les sommets d'un modèle, pour ainsi les remplacer par un seul (voir la description de la technique de Rossignac et Borrel à la section 2.2.1).

Low & Tan

La méthode par agrégat développée par Rossignac et Borrel est l'une des techniques les plus rapides, mais offre parfois de bien piètres résultats. Confiants de pouvoir améliorer la technique, Low et Tan [LT97] modifient la technique de trois façons : une meilleure heuristique de silhouette, une meilleure hiérarchisation et l'introduction d'arêtes grasses (*thick edges*). Commençons par l'heuristique de silhouette.

La technique originale évalue la probabilité d'un sommet à se situer sur une arête silhouette à $1/\theta$, où θ est l'angle maximum de toutes les paires d'arêtes incidentes au sommet[1]. Low et Tan considèrent quant à eux que $\cos(\theta/2)$ constitue une bien meilleure approximation, compte tenu que $1/\theta$ tend vers l'infini lorsque θ diminue (ce qui éclipserait les autres poids).

La technique originale subdivise uniformément la boîte englobante du modèle à simplifier. Critiquant le choix d'une telle subdivision, Low et Tan utilisent plutôt une agrégation à cellules flottantes (*floating-cell clustering*). Leur technique de subdivision a l'avantage d'être composée d'agrégats centrés autour du sommet le plus important. Puisque ce sommet est situé au centre, la distance moyenne aux autres sommets de l'agrégat est ainsi bornée supérieurement par la demi-diagonale de la cellule, minimisant ainsi le déplacement que subiront ces sommets. Ce simple ajout permet d'obtenir de bien meilleurs représentants, puisque les sommets sont, en moyenne, plus près de leur position originale. Ceci améliore grandement la fidélité des modèles simplifiés et, incidemment, la transition entre les différentes représentations.

Les auteurs introduisent également la notion d'arête grasse (*thick edge*), qui consiste à mettre une épaisseur pour les segments utilisés afin de dessiner les triangles dégénérés. Cette extension, quelque peu satisfaisante en apparence, n'est en fait appropriée que pour les régions rectangulaires d'un modèle qui se dégénèrent en ligne, ce qui est un facteur, hélas, incontrôlable de la technique.

[1] Les auteurs n'offrent malheureusement aucune justification de cette heuristique (elle est uniquement mentionnée à la section 7.4.2 des notes de cours Siggraph 1995 [Ros95]).

Finalement, cette technique n'offre que de modestes améliorations à la technique de Rossignac et Borrel et, puisqu'elle a été brevetée, n'a su trouvé, à notre connaissance, d'autres intérêts que théoriques.

Luebke & Erikson

Parallèlement, Luebke et Erikson développent leur propre technique de simplification [LE97]. Basé sur les travaux de Luebke [Lue96], cet article généralise les travaux de Rossignac et Borrel de plusieurs façons.

Tout d'abord, ils développent un critère de dépendance du point de vue qu'ils utilisent afin de manipuler des scènes complexes en temps réel. À chaque image, une partie de la scène se voile (se simplifie) alors qu'une nouvelle se dévoile (augmentant ainsi les détails de la scène). La technique nécessite en précalcul la construction d'un arbre de sommets (*vertex tree*), effectuée à l'aide d'un *octree*. La mise à jour en temps réel d'une liste de triangles actifs, construite à partir de cet arbre, permet d'effectuer rapidement la gestion de la scène.

Cette technique tire avantage de la cohérence entre les images en ne faisant qu'une mise à jour mineure de la liste de triangles actifs, amortissant ainsi les coûts associés à sa construction. Elle peut également permettre à l'usager de spécifier un budget de polygones, et ainsi forcer la liste de triangles actifs à respecter du mieux possible un nombre fixe de polygones. Elle est cependant restreinte à une scène statique (à cause du coût élevé de la construction de l'arbre de sommets) et souffre également de résultats parfois piteux, communs à toutes les techniques de simplification par agrégation.

Somme toute, les techniques de simplification par agrégation ont l'avantage d'être très rapides, mais offrent toutefois de mauvais résultats lors de fortes simplifications. Ceci vient du fait que les agrégats, étant construits à partir des sommets, bloquent en grande partie l'information véhiculée par les faces. Ceci entraîne un nombre important de dégénérations, mais aussi, et surtout, de fortes exagérations qui ne pardonnent pas.

2.5.2 Les méthodes bornées

Analyse multirésolution de maillage arbitraire

Partant de leurs travaux antérieurs d'optimisation de maillage [HDD+93] ainsi que du doctorat de Lounsbery [Lou94], Eck *et al.* ont développé leur technique de simplification [EDD+95]. Considérant un modèle original M_0, les auteurs le partitionnent en r régions triangulaires, desquelles ils reconstruisent un maillage triangulaire grossier. Utilisant ce complexe de base (*base complex*), ils établissent, pour chaque triangle, une paramétrisation locale ρ_i qu'ils regroupent ensemble en une paramétrisation globale continue ρ. Ils effectuent alors un rééchantillonnage grâce à des subdivisions récursives 4-1 (voir figure 2.3). Puisque ρ est valide aussi bien pour M_0 (le modèle original) que pour M_j (le modèle rééchantillonné 4-1 à partir du complexe de base), on peut calculer l'erreur maximale de déviation, voire l'utiliser comme critère d'arrêt lors de la subdivision. Il est d'ailleurs possible, grâce à cette paramétrisation, de facilement modifier le modèle, et ce, à des niveaux différents de subdivision.

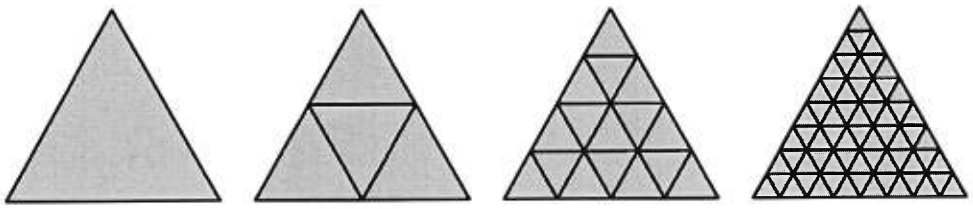


FIG. 2.3: Subdivision 4-1.

La technique n'est toutefois pas triviale. Elle exige la construction d'un diagramme de Voronoi afin d'obtenir une triangulation de Delaunay pour le complexe de base, et demande également des notions de paramétrisation relativement évoluées. Elle utilise également la théorie des ondelettes (*wavelets*) afin de compresser la table d'harmoniques. Le résultat final est cependant de très bonne qualité (quoique lissant un peu trop rapidement les modèles), mais est passablement coûteux en espace et en temps de calcul. Une analyse plus détaillée pourrait intéresser le lecteur [LDW97].

Enveloppes de simplification

La technique des enveloppes de simplification (*simplification envelopes*) constitue une autre façon de simplifier tout en garantissant une borne supérieure de l'erreur [CVM+96]. Cette technique consiste tout simplement à contraindre la simplification à s'effectuer entre deux couches qui emprisonnent le modèle original.

Afin de construire ces couches, les auteurs "gonflent" et "dégonflent" le modèle le long de ses normales géométriques. Chaque triangle de la surface crée alors un prisme tronqué qu'ils nomment "prisme fondamental" (*fundamental prism*).

Ensuite, de façon itérative, un sommet est enlevé si la retriangulation du trou ainsi créé (gracieuseté de Turk [Tur92]) demeure strictement à l'intérieur des deux couches construites précédemment. Le modèle résultant est garanti d'être à une certaine distance maximale ϵ de la surface originale (les auteurs préfèrent utiliser une notation centile % puisqu'elle est invariante de l'échelle du modèle).

Cette technique, quoique garantissant une erreur, ne peut malheureusement pas garantir de simplification, puisqu'un ϵ trop petit, ou un modèle extrême, pourraient entraîner que toute retriangulation sorte des prismes fondamentaux. Quoi qu'il en soit, il s'agit ici d'une technique assez formelle, s'appliquant bien à des variétés 3D.

2.5.3 Les méthodes d'images

La plupart des techniques ne manipulent que les positions et les attributs. Mais quelques techniques osent faire le rendu d'images lors de la simplification, et s'en servent parfois pour conserver les détails, mais parfois aussi pour guider la simplification. Résumons quelques-unes de ces techniques.

Les imposteurs

On peut facilement imaginer que la complexité de la partie visible d'une scène dépasse les capacités matérielles (par exemple, voir une ville complète à vol d'oiseau). Une telle situation oblige, pour que l'on retrouve un temps de rendu interactif, de réduire de beaucoup les petits détails qui, au loin, n'ont plus aucune importance compte tenu de la taille qu'ils occupent à l'écran. Une façon de réduire la complexité consiste à remplacer une partie de la scène par des *imposteurs* [MS95, SLS+96, SS96, SDB97, DSSD99].

L'idée des imposteurs est très simple : il suffit de remplacer une géométrie complexe par

des polygones texturés, plus simples à afficher, mais offrant un résultat visuel comparable. Les imposteurs s'inspirent en fait directement de la technique du *matte painting*, bien connue au cinéma, dans laquelle une partie de la scène est remplacée par une image, souvent placée près de la caméra [2].

Cependant, il existe encore plusieurs problèmes à la technique. Les panneaux texturés demeurent encore relativement compliqués à construire et sont encore plus difficiles à gérer correctement (la transition entre ces panneaux et la polygonisation originale reste encore fort apparente à l'œil). D'autres techniques remplaçant la géométrie éloignée par une géométrie simplifiée existent [Ali96, AL97, AL98, RAPL98, DSV97, DSV98, SLS+96, SS96]. Il est clair que de telles techniques constituent une solution de choix pour la gestion de scènes énormes (comme des villes entières), et l'amélioration graduelle de ces techniques offrira probablement, dans les années à venir, de meilleures solutions en termes d'efficacité et de qualité.

Les cartes de relief

Développant premièrement les techniques nécessaires afin de bien conserver les coordonnées de texture [CMO97], Cohen *et al.* réussissent à conserver l'apparence détaillée d'un modèle en y apposant une carte de relief (*bump map*) faisant office d'une fine polygonisation [COM98]. Ils arrivent donc à fortement réduire le nombre de polygones dans un modèle tout en gardant, au rendu, une grande quantité de détails (enregistrés dans la carte de reliefs). Ils en arrivent même à quantifier l'erreur finale puisqu'ils peuvent calculer la déviation de la carte (une simple texture). Les résultats sont pour le moins surprenants, mais nécessitent un modèle original hautement détaillé. Cette technique offre un attrait particulier, compte tenu du support matériel des cartes de relief, qui a récemment fait son apparition dans les cartes accélératrices courantes. Elle introduit cependant une panoplie de nouveaux problèmes tels la paramétrisation (et les problèmes de continuité associés) ainsi que la consommation mémoire (texture versus géométrie équivalente).

La simplification par images

Ayant amélioré la simplification par quadriques d'erreur de multiples façons, telles la simplification amnésique et la conservation de volume, Lindstrom et Turk [LT99, LT98] ont

[2] Un exemple classique de *matte painting* consiste à remplacer une ville entière par une image de cette ville, ce qui réduit de beaucoup les coûts de production. L'utilisation d'une image de fond (*backdrop*) lors de scènes filmées dans des studios constitue une autre forme *matte painting*.

eu l'idée d'utiliser directement des images de rendu afin de guider la simplification [LT00b].

Rappelons-nous que les techniques de simplification par fusion utilisent généralement une métrique géométrique (basée sur la position des sommets qui fusionnent) pouvant parfois considérer les attributs au sommet (tels la couleur, la normale ou la coordonnée de texture). Dans la technique de simplification par images, on calcule plutôt le coût d'une fusion en observant l'impact visuel de son application et ce, selon un nombre relativement élevé de points de vue fixés d'avance. On simule donc chacune des fusions potentielles, et on compare les images avant et après son application (en ne considérant, évidemment, que les portions d'images qui changent lors de la fusion). Un coût proportionnel à l'impact visuel (calculé simplement grâce à une erreur RMS) est associé à chaque arête et guide la simplification.

Les résultats obtenus peuvent offrir une géométrie moins près des valeurs originales, mais puisque l'on base notre simplification sur le rendu, l'apparence finale de cette géométrie sera potentiellement plus fidèle à l'apparence originale du modèle, ce qui est, de toute façon, l'objectif premier de la simplification : réduire la complexité en conservant la ressemblance.

La technique repose toutefois sur une heuristique de comparaison faible (l'erreur RMS) et, compte tenu des exigences reliées à la comparaison d'images, prend un temps considérable à calculer. Elle offre toutefois une intégration facile des textures, de l'éclairage (lequel peut même être adapté selon le modèle d'illumination), ainsi que de tous les attributs affectant l'apparence du modèle. Elle permet aussi de simplifier automatiquement l'intérieur d'un modèle, puisqu'il n'apparaît pas dans les images utilisées. Ceci souligne l'importance capitale du choix des images. Une composante du modèle pourrait n'apparaître dans aucune des images utilisées, mais être tout de même visible d'un autre point de vue. L'échantillonnage des images doit donc couvrir adéquatement toutes les parties observables du modèle, ce qui est difficile à garantir.

2.5.4 Les méthodes dépendant du point de vue

Les méthodes de simplification discutées jusqu'à présent s'appliquent principalement de façon globale au modèle. Les méthodes dépendant du point de vue tentent de simplifier inégalement un modèle en exploitant le fait qu'on ne voit généralement jamais un modèle en entier. Ces parties qui sont peu ou pas visibles peuvent être simplifiées davantage puisqu'elles sont moins importantes à l'apparence du modèle.

Xia & Varshney

Dans leur article [XV96], ensuite amélioré [XESV97], Xia *et al.* assemblent les structures de fusion dans un arbre (*merge tree*). Leur algorithme repose sur cette structure binaire dans laquelle les deux sommets enfants fusionnent en leur parent (le passage de parent à enfants correspond donc à une fission de sommets, ou encore un *vsplit*). Ils construisent cet arbre de bas en haut, en effectuant, à chacun des niveaux, le plus de fusions indépendantes possible (c'est-à-dire, des fusions qui n'affectent jamais un voisinage précédemment affecté par une autre fusion). Ils obtiennent ainsi une hiérarchie binaire, mais multiple (ne fusionnant pas en un seul parent), qu'ils peuvent ensuite manipuler en temps réel grâce à un front actif qui limite la profondeur de l'arbre. Le front actif peut s'abaisser dans l'arborescence, raffinant ainsi une portion du modèle, ou encore s'élever et la simplifier.

En considérant des critères dépendants du point de vue, on peut ajuster les détails d'un modèle afin d'offrir une qualité plus appropriée au contexte. Ainsi, considérer l'illumination locale permet de conserver les reflets lumineux (*highlights*). La projection en espace écran permet, quant à elle, de reconnaître les parties fines et de simplifier un modèle selon la distance et l'importance des détails qui le composent. Finalement, l'orientation des faces permet de simplifier les faces faisant dos (et n'étant pas visibles) et de conserver les faces de silhouette (permettant de reconnaître l'objet par son contour). L'intégration de ces critères permet de conserver des temps interactifs pour des modèles hautement détaillés (10^5 triangles).

Hoppe

Hoppe a développé sa propre technique de simplification dépendant du point de vue [Hop97, Hop98a]. Contrairement à Xia *et al.*, Hoppe commence par construire un simple maillage progressif. Ensuite, il construit sa hiérarchie de haut en bas, grâce à la liste des structures de fusions. En utilisant un critère de volume de vue plutôt que d'illumination, Hoppe réussit à simplifier les composantes situées à l'extérieur du volume de vue, aussi bien que celles situées loin de l'observateur. La technique permet d'afficher des champs d'élévation (*height fields*) de haute définition (quelques millions de triangles) en temps réel.

Hoppe n'est certes pas le seul à s'attaquer au rendu de terrains. Des travaux plus spécialisés et plus appropriés existent, notamment [LKR+96, CPS97, DWS+97, Ste97, Ste98]. Le contexte de la présente recherche étant plutôt la simplification de modèles squelettiques, nous n'allons pas en discuter davantage.

Luebke & Erikson

Bien que mentionnée précédemment, la technique d'agrégats de Luebke et Erikson [LE97] est en fait aussi une technique de simplification en fonction du point de vue. Leur arbre de sommets (*vertex tree*) constitue l'équivalent direct de l'arbre de fusion (*merge tree*) de Xia *et al.* [XV96, XESV97], à la différence que l'arbre de sommet n'est pas binaire, puisque les parents correspondent au représentant d'un agrégat. Les techniques diffèrent aussi par leur construction (des agrégations plutôt que des fusions), ainsi que par leur application (une scène entière plutôt qu'un modèle). Elles sont autrement équivalentes.

Les techniques de simplification dépendant du point de vue permettent de simplifier un modèle selon plusieurs critères (illumination, orientation, visibilité, texture, etc.). Elles demandent toutefois des calculs plus élaborés et risquent donc plus de voir leurs gains grignotés par les coûts de gestion associés. Elles offrent toutefois, grâce à leurs structures hiérarchiques, une façon de contrôler localement la simplification sur les différentes parties d'un même modèle. C'est pour cette raison qu'elles sont très appropriées pour le rendu de terrains ou tout autre modèle dont on ne voit qu'une partie à la fois.

2.5.5 Les évaluations de modèles simplifiés

La disparité des techniques de simplification, tant du point de vue de l'application (variétés, champs d'élévation, etc.) que du point de vue de la généralité (points, normales, couleurs, textures, etc.), a fait que peu de gens se sont attaqués au problème de la comparaison des modèles simplifiés. Les travaux de Cignoni *et al.* [CMS98, CRS98] sont, à notre connaissance, les seuls travaux destinés exclusivement à la comparaison de maillages simplifiés. Autrement, les lecteurs doivent se baser sur les comparaisons (plutôt informelles) éparpillées dans les divers articles décrivant chacune des méthodes. Il existe également quelques articles, écrits par diverses personnes, qui passent en revue les diverses techniques de simplification : Erikson [Eri96], Luebke [Lue97], ainsi que Garland [Gar99].

Chapitre 3

Cadre de recherche

Qui prend la mer sans savoir où il va
n'arrive jamais à bon port.

Le présent travail s'inscrit dans un cadre bien particulier. Tout d'abord, il est important de spécifier qu'il a en partie été développé grâce à un financement de recherche de la compagnie *Electronic Arts Canada*. Un chef de file dans le domaine des jeux vidéo, *Electronic Arts* a, bien évidemment, des besoins très particuliers. Qu'il s'agisse de rapidité d'exécution ou de rendu, ou même d'une consommation mémoire fort restreinte, le développement de jeux vidéo est soumis à d'importantes contraintes. Nous allons passer en revue quelques-unes de ces contraintes, de même que la plupart des solutions couramment utilisées. Nous présenterons ensuite brièvement nos solutions face à un tel contexte, et nous en discuterons plus en détail dans le prochain chapitre.

3.1 Consommation mémoire

Développant beaucoup sur des consoles de jeux (*Playstation 1* et *2*, *Nintendo 64*, etc.), *Electronic Arts* est soumis à des contraintes d'espace très strictes. Question de réduire au maximum les coûts de production de ces consoles, les fabricants réduisent souvent au mininum la mémoire (principale ou vidéo). Les développeurs de jeux pour consoles sont donc plus restreints en espace mémoire et doivent utiliser les ressources avec parcimonie.

3.1.1 Modèle squelettique

Il existe plusieurs façons d'enregistrer l'animation des modèles. Une façon simple, mais coûteuse en mémoire, consiste à sauvegarder tous les sommets du modèle pour chacune des postures de l'animation. Bien qu'efficace en termes d'utilisation (chaque nouvelle posture correspond tout simplement à sélectionner un tableau de sommets), cette méthode est très gourmande en mémoire et nettement inappropriée pour des modèles complexes et fortement animés. De plus, lorsque l'on veut appliquer la même animation à deux modèles distincts, chaque modèle doit avoir sa propre version de l'animation.

On peut également animer un modèle en transformant l'espace dans lequel il se situe. La forme la plus élémentaire consiste tout simplement à utiliser une matrice de transformation. Cette matrice homogène 4×4 permet d'effectuer des transformations simples comme des rotations et des translations, mais aussi des changements d'échelle et des cisaillements qui peuvent déformer le modèle globalement. On peut également déformer localement en utilisant des techniques de déformations d'espace comme les FFD (*free-form deformations* [Gor69, GR74, SP86, Com89, Coq90]). Il est toutefois plus intuitif de définir de telles déformations non pas en termes de déformation de l'espace, mais en termes de rotation de membres, les uns relatifs aux autres.

L'animation squelettique définit chacun des sommets d'un modèle selon un squelette. Le mouvement d'un "os" de ce squelette entraîne le déplacement des sommets qui lui sont associés. Une animation s'effectue maintenant sur les os, et la peau associée (le modèle triangulaire) est transformée afin de recouvrir le squelette. Partager un squelette indique que l'on peut partager les différentes séquences d'animation qui y sont associées. Ceci a pour effet direct de réduire les besoins en mémoire puisque la taille d'une animation (une série de matrices de transformation pour les quelques dizaines d'os) est bien faible comparée à la taille de tous les sommets d'un modèle. Ceci explique en bonne partie pourquoi l'animation squelettique constitue actuellement une technique de choix pour l'animation de personnages synthétiques. Elle comporte toutefois quelques problèmes.

3.1.2 Technique des sommets pondérés

Le problème le plus évident se situe au niveau des articulations. Lorsqu'un membre fléchit, les deux os se replient l'un sur l'autre. Ceci entraîne que la transformation rigide étire la partie externe du membre et comprime la partie interne, comme on peut remarquer à la figure 3.1(a).

Une solution simple, mais fort efficace, consiste à associer les sommets en périphérie des articulations aux deux os de l'articulation, ou même à un nombre arbitraire d'os. Chacune de ces représentations d'un même sommet est pondérée par un poids w_i ; la somme des pondérations de toutes les représentations d'un même sommet est 1. Pour calculer la position finale, il ne suffit plus de multiplier un seul sommet par une matrice. Il faut dorénavant multiplier chaque représentation par la matrice de l'os associé, et pondérer ce point grâce au poids respectif. La somme de toutes ces représentations transformées pondérées donne la position finale du sommet. Cette technique est appelée technique des sommets pondérés (*weighted vertices*), et est illustrée à la figure 3.1(b).

(a) Transformation rigide

(b) Transformation pondérée

FIG. 3.1: Comparaison entre des animations rigides et des animations pondérées (technique des *weighted vertices*). Notez comment le coude est aplati si ses sommets ne sont pas pondérés. Ces images ont été inspirées du démo technique *vertexweight* de NVIDIA.

Ainsi, la position d'un sommet $\mathbf{p}_i$ dans un modèle squelettique défini par un seul membre est :

$$\mathbf{P}_i = \mathbf{M}_i \mathbf{p}_i \tag{3.1}$$

où $\mathbf{P}_i$ est le point final en coordonnées monde, $\mathbf{M}_i$ est la matrice homogène 4×4 permettant de passer du référentiel squelette au référentiel monde, et $\mathbf{p}_i$ correspond au point en coordonnées os.

La même équation, dans un contexte de sommets pondérés, devient :

$$\mathbf{P}_i = \sum_{i=1}^{n} \mathbf{M}_i \mathbf{p}_i w_i \qquad \sum_{i=1}^{n} w_i = 1 \tag{3.2}$$

dans laquelle w_i correspond à chacun des poids d'une représentation et n est le nombre de représentations (généralement 2).

Étant donné que l'animation squelettique exige de transformer par une matrice une ou plusieurs représentations pour chacun des points du modèle, cette technique nécessite plus de temps de calcul qu'une technique où on aurait déjà tous les points déjà transformés à chaque posture. Mais ce coût est fort raisonnable, et la technique d'animation squelettique est aisément praticable en temps réel, même avec des représentations pondérées.

L'animation squelettique permet également de correctement interpoler deux postures, et ainsi d'en créer de nouvelles qui ne font pas partie de l'animation originale. Ceci repose sur le fait que le squelette est construit hiérarchiquement. On définit les transformations d'un os dans le référentiel de l'os parent. Par exemple, les transformations du poignet sont définies en fonction de l'avant-bras, et dépendent donc des transformations du coude, de l'épaule ainsi que du torse. De plus, les transformations d'un os se réduisent aux transformations rigides, et même tout simplement à une rotation (décrite par un quaternion[1]) suivie d'une translation (décrite par un vecteur). On peut donc facilement comprendre la simplicité de l'interpolation entre deux positions d'un os, et, par extension, d'un squelette en entier. Bien évidemment, ceci entraîne un coût supplémentaire, mais il est bien moindre que de tenter d'interpoler tous les points d'un modèle fixe. Ceci serait faux de toute façon puisqu'une simple interpolation linéaire de points ne tient pas compte de l'arc de mouvement du modèle.

3.2 Rapidité d'affichage

Pour améliorer les vitesses de rendu, les jeux vidéo ont souvent recours à des structures optimisées. Quelques-unes de ces structures dépendent du matériel afin d'offrir les performances désirées. Nous allons étudier les structures les plus courantes, en prenant bien soin de discuter de leur applicabilité dans le présent travail.

[1]La description d'un quaternion dépasse quelque peu le cadre de ce travail. Pour plus d'informations, nous référons le lecteur aux articles de Shoemake [Sho85, Sho87], ainsi qu'à son excellent tutoriel disponible sur le Web *ftp ://ftp.cis.upenn.edu/pub/graphics/shoemake/quatut.ps.Z*

3.2.1 Modèles triangulaires

Afin d'atteindre les performances désirées, on se doit d'utiliser le matériel (c'est-à-dire les cartes accélératrices 3D), ce qui entraîne que le format de rendu d'un modèle ne sera composé que de triangles. Le triangle constitue la primitive de choix pour le rendu, compte tenu de sa simplicité et des caractéristiques en découlant (un triangle est planaire, convexe et n'a que trois sommets). Il constitue, la majorité du temps, la seule primitive de rendu implantée de façon matérielle. Les modèles utilisés dans les jeux vidéo sont donc composés de triangles[2].

Ces propriétés des triangles expliquent qu'ils puissent être rendus rapidement. Cependant, ils doivent être présents en quantité considérable afin de pouvoir représenter des modèles relativement complexes, comme un être humain, par exemple. Cette explosion du nombre de triangles peut en partie être amortie par certains encodages judicieux.

Listes d'affichage

Les listes d'affichage (*display lists*) constituent l'une des formes les plus efficaces pour le rendu. Elles consistent à préenregistrer dans le format interne du matériel les valeurs nécessaires au rendu d'un modèle. On peut ensuite rappeler la liste (souvent par numéro) et laisser le matériel se charger complètement de l'affichage. Le gain en vitesse repose sur deux points importants : le format des données ainsi que leur emplacement.

Tout d'abord, les données sont prétransformées dans le format interne de la carte graphique. Ceci a pour effet d'éliminer complètement le coût de conversion d'un format à un autre. Ensuite, les données sont stockées non pas dans la mémoire principale de l'ordinateur, mais dans la mémoire interne de la carte vidéo. Il n'y a donc plus besoin de transférer les données sur le bus, et on soulage ainsi beaucoup ce goulot d'étranglement du pipeline de rendu.

Il est clair que, puisque les listes d'affichage résident dans la mémoire de la carte accélératrice, leur taille est limitée par la mémoire de celle-ci. On peut donc plus facilement excéder la capacité maximale puisque l'on dispose généralement de moins de mémoire que celle de l'ordinateur. Mais les listes d'affichage ont un désavantage encore plus important : elle ne peuvent pas être modifiées. Afin d'être le plus optimisé possible, elles n'ont pas été conçues pour être changées. La seule façon de changer une liste est de la remplacer, ce qui revient à ne pas les utiliser si on le fait à chaque fois. Pour cette raison, les listes d'affichage ne peuvent of-

[2]Il est possible qu'un modèle soit construit à partir de primitives plus évoluées (telles les surfaces paramétriques), mais le modèle final, utilisé dans le jeu, sera triangulé.

frir de gain pour les modèles dynamiques, et s'appliquent donc mal aux modèles squelettiques de notre cadre de recherche.

Les modes différés

Les modes différés constituent une autre façon de réduire la circulation sur le bus. S'attaquant aux coûts reliés à transférer différents types d'informations (sommet, normale, couleur, coordonnée de texture, etc.), ils les regroupent ensemble dans un tableau. Ce tableau, étant contigu en mémoire, peut être transféré plus efficacement à la carte vidéo par un accès mémoire direct (DMA, pour *direct memory access*). Ces techniques constituent la meilleure alternative aux listes d'affichage et sont couramment utilisées dans les libraries graphiques actuelles (les *vertex arrays* sous *OpenGL* et le *retained mode* sous *DirectX*).

Les encodages de faces

Puisque le bus est un goulot d'étranglement important du pipeline graphique, les gens ont dû développer plusieurs techniques pour réduire l'utilisation de cette étroite route vers la carte accélératrice. Afin de réduire le nombre de sommets à envoyer sur le bus, il existe quelques encodages géométriques bien connus.

1. Quadrilatères et polygones

   Le premier encodage, s'il en est un, est celui des quadrilatères et des polygones. Sachant que tout polygone est finalement triangulé avant d'être rendu, une implantation pourrait décider de n'envoyer que des triangles à la carte. Un polygone à n côtés devrait donc être remplacé par $n - 2$ triangles. Ainsi, n'envoyer que n sommets offre un gain en comparaison d'envoyer $3(n - 2)$ sommets pour les triangles. Cependant, la triangulation de polygones n'est pas triviale. En effet, les sommets spécifiés ne sont peut-être même pas coplanaires, et la face définie n'est pas restreinte à être convexe. Pour ces raisons, l'utilisation de polygones, et parfois même de quadrilatères, entraîne une chute de performance considérable. Heureusement, on peut tout trianguler initialement, et ne travailler ensuite qu'avec des triangles. Il existe d'ailleurs quelques encodages destinés spécifiquement aux modèles triangulaires.

2. Éventails et paravents

   Lorsque des triangles forment un maillage connecté, ils partagent nécessairement des sommets. Prenons, par exemple, un hexagone composé de triangles reliés à un point

central (voir figure 3.2). Une approche naïve dessinerait ce maillage en utilisant $6 \times 3 = 18$ sommets. Mais on pourrait aussi spécifier une seule fois le sommet central, et l'utiliser dans tous les triangles subséquents (voir figure 3.2(b)). En spécifiant un éventail de triangles (*triangle fan*), on n'utilise plus que trois sommets pour le premier triangle, et un seul pour chaque triangle subséquent. Ceci exige toutefois de pouvoir conserver deux sommets en mémoire, mais c'est peu en comparaison des gains que cela peut amener. Les éventails, étant définis autour d'un point, sont très locaux, et sont donc limités dans la superficie maximale qu'ils peuvent couvrir sur le modèle. Pour encoder une série de triangles parcourant la surface du modèle, il faut utiliser un autre encodage.

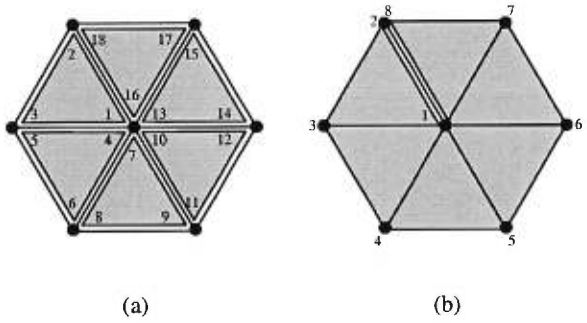


FIG. 3.2: Exemple montrant comment l'utilisation d'éventails permet d'épargner le nombre de sommets à spécifier pour un groupe de triangles.

Cet autre encodage construit des paravents de triangles (*triangle strips*). La figure 3.3 montre plus clairement l'encodage d'un éventail dans le cas d'une bande de triangles. Un paravent est plus longitudinal, alors qu'un éventail est radial. Ceci permet aux paravents de pouvoir recouvrir une plus grande superficie, pouvant même, pour certains modèles, recouvrir totalement la surface. Les travaux d'Evans *et al.* [ESV96] discutent plus en détail de la construction de paravents.

L'utilisation des éventails et des paravents nécessite toutefois de conserver tous les triangles qui les composent. Ils s'appliquent donc mal aux maillages progressifs. En effet, enlever arbitrairement un triangle dans le modèle brise l'encodage de triangles (un éventail ou un paravent) qui l'utilise. Puisque l'opération de fusion de sommets (*edge collapse*) enlève justement des faces, l'utilisation de maillages progressifs est proscrit si l'on veut conserver de tels encodages. Puisque les paravents sont deux fois plus rapides que d'envoyer les triangles séparément, on doit simplifier de moitié avant d'avoir un gain dans le temps de rendu. En plus, pour un même

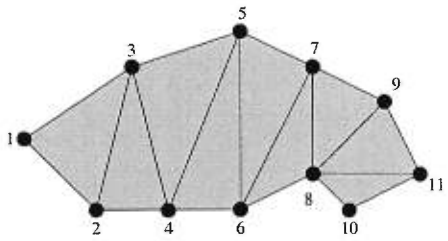


FIG. 3.3: Exemple montrant un encodage de paravents de triangles.

temps, on a maintenant une qualité bien moindre que d'utiliser le modèle original. On doit donc simplifier davantage le modèle original avant d'avoir un gain valable.

3.2.2 Synthèse

Toutes ces techniques améliorent la vitesse de rendu, et c'est pour cela qu'elles sont constamment utilisées dans les jeux vidéo. Les listes d'affichage sont toutefois trop strictes pour les modèles dynamiques comme les modèles squelettiques. Les encodages de faces utilisés dans un mode différé constituent la façon courante de tirer profit au maximum des capacités graphiques actuelles, et c'est pour cela que nous avons jugé bon de les permettre dans notre travail.

3.3 Surgissements de transition

Compte tenu du coût relié à l'utilisation des maillages progressifs, les développeurs de jeux vidéo ont plutôt l'habitude de construire différents niveaux de détails fixes parmi lesquels ils choisissent le plus approprié pour le rendu (cette technique est très proche de la technique de Crow [Cro82] décrite précédemment). Nonobstant le fait que chaque niveau de détails doit être construit à la main par un artiste, cette approche est l'une des plus simples à implanter, et permet d'offrir des gains appréciables. Elle a cependant le désavantage d'introduire du surgissement.

Ce surgissement (*popping*) survient lorsque l'on effectue une transition entre les différents modèles. Ces transitions, définies typiquement grâce à la distance à la caméra, peuvent être amplifiées par répétition excessive lorsque le modèle oscille autour de ce seuil de distance. Puisque les modèles diffèrent souvent considérablement en nombre de triangles, on aperçoit typiquement des changements notoires d'apparence. Ces changements, qu'il s'agisse de changements de silhouette, d'illumination ou simplement de détails, sont suffisamment importants

pour que la transition soit clairement visible.

Une façon courante de réduire le problème consiste à faire graduellement disparaître un niveau de détails et de le remplacer progressivement par un autre. Cette technique, utilisant les capacités matérielles de transparence (*alpha blending*), introduit de nouveaux problèmes d'ordonnancement de polygones. En effet, pour être rendues correctement, les faces transparentes doivent être dessinées de derrière à devant, ce qui est coûteux à ordonner, voire même impossible (lorsque les faces s'intersectent ou dans des cas limites comme à la figure 3.4). De plus, cette technique ne fait que superposer les modèles, et une variation de silhouette, par exemple, ne serait que faiblement remplacée, et non déplacée comme on s'y attendrait (la figure 5.1 donne un exemple de ce que nous entendons). Finalement, cette technique se retrouve effectivement à faire le rendu de deux modèles, ce qui entraîne un coût supplémentaire lors de la transition.

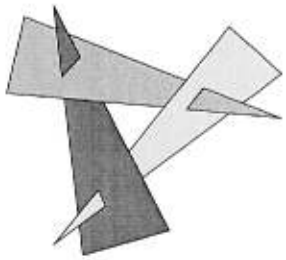


FIG. 3.4: Trois triangles ne pouvant pas être ordonnés de derrière à devant.

3.4 Solution proposée

3.4.1 Démarche

Nous avons commencé par évaluer les besoins ainsi que les problèmes reliés à diverses solutions. Voici une synthèse de nos réflexions et des différentes avenues envisagées.

Modèles triangulaires

Puisque l'étape de rendu s'effectue à l'aide de triangles, nous avons décidé de développer une technique de simplification de maillages triangulaires. Puisque tout polygone peut être triangulé, et que toute surface paramétrique peut être approximée grâce à des triangles, cette décision nous limite peu.

Nous avons opté pour une technique de simplification par fusion, puisque la fusion de sommets nous semble être l'opération la plus délicate qui puisse simplifier une surface triangulaire. C'est donc dans l'esprit des maillages progressifs de Hoppe [Hop96] (section 2.3.2) que nous avons développé notre technique.

Ne voulant toutefois pas restreindre les modèles à être uniquement des variétés 3D, nous avons dû généraliser la fusion de sommets afin d'accepter des cas plus particuliers, pouvant enlever plus de deux faces, par exemple. Cette approche, analogue à celle de Garland et Heckbert [GH97], sera décrite plus en détail dans le chapitre 4.

Modèles articulés

Puisque les modèles sont articulés, nous devons obtenir un modèle dont la simplification peut elle aussi être articulée. Comme nous l'avons mentionné précédemment, recalculer complètement la simplification à chaque fois est hors de question. Il devient donc évident que le modèle simplifié doit demeurer articulé.

On effectue la simplification selon une certaine métrique. Une métrique est une valeur scalaire représentant un coût. Ce coût dépend de diverses valeurs : longueur, aire, volume, etc. Pour que ces métriques de simplification concordent, les calculs doivent se faire dans un référentiel commun. Autrement, les métriques associées aux faces reliant deux os n'ont plus aucun sens.

Une première solution serait de simplifier chaque os séparément. Ceci implique cependant que les régions autour des articulations ne pourront pas être simplifiées autant, et de fortes discontinuités apparaîtront. Une simplification majeure du modèle conservera ainsi une forte polygonisation autour des articulations, et le modèle ne pourra jamais être réduit en deçà du nombre de polygones de suture (polygones s'étendant sur plus d'un os).

Nous avons donc décidé d'effectuer une simplification sur un modèle statique (en utilisant une pose unique de l'animation) et de répercuter la simplification dans le modèle squelettique. Cette approche a l'avantage de permettre une simplification complète (dégénérant tous les triangles) sans pour autant empêcher le modèle d'être animé comme avant. La qualité de la simplification dépend malheureusement du choix de la posture. Nous allons discuter plus en détail de ces implications dans la section 4.4.2.

Gestion rapide

La gestion de la représentation simplifiée du modèle devait également être rapide. Même si les techniques de simplification sont de plus en plus rapides (quelques secondes à peine pour un modèle composé de dizaines de milliers de triangles), elles ne peuvent pas encore être effectuées en temps réel. On doit donc effectuer une simplification en précalcul, et ensuite, gérer la structure résultante en temps réel (ce qui est possible avec les maillages progressifs, par exemple).

C'est cette contrainte de temps de gestion qui nous a éloignés des techniques dépendant du point de vue. Croyant initialement pouvoir exploiter la direction de la vue dans notre technique, nous en sommes venus à croire que les coûts de gestion associés, tant au niveau de la mémoire qu'au niveau du temps de calcul, seraient encore un peu trop exigeants compte tenu des ressources actuelles. Il ne serait toutefois pas surprenant de voir de telles techniques apparaître au fur et à mesure que les limites matérielles seront repoussées.

Faible empreinte mémoire

Compte tenu que la mémoire peut potentiellement être une ressource fortement limitée, les structures nécessaires à la simplification doivent être raisonnablement compactes. Mais conscients de la croissance relativement récente, nous ne nous sommes pas limités outre mesure.

La technique de simplification devait donc s'appliquer à un modèle triangulaire, mais devait également être relativement simple et compacte en mémoire afin de pouvoir être affichée rapidement.

3.4.2 Notre technique

Nous avons évalué la possibilité d'utiliser un modèle multi-résolution comparable aux maillages progressifs de Hoppe (voir section 2.3.2). Nous avons ainsi construit un maillage progressif articulé permettant de manipuler les structures de fusion pour avoir accès à une multitude de modèles. Une telle représentation comporte toutefois quelques désavantages. Tout d'abord, le modèle doit gérer les faces actives, et cette gestion empêche d'utiliser des encodages de faces tels les éventails et les paravents. De plus, le sous-ensemble des sommets non

utilisés par le modèle est difficile à connaître, essentiellement parce qu'il est éparpillé parmi tous les sommets. Il est donc plus rapide de transformer l'ensemble des sommets, même si on n'en utilise qu'un faible sous-ensemble avec les faces actives. Compte tenu de ces deux points, l'utilisation d'un maillage progressif articulé offre très peu de gains en termes de performance : la transformation du modèle squelettique en modèle statique est aussi coûteuse qu'avant, peu importe la complexité du modèle simplifié.

Il devenait donc préférable d'utiliser notre technique uniquement pour générer automatiquement un nombre relativement petit de modèles simplifiés. Contrairement au maillage progressif articulé, ces modèles squelettiques simplifiés offrent un gain en performance puisqu'ils sont de plus en plus petits en taille. Les informations de fusion (les structures d'*edge collapse*) nous donnent toutefois des informations supplémentaires sur l'origine des sommets dans un modèle simplifié. Il serait dommage de perdre bêtement ces informations.

Nous avons ainsi décidé de conserver ces informations, et de construire une table de correspondances de sommets entre chacun des modèles simplifiés. Ceci permet, avec peu de mémoire, d'effectuer du morphisme entre les différents niveaux de détails. Ces structures de morphisme permettent de passer linéairement d'un modèle haute résolution à un modèle basse résolution, et cela, tout en douceur. Ainsi, les sommets du modèle haute résolution se déplacent doucement jusqu'à leur position dans le modèle basse résolution, ce qui entraîne la dégénération des faces qui ne sont présentes que dans le modèle le plus défini. Lorsque la transition est complétée, il suffit dorénavant d'utiliser le modèle le plus simplifié. On peut même construire un nombre arbitraire de modèles simplifiés et les gérer ainsi par paire.

Cette approche comporte plusieurs avantages justifiant sa pertinence. Tout d'abord, chaque niveau de détails peut être optimisé pour l'affichage. Ceci implique que chaque modèle peut être encodé à l'aide d'éventails et de paravents, et cela, indépendamment des autres modèles. Ces bandes encodées demeurent tout à fait valides, même lors du morphisme, puisque les faces ne sont que dégénérées.

Ensuite, cette technique demeure très souple. Nous entendons par là que l'utilisation du morphisme n'est pas du tout obligatoire : elle est optionnelle. Ainsi, un contexte trop restreint en temps pourrait retirer le morphisme complètement, ou l'utiliser seulement lorsque les ressources le permettent.

Les structures nécessaires au morphisme sont très simples : une simple table de correspondances. Dans notre implantation, il ne s'agit que d'indices, et leur nombre est linéaire avec la

taille du modèle le plus complexe.

Finalement, la technique est peu coûteuse. Lorsque le morphisme n'est pas utilisé, on obtient tout simplement des modèles squelettiques simplifiés, parmi lesquels on peut choisir : il n'y a alors aucun coût supplémentaire. L'utilisation du morphisme demande toutefois de transformer les deux modèles squelettiques que l'on combine. Cette approche entraîne évidemment un peu plus de calcul, mais ce calcul est fort comparable au calcul nécessaire pour la technique de transition par transparence couramment utilisée, et offre une transition d'une qualité fort supérieure.

3.5 Cadre matériel

Nous avons développé nos travaux principalement sur des ordinateurs personnels. Notre implantation a été codée en C++, et nous avons découplé le code d'interface du code de gestion. Le rendu a été fait grâce à la librairie graphique OpenGL (ou Mesa, qui constitue une implantation logicielle équivalente). L'essentiel des travaux a été effectué sous Linux, avec les outils de développement traditionnels, mais il est à noter que des versions intermédiaires fonctionnant sous IRIX, Windows et MacOS ont également été développées. Tous les tests ont été effectués sur un ordinateur équipé d'un processeur Athlon cadencé à 600 MHz, disposant de 256 Mo de mémoire principale et roulant sous Linux. L'affichage a été accéléré grâce à une carte GeForce256 de NVIDIA, munie de 32 Mo de mémoire vidéo.

Nous allons maintenant décrire les différents aspects de notre technique. Nous allons débuter par la simplification, ainsi que les points particuliers en faisant partie. Nous allons ensuite décrire comment nous créons et utilisons les structures reliées au morphisme. Nous offrirons finalement quelques résultats et discuterons des extensions que nous pourrions apporter à la technique.

Chapitre 4

Simplification

Things should be made as simple as possible,
but no simpler.

Albert Einstein

4.1 Description

La simplification de maillage s'effectue en réduisant le nombre de triangles dans un modèle tout en tentant de conserver son apparence globale. Pour ce faire, nous utilisons une technique de simplification par fusion inspirée des maillages progressifs (section 2.3.2).

Notre simplification s'opère donc en deux parties : une étape de précalcul, pouvant prendre un temps arbitrairement long, ainsi qu'une manipulation des structures créées, effectuée lors de l'exécution. Le précalcul nous permet d'effectuer à l'avance des opérations qui s'avéreraient trop coûteuses à l'exécution. Bien qu'il soit très court (quelques secondes à peine), ce précalcul demeure quand même trop lent pour être praticable en temps réel (où le rafraîchissement s'effectue typiquement à 30 Hz). C'est lors l'exécution que les structures résultantes sont gérées pour modifier et afficher le modèle. Seule cette étape (de même que le morphisme, chapitre 5) a besoin d'être effectuée en temps réel.

4.1.1 Fusion et fission de sommets

L'opération de base de notre technique de simplification est une fusion de sommets (*edge collapse*). Elle consiste à fusionner deux sommets voisins en un seul et à enlever toutes les

faces qui s'en retrouvent dégénérées en un segment. En conservant dans l'ordre toutes les modifications ainsi apportées au modèle 3D, on arrive à pouvoir inverser la fusion. Ces fissions de sommets (*vertex splits*) appliquées dans l'ordre inverse de leur création permettent de reconstruire exactement le modèle original.

La figure 4.1 montre une fusion de deux sommets. Les valeurs A et B constituent les indices des sommets contenus dans un tableau contigu global. Les points $\mathbf{P}_A$ et $\mathbf{P}_B$ sont les positions géométriques originales et $\mathbf{P}_A{}'$ constitue la position finale suite à la fusion de B en A. On peut séparer les faces voisines de A ou B en trois groupes : les faces ne touchant qu'à A, les faces ne touchant qu'à B et les faces touchant aux deux à la fois. Ce dernier groupe contient les faces qui dégénèrent et qui doivent être retirées du modèle afin de véritablement simplifier le modèle. Les faces n'utilisant que B doivent maintenant plutôt utiliser A. Celles utilisant A demeurent valides : c'est en déplaçant $\mathbf{P}_A$ que les faces seront modifiées.

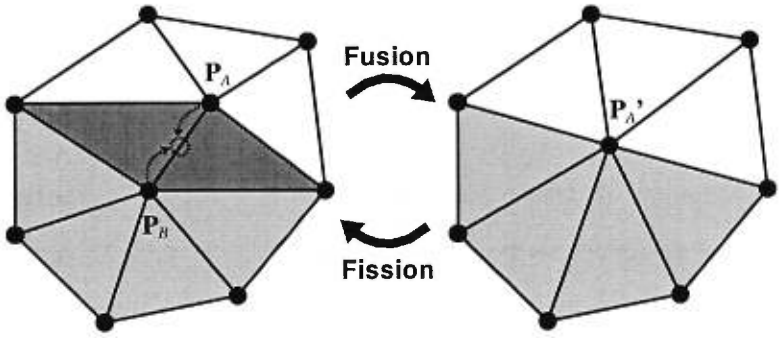


FIG. 4.1: Exemple de fusion et de fission de sommets.

Lors de la fission, on doit ajouter les faces partageant A et B. On doit aussi parcourir toutes les faces qui étaient exclusives à B pour changer leur sommet A de façon à ce qu'elles soient déconnectées de A et reconnectées à B. Afin de ramener la position du sommet A de $\mathbf{P}_A{}'$ à $\mathbf{P}_A$, on doit tout simplement appliquer le vecteur inverse à celui utilisé initialement pour déplacer $\mathbf{P}_A$ vers $\mathbf{P}_A{}'$.

On devrait normalement aussi déplacer $\mathbf{P}_B$ de la même façon, mais puisque nous n'avons pas l'intention de sauvegarder le modèle multi-résolution sur disque, nous profitons du fait que $\mathbf{P}_B$ demeure en mémoire même lorsqu'il n'est pas utilisé. Ainsi, lorsque les faces du bas changent A pour B (lors de la fission), elles utilisent alors $\mathbf{P}_B$ qui était toujours résidant en

mémoire.

4.1.2 Conservation des coins

Un coin (*wedge*) est une abstraction introduite par Hoppe [Hop96]. Plutôt que de définir un triangle grâce à trois groupes de positions, normales, couleurs, etc., on peut plutôt définir un triangle à l'aide de trois coins. Chaque coin correspond à une paire position-attributs. Tous les coins autour d'un sommet ont le même indice de position. La différence se situe au niveau des attributs : les coins peuvent partager ou non les attributs. Une région est dite lisse (*smooth*) lorsque les faces adjacentes partagent les mêmes attributs. Par contre, une région non lisse offrira des discontinuités apparentes.

La figure 4.2 montre un exemple d'utilisation de coins. Puisque les deux triangles d'une face du cube ont la même normale, ils partageront le même coin autour d'un sommet. Ainsi, comme on peut le voir autour du sommet, on a trois paires de triangles, chacune ayant les mêmes attributs. Chacune de ces paires utilise un seul et même coin autour du sommet. On a donc six triangles partageant trois coins et un seul sommet.

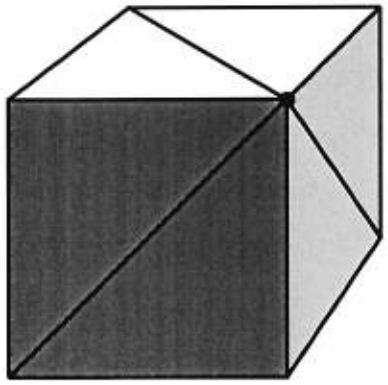


FIG. 4.2: Exemple de coins autour d'un sommet. Autour du point noir, cet exemple utilise trois coins (chacun partagé par deux triangles) et un seul sommet.

Lors de la fusion de sommets, on doit gérer ces coins de façon appropriée. Autrement, chaque fusion entraînerait l'introduction de discontinuités de plus en plus apparentes : un modèle aura donc tôt fait de perdre son apparence lisse. Nous avons donc décidé d'analyser la continuité des faces qui dégénèrent. Lorsqu'une face dégénère, elle approche typiquement deux paires de coins ensemble, à raison d'une paire par arête se rapprochant (voir figure 4.3). Il est raisonnable de conserver lisse l'unification des coins voisins si les deux paires étaient

initialement lisses. Cette approche est la même que celle de Hoppe [Hop96, Hop98b, Hop99].

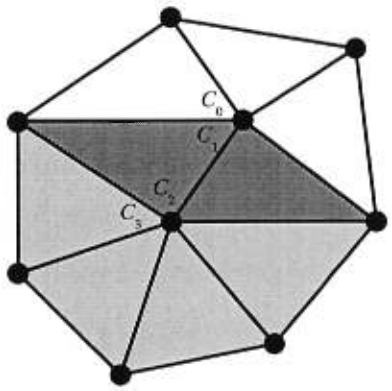


FIG. 4.3: Si les coins C_0 et C_1 sont lisses entre eux, et que les coins C_2 et C_3 le sont aussi, alors C_0 et C_3 devraient également l'être suite à la fusion. Nous calculons la valeur des attributs en considérant toutes les faces partageant le même coin après la fusion.

4.2 Algorithme

La simplification par fusion consiste à associer un coût à chaque arête et à itérativement effectuer la fusion des sommets reliés par l'arête la moins dispendieuse. En prenant soin de mettre à jour le coût des arêtes modifiées lors de la simplification, on réussit à tenir compte de l'impact des différentes fusions passées ainsi que de la configuration courante du modèle. On peut de cette façon adapter la simplification selon les transformations qu'elle opère sur le modèle. Pour des raisons d'efficacité, nous enregistrons les arêtes dans un monceau (*heap*), permettant d'efficacement retirer et mettre à jour les éléments.

4.2.1 Construction du monceau

De façon similaire à Garland et Heckbert [GH97], nous conservons les arêtes dans un monceau. Cette structure de données partiellement ordonnée permet de conserver à la racine l'arête ayant le coût le plus faible, et de la retirer efficacement. Le monceau a également l'avantage d'être un arbre binaire équilibré, ce qui améliore la vitesse des mises à jour.

Nous commençons évidemment par remplir le monceau avec toutes les arêtes contenues dans le modèle original. Nous devons toutefois prendre soin de ne pas avoir deux arêtes équivalentes, question de ne pas fusionner deux fois les mêmes sommets. Ceci vient du fait

que la fusion de A à B est équivalente, en termes de résultats, à la fusion de B à A. Afin d'éviter de tels cas, nous construisons pour chaque sommet la liste des sommets voisins, lesquels nous sont donnés par les informations des faces. Cette liste de sommets est construite de façon à n'avoir aucune répétition, et permet donc de déterminer toutes les arêtes touchant à un sommet en particulier. Nous construisons donc le monceau grâce à ces arêtes en n'y insérant que les arêtes dont $A < B$.

Puisque la fusion de deux sommets entraîne une mise à jour des arêtes touchant le sommet résultant, on doit aussi permettre de rapidement mettre à jour nos arêtes dans le monceau. Afin d'éviter une fastidieuse recherche en profondeur, nous avons jumelé le monceau à un tableau de correspondances. Ce tableau permet de conserver la position de toutes les arêtes dans le monceau de façon à pouvoir y accéder instantanément. Ce tableau étant intimement lié au monceau, nous avons dû redéfinir les opérateurs de mise à jour au niveau du monceau de façon à rafraîchir aussi les valeurs de cette structure associée. Ainsi, lorsqu'un élément du monceau est déplacé, sa position est mise à jour dans le tableau de correspondances. La structure résultante permet d'obtenir l'arête la moins coûteuse ainsi que de mettre à jour n'importe quelle arête de façon fort efficace.

4.2.2 Fusion itérative

L'algorithme repose sur un fait simple : tant qu'il reste des arêtes dans le monceau, on peut encore fusionner deux sommets. On retire alors la racine du monceau, laquelle contient l'arête la moins coûteuse, et on effectue la fusion de ses deux sommets. On remplit donc la structure de fusion tel que décrit à la section 4.1.1. On modifie le modèle (position de $\mathbf{P}_A$, valeurs des attributs, faces à enlever ou altérer) pour qu'il subisse la fusion et on met à jour dans le monceau les arêtes affectées. Puisque certaines arêtes ne sont plus valides, notamment celles des faces dégénérant et qui touchaient à B, on doit les retirer du monceau.

Notre implantation permet de choisir différentes formes de coûts d'arêtes, appelés *métriques de simplification* (voir section 4.3 pour leur description). Dépendant du choix de la métrique, il se peut que l'on doive mettre à jour des structures supplémentaires. Si tel est le cas, ce travail supplémentaire doit être effectué lors de chacune des fusions (pour des mises à jour particulières), mais parfois même avant d'avoir commencé et après avoir fini (pour l'allocation, l'initialisation et la désallocation de ces structures additionnelles).

4.2.3 Calcul des attributs

On doit aussi recalculer les attributs des faces de façon à ce qu'ils soient plus représentatifs du modèle dans son état actuel. Nous utilisons les coins afin de guider le calcul des nouveaux attributs. Pour chacun des deux sommets fusionnant ensemble, nous construisons la liste de toutes les faces partageant le même coin. Nous calculons ensuite, pour chaque groupe de faces, un attribut moyen tenant compte de toutes les faces. Pour le calcul de la normale, par exemple, nous calculons une normale moyenne pondérée par l'angle d'ouverture de la pointe du triangle.

4.3 Métriques

Notre implantation permet de sélectionner parmi plusieurs métriques celle que l'on désire utiliser. Nous allons décrire chacune de ces métriques en décrivant, pour chacune, ses forces et ses faiblesses. Elles sont présentées en ordre croissant de temps de calcul et sont accompagnées d'images montrant différents résultats. Tous les résultats ont été recueillis sur un Athlon 600 MHz équipé de 256 Mo de mémoire, dont l'affichage est accéléré par une carte accélératrice GeForce256. Les temps obtenus pour chaque métrique correspondent au temps moyen calculé grâce à dix répétitions de la simplification. Le modèle original, comportant plus de 13 000 faces, est présenté à la figure 4.4.

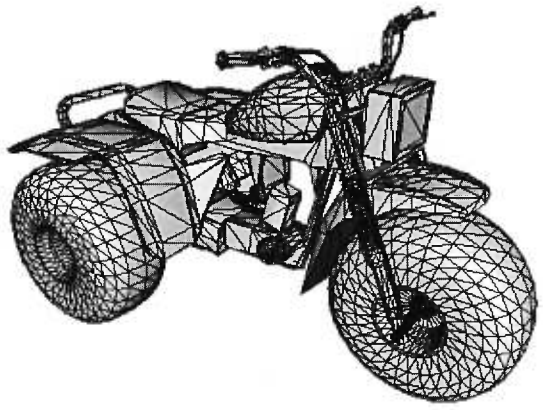
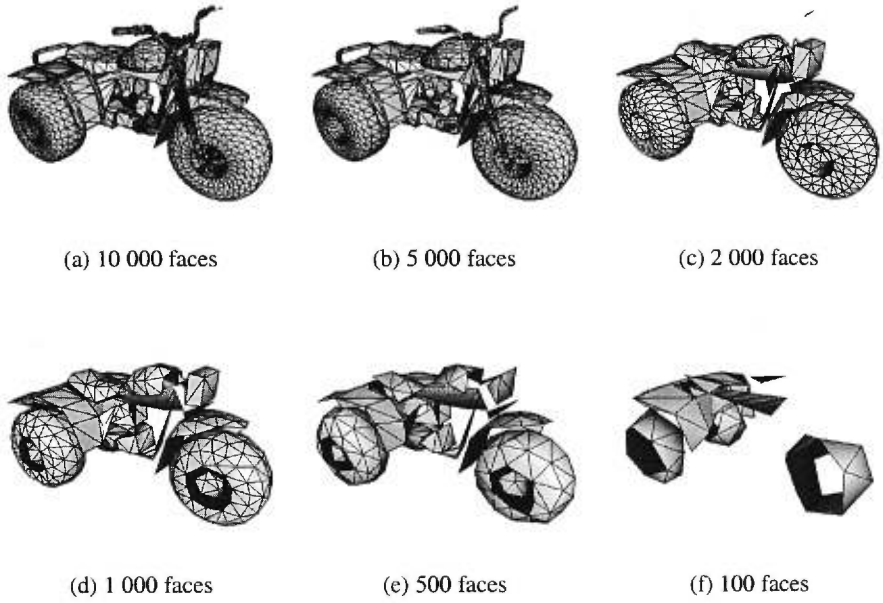


FIG. 4.4: Modèle original comportant 13 594 triangles.

4.3.1 SEM

La métrique de l'arête la plus courte (SEM, pour *Shortest Edge Metric*) constitue la métrique la plus élémentaire. Elle utilise simplement la longueur[1] des arêtes afin de toujours fusionner les deux sommets reliés les plus rapprochés. En prenant le point milieu de l'arête comme position finale du sommet résultant, on réussit à obtenir des résultats d'une qualité surprenante (figure 4.5) pour une approche aussi simpliste.



(a) 10 000 faces (b) 5 000 faces (c) 2 000 faces

(d) 1 000 faces (e) 500 faces (f) 100 faces

FIG. 4.5: Exemples de simplification par arête la plus courte. La simplification a pris en moyenne 626 ms à calculer. Cette technique simplifie d'abord les petites parties d'un modèle, ce qui peut poser des problèmes, comme le guidon qui dégénère avant la poignée (c).

Compte tenu de sa nature, la métrique des arêtes les plus courtes a tendance à enlever hâtivement les détails. Elle ne tient cependant pas compte du voisinage de l'arête, et peut donc entraîner de fortes déformations. C'est pour cette raison que nous avons considéré l'utilisation des quadriques d'erreur.

[1] Pour des raisons d'efficacité, nous avons plutôt utilisé la longueur au carré, évitant ainsi le calcul d'un radical, de toute façon inutile puisque l'ordre est le même.

4.3.2 QEM

Les quadriques d'erreur (QEM, pour *Quadric Error Metric*) ont été introduites par Garland et Heckbert [GH97] afin d'obtenir rapidement et avec peu de mémoire une simplification de qualité. Commençons tout d'abord par décrire ce que sont les quadriques d'erreur.

Description des quadriques

Une quadrique peut se noter $(\mathbf{A}, \mathbf{b}, c)$, où $\mathbf{A}$ est une matrice symétrique 3×3, $\mathbf{b}$ est un vecteur de dimension 3 et c est un scalaire. Une quadrique a besoin de 10 valeurs, et pourrait aussi être représentée par sa forme homogène, c'est-à-dire une matrice symétrique 4×4. Elle peut être utilisée afin de représenter un plan. La distance au carré d^2 d'un point $\mathbf{p}$ à ce plan peut être évaluée à l'aide de l'équation suivante :

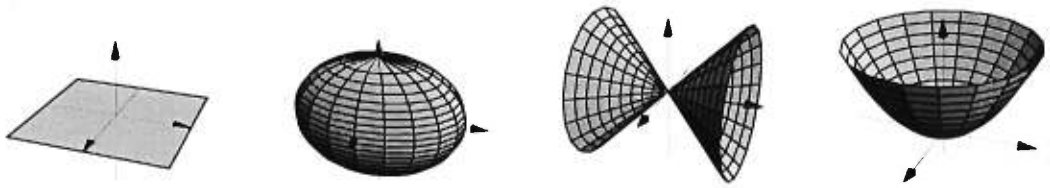
$$d^2 = Q(\mathbf{p}) = \mathbf{p}^\mathsf{T}\mathbf{A}\mathbf{p} + 2\mathbf{b}^\mathsf{T}\mathbf{p} + c. \tag{4.1}$$

L'avantage des quadriques se clarifie lorsqu'on les combine, par une simple addition terme à terme des composantes $\mathbf{A}$, $\mathbf{b}$ et c. La quadrique résultante tient compte des deux plans utilisés, et définit maintenant non plus un plan, mais une surface quadrique (voir figure 4.6). La combinaison des multiples quadriques peut donc servir à définir des surfaces quadriques, d'où l'origine de leur nom.

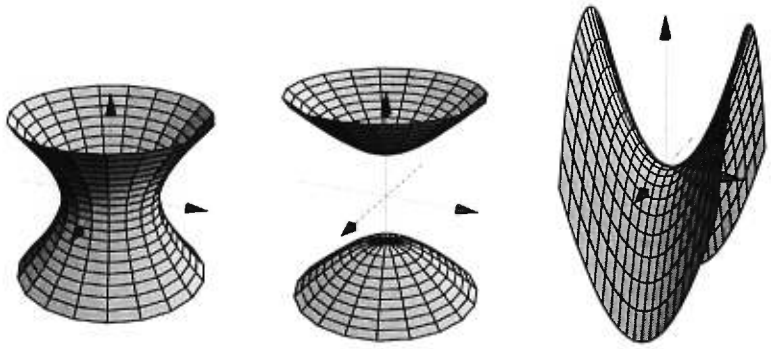
L'utilisation d'une quadrique d'erreur permet également de déterminer la position géométrique minimisant l'erreur qu'elle définit (équation 4.1). Pour ce faire, on doit solutionner $\mathbf{p}_{opt} = -\mathbf{A}^{-1}\mathbf{b}$, et l'équation de distance se simplifie en $Q(\mathbf{p}_{opt}) = -\mathbf{b}^\mathsf{T}\mathbf{A}^{-1}\mathbf{b} + c$ (voir les travaux de Garland et Heckbert [GH97, GH98] pour plus de détails). Ceci nécessite toutefois d'inverser $\mathbf{A}$, ce qui est impossible lorsque le déterminant est nul (indiquant un nombre infini de solutions). Dans de telles circonstances, on peut plutôt évaluer quelques positions et choisir celle offrant le meilleur coût lorsque évaluée avec l'équation 4.1. Cette approche permet en quelque sorte à la simplification par quadriques d'erreur d'optimiser la position des sommets qui fusionnent.

Simplification

On commence par associer à chaque sommet la somme des quadriques définies par les plans de support de ses faces adjacentes. Lorsque vient le temps de calculer le coût d'une arête, on



(a) Plan (b) Ellipsoïde (c) Cône (d) Paraboloïde

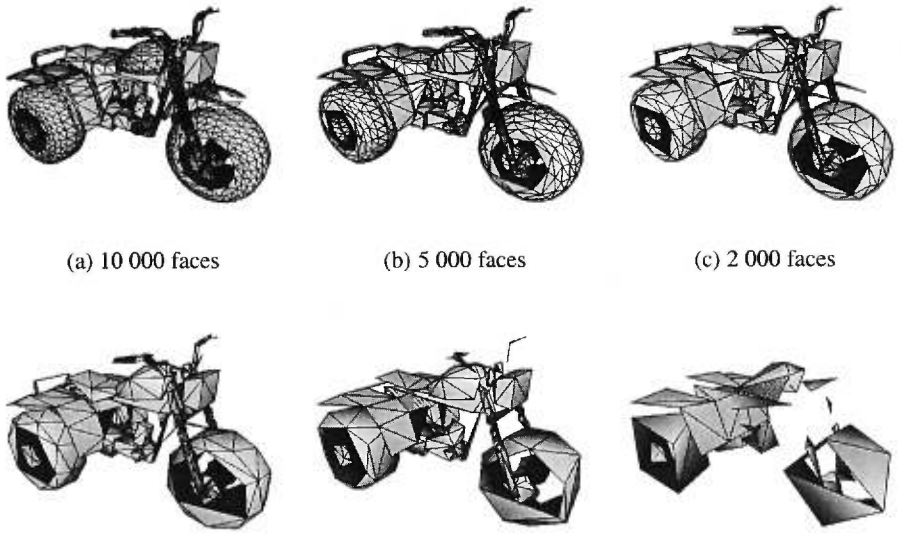


(e) Hyperboloïdes (f) Paraboloïde hyperbolique

FIG. 4.6: Quelques surfaces quadriques.

évalue tout simplement la somme des quadriques des deux sommets de l'arête. Lorsqu'un point B fusionne en A, on additionnera tout simplement la quadrique de B à celle de A, autrement dit $Q'_A = Q_A + Q_B$. Ceci entraîne cependant que lorsqu'une face est partagée entre A et B, alors sa quadrique, qui était contenue dans Q_A aussi bien que dans Q_B, sera dorénavant comptée deux fois dans la nouvelle Q'_A. Bien que ceci introduise quelques erreurs, les résultats demeurent de très haute qualité, et sont obtenus beaucoup plus efficacement que si on devait gérer par inclusion/exclusion un historique de plans (comparable à Ronfard et Rossignac [RR96]).

Les quadriques d'erreur offrent d'excellents résultats et ce, dans des temps très rapides. La figure 4.7 illustre jusqu'à quel point les quadriques d'erreur ont tendance à mieux conserver toutes les parties d'un modèle. Le guidon, par exemple, demeure plus longtemps que dans la technique de l'arête la plus courte (figure 4.5).



(a) 10 000 faces (b) 5 000 faces (c) 2 000 faces

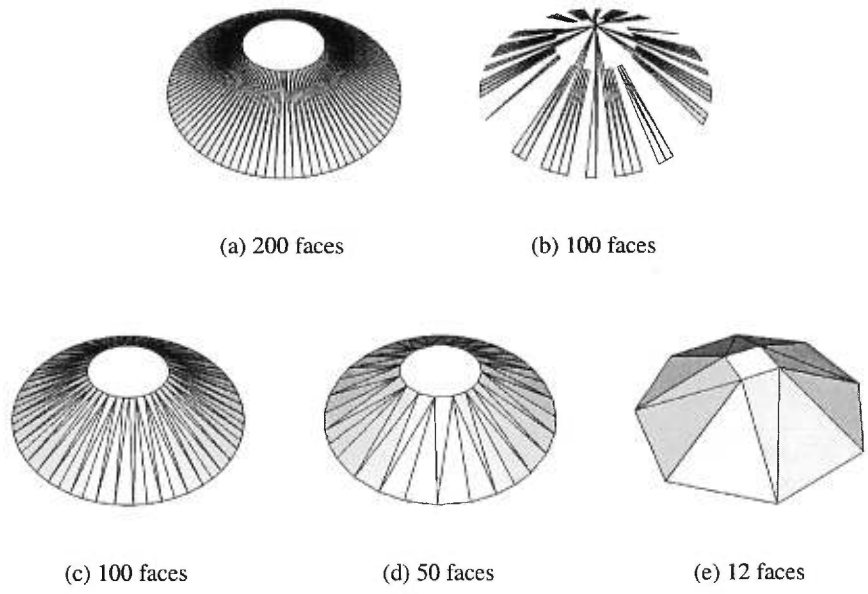
(d) 1 000 faces (e) 500 faces (f) 100 faces

FIG. 4.7: Exemples de simplification par quadriques d'erreur. Le temps moyen de simplification est de 809 ms. Cette métrique, contrairement à l'arête la plus courte, tente de conserver les discontinuités entre les faces. Ainsi, on peut simplifier davantage tout en conservant une fidélité plus grande au modèle.

4.3.3 QEM contrainte

Un des problèmes de la simplification par quadriques d'erreur se situe au niveau des triangles de bordure : le contour d'un modèle est généralement très mal conservé à cause du

placement optimal des quadriques. Dans l'exemple illustré à la figure 4.8, on remarque que les plans de support de tous les triangles s'intersectent en un seul point. Ce point constitue donc la position optimale suite à une fusion. La figure 4.8(b) montre clairement que l'approche classique introduit d'importants artefacts. La solution proposée par Garland et Heckbert [GH97] consiste tout simplement à ajouter aux sommets des arêtes de contour une quadrique définie par un plan perpendiculaire à l'arête. Afin de mieux différencier les arêtes de contour entre elles, on peut également pondérer la quadrique par la longueur de l'arête. De cette façon, les longues arêtes de contour sont conservées plus longtemps que les courtes. Les figures 4.8(c) à 4.8(e) montrent comment les contraintes réussissent à bien conserver les arêtes de contour.



(a) 200 faces (b) 100 faces

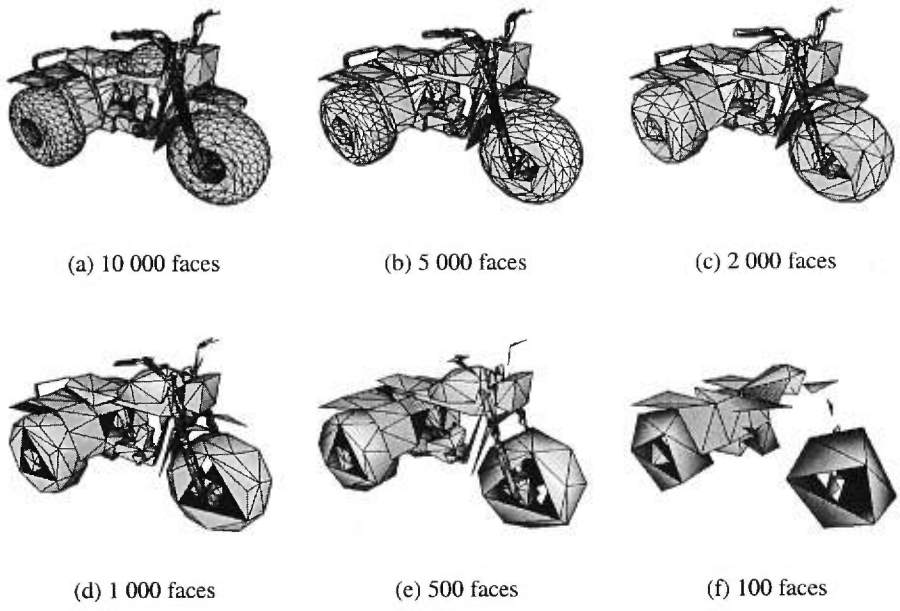
(c) 100 faces (d) 50 faces (e) 12 faces

FIG. 4.8: Problème et solution des arêtes de bordure. La figure (a) constitue le modèle original. La figure (b) illustre le problème lorsque les arêtes de contour ne sont pas traitées. Les figures (c), (d) et (e) montrent comment les quadriques de bordure aident à conserver une simplification de qualité.

Une telle approche nécessite obligatoirement plus de calculs, puisqu'on doit déterminer les arêtes de contour. Mais le calcul de ces arêtes de contour, effectué lors de l'initialisation, n'est habituellement qu'une faible portion du temps de calcul nécessaire à la simplification, et les temps de calcul demeurent donc fort comparables. Il est cependant clair que ce temps dépend considérablement du nombre d'arêtes de contour présentes dans le modèle.

La figure 4.9 (en particulier les figures 4.9(a) à 4.9(c)) permet d'observer l'impact des

quadriques contraintes sur le modèle de référence.



(a) 10 000 faces (b) 5 000 faces (c) 2 000 faces

(d) 1 000 faces (e) 500 faces (f) 100 faces

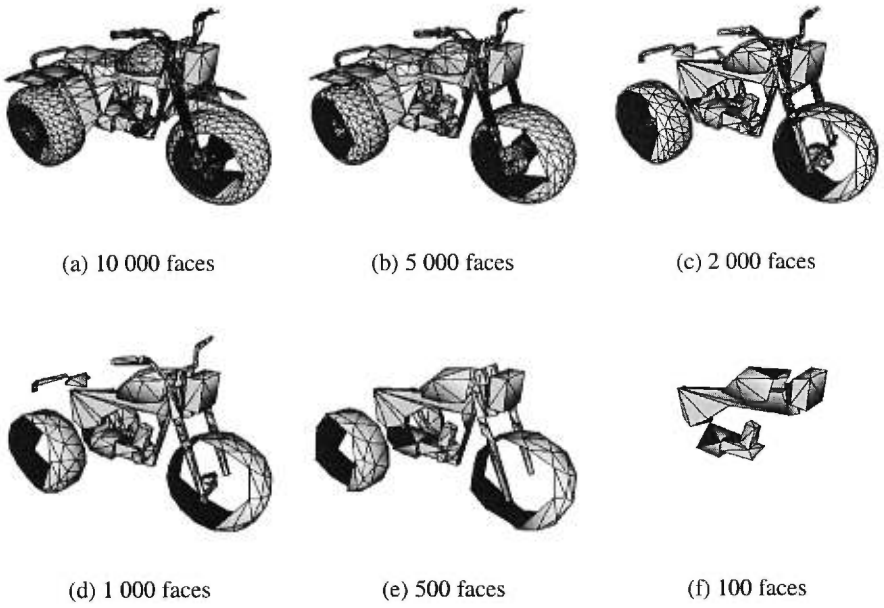
FIG. 4.9: Exemples de simplification par quadriques d'erreur contraintes, c'est-à-dire considérant les arêtes de contour. Le temps moyen de simplification est de 829 ms.

4.3.4 QEM, cadre amnésique

Lindstrom et Turk [LT99] ont été les premiers à proposer qu'un cadre amnésique, c'est-à-dire ne conservant pas les structures de métriques reliées à la simplification, pourrait améliorer les résultats. Un tel cadre demande cependant de recalculer les structures au besoin. Dans une simplification par quadriques d'erreur, ceci implique de reconstruire les quadriques de sommets en se basant non plus sur les faces originales du modèle, mais sur les faces courantes lors de la simplification. Ces nouvelles quadriques ne tiennent évidemment plus compte du modèle original, mais représentent de façon plus appropriée le modèle simplifié courant. Selon Hoppe [Hop99], cette amélioration de la qualité s'explique par le fait que la combinaison de quadriques peut parfois offrir une quadrique résultante correspondant mal au voisinage du sommet auquel elle est associée. En recalculant la quadrique en se basant sur les faces courantes du modèle, on se retrouve à obtenir des quadriques plus justes, améliorant ainsi la simplification.

Un cadre amnésique offre généralement de meilleurs résultats, comme en témoigne la figure 4.10 en comparaison de la figure 4.7. Les temps de calcul sont toutefois considérablement

augmentés. Ils demeurent toutefois encore bien raisonnables, prenant maintenant environ 50% plus de temps.



(a) 10 000 faces (b) 5 000 faces (c) 2 000 faces

(d) 1 000 faces (e) 500 faces (f) 100 faces

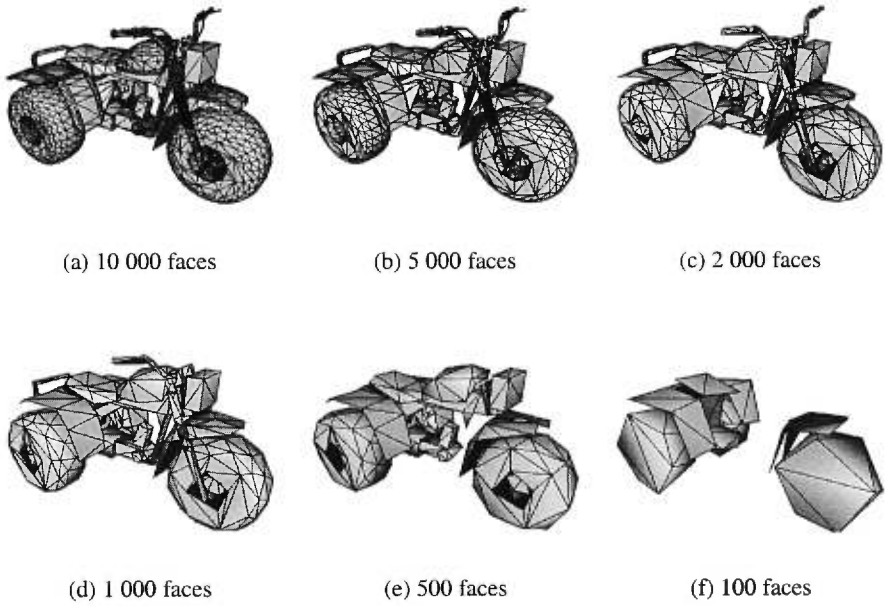
FIG. 4.10: Exemples de simplification par quadriques d'erreur dans un cadre amnésique. Le temps moyen de simplification est de 1 211 ms.

4.3.5 QEM, cadre amnésique contraint

On doit aussi potentiellement tenir compte des arêtes de contour. Ceci implique dorénavant qu'à chaque fois que les quadriques de sommets sont recalculées, on doit également déterminer si le sommet fait partie d'une arête de contour. On doit aussi connaître la face de contour associée afin de pouvoir définir une quadrique perpendiculaire. Pour ce faire, il suffit de parcourir les faces voisines à un sommet et d'ajouter les quadriques de contour au besoin.

Cette métrique est évidemment la plus coûteuse à calculer (presque deux fois plus lente qu'un cadre amnésique simple), mais offre des résultats généralement supérieurs aux métriques précédentes (voir figure 4.11).

La figure 4.12 résume plus visuellement les vitesses relatives des différentes métriques. Les résultats ont été obtenus en effectuant de multiples simplifications sur différents modèles, toujours en utilisant la même machine 600 MHz.



(a) 10 000 faces (b) 5 000 faces (c) 2 000 faces

(d) 1 000 faces (e) 500 faces (f) 100 faces

FIG. 4.11: Exemples de simplification par quadriques d'erreur dans un cadre amnésique en considérant les arêtes de contour. Le temps moyen de simplification est de 2 059 ms.

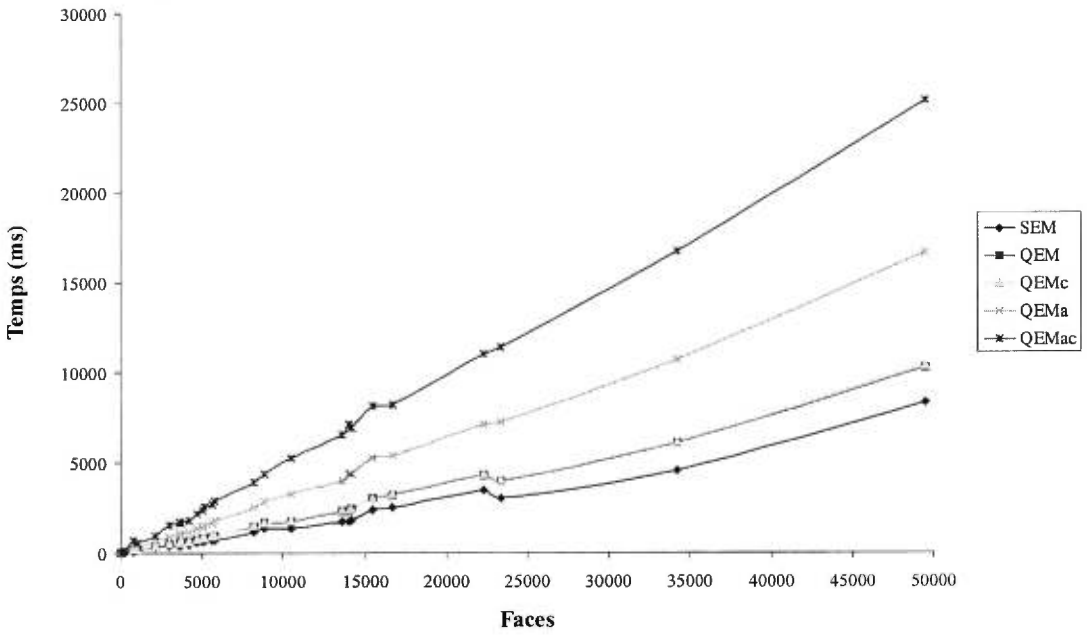


FIG. 4.12: Comparaison du temps de précalcul pour les différentes métriques en fonction du nombre de faces dans le modèle.

Les métriques décrites ici constituent un bien faible échantillonnage de toutes les techniques que nous aurions pu utiliser [Hop96, RR96, CVM+96, CMO97, COM98, CMS98, EM99, LT98, LT99, LT00b]. Nous cherchions toutefois plutôt à intégrer l'animation avec la simplification. Nous avons donc concentré nos efforts pour trouver une solution simple et élégante au problème que pose l'utilisation de modèles squelettiques. Le choix de la métrique demeure libre et constitue selon nous un problème distinct.

4.4 Extension pour un squelette

L'utilisation de l'animation squelettique ajoute un obstacle de taille à la simplification. En fait, jusqu'à présent, la simplification de modèles animés n'a été que marginalement explorée. Les travaux récents de Schmalstieg et Fuhrmann [SF99] constituent, à notre connaissance, la seule technique de simplification pouvant être appliquée sur des modèles animés.

Dans cette technique, les auteurs décomposent le modèle (ou même une scène en entier) en régions disjointes et indépendantes. Ils appliquent alors leur simplification sur chacune des régions, en prenant bien soin de ne jamais fusionner des sommets situés dans des régions différentes. Cette approche empêche la fusion de certaines arêtes (toutes celles reliant deux régions, en fait), mais ceci ne semble pas empêcher de simplifier toute la surface (i.e. dégénérer toutes les faces). Certaines faces doivent cependant disparaître par une fusion de l'arête située complètement d'un côté de la bordure entre deux régions. La figure 4.13 illustre cette situation.

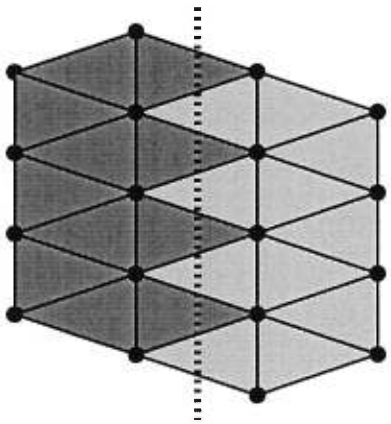


FIG. 4.13: La ligne pointillée traverse les arêtes ne pouvant pas être utilisées lors de fusions. Les faces ainsi traversées doivent obligatoirement dégénérer grâce aux fusions des deux sommets situés de part et d'autre du trait.

Considérons toutes les arêtes d'un modèle comme l'espace de recherche de la simplification. Un cadre non-restreint permet de choisir l'arête optimale (selon une certaine métrique) parmi toutes les arêtes contenues dans cet espace. En empêchant la fusion de certaines arêtes, on réduit ainsi l'espace de recherche, voire même à extraire l'arête optimale des choix potentiels. Il est donc clair que d'empêcher des arêtes de fusionner peut entraîner une réduction de la qualité de la simplification. Mais pour une utilisation où les régions simplifiées contiennent considérablement plus d'arêtes que les bordures en ont, cet impact demeure potentiellement faible. Une telle approche peut donc généralement offrir d'excellents résultats.

Nous avons toutefois préféré prendre une approche différente et n'avoir aucune restriction sur le choix des fusions. Il est donc de notre opinion qu'une simplification, même appliquée à des modèles squelettiques, doit pouvoir théoriquement offrir d'aussi bons résultats qu'une technique identique appliquée plutôt à des modèles statiques. C'est avec cette idée en tête que nous avons étendu notre technique aux modèles squelettiques.

4.4.1 Aperçu

Notons tout d'abord que les métriques doivent être cohérentes entre elles. Pour qu'elles le soient, elles doivent être calculées dans un référentiel commun. Nous devons donc transformer le modèle squelettique en un modèle statique sur lequel on pourra calculer correctement les métriques.

Sachant simplifier un modèle statique, il suffit d'effectuer la simplification normalement sur ce modèle, et d'en répercuter ensuite les modifications sur le modèle squelettique. Ainsi, nous pouvons obtenir un modèle articulé qui, lorsqu'on applique les transformations associées aux articulations, redonnera exactement le modèle statique obtenu lors de la simplification.

Nous avons donc un modèle articulé détaillé $\mathcal{M}_d$ que l'on veut simplifier. Ne pouvant pas le simplifier directement, nous construisons un modèle statique M_d provenant d'une pose fixe du modèle articulé, que l'on simplifiera en M_s. Les structures de simplification sont ensuite interprétées dans le modèle articulé détaillé $\mathcal{M}_d$ de façon à obtenir non plus M_s, mais bel et bien directement $\mathcal{M}_s$. Ce modèle articulé simplifié $\mathcal{M}_s$ peut ainsi être utilisé pour reconstruire exactement le modèle statique simplifié. Comme on le décrit à la section 4.4.2, cette étape ne consiste qu'à interpréter dans le modèle squelettique les structures de fusion calculées originalement pour le modèle statique. La figure 4.14 illustre ce cheminement.

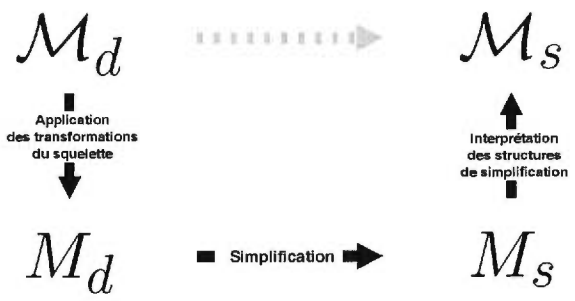


FIG. 4.14: Pour simplifier un modèle squelettique $\mathcal{M}_d$, nous utilisons une pose statique M_d, que nous simplifions en M_s, et dont les structures seront interprétées pour donner le modèle squelettique simplifié $\mathcal{M}_s$.

4.4.2 Répercussions sur le modèle squelettique

Rappelons que chaque fusion (ou fission) est définie par deux indices de sommets, une translation, une série de faces à enlever ou à relier, ainsi qu'une liste de modifications d'attributs. Ces informations, quoique suffisantes pour un modèle statique, sont toutefois un peu faibles dans le cadre d'un modèle squelettique.

Tout d'abord, chaque sommet d'un modèle squelettique peut avoir un nombre arbitraire (mais généralement limité à deux) de représentations pondérées. Bien que les faces soient toujours décrites grâce aux indices des sommets combinés, par opposition aux représentations, la translation s'appliquant à $\mathbf{P}_A$ doit affecter toutes les représentations de A. De la même façon, les attributs pondérés (c'est le cas des normales) doivent également modifier toutes leurs représentations.

Il est à noter que l'on ne pourrait pas tout simplement appliquer la translation sur le modèle statique découlant de l'application d'une pose du modèle squelettique. Ceci vient du fait que la translation est en coordonnées monde, et qu'elle perd son sens lorsque les membres sont animés. Prenons l'exemple d'une fusion dont la translation associée est longitudinale à un membre. La figure 4.15(a) montre le vecteur déplaçant $\mathbf{P}_A$ longitudinalement sur le membre. La figure 4.15(b) affiche maintenant le membre ayant subi une rotation de 90 degrés. Comme on peut le constater, le vecteur $\mathbf{V}_{AA'}$, lorsque appliqué directement sur le modèle statique, ne définit plus un déplacement longitudinal, mais plutôt transversal, et fait ainsi apparaître une pointe fort inappropriée. La façon correcte de procéder consiste à transformer la translation dans l'espace de l'os et de l'appliquer au modèle, tel qu'illustré à la figure 4.15(c). Dans le cas où le sommet est décrit par plusieurs représentations pondérées, chacune d'entre elles devra subir une telle transformation.

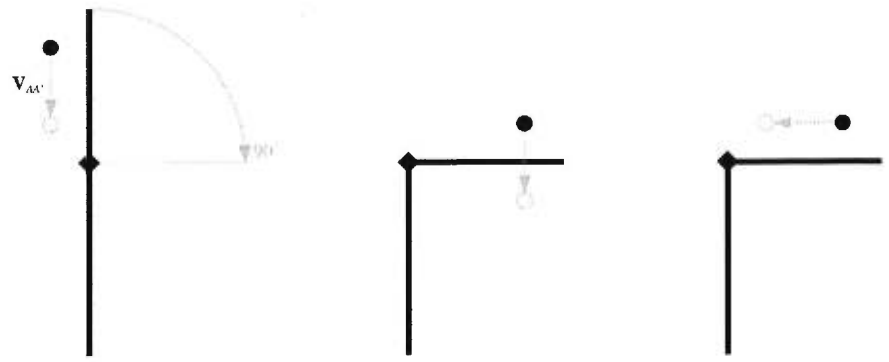


FIG. 4.15: Exemple expliquant pourquoi on doit transformer la translation d'une fusion dans le système de coordonnées des os. La figure (a) indique que le sommet associé à l'os du haut subit une translation $\mathbf{V}_{AA'}$. La figure (b) illustre ce qui arrive lorsqu'on applique cette translation dans une posture différente. La figure (c) illustre la façon correcte de procéder, c'est-à-dire en transformant la translation pour qu'elle soit relative à l'os associé au sommet.

Lorsqu'un sommet du modèle statique est déplacé, on doit traduire ce déplacement dans toutes ses représentations de façon à ce qu'une fois recombinées, elles fournissent la position déplacée. Il existe en théorie un nombre infini de solutions à ce problème. Heureusement, il existe une solution simple et adéquate qui utilise le vecteur de translation de façon invariante dans chacun des référentiels os. Cette solution repose sur le fait que la translation des représentations pondérées ne modifie pas leurs positions relatives.

La technique des sommets pondérés peut être décrite comme suit :

$$\mathbf{P} = \sum_{i=1}^{n} \mathbf{P}_i w_i \qquad \sum_{i=1}^{n} w_i = 1 \tag{4.2}$$

où n correspond au nombre de représentations (généralement deux), chacune décrite grâce à un point $\mathbf{P}_i$ pondéré d'un poids w_i. Considérant ceci, la translation du sommet $\mathbf{P}$ peut maintenant

s'écrire :

$$\begin{aligned}\mathbf{P}' &= \mathbf{P} + \Delta\mathbf{P} \\ &= \sum_{i=1}^{n} \mathbf{P}_i w_i + \Delta\mathbf{P} \sum_{i=1}^{n} w_i \\ &= \sum_{i=1}^{n} (\mathbf{P}_i + \Delta\mathbf{P}) w_i\end{aligned} \tag{4.3}$$

ce qui indique que l'on peut faire subir la translation à chacune des représentations $\mathbf{P}_i$ pour que leur somme pondérée corresponde bel et bien à la position $\mathbf{P}'$. La figure 4.16 montre plus visuellement que d'appliquer la translation sur chacune des représentations d'un sommet pondéré est équivalent à faire subir la translation au sommet résultant.

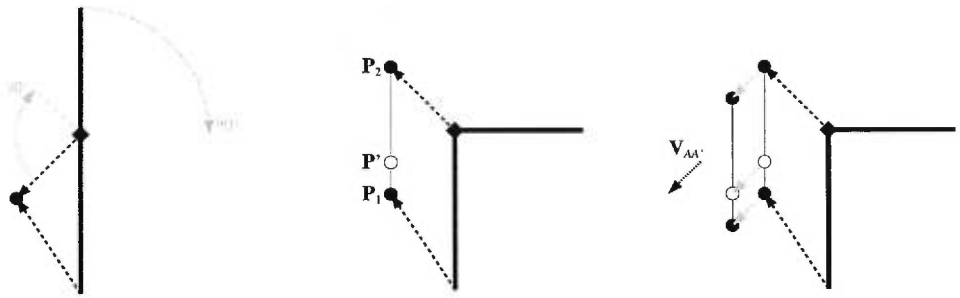


FIG. 4.16: Lorsque l'on fléchit un membre (a), on change la position-monde des représentations pondérées. La figure (b) illustre comment les représentations pondérées $\mathbf{P}_1$ et $\mathbf{P}_2$ s'interpolent en $\mathbf{P}'$. Si $\mathbf{P}'$ subit une translation $\mathbf{V}_{AA'}$ (suite à une fusion), faire subir cette même translation à chacune des représentations entraîne, en (c), que leur combinaison pondérée offre maintenant la nouvelle position de $\mathbf{P}'$.

Comme on peut le remarquer, si l'on fait subir une translation identique à chacune des représentations pondérées d'un sommet, la valeur interpolée du sommet ainsi représenté subira une translation identique. Afin d'arriver à ce résultat, chacune des représentations doit subir une translation dans son propre référentiel. On doit donc transformer le vecteur de translation du référentiel monde à chacun des référentiels os, et l'appliquer à la représentation s'y situant. Pour des questions d'optimisation, nous précalculons et conservons les matrices passant directement du référentiel monde au référentiel de chaque os. Ceci peut être facilement effectué en appliquant l'inverse des transformations de chacun des os (contenues dans la pose de l'ani-

mation). Ainsi, chaque translation-monde d'un sommet entraînera la translation-os de toutes ses représentations. Il en va de même pour les attributs dépendant de la posture, comme les normales.

Cette approche n'est cependant pas exempte de problèmes. Tout d'abord, les sommets sont pondérés, et ceci entraîne un déplacement du sommet résultant selon le mouvement de l'articulation. Ce mouvement n'est cependant pas pris en compte par notre approche, et il se peut qu'une translation aille à l'encontre du mouvement produit par les pondérations. Ceci vient du fait que nous n'utilisons qu'une seule posture, perdant ainsi toutes les informations véhiculées par toutes les autres postures de toutes les autres animations. Cette erreur peut cependant être réduite en partant d'une posture dont les articulations sont au milieu de leurs extrêmes. Ceci tente tout simplement de minimiser l'erreur maximale en la distribuant en deux parties plus ou moins égales. Une approche plus évoluée est discutée au chapitre des extensions (chapitre 6).

Nous obtenons finalement un modèle squelettique multi-résolution, comparable aux maillages progressifs, mais pouvant toujours être animé grâce aux mêmes séquences d'animation squelettique. Ce modèle multi-résolution permet alors de choisir le niveau de simplification désiré pour construire les différents niveaux de détails discrets, ainsi que les structures de morphisme associées (voir chapitre 5).

4.5 Résultats

La figure 4.17 montre les résultats d'une simplification par quadriques d'erreur dans un cadre amnésique contraint. La simplification a pris 623 ms sur un Athlon 600 MHz ayant 256 Mo de mémoire. Comme on peut l'observer, le modèle conserve sa capacité à être animé. La dernière colonne illustre aussi comment une simplification trop forte peut dégénérer des membres entiers (le bras gauche, par exemple). Le modèle de joueur, ainsi que la séquence d'animation sont une courtoisie d'*Electronic Arts Canada.*

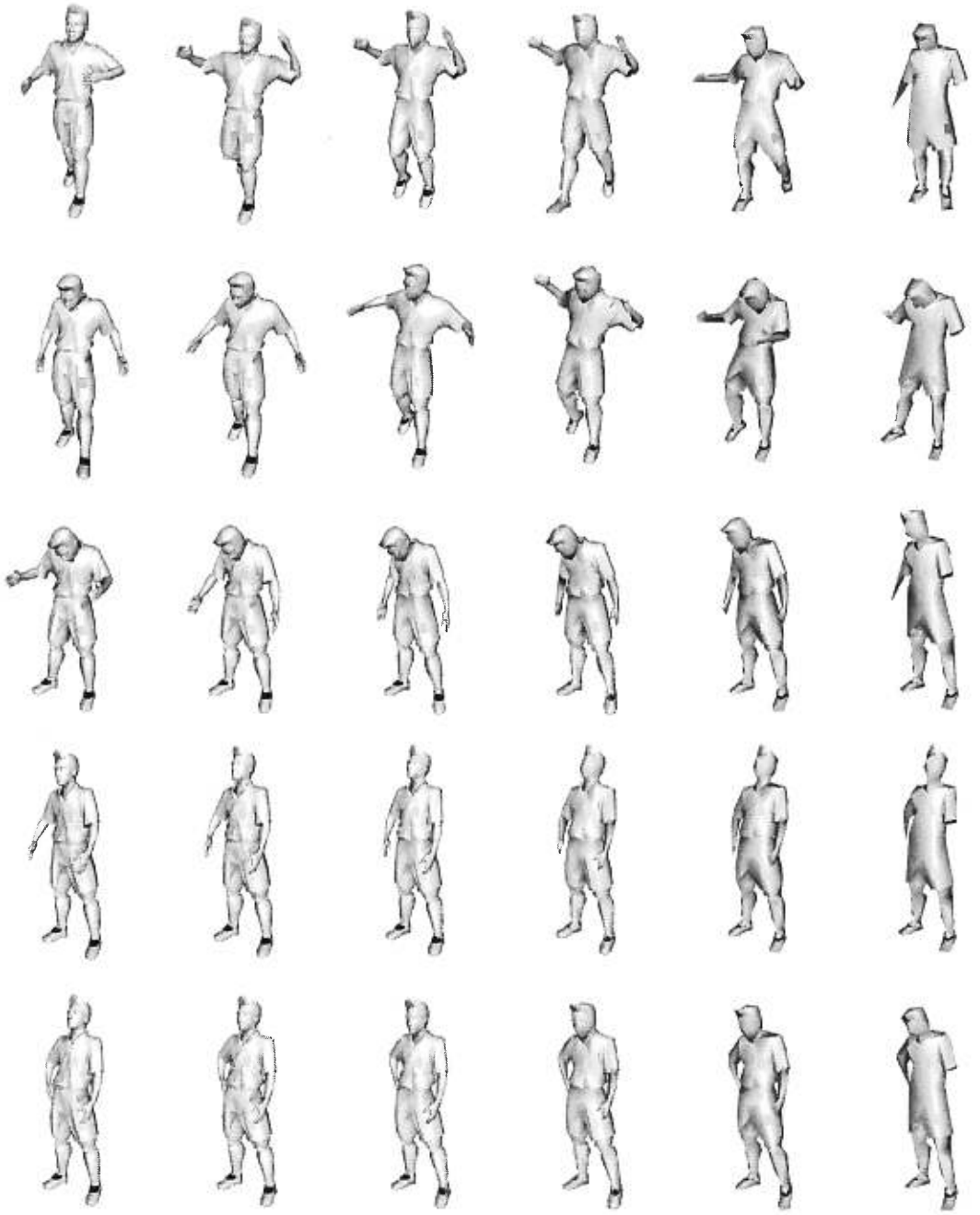


FIG. 4.17: Simplification d'un modèle squelettique. La pose utilisée pour simplifier se situe en haut à gauche. À chaque colonne correspond un niveau de détails. L'animation est présentée de gauche à droite, du haut vers le bas.

Chapitre 5

Morphisme

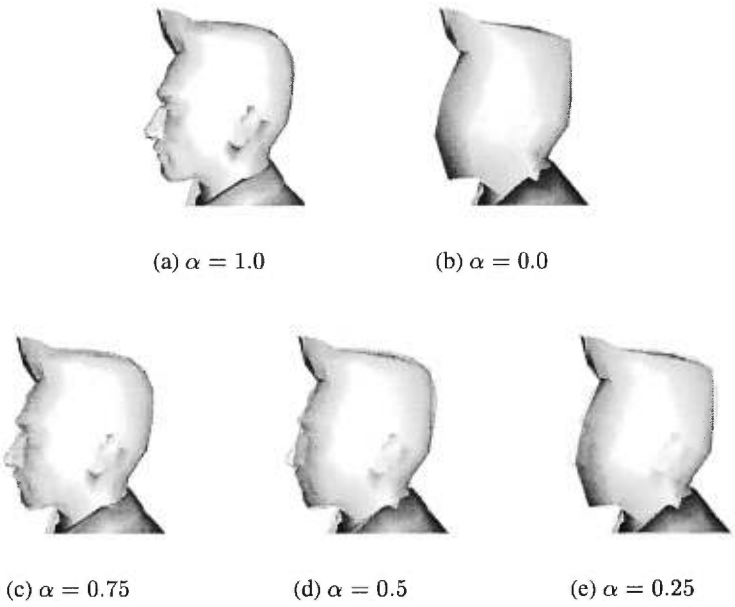
Chi va piano, va sano è lontano

(Qui va lentement, va sûrement et loin)

Proverbe italien

Toute transition subite entre deux images, ou même entre deux modèles géométriques, introduit d'importants artefacts dus au surgissement (*popping*). Une façon de réduire ce problème consiste à composer les éléments (images ou modèles) grâce notamment à la transparence (*alpha blending*). Comme on peut l'observer à la figure 5.1, un élément disparaît graduellement pour laisser sa place au nouvel élément qui apparaît. Lors de la transition, le surgissement est ainsi remplacé par un flou souvent moins perceptible, mais peu de contrôle existe sur la façon dont les éléments apparaissent. Une approche sémantiquement plus correcte consiste plutôt à faire croître graduellement les composantes sujettes au surgissement. Il s'agit alors de modifier l'élément source pour qu'il se transforme graduellement et structurellement en l'élément destination. De là est né le morphisme.

Le morphisme a pris son essor vers 1992 grâce aux travaux de divers chercheurs et a continué de se développer dans les années subséquentes. Qu'il s'agisse de techniques s'appliquant à des images [BN92, LWCS96, LCHS96, SD96, Wol98] ou des techniques s'appliquant à des modèles 2D [SG92, SGWM93, GH94, GG95] ou même 3D [Hug92, KCP92, Hop96, MJT96, CC98, COLS98, KSK98, LV98, LSS+98, GSL+99, LDSS99, TO99, ACOL00, KSK00, ZSH00], les références sur le morphisme abondent. L'idée de base du morphisme consiste tout simplement à faire croître ou à déformer graduellement une composante qui ne ferait autrement qu'apparaître.



(a) $\alpha = 1.0$ (b) $\alpha = 0.0$

(c) $\alpha = 0.75$ (d) $\alpha = 0.5$ (e) $\alpha = 0.25$

FIG. 5.1: Exemple de transition par transparence. Les figures (a) et (b) constituent les deux niveaux de détails entre lesquels on effectue la transition. Les figures (c) à (e) montrent la transition par transparence.

La simplification par fusion permet cependant d'effectuer un morphisme bien particulier. Le géomorphisme décrit par Hoppe [Hop96] est l'opération par laquelle on transforme un modèle géométrique en un autre en confondant leurs sommets et leurs faces. Cependant, la forme géométrique ne détermine pas entièrement l'apparence d'un modèle. Ainsi, le géomorphisme devient plus généralement morphisme lorsqu'il considère également les attributs des faces, tels les couleurs, les normales, les textures, etc. C'est pour ce type de morphisme que nous avons opté. En comparaison d'un cadre général, le morphisme que nous effectuons est relativement simple. La forme même de notre technique de simplification, ou plus particulièrement le format des maillages progressifs, offre toute l'information nécessaire pour effectuer un morphisme simple et efficace. La parenté entre les différents niveaux de détails créés nous offre un lien privilégié que nous avons avantage à utiliser.

5.1 Aperçu

Le morphisme doit tout d'abord faire correspondre les sommets d'un modèle de départ M_d aux sommets d'un modèle simplifié M_s. En ayant ces correspondances, on peut ensuite

déplacer chacun des sommets de M_d vers un sommet de M_s. En déplaçant ces points, on se retrouve à faire correspondre les faces de M_d avec celles de M_s, à l'exception des faces supplémentaires de M_d qui finiront plutôt dégénérées (i.e. ne seront plus visibles). C'est en déplaçant graduellement ces points qu'on arrive à effectuer le géomorphisme. Si le modèle pouvait contenir des polygones autres que des triangles, il est clair que le déplacement arbitraire d'un des sommets pourrait détruire la planarité de la face. C'est pour cette raison, ainsi que par souci de simplicité, que notre technique ne s'applique qu'à des modèles triangulaires.

Mais faire correspondre uniquement les sommets, et incidemment les faces, n'est pas suffisant : il faut aussi que les attributs (normales, couleurs, coordonnées de textures, etc.) de ces faces soient les mêmes pour que les modèles soient identiques. En effet, on ne peut pas simplement conserver intacts les attributs associés aux faces puisqu'ils sont modifiés lors de la fusion de sommets.

On doit donc faire correspondre les sommets, pour que les faces de M_d deviennent identiques à celles de M_s (ou dégénèrent). Mais on doit aussi faire correspondre les attributs de ces faces, pour que leur apparence finale soit identique. Voyons maintenant comment nous construisons et utilisons les structures nécessaires pour effectuer notre morphisme entre deux niveaux de détails donnés par un maillage progressif.

5.2 Correspondances

Notons tout d'abord que l'on peut aisément construire la correspondance entre deux simplifications distinctes venant d'un même maillage progressif. Il est possible d'établir les correspondances de sommets en ne traitant que les structures de fusion, puisque les sommets fusionnent ensemble. Mais puisque les coins sont mis à jour à chaque fusion, nous avons jugé préférable de traiter plus uniformément la correspondance de sommets ainsi que des attributs.

Nous commençons donc par raffiner le modèle jusqu'à la configuration la plus détaillée M_d. Ensuite, nous appliquons itérativement chacune des fusions de sommets en analysant également la structure de fusion utilisée. Ceci nous permet de construire rapidement toutes les correspondances (sommets ou attributs) nécessaires pour effectuer le morphisme entre les deux niveaux de détails spécifiés.

5.2.1 Correspondance des sommets

La correspondance des sommets se fait de façon très simple. À chaque fusion, nous savons que le sommet $\mathbf{P}_B$ fusionne vers le sommet $\mathbf{P}_A$. Il suffit tout simplement de noter que $\mathbf{P}_B$, dans le modèle détaillé M_d, doit maintenant se situer à la position du sommet $\mathbf{P}_A$ dans le modèle simplifié M_s. Puisque les sommets sont contigus en mémoire, nous n'avons qu'à conserver deux entiers : l'indice B de $\mathbf{P}_B$ et l'indice A de $\mathbf{P}_A$. Le vecteur reliant ces sommets peut être recalculé en temps réel lors du morphisme.

Ces correspondances permettent d'effectuer le morphisme entre les sommets qui fusionnent en un autre, mais elles ne contiennent pas tous les sommets qui se déplacent lors de la simplification. En effet, il est possible qu'une suite de fusions s'opère toujours vers un seul et même $\mathbf{P}_A$. Puisqu'à chaque fusion, le sommet résultant peut être déplacé, il se peut que $\mathbf{P}_A$ dans le modèle détaillé M_d n'occupe plus la même position que $\mathbf{P}_A$ du modèle simplifié M_s. Dans de tels cas, nous devons également faire correspondre $\mathbf{P}_A$ vers lui-même, de façon à pouvoir effectuer un morphisme entre les positions distinctes dans chacun des deux modèles (M_d et M_s). Une façon simple de procéder consiste à faire pointer l'indice de $\mathbf{P}_A$ vers lui-même. Ceci indique effectivement que $\mathbf{P}_A$ du modèle détaillé M_d doit être déplacé vers $\mathbf{P}_A$, mais du modèle simplifié M_s cette fois.

5.2.2 Correspondance des attributs

La correspondance des attributs demande à peine plus de travail. Plutôt que d'être effectuée sur les indices de sommets, cette correspondance utilise les indices de coins. Le calcul supplémentaire vient de la mise à jour des coins des faces (voir section 4.1.1) ; la correspondance elle-même est identique à celle construite pour les sommets.

Lorsqu'une face est connectée à un nouveau sommet (lorsque $\mathbf{P}_B$ fusionne en $\mathbf{P}_A$), le coin qui change de sommet doit être noté afin de conserver l'information de correspondance. De la même façon, les coins des faces qui ne sont que déformées lors d'une fusion (parce que $\mathbf{P}_A$ se déplace) doivent également être notés.

Puisque la mise à jour des coins est relativement complexe, nous avons évité d'analyser la structure de fusion et d'effectuer la simplification du modèle tout au long du calcul des correspondances. Ainsi, les correspondances nous sont données par les valeurs avant et après chacune des fusions, et c'est le code de simplification qui s'occupe de mettre à jour les coins.

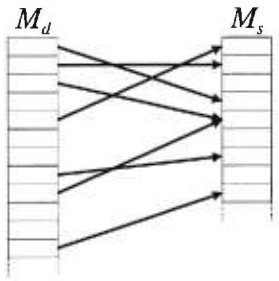


FIG. 5.2: Lorsque les correspondances sont construites et compactées, nous obtenons une table de correspondances indiquant, pour chacun des sommets de M_d ayant changé, vers quel sommet de M_s il doit se diriger.

5.2.3 Structures obtenues

Une fois les correspondances construites, il suffit ensuite de les compacter. Ainsi, si un sommet $\mathbf{P}_A$ correspond à $\mathbf{P}_B$ qui correspond à $\mathbf{P}_C$, on conservera finalement plutôt que $\mathbf{P}_A$ correspond directement à $\mathbf{P}_C$. Cette approche est valide puisque si $\mathbf{P}_B$ correspond à $\mathbf{P}_C$, c'est que $\mathbf{P}_B$ n'existe plus dans le modèle, puisqu'il a fusionné vers $\mathbf{P}_C$. Ceci permet tout simplement des liens plus directs lors du morphisme, et réduit ainsi les temps de calcul.

Dans cette même optique, nous ne conservons pas des correspondances de coins, mais plutôt des correspondances d'attributs tels la normale et la couleur. Ainsi, lors du compactage des correspondances de coins, on aboutira plutôt à une correspondance d'indices de normales et d'indices de couleurs. Cette transformation est très facile à faire : il s'agit tout simplement de faire correspondre l'indice de la normale du coin C_A avec l'indice de normale du coin C_B. L'avantage principal réside dans le fait que lors du morphisme, on n'aura pas à lire le coin afin de trouver les attributs à modifier puisqu'on les a directement. Ceci consomme cependant plus de mémoire, mais cette hausse est fort raisonnable en comparaison de la taille du modèle.

On obtient finalement une liste de correspondances pour les sommets (comme illustré à la figure 5.2), ainsi qu'une liste de correspondances pour chacun des attributs. Pour des raisons de temps, nous n'avons implanté que les normales, mais le cadre s'étend facilement à autant d'attributs que désiré. Chaque correspondance est composée de deux entiers : les indices source et destination. Cette structure est très compacte en comparaison de la taille des modèles, et contient toute l'information nécessaire à un morphisme rapide et de qualité. La figure 5.3 montre que l'on peut également construire une chaîne de modèles simplifiés avec leurs correspondances respectives.

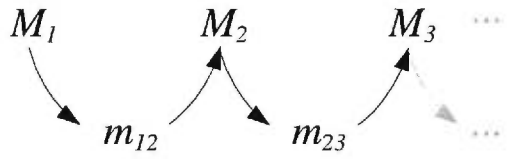


FIG. 5.3: Exemple d'enchaînement des morphismes entre des niveaux de détails. Chaque m_{ij} correspond aux structures de morphisme pour effectuer la transition entre les modèles M_i et M_j.

5.3 Gestion des structures conservées

Une fois les correspondances construites, on peut commencer à effectuer le morphisme. Nous calculons le morphisme grâce à un paramètre de transition variant entre 0 et 1 (0 donnant exactement le modèle détaillé M_d et 1 donnant exactement le modèle simplifié M_s). Voyons comment ce paramètre s'applique sur les différentes composantes du modèle.

5.3.1 Morphisme des positions géométriques

On sait dorénavant vers quel sommet du modèle simplifié pointe chacun des sommets potentiellement modifiés dans le modèle détaillé. La position d'un sommet lors du morphisme peut être définie comme suit :

$$\mathbf{P}'_i = \mathbf{P}_i + (\mathbf{P}_{f(i)} - \mathbf{P}_i)t \qquad 0 < t < 1 \tag{5.1}$$

où $\mathbf{P}_i$ est la position du sommet dans M_d, $\mathbf{P}_{f(i)}$ est la position correspondante dans M_s (f étant la fonction de correspondance du morphisme) et t est le paramètre de transition. Il est possible de conserver le vecteur $(\mathbf{P}_{f(i)} - \mathbf{P}_i)$ afin d'éviter de le recalculer à chaque fois lors du morphisme entre deux modèles statiques.

Cette équation peut toutefois être réécrite ainsi :

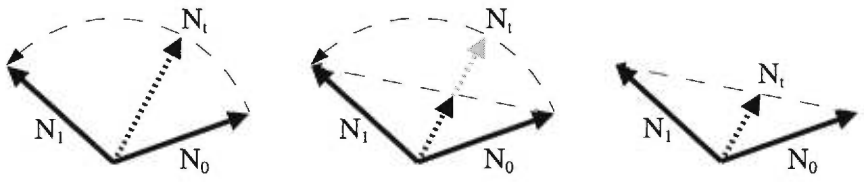
$$\mathbf{P}'_i = \mathbf{P}_i(1 - t) + \mathbf{P}_{f(i)}t \qquad 0 < t < 1 \tag{5.2}$$

ce qui offre un gain considérable dans notre implantation. Ceci vient du fait que le morphisme s'effectue sur le modèle M_d. On doit copier complètement ce modèle, et manipuler ses sommets et attributs de façon à les faire correspondre, à $t = 0$, aux sommets et attributs correspondants dans M_s. Puisque l'on part d'une copie de M_d, on a déjà $\mathbf{P}_i$, qu'il nous suffit de

multiplier par $(1 - t)$ que l'on calcule une seule fois pour tout le morphisme. Le calcul de la composante $\mathbf{P}_{f(i)}t$ est inévitable. Notre implantation finale avec cette approche est équivalente, en termes de coûts de calcul, à une approche où l'on conserverait les vecteurs de morphisme. Autrement dit, si on précalcule et conserve les vecteurs $(\mathbf{P}_{f(i)} - \mathbf{P}_i)$ et que l'on ne fait que les appliquer lors du morphisme, on n'est pas plus rapide que de recalculer la position grâce à l'équation 5.2. C'est pour cette raison que nous n'avons pas besoin de plus d'information que les correspondances pour faire un morphisme simplement et rapidement. La simplicité des calculs nécessaires fait des équations 5.1 et 5.2 des candidats intéressants pour un support matériel accéléré.

5.3.2 Morphisme des attributs

L'interpolation entre deux normales doit faire varier la normale sur le grand arc reliant la pointe des deux normales sur la sphère unitaire. Sachant que l'axe de rotation amenant $\mathbf{N}_0$ sur $\mathbf{N}_1$ est donné par le produit vectoriel $\mathbf{N}_0 \times \mathbf{N}_1$, et que le cosinus de l'angle autour de cet axe est donné par le produit scalaire $\mathbf{N}_0 \cdot \mathbf{N}_1$, il suffit de faire subir à $\mathbf{N}_0$ une rotation de $(\arccos(\mathbf{N}_0 \cdot \mathbf{N}_1))t$ autour de l'axe de rotation pour obtenir la normale interpolée $\mathbf{N}_t$. L'utilisation de quaternions [Sho85, Sho87] permet d'alléger quelque peu ces calculs.



(a) Grand arc (b) Linéaire renormalisé (c) Linéaire simple

FIG. 5.4: Différentes façons d'interpoler deux normales lors du morphisme. L'utilisation de quaternions (a) est cinq fois plus lente que l'interpolation linéaire renormalisée, et dix fois plus lente si l'on néglige la normalisation.

En pratique, toutefois, nous avons observé que les normales à interpoler restaient généralement très proches l'une de l'autre. Une approche fausse, mais plus efficace, consiste à interpoler linéairement les pointes des deux normales. Ceci a pour effet de couper court le grand arc, et offre une interpolation dont la vitesse angulaire n'est pas constante (voir figure

5.4). Le vecteur résultant n'étant plus de longueur unitaire, on doit le renormaliser si l'on veut conserver corrects les calculs d'illumination qui en dépendent. Il est toutefois possible de ne pas conserver normalisé le vecteur interpolé. Ceci entraîne malheureusement des erreurs au niveau des calculs d'illumination.

Notre expérience semble toutefois indiquer que de telles erreurs sont souvent peu perceptibles. Ceci s'explique de deux façons. Tout d'abord, le morphisme s'effectue en peu de temps (environ une seconde), et puisque les normales demeurent généralement très près de leurs valeurs initiales, l'erreur ainsi véhiculée s'en retrouve en quelque sorte noyée dans la variation globale du modèle. Également, rappelons-nous que le but du morphisme est d'enrayer le surgissement. Ainsi, même si cette interpolation des normales est inexacte, elle est tout de même régulière, ce qui est suffisant dans la plupart des cas pour offrir une transition harmonieuse.

Les autres attributs (couleurs, coordonnées de texture, etc.) peuvent être modifiés grâce à une interpolation linéaire simple (comme pour les sommets) ou un peu plus sophistiquée (comme pour les normales). Les couleurs, par exemple, peuvent être interpolées dans l'espace RGB habituel, ou être transformées dans l'espace $L^*u^*v^*$ [FDFH90] pour une interpolation perceptuellement plus linéaire. Les coordonnées de textures 2D peuvent être interpolées linéairement dans leur espace UV à condition de ne pas dévoiler ainsi des zones initialement inutilisées dans la texture, et à condition de faire attention au problème de *wrap-around*. Les travaux de Cohen *et al.* [CMO97, COM98] expliquent plus en détail quelques-unes des conditions à respecter lors du calcul de nouvelles coordonnées de texture.

En résumé, le morphisme part d'une copie du modèle détaillé M_d. En utilisant un paramètre t, $0 < t < 1$, lequel peut être arbitrairement fixé, on applique toutes les correspondances. On déplace ainsi chaque sommet d'une portion t du trajet l'amenant à sa position finale dans le modèle simplifié M_s. On transforme également chacun des attributs grâce à ce même paramètre t. On peut donc partir de M_d (lorsque $t = 0$), le modifier graduellement pour qu'il corresponde finalement au modèle M_s (lorsque $t = 1$). À ce moment, le morphisme se termine et on peut utiliser le modèle M_s, plus petit et donc plus rapide à afficher. La progression du morphisme est arbitraire et il en revient à l'usager de trouver une valeur t appropriée pour qu'il s'effectue dans le sens et selon la durée désirés. Le morphisme n'est d'ailleurs pas tenu d'être linéaire entre 0 et 1.

5.4 Résultats

La figure 5.5 illustre les résultats du morphisme appliqué sur un modèle statique. Les structures de morphisme ont été obtenues en 330 ms (Athlon 600 MHz, 256 Mo de mémoire). Le morphisme est effectué sur la quasi-totalité du modèle (environ 7 000 sommets et 10 000 normales). Malgré ce fait, le temps de calcul est inférieur au tiers du temps nécessaire pour le rendu (9.8 ms de morphisme contre 35 ms pour le rendu dans un code non optimisé pour le rendu en temps réel). En effectuant une série de morphismes répétés, nous avons déterminé que nous pouvions effectuer plus de 1.5 million d'interpolations (sommets et normales) par seconde. Pour obtenir des résultats en temps réel (60 Hz) sur une scène comportant plusieurs modèles, il faudrait améliorer encore les résultats. Une implantation plus optimisée combinée aux performances accrues du matériel permettrait probablement d'utiliser notre technique en temps réel dans les années à venir.

Nous avons également quelques animations montrant plus clairement l'impact du morphisme sur la qualité de la transition. Ces animations sont disponibles à l'adresse suivante : http ://www.iro.umontreal.ca/labs/infographie/theses/houlejo/.

(a) $t = 0.0$ (b) $t = 0.2$ (c) $t = 0.4$

(d) $t = 0.6$ (e) $t = 0.8$ (f) $t = 1.0$

FIG. 5.5: Morphisme effectué entre deux niveaux de détails statiques. Le modèle le plus détaillé comporte 13 584 faces, alors que le modèle le plus simplifié en comporte 2 000.

5.4.1 Morphisme total

Lorsqu'on effectue un morphisme vers un modèle totalement simplifié (dont il ne reste aucune face), le morphisme devient curieusement un changement d'échelle. Ceci s'explique par le fait que chaque région du modèle dégénère en un seul et même point. En considérant chacun de ces points comme étant l'origine d'un référentiel, chaque vecteur d'interpolation de sommets revient simplement à un changement d'échelle proportionnel au paramètre t. La figure 5.6 montre jusqu'à quel point cette particularité permet de bien distinguer les parties disjointes d'un modèle. Bien que cette caractéristique n'ait pas vraiment d'utilité dans notre recherche, nous avons jugé bon de la mentionner par souci de complétude.

(a) $t = 0.25$ (b) $t = 0.5$ (c) $t = 0.75$

FIG. 5.6: Exemple de morphisme où chacune des régions du modèle dégénèrent en un seul point. Ce cas particulier de morphisme correspond à un changement d'échelle effectué sur chacune des régions disjointes du modèle.

5.5 Extension pour un modèle squelettique

Nous avons opté pour une approche simple mais efficace d'effectuer le morphisme sur un modèle squelettique. Elle consiste à transformer d'abord le modèle squelettique en modèle statique, et ensuite d'appliquer les structures de morphisme normalement. Pour ce faire, nous n'avons qu'à utiliser un maillage progressif statique et d'appliquer la démarche expliquée à la section 5.2. Procéder de cette façon est tout à fait valide puisque la connexité d'un modèle squelettique $\mathcal{M}_i$ est la même que pour son modèle statique M_i. Puisque la simplification s'opère à partir de cette connexité, les dépendances en découlant restent elles aussi valides. On demeure également cohérent avec le fait que les structures de fusion sont remplies grâce au modèle statique, et non squelettique, et qu'elles sont ensuite interprétées dans le modèle squelettique (voir figure 4.14). Cette approche a le mérite de conserver la simplicité du morphisme et de son

implantation. Elle arrive ainsi au résultat escompté et cela, avec relativement peu de calculs.

On doit donc transformer le modèle articulé détaillé $\mathcal{M}_d$ en modèle statique M_d, et faire de même pour le modèle simplifié $\mathcal{M}_s$ pour obtenir M_s. Ensuite, il s'agit tout simplement d'effectuer le morphisme entre M_d et M_s. Cette approche nécessite évidemment de transformer deux modèles squelettiques en deux modèles statiques. Cependant, en comparaison de la technique de transparence (*alpha-blending*) décrite à la section 3.3, il s'agit d'un coût tout à fait acceptable puisqu'on se retrouve à faire le même calcul sous-jacent (transformer deux modèles squelettiques), et on remplace le rendu du modèle simplifié par le morphisme. Les coûts relatifs sont très comparables, et notre technique offre des transitions beaucoup plus agréables que celles offertes par la technique utilisant la transparence.

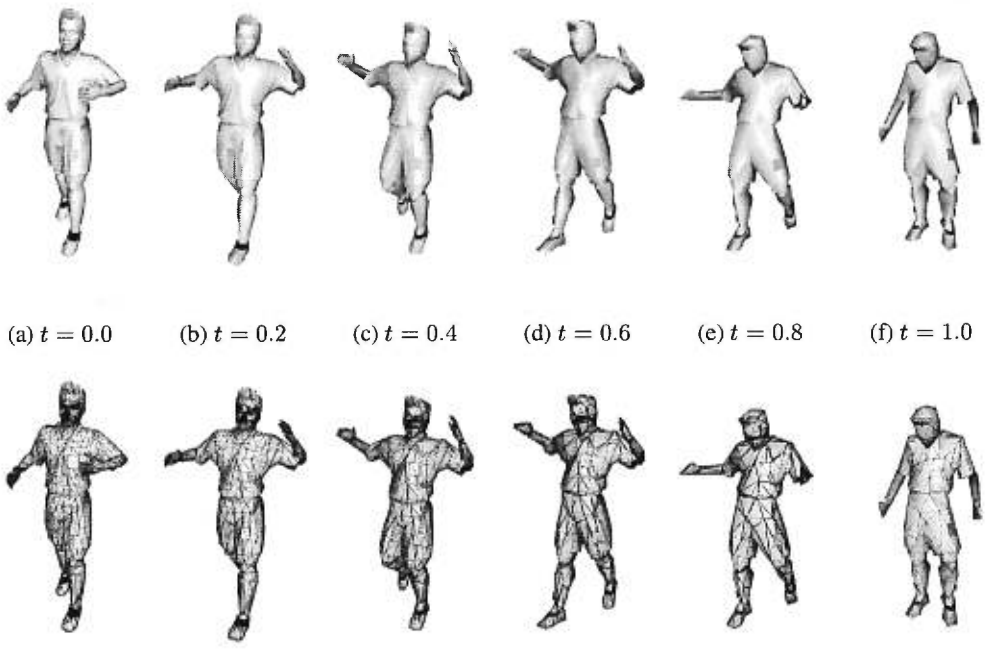
On aurait aussi pu tenter d'effectuer le morphisme directement entre les modèles squelettiques. On aurait donc dû construire un modèle articulé morphé, et le transformer ensuite pour l'affichage en un modèle statique. Cette approche comporte toutefois quelques désavantages. Plutôt que d'effectuer l'interpolation du morphisme dans l'espace du modèle, on doit dorénavant l'effectuer dans l'espace de chaque os. Or s'assurer que ces interpolations demeurent identiques dans chaque espace ne nous semble pas évident. Une telle approche a aussi le désavantage de devoir interpoler un plus grand nombre de valeurs (sommets, normales, etc.) à cause de la présence des représentations pondérées[1]. Ceci accroît de façon inutile les temps de calcul nécessaires pour obtenir un morphisme comparable, voire même identique.

Notre approche a donc l'avantage d'interpoler un nombre réduit de sommets. Bien qu'il soit vrai que l'on doive transformer le modèle simplifié squelettique $\mathcal{M}_s$ en modèle statique M_s, nous considérons qu'il s'agit là d'un coût relativement faible à payer puisque le modèle simplifié est une fraction plutôt faible du modèle détaillé (typiquement le cinquième de la taille). Une approche encore plus performante prendrait soin de ne transformer que les points qui subissent le morphisme. Par souci de simplicité, notre implantation utilise une approche beaucoup plus naïve, mais suffisamment efficace pour être appliquée en temps réel.

[1] Il ne s'agit pas là d'un nombre négligeable puisque les modèles squelettiques utilisés étaient composés environ à moitié de sommets pondérés, et que chaque sommet pondéré utilise habituellement une normale similairement pondérée.

5.6 Résultats

Un exemple de morphisme effectué sur un modèle squelettique animé est illustré à la figure 5.7. Le morphisme s'effectue sur la quasi-totalité du modèle (moins de 5% des sommets et des normales ne subissent aucun changement). Pour le modèle illustré, le nombre maximum d'interpolations par seconde se situe aux alentours de 600 000, ce qui est moins de la moitié des résultats pour les modèles statiques. Dans notre implantation, le morphisme entre les deux modèles squelettiques prend un temps moyen à peine inférieur au temps nécessaire pour le rendu du modèle statique résultant. Ceci implique donc que l'utilisation de notre technique réduit la performance de moitié, mais ce, uniquement lors des transitions. Nos résultats indiquent qu'il est quand même possible d'appliquer notre technique en temps réel.



(a) $t = 0.0$ (b) $t = 0.2$ (c) $t = 0.4$ (d) $t = 0.6$ (e) $t = 0.8$ (f) $t = 1.0$

FIG. 5.7: Exemple de morphisme appliqué à un modèle squelettique. Le modèle détaillé comporte 4 308 faces alors que le modèle simplifié en comporte 500.

Nous avons décrit comment effectuer le morphisme sur des modèles statiques, et comment utiliser ces structures pour des modèles squelettiques. Notre approche demeure simple et efficace, et permet d'effectuer des morphismes dans des temps que nous considérons négligeables

en comparaison de l'amélioration de la transition entre deux niveaux de détails. Elle permet ainsi d'obtenir des niveaux de détails articulés dont les transitions sont efficaces en temps, concises en mémoire et lisses dans leur apparence.

Chapitre 6

Extensions

There is a difference between knowing the path,
and walking the path...

Morpheus, *The Matrix*

Nous avons décrit comment, à partir d'un modèle squelettique quelconque, on peut construire une série de modèles squelettiques simplifiés pouvant être animés grâce aux mêmes séquences d'animation que le modèle original. Nous avons également décrit comment réduire le problème de surgissement lors des transitions entre les niveaux de détails en utilisant du morphisme.

Notre technique n'est cependant pas exempte d'améliorations potentielles. Voici quelques-unes des améliorations auxquelles nous avons pensé. Nous avons trouvé logique de les classer en respectant la présentation du présent mémoire. Nous discuterons donc des extensions ayant trait à la simplification, puis de celles concernant le morphisme.

6.1 Simplification

6.1.1 Métriques

Nous avons utilisé quelques métriques, mais il est clair qu'il en existe beaucoup d'autres. Les travaux conjoints de Bredy [Bre00] ont permis d'ouvrir de nouvelles avenues en comparant diverses métriques de simplification. Les critères de simplification étudiés sont aussi variés que l'aire des faces affectées, la conservation de volume ainsi que le point de vue. Nous sommes conscients qu'il existe encore beaucoup de métriques à évaluer et à développer, et les travaux de Bredy ont ouvert des portes qui seraient restées autrement closes.

Il serait peut-être possible d'analyser le modèle pour en extraire différentes caractéristiques telles ses propriétés de réflexion, sa distribution d'illumination de même que sa transparence. En ayant une idée de l'importance relative de ces différents facteurs à travers le modèle, on pourrait effectuer une simplification intégrant davantage notre perception du modèle et de ses diverses caractéristiques. Cette approche se distingue de la simplification par images (section 2.5.3) dans la mesure où chaque composante peut être séparée des autres, alors qu'une image noie toutes les informations dans une seule et même valeur.

Nous sommes également conscients qu'il n'existe aucune métrique parfaite, et que le choix de la métrique repose essentiellement sur les besoins qui caractérisent l'usage des modèles simplifiés. Toutefois, les performances actuelles de notre technique indiquent que nous disposons de beaucoup de latitude dans nos temps de calcul. On pourrait donc imaginer une métrique beaucoup plus coûteuse dans laquelle une multitude de sous-métriques seraient utilisées et comparées afin d'offrir un modèle simplifié d'une plus grande qualité. On pourrait ainsi faire tourner l'algorithme pendant des heures, voire même des jours, et obtenir potentiellement un modèle simplifié d'une qualité considérablement améliorée. Une telle approche permet tout simplement de prendre avantage de l'efficacité des techniques actuelles pour en améliorer la qualité.

6.1.2 Attributs

Il est clair que notre implantation est quelque peu déficiente en ce qui a trait au nombre d'attributs traités. La raison principale, outre le temps que nous disposions, est que nous cherchions plutôt à effectuer une simplification sur un modèle articulé, ce qui n'avait pas encore été effectué auparavant. Il est cependant clair qu'une technique plus complète devrait considérer les attributs dans ses métriques, et devrait également les modifier lors des fusions. Notre implantation ne considère que les sommets, et ne fait que gérer l'impact des fusions sur les normales. Une simplification aurait fort avantage à considérer bien plus que les normales.

Les couleurs et les textures constituent les attributs typiques que les techniques de simplification considèrent. Ainsi, les travaux de Hoppe [Hop96, Hop99] ainsi que ceux de Garland et Heckbert [GH98] constituent une base solide pour l'intégration de ces attributs dans le processus de simplification. Les travaux de Cohen *et al.* [CMO97, COM98] décrivent une technique plus appropriée pour la conservation des coordonnées de texture, puisqu'elle tient compte de la distorsion. Finalement, les travaux récents de Lindstrom et Turk [LT00b, LT00a] utilisent

le résultat du rendu comme critère de simplification. Une telle métrique constitue, selon nous, une approche nouvelle et prometteuse pour une intégration des différents attributs en une seule métrique.

L'utilisation de plus en plus courante des textures[1] semble entraîner une propension à leur intégration. Une approche simpliste, mais pouvant s'avérer convenable, consisterait à effectuer une simplification géométrique comme nous le faisons actuellement, mais en modifiant tout simplement les coordonnées de texture, sans les intégrer dans la métrique. Bien que cette approche pourrait introduire quelques erreurs, nous croyons qu'une reparamétrisation automatique des coordonnées de texture pourrait, à défaut d'être parfaite, offrir une solution grossière à un artiste qui pourrait corriger *a posteriori* le modèle.

6.1.3 Intervention d'un artiste

Aucune métrique n'est parfaite. Qu'il s'agisse des sommets ou des attributs, toute méthode automatique, de par son cadre strict, peut donner des résultats extrêmes indésirables.

C'est pour cette raison que nous avons pensé développer notre technique de simplification comme un outil interactif. Puisqu'il est déjà fastidieux pour un artiste de créer un modèle, l'effort s'en retrouve multiplié lorsqu'il doit en créer plusieurs à différentes résolutions. Nous croyons que la simplification doit faciliter la vie de l'artiste, sans pour autant l'éloigner complètement du processus de création des modèles, y compris les différents niveaux de détails.

Nous croyons donc que l'artiste peut utiliser notre technique pour générer des modèles, mais qu'il doit quand même conserver un droit de veto, question de s'assurer que les modèles générés respectent bien son intention originale. Ainsi, si un modèle simplifié ne lui plaît pas, il doit pouvoir le modifier à sa guise. Si un sommet se déplace trop loin, il doit pouvoir le modifier à la main. Si les couleurs ou les coordonnées de textures générées ne lui plaisent pas, il doit pouvoir les changer comme bon lui semble. L'artiste pourrait également corriger les problèmes potentiellement introduits dans le processus de simplification.

En poussant davantage le concept de l'intervention de l'artiste, on peut imaginer une technique dans laquelle l'artiste spécifie directement la région du modèle à simplifier. Ainsi, il pourrait décider dans quelle région il y a trop de détails, et demander à l'application de réduire la complexité du modèle uniquement là où il le désire. Ceci permettrait de réduire la taille des

[1] Ceci s'explique en grande partie par le support matériel des textures ainsi que des textures multiples (*multi-textures*), lesquelles se superposent sur le modèle.

structures manipulées et offrirait également une plus grande latitude à l'artiste.

Une simplification dans laquelle l'artiste intervient doit cependant être étroitement liée à ses outils de modélisation. N'ayant évidemment pas le loisir de construire un modeleur 3D étoffé, nous avons dû abandonner très tôt cette idée. Il serait toutefois possible de construire notre système de simplification à l'intérieur d'un système de modélisation commercial comme *Maya*, *Softimage* ou *3D Studio Max*. Il s'agit ici d'une avenue envisageable, mais dont notre inexpérience entrave le cheminement.

6.1.4 Animation

Notre technique de simplification ne considère qu'une seule posture de l'animation. Une meilleure approche consisterait à additionner ensemble les métriques de chacune des postures de l'animation. Puisque seul l'ordre relatif des métriques importe (plus grand *versus* plus petit), la composition linéaire des coûts permettrait toujours de décrire la meilleure arête, mais non plus uniquement en termes d'impact sur le modèle, mais également en termes d'impact sur l'animation en entier. On pourrait même considérer plus d'une animation, et ainsi obtenir un nombre important de postures représentant mieux toutes les configurations possibles du modèle. Bien évidemment, cette approche multiplie les temps de calculs, et potentiellement les exigences mémoire aussi.

Une approche moins coûteuse consisterait à effectuer la simplification en considérant une série de postures extrêmes dans lesquelles les positions limites du squelette sont atteintes. L'utilisation d'une telle séquence permettrait d'obtenir des coûts représentant plus adéquatement l'impact potentiel global d'une fusion dans l'ensemble des postures que le squelette peut prendre. Ceci réduirait le nombre de postures nécessaires afin de représenter toute l'amplitude des configurations atteignables du modèle. Il est toutefois possible que certaines postures doivent être pondérées de façon à mieux refléter leur importance dans l'ensemble des postures possibles. Une étude plus approfondie aiderait probablement à identifier et mieux gérer de tels cas.

Cette extension relativement simple à implanter complèterait bien l'intégration de l'animation squelettique dans la simplification. En effet, nous avons développé une technique de simplification s'appliquant à des modèles articulés, mais nous n'avons pas utilisé ces modèles (i.e. leurs animations) pour la guider.

6.1.5 Sommets pondérés

Cette extension découle directement de l'extension précédente. Tout comme nous n'avons pas intégré l'animation dans les critères de simplification, nous ne l'avons pas utilisée non plus pour optimiser les valeurs en découlant. Ainsi, lors d'une fusion, nous conservons intacts les poids associés aux sommets. Cette approche est valide pour une simplification conservant un sous-ensemble des sommets originaux, c'est-à-dire ne tentant pas d'optimiser la position du sommet fusionné. Mais dès que le sommet résultant d'une fusion est différent d'un des deux sommets qui fusionnent, il se peut que le poids associé à ses représentations pondérées décrivent mal la déformation désirée.

Afin de trouver un meilleur poids, on pourrait tout simplement en essayer un nombre arbitraire et comparer les résultats dans les différentes postures. On conserverait ainsi l'ensemble des poids offrant les positions les plus proches des positions originales du modèle.

Cette approche permettrait probablement d'améliorer les résultats en offrant une déformation plus fidèle à la déformation initialement définie sur le modèle. Les coûts supplémentaires de calcul semblent passablement faibles dans la mesure où l'on peut précalculer toutes les matrices de transformation de toutes les postures. L'intégration des poids dans le calcul du sommet optimal permettrait probablement d'améliorer les résultats, mais le coût supplémentaire en temps de calcul est très variable.

6.1.6 Évaluation de la qualité

L'évaluation de la qualité de la simplification et, incidemment, des modèles simplifiés, demeure toujours un problème ouvert. Les outils de comparaison tels que *Metro*, développé par Cignoni *et al.* [CRS98], demeurent encore fort marginaux. Quant à eux, Lindstrom et Turk [LT00b] utilisent plutôt les images de rendu pour déterminer la qualité des modèles (voir section 2.5.3).

Quelle qu'elle soit, la technique d'évaluation utilisée doit satisfaire les besoins de la simplification. Si on désire uniquement obtenir un rendu similaire, l'approche de Lindstrom et Turk pourrait s'avérer préférable. Pour des données scientifiques, par exemple, une approche plus numérique, comme *Metro*, est probablement nécessaire. Une approche hybride pourrait certainement être développée, mais nous imaginons encore mal comment départager lorsque les évaluations se contredisent. Malgré tout, les techniques d'évaluation de la qualité de modèles demeurent fort discutées puisqu'on définit encore bien mal ce qu'est un bon modèle simplifié.

C'est pour cette raison, entre autres, que nous n'avons pas insisté sur l'évaluation de la qualité des modèles générés.

Nous sommes toutefois conscients que notre simplification, puisqu'elle s'applique à des modèles animés, ajoute une dimension au problème : la variation dans le temps. Puisque les modèles sont dynamiques, nous devons tenir compte de l'impact de la simplification non plus sur le modèle seul, mais aussi sur les différentes postures qu'il peut prendre. L'approche la plus simple pour tester une simplification articulée consisterait à appliquer n'importe laquelle des techniques d'évaluation de qualité des modèles statiques sur toutes les postures d'une animation. Ceci multiplie bien évidemment les temps de calcul, et chacune des postures pourrait être pondérée selon son importance dans les animations. L'utilisation d'une animation de postures extrêmes (comme en 6.1.4) pourrait s'avérer encore une fois utile.

6.2 Morphisme

6.2.1 Attributs

Il est clair que l'interpolation linéaire n'est pas toujours appropriée. L'exemple des normales indique clairement que la nature même des attributs peut parfois exiger une approche particulière. Comme nous l'avons déjà décrit à la section 5.3.2, une simple interpolation linéaire tend à couper court les normales. C'est pour cette raison que nous devons interpoler sur le grand arc, ce qui est évidemment plus coûteux. Similairement, une interpolation simple des couleurs dans l'espace RGB peut offrir des résultats étranges qui seraient partiellement réglés dans un espace perceptuel plus juste, comme $L^*u^*v^*$.

L'interpolation linéaire a cependant un avantage considérable : la simplicité. Ainsi, pour être utilisable, une technique plus adéquate devra performer comparablement à l'interpolation linéaire, sinon elle risque de rendre impraticable l'utilisation du morphisme en temps réel. Cette condition peut potentiellement empêcher certains critères d'être utilisés lors du morphisme (comme la conservation de volume, par exemple).

6.2.2 Morphisme généralisé

Par morphisme généralisé, nous n'entendons pas un morphisme entre deux modèles quelconques, mais plutôt entre n'importe lequel des modèles générés. En effet, il est possible d'effectuer directement un morphisme entre n'importe quelle paire de niveaux de détails.

L'approche envisagée est très similaire à la compaction des correspondances. Tout comme on pouvait faire correspondre directement $\mathbf{P}_A \rightarrow \mathbf{P}_C$ lorsque $\mathbf{P}_A \rightarrow \mathbf{P}_B \rightarrow \mathbf{P}_C$, on peut enchaîner les tables de correspondances afin d'en obtenir une seule qui permet de passer directement d'un modèle détaillé $\mathcal{M}_1$ à un modèle simplifié $\mathcal{M}_3$ sans savoir quoi que ce soit sur $\mathcal{M}_2$. Évidemment, cet enchaînement permet de construire les correspondances entre n'importe quelle paire de modèles $\mathcal{M}_i \rightarrow \mathcal{M}_{i+j}$.

6.2.3 Gestion des transitions

Le passage d'un modèle simplifié à un autre devrait être géré de façon automatique. Un contrôle des transitions basé sur la distance à la caméra semble être un critère adéquat pour guider le morphisme, à condition bien sûr que l'angle d'ouverture reste constant. Un critère intégrant les deux (distance à la caméra et angle d'ouverture) correspond tout simplement à considérer la taille du modèle dans la fenêtre de rendu.

Une approche naïve consisterait à spécifier une distance pour chaque niveau de détails. De là, on pourrait définir t comme étant la proportion de la distance parcourue du modèle dans l'intervalle dans lequel il se situe. Ceci entraînerait toutefois que le modèle subirait toujours un morphisme, puisqu'il serait toujours dans un intervalle (sauf quand $t = 0$ ou $t = 1$).

Une meilleure approche consisterait à définir des zones de transitions. Le morphisme ne serait calculé que lorsque le modèle se situe à l'intérieur de ces zones. À l'extérieur de ces zones, un seul niveau de détails (modèle articulé ne subissant aucun morphisme) serait utilisé. Cette approche permet d'amortir les coûts associés au morphisme en permettant à la majorité des modèles de ne pas en subir.

Utiliser de telles zones de transition nécessiterait toutefois d'étudier différentes configurations (taille des zones, distance entre les zones, etc.). En effet, le choix de ces zones est primordial tant sur l'aspect efficacité que sur l'aspect qualité. Une étude sérieuse permettrait probablement de construire des zones appropriées en considérant le temps de calcul dont l'application dispose pour effectuer le morphisme.

Puisque la valeur du paramètre t peut être arbitrairement fixée, il est possible de commencer un morphisme, puis de l'arrêter soudainement pour l'inverser et revenir au modèle original. Ceci peut évidemment être très utile lorsqu'un modèle oscille près d'une zone de transition.

6.2.4 Accélération matérielle

Il existe présentement des cartes accélératrices 3D permettant d'accélérer l'affichage des modèles squelettiques. Nous n'avons toutefois pas utilisé ces capacités à cause du morphisme. En effet, afin d'effectuer notre morphisme adéquatement, nous devons accéder aux positions des modèles détaillé et simplifié (voir section 5.3.1). De par la nature du matériel, il est impossible d'obtenir la position finale d'un sommet de modèle articulé s'il est calculé par la carte. Le matériel n'est bon que dans un sens : afficher rapidement. Lorsqu'il nous informe de son état interne, il subit une forte pénalité.

Nous évaluons toutefois la possibilité d'effectuer le morphisme complètement au niveau de la carte. Puisque de toute façon, l'accélération des modèles squelettiques n'est possible qu'en ayant tout le modèle au niveau de la carte, nous croyons qu'il est possible également d'effectuer le morphisme à ce niveau, là même où toutes les informations sont disponibles. Une collaboration étroite avec un fabricant de cartes est cependant préconisée.

Ces extensions constituent un échantillon relativement restreint des améliorations que nous aurions pu et que nous devrions faire. En rétrospective, nous prenons conscience que notre recherche s'inscrit dans un cadre très restreint et qu'en bout de ligne, il en reste encore beaucoup à faire.

Chapitre 7

Conclusion

Nous avons présenté une technique de simplification pouvant être appliquée à des modèles articulés. Notre technique utilise l'héritage de la simplification de modèles statiques. Elle transforme le modèle squelettique en un modèle statique, effectue la simplification sur le modèle statique, et répercute ensuite les structures de simplification sur le modèle squelettique. Cette approche a l'avantage de travailler sur un nombre réduit de sommets, puisqu'elle évite de manipuler inutilement les représentations pondérées des sommets du modèle articulé. Notre implantation effectue efficacement ces simplifications et permet de construire une série de niveaux de détails à partir d'un modèle articulé simple. Ces niveaux de détails permettent d'alléger les coûts d'affichage en réduisant la complexité du modèle à afficher. Le choix du niveau de détails s'effectue généralement en relation avec la taille du modèle à l'écran.

La transition effectuée entre deux niveaux de détails introduit cependant du surgissement à cause de la différence d'apparence plus ou moins marquée entre les modèles. Notre technique permet également d'effectuer du morphisme entre ces niveaux de détails, et réussit à améliorer considérablement la qualité des transitions. Notre morphisme s'effectue grâce à de simples interpolations linéaires entre les sommets et les attributs des niveaux de détails. Cette interpolation peut être étendue sur un temps arbitraire, mais nos résultats semblent indiquer que quelques secondes suffisent pour offrir une grande qualité. Les coûts reliés à ce calcul sont faibles en mémoire, mais moyen en calcul (un affichage avec morphisme prend environ deux fois plus de temps que sans morphisme). Puisque la transition s'effectue typiquement en quelques secondes au plus, la pénalité totale due au morphisme demeure toutefois raisonnablement faible. Malgré tout, notre implantation, qui n'est pas particulièrement optimisée, arrive toutefois à afficher en temps réel un modèle articulé relativement complexe subissant du

morphisme.

L'application de notre technique était dictée par le cadre de notre recherche. En effet, nous avons développé notre technique en analysant les besoins d'*Electronic Arts* (une compagnie de jeux vidéo). Notre technique pourrait donc servir à construire un ensemble de niveaux de détails articulés, tâche qui était originalement imputée à un artiste. Ces niveaux de détails peuvent ensuite être utilisés dans un jeu tel un titre de sport de façon à réduire la charge d'affichage (une dizaine de joueurs, par exemple). L'utilisation du morphisme permet d'améliorer la qualité de la transition entre deux niveaux de détails d'un même joueur. Quelques animations permettant de mieux juger l'apport du morphisme sont disponibles à l'adresse suivante : http ://www.iro.umontreal.ca/labs/infographie/theses/houlejo/. Finalement, l'utilisation du morphisme peut être contrôlée en fonction des ressources matérielles dont l'application dispose.

Nous sommes persuadés que la présente technique constitue une façon praticable d'améliorer les performances graphiques d'une application en temps réel. Nous sommes cependant conscients que notre implantation, dans son état actuel, n'est pas suffisamment efficace pour permettre du morphisme simultané sur un grand nombre de modèles. Une optimisation du code permettrait peut-être d'obtenir les performances nécessaires. Nous évaluons d'ailleurs la possibilité d'implanter de façon matérielle notre algorithme, ce qui offrirait en toute vraisemblance un gain majeur au point de vue de la performance.

Il est clair à notre esprit que notre recherche aidera à l'utilisation grandissante des modèles articulés multi-résolutions dans les applications en temps réel tels les jeux vidéo. Nous sommes aussi convaincus que le morphisme entre ces modèles constitue une façon avantageuse d'effectuer les transitions. Nous souhaitons d'ailleurs voir naître, dans les prochaines années, de nouvelles techniques repoussant toujours les limites courantes, tant au niveau de qualité de la simplification, qu'au niveau des transitions entre les représentations.

Bibliographie

[ACOL00] Marc Alexa, Daniel Cohen-Or et David Levin. « As-rigid-as-possible shape interpolation ». Dans *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, pages 157–164, juillet 2000.

[AL97] Daniel G. Aliaga et Anselmo A. Lastra. « Architectural walkthroughs using portal textures ». Dans *IEEE Visualization '97*, pages 355–362, novembre 1997.

[AL98] Daniel G. Aliaga et Anselmo A. Lastra. « Smooth transitions in texture-based simplification ». *Computers & Graphics*, volume 22, numéro 1, pages 71–81, février 1998.

[Ali96] Daniel G. Aliaga. « Visualization of complex models using dynamic texture-based simplification ». Dans *IEEE Visualization '96*, octobre 1996.

[Bar83] Brian A. Barsky. « A study of the parametric uniform B-spline curve and surface representations ». Technical Report UCB/CSD 83/118, Computer Science Division, Electrical Engineering and Computer Sciences Department, University of California, mai 1983.

[Béz66] Pierre Bézier. « Définition numérique des courbes et surfaces I ». *Automatisme*, volume XI, pages 625–632, 1966.

[BN92] Thaddeus Beier et Shawn Neely. « Feature-based image metamorphosis ». Dans *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 35–42, juillet 1992.

[Bre00] Nicolas Bredy. « Simplification de maillages et métriques d'erreur ». Rapport de stage, Université de Montréal, mai 2000.

[CC78] Edwin E. Catmull et James H. Clark. « Recursively generated B-spline surfaces on arbitrary topological meshes ». *Computer-Aided Design*, volume 10, pages 350–355, septembre 1978.

[CC98] Eyal Carmel et Daniel Cohen-Or. « Warp-guided object-space morphing ». *The Visual Computer*, volume 13, numéro 9–10, pages 46–478, 1998.

[Cla76] James H. Clark. « Hierarchical geometric models for visible surface algorithms ». *Communications of the ACM*, volume 19, numéro 10, pages 547–554, octobre 1976.

[CMO97] Jonathan Cohen, Dinesh Manocha et Marc Olano. « Simplifying polygonal models using successive mappings ». Dans *IEEE Visualization '97*, pages 395–402, novembre 1997.

[CMS98] P. Cignoni, C. Montani et R. Scopigno. « A comparison of mesh simplification algorithms ». *Computers & Graphics*, volume 22, numéro 1, pages 37–54, février 1998.

[COLS98] Daniel Cohen-Or, David Levin et Amira Solomovici. « Three dimensional distance field metamorphosis ». *ACM Transactions of Graphics*, volume 17, numéro 2, pages 116–141, avril 1998.

[Com89] Peter Comninos. « Fast bends or fast free-form deformation of polyhedral data ». Dans *Computer Graphics '89*, pages 225–242, novembre 1989.

[COM98] Jonathan Cohen, Marc Olano et Dinesh Manocha. « Appearance-preserving simplification ». Dans *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 115–122. Addison Wesley, juillet 1998.

[Coo74] Steven A. Coons. *Surface patches and B-spline curves*. Academic Press, 1974.

[Coq90] Sabine Coquillart. « Extended free-form deformation : a sculpturing tool for 3D geometric modeling ». Dans *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 187–196, août 1990.

[CPS97] P. Cignoni, E. Puppo et R. Scopigno. « Representation and visualization of terrain surfaces at variable resolution ». *The Visual Computer*, volume 13, numéro 5, pages 199–217, 1997.

[Cro82] Franklin C. Crow. « A more flexible image generation environment ». Dans *Computer Graphics (SIGGRAPH '82 Proceedings)*, volume 16, pages 9–18, juillet 1982.

[CRS98] P. Cignoni, C. Rocchini et R. Scopigno. « Metro : measuring error on simplified surfaces ». *Computer Graphics Forum, Proceedings of Eurographics '98*, volume 17, numéro 2, pages 167–174, juin 1998.

[CVM+96] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick P. Brooks, Jr. et William Wright. « Simplification envelopes ». Dans *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 119–128. Addison Wesley, août 1996.

[DSSD99] Xavier Decoret, Gernot Schaufler, François X. Sillion et Julie Dorsey. « Multi-layered impostors for accelerated rendering ». Dans *Proceedings of Eurographics '99*, septembre 1999.

[DSV97] Lucia Darsa, Bruno Costa Silva et Amitabh Varshney. « Navigating static environments using image-space simplification and morphing ». Dans *1997 Symposium on Interactive 3D Graphics*, pages 25–34, avril 1997.

[DSV98] Lucia Darsa, Bruno Costa Silva et Amitabh Varshney. « Walkthroughs of complex environments using image-based simplification ». *Computers & Graphics*, volume 22, numéro 1, pages 55–69, février 1998.

[DWS+97] Mark A. Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich et Mark B. Mineev-Weinstein. « ROAMing terrain : real-time optimally adapting meshes ». Dans *IEEE Visualization '97*, pages 81–88, novembre 1997.

[EDD+95] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery et Werner Stuetzle. « Multiresolution analysis of arbitrary meshes ». Dans *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 173–182. Addison Wesley, août 1995.

[EM99] Carl Erikson et Dinesh Manocha. « GAPS : general and automatic polygonal simplification ». Dans *ACM Symposium on Interactive 3D Graphics*, pages 79–88, avril 1999.

[Eri96] Carl Erikson. « Polygonal simplification : an overview ». Rapport technique TR96-016, University of North Carolina, Department of Computer Science, 1996.

[ESV96] Francine Evans, Steven S. Skiena et Amitabh Varshney. « Optimizing triangle strips for fast rendering ». Dans *IEEE Visualization '96*, octobre 1996.

[FDFH90] James D. Foley, Andries van Dam, Steven K. Feiner et John F. Hughes. *Computer graphics : principles and practice, 2nd edition*, chapitre 13, page 584. The Systems Programming Series. Addison-Wesley, 1990.

[Gar99] Michael Garland. « Multiresolution modeling : survey & future opportunities ». State of the art report (STAR), Eurographics '99, 1999.

[GG95] Eli Goldtein et Craig Gotsman. « Polygon morphing using a multiresolution representation ». Dans *Graphics Interface '95*, pages 247–254, mai 1995.

[GH94] Leonidas J. Guibas et John Hershberger. « Morphing simple polygons ». Dans *Proceedings of the Tenth Annual Symposium on Computational Geometry*, pages 267–276, juin 1994.

[GH97] Michael Garland et Paul S. Heckbert. « Surface simplification using quadric error metrics ». Dans *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 209–216. Addison Wesley, août 1997.

[GH98] Michael Garland et Paul S. Heckbert. « Simplifying surfaces with color and texture using quadric error metrics ». Dans *Visualization '98 Proceedings*, pages 263–269, octobre 1998.

[Gor69] W. Gordon. « Free-form surface interpolation through curve networks ». Rapport technique GMR-921, General Motors Research Laboratories, 1969.

[GR74] William J. Gordon et Richard F. Riesenfeld. « Bernstein-Bezier methods for the computer-aided design of free-form curves and surfaces ». *JACM*, volume 21, numéro 2, pages 293–310, avril 1974.

[GSL+99] Arthur Gregory, Andrei State, Ming C. Lin, Dinesh Manocha et Mark A. Livingston. « Interactive surface decomposition for polyhedral morphing ». *The Visual Computer*, volume 15, numéro 9, pages 453–470, 1999.

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald et Werner Stuetzle. « Surface reconstruction from unorganized points ». Dans *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 71–78, juillet 1992.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald et Werner Stuetzle. « Mesh optimization ». Dans *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 19–26, août 1993.

[Hop96] Hugues Hoppe. « Progressive meshes ». Dans *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. Addison Wesley, août 1996.

[Hop97] Hugues Hoppe. « View-dependent refinement of progressive meshes ». Dans *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 189–198. Addison Wesley, août 1997.

[Hop98a] Hughes Hoppe. « Smooth view-dependent level-of-detail control and its application to terrain rendering ». Dans *Visualization '98*, pages 35–42, octobre 1998.

[Hop98b] Hugues Hoppe. « Efficient implementation of progressive meshes ». *Computers & Graphics*, volume 22, numéro 1, pages 27–36, février 1998.

[Hop99] Hughes Hoppe. « New quadric metric for simplifying meshes with appearance attributes ». Dans *IEEE Visualization 1999*, pages 59–66, octobre 1999.

[Hug92] John F. Hughes. « Scheduled Fourier volume morphing ». Dans *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 43–46, juillet 1992.

[KCP92] James R. Kent, Wayne E. Carlson et Richard E. Parent. « Shape transformation for polyhedral objects ». Dans *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 47–54, juillet 1992.

[KSK98] Takashi Kanai, Hiromasa Suzuki et Fumihiko Kimura. « Three-dimensional geometric metamorphosis based on harmonic maps ». *The Visual Computer*, volume 14, numéro 4, pages 166–176, 1998.

[KSK00] Takashi Kanai, Hiromasa Suzuki et Fumihiko Kimura. « Metamorphosis of arbitrary triangular meshes ». *IEEE Computer Graphics & Applications*, volume 20, numéro 2, pages 62–75, mars – avril 2000.

[LC87] William E. Lorensen et Harvey E. Cline. « Marching cubes : A high resolution 3D surface construction algorithm ». Dans *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–169, juillet 1987.

[LCHS96] Seung-Yong Lee, Kyung-Yong Chwa, James Hahn et Sung Yong Shin. « Image morphing using deformation techniques ». *The Journal of Visualization and Computer Animation*, volume 7, numéro 1, pages 3–23, janvier-mars 1996.

[LDSS99] Aaron Lee, David Dobkin, Wim Sweldens et Peter Schröder. « Multiresolution mesh morphing ». Dans *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series, pages 343–350, août 1999.

[LDW97] Michael Lounsbery, Tony D. DeRose et Joe Warren. « Multiresolution analysis for surfaces of arbitrary topological type ». *ACM Transactions on Graphics*, volume 16, numéro 1, pages 34–73, janvier 1997.

[LE97] David Luebke et Carl Erikson. « View-dependent simplification of arbitrary polygonal environments ». Dans *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 199–208. Addison Wesley, août 1997.

[LKR+96] Peter Lindstrom, David Koller, William Ribarsky, Larry F. Hughes, Nick Faust et Gregory Turner. « Real-time, continuous level of detail rendering of height fields ». Dans *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 109–118. Addison Wesley, août 1996.

[LLM+90] Michael Lounsbery, Charles Loop, Stephen Mann, David Meyers, James Painter, Tony DeRose et Kenneth Sloan. « Testbed for the comparison of parametric surface methods ». Dans *Curves and Surfaces in Computer Vision and Graphics (Proceedings of SPIE)*, volume 1251, pages 94–105, 1990.

[Lou94] J. Michael Lounsbery. *Multiresolution analysis for surfaces of arbitrary topological type*. Ph.D. thesis, Department of Computer Science and Engineering, University of Washington, septembre 1994.

[LSS+98] Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar et David Dobkin. « MAPS : multiresolution adaptive parameterization of surfaces ». Dans *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 95–104, juillet 1998.

[LT97] Kok-Lim Low et Tiow-Seng Tan. « Model simplification using vertex-clustering ». Dans *1997 Symposium on Interactive 3D Graphics*, pages 75–82, avril 1997.

[LT98] Peter Lindstrom et Greg Turk. « Fast and memory efficient polygonal simplification ». Dans *Proceedings of IEEE Visualization '98*, pages 279–286, octobre 1998.

[LT99] Peter Lindstrom et Greg Turk. « Evaluation of memoryless simplification ». *IEEE Transactions on Visualization and Computer Graphics*, volume 5, numéro 2, pages 98–115, avril – juin 1999.

[LT00a] Peter Lindstrom et Greg Turk. « Image-driven mesh optimization ». Rapport technique GIT-GVU-00-11, Georgie Institute of Technology, College of Computing, juin 2000.

[LT00b] Peter Lindstrom et Greg Turk. « Image-driven simplification ». *ACM Transactions on Graphics*, volume 19, numéro 3, juillet 2000. À paraître.

[Lue96] David Luebke. « Hierarchical structures for dynamic polygonal simplification ». Rapport technique TR96-006, University of North Carolina, Department of Computer Science, janvier 1996.

[Lue97] David Luebke. « A survey of polygonal simplification algorithms ». Rapport technique TR97-045, University of North Carolina, Department of Computer Science, décembre 1997.

[LV98] Francis Lazarus et Anne Verroust. « Three-dimensional metamorphosis : a survey ». *The Visual Computer*, volume 14, numéro 8–9, pages 373–389, 1998.

[LWCS96] Seungyong Lee, George Wolberg, Kyung-Yong Chwa et Sung Yong Shin. « Image metamorphosis with scattered feature constraints ». *IEEE Transactions on Visualization and Computer Graphics*, volume 2, numéro 4, décembre 1996.

[MJT96] Chen Min, Mark W. Jones et Peter Townsend. « Volume distortion and morphing using disk fields ». *Computers & Graphics*, volume 20, numéro 4, pages 567–575, juillet 1996.

[MS95] Paulo W. C. Maciel et Peter Shirley. « Visual navigation of large environments using textured clusters ». Dans *1995 Symposium on Interactive 3D Graphics*, pages 95–102, avril 1995.

[RAPL98] Matthew M. Rafferty, Daniel G. Aliaga, Voicu Popescu et Anselmo A. Lastra. « Images for accelerating architectural walkthroughs ». *IEEE Computer Graphics & Applications*, volume 18, numéro 6, novembre – décembre 1998.

[RB92] Jarek R. Rossignac et Paul Borrel. « Multi-resolution 3D approximations for rendering complex scenes ». Rapport technique RC 17697 (#77951), IBM Research Division, 1992.

[RB93] Jarek Rossignac et Paul Borrel. « Multi-resolution 3D approximations for rendering complex scenes ». *Geometric Modeling in Computer Graphics*, pages 455–465, juin 1993.

[Ros95] Jarek Rossignac. « Geometric simplification ». Dans *Course notes 32 on "Interactive Walkthrough of Large Geometric Databases*, pages D1–D14. SIGGRAPH '95, 1995.

[RR96] Rémi Ronfard et Jarek Rossignac. « Full-range approximation of triangulated polyhedra ». *Computer Graphics Forum, Proceedings of Eurographics '96*, volume 15, numéro 3, pages 67–76, août 1996.

[SD96] Steven M. Seitz et Charles R. Dyer. « View morphing : synthesizing 3D metamorphoses using image transforms ». Dans *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 21–30. Addison Wesley, août 1996.

[SDB97] François Sillion, George Drettakis et Benoit Bodelet. « Efficient impostor manipulation for real-time visualization of urban scenery ». *Computer Graphics Forum, Proceedings of Eurographics '97*, volume 16, numéro 3, pages 207–218, août 1997.

[SF99] Dieter Schmalstieg et Anton Fuhrmann. « Coarse view-dependent levels of detail for hierarchical and deformable models ». Rapport technique TR-186-2-99-20, Vienna University of Technology, 1999.

[SG92] Thomas W. Sederberg et Eugene Greenwood. « A physically based approach to 2D shape blending ». Dans *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 25–34, juillet 1992.

[SGWM93] Thomas W. Sederberg, Peisheng Gao, Guojin Wang et Hong Mu. « 2D shape blending : an intrinsic solution to the vertex path problem ». Dans *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 15–18, août 1993.

[Sho85] Ken Shoemake. « Animating rotation with quaternion curves ». Dans *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 245–254, juillet 1985.

[Sho87] Ken Shoemake. « Quaternion calculus and fast animation, computer animation : 3-D motion specification and control ». Dans *SIGGRAPH 1987 Tutorial*, pages 101–121, 1987.

[SLS+96] Jonathan Shade, Dani Lischinski, David Salesin, Tony DeRose et John Snyder. « Hierarchical image caching for accelerated walkthroughs of complex environments ». Dans *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 75–82. Addison Wesley, août 1996.

[SP86] Thomas W. Sederberg et Scott R. Parry. « Free-form deformation of solid geometric models ». Dans *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 151–160, août 1986.

[SS96] Gernot Schaufler et Wolfgang Stürzlinger. « A three dimensional image cache for virtual reality ». *Computer Graphics Forum, Proceedings of Eurographics '96*, volume 15, numéro 3, pages 227–236, août 1996.

[Ste97] A. James Stewart. « Hierarchical visibility in terrains ». Dans *Eurographics Rendering Workshop 1997*, pages 217–228. Springer Wien, juin 1997.

[Ste98] A. James Stewart. « Fast horizon computation at all points of a terrain with visibility and shading applications ». *IEEE Transactions on Visualization and Computer Graphics*, volume 4, numéro 1, janvier – mars 1998.

[SZL92] William J. Schroeder, Jonathan A. Zarge et William E. Lorensen. « Decimation of triangle meshes ». Dans *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, juillet 1992.

[TO99] Greg Turk et James O'Brien. « Shape transformation using variational implicit functions ». Dans *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series, pages 335–342, août 1999.

[Tur92] Greg Turk. « Re-tiling polygonal surfaces ». Dans *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, juillet 1992.

[Wol98] George Wolberg. « Image morphing : a survey ». *The Visual Computer*, volume 14, numéro 8–9, pages 360–372, 1998.

[XESV97] Julie C. Xia, Jihad El-Sana et Amitabh Varshney. « Adaptive real-time level-of-detail-based rendering for polygonal models ». *IEEE Transactions on Visualization and Computer Graphics*, volume 3, numéro 2, avril–juin 1997.

[XV96] Julie C. Xia et Amitabh Varshney. « Dynamic view-dependent simplification for polygonal models ». Dans *IEEE Visualization '96*, octobre 1996.

[ZSH00] Malte Zöckler, Detlev Stalling et Hans-Christian Hege. « Fast and intuitive generation of geometric shape transitions ». *The Visual Computer*, volume 16, numéro 5, pages 241–253, 2000.