Cohérence temporelle dans le rendu de silhouette stylisée

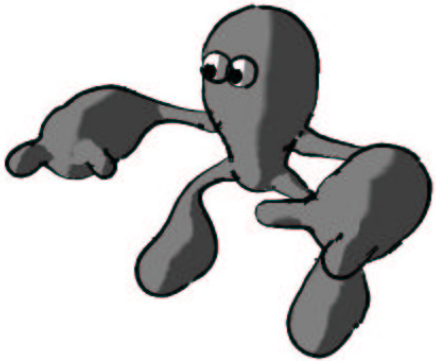
Jean-Francois Botalla-Gambetta

Troisième année ENSIMAG - INPG
DEA Imagerie, Vision, Robotique

Cohérence temporelle dans le rendu de silhouettes stylisées

Rapport de DEA

Jean-François BOTALLA-GAMBETTA



laboratoire GRAVIR (CNRS, INPG, INRIA, UJF)
équipe *i*MAGIS
sous la direction de Nicolas HOLZSCHUCH

Juin 2002

Merci à ...

Claude PUECH pour m’avoir accueilli dans son laboratoire et m’avoir permi de participer au symposium NPAR 2002.

Nicolas HOLZSCHUCH pour l’encadrement de ce stage.

Gilles DEBUNNE pour sa formidable *track-ball*, sa disponibilité et sa patience pour la vidéo.

Stéphane GRABLI pour ses conseils sur l’implémentation ainsi que sa relecture.

Mes relecteurs, Sylvain PARIS, Joëlle THOLLOT, Sandrine et Pierre.

Table des matières

Chapitre 1

Introduction et état de l'art

1.1 Le Rendu Non Photoréaliste

Le sujet de cette étude se situe dans le thème du rendu non photoréaliste. Il s'agit dans un premier temps d'un rendu : cette opération consiste à passer d'un modèle 3D à une représentation en 2D de ce modèle depuis un point de vue. L'accroissement de la vitesse des ordinateurs ainsi que l'augmentation des performances des cartes graphiques ont permi d'utiliser une partie de la puissance de calcul à la création d'effets et de rendus graphiques divers. La première tendance est le photoréalisme qui consiste à produire un rendu se rapprochant le plus fidèlement possible de la réalité (Fig 1.1).

FIG. 1.1 – Simulation d'un éclairage d'un bureau.

Le Rendu Non Photoréaliste (NPR) est devenu ensuite un thème de recherche important vers le début des années 90. Le NPR se place en opposition au rendu photoréaliste. D'une manière générale, le NPR tend à rassembler un ensemble de rendus expressifs et à développer des outils pour les artistes. Les limites du domaine sont pour l'instant assez mal définies comme le révèlent les nombreuses discutions autour de ce sujet comme celle lancée par David H. Salesin lors de son introduction au Symposium NPAR 2002 : en effet le NPR peut consister à imiter des techniques de dessin traditionel ou de peinture, en essayant d'utiliser les règles établies par les artistes. Le NPR regroupe aussi l'animation non-réaliste (avec la physique cartoon par exemple), ainsi que les algorithmes de composition automatique d'ornement ou encore le dessin technique avec des soucis de communication visuels (voir figure 1.3). Par exemple, de nombreux travaux menés par Winkenbach et Salesin ([WS94] et [WS96]) traîtent d'un rendu basé sur des ombrages réalisés avec des hachures : il s'agit du dessin à l'encre et au

crayon (*pen-and-ink* en anglais). Ils simulent le déplacement du crayon sur le papier (avec ses irrégularités) et construisent des textures procédurales dont la disposition des hachures suggère l'éclairage et la matière du modèle (figure 1.2).

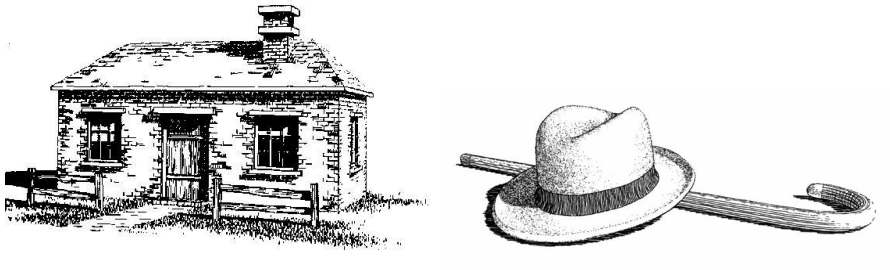


FIG. 1.2 – Deux différents type de shading ; respectivement à base de trait sur modèles polygônaux, à base de trait sur modêle paramétrique (issus de [WS94] et [WS96]).

FIG. 1.3 – Animation avec physique cartoon, algorithme d'ornement et dessin technique générés.

Le *cartoon shading* est l'ombrage d'un modèle basé sur l'utilisation d'une palette de couleurs restreinte (voir figure 1.4) : il s'agit d'une première caractéristique de la bande dessinée. Un autre aspect de la bande dessinée est l'utilisation d'un trait cerné foncé autour des modèles : il s'agit d'un liseret régulier et fin. Ces traits de silhouette peuvent également avoir l'apparence d'un trait peint avec un pinceau ou par une autre technique : on parle alors de trait stylisé. Cependant, de nouveaux problèmes apparaîssent avec ce type de trait : à cause de leur irrégularité (qui créent leurs styles) et du fait que l'œil soit sensible aux discontinuités, chaque saut, vibration ou écart d'un trait est immédiatement perçu. Il s'agit d'une incohérence temporelle. Dans une animation comportant des traits ayant un comportement discontinu, les perturbations sont visibles et donnent une impression de vibration des objets : il serait intéressant de contrôler ses problèmes et de créer des sèquences sans perturbations. Rendre les traits cohérents entre deux frames d'une animation est un problème compliqué qui n'a pas encore été résolu. Le sujet de mon stage se place entre le rendu avec un style cartoon et l'animation des images crées : nous tenterons d'apporter des débuts de solution pour obtenir une cohérence temporelle entre deux frames d'une séquence animée.

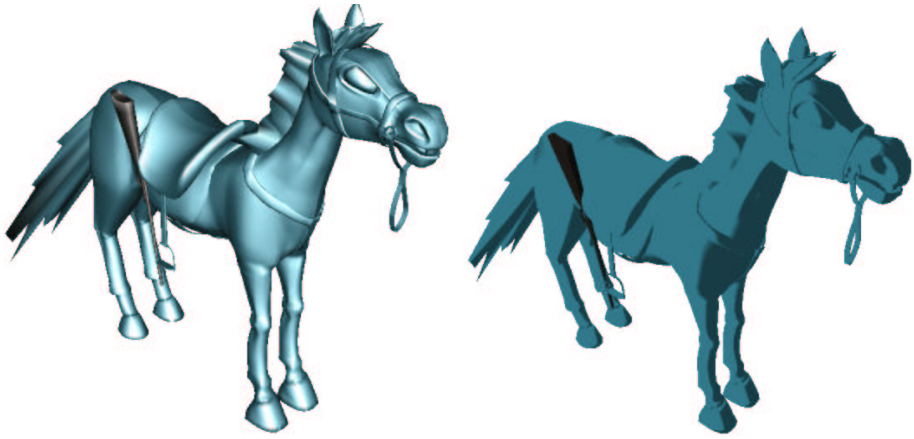


FIG. 1.4 – Ombrage de gouraud,.

1.2 Etat de l'art

1.2.1 L'aspect et l'ombrage des faces et des polygones

Le rendu des ombrages

Dans [LMHB00] Lake *et al.* décrivent un algorithme de rendu des couleurs à base d'aplats. Pour un matériau, deux couleurs (ou plus) sont calculées : une couleur claire (directement face à la lumiere) et une ombrée. Ensuite, suivant l'orientation de la face par rapport à la lumière, on décide d'appliquer la couleur claire ou la couleur sombre. Outre la facilité d'implémentation de cette technique, le rendu est assez rapide : l'algorithme utilise des modèles d'objets multi-résolutions. L'inconvénient est donc de ne pas pouvoir utiliser directement n'importe quel type de modèle pour ce rendu. Le résultat visuel obtenu est plutôt un dessin de style cartoon : ce type d'éclairage est également appelé *Cel-Shading* (Fig 1.5). L'ombrage du rendu recherché dans notre étude est inspiré de cette méthode.

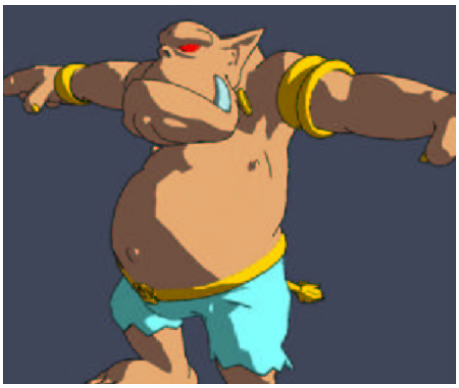


FIG. 1.5 – Ombrage à base d'applat issu de [LMHB00].

Couleurs non photoréalistes

Afin d'accentuer la compréhension d'une image, des travaux ont également été effectués sur des modèles d'éclairage non photoréaliste. Pour rendre la géométrie d'une image plus claire, Gooch *et al.* [GGSC98] proposent un modèle d'illumination non photoréaliste. En effet le modèle classique de Phong ne permet pas d'apprécier complètement la forme de l'objet : les parties ombrées, à cause de leur faible luminosité, sont mal ou difficilement interprétés par l'observateur. Ajouter les reflets et la silhouette lors du rendu, n'ajoute pas beaucoup d'informations, car les reflets seront perdus dans les zones très claires et les traits de silhouette seront difficilement perceptibles dans les parties sombres. Pour éviter d'avoir ce genre de problème et de rendre les couleurs de l'objet plus riches en informations, la technique employée par Gooch consiste à modifier l'intervalle de couleur. Tout d'abord en utilisant le concept de couleur chaude et couleur froide. Une couleur chaude (rouge, orange et jaune) donne l'impression que l'objet avance vers l'observateur, alors qu'au contraire une couleur froide (bleu, violet et vert) fait « reculer » l'objet. Un dégradé de couleur, du bleu pur vers le jaune pur, est créé. Par ailleurs, l'utilisation de nuance de couleurs permet d'avoir beaucoup d'informations avec un intervalle de couleurs restreint. Un second dégradé, du noir vers la couleur pure de l'objet, est généré. En mélangeant ces deux dégradés, on obtient une plage de couleurs moins réaliste, mais donnant une meilleure perception la forme de l'objet (figure 1.6). L'ajout des reflets et des contours à cette plage de couleurs est également possible. Gooch décrit aussi une technique pour rendre les reflets afin de donner à l'objet une texture particulière, comme par exemple le métal.

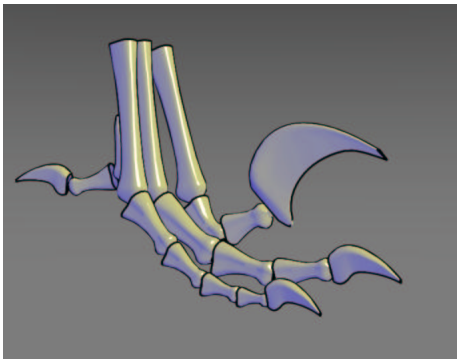


FIG. 1.6 – Eclairage avec couleur chaude et froide ([GGSC98]).

1.2.2 La détection et le rendu de contour

Un second aspect du rendu NPR avec un style bande dessinée est l'accentuation des contours et des plis de l'objet affiché. Il s'agit de tous les traits de silhouettes visibles, des plis (dans les tissus par exemple) et la transition entre deux zones de couleurs différentes.

Lake *et al.* décrivent dans [LMHB00] une technique de détection de contour. Cette technique traite toutes les arêtes du modèle : il s'agit d'un traitement brut. Une liste est créée en pré-traitement. Ensuite chaque arête subit des tests. Si l'angle formé par les deux faces communes à une arête est supérieure à un seuil, cette arête est considérée comme un pli. Si les deux faces d'une même arête ont des matériaux différents, il s'agit d'une arêtes de délimitation entre deux aplats. Enfin, si l'arête

est commune à une face avant (dont la normale pointe vers l'observateur) et à une face arrière, cette arête est un trait de silhouette. Les différents traits peuvent ensuite être affichés en même temps que les aplats. Pour styliser ces traits, Lake décrit une technique qui consiste à remplacer les arêtes par des traits courbés ou droits suivant l'angle de deux arêtes consécutives sur le contour. Le résultat donne des contours plus arrondis et plus naturels (voir les deux premiers dessins de la figure 1.8).

Alors que la plupart des algorithmes déjà décris travaillait avec l'espace objet, d'autre se servent de l'espace image. En effet, en utilisant la cohérence d'une image (c'est à dire, la continuité de la plupart des traits de silhouettes), des algorithmes travaillent sur des images intermédiaires au rendu. Philippe Decaudin dans [Dec96] détaille un des premiers algorithmes permettant un rendu style « dessin animé » d'une animation (voir figure 1.7). La détection de silhouette dans cet algorithme est inspirée des travaux de Saito et Takahashi (dans [ST90]). Elle consiste à effectuer un premier rendu hors écran (non affiché) afin d'obtenir une carte de profondeur (du Z-Buffer) de l'image. Ce résultat est ensuite traité par un filtre détection de contour classique : un gradient de l'image en Z est obtenu, et grâce à un seuil, la silhouette peut être détectée. Pour traiter les plis (ou arêtes vives), une image avec la représentation des normales est utilisée, et de la même manière que précédemment, les plis sont détectés grâce aux discontinuités. Le rendu obtenu est un renforcement des arêtes et des plis. L'inconvénient de cette méthode réside dans la nécessité d'effectuer plusieurs rendus intermédiaires : à chaque fois les transferts et le traitement des images ralentissent le processus. De plus aucune information 3D sur les arêtes trouvées ne peut etre obtenue de manière simple : ceci est un inconvénient important pour effectuer du style. Cette technique ne sera donc pas utilisée par notre algorithme.

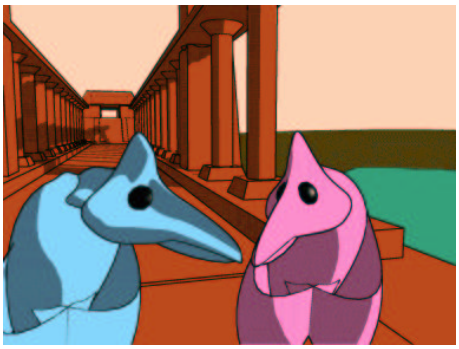


FIG. 1.7 – Rendu obtenu par les travaux de Decaudin dans [Dec96].

Comme les deux espaces (image et objet) comportent chacun des avantages, des recherches ont été menées afin de développer des algorithmes travaillant à la fois dans l'espace image et dans l'espace objet. C'est le cas de Northrup et Markosian dans [NM00]. En débutant dans l'espace objet, l'algorithme commence par détecter les bords de l'objet en regardant les faces avants et les faces arrières. Ici, au lieu de tester toutes les arêtes, un algorithme aléatoire tire et teste une arête. Si l'arête passe le test, on teste ces voisines. Cette technique donne de bons résultats sans avoir à tester tout le modèle. Ensuite, les arêtes effectivement visibles sont déterminées et une liste d'arêtes est construite. L'algorithme se sert ensuite de l'espace image afin de corriger les éventuels chevauchements entre deux arêtes consécutives sur un contour. L'espace image est beaucoup plus proche de la démarche du peintre avec son outil, et se prête donc mieux à la construction du trait en lui même. Un chemin régulier suivant la silhouette de l'objet est alors obtenu. Pour styliser ce contour, le chemin est transformé en bande de triangle qui va ensuite être texturée. Les textures sont générées et traitées afin de donner

l'impression d'un trait fait à la main (irrégularité, estompation). L'utilisateur peut alors travailler les textures de traits afin d'obtenir un rendu de style différent (voir la sandale de la figure 1.8). Northrup *et al.* ne traitent pas du comportement des traits entre deux images d'une animation : il y a des incorhérences et des sauts de trait. La méthode employée par notre algorithme est inspirée de cette technique, en essayant d'améliorer la cohérence temporelle des traits.

Un autre espace peut être également utilisé pour la détection de contour : l'espace dual [PBD+01]. L'intérêt ici de cet espace réside dans la quantité d'informations plus importante pour chaque élément. Une face étant représentée par un point dans l'espace dual, une arête est représentée par un segment. Les arêtes de silhouette sont recherchées dans l'espace dual en considérant les segments des arêtes duales qui intersectent un plan : ce plan dépend du vecteur de vision. Cette recherche peut être accélérée en utilisant un *octree*. L'algorithme de Barequet *et al.*, tire également partie des informations du point de vue pour calculer les silhouettes de la vue suivante, sans tout re-tester : les résulats obtenus sont rapides, mais l'implémentation est assez lourde.

Hang dans [ZI97] décrit lui une approche appelée *Gauss Map*, permettant de repérer rapidement les faces orientées vers l'arrière. Cette méthode, basée sur l'étude des normales projetées sur une sphère, peut être utilisée afin de trouver les faces avants et arrières partageant une même arête, comme le fait Gooch dans [GSG+99].

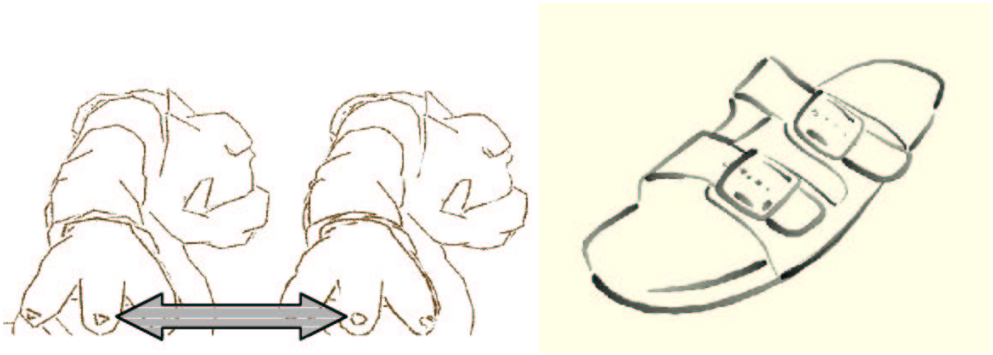


FIG. 1.8 – Contour simple d'un personnage, contour arrondis et rendu artistique des contours d'une sandale.

1.2.3 Cohérence temporelle

Pour l'instant, peu de travaux ont été menés sur la cohérence temporelle des silhouettes stylisées : les animations obtenues par les algorithmes existant ont l'inconvénient de provoquer des discontinuités temporelles ainsi que la vibration des silhouettes.
On peut citer les travaux de Kalnins *et al.* dans [KMM+02], qui utilise une technique permettant de contrôler une série de point le long du trait stylisé. La méthode consiste à rechercher la nouvelle position de ces points dans la nouvelle image. C'est cette partie du rendu des silhouettes qui sera plus approfondie dans cette étude.

1.2.4 Applications

Les applications visées par le rendu cartoon sont nombreuses. Tout d'abord le rendu style illustration schématique permet, en plus d'un certain rendu artistique, une meilleure compréhension visuelle : à l'opposé d'une photo (réaliste par définition) comportant beaucoup trop de détails inutiles pour la plupart, une illustration peinte, permet de ne représenter que les traits importants. L'attention de l'observateur n'est plus perturbé par les détails superflus. Des points, des parties ainsi que des détails peuvent être accentués et d'autres estompés. Le rendu NPR est donc capable de communiquer plus d'informations par rapport à une illustration réaliste classique. Les applications concernées peuvent être les dessins d'illustration d'un manuel d'utilisation, le dessin d'architecte ainsi que des applications de visualisation de projets.

Ce style de résultat convient cependant mieux à la bande dessinée ou au dessin animé : les applications sont alors la construction rapide de story-board, ou d'animations (film ou jeux vidéo).

Les jeux vidéos commencent à exploiter les techniques de rendu non photoréaliste dans leur moteur d'affichage temps réel. C'est le cas de Jet Set Radio Future sur XBox (Fig 1.9 a), et plus récemment d'Auto Modellista sur PlayStation 2 (Fig 1.9 b). Ces deux jeux ont la particularité d'avoir un éclairage de type *cell shading* ainsi qu'une détection de contour. Cependant dans Jet Set Radio Future seul les personnages bénéficient de ce rendu, les décors sont eux affichés en 3D avec un éclairage classique. Dans Auto Modellista les modèles traités sont des voitures ainsi que des éléments de décor simples : de part la simplicité des modèles, beaucoup de pré-calculs peuvent être utilisés. De plus, tout les jeux exploitant une détection de contour n'effectuent actuellement aucun rendu stylisé des silhouettes. L'épaisseur des lignes d'arête peut varier suivant la nature du trait (silhouette ou pli), mais l'utilisation de traits texturés imitant un pinceau ou un crayon n'a pas été commercialisée par l'industrie du jeux vidéo.

FIG. 1.9 – a.) Jet Set Radio Future sur XBox (c)2002 Sega et b.) Automodelista sur PlayStation 2 (c)2002 Capcom

L'industrie du film d'animation est également de plus en plus interessée par le NPR : des productions commencent à utiliser des rendus avec un style cartoon (Fig 1.10) et développent leurs propres outils de rendu. Les contraintes pour un film d'animation ne sont pas les mêmes et on ne parle plus de temps réel ni interactif. Pour l'instant, les productions avec un style particulier pour les traits sont

rares.

FIG. 1.10 – Molly : Star-Racer (c)2002 Sav ! The World

Chapitre 2

Méthodes utilisées

Dans ce chapitre nous décrirons le rendu : l'ombrage de type cartoon avec les aplats et le renforcement des silhouettes avec des traits stylisées. A la fin de cette section, la méthode de récupération des silhouettes et la structure géométrique nécessaire seront décrites. Ensuite l'étude se placera au niveau de l'animation de ce rendu avec le problème de la cohérence temporelle des traits, à savoir éviter les sauts, vibrations et apparitions de série de traits dans une frame.

2.1 Le rendu

2.1.1 Rendu des images fixes

Le rendu utilisé ici est un rendu avec un style bande dessinée. Les caractéristiques principales de ce type de rendu sont l'accentuation des arêtes de contour ainsi qu'un ombrage spécial, basé sur l'utilisation d'aplats. Nous allons décrire les techniques utilisées pour rendre ces deux caractéristiques.

Rendu par aplats

Aplat : Teinte plate, unie et soutenue sur toute sa surface.

Le rendu avec un style bande dessinée passe par l'utilisation de la technique du *cell shading* (ou *cartoon shading*), qui consiste à afficher le modèle en utilisant uniquement des aplats.

Par défaut, la carte graphique affiche les polygones avec un seul type d'ombrage (Gouraud). Pour obtenir un *cartoon shading* il faut donc faire faire le calcul des couleurs et des ombrages par le logiciel (Fig 2.1).

La lumière ne provoque pas de dégradé sur les surfaces : le but d'un ombrage et des couleurs est, entre autres, de révéler le volume du modèle. Un ombrage simplifié, basé sur des aplats, ne comporte en général que quelques nuances de la même couleur, par opposition à l'ombrage classique tel que l'ombrage de Phong ou Gouraud avec ses dégradés. Les volumes ainsi que les indications de sources lumineuses sont donc suggérés par l'utilisation de plusieurs aplats (deux en général) avec différents tons d'une même couleur : par exemple un aplat d'une couleur est utilisé pour les surfaces faisant faces à la lumière, et un autre aplat avec la même couleur mais un ton plus sombre est utilisé pour une

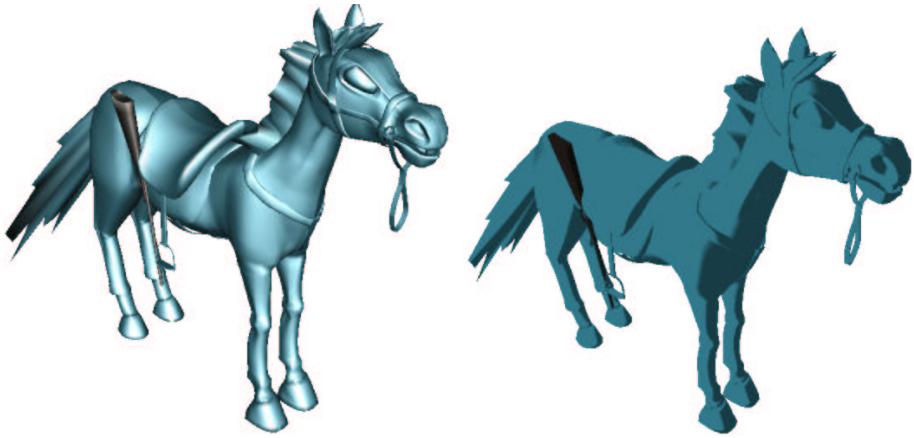


FIG. 2.1 – Poney avec un ombrage de Gouraud (à gauche) et un ombrage cartoon (à droite)

surface avec un matériau identique mais ombré. Ce faible nombre de nuance accentue les contrastes, et rend donc le dessin plus lisible. L'algorithme permettant ce rendu est ici décrit.

On utilisera maintenant la notation $\overrightarrow{V}$ pour le vecteur lumière, c'est à dire, le vecteur qui part la source lumineuse et qui arrive sur un point de la scène.

– La méthode utilisée consiste à parcourir toutes les faces du modèle et à effectuer un test d'orientation par rapport à la source lumineuse.

  Les faces dont la normale forme un angle suffisamment grand avec le vecteur de lumière, sont faces à la source lumineuse, donc claires : les autres sont ombrées (figure 2.2). Le résultat du produit scalaire $\overrightarrow{N}.\overrightarrow{V}$ détermine la couleur de la face : on s'en sert comme coordonnée u dans une texture 1D. La formule suivante permet d'obtenir cette coordonnée u :$u = \frac{\overrightarrow{N}.\overrightarrow{V}}{2} + 0.5$

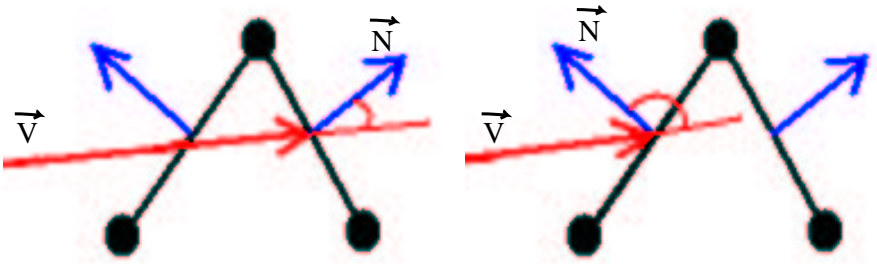


FIG. 2.2 – A gauche la face testée est ombrée, et à droite elle est claire

Il existe cependant un inconvénient à cette méthode : le bord de l'aplat de couleur se trouve morcelé.
Le calcul s'effectue par vertex : chaque sommet a la normale de la face (figure 2.3). Lors de l'affichage d'une face, les coordonnées de texture permettent d'assigner une couleur par vertex. La carte graphique effectue ensuite une interpolation entre les couleurs des trois sommets d'un

triangle. Ceci à pour effet de gommer légèrement le morcellement de l'aplat par une petite zone de transition entre deux couleurs : l'aplat semble plus lisse.

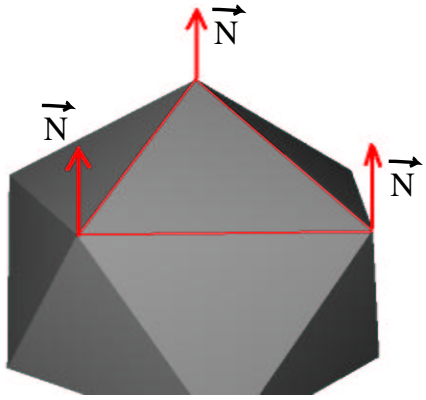


FIG. 2.3 – Les sommets d'une face possèdent également le vecteur normal de cette face

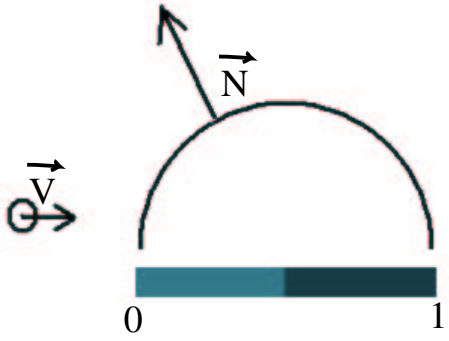


FIG. 2.4 – Fonctionnement global du cartoon shading

Les coordonnées sont calculées à chaque frame, ce qui permet de gérer une source lumineuse en mouvement. L'avantage est de pouvoir mieux apprécier les formes quand l'ombrage n'est pas fixe et que les zones ombrées évoluent sur la surface (comme le préconise Gooch dans [GSG+99]).
Ce calcul est donc rapide : la méthode comporte cependant un inconvénient : l'auto-ombrage ainsi que les ombres portées ne sont pas traités par l'algorithme.

– Les coordonnées obtenues par le test sur les faces servent ensuite à choisir la bonne tonalité dans une texture 1D. Ces textures sont générées en pré-traitement.
Pour obtenir cet effet sur les modèles, il faut tout d'abord traiter les couleurs propres des objets. La composante diffuse de chaque objet du modèle est extraite afin de créer une texture 1D (comme dans [LMHB00]). A partir de la couleur principale, une ou plusieurs nuances sont calculées. On a choisi d'assombrire la couleur principale afin de créer la couleur ombrée.
La répartition et le nombre de nuance peuvent être plus importants suivant le dessin. Par exemple, la peau du personnage de la figure 2.6 pourrait avoir une texture 1D comportant trois nuances à des luminosités différentes (2.8).
Ce traitement est effectué une seule fois lors du chargement du modèle : à chaque matériau

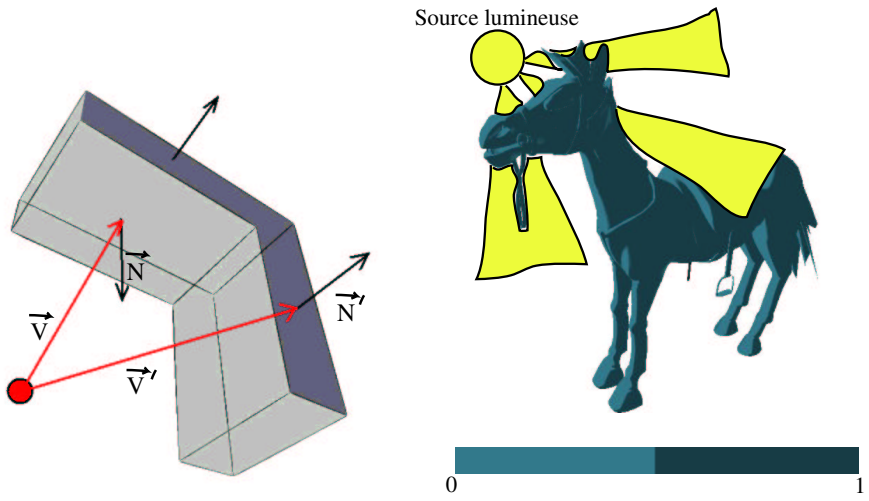


FIG. 2.5 – Exemple d'ombrage avec un *cell shading*

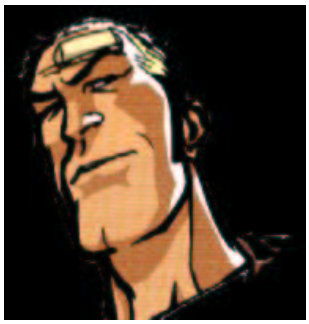


FIG. 2.6 – 7 Secondes de Gérald Parel Edition Delcourt

FIG. 2.7 – Exemple de texture 1D

FIG. 2.8 – Texture 1D qu'aurait le personnage de la bande dessinée 7 Secondes (Fig 2.6)

correspond une texture 1D.

Naturellement, il est nécessaire pour la suite de désactiver le rendu des éclairages et des ombrages par la carte graphique : l'ombrage sera rendu uniquement par la couleur de la texture.

Rendu d'arête

« Pour évoquer, faire sentir à l'oeil les formes, nous allons donc devoir les entourer d'un trait. Sans doute, il n'existe pas en réalité de cerne autour des objets. Ceux-ci, comme les êtres, les animaux, les plantes, sont libres dans l'espace, aucune ligne noire n'en limite les formes. Mais nous y aurons recours pour représenter sur le papier le contour de celles-ci »
Cours de Dessin-Peinture Ecole ABC de Paris

Pour l'instant, nous décrivons les techniques de rendu des silhouettes d'un modèle 3D. Les cartes graphiques ne savent pas dessiner directement des traits stylisés. Pour afficher et donner du style aux silhouettes et aux plis, il faut construire une bande polygonale le long des traits de silhouette afin d'avoir un support pour une texture de trait : le style en lui même sera donné par la texture.

Le rendu stylisé des arêtes de silhouettes est basé sur l'utilisation de textures de trait. La démarche consiste, à partir de la liste des segments de silhouette obtenus et projetés, à constuire un chemin. Les segments sont chaînés entre eux, afin de former une ligne brisée. Ensuite une bande de polygone est crée puis texturée.

- Dans un premier temps l'algorithme utilisé parcours la chaîne en cherchant de nouveaux segments parmi les arêtes de silhouette ou de pli.
- Le dernier segment d'un trait trouvé (les critères de césure du trait seront abordés plus loin), l'algorithme remonte la chaîne, en construisant un tronçon pour chaque segment.

Définition 1 On appelle tronçon l'élement unitaire de bande de polygones (Fig 2.9).

Les sommets $Pt1$,$Pt2$,$Pt3$ et $Pt4$ des triangles du tronçon sont déterminés à partir des sommets de l'arête Pt_debut et Pt_fin et du vecteur $\overrightarrow{N}$ orthogonal à $\overrightarrow{V}$:

$$\begin{aligned} Pt1 &= Pt\_debut + \overrightarrow{N} \\ Pt2 &= Pt\_fin + \overrightarrow{N} \\ Pt3 &= Pt\_debut - \overrightarrow{N} \\ Pt4 &= Pt\_fin - \overrightarrow{N} \end{aligned}$$

Lors du chaînage de deux tronçons consécutifs, si l'angle entre les deux segments est faible, des chevauchements peuvent apparaître, ainsi que des trous dans le trait (voir figure 2.11 premier dessin). Ces chevauchements deviennent visibles si la texture utilisée est translucide : le polygone de chevauchement est affiché deux fois, et est donc plus opaque. Pour résoudre ces problèmes, un triangle est créé afin de combler le trou laissé par les deux bandes au niveau d'un angle. Les sommets sont également modifiés pour corriger le chevauchement (voir le second dessin de la figure 2.11). Dans [NM00], Northrup et Markosian n'utilisent pas de polygones supplémentaires pour la jonction : ils

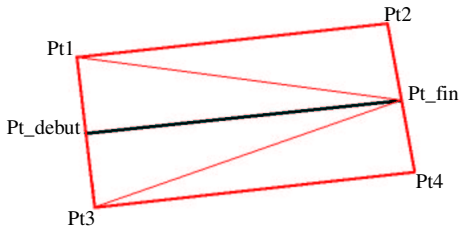


FIG. 2.9 – Un tronçon de la bande de polygone

interpôlent la normale au niveau de l'angle (2.10). Le rajout d'un triangle rend la bande plus lisse et déforme moins la texture utilisée.

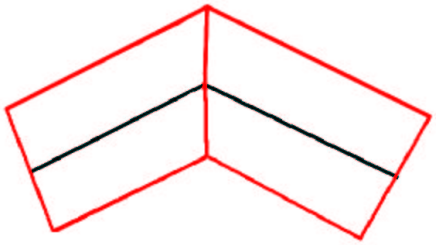


FIG. 2.10 – Gestion de deux polygones consécutifs par Northrup et Markosian.

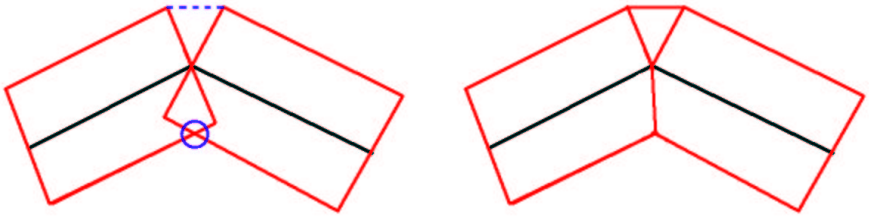


FIG. 2.11 – Problèmes de jonction entre deux tronçons consécutifs

Pour connecter correctement deux tronçon, il faut déterminer la configuration possible des deux vecteurs (Fig 2.12). Ceci est d'eterminé grâce au signe du produit scalaire $\overrightarrow{N}.\overrightarrow{V_2}$

Il faut ensuite gérer le problème de chevauchement. La configuration de l'angle entre les deux arêtes étant connue, il suffit de calculer l'intersection entre les deux triangles qui se chevauchent (voir figure 2.11).

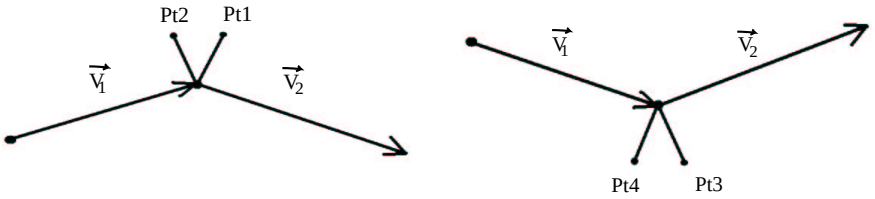


FIG. 2.12 – Deux configurations possibles

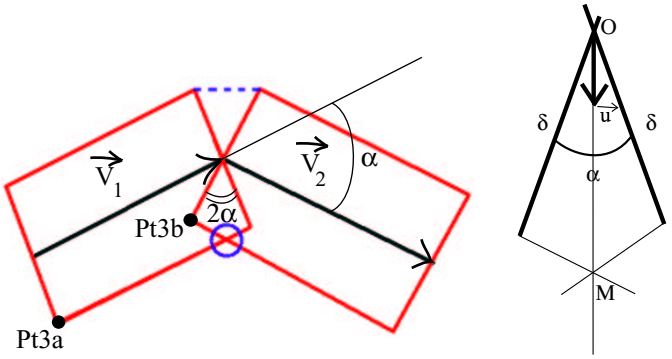


FIG. 2.13 – Intersection des deux tronçons : à droite l'aggrandissement sur le triangle de chevauchement

$$
\begin{aligned}
OM &= \frac{\delta}{sin(\frac{\alpha}{2})} = \beta \\
\overrightarrow{OM} &= \beta . \frac{\vec{u}}{\|\vec{u}\|} \\
\text{avec } \vec{u} &= \frac{\vec{v_1}}{\|\vec{v_1}\|} + \frac{\vec{v_2}}{\|\vec{v_2}\|}
\end{aligned}
\tag{2.1}
$$

Un Exemple de construction d'une chaîne de trois tronçons se trouve à la figure 2.14.

Une fois la bande de triangle créee, vient la partie artistique du processus de rendu. Le début de chaque trait a été stocké lors de leur création. L'étape de rendu consiste à parcourir la liste chaînée de tronçon et à afficher les différents triangles.

Pour cela une texture est utilisée. Cette texture peut être un trait au pinceau scanné (Fig 2.15) ou encore un trait réalisé avec l'aide d'un logiciel de graphisme.

Ce trait est texturé sur plusieurs polygones consécutifs (figure 2.16) : il faut donc déterminer les coordonnées (u,v) dans la texture 2D (voir annexe pour les détails d'implémentation).

Les problèmes de visibilité seront résolus par le logiciel (une section suivante décrit la méthode),

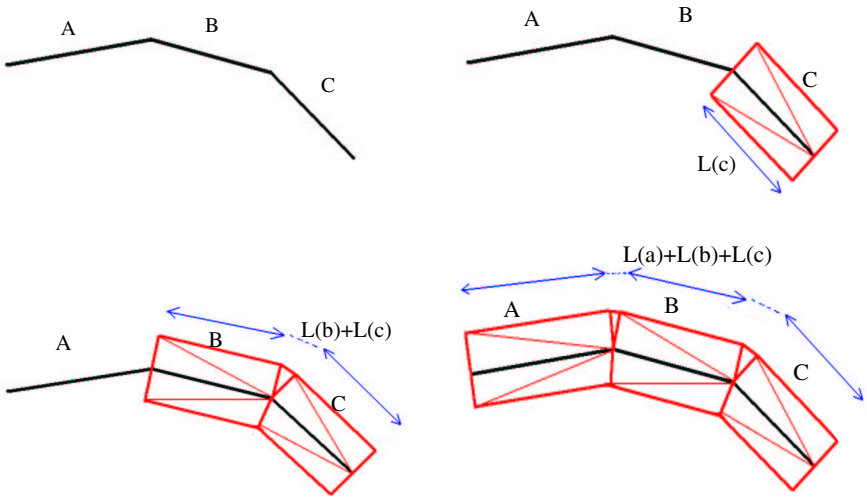


FIG. 2.14 – Construction d'une bande de polygones avec trois tronçon

FIG. 2.15 – Trait obtenu au pinceau et à l'aquarelle noire.

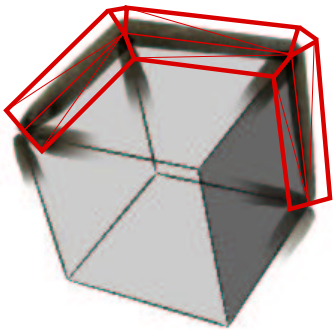


FIG. 2.16 – Une bande de polygones de trois tronçons texturée par un trait.

le test de visibilité de la carte graphique peut être désactivé.

Les traits sont tracés sur un fond blanc, que ce soit sur une feuille de dessin ou un fichier image. Il est donc nécessaire de rendre transparent ce fond blanc, lors de l'affichage des traits de silhouette. Pour cela on effectue un mélange avec la couleur déja proésente.

La fonction de mélange est toujours la même, pour une composante de couleur :

$$couleur = min(1, c * a + c' * b)$$

avec *couleur* l'intensité de la composante à afficher.
c l'intensité de la composante précédente de cet emplacement.
c' l'intensité de la composante candidate à l'affichage.

Il faut qu'une couleur blanche soit transparente : l'anciennne couleur est conservée. Au contraire une couleur noire assombrira la couleur déjà présente. La multiplication de l'ancienne couleur par la nouvelle couleur respecte ces comportements : à savoir un couleur noire ($c' = 0$) donne après mélange une couleur sombre ($c * c' = 0$) et une couleur blanche ($c' = 1$) conserve la même couleur ($c * c' = c$).

On choisi donc $a = 0$ et $b = c'$.

Pour les couleurs grises on obtient un effet transparent assez intéressant.

2.1.2 Détection des arêtes de contour et élimination des arêtes cachées

Nous allons maintenant décrire la méthode pour extraire les silhouettes et le pli afin d'obtenir la liste des arêtes de contour. L'algoritme de détection commence par créer une liste d'arêtes qu'il faut ensuite filtrer avec un test de visibilité.

Détection de contour

Les arêtes participant à la silhouette sont détectées sur le modèle en 3D (dans l'espace objet). Il s'agit d'un test sur les normales des deux faces adjacentes à une arête. Les informations d'adjacence sont nécessaires à l'algorithme, c'est donc une structure de données particulière qui sera utilisée (appelée *winged edge* et qui sera décrite dans la section géométrie).

Tout d'abord, quelques définitions géométriques :

Définition 1 Un polygone est un polygone avant, si sa normale est orientée vers l'utilisateur.

Définition 2 Un polygone est un polygone avant, si sa normale n'est pas orientée vers l'utilisateur.

Définition 3 Une arête appartient à la silhouette si elle est adajacente à un polygone avant et à un polygone arrière.

On suppose dans la suite que l'on a à faire à des surfaces de variétés et donc qu'une arête est bordée par deux triangles au plus.

L'algorithme procède en parcourant toutes les arêtes du modèle.

Le test effectué sur chaque arête consiste à examiner le signe du produit scalaire des vecteurs normals. aux deux faces avec les vecteurs de vue respectifs, c'est à dire $\overrightarrow{N}.\overrightarrow{V}$ et $\overrightarrow{N'}.\overrightarrow{V'}$

D'après les deux premières définitions : le produit scalaire du vecteur de vision avec la normale à la face permet de déterminer l'orientation de la surface :
Si $\overrightarrow{N}.\overrightarrow{V} > 0$ alors le polygone est orienté vers l'arrière.
Si $\overrightarrow{N}.\overrightarrow{V} < 0$ alors le polygone est orienté vers l'avant.
Si $\overrightarrow{N}.\overrightarrow{V} = 0$ le polygone est parallèle à la direction de vision

D'après la définition 3 on détermine ensuite les arêtes de silhouettes.

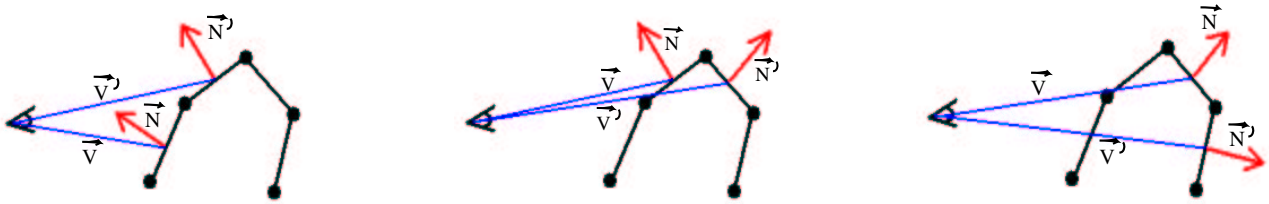


FIG. 2.17 – Configurations possibles lors de la détection de silhouette : la première figure est le cas d'une arête ayant ses deux faces orientées vers l'observateur. La seconde figure est le cas d'une arête ayant une face avant et une face arrière : il s'agit donc d'une arête de silhouette. La troisième figure est le cas d'une arête ayant deux faces arrières.

Définition 4 Une arête est une arête de pli si l'angle formé par ses deux faces adjacentes est plus faible qu'un seuil donné.

En utilisant le produit scalaire $\overrightarrow{N}.\overrightarrow{N'}$, on identifie les arêtes de pli.

FIG. 2.18 – Configurations possibles lors de la détection de pli : dans le premier cas, l'angle est insuffisant pour une arête de pli. Dans le second, le seuil est dépassé : il s'agit d'un pli.

Pour une arête de silhouettes détectée, les deux sommets sont projetés : des coordonnées 2D sont obtenues. Au fur et à mesure de la détection d'arêtes, une structure de données similaire à la structure *winged edge* (décrite plus loin) est mise à jour. Cette structure stocke les arêtes de silhouettes projetées dans l'espace image ; les informations d'adjacences issues du *winged edge* sont intégrées.

La nouvelle structure contient donc des segments liés entre eux, formant la silhouette du modèle observé.

Le même traitement est effectué pour les plis.

Ce passage dans l'espace image est nécessaire au rendu stylisé : en effet, la paramétrisation des arêtes doit s'effectuer dans l'espace image, car elle concerne les traits. Par exemple, une arête ayant une longueur de 1 mètre peut avoir, sous un certain angle une longueur apparente de quelques centimètres. Ce qui est important pour la paramétrisation des bandes de polygones, c'est la longueur apparente du trait (c'est à dire la longueur du trait sur le papier ou la photo), et non pas la longueur réelle de l'arête sur l'objet.

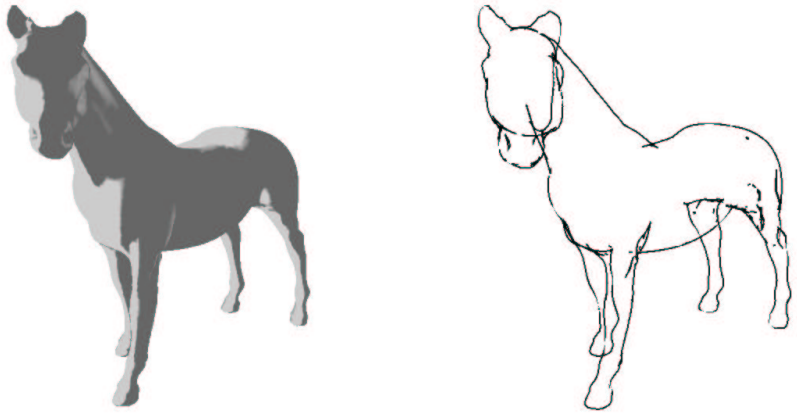


FIG. 2.19 – Détection de contour sans traitement de la visibilité d'un modèle de cheval

Elimination des silhouettes cachées

Les traits étant calculés par le programme et affichés en 2D, les problèmes de visibilité ne sont pas résolus par la carte graphique comme les autres éléments affichés. De plus la construction des chemins des traits ne doit utiliser que des arêtes visibles. Il est donc nécessaire de filtrer la liste des arêtes trouvées dans la partie précédente, pour éliminer celles qui sont occultées.

Pour une arête donnée, les deux sommets sont examinés. Lors de la projection (dans l'espace image) des extrémités d'un trait, une coordonnée en Z était également obtenue : cette valeur correspond à la valeur que la carte aurait placé dans le Z-buffer si le point avait été affiché à l'écran. Le rendu des polygones pleins ayant déjà été effectué le Z-buffer contient la profondeur des aplats. Pour connaître la visibilté du point, il suffit de comparer cette valeur Z avec la valeur lue dans le Z-buffer au même emplacement.

Le Z-buffer contient des flottants entre 0 et 1. 0 correspond a un pixel situé sur le plan de projection, et 1 correspond a un point à l'infini.

Si $A.z > Z - buffer(A.x, A.y)$ le point est caché.
Sinon le point est visible.

La visibilité des deux extrémités d'un segment est ainsi déterminée. Un trait est considéré comme

caché si ses sommets sont tous les deux invisibles ; sinon, le trait est considéré comme visible.

Avec ce test, seuls les cas de la figure 2.20 (la légende des couleurs est sur la figure 2.22) sont traîtés. Il s'agit des cas les plus courants dans les modèles complexes. Les cas de la figure 2.21 ne peuvent être résolus avec cette méthode : ce genre de cas apparaît lorsqu'il y a des arêtes longues (ce qui est rare pour les modèles dont le maillage est très détaillé) ; ils se produisent le long des silhouettes, car ils concèrnent toutes les arêtes à moitiés cachées (coupant donc une silhouette). Dans la plupart des cas, le résultat visuel n'est pas trop affecté par ces artéfacts : de plus il ne s'agit pas ici du thème étudié.

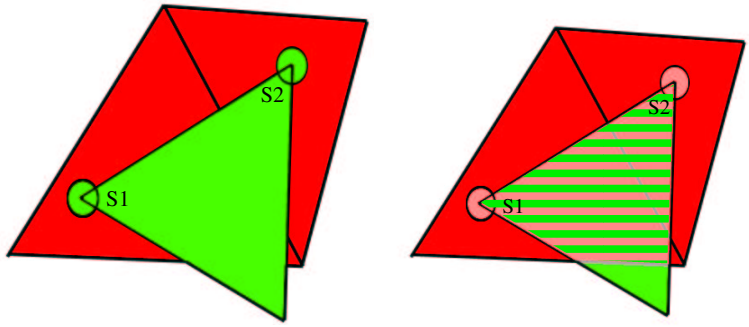


FIG. 2.20 – Cas pris en compte par la détection de silhouette visible implémentée

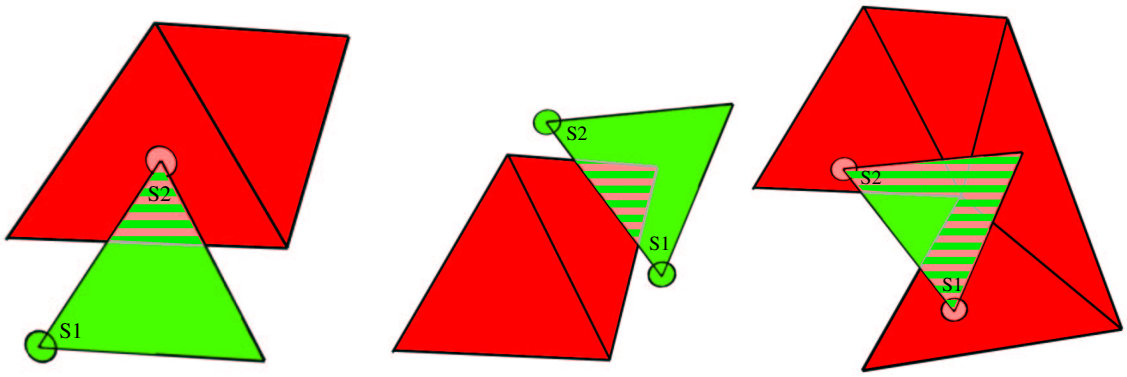


FIG. 2.21 – Cas non pris en compte par la détection de silhouette visible implémentée

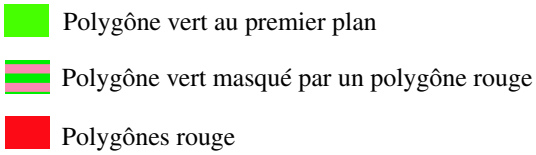


FIG. 2.22 – Légende des cas de visibilité

2.1.3 Structure géométrique

Dans le cadre de la détection de silhouettes et de plis, il est nécessaire de connaître certaines informations géométriques du modèle. Le graphe de scène du fichier 3D ne fournissant pas de renseignements géométriques, ce type d'information est généré puis stocké dans une structure de données particulière appelée *winged edge*. Introduite par Baumgart dans [Bau72], il s'agit d'une des plus anciennes structures pour les B-rep (Boundary Representation) : ce sont des approches qui consistent à modéliser les objets par une discrétisation de leur frontière.

Cette structure peut être utilisée pour toutes les faces sans trous, ce qui est le cas de nos modèles 3D. Il existe cependant des extensions afin de prendre en compte les configurations avec trous.

Le modèle est composé d'une hiérarchie d'éléments à différent niveau. On distingue les objets, les faces, les arêtes et les sommets. Un objet est naturellement composé de plusieurs faces, qui sont elles-mêmes composées de plusieurs arêtes. Une arête quelconque est elle composée de deux sommets.

Dans la structure *winged edge*, les arêtes sont orientées. Par définition, une face est composée des arêtes orientées dans le sens inverse des aiguilles d'une montre vu de l'extérieur du solide (Fig 2.23).

FIG. 2.23 – Sens d'orientation des arêtes d'une face

Pour une arête orientée donnée, la face située à sa gauche s'appelle la face A et la face de droite la face B (Fig 2.24).

FIG. 2.24 – Faces adjacentes à une arête

Une arête orientée possède donc deux sommets, une ou deux faces adjacentes et une arête non-orientée est composée de une ou deux arêtes orientées (une dans chaque sens ou une seule si l'arête n'a qu'une face voisine).

Voir la structure de classe en annexe. L'avantage d'utiliser une telle structure réside dans la facilité et la rapidité d'accès aux différents éléments du modèle. A partir d'un sommet du modèle, on peut retrouver les arêtes orientées partant de ce point, et ainsi connaître les faces ayant comme point commun ce sommet. Comme la structure contient beaucoup d'informations elle prend de la place en mémoire : mais pour la taille assez faible des modèles traités, cela ne représente pas un réel problème.

Pour les besoins de l'algorithme de rendu, des structures de données simplifiées et dérivées des *winged edges* ont également été utilisées : il s'agit d'une structure appelée *silhouette edge* qui comporte des arêtes orientées, des arêtes non orientées ainsi que des sommets. Les coordonnées des sommets sont dans l'espace image. En fait, cette structure est utilisée juste après la détection des silhouettes et des plis, lors du chaînage des segments. Une seconde structure dérivée est construite après l'étape de chaînage : cette structure contient des tronçons et des sommets.

2.2 Critères de césure et cohérence temporelle

Les critères de césure seront étudié dans un premier temps. Puis, les différentes méthodes que nous proposons pour aller vers la cohérence temporelle des traits seront décrites.

2.2.1 Critères de césure

Lors de la création des traits le long de la silhouette, il est important de placer ces traits de manière réaliste. Or, la disposition des traits, l'épaisseur, les choix de fin de trait sont autant de paramètres qui appartiennent au style. Il s'agit d'un sujet d'étude à part entière : il faut examiner les décisions de l'artiste lors du dessin.
Quelques aspects simplifiés seulement ont été ici étudiés.

Un trait simple peut se décomposer en trois parties. La tête du trait correspond à l'endroit où le pinceau est posé sur la feuille de papier. Le corps du pinceau est la partie régulière du trait : quand un trait s'allonge, c'est cette partie qui doit être allongée. Enfin la queue du trait correspond à la fin du trait quand le pinceau est relevé : le trait s'affine et devient moins couvrant. Par ailleurs, un trait peut

FIG. 2.25 – Les différentes parties du trait

avoir plusieurs aspects très différent : la tête peut être effilée, ou encore la tête et la queue peuvent être suprimés (voir les exemples de la figure 2.26).

La choix de la césure du trait intervient pendant l'étape de chaînage. Le question qui se pose est « quand doit on arrêter le trait ». Lorsque l'on peint un angle aigu, afin de mieux évoquer une pointe, il

FIG. 2.26 – Différents types de trait

faut affiner le trait lors du tracé au niveau de l'angle. Si le trait n'est pas interompu, le coup de pinceau reste large au niveau l'angle : l'impression de pointe sera perdue (voir Fig 2.27 a.)).
Si le trait est interrompu sur l'angle, celui-ci sera affiné (si l'on utilise une texture de pinceau avec une queue effilée) et aura un aspect pointu (voir Fig 2.27 b.). L'œil humain est très sensible aux discontinuités. Les travaux de psychologie de Kellman et Shipley (dans [KS91]) ont mis en évidence que l'œil considère comme continue deux segments (dont l'intersection est masquée) lorsque ceux-ci forme un angle supérieur à 90 degrés. Inversément deux segments avec un angle inférieur à 90 degrés sont perçus comme deux segments différents, avec donc une discontinuité. Bien entendu ce cas est différent du notre, car nos angles ne sont pas masqués. Mais on peut permettre cette discontinuité (autrement dit une césure) suivant la valeur de l'angle formé par les deux segments.

Pour contrôler cet effet un test est effectué lors du chaînage de deux segments consécutifs.

FIG. 2.27 – Tracé à l'encre de chine d'un aileron (angle aigu). a.) un seul trait b.) deux traits avec césure sur l'angle

D'autres tests au niveau de l'angle peuvent être envisagés : un test sur la courbure, permettrait de déceler les grandes variations et d'obtenir un contrôles du trait de silhouette au second ordre.

Un second critère de fin de trait, est la longueur. Un trait au pinceau n'est pas indéfiniment long.

Lors du chaînage d'un nouveau segment, on teste la longueur totale avec un seuil de longueur :

$$l = longueur(chaine\_construite) + longueur(nouveau\_segment)$$

Si la distance totale est inférieure au seuil : les deux tronçons peuvent être chaînés. Sinon, la longueur

FIG. 2.28 – Tracé généré par l'application d'un aileron (angle aigu). a.) un seul trait b.) deux traits avec césure sur l'angle

totale est trop importante : on arrête le trait ici.

Tous ces critères doivent être paramétrables et appartiennent au style.

2.2.2 Stratégies vers la cohérence temporelle

Avec les rendus de silhouette sans style, il n'existait pas de problèmes de cohérence et de continuité pour les animations. Les silhouettes et les plis étaient de simples liserets réguliers.
L'utilisation du style implique un début de trait et une fin de trait : de plus le trait peut ne plus être régulier le long de la silhouette suivant la texture utilisée. Tout ceci provoquent des sauts et des discontinuités si les traits et les irrégularités bougent beaucoup d'une frame à la suivante. Par exemple sur la figure 2.29, pour la troisième frame les traits ne sont plus construits et détectés dans le même ordre. Ceci est dû au fait que les arêtes sont orientés et que le chaînage ne se fait que dans un sens : si les segments sont traîtés successivement en tirant à chaque fois l'arête précédente (le A, puis le B,C,D...) on obtient un succession de petits traits. Le problème ici est d'avoir tiré le A avant l'arête qui commençait le trait (en haut, hors de l'image) à la frame précédente. Le long trait de la frame précédente est maintenant arrété au niveau de A, début d'une suite de petits traits.
Le début d'un trait peut également sauter d'une grande distance.

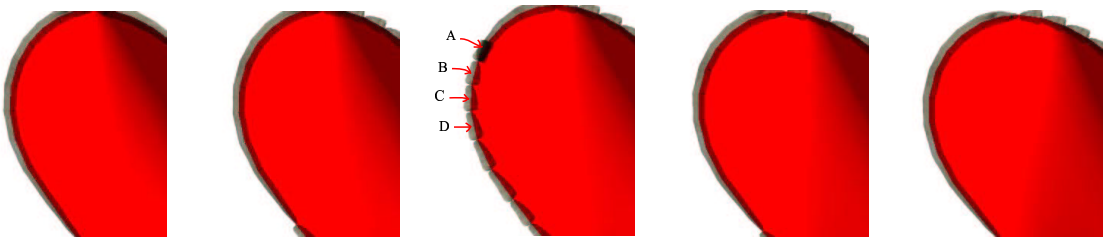


FIG. 2.29 – Cinq frames consécutives d'une animation : des discontinuités apparaissent sur la troisième frame

A cause de l'orientation du style des textures appliquées à la bande de polygones de silhouette, il devient nécessaire de traiter la cohérence temporelle. Cette cohérence passe par l'utilisation d'informations de la frame précédente.

Nous proposons ici un début de solution possible au problème : cinq stratégies ont été élaborées.

1. **Première stratégie :** commencer un trait dans le voisinage d'un trait de la frame précédente.
   Les irrégularités dépendent de l'ordre dans lequel on traite les arêtes de silhouette. Une première solution peut consister à essayer de traiter les traits toujours dans le même ordre d'une frame à l'autre. Pour cela, lors de la création des chaînes de polygones on mémorise le début de chaque chaîne. Une liste de vertex est construite avec un lien entre le vertex dans l'espace image et le vertex dans l'espace objet.
   Ensuite lors du calcul la frame suivante, on tire le premier vertex de la liste de vertex de début de chaîne, on recherche les arêtes de silhouette qui sont voisines de ce vertex et on construit la chaîne à partir de l'arête trouvée. Le début de l'arête de silhouette dans une frame est proche du début de trait dans la frame précédente (Fig 2.30).

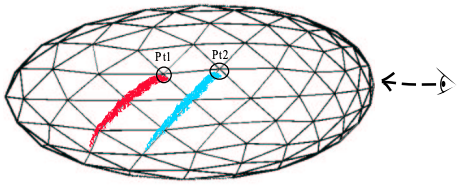


FIG. 2.30 – Maillage : le trait rouge est un trait de silhouette du point de vue à droite. Le trait bleu est le même trait à la frame suivante

   A partir d'un vertex donné, les arêtes adjacentes sont examinées (flèche rouge de la figure 2.31). Si aucune n'est une arête de silhouette encore inutilisée alors on recommence récursivement la recherche à partir des vertices nouvellement atteints (les vertices en vert sur la figure 2.31), en évitant de tester deux fois la même arête.

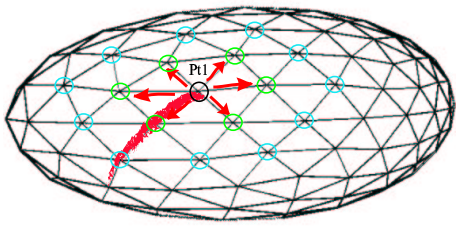


FIG. 2.31 – Exploration des traits

   Si une arête de silhouette est détectée, il s'agit de la plus proche en nombre d'arête de l'ancien début de trait. Le chaînage est effectué depuis cette arête.

   Il existe cependant des inconvénients à cette méthode. A partir du point le plus proche, l'arête à autant de chance d'être orientée dans la même direction que dans la direction opposée.

2. **Seconde stratégie :** privilégier une direction voisine à la direction du trait dans la frame précédente.
   Une solution au problème précédent peut consister à privilégier une direction lors du choix du trait. Un parcours semblable à celui de la première stratégie est effectué, à la différence que l'algorithme ne s'arrête pas sur la première arête trouvée. On fixe une profondeur à atteindre (trois par exemple), et on explore toutes les arêtes se trouvant à trois segments de distance de l'ancien début de trait. Pour chaque arête de silhouette trouvée, un calcul de poids est effectué de la manière suivante :

   $$poids = penalite1 * d + penalite2 * \theta$$

   Les notations utilisées ici correspondent à celles de la figure 2.32 : d est la distance entre l'ancien vertex de départ et le vertex de début de l'arête testée. La distance utilisée est la distance euclienne classique : ce calcul est effectué dans l'espace image. En effet le début du trait doit se trouver visuellement proche, sur le dessin, du trait prècèdent. θ est l'angle entre les vecteurs portant les deux arêtes. La startégie consiste ensuite à choisir comme arête de départ du trait, l'arête ayant le poids le plus faible à la fin de l'exploration.

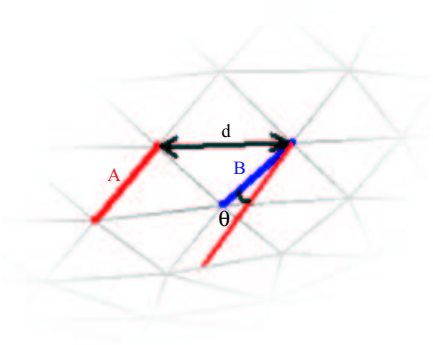


FIG. 2.32 – Choix de l'arête en fonction de la direction et de la distance : A est l'arête de départ dans la frame précédente, et B est l'arête testée

   Avec cette technique, les traits sont tracés avec la bonne direction au bon endroit. Il reste malgré tout des problèmes de longueur de trait : il est possible qu'un trait grandisse beaucoup trop d'une frame à la suivante.

3. **Troisième stratégie :** contrôler l'accroissement de la longueur d'un trait.
   Pour régler ce problème, lors de la construction des chaînes, la longueur total des tronçons est également stockée. Ensuite, pour le calcul des traits de la frame suivante, le critère de césure de chaîne limite l'acroissement du trait : on ne permet un accroissement que de 20 pixels par rapport à la longueur du trait à la frame précédente (cette valeur peut être paramétrable) : 20 pixels semblent convenir pour une résolution en pixel de $500 * 500$).
4. **Quatrième stratégie :** combler les espaces encore vide par les plus longs traits possibles.
   Utiliser uniquement les informations de la frame précédente pour déterminer les chaînes de trait provoque des trous dans la silhouette au bout de quelques frames. Ceci à cause de l'apparition possible de nouvelles arêtes de contour, ainsi que le contrôle de la longueur des traits de la

stratégie précédente.
Il faut donc combler ces trous. Une possibilité est d'utiliser dans un premier temps l'information de la frame d'avant pour calculer les traits (en appliquant les trois premières stratégies), puis de parcourir les arêtes de silhouette restantes et de construire des traits à partir de celles-ci. Le problème est que prendre ces arêtes au hasard peut provoquer des suites de petits traits (Fig 2.33) : en effet, la structure *winged edge* n'a pas d'ordre particulier.

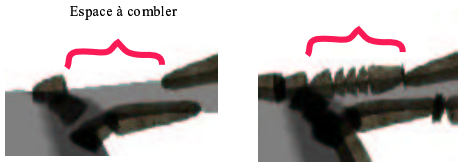


FIG. 2.33 – Partie comblée alétoirement

Il faut pouvoir optimiser le choix des arêtes à traiter en premier. Choisir des arêtes imédiatement voisines des fins ou des début des traits déjà tracés (Fig 2.34) permettrait d'éviter des collections de petits traits.

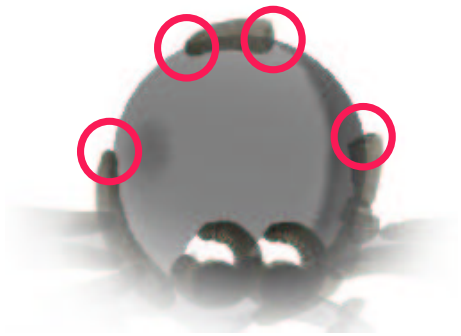


FIG. 2.34 – Partie comblée alétoirement

Ceci donne le résultat de la figure 2.35. Si la chaîne d'arête entre deux traits est favorable, l'espace est comblé par un seul trait.

FIG. 2.35 – Partie comblée

Le remplissage de ces espaces est naturellement une discontinuité, car on comble avec un trait qui n'existait pas avant. Il est cependant plus intéressant de combler avec des traits longs (respectant les critères de césure) que des traits courts.

Enfin certains espaces ne peuvent être comblés par cette méthode. Ces derniers peuvent être remplis par tirage aléatoire des arêtes. Si ces espaces ont pas été précédement traîtés nous

n’avons pas suffisament d’information pour élaborer une stratégie.

5. **Cinquième stratégie :** limiter la multiplication des petits traits parasites.
   Dans certain cas, il faut combler ou tracer de petits traits.Or tout les traits tracés lors d’une frame servent de base aux traits de la frame suivante, les petits traits placés se multiplient rapidement (Fig 2.36). Ceci est dû aux espaces qu’il est nécessaire de combler (parfois par des traits courts), et que plus un trait est court plus il est stable. Le trait court correspond en général à une ou quelques arêtes, et est plus proche topologiquement de l’arête.

FIG. 2.36 – Cinq rotations successives de 180 degrés d’un modèle de coeur : des petits traits apparaissent à partir de la seconde image. Sur la derniére image, le filtrage des traits courts est activé.

Il peut être utile de ne se servir comme trait de base que de traits possédant une longueur minimum : le seuil est fixé par l’utilisateur et doit être un compromis entre un seuil trop bas qui provoquera la multiplication de petits traits (donc stables), et un seuil trop élevé qui provoquera trop de recalcul sans connaissance de la frame précédente (donc trop de discontinuité temporelle).

Chapitre 3

Résultats

3.1 Résultats

3.1.1 Images fixes

Le rendu des images fixes est comparable au résultat de Northrup *et al.* dans [NM00] au niveau des silhouettes et à Lake *et al.* dans [LMHB00] pour l'ombrage de type cartoon.
Voici quelques résultats, d'autres se trouvent en annexe.

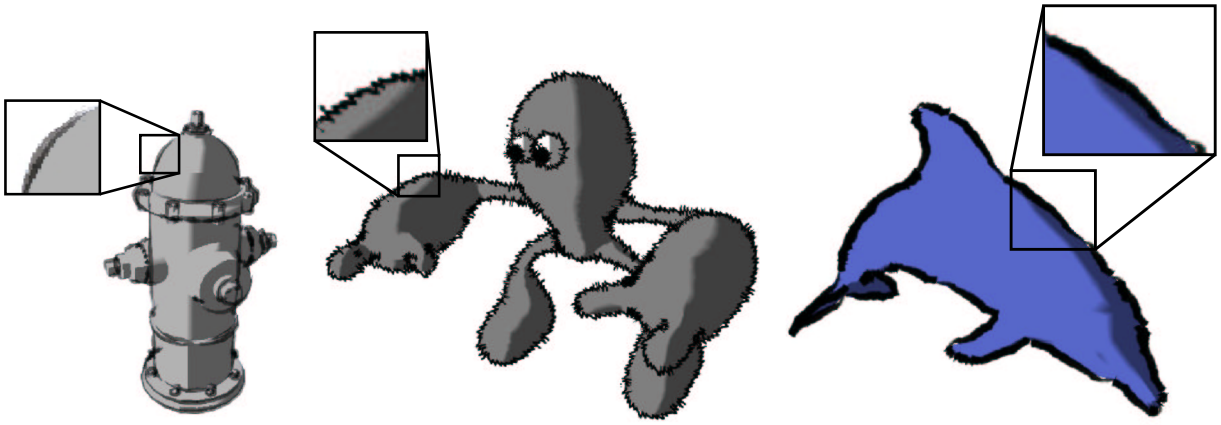


FIG. 3.1 – Quelques images avec différentes textures pour les traits.

3.1.2 L'animation

Les cinq statrégies proposées permettent un meilleure cohérence.
Des tests d'animations ont été réalisés sur plusieurs modèles. Un de ces modéles était un cœur (figure 3.2) : l'avantage de ce modèle est qu'il est à la fois simple à observer et qu'il comporte plusieurs configurations, comme des discontinuités et des longues courbes.
Voici les résultats obtenus pour une rotation autour d'un axe vertical. Dans la première série d'images (figure 3.3), on constate que les traits restent cohérents dans la majorité des cas. Entre la seconde et la troisième frame, trois traits ont disparus remplacés par un plus long. Le système a cependant conservé les autres traits avec leurs positions et leurs longueurs. La cinquième image voit l'apparition d'un trait

parasite, qui arrivera à rester malgré le filtre sur les quatres images suivantes.

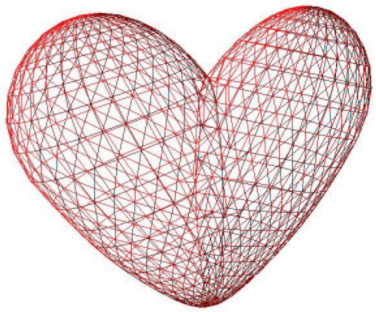


FIG. 3.2 – Coeur en *wire-frame*

On constate que sur l'animation sans le contrôle de la cohérence, les problèmes d'apparition de traits sont plus aléatoires et plus fréquents. Ce phénomène est toujours lié à l'ordre et l'orientation des arêtes de silhouettes traîtées : tout ceci est corrélé à l'ordre de la structure *winged edge*. C'est le cas entre l'image 1 et 2, puis 2 et 3 et enfin 8 et 9 ou de nombreux traits apparaissent où disparaissent sur une grande partie de la silhouette.

3.1.3 Problèmes de silhouette

Des traits parasites apparaissent sur la silhouette le long de trait (Fig 3.5) : ces traits se superposent sur les traits déjà existant (créant ainsi une tâche sombre). La cause de ces traits est la forme des chaînes d'arêtes de silhouette. Les arêtes de silhouette sur un maillage ne forment pas une ligne régulière et lisse le long de la surface. En réalité, il s'agit d'une ligne brisée (voir Fig 3.6). Afin d'obtenir une ligne régulière il est nécessaire de nettoyer les arêtes trouvées. La cohérence temporelle est également affectée par ces traits parasites : on peut donc penser que le filtrage permettra aussi d'améliorer la cohérence.

3.1.4 Performances de l'algorithme

L'algorithme est divisé en trois parties distinctes :

- La première partie est la détection de silhouette : chaque arête du modèle est examinée en temps constant. La complexité est en $O(n)$ avec n le nombre d'arêtes.
- La seconde partie est la gestion de la visibilté des silhouettes : le vecteur d'arête est parcouru puis un nouveau vecteur est crée avec des insertion en tête. La complexité est donc en $O(k^2)$ avec k le nombre d'arêtes de silhouette ou de pli.
- La dernière partie consiste à rechercher les débuts de trait : on cherche autant de début de traits que l'on avait de traits à la frame précédente. Ensuite les arêtes de silhouettes sont parcourus deux fois : une fois lors du chaînage et une fois lors de l'affichage. La compléxité est donc en $O(k)$.

La compléxité globale de l'algorithme est donc en $O(k^2+n)$.

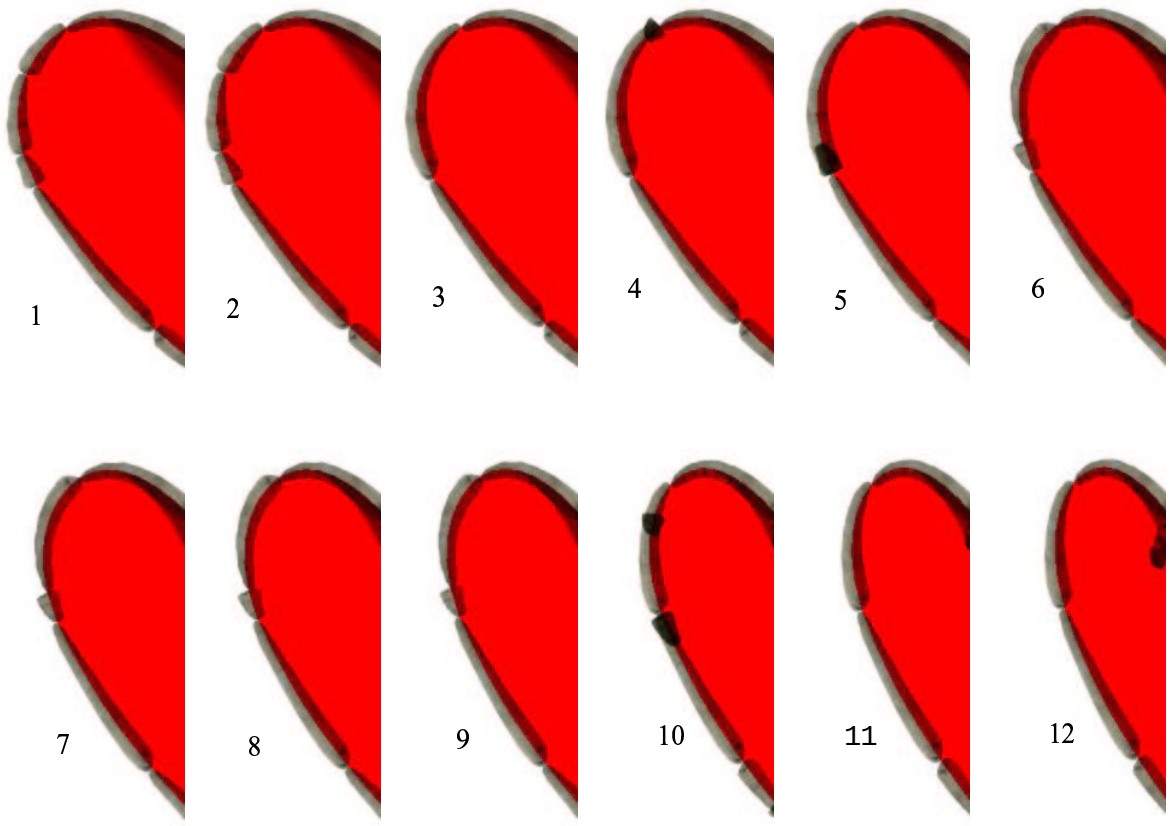


FIG. 3.3 – Animation avec les stratégies de cohérence

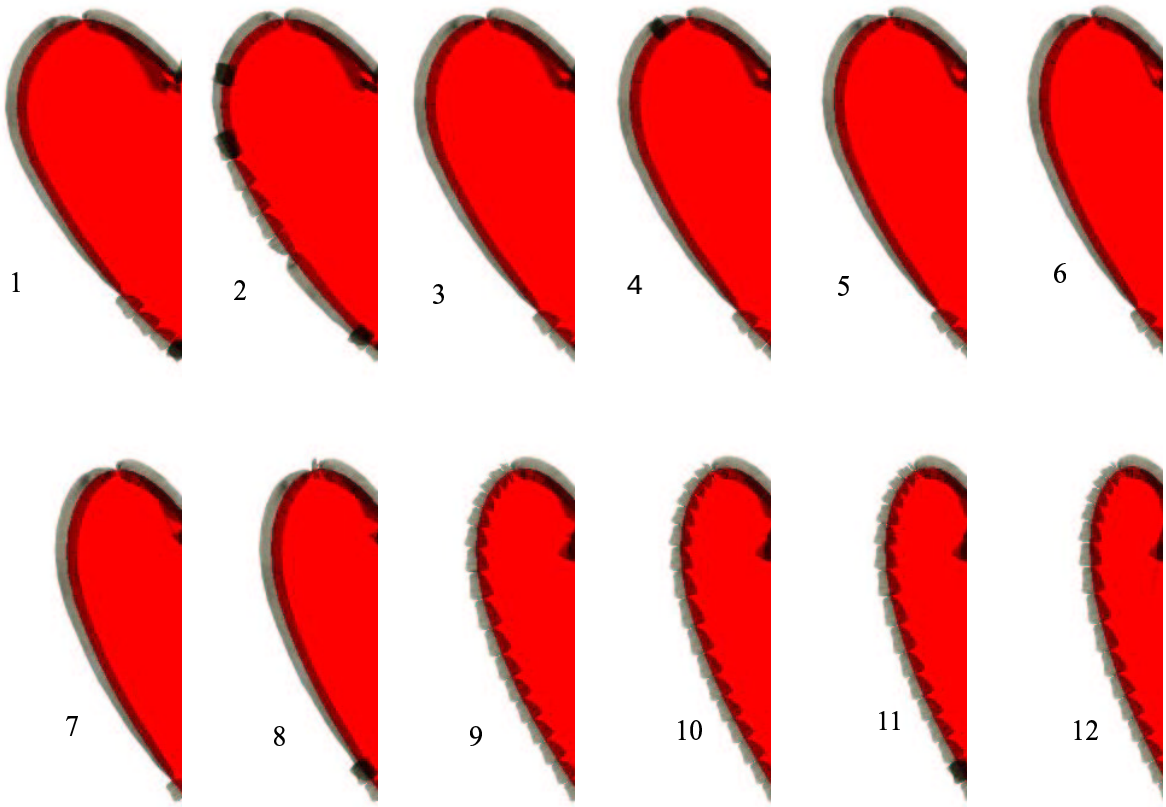


FIG. 3.4 – Animation sans les stratégies de cohérence

FIG. 3.5 – Un trait parasite

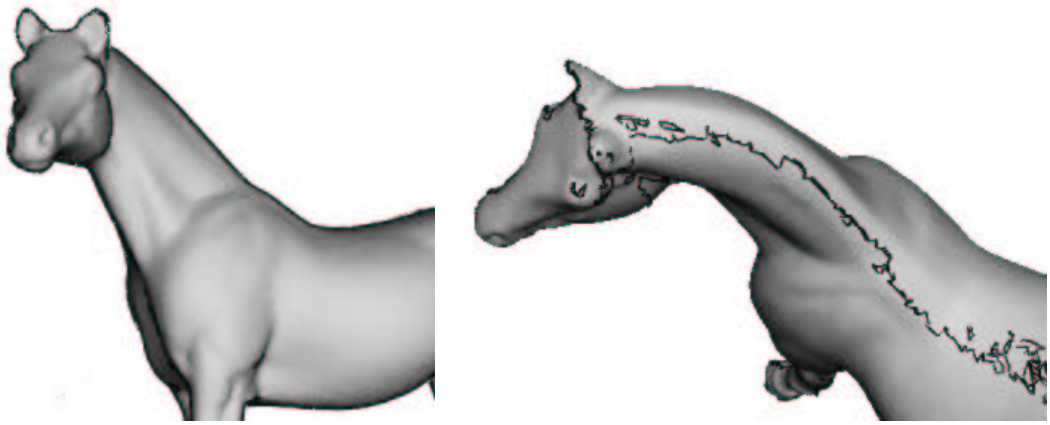


FIG. 3.6 – Forme de la silhouette sur un modèle

L'application a été testée sur un Athlon AMD à 1,7 GHz.
Les modèles testés sont ceux de la figure 3.7. Pour les résultats des tests (tableau 3.1), les tests sont effectués au début sans l'affichage des contours, uniquement avec l'ombrage. Les deux derniers tests sont effectués avec l'affichage des silhouettes et des plis.
La détection des silhouettes est un problème lourd en calcul et n'a pas été optimisée : il est donc normal que le nombre d'images par seconde baisse avec cette détection. Le taux baisse également avec le traitement de la visibilité, ceci à cause du parcours du vecteur des arêtes de contour. Chaque arête du vecteur est testée : le vecteur est ensuite reconstruit de manière à préserver l'ordre à l'aide d'insertion en tête. Le coût de traitement d'une telle opération étant important, l'utilisation d'une liste pour la gestion des arêtes visibles semble plus appropriée.
L'étape d'affichage consiste à parcourir, chaîner puis afficher à l'écran les silhouettes et les plis visibles : le coût de l'algorithme utilisé ici est linéaire : ceci explique la faible différence d'images par seconde. Enfin le calcul de la cohérence temporelle implique des parcours récursifs du maillage afin de détecter la meilleure arête : il s'agit donc d'une opération coûteuse.

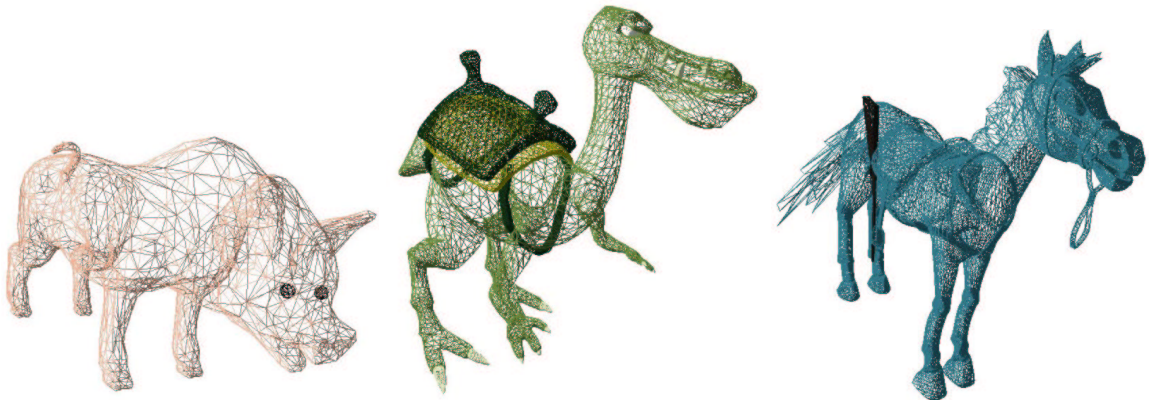


FIG. 3.7 – Les trois modèles utilisés en *wire-frame*.

	ombrage	ombrage + contour (sans affichage)	ombrage + contour + visibilité (sans affichage)	ombrage + contour + visibilité (avec affichage)	ombrage + contour + visibilité + cohérence (avec affichage)
cochon (2888 faces)	200	125	67	61	33
dinosaure (15945 faces)	72	26	17	15	11
poney (31432 faces)	39	14	9	7.7	5.2

TAB. 3.1 – Performances de l'application suivant les modèles testés (en images par seconde)

	cochon	dinosaure	poney
nombre de face	2888	15945	31432
nombre maximum de coup de pinceau (environ)	500	1500	3000
nombre minimum de coup de pinceau (environ)	50	150	400

TAB. 3.2 – Nombres de coup de pinceau, suivant les modèles.

Lors de l'analyse de la complexité et des performances de l'algorithme, il est également important de prendre en compte le nombre de traits stylisés tracés à l'écran (voir tableau 3.2). Les bornes ont été déterminées en essayant de trouver l'angle de vue qui minimisait ou maximisait le nombre de traits et en jouant sur les paramètres de césure. La borne inférieure dépend naturellement des ces paramètres. Pour la borne supérieure, la longueur maximum d'un trait était fixée à zéro de manière à ce que chaque tronçon ne soit pas chaîné à un autre. Cette borne correspond donc au nombre d'arête de silhouette et de pli du modèle.

3.2 Discussions et travaux futurs

3.2.1 Discussions

L'animation de silhouette avec des traits stylisés pose des problèmes importants de comportement du trait. Lors d'une translation, le trait ne doit pas bouger et doit toujours être tracé de la même manière. Le changement d'échelle conserve les arêtes de silhouette, mais les traits s'allongent ou se rétrécissent : le trait glisse le long de la silhouette et tend à s'allonger au maximum (voir figure 3.8).

FIG. 3.8 – Cinq zooms arrières consécutifs.

Lors d'une rotation ou d'une déformation du modèle, des traits disparaissent et d'autres apparaissent : les traits peuvent également glisser ou changer de longueur.

Ces phénomènes ne sont pas décris, et n'existent pas ailleurs qu'en animation NPR. La question de la crédibilité de ce genre de comportement est donc subjective, et d'une manière générale l'animation de modèle 3D en NPR pose souvent le problème de comparaison et de validation. Il n'existe pour l'instant pas de règles établies.
Cependant, du point de vue de la continuité, ces comportements sont cohérents, donc acceptables.

3.2.2 Travaux futurs

Il serait intéressant de mieux filtrer les arêtes de silhouette : dans un premier temps pour gérer tous les cas de visibité, et également pour éviter les traits parasites qui perturbent la cohérence temporelle : en effet ce processus est sensible à ce genre de trait (qui ont une direction opposée à la direction générale du trait).

Une amélioration de l'image obtenue serait possible si certaines informations de césure se trouvaient directement sur le modèle 3D : construire un modèle avec des informations de fin de trait et d'angle à préserver placées par l'animateur pourait être intéressant.

Lorsque l'objet est observé de loin, il devient inutile de tracer tous les traits, ce qui rend le dessin plus difficile à comprendre : certains traits doivent être ignorés : il pourrait être intéressant de ne pas tracer les traits trop cours, et de ne garder que les plus longs (qui sont souvent les plus caractéristiques).

Chapitre 4

Conclusion

Le but de la présente étude était l'animation de modèle 3D avec un rendu cohérent de type cartoon.

Il s'agissait dans un premier temps d'obtenir un rendu se rapprochant de la bande dessinée. Des travaux antérieurs ont été utilisés et implémentés. Le *cartoon shader* basé sur les aplats permit d'obtenir des ombrages et des couleurs similaires à la bande dessinée. Les silhouettes de contour (appellées aussi traits cernés) ont été renforcées par le rendu à base de texture sur bandes de polygones. Des critères de césures ont été ici explicités.

La seconde étape était le rendu cohérent lors de mouvement et d'animation des modèles. Il n'éxiste pas encore de travaux aboutis sur ce sujet.

C'est dans ce sens, que nous proposons des stratégies vers la cohérence, qui se résument en cinq points :

- Placer le départ d'un trait dans le voisinage du départ de ce trait dans la frame précédente.
- Privilégier une direction de départ voisine à celle du trait de la frame précédente.
- Contrôler l'accroissement de ce trait.
- Combler les espaces par des traits les plus longs possibles.
- Ne pas tenir compte des traits trop courts de la frame précédente.

Ces cinq points ont permi de rendre plus stable les traits de silhouette, malgrés, les traits parasites.
Il reste cependant beaucoup de travail à faire, car il s'agit d'un problème difficile, motivé par de réelles applications.
Il serait enfin souhaitable de réutiliser des algorithmes de détection de silhouette et de visibilité plus performant afin d'aller vers le temps réel pour des modèles plus volumineux.

Bibliographie

[Bau72] W. Baumgart. Winged edge polyhedron representation. Technical report, Stanford AI Laboratory, 1972.

[Dec96] Philippe Decaudin. Rendu de scène 3d imitant le style "dessin animé". Technical report, INRIA, 1996.

[GGSC98] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A non-photorealistic lighting model for automatic technical illustration. *Proceedings of SIGGRAPH 98*, pages 447–452, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[GSG+99] Bruce Gooch, Peter-Pike Sloan, Amy Gooch, Peter Shirley, and Richard Riesenfeld. Interactive Technical Illustration. *Interactive 3D Conference Proceedings*, April 1999.

[KMM+02] Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, and Adam Finkelstein. Wysiwyg npr : Drawing strokes directly on 3d models. In *SIGGRAPH 2002*, 2002.

[KS91] P.J. Kellman and T.F. Shipley. A theory of visual interpolation in object perception. *Cognitive Psychology*, 23(2) :141–221, 1991.

[LMHB00] Adam Lake, Carl Marshall, Mark Harris, and Marc Blackstein. Stylized rendering techniques for scalable real-time 3d animation. In *Non-Photorealistic Animation and Rendering 2000 (NPAR '00)*, Annecy, France, June 5-7,2000.

[NM00] J.D. Northrup and Lee Markosian. Artistic silhouettes : A hybrid approach. In *Non-Photorealistic Animation and Rendering 2000 (NPAR '00)*, Annecy, France, June 5-7,2000.

[PBD+01] M. Pop, G. Barequet, C.A. Duncan, M.T. Goodrich, W. Huang, and S. Kumar. Efficient perspective-accurate silhouette computation. In *Proc. 17th Ann. ACM Symmp. on Computational Geometry (SoCG)*, Annual Conference Series, pages 60–68, 2001.

[ST90] T. Saito and T. Takahashi. Comprehensible rendering of 3-d shapes. *Computer Graphics*, 24(4) :197–206, august 1990.

[WS94] Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. In *Computer Graphics proceedings, SIGGRAPH'94*, Annual Conference Series, 1994.

[WS96] Georges Winkenbach and David H. Salesin. Rendering Parametric Surfaces in Pen and Ink. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 469–476. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[ZI97] H. Zhang and K. Hoff III. Fast backface culling using normal masks. In *Proc. 1997 Symposium on Interactive 3D Graphics*, pages 103–106, April 1997.

Annexe

Images

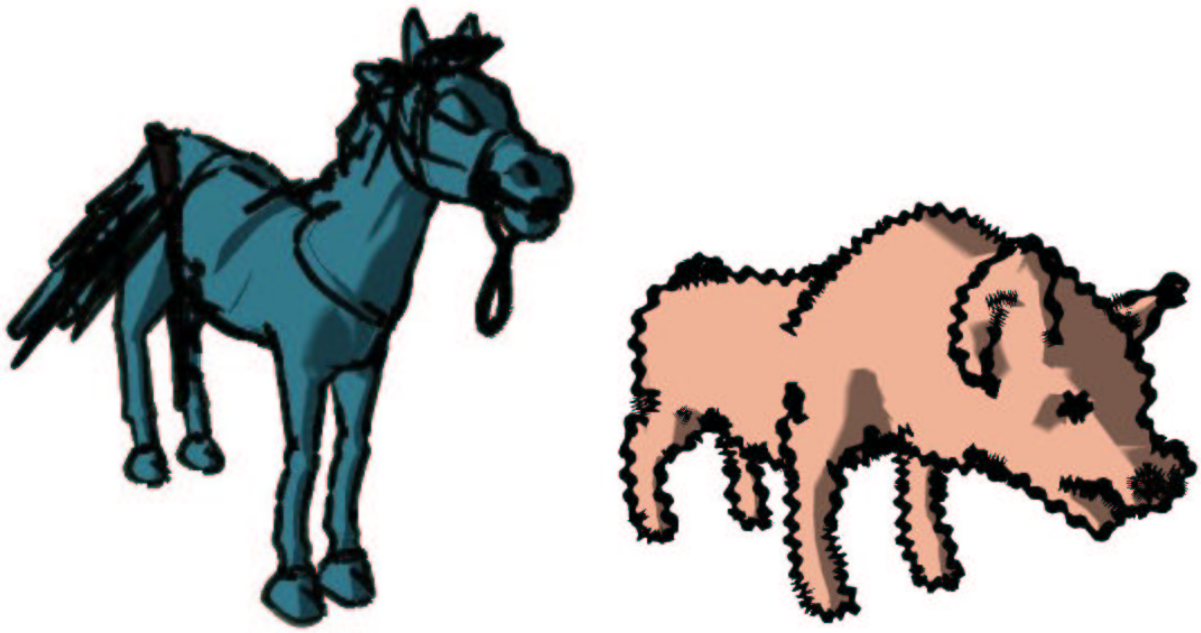


FIG. 4.1 – Un poney (traits à l'aquarelle noire) et un cochon (traits de vague créé avec Gimp).

FIG. 4.2 – Dinosaure avec différents styles de trait (traits à l'aquarelle noire).

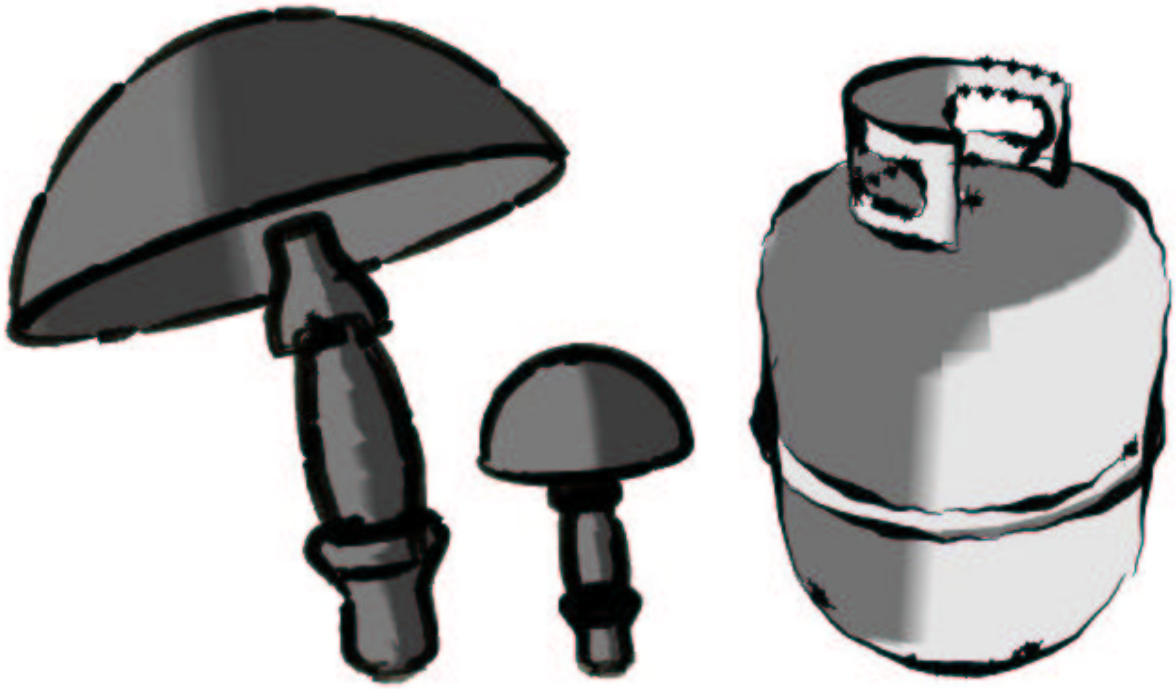


FIG. 4.3 – Des champigons (traits à l'aquarelle noire) et une bouteille de propane (traits à l'encre de chine).

FIG. 4.4 – Poisson avec différent style de trait (logo NPR et crayon papier).

Coordonnées de texture

Pour appliquer la texture de trait sur la bande de polygône, il faut spécifier les coordonnées de texture. Lors de la construction de chaîne la longueur de chaque tronçon a été calculée. est possible.

Avec *total* qui est la longueur totale du trait, *longueur_cour* qui est la longueur à partir du sommet du tronçon jusqu'à la fin du trait et *long_tronon* la longueur du tronçon courant, on cherche les coordonnées (u,v) dans l'espace de texture pour chaque sommet des triangles d'un tronçon :

$$a = ((total - longueur\_cour)/total\ , 0.0) \tag{4.1}$$
$$b = ((total - longueur\_cour)/total\ , 0.5) \tag{4.2}$$
$$c = ((total - longueur\_cour)/total\ , 1.0) \tag{4.3}$$
$$d = ((total - longueur\_cour\ + long\_tronon)/total, 0.0) \tag{4.4}$$
$$e = ((total - longueur\_cour\ + long\_tronon)/total, 0.5) \tag{4.5}$$
$$f = ((total - longueur\_cour\ + long\_tronon)/total, 1.0) \tag{4.6}$$

Avec les valeurs correspondantes sur le tronçon (Fig 4.5).

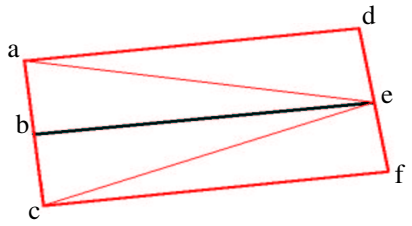


FIG. 4.5 – Un tronçon de la bande de polygone avec les coordonnées aux sommets

La structure de classe *winged edge*

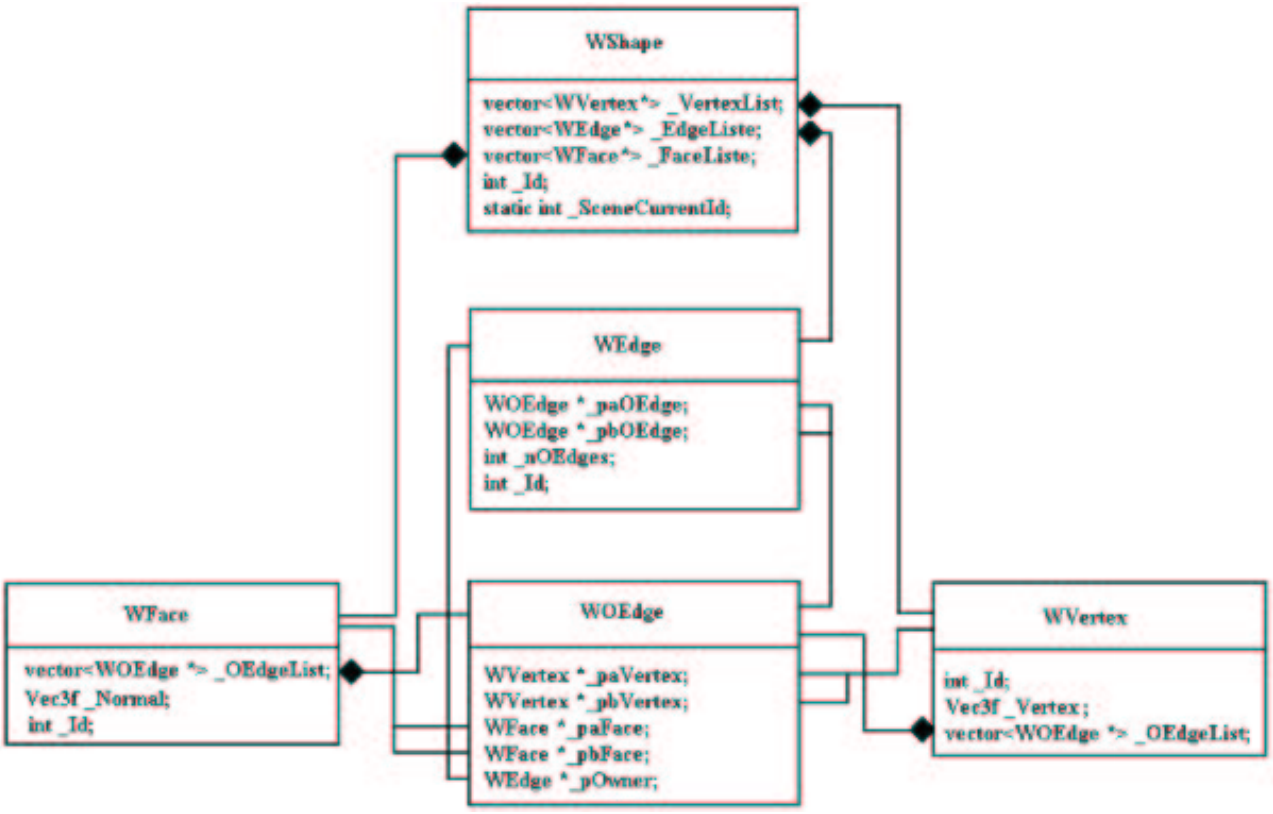


FIG. 4.6 – Structure de classe

La construction de la bande de polygone est effectuée par l'algorithme suivant (à noter que la construction commence par une arête) :

Construction de la chaîne de polygone

Voici l'algorithme de construction de la bande de polygones.

```
structure arete {
  booleen utilise;
  entier longueur;
  suivant arete;
}

Construction_chaine(arete a)
{
  Si (non a.utilise) alors
    a1.utilise = vrai ;
    Pour toute a1 appartenant aux arêtes incidentes de a
      Si critere de trait(a,a1) alors
         Trouve = vrai
         Construction_chaine(arete a)
         a.longueur = a1.longueur + module(a)
         a.suivant = a1
      FinSi
    FinPour
    Si (non trouve) alors
      a.longueur = module(a)
      a.suivant = null
    Finsi
  Finsi
}
```