Dégradation de scènes 3D

Marc GLISSE

14 septembre 2001

Table des matières

1 Introduction

Motivations Un des éléments les plus recherchés en infographie est le réalisme des scènes. Dans de nombreux domaines comme les jeux, la réalité virtuelle, la simulation, etc, il serait souhaitable d'avoir des images aussi proches que possible de la réalité. Ce travail demande beaucoup de temps et d'habileté. Le réalisme passe par les effets d'usure, de salissement qui touchent les objets les plus variés. La modélisation de ce vieillisement des surfaces augmente le réalisme des scènes, mais aussi la quantité de travail laissée à l'artiste chargé de créer la scène. Pour lui faciliter le travail, on essaie de développer des méthodes automatiques.

Cadre du stage Ces travaux ont été réalisés au cours de mon stage de maîtrise à l'INRIA dans le projet REVES sous la direction conjointe de George Drettakis et de Pierre Poulin (Université de Montréal). Ils s'appuient sur le rapport prédoctoral d'Éric Paquette ([Paq99]).

Méthodes étudiées Je me suis intéressé à quelques dégradations très particulières. J'ai commencé par étudier les éraflures que l'on remarque sur les carrosseries des voitures, ou sur n'importe quelle surface polie. Je me suis ensuite intéressé aux traces laissées par les gouttes qui coulent le long d'une vitre ou d'un mur. Pour terminer j'ai étudié un model de dépôt de la poussière sur certaines surfaces.

2 État de l'art

On commence par une petite introduction au sujet. On part d'un modèle de scène 3D sous forme par exemple d'un ensemble de polygones. On projette cette scène sur un plan pour avoir une image 2D. Pour chaque pixel de l'image, on cherche à déterminer une couleur. Pour cela on attribue à chaque surface de la scène un "shader". La scène est globalement éclairée par un certain nombre de sources, directes ou non. Le shader est ce qui détermine localement l'interaction entre la lumière et la surface. Cette interaction est représentée par une BRDF (bidirectional reflectance distribution function). Il s'agit d'une fonction qui en chaque point pour chaque longueur d'onde associe aux angles d'incidence et de réflexion la proportion de lumière réfléchie. Il s'agit d'une fonction de 7 variables, et on représente généralement la BRDF de façon simplifiée. On élimine souvent la dépendance en la longueur d'onde. Le modèle le plus basique, dû à Phong ([Pho75]), n'utilise que le cosinus de l'angle entre les deux directions.

Un certain nombre de recherches ont déjà été faites sur les sujets que j'ai étudiés. En voici un aperçu.

2.1 Éraflures

Les éraflures (ou scratchs) sont une des irrégularités les plus visibles, il est donc nécessaire de les introduire dans les scènes. La technique la plus simple et la plus intuitive consiste à modifer la géométrie de la surface pour lui donner celle de la vraie vrai éraflure. Cependant cette technique est très coûteuse. Quand on voit une éraflure, au moins de loin, on ne remarque généralement pas la géométrie mais uniquement les effets lumineux. Il s'agit donc de représenter les interactions locales de la lumière avec le matériau. Plusieurs models de BRDF ont déjà été proposés. Le plus connu est celui de Phong ([Pho75]) qui utilise un paramètre de rugosité pour contrôler la partie spéculaire de la réfractance. Poulin et Fournier ([FP90]) franchissent une nouvelle étape en utilisant de fins cylindres attachés en parallèle pour décrire un modèle anisotrope. Une caractéristique commune à ces modèles est la continuité de leur résultat sur la surface des objets. Cela ne correspond pas à la réalité, où l'on remarque souvent des apparences non-uniformes. Pour rendre de tels phénomènes, on a recours à des techniques de plaquage de textures 2D. Blinn ([Bli78]) a introduit le "bump mapping". Il s'agit d'une carte d'altitudes, utilisée pour simuler une déformation de la surface. On modifie ainsi la normale à la surface, on peut même gérer l'auto-ombrage. Cependant dès que les bumps sont importants, cette méthode est mauvaise. En effet elle ne permet pas de prendre en compte l'auto-masquage, et ne donne pas des contours réalistes. Le "displacement mapping" consiste lui à déformer réellement les objets. Tous les phénomènes sont donc bien pris en compte, mais malgré les perfectionnements récents cette technique reste coûteuse. On va distinguer deux types d'éraflures. Les micro-éraflures, qui sont loin en-dessous de l'échelle du pixel, ne sont en général pas perceptibles individuellement, mais donnent un aspect homogène au matériau dont la microtexture peut être directement prise en compte par la BRDF. Au contraire, les éraflures en V, qui proviennent par exemple d'une action mécanique extérieure, sont à l'échelle du pixel et induisent une discontinuité dans les rapports entre la lumière et le matériau avec une certaine dépendance en les directions de la lumière incidente et du point de vue. Notamment pour des raisons d'aliassage, les techniques de bump mapping ou de displacement mapping sont inefficaces sur ce problème. À cette échelle, une approche fondée sur des BRDFs variables semble plus adaptée. Cependant, bien que cela donne des résultats rapides et justes, enregistrer une table complète de BRDFs (c'est à dire une fonction : $\mathbb{R}^6 \to \mathbb{R}$) est bien trop coûteux pour le cas particulier des éraflures qui sont localisées. La méthode retenue consiste alors à decrire la position géométrique des éraflures (par exemple sur une texture noire et blanche), puis à faire appel selon les cas à la BRDF usuelle ou à celle spécifique aux éraflures, orientée selon la direction de l'éraflure. Il reste maintenant à déterminer la BRDF d'une éraflure en s'appuyant sur un modèle physique. Le model présenté ici est celui de [MDG01]. La forme des éraflures est analysée à l'aide d'appareils de mesure très précis, et il apparaît qu'on peut modéliser, au moins d'une façon approximative, les éraflures comme dans le schéma 1. Le profil est régulier et symétrique. Il se divise en six zones distinctes :

- 4 zones centrales correspondant à l'éraflure elle-même,
- 2 zones non éraflées de part et d'autre.

On suppose que les 4 zones centrales ont la même largeur pour simplifier les calculs (cela est relativement cohérent avec la forme des éraflures analysées). Comme on ne s'intéresse ici qu'à des modifications invisibles de la géométrie, on peut supposer que la largeur d'une éraflure ne dépassera jamais un demi-pixel. Bien entendu, la proportion du pixel éraflée est inversement proportionnelle à la distance de l'oeil. On introduit alors un paramètre p_s qui varie entre 0 et $0,5$ et qui représente cette proportion. On note l_k ($k = 1, \dots, 6$) les proportions de chacune des 6 zones.

$$\sum_{k=1}^{6} l_k = 1,$$
$$l_1 = l_6 = \frac{1 - p_s}{2} = l_p,$$
$$l_2 = l_3 = l_4 = l_5 = \frac{p_s}{4} = l_s.$$

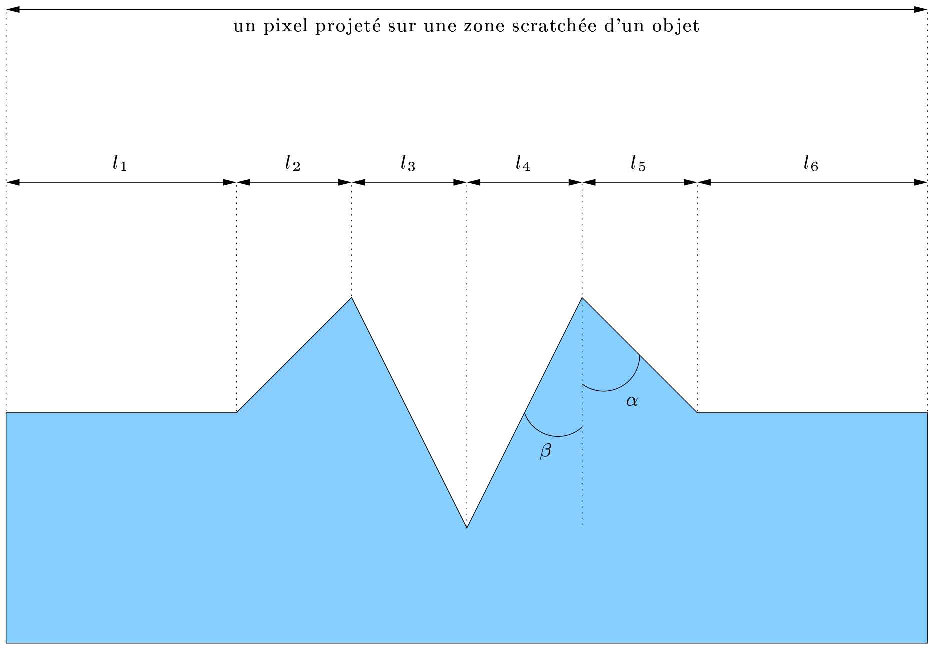


FIG. 1: Coupe de la géométrie d'une éraflure.

Pour décrire la géométrie de l'éraflure, on a aussi besoin des paramètres suivants. On note α l'angle externe et β l'angle interne (figure 1). Pour simplifier les formules, on définit α_k par :

$$\forall k \in \{1, 2, 5, 6\}, \alpha_k = \alpha \quad \text{et} \quad \forall k \in \{3, 4\}, \alpha_k = \beta$$

Ces angles sont déterminés par mesure directe sur les profils observés. Une remarque que l'on peut faire est que physiquement on aimerait conserver la quantité de matière lors de la formation de l'éraflure, ou au moins éviter une création de matière. Ceci se traduit par la condition très simple suivante : $\tan(\alpha_s) \geq 3 \tan(\beta_s)$. Pour avoir une idée, quelques exemples de valeurs pour α et β sont donnés dans le tableau 2.1. Ces valeurs peuvent varier selon

Type de surface	α (degrés)	β (degrés)
alliage métallique 1	70	40
alliage métallique 2	80	35
plastique 1	50	20
plastique 2	75	48
surface peinte	75	35

TAB. 1: mesures des angles des éraflures

les outils utilisés pour faire les éraflures. La texture 2D est utilisée pour modifier non la couleur du materiau mais sa BRDF. Plusieurs couleurs peuvent être utilisées pour rendre différents types d'éraflures et rendre mieux compte de la réalité. On calcule en fait deux BRDFs : $f_{r,scratch}$ pour l'éraflure elle-même et f_r pour les zones non-éraflées. La texture nous permet alors de choisir entre les deux. $f_{r,scratch}$ est calculée à partir des BRDFs des différentes zones : $f_{r,k}$ et de leurs normales respectives : N_k. On utilisera aussi $\omega_i(\theta_i, \phi_i)$ et $\omega_r(\theta_r, \phi_r)$ pour désigner les directions de la source de lumière et de l'oeil. La figure 2 précise les notations. Avec des considérations géométriques simples, on trouve :

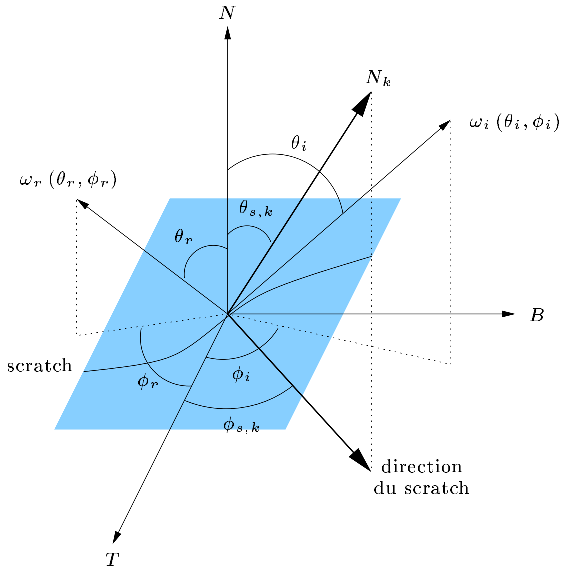


FIG. 2: Géométrie locale d'une éraflure.

$$f_{r,scratch}(\omega_i, \omega_r) = \sum_{k=1}^{k=6} f_{r,k}(\omega_i, \omega_r) \frac{(N_k|\omega_i)(N_k|\omega_r)}{\sin(\alpha_k)\cos(\theta_i)\cos(\theta_r)} l_k G_k \tag{1}$$

où les G_k sont donnés par :

$$G_1 = \max\left(0, \min\left(1, \frac{(l_p + l_s)\tan(\alpha_2) - l_s\tan(\theta_i)\cos(\phi_i - \phi_{s,2})}{l_p\tan(\alpha_2)}, \frac{(l_p + l_s)\tan(\alpha_2) - l_s\tan(\theta_r)\cos(\phi_r - \phi_{s,2})}{l_p\tan(\alpha_2)}\right)\right)$$

$$G_6 = \max\left(0, \min\left(1, \frac{(l_p + l_s)\tan(\alpha_5) - l_s\tan(\theta_i)\cos(\phi_i - \phi_{s,5})}{l_p\tan(\alpha_5)}, \frac{(l_p + l_s)\tan(\alpha_5) - l_s\tan(\theta_r)\cos(\phi_r - \phi_{s,5})}{l_p\tan(\alpha_5)}\right)\right)$$

$$\forall k \in \{2, 5\}, \qquad G_k = \max\left(0, \min\left(1, \frac{1}{cos(\phi_i - \phi_{s,k})}, \frac{1}{cos(\phi_r - \phi_{s,k})}\right)\right)$$

$$\forall k \in \{3, 4\}, \qquad G_k = \max\left(0, \min\left(1, \frac{2\sin(\alpha_k)\cos(\theta_i)}{\cos(\theta_i)\sin(\alpha_k) + \cos(\alpha_k)\sin(\theta_i)\cos(\phi_i - \phi_{s,k})}, \frac{2\sin(\alpha_k)\cos(\theta_r)}{\cos(\theta_r)\sin(\alpha_k) + \cos(\alpha_k)\sin(\theta_r)\cos(\phi_r - \phi_{s,k})}\right)\right).$$

Dans ces formules, les $f_{r,k}$ dépendent des N_k. Les coefficients de la formule (1) viennent de l'inclinaison des surfaces et les G_k de ce que certaines zones peuvent être partiellement voire totalement masquées par d'autres. Un calcul permet de vérifier la cohérence physique de cette formule :

- $f_{r,scratch} \geq 0$,
- $f_{r,scratch}$ est symétrique en ω_i et ω_r,
- il n'y a pas création d'énergie.

2.2 Poussière

Un phénomène encore plus courant est le dépôt de poussière. On trouve de la poussière partout, sur le moindre objet. Des particules d'objets, d'animaux, de plantes, se détachent, se déplacent au gré des courants d'air, puis se déposent jusqu'à ce qu'un frottement ou un autre courant d'air les détache. Blinn ([Bli82]), cherchant à représenter les anneaux de Saturne, s'est intéressé à la représentation de la poussière. En effet les anneaux son composés de fines particules de glace dont les propriétés de réflexion sont proches de celles de la poussière. Le modèle permet le rendu d'une couche homogène semi-transparente composée de molécules de faible albédo. La principale difficulté est alors de savoir où doit se trouver la poussière et en quelle quantité. Cette tâche revient naturellement à l'artiste chargé de réaliser la scène. Cependant il est possible de l'aider. Hsu et Wong ([HW95]) donnent un modèle qui calcule l'épaisseur de la couche de poussière qui devrait être présente sur une surface selon la position de sources de poussière. Le modèle est très proche de celui de Phong pour la lumière ([Pho75]) : l'accumulation de poussière dépend de l'angle entre la normale à la surface et la direction de la source. Pour simuler l'accumulation plus lente de la poussière dans les endroits difficiles d'accès, un rayon est lancé aléatoirement au-dessus de la surface. Inversement, la poussière dans les endroits reclus sera difficile à enlever. L'accessibilité, une fois calculée, pourra donc être réutilisée pour réduire la quantité de poussière à la fois des objets difficiles d'accès et de ceux où la poussière risque d'être enlevée par frottement. Si l'influence des sources de poussière est rapide à calculer, l'accessibilité est beaucoup plus coûteuse. Il est en effet nécessaire d'envoyer un grand nombre de rayons pour réduire le bruit présent dans l'image. Il y a aussi une composante de bruit qui dépend de la résolution de l'image et qui sera inacceptable dans une animation. Les auteurs suggèrent un stockage des valeurs d'accessibilité dans des textures pour régler ces problèmes. Dans tous les cas, une simulation du déplacement de la poussière donnerait, à un coût plus élevé, un résultat plus réaliste.

3 Mes travaux

3.1 Éraflures

J'ai adopté pour modéliser les éraflures la technique décrite au paragraphe 2.1. L'ensemble des éraflures est défini par une liste de vecteurs de points 2D. Une éraflure est donc une ligne brisée. On définit ainsi une texture qui sera ensuite plaquée sur l'objet par maya. J'ai fait le choix de mettre un Phong-shader comme BRDF des zones non éraflées et des zones 1, 2, 5 et 6 des éraflures et un autre pour les deux denières zones. Mais ceci n'est qu'un choix pour les exemples et peut tout à fait être modifié. Ce choix est motivé par le fait que sur les bords, l'éraflure est comme le reste de la surface du matériau, alors que dans l'éraflure, il se peut que l'on voit une couche inférieure, comme le métal sous la peinture. On applique ensuite la formule 1. J'ai cependant fait une correction à une des formules :

$$\forall k \in \{2,5\}, \qquad G_k = \max\left(0, \min\left(1, \frac{1}{(N_k|\omega_i)}, \frac{1}{(N_k|\omega_r)}\right)\right).$$

(a) une éraflure sombre. (b) une éraflure peu visible. (c) une éraflure claire.

FIG. 3: Une éraflure selon l'éclairage. L'éraflure est artificiellement élargie. Les paramètres choisis ici sont : $\alpha = 70$, $\beta = 40$, $PerturbNormalSize = .5$.

En effet G_k représente la proportion de la zone k visible et non à l'ombre. Pour les zones 2 et 5, l'éclairage et la visibilité sont binaires : soit la face est visible soit elle ne l'est pas, mais ne peut l'être partiellement (c'est visible sur le schéma). Maintenant pour savoir si une zone est visible, il suffit de regarder le signe du produit scalaire de sa normale avec la direction de l'oeil. Pour garder l'esprit des formules de [MDG01], on remplace les comparaisons de signe par des min et max, et on obtient la formule ci-dessus. Pour améliorer la ligne d'éraflure, un dégradé est fait sur le bord. On dispose alors d'un certain nombre de paramètres pour régler l'apparence des éraflures (je ne considère pas ici les paramètres des Phong-shaders, mais uniquement ceux spécifiques aux éraflures) :

- CrackSize définit la largeur de l'éraflure dans la texture ;
- PerturbNormalSize règle la proportion de l'éraflure sur laquelle faire un dégradé ;
- Lambda est la largeur réelle de l'éraflure, et sert dans le calcul de p_s ;
- Alpha et Beta sont les angles de l'éraflure.

J'ai choisi de régler α et β une fois pour toutes les éraflures sans le laisser varier d'une éraflure à l'autre pour faciliter l'interfaçage. Cependant il est tout à fait possible de faire des éraflures différentes en superposant deux tels shaders et en envoyant le résultat du premier dans le second. Chacun peut alors avoir ses propres paramètres. On remarque sur l'exemple 3 que les éraflures peuvent apparaître soit plus claires soit plus sombres que le fond selon l'éclairage. Sur l'image 4, la plaque est recouverte d'un grand nombre d'éraflures aléatoires. On remarque que les éraflures brillantes sont réparties sur un cercle, dont le rayon est lié aux angles α et β. Ce type d'effet se remarque parfois l'été sur les carrosserie des voitures.

Critique Les effets de ces éraflures sont relativement réalistes. Cependant, un certain nombre de points pourraient être améliorés :

- les éraflures apparaissent relativement larges. Pour obtenir des éraflures aussi fines que dans la réalité, on est obligé d'utiliser une grande résolution d'images.
- dans la réalité, les éraflures brillantes sont beaucoup plus brillantes, alors que les éraflures sombres sont souvent presque invisibles. Dans mes exemples, les deux sont aussi visibles l'une que l'autre. Ce problème n'est pas spécifique aux éraflures. Il vient

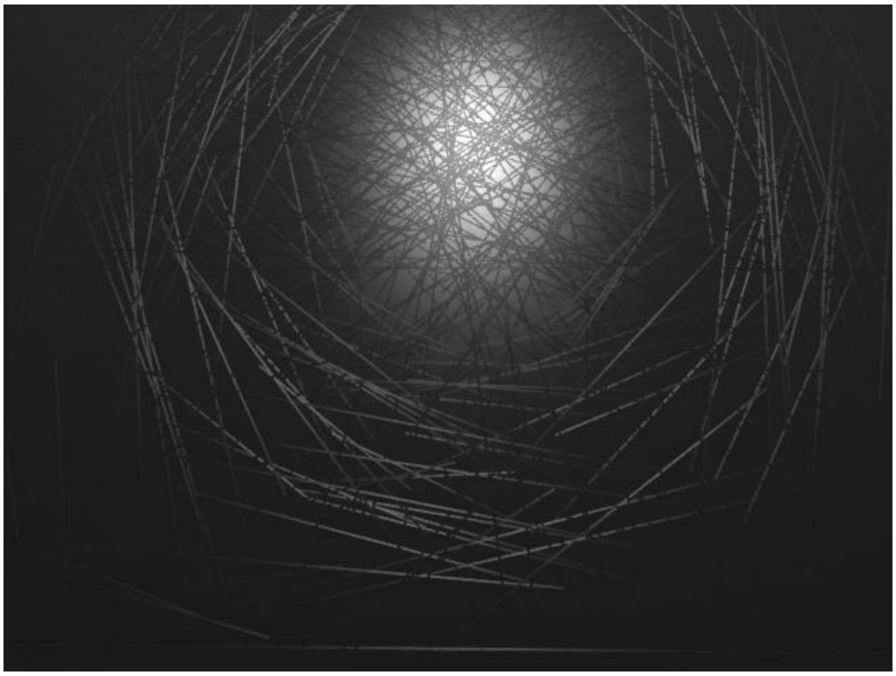
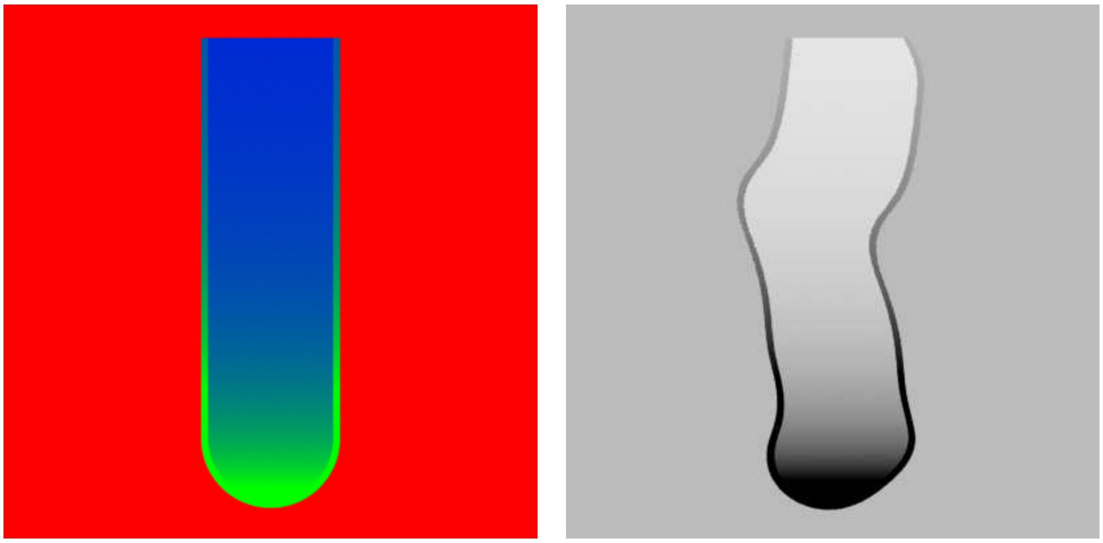


FIG. 4: Un effet classique.

de ce que notre œil voit les zones brillantes plus larges qu'elles ne sont vraiment.
- je ne me suis pas beaucoup préoccupé du positionnement des éraflures. Pour certaines surfaces, le positionnement doit se faire à la main : en bas d'une porte là où les gens donnent des coups, sur le côté exposé d'une montre, etc. Dans le cas de surfaces avec un très grand nombre d'éraflures (expérimentalement : une vitre en plastique), il pourrait être plus économique de ne pas définir les éraflures individuellement mais de donner plutôt une règle de positionnement.
- je ne me suis occupé que des phénomènes de réflexion sur une surface. Un model similaire pourrait être mis en place pour la réfraction car les éraflures sur les vitres sont souvent visibles.

3.2 Gouttes

Pour les gouttes, je ne me suis pas inspiré d'un article en particulier. Il ne s'agit en effet que d'une texture 2D classique. Le but est de représenter la trace laissée par une goutte qui coule le long d'un mur. L'idée est qu'ensuite en juxtaposant un certain nombre de ces traces de gouttes, on pourra rendre l'aspect de saleté de certaines vitres. La forme de base d'une goutte est présentée dans l'image 5(a). Trois couleurs interviennent : la couleur du fond, la couleur que dépose la goutte sur son bord, et la couleur laissée par la goutte derrière elle. Il est ensuite possible de régler la taille de la goutte et la distance sur laquelle elle a coulé. Enfin, pour augmenter le réalisme, on bruite la forme de la goutte (cf. figure 5(b)). Une goutte est définie par sa position finale ainsi que la longueur sur laquelle elle a coulé. Vu de loin, ce modèle donne des résultats assez convaincants, mais de près on réalise que des gouttes se chevauchent. Un modèle simulant la descente des gouttes une à une, même de façon simplifiée, donnerait de meilleurs résultats, mais je ne l'ai pas fait



(a) goutte droite.

(b) goutte bruitée.

FIG. 5: Un modèle de goutte.

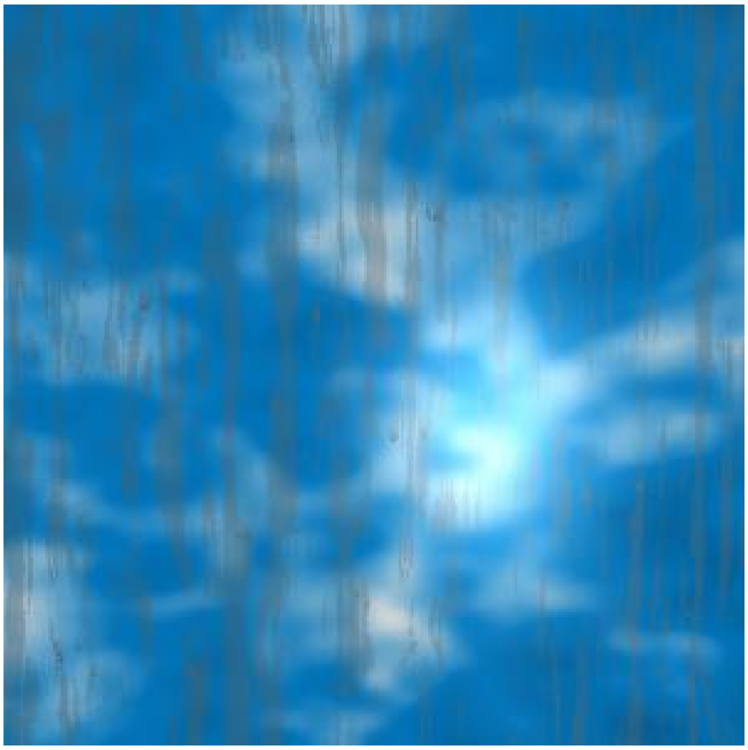


FIG. 6: La fenêtre de la scène présentée en figure 9.

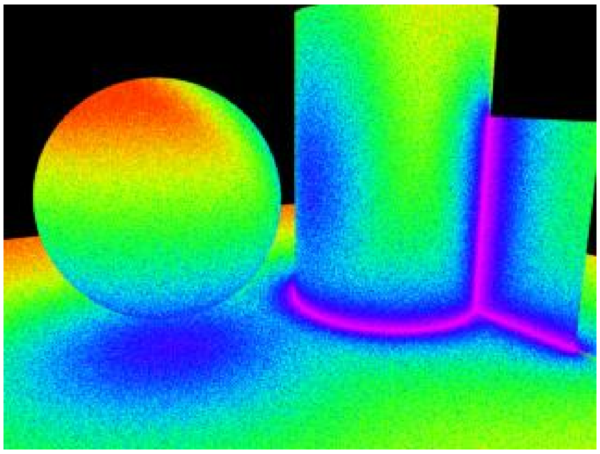


Fig. 7: La méthode de Miller.

pour des raisons techniques que j'explique dans l'annexe A. De toute manière, comme pour les éraflures, le but est de donner un effet convaincant de loin et pas un modèle parfait permettant de zoomer à volonté.

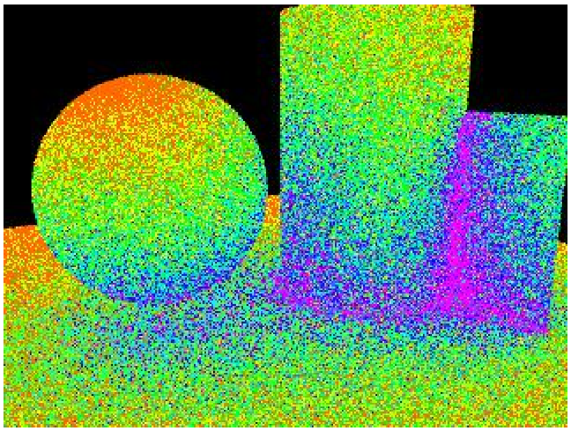
3.3 Poussière

Je me suis intéressé en particulier au lieu de dépôt de la poussière. Les méthodes décrites en 2.2 sont nombreuses, et j'ai choisi d'implémenter une version se fondant uniquement sur l'accessibilité. Techniquement la façon la plus simple d'accéder à la géométrie était de lancer des rayons. Un certain nombre de choix se présentent.

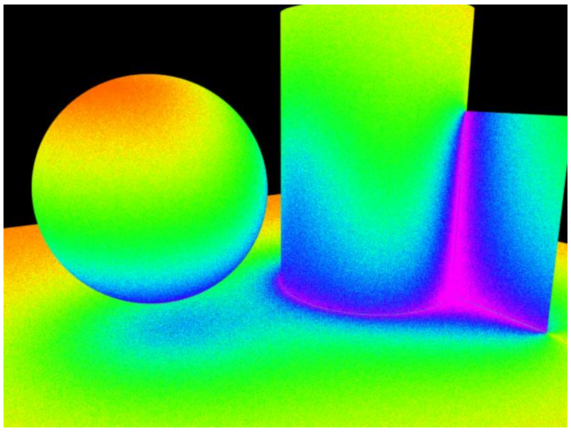
1. L'accessibilité est définie par Miller ([Mil94] et image 7) comme le rayon de la plus grande sphère pouvant être collée à la surface sans intersecter de géométrie. Cependant, j'ai plutôt utilisé une autre définition. Je définis un coefficient de non-accessibilité comme l'intégrale, sur l'hémisphère autour de la normale à la surface, de l'inverse de la distance à l'objet le plus proche.
2. Pour calculer l'accessibilité, je dois lancer des rayons. Je peux alors soit envoyer un certain nombre de rayons aléatoires, soit déterminer à l'avance un certain nombre de directions privilégiées bien réparties sur la demi-sphère dans lesquelles lancer les rayons. La première technique permet d'avoir un résultat acceptable avec peu de rayons, mais il y a un bruit important dans l'image. La seconde génère de forts phénomènes d'aliasing. Les deux problèmes sont réduits par une augmentation du nombre de rayons lancés. Des techniques mixtes permettent de réduire le nombre de rayons à lancer pour un résultat acceptable.
3. Il est aussi possible de prendre en compte l'orientation de la surface pour déterminer la quantité de poussière qui se dépose dessus. En effet la poussière se dépose bien plus facilement sur une surface orientée vers le haut que vers le bas. Si le modèle est utilisé pour mettre de la poussière comme c'est le cas ici, on prendra ce phénomène en compte. Cependant le model d'accessibilité peut aussi servir à modéliser l'encrassement d'objets régulièrement manipulés et dans ce cas on n'utilisera pas l'orientation de l'objet.

On peut dans tous les cas régler un certain nombre de paramètres, qui sont :

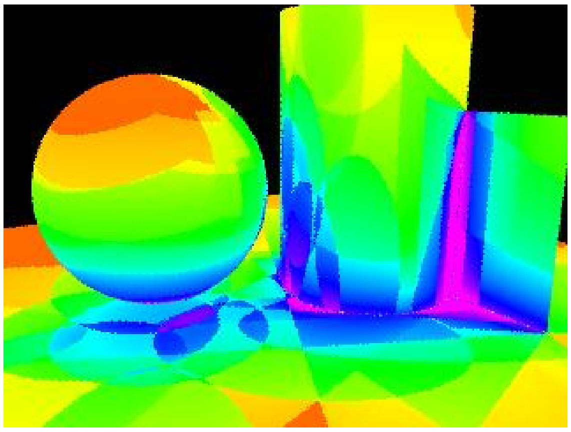
- un coefficient d'adhérence de la surface,



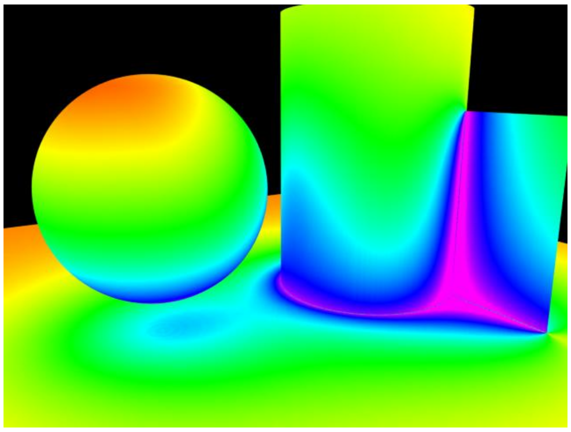
(a) aléatoire (5 rayons)



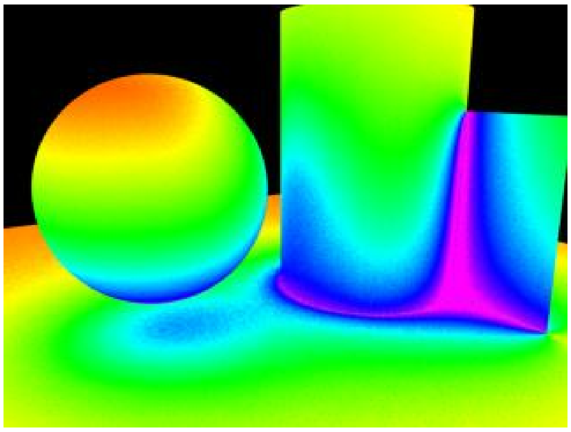
(b) aléatoire (50 rayons)



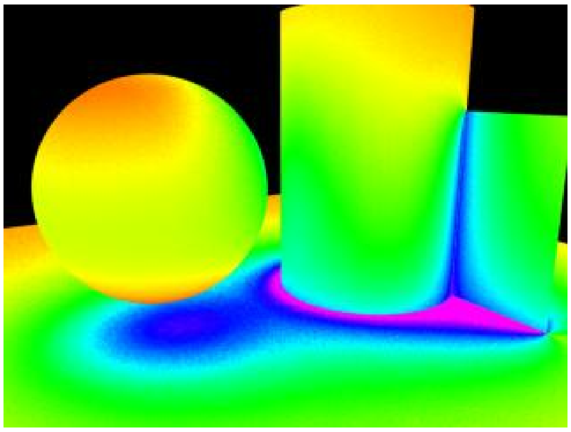
(c) déterministe (12 rayons)



(d) déterministe (2219 rayons)



(e) pseudo-déterministe normal



(f) pseudo-déterministe avec orientation

Fig. 8: Différentes options pour placer la poussière.

- l'angle sur lequel on veut travailler ($\pi/2$ pour une demi-sphère),
- la distance dont on se décale de la surface avant tout calcul (cela évite un certain nombre de bugs liés à la précision des calculs),
- le nombre de rayons à envoyer.

La figure 8 présente sur un exemple des résultats obtenus avec différentes options. Sur ce type de scènes, ma définition de la non-accessibilité semble plus convaincante que celle de Miller, plus adaptée elle à la crasse sur des objets régulièrement manipulés. Cependant la technique consistant à utiliser le lancer de rayons pour accéder à la géométrie donne des temps de calculs trop importants et il serait bon de calculer l'accessibilité par une méthode plus directe.

4 Résultats

La scène 9 utilise les différentes techniques présentées jusqu'ici. Il y a une unique lampe dans la pièce, dont on peut deviner la position grâce au reflet sur la plaque. De la poussière a éte accumulée dans toute la pièce avec des coefficients différents suivant les objets et en tenant compte de l'orientation des surfaces. La vitre initialement uniformément sale a été partiellement lavée par une petite pluie. Le ciel est une texture 3D de nuages de maya associée à une sphère englobant la scène. Le rendu est fait par ray-tracing (obligatoire pour le bon fonctionnement du shader de poussière).

5 Conclusion

À travers ces trois exemples de plug-ins, on a vu certaines méthodes utiles pour le vieillissement des scènes. Cependant, on voit en même temps la faiblesse de ce travail. À chaque nouveau petit effet, il faut refaire un nouveau plug-in ; pour rendre la multitude des effets du vieillissements il faudrait un nombre invraisemblable de plug-ins. Dans sa thèse, Éric Paquette va essayer de simuler tous les effets pouvant se modéliser comme une succession d'impacts sur une surface rigide. Cela fait déjà une catégorie assez vaste, mais cela reste insuffisant. D'autres techniques sont donc à l'étude. L'une d'entre elles consisterait à partir de photographies de surfaces réelles ou de zones de surface vieillies à la main par un artiste et à en tirer un shader réaliste pour une scène beaucoup plus vaste. Cette idée paraît simple mais elle est en fait complexe à mettre en œuvre et à ma connaissance n'est pas beaucoup utilisée. Le but ultime de la thèse d'Éric est de mettre en oeuvre une combinaison automatique des différents types d'usure pour éviter la partie la plus fastidieuse de la tâche.

A Description des plugins maya

J'ai utilisé pour ce stage le logiciel Maya® de Alias|Wavefront. Ce logiciel permet de plaquer des textures sur les géométries de nos scènes, et autorise l'utilisation de shaders utilisateurs sous forme de plugins que l'on peut charger dans Maya. Il fournit aussi une interface simple pour paramétrer ces plugins. Je vais essayer de donner un apreçu de son fonctionnement. Maya est fondé sur un principe de graphes de dépendances. Chaque objet est un nœud. Par exemple une sphère est un nœud avec des attributs tels que sa taille, la précision de son maillage, etc. Elle est connectée à un nœud de positionnement indiquant la position de son centre, son angle de rotation, son échelle. Une fois qu'on a plusieurs objets,

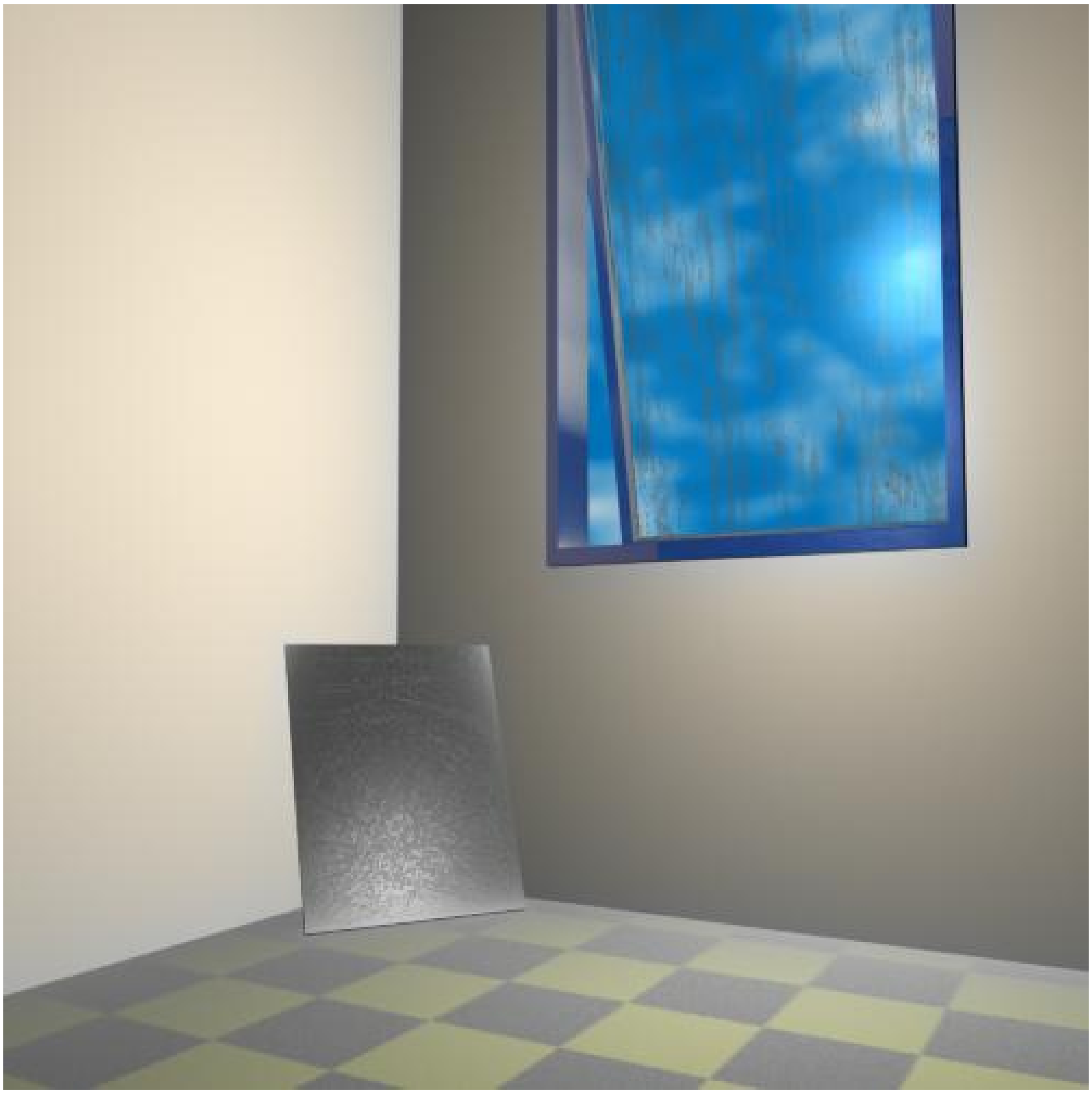


Fig. 9: Une pièce peu fréquentée...

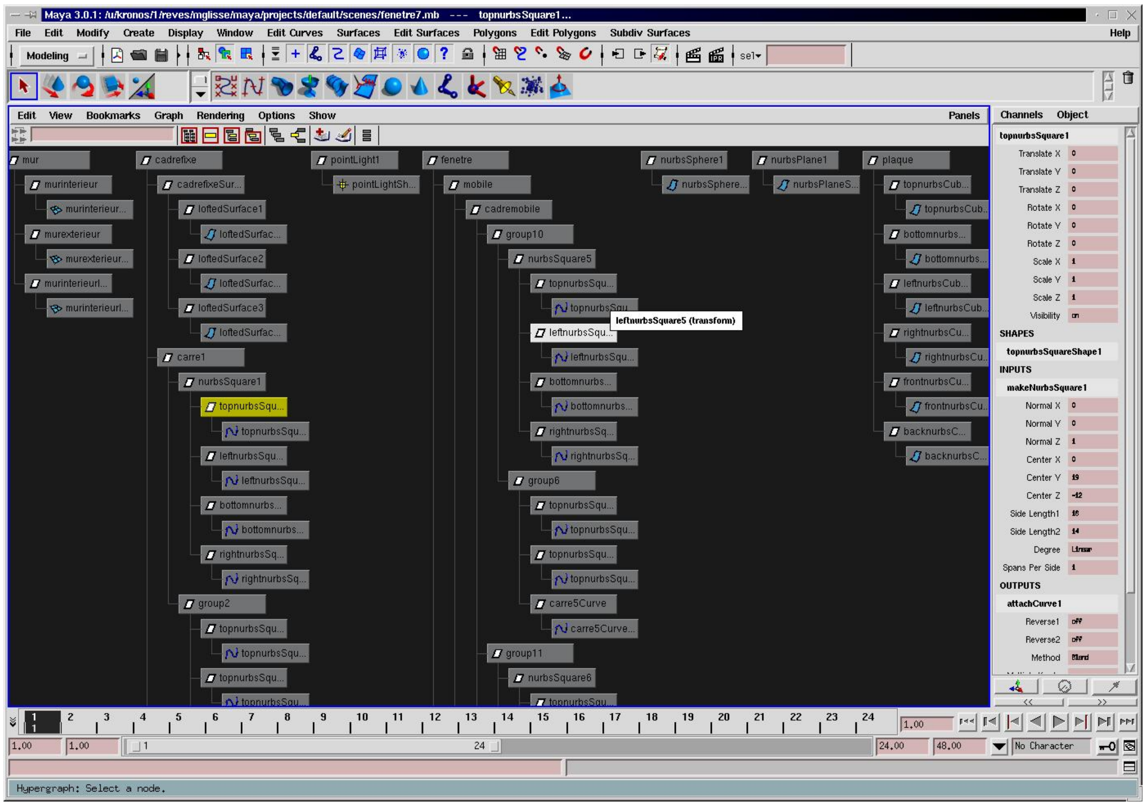


FIG. 10: Un exemple de “Hypergraph Panel” pour la scène 9.

on peut les grouper en un nœud que l’on déplacera ou changera de taille globalement. C’est en passant à travers tous ces nœuds de la racine de l’arbre jusqu’au feuilles qu’on obtient les valeurs réellement utiles. De même pour les shaders, il y a au moins un nœud expliquant l’interaction de la lumière avec la surface et un noeud pour positionner cette texture sur la surface. Ces diverses connexions sont affichées dans les “panels” suivants : hypergraph (figure 10) et hypershade (figure 11). L’éditeur de connexions permet de gérer soi-même les liens entre les différents nœuds, ce qui est notemment très pratique dans l’exemple 11. Maya est codé en C++ et je vais essayer de donner un aperçu de la classe de shaders que j’ai utilisée. La classe de base pour les nœuds de dépendance définis par l’utilisateur est MPxNode. Un nœud calcule des attributs de sortie à partir d’attributs d’entrée. Quand un attribut change, un nouveau calcul se propage à travers le graphe jusqu’à ce que toutes les valeurs affectées soient mises à jour. Cette classe possède ainsi plusieurs méthodes utiles dont voici quelques exemples :

- addAttribute permet de créer un nouvel attribut,
- attributeAffects permet de définir les relations de dépendance entre attributs,
- compute est appelée pour mettre à jour les attributs sortants.

Chacun de mes plug-ins est une classe dérivée de celle-ci. En plus des attributs que je définis moi-même et de la couleur de sortie, un certain nombre d’attributs sont fournis par maya comme la liste des lumières, la normale à la surface, la matrice de changement de coordonnées entre l’espace caméra et le monde réel, la distance à l’œil, le nombre de réflexions ou de réfractions que le rayon à déjà subi en cas de ray-tracing, etc. Prenons l’exemple de la vitre. Le shader directement sur la vitre est un shader maya dont le plug-in

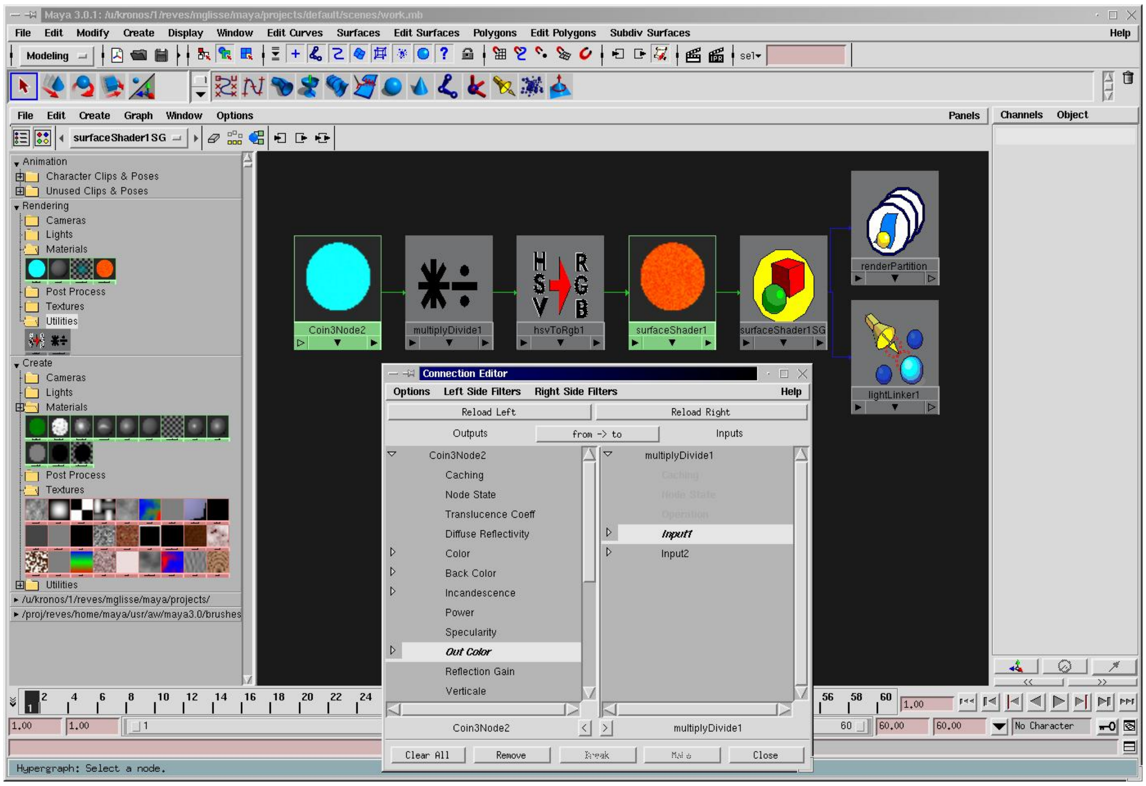


FIG. 11: Un exemple de “Hypershade Panel” pour la poussière en fausses couleurs (figure 8).

de gouttes détermine la transparence. Un des principaux problèmes que j’ai rencontrés avec maya est que l’on ne peut accéder aux attributs d’entrée qu’au sein de la méthode compute. Il n’est donc pas prévu de méthode propre pour faire un calcul au début du rendu dont le résultat soit utilisé ensuite par les appels à compute en chaque point. Par exemple pour les gouttes, il aurait été agréable de pouvoir d’après les paramètres entrés par l’utilisateur faire simuler la chute des gouttes sur une vitre puis utiliser le résultat comme une image 2D à plaquer. Apparemment il y a des possibilités, mais ce n’est pas ce qui est recommandé à l’utilisateur et la documentation est obscure sur ce point.

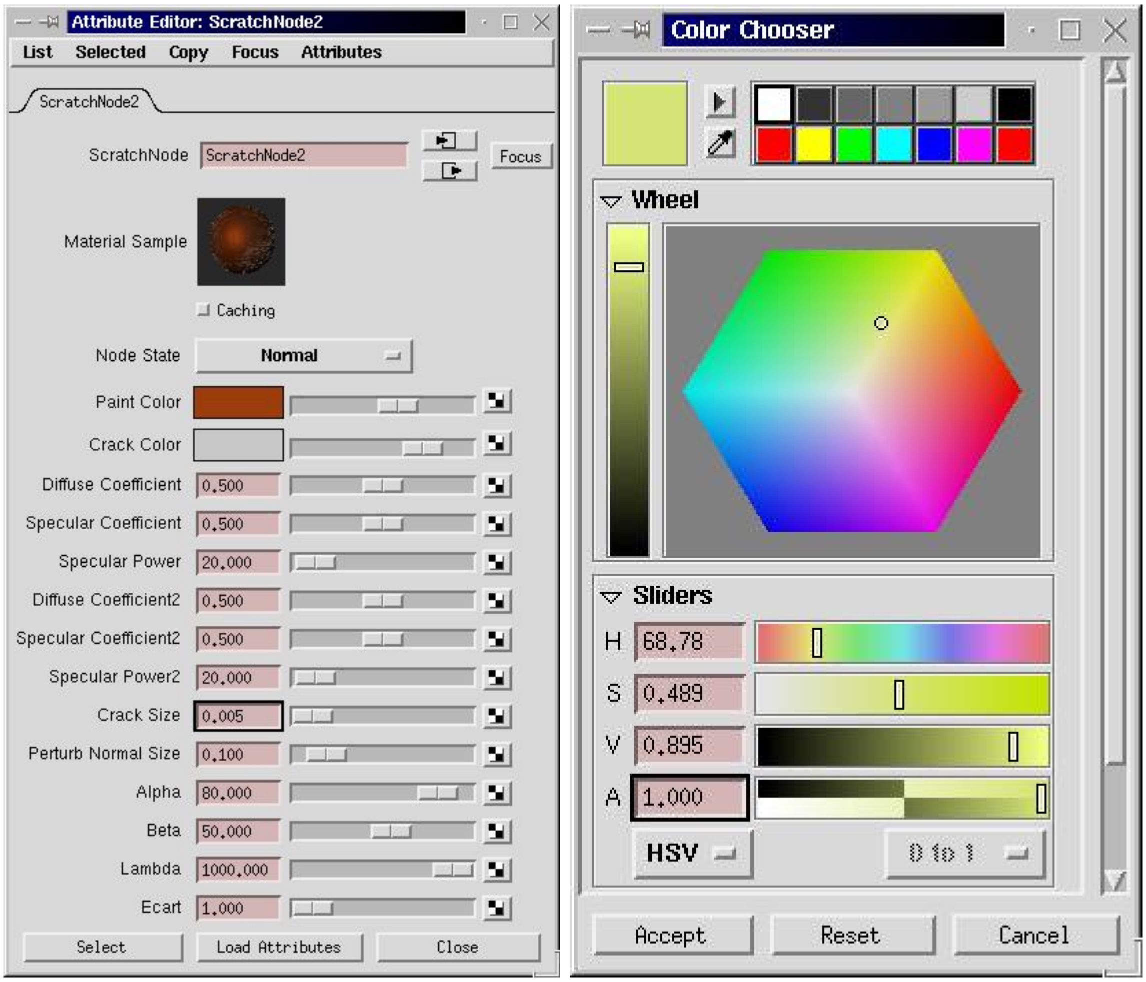
B Utilisation de mes programmes

On charge un plug-in maya grâce au Plug-in Manager. Une fois le plug-in chargé, on peut créer un nœud correspondant comme n’importe quel shader prédéfini de maya. On l’assigne aussi de la même façon à un objet.

B.1 Éraflures

Les sources de ce plug-in sont disponibles sur : `http://www.eleves.ens.fr:8080/home/glisse/inria/scratchNode.cpp`. La figure 12 présente l’éditeur d’attributs associé aux éraflures. Voici la signification des attributs qui apparaissent :

- Paint Color : couleur normale de la surface ;
- Diffuse Coefficient : proportion de la lumiere diffusée par la surface ;



(a) Attributs

(b) Choix d'une couleur

FIG. 12: Les attributs des éraflures.

- Specular Coefficient : proportion de la lumière ayant un comportement spéculaire ;
- Specular Power : puissance intervenant dans la spécularité (coefficient $\cos(\theta)^n$) ;
- Crack Color, et les attributs se terminant par 2 : la même chose pour la partie centrale de l'éraflure. Ce sont les attributs usuels d'un shader de type "Phong", sauf qu'on distingue parfois la couleur diffuse et la couleur spéculaire ;
- Crack Size : largeur d'une éraflure sur l'image. Il faut être proche de la résolution de l'image ;
- Perturb Normal Size : proportion de l'éraflure sur laquelle on fait un dégradé ;
- alpha : l'angle extérieur de l'éraflure ;
- beta : l'angle intérieur de l'éraflure ;
- lambda : la largeur réelle de l'éraflure ;
- ecart : devrait permettre d'espacer un peu les scratchs, mais son rôle empiète sur celui du nœud de plaquage de la texture. Cet attribut sera reconverti en un choix entre différentes dispositions des scratchs.

La disposition des éraflures est définie avant la compilation dans un fichier `./cracks/test.crk`. Si on veut changer cette liste, il faut soit écrire un nouveau fichier à la main avec la même syntaxe, soit en générer un automatiquement. Pour la figure 3, j'ai écrit les coordonnées moi-même. Pour la plaque présente dans la scène, j'ai utilisé un générateur aléatoire dont le principe est le suivant : on prend deux point au hasard dans un carré et on trace une éraflure entre les deux ; le processus est itéré jusqu'à obtention d'un nombre suffisant d'éraflures ; on considère éventuellement seulement un carré au centre du carré d'origine pour éviter d'avoir des bords trop peu rayés. Dans une seconde version, on a plusieurs fichiers de listes d'éraflures compilés entre lesquels on peut choisir grâce au paramètre *écart* (qui mériterait d'ailleurs d'être rebaptisé pour l'occasion).

B.2 Gouttes

Les sources sont sur :
`http://www.eleves.ens.fr:8080/home/glisse/inria/goutteNode.cpp` .
L'éditeur d'attributs a l'aspect présenté par la figure 13(a). Les attributs sont :
- Back Color : couleur de la fenêtre avant la chute de gouttes ;
- Sale Color : couleur de goutte séchée ;
- Propre Color : couleur derrière la goutte ;
- Crack Size : amplitude du bruit ;
- Perturb Normal Size : proportion de la goutte à l'intérieur (par opposition au bord) ;
- Degrade Bord : coefficient de saleté du bord ;
- Degrade Interieur : coefficient de saleté de l'intérieur ;
- Frequencemin : fréquence minimale du bruit ;
- Frequencemax : fréquence maximale du bruit ;
- Amortissement Degrade : coefficient pour régler la vitesse du dégradé ;
- Taille Goutte : multiplie la taille de la goutte ;
- Espace Goutte : multiplie les coordonnées du point sur la texture, en redondance avec le positionnement de la texture ;
- Longueur Goutte : multiplie la longueur des gouttes.

Le fichier définissant les caractéristiques de chaque goutte est très semblable à celui des scratchs. Il se trouve à l'adresse `./cracks/test.gout`. La modélisation a simplement l'avantage de laisser plus de liberté pour régler indépendamment les différentes caractéristiques de la goutte interactivement. Les gouttes ont elles aussi été générées aléatoirement par un point aléatoire dans un rectangle contenant le carré utile et descendant un peu plus bas (pour éviter que les gouttes s'arrêtent en bas de la fenêtre alors que c'est en pratique l'endroit qui en reçoit le plus).

B.3 Poussière

Le shader de poussière présente un petit défaut de conception qu'un autre corrigera peut-être pour moi. Il ne faut pas charger ce plug-in directement sous peine de planter maya. Il est nécessaire de charger auparavant un autre plug-in tel que celui des gouttes ou celui des éraflures. Des sources sont disponibles sur (remplacer X par un chiffre) :
`http://www.eleves.ens.fr:8080/home/glisse/inria/poussiere.X.cpp` .
Il existe un assez grand nombre de versions de ce plug-in, je vais donc essayer de décrire la signification générale des attributs.

(a) Gouttes (b) Poussière

FIG. 13: Les attributs de deux plug-ins.

- Back Color : c'est la couleur de la surface sous la poussière. Cette couleur est appliquée directement, elle est normalement calculée par un shader adéquat pour cette surface ;
- Color, Incandescence, Diffuse Reflectivity, Power, Specularity et Reflection Gain : attributs classiques d'un shader, ici pour la poussière. Il est préférable de faire un shader spécifique au rendu de la poussière et d'envoyer son résulat dans incandescence en mettant Color à noir et Reflection Gain à 0 pour un meilleur résultat.
- Verticale : indique l'inportance à apporter à l'orientation de la surface. À 0, la poussière ne dépasse pas des orientations horizontales. À 1, elle s'arrête juste aux surfaces orientées vers le bas. En mettant un très grand coefficient, l'effet de l'orientation disparaît ;
- Difftheta : pas de la variation de θ entre 0 et Thetamax (version déterministe) ;
- Thetamax : angle maximal sous lequel envoyer des rayons.

- Nbpoints : dans la version aléatoire, c'est le nombre de rayons envoyés de chaque point pour calculer l'accessibilité. Dans la version déterministe, c'est un coefficient jouant sur le nombre de rayons envoyés pour chaque θ, qui devrait en théorie être proche de 2∗Thetamax/Difftheta, mais que l'on règle en fait au jugé pour réduire l'aliassage ;
- Epsilon : distance dont on s'écarte de la surface avant de lancer des rayons. Expérimentalement, il semble qu'on puisse même prendre 0 ;
- Alpha : si un rayon intersecte une surface à une distance trop faible, on arrondit à Alpha ;
- Beta : si un rayon intersecte une surface à une distance trop grande, on arrondit à Beta ;
- Gamma : quantité de poussière ;
- Delta : amortissement dans le calcul de l'accessibilité. Se règle à l'œil ;
- Coef Poussiere : amortissement dans le dépôt de la poussière ;
- Max Poussiere : ancien attribut disparu de la plupart des versions.

La version que j'imaginais à terme de ce plug-in ne comprenait pas de Shader ni pour le fond ni pour la poussière, mais il était pratique d'en avoir un pour les essais, c'est pourquoi les différentes versions de ce programme ne sont pas forcément compatibles.

Bibliographie

Références

[Bli78] J. F. Blinn, *Simulation of wrinkled surfaces*, Proceedings of SIGGRAPH '78, ACM SIGGRAPH, August 1978, pp. 286–292.

[Bli82] ———, *Light reflexion functions for simulation of clouds and dusty surfaces*, Proceedings of SIGGRAPH '82, ACM SIGGRAPH, July 1982, pp. 21–29.

[FP90] A. Fournier and P. Poulin, *A model for anisotropic reflexion*, Proceedings of SIGGRAPH '90, ACM SIGGRAPH, August 1990, pp. 273–282.

[HW95] Siu-Chu Hsu and Tien-Tsin Wong, *Simulating dust accumulation*, IEEE Computer Graphics and Applications **15** (1995), no. 1, 18–25.

[MDG01] S. Merillou, J.M. Dischler, and D. Ghazanfarpour, *Surface scratches : measuring, modeling and rendering*, The Visual Computer **17** (2001), no. 1, 30–45.

[Mil94] Gavin Miller, *Efficient algorithms for local and global accessibility shading*, Proceedings of SIGGRAPH '94, ACM SIGGRAPH, July 1994, pp. 319–326.

[Paq99] É. Paquette, *Simulation des effets de déterioration de surfaces rigides*, Examen prédoctoral, Décembre 1999.

[Pho75] Bui-T. Phong, *Illumination for computer generated pictures*, Communications of the ACM **18** (1975), no. 6, 311–317.