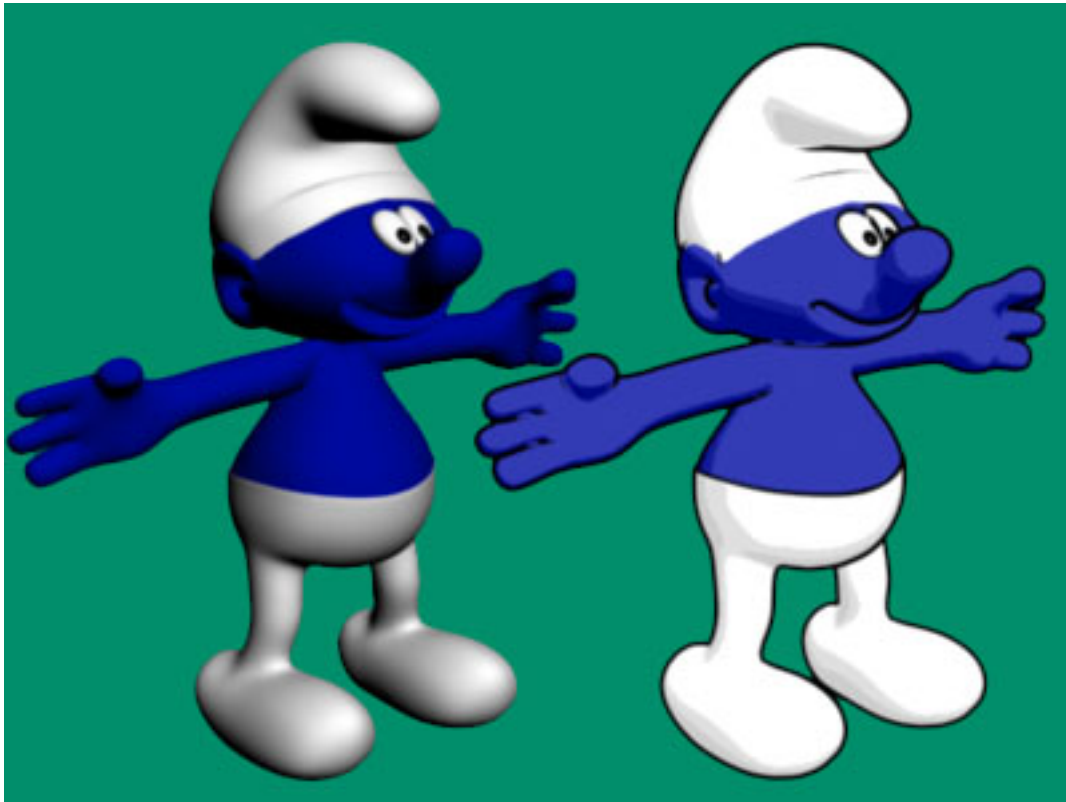


Cel-shading : le rendu cartoon



par Cyril Doillon ([Page Perso](#))

Date de publication : 17 février 2009

Dernière mise à jour :

Le cel-shading est une technique de rendu 3D de plus en plus utilisée dans les jeux et les applications 3D. Le principe est de se rapprocher de plus en plus des dessins animés ou des bandes dessinées. Ce tutoriel va vous donner les moyens d'arriver à de tels résultats par différentes techniques.

Commentez cet article :

I - Présentation du cel-shading.....	4
I-A - Historique.....	4
I-B - Principe.....	5
II - Les techniques d'ombrage.....	7
II-A - Rappel sur la lumière.....	7
II-B - Avec une texture 1D.....	8
II-C - Pixel Shader.....	10
III - Les contours.....	11
III-A - Multi-passes.....	11
III-B - Enveloppe.....	12
III-C - Filtre sur la profondeur.....	14
IV - Conclusion.....	16
V - Remerciements.....	16

I - Présentation du cel-shading

I-A - Historique

A l'image de l'évolution de la peinture artistique, la réalisation de scènes 3D virtuelles a voulu se rapprocher de plus en plus de la réalité avec des environnements de plus en plus détaillés. Mais aux alentours des années 2000, un nouveau courant est arrivé qui voulait se détacher du réalisme pour se rapprocher d'un visuel beaucoup plus "dessins animés". L'optique de cela est de donner l'impression de se retrouver projeté dans l'univers de son enfance, dans lequel on pouvait maintenant entièrement interagir, comme si nous étions le "héros" de ce dessin animé.



Jet Set Radio (2000)

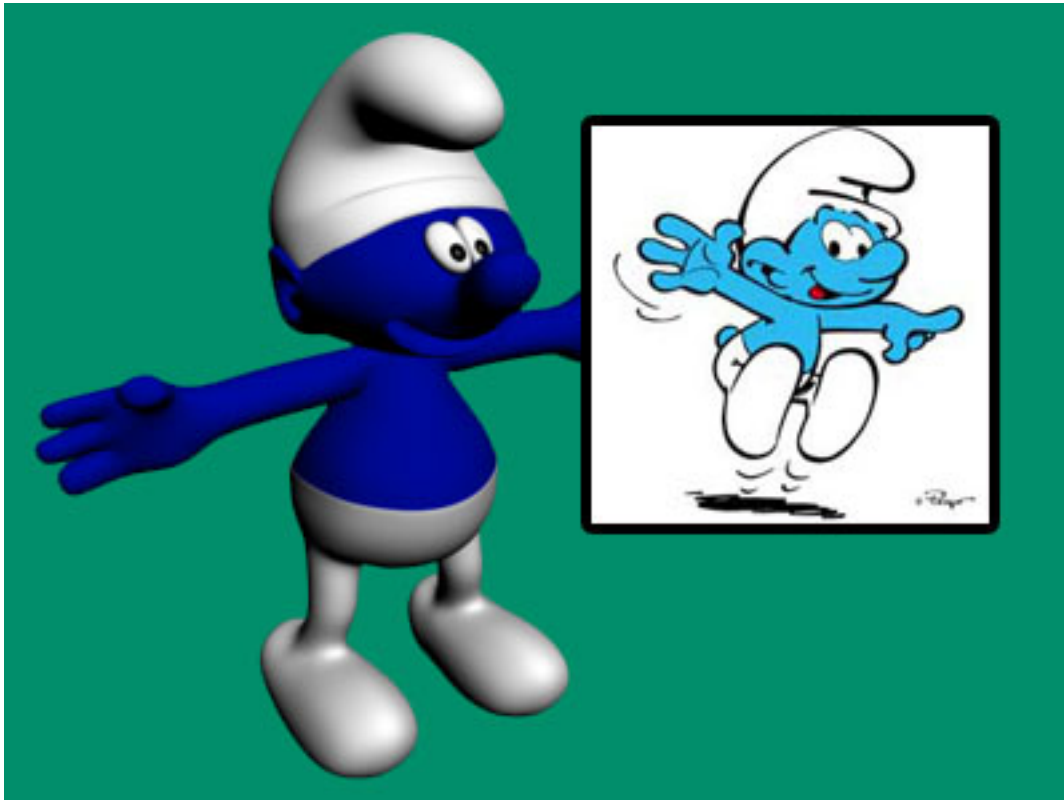
C'est ainsi que le "cel-shading" est né, littéralement "ombrage de celluloïd". Cette technique, aussi appelée "toon-shading", permet d'afficher tous types d'objets 3D, réalistes ou non, sous la forme de dessins illustrés comme les bandes dessinées. Cette technique est apparue pour la première fois au grand public par le jeu "Jet Set Radio" de Sega en 2000. Elle s'est ensuite popularisée par le succès du jeu "XIII" qui est une adaptation très réussie de la série de bandes dessinées du même nom. Dans "XIII", les développeurs avaient même ajouté des effets supplémentaires comme des bulles, des onomatopées, des cases en surimpressions, ... pour donner encore plus l'impression d'être plongé directement dans l'histoire à travers le héros. Il est très courant de voir actuellement de nouvelles productions avec ce type de rendu notamment pour des jeux qui veulent faire oublier le réel aux joueurs.



XIII (2003)

I-B - Principe

L'objectif du cel-shading est de modifier notre rendu pour s'approcher au maximum du dessin des bandes dessinées par exemple. Il faut donc trouver visuellement ce qui caractérise un dessin cartoon par rapport à tous autres dessins ou peintures.



Objet 3D VS Bande dessinée

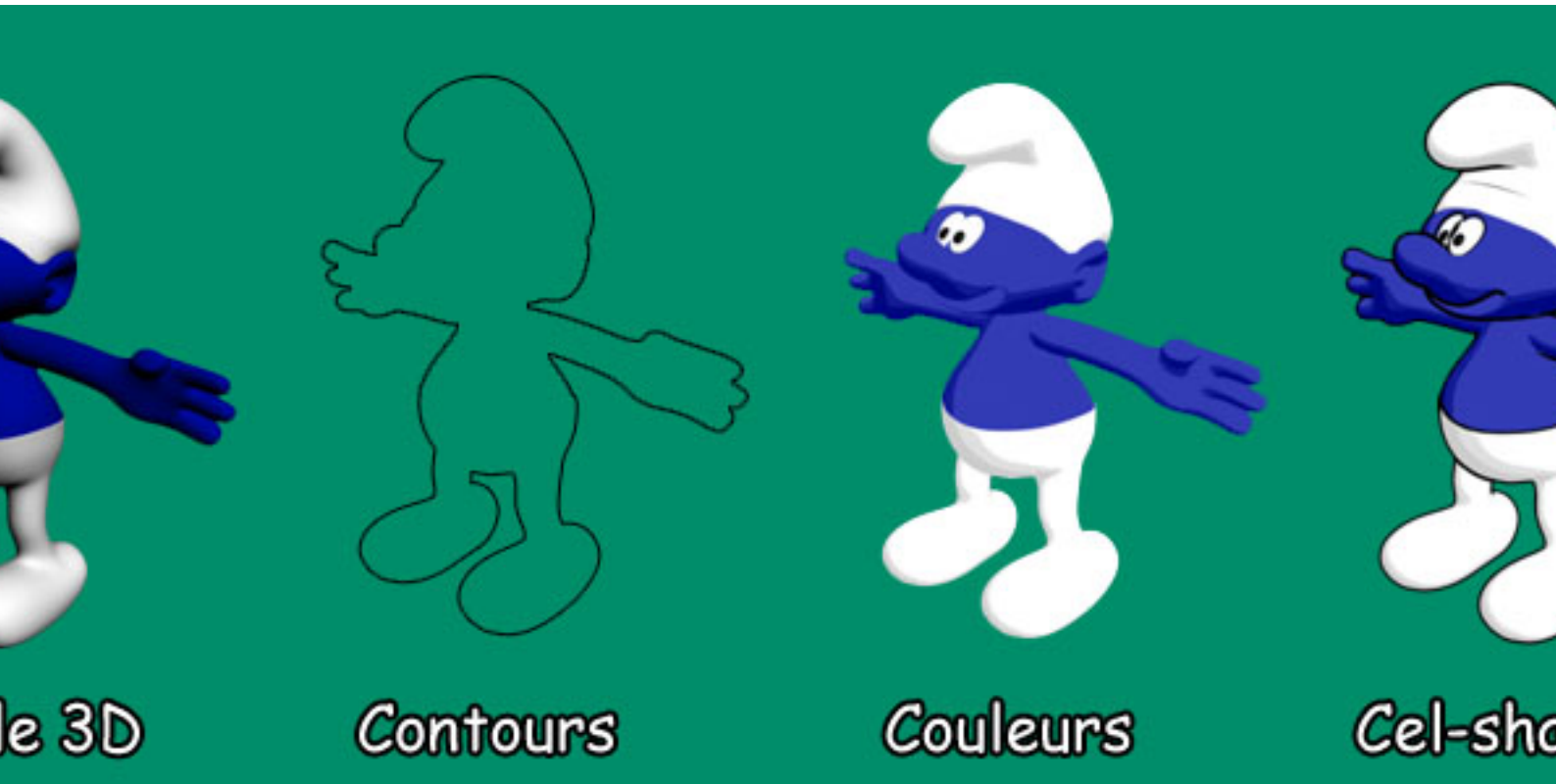
Tout d'abord, la première caractéristique qui saute aux yeux est le contour noir de chaque objet ou personnage du dessin. Ce trait noir, généralement à l'encre, correspond au trait du dessinateur pour dessiner ces objets. Qui n'a jamais griffonné de simples traits noirs pour représenter un petit personnage sur un coin de feuille ? Ce simple contour permet de se détacher du réel car, tout simplement, il n'existe pas dans la vie réelle. Cet élément est donc une des bases principales pour obtenir un rendu cartoon. Nous devons donc trouver une méthode pour créer cette bordure sur nos objets 3D.

💡 *Le trait noir était aussi utile pour se détacher de la nécessité de couleur dans les bandes dessinées surtout lorsque les planches apparaissaient dans des journaux en noirs et blanc.*

Ceci est déjà intéressant pour afficher des scènes 3D sous forme de dessins cartoons, mais un autre élément entre en jeu. Les dessins cartoons sont souvent produits plus rapidement que les dessins réalistes. Est-il possible de passer une semaine sur un dessin d'une planche de bande dessinée quand l'intégralité de l'album est à réaliser ? Ainsi, les couleurs sont beaucoup moins travaillées, notamment au niveau de la qualité des jeux de lumières de l'environnement. Plutôt que de réaliser de beaux dégradés de couleurs, l'artiste va souvent se contenter de n'utiliser que quelques nuances de couleurs pour les différentes luminosités des objets. Il faut donc essayer de reproduire ce dégradé simplifié sur nos objets 3D.

💡 *Pour les anciennes bandes dessinées, le nombre de couleurs utilisables était souvent limité par l'imprimeur donc l'artiste devait obligatoirement se limiter.*

Le cel-shading possède deux composantes bien distinctes : des dégradés de lumière simplifiés et des contours. Nous allons donc voir les deux effets indépendamment car aucun lien n'existe entre eux à part la volonté d'afficher un rendu cartoon. De plus, les deux techniques peuvent être utilisées l'une sans l'autre sans problème en fonction des besoins de votre application 3D ou de votre jeu.



Composition du cel-shading

II - Les techniques d'ombrage

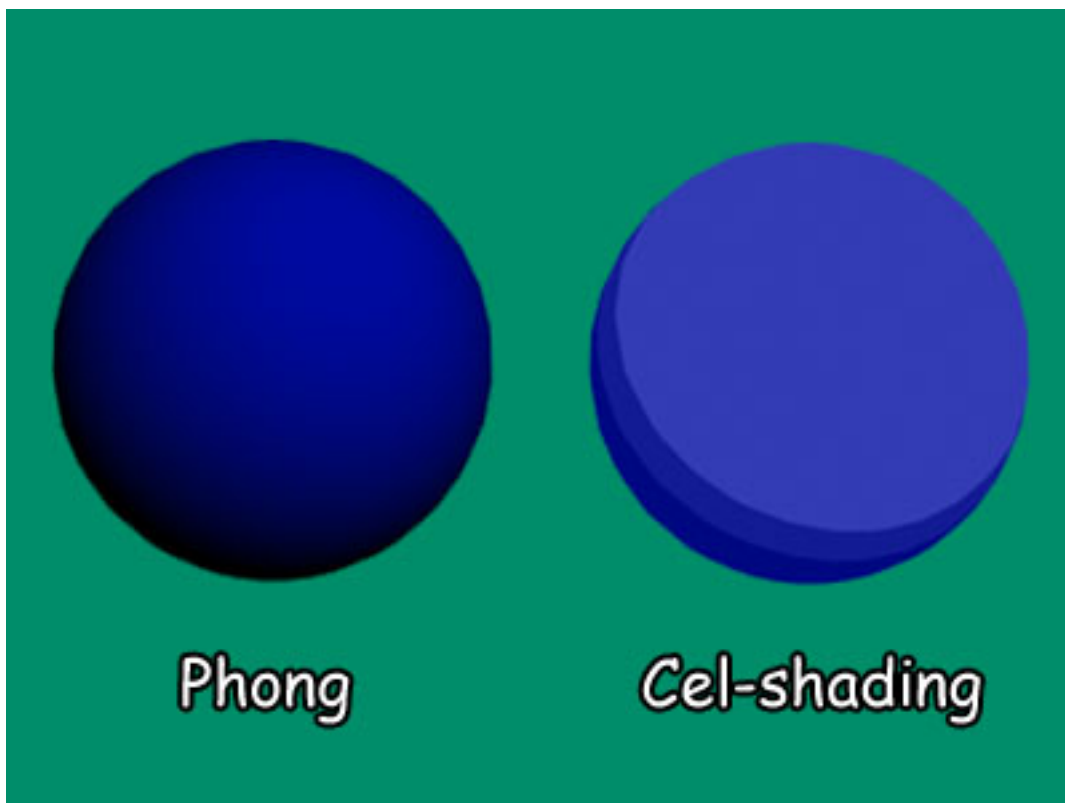
II-A - Rappel sur la lumière

Pour réaliser notre rendu cel-shading, nous allons commencer par la partie "ombrage simplifié" de la technique. Pour cela, nous allons tout d'abord se rappeler comment est calculée la lumière dans les modèles classiques de rendu. Le modèle le plus souvent utilisé est "l'ombrage de Phong". Celui-ci se compose de trois éléments de lumières : ambiant, diffuse et spéculaire.

- La lumière ambiante correspond à la couleur apparaissant dans la scène lorsque toutes les lumières sont éteintes.
- La couleur diffuse correspond à la réaction de l'objet à la lumière.
- La couleur spéculaire correspond à la lumière réfléchiée sur l'objet.

Pour simplifier, nous n'allons pas prendre en compte cette lumière "spéculaire". La couleur résultante sur un point de l'objet s'obtient par un calcul très simple. Il suffit d'ajouter la partie ambiante et la partie diffuse, où cette dernière est modulée en fonction de l'angle formé entre la normale au point et la direction de la lumière.

L'ombrage résultant par cette technique sur une sphère est un dégradé de couleur allant d'une zone claire (zone éclairée directement) à une zone foncée (zone non éclairée). Ce dégradé provient de l'évolution de la normale de la surface de l'objet. Cela donne un réalisme et une profondeur à l'objet. Or, pour un rendu cartoon, nous voulons "casser" ce réalisme et donc ce dégradé en influant sur cette composante "diffuse".



Simplification de la lumière

II-B - Avec une texture 1D

⚠ Il est fortement conseillé de se renseigner sur le **Multitexturing avant de lire cette partie.**


La première approche pour créer un effet de lumière "simplifié" est l'utilisation d'une Texture 1D pour la lumière. La technique est assez simple mais propose une approche originale. Plutôt que de calculer explicitement la lumière en chaque point de l'objet, nous allons le simuler. Pour cela, nous allons décomposer la couleur diffuse de notre objet en deux éléments : couleur et lumière. La partie "couleur" correspond à la couleur de l'objet lorsqu'il n'est pas éclairé par une lumière. Celle-ci peut être une simple couleur pour la surface complète ou une texture 2D classique.

Pour la composante "lumière" de la couleur, nous allons utiliser une texture à une dimension qui va simuler notre dégradé de lumière. Plus la texture aura une grande résolution, plus la précision du dégradé sera grande. Comme vous l'avez sûrement compris, l'objectif pour nous est de réduire la qualité de la texture pour "simplifier" la lumière.

L'utilisation de cette texture nécessite souvent que la carte graphique accepte le multitexturage car nous avons besoin de deux textures par objet, une pour la couleur et une pour la lumière. La modulation de ces textures va se faire par multiplication. Ainsi, les zones utilisant la partie foncée de la lumière apparaîtront foncées, et inversement pour la partie claire. Au niveau des coordonnées de texture, la partie "couleur" utilise les coordonnées classiques de notre objet et la partie "lumière" sera plus particulière. Pour chaque sommet, nous avons besoin d'une coordonnée entre 0 et 1 sur notre texture. Cette valeur sera calculée grâce à l'angle entre la direction de la lumière L et la normale au sommet N . Plutôt que de calculer l'angle directement, nous allons directement utiliser le produit scalaire des deux vecteurs. Pour un produit scalaire égal à 1, la lumière est maximale donc la coordonnée doit être à 1. Pour un produit scalaire inférieur ou égal à 0, la lumière est minimale donc la coordonnée doit être à 0. Ainsi :

```
U = L.N  
Si U < 0 alors U = 0
```

- Avec L, direction normalisée de la lumière
- Avec N, normale de la surface au point
- Avec U, coordonnée résultante sur la texture de "lumière"

 Pour éviter le dernier test pour un produit scalaire négatif, nous pouvons définir la texture pour qu'elle se "prolonge" pour des valeurs sortant du domaine $[0, 1]$.

Par cette technique, il nous suffit donc de "seuiller" la texture de lumière pour obtenir notre effet "cartoon". Pour une texture à 16 niveaux de gris par exemple, nous pouvons la transformer pour n'utiliser que 4 niveaux et ainsi "simplifier" le modèle de lumière.




Texture de lumière classique



Texture de lumière seuillée

Texture 1D de lumière

II-C - Pixel Shader

 Il est fortement conseillé de se renseigner sur l'utilisation des **Pixel Shader** (ou **Fragment Shader**) avant de lire cette partie.

Avec les évolutions des cartes graphiques, le rendu d'objet 3D réaliste a été grandement amélioré. L'apparition des "pixel shader", qui permettent un calcul défini par l'application pour chaque pixel de l'écran, fût l'une des améliorations qui augmenta les capacités des affichages. Grâce à eux, il est maintenant possible de calculer directement la couleur d'un pixel de façons très variées. On peut notamment calculer la lumière en un point de l'écran.

Le calcul utilisé dans les pixels shaders est identique aux méthodes classiques à savoir avec des parties ambiantes, diffuses et spéculaires. Nous n'avons ainsi plus besoin d'une texture supplémentaire dédiée à la lumière pour dessiner nos objets 3D car le calcul se fait directement sur la couleur du pixel. Grâce aux informations de normales et de lumière, il est possible de calculer notre produit scalaire, vu précédemment, pour obtenir notre "pourcentage" de noir entre 0 et 1 en ce point. Celui-ci pourra ainsi être modulé par la couleur de l'objet, provenant d'une texture ou d'une simple couleur diffuse.

Notre objectif est toujours le même, "simplifier" les nuances de couleurs de la lumière. Nous pouvons donc agir directement sur notre valeur de lumière au sein du pixel shader. Il suffit donc de "seuiller" la valeur avec des valeurs prédéfinies. Par exemple :

```

U = L.N
Si U < 0.2 alors U = 0
Si U >= 0.2 et U < 0.4 alors U = 0.2
Si U >= 0.4 et U < 0.5 alors U = 0.4
Si U >= 0.5 alors U = 1
    
```

Nous obtenons ainsi une nuance de quatre couleurs répartie non-uniformément sur la plage de valeur [0,1]. Notre résultat est donc identique à la version précédente si toutefois, la texture 1D correspond au même seuillage que celui du pixel shader. Cette technique nécessite que la carte graphique supporte les pixels shader mais permet de gagner un emplacement de texture pour chaque objet 3D à dessiner.

III - Les contours

III-A - Multi-passes



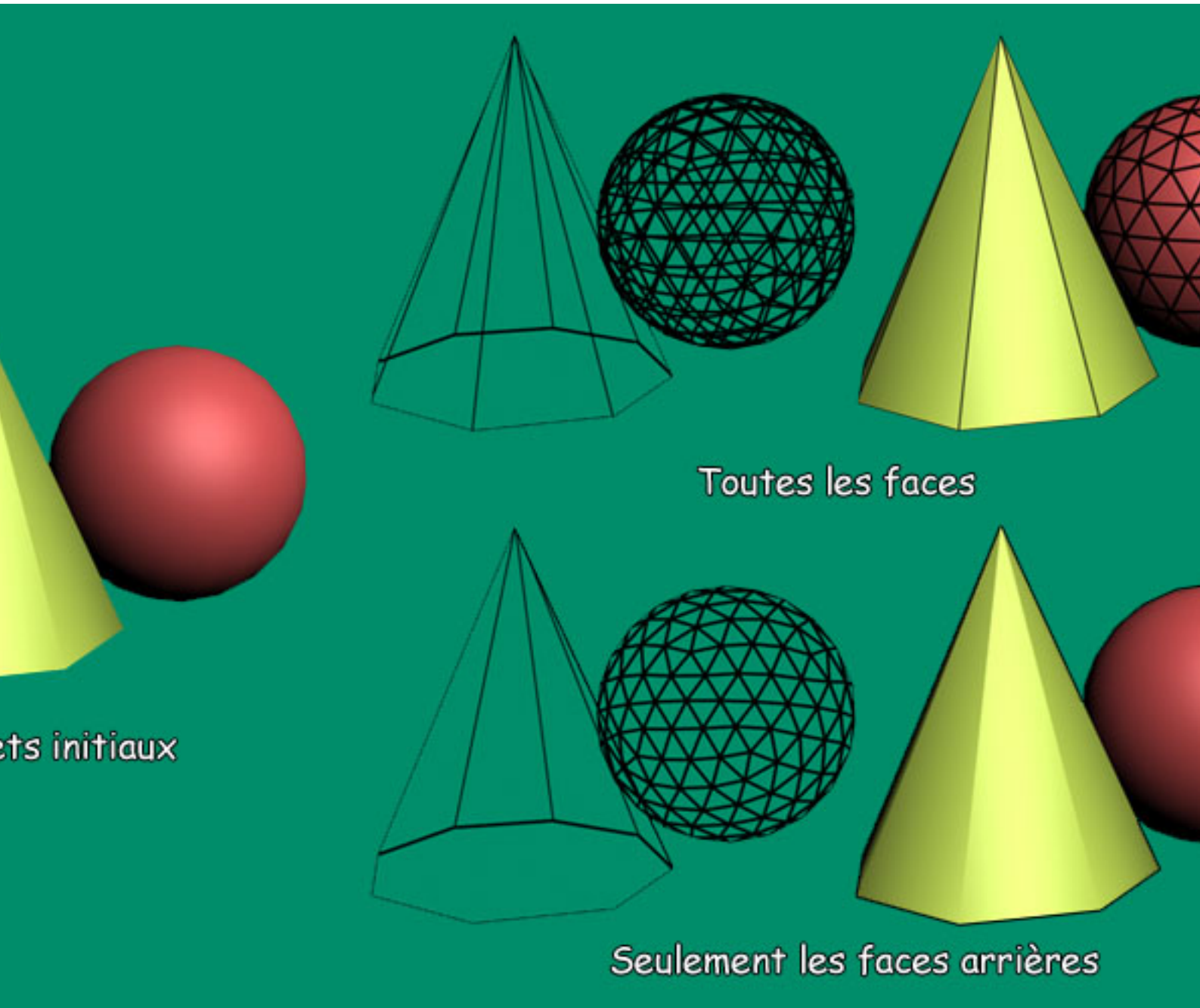
*Il est fortement conseillé de se renseigner sur le **Backface Culling** avant de lire cette partie.*

La deuxième grosse composante du "cel-shading" se situe dans l'affichage des contours de chaque objet. Contrairement à la partie précédente où nous modifions uniquement le dessin de l'objet, ici, nous voulons rajouter un élément supplémentaire. Comment dessiner des pixels noirs supplémentaires et qui suivent les contours de l'objet ?

La première approche que nous allons aborder va essayer d'utiliser la même géométrie que notre objet 3D. De base, chaque objet est dessiné par une composition de nombreux triangles. La particularité liée à l'utilisation de cette forme est que, quelle que soit l'orientation de l'objet, le contour de l'objet correspond toujours à une ou plusieurs arêtes de triangles. Nous allons donc utiliser cette propriété pour faire apparaître les contours de la forme.

Nous allons donc dessiner le modèle de l'objet une seconde fois mais en se contentant de dessiner les arêtes des faces. Si nous dessinons comme cela, nous n'aurons pas un résultat acceptable car l'objet sera dessiné normalement et chacun de ses triangles sera entouré d'un trait noir. Nous allons d'abord filtrer pour ne dessiner que les faces n'étant pas dirigées vers la caméra. Cela conserve effectivement les contours de l'objet car chaque bordure de celui-ci est composée au minimum d'un triangle vers la caméra et d'un autre dans le sens opposé. Nous évitons ainsi de dessiner les arêtes qui se situent au premier plan sur l'objet et nous conservons des surfaces colorées uniformes.

Le principe est donc de dessiner une première fois l'objet 3D texturé et coloré normalement et de le redessiner une seconde fois en mode filaire, en noir et uniquement les faces "arrières". Pour modifier l'aspect du contour, il est possible de changer la couleur du trait ainsi que son épaisseur pour forcer l'effet cartoon. Cette technique possède un désavantage assez important car, quelle que soit la distance de l'objet à la caméra, l'épaisseur du trait sera la même. Ainsi, plus l'objet s'éloignera, plus le contour sera prononcé sur celui-ci car l'épaisseur correspond à une taille fixe à l'écran.



Création du contour d'objets 3D

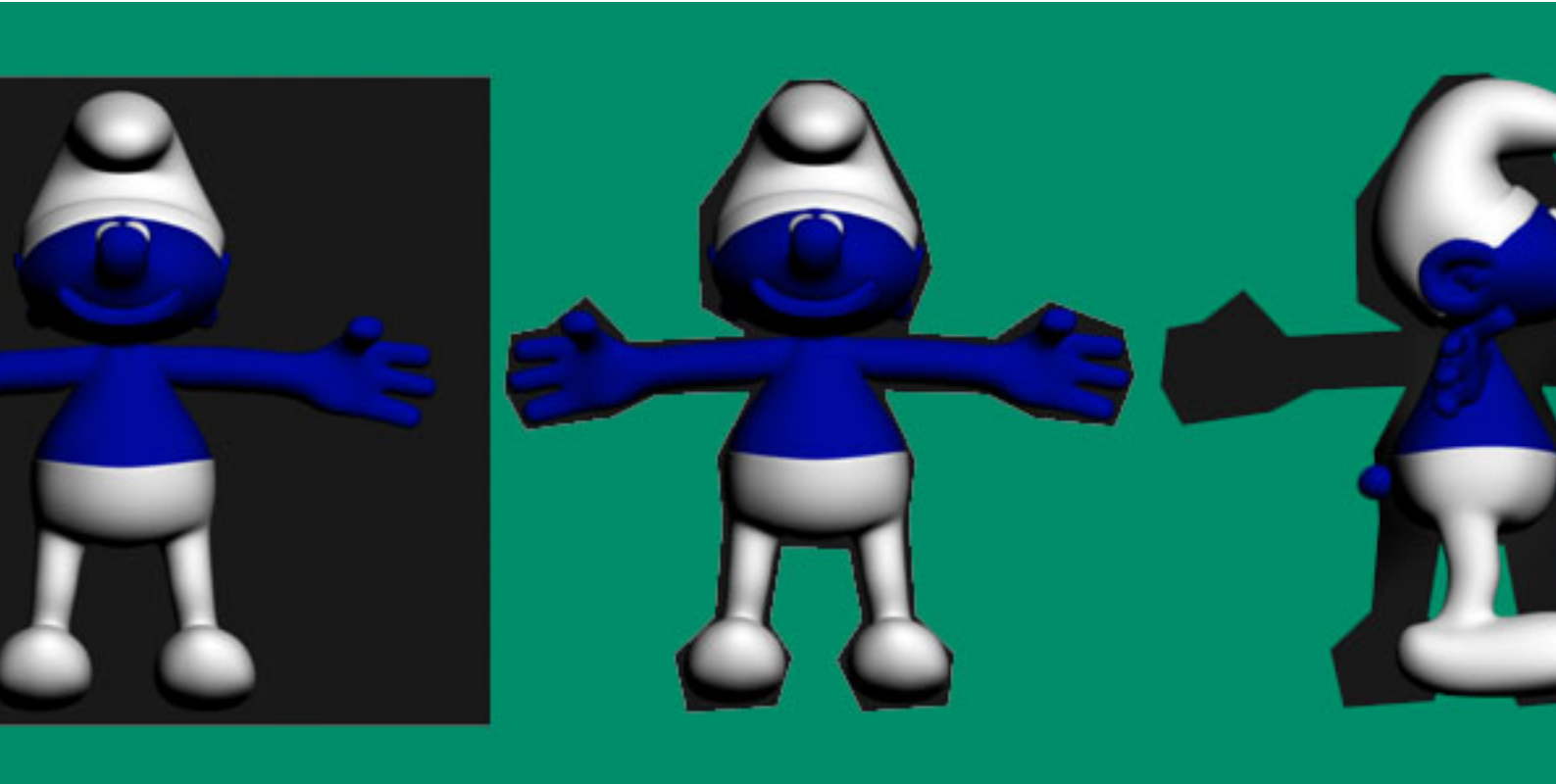
III-B - Enveloppe

! Il est fortement conseillé de se renseigner sur le **Backface Culling** avant de lire cette partie.

Nous nous sommes rendus compte que l'utilisation du même modèle pour dessiner ses contours posait des inconvénients car il oblige à dessiner deux fois chacun des objets et pouvait avoir des effets non désirés sur les objets petits ou éloignés. Pour éviter ce dernier effet, nous allons essayer d'ajouter un nouveau modèle pour chaque objet devant recevoir des contours. Mais comment doit être ce nouveau modèle ?

Le principe est simple. Prenons par exemple un simple rectangle noir et plaçons le derrière notre modèle du point de vue de la caméra. Ce rectangle va dépasser de notre modèle à certains endroits pour former un contour noir mais

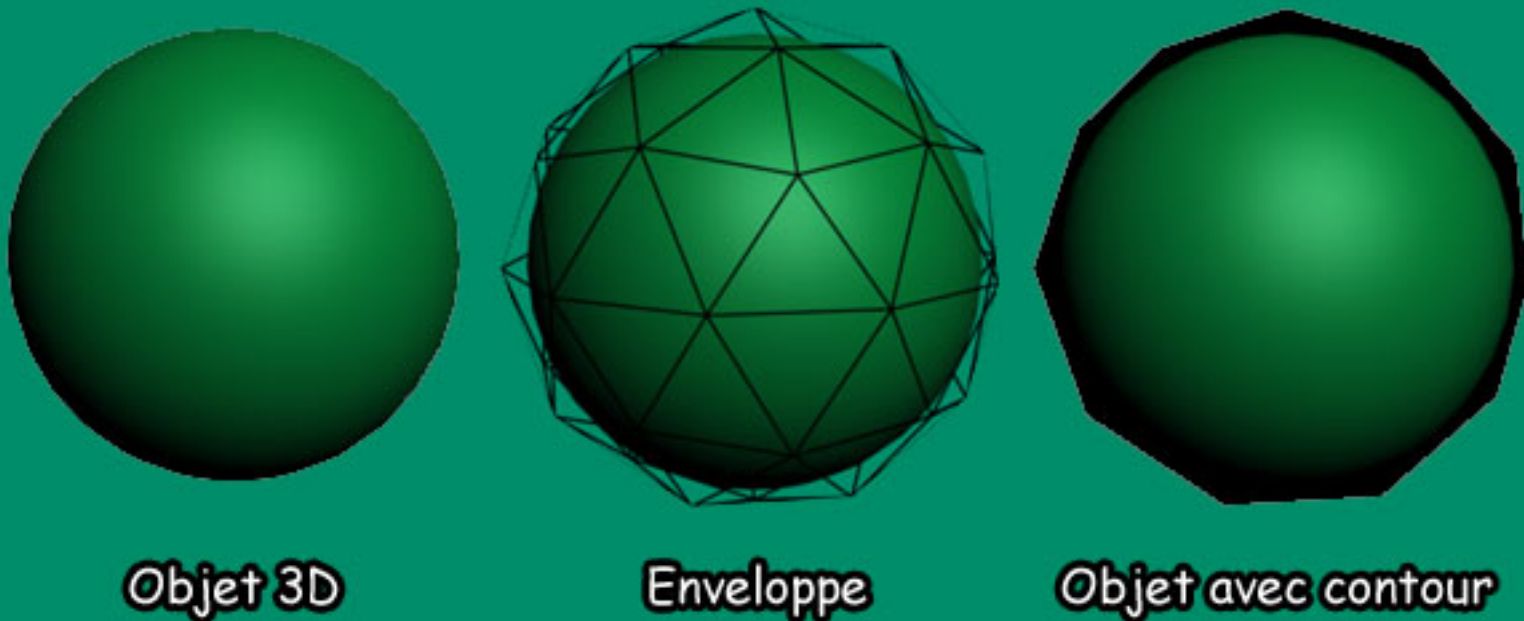
ne suivant pas exactement le modèle initial. Améliorons le rectangle pour qu'il suive assez fidèlement le contour du modèle, nous arrivons à un résultat assez propre.



Rectangle / Adaptation de la forme / Changement d'angle du modèle


Il nous reste cependant un problème, si la caméra bouge, le profil du modèle ne sera plus le même et il faudrait donc modifier notre forme de contour. Pour résoudre ce problème nous allons utiliser un mesh complètement en 3D qui va entourer globalement l'ensemble du mesh initial. Nous nous retrouvons ainsi avec des faces à l'avant du modèle et d'autre à l'arrière. Seules ces dernières nous intéressent donc nous allons dessiner l'enveloppe unique les faces dirigées vers la caméra et vers le centre de l'enveloppe. Ainsi n'apparaîtront que les faces formant le contour de l'objet.

Cette technique augmente significativement la quantité de triangles affichés pour un seul objet et cela peut poser des gros problèmes pour des scènes très complexes. Par contre, elle offre une plus grande souplesse sur la forme des contours qui ne sont plus limités à suivre exactement la forme de l'objet mais peuvent utiliser des angles beaucoup plus marqués. De même, le problème des tailles fixes des contours disparaît car les contours sont intégralement intégrés à la scène et aux objets 3D. Ainsi, plus l'objet sera loin, plus ces contours seront petits.




III-C - Filtre sur la profondeur

Avec les améliorations des cartes graphiques, les techniques de rendu ont aussi évolué. Une nouvelle technique appelée "deferred shading" propose une nouvelle approche du dessin de scène 3D. Cette technique a comme principal intérêt pour nous de calculer l'image finale par pixel de l'écran. Nous pouvons ainsi utiliser des techniques de traitement d'images pour créer les contours de nos objets 3D.

 *Le "deferred shading" a pour principe de dessiner chaque composante (couleur, normal, profondeur, ...) dans des textures différentes et ensuite de calculer la couleur de chaque pixel grâce à un calcul de lumière classique. L'intérêt de la technique est de pouvoir ajouter un grand nombre de lumières dynamiques sans impacter les performances.*

Les techniques de traitement d'image utilisent les convolutions qui calculent un pixel donné à partir d'une zone de pixels centrée sur celui-ci. Des filtres usuels comme Sobel ou Prewitt permettent la détection des contours d'une image. Le principe de ces filtres est simple, si une différence (de couleur généralement) existe entre deux pixels consécutifs cela crée un contour plus ou moins prononcé sur ces pixels. Soit, pour un pixel, plus la différence entre sa valeur et les valeurs de pixels l'entourant est grande, plus le contour sera marqué. Généralement, nous n'utilisons que les pixels directement consécutifs soit un calcul sur 9 pixels (filtre 3x3), mais il peut être utile de prendre des niveaux supplémentaires sur 25 pixels par exemple (filtre 5x5).

 *Vous pourrez trouver plus d'informations sur le traitement d'images dans les tutoriels : **Introduction au traitement numérique d'image** de Florent Humbert et **Les filtres usuels en traitement d'images** de Xavier Philippeau.*

Dans notre cas, nous avons plus d'informations que sur les simples photographies car nous possédons les éléments de profondeur de l'image. Effectivement, un contour sur une image cartoon correspond à une différence de profondeur assez grande entre l'objet à entourer et ce qui se trouve derrière lui. Plutôt que d'appliquer notre filtre sur la couleur de l'image, nous allons directement l'appliquer sur notre image de profondeur. Ainsi, les grandes différences de profondeurs seront marquées par un contour.

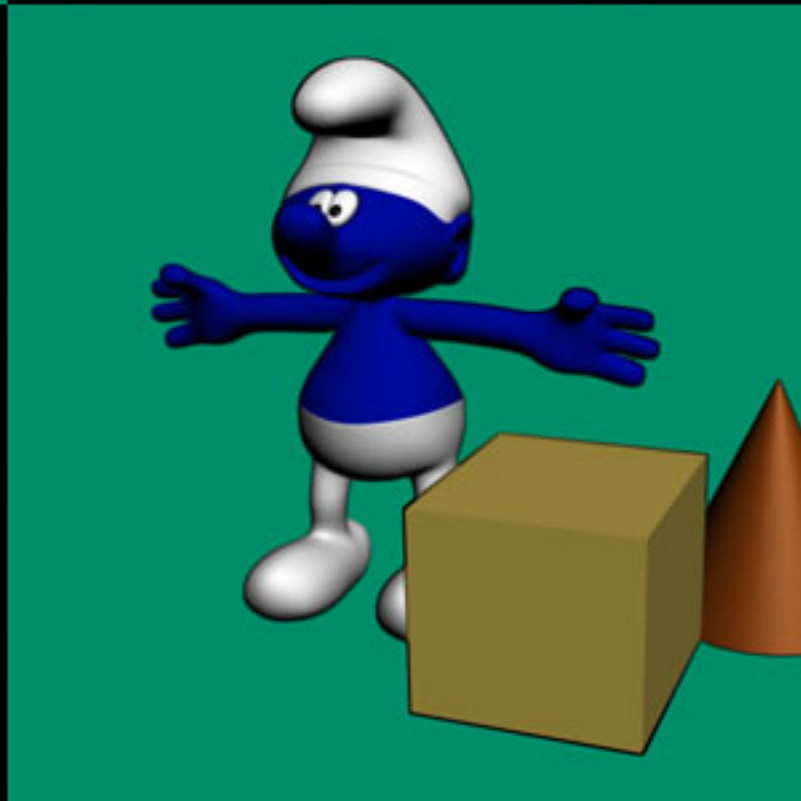
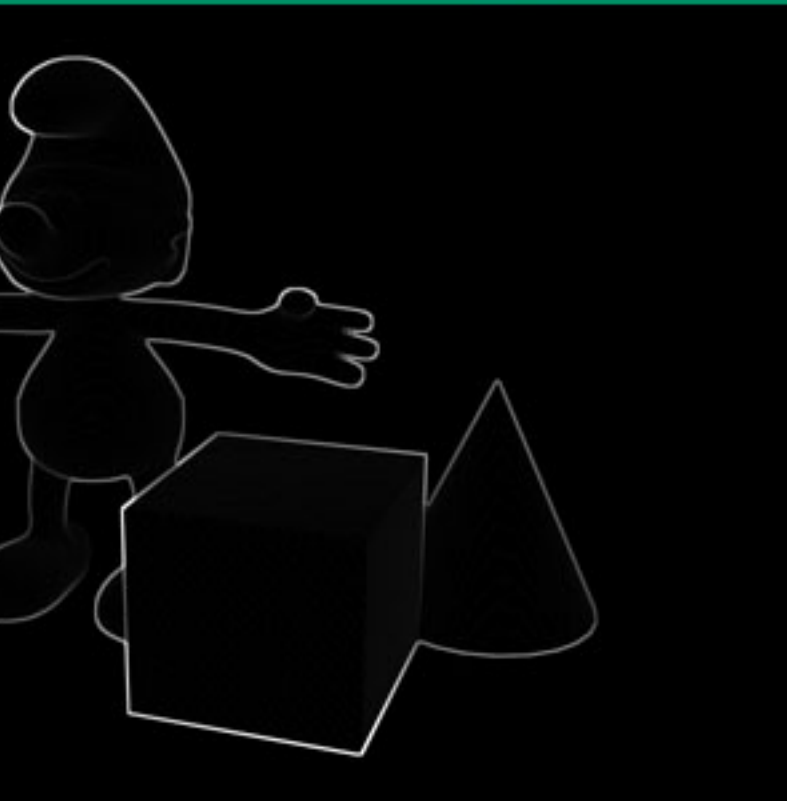
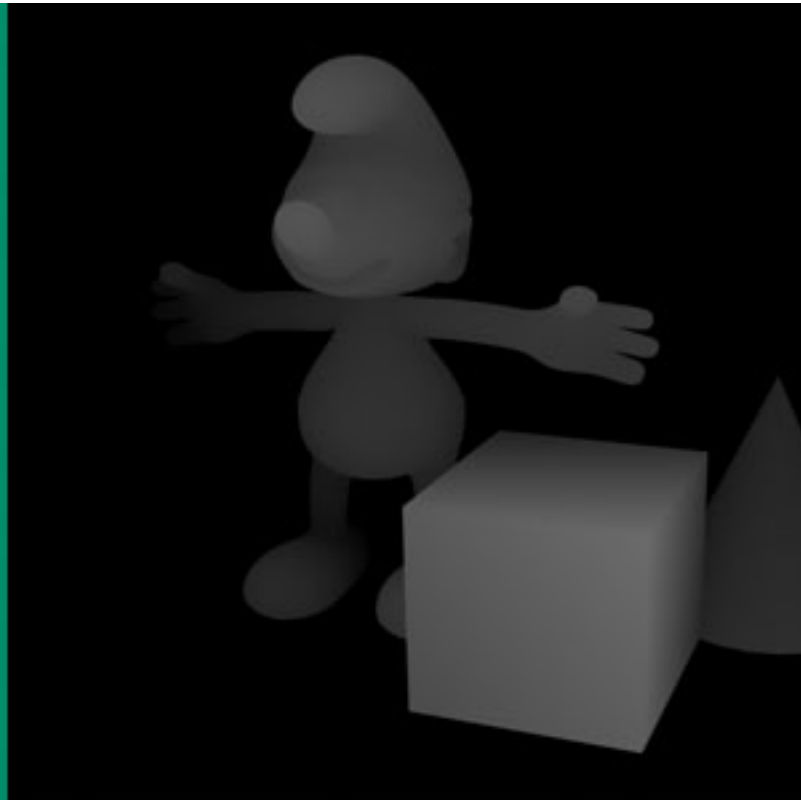
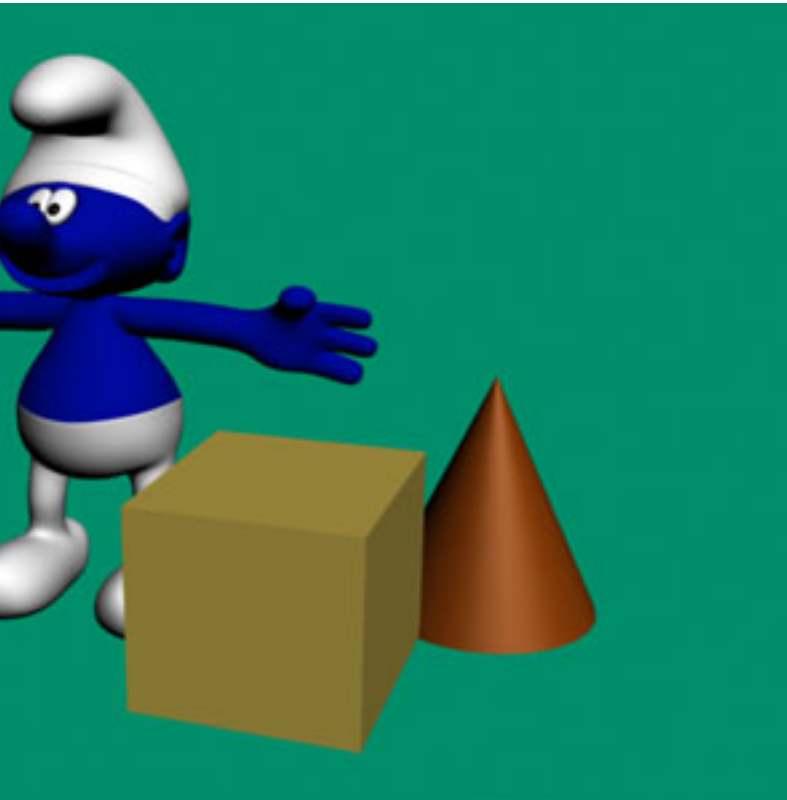


Image initiale / Tampon de profondeur / Détection de contours / Image finale

Cette technique a comme principal intérêt de réduire la quantité de géométrie utilisée pour chaque objet car dans les deux techniques précédentes, soit l'objet était dessiné deux fois, soit un modèle supplémentaire intervenait. Ici, le calcul se fait directement en 2D sur l'image écran de la scène 3D grâce aux informations de profondeur de chaque pixel.

IV - Conclusion

Nous avons pu voir dans ce tutoriel les différentes techniques utilisées pour dessiner des scènes 3D à la façon des bandes dessinées ou des dessins animés. La technique de "cel-shading" n'est donc pas une technique mais un ensemble de deux composantes : les couleurs simplifiées et les contours. Dans vos applications 3D ou vos jeux, il est tout à fait possible d'utiliser l'une sans l'autre en fonction de l'aspect que vous voulez donner à vos modèles 3D. Il est important de faire des compromis, notamment pour les contours, entre la quantité de géométrie, la difficulté de création des modèles et les performances pour trouver la méthode qui vous correspond le mieux. Le "cel-shading" continue son évolution au fil du temps par l'apparition de graphisme plus ou moins subtil comme le dernier Prince of Percia qui utilise des textures assez détaillées et fines ainsi que des contours pour un rendu à mi-chemin entre le cartoon et le réalisme.



Prince of Percia (2008)

Commentez cet article :

V - Remerciements

Le modèle 3D du schtroumpf provient du site 3DValley :  http://www.3dvalley.com/models/models_characters.shtml

Je remercie également **bafman**, **lrmadDen**, **khayyam90**, **loka**, **millie** et **pi-2r** pour leurs avis et conseils et plus particulièrement **Alp** pour ses avis et sa relecture.