Textures volumiques auto-zoomables pour une stylisation temporellement cohérente en temps réel

Pierre Bénard, Adrien Bousseau, Joëlle Thollot

AFIG 2008 / Toulouse
IRIT Presse

Textures volumiques auto-zoomables pour une stylisation temporellement cohérente en temps réel

Pierre Bénard, Adrien Bousseau, Joëlle Thollot

Grenoble Universités, INRIA

Figure 1: *Palette de styles (collage, aquarelle et rendu binaire) temporellement cohérents obtenus avec nos textures volumiques auto-zoomables (colonne de droite).*

Abstract

Les méthodes de stylisation dont l'objectif est de représenter une scène 3D dynamique avec des marques 2D comme des pigments ou des coups de pinceau, sont généralement confrontées au problème de la cohérence temporelle. *Dans cet article, nous présentons une méthode de stylisation en temps réel et temporellement cohérente basée sur des textures : les* ***textures auto-zoomables****. Une texture auto-zoomable est une texture plaquée sur les objets 3D et enrichie d'un nouveau mécanisme de* ***zoom infini****. Ce mécanisme maintient une taille quasi constante en espace image des éléments de texture. Lors de la stylisation, ce mécanisme renforce l'apparence 2D des marques de style tout en restant fidèle au mouvement 3D des objets représentés. Nous illustrons cette propriété par une variété de styles comme l'aquarelle ou le rendu par marques binaires. Bien que notre technique de zoom infini puisse être utilisée aussi bien pour des textures 2D que 3D, nous nous sommes attachés dans cet article au cas 3D (que nous appelons* ***textures volumiques auto-zoomables****), ce qui évite la définition d'une paramétrisation des surfaces 3D. En intégrant notre méthode à un moteur de rendu, nous validons la pertinence de ce compromis entre qualité et rapidité.*

Keywords: informatique graphique, rendu non-photoréaliste, rendu temps réel, textures volumiques

1. Introduction

De plus en plus de jeux vidéo tentent de se démarquer de la traditionnelle quête de l'ultra-réalisme pour des rendus plus stylisés. Plusieurs jeux récents se distinguant par leur originalité graphique (style BD pour *XIII*, Aquarelle/Sumi-e pour *Ōkami* ou prochainement « Illustrative Art Style » pour *Prince of Persia*) ont reçu un très bon accueil de la part des joueurs. Cette nouvelle tendance implique de nouvelles difficultés techniques induites par l'intégration de styles de rendu non-photoréalistes complexes dans les moteurs de jeux.

Dans cet article, nous présentons une méthode de stylisation simple et rapide avec une forte cohérence temporelle. En effet, la principale difficulté, partagée par tous les algorithmes de stylisation, consiste à maintenir la cohérence temporelle de la stylisation lors de l'animation d'une scène 3D. L'objectif est de représenter un mouvement 3D (celui de la caméra ou des objets de la scène) tout en préservant les caractéristiques 2D des marques de style (pigments, coups de pinceaux…) du médium simulé. Pour remplir parfaitement cet objectif sans produire d'artefacts visuels, la stylisation d'une animation doit satisfaire simultanément trois contraintes contradictoires. Premièrement, afin de préserver l'apparence 2D du médium, les marques de style devraient avoir une taille constante et une distribution uniforme dans l'image. Deuxièmement, pour éviter des effets de glissement du style sur la scène (effet *rideau de douche*), les marques de style devraient respecter exactement le mouvement 3D des objets qu'elles recouvrent. Enfin, pour éviter des effets de clignotement ou de variation aléatoire des marques, une continuité temporelle suffisante entre chaque image de l'ani-

mation serait nécessaire. De nombreuses solutions ont été proposées afin de trouver un compromis acceptable à ces contraintes [GG01, SS02]. Elles reposent cependant, pour la plupart, sur des algorithmes coûteux et des structures de données peu compatibles avec les pipelines de rendus existants.

Dans cet article nous présentons les *textures auto-zoomables* qui facilitent l'intégration de stylisation temporellement cohérente dans les pipelines de rendu temps réel. Notre méthode s'adresse plus particulièrement au problème de la stylisation des régions de couleur d'une image, le traitement des contours n'étant pas abordé. Notre approche se base sur des textures comme support de la stylisation, afin d'éviter la mise en place des structures de données complexes. Cela rend notre approche particulièrement adaptée aux médiums présentant une texture caractéristique comme l'aquarelle, le fusain... De plus, la gestion optimisée des textures par les cartes graphiques modernes rend notre méthode particulièrement adaptée au rendu temps réel.

Contrairement au texturage traditionnel, notre approche bénéficie d'un mécanisme de **zoom infini** produisant une distribution uniforme des éléments de texture à l'écran, quelle que soit la distance des objets à la caméra. Ce mécanisme préserve la majorité des caractéristiques 2D du médium traditionnel tout en assurant la cohérence temporelle lors de la navigation dans un univers 3D. Ce mécanisme de zoom infini peut être appliqué aux textures 2D comme 3D. Cependant, nous avons choisi de développer dans cet article le cas des textures 3D (les **textures volumiques auto-zoomables**), ce qui évite le calcul complexe ou la définition manuelle d'une paramétrisation à la surface des objets 3D. Ces textures volumiques peuvent facilement être produites soit procéduralement, soit par synthèse à partir d'exemples 2D.

Afin de démontrer l'efficacité de cette approche, nous proposons son intégration au moteur de rendu OGRE[†]. Les mesures de performance réalisées dans cet environnement indiquent un impact faible de la méthode sur la vitesse d'affichage pour des scènes complexes. Les différents styles présents dans cette implémentation illustrent l'utilité et la simplicité de la méthode pour le rendu temps réel non-photoréaliste.

2. Travaux précédents

Afin de répondre à la contradiction entre le respect du style 2D et celui du mouvement 3D, Meier [Mei96] propose dans son article précurseur de découpler l'apparence des marques de style et leur mouvement. Pour conserver une apparence 2D, les marques de style (dans ce cas des coups de pinceau) sont dessinés avec des imposteurs de taille constante. Pour préserver le mouvement 3D, chaque marque est associée à un point d'ancrage sur l'objet 3D qu'elle stylise. Bien que cette approche ait été améliorée et étendue à de nombreux styles (peinture [Dan99, VBTS07], *stippling* [PFS03], aquarelle [BKTS06a]), la structure de donnée nécessaire à la gestion des points d'ancrage et le surcoût introduit par le rendu individuel de chaque marque rendent son utilisation dans des moteurs de jeux vidéo peu praticable.

Ces limitations, dues à la gestion individuelle des marques de style, peuvent être évitées en traitant les marques globalement, sous forme de textures. La méthode de *Dynamic Canvas* [CTP*03] consiste à appliquer une texture de papier sur l'écran afin de styliser en temps réel un environnement 3D dans un contexte de navigation interactive. Le cœur de cette approche consiste à contrebalancer les rotations 3D de la caméra par des transformations 2D de la texture. Les translations en profondeur de la caméra sont elles compensées par un mécanisme de zoom infini qui maintient une taille quasi constante de la texture à l'écran. Bien qu'offrant un très bon compromis entre l'apparence 2D du papier et les mouvements 3D de la caméra, cette approche est limitée à la navigation dans des scènes statiques. Coconu et coll. [CDH06] et Breslav et coll. [BSM*07] adoptent une approche similaire en appliquant à une texture de stylisation la transformation 2D approximant au plus proche la transformation 3D de l'objet à représenter. Tout comme *Dynamic Canvas*, ces méthodes offrent une très bonne préservation de l'apparence 2D des marques de style, la texture n'étant déformée qu'en espace image. En contrepartie, l'approximation introduite par l'application d'une transformation 2D à la place d'une transformation 3D peut produire des effets de glissement pour des mouvements 3D extrêmes. Une autre approche du zoom infini a été proposée récemment par Han et coll. [HRRG08]. Ils développent un algorithme de synthèse de texture 2D multi-échelle qui génère des éléments de textures durant le zoom. Bien que l'illusion produite par cette dernière méthode soit plus précise pour certaines textures, elle semble trop lente pour le rendu temps réel de scènes complexes. Dans cet article, nous étendons le mécanisme de zoom infini de *Dynamic Canvas* aux textures en espace objet qui permettent la stylisation d'objets dynamiques en temps réel sans aucun glissement.

Ces différentes approches à base de textures sacrifient le respect du mouvement 3D au profit de l'apparence 2D, mais le compromis inverse est également possible. C'est le choix fait par les méthodes d'*art map* [KLK*00] et *tonal art map* [PHWF01] qui plaquent directement les textures de stylisation sur les objets 3D de la scène. Si elles sont plaquées de façon naïve, les marques de style restent certes parfaitement attachées aux objets mais sont fortement déformées par la projection perspective. La solution des *art maps* consiste à utiliser le mécanisme du *mip-mapping* pour adapter l'échelle de la texture en fonction de la distance à l'écran. Cette approche permet de maintenir une taille de marques presque constante dans l'image, tout en tirant profit de la gestion optimisée des *mipmaps* par les cartes graphiques. Freudenberg et coll. [FMS01] ont prouvé la performance de ces méthodes

[†] http://www.ogre3D.org

en intégrant un rendu non-photoréaliste de type *art map* dans le moteur de jeux *Fly3D*.

Ces approches souffrent cependant de quelques limitations. Tout d'abord, afin d'obtenir des transitions cohérentes entres les différents niveaux d'une *art map*, chaque niveau doit contenir les marques du niveau précédent [PHWF01]. La prise en compte de cette contrainte limite l'application des *art maps* aux styles binaires de type *hatching* ou *stippling*. Ensuite, l'utilisation de *mipmaps* n'est efficace que lorsque l'objet s'éloigne de l'écran (rétrécissement de la texture). Lorsque l'objet s'approche trop près de la caméra, la texture subit un simple agrandissement. Les *mipmaps* ne permettent pas non plus la correction des déformations perspectives dues à l'orientation de la surface par rapport à l'écran. Les *ripmaps* [KLK*00] permettent de corriger une partie de ces déformations mais ne sont pas aussi bien supportées par les cartes graphiques modernes. Enfin, le plaquage des textures nécessite la définition d'une paramétrisation à la surface des objets. Praun et coll. [PHWF01] proposent d'automatiser cette paramétrisation grâce aux *lapped textures*, mais ceci nécessite la mise en place d'une structure de données supplémentaire.

Tout comme les *art maps*, la méthode des textures autozoomables décrite dans cet article corrige uniquement les déformations dues à la distance de l'objet à l'écran. Cependant, contrairement au *mipmapping*, le mécanisme de zoom infini permet l'obtention de marques de taille constante pour n'importe quelle distance. De plus, en combinant plusieurs échelles de textures à la fois, le zoom infini assure des transitions douces quelle que soit la texture utilisée. Enfin, notre méthode étant basée sur des textures volumiques, elle ne nécessite aucune définition de paramétrisation supplémentaire des surfaces 3D, qui est souvent complexe pour limiter les contractions/dilatations ou, dans le cas d'atlas de textures, pour gérer les raccordements sans discontinuité.

3. Textures volumiques auto-zoomables

Dans cet article, nous désignons par *zoom infini* sur un signal la croissance infinie de sa fréquence perceptible. Dans le cas d'un son, cela correspond à une montée dans les aigus audibles à l'infini, comme l'a montré Shepard [She64]. Dans le cas d'une image 2D, Cunzi et coll. [CTP*03] et Han et coll. [HRRG08] considèrent que ce processus correspond à l'apparition infinie de nouveaux détails visibles. Ils ont cependant développé des solutions diamétralemnt opposées, comme nous l'avons décrit dans la section précédente.

Nous proposons d'étendre le mécanisme de zoom infini de Cunzi et coll. aux textures en espace objet (section 3.1). Notre implémentation (section 3.3) montre que cette approche, très peu coûteuse en temps de calcul et relativement peu en place mémoire, est parfaitement intégrable à un moteur de rendu de jeux vidéo comme OGRE. Les résultats obtenus (section 3.4) illustrent les avantages de cette méthode temps réel pour des scènes dynamiques profondes, sans dégradation importante de l'apparence 2D du médium.

3.1. Zoom infini en espace objet

L'objectif contradictoire du mécanisme de zoom infini est de maintenir une taille globalement constante de la texture volumique à l'écran, tout en préservant l'impression de grossissement / rétrécissement des éléments de texture lors du déplacement en profondeur de la caméra.

La méthode de *Dynamic Canvas* [CTP*03] utilise un mécanisme de zoom infini 2D basé sur la *fractalisation* d'une texture plaquée sur l'écran, de la même façon que Perlin [Per85] pour une fonction procédurale. Dans cette approche, n octaves (fréquence doublée) du motif original sont mélangées de façon linéaire afin de créer une image autosimilaire. Quand l'observateur se déplace en profondeur, la fréquence des octaves est décalée de façon continue pour produire une illusion de zoom. Ce mécanisme procure l'illusion d'une texture grossissant à l'infini tout en conservant une taille quasi constante à l'écran. Cependant, le calcul du facteur de zoom dépend d'une distance subjective entre un plan virtuel et la caméra. Cette distance étant fixée avant la navigation, elle ne tient pas compte de la profondeur réelle de la scène. Par conséquent, il peut se produire des glissements de la texture sur les objets qui ne sont pas exactement à cette distance. Nous proposons dans cet article l'extension de ce principe de fractalisation aux textures en espace objet, en nous affranchissant de l'approximation de la distance subjective.

Suivant cette approche, une **texture volumique autozoomable** est la somme pondérée de n octaves Ω_i du motif volumique d'origine. Notez que le mélange de plusieurs octaves assure la continuité temporelle du zoom au prix d'une perte de contraste, comme discuté en section 5. Nous avons défini empiriquement qu'il fallait au moins $n = 4$ octaves pour tromper la perception humaine, tandis qu'utiliser un nombre supérieur d'octaves n'a pas d'impact réel sur l'impression de continuité mais dégrade sensiblement la texture. Chaque objet 3D est ensuite plongé dans une texture volumique autozoomable. Pour préserver un sentiment convainquant de zoom, chaque octave est sujette aux règles de la projection perspective : une octave apparaitra deux fois plus grande quand elle sera deux fois plus proche de la caméra. Il faut néanmoins s'assurer de maintenir une taille quasi constante en espace image. Pour cela, nous introduisons la notion de **cycle de zoom** qui a lieu chaque fois que la taille de l'octave a doublé. Chaque octave est alors remplacée par l'octave suivante dans le cycle et une nouvelle octave de fréquence supérieure est créée, comme illustré sur la figure 2.

3.2. Algorithme proposé

En pratique, un objet est plongé dans une texture volumique auto-zoomable en dérivant ses coordonnées de texture