Les effets spéciaux dans les jeux vidéo

Thibaut Castanié
Vincent Bazia
M2 IMAGINA

27 novembre 2015

Table des matières

Introduction

Les jeu-vidéos ont pour ambition d'être le plus proche possible de la réalité visuelle. Pour cela ils exploitent de mieux en mieux la puissance grandissante des machines, afin de posséder des modèles de plus en plus détaillés, des textures avec un plus grand niveau de détails, mais aussi des effets visuels de plus en plus proches de la réalité. La majorité des effets sont calculés "post-traitement", c'est à dire après que le rendu de la scène globale soit effectué. Ils se doivent donc d'être peu coûteux en puissance de calcul afin de ne pas dégrader l'expérience du joueur.

1 Depth of field

1.1 Présentation

La profonfeur de champ est l'effet qui va faire que les objets qui se trouvent à une certaine distance apparaissent nets, tandis que les objets qui sont plus près ou plus loin apparaissent flous.

La profondeur de champ est couramment utilisée dans la photographie et la cinématographie afin de diriger l'attention du spectateur ainsi que pour faciliter le discernement des profondeur dans une scène.

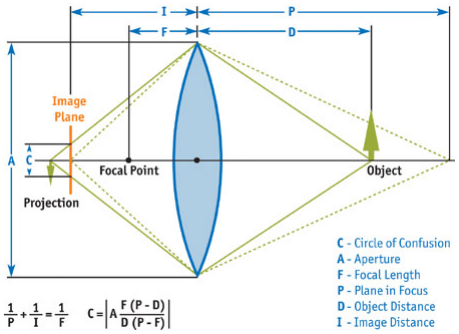
De base, cet effet résulte de la nature des lentilles. Au niveau de notre oeil, par exemple, seuls les rayons lumineux d'objets à une certaine distance viendront converger en un point. Nous verrons ces objets nets, tandis le reste sera plus ou moins flou. Dans le cas des mondes 3D, la profondeur de champ naturelle n'existe pas. Pour obtenir cet effet, nous devons le simuler afin d'approximer le comportement d'une lentille.

On dénombre cinq techniques principales dans la gestion de la profondeur de champ. Seulement pour chacune, il faut faire un choix entre la qualité et la vitesse d'exécution.

Nous allons nous intéresser à la méthode Reverse-Mapped Z-Buffer. Cette technique rapide, est la plus susceptible d'être utile dans le développement d'applications temps réel.

1.2 Reverse-Mapped Z-Buffer

La première étape consiste à déterminer une lentille virtuelle.



Le Cercle de Confusion (c) est la zone où les objets seront nets. Image Plane est la rétine de notre lentille virtuelle et la focale détermine la distance à laquelle les objets seront nets. Grâce à ces différents paramètres, nous pouvons déterminer les objets qui seront nets de la scène.

La première étape de cette méthode consiste à afficher les différences de profondeur Z entre les objets et le cercle de confusion de la scène en couleur.

Afin de rendre flou les éléments hors du cercle de confusion, la méthode la plus répandue est de générer des mipmaps de la texture de notre scène afin d'obtenir différentes qualités de cette texture.

Enfin, on applique aux objets de notre scène les textures des plus ou moins bonne qualité en fonction de la différence de profondeur Z.

Cette technique est la plus rapide. C'est le choix priviliégié dans le développement d'applications qui nécessitent un rendu rapide en temps réel.

2 Glows (blooming)

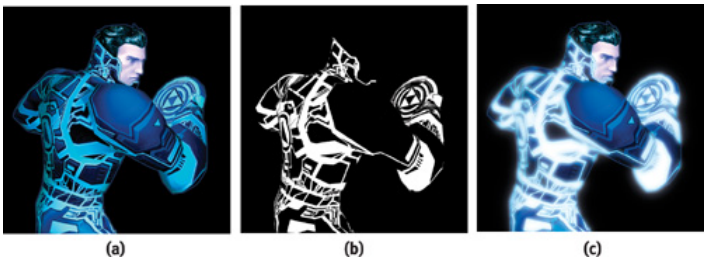
2.1 Présentation

L'effet de *blooming* (lueur en français), est un phénomène permettant d'augmenter grandement le réalisme d'une image, pour un coût en performance assez

faible. Dans le monde réel, cet effet est visible par l'oeil humain quand il regarde des objets très brillants situés dans un environnement sombre. En effet, la transition du lumineux au sombre étant brutale, elle devient floue, la lumière débordant dans la zone obscure. Un objet possédant une brillance trop élevée aura d'autres effets sur la vision, tel l'apparition de stries ou d'effets de lentille, ce qui n'est pas pris en compte dans la génération classique d'un effet de *blooming*. Le flou lumineux est souvent utilisé de façon non-naturelle, dans le jeu-vidéo et les images de synthèse, afin de donner un aspect futuristique à une scène, donner de l'importance à un personnage ou bien accentuer au maximum l'effet aveuglant une lumière.

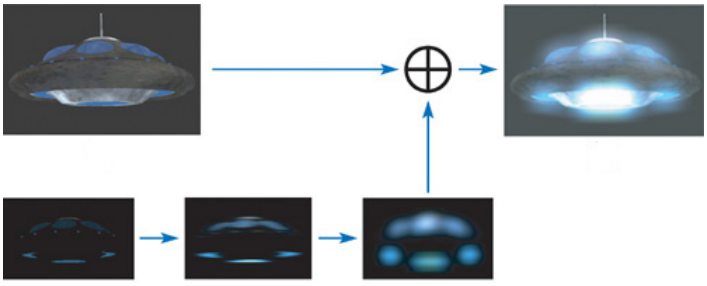
2.2 Mise en place

Lorsque l'effet doit être appliqué sur des objets simples, la méthode la plus efficace est d'appliquer une texture de type *"glow"* sur l'objet. Ainsi la texture le suivra exactement pendant son animation. Un effet de lueur "fixe" est alors appliqué en post-traitement sur l'ensemble de la texture *"glow"*. Cette technique a l'avantage de pouvoir désactiver l'effet à tout moment, et aussi de le modifier à n'importe quelle étape du développement.



Ici, l'image **(a)** possède la *glow-texture* de l'image **(b)**, sans que l'effet soit activé. L'image **(c)** possède l'effet de *blooming* activé.

Lorsque la scène possède des sources de lueurs différentes ou que celles-ci possèdent des formes complexes, donc des variations dans la puissance de l'effet, la texture *"glow"* peut être couplée à d'autres textures telles que l'*alpha*. Ceci permet de séparer l'effet du reste de l'image. On peut ainsi appliquer des effets de flou afin d'accentuer le *glowing* de l'effet.



Les textures peuvent être modifiées très facilement sur une durée afin d'animer les effets de *glow*, pour un coût de calcul faible.

3 God rays

3.1 Présentation

God rays, ou Rayon crépusculaire, est une expression pour décrire les rayons de lumière qui proviennent du soleil quand il est, en partie, couvert. Ils apparaissent sous formes de raies lumineuses entrecoupés de zones ombragées. En réalité ces rayons sont parallèles, alors que la perspective nous les fait voir orientés. Les particules en suspension dans l'air facilitent leur visibilité.

God Rays dans le monde réel.

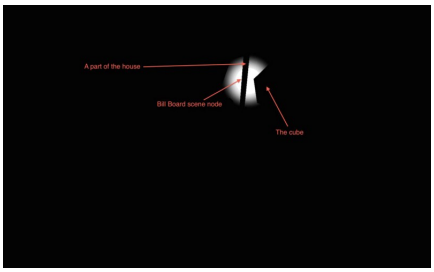
Dans le développement graphique 3D, on les appelle aussi *"Volumetric Light Scattering"*. Leur implémentation va permettre de rendre nos mondes plus réalistes. De plus, cet effet peut être utilisé dans le but de changer l'ambiance d'une scène.

3.2 Mise en place

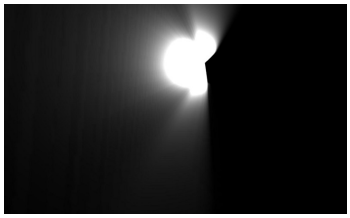
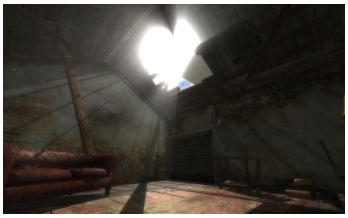
La scène sans God Rays.

La méthode consiste dans un premier temps à définir ce que l'on veut afficher normalement et ce que l'on veut définir comme obstacle aux rayons de lumière. Nous allons afficher ces obstacles en noir.

Pour cela, on sauvegarde dans un premier temps les noeuds de notre scène. Puis on rend tous nos noeuds d'obstacle dans le standard SOLID Material avec une émission de couleur noir.On rend la scène puis on réinitialise les noeuds à la configuration précédente.



Ensuite, on applique un shader qui va se charger de disperser cette lumière au sein de notre rendu. Grâce à la ré-initialisation de nos noeud, on remet ensuite en place les couleurs de bases.

Enfin, pour un rendu plus réaliste, on va adoucir les rayons en faisant un léger dégradé avec les couleurs de base des obstacles.

On obtient ainsi un bel effet de God Rays qui change totalement l'ambiance de notre pièce.

4 Motion Blur

4.1 Présentation

Cet effet est observable dans le monde réel, lorsqu'on regarde le paysage défiler à travers la vitre d'une voiture : si on n'observe aucun point du paysage en particulier, notre cerveau n'est pas capable d'analyser les images qui défilent rapidement et produit une sorte de brouillard visuel.

Le flou de mouvement est le meilleur moyen de simuler de la vitesse dans un jeu vidéo. Ainsi, dans les jeux de courses, c'est l'effet le plus important car il augmente grandement le réalisme et la sensation de vitesse. Il permet aussi d'adoucir l'apparence d'un jeu dont le rendu est de 30 images par secondes, ou moins. Cette façon d'ajouter du flou exige néanmoins une importante puissance de calcul.

Une scène sans, et avec flou cinétique

4.2 Fonctionnement

Il y a deux façons principales pour ajouter du flou cinétique dans un jeu vidéo.

La première est d'utiliser les mouvements de la caméra, afin de définir sa vitesse et sa direction, pour créer un flou cinétique et un flou radial. Cette méthode modifie ainsi le comportement entier de l'écran, mais a le mérite d'être moins chère en puissance de calcul.

La seconde consiste à utiliser un *shader* pour créer un tampon de vitesse, afin de marquer l'intensité du mouvement de l'effet. Cette méthode doit être appliquée sur chaque objet sur lequel on désire appliquer du flou, elle coûte donc plus cher en puissance de calcul que la précédente.

Ces deux méthodes sont coûteuses et dépendent grandement des limites de la plateforme sur laquelle le jeu s'exécute. En effet, la scène nécessite d'être rendue dans un processus différent afin de pouvoir générer le tampon de vitesse pour chaque pixel. Cela est problématique quand l'application n'autorise pas le passage de la scène dans le canal de rendu plus d'une fois par boucle de rendu.

Conclusion

Les effets appliqués dans les jeu-vidéos permettent d'améliorer grandement le rendu d'une scène en utilisant le moins de puissance de calcul possible. La plupart d'entre eux utilisent donc des astuces afin d'obtenir un effet qui puisse "tromper" le joueur en le faisant croire qu'il est naturel. En effet, à l'heure actuelle, un jeu ne peut se permettre de calculer en direct les multiples rebonds de la lumière sur des surfaces, ou bien ses interactions sur l'oeil humain. L'emploi de techniques rapides à calculer est donc nécessaire afin d'assurer une expérience de jeu fluide.

Bibliographie

- *n*Vidia, Randima Fernando, *GPU Gems 1*, Addison-Wesley Professional, 2004
- *n*Vidia, Hubert Nguyen, *GPU Gems 3*, Addison-Wesley Professional, 2007
- Julien Moreau-Mathis, *Gods Rays? What's that?*, `medium.com`, 2014
- Unreal Engine 4 documention, `https://docs.unrealengine.com/`