INF8702

INFOGRAPHIE AVANCEE

REFLEXION DYNAMIQUES AVEC CUBEMAPS EN UTILISANT LE NUANCEUR DE GEOMETRIE

Yves Israel Ngudie

Automne 2016

I. Introduction

En infographie ou dans les applications de rendu 3D, la réflexion rajoute un certain niveau de réalisme a la scène. Nous avons vu en classe la méthode du lancer de rayons qui applique une illumination globale à notre scène et permet de calculer les réflexions statiques et dynamiques, venant de plusieurs sources de lumières. Le problème de cette technique est qu'elle est trop couteuse pour être utilisée en temps réel. Nous avons aussi vu l'illumination basée sur l'environnement (Image Based Lighting), qui offre une méthode moins couteuse d'implémenter des réflexions statiques en temps réel.

Ce projet explore l'implémentation de la réflexion dynamique à l'aide des cubes maps ou mappage d'environnement. Premièrement, nous parlerons de la motivation et de notre intérêt pour cette technique. Ensuite, nous expliquerons notre implémentation utilisant le nuanceur de géométrie. Enfin, Nous poursuivrons en exposant les difficultés rencontrées et/ou améliorations possibles.

II. Motivation

Nous avons implémenté une réflexion statique d'un environnement dans notre dernier travail pratique. Mais il y a deux problèmes principaux qui ont motivé ce projet. Le premier est que l'environnement réfléchi est infini et non local, c'est-à-dire les objets voisins ne sont pas réfléchis. Le second est que les réflexions sont statiques. Ainsi, nous avons souhaité étudier comment utiliser cette technique pour avoir des réflexions locales et dynamiques. La solution aux deux problèmes est de générer nos textures d'environnement ou cube maps de façon dynamique pendant le rendu. Ainsi, cette solution nous permet d'avoir toutes les réflexions et de les rendre dynamiques, puisqu'on redessine chaque le cube map.

De nos jours, les cubes maps ou textures d'environnement sont très utilises dans les moteurs de rendu 3D pour les jeux vidéo ou autres applications. Ils permettent d'obtenir des résultats efficaces, réalistes et simples à implémenter. Nous avons ainsi choisi ce projet pour un peu plus comprendre et aussi maitriser cette technique.

III. Implémentation

1ere phase : Génération des cubes maps

Comme dit plus-haut, l'idée de la solution est de dessiner des cubes maps de façons dynamiques. Un cube map ayant 6 textures, une pour chaque face, il nous faudra dessiner 6 fois notre scène, en changeant chaque fois le point de vue de la camera, pour regarder dans la direction de la face qu'on doit dessiner. De ce fait nous aurons pour un objet donné, un cube map représentant son environnement. Ensuite, nous utiliserons cette texture pour colorier notre objet en utilisant le rayon réfléchi entre la normale et l'observateur pour trouver le texel correspondant.

Par contre, au lieu de dessiner 6 fois notre scène, nous allons utiliser une configuration du nuanceur de géométrie qui nous permettra de le faire en 1 seule fois (single pass). Pour ce faire, il faudra fournir au nuanceur de géométrie les 6 matrices de vue-projections (precalculés par l'application). Le nuanceur de géométrie va ainsi exécuter 6 instances par sommet. Pour chaque instance correspondant à une face du cube, nous pourrons ainsi avoir le rendu de la scène de ce point de vue donné. Notons que pour cette phase, le pipeline graphique doit être configure pour dessiner sur nos textures et non sur l'écran

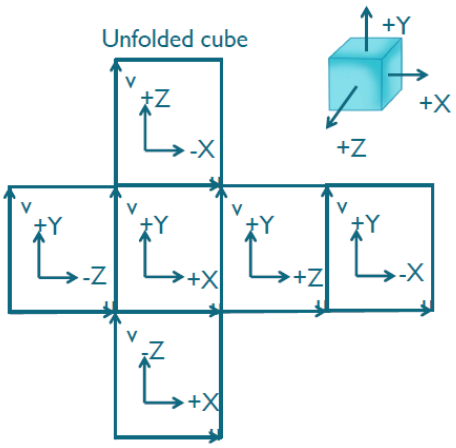

Figure 1. Cube map

Source : http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/PDFs/8-ReflectionsBasedOnLocalCubemaps.pdf

2eme phase : Calcul de la réflexion et correction locale

La deuxième phase consiste à appliquer la bonne couleur dans le nuanceur de fragment. Pour cela il suffit de calculer le vecteur réfléchi entre l'observateur et la normal et d'utiliser ce vecteur

trouvé pour accéder à notre texture cube. Pour un cube map local, c'est à dire utilisé pour des réflexions dynamiques, il est préférable d'appliquer une correction locale à ce vecteur afin d'éviter des problèmes de fausses réflexions.

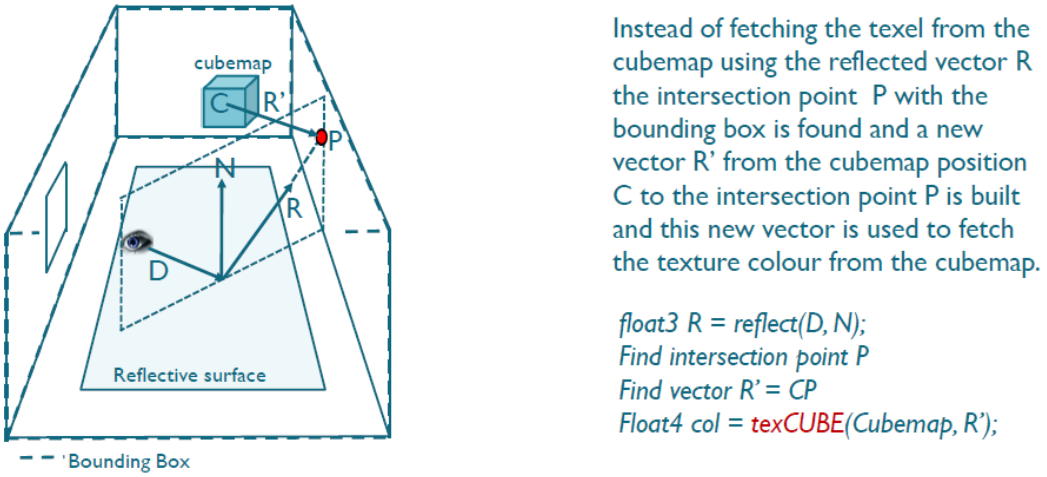

Figure 2. Calcul du vecteur réfléchi corrigé

Source : http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/PDFs/8-ReflectionsBasedOnLocalCubemaps.pdf

3e phase : Mélange des couleurs

Réflexion dynamique

Maintenant qu'on a la bonne couleur, on peut l'appliquer à notre fragment en tenant compte de la nature du matériau de l'objet. Pour un objet miroir, il nous suffira d'appliquer la couleur. Pour objet partiellement réfléchissant, il sera utile de fournir une variable indiquant la quantité de réflexion possible.

Réflexion statique

Pour la réflexion statique, il n'est pas besoin d'appliquer une correction locale. Le vecteur réfléchi suffit pour obtenir le bon texel. Nous devrons ensuite combiner cette couleur avec celle obtenue du cube map local. Pour ainsi compléter le processus.

IV. Difficultés

Lors du mélange des couleurs des cubes maps dynamiques et statiques, nous serons confrontés à décider quand est ce que la couleur du cube map local doit obstruer celle du cube map statique. Autrement, les réflexions auront l'air d'être transparentes.

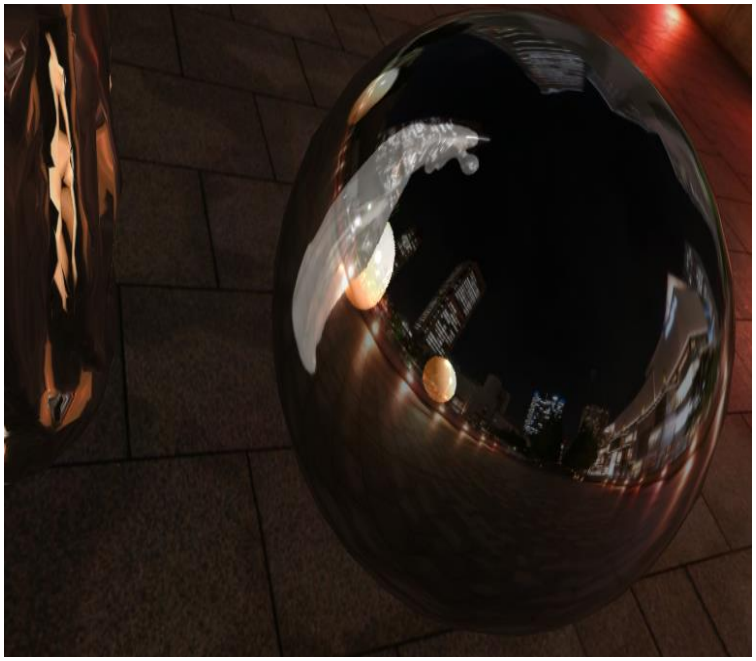

Figure 3. Problème de mélange des couleurs.

Dans la figure 3, nous pouvons voir les lumières de l'immeuble à travers la venus. Il faudrait plutôt que la statue obstrue totalement les pixels de la skybox (environnement).

La figure 4 montre également le même problème plus accentue. En effet, dans cette expérience, nous utilisons les texels de la skybox pour influencer l'illumination de nos sphères.

Figure 4. Problème de mélange de couleurs. Les couleurs des cubes maps sont additionnés

V. Améliorations possibles

Interaction des lumières

La figure ci-dessous montre l'utilisation des cubes maps pour influencer la composante spéculaire dans l'illumination de nos objets. Cela donne un effet d'interaction des lumières réfléchies par les sphères dorées (qui sont totalement réfléchissante) et qui éclairent le modèle venus. Pour les sphères, nous calculons aussi la sécularité en prenant un pixel dans la direction de l'observateur et le vecteur réfléchi. Cela nous permet de simuler les lumières de l'environnement.

Toutes ces techniques sont des implémentations naïves mais qui peuvent être améliorées en faisant par exemple un prétraitement de notre texture d'environnement pour repérer les texels qui pourraient être considérés comme source lumineuse. Cela nous permettrait même d'ajouter de l'ombre à notre scène.

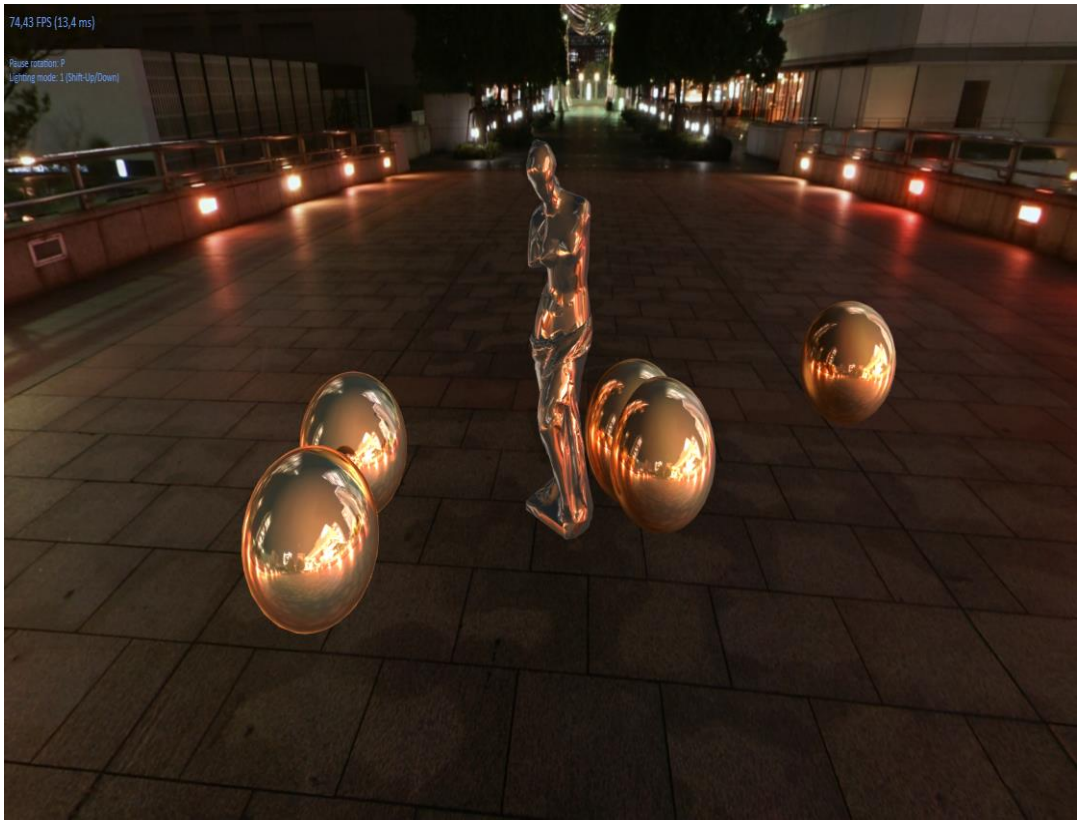

Figure 5. Ajout de la sécularité pour avoir des sphères brillantes

VI. Conclusion

Dans ce projet, il a été question de présenter la technique de réflexion dynamique, utilisant le nuanceur de géométrie pour optimiser la phase de génération des textures. Nous avons premièrement motivé l'utilisation de cette technique, et avons expliqué globalement son implémentation théorique. Ensuite, nous avons présenté quelques difficultés et améliorations possibles.

Bien que n'ayant pas eu le temps d'explorer en détails les implémentations utilisées dans l'industrie pour cette technique, nous avons pu pousser un peu plus loin notre compréhension sur l'utilisation des mappages d'environnement ou de l'illumination base sur les textures (Image based lighting). Nous avons aussi eu l'occasion de programmer des nuanceurs en HLSL (DirectX) et spécialement le nuanceur de géométrie. Nous pourrons ainsi facilement appliquer les mêmes concepts en GLSL (OpenGL)

Application

Le code source est fourni. Il est code en C# et utilise le wrapper SharpDX pour DirectX11. Le code de l'application est principalement base du livre de référence de Justin Stenning : Direct3DRendering Cookbook. Le code du projet est aussi disponible sur GitHub : https://github.com/yvesis/INF8702

Instructions

La touche L : Réflexions statique et dynamiques normales avec des sphères ayant des coefficients de réflexion aléatoires

La touche M : Illumination phong en utilisant les texels des cubes maps comme sources de lumières

La touche C : Ajout de sécularité pour simuler des sphères éclairant leur entourage

Références

Livre de référence sur lequel est base notre code

- Stenning, J.2014. Direct3DRendering Cookbook. Packt Publishing

Lectures et articles sur le sujet :

- GPU Gems. Chapter 19. Image-Based Lighting. Kevin Bjork, 2004. Tiré de http://http.developer.nvidia.com/GPUGems/gpugems_ch19.html
- Lagarde, Sebastien. nd. **Image based lighting and parralax corrected cubemap.** *Tiré de https://seblagarde.wordpress.com/2012/09/29/image-based-lighting-approaches-and-parallax-corrected-cubemap/*
- https://community.arm.com/groups/arm-mali-graphics/blog/2015/04/13/dynamic-soft-shadows-based-on-local-cubemap

Blog sur l'ombrage basé sur le mappage d'environnement :

- ARM,nd. **Dynamic Soft Shadows Based on Local Cubemap**. Tire de https://community.arm.com/groups/arm-mali-graphics/blog/2015/04/13/dynamic-soft-shadows-based-on-local-cubemap