UNIVERSITÉ LIBRE DE BRUXELLES
Faculté des Sciences
Département d'Informatique

Les ombres au sein des jeux et des animations

Bruno Rocha Pereira
Pierre Gérard
Quentin Ravau
Antoine Carpentier

Superviseurs : Tom Lenaerts et Jean-Sébastien Lerat

Année académique 2014 - 2015

Contents

Chapter 1

Introduction

Une ombre est une "zone sombre résultant de l'interception de la lumière ou de l'absence de lumière"[1]. C'est un élément indispensable au réalisme d'une scène d'animation ou de jeu vidéo. En effet ce sont les ombres qui vont apporter l'information quand à la position relative et à la taille des objets qui créent l'ombre. Dans le cas d'objets complexes, elles permettent d'obtenir des informations sur la forme des objets. Dans le monde réel, on est souvent confronté à plusieurs sources lumineuses qui apportent chacune leur lot d'informations en plus.

Nous allons, lors de ce projet d'année, étudier l'impact des ombres sur le réalisme des animations et des jeux vidéos. Cette étude sera réalisée en utilisant et comparant différents algorithmes de génération d'ombre, plus ou moins réalistes.

Nous nous intéresserons plus particulièrement aux algorithmes en temps réel car ceux-ci sont plus intéressant et plus attractifs pour une présentation pour le Printemps des Sciences[2].

La littérature scientifique distingue deux types d'ombres : les *soft shadows* et les *hard shadows*. Les premières ont des bords diffus et les secondes n'en ont pas. Plus la source de lumière est proche d'un objet, plus les bords d'une ombre réaliste sont diffus et inversément. Dans ce projet, nous nous intéresserons uniquement aux algorithmes de génération des *soft shadows* car ils permettent également de générer des *hard shadows*.

Nous testerons les différents aspects et effets des ombres dans différents scénarios que nous metterons en application. Nous ferons varier les objets, les sources lumineuses, leur nombre et leur mouvement afin de présenter des situations se rapprochant de la réalité.

[1] http://www.larousse.fr/dictionnaires/francais/ombre/55933
[2] http://www.printempsdessciences.be

Chapter 2

Présentation des articles

Shadow Algorithms Data Miner

[Andrew Woo, 2012]

Ce livre se veut la référence pour les différents types d'ombres et d'algorithmes existants dans la littérature spécialisée. Il explique les *hard shadows* et les *soft shadows*, ainsi que les principaux types d'algorithmes utilisés comme le *shadow mapping*, le *shadow volume* et le *ray tracing*. De plus, l'article va aussi plus en profondeur en expliquant les problèmes que l'on pourrait rencontrer, notamment dus au *self shadowing* ou au *bump mapping*.

Ainsi, même si tous les chapitres ne nous intéresseront peut-être pas, cet article répondra certainement à beaucoup de nos questions et nous a déjà permis d'orienter nos recherches dans une certaine direction.

A survey of Real-Time Soft Shadows Algorithms

[Hasenfratz et al., 2003]

Cet article décrit le principe des ombres et discute leur importance. Il nous explique que les ombres influencent notre perception de la position relative et la taille des objets.

Nous avons décidé de retenir cet article car il décrit les différences existantes entre les *soft shadows* et les *hard shadows*. Il décrit également deux types d'algorithmes (le *shadow mapping algorithm* et le *shadow volume algorithm*) de manière claire et précise, en spécifiant les points faibles et les points forts de chacun. Cet article présente également des améliorations possibles aux différents algorithmes, qui permettent de réduire le coûts des algorithmes et donc se rapprocher encore plus du temps réel.

Cet article nous aidera donc beaucoup pour les algorithmes de *shadow mapping* et de *shadow volume*.

A Survey of Shadow Algorithms

[Andrew Woo, 1990]

Bien que cet article date un peu, il nous sera utile car il caractérise les différents types d'ombres. Il décrit également les algorithmes de génération d'ombres à la base des techniques actuelles et discute de leurs complexités, leurs avantages et leurs inconvénients. Il examine les *hard shadows*, *soft shadows* ainsi que les ombres d'objets transparents. Cet article devrait donc nous indiquer quel algorithme serait le mieux adapté à différentes situations que nous pourrions rencontrer.

Algorithms for Dynamic Shadows

[Savchenko, 2005]

Cet article part d'une base très simple, c'est à dire simplement dessiner un rond en dessous des objets qui bougent vers des exemples d'algorithmes plus complexes et réalistes. De part sa manière progressive d'expliquer la problématique et les solutions relativement simples de la représentation des ombres, cet article nous aidera à mieux comprendre les différentes difficultés qui nous attendent et à aborder les algorithmes de manière didactique.

GEARS: A General and Efficient Algorithm for Rendering Shadows

[Lili Wang and Popescu, 2014]

Cet article décrit un algorithme de *shadow mapping* qui permet d'obtenir une scène interactive tout en gardant une ombre correcte et précise. De plus, il offre un niveau de *FPS* tout à fait correct. Il est donc tout à fait compatible avec notre optique de nous orienter vers des algorithmes en temps réel. Cet algorithme est, selon ses auteurs, le plus performant parmi les algorithmes actuels de *soft shadows* en temps réel.

Cet article nous conforte dans l'idée de nous diriger vers les algorithmes de représentation de l'ombre en temps réel et d'éviter l'algorithme *ray tracing* qui fournit moins de *FPS* et moins de précision que les algorithmes de *shadow mapping* et *shadow volume*.

A Survey of Shadow Volume Algorithms in Computer Graphics

[Kolivand and Sunar, 2013]

Cet article présente les différentes recherches publiées sur le sujet des algorithmes de *shadow volume*. Pour chaque publication un avis critique est donné. Cela nous sera utile pour, par exemple, savoir quel papier n'est plus d'actualité. L'auteur passe ensuite en revue les différents algorithmes disponibles et précise les améliorations que les chercheurs y ont apportées, avec les avantages et les inconvénients que chacunes de ces améliorations peut apporter.

Il nous sera utile pour implémenter et optimiser nos algorithmes de *shadow volume*.

An Improved Physically-Based Soft Shadow Volume Algorithm

[Jaakko Lehtinen, 2006]

Cet articles identifie et analyse plusieurs problèmes de performance dans les algorithmes *shadow volume* de haute qualité et présente une méthode améliorée qui atténue ces problèmes en remplaçant une structure d'accélération spatielle trop conservatrice par une autre plus efficace. Il nous permettra d'améliorer de façon substantielle nos algorithmes de *shadow volume*.

A Survey of Real-Time Hard Shadow Mapping Methods

[Scherzer et al., 2011]

Nous nous sommes également penchés vers les algorithmes de génération des *hard shadows*. Pour des raisons de polyvalence, de vitesse et de robustesse, les algorithmes de type *shadow mapping* sont les plus adaptés. Cet article propose une vision générale des *hard shadows*

Toutefois, nous n'implémenterons pas d'algorithme de *hard shadows* en tant que telles, car celles-ci n'apparaissent que rarement dans la nature, et sont donc moins réalistes. Nous les utiliserons dans le but de les comparer avec les *soft shadows* et de démontrer l'apport de réalisme dans ces dernières.

Chapter 3

Etat de l'art

3.1 Introduction

Les ombres sont un des éléments les plus important dans la représentation d'une scène. Ce sont elles qui vont donner des informations indispensables pour la perception humaine d'une scène non réelle mais qui se veut réaliste. Une implémentation d'ombre se doit donc d'être la plus précise et correcte pour ne pas briser le réalisme d'une scène générée par ordinateur.

Plusieurs types d'algorithmes permettant de générer ces ombres ont été présentés au fil du temps dans la littérature scientifique. Ceux-ci représentent des ombres qui peuvent être divisées en deux grandes catégories :

- les *Hard Shadow*,
- les *Soft Shadow*.

Les ombres de type *Hard Shadow* seront moins considérés ici car elles sont moins réalistes. En effet, ces ombres sont uniformément noire et ne représentent que l'ombre générée par un point lumineux. D'un autre côté, les ombres de type *Soft Shadow* sont beaucoup plus réalistes et sont celles qui sont les plus utilisés. Les algorithmes permettant de faire des ombres de type Soft Shadow seront donc ceux qui seront le plus étudiés ici.

Il existe plusieurs types de *Soft Shadow algorithms*, les principaux sont ceux de:

- *Ray tracing*
- *Shadow mapping*
- *Shadow volume*

3.2 Ray tracing

L'algorithme de *Ray tracing* a été présenté pour la première fois en 1968 par Arthur Appel[Appel, 1968] sous le nom de *Ray casting*. Il a ensuite été continué, sous le nom de *Ray tracing* cette fois, en 1978 par Whitted [Whitted, 1978], qui y a rajouté la réflexion et la réfraction de la lumière. Cet algorithme consiste à tracer un rayon depuis le point de vue jusque chaque pixel créant une *ray surface*. La surface la plus proche du point de vue sera donc celle qui sera visible. A partir de chaque pixel, il faudra ensuite relier la source lumineuse. Si ce rayon a une intersection avec un quelconque objet, ce pixel sera dans l'ombre. Ceci n'est évidemment pas optimal puisqu'il nécessite un calcul pour chaque pixel de la scène.

Nous allons implémenter cet algorithme mais allons cependant nous concentrer sur deux types d'implémentation d'ombres : les *Shadow Volumes algorithms* et les *Shadow Map algorithms*. Ce sont en effet ceux qui sont actuellement les plus utilisés. Nous allons premièrement mettre en pratique les bases de l'algorithme, puis ensuite nous tourner vers des améliorations ou des spécialisations de ceux-ci.

3.3 Shadow volume

L'algorithme de *Shadow Volume* a été introduit par Crow [Crow, 1977]. Il a ensuite été implémenté grâce à l'accélération matérielle[Fuchs et al., 1986] mais n'a été que peu utilisé jusqu'à la proposition de Tim Heidmann d'accélérer matériellement cet algorithme sur du matériel accessible au grand public. Cette logique a donné naissance à l'algorithme de *z-pass*[Heidmann, 1991].

La méthode de *z-pass* consiste à premièrement initialiser un *stencil buffer* à zéro et un *depth buffer* avec les valeurs de profondeurs des objets visibles pour ensuite rastérisater les côtés des *shadow volumes*. Pour chaque partie de *shadow volume*, il s'agit ensuite d'incrémenter le pixel du *stencil buffer* correspondant si la face à une normale dans le sens inverse (on rentre alors dans le *shadow volume*) et décrémenté lorsque l'on en ressort. Le *shadow count* représentera alors le niveau d'ombre dans lequel est plongé le point fixé et l'absence d'ombre si celui-ci est égal à 0.[Andrew Woo, 1990] Ce comptage peut être théoriquement réaliser jusqu'à une distance infinie, grâce à une méthode appelée *z-fail* ou *Depth fail*[Bilodeau and Songy, 1999, Carmack, 2000].

Les algorithmes de *z-pass* ont néanmoins un défaut dans le cas où l'on place l'observateur dans l'ombre (dans un ou plusieurs shadow volumes). Ce problème a été en partie résolu par HORNUS et autres[Hornus et al., 2005], qui vont proposer d'aligner la vue de la source lumineuse avec celle de l'observateur. Cette technique a été développée en comparant l'algorithme de z-pass, dont ils s'inspirent et qu'ils ont amélioré, avec celui de z-fail. Cette solution n'est pas encore optimale mais à ce jour, aucune autre alternative n'a été proposée.

Dans l'article [Lloyd et al., 2004], les auteurs présentent une nouvelle technique qui utilise le *Culling and Clamping (CC)* permettant d'éviter de générer des *shadow volumes* qui sont eux-même dans l'ombre ou qui n'interviennent pas dans l'image finale, ce qui a pour but d'améliorer les performances et donc d'accélérer la génération des ombres dans une scène.

Aila et Akenine-Molle font remarquer en 2004 [Aila and Akenine-Möller, 2004] que les performances de génération des ombres sont inversément proportionnelles à la taille des *shadow volumes*. Pour remédier à cela, ils proposent un nouvel algorithme visant à réduire le temps de rastérisation. Cet algorithme est composé de deux étapes. La première étape consiste à trouver des zones de 8x8 pixels dont les bords sont soit complêtement dans l'ombre soit complêtement illuminés. La seconde étape verra s'effectuer une génération pixel par pixel de l'ombre des pixels se trouvant aux bords de l'ombre.
En 2004, Chan et Durand[Chan and Durand, 2004] utilisèrent une technique utilisant à la fois un algorithme de *shadow mapping* et un algorithme de *shadow volume*. Le premier est d'abord utilisé pour créer une *hard shadow* et obtenir la silhouette de l'ombre. L'algorithme de *shadow volume* est ensuite utilisé pour généré une ombre correcte (*soft shadow*) à partir de cette silhouette.

3.4 Shadow mapping

L'algorithme de *Shadow Mapping* a, quant à lui, été introduit par Lance Williams [Williams, 1978].

Le principe des *Shadow Mapping Algorithm* est de dresser dans un premier temps une carte de disparité (*depth map/image*) de la scène, comme vue depuis la source de lumière. Pour chaque *texel*, la profondeur de l'objet le plus proche de la source lumineuse sera stockée. Cet algorithme n'est pas optimal et la technique du *Percentage closer filtering*[Reeves et al., 1987, Fernando, 2005] résoud un problème d'aliasing présent.

Dans le début des années 2000, plusieurs algorithmes utilisant un *filtering* furent présentés. Celui-ci permet d'utiliser une shadow map de basse résolution tout en présentant des résultats convaincants. En 2005, Donelly et Lauritzen [Donnelly and Lauritzen, 2006] proposent un algorithme utilisant la variance de la distribution des profondeurs, visant à réduire fortement l'aliasing habituellement présent dans les algorithme de *shadow mapping* basiques tout en nécessitant peu de stockages et de calculs supplémentaires. En 2008, un nouvel algorithme est présenté[Annen et al., 2008], proposant une autre méthode pour réduire l'aliasing mais avec une technique encore plus efficace et produisant moins d'artefacts graphiques.

Cependant, l'utilisation de *shadow map* de basse résolution entraine un flou forcé, empêchant la création d'ombres nettes. D'autres algorithmes ont été proposés pour améliorer la précision sans demander de ressources trop énormes.

Les premiers sont les algorithmes appelés *Perspective Shadow Map*[Wimmer et al., 2004, Stamminger and Drettakis, 20
Lloyd et al., 2008] utilisent le *warping*, qui permet d'avoir de bons résultats mais dégénerent en *shadow map* ordinaire.

Les seconds utilisent le *Partitioning*. Cette approche permet de diviser le frustum de vue et d'utiliser une *shadow map* pour chaque sous-frustum. Cependant, pour être le plus précis possible, cette technique requiert un grand nombre de subdivisions, ce qui affecte les performances.

Un des seuls algorithmes qui présente une précision au pixel près et qui présente des bonnes performances pour le temps réel est celui qui a été présenté par Sintorn et Assarsson[Sintorn and Assarsson, 2009]. Celui-ci se focalise sur les ombres de la pilosité, qui nécessite de la précision et obtient pourtant des résultats corrects.

L'algorithme GEARS[Wang et al., 2014] rajoute un élément pris en compte, la dynamicité de la scène illuminée ainsi que celle de la lumière tout en gardant des excellentes performances.

Chapter 4

Description de l'implémentation

Dans ce chapitre, nous décrirons les langages et bibliothèques utilisés ainsi que les algorithmes que nous allons implémenter.

4.1 Render 2D et 3D

OpenGL s'est imposé comme l'API de choix étant donné sa spécification ouverte, ses fonctions bas niveau et sa disponibilité sur un grand nombre de plate-formes. Nous utiliserons OpenGL pour créer des scènes 3D, animer une/des caméra(s) et une/des source(s) de lumière mais également pour générer des ombres avec les algorithmes retenus.

Nous essayerons d'utiliser un maximum les fonctions modernes d'OpenGl majoritairement utilisées dans les jeux et animations d'aujourd'hui. Cependant, les consignes nous imposent virtualbox qui ne permet pas, au premier abord, d'utiliser de tels outils. Nous allons donc faire un maximum de recherche pour passer outre cette limitation.

4.2 Langages et bibliothèques utilisés

Dans cette section, nous allons présenter rapidement les langages de programmation et bibliothèques que nous allons utiliser dans notre projet.

- Nous allons utiliser **Python 2.7** comme langage de programmation car il permet un développement rapide et possède des bindings vers les bibliothèques OpenGL, OpenCL, numPy etc... écrites en C/C++. Il permet donc d'allier la rapidité d'écriture des langages de scripts à la rapidité d'exécution des langages compilés. Nous utiliserons aussi Python Package Index et les virtualenv pour nous faciliter la tâche.

- Nous allons utiliser **pyQt** comme librairie qui s'interfacera avec la célèbre bibliothèque **Qt 5**. Nous utiliserons cette interface graphique étant donnée sa facilité d'utilisation et son caractère complet. De plus nous l'avons pour la plupart déjà utilisée durant notre cursus.

- Nous allons également utiliser **OpenCL** pour profiter de puissance de calcul des cartes graphiques modernes et ainsi utiliser des algorithmes puissant en temps réel. OpenCL nous permettra d'être très polyvalent puisque le même programme peut s'exécuter indifféremment sur un CPU ou n'importe quel GPU muni du pilote approprié sans devoir être recompilé.

- Nous allons enfin utiliser **vispy**, une librairire de visualisation de données grâce à openGL, de par sa facilité d'utilisation, l'utilisation d'openGL moderne et sa documentation claire.

4.3 Algorithmes retenus

Après avoir passé en revue les différents articles et exploré les connaissances en matière d'algorithmes d'ombres, nous avons décidé de nous orienté vers 3 types d'implémentations :

4.3.1 Shadow volume

L'avantage de ces algorithmes est leur précision et leur réalisme mais ils peuvent être moins efficaces que le *shadow mapping*. Nous allons les implémenter car ils sont très répandus dans le monde du jeu vidéo et des animations. C'est en effet le type d'algorithme utilisé pour générer des ombres dans le moteur de rendu des studios Pixar, *PhotoRealistic RenderMan*.

4.3.2 Shadow mapping

Ces algorithmes permettent de ne générer les ombres que du point de vue de la caméra et donc de s'épargner un grand nombre de calculs inutiles, ce qui améliore les performances mais leur réalisme dépend fortement de la résolution de la *shadow map*. Les implémentations les plus récentes de ce type d'algorithmes sont pour le moment les plus réalistes et les plus précises.

Bibliography

[Aila and Akenine-Möller, 2004] Aila, T. and Akenine-Möller, T. (2004). A hierarchical shadow volume algorithm. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 15–23. ACM.

[Andrew Woo, 1990] Andrew Woo, Pierre Poulin, A. F. (1990). A survey of shadow algorithms. In *IEEE Computer Graphics and Applications*, volume 10, pages 13–32.

[Andrew Woo, 2012] Andrew Woo, P. P. (2012). Shadow algorithms data miner. A K Peters/CRC Press.

[Annen et al., 2008] Annen, T., Mertens, T., Seidel, H.-P., Flerackers, E., and Kautz, J. (2008). Exponential shadow maps. In *Proceedings of graphics interface 2008*, pages 155–161. Canadian Information Processing Society.

[Appel, 1968] Appel, A. (1968). Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 37–45. ACM.

[Bilodeau and Songy, 1999] Bilodeau, B. and Songy, M. (1999). Real time shadows. In *Creative Labs Sponsored Game Developer Conference, Creative Labs Inc.*

[Carmack, 2000] Carmack, J. (2000). Z-fail shadow volumes. In *Internet Forum.*

[Chan and Durand, 2004] Chan, E. and Durand, F. (2004). An efficient hybrid shadow rendering algorithm. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, pages 185–195. Eurographics Association.

[Crow, 1977] Crow, F. C. (1977). Shadow algorithms for computer graphics. In *ACM SIGGRAPH Computer Graphics*, volume 11, pages 242–248. ACM.

[Donnelly and Lauritzen, 2006] Donnelly, W. and Lauritzen, A. (2006). Variance shadow maps. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 161–165. ACM.

[Fernando, 2005] Fernando, R. (2005). Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches*, page 35. ACM.

[Fuchs et al., 1986] Fuchs, H., Goldfeather, J., Hultquist, J. P., Spach, S., Austin, J. D., Brooks Jr, F. P., Eyles, J. G., and Poulton, J. (1986). Fast spheres, shadows, textures, transparencies, and image enhancements in pixel-planes. In *Advances in computer graphics I*, pages 169–187. Springer.

[Hasenfratz et al., 2003] Hasenfratz, J.-M., Lapierre, M., Holzschuch, N., Sillion, F., GRAVIR, A., et al. (2003). A survey of real-time soft shadows algorithms. In *Computer Graphics Forum*, volume 22, pages 753–774. Wiley Online Library.

[Heidmann, 1991] Heidmann, T. (1991). Real shadows, real time. *Iris Universe*, 18:28–31.

[Hornus et al., 2005] Hornus, S., Hoberock, J., Lefebvre, S., and Hart, J. (2005). Zp+: correct z-pass stencil shadows. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 195–202. ACM.

[Jaakko Lehtinen, 2006] Jaakko Lehtinen, Samuli Laine, T. A. (2006). An improved physically-based soft shadow volume algorithm. In *Eurographics*, volume 25. The Eurographics Association and Blackwell Publishing.

[Kolivand and Sunar, 2013] Kolivand, H. and Sunar, M. S. (2013). A survey of shadow volume algorithms in computer graphics. In *IETE Technical Review*, volume 30. Medknow Publications & Media Pvt. Ltd.

[Lili Wang and Popescu, 2014] Lili Wang, Shiheng Zhou, W. and Popescu, V. (2014). Gears: A general and efficient algorithm for rendering shadows. In *Computer Graphics Forum*, volume 33, pages 264–275. Wiley Online Library.

[Lloyd et al., 2008] Lloyd, D. B., Govindaraju, N. K., Quammen, C., Molnar, S. E., and Manocha, D. (2008). Logarithmic perspective shadow maps. *ACM Transactions on Graphics (TOG)*, 27(4):106.

[Lloyd et al., 2004] Lloyd, D. B., Wendt, J., Govindaraju, N. K., and Manocha, D. (2004). Cc shadow volumes. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, pages 197–205. Eurographics Association.

[Reeves et al., 1987] Reeves, W. T., Salesin, D. H., and Cook, R. L. (1987). Rendering antialiased shadows with depth maps. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 283–291. ACM.

[Savchenko, 2005] Savchenko, S. (2005). Algorithms for dynamic shadows. In *Dr Dobb's Journal*, volume 30, page 50. ProQuest Central.

[Scherzer et al., 2011] Scherzer, D., Wimmer, M., and Purgathofer, W. (2011). A survey of real-time hard shadow mapping methods. In *Computer Graphics Forum*, volume 30, pages 169–186. Wiley Online Library.

[Sintorn and Assarsson, 2009] Sintorn, E. and Assarsson, U. (2009). Hair self shadowing and transparency depth ordering using occupancy maps. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 67–74. ACM.

[Stamminger and Drettakis, 2002] Stamminger, M. and Drettakis, G. (2002). Perspective shadow maps. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 557–562. ACM.

[Wang et al., 2014] Wang, L., Zhou, S., Ke, W., and Popescu, V. (2014). Gears: A general and efficient algorithm for rendering shadows. In *Computer Graphics Forum*. Wiley Online Library.

[Whitted, 1978] Whitted, T. (1978). A scan line algorithm for computer display of curved surfaces. *ACM SIGGRAPH Computer Graphics*, 12(SI):8–13.

[Williams, 1978] Williams, L. (1978). Casting curved shadows on curved surfaces. In *ACM Siggraph Computer Graphics*, volume 12, pages 270–274. ACM.

[Wimmer et al., 2004] Wimmer, M., Scherzer, D., and Purgathofer, W. (2004). Light space perspective shadow maps. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, pages 143–151. Eurographics Association.