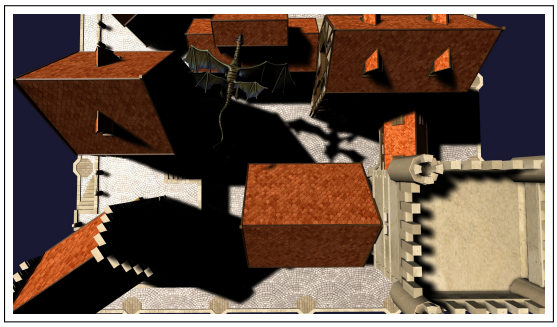
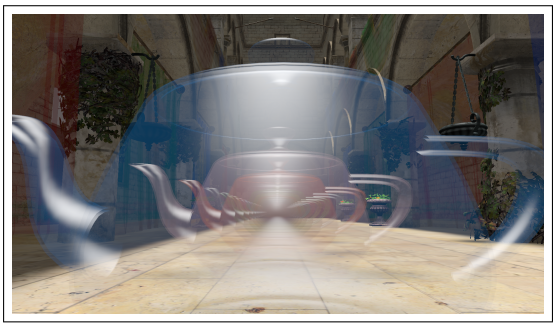
Chef-d'Œuvre

Moment Based Rendering

Conception détaillée





Baptiste Delos, Mehdi Djemai, Alban Odot,
Pierre Mézières et Jean-Baptiste Sarazin

Encadrant : Mathias Paulin

30 janvier 2019

Université Toulouse III Paul Sabatier
Université de Toulouse

IRIT

1 Introduction

L'objectif de ce projet est d'implanter des méthodes de calcul d'ombre et de transparence basées sur les moments des profondeurs et des indices de transmittance des fragments. Il s'agira de comparer ces techniques avec d'autres méthodes de calcul d'ombrage et de transparence. Les comparaisons seront réalisées selon deux critères : la qualité visuelle et la performance. Le premier, étant subjectif, ne demandera pas de mise en place d'outils ou de modules dédiés et sera donc orienté utilisateur. Le second, plus quantitatif, sera à l'origine d'un module de mesure de temps et d'analyse de données. Ce projet est donc structuré sur deux axes principaux : l'implantation de méthodes de rendu d'une part et les mesures et comparaisons de celles-ci d'autre part.

Pour rappel, en accord avec les spécifications de ce projet, nous allons développer au total six méthodes de rendu :

Ombrage
- Percentage-Closer Filtering [RSC87]
- Four Moment Shadow Mapping [PK15]
- Variance Shadow Mapping [DL06]

Transparence
- Depth Peelling [NVI01]
- Moment Based Order Independant Transparency [MKKP18]
- Weighted Blend Order Independant Transparency [MB13]

L'objectif du projet est d'intégrer, sous forme d'un *plugin*, ces différentes méthodes au sein d'un moteur 3D. Nous détaillons ici la conception de ces méthodes au coeur de notre base logicielle. Afin de faciliter au mieux le développement du projet, plusieurs améliorations ont d'ores et déjà été identifiées pour le moteur. Ce rapport de conception vient donc poser les bases de la conception du projet. La vue d'ensemble du système à élaborer fera l'objet de la première partie. Nous explorerons dans un second temps la conception du système plus en profondeur, à l'échelle des modules qui le composent. Nous tenterons d'établir par la suite un cadre de tests pour les fonctions identifiées, pour enfin aboutir sur une révision en détail du planning prévisionnel et de la gestion des risques, établis dans le rapport de spécifications.

2 Vue d'ensemble détaillée du système

Cette première partie présente en profondeur, sur la base de la structure générale abordée dans les spécifications du projet, l'architecture détaillée du système et les interactions utilisateur qu'elle rend possibles. Ces dernières sont mises en évidence aux moyens de diagrammes de séquences et d'activités qui permettent d'apprécier davantage les communications entre les modules et les actions envisageables par l'utilisateur pour le paramétrage des techniques de rendu.

Pour rappel, le projet est découpé en trois modules :

- Module d'ombrage
- Module de transparence
- Module de comparaison

L'exclusion mutuelle de ces différents modules au sein du système représente un atout important pour le développement de ce dernier. La seule communication effectuée entre les modules provient du module de comparaison. En effet, les modules d'ombrages et de transparence doivent pouvoir accéder au module de comparaison afin de pouvoir sauvegarder les images générées par les méthodes qu'ils comportent.

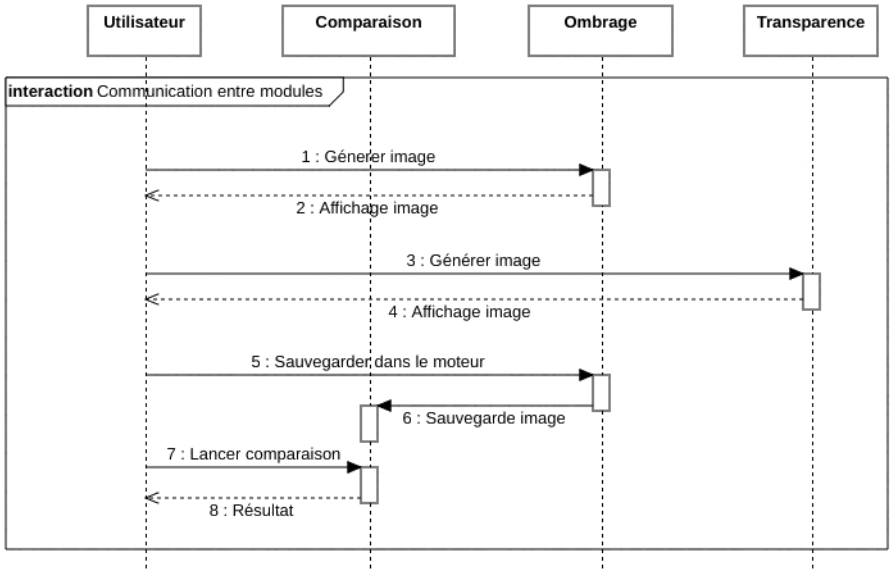


FIGURE 1 – Communication entre les modules

La figure 1 présente un diagramme de séquence possible simplifié qui donne un aperçu des interactions des différents modules du projet, dont on trouvera en annexe (figure 9), une version plus complète. Il met en avant, pour l'utilisateur, la possibilité d'interagir avec les trois modules du projet dans plusieurs buts : modifier les paramètres de rendu dans les deux modules de rendu (ombrage et transparence) et lancer une comparaison entre les images générées.

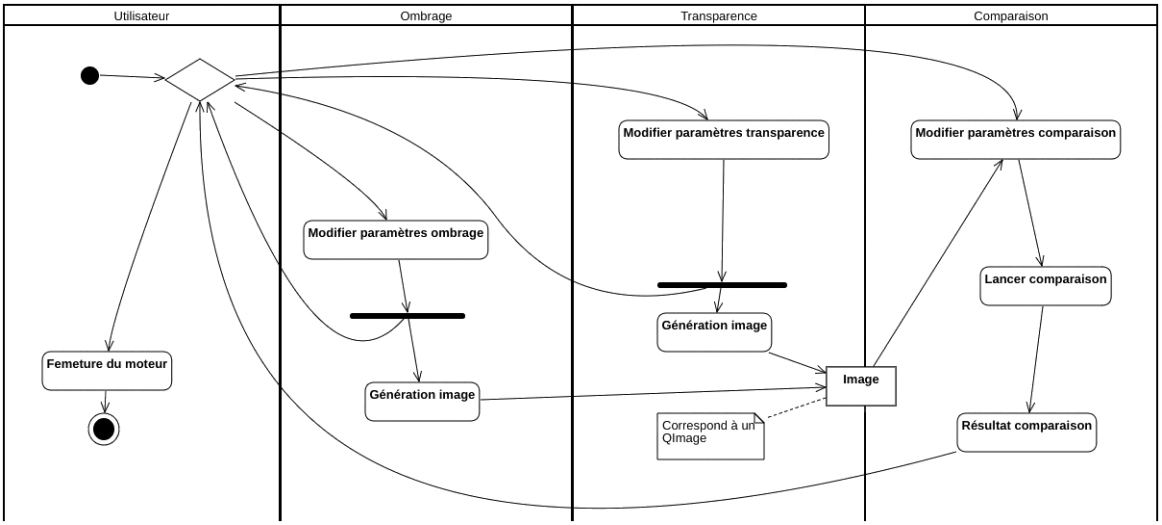


FIGURE 2 – Diagramme d'activité de l'extension

Note : On peut noter que le format des images sauvegardées dans le moteur correspondra au type *QImage* de Qt. En effet, Qt permet de récupérer directement le *framebuffer* d'une fenêtre OpenGL dans ce type.

3 Détails des modules du système

3.1 Améliorations du moteur

Plusieurs améliorations du moteur 3D sont nécessaires au bon développement du projet. Nous avons tenté d'identifier les améliorations qui permettrait de faciliter la mise en place de ce projet. Nous allons donc décrire de manière succincte quelles solutions de conception que nous prévoyons, ou celle que nous avons déjà mises en place.

3.1.1 Hiérarchie de *renderers*

Le projet nécessite d'utiliser différents *renderers*. Or, le moteur de base ne permet pas d'en interchanger facilement. Pour éviter une utilisation chaotique des différentes méthode de rendu, nous avons choisi d'organiser ces gestionnaires de rendu selon le patron de conception **Stratégie** : l'objectif est alors de pouvoir permuter d'algorithme de manière dynamique.

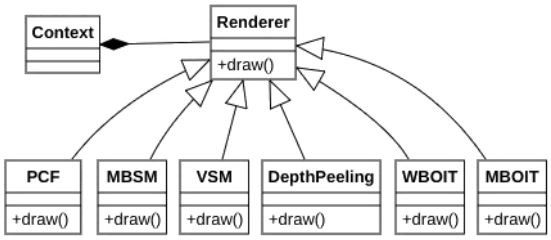


FIGURE 3 – Hiérarchie des *renderers*

3.1.2 Gestion avancée de la caméra

Afin de fournir une réelle comparaison entre les différentes méthodes, nous avons amélioré la gestion de la caméra pour pouvoir, lors du chargement de la scène, fixer différents points de vues. Ces différentes positions et orientations seront sélectionnables directement par l'utilisateur dans une *combobox* située à côté des caméras. La sélection d'un point de vue viendra modifier la position et l'orientation de la caméra courante.

3.1.3 Hiérarchie des *frameBuffer*

Les *frame buffers* nous permettent de sauvegarder une combinaison de buffers d'écrans, souvent indispensables pour la mise en œuvre des méthodes de rendu. Dans la mesure où OpenGL permet de créer des *frameBuffer* personnalisés, pour chaque méthode, nous pourrons donc définir le *frame buffer* dont nous avons besoin. Afin de clarifier le code, mais également afin de faciliter leur réutilisation, nous tenterons de mettre en place une hiérarchie de *frame buffers*. Cela nous permettra aussi de réduire les sources d'erreurs possibles, en évitant d'éventuelles copies maladroites de code.

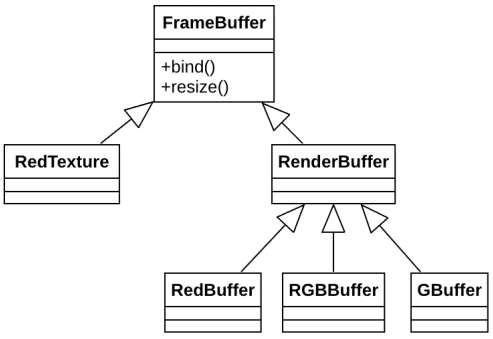


FIGURE 4 – Hiérarchie des *frameBuffers*

3.2 Implantation du *plugin*

L'implantation du plugin suit le modèle requis dans le moteur. Ce modèle est représenté par une classe qui implante l'interface *PluginInterface* du moteur. Cette classe permet d'ajouter directement dans le moteur les données utiles du plugin, comme les différents *widgets* et *renderers* utilisés. Trois *widgets* correspondent donc aux trois modules du projet. De même, six *renderers* se chargeront de la gestion des six techniques de rendus utilisées.

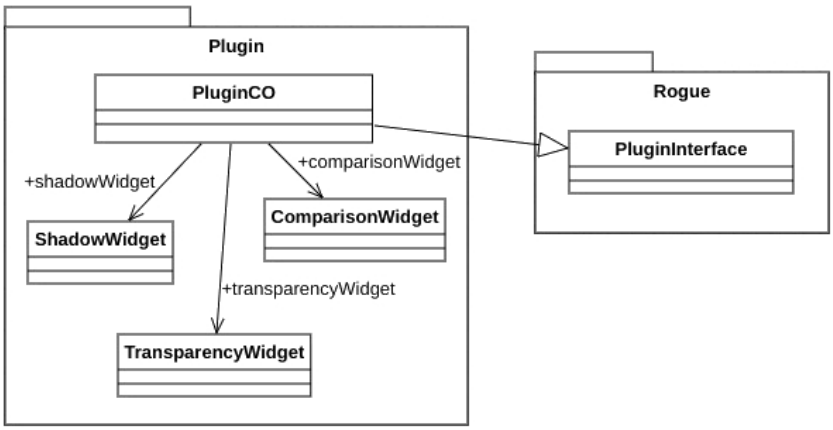


FIGURE 5 – Présentation du plugin

3.3 Module d'ombrage

Pour rappel du rapport de spécifications, le module d'ombrage doit permettre, au travers d'une interface graphique, d'effectuer un rendu des ombres via trois *renderers*, un pour chaque méthode de rendu. Ces méthodes de rendu sont le *pourcentage closer filtering* [RSC87] (PCF), le *variance shadow mapping*[DL06] (VSM) et le *moment based shadow mapping*[PK15] (4MSM). Pour chacune des méthodes nous pourrons régler les hyperparamètres des méthodes, lancer un rendu et garder en mémoire le résultat. Nous détaillons ici l'architecture de ce module et son intégration au sein du moteur Rogue.

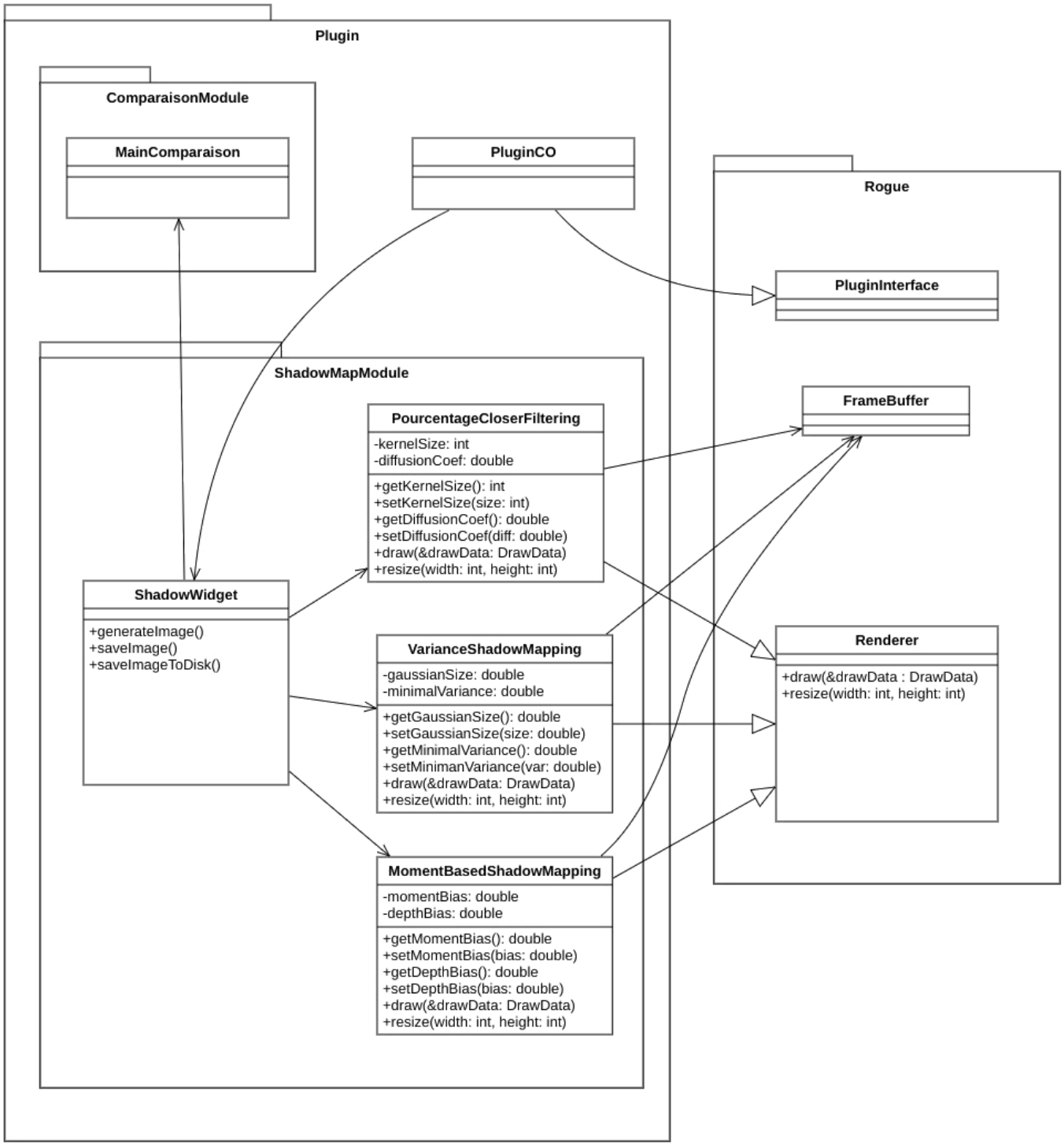


FIGURE 6 – Diagramme de classe du module d'ombrage

3.4 Module de transparence

Sur le modèle du module d'ombrage, le module chargé de la gestion de la transparence permettra à l'utilisateur le paramétrage de trois méthodes de rendu : la méthode de référence du *Depth Peeling* [NVI01], *Weighted-Blended OIT* [MB13] et l'algorithme principal *Moment-Based OIT* [MKKP18]. L'intégration de ces méthodes repose sur l'organisation du moteur en unités de rendu (*renderers*), décrite en 3.1.1. Nous nous intéressons ici au détail de

l'implantation du module de transparence. (Paquet *TransparencyModule* de la figure 7)

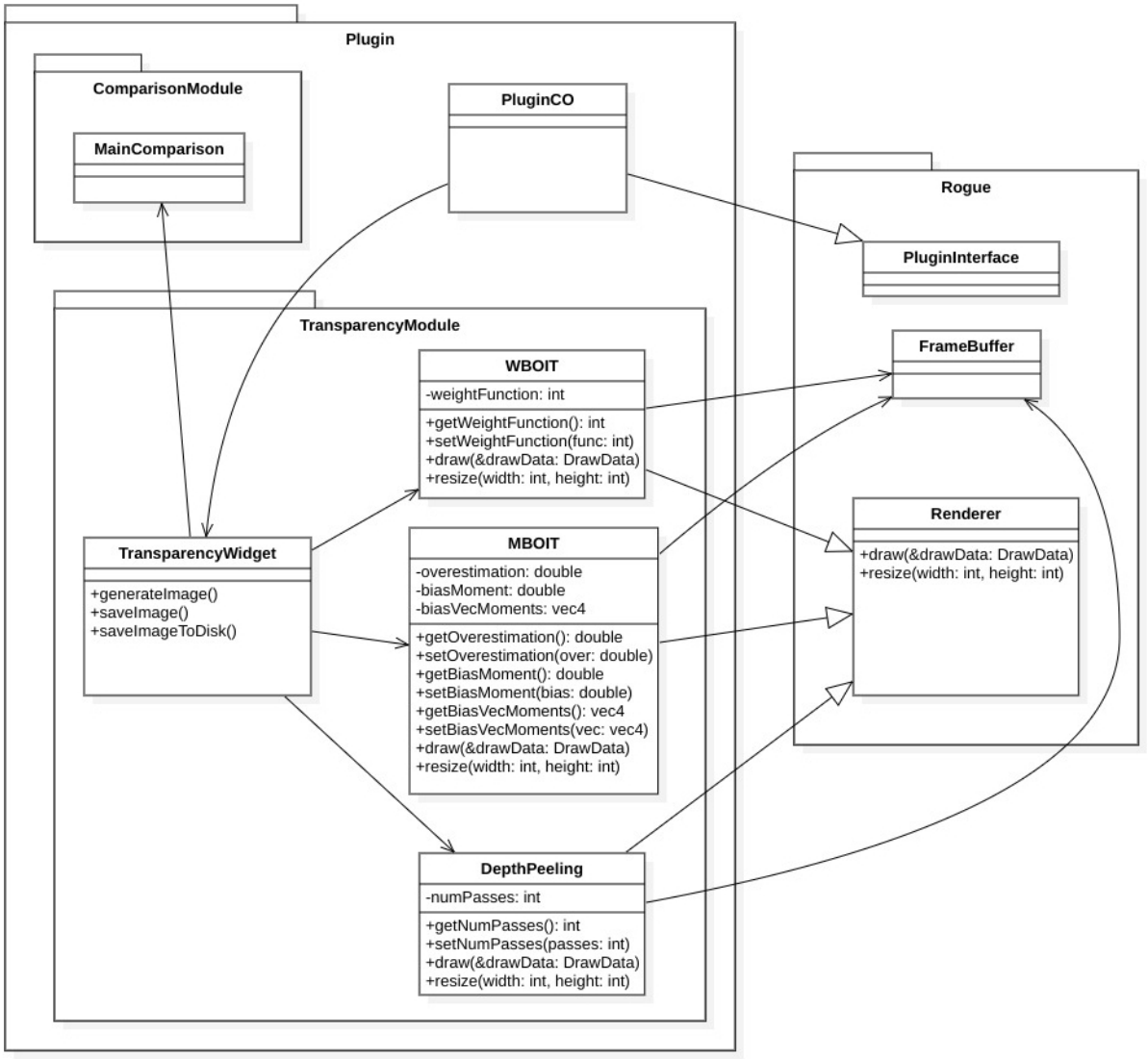


FIGURE 7 – Diagramme de classe du module de transparence

3.5 Module de comparaison

D'un point de vue conception, ce module est sûrement le plus simple à mettre en place. Il requiert une classe **MainComparison** permettant de stocker les images sauvegardées avec leurs informations de génération, ainsi qu'une classe **OperatorsComparison** permettant d'appliquer différents opérateurs de comparaison sur deux images.

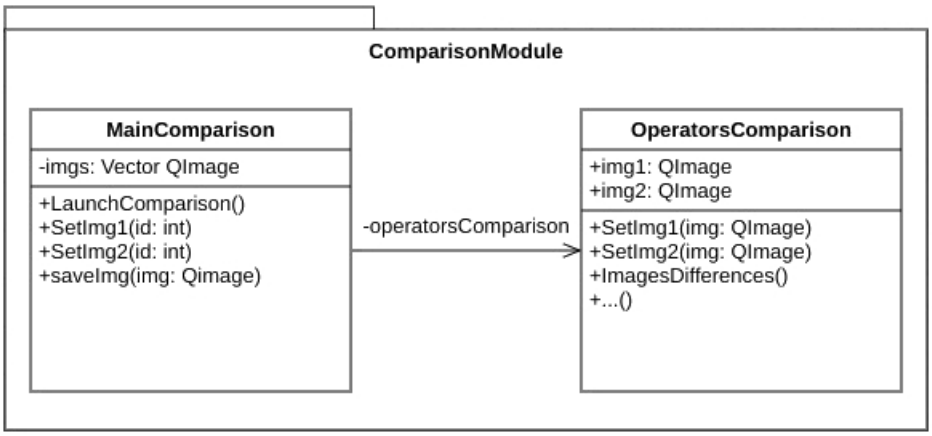


FIGURE 8 – Diagramme de classe du module de comparaison

Note : On ne s'est pas totalement arrêté sur les méthodes de comparaison qui seront mis en place dans la version finale du plugin. Il y aura au moins un simple opérateur de différence d'images et le calcul de l'erreur quadratique. Si le temps nous le permet, nous nous orienterons, dans la mesure du possible, vers des métriques de similarité plus robustes. Néanmoins, nous dirigeons actuellement nos efforts sur la mise en place des méthodes de rendue basée sur les moments, objectif phare du projet.

On peut aussi noter que le moteur Rogue possède un petit système de pré-traitement sur les *shaders*. En effet, il permet de réaliser des inclusions entre *shader*, ce qui permettra de factoriser le code des *shaders* des différentes méthode de rendu. On peut notamment penser aux méthodes basées sur les moments, dont le code pourra être indubitablement factorisé.

4 Tests unitaires

Il s'agit dans cette partie d'établir un cadre de tests unitaires pour le code produit. Le problème qui vient se poser réside dans le fait que le cœur du code du projet se trouve dans les shaders correspondant aux différentes méthodes de rendu. Réaliser un cadre formel de tests unitaires pour nos shaders est relativement complexe. Nous avons choisi par conséquent de définir des scènes simples afin de construire divers scénarii d'ombrage et de transparence, et pouvoir ainsi tester simplement les shaders.

4.1 Module d'ombrage

La validation des calculs d'ombrage passe par la génération de textures spécifiques. De ce fait, nous générerons des cartes de profondeur ou carte d'ombre pour chaque type de lumière. Cette validation se fera sur des scènes simples et synthétiques. La liste des tests qui sera effectués est la suivante :

N°	Principe du test	Note
0	Une scène sans objet	Nous devrions obtenir une texture blanche.
1	Une scène avec une sphère centrée sur la lumière	Nous devrions obtenir une texture avec une couleur unie, dont la valeur représentera le rayon de la sphère.
2	Une scène avec deux objets l'un devant l'autre	Nous devrions obtenir texture avec une rupture de profondeur, les valeurs devant être conformes aux distances mesurées sur la scène.

À l'aide d'un *frame buffer*, nous pouvons obtenir une texture représentant pour chaque fragment la profondeur, appelée z, et comparer ce z à la distance caméra/objet sur la scène. Pour les moments d'ordre deux à quatre qui seront calculés pour les méthodes *VSM* et *4MSM*, nous pourrons vérifier que les valeurs en sortie des shaders sont bien égales à des puissances de z.

Ceci étant, pour le calcul des cartes d'ombres, nous testerons ensuite visuellement les résultats de nos rendu des lumières à l'aide de scènes synthétiques :

N°	Principe du test	Note
3	Un objet et un plan sur lequel l'ombre se projette et une lumière.	Nous allons pouvoir observer la qualité de l'ombrage (aliasing)
4	Deux objets et un plan sur lequel l'ombre se projette et deux lumières placées pour que les ombres s'intersectent.	Observer le niveau d'ombre entre une zone non occultée, une zone occultée par un objet et un zone occulté par les deux objets.
5	Plusieurs objets et plusieurs lumières	Observer le recouvrement d'ombre qui devrait occasionner une fuite de lumière dans le cas de la méthode VSM mais pas dans les autres méthodes.

4.2 Module de transparence

Par rapport aux calculs d'ombrage, les calculs de transparence ne nécessitent pas la génération de textures spécifiques. Nous allons donc pouvoir concentrer les tests sur la visualisation du résultat.

N°	Principe du test	Note
6	Scène sans objet transparent	Vérifier que le rendu des objets opaques est bien le même avec ou sans objet transparent
7	Un objet transparent devant un objet opaque	Vérifier que l'objet transparent est bel et bien transparent
8	Intersection entre un objet transparent et un objet opaque	Vérifier que l'objet transparent ne ressort pas de l'objet opaque
9	Intersection entre un objet transparent et un autre objet transparent	Vérifier s'il n'y a pas d'artefacts apparent et que le résultat est cohérent
10	Succession d'objets transparents	Vérifier que la transmittance baisse au fur et à mesure de la rencontre avec les objets transparent.

4.3 Module de comparaison

Le module de comparaison présente deux objectifs, à savoir la mesure des performances et et la comparaison visuelle des images. Pour la mesure de comparaison, nous créerons des images de synthèse de petites tailles dont nous pouvons calculer à la main les résultats, et ceux pour l'erreur quadratique moyenne et la différence entre deux images.

N°	Principe du test	Note
11	Deux images identiques.	Vérifier que l'erreur quadratique moyenne soit égale à 0 et la différence donne un résultat nul.
12	Une image blanche et une image noire	Vérifier que l'erreur quadratique moyenne soit égale à 255^2, la différence donnera la $1^{ère}$ image en sortie.
13	Deux images de petite taille (4x4), couleur prédéfinie au moment du test.	Vérifier que les résultats calculés à la main sont les même.

5 Mise à jour du planning prévisionnel et des risques

Une part du travail de l'équipe consiste à évaluer notre avancement et de le comparer à nos attentes au cours de réunions de rétrospectives à la fin de chaque itération. Pour cela, nous avons recours à des prototypages de fonctionnalités afin d'avoir un point de vue plus détaillé sur l'implantation (temps, difficultés) de la version finale. De ce fait, nos essais nous permettent la mise à jour du planning prévisionnel et de la matrice des risques originaux.

5.1 Planning prévisionnel

Le planning prévisionnel n'a pas subit de modifications majeures, seules quelques dates ont été corrigées pour représenter davantage nos prototypages et améliorations apportés au moteur. Suite à la formulation des spécifications du projet, l'élaboration de tests unitaires pour chacune des composantes des modules du système forme un ensemble de tâches supplémentaires. La vision affinée de ce planning prévisionnel, réalisé initialement dans le rapport de spécifications, est disponible en annexe (Section 6.2).

5.2 Gestion des risques

Au niveau de la gestion des risques, nous avons apporté quelques corrections, notamment sur les parties concernant le mauvais choix de moteur et de conception. En effet, nos premiers essais confirment que l'implantation des différents modules et fonctionnalités pourra être réalisée comme nous l'avions imaginé, leurs probabilités d'apparition ont donc été revues à la baisse. La compatibilité MAC est, quant à elle, assurée par le choix de version de nos shaders (GLSL 4.10) et le fait que l'équipe ait la possibilité de compiler sous MAX OS. La probabilité d'apparition est donc ici également revue à la baisse. Les méthodes ont été comprises par l'ensemble de l'équipe, seuls quelques détails techniques pourront éventuellement poser problème, créant ainsi un dépassement des délais. La probabilité d'apparition de ce risque à donc été diminuée.

Enfin nous avons identifié deux nouveaux risques à savoir :

— Une amélioration du moteur non identifiée, qui surviendrait lors du passage de la phase prototype à la phase finale d'une méthode d'ombre ou de transparence.
— Une tâche prend trop de temps à être réalisée, découlant des possibles incompréhensions ou améliorations non identifiées.

6 Annexes

6.1 Diagramme de séquence

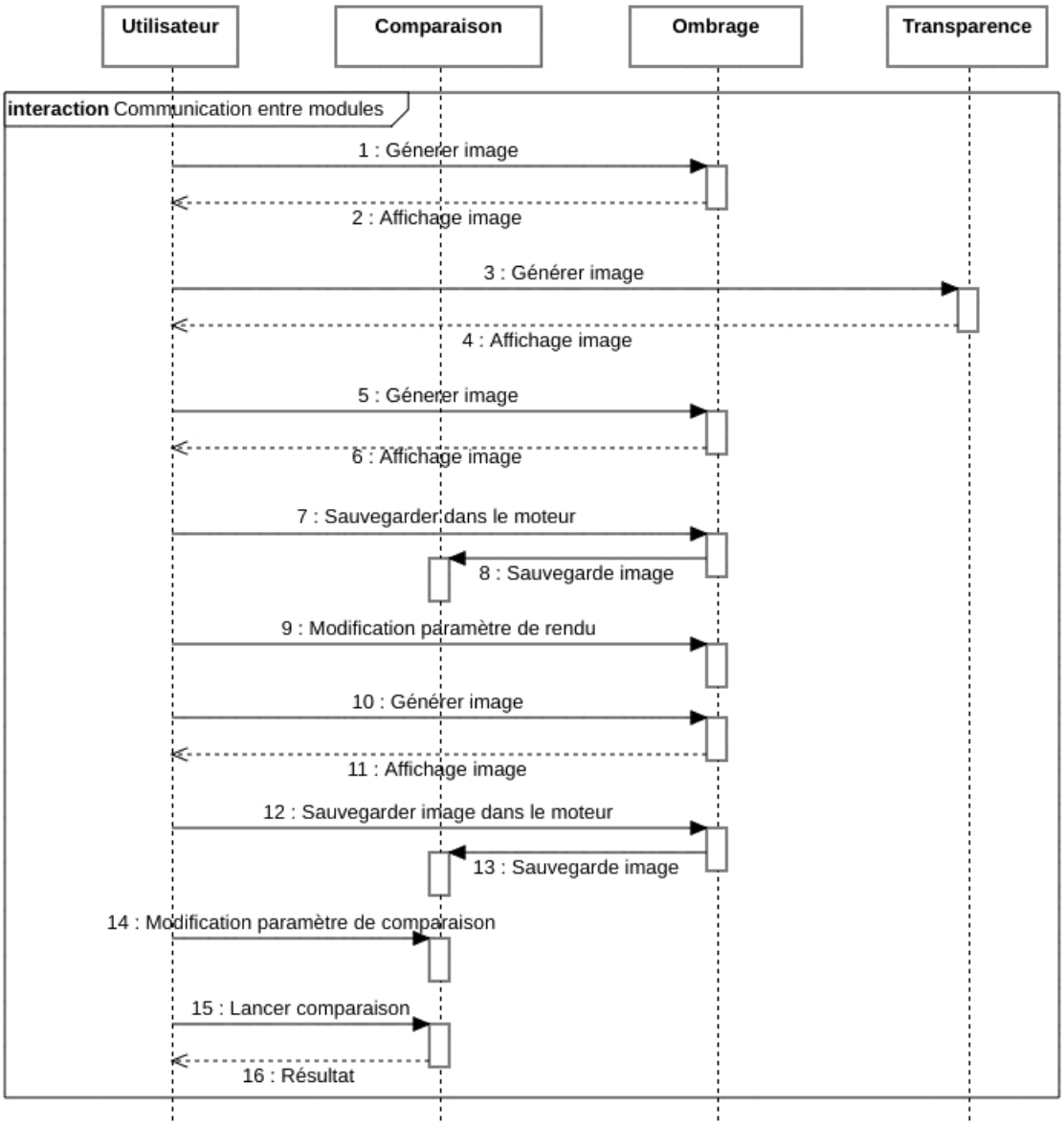
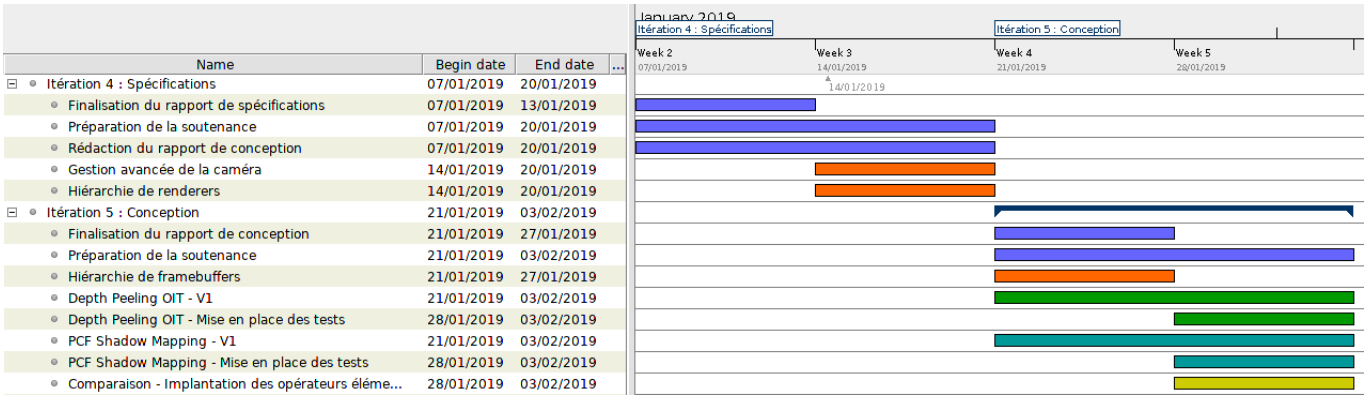


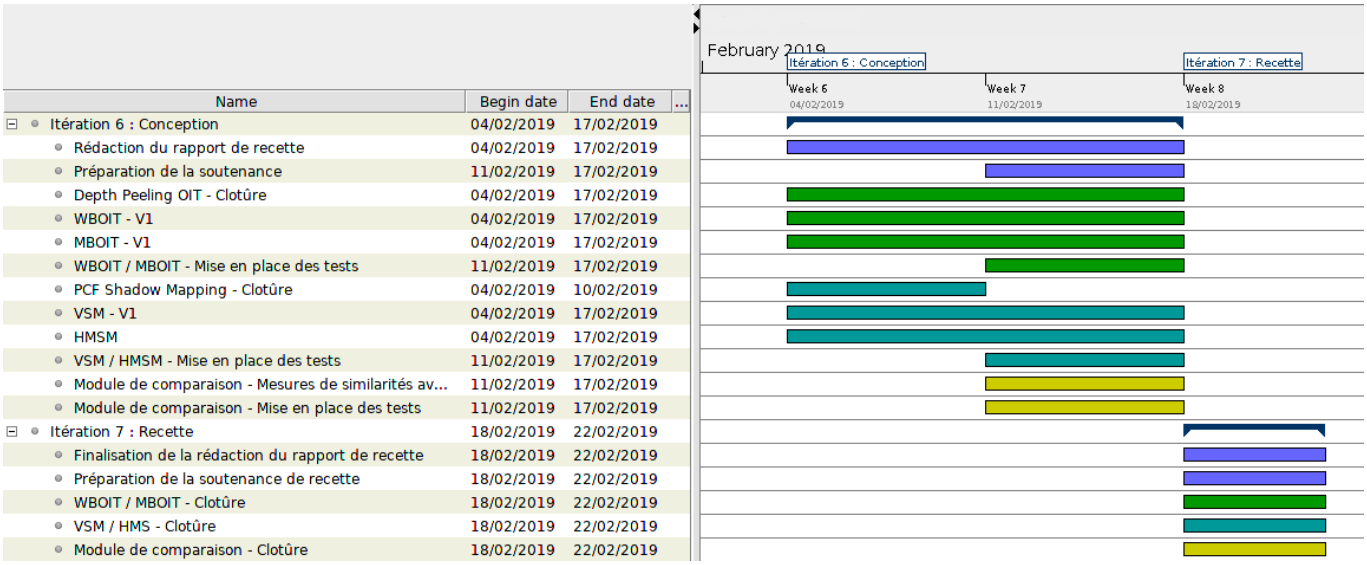
FIGURE 9 – Diagramme de séquence possible

6.2 Planning prévisionnel révisé



Name	Begin date	End date
Itération 4 : Spécifications	07/01/2019	20/01/2019
Finalisation du rapport de spécifications	07/01/2019	13/01/2019
Préparation de la soutenance	07/01/2019	20/01/2019
Rédaction du rapport de conception	07/01/2019	20/01/2019
Gestion avancée de la caméra	14/01/2019	20/01/2019
Hiérarchie de renderers	14/01/2019	20/01/2019
Itération 5 : Conception	21/01/2019	03/02/2019
Finalisation du rapport de conception	21/01/2019	27/01/2019
Préparation de la soutenance	21/01/2019	03/02/2019
Hiérarchie de framebuffers	21/01/2019	27/01/2019
Depth Peeling OIT - V1	21/01/2019	03/02/2019
Depth Peeling OIT - Mise en place des tests	28/01/2019	03/02/2019
PCF Shadow Mapping - V1	21/01/2019	03/02/2019
PCF Shadow Mapping - Mise en place des tests	28/01/2019	03/02/2019
Comparaison - Implantation des opérateurs éléme...	28/01/2019	03/02/2019

(a) Itérations 4 et 5



Name	Begin date	End date
Itération 6 : Conception	04/02/2019	17/02/2019
Rédaction du rapport de recette	04/02/2019	17/02/2019
Préparation de la soutenance	11/02/2019	17/02/2019
Depth Peeling OIT - Clotûre	04/02/2019	17/02/2019
WBOIT - V1	04/02/2019	17/02/2019
MBOIT - V1	04/02/2019	17/02/2019
WBOIT / MBOIT - Mise en place des tests	11/02/2019	17/02/2019
PCF Shadow Mapping - Clotûre	04/02/2019	10/02/2019
VSM - V1	04/02/2019	17/02/2019
HMSM	04/02/2019	17/02/2019
VSM / HMSM - Mise en place des tests	11/02/2019	17/02/2019
Module de comparaison - Mesures de similarités av...	11/02/2019	17/02/2019
Module de comparaison - Mise en place des tests	11/02/2019	17/02/2019
Itération 7 : Recette	18/02/2019	22/02/2019
Finalisation de la rédaction du rapport de recette	18/02/2019	22/02/2019
Préparation de la soutenance de recette	18/02/2019	22/02/2019
WBOIT / MBOIT - Clotûre	18/02/2019	22/02/2019
VSM / HMS - Clotûre	18/02/2019	22/02/2019
Module de comparaison - Clotûre	18/02/2019	22/02/2019

(b) Itérations 6 et 7

FIGURE 10 – Planning prévisionnel évalué sur les quatre prochaines itérations

6.3 Matrice des risques

Gris : Risque déjà identifié et modifié
Bleu : Nouveau risque

Risque	Probabilité apparition	Impact	Prévention	Solution
Indisponibilité d'un membre de l'équipe en cours de projet	20%	moyen	Faire en sorte que tous les membres du projet puissent reprendre le travail d'un autre	Répartir la charge de travail efficacement sur les autres membres du groupe
Recommencer le projet en cas de mauvais choix de moteur graphique à utiliser comme base	5%	fort	Vérifier que les fonctionnalités nécessaires de base sont parfaitement fonctionnelles	Développer de manière à ce que le code soient facilement portable sur un autre moteur (éviter la duplication de code, réduire l'utilisation de fonctions intrinsèques au moteur)
Recommencer le projet en cas d'une mauvaise conception de départ	10%	fort	-Passer beaucoup de temps sur la conception initiale et l'implantation utilisera des modèles connus pour leur flexibilité.	Repenser la conception en essayant de modifier le moins possible l'existant en utilisant des design pattern
Compatibilité MAC non fonctionnelle	5%	fort	-Commencer par rendre le projet compatible avant de commencer le développement. -S'assurer de la compatibilité directement avec la machine du client.	Contacter un expert MAC
Copyright/Copyleft des shaders fournis sur les papiers scientifiques	10%	moyen	Bien se renseigner durant la phase de réflexion	-Réécrire les shaders utilisées -Ne sera un problème qu'en cas de mise sur le marché du produit
Incompréhension et/ou incapabilité d'implémenter une méthode présente dans les papiers	10%	moyen	Passer du temps à bien comprendre les papiers	-Demander de l'aide aux autres membres du projet. -Demander de l'aide à M.Paulin
Rendu côté graphique pas assez performant (lag)	30%	fort	Limiter le nombre de données à envoyer à la carte graphique	-Réduire le nombre de données envoyé sur la carte graphique -Changer ou modifier le moteur si l'optimisation ne suffit pas
Mauvaise rédaction des documents à fournir/diapos	80%	faible	-Faire valider chaque partie par tous les membres du groupes -Envoyer une première version terminée pour correction par M.Paulin	Accepter la critique et ne pas reproduire les mêmes erreurs pour les documents suivants
Dépassement des délais	60%	fort	-Estimer le temps nécessaire à la réalisation des différentes tâches -Répartir correctement les tâches	Faire des sessions de programmation intensives

Risque	Probabilité apparition	Impact	Prévention	Solution
Une amélioration du moteur non identifiée surgit	35%	fort	Passer suffisamment de temps sur le rapport de conception	-Soit, si possible, repenser la conception pour ne pas avoir à réaliser l'amélioration -Soit réaliser l'amélioration et réévaluer le planning
Une tâche prends trop de temps à être réalisée	30%	moyen fort	Planning prévisionnel réalisé avec parcimonie	Dépends énormément des autres tâches. Réévaluation du planning pour essayer d'équilibrer le temps avec les autres tâches ou annuler la/les tâches.

Références

[DL06] William Donnelly and Andrew Lauritzen. Variance shadow maps. 2006.

[MB13] Morgan McGuire and Louis Bavoil. Weighted blended order-independent transparency. 2013.

[MKKP18] Cedrick Münstermann, Stefan Krumpen, Reinhard Klein, and Christoph Peters. Moment-based order-independent transparency. 2018.

[NVI01] Cass Everitt NVIDIA. Interactive order-independent transparency. 2001.

[PK15] Christoph Peters and Reinhard Klein. Moment shadow mapping. 2015.

[RSC87] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. 1987.