RAY OF GOD
Rapport Spécifications

Yuko Baudet - Valentin Bonnesoeur - Bastien Schatt

03 décembre 2014

Clients : Mathias Paulin - David Vanderhaeghe

Table des matières

1 Introduction

Ce rapport entre dans le cadre du Chef d'oeuvre du Master Image et Multimédia. Notre sujet concerne le développement d'une approche physique réaliste d'un modèle d'ombrage volumique en temps réel destiné au laboratoire de l'IRIT. Ce deuxième rapport sert à définir les spécifications.

Dans un premier temps nous rappellerons la problématique de notre chef d'oeuvre. Ensuite nous établirons le cahier des charges et nous y spécifierons les priorités de chaque exigences. Dans un second temps nous déterminerons les modules du système. Enfin, nous illustrerons notre planning prévisionnel à l'aide de diagrammes de Gantt.

2 Problématique du chef d'oeuvre

Cette partie est tirée de notre précédent rapport *« Méthodes et Algorithmes »* concernant les méthodes et algorithmes que nous utiliserons pour mener à bien notre projet. Le lecteur est convié à le relire afin de comprendre les points techniques.

La simulation de l'éclairage dans les milieux participants, comme la fumée ou le brouillard, augmente considérablement le réalisme du rendu de scènes complexes. Différentes méthodes de simulation d'éclairage existent mais ne sont pas réalisables en temps réel, comme les méthodes de *Ray Tracing* de Monte Carlo (utilisées par Kajiya et Herzen en 1984 [2], par exemple), qui peuvent nécessiter de plusieurs minutes à plusieurs heures (suivant la complexité de la scène) pour effectuer le rendu d'une seule image. Les méthodes de rendus temps réel pour simuler un milieu participant sont tirées ou dérivées du *« fog model »* qui existait dans les API graphiques avant l'introduction des *shaders* (OpenGL 2, DirectX 9). Néanmoins, ces approches excluent un grand nombre d'effets physiquement réalistes, comme le halo autour des sources de lumière, l'atténuation des ombres et des tâches spéculaires des surfaces ou l'ombrage volumique résultant de l'occlusion par les objets présents sur la scène.
Le domaine de la simulation de l'éclairage dans les milieux participants à tout de même évolué ces dernières années. Aujourd'hui il existe, plusieurs méthodes permettant d'avoir une approche en temps réel, par exemple en

pré-calculant et en exploitant certaines données sous formes de textures, tout en intégrant les effets précédemment mis de côté.
L'article *« Real-Time Volumetric Shadows using 1D Min-Max Mipmaps »*[1] est celui que nous devons mettre en place pour ce Chef d'oeuvre.
Une fois implémenté nous devrions être capable de visualiser une scène entièrement contenue dans un milieu participant, comme du brouillard. Les rayons de la lumières seront alors clairement visibles : on appelle ce phénomène *Ray of God*.

3 Exigences Fonctionnelles

3.1 Demande du client

Notre chef d'oeuvre répond une demande particulière du client. Il s'agit d'implémenter l'article de Chen et col. [1] dans le moteur graphique fourni par le client. Nous devons donc réaliser un module de rendu de scène avec milieu participant pouvant être facilement ajouté à un projet existant.

En plus de cette exigence, une des volontés du client est de rendre notre programme interactif : l'utilisateur devra être en mesure de pouvoir modifier les paramètres de l'application (la densité du milieu par exemple). Il nous a aussi été demandé de pouvoir visualiser les étapes de notre algorithme. Par exemple, le programme pourra afficher les arbres mip-map construits sur les rayon de vue. Cet aspect du projet sera traité dans un second temps car le résultat final du projet n'en dépend pas.

C'est donc à partir de cette demande que nous avons défini les différentes fonctionnalités à implémenter ainsi que leur importance.

3.2 Analyse fonctionnel

Le cahier des charges formalise avec précision le besoin du demandeur, en terme de fonctions. Chaque fonctionnalité est soumise à des contraintes que nous nous devons de respecter. Nous ne faisons pas référence à des solutions techniques.

Nous avons partagé le projet en deux étapes en fonction de l'importance des fonctionnalités à implémenter. La première étape doit impérativement être validée avant la fin du projet. Elle constitue le coeur même de notre chef d'oeuvre.

Détails des tâches :

Tâche 1 : Calculer le rendu d'une scène contenant un milieu participant

- Calcul de la *Depth map* et de la *Shadow map*
- Rectification épipolaire de la *Shadow map*
- Décomposition en valeurs singulières (SVD) de la luminance
- Calcul de l'approximation de la luminance grâce à la SVD
- Calcul d'un arbre min-max sur chaque ligne de la *Shadow map* rectifiée
- Calcule de la luminance finale de la scène en tenant compte des paramètres (milieu participant, valeur du coefficient de diffusion du milieu)

Contraintes : Qualité et temps de rendu équivalent à ceux de l'article de référence[1], voir figure 1.

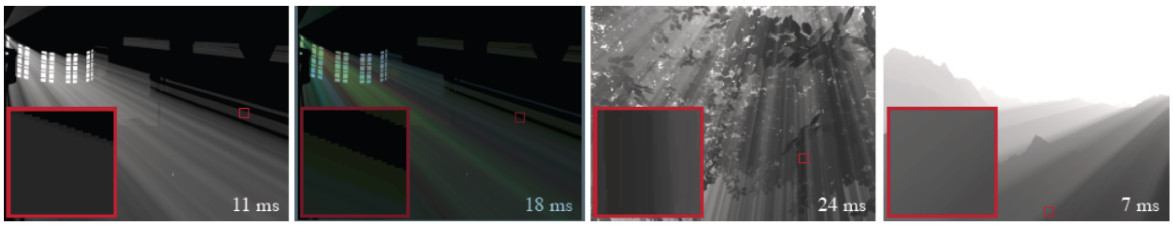

FIGURE 1 – Qualité et temps de rendu de référence. Cette image est tirée de l'article de Chen et col. [1]

Les tâches suivantes permettront d'ajouter les fonctionnalités permettant l'interaction avec l'utilisateur. Cependant, elles seront traitées dans un second temps car elles ne sont pas nécessaire au résultat final que nous devons obtenir à la fin de ce chef d'oeuvre. En effet, ces fonctionnalités sont importantes car elles répondent à un besoin du client, mais nous avons souhaité faire apparaître un ordre de priorité des tâches dans cette section du rapport.

Tâche 2 : Ajouter la possibilité de modifier le coefficient de diffusion d'une scène via l'interface

Tâche 3 : Ajouter la possibilité d'afficher les étapes de l'algorithme via l'interface

- Affichage de la *Depth map* du point de vue de la caméra
- Affichage de la *Shadow map* du point de vue de la source de lumière
- Affichage de la *Shadow map* rectifiée
- Affichage des les lignes de la *Shadow map* rectifiée à l'aide des arbres min-max avec le langage dot. Ce langage permet la description de graphe dans un format texte et de générer un fichier png afin d'afficher le graphe.

4 Vue générale du système

4.1 L'existant

Le moteur fourni par le client est organisé comme dans la figure 2.

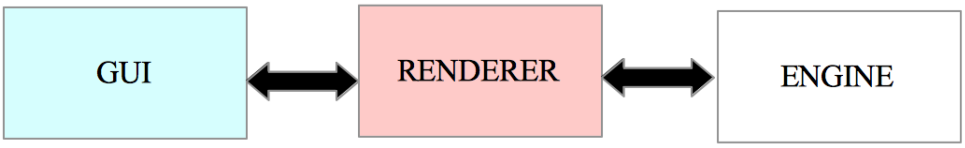

FIGURE 2 – Moteur du client

On retrouve un paradigme très connu, appelé *modèle-vue-controleur* (MVC). Sur le schema ci-dessus, la vue correspond à la partie *GUI*, le contrôleur correspond au module *Renderer* et *l'Engine* est donc le modèle.

Dans ce paradigme, la vue gère l'interface utilisateur. Dans le moteur fourni la vue est principalement implémenté à l'aide de *Qt*. Cette librairie permet de créer des fenêtre de manière très simple et de récupérer les actions utilisateurs associées à ces fenêtres. De plus la librairie permet d'intégrer facilement une zone de rendu *OpenGL*.

Dans ce cas, le modèle représente la scène. En interne, la scène est représentée sous la forme d'un graphe de scène.

Finalement le contrôleur gère les évènements et la synchronisation entre l'interface et la scène : quand la vue envoie un évènement (modification de la scène par l'utilisateur par exemple) le contrôleur est chargé de mettre la scène à jour. Dans ce module, le dessin de la scène est paramétré (choix des dimension des textures, choix et configurations des *shaders*) et délégué au module *Engine* qui parcourt le graphe de scène et fait des appels à *l'API OpenGL* pour rendre les primitives dans un buffer. Une fois que tout les rendu de primitives sont terminés sur la carte graphique, le buffer est automatiquement affiché dans l'interface grâce à *Qt*.

Dans le code fourni, le calcul de la *Depth map* est déjà effectué, tout comme le rendu de scène (sans milieu participant). Il existe aussi un moyen d'afficher différentes étapes du rendu (*Depth map* par exemple) à l'aide d'une liste déroulante.

4.2 L'implémentation

Afin d'implémenter toutes les fonctionnalités vue dans le cahier des charges, nous devrons modifier les trois modules vu ci-dessus.

Tout d'abord, le module *Renderer* devra être modifié afin d'ajouter de nouvelles étapes correspondant aux étapes clé de l'algorithme étudié (se reporter au rapport « Méthodes et Algorithmes »pour plus de détails). En plus de devoir ajouter certains *shaders* , certains (fourni avec le moteur du client) devront être modifiés afin de permettre le rendu final du milieu participant englobant la scène.

Le module *GUI* devra être modifié afin de permettre la visualisation des étapes intermédiaires de l'algorithme à implémenter. L'utilisateur pourra donc communiquer avec le programme à l'aide de différents widgets d'action lui permettant de régler l'intensité du milieu participant de la scène ou d'afficher les étapes qu'il souhaite visualiser.

En principe le module *Engine* n'a pas besoin d'être modifié. Cependant, il est probable que quelques petites modifications devront être effectuées pour régler certains problèmes qui seront rencontrés au cours du développement et purement lié à des aspects techniques.

Pour résumé, notre implémentation peut se schématiser comme dans la figure 3.

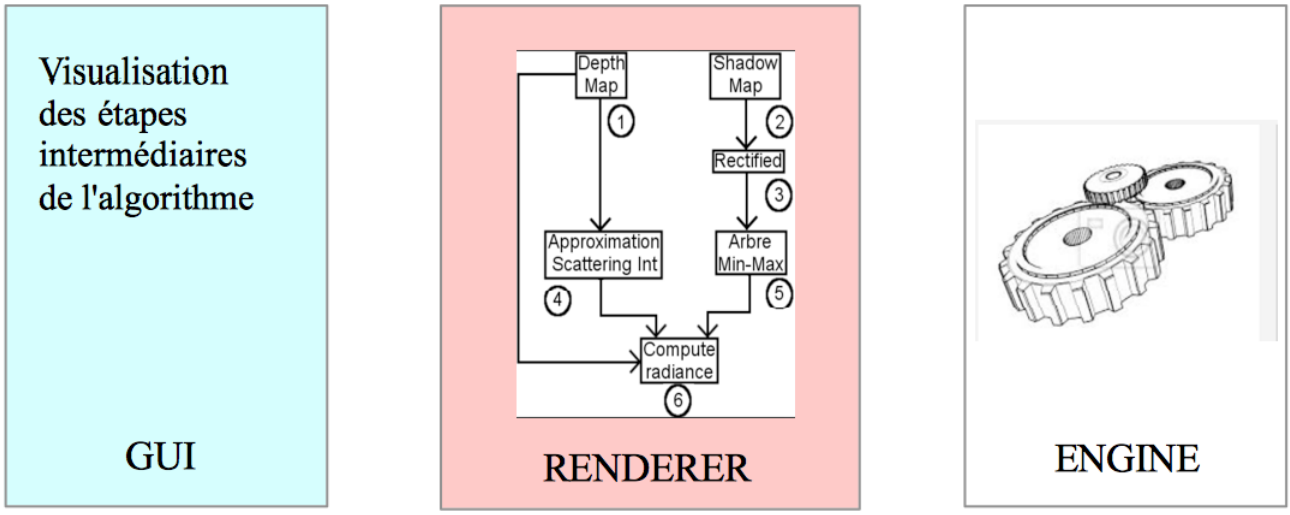

FIGURE 3 – Vue générale du système

4.3 Test de Validation

Chaque étape de notre chef d'oeuvre devra être validées. En effet, certaines étapes dépendent les unes des autres, et les erreurs potentielles de-

vront être détectées rapidement afin de ne pas fausser les autres étapes. Cela permet de gagner du temps car si des incohérences surviennent, les tests de validation permettront rapidement de savoir d'où vient le problème. De plus les tests ci-dessous devront tous passer lors de la recette.

Validation de la *Shadow map* et de la *Depth map* :
Une fois le développement des *Shadow map* et *Depth map* effectués, il faut pouvoir les afficher afin de percevoir les potentielles incohérences.

Validation de la *single scattering integral* :
Le calcul de la luminance est difficile à tester car le résultat est visuel. Pour la scène de test, il faudra au préalable que le groupe ait analyser la luminance afin de savoir si les résultats sont cohérents ou non.

Validation de la décomposition en valeurs singulières :
Des tests unitaires sont à prévoir une fois que le calcul de SVD sera implémenté.

Validation des arbres min-max :
Faire tourner la structure sur des données plus simples que les données de la scène de test afin de vérifier le min et le max des feuilles et des noeuds de l'arbre. Ces tests pourront se coupler avec les fichiers png généré grâce au langage dot afin de visualiser facilement le contenu des arbres.

5 Planning prévisionnel et Risques

5.1 Les étapes principales

La conception de la tâche 1 de notre chef d'oeuvre se déroulera en trois grandes étapes. L'annexe A est une vue d'ensemble de la phase de réalisation du projet. Ces étapes ne doivent pas être faites indépendemment les unes des

autres, mais dans l'ordre établi par le diagramme suivant :

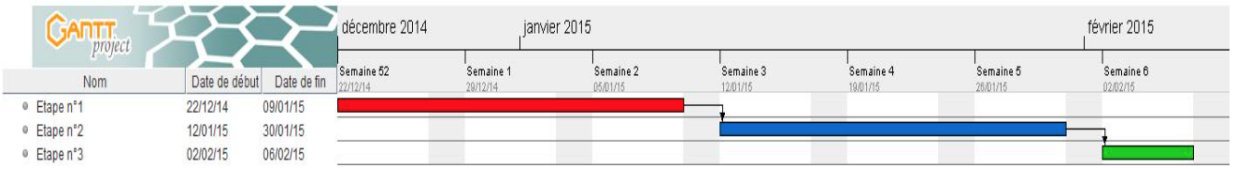

FIGURE 4 – Les grandes étapes

Chaque sous-tâches des étapes ne sont pas que phases de développement pures. Elles comprennent des tests de validations.

Etape 1

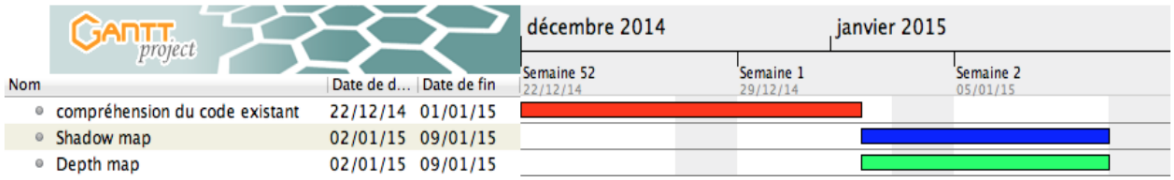

FIGURE 5 – Etape 1

Une fois que nous aurons pris le code du moteur en main, nous commencerons le développement des *Shadow map* et *Depth map*. Ces deux tâches étant indépendantes nous pouvons les réalisées en parallèles.

Etape 2

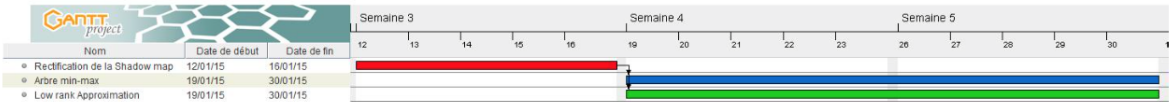

FIGURE 6 – Etape 2

Une fois la *Shadow map* calculée, nous pouvons lui appliquer la rectification épipolaire.

La *single scattering intégrale* : $L(\alpha, \beta) = \int_0^{D(\alpha,\beta)} V(\alpha, \beta, \gamma) I(\beta, \gamma) \mathrm{d}\gamma$ peut se découper en deux parties :

- le terme de la visibilité : $V(\alpha, \beta, \gamma)$
- tout les termes autres que la visibilité : $I(\beta, \gamma)$

Ainsi nous pourrons traiter en parallèle les deux fonctions suivantes :

- Approximation de I grâce à la décomposition en valeurs singulières
- Création d'un arbre min-max pour chaque ligne de la *Shadow map* rectifiée. Les arbres permettent de trouver rapidement quel segment est visible ou dans l'ombre.
- La création des fichiers *dot* peut s'effectuer en parallèle de la création des arbres car la structure des fichiers n'est pas propre à aux arbres min-max, mais fonctionnent pour tous les graphes

Etape 3

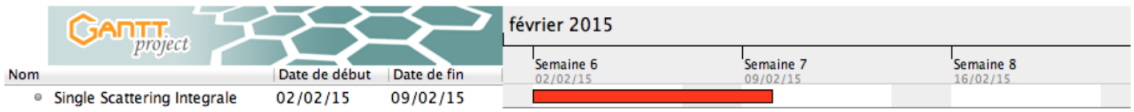

FIGURE 7 – Etape 3

Le calcul final de la luminance ne pourra être effectué une fois que toutes les tâches précédentes auront été implémentées et tester, car son implémentation en dépend fortement. En effet, ce calcul illustrera le résultat final de tous notre chef d'oeuvre, et avant d'y arriver, il faudra que toutes les fonctionnalités fonctionnent parfaitement. Ainsi, une fois que I est approximé et qu'il est rapide de trouver les segments éclairés grâce aux arbres, nous pouvons réunir les deux afin de calculer la luminance finale approximée grâce à la single scattering intégrale.

5.2 Tâches secondaires

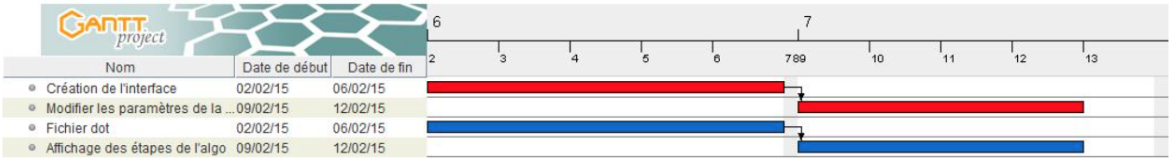

FIGURE 8 – Les tâches secondaires

La création de l'interface ainsi que la création des fichiers *dot* peuvent se faire en parallèle des tâches principales. En effet, cela ne nécessite pas le développement des tâches précédentes. Cependant, les deux autres étapes sont dépendantes des tâches principales et ne pourront être traitées qu'une fois les étapes précédentes validées.

5.3 Les risques

Lors de la conception du projet, le groupe devra faire face à des imprévus. Le but de cette partie est de prévoir les risques les plus probables et de trouver une solution afin de ne pas perdre plus de temps.

- **Mauvaise gestion du planning**
  Conséquence : risque de ne pas pouvoir implémenter toute les fonctionnalités
  Solution :
  - x revoir le planning à temps si le groupe perçoit qu'il ne pourra pas tenir les délais indiqué lors des premières versions du planning
  - x dans le pire des cas, mettre de côté les fonctionnalités les moins importantes (c'est-à-dire les taches secondaires indiqués dans la section précédente)
- **Impossibilité de terminer l'étape 1 (calcul des cartes de pronfondeurs) :**
  Conséquence : impossibilité d'achever les autres tâches
  Probabilité d'echec : cela parait peu probable puisque le rendu de la *Depth map* est fourni dans le code du client et nous avons déjà effectué plusieurs implémentations de test concernant le calcul de la *Shadow map*.

- **Impossibilité de terminer l'étape 2 (implémentation des étapes clés de l'algorithme) :**
  Conséquence : impossibilité d'implémenter l'étape 3
  Probabilité d'echec : la rectification épipolaire qu'on a prévu d'implémenter dans cette étape semble assez compliqué à première vue et à l'heure actuelle c'est l'étape qui nous semble la plus difficile !
  Solution :
  - x si le problème concerne la rectification épipolaire, aucune solution de secours n'est possible !
  - x si le problème concerne l'implémentation du *low-rank approximation*, utiliser une librarie pour calculer la SVD (par exemple la librairie Eigen)
  - x si le problème concerne la génération de l'arbre minmax, on pourrait tenter de continuer avec une approche plus naïve au risque d'augmenter considérablement le temps de calcul (l'arbre étant une structure permettant d'accélérer l'algorithme) !

- **Impossibilité de terminer l'étape 3 (calcul final de la scène dans un milieu participant) :**
  Conséquence : impossible d'avoir le rendu attendu !
  Probabilité d'echec : une fois que toutes les étapes précédentes ont été implémentées, le calcul final assemble simplement ces étapes. Cependant cette étape est prévu assez tardivement dans le planning, en cas de retard dans les étapes précédentes cette étape finale pourrait ne pas être implémenté à temps.
  Solution : se rabattre rapidement sur une implémentation moins réalise mais plus facile (comme le premier papier de notre bibliographie) si il reste assez de temps

- **Étape non validée par le client :**
  Conséquence :
  - x perte de temps
  - x implémenter de nouveau l'étape (entièrement ou une partie)

  Solution :
  - x communiquer avec le client dès que le groupe a un doute sur un souhait du client
  - x devancer les questionnements qui peuvent se poser lors de l'implémentation pendant les rendez vous avec le client

Références

[1] Chen. Real-time volumetric shadows using 1d min-max mipmaps. 2011.

[2] Kajiya and Herzen. Ray tracing volume densities. *SIGGRAPH 84*, 1984.

A ANNEXE - Planning général

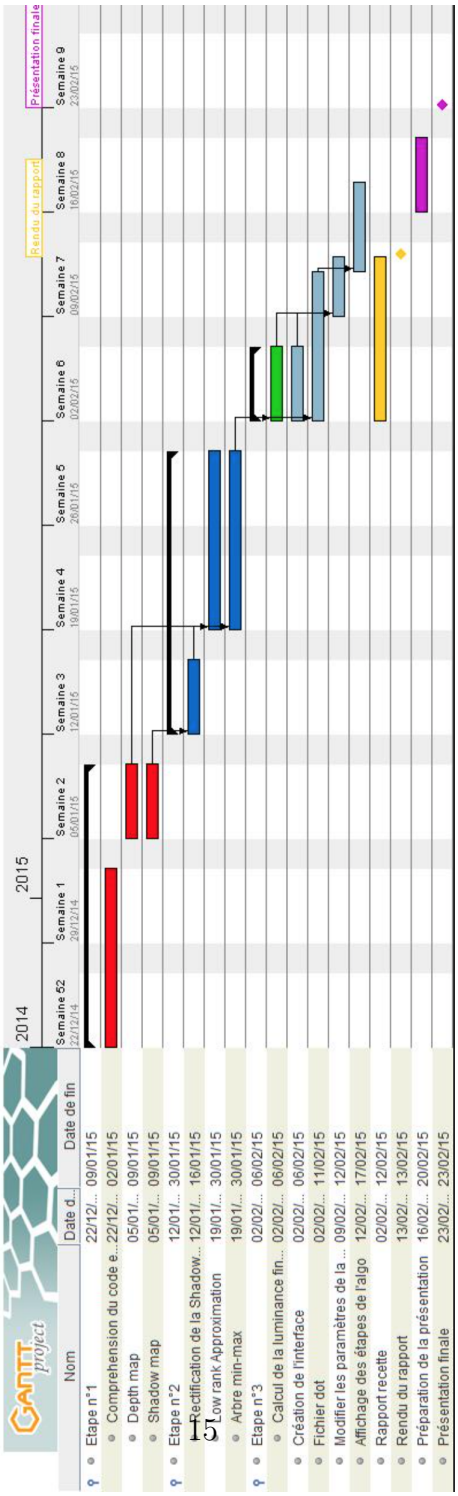

FIGURE 9 – Planning général du chef d'oeuvre