Intégration d'une méthode de mélange de couleurs physiquement réaliste dans un logiciel de simulation de peinture

Nicolas Barroso Valentin Chevrier Hugo Lévêque Ilyes Naidji Moearii Teraitetia

Client: David Vanderhaeghe
Université Paul Sabatier

Décembre 2016

1 Introduction

Ce second rapport rentre dans le cadre du Chef d'Oeuvre proposé lors du Master Informatique Graphique et Analyse d'Images. Nous avons été chargés par notre client, monsieur David Vanderhaeghe, d'implémenter une solution au mélange de pigments dans un moteur de simulation de peinture réaliste.

Ce rapport a pour but d'exposer les objectifs ainsi que le planning prévu pour la réalisation de ce projet. Pour cela nous commencerons par présenter le cahier des charges en précisant toutes les fonctionnalités que notre programme devra offrir. A partir de cette analyse fonctionnelle, nous pourrons découper notre projet en tâches et en modules. Cela nous permettra de mieux analyser les dépendances entre les différentes tâches à accomplir et donc de prévoir en amont les risques que nous pourrons rencontrer lors du développement.

2 Cahier des charges

2.1 Objectifs

L'objectif principal de ce projet est d'implémenter un prototype de mélange de couleurs physiquement réaliste s'appuyant sur la méthode de Kubelka-Munk en proposant à l'utilisateur une palette de pigments à mélanger.

Une fois le premier objectif atteint, nous intégrerons le prototype dans un logiciel de simulation de peinture déjà existant plus complexe.

2.2 Exigences

Le cahier des charges décrit avec précision les besoins du client en termes de fonctionnalités. A chacune de ces fonctionnalités, nous avons attribué un niveau de priorité (voir Annexe 1 p.5).

2.2.1 Exigences fonctionnelles

- ❖ Priorité de niveau 1 :
  - EF1 : L'application doit utiliser la méthode de mélange de pigments décrite dans l'article de Baxter et col.[1].
  - EF2 : Le mélange de couleurs doit être contrôlé par les coefficients d'absorption et de diffusion des pigments à mélanger.
- ❖ Priorité de niveau 2 :
  - EF3 : L'application doit permettre à l'utilisateur de définir les coefficients d'absorption et de difusion à l'aide d'une interface graphique.
- ❖ Priorité de niveau 3 :
  - EF4 : L'application doit permettre la visualisation du mélange de couleurs en temps réel.
  - EF5 : L'application doit proposer à l'utilisateur une gamme de pigments avec des coefficients d'absorption et de diffusion définis pour le mélange.

2.2.2 Exigences logicielles

- ❖ Priorité de niveau 1 :
  - EL1 : L'application doit être codée en C++.
  - EL2 : L'application doit utiliser l'A.P.I OpenGL.
  - EL3 : Tous les calculs concernant le mélange de couleurs doivent pouvoir s'effectuer sur CPU et GPU.
- ❖ Priorité de niveau 2 :
  - EL4 : L'interface graphique doit être codée en utilisant QT.
- ❖ Priorité de niveau 3 :
  - EL5 : L'application doit être intégrée dans le moteur de simulation de peinture du client.

3 Découpage en tâches

3.1 Description des travaux par tâches

Suite à la description des différentes fonctionnalités que doit offrir notre application, nous avons extrait les principales tâches à réaliser :

1. Intégrer des bases de données de spectres d'absorption et de diffusion de pigments dans notre application.
2. Intégrer les librairies concernant la manipulation et la représentation de spectre.
3. Réaliser une maquette de l'interface graphique.
4. Développer l'interface graphique QT.
5. Développer le contexte OpenGL.
6. Développer le mélange selon Kubelka-Munk sur CPU puis sur GPU.
7. Calculer le spectre de réflectance du mélange sur CPU puis sur GPU.
8. Convertir le spectre de réflectance dans l'espace XYZ sur CPU puis sur GPU.
9. Convertir en sRGB sur CPU puis sur GPU.
10. Appliquer une correction Gamma sur CPU puis sur GPU.
11. Réaliser des tests de validation.
12. Comparer les résultats.

Les tâches 6,7,8,9 et 10 font partie du pipeline de calcul et sont par conséquent dépendantes chronologiquement entre elles au niveau des tests d'intégration. On ne pourra pas tester la tâche 10 avant d'avoir testé et vérifié la cohérence des résultats obtenus par la tâche précédente 9, etc... jusqu'à la 6ème tâche.

Une fois ces tâches réalisées, nous devrons nous intéresser à l'intégration de ces fonctionnalités au moteur de rendu de peinture réaliste déjà existant. Il faudra alors :

13. Analyser le code existant.
14. Intégrer les modules Noyau et Calcul dans le moteur.
15. Développer un widget permettant à l'utilisateur de sélectionner et de mélanger des pigments à l'aide d'une palette de couleurs.

3.2 Analyse des risques

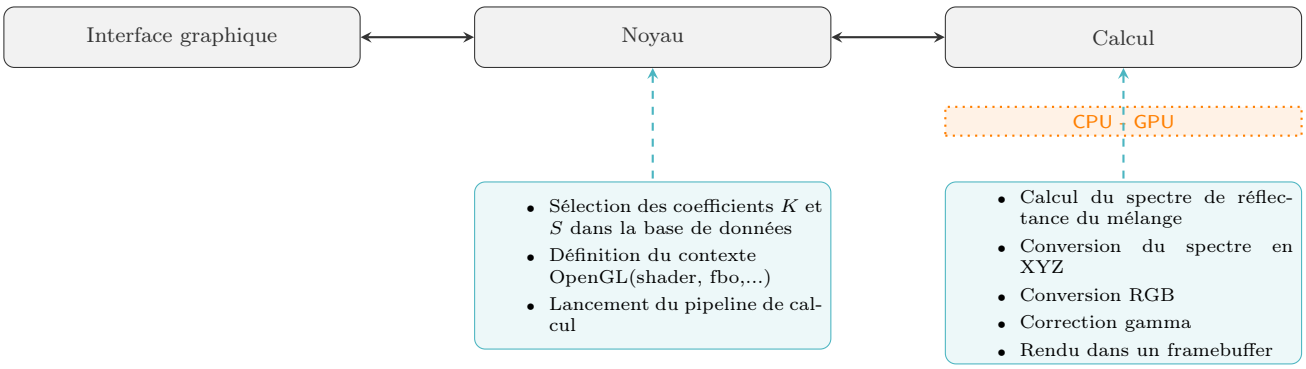
Pour chaque tâche nous avons identifié les risques éventuels et les solutions à apporter.

n°	Description	Impact	Probabilité	Solution
1	Nous ne trouvons pas de base de données de spectres K et S de pigments.	faible	moyen	Donner à l'utilisateur un moyen de rentrer des valeurs de K et S (via l'interface)
2	Nous ne trouvons pas de librairies de manipulation et de représentation de spectre en C++.	faible	moyen	Définir et créer une structure de représentation et manipulation de spectre.
6-10	Une tâche pose problème et corrompt le pipeline de calcul.	important	moyen	Mettre en place une procédure de debug ou faire appel à un consultant.
11	Les tests de validation ne sont pas concluants.	critique	faible	Revoir les spécifications et les méthodes.
14	Notre module NOYAU est incompatible avec le moteur de simulation de peinture.	faible	moyen	Créer un wrapper afin de faire correspondre les entrées et sorties des modules en communication.

4 Découpage en modules

Notre application sera composée de plusieurs modules :

- Module IG : Un module gérant l'interface graphique. Il devra permettre à l'utilisateur de définir les pigments à mélanger et afficher la couleur résultante.
- Module NOYAU : Ce module sera chargé de gérer les appels de l'interface graphique afin de lancer les processus correspondants et de retourner les résultats obtenus via le module CALCUL. En outre, il devra initialiser le contexte OpenGL, à savoir les shaders et les FBO(Frame Buffer Objects). Il sera donc capable de lancer le pipeline de rendu tout en sélectionnant les coefficients K et S adéquats dans la base de données.
- Module CALCUL : Un module gérant le pipeline de calcul. Nous le développerons d'abord dans l'optique d'effectuer les calculs sur CPU avant de les effectuer sur GPU (calcul du spectre de réflectance du mélange, calcul de la conversion XYZ, conversion RGB, correction gamma et rendu dans un frameBuffer).



5 Tests de validation

Afin de vérifier l'adéquation de notre application aux besoins de notre client, nous mettrons en place une série de tests de validation. Ces tests vérifieront la correspondance entre la spécification de chaque fonctionnalité prévue et le résultat obtenu.

❖ **Test de la fonctionnalité EF1** :
Dans un premier temps nous effectuerons différents mélanges de peinture réelle. Chaque mélange obtenu sera numérisé avec un capteur numérique. Ensuite nous comparerons les mélanges numérisés avec les mélanges obtenus via notre application. Il est primordial que ces comparaisons s'effectuent sur le même moniteur d'affichage afin que les deux mélanges à comparer subissent la même déformation à l'affichage. On pourra par exemple prendre

en référence les mélanges réalisés dans l'article de Baxter et col.[1].

❖ **Test de la fonctionnalité EF2** :
Afin de tester si le calcul du mélange est contrôlé par les coefficients d'absorption et de diffusion de ces pigments, nous ferons varier ces coefficients. Si le mélange résultat diffère à chaque variation des coefficients, alors le test sera validé.

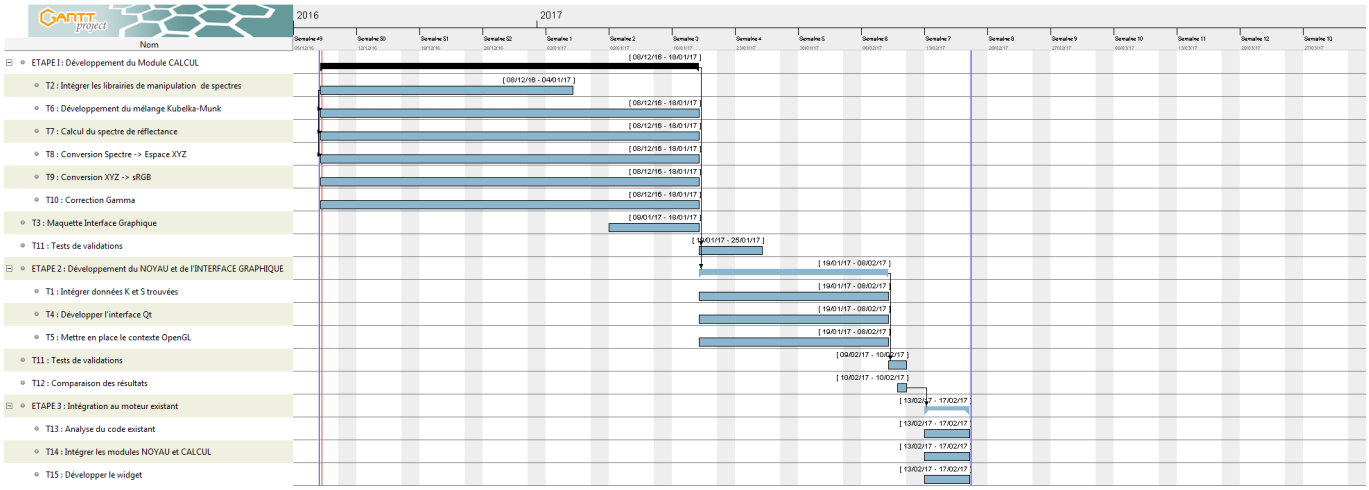
❖ **Test de la fonctionnalité EF3** :
Pour valider cette fonctionnalité, il suffit de demander à l'utilisateur de notre application de lancer le calcul du mélange après avoir sélectionné les pigments à mélanger. En sélectionnant des pigments pré-définis, l'utilisateur sélectionnera les coefficients d'absorption et de diffusion du pigment associé indirectement.

❖ **Test de la fonctionnalité EF4** :
Lorsque l'utilisateur lance le calcul du mélange de couleur via l'interface graphique, il ne doit pas percevoir de latence entre le moment où le calcul est lancé et le moment où le résultat du mélange s'affiche.

❖ **Test de la fonctionnalité EF5** :
On considère que le test est réussi, si l'utilisateur peut choisir 5 pigments ou plus pour le mélange de couelurs à partir de l'interface graphique.

6 Planning prévisionnel

Nous avons décidé de séparer le sujet en trois étapes. La première regroupera l'implémentation des différentes méthodes à mettre en oeuvre pour effectuer le mélange des pigments. Nous nous concentrerons dans un deuxième temps, une fois les tests de validations effectués, sur la réalisation d'un prototype permettant de réaliser aisément des mélanges de pigments. Enfin, si nous n'avons pas pris trop de retard, nous nous pencherons sur l'adaptation de notre méthode pour l'intégration dans le moteur du client.



Annexe 1

Définition des impacts des risques sur le projet

Impact	Description
Critique	Le risque peut compromettre une ou plusieurs fonctionnalités de notre application.
Important	Le risque peut engendrer un retard important lors de la livraison de nos livrables.
Faible	Le risque peut nécéssiter une réorganisation des moyens accordés aux différentes tâches du projet.
Nul	Le risque n'a pas d'impact sur le projet.

Définition des probabilités des risques sur le projet

Probabilité	Description
Très forte	Est quasi certain
Forte	A de fortes chances de se produire
Moyenne	Peut apparaître
Faible	Improbable

Définition des niveaux de priorité des exigences

Niveau	Description
1	Cette exigence ne peut pas être ignorée sans quoi la recette ne pourra pas être validée.
2	Cette exigence aura un impact important lors de la validation de la recette, elle doit être traitée avant les exigences de priorité inférieure.
3	Cette exigence aura un impact limité sur la validation de la recette. Elle est demandée mais reste secondaire par rapport aux exigences de niveau supérieur.

Références

[1] William Baxter, Jeremy Wendt, and Ming C Lin. Impasto : a realistic, interactive model for paint. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 45–148. ACM, 2004.