Logiciel d'expérimentation pour les textures multi-échelles

HAEHNEL Jonathan

Projet 150h - M2 ISI - Université de Strasbourg

19 décembre 2013

Plan de la présentation

La méthode : On-the-Fly Multi-scale Infinite Texturing (1)

- Développée par l'équipe IGG
- Texturage de grandes scènes en temps réel
- Deux aspects : Texturage infini et texturage multi-échelle

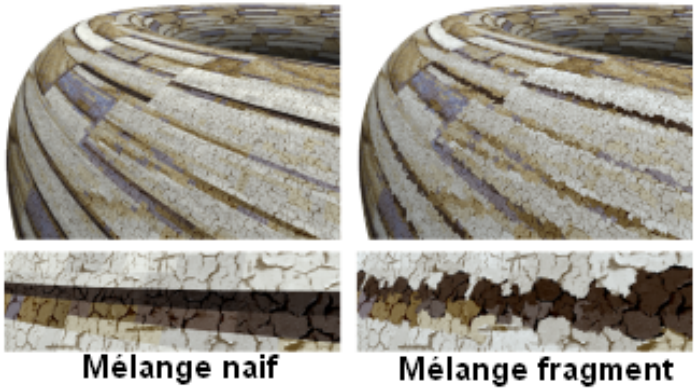
Texturage infini

- Répétition d'une texture d'entrée sur la scène
- $\notin$ dans le cadre de mon application

La méthode : On-the-Fly Multi-scale Infinite Texturing (3)

Texturage multi-échelle

- Deux textures d'entrées à deux échelles différentes
- Mélange astucieux entre ces deux textures selon la distance de visualisation de la scène
- Respect des motifs des deux textures → rendu réaliste
- Mélange naif → flou et transparence peu naturel

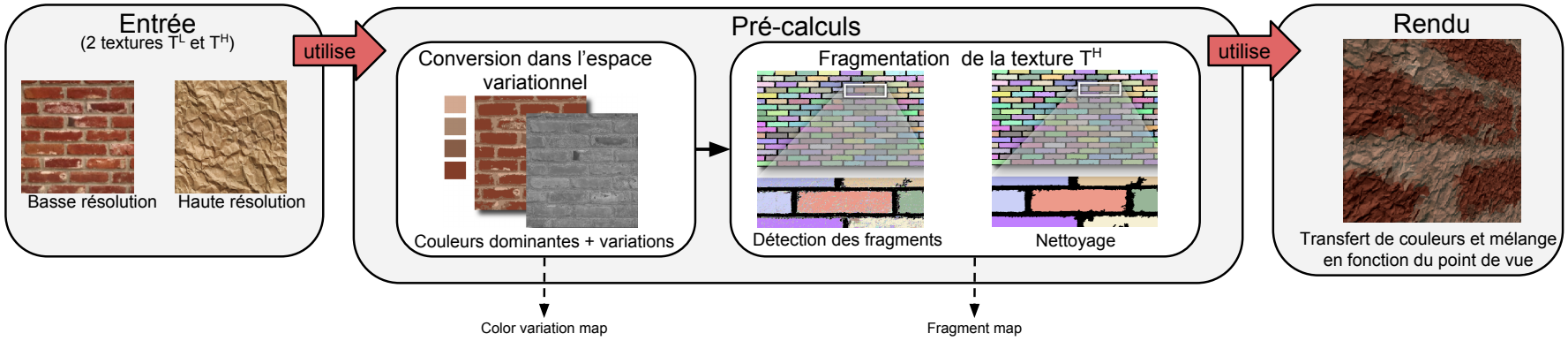


Chaîne de traitement

- Découpage en **trois phases séquentielles**

→ Entrée : deux textures à des échelles différentes

→ Sortie : une texture multi-échelle en fonction du point de vue



Problématique

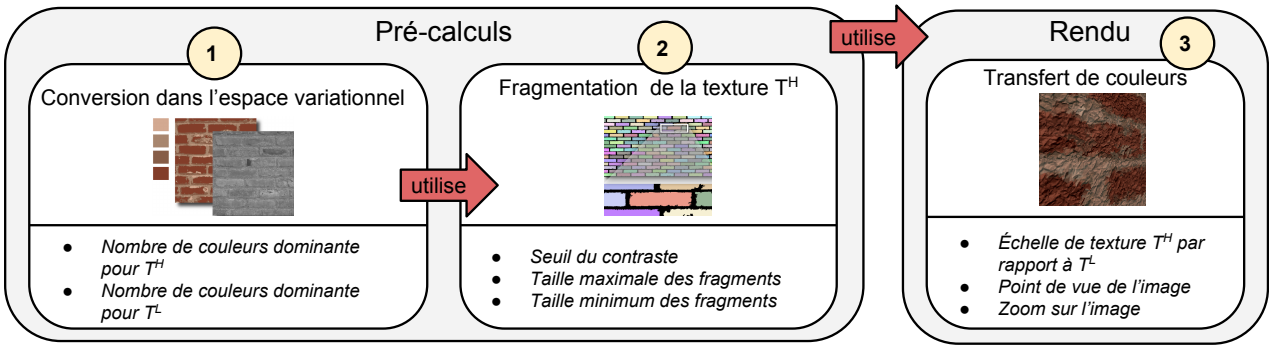
Problématique générale

- Effectuer des tests de notre méthode

- Chaque phase est actuellement réalisée dans un logiciel différent
- Allers-retours entre les différents logiciels et interfaces

→ Besoin d'uniformisation

- Un grand nombre de paramètres à prendre en compte



Objectifs du projet

1. Une application fonctionnelle permettant de tester et paramétrer notre méthode de texturage multi-échelle
2. Une interface uniforme et ergonomique pour chacune des phases
3. Une application évolutive et modulable

L'existant

Pourquoi me fournir certaines parties en boite noire ?

- Gain de temps important
- Se focaliser sur l'interopérabilité entre les phases et sur la GUI

Travail	Personnel	Existant
Phase 1	Algorithme de K-means	Divers algorithmes (PCA, Quantification initiale)
Phase 2	-	Algorithme de fragmentation (croissance de région)
Phase 3	Mélange de couleurs	-

Les structures de données (1)

Langages et librairies utilisés

- C++
- Qt pour les interfaces graphiques

- Tout au long de la pipeline, un grand nombre de données générées
- Nécessité de stocker ces données

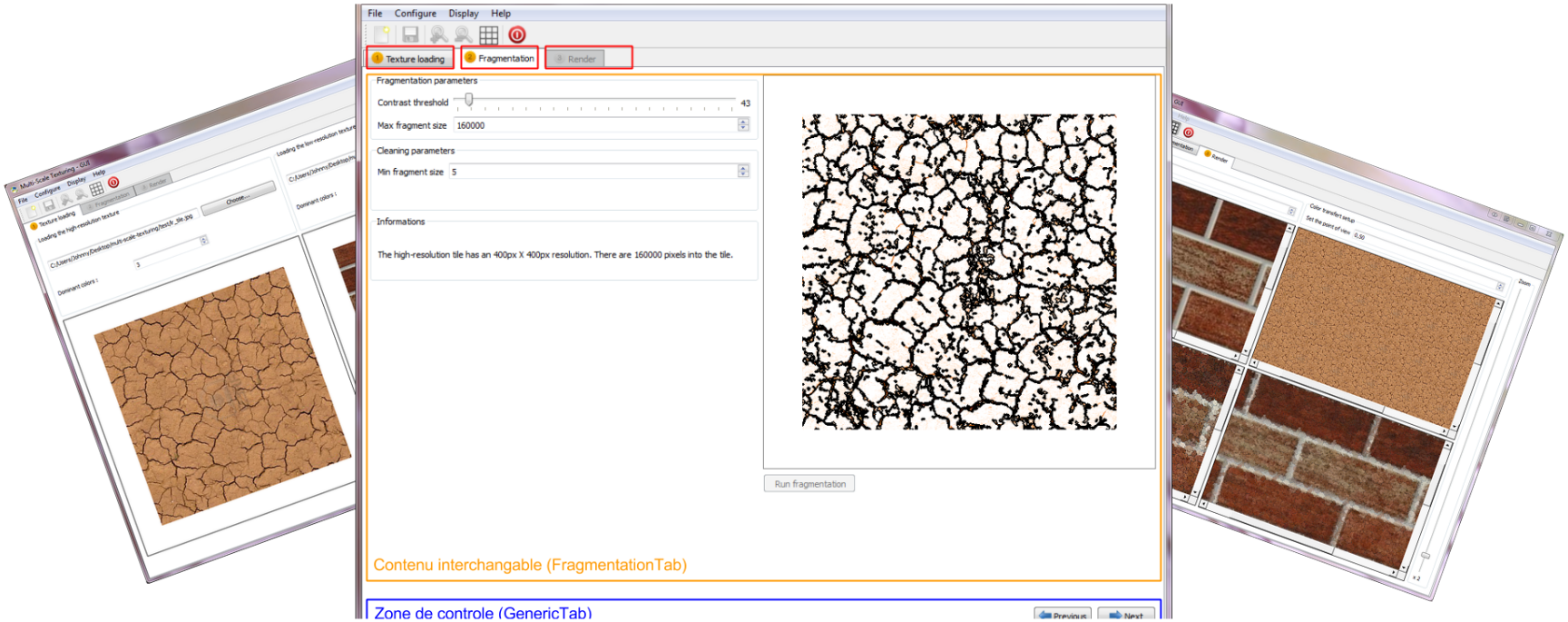
→ Structures et classes cohérentes et modulables

Les structures de données (2)

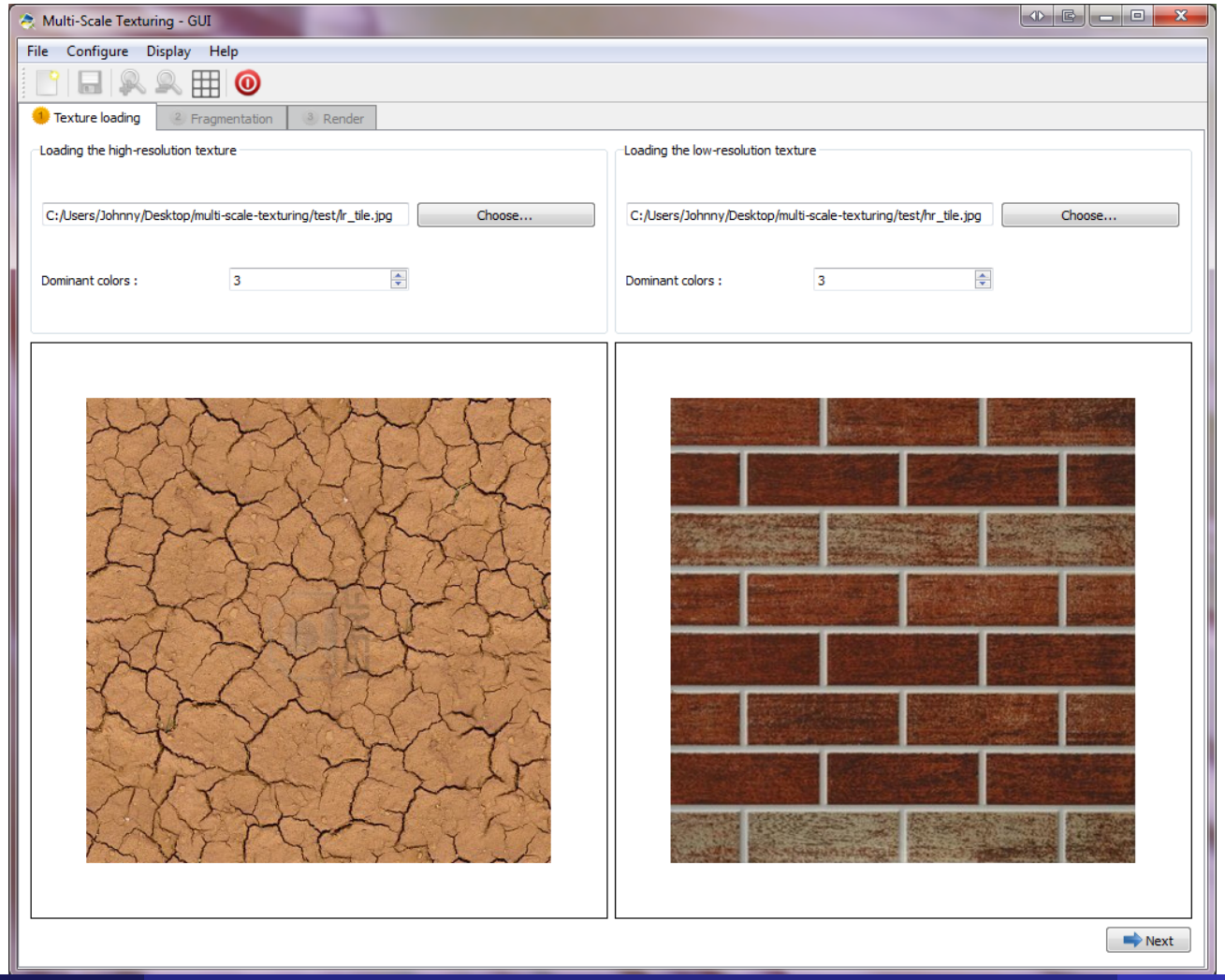
- Chaque pixel de la texture est stockée sous 3 formes différentes.
  - Un triplet d'entiers RGB
  - Un entier (couleur dominante) et une flottant (variation locale)
  - Un vecteur 2D des coordonnées vers le centre du fragment.
- Matrice<T> générique pour instancier la matrice de son choix
- Classe conteneur permettant de regrouper les trois matrices

L'interface graphique

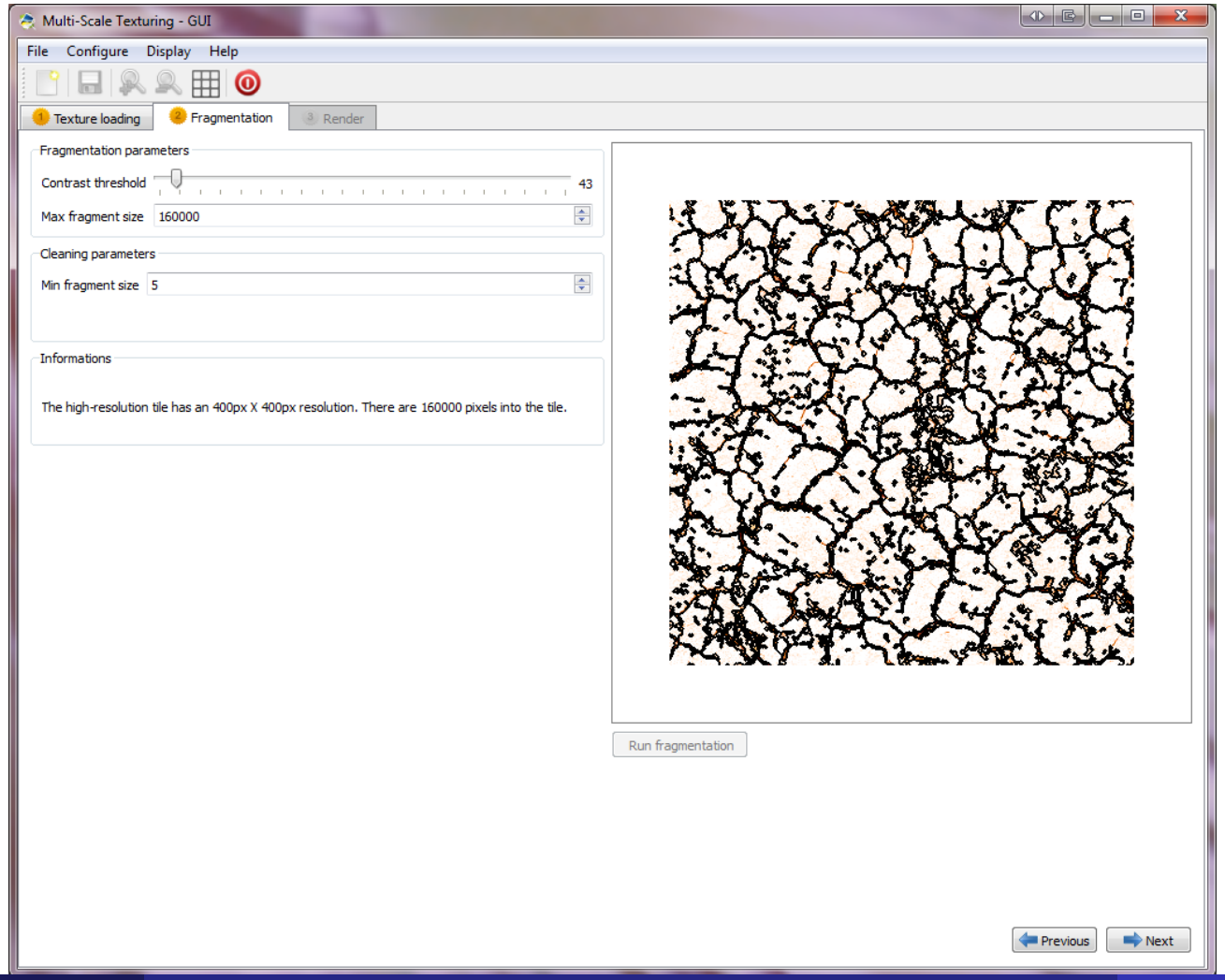
- Interface sous forme "installeur de programme"
- Chaque onglet hérite d'un onglet générique → évolutivité



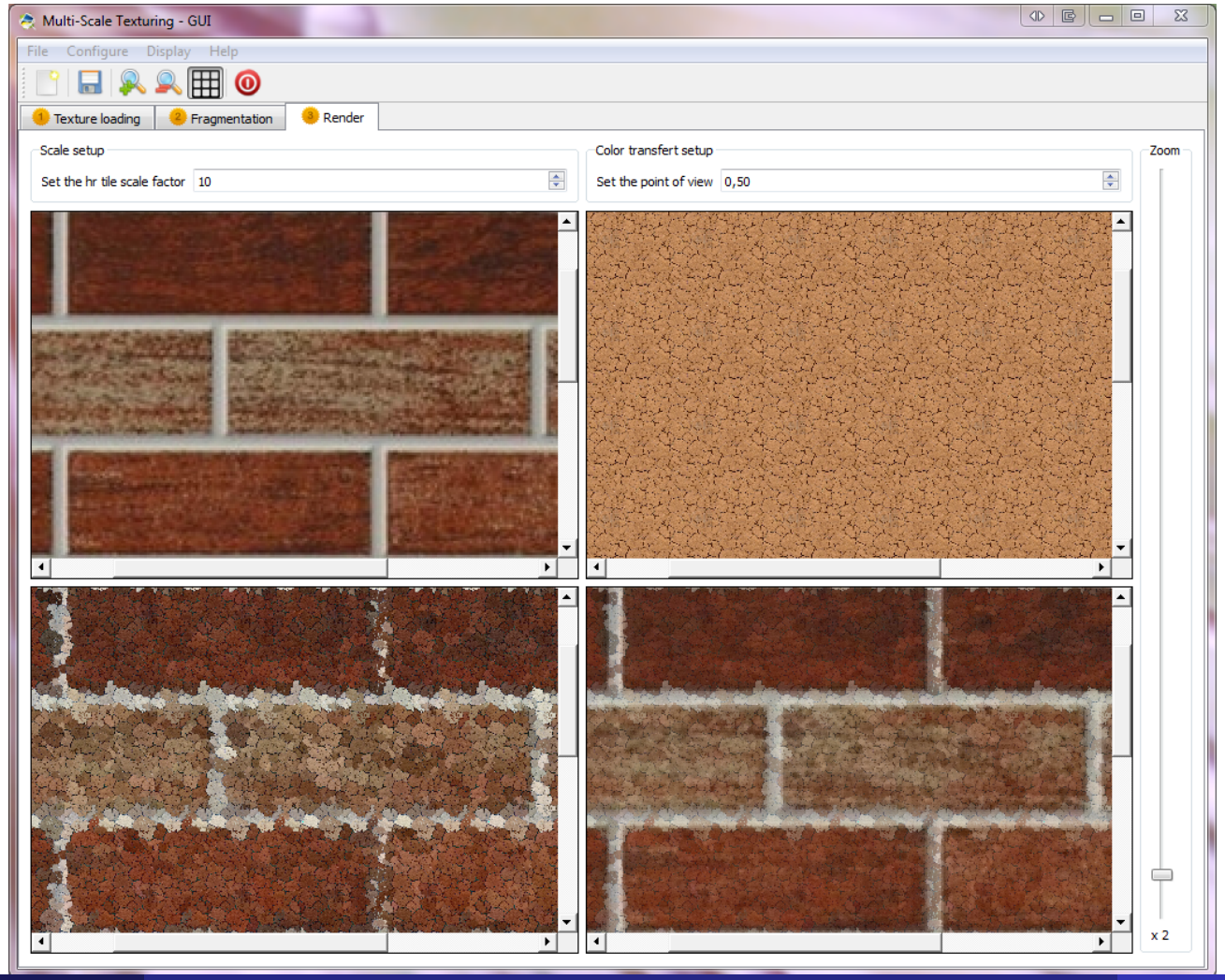
L'interface graphique : segmentation en couleur dom. (1)



L'interface graphique : fragmentation (2)



L'interface graphique : rendu (3)



Conclusion

- Application fonctionnelle et évolutive (Doxygen, UML et dossier technique)
- Des améliorations possibles

Perspectives d'évolution

1. Accepter des images d'entrées de tailles quelconques
   - Plus grande flexibilité dans les tests
2. Passer le rendu final en temps réel
   - Utilisation du GPU (OpenGL, OpenCL, ...)
   - Gain en interactivité pour l'utilisateur

Fin de la présentation

Merci de votre attention ! Des questions ?