Interface Graphique pour exploiter un code Eléments Discrets (IGED)

Harold Trannois, Jérôme Fortin, Patrice Coorevits, Mohamed Guessasma, Emmanuel Bellenger, C Cogné

Interface Graphique pour exploiter un code Eléments Discrets (IGED)

H. Trannois[1], J. Fortin[1], P. Coorevits[2], M. Guessasma[2], E. Bellenger[2], C. Cogné[2]

Université de Picardie Jules Verne - Laboratoire des Technologies Innovantes EA 3899
[1] INSSET, 48 rue Raspail, 02100 Saint-Quentin, France
{harold.trannois,jerome.fortin}@u-picardie.fr

[2] IUT de l'Aisne, 48 rue d'Ostende, 02100 Saint-Quentin, France
{patrice.coorevits,mohamed.guessasma,emmanuel.bellenger}@u-picardie.fr

Résumé — L'objectif de ce papier est de proposer un outil de visualisation graphique de résultats de calculs effectués à l'aide d'un code Eléments Discrets : **I**nterface **G**raphique pour code **E**léments **D**iscrets (**IGED**). Il s'agit d'un logiciel de post-processing écrit en C^{++} basé sur des librairies open source portables, permettant à **IGED** d'être compatible avec différents OS (Windows, Linux, Unix, MacOS,...).

Mots clés — MED ; Interface Graphique ; Opensource ; HDF5 ; OpenSceneGraph.

1 Introduction

Dans le domaine de la simulation numérique en mécanique et plus généralement des sciences de l'ingénieur, la Méthode des Eléments Finis (**MEF**) est largement utilisée. Ainsi, tous les logiciels mettant en œuvre cette méthode proposent des outils évolués de post-traitement et de visualisation des résultats. Ces outils ont atteint aujourd'hui une grande maturité ce qui a participé avec les progrès de l'informatique et en parallèle des aspects théoriques, au développement de la MEF dans l'industrie et la recherche. Depuis quelques années, l'utilisation de plus en plus importante de la Méthode d'Eléments Discrets (MED) dans de nombreux domaines [1], nécessite un travail important pour proposer des outils équivalents à ceux trouvés dans la plupart des codes Eléments Finis. Ce travail sur la visualisation et le post-traitement des résultats doit participer au développement de l'utilisation de la MED. L'objectif de ce papier est de proposer un outil de visualisation graphique de résultats de calculs effectués à l'aide d'un code Eléments Discrets [2] : **I**nterface **G**raphique pour code **E**léments **D**iscrets (**IGED**). Il s'agit d'un logiciel de post-processing écrit en C^{++} basé sur des librairies open source portables, permettant à **IGED** d'être compatible avec différents OS (Windows, Linux, Unix, MacOS,...). Le logiciel fonctionne en deux temps : traitement modulaire des Eléments Discrets provenant des résultats du calcul puis affichage et manipulation de tous les éléments. Le flux de données entre le code Eléments Discrets et l'IGED est converti en Hierarchical Data Format (HDF) [3], au même titre que le codage vidéo, codée les données en HDF est nécessaire pour un traitement fluide. La qualité des librairies a été aussi déterminante que ce soit par exemple HDF pour la manipulation des fichiers de données ou OpenSceneGraph (OGS) [4] pour la gestion des scènes 3D en OpenGL. Pour finir, l'interface graphique a été développée à l'aide du toolkit WxWidget [5] portable sur tous les OS.

2 Architecture

IGED a été conçu pour être le plus ouvert possible en s'appuyant sur des librairies reconnues et libres. Nous avons porté une attention toute particulière à la documentation du code et à l'environnement de développement par un "travail collaboratif". Les outils utilisés sont tous issus de l'open source. Le résultat est une application reposant sur l'idée d'un traitement de flux à l'aide de filtres. Cette architecture permet de découpler entièrement **IGED** du code de calcul ED, (Figure 1).

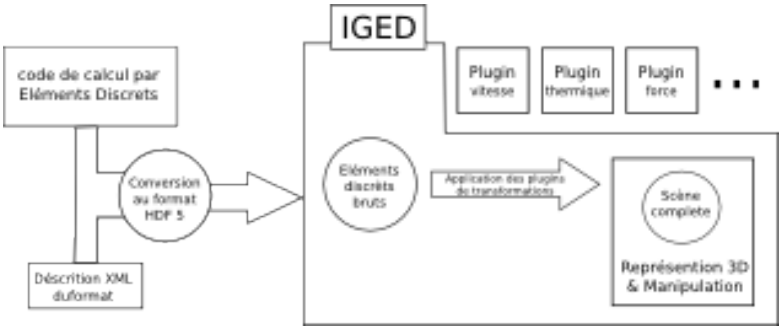

Figure 1 – Architecture de l'Interface Graphique IGED

IGED repose sur une organisation arborescente des éléments discrets à représenter. On retrouve cette notion dans le stockage des données au format HDF et aussi dans l'organisation de la représentation graphique OSG. La taille des données à traiter étant conséquente, il n'est pas raisonnable de tout stocker en mémoire ; pour lire un film on ne le charge pas entièrement en mémoire mais on lit séquenciellement, petits morceaux par petits morceaux. Organiser les données en arbre, va permettre de charger en mémoire les données branche par branche.

2.1 Lecture des fichiers de données

La transformation repose sur un fichier XML décrivant l'organisation des données et un outil de conversion (inclus dans IGED). Ainsi, on transforme avant utilisation tous fichiers de données résultats au format HDF. Le Format HDF a été adopté comme le format natif d'IGED ; il existe depuis relativement longtemps et a été développé pour manipuler de grand volume de données. La lecture et l'écriture sont très rapides grâce notamment à l'utilisation de divers drivers permettant d'exploiter différents types d'architecture : simple fichier sur un système de fichier standard, plusieurs fichiers sur un système de fichier standard, plusieurs fichiers sur un système de fichiers parallèles et d'autres situations. La première version de l'organisation des données dans le fichier HDF propose l'architecture suivante (Figure 2) :

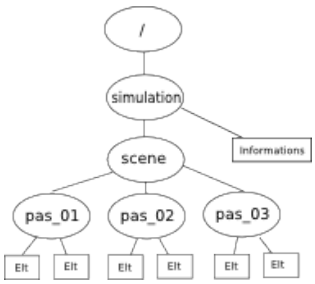

Figure 2 – Arbre HDF

En partant de la racine '/' on accède à la simulation et ses données globales, puis on accède

à chaque scène et pour terminer au bloc de données pour chaque type d'éléments discrets. Dans un fichier HDF il y a des ensembles de données appelés des DATASET qui sont réunis dans des groupes appelés GROUP. Chaque GROUP ou DATASET possède un nom, c'est le nom du DATASET qui va déterminer la représentation d'un élément discret. L'arbre représentant l'organisation des données ressemble à celui de la Figure 2. L'accès au premier DATASET se fait par le $path/scene/pas_01/$. Les données ne sont pas seulement organisées sous forme d'arbre dans les fichiers, mais aussi dans la mémoire. OSG organise aussi les différents éléments d'une scène 3D sous forme arborescente et même un peu plus sous forme de graphe. IGED réutilise les mécanismes de OSG pour cette partie, les éléments discrets de IGED sont des spécialisations des GROUP d'OSG. Lors de la lecture des données on construit l'arbre de la scène pour chaque pas de simulation, Figure 3.

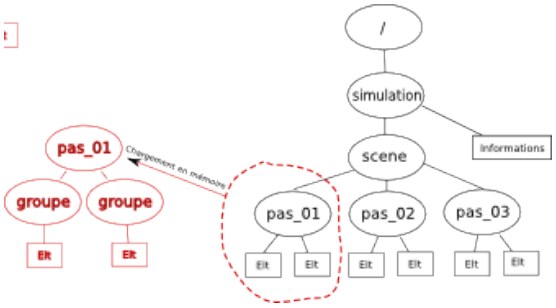

Figure 3 – Représentation en mémoire

2.2 Documentation

IGED utilise un outil de suivi de projet "Trac" visible à l'adresse suivante $http://iged.insset.u-picardie.fr/$, Figure 4. Trac est un système Open Source de gestion complète de projet par Internet, développé en Python. Trac inclut : un Wiki, une Gestion de feuilles de route, un Historique, un Rapport de bugs, un Explorateur subversion. L'affichage Web de Trac fonctionne grâce au moteur de template ClearSilver.

Figure 4 – Trac d'IGED : http ://iged.insset.u-picardie.fr/

3 Exploitation

La visualisation des phénomènes mécaniques est la partie la plus importante. **IGED** reprend les principales possibilités des logiciels 3D : plusieurs vues simultanées, rotations, déplacements. On a, cependant, ajouté la possibilité de suivre un corps dans le temps et de tracer les courbes associées : on peut ainsi à tout moment afficher les informations de ce corps. Actuellement l'interface permet :

- Affichage 3D - des isovaleurs - en mode filaire / faces cachées ;
- Rotation, zoom, translation dynamique à l'aide de la souris ;
- Possibilité de "découper" la structure pour examiner un champ dans les parties non visibles ;
- Enregistrement aux formats mpeg pour les films, jpeg pour les images et svg pour les données exploitées.

Dans IGED un élément discret est une entité insécable, c'est l'élément unitaire. Il est ainsi possible de le sélectionner, de le manipuler, de le tracer, de le décorer... Cette approche est très différentes des représentations par maillage. Plus en adéquation avec le monde granulaire, IGED va permettre d'observer et de disséquer le milieu expérimental virtuel et d'en extraire les caractéristiques afin de les comparer à la théorie (Figure 5) :

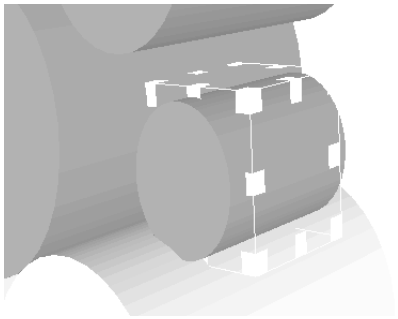

Figure 5 – Sélection d'un élément

Le dessin d'un élément dans IGED se fait via des plugins, cela permet de disposer d'une bibliothèque de représentation adaptée à chaque phénomène étudié. Par défaut le nom du DATASET est le nom du plugins ce qui donne, par exemple, pour 2 éléments sphériques se rapprochant, de la structure de la Figure 6.

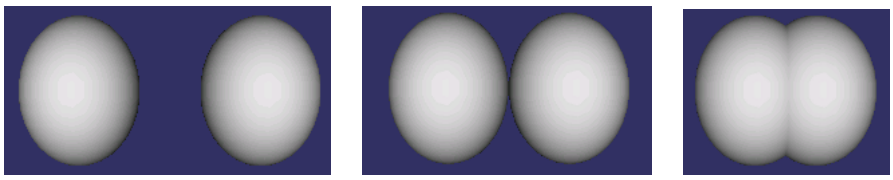

Figure 6 – Exemple de représentation de 2 éléments sphériques

Dans cet exemple, on a utilisé le plugins Sphere, si on change juste le nom des DATASET en Diamant par exemple qui est une représentation sous forme d'octaèdre, on obtiendra la Figure 7.

On voit qu'on peut très simplement changer la représentation. Les décorateurs sont de petits dessins que l'on va rajouter sur quelques ou tous les éléments permettant de matérialiser graphiquement les données (vitesse, force, température) ou des marqueurs permettant de suivre tout au long de la simulation des éléments particuliers (à condition que le code de calcul affecte une identi-

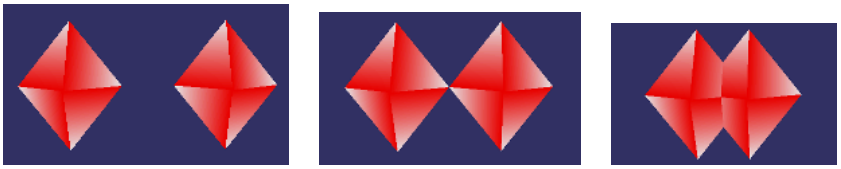

Figure 7 – Exemple de représentation de 2 éléments Diamants

fication aux éléments). Comme exemple, on a représenté un vecteur vitesse sur nos deux éléments Figure 8.

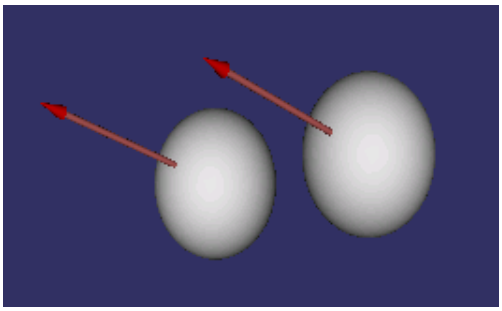

Figure 8 – Représentation du vecteur vitesse

L'implémentation des décorateurs repose sur le Design Pattern du même nom. Elle permet de modifier l'arbre représentant la scène en ajoutant des feuilles et des noeuds. Le Décorateur ou les Décorateurs sont 'plus hauts' dans la hiérarchie que l'élément décoré, mais ils font tous partie du même GROUP, Figure 9.

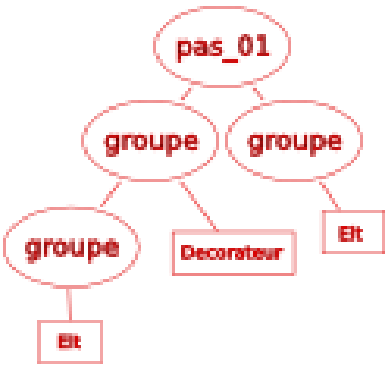

Figure 9 – Représentation du décorateur

L'activation d'un plugin entraîne une modification dans la représentation du corps. En d'autres termes, on attache dynamiquement des responsabilités à l'objet corps en utilisant le pattern Décorateur. Les modifications sont cumulables les unes avec les autres. On peut activer, par exemple, la représentation des forces avec celle des températures, Figure 10.

Concernant l'exploitation des données, elle est centralisée et les courbes peuvent être affichées en parallèle de la représentation graphique. On peut afficher l'évolution du coefficient de frottement global, les températures, la pression du fluide, l'orientation des contacts... Le paramètrage des figures se fait via des interfaces *adhoc* pour répondre au mieux aux exigences les plus diverses en termes de tailles et de formats.

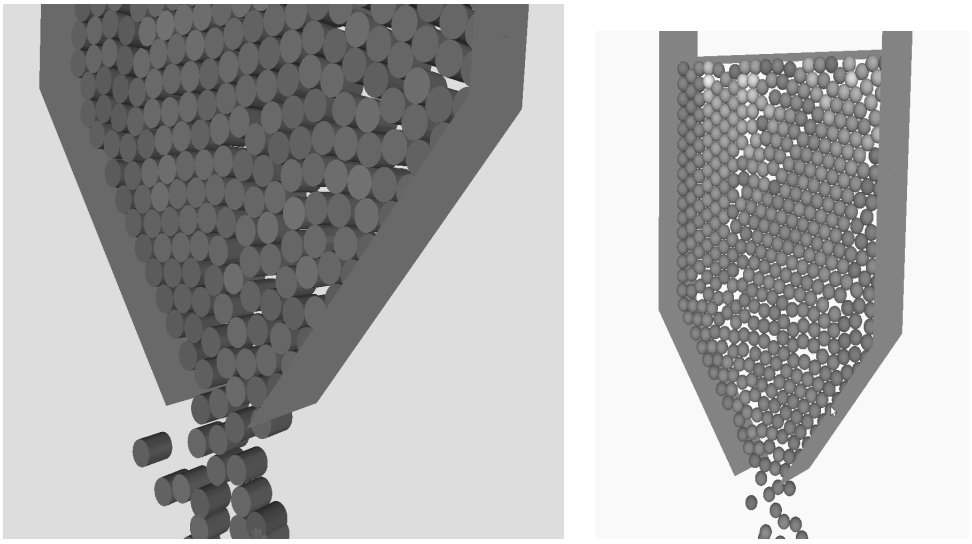

Figure 10 – Exemple de Visualisation avec IGED

4 Conclusion

Le langage C^{++} a été retenu pour le développement. Ce langage compilé permet d'exploiter pleinement les capacités d'*OpenGL* des cartes graphiques, ce qui est nécessaire pour le traitement des scènes 3D riches en corps. De plus, les librairies, telles que OSG, étant elles aussi écrites en C++, on a pu les exploiter et les réutiliser pleinement via l'héritage. L'organisation arborescente des données dans le fichier HDF a permis l'implémentation d'un cache, celui-ci est géré à l'aide de thread dont le nombre paramétrable permet jusqu'à maintenant d'avoir des animations fluides. IGED a un cache threadsafe et OSG est aussi threadsafe. L'interface graphique a été développée à l'aide du toolkit WxWidget portable sur tous les OS. C'est un outil de visualisation qui permet d'exploiter au mieux les données issues d'un code Eléments Discrets. Il fournit une représentation qui vient étayer le discours du chercheur. Pour finir, le respect des *design patterns* lors du développement, doit permettre une prise en main relativement rapide par d'autres développeurs. La présentation de l'**IGED** s'appuiera sur diverses applications numériques issues du code ED MULTICOR développé au Laboratoire des Technologies Innovantes et du code LMGC90 développé au LMGC de Montpellier. **Remerciements :** nous remercions Frédéric Dubois du LMGC (UMR 5508) pour les nombreuses discussions et propositions pour développer IGED.

Références

[1] F. Dubois, M. Jean, Lmgc90 une plateforme de développement dédiée à la modèlisation de problèmes d'intéraction. *6 ème colloque national en calcul des structures*, page 111–page 118, 2003

[2] J. Fortin, O. Millet, G. de Saxcé, Numerical simulation of granular materials by an improved discrete element method *Int. J. Numer. Meth. Engng*, 62, page 639 – page 663, 2005

[3] Hierarchical Data Format (HDF) http ://www.hdfgroup.org

[4] 3D graphics toolkit http ://www.openscenegraph.org/projects/osg

[5] CrossPlatform GUI Library http ://www.wxwidgets.org/