Laurent HORY
Adrien KRÄHENBÜHL

Licence Mathématiques
Informatique
Université Henry Poincaré Nancy
II

Projet Thématique Individuel

MoProViewer

Logiciel de visualisation moléculaire

Année 2008/2009

Sommaire

Introduction

Le projet qui nous a été proposé porte sur logiciel MoProViewer. C'est un logiciel réalisé et utilisé par le Laboratoire de Cristallographie et Modélisation des Matériaux Minéraux et Biologiques (LCM3B) pour analyser la densité électronique des molécules cristallisées. Il répond à des besoins de visualisation moléculaire spécifiques à la cristallographie. Le logiciel ayant déjà une base fonctionnelle, notre rôle était d'ajouter de nouvelles fonctionnalités ainsi que d'effectuer certaines corrections de bugs. Cependant nos objectifs ont rapidement changés et nous avons finalement repensé et implémenté une nouvelle architecture. Nous verrons donc pourquoi l'architecture existante n'était pas satisfaisante puis nous aborderons les démarches suivies afin d'ajouter de nouvelles fonctionnalités et de corriger les problèmes existants.

I. Un programme plein d'ambition

MoPro est ce que l'on peut appelé un programme "modulaire". Il regroupe différents programmes se complétant et quasi indépendants :

- ImportToMoPro, qui permet de convertir différents types de fichiers au format MoPro
- VMoPro qui à partir de fichier MoPro permet de visualiser les molécules et leurs propriétés en deux dimensions
- Flat-Solvent qui génère les facteurs de l'enveloppe moléculaire MoProGUI, l'interface entre les différents sous-programmes

L'une des originalités de ce logiciel est de combiner plusieurs langages : Pascal, Java, C++...

a) Il faut remplacer Jmol

MoPro utilise également un autre sous-programme, JMol, dans sa version greffon. C'est un logiciel OpenSource écrit en Java et disposant de nombreuses fonctionnalités. Il permet de visualiser des molécules et leurs caractéristiques en trois dimensions. Cependant le LCM3B n'était pas entièrement satisfait par ce programme, du fait de sa lourdeur, de l'impossibilité de lire des fichiers MoPro (format créé par le LCM3B) et de l'absence de certaines fonctions spécifiques à la cristallographie.

C'est pourquoi le projet MoProViewer a été lancé, afin de remplacer JMol au sein de MoPro et d'étendre les possibilités de la visualisation 3D. Les avantages qu'apportent un développement en interne sont nombreux : maîtrise du code, ajouts simplifiés de fonctionnalités, meilleure intégration à MoPro...

b) Principes de fonctionnement

MoProViewer a pour but principal de permettre la visualisation de molécules en trois dimensions. Les données nécessaires à cette visualisation sont stockées dans des fichiers sous différents formats :

- les fichiers de la Protein Data Bank (PDB) : La PDB est une banque mondiale de données sur les molécules, consultable par tout le monde sur internet (http://www.pdb.org). Tout les biologistes peuvent proposer des molécules, qui seront consultables une fois validées.
- les fichiers MoPro : Ce format à été mis au point par le LCM3B pour son logiciel MoPro. Il décrit des molécules avec notamment des informations

concernant les coordonnées, la charge et les liaisons entre atomes.

C'est à partir de ces deux formats de fichiers que MoProViewer doit être capable en premier lieu d'afficher les molécules, et ensuite d'afficher et de calculer diverses propriétés, comme nous le verrons par la suite. Il y a donc deux aspect du programmes à rendre cohérents :

- La représentation des molécules au sein du programme, qui doit permettre un accès facile et rapide au données out en minimisant l'occupation mémoire.
- La représentation graphique des molécules, devant être la plus interactive possible, tout en conservant un minimum des habitudes que peut avoir l'utilisateur sur les programmes phares de visualisation moléculaire comme par exemple RasMol, PyMOL.

Un autre point essentiel de MoProViewer est son mode de fonctionnement, qui doit justement être double. En effet il doit pouvoir d'exécuter de deux façons différentes : en temps que programme indépendant (standalone) et en tant que greffon (plugin). Tout au long de la programmation du logiciel, il faudra donc garder à l'esprit que celui-ci doit pouvoir s'exécuter dans ces deux modes. Il a donc été choisi de faire le développement en standalone en anticipant la mise en place du mode greffon. On peut noter ici que la partie greffon à été laissée de côté durant le stage, pour se concentrer sur le coeur du programme.

c) Combiner Qt et Opengl

Concernant la programmation proprement dite, notre référent au laboratoire, Benoit GUILLOT, avait déjà codé une bonne partie du logiciel. Celui-ci est basé sur le C++ et utilise les bibliothèques Qt pour les différents composants graphiques. M GUILLOT a choisit d'utiliser une autre bibliothèque, basée sur Qt, pour l'affichage des molécules : QGLViewer. Cette bibliothèque est une interface entre OpenGL et Qt et permet une utilisation plus intuitive d'OpenGL au sein de Qt.

La partie délicate de la programmation consiste donc à proposer une représentation objet à l'aide de la librairie Qt des molécules qui permette un affichage OpenGL simple et fluide des molécules. Nous nous sommes penchés longuement sur cette question, afin de recoder proprement la base de MoProViewer. Notre objectif était un chargement plus rapide des molécules et un accès facilité aux nombreuses données depuis l'affichage des molécules.

II.Un programme à optimiser

Lors de la présentation que nous a fait M GUILLOT, nous avons été complètement emballés par ce programme. Il était en pleine construction mais permettait déjà de visualiser les molécules et de nombreuses fonctionnalités étaient implantées. C'est en découvrant le code que nous avons pris conscience des problèmes. En effet, venant de débuter dans la programmation, M GUILLOT à développé ce programme en même temps que ses connaissances, ce qui à engendré de grandes disparités dans sa conception.

a) Le squelette à consolider

Concernant la représentation des molécules, des acides aminés et des atomes, elle était quelque peu confuse : on retrouvait par exemple des fonctions concernant les atomes dans la classe représentant les molécules. Certains attributs étaient redondants, comme le nom des atomes, et quasiment aucun n'était protégé. Ils étaient accessibles (et accédés) directement depuis les autres classes sans passer par les fonctions "get" et "set" habituellement utilisé en programmation objet.

b) Opengl à accélérer

En ce qui concerne OpenGL, M. GUILLOT rencontrait quelques problèmes au moment où il nous présentait son logiciel. Ainsi lors de l'affichage en "Ball and Sticks", on pouvait constater certaines lenteurs lors de la manipulation de molécules volumineuses, alors même que la scène ne semblait pas contenir de nombreux détails. Et de fait, nous nous sommes rendus compte à la lecture du code que certains calculs étaient effectués plusieurs fois, ce qui alourdissait les traitements et ralentissait l'affichage. Juste avant que nous ne nous attaquions au code, M. GUILLOT a commencé à optimiser l'affichage en utilisant la technique des "Display Lists", ce qui a rendu l'affichage plus fluide une fois la molécule chargée.

c) Des options à ajouter

A l'origine nous ne devions pas retoucher ce qui avait été fait, mais nous pencher sur des options à ajouter. Nous avions entre autres une option concernant la gestion de la lumière, un menu permettant de modifier de la couleurs des atomes depuis un tableau de Mendeleïev ou encore un menu permettant de régler les préférences d'affichages (voir annexe n°1). M GUILLOT nous a également demandé d'ajouter une vue de la molécule sous forme d'arbre, dans un panneau venant se greffer à l'interface existante. Cet arbre, que nous avons finalement réussi à mettre au point, permet de sélectionner un atome ou un acide aminé complet en cliquant simplement sur son nom.

III.Repenser MoProViewer

a) Une nouvelle organisation des atomes

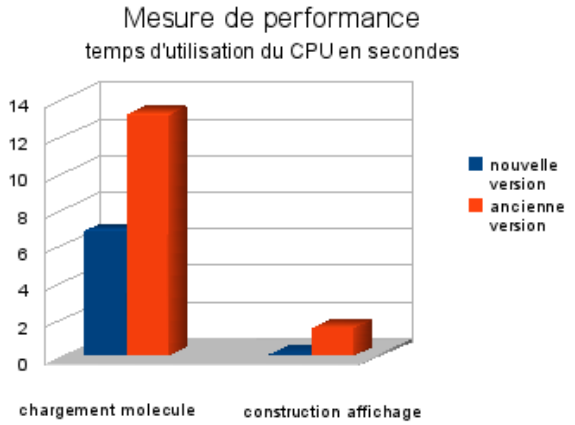
La représentation accordée à la molécule et aux atomes est vitale car toute la suite du programme repose sur elle. Un élément a cependant été décisif et nous a montré l'importance de cette architecture : ce sont les acides aminés. Effectivement dans la représentation qui était utilisée jusque là, il était difficile de définir à quel acide aminé appartenait un atome, et cela posait de grande difficulté pour la mise en place de certaines fonctionnalités. Afin d'éviter de retomber dans des pièges de conception, nous avons étudié les besoins de futurs utilisateurs. Nous avons dû faire un compromis entre l'optimisation possible et le temps qui nous était accordé pour construire un nouveau squelette.

La nouvelle architecture du squelette de MoProViewer est composée de trois classes : Atom, AminoAcid et Molécule. A l'image des poupée russes, la classe AminoAcid contient une liste d'atomes, tout comme la classe Molecule contient une liste d'acides aminés. La classe AminoAcid fait donc son apparition, et ceci pour faciliter la sélection sur laquelle nous reviendrons plus en détails.

Un des aspect important du squelette sur lequel nous avons réfléchis est la meilleure façon d'organiser les atomes pour représenter les liaisons inter-atomiques. Dans la version que nous a transmis Benoit, chaque atome possédait la liste des atomes auxquels il était lié, mais deux problèmes se posaient :

- Un atomes n'était pas accessible facilement, sa recherche nécessitant un parcours de tous les atomes.
- La création des liaisons ne prenait pas en compte le nombre limité de liaisons possibles pour un atome, et chaque liaison était calculée deux fois.

Hasard du calendrier, c'est a ce stade de notre réflexion que nous avons aborder durant le cours d'AGAP (Arbres Graphes Algorithmique et Programmation) de Mme Noëlle CARBONELL les matrices et listes d'adjacence. Etant donné qu'un atome ne fait pas plus de 7 liaisons et qu'il peut y avoir plus de 8000 atomes, c'est la liste d'adjacence qui a été retenue afin d'éviter une occupation inutile de la mémoire. De plus, l'algorithme de calcul des liaisons, et donc de construction de la liste d'adjacence, à donc été revu et optimisé : on accède maintenant directement à l'atome grâce à son numéro, et le calcul d'une liaison est pris en compte pour les deux atomes concernés. Le temps de chargement d'une molécule à ainsi été améliorer de plus de 50%.



Afin de connaître le gain réel de performances de la nouvelle architecture, nous avons utilisé un algorithme (annexe n°2) mesurant le nombre de tics effectués par MoProViewer lors de l'exécution d'une fonction ou d'un partie de code, ainsi que le nombre de tics que le processeur exécute en une seconde. Avec ces données nous en avons déduit le temps en seconde où le CPU était sollicité. Les mesures ont portées sur les deux parties les plus critiques du logiciel :

- le chargement d'une molécule qui comprend la lecture du fichier source et la création de la molécule

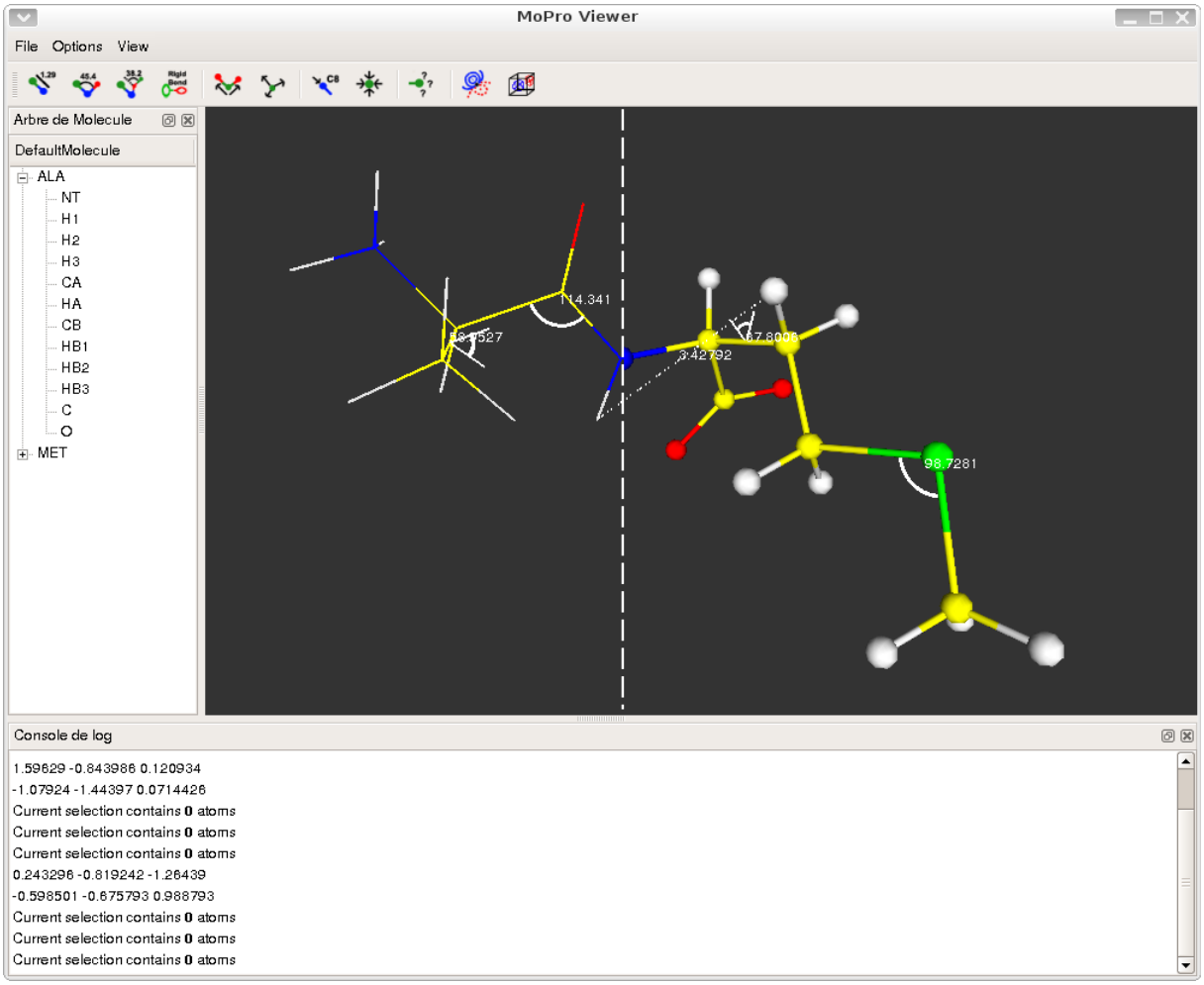
- L'affichage qui comprend la création du modèle en trois dimension à l'aide des fonction OpenGL.

b)Deux techniques Opengl pour deux modes d'affichage

Le programme doit utiliser deux modes d'affichage : un mode "Points and Lines" (à gauche des tirets blanc sur l'image), utilisé pour les travaux réalisé sur les molécules, et un mode "Ball and Sticks" (à droite des tirets blanc) pour le rendu final.

Pour le mode "Points and Lines" nous avions commencé par opter pour des "Vertex Array" optimisant ainsi l'affichage de ce mode. Mais la sélection n'était pas possible avec cette technologie. Nous avons alors remplacé ceux-ci par des "Vertex Buffer Object" qui améliorent le rendu par rapport aux "Vertex Array" et nous permettent d'implanter plus facilement la sélection. Quand au mode "Ball and Sticks", M GUILLOT nous a parlé des "Display Lists", permettant un meilleure rendu que les "glVertex". Nous avons décidé de mettre en pratique cette technique et de regrouper les différents éléments de base de l'affichage des molécules dans des "Display List" distinctes. Nous en avons donc créé deux : une pour l'ensemble des sphères et l'autre pour l'ensemble des liaisons.

Pour ces deux modes, nous avons deux structures communes nécessaires aux "Display Lists" et aux "VBOs", ce sont le tableau des coordonnées des atomes et le tableau des couleurs des atomes. Ces tableaux sont initialisés lors du chargement d'une molécule. Suivant le mode d'affichage, ces tableaux sont utilisés en l'état ou via un autre tableau indexant ces 2 tableaux.



c) La sélection au coeur de MoProViewer

La sélection est un outils incontournable de MoProViewer. Il existe 3 modes de sélections.

- La sélection depuis l'arbre : Elle permet de sélectionner les atomes un par un ou un acide aminé complet. Il suffit pour cela de cliquer sur l'identifiant (le nom de l'atome ou de l'acide aminé) désiré pour que celui ci soit sélectionné. Cette sélection est de loin la plus simple, mais un problème majeur nous bloquais dans ce mode. Nous avions utilisé un objet nommé "QTreeView" mais nous ne pouvions récupérer que le nom de l'atome sélectionné. Cependant notre consultation d'un atome se fais par l'intermédiaire de son numéro. Deux solutions s'offrait alors à nous. La première consistait à utiliser un dictionnaire afin d'associer le numéro au nom de l'atome, mais cela ralentissait la construction de la molécule et augmentait inutilement l'occupation mémoire. Nous avons opté pour une autre solution : changer d'objet graphique et choisir un QTreeWidget et créer un autre objet, héritant de QTreeWidgetItem, auquel on a ajouter un attribut contenant le numéro de l'atome qu'il représente.

- La sélection standard (voir annexe n°3) : C'est la sélection directe depuis l'affichage 3d. Il suffit de cliquer sur l'atome désiré et il est immédiatement sélectionné. Pour ce mode nous avions tout d'abord choisi la sélection lors d'un clic avec le bouton gauche de la souris. Un clic du bouton gauche de la souris accompagné de la touche "SHIFT" permet d'ajouter à la sélection et un clic du bouton gauche de la souris accompagné de la touche "CTRL" permet d'enlever l'atome de la sélection courante. Notre expérience d'utilisateur d'outils informatique nous avait fait pencher pour le choix d'une désélection globale lors d'un clic gauche en dehors de la molécule. M GUILLOT nous a expliqué que l'usage sur les logiciel de cristallographie était diffèrent et que le travail sur des modèles 3d étant fastidieux, un clic sur une zone vide est courant. C'est pourquoi il est préférable d'effectuer la désélection uniquement sur demande explicite de l'utilisateur.

- La sélection spécifique : C'est un mode de sélection réservé a certains outils. Nous commençons par choisir l'outil adapté, comme la mesure de distance, d'angle ou autres, puis nous sélectionnons le nombre d'atomes demandé et lorsque ce nombre est atteint, l'action désirée est effectuée sur la sélection correspondante sans prendre en compte le mode de sélection standard ni le modifier. Ainsi après avoir utilisé un outils nous pouvons reprendre nos travaux en cours sur la sélection courante sans que celle-ci n'ai subis de modifications.

Conclusion

MoProViewer a donc été repenser. Nous avons maintenant une base solide et évolutive. Le modèle de représentation des molécules choisit (annexe n°4) permet un chargement rapide et un affichage fluide des molécules. En effet nous avons pu implémenter un nouvel algorithme de construction des liaisons pour accélérer le chargement. Quand à la fluidité gagné en OpenGL, elle provient de la mise en place de deux nouvelles techniques d'affichages : les "Vertex Buffer Objects" et les "Display Lists". Ce sont donc autant d'avantages pour l'utilisateur que pour la suite du développement du projet.

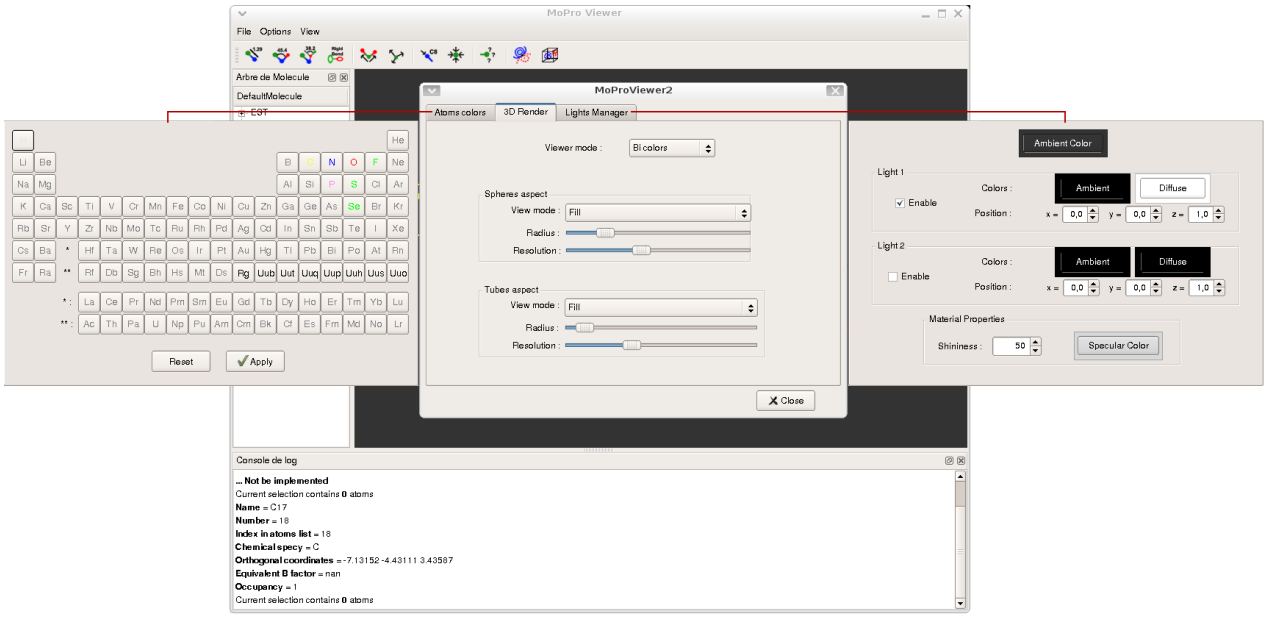
On retiendra également de ce projet qu'il nous aura beaucoup apporté personnellement. Nous avons ainsi découvert et utiliser le langage C++ et les librairies Qt et OpenGL. Ce sont de nouvelles compétences non négligeables qui étendent nos connaissances. Ce sont également des notions de physique et de cristallographie que nous avons (re)découvertes avec plaisir. L'application de l'informatique dans ce domaine est plus qu'une nécessité pour approfondir et développer la recherche.

En conclusion, ce projet nous a été bénéfique sur de nombreux plans, en plus d'être enrichissant. Nous avons pu mettre en œuvre des démarche de conception ce qui nous a fait prendre conscience de l'importance d'une conception robuste. De même pour les algorithmes, leurs choix s'est avéré prépondérant pour les performances de MoProViewer.

Ce projet nous a donné envie de poursuivre le développement de MoProViewer avec M. GUILLOT, en dehors du cadre des PTI. Il reste en effet de nombreuses fonctionnalités à implanter et nous sommes impatient de relever ces nouveaux chalenges.

Annexe n°1

Les trois onglets du menu des préférences :

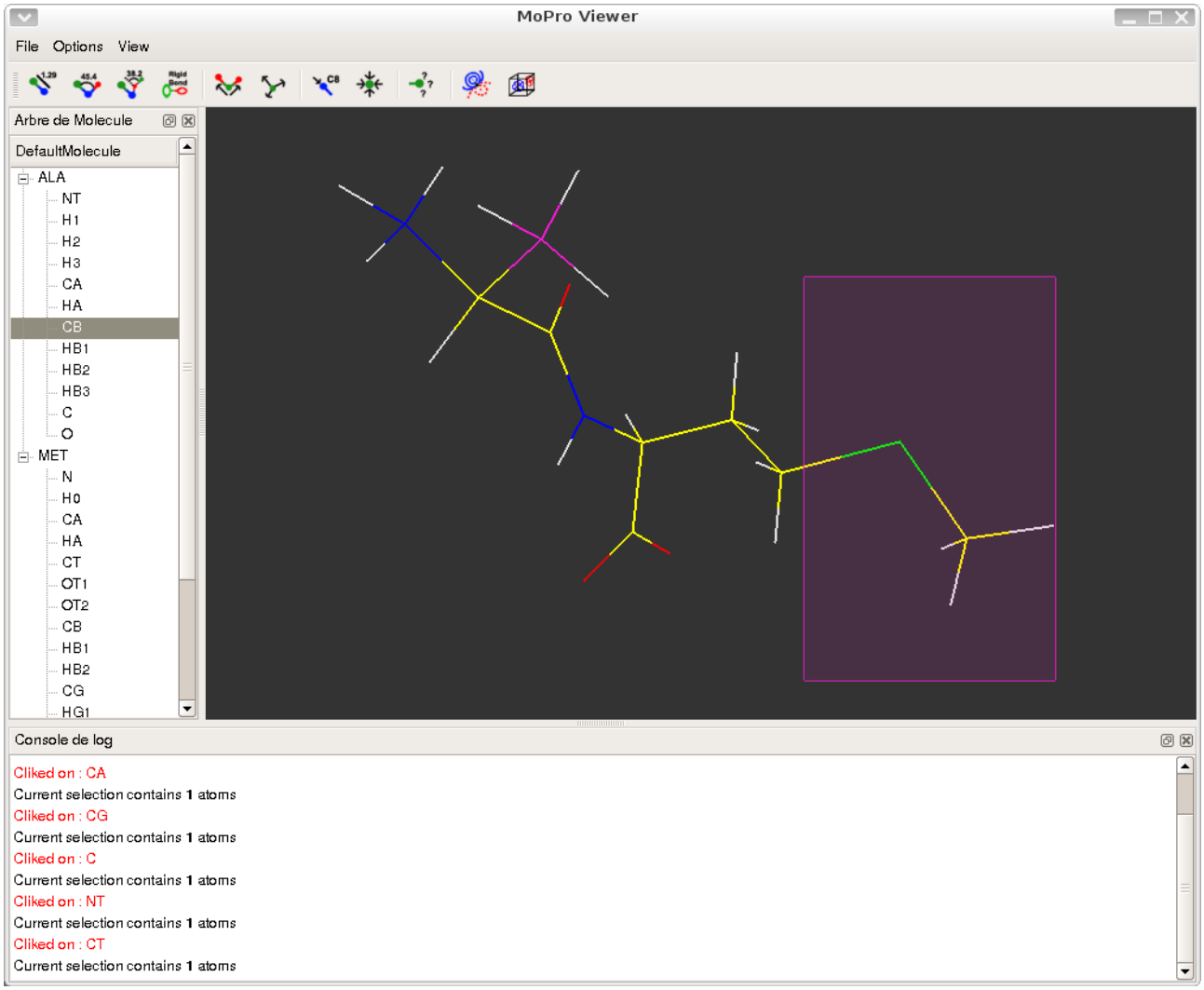


Annexe n°2

Algorithme pour le calcul du gain des performances :

```
double clockticks, cticks;
clock_t tcstart, tcend;
struct tms tmstart;
struct tms tmend;

clockticks = (double) sysconf(_SC_CLK_TCK);

cout << "Nombre de tics d'horloge par seconde:"
      << clockticks
      << endl;

tcstart = times(&tmstart);

      ma_fonction_a_mesurer();

tcend = times(&tmend);

cticks =    tmend.tms_utime
            + tmend.tms_stime
            - tmstart.tms_utime
            - tmstart.tms_stime;

cout << "Temps CPU utile "
      << cticks/clockticks
      << " secondes\n"
      << endl;
```

Annexe n°3


MoPro Viewer
File Options View
Arbre de Molecule
DefaultMolecule
ALA
NT
H1
H2
H3
CA
HA
CB
HB1
HB2
HB3
C
O
MET
N
H0
CA
HA
CT
OT1
OT2
CB
HB1
HB2
CG
HG1
Console de log
Cliked on : CA
Current selection contains 1 atoms
Cliked on : CG
Current selection contains 1 atoms
Cliked on : C
Current selection contains 1 atoms
Cliked on : NT
Current selection contains 1 atoms
Cliked on : CT
Current selection contains 1 atoms

Annexe n°4

Diagramme de classe de la nouvelle version de MoProViewer :