Bucquet Anthony
Grippari Guillaume
Hamon Gwénaëlle
Lataud Quentin
Turin Arnaud

22 Mai 2011

Rapport de Projet de Groupe

Animation 3D en cristallographie

Janvier – Mai 2011

Encadrant : Thierry Duffar

Sommaire

1. Introduction :

Dans le cadre de la première année d'école d'ingénieur à Phelma, il nous est demandé d'effectuer sur une période d'environ quatre mois un projet en groupe. Son but est de nous initier à la gestion de projet, de nous familiariser à l'organisation en groupe, tout en appliquant les connaissances acquises au cours de notre formation scientifique.
Ainsi, nous avons dû constituer un groupe d'étudiants qui avaient les mêmes goûts et qui pourraient donc s'entendre pour le choix d'un sujet de projet. Ainsi, nous nous sommes vite mis d'accord sur un sujet proposé par Thierry Duffar : *Animation 3D en cristallographie*.

Le but de ce projet était alors de pouvoir aider à apporter un meilleur enseignement en cours de cristallographie à Phelma, voire ailleurs. En effet, il est souvent dur pour les élèves de bien comprendre les subtilités de ce cours qui porte sur des représentations 2D d'un objet qu'on étudie dans l'espace, donc en 3D. Notre but était alors d'afficher des mailles cristallographiques en 3D, en utilisant une des technologies existantes, tout en respectant certaines contraintes qui seront expliquées dans le cahier des charges. Il a fallu pour cela effectuer des études préliminaires de ce qui existait déjà, et des différentes possibilités qui s'offraient à nous, avant de s'attaquer à la réalisation. Nous avons bien évidemment rencontré des difficultés qui nous ont poussés à changer d'avis sur certains points que nous détaillerons plus tard.

Afin de comprendre la démarche que nous avons utilisée pour mener ce projet à son terme, notre rapport se structure de la façon suivante :
Tout d'abord, nous présentons la problématique ainsi que le cahier des charges en présentant nos études préliminaires. Ensuite, nous expliquons comment nous avons développé notre programme, incluant les problèmes rencontrés, avant de décrire les résultats obtenus et les améliorations possibles. Enfin, nous conclurons sur un retour d'expérience sur la gestion de projet de groupe.

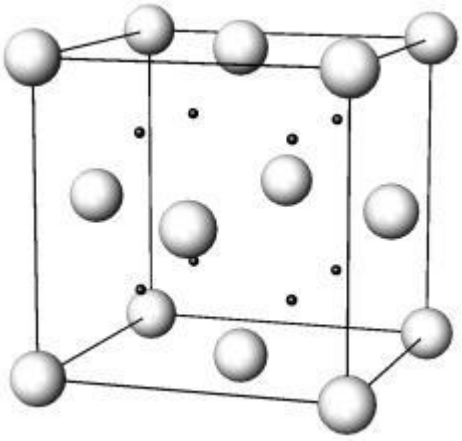
2. Etudes préliminaires :

Avant de nous lancer dans la réalisation du projet, il a fallu effectuer des recherches pour savoir quel genre de logiciel existait déjà pour afficher des mailles, et quels étaient les différents moyens dont on disposait pour afficher du 3D. Il fallait pour cela définir clairement le but du projet et son cahier des charges, afin de choisir au mieux la technique à utiliser pour mener à bien ce projet.

2.1. Cahier des charges :

2.1.1 Objectifs :

L'objectif principal de notre projet est d'afficher, à l'aide d'un rétroprojecteur, des cristaux en 3D, pour les cours de cristallographie de Phelma. L'idéal serait de pouvoir afficher le plus possible de mailles différentes, si possible un nombre illimité, choisies par l'enseignant. L'idée est de créer un programme ou une vidéo, que l'enseignant pourrait diffuser, montrant le cristal qu'il souhaite. Les cristaux devront pouvoir être tournés pour voir selon différents angles de façon fluide.



4

Pas toujours facile de visualiser les mailles, même basiques !

Pour une meilleure visibilité, il serait préférable de pouvoir modifier certaines options, comme la taille des particules, ou leur couleur. Il devra bien entendu être simple d'utilisation.

2.1.2 Contraintes :

Le projet s'adresse à un public bien précis : il s'adresse principalement à des enseignants en cristallographie ainsi qu'à des étudiants. L'affichage doit donc pouvoir être réalisé en amphi, avec l'aide d'un rétroprojecteur simple, ou sur un simple ordinateur. Il ne s'agit pas de rééquiper les bâtiments de Phelma avec des technologies chères, nécessitant deux projecteurs polarisés par exemple. De plus, il doit pouvoir être vu par tous les étudiants d'un amphi, c'est-à-dire jusqu'à 90 personnes. Ainsi, étant donné que le budget doit être le plus restreint possible, si nous devons fournir des lunettes à tous ces étudiants, elles doivent être les moins chères possibles.

2.2 Etat de l'art :

À l'heure actuelle, il existe de nombreux logiciels permettant d'afficher des cristaux en perspective, de les retourner pour les observer sous plusieurs angles, comportant de nombreuses options semblables à celles que nous comptions intégrer à notre travail. Mais il ne semble pas qu'il y ait de logiciels permettant de les afficher en 3 dimensions, bien que la 3D soit en plein essor, et soit utilisée de plus en plus dans de nombreux domaines notamment dans le cinéma.
De plus, nous avons pris connaissance de l'existence de fichiers contenant des données cristallographiques, des fichiers .cif. Ces fichiers sont en accès libre sur internet et contiennent toutes les données nécessaires à la construction d'une maille (place des atomes, symétries etc...).

2.3. Outils :

2.3.1. Méthodes d'affichage 3D :

Nos premières recherches se sont tournées vers les différentes techniques d'affichage en trois dimensions, afin de choisir la plus adaptée à notre cahier des charges et à nos compétences.
Nous avons rapidement considéré la possibilité de filmer nous même avec plusieurs caméras décalées des maquettes de mailles dont Phelma dispose dans la salle de cristallographie. Nous avons vite abandonné cette possibilité, car elle demandait trop de contrainte (trouver le matériel audiovisuel adéquate, obtenir des dispositifs pour faire tourner les maquettes de façon fluide et uniforme), et ne pouvait pas répondre aux demandes : grande variété de mailles affichables, taille et couleur des atomes modifiable).

Il restait alors plusieurs méthodes d'affichage en trois dimensions :

La séparation chromatique :

Elle cherche à recréer le même principe que la vision 3D qu'a l'homme : si l'homme voit en 3D, c'est parce que ses deux yeux sont légèrement décalés, et qu'il voit ainsi deux images différentes qui lui permettent d'apprécier les distances, et ainsi de créer un environnement en trois dimensions.
La séparation chromatique consiste à décaler légèrement une image par rapport à elle-même, en créant un anaglyphe: une des images sera à dominante rouge, et l'autre à dominante cyan. L'observateur, lui, met des lunettes dont un verre est rouge, l'autre cyan, ce qui lui permet d'observer deux images différentes avec les deux yeux, et ainsi de recréer l'illusion de profondeur. Cet affichage est faisable en programmation grâce à une bibliothèque 3D, nommée OpenGL.

Le papier holographique :

Le principe est de réussir à imprimer un papier capable de générer des interférences, de sorte que l'image observée varie selon l'angle d'observation. Cette méthode est notamment utilisée sur les cartes bleues. Voir *schéma 1* ci-contre.
Sa réalisation est cependant très difficile et nécessite des outils et des matériaux spécifiques dont Phelma ne disposait pas. De plus, cette méthode ne permettait pas de pouvoir tourner les images à notre guise, ou de modifier des paramètres.

La polarisation de la lumière :

Cette technique est celle qui est de plus en plus utilisée pour les projections cinématographiques. Elle est basée sur le principe de la polarisation : elle nécessite deux caméras qui polarisent la lumière émise selon deux directions orthogonale. Le spectateur, lui, possède aussi des lunettes polarisantes dont les deux verres sont polarisés différemment. Ainsi, celui-ci peut dissocier ses deux yeux, et voir avec l'œil droit l'image qui correspond à l'œil droit, et de même avec le gauche. Cela recrée l'effet de perspective, tout comme la technique de séparation chromatique, mais conserve une bonne qualité d'image et surtout de couleurs.

Ainsi, après avoir listé les principales possibilités qui s'offraient à nous, nous avons décidé d'adopter la méthode la plus économe, et certainement la plus simple. Nous

avons donc choisi de nous lancer dans la programmation d'un logiciel qui serait capable d'afficher des anaglyphes, en utilisant le langage C et une bibliothèque : OpenGL.

2.3.2. Langage de programmation :

Langage C et Code::Blocks:

La réalisation de ce projet nécessitait tout d'abord de choisir un langage de programmation adapté, mais surtout que nous étions capables de manipuler. N'ayant étudié que le langage C, nous avons vite décidé d'utiliser le langage C. Etant donné que nos PC étaient tous sous Windows, nous avons utilisé le logiciel Code::Blocks, qui fournit une interface permettant de programmer, de compiler et de débuguer sous Windows. Mais la librairie Windows n'accepte pas le C, nous avons dû utiliser le compilateur C++.

Bibliothèque OpenGL :

Afin de générer des mailles en 2D et en 3D, nous avons dû utiliser une librairie spéciale qui permet de concevoir des applications générant des images : OpenGL (**Open G**raphics **L**ibrary). Elle utilise les représentations de la géométrie projective, et regroupe environ 250 fonctions différentes qui peuvent être utilisées pour afficher des images tridimensionnelles complexes à partir d'images 2D. Elle est utilisée pour la majorité des applications scientifiques, industrielles ou artistiques, ainsi que dans l'industrie du jeu vidéo.

2.3.3. Budget:

Afin de mener à bien notre projet, il nous a fallu établir un budget prévisionnel. Après avoir effectué nos choix sur la méthode à adopter, nous avons conclu que notre budget devait servir à acheter des lunettes 3D (rouge et cyan), en quantité suffisante pour qu'un amphi puisse assister à un cours de cristallographie. Nous avions donc prévu 60€ d'achat pour une centaine de lunettes. La prévision du budget n'était pas très difficile, car notre projet ne risquait pas d'avoir besoin de matériel supplémentaire imprévu. Ainsi, lors de l'achat des lunettes, nous avons dépensé 56€50, ce qui ne dépassait pas le budget prévisionnel.

Une fois toutes ces études préliminaires effectuées, nous étions prêts à démarrer notre projet, donc à passer à sa réalisation.

3. Réalisation :

À l'issue des études préliminaires, nous avons pu alors définir la structure finale du programme, lister les tâches à effectuer, faire un organigramme des tâches (*annexe 1*) ainsi qu'un diagramme de Gantt (*annexe 2*) afin de pouvoir terminer à temps le projet.

Le programme devait avoir la structure suivante :

- Initialisation : Contient toutes les fonctions à exécuter avant l'ouverture de la boucle. ex : ouverture de la fenêtre, allocation de mémoire

- Boucle principale : Cette boucle sera exécutée jusqu'à ce qu'on appuie sur la croix ou qu'on appuie sur echap ou qu'une erreur soit présente dans la boucle.

- Options : Fait la réception de l'interface utilisateur/programme. Récupère les options par des boutons dans la fenêtre par un système de fonction callback.

- Affichage des points, de lignes etc. : Affiche les atomes en fonction des options données précédemment.

- Caméra 2 vues : Réalise la projection et l'affichage du bleu et du rouge dans la fenêtre. - Rafraichir l'image

- Fin de la boucle Conclusion : Fermeture la fenêtre, désallocations.

Cela nécéssitait de bien définir les tâches à effectuer, et de les lister.
Voici les principales tâches que nous avons réalisées tout au long de ce projet :

Géométrie de base : Créer les types pour la géométrie : Vecteur (= Point) , Matrice (4 x 4, lisez le tuto OpenGL pour comprendre pourquoi), Ligne (2 Points)
Faire les fonctions qui vont avec : somme, différence, multiplication par un

scalaire, produit scalaire, produit vectoriel, produit, norme, normer...

Structure de maille et lecture de fichier : Mettre dans une structure Maille tous les paramètres nécessaires à son affichage. Faire également une fonction de lecture dans le fichier généré pour remplir cette structure.

Fonctions basiques d'OpenGL : Faire les fonctions de base d'OpenGL qui serviront pour la suite. Entre autres : afficher boule (avec les paramètres taille, couleur, position), afficher une ligne, afficher un plan, mettre la couleur...

Boutons de la fenêtre : Ajout et disposition des boutons dans la fenêtre.

Affichage d'une maille : Faire la fonction qui prend en paramètre une structure maille et qui dessine cette maille. Elle pourra prendre en compte des paramètres passés en option.

Évènements : Réglage de la fonction CallBack des évènements.

Gestion de la caméra : Faire le réglage de la caméra (Freefly et Trackball). Créer la structure qui contiendra toutes les infos pour la caméra. Faire les fonctions de déplacement de la caméra (qui correspondront à des évènements clavier|souris).

Gestion des erreurs : Faire les fonctions d'affichage des messages d'erreur.

Affichage des anaglyphes : On règle l'affichage cyan/rouge avec les codes trouvés sur internet. On effectuera également des tests avec un rétroprojecteur.

Achat de lunettes : Passage de la commande des lunettes.

Communication : Réalisation de la vidéo de présentation du projet.

Finalisation du programme : On termine le projet, on effectue des derniers tests.

Rédaction du rapport final : Rédaction du rapport final.

Préparation de la soutenance.

3.1. Répartition des tâches :

Nous avons établi lors de la seconde séance une liste des éléments logiciels à gerer. Les trois principales parties étaient la création de la maille, l'affichage et la lecture des fichiers.

Chacune de ces parties nécessitaient plusieurs étapes de codage, ainsi le travail a pu être séparé entre les membres du groupe et être planifié sur la période du projet.

Anthony s'est alors occupé de répartir les tâches entre les 5 membres du groupe, en formant des petites équipes qui travaillent conjointement sur chaque partie importante du projet. Avec ce système, nous avions en permanence plusieurs personnes travaillant sur les mêmes éléments, ce qui a permis de palier aux difficultés rencontrées et donc d'avancer plus rapidement que si une personne devait gérer une partie à elle seule.

3.2. Description de la réalisation :

3.2.1. Choix :

Notre programme est séparé en 3 grandes parties : la lecture des fichiers .cif, la représentation des mailles décrites dans ces fichiers et l'interface du logiciel.

Lecture des fichiers .cif:

Après avoir longtemps cherché à réutiliser les programmes déjà existants qui pouvaient générer des mailles en 2D et proposait les options dont nous avions besoins, nous avons finalement conclu qu'aucun ne pouvait être compatible avec notre code : en effet, nous n'avons pas trouvé de code source de ces logiciels, et aucun d'entre eux ne généraient des fichiers que l'on pouvait exploiter. Ainsi, nous nous sommes finalement résolus à recommencer nous même tout le programme, à l'aide des fichiers .cif : chaque fichier correspond à une maille caractéristique (NaCl par exemple, ou la blende ZnS). Ceux-ci sont en libre accès sur internet et contiennent de nombreuses informations nécessaires à construire cette maille. Notre but était alors de lire ces informations et en sélectionnant uniquement celles qui nous intéressaient. Nous devions ensuite à partir des informations de symétrie lue, calculer la position de chaque atome et la mettre en mémoire dans un type maille, qui est ensuite utilisé pour afficher la maille. Pour la structure maille, nous avons choisi de ranger les atomes dans un octree. C'est un arbre à 8 branches, les

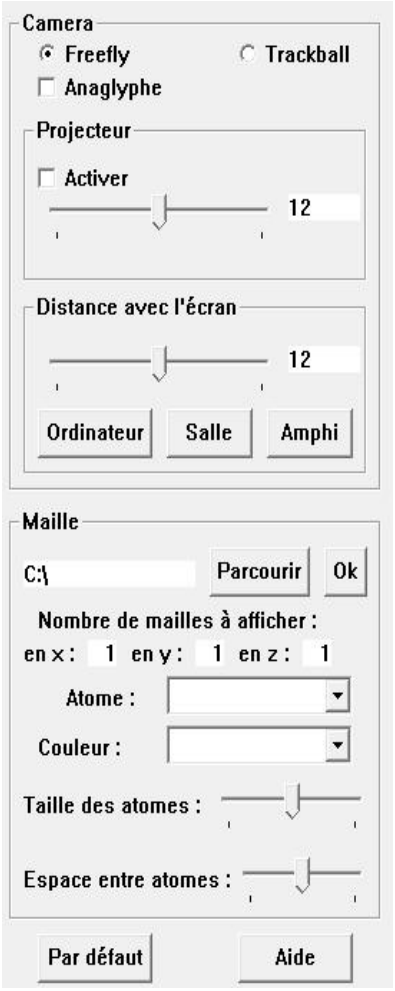
atomes y sont rangés en fonction de leur position dans l'espace. Nous avons décidé d'utiliser un automate pour lire le fichier. L'algorithme consiste à lire les caractères un par un et à repérer les chaines de caractères qui précèdent les informations utiles à la création de la maille. Lorsqu'une chaîne intéressante est lue, le programme va alors stocker la valeur numérique (dans le cas d'un angle ou d'une longueur) dans une structure maille préalablement créée, ou la ligne qui contient l'équation dans le cas des symétries. Ces chaînes sont ensuite lues une par une puis les informations sont rangées dans un arbre. Celui-ci est ensuite utilisé pour générer les atomes restant et pour supprimer les doublons (beaucoup de symétries créent des atomes en plusieurs fois, et il est nécessaire de n'afficher qu'une fois chaque atome pour la fluidité de l'affichage)

Représentation des mailles :

Elle s'est séparée en 2 sous parties : d'une part le dessin en 3D de la maille, d'autre part, la gestion d'une caméra mobile.
Pour ces 2 parties, nous avons choisi de créer d'abord un ensemble de fonctions de géométrie de base, avec tous les opérateurs vectoriels et matriciels en dimension 4 (la 4^{e} dimension sert à gérer les transformations affines, comme les translations). Cet ensemble de fonctions nous a servi à la fois pour les dessins et pour les mouvements de la caméra.
Pour la partie dessin, nous avons commencé par générer des formes de base (sphère, ligne, cylindre, …) en utilisant la bibliothèque OpenGL déjà existante. À l'aide de ces fonctions nous avons créé la fonction afficher_maille, qui, à partir de la structure maille, affiche tous les atomes présents, avec une couleur différente en fonction de leur type, qui peut être changée par l'utilisateur et un rayon que l'on peut faire varier pour chaque type d'atome. Nous avons enfin créé les anaglyphes à partir du dessin, en associant 2 caméras légèrement décalées et en associant le rouge ou le bleu à chaque image.
Pour la partie caméra, après avoir hésité entre les 2 types de mobilité les plus utilisées, à savoir Freefly et Trackball, nous avons décidé de faire les 2 et de laisser le choix à l'utilisateur d'utiliser l'une ou l'autre. Une caméra Freefly a un déplacement totalement libre dans l'espace, tandis qu'une caméra Trackball fixe toujours le même point et tourne autour. Il a donc fallu créer toutes les fonctions de déplacement et de pivotement qui ont ensuite été associées à des touches du clavier et à la souris. Le codage de ces fonctions a été largement facilité par la création au préalable des fonctions de manipulation des structures (notamment les fonctions basiques sur des vecteurs)

L'interface :

Figure 2 : Allure de la boîte de dialogue



Nous avons choisi d'avoir une fenêtre avec un menu à droite, toujours visible, qui contient tous les boutons du programme. Nous avons dû pour cela créer la fenêtre, créer les boutons et les associer à leur fonction réciproque. Nous avons évidemment utilisé des fonctions existantes mais qui n'était pas toujours facile à employer. Il a fallu enfin gérer les erreurs potentielles.

Toutes ces parties ont nécessité des tests pour vérifier que les fonctions créées fonctionnaient, il a fallu pour cela créer des fonctions de test que nous avons ensuite supprimées. Enfin, nous avons fait des tests finaux sur plusieurs fichiers .cif téléchargés.

3.2.2. Problèmes rencontrés

Difficultés de départ :

La première difficulté était de se mettre à coder, pour la plus part des membres du groupe le langage C était quelque chose de tout nouveau, il a donc fallu apprendre par nous-mêmes, sur des sites internet et entre nous. Nous avons donc mis du temps à démarrer et être efficace.

La deuxième difficulté était de coder à 5 sur un même programme. Nous avons réglé très vite le problème en utilisant un serveur commun (SVN) qui nous permettait de partager nos codes et de compléter le code des autres, même depuis chez nous, ce qui s'est avéré vraiment très pratique.

Difficultés dans le code :

Nous avons rencontré beaucoup de problèmes dans le code, nous allons ici énoncer les principaux.

- La lecture de fichier :

Les premiers furent rencontrés lors de l'écriture de l'automate de lecture du fichier. En effet les fichiers .cif ne sont pas soumis à des normes d'écriture précises. Ainsi les informations utiles étaient toujours précédées d'une chaîne de caractère, mais la difficulté résidait dans le fait que ces chaînes pouvaient être séparées par différents caractères (un saut de ligne, une tabulation, une virgule ou une chaîne d'espacement). L'automate plantait alors souvent au début car nous lui demandions de lire une valeur numérique qu'il ne parvenait pas à trouver. Ce problème a pu être résolu en créant plusieurs switch qui nous ont permis de créer un algorithme qui peut lire la grande majorité des fichiers .cif.

```
loop_
_symmetry_equiv_pos_as_xyz
x,y,z
z,-x,y
-y,z,-x
x,-y,z
-z,x,-y
y,-z,x
-x,y,-z
x,-z,-y
-z,y,x
y,-x,-z
-x,z,y
z,-y,-x
-y,x,z
```

Figure 3 : Exemple de syntaxe des symétries dans un fichier .cif

La partie symétrie a ensuite posé problème car nous devions avoir une approche différente, étant donné que les informations textuelles devaient être converties en données utilisables dans une équation.

- Les fonctions géométrie :

La difficulté en géométrie était d'effectuer des changements de repère - en translation, en rotation, en changeant les angles entre les vecteurs et en changeant la norme des vecteurs – en considérant tous les cas possibles. Nous avons utilisé les matrices de passage pour faire ces transformations. Ces changements de repère nous étaient utiles pour dessiner des cylindres (la fonction d'OpenGL ne les dessinait que suivant la verticale) et pour obtenir la base vectorielle de la maille.

La caméra s'est faite sans trop de difficultés grâce aux fonctions de manipulation des structures préalablement codées. Il a fallu néanmoins corriger deux bugs qui menaient le logiciel à planter lors du passage entre les caméras Freefly et Trackball.

La gestion de l'interface a posé, par contre, beaucoup de problème. D'abord dans la création de tous les boutons et ensuite dans leur association à leurs évènements respectifs. L'un des premiers problèmes a été de faire cohabiter l'API Windows et la bibliothèque OpenGL. Un deuxième problème est survenu quand nous avons voulu associer des évènements avec les boutons. Globalement, nous avons du supprimer le parentage des boutons qui était très utile lors du redimensionnement de la fenêtre et donc refaire la fonction de redimensionnement. Ensuite, il a fallu lier les sliders avec les boites d'édition. Une fois ces problèmes réglés, l'association a des actions a été faite assez facilement, en utilisant une variable globale qui englobe

tous les paramètres des boutons cochés ou décochés.

Difficultés d'organisation :

Lors du premier mois du projet, le rendement du groupe n'a pas été satisfaisant. Le cahier des charges nous a pris plus de temps que prévu et nous nous sommes vraiment lancé dans le projet deux semaines après le date de début de projet. Néanmoins nous avons pu rattraper notre retard pendant le mois d'avril dès que l'algorithme de lecture des fichiers a été finalisé (nous ne pouvions auparavant tester le logiciel qu'avec des mailles créées manuellement, ce qui restreignait grandement les possibilités de debugger et d'améliorer le programme). Notre retard s'est vu ainsi être rattrapé fin avril et le mois de mai nous a permis de le finaliser et de l'améliorer sans contrainte de temps.

4. Résultats :

4.1 Objectifs atteints

Maintenant que le projet est arrivé à terme, nous pouvons revenir sur le cahier des charges et comparer avec ce que nous avons réussi à produire.

Le grand objectif principal, à savoir la possibilité d'afficher des mailles en 3D, a été largement atteint. Nous offrons de nombreux paramètres d'affichage (couleur des atomes, taille des atomes, deux types de caméra, réglage de la distance 3D etc..) qui permettent un grand confort lors de l'utilisation du logiciel.

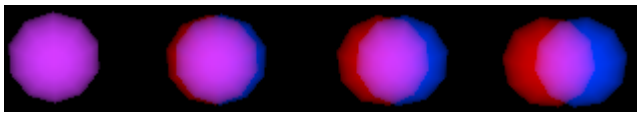
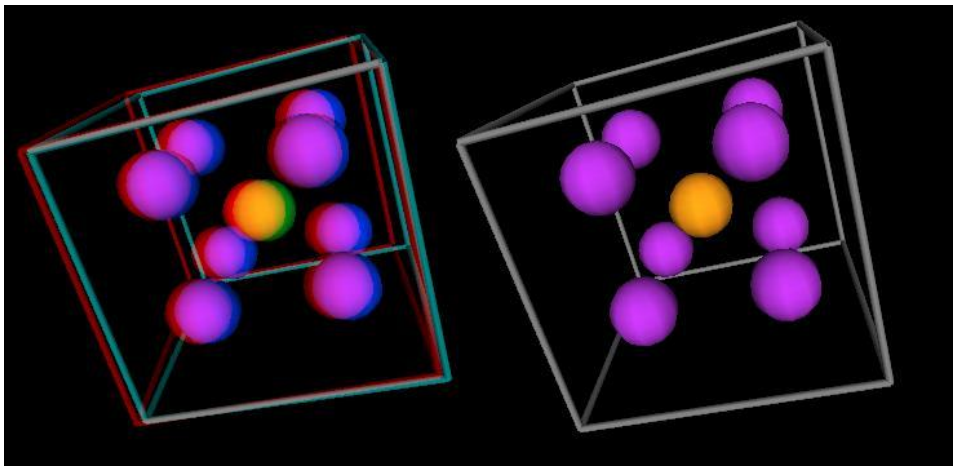


Figure 4: Le réglage de l'écartement virtuel des yeux permet une vision optimale quelque soit l'écartement de l'écran

Le rendu 3D est très convenable lorsque les paramètres sont bien réglés. Les paramètres définis par défaut doivent toujours être modifiés légèrement par l'utilisateur pour optimiser l'affichage, mais notre logiciel dispose d'une prise en main rapide et les réglages sont simples à effectuer même lors de la première utilisation.

Figure 5 : Exemple d'affichage 3D/2D fourni par le logiciel sur une maille NaCl cubique simple



Le choix de supporter les fichiers .cif nous permet de générer la grande majorité des structures disponibles dans la base de données du site crystallography.net, qui recense l'ensemble des fichiers des mailles étudiées en cours de cristallographie.
L'installation du logiciel se fait aussi très simplement et très intuitivement. Nous avons ainsi programmé un exécutable ouvrant un installeur qui prend en charge l'ensemble des fichiers nécessaire au fonctionnement du logiciel et il ne nécessite pas plus d'une minute pour pouvoir utiliser le logiciel.

4.2 Améliorations possibles :

Bien que nous soyons très satisfaits du programme final, il reste tout de même certaines choses qui pourraient être améliorées :

Tout d'abord, il existe un problème au niveau des fichiers .cif. Cela de dépend pas de notre programme, mais des données des fichiers .cif qui ne sont pas toujours cohérentes. En effet, certains d'entre eux affichent trop d'atomes, et ne représentent pas une maille primitive.

Il serait aussi intéressant de pouvoir tracer les arrêtes des mailles pour plus de visibilité, ainsi que les liaisons interatomiques. Cependant, les calculs pour pouvoir les tracer sont très difficiles à faire car ils demandent de nombreux autres paramètres (la taille des atomes réels notamment), et nécessitent de prendre en compte des interactions supplémentaires (Wand der Waals, liaisons H …) qui sont

très difficilement prévisibles si l'on ne fait pas du cas par cas.

Il serait aussi utile pour les utilisateurs de pouvoir afficher les sites interstitiels d'une maille pour pouvoir prévoir les alliages possibles. (Et si possible de calculer en plus leur taille maximale pour pouvoir prévoir la des alliages par insertion ou par substitution). Mais cela demande des calculs qui nous étaient impossibles de faire dans le temps imparti.
Nous avons également essayé de programmer une fonction de rotation automatique de la maille, mais cette fonction nécessitaient d'insérer un bouton dans la boîte de dialogue déjà assez chargé, un slider pour régler la vitesse de rotation (cette vitesse doit toujours être adaptée à la taille de la maille) et nous n'avons pas réussi à corriger un bug qui entrainait la paralysie de l'affichage et obligeait l'utilisateur à relancer le programme.

Enfin, le problème peut-être le plus gênant pour l'utilisation du logiciel est qu'il ne fonctionne qu'avec Windows®. En effet, l'API (Application Programming Interface) utilisée pour construire la fenêtre n'est compatible qu'avec Windows®. Il faudrait reprendre le programme presque au début (à l'affichage de la fenêtre) et programmer différemment pour le faire tourner sur d'autres systèmes d'exploitation. Cependant, notre tuteur à qui le projet était initialement destiné utilise Windows, et c'est aussi le système d'exploitation utilisé par la plupart des personnes susceptibles de l'utiliser.

Conclusion

Le projet une fois finalisé aura largement dépassé nos attentes lors de l'élaboration du projet. Nous avons pu ajouter de nouveaux objectifs en cours de développement du logiciel que nous sommes parvenus à atteindre qui ont largement amélioré le logiciel final. Ce projet aura ainsi mobilisé nos compétences du début jusqu'à la fin, nous a permis d'en acquérir de nouvelles et aura nécessité une bonne organisation tout au long de la période imposée. Le mois de mai aura apporté à l'ensemble du groupe une grande satisfaction devant les tests effectués sur écran et sur vidéoprojecteur. Nous espérons ainsi que ce programme pourra être utilisé lors de prochains cours de cristallographie, qu'ils soient dispensés en salle comme en amphithéâtre, et que les élèves, même extérieurs à Phelma, l'utiliseront pour faciliter la compréhension des cours. Nous remercions notre tuteur Thierry Duffar, qui nous a été de très bons conseils, et nous a suggéré des améliorations au cours de la réalisation du projet.

6.Gestion de projet :

6.1. Répartition des tâches :

Anthony Bucquet : Chef de groupe, supervise et corrige les codes. Lecture des fichiers .cif, gestion de l'interface.
Guillaume Grippari : Partie géométrie et OpenGL
Gwénaëlle Hamon : élaboration de la structure maille, achat du matériel (lunettes), communication et rédaction.
Arnaud Turin : Budget, partie géométrie et OpenGL.
Quentin Lataud : Gestion de la caméra, communication, rédaction.
Tout le groupe a participé à la finalisation du programme, aux derniers tests et à la recherche d'éventuels bugs.

6.2. Analyse du Gantt :

Lors de la réalisation du diagramme de Gantt (*Annexe 2 :* Diagramme de Gantt), nous avions séparé les tâches en fonction des membres qui devaient les effectuer. En pratique, nous avons peu respecté la répartition initiale, car nous n'avions pas assez bien évalué les compétences de chacun : ainsi, certaines tâches ont mis plus de temps que prévu, et d'autres moins. Il fallait alors entamer les tâches suivantes selon les disponibilités de chacun, ce qui a entraîné un remaniement continu de la répartition des tâches. Mais les tâches étaient souvent terminées à temps selon les prévisions du diagramme. Nous avons juste eu un retard au niveau de la lecture des fichiers .cif qui s'est avérée bien plus compliquée qu'il n'en avait l'air, et qui empêchait ensuite de tester les nouvelles fonctions. Cependant, une fois la lecture rendue possible, tout s'est accéléré, et nous avons vite rattrapé le retard, car par chance les fonctions non testées fonctionnaient – ou presque.

6.3. Retour d'expérience, analyse critique de la gestion :

Thème	**Succès**
Gestion du temps	Nous avions prévu assez large dans le diagramme de Gantt, afin de se permettre des retards liés à des imprévus.
Communication interne	Le forum, les rencontres hebdomadaires ainsi que le serveur commun (SVN) ont assuré une bonne communication.
Gestion des risques	Lors du choix d'utiliser des fichiers .cif, il y avait de fortes chances pour qu'on rencontre des problèmes pour les lire. Cela nous a ralenti, mais prendre ce risque a permis de pouvoir adapter notre logiciel au plus grand nombre de mailles possibles.
Planification	Bonne planification, malgré un retard dû au changement de méthode de chargement des mailles (utilisation des fichiers .cif).
Aspects techniques et scientifiques	Bonne recherche sur les méthodes d'affichage 3D (anaglyphes) Grande progression en programmation pour ceux qui avaient un niveau plus faible.

Thème	**Erreurs à éviter**	**Causes identifiées**
Gestion de l'équipe projet	Devoir terminer la tâche d'un autre membre du groupe car il n'arrive pas à la terminer seul.	Mauvaise répartition des tâches en fonction des compétences.
Gestion du projet	Ne pas s'y prendre au dernier moment pour rédiger les rapports.	Nous ne voulions pas rédiger avant d'avoir atteint les objectifs prévus.

6.4. Apport personnel du projet :

Anthony Bucquet: J'ai été assez impliqué et motivé par ce projet de groupe. En tant que chef du projet, j'ai beaucoup appris en ce qui est d'organiser et de gérer un projet, mais aussi en qui est de motiver les autres membres du groupe pour être le plus efficace possible. Au final, le retour m'a été positif, puisqu'on a fini le projet dans les temps. Au niveau scientifique, j'en ai appris davantage en programmation en C, notamment dans l'utilisation des API , des fichiers et de la 3D.

Gwénaëlle Hamon : Ce projet de groupe a été très bénéfique pour moi. En effet, il traite d'un domaine qui m'intéresse (la cristallographie), et la réussite du programme me tenait vraiment à cœur. Il a aussi été intéressant au niveau humain, et très formateur : chaque membre avait des connaissances utiles au groupe (cristallographie, informatique...), mais différentes, ce qui a permis une grande entraide, et un apport de connaissances constant entre tous les membres.

Guillaume Grippari: L'informatique étant un domaine qui m'intéresse, ce projet demandant des compétences en programmation poussées m'a beaucoup intéressé. J'ai aussi pu suivre l'évolution d'un projet, de la présentation de l'objectif à son accomplissement. J'ai ainsi réalisé l'importance de bien cerner le problème, de sélectionner les idées et d'établir un cahier des charges adéquat. Sans la lourde étape d'état de l'art, nous n'aurions peut-être pas pu terminer à temps.

Arnaud Turin : J'ai beaucoup aimé le fait de réaliser un produit dans sa globalité, avec une équipe et un temps assez importants pour nous permettre de faire un logiciel abouti. J'ai trouvé intéressant de créer quelque chose d'utile. Nous avions toujours en tête les désirs du «client» et nos choix se faisaient toujours pour répondre au mieux à ses attentes. Ce projet m'a aussi fait découvrir la création de logiciel, ce qui m'a beaucoup plu et m'a orienté pour mon choix de filière.

Quentin Lataud : Ce projet aura été très bénéfique pour moi. Il m'aura permis de me perfectionner en codage et surtout de m'apporter de nouvelles notions. J'ai aussi appris à travailler en mode projet, et le délai réduit m'aura appris l'importance du respect de l'échéance des étapes intermédiaires du projet. J'estime que ma satisfaction du résultat obtenu vaut largement l'effort fourni. Enfin ce projet m'aura également permis de créer de nouvelles amitiés au sein du groupe.

5.5 Evaluation latérale :

Evalué	Evaluateur 1	Evaluateur 2	Evaluateur 3	Evaluateur 4	Evaluateur 5	Moyenne
Bucquet	20	20	19	20	20	**19,8**
Grippari	18	17	16	20	18	**17,8**
Hamon	18	18	17	20	18	**18,2**
Lataud	18	18	17	20	18	**18,2**
Turin	18	19	18	20	19	**18,8**

7. Bibliographie :

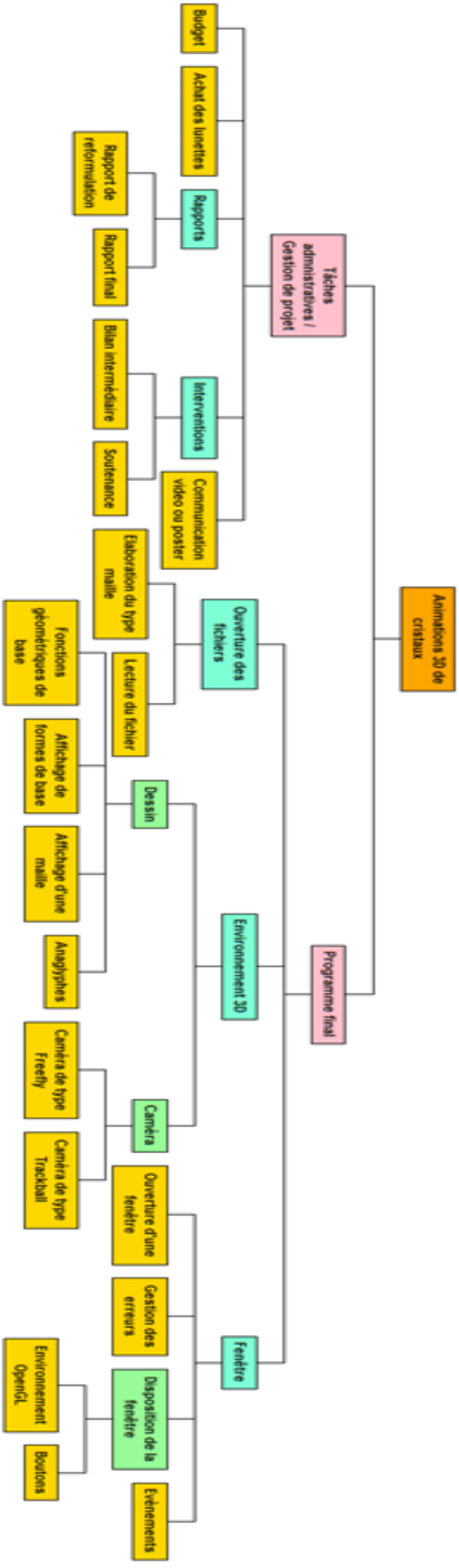
www.lesiteduzero.com/ : Site d'apprentissage de la programmation.

http://nehe.gamedev.net/ : Site de référence pour la gestion de la bibliothèque OpenGL

http://paulbourke.net/texture_colour/anaglyph/ : Bouts de codes pour afficher des anaglyphes.

http://www.crystallography.net/ : Base de données de fichiers .cif.

Annexe 1 :
Organigramme des tâches :



Annexe 2 :
Diagramme de Gantt

effectué après les recherches et la décision de la démarche à adopter