Compte-rendu du comité de pilotage

**Salle** : M209

**Date** : 25 mars 2015 à 15h45

**Participants** : Hervé Quinquenel, Antoine Pinte, Emmanuel Bardière (commanditaire), Vincent De Olivera, Olivier Dissard, Patricia Bordin, Emmanuel Fritsch ; David Niamien, Alban Kraus, Maryame Rhezali (équipe projet)

# Introduction

Olivier Dissard invite l’équipe projet à s’asseoir : nous sommes entre collègues, cette réunion n’est pas évaluée. Le chef de projet (Alban) doit rédiger le compte-rendu de la réunion, et un secrétaire doit noter ce qui est dit lors de la présentation. Lors de cette réunion seront prises des décisions, et on attend que notre rendu final respecte ces décisions, à moins que nous expliquions précisément pourquoi nous n’en aurions pas tenu compte.

# Présentation du projet

Par Alban, David, et Maryame.

Après une présentation des intervenants [diapo 1], Alban annonce le plan de l’exposé : contexte du projet, analyses diverses, proposition de solution technique, récapitulatif sur la gestion du projet [diapo 2]. David prend la parole pour énoncer le contexte du projet [diapo 3]

[diapo 4] Ce projet fait suite à deux autres projets : l’un réalisé au DIAS autour du logiciel de visualisation de nuages de points 3D *CloudCompare*, l’autre à l’IGN dans le cadre du projet iTowns. iTowns est une description de la ville en 3D à partir de véhicules dédiés (Stéréopolis). Les résultats du projet font l’objet d’une diffusion sur Internet via des applications Web.

[diapo 5] David présente les analyses menées pour le projet : une analyse fonctionnelle du besoin du commanditaire synthétisée par un diagramme de cas d’utilisations, suivie d’une analyse technique aboutissant à un diagramme de classes.

[diapo 6] Le besoin du commanditaire est reformulé de la manière suivante : « Réaliser une étude de performance de différentes structures de données afin d’assurer la fluidité de l’affichage d’un très gros nuage de points ». Le diagramme des cas d’utilisation qui suit présente trois fonctionnalités et deux acteurs. La première fonctionnalité est de lire le nuage passé en entrée. La deuxième consiste à découper ce nuage selon trois méthodes différentes : Octree, kd-tree, R-tree. Ces deux fonctionnalités impliquent l’acteur Administrateur. La troisième fonctionnalité, réalisée par l’acteur Utilisateur est intitulée « Afficher les données », ce qui comprend une requête des points situés dans un certain champ de vision, et ensuite leur affichage.

[diapo 7] David entre ensuite dans le détail de ces fonctionnalités. *Lire les données* nécessite aussi d’être capable de stocker les données lues en mémoire. Pour *découper les données*, on doit nécessairement choisir une structure de données, ici sous forme d’arbre, dont il faudra stocker les feuilles et les nœuds. La fonctionnalité d’*Afficher les données* consiste d’une part à obtenir les points à afficher, au moyen d’une requête sur la structure de données, et d’autre part de réaliser leur affichage à l’écran.

[diapo 8] David s’intéresse à présent aux structures de données envisagées. Un *Octree* est une subdivision spatiale de l’espace en trois dimensions de type arbre, dans laquelle chaque nœud peut compter jusqu’à huit enfants (huit octants). En pratique, on calcule un cube entourant le nuage de points, et on le découpe par trois plans parallèles aux faces, pour donner huit petits cubes. Et on recommence. Un des avantages de cette méthode est qu’elle est hiérarchique. Son inconvénient est qu’elle est coûteuse en temps de calcul, et aussi qu’un petit cube ne contenant aucun point sera quand même stocké.

[diapo 9] Maryame prend la parole au sujet du kd-tree. Il s’agit d’une structure de données réalisant une partition de l’espace à k dimensions, où k vaut ici 3. Un plan coupe le nuage à la médiane, puis on réalise la même opération sur les autres dimensions. La racine de l’arbre est une boîte englobant toute la scène. Un voxel est la branche courante de l’arbre à partir de laquelle on cherche à construire deux branches filles, dites sous-voxels. Un nœud est alors défini par un plan séparateur et deux fils, les sous-voxels ; et une feuille est la liste des objets contenus dans le voxel englobant. L’algorithme se termine en fonction du nombre d’objets dans les feuilles et de la profondeur de la subdivision.

[diapo 10] Alban prend alors la parole pour définir les R-tree. Contrairement aux deux arbres précédemment exposés, on ne réalise pas une partition de l’espace par des plans. Il s’agit plutôt de stocker un certain nombre de points, à définir, dans une boîte englobante. On calcule ensuite la boîte englobante d’un certain nombre de ces boîtes, jusqu’à obtenir la boîte englobant le nuage tout entier. Cette méthode est paramétrée par le nombre de points dans les petites boîtes et le nombre de boîtes dans une boîte plus grande.

[diapo 11] Maryame aborde maintenant le diagramme de classes. On commence par une classe arbre, abstraite, qui est généralisée avec une implémentation différente par chacune de nos trois méthodes. Cette classe contient un nœud, qui est en fait le nœud racine, lui aussi une classe abstraite. Ce nœud est lui-même composé de nœuds, mais il est également associé à un voxel, qui est défini au moyen de deux points, et à un plan séparateur. Enfin, les nœuds de dernier niveau sont composés de feuilles, qui sont des fichiers de points.

[diapo 12] Maryame enchaîne à présent sur la solution technique, qui comprend le choix d’une structure de stockage de données, le choix d’un langage de programmation, de librairies externes, et la définition d’indicateurs de performance.

[diapo 13] Pour rappel, le besoin a deux facettes : d’abord, il faut construire le découpage, qui signifie l’écriture de données, et ensuite, il faut récupérer les données, ce qui est une lecture seule. La base de données et le système de fichiers permettent de stocker des données. Pour garantir un accès rapide en lecture, la base de données utilise la redondance (plusieurs pointeurs vers la même donnée) et la réplication de données (copie physique). Le système de fichiers privilégie la navigation entre fichiers voisins. Un système de fichiers est plus facile à utiliser, et répond au besoin du commanditaire : on choisit donc cette solution.

[diapo 14] Maryame parle alors d’un test de vitesse sur les appels récursifs dans l’implémentation de la suite de Fibonacci en différents langages. L’algorithme a été écrit de manière similaire en JavaScript, en Python, et en C/C++. La durée d’exécution dans les langages interprétés (JS et Python) est supérieure à trente secondes et seulement de deux secondes en C++. Le langage C++ a donc été choisi.

[diapo 15] Maryame poursuit sur le choix des librairies. Le choix peut se porter sur GLUT ou sur libQGLViewer, qui est une librairie écrite en C basée sur la bibliothèque Qt. QGLViewer permet la création rapide d’un visualiseur de scènes 3D OpenGL, alors qu’avec GLUT il faut redéfinir la caméra. De plus, en comparant le nombre de frames per second entre GLUT et QGLViewer, il s’avère que QGLViewer est plus rapide ; c’est donc celui qu’on va choisir.

[diapo 16] Alban explique alors que dans un projet d’étude de performance, il est nécessaire de choisir des indicateurs de performance. Étant donné que le but du projet est de comparer les performances d’affichage, les indicateurs portant sur l’affichage sont les plus importants. Assez naturellement, on mesure le nombre d’images par secondes lors d’un déplacement de la caméra. Pour que les résultats donnés par cet indicateur soient corrects, il est nécessaire de codifier le déplacement de la caméra pour qu’il soit le même pour toutes les méthodes. On compare cet indicateur avec le nombre d’images par seconde lorsque la caméra ne se déplace pas. L’indicateur suivant, la vitesse des requêtes, influe sur le temps d’affichage. En plus de ces mesures de temps, on peut réaliser des mesures d’espace grâce à la quantité de mémoire vive maximale utilisée lors d’une requête. Il faut aussi se demander si la mémoire vive disponible est suffisante pour cette tâche. Dans tous les cas, un nuage de points complet prendrait plus de cinquante gigaoctets en mémoire, donc on ne pourra pas lire le fichier brut d’une traite.

Les indicateurs suivants concernent la construction du système de fichiers. Ils sont importants, mais ne mesurent pas à proprement parler les performances d’affichage. Il s’agit du temps et de l’espace mémoire utilisé pour cette opération, et éventuellement un rapport mesurant un éventuel élargissement du nuage de points une fois découpé.

[diapo 17] Alban passe à présent à la partie concernant la gestion de projet.

[diapo 18] Ce projet a la particularité d’être encadré en méthode agile, ce qui se caractérise par des réunions au début et à la fin de chaque séance, où un point est fait sur le travail à faire et le travail réalisé. Le commanditaire est ainsi tenu au courant des avancées du projet, est à même de valider ou d’infirmer rapidement les conclusions de l’équipe projet, et de manière générale, chacun connaît l’avancement et les difficultés des autres. Il a cependant été mis en place des outils plus classiques de gestion de projet : un diagramme de Gantt prévisionnel et réel, un compte-rendu à la fin de chaque séance de travail, et un gestionnaire de versions sur GitHub. Pour résumer la chronologie du projet, l’état d’avancement est en accord avec ce qui a été planifié, néanmoins un risque pourra être la sous-estimation du temps nécessaire au codage due à l’inexpérience de l’équipe projet. À ce titre, les commentaires des membres du Comité sont les bienvenus.

[diapo 19] Alban présente le diagramme de Gantt prévisionnel, qui fait figurer une étude bibliographique, suivie d’un temps de codage, qui précède l’étude expérimentale des performances ; enfin, un peu de temps est alloué à la préparation de la soutenance finale. Le diagramme réel distingue plus précisément la partie bibliographique de l’étude du besoin et de la solution technique. Le codage a été ébauché sur l’implémentation des classes du diagramme de classes, et dans la matinée précédent le comité de pilotage, une version améliorée de l’étude fonctionnelle a été rédigée : Alban en donne un exemplaire imprimé, mais les experts siégeant à ce Comité n’auront pas le temps d’en prendre connaissance pour la suite de la réunion.

# Commentaires

Le commanditaire, Emmanuel Bardière, reproche de suite à la présentation d’amener trop sèchement les fonctionnalités du système.

## Des insertions dans les données découpées

Patricia Bordin, après évocation de la méthode du kd-tree, souhaite s’assurer que cette méthode contrarie l’insertion d’autres nuages de points dans le nuage découpé. Emmanuel Bardière fait alors remarquer qu’il siège à ce Comité non seulement en tant que commanditaire mais aussi en tant qu’expert technique, et qu’en tant que commanditaire il a la capacité de comprendre l’implicite dans l’exposé fait par l’équipe, ce qui n’est pas forcément le cas de ses collègues. Il n’est notamment pas certain que la présentation a suffisamment fait ressortir l’absence du besoin d’insérer des points dans le nuage découpé ; il confirme qu’il n’y en a effectivement pas besoin.

## Les données en entrée

Antoine Pinte demande quel type de fichier sera traité par le projet. Emmanuel Bardière déclare ne pas souhaiter répondre à cette question, et Alban explique alors que l’équipe ne dispose pas encore de donnée et ne peut pas répondre à cette question ; cependant, il a apparemment été compris que le nuage serait vraisemblablement un fichier au format .ply. Emmanuel Fritsch souligne alors que le comportement des algorithmes pourrait varier notablement en fonction du nuage d’entrée ; par exemple, des nuages en filaments tels que des données de tremblements de terre[[1]](#footnote-1), qui peuvent être représentées à plat, pourraient ne pas produire les mêmes résultats que des nuages en volume comme une ville. Suite à ce commentaire, Emmanuel Bardière indique à l’équipe que, pour cette raison, les indicateurs devront être croisés avec la nature du nuage en entrée, ainsi que sa densité : sur mille points, les trois méthodes peuvent donner des résultats semblables, mais sur un million de points ?

## Le livrable

Emmanuel Fritsch reprend alors la parole pour faire remarquer que la présentation ne mentionnait pas non plus le livrable du projet. Emmanuel Bardière répond que le livrable du projet est une application permettant la visualisation d’un nuage et une étude des performances de l’affichage. Emmanuel Fritsch poursuit en remarquant que le livrable aurait dû être mentionné dans l’introduction de la présentation, qu’il ne l’a pas été, ni même dans les documents portés à sa connaissance. Hervé Quinquenel renchérit, déclarant qu’il avait eu l’impression que le livrable devait être semblable à celui d’un projet recherche. Emmanuel Bardière intervient alors en assurant que les trois méthodes de découpage exposées étaient une exigence du commanditaire, et qu’il n’était pas envisagé d’étudier d’autres méthodes pour l’instant.

## Citer les sources

Emmanuel Fritsch demande à revenir sur l’étude des langages de programmation, et souhaite que l’équipe explique l’algorithme utilisé. Alban explique alors la définition de la suite de Fibonacci, pourquoi l’algorithme est qualifié d’exponentiel, et en quoi il est un bon candidat pour un test sur la récursivité : il génère un grand nombre d’appels récursifs et des nombres peu élevés.

Vincent De Oliveira s’étonne de ce test : après une rapide recherche, il n’a pas retrouvé de résultats corroborant les mesures de temps effectuées entre les différents langages de programmation, au contraire semble-t-il. Il demande alors à l’équipe de préciser les modalités du test. Alban explique que l’algorithme a été codé de manière semblable dans les trois langages ; le JavaScript a ensuite été interprété dans [Mozilla] Firefox, Python a été interprété, et le C a été compilé avec GCC sans optimisation. Il discute également la pertinence de comparer un langage compilé avec des langages interprétés. Vincent De Oliveira explique alors que l’interprétation de JavaScript dans un navigateur induit un biais dans les résultats, car le navigateur consomme de la mémoire vive ; et il est heureux que l’équipe n’ait pas utilisé le DOM, qui ralentit l’exécution. Il conclut que la mesure aurait dû être effectuée avec nodejs[[2]](#footnote-2).

Emmanuel Bardière poursuit la conversation en réclamant la source du tableau présentée à la diapo 15 à propos du choix de la librairie. Sachant que libQGLViewer utilise GLUT en interne, il est très étonné de tels résultats. Maryame se souvient de la source, et affirme son intention de l’ajouter à la présentation[[3]](#footnote-3). De même, Emmanuel Bardière souhaiterait que la source et la configuration matérielle utilisée lors des tests sur les langages soit mentionnée. Emmanuel Fritsch poursuit sur la comparaison des librairies : il a l’impression que l’exposé s’est étendu sur libQGLViewer et que trop peu de renseignements sur les autres librairies ont été collectés.

Patricia Bordin revient à la diapo 14 où sont présentés les indicateurs. Elle demande quelles ont été les sources bibliographiques. L’équipe commence à répondre que certains sont intuitifs. Emmanuel Bardière intervient pour souligner qu’aucun indicateur n’a été imposé par le commanditaire. Cependant, l’équipe projet mentionne que certains des indicateurs ont été vaguement évoqués au cours de discussions avec lui. Emmanuel Bardière conclut en reprochant à l’équipe de ne pas avoir parlé de bibliographie à propos de ces indicateurs.

Olivier Dissard poursuit la discussion, en relevant le fait que le projet ne dispose d’aucune indication sur le fonctionnement du matériel. Il raconte alors l’exemple suivant : un projet travaillait sur la compression JPEG2000, mettant en œuvre deux algorithmes ayant le même principe et sensiblement le même codage. Cependant, il fut mesuré que l’un était bien plus rapide que l’autre ; l’explication de ce phénomène étant *a priori* malaisée. Il fut finalement avéré que la différence résidait dans l’organisation du fichier produit par chacun de ces algorithmes : le plus rapide lisait un fichier où figuraient en premier des créneaux grossiers, qui se précisaient par la suite. Olivier Dissard revient alors à la présentation, expliquant que l’équipe a bien expliqué la structure des données, mais n’a pas expliqué comment le matériel percevait cette structure. Cette partie doit faire l’objet d’une étude bibliographique complémentaire. Il fait appel à la remarque de Vincent De Oliveira, précisant que comprendre comment fonctionne le matériel était bien plus important que savoir appuyer sur un bouton pour déclencher un test. Antoine fait écho à cette remarque, en expliquant que le même code optimisé différemment ne produira pas le même résultat.

Olivier Dissard souligne finalement que toute comparaison est impossible tant qu’on ne connaît pas la structure des fichiers issus du découpage : cela doit faire l’objet d’une recherche bibliographique.

## Une caméra qui se déplace

Au vu de notre indicateur numéro 1[[4]](#footnote-4), Antoine Pinte se demande comment l’équipe va s’y prendre pour assurer le même déplacement de la caméra pour toutes les méthodes. Alban propose de ne pas permettre à l’utilisateur de déplacer la caméra lors de la mesure de cet indicateur, et que la trajectoire et la vitesse de la caméra soient déterminés, par exemple dans un fichier. Antoine Pinte demande alors si le temps statique concerne bien l’affichage seul.

Emmanuel Fritsch intervient : lors de la mesure en caméra statique, récupère-t-on tous les points ? Emmanuel Bardière le nie : le premier indicateur est mesuré lors d’un déplacement de la caméra, le deuxième lorsque la caméra est statique, où on n’a alors pas de requête à faire. Emmanuel Fritsch demande alors la différence entre les deux indicateurs. Emmanuel Bardière répond que dans le deuxième indicateur, on ne fait pas de requête. Emmanuel Fritsch oppose que dans ce cas statique, l’écran est rafraîchi à une vitesse maximale. Emmanuel Bardière rappelle alors le principe du double-buffering : on écrit dans l’un quand l’autre est affiché, puis on échange les deux ; et même lorsque la caméra est statique on observe une variation de la vitesse de rafraîchissement.

## Conclusion

Olivier Dissard conclue alors la séance. L’application développée au cours du projet a besoin de neutralité ; de plus, les acteurs du projet doivent être capables de comprendre les mécanismes matériels impliqués lors des tests, ce qui manque encore. La première phase de défrichage du projet a été bien réalisée, il reste à présent à concevoir une application qui répond intelligemment. Il faut à présent réaliser un compte-rendu du comité de pilotage, en mentionnant tous les points techniques abordés. La suite du projet sera à voir en concertation avec Emmanuel Bardière dans le cadre de la méthode agile. Le rendu du projet devra répondre aux exigences exposées au cours de la réunion.

Emmanuel Bardière dit que le projet est ambitieux.

1. De telles données ont été vues en cours [↑](#footnote-ref-1)
2. Utilisé en cours [↑](#footnote-ref-2)
3. Ce qui a été fait, voir sur GitHub [↑](#footnote-ref-3)
4. Nombre d’images par seconde lors d’un déplacement de la caméra [↑](#footnote-ref-4)