Structures de données II

Enoncé du projet

Université de Mons – Année académique 2021– 2022

Professeur V. Bruyère – Assistant G. Devillez

Le projet se fait par groupe de 2 étudiants et comporte les étapes suivantes :

1. Lecture et compréhension du sujet et de l'énoncé du problème ;
2. Résolution d'un exercice préliminaire concernant la structure de données utilisée pour ce projet ;
3. Implémentation et tests ;
4. Remise du code, du rapport final et entretien.

Certaines ressources (e.g., exemples donnés pour les tests) seront déposées en temps voulu sur le site du cours de la plate-forme e-learning [1].

1 Sujet et référence

Une copie du Chapitre 12 (*Binary Space Partitions*) du livre *Computational Geometry* [2] est disponible sur la plate-forme e-learning. Cette référence constitue votre ressource principale sur le sujet.

Ce chapitre traite du problème de la visualisation d'objets en 3D, comme par exemple dans les jeux vidéos, qui peut être résumé comme suit :

> *À partir d'un point de vue donné (coordonnée et orientation dans l'espace de la "caméra" ou de l'œil), comment rendre une scène composée d'objets en 3D, de telle sorte qu'on n'affiche pas les (parties des) objets qui sont cachés par d'autres.*

En ce qui concerne le projet, il se limite à la visualisation d'objets dans un plan (en 2D) à partir d'un œil situé dans ce même plan. Cette situation est également traitée dans le chapitre.

La solution proposée est d'utiliser un type particulier d'arbres binaires de recherche : les arbres BSP (*Binary Space Partitions trees*). À partir de cette structure de données, on peut répondre au problème évoqué ci-dessus, en appliquant un algorithme appelé *l'algorithme du peintre*.

Vous commencerez donc le projet par la lecture de ce chapitre qui définit les arbres BSP, explique comment on peut les construire, et donne l'algorithme du peintre. Vous noterez que la construction des arbres BSP peut être faite de différentes manières (voir également ci-dessous, Section 2).

Remarques importantes :

Dans ce projet, chaque objet sera un segment coloré dans un plan orthonormé. Un objet sera donc entièrement déterminé par deux couples de points (extrémités) et une couleur.

1. `https://moodle.umons.ac.be/course/view.php?id=184`
2. de Berg, M. , van Kreveld, M., Overmars, M. and Schwarzkopf, O., *Computational Geometry – Algorithms and Applications*, Second Ed., Springer, 2000.

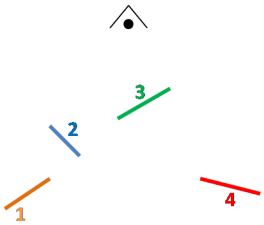

FIGURE 1 – Exemple de représentation d'un ensemble d'objets et d'un œil.

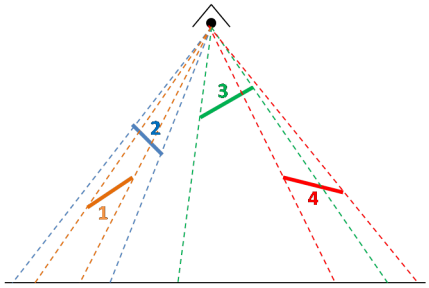

FIGURE 2 – Représentation en 2D de la projection par rapport à l'œil.

FIGURE 3 – Représentation de ce que l'œil voit dans la scène représentée par la figure 1

Le but est d'obtenir, à partir d'un ensemble d'objets et d'un œil (exemple : figure 1), la représentation graphique de ce que voit l'œil (exemple : figure 3).

Vous pouvez remarquer dans le resultat obtenu à la figure 3, qu'il n'y a pas de trace de l'objet 1. Cela est dû au fait que l'objet 2 le cache complètement (cette situation peut être mieux visualisée en figure 2). Ce phénomène se produit également sur une partie de l'objet 4 cachée par l'objet 3.

2 Enoncé du problème

L'énoncé du problème principal est le suivant : Etant donné un arbre BSP représentant une scène dans un plan (ensemble de segments) et un œil (un point dans ce plan), donner un algorithme qui affiche la représentation graphique de ce que voit l'œil.

Un format de fichier permettant de stocker des scènes en 2D (c'est à dire une liste de segments et de leur couleur associée) vous sera donné sur la plate-frome e-learning. Le design de vos classes doit donc également tenir compte de cette particularité : il faudra pouvoir lire et charger ces fichiers.

Il y a deux étapes importantes à la réalisation du problème :

1. *la construction d'un arbre BSP représentant la scène.*

Comme expliqué dans le Chapitre *Binary Space Partitions*, une même scène peut être représentée par plusieurs arbres BSP différents. En effet, la construction de l'arbre BSP dépend de la manière dont on partitionne l'espace. Il existe différentes heuristiques qui précisent comment on peut découper l'espace : aléatoirement, en prenant le premier segment dans la liste, en essayant de minimiser le nombre d'objets découpés par une partition,...

Ces arbres BSP peuvent être très différents en termes de taille (nombre de noeuds) et de hauteur.

En ce qui concerne la construction des arbres BSP, chaque groupe d'étudiants fournira :
- une heuristique construisant un arbre BSP aléatoirement (tel que décrit dans le chapitre du livre) ;
- une heuristique construisant un arbre BSP en choisissant de partitionner l'espace en prenant la droite qui contient le premier segment de la liste, puis le second, ...
- une autre heuristique spécifique pour créer un arbre BSP et qui sera attribuée à ce groupe d'étudiants ultérieurement.

2. *l'application de l'algorithme du peintre (problème principal).*

   Une fois un arbre BSP construit, on peut alors appliquer l'algorithme du peintre pour obtenir la représentation graphique de ce que voit l'œil. Comme cet algorithme doit parcourir tous les noeuds de l'arbre, le choix de l'heuristique permettant d'obtenir l'arbre aura une influence sur sa rapidité. Vous fournirez un calcul détaillé de la complexité dans le pire des cas de cet algorithme.

3 Exercice préliminaire

Pour se familiariser avec les arbres BSP, il vous est demandé de réaliser l'exercice préliminaire suivant :

Etant donné un arbre BSP représentant une scène dans un plan (ensemble de segments) et deux points x et y dans ce plan, donnez un algorithme récursif en pseudo-code qui indique si le segment d'extrémités x et y appartient à la scène. Veuillez accompagner votre algorithme :
- d'une explication de son fonctionnement ; et
- d'une discussion autour de sa complexité (ne vous limitez pas au pire des cas).

Nous convenons, pour cet exercice, que les segments contenus dans un même nœud de l'arbre BSP sont stockés dans une liste chaînée. Pour votre implémentation finale, vous êtes libres de choisir la structure qui vous semble la plus adaptée.

Remarque : Cet exercice préliminaire n'est qu'une mise en route du projet. Il ne sera pas nécessaire à la résolution du problème principal.

Cet exercice préliminaire est l'occasion de vérifier si vous avez bien compris l'énoncé du projet mais aussi d'avoir un retour sur la qualité de votre rédaction et la structure de votre rapport. Ne négligez donc pas cet exercice et n'oubliez pas de définir les notions et structures importantes.

4 Remise finale

Après la remise de l'exercice préliminaire, vous pourrez commencer à travailler sur le sujet principal du projet. Ceci nécessitera de fournir un code répondant aux contraintes de la sous-section 4.1 et un rapport complet contenant les éléments demandés dans la sous-section 4.2.

Le projet sera évalué aussi bien sur le code que vous remettrez que sur le rapport. Veillez donc à accorder suffisamment d'attention aux deux. Un code ou un rapport de trop mauvaise qualité peut se solder par un échec pour le projet même si l'autre était de bonne qualité.

4.1 Programme

Vous implémenterez en Java votre projet et utiliserez les exemples de scènes en 2D – qui seront donnés ultérieurement – pour le tester. Ces scènes ne sont que des exemples et il doit donc être possible d'utiliser d'autres scènes facilement.

Votre code devra être documenté (format javadoc) et sera testé avec la version 11 de Java. Le résultat final est d'avoir deux programmes de test pour vos classes :

1. Un programme interactif qui doit permettre de :
   — charger un fichier contenant une scène en 2D ;
   — choisir la position du point de vue (de l'œil) ;
   — choisir une des heuristiques de construction des arbres BSP disponibles ;
   — afficher la représentation graphique de ce que voit l'œil en appliquant l'algorithme du peintre.
2. Un programme "console" qui permet de comparer les différentes heuristiques de construction des arbres BSP, mais qui n'affiche pas de représentation graphique de ce que voit l'œil à l'écran. Ces heuristiques seront comparées en fonction des critères suivants :
   — la taille de l'arbre BSP généré par l'heuristique,
   — la hauteur de cet arbre,
   — le temps CPU utilisé pour construire cet arbre,
   — le temps CPU utilisé pour appliquer l'algorithme du peintre sur cet arbre.

Les classes contenant les méthodes `main` permettant de lancer la version interactive et la version console de votre programme seront clairement identifiées par un nom commençant par `Test`.

4.2 Rapport final

Votre rapport final doit contenir :
— un diagramme UML – reflétant la structure de votre programme – accompagné d'une description des différentes classes et interfaces utilisées ;
— les explications en français de chaque structure de données *importante* utilisée ;
— les explications en français du principe et du fonctionnement de chacun de vos algorithmes *importants* ;
— une note sur la complexité dans le pire des cas de chacun de vos algorithmes *importants* ;
— une section donnant et commentant les résultats de vos tests de comparaison des différentes heuristiques de construction des arbres BSP ;
— un court mode d'emploi de votre programme (comment compiler, exécuter et utiliser votre programme)
— une conclusion comprenant une réflexion sur le projet (apports, difficultés, comparaison de vos résultats avec la théorie, ...).

Le but de ce rapport final est de pouvoir comprendre grâce à sa lecture (par une personne ne connaissant pas le projet) :
— votre raisonnement ;
— votre implémentation ;
— et votre résolution du problème.

N'oubliez donc pas de bien définir les concepts et structures importants. Il faut aussi bien expliquer votre façon de comparer les heuristiques (comment vous obtenez les données, comment vous comparez les heuristiques, sur quels aspects, ...).

5 Calendrier

— **vendredi 12 novembre 2021 : composition des groupes**
Pour cette date, vous aurez communiqué la composition de votre groupe à l'endroit prévu à cet effet sur Moodle. Le projet se fait par groupe de **deux étudiants**.

— **jeudi 23 décembre 2021 à 12h00 : remise du rapport de l'exercice préliminaire**
Vous déposerez via la plate-forme e-learning votre rapport au format pdf comprenant vos réponses aux questions. Si un groupe désire un feed-back sur son rapport, il peut prendre rendez-vous par e-mail avec Gauvain Devillez (`gauvain.devillez@umons.ac.be`).

— **mardi 19 avril 2022 à 12h00 : remise du code et du rapport final**

a) *Code*
Vous déposerez via la plate-forme e-learning une archive (zip, jar, tgz,...) *ne contenant que les fichiers .java* de votre programme et les fichiers utiles à la compilation. Vous pouvez donc joindre les fichiers gradle, ant, maven, ... mais il n'est pas nécessaire de joindre les scènes.

b) *Rapport final*
Vous déposerez via la plate-forme e-learning votre rapport final, au format PDF.

— **Après le mardi 19 avril 2022 : défense orale**
Une date et un horaire de passage pour une défense seront communiqués à chaque groupe. Lors de cet entretien, vous présenterez brièvement la manière dont vous avez abordé le projet (15 minutes), et des questions seront ensuite posées **à chaque étudiant** du groupe.

6 Remarques supplémentaires

1. Les dates des dépôts sont strictes. Le site n'acceptera plus de dépôt après midi.
2. Si vous avez des questions – pendant toute la durée du projet – vous prendrez rendez-vous par e-mail avec G. Devillez (`gauvain.devillez@umons.ac.be`).

Bon travail !