Structure de données II
Rapport de projet

HUYLENBROECK Florent
DACHY Corentin
BA3 Info

15 avril 2019

Contents

1 Introduction

1.1 Notions théoriques

- Une **scène** S est un ensemble de segments colorés dans un repère fini.
- Un **BSP** (*Binary Space Partition* ou *partition binaire de l'espace*) τ est un arbre binaire permettant de représenter une scène en fragmentant celle-ci en sous-espaces convexes, lesquels contiennent au plus un segment. Son fonctionnement est le suivant :

  Soit v la racine de τ.

  - Si v est une feuille, alors $S(v)$ (sous-ensemble de segments de la scéne contenus dans v) contient 0 ou 1 segment.
  - Sinon v est un noeud interne du BSP. Dans ce cas, v contient une équation de droite de la forme $\ell \equiv ax + by + c = 0$ et un ensemble $S(v)$
    Le fils gauche v^- de v sera la racine du sous-BSP τ^- décrivant la partie de la scène située sous la droite ℓ, et le fils droit v^+ de v sera la racine du sous-BSP τ^+ décrivant la partie de la scéne située au-dessus de la droite ℓ.
    $S(v)$ contient les segments totalement inclus à la droite ℓ. Leur nombre n'est pas limité comme dans les feuilles.

  Si une droite de séparation ℓ_v d'un noeud interne v intersecte un segment en un point autre qu'une extrémité, ce segment est coupé en deux parties et celles-ci sont ajoutées dans τ^- et τ^+ en conséquence.
- La **taille** t d'un BSP est le nombre total de segments contenus dans chacuns des noeuds de celui-ci.
- Une **heuristique** de construction d'un BSP intervient au moment de choisir le segment de la scène par lequel passera la prochaine droite de séparation ℓ.

1.2 Enoncé du problème

Pour ce projet, il nous a été demandé à partir de fichier *.txt* au format Scene2D de :

- Construire un BSP de la scène décrite dans le fichier.
- Appliquer l'algorithme du peintre sur ce BSP.
- Fournir une application ayant deux modes, console et graphique, afin de tester différentes heuristiques de construction pour les BSP. L'application console doit aussi mesurer la hauteur/taille du BSP, le temps CPU pris par la construction et par l'excecution de l'algorithme du peintre (*Voir section suivante).*
- Trois heuristiques nous ont été imposée pour ce projet :
  - RANDOM - Le segment est choisi au hasard parmis $S(v)$.
  - ORDERED - Le segment choisi est le premier dans $S(v)$.
  - FREE_SPLIT - Le segment choisi est le premier dans $S(v)$ dont les deux extrêmités sont chacune sur une droite de séparation ℓ. Si aucun segment satisfaisant ces condtitions n'est trouvé dans $S(v)$, alors le segment y est choisi au hasard.

2 Calcul de la complexité de l'algorithme du peintre

Introduisons la notion d'*oeil* e dans une scène S. Celui-ci servira de point de vue des éléments de la scène. L'algorithme du peintre est un algorithme simple de rendu d'une scène sur un objet de dimension inférieure. (*Par exemple, afficher la vue en 2D par un oeil d'une scène en 3D.*) Il permet de déterminer quels fragments de la scène sont visible par l'oeil en fonction de sa position dans le repère.

Algorithm 1 painter's algorithm

Entrées : v : Noeud, racine du BSP.
e : Point, position de l'oeil dans la scène.

Sorties : /

Effets : Les appels à la fonction de dessin sont effectués dans l'ordre correct d'affichage pour l'oeil e.

```
1: procedure PAINTERSALGORITHM(A, B)
2:     if v est une feuille then
3:         Dessiner les fragments de S(v)
4:     else
5:         if e est au-dessus de la droite de séparation ℓ de v then
6:             PAINTERSALGORITHM(v+, e)
7:             Dessiner les fragments de S(v)
8:             PAINTERSALGORITHM(v−, e)
9:         else if e est en desous de la droite de séparation ℓ de v then
10:            PAINTERSALGORITHM(v−, e)
11:            Dessiner les fragments de S(v)
12:            PAINTERSALGORITHM(v+, e)
13:        else
14:            PAINTERSALGORITHM(v+, e)
15:            PAINTERSALGORITHM(v−, e)
16:        end if
17:    end if
18: end procedure
```

NOTE : la fonction *Dessiner les fragments de* $S(v)$ est en $O(s)$ où s est le nombre de segments dans $S(v)$.

La complexité de l'algorithme du peintre va dépendre de la taille de τ. Elle va donc dépendre de la fragmentation des segments de S lors de la construction de τ.
On remarque en effet que tout le BSP est parcouru par l'algorithme car dans chaques cas, en partant d'un noeud interne, on applique l'algorithme sur ses deux fils. La récursion s'arrête uniquement lorsqu'on arrive à une feuille. Nous sommes donc pour l'instant en $O(n)$ où n est le nombre de noeud de τ

A chaque noeud, l'algorithme dessine tous les segments contenus dans $S(v)$ (sauf quand e se situe sur la droite de séparation ℓ du noeud, auquel cas il n'est pas nécessaire de dessiner les fragments dans $S(v)$ car ils ne seront pas visible depuis e). Nous avons donc une complexité de $O(n \cdot s)$ ce qui donne, par la manière dont nous avons défini la taille d'un BSP, $O(t)$.

3 Comparaison des heuristiques

4 Mode d'emploi du logiciel de test

Mode graphique

Ouvrir un terminal, se déplacer dans le dossier *code* du logiciel et executer la commande :

```
$ ant testgui
```

Le logiciel consiste en une fenêtre séparée en deux parties. La partie du haut permet de visualiser le BSP et celle du bas ce que voit l'oeil. Afin d'ouvrir un BSP, il faut se rendre dans le menu *Actions* → *Open* et choisir un fichier Scene2D valide. Il est possible de changer l'heuristique de construction via le menu *Heuristics*.

Mode console

Ouvrir un terminal, se déplacer dans le dossier *code* du logiciel et executer la commande :

```
$ ant test
```

Il est possible de spécifier les conditions de départ des test grâce aux paramètres suivants :

- *path* - Le chemin jusqu'au fichier Scene2D. La valeur par défaut est "octogone.txt".
- *n* - Le nombre de répétitons du test. La valeur par défaut est 100.
- *ex* - L'abscisse de l'oeil. La valeur par défaut est 0.
- *ey* - L'ordonnée de l'oeil. La valeur par défaut est 0.

Pour modifier ces paramètres, il faut utiliser le modificateur *-D*.

```
$ ant test -Dpath=somepath.txt
```

Pour modifier plusieurs paramétres, plusieurs *-D* sont nécessaires.

```
$ ant test -Dpath=somepath.txt -Dn=1000 -Dex=10.05 -Dey=3.141
```

Javadoc

Ouvrir un terminal, se déplacer dans le dossier *code* du logiciel et executer la commande :

```
$ ant doc
```

5 Conclusion