M3, Infographie 2D. Examen

jeudi 18 décembre 2003

La précision et la clarté de votre rédaction sont *fondamentales*. Chaque réponse doit être accompagnée d'une justification. Lisez attentivement l'énoncé avant toute tentative de raisonnement. Le barème est donné à titre indicatif. Cours et TD sont autorisés. Durée 3 heures.

Exercice 1. [4 pts] Soient $u = (x_u, y_u)$ et $v = (x_v, y_v)$ deux vecteurs du plan euclien. On définit la quantité

$$S(u,v) := \begin{vmatrix} x_u & x_v \\ y_u & y_v \end{vmatrix} = x_u y_v - x_v y_u. \tag{1}$$

1. [2 pts] Démontrez que $|S(u,v)|$ est la surface en gris délimitée par le parallélogramme de la figure 1 et que le signe de $S(u,v)$ est celui de l'angle $\widehat{uOv}$, i.e. positif dans le sens trigonométrique, négatif dans le sens des aiguilles d'une montre.

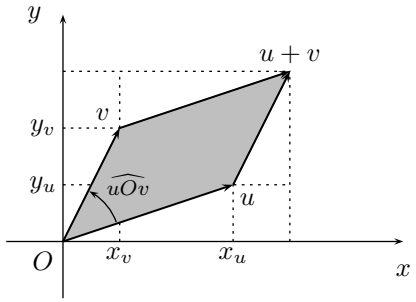

FIGURE 1. Calcul de $S(u,v)$

2. [2 pts] Écrivez un algorithme $\mathsf{Sens}(u,v)$ qui renvoie 0 si les vecteurs u et v sont colinéaires, et le signe (± 1) de l'angle $\widehat{uOv}$ sinon. Quelle est la complexité de cet algorithme ?

Exercice 2. [9 pts] L'objectif de ce problème est d'étudier l'algorithme de *Jarvis* pour déterminer l'enveloppe convexe d'un ensemble de points. Le principe est très simple, il consiste à tendre deux cordes autour des points en partant du point B d'ordonnée la plus petite (et d'abscisse la plus petite en cas d'égalité), une par la gauche, l'autre par la droite. Ces deux cordes se rencontrent sur le point H le plus haut.

L'algorithme consiste à déterminer itérativement chaque point de contact P entre la corde et le nuage de points en commençant par $P := B$. Pour le parcours droit (le principe est symétrique pour le parcours gauche), le point de contact suivant N est celui qui a le plus petit angle polaire avec P, i.e. tel que $\widehat{PxN}$ est minimal (en cas d'égalité, on choisit le plus éloigné).

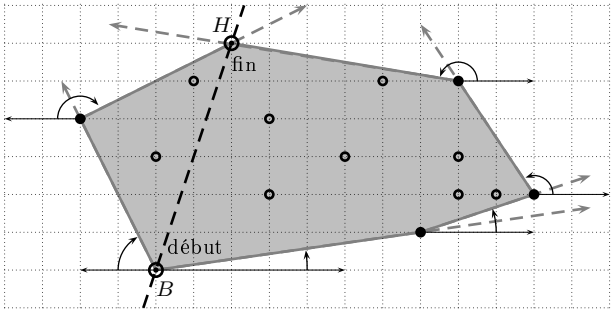

FIGURE 2. Application de l'algorithme de Jarvis.

1. [2 pt] Sans écrire l'algorithme formellement, estimez sa complexité.

Plutôt que de calculer l'ensemble des angles polaires $\widehat{PxN}$ avec les points N restants et de choisir le plus petit Q, on peut exploiter avantageusement les résultats du premier exercice sur la fonction S définie en (1). Le point de contact courant est P et on initialise le point de contact suivant Q à l'un des points restants. Pour tous les points N qui ne sont pas à gauche du segment $[BH]$, on calcule $S(\overrightarrow{PQ}, \overrightarrow{PN})$. Si S est négatif, alors on met à jour $Q \leftarrow N$ (cf. (1)).

2. [1 pt] Montrez qu'en procédant de la sorte, on obtient bien le point qui forme l'angle polaire le plus petit.

3. [2 pt] Ecrivez l'algorithme $\mathsf{SuivantD}(P)$ qui renvoie le prochain point de contact Q dans le parcours à droite à l'aide de l'algorithme Sens.

4. [2 pt] Calculez la complexité de cet algorithme.

5. [2 pt] Écrivez l'algorithme $\mathsf{EC}(M)$ qui renvoie l'enveloppe convexe d'un ensemble de n points $M_1, M_2, \ldots, M_n$ (tableau $M[1,n]$) sous la forme d'un tableau contenant les indices des points de l'enveloppe dans l'ordre trigonométrique de parcours en partant de B. Vous utiliserez les algorithmes $\mathsf{SuivantD}$ et son symétrique $\mathsf{SuivantG}$.

6. [2 pts] Calculez la complexité de cet algorithme et comparez la à celle que vous avez obtenue dans la première question.

Exercice 3. [7 pts] L'outil `cron` des systèmes UNIX permet à chaque utilisateur de programmer des tâches à exécuter de manière récurrente (par exemple une sauvegarde toutes les nuits à 2 heures). La planification de ces tâches est stoquée dans un fichier texte dédié, la *crontable*, sous forme de règles (une ligne par règle et par tâche). Il existe un fichier par utilisateur ayant recours à `cron`. C'est le démon (`crond`) qui se charge

d'exécuter ces tâches en se réveillant toutes les minutes. Ces fichiers sont en général placés dans `/var/spool/cron/`. La *crontable* d'un utilisateur `toto` est stoquée dans un fichier du même nom `toto` dans le répertoire `/var/spool/cron/`). L'édition des fichiers se fait nécessairement par le biais de la commande `crontab`. Tandis qu'un fichier de log est maintenu par le démon dans `/var/log/cron`.

Exemple : la ligne (règle)

```
0 22 * * * /home/toto/bin/archivage &
```

dans le fichier *crontable*, planifie le lancement de l'exécutable `archivage` tous les soirs à 22 heures.

Le fichier associé à la *crontable* obéit à la syntaxe suivante : une ligne ⇔ une tâche à planifier et chaque ligne doit comporter obligatoirement 6 champs dans cet ordre :

(1) les minutes de 0 à 59,

(2) les heures de 0 à 23,

(3) le jour du mois de 0 à 31,

(4) les mois de 0 à 12,

(5) le jour de la semaine de 0 à 7 (de dimanche à samedi),

(6) la tâche à exécuter.

Valeurs particulières :

- Le caractère `*` est un caractère particulier qui signifie d'utiliser toutes les valeurs de la plage de temps (colonne 1 à 5)
- La notation `-` : 2 nombres séparés par un tiret indique un intervalle de temps (exemple : `0-15` dans la première colonne, indique toutes les minutes comprises entre 0 et 15)
- La notation `,` : 2 nombres ou plus séparés par des virgules : indique une liste de valeurs (`0,15,30,45` dans la première colonne indique les minutes 0, 15, 30 et 45, c'est-à-dire toutes les quarts d'heure).
- La notation `/` combinant des valeurs au moyen d'un pas : permet de désigner des incréments de valeurs (exemple : `*/2` indique toutes les valeurs paires, `*/10` toutes les 10 minutes).

Il est bien sûr possible de combiner les différentes notations, ainsi `0-15/2` indique toutes les valeurs paires des minutes comprises entre 0 et 15). L'objectif de l'exercice est de proposer un outil simple avec une interface graphique écrite en Tk permettant de lire et d'éditer la *crontable* de l'utilisateur. L'interface présentera une liste de règles définies dans la *crontable*.

1. [2 pt] En se basant sur la norme CUA (on considère que l'application est un outil de modification de paramètres), décrire en quelques lignes une interface pour cette application.

2. [2 pt] Écrire le code d'un outil en Tcl-Tk permettant de lire et d'afficher le contenu de la *crontable*. On présentera en particulier les portions de codes permettant la lecture des informations de la *crontable* utilisateur. Il sera fait usage de la commande `crontab -l` qui affiche le contenu sur la sortie standard (on ne cherchera pas à accéder directement au fichier en `/var/spool/cron`). Les règles ainsi récupérées seront affichées dans une liste. Un bouton permettra de quitter l'outil.

3. [2 pt] On veut maintenant pouvoir éditer les règles de la *crontable* :

(a) Un bouton AJOUTER qui doit permettre d'ajouter une nouvelle règle. On propose une transformation de l'interface utilisateur : le contenu de la règle sera spécifié par l'utilisateur dans un champ de texte dédié et un click sur un bouton permettra d'ajouter la règle dans la liste. Donnez le code correspondant à ces nouveaux éléments de l'interface graphique. (On ne cherchera pas dans cette question à vérifier la syntaxe de la règle donnée par l'utilisateur).

(b) Décrire l'interface et le code mettant à jour la *crontable* utilisateur. On utilisera la commande UNIX `crontab <nomDuFichier>` qui remplace le contenu de la *crontable* utilisateur par le contenu du fichier spécifié (la commande `crontab` utilise l'entrée standard UNIX si le nom de fichier spécifié est `'-'`).

(c) Des erreurs peuvent survenir pendant l'exécution de cette commande (dues à des erreurs de syntaxe dans les règles). Décrire en quelques lignes les mécanismes pouvant être mis en œuvre pour les prendre en compte.

4. [2 pt] Modification d'une règle existante : on veut étendre le mécanisme précédent pour permettre l'édition de règles existantes. Lorsque l'utilisateur sélectionne une règle dans la liste, il doit pouvoir supprimer la règle ou la modifier. Tout en gardant la possibilité d'ajouter de nouvelles règles.

(a) Proposer le code gérant l'affichage d'une fenêtre permettant l'édition d'une règle. Proposer une procédure permettant de créer un telle fenêtre pour toute règle passée comme argument.

(b) Un bouton MODIFIER dans la fenêtre principale permet d'éditer la règle sélectionnée dans la liste en créant une fenêtre d'édition utilisant le code proposé en 4.(a). De même le bouton AJOUTER ajoute une nouvelle entrée avec des paramètres par défaut dans la liste et ouvre une fenêtre d'édition en s'appuyant sur cette nouvelle règle. Enfin un bouton SUPPRIMER permet de retirer une règle dans la liste de règles de la fenêtre principale. Décrire le code correspondant en Tcl-Tk.

(c) Pour éviter que l'utilisateur ne supprime une règle alors que d'autres règles sont en cours d'édition on se propose de transformer la fenêtre décrite en 4.a comme une boîte de dialogue modale. Décrire les transformations.