Programmation et Structures de Données 1995-96
Projet 1: Récursivité et graphisme

Frédéric Vivien, bureau 306, e-mail: Frederic.Vivien

29 septembre 1995

1 Documentation sur le graphisme en *Caml Light*

Une documentation complète est disponible à l'URL suivant (à ouvrir avec `mosaic` ou `netscape`):

`http://www/~fvivien/Enseignement/caml-graph.html`

Voici la marche à suivre pour pouvoir utiliser le graphisme en Caml. Les instructions ci-dessous constituent une recette de cuisine quelque peu magique...

1. Créer un "toplevel" qui contiendra toutes les fonctions graphiques par la commande suivante:

   ```
   camlmktop -o camlgraph -custom unix.zo graphics.zo -lunix
   -lnsl -lgraph -lX11 -lsocket
   ```

   Ce "toplevel" doit être créé dans la "command-tool" ou "shell-tool" dans laquelle vous lancerez `camllight` *avant* ce lancement.

2. Appeler camllight avec ce "toplevel": `camllight camlgraph`

3. Enfin, ouvrir le module `graphics` avec la commande (sous camllight): `#open "graphics";;`. Attention au `#` devant `open`, il fait partie de la commande, ce n'est pas le symbole d'invite.

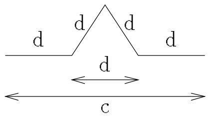
Voici une liste non exhaustive de fonctions de graphismes qui pourraient vous être utiles... Vous pourrez trouver une description **en français** de ses fonctions dans le Manuel de référence de Caml.

- `value open_graph: string->unit` Show the graphics window or switch the screen to graphic mode. The graphics window is cleared. The string argument is used to pass optional information on the desired graphics mode, the graphics window size, and so on. Its interpretation is implementation-dependent. If the empty string is given, a sensible default is selected.
- `value close_graph: unit->unit` Delete the graphics window or switch the screen back to text mode.
- `value clear_graph: unit->unit` Erase the graphics window.
- `value_size_x: unit ->unit`, `value_size_y: unit ->unit` Return the size of the graphics window. Coordinates of the screen pixels range over 0 ... `size_x()-1` and 0 ... `size_y()-1`. Drawings outside of this rectangle are clipped, without causing an error. The origin (0,0) is at the lower left corner.

- `value set_color: color->unit` Set the current drawing color.
- `value moveto : int ->int ->unit` Position the current point.
- `value current_point: int ->int * int` Return the position of the current point.
- `value lineto : int ->int ->unit` Draw a line with endpoints the current point and the given point, and move the current point to the given point.

2 Le flocon de von Koch

Pour le flocon de Von Koch, il faut définir le motif de base dont la taille dépend du paramètre c et de la génération n. Le motif de base du flocon est le suivant:



À la génération 0, on avance simplement de la longueur d'un côté, c. Sinon, il faut tracer les quatre morceaux du motif de base, ces quatre morceaux étant eux mêmes des motifs.

Écrire une procédure `avance distance angle` qui trace à partir du point courant un segment de longueur `distance` dans la direction définie par `angle`.

Écrire une procédure récursive `motif` qui trace les quatre motifs de la génération $n+1$ en s'appellant pour tracer des motifs de génération n. Cette procédure sera donc de la forme:

```
#let motif generation distance angle =
#    if generation = 0
#       then avance distance angle
#       else begin
#            ...
#            end;;
```

À partir de la procédure `motif` dessiner un flocon en plaçant un motif de génération n sur chaque côté d'un triangle équilatéral.

Modifiez ensuite légèrement le motif (angles, longueurs des segments, ...) pour obtenir des flocons différents.

Directive

Le problème tel qu'il est énoncé ici peut-être résolu avec la seule récursivité. En particulier tous les paramètres nécessaires sont passés à la procédure `motif`. Il n'est donc pas nécessaire (et il est donc **interdit**...) d'utiliser des tableaux ou autres artifices pour passer plus d'informations...

Remarque

Une solution peut s'écrire en moins de 10 lignes...

3 Les tours de Hanoi

La règle du jeu Le jeu des tours de Hanoi consiste en une plaquette de bois sur laquelle sont plantées trois tiges. Sur ces trois tiges sont enfilés des disques de bois de diamètre tous différents. La seule règle du jeu est de ne jamais poser un disque sur un disque plus petit que lui. Au début du jeu tous les disques sont posés sur la tige de gauche. Le but du jeu est de déplacer tous les disques d'une tige sur l'autre, sans jamais violer la règle, pour finalement les amener tous sur la tige de droite.

L'algorithme Supposons que les tiges s'appellent `A`, `B` et `C`, que n soit le nombre de disques, tous posés sur la tige `A` au départ, et qu'il faut les déplacer sur la tige `C`. La solution simple de ce jeu réside dans la remarque suivante: si on sait déplacer $n-1$ disques, il suffit de déplacer les $n-1$ plus petits de `A` vers `B`, puis de déplacer le disque restant de `A` vers `C`, et enfin de déplacer les $n-1$ disques de `B` vers `C`.

Le programme Il vous est demandé d'écrire une procédure récursive qui résolve le problème et qui affiche les trois tiges et les disques après chaque mouvement.

Commencez par écrire une procédure récursive qui se contente de résoudre le problème et d'afficher les mouvements avec la procédure ci-dessous:

```
#let mouvement de vers =
#    print_string
#       ("Déplace un disque de la tige " ^ de ^ " à la tige " ^ vers);
#    print_newline();;
```

Une fois cette procédure écrite on souhaite remplacer l'appel de la procédure `mouvement` par celui d'une fonction qui dessinerait l'état du jeu.

L'affichage nécessite de garder en mémoire le nombre et l'identité des disques présents sur chaque tige. Pour garder en mémoire les données nécessaires, vous pouvez utiliser un tableau linéaire `nombre_de_disques` dont la $i^{ème}$ case conserve le nombre de disques présent sur la $i^{ème}$ tige, et un tableau à deux dimensions dont la case (i, j) conserve l'identité du $j^{ème}$ disque présent sur la $i^{ème}$ tige.

Écrire une nouvelle procédure mouvement qui répercute en mémoire le mouvement, et qui affiche la nouvelle configuration.

4 Les dates et le travail demandé

Il vous est demandé de rédiger les deux programmes, de m'en donner les listings et d'être en mesure d'en faire une démonstration pour le prochain TD (celui du 6 octobre). Lors de ce TD je vérifierai que les programmes tournent... Ce projet peut être réalisé seul ou en binôme[1]

Toute personne qui ne pourrait assister au prochain TD voudra bien me remettre les listings et me contacter avant vendredi soir...

Vous voudrez bien excuser le manque flagrant de confiance que dénote ces quelques lignes. J'espère qu'aucun d'entre vous n'essayera de me jouer des tours, mais l'expérience des années précédentes incite à la prudence.

Ce projet sera noté car je serai obligé de vous donner une note de contrôle continu à la fin du module. Pour avoir une (très) bonne note, il suffit d'écrire deux beaux programmes qui utilisent proprement la récursivité.

Si vous avez le moindre problème, mes coordonnées se trouvent en en-tête du sujet...

1. Un unique trinôme sera autorisé si le nombre des élèves est impair et si aucun élève ne travaille seul!