Utilisation du module Graphics

laurent.jospin.59@free.fr, http://jospin.lstl.fr

Lycée Saint-Louis, Paris

1 Le module graphics

1.1 Installation et chargement

Le module graphics est installé avec les versions de OCaml jusqu'à 4.08.1. Dans les versions ultérieures il doit être installé séparémment. Si OCaml a été installé à l'aide de `opam`, on installe le module en tapant dans le terminal : `opam install graphics`.

Suivant les versions de ocaml installé et l'environnement, l'utilisation du module graphics peut se faire simplement avec

```
1 #load "graphics.cma";;
```

ou

```
1 #use "topfind";;
2 #require "graphics";;
```

Les fonctions du module `Graphics` sont ensuite utilisables à l'aide de `Graphics.open_graph "800x600"` par exemple. On peut également importer toutes les fonctions du module à l'aide de `open Graphics;;`.

La fonction `open_graph` prend en argument une chaîne de caractères décrivant les dimensions voulues pour la fenêtre. Elle peut être de la forme `"800x600+200+100"` pour que la fenêtre soit de taille 800×600 (pixels) et fabriquées initialement avec un décalage de 200 par rapport au bord gauche de l'écran et de 100 par rapport au bord haut. Sous environnement Linux, il faut commencer la chaîne de caractères par un espace avant les dimensions. Sous Mac, la fonction `open_graph` ignore les dimensions passées en argument. Pour une solution plus indépendante du système d'exploitation, on pourra faire `open_graph ""` puis `resize_window 800 600`.

1.2 Fonctions principales

La page `https://ocaml.github.io/graphics/graphics/Graphics/index.html` décrit succintement les fonctions du module. On trouve notamment :

- `rgb : int -> int -> int -> color` permet d'obtenir une couleur à partir de 3 entiers entre 0 et 255. Le type `color` est simplement un alias du type `int` et le résultat est simplement le nombre $\overline{rgb}^{256}$
- `set_color : color -> unit` permet de définir la couleur courante. Tous les tracés utiliseront cette couleur.
- `plot : int -> int -> unit` dessinne le pixel dont les coordonnées sont passées en argument.
- `moveto : int -> int -> unit` permet de déplacer la position courante.
- `lineto : int -> int -> unit` permet de tracer un segment de la position courante au point dont les coordonnées sont passées en argument.
- `clear_graph : unit -> unit` efface la fenêtre graphique.
- `draw_circle x y rayon` (et `fill_circle`), `fill_rect x y largeur hauteur` font bien ce qu'on imagine.
- `size_x` et `size_y : unit -> int` renvoient les dimensions de la fenêtre.

2 Utilisation

2.1 Découverte

1. En utilisant le module `Random` (chercher les fonctions utiles sur `https://ocaml.org/api/Random.html`), écrire une fonction qui remplit n rectangles aléatoires avec une couleur aléatoire dans la fenêtre graphique. Les rectangles auront leurs dimensions tirées aléatoirement entre 1 et la taille de la fenêtre divisée par deux.

2.2 Tracé de ligne pixel par pixel

Pour bien observer l'effet des fonctions suivantes, utilisant la fonction `plot` pour tracer des lignes, redéfinir la fonction `plot` pour qu'elle dessine des carrés de taille 10 :

```
1 let plot x y = fill_rect (10*x) (10*y) 10 10;;
```

2. Ecrire en utilisant uniquement la fonction plot, une fonction `trace_ligne x1 y1 x2 y2` dont le résultat est équivalent à `moveto x1 y1; lineto x2 y2`.

Ce type de fonctions n'est plus programmées que par quelques personnes qui développent des bibliothèques graphiques simples dans différents langages de programmation. On notera que suivant la pente de la droite tracée, il y a un unique point pour chaque abscisse, ou un unique point pour chaque ordonnée. Pour donner plus d'intéret à cette question, il est intéressant de se replacer dans un contexte où la moindre optimisation était bénéfique du fait de la lenteur des processeurs (mon premier ordinateur avait un processeur intel 80486dx20 cadencé à 20 Mhz... plus de 50 fois plus lent qu'un quelconque processeur de smartphone).

3. Reecrire cette fonction sans utiliser les flottants sans division ni multiplication. (Les divisions et multiplications étaient alors comptées comme nécessitant de l'ordre de 10 tics processeurs contre 1 pour les additions/soustractions).

2.3 Dégradé

4. Ecrire une fonction qui produit un dégradé horizontal sur un rectangle dont les dimensions sont fournies en argument, ainsi que les deux couleurs sous la forme de triplets d'entiers.

Après avoir programmé intuitivement une interpollation linéaire entre les deux couleurs, on pourra remplacer cette interpollation par la racine carrée de l'interpollation linéaire des carrés des composantes des couleurs, pour comprendre cette proposition, on pourra regarder une petite vidéo très intéressante, et très vulgarisée appelée *Why computer color is broken*. Cet aspect de la représentation des couleurs est très peu connue.

2.4 Interaction avec l'utilisateur

5. L'appel à `wait_next_event [Button_up]` attend jusqu'à ce qu'un bouton de la souris soit enfoncé puis relâché dans la fenêtre graphique et renvoie un enregistrement contenant entre autre les champs `mouse_x` et `mouse_y`. Ecrire une fonction qui trace le segment qui joint les positions entre les clicks successifs de la souris.