TP de programmation n° 6

Julien Reichert

Lundi 18 novembre 2013

Les graphismes en OCaml

Dans cette leçon, nous aborderons le module `Graphics` et, pour les bourrins, la librairie `SDL`. Illustrer un programme par des graphismes fait bonne impression sur le jury, sous réserve que ce ne soit pas au détriment de la compréhension du texte ni **surtout** du reste de la programmation.

1 Le module de base

Le module `Graphics` n'est pas dans la librairie standard. Il faut donc le charger avant de l'ouvrir pour l'utiliser, donc écrire dans un terminal les lignes de code suivantes :

```
#load "graphics.cma";;
open Graphics;;
```

Une fois le module ouvert, vous aurez accès à ses fonctions sans devoir les préfixer de `Graphics.`, ce qui est recommandé vu leur nombre.

Pour commencer, il faut ouvrir une fenêtre graphique, ce qui se fait par la fonction `open_graph`, qui prend en argument une chaîne de caractères. Cet argument peut être `""` afin d'ouvrir une fenêtre par défaut, ou `" 800x600"` pour ouvrir une fenêtre de taille 800 par 600 (attention à bien mettre une espace avant, ce qui n'est d'ailleurs pas le cas en Caml light). L'espace est en fait un délimiteur, car on peut choisir l'écran où ouvrir la fenêtre graphique en le précisant avant l'espace [1].

La fenêtre ne peut être ouverte qu'une fois, donc tenter d'en ouvrir une autre

1. je ne l'ai jamais fait...

écrasera la précédente. Pour la réinitialiser, on utilise `clear_graph ()`, pour la redimensionner `resize_window largeur hauteur`, pour changer son titre `set_window_title titre` et pour la fermer `close_graph ()`.

Le type couleur est un entier, qui est égal à 256^2 `R + 256 G + B` selon le code RGB. Il existe d'ailleurs une fonction `rgb` qui à trois entiers associe la couleur correspondante. Les couleurs principales sont prédéfinies en OCaml. La fonction `set_color` permet de changer la couleur du pinceau. Ensuite, les fonctions de dessin prennent le relais [2] :

- `plot x y` imprime le point `(x,y)` ;
- `moveto x y` déplace le pointeur à la position `(x,y)`, `rmoveto x y` déplace le pointeur selon le vecteur `(x,y)` ;
- `lineto x y` trace une ligne depuis le pointeur, qu'il déplace, jusqu'à `(x,y)`, `rlineto x y` trace une ligne depuis le pointeur, qu'il déplace, selon `(x,y)` ;
- `draw_rect x_coin_bg y_coin_bg lrg htr` dessine un rectangle sans modifier la position du pointeur ;
- `draw_poly_line` dessine les lignes joignant deux points consécutifs du tableau en argument, `draw_poly` fait de même et ferme le polynôme formé ;
- `draw_segments` dessine les segments définis par le tableau de quadruplets d'entiers en argument ;
- `draw_arc x_centre y_centre rayon_x rayon_y angle_dp angle_ar` dessine un arc, `draw_ellipse x_centre y_centre rayon_x rayon_y` (cas particulier) dessine une ellipse, `draw_circle x_centre y_centre rayon` (cas particulier) dessine un cercle ;
- Pour les suffixes `rect`, `poly`, `ellipse`, `circle` et `arc`, remplacer `draw` par `fill` remplit la zone au lieu de dessiner une figure ; dans le dernier cas, il s'agit de la zone définie par arc et corde.

Il est également possible d'imprimer du texte dans une fenêtre graphique, ce qui se fait par les fonctions `draw_char` et `draw_string`. Le caractère ou la chaîne a son coin en bas à gauche au niveau du pointeur, qui est déplacé au coin en bas à droite [3]. La taille des caractères, qui dépend aussi de l'implémentation, se modifie par `set_text_size`, la fonte (idem) par `set_font`. Afin de savoir la taille du rectangle dans lequel, avec les paramètres actuels, le texte `str` serait affiché, on dispose de `text_size str`.

Certaines informations peuvent aussi se récupérer : `point_color x y` retourne la couleur du point en `(x,y)`, en déclenchant une erreur `BadMatch` s'il est hors du champ, `current_x ()`, `current_y ()` et `current_point ()` retournent la position courante, `mouse_pos ()` retourne les coordonnées de la souris, `button_down ()` retourne un booléen indiquant si un bouton est enfoncé, et `read_key ()` attend qu'une touche soit enfoncée **alors que la fenêtre graphique a le focus** et retourne le caractère correspondant. Les flèches sont malheureusement ignorées (c'est un gros défaut du module).

2. pour toutes ces fonctions, en cas de débordement, aucune erreur n'est déclenchée
3. donc pensez à ajouter des espaces si vous utilisez plusieurs fois les fonctions

Les événements ajoutent de l'interactivité, plus ou moins en temps réel. Il y en a cinq : `Button_down`, `Button_up`, `Key_pressed`, `Mouse_motion` et `Poll`, ce dernier se produisant à tout moment. Le statut est un type produit contenant les informations suivantes : `type status = { mouse_x : int; mouse_y : int; button : bool; keypressed : bool; key : char; }`. Ceci est utilisé par la fonction `wait_next_event`, qui attend qu'au moins un événement de la liste donnée en argument se produise, et retourne le statut à ce moment-là. Si la souris est hors de la fenêtre graphique, les valeurs `mouse_x` et `mouse_y` peuvent ne pas être exploitables par d'autres fonctions sans déclencher d'erreur.

2 La librairie SDL

Le manuel, qui n'est pas sur le site officiel du langage, se trouve ici :

http ://ocamlsdl.sourceforge.net/doc/html/index.html

Il existe aussi un tutoriel qui donne quatre exemples de base, et surtout qui indique comment installer la librairie [4] :

http ://vog.github.io/ocamlsdl-tutorial/

Dans l'idée, OCamlSDL permet de faire des graphismes avancés, mais c'est très bas niveau... rien que pour dessiner une ligne, on doit imprimer tous ses points en utilisant une boucle. Il paraît que d'autres médias (son, texte, images) sont plus faciles à manipuler, il existe même un module pour gérer des CD-ROM et des manettes, mais pour l'utilisation que vous aurez d'illustrations dans vos programmes (si tant est qu'il y en ait besoin), il est peu probable que vous ayez à vous servir de cette librairie.

3 Exercices

Exercice 1 : S'il te plaît, dessine-moi un mouton !

Exercice 2 : Écrire une fonction qui dessine un carré dans une fenêtre graphique et sur lequel certaines touches du clavier auront des effets divers, notamment le déplacer, le rétrécir, l'agrandir, le tourner, le remplir, le vider.

Exercice 3 : Écrire une fonction qui affiche dans une fenêtre graphique le texte que l'utilisateur saisit. Trouvez notamment comment effacer un caractère.

4. qui est du coup peut-être absente de la clé agrèg, à vérifier