Idées de mini-projet

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

Lycée Saint-Louis, Paris

Il est important de bien structurer le programme, en particulier, il est important de séparer les fonctions qui modifient l'état du jeu des fonctions qui représentent l'état du jeu. On pourra utiliser `draw_string` et `set_text_size` pour l'affichage du texte dans la fenêtre graphique. Rechercher sur internet pour les mécanismes des jeux moins connus.

On pourra également s'intéresser à d'autres fonctions du module caml graphics présentés sur la page `https://caml.inria.fr/pub/docs/manual-ocaml/libref/Graphics.html`. En particulier, on notera la possibilité de jouer un son et de gérer des images.

1 Jeux avec attente d'une action

Exercice 1 (Jeu de la vie - (trop ?) facile)**.** Programmer le jeu de la vie (cf internet) et une évolution graphique.

Exercice 2 (Morpion - (trop ?) facile)**.** Programmer le jeu du morpion (et une interface graphique). Possibilité de faire le tic tac toe, le puissance 4, une intelligence artificielle etc.

Exercice 3 (Démineur - (trop ?) facile)**.** Programmer le jeu du démineur (et une interface graphique) (apparu dans les années 1960). De nombreuses variantes existent, en particulier une variante avec des cases hexagonales.

Exercice 4 (2048 - plutôt facile)**.** Programmer le jeu 2048 (apparu en 2014)

Exercice 5 (Jeux de dames, Jeu d'échec, Jeu de go, Awalé etc.)**.**

2 Jeux en temps réel

A noter qu'il n'y a pas de fonction pour patienter un certain temps avec caml sur windows, il faut donc une boucle infinie qui alterne vérification du tampon d'événements et affichage. Pour que la consultation du tampon d'événements ne soit pas bloquante, il faut tester si une touche à été appuyée grâce à la fonction `key_pressed`, puis utiliser `read_key` dans ce cas.. Un squelette d'application pourrait etre celui-ci :

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

let commence_partie () = [];;

let evolution_etat_jeu etat deltat appui touche=
   if appui then begin print_char touche; print_newline(); flush stdout end; etat;;

let affichage etat = ();;

let jeuEnCours=ref true and toucheAppuyee=ref false and touche = ref ' '
    and etat = ref (commence_partie ()) and temps = ref (Sys.time()) in
   open_graph "800x600";
   while !jeuEnCours do
      toucheAppuyee := key_pressed();
      if !toucheAppuyee then touche := read_key();
      etat := evolution_etat_jeu !etat (Sys.time() -. !temps) !toucheAppuyee !touche;
      temps := Sys.time ();
      affichage etat
   done;;
```

On peut aussi utiliser `wait_next_event` à la place avec une structure similaire.

Exercice 6 (JezzBall – Difficile)**.** Programmer le jeu JezzBall (apparu en 1992).

Exercice 7 (Tetris – Difficile)**.** Programmer le jeu Tetris (apparu en 1984). Une variante plus orginale bien que pas nécessairement aussi plaisante à jouer : Hexis.

Exercice 8 (Breakout – Plus difficile)**.** Programmer un jeu de casse-brique avec une plateforme à déplacer. Breakout est l'ancêtre du genre (apparu en 1976).

Exercice 9 (Puzzle Bubble – Plus difficile)**.** Programmer le jeu Puzzle Bobble (apparu en 1994).

3 Algorithmes de rendu visuel

On appelle scène un ensemble de polygones en dimension 3 à représenter graphiquement. On appelle caméra l'ensemble des paramètres définissant le point d'observation d'une scène ainsi que sa direction.

Exercice 10 (Outils)**.**

1. Programmer des fonctions pour transformer des coordonnées du plan en coordonnées entières de la fenêtre graphique ouverte de Caml. On centrera le repère et on utilisera une variable pour définir l'échelle en pixels de l'unité.
2. Programmer des fonctions pour projeter des coordonnées de l'espace sur un plan avec une perspective cavalière dont les paramètres seront définis au début du programme. On tiendra évidemment compte de la caméra pour représenter la scène.

Exercice 11 (Z-buffering – Difficile et lent)**.**

1. Ecrire une fonction qui remplit un polygone avec une couleur donnée, tout en remplissant un tampon de profondeur (z-buffer).
2. Ecrire une fonction qui trace une liste de polygones sans se préoccuper de l'ordre mais en faisant attention de ne pas remplacer un pixel correspondant à un polygone plus proche par un autre plus lointain.
3. Appliquer avec une figure où quatres rectangles se superposent circulairement (1 masque une partie de 2, 2 masque une partie de 3, 3 masque une partie de 4, et 4 masque une partie de 1).

Exercice 12 (Algorithme du peintre avec arbre binaire de séparation d'espace – Plus difficile mais plus rapide)**.**

Exercice 13 (Lancer de rayons – Très difficile)**.** Programmer un algorithme de lancer de rayons pour la représentation graphique d'une scène.