2I008 Procs et Foncs - Projet Robozzle (Partie 1)

Equipe pédagogique 2I008

Année 2014 - 2015

L'objectif de ce projet est de réaliser un clone du jeu Robozzle, en OCaml. Il s'agit d'un jeu de puzzles, dont le but est de ramasser toutes les étoiles des niveaux, en donnant des instructions de déplacement à un robot. Vous pouvez découvrir et vous familiariser avec le jeu à l'adresse suivante : http://www.robozzle.com/js/index.aspx. C'est d'ailleurs également à cette adresse que vous pourrez choisir des niveaux, et récupérer l'ID leur correspondant.

1 Puzzle : structures de données et importation

Dans ce premier exercice, l'objectif est de mettre en place l'importation des puzzles pour les utiliser dans votre programme OCaml. Nous effectuerons l'affichage de ces puzzles dans l'exercice suivant. L'affichage graphique de l'état du jeu sera effectué à l'aide de la librarie SDL, et nous vous fournissons des fonctions intermédiaires d'affichage des différents éléments de jeu dans le module G. Vous pouvez récupérer dans le répertoire /Vrac/2i008/ le fichier projet-robozzle.tar.gz contenant :

- un Makefile,
- le module G (fichiers g.ml et g.mli),
- l'interface du module Vm (vm.mli),
- les sprites du jeu (sprites.png),
- la font utilisée pour les affichage de textes (font.ttf),
- un script python permettant de récupérer des puzzles (level_fetcher.py),
- un premier fichier de puzzle pour vos tests (puzzles/*.rzl).

Question 1.1. Créer un module Puzzle (puzzle.ml, puzzle.mli), dans lequel seront déclarés les types nécessaires à la description complète d'un puzzle robozzle :

- un type somme direction qui permet d'encoder la direction courante du robot (nord, sud, est ou ouest),
- un type somme color représentant la couleur d'une case (valeur Empty si pas de couleur),

— un type somme cell représentant une cellule : une cellule a une étoile à ramasser ou non, ainsi qu'une couleur,
— un type enregistrement map qui contient le nombre de lignes du puzzle, le nombre de colonnes, et la map représentées sous forme d'une liste de cellules,
— un type enregistrement t représentant le puzzle et comprenant les attributs suivants :
  — nom du puzzle,
  — colonne de départ,
  — ligne de départ,
  — direction du robot,
  — tailles des fonctions F1 à F5 du robot, sous forme d'une liste d'entier,
  — map du puzzle.

Question 1.2. Implémenter la fonction split : string -> string list dans le module Puzzle, qui sépare une chaine de caractères en une liste de sous-chaines, avec comme séparateur le caractère ','. Cette fonction ne doit pas être visible depuis l'extérieur du module Puzzle.

Question 1.3. Implémenter la fonction parse dans le module Puzzle, qui prend en paramètre le nom d'un fichier de puzzle, et retourne le puzzle correspondant, de type t. Vous utiliserez les fonctions de la librairie standard d'OCaml input_line pour lire une ligne dans un fichier et int_of_string pour convertir une chaine de caractères en l'entier qu'elle représente. Un exemple de fichier à parser est le fichier puzzles/p644.rzl, et le fichier puzzles/p644-com.rzl contient des commentaires expliquant chaque ligne du fichier.

2 Machine virtuelle Robozzle

Vous l'aurez peut-être constaté, il est possible (et même obligatoire pour certains niveaux) d'exploiter la pile d'appel de fonctions dans Robozzle (essayez de résoudre par exemple le puzzle 330 « Learning stack »). Nous choisissons dans notre projet d'optimiser certains appels récursifs : comme dans le cas d'OCaml, les appels récursifs terminaux peuvent ne pas empiler leur contexte sur la pile (dans le cas de Robozzle, ne pas empiler l'adresse de retour de la fonction, puisqu'il n'y a plus d'instruction à exécuter après l'appel). L'architecture générale du projet sera donc la suivante : le joueur saisira sa solution à l'aide de la partie éditeur (module Editor). Cette solution se présente sous la forme d'un programme (module Ast (Arbre de Syntaxe Abstraite)) qui sera compilé vers du bytecode exécutable par une machine virtuelle (module Vm). Commençons par implanter le module Vm pour disposer de l'affichage graphique, et du moteur d'exécution de programmes compilés Robozzle. Vous disposez dans le fichier vm.mli des types commentés de la machine virtuelle Robozzle.

Question 2.1. Implémenter les accesseurs get_pos, get_map et get_dir.

Question 2.2. Implémenter les fonctions utilitaires suivantes :
- get_cell : pos -> Puzzle.map -> Puzzle.cell qui retourne la cellule à la position donnée d'une map,
- get_color : Puzzle.cell -> Puzzle.color qui retourne la couleur d'une cellule donnée,
- is_empty : Puzzle.cell -> bool qui retourne vrai si la cellule donnée contient une étoile, et faux sinon.

Ces fonctions ne devront pas être visible depuis l'extérieur du module Vm.

Question 2.3. Implémenter les fonctions de jeu suivantes :
- is_solved : state -> bool qui retourne vrai si la puzzle est résolu (plus aucune étoile sur la map), et faux sinon,
- is_out_of_map : state -> bool qui retourne vrai si le robot est sorti de la map ou sur une case vide, et faux sinon,
- is_out_of_instr : state -> bool qui retourne vrai s'il n'y a plus d'instructions de bytecode a exécuter, et faux sinon.

Question 2.4. Implémenter la fonction init : Puzzle.t -> state qui initialise la machine virtuelle à partir des données d'un puzzle.

Question 2.5. Implémenter la fonction draw : int -> int -> int -> state -> int -> int -> unit qui affiche la map et le robot à l'écran. Vous utiliserez donc le module G, documenté dans g.mli. Les arguments de l'appel Vm.draw offx offy csize state nb_steps anim_frame sont les suivants :
- offx (resp. offy) correspond à la position en x (resp. en y) de l'affichage de la map,
- csize correspond à la taille d'une cellule, en pixels,
- nb_steps correspond au nombre de pas pour l'animation entre deux cases,
- anim_frame correspond au numéro de sprite pour l'animation.

Dans un premier temps, vous pouvez ne pas utiliser les paramètres pour l'animation du robot. Il est important d'appeler la fonction G.sync () à la fin de cette fonction draw pour valider l'affichage.

Question 2.6. Implémenter une fonction principale dans le module Main. Cette fonction doit initialiser le module G, parser un fichier de puzzle, initialiser une VM à partir de ce puzzle, puis l'afficher. Vous pouvez utiliser la fonction delay du module G pour admirer le résultat.

Vous devriez avoir maintenant un puzzle affiché, la suite consistera à implémenter l'évaluateur de bytecode de la machine virtuelle Robozzle, puis de compiler un programme Robozzle saisi par le joueur.