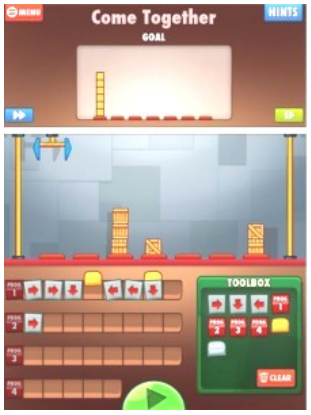

Introduction à l'algorithmique avec le jeu Cargo-Bot

Pascal Lafourcade Gilles Mounier Benjamin Wack

14 juin 2013

Afin d'initier les élèves à l'algorithmique, nous utilisons une version "réelle" du jeu *Cargo-Bot*, inventé par *Rui Viana*. Ce jeu consiste à programmer une grue qui manipule des containers de couleurs différentes.

Objectif du jeu

Programmer une grue pour placer des containers colorés dans une configuration d'arrivée à partir d'une configuration de départ donnée.

Règles du jeu

Le jeu comporte une grue et des containers de 4 couleurs : Rouge, Jaune, Vert et Bleu. La grue ne peut effectuer qu'un ensemble d'actions élémentaires appelées *commande* ou *instruction*. Un *algorithme* ou un *programme* est une séquence finie d'instructions. La grue dispose de 4 programmes contenant chacun au maximum 8 instructions. La grue peut effectuer uniquement les 7 instructions suivantes :

- 7 **instructions** :
  - 3 *déplacements* :
    - ⋆ Déplacer la grue d'une position vers la droite ou vers la gauche.
    - ⋆ Baisser la pince de la grue :
      - · Si la pince est vide et qu'il y a un container en dessous elle le saisit et remonte. S'il n'y a pas de container elle descend et remonte à vide.
      - · Si la pince contient un container, elle le dépose et remonte à vide.
  - 4 *appels de programmes* : les 4 instructions PROG 1, PROG 2, PROG 3 et PROG 4, lancent l'exécution des programmes associés.

  Chacune de ces instructions peut être étiquetée comme suit :

- 6 **étiquettes** : Elles sont placées au-dessus d'une instruction. En fonction de la couleur du container dans la pince elles permettent d'effectuer ou pas cette instruction. Les étiquettes représentant les 4 couleurs (, , ,) permettent d'effectuer l'instruction uniquement si la pince contient un container de la même couleur que l'étiquette. Il y a aussi une étiquette représentant *toutes les couleurs* qui effectue une instruction seulement si la pince contient un container (indépendemment de sa couleur) et une autre étiquette *sans couleur* qui n'exécute l'instruction que si la pince est vide.

Remarques : L'instruction a un effet très différent selon le moment où elle s'exécute, ce qui met l'accent sur la notion d'exécution d'un programme dans le temps. La grue effectue les actions programmées dans l'ordre, en commençant au programme PROG 1. Il n'y a pas de passage automatique au programme suivant : une fois un programme terminé, on revient au programme qui l'a appelé s'il y en a un, ou sinon on s'arrête. Enfin, le problème est considéré comme résolu dès que la configuration finale est atteinte, et ce même si le programme comporte encore des instructions à exécuter.

Dans Cargo-Bot, le langage dans lequel on programme est un peu particulier à deux titres : les instructions sont des actions très élémentaires, et la taille des programmes est limitée (huit instructions au maximum). Pour résoudre un problème, il faut donc expliciter chaque pas de la procédure à suivre, tout en recherchant une certaine économie de moyens.

Matériel fourni

Les containers sont représentés par des cubes de couleur. Chaque problème est imprimé sur une feuille A3, qui constitue à la fois l'énoncé, l'emplacement sur lequel on manipule les cubes, et l'environnement de programmation. Les problèmes sélectionnés abordent les notions clefs de la programmation de manière progressive et pédagogique : séquence d'instructions, appel de programme, instructions conditionnelles et appels récursifs.

Les énoncés avec leurs solutions, une feuille de problème vierge pour créer vos propres énoncés et les fichiers permettant de construire le jeu par vous-mêmes sont disponibles en ligne[1].

Enfin nous proposons une version jouable en ligne dans n'importe quel navigateur internet moderne, dans laquelle il est possible d'incorporer vos propres productions[2].

Progression pédagogique

Séquence : La composition d'instructions la plus courante dans les algorithmes est la composition séquentielle. Si P_1 et P_2 sont des programmes, alors "P_1 suivi de P_2" est encore un programme. Se cachent derrière cette notion le fait qu'un algorithme doit décomposer la résolution d'un problème en étapes élémentaires, et que ces étapes doivent être exécutées dans un ordre précis. Les problèmes qui demandent de translater une pile des containers, illustrent l'exécution séquentielle d'instructions comme par exemple *Jeu (démo).xlsx* et *Invert.xlsx*

Appel de programme : Rapidement, on atteint la taille limite des 8 instructions pouvant être écrites dans un programme. Cela nous oblige à appeler d'autres programmes pour avoir la place d'écrire toutes les instructions nécessaires. Cependant, il apparaît vite que ce mécanisme est plus une force qu'une faiblesse : si une même suite d'instructions doit être exécutée plusieurs fois, on l'écrit dans un programme que l'on appelle plusieurs fois. L'ensembles des problèmes proposés possèdent des solutions utilisant un ou plusieurs appels de programme. Exemple de problèmes : *Split*, *From Beneath*.

Instructions conditionnelles : Les étiquettes permettent de paramétrer l'action de la grue en fonction de son contenu, et donc d'explorer des branches différentes du programme au fur et à mesure de son exécution. Une utilisation simple consiste à déplacer la grue dans des directions différentes selon la couleur des containers, mais il est aussi possible de soumettre l'appel à un programme à une condition. Exemple de problèmes : *From Beneath*.

Appels récursifs : Plutôt que de répéter huit fois explicitement l'appel 

, il est à la fois plus économique et plus général d'écrire un programme signifiant "Appeler 

et recommencer". C'est possible grâce à une construction qu'un débutant s'interdit souvent bien qu'elle soit parfaitement autorisée dans les règles du jeu : le programme 

a le droit de s'appeler lui-même. Exemple de problèmes : *Re-Curses*.

On peut ici également faire valoir la concision des programmes que permettent d'obtenir ces appels récursifs, comme le montre le problème *Jeu (démo)* qui admet une solution récursive plus courte que la solution "naïve".

Enfin, combiné avec les étiquettes conditionnelles, l'appel récursif permet d'obtenir des comportements similaires à celui d'une boucle *tant que*. De tels programmes restent assez complexes à écrire.

On notera enfin qu'il n'y a pas de notion de variable. La position des containers à chaque exécution d'une instruction peut être vue comme l'état de la "mémoire".

[1] `http://www-verimag.imag.fr/~plafourc/Cargo-bot/`
[2] `http://www-verimag.imag.fr/~wack/CargoBot/`