Concepts de programmation solutions

Tutoriel partie 1 :

Plusieurs solutions possibles, la plus simple et courte et de simplement avancer vers la première étoile, se tourner *deux fois* pour face à l'autre étoile et avancer pour récupérer cette dernière.

F1 : ↑ ↑ ← ← ↑ ↑ ↑ ↑

Tutoriel partie 2 :

C'est le même puzzle que le précédent mais on n'a pas assez de cases dans F1 pour le résoudre de la même façon. Comme expliqué dans l'énoncé on peut mettre deux fois « Avancer tout droit » dans F2 pour avoir une commande F2 qui exécutera 2 actions ! On peut donc reprendre la solution d'avant et remplacer chaque paire 'Avancer tout droit' par F2.

F1 : F2 ← ← F2 F2

F2 : ↑ ↑

Lien avec la programmation : Que sont F1 et F2 ? Ce sont des suites d'instructions regroupées sous un nom. Ces instructions peuvent être réutilisées chaque fois qu'on le veut en appelant le nom de leur groupe. *Dans la programmation, l'équivalent s'appelle une fonction*.

Tutoriel partie 3 :

On a seulement deux emplacements ici mais on doit avancer de 8 cases, comment faire vu que 2 'Avancer tout droit' est une impasse ? On a seulement F1 comme fonction disponible MAIS on peut l'utiliser comme commande à l'intérieur d'elle-même ! Que se passe-t-il alors ? F1 se lance et joue les instructions de mouvement qu'on a placées et dès qu'elle tombe sur la commande F1 elle rejoue les instructions données *dont de nouveau l'appel à F1*. La fonction F1 va donc faire des boucles ! Vu qu'on doit répéter 'Avancer tout droit' voici ce que contient F1.

F1 : ↑ F1

Lien avec la programmation : Avoir une manière simple de répéter des informations autant de fois que nécessaire c'est pratique et parfois indispensable, on appelle ces répétitions des *boucles* en programmation. La boucle que vous avez utilisée dans cet exercice est un type appelée *boucle récursive* et c'est la boucle la plus dur à appréhender, félicitations !

Définition du Larousse en ligne de l'adjectif *récursif, récursive* : « Se dit d'un programme informatique organisé de manière telle qu'il puisse se rappeler lui-même, c'est-à-dire demander sa propre exécution au cours de son déroulement. »

Tutoriel partie 4 :

Si on doit résumer la situation de façon concise en instructions on pourrait qu'il faut :

- Avancer sur toutes les cases (Même la bleue ! Car tourner ne fait qu'une rotation de notre robot et il va falloir continuer à avancer après avoir tourné)
- Tourner à droite sur la case bleue
- Boucler parce que 3 instructions ne nous permettraient pas sinon de finir ce puzzle.

C'est donc à pic que tombe le carré bleu, en le combinant avec 'Tourner à droite' on forcer le robot à seulement tourner à droite sur les cases bleues.

F1 : ↑ →(Bleue) F1

Lien avec la programmation : Exécuter des actions seulement si des prérequis sont remplis multiplie les possibilités et permet à nos programmes de pas être linéaire et pouvoir répondre à différents cas. C'est ce qu'on appelle dans le milieu *des conditions*. « SI vrai ALORS faireCeci {SINON faireCela} (optionnel, seulement dans le cas où on veut effectuer une action si la condition n'est pas remplie) ».

Niveau : Stairs (« Escaliers ») :

On n'a définitivement pas assez d'instructions pour résoudre ce problème seulement avec des instructions de mouvement, ce qui signifie qu'on va devoir faire appel à une boucle.

Mais avant quoi que ce soit, quelles seraient les instructions en français pour résoudre ce problème ?

- Avancer
- Tourner à gauche puis avancer
- Tourner à droite
- Répéter les instructions précédentes

En version Robozzle ci-dessous.

F1 : ↑ ← ↑ → F1

Niveau : Smart reuse (« Réutilisation intelligente ») :

STOP ! Avez-vous lu le titre ? Si la réponse est non c'est le moment. « Réutilisation intelligente » … vous vous souvenez de comment on appelle ce qui nous permet de regrouper

plusieurs instructions sous un nom ? Dans le cas où vous avez dit « une fonction », bingo ! Si on devait réaliser ce puzzle seulement avec des instructions de mouvements ça donnerait :

↑ ↑ ← ↑ ↑ → ↑ ↑ ← ↑ ↑ ← ↑ ↑ → ↑ ↑ → ↑ ↑ ← ↑ ↑ → ↑ ↑ → ↑ ↑

(Ceci n'est pas un code konami)

Ça fait beaucoup d'instructions, et de travail par la même occasion. On pourrait regrouper une suite d'actions dans une fonction. Voici les instructions groupées qu'on pourrait ensuite réutiliser :

(↑ ↑ ←) (↑ ↑ →) (↑ ↑ ←) (↑ ↑ ←) (↑ ↑ →) (↑ ↑ →) (↑ ↑ ←) (↑ ↑ →) (↑ ↑ →) (↑ ↑)

Et donc nos fonctions Robozzle suivraient ce qui suit.

F1: F2 F3 F2 F2 F3 F3 F2 F3 F3 F2/F3

F2: ↑ ↑ ←

F3: ↑ ↑ →

A noter que la dernière instruction peut être soit F2 ou F3 car après avoir avancé de deux cases le puzzle est fini dans tous les cas.

<u>Niveau : Sh :</u>

Attention à la précipitation ! On n'a pas assez d'instructions pour se retourner et avancer assez pour ramasser toutes les étoiles. Une boucle ne nous avancerait pas non plus parce que si on fait un tour à 180° puis on avance, au prochain tour de boucle on se retournera à nouveau avant d'avancer ce qui nous remet au point de départ. Et pas de conditions possibles dans ce sens-là car toutes les cases sont bleues.

Quelle est l'autre option ? On peut remarquer qu'il n'y qu'une seule case rouge à l'extrémité gauche. C'est donc une case *différente* de toutes les autres : parfait pour une condition !

On peut donc toujours avancer tout droit et dès qu'on tombe sur une case rouge tourner 2 fois (pour faire un tour à 180°) et boucler le tout. Ce qui nous donne.

F1 : ↑ ←(Rouge) ←(Rouge) F1

<u>Niveau : Stoplight (« Feu rouge ») :</u>

Pour se simplifier la vie faisons la liste des mouvements nécessaires pour arriver au bout de ce puzzle et regroupons-les si il est possible de faire une série.

(↑ ↑ ←) (↑ ↑ ←) ↑ ↑ ↑ ↑ (↑ ↑ ←) (↑ ↑ ←) ↑ ↑

On remarque qu'on ne tourne pas quand on tombe sur case verte.

Comme on n'a qu'une seule fonction il n'est pas possible de les grouper et de les appeler quand nécessaire, l'option restante pour répéter des instructions et qui, de plus, se fera autant de fois que nécessaire sans qu'on doive le calculer est une boucle infinie. On doit donc avancer deux fois puis tourner à gauche seulement si on est sur une case bleue.

F1: ↑ ↑ ←(Bleue) F1

Une solution alternative est de relancer la boucle avant de tourner mais seulement quand on est sur une case verte, cependant cette solution utilise une case de plus.

F1: ↑ ↑ F1(Verte) ← F1

Lien avec la programmation : Cette deuxième méthode nous apprend une chose importante sur la récursivité, car lorsqu'on lance une série d'instructions en plein milieu de la fonction alors que toutes les précédentes instructions n'ont pas encore été effectuées on peut remarquer que les nouvelles instructions se glissent « au milieu » et seront lancées avant les autres. Le premier appel à la fonction n'est en fait pas fini au moment où on fait appel une seconde fois à la fonction.

<u>Niveau : Mountains (« Montagnes ») :</u>

La première étape importante est d'analyser ce qu'on reçoit. Si on observe bien on voit qu'on va devoir prendre des tournants mais dans un cas il faudra tourner à droite et dans l'autre tourner à gauche. Ne pas hésiter à se mettre à la place du robot avec ses mouvements limités et d'avancer étape par étape, d'ailleurs il y a un bouton 'Step' juste à droite de 'Go !' qui vous permettra de voir chaque instruction exécutée une par une.

Maintenant existe-t-il un lien entre les tournants à droite et ceux à gauche ? Oui ! Les tournants à droite sont toujours précédés de cases bleues et ceux à gauche de cases vertes. Cependant on ne peut pas voir de quelle couleur était la case avant la rouge une fois qu'on l'a passée car le programme ne permet pas de garder en mémoire des informations ! Par contre on peut utiliser une approche différente c'est avoir un comportement pour les cases bleues et de changer de « mode » dès qu'on arrive sur une case verte. Et chaque mode serait une des deux fonctions, F1 ou F2.

Si on pose sur papier (ou sur écran, on n'est pas difficile après tout) le cheminement de notre robot, il sera équivalent à ci-dessous.

Mode bleu (F1) :

- SI je suis sur une case verte
    - Changer en mode vert (F2)
- SINON
    - Avancer
    - Tourner à droite sur les cases rouges
    - Boucler en se rappelant

Mode vert (F2) :

- SI je suis sur une case bleue
    - Changer en mode bleu (F1)
- SINON
    - Avancer
    - Tourner à gauche sur les cases vertes
    - Boucler en se rappelant

Ce qui nous donne en version Robozzle.

F1 : F2(Verte) ↑ →(Rouge) F1

F2 : F1(Bleue) ↑ ←(Rouge) F2

Lien avec la programmation : Dans ce cas c'est plutôt une absence de lien qui nous a compliqué la tâche. Si on avait pu, une fois sur une case rouge, savoir de quelle couleur est la case précédente on n'aurait pas eu besoin de créer ces modes. Lorsqu'on enregistre des informations (sous un nom, sinon il serait plutôt compliqué de les récupérer ces informations) dans un programme, cela s'appelle *une variable*.

Niveau : Hypnotize (« Hypnotiser ») :

Les choses deviennent plus compliquées ici, il ne faut pas oublier le but principal qui est de récupérer toutes les étoiles et il y en a 2. Chacune est sur une case verte, une au centre et l'autre derrière le point de départ.

Les « points d'intérêts » sont les cases rouges et vertes, les rouges « indiquent » qu'il faut tourner et les vertes qu'on est à un des bouts du parcours. En suivant le parcours dans un sens il

faut tourner à gauche sur les cases rouges et pour retourner sur ses pas il faut tourner à droite. Rien ne sert de retourner directement parce qu'on ne pourrait pas mettre de condition sur une seule case bleue et qu'il ne faut pas tourner sur toutes les cases bleues.

En résumé, il faut toujours avancer et dès qu'on tombe sur une case rouge on tourne à gauche. Ainsi on arrive au bout sur une case verte et à ce moment il faut se retourner et répéter la première étape sauf que là il faut tourner à droite sur les cases rouges.

F1: ↑ ←(Rouge) F3(Verte) F1

F2: →(Rouge) ↑ F2

F3: →(Verte) →(Verte) F2