ALEXANDRE Ewan, Groupe 1
02/09/2025

TP : Résolution de Labyrinthes à l'aide de bloc de codes

Sommaire :

1) Problématique

Nous cherchons à comprendre ce que le jeu nous apprend sur la façon dont le coding fonctionne.

2) Analyse

Pour trouver la réponse, il faut d'abord comprendre le fonctionnement de chaque bloc de code.

Au premier niveau les blocs de codes sont les suivants :

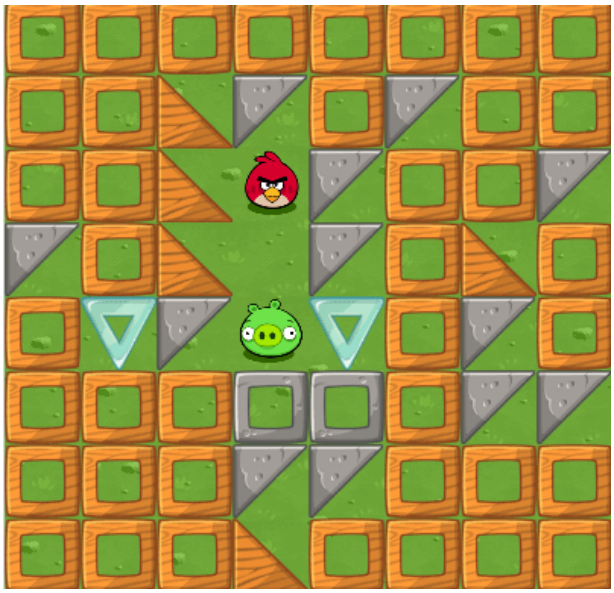
Ici move forward permet de faire avancer le personnage vers l'avant, dans le sens où il regarde.

Ici turn left permet de tourner son personnage à sa gauche

Ici turn right permet de tourner son personnage à sa droite

3) Réalisation et progression des niveaux

Le premier labyrinthe est simple, le personnage a simplement besoin d'avancer pour atteindre sa cible :



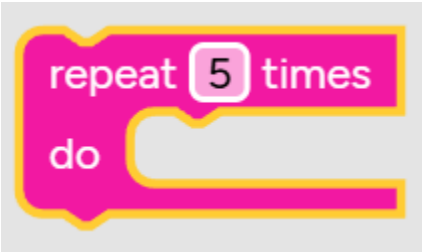
Ici on voit que le cochon vert se trouve deux cases plus bas que l'oiseau rouge, l'oiseau est orienté vers le cochon vert, donc ici il peut simplement avancer de deux cases en utilisant *move forward* deux fois, c'est en tout cas le seul moyen possible ici d'atteindre le cochon avec les blocs de codes donnés.

Configuration :

Le jeu dispose d'un système de nombre de blocs maximaux, cela permet à l'utilisateur de pouvoir voir la meilleure configuration à utiliser pour finir le niveau, cela se retranscrit dans la réalité comme créer le code le plus court pour exécuter une tâche donnée.

Les 5 premiers niveaux permettent de mettre en pratique l'alignement de plusieurs codes, soit en utilisant successivement le même code soit en alternant entre plusieurs codes différents.

Le 6ème niveau fait apparaître un nouveau bloc de code :



Ce code permet de répéter une action un x nombre de fois, cela permet de gagner du temps et de rendre le code plus simple dans son ensemble. Cela règle le problème de devoir à chaque fois répéter le code plusieurs fois à la suite.
Et cela montre que l'on peut simplifier son code entier avec simplement une action, comme avec de vrais codes.

L'avantage de cette action c'est donc de simplifier la répétition d'une action cependant c'est aussi son inconvénient, il faut prendre en compte la façon dont le code va finir, par exemple si on met *turn left* dans ce bloc il va falloir savoir quelle orientation le personnage va prendre à la fin du *repeat.*

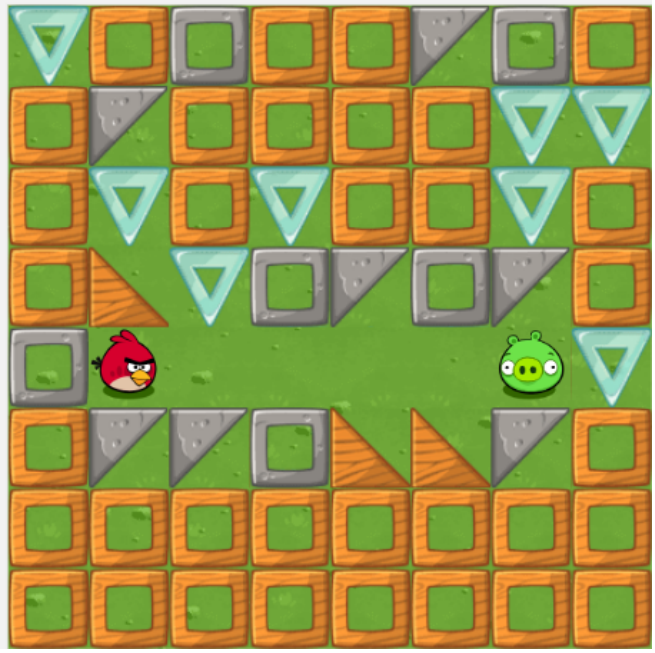
Le 6ème niveau jusqu'au 8ème sont simplement des mises en application de l'action *repeat* dans plusieurs configurations.

Le 9ème niveau introduit une nouvelle fonctionnalité : les blocs de codes immuables : cela permet d'avoir une base pour savoir par quoi commencer,

Néanmoins c'est aussi une contrainte car il va falloir adapter notre code autour de celui-ci, mais cela nous met simplement en situation lorsqu'il s'agit de devoir adapter son code par rapport à une action.

Le 10ème niveau lui va ajouter une action qui va nous permettre de répéter plusieurs actions jusqu'à que l'exigence soit respectée, cela va permettre l'automatisation du code, plus besoin d'indiquer de combien de boucles on a besoin. Tout se fera autant de fois qu'il le faut jusqu'à que l'exigence soit respectée, dans notre cas l'exigence c'est d'atteindre le cochon vert, donc chaque action se répétera autant de fois qu'il le faut jusqu'à qu'on l'atteigne.

Voici la configuration du 10ème niveau :



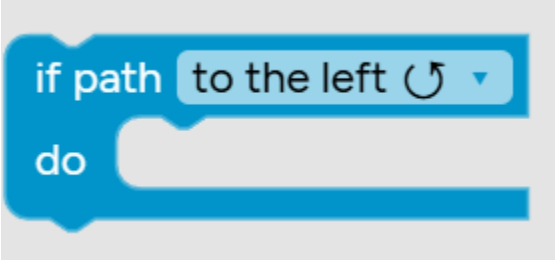
Ici on se retrouve dans le même cas que le premier niveau, il faut simplement faire avancer l'oiseau avec *move forward,* sauf que cette fois ci on va utiliser la configuration suivante :

Ce niveau 10 nous permet de comprendre à quel point l'automatisation d'une tâche peut être importante, plus besoin d'aligner plusieurs fois le même bloc de code, ici l'action *repeat until cochon vert* s'occupe de tout.

Les niveaux 10 à 13 permettent donc de mettre en application cette automatisation.

Le niveau 14 rajoute un nouveau bloc, celui-ci permet d'effectuer une action lorsqu'une exigence est respectée. Contrairement au code d'avant l'action SE FAIT lorsque l'exigence est respectée tandis que celui d'avant fait ARRÊTER le code lorsque l'exigence est respectée.

Ce code permet une meilleure automatisation, plus simple et plus efficace, ici pas besoin d'ajuster le code plusieurs fois à certains points dans l'avancement, cela se fera automatiquement lorsque l'exigence sera respectée.

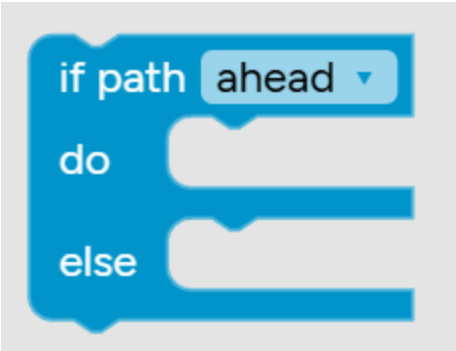


Ici par exemple l'exigence est que s'il y a un chemin sur la gauche du point de vue du personnage, alors celui-ci fera X action. Le plus logique est donc de faire tourner le bonhomme à gauche pour qu'il puisse continuer à suivre le chemin.

De plus ce code permet de s'adapter à plusieurs situations, par exemple on peut rajouter un code à celui d'avant dans lequel le personnage pourrait tourner à droite lorsqu'un chemin se trouve à sa droite. Comme ça le personnage pourra s'adapter à n'importe quel chemin, sauf si c'est un piège dans ce cas il faut choisir l'action automatisé qui ne fera pas échouer le but du code.

Encore une fois, les exercices 14 jusqu'à 17 sont simplement des mises en situation.

Finalement, les exercices 18 à 20 eux ajoutent encore un nouveau bloc de code qui permet d'exécuter une action si l'exigence n'est PAS respectée. Ces différents types de blocs de codes qui automatisent le code en général permettent de faire gagner beaucoup de possibilités de simplification du code et donc ils permettent une grande polyvalence.



Ce code-ci permet de rendre plus simple l'automatisation d'avant, cette fois-ci il ne se limite pas qu'à une action mais deux ce qui donne plus de possibilité de finir les niveaux en utilisant un chemin qui pourrait possiblement ne pas être pris si seul le code d'avant avait été utilisé.

Retour d'expérience :

En conclusion, ce TP nous permet de comprendre comment le coding fonctionne, on nous met dans différentes situations qui nous fait apprendre petit à petit toutes les possibilités d'actions que l'on peut utiliser pour réussir à exécuter un code, on nous fait comprendre comment on peut le simplifier, comment il faut réfléchir quand on utilise telle ou telle action, il faut visualiser le résultat engendré par les différentes actions. Donc ce TP nous apprend d'une certaine façon à coder grâce à l'expérimentation que l'on fait pour chaque niveau en utilisant plusieurs bloc de codes qui s'apparentent à de véritables codes utilisés dans certains langages de développement.