Published on *Le site de la Fondation La main à la pâte* (https://www.fondation-lamap.org)

1, 2, 3, codez ! - Activités cycle 3 - Etape 2.4: Récolter des ressources, gérer son score

Résumé	Les élèves complètent leur programme en ajoutant des ressources à aller chercher (nouveaux lutins) et en créant une variable pour leur score (ce score augmente lorsqu'on récolte des ressources). Ils apprennent à programmer des instructions conditionnelles (si… alors) et à utiliser des capteurs.
Notions	« Algorithmes » : • Une boucle permet de répéter plusieurs fois la même action un test permet de choisir quelle action effectuer si une condition est vérifiée ou non • une condition est une expression qui est soit vraie, soit fausse « Machines » : • Une variable est un nom que l'on donne à une zone de mémoire. Elle permet de stocker une valeur et de la réutiliser plus tard, ou de la modifier. « Langages » : • Certaines instructions s'exécutent simultanément à d'autres : on parle de programmation parallèle.
Matériel	Pour la classe • Un vidéo projecteur • Version agrandie de la Fiche 32 Pour chaque binôme • Un ordinateur connecté à Internet ou, en l'absence de connexion Internet de bonne qualité, un ordinateur sur lequel le logiciel *Scratch* a été préalablement installé . • Les nouveaux lutins qui seront utilisés à cette séance, téléchargeables ici. Pour chaque élève • Fiche 32
Lexique	Boucle, test, capteur, nombre aléatoire

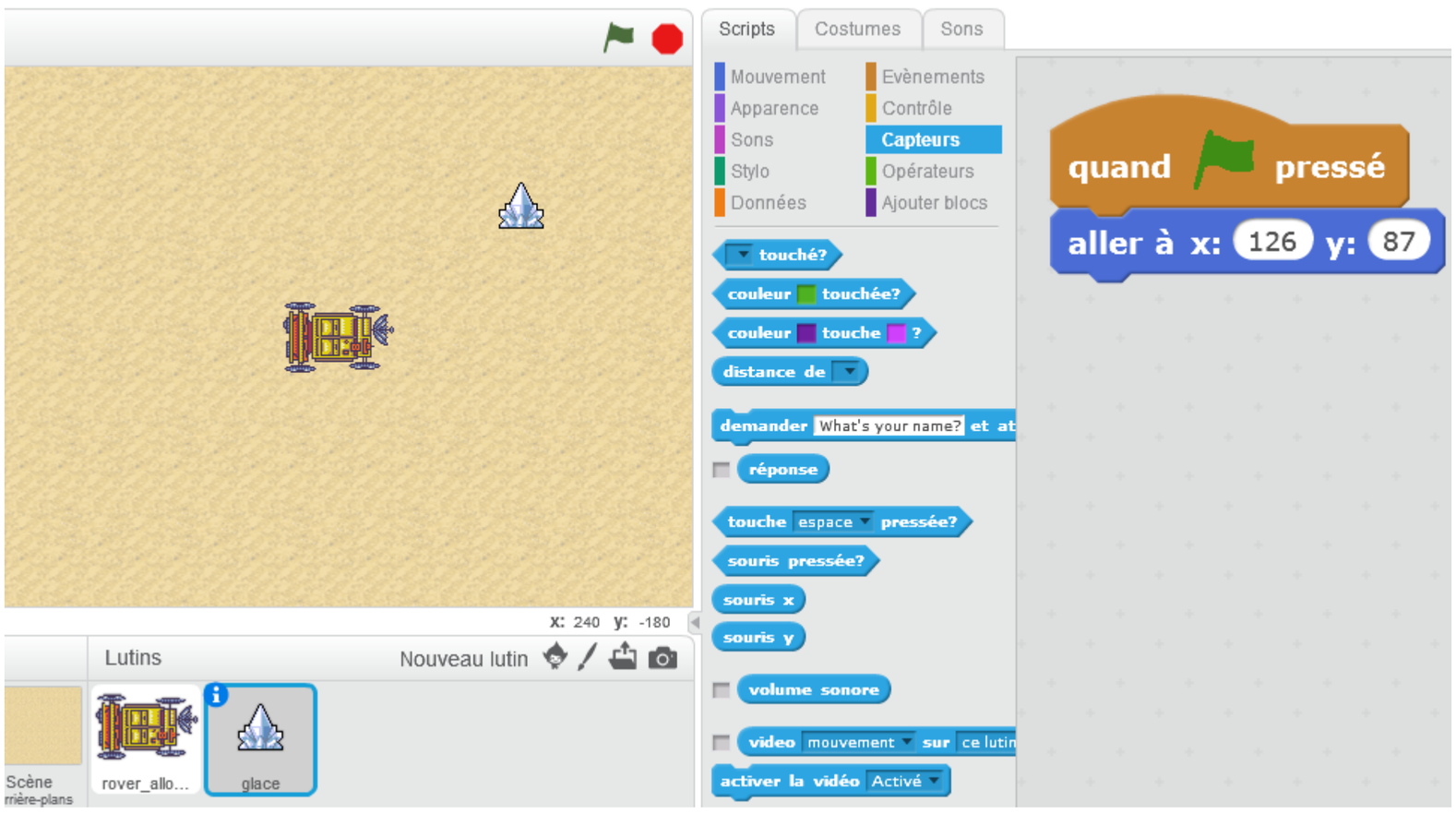
Notes pédagogiques :

- Il s'agit de l'étape centrale du projet car, pour gérer les ressources, les élèves vont devoir découvrir et mobiliser de nombreuses notions nouvelles : tests, variables, capteurs, opérateurs.
- Afin de ne pas aborder toutes les nouveautés d'un coup, nous proposons un découpage en plusieurs tâches élémentaires. Même dans ce cas, la tâche 2 est relativement complexe et nécessite un guidage de l'enseignant.
- À partir de maintenant, il est illusoire de chercher à ce que tous les élèves avancent au même rythme (sinon, les groupes ayant le plus de facilité vont très vite s'ennuyer et se dissiper). Le découpage en étapes et tâches permet de faciliter la gestion de classe par l'enseignant : chacun, où qu'il en soit, a quelque chose à faire.
- Nous conseillons de former des binômes relativement homogènes plutôt que des binômes associant un élève en difficulté et un élève plus avancé dans la programmation (car dans ce cas, l'expérience montre que l'élève en difficulté est passif et laisse faire l'autre).

L'enseignant explique que l'équipage doit s'approvisionner en diverses ressources pour survivre, notamment en eau (sous forme de glace, sur cette planète) et en nourriture (sous forme de végétaux). Le rover va devoir les récupérer, et un « score » permettra de compter le nombre de ressources récoltées.

Tâche 1 : importer une ressource (la glace) sous la forme d'un nouveau lutin (5 minutes)

Les élèves reprennent leur programme enregistré à la séance précédente et importent un nouveau lutin : la glace (image disponible dans le sous-répertoire « Lutins » du dossier mis à disposition, comme précédemment).
L'enseignant veille à ce qu'ils pensent à initialiser manuellement la position de cette ressource, comme ils l'avaient fait à la séance précédente pour le rover. Ils peuvent positionner la glace n'importe où sur l'écran, pourvu que les lutins (glace et rover) ne se chevauchent pas.
Cela donne, par exemple :



Note pédagogique :
On remarque ici que chaque lutin possède sa propre zone de programme (on passe du programme du rover à celui de la glace en cliquant sur le lutin voulu). Il y a potentiellement autant de programmes que de lutins : tous ces programmes s'exécutent en parallèle.

Tâche 2 : faire dire « bravo » à la ressource lorsqu'elle est touchée par le rover (20 minutes)

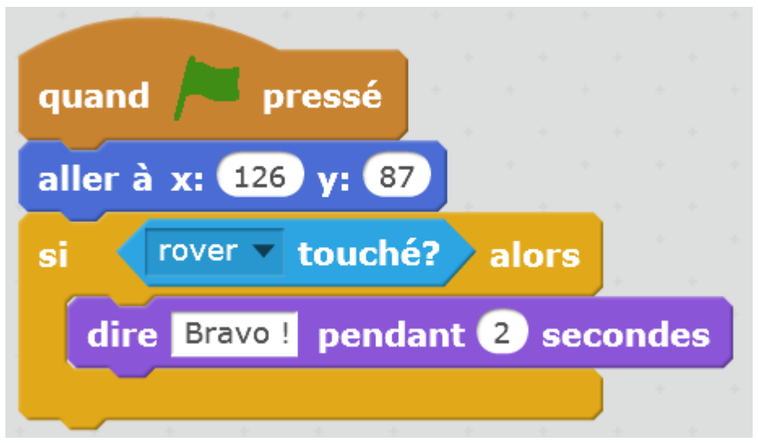
Les binômes doivent modifier le programme de la glace pour que celle-ci dise « bravo » lorsqu'elle est touchée par le rover. L'enseignant les laisse tâtonner, puis passe dans les groupes pour les guider en cas de blocage.
Cette tâche suppose :

- De savoir faire dire « bravo » au lutin (tous les élèves savent le faire à ce stade)

- De savoir déclencher une instruction uniquement lorsqu'une certaine condition est remplie. Cela se fait via la catégorie «**contrôle** », dans lequel on trouve l'instruction «**si… alors…** »

- De savoir détecter quand un lutin en touche un autre. Cela se fait via la catégorie «**capteurs** » de la palette d'instructions (instruction «**.. touché ?** »). Une fois qu'on l'a sélectionné, un clic sur la petite flèche permet de faire défiler les lutins déjà créés. Ici, on est dans le programme de la glace et on veut tester si ce lutin touche le rover, on clique donc sur « rover ».

Le programme du lutin « glace » devient alors :

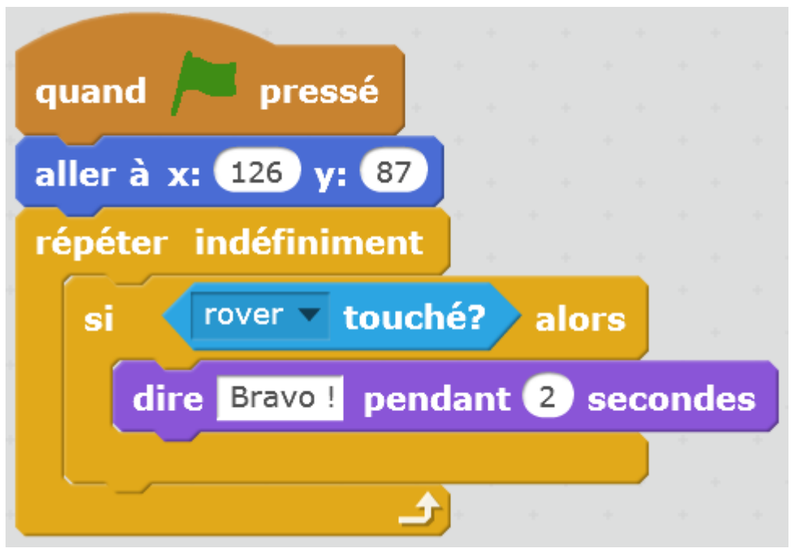


Malheureusement, lorsque l'on lance le programme et que l'on dirige le rover vers la glace, cela ne fonctionne pas. Pourquoi ? La classe peut discuter collectivement de ce que fait ce programme, pas à pas :

- La glace est placée dans la position que l'on a choisie
- Un test est effectué : si la glace touche le rover, alors elle dit « bravo »
- Puis… plus rien.

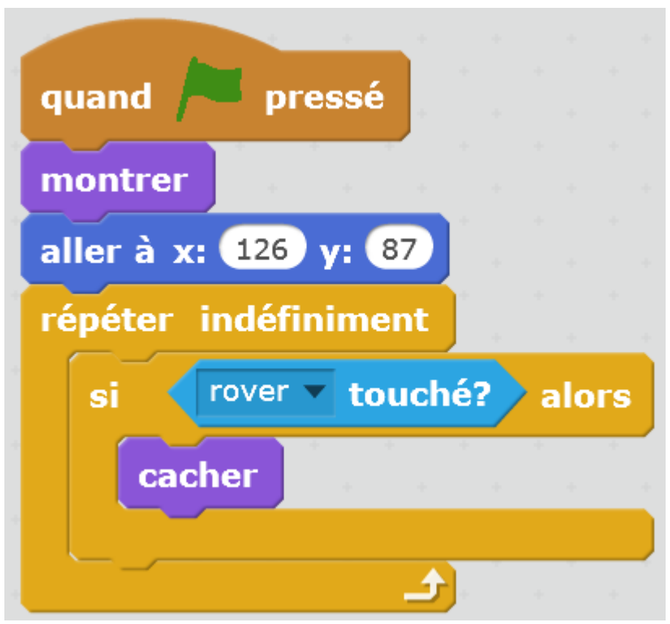
On remarque, en lisant ce programme, que le test n'est effectué qu'une seule fois, au lancement du programme (juste après qu'on a initialisé la position de la glace). Or, à ce moment-là, les 2 lutins ne se touchent pas. Donc, le message « bravo » ne s'affiche pas. C'est normal.
Pour que le programme fonctionne correctement, il faut que le test « si la glace touche le rover » soit effectué en permanence, de façon à pouvoir déclencher l'action voulue dès que la condition sera remplie.

Pour cela, il suffit de placer ce test dans une boucle «**répéter indéfiniment** », que l'on trouve dans la catégorie «**contrôle** ». Le programme de la glace devient :



Tâche 3 : faire disparaître la ressource quand elle est touchée (10 minutes)

Cette tâche très facile nécessite simplement de remplacer, dans le programme de la glace, l'instruction « dire Bravo » par une instruction faisant disparaître ce lutin. Il s'agit de l'instruction «**cacher** » que l'on trouve dans la catégorie «**apparence** ».
Attention, une fois qu'on exécute le programme, la ressource est désormais toujours cachée (car on ne lui a jamais dit de se montrer à nouveau !). Il faut donc ajouter l'instruction «**montrer** » juste après l'instruction « quand drapeau vert pressé ».
Le programme de la glace devient donc :



Tâche 4 : créer une variable « score » (5 minutes)

L'enseignant rappelle aux élèves qu'ils doivent créer un score et l'augmenter à chaque fois que la ressource est touchée. Les élèves peuvent explorer les différentes catégories d'instructions : la réponse se trouve dans la catégorie orange intitulée «**Données** », via la commande «**créer une variable** ».

Notes pédagogiques :

- La variable ainsi créée peut être accessible par un seul lutin (celui dans le programme duquel elle est créée) ou par tous. On parle (dans d'autres langages de programmation) respectivement de variable locale ou de variable globale.
- Pour qu'un programme soit facile à comprendre, il est important de donner des noms explicites aux variables que l'on crée. Cette bonne habitude limite également les bugs de programmation. Le nom de la variable peut donc être, tout simplement : « score ». Certains élèves utilisent des noms qui n'ont pas de sens, ou qui révèlent une confusion entre la variable et les manipulations de cette variable (par exemple, ils nomment la variable « ajouter 1 au score »).
- On remarque que lorsque la variable est créée, elle est affichée à l'écran, ainsi que sa valeur. Pour faire disparaître cet affichage, il suffit de décocher la case à gauche du nom de la variable, dans la palette « données ».

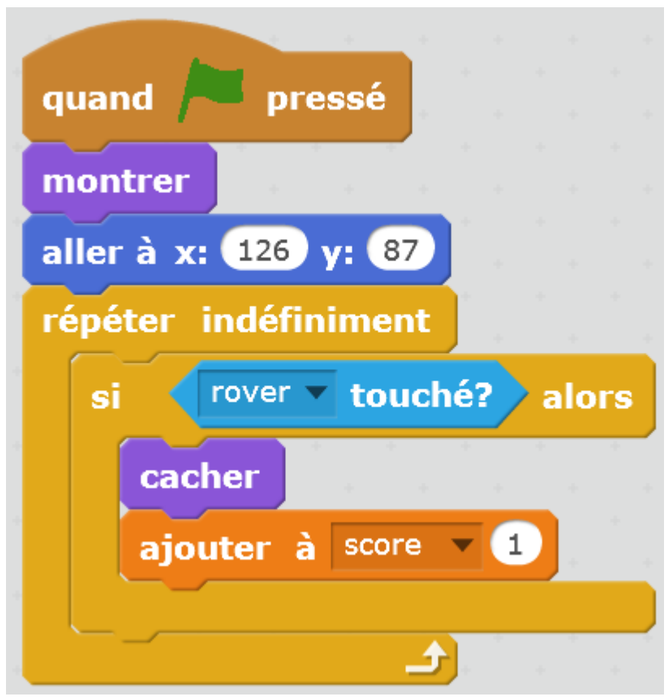
Ici, il s'agit de compter un score. Cette variable pourra sans doute être manipulée par plusieurs lutins (les différentes ressources…) : on doit donc la rendre accessible à tous les lutins.
L'enseignant fait remarquer aux élèves qu'ils ont déjà manipulé des variables dans les séances précédentes (l'abscisse X et l'ordonnée Y, qui donnent la position d'un lutin à l'écran). Ces variables étaient déjà disponibles et les élèves ont pu les manipuler (faire des tests, leur affecter des valeurs…) sans avoir besoin de les créer.

Tâche 5 : augmenter le score lorsqu'on récolte une ressource (10 minutes)

Les élèves décident de la façon dont le score doit être augmenté (par exemple, on l'augmente de 1 à chaque fois que le rover touche une ressource).

Dans un second temps, les élèves cherchent l'instruction permettant d'augmenter le score de 1 :

Il suffit de la placer dans le programme de la glace, à l'intérieur du test « si rover touché », juste en-dessous ou au-dessus de l'instruction « cacher ».



Notes pédagogiques

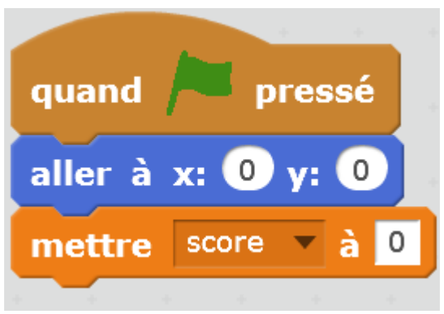
- Les variables X et Y, préexistantes et liées à la position du lutin, sont accessibles dans la palette « mouvement », ainsi que les instructions qui s'y rapportent (affecter une valeur, augmenter cette valeur…). Les variables créées par l'utilisateur, comme le score ici, sont dans la palette « données ». Deux commandes existent :
  - Mettre « score » à 0 : cette commande permet de stocker la valeur zéro dans la variable « score ». On peut changer « 0 » en n'importe quelle autre valeur.
  - Ajouter à « score » 1 : cette commande prend l'ancienne valeur de la variable score et lui ajoute 1 : cette nouvelle valeur est maintenant enregistrée dans la variable score. C'est la commande qui nous intéresse ici.
- Pour se familiariser avec ces commandes, nous conseillons de laisser les élèves les expérimenter quelques minutes avec l'affichage des variables qu'ils manipulent.
- Un exercice en ligne permet de se familiariser avec les manipulations de variable.

Tâche 6 : initialiser le score à zéro (10 minutes)

Les élèves testent à plusieurs reprises ce qui se passe quand le rover touche la glace. Le programme semble bien fonctionner (la glace est présente, le rover la touche, elle disparait et le score est augmenté de 1). Cependant, si on arrête le programme et qu'on le relance, le score n'est pas remis à zéro.
Pour initialiser la variable à zéro, il suffit d'ajouter l'instruction « mettre score à 0 » en début de programme.

Notes pédagogiques :

- *A priori*, cette initialisation peut être faite dans le programme de n'importe quel lutin : l'important étant qu'elle soit faite une fois, et une seule. Mais le score est une variable qui sera sans doute manipulée par d'autres lutins (la végétation, quand on l'aura importée). Il n'y a pas de raison de privilégier la glace par rapport à la végétation. Pour cette raison, nous conseillons de le faire dans le programme du rover (qui est notre programme « principal »), juste en dessous de l'initialisation de sa position.
- On peut aussi décider que notre programme principal est celui de l'arrière-plan plutôt que celui d'un lutin. Dans ce cas, on mettra toutes les initialisations dans l'arrière-plan, sous une commande « quand drapeau vert pressé ».
- Lorsqu'on crée une variable, c'est une très bonne habitude à prendre que de l'initialiser tout de suite !



Attention : ici, on est dans le programme du rover !

Tâche 7 : faire réapparaître une ressource à une position aléatoire (15 minutes)

Le jeu peut être rendu plus amusant si les ressources réapparaissent, après avoir été récoltées, mais pas toujours au même endroit. Une position aléatoire est préférable. Pour cela, il faut, utiliser l'instruction « aller à… », que les élèves connaissent déjà, ainsi qu'une nouvelle instruction disponible dans la catégorie verte « **opérateurs** ». Cette nouvelle commande s'appelle « **nombre aléatoire entre … et …** ».
Puisque, dans l'écran de *Scratch*, l'abscisse X varie entre -240 à +240 et l'ordonnée Y varie entre -180 à +180, le positionnement du lutin n'importe où sur la scène est donc donné par la commande :

Il n'est plus utile de cacher le lutin, puisqu'il est simplement déplacé. Le programme, pour la glace, devient donc :

On peut également faire en sorte que la position initiale de la ressource soit elle-même aléatoire plutôt que fixée (à X=126 et Y=87 dans notre exemple).

Notes pédagogiques :

- Le guidage de l'enseignant peut être assez léger. Par exemple, faire rappeler par un élève les valeurs entre lesquelles varient X et Y, puis travailler sur le vocabulaire en cherchant un synonyme de « au hasard » (aléatoire).
- Dès lors, les élèves vont chercher, dans la catégorie « opérateur », l'instruction qui permet de donner une valeur au hasard entre -240 et 240 (pour X) et entre -180 et 180 (pour Y). Ils trouveront sans problème.

Tâche 8 : importer une nouvelle ressource (la végétation) et refaire le même travail que pour la glace (20 minutes)

Les élèves doivent maintenant introduire une seconde ressource (la végétation) et refaire tout le travail effectué avec la glace :

- Importer le lutin « végétation »
- Tester indéfiniment si ce lutin touche le rover et, si c'est le cas :
  - augmenter le score de 1
  - faire réapparaître le lutin ailleurs sur la carte (position aléatoire).

Cette tâche est très utile aux élèves car elle leur permet de reprendre et consolider les différentes notions abordées précédemment.

Conclusion et traces écrites

La classe synthétise collectivement ce qui a été appris au cours de ces différents travaux :

- *On peut créer différentes variables dans un programme. Il est préférable d'initialiser chaque variable, c'est-à-dire lui donner une valeur au lancement du programme.*
- *Un programme informatique peut générer des nombres aléatoires. Ainsi, chaque exécution peut donner un résultat différent.*

Les élèves notent ces conclusions dans leur cahier de sciences. L'enseignant, quant-à-lui, met à jour l'affiche « qu'est-ce que l'informatique ? ».

Prolongement

Les variables, instructions conditionnelles et boucles peuvent être retravaillées à l'occasion de plusieursexercices en ligne. Attention, ces exercices n'utilisent pas *Scratch*.

Ranger sa chambre

Pas de réponse	Réponse **fausse**	Réponse **juste**
0	0	+12

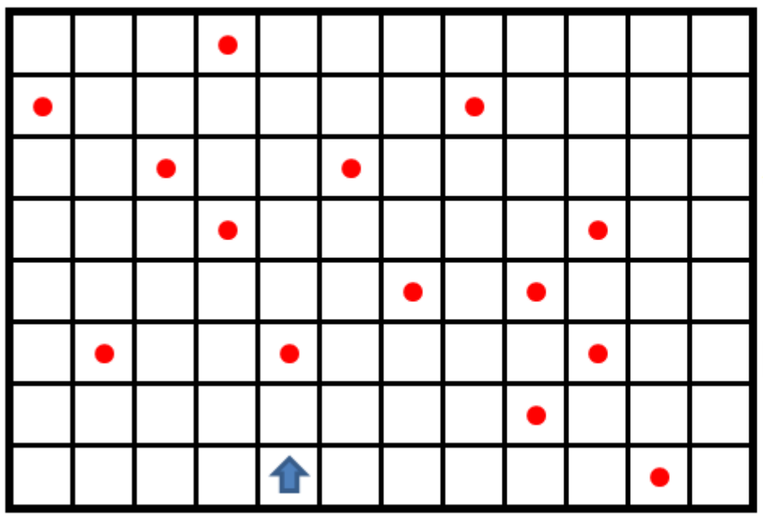
Dans ce sujet, vous ne pouvez pas perdre de points. Plus vous vous rapprochez de la meilleure solution, plus vous gagnez de points.

Pour l'aider à ranger sa chambre représentée par la grille ci-dessous, Castor a construit un robot, représenté par une flèche bleue. Les déplacements du robot peuvent être programmés grâce aux quatre commandes suivantes :

- **A** pour *Avance* : le robot avance d'une case s'il le peut.
- **F** pour *Fonce* : le robot avance jusqu'à rencontrer le bord de la chambre.
- **G** pour *Gauche* : le robot tourne d'un quart de tour vers la gauche.
- **D** pour *Droite* : le robot tourne d'un quart de tour vers la droite.

Pour programmer le robot, écrivez une suite de commandes (10 maximum) dans la zone de texte à côté de la grille, puis clique sur "Exécuter". Il exécutera les commandes une par une, dans l'ordre, puis recommencera depuis le début, et ceci dix fois de suite avant de s'arrêter.

Ecrivez un programme qui permet au robot de passer sur le plus d'objets possibles (qu'il va ramasser). Vous pouvez faire autant d'essais que vous voulez, c'est le meilleur qui compte.



Score : objets
Vos commandes :

Exécuter

Jeu des variables ☆☆☆★

Version ☆☆ | **Version ☆☆☆** | **Version ★★★★**

Un programme est une séquence d'instructions exécutées les unes après les autres. Dans un programme, on manipule une ou plusieurs **variables**. Une variable peut être vue comme le nom d'une boîte dans laquelle on stocke un nombre.

Une instruction d'un programme peut modifier le nombre stocké dans une variable. Par exemple, l'instruction « a = a + 1 » ajoute 1 au nombre stocké dans la variable « a ». Commencez par étudier les exemples ci-dessous pour bien comprendre comment fonctionnent les variables.

	Avant :	**Programme :**	**Après :**
Exemple 1	a vaut 0	a = 4	a vaut 4
Exemple 2	a vaut 2	a = a + 1	a vaut 3
Exemple 3	a vaut 1	a = a + 1 a = a + 3	a vaut 5
Exemple 4	a vaut 4	a = 1 a = a + 2	a vaut 3

À vous de jouer ; complétez les cases dans la colonne de droite :

	Avant :	**Programme :**	**Après :**
Question 1	a vaut 2	a = 3	a vaut ☐
Question 2	a vaut 1	a = a + 1	a vaut ☐
Question 3	a vaut 6	a = 4 a = a + 3	a vaut ☐

Valider | Recommencer

Vous n'avez pas encore obtenu de points sur cette version.

<< Etape III-2.3 | Séquence III-2 | Etape III-2.5 >>

Source URL: https://www.fondation-lamap.org/fr/page/34522/1-2-3-codez-activites-cycle-3-etape-24-recolter-des-ressources-gerer-son-score