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 2.8: Pimenter un peu le jeu

Résumé	Les élèves finalisent leur jeu vidéo en ajoutant quelques éléments pour le rendre plus palpitant : un compte à rebours, une tornade qui se déplace aléatoirement et de plus en plus vite, etc. Les concepts abordés lors des séances précédentes, test, boucle, variable, événement, sont ainsi tous remobilisés.
Notions	« Machines » : • Les machines qui nous entourent ne font qu'exécuter des ordres (instructions) • En combinant des instructions élémentaires, nous pouvons leur faire exécuter des tâches complexes « Algorithmes » • Un algorithme est une méthode permettant de résoudre un problème. • Une boucle permet de répéter plusieurs fois la même action • Certaines boucles, dites « infinies », ne s'arrêtent jamais. • Certaines boucles, dites « itératives » sont répétées un nombre prédéfini de fois. • Certaines boucles, dites « conditionnelles », sont répétées jusqu'à ce qu'une condition soit remplie. « Langages » : • Pour donner des instructions à une machine, on utilise un langage de programmation, compréhensible à la fois par la machine et par l'être humain • *Scratch* est un environnement de programmation graphique, qui utilise un langage simple. • Un programme est l'expression d'un algorithme dans un langage de programmation • Certaines instructions ne s'exécutent qu'au déclenchement d'un évènement : on parle de programmation évènementielle. • Certaines instructions s'exécutent à la suite les unes des autres : on parle de programmation séquentielle. • L'exécution d'un programme est reproductible (si les instructions ne changent pas, ni les données à manipuler, le programme donne toujours le même résultat)
Matériel	Pour chaque binôme • Un ordinateur avec *Scratch* et le programme enregistré à la séance précédente

Situation déclenchante

Les élèves auront sans doute remarqué que leur jeu est jouable mais qu'il manque d'intérêt car il ne présente pas de difficulté majeure. Si l'on fait attention aux obstacles, le jeu peut ne jamais s'arrêter.

La classe réfléchit collectivement à une façon de mettre fin au jeu et d'introduire de la difficulté. Plusieurs options sont possibles :

- option 1 : ajouter un compte à rebours. Quand le temps imparti est écoulé, le jeu s'arrête. La performance du joueur s'apprécie au nombre de « vies » qu'il lui reste et au nombre de ressources qu'il a récoltées (son « score »).
- option 2 : ajouter un nouvel élément qui va rendre le jeu de plus en plus difficile, et y mettre fin à un moment donné. Par exemple, on peut introduire un nouveau piège (une tornade, en écho aux problèmes météorologiques évoqués dans la Séance 1.3 et la Séance 1.4, sur le codage binaire). La tornade se déplace de plus en plus vite, et on ne peut pas prévoir sa direction. À un moment ou un autre, on ne pourra plus l'éviter et le jeu prendra alors fin.

Notes pédagogiques :

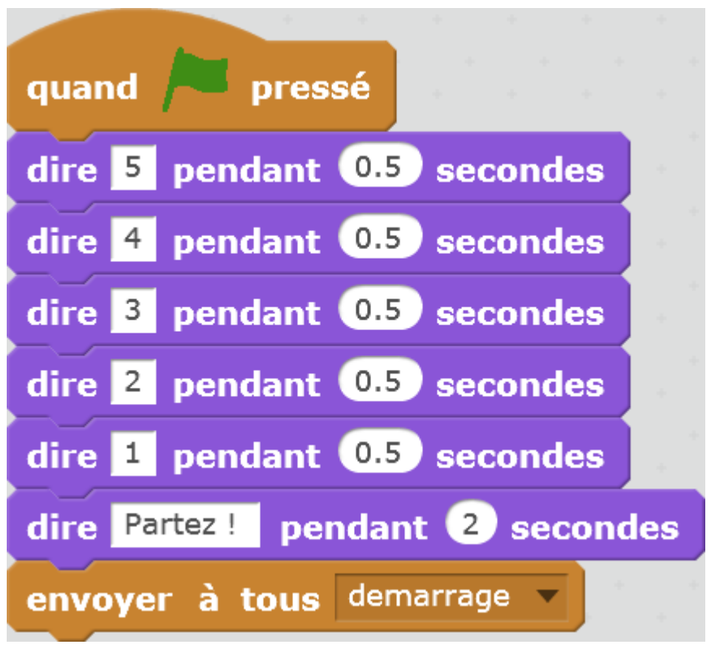
- Il est très probable que les élèves imaginent d'autres façons de faire. Nous conseillons à l'enseignant d'engager un débat dans la classe afin de choisir la ou les options qui seront suivies (rien n'impose que tous les groupes choisissent la même option !).
- L'enseignant veillera cependant à ce que les élèves aient une idée de la façon dont ils vont procéder. Si une option semble séduisante mais irréaliste, mieux vaut sans doute opter pour une autre qui soit faisable !

Nous décrivons ci-dessous ces 2 options citées, ainsi que quelques autres qui ne sont pas forcément une manière de pimenter le jeu, mais de le rendre plus présentable ou d'éviter quelques bugs.
Les tâches ci-dessous sont de difficulté très variables, et toutes **indépendantes** les unes des autres.

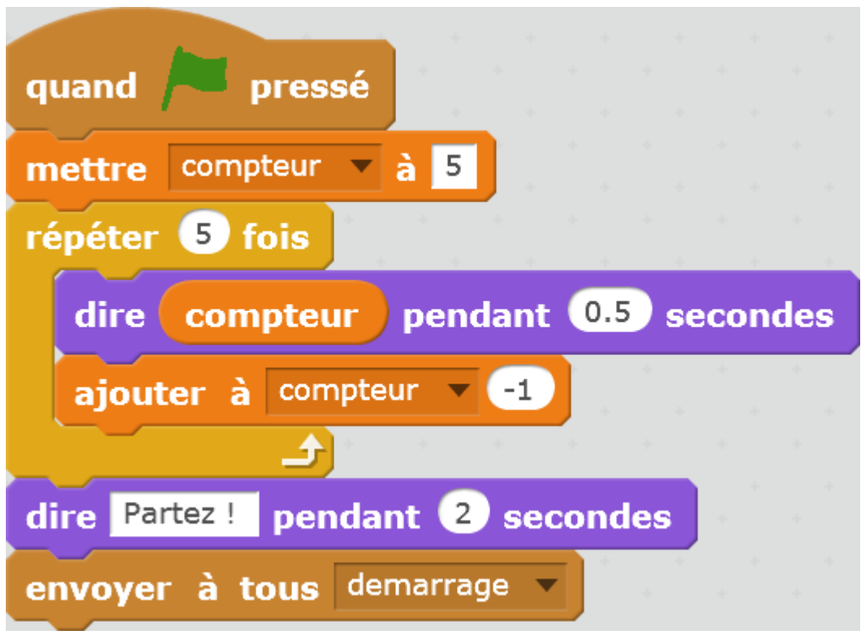
Notre conseil : faire au moins la tâche 2 ou la tâche 3 pour que le jeu présente un intérêt.

Tâche 1 : faire apparaître un compte à rebours au lancement du jeu (15 minutes)

On peut décider que le jeu ne démarre qu'après un décompte *5, 4, 3, 2, 1, partez !* Ceci peut se faire très simplement, comme ceci :



… ou de façon plus subtile : en utilisant une nouvelle variable et une boucle.



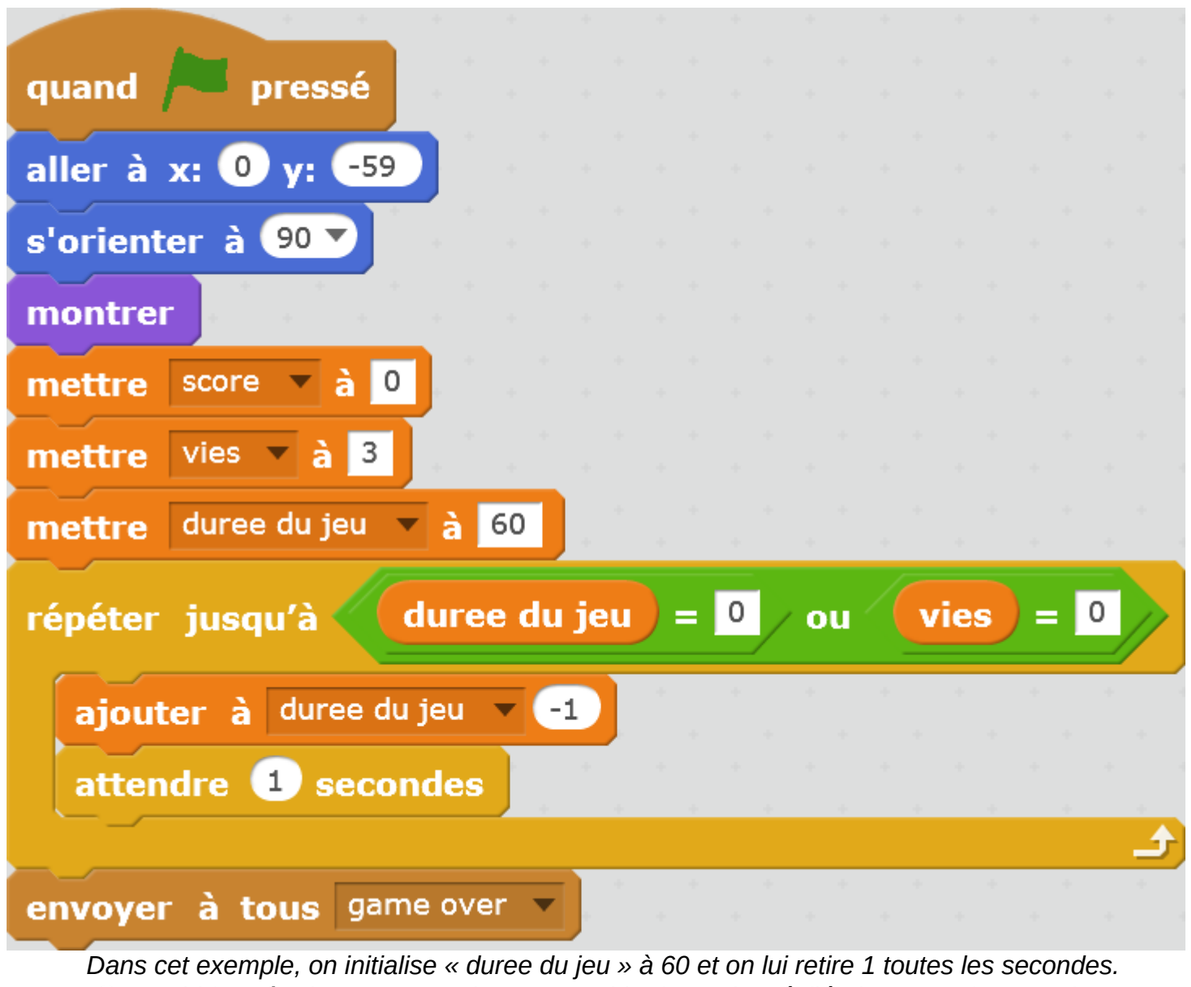
Suivant le niveau des élèves, on peut se contenter de la manière « simple » ou leur demander d'utiliser une variable et une boucle. On peut aussi, pour la variante plus complexe, leur donner tous les éléments, dans le désordre et non reliés entre eux, et leur demander de tout remettre en ordre pour que ce programme donne le même résultat que le premier.

Tâche 2 : limiter la durée du jeu (15 minutes)

Notes pédagogiques :

- Cette tâche est similaire à la première, ci-dessus, mais ici il faudra absolument passer par la forme complexe (variable plus boucle) car il n'est pas pensable d'écrire à la main un décompte, seconde par seconde, qui durerait plusieurs minutes.
- Si les élèves ont déjà mis en place un compte à rebours (tâche 1), ils réussiront très facilement cette nouvelle tâche. Sinon, ils prendront un peu de temps et auront peut-être besoin d'être (légèrement) guidés.
- Cette tâche permet de revenir sur les variables, les tests, les boucles, ainsi que les opérateurs logiques.

Limiter la durée du jeu est une façon très simple de le rendre plus intéressant. Il suffit de créer une variable « duree du jeu », de lui donner une valeur initiale et de la diminuer au fur et à mesure, en introduisant une temporisation afin de pouvoir contrôler la vitesse du processus.
Le programme s'arrête (message « game over ») soit lorsque le compte à rebours est arrivé à 0, soit quand le nombre de vies est arrivé à 0.
Le programme du rover doit être modifié pour comporter :

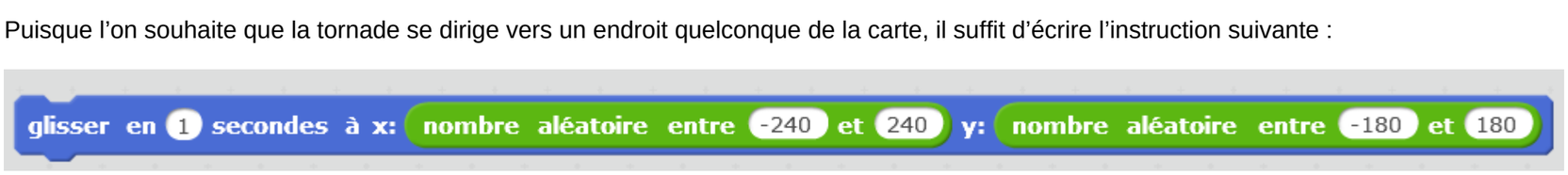


Dans cet exemple, on initialise « duree du jeu » à 60 et on lui retire 1 toutes les secondes.
On peut bien sûr changer ces valeurs pour obtenir une longévité plus ou moins grande.

Tâche 3 : ajouter une tornade qui se déplace aléatoirement (15 minutes)

La création du lutin à partir d'une image et à l'initialisation de sa position sont désormais connues des élèves. On peut, par exemple, faire partir la tornade du coin inférieur gauche (X = -230 et Y = -170).
Il faut ensuite trouver une façon de la déplacer vers un endroit aléatoire. Si l'on souhaite que ce mouvement soit instantané, il suffit de changer les valeurs de X et de Y… mais il serait préférable de voir la tornade de déplacer. La commande adéquate est ‹**glisser en … secondes à x=… et y=…** », disponible dans la catégorie « mouvement ».

Puisque l'on souhaite que la tornade se dirige vers un endroit quelconque de la carte, il suffit d'écrire l'instruction suivante :



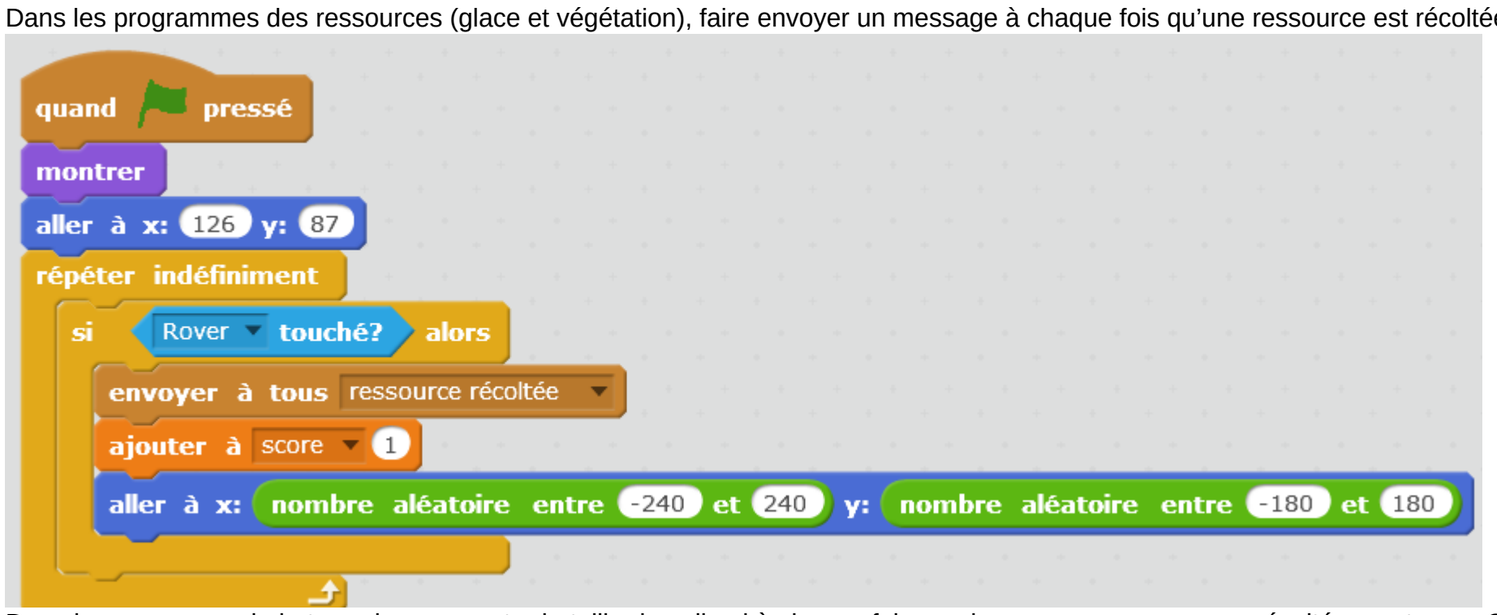
Les élèves, en cliquant plusieurs fois sur cette instruction, confirment que la tornade se déplace dans une direction aléatoire à chaque fois. Ce déplacement se fait toujours en 1 seconde… donc si le point d'arrivée est proche, le déplacement est très lent, et s'il est éloigné, plus rapide.
Reste maintenant à connecter cette instruction au reste du programme de la tornade (on a changé le temps en « 2 secondes » de façon à ralentir un peu la tornade).

Il ne faut pas oublier l'essentiel : si le rover touche la tornade, la partie prend fin. On ajoute donc l'instruction suivante au programme de la tornade :

Tâche 4 : faire grossir la tornade (15 minutes)

Pour corser encore un peu le jeu, on peut faire en sorte que la tornade grossisse au fur et à mesure que des ressources sont récoltées. À ce stade du projet, une telle tâche ne présente pas de difficulté majeure.

- Dans les programmes des ressources (glace et végétation), faire envoyer un message à chaque fois qu'une ressource est récoltée.



- Dans le programme de la tornade, augmenter la taille de celle-ci à chaque fois que le message « ressource récoltée » est reçu. On utilise pour cela la commande **« ajouter… à la taille** » disponible dans la catégorie « apparence ».

- Puisque, désormais, on modifie la taille de la tornade dans le programme, alors il faut penser à initialiser cette taille au lancement du programme, grâce à la commande « **mettre à 100% de la taille initiale** » de la catégorie « apparence ».

Tâche 5 : faire accélérer la tornade au fur et à mesure (20 minutes)

Cette tâche, très difficile, est plutôt réservée à des élèves de collège (cycle 4). Néanmoins, il est possible que certains élèves de cycle 3 très en avance puissent avoir besoin d'un *challenge*. Voilà de quoi les occuper !

Il s'agit de faire en sorte que la tornade accélère à chaque fois que l'on a récolté une ressource (glace ou végétation). Pour faire accélérer la tornade, il faut créer une variable « vitesse_tornade » (initialisée à 1, dans le même programme que pour les autres variables) et l'augmenter, par exemple de 10% à chaque événement (astuce : augmenter de 10% la vitesse revient à la multiplier par 1,1).

Le programme de la tornade est donc modifié :

Attention, il faut tenir compte de la valeur de « vitesse_tornade » dans le programme qui commande son mouvement (instruction « glisser »). Cela se fait par exemple en incluant « vitesse_tornade » dans le calcul du temps de glissement, comme ceci :

Ainsi, plus « vitesse_tornade » est grande, plus le temps mis pour effectuer un mouvement donné est court : la tornade est de plus en plus rapide, c'est bien ce que l'on cherche.

Tâche 6 : simuler un monde torique (joindre les côtés de la scène) (20 minutes)

Le jeu serait plus amusant si le rover n'était pas bloqué par les bords de la scène. Il faudrait faire en sorte que, lorsqu'il atteint le bord droit de la scène, il continue sa course et réapparaisse sur le bord gauche, et inversement. Même chose avec les bords haut et bas.

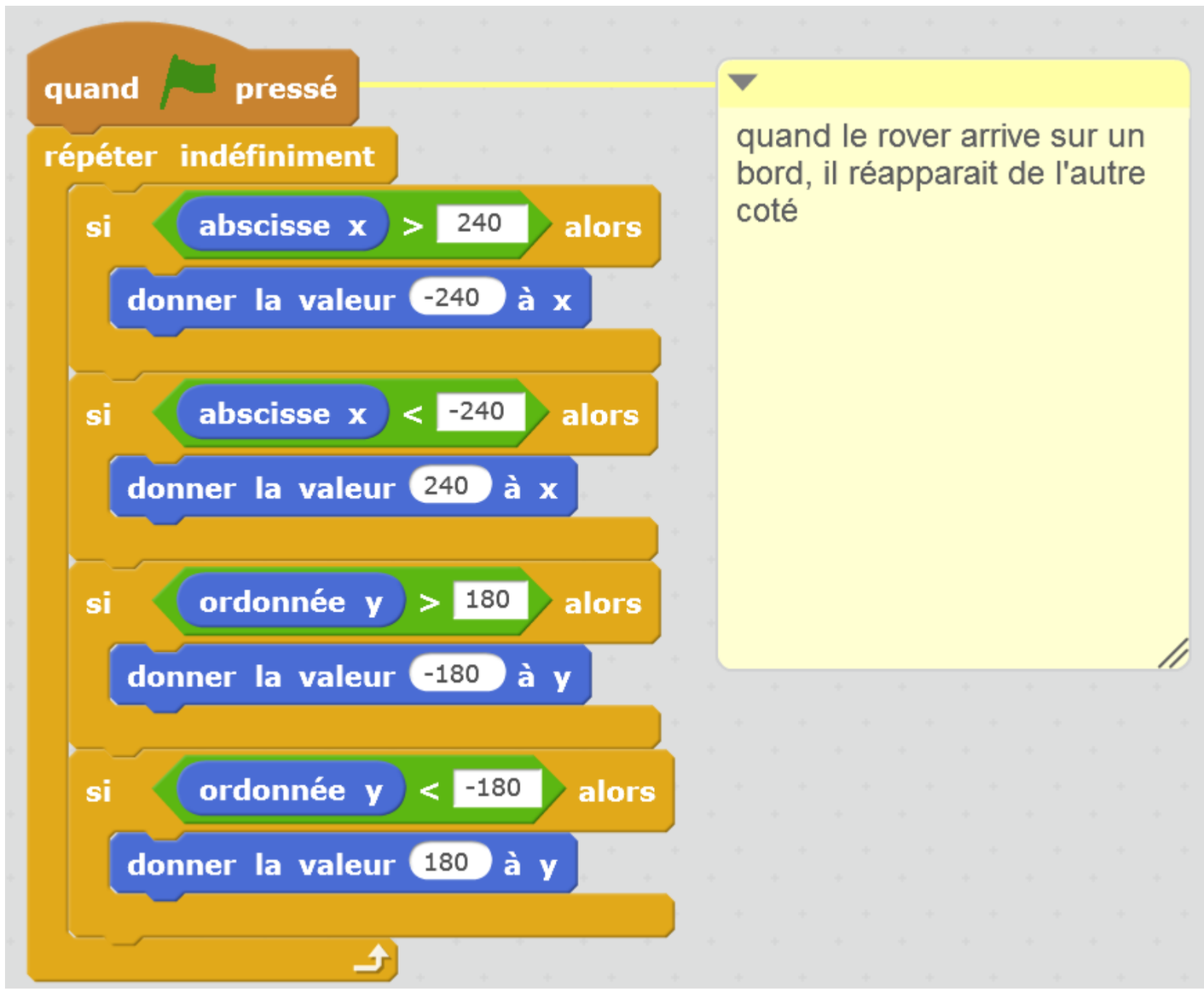
Notes scientifiques

- Un espace plan que l'on plie de façon à ce que le bord gauche et le bord droit se rejoignent s'appelle un espace « cylindrique ». Si l'on rajoute un second pliage permettant de joindre le bord haut et le bord bas, on parle alors d'un espace « torique ». Cet espace ressemble à un *donut* ou une chambre à air.
- Beaucoup de jeux vidéo sont basés sur un monde torique, même si un tel monde ne ressemble pas à une planète réelle (sur Terre, lorsqu'on atteint le pôle Nord, on ne se retrouve pas au pôle Sud !).

Il laisse les élèves tâtonner et les guide en cas de difficulté, tout d'abord en explicitant leur algorithme : si l'abscisse X du rover dépasse 240 (extrémité droite), alors elle doit passer à -240 (extrémité gauche). Ensuite, il peut montrer les différents blocs nécessaires à la construction du programme : une boucle «répéter indéfiniment», une structure de contrôle « si … alors », un opérateur « supérieur à », la valeur de la variable X (bloc bleu « abscisse X »), et l'instruction permettant de changer cette valeur (bloc bleu « donner la valeur … à X »). Les blocs s'emboitent de la façon suivante :

Note pédagogique
Il est possible, selon la forme des lutins, qu'il soit nécessaire d'introduire une petite marge (au lieu de prendre 240 comme valeur d'extrémité de l'écran, prendre 235).

Une fois que chacun a compris comment programmer le passage du bord droit au bord gauche, il est facile de programmer le passage du bord gauche au bord droit puis, verticalement (variable Y), du bord haut au bord bas, et du bord bas au bord haut.
Finalement, le sous-programme du rover permettant de simuler un monde torique est :



Notes pédagogiques

- La capture d'écran ci-dessus montre qu'il est possible d'ajouter des commentaires à un programme. C'est une très bonne habitude à prendre, de manière à rendre le programme plus clair pour soi, et surtout pour les autres qui vont le lire !
- À ce stade, il devient nécessaire de supprimer l'instruction « rebondir si le bord est atteint » dans les sous-programmes permettant de piloter le rover à l'aide des flèches.

Tâche 7 : éviter que les ressources et les pièges ne se superposent (20 minutes)

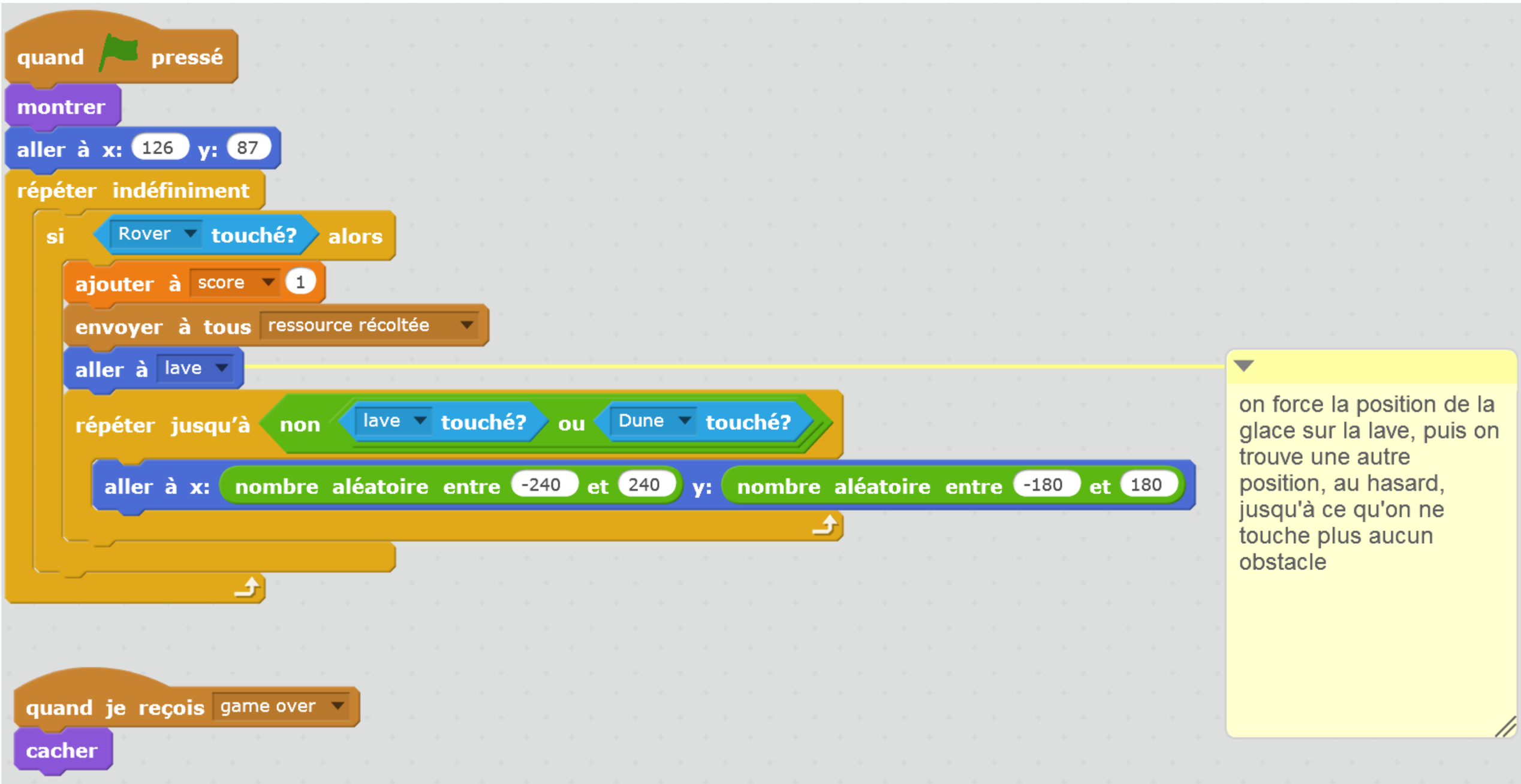
Tout comme la tâche 5 ci-dessus, cette tâche cible les élèves de cycle 4 ou les élèves de cycle 3 qui seraient très en avance sur le reste de la classe.
Lorsque les ressources (glace, végétation) sont récoltées, elles réapparaissent aléatoirement sur la scène. Il est parfaitement possible qu'une ressource réapparaisse à l'endroit où un piège (dune ou lave) se trouve déjà, ce qu'il faut éviter car on ne peut pas avoir 2 objectifs contradictoires : récolter la ressource et éviter l'obstacle.

Comment faire en sorte que ce cas de figure ne puisse pas se produire ? Il faut tirer au hasard une nouvelle position tant que la ressource touche un piège. Mais, comme initialement la ressource ne touche pas un piège, la boucle ne va pas s'exécuter. L'astuce consiste alors à introduire une étape préalable, pour forcer la boucle à s'exécuter une première fois. L'algorithme devient :

- 1/ Placer la ressource n'importe où par tirage aléatoire de ses coordonnées (ou bien, si on préfère, la placer sur un piège)
- 2/ Puis, effectuer la boucle suivante : tant que la ressource n'est pas à un emplacement libre de tout piège, lui donner une nouvelle position aléatoire.

Les élèves devront être guidés, soit en leur donnant l'astuce ci-dessus (après les avoir laissé chercher un peu), soit en leur fournissant le programme final et en leur demandant de l'analyser pour comprendre ce qu'il fait, et pourquoi.

Le programme final de la glace ou de la végétation devient donc :



Conclusion et trace écrite

Le jeu vidéo est désormais suffisamment intéressant pour permettre aux élèves d'y jouer en se lançant des défis.

Les élèves mettent à jour la liste des instructions *Scratch* qu'ils connaissent.

L'enseignant anime un bilan collectif permettant aux élèves d'exprimer ce qu'ils ont appris au cours de ce projet, les difficultés qu'ils ont eues, les désirs éventuels que cela a fait naître (certains ont sans doute déjà commencé à faire d'autres programmes *Scratch* à la maison), etc.

Notes pédagogiques :

- Cette activité de programmation a probablement été nouvelle pour la plupart des élèves. Afin qu'ils n'oublient pas comment utiliser *Scratch*, mais aussi afin de libérer leur créativité, nous conseillons de leur proposer, plus tard dans l'année, de créer un projet personnel. Il peut s'agir d'un jeu vidéo (projet assez complexe comme on l'a vu), ou plus simplement d'une carte animée (pour Halloween, Noël…), ou encore des questionnaires interactifs… les possibilités sont immenses !
- C'est l'objet de l'étape suivante que d'explorer quelques-unes des fonctionnalités supplémentaires de *Scratch*, afin de donner des idées d'autres activités possibles.

<< Etape III-2.7 Séquence III-2 Etape III-2.9 >>

Source URL: https://www.fondation-lamap.org/fr/page/34526/1-2-3-codez-activites-cycle-3-etape-28-pimenter-un-peu-le-jeu