FONDATION
La main à la pâte
POUR L'ÉDUCATION À LA SCIENCE

Cycle 3

Programmer avec Scratch : algorithmique branchée (cycle 3)

Une séquence du projet *1,2,3... CODEZ !*

Résumé

Au cours de cette séquence en neuf étapes, les élèves découvrent la programmation avec Scratch, dans l'optique de programmer un jeu vidéo pour simuler la mission d'exploration spatiale préparée lors de la séquence « Préparer la mission ». Cette séquence 2 se passe essentiellement devant l'ordinateur. Néanmoins, on y trouve quelques séances débranchées (sans ordinateur), destinées à mieux comprendre certains concepts, comme les variables ou les opérateurs logiques.

Séquence 2 : Simuler la mission dans *Scratch*

Cette séquence est consacrée à la programmation d'un jeu vidéo qui va simuler notre mission d'exploration spatiale, dans la continuité de ce qui a été fait lors de la séquence 1.
Cette séquence 2 se passe essentiellement devant l'ordinateur. Néanmoins, on y trouve quelques séances débranchées (sans ordinateur) destinées à mieux comprendre certains concepts, comme les variables ou les opérateurs logiques. Ces séances débranchées, idéalement, doivent être mises en œuvre sur un autre temps que le temps dédié à la programmation (en séance de mathématiques ou français, selon les cas), afin de ne pas interrompre le projet.

Avant de se lancer dans une activité de programmation : quelques conseils

Programmer un jeu vidéo est extrêmement motivant pour les élèves… mais pour que cela se passe bien, il faut prendre quelques précautions.

Scratch, un environnement idéal pour apprendre à programmer

Il existe de très nombreux outils d'apprentissage de la programmation. Nous avons fait le choix d'utiliser le logiciel *Scratch*[19], en raison de sa qualité remarquable, de sa simplicité d'utilisation, de sa gratuité, et de sa communauté très active, y compris dans le champ de l'éducation (primaire et collège).
Ce choix est cependant arbitraire et, bien entendu, l'enseignant peut suivre la même progression en utilisant d'autres environnements (comme *snap*, très similaire à *Scratch*, et lui aussi gratuit, ou *kodu* et *tangara*, qui sont payants…).
Une alternative à *Scratch* « seul » est possible dans le cas où la classe possède un robot, dont la programmation reprend les mêmes concepts, mais appliqués à un objet physique. Le robot Thymio, par exemple, se programme à l'aide du langage VPL et/ou *Scratch* (voir, à ce sujet, la séquence 3 de la progression pour le cycle 2, pages 190 et suivantes).

S'assurer de la maîtrise de quelques compétences TICE de base

Programmer nécessite d'interagir avec un ordinateur, ce qui suppose de maîtriser quelques compétences élémentaires :
- Utiliser le clavier et la souris (c'est très souvent le cas en cycle 3, mais pas toujours) ;
- Savoir lancer un programme en double-cliquant sur son icône ;

19. Scratch est disponible soit en ligne (sans installation préalable, mais qui nécessite un accès Internet de bonne qualité, à cette adresse : https://scratch.mit.edu), soit hors ligne (ce qui nécessite une installation préalable, après avoir téléchargé le logiciel ici : https://scratch.mit.edu/scratch2download).

• Savoir enregistrer son travail dans un fichier, et ce fichier dans un dossier (ou répertoire) ;
• Savoir récupérer le travail que l'on a enregistré auparavant.
Si certains élèves ne maîtrisent pas ces compétences de base, ils pourront les acquérir à travers ce projet de programmation, mais ils risquent de perdre beaucoup de temps lors des premières séances.

Travailler en demi-groupes

L'idéal est de disposer d'un ordinateur pour 2 élèves (3 au maximum). Afin de permettre une telle répartition, et aussi afin de faciliter la gestion de classe pendant les activités de programmation (au cours desquelles l'enseignant est très sollicité), nous conseillons de travailler en demi-groupes : la moitié de la classe programme pendant que l'autre moitié travaille à autre chose, en autonomie.

Préparer l'environnement de travail

Il est essentiel que l'enseignant ait bien préparé l'environnement de travail avant la première séance :
• *Scratch* doit être installé sur toutes les machines (ou accessible en ligne) ;
• Un raccourci vers *Scratch* doit être présent sur le bureau ;
• De même, un répertoire dédié au projet (et à la classe) doit être facilement accessible, depuis le bureau ou sur une clef USB dédiée au groupe. Ce répertoire contiendra les fichiers nécessaires au projet (images à importer, sauvegarde des programmes…). Nous mettons, sur le site Web dédié au projet, tous fichiers utiles (voir page 349).
– Les utilisateurs les plus avancés pourront mettre les fichiers utiles directement dans les sous-répertoires du logiciel *Scratch* (par exemple : <*Scratch*>/Medias/Costumes/MissionMars et <*Scratch*>/Medias/Background/MissionMars, où <*Scratch*> désigne le répertoire d'installation du logiciel)

Faire soi-même le projet avant !

C'est sans doute le conseil le plus important, bien qu'il paraisse aller de soi. Il est primordial que l'enseignant ait pris 2 ou 3 heures sur son temps de préparation AVANT la première séance de classe, pour se familiariser avec *Scratch* et pour réaliser les tâches que les élèves auront à réaliser au fil du projet. Sinon, il risque fort de ne pas être capable d'aider les élèves quand ceux-ci en auront besoin.
Ce n'est pas difficile (il suffit de suivre le déroulement proposé dans cette séquence), et c'est même assez amusant !

Étapes de réalisation du projet

Le tableau récapitulatif ci-après liste les différentes étapes et, pour chaque étape, les différentes tâches à réaliser pour construire le jeu vidéo. Si l'enseignant choisit un autre scénario pour le jeu, il faudra bien entendu adapter cette progression.
À noter : dans la plupart des classes de cycle 3, réaliser l'ensemble du projet nécessitera 6 ou 7 séances de 1 heure. Certains binômes auront produit un jeu plus complet et complexe que d'autres, mais tout le monde aura produit un jeu satisfaisant et valorisant.
Nous conseillons, au moins au début du projet, de faire 2 séances par semaine, afin d'éviter que les élèves aient oublié, d'une séance à l'autre, ce qu'ils ont appris, car la programmation est une activité véritablement nouvelle pour eux.

Des repères de difficulté

Pour se repérer dans ces différentes étapes et tâches, nous utiliserons un symbole inspiré des pistes de ski :

●	Piste verte : facile Tous les élèves y arrivent sans problème.
●	Piste bleue : moyenne La plupart des élèves y arrivent seuls, certains ont besoin d'être un peu guidés.
●	Piste rouge : difficile La plupart des élèves ont besoin d'être guidés (plus ou moins selon leur aisance).
●	Piste noire : très difficile Tous les élèves ont besoin d'être guidés. Ces tâches sont facultatives.

Sauf indication contraire, toutes les étapes sont branchées ; les durées indiquées sont des durées moyennes.

Étape	Titre	Page	Tâche	
Étape 1 	Découverte de l'environnement de programmation *Scratch*	241	Tâche 0 : démonstration, par l'enseignant, du jeu final (5 minutes)	●
			Tâche 1 : lancer *Scratch* et découvrir son interface (10 minutes)	●
			Tâche 2 : explorer librement *Scratch* (15 minutes)	●
			Tâche 3 : faire de petits exercices (20 minutes)	●
Étape 2 	Planter le décor, et sauvegarder son travail	250	Tâche 1 : changer le lutin (5 minutes)	●
			Tâche 2 : changer l'arrière-plan (5 minutes)	●
			Tâche 3 : enregistrer son programme *Scratch* (5 minutes)	●
Étape 3 	Piloter le rover	253	Tâche 1 : faire avancer le rover vers la gauche (10 minutes)	●
			Tâche 2 : faire avancer le rover dans n'importe quelle direction (5 minutes)	●
			Tâche 3 : piloter le rover à l'aide des flèches (15 minutes)	●
			Tâche 4 : rebondir sur les bords (5 minutes)	●
			Tâche 5 : initialiser la position du rover (5 minutes)	●
			Tâche 6 : comprendre les coordonnées X et Y du rover (20 minutes)	●
Étape 4 	Récolter des ressources, gérer son score	259	Tâche 1 : importer une ressource (la glace) sous la forme d'un nouveau lutin (5 minutes)	●
			Tâche 2 : faire dire « bravo » à la ressource lorsqu'elle est touchée par le rover (20 minutes)	●
			Tâche 3 : faire disparaître la ressource quand elle est touchée (10 minutes)	●
			Tâche 4 : créer une variable « score » (5 minutes)	●
			Tâche 5 : augmenter le score lorsqu'on récolte une ressource (10 minutes)	●
			Tâche 6 : initialiser le score à zéro (10 minutes)	●

Étape 4 (suite) 	Récolter des ressources, gérer son score		Tâche 7 : faire réapparaître une ressource à une position aléatoire (15 minutes)	
			Tâche 8 : importer une nouvelle ressource (la végétation) et refaire le même travail que pour la glace (20 minutes)	
Étape 5 	Activités branchées et débranchées pour mieux s'approprier certains concepts algorithmiques Cette étape, optionnelle, ne concerne pas la programmation du jeu vidéo et ne doit pas l'interrompre (à faire en parallèle, sur le temps de mathématiques ou de français)	266	Activité 1 : évaluation formative sur le concept de boucle (branchée, 10 à 20 minutes)	
			Activité 2 : un jeu de cartes pour s'approprier la notion de variable (débranchée, 1 heure)	
			Activité 3 : un jeu de cartes pour travailler les opérateurs logiques (débranchée, 1 heure)	
			Activité 4 : comprendre qu'un algorithme n'est pas toujours parfait : le jeu du voyageur de commerce (débranchée, 1 heure)	
Étape 6 	Éviter des obstacles, gérer son nombre de vies	283	Tâche 1 : ajout de nouveaux lutins (5 minutes)	
			Tâche 2 : création et initialisation d'une variable « nombre de vies » (5 minutes)	
			Tâche 3 : perdre une vie quand le rover touche la lave (30 minutes)	
			Tâche 4 : refaire la même chose avec la dune (10 minutes)	
Étape 7 	Mettre fin au jeu : « game over »	287	Tâche 1 : faire apparaître « game over » quand le nombre de vies vaut 0 (15 minutes)	
			Tâche 2 : arrêter le jeu quand apparaît « game over » (15 minutes)	
Étape 8 	Pimenter un peu le jeu	290	Tâche 1 : faire apparaître un compte à rebours au lancement du jeu (15 minutes)	
			Tâche 2 : limiter la durée du jeu (15 minutes)	
			Tâche 3 : ajouter une tornade qui se déplace aléatoirement (15 minutes)	
			Tâche 4 : faire grossir la tornade peu à peu (15 minutes)	
			Tâche 5 : faire accélérer la tornade peu à peu (20 minutes)	
			Tâche 6 : simuler un monde torique (joindre les côtés de la scène) (20 minutes)	
			Tâche 7 : éviter que les ressources et les pièges ne se superposent (20 minutes)	
Étape 9 	Prolongements possibles en *Scratch*	298	À ce stade, le projet est terminé. Nous proposons ici quelques pistes pour explorer d'autres fonctionnalités de *Scratch*, par exemple pour alimenter de futurs projets personnels d'élèves.	

Étape 1 – Découverte de l'environnement de programmation *Scratch*

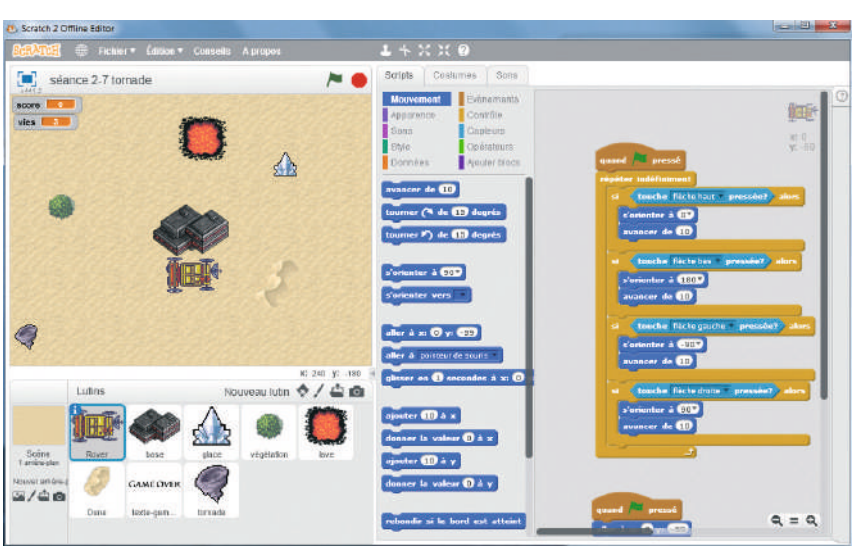
Résumé	Les élèves découvrent *Scratch*, un environnement de programmation adapté à l'école primaire. Ils apprennent à lancer un programme et à enchaîner quelques instructions simples.
Notions (*cf. scénario conceptuel, page 213*)	« Machines » • Les machines qui nous entourent ne font qu'exécuter des 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. « 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 événement : on parle de « programmation événementielle ». • 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 la classe : • Un vidéo projecteur • Version agrandie en A3 ou A2 de la Fiche 32, page 249 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 (voir note en bas de la page 237). Pour chaque élève : • Fiche 32, page 249
Lexique	Programme, script, lutin, instruction, événement
Durée	1 h

Notes pédagogiques

• Apprendre à programmer se fait en programmant, pas en regardant quelqu'un programmer ! Il est intéressant de réfléchir à 2 sur un même problème (probablement plus que de programmer seul), mais il est important d'être actif. Nous conseillons donc de mettre les élèves par petits groupes devant les machines (idéalement, 2 élèves par machine) et de leur demander de « passer la main » (donner le clavier et la souris à son voisin) toutes les 10 ou 15 minutes.
• Nous conseillons, si possible, d'organiser la classe en demi-groupes de façon à ne pas avoir trop de binômes à gérer. Pendant que la moitié de la classe travaille sur *Scratch*, l'autre moitié fait autre chose, en autonomie.
• Si possible, faire 2 séances de *Scratch* par semaine, surtout au début du projet.
• Cette étape de découverte de *Scratch* est, volontairement, très directive (les tâches 0 et 1 sont même une démonstration de l'enseignant !). C'est la seule qui se présente sous cette forme. Tous les binômes doivent accomplir une série de tâches élémentaires. À la fin de chaque tâche, une mise en commun permet de s'assurer que chacun a compris et sait faire. Les autres étapes seront moins dirigées, les élèves devenant plus autonomes et avançant chacun à leur rythme.
• Pour gagner du temps, allumer les ordinateurs avant le début de la séance.

Tâche 0 : démonstration, par l'enseignant, du jeu final (5 minutes)

L'enseignant explique que l'on va simuler la mission d'exploration (à défaut de pouvoir la vivre) à travers un jeu vidéo que l'on va programmer soi-même.
Depuis son poste informatique, il lance *Scratch* et montre le jeu vidéo « final » (celui qu'il a lui-même réalisé à l'avance[20]) sans expliquer comment fonctionne le programme, mais en faisant simplement une démonstration du jeu.



Capture d'écran : version finale du jeu vidéo réalisé par l'enseignant
(NB : un programme corrigé est disponible sur le site du projet)

20. Rappel : il est absolument indispensable que l'enseignant fasse lui-même le projet avant de le proposer aux élèves ! Il suffit de suivre les étapes décrites dans la séquence. Un enseignant débutant en *Scratch* prendra environ 3 heures pour faire l'intégralité du projet (y compris les pistes noires).

Notes pédagogiques

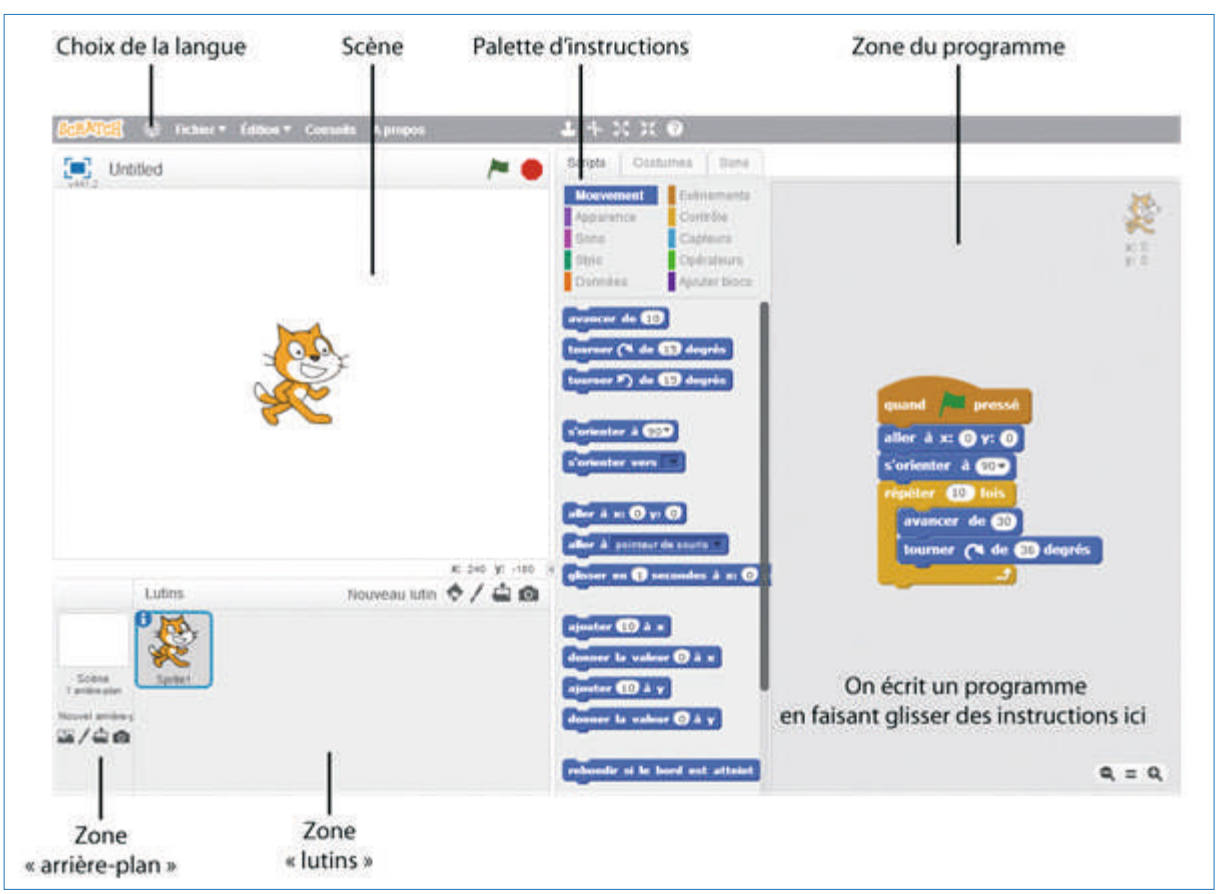
• Cette démonstration est très importante, car elle motive énormément les élèves et les rassure : ils vont vraiment programmer un « vrai » jeu vidéo ! Elle permet par ailleurs de faire le lien entre cette activité de programmation et les activités débranchées de la séquence 1. *« Voici notre rover, que l'on va apprendre à diriger. Il va récolter des ressources pour permettre aux humains de vivre à la base : de l'eau, de la nourriture… (pour chaque ressource récoltée, le score augmente). Mais attention aux pièges ! S'il tombe dans un piège, il perd une vie. S'il n'a plus de vie, la partie est terminée. »*

• Il est important d'expliquer aux élèves que programmer un tel jeu ne se fait pas en une séance mais nécessitera plusieurs séances (typiquement, 6 ou 7 séances en fonction de leur niveau initial de maîtrise de l'ordinateur et de leurs exigences).

Tâche 1 : lancer Scratch et découvrir son interface (10 minutes)

Chaque binôme lance *Scratch* en double-cliquant sur son icône et refuse les éventuelles propositions de mises à jour.

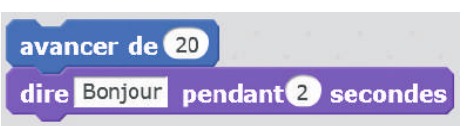
Si c'est la première fois que *Scratch* est utilisé sur cette machine, il se peut que le logiciel soit en anglais. Dans ce cas, les élèves peuvent très facilement le passer en français en cliquant sur l'icône représentant un globe terrestre (en haut à gauche, à côté du logo « *Scratch* »).



L'enseignant explique aux élèves que *Scratch* est un langage de programmation conçu spécifiquement pour apprendre à programmer. Lorsque l'on ouvre le logiciel, il y a un « lutin » à l'écran (un chat). On peut lui donner des instructions simples.

Il réalise une petite démonstration (les élèves devront, en fin de séance, refaire ces petits exercices). Par exemple, pour demander au chat de se déplacer de 10 pas, il suffit de faire glisser l'instruction « avancer de 10 » depuis la palette d'instructions vers la zone du programme. Si l'on clique ensuite sur cette inscription, on remarque que le chat avance bien de 10 pas (1 pas = 1 pixel de l'écran).

Si l'on souhaite avancer de 20 pas, il suffit de changer « 10 » en « 20 » en cliquant dans la zone dédiée. Si maintenant on souhaite que le chat avance de 20 pas, puis dise « Bonjour », il suffit de coller la nouvelle instruction à la fin du programme. L'instruction « dire bonjour » n'existe pas, mais il y a une instruction « dire Hello » dans la catégorie « apparence » de la palette d'instructions. Il suffit de prendre cette instruction puis de remplacer le texte « Hello » en « Bonjour » en cliquant sur ce texte. Écrire un programme se fait simplement en emboîtant des instructions entre elles.



Si maintenant on veut que le chat fasse cela chaque fois que l'on clique sur le drapeau vert (en haut à droite de la scène, le drapeau vert permet de lancer le programme), alors il faut rajouter l'instruction « Quand drapeau vert pressé » à chercher dans la catégorie « événements » des instructions. Cela donne :

Finalement, l'enseignant montre comment supprimer une instruction (ou tout un bloc d'instructions) : il suffit de faire glisser cette instruction (ou ce bloc) depuis la zone du programme vers la palette des instructions.

L'enseignant présente très rapidement l'interface de *Scratch*, qui comprend :

• Une « scène » : c'est là que se déroule le « jeu » (ou, plus généralement, le programme… On peut faire autre chose que des jeux dans *Scratch* !).

• Une zone « lutins » : les lutins sont les personnages ou les objets qui seront manipulés dans le programme (ils peuvent se déplacer, changer de forme, parler, interagir avec les autres lutins…). Lorsqu'on lance *Scratch*, il n'y a qu'un seul lutin affiché à l'écran : un chat (plus tard, on ajoutera d'autres lutins et on supprimera le chat).

• Une zone « arrière-plan », juste à côté des lutins. L'arrière-plan est fixe, contrairement aux lutins qui peuvent bouger. Par défaut, dans *Scratch*, l'arrière-plan est un fond blanc uni (plus tard, on le modifiera).

• Un onglet « script » qui permet d'accéder à :

– Une palette d'instructions (colonne centrale, à droite de la scène). C'est ici que l'on va trouver les instructions (ou « blocs ») que l'on va pouvoir utiliser dans notre programme. Il y a de nombreuses instructions, qui sont regroupées par couleurs (exemple : tout ce qui concerne le mouvement du lutin est dans un onglet bleu foncé, tout ce qui concerne son apparence est dans un onglet violet, etc.).

– Une zone « programme », à droite de la palette d'instructions. C'est ici que l'on va écrire le programme, tout simplement en prenant des instructions depuis la palette et en les faisant glisser dans cette zone.
• Les autres onglets (costumes, sons) sont inutiles pour le moment.

Tâche 2 : explorer librement Scratch (15 minutes)

Les élèves disposent de 15 minutes pour explorer librement *Scratch*.
Pour le moment, ils ne doivent pas chercher à modifier la scène ou le lutin (ce sera fait lors de l'étape suivante, page 250) mais simplement manipuler des instructions simples et les agencer pour observer ce qui se passe. L'enseignant les encourage à explorer les différentes catégories d'instructions, en particulier :

• Catégorie « mouvement » (bleu foncé)
• Catégorie « apparence » (violet)
• Catégorie « événement » (marron)
• Catégorie « contrôle » (orange)

Classe de CE2 d'Emmanuelle Wilgenbus (Antony)

Note pédagogique

Bien penser, dès maintenant, à instaurer une règle d'alternance afin que ce ne soit pas toujours le même élève qui contrôle le clavier et la souris.

Tâche 3 : faire de petits exercices (20 minutes)

L'enseignant donne une série de petits exercices (qui reprennent, pour la plupart, ce qui a été fait avant sous forme de démonstration), que les élèves exécutent. Après chaque exercice, une rapide mise en commun permet de s'assurer que chacun sait faire l'exercice.

Exercice 1 – faire avancer le chat de 10 pas

Exercice 2 – faire avancer le chat de 20 pas

Plusieurs solutions possibles :

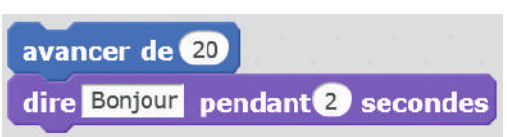
ou

On préférera la seconde solution, plus élégante et facile à lire.

Exercice 3 – remettre le chat au centre de la scène

Certains élèves vont sans doute trouver l'astuce…, mais pour la plupart, il faudra la leur montrer. Malgré tout, il est indispensable pour eux de voir cette instruction dès maintenant, car, à force de déplacer le chat, ils vont le faire sortir de l'écran et ne sauront pas comment le récupérer.

Exercice 4 – faire avancer le chat de 20 pas et lui faire dire « Bonjour »



Bien préciser que « dire » bonjour signifie pour nous « écrire » bonjour : on veut faire afficher une bulle à l'écran avec le texte « bonjour » dedans, on ne veut pas faire parler le chat !
NB : il est possible de faire parler le chat (lui faire émettre des notes de musique ou jouer un fichier son que l'on aura importé ou enregistré dans *Scratch*) ; nous le déconseillons fortement en classe.

Exercice 5 – répéter 3 fois : faire avancer le chat de 20 et lui faire dire « bonjour »

Inciter les élèves qui ne trouvent pas à chercher dans la catégorie « contrôle » (orange). Ils y trouveront une instruction qui ressemble (« répéter 10 fois ») et qu'il est facile de modifier en remplaçant 10 par 3. Cette boucle enserre les autres instructions. Tout ce qui se trouve à l'intérieur de la boucle est exécuté 3 fois.

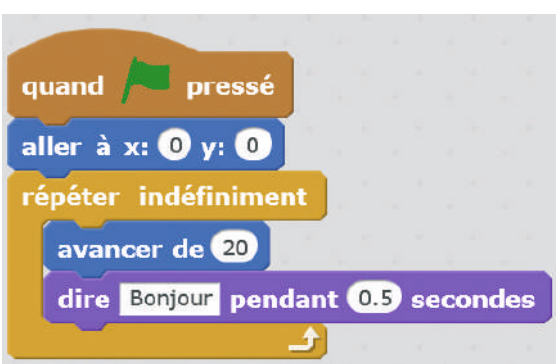
On voit le chat qui s'arrête 2 secondes entre chaque mouvement… Pour raccourcir cette pause, il suffit de raccourcir la durée pendant laquelle il dit « bonjour » (si on écrit « 0,5 » au lieu de 2, il ne s'arrêtera qu'1/2 seconde à chaque fois).

Exercice 6 – répéter indéfiniment : faire avancer le chat de 20 et lui faire dire « bonjour »

Cela se fait très simplement, sur le même modèle que l'exercice précédent, mais avec un autre type de boucle.

Exercice 7 – même chose quand on clique sur le drapeau vert

Il suffit a priori de rajouter l'instruction « quand drapeau vert pressé » (issue de la catégorie « événement »), mais cela est encore mieux si on demande au chat de repartir du centre de la scène.



Expliquer, à ce moment, le rôle du bouton rouge (situé à côté du drapeau vert). Un clic sur ce bouton rouge met fin à l'exécution du programme (qui, sinon, ne s'arrête jamais dans le cas présent).

Bilan et conclusion

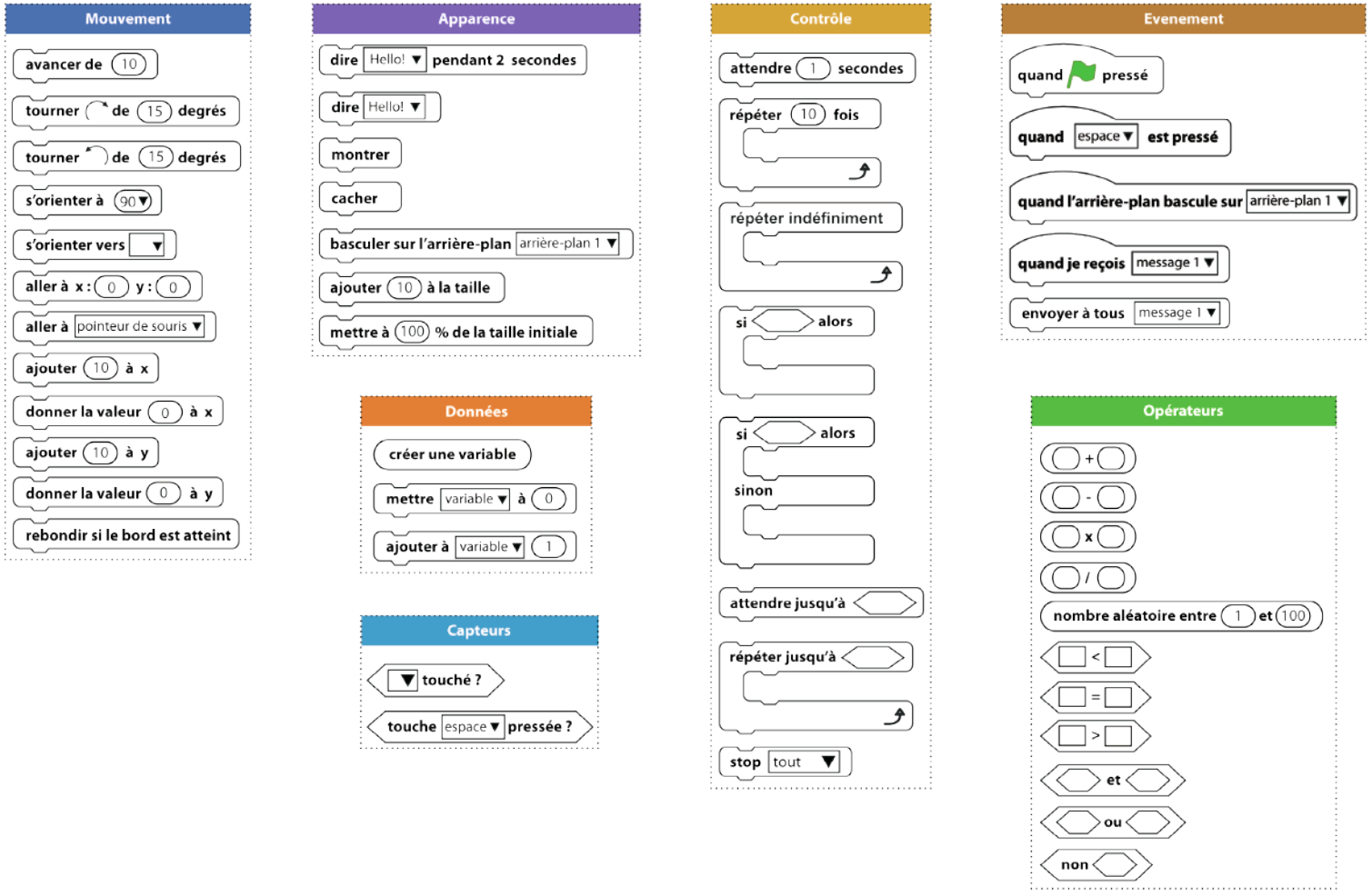
La classe synthétise collectivement ce qui a été appris au cours de cette séance, notamment en listant les commandes *Scratch* qui sont désormais connues de tous :

- *Avancer de 10*
- *Aller à (0,0)*
- *Dire « bonjour » pendant 2 secondes*
- *Répéter 10 fois*
- *Répéter indéfiniment*
- *Quand drapeau vert est pressé*

Il peut être très utile aux élèves de colorier les instructions *Scratch* au fur et à mesure qu'elles sont découvertes et comprises. Ainsi, après chaque nouvelle étape, on pourra constater les progrès du binôme ou de la classe entière.

La Fiche 32 « Quelques commandes utiles dans », page 249, peut être photocopiée pour chaque élève et agrandie pour la classe entière. Cette fiche sera enrichie plus tard, quand les élèves auront manipulé des tests (page 260), des capteurs (page 260) des variables (page 262), et des opérateurs (page 264).

Quelques commandes utiles en Scratch



Étape 2 – Planter le décor et sauvegarder son travail

Résumé	Les élèves apprennent à personnaliser la scène dans *Scratch* (lutin et arrière-plan), ainsi qu'à enregistrer leur travail pour le réutiliser plus tard. Ils discutent ensuite des différentes étapes qui leur permettront de réaliser leur jeu vidéo.
Notions (*cf. scénario conceptuel, page 213*)	Idem étape 1 (voir page 241)
Matériel	Idem étape 1 (voir page 241)

Notes pédagogiques

• L'objectif de ce guide pédagogique est l'apprentissage de l'informatique (ici, plus particulièrement, la programmation) : nous ne décrivons donc pas les éventuelles séances de prolongement possibles, comme celles que l'on peut faire en arts plastiques / TICE afin de dessiner un rover personnalisé ou un arrière-plan pour notre jeu vidéo. Nous proposons simplement, dans cette étape, d'importer des éléments que nous mettons à disposition des classes. Cela présente un double avantage : un gain de temps et une certaine homogénéité entre les programmes réalisés par les élèves, ce qui facilitera leur comparaison.

• Bien sûr, l'enseignant peut décider de consacrer une heure à faire dessiner ces éléments par les élèves. Attention, dans ce cas pour l'arrière-plan : il doit être relativement homogène car des éléments de décor (obstacles et ressources) seront ajoutés plus tard, sous la forme de nouveaux lutins.

Tâche 1 : changer le lutin (5 minutes)

L'enseignant explique qu'il est possible de supprimer le lutin « chat » et d'en créer un autre à la place, plus en phase avec notre projet de mission spatiale : un rover.

• Pour supprimer le chat, il faut cliquer (bouton droit) sur son icone, dans la zone des lutins, et choisir « supprimer ».

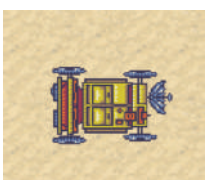
• Il y a 4 façons différentes de créer un nouveau lutin, accessibles depuis la barre d'outils « nouveau lutin » en bas à droite de la scène (nous mettons en gras la méthode que nous préconisons ici).

	*Choisir un lutin dans la bibliothèque* *Scratch* est livré avec une centaine de lutins prédéfinis dans la bibliothèque. Ces lutins peuvent être pratiques pour un projet d'élève, mais sont de styles très hétérogènes.
	*Dessiner un nouveau lutin* *Scratch* possède un outil intégré de dessin permettant aux élèves de créer « à la main » leur propre rover.
	***Importer le lutin depuis un fichier*** C'est l'option qui nous intéresse ici car l'objectif de ce projet n'est pas d'apprendre à dessiner à l'ordinateur, mais d'apprendre à programmer. Nous conseillons à l'enseignant d'avoir mis à disposition les fichiers nécessaires au projet (dont, ici, le rover) dans un répertoire facilement accessible par les élèves*. Pour le rover, il y a 2 possibilités : un rover « carré » et un autre plus allongé. C'est ce dernier qui sera utilisé dans les captures d'écran qui suivront.
	*Nouveau lutin depuis une webcam* Cet outil peut être très pratique pour des projets personnels (on peut ajouter son propre visage comme nouveau lutin) mais n'a pas d'intérêt dans le cadre de ce projet-ci.

Note pédagogique

Nous avons remarqué que, sur certaines machines, l'importation du lutin depuis un fichier fonctionne mal. Si l'importation échoue, il existe une façon très simple de remédier au problème : sauvegarder le travail en cours, éteindre *Scratch*, relancer *Scratch*… et recommencer l'import du fichier. Après cette petite manœuvre, ça marche !

Tâche 2 : changer l'arrière-plan (5 minutes)



De la même façon que précédemment, il est possible de changer l'arrière-plan de la scène, soit à partir d'une image issue de la bibliothèque, soit à partir d'un fichier fourni par l'utilisateur, soit encore en le dessinant soi-même.
Nous conseillons de choisir l'option « Importer l'arrière-plan depuis un fichier » de prendre le fichier *sol_martien.png* (dans le répertoire « scènes » des fichiers mis à disposition). Voici un aperçu du rover, sur le fond choisi.

Tâche 3 : enregistrer son programme Scratch (5 minutes)

L'enseignant explique qu'il faut enregistrer le programme actuel (même s'il ne contient pas encore grand-chose), pour éviter d'avoir à tout refaire à la prochaine étape.

* Tous les fichiers utiles sont disponibles sur le site Web du projet, voir page 349.

Cas n°1 : *Scratch* installé en local	Cas n°2 : utilisation de *Scratch* en ligne
L'enregistrement se fait en cliquant sur le menu « fichier » puis l'option « sauvegarder ». Il faut ensuite se déplacer dans le répertoire dédié au projet et à la classe (un raccourci depuis le bureau est, encore une fois, fortement conseillé), puis choisir un nom de fichier. Ce nom de fichier peut, par exemple, comporter les prénoms des élèves, de façon qu'ils puissent facilement retrouver leur propre programme plus tard. L'import se fera soit en double-cliquant directement sur le fichier sauvegardé (ce qui lance le logiciel *Scratch*), soit en lançant *Scratch* puis en cliquant sur le menu « fichier » et l'option « ouvrir ».	L'enregistrement se fait en cliquant sur le menu « fichier » puis l'option « télécharger dans votre ordinateur ». L'import se fera plus tard depuis le même menu, en cliquant sur l'option « importer depuis votre ordinateur ». Le module Mooc « Découvrir la programmation créative » de la formation class' Code enseigne comment créer un « studio » *Scratch* en ligne pour une classe (voir pages 354).

Bilan et conclusion

La classe revient sur ce qu'elle a appris à faire dans *Scratch* : importer un lutin ou un arrière-plan ; sauvegarder et reprendre son travail.

L'enseignant peut montrer à nouveau la démonstration du jeu « final », afin de faire expliciter par les élèves les tâches qui restent à faire. Par exemple :

- Piloter le rover à l'aide des flèches
- Importer d'autres lutins pour gérer les ressources et les obstacles
- Faire en sorte qu'on gagne un point quand on récolte une ressource et qu'on perde une vie quand on touche un obstacle
- Faire en sorte qu'une ressource disparaisse lorsqu'elle est ramassée et réapparaisse ailleurs sur la scène (à un endroit aléatoire)
- Faire en sorte que la tornade se promène au hasard sur la scène
- Faire en sorte que le jeu prenne fin lorsqu'on n'a plus de vie (avec « game over » qui apparaît et tout le reste qui disparaît).

On peut aussi imaginer d'autres activités :

- Introduire un compte à rebours pour pimenter le jeu (il faut récolter le plus de ressources en un temps donné)
- Personnaliser le jeu en dessinant ses propres lutins et son propre arrière-plan
- Etc.

Ces étapes seront reprises par la suite et découpées en tâches élémentaires quand cela se révèle nécessaire. Chaque binôme pourra avancer à son rythme, l'important étant d'arriver à un jeu jouable à la fin du projet.

Étape 3 – Piloter le rover

Résumé	Les élèves réalisent leur premier programme, leur permettant de piloter le rover à l'aide des flèches du clavier. Ils se familiarisent avec le système de coordonnées.
Notions (*cf. scénario conceptuel, page 213*)	Idem étape 1 (voir page 241)
Matériel	Idem étape 1 (voir page 241) Plus, pour chaque élève • une photocopie de la Fiche 33, page 258

Une fois que chaque binôme a réussi l'importation de son programme (ne contenant, pour l'instant, que le rover et l'arrière-plan), la classe revient sur la liste des étapes à réaliser pour programmer le jeu vidéo. La première chose à faire, c'est de piloter le rover. Le plus simple est de piloter le rover à l'aide des touches « flèches » du clavier.

Note pédagogique

Pour cette étape, les élèves auront encore besoin d'être guidés. Ensuite, ils auront acquis les automatismes leur permettant d'être bien plus autonomes, et chaque binôme avancera à son rythme.

Tâche 1 : faire avancer le rover vers la gauche (10 minutes)

Les élèves savent déjà comment faire avancer le rover vers la droite… il suffit de lui dire d'avancer, puisque, par défaut, il est déjà orienté vers la droite. Le faire avancer vers la gauche est une tâche un peu plus difficile, car les élèves doivent d'abord demander au rover de s'orienter vers la gauche, avant d'avancer.

Ils travaillent en autonomie et tâtonnent, l'enseignant passant régulièrement dans les groupes pour s'assurer que personne n'est bloqué. Il peut les guider en les incitant à chercher une instruction « s'orienter ».

Note pédagogique

Deux instructions de ce type sont disponibles :

– « s'orienter vers… » qui ne nous intéresse pas car la seule option disponible, accessible en cliquant sur la petite flèche, est « pointeur de souris » (le lutin, dans ce cas, s'oriente vers la position du pointeur de la souris).

– « s'orienter à… » qui est celle qui nous intéresse. Lorsque l'on clique sur le nombre présent dans l'instruction (en général, ce nombre par défaut est « 90 »), une bulle d'aide nous explique que l'angle 0° désigne le haut de l'écran ; 90° désigne la droite, etc.

Il faut donc choisir ici « s'orienter à -90 »

Finalement, le programme qui permet au lutin de se déplacer vers la gauche est :

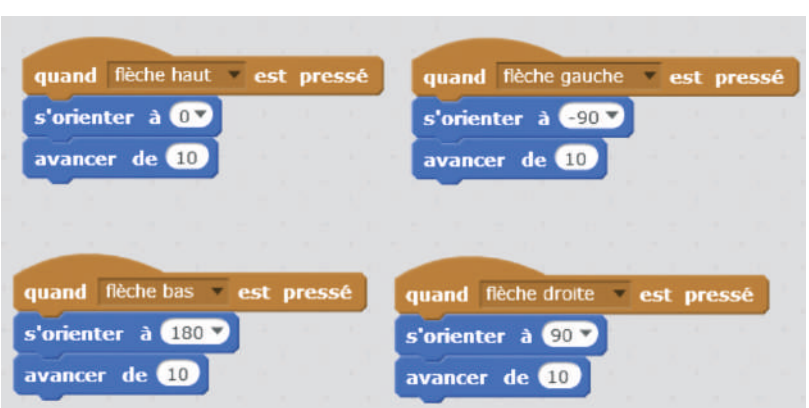
Tâche 2 : faire avancer le rover dans n'importe quelle direction (5 minutes)

Les élèves doivent maintenant être capables de faire avancer le rover dans n'importe quelle direction (droite, gauche, haut et bas), en reprenant exactement la même méthode que précédemment.
NB : désormais, on a besoin de l'instruction « s'orienter à 90 » pour lui dire d'aller à droite… car le lutin n'est plus orienté, par défaut, dans cette direction.

Tâche 3 : piloter le rover à l'aide des flèches (15 minutes)

Les élèves doivent maintenant faire en sorte que le rover se déplace quand ils pressent les flèches du clavier. Ils cherchent en autonomie comment faire. Certains se rappellent la commande « quand drapeau vert pressé » vue lors de la toute première séance *Scratch*. C'est un événement qui permet de déclencher une action. Ici aussi, on cherche un événement : l'action se déclenche quand une touche du clavier est pressée.
La commande « quand (espace) est pressé » nous intéresse, sauf qu'il faut remplacer « espace » par une des flèches (flèche droite pour aller à droite). Cela se fait de la même manière que précédemment :

Finalement, la zone du programme du rover comporte 4 sous-programmes, chacun décrivant le déplacement dans une direction particulière. Voici à quoi peut ressembler le programme :



Notes pédagogiques

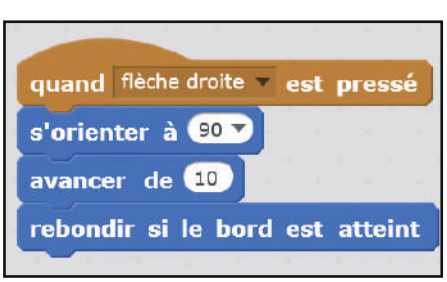
- On remarque ici qu'on peut faire coexister plusieurs sous-programmes dans le même programme. Chacun s'exécute lorsque l'événement déclencheur (ici, presser une touche du clavier) est détecté.
- Certains élèves paniquent parfois en pensant que tout leur programme a disparu suite à une fausse manœuvre. En général, cela n'est pas le cas (le programme

n'est pas effacé). Ils ont simplement cliqué sur la scène (qui a sa propre zone de programme… mais vide puisqu'on n'y a rien mis pour le moment) au lieu du lutin. Parfois, ils ont bien sélectionné le lutin, mais ont cliqué sur l'onglet « costumes » au lieu de l'onglet « scripts ». Il suffit de retourner sur le lutin et sur l'onglet « scripts » pour voir réafficher le fameux programme !

Classe de CM1 de Caroline Vinel (Paris)

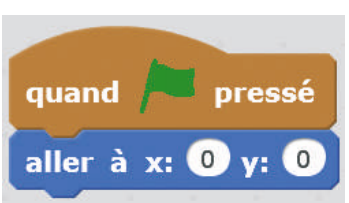
Tâche 4 : rebondir sur les bords (5 minutes)

Les élèves cherchent comment faire en sorte que le rover rebondisse sur les bords. Par exemple, si on le dirige vers la droite et qu'il atteint l'extrémité droite de l'écran, le rover doit rebondir afin de ne pas sortir de l'écran.
Cela se fait très simplement en ajoutant l'instruction « rebondir si le bord est atteint » en bas de chacun des sous-programmes faits précédemment. Par exemple :



Tâche 5 : initialiser la position du rover (5 minutes)

L'enseignant rappelle que, lorsqu'on lance le programme (drapeau vert), le rover doit se situer au centre de l'écran. Les élèves reprennent sans difficulté les instructions qu'ils avaient vues lors de la première séance *Scratch*.



Notes pédagogiques

• Le programme peut maintenant être exécuté en cliquant sur le drapeau vert. Si on le souhaite, on peut faire disparaître les programmes pendant l'exécution, en cliquant sur le bouton « plein écran » en haut à gauche de la scène.
• Ne pas oublier, chaque fois, de sauvegarder le travail réalisé ! (*cf.* page 251)

Tâche 6 : comprendre les coordonnées X et Y du rover (20 minutes)

La tâche précédente a mis en évidence les coordonnées X et Y du rover, à travers l'instruction « aller à (X = 0, Y = 0) ». Les étapes suivantes (ressources, pièges…) nécessiteront de manipuler ces coordonnées ; il importe donc de comprendre comment cela fonctionne.
L'enseignant demande aux élèves de visualiser les coordonnées X et Y qui s'affichent en bas à droite de la scène. On remarque que les coordonnées affichées changent en fonction de la position de la souris.

• Que valent X et Y quand la souris est au centre de la scène ? (réponse : X = 0, Y = 0)
• Et quand la souris est tout à droite ? (réponse : X = 240, Y peut prendre n'importe quelle valeur selon la position de la souris)
• Et quand la souris est tout à gauche ? (X = -240)
• Et quand la souris est tout en haut ? (Y = 180) ou tout en bas ? (Y = -180)
La classe conclut collectivement que X indique la position selon l'axe horizontal (axe imaginaire, non tracé) et Y indique la position selon l'axe vertical (lui aussi imaginaire).
Les élèves peuvent remarquer que, dans la catégorie « mouvement » de la palette d'instructions, de nombreuses instructions font intervenir les variables X et Y. Ici, il ne s'agit plus de la position de la souris, mais de celle du lutin sélectionné. Le rover possède son propre jeu de variables X et Y.
L'enseignant peut distribuer la Fiche 33 à chaque élève et lui proposer de petits exercices :
• Place le lutin sur la scène, aux coordonnées X = -100, Y = 100
• Que se passe-t-il si on ajoute 50 à X ? Où est désormais le lutin ?
• Que se passe-t-il si maintenant on met Y à 0 ? Où est le lutin ?

Notes pédagogiques

• Pour faciliter la compréhension de ces coordonnées, l'enseignant peut faire un parallèle avec ce que les élèves ont déjà vu en géographie : latitude/longitude. Ici, l'unité n'est pas le degré (on ne raisonne pas en angle), mais le pixel. De même, dans un jeu de bataille navale, on repère la position des bateaux par 2 coordonnées (une lettre et un chiffre). Au choix de l'unité ou du symbole près, il s'agit exactement de la même chose : repérer la position d'un point sur une surface, ce qui nécessite 2 coordonnées car une surface est un espace à 2 dimensions.
• De même, il peut être utile à certains élèves de se raccrocher à des situations concrètes au sujet des nombres négatifs. Les exemples ne manquent pas, qu'il s'agisse des dates ou de la mesure de la température (que signifie « -10 °C » : est-ce plus chaud ou plus froid que « 0 °C ». Et « -20 °C », est-ce plus chaud ou plus froid que « -10 °C » ?).

Conclusion et traces écrites

À la fin de cette séance, il importe de faire le point sur les nouvelles commandes *Scratch* que les élèves ont appris à utiliser :

- S'orienter à (90)
- Quand (espace) est pressé
- Quand (drapeau vert) est pressé
- Aller à (X =…, Y = …)

Les élèves colorient ces commandes sur la Fiche 32 (page 249) qu'ils avaient déjà utilisée.

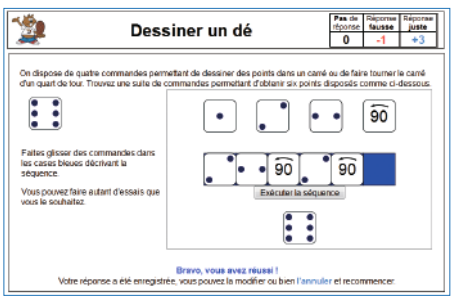
Par ailleurs, cette séance offre l'occasion de prendre du recul par rapport aux activités de programmation et de revenir sur certains concepts :

- *Un programme est un algorithme exprimé dans un langage particulier, appelé « langage de programmation », compréhensible à la fois par la machine et par l'être humain.*
- *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).*
- *L'ordinateur ne fait qu'exécuter les instructions qu'on lui donne : ni plus ni moins.*
- *La position d'un élément à l'écran est repérée grâce à 2 coordonnées. Dans Scratch, on les appelle X et Y. X varie entre -240 et 240 ; Y varie entre -180 et 180.*

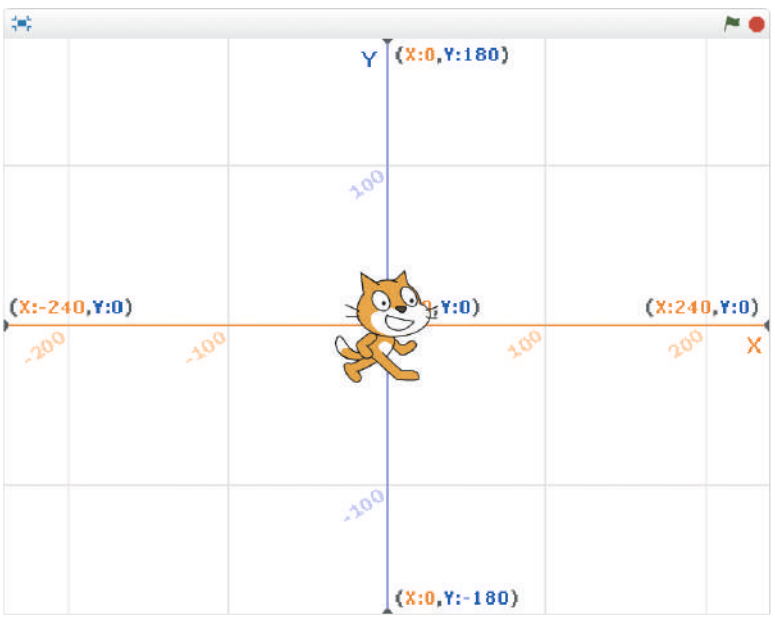
Les élèves notent ces conclusions dans leur cahier de sciences. L'enseignant, quant à lui, met l'affiche « qu'est-ce que l'informatique ? » à jour.

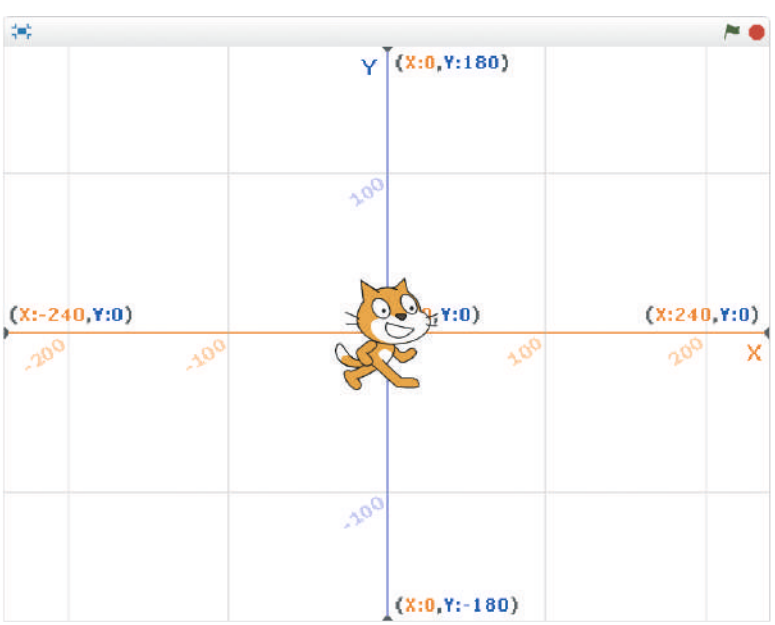
Exercice en ligne

L'orientation du lutin peut être retravaillée à l'occasion d'un exercice en ligne sur le site Web du projet (voir page 349). Cet exercice n'utilise pas *Scratch*.



FICHE 33
Les coordonnées X et Y dans *Scratch*





Étape 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 (*cf. scénario conceptuel, page 213*)	« 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	Idem séances précédentes
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.

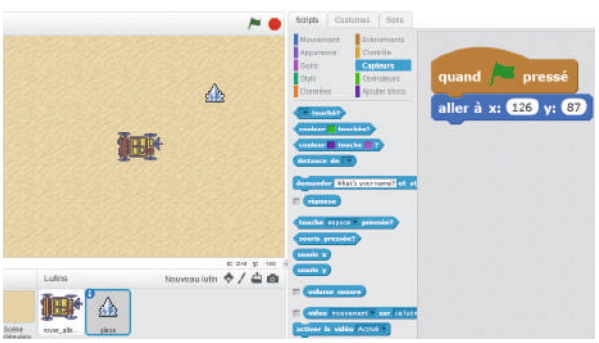
• À partir de maintenant, il est illusoire de chercher à ce que tous les élèves avancent au même rythme (sinon, les groupes ayant le plus de facilité vont très vite s'ennuyer et se dissiper). Le découpage en étapes et en tâches permet de faciliter la gestion de classe par l'enseignant : chacun, où qu'il 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 a sélectionné l'instruction, 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 :

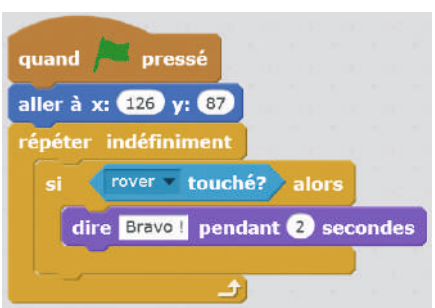
Malheureusement, lorsqu'on lance le programme et qu'on dirige le rover vers la glace, cela ne fonctionne pas. Pourquoi ? La classe peut discuter collectivement de ce que fait ce programme, pas à pas :

- 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 :

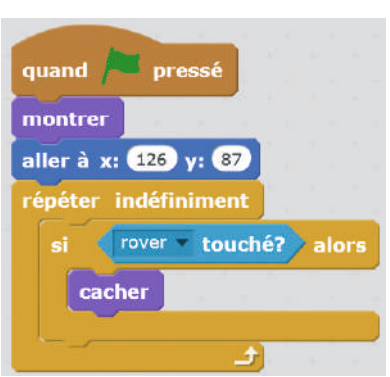


Tâche 3 : faire disparaître la ressource quand elle est touchée (10 minutes)

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 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 ».

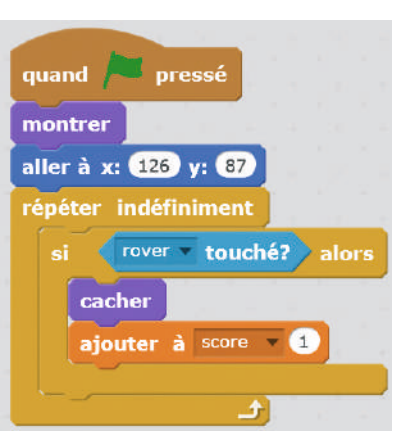
Ici, il s'agit de compter un score. Cette variable pourra sans doute être manipulée par plusieurs lutins (les différentes ressources…) : on doit donc la rendre accessible à tous les lutins.
L'enseignant fait remarquer aux élèves qu'ils ont déjà manipulé des variables dans les séances précédentes (l'abscisse 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 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.
• Une activité débranchée permet de se familiariser avec les manipulations de variable (*cf.* page 268).

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 disparaît 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

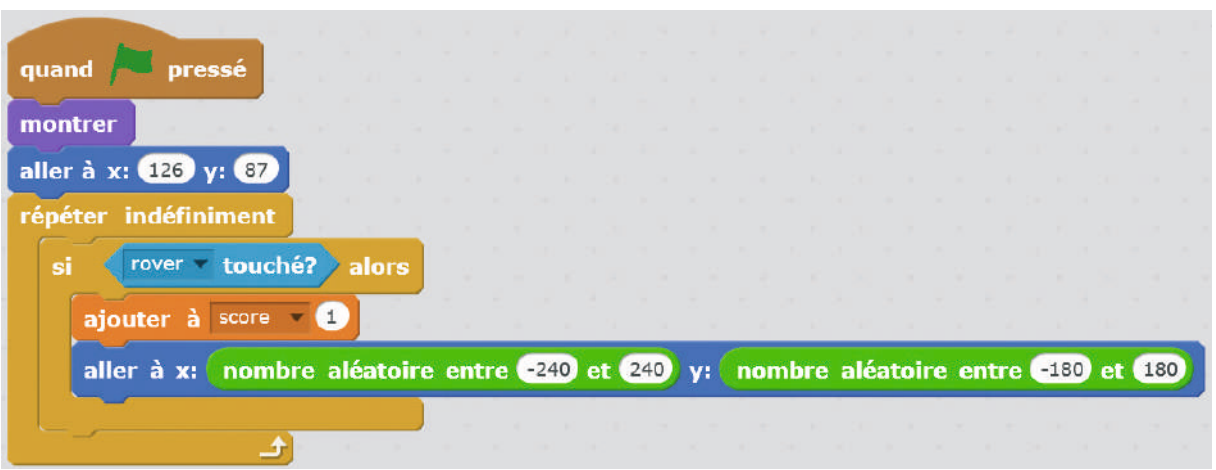
• A priori, cette initialisation peut être faite dans le programme de n'importe quel lutin : l'important étant qu'elle soit faite une fois, et une seule. Mais le score est une variable qui sera sans doute manipulée par d'autres lutins (la végétation, quand on l'aura importée). Il n'y a pas de raison 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 !

Initialisation de la position du rover et du score, 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 de consolider en tout autonomie les différentes notions abordées précédemment.

Conclusion et traces écrites

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

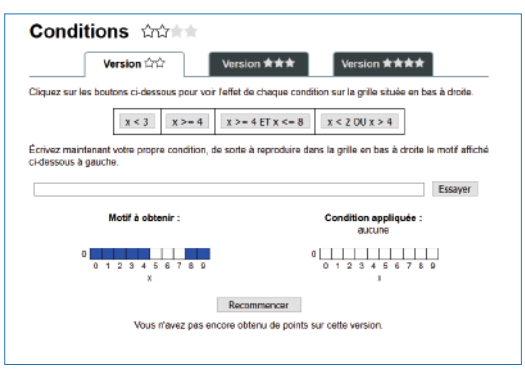
• *On peut créer différentes variables dans un programme. Il est préférable d'initialiser chaque variable, c'est-à-dire de lui donner une valeur au lancement du programme.*

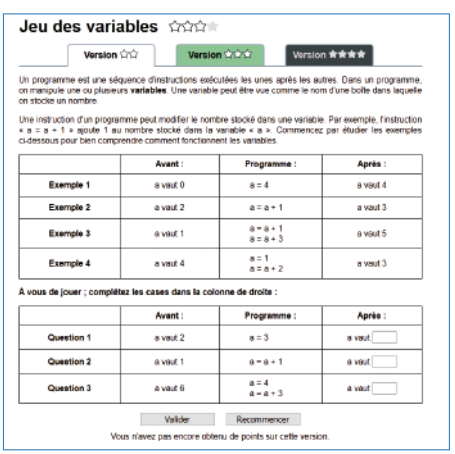
• *Un programme informatique peut générer des nombres aléatoires. Ainsi, chaque exécution peut donner un résultat différent.*

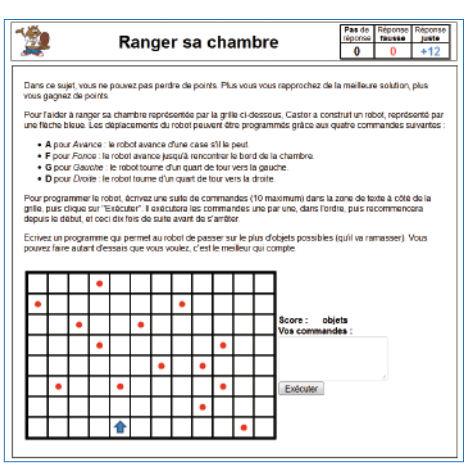
Les élèves notent ces conclusions dans leur cahier de sciences. L'enseignant, quant à lui, met l'affiche « qu'est-ce que l'informatique ? » à jour.

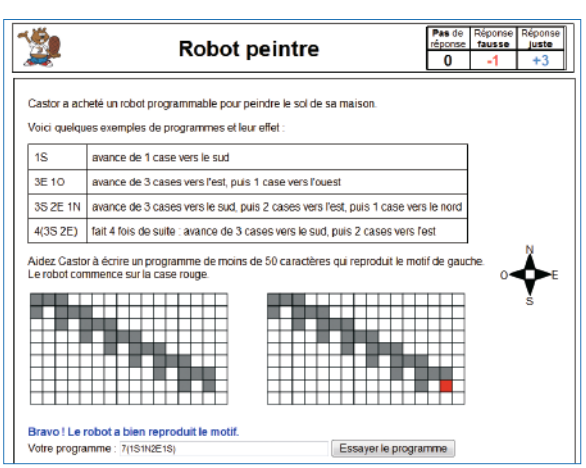
Exercices en ligne

Les variables, instructions conditionnelles et boucles peuvent être retravaillées à l'occasion de plusieurs exercices en ligne sur le site Web du projet (voir page 349). Ces exercices n'utilisent pas *Scratch*.









Étape 5 – Activités branchées et débranchées pour mieux s'approprier certains concepts algorithmiques

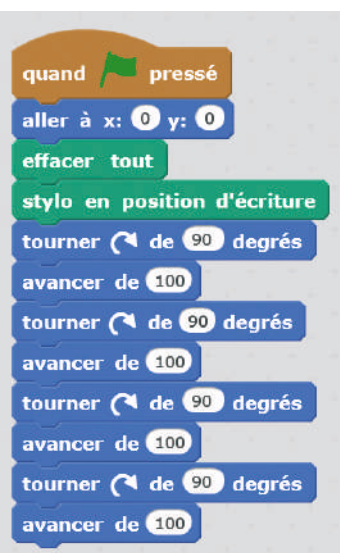
Résumé	En parallèle de leur activité de programmation, les élèves revoient et approfondissent certains concepts algorithmiques abordés lors de l'étape 4 : variables, tests, boucles, opérateurs logiques, jusqu'à la notion d'algorithme elle-même.
Notions (*cf. scénario conceptuel, page 204*)	« Algorithmes » • 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. • On peut utiliser des connecteurs logiques comme ET, OU, NON pour fabriquer des expressions logiques. • Parfois, on se contente d'un algorithme qui ne donne pas une solution parfaite. « 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. « Machines » • Pour certaines tâches, les ordinateurs sont bien plus rapides que les hommes.
Matériel	Pour l'activité 1 (évaluation formative sur le concept de boucle) • Salle informatique Pour l'activité 2 (jeu de cartes pour s'approprier la notion de variable) • Pour chaque groupe de 8 élèves : – 4 ardoises, 4 feutres, 4 chiffons – 1 jeu de cartes massicotées à partir de la Fiche 34, page 278 et de la Fiche 35, page 279 (de préférence sur papier Canson pour une meilleure longévité). Imprimer les cartes sur du papier coloré (une couleur par jeu) pour pouvoir facilement reséparer les jeux mélangés. – Un dé à jouer à 6 faces Pour l'activité 3 (jeu pour travailler les opérateurs logiques) • Pour chaque élève : – Fiche 36, page 280 • Pour chaque groupe de 4 élèves : – Fiche 37, page 281 Pour l'activité 4 (problème du voyageur de commerce) • (Facultatif) pour chaque groupe : – une planche de 20 cm x 20 cm x 2 cm (de préférence) sur laquelle on a planté, bien droit, une vingtaine de clous, de façon aussi aléatoire que possible – une ficelle d'environ 2 mètres de longueur, nouée à l'un des clous – un stylo feutre • Pour chaque élève : – Fiche 38, page 282
Lexique	Boucle, variable, test, condition, expression logique

Avant-propos : quand faire ces activités ?

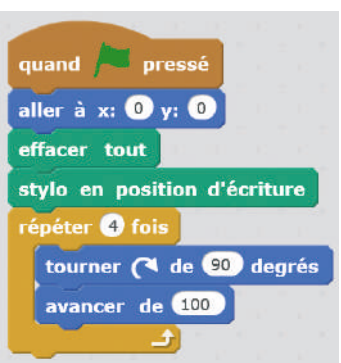
Cette étape est différente des autres au sens où elle ne contient pas de tâches nécessaires à la programmation du jeu vidéo, mais consiste en une série d'activités (débranchées, pour la plupart) qui permettent de revenir sur certains concepts algorithmiques utilisés notamment dans l'activité de programmation.
Ces activités sont complètement indépendantes les unes des autres et optionnelles : on peut parfaitement poursuivre le projet sans les réaliser.
Afin de ne pas casser la dynamique du projet en cours (programmation du jeu vidéo dans *Scratch*), nous conseillons de mener les activités débranchées proposées ici sur un autre temps que celui dédié à la programmation qui doit suivre son cours. L'idéal est de le faire sur le temps accordé aux mathématiques ou au français.
L'activité 1, branchée, peut quant à elle facilement s'intégrer à une séance de programmation (il suffit de prendre 10 minutes avant de se replonger dans le projet en cours).

Activité 1 : évaluation formative sur le concept de boucle (branchée, 10 à 20 minutes)

Cette activité est fortement conseillée après l'étape 4, au cours de laquelle les élèves ont manipulé plusieurs fois des boucles dans *Scratch*. Ce concept de boucle peut être consolidé grâce à une série de petits exercices d'évaluation formative. Ceux-ci permettront également d'explorer plus avant les boucles disponibles en *Scratch* (car toutes celles que nous utilisons dans notre programme sont du type « répéter indéfiniment », mais il y en a d'autres !).
Tout comme cela est proposé au cycle 2 avec *Scratch Junior* (*cf.* Séance 3, page 155), on peut proposer aux élèves de simplifier l'écriture d'un programme en utilisant des boucles. Le programme ci-dessous, à gauche, fait dessiner un carré par le lutin. Le programme de droite arrive au même résultat en utilisant une boucle. Suivant le niveau des élèves, on peut soit leur demander de réaliser le programme de droite par eux-mêmes, soit 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 celui de gauche.



Le lutin dessine un carré de 100 pixels de côté



Ce programme fait exactement la même chose, avec une boucle

Nous conseillons à l'enseignant de répéter ce type d'exercice autant de fois que nécessaire pour que l'utilisation des boucles soit bien comprise et devienne naturelle.

Activité 2 : un jeu de cartes pour s'approprier la notion de variable (débranchée, 1 heure)

Cette activité a pour objectif d'aborder la notion de variable et propose pour cela d'initier les élèves à un jeu de cartes. Il est intéressant de donner aux élèves l'occasion de jouer plusieurs fois à ce jeu, car ils deviennent alors capables d'élaborer des stratégies complexes. Le travail peut être réalisé sur des temps de calcul mental (manipulation des scores), de français (travail sur la signification des cartes), ou en accompagnement personnalisé par petits groupes de 8 ou 16 élèves. Si le travail est effectué en classe entière, ne pas faire l'économie d'une discussion collective sur la signification des cartes, avant la mise en œuvre du jeu. Si le travail est effectué en demi-groupes (ce qui est bien plus confortable), préciser la signification des cartes qui posent problème lorsque la difficulté se présente.
Les variables manipulées dans le jeu sont les scores des joueurs, et certaines cartes affectent ces scores. Dans un premier temps de découverte du jeu, seules des cartes proposant des manipulations simples des scores sont intégrées au jeu. Puis des cartes plus complexes sont introduites, incluant des manipulations des scores en fonction de certaines conditions.

Situation déclenchante – Présentation du jeu

Les soirées sont longues à la base installée sur la planète lointaine. Les explorateurs, une fois accomplies leurs missions de travail, se détendent en faisant du sport en salle et en jouant à des jeux de société. Leur jeu préféré est un jeu de cartes auquel les élèves vont jouer. Le jeu se joue à 4 équipes (idéalement des binômes) nommées A, B, C, D. On organise donc des tablées de 8 élèves. Le but de chaque équipe est d'accumuler le plus de points possible, de la façon suivante :
– Les équipes A, B, C et D démarrent le jeu avec un score initial de 1 point, inscrit sur une ardoise.
– Chaque équipe reçoit 4 cartes distribuées au hasard, et les cartes restantes sont placées en pile (la pioche), face cachée.
– Les élèves prennent connaissance de leurs cartes, sans les montrer aux élèves des équipes concurrentes.
– À tour de rôle, les équipes se mettent d'accord pour choisir une de leurs 4 cartes. Ils la lisent à voix haute, la posent sur la table face visible et obéissent aux instructions qu'elle donne. Certaines de ces instructions affectent le score d'une ou de plusieurs équipes. Dans ce cas, les scores sont mis à jour sur les ardoises. L'équipe qui vient de jouer retire une carte dans la pile.
– Lorsque toutes les cartes ont été jouées, l'équipe qui a le plus de points est déclarée gagnante.
L'enseignant donne un exemple au tableau : il dessine la position des équipes A, B, C et D autour d'une table en vue du dessus, avec les scores à la valeur 1. Puis il lit à voix haute (ou fait lire par un élève) une carte jouée par l'équipe A… et demande à la classe comment les scores vont changer sous l'effet de cette carte. Les changements des scores sont effectués. L'enseignant lit ensuite (ou fait lire) une carte jouée par l'équipe B… et ainsi de suite. Il continue aussi longtemps que nécessaire pour que toute la classe estime avoir compris le jeu.
Alternative : on peut préférer organiser le jeu sur un mode coopératif. Dans ce cas, les 4 équipes d'une table coopèrent, sans s'informer mutuellement sur la nature des cartes qu'elles ont en main, pour que la somme de leurs scores finaux soit la plus élevée possible.

Jeu avec les cartes 1 à 24

Les élèves jouent au jeu avec seulement les cartes 1 à 24 de la Fiche 34. Parmi ces cartes, celles qui affectent les scores donnent explicitement le nom des scores à changer (A, B, C et/ou D) et les changements de score ne dépendent pas de conditions.

Lors de la mise en commun, le professeur demande aux élèves si leur score a toujours eu la même valeur au cours du jeu ou si cette valeur a changé. Il introduit l'adjectif « variable », pas encore dans son sens informatique. La classe discute du rôle de l'ardoise (elle permet de noter les valeurs actuelles des scores et de les modifier facilement).

Il peut ensuite, à condition que les élèves aient déjà une première pratique de *Scratch* :

– leur demander de proposer un nom pour les scores des 4 équipes d'une table, par exemple score A, score B, score C et score D ;

– leur montrer comment initialiser ces 4 variables à la valeur 1 ;

– introduire le terme « variable » dans son sens informatique : une variable est un espace de mémoire dans lequel on peut stocker une valeur, pour la réutiliser ou la modifier plus tard ;

– commencer un travail de traduction en *Scratch* des cartes les plus simples (cartes 1 à 8 pour commencer, et davantage si la classe se prend au jeu, voir paragraphe « Traduction des cartes en langage *Scratch* », plus bas). Les cartes traduites sont dorénavant remplacées dans le jeu par leur version en langage *Scratch*.

Jeu avec les cartes 1 à 36

Les élèves rejouent au jeu en ajoutant les cartes 25 à 36 de la Fiche 35. Parmi les nouvelles cartes, certaines désignent des scores à changer, relativement à la position du joueur (score du joueur, score de son voisin de droite ou de son voisin d'en face par exemple). D'autres introduisent des conditions (si votre score… alors…).

Le travail de traduction en *Scratch* des cartes se poursuit progressivement, toujours en remplaçant les cartes au fur et à mesure de leur traduction.

Jeu avec les cartes 1 à 48

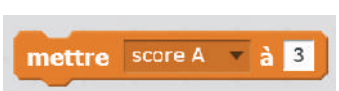
Les élèves rejouent au jeu en ajoutant les cartes 37 à 48 de la Fiche 35. Parmi ces nouvelles cartes, deux abordent les tirages aléatoires. Les 8 dernières cartes sont à compléter par les élèves.

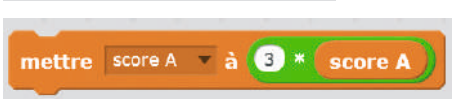
La traduction des cartes se poursuit, sans chercher l'exhaustivité.

Traduction des cartes en langage Scratch

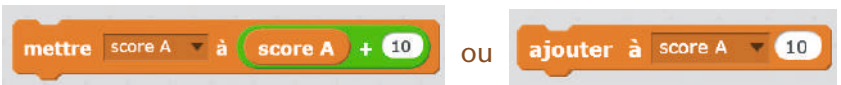
Selon l'avancement du travail avec *Scratch*, certaines cartes sont traduites du français au langage *Scratch*, progressivement (cartes 1 à 8 pour commencer).

Par exemple :

- la carte n°1 s'écrit 

- la carte n°5 s'écrit 

- la carte n°9 peut s'écrire de 2 façons différentes :



La traduction est répartie entre les différents groupes d'élèves et discutée. Attention, certaines traductions sont délicates (surtout pour les cartes à partir du numéro 25), d'autres sont impossibles (soit que les cartes n'affectent pas les scores, soit que la stratégie des joueurs dépende du contexte : valeur de leur score, et des scores des autres équipes, cartes restant dans leur main).
Certaines cartes, comme par exemple la n° 32, nécessitent de créer une variable supplémentaire pour stocker une valeur de façon temporaire.

Conclusion et mise en commun

Les élèves effectuent collectivement la synthèse de ce que le jeu de cartes leur a enseigné :
• *Au cours de nombreux jeux, les scores des joueurs changent de valeur (on dit qu'ils sont variables). Ils sont mémorisés à chaque étape du jeu. De même, la position des joueurs sur un plateau de jeu, le nombre de pions qu'ils possèdent, etc. sont mémorisés et varient.*
• *Dans un langage de programmation, une variable permet de mémoriser une valeur qui peut varier au cours de l'exécution du programme.*

Prolongement

Les élèves recherchent différents contextes dans lesquels les ordinateurs utilisés dans la vie courante utilisent des variables, pour stocker des données :
– l'heure affichée sur une montre digitale ;
– la vitesse de déplacement d'une voiture affichée sur un écran numérique ;
– le solde d'un compte bancaire.
Ils recherchent à quelles occasions ces variables changent de valeur ou sont utilisées.
Par exemple, l'heure de la montre change toutes les secondes, et à une certaine heure une alarme se déclenche.
Le solde d'un compte bancaire change à chaque fois qu'on effectue une dépense ou un encaissement.

Activité 3 : un jeu de cartes pour travailler les opérateurs logiques (débranchée, 1 heure)

Au cours de l'étape, les élèves ont utilisé des tests (si le rover récolte une ressource, alors le score augmente). L'enseignant explique à la classe que le groupe de mots « Le rover récolte une ressource » est une « expression » qui peut être soit vraie, soit fausse. Une telle expression s'appelle une « condition ».
L'activité proposée ici se déroule en 2 temps :
• Dans un premier temps, les élèves se familiarisent avec les expressions logiques en analysant une scène (Fiche 36).
• Dans un second temps, à l'aide de vignettes présentant des conditions et des connecteurs logiques, ils expriment la condition qui doit être remplie pour que l'alarme de la station spatiale se déclenche (Fiche 37).

Exercice de logique (individuel)

Chaque élève reçoit la Fiche 36 et doit indiquer, dans chaque situation, si l'expression est vraie ou fausse, ou encore si rien ne lui permet de le savoir.

Les réponses sont :
• Expression 1 : VRAIE (c'est assez évident !)
• Expression 2 : FAUSSE (là aussi !)
• Expression 3 : VRAIE (une porte est toujours, soit dans la position ouverte, soit dans la position fermée).
• Expression 4 : FAUSSE (une porte ne peut pas être ouverte et fermée en même temps)
• Expression 5 : on ne peut pas savoir (rien ne nous renseigne sur l'état de la porte, à moins que nous ne puissions la voir en ce moment)
• Expression 6a : VRAIE (les 2 conditions sont réalisées en même temps)
• Expression 6b : VRAIE (il suffit que l'une des conditions soit vérifiée, ce qui est le cas)
• Expression 7a : FAUSSE (la seconde condition n'est pas réalisée)
• Expression 7b : VRAIE (il suffit que l'une des conditions soit vérifiée, ce qui est le cas)

Les réponses des élèves sont comparées pour les différentes expressions de la fiche documentaire. L'enseignant veille à ce que la classe atteigne un consensus pour chacune d'entre elles. Il est probable que les expressions comportant un « OU » seront plus difficiles à évaluer pour les élèves.

• Pour que l'expression « A ou B » soit vraie, il suffit que l'une des deux sous-expressions A et B soit vraie. Il est possible, mais pas nécessaire, que A et B soient vraies à la fois.
• Pour que l'expression « A et B » soit vraie, il faut qu'à la fois A et B soient vraies.

Au besoin, la classe invente de nouvelles expressions simples pour manipuler ces conditions logiques et teste si ces expressions sont vraies ou fausses sur l'une et l'autre des illustrations de la Fiche 36 (ou sur des situations quotidiennes de l'école). Si cela est plus simple pour les élèves, présenter les expressions à l'intérieur d'un test plutôt que de façon isolée. Par exemple : « La cloche de l'école sonne si c'est l'heure de la récréation ou si c'est l'heure de la reprise ou si c'est l'heure de la fin des cours. »

Manipuler les expressions logiques : programmer l'alarme de la base

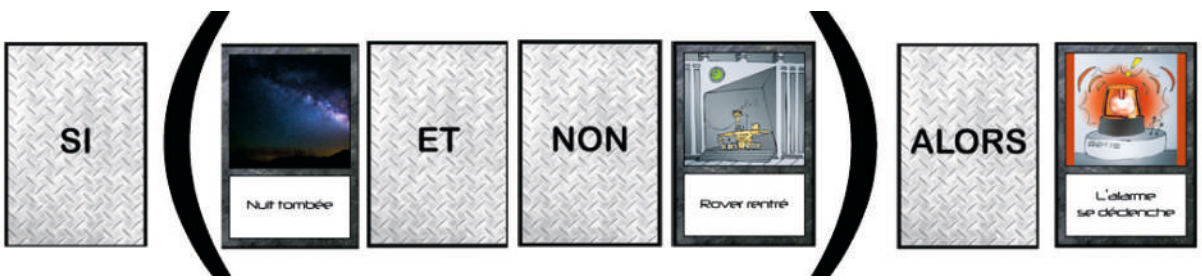
Une fois cette petite gymnastique acquise par les élèves, l'enseignant leur distribue la Fiche 37. Les élèves sont répartis en petits groupes (maximum 4 élèves par groupe). Cette fiche permet de réinvestir les notions précédentes dans le contexte de notre scénario, à savoir l'exploration de la planète. La consigne à donner à chaque groupe est la suivante : *découpez les vignettes puis combinez-les de façon à former des expressions logiques. On souhaite décrire la condition qui permet de déclencher l'alarme de notre station spatiale.*

Par exemple : l'alarme se déclenche :
• SI « le sas de sécurité est ouvert » ;
• OU si « la nuit tombe » ET « le rover n'est pas présent dans la base » ;
• OU si « le niveau d'oxygène est critique » ET « la base est occupée » ;
• OU si « l'énergie est basse » ET « la base est occupée » ;
• OU si « le groupe électrogène NE fonctionne PAS ».
• Etc.

Dans un premier temps, l'enseignant veille à ce que toutes les vignettes soient bien comprises, qu'il s'agisse des conditions (cartes présentant des dessins) ou des connecteurs logiques (SI, ALORS, ET, OU, NON). Le « NON » est nouveau, et nécessite une explication. Le fait que le rover ne soit pas présent dans la base s'écrit : « NON (le rover n'est pas présent dans la base) ». Il peut être nécessaire de réfléchir collectivement à quelques cas simples, d'abord à l'oral, puis à l'aide des vignettes. Dès que le principe est compris des élèves, on peut les laisser en autonomie pour trouver d'autres conditions et les écrire.

La première condition s'écrit :

La seconde s'écrit :



Notes pédagogiques

• Les parenthèses permettent de rendre les expressions plus facilement lisibles, et font partie intégrante de la syntaxe. Déplacer des parenthèses peut changer le sens d'une expression ! Les élèves peuvent placer leurs cartes sur une feuille blanche et dessiner les parenthèses sur la feuille.

• En fonction de l'aisance des élèves, on pourra leur demander soit d'écrire chaque condition séparément, soit d'écrire une seule expression qui rassemble toutes les conditions faisant que l'alarme se déclenche (toutes ces conditions sont connectées par des « OU ». Dans ce cas, il peut être nécessaire d'imprimer plusieurs copies de la Fiche 37 pour que chaque groupe puisse disposer de davantage de vignettes (en particulier, les connecteurs logiques).

Pour chacune des conditions nécessaires au déclenchement de l'alarme, l'enseignant veille à ce que les différentes propositions des élèves soient présentées et discutées.
Collectivement, la classe construit une expression unique qui les regroupe toutes. Pour plus de visibilité, ne pas hésiter à multiplier les parenthèses et à écrire l'expression sur plusieurs lignes :

La classe synthétise collectivement ce qui a été appris au cours de cette séance :

• *Dans un algorithme, on peut utiliser des tests qui disent quelle instruction effectuer quand une condition est vérifiée ou non.*

• *Une condition est une expression qui peut être soit vraie, soit fausse (mais pas les deux).*

• *On peut utiliser des connecteurs logiques comme ET, OU, NON pour fabriquer des expressions logiques.*

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 ? » démarrée en début de projet en recopiant ce que la classe a appris sur la notion de logique au cours de cette séance.

Activité 4 : comprendre qu'un algorithme n'est pas toujours parfait – le jeu du voyageur de commerce (débranchée, 1 heure)

Notes pédagogiques

• Cette activité, optionnelle, cible davantage les élèves plus âgés (6e et au-delà). Elle consiste à approfondir la notion d'algorithme à travers un problème simple : trouver le plus court chemin permettant de passer par différents endroits. Elle a pour but de montrer que, parfois, un problème ne peut pas être résolu, même s'il existe un algorithme qui, a priori, est censé fonctionner. Il faut alors se contenter d'une solution approchée, imparfaite mais suffisante.

• Ici, il existe un algorithme simple pour résoudre le problème (essayer tous les chemins et choisir le plus court), mais cet algorithme ne peut pas être mis en œuvre dans la pratique, car le nombre de chemins à tester devient très rapidement gigantesque si le nombre de points par lesquels il faut passer augmente.

Situation déclenchante

Au cours des séances précédentes, les élèves ont complété leur programme pour y ajouter un défi : le rover doit explorer la carte pour récupérer le plus de ressources possible.

L'enseignant explique que le carburant est limité et précieux et qu'il faut l'économiser au maximum. Ainsi, on se place dans un problème « classique » d'optimisation : « *On connaît à l'avance le nombre et l'emplacement des ressources. On doit trouver un chemin qui passe par toutes ces ressources et revienne au point de départ… et qui soit le plus court possible.* »

L'enseignant demande aux élèves s'il existe une méthode (un algorithme) qui permet de donner la solution à ce problème. La discussion collective montre que cette question peut être décomposée en 2 questions plus précises :

• *Le problème a-t-il une solution ?* Parmi les chemins qui passent par toutes les ressources, est-ce qu'il y a un chemin plus court que les autres (éventuellement, ex aequo) ? La réponse à cette question est « oui » : il existe plusieurs chemins possibles, il y a donc forcément un (ou plusieurs) chemin(s) plus court(s) que les autres.

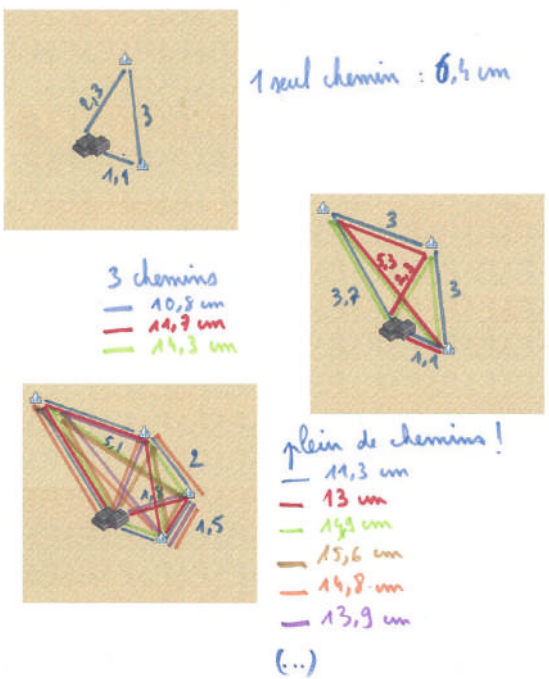
• *Existe-t-il une méthode qui nous permette de trouver, à coup sûr, le chemin le plus court possible ?* La classe peut proposer une méthode pour trouver ce chemin : il suffit de tester tous les chemins possibles et de mesurer leur longueur, pour sélectionner le plus court.

En fonction du matériel disponible, l'enseignant peut proposer soit 1 seule activité (basée sur la fiche documentaire), soit 2 activités (d'abord, la fiche documentaire, puis l'expérience à l'aide des planches cloutées)

Recherche du plus court chemin (par groupes)

L'enseignant distribue la Fiche 38 à chaque élève (ceux-ci sont répartis par groupes, mais en raison du nombre de chemins à tester et à dessiner, il est préférable de donner une fiche à chacun). Il s'agit

de trouver tous les chemins possibles pour aller chercher 2 ressources, ou 3, ou 4, et de mesurer la longueur de ces chemins à l'aide d'une règle.
NB : on suppose qu'on ne passe par la base que deux fois : au départ et à l'arrivée.



Mise en commun

La discussion collective permet de faire ressortir le fait que le nombre de chemins possibles augmente très vite avec le nombre de ressources à aller chercher.

• Pour 2 ressources : si on note B la base, 1 et 2 les ressources, on a 2 trajets possibles : B12B ou B21B… ce qui correspond en fait au même chemin effectué dans un sens ou dans l'autre.

• Pour 3 ressources, il y a 6 trajets possibles, soit 3 chemins réellement différents :

– B123B et B321B (même chemin dans les deux sens)
– B132B et B231B (même chemin dans les deux sens)
– B213B et B123B (même chemin dans les deux sens)

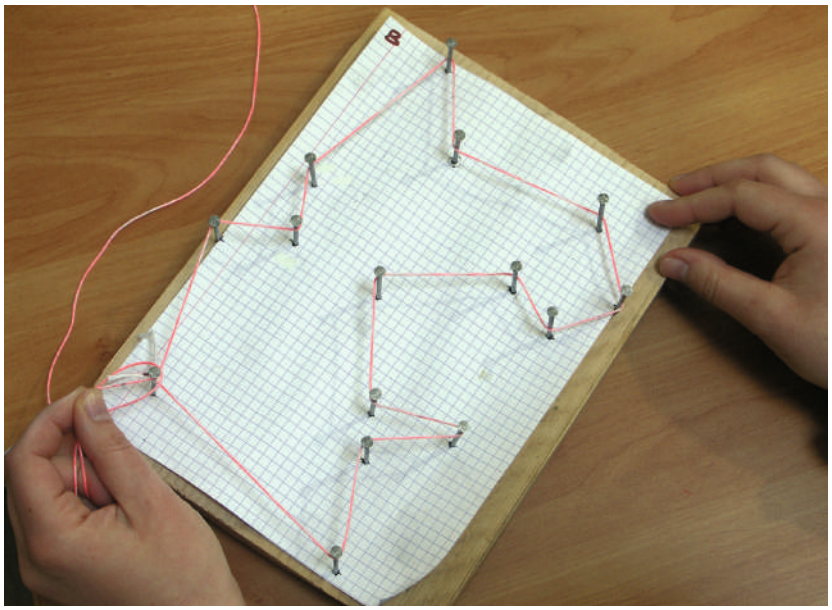
• Pour 4 ressources, il y a 24 trajets possibles soit 12 chemins réellement différents :

– B1234B et B4321B (même chemin dans les deux sens)
– B1243B et B3241B (idem)
– B1324B et B4231B
– B1342B et B2431B
– B1423B et B3241B
– B1432B et B2341B
– B2134B et 4312B
– B2143B et B3412B
– B2314B et B4132B
– B2413B et B3142B
– B3124B et B4214B
– B3214B et B4123B

Quand on élimine les doublons, il ne reste « que » 12 possibilités, ce qui est bien trop grand pour permettre aux élèves de toutes les trouver et de toutes les dessiner sur un même schéma.

(Facultatif) Recherche du plus court chemin (par groupes)

S'il dispose d'une planche cloutée pour chaque groupe, l'enseignant distribue cette planche et demande aux élèves de trouver le chemin le plus court permettant à la ficelle de passer par tous les clous (et de revenir au point de départ). Pour chaque trajet testé, on met une marque sur la ficelle afin d'indiquer la longueur atteinte (qui doit être la plus courte possible). Cette activité est similaire à la précédente, mais le fait d'avoir un support physique (les clous, la ficelle) permet aux élèves d'éliminer facilement certains « mauvais » chemins et d'avoir une intuition de ce que peut être un « bon » chemin.



Mise en commun

Cette nouvelle mise en commun permet de faire ressortir plusieurs choses :

• Il existe un très grand nombre de chemins possibles (personne n'a pu tous les tester).

• Au début de l'activité, les progrès sont rapides (on trouve des chemins de plus en plus courts), puis les progrès sont minimes (on a beau multiplier les essais, on ne gagne plus que quelques millimètres).

• Certains élèves peuvent penser avoir trouvé LE meilleur chemin. Il n'est pas possible de confirmer cette affirmation, mais on remarque que les « bons » chemins sont ceux qui évitent les croisements et limitent les retours en arrière.

Notes scientifiques

• Le nombre de chemins différents, pour n ressources + 1 base (ou n + 1 clous, comme dans la seconde expérience), est $n!$ (ce qui se lit « factorielle n »). Ce nombre se construit en multipliant n par tous les entiers situés en dessous de lui, jusqu'à 1. Par exemple :

– $1 ! = 1$

– $2 ! = 2 \times 1 = 2$

– $3 ! = 3 \times 2 \times 1 = 6$

– $4 ! = 4 \times 3 \times 2 \times 1 = 24$

• Si l'on élimine tous les chemins identiques mais pris dans un sens ou dans l'autre, cela donne, pour n ressources, un nombre $n!/2$ de chemins possibles.

• Cette fonction (qui, à un nombre n de ressources associe $n!/2$ chemins) augmente très rapidement :

– 5 ressources : 60 chemins
– 6 ressources : 360 chemins
– 7 ressources : 2 520 chemins
– 10 ressources : près de 2 millions de chemins
– 20 ressources : un milliard de milliards de chemins possibles
– Pour 100 ressources : le nombre de chemins possibles s'écrit avec 157 chiffres !

• Ce problème est devenu un « classique » en algorithmique, connu sous le nom du « problème du voyageur de commerce ». Un représentant doit démarcher plusieurs clients répartis sur le territoire. Quel chemin doit-il emprunter pour tous les démarcher une fois, en un minimum de temps ?

L'enseignant explique à la classe que le nombre de chemins augmente de façon vertigineuse (il peut expliquer que, pour 20 ressources, il y a plus de 1 milliard de milliards de possibilités). Il demande aux élèves s'ils connaissent un appareil qui permet de trouver le meilleur chemin parmi plusieurs. Il est probable que certains élèves citent le GPS. En remplaçant les ressources par des villes, le problème que doit résoudre un GPS est du même genre (quoiqu'un peu différent : le voyageur de commerce est obligé de passer par toutes les villes, ce qui n'est pas le cas lors de l'utilisation d'un GPS qui doit déterminer le chemin le plus court pour aller d'un endroit à un autre, en passant en général par plusieurs endroits intermédiaires). Comment faire, sachant qu'il existe, non pas 20 villes en France, mais des dizaines de milliers ?

La classe convient qu'il n'est pas possible de tester tous les chemins possibles (même un supercalculateur ne pourrait pas faire ce test en un temps inférieur à l'âge de l'Univers !). Le calculateur contenu dans un GPS procède de façon analogue à notre expérience de la planche à clous : il détermine un chemin au hasard, qu'il fait évoluer plusieurs fois de façon à réduire la longueur totale du trajet. Quand il ne parvient plus à réduire cette longueur de façon significative, il arrête la recherche et propose le meilleur chemin parmi ceux qu'il a explorés. Pour augmenter ses chances, il peut éventuellement tirer plusieurs chemins au hasard et faire évoluer chacun d'entre eux indépendamment.

Le GPS ne fournit donc pas le meilleur chemin possible, mais un chemin « plutôt bon ». Il n'est pas surprenant que 2 GPS différents fournissent, pour un même objectif, 2 solutions différentes car ils diffèrent à la fois par leur carte (qui peut être plus ou moins riche, plus ou moins actualisée) et par leur algorithme.

Conclusion et traces écrites

La classe synthétise collectivement ce qui a été appris au cours de cette séance :

• *Parfois, un problème est si long ou si compliqué à résoudre que l'on doit se contenter d'un algorithme qui ne donne pas une solution parfaite.*

• *Pour certaines tâches, comme trouver le meilleur chemin parmi de nombreuses possibilités, les ordinateurs sont bien plus rapides que les hommes.*

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

Prolongement

• Le visionnage du film *The Imitation Game* racontant la vie d'Alan Turing permet d'illustrer, dans un autre contexte (ici, il s'agit de casser un chiffrement utilisé par les nazis pendant la Seconde Guerre

mondiale), qu'il est parfois impossible de tester toutes les possibilités d'un problème et qu'il faut trouver une méthode pour restreindre la taille du problème en se concentrant sur les possibilités « intéressantes ». Même dans ce cas, le champ des possibles est très supérieur à la capacité de traitement d'un ou plusieurs êtres humains. Pour résoudre ce problème, Alan Turing et son équipe ont non seulement dû trouver un algorithme efficace, mais également fabriquer une machine pour l'exécuter : cette machine a conduit aux ordinateurs que nous connaissons aujourd'hui.

• On peut comparer les itinéraires fournis par plusieurs services de cartographie (Google Maps, Mappy, ViaMichelin…) ou plusieurs GPS, et ainsi vérifier que ces outils ne fournissent pas LE meilleur chemin (auquel cas, ils donneraient tous le même), mais un chemin « plutôt bon ».

FICHE 34
Cartes à jouer (1/2)

Carte n°1 Le score A vaut maintenant 3.	Carte n°2 Le score B vaut maintenant 4.	Carte n°3 Le score C vaut maintenant 5.	Carte n°4 Le score D vaut maintenant 6.
Carte n°5 Le score A est multiplié par 3.	Carte n°6 Le score B est multiplié par 3.	Carte n°7 Le score C est multiplié par 2.	Carte n°8 Le score D est multiplié par 2.
Carte n°9 Le score A augmente de 10.	Carte n°10 Le score B augmente de 8.	Carte n°11 Le score C augmente de 6.	Carte n°12 Le score D augmente de 4.
Carte n°13 Faire la somme des scores A et B. Le résultat obtenu est la nouvelle valeur du score A.	Carte n°14 Faire la somme des scores B et C. Le résultat obtenu est la nouvelle valeur du score B.	Carte n°15 Faire la somme des scores C et D. Le résultat obtenu est la nouvelle valeur du score C.	Carte n°16 Faire la somme des scores D et A. Le résultat obtenu est la nouvelle valeur du score D.
Carte n°17 Echanger les valeurs actuelles des scores A et C pour trouver les nouvelles valeurs de ces scores.	Carte n°18 Echanger les valeurs actuelles des scores B et D pour trouver les nouvelles valeurs de ces scores.	Carte n°19 Echanger les valeurs actuelles des scores A et B pour trouver les nouvelles valeurs de ces scores.	Carte n°20 Echanger les valeurs actuelles des scores C et D pour trouver les nouvelles valeurs de ces scores.
Carte n°21 Votre score vaut maintenant 10.	Carte n°22 Votre score vaut maintenant 10.	Carte n°23 Le score de l'équipe d'en face vaut maintenant 0.	Carte n°24 Le score de l'équipe d'en face vaut maintenant 0.

FICHE 35
Cartes à jouer (2/2)

Carte n°25 Si certains scores ont des valeurs paires, les diviser par deux pour obtenir leur nouvelle valeur. Conserver les valeurs des autres scores inchangées.	Carte n°26 Si certains scores sont des multiples de 3, les diviser par trois pour obtenir leur nouvelle valeur. Conserver les valeurs des autres scores inchangées.	Carte n°27 Si votre score vaut 0, remplacez-le par le meilleur score.	Carte n°28 Si vous avez le meilleur score, votre score vaut dorénavant 0. Sinon, conservez votre score inchangé.
Carte n°29 Votre nouveau score vaut maintenant la somme des scores de vos deux voisins immédiats.	Carte n°30 Echangez votre score avec celui de votre voisin de droite.	Carte n°31 Si votre score est pair, divisez-le par 2, sinon, ajoutez-lui 1.	Carte n°32 Faites tourner tous les scores dans le sens des aiguilles d'une montre (chaque équipe récupère le score de l'équipe qui est à sa droite).
Carte n°33 Choisissez votre nouveau score parmi tous les scores actuels, le vôtre compris.	Carte n°34 Si votre score est inférieur ou égal à 10, multipliez-le par lui-même pour trouver sa nouvelle valeur.	Carte n°35 Tous les scores supérieurs ou égaux à 5 perdent 5 points.	Carte n°36 Tous les scores inférieurs ou égaux à 5 augmentent de 5 points.
Carte n°37 Laissez tous les scores inchangés.	Carte n°38 Volez en tout 10 points à une ou plusieurs équipes concurrentes, sauf si la somme de leurs scores n'atteint pas 10.	Carte n°39 Tirez un dé à 6 faces. La face qui apparaît sur le dessus du dé correspond à votre nouveau score.	Carte n°40 Tirez un dé à 6 faces. La face qui apparaît sur le dessus du dé correspond au nouveau score de l'équipe en face de vous.
Carte n°41 Votre score …	Carte n°42 Votre score …	Carte n°43 Votre score …	Carte n°44 Votre score …
Carte n°45 Tous les scores …	Carte n°46 Tous les scores …	Carte n°47 Tous les scores …	Carte n°48 Tous les scores …

FICHE 36
Expressions logiques

Consigne: Indique pour chacune des expressions ci-dessous si elle est VRAIE, FAUSSE, ou si tu ne peux pas le savoir.

Expression 1 : Tous les chats sont des animaux
Expression 2 : Tous les animaux sont des chats
Expression 3 : En ce moment, la porte du gymnase est ouverte ou fermée
Expression 4 : En ce moment, la porte du gymnase est ouverte et fermée
Expression 5 : En ce moment, la porte du gymnase est ouverte

Expression 6

Expression 6a : Le chien est sur l'herbe et le chat est dans l'arbre
Expression 6b : Le chien est sur l'herbe ou le chat est dans l'arbre

---

Expression 7

Expression 7a : Le chien est sur l'herbe et le chat est dans l'arbre
Expression 7b : Le chien est sur l'herbe ou le chat est dans l'arbre

FICHE 37
Sécuriser la base

Consigne : Découpe les vignettes et combine-les de façon à écrire une condition qui déclenche l'alarme.

FICHE 38
Recherche du plus court chemin

Consigne : Le rover part de la base et va prendre toutes les ressources (ici, des cristaux de glace pour l'approvisionnement en eau), avant de retourner à la base. Dans chacun des cas, réponds aux 2 questions suivantes :

- Combien y a-t-il de chemins possibles (on suppose que le rover se déplace en ligne droite entre 2 étapes) ?
- Quel est le plus court chemin ?

Étape 6 – Éviter des obstacles, gérer son nombre de vies

Résumé	Les élèves complètent leur programme en introduisant des obstacles à éviter (nouveaux lutins) et en créant une variable pour leur nombre de « vies ». Ils réinvestissent les notions de test, de boucle et de variable abordées précédemment et approfondissent la notion d'événement.
Notions (*cf. scénario conceptuel, page 213*)	Idem étape précédente
Matériel	Pour chaque binôme : • Un ordinateur avec *Scratch* et le programme enregistré à la séance précédente

Note pédagogique

Il peut être utile, de temps en temps, de montrer le jeu « final » (version de l'enseignant) afin de relancer la motivation et de faire le point sur les étapes à venir. Il ne s'agit pas de donner la solution aux élèves en leur faisant lire le programme, mais simplement de faire une démonstration du jeu.

Pour pimenter le jeu simulant la mission spatiale, l'enseignant explique qu'il va falloir introduire des obstacles (un lac de lave, une dune de sable) et une nouvelle variable : le nombre de « vies ». Le rover démarre le jeu avec 3 vies et perd 1 vie, en revenant à la base, chaque fois qu'il touche un obstacle. Le jeu s'arrête lorsque le nombre de vies atteint 0.

Tâche 1 : ajout de nouveaux lutins (5 minutes)

Se référant à ce qu'ils ont appris lors des précédentes séances, les élèves importent sans difficulté les 3 nouveaux lutins (la base, la dune et la lave) et les initialisent en fixant leur position en un endroit précis de l'écran. Ils peuvent décider, par exemple, de placer la base au centre, ce qui nécessite de décaler la position initiale du rover (puisqu'on l'avait placé au centre).

Cela n'est pas indispensable, mais on peut vouloir préciser la « profondeur » des différents lutins (rover, base, lave, dune, végétation, glace), par exemple pour éviter que le rover ne roule sur la base. Décider quel lutin sera au premier plan se fait via l'instruction « envoyer au premier plan » disponible dans la catégorie « apparence » des instructions.

Tâche 2 : création et initialisation d'une variable « nombre de vies » (5 minutes)

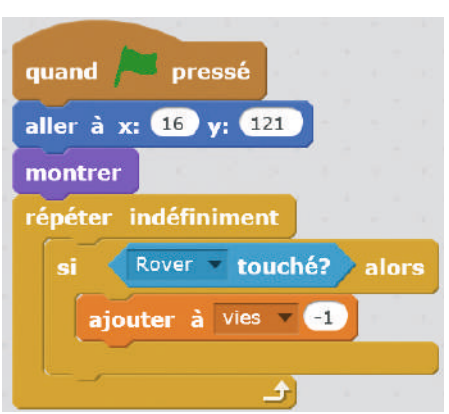
De même, ils créent facilement une nouvelle variable « vies » qu'ils initialisent à la valeur 3 (au même endroit que l'initialisation du score, par exemple dans le programme du rover). Le programme du rover contient donc désormais (en plus des sous-programmes permettant de le diriger) :

Tâche 3 : perdre une vie quand le rover touche la lave (30 minutes)

À la séance précédente, toucher une ressource permettait d'augmenter de 1 la valeur de la variable « score ». De même, ici, toucher la lave ou la dune doit faire diminuer de 1 la valeur de la variable « vie ». Nous conseillons de résoudre d'abord la tâche pour un des obstacles (la lave), puis de recommencer pour l'autre (la dune).

Comment soustraire dans Scratch ?

Les élèves cherchent comment contrôler/programmer cette diminution. Il n'y a pas de commande « soustraire », mais seulement « ajouter ». Au besoin, discuter collectivement du fait qu'il faut ajouter la valeur -1 pour soustraire 1.
Pour la lave, le programme est :



Programme provisoire pour la lave

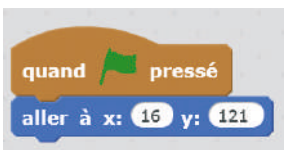
Ce programme a cependant un défaut : lorsque le rover entre en contact avec un obstacle, il le touche pendant un certain temps (quelques dixièmes de secondes, quelques secondes si on ne bouge plus) : pendant tout ce temps, la variable « vies » décroît. Au bout de quelques secondes, on se retrouve avec des valeurs négatives très grandes (-4000…). La seule façon d'éviter cela est de rompre immédiatement le contact entre le rover et l'obstacle.
Puisque l'obstacle est fixe, c'est donc le rover qui doit se déplacer. Or, cela ne peut pas être fait dans le programme de la lave ou de la dune : il faut être dans le programme du rover pour commander la position du rover.

Comment déplacer le rover ?

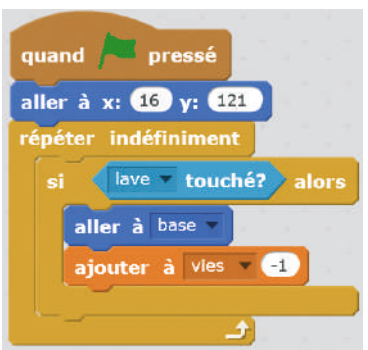
Il y a (au moins) deux solutions possibles à ce problème :

• **Solution 1**

Supprimer le programme que l'on a fait pour la lave (sauf l'initialisation de sa position) et en faire un similaire dans le programme du rover. Dans le programme du rover, on ajoute une commande disant, par exemple, de rentrer à la base. Notons, en passant, que pour rentrer à la base, on peut soit dire au rover d'aller en (X=0, Y=0), soit dire au rover d'aller à la position du lutin « base ». Cette dernière solution est meilleure car elle marchera même si on décide de placer la base ailleurs.



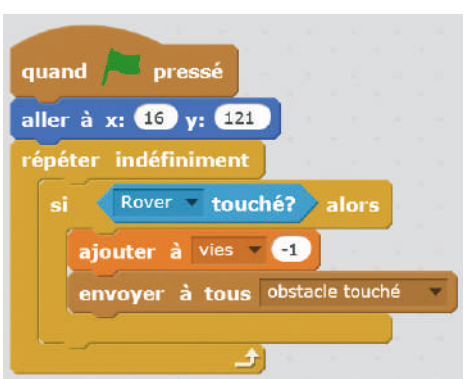
Programme de la lave



Programme du rover

• **Solution 2** (plus élégante… et plus pratique pour la suite)

Une solution plus élaborée consiste à garder le programme que l'on a fait pour la lave, en y ajoutant une nouvelle instruction qui va envoyer un message aux autres programmes (celui du rover en particulier). Ce message, dans le programme du rover, déclenchera une action (retourner à la base). Tout comme les noms de variables, les intitulés des messages doivent être explicites. Ici, notre message est par exemple « obstacle touché ».



Programme de la lave

Programme du rover

Le programme de la lave envoie un message aux autres programmes (celui du rover en particulier). Dans le programme du rover, la réception du message est un événement qui déclenche une action (aller à la base).

Les commandes permettant d'envoyer un message ou de déclencher une action lorsqu'un message est reçu se situent dans la catégorie « événements ».

Notes pédagogiques

• On peut aussi imaginer une autre solution, par exemple programmer le rover pour qu'il rebondisse s'il touche un obstacle (ainsi, le contact ne dure pas).
• La seconde solution peut être considérée comme plus élégante si l'on souhaite insister sur le caractère « événementiel » de ce programme. Chaque fois qu'un événement se produit (par exemple, toucher un obstacle), le programme peut envoyer un message qui sera utilisé par d'autres programmes. La notion d'événement a déjà été introduite lors des séances précédentes (« quand la flèche haute est touchée », « quand le drapeau vert est touché »…), mais c'est la première fois que l'on utilise ce nouveau type d'événement : la réception d'un message diffusé par un programme.
• Nous laissons le choix à l'enseignant d'utiliser l'une ou l'autre méthode selon qu'il préfère consacrer cette séance à des révisions de notions déjà vues ou s'il souhaite introduire ici la notion, nouvelle, de message. Bien sûr, il a tout intérêt à faire ce choix en tenant compte des idées proposées par les élèves et de leur maîtrise des notions précédentes !
• À noter : l'envoi et la réception de message seront (à nouveau) utilisés, plus tard, pour gérer la fin du jeu (voir « game over », page suivante).

Tâche 4 : refaire la même chose avec la dune (10 minutes)

Maintenant que les élèves ont réussi à gérer un des obstacles (la lave), ils doivent refaire la même chose pour l'autre (la dune).
Ce petit exercice permet de mieux s'approprier ce qui a été fait précédemment, en particulier l'envoi et la réception de message.

Conclusion et traces écrites

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

Étape 7 – Mettre fin au jeu : « game over »

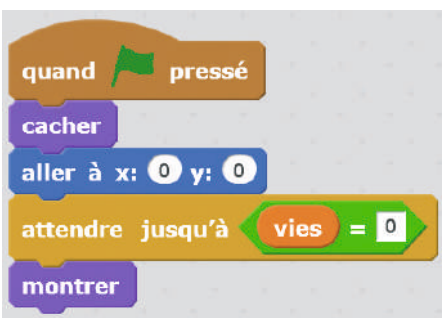
Résumé	Les élèves complètent leur programme en introduisant un test sur le nombre de vies restantes : le message « game over » apparaît et le programme s'arrête quand toutes les vies sont épuisées.
Notions (*cf. scénario conceptuel, page 213*)	Idem étape précédente
Matériel	Pour chaque binôme : • Un ordinateur avec *Scratch* et le programme enregistré à la séance précédente

Les élèves doivent maintenant faire en sorte que le jeu s'arrête lorsque le nombre de vies restantes vaut zéro. Cela suppose plusieurs choses :

- Faire apparaître « game over »
- Faire en sorte qu'on ne puisse plus jouer (sauf en relançant le programme)

Tâche 1 : faire apparaître « game over » quand le nombre de vies vaut 0 (15 minutes)

Faire apparaître « game over » peut se faire de plusieurs façons. La plus simple consiste à importer un lutin dont l'apparence est « game over » (lutin fourni). Ce lutin doit apparaître lorsque le nombre de vies vaut zéro. Cela se fait très facilement, à l'aide de l'instruction « montrer » dans la catégorie « apparence ». Note : il ne faut pas oublier, au lancement du programme, de cacher le lutin « game over » !
Le programme de ce lutin « game over » est :



Programme du lutin « game over »

Note pédagogique

Il est aussi possible d'importer un lutin « texte » depuis la bibliothèque « *Scratch* ». Ce lutin s'appelle « awesome ! » : en cliquant sur l'onglet « costume », on peut changer la couleur, la police de caractère… et éditer le texte.

Tâche 2 : arrêter le jeu quand apparaît « game over » (15 minutes)

En l'état actuel, on peut continuer à jouer lorsque « game over » s'affiche, ce qui n'est pas satisfaisant. Pour mettre fin au jeu, il y a plusieurs stratégies possibles :

• Méthode 1

Déclencher l'arrêt de tous les programmes quand le nombre de vies vaut zéro. Cela se fait via la commande « Stop tout » de la catégorie « contrôle ».

Programme du rover

En théorie, cela suffit à stopper le jeu. Malheureusement, en pratique, un bug de *Scratch* (présent à l'heure où nous publions ces lignes) fait que, malgré l'instruction « stop tout », certains programmes s'arrêtent et d'autres pas (il est encore possible de bouger le rover). Pour cette raison (et aussi car elle est visuellement plus satisfaisante), on préfère la seconde méthode, ci-après.

• Méthode 2

Faire en sorte que tous les autres lutins disparaissent lorsque le nombre de vies vaut zéro. Ainsi, le seul lutin encore à l'écran est « game over », et le jeu est bel et bien terminé. Cette méthode suppose qu'un des programmes (par exemple celui du rover) envoie un message. Ce message peut s'intituler tout simplement « game over ».

Tous les lutins (à l'exception du lutin « game over », bien sûr) se cachent lorsqu'ils reçoivent ce message. Puisqu'on leur demande de se cacher à la fin… il faut penser à leur demander de se montrer au lancement du programme !

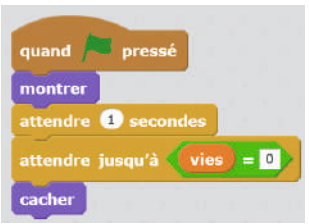
Programme du rover

Programme du lutin « game over »

La dernière instruction du programme du rover (se cacher quand on reçoit « game over ») est également présente dans les programmes des autres lutins (sauf le lutin « game over » qui, lui, se montre à ce moment-là).

Notes pédagogiques

• On peut faire une variante simplifiée (et moins élégante) de la méthode 2 si l'on ne souhaite pas utiliser l'envoi ou la réception de messages. Cette variante consiste à demander à chaque lutin de se cacher lorsque le nombre de vies atteint zéro.



La pause de 1 seconde permet de s'assurer que la variable « vies » a eu le temps, dans le programme du rover, d'être initialisée à 3 au lancement du programme. Sinon, comme elle vaut 0, les lutins se cachent immédiatement.

• Introduire une pause, même d'une fraction de seconde, peut permettre de résoudre des bugs assez subtils liés à des problèmes de synchronisation entre les différents programmes. *Scratch* donne l'illusion qu'il exécute tous les programmes en parallèle, mais en réalité il les exécute les uns après les autres (très rapidement… d'où l'illusion de la simultanéité). Introduire une pause ou envoyer/recevoir un message sont des manières de forcer l'exécution d'un programme avant un autre.

Conclusion et traces écrites

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

Étape 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 (*cf. scénario conceptuel, page 213*)	Idem étapes précédentes, plus : « Algorithmes » • Certaines boucles, dites « conditionnelles », sont répétées jusqu'à ce qu'une condition soit remplie.
Matériel	Idem étapes précédentes

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 3, page 227 et la Séance 4, page 233, 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 à un autre, on ne pourra plus l'éviter et le jeu prendra alors fin.

Notes pédagogiques

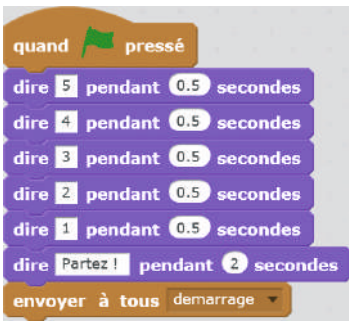
• Il est très probable que les élèves imaginent d'autres façons de faire. Nous conseillons à l'enseignant d'engager un débat dans la classe afin de choisir la ou les options qui seront suivies (rien n'impose que tous les groupes choisissent la même option !).

• 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 !

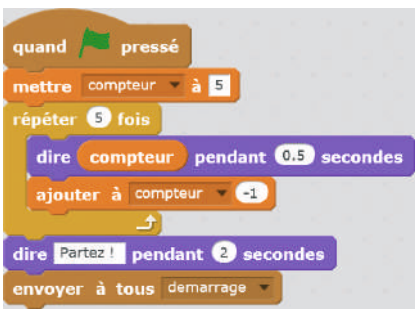
Nous décrivons ci-dessous les 2 options citées, ainsi que quelques autres qui ne sont pas forcément une manière de pimenter le jeu, mais de le rendre plus présentable ou d'éviter quelques bugs.
Les tâches ci-dessous sont de difficulté très variable, et toutes indépendantes les unes des autres.
Notre 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 !* Cela peut se faire très simplement, comme ceci (programme, à mettre par exemple, dans la scène) :



… 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.
En tout état de cause, il faut alors faire en sorte que les programmes des lutins ne démarrent qu'après le compte à rebours (la réception du message sera l'événement déclencheur des programmes).

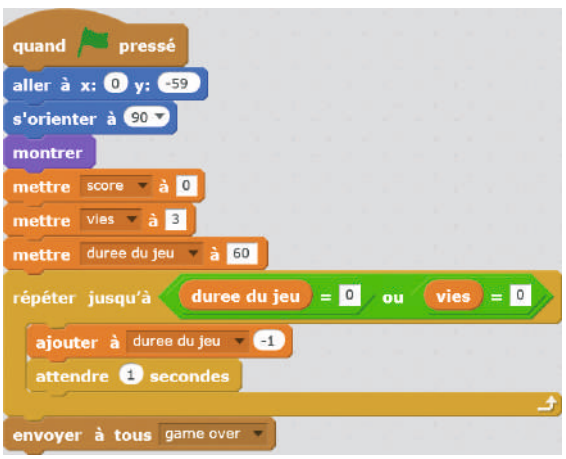
Tâche 2 : limiter la durée du jeu (15 minutes)

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 « durée 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 la durée du jeu est arrivée 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 « durée du jeu » à 20 et on lui retire 1 toutes les secondes. On peut bien sûr changer ces valeurs pour jouer plus ou moins longtemps.

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

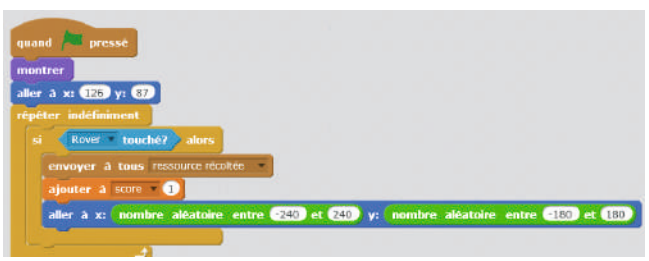
La création du lutin à partir d'une image et l'initialisation de sa position sont désormais connues des élèves. On peut, par exemple, faire partir la tornade du coin inférieur gauche (X = -230 et Y = -170).
Il faut ensuite trouver une façon de la déplacer vers un endroit aléatoire. Si l'on souhaite que ce mouvement soit instantané, il suffit de changer les valeurs de X et de Y… mais il serait préférable de voir la tornade se 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 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 quel que soit le nombre de vies restant. On ajoute donc l'instruction suivante au programme de la tornade :

Tâche 4 : faire grossir la tornade peu à peu (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 chaque fois qu'une ressource est récoltée.



Dans le programme de la tornade, augmenter la taille de celle-ci 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 ».

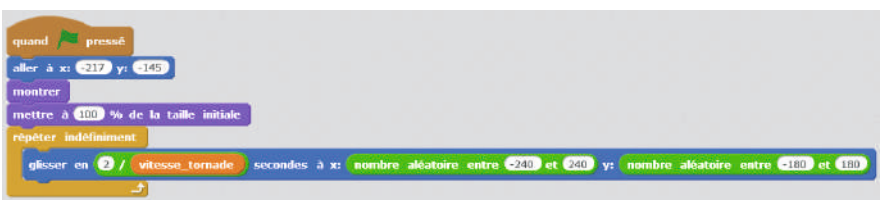
Tâche 5 : faire accélérer la tornade peu à peu (20 minutes)

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 chaque fois que l'on a récolté une ressource (glace ou végétation). Pour faire accélérer la tornade, on peut par exemple 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 la valeur « 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 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 !).

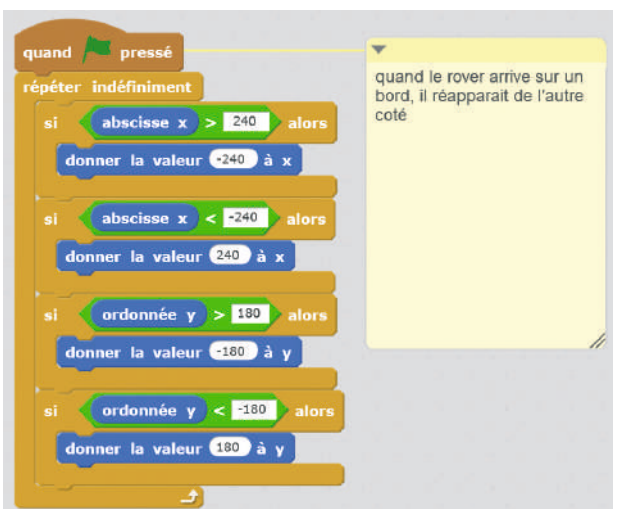
L'enseignant laisse les élèves tâtonner et les guide en cas de difficulté, tout d'abord en explicitant leur algorithme : si l'abscisse X du rover dépasse 240 (extrémité droite), alors elle doit passer à -240 (extrémité gauche). Ensuite, il peut montrer les différents blocs nécessaires à la construction 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'emboîtent 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

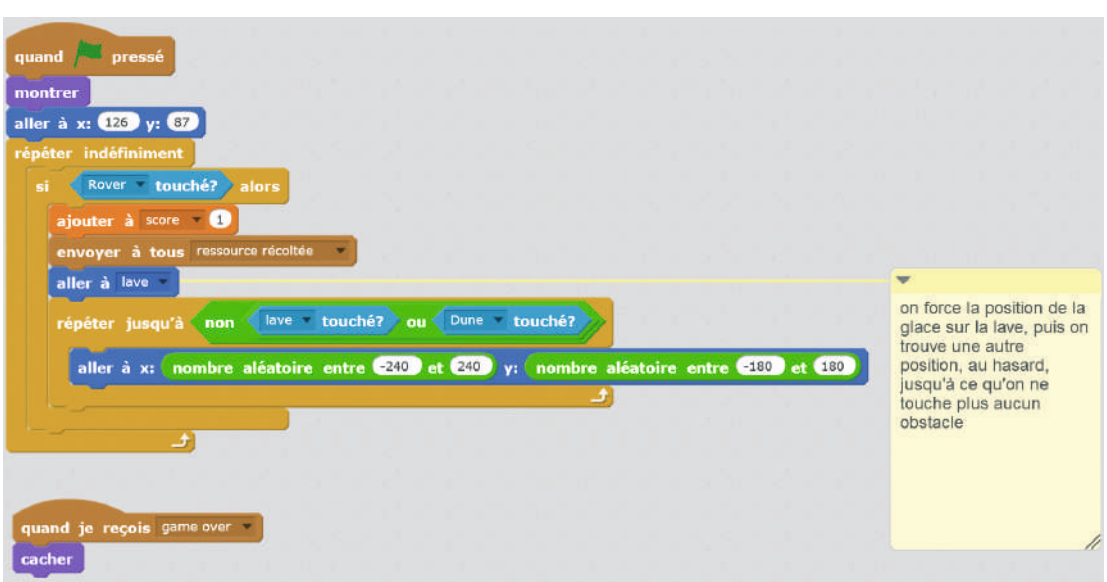
• La capture d'écran ci-dessus montre qu'il est possible d'ajouter des commentaires à un programme. C'est une très bonne habitude à prendre, de manière à rendre le programme plus clair pour soi, et surtout pour les autres qui vont le lire !
• À ce stade, il devient nécessaire de supprimer l'instruction « 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 laissés 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 :



Remarque : la condition [NON (lave touchée OU dune touchée)] peut aussi s'écrire [NON (lave touchée)] ET [NON (dune touchée)]. Au besoin, cette activité peut être rapprochée de celle réalisée à l'étape 5 (activité 3, page 270).

Conclusion et traces écrites

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.

Étape 9 – Prolongements possibles en *Scratch*

Résumé	Nous proposons ici quelques pistes pour explorer d'autres fonctionnalités de *Scratch*, par exemple pour alimenter des projets personnels d'élèves.
Notions (*cf. scénario conceptuel, page 213*)	Idem Étape 1, page 241
Matériel	Pour chaque binôme : • Un ordinateur avec *Scratch* et le programme enregistré à la séance précédente
Durée	1 à 4 séances

Avant-propos

Au cours des séances précédentes, les élèves se sont familiarisés avec les fonctionnalités et concepts essentiels de *Scratch* :

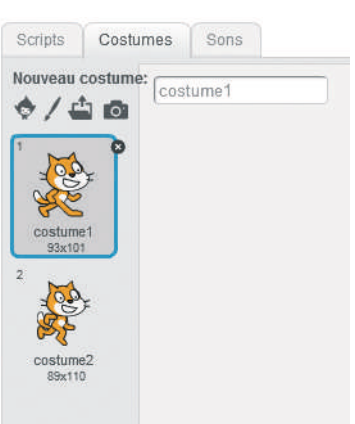
- Lutins
- Scènes
- Blocs d'instruction
- Mouvements
- Apparences
- Evénements
- Boucles
- Tests
- Variables
- Calculs et opérateurs logiques
- Capteurs

Il existe néanmoins de nombreuses fonctionnalités qui n'ont pas été abordées. Certaines sont très simples d'emploi et peuvent grandement enrichir les programmes que les élèves pourront réaliser par eux-mêmes : jeux, cartes animées, questionnaires interactifs, dessins… Nous en présentons la liste ci-dessous.

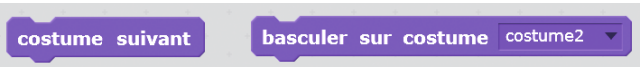
Costumes

Chaque lutin peut posséder plusieurs « costumes » qui correspondent à son apparence physique. Un nouveau costume peut être une petite variation d'un costume précédent (changement de position d'un bras ou d'une jambe…) ou être très différent des autres costumes.

Créer un nouveau costume est possible après avoir sélectionné le lutin concerné et cliqué sur la catégorie « costumes ». Ce nouveau costume peut être basé sur l'ancien costume, sur une image importée ou un dessin réalisé par l'utilisateur.

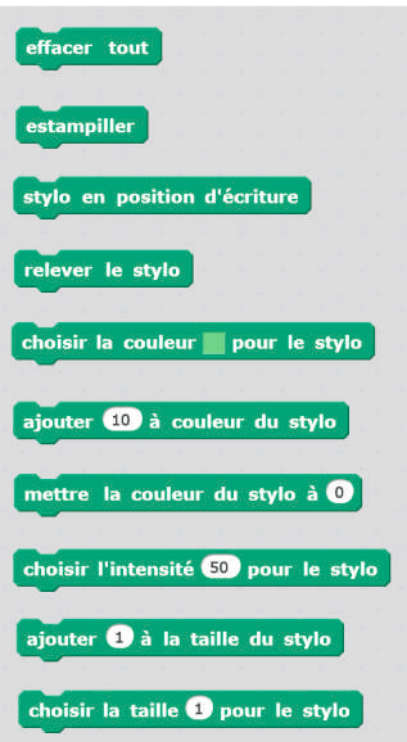


Ici, le chat (lutin par défaut dans *Scratch*) possède 2 costumes. On peut lui faire changer de costume tout en le faisant avancer, ce qui donne une illusion de marche.
Les 2 instructions permettant de changer de costume sont disponibles dans le bloc « apparence » :



Dans la continuité du travail réalisé au cours de cette séquence, on peut demander aux élèves, par exemple, de définir un nouveau costume pour le rover correspondant à un rover « brûlé » ou « cassé », à activer quand le rover rencontre un obstacle.

Stylo



Chaque lutin possède un « stylo » qui, lorsqu'il est en position basse, laisse une trace sur le sol (c.-à-d. l'écran). Par défaut, le stylo est en position « haute », et ne laisse donc pas de trace.
Les instructions accessibles via la catégorie vert « stylo » permettent de relever ou d'abaisser le stylo, de changer l'intensité du tracé, son épaisseur, sa couleur…
Il est ainsi possible d'utiliser *Scratch* pour réaliser, par exemple, des figures géométriques.

Paroles et questions/réponses

Chaque lutin peut afficher du texte à l'écran (instruction « dire », accessible dans la catégorie « apparence »). Il peut aussi, de façon plus intéressante, interagir avec l'utilisateur, en posant des questions (instruction « demander », accessible dans la catégorie « capteurs »). Il attend alors la réponse tapée au clavier par l'utilisateur et l'enregistre dans une variable prédéfinie appelée « réponse ». La réponse, comme toute variable, peut être manipulée par le programme.
Ce petit programme, par exemple, salue l'utilisateur en l'appelant par son prénom :

Celui-ci pose une question et félicite l'utilisateur en cas de réponse correcte.
Il est donc possible (et fort intéressant !) d'utiliser *Scratch* pour demander aux élèves de programmer des quiz ou des évaluations dans différentes matières.
Dans le cadre du projet réalisé au cours de cette séquence, le joueur peut par exemple se voir demander la durée du jeu, au lancement du programme, ou le nombre de vies qu'il souhaite avoir.

Mais aussi...

Scratch permet d'aller encore plus loin et propose des fonctionnalités avancées, mais qui nous paraissent un peu complexe pour le cycle 3. Notamment :

• Les clones, qui permettent de dupliquer un lutin en autant d'exemplaires que souhaité, chacun étant toujours considéré comme le même lutin que l'original (il obéit aux mêmes programmes). Dans le cadre du projet réalisé au cours de cette séquence, on peut, par exemple, souhaiter faire afficher un nombre aléatoire de ressources « végétation » et les placer au hasard dans la scène. Les clones sont accessibles via la catégorie « contrôle ».

• Les fonctions, qui permettent de définir soi-même de nouveaux blocs d'instructions. Cela peut se révéler très utile si l'on réutilise souvent le même ensemble d'instructions. Plutôt que de les répéter, les regrouper dans une fonction permet de gagner du temps et d'aérer le code, tout en réduisant les erreurs possibles. Les fonctions sont accessibles via le menu violet foncé « ajouter bloc » (créer un nouveau bloc).

• Le son : l'utilisation est très simple (chaque lutin peut jouer un son préenregistré, ou une note dont on règle la hauteur et le volume), mais peut devenir vite laborieuse dans un contexte de classe…

Cette ressource est issue du projet thématique *1,2,3... CODEZ !*, paru aux Éditions Le Pommier.

Retrouvez l'intégralité de ce projet sur : https://www.fondation-lamap.org/projets-thematiques.

Fondation *La main à la pâte*
43 rue de Rennes
75006 Paris
01 85 08 71 79
contact@fondation-lamap.org

Site : www.fondation-lamap.org

FONDATION La main à la pâte
POUR L'ÉDUCATION À LA SCIENCE