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

1, 2, 3, codez ! - Activités cycle 3 - Séance 1.1: Comment diriger un véhicule à distance ?

Résumé	Les élèves doivent donner des instructions pour piloter un véhicule à distance. Pour ce faire, ils définissent un langage de programmation et explorent la différence entre une langue naturelle et un langage. Ils découvrent également la notion de bug.
Notions	« Machines » • Les machines qui nous entourent ne font qu'exécuter des ordres (instructions) « Langages » • En informatique, on invente et on utilise des 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 • Un langage de programmation est différent d'une langue naturelle - Il possède très peu de mots et de règles de grammaire - Il ne laisse place à aucune ambiguïté - Il est compréhensible par certaines machines • Il existe de nombreux langages de programmation, adaptés à différents usages • Un bug est une erreur dans un programme. • Un bug minime en apparence peut avoir des conséquences énormes.
Matériel	Pour la classe • Fiche 28 (facultatif) • Une salle informatique, avec une connexion Internet

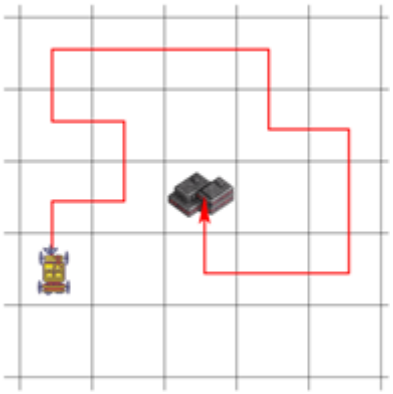
Lexique	Langage de programmation, instruction, bug
Durée :	1 h 30 (éventuellement en 2 fois 45 minutes)

Situation déclenchante – présentation du projet

L'enseignant explique à la classe que le projet consiste à simuler une mission d'exploration sur une planète lointaine. Pour le moment, la classe va préparer la mission : réfléchir à la façon dont on va se déplacer, communiquer…. Dans un second temps, la classe va « jouer » la mission, à travers un programme de simulation que chacun va pouvoir créer (petit jeu vidéo).
La mission est habitée et, sur la planète, l'équipage dispose déjà d'une base et d'un véhicule terrestre (de type « rover »). L'environnement est hostile, donc lors des sorties d'exploration, une personne doit toujours rester à la base par sécurité. Si les personnes qui sont sur le terrain ne sont plus en mesure de piloter le rover (par exemple, si elles ont perdu connaissance), la personne d'astreinte doit pouvoir diriger le rover à distance pour le ramener à la base, sans avoir besoin de parler à l'équipage. Les ordres de déplacement sont donnés au rover sous forme d'ondes, mais il faut inventer un langage pour donner ces ordres.

La question est donc : quel langage utiliser pour piloter un rover à distance ?

L'enseignant affiche ou projette une carte de la zone d'exploration (Fiche 28). Cette zone est quadrillée et un parcours est dessiné de façon à pouvoir rentrer à la base en évitant les zones dangereuses. On ne peut pas faire de raccourci : il faut absolument suivre tout le parcours dessiné, dans le sens de la flèche.

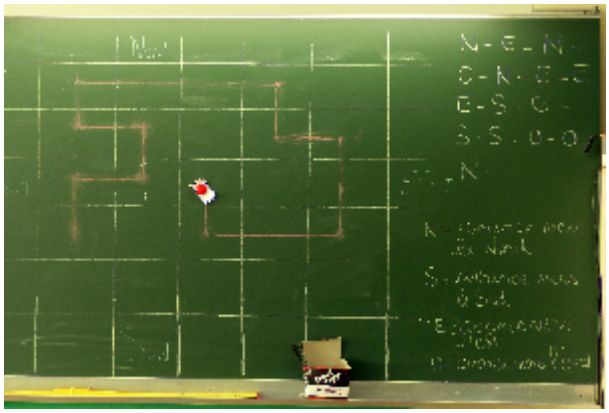


Recherche : définition d'un langage (par binômes)

L'enseignant distribue la Fiche 28 aux élèves, répartis par binômes. Ils doivent définir quels types d'instructions donner au rover pour lui faire suivre le parcours imposé afin de rejoindre la base. Les déplacements se font carreau par carreau, et pas en diagonale.

Mise en commun

Après quelques minutes, la classe met en commun les travaux des différents binômes. On peut, par exemple, dessiner ou projeter le parcours au tableau et, lorsqu'un binôme présente sa solution, vérifier collectivement qu'elle est correcte en l'exécutant au tableau (on prend un objet quelconque pour représenter le rover, cet objet doit suivre scrupuleusement les instructions données).



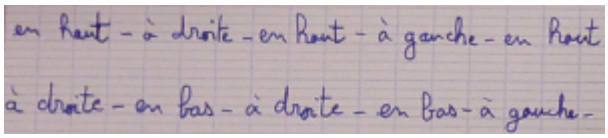
Classe de CM1 de Carole Vinel (Paris)

On remarque qu'il existe (au moins) deux logiques pour diriger le rover. On peut lui donner des directions « absolues » (va au Nord, va à l'Ouest…) ou, au contraire, des directions relatives, c'est-à-dire qui dépendent de l'orientation du rover (tourne vers la droite, avance, tourne vers la gauche, recule…).
Note : il est préférable de découper l'instruction « avance d'une case vers la droite » en 2 instructions bien distinctes : tourne vers la droite (sous-entendu : en restant sur place), puis avance d'une case.

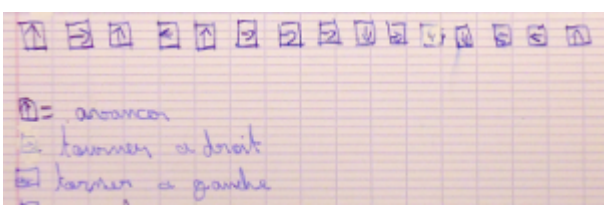
Notes pédagogiques

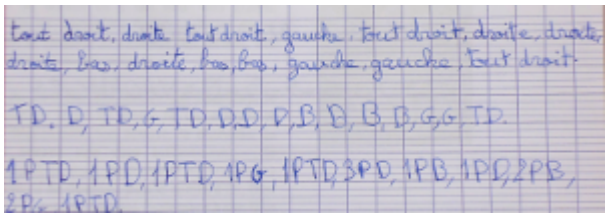
- La première logique (Nord, Ouest…) est dite « allocentrée » tandis que la seconde logique (droite, gauche…) est dite « autocentrée ».
- Les élèves n'ont pas besoin de connaître ces termes, qui ne seront plus utilisés par la suite.
- Une troisième logique peut (plus rarement) être proposée : il s'agit de donner des coordonnées aux cases (A1, A2, B1…) et, comme dans un jeu de bataille navale, coder les déplacements en donnant le nom de la case de départ et d'arrivée. Exemple, « va de A1 vers A2 ». À noter : le chemin « A1 vers A2 » n'est pas ambigu car les cases sont adjacentes. En revanche, le chemin « A1 vers B7 » est ambigu (et, donc, non satisfaisant) : il y a plusieurs façons d'aller de la case A1 à la case B7. Nous ne détaillons pas cette méthode dans ce qui suit.

Il est probable que les 2 méthodes aient été trouvées par les uns ou les autres. Si cela n'est pas le cas, l'enseignant introduit l'autre méthode lors de cette mise en commun.

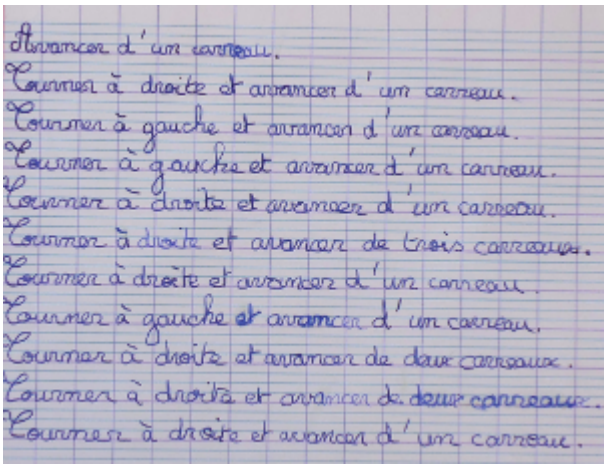


langage allocentré sans gestion de la





évolution de la proposition d'un élève pendant la phase de recherche



langage autocentré entièrement rédigé

Quelques propositions d'élèves. Classe de CM2 d'Anne-Marie Lebrun (Bourg-la-Reine).

L'enseignant fait remarquer que les instructions sont exprimées dans un langage particulier, avec un vocabulaire très restreint, et non ambigu : chaque instruction est parfaitement explicite et ne peut pas donner lieu à plusieurs interprétations. Il s'agit d'un langage de « programmation ».
Ce langage peut encore être simplifié. Par exemple, il est inutile de dire « Va vers l'Est » quand on peut simplement dire « Est » ou « Va à droite » quand on peut simplement dire « Droite » (si on a bien défini au préalable ce que l'on entend par « Droite »,par exemple, « va d'une case vers la droite » et non pas « pivote sur toi-même d'un quart de tour vers la droite »).

Collectivement, la classe explicite le vocabulaire de 2 langages, par exemple :

Langage allocentré (ou « absolu »)	**Langage autocentré (ou « relatif »)**
• **Nord** (signifie « avance d'une case vers le Nord ») • **Sud** • **Est** • **Ouest**	• **Avancer** (signifie « avance d'une case droit devant soi ») • **Droite** (signifie « pivote sur place d'un quart de tour vers la droite ») • **Gauche**

On remarque que le langage allocentré nécessite 4 mots de **vocabulaire** tandis que le langage autocentré n'a besoin que de 3 mots. Certains élèves pourront proposer l'instruction « Recule », mais on peut remarquer que le rover se retrouve dans la même case s'il recule d'une case ou s'il fait « Droite, Droite, Avance»). Dans ce dernier cas, il a changé d'orientation. Si on souhaite qu'il reprenne son orientation initiale, il faut écrire « Droite, Droite, Avancer, Droite, Droite ».

On remarque également qu'il est possible de réduire encore le lexique de ce langage autocentré. « Gauche » par exemple peut se dire « Droite, Droite, Droite ». Ainsi, 2 mots peuvent suffire. Pour plus de clarté, on peut décider de garder 3 ou 4 mots, selon ce qui sera

décidé par les élèves.

L'enseignant fait remarquer que la **grammaire** est également très simple. Il n'y a pas de genre, de nombre, de mode, de temps… La seule règle présente ici est celle de la séquence : quand 2 instructions se suivent, par exemple « droite avance », cela signifie qu'elles doivent être exécutées l'une après l'autre.
Pour plus de clarté lors de l'écriture et la lecture, on peut décider (ou non !) d'introduire une règle supplémentaire, comme séparer les instructions par des virgules.
Enfin, la classe remarque que ces langages ne permettent pas de faire autre chose qu'un déplacement sur un quadrillage (on ne peut pas afficher du texte ou faire des calculs) : les langages de programmation sont spécialisés. L'enseignant peut faire remarquer qu'il existe d'autres langages similaires (peu de « mots », peu de règles de grammaires, peu ou pas d'ambigüité…) : la notation musicale par exemple.

Sensibilité aux erreurs

L'enseignant demande aux élèves ce qui se passe si on introduit une erreur dans un programme (par exemple, si on oublie une instruction). On peut prendre un exemple concret, en se basant sur le parcours initial du rover (Fiche 28). Que se passe-t-il si l'on saute une instruction ? On s'aperçoit que quel que soit le langage utilisé, on rate l'objectif. On remarquera qu'une erreur dans un langage autocentré peut conduire plus loin de l'objectif qu'une erreur dans un langage allocentré. Cependant, dans les deux cas, il s'agit d'un bug et on notera deux choses. Premièrement, l'objectif n'est pas atteint, donc c'est un échec aussi important dans un cas que dans l'autre. Deuxièmement, si le terrain présente des obstacles (crevasses…), alors on ne veut pas se tromper... même pas un tout petit peu. Les deux bugs sont donc aussi problématiques l'un que l'autre !

La classe discute des différentes origines possibles d'un bug. Il peut s'agir d'une erreur dans l'algorithme (la méthode), ou d'une erreur dans le programme (l'expression de l'algorithme dans le langage choisi : une erreur de syntaxe, par exemple), ou encore d'une erreur matérielle (liée, par exemple, à une panne d'un élément de la machine, ou à une erreur dans la transmission des instructions, comme ici).

> **Note pédagogique**
> Nous avons choisi d'utiliser l'orthographe anglaise « bug » (qui signifie « bestiole » en anglais) plutôt que l'orthographe francisée « bogue » de façon à respecter l'origine historique de ce mot. C'est en effet Grace Hopper qui a popularisé l'expression « bug », inventée par Thomas Edison, lorsqu'elle a découvert l'origine d'une panne dans son ordinateur : un insecte grillé au cœur des circuits. Voir l'éclairage scientifique sur Grace Hopper.

Conclusion et trace écrite

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

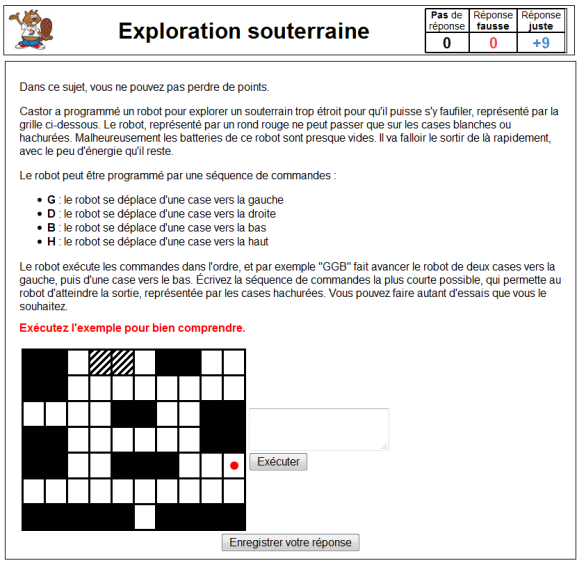
- *En informatique, on invente et on utilise des langages*
- *Pour donner des instructions à une machine, on utilise un langage de programmation, compréhensible à la fois par la machine et par l'être humain*

- *Un langage de programmation est différent d'une langue naturelle*
  - Il possède très peu de mots et de règles de grammaire
  - Il ne laisse place à aucune ambiguïté
- *Il existe de nombreux langages de programmation, adaptés à différents usages*
- *Un bug est une erreur dans un programme.*
- *Un bug minime en apparence peut avoir des conséquences énormes.*

Les élèves notent ces conclusions dans leur cahier de sciences. L'enseignant, quant-à-lui, prépare une affiche intitulée « Qu'est-ce que l'informatique ? ». Cette affiche sera remplie au fur et à mesure du projet et permettra de dresser un panorama général de cette science (notions de langage, d'algorithme, de programme, de machine, d'information…). Il commence par recopier ce que la classe a appris sur la notion de langage au cours de cette séance.

Exercice en ligne

Si l'école dispose d'une salle informatique, on peut prolonger cette séance par un court exercice en ligne permettant de retravailler les notions de programme et de bug. L'activité peut aussi être menée collectivement au vidéoprojecteur ou sur TNI, quelques jours après cette séance, de façon à en renforcer les acquis.



Prolongement débranché (Cycle 4)

Cette séance peut être prolongée par un exercice de traduction d'un langage vers un autre langage, à faire en binôme :

- Traduire la séquence d'instructions « Nord, Ouest, Nord, Est, Est, Est, Sud, Est, Sud, Ouest, Ouest, Nord, Nord » (langage allocentré) en une expression autocentrée. Vérifier sur une feuille quadrillée que les 2 expressions conduisent bien au même résultat.
- Traduire (pour un rover orienté initialement vers le Nord) « Droite, Avancer, Avancer, Gauche, Avancer, Gauche, Avancer, Avancer, Avancer, Avancer, Droite, Avancer » en une expression allocentrée et, de même, vérifier sur une feuille quadrillée le résultat.

Si ce prolongement a été traité, ajouter dans la trace écrite la notion suivante :

- *On peut traduire le même ensemble d'instructions d'un langage vers un autre.*

Séquence III-1 Séance III-1.2 >>

Source URL: http://www.fondation-lamap.org/fr/page/34514/1-2-3-codez-activites-cycle-3-seance-11-comment-diriger-un-vehicule-a-distance