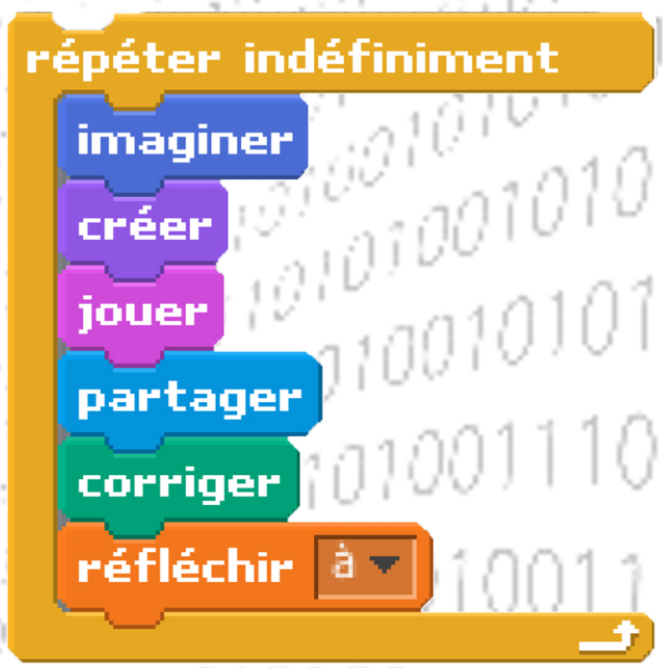
DSDEN de Côte d'Or
équipe départementale ERUN

Programmation

et robot éducatif

pour le cycle 3



Quelques définitions

(issues du dictionnaire Larousse)

Robot :
(du tchèque robota, travail forcé, mot créé en 1920 par K. Čapek)
« Appareil automatique capable de manipuler des objets ou d'exécuter des opérations selon un programme fixe, modifiable ou adaptable. »

Codage :
« Action de coder un texte, des données informatiques ; chiffrement, encodage. »

Programmer :
- « Établir à l'avance une suite d'opérations ; planifier, déterminer à l'avance le moment et les modalités d'une action. »
- « Écrire les programmes informatiques correspondant à l'algorithme de résolution d'un problème. »
- « Commander un dispositif à l'aide d'un programme. »

La programmation à l'école primaire

I. Socle commun de connaissances, de compétences et de culture

Domaine 1 : les langages pour penser et communiquer
→ Comprendre, s'exprimer en utilisant les langages mathématiques, scientifiques et informatiques
• Il sait que des **langages informatiques** sont utilisés pour **programmer** des outils numériques et réaliser des traitements automatiques de données. Il connaît les principes de **base de l'algorithmique et de la conception des programmes informatiques**. Il les met en oeuvre pour **créer des applications simples**.

Domaine 2 : les méthodes et outils pour apprendre
→ Coopération et réalisation de projets
• L'élève **travaille en équipe, partage des tâches**, s'engage dans un dialogue constructif, accepte la contradiction tout en défendant son point de vue, fait preuve de diplomatie, négocie et recherche un consensus.
• Il apprend à gérer un projet, qu'il soit individuel ou collectif. Il en **planifie les tâches**, en fixe les étapes et évalue l'atteinte des objectifs.

Domaine 4 : les systèmes naturels et les systèmes techniques
→ Démarches scientifiques
• L'élève sait mener une **démarche d'investigation**. Pour cela, il décrit et questionne ses observations ; il prélève, organise et traite l'information utile ; il formule des hypothèses, les teste et les éprouve ; il manipule, explore plusieurs pistes, procède par essais et erreurs ; il modélise pour représenter une situation ; il analyse, argumente, mène différents types de raisonnements (par analogie, déduction logique...) ; il rend compte de sa démarche. Il exploite et communique les résultats de mesures ou de recherches en utilisant les langages scientifiques à bon escient.

II. Programme du cycle 1

→ Apprendre en jouant
→ Apprendre en réfléchissant et en résolvant des problèmes
→ Apprendre en s'exerçant
→ Apprendre en se remémorant et en mémorisant

1. Mobiliser le langage dans toutes ses dimensions
◦ Oser entrer en communication
◦ Comprendre et apprendre
◦ Échanger et réfléchir avec les autres

2. Explorer le monde
• Consolider la notion de chronologie
• L'espace
◦ Faire l'expérience de l'espace
◦ Représenter l'espace
Par l'utilisation et la production de représentations diverses (photos, maquettes, dessins, plans…) et également par les échanges langagiers avec leurs camarades et les adultes, les enfants apprennent à **restituer leurs déplacements** et à en effectuer à partir de consignes orales comprises et mémorisées. Ils établissent alors les **relations entre leurs déplacements et les représentations** de ceux-ci.
◦ Découvrir différents milieux
L'enseignant conduit les enfants de l'observation de l'environnement proche (la classe, l'école, le quartier...)

Ce qui est attendu des enfants en fin d'école maternelle
- Ordonner une suite de photographies ou d'images, pour rendre compte d'une situation vécue ou d'un récit fictif entendu, en marquant de manière exacte succession et simultanéité.
- Situer des objets par rapport à soi, entre eux, par rapport à des objets repères.
- Se situer par rapport à d'autres, par rapport à des objets repères.
- Dans un environnement bien connu, réaliser un trajet, **un parcours à partir de sa représentation (dessin ou codage)**.
- Élaborer des premiers essais de représentation plane, communicables (construction d'un code commun).
- Orienter et utiliser correctement une feuille de papier, un livre ou un autre support d'écrit, en fonction de consignes, d'un but ou d'un projet précis.
- Utiliser des marqueurs spatiaux adaptés (devant, derrière, droite, gauche, dessus, dessous…) dans des récits, descriptions ou explications.

3. Explorer le monde du vivant, des objets et de la matière
◦ Utiliser des outils numériques
Dès leur plus jeune âge, les enfants sont en contact avec les nouvelles technologies. Le rôle de l'école est de leur donner des repères pour en comprendre l'utilité et commencer à les utiliser de manière adaptée (tablette numérique, ordinateur, appareil photo numérique…).
Ce qui est attendu des enfants en fin d'école maternelle
- Réaliser des constructions ; construire des maquettes simples en fonction de plans ou d'instructions de montage.
- Utiliser des objets numériques : appareil photo, tablette, ordinateur.

III. Programme du cycle 2

• Observer et utiliser des objets techniques et identifier leur fonction (Par l'usage de quelques objets techniques, actuels ou anciens, identifier leur domaine et leur mode d'emploi, leurs fonctions. Dans une démarche d'observation, démonter- remonter, procéder a des tests et essais.
• Commencer à s'approprier un environnement numérique
• L'espace
◦ **Se repérer dans l'espace et le représenter** : Se repérer dans son environnement proche - Produire des représentations des espaces familiers (les espaces scolaires extérieurs proches, le village, le quartier) et moins familiers (vécus lors de sorties) - Lire des plans, se repérer sur des cartes.

Mathématiques :
• **Se repérer dans son environnement proche**. Situer des objets ou des personnes les uns par rapport aux autres ou par rapport à d'autres repères.
→ Vocabulaire permettant de **définir des positions** (gauche, droite, au-dessus, en dessous, sur, sous, devant, derrière, près, loin, premier plan, second plan, nord, sud, est, ouest…)
→ Vocabulaire permettant de **définir des déplacements** (avancer, reculer, tourner à droite/a gauche, monter, descendre…).
• **S'orienter et se déplacer en utilisant des repères. Coder et décoder** pour prévoir, représenter et réaliser des déplacements dans des espaces familiers, sur un quadrillage, sur un écran.

IV. Programme du cycle 3

Contributions au socle commun :
→ La maîtrise des techniques et la connaissance des règles des outils numériques se construisent notamment à travers l'enseignement des sciences et de la technologie où les élèves apprennent à connaître l'organisation d'un environnement numérique et à utiliser différents périphériques ainsi que des logiciels de traitement de données numériques (images, textes, sons...). En mathématiques, ils apprennent à utiliser des **logiciels de calculs et d'initiation à la programmation.**

Sciences et technologie
• Mobiliser des outils numériques
    ◦ Utiliser des outils numériques pour :
        - communiquer des résultats ;
        - traiter des données ;
        - simuler des phénomènes ;
        - représenter des objets techniques.
    ◦ Identifier des sources d'informations fiables.
• Repérer et comprendre la communication et la gestion de l'information
• Environnement numérique de travail.
    ◦ Le stockage des données, notions d'algorithmes, les objets programmables.
    ◦ Usage des moyens numériques dans un réseau.
    ◦ Usage de logiciels usuels

En CM1 et CM2 l'observation de communications entre élèves, puis de systèmes techniques simples permettra de progressivement distinguer la notion de signal, comme grandeur physique, transportant une certaine quantité d'information, dont on définira (cycle 4 et ensuite) la nature et la mesure.

V. Cadre de référence des compétences numériques

Domaine 3 – Création de contenu

3.4 Programmer

De quoi s'agit-il ?
Écrire des programmes et des algorithmes pour répondre à un besoin (automatiser une tâche répétitive, accomplir des tâches complexes ou chronophages, résoudre un problème logique, etc.) et pour développer un contenu riche (jeu, site web, etc.) avec des environnements de développement informatique simples, des logiciels de planification de tâches, etc

cf http://eduscol.education.fr/cid111189/projet-de-cadre-de-reference-des-competences-numeriques-pour-l-ecole-et-le-college.html

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

1, 2, 3, codez ! - Activités cycle 3 - Séquence 1 : Préparer la mission

	Séance	Titre	Résumé
	Séance 1	Comment diriger un véhicule à distance ?	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.
	Séance 2	Comment encoder un message avec des nombres ?	Les élèves doivent encoder un message textuel en utilisant seulement des nombres. Pour cela, ils font des propositions puis conviennent d'un tableau de correspondance entre caractères textuels et nombres, commun à toute la classe. Ils utilisent ce tableau pour encoder un message qu'ils doivent envoyer, puis pour décoder un message qu'ils ont reçu.
	Séance 3	Comment coder des informations en binaire ?	Les élèves doivent maintenant se contenter de deux symboles (0 et 1) pour transmettre des messages. Ils explorent les moyens d'encoder différentes informations (les 4 points cardinaux Nord/Sud/Est/Ouest, les 7 jours de la semaine, etc.) en associant des 0 et des 1, et découvrent ainsi le codage binaire de l'information.

	Séance 4	Comment encoder et décoder un message en binaire ?	Dans la continuité de la séance précédente, les élèves appliquent leurs découvertes à l'encodage d'un court message textuel en binaire, puis au décodage d'un message en binaire qu'ils reçoivent.

Activités cycle 3 Séquence III-2 >>

Source URL: http://www.fondation-lamap.org/fr/page/34513/1-2-3-codez-activites-cycle-3-sequence-1-preparer-la-mission

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

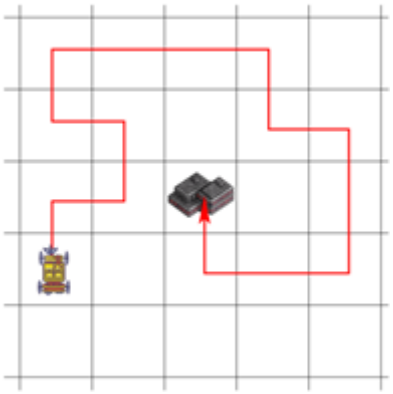
Lexique	Langage de programmation, instruction, bug
Durée :	1 h 30 (éventuellement en 2 fois 45 minutes)

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.

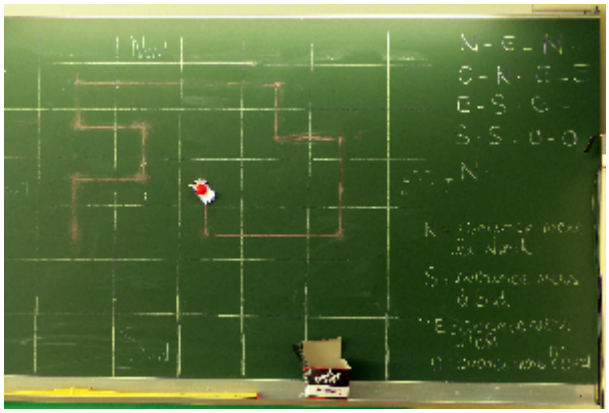


Recherche : définition d'un langage (par binômes)

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).



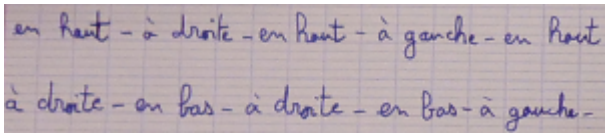
Classe de CM1 de Carole Vinel (Paris)

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.

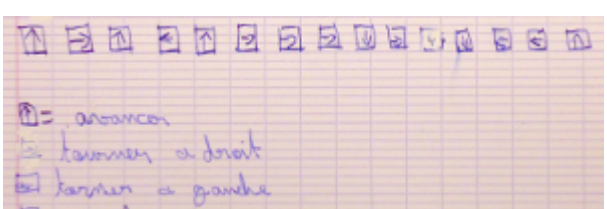
Notes pédagogiques

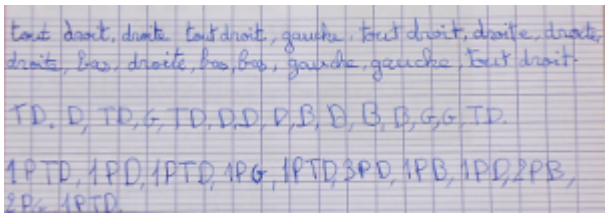
- La première logique (Nord, Ouest…) est dite « allocentrée » tandis que la seconde logique (droite, gauche…) est dite « autocentrée ».
- Les élèves n'ont pas besoin de connaître ces termes, qui ne seront plus utilisés par la suite.
- Une troisième logique peut (plus rarement) être proposée : il s'agit 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.

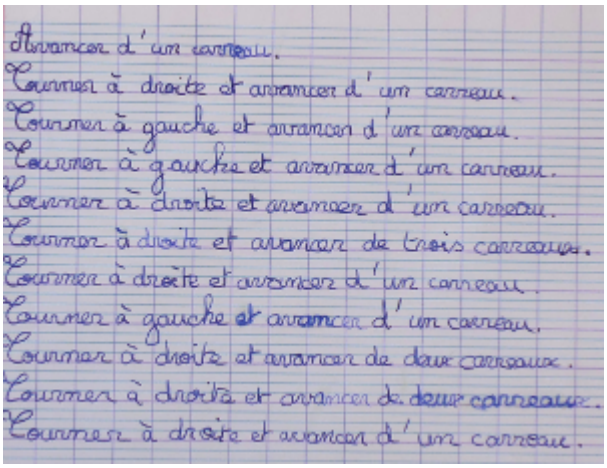


langage allocentré sans gestion de la





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.

Exploration souterraine

Pas de réponse	Réponse fausse	Réponse juste
0	0	+9

Dans ce sujet, vous ne pouvez pas perdre de points.

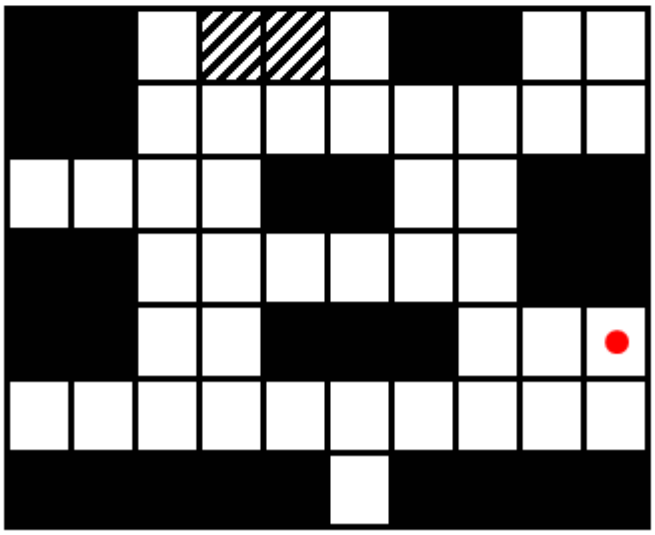
Castor a programmé un robot pour explorer un souterrain trop étroit pour qu'il puisse s'y faufiler, représenté par la grille ci-dessous. Le robot, représenté par un rond rouge ne peut passer que sur les cases blanches ou hachurées. Malheureusement les batteries de ce robot sont presque vides. Il va falloir le sortir de là rapidement, avec le peu d'énergie qu'il reste.

Le robot peut être programmé par une séquence de commandes :

- **G** : le robot se déplace d'une case vers la gauche
- **D** : le robot se déplace d'une case vers la droite
- **B** : le robot se déplace d'une case vers la bas
- **H** : le robot se déplace d'une case vers la haut

Le robot exécute les commandes dans l'ordre, et par exemple "GGB" fait avancer le robot de deux cases vers la gauche, puis d'une case vers le bas. Écrivez la séquence de commandes la plus courte possible, qui permette au robot d'atteindre la sortie, représentée par les cases hachurées. Vous pouvez faire autant d'essais que vous le souhaitez.

Exécutez l'exemple pour bien comprendre.



Exécuter

Enregistrer votre réponse

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

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 1.2: Comment encoder un message avec des nombres ?

Résumé	Les élèves doivent encoder un message textuel en utilisant seulement des nombres. Pour cela, ils font des propositions puis conviennent d'un tableau de correspondance entre caractères textuels et nombres, commun à toute la classe. Ils utilisent ce tableau pour encoder un message qu'ils doivent envoyer, puis pour décoder un message qu'ils ont reçu.
Notions	« Information » • Un caractère peut être représenté par un nombre. • Un texte, qui est une succession de caractères, peut-être représenté par une succession de nombres. • L'encodage du texte est le remplacement de ses caractères par les nombres qui leur correspondent. Le décodage est l'opération inverse.
Matériel	Pour la classe • Fiche 29
Lexique	Table de correspondance, encodage, décodage
Durée :	1h15

Note scientifique préalable sur le vocabulaire

- Dans le langage courant, les termes « coder », « chiffrer », « crypter » sont souvent confondus et employés à mauvais escient. Le terme « coder », par exemple, est parfois utilisé pour signifier « programmer » (on lit aussi « écrire du code »), parfois pour « représenter de l'information » (par exemple, le codage binaire), ou encore pour « altérer un message pour le rendre incompréhensible » (en référence à la notion de « code » secret…).
- Dans tout ce qui suit, nous tâcherons d'employer les mots dans leur signification

scientifique, c'est-à-dire

- **Coder** signifie représenter de l'information. Au cours de cette séance, les élèves vont représenter un message textuel à l'aide de nombres. Lorsqu'ils transformeront les lettres en nombre, on parlera d'**encodage** et, lorsqu'ils effectueront la transformation inverse, de **décodage**. C'est l'objet de cette séance, puis des deux suivantes sur le binaire.
- **Chiffrer** un message consiste à le déformer pour le rendre incompréhensible à toute personne non destinataire du message (qui ne possède pas la clé pour le déchiffrer). C'est l'objet de la Séance III-3.4.
- Les termes **crypter/décrypter** (que nous éviterons ici) sont en général employés comme synonymes de chiffrer/déchiffrer, mais il existe une nuance. La cryptanalyse consiste à **casser** le chiffrement (on devine quelle est la clé alors qu'on n'est pas censé la connaître).

Situation déclenchante

L'enseignant explique à la classe que l'équipe du poste de contrôle communique avec le rover et avec l'équipe de terrain en utilisant des instruments électroniques. Ces instruments permettent d'envoyer et de recevoir seulement des nombres, pas de lettres. Tout message textuel à transmettre doit donc être converti en une succession de nombres avant l'envoi (c'est l'opération d'encodage du message), puis reconverti en une succession de caractères textuels à l'arrivée (c'est l'opération de décodage du message). Il ne peut pas y avoir d'espace ou de virgule entre les nombres : les nombres arrivent tous « collés » à la suite les uns des autres.

La question est donc est la suivante : comment encoder des messages textuels sous forme de nombres, puis les décoder ?

Recherche : stratégie d'encodage et de décodage (par binômes)

L'enseignant annonce que l'équipe de terrain souhaite prolonger sa sortie au-delà de l'horaire initialement prévu. Mais le vent semble se lever. Il faut envoyer au poste de contrôle un message qui commence comme ceci :

ENVOYEZ BULLETIN METEO.

Par binômes, les élèves recherchent une stratégie d'encodage de ce texte en nombres. Il ne s'agit pas pour le moment d'effectuer l'encodage proprement dit, mais seulement d'en proposer les modalités.

Lorsqu'un binôme annonce avoir trouvé une stratégie d'encodage, l'enseignant donne en secret à chaque élève un petit papier sur lequel est écrit un court message textuel contenant au moins un espace ou un point : par exemple « PLUS VITE. », « TRES BIEN. » ou « TOUT VA BIEN.». Il demande aux élèves d'encoder le message à destination de leur camarade, qui doit ensuite le décoder. L'enseignant rappelle que les nombres du message encodé devront être tous collés les uns à la suite des autres, sans espace ni ponctuation. Les deux élèves vérifient si la transmission de l'information s'est bien passée. Si ce n'est pas le cas, le binôme peut identifier le problème et changer ou améliorer sa stratégie.

Classe de CM2 de Christelle Crusberg (Champigny-sur-Marne)

Notes pédagogiques

- Le texte à encoder contient uniquement des lettres majuscules sans accents, des points et des espaces. De ce fait, il est probable que la plupart des propositions des élèves consistent à associer chaque lettre de l'alphabet à un nombre, conformément à l'ordre alphabétique (1 pour coder A, 2 pour coder B, etc. jusqu'à 26 pour coder Z) et à ajouter une correspondance entre le point, l'espace et deux autres nombres (par exemple 27 et 28 ou bien 27 et 0).
- Toutefois, certains groupes imaginent de coder par des nombres différents les lettres majuscules (de 1 à 26 par exemple) et minuscules (de 27 à 52 par exemple). La question des lettres accentuées peut également émerger, ainsi que celle des caractères de ponctuation autres que le point. Bien que le message à coder ne comporte pas de chiffres, certains groupes peuvent chercher à les prendre en compte : ils peuvent alors associer les chiffres 0 à 9 aux nombres 0 à 9, et le codage des lettres ne commence qu'à partir de 10.
- Toutes ces propositions doivent être valorisées, même si on effectue ensuite, pour des raisons de simplicité, le choix de ne pas coder les chiffres et d'en rester à un codage identique pour toutes les variantes d'une lettre (A, a, â, à tous codés par 01, par exemple).
- Enfin, certains élèves risquent d'anticiper sur la question du chiffrement du message en proposant une correspondance peu intuitive entre caractères textuels et nombres, voire en proposant une correspondance instable (qui change tous les 10 caractères par

exemple). On proposera à ces élèves de retenir leurs idées pour une autre séance (Séance III-3.4).

Mise en commun

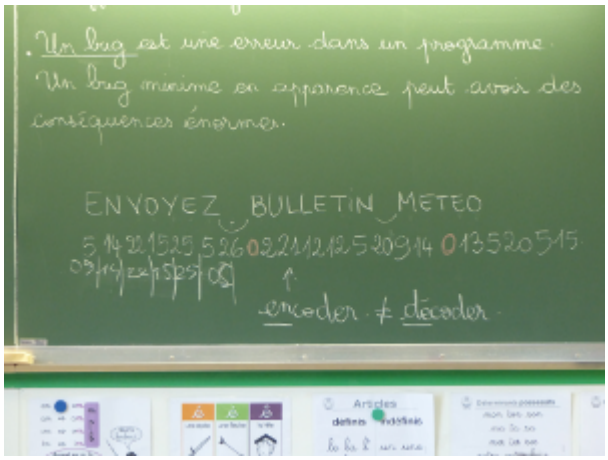
Lors de la mise en commun, l'enseignant demande à un premier binôme de présenter sa proposition. Celle-ci est confrontée aux propositions des autres groupes (voir notes pédagogiques ci-dessous), en pesant les avantages et les inconvénients de chacune. Puis, la classe s'arrête sur un choix de consensus : celui d'une table de correspondance biunivoque (aucune ambigüité, ni à l'encodage, ni au décodage) entre les caractères textuels que l'on souhaite utiliser dans les messages et des nombres.
La table de correspondance retenue pour la suite est la suivante (noter que la table véritablement utilisée peut varier d'une classe à l'autre, notamment pour l'espace et le point) :

Caractère textuel	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
Nombre	01	02	03	04	05	06	07	08
Caractère textuel	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**
Nombre	09	10	11	12	13	14	15	16
Caractère textuel	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**
Nombre	17	18	19	20	21	22	23	24
Caractère textuel	**Y**	**Z**	**point**	**espace**				
Nombre	25	26	27	28				

Noter que les nombres 1 à 9 (correspondant aux lettres A à I) ont été notés 01 à 09, de façon à ce que tous les nombres utilisés pour coder les caractères textuels soient écrits avec autant de chiffres (2 en l'occurrence). Ainsi, 0221 se lit 02 21 et se décode en BU et 2201 se lit 22 01 et se décode en VA. Alors que si les lettres A à I étaient codées par les nombres 1 à 9, le texte encodé 221 pourrait être lu soit 2 21, soit 22 1 et être décodé indifféremment en BU ou en VA.

Exercice – Encoder un message (par groupes)

L'enseignant distribue aux élèves le haut de la Fiche 29 (table de correspondance et consigne 1) et leur demande d'effectuer l'encodage du message destiné au poste de contrôle (consigne 1), en utilisant la table de correspondance choisie par la classe. Une fois les messages encodés, l'enseignant affiche les résultats et annonce que le message vient d'être envoyé au poste de contrôle. Il insiste sur le fait que le message de départ et le message encodé contiennent la même information sous deux formes différentes.



Discussion sur l'importance d'un encodage sur 2 chiffres pour chaque caractère. Classe de CM1 de Carole Vinel (Paris)

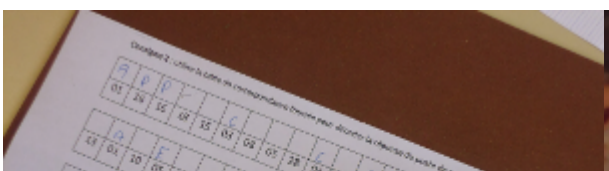
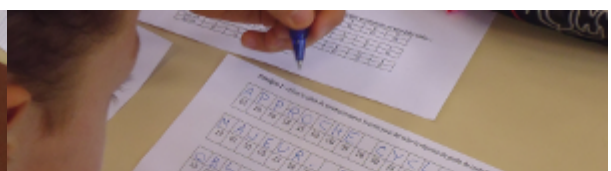
Défi – Décoder un message (par groupes)

L'enseignant annonce que la classe vient de recevoir la réponse du poste de contrôle, sous forme encodée. Il va maintenant s'agir de décoder cette réponse.
Il distribue la partie basse de la Fiche 29 (consigne 2) et laisse aux élèves le temps de mener à bien le décodage.

La classe partage le message décodé :

APPROCHE CYCLONE MAJEUR. RETOUR PARKING OBLIGATOIRE. INDIQUEZ DELAI PREVU.

La mission ne peut donc pas être prolongée, le rover doit rentrer immédiatement à la base.

caractère dans le texte (à droite).
Classe de CM2 d'Anne-Marie Lebrun (Bourg-la-Reine).

Note pédagogique
L'exercice d'encodage et le défi de décodage peuvent être menés de façon collaborative : un élève donné encode ou décode seulement une ligne du message, et les résultats sont mis en commun. Nous proposons toutefois de distribuer les messages en entier aux élèves, ce qui simplifie la logistique de la séance, permet aux élèves les plus rapides d'avoir constamment du travail et à tous de garder mémoire du travail collectif accompli.

Conclusion et trace écrite

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

- *Un caractère peut être représenté par un nombre.*
- *Un texte, qui est une succession de caractères, peut donc être représenté par une succession de nombres.*
- *L'encodage du texte est le remplacement de ses caractères par les nombres qui leur correspondent. Le décodage est l'opération inverse.*

Les élèves notent ces conclusions dans leur cahier de sciences. L'enseignant, quant-à-lui, complète l'affiche « Qu'est-ce que l'informatique ? » avec ce que la classe a appris au cours de cette séance sur les notions d'information et d'algorithme.

Prolongements

Avec les plus grands, le message à décoder peut être un peu plus long. Par exemple, le message ci-dessous a la propriété amusante d'être un pangramme (toutes les lettres de l'alphabet y sont présentes au moins une fois) :

WARNING. APPROCHE CYCLONE MAJEUR. VENTS EXCEPTIONNELLEMENT FORTS ATTENDUS. RETOUR PARKING OBLIGATOIRE. INDIQUEZ DELAI PREVU.

<< Séance III-1.1 Séquence III-1 Séance III-1.3 >>

Source URL: http://www.fondation-lamap.org/fr/page/34515/1-2-3-codez-activites-cycle-3-seance-12-comment-encoder-un-message-avec-des-nombres

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 1.3: Comment coder des informations en binaire ?

Résumé	Les élèves doivent maintenant se contenter de deux symboles (0 et 1) pour transmettre des messages. Ils explorent les moyens d'encoder différentes informations (les 4 points cardinaux Nord/Sud/Est/Ouest, les 7 jours de la semaine, etc.) en associant des 0 et des 1, et découvrent ainsi le codage binaire de l'information.
Notions	« Information » • Un ordinateur représente n'importe quelle information par un code qui n'utilise que 2 symboles, 0 et 1, appelés bits : c'est le code binaire. • Le codage binaire permet de représenter toutes sortes de données, notamment des nombres ou des caractères textuels. • Plus on juxtapose de bits, plus on peut représenter d'éléments différents.
Matériel	Pour chaque élève • Fiche 30
Lexique	Liste d'éléments, bit, codage binaire
Durée :	1h30

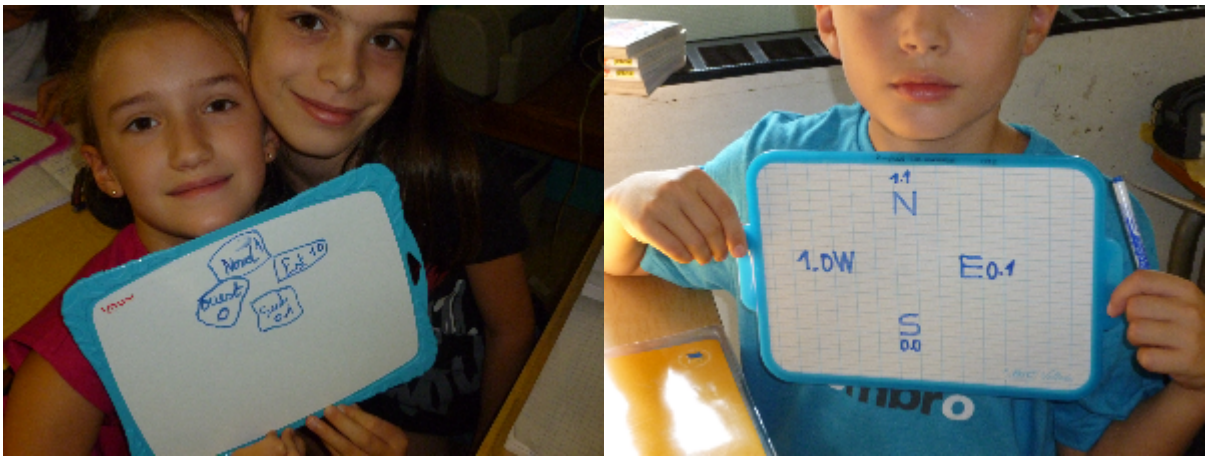
Situation déclenchante

L'enseignant explique à la classe qu'en réalité, les instruments électroniques ne peuvent pas transmettre directement les nombres : ils transmettent des flux de signaux lumineux ou électriques. Ces signaux n'ont que deux états : NON (pas de signal) / OUI (signal) aussi appelés 0 et 1. Les élèves doivent donc repenser leur système de codage avec cette nouvelle contrainte.

La question est donc est la suivante : comment encoder des informations en utilisant seulement des 0 et des 1 ?

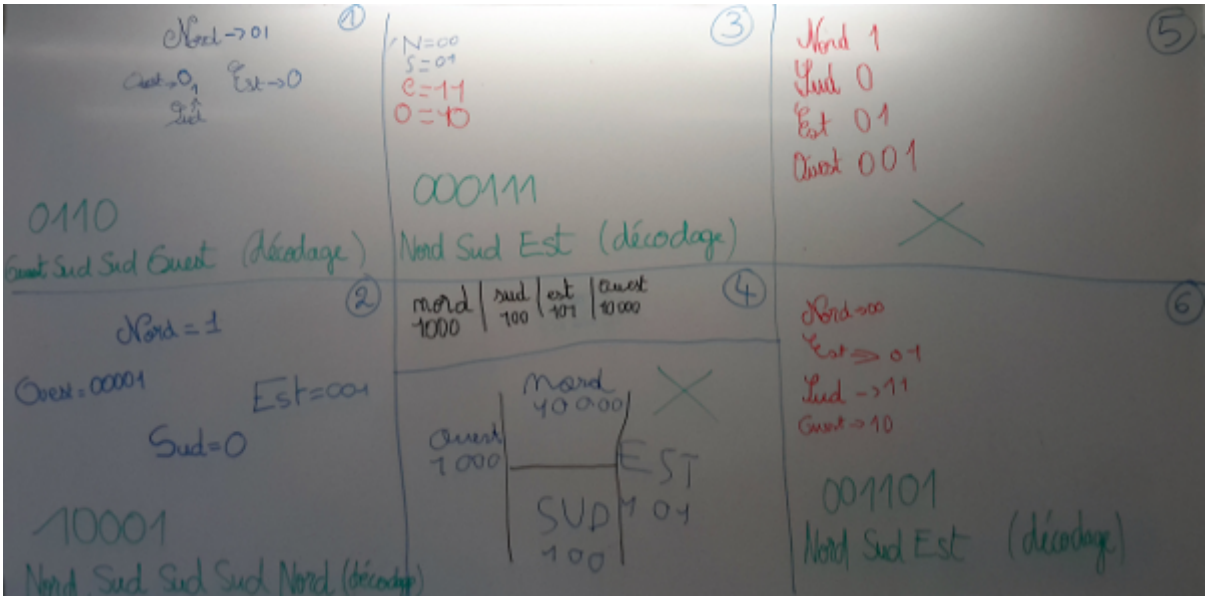
Recherche : stratégie d'encodage utilisant seulement des 0 et 1 (par binômes et collectivement)

L'enseignant rappelle que le rover peut être dirigé depuis le poste de contrôle en utilisant les 4 mots Nord/Sud/Est/Ouest. Il demande aux élèves de réfléchir en binômes à un moyen de transmettre ces 4 mots en utilisant seulement des 0 et des 1.



Propositions de quelques élèves. Classe de CM1-CM2 de Thyphaine Collignon (Vernaison).

Après quelques minutes, les groupes partagent leurs propositions (voir les notes pédagogiques ci-dessous).
La classe discute les propositions et constate que celles qui fonctionnent consistent toutes à assembler plusieurs 0 et 1.



Mise en commun et discussion sur l'importance d'encoder chacune des informations sur un même nombre de chiffres, en vue du décodage. Classe de CM2 de Christelle Crusberg (Champigny-sur-Marne).

Notes pédagogiques

- Une première proposition peut consister à remplacer :
  « Nord » par « 00 »,
  « Sud » par « 11 »,
  « Est » par « 01 »,
  « Ouest » par « 10 ».
  Dans ce type de proposition, chaque point cardinal est codé par une succession de deux « 0 ou 1 » (un doublet). Toutes les combinaisons de deux « 0 ou 1 » sont utilisées. Il est important de faire remarquer aux élèves que selon les groupes, ce n'est pas forcément le même point cardinal qui est associé à un doublet donné : toutes les correspondances point cardinal / doublet se valent.
- Une deuxième proposition peut consister à remplacer :
  « Nord » par « 1000 »,
  « Sud » par « 0100 »,
  « Est » par « 0010 »,
  « Ouest » par « 0001 ».
  Dans ce type de proposition, chaque point cardinal est codé par une succession de quatre « 0 ou 1 ». Seules quelques-unes des combinaisons de quatre « 0 ou 1 » sont utilisées (par exemple, la combinaison 0000 ne l'est pas).
- Une troisième proposition peut consister à remplacer :

  « Nord » par :

  ```
      1
  0       0
      0
  ```

  « Sud » par :

  ```
      0
  0       0
      1
  ```

  « Est » par :

  ```
      0
  0       1
      0
  ```

  « Ouest » par :

  ```
      0
  1       0
      0
  ```

  Face à ce type de proposition, l'enseignant précise que l'idée est bonne, mais que les 0 et les 1 ne peuvent être transmis que les uns à la suite des autres, donc sans positionnement dans l'espace. Cette troisième proposition devient alors équivalente à la proposition 2 : pour chaque point cardinal, on transmet quatre « 0 ou 1 » dont un seul est un « 1 ». Il y a quatre positions possibles pour le « 1 », une par point cardinal.

Exercice : choix du nombre de 0 et de 1 à assembler pour coder les jours de la semaine (par groupes)

L'enseignant indique que tous les messages échangés entre la base et le rover sont datés. Il faut donc être capable d'indiquer avec des 0 et des 1, entre autres choses, le jour de la semaine. L'enseignant demande aux élèves de proposer une façon de coder les 7 jours de la semaine en alignant le moins possible de « 0 ou 1 ».

Classe de CM2 de Christelle Crusberg (Champigny-sur-Marne)

Mise en commun

La mise en commun révèle qu'il suffit d'aligner trois « 0 ou 1 », par exemple comme ceci (mais il y a une multitude de possibilités équivalentes) :

000 pour lundi
001 pour mardi
010 pour mercredi
011 pour jeudi
100 pour vendredi
101 pour samedi
110 pour dimanche

Il reste même une combinaison (ici 111) qui n'est pas utilisée.
L'enseignant guide la formulation orale d'une conclusion avec pour support la Fiche 30 projetée et distribuée à chacun : *On peut coder les éléments d'une liste par une succession de « 0 ou 1 » aussi appelés bits* (contraction de « binary digit », soit « nombre binaire »).

- Avec un seul bit, on peut coder tous les éléments d'une liste qui n'a pas plus de 2 éléments (blanc/noir, éteint/allumé sont des exemples de listes à deux éléments) : on associe le « 0 » à un élément de la liste, et le « 1 » à l'autre élément de la liste (la classe note des exemples de listes à deux éléments sur la Fiche 30).
- Avec deux bits juxtaposés, on peut coder tous les éléments d'une liste qui n'a pas plus de 4 éléments (comme Nord/Sud/Est/Ouest), car il existe 4 façons différentes d'agencer 2 chiffres valant 0 ou 1 : « 00 », « 01 », « 10 » et « 11 » (la classe note sur la Fiche 30 des exemples de listes codables sur 2 bits, mais pas sur 1 bit : ces listes ont 3 ou 4 éléments).
- Avec trois bits juxtaposés, on peut coder tous les éléments d'une liste qui n'a pas plus de 8 éléments (par exemple les 7 jours de la semaine), car les 8 seules combinaisons de 0 et de 1 possibles sont : « 000 », « 001 », « 010 », 011 », « 100 », « 101 », « 110 » et « 111 » (La classe note sur la Fiche 30 des exemples de listes codables sur 3 bits, mais pas sur 2 bit (ces listes ont 5, 6, 7 ou 8 éléments).
- Plus on juxtapose de bits, plus on peut représenter d'éléments : 16 éléments au maximum en juxtaposant quatre bits, 32 éléments au maximum en juxtaposant cinq bits, 64 éléments au maximum en juxtaposant 6 bits, etc.

Chaque élève propose sur la Fiche 30 une liste libre codable sur 4 bits mais pas sur 3 bits (une telle liste doit comporter entre 9 et 16 éléments inclus) : les élèves proposent par exemple les 12 mois de l'année, les 10 chiffres 0 à 9, les 12 heures du cadran des montres, les 10 doigts des mains, les prénoms de leurs 9 cousins germains, 16 noms d'animaux de la ferme, etc.

Défi (par binôme)

L'enseignant demande aux élèves de trouver combien de bits il faut juxtaposer pour pouvoir coder les 26 lettres de l'alphabet.

Après quelques minutes, la mise en commun permet de conclure que 4 bits ne suffisent pas pour coder les 26 lettres de l'alphabet (codage de 16 éléments au maximum : 2x2x2x2 = 16), mais que 5 bits sont suffisants (codage de 32 éléments au maximum : 2x2x2x2x2 = 32).

Conclusion et trace écrite

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

- *On peut encoder les éléments d'une liste par une succession de « 0 ou 1 » aussi appelés bits. On parle alors d'un codage « binaire ».*
- *Plus on juxtapose de bits, plus on peut représenter d'éléments : 2 éléments au maximum avec un seul bit, 2x2=4 éléments au maximum en juxtaposant deux bits, 2x2x2=8 éléments au maximum en juxtaposant 3 bits, 2x2x2x2=16 éléments au maximum en juxtaposant quatre bits, 2x2x2x2x2=32 bits en juxtaposant cinq bits, … 2x2x2x2x2x2x2x2=256 éléments en juxtaposant huit bits, etc.*
- *Le codage binaire permet de représenter toutes sortes de données, notamment des nombres ou des caractères textuels.*
- *Un ordinateur représente n'importe quelle information en utilisant un codage binaire.*

Les élèves notent ces conclusions dans leur cahier de sciences tandis que l'enseignant complète l'affiche « Qu'est-ce que l'informatique ? ».

Prolongement

Un exercice en ligne permet de se réapproprier le concept d'énumération binaire.

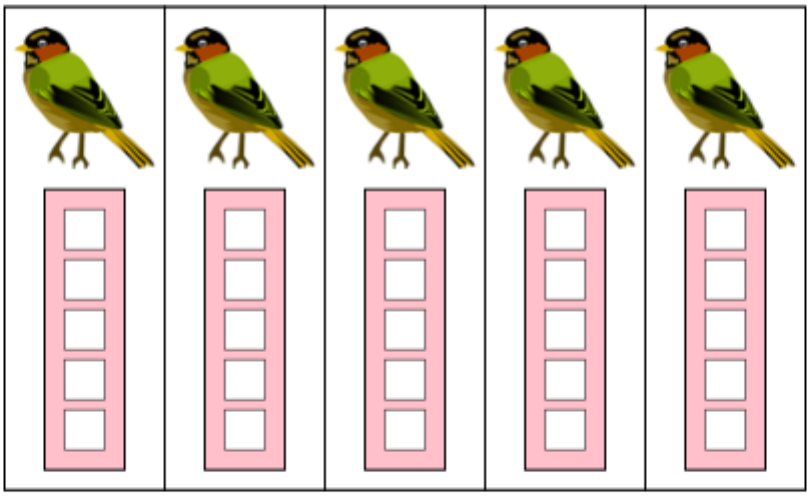
Marquage d'oiseaux

Version ★★ | Version ★★★ | Version ★★★★

Cliquez sur les cases pour dessiner des motifs différents sur les pattes des oiseaux.

Cliquez ensuite sur le bouton pour les laisser s'envoler.

Les oiseaux se mélangeront, et il vous faudra les remettre à leur place d'origine.



Laisser les oiseaux s'envoler | Effacer les marques

Prolongement (pour le cycle 4)

En cycle 4, on peut insister davantage sur une façon commode de retrouver toutes les combinaisons de 0 et de 1 sur, par exemple, 5 bits : si on demande aux élèves de retrouver les 32 combinaisons de 5 bits possibles, ils s'aperçoivent que c'est très difficile de tous les trouver et de ne pas proposer les mêmes plusieurs fois.
On peut alors suggérer une approche récursive :

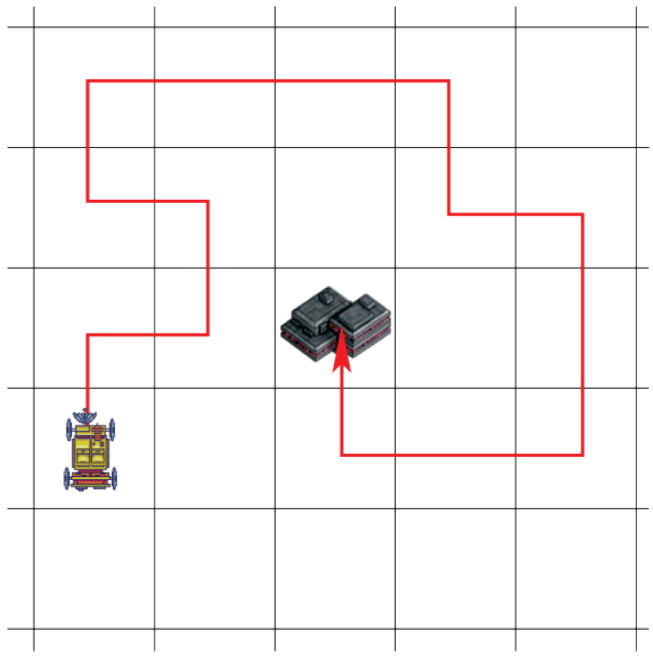
- Avec seulement 1 bit, on a la liste très facilement : 0, 1.
- Pour deux bits, il suffit de recopier deux fois la liste sur 1 bit : une fois en ajoutant un 0 devant, une fois en ajoutant un 1 devant : on obtient 00, 01, 10, 11
- Pour trois bits, on procède de même : on recopie deux fois la liste obtenue pour 2 bits, une fois en ajoutant un 0 devant, une fois en ajoutant un 1 devant : 000, 001, 010, 011, 100, 101, 110, 111.Et ainsi de suite pour 4 bits, 5 bits, etc., de proche en proche.

<< Séance III-1.2 | Séquence III-1 | Séance III-1.4 >>

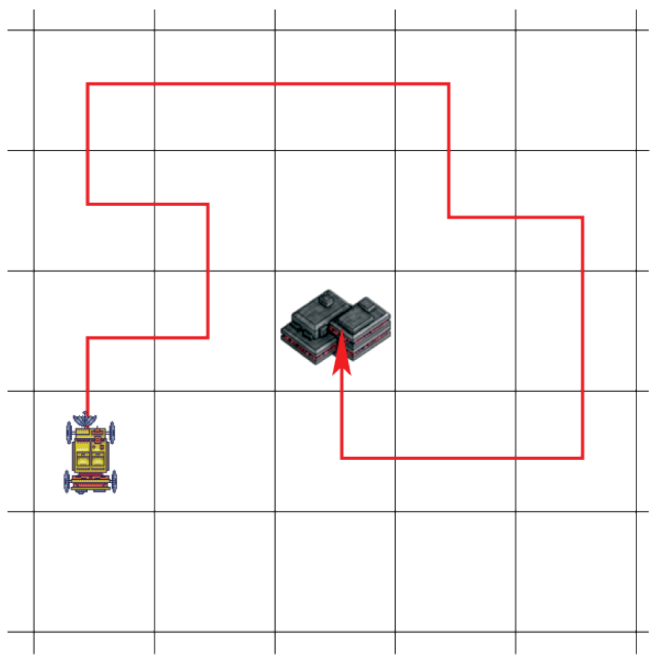
Source URL: http://www.fondation-lamap.org/fr/page/34516/1-2-3-codez-activites-cycle-3-seance-13-comment-coder-des-informations-en-binaire

FICHE 28
Piloter un véhicule à distance

Consigne : Écris une suite d'instructions permettant au véhicule de suivre case à case le parcours rouge jusqu'à la base.



Consigne : Écris une suite d'instructions permettant au véhicule de suivre case à case le parcours rouge jusqu'à la base.



FICHE 29
Encoder et décoder un message

Table de correspondance entre caractères utilisés dans les messages et nombres servant à les coder :

Caractère	A	B	C	D	E	F	G	H
Nombre	01	02	03	04	05	06	07	08

Caractère	I	J	K	L	M	N	O	P
Nombre	09	10	11	12	13	14	15	16

Caractère	Q	R	S	T	U	V	W	X
Nombre	17	18	19	20	21	22	23	24

Caractère	Y	Z	.	Espace
Nombre	25	26	27	28

Consigne 1 : utilise la table de correspondance fournie pour encoder le message destiné au poste de contrôle.

E	N	V	O	Y	E	Z		B	U	L	L	E	T	I	N		M	E	T	E	O	.	

P	R	O	L	O	N	G	A	T	I	O	N		S	O	R	T	I	E		S	A	U	F

	R	I	S	Q	U	E		T	E	M	P	E	T	E	.		M	E	R	C	I	.

Consigne 2 : utilise la table de correspondance fournie pour décoder la réponse du poste de contrôle.

01	16	16	18	15	03	08	05	28	03	25	03	12	15	14	05	28

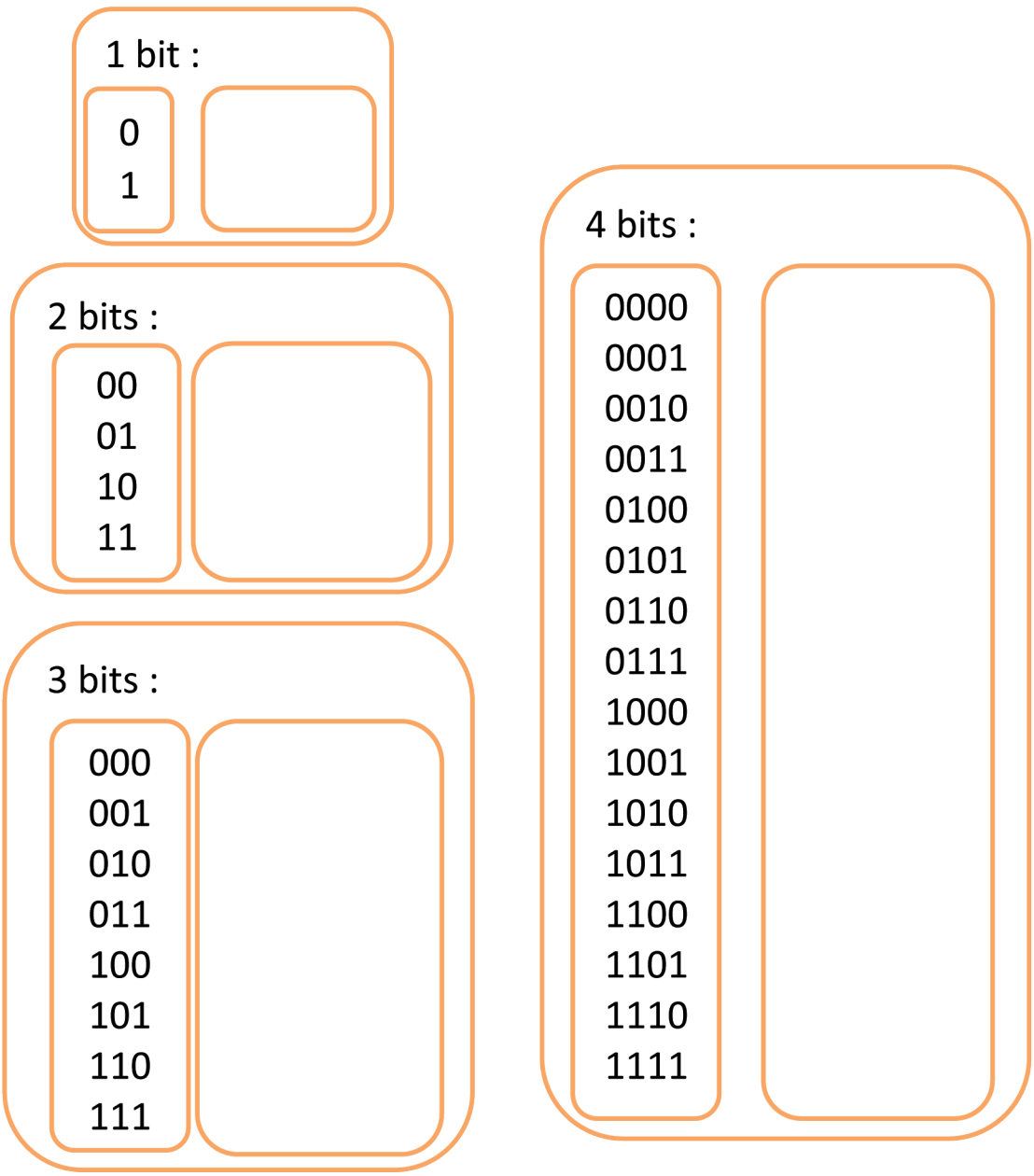
13	01	10	05	21	18	27	28	18	05	20	15	21	18	28	16	01	18	11	09	14	07	28

15	02	12	09	07	01	20	15	09	18	05	27	28	09	14	04	09	17	21	05	26	28

04	05	12	01	09	28	16	18	05	22	21	27

FICHE 30
Coder les éléments d'une liste en binaire

Consigne : dans chaque bulle vide, écris un exemple de liste dont tous les éléments peuvent être codés sur 1 bit, 2 bits, 3 bits ou 4 bits.



Défi : détermine combien de bits il faut juxtaposer au minimum pour pouvoir coder chacune des 26 lettres de l'alphabet.

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

1, 2, 3, codez ! - Activités cycle 3 - 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 III-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*, en raison de sa qualité remarquable, sa simplicité d'utilisation, sa gratuité, et 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…).

Nous proposons également une alternative à Scratch « seul » 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 178 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 systématique) ;
- Savoir lancer un programme en double-cliquant sur son icône ;
- 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-groupe

L'idéal est de disposer d'**un ordinateur pour 2 élèves** (3 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-groupe : la moitié de la classe programme pendant que l'autre moitié travaille à autre chose, en autonomie.

Préparer l'environnement de travail

Afin de gagner du temps, il est utile d'avoir préparé l'environnement de travail :

- *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 clé USB dédiée au groupe. Ce répertoire contiendra les fichiers nécessaires au projet (images à importer, sauvegarde des programmes…). Nous mettons sur ce site Web dédié au projet, tous les fichiers utiles, téléchargeables au fil des séances.
  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 à 3 heures sur son temps personnel 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 ! Le tableau récapitulatif ci-après liste les différentes étapes et, pour chaque étape, les différentes tâches à réaliser pour construire un jeu vidéo. Si l'enseignant choisit un autre scénario pour le jeu, il faudra bien entendu adapter cette progression.

Note : vous pouvez **tester le jeu** en ligne (ou en faire la démonstration aux élèves), en cliquant sur l'image ci-dessous. Récoltez de l'eau et de la nourriture (végétaux) tout en évitant les pièges !

À noter : dans la plupart des classes de cycle 3, réaliser l'ensemble du projet **nécessitera 6 ou 7 séances d'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.

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

Des repères de difficulté

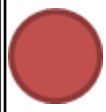
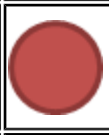
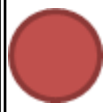
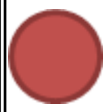
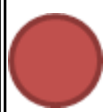
Pour se répérer dans ces différentes étapes et tâches, nous utiliserons un symbole inspiré des pistes de ski :

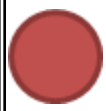
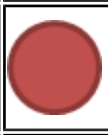
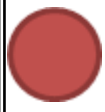
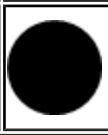
	Piste verte : facile *Tous les élèves y arrivent sans problème.*
	Piste bleue : moyenne *La plupart des élèves y arrivent seuls, certains ont besoin d'être un peu guidés.*

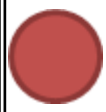
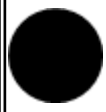
	Piste rouge : difficile *La plupart des élèves ont besoin d'être guidés (plus ou moins selon leur aisance).*
	Piste noire : très difficile *Tous les élèves ont besoin d'être guidés. Ces tâches sont facultatives.*

Listing des tâches

	Etape	Titre	Tâche	
	Etape 1	Découverte de l'environnement de programmation *Scratch*	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 : petits exercices (20 minutes)	
	Etape 2	Planter le décor, et sauvegarder son travail	Tâche 1 : changer le lutin (5 minutes)	
			Tâche 2 : changer l'arrière-plan (5 minutes)	
			Tâche 3 : enregistrer son programme *Scratch* (5 minutes)	
	Etape 3	Piloter le rover	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)				
	Etape 4	Récolter des ressources, gérer son score	Tâche 1 : importer une ressource (la glace) sous la forme d'un nouveau lutin	
			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)	
			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)	
	Etape 5	Activités branchées et débranchées pour mieux s'approprier certains concepts algorithmiques	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)				
	Etape 6	Éviter des obstacles, gérer son nombre de vies	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)	
	Etape 7	Mettre fin au jeu : « game over »	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)	
	Etape 8	Pimenter un peu le jeu	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 (15 minutes)	
			Tâche 5 : faire accélérer la tornade au fur et à mesure (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)			
	Etape 9	Prolongements possibles en *Scratch*	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.

<< Séquence III-1 Activités cycle 3 Séquence III-3 >>

Source URL: http://www.fondation-lamap.org/fr/page/34518/1-2-3-codez-activites-cycle-3-sequence-2-simuler-la-mission-dans-scratch

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

1, 2, 3, codez ! - Activités cycle 3 - Etape 2.1: Découverte de Scratch

Résumé	Les élèves découvrent *Scratch*, un environnement de programmation adapté à l'école primaire. Ils apprennent à lancer le programme et enchaîner quelques instructions simples.

Notions	« Machines » : • Les machines qui nous entourent ne font qu'exécuter des ordres (instructions) • En combinant des instructions élémentaires, nous pouvons leur faire exécuter des tâches complexes « Algorithmes » • Un algorithme est une méthode permettant de résoudre un problème. • Une boucle permet de répéter plusieurs fois la même action • Certaines boucles, dites « infinies », ne s'arrêtent jamais. • Certaines boucles, dites « itératives » sont répétées un nombre prédéfini de fois. « Langages » : • Pour donner des instructions à une machine, on utilise un langage de programmation, compréhensible à la fois par la machine et par l'être humain • *Scratch* est un environnement de programmation graphique, qui utilise un langage simple. • Un programme est l'expression d'un algorithme dans un langage de programmation • Certaines instructions ne s'exécutent qu'au déclenchement d'un évènement : on parle de programmation évènementielle. • Certaines instructions s'exécutent à la suite les unes des autres : on parle de programmation séquentielle. • L'exécution d'un programme est reproductible (si les instructions ne changent pas, ni les données à manipuler, le programme donne toujours le même résultat)

Matériel	Pour la classe • Un vidéo projecteur • Version agrandie de la Fiche 32 Pour chaque binôme • Un ordinateur connecté à Internet ou, en l'absence de connexion Internet de bonne qualité, un ordinateur sur lequel le logiciel *Scratch* a été préalablement installé . Pour chaque élève • Fiche 32
Lexique	Programme, script, lutin, instruction, événement
Durée :	1 heure

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.

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 vont devoir 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) sans expliquer comment fonctionne le programme, mais en faisant simplement une démonstration du jeu.

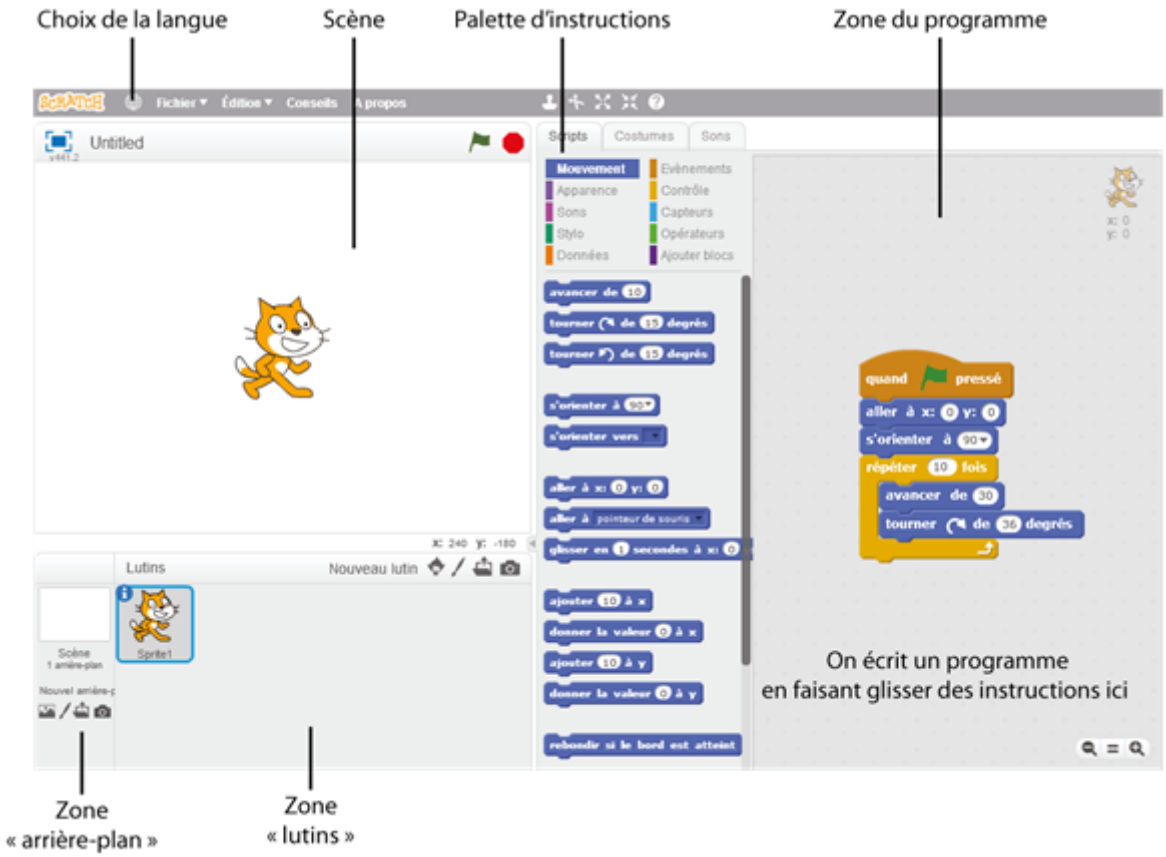
Notes pédagogiques :

- 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 et de leurs exigences).

Capture d'écran : version finale du jeu vidéo réalisé par l'enseignant
*(**NB :** un programme corrigé est disponibleici, mais encore une fois, il est important que l'enseignant tente lui-même de le réaliser !)*

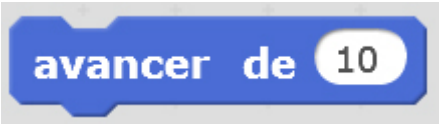
Tâche 1 : lancer Scratch et découvrir son interface (10 minutes)

Chaque binôme lance *Scratch* en double-cliquant sur son icône.
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* »).

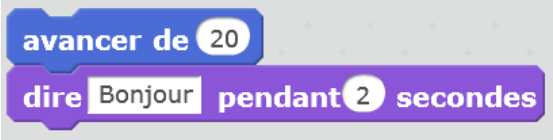


L'enseignant explique aux élèves que *Scratch* est un langage de programmation conçu spécifiquement pour apprendre à programmer. Lorsque l'on ouvre le logiciel, il y a un lutin à l'écran (un chat). On peut lui donner des instructions simples.

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 emboitant des instructions entre elles.



Si maintenant on veut que le chat fasse cela à 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 d'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 couleur (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) 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

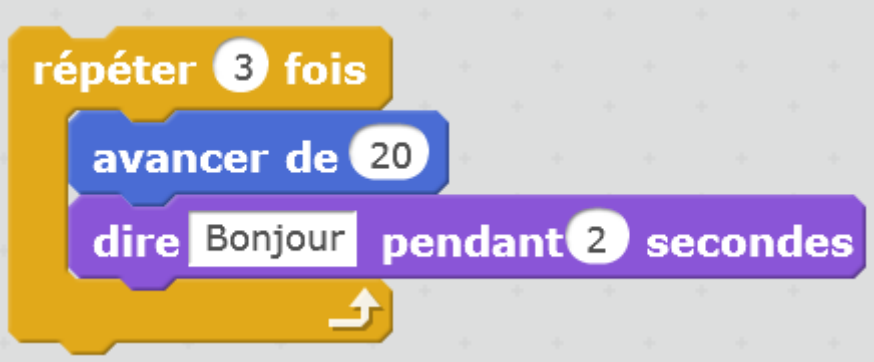
Certains élèves vont sans doute trouver l'astuce… mais pour la plupart, il faudra la leur montrer. Malgré tout, il est indispensable pour eux de voir cette instruction dès maintenant, car, à force de déplacer le chat, ils vont le faire sortir de l'écran et ne sauront pas comment le récupérer.

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.

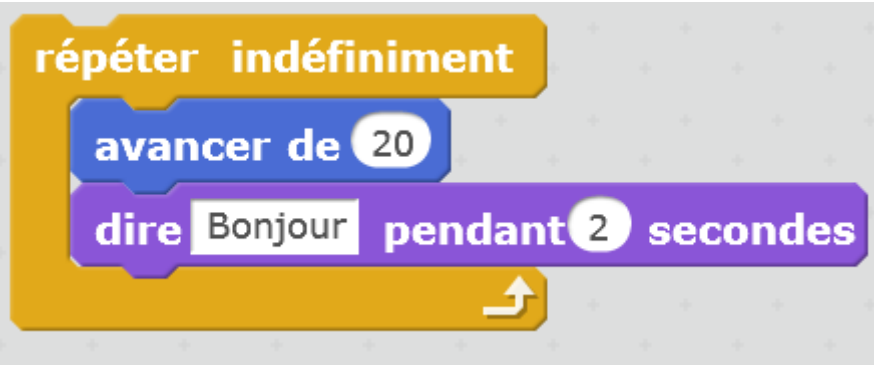
Exercice 5 : répéter 3 fois : faire avancer le chat de 20 et lui faire dire « bonjour »



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.

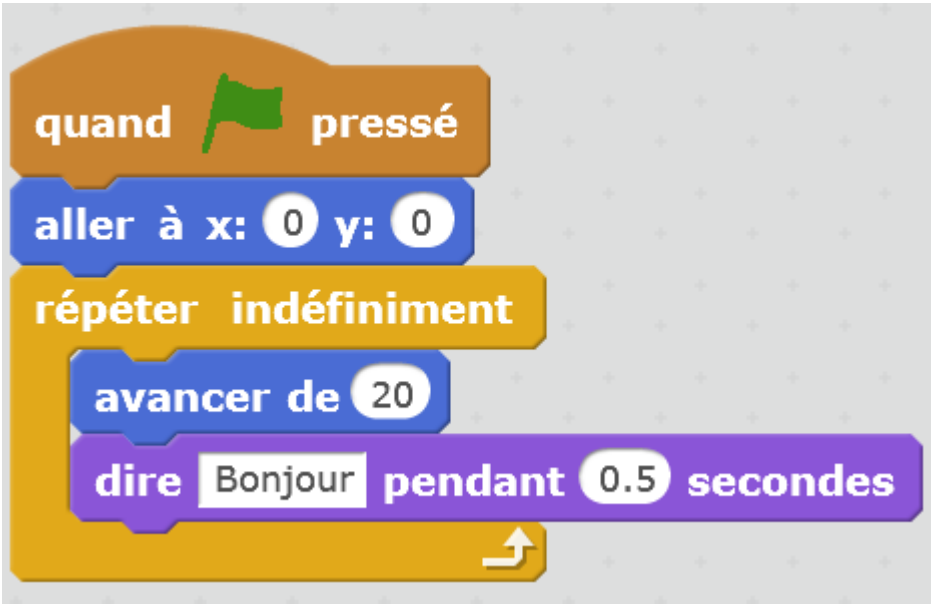
On voit le chat qui s'arrête 2 secondes entre chaque mouvement… pour raccourcir cette pause, il suffit de raccourcir la durée pendant laquelle il dit « bonjour » (si on écrit «0.5 » au lieu de 2, il ne s'arrêtera qu'1/2 seconde à chaque fois).

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 Scratch » 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, des capteurs, des variables, et des opérateurs.

Source URL: http://www.fondation-lamap.org/fr/page/34519/1-2-3-codez-activites-cycle-3-etape-21-decouverte-de-scratch

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

1, 2, 3, codez ! - Activités cycle 3 - Etape 2.2: 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	« 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 évènement : on parle de programmation évènementielle. • Certaines instructions s'exécutent à la suite les unes des autres : on parle de programmation séquentielle. • L'exécution d'un programme est reproductible (si les instructions ne changent pas, ni les données à manipuler, le programme donne toujours le même résultat)

Matériel	Pour la classe • Un vidéo projecteur • Version agrandie de la Fiche 32 Pour chaque binôme • Un ordinateur connecté à Internet ou, en l'absence de connexion Internet de bonne qualité, un ordinateur sur lequel le logiciel *Scratch* a été préalablement installé . Pour chaque élève • Fiche 32

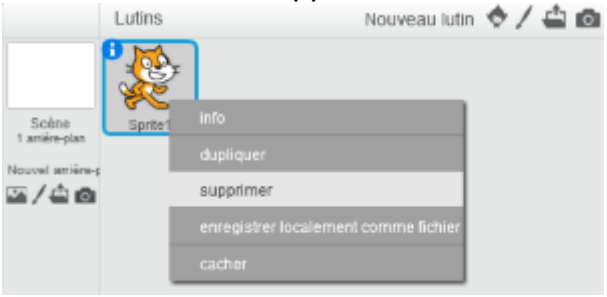
Notes pédagogiques :

- 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 ».



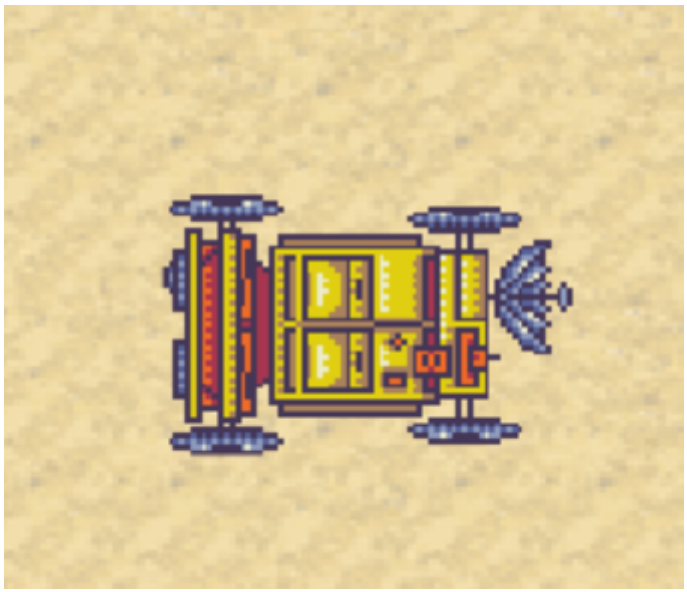
- Il y a 4 façons différentes de créer un nouveau lutin, accessibles depuis la barre d'outils « nouveau lutin » en bas à droite de la scène (nous mettons en gras la méthode que nous préconisons ici).

	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.

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 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 à ce 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 ».

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 apparait, et tout le reste qui disparait).

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 s'avère nécessaire. Chaque binôme pourra avancer à son rythme ; l'important étant d'arriver à un jeu jouable à la fin du projet.

Source URL: http://www.fondation-lamap.org/fr/page/34520/1-2-3-codez-activites-cycle-3-etape-22-planter-le-decor-et-sauvegarder-son-travail

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

1, 2, 3, codez ! - Activités cycle 3 - Etape 2.3: Piloter le rover

Résumé	Les élèves réalisent leur premier programme, leur permettant de piloter le rover à l'aide des flèches. Ils se familiarisent avec le système de coordonnées.

Notions	« Machines » : • Les machines qui nous entourent ne font qu'exécuter des ordres (instructions) • En combinant des instructions élémentaires, nous pouvons leur faire exécuter des tâches complexes « Algorithmes » • Un algorithme est une méthode permettant de résoudre un problème. • Une boucle permet de répéter plusieurs fois la même action • Certaines boucles, dites « infinies », ne s'arrêtent jamais. • Certaines boucles, dites « itératives » sont répétées un nombre prédéfini de fois. « Langages » : • Pour donner des instructions à une machine, on utilise un langage de programmation, compréhensible à la fois par la machine et par l'être humain • *Scratch* est un environnement de programmation graphique, qui utilise un langage simple. • Un programme est l'expression d'un algorithme dans un langage de programmation • Certaines instructions ne s'exécutent qu'au déclenchement d'un évènement : on parle de programmation évènementielle. • Certaines instructions s'exécutent à la suite les unes des autres : on parle de programmation séquentielle. • L'exécution d'un programme est reproductible (si les instructions ne changent pas, ni les données à manipuler, le programme donne toujours le même résultat)

Matériel	Pour la classe • Un vidéo projecteur • Version agrandie de la Fiche 32 Pour chaque binôme • Un ordinateur connecté à Internet ou, en l'absence de connexion Internet de bonne qualité, un ordinateur sur lequel le logiciel *Scratch* a été préalablement installé . Pour chaque élève • Fiche 32 • une photocopie de la Fiche 33

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)

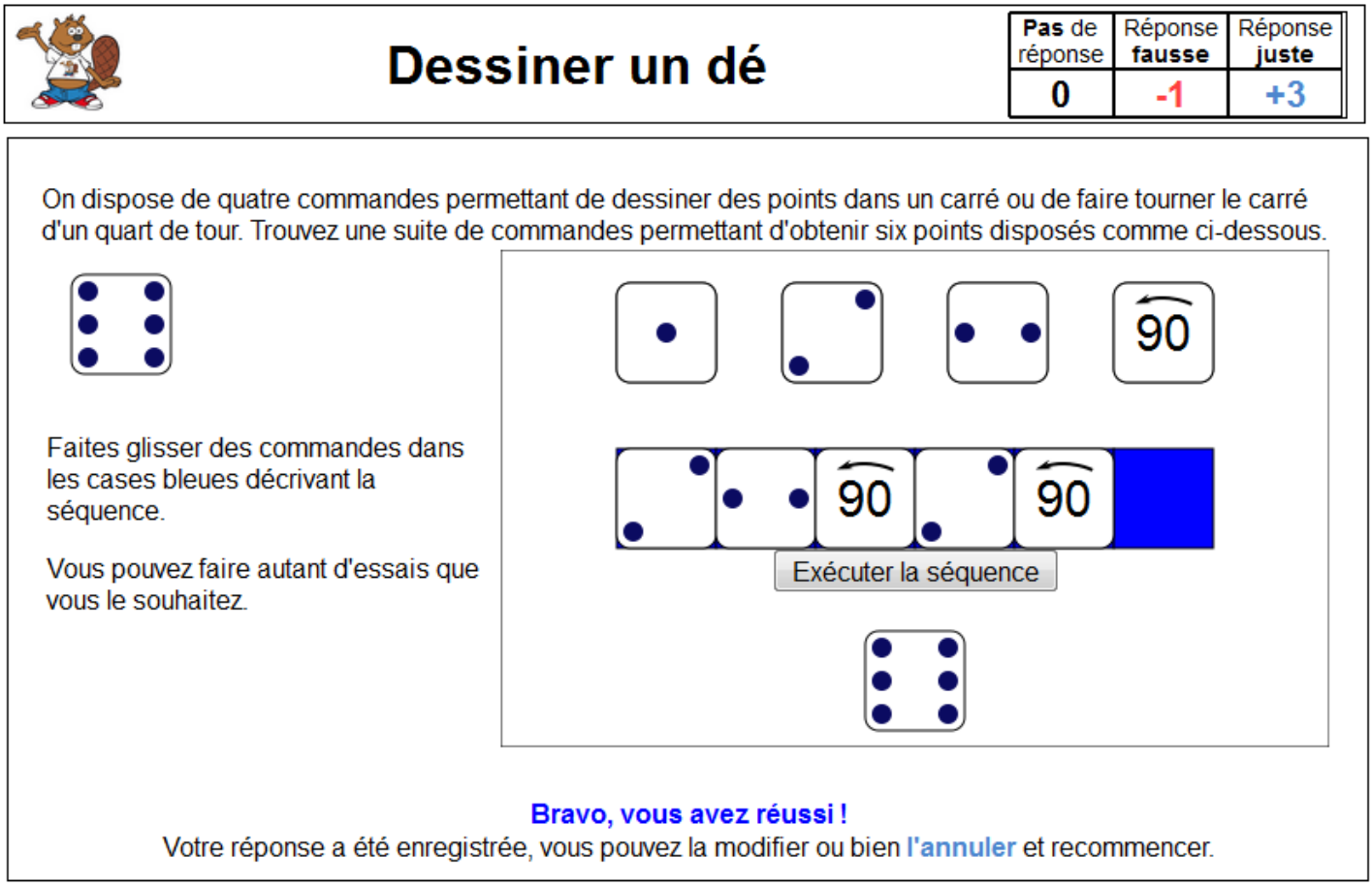
Les élèves savent déjà comment faire avancer le rover vers la droite… il suffit de lui dire d'avancer, puisque, par défaut, il est déjà orienté vers la droite. Le faire avancer vers la gauche est une tâche un peu plus difficile, car les élèves doivent d'abord demander au rover de s'orienter vers la gauche,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

> **Notes pédagogiques :**
>
> - 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

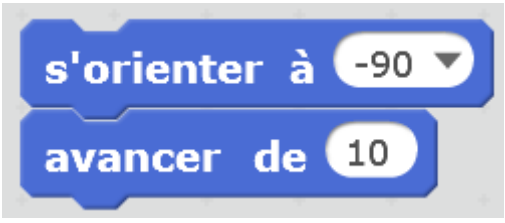
le haut de l'écran ; 90° désigne la droite, etc.

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

- Un exercice en ligne permet de se réapproprier la notion d'orientation. Attention, cet exercice n'utilise pas *Scratch*.



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 changer « 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 :

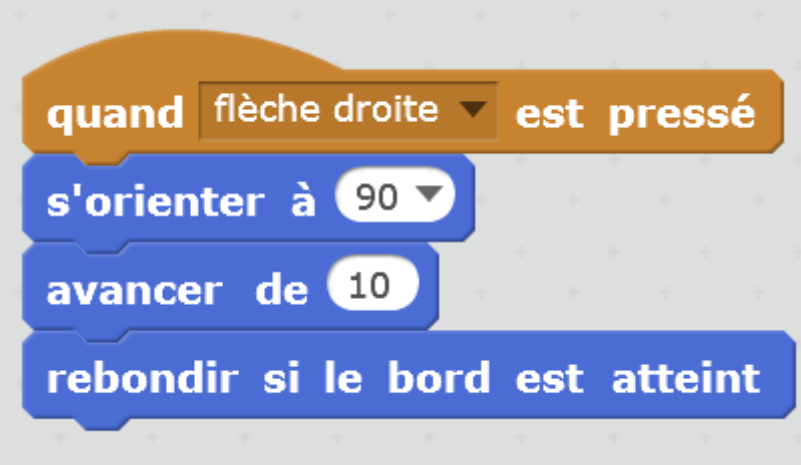
Notes pédagogiques :

- On remarque ici qu'on peut faire co-exister 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)

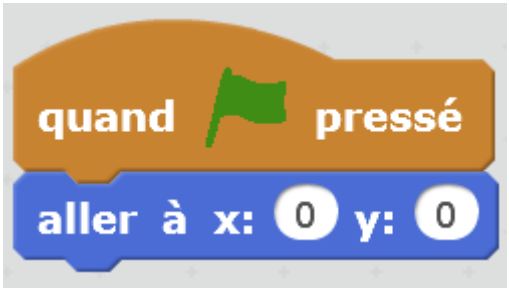
Tâche 4 : rebondir sur les bords (5 minutes)

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*.



Note pédagogique :

- 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, à chaque fois, de sauvegarder le travail réalisé !

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…) nécessiterons 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'affiche en bas à droite de la scène. Ceux-ci remarquent 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 des 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 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 :

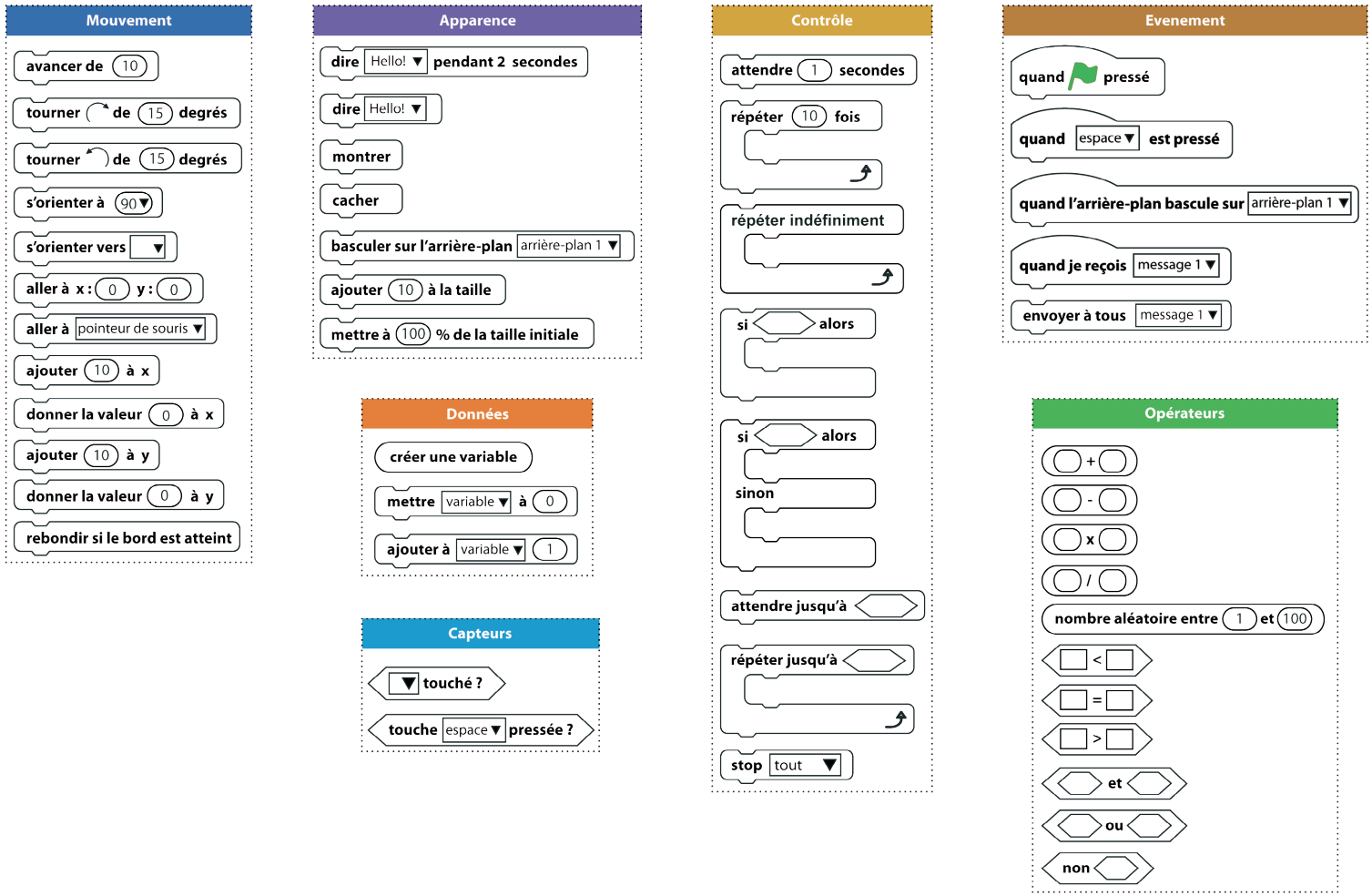
- *Un programme est un algorithme exprimé dans un langage particulier, appelé langage de programmation, compréhensible à la fois par la machine et par l'être humain.*
- *L'exécution d'un programme est reproductible (si les instructions ne changent pas, ni les données à manipuler, le programme donne toujours 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 à jour l'affiche « qu'est-ce que l'informatique ? ».

<< Etape III-2.2 Séquence III-2 Etape III-2.4 >>

Source URL: http://www.fondation-lamap.org/fr/page/34521/1-2-3-codez-activites-cycle-3-etape-23-piloter-le-rover

Quelques commandes utiles en Scratch



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

1, 2, 3, codez ! - Activités cycle 3 - Séquence 3 : Donner des nouvelles

	Séance	Titre	Résumé
	Séance 1	Comment transmettre une image ?	Les élèves doivent trouver comment transmettre une image à distance. Pour ce faire, ils découvrent qu'une image peut être représentée par une grille de pixels. Ils abordent la notion de résolution, en voyant qu'augmenter le nombre de pixels améliore la lisibilité de l'image mais en ralentit la transmission.
	Séance 2	Comment encoder une image en noir et blanc ?	Les élèves appliquent leurs découvertes de la séance précédente à l'encodage numérique d'images en noir et blanc. Ils visualisent tout d'abord un même fichier avec un logiciel éditeur de texte et avec un logiciel éditeur d'images, pour comprendre comment est effectué l'encodage. Puis ils encodent eux-mêmes un petit damier et le visualisent.
	Séance 3	(Optionnelle) Comment encoder une image en gris ou en couleurs ?	Les élèves approfondissent le travail de la séance précédente en abordant l'encodage numérique d'images en niveaux de gris et en couleurs.
	Séance 4	Comment assurer la confidentialité de nos échanges ?	Afin de protéger leurs communications, les élèves s'initient au chiffrement, à l'aide d'un algorithme simple (dit « de César »), consistant à décaler les lettes d'un message.

	Séance 5	(Optionnelle) Comment s'assurer que nos données sont bien transmises ?	Les élèves se rendent compte qu'il est possible de détecter et corriger des erreurs introduites lors du stockage ou du transfert d'un fichier, à condition de rajouter de l'information supplémentaire et de bien la choisir. Ils deviennent ainsi capables d'exécuter un « tour de magie ».

<< Séquence III-2 Activités cycle 3

Source URL: http://www.fondation-lamap.org/fr/page/34528/1-2-3-codez-activites-cycle-3-sequence-3-donner-des-nouvelles

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 3.1: Comment transmettre une image ?

Résumé	Les élèves doivent trouver comment transmettre une image à distance. Pour ce faire, ils découvrent qu'une image peut être représentée par une grille de pixels. Ils abordent la notion de résolution, en voyant qu'augmenter le nombre de pixels améliore la lisibilité de l'image mais en ralentit la transmission.
Notions	« Information » • une image peut être représentée sous la forme d'une grille de carrés appelés pixels.
Matériel	Pour la classe • De nombreux aimants pour fixer les productions au tableau ou de la patafix. Par groupe (4 groupes différents : A, B, C et D) • Loupes à main ou loupes binoculaires • Journaux imprimés • Si l'on n'a pas de loupes, morceaux d'affiches publicitaires grand format • Image A de la Fiche 39 en autant d'exemplaires que d'élèves dans le groupe A ; idem pour les images B, C et D dans les trois autres groupes • Fiche 40 imprimée ou photocopiée sur transparent ou sur papier calque et prédécoupée en 3 grilles. Prévoir un peu plus de grilles que d'élèves. Par élève • Papier calque (1/4 de feuille A4) et crayon à papier bien taillé, ou papier transparent et marqueur fin indélébile • Scotch ou trombones

Lexique	Image, pixel, résolution
Durée :	1h30

Situation déclenchante

L'enseignant explique que les explorateurs veulent photographier leurs découvertes, et transmettre les photographies à la base. « *Comment pourrait-on transmettre des photographies sur de longues distances ?* ». Les élèves proposent plusieurs possibilités : coursier, pigeon voyageur, facebook, scanner ou e-mail.
Que les élèves pensent ou non à la numérisation de la photographie, l'enseignant leur pose alors cette question : «*Mais au fait, qu'est-ce qu'une image ?* »

Recherche : qu'est-ce qu'une image ? (par groupes)

L'enseignant distribue les journaux imprimés à chaque groupe. Il demande aux élèves de réfléchir à ce qui constitue une image. Les élèves évoquent les matériaux : papier, carton, encre. Quand ce mot est prononcé, l'enseignant distribue les loupes. «*Pouvez-vous me dire comment est distribuée l'encre sur les images ? Quelle est sa couleur ?* »
Très vite, les élèves vont repérer que les impressions sur du journal sont composées d'une multitude de petits points, et que les couleurs de ces points sont en fait très limitées. L'enseignant introduit le terme « pixel » (de l'anglais « picture element ») et aide à formuler une conclusion qui peut ressembler à ceci : «*Une photographie est constituée de petits point colorés, les pixels. De loin, on ne voit plus les pixels, mais une image qui semble continue.* ».

Observation de magazines à l'aide loupes binoculaires. Des loupes à main peuvent tout à fait suffire, si l'on choisit un papier à impression grossière, comme du papier journal. Classe de 6è de Fatima Rahmoun (Paris)

Notes pédagogiques

- Selon la qualité des loupes et la finesse de l'impression (imprimantes lasers vs imprimantes à jet d'encre, par exemple), les pixels chevauchants risquent d'être difficiles à discerner dans le cas des magazines et photographies. Voilà pourquoi nous conseillons le papier journal. Ne pas hésiter toutefois à vérifier avec les

loupes de la classe, avant la séance, si les pixels sont visibles sur d'autres supports que le papier journal.
- À défaut de loupe, on peut observer les pixels à l'œil nu sur les affiches grand format des spots publicitaires. Les pixels ne sont plus visibles lorsque l'on s'éloigne suffisamment des affiches.
- Lors de la séance suivante, on observera de nouveau des pixels, mais sur écran d'ordinateur, pour arriver à la même conclusion que les images sont constituées de petites taches discontinues de différentes teintes (voir note scientifique ci-dessous).

Notes scientifiques

- Pour des raisons techniques, les pixels des écrans sont de petits carrés juxtaposés (pour être plus précis, sur les écrans couleur, chaque pixel carré est en fait constitué de trois sous-pixels de forme rectangulaire juxtaposés de gauche à droite : un sous-pixel rouge, un sous-pixel vert et un sous-pixel bleu, voir ci-dessous). Dans le cas des supports papier, les disques colorés peuvent, eux, se chevaucher (le blanc du papier sert lui aussi dans la recomposition des couleurs).
- La couleur des pixels dépend grandement du support utilisé. Sur un écran d'ordinateur, de tablette ou de smartphone, les pixels existent en Rouge, Vert et Bleu (on parle donc de mode RVB). Sur les images imprimées en quadrichromie, les couleurs sont Cyan, Magenta, Jaune et Noir (on parle alors de mode CMJN). Des impressions en bichromie se contentent de deux couleurs d'encre complémentaires (bleu et orange, par exemple). La combinaison de ces quelques couleurs permet de reconstituer une grande diversité de sensations colorées.

Exercice : combien de pixels faut-il pour notre image ? (par binômes)

L'enseignant remet cette conclusion dans son contexte : «*Pour transmettre une image, il suffit d'en transmettre tous les pixels, un par un.* » Il leur propose alors un exercice pour qu'ils puissent s'approprier et approfondir cette notion : ils vont pixelliser une image, c'est-à-dire la remplacer par une grille de pixels.
Il scinde la classe en quatre groupes distincts, qui chacun pixelliseront une des quatre images (A, B, C ou D) de la Fiche 39. À chaque élève, il distribue :

- un exemplaire de l'image de son groupe, en précisant bien que chaque groupe ne doit pas montrer son image aux autres groupes. ;
- la grille 1 de la Fiche 40, imprimée sur papier transparent ou sur papier calque ;
- du scotch ou des trombones.

Les élèves doivent superposer l'image et la grille en faisant correspondre les repères en forme de « L », assembler les 2 couches à l'aide de scotch ou de trombones, puis noircir intégralement les cases par lesquelles passe le trait de l'image en arrière-plan.
Eventuellement, l'enseignant peut utiliser l'exercice en ligne du Cycle 2 pour bien ancrer la consigne de la pixellisation.

Classe de 6è de Fatima Rahmoun (Paris)

Lorsque les élèves ont terminé, ils notent la lettre de l'image (A, B, C ou D) et le numéro de la grille (1 pour le moment) sur leur production. Chaque groupe remet à l'enseignant un ou deux exemplaires de l'image ainsi pixellisée avec 64 pixels (choisir des coloriages dont les cases coloriées sont bien sombres). L'enseignant accroche les productions en 4 colonnes (« Image A », « Image B », etc.) en en laissant la place pour tracer ultérieurement 3 lignes (qui seront nommées « grille 1 », « grille 2 » et « grille 3 »). Les images pixellisées avec la grille 1 ne sont pas reconnaissables.

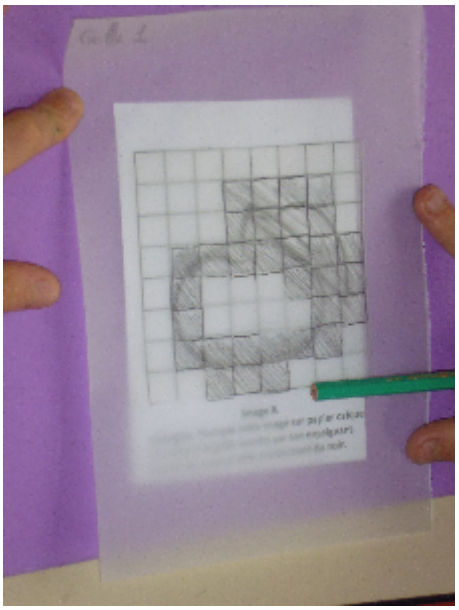


Image A pixellisée avec la grille 1. Classe de CM2 d'Anne-Marie Lebrun (Bourg-la-Reine)

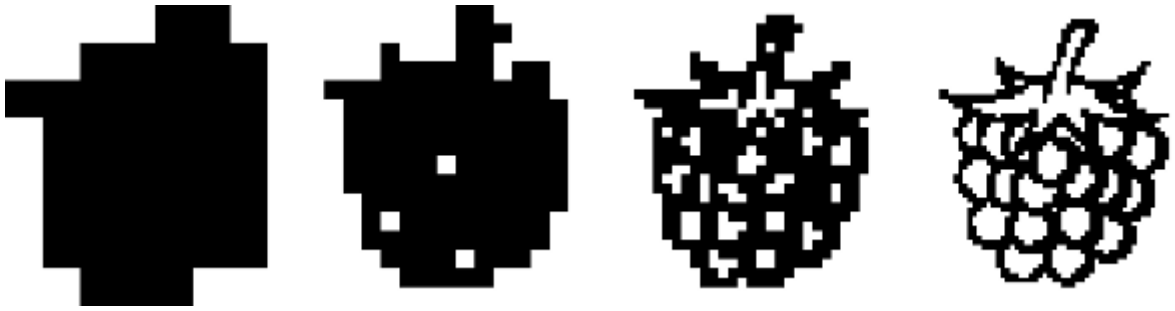
« Comment peut-on améliorer ces images pour en reconnaître le contenu ? » Les élèves auront deux idées : soit on utilise des nuances de gris plutôt que du noir ou du blanc exclusivement, soit on ajoute plus de pixels. La première option, si elle émerge spontanément, sera notée au tableau, et approfondie lors d'une prochaine séance (Séance III-3.3).
Pour explorer la seconde proposition, l'enseignant distribue les grilles plus fines de la Fiche 40 : grille 2 (512 pixels), et grille 3 (2048 pixels) Chaque binôme doit pixelliser l'image de son groupe une nouvelle fois, soit avec la grille 2, soit avec la grille 3.

(Bourg-la-Reine)

Notes pédagogiques

- Distribuer les grilles 2 et 3 en fonction de la rapidité des élèves au sein d'un même groupe, pour ne pas allonger la séance plus que nécessaire. Chaque élève doit avoir fait l'exercice au moins une fois.
- Une organisation alternative consiste à distribuer directement les grilles 1, 2 et 3 au sein des groupes, plutôt que de faire utiliser d'abord la grille 1 par tous. Cette astuce fait gagner ¼ d'heure mais la discussion sur la façon d'améliorer le premier résultat ne peut avoir lieu.
- Prévoir de répéter les consignes « coloriez les cases entièrement ou pas du tout » et « coloriez entièrement les cases sous lesquelles passe le trait noir de l'image » un grand nombre de fois. Ne pas faire l'économie d'une démonstration au tableau de la façon de faire.

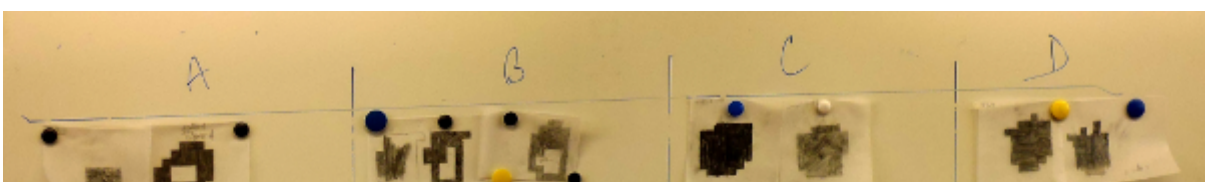
L'enseignant demande aux binômes utilisant la Grille 2 d'afficher leur résultat sous les images précédentes de leur groupe. Si les élèves des trois autres groupes pensent reconnaitre ce qui est figuré sur l'image, l'enseignant la sous-titre avec leur interprétation (pomme ? pêche ? poire ?). Puis il invite les binômes utilisant la Grille 3 et note les nouvelles interprétations.



Ci-dessus, de gauche à droite, l'image D pixellisée avec les grilles 1, 2 et 3. Tout à droite, une pixellisation avec une grille de 64x64= 4096 pixels (trop longue à utiliser en classe).

Mise en commun

L'enseignant demande aux élèves si l'ajout de pixels est une réponse efficace au problème posé (comment rendre l'image intelligible malgré la pixellisation). Il introduit alors le terme « résolution » : «*Quand on augmente le nombre de pixels, on augmente la résolution de l'image, et on reconnait mieux ce qui est dessiné.* ».
Grâce à la comparaison des images pixellisées avec différentes résolutions, il peut nuancer le besoin en résolution. Certaines images étaient reconnaissables dès l'utilisation de la Grille 2, pour d'autres il a fallu attendre le traitement avec la Grille 3. Pourtant l'enseignant rappelle qu'il faudra transmettre tous les pixels un par un à la base pour que l'image puisse être reconstituée. Il souligne donc le nécessaire compromis entre résolution et facilité de transmission : «*Si nous avions des moyens limités, de quelle résolution pourrions-nous nous contenter ?* » En fonction des images pixellisées, la classe débat et choisit une résolution de compromis : par exemple la résolution la plus grossière qui permette d'identifier au moins 3 des 4 sujets, ou bien la résolution qui permet de différencier (à défaut d'identifier) les 4 sujets sans ambiguïté.



Lebrun (Bourg-la-Reine)

Conclusion et trace écrite

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

- *Une image est composée de pixels.*
- *Pour transmettre une image, il suffit de transmettre tous ses pixels un par un.*
- *Plus on utilise de pixels, plus l'image pixellisée est fidèle à l'originale, mais plus elle occupe de place mémoire et plus elle est longue à transmettre.*

Les élèves notent ces conclusions dans leur cahier de sciences. L'enseignant met à jour la portion « Information » de l'affiche intitulée « Qu'est-ce que l'informatique ? ».

Prolongements

- La pixellisation d'images peut donner lieu à des applications artistiques. Nous pouvons citer par exemple le « post-it art » (cf. exemples ici :http://www.postitwar.com) pour réaliser des affiches ou décorer des murs avec des post-it© tenant lieu de pixels. Cela peut être une bonne occasion d'ouverture vers l'histoire de l'art (pointillisme en peinture). Voir aussi l'activité proposée au cycle 2 : Séance II-1.4 et Fiche 18.
- Ce travail peut être poussé jusqu'à la photomosaïque. Dans cet art, chaque pixel est en fait… une image à part entière. En vue rapprochée, on peut profiter des détails de la myriade de photos miniatures, tandis que de loin on découvre une toute autre image. Par exemple, les photos miniatures peuvent être celles des visages des élèves, et l'image globale peut-être un panorama de l'école, un animal fantastique, un paysage, etc. Des logiciels gratuits, tels qu'AndreaMosaic (http://www.andreaplanet.com/andreamosaic) permettent de créer des photomosaïques.

Séquence III-3 Séance III-3.2 >>

Source URL: http://www.fondation-lamap.org/fr/page/34529/1-2-3-codez-activites-cycle-3-seance-31-comment-transmettre-une-image

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 3.2: Comment encoder une image en noir et blanc ?

Résumé	Les élèves appliquent leurs découvertes de la séance précédente à l'encodage numérique d'images en noir et blanc. Ils visualisent tout d'abord un même fichier avec un logiciel éditeur de texte et avec un logiciel éditeur d'images, pour comprendre comment est effectué l'encodage. Puis ils encodent eux-mêmes un petit damier et le visualisent.
Notions	« Information » • Une image peut être représentée sous la forme d'une grille de carrés appelés pixels. • Un ordinateur représente n'importe quelle information par un code qui n'utilise que 2 symboles, 0 et 1, appelés bits : c'est le code binaire. • En noir et blanc, chaque pixel peut être représenté par un unique bit.
Matériel	Par binôme • Ordinateur avec logiciel éditeur de texte basique (Bloc-notes par exemple) et un logiciel éditeur d'image basique et dossier accessible aux élèves contenant les fichiers III-3.2_recherche_NB et III-3.2_defi_NB_vierge. • Loupe • vidéoprojecteur Pour la classe • Fiche 41 projetée au tableau
Lexique	Image, pixel, encodage
Durée :	1 heure

Note pédagogique

- Nous recommandons fortement d'utiliser l'éditeur d'images XnView dans sa version minimale, car il cumule les avantages suivants :
  - Il est gratuit pour un usage éducatif ;
  - Il lit correctement les fichiers de formats pbm (et pgm ou ppm pour la Séance III-3.3) ;
  - Son interface est simple;
  - Il permet de zoomer sans saisie de texte à l'aide des loupes « + » et « - » ;
  - Il permet de réactualiser une image déjà ouverte (commande « Réouvrir » du menu « Fichier » ou raccourci clavier « Ctrl R »), ce qui facilite grandement les essais/erreurs des élèves.
- XnView peut être téléchargé ici : http://www.xnview.com/fr/xnview

Situation déclenchante

L'enseignant reprend au tableau l'image de la pomme (Fiche 39, image A) pixellisée à la séance précédente sur 256 pixels (grille 2) : «*nos explorateurs veulent transmettre cette image à la base, sans se déplacer. Comment peuvent-ils faire ?* ».
Le débat qui s'ouvre permet d'énoncer des moyens, des concepts très variés. L'enseignant note au tableau les différentes idées. En particulier, plusieurs pistes méritent d'être creusées : «*il y a des pixels blancs et des pixels noirs* », «*on peut décrire chaque pixel comme en morse avec une lampe-torche* ».
Les propositions permettront de rappeler la notion de codage binaire (0 pour les pixels noirs, 1 pour les pixels blanc) si cette notion a été abordée à la séquence III-1. «*Imaginez que les explorateurs ne puissent envoyer qu'un texte simple (un SMS) pour décrire cette image.* »
Petit à petit émerge l'hypothèse qu'un texte composé de 0 et de 1 doit suffire à décrire l'image.

Observation : comprendre l'encodage d'une image en noir et blanc (collectivement puis par binômes)

Afin de vérifier si un texte rempli de 0 et de 1 peut suffire à représenter une image pixellisée en noir et blanc, l'enseignant montre aux élèves comment ouvrir le fichier texte III-3.2_recherche_NB.pbm avec un éditeur de texte minimaliste (Bloc-notes, sous Windows, par exemple). Le contenu du fichier est le suivant :

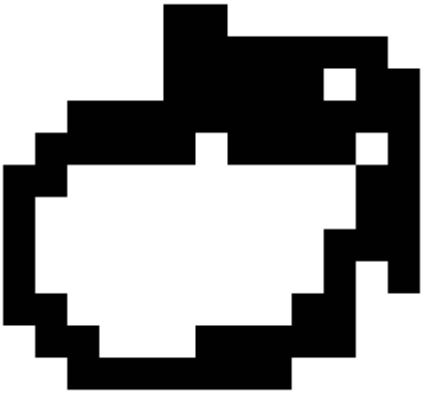
```
P1
16 16
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0
0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 1 1 1 1 1 0 1 1 1 1 0 1 0
0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0
0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0
0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0
```

0 0 1 0 0 0 0 0 0 0 0 1 0 1 0
0 0 1 1 0 0 0 0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0 1 1 1 1 1 0 0 0
0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Il explique que ce fichier est une version numérique de l'image de la pomme pixellisée avec 16x16 pixels. Il demande si la grille de 0 et de 1 semble effectivement dessiner la pomme : c'est le cas, on reconnait un peu la feuille et le contour, dessinés par des « 1 ». Si les élèves ne sont pas convaincus, on peut remplacer automatiquement tous les « 0 » par des « . » dans le bloc-notes (par « Ctrl H »), la pomme apparaît alors très clairement (annuler aussitôt cette modification par une commande « Ctrl Z »).

L'enseignant demande s'il y a d'autres choses que cette grille de 0 et de 1 dans le fichier. Les élèves s'interrogent sur les 2 premières lignes du fichier et peuvent faire l'hypothèse que 16 et 16 correspondent aux dimensions de la grille de pixels. L'enseignant complète en expliquant que ces premières lignes définissent le format des données écrites dans ce fichier : « P1 » désigne le codage binaire (des 0 et des 1 exclusivement), et la ligne « 16 16 » annonce qu'un tableau de 16 colonnes et 16 lignes de données va suivre. Effectivement, il y a 16x16=256 bits (des 0 ou des 1) inscrits dans le corps du fichier (description de la grille de pixels proprement dite).

L'enseignant montre ensuite comment visualiser ce même fichier, non plus avec un éditeur de texte, mais avec un éditeur d'images, comme XnView (cf note pédagogique en début de séance). La pomme apparaît en tout petit, il faut zoomer fortement pour la voir effectivement. La classe constate, par comparaison entre la grille de 0 et de 1 et l'image de la pomme, que les pixels noirs sont encodés par « 0 » et les pixels blancs par « 1 ».



Les enfants appliquent immédiatement ce qui a été montré par l'enseignant : ouverture du fichier d'une part avec l'éditeur de texte et d'autre part avec l'éditeur d'image (voir partie haute de la Fiche 41). Ils observent la pomme miniature à la loupe sur l'écran avant d'avoir zoomé puis zooment autant que nécessaire.

Une synthèse collective est effectuée concernant le format du fichier : P1 sur la première ligne, le nombre de colonnes puis le nombre de lignes sur la deuxième, puis une grille de 0 (pour le blanc) et de 1 (pour le noir) de dimension adéquate sur les lignes suivantes.

Notes scientifiques

- L'extension pbm du fichier signifie «*portable bit map* » ou « carte transportable binaire ».
- Si, dans la grille de pixels, on indique des valeurs différentes de 0 et 1, elles sont interprétées modulo 2 : 0 donne du blanc, 1 donne du noir, 2 (puis toutes les valeurs paires) donne de nouveau du blanc, 3 (puis toutes les valeurs impaires) donne du noir, etc.

Défi : encodage numérique d'un damier (par binômes)

L'enseignant lance aux élèves le défi suivant : ils doivent créer avec l'éditeur de texte un fichier représentant un petit damier noir et blanc de 5 lignes et 5 colonnes, avec des cases noires aux angles. Le fichier, vide pour le moment, s'appelle III-3.2_defi_NB_vierge.pbm.
Puis ils doivent sauvegarder le fichier et l'ouvrir avec l'éditeur d'image, sans fermer l'éditeur de texte, et appeler l'enseignant lorsqu'ils ont, côte à côte, d'une part le fichier ouvert avec l'éditeur de texte, et d'autre part le damier visible avec l'éditeur d'image.

L'encodage que les élèves doivent proposer est le suivant :

```
P1
5 5
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
```

Conclusion et trace écrite

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

- *On peut représenter les pixels d'une image par des nombres.*
- *Chaque pixel d'une image en noir et blanc est représenté soit par un 0 (pixel blanc), soit par un 1 (pixel noir).*
- *Pour transmettre une image en noir et blanc, on l'encode puis on transmet la version encodée, que l'on décode à l'arrivée pour reconstituer l'image.*

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

<< Séance III-3.1 Séquence III-3 Séance III-3.3 >>

Source URL: http://www.fondation-lamap.org/fr/page/34530/1-2-3-codez-activites-cycle-3-seance-32-comment-encoder-une-image-en-noir-et-blanc

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 3.3 (optionnelle): Comment encore une image en gris ou en couleurs ?

Résumé	Les élèves approfondissent le travail de la séance précédente en abordant l'encodage numérique d'images en niveaux de gris et en couleurs.
Notions	« Information » • Une image peut être représentée sous la forme d'une grille de carrés appelés pixels. • Plus on juxtapose de bits, plus on peut représenter d'éléments différents. • en niveau de gris, chaque pixel peut être représenté par plusieurs bits • en couleur, chaque pixel peut être représenté par 3 nombres (sur un ou plusieurs bits) représentant une quantité de rouge, de vert et de bleu. • Plus on juxtapose de bits, plus on peut représenter d'éléments différents.
Matériel	Par binôme • Ordinateur avec un traitement de texte basique (Bloc-notes par exemple) et un logiciel éditeur d'image basique (XnView par exemple) • Répertoire accessible aux élèves et contenant les fichiers III-3.3_recherche_gris, III-3.3_defi_gris_vierge, III-3.3_recherche_couleur et III-3.3_defi_couleur_preremplie. • Loupe à main Pour la classe • Vidéoprojecteur • Fiche 42 et Fiche 43 projetées au tableau

Lexique	Image, pixel, codage
Durée :	1h30, éventuellement en 2 fois 45 min

Situation déclenchante

L'enseignant fait ré-émerger les conclusions de la séance précédente : on peut encoder une image en noir et blanc par une grille de pixels, en indiquant « 0 » pour un pixel blanc et « 1 » pour un pixel noir.
Si les élèves avaient proposé à la Séance III-3.1 d'améliorer la représentation des images de fruits en coloriant plus ou moins fortement les cases, l'enseignant revient sur cette proposition qu'il avait notée : «*vous aviez eu l'idée, lorsque nous avions pixellisé les images de fruits, de ne pas faire seulement des cases noires et blanches, mais aussi des cases grises.* ».
Que cette proposition ait été faite ou pas par les élèves, l'enseignant ouvre ensuite une image en niveaux de gris à l'aide de l'éditeur d'image (fichier III-3.3_recherche_gris), et zoome suffisamment pour que les pixels soient visibles. Il demande si cette image comporte seulement des pixels noirs et blancs. Les élèves constatent qu'il y a des pixels gris clairs, gris moyen, gris foncé, en plus des pixels noirs et blancs, ce qui amène à la première activité de recherche.

Recherche : encodage d'une image en nuances de gris (par binômes et collectivement)

L'enseignant demande si des 0 et des 1 vont suffire pour encoder les pixels noirs, blancs et gris plus ou moins clair. Les élèves répondent que non, et proposent de mettre aussi d'autres chiffres que le 0 et le 1. L'enseignant demande aux élèves de chercher comment cette image a été encodée (ils ont accès au fichier III-3.3_recherche_gris et peuvent l'ouvrir avec les logiciels de leur choix). En cas de difficulté, l'enseignant peut suggérer d'ouvrir ce fichier d'une part avec un logiciel éditeur de texte et d'autre part avec un logiciel éditeur d'image (voir partie haute de la Fiche 42).

Après quelques minutes, la synthèse est faite collectivement. Les élèves auront constaté, en comparant le fichier ouvert avec l'éditeur de texte d'une part et avec l'éditeur d'images d'autre part :

- qu'il y a une indication P2 sur la première ligne, et non plus P1 (l'enseignant précise que cela indique que la suite sera encodée en niveaux de gris).
- qu'il y a, sur la deuxième ligne, une indication du nombre de lignes et du nombre de colonnes (ici 15 15), comme c'était le cas pour l'encodage des images en noir et blanc.
- qu'il y a une ligne supplémentaire contenant l'indication 7, justement le chiffre maximum apparaissant dans la grille de pixel qui vient ensuite.
- que la grille de pixels comporte des chiffres entre 0 et 7, c'est-à-dire 8 chiffres différents (et non plus seulement des 0 et des 1 comme c'était le cas pour un encodage en noir et blanc).
- que plus le chiffre est petit, plus le pixel correspondant est sombre (pixels noirs encodés par 0, pixels gris de plus en plus clairs encodés par les chiffres 1 à 6, pixels blancs

encodés par 7). C'était le contraire en noir et blanc (0 correspondait alors aux pixels blancs).

Si certaines de ces observations n'ont été faites par aucun binôme, l'enseignant guide la classe. Si des interrogations subsistent, l'enseignant peut encourager les élèves à procéder à des modifications du fichier dans l'éditeur de texte, et à actualiser l'image dans l'éditeur d'images (Fichier/Rouvrir), pour voir ce qui est alors modifié.

Notes scientifiques

- L'encodage en niveaux de gris se fait classiquement sur un nombre de nuances correspondant à une puissance de 2 (par exemple 8, 16, 32 ou 256 niveaux de gris), ce qui correspond à un encodage de chaque pixel sur respectivement, 3 bits, 4 bits, 5 bits et 8 bits) (voir Séance III-1.3).
- L'extension pgm des fichiers image signifie « portable grayscale map » ou « carte transportable en nuances de gris ».

Défi : encoder un dégradé de gris(en binômes)

L'enseignant lance aux élèves le défi suivant : ils doivent créer avec l'éditeur de texte un fichier représentant un dégradé de gris, comme celui projeté au tableau (signaler le pixel blanc tout à gauche s'il n'est pas visible). Le fichier, vide pour le moment, s'appelle III-3.3_defi_gris_vierge.pgm.

Puis les élèves doivent sauvegarder le fichier et l'ouvrir avec l'éditeur d'image, sans fermer l'éditeur de texte, et appeler l'enseignant lorsqu'ils ont, côte à côte, d'une part le fichier ouvert avec l'éditeur de texte, et d'autre part le dégradé visible avec l'éditeur d'image.
Un encodage possible, que la majorité des binômes propose, est le suivant :

```
P2
16 1
15
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

Pour arriver à ce résultat, les élèves doivent avoir compté le nombre de niveaux de gris sur le dégradé (16 en l'occurrence), et avoir adapté la troisième ligne en conséquence. Ils doivent également avoir adapté le format de la grille de pixels (16 colonnes et une seule ligne) et avoir encodé les pixels avec des valeurs décroissantes de 15 à 0, pour que le pixel blanc soit à gauche et le noir à droite conformément à la demande du défi.
Un autre encodage possible parmi d'autres est par exemple :

```
P2
16 1
31
31 29 27 25 23 21 19 17 15 13 11 9 7 5 3 0
```

Recherche : encodage d'un pixel en couleur (par binômes et collectivement)

L'enseignant rappelle que les images peuvent aussi être en couleurs. Il signale le fichier III-3.3_recherche_couleur, qui permet d'encoder un unique pixel en couleur. Il demande aux élèves d'essayer de comprendre comment fonctionne l'encodage grâce à ce fichier, comme ils l'ont fait dans le cas des niveaux de gris. Il leur suggère de ne toucher qu'à la dernière ligne pour commencer, et de regarder le résultat à l'écran avec les loupes (qu'il distribue).

L'encodage proposé dans le fichier, qui donne un pixel rouge, est celui-ci :

```
P3
1 1
7
7 0 0
```

Les élèves peuvent essayer, pour la troisième ligne, les cas extrêmes 0 0 0 (pixel noir), 7 7 7 (pixel blanc), 0 7 0 (pixel vert), 0 0 7 (pixel bleu), 0 7 7 (pixel cyan), 7 0 7 (pixel magenta), 7 7 0 (pixel jaune) et une multitude de combinaisons intermédiaires de leur choix (par exemple 7 6 2 qui donne un jaune d'or ou 1 5 6 qui donne un bleu ciel).

Lors de la mise en commun, les élèves rapportent qu'à la loupe, on voit de petites lumières rouges, vertes et bleues, de forme rectangulaire, juxtaposées de façon répétitive dans cet ordre répété de gauche à droite de l'écran. Lorsqu'on indique 0 0 0 sur la dernière ligne du fichier, les lumières de l'écran (aussi bien les rouges que les vertes et les bleues) sont peu lumineuses à la loupe, alors qu'avec 7 7 7, elles sont très lumineuses. Avec 7 0 0, seules les lumières rouges sont très lumineuses, etc. La classe synthétise collectivement que les trois valeurs de la dernière ligne indiquent le niveau de luminosité du pixel dans le rouge, le vert et le bleu (le pixel carré a donc 3 sous-pixels rectangulaires, de trois teintes R, V et B). La sensation que l'on a à distance de l'écran dépend de la balance entre les luminosités dans les trois teintes. La couleur obtenue pour les combinaisons suivantes est gardée visible au tableau en vue du défi suivant :

0 0 0	7 7 7	7 0 0	0 7 0	0 0 7	0 7 7	7 0 7	7 7 0
Noir	Blanc	Rouge	Vert	Bleu	Cyan	Magenta	Jaune

L'enseignant peut indiquer que le plus souvent, il y a 256 niveaux de rouge, de vert et de bleu possible, et non pas 8. Il précise que pour le défi suivant, on va au contraire réduire le nombre de niveaux de rouge, de vert et de bleu, pour n'en avoir que 2 : éteint ou allumé, comme pour le codage en noir et blanc.

Notes scientifiques

- L'extension de fichier ppm signifie « Portable Pixel Map » pour « Carte transportable de pixels ».
- Usuellement, les sous-pixels sont encodés sur 8 bits, ce qui correspond à 256

niveaux d'intensité possibles (valeurs de 0 à 255). Il faut donc 24 bits pour décrire un pixel en RVB (voir Séance III-1.3).

Défi : encoder un mini-phare en couleur

L'enseignant lance le dernier défi. Il s'agit d'encoder un petit phare en couleur, en utilisant seulement deux niveaux de luminosité pour les sous-pixels : 0 pour un sous-pixel éteint et 1 pour un sous-pixel allumé.

Autrement dit, le fichier III-3.3_defi_couleur_preremplie à modifier commence par :

```
P3
5 5
1
```

Suivi d'une grille de valeurs 0 (pixel éteint) ou 1 (pixel allumé). Le fichier est pré-rempli avec le nombre adéquat de 0 (15 par ligne, sur 5 lignes). Une ligne de commentaire commençant par le caractère # facilite le repérage des blocs de trois sous-pixels.
Les élèves qui réussissent le défi proposent l'encodage suivant :

```
P3
5 5
1
#    #    #    #    #
0 1 1 0 1 1 1 0 0 1 1 0 1 1 0
0 1 1 0 1 1 1 1 1 0 1 1 0 1 1
0 1 1 0 0 0 0 0 0 0 1 1 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
```

Il n'y a cette fois qu'une possibilité, puisque l'encodage sur 1 bit (0 ou 1 pour chaque sous-pixel) est imposé dans le défi.

Conclusion et trace écrite

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

- *On peut représenter les pixels d'une image par des nombres*
- *Chaque pixel d'une image en nuances de gris est représenté par un nombre. Si l'image comporte 8 nuances de gris, ce nombre est compris entre 0 (pixel noir) et 7 (pixel blanc).*
- *Chaque pixel d'une image colorée est représenté par 3 nombres qui indiquent l'intensité du rouge, du vert et du bleu de ce pixel.*
- *Pour transmettre une image en nuances de gris ou en couleur, on l'encode puis on transmet la version encodée, que l'on décode à l'arrivée pour reconstituer l'image.*

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

Prolongements

Les élèves peuvent s'exercer en prenant des images bien connues (le personnage de Mario©, la fusée de Tintin©, l'épée de Minecraft©…) et, dans un premier temps, les pixelliser à l'aide de grilles telles que celles utilisées dans la Séance III-3.1 (grille 16x16 par exemple) puis, dans un second temps, encoder ces images dans un fichier PBM, PGM ou PPM afin de pouvoir les afficher à l'écran.
En sciences comme en arts plastiques, on peut également approfondir la recomposition des couleurs par des triplets rouge-vert-bleu.

Notes scientifiques

- La synthèse additive (appelée parfois « couleur lumière » à l'école) permet d'ajouter des flux colorés afin de reconstituer toutes les couleurs du spectre visible. Inversement, la synthèse soustractive (appelée parfois « couleur matière » à l'école) combine des pigments colorés qui, mis ensemble, absorbent certaines couleurs de la lumière environnante et réfléchissent les autres.
- En utilisant trois lampes-torches recouvertes de papiers transparents colorés (emballages de bonbons) rouge, vert et bleu, les élèves peuvent tenter de reconstituer diverses couleurs, de plus en plus claires : cyan, magenta, jaune, blanc.

<< Séance III-3.2 Séquence III-3 Séance III-3.4 >>

Source URL: http://www.fondation-lamap.org/fr/page/34531/1-2-3-codez-activites-cycle-3-seance-33-optionnelle-comment-encore-une-image-en-gris-ou

FICHE 39
Des images à transmettre

Image A

Consigne : Pixellise cette image sur papier calque en utilisant la grille fournie par ton enseignant. Colorie entièrement les cases si elles contiennent du noir.

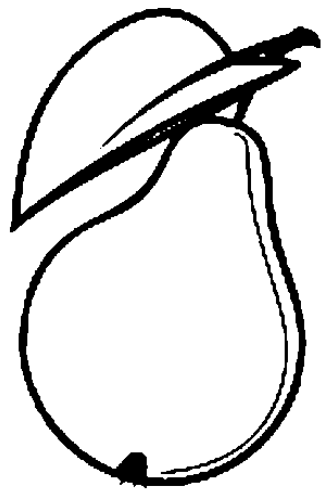


Image B

Consigne : Pixellise cette image sur papier calque en utilisant la grille fournie par ton enseignant. Colorie entièrement les cases si elles contiennent du noir.

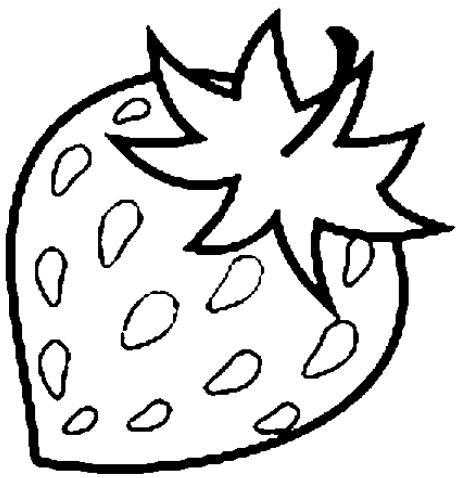


Image C

Consigne : Pixellise cette image sur papier calque en utilisant la grille fournie par ton enseignant. Colorie entièrement les cases si elles contiennent du noir.

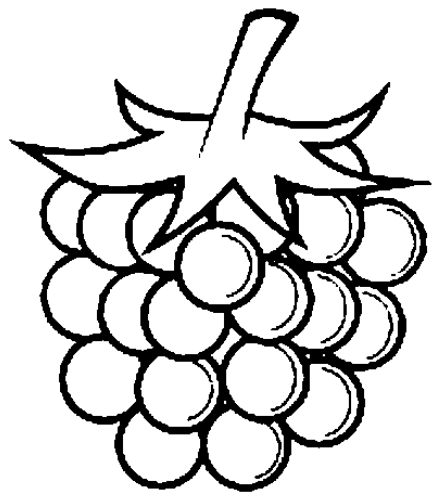
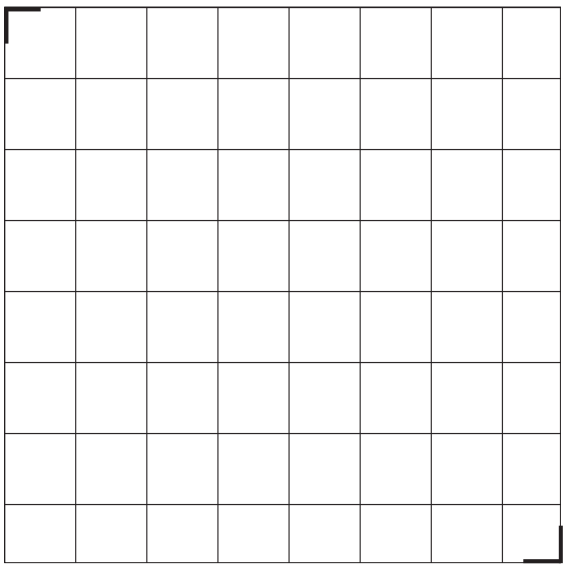


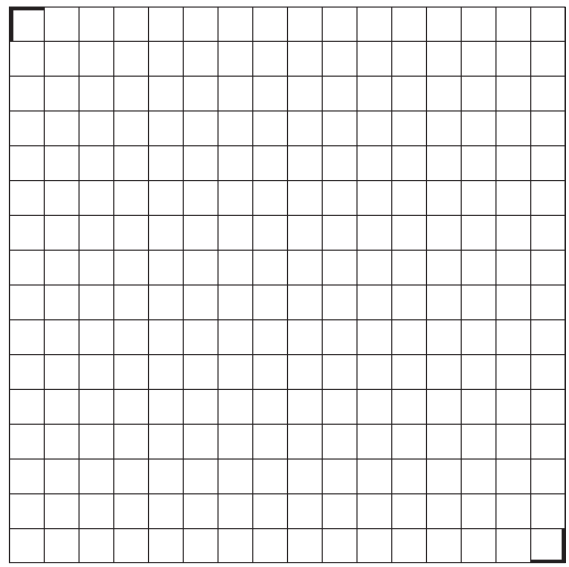
Image D

Consigne : Pixellise cette image sur papier calque en utilisant la grille fournie par ton enseignant. Colorie entièrement les cases si elles contiennent du noir.

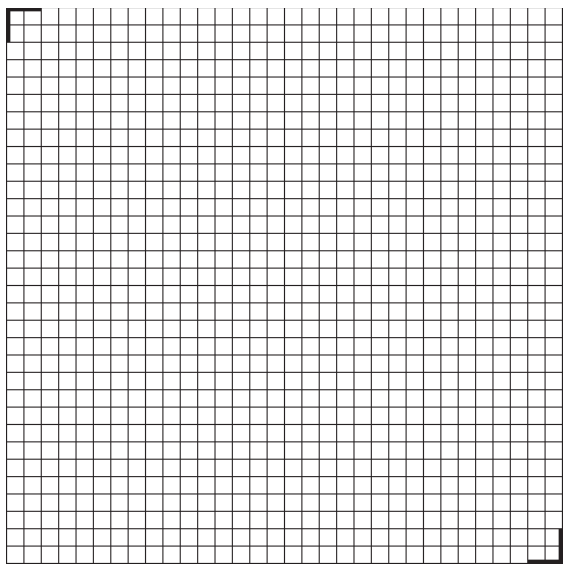
FICHE 40
Des grilles plus ou moins fines



Grille 1



Grille 2

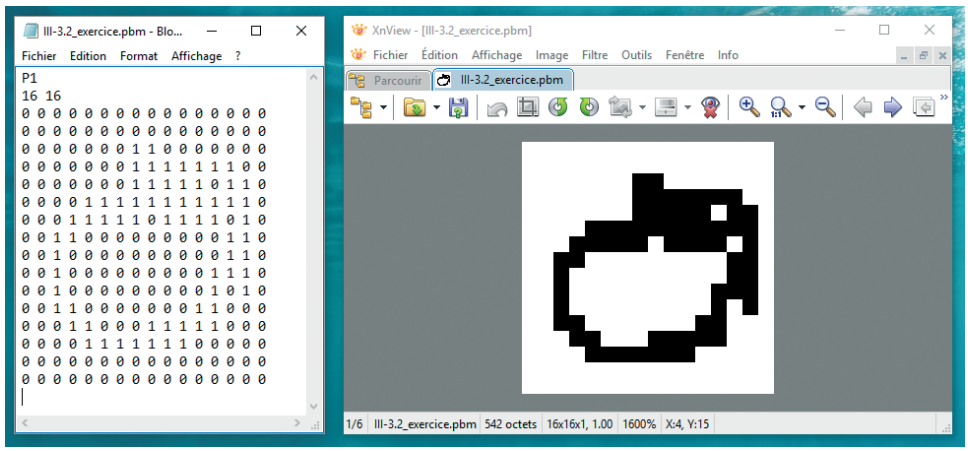


Grille 3

FICHE 41
Comment encoder des images en noir et blanc ?

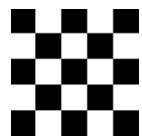
Recherche : comprendre l'encodage d'une image en noir et blanc

Le même fichier, nommé III-3.2_recherche_NB, est ouvert à gauche à l'aide d'un éditeur de texte et à droite à l'aide d'un éditeur d'image :



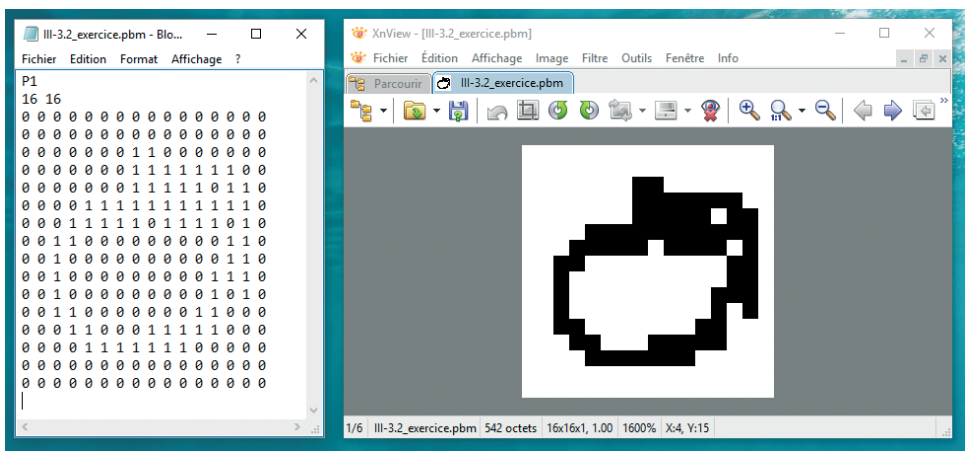
Consigne : explique ce que tu peux en déduire sur la façon dont est encodée une image en noir et blanc.

Défi : encode ce petit damier dans le fichier III-3.2_defi_NB_vierge.pbm et visualise-le à l'écran.



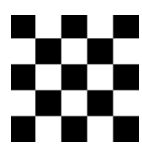
Recherche : comprendre l'encodage d'une image en noir et blanc

Le même fichier, nommé III-3.2_recherche_NB, est ouvert à gauche à l'aide d'un éditeur de texte et à droite à l'aide d'un éditeur d'image :



Consigne : explique ce que tu peux en déduire sur la façon dont est encodée une image en noir et blanc.

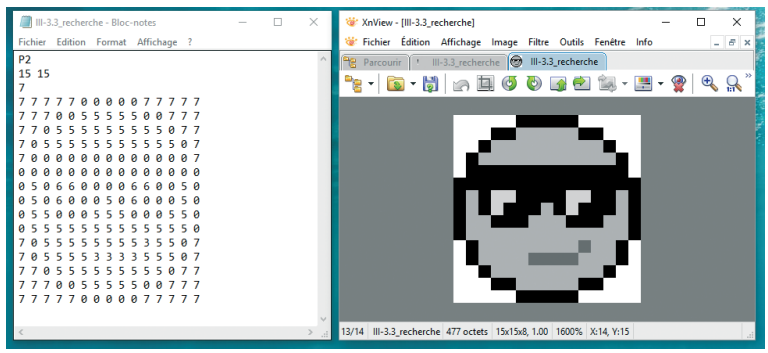
Défi : encode ce petit damier dans le fichier III-3.2_defi_NB_vierge.pbm et visualise-le à l'écran.



FICHE 42
Encoder des images en nuances de gris ?

Recherche : comprendre l'encodage d'une image en nuances de gris

Le même fichier, nommé III-3.3_recherche_gris, est ouvert à gauche à l'aide d'un éditeur de texte et à droite à l'aide d'un éditeur d'image :



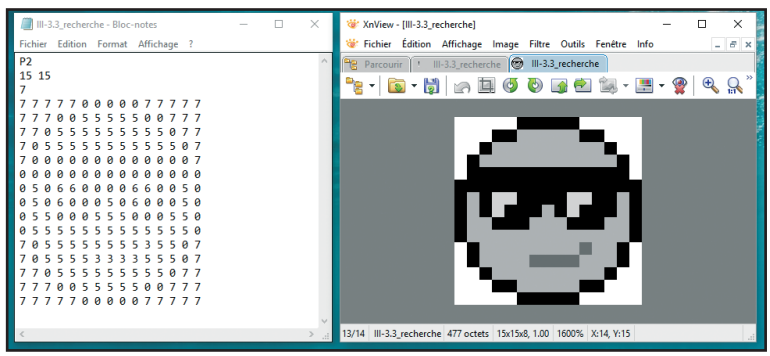
Consigne : Explique ce que tu peux en déduire sur la façon dont est encodée une image en nuances de gris.

Défi

Encode ce dégradé de gris dans le fichier III-3.3_defi_gris_vierge.pbm et visualise-le à l'écran (attention, il y a un pixel tout blanc à gauche).

Recherche : comprendre l'encodage d'une image en nuances de gris

Le même fichier, nommé III-3.3_recherche_gris, est ouvert à gauche à l'aide d'un éditeur de texte et à droite à l'aide d'un éditeur d'image :



Consigne : Explique ce que tu peux en déduire sur la façon dont est encodée une image en nuances de gris.

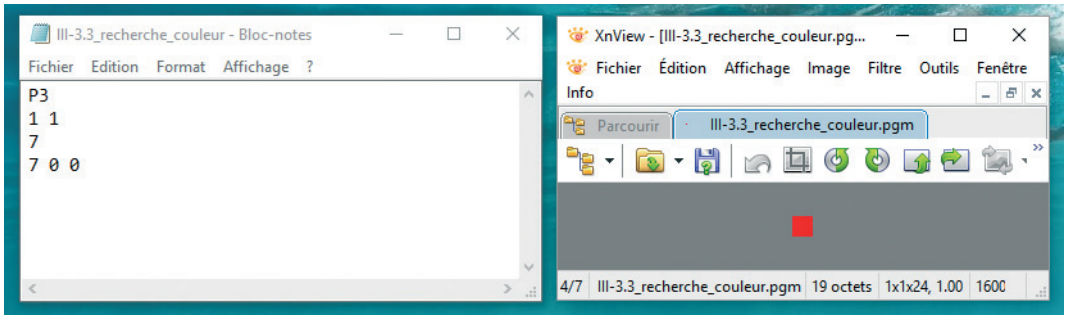
Défi

Encode ce dégradé de gris dans le fichier III-3.3_defi_gris_vierge.pbm et visualise-le à l'écran (attention, il y a un pixel tout blanc à gauche).

FICHE 43
Comment encoder des images en couleurs ?

Recherche : comprendre l'encodage d'un pixel coloré

Le même fichier, nommé III-3.3_recherche_couleur, est ouvert à gauche à l'aide d'un éditeur de texte et à droite à l'aide d'un éditeur d'image :



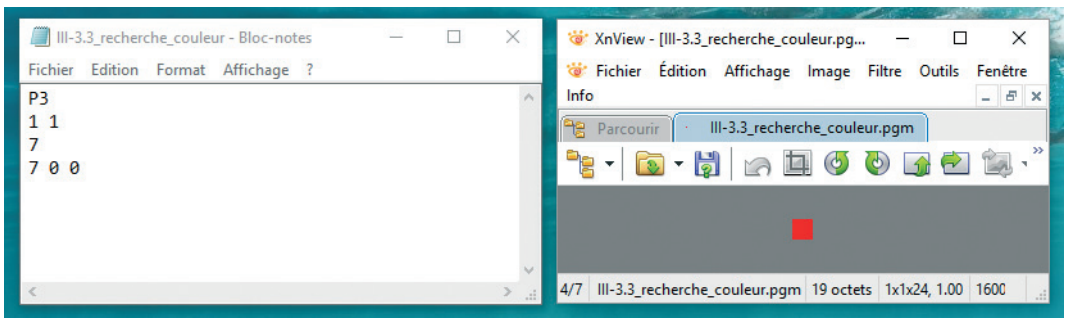
Consigne : Remplace les chiffres 7, 0 et 0 de la quatrième ligne du fichier par d'autres valeurs entre 0 et 7, sauvegarde, mets à jour l'affichage dans l'éditeur d'image et observe ce qui change (utilise ta loupe !). Explique ce que tu peux en déduire sur la façon dont est encodé un pixel coloré.

Défi

Encode ce phare dans le fichier pré-rempli III-3.3_defi_couleur_prerempli.pbm et visualise-le à l'écran.

Recherche : comprendre l'encodage d'un pixel coloré

Le même fichier, nommé III-3.3_recherche_couleur, est ouvert à gauche à l'aide d'un éditeur de texte et à droite à l'aide d'un éditeur d'image :



Consigne : Remplace les chiffres 7, 0 et 0 de la quatrième ligne du fichier par d'autres valeurs entre 0 et 7, sauvegarde, mets à jour l'affichage dans l'éditeur d'image et observe ce qui change (utilise ta loupe !). Explique ce que tu peux en déduire sur la façon dont est encodé un pixel coloré.

Défi

Encode ce phare dans le fichier pré-rempli III-3.3_defi_couleur_prerempli.pbm et visualise-le à l'écran.