Année universitaire 2016-2017

Une séquence de cycle 3 pour entrer dans la pensée informatique

Loïc LE GAT

RAPP - D.U. ADAPTÉ MEEF 1er DEGRÉ

Master « Métiers de l'Enseignement, de l'Education et de la Formation »

ESPE d'Aquitaine

160, avenue de Verdun – B.P. 90152 - 33705 Mérignac – Tél. : 05 56 12 67 00 – Fax : 05 56 12 67 99

ESPE Aquitaine	Mémoire de Master 2	Année universitaire 2016-2017
	Master « des Métiers de l'Enseignement, de l'Education et de la Formation »	

L'ESPE	
Intitulé	Ecole Supérieure du Professorat et de l'Education
Adresse postale	160, avenue de Verdun – B.P. 90152 – 33705 Mérignac

LE MEMOIRE	
Titre	Une séquence de cycle 3 pour entrer dans la pensée informatique
Etudiant	Loïc Le Gat
Directeur de mémoire	Stéphane Brunel
Mots-clefs	Pensée informatique, algorithme, programmation, Scratch

SOMMAIRE

Résumé :

Ce rapport présente la mise en œuvre d'une séquence d'enseignement, en classe de CM2 et en période 4, introductive à la pensée informatique. Après avoir défini la notion de pensée informatique comme un ensemble de stratégies et de concepts employés dans la résolution de problèmes, le rapport décrit la mise en place didactique et pédagogique d'une telle séquence, proposant diverses activités en débranché puis utilisant un logiciel de programmation adapté aux élèves de cycle 3. Ces activités sont à la fois des vecteurs de découverte et de consolidation des processus liés à la pensée informatique et s'appuient sur un objet, produit d'une réflexion scientifique et utilisant une syntaxe propre aux environnements numériques : l'algorithme. La dernière partie du rapport s'attache à analyser les productions des élèves tout au long de la séquence pour mettre en évidence ses réussites et ses échecs.

Abstract :

This internship aims to present how to conceive and lead a teaching module at CM2 level to introduce computational thinking. It will first underline the notion as a set of strategies and skills used in problems resolution. Then it will describe the didactical and pedagogical variables used to serve the main purpose: developing the computational thinking. It underscores that various activities, from computer free tasks to using a programing software, can be implemented to let children discovering and manipulating new concepts like algorithms. The last part of the internship will analyze some students' productions to point out either successful learnings and failures.

Introduction

Le président de la République François Hollande, lors du lancement du « plan pour le numérique à l'école » le 7 mai 2015, a annoncé : « Le numérique ce n'est pas simplement un outil, ce n'est pas simplement des pédagogies, des contenus. C'est aussi une culture, […] chaque collégien doit être doté des moyens de comprendre ce qui se lit, ce qui se voit sur les tablettes numériques ou sur les ordinateurs, d'en comprendre les enjeux en termes de citoyenneté, d'avoir aussi une bonne analyse de ce qu'est la programmation, comment se créent un certain nombre de contenus et de ressources. [...] À partir de la rentrée 2016, dès l'école primaire, tous les enfants seront éveillés au codage et à la culture digitale. » S'appuyant sur ces objectifs, le but de la séquence d'enseignement étudiée dans ce rapport est d'introduire les procédures caractéristiques de la pensée informatique. Il n'est pas de savoir utiliser parfaitement un langage de programmation ou un ordinateur mais bien de développer des compétences spécifiques pour résoudre un problème à l'aide d'un outil informatique. Ce rapport cherche ainsi à observer et analyser le comportement d'élèves découvrant la pensée informatique et mettant en œuvre les mécanismes cognitifs qu'elle engendre.

La première partie sera consacrée à la définition des notions liées au concept de *computational thinking*[1]. Le rapport décrira, dans une deuxième partie, la mise en place de la séquence d'enseignement dans une classe de CM2 tandis que la troisième partie présentera les productions des élèves et reviendra sur les réussites et échecs de ladite séquence.

La mise en place de ce sujet d'étude s'est déroulée au sein d'une classe de CM2 de vingt-huit élèves. Après recensement, plus de la moitié de la classe affirme ne pas savoir utiliser un ordinateur de façon autonome et la majorité des élèves utilise uniquement un ordinateur pour rechercher des informations sur internet et éventuellement traiter du texte. Un élève cependant est un utilisateur confirmé de Scratch et conçoit des petits jeux (casse-briques notamment) avec son père. Un autre élève possède chez lui un robot Thymio. La classe n'est pas équipée en matériel informatique (pas de TNI, ni d'ordinateurs/tablettes) mais l'école possède une salle informatique avec dix ordinateurs connectés.

[1] Que l'on traduira ici par pensée informatique (plutôt que pensée computationnelle).

1 Cadre théorique

1.1 La pensée informatique

Alexander Repenning (REPENNING 12) décrit la pensée informatique « comme une approche qui se concentre sciemment sur des concepts et des stratégies de résolution de problèmes de nature générale. Il s'agit entre autres de comprendre la relation entre processus séquentiels et processus parallèles. » En cela, elle est différente de ce que l'on appelle couramment la programmation : « La pensée informatique n'est pas la même chose que la programmation. Elle repose bien plus sur la compréhension des concepts de programmation, qui sont applicables à différents langages. Par contre, chaque langage de programmation se distingue par sa syntaxe, que les utilisatrices et utilisateurs doivent maîtriser pour pouvoir écrire un programme fonctionnel. Les détails syntaxiques d'un langage de programmation ne sont cependant pas très importants pour la compréhension des concepts de programmation. Cela ne signifie pas que la connaissance de divers langages de programmation soit sans importance pour la pensée informatique, mais que la pensée informatique privilégie les approches pédagogiques, qui s'inspirent de l'acquisition de la langue maternelle. » La pensée informatique doit être donc perçue comme un langage au même titre que n'importe quelle langue : elle est le fruit d'une logique de pensée et possède ses propres codes syntaxiques.

Si à ce titre il s'agit bien d'un langage à apprendre, c'est également un grand vecteur d'apprentissages et donc un langage pour apprendre ; ce que professeur de recherche sur l'apprentissage, directeur du Centre Okawa et directeur du groupe Lifelong Kindergarten au MIT Media Lab, Mitchel Resnick (RESNICK 12) appelle : « code to learn, learn to code[2]. » Dans cette vidéo de présentation du logiciel Scratch, il rappelle que, si les nouvelles générations sont habituées aux environnements numériques au sens large, elles n'en sont que consommatrices.

Un des enjeux de l'enseignement de la pensée informatique à l'école est donc de former autant des utilisateurs de ces nouvelles technologies que des créateurs. Les élèves ont à apprendre à utiliser les nouveaux langages mais aussi à les manipuler. Comme tout

[2] « Programmer pour apprendre, apprendre à programmer »

langage, la programmation possède ses propres codes qu'il convient de connaître et de pratiquer.

La figure ci-dessous (cf. Figure 1) résume les principales compétences développées lors d'un projet de conception d'algorithmes (EVIAN 16) :

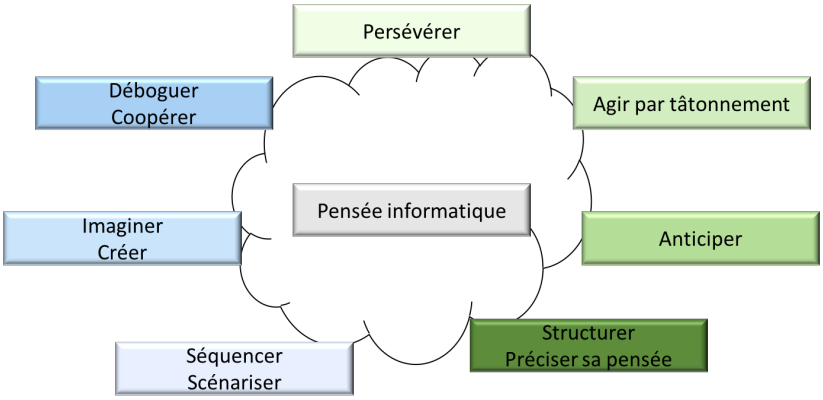


Figure 1 : les compétences liées la pensée informatique

- La persévérance : les activités impliquent de la résolution de problèmes, l'élève agit en chercheur de solutions multiples.
- Les essais et erreurs multiples ont à voir avec la gestion de l'erreur pratiquée dans la classe : l'erreur est désacralisée, le tâtonnement permet de multiplier les actions et rétro actions en développant une attitude réflexive sur ses choix. Elle permet ainsi de progresser vers la solution.
- L'anticipation ainsi que la structuration du raisonnement sont nécessaires à la programmation. Ceci permet de développer des capacités à gérer son projet et le mener à bout.
- La scénarisation oblige à découper ou organiser ses tâches en sous-tâches réalisables.
- La créativité est sollicitée tant esthétiquement que fonctionnellement. La curiosité est multipliée par les nombreuses possibilités d'actions et de variables offertes.
- Enfin, lors des débogages, précision et clarté du langage sont obligatoires afin d'exposer ses choix, argumenter et négocier.

1.2 L'objet algorithme

Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre un problème ou d'obtenir un résultat. Il peut être vu comme un produit de la pensée informatique. La séquence d'instructions qu'il propose peut être traduite dans un langage de programmation. Pour cela, l'algorithme doit être transformé en respectant les codes syntaxiques du langage utilisé. Le domaine qui étudie les algorithmes est appelé l'algorithmique. Ils se retrouvent dans de nombreuses applications telles que le fonctionnement des ordinateurs, la cryptographie, le traitement d'images ou de l'information.

La conception puis l'écriture d'un algorithme sont donc le fruit d'une pensée et sa transcription dans un langage permettant la communication (généralement avec une machine). Il est ainsi intimement lié au cheminement réflexif de son auteur et reflète ses stratégies de résolution du problème traité.

1.3 La place dans les programmes de l'école élémentaire

Le terme « algorithme » a fait explicitement son apparition dans le bulletin officiel de 2015 et bénéficie même d'un document d'accompagnement spécifique *Initiation à la programmation aux cycles 2 et 3* (EDUSCOL 16) qui précise :

« L'initiation à la programmation constitue une nouveauté importante pour les cycles 2 et 3. Elle s'inscrit dans les objectifs du socle commun de connaissances, de compétences et de culture », où il est précisé, dans le domaine 1 (Les langages pour penser et communiquer) : « [L'élève] *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 œuvre pour créer des applications simples.* ». Il s'agit aux cycles 2 et 3 d'amorcer un travail qui sera poursuivi au cycle 4. Ce document précise par ailleurs les objectifs des apprentissages :

« Aux cycles 2 et 3, les ambitions sont assez modestes : il s'agit de savoir coder ou décoder pour prévoir ou représenter des déplacements, de programmer les déplacements d'un robot ou ceux d'un personnage sur un écran. Des activités géométriques, consistant en la

construction de figures simples ou de figures composées de figures simples, sont également proposées. »

Dans le bulletin officiel, la notion d'algorithme se retrouve dans le thème « Nombres et calculs » : « Mettre en œuvre un algorithme de calcul posé pour l'addition, la soustraction, la multiplication. » et la notion de programmation apparaît dans le thème « Espace et géométrie » en lien avec l'objectif « (Se) repérer et (se) déplacer en utilisant des repères » au cycle 2. Il y est précisé : « Dès le CE1, les élèves peuvent coder des déplacements à l'aide d'un logiciel de programmation adapté, ce qui les amènera au CE2 à la compréhension, et la production d'algorithmes simples. »

Au cycle 3, on retrouve cette notion en Sciences au sein du thème « Matériaux et objets techniques » : « Les élèves découvrent l'algorithme en utilisant des logiciels d'applications visuelles et ludiques. Ils exploitent les moyens informatiques en pratiquant le travail collaboratif. Les élèves maitrisent le fonctionnement de logiciels usuels et s'approprient leur fonctionnement. » mais aussi en Mathématiques : « si la maitrise des techniques opératoires écrites permet à l'élève d'obtenir un résultat de calcul, la construction de ces techniques est l'occasion de retravailler les propriétés de la numération et de rencontrer des exemples d'algorithmes complexes. »

On vient de le voir, les programmes mettent eux-aussi en lumière le fait que la notion de d'algorithme, sous-jacente à celle de la pensée informatique, est perçue à l'école élémentaire autant comme un objectif d'apprentissage que comme un outil structurant la pensée et donc au service d'autres apprentissages.

1.4 Le logiciel Scratch

Scratch est un logiciel de programmation visuelle et multimédia. Il s'appuie sur la logique de programmation classique mais simplifie sa mise en œuvre. Il n'est en effet pas nécessaire de connaître ni d'apprendre le vocabulaire spécifique puisque l'utilisateur manipule des blocs représentant les invariants du langage de programmation : « si », « tant que », « jusqu'à ce que », etc. L'empilement des blocs formant l'algorithme représente bien la logique séquentielle d'un langage puisque les instructions sont effectuées successivement.

La figure ci-dessous (cf. Figure 2) décrit l'environnement de Scratch :

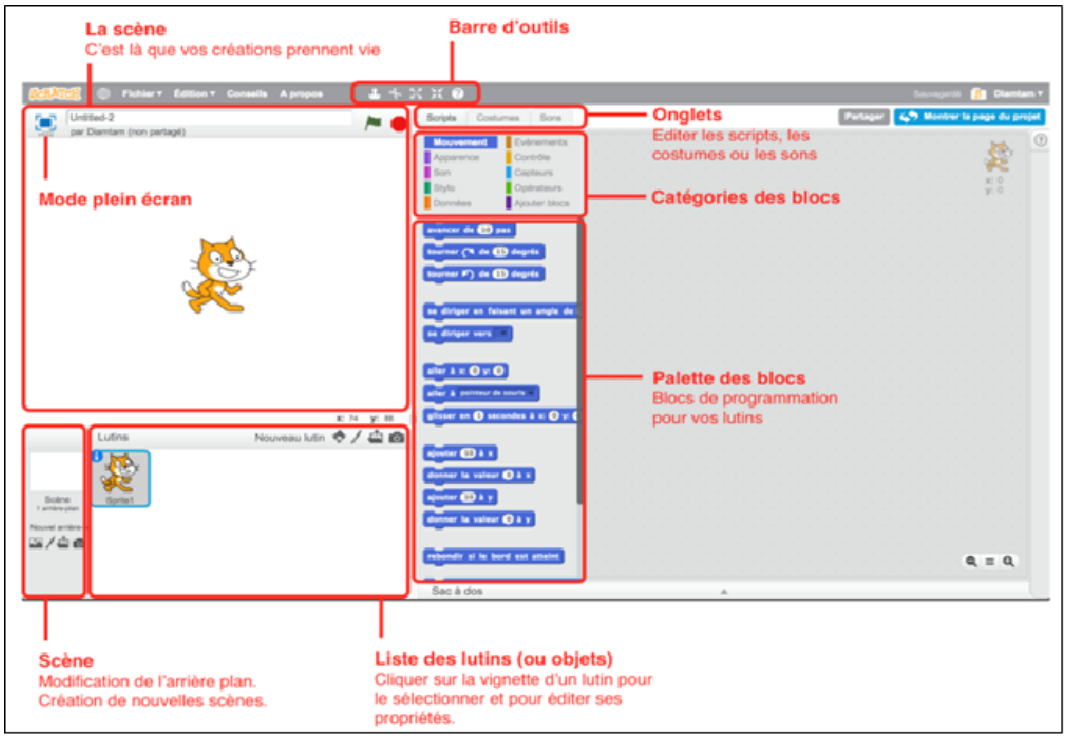


Figure 2 - capture d'écran d'un projet Scratch.

Les trois principales zones de l'interface sont :

- La scène : l'espace de réalisation des actions du lutin.
- Les blocs de programmations classés par catégories : mouvement, événements, apparence, contrôle etc.
- La fenêtre d'écriture du code : elle reçoit les différents blocs qui seront ensuite interprétés par le lutin.

Scratch ne nécessite aucune connaissance de la syntaxe de programmation. Les instructions possibles sont présentées sous forme de blocs s'empilant. Un système de formes permet de repérer rapidement comment introduire des conditions, des variables et des opérations dans l'algorithme. La syntaxe est ainsi prise en charge par le logiciel, permettant à l'élève de se concentrer sur la logique de son algorithme.

Après avoir introduit les notions fondamentales contenues dans le terme « pensée informatique », nous allons voir maintenant un exemple de séquence mise en place dans une classe de CM2.

2 Elaboration et mise en œuvre didactique

2.1 Présentation de la séquence : conception

La séquence proposée a pour objectif final la découverte et le développement de la pensée scientifique. Elle ne vise pas à découvrir directement un langage de programmation spécifique mais plutôt la logique de la programmation. Elle utilise la notion d'algorithme et son écriture via un logiciel de programmation adapté à la fois comme un outil pour exprimer cette pensée scientifique et comme le résultat du traitement d'un problème (sous forme d'un algorithme).

La présente partie s'attache à présenter la mise en œuvre de cette progression à travers différentes situations de communication et de réflexion, dont les variables pédagogiques et didactiques seront précisées ensuite.

Pierre Tchounikine, dans *Initier les élèves à la pensée informatique,* propose un classement des objectifs pédagogiques des séances autour de la pensée informatique. La séquence décrite dans ce rapport vise plusieurs catégories d'objectifs parmi celles définies par l'auteur (TCHOUNIKINE, 17, pages 8, 9 et 10) :

- Des objectifs pédagogiques autour de l'algorithmique et de la résolution de problèmes : faire comprendre la notion d'algorithme et les notions sous-jacentes (action/instruction, séquence, boucle, conditions, etc.) ; faire pratiquer la construction d'algorithmes ; faire comprendre et pratiquer la démarche : modélisation – décomposition en sous problèmes – algorithme – codage.
- Des objectifs pédagogiques autour de la notion de langage : faire comprendre la notion de langage en tant que telle (comme un système de codage et un outil de communication).
- Des objectifs pédagogiques autres qu'informatiques : travailler des compétences non-informatiques en numération et géométrie.

Afin de faire prendre conscience aux élèves de ce qu'est la pensée informatique, il m'a semblé important de leur faire découvrir de façon concrète ce qu'est un algorithme. En effet, s'intéresser aux algorithmes c'est toucher du doigt les grandes caractéristiques de la pensée informatique puisqu'ils en sont le résultat comme suite d'instructions claires et finies

permettant de résoudre un problème. Chaque problème abordé l'a été, dans un premier temps, sans faire de lien avec un environnement numérique. Ainsi, la pensée informatique peut être introduite sans surcharger les élèves cognitivement. Les premières séances se déroulent donc « en débranché » : aucun matériel informatique ou numérique n'est disponible, les spécificités d'un langage de programmation sont abordées dans des situations de communication entre les élèves. Ces situations mettent en évidence les grandes différences entre une communication entre humains et celle entre humains et machines. Par une série d'essais, erreurs, analyse puis modifications des instructions données, les élèves sont amenés à percevoir les grandes caractéristiques d'un algorithme. Dans un deuxième temps, ces situations orales sont retranscrites sur feuille et simule l'écriture d'un code informatique en respectant ses spécificités.

A la suite à cette première étape, les élèves découvrent un logiciel permettant de communiquer avec un robot : Scratch. Les activités de découverte de l'environnement logiciel reprennent celles menées en débranché afin que la classe puisse réinvestir les apprentissages. Les élèves apprennent ainsi à connaître l'organisation d'un environnement numérique. Ils décrivent un système technique par ses composantes et leurs relations et découvrent l'algorithmique de manière autant concrète que ludique.

2.2 Mise en place

La séquence est composée de six séances (cf. Annexe 1). La succession des séances suit la logique suivante :

1. Activité en débranché avec travail collectif
2. Reprise de l'activité en individuel ou en groupe avec écriture de tout ou partie de l'algorithme sur feuille
3. Ecriture en individuel de l'algorithme sur le logiciel Scratch

Les séances en débranché s'effectuent dans la cour et sont avant tout des temps de discussion en classe entière ou en groupe. Elles ont pour principal objectif l'appropriation du lexique et la confrontation des différentes procédures proposées par la classe.

Les reprises de ces séances sur feuille sont, elles, dédiées à conserver une trace écrite et donnent à voir aux élèves que la pensée informatique peut s'écrire et prendre la forme d'un algorithme. Ils découvrent ainsi les premiers codes spécifiques au langage

algorithmique. La dernière séance de retranscription du travail oral (séance 4) s'effectue en groupe. Cette modalité a été choisie en raison de la difficulté de réinvestir la notion de condition appliquée à du tri de nombres.

Les deux séances de découverte de Scratch ont nécessité l'organisation de la classe en quatre groupes de sept élèves, seuls dix ordinateurs étant disponibles. Il m'a semblé important que les élèves puissent travailler en autonomie sur le logiciel. Des binômes auraient présenté le risque de voir un élève prendre le contrôle de l'activité. Un travail individuel permet en outre à chacun d'explorer librement le logiciel, sans être influencé ou guidé. Ces deux séances ont donc proposé quatre ateliers tournants.

2.2.1 Séance 1

La première séance veut faire découvrir l'objet algorithme. Sa définition est appréhendée en découvrant les codes de la communication avec un robot. Ces codes sont mis en évidence à travers le jeu du robot idiot, qui propose une situation de communication entre le groupe classe et un élève choisi pour être le robot. Le but du jeu est de le guider jusqu'à la sortie d'un labyrinthe matérialisé dans la cour suivant le schéma de la figure ci-dessous (cf. Figure 3).

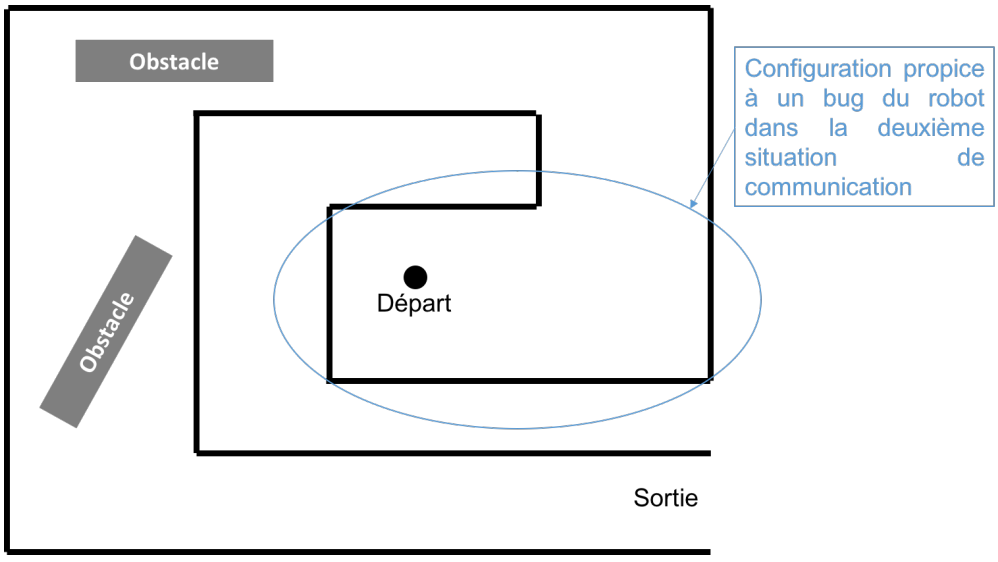


Figure 3 : schéma du labyrinthe utilisé dans le jeu du robot idiot.

Le robot idiot doit être préalablement informé de l'attitude qu'il doit adopter lorsqu'il reçoit un ordre. Ses réponses aux instructions données par la classe doivent montrer les spécificités de la communication avec une machine : les instructions données

doivent être claires et finies sinon le robot n'est pas en mesure de répondre à la sollicitation ou se met en boucle (notion de « bug »). Le labyrinthe est conçu de telle façon qu'un bug puisse rapidement apparaître, notamment lorsque la classe réfléchit à des instructions itérées du type :

« Répète indéfiniment :

« avance jusqu'au mur »

« tourne de 90° à droite » »

Un élève de la classe connaissant déjà Scratch, et ayant de bonnes connaissances informatiques, a été désigné pour être le robot. Voici les règles comportementales qui lui ont été dictées :

- Une instruction non finie doit entraîner une réalisation en boucle. Par exemple : « avance » provoque un mouvement infini du robot dans la direction d'orientation. De même, « tourne » ne doit entrainer aucune réaction car l'instruction n'est pas claire : il manque un sens de rotation, le robot n'est pas capable de décider de ce sens, il est idiot.
- Il n'existe aucun moyen d'arrêter le robot lorsqu'il est en train d'exécuter une instruction.
- Il faut attendre la fin de la tâche en cours pour passer à la suivante.

La séance se décompose en deux grandes phases. Dans une première partie, la classe, organisée en groupes de quatre élèves, réfléchit à sa suite d'instructions à donner au robot pour qu'il sorte du labyrinthe. Ce dernier ne présente pour l'instant pas d'obstacles. Les premiers essais doivent souligner l'importance de délivrer un ordre précis et fini. Les instructions sont données au robot au fur et à mesure. Cette modalité permet de se concentrer sur la précision des instructions à donner. Ce temps est également l'occasion de se familiariser avec l'espace de déplacement du robot. En effet, la vision dans le plan et la perception de l'espace posent encore problème chez certains élèves. Cette première activité

a donc aussi pour but de libérer ces élèves de cette charge cognitive. Chaque groupe va proposer sa solution et, à chaque instruction, le comportement du robot est étudié par la classe. Il s'agit ainsi de placer les élèves en situation d'analyse de l'erreur pour adapter les instructions futures. Une fois cette étape terminée, le travail est repris en ajoutant des obstacles le long du parcours.

Ce travail mené, une deuxième phase voit l'enseignant provoquer une rupture en modifiant la consigne : les élèves doivent automatiser leur algorithme : celui-ci doit être donné au robot au début sans pouvoir être modifié par la suite et doit alléger l'algorithme. Cette situation force les élèves à réfléchir à des conditions « automatiques » de réalisation et d'arrêt des instructions. Elle amène la classe à utiliser des ordres conditionnés et représente donc une première approche des bouches « si », « tant que » etc. Pour autant, cette première introduction d'une démarche de codage par anticipation n'est pas l'objectif de la séance et la notion sera reprise plus tard. Ce travail, de par la configuration du labyrinthe, doit induire la notion de bug : ici l'itération des mêmes instructions va amener le robot à tourner en boucle et rester bloqué dans le labyrinthe. Les élèves vont donc percevoir l'importance de prévoir une condition d'arrêt.

Pour conclure l'activité, un algorithme plus complexe est testé sur le robot idiot. Il est proposé aux élèves de faire intervenir le hasard dans leurs instructions. Il s'agit en effet de l'unique façon de toujours sortir d'un labyrinthe sans en visualiser le plan. Cet algorithme fonctionne pour tous les types de labyrinthe. La contrainte réside dans le fait qu'il peut tourner un certain temps avant d'arriver à la solution mais finira toujours par sortir (en considérant qu'il peut être exécuté indéfiniment par le robot). Cet algorithme demande au robot de se déplacer jusqu'à un obstacle et de tirer au sort son orientation (à droite ou à gauche). Il fait appel à une boucle de condition dont le résultat est soit « tourne à droite » soit « tourne à gauche », le tout inséré dans une boucle itérative jusqu'à la sortie.

La séance s'achève sur une phase collective pendant laquelle les élèves sont amenés à reformuler les apprentissages et donner une définition d'un algorithme. L'enseignant apporte également des éléments historiques en mentionnant les travaux du scientifique Al Khwarizmi au IX^{e} siècle.

2.2.2 Séance 2

La deuxième séance sera brièvement décrite. Elle reprend l'activité effectuée en séance 1. Un labyrinthe sur une feuille A4 est proposé et chaque élève doit écrire individuellement son algorithme permettant au robot d'en sortir. Cette écriture est encadrée par des règles qui doivent permettre de faire percevoir à l'élève l'aspect séquentiel d'une écriture algorithmique : on ne peut écrire qu'une instruction par ligne. L'objectif est ici de retravailler la perception de l'espace et l'orientation tout en se familiarisant avec la syntaxe propre aux codes informatiques.

2.2.3 Séances 3 et 4

Ces deux séances reprennent le schéma des deux premières et s'intéressent à une activité de numération. L'objectif est de découvrir un algorithme permettant de trier quatre nombres décimaux. Il s'agit autant de consolider et réinvestir les acquis en numération que d'aborder les boucles de condition. En effet, dans cette activité de tri, la classe doit mettre en œuvre une procédure quasi automatisée s'appuyant sur la comparaison successive de deux nombres.

La première séance se déroule à nouveau dans la cour. La classe est organisée en sept groupes de quatre élèves. Les groupes ont été constitués de manière homogène pour certains et hétérogène pour d'autres. Il peut sembler en effet intéressant d'observer le comportement d'élèves peu à l'aise avec la pensée informatique lorsqu'ils sont entourés soit d'élèves de même niveau soit d'élèves plus expérimentés.

Chaque élève du groupe reçoit une pancarte sur laquelle est inscrit un nombre décimal parmi : 8,01 ; 8,09 ; 8,1 et 8,11. Le choix de cette activité participe au tissage effectué entre cette séquence et les apprentissages menés sur la même période en numération. Le groupe possède également une fiche sur laquelle est matérialisé le parcours avec les différentes étapes mentionnées. Une première phase d'entrée dans l'activité voit chaque groupe s'organiser en ligne et par ordre croissant sans qu'aucune stratégie ne soit proposée. Après une discussion collective sur les procédures observées et leur efficacité respective, l'enseignant dévoile le terrain qui sera à disposition de chaque groupe pour l'activité suivante. Il est précisé que cette deuxième phase doit amener les élèves à trouver une procédure efficace et universelle permettant de trier une série de nombres. Ce terrain,

schématisé sur la figure 4 ci-dessous (cf. Figure 4), représente un outil dont doivent s'emparer les élèves et qui reproduit les différentes étapes de l'algorithme final :

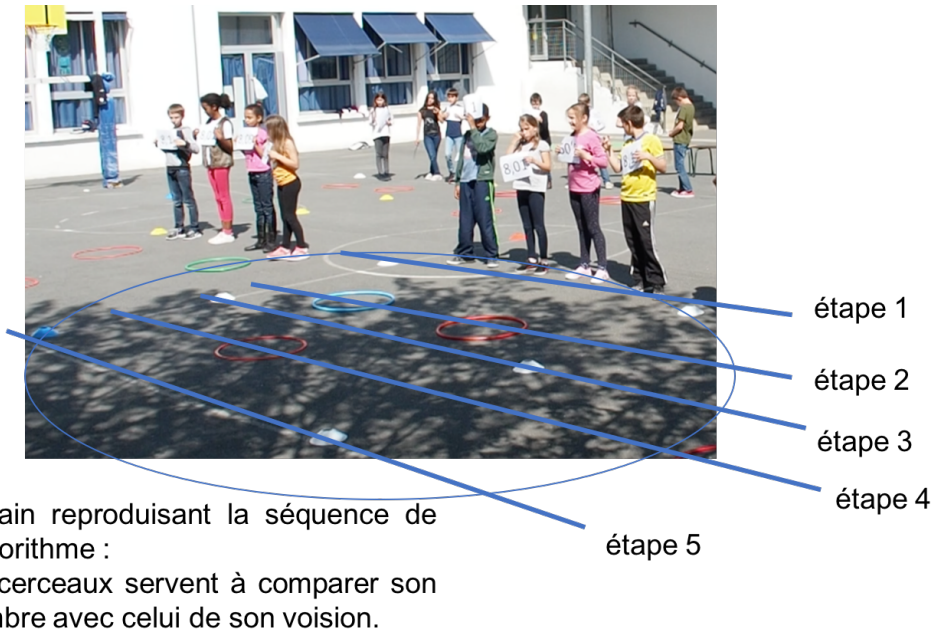


Figure 4 : une photo de l'organisation spatiale du parcours.

Les élèves doivent évoluer sur ce parcours, depuis une disposition initiale dans laquelle les nombres ne sont pas triés vers la situation finale dans laquelle les nombres sont disposés en ordre croissant. Pour aider le groupe, des contraintes sont données :

- Une ligne de plots représente une avancée vers la solution.
- On utilise les cerceaux pour se comparer avec son voisin.
- On ne peut pas reculer.
- On ne doit se comparer qu'une seule fois avec un camarade.

Ces contraintes ainsi que l'organisation spatiale ont pour fonction de guider les groupes vers la solution.

Des phases de réflexions et de discussions sont prévues régulièrement pour faire progresser tous les groupes vers la solution. La figure 5 ci-après schématise l'algorithme de tri sur le parcours :

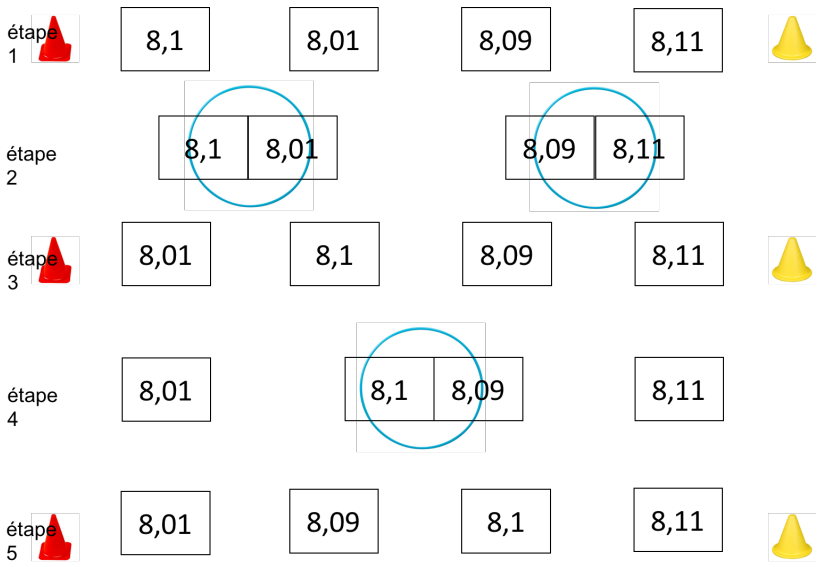


Figure 5 : schéma de l'algorithme de tri.

La mise en œuvre de l'algorithme et donc la progression des élèves de l'étape 1 jusqu'à la 4 doit rendre concrète la notion séquentielle contenue dans un algorithme : chaque étape permet d'évoluer vers la solution.

Suite à cette activité « en débranché », dans une deuxième séance, chaque groupe devra reproduire ce schéma sur feuille et écrire les instructions à donner au nombre 8,09 pour qu'il progresse vers la solution. Cette écriture doit faire apparaître le mot clé « si » et les deux conséquences possibles : soit on inverse sa position avec le nombre comparé soit on reste à sa place. Le travail sur le nombre 8,09 est intéressant car il fait intervenir la même boucle de condition « si » mais amenant aux deux possibilités : d'abord la comparaison implique au nombre de conserver sa place dans la série puis l'étape 4 entraîne un changement de position.

2.2.4 Séances 5 et 6

Les deux dernières séances de la séquence visent, dans un premier temps, la découverte d'un environnement informatique permettant d'écrire et de tester des algorithmes (les fiches découverte sont en annexes 2 et 3). Dans un deuxième temps, chaque élève devra écrire un algorithme codant le déplacement du lutin Scratch à travers un labyrinthe (identique à celui de la séance 2).

La première séance consiste en une activité semi-libre de découverte de Scratch. Individuellement, à l'aide d'une fiche d'activité et de l'étayage de l'enseignant, chaque élève

explore le logiciel : les blocs d'instructions, les précisions modifiables dans les blocs (par exemple l'angle d'orientation du lutin ou le nombre de pas d'un déplacement). Cette prise en main prépare le travail de programmation de la séance suivante et fait prendre conscience aux élèves du caractère créatif d'un tel logiciel.

La dernière séance propose aux élèves de travailler en autonomie à la réalisation d'un algorithme codant les déplacements du lutin. Pour faciliter ce travail, l'enseignant aura écrit le début de l'algorithme codant une remise à zéro des traces de stylo sur l'écran représentant le chemin parcouru par le lutin et une position initiale fixée à l'entrée du labyrinthe. L'objectif est que les élèves, par essais successifs et analyse des erreurs, parviennent à faire avancer le lutin et à l'orienter jusqu'à la sortie. Pour cela, la fonction Scratch « stylo » est utilisée, qui permet de visualiser la trajectoire du lutin lorsqu'il effectue les actions ordonnées par le concepteur du code. Le labyrinthe proposé est identique à celui vu en séance 2. Cependant, les déplacements ne s'expriment pas en nombre de carreaux mais en nombre de pas dont la mesure n'est pas connue. Un premier travail de tâtonnement est donc indispensable pour fixer la distance du déplacement du lutin.

Suite à la présentation du contenu didactique des séances de la séquence, la dernière partie de ce rapport va traiter des observations relevées pendant les différentes activités.

3 Résultats et analyse

3.1 Généralités

Cette partie présente principalement les productions des élèves. Les films réalisés pendant les séances 1 et 3 seront brièvement commentés, l'absence de son pendant la prise ne permettant pas d'analyser les différents échanges langagiers.

Les algorithmes produits en séance 6 sont les premiers marqueurs visibles des apprentissages réalisés tout au long de la séquence. Un temps sera prévu ultérieurement pour améliorer ces algorithmes (pour les simplifier notamment).

3.2 Méthodes et outils

Les productions d'élèves lors des séances d'écriture des algorithmes sont présentées en annexe ainsi que des exemples de code Scratch écrits lors de la dernière séance. Les productions choisies sont celles d'élèves connaissant déjà l'environnement Scratch et

d'élèves peu à l'aise avec du matériel informatique. Les différentes productions demandées aux élèves (algorithme écrit à la main, schéma ou code informatique) avaient pour objectif de donner à voir que la pensée informatique n'est pas toujours liée à l'utilisation de matériel informatique et peut s'exprimer de différentes façons, en s'affranchissant de certains codes.

Il s'agira, dans cette partie, en étudiant les différentes productions, de voir en quoi la progression a permis ou non à ces derniers d'entrer dans la pensée informatique.

3.3 Séances 1 et 2

Les phases de discussion entre le groupe et le robot et entre élèves ont été riches. La première séance a ainsi permis d'introduire les grandes caractéristiques de la pensée informatique. Globalement, la vidéo révèle que les élèves se sont rapidement mis en activité et se sont approprié la tâche. Les premiers essais d'instructions ont mis en évidence le caractère fini des ordres donnés et l'impossibilité d'arrêter le robot lorsqu'un ordre est erroné. En effet, plusieurs élèves, voyant que leur instruction faisait boucler le robot, ont voulu stopper son action en lui criant « stop ! », sans effet bien entendu.

Les discussions quant à une automatisation des instructions ont fait émerger comme prévu la notion de bug puisque le robot s'est rapidement retrouvé pris au piège dans une partie du labyrinthe. La dernière phase et l'implémentation d'un algorithme permettant de sortir quoi qu'il arrive en faisant intervenir le hasard s'est révélée beaucoup plus transmissive mais la classe a pu prendre conscience de ses effets lorsqu'il a été testé par le robot idiot pendant de longues minutes.

Le travail écrit de la séance 2 devait mettre en évidence les réussites et échecs dans l'apprentissage de la notion d'algorithme. Il en ressort qu'une majorité des élèves a respecté la syntaxe algorithmique (cf. Elève 1 en annexe 4). On notera quand même que quelques élèves n'ont pas réinvesti l'écriture d'une instruction par ligne (cf. Elève 2 en annexe 4). Presque tous les élèves ont proposé un algorithme permettant effectivement au robot de sortir du labyrinthe. Certains ont néanmoins eu besoin d'aide pour orienter correctement le sens de déplacement du robot, dénotant d'un manque d'abstraction lors du passage d'un labyrinthe en 3D à celui en 2D sur la feuille. Il est intéressant de noter qu'un élève a réinvesti le dernier algorithme testé en séance 1 en faisant intervenir le hasard. L'élève 3 dont la

production est en annexe 4 a réussi à réinvestir cet apprentissage lors de la séance de réécriture (séance 2).

Il aurait pu être intéressant, lors de la séance 1, de donner le plan du labyrinthe aux élèves en amont et de concevoir l'algorithme par anticipation. Ils auraient ainsi écrit, en groupe, l'algorithme en inscrivant chaque instruction sur une pancarte et auraient constitué « physiquement » la séquence de déplacement en plaçant les pancartes les unes sous les autres. Cette séquence aurait été placée sur le sol à côté du labyrinthe et interprétée ensuite par le robot. Cette activité aurait fait percevoir la notion de programmation par anticipation : écriture du code en entier puis exécution. Un travail sur les erreurs relevées dans le code aurait finalement permis d'induire le raisonnement itératif : conception, test, erreurs observées, modification, conception, etc.

3.4 Séances 3 et 4

Ces deux séances ont paru beaucoup plus abstraites aux élèves. La classe a eu des difficultés à se saisir de l'objectif. A priori, ceci est pour partie dû à un manque de mise en perspective de ces deux activités dans la séquence complète et à la difficulté de la tâche proposée. Les algorithmes de tri de nombres mettent en jeu des processus complexes et la volonté d'intégrer un travail sur des nombres décimaux s'est révélée trop ambitieuse. Une activité sur des nombres entiers auraient développé les mêmes procédures sans complexifier l'activité. Pour la séance 3, les groupes homogènes composés d'élèves peu à l'aise avec les concepts n'ont pas réussi à relier l'activité avec l'écriture d'un algorithme (cf. Annexe 5, groupe 2). Heureusement deux groupes ont réussi à trouver les déplacements recomposant ainsi l'algorithme. Ils en ont fait la démonstration devant le groupe classe pour que tout le monde puisse se rendre compte visuellement de la progression étape par étape vers la solution. Le schéma réalisé par le groupe 1 ainsi que le code proposé par le groupe 3 (cf. Annexe 5) montrent que certains élèves pourtant non experts ont su réinvestir le travail de la séance en débranché.

Au regard de l'objectif d'apprentissage visé, une activité plus ludique pourrait maintenant être proposée avec Scratch, remplaçant cette activité de tri et rendant la compréhension des boucles de condition plus concrète.

3.5 Séances 5 et 6

Les activités de découverte de Scratch ont plu aux élèves, justifiant le pouvoir enrôlant du travail sur un logiciel de programmation. Bien que ces activités n'aient pas fait appel à la créativité des élèves, ceux-ci ont tout de même pris du plaisir à écrire leur algorithme et à observer les conséquences de leurs actions sur le comportement du lutin. En annexe 6 se trouvent quatre productions faisant suite à l'activité de découverte de la séance 5. On constate des comportements différents dans l'appropriation des possibilités du logiciel. L'élève 1, pourtant peut à l'aise lors des premières séances a utilisé une boucle. L'élève 2, utilisateur régulier de Scratch, a joué sur les paramètres hors code (création de lutins, modification de l'arrière-plan) et semble percevoir avant tout ce logiciel comme un outil pour la conception de jeux vidéo. L'élève 3 a largement exploré les différentes catégories d'instructions mais ne s'est pas totalement saisi de la syntaxe : on remarque une utilisation successive d'instructions identiques plutôt qu'un regroupement de ces mouvements (trois blocs « avancer de 10 » équivalent à « avancer de 30 » ou « répéter 3 fois » + « avancer de 10 »). Enfin, l'élève 4 a bien exploré les différentes actions et a joué sur les paramètres intrinsèques des instructions, mais sans chercher à obtenir un programme cohérent.

Les captures d'écran de la séance 6 sont en annexe 7. Elles montrent les travaux de trois élèves. Le premier est en réussite mais son algorithme n'est pas optimisé : plusieurs blocs de déplacement se succèdent. L'élève 2 utilise l'instruction « tourner » alors que le 3 utilise « s'orienter ». Cette dernière instruction est bien plus simple car elle ne tient pas compte de la précédente orientation alors que « tourner » demande un angle calculé entre la précédente orientation et celle désirée. Beaucoup d'élève n'ont pas eu le temps de terminer leur production car ils n'avaient que trente minutes (pour des raisons matérielles et logistiques) pour écrire leur code. Cette séance sera donc reprise afin de terminer/optimiser les algorithmes et de verbaliser les principaux apprentissages des séances 5 et 6. Notons pour finir que l'élève expert a, quant à lui, développé une communication entre deux lutins, comme le montre la figure 6 ci-après :

Figure 6 - captures d'écran d'une production d'élève de la séance 6

3.6 La question de l'évaluation

Une fois ces observations faites, se pose la question de l'évaluation : comment mettre en évidence une progression dans la maîtrise des différentes compétences mises en jeu ?

Comme on l'a vu, celles-ci sont diverses et surtout transversales. Ces évolutions doivent donc pouvoir s'observer dans des activités touchant de nombreux domaines. On pense notamment à la résolution de problèmes : le développement de la pensée informatique doit pouvoir s'observer lors d'activités de résolution de problèmes, entre autres dans la décomposition d'un problème en sous-problèmes (rejoignant le travail sur la recherche de questions intermédiaires ou l'organisation des données d'un problème, thèmes abordés en période 3 et 4).

L'acquisition et la consolidation des compétences mises en jeu seront vérifiées à moyen voire long terme et ne pourront pas être discutées ici. Dès que la maîtrise des fonctions de base de Scratch sera assurée pour tous les élèves de la classe, le logiciel servira d'outil à la résolution d'un problème complexe et le comportement des élèves sera évalué selon plusieurs critères :

- Capacité à comprendre le problème et à le séquencer

- Imagination et création de l'algorithme
- Capacité à analyser les erreurs lors du test
- Capacité à modifier son code

Cette évaluation formative se déroulera sur plusieurs séances et veillera à être différenciée. Les élèves les moins à l'aise bénéficieront d'une aide pour l'écriture de l'algorithme, cette compétence n'étant pas la plus importante.

3.7 Vers l'utilisation du logiciel Scratch pour aborder des problèmes complexes

Comme le montre la figure ci-dessous (cf. Figure 7), tirée des travaux de A. Repenning (REPENNING, 14, page 16) et inspirée de la théorie de Vygotsky sur la zone de développement proximal (ZDP), l'apprentissage est favorisé lors de la résolution de situations résistantes. Elles induisent en effet chez l'élève la recherche de procédures non évidentes nécessitant l'utilisation d'outils d'aide. L'utilisation de Scratch comme outil pour structurer son résonnement et tâtonner vers la solution est particulièrement propice à l'entrée dans la ZDP.

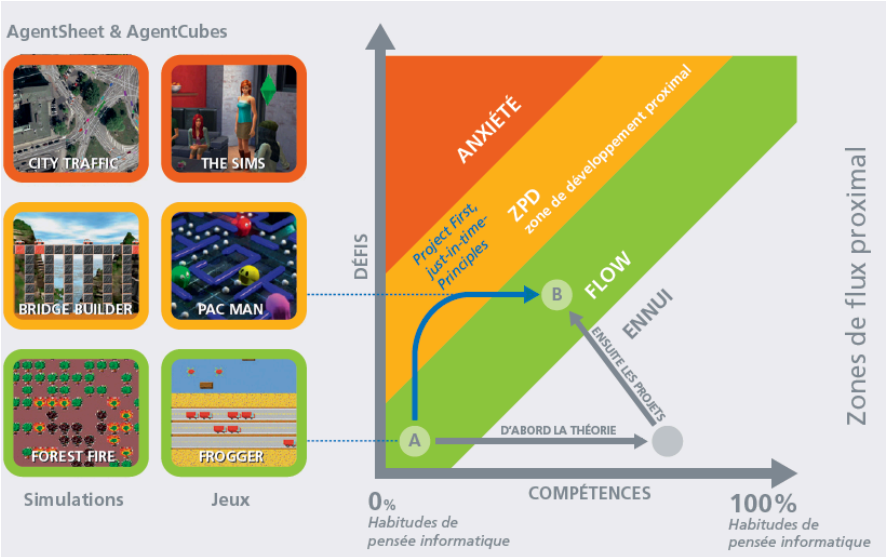


Figure 7 - des défis complexes pour favoriser l'apprentissage.

Le logiciel Scratch et ses possibilités quasi infinies de création doivent donc être mis à profit pour la réalisation de défis complexes, qui permettront de consolider les acquis sur la logique algorithmique et développeront les compétences liées à la pensée informatique. Ils devront induire une véritable démarche de projet avec :

- Une phase d'analyse et de séquençage du problème.
- Une phase créative avec l'imagination du fonctionnement général de l'algorithme amenant la solution.
- Une phase de conception avec écriture, test, analyse des erreurs et débogage.
- Une phase de présentation de son algorithme.

Ces défis pourront par exemple être créés par une partie de la classe et soumis à la résolution de l'autre partie. Le défi proposé en séance 6 est par comparaison peu vecteur d'enrôlement et travaille principalement le repérage de l'élève dans l'espace. Ce n'est donc pas une activité optimale pour faire progresser les élèves. Cette dernière séance doit être avant tout prise comme un pas de plus dans la prise en main d'un logiciel de programmation et la découverte d'une nouvelle logique de communication.

Cette séquence doit être poursuivie pour consolider les premiers acquis et proposer rapidement des défis à la fois plus motivants et plus résistants, seuls à même de développer les différentes compétences de la logique informatique. Ces défis pourront être réalisés en binôme dans une première phase d'analyse du problème et de découpage en sous-problèmes. Ensuite, l'imagination du code et ses modifications successives se feront individuellement pour laisser libre court à la créativité de chacun : l'algorithme produit doit rester avant tout le fruit d'une réflexion et donc d'une façon de traiter les données. Les futures séances s'inscriront ainsi dans une pédagogie de projet et prépareront les activités de programmation d'un robot, qui se dérouleront courant juin 2017.

Conclusion

Malgré de fortes contraintes matérielles liées à la presque absence de matériel informatique dans l'école, la mise en place de cette séquence s'est révélée riche, autant pour la classe que pour moi. La progression de la séquence m'a paru relativement cohérente et a permis d'amener la majorité des élèves vers la compréhension de la notion d'algorithme. Les séances 3 et 4 auraient cependant pu se montrer plus ludiques et abordables, l'introduction d'interdisciplinarité ayant perturbé les élèves. Elles n'ont pas provoqué d'avancées significatives dans la compréhension des notions étudiées mais on

quand même mis en jeu des compétences transversales notamment celles liées aux pratiques langagières et à la construction du nombre.

La séquence venant tout juste de se terminer, le dispositif évaluatif n'a pas encore été mis en place. Il devra déterminer si la classe s'est emparée des procédures propres à la résolution de problèmes.

Maintenant que le nouveau langage lié à la pensée informatique est connu, il va être possible d'utiliser cette syntaxe pour la conception d'algorithmes résolvant des problèmes complexes dans de nombreux domaines (en littérature ou étude de la langue, en numération, en géométrie, en grandeurs et mesures, etc.). L'utilisation du logiciel Scratch ouvre des perspectives nouvelles de travail des connaissances et des compétences dans de nombreux domaines, notamment mathématiques. Elle sera l'occasion de consolider la maîtrise de la logique algorithmique et les compétences propres aux différents domaines.

Le travail sur la pensée informatique présenté entre pleinement dans le cadre des programmes de 2015 et permet la mise en œuvre d'une véritable démarche de projet : il favorise la prise d'initiative, la créativité et l'analyse des activités proposées. Il est donc le vecteur de nombreux apprentissages et favorise le processus de métacognition. De plus, il donne une large part au jeu pédagogique puisque les élèves apprennent en travaillant sur des défis ludiques et interprètent l'erreur comme une étape nécessaire vers la résolution.

Cette première séquence sera suivie d'une nouvelle, entièrement consacrée à la résolution de défis complexes à l'aide de Scratch puis d'une dernière séquence consacrée, elle, à la découverte de la robotique. La classe travaillera sur le robot Thymio et aura à charge d'inventer un défi qui sera ensuite réalisé par le robot. Elle réinvestira ainsi les apprentissages de la présente séquence puisque ce robot accepte des programmes édités avec Scratch et pourra être contrôlé par les algorithmes des élèves. Cet objectif a été annoncé dès la mise en place des premières séances afin de donner du sens à la notion d'algorithme.

Afin de mettre en lumière l'état de métacognition des élèves concernant la pensée informatique, la notion d'algorithme pourra être appliquée à la consolidation des techniques

opératoires (division notamment) vues à l'école élémentaire. Elles seront dès lors perçues comme un objet scientifique à part entière permettant la résolution de calculs complexes.

Bibliographie

(REPENNING, 14). Alexander Repenning, La pensée informatique dans la formation des enseignants. In *Cahiers de la fondation Hasler*, décembre 2014.

(TCHOUNIKINE, 17). Pierre Tchounikine, Initier les élèves à la pensée informatique et à la programmation avec Scratch, Université de Grenoble-Alpes, version 2 du 25 mars 2017.

Sitographie

(EDUSCOL 16). eduscol.education.fr/ressources-2016 - Ministère de l'Éducation nationale, de l'Enseignement supérieur et de la Recherche - Mars 2016, URL : https://cache.media.eduscol.education.fr/file/Initiation_a_la_programmation/92/6/RA16_C2_C3_MATH_inititation_programmation_doc_maitre_624926.pdf

(EVIAN, 16). Equipe de la circonscription d'Evian, PLAIRE : Pensée Logique, Algorithmes et Informatique des Robots d'Evian, 9 octobre 2016, URL : http://www.educavox.fr/innovation/pedagogie/robots-d-evian-2016-une-experimentation-de-programmation-en-milieu-scolaire

(RESNICK, 12). Mitchel Resnick, « Apprenons aux enfants à programmer », 2012, URL : https://www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code?language=fr, consulté le 27/03/2017.

(SOCLE COMMUN 15). Bulletin Officiel n°17 du 23 avril 2015. URL : http://cache.media.education.gouv.fr/file/17/45/6/Socle_commun_de_connaissances,_de_competences_et_de_culture_415456.pdf

Annexes

Annexe 1 : le plan de la séquence

<table>
<tr><td colspan="4">Sciences – Cycle 3 – CM2
Plan de séquence – découverte des algorithmes</td></tr>
<tr><td>Objectifs pédagogiques</td><td colspan="3">- Comprendre la notion d’algorithme
- S’engager dans une démarche de projet en faisant appel à la pensée scientifique
- Découvrir un logiciel de programmation : Scratch
- Comprendre et utiliser la notion de nombre décimal</td></tr>
<tr><td>Compétences socle commun</td><td colspan="3">Mathématiques
- mettre en œuvre un algorithme de calcul posé pour l’addition, la soustraction, la multiplication, la division.
- Résolution de problèmes ; organiser les données
- (Se) repérer et (se) déplacer dans l’espace en utilisant ou en élaborant des représentations.

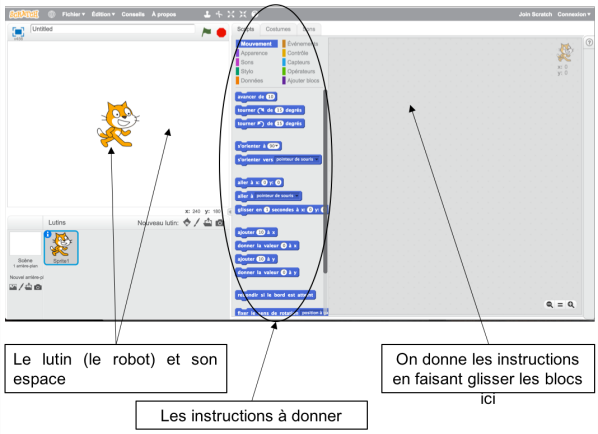
Chercher
- S’engager dans une démarche, observer, questionner, manipuler, expérimenter, émettre des hypothèses, en mobilisant des outils ou des procédures mathématiques déjà rencontrées, en élaborant un raisonnement adapté à une situation nouvelle.
- Tester, essayer plusieurs pistes de résolution.
Raisonner
- Résoudre des problèmes nécessitant l’organisation de données multiples ou la construction d’une démarche qui combine des étapes de raisonnement.
- Progresser collectivement dans une investigation en sachant prendre en compte le point de vue d’autrui.</td></tr>
<tr><td>Objectifs langagiers</td><td colspan="3">- Algorithme ; instructions ; boucle de condition ; variable ; bug ; ambiguë, processus fini</td></tr>
<tr><td>Séance</td><td>Objectif général</td><td>Déroulement</td><td>Remarques</td></tr>
<tr><td>1
Le robot idiot
1ère partie</td><td>Découvrir la notion d’algorithme et d’instructions
Comprendre qu’une</td><td>1) Entrée dans l’activité, recueil des représentations initiales
2) Présentation du jeu du robot idiot
3) Jeu
4) Conclusion et définition d’un</td><td>https://pixees.fr/dis-maman-ou-papa-cest-quoi-un-algorithme-dans-ce-monde-numerique</td></tr>
</table>

	instruction doit comporter une condition d'arrêt	algorithme	
2 **Le robot idiot 2ème partie**	Ecriture de l'algorithme mis en œuvre en séance 1 Découverte de la syntaxe algorithmique	1) Rappels 2) Verbalisation de la tâche : un plan d'un labyrinthe, des carreaux représentant les pas, etc. 3) Ecriture de son algorithme 4) Correction collective et écriture de la trace écrite	
3 **Tri de nombres 1ère partie**	Concevoir collectivement un algorithme pour ordonner des nombres par ordre croissant (informatique en débranché)	Séance filmée pour analyse La classe est divisée en 4 groupes de 7 élèves 1) Chaque groupe répond à la consigne : vous devez vous organiser par ordre croissant le plus rapidement possible (matérialiser le début et la fin) → M chronomètre 2) Discussion collective sur les procédures 3) Même consigne en appliquant un algorithme → M chronomètre	
4 **Tri de nombres 2ère partie**	Réinvestir l'algorithme de tri	En groupe, les élèves doivent retrouver le schéma des cinq étapes de l'algorithme et écrire les instructions permettant à un nombre de se retrouver rangé correctement à la fin de l'algorithme : introduction des boucles conditionnelles.	
5 **Découverte de Scratch**	Découverte du logiciel Scratch : réaliser un code simple pour que le lutin se déplace sur l'écran	Découverte semi-libre du logiciel Scratch (appui sur une fiche d'activité).	

6 **Ecrire d'un algorithme de déplacement**	Concevoir un algorithme pour aider le lutin à sortir d'un labyrinthe Réinvestissement des instructions de déplacement : - Avancer - S'orienter - Tourner	En individuel, codage du déplacement du lutin jusqu'à la sortie d'un labyrinthe.	

Annexe 2 : la fiche d'activité de la séance 5

Sciences - Découverte du logiciel Scratch

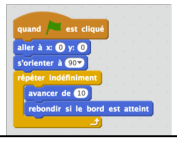


Pour écrire son algorithme :

1) Choisir une condition de départ dans la catégorie « Evènements ».
2) Choisir une action, par exemple « avancer de » dans la catégorie « Mouvement ». Il faut que les blocs sont bien emboîtés.
3) Lorsque l'algorithme est terminé, on le fait fonctionner en appuyant sur le drapeau vert et on observe ce que fait le lutin.

Exemple :

Voici l'algorithme qui demande au lutin d'avancer de 10 pas indéfiniment vers la droite de son espace et de rebondir si le bord est atteint :



A toi de jouer, tu peux créer l'algorithme que tu souhaites. Pour t'aider à découvrir toutes les instructions, voici des propositions :

Algorithme 1

1) Faire démarrer l'algorithme quand le drapeau vert est cliqué
2) Positionner le lutin en x = 0 et y = 0 (au centre de sa zone) grâce à l'instruction « aller à x : 0 y : 0)
3) Faire attendre le lutin 2 secondes
4) Orienter le lutin vers la droite (instruction « s'orienter à 90° »)
5) Faire attendre le lutin 2 secondes
6) Faire avancer le lutin de 50 pas vers la droite
7) Faire attendre le lutin 2 secondes
8) Faire tourner le lutin de 90° vers le bas
9) Faire attendre le lutin 2 secondes
10) Faire avancer le lutin de 50 pas

Algorithme 2

1) Faire démarrer l'algorithme quand la touche « espace » est enfoncée
2) Positionner le lutin en x = 0 et y = 0 (au centre de sa zone)
3) Faire attendre le lutin 2 secondes
4) Orienter le lutin vers la droite (instruction « s'orienter à 90° »)
5) Faire attendre le lutin 2 secondes
6) Faire dire au lutin : « bonjour » pendant 3 secondes
7) Faire attendre le lutin 2 secondes
8) Faire avancer le lutin de 5 pas indéfiniment jusqu'à toucher la bande verte
9) Faire attendre le lutin 2 secondes
10) Faire dire au lutin : « arrivé ! »

Lorsque tu as terminé un algorithme, n'oublie pas de l'enregistrer : fichier → télécharger dans votre ordinateur → nommer le fichier « prénom-numéro… »

Annexe 3 : la fiche d'activité de la séance 6

Sciences - Découverte du logiciel Scratch

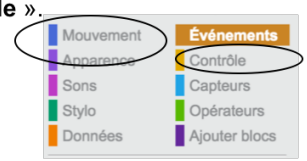
Objectif de la mission : aider le chat à sortir du labyrinthe

Pour réaliser cette mission, tu vas devoir écrire un algorithme pour que le chat se déplace sans traverser les murs !

Quelques règles :

1) Pour t'aider dans ta mission, le maître a écrit le début de l'algorithme, tu ne dois pas modifier ces blocs :

2) Tu vas devoir utiliser les instructions des catégories « **apparence** », « **mouvement** » et « **contrôle** ».



3) Pour faire tourner le chat, tu peux utiliser soit l'instruction 

ou bien tu pourras utiliser les instructions 

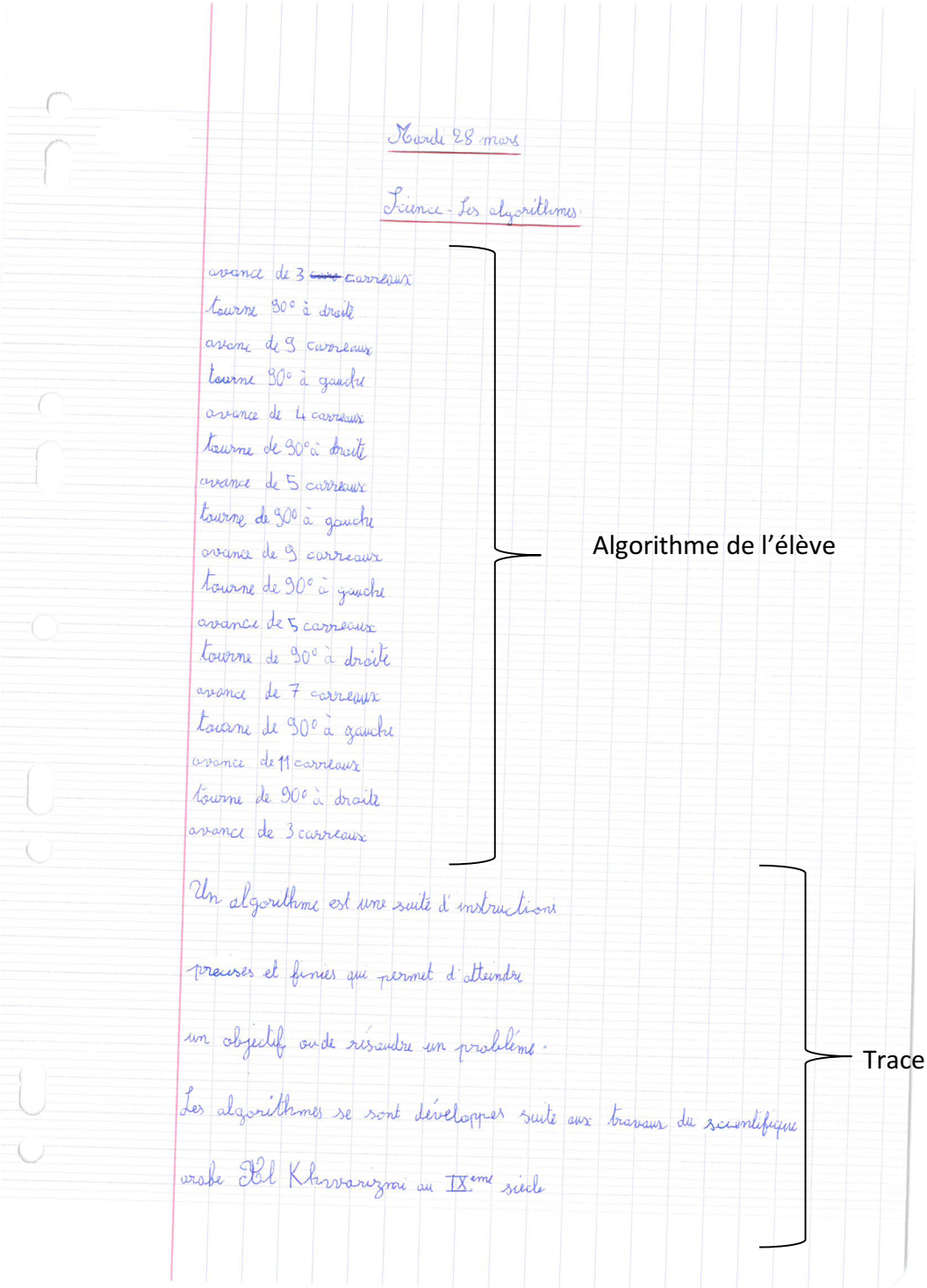
4) Chaque fois que tu décides de faire tourner le chat, tu dois placer une instruction « attendre 1 seconde » avant ET après :

5) Lorsque le chat est arrivé à la sortie, il doit dire : « Merci de m'avoir aidé, jeune Padawan ! »

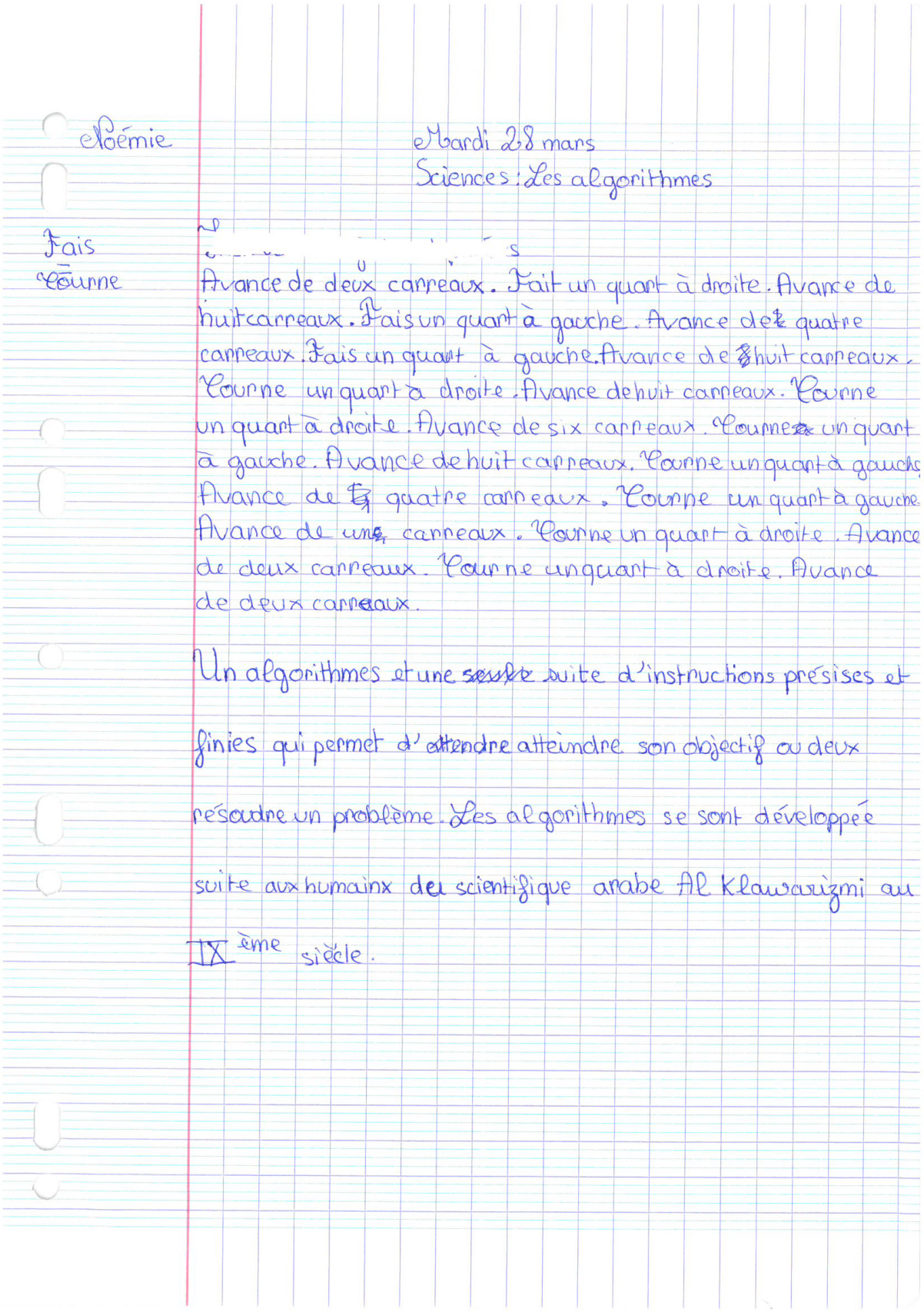
Bonne chance !

Annexe 4 : des productions d'élèves de la séance 2 (algorithme de déplacement d'un robot).

Elève 1, peu familier des environnements numériques, en réussite ici :



Elève 2, peu familier des environnements numériques, en difficulté ici :



Noémie

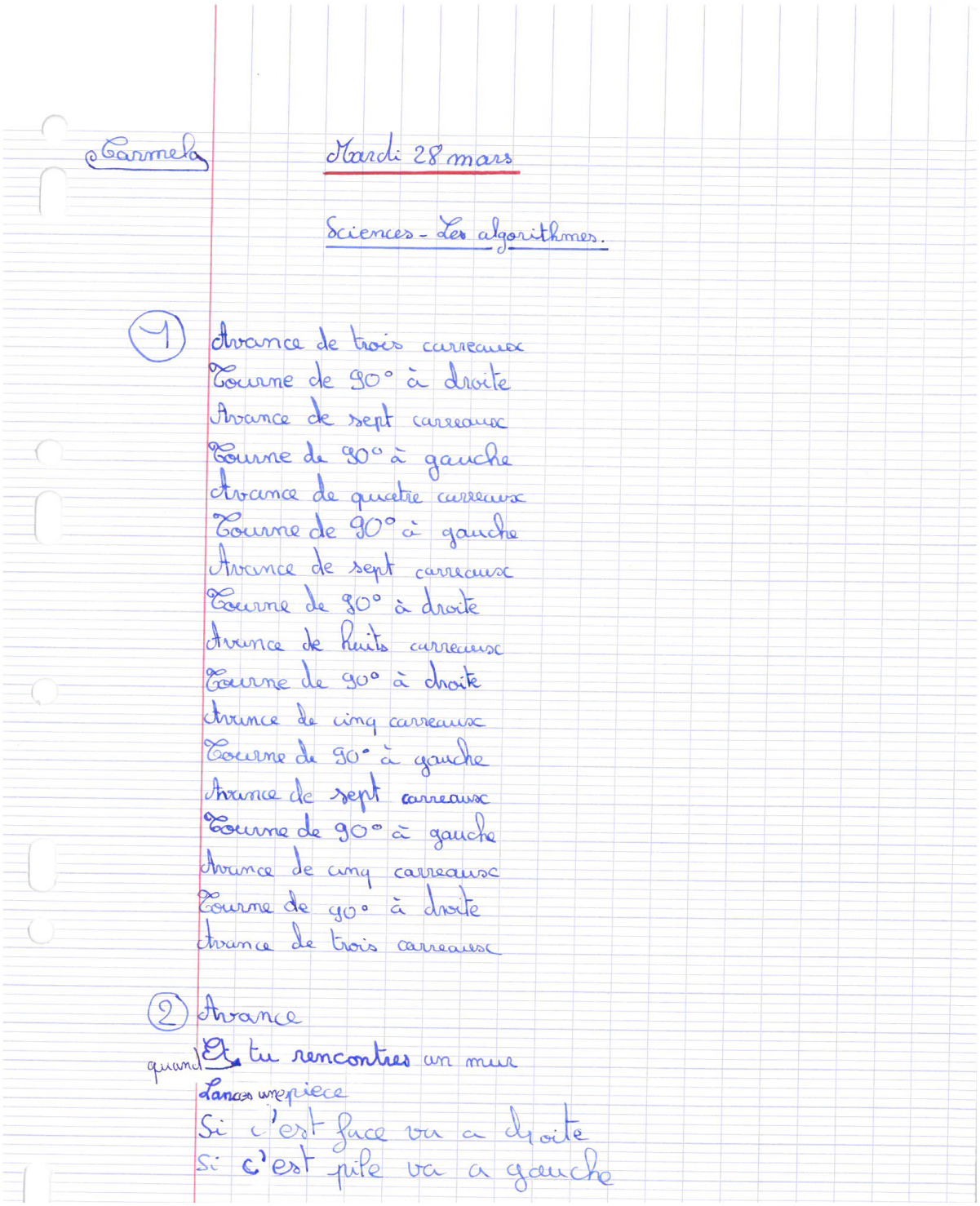
Mardi 28 mars
Sciences : Les algorithmes

Fais
Tourne

Avance de deux carreaux. Fait un quart à droite. Avance de huit carreaux. Fais un quart à gauche. Avance de quatre carreaux. Fais un quart à gauche. Avance de huit carreaux. Tourne un quart à droite. Avance de huit carreaux. Tourne un quart à droite. Avance de six carreaux. Tourne un quart à gauche. Avance de huit carreaux. Tourne un quart à gauche. Avance de quatre carreaux. Tourne un quart à gauche. Avance de une carreaux. Tourne un quart à droite. Avance de deux carreaux. Tourne un quart à droite. Avance de deux carreaux.

Un algorithmes et une ~~sesuite~~ suite d'instructions présises et finies qui permet d'~~attendre~~ atteindre son objectif ou deux résoudre un problème. Les algorithmes se sont développée suite aux humainx des scientifique arabe Al Khawarizmi au IX ème siècle.

Elève 3 ayant utilisé une boucle de condition :



Carmela

Mardi 28 mars

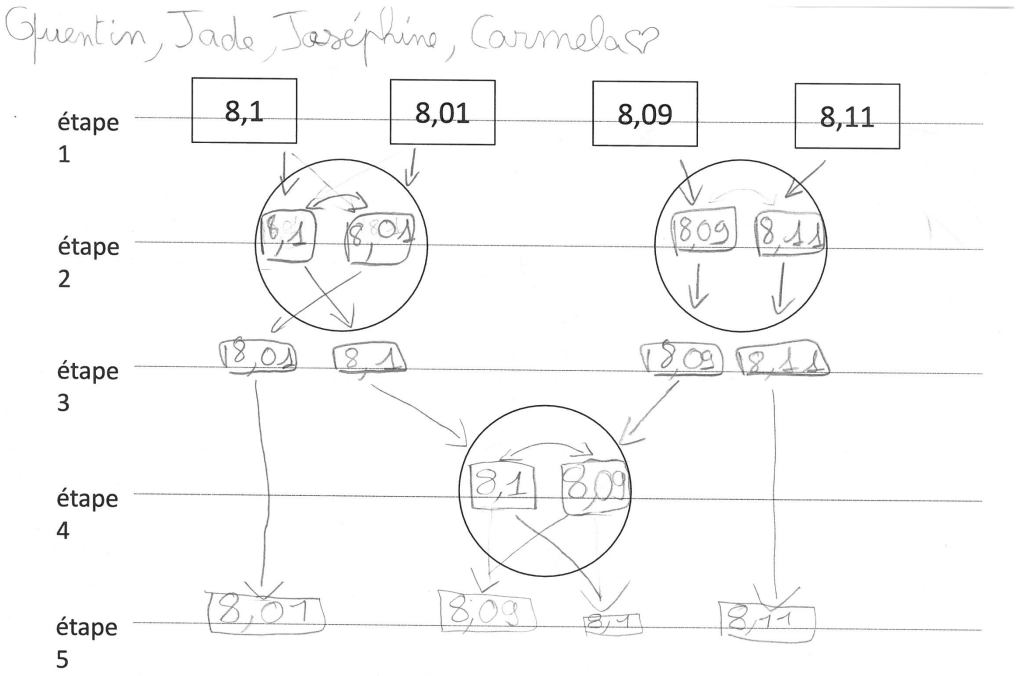
Sciences - Les algorithmes.

1) Avance de trois carreaux
Tourne de 90° à droite
Avance de sept carreaux
Tourne de 90° à gauche
Avance de quatre carreaux
Tourne de 90° à gauche
Avance de sept carreaux
Tourne de 90° à droite
Avance de huits carreaux
Tourne de 90° à droite
Avance de cinq carreaux
Tourne de 90° à gauche
Avance de sept carreaux
Tourne de 90° à gauche
Avance de cinq carreaux
Tourne de 90° à droite
Avance de trois carreaux

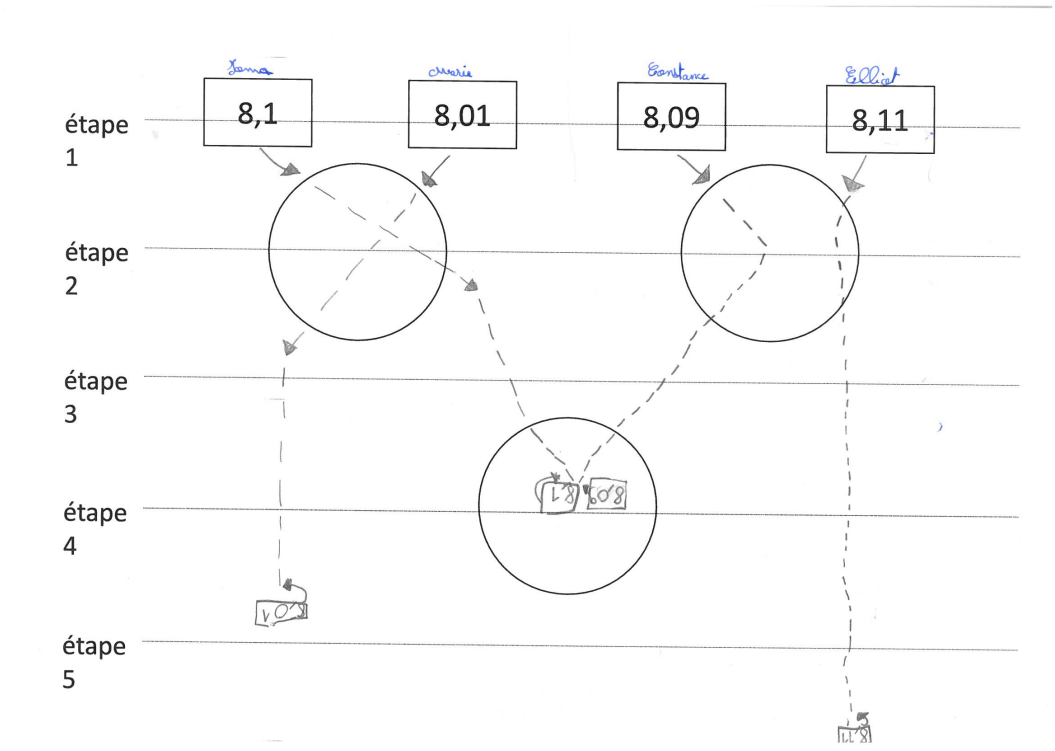
2) Avance
quand tu rencontres un mur
Lance une pièce
Si c'est face va à droite
Si c'est pile va à gauche

Annexe 5 : des productions d'élèves de la séance 4 (algorithme de tri de nombres décimaux).

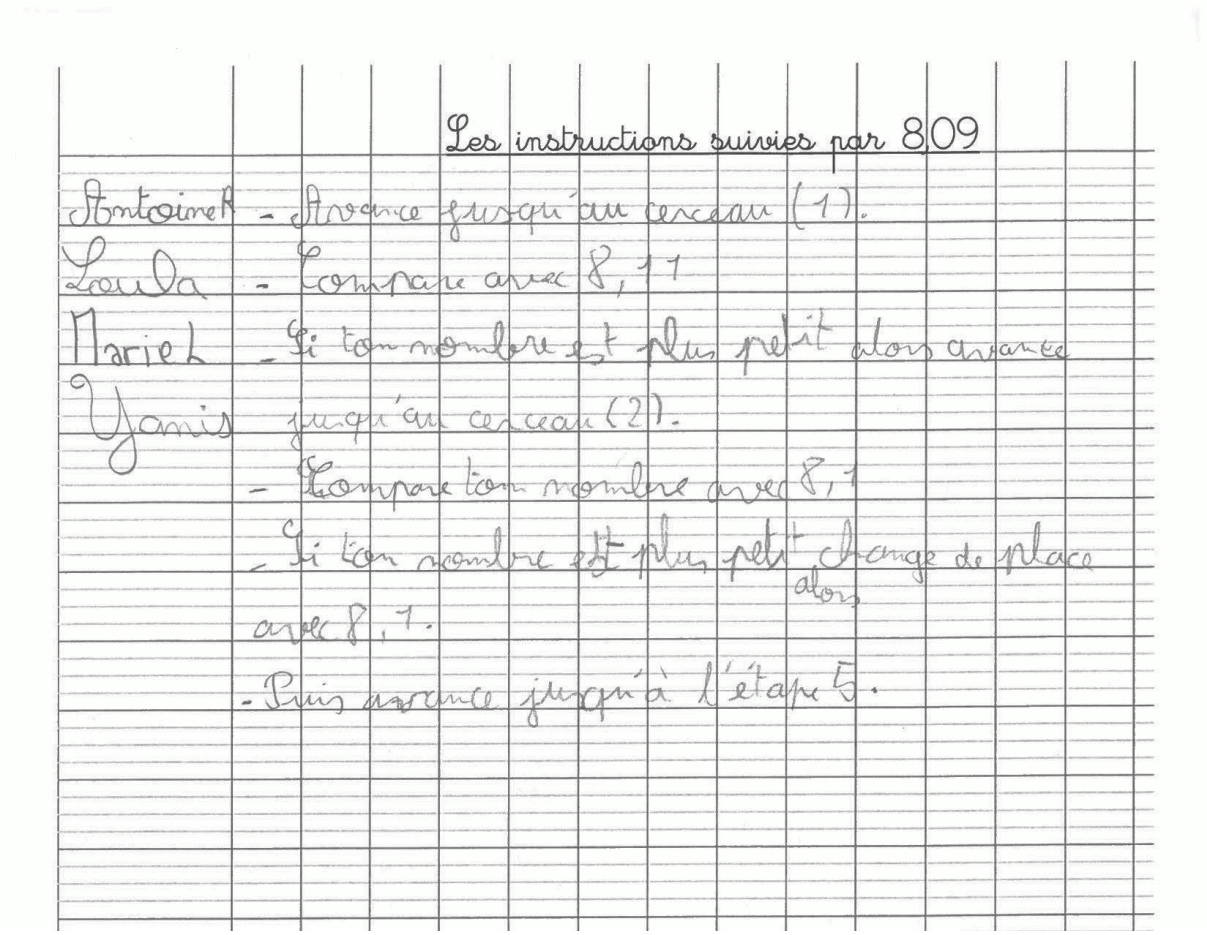
Groupe d'élève 1, non familiers des environnements informatiques, en réussite :



Groupe d'élèves 2, non familiers des environnements informatiques, en difficulté :



L'algorithme du groupe 3, utilisant une boucle de condition avec comparaison :

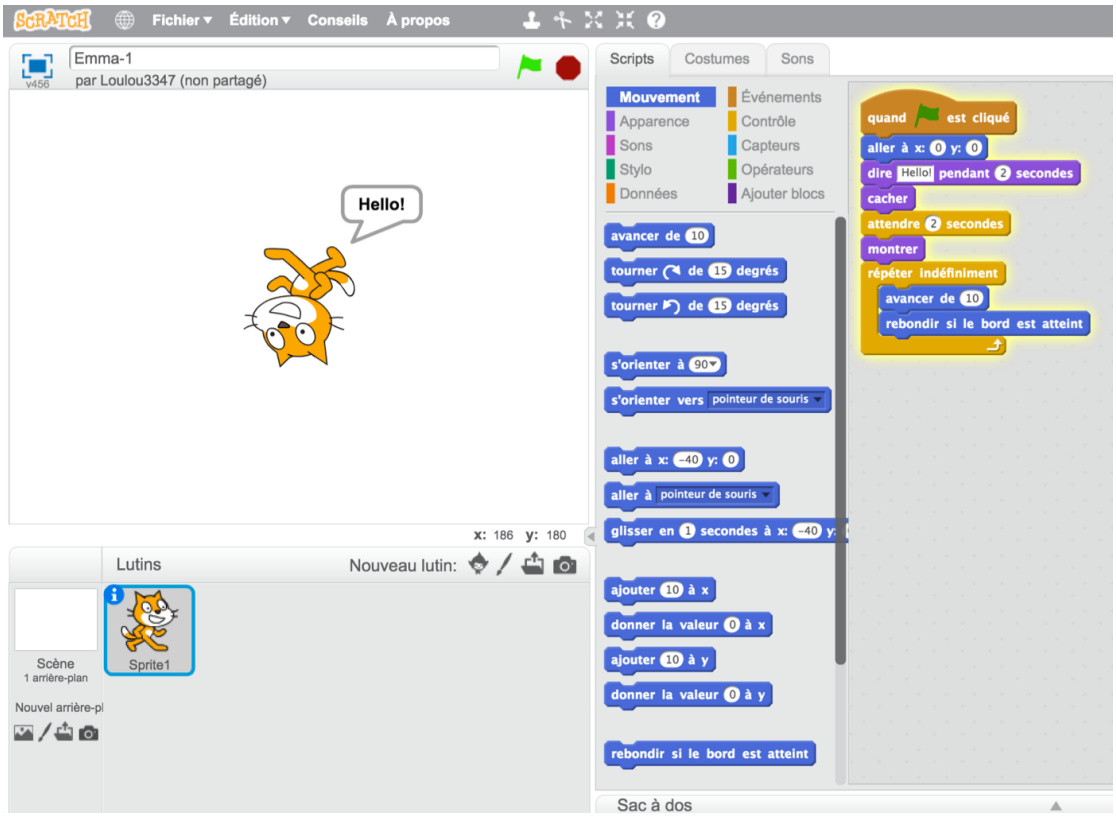


Les instructions suivies par 8,09

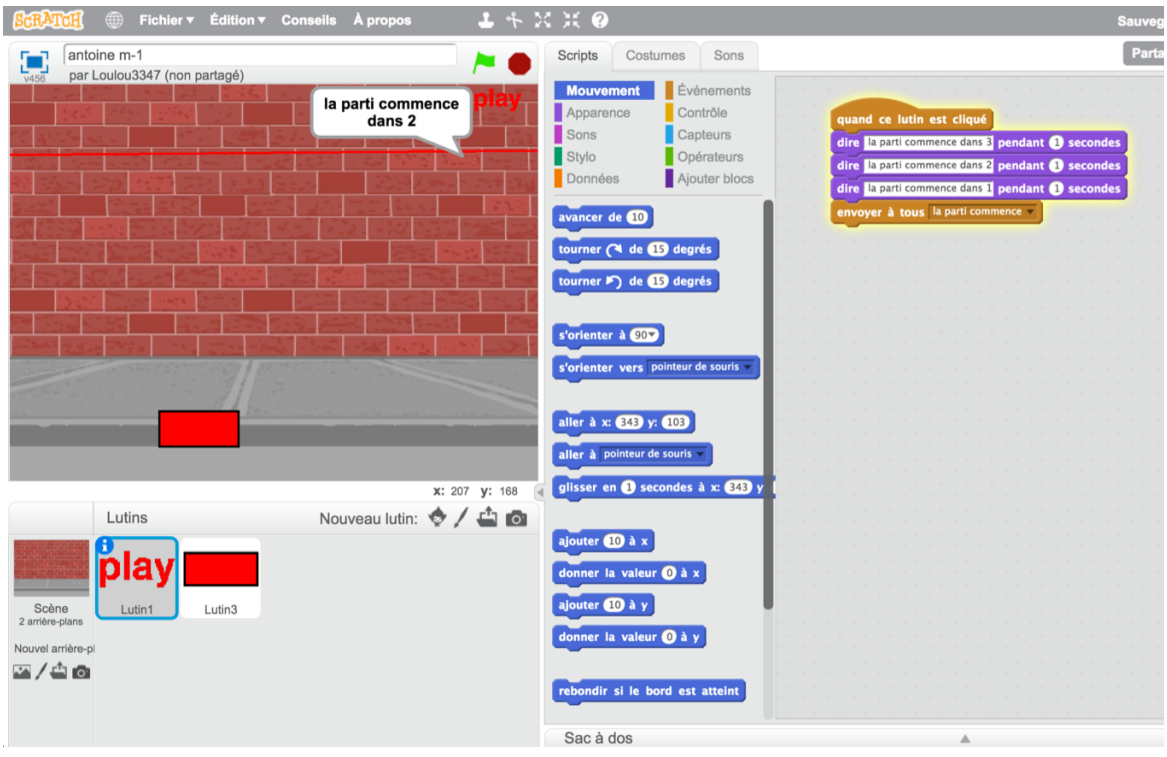
Antoine R - Avance jusqu'au cercaau (1).
Louda - Compare avec 8,11
Mariel - Si ton nombre est plus petit alors avance
Yanis jusqu'au cercaau (2).
- Compare ton nombre avec 8,1
- Si ton nombre est plus petit alors change de place avec 8,1.
- Puis avance jusqu'à l'étape 5.

Annexe 6 : des algorithmes libres suite à la séance 5.

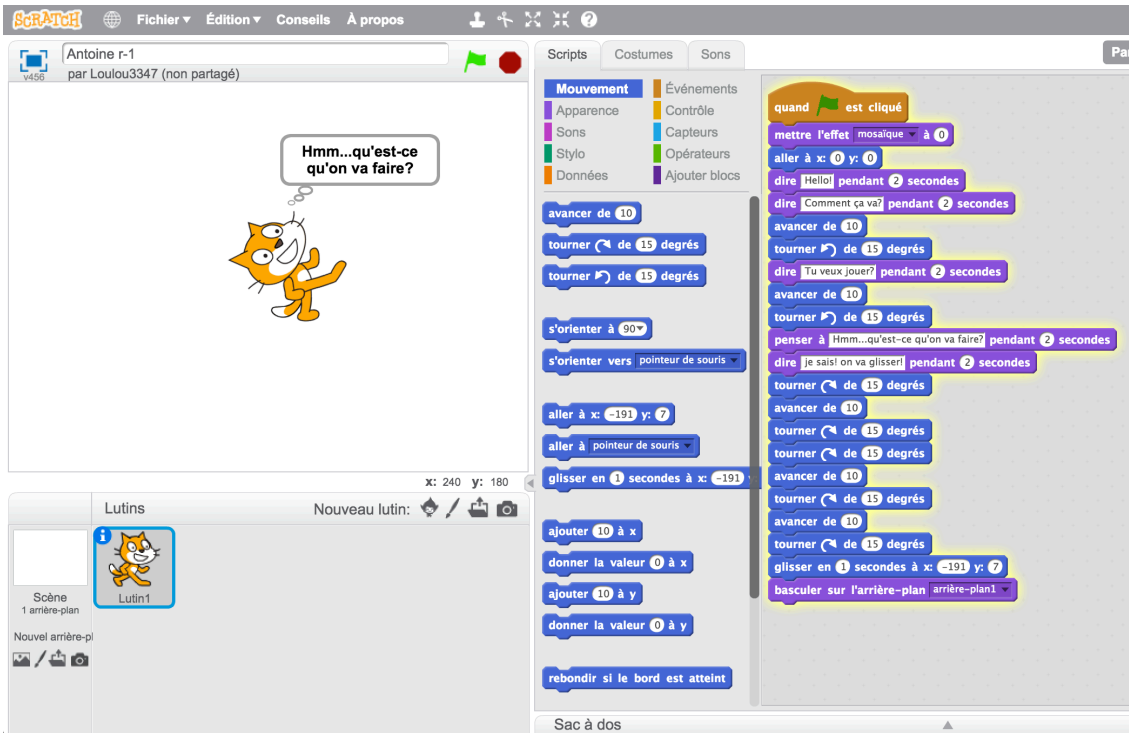
Elève 1



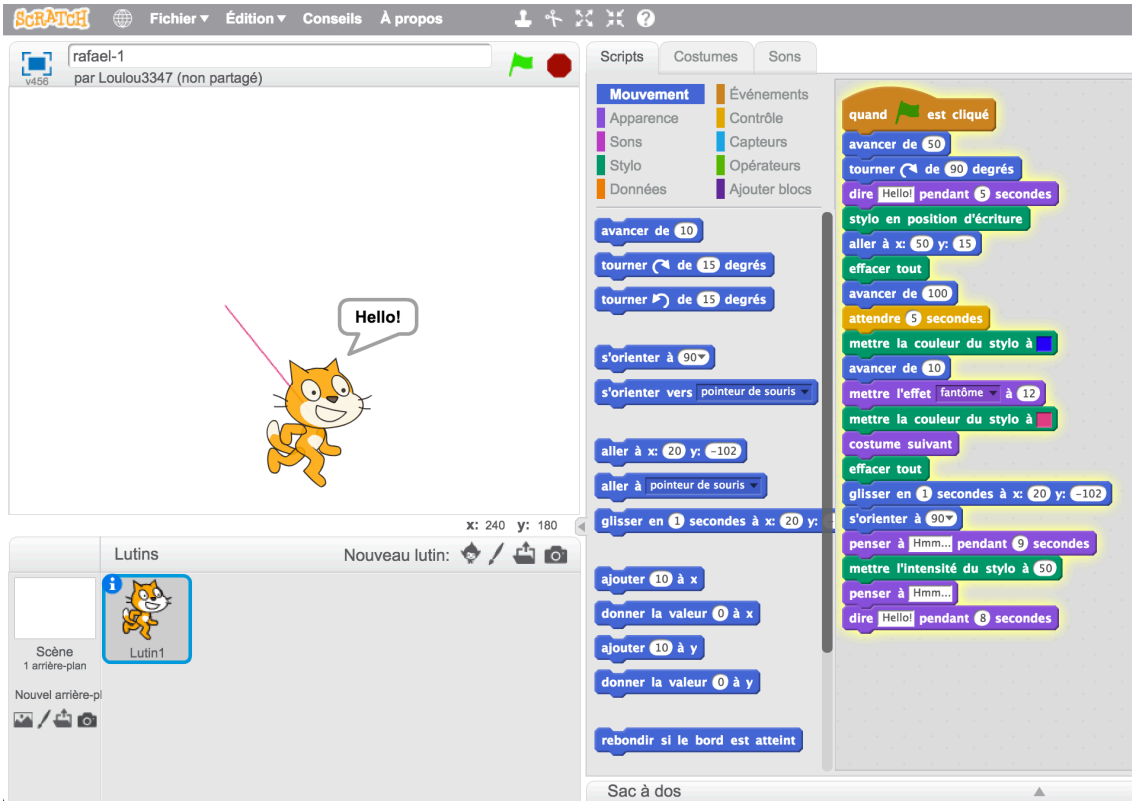
Elève 2



Elève 3

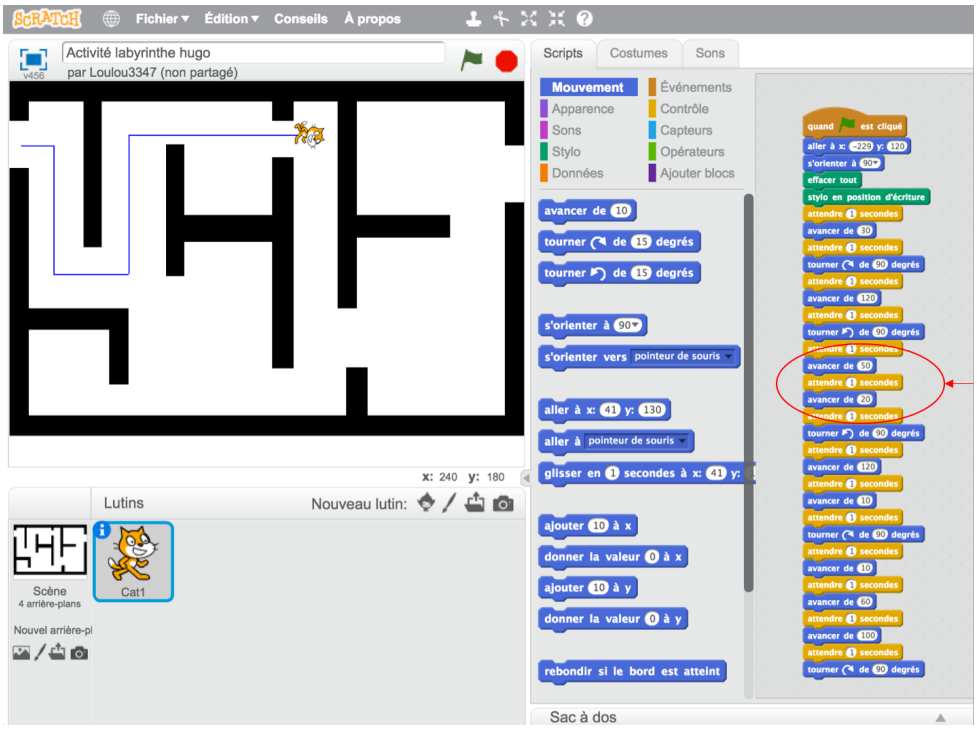


Elève 4

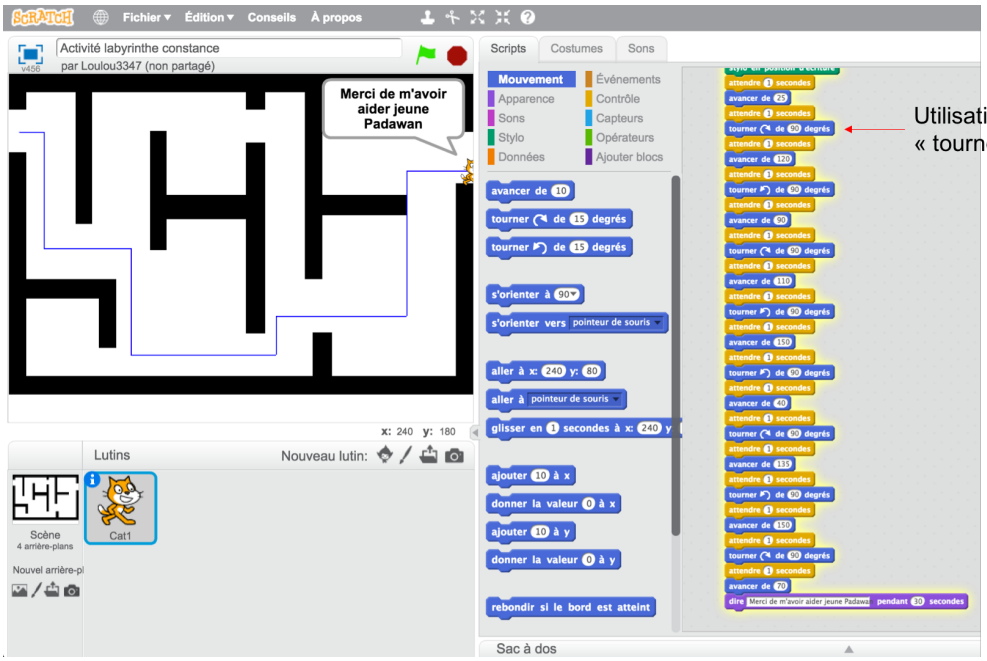


Annexe 7 : des algorithmes d'élèves pour coder le déplacement du lutin dans un labyrinthe.

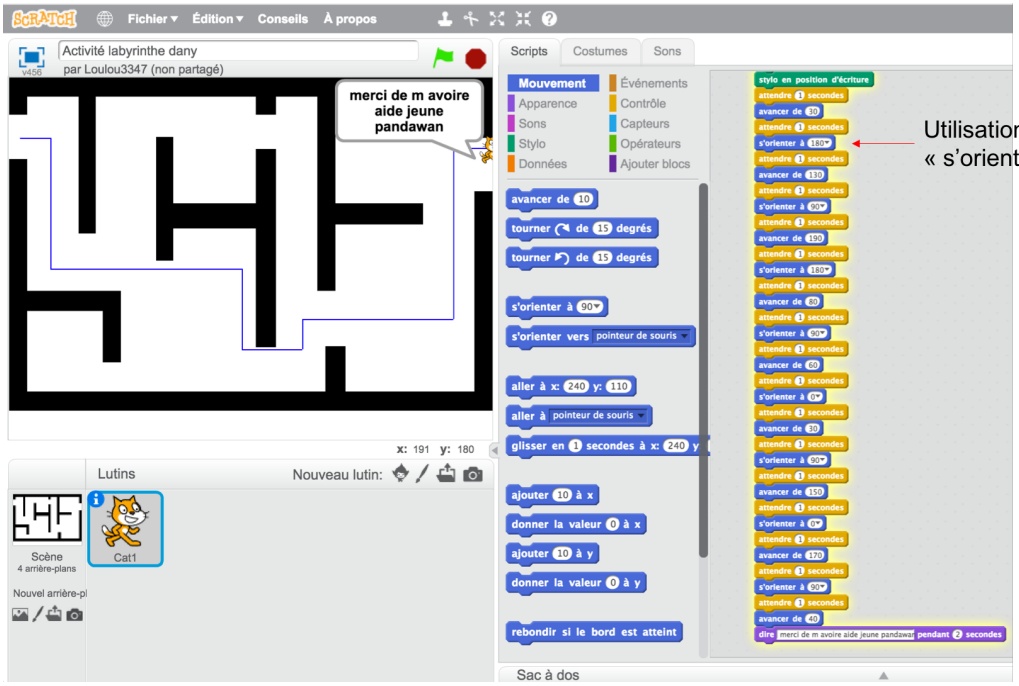
Elève 1



Elève 2



Elève 3



Utilisation de l'instruction « s'orienter »