Analyse comparative d'une activité d'apprentissage de la programmation en mode branché et débranché

Margarida Romero, Benjamin Lille, Thierry Viéville, Marie Duflot-Kremer, Cindy de Smet, David Belhassein

Analyse comparative d'une activité d'apprentissage de la programmation en mode branché et débranché

Margarida Romero[1], Benjamin Lille[2], Thierry Viéville[1,4], Marie Duflot-Kremer[3], Cindy De Smet[1], David Belhassein[5]

[1] Université Côte d'Azur, Laboratoire d'Innovation et Numérique pour l'Éducation (LINE), [2] Université Laval, Kids Code Jeunesse, [3] Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France, [4] Inria, Mnémosyne, [5] Académie de Nice.

margarida.romero@unice.fr, benjamin.lille.1@ulaval.ca, marie.duflot-kremer@loria.fr, thierry.vieville@inria.fr, cdesmet@unice.fr, david.belhassein@ac-nice.fr

Résumé. L'introduction de la programmation à l'école peut être un levier pour développer la pensée informatique en lien avec une démarche de résolution de problèmes. Dans ce contexte, nous nous intéressons aux différents types d'activités d'apprentissage de la programmation dans le but d'établir un protocole pour comparer les activités branchées et débranchées à l'école, et plus particulièrement de voir dans quelle mesure une activité débranchée permet un transfert de compétences vers l'apprentissage de la programmation. Nous discutons la méthodologie et les résultats en lien aux observations réalisées au cours des activités de formation Class'Code.

Mots-clés. Programmation, Code, Pensée informatique, Informatique débranchée, Scratch.

1 Introduction

L'apprentissage de l'informatique à l'école a connu une évolution importante au cours des dernières décennies (Baron & Bruillard, 2013). Au cours des dernières années, l'apprentissage de la programmation s'est popularisé à l'école, d'une part, grâce à l'accessibilité d'outils de programmation visuelle comme Scratch (Resnick et al., 2009), et d'autre part, par la prise de conscience du besoin d'appréhender et de démystifier le numérique pour permettre aux citoyens de développer une approche critique et créative face à ses enjeux. Les activités de programmation ont d'ailleurs déjà commencé à être intégrées dans les curriculums officiels en France, en Angleterre, dans certaines provinces canadiennes comme la Colombie-Britannique et bien d'autres pays (Heintz, Mannila, & Färnqvist, 2016). Nous pouvons considérer que pour comprendre le numérique, il faut connaître certains de ses principes, ses usages et ses enjeux. Dans le but de dépasser une appréhension du numérique comme un ensemble de connaissances techniques et procédurales, Wing (2006) propose le concept de pensée informatique (*computational thinking*) comme la capacité d'utiliser des méthodes et concepts informatiques pour résoudre des problèmes. Le concept de pensée informatique fait appel à des stratégies de résolution de problèmes dans plusieurs domaines. Depuis la proposition du concept de pensée informatique, de nombreuses études ont été réalisées pour la conceptualiser et l'évaluer dans le cadre de différentes activités d'apprentissage de la programmation (Grover & Pea, 2013). Le développement de la pensée informatique peut entraîner la découverte de notions inédites comme celles liées aux algorithmes ou au codage de l'information. Au moment de mettre en place des activités d'apprentissage de la programmation, les enseignants pensent souvent aux besoins de matériel informatique (Romero & Netto, 2018). Mais a-t-on besoin d'un ordinateur pour s'initier à la pensée informatique ? Il se trouve que la réponse n'est pas triviale et que des activités de type débranché qui se font en transposant les notions informatiques à travers la manipulation d'objets du quotidien et le mouvement des apprenants se sont révélées très prometteuses. Dans cette étude nous présentons un protocole pour étudier la pensée informatique dans un contexte d'apprentissage de la programmation avec des activités débranchées et avec le logiciel de programmation visuelle Scratch. Le choix de Scratch est motivé par la popularité de cette plateforme au niveau de l'enseignement primaire au niveau international (Lye & Koh, 2014).

2 Apprentissage de la programmation par des activités débranchées

2.1 Apprentissage de la programmation créative

Les effets des activités de programmation sur le développement des concepts et des processus informatiques sont à questionner à plusieurs niveaux. L'apprentissage de la programmation de manière procédurale permet d'apprendre une certaine séquence d'instructions, mais il ne garantit pas le développement de la pensée informatique (Romero, Noirpoudre, & Viéville, 2018). Quand on dépasse le simple apprentissage de la programmation, par exemple à partir d'activités débranchées ou en orientant le travail vers l'apprentissage de la

pensée informatique, on peut alors arriver à établir un véritable effet positif, au niveau primaire et début de collège, y compris avec des enseignants nouvellement formés (Moreno-León & Robles, 2015). Le levier est le passage de l'apprentissage procédural de la programmation à l'intégration interdisciplinaire de la programmation créative (Resnick & Siegel, 2015; Romero, 2016). Il s'agit de passer au delà du simple apprentissage de la programmation pour envisager un apprentissage par le biais de la programmation qui puisse permettre le développement de la pensée informatique. L'approche procédurale engage l'élève dans une séquence de programmation où les paramètres sont fixés par l'enseignant tandis que l'approche créative offre une marge de créativité à l'élève tant au niveau de la procédure que du produit créé (Romero, 2016). La pertinence du développement d'activités d'apprentissage de la programmation pour le développement de la pensée informatique est bien établie dans différentes études (Grover & Pea, 2013) où les auteurs reportent de effets positifs sur par exemple la capacité à résoudre des problèmes (au sens de Torp, 2002), et dans une moindre mesure le raisonnement et la spatialisation. Ces résultats ont pu être établis au niveau de groupes d'étudiants universitaires (donc des futurs enseignants) et aussi d'élèves du secondaire. Une révision de la bibliographie sur différentes études autour de l'apprentissage de la programmation est disponible en ligne par le biais du projet Class'Code (Romero et al., 2018).

2.2 Programmation débranchée : une diversité d'approches pédagogiques

Devant l'émergence et la diversité des activités d'apprentissage de la programmation à l'école, un nombre croissant d'études se sont intéressées à la pertinence et l'efficacité de ces différentes approches. Cette appréhension peut se faire par le biais de l'outil informatique, mais également par des activités débranchées qui mobilisent des concepts et des processus informatiques, de Bell, Witten et Fellows (1998) à Duflot (2016), il existe une grande diversité d'activités débranchées. D'une part, on peut proposer des activités débranchées de manière très procédurale : on fournit un mode d'emploi à suivre sur lequel les élèves doivent suivre des instructions préétablies. D'autre part, on peut aussi proposer de telles activités sous forme de "tour de magie". Typiquement un tour de cartes dont l'explication repose sur un algorithme impossible à deviner, qui ne se révèle que quand on fournit le "truc à savoir". Ces deux situations extrêmes ne sont pas optimales. Notre expérience est qu'il est préférable de proposer une activité de recherche avec des jalons atteignables, comme mis en œuvre après expérimentation à grande échelle par Calmet, Hirtzig et Wilgenbus (2016). On évite aussi les longues explications à retenir avant de commencer, et on met immédiatement les personnes en situation : l'un fait le robot, se lève se positionne, l'autre fait la programmatrice et donne les instructions et on dévoile l'activité au fur et à mesure de l'action. On choisit souvent une modalité minimale pour commencer, puis à l'instar des jeux vidéo à niveaux on enrichit l'activité d'enjeux un peu plus complexes. Bien entendu il ne faut pas se limiter à faire l'activité, mais prendre le temps du recul, expliquer le lien avec la notion à s'approprier, peut-être inclure un élément historique illustratif, comme mis en place par Viéville et Tort (2013). La prise de parole, par exemple sous forme de discussion ou de questions-réponses permet de récolter des informations complémentaires sur l'activité (Duflot et al., 2015).

Un autre aspect est lié à la construction ou la mise en place des objets du quotidien qui vont permettre de faire l'activité (par exemple organiser les chaises pour faire un labyrinthe au robot, ou construire un graphe sur lequel on se promènera en exécutant un algorithme). Il est très intéressant d'impliquer les élèves dans cette étape (ou de leur proposer d'animer ensuite l'activité), pour les rendre actrices et acteurs de leur propre apprentissage, et on sait combien l'engagement est un levier majeur. Il est aussi très important que ces activités soient contaminantes, au sens où les apprenants actuels peuvent devenir les animateurs de demain. Donner envie de devenir celle ou celui qui enseigne est vraiment un levier en termes d'engagement pour certains élèves. Nous avons constaté cela lors des actions de terrain, et cela a été confirmé par des approches telles que l'apprentissage orienté objet (Hannan, Chatterjee, & Duhs, 2013).

2.2 Le potentiel des activités de programmation débranchée en éducation

Les bénéfices en matière didactique des activités débranchées sont abordés par exemple dans Wohl, Porter et Clinch (2015) ou Brackmann et al. (2017). L'étude de Wolf et ses collaborateurs a permis de tester l'apprentissage de compétences de compréhension de la notion d'algorithme (mesuré par la capacité à décrire une procédure), de la prédiction logique et débogage (*debugging*) avec des enfants de 5 à 7 ans, et montre un véritable apprentissage de ces notions, plus particulièrement avec des activités débranchées (sans qu'il y ait une étude comparative explicite). La seconde étude, sur des compétences similaires (décomposition d'un problème,

reconnaissance de structure, conception d'algorithmes, abstraction d'un processus d'un contexte à un autre) et avec des enfants de 10 à 12 ans, établit de manière significative l'apport didactique des activités débranchées par rapport à un groupe contrôle. Nous prenons en considération les aspects didactiques et nous appuyons sur le référentiel de Curzon, Dorling, Ng, Selby et Woollard (2014) et du travail didactique en amont de Calmet, Hirtzig et Wilgenbus (2016). Les travaux de Bell, Alexander, Freeman et Grimley (2009) ont d'ailleurs permis le développement d'un curriculum de programmation débranchée (https://csunplugged.org) et permis de poser une définition précise que nous résumons ici. La programmation débranchée (ou *unplugged computing*) vise la découverte, voire même l'acquisition de concepts informatiques sans l'utilisation d'outils numériques. L'apprentissage de l'informatique sous forme débranchée ne se limite pas à un niveau basique de compréhension d'un algorithme simple. Cela inclut par exemple avec Duflot (2016), la compréhension d'une machine programmable et avec Calmet, Hirtzig et Wilgenbus (2016), des notions liées aux données et leur représentation, aux réseaux et à la robotique. Pour chacun de ces aspects une étude serait à monter. Dans ce contexte, les activités de programmation débranchées font appel aux interactions entre l'élève et son environnement spatial dans le cadre d'une activité qui doit être porteuse de sens pour le développement de l'activité (Shelton, 2016). Les travaux de recherche portant sur les effets de la programmation débranchée sont actuellement peu nombreux, mais apportent des éclairages pertinents. Les recherches de Faber, Wierdsma, Doornbos, van der Ven et de Vette (2017) ont pour leur part porté sur le design du dispositif d'enseignement lors d'activités de programmation débranchée. Ils émettent des recommandations comme la prise en compte de la disparité du niveau de compétences des élèves dans la conception des activités débranchées afin de proposer des activités à complexité variée. Ils recommandent aussi de clairement expliciter, après une activité débranchée, la façon dont le concept sera déployé lorsque les élèves travailleront en mode numérique. Depuis une perspective enseignante, les activités débranchées amènent aussi les enseignants à développer leur sentiment de confiance envers l'informatique ainsi que leur compréhension des concepts relatifs à la pensée informatique. Ils acquièrent également des techniques d'enseignement relatives à l'introduction de la pensée informatique à intégrer dans leur pratique (Curzon et al., 2014). Le sentiment de confiance des enseignants par rapport au domaine de l'informatique est particulièrement important dans un contexte où la programmation fait son entrée dans les curriculums officiels. Les travaux de Wohl, Porter et Clinch (2015) sont ceux qui s'apparentent le plus aux objectifs de notre recherche. Ils ont comparé l'efficacité entre les activités débranchées, la programmation tangible avec des Cubelets et la programmation sur interface numérique comme Scratch. Ils avancent que la programmation tangible a suscité le plus d'engagements chez les élèves, mais que c'est grâce aux activités débranchées que les élèves ont développé une plus grande compréhension des concepts d'algorithme, de données et de leur traitement, de prédiction logique et de débogage, selon le référentiel de (Curzon et al, 2014)

2.3 Les plus-values de la programmation débranchée

Nous présentons ici deux aspects différentiels de la programmation débranchée par rapport à la programmation réalisée avec des outils de programmation visuelle sur un dispositif numérique.

1) *La charge cognitive liée à l'usage d'une machine*. Lors des activités branchées, la machine demande un apprentissage technique non négligeable et intègre une charge cognitive considérable. À l'inverse, une activité d'informatique débranchée est moins surprenante pour les élèves et les enseignants, car de telles activités ludiques sont pratiquées par ailleurs sur d'autres sujets. Cela simplifie le travail en groupe ou en classe entière tout en évitant les petits problèmes techniques sans rapport avec les notions étudiées. Utiliser la machine impose une charge cognitive (Sweller, 1994) qui peut limiter la réflexion sur les grands principes. En pratique, certains élèves ont également des difficultés à écouter les consignes ou à interagir entre eux quand ils travaillent sur ordinateur, tant l'écran peut focaliser leur attention. Ces faits ont été constatés lors des expérimentations mises en place lors de la création du manuel «1,2,3 codez» (Calmet et al., 2016). D'autre part, en faisant du débranché il est possible de distinguer plus facilement la compréhension des concepts, de l'apprentissage des usages d'un outil technologique.
2) *La cognition incarnée*. Le fait de jouer avec son corps et d'apprendre sous forme de gestes ou d'actions concrètes est une activité engageante aussi bien d'un point de vue physique que cognitif, puisqu'on met en jeu les mémoires procédurales, et épisodiques (il y a souvent une scénarisation de l'activité) en plus des mémoires sémantiques en les faisant interagir. C'est un constat quotidien qui est confirmé par des études telles que Owen et collaborateurs (2016) pour la mémoire procédurale, tandis que ce type de lien entre mémoire épisodique et sémantique est bien établi (Tulving, 1972).

3) *L'analogie tangible*. Il nous semble que le point principal est le fait de construire une analogie tangible des notions abstraites rencontrées en informatique. Cette notion de "métaphore" permet de créer une situation concrète dans laquelle on s'approprie des mécanismes qui vont servir de base à la construction d'une représentation de la notion abordée. En désignant une chose par une autre qui partage avec elle une qualité essentielle, on offre une chance de prendre du recul et de jeter un regard diversifié sur l'objet de l'apprentissage. Tout aussi important est le moment où la métaphore arrive à ses limites : dès que l'apprenant dit « ce n'est pas pareil », le pari est gagné, la réflexion sur le sujet est lancée. Par exemple, on explique le fonctionnement du protocole de transport TCP/IP en faisant jouer à transmettre des messages avec des post-its au sein de la classe, permettant de manipuler concrètement la notion d'adressage, de connectivité, la nécessité d'accusés de réception et de relance sur minuterie, jusqu'à réaliser que les datagrammes qui circulent sur internet doivent bénéficier de fonctionnalités supplémentaires. Cet aspect reste à mieux analyser, par exemple à partir des travaux tels que Sander (2000).

3 Objectifs de recherche

L'objectif de cette étude est l'analyse de l'intérêt des activités débranchées pour le développement de la pensée informatique et l'usage créatif de la programmation visuelle. Dans le but d'analyser l'influence d'une activité de programmation débranchée sur le développement de la pensée informatique, un protocole quasi-expérimental est mis en place auprès d'élèves n'ayant pas réalisé encore d'activités de programmation informatique. L'ensemble des participants va réaliser un test d'évaluation adapté du Computational Thinking Test (CTT, Román-González, Pérez-González, & Jiménez-Fernández, 2017) pour les enfants de l'école primaire. Nous nommons teen-CTT (tCTT) cette variante visant à être utilisée avec des élèves de 10 ans et plus.

Au niveau didactique de l'informatique, les compétences mises en jeu ici sont la compréhension de la séquence d'instruction et son codage sous forme pictographique (c'est le niveau 1 du référentiel de (Cuzon et al, 2014)[1]. Cela inclut par exemple de savoir vérifier un programme simple, et il y a aussi le déploiement d'un raisonnement logique très court sur une des questions pour prédire le résultat.

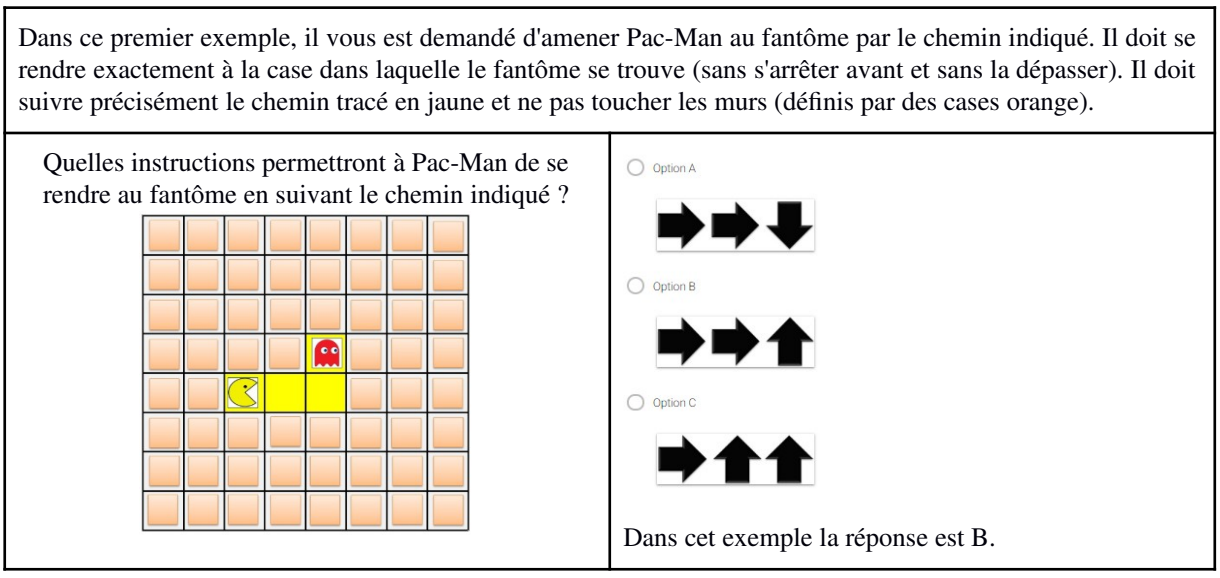

Figure 1. Exemple de question du test de pensée informatique (CTT).

Le test est proposé à l'ensemble des élèves participants en pré-test et en post-test afin d'analyser les effets des activités de programmation débranchée et de programmation visuelle (avec une interface telle que Scratch) sur la pensée informatique. À la différence de l'étude de Wohl et collaborateurs (2015) qui comprenait l'intervention d'une ou d'un enseignant qui devait introduire les concepts en question, nous cherchons plutôt à éviter les différences pouvant être dues à l'effet enseignant, afin d'être en mesure de cibler l'effet des activités débranchées elles-mêmes.

Les questions du test CTT et sa variante pour enfants tCTT visent la compréhension des processus de séquence d'instructions de déplacement de manière absolue, d'itération d'instructions et d'ordre logique d'instructions. Nous présentons dans le tableau ci-dessous les concepts travaillés dans chacune des questions du tCTT.

	Q1	Q2	Q3	Q4	Q5	Q6
Identification d'une séquence d'instructions de déplacement absolu	x		x			
Identification de l'instruction manquante dans une séquence d'instructions de déplacement absolu		x				
Itération d'instructions				x		x
Identification du nombre d'itérations dans une séquence d'instructions de déplacement absolu						x
Ordre logique entre instructions de type d'instructions (déplacement et action).					x	

Les questions 1 à 5 présentent des choix de réponse prédéfinis qu'il faut sélectionner. La question 6 invite l'apprenant à donner une réponse ouverte sur laquelle est attendu un nombre.

4 Méthodologie

Pour analyser l'effet des activités débranchées et de l'apprentissage de Scratch sur le développement de la pensée informatique, nous développons une approche quasi-expérimentale. Le choix de Scratch s'impose pour trois raisons. C'est un outil conçu pour permettre l'apprentissage massif de la programmation de la manière la plus aisée possible. C'est aussi un objet éducatif bien étudié en science de l'éducation donc qui nous permet d'appuyer notre démarche sur des éléments préétablis. C'est aussi devenu un standard utilisé par les enseignants qui participent à cette étude en leur permettant de mutualiser leur investissement avec le travail scolaire usuel. Après le pré-test, le groupe réalisé une activité débranchée consistant en un déplacement en forme de carré de deux pas (chaussure) de côté en ayant les yeux bandés afin de mettre l'accent sur le concept d'exécution de commande. Au cours de cette activité, les enfants sont invités à sortir de la classe et se mettre en file droite. Les enfants suivent les instructions de déplacement proposées par le facilitateur et ensuite écrivent sur papier le programme en langage naturel. L'activité de programmation débranchée est observée de manière à prendre des notes sur des éventuelles difficultés. Le programme écrit en langage naturel est conservé pour son analyse.

4.1 Démarche d'élaboration du protocole sous une approche d'ingénierie coopérative

Sous une approche de recherche collaborative (Desgagné, Bednarz, Lebuis, Poirier, & Couture, 2001) et d'ingénierie coopérative (Joffredo-Le Brun, Morellato, Sensevy, & Quilio, 2017; Sensevy, Forest, Quilio, & Morales, 2013), nous avons développé le protocole de recherche-action à partir de l'expérience de transposition didactique des activités débranchées (Duflot et al., 2015) et d'apprentissage de la programmation (Canellas, De La Higuera, Peinchaud, & Roche, 2016) développées au cours de différentes activités de médiation scientifique (Duflot et al., 2015) et de formation (Romero, Lille, & Patino, 2017; Romero & Vallerand, 2016). Au cours de cette démarche, nous avons pu collaborer avec des enseignants et des conseillers pédagogiques dans l'amélioration des activités débranchées et branchées. Nous présentons l'élaboration du protocole à partir de sa préparation par la mise à l'essai exploratoire (phase 1), la mise à l'essai de la première version du protocole (phase 2) et le déroulement du protocole (phase 3). Nous présentons ces trois phases ci-dessous.

- *Phase 1 : Préparation du protocole à partir de mises à l'essai exploratoires*. En préalable à cette étude, les observations préliminaires ont permis d'observer les enfants en contexte d'apprentissage de la programmation par des activités débranchées et des activités branchées (environ 2 x 30 enfants dont certains en classe double permettant d'observer un peu la progression sur un an, et plusieurs dizaines

d'enfants sur des temps courts lors d'ateliers (Fête de la Science), au cours d'événements scientifiques) et des enseignants en situation de formation (50 cadres de l'EN lors d'une formation à l'ESEN, 50 enseignant.e.s lors d´une formation Class´Code intensive et 400 enseignant.e.s dont des ERUN lors de formation présentielle de la formation hybride Class´Code en 2017-2018).

- *Phase 2 : mise à l'essai de la première version du protocole*. Afin de nous assurer que le protocole de recherche est bien adapté aux élèves du primaire, nous avons réalisé une première activité avec une classe. Ainsi, nous avons mis à l'essai le protocole de recherche avec des élèves de CM1 de l'École Jean-Marie-Hyvert (Nice, France). L'ensemble du protocole est adapté à l'âge des élèves, tant du point de vue de l'assignation des élèves au groupe contrôle (juste Scratch) ou débranché (activité débranchée et Scratch) que le test de pensée informatique. Cependant, des difficultés de compréhension sur l'usage de l'outil Scratch nous ont porté à améliorer les consignes de l'activité Scratch. Ces améliorations font suite à certaines incompréhensions par rapport à l'usage de l'outil Scratch. Tandis que le projet Scratch intégrait déjà les lutins (*sprites*), celui-ci ne contenait aucun bloc de programmation, ce qui faisait que la tâche était un peu trop ouverte pour les participants. Les efforts étaient donc davantage orientés vers la compréhension des affordances de l'outil plutôt que vers la compréhension du défi algorithmique. Afin de recentrer les efforts des élèves sur la tâche, nous avons structuré davantage l'activité en faisant une sélection de blocs de code permettant d'accomplir la tâche. Cette sélection a été faite dans le but de permettre aux participants de compléter le défi de différentes façons et ainsi d'offrir une marge de créativité aux participants.
- *Phase 3 : Déroulement du protocole***.** Suite à la mise à l'essai du protocole et les ajustements au niveau de la structuration de l'activité de Scratch nous mettons à l'essai le protocole détaillé auprès d'un groupe de CM2. Les modifications apportées suite à la première mise à l'essai avec les CM1 de la phase 2 ont permis de recentrer la tâche sur la compréhension de la séquence algorithmique. Nous présentons le déroulement détaillé au cours de la prochaine section.

4.2 Déroulement du protocole auprès d'une classe de CM2

Participants. Le protocole a été déployé avec des élèves de CM2 de l'École Jean-Marie-Hyvert. Lors de l'accueil des enfants (n= 14 filles et n= 9 garçons). Pour distribuer les élèves de manière homogène, nous les invitons à nous indiquer leur mois de naissance afin de répartir de manière homogène les élèves selon leur naissance. Deux groupes sont ainsi formés et pris en charge pour la réalisation des activités selon le protocole quasi-expérimental branché-débranché. Toutes les activités, tant débranchées comme branchées, sont réalisées par l'ensemble des enfants de manière individuelle.

Tableau 1. Protocole quasi-expérimentale branché-débranché.

	Pré-test	**Activité #1**	**Activité #2**	**Activité #3**
Groupe Débranché (n= 7 filles et n= 3 garçons).	Test de pensée informatique	Activité débranchée	Activité Scratch	Post-test (tCTT)
Groupe Contrôle (n= 7 filles et n= 6 garçons).	Test de pensée informatique	Activité Scratch	Post-test (tCTT)	Activité débranchée

Les élèves du groupe contrôle réalisent directement l'activité de programmation sur Scratch de manière individuelle. Étant donné que c'est leur première utilisation de Scratch, une introduction sommaire à l'interface est réalisée par l'expérimentateur pour présenter : l'écran de visualisation, le drapeau vert d'activation, la fenêtre de code et la bibliothèque (présentation sommaire de la catégorie événement et mouvement). Ensuite les élèves sont invités à remixer le fichier Scratch https://scratch.mit.edu/projects/227909326/. L'objectif de l'activité est de faire déplacer le chat Scratch jusqu'à #VibotLeRobot (Romero & Loufane, 2016).

Les élèves du groupe débranché sont invités à sortir dans le couloir et se placer alignés sur le mur afin de suivre des instructions. Le facilitateur de l'activité invite les élèves à faire un déplacement en forme de carré à partir d'instructions simples : avancer de deux pas et tourner à droite d'un quart de tour. A la fin de cette activité, ils retournent à la salle informatique où ils réalisent la même activité sur Scratch que le groupe contrôle.

A la fin des activités débranchées et branchées, l'ensemble des élèves réalisent à nouveau le test de pensée informatique en post-test. Par souci d'équité, les élèves du groupe contrôle réalisent l'activité débranchée après le post-test. Tous les élèves sont invités à partager leurs impressions et faire des commentaires libres à la suite des activités.

5 Résultats

Dans cette section, nous présentons la comparaison des résultats au tCTT entre les groupes contrôle et débranché et ensuite l'analyse des erreurs aux différentes questions.

5.1 Comparaison des résultats au tCTT

Les différences entre le groupe control (Scratch, n=13) et le groupe débranché (activité débranché et Scratch, n=10) sont réalisées à partir de l'analyse des résultats aux réponses du test de pensée informatique (CTT), que nous avons simplifié pour un usage avec les élèves du primaire (tCTT). Sur un ensemble de 23 élèves, les différences entre le groupe débranché développant des activités débranchées (n=10) et branchés (n=13) ne permettent pas de valider statistiquement des différences. D'autre part, nous constatons que les participants du groupe contrôle ont des résultats (m=4.08 ; sd=1.038) supérieurs au groupe débranché (m=3.5 ; sd=1.179) dès le pre-test. Bien que les deux groupes s'améliorent dans le post-test, le groupe contrôle réalise aussi un meilleur post-test (m=4 ; sd=1.155) que le groupe débranché (m=4.62 ; sd=0.65). La comparaison de moyennes des résultats du tCTT en pre-test par le test t de student (t(21)=-1,247 ; p=0.226) ni en post-test (t(21)=-1,622 ; p=0.120) ne permet pas d'observer des différentes entre le groupe contrôle et le groupe débranché.

5.2 Analyse des erreurs au cours des réponses

Au niveau de l'analyse des réponses à chacune des questions nous avons pu observer différents types d'erreurs. Si certaines erreurs peuvent s'expliquer par l'incompréhension de la notion d'itération (question 6), d'autres questions posent problème au niveau des consignes et des références à Pac-Man, comme jeu méconnu des élèves contemporains. Ainsi, les questions de déplacement faisant appel à la métaphore Pac-Man (questions 1 ,2, 4 et 6 du CTT et tCTT) présentent des erreurs sur le nombre de cases pour atteindre effectivement le fantôme. Les élèves ne comptent pas la case correspondant au fantôme (2 erreurs q1 et 5 erreurs en q2 en pré-test ; 0 erreurs en q1 et 0 erreurs en q2 au post-test). La quatrième question mobilise une compréhension des itérations, amène l'élève à décider le nombre d'itérations de la commande avancer nécessaire. Les élèves ont tendance à compter la case sur laquelle Pac-Man se situe dans le calcul du nombre d'itérations nécessaires de la commande "avancer d'une case" afin d'atteindre le fantôme. Les élèves comptant la case de départ (n=5) calculaient ainsi un nombre supérieur à celui de la bonne réponse (n=18). La sixième question mobilise une compréhension similaire des itérations que celle mobilisé dans la quatrième question. Elle amène l'élève à décider le nombre d'itération de la commande avancer nécessaire. Au lieu de calculer du nombre d'itérations, certains élèves ont tendance à compter le nombre de cases totales entre Pac-Man et le fantôme (réponses d'ordre 8 (n=2) s'ils ne comptent pas la case du fantôme ou 9 (n=2) s'ils comptent la case du fantôme). Il est également important de mentionner que la cinquième question du tCCT test ne fut pas considérée lors de l'analyse. Les participants ont eu de la difficulté à interpréter cette cinquième question causant ainsi des erreurs davantage liées à la compréhension de la question que sur une mobilisation de la pensée informatique.

5 Discussion

L'étude nous a permis d'avancer dans l'opérationnalisation d'un dispositif visant la comparaison d'une démarche intégrant une activité d'informatique débranchée à une démarche introduisant l'apprentissage de la programmation visuelle avec Scratch directement. Les trois phases itératives d'amélioration du protocole ont permis d'ajuster tant les activités branchées (Scratch) que la distribution des groupes dans l'échantillon et l'adaptation du test CTT aux enfants du primaire. Après le déroulement d'une quasi expérimentation avec 23

élèves en CM2 nous ne pouvons pas établir des différences statistiquement significatives, mais nous sommes en disposition de déployer le protocole auprès d'un plus large échantillon qui nous permettra de répondre à la question de recherche visant à identifier un possible effet de la réalisation d'une activité de programmation débranchée avec la réalisation d'une première activité de programmation visuelle avec Scratch.
L'analyse des erreurs commises par les élèves dans le tCTT nous ont également amené à porter un regard critique sur l'emploi de la métaphore Pac-Mac dans le CTT et tCTT étant donné que nous ne pouvons assumer que tous les répondants connaissent les mécaniques de déplacement et de capture de fantômes inhérentes à ce jeu vidéo. La prochaine itération du test contiendra ainsi des avatars différents et les mécaniques de déplacement seront brièvement expliqués aux participants afin d'éviter des incompréhensions.
Malgré ces limites, nous disposons d'observations qualitatives qui permettent d'éclairer la suite des travaux. Si les activités débranchées ne requièrent pas d'outil informatique, nos observations qualitatives lors des dizaines d'activités expérimentées dans le cadre de Class´Code montrent que leur intérêt va bien au-delà de pallier le manque de matériel. Lors de cette expérimentation, malgré une majeure structuration de l'activité de programmation entre la mise à l'essai initiale du protocole et le déroulement de la deuxième mise à l'essai, certaines incompréhensions demeurèrent. Au moment de réaliser l'activité de déplacement sur Scratch, les élèves se demandaient s'ils pouvaient ajouter des blocs de même nature que ceux déjà sélectionnées par l'équipe de recherche. Quelques incompréhensions ont également émergé quant à la façon d'assembler les blocs afin de créer les séquences de code. Après cette deuxième itération, nous avons ajouté des commentaires précisant que les élèves peuvent utiliser les mêmes blocs plus d'une fois. Nous avons également ajouté un exemple de fonction permettant de faciliter la compréhension de création de celles-ci. Par ailleurs, afin de conserver les artéfacts produits avec l'outil Scratch, un compte Éducateur fut créé suite à la deuxième itération.

6 Conclusion

L'apprentissage de l'informatique est une nécessité à l'ère du numérique pour aller au-delà d'un rôle de consommateur numérique et permettre de développer une citoyenneté critique et créative (Romero et al., 2017). Elle peut se faire de manière très procédurale ou de manière plus engageante et créative pour les apprenantes (Romero, 2016). Notre expérience est que les enseignants ou les médiateurs scientifiques ont aujourd'hui une pratique éclairée, leurs actions de terrains sont évaluées auprès des publics (satisfaction et mesures de ce qui a pu être partagé) et où les acteurs de cette médiation scientifique réfléchissent collégialement à leurs pratiques et aux marges d'amélioration à y apporter.
Mais il serait intéressant de considérer plus systématiquement la mise en place d'études en sciences de l'éducation sur le développement de la pensée informatique par le biais de différents types d'activités branchées et débranchées. Notre proposition ici est de montrer comment une telle étude peut être menée très simplement de façon à permettre à d'autres collègues de créer des liens entre recherche et éducation, et ainsi d'offrir les meilleures opportunités d'apprentissage aux enfants.

Références

Baron, G.-L., & Bruillard, É. (2013). École et dispositifs technologiques: points de vue croisés. *Carrefours de l'éducation*, (2), 117–129.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, *13*(1), 20–29.

Bell, T., Witten, I. H., & Fellows, M. (1998). *Computer Science Unplugged: Off-line activities and games for all ages*. Computer Science Unplugged.

Calmet, C., Hirtzig, M., & Wilgenbus, D. (2016). *1,2,3 Codez*. Editions le Pommier.

Canellas, C., De La Higuera, C., Peinchaud, É., & Roche, M. (2016). Class' Code a un an... et c'est un commencement. *1024 – Bulletin de la société informatique de France*, (9), 19-24.

Curzon, P., Dorling, M., Ng, T., Selby, C., & Woollard, J. (2014). Developing computational thinking in the classroom: a framework.

Desgagné, S., Bednarz, N., Lebuis, P., Poirier, L., & Couture, C. (2001). L'approche collaborative de recherche en éducation: un rapport nouveau à établir entre recherche et formation. *Revue des sciences de l'éducation*, *27*(1), 33–64.

Duflot, M. (2016). Jouer à «robot-idiot» pour s'initier aux algorithmes. Pixees. Consulté à l'adresse https://pixees.fr/dis-maman-ou-papa-cest-quoi-un-algorithme-dans-ce-monde-numerique-%E2%80%A8/

Duflot, M., Quinson, M., Masseglia, F., Roy, D., Vaubourg, J., & Viéville, T. (2015). When sharing computer science with everyone also helps avoiding digital prejudices. In *Scratch2015AMS*. Amsterdam, Netherlands. Consulté à l'adresse https://hal.inria.fr/hal-01154767

Faber, H. H., Wierdsma, M. D., Doornbos, R. P., van der Ven, J. S., & de Vette, K. (2017). Teaching Computational Thinking to Primary School Students via Unplugged Programming Lessons. *Journal of the European Teacher Education Network*, *12*, 13–24.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher*, *42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Hannan, L., Chatterjee, H., & Duhs, R. (2013). Object Based Learning: A powerful pedagogy for higher education. *Museums and Higher Education Working Together: Challenges and Opportunities. ed./Anne Boddington*, 159–168.

Heintz, F., Mannila, L., & Färnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education. In *Frontiers in education conference (FIE), 2016 IEEE* (p. 1–9). IEEE.

Joffredo-Le Brun, S., Morellato, M., Sensevy, G., & Quilio, S. (2017). Cooperative engineering as a joint action. *European Educational Research Journal*, 147490411769000. https://doi.org/10.1177/1474904117690006

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61.

Moreno-León, J., & Robles, G. (2015). Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (p. 132–133). ACM.

Owen, K. B., Parker, P. D., Van Zanden, B., MacMillan, F., Astell-Burt, T., & Lonsdale, C. (2016). Physical activity and school engagement in youth: a systematic review and meta-analysis. *Educational Psychologist*, *51*(2), 129–145.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … others. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60–67.

Resnick, M., & Siegel, D. (2015, novembre 10). A Different Approach to Coding. Consulté à l'adresse https://medium.com/bright/a-different-approach-to-coding-d679b06d83a#.b9jcw2js5

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, *72*, 678–691.

Romero, M, Noirpoudre, S., & Viéville, T. (2018). Apprentissage du code ? Que disent les sciences de l'éducation. *Pixees*. Consulté à l'adresse https://pixees.fr/apprentissage-du-code-que-disent-les-sciences-de-leducation/

Romero, M. (2016). De l'apprentissage procédural de la programmation à l'intégration interdisciplinaire de la programmation créative. *Formation et profession*, *24*(1), 87-89. https://doi.org/10.18162/fp.2016.a92

Romero, M. Lille, B., & Patino, A. (Éd.). (2017). *Usages créatifs du numérique pour l'apprentissage au XXIe siècle* (Vol. 1). Québec: Presses de l'Université du Québec.

Romero, M.., & Loufane. (2016). *ViBot, le robot*. Québec, QC: Publications du Québec.

Romero, M., & Netto, S. (2018). *Séminaire de recherche organisé par l'équipe pédagogique du master IME et le laboratoire TECHNÉ de l'université de Poitiers « Apprendre à programmer à l'école : représentations sociales et pratiques des enseignants »* [Vidéo]. Consulté à l'adresse https://www.youtube.com/watch?v=dTEKcW2bA7M

Romero, M., & Vallerand, V. (2016). *Guide d'activités technocréatives pour les enfants du 21e siècle* (Vol. 1). Québec, QC: Livres en ligne du CRIRES. Consulté à l'adresse http://lel.crires.ulaval.ca/public/guidev1._guide_dactivites_technocreatives-romero-vallerand-2016.pdf

Sander, E. (2000). *L'analogie, du naïf au créatif: analogie et catégorisation*. Editions L'Harmattan.

Sensevy, G., Forest, D., Quilio, S., & Morales, G. (2013). Cooperative engineering as a specific design-based research. *ZDM*, *45*(7), 1031–1043.

Shelton, C. (2016). Time to plug back in? The role of "unplugged" computing in primary schools. *ITTE Newsletter*, (2016).

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction*, *4*(4), 295–312.

Torp, L. (2002). *Problems as possibilities: Problem-based learning for K-16 education*. ASCD.

Tulving, E. (1972). Episodic and semantic memory. *Organization of memory*, *1*, 381–403.

Viéville, T., & Tort, F. (2013). Donner du sens aux éléments de technologie : l'exemple des URI. In B. Drot-Delange, G.-L. Baron, & E. Bruillard (Éd.), *Didapro5 - DidaSTIC : Didactique de l'informatique et des STIC en milieu éducatif*. Clermont-Ferrand, France: Université Blaise Pascal , Université Paris V, ENS Cachan/IFé. Consulté à l'adresse https://hal.inria.fr/hal-00843963

Wohl, B., Porter, B., & Clinch, S. (2015). Teaching Computer Science to 5-7 year-olds: An initial study with Scratch, Cubelets and unplugged computing. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (p. 55–60). ACM.