Reconnaissance de Motifs et Répétitions : Introduction à la Pensée Informatique

Yvan Peter, Marielle Léonard, Yann Secq

Reconnaissance de Motifs et Répétitions : Introduction à la Pensée Informatique

Yvan Peter[1], Marielle Léonard[2], Yann Secq[1]

[1] Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France
{prenom}.{nom}@univ-lille.fr
[2] TECHNÉ - France-IOI
marielle.leonard59@gmail.com

Résumé Cet article présente une expérimentation d'initiation d'élèves de primaire à la pensée informatique. L'objectif est de développer la capacité de reconnaissance de motifs et leur expression sous la forme de répétitions via des activités déconnectées et en ligne. Une analyse des traces et des questionnaires pré et post activités ont été réalisés afin de mesurer l'évolution des compétences. Cet article porte sur la session 2018 impliquant 20 classes de CM1/CM2. Les résultats montrent un impact positif des activités et mettent en relief les compétences acquises.

Mots clés : Pensée Informatique · Programmation · Séquence pédagogique · École primaire

Abstract. This article presents an experimentation to introduce primary school pupils to Computational Thinking. The aim is to enhance their capability to spot repetitive patterns and to express them as loops. Unplugged and plugged-in activities are used to train the pupils. Trace analysis and pre and post questionnaires were used to measure the impact of the intervention. This article deals with the 2018 session involving 20 classes. The results show a positive impact of the activities and give information about the competencies acquired.

Keywords: Computational Thinking · Programming · Pedagogical Sequence · Primary school

1 Introduction

La pensée informatique [3] (PI) correspond à des compétences d'abstraction et de résolution de problèmes qui peuvent être exploitées dans de nombreux

3. Traduction imparfaite du terme *Computational Thinking* dont le périmètre et la nature sont encore sujet à discussion.

contextes et appliquées à des domaines divers, aussi bien à l'école qu'à l'extérieur. La PI en tant que compétence cognitive ne se réduit pas à la programmation. Toutefois, elle vise à exprimer les choses d'une manière qui se prête à un traitement informatique. En ce sens, elle permet d'aborder la dimension algorithmique et la programmation en général. Dans son article fondateur, J. Wing appelle à l'enseignement de la pensée informatique pour tous et assez tôt dans la scolarité [12]. Depuis, celle-ci a été appliquée à de nombreux domaines tels que les mathématiques et les sciences expérimentales [11], les arts [5] et même l'apprentissage des langues [8].

Cet article présente notre travail sur l'acquisition de compétences de base de la PI avec des élèves d'école primaire (8 - 10 ans). Nous nous sommes focalisés sur la capacité des élèves à reconnaître des motifs répétitifs et à les exprimer sous forme de boucles. Cette abstraction et cette expression ouvrent la voie à un traitement systématique des données. Nous présentons dans cet article les résultats de l'expérimentation de 2018. Celle-ci a impliqué 20 classes venant de 16 écoles pour un total de 447 élèves. Nous avons fait passer aux élèves des questionnaires avant et après l'activité pour mesurer leur capacité à identifier et coder les répétitions. Nous avons également collecté les traces d'activité de programmation sur la plate-forme.

Dans la suite, nous passons en revue l'état de l'art concernant la pensée informatique, comment elle est introduite à l'école et comment est réalisée l'évaluation des compétences. La section 3 présente la séquence pédagogique que nous avons définie et le contexte expérimental. Nous discuterons de l'analyse des résultats dans la section 4 avant de terminer sur la conclusion et les perspectives.

2 État de l'art

2.1 La pensée informatique

Les premiers concepts de PI datent des travaux de Seymour Papert avec le langage Logo [6]. Plus récemment, l'article de Jeannette Wing présentant la PI comme une compétence de base au même titre que la lecture, l'écriture et l'arithmétique a déclenché un vif intérêt au sein des communautés éducatives et de recherche [12]. Wing insiste sur le fait que la PI ne se réduit pas à la programmation mais correspond plutôt à la capacité de manipuler des abstractions et de résoudre les problèmes qui peut être appliquée à de nombreux domaines. Elle prône l'apprentissage de ces compétences dès l'école.

Depuis, un nombre croissant de travaux de recherche se sont développés pour explorer les moyens d'introduire la PI à l'école : quels sont les concepts fondamentaux à enseigner ? Quelles technologies se prêtent le mieux à ces apprentissages ? etc. Ces problématiques sont d'autant plus importantes que de nombreux pays ont commencé à introduire la PI et plus généralement «le code» dans les programmes à différents niveaux scolaires.

Rich *et al.* ont définit des *Trajectoires d'Apprentissage* (TA) en se basant sur une étude de la littérature. Une TA définit les concepts qui peuvent être abordés en fonction de la classe et avec quel niveau de détails [7]. Elle est formalisée

par un ensemble d'*objectifs d'apprentissage* associés à un *chemin d'apprentissage* permettant la réalisation de ces objectifs ainsi que des exemples d'activités qui peuvent être réalisées. Les auteurs relèvent que la plupart des travaux sont relatifs à un seul objectif d'apprentissage ou que, quand il y en a plusieurs, ils sont indépendants. Ils observent également que ces objectifs ont pu être appliqués à différents niveaux dans la mesure où ils s'adressent en général à des novices. Pour cette raison, ils se sont inspirés des approches pédagogiques et des programmes de mathématiques pour définir l'ordre d'introduction des concepts de la PI (chemin d'apprentissage). L'article illustre leur approche sur les concepts de Séquence, Répétition et Conditionnelle.

Ching *et al.* ont réalisé une étude des technologies disponibles pour l'enseignement de la PI [2]. Ils ont identifié *les robots pédagogiques*, *les kits robotiques*, *les jeux de plateau*, *les applications en réalité augmentée*, *les applications ou sites Web pour la programmation* (graphique ou textuelle), *les outils de développement orientés animation ou jeu*. Ces différentes catégories se distinguent selon qu'elles utilisent une manipulation physique ou une interaction sur l'écran et selon que la rétroaction est concrète (e.g. mouvement d'un robot) ou abstraite (i.e., retour sur l'écran). L'étude montre que les concepts abordés avec ces technologies vont des séquences et boucles à des concepts plus avancés et peuvent impliquer des activités créatives ou de résolution de problème.

Gouws *et al.* proposent un cadre de description des compétences liées à la PI [3]. En se basant sur une revue de la littérature, ils définissent un ensemble de compétences qui peuvent être abordées à travers la programmation : *Processus et Transformations*, *Modèles et Abstractions*, *Motifs et Algorithmes*, *Outils et Ressources*, *Inférence et Logique*, *Évaluations et Améliorations*. Ils combinent ces compétences avec un niveau de maîtrise inspiré de la taxonomie de Bloom : *Reconnaître*, *Comprendre*, *Appliquer et Assimiler*. Ce cadre peut être utilisé pour l'analyse et pour la conception d'activités.

Weintrop *et al.* s'intéressent à l'introduction des pratiques de la PI en mathématiques et en sciences de manière à permettre une définition des activités de PI indépendante de l'informatique [11]. Les auteurs définissent une taxonomie de 22 pratiques groupées selon les catégories suivantes : *Données et Informations*, *Modélisation et Simulation*, *Calcul*, *Résolution de problèmes* et *Approche Systémique*.

2.2 Évaluation des compétences de la pensée informatique

Brennan & Resnick articulent la PI autour de trois dimensions : Les *concepts algorithmiques* (boucles, parallélisme...), les *pratiques* (développement itératif, débugage...) et les *perspectives* (expression personnelle, connexion avec les autres...) [1]. Ils proposent d'évaluer ces dimensions à travers l'analyse des productions, des entretiens basés sur les productions et des projets de développement.

Le rapport du *Standford Research Institute* (SRI) produit par Snow *et al.* s'intéresse aux moyens d'évaluer les compétences de la PI (résolution de problème, abstraction...) dans le contexte de l'enseignement *Exploring Computer*

Science (ECS) qui se déroule sur un an au lycée aux États-Unis [10]. A cette fin, ils proposent des patrons de conception pour la création d'évaluations pertinentes des connaissances et des pratiques. Le rapport couvre l'évaluation des unités suivantes du cours : Interaction Homme-Machine, Résolution de problème, Conception Web et Introduction à la programmation. Ces évaluations incluent des quiz, des problèmes, de la lecture de code et le traçage de son exécution.

Grover *et al.* combinent évaluation formative et sommative dans le cadre d'un module d'introduction des concepts de la PI d'une durée de 6 semaines dans le secondaire [4]. Les évaluations reposent sur des questionnaires à choix multiples dont la plupart comprennent des extraits de code en Scratch. Certains exercices impliquent de remettre en ordre des blocs de code ou des activités de traçage et de debugage de code.

Seiter *et al.* proposent un cadre pour l'évaluation des compétences de la PI à l'école primaire baptisé *Progression of Early Computational Thinking* (PECT) [9]. Ce cadre s'appuie sur l'évaluation de programmes en Scratch (utilisation d'instructions spécifiques) dans le cadre de patrons de conceptions classiques liés à au développement d'animations ou de jeux (e.g., animation, gestion des collisions...). Ces patrons sont mis en relation avec des concepts de la PI. Le cadre proposé a été évalué sur la base des programmes disponibles sur le site Web de Scratch.

Les approches basées sur l'étude du code produit par les apprenants sont intéressantes à titre exploratoire mais chronophages. L'autre mode d'évaluation majoritaire utilise des QCM éventuellement basés sur de la lecture de code.

3 Cadre expérimental

3.1 Participants et organisation

L'expérimentation de 2018 a impliqué des élèves de CM1/CM2 de 16 écoles du secteur. Vingt classes ont participé pour un total de 447 élèves. L'âge des élèves était compris entre 8 et 10 ans avec un équilibre des genres (49% de filles). L'expérimentation s'est déroulée sur une semaine avec 5 classes par jour (hors mercredi). Les classes sont venues à l'université pour la journée (sous forme de sortie scolaire). Afin de prendre en charge un aussi grand nombre d'élèves, ceux-ci sont encadrés par des étudiants de deuxième année de DUT Informatique, eux-mêmes encadrés par les enseignants. La séquence pédagogique et les activités sont présentées au préalable aux étudiants de façon à ce qu'ils puissent encadrer et accompagner les élèves dans leurs apprentissages.

3.2 Progression pédagogique

La progression pédagogique est présentée figure 1. Celle-ci inclut des activités débranchées et en ligne liées à l'identification de motifs répétitifs ainsi que leur expression sous forme de séquences d'instructions et de boucles. Nous avons trois phases décrites dans la suite. Les deux premières durent 1h30 chacune, la dernière dure 2 heures. Les élèves suivent les deux premières phases le matin et la troisième l'après-midi.

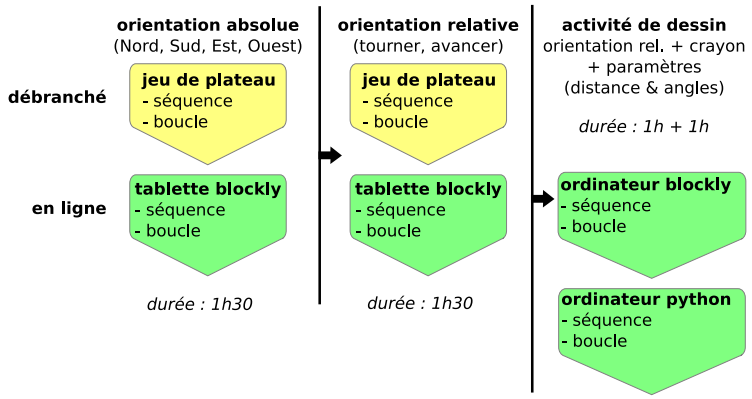


Figure 1. Progression pédagogique.

Orientation absolue Dans cette phase, les élèves doivent déplacer un personnage sur une grille jusqu'à une case définie en utilisant une orientation absolue (Nord, Sud, Est, Ouest). Ils démarrent l'activité avec un jeu de plateau (figure 2). Les élèves travaillent en groupe de 4 à 6 et prennent à tour de rôle différentes fonctions : élaborer une solution (i.e., algorithme), *pointeur d'instruction* (qui annonce l'ordre à exécuter), et *processeur* (qui exécute l'ordre en déplaçant le personnage). La complexité des chemins évolue avec l'adjonction d'obstacles et de bonus. Quand la séquence d'ordres gagne en longueur, la frustration des élèves augmente. C'est le bon moment pour introduire la notion de répétition (*répéter n fois*).

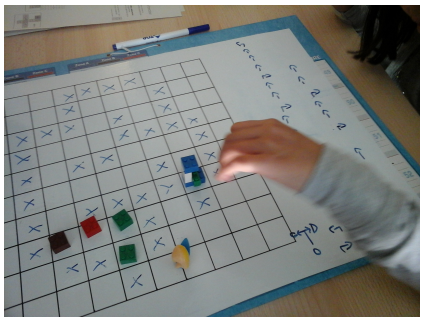


Figure 2. Jeu de plateau.

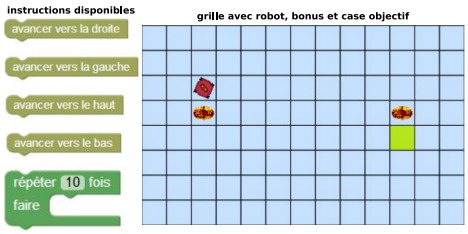


Figure 3. Puzzle sur tablette avec le jeu d'instruction

Quand les concepts principaux d'instruction, de séquence, de boucle, d'exécution (et de bug...) sont compris, les élèves passent sur tablette par binômes pour une activité similaire. Ils utilisent un langage de programmation visuelle basée sur des blocs (*Blockly*) (figure 3). Pour chaque puzzle, les élèves disposent

d'un jeu d'instructions spécifique et peuvent être limités en nombre d'instructions. Les activités évoluent à nouveau de la séquence à la boucle pour renforcer les concepts.

Orientation relative Cette phase suit une organisation similaire. Le changement principal tient à l'utilisation d'un personnage orienté qui implique un jeu d'ordres différent (*tourner à gauche*, *tourner à droite*, *avancer*). Cela implique également de se rappeler l'orientation du personnage dans la phase de planification. Le personnage tourne obligatoirement de 90°. Cette phase prépare la phase de dessin pour laquelle l'orientation est nécessaire.

Activité de dessin La dernière phase est réalisée en salle de TP avec un ordinateur par élève. Les élèves réalisent des activités de dessin avec une tortue (dans l'esprit de Logo). Le jeu d'instructions est similaire à la phase précédente avec en plus la gestion du crayon (lever et baisser pour dessiner) et la paramétrisation des instructions (e.g., `avancer(`*distance*`)` ou `tourner à droite(`*angle*`)`). Les élèves utilisent la même plate-forme que sur les tablettes. Dans la première partie, ils continuent d'utiliser *Blockly* pour dessiner. Dans la seconde partie, nous introduisons la programmation *Python*, les faisant ainsi passer d'une programmation par blocs à une programmation textuelle. Afin de faciliter les choses, les élèves utilisent des instructions en français (e.g., `avancer(10)`). Toutefois, ils utilisent la syntaxe *Python* pour les boucles.

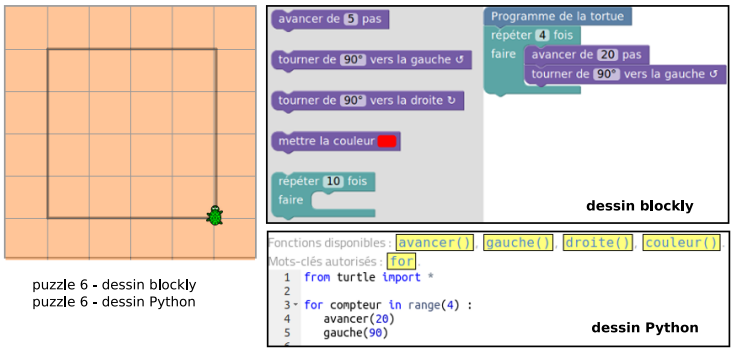
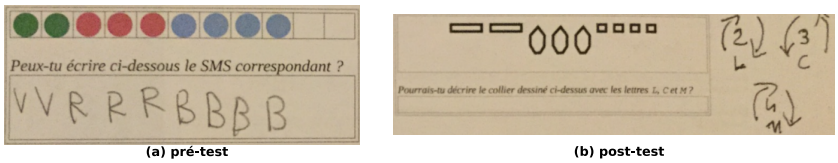


Figure 4. Passage de la programmation par blocs à textuelle.

Les langages textuels sont en général introduits beaucoup plus tard dans la scolarité. L'introduire dans la dernière séquence sur un contexte similaire et avec la même interface (figure 4) nous donne des indices sur la capacité des élèves à transférer leurs compétences de la programmation par blocs à textuelle pour cette classe d'âge.

3.3 Collecte des données

Pré et post tests Les élèvent passent un test au début et à la fin de la journée de manière à mesurer l'évolution de leur capacité à identifier des motifs répétitifs et à les exprimer sous la forme de répétitions qui peuvent être traitées par des boucles. Les tests portent sur le codage de motifs avec des lettres. Les élèves sont informés qu'ils peuvent utiliser la notation qui leur semble la plus appropriée, y compris raccourcie. Le pré test consiste à coder une représentation graphique d'une partition (figure 5(a)). Le post test porte sur le codage d'instructions pour la réalisation d'un collier de pâtes (figure 5(b)). Nous avons choisi deux contextes différents pour éviter que les élèves ne se contentent de se rappeler des motifs du pré test.



(a) pré-test (b) post-test

Figure 5. Motifs des pré et post tests.

Le tableau 1 présente les motifs utilisés, qui passent d'une séquence d'une instruction et jusqu'à trois instructions pour les plus complexes. La notation présentée ici correspond au pré test mais nous avons les mêmes motifs pour le post test. Le *motif* correspond à ce qui est présenté aux élèves et l'*encodage* à ce qui est attendu des élèves, la *correspondance* définit à quoi correspond le *type*.

Table 1. Motifs des tests

Type	Correspondance	Motif	Encodage
`1i`	motif 1 instruction	`RRRRRRRRRRR`	`11xR`
`Nx1i`	N x motifs 1 instruction	`VVRRRBBBB`	`2xV 3xR 4xB`
`2i`	motif 2 instructions	`BRBRBRBRBR`	`5x(BR)`
`3i+2i`	motifs 2 instructions + 1 instruction	`VRRVRRBBBBB`	`2x(VRR) 4B (2x(V 2R) 4B)`

Activités de programmation Les activités de programmation ont été réalisées sur la plate-forme de France-IOI[4]. Les quatre séquences comprennent un ensemble de puzzles de difficulté croissante. On retrouve successivement des séquences, des boucles à une instruction, un mélange des deux, des boucles à plusieurs instructions et jusqu'à des boucles imbriquées. La première séquence comporte 24 puzzles avec une difficulté très progressive afin que les élèves puissent

4. Association qui organise le concours Algorea.

construire leurs compétences. Les séquences suivantes comprennent de 15 à 18 puzzles. Elles démarrent avec des puzzles simples (séquence) de manière à ce que les élèves puissent transférer leurs compétences sur un nouveau jeu d'instructions puis la difficulté augmente. La figure 6 présente quelques puzzles qui sont analysés dans la suite.

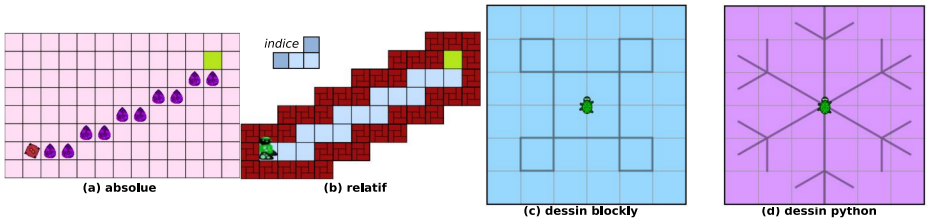


Figure 6. Puzzles difficiles dans chaque phase.

La plate-forme passe d'un puzzle au suivant après validation mais elle permet également de sélectionner un puzzle spécifique. Chaque séquence réalisée sur la plate-forme dure 30 à 45 minutes suivant les groupes. Cela explique que les élèves n'ont pas réalisé le même nombre de puzzles suivant leur rapidité. Quand de nouveaux concepts sont introduits (e.g., la boucle), le premier puzzle comprend une aide pour la réalisation.

4 Résultats

4.1 Analyses des questionnaires

Les questionnaires pré et post activité ont été codés en fonction de la reconnaissance ou non des motifs par l'élève pour arriver à une valeur entre 0 et 1. Pour les 447 élèves, nous obtenons une moyenne $m = 0,147$, $sd = 0,07$ pour le pré test et $m = 0,241$, $sd = 0,12$ pour le post test. Un t-test apparié nous donne une valeur $t(446) = -6,76$ ($p <, 0001$) qui confirme l'impact significatif de la séquence pédagogique sur la capacité des élèves à repérer et encoder des motifs répétitifs.

Table 2. Résultats détaillés.

	`1i`	`Nx1i`	`2i`	`3i+2i`
pré test	118 (27%)	102 (23%)	29 (8%)	14 (4%)
post test	166 (38%)	140 (32%)	70 (17%)	56 (14%)

Le tableau 2 présente un détail de résultats pour les différents types de motifs de complexité croissante. On note que la reconnaissance des motifs à une

instruction et des séquences de boucles à une instruction augmente de manière significative (+40% pour les deux). Le plus intéressant est probablement que la reconnaissance des motifs plus complexes (2 et 3 instructions) a progressé encore plus. Cela pourrait impliquer qu'après un apprentissage sur des motifs courts, la compétence est généralisée rapidement à des motifs plus complexes par les élèves. La figure 5 montre le cas idéal d'un élève qui n'a utilisé aucune notation de répétition sur le pré test mais l'a appliqué avec succès sur le post test.

4.2 Analyse des activités

Nous avons eu un export des traces d'activités des participants sur la plateforme. Celles-ci sont malheureusement limitées car nous n'avons que la dernière validation de l'élève pour chaque puzzle et le moment où cette validation est faite. Nous n'avons donc pas l'historique des essais des élèves. La validation du puzzle peut être positive (i.e., le puzzle est réussi) ou négative. Pour chaque séquence d'activité en ligne, nous présentons un graphe indiquant pour chaque puzzle le taux de succès (nombre validations positives / nombre de validations total) et le nombre total de validations (essais). Nous présentons également les transitions entre les niveaux de difficulté (e.g. de séquence à boucle) ce qui nous permet de visualiser les paliers de difficulté.

Pour les deux premières phases, les élèves partagent une tablette à deux et résolvent un puzzle chacun leur tour. En pratique, ils collaborent généralement pour résoudre le puzzle même s'ils n'en ont pas eu la consigne. Ceci explique le nombre d'essais maximal autour de 200. Pour la dernière phase, les élèves travaillent seuls sur ordinateur et nous avons potentiellement 447 essais (nombre d'élèves impliqués). Nous avons perdu des essais sur la première et la dernière séquence suite à des problèmes techniques, ce qui explique des chiffres inférieurs.

Orientation absolue La figure 7 montre les résultats de la première phase. Nous avons une progression mesurée de la difficulté des puzzles donnant un taux de succès supérieur à 90%. On observe une diminution du nombre d'essais quand on aborde les puzzles avec boucles qui indique que certains élèves commencent à rencontrer des difficultés. Toutefois, la chute brutale correspond aux boucles à plusieurs instructions (i.e., motifs plus longs) avec un taux de succès qui tombe à 66% pour le puzzle 19 (figure 6(a)) (le puzzle 18 est un tutoriel).

Orientation relative La figure 8 correspond à la seconde phase. Le taux de succès proche ou supérieur à 90% indique que les élèves ont réussi à transférer leur compétences sur le nouveau langage. On note également une amélioration de leur capacité à gérer les boucles à une instruction. En effet, on observe encore 137 essais avec un succès raisonnable pour le puzzle 7 (à comparer aux 51 essais du puzzle 17 de la phase précédente). A nouveau, passer aux boucles à plusieurs instructions présente une difficulté majeure. Le puzzle 8 (figure 6(b)) qui a un nombre d'essais significatif atteint un taux de succès de 56% seulement. On peut noter qu'étant le premier puzzle de ce type, il y avait une indication sur le motif à traiter.

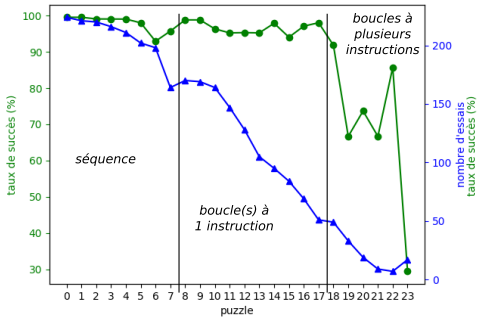


FIGURE 7. Orientation absolue.

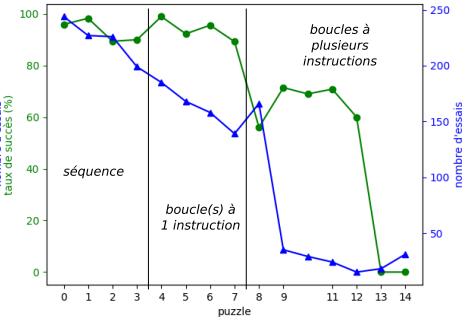


FIGURE 8. Orientation relative.

Dessin en Blockly La dernière phase introduit de nouvelles difficultés. D'abord les instructions sont paramétrées. Ensuite, on commence à avoir des angles différents de 90°que les élèves n'ont pas encore vu en classe.

A nouveau, le nombre d'essais et le taux de succès des premiers puzzles (y compris avec boucles) indiquent que le changement de langage n'est pas un problème pour les élèves une fois que les compétences liées aux séquences et boucles sont intégrées.

Les puzzles qui combinent séquences d'instructions et boucles à plusieurs instructions restent difficiles (puzzle 9 - 11, 9 fournissant des indices) dans la mesure où l'on voit le nombre d'essais chuter. La figure 6(c) montre le puzzle 11 pour lequel nous avons encore 129 essais mais avec un taux de succès de 58%. Peu d'élèves ont abordé les puzzles avec des boucles imbriquées mais avec un taux de succès légèrement supérieur à 60%.

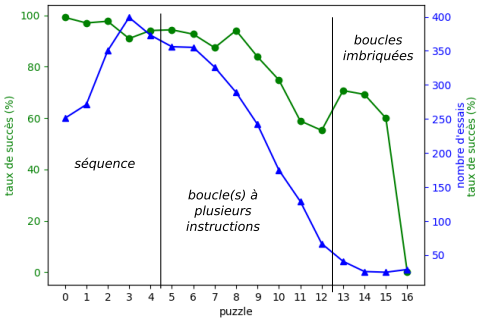


FIGURE 9. Dessin en *Blockly*.

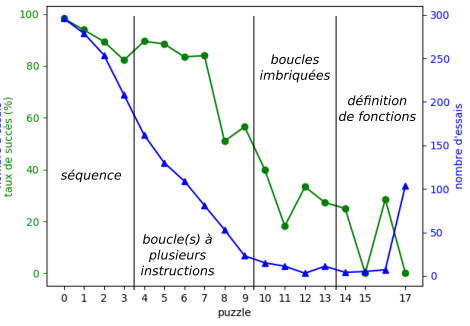


FIGURE 10. Dessin en *Python*.

Dessin en Python La figure 4 permet de voir que les élèves peuvent utiliser des boutons pour générer les instructions afin d'éviter d'avoir à tout taper au

clavier. Ils doivent malgré tout adapter les paramètres de ces instructions à leurs besoins. La première séquence de puzzles a toujours un bon taux de succès avec 82% et une bonne participation [5]. C'est un résultat intéressant qui montre que les élèves ont bien transféré leurs compétences d'un langage visuel à un langage textuel.

Les puzzles impliquant séquence et boucles sont toujours difficiles (puzzles 8 et 9) avec un taux de succès à peine supérieur à 50%. Les boucles imbriquées sont également un point difficile. Le puzzle 10 est un tutoriel. Le puzzle 11 présenté figure 6(d) atteint seulement 18% de taux de succès avec un faible nombre d'essais. Le dernier puzzle correspond à une activité où les élèves peuvent dessiner librement. Il n'y a donc pas de condition de validation. Le graphe montre qu'un bon nombre d'élèves en on profité.

5 Conclusion

Cet article porte sur l'apprentissage des concepts fondamentaux et des compétences de la pensée informatique par des élèves de 8 - 10 ans. Nous avons conçu une séquence pédagogique pour initier les élèves aux notions d'instruction, de séquence et de boucles et leur faire pratiquer ces compétences sur différents langages (libre (activité débranchée), blockly et textuel). Cette séquence a évolué depuis nos premières expérimentations en 2014 et nous avons un nombre croissant de classes impliquées. Nous cherchons également à améliorer nos instruments de mesure pour vérifier l'impact sur les élèves (questionnaires et analyse des activités).

Les résultats de l'expérimentation de 2018 sont encourageants. L'analyse statistique des questionnaires montre l'impact significatif de la séquence et nous avons une augmentation significative des élèves qui identifient les motifs répétitifs et sont capables de les synthétiser sous forme de boucles. Il semble également qu'une fois que les élèves ont acquis cette compétence pour des motifs de longueur 2 (2 instructions dans la boucle), ils sont capables de généraliser à des motifs plus longs.

Les résultats de l'analyse des activités sont à prendre avec précaution dans la mesure où, bien sûr, ils sont aidés par les étudiants voire les professeurs et parents accompagnant. Avec un étudiant pour 4 à 6 élèves, on peut toutefois espérer que cela n'a pas un impact trop fort sur les résultats.

Les élèves reçoivent un diplôme à la fin de la journée qui donne l'adresse de la plate-forme ainsi que leur code d'identification. Nous avons observé environ 300 accès dans les jours suivants et jusqu'à deux mois après (date de la récupération des données). Toutes les séquences ont été utilisées y compris le dessin en Python pour laquelle il y a eu 271 essais. Ces éléments sont encourageants et montrent la motivation des élèves.

Nous avons réalisé une nouvelle expérimentation en janvier 2019 qui doit encore être dépouillée. Nous avons bénéficié d'une nouvelle version de la plate-forme qui nous donnera des informations plus précises (début et fin des activités,

5. Nous avons perdu des traces suite à un problème technique.

essais réalisés). Cela nous permettra d'affiner notre analyse et d'aborder d'autres points comme les stratégies de résolution des élèves. Nous avons également fait évoluer nos questionnaires pour identifier un biais potentiel lié à la disposition des motifs à coder.

Remerciements

Nous remercions l'association France-IOI pour la mise à disposition de la plate-forme. Ce travail est partiellement soutenu par le projet Interreg Dig-e-lab.

Références

1. Brennan, K., Resnick, M. : New frameworks for studying and assessing the development of computational thinking. annual American Educational Research Association meeting, Vancouver, BC, Canada pp. 1–25 (2012)
2. Ching, Y.H., Hsu, Y.C., Baldwin, S. : Developing Computational Thinking with Educational Technologies for Young Learners. TechTrends (1980) (2018)
3. Gouws, L., Bradshaw, K., Wentworth, P. : Computational thinking in educational activities. In : Proceedings of the 18th ACM conference on Innovation and technology in computer science education - ITiCSE '13. p. 10 (2013)
4. Grover, S., Cooper, S., Pea, R. : Assessing computational learning in K-12. Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE '14 pp. 57–62 (2014)
5. Knochel, A.D., Patton, R.M. : If Art Education Then Critical Digital Making : Computational Thinking and Creative Code. Studies in Art Education : A Journal of Issues and Research **57**(1), 21–38 (2015)
6. Papert, S. : Mindstorms : Children, computers, and powerful ideas. Basic Books, Inc. (1980)
7. Rich, K.M., Strickland, C., Binkowski, T.A., Moran, C., Franklin, D. : K–8 Learning Trajectories Derived from Research Literature : Sequence , Repetition , Conditionals. acm Inroads **9**(1), 46–55 (2018)
8. Sabitzer, B., Jarnig, M. : Computational Thinking Through Modeling In Language Lessons. In : IEEE Global Engineering Education Conference (EDUCON). pp. 1913–1919 (2018)
9. Seiter, L., Foreman, B. : Modeling the learning progressions of computational thinking of primary grade students. Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research pp. 59–66 (2013)
10. Snow, E., Tate, C., Rutstein, D., Bienkowski, M. : Assessment Design Patterns for Computational Thinking Practices in Secondary Computer Science. Tech. Rep. December, SRI International (2017)
11. Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., Wilensky, U. : Defining Computational Thinking for Mathematics and Science Classrooms. Journal of Science Education and Technology **25**(1), 127–147 (2016)
12. Wing, J.M. : Computational Thinking. COMMUNICATIONS OF THE ACM **49**(3), 33–35 (2006)