Objet Tangible ou Simulation Numérique : Deux Situations Équivalentes pour l'Apprentissage de la Programmation ?

Grégoire Fessard, Patrick Wang, Ilaria Renna

Objet Tangible ou Simulation Numérique : Deux Situations Équivalentes pour l'Apprentissage de la Programmation ?

Grégoire Fessard[1], Patrick Wang[1] et Ilaria Renna[1]

Institut Supérieur d'Électronique de Paris, Paris, France
{gregoire.fessard, patrick.wang, ilaria.renna}@isep.fr

Résumé. La programmation physique est une approche visant à programmer et interagir avec un objet tangible programmable pour apprendre des concepts clés de la programmation. Cette approche présente de nombreux avantages en termes de motivation, de créativité, et d'apprentissage de la programmation. Cependant, ces apprentissages sont rarement comparés à ceux résultant de la programmation d'une simulation numérique de l'objet tangible utilisé. Cet article présente les résultats d'une telle étude comparative. Celle-ci cherche à déterminer si ces deux situations amènent à des différences significatives dans l'apprentissage de concepts clés de la programmation. Les participants à cette étude ont été séparés en deux groupes : un programmant une carte électronique (la BBC micro:bit), l'autre programmant une simulation de cette carte. Les résultats montrent qu'il y a un apprentissage des concepts fondamentaux de la programmation quel que soit le groupe. Cependant, aucune différence significative n'a été constatée lorsque les gains d'apprentissage ont été comparés entre les deux groupes.

Mots-clé: Apprentissage de l'informatique · Programmation physique · Programmation numérique · Programmation par blocs.

Abstract. Physical computing is about programming and interacting with a tangible object to learn key concepts of Computer Science. This approach presents several benefits, namely increased student motivation, creativity, and learning gains. Yet, these learning gains hardly are compared with those resulting from the programming of a simulation of a tangible object. This article presents the results of a comparative study we conducted to explore this issue. With this study, we aimed at determining whether the programming of a tangible object or its digital simulation yields significantly different learning gains. Participants were divided into two groups: one programmed an electronic board (the BBC micro:bit) while the other programmed a simulation of it. The results of this experiment suggest that, while both groups significantly improved their understandings of fundamental programming concepts, no significant difference was found when comparing the learning gains between the two groups.

Keywords: Computer Science Education · Physical computing · Digital computing · Block-based programming.

1 Introduction

La programmation physique (ou *physical computing* en Anglais) est un paradigme dont l'objectif est de « concevoir des objets tangibles et interactifs en utilisant des composants programmables » [14]. Cette approche se distingue d'un enseignement de la programmation utilisant historiquement la console comme seule interface lors de l'exécution d'un programme. Suite à l'émergence des robots et cartes programmables, la programmation physique est devenue de plus en plus exploitée et, en conséquence, a fait l'objet de nombreuses études portant sur l'initiation à la programmation, en particulier chez les plus jeunes [7,18]. Pour autant, et bien que focalisées sur les effets de la programmation physique chez les apprenants, ces études ne comparent pas leurs résultats avec des situations relevant de la programmation *numérique* (i.e., la programmation d'une simulation numérique d'un objet tangible).

Dans cet article, nous présentons une étude menée afin d'investiguer si une des deux situations (programmation physique ou numérique) conduit à des gains d'apprentissage en programmation plus importants que l'autre. Pour cett étude préliminaire, nous avons utilisé la carte électronique BBC micro:bit que nous avons proposée sous sa forme tangible ou simulée à des débutants en programmation afin de proposer deux expériences de programmation identiques. Un langage de programmation par blocs a été utilisé pour programmer cette carte. Pour cette étude comparative, nous nous sommes plus particulièrement intéressés à l'apprentissage de trois concepts fondamentaux de la programmation : les variables, les structures conditionnelles, et les structures itératives.

Dans la section qui suit, nous allons présenter un état de l'art portant sur l'enseignement de la programmation en nous intéressant principalement à la programmation physique et à l'utilisation de langages de programmation par blocs. En Sect. 3 et 4, nous décrivons l'environnement de programmation utilisé pour notre expérimentation ainsi que la conception du protocole expérimental. La Sect. 5 présente les résultats de notre expérimentation. Enfin, les deux dernières sections proposent une discussion de ces résultats et des pistes de travaux futurs.

2 État de l'art

L'apprentissage de la programmation présente des difficultés bien répertoriées dans la littérature. Par exemple, du Boulay identifie, entre autres, trois difficultés : la maîtrise de la syntaxe d'un langage de programmation, la compréhension de la structure d'un programme ou d'un algorithme, et la compréhension du rapport entre un programme et la machine qui peut ainsi être programmée [1].

La programmation visuelle, et en particulier la programmation par blocs, peut être utilisée pour pallier les deux premières difficultés. Ces langages de

programmation permettent de se focaliser sur la conception d'un algorithme sans se soucier de la syntaxe d'un langage de programmation. Une étude comparative a montré que la programmation par blocs entraînait une meilleure compréhension des concepts élémentaires de la programmation qu'un langage de programmation standard [21]. Cet aspect explique les nombreuses études mettant en œuvre des langages utilisant des blocs pour l'introduction à la programmation [12,21].

Le constructionnisme [13] est une théorie souvent mentionnée dans les travaux de recherche portant sur l'apprentissage de la programmation et qui peut pallier la troisième difficulté. Cette théorie stipule que l'acquisition de nouvelles connaissances peut se produire grâce à la *construction* d'un objet synthétisant des objectifs pédagogiques identifiés. Le constructionnisme a ouvert la voie à de nombreuses études mettant en œuvre des objets programmables tangibles [3,7,9] ou simulés [4,12] pour l'apprentissage de la programmation.

La programmation physique s'inspire donc du constructionnisme en proposant de programmer des robots ou des cartes électroniques. L'état de l'art sur la programmation physique met l'accent sur les bienfaits d'une telle approche : gain de motivation et de créativité [18], inclusion de populations sous-représentées [17], et apprentissage de la programmation [3]. Cependant, des résultats similaires concernant l'apprentissage de la programmation ont été obtenus lorsque des objets programmables numériques ont été utilisés [4,10,12].

Bien que ces deux situations mènent à un apprentissage de la programmation, peu de travaux ont cherché à déterminer si une situation est mieux adaptée que l'autre. Concernant l'apprentissage de la programmation, nous n'avons pu identifier que trois articles mettant en œuvre un objet tangible et sa simulation [2,5,6]. Cependant, ces études n'ont pas été conçues pour comparer les gains d'apprentissage entre ces deux situations mais plutôt pour proposer deux systèmes complémentaires. Notre étude cherche donc à combler ce vide dans l'état de l'art et à apporter une réponse à la question de recherche suivante : « y a-t-il des gains d'apprentissage différents lorsqu'un apprenant programme un objet tangible ou une simulation numérique de cet objet tangible ? » En particulier, nous nous focalisons sur l'apprentissage de trois concepts fondamentaux de la programmation : les variables, les structures conditionnelles, et les structures itératives. Nous avons donc reformulé notre hypothèse de recherche en trois sous-questions, une pour chaque concept de programmation mentionné précédemment.

3 Environnement de programmation

L'environnement de programmation mis en œuvre dans cette étude est composé de deux éléments : la BBC micro:bit en tant qu'objet programmable et une interface Web permettant de programmer une micro:bit grâce à un langage de programmation par blocs. Étant donné le contexte de notre étude, nous avons conçu deux versions distinctes de cet environnement : une première permettant de programmer une carte micro:bit tangible et une seconde permettant de programmer une simulation numérique de cette carte. Comme nous allons le décrire dans la suite de cette section, ces deux environnements ont été conçus pour pro-

poser une expérience de programmation pratiquement identique selon que l'on programme l'objet tangible ou la simulation.

3.1 Objet programmable

La micro:bit [1] est une carte programmable conçue en 2016 et ayant fait l'objet d'études dans le domaine de l'apprentissage de la programmation [16,18,19]. Cette carte est équipée, entre autres, de deux boutons-poussoirs et d'une grille de diodes électroluminescentes permettant la gestion d'entrées-sorties (voir Fig. 1). Ces LED peuvent être utilisées pour afficher aussi bien des informations alphanumériques que des formes pré-définies (voir la croche illustrée en Fig. 2), et un mécanisme de défilement permet d'afficher des messages arbitrairement longs.

La simulation numérique de la micro:bit a repris un programme *open source* écrit en Javascript [2] permettant de mimer à l'identique le comportement d'une carte micro:bit tangible. Ce programme nous a donc permis d'intégrer facilement la simulation de la micro:bit dans l'interface Web de programmation.

Sous sa version tangible, la micro:bit peut être reliée à un ordinateur par un câble USB. En transférant un fichier binaire dans son espace de stockage interne, il est possible de mettre à jour le programme exécuté par la carte [3]. La micro:bit simulée peut quant à elle être programmée en faisant appel à des fonctions Javascript implémentées dans l'interface Web de programmation.

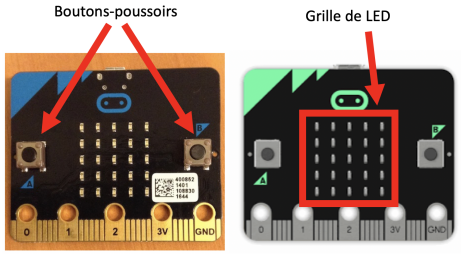

Fig. 1. Une carte micro:bit tangible (à gauche) et sa simulation numérique (à droite).

3.2 Interface de programmation

Pour programmer la micro:bit, nous avons développé une interface Web de programmation illustrée en Fig. 2. Cette interface implémente la bibliothèque Blockly [4] qui permet de construire des programmes à l'aide d'un langage de programmation par blocs.

1. https://microbit.org/
2. https://github.com/Microsoft/pxt-microbit
3. Cette étape a été automatisée pour rendre ce processus identique entre les deux objets, comme mentionné en Sect. 3.2.
4. https://developers.google.com/blockly/

La Fig. 2 illustre plus spécifiquement l'interface qui a été développée pour la programmation de la simulation de la micro:bit. Cette interface est divisée en trois zones. La Zone 1 affiche les énoncés des exercices sur lesquels devront travailler les participants à l'expérimentation. Un menu déroulant permet de passer d'un exercice à l'autre. La Zone 2 correspond à l'environnement de construction d'un programme. Celle-ci se compose d'une boîte à outils (sur fond gris) qui contient les blocs de programmation. Elle est composée de cinq catégories : Affichage, Math, Variables, Logique, et Boucle. Dans chacune de ces catégories, il est possible de trouver des blocs de programmation spécifiques. L'utilisateur peut ainsi construire des programmes en glissant et déplaçant ces blocs depuis la boîte à outils vers l'espace central (sur fond blanc) et en les emboîtant à la manière d'un puzzle. Enfin, la Zone 3 affiche la simulation de la carte micro:bit.

L'interface utilisée pour programmer l'objet tangible reprend à l'identique les Zones 1 et 2. Le contenu de la Zone 3 est simplement remplacé par un texte expliquant comment exécuter un programme sur la carte micro:bit tangible.

Quel que soit l'objet programmable utilisé, l'exécution d'un programme se fait grâce au bouton « Lancer le programme » situé en bas de la Zone 2. Cela est rendu possible grâce à Blockly qui peut traduire chaque bloc en Javascript ou Python, deux langages utilisés pour programmer respectivement la simulation et la carte tangible. L'environnement de programmation propose ainsi la même expérience utilisateur quelle que soit la modalité utilisée.

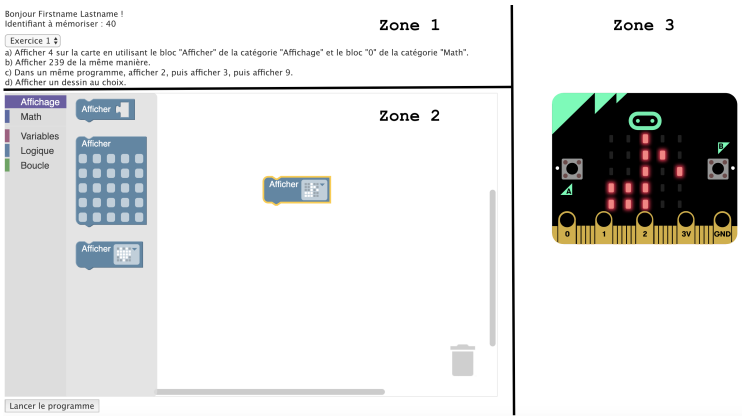

Fig. 2. Environnement de programmation par blocs présentant une simulation numérique de la micro:bit.

4 Protocole expérimental

Pour répondre aux questions de recherche formulées précédemment, nous avons conduit une expérimentation sur une demi-journée avec 36 collégiens et lycéens

âgés de 14 à 17 ans (dont 9 filles et 27 garçons, avec une moyenne d'âge de 15.2 ans). Sur ces 36 participants, 23 ont déclaré n'avoir aucune connaissance en programmation et 13 ont déclaré avoir des connaissances élémentaires de programmation. Ainsi, nous avions des participants capables de comprendre et d'apprendre des exercices proposés, ce qui nous permettait de quantifier l'utilité de l'environnement d'apprentissage conçu. Ces participants ont été répartis dans deux groupes : un groupe programmant l'objet tangible, l'autre programmant la simulation de la micro:bit. La composition des groupes a été décidée en amont de l'expérimentation et ce afin de garantir leur homogénéité en termes d'âge, de genre, et de connaissances initiales en programmation. Cette répartition a été validée par l'analyse *a posteriori* des résultats (voir Sect. 5.1). Le protocole expérimental conçu pour cette étude comporte un questionnaire pré-post test ainsi qu'une série d'exercices portant sur les concepts de variables, de structures conditionnelles, et de structures itératives et dont les énoncés sont affichés dans la Zone 1 de l'interface de programmation comme vu précédemment. L'hypothèse expérimentale que nous souhaitions tester était que la programmation d'un objet tangible ou d'une simulation équivalente de cet objet tangible conduirait à des gains d'apprentissage différents pour chacun de ces trois concepts, sans se risquer à émettre un jugement sur l'une ou l'autre de ces situations.

L'expérimentation a suivi le déroulement suivant. Dans un premier temps, l'environnement de programmation et le langage de programmation par blocs ont été présentés aux participants. Durant cette phase, aucune notion de programmation n'a été présentée en dehors de l'affichage d'informations en sortie (i.e., sur la grille de LED) puisque nécessaire pour la compréhension des questions du pré-post test. Puis, les participants ont répondu au pré-test, ont ensuite travaillé sur les exercices de programmation, et enfin ont répondu au post-test. Au cours de cette séquence, les participants devaient travailler de façon individuelle et pouvaient demander de l'aide uniquement en cas de problème technique.

4.1 Questionnaire pré-post test

Le questionnaire pré-post test est composé de 14 questions à choix multiple : 5 questions portent sur les variables, 5 sur les structures conditionnelles, et 4 sur les structures itératives. Nous nous sommes servis de la taxonomie de Bloom révisée [8] pour concevoir les énoncés des questions et pour définir un ordre de difficulté croissant pour les exercices de chaque notion.

Ces énoncés ont aussi été conçus pour que chaque question ne porte que sur un seul et unique concept de la programmation à la fois. Par exemple, cela implique que nous n'avons pas incorporé de question utilisant la notion de variable de boucle puisque celle-ci requiert des connaissances à la fois sur le concept de variable et de structure itérative. Cette précision est primordiale puisque, sans cette distinction, il aurait été beaucoup plus compliqué de quantifier les gains d'apprentissage pour chaque notion séparément.

La Fig. 3 présente trois exemples de questions utilisées dans le pré-post test. Pour chaque question, une seule réponse correcte est possible. Les réponses erronées reprennent quant à elles des erreurs classiques de programmation réper-

toriées dans la littérature [15]. Cela nous a permis de proposer des réponses plausibles qu'un débutant en programmation pourrait assimiler comme vraies.

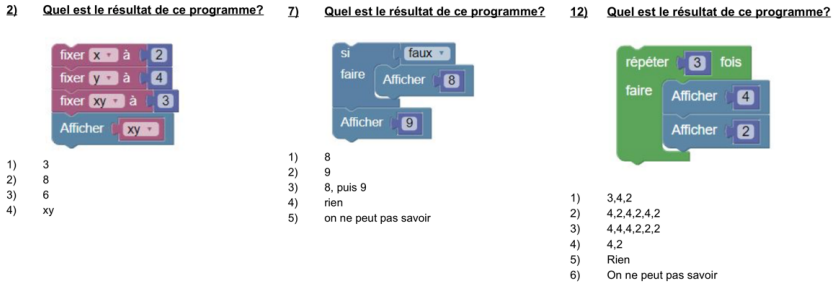

Fig. 3. Exemples de questions du pré-post test pour chaque concept étudié.

4.2 Exercices de programmation

Les participants ont travaillé sur 15 exercices de programmation portant sur les trois concepts mentionnés précédemment. Sur ces 15 exercices, 4 portent exclusivement sur le concept de variable, 3 sur le concept de structure conditionnelle, 4 sur le concept de structure itérative. Nous avons aussi proposé un exercice de prise en main de l'environnement de programmation (à réaliser avant tous les autres) et 3 exercices mêlant plusieurs concepts cités plus haut (à réaliser après tous les autres). Deux exemples d'exercices sont illustrés en Fig. 4, un portant sur les variables et l'autre sur les structures conditionnelles.

La conception de ces exercices a suivi une approche identique à celle utilisée pour la construction du questionnaire pré-post test. Nous nous sommes basés sur l'état de l'art de Qian et Lehman [15] portant sur les erreurs de programmation fréquemment commises par des apprenants pour concevoir des exercices pouvant amener nos participants à reproduire et à apprendre de leurs erreurs. Les exercices relatifs à un même concept ont aussi été proposés dans un ordre de difficulté croissant. Les premiers exercices de chaque concept avaient tous un énoncé détaillé avec une résolution guidée ; puis les exercices suivants demandaient des manipulations plus complexes tout en supprimant progressivement le guidage de l'énoncé. Par exemple, on peut voir en Fig. 4 que l'énoncé de l'exercice 6 (premier exercice sur les structures conditionnelles) est beaucoup plus détaillé que celui de l'exercice 5 (dernier exercice sur les variables).

Il était demandé aux participants de travailler les exercices dans l'ordre proposé. Cependant, aucun mécanisme de vérification n'a été implémenté et les participants étaient libres de passer à l'exercice suivant quand ils le souhaitaient. Ils pouvaient aussi revenir sur un exercice précédent et consulter leurs programmes tels qu'ils les avaient laissés dans la Zone 2 de l'interface de programmation.

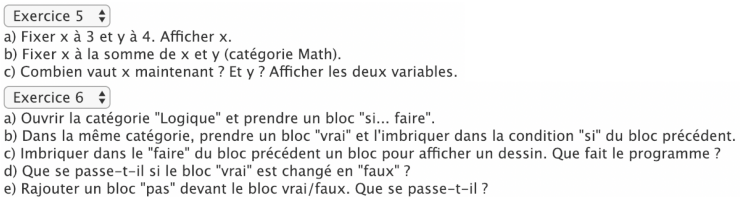

Fig. 4. Deux exemples d'exercices à réaliser.

5 Résultats

5.1 Validation du protocole expérimental

Comme précisé en Sect. 4, l'expérimentation a rassemblé 36 participants qui ont été séparés en deux groupes :

— 18 participants ont programmé la micro:bit tangible. Ce groupe était composé de 5 filles et 13 garçons avec une moyenne d'âge de 15.6 ans. Dans ce groupe, 11 participants ont déclaré n'avoir aucune connaissance en programmation et 7 ont affirmé avoir des connaissances élémentaires.

— 18 participants ont programmé la simulation de la micro:bit. Ce groupe était composé de 4 filles et 14 garçons avec une moyenne d'âge de 14.8 ans. Dans ce groupe, 12 participants ont déclaré n'avoir aucune connaissance en programmation et 6 ont affirmé avoir des connaissances élémentaires.

Nous avons d'abord cherché à confirmer l'homogénéité du niveau initial des participants entre les deux groupes. Pour cela, nous avons analysé les résultats obtenus au pré-test par les deux groupes avec un test de Student bilatéral pour deux échantillons non-appariés. Les résultats de ce test ($t = 1.499, p = 0.145 > 0.05$) indiquent que les moyennes au pré-test des deux groupes ne sont pas significativement différentes. Ce premier résultat confirme que la composition *a priori* des groupes n'a pas introduit de biais initial dans l'expérimentation.

5.2 Apprentissage de la programmation

Le Tableau 1 récapitule les résultats obtenus au pré-post test pour chaque groupe et en fonction de chaque concept évalué.

Tableau 1. Résultats au pré-post test pour chaque groupe de participants.

Groupe	Variables (5 pts)		Conditions (5 pts)		Boucles (4 pts)	
	Pré-test	Post-test	Pré-test	Post-test	Pré-test	Post-test
Tangible	$\mu = 4.167$	$\mu = 4.222$	$\mu = 1.667$	$\mu = 4.167$	$\mu = 2.389$	$\mu = 3.444$
	$\sigma^2 = 0.735$	$\sigma^2 = 0.889$	$\sigma^2 = 1.059$	$\sigma^2 = 1.441$	$\sigma^2 = 0.840$	$\sigma^2 = 0.497$
Simulation	$\mu = 3.611$	$\mu = 4.000$	$\mu = 1.389$	$\mu = 3.222$	$\mu = 2.222$	$\mu = 3.111$
	$\sigma^2 = 2.605$	$\sigma^2 = 0.941$	$\sigma^2 = 0.605$	$\sigma^2 = 1.595$	$\sigma^2 = 0.771$	$\sigma^2 = 0.928$

Apprentissage global. Nous souhaitions vérifier que le protocole expérimental conçu conduisait bien à un gain significatif d'apprentissage des concepts élémentaires de la programmation. Pour cela, nous avons calculé par deux fois la valeur de t dans le cadre d'un test de Student unilatéral pour des échantillons appariés : une fois pour le groupe programmant l'objet tangible, une seconde fois pour le groupe programmant la simulation. Concernant le groupe programmant l'objet tangible, les résultats ($t = 8.574, p \ll 0.05$) suggèrent que les notes obtenues au post-test sont significativement meilleures que celles obtenues au pré-test. Les résultats du groupe programmant la simulation ($t = 5.242, p \ll 0.05$) conduisent aux mêmes conclusions. Ce point valide la conception des exercices proposés aux participants dans le cadre d'une initiation à la programmation.

Nous avons ensuite comparé les gains d'apprentissage entre les deux groupes avec un test de Student bilatéral pour deux échantillons non-appariés. Le gain d'apprentissage d'un individu a été mesuré en soustrayant sa note du pré-test à sa note du post-test. Les résultats ($t = 0.687$ et $p = 0.497$) indiquent que les gains d'apprentissage sont équivalents entre les deux groupes.

Notion de variable. Nous avons commencé par analyser le gain d'apprentissage concernant la notion de variable pour chacun des deux groupes. Les résultats du test de Student unilatéral pour des échantillons appariés ($t = 0.325$ et $p = 0.375$ pour le groupe programmant la carte, $t = 1.046$ et $p = 0.155$ pour le groupe programmant la simulation) suggèrent qu'il n'y a pas eu d'apprentissage significatif pour aucun des deux groupes. Nous n'avons donc pas jugé utile de poursuivre l'analyse des données en comparant les gains d'apprentissage entre les deux groupes.

Notion de structure conditionnelle. Pour chaque groupe, nous avons cherché à déterminer si les notes au post-test étaient significativement meilleures que les notes au pré-test. Pour cela, nous avons conduit par deux fois un test de Student unilatéral pour des échantillons appariés. Les résultats ($t = 6.556$ et $p \ll 0.05$ pour le groupe programmant la carte, $t = 6.761$ et $p \ll 0.05$ pour le groupe programmant la simulation) suggèrent qu'il y a effectivement eu une amélioration significative des notes entre le pré-test et le post-test concernant la notion de structure conditionnelle.

Nous avons continué cette analyse en comparant les gains d'apprentissage entre les deux groupes en appliquant un test de Student pour deux échantillons non-appariés. Les résultats de ce test ($t = 1.425$ et $p = 0.163$) indiquent que les moyennes des gains d'apprentissage étaient similaires entre les deux groupes. Cela implique que, dans le cadre de notre expérimentation, la programmation d'un objet tangible ou d'une simulation n'a pas entraîné de différence significative dans l'apprentissage et la compréhension du concept de structure conditionnelle.

Notion de structure itérative. Enfin, nous avons procédé à une analyse similaire pour évaluer les apprentissages concernant la notion de structure itérative.

Les résultats des tests de Student unilatéraux pour des échantillons appariés ($t = 6.174$ et $p \ll 0.05$ pour le groupe programmant la carte, $t = 3.915$ et $p \ll 0.05$ pour le groupe programmant la simulation) suggèrent que les notes obtenues au post-test étaient significativement meilleures que celles du pré-test pour les deux groupes.

Cependant, les résultats du test de Student bilatéral pour deux échantillons non-appariés ($t = 0.586$ et $p = 0.562$) montrent encore une fois que les gains d'apprentissage ne sont pas significativement différents entre les deux groupes.

6 Discussion

Les résultats de notre étude suggèrent qu'il n'y a pas de différence significative d'apprentissage selon qu'un apprenant programme un objet tangible ou sa simulation. De plus, les résultats concernant la notion de variable n'ont pas montré de gain d'apprentissage significatif entre le pré-test et le post-test. Cela peut s'expliquer par les bons scores obtenus dès le pré-test pour les deux groupes de participants (voir Table 1). Nous y voyons deux explications possibles : les questions étaient trop simples ou alors la notion de variable est plus facile à comprendre (ce qui est aussi un résultat présenté dans d'autres études [21]).

L'expérimentation décrite dans cet article présente plusieurs points d'amélioration possibles. Premièrement, les participants n'ont pu interagir avec l'environnement de programmation qu'une demi-journée. Bien que nos résultats montrent un gain d'apprentissage significatif pour chacun des groupes, il serait intéressant d'analyser des données provenant d'une expérimentation plus longue et ancrée dans un contexte institutionnel. Un tel résultat permettrait de trancher quant à l'intérêt d'introduire des objets tangibles programmables dans les cours d'introduction à la programmation. Une telle expérimentation est d'ores et déjà prévue dans la suite de nos travaux.

Un second point d'amélioration porte sur la manipulation de l'objet programmable par les apprenants. En effet, les participants n'étaient pas incités à interagir physiquement avec la micro:bit tangible en raison de la conception des exercices de programmation : le maniement de la carte n'avait pas de répercussion sur le programme exécuté. Or, la micro:bit présente plusieurs dispositifs permettant de récupérer des informations en entrée tels que les deux boutons-poussoirs mentionnés en Sect. 3.1 ou encore un accéléromètre (qui sont d'ailleurs aussi disponibles sur la simulation). Puisque notre protocole expérimental n'incluait pas la notion de données en entrée/sortie (hormis l'affichage d'informations sur les LED), nous n'avons pas conçu d'exercice utilisant ces composants. Il serait donc pertinent de comparer nos résultats avec ceux d'une étude mettant plus l'accent sur un interaction conséquente avec l'objet programmable tangible.

nfin, nous avons fait le choix d'utiliser la BBC micro:bit pour cette expérimentation. Cette carte présente les avantages d'être peu encombrante, peu coûteuse, et disponible sous une forme numérique. Un point à explorer concerne donc l'utilisation d'un robot et de sa simulation afin de mener une étude comparative semblable à celle présentée dans cet article.

7 Conclusion et travaux futurs

Cet article présente une étude comparative des gains d'apprentissage lorsqu'un apprenant programme un objet tangible ou sa simulation. Les résultats indiquent que, bien qu'un apprentissage significatif soit observé dans les deux cas, ces situations semblent globalement équivalentes concernant l'apprentissage de la programmation, et plus spécifiquement des notions de structures conditionnelles et itératives. Ces résultats nous conduisent à investiguer plus en détail les différences entre programmation physique et numérique en concevant une nouvelle expérimentation insistant sur les manipulations de l'objet programmable.

En plus des pistes d'améliorations décrites plus haut, nous avons identifié trois axes de travaux futurs. Le premier concerne l'analyse des traces d'utilisation de l'environnement de programmation afin de déterminer les stratégies mises en place par les apprenants pour résoudre les exercices proposés. Cela nous permettrait aussi d'étudier l'évolution de la compréhension des concepts étudiés à travers les actions réalisées par chaque apprenant tout au long des exercices. Le second concerne l'identification des différences d'apprentissage lorsque la programmation physique est utilisée dans l'apprentissage des STEM (*Science, Technology, Engineering, Mathematics*). Des articles présentent déjà ce type de pratique [11,20] et il s'agit de conduire une étude comparative reprenant les caractéristiques de celle décrite dans cet article (i.e., homogénéité des connaissances initiales, moyennes d'âge similaires, répartition équilibrée entre filles et garçons). Enfin, le dernier axe de recherche porte sur l'étude comportementale des apprenants lorsqu'ils font face à un environnement de programmation impliquant un objet tangible ou une simulation. Ce dernier point permettrait de fournir des compléments de réponses quant à l'ampleur du gain de motivation ou de créativité résultant de la programmation physique et/ou numérique.

Références

1. du Boulay, B. : Some Difficulties of Learning to Program. Journal of Educational Computing Research **2**(1), 57–73 (1986)
2. Brauner, P., Leonhardt, T., Ziefle, M., Schroeder, U. : The Effect of Tangible Artifacts, Gender and Subjective Technical Competence on Teaching Programming to Seventh Graders. In : Teaching Fundamentals Concepts of Informatics. pp. 61–71 (2010)
3. Cliburn, D.C. : Experiences with the LEGO Mindstorms Throughout the Undergraduate Computer Science Curriculum. In : Proceedings. Frontiers in Education. 36th Annual Conference. pp. 1–6 (2006)
4. Cooper, S., Dann, W., Pausch, R., Pausch, R. : Teaching Objects-First in Introductory Computer Science. SIGCSE Bull. **35**(1), 191–195 (2003)
5. Fagin, B. : Ada/Mindstorms 3.0. IEEE Robotics & Automation Magazine **10**(2), 19–24 (2003)
6. Garrett, A., Thornton, D. : A Web-Based Programming Environment for LEGO Mindstorms Robots. In : Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2. pp. 349–350 (2005)

7. Hodges, S., Scott, J., Sentance, S., Miller, C., Villar, N., Schwiderski-Grosche, S., Hammil, K., Johnston, S. : .NET Gadgeteer : A New Platform for K-12 Computer Science Education. In : Proceeding of the 44th ACM Technical Symposium on Computer Science Education. pp. 391–396 (2013)
8. Krathwohl, D.R. : A Revision of Bloom's Taxonomy : An Overview. Theory Into Practice **41**(4), 212–218 (2002)
9. Magnenat, S., Riedo, F., Bonani, M., Mondada, F. : A Programming Workshop Using the Robot "Thymio II" : The Effect on the Understanding by Children. In : 2012 IEEE Workshop on Advanced Robotics and its Social Impacts. pp. 24–29 (2012)
10. Major, L., Kyriacou, T., Brereton, P. : The Effectiveness of Simulated Robots for Supporting the Learning of Introductory Programming : a Multi-Case Case Study. Computer Science Education **24**(2-3), 193–228 (2014)
11. Maschietto, M., Soury-Lavergne, S. : Designing a Duo of Material and Digital Artifacts : the Pascaline and Cabri Elem e-Books in Primary School Mathematics. ZDM **45**(7), 959–971 (2013)
12. Meerbaum-Salant, O., Armoni, M., Ben-Ari, M. : Learning Computer Science Concepts with Scratch. Computer Science Education **23**(3), 239–264 (2013)
13. Papert, S., Harel, I. : Situating Constructionism. Constructionism **36**(2), 1–11 (1991)
14. Przybylla, M., Romeike, R. : Key Competences with Physical Computing. KEYCIT 2014–Key Competencies in Informatics and ICT (Preliminary Proceedings) p. 216 (2014)
15. Qian, Y., Lehman, J. : Students' Misconceptions and Other Difficulties in Introductory Programming : A Literature Review. ACM Trans. Comput. Educ. **18**(1), 1–24 (2017)
16. Schmidt, A. : Increasing Computer Literacy with the BBC micro:bit. IEEE Pervasive Computing **15**(2), 5–7 (2016)
17. Sentance, S., Schwiderski-Grosche, S. : Challenge and Creativity : Using .NET Gadgeteer in Schools. In : Proceedings of the 7th Workshop in Primary and Secondary Computing Education. pp. 90–100. ACM (2012)
18. Sentance, S., Waite, J., Hodges, S., MacLeod, E., Yeomans, L. : Creating Cool Stuff : Pupils' Experience of the BBC micro :bit. In : Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education. pp. 531–536 (2017)
19. Sentance, S., Waite, J., Yeomans, L., MacLeod, E. : Teaching with Physical Computing Devices : the BBC micro:bit Initiative. In : Proceedings of the 12th Workshop on Primary and Secondary Computing Education. pp. 87–96 (2017)
20. Wang, P., Renna, I., Amiel, F., Zhang, X. : Learning with Robots in CS and STEM Education : A Case Study with ISEP-R0B0. In : Proceedings of the 4th Workshop on Robots for Learning at ACM/IEEE HRI 2018. pp. 16–21 (2018)
21. Weintrop, D., Wilensky, U. : Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. ACM Trans. Comput. Educ. **18**(1), 3 :1–3 :25 (2017)