Réalité augmentée et interface tangible au service des novices en programmation

Julie Henry, Bruno Dumas

Réalité augmentée et interface tangible au service des novices en programmation

Augmented Reality and Tangible Interaction to Help Programming Novices

Julie Henry
julie.henry@unamur.be
Faculté Informatique, Institut NADI, Université de Namur
Namur, Belgique

Bruno Dumas
bruno.dumas@unamur.be
Faculté Informatique, Institut NADI, Université de Namur
Namur, Belgique

ABSTRACT

Introductory programming courses are a challenge for any teacher and a barrier for students. Among other factors, novice programmers often have misconceptions, leading them to build incorrect mental models. A *concept inventory* coupling tangible interfaces and augmented reality (TARCI) has been developed according to a design-oriented methodology, in close collaboration with the future users of the system (students, teachers). With TARCI, teachers should quickly become aware of misconceptions in their students and thus be able to correct them. Preliminary tests were carried out during development, in particular to improve TARCI: first with eight experts and then with nine students. These tests show promising results.

CCS CONCEPTS

• **Human-centered computing** → **HCI theory, concepts and models**; *Pointing*; Visualization techniques.

KEYWORDS

Teaching programming, Concept inventory, Design oriented research, Misconceptions, Mental model

RÉSUMÉ

Les cours d'introduction à la programmation constituent un défi pour tout enseignant et un obstacle pour les étudiants. Entre autres facteurs, les programmeurs novices ont souvent des représentations erronées, ce qui les conduit à construire des modèles mentaux incorrects. Un *concept inventory* couplant interface tangible et réalité augmentée (CITRA) a été développé selon une méthodologie orientée-conception, en étroite collaboration avec les futurs utilisateurs du système (étudiants, enseignants). Avec CITRA, les enseignants devraient prendre conscience rapidement des représentations erronées présentes chez leurs étudiants et ainsi, pouvoir les corriger. Des tests préliminaires ont été menés durant le développement, notamment pour améliorer CITRA : d'abord avec huit experts et ensuite, avec neuf étudiants. Ces tests montrent des résultats prometteurs.

MOTS-CLÉS

Enseignement de la programmation, Concept inventory, Recherche orientée par la conception, Représentations erronées, Modèle mental

ACM Reference Format:
Julie Henry and Bruno Dumas. 2021. Réalité augmentée et interface tangible au service des novices en programmation: Augmented Reality and Tangible Interaction to Help Programming Novices. In *32e Conférence Francophone sur l'Interaction Homme-Machine (IHM '21 Adjunct), April 13–16, 2021, Virtual Event, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3451148.3458643

1 INTRODUCTION

Depuis des décennies, l'apprentissage de la programmation est considérée difficile pour les novices [18], quel que soit leur âge. Les cours d'introduction à la programmation constituent un défi pour tout enseignant. Ils se traduisent souvent par un taux d'échec et d'abandon assez élevé chez les étudiants. De plus en plus de dispositifs d'aide à l'apprentissage sont développés pour assister l'exploration des concepts de base de la programmation et de la pensée informatique [5, 13, 25]. L'interaction tangible, notamment, est une approche privilégiée pour travailler avec des enfants [21]. Les exemples d'initiation à la programmation en utilisant un dispositif tangibles existent [7–10, 15, 16, 24]. De façon étonnante, cette approche est très peu présente lorsqu'il s'agit d'enseigner à des jeunes adultes [6]. Les interfaces tangibles sont parfois même couplées à de la réalité augmentée pour promouvoir la pensée computationnelle [2, 4, 11], rendant le processus d'apprentissage actif.

Si de nombreux facteurs entrent en jeu lorsqu'il s'agit d'expliquer les difficultés éprouvées par les programmeurs débutants [14], un facteur plus particulièrement peut aider les enseignants à améliorer leur enseignement : les représentations erronées [22] que possèdent préalablement les étudiants. Celles-ci entravent leur progrès et peuvent les décourager de poursuivre leur apprentissage. Ces représentations erronées sont identifiables à l'aide d'un *concept inventory* (CI), à savoir un test à choix multiples basé sur la recherche qui "mesure les connaissances d'un étudiant sur un ensemble de concepts tout en capturant les représentations erronées qu'il/elle peut avoir sur le sujet considéré" [23].

Parce que nous voyons un intérêt à mettre des étudiants universitaires en présence de dispositifs tangibles et parce que le CI a fait ses preuves en ce qui concerne les concepts de bases en programmation [12], l'hypothèse est posée qu'un CI couplant interface tangible et réalité augmentée (CITRA) pourrait aider les enseignants

à prendre conscience, lors de séances de travaux pratiques, des représentations erronées présentes chez leurs étudiants et à rectifier le tir de manière appropriée lors des séances suivantes [17]. Les étudiants eux-mêmes, à travers la manipulation du dispositif, devraient percevoir leurs incompréhensions et les corriger. Pour qu'un tel dispositif réponde au mieux aux besoins, son développement doit se faire en collaboration avec les futurs utilisateurs, à savoir les étudiants mais également les enseignants et autres professionnels de l'apprentissage.

2 DÉVELOPPEMENT DU DISPOSITIF CITRA

CITRA est développé suivant une méthodologie inspirée de la *design-oriented research* [1] (Fig. 1 et 2). Il s'agit de mener un processus itératif qui articule des phases de conception, de mise en œuvre auprès de différents publics et d'analyse des données collectées. De ce fait, le développement est non seulement une collaboration entre le chercheur développant le dispositif et des experts, mais également entre le chercheur et des futurs utilisateurs, à savoir des étudiants et des professionnels de l'apprentissage (enseignants, assistants, tuteurs, etc.).

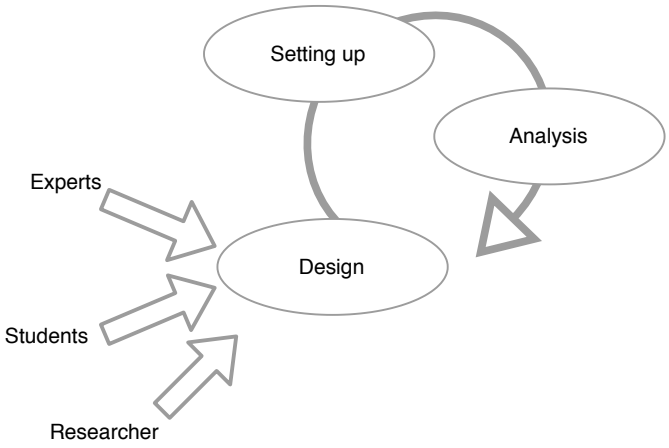

Figure 1: Une méthodologie de recherche itérative et participative

Deux aspects sont à distinguer dans CITRA : l'aspect didactique, à savoir les représentations erronées à identifier et la façon de le faire, et la conception de l'interface, à savoir la forme du dispositif et la définition des interactions possibles avec celui-ci. Pour chacun de ces aspects, des experts ont été invités à collaborer.

2.1 Des représentations erronées tangibles et augmentées

CITRA est un CI. En ce sens, il doit permettre de mesurer la compréhension de concepts de base en programmation. Il ne s'agit pas de rendre tangible un questionnaire à choix multiple. L'utilisateur est invité à résoudre une série de problèmes de programmation aux moyens de blocs tangibles. Chaque bloc est "augmenté" dans la lignée de systèmes basés sur les frameworks ARToolkit ou

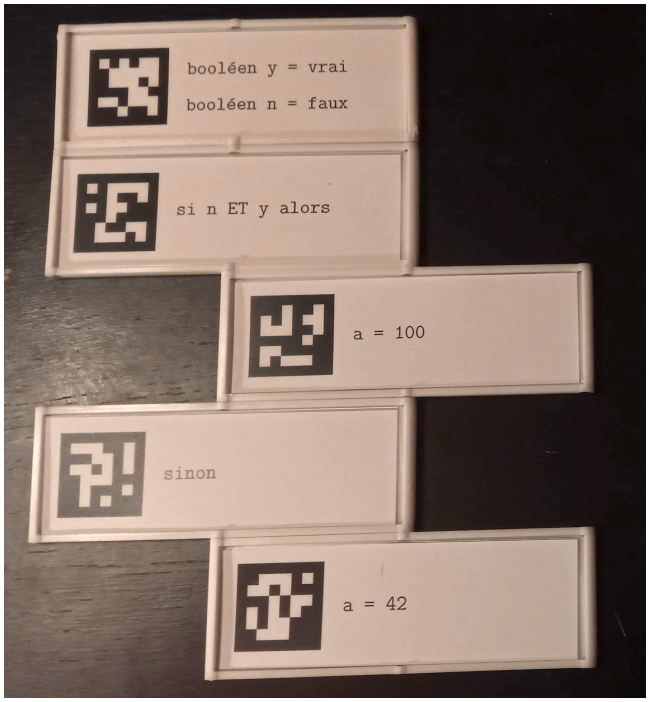

Figure 2: Des blocs tangibles à assembler

ARTagest[3] : des retours adaptés aux réponses proposées par l'utilisateur mettent l'accent sur les représentations erronées présentes dans celles-ci.

Il s'agit donc de construire un programme répondant à un court problème centré sur la manipulation d'un concept de base en programmation. Dans le cas du prototype décrit ici, les concepts sont la variable, la structure conditionnelle et la boucle. Plusieurs problèmes ont ainsi été prévus, permettant d'identifier des représentations erronées majeures telles que le sens de lecture d'une affectation de variable, le nombre d'itérations d'une boucle en fonction de sa condition de sortie ou encore la compréhension de l'algèbre de Boole au sein d'une condition. Ces problèmes s'inspirent des travaux de recherche de [12].

Un problème est composé d'un énoncé et d'un set de blocs à assembler qui lui est propre (Fig. 2). Chaque bloc représente une instruction écrite en pseudo-code. Ces instructions peuvent être incorrectes, illustrant de façon directe quelques-unes des représentations erronées que peuvent avoir les apprenants. La solution proposée par l'assemblage de blocs doit être évaluée et le résultat de cette évaluation doit être signifié d'une façon explicite. Ainsi, les informations suivantes doivent être fournies : les valeurs des variables utilisées dans le programme lorsque celui-ci est exécuté ; en cas d'erreur dans le code, la ligne de code impactée et l'erreur produite ; une explication théorique du concept testé corrigeant les représentations erronées qu'un apprenant peut s'en faire (Fig. 5).

2.2 La conception de l'interface

Le choix d'un langage de programmation en blocs tangibles prend son inspiration dans les nombreux dispositifs existants. Les blocs

ont été imprimés en 3D et s'assemblent par magnétisme les uns en-dessous des autres, pour une lecture du programme de haut en bas. L'assemblage est possible selon plusieurs positions, permettant de créer une indentation (Fig. 3, 4 et 5).

Chaque bloc représente une instruction. Dans le prototype, cette instruction a été voulue interchangeable. Il s'agit donc d'une simple bandelette de papier sur laquelle est imprimée l'instruction. Par ailleurs, chaque instruction possède un tag unique lui permettant d'être "scannée et reconnue".

CITRA comprend non seulement des blocs tangibles, mais également une application fonctionnant sur tablette ou smartphone permettant d'augmenter les interactions. Elle est composée de trois écrans : un écran d'accueil, un écran pour scanner et un écran affichant les retours à l'utilisateur. Il est facile de naviguer entre ces écrans : toucher un écran permet de passer à l'écran suivant, un bouton retour permet de revenir en arrière. L'écran d'accueil explique le fonctionnement du dispositif (Fig. 3). Le second écran renvoie ce qui est capturé par la caméra de votre tablette/smartphone. C'est à partir de cet écran que les tags des instructions vont être scannées. Le nombre d'instructions reconnues (et donc de blocs reconnus) est indiqué, permettant de corriger le scan au besoin (Fig. 4). Le troisième écran affiche les retours associés au programme scanné (Fig. 5).

3 ÉVALUATIONS PRÉLIMINAIRES

Les premières itérations de conception de CITRA ont été réalisées en collaboration avec des experts en ingénierie logicielle, en IHM et en didactique de l'informatique, jusqu'à l'obtention d'un premier prototype fonctionnel.

Deux phases de tests ont alors été organisées. La première phase consistait à soumettre le dispositif à huit professionnels de l'enseignement en informatique, responsables de cours ou de séances pratiques, et donc susceptibles d'utiliser le dispositif avec leurs étudiants. La deuxième phase mettait en jeu le dispositif auprès de neuf étudiants suivant un cours d'introduction à la programmation, en première année d'université (Fig. 6). L'objectif de ces tests était d'obtenir des retours sur l'expérience utilisateur avec le prototype, mais également sur le potentiel de CITRA à être utilisé dans un contexte d'enseignement/apprentissage.

3.1 Tests avec le personnel enseignant

Trois problèmes à résoudre étaient proposés durant cette première phase de tests, chacun disposant d'un set de blocs comprenant des instructions en pseudo-code. Des observations ethnographiques ont été réalisées et un questionnaire d'expérience utilisateur (UEQ)[19, 20] a été rempli par l'ensemble des participants.

- Le problème 1 met en jeu un set de 12 blocs et teste la compréhension du concept de variable. Il s'agit d'écrire un programme dont le résultat est un échange de valeur entre deux variables.
- Le problème 2 met en jeu un set de huit blocs et teste la compréhension du concept de structure conditionnelle. Il s'agit d'écrire un programme dont le résultat est l'affectation d'une valeur définie à une variable lorsqu'une condition mettant en jeu l'algèbre de Boole est vérifiée.

Figure 3: Écran d'accueil

- Le problème 3 met en jeu sept blocs et teste la compréhension du concept de boucle. Il s'agit d'écrire un programme respectant un nombre défini d'itérations de boucle.

Six experts (75%) ont trouvé le problème 2 trop difficile pour des novices. De plus, le pseudo-code utilisé pour écrire les instructions a perturbé certains experts. Ainsi deux experts (25%) ont été étonnés que le typage soit explicite ("entier a = 2"). Six experts ont omis le bloc "fin si". Deux experts se disent en difficulté lorsqu'il s'agit d'avoir une vue globale de l'ensemble des blocs disponibles dans un set.

Concernant l'application, des problèmes d'ergonomie ont été signalés pour l'écran permettant de scanner la solution proposée. Le bouton pour lancer le scan était mal positionné, rendant difficile la manipulation de la tablette. De plus, le succès du scan et la reconnaissance des instructions n'étaient pas perçus par les experts. Les retours proposés suite à l'évaluation d'une solution ont également fait l'objet de commentaires. Ainsi, selon les experts, les problèmes d'indentation, générant jusque là un simple avertissement, devaient

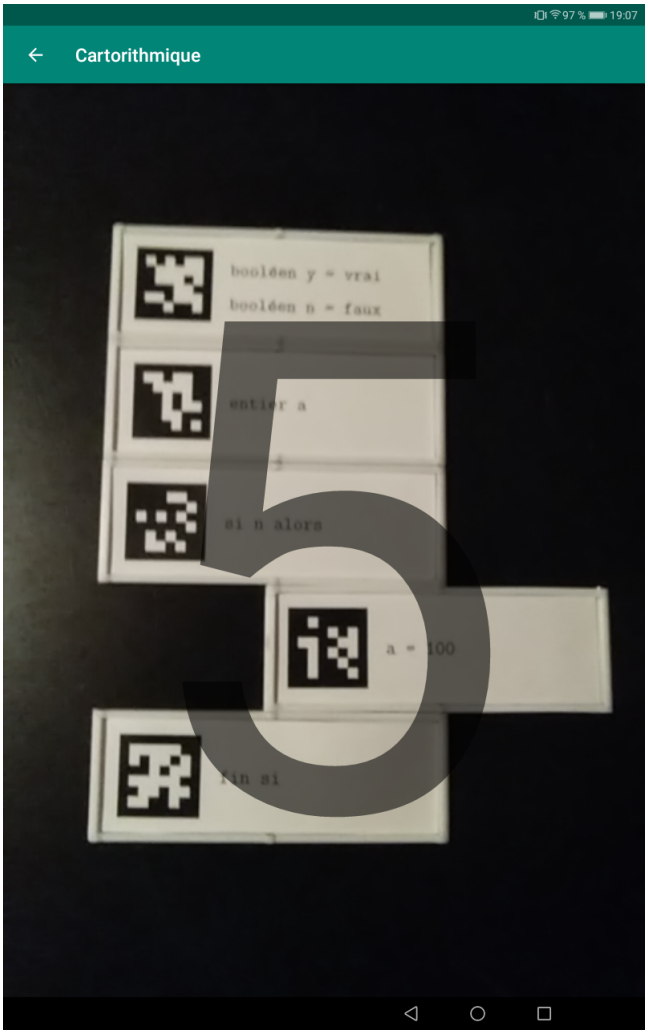

Figure 4: Nombre de blocs reconnus

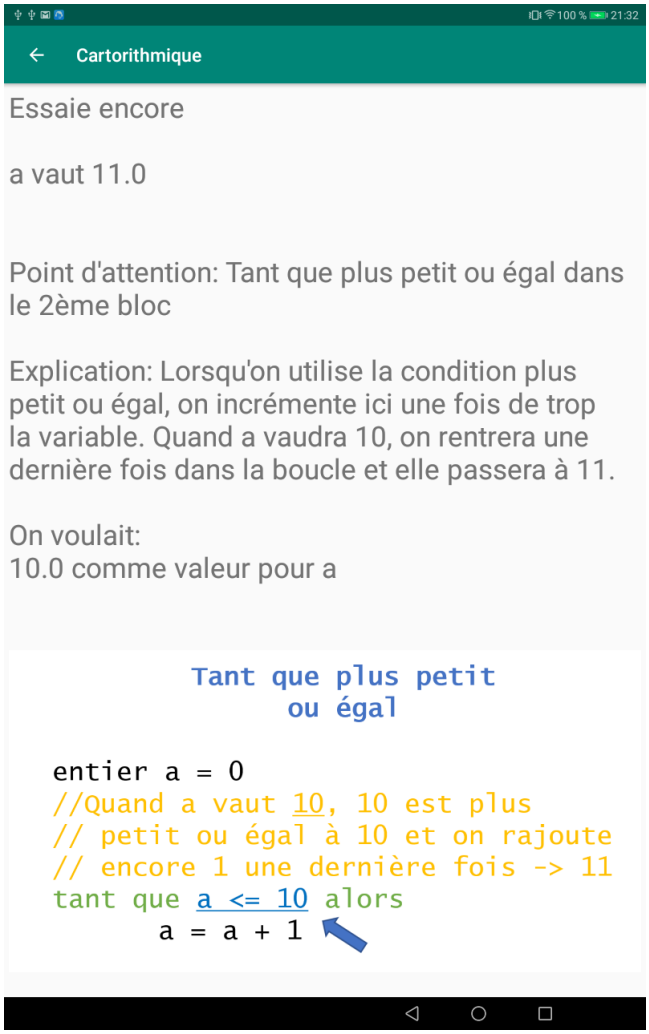

Figure 5: Feedback pour une solution proposée

être signalés comme une erreur. La couleur rouge, utilisée pour mettre l'accent sur certaines informations, a été jugée trop agressive. Enfin, la disposition générale des informations, les différentes polices utilisées et la taille insuffisante des caractères rendaient la lecture difficile. La plupart des experts n'ont pas pris connaissance de l'entièreté des informations, plus attirés visuellement par les images que par le texte.

De façon générale, les experts ont accueilli positivement le dispositif. En outre, celui-ci s'avère plutôt facile à utiliser et à apprendre : une erreur observée au niveau des interactions avec le dispositif n'est plus reproduite par la suite. Pour appuyer ces résultats, un questionnaire d'expérience utilisateur a été rempli, à la fin du test, par chacun des experts. Six critères ont été évalués via une échelle de Likert allant de -3 à 3, par pas de 1 : attractivité, perspicacité, efficacité, fiabilité, stimulation et innovation.

Les données collectées durant cette première phase de tests ont conduit à la version du prototype décrite dans cet article. Les instructions "fin si" et "fin tant que" ont été supprimées. L'indentation

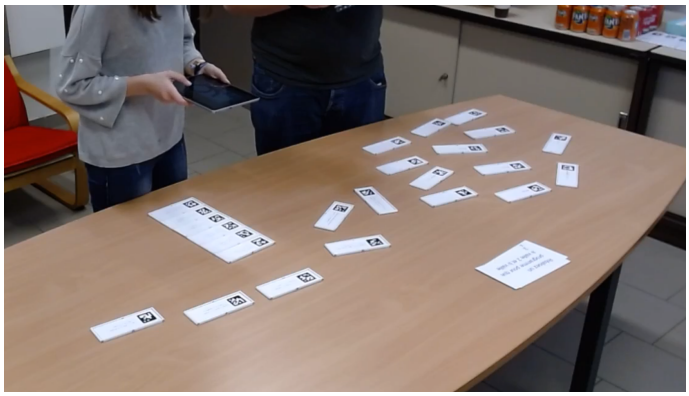

Figure 6: Un étudiant utilisant le prototype

des blocs a été rendue obligatoire. Le bouton lançant un scan a été

remplacé par une simple pression sur l'écran tactile. Enfin, l'ensemble des messages à destination de l'utilisateur a été revu pour gagner en lisibilité.

3.2 Tests avec les étudiants

Les neuf étudiants soumis à la deuxième phase de tests suivaient, depuis un quadrimestre, un cours universitaire d'introduction à la programmation (en 1ère année). Ces étudiants étaient issus de deux filières différentes : quatre d'entre eux étudiaient l'informatique (INFO) et cinq, l'ingénierie de gestion (INGE). Ils ont été choisis aléatoirement parmi un groupe de 140 étudiants. L'expérimentation a été filmée dans son intégralité selon deux points de vue : un point de vue global captant les interactions des étudiants avec les blocs et avec le chercheur présent, et un point de vue plus focalisé sur les interactions des étudiants avec l'application (sur tablette). Les vidéos obtenues ont été annotées et les dialogues ont été entièrement retranscrits.

Seul le problème 1 a été proposé aux 9 étudiants. Pour mesurer l'effet du pseudo-code, compte tenu des observations faites chez les experts, les instructions ont été "traduites" (avec les représentations erronées) en langage naturel.

Pseudo-code : *integer a = 3*
Langage naturel : *Je déclare une variable de type entier, je la nomme "a" et je l'initialise à 3*

Les deux sets de blocs ont été proposées en parallèle aux étudiants. Ceux-ci étaient libres de choisir par lequel ils souhaitaient commencer.

Six étudiants sur neuf (5 INGE et 1 INFO - 66%) ont préféré commencer le test avec les instructions rédigées en langage naturel. Un étudiant a choisi de commencer avec les instructions en pseudo-code parce qu'il craignait de mal saisir certaines nuances en lisant le langage naturel. Sur les six étudiants qui ont été confrontés d'abord au langage naturel, seuls deux étudiants (22%) ne sont pas parvenus à résoudre le problème. Pour un des deux (1 INFO), l'utilisation du pseudo-code ne l'a pas plus aidé à proposer une solution. Trois de ces six étudiants ont mal choisi le bloc décrivant une assignation de variable, montrant qu'ils inversaient le sens de lecture du symbole "=".

Sur l'ensemble des tests, le langage naturel a été considéré comme plus difficile par 8 étudiants (88%), notamment à cause de la longueur des textes. Le pseudo-code est, selon les étudiants, plus proche de ce qu'ils connaissent (le langage Python). Pourtant, un étudiant (1 INGE) a précisé mieux comprendre l'instruction (dans son fonctionnement) grâce au langage naturel et trois étudiants (3 INFO) ont ignoré le typage, ne le comprenant pas. Les étudiants admettent qu'il n'en aurait peut-être pas été de même s'ils avaient manipulé le dispositif avant d'avoir commencé les cours d'introduction à la programmation. Un étudiant a suggéré une coloration des blocs par type d'instructions.

Seuls deux étudiant ont connu des difficultés avec l'application, notamment pour revenir sur un écran précédent et pour scanner (en touchant l'écran). Ces difficultés ont été facilement surmontées.

Au niveau des messages informatifs, trois étudiants (3 INFO - 33%) n'ont pas pu localiser l'erreur, ne comprenant pas l'information donnée (le numéro du bloc erroné). Un d'eux a précisé que pour lui, un "bloc" correspondait à plusieurs instructions et non une seule. Quatre étudiants (44%) n'ont pas lu l'intégralité des messages affichés après le scan, se contentant des premières lignes (localisation de l'erreur).

Un seul étudiant (1 INGE - 11%) a réussi le problème dans les deux langages en un seul essai. Un seul étudiant (1 INFO) a expérimenté le dispositif avant de se lancer dans la résolution du problème, notamment pour voir le type de retours fournis.

De façon générale, les étudiants ont accueilli positivement le dispositif et l'idée qu'il pourrait être utilisé durant les séances pratiques. Un seul étudiant (1 INGE) a évoqué le risque d'un côté ludique trop important, en opposition au caractère sérieux de l'apprentissage.

4 ANALYSE

En ce qui concerne la première phase de tests, les résultats du questionnaire UEQ (Fig. 7) ont été comparés aux valeurs d'un dataset[1]. Il en ressortait que deux aspects étaient à améliorer plus particulièrement, la compréhensibilité (ou clarté - Est-il facile de comprendre comment fonctionne le système ou de se familiariser simplement avec le système ?) et la contrôlabilité (ou fiabilité - . L'utilisateur a-t-il le sentiment de contrôler l'interaction ? L'interaction avec le système est-elle sûre et prévisible ?).

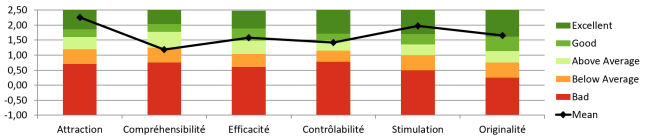

Figure 7: Les résultats du questionnaire d'expérience utilisateur

Durant la deuxième phase des tests, les interactions avec l'application semblent avoir été améliorées. Du travail reste à faire sur les retours, notamment parce que les étudiants ne lisent pas l'intégralité des informations affichées et passent donc à côté de certains apprentissages tels que la prise de conscience de leurs idées fausses notamment. Par contre, l'utilisation du dispositif a permis de se rendre compte des difficultés qu'ont les étudiants à lire les instructions en langage naturel (et finalement à comprendre les concepts de base en programmation). Leur préférence pour le pseudo-code est expliquée par sa proximité avec le langage de programmation Python et par les routines qu'ils ont mises en place lors de leur apprentissage. Il apparaît ainsi qu'il sont capables d'écrire une assignation, sans en comprendre le fonctionnement, juste par rétention d'un motif à reproduire.

5 CONCLUSION

Un *concept inventory* couplant interface tangible et réalité augmentée (CITRA) a été élaboré selon une méthodologie itérative mettant l'accent sur la collaboration entre les experts, les étudiants et les enseignants. Il vise à aider les enseignants en charge des cours d'introduction à la programmation à prendre conscience des représentations erronées que possèdent leurs étudiants. Parce que

[1]Données issues de 452 études auxquelles ont participé 20 190 personnes et portant sur l'analyse de logiciels, pages webs, réseaux sociaux, web shops, etc. - https://www.ueq-online.org/

celles-ci entravent leur progrès et peuvent les décourager de poursuivre leur apprentissage, il s'agit de les identifier rapidement pour les corriger.

CITRA se compose d'une application fonctionnant sur tablette ou smartphone et de blocs tangibles. Pour répondre à un problème posé, l'utilisateur doit assembler les blocs. Chaque bloc représente une instruction écrite en pseudo-code. Ces instructions peuvent être incorrectes, illustrant de façon directe des représentations erronées que possèdent les novices en programmation. La solution proposée est évaluée via l'application qui affiche alors des informations permettant notamment de corriger les représentations erronées présentes dans la solution.

Le prototype développé a été soumis à deux phases de tests : une première phase auprès de huit professionnels de l'enseignement en charge de cours ou de séances pratiques en programmation ; une deuxième phase auprès de neuf étudiants suivant un cours d'introduction à la programmation. L'objectif de ces tests était d'obtenir des retours sur l'expérience utilisateur avec le prototype, mais également sur le potentiel de CITRA à être utilisé dans un contexte d'enseignement/apprentissage.

Bien que les résultats obtenus itérations soient prometteurs, il reste encore du travail. Tout d'abord, il s'agit d'améliorer le retour d'information présenté par l'application. En effet, CITRA n'aide visiblement pas les étudiants à prendre conscience de leurs représentations erronées, même si l'enseignant peut lui en prendre conscience. Deuxièmement, il est nécessaire de développer des problèmes supplémentaires pour permettre d'identifier plus de représentations erronées. Des tests supplémentaires doivent être réalisés avec de vrais novices (en début de parcours) pour mieux rendre compte de l'impact de l'utilisation de CITRA dans un contexte d'apprentissage de la programmation. Enfin, il serait intéressant de comparer l'usage de cette solution à une approche non outillée pour mesurer l'apport de la combinaison interface tangible et réalitée augmentée à l'apprentissage de la programmation.

RÉFÉRENCES

[1] Terry Anderson and Julie Shattuck. 2012. Design-based research : A decade of progress in education research ? *Educational researcher* 41, 1 (2012), 16–25.

[2] Barbara Cleto, João Martinho Moura, Luís Ferreira, and Cristina Sylla. 2018. Codecubes-playing with cubes and learning to code. In *Interactivity, Game Creation, Design, Learning, and Innovation*. Springer, 538–543.

[3] Mark Fiala. 2005. Comparing ARTag and ARToolkit Plus fiducial marker systems. In *IEEE International Workshop on Haptic Audio Visual Environments and their Applications*. IEEE, 6–pp.

[4] Anna Gardeli and Spyros Vosinakis. 2019. ARQuest : A tangible augmented reality approach to developing computational thinking skills. In *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*. IEEE, 1–8.

[5] Paul Gross and Kris Powers. 2005. Evaluating assessments of novice programming environments. In *Proceedings of the first international workshop on Computing education research*. ACM, 99–110.

[6] Julie Henry, Bruno Dumas, and Antoine Bodart. 2018. Programmation tangible pour les enfants : analyse de l'existant, classification et opportunités. 30eme conférence francophone sur l'interaction homme-machine, Oct 2018, Brest, France. 9p.

[7] Michael S. Horn and Robert J. K. Jacob. 2007. Designing Tangible Programming Languages for Classroom Use. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction* (Baton Rouge, Louisiana) *(TEI '07)*. Association for Computing Machinery, New York, NY, USA, 159–162.

[8] Michael S. Horn and Robert J. K. Jacob. 2007. Tangible Programming in the Classroom with Tern. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems* (San Jose, CA, USA) *(CHI EA '07)*. Association for Computing Machinery, New York, NY, USA, 1965–1970.

[9] Michael S Horn, Erin Treacy Solovey, R Jordan Crouser, and Robert JK Jacob. 2009. Comparing the use of tangible and graphical programming languages for informal science education. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 975–984.

[10] Felix Hu, Ariel Zekelman, Michael Horn, and Frances Judd. 2015. Strawbies : explorations in tangible programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*. ACM, 410–413.

[11] Qiao Jin, Danli Wang, Xiaozhou Deng, Nan Zheng, and Steve Chiu. 2018. AR-Maze : a tangible programming tool for children based on AR technology. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*. 611–616.

[12] Henry Julie and Dumas Bruno. 2019. Towards the Identification of Profiles based on the Understanding of Programming Concepts : the Case of the Variable. In *2019 IEEE FIE Conference*. IEEE, 1–8.

[13] Caitlin Kelleher and Randy Pausch. 2005. Lowering the barriers to programming : A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)* 37, 2 (2005), 83–137.

[14] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A study of the difficulties of novice programmers. In *ACM SIGCSE Bulletin*, Vol. 37. ACM, 14–18.

[15] Jean-Baptiste Marco, Nadine Baptiste-Jessel, and Philippe Truillet. 2018. TaBGO : programming with tangibles blocks. In *Proceedings of the 30th Conference on l'Interaction Homme-Machine*. 179–185.

[16] Sofia Papavlasopoulou, Michail N Giannakos, and Letizia Jaccheri. 2017. Reviewing the affordances of tangible programming languages : Implications for design and practice. In *2017 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 1811–1816.

[17] Yizhou Qian, Susanne Hambrusch, Aman Yadav, Sarah Gretter, and Yue Li. 2020. Teachers' perceptions of student misconceptions in introductory programming. *Journal of Educational Computing Research* 58, 2 (2020), 364–397.

[18] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming : A review and discussion. *Computer science education* 13, 2 (2003), 137–172.

[19] Martin Schrepp, Andreas Hinderks, and Jörg Thomaschewski. 2014. Applying the user experience questionnaire (UEQ) in different evaluation scenarios. In *International Conference of Design, User Experience, and Usability*. Springer, 383–392.

[20] Martin Schrepp, Andreas Hinderks, and Jörg Thomaschewski. 2017. Design and Evaluation of a Short Version of the User Experience Questionnaire (UEQ-S). *International Journal of Interactive Multimedia and Artificial Intelligence* 4 (01 2017), 103.

[21] Orit Shaer, Eva Hornecker, et al. 2010. Tangible user interfaces : past, present, and future directions. *Foundations and Trends® in Human–Computer Interaction* 3, 1–2 (2010), 4–137.

[22] Juha Sorva, Ville Karavirta, and Lauri Malmi. 2013. A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)* 13, 4 (2013), 15.

[23] Lea Wittie, Anastasia Kurdia, and Meriel Huggard. 2017. Developing a concept inventory for computer science 2. In *FIE Conference*. IEEE, 1–4.

[24] Peta Wyeth and Helen C Purchase. 2002. Tangible programming elements for young children. In *CHI'02 extended abstracts on Human factors in computing systems*. ACM, 774–775.

[25] Junnan Yu and Ricarose Roque. 2018. A survey of computational kits for young children. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*. ACM, 289–299.