TaBGO : Programmation par blocs tangibles Résumé

Jean-Baptiste Marco, Nadine Jessel, Philippe Truillet

TaBGO : Programmation par blocs tangibles

Jean-Baptiste Marco
Université de Toulouse
IRIT
Toulouse, France
Jean-Baptiste.Marco@irit.fr

Nadine Baptiste-Jessel
Université de Toulouse
IRIT, ESPE
Toulouse, France
Nadine.Baptiste@irit.fr

Philippe Truillet
Université de Toulouse
IRIT
Toulouse, France
Philippe.Truillet@irit.fr

Résumé
Dans cet article, nous décrivons la solution TaBGO (Tangible Block Goes Online) permettant de rendre accessible par des non-voyants l'utilisation d'un langage de programmation par blocs de type Scratch. Ce dispositif comprend des objets tangibles représentant les différents blocs et les différentes variables et un logiciel automatique de reconnaissance de ces objets permettant de générer et d'exécuter le programme conçu par l'utilisateur.

Mots Clés
Algorithmique ; accessibilité ; programmation par blocs ; déficients visuels ; interacteurs tangibles ; reconnaissance optique

Abstract
In this article, we describe the solution TaBGO (Tangible Block Goes Online), which makes a visual programming language like Scratch more accessible to blind and visually impaired people. This system contains some tangible objects representing the different blocks and the different variables and a software recognizing these objects automatically in order to generate and to execute the program produced by the user.

Author Keywords
Algorithmic; accessibility; visual block programming; visually impaired people; tangible objects; optical recognition

CSS concepts
Human-centered computing → **Accessibility** → ***Accessibility systems and tools***

Introduction

L'informatique a joué et joue un rôle important dans l'inclusion des personnes déficientes visuelles dans nos sociétés. Elle permet notamment l'accès à l'information au travers d'outils dédiés (synthèse vocale, plages brailles, etc.). Néanmoins, malgré la résolution de l'Assemblée Générale des Nations Unies [9], dont l'objectif est de « permettre aux personnes handicapées de vivre de façon indépendante et de participer pleinement à tous les aspects de la vie » (cf. article 9), il reste des situations pour lesquelles l'accessibilité reste délicate.
Depuis la réforme du collège en 2016[1], les outils de programmation visuelle par blocs (comme le langage Scratch[2]) sont préconisés pour permettre la découverte de la programmation. Il s'avère que ces outils sont peu, voire pas du tout accessibles ce qui peut amener à l'exemption de l'épreuve d'algorithmique au Diplôme National du Brevet[3].

Nous proposons dans cet article une solution permettant de rendre accessible l'utilisation de ces langages par blocs pour des élèves non-voyants. La solution globale comprend deux aspects. Il s'agit d'une part de proposer un système permettant la manipulation physique de briques en bois représentant les blocs de base du langage Scratch. Ces pièces, une fois assemblées, forment un algorithme qui est interprété par un ordinateur à l'aide d'outils de reconnaissance visuelle. D'autre part, le deuxième enjeu est d'intégrer des outils d'interaction (robots, sorties tactiles ou sonores) rendant l'exécution des algorithmes compréhensibles pour un utilisateur non-voyant.

Travaux connexes

Plusieurs travaux de recherche se sont intéressés à l'apprentissage par objets tangibles de la programmation pour des étudiants non-voyants ou non.

Par exemple, Capovilla et al. [3] proposent un dispositif haptique constitué d'un plateau et de briques LEGO permettant de représenter dans l'espace les entrées et la structure d'un algorithme. Ainsi un élève non-voyant peut percevoir le contenu d'un algorithme par le toucher, ce qui facilite sa compréhension du programme.
Zuckerman et al. [4] ont proposé « *Flow Blocks* » qui permet la découverte du principe de causalité de la programmation en proposant la construction de structures physiques constituées de blocs connectés magnétiquement et un feedback visuel résultant de la structure créée.
Un autre projet intéressant est Quetzal introduit par [10] un langage de programmation avec des objets tangibles interconnectables et une application de reconnaissance optique afin de contrôler des LEGO Mindstorms.

Sanchez et Aguayo [7] ont quant à eux implémenté un langage de programmation basé sur des interfaces sonores. Ce langage utilise un synthétiseur vocal et propose des types de variables permettant d'enregistrer des sons.

[1] Bulletin Officiel spécial n°11 du 26 novembre 2015, http://cache.media.education.gouv.fr/file/MEN_SPE_11/35/1/BO_SPE_11_26-11-2015_504351.pdf

[2] https://preview.scratch.mit.edu

[3] Arrêté du 28 mars 2018, article 4, https://www.legifrance.gouv.fr/affichTexte.do;jsessionid=A2396BAB1360E8551D4D85DEFFF0A658.tplgfr32s_2?cidTexte=JORFTEXT000036843926&dateTexte=&oldAction=rechJO&categorieLien=id&idJO=JORFCONT000036843607

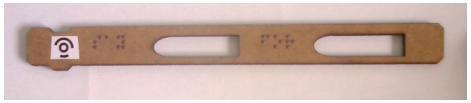

Figure 0 : Bloc tangible « Scratch »

Enfin, le robot pédagogique Roamer [6] a été utilisé au Royaume-Uni. Ce robot a la capacité de suivre des séquences d'instructions simples (Avancer, Reculer, Tourner à droite …) et son clavier a pu être adapté aux non-voyants en utilisant des matériaux tactiles. L'enfant peut ainsi concevoir un algorithme, l'exécuter et comprendre la sortie du programme en percevant tactilement les feedbacks du robot.

Ces solutions, quoiqu'intéressantes ne s'appliquent que peu à la programmation par blocs pour non-voyants. En effet, ces langages tels que Scratch offrent une interface visuelle particulière dont les instructions sont des blocs ayant une forme spécifique au langage. Les travaux évoqués précédemment proposent seulement des outils généraux d'apprentissage de l'algorithmique pour les déficients visuels et n'offrent pas, par exemple, la possibilité d'emboîter des blocs d'instructions pour concevoir un algorithme.

Quelques enseignants spécialisés en France se sont aussi penchés sur le problème de la construction des algorithmes de manière *débranchée* (c'est à dire programmer sans utiliser d'ordinateur). La Structure d'Accompagnement des Elèves Déficients Visuels à Nancy [5] a par exemple proposé une adaptation en relief du logiciel Scratch. Le prototype est constitué de briques en plastique imprimées en 3D qui sont conformes à la représentation des instructions du langage Scratch. De plus, des étiquettes en braille décrivant chaque instruction ont été collées sur le dessus de chaque pièce. Néanmoins, ce prototype permet uniquement la manipulation physique des pièces et ne propose pas de retour du programme par le logiciel.
Boissel [2] a quant à elle développé un outil, -la mallette « *Accessi DV Scratch* »-rendant possible l'utilisation du logiciel Scratch par des non-voyants. Ce dispositif est constitué d'un ensemble de pièces LEGO emboitables et adaptées en braille et pouvant reproduire toutes les commandes offertes par le logiciel. La lecture d'un script est aisée pour le déficient visuel qui parvient à suivre le programme tactilement grâce à l'emboîtement des instructions. De plus, la modification d'un script est également possible. Enfin, Aymard [1] a aussi mené un projet de création d'un support débranché de Scratch adapté aux non-voyants. Il a, comme pour les deux projets précédents, proposé un prototype pour l'utilisation de Scratch déconnecté. Son outil comporte un plateau sur lequel seront entreposées des briques aimantées formant ainsi un algorithme. De plus, cette adaptation débranchée offre une perception de l'exécution du script. La fenêtre de visualisation du logiciel a été remplacée par un quadrillage permettant le repérage cartésien et le tracé de dessins en reliefs. Ainsi l'élève peut reconnaître les effets graphiques de l'algorithme représenté par une texture en relief.

Ces travaux spécifiques montrent des adaptations concrètes pour faciliter l'accessibilité de l'algorithmique pour les déficients visuels par le biais de la manipulation de blocs physiques. Ces outils peuvent également favoriser l'interaction entre élèves voyants et non-voyants puisqu'ils peuvent être aisément utilisables par un voyant.
Cependant, ces adaptations comportent des limites car elles n'ont pas été pensées de sorte que l'algorithme puisse être partagé et directement exécuté par un ordinateur rendant la pratique de la programmation restreinte à la conception.

Notre proposition

Description du dispositif

Notre dispositif consiste donc en une version tangible du logiciel Scratch couplé avec une reconnaissance optique des blocs et des valeurs pouvant être contenues (cf. Figure 1). Nous avons réalisé différents prototypes constitués d'un

```
"blocks": {
    "SA/JF:xejUvtEdMgRZ8M": {
        "opcode": "motion_movesteps",
        "next": null,
        "parent": null,
        "inputs": {
            "STEPS": [
                1,
                [
                    4,
                    "3"
                ]
            ]
        },
        "fields": {
        },
        "topLevel": true,
        "shadow": false,
        "x": 391,
        "y": 205
    }
},
```


Figure 2 : Représentation en JSON d'un script Scratch

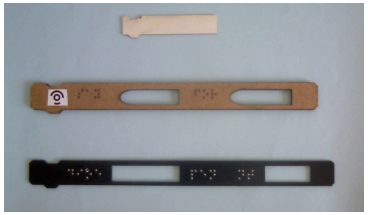

Figure 3 : Les différents prototypes de bloc Scratch réalisés

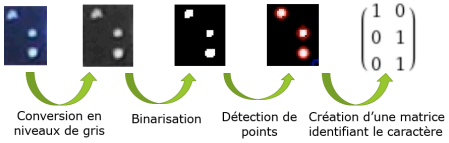

Figure 4 : Reconnaissance du caractère représentant le chiffre 5

ensemble de blocs de bois par découpe laser. Ces pièces correspondent aux briques de base du langage Scratch.

Reconnaissance des algorithmes construits
L'algorithme qui été construit physiquement doit être reconnu par la machine. Cette reconnaissance se déroule en deux phases importantes :

- La reconnaissance de la nature des blocs
- La reconnaissance des paramètres des blocs

La méthode de reconnaissance de la nature des blocs se base sur l'utilisation d'une caméra ou d'un appareil photo pour prendre un cliché du dispositif. Concernant la reconnaissance du type des blocs, nous nous servons de la librairie *TopCodes*[4] permettant de prototyper rapidement des applications de réalité augmentée. Chaque symbole TopCodes encode une valeur unique et nous pouvons ainsi associer chaque bloc physique Scratch. Cette technologie est peu invasive et ne nécessite qu'une caméra pour être utilisée.

Certains blocs du langage Scratch prennent des paramètres en entrée (valeur numérique ou chaîne de caractères). Ces entrées sont représentées dans notre dispositif par des cubes algébriques qui sont des cubes rotatifs munis de points braille (cubarithmes[5]) permettant d'écrire l'ensemble des caractères de l'alphabet, les chiffres et les opérations mathématiques. Les cubes sont de couleur noire à l'origine dont nous avons repeint les points braille en blanc pour faciliter la reconnaissance optique [8]. Cette reconnaissance s'appuie sur l'utilisation de la bibliothèque de traitement d'image OpenCV[6].

Génération de code
Enfin un projet Scratch (version 3) est constitué d'un fichier au format JSON [7] contenant des informations sur l'ensemble du script que l'on produit. Nous avons implémenté un algorithme en Java permettant de générer un texte au format JSON avec les informations reconnues en entrée via les TopCodes et les cubarithmes, conforme à la structure d'un projet Scratch.

La Figure 2 présente la transcription en texte au format JSON du bloc « *Avancer de 3 pas* ». Ce fichier JSON résultant est importé dans l'environnement Scratch 3 et le programme est automatiquement généré.

Intégration
Après avoir réalisé les étapes de reconnaissance d'image et de génération de code, nous avons procédé à la phase d'intégration qui consiste à faire le lien entre ces deux programmes précédents afin de pouvoir générer un projet Scratch automatiquement en une seule exécution.
Nous avons ainsi implémenté un programme en Java réalisant l'intégration complète.

Preuve de Concept et tests réalisés

Conception des blocs physiques
Nous avons réalisé plusieurs prototypes (cf. Figure 3) pour la construction des blocs. Nous avons notamment testé différentes tailles et différents matériaux pour la

[4] http://users.eecs.northwestern.edu/~mhorn/topcodes

[5] http://www.enfant-aveugle.com/spip.php?article265

[6] https://opencv-java-tutorials.readthedocs.io/en/latest/

[7] https://www.w3schools.com/js/js_json_intro.asp

représentation physique des briques du simple carton à des blocs en plastique coloré avec description en braille du bloc.

Nous avons finalement décidé de concevoir des blocs en bois de grande taille (8,5 x 3 cm environ) pour obtenir des objets solides et manipulables. Nous avons pour projet d'ajouter à ceux-ci les informations à la fois en Braille et en noir pour leur identification par des voyants et des non-voyants.

Chaîne des traitements
Le processus de génération d'un projet Scratch à partir d'un algorithme assemblé avec des blocs se déroule de la manière suivante :

1. Reconnaissance des TopCodes
2. Reconnaissance des informations des cubarithmes (cf. Figure 4). Le principe est de reconnaître l'emplacement des cubarithmes dans un premier temps avant de procéder à la reconnaissance des valeurs.
3. Connaissant l'emplacement des cubarithmes, nous pouvons alors appliquer l'algorithme de reconnaissance de caractères braille.
4. Création d'une liste contenant les TopCodes ainsi que les cubarithmes reconnus. Cette liste est ensuite triée relativement aux coordonnées spatiales des objets reconnus.
5. Génération du code JSON pour chaque élément de la liste obtenue, génération.
6. Sauvegarde du fichier généré exploitable directement par Scratch 3.0.

La Figure 5 montre notre dispositif actuel qui comprend un algorithme que nous avons construit à partir des briques tangibles en bois avec les cubes algébriques représentant les valeurs des blocs.

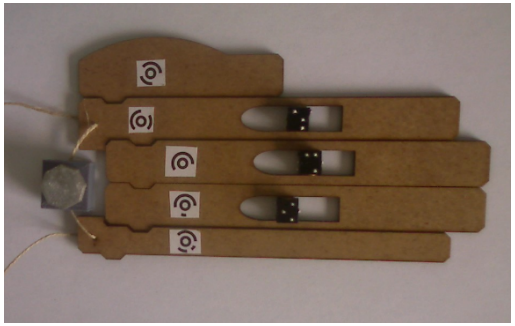

Figure 5 : Algorithme de test de faisabilité

Nous sommes enfin parvenus à transcrire un projet JSON conforme au formalisme Scratch 3 à partir d'un cliché du dispositif physique. La Figure 6 représente le programme Scratch qui a pu être obtenu à la suite de l'exécution du programme Java effectuant toute l'intégration.

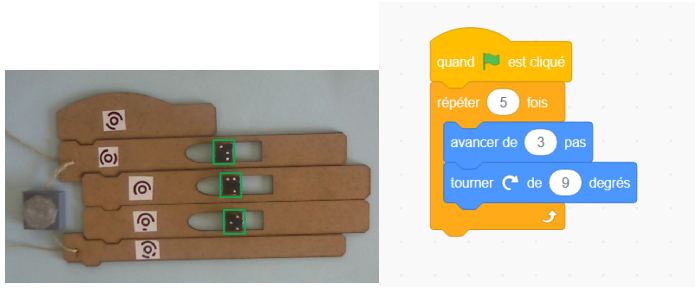

Figure 6 : Programme Scratch tangible et programme produit sur ordinateur

Le Tableau 1 montre l'ensemble des objets reconnus stockés dans la liste Java qui sert à la génération de code.

Entité n°	Type	Valeur	Sémantique
1	TopCode	199	Bloc « Drapeau »
2	TopCode	227	Bloc « Début de boucle »
3	Cubarithme	5	Nombre
4	TopCode	31	Bloc « Avancer de »
5	Cubarithme	3	Nombre
6	TopCode	55	Bloc « Tourner de »
7	Cubarithme	9	Nombre
8	TopCode	229	Bloc « Fin de boucle »

Tableau 1 : Tableau représentant la liste des objets reconnus

Les premiers tests effectués ont permis de valider la faisabilité technique de notre proposition. Le code JSON généré sur une sélection de 50 blocs a été testé avec succès.

Evaluation

L'acceptation par les utilisateurs non-voyants d'un tel dispositif débranché a déjà été évaluée par les travaux de [1, 2, 5]. Notre proposition permet de les rendre autonomes par la génération et l'exécution automatique de leur programme.

Cependant, l'évaluation de l'efficacité pédagogique du dispositif proposé ne sera possible que lorsque le retour non visuel du programme ainsi généré sera implémenté (robot ou sortie sonore, etc.). Pour les programmes Scratch dont l'exécution est seulement sonore, l'évaluation est d'ores et déjà possible.

Dans une situation de classe, les élèves voyants et l'enseignant peuvent visualiser automatiquement sur ordinateur le code Scratch des élèves non-voyants. L'apport par exemple en termes de collaboration et d'inclusion de cette possibilité reste à évaluer.

Conclusion et perspectives

Nous avons réalisé un prototype fonctionnel visant à rendre accessible la programmation par blocs pour les déficients visuels à partir de la construction de l'algorithme jusqu'à son exécution par une machine. Nous avons ainsi conçu un prototype en bois représentant les différents blocs et un algorithme reconnaissant automatiquement à l'aide d'une caméra le programme élaboré et le sauvegardant sous la forme d'un projet exécutable via l'environnement Scratch 3.

De nombreuses possibilités sont encore envisageables pour enrichir notre prototype. Par exemple, il sera intéressant d'implémenter d'autres blocs proposés par Scratch mais le processus reste le même. D'autre part, de nombreux exercices proposés par les enseignants pour apprendre la programmation ont comme résultat le dessin d'une figure géométrique. Dans ce cadre, il est important de réfléchir à un résultat de l'exécution accessible pour les déficients visuels (grâce à des robots, des dispositifs à retour d'effort ou un feedback sonore spatialisé…).

Remerciements

Les auteurs remercient le CampusFab[8] et le CTEB[9] pour leur aide apportée durant le projet.

[8] https://fablab.univ-tlse3.fr

[9] https://www.cteb.fr

Bibliographie

1. Aymard P., Algorithmique Scratch et cécité… Exemple d'un support débranché et adapté, 2018, http://revue.sesamath.net/spip.php?article1082
2. Boissel S., Mallette Accessi DV Scratch « Scratch débranché en braille et gros caractères », dans la nouvelle revue de l'adaptation et de la scolarisation (N°77), pp 183-192, INSHEA, 2017
3. Capovilla D., Krugel J., Hubwieser P., Teaching algorithmic thinking using haptic models for visually impaired students, in LaTiCE, 2013, pp. 167-171, 21-24 March 2013,

   http://dx.doi.org/10.1109/LaTiCE.2013.14
4. Horn M. S., Robert J. K., Tangible programming in the classroom: a practical approach. In CHI '06 Extended Abstracts on Human Factors in Computing Systems (CHI EA '06). ACM, New York, NY, USA, 869-874. DOI=http://dx.doi.org/10.1145/1125451.1125621
5. Observatoire des Ressources Numériques adaptées, Scratch 3D Magnet, janvier 2018, 13 pages,

   http://inshea.fr/sites/default/files/fichier-orna/Orna_Scratch3DMagnet.pdf
6. Renaud P., Virey M., Le Roamer : un robot déjà ancien au service d'apprentissages bien actuels, dans la nouvelle revue de l'adaptation et de la scolarisation (N°52), pp 231-239, INSHEA, 2010
7. Sanchez J., Aguayo F., Blind Learners Programming Through Audio, in CHI 2005, pp. 1769-1772, April 2-7 2005, Portland, https://doi.org/10.1145/1056808.1057018
8. Shreekanth T., Udayashankara V., A Review on Software Algorithms for Optical Recognition of Embossed Braille Characters, in International Journal of Computer Applications, volume 81, No.3, November 2013, pp. 25-35
9. UN General Assembly, Convention on the Rights of Persons with Disabilities: resolution / adopted by the General Assembly, 24 January 2007, A/RES/61/106, available at: http://www.refworld.org/docid/45f973632.html [accessed at: 17 July 2018]
10. Zuckerman O., Grotzer T., and Leahy K. Flow blocks as a conceptual bridge between understanding the structure and behavior of a complex causal system. In Proceedings of the 7th international conference on Learning sciences (ICLS '06). International Society of the Learning Sciences 880-886. https://dl.acm.org/citation.cfm?id=1150162