AccessiBlock : propositions pour un environnement accessible et observable d'apprentissage de la programmation pour l'école élémentaire et le collège

Christophe Declercq

ESPE de l'académie de Nantes, Centre de Recherches en Education de Nantes
Christophe.Declercq@univ-nantes.fr

Résumé

On présente un prototype d'environnement de programmation accessible et observable. Fondé sur la bibliothèque Blockly, il permet de rendre entièrement contrôlable au clavier la saisie et l'édition d'un programme. Plusieurs propositions de micro-mondes d'apprentissage de la programmation sont faites pour rendre l'exécution des programmes perceptible aux élèves en fonction de leurs capacités sensorielles. Un mini-langage de commande est aussi défini pour permettre le cas échéant de substituer à la commande au clavier, une commande vocale. On utilise finalement ce langage de commande pour enregistrer l'activité de l'élève. Les perspectives de ce travail en cours sont, maintenant, d'expérimenter le prototype dans des classes avec des élèves en situation de handicap moteur et/ou sensoriel, de recueillir des données d'apprentissage pour pouvoir ensuite étudier, le plus finement possible, les processus à l'œuvre quand l'élève élabore par tâtonnement, sa solution à une situation problème d'apprentissage de la programmation.

Mots clés : accessibilité numérique, apprentissage de la programmation, analyse de l'activité

1 Introduction

L'utilisation du langage Scratch comme environnement d'apprentissage de la programmation à l'école et au collège – cycles 3 et 4 de l'enseignement scolaire en France, soit de 9 à 14 ans – a été signalée comme présentant un défaut d'accessibilité pour les élèves en situation de handicap : en effet, ni l'édition d'un programme, ni sa lecture ne sont possibles en dehors du mode interactif et visuel. L'élève ne peut déplacer et assembler les blocs que par interaction avec le dispositif de pointage – souris ou alternative.

Patrick Raffinat (2017) a étudié l'environnement Blockly et a montré qu'il était possible de rendre contrôlable au clavier cet environnement d'édition par blocs de programmes. On s'est donc intéressé aux langages pouvant être mis en œuvre à l'aide de l'environnement Blockly, représentant ainsi une alternative potentiellement accessible au langage Scratch.

On a signalé aussi la difficulté particulière de rendre accessible un environnement de programmation : il ne suffit pas de rendre accessible la lecture et l'écriture d'un programme, il faut de plus rendre perceptible et contrôlable son exécution. Cela veut dire que toutes les interactions en entrée et en sortie doivent être perceptibles par l'utilisateur du programme quelles que soient sa situation et ses capacités.

2 Les propositions d'AccessiBlock pour l'accessibilité

Pour permettre la progressivité des apprentissages et adapter l'environnement à ses utilisateurs, nous proposons un environnement d'apprentissage de la programmation qui soit « inclusif », « modulaire » et bien sûr « accessible ».

2.1 Un environnement inclusif

Un environnement inclusif doit permettre à un groupe d'élèves de travailler ensemble à la construction de leur projet ou à l'écriture de leur programme. L'objectif est que le même environnement puisse être utilisé par exemple par un élève déficient visuel ou auditif ou en situation de handicap moteur, scolarisés en inclusion dans un groupe d'élèves « ordinaires ». L'environnement doit donc permettre à la fois l'ajout d'un bloc par glisser/déposer ou par usage de commandes au clavier ou par commandes vocales.

L'observation d'élèves de cycle 3 en situation de résolution de problèmes de labyrinthe avec Blocklygames nous a permis de postuler que les élèves utilisent principalement un schème de construction ascendant dont les opérations de base sont : choisir un bloc, le tester, en choisir un nouveau, le tester, les assembler, tester la construction obtenue, imbriquer dans un bloc structuré – répétition, conditionnelle –, tester la construction etc. Ces modalités sont naturellement mises en œuvre avec l'interface usuelle de Blockly, mais peuvent aussi être commandées par une interface clavier ou vocale, à condition d'identifier les commandes élémentaires à proposer.

Nous avons donc défini une interface permettant de sélectionner un bloc par utilisation des commandes haut, bas – pour déplacement dans une séquence de blocs – et gauche, droite – pour déplacement vers le bloc englobant ou le premier bloc englobé. Ces commandes peuvent être émises par le clavier ou par commande vocale. Les commandes applicables au bloc sélectionné permettent d'ajouter un nouveau bloc dans l'une des quatre directions précédemment citées.

Le langage de commande comprend donc :

- Les commandes de déplacement, par rapport au bloc courant :
  - « Va avant », pour déplacer le focus vers le bloc précédent dans une séquence (raccourci clavier : haut),
  - « Va après », pour déplacer le focus vers le bloc suivant dans une séquence (raccourci clavier : bas),
  - « Va autour », pour déplacer le focus vers le bloc englobant (raccourci clavier : gauche),
  - « Va dans », pour déplacer le focus vers le premier bloc contenu (raccourci clavier : droit),
- Les commandes d'ajout de nouveaux blocs, qui sont associées au menu contextuel du bloc courant :
  - « Ajoute avant », insére un bloc à choisir avant le bloc courant,
  - « Ajoute après », insére un bloc à choisir après le bloc courant,
  - « Ajoute autour », insère le bloc courant dans un nouveau bloc à choisir,
  - « Ajoute dans », insère dans le bloc courant à une position spécifiée, un nouveau bloc à choisir.

2.2 Un environnement modulaire et accessible

La modularité permet d'enrichir progressivement la liste des blocs disponibles dans le langage et ainsi les domaines d'application possibles. Les différentes catégories de blocs sont accessibles par navigation clavier ou commande vocale, dès lors qu'une commande d'ajout a été enclenchée. Le module « calcul sur les entiers » permet d'ajouter les variables entières, les opérations arithmétiques, la boucle de m à n. Le module « conditions » permet d'ajouter les variables logiques, les opérations de comparaisons et les instructions conditionnelles et répétitives. Le module « scripts et événements » permet d'ajouter les notions au programme du cycle 4, à savoir le déclenchement de scripts par des événements particuliers permettant par exemple de déclencher des actions lors de l'appui d'une touche clavier.

Concernant l'accessibilité, l'objectif est de respecter à la fois les normes du web pour les contenus numériques en ligne WCAG 2.0 et les normes des outils auteurs ATAG, car ces dernières s'appliquent au résultat de la conception – le programme dans notre cas – pour prévoir son accessibilité. Techniquement, la solution passe en particulier par la généralisation de l'usage des balises ARIA – application Internet riche accessible – dans l'objectif de rendre le système compatible avec les technologies d'aide – retour vocal en particulier – existantes.

3 Propositions de micro-mondes perceptibles

Dédié à l'apprentissage de la programmation, chaque micro-monde contient une ou plusieurs des notions informatiques – variable, boucle, conditions – selon l'apprentissage visé. L'adaptation aux différents élèves et à leurs capacités se fait principalement sur les primitives de communication fournies dans chaque micro-monde. Les entrées élémentaires se font nécessairement en mode clavier et commande vocale. De même les sorties élémentaires doivent permettre simultanément un affichage à l'écran et par synthèse vocale.

3.1 Un micro-monde musical et sonore

Dédié à l'apprentissage de la boucle, ce micro-monde est fait pour être perceptible pour tous les élèves ayant des capacités auditives, l'objectif étant de reproduire une mélodie donnée. Une alternative visuelle peut être fournie. La compétence travaillée est la reconnaissance de motifs à répéter.

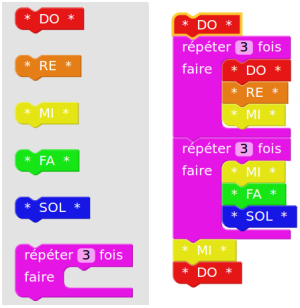

FIGURE 1 – Édition d'un programme musical

Les programmes à éditer sont ici composés exclusivement des notes DO, RE, MI, FA, SOL et de la boucle bornée. Les couleurs proposées ici respectent le codage couleur standard des notes mais peuvent être modifiées selon les capacités de perception visuelle des contrastes.

3.2 Un micro-monde visuel, contrasté et paramétrable

On propose un environnement de création de « pixel art » par déplacement d'un pinceau qui peint les pixels à la demande. C'est une extrapolation de la situation « Robot peintre » (Drot-Delange & Tort, 2018). Les instructions de déplacement fournies sont relatives pour permettre la répétition de motifs dessinés dans diverses orientations.

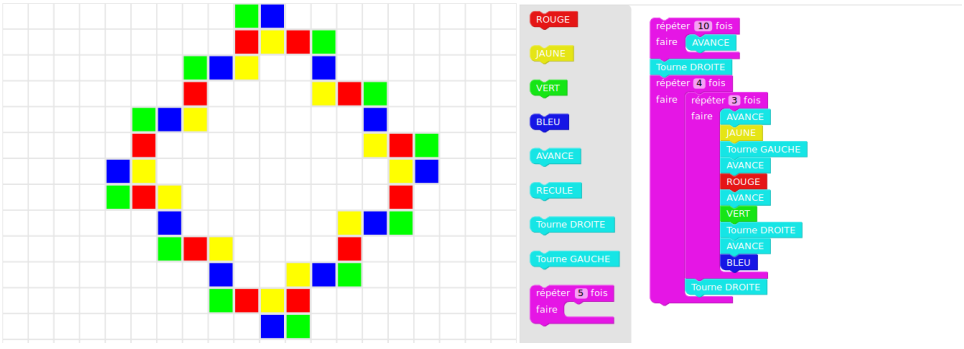

FIGURE 2 – Édition d'un programme de dessin

Le choix d'une taille importante des pixels est fait pour améliorer l'accessibilité pour les déficients visuels. La gamme de couleurs peut aussi être modifiée, le codage ne faisant varier que la teinte dans le système teinte/saturation/valeur.

4 Observer l'activité des élèves

Lors de l'utilisation de l'environnement, on enregistre l'ensemble des actions effectuées par l'utilisateur, à savoir les commandes de déplacement et d'édition du programme, ainsi que les exécutions et leurs résultats respectifs.

4.1 Un langage de traces, exécutable

Ce langage, initialement défini pour permettre la commande vocale, permet en fait de reconstituer *a posteriori* une partie importante de l'activité de l'élève. On dispose en effet du scénario complet de ses essais-erreurs et de toutes les versions successives de sa production. Cette trace peut être rejouée dans l'environnement lui-même pour revoir la session. On notera en particulier que cette trace est particulièrement économe en ressources par rapport à une captation vidéo.

Cependant, si on souhaite analyser aussi les interactions langagières entre élèves, il faudra alors synchroniser la trace de la session avec un enregistrement audio réalisé de manière externe.

4.2 Mesures et recherche de motifs

L'ambition du projet est de permettre des recueils de données, pour analyser la construction de notions informatiques chez les élèves. On cherchera en particulier, dans les situations évoquées ici, à mesurer à quel « moment » apparaît à l'élève la nécessité d'utiliser une boucle.

Une des pistes est de tenter de définir des métriques, à partir des activités qui seront observées, pour mesurer, par exemple, le « taux d'essai-erreur » – qui reste à définir – à partir de ces traces contenant les moments précis où des tentatives d'exécution ont été déclenchées.

On cherchera aussi à repérer des motifs dans les activités, pour identifier si l'élève a plutôt utilisé une démarche descendante – ajout de boucles puis remplissage de leur corps – ou plutôt ascendante, ou si sa démarche semble aléatoire ou guidée par la volonté d'essayer toutes les combinaisons possibles, etc.

5 Conclusions provisoires et perspectives

On a décrit un travail en cours qui actuellement en est au stade du premier prototype opérationnel. Le travail méthodologique de recueil et d'analyse de données est maintenant à construire. En parallèle, l'accessibilité du prototype doit maintenant être confrontée à des utilisateurs réels en situation de handicap ou non.

C'est alors que l'on pourra mesurer son utilité et son acceptabilité pour le développement de compétences en programmation chez les élèves en situation de handicap. On pourra alors le comparer aux approches qui ont déjà été publiées dans ce contexte : l'intérêt de la robotique pédagogique pour des élèves déficients moteurs a été démontré par Greff (2016). Boissel (2017) a conçu une mallette pédagogique à base d'objets tangibles gravés en braille, pour permettre aux élèves déficients visuels d'apprendre à programmer en Scratch. La diversité de ces deux approches et de la nôtre incite à rechercher des complémentarités entre instruments dans un objectif partagé de permettre, de manière inclusive, l'apprentissage de la programmation pour tou-te-s les élèves.

Références

Boissel, S. (2017). Mallette Accessi DV Scratch « Scratch débranché en braille et gros caractères ». *La nouvelle revue de l'adaptation et de la scolarisation*, *77,(1)*, 183–192.

Drot-Delange, B., & Tort, F. (2018). Concours castor, ressource pédagogique pour l'enseignement de l'informatique ? Étude exploratoire auprès d'enseignants. In *Didapro 7 – DidaSTIC*.

Greff, E. (2016). Le robot blue-bot et le renouveau de la robotique pédagogique. *La nouvelle revue de l'adaptation et de la scolarisation*, *75,(3)*, 319–335.

Raffinat, P. (2017). Programmation visuelle et handicap : quelques pistes. *Revue MathémaTICE*, *54*.