Initier à la programmation des étudiants de master de sciences de l'éducation ? Un compte rendu d'expérience

Georges-Louis Baron, Emmanuelle Voulgre

Initier à la programmation des étudiants de master de sciences de l'éducation ? Un compte rendu d'expérience

Georges-Louis Baron & Emmanuelle Voulgre
georges-louis.baron@parisdescartes.fr, emmanuelle.voulgre@parisdescartes.fr

Université Paris Descartes, laboratoire EDA

Résumé. Le texte qui suit relate une expérimentation d'initiation à la programmation menée auprès d'étudiants de master de sciences de l'éducation dans le cadre d'une pédagogie de projets. Un corpus de 63 réalisations en *SCRATCH* a été obtenu pendant une durée de 3 ans. Son analyse montre que ces étudiants ont été capables de concevoir des programmes faisant interagir plusieurs entités à l'écran et interagissant aussi avec l'usager, mettant en œuvre des structures de contrôle parfois élaborées. La plupart des groupes ont adopté, pour répondre à leur cahier des charges, des solutions efficaces du point de vue du résultat final, mais peu sophistiquées du point de vue informatique, ce qui est tout à fait logique étant donné le caractère limité de l'initiation. Des prolongements et même des applications en classe apparaissent possibles, mais à condition qu'une formation d'une durée suffisante soit organisée.

Mots-clés: *SCRATCH*, Programmation, enseignement supérieur

Contexte et éléments de problématique

Dans la sphère éducative, ce qu'on entend par *TIC* (ou, pour reprendre le mot maintenant à la mode *numérique*) peut être vu comme un tétraèdre. L'une de ses faces est liée aux outils informatiques traditionnels et aux enjeux sociaux de l'informatique. Elle fait plutôt l'objet d'une prise en compte en milieu scolaire dans le cadre du Brevet informatique et internet (B2i) et, à l'université, dans celui du certificat informatique et internet (C2i), que les étudiants français sont censés avoir acquis en fin de licence.

Une autre face est liée à la technologie éducative, en particulier dans sa composante conception pédagogique. La troisième est davantage didactique et correspond à ce qu'il est important de savoir de l'utilisation d'instruments spécialisés dans différentes disciplines (par exemple s'agissant de logiciels de géométrie dynamique en mathématiques, de logiciels de cartographie en géographie, etc.). Enfin, la quatrième face est liée à l'apprentissage de l'informatique elle-même dans sa dimension science du traitement de l'information, qui s'organise autour de l'algorithmique et de la programmation. Cet aspect, un temps, support d'espoir pour l'enseignement primaire, est devenu depuis les années 1990 *persona non grata* au sein de l'école dans un discours bien articulé de l'institution scolaire insistant sur l'aspect « outil » du numérique et considérant comme sans objet tout ce qui pourrait relever d'une matière *informatique* à l'école primaire (Baron & Bruillard, 2011). C'est à ce dernier aspect que s'intéresse cette contribution qui se focalise sur l'utilisation possible du système *SCRATCH* pour former des étudiants en sciences de l'éducation.

Culture informatique et programmation

Elle est bien loin l'année 1970, ce moment pivot de l'entrée de l'informatique dans le champ éducatif, qui a vu la coïncidence d'un certain nombre de phénomènes d'audience sans doute relativement discrète dans le grand public. On y relève en particulier :

- La tenue à Sèvres d'un colloque organisé par le CERI-OCDE et intitulé « L'enseignement de l'informatique à l'école secondaire » (CERI-OCDE, 1971).
- Le déroulement à Amsterdam de la première conférence mondiale *Ordinateurs et éducation* (WCCE). C'est à ce dernier colloque que pour la première fois est énoncée une alternative qui va avoir une grande fortune pendant une vingtaine d'années : il faut que les enfants apprennent à programmer, plutôt qu'ils se fassent programmer par l'ordinateur (Papert, 1970).

En France, l'intérêt officiel, à ce moment éloigné, portait sur l'informatique comme *démarche*, plutôt que sur la programmation ou sur l'enseignement assisté par ordinateur. Évidemment, à cette époque les ordinateurs étaient des raretés, réservées à des contextes professionnels ou universitaires. Dans l'enseignement scolaire, seuls quelques lycées techniques en étaient équipés avant 1972 (Baron, 1990). Cette focalisation persistera assez longtemps : les premières expériences d'équipement de lycées en mini-ordinateurs, dans la décennie 1970, ne concernent que les lycées, tout comme les premiers plans d'équipement en micro-ordinateurs, en 1979 (58 lycées, 10000 micros). Mais l'intérêt pour l'informatique dans l'enseignement primaire croît. Dès le début des années 1980, la pensée de Papert se répand. À l'école primaire, LOGO, porté par le succès de son ouvrage *Mindstorms : children, computers, and powerful ideas* (Papert, 1980), traduit en français dès 1981 (*Jaillissement de l'esprit : ordinateurs et apprentissages*) devient une approche estimée, d'autant plus qu'elle n' a encore été confrontée à la réalité que dans le cadre d'expérimentations (Robert, 1985).

Le plan informatique pour tous, en 1985 a fourni un équipement à l'ensemble des écoles primaires. Il promeut LOGO, mais l'engouement institutionnel ne dure guère : l'ère de *l'outil informatique* semble venue et on s'intéresse alors davantage au ministère à ce que l'on appelle les « facilitateurs d'écrits ». Les vertus et jusqu'au souvenir de Logo semblent peu à peu s'estomper, du moins en France.

La situation a changé à nouveau dans les années 2000, avec l'introduction d'un brevet informatique et internet (B2i), listant des compétences à acquérir sans pour autant définir de curriculum. Parmi ces compétences, algorithmique et programmation ne figurent pas. Ce n'est que récemment que l'intérêt pour cet aspect se développe.

Au cours du temps, les environnements de programmation ont beaucoup évolué. Une avancée significative concerne ceux qui sont destinés à être mis en œuvre par des enfants. En particulier, dans les années 2000 s'est diffusé une sorte de descendant spirituel de Logo : *SCRATCH*. Ce langage, élaboré au sein du Massachusetts *Institute of Technology,* est fondé sur le langage Smalltalk. Il repose sur une interface à la LEGO, le groupe qui l'a développé ayant aussi participé à la création de Logo-Lego : on assemble à l'écran des *briques de programmation* de différents types (Resnick et al., 2009).

On peut télécharger librement le logiciel, ainsi qu'une documentation élaborée par une communauté et une très large bibliothèque de programmes qu'il est possible de reprendre et de modifier. On peut aussi mettre en ligne ses propres productions.

Quelques spécificités du système SCRATCH

Ce langage orienté objets possède un certain nombre de caractéristiques qui le rendent très intéressant pour des non scientifiques. Le système est conçu pour permettre les essais et erreurs : il ne suppose pas qu'on ait d'abord écrit un algorithme puis qu'on le code. Au contraire, on peut littéralement jouer avec des situations, modifier des paramètres à la volée, bricoler en somme. Cette approche favorisant le tâtonnement expérimental, comme disait C. Freinet, est adaptée à de jeunes enfants (Wilson & Moffat, 2010). Il existe une grande bibliothèque d'objets qu'on peut facilement doter de différents attributs : des sons (des fichiers audio qu'on peut invoquer par leur nom), des apparences (des costumes) et, bien entendu, des scripts.

À titre d'exemple, l'illustration suivante montre un script très simple issu d'un programme réalisé par des étudiantes de master de sciences de l'éducation mettant en scène la comptine « une souris verte », bien connue de tous les jeunes français[1] . Le script est attaché à un objet (le bol) réagissant au message *bol*. A la réception de ce message, il apparaît avec le costume 1 (un bol rempli d'huile) puis 4 secondes plus tard avec le costume 2 (un bol rempli d'eau). Après 1,5 secondes, un message est envoyé à tous, qui ce qui entraînera l'apparition d'un escargot qui attend ce message pour se montrer).

[1]« Une souris verte qui courait dans l'herbe ; je l'attrape par la queue, je la montre à ces messieurs, ces messieurs me disent : trempez la dans l'huile, trempez là dans l'eau, cela vous fera un escargot tout chaud ».

Illustration 1: Script extrait d'un projet

Il est donc assez facile de créer des animations interactives. Bien entendu, il est aussi possible de programmer des algorithmes mathématiques et la plupart des livres contemporains de mathématiques en lycée comprennent des exercices de programmation en *SCRATCH*[2].

Question de recherche

Depuis les années 1980 et 1990, le premier auteur de cette contribution s'est intéressé à l'enseignement de l'informatique pour des publics non scientifiques. L'expérience lui a montré qu'il était possible de familiariser ces publics à des notions comme la structuration de textes, la programmation de formules à l'aide d'un tableur et, dans une certaine mesure, à la programmation dans un langage impératif. L'arrivée de systèmes comme SCRATCH lui a semblé renouveler la question. Il se trouve qu'il a actuellement la charge de dispenser des enseignements de master liés aux technologies de l'information et de la communication (TIC) à des étudiant-e-s de sciences de l'éducation. Il a donc décidé de mener une étude exploratoire pour tester dans quelle mesure un tel outil pouvait être utilisé en sciences de l'éducation, au-delà des formations de type C2i.

Il s'agit d'une population intéressée par l'éducation et, pour une part, se destinant à l'enseignement primaire. Largement féminine, elle est en majorité de formation littéraire, peu au fait de l'informatique et entretient souvent des rapports compliqués voire douloureux avec les mathématiques. Les personnes suivant les cours sont néanmoins en général intéressées par les technologies de l'information et de la communication et leurs usages éducatifs et certaines souhaitent y acquérir des connaissances.

La principale question était de savoir comment les étudiant-e-s pourraient se familiariser avec la programmation et en tirer des connaissances transférables ailleurs. L'hypothèse était qu'il serait ainsi possible de leur transmettre un certain nombre de concepts liés à l'informatique.

Méthodologie

Cadre de l'intervention

L'action s'est déroulée pendant 3 ans sur le même modèle, au sein d'un cours de premier semestre de master 1 portant sur l'informatique en éducation.

Une première séance de deux heures a été consacrée à un exposé historique de ce qu'est un langage de programmation, avec une présentation de la notion d'algorithme. Puis SCRATCH et ses principes ont été brièvement présentés et un exemple a été expliqué (la programmation d'une sorte de jeu de PacMan). Cet exemple ne comportait pas de variables mais faisait intervenir l'envoi de messages. Il comportait des instructions de mouvement et une boucle générale infinie avec des tests permettant de déterminer si les parois étaient atteintes, la cible touchée… Des liens vers des ressources ont aussi été fournis aux étudiants, avec pour consigne de s'organiser en groupes de 2 ou 3, d'écrire un programme à leur convenance et de l'accompagner d'un texte de documentation explicitant les objectifs du projet et discutant des choix faits et des difficultés rencontrées. Cette activité a été présentée comme

[2]Par exemple *MathX classe de seconde,* Didier 2009 ou *Maths-repères* classe de seconde, Hachette 2009.

intervenant dans le contrôle continu, le critère de réussite étant de parvenir à faire un programme simple et documenté. Le programme pouvait tourner de manière imparfaite, mais alors les problèmes rencontrés devaient être discutés. La semaine suivante a comporté un moment de mise en commun et une intervention de recadrage discutant des problèmes rencontrés et donnant d'autres exemples. Les séances suivantes ont porté sur des sujets différents. Une présentation/discussion des programmes a été organisée après quelques semaines.

L'enseignant n'est pas entré dans une procédure classique de spécification puis de programmation d'un algorithme, mais il a incité les étudiant-e-s à se fixer un objectif puis à tâtonner pour l'atteindre en prenant connaissance des fonctionnalités du système. Son intention pédagogique n'était pas de développer une quelconque maîtrise de la programmation, mais simplement de conduire les étudiants à créer un programme simple et à réfléchir sur ce qu'est la programmation. Il a été conjecturé au début que cette approche susciterait de l'intérêt, que les étudiants prendraient d'abord assez facilement en main le système, rencontreraient rapidement des difficultés nombreuses, en particulier pour synchroniser leurs scripts, mais que la plupart parviendraient à les surmonter et, finalement, pourraient acquérir une première expérience de la programmation.

Un accompagnement a été offert aux équipes qui le souhaitaient et des plages de temps ont été réservées à la discussion des difficultés rencontrées. Ainsi à partir des questions posées et des réponses apportées, certains groupes ont réalisé des programmes très élaborés. Dans la suite du cours, l'expérience acquise a été utilisée à plusieurs reprises pour faire réfléchir au fait que toute interaction avec un ordinateur était sous le contrôle d'un logiciel et pour faire pratiquer des formes élémentaires de programmation par mise à jour de variables d'environnement.

Modalités d'analyse des productions étudiantes

Pour analyser les programmes élaborés par les étudiant-e-s, nous les avons tout d'abord repris un par un. Nous avons évalué leur complexité en prenant en compte différents éléments dans la structure des programmes :

- S'agissant des objets : leur nombre, les costumes définis, les sons associés, le nombre de scripts qui leur sont attachés.
- En ce qui concerne les scripts attachés aux différents objets : la présence dans chacun d'instructions de mouvements, de changement d'apparence, d'émission de messages et d'entrées d'informations (les inputs), les structures de contrôle (les boucles et instructions conditionnelles, l'utilisation de messages), la mise en œuvre de capteurs, la présence de variables et d'opérateurs entre ces dernières.
- Finalement nous avons aussi estimé le degré d'interactivité des programmes produits (certains réalisent de simples projections de films, d'autres sont hautement interactifs) et noté dans quelle mesure ils fonctionnaient comme souhaité.

Nous avons ensuite, procédé à une description analytique de chacun dans une série de tableaux, afin d'identifier la thématique et les auteurs, le niveau de complexité, la structure du projet et sa composition en fonction du nombre et de l'agencement des différentes briques programmes utilisés. De manière synthétique, les programmes réalisés ont été classés en 3 grandes catégories en fonction de leur complexité.

- 1 : programmes courts avec différents objets et scripts, mais pas ou peu d'interactivité avec l'usager (au maximum une demande du genre « voulez vous écouter la suite ? ») aiguillant vers des segments différents selon la réponse ; pas de communication entre objets.
- 2 : programmes simples mettant en outre en œuvre de la communication avec le joueur.
- 3 : programmes plus complexes organisant une communication fine entre objets (par envoi de messages) et avec l'usager pour synchroniser des processus

S'agissant des écrits des étudiant-e-s, nous avons procédé à une première lecture pour identifier les thématiques dominantes puis à un tri des éléments relatifs à ces dominantes dans un tableau de quatre colonnes triant ainsi les projets, les émotions, les buts pédagogiques et les difficultés exprimés dans les textes.

Il importe de bien rappeler que l'échantillon considéré est assez modeste et que les effectifs varient grandement d'une année sur l'autre. L'intervention a également un peu varié : l'enseignant a essayé de procéder de manière analogue chaque année mais il a forcément tenu compte de l'expérience des années précédentes, de l'effectif de la classe (entre une quinzaine et plus de soixante) et des compétences des étudiants. Les résultats qui suivent ont un caractère purement indicatif. Certains de ces derniers peuvent cependant servir d'hypothèses à confirmer auprès de publics plus étendus.

Résultats

Description du corpus

L'activité faisait partie du contrôle continu et tous les étudiants assidus en cours l'ont donc accomplie. Au total, 63 projets ont été analysés : 35 en 2010, 20 en 2011 et 8 en 2012 ; les différences d'effectifs sont liées au public du cours et à la taille des équipes. La majorité de ces dernières ont comporté 2 personnes, occasionnellement 3. Un petit nombre de projets ont été réalisés par une personne seule.

Ce qui frappe d'emblée, c'est la diversité des productions et leur très large spectre. Certains programmes sont très peu évolués du point de vue informatique et répondent a minima à la commande : avoir écrit un programme, décrit son fonctionnement attendu, l'organisation du travail pour sa réalisation, et analysé les dysfonctionnements éventuels.

Mais d'autres groupes se sont littéralement pris au jeu et ont passé dans la réalisation du programme un temps très important. On a parmi les productions, des réalisations très soignées, mettant en scène un scénario parfois amusant, mais du point de vue informatique purement linéaires et sans interactivité avec l'usager. Un petit nombre d'autres, enfin, sortent du lot : elles sont généralement le fait de groupes comprenant une personne ayant déjà fait un peu de programmation. Ce sont à peu près les seules qui utilisent de manière un peu sophistiquée les variables.

L'analyse révèle une complexification croissante au cours du temps. En 2010, 30 des 35 projets sont constitués de moins de 6 scripts, ce qui est finalement assez peu. En 2011, la majorité des projets en ont au moins 6, et en 2012, 5 projets sur 8 en comptent au moins 18. De manière similaire, s'agissant de sons, en 2010 et 2011 la majorité des projets en comptaient moins de 4 alors que tous les projets de 2012 en ont au moins 4. Notons que les sons peuvent être des enregistrements disponibles dans la bibliothèque, importés ou des productions originales enregistrées par les étudiants eux-mêmes. Si 21 programmes ont été classés de niveau 1 en 2010, c'est le cas de la moitié d'entre eux la seconde année et d'un seul en 2012.

Description d'éléments revenant régulièrement dans les projets

Plusieurs briques de programmation sont utilisées quasiment dans tous les projets en plus de la brique « quand drapeau vert pressé », qui sert à faire démarrer le programme.

a) Mouvements

Les plus fréquentes sont les briques de mouvement : La majorité des projets en ont 2 à 15. 3 seulement en ont plus de 40. Ces briques permettent de faire bouger les objets à l'écran en fonction de nombres à indiquer dans des espaces blancs.

L'image suivante illustre la programmation d'un déplacement complexe obtenu à la fois par déplacement entre des points de coordonnées donnés et par des changements d'apparences (obtenus par le basculement sur un autre costume). Le réglage du mouvement a dû donner lieu à un assez long travail de réflexion et de tâtonnement.

Illustration 2: Exemple de portion de script gérant un déplacement complexe.

b) Briques de contrôle et tests

S'agissant de briques de contrôle, le langage en offre une grande variété : des déclencheurs conditionnels (« *quand touche X pressée* », « *quand objet Y pressé* »…), des tests (du type *« si… alors… sinon »*, « *quand condition »…)*, de la gestion de message (« *envoyer message, quand je reçois message x »), des* briques *de boucle (« répéter n fois », « répéter indéfiniment si », « répéter jusqu'à »…).* C'est également une brique contrôle « stop » qui permet d'arrêter le programme, ou de faire attendre une partie du programme avant de poursuivre l'action.

Parmi les plus utilisées, il y a « attendre x seconde », qui modère la chronologie des actions en permettant des pauses dans l'exécution du script. On trouve aussi des boucles : « répéter x fois », « répéter indéfiniment » (avec un test de sortie de boucle). Plus rarement, on a observé l'utilisation de boucles « répéter jusqu'à ».

Un problème souvent rencontré est celui de la synchronisation d'actions. Sous sa forme la plus simple, celle-ci est réalisée en affectant des durées fixes à différentes actions. Par exemple, « Dire quelque chose pendant x secondes » associé à « attendre x secondes » permet de synchroniser des objets. Pendant que l'un parle, l'autre attend. On est ainsi dans une logique de construction de scénario temporisé.

Dans environ 4 scripts sur 10 (4 en 2010, 8 en 2011 et 6 en 2012), les étudiant-e-s ont utilisé la technique, plus efficace, de l'envoi de messages à tous quand un processus est terminé.

c) Variables

Les variables sont des briques qui s'insèrent dans une autre pour préciser ce qui est à prendre en compte par le programme. Par exemple, le programme devra s'attacher à la longueur d'un texte écrit, à un mot contenu dans un texte, à une variable cible. Seuls 8 projets utilisent ce domaine.

Exemples de programmes

À titre d'exemple, nous présentons ici un certain nombre de programmes effectués en 2010, 2011 et 2012, qui nous semblent significatifs de ce que peuvent produire les étudiant-e-s. La plupart correspondent à des jeux, mais il y a aussi quelques narrations.

a) 34-2010-Tour de Chine-3

Il s'agit d'un projet très soigné visant à présenter la Chine. D'une taille importante à cause des sons et des images (plus de 8 Mo), c'est une narration comprenant une dizaine d'objets chacun doté de leurs propres scripts et faisant intervenir 9 arrière-plans différents minutieusement agencés. Les structures de contrôle sont assez rudimentaires (des répéter n fois), mais efficaces : l'usage des messages est bien maîtrisé, ce qui permet de synchroniser fonctionnellement l'arrivée et le comportement des objets. L'ensemble témoigne d'un investissement très fort produisant un résultat remarquable, sans néanmoins embarquer une programmation très sophistiquée.

b) 2-2012-Under the sea-2

Un crabe et une étoile de mer veulent manger une assiette de frites. Le joueur doit conduire l'étoile à l'assiette sans que le crabe ne touche l'étoile. Le jeu s'arrête soit parce que le joueur gagne (l'étoile touche l'assiette de frites), soit parce que le joueur perd (le crabe touche l'étoile). La complexité du programme vient d'une utilisation relativement maîtrisée de l'envoi de message pour synchroniser les actions et de la mise en œuvre de différents types de briques de contrôle et notamment un « *répéter jusqu'à* ».

c) 4-2012-Rond Carré jeu-3

Il s'agit de faire déplacer un carré rouge dans des espaces successifs sans toucher des ronds bleus qui les traversent. La position des ronds est déterminée par deux briques mouvements. Si le rond monte et descend (verticalement = parallèle à l'axe des ordonnées), x est identique, donc fixe, dans les deux mouvements et y est une variable glissante ; si le mouvement va de droite à gauche et de gauche à droite, (horizontalement = parallèle à l'axe des abscisses), x est la variable glissante et y est la variable fixe. La brique « glisser en 1 seconde à x : -75 y -91 » associée à la brique « glisser en 1 seconde à x : -75 y : 33 » permet de délimiter le segment sur lequel glisse le rond bleu. Notons l'indicateur de vitesse (ici, 1 seconde) qui permet de moduler la difficulté du jeu en faisant déplacer les obstacles plus ou moins vite. Dans ce projet un arrière-plan donne le mode d'emploi pour les joueurs.

d) 7-2012-Chat objets en anglais-3

Un chat (objet1) donne une leçon d'anglais au joueur. Ce dernier doit cliquer de façon successive, sur des images présentes sur l'interface en fonction des instructions du chat (*click on the banana, on the laptop*, etc), qui s'accompagnent de l'instanciation d'une variable globale (*cible*). Chacun des objets comprend un script vérifiant, quand il est pressé, la valeur de cette variable et répondant en conséquence. Si la réponse est fausse, un message du type *Oups, I'm not the banana* apparaît et le joueur doit cliquer à nouveau sur un autre objet. Quand la réponse est juste, le script renvoie un feedback positif et donne une instruction au joueur *press the A to continue…*, ce qui déclenche une nouvelle demande et la mise à jour de la variable cible.

Ce programme est assez sophistiqué. Il comporte cependant un défaut d'analyse : si on appuie sur la barre d'espace *avant* d'y avoir été invité, à la fin, cela déclenche la séquence finale *very good, you are a champion*, car le script correspondant, attaché au chat, ne vérifie pas si les interactions précédentes ont eu lieu et si tous les objets ont bien été cliqués à bon escient. Ce défaut, qui n'a pu être repéré par ses auteures faute de temps, ne remet pas en cause le caractère remarquable de la production.

Opinions exprimées par les étudiant-e-s

a) Émotions diverses

Dans les textes d'accompagnement, nous avons identifié diverses émotions. L'idée de plaisir est particulièrement attachée à la découverte des fonctionnalités du logiciel, puis à la compréhension des modalités de programmation d'un cas ayant posé un problème. À titre d'exemples nous avons relevé ces citations « j'ai trouvé intéressant d'avoir cette attitude de découvreur, de jouer avec les combinaisons, de me confronter à un outil nouveau » et « Chaque difficulté franchie était une émulation à la curiosité intellectuelle pour aller plus loin dans l'élaboration de cette création de script. ». Un étudiant exprime le fait qu'il a découvert « un logiciel de programmation bien plus convivial que celui découvert en 1988 en classe de CM1 sur des MO5 ou TO7 ».

D'autres émotions sont relatives à la pédagogie. On y trouve des opinions positives, comme « Ce projet a été très intéressant et est une manière de nous évaluer de façon moins magistrale » ou semi-positives « très bon support illustratif et intéressant pour un cours traditionnel magistral en classe et un peu ennuyeux ». La notion du ludique « nous avons appris en jouant » apparaît aussi. La fierté d'avoir achevé un projet est parfois nuancée « nous sommes assez fières du résultat », « nous sommes fiers de notre toute petite programmation ».

Sur l'utilité d'un point de vue professionnel, les avis sont plus divers. On trouve ainsi : « Je ne vois pas forcément le lien entre la programmation et ma formation de future Conseiller Principal d'Éducation » ou « C.P.E, je ne pense pas pouvoir l'intégrer dans mon travail ». Pour d'autres, le lien avec la pratique professionnelle fonctionne : « projet en adéquation avec mon travail (enseignante de la langue et

culture portugaise) » ; « Grâce à ce mini-projet je peux mieux faire mon cours avec les enfants. En particulier avec ceux qui n'ont pas beaucoup d'intérêt pour le chinois. » *SCRATCH* est majoritairement jugé intéressant pour les enseignants et pour des projets éducatifs spécifiques.

b) Difficultés rencontrées

Une majorité de répondants ont dit que l'engagement dans le projet leur a pris beaucoup de temps, davantage que ce qu'ils avaient pensé en choisissant le cours. Les plus grandes difficultés sont liées à l'idée du projet et à sa mise en œuvre, c'est-à-dire à l'assemblage des briques de programmes à utiliser pour parvenir à l'action souhaitée. Les solutions sont apportées dans une majorité des cas par une démarche de tâtonnement. Rares sont les groupes qui ont étudié le guide d'utilisation de *SCRATCH* avant de commencer.

Quelques fois, les solutions aux problèmes ont pu être trouvées par des échanges construits entre différents groupes par observations des programmes des uns puis adaptations pour le projet de l'autre. D'autre fois, des aides familiales ont été sollicitées (neveu, frère). Enfin des recherches sur Internet ont permis d'ouvrir des pistes de solutions. Parmi les problèmes le plus souvent relevés, nous en avons identifié 3 grands types :

- Conception de l'interaction entre personnages ;
- Mise en mouvement des objets (les faire démarrer à un certain endroit et à un certain moment ; orienter l'objet en mouvement, synchroniser des déplacements, ne pas faire disparaître l'objet du cadre…) ;
- Gérer l'initialisation du script et son achèvement (rétablir l'état des personnages du départ).

De façon plus globale, des difficultés de compréhension sont apparues quant au repérage dans l'espace, à la manipulation des blocs de contrôle et à la synchronisation.

c) Idées de projets futurs

Des idées de projets sont formulées concernant l'action des enseignants afin de rendre plus attractive une notion et l'illustrer comme par exemple « les activités de gymnastiques pour enseigner aux élèves », « illustrer un album que je lis à mes élèves ».

On note également un projet comme « sensibiliser des enfants de maternelle à parler des images proposées par le programme », « développer leur imaginaire et parler », « combiner son imaginaire, ses propres mots et le langage simple et expérimental de la programmation » ou encore pour des activités musicales, d'écritures, de bio-citoyenneté, et de poésie.

Des projets autour d'un travail collaboratif sont évoqués pour « échanger et pratiquer pour assimiler un nouveau savoir. », « apprendre à organiser les données », « apprendre la division du travail ». Cette collaboration peut aussi être avec l'enseignant pour « imaginer avec eux la possibilité de transposer et faire évoluer un scénario »

À partir des récits des étudiants, nous pouvons conclure que les projets d'utilisation de *SCRATCH* dans un cadre scolaire devraient posséder certaines qualités :

- Capter l'attention des élèves et tirer leur curiosité à partir de photos, dessins animés, sons, atmosphères adéquates à l'histoire et un travail sur l'esthétique ;
- Créer l'impression d'être dans un jeu tout en ayant des objectifs pédagogiques ;
- Créer des liens avec des événements spécifiques (Halloween, Noël, Nouvel an, etc.) ;
- Permettre la compréhension au travers de tâtonnements essais et erreurs.

Enfin, nombreux sont les étudiants qui expriment l'idée que dès l'école primaire, *SCRATCH* permet d'aborder « les premières formes de la programmation ». La découverte de notions informatiques telles que les boucles, les tests, permettent aux élèves de découvrir « qu'ils peuvent être acteur actif avec un ordinateur » ou encore, aussi associé aux notions de mathématiques, découvrent que les algorithmes permettent de « proposer un entraînement » notamment aux opérations de bases (soustraction, addition, multiplication, division).

Discussion et perspectives

Nous pensons avoir confirmé qu'il était possible, dans des conditions suffisamment favorables (le cadre d'une pédagogie de projets), de faire produire des programmes relativement intéressants par des personnes non formées auparavant en informatique. *SCRATCH* s'est révélé être un support adapté pour cela, permettant de faire comprendre ce qu'est un programme et de réaliser un projet mettant en œuvre des structures de contrôle parfois sophistiquées. Sa structure à partir d'assemblage de briques ne pose pas de problème majeur à des novices et il est facile d'y inclure des animations et des sons.

En revanche, il est indéniable que des problèmes se révèlent très vite au-delà d'un certain niveau de complexité. Une tendance forte a été d'utiliser le langage pour animer des narrations fonctionnant sur un principe de minutage. Ainsi, un des groupes a, l'année suivante, réalisé selon ce principe un projet de C2i2e qui a la forme d'un film court (12,2 Mo) amenant à réfléchir sur les dangers des boissons alcoolisées.

En fait, si l'utilisation de messages pour synchroniser des processus ne pose pas de problème particulier, la mise en œuvre de variables est plus délicate, peut-être parce que leur mise en œuvre suppose la manipulation de plusieurs types de blocs. L'utilisation de la brique « demander *Texte* et attendre », qui place automatiquement l'entrée de l'usager dans la variable-brique *reponse*, ne pose en effet pas de difficulté.

Un point relativement encourageant est que les étudiant-e-s se destinant à l'enseignement primaire ont eu des idées de ce qui pourrait être fait avec un tel système. Il resterait à étudier comment sa mise en œuvre pourrait être réalisée en classe. Pour l'instant, c'est difficile : la programmation ne rentre pas dans le cadre des activités préconisées officiellement. D'ailleurs, il serait nécessaire de former les enseignants auparavant.

Nous avons mené, en 2012, une étude exploratoire avec des élèves d'une municipalité de la banlieue parisienne, dans un cadre un peu différent : celui de séances hors du temps scolaire. Nous avons intéressé la municipalité, formé les animateurs et ensuite mené une courte expérimentation pendant les vacances d'été, suivie par une des étudiantes de M1 qui avait fait la preuve d'une appropriation suffisante du système. Les conclusions sont à prendre avec précaution : les enfants ont produit des programmes simples, décalquant souvent ceux qui leur avaient été montrés dans une première phase. Mais l'expérimentation ne s'est pas poursuivie après le départ de l'étudiante.

Pour avoir des résultats plus solides, il faudrait pouvoir organiser une expérimentation contrôlée, éventuellement sous la forme d'un jeu-concours, comme cela se passe parfois dans certains endroits. Il serait en particulier indispensable d'avoir préalablement formé les enseignants pendant une durée suffisante, toute la difficulté étant de savoir ce qui serait suffisant. Cela n'apparaît pas hors de portée, mais reste pour l'instant en perspective.

Références

Baron, G.-L. (1990). L'informatique en éducation, le cas de la France. *Revue Française de Pédagogie, 92*, 57 – 78.

Baron, G.-L., & Bruillard, E. (2011). L'informatique et son enseignement dans l'enseignement secondaire général français. Enjeux de pouvoir et de savoirs. In J. Lebeaume, A. Hasni, & I. Harlé (Eds.), *Recherches et expertises pour l'enseignement scientifique* (De Boeck., pp. 79–90). Bruxelles. http://www.stef.ens-cachan.fr/annur/bruillard/2010_B&B_DeBoeck.pdf.

CERI-OCDE. (1971). *L'enseignement de l'informatique à l'école secondaire*. Paris: OCDE.

Papert, S. (1970). Teaching children thinking. In B. Scheepmaker (Ed.), *Proceedings of the IFIP World Conference on Computer Education* (Vol. 1, invited papers, pp. 61–66). Amsterdam: IFIP.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Basic Looks, Inc. http://portal.acm.org/citation.cfm?id=SERIES11430.1095592.

Papert, S. (1981). *Jaillissement de l'esprit : ordinateurs et apprentissage*. Paris: Flammarion.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Silverman, B. (2009). *SCRATCH*: programming for all. *Communications of the ACM, 52*(11), 60–67. http://dl.acm.org/citation.cfm?id=1592761.1592779.

Robert, F. (1985). L'utilisation de l'ordinateur dans l'enseignement primaire : l'exemple de la France. *Enfance*, *38*(1), 19–30. doi:10.3406/enfan.1985.2857

Wilson, A., & Moffat, D. C. (2010). Evaluating *SCRATCH* to introduce younger schoolchildren to programming. http://scratched.media.mit.edu/sites/default/files/wilson-moffat-ppig2010-final.pdf.