Une analyse cognitive et didactique du langage de programmation ScratchJr

Vassilis Komis

Équipe ICTE, Laboratoire de Didactique des Sciences, Département de l'Éducation, Université de Patras, Grèce

Résumé

Peut – on envisager l'apprentissage de la programmation et plus généralement le développement de la pensée informatique dans le cadre de l'école maternelle ? Y-a-t-il des environnements de programmation qui sont adaptés à cette approche ? Dans cet article nous essayons de donner quelques réponses à ces questions en examinant, à l'aide d'une analyse cognitive et didactique, l'environnement de programmation ScratchJr. Suite de cette analyse préliminaire nous avançons l'hypothèse qu'il faut un travail plus approfondi de scénarisation pédagogique sur les aspects didactiques et cognitifs pour exploiter de manière efficace et appropriée cet environnement.

Contexte

L'apprentissage de la programmation et le développement de la pensée informatique à la petite enfance reviennent les dernières années à l'ordre du jour, grâce à des avancées technologiques de deux ordres. D'un côté, il y a eu l'apparition des tablettes et des environnements de programmation visuelle destinés à des petits enfants et aux renouvellements des approches pédagogiques, d'autre côté, relatives plutôt du courant épistémologique promouvant un enseignement de la science informatique à tous les élèves de l'enseignement général (Wing, 2006), qui remet à l'ordre du jour des approches, connues depuis longtemps, organisés autour du langage Logo (Papert, 1980).

La pensée informatique occupe, depuis une décennie, une place croissante au sein de la recherche en éducation. En reprenant de vieux termes, tels que la pensée algorithmique et l'apprentissage de la programmation en tant qu'une approche cognitive, et en les inscrivant dans un contexte technologique et pédagogique plus favorable aux innovations techno-pédagogiques, ce courant de pensée essaye peu à peu à trouver une place dans les prescriptions des programmes scolaires et les pratiques pédagogiques. Peut-on, dans ce contexte, envisager un développement de la pensée informatique à l'école maternelle ? Ce niveau constitue un niveau scolaire où nous ne disposons pas beaucoup de travaux, à part les travaux effectués il y a longtemps (autour des années 1980), à partir du langage Logo. En fait, c'est le courant pédagogique et technologique de l'environnement Logo qui constitue à nouveau la source d'inspiration pour introduire des environnements de programmation et de robotique aux écoles maternelles (Komis et Misirli, 2011, Misirli et Komis, 2014). Dans ce contexte, on trouve à nouveau un intérêt particulier pour la programmation de type Logo en tant qu'approche pédagogique apte à favoriser le développement d'une certaine pensée informatique, adaptée, bien attendu, au niveau

cognitif et développemental des enfants de cet âge. En grande partie, ces initiatives revêtent plusieurs formes : d'un côté, on trouve des prescriptions dans les programmes scolaires des usages des outils programmables ainsi que des initiatives périscolaires (p.e. une heure de code). D'autre côte, plusieurs outils technologiques innovants ont vu le jour les dernières années, pouvant s'inscrire sous cette perspective : tels sont les environnements informatiques (p.e. ScratchJr), les jouets et les robots programmables (comme les Bee-Bot et les Probot) et les kits de constructions robotiques (comme les Lego MindStorms). Dans ce travail, nous mettons l'accent sur l'environnement de programmation ScratchJr, et à partir d'une analyse cognitive et didactique de son interface, nous examinons son appropriation et sa pertinence pour l'apprentissage de la programmation pendant la petite enfance.

Une brève présentation de l'environnement ScratchJr

Le logiciel ScratchJr (http://www.scratchjr.org/) est une application native sur tablettes, disposant une interface graphique basée sur un langage entièrement visuel et fonctionnant donc par manipulation gestuelle. Selon ses concepteurs, il permet aux enfants de 5 à 7 ans de programmer des jeux interactifs sous une approche de résolution des problèmes, de conception de projets et de développement de la créativité, considérant ainsi le codage en tant qu'une nouvelle forme d'alphabétisation (Resnick et al., 2013).

L'application ScratchJr dispose une interface ergonomique, d'une grande simplicité et transparence, qui semble, à la première vue, être adaptée au public auquel il est destiné. L'interface de l'application revêt des aspects conviviaux et ludiques de manière à motiver les jeunes enfants à programmer avec le logiciel (Portelance et al., 2015). Il s'agit d'un environnement qui, par sa conception, permet de travailler avec des concepts fondamentaux en informatique et en programmation. Dans ce travail, nous n'examinons pas les aspects ergonomiques concernant les différentes manipulations de l'interface mais nous mettons l'accent sur certains concepts de la science informatique que cet environnement logiciel nous semble permettre de mettre en œuvre.

Le principal écran d'utilisateur (figure 1) permet aux enfants d'utiliser un environnement de type « Glisser – Déposer » pour manipuler des icônes de différentes formes et couleurs sur l'espace – écran afin de créer des scripts de programmation. Cet espace comporte trois parties principales :

a) la partie basse pour composer des scripts et des programmes à l'aide d'une palette de commandes (briques colorées) qui comprend six catégories différentes : les blocs jaunes comportent les différents modes de lancement d'un script ou d'un programme, les blocs bleus comportent les commandes des déplacements sur l'écran, les blocs mauves gèrent l'apparence des personnages à programmer, les blocs verts contrôlent les sons, les blocs orange et rouge comportent les différents structures de contrôle. Les icônes sont très ergonomiques (assez larges et d'une apparence suggérant leur fonctionnalité) pour que les enfants puissent les utiliser aisément.

b) La partie centrale qui constitue la scène sur laquelle se déroule l'action des personnages programmés qui se déplacent et interagissent entre eux dans un contexte d'histoire interactive ou de jeux.

c) La partie de gestion de l'environnement (création des personnages et des scènes, sauvegarde, manipulation du projet, etc.)

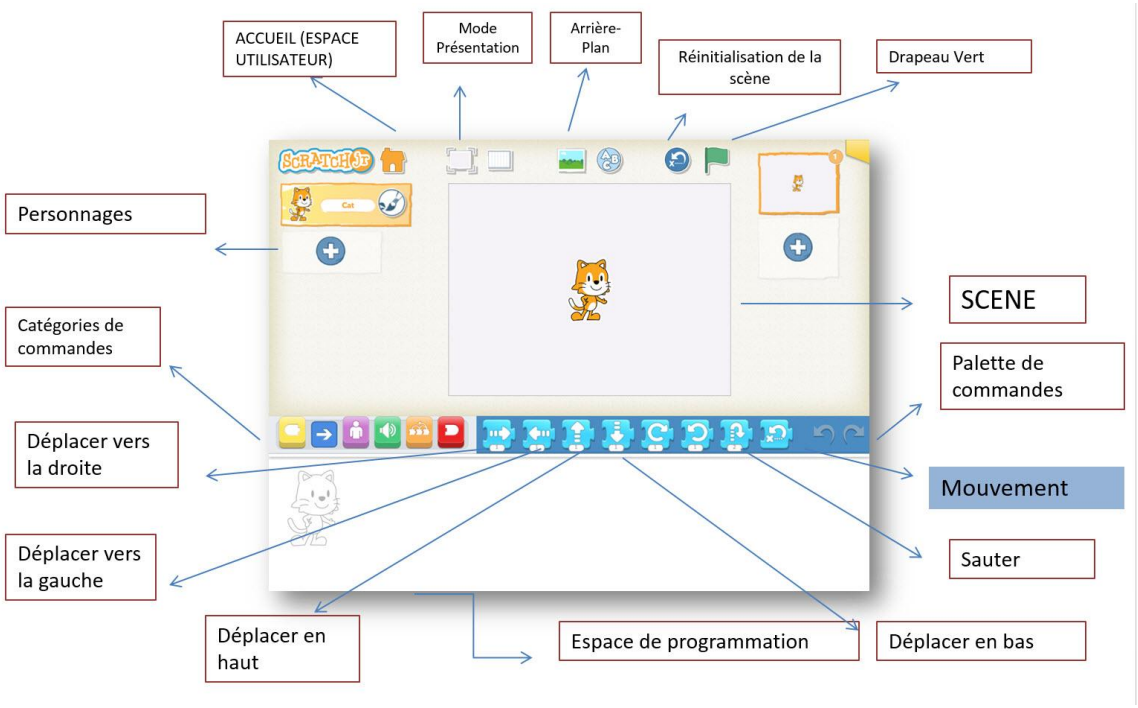

Figure 1 : l'interface de programmation de ScratchJr

Le langage de programmation ScratchJr : aspects syntaxiques et sémantiques

Apprendre à programmer avec l'environnement de ScratchJr consiste en l'apprentissage d'un langage de communication avec la tablette informatique pour qu'elle exécute des tâches précises concernant principalement les actions des personnages qui se déplacent sur la scène et interagissent entre eux. Comme chaque langage de programmation, ScratchJr dispose sa propre syntaxe et d'une sémantique liée à la signification de ses éléments de base. En apprenant un langage de programmation, et c'est également le cas pour ScratchJr, on développe des *connaissancessyntaxiques*, des *connaissancessémantiques* et des *connaissancesstratégiques*, qui sont principalement liées au transfert dans d'autres domaines de connaissances (Fay et Mayer, 1988).

Les connaissances syntaxiques dans ScratchJr

La structure syntaxique (grammaire) du langage ScratchJr est liée à la forme de son interface, qui, comme tous les langages de programmation (symboliques ou visuels), comporte trois composants essentiels (Fay et Mayer, 1988) : les éléments de base (*briques*), les règles de composition de ces éléments en *instructions* (ou *commandes*) et les règles de composition des commandes en *scripts* ou en *programmes*. La syntaxe du

ScratchJr est relativement simple et intuitive, étant donné qu'elle fonctionne sous les principes de « Glisser – Déplacer » et de la programmation visuelle dans laquelle les morceaux de programmes (les scripts) et les programmes sont écrits par assemblage d'éléments graphiques. Sa syntaxe concrète est composée d'icônes graphiques, qui sont disposées spatialement sous forme de puzzle pour former des scripts et des programmes (figure 1).

Les connaissances sémantiques dans ScratchJr

La sémantique d'un langage de programmation est liée à la signification que l'on attache aux différentes constructions syntaxiques effectuées dans son environnement. Dans ce sens, la sémantique est très difficile à définir et détermine si l'environnement de programmation peut supporter des styles de programmation différents, comme la programmation impérative, la programmation logique ou la programmation orienté-objets.

La structure sémantique d'un langage de programmation n'est pas arbitraire mais elle dispose une structure logique qui dépend des composants fonctionnels du domaine de la programmation. Trois composants fonctionnels décrivent la sémantique de commandes (instructions) dans un langage de programmation : a) *Opérations* : les actions qui peuvent être effectuées, b) *Objets* : les entités sur lesquelles sont effectuées les opérations et c) *Emplacements*: l'endroit où se déroule une opération. Chaque aspect d'une commande de programmation peut être décrit comme une opération qui est appliquée à un objet à un certain emplacement (Fay et Mayer, 1988). Nous décrirons de manière plus détaillée tous ces aspects dans la section suivante. Notre objectif est de montrer que la sémantique du ScratchJr n'est ni simple ni toujours intuitive (c'est que son aspect visuel laisse sous-entendre). Et, par conséquent, un enseignement approprié du ScratchJr, sous une problématique gérée par les apports de la didactique de l'informatique s'avère nécessaire si on veut en tirer des profits cognitifs pendant son apprentissage.

Bien que ScratchJr, en tant qu'héritier direct du langage Logo, comporte plusieurs fonctionnalités sémantiques inscrites dans le paradigme de la programmation impérative (voir par exemple la séquence de commandes pour exécuter un mouvement dans l'espace ou le branchement non conditionnel ALLER à une autre Scène), d'autres paradigmes peuvent être également supportés : la programmation multi-agent, étant donné que plusieurs personnages peuvent être programmés indépendamment et agir de manière concurrente ainsi que des rudiments de la programmation orienté-objets étant donné que le programmeur peut définir plusieurs personnages et leur interaction à partir des messages ou des événements. Dans un autre cas, ScratchJr permet de déclencher une action d'un personnage par une collision avec un autre personnage supporte la programmation par événements. Plus précisément, ScratchJr supporte plusieurs principes de la programmation impérative: affectation, programmation structurée, GOTO, commandes, état, paramètres, instructions et structures de contrôles mais dans sa version actuelle ne supporte pas la structure conditionnelle. De plus, certains principes d'autres paradigmes de programmation sont supportés par ScratchJr, tels que la concurrence, le parallélisme, les messages, le concept d'objets et la coordination – synchronisation.

Les connaissances stratégiques dans ScratchJr

Les connaissances stratégiques sont principalement liées à des capacités transversales se développant à long terme et à la possibilité du transfert dans d'autres domaines. Nous ne traitons pas cet aspect dans ce travail.

ScratchJr : Quels concepts ? Quels apprentissages ?

Une analyse a priori de l'environnement ScratchJr sous un angle cognitif et didactique nous permet d'avancer l'hypothèse que l'on peut aborder plusieurs concepts informatiques (tels que l'alphabétisation numérique : interfaces tactiles et appareils mobiles, la création numérique et l'interactivité), des concepts transversaux relatifs au transfert des connaissances (tels que l'énumération, la direction et l'orientation qui ont trait au développement de la latéralisation, la pensée logique, l'expression orale, l'argumentation, etc.) et certains concepts de programmation (relatifs à des connaissances syntaxiques et sémantiques et à des connaissances stratégiques).

Dans ce travail nous nous limiterons sur les concepts de programmation supportés par ScratchJr, tels que : Instruction, Script, Séquence, Itération, Exécution de processus ou de programme, Parallélisme et Programmation concurrente, Message (appel à une fonction), Programme, Personnage, Scène, Projet, Coordination et Synchronisation.

Leconceptd'animationinformatique

Néanmoins, avant de commencer à présenter les différents concepts de programmation de ScratchJr, il faut se référer à un concept plus général, qui est le concept d'animation. Il s'agit d'un concept central, car la principale activité effectuée par ScratchJr est liée à la création d'*animations informatiques*. C'est probablement la connaissance stratégique principale que nous devons développer en apprenant ScratchJr. Il est évident que cette compétence ne peut pas être développée de manière directe mais elle apparait progressivement en programmant dans cet environnement. La programmation en ScratchJr consiste donc à créer des animations informatiques en plaçant sur une ou plusieurs *scènes* ou *pages* (les emplacements ou se déroulent les opérations) des *personnages*, qui sont les objets programmés. Les comportements des personnages sont décrits à l'aide d'*instructions* qui se rassemblent en *scripts* pour créer des *programmes* (collections de scripts). En d'autres termes, chaque aspect d'une commande de programmation en ScratchJr peut être décrit comme une opération qui est appliquée à un objet (un personnage selon la terminologie de ScratchJr) à un emplacement (un lieu précis sur une scène, voir figure 1).

Le concept d'instruction (brique) et ses multiples aspects

Les instructions en ScratchJr revêtent une forme visuelle (graphique). Elles sont appelées briques de programmation, et sont regroupées en palettes de couleurs différentes (figure 2). Même si la forme visuelle de chaque brique suggère a priori, d'une manière ou d'une autre, ses fonctionnalités, les enfants doivent créer des modèles mentaux opérationnels pour bien les utiliser dans la conception des programmes. Il y a deux catégories de briques : les *briques d'action* et les *briques de contrôle*. Plusieurs briques assemblées dans un puzzle forment un script et un ou plusieurs scripts forment un programme qui

peut être sauvegardé sous un projet. Un script peut être exécuté directement dans l'espace de programmation, permettant ainsi de voir ses effets directs. Un programme est exécuté soit en mode de programmation directe, c'est-à-dire dans l'état du programme où sont visibles les palettes de commandes, soit en mode plein écran, c'est-à-dire dans l'état du programme où n'est visible que la scène. Les deux modes d'exécution n'ont pas toujours des résultats identiques à cause des façons différentes dont ScratchJr gère l'initialisation des places de personnages.

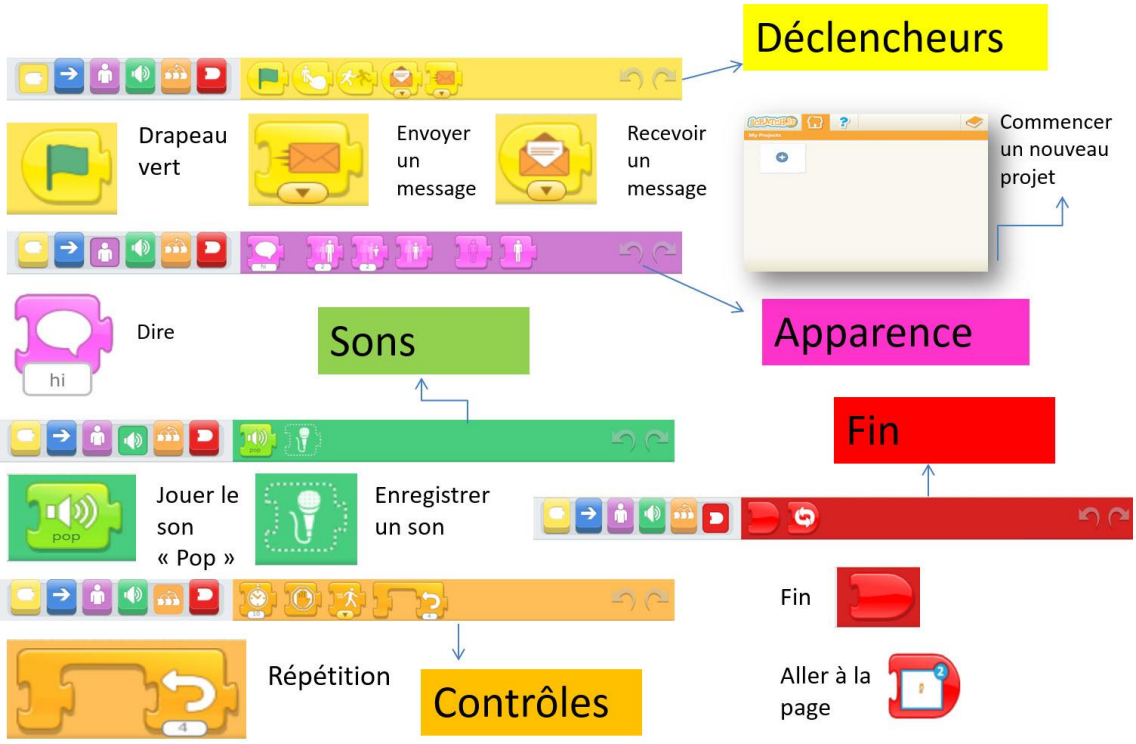

Figure 2 : les différentes commandes de ScratchJr

On peut noter que les deux catégories de briques (briques d'action et briques de contrôle) ne représentent pas la même difficulté conceptuelle (Olabe et al., 2015). Les briques d'action (mouvement, apparition, mise en récit) sont les instructions qui mettent en œuvre les actions des personnages tandis que les briques de contrôle (contrôle et déclenchement d'actions) déterminent quand ces actions se produisent. Construire des modèles mentaux adéquats pour les briques d'action est une opération cognitive plus évidente que construire des modèles mentaux pour les briques de contrôle. Si dans le premier cas les effets d'une action, sont immédiats, l'interface de ScratchJr le permettant par sa conception, les effets d'une structure de contrôle ne le sont pas, étant donné que certaines actions ne se déclencheront que si les conditions du contrôle sont remplies. Une intervention didactique s'avère donc nécessaire pour construire des modèles mentaux appropriés.

Les valeurs et la nature des instructions et le problème d'initialisation

Plusieurs instructions dans ScratchJr nécessitent une valeur pour être exécutées, il s'agit dans ce cas d'une règle syntaxique du langage de programmation. Les éléments de base en ScratchJr sont les briques et dans certains cas elles sont transformées en commandes opérationnelles par l'affectation de valeur. Nous trouvons ici la différence entre une *instruction générale* et une *instruction spécifique* : une instruction générale (en Logo par exemple l'instruction AVANCE) devient une instruction spécifique (AVANCE 50) quand on lui attribue une valeur (initialisation). Dans ScratchJr toutes les instructions, si nécessaire, disposent une valeur par défaut, qui est différente de zéro (voir figure 2). En d'autres termes, l'initialisation d'une valeur dans une instruction n'est pas donc nécessaire quand on l'introduit dans un script. Il s'agit dans ce cas, d'une aide cognitive offerte par les concepteurs de l'environnement ScratchJr. Néanmoins, cette fonctionnalité devient un problème didactique quand on a besoin d'introduire le concept d'initialisation de valeur ou quand on doit modifier la valeur par défaut, qu'il faut tenir compte dans un scénario d'apprentissage de ScratchJr. De toute manière, le concept d'initialisation est très difficile et tout enseignement de ScratchJr doit le prendre en considération en tant qu'obstacle cognitif pour les enfants et en tant que problème didactique pour les enseignants.

La question d'initialisation concerne également la position d'un personnage. En plaçant manuellement un personnage dans la scène on initialise sa position (en glissant le personnage à un point de départ), que l'on peut gérer par la suite soit avec une commande spéciale sur la ligne de commandes (réinitialisation de tous les personnages de la scène, figure 1) soit par une commande de mouvement qui gère de manière séparée la place initiale chaque personnage. Assigner à un personnage une position initiale consiste à lui attribuer sur un système cartésien (visualisable en ScratchJr par un affichage des grilles) un point de départ ce qui pose des difficultés conceptuelles concernant cette gestion.

Une autre question à laquelle on doit faire face pendant l'enseignement du concept d'instruction consiste en sa nature. Il s'agit, dans ce cas, de la différence entre les *instructions absolues* et les *instructions relatives*. Pour une instruction absolue le résultat de son exécution ne dépend pas de l'état actuel du programme, mais seulement des valeurs spécifiées dans l'instruction. C'est par exemple l'instruction qui initialise la position de départ d'un personnage. En revanche, pour une instruction relative le résultat de son exécution dépend à la fois des valeurs spécifiées dans l'instruction et de l'état actuel du programme. C'est le cas pour les instructions principales de mouvement (Déplacer vers la gauche, Déplacer vers la droite, Déplacer vers le haut, Déplacer vers le bas).

Dans toute la tradition de l'approche Logo à l'école primaire, les instructions de mouvement permettaient à faire face aux questions de direction et d'orientation. La plus grande classe de problèmes auxquels l'apprentissage du Logo donnait du sens était donc celui de la latéralisation (tourner à gauche, tourner à droite, avancer, reculer) et tout cela par rapport à un point extérieur de l'enfant, qui devait à se décentrer pour résoudre les problèmes relatifs. Dans ce point, il faut noter, qu'en ScratchJr, une différence par

rapport au Logo habituel où le personnage de la tortue à part sa position cartésienne disposait également une orientation (la direction vers laquelle se dirigeait la pointe de la tortue qui était modifiable par les commandes tourner à droite et tourner à gauche). Les personnages du ScratchJr ne changent pas d'orientation ! Ils n'effectuent par conséquent que des rotations fictives ne modifiant pas la direction de leur mouvement, qui ne s'effectue qu'en horizontale ou en verticale. Ce fait pose des problèmes cognitifs et didactiques importants qui sont liés à la décentration perceptive et à la latéralisation.

Les notions du personnage et de la scène

Le personnage est l'entité de base en programmation en ScratchJr. C'est l'objet sur lequel on applique des opérations qui sont contrôlées par les commandes. Le personnage est géré par un ou plusieurs scripts. Un personnage acte sur une scène. De plus, on peut avoir plusieurs personnages sur plusieurs scènes et à l'aide de la commande ALLER à la PAGE (figure 2), un personnage peut changer de scène.

La scène est l'espace où se déroule l'animation. Les actions des personnages se déroulent sur la scène. On peut avoir plusieurs scènes sur le même projet mais qu'une scène est visible sur l'interface du ScratchJr. Ce fait pose des problèmes de contrôle mental de l'action et des interactions.

Le concept de séquence, et les notions de script et de programme

La séquence constitue la première structure importante de la programmation. Une séquence est un ensemble d'instructions (briques) placées l'une après l'autre. La machine, après le traitement de l'instruction courante traite la prochaine instruction et, par conséquent, pendant une exécution séquentielle, les instructions dans le script ou le programme sont exécutées dans le même ordre où elles apparaissent dans la séquence. La séquence donc est une structure de contrôle implicite. Ce contrôle implicite peut devenir explicite si on utilise des briques spéciales qui déclenchent une exécution modifiée du programme.

Les briques en ScratchJr sont organisées en forme de puzzle de manière *séquentielle.* Chaque puzzle constitue un élément de base en ScratchJr qui s'appelle script. Un *script* est une collection structurée de briques qui spécifie le comportement d'un personnage. Il est construit de la gauche vers la droite. En programmation structurée, il est essentiel d'indiquer la commande avec laquelle un script se déclenche et la commande avec laquelle un script s'arrête. Dans ScratchJr ce principe n'est pas toujours obligatoire.

En revanche, en programmation par événements, qui est supportée également par ScratchJr à l'aide de certaines briques qui sont déclencheurs d'actions de personnages, il n'est pas nécessaire de définir le moment d'exécution d'un script. En touchant le script en mode direct, en recevant un message ou en venant en contact avec un autre personnage, le script peut être déclenché. Dans ce contexte, on peut également noter que plusieurs scripts assignés au même personnage fonctionnement en parallèle (programmation parallèle ou concurrente).

Les notions de programme et de projet

En général, un *programme* est une collection de scripts. Un *programme* dans ScratchJr se compose de la définition de tous les personnages (comment ils ont l'air) et de tous les scripts de chaque personnage (comment ils se comportent). Un programme oblige les personnages à effectuer la série d'actions décrites par les instructions des scripts qui leur sont associés. Le programme en ScratchJr est une notion difficile. L'espace de programmation change chaque fois que l'on change de personnage, donc il n'y a pas d'aperçu global du programme. Ce dernier s'exécute en appuyant le drapeau vert (dans l'espace de programmation ou en mode plein écran).

Un projet contient au moins une scène qui est l'espace où se déroule l'animation. Un projet est géré par un programme et on l'exécute en appuyant sur le drapeau vert. L'association d'un projet à un programme n'est pas apparente mais elle peut probablement se faire quand on sauvegarde le projet. Il faut donc construire un modèle mental adéquat de cette situation.

L'itération (lastructurerépétitive)

L'itération (ou répétition) est la deuxième, après la séquence, structure essentielle en programmation informatique. Elle permet de répéter un nombre de fois les commandes qui se trouvent à son intérieur. Les langages de programmation matérialisent plusieurs formes de répétition. ScratchJr n'en met en œuvre que deux :

a) la première correspond à la répétition d'un nombre de fois connu à l'avance (la commande FOR). Cette répétition est initialisée et le nombre de répétitions est fixé à quatre par défaut. Ceci pourrait être source de problèmes cognitifs pour les enfants, si on n'effectue pas un travail didactique sur les questions concernant d'un côté l'identification des actions à répéter (et construire le segment de commandes associées) et le nombre d'itérations.

b) la deuxième correspond à la répétition perpétuelle. Cette forme de répétition exécute un script de manière continue. Elle permet donc de répéter toutes les commandes qui se trouvent avant elle.

L'initiation à la programmation concurrente et au parallélisme

La programmation concurrente constitue un paradigme de programmation relativement récent. Dans ScratchJr ce modèle est exprimé par le principe de simultanéité : deux ou plusieurs personnages agissent simultanément sur la même scène ou sur des scènes différentes. En fait, quand un programme est exécuté, les scripts de tous les personnages sont exécutés simultanément. Cela signifie que les scripts sont exécutés en même temps (une instruction après l'autre) sans qu'un script attende la fin de l'exécution de l'autre, ou d'être dépendants l'un de l'autre (communication à l'auteur par les développeurs du ScratchJr). Le concept de la concurrence est très difficile mais on le trouve dans la vie courante. A l'âge de 5 à 7 ans, les enfants ne le construisent pas, mais ils peuvent pourtant l'expliquer dans les cas simples.

La communication et la coordination (messages et contacts)

La communication et la coordination entre personnages est effectuée en ScratchJr avec des briques déclencheuses. Il y a deux types de communication : la communication directe et la communication indirecte. La communication directe entre personnages est effectuée avec les briques de messages : il y a un *expéditeur* qui envoie le message et un *récepteur*, dont le script se déclenche quand il reçoit le message. La communication (coordination) entre personnages peut être indirecte : quand deux personnages viennent en contact, le script est déclenché. Il s'agit également d'un concept très difficile que l'on doit étudier pour construire des enseignements adéquats.

Discussionetperspectives

Sous un point de vue de didactique de l'informatique, en apprenant le langage ScratchJr à la maternelle, on développe, entre autres, des connaissances syntaxiques, des connaissances sémantiques et probablement des connaissances stratégiques liées à des compétences de programmation et au transfert dans d'autres domaines. Il s'agit en effet, de l'apprentissage d'un langage de communication avec la tablette pour construire des animations informatiques. Cet apprentissage, s'il y en a, se structure autour des compétences de base en programmation informatique. Des études récentes se sont développées autour de cet apprentissage (Portelance et al., 2015 ; Touloupaki et Baron, 2015) qui nous donnent des résultats prometteurs. Néanmoins, d'autres études sont nécessaires pour mieux comprendre et baliser le terrain d'un point de vue d'enseignement et d'apprentissage de l'informatique.

L'analyse d'ordre cognitif et didactique que nous avons entrepris dans ce texte montre que l'environnement de programmation ScratchJr pourrait s'inscrire dans une approche d'apprentissage de concepts de base de programmation et de développer des compétences associées. Néanmoins, notre analyse montre que la sémantique du ScratchJr n'est ni simple ni toujours intuitive, c'est que son aspect visuel et son ergonomie syntaxique laisse sous-entendre. Par conséquent, un enseignement approprié du ScratchJr, sous une problématique gérée par les apports de la didactique de l'informatique s'avère nécessaire si on veut en tirer des profits cognitifs pendant son apprentissage.L'état actuel des études didactiques concernant l'apprentissage de la programmation pendant la petite enfance à l'aide des environnements visuels de programmation n'étant qu'à ses débuts, plusieurs questions restent ouvertes. Parmi ces questions, on peut citer :

- les conceptions initiales et les modèles mentaux des élèves sur les opérations, les commandes et les objets du ScratchJr,
- les problèmes didactiques sur les connaissances syntaxiques, sémantiques et stratégiques auxquels nous devons faire face et leurs origines,
- l'éventail des scénarios (en tant qu'ensemble d'activités et de situations) didactiques pour faire face à ces problèmes.

Bibliographie

Fay, A. L., & Mayer, R. E. (1988). Learning Logo: A cognitive analysis. In R. E. Mayer (Ed.), *Teaching and learning computer programming: Multiple researcher perspectives* (pp. 55-74). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Komis, V., Misirli, A. (2011). Robotique pédagogique et concepts préliminaires de la programmation à l'école maternelle: une étude de cas basée sur le jouet programmable Bee-Bot, in Baron G.-L., Bruillard, E., Komis V., *Sciences et technologies de l'information et de la communication en milieu éducatif: Analyse de pratiques et enjeux didactiques*, pp. 271-281, Athènes: New Technologies Editions

Misirli, A., & Komis, V. (2014). Robotics and Programming Concepts in Early Childhood Education: A Conceptual Framework for Designing Educational Scenarios. in C. Karagiannidis, P. Politis, & I. Karasavvidis (ed.), *Research on e-Learning and ICT in Education* (pp. 99–118). New York, NY: Springer New York. http://doi.org/10.1007/978-1-4614-6501-0_8

Olabe, J. C., Basogain, X., Olabe, M., (2015). HelloScratchJr.org: Curricular Design and Assessment Tools to Foster the Integration of ScratchJr and Computational Thinking into K-2 Classrooms in Proceedings of 7th international Scratch conference August 12-15, 2015, Amsterdam, pp. 31-41. (dernier accès, le 10 octobre 2015:http://helloscratchjr.org/href=index.php?option=com_content&view=article&id=77&catid=14&Itemid=272)

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Portelance, D. J., Strawhacker, A. L., &Bers, M. U. (2015). Constructing the ScratchJr programminglanguage in the earlychildhood classroom. *International Journal of Technology and Design Education*. http://doi.org/10.1007/s10798-015-9325-0

Touloupaki, S., Baron, G.-L. (2015), Programmation à l'école primaire ? une approche exploratoire en cycle 2, In actes provisoires du colloque éTIC2 (dernier accès le 10 octobre 2015 : http://colloque-etic.fr/media/pdf/24.pdf)

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33. http://doi.org/10.1145/1118178.1118215