Sévastiani Touloupaki[a], Georges-Louis Baron[a] & Vassilis Komis[b]

a. Laboratoire EDA, Université Paris Descartes
sevina.touloupaki@gmail.com, georges-louis.baron@parisdescartes.fr
b. Équipe ICTE, Université de Patras
komis@upatras.gr

Un apprentissage de la programmation dès l'école primaire : le concept de message sur ScratchJr

Résumé

Ce travail est inscrit dans le cadre du projet DALIE de l'Agence Nationale de la Recherche française et son objet d'étude a été d'analyser la manière dont l'usage du logiciel ScratchJr peut permettre aux élèves de primaire de s'approprier des notions de programmation. Dans ce contexte, nous avons choisi de mener une étude exploratoire dans une école primaire française auprès de douze élèves de cours préparatoire. Nous avons élaboré un scénario pédagogique de dix séances (une activité initiale, une activité de préparation psychologique, sept activités d'apprentissage, commençant par la familiarisation avec les concepts de l'interface et se terminant à l'apprentissage des concepts clés de programmation tels que la répétition et les messages, et une activité d'évaluation) qui visait au développement des compétences de programmation chez les jeunes élèves. Même si nous avons étudié plusieurs concepts différents de programmation comme la séquence, la synchronisation, l'itération, etc., nous avons choisi de nous concentrer dans cet article seulement sur l'analyse du concept des messages. Pour cela, nous allons analyser d'une part les réponses des élèves aux entretiens semi-directifs avant et après la mise en place du scénario pédagogique et d'autre part les programmes développés par les derniers au cours de la séance d'évaluation. Nous allons du coup plus loin que ce qu'on avait fait à Ioannina Touloupaki et al. (2016), où avaient été présentés les résultats qui ressortent seulement par les programmes élaborés au cours de la séance d'évaluation par les élèves de l'école maternelle en Grèce et du primaire en France.

Mots clés : didactique de programmation, ScratchJr, message, école primaire, programmation visuelle, programmation orientée objet

1 Introduction

L'apprentissage de la programmation pendant la petite enfance revient à l'ordre du jour depuis quelques années. Plus précisément, nous traversons une période florissante en ce qui concerne les environnements de programmation destinés aux plus jeunes élèves et cela facilite l'étude de l'apprentissage de programmation dès l'école élémentaire.

Pourtant l'apprentissage de la programmation a une longue histoire à l'école élémentaire, qui remonte aux années 60 et à la création du langage Logo par Seymour Papert, Wallace Feurzeig et leur équipe. Ces derniers ont défendu à l'époque l'émergence et la fondation d'une pensée logique, qui renforce la construction de procédures mathématiques (Feurzeig, 2010).

Cet auteur nous explique que l'expérimentation avec Logo permet à l'élève d'améliorer sa pensée critique et de participer au processus éducatif d'une manière active. Il nous explique que l'objectif principal de l'élaboration du Logo était le développement d'une série des compétences qui pourraient avoir des effets positifs dans d'autres domaines.

Selon Clements et Gullo (1984), l'apprentissage de la programmation pourrait jouer un rôle important pour l'amélioration de certains aspects du processus de résolution des problèmes. Ils montrent que l'apprentissage de la programmation peut renforcer le développement d'une pensée divergente et améliorer les résultats des participants aux épreuves métacognitives en limitant le nombre des erreurs réalisées. Selon Clements et Meredith (1992), l'apprentissage de la programmation dans un environnement Logo renforce également le développement des compétences en mathématiques, en résolution de problèmes et en langage. Plus près de nous, Duncan *et al.* (2014) soulignent également le fait que la programmation pourrait faciliter l'apprentissage de concepts utiles pour d'autres matières, comme la littérature ou les mathématiques.

Dans un ouvrage récent, Misirli et Komis (2016) décrivent les résultats de l'expérimentation d'un scénario pédagogique fondé sur l'usage des jouets programmables Bee-Bot. Cette recherche a eu deux objectifs principaux : d'une part, de construire des concepts de base de programmation tels que la séquence, l'algorithme, le programme et la commande et d'autre part, de développer des compétences liées à la pensée informatique chez les jeunes enfants. Il s'agit d'une étude qui a été menée dans 34 classes de maternelle en Grèce, auprès de très jeunes élèves âgés de 4 à 6 ans.

Misirli et Komis (2016) montrent que les jeunes enfants arrivent à résoudre un problème d'orientation dans l'espace en deux dimensions à l'aide du jouet programmable, après la mise en place d'un scénario pédagogique adapté à leurs besoins. Pour aller plus en détail, ils sont arrivés d'abord à verbaliser un algorithme, puis à passer à la programmation (par cartes et sur le jouet programmable) et enfin à le déboguer lorsqu'il y avait une erreur de programmation. Un pseudo-langage qui comprend des cartes qui affichent les commandes qui apparaissent sur le robot a été utilisé en tant que stratégie didactique adéquate, afin de favoriser la description de l'algorithme ainsi que la détection et la correction des erreurs.

Le langage de programmation Logo a beaucoup évolué au cours des années qui ont suivi et plusieurs versions sont disponibles en ce moment dans le marché. Un de ses descendants actuels est le langage de programmation Scratch, qui a constitué la base pour la création de ScratchJr, son petit frère, destiné aux élèves de 5 à 7 ans.

1.1 Le logiciel ScratchJr

ScratchJr a été créé par le laboratoire Média du MIT, l'université Tufts et PICO (Playful Invention Company) afin d'initier les élèves de la grande section de l'école maternelle jusqu'au CE1 aux notions de programmation. L'application est disponible en anglais sous forme d'application gratuite, pour iPad et pour tablette Android. Le code et l'interface ont été simplifiés pour qu'ils puissent correspondre au développement cognitif, personnel, social et affectif des jeunes élèves (Resnick *et al.*, 2013).

ScratchJr, comme son ancêtre Scratch, utilise aussi un environnement de type « Drag and Drop », c'est-à-dire que les enfants doivent cliquer et glisser les icônes sur l'espace de programmation afin de créer des scripts. Il y a une palette de briques colorées qui comprend six catégories différentes. Par exemple, les blocs jaunes permettent de décider du mode de départ du programme, les flèches bleues des déplacements sur l'écran, etc. Les icônes sont assez larges pour que les enfants puissent les utiliser facilement. L'interface est conviviale, joyeuse, amicale et ludique de manière à motiver les jeunes enfants à programmer avec le logiciel. En utilisant ScratchJr, les enfants peuvent exprimer leur créativité, pendant qu'ils programment leurs histoires et leurs jeux interactifs. Ils peuvent modifier facilement leurs

caractères, ajouter leurs propres voix ou des sons, et même insérer des photos d'eux-mêmes (Resnick *et al.*, 2013).

ScratchJr a été créé afin de pouvoir s'intégrer à la classe et promouvoir trois catégories d'apprentissage à travers la programmation : la résolution de problèmes, le développement des connaissances fondatrices (la séquence, l'estimation, la composition, la décomposition, etc.) et l'apprentissage de la lecture, de l'écriture et des mathématiques. Plus précisément, les scripts de programmation sont écrits en poursuivant la direction de l'écriture, de gauche à droite. En outre, le logiciel met à la disposition des utilisateurs une grille, qui facilite une exploration quantitative (mesurer, compter, etc.). La communauté de ScratchJr fournit également du matériel curriculaire et des sources en ligne pour les enseignants, pour soutenir leur travail (Resnick *et al.*, 2013).

Nous pouvons citer les travaux de Dylan Portelance, Marina Umaschi Bers et Amanda Strawhacker (2015) sur l'apprentissage de la programmation à travers le logiciel ScratchJr par les très jeunes élèves. Soixante-deux élèves dès la grande section de l'école maternelle jusqu'à CE1 ont été exposés à un curriculum de six semaines en utilisant le logiciel ScratchJr. Ils ont appris des concepts fondamentaux de programmation et ils ont appliqué ces concepts à leurs propres projets. Les chercheurs s'intéressent à distinguer les blocs de programmation que les jeunes élèves préfèrent utiliser pour leurs projets après avoir appris tous les blocs disponibles sur l'application. L'analyse des données montre que les blocs qui sont utilisés le plus souvent par les élèves sont ceux qui sont responsables du mouvement du personnage dans la scène.

Dans un contexte un peu plus large, Dylan Portelance (2015) a créé sous la direction de Marina Umaschi Bers une activité novatrice qui s'appelle « Code and Tell », dont la conception vise à initier les jeunes élèves aux concepts de base de programmation à travers l'application ScratchJr. C'est une étude exploratoire mise en place afin de comprendre comment cette activité facilite l'apprentissage des concepts de pensée informatique par les jeunes élèves lorsque celui-ci est accompagné d'entretiens directifs entre les élèves. Soixante-six élèves de CE1 ont été exposés à un curriculum de treize séances pour réaliser trois genres de projets différents : collages, histoires et jeux interactifs avec ScratchJr. Au cours de chaque unité du curriculum, les élèves participaient à deux grandes sous-unités : une période consacrée à créer leurs propres projets (soit collages, soit histoires, soit jeux) et une deuxième période consacrée à participer à l'activité « Code

and Tell ». Cette dernière est constituée d'entretiens directifs menés auprès des élèves par groupes de deux, c'est-à-dire que chaque élève posait quatre questions à son binôme, dont les trois premières étaient prédéfinies par le chercheur et la dernière était de leur propre choix. L'analyse des données a conduit l'auteur à créer trois grandes catégories de projets en fonction du comportement des élèves, pendant la présentation de leurs projets : les projets « descriptifs », les projets « démonstratifs » et les projets « imaginatifs ». Il souligne le fait que chaque catégorie a un lien fort avec la pensée informatique. En fonction de la complexité avec laquelle les élèves ont présenté leurs projets, Portelance a créé trois sous-catégories à l'intérieur de chaque grande catégorie, les « simples », les « intermédiaires » et les « complexes » (Touloupaki, 2016).

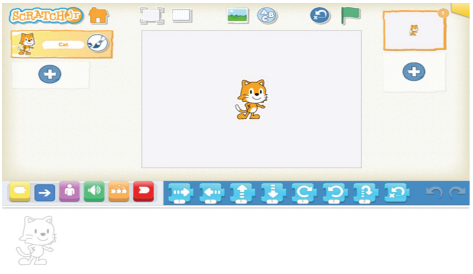

Figure 1 : L'interface de ScratchJr.

1.2 La singularité de ScratchJr et le concept de message

ScratchJr, comme Scratch, suit un paradigme de programmation orientée objet fondé sur les principes de Smalltalk. Smalltalk a été le premier langage de programmation orientée objet, développé par Alan Kay, en 1971 (Kay, 1993). La programmation orientée objet ou programmation par objets analyse les problèmes de manière à identifier les « acteurs » qui le composent et puis à déterminer ce qu'est et ce que doit faire chaque acteur. Ce paradigme utilise l'objet comme principe unificateur (Cloutier, 1988).

Un objet a un état qui est l'ensemble de ses propriétés et un comportement spécifique, c'est-à-dire un ensemble de réactions qui peuvent être déclenchées par d'autres objets à travers l'envoi de messages. Il est également doté d'un ensemble de scripts qui définissent son comportement. Par

exemple, un objet peut modifier certains de ses attributs à la demande d'un autre. Le programmeur doit décrire quels échanges entre les acteurs (objets) produiront les comportements adéquats pour la résolution d'un problème (Cloutier, 1988). Les objets qui ont les mêmes attributs constituent une classe d'objets (Ben-Ari, 1996).

Dans ce contexte, ScratchJr permet de définir plusieurs personnages, de créer pour chacun d'entre eux des scripts spécifiques et de les faire communiquer en utilisant les messages (Komis, 2016). C'est pourquoi ce paradigme permet de mettre en place en même temps des formes de la programmation concurrente.

En effet, les scripts des personnages sur ScratchJr s'exécutent de manière concurrente et pas l'un après l'autre. L'utilisateur devra avoir des mécanismes permettant de définir l'ordre de l'exécution des scripts des objets, pour résoudre ses problèmes. Un tel mécanisme est rendu assez facile par le concept de messages.

Ce dernier est implémenté par deux commandes : la commande « envoyer message », qui envoie un message de couleur spécifique à un personnage et la commande de « quand message reçu », avec laquelle le script du personnage dont la commande du message a la couleur est déclenché.

La commande « envoyer message » se fait dans un script de programmation, comme toutes les autres commandes sur ScratchJr. Son seul paramètre est la couleur. La commande « quand message reçu » sert à déclencher un événement et elle se place au début d'un script. En ce qui concerne la sémantique, lorsque l'exécution du script arrive à l'envoi d'un message de couleur spécifique, ce message est envoyé et les scripts des personnages, qui prévoient l'arrivée d'un message de la même couleur sont déclenchés de manière concurrente.

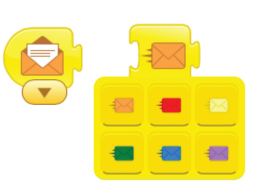

Figure 2 : Les commandes de messages.

2 Problématique et questionnements de recherche

Nous avons choisi pour objet d'étude, l'analyse de la manière dont l'usage du logiciel ScratchJr peut permettre aux élèves de CP de s'approprier des notions de programmation. Plus précisément, notre problématique est formulée de la manière suivante :

- *Comment initier les enfants de 6 à 7 ans aux notions de base de programmation à travers le logiciel ScratchJr et plus précisément au concept des messages ?*

Les messages sont un concept de programmation de haut niveau, peu étudié dans la littérature scientifique au niveau de la petite enfance. Notre objectif principal a été la compréhension de la fonctionnalité des messages par les élèves et leur utilisation opérationnelle, afin de résoudre un problème donné. Dans ce contexte, une autre raison pour laquelle nous avons choisi d'étudier ce concept était sa capacité potentielle à contribuer au développement d'une pensée informatique chez les jeunes enfants. Plus précisément, nos questionnements sont les suivants :

- Comment les élèves perçoivent-ils les instructions des messages ?
- Comment les élèves arrivent-ils à utiliser les commandes des messages de manière opérationnelle afin de pouvoir résoudre un problème d'évaluation ?
- Comment les élèves arrivent-ils à construire des représentations complètes autour du concept des messages ?

3 Méthodologie

3.1 Recherche de conception

Nous avons choisi de mettre en place une recherche de conception ou *design-based research* en anglais. Il s'agit d'une approche qui met au centre la conception d'un scénario pédagogique (le *design*) et l'évaluation de sa

capacité à transmettre les savoirs choisis par son créateur à travers la mise en place itérative d'une série d'interventions dans un terrain choisi. La recherche de conception permet une compréhension profonde du processus de l'apprentissage (Cobb *et al.*, 2003).

Cobb *et al.* (2003) font référence à cinq de ses caractéristiques principales. D'après eux, une telle recherche a comme objectif le développement d'une série des théories à propos du processus de l'apprentissage et des stratégies didactiques mises en place, afin de pouvoir faciliter ce processus d'apprentissage. Ensuite, la recherche de conception a un caractère interventionniste, car un de ces objectifs est d'investiguer les possibilités d'une amélioration de l'éducation à travers l'étude des stratégies différentes d'enseignement. Puis, une recherche de conception a deux facettes : le potentiel et le réfléchissant. Ces deux facettes constituent une quatrième caractéristique qui est l'itération. La cinquième caractéristique de la recherche de conception est ses racines pragmatiques, c'est-à-dire que les théories qui émergent d'une recherche de conception doivent avoir un sens spécifique pour cette conception.

Au cours de la mise en place d'une recherche de conception, un des objectifs principaux est d'améliorer la conception du début à travers des évaluations itératives des différentes conjectures, en faisant en même temps des analyses autour du raisonnement des élèves et de l'environnement d'apprentissage (Cobb *et al.*, 2003).

Dans ce cadre, le chercheur fait une analyse holistique des interventions éducatives, c'est-à-dire qu'il perçoit les interventions comme des interactions entre les enseignants, les apprenants et les matériels utilisés (The Design-Based Research Collective, 2003).

3.2 Étude de cas

Une étude de cas a été effectuée. Il s'agit d'une méthode mixte de la recherche, qui permet une approche exploratrice, descriptive et explicative du phénomène étudié (Yin, 2003). L'étude de cas peut aussi avoir une direction plutôt qualitative ou quantitative en fonction de la posture du chercheur, de la question de départ, du terrain choisi, etc. Dans le cadre de notre projet, la méthode reste mixte, mais elle est caractérisée par une visée plutôt qualitative.

Nous avons choisi ce type de recherche parce que nous voulions nous concentrer non seulement sur le terrain mais aussi sur les comportements et les interactions de l'échantillon, par rapport au logiciel ScratchJr et aux notions de programmation en général. Nous nous fonderons donc sur des données empiriques de terrain (approche inductive) et les interpréterons pour arriver aux résultats et, à terme, à la construction de modèles et de théories, toujours en rapport avec la population étudiée. Il s'agit d'une stratégie de recherche préférable lorsque les circonstances et le problème de la recherche sont convenables et lorsque le chercheur ne contrôle pas les comportements d'échantillon. L'étude de cas est une méthode empirique qui examine un phénomène contemporain où les frontières entre ce phénomène et le contexte ne sont pas visibles (Yin, 2003).

La formulation de notre questionnement est compatible avec cette méthode, car elle est adaptée pour réaliser une recherche dont les questions de départ sont posées sous la forme « Comment ? » et « Pourquoi ? » (Yin, 2003).

3.3 Le scénario pédagogique

Ce scénario pédagogique forme le noyau de notre projet de recherche et il comprend une série d'activités qui ont comme but l'introduction de concepts préliminaires de programmation comme la séquence et l'itération, mais surtout le concept des messages, à travers le logiciel ScratchJr.

Tout d'abord, nous avons distingué deux grandes catégories de concepts : les concepts d'interface et les concepts de programmation. Parmi les concepts d'interface disponibles sur ScratchJr, nous avons choisi de travailler sur l'accueil, la scène, l'initialisation de la scène, le mode de présentation, le paysage, le drapeau vert, l'enregistrement d'un projet, l'éditeur graphique, les catégories des commandes, l'espace de programmation, le script de programmation et les personnages. Ce sont les concepts de base du logiciel qu'on devait enseigner aux élèves, afin de pouvoir les initier à la programmation.

Ensuite, nous avons distingué deux catégories de concepts de programmation, basées sur la facilité d'acquisition des élèves : les concepts de niveau haut (comme la séquence, les messages, etc.) et les concepts de niveau bas (comme le déplacement du personnage vers la gauche ou la droite, etc.). En rendant compte du développement cognitif, affectif et social des enfants de notre échantillon, nous avons décidé des concepts de programmation que

nous avons explorés dans notre scénario. Plus précisément, notre ambition était d'examiner d'une part les représentations des enfants à propos des concepts choisis, avant l'application du scénario et d'autre part s'il y a du progrès cognitif après l'application du scénario.

Parmi les concepts de programmation disponibles sur ScratchJr, nous avons choisi d'étudier les suivants : les instructions, les valeurs des instructions, la séquence, l'itération (boucle), la programmation concurrente/ la synchronisation, le programme et les messages.

Tableau 1 : Les concepts de programmation en jeu.

Concept de programmation	Description du concept de programmation
Instruction	Chaque instruction oblige un personnage à effectuer une action.
Valeur d'instruction	Plusieurs instructions nécessitent une valeur pour être exécutées.
Programme	Un programme est un ensemble d'instructions destinées à être exécutées par le logiciel ScratchJr.
Script de programmation	Un script de programmation est une connexion de commandes structurées qui spécifie le comportement d'un personnage. Les commandes en ScratchJr sont organisées en forme de puzzle de manière séquentielle. Un script s'exécute de gauche à droite.
Séquence	Une série d'instructions placées l'une après l'autre.
Itération	La répétition est une boucle itérative qui permet de répéter un nombre de fois prédéfini les commandes qui se trouvent à l'intérieur.
Programmation concurrente	En cliquant sur le drapeau vert, tous les scripts s'exécutent simultanément. Cela signifie que les scripts sont exécutés en même temps sans qu'un seul script ait besoin d'attendre la fin de l'autre ou sans que les scripts soient dépendants les uns des autres.
Messages	Les personnages communiquent à l'aide de deux commandes de messages, la commande « envoyer message » et la commande « quand message reçu ». Les deux personnages sont coordonnés lorsqu'ils échangent des messages.

En ce qui concerne le scénario pédagogique, nous avons conçu et réalisé, pendant un mois et demi, sept activités d'enseignement, une activité initiale et une activité finale d'évaluation. Chaque jour, nous avons animé une séance de trente à quarante minutes.

Au cours de l'activité initiale, nous avons d'abord conduit un entretien personnel (pré-test) pour détecter les représentations initiales des élèves à propos des concepts étudiés et d'autre part, une activité de préparation psychologique et cognitive. Pendant la séance de préparation, nous avons présenté aux élèves certains concepts de la programmation. Nous leur avons présenté l'utilisation de la tablette, en leur apprenant les processus du verrouillage et du déverrouillage. Ensuite, nous avons fait une première familiarisation avec la manipulation de la tablette et une initiation au logiciel ScratchJr.

Ensuite, nous sommes passés aux séances d'enseignement. Au cours de la première, nous avons exploré comment ouvrir l'application, démarrer un nouveau projet, insérer et effacer des briques de programmation, utiliser les commandes de mouvement afin de déplacer le personnage dans la scène, définir la position du personnage mais aussi le processus de l'enregistrement d'un projet. Nous avons aussi essayé de comprendre la signification des instructions pour le personnage.

Au cours de la deuxième séance, nous avons présenté le principe « début-fin » de la programmation (séquence) et comment : sélectionner un arrière-plan, présenter un projet en plein écran et introduire des nouveaux personnages dans la scène. Nous avons aussi appris comment faire tourner le personnage à gauche et à droite et comment le faire sauter. Nous avons appris également l'icône pour initialiser la position du personnage.

Pendant la troisième activité, nous avons montré comment enregistrer un son, écrire un message et effacer un personnage sur la scène. Nous avons aussi réalisé une révision de toutes les briques de mouvement apprises jusqu'alors.

Au cours de la quatrième activité, nous avons abordé un concept principal de programmation, celui de la boucle, c'est-à-dire apprendre à répéter une commande et un motif de commandes. Pour enseigner ce concept, nous avons choisi de présenter aux élèves des projets déjà prêts qui contiennent la nouvelle commande et les laisser expérimenter et découvrir la fonctionnalité de la commande à l'aide de nos questions. Au cours de la cinquième activité, nous avons résolu un problème en utilisant la commande de la répétition.

Les sixième et septième activités ont été dédiées au concept de message. Plus précisément, au cours de la sixième nous avons enseigné aux élèves la fonctionnalité des messages. Pour cela, nous avons présenté des projets déjà prêts qui contiennent les commandes des messages et nous les avons laissés expérimenter et découvrir la fonctionnalité de chacune des commandes, à l'aide de nos questions. Pour faciliter la compréhension du concept par les élèves, nous avons choisi d'utiliser une partie du matériel proposé par le site ScratchJr, les images des commandes du logiciel à imprimer. Nous avons donc construit les commandes du ScratchJr en version papier, que les élèves ont utilisées pour comprendre la fonctionnalité des messages et pour résoudre le problème de la septième activité. Dans la figure qui suit, nous présentons le matériel utilisé au cours de ces deux séances.

Figure 3 : Les commandes et les personnages du ScratchJr en version papier (matériel utilisé pour la mise en place du scénario pédagogique).

Au cours de l'activité d'évaluation, nous avons mené un entretien personnel (post-test) avec les questions de l'activité initiale et un même problème à résoudre pour tous les élèves.

Nous avons mis en place l'activité initiale et l'activité finale, afin de pouvoir détecter le développement cognitif des enfants à propos des concepts étudiés, avant et après l'application de notre scénario pédagogique.

Enfin, nous avons mené une séance dédiée à la découverte de l'éditeur graphique, car les élèves voulaient savoir sa fonctionnalité.

3.4 L'échantillon

Notre recherche a été réalisée dans une école primaire en banlieue parisienne auprès de 12 élèves, 8 filles et 4 garçons. Ces élèves étaient les seuls dons les parents avaient signé l'autorisation de participer à notre recherche. Les élèves ont travaillé par groupes de quatre, avec quatre tablettes pour réaliser les activités proposées. Nous avons travaillé avec trois groupes de quatre élèves. Les groupes ont été constitués par l'enseignante de la classe, de manière à pouvoir représenter tous les niveaux scolaires différents présents dans la classe. Les tablettes que nous avons utilisées étaient des iPad. L'enseignante de la classe n'a pas participé à notre étude.

3.5 Techniques et outils de recueil de données

Pour recueillir nos données, nous avons choisi d'utiliser comme techniques l'observation participante et les entretiens semi-directifs. Pour faciliter le recueil de données, nous avons utilisé un enregistrement audiovisuel du projet, ainsi que nos notes personnelles avec les incidents critiques et les grilles d'analyse pour l'activité initiale et l'activité finale. Nous avons également utilisé des captures d'écran des programmes élaborés par les élèves tout au long du processus.

4 Résultats

Une première étude empirique menée en 2016–2017 nous avait permis d'avoir des résultats préliminaires à propos de l'initiation des élèves de CP aux concepts de programmation et plus précisément du concept du message. Nous avons dès lors analysé d'une part les données recueillies au cours de la séance d'évaluation et, d'autre part, ceux recueillis par le pré-test et post-test.

Dans un premier temps, nous avons analysé les programmes élaborés[1] par les élèves face à un même problème d'évaluation et les enregistrements vidéo des séances de l'évaluation. Nous sommes arrivés donc à distinguer trois types de construction du concept de messages en fonction de la performance des élèves face à ce problème : une « construction complète », une « construction partielle » et une « construction incomplète ».

Pour arriver à cette typologie nous avons défini une série des critères qu'un programme devait satisfaire afin de pouvoir être caractérisé « complet ». Dans le Tableau 2 nous présentons les critères pour une construction complète de la notion des messages par les élèves. Les critères sont au nombre de deux (A, B), parce que nous avons choisi de ne pas enseigner le paramètre des commandes des messages, c'est-à-dire la couleur.

La classification des élèves selon la typologie décidée s'est fondée sur l'énumération du nombre des critères que leur programme satisfaisait. Pour que les critères A et B soient satisfaits, il faut que leurs deux parties soient satisfaites aussi. C'est-à-dire le personnage (enfant ou papillon) dont le script doit contenir la commande concernée, mais aussi la position à laquelle se trouve la commande concernée sur le script du personnage.

Tableau 2 : Critères de la construction complète de la notion des messages.

A. La commande « envoyer message » se trouve après la commande de mouvement sur le script du personnage « enfant ».
B. La commande « quand message reçu » se trouve au début du script du personnage « papillon ».

Dans le Tableau 3 nous présentons les types de construction de la notion des messages que nous avons distingués au cours de notre analyse, ainsi que le nombre d'élèves qui appartiennent à chaque catégorie. Les critères qui apparaissent sur le tableau sont les critères satisfaits pour chaque type de construction. Par exemple, bien que pour une « construction complète » nous devions avoir satisfait les deux critères, A et B, pour une « construction partielle » nous devons avoir satisfait seulement le critère B. De la même manière fonctionne le tableau pour le troisième type de construction de la notion des messages.

1 Une partie de cette analyse a été présenté au colloque < http://didinfo2016.etpe.gr/> en Grèce.

Tableau 3 : Types de construction de la notion de messages.

Types de construction	Critères satisfaits	Nombre d'élèves
Construction complète	A & B	8
Construction partielle	B	3
Construction incomplète	—	1

Comme nous pouvons observer sur le Tableau 3, la plupart des élèves de notre échantillon arrivent à construire de manière complète le concept de message. Plus précisément, 8 sur 12 résolvent correctement le problème donné en utilisant les messages de manière opérationnelle. Trois arrivent à une construction partielle et 1 arrive à construire la notion de manière incomplète.

Figure 4 : Exemple d'une construction complète.

En analysant les erreurs des élèves sous le prisme de la théorie de Mayer (1988) sur les savoirs syntaxiques et sémantiques de la programmation, nous pouvons remarquer que ceux qui arrivent à une construction partielle ou incomplète du concept des messages font des erreurs qui concernent surtout la sémantique du langage ScratchJr. Pourtant, nous avons aussi distingué un autre type d'erreur qui émerge dans les programmes des élèves en raison de la non-compréhension de l'énoncé du problème. Nous avons choisi d'appeler ce type d'erreur « erreur compréhension énoncé ».

Figure 5 : Exemple d'une construction partielle.

Dans la figure 5 par exemple, nous pouvons distinguer une « erreur syntaxique », car l'élève a ajouté « envoyer message » (opération nº 1) au script du personnage « papillon » (objet nº 2) et pas au script du bon personnage,

c'est-à-dire du personnage « enfant » (objet nº 1). En plus, il a fait une « erreur sémantique », car il a modifié la couleur du « envoyer message » et d'une telle manière que même s'il avait utilisé cette commande dans le script du personnage « enfant », les deux personnages n'auraient pas pu échanger des messages.

Figure 6 : Exemple d'une construction partielle.

Dans la figure 6 nous pouvons remarquer deux « erreurs sémantiques », une « erreur syntaxique » et une « erreur compréhension énoncé ». L'élève a fait une « erreur syntaxique », car elle a utilisé « envoyer message » dans le script du personnage « papillon », au lieu de l'utiliser seulement sur le script du personnage « enfant ». C'est une « erreur sémantique » en même temps, car il n'est pas possible au personnage qui reçoit le message d'être le personnage qui l'envoie de nouveau après. Selon la sémantique du langage ScratchJr « envoyer message » est une opération qui devrait être effectuée sur un objet. Par ailleurs, le fait qu'elle ait utilisé « envoyer message » dans le script du personnage « enfant », mais pas à la bonne position, nous montre soit qu'elle n'a pas bien compris l'annonce du problème et dans ce cas elle fait une « erreur compréhension énoncé », soit qu'elle n'a pas bien compris la sémantique de la commande « envoyer message » et c'est pourquoi elle ne comprend pas le fait que la position de la commande sur le script du personnage joue un rôle important, car elle définit le moment ou l'autre personnage va déclencher son script.

Figure 7 : Exemple d'une construction partielle.

La figure 7 nous présente une erreur qui est en même temps « syntaxique » et « sémantique », car l'élève a oublié d'utiliser la commande « envoyer message » (opération nº 1) sur le script du personnage « enfant » (objet

nº 1). Cela est une erreur syntaxique, car le concept des messages comprend deux commandes qui sont effectuées sur deux personnages différents et l'élève a utilisé seulement la commande du « quand message reçu » sur le personnage « papillon ». En ce qui concerne la sémantique, il s'agit d'une erreur, car le personnage papillon ne peut pas recevoir un message qui ne lui est pas envoyé par le personnage « enfant ».

Figure 8 : Exemple d'une construction incomplète.

Ensuite, dans la figure 8 nous pouvons distinguer quatre erreurs. D'abord une « erreur syntaxique », car l'élève a oublié d'utiliser la commande du « quand message reçu ». Après, il a fait une « erreur sémantique », car il a utilisé la commande « envoyer message » sur le script du personnage « papillon », qui devrait avoir la commande du « quand message reçu ». Les deux personnages ne vont pas communiquer. En plus, il a fait une autre erreur lorsqu'il a utilisé la commande « envoyer message » sur le script du personnage « enfant ». Il se trouve que c'est le même cas que pour la figure 6 : soit « erreur compréhension énoncé », soit « erreur sémantique ».

En analysant les erreurs des élèves nous pouvons remarquer que la plupart d'entre eux se trompent avec la commande « envoyer message » et sa position dans le script, ou avec le caractère, ici « papillon » ou « enfant », dont le script devrait la contenir. Nous pouvons supposer que la forme de la commande du « quand message reçu » aide les enfants à la poser au bon endroit – c'est-à-dire au début du script.

En faisant une comparaison entre la performance face au problème de l'évaluation et les réponses aux entretiens de post-test, nous pouvons souligner le fait qu'il existe une corrélation assez importante entre les deux, pour une partie de ceux qui arrivent à résoudre correctement le problème de l'évaluation et pour tous les élèves qui n'arrivent pas à le résoudre correctement.

Les 4 élèves dont nous avons analysé les erreurs auparavant éprouvent une série des difficultés en ce qui concerne la verbalisation de leurs connaissances autour du concept des messages. Plus précisément, ils construisent des « représentations incomplètes » des commandes « envoyer message » et

du « quand message reçu » au post-test. Parmi ceux qui arrivent à résoudre correctement le problème, 3 élèves construisent une représentation complète du « envoyer message » et 4 élèves construisent une représentation complète du « quand message reçu ». Pour les autres qui arrivent à résoudre correctement le problème d'évaluation, il n'y a pas de corrélation avec leurs réponses au post-test.

L'analyse nous a également montré que la commande « envoyer message » pose la plupart des problèmes aux élèves tant au cours de la séance d'évaluation, qu'au cours de la séance de post-test. Nous pouvons donc faire l'hypothèse que le savoir autour du « envoyer message » n'est pas complètement construit dans le cerveau des élèves et c'est pour cette raison qu'ils n'arrivent pas à bien verbaliser leurs connaissances autour de cette commande, alors qu'ils arrivent mieux pour la commande du « quand message reçu ».

En faisant une comparaison entre les réponses des élèves à l'entretien personnel avant et après l'application du scénario pédagogique, on peut remarquer le fait que la plupart de nos sujets (8 sur 12) réalisent un progrès cognitif, car leurs réponses à propos des commandes « envoyer message » et « quand message reçu » au post-test sont meilleurs que celles au pré-test. Seulement 4 élèves ne font pas de progrès.

Les difficultés des jeunes à s'exprimer autour des commandes des messages sont tout à fait normales, si on prend en compte le fait que nous avons mené seulement deux séances d'enseignement du concept, à cause de la durée courte de notre étude. Au cours de ces deux séances, nous avons fait attention à la compréhension de la fonctionnalité des messages et leur utilisation opérationnelle face aux problèmes donnés. Si on analyse du coup nos résultats par ce point de vue, nous pouvons dire que notre objectif a été atteint.

5 Discussion

Notre étude a eu un statut exploratoire et souffre de sérieuses limites. L'échantillon est d'abord de faible taille et nos observations ont été de courte durée. Pourtant, nous pensons que leur validité, concernant les difficultés rencontrées, va au-delà de notre étude.

D'autres recherches utilisant des échantillons plus importants seraient nécessaires, afin de pouvoir mieux comprendre le processus de l'appropriation du concept des messages par les jeunes élèves. Nous nous intéressons en particulier à comprendre la manière dont les élèves construisent leurs représentations autour du concept des messages. Il nous paraît intéressant aussi de comprendre pourquoi la commande « envoyer message » pose plusieurs difficultés aux élèves que la commande du « quand message reçu ».

Nous souhaitons donc examiner davantage le concept des messages, mais aussi d'autres concepts qui sont supportés pas le logiciel ScratchJr et appartiennent à la programmation événementielle, comme le concept « Toucher » (*Start on tap* en anglais) ou « Contact » (*Start on bump* en anglais). C'est pourquoi nous avons mené, au cours du dernier semestre, une étude auprès de 22 élèves de deux classes de grande section de l'école maternelle en Grèce, afin de pouvoir examiner également comment réagissent les plus jeunes face au même concept. Les résultats préliminaires de cette étude nous montrent des similarités en ce qui concerne les difficultés rencontrées et les types d'erreurs chez les plus jeunes élèves face au concept de message.

Références

Ben-Ari, M. (1996). Understanding Programming Languages (1st éd.). New York, NY, USA : John Wiley & Sons, Inc.

Clements, D. H., & Gullo, D. F. (1984). Effects of Computer Programming on Young Children's Cognition. American Psychological Association Inc., 76(6), 1051–1058.

Clements, D. H., & Meredith, J. S. (1992). Research on Logo : Effects and Efficacy. Logo Foundation, p. 15.

Cloutier, J.-F. (1988). Apport de différents paradigmes de programmation comme autant d'outils de pensée (p. 195–203). Présenté à Colloque francophone sur la didactique de l'informatique, Paris. Consulté à l'adresse <https://www.epi.asso.fr/fic_pdf/dossiers/d07p195.pdf>.

Cobb, P., Confrey, J., DiSessa, A., Lehrer, R., & Shauble, L. (2003). Design Experiments in Educational Research. Educational Researcher, 32(1), 9–13.

Duncan, C., Bell, T., & Tanimoto, S. (2014). Should Your 8-year-old Learn Coding ? In Proceedings of the 9th Workshop in Primary and Secondary Computing Education (p. 60–69). New York, NY, USA : ACM. <https://doi.org/10.1145/2670757.2670774>.

Feurzeig, W. (2010). Toward a Culture of Creativity : A Personal Perspective on Logo's Early Years and Ongoing Potential. International Journal of Computers for Mathematical Learning, 15(3), 257–265. <https://doi.org/10.1007/s10758-010-9168-4>.

Kay, A. C. (1993). The Early History of Smalltalk. In The Second ACM SIGPLAN Conference on History of Programming Languages (p. 69–95). New York, NY, USA : ACM. <https://doi.org/10.1145/154766.155364>.

Komis, V. (2016). Une analyse cognitive et didactique du langage de programmation ScratchJr (p. 11). Présenté à Didapro6-DidaSTIC, Namur, Belgique. Consulté à l'adresse <https://didapro6.sciencesconf.org/76475/document>.

Mayer, R. E. (Éd.). (1988). Teaching and Learning Computer Programming : Multiple Research Perspectives. Hillsdale, N.J : Routledge.

Misirli, A., & Komis, V. (2016). Construire les notions de l'orientation et de la direction à l'aide des jouets programmables : une étude de cas dans les écoles maternelles en Grèce. In, F. Villemonteix, J. Béziat, G-L.Baron (Ed.), L'école primaire et les technologies informatisées.

Portelance, D. J. (2015). Code and Tell : An exploration of peer interviews and computational thinking with ScratchJr in the early childhood classroom. Tufts University. Consulté à l'adresse <https://ase.tufts.edu/DevTech/resources/Theses/DPortelance_2015.pdf>.

Portelance, D. J., Strawhacker, A. L., & Bers, M. U. (2015). Constructing the ScratchJr programming language in the early childhood classroom. International Journal of Technology and Design Education, 1–16. <https://doi.org/10.1007/s10798-015-9325-0>.

Resnick, M., Kazakoff, E. R., Bonta, P., Silverman, B., Bers, M. U., & Flannery, L. P. (2013). Designing ScratchJr : Support for Early Childhood Learning Through Computer Programming (p. 10). New York, NY, USA. Consulté à l'adresse <http://ase.tufts.edu/DevTech/publications/scratchjr_idc_2013.pdf>.

The Design-Based Research Collective. (2003). Design-Based Research : An Emerging Paradigm for Educational Inquiry, 32(1), 5–8.

Touloupaki, S. (2016). Analyse d'une recherche sur la pensée informatique. Consulté 24 juin 2016, à l'adresse <http://www.adjectif.net/spip/spip.php?article396>.

Touloupaki, S., Konstantinopoulou, M., Nicolos, D., Baron, G.-L., & Komis, V. (2016). Η οικοδόμηση της έννοιας του « μηνύματος » στη γλώσσα προγραμματισμού ScratchJr από μαθητές 5–7 ετών. (p. 8). Présenté à didinfo2016, Ioannina. Consulté à l'adresse <http://www.etpe.gr/custom/pdf/etpe2382.pdf>.

Yin, R. K. (2003). Case Study Research : Design and Methods (3rd Revised edition). Thousand Oaks, Calif : SAGE Publications Ltd.