Analyse d’une recherche sur la pensée informatique

Pour citer cet article :

Touloupaki Sévina (2016). Analyse d’une recherche sur la pensée informatique : "Code and tell" de Dany Portelance. *Adjectif.net*. [En ligne] http://www.adjectif.net/spip/spip.php?article396

Résumé :

Le texte qui suit est une recension du master soutenu par Dylan Portelance en 2015 : “An Exploration of Peer Interviews and Computational Thinking With ScratchJr in the Early Childhood Classroom”. Il s’agit d’analyser l’acquisition d’une pensée informatique par des jeunes enfants.

Mots clés :

Apprentissage de l’informatique, École primaire, Culture informatique, Environnement informatique, Robotique, Politiques publiques, Programmes scolaires, IPT

1. Introduction

Dylan Portelance a présenté, pour l’obtention du master qu’il a préparé sous la direction de Marina Umaschi Bers, une activité novatrice dénommée « Code and Tell », dont la conception vise à initier les jeunes élèves à la programmation avec le logiciel ScratchJr [1]. Il s’agit d’une étude exploratoire qui vise à comprendre comment cette activité facilite l’apprentissage de concepts de pensée informatique par les jeunes élèves, lorsque celui-ci est accompagné d’entretiens directifs entre les élèves.

Tout d’abord, l’auteur explique sa motivation à travailler sur la programmation en soulignant la grande importance qui a été donnée ces dernières années à cette activité. Plus précisément, il met l’accent sur les effets positifs qui l’accompagnent et peuvent être aussi utiles dans d’autres domaines. Il cite notamment Douglas Rushkoff (2011) qui a considéré l’apprentissage du codage comme aussi important que celui de la lecture et de l’écriture. En outre, l’apparition de ScratchJr lui a donné envie d’étudier de quelle manière un tel logiciel peut aider à acquérir une pensée informatique, à laquelle la programmation appartient.

L’ auteur se réfère à Seymour Papert en tant que père du concept, puisqu’il a été le premier à parler d’un nouveau mode de pensée auquel nous pouvons avoir accès à travers la résolution des problèmes (Papert, 1993). Portelance fait bien sûr référence à l’ambassadrice contemporaine de la pensée informatique, Jeannette Wing (2006), qui définit le terme comme une série des méthodes d’analyse utilisée afin de combiner l’homme et la machine face à la résolution des problèmes. L’auteur souligne aussi le fait que Wing comme Papert se réfèrent aux techniques utilisées par les machines et les humains. Pour approfondir la définition de la pensée informatique l’auteur cite les trois dimensions suivantes : « concepts », « pratiques » et « perspectives », émergeant de l’analyse du terme par Brennan et Resnick (2012).

2. Notion de pensée informatique et développement de l’application Scratch Junior

Portelance explique qu’après la définition du terme par Jeannette Wing (2006), un mouvement fort se déclenche

pour que ce nouveau groupe de compétences puisse faire partie de l'éducation, comme un objectif principal, équivalent à la lecture et l'écriture.

Plusieurs efforts ont été effectués afin d'évaluer l'apprentissage d'une pensée informatique. L'auteur cite quatre approches méthodologiques différentes. La première se fonde sur l'analyse des projets créés par les élèves à travers un environnement de programmation. La deuxième comprend une triangulation entre les projets des élèves et d'autres données concernant leur apprentissage, comme leurs notes à un quiz donné ou le niveau d'accompagnement de ces élèves pendant l'apprentissage. La troisième approche se focalise sur la performance active de l'élève à travers un environnement d'évaluation. Une dernière approche se fonde sur les entretiens entre enseignants et élèves pour détecter les concepts de la pensée informatique dans la parole des élèves.

Par la suite, l'auteur met l'accent sur le cas des jeunes élèves et l'existence rare d'environnements consacrés à leur apprentissage de la programmation et à leur évaluation. Sur ce point, il fait référence aux travaux de sa directrice, Marina Umaschi Bers, qui a beaucoup travaillé sur l'apprentissage de la programmation par les jeunes élèves. Elle a prouvé que les élèves peuvent utiliser dès quatre ans des interfaces simples de programmation.

Portelance met aussi l'accent sur le système d'évaluation développé par Bers (2006, 2012) : le Positive Technological Développement (PTD). Selon cet auteur, les activités qui impliquent la technologie doivent favoriser la communication, la collaboration, la construction d'une communauté, la créativité, etc. Portelance remarque qu'il est important, lorsqu'on organise une activité d'évaluation de la pensée informatique, de prendre en compte la théorie du développement cognitif de Jean Piaget (2007), qui explique les différents stades de développement et ses caractéristiques, ainsi que la Zone Proximale de Développement (ZPD) de Lev Vygotsky (1978).

Le manque de références sur l'évaluation de l'apprentissage de la pensée informatique à travers l 'environnement ScratchJr a conduit Portelance à créer l'activité « Code and Tell » afin d'évaluer si elle peut donner aux élèves la possibilité d'apprendre la pensée informatique.

ScratchJr est une version récente du projet Scratch destinée aux enfants de 5 à 7 ans, qui a été créée par le laboratoire Média du MIT, l'université Tufts et PICO (Playful Invention Company) afin d'initier les jeunes élèves aux notions de programmation. Le projet ScratchJr comprend trois éléments :

1. l'application, qui comprend le langage de programmation et l'interface qui ont été conçus afin de correspondre au développement cognitif, personnel, social et affectif des jeunes enfants ;
2. les ressources curriculaires et les possibilités de faire des mathématiques, de la lecture et de l'écriture ;
3. les sources en ligne dédiées aux enseignants, afin de leur présenter le logiciel ScratchJr, les activités possibles qu'ils peuvent élaborer avec ce logiciel et des pratiques d'enseignement.

En utilisant ScratchJr, les enfants peuvent exprimer leur créativité en programmant des histoires et des jeux interactifs. Plus précisément, le code a été simplifié et se base sur des images sous forme d'icônes/briques. ScratchJr utilise un environnement de type « Drag and Drop » (cliquer et glisser, en français), c'est-à-dire que les enfants doivent cliquer et glisser les icônes sur l'espace de programmation afin de créer des scripts de programmation. À l'aide d'un tel environnement, l'utilisateur crée des programmes syntaxiquement corrects, c'est-à-dire qu'il ne peut pas mettre ensemble des briques qui ne sont pas créées pour être ensemble. Cet avantage diminue le sentiment de frustration qui accompagne souvent la programmation (Resnick et al., 2013).

3. Méthodologie de l'auteur

La recherche de Portelance s'est déroulée dans trois classes d'une école publique de la banlieue de Greater Boston (état du Massachusetts, Etats-Unis) [2], qui avait des relations professionnelles avec le Tufts University, proche géographiquement de l'université. Cette école était équipée de trente IPads. Soixante-six élèves ont participé à ce programme. Les enseignants de trois classes étaient présents pendant le déroulement de la recherche et ils pouvaient intervenir dès qu'ils se sentaient à l'aise.

Portelance a choisi de répondre à la question suivante :

> *Comment l'activité "Code and Tell" peut-elle être utilisée afin de donner aux jeunes élèves des possibilités d' apprendre la pensée informatique ?*

Pour répondre à cette question, Portelance a fait le choix de mener une étude exploratoire en utilisant la méthodologie « design-based research » (recherche de conception, en français). Cette méthodologie se fonde sur la conception de scénarios pédagogiques pour enseigner quelque chose dans un terrain choisi, sur l'évaluation de ces scénarios et sur leur amélioration afin de ré-intervenir sur le même terrain pour examiner à nouveau ce qui va se passer. Le recueil de données a été réalisé à l'aide d'enregistrements audio-visuels et de notes prises sur le terrain par les chercheurs.

L'auteur explique avoir utilisé aussi une approche nommée « conjecture mapping » (Sandoval, 2014) qui permet de guider la recherche de développement. L'idée est de partir d'une conjecture de haut niveau (ici, que l'activité peut aider à l'apprentissage de la pensée informatique par de jeunes élèves) qu'on aligne avec une implémentation (c'est-à-dire une mise en place sur le terrain, comprenant l'organisation de l'environnement de l'apprentissage en regroupant les outils nécessaires, la structure de l'activité et la structure de la participation). La carte de conjecture comprend aussi une description du processus de médiation, c'est-à-dire des liens entre la mise en place et les résultats visés (ici, l'utilisation de la pensée informatique au-delà des concepts sur le logiciel). Enfin, la méthodologie implique de préciser les « outcomes », c'est-à-dire les résultats qui démontrent un apprentissage réussi (ici, le développement de la pensée informatique).

Portelance a créé un curriculum d'une durée de treize séances, chaque séance durant une heure, avec deux séances par semaine. Le programme comprend trois grandes unités, car les élèves apprennent à réaliser trois genres de projets différents : collages, histoires et jeux interactifs. Chaque unité comprend deux grandes sous-unités :

- une période consacrée à créer leurs propres projets (soit collages, soit histoires, soit jeux),
- une deuxième période consacrée à participer à l'activité « Code and Tell ».

L'activité « Code and Tell » est constituée d'entretiens directifs menées par les élèves par groupes de deux, c'est-à-dire que chaque élève posait quatre questions à son binôme, dont les trois étaient prédéfinies par Portelance et la dernière était de leur propre choix. Les binômes restaient les mêmes pendant toute l'activité, sauf dans le cas d'absence ou d'autres difficultés. Les élèves ont participé trois fois au total à cette activité au cours du curriculum (jour quatre, huit et douze). En répondant aux questions données les élèves ont présenté leurs programmes d'une manière détaillée. Après la troisième participation, ils ont invité leurs familles et ils leur ont présenté leur production.

Au cours de la première unité, les élèves ont appris les bases pour créer un collage à l'aide du logiciel ScratchJr. Ensuite, ils ont consacré une séance à créer leur propre collage et puis les ont présentés à leurs camarades en répondant à leurs questions, dans le cadre de l'activité « Code and tell ». Pendant la deuxième unité, les élèves ont appris les concepts de l'interface et de programmation qui les ont aidés à réaliser une histoire avec ScratchJr. Après, ils ont présenté leur travail à leurs camarades au cours de l'activité « Code and tell ». Au cours de la troisième unité du curriculum, les élèves ont appris les différents blocs qui peuvent déclencher un programme, afin de pouvoir créer leurs propres jeux à l'aide de ScratchJr. À la fin de cette séance, ils ont présenté leur projet à leurs camarades en répondant à leurs questions.

Portelance a retenu, dans le corpus, les enregistrements audio-visuels qui étaient complets (trois enregistrements par élève) et ceux des élèves qui étaient d'accord pour qu'il les utilise. C'est pourquoi il a analysé finalement les enregistrements de trente-six élèves. L'auteur a choisi une démarche qualitative afin de pouvoir réaliser une analyse holistique et compréhensive du phénomène étudié.

4. Résultats de la recherche

4.1. Typologies des projets et trois niveaux de complexité par type de projet

L'analyse des données a conduit l'auteur a créé 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 ».

4.1.1. les projets descriptifs

En ce qui concerne les projets descriptifs, l'auteur note que la capacité à décrire quelque chose en se référant aux points les plus importants, demande un niveau d'abstraction qui est nécessaire pour la pensée informatique.

Les élèves qui décrivent leurs projets d'une manière simple, restent au niveau de la description des caractéristiques de différents éléments de leurs projets et pas de leurs rôles ou leurs fonctionnalités. Par la suite, l'auteur cite des extraits des entretiens réalisés pour que le lecteur puisse comprendre les différentes catégories détectées. Les élèves qui décrivent leurs projets d'une manière intermédiaire expliquent la situation existante entre les éléments qui constituent leurs projets. Les élèves qui décrivent leurs projets d'une manière complexe parlent des fonctionnalités individuelles de chacun des éléments qui constituent leurs projets.

4.1.2. les projets démonstratifs

La démonstration des projets implique une compréhension du fond du projet et de sa fonctionnalité, mais aussi un intérêt pour la bonne transmission de l'information. Le processus de démonstration est important pour la pensée informatique selon Portelance, car souvent, les solutions de problèmes en informatique ont une forme démonstrative.

Les élèves qui démontrent leurs projets d'une manière simple, présentent à leurs camarades les éléments de leurs projets séparément. Les élèves qui démontrent leurs projets d'une manière intermédiaire présentent à leurs camarades les différents éléments de leurs projets comme s'ils faisaient partie d'un système holistique. Enfin, les élèves qui démontrent leurs projets d'une manière complexe, présentent les différents éléments de leurs projets comme s'ils faisaient partie d'un système dynamique où des choses différentes peuvent se passer en fonction des circonstances.

4.1.3. les projets imaginatifs

L'auteur met l'accent sur les projets imaginatifs, car la capacité à penser aux limites et aux différentes possibilités d'une solution est indispensable pour la pensée informatique. L'élève qui imagine un projet, illustre ses différentes intentions. Souvent il se réfère à d'autres éléments qui pourraient faire partie de son projet ou à ce qui se passe au-delà de ce que le projet montre. Par exemple, un élève qui imagine pourrait faire référence à l'ensemble des fonctionnalités dont dispose un personnage, même s'il s'agit seulement d'un chat qui se déplace à droite dans la scène. Les élèves qui imaginent présentent leurs projets en faisant un croisement entre ce qu'ils imaginent pour leurs projets et ce que leurs projets présentent eux-mêmes.

Les élèves qui imaginent leurs projets d'une manière simple, font référence aux éléments physiques qui constituent leurs projets à l'opposition des éléments conceptuels. Les élèves qui imaginent leurs projets d'une manière intermédiaire, font attention aux éléments conceptuels du projet en plus des éléments physiques du projet. Ils peuvent parler des personnages, des arrières-plan ou des actions, mais ils font attention en particulier au concept représenté par ses éléments. Enfin, les élèves qui imaginent leurs projets d'une manière complexe font attention à l'expérience de l'utilisateur ou au joueur de leur projet. Ils ne parlent pas seulement de ce qui se passe dans la scène ou des concepts auxquels participent les composantes physiques de leurs projets. Ils mettent l'accent sur les réactions, les difficultés et l'humeur de l'utilisateur. Ces élèves-là pensent à l'utilisateur lorsqu'ils présentent leurs projets, c'est pourquoi ils les présentent d'une manière complexe.

4.2. Existence de motifs dans l'activité des élèves

L'auteur a codé les enregistrements vidéos en fonction des catégories et sous-catégories susmentionnées, afin d'examiner s'il existe des motifs au comportement des élèves tout au long de l'activité. La réalisation de cette partie de la recherche a nécessité l'utilisation d'une approche qualitative. L'auteur a analysé trois graphiques qui représentent les fréquences des élèves à présenter leurs projets descriptifs, démonstratifs et imaginatifs, d'une manière simple, intermédiaire ou complexe.

L'analyse des graphiques révèle que les motifs détectés sur les projets descriptifs et les projets démonstratifs sont semblables. Plus précisément, les descriptions et les démonstrations sont simples le plus souvent pour les collages, intermédiaires le plus souvent pour les histoires, et complexes le plus souvent pour les jeux.

Concernant les projets imaginatifs, les résultats de l'analyse paraissent plus intéressants pour Portelance. La moitié de l'échantillon présente un niveau de complexité simple. Les projets imaginatifs sont intermédiaires le plus souvent sur les collages, alors qu'ils paraissent complexes sur les histoires et les jeux.

Conclusion et perspectives

L'auteur clôt le chapitre des résultats en indiquant trois profils qui émergent de l'analyse : l'élève qui décrit, l'élève qui démontre et l'élève qui imagine. Bien que ces modèles ne correspondent pas à tous les élèves de l'échantillon, ils constituent trois chemins différents d'apprentissage qui sont liés à la pensée informatique et que les enseignants peuvent rencontrer potentiellement dans leurs classes. Cela montre, selon Portelance, la pléthore de manifestations possibles de développement de la pensée informatique. L'auteur souligne que l'activité « Code and Tell », outre qu'elle met en lumière le pluralisme de l'apprentissage de la pensée informatique, a favorisé également le développement de ce type de pensée.

Enfin, l'auteur fait référence aux limites et aux perspectives de cette recherche, en mettant l'accent sur le fait que l'activité « Code and Tell » peut être utilisée comme outil pour évaluer la pensée informatique même si elle n'était pas prévue pour cela, comme les portfolios des projets et les problèmes à résoudre de type « Solve It », méthodes déjà présentes en littérature.

Références bibliographiques

Bers, M. U. (2006). The role of new technologies to foster positive youth development. *Lawrence Erlbaum*, *11*(1), 58.

Bers, M. U. (2012). *Designing Digital Experiences for Positive Youth Development : From Playpen to Playground* (1 edition). Oxford ; New York : Oxford University Press.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the developement of computational thinking (p. 25). Présenté à AERA. Consulté à l'adresse http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Papert, S. A. (1993). *Mindstorms : Children, Computers, And Powerful Ideas* (2 edition). New York : Basic Books.

Piaget, J. (2007). *The Child's Conception of the World*. Rowman & Littlefield.

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

Rushkoff, D. (2011). *Program or Be Programmed : Ten Commands for a Digital Age* (1st edition). Berkeley, CA : Soft Skull Press.

Vygotsky, L. (1978). *Mind in Society - Development of Higher Psychological Processes* (New Ed). Harvard University Press.

Wing, J. M. (2006). Computational thinking. *Commun. ACM*, *49*(3), 33–35.

http://doi.org/10.1145/1118178.1118215

[1] ScratchJr est une version récente du projet Scratch destinée aux enfants de 5 à 7 ans, qui a été créée par le laboratoire Média du MIT, l'université Tufts et PICO (Playful Invention Company) afin d'initier les jeunes élèves aux notions de programmation

[2] Le niveau considéré est équivalent au CE1 dans le système éducatif français