Apprentissage de la programmation informatique à l'école

Apprentissage de la programmation avec le micromonde Pixel'Art : analyse de la conceptualisation de la boucle par des élèves de cycle 3.

RÉSUMÉ

L'application PixelArt - qui permet aux élèves de programmer les actions d'un curseur (déplacement, orientation et coloration) afin qu'il produise un dessin - est utilisée pour l'apprentissage de la programmation au cycle 3. Il est destiné à travailler les premières notions d'informatique, la boucle (ou répétition) en particulier. On présente dans cet article une analyse de l'activité d'élèves pour comprendre la manière dont ils construisent des solutions, à quel moment ils prennent conscience de l'intérêt d'utiliser la boucle et comment ils appréhendent cet usage. Cette étude qualitative mobilise les cadres théoriques de la problématisation et de la genèse instrumentale. La méthodologie combine une analyse des échanges langagiers des élèves et une analyse des logs enregistrés au cours de l'activité de programmation.

Michael **ZEYRINGER**
Master MEEF
Mention PIF
Parcours Enseignement
Expertise et Apprentissage
Inspé Académie de Nantes

MOTS CLÉS :

apprentissage de la programmation, analyse de l'activité des élèves, répétition, boucle, genèse instrumentale, problématisation

Dans les programmes de 2015 l'apprentissage de la programmation informatique apparaît dès le cycle 2 (Ministère de l'Education Nationale, 2015). L'Académie des Sciences (2013) propose trois notions de base à appréhender dès l'école primaire pour écrire un programme informatique : la séquence, le test (instruction conditionnelle) et la boucle (itération ou répétition). D'après Fabienne Viallet et Patrice Venturini (2010) l'appréhension de la notion de boucle représente une réelle difficulté dans l'apprentissage de la programmation. Nous avons fait le choix de nous concentrer, dans le cadre d'une étude qualitative, sur l'analyse de la conceptualisation de la boucle par des élèves de cycle 3 au cours de leurs activités avec l'application Pixel'Art[1], dans le cadre d'une séquence d'apprentissage de la programmation.

Nous présentons dans cette étude une association de deux cadres théoriques, celui de l'apprentissage par problématisation et celui de la genèse instrumentale, qui nous permet d'envisager l'analyse des processus de conceptualisation des élèves, tournés à la fois vers le problème et vers l'instrument. En nous appuyant sur ces cadres théoriques nous tentons dans cette étude de construire et mettre en œuvre de manière exploratoire une méthodologie innovante.

1. Christophe Declercq est le concepteur de l'application (Declercq & Tort, 2018), adaptée pour les besoins que nous avons définis pour l'expérimentation. L'application web PixelArt3, développée en Javascript côté client, est diffusée sous licence libre et est disponible au téléchargement à l'adresse : https://gitlab.univ-nantes.fr/declercq-c/PixelArt3

DIDACTIQUE DE L'INFORMATIQUE ET CADRES THÉORIQUES

Un état de l'art effectué par Fabienne Viallet et Patrice Venturini (2010) met en évidence l'absence de cadre théorique propre à la didactique de l'informatique pour analyser l'activité des élèves. C'est le concept de genèse instrumentale, théorisé par Rabardel, qui semble susciter un intérêt plus partagé parmi les didacticiens de l'informatique (Nijimbere, 2013). Cependant d'après Baron et Bruillard le cadre théorique de la genèse instrumentale « ne prend pas en compte de manière spécifique les situations d'enseignement », il est également nécessaire d'envisager les dynamiques sociales dans la recherche (Baron & Bruillard, 2001). La proposition de Christian Orange (1990) au sujet de la didactique de l'informatique nous a aidé dans notre cheminement : « ainsi pour analyser de manière critique la didactisation, le didacticien doit-il travailler dans trois directions, correspondant à trois registres différents : les registres épistémologique, psychologique et pédagogique ».

La boucle est l'un des premiers objets de savoir construit en informatique (Viallet & Venturini, 2010). Elle a été choisie, à l'issue d'un débat durant les années 65-70, comme étant la structure de contrôle à privilégier pour réaliser des itérations, afin de permettre une écriture plus lisible des programmes ainsi que leur vérification (preuves de programmes). Cette pratique est justifiée notamment par la capacité de l'ordinateur à effectuer des tâches de manière répétitive. La boucle constitue donc un objet de savoir tout à fait légitime et pertinent à enseigner.

En psychologie cognitive, les travaux de Schwill (Schwill, 2001) et Mendelsohn (Mendelsohn, 1985) contribuent d'une part à démontrer la possibilité d'appréhender des concepts de base de l'informatique dès l'école primaire et nous fournissent d'autre part des outils pour l'analyse des réussites et des difficultés des élèves dans des tâches de programmation. Schwill conclut que malgré le peu de résultats empiriques il est probable que la méthode utilisée par les enfants pour l'identification de motifs répétés (suites) soit généralement intuitive à partir de 5 ans dans des situations simples.

Dans le registre pédagogique, Seymour Papert a développé une théorie de l'apprentissage, le constructionnisme (en appui sur le constructivisme de Piaget), pour l'analyse l'activité de l'enfant face à l'ordinateur (Papert, 1980). Dans le cadre de notre étude nous rejoignons Lagrange et Rogalski (2015) qui proposent de « rompre

avec le constructivisme naïf dénoncé par Crahay, où partant de situations très ouvertes, l'élève est finalement conduit par l'enseignant pour l'écriture d'un programme. ». Nous souhaitons nous inscrire dans une approche socio-constructiviste. Nous émettons l'hypothèse que les interactions entre pairs et enseignant-élèves font avancer la construction du savoir en jeu et nous postulons que l'élève ne peut apprendre en informatique sans être confronté à des problèmes de programmation. Nous nous inscrivons donc dans une pédagogie de la construction du problème, « soucieuse de déployer complètement le processus de problématisation » (Fabre, 1999), c'est pourquoi nous utilisons également le cadre théorique de la problématisation (ibidem) (Orange, 2012) dans notre étude.

Nous postulons que l'élève ne peut apprendre en informatique sans être confronté à des problèmes de programmation.

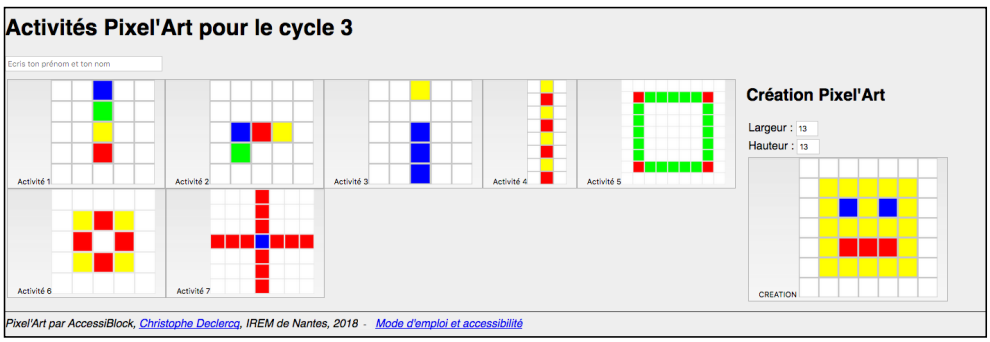
2. Grammaire d'un langage.

3. « La sémantique d'un langage de programmation est liée à la signification que l'on attache aux différentes constructions syntaxiques effectuées dans son environnement » (Komis V. , 2016).

CHOIX D'UN ENVIRONNEMENT DE PROGRAMMATION ET DÉFINITION DU PROBLÈME À TRAITER PAR LES ÉLÈVES

Malgré une interface souvent attrayante les langages proposés posent des questions didactiques, comme l'analyse Vassilis Komis, et des questions restent ouvertes sur « les problèmes didactiques sur les connaissances syntaxiques[2], sémantiques[3] et stratégiques auxquels nous devons faire face et leurs origines » (Komis, 2016). Nous avons fait le choix d'utiliser une application, Pixel'Art, qui constitue un environnement de programmation avec un bagage d'instructions réduit et la possibilité de traiter des tâches mettant en jeu un grand nombre de répétitions, incitant à l'usage de la boucle. Cet environnement a été construit en extrapolant une situation du concours Castor, le robot peintre, identifiée comme permettant l'acquisition du concept de boucle par les élèves (Drot-Delange & Tort, 2018).

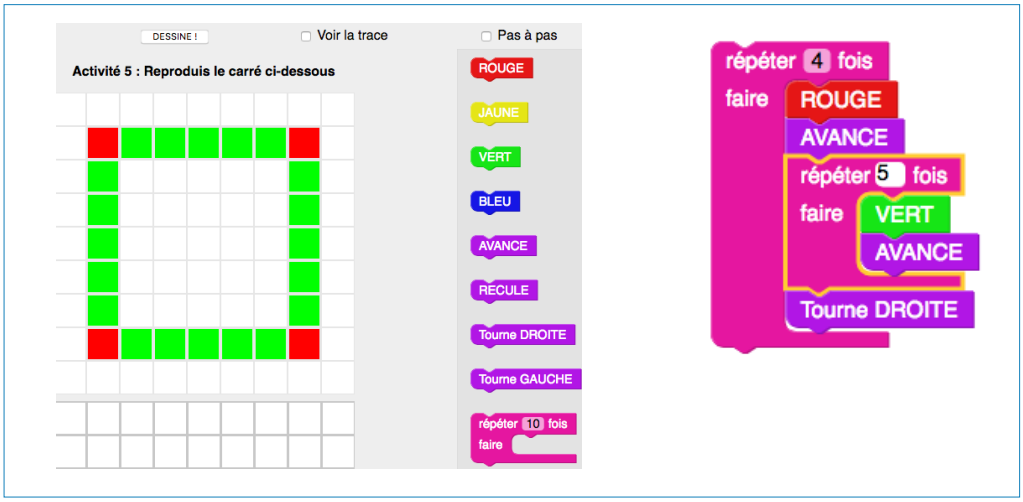
FIGURE N°1
Progression des tâches proposées aux élèves dans le micromonde Pixel'Art



Nous constituons ainsi un micromonde (Papert, 1980) qui selon Cédric Libert et Wim Vanhoof « favorise chez les élèves la découverte et l'assimilation de nouveaux concepts » (Libert & Vanhoof, 2017). Nous utilisons la démarche présentée par Libert et Vanhoof où l'enseignant met l'élève face à un « problème difficile à résoudre » (ibidem) en ne disposant que des instructions qu'il aura manipulées lors de la résolution préalable de problèmes plus simples. La théorie de la genèse instrumentale nous conforte dans ce choix, l'usage d'un micromonde va dans le sens d'un

« contrôle de l'ouverture du champ des actions possibles, comme de l'activité requise (qui) constituent donc deux dimensions importantes de l'usage éducatif des instruments » (Rabardel, 1995).
La tâche à traiter par les élèves est de faire reproduire au curseur, en le programmant, un dessin dans l'application Pixel'Art. La tâche numéro 5 présentée ci-dessous vise à rendre nécessaire l'usage de la boucle dans la construction du programme.

FIGURE N°2
Tâche 5 et solution « experte »



LE CADRE THÉORIQUE DE LA PROBLÉMATISATION

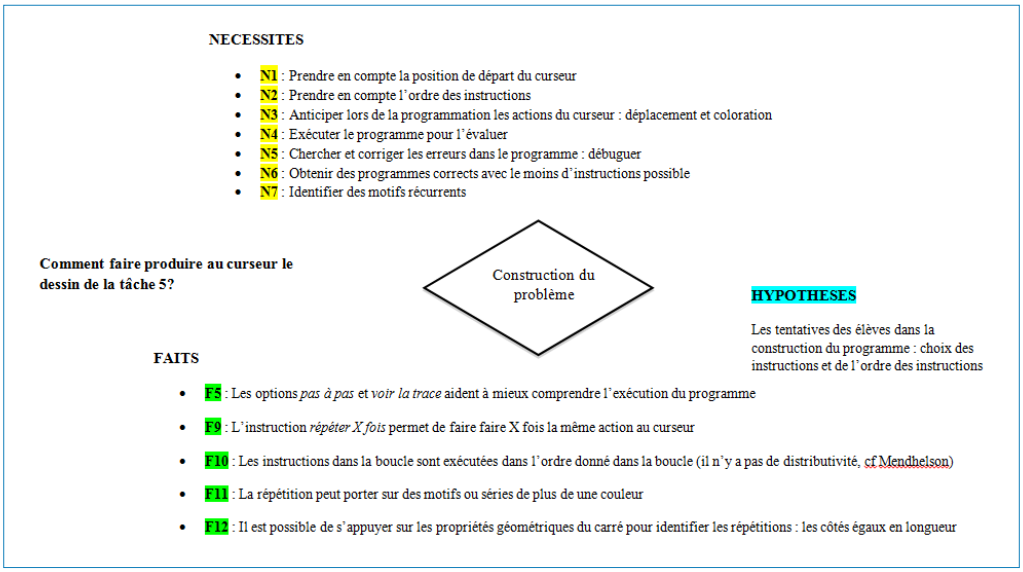
C'est en se référant à La théorie de l'enquête de Dewey et au Rationalisme appliqué de Bachelard que Fabre et Orange mettent le problème au centre de l'apprentissage. Dewey comme Bachelard définissent une théorie de la construction et non pas seulement de la résolution des problèmes (Fabre, 2005). Un problème est une expérience qui résiste, où l'on n'a pas accès directement à la solution. Il est constitutif du savoir. D'après Fabre et Musquer, problématiser c'est « développer un questionnement visant à identifier les données et les conditions du problème et à les mettre en tensions. De cette interaction résultent des hypothèses de solutions qui seront ensuite validées ou non. » (Fabre & Musquer, 2009). Nous préférons utiliser le mot « faits » plutôt que « données » ou « informations » utilisés par Fabre et Orange. En effet, ces mots ont une signification précise en informatique, l'emploi de ces termes pour la problématisation appliquée dans un contexte d'enseignement de l'informatique nous paraît être source de confusions. Le mot « fait » nous semble bien rendre compte du caractère assertorique.

A l'opposé, les conditions du problème ou nécessités ont un caractère apodictique, d'universalité. Dans le cas que nous allons étudier, les faits seront dépendants du contexte particulier de la séance d'apprentissage et donc fortement liés à l'environnement de programmation choisi. Les nécessités qui émergeront seront au contraire indépendantes de l'environnement de programmation et constituent les règles à suivre pour aboutir à la solution. Les tentatives des élèves dans la construction de leurs programmes représenteront les hypothèses de solutions, elles seront validées ou non lors de l'exécution des programmes. Selon Rogalski cela présente une difficulté pour l'élève: « une propriété difficile à intégrer [...] est le caractère différé d'une exécution du programme » (Rogalski, 2015).

C'est au cours des débats et des échanges argumentatifs entre les élèves et entre les élèves et l'enseignant que la problématisation devient appréhendable. D'après Orange, ces échanges ont une fonction fondamentale : « ils permettent d'explorer et de délimiter le champ des possibles et de repérer ainsi des conditions de possibilité des solutions, ce que nous appelons des nécessités. » (Orange, 2012). Un indicateur de la qualité de la problématisation est donc l'identification et l'explicitation des nécessités.

Les observations empiriques faites dans d'autres contextes avec d'autres environnements de programmation nous aident à anticiper les faits pour notre situation. En ce qui concerne les nécessités en jeu dans cette situation, elles sont formulées à partir des éléments que nous postulons comme incontournables pour écrire un programme correct. Une analyse épistémologique plus fine permettrait de préciser les nécessités. Le losange de problématisation ci-après modélise a priori la problématisation. Le codage proposé est ensuite utilisé dans le cadre de l'analyse du corpus.

FIGURE N°3
Losange de problématisation a priori relatif à la tâche 5



LA GENÈSE INSTRUMENTALE

D'après Pierre Rabardel « l'instrument n'est pas conceptuellement neutre, qu'il soit issu des technologies contemporaines ou traditionnelles » (Rabardel, 1995). L'instrument est construit par l'utilisateur de l'artéfact[4] au cours de l'activité qui a lieu avec l'artéfact, c'est la genèse instrumentale.

L'instrumentation désigne ce qui dans le processus de genèse instrumentale est orienté vers le sujet lui-même. L'instrumentalisation désigne les processus dirigés vers l'artefact, comme par exemple la sélection ou la production de fonctions, l'attribution de propriétés, le détournement voire la transformation de l'artefact. Lorsque l'utilisateur s'approprie un artéfact il élabore progressivement des représentations relatives à l'instrument. Il construit également progressivement des schèmes d'utilisation qui permettent la répétition de l'action et son adaptation à des situations différentes. Un instrument est donc constitué à la fois d'un artéfact et des schèmes d'utilisation que l'utilisateur lui a associé progressivement lors de la genèse instrumentale.

Nous proposons, pour rendre compte de la genèse instrumentale, d'utiliser

4. Selon Rabardel, un artéfact est « donc tout objet technique ou symbolique ayant subi une transformation d'origine humaine, si petite soit-elle. Il donne l'exemple du dispositif de pilotage du bras manipulateur d'un petit robot qui déplace des objets dans l'espace. De ce point de vue, l'appellation d'artéfact est directement mise en relation avec toute action ou activité humaine » (Nijimbere, 2013). L'artéfact n'est pas nécessairement un dispositif matériel.

en complément du modèle SAI (Rabardel, 1995) mobilisé pour l'analyse de l'usage des instructions et commandes (figure 4) une catégorisation de la posture des élèves décrite par Declercq et Tort (Declercq & Tort, 2018) : le programmeur « pas à pas » et le programmeur. Le programmeur « pas à pas » aurait régulièrement recours aux commandes, notamment dessin, pas à pas et voir la trace. Le programmeur, lui, anticiperait davantage et n'utiliserait l'évaluation de son programme qu'en fin de parcours.

FIGURE N°4
En quoi les logs nous renseignent-ils sur la genèse instrumentale ?

Codage	Activité	Instrument	Objet	Logs
G1	L'élève programme le déplacement du curseur.	Instructions *avance* et *recule*	Déplacement	Création de blocs **dessin_avance dessin_recule**
G2	Il programme le changement d'orientation du curseur.	Instructions *tourne à droite* et *tourne à gauche*	Changement d'orientation	Création de blocs **dessin_droite dessin_gauche**
G3	Il programme la coloration d'une case.	Instructions *rouge*, *jaune*, *bleu*, *vert*	Coloration d'une case	Création d'un bloc **dessin_bleu dessin_jaune dessin_rouge dessin_vert**
G4	Il programme la répétition d'une action ou d'une série d'actions.	Instruction *répéter X fois*	Répétition X fois	Création d'un bloc **controls_repeat**
G5	Il observe et analyse l'exécution du programme.	Commandes *pas à pas et voir la trace*	Temps de pause entre chaque action (déplacement, orientation, coloration) Trace de l'orientation du curseur dans chaque case	**Dessine pas à pas**
G6	Il débogue ou modifie le programme.	Suppression d'instructions	Modification des actions programmées précédemment	**Destruction d'un bloc**
G7	Il évalue son programme.	Commande *dessine*	Exécution du programme	**Dessine**

MÉTHODOLOGIE

Recueil de données

Il s'agit de rechercher des indices pouvant témoigner de la genèse instrumentale et de l'avancée dans la problématisation par les élèves, à la fois lorsqu'ils travaillent sur l'ordinateur et au moment des échanges au sujet des problèmes qui se posent à eux. Nous disposons de quelques captures vidéo exploitables de l'activité de deux binômes de deux classes de cycle 3 filmés sur la totalité des deux séances et de moments d'échanges en grand groupe (mise en commun, recherche d'une solution).

En complément de la captation vidéo, nous avons appareillé le micro-monde PixelArt pour enregistrer l'horodatage de toutes les interactions de l'élève avec le système : ajout d'instructions, exécutions, passage d'une activité à l'autre, suppression d'instruction. Le traçage des logs a été paramétré pour pouvoir observer la création et la destruction des blocs d'instructions ainsi que l'activation des commandes (dessine et pas à pas, mais pas voir la trace). L'activité de six binômes a ainsi pu être enregistrée. Les informations obtenues ne permettent cependant pas d'observer l'ordre dans la construction des séquences d'instructions, c'est à dire l'imbrication des blocs d'instructions. Il n'est pas non plus possible de savoir quel bloc a été supprimé.

Ce que nous ne pouvons pas observer par le biais des logs, ce sont les difficultés que les élèves rencontrent dans l'usage de la boucle, par exemple en ce qui concerne l'ordre des instructions à l'intérieur de la boucle. La

multiplication de création de blocs de boucle indique que les élèves sont aux prises avec un problème. Lorsque ce cas se présente pour un binôme qui a été filmé l'information obtenue par les logs nous aide à pointer dans la captation vidéo la séquence à analyser.

FIGURE N°5
Exemple d'une suite de logs suggérant que les élèves sont en prise avec un problème lié à la répétition (classe 1, binôme 1, séance 2)

09:19:08 : Dessine
09:19:28 : Création d'un bloc dessin_avance
09:19:31 : Création d'un bloc dessin_vert
09:19:43 : Création d'un bloc controls_repeat
09:19:55 : Destruction d'un bloc
09:19:57 : Destruction d'un bloc
09:20:08 : Dessine
09:20:41 : Création d'un bloc controls_repeat
09:20:56 : Dessine
09:21:14 : Destruction d'un bloc
09:21:37 : Création d'un bloc controls_repeat
09:24:10 : Création d'un bloc controls_repeat
09:24:20 : Destruction d'un bloc
09:24:36 : Dessine

ANALYSE DES DONNÉES

Nous opérons de deux façons pour cibler les passages à transcrire. L'analyse des fichiers de logs générés après les sessions de programmation des binômes filmés nous aide à cibler des moments où les élèves utilisent la boucle ou bien sont confrontés à un problème. Cette démarche est utilisée lorsque le corpus pour l'activité enregistrée est particulièrement long, elle permet de se concentrer sur les passages qui nous intéressent en priorité, c'est à dire ceux où les élèves manipulent la boucle. Lors de la visualisation des vidéos nous repérons les moments où les échanges entre élèves ou entre les élèves et l'enseignant présentent un intérêt potentiel en termes de problématisation ou de genèse instrumentale. Nous analysons ensuite les transcriptions pour identifier les éléments qui témoignent d'une problématisation ou de la genèse instrumentale. L'analyse des transcriptions peut nous amener à effectuer un retour sur les fichiers de logs pour valider ou invalider nos hypothèses.

> Nous repérons les moments où les échanges entre élèves ou entre les élèves et l'enseignant présentent un intérêt potentiel en termes de problématisation ou de genèse instrumentale..

FIGURE N°6
Exemple de traitement du corpus issus des captations vidéo

		F	N	GI
K14	Mets efface.			
W14	Pas toute de suite...(W entève les blocs qui étaient dans la boucle et les met sur le côte droit) On va les garder pour...bah pour qu'on puisse se corriger. (W finit par remettre la boucle autour des instructions, toujours sur le côté droit de l'écran). Donc efface.		N5	+
Chœur	Avance, rouge, avance, vert, avance, vert, avance, vert, avance, vert, avance, vert...			G1 G3
W15	Tourner à droite et c'est tout. Donc ça (les blocs gardés à droite de l'écran), on le met à la poubelle. On fait répéter quatre fois. Et normalement ça y est.			G2 G4
K15	(Le résultat est le même, les cases rouges sont décalées) Ahhh !... (Temps d'observation) Au moins on est pareil...			G7

De plus, deux indicateurs peuvent être calculés à partir de l'enregistrement des logs. Le premier indicateur, que nous avons nommé taux d'anticipation est le rapport entre le nombre d'instructions et le nombre total d'instructions et de commandes. Proche de 1, il indique que l'élève a anticipé la totalité du programme avant de l'exécuter une seule fois. Plus l'indicateur se rapproche de 0, plus il témoigne de nombreuses tentatives d'exécution ou d'hésitations : ajout ou suppression d'instructions.
Le second indicateur, que nous avons nommé taux d'efficacité est le rapport entre le nombre minimal d'instructions pour résoudre le problème, et le nombre total d'instructions utilisé par l'élève. Plus ce taux est proche de 1, plus la solution de l'élève, à condition qu'elle soit correcte, a été anticipée et s'approche de la solution optimale. Un taux faible peut témoigner, en particulier, d'une longue séquence d'instructions identiques, qui aurait pu être remplacée par une répétition. L'évolution de ces deux taux durant les activités de programmation met en évidence la progression de la genèse instrumentale.

RÉSULTATS

Tous les binômes observés ont réussi à écrire un programme correct utilisant au moins une boucle. Nous remarquons que malgré un repérage parfois précoce de la répétition c'est sur le placement des instructions dans et en dehors de la boucle que les élèves concentrent leur action. Le repérage de motifs récurrents, même s'il peut être mis en œuvre par de jeunes enfants (Schwill, 2001), n'a pas été effectué de manière systématique voir même pas du tout par les élèves observés.
L'activité de problématisation a lieu autour de quatre nécessités identifiées : le respect de l'ordre des instructions (N2), l'exécution du programme (N4), sa correction (N5) et son optimisation qui mène à l'usage de la boucle (N6). Des nécessités et faits identifiés a priori n'ont donc pas été évoqués par les binômes durant leurs activités de programmation, en particulier la nécessité d'anticiper les actions du curseur et celle de repérer des motifs récurrents. Dans le corpus que nous avons traité la nécessité d'anticiper n'est évoquée que par l'enseignant (PE7) lors d'une mise en commun à l'issue de la deuxième séance. Par contre ce qui avait été conçu pour aider au débuggage, le retour instrumental, est ici utilisé par les élèves dans une réelle méthode de conception (exprimé par W2 et confirmé par l'analyse de l'activité du binôme).

E1	(inaudible) Nous on a essayé de faire plus court, et en fait faut faire répéter deux fois.
PE6	Voilà. I ?
I1	Moi je trouve que l'autre était assez compliqué parce que fallait bien regarder les couleurs et un moment on s'est trompé…(inaudible).
PE7	Etre attentif à ce qu'on demande au curseur et il faut anticiper le plus possible son déplacement…sinon on se trompe. D'accord ? W ?
W2	Nous notre technique c'est faire étape par étape, déjà voir si ça marche. Et utiliser le euh, surtout le voir la trace,…

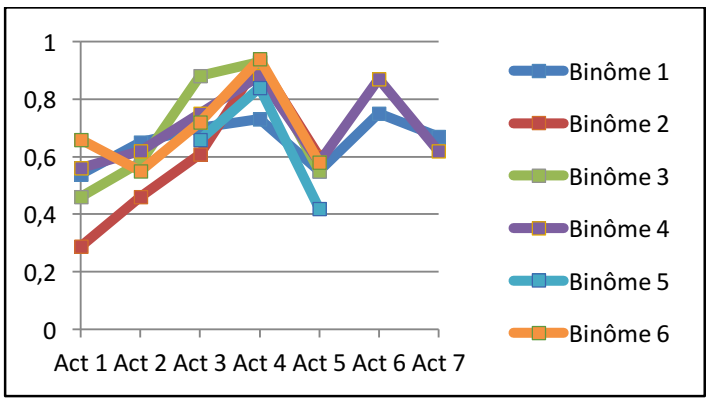
Si l'on confronte le moment d'apparition de la première boucle dans les programmes conçus aux taux d'anticipation et d'efficacité obtenus des binômes on ne peut pas considérer qu'une apparition précoce de la boucle dans la programmation soit un indicateur pour caractériser la posture des élèves.

FIGURE N°7
Premiers usages de la boucle par les binômes mis en relation avec les taux d'anticipation et d'efficacité pour la tâche 5

	Binôme 1	Binôme 2	Binôme 3	Binôme 4	Binôme 5	Binôme 6
1ère apparition de la boucle	2 min 29 18re instuction	2 min 16 4re instuction	50 sec 2re instuction	58 sec 3re instuction	5 min 20 21re instuction	24 sec 1re instuction
Taux anticipation	0,55	0,6	0,55	0,58	0,42	0,58
Taux efficacité	0,15	0,06	0,12	0,13	0,12	0,12

Ainsi le binôme 1 ne réalise une boucle qu'au bout de la 18e instruction programmée et obtient pourtant le meilleur taux d'efficacité pour cette tâche. La perception rapide de la possibilité de l'usage de la boucle n'est donc apparemment pas en lien direct avec la capacité des élèves à traiter de manière efficace les problèmes relatifs à son usage.
On peut néanmoins distinguer deux approches, descendante (binômes 2, 3, 4 et 6) et ascendante (binômes 1 et 5). L'approche ascendante, évoquée plus haut (par W2 dans le corpus), consiste à construire le corps de la boucle, le tester, puis l'insérer dans une boucle. Ce qui, à première vue, pourrait sembler être un manque d'anticipation se révèle donc être une stratégie tout à fait pertinente d'un programmeur « par étapes ».

FIGURE N°8
Evolution du taux d'anticipation chez les 6 binômes



Nous observons une progression du taux d'anticipation de tous les binômes depuis la tâche 1 (où l'usage de la boucle n'était pas pertinent) jusqu'à la tâche 4. Pour les tâches 5, 6 et 7 incitant à l'usage de la boucle le taux d'anticipation baisse mais reste néanmoins supérieur à celui obtenu au début du scénario. Le temps passé par les binômes sur chaque tâche est variable, ce qui explique qu'en fin de compte seuls deux binômes aient eu le temps de s'engager dans les sept tâches proposées en deux séances.

DISCUSSION

La mobilisation combinée des deux cadres théoriques, problématisation et genèse instrumentale, nous paraît offrir des résultats encourageants pour l'analyse de l'activité de l'élève lorsqu'il construit un programme sur un ordinateur. Néanmoins un travail théorique plus approfondi de didactique comparée sur l'association des deux cadres et les éventuelles relations entre leurs concepts nous paraît indispensable. A notre connaissance un tel effort a déjà été entamé en didactique de l'EPS (Lebouvier, 2015).

La mobilisation combinée des deux cadres théoriques, problématisation et genèse instrumentale, nous paraît offrir des résultats encourageants pour l'analyse de l'activité de l'élève

La méthodologie employée peut être améliorée. La capture des productions finales et intermédiaires des élèves nous paraît maintenant incontournable pour analyser l'activité des élèves. Enfin une évolution du programme d'enregistrement des logs, notamment pour prendre en compte l'usage de voir la trace et permettre une identification des blocs supprimés, donnerait des indicateurs supplémentaires.

Les postures de programmeur et de programmeur pas à pas (Declercq et Tort, 2018) sont pertinentes pour analyser l'activité des élèves dans des tâches de programmation relativement simples, où l'on peut raisonnablement attendre des élèves qu'ils produisent un programme en une seule fois, c'est à dire avec une anticipation optimale. Pour des tâches plus complexes la description du profil programmeur ne semble pas satisfaisante. On ne peut dans ce cas attendre des élèves qu'ils proposent un programme correct en un seul essai. La question serait donc de définir plus précisément ce qu'est une démarche par essais-erreurs, propre au programmeur pas à pas et en quoi elle se différencie d'une stratégie qui nous semble plus évoluée, celle de procéder « par étapes ».

CONCLUSION

La pensée algorithmique est un concept difficile à appréhender. Libert et Vanhoof remarquent que « peu d'articles décrivent des résultats empiriques concernant le développement de la pensée informatique grâce à la programmation ». Selon eux, ces articles « concluent généralement qu'il n'y a pas d'amélioration de la capacité à résoudre des problèmes avec la capacité à programmer » (Libert & Vanhoof, 2017). C'est peut-être en avançant sur l'identification et la définition de ces postures (programmeur « pas à pas », programmeur « par étapes », programmeur) et des stratégies qui leur sont propres que la question de la pensée algorithmique[5] pourra être davantage précisée, en particulier dans son développement dans un cadre scolaire[6].

Les résultats obtenus avec les outils que nous avons mobilisés et développés (cadres théoriques, méthodologie et ressources pédagogiques avec l'application Pixel'Art) nous paraissent encourageants et ouvrent des perspectives diverses au niveau de la recherche, en particulier sur la question du développement de la pensée algorithmique ■

5. Les termes « pensée informatique » et « pensée computationnelle » sont également utilisés dans la littérature.

6. L'Angleterre a introduit l'enseignement de l'informatique dans les programmes scolaires quelques années avant la France, des propositions de contenus ont été élaborées qui décrivent ce que peut être la pensée algorithmique (Curzon, P., Dorlin, M., Ng, T., Selby, C., & Woollard, J., 2014).

BIBLIOGRAPHIE

Académie des Sciences. (2013). L'enseignement de l'informatique en France : il est urgent de ne plus attendre.

Baron, G-L., & Bruillard, E. (2001). Une didactique de l'informatique ? *Revue Française de Pédagogie, n°135*.

Curzon, P., Dorlin, M., Ng, T., Selby, C., & Woollard, J. (2014). *Developing computational thinking in the classroom : a framework*. Computing at School.

Declercq, C., & Tort, F. (2018). Organiser l'apprentissage de la programmation au cycle 3 avec des activités guidées et/ou créatives. In RJC EIAH 2018. Besançon, France. Consulté sur https://hal.archives-ouvertes.fr/hal-01765408

Drot-Delange, B., & Tort, F. (2018). Concours Castor, ressource pédagogique pour l'enseignement de l'informatique ? Etude exploratoire auprès d'enseignants. In Didapro 7 - didastic.

Fabre, M. (1999). *Situations-problèmes et savoir scolaire*. Paris: Presses Universitaires de France.

Fabre, M., & Musquer, A. (2009). Les inducteurs de problématisation. *Les Sciences de l'éducation - Pour l'ère nouvelle, 42*, pp. 111-129.

Fabre, M. & Orange, C. (1997). Construction des problèmes et franchissement d'obstacles. *ASTER Obstacles : travail didactique, n°24*, Institut national de recherche pédagogique, Paris

Komis, V. (2016). Une analyse cognitive et didactique du langage de programmation Scratch Jr. Didapro 6 - DidaSTIC.

Lebouvier, B. (2015). Expérience et problématisation en EPS, une étude en course de relais, *Carrefours de l'éducation 2015/2 (n° 40)*, p. 31-49. DOI 10.3917/cdle.040.0031

Libert, C., & Vanhoof, W. (2017). La programmation par passage de messages pour aider à développer la pensée informatique. Dans J. Henry, A. Nguyen, & E. Vandeput, *L'informatique et le numérique dans la classe. Qui, quoi, comment ?* (pp. 123-133). Namur: Presses Universitaires de Namur.

Mendelsohn, P. (1985). L'analyse psychologique des activités de programmation chez l'enfant de CM1 et CM2. Enfance. L'ordinateur et l'écolier : recherches expérimentales. , 38, pp. 213-221.

Ministère de l'Education Nationale. (2015). Programmes d'enseignement du cycle des apprentissages fondamentaux (cycle 2), du cycle de consolidation (cycle 3) et du cycle des approfondissements (cycle 4).

Orange, C. (1990). Didactique de l'informatique et pratiques sociales de référence. *Bulletin de l'EPI, 60*.

Orange, C. (2012). *Enseigner les Sciences. Problèmes, débats et savoirs scientifiques en classe*. Bruxelles : De Boeck.

Papert, S. (1980). *Jaillissement de l'esprit, Ordinateurs et apprentissages*. Flammarion.

Rabardel, P. (1995). *Les Hommes et les technologies; approche cognitive des instruments contemporains*. Armand Colin. Consulté sur https://hal.archives-ouvertes.fr/hal-01017462

Rogalski, J. (2015). Psychologie de la programmation, didactique de l'informatique. Déjà une histoire... In Informatique en éducation : perspectives curriculaires et didactiques. Presses universitaires Blaise-Pascal.

Schwill, A. (2001). Ab wann kann man mit Kindern Informatik machen? / Eine Studie über informatische Fähigkeiten von Kindern. 9. GI-Fachtagung Informatik und Schule (pp. 13-30). INFOS.

Xinogalos, S. (2012). An Evaluation of Knowledge Transfer from Microworld Programming ti Conventionnal Programming. *Journal of Educational Computing Research, n°47*.