Élaboration d'un modèle praxéologique de connaissances algorithmiques à partir d'un objet-frontière avec l'algèbre : Le pattern

Gaëlle Walgenwitz

Année universitaire 2022/2023

Master ***Métiers de l'enseignement, de l'éducation et de la formation***

Mention Pratique et Ingénierie de la Formation

Parcours : Didactique des sciences et du numérique

Élaboration d'un modèle praxéologique de connaissances algorithmiques à partir d'un objet-frontière avec l'algèbre : Le pattern

Présenté par Gaëlle Walgenwitz

Mémoire de M2 encadré par Emmanuel Beffara et Marie Caroline Croset

TABLE DES MATIERES

TABLE DES ILLUSTRATIONS

AVANT-PROPOS

Remerciements

Je remercie tout particulièrement mes tuteurs, Marie Caroline Croset et Emmanuel Beffara, pour l'opportunité qu'ils m'ont offerte en acceptant de m'encadrer, pour leur disponibilité et leurs conseils tout au long du stage. J'ai apprécié leurs encouragements et leur soutien pendant cette passionnante expérience de la recherche, ainsi que celle plus coûteuse de la rédaction du mémoire. Leur confiance en mes capacités à élaborer une problématique et à mener les recherches pour tenter d'y répondre a été précieuse.

Je souhaite également remercier l'équipe MeTAH pour son accueil chaleureux et les conseils partagés. J'ai apprécié de plonger dans cet environnement pendant quelques mois et de partager l'ambiance d'un laboratoire de recherche.

Je tiens à exprimer ma gratitude au groupe "Informatique de l'école au lycée" de l'IREM de Grenoble pour m'avoir permis de les rejoindre il y a trois ans. Le travail accompli ensemble m'a permis de gagner en confiance et en compétences, ce qui m'a encouragée à me lancer dans ce master.

Je remercie également Christophe Declercq et Marielle Léonard d'avoir accepté de faire partie du jury.

Enfin, je suis reconnaissante envers mon mari et mes fils de m'avoir encouragée et soutenue dans ce projet ambitieux de reprise d'études.

1. INTRODUCTION

Depuis la réforme de l'enseignement au collège de 2015, les programmes de mathématiques du cycle 3 et 4 prévoient une initiation des élèves à l'algorithmique et à la programmation notamment en la développant par une démarche de projet quelques programmes simples, sans viser une connaissance experte et exhaustive d'un langage ou d'un logiciel particulier (Gaubert-Macon, 2022).

Les documents d'accompagnement au programme de mathématiques de cycle 4 donne comme objectif de l'enseignement de l'algorithmique et de la programmation de permettre aux élèves d'acquérir des méthodes qui construisent la *pensée algorithmique* et de développer les compétences informatiques des élèves. Pour autant la question de la caractérisation de ces compétences est peu abordée.

Comment les enseignants de mathématiques s'approprient-ils l'informatique comme discipline scolaire ? De quels outils disposent-ils pour analyser les connaissances en informatique des élèves ?

Dans leur article Piolti-Lamorthe et al. (2023)mettent en parallèle plusieurs types de présentations des *patterns* pour illustrer leurs potentialités communes pour le développement de la créativité, de la pensée algorithmique et de la pensée algébrique. En effet, les activités de généralisation de pattern font l'objet de nombreuses recherches, et ce aussi bien du côté de l'algèbre que de l'algorithmique. Le pattern pourrait donc jouer un rôle de facilitateur dans la rencontre entre deux contenus disciplinaires, et ainsi leur permettre de s'apporter mutuellement des ressources pour l'enseignement.

Dans ce mémoire, nous commencerons par définir la notion de *pattern* puis nous présenterons le rôle de la généralisation de pattern pour le développement de la pensée algébrique et algorithmique. La didactique de l'informatique manque à l'heure actuel de cadre théorique unificateur, nous proposerons de prendre appui sur un cadre issu de la didactique des mathématiques et présenterons la Théorie Anthropologique du Didactique de Chevallard (1992, 1999) comme cadre théorique.

Nous prendrons appui sur les apports théoriques de la revue de littérature pour élaborer une situation expérimentale de généralisation de motif en contexte de programmation, pour laquelle nous proposerons un modèle de connaissances algorithmiques à priori. Enfin, une analyse des productions et traces d'activités des élèves nous permettra d'enrichir le modèle de connaissances algorithmiques.
Nous espérons ainsi apporter de premiers éléments de réponses aux questions posées précédemment.

2. REVUE DE LITTERATURE

2.1. PATTERN

Dans leur article, Piolti-Lamorthe et ses collègues (2023) interrogent les potentialités mathématiques des patterns aux travers d'activités dont la mise en œuvre illustre la façon dont ils développent les pensées algorithmique et algébrique.

Les auteurs donnent comme définition d'un **pattern** : « un ensemble de nombres ou d'objets dont tous les éléments sont reliés les uns aux autres par **une règle spécifique** », règle qui est le plus souvent non explicitée.

Un pattern peut être constitué d'éléments *alphanumériques* ou être *figuratif* (formes géométriques, couleurs…).

On retrouve dans la littérature deux types de patterns (Liljedahl, 2004) :

- les patterns **répétitifs** : ils sont de structure cycliques et générés par répétition d'une plus petite partie, d'une unité discernable, **un motif** (Figure 1). Et on appellera **motif élémentaire** la plus petite unité de répétition qui permet de générer le pattern.

A B A B A B A B

1234 1234 1234

Figure 1 : Exemples de pattern répétitifs

- les pattern **numériques** (Liljedahl, 2004), appelés aussi **évolutifs** (Léonard et al., 2022): ils sont construits de sorte que la valeur de chaque élément dépend de la **valeur du ou des élément(s) précédent(s)** ou de celle de la **position** (Figure 2). Et ce sont les valeurs de chaque élément, ou celles qui leur sont associées, qui créent le pattern.

1 4 9 25 36

Figure 2 : Exemples de patterns évolutifs

En quoi cet objet-frontière, c'est-à-dire qui facilite la communication entre deux mondes et le rapprochement de contenus divers (Star & Griesemer, 1989) peut-il créer une opportunité de mieux caractériser le raisonnement des élèves en informatique ?

2.2. PATTERN ET PENSEE ALGEBRIQUE

2.2.1. PENSEE ALGEBRIQUE

Depuis plusieurs décennies, les recherches dans le domaine de l'enseignement de l'algèbre portent sur le développement d'une pensée algébrique dès le primaire (Kieran, 1992, 2007; Wagner & Kieran, 2018; Cai & Knuth, 2011; Radford, 2006, 2008, 2014). L'idée n'est pas de mettre en place « *un enseignement précoce de l'algèbre du secondaire, ni d'une « preéalgèbre » [...]. Il s'agit plutôt d'amener les élèves à développer la pensée algébrique sans nécessairement utiliser le langage littéral de l'algèbre » (Squalli, 2002)*

La pensée algébrique présente les **trois caractéristiques** suivantes (Radford, 2014) :

- L'indétermination : capacité à exploiter un problème qui implique des valeurs non connus (inconnues, variables, etc.).

- La dénotation : capacité à nommer ou symboliser cette indéterminée (à l'aide du code alphanumérique, mais aussi à l'aide du langage naturel, des gestes ou de signes non conventionnels)

- Le raisonnement analytique : capacité à traiter les quantités indéterminées comme si elles étaient connues et à parvenir à réaliser des opérations sur ces nombres inconnus.

La pensée algébrique est donc une manière particulière de raisonner face à des problèmes dans lesquels une ou des valeurs ne sont pas connues, et pour lesquels on utilise une certaine manière, qui n'est pas forcement la lettre, de symboliser ces inconnues. Parmi eux, les situations de généralisations de pattern qui demandent d'identifier une répétition ou de formuler une règle spécifique pour pouvoir donner l'expression d'un cas général du pattern.

2.2.2. GENERALISATION D'UN PATTERN

La pensée arithmétique correspond à un raisonnement qui ne s'appuie que sur des données numériques connues ainsi que sur les opérateurs usuels (Larguier, 2015). Le passage d'un mode de pensée arithmétique à un mode de pensée algébrique est source de difficultés et peut être entravé par d'importants obstacles (Vergnaud, 1988; Kieran, 1992)

La généralisation est reconnue par plusieurs auteurs comme étant une composante très importante de la pensée algébrique (Grugeon, 1995; Kaput, 2008; Lee, 1996; Radford, 2008, 2014; Squalli, 2000).

Les problèmes de généralisation de pattern permettent l'entrée dans l'« algèbre avant la lettre » (Bronner, 2015) et de produire des généralités algébriques sans recourir à un substitut symbolique (Radford, 2014 ; Squalli, 2015 ; (Vlassis et al., 2017). Ils nécessitent d'identifier des régularités, des relations entre les motifs, de décrire cette relation sous forme d'une règle en langage naturel ou mathématique. Les situations de dénombrement dans le contexte de pattern *figuratif* et *évolutif* constitue un support intéressant pour amener une réflexion sur les caractéristiques de la pensée algébrique (Demonty et al., 2015). Il s'agit d'**activités fondamentales** pour développer cette pensée

algébrique (Mason, 1996) dans la mesure où l'expression de la généralité requiert l'utilisation d'une indéterminée (Vlassis et al., 2017).

Généraliser un pattern correspond à la capacité à **identifier un point commun** parmi certains éléments d'une séquence S et avoir conscience que ce **point commun s'applique à tous les termes** de S et à pouvoir l'utiliser pour fournir **une expression directe** de n'importe quel terme de S.(Radford, 2006).

Squalli (2015) propose un modèle simplifié du processus de généralisation de Dörfler (1991) qui permet de décrire le « **développement idéal** » du processus grâce à une structuration des étapes d'un problème de généralisation. Dans leur article (Demonty et al., 2015) propose une activité « Post-it » (Annexe 1) qui illustre la progression mise en œuvre par ce modèle pour amener les élèves à une généralisation algébrique au sens de (Radford, 2008).

Il est possible de décrire le processus de généralisation d'une part du point de vue **phénoménologique**, c'est-à-dire en s'attachant à la démarche mise en œuvre pour généraliser, et d'autre part du point de vue **sémiotique**, c'est-à-dire en étudiant les signes permettant de l'observer. Radford (2006, 2008) identifie 3 types de raisonnement en situation de généralisation de pattern (Figure 3):

- L'**induction naïve** : elle consiste à rechercher un motif inconnu en analysant les caractéristiques d'un seul motif connu ou à généraliser de façon abusive en identifiant de manière incorrecte la structure du motif.

- La **généralisation arithmétique** : elle consiste à identifier un point commun à travers l'analyse de plusieurs termes de la suite. Elle permet d'identifier un terme à partir d'un terme proche mais ne permet pas de prédire un terme lointain.

- La **généralisation algébrique** : elle consiste à identifier une régularité à travers l'analyse de plusieurs termes de la suite et à étendre cette régularité aux autres termes. Elle permet de proposer une expression directe de n'importe quel terme.

Dans le cas de la généralisation algébrique, Radford (2006, 2008) caractérise les signes du processus généralisation observés dans la trace écrite des élèves en trois types symbolisation :

- **Factuelle** : l'inconnue est symbolisée à partir d'un nombre
- **Contextuelle** : l'inconnue est symbolisée par un substitut symbolique et garde des traces de la situation qui l'a vue naitre.
- **Symbolique** : l'inconnue est symbolisée par une lettre et permet de proposer une écriture mathématique de la régularité correcte et détachée de la situation d'origine.

Etape 1 Etape 2 Etape 3 Etape 4 ? Etape 100 ? Etape n ?

Induction naïve	Généralisation arithmétique	Généralisation algébrique		
Analyse **seul terme** Stratégie d'**essais-erreurs** **Raisonnement erroné** (proportionnalité)  Etape 3 Etape 4 3 x 3 + 2 ---> 4 x 4 + 2	Identification **point commun local** Identification d'un terme à partir d'un **terme proche**  ---> +3 ---> +3 ---> +3	Identification **régularité** – **Généralisation** pour **n'importe quel terme**		
		Symbolisation écrite		
		Factuelle	Contextuelle	Symbolique
		L'inconnue est symboliser par un nombre 100 x 3 + 2	L'inconnue est symbolisée par un substitut : ? , v, une lettre, … **mais** garde trace de la situation n colonnes de 3 + 2	Généralisation détachée de la situation et écriture correcte 3n + 2

Figure 3 : Typologie de Radford (2006, 2008) appliquée à la situation « Post-It »

Cette typologie nous permet de caractériser les raisonnements des élèves mais aussi de percevoir les enjeux du passage de la pensée arithmétique à la pensée algébrique.

Si les problèmes de généralisation de pattern sont des situations propices aux développements de la pensée algébrique, on peut se demander en quoi la généralisation de pattern est une activité tout aussi emblématique de la pensée algorithmique ?

2.3. PATTERN ET PENSEE ALGORITHMIQUE

2.3.1. LA PENSEE INFORMATIQUE

La pensée informatique[1] est introduite et définie par (Wing, 2006) comme « la capacité à résoudre des problèmes, à concevoir des systèmes et à comprendre le comportement humain en s'appuyant sur des concepts fondamentaux de la discipline et en y incluant une large collection d'outils intellectuels ».

Selon Wing (2006), la pensée informatique ne doit pas être réservée aux informaticiens mais au contraire, elle doit faire l'objet d'un enseignement à inclure aux enseignements fondamentaux. Depuis cet article fondateur (Baron & Drot-Delange, 2016) de nombreux travaux sur l'enseignement de la pensée informatique sont menés par la recherche. Les questions abordées portent sur l'étude de curricula de l'enseignement de l'informatique, et de modèles de compétences et concepts informatiques à enseigner (Brennan & Resnick, 2012; Gouws et al., 2013; Grover, 2017; Rich et al., 2018), ou encore sur l'efficacité des différentes approches pour cet enseignement (Li et al., 2022).

Brennan & Resnick (2012) articulent la pensée informatique autour de trois dimensions repérables à travers l'analyse des productions et d'entretiens avec les élèves portant sur leur activité :

- Les **concepts algorithmiques** : Séquences, boucles, évènements, parallélisme, structures conditionnelles, opérateurs et données.
- Les **pratiques informatiques** : Améliorations successives et itératives, test, débogage, …
- Les **perspectives** : expression personnelle, relation aux autres et au monde technologique

En France, l'enseignement de l'informatique, inscrit au programme des cycles 2, 3 et 4 depuis 2016, est pris en charge par les enseignants de mathématiques et de sciences et technologie. Toutefois, la pensée informatique comme mode de raisonnement n'est pas structurante des contenus des programmes (Drot-Delange & Tort, 2018), et l'enseignement de l'informatique s'est déployé à partir d'une impulsion nationale jusque dans les classes, sans intention spécifique d'apprentissage de la « pensée informatique » (Gaubert-Macon, 2022). Alors afin de mieux caractériser le raisonnement des élèves et d'évaluer les apprentissages des élèves, de quel modèle des compétences informatiques l'enseignant de mathématiques pourrait-il s'emparer ?

2.3.2. LES COMPETENCES DE LA PENSEE INFORMATIQUE

Les modèles de compétences de la pensée informatique sont multiples et ont fait l'objet de nombreuses recherches dont la revue de littérature de Gouws et al. (2013) qui dressent un ensemble de compétences en lien avce la programmation. Nous faisons le choix de nous appuyer sur les travaux de Christophe Declercq (2021).

En effet, il propose une adaptation des compétences proposées par (Selby & Woollard, 2013) et un modèle de 5 compétences mises en œuvre en contexte de l'apprentissage de la programmation :

Évaluer : Capacité à attribuer mentalement une valeur (résultat, type...) à un programme donné.

[1] Traduction de Computationnal Thinking

Anticiper : Capacité à se mettre dans la posture du programmeur qui doit décrire dans un algorithme l'enchaînement séquentiel/répétitif/conditionnel des instructions, avant le début de son exécution.

Décomposer : Capacité à transformer un problème complexe en un ensemble de problèmes plus simples équivalents au problème initial.

Généraliser : Capacité à inférer un problème général à partir d'une instance de ce problème et à repérer dans un problème particulier la répétition de traitements ou de données suivant un même schéma.

Abstraire : Capacité à « faire abstraction » des informations non pertinentes et à créer des solutions où la manière dont un problème est résolu peut être « abstraite » à l'aide d'une interface pertinente.

Au cycle 3 et 4, les élèves apprennent à utiliser des logiciels de calculs et d'initiation à la programmation, et les attendus de fin de cycle 4 sont « Écrire, mettre au point un programme simple ». Les connaissances à acquérir sont les bases de l'algorithmique et de la programmation. La modélisation proposée Declercq (2021) permet une caractérisation plus fine des compétences mises en œuvre par les élèves, et notamment en situation de généralisation de motif.

2.3.3. GENERALISATION DE MOTIF

Les **problèmes de type itératif** font l'objet de plusieurs recherches (Declercq & Tort, 2018; Declercq & Zeyringer, 2018; Drot-Delange & Tort, 2018; Peter et al., 2019; Léonard et al., 2022) et sont présentés comme des situations favorables à l'apprentissage de premiers concepts informatiques et au développement des compétences de la pensée informatique. Ce type de problèmes répondent parfaitement aux exigences du programme et prennent la forme de problèmes de programmation de déplacement (absolu ou relatif) d'un objet sur une grille.

Les activités de programmation de **dessin de patterns** sont des problèmes de type itératif. Elles nécessitent d'**identifier un motif** et de l'**exprimer** afin de le **généraliser.** La généralisation est une compétence de haut niveau (Declercq, 2021). Elle nécessite d'être capable d'identifier et de formuler un problème général à partir d'un cas particulier et ainsi de proposer une résolution systématique adaptée à tout cas particulier de ce problème.

Dans le cas d'un problème de généralisation de motif, il s'agit de repérer la répétition d'un motif, d'exprimer algorithmiquement la **séquence d'instructions** répétée afin de pouvoir la généraliser à l'aide d'une **boucle**.

2.3.4. DEUX CONCEPTS ALGORITHMIQUES FONDAMENTAUX

A. LA SEQUENCE D'INSTRUCTIONS

Une **séquence d'instructions**, est une suite finie d'instructions pour laquelle il est clairement établi (Cogis & Palaysi, 2021):

- quelles sont les instructions qui la constituent ;
- quel est l'ordre dans la suite.

Exécuter une séquence d'instructions consiste à exécuter dans l'ordre prescrit chacune des instructions de la séquence.

B. LA BOUCLE

Lorsqu'une même séquence d'instructions est exécutée plusieurs fois consécutives dans un programme, on obtient une **itération.** Pour **exprimer** une itération dans un programme, il est alors possible soit de réécrire la séquence d'instructions concernée autant de fois que nécessaire, soit d'utiliser un moyen de modifier l'exécution séquentielle (Laborne et al., 1985) comme la structure de **boucle**.

La construction d'une boucle est une **tâche cognitive** qui consiste à mener trois activités à la fois (Rogalski & Samurçay, 1986):

- **Planification répétitive** du traitement : il s'agit de l'élaboration et l'expression de l'invariant dans les actions à répéter (invariant de boucle).
- Identification du **statut fonctionnel d'arrêt** : il s'agit de préciser le moment d'arrêt : Par rapport à quelle variable ? Quelle valeur ? Par rapport à quel aspect du problème à résoudre ?
- Détermination de **l'état initial** : il s'agit de préciser dans quel état sont ou doivent se trouver les variables transformées dans l'invariant de boucle.

La mise en place de l'itération dans l'écriture d'un programme est un processus au cours duquel la signification et l'expression de l'itération évoluent (Laborne et al., 1985) et qui fait appel à l'ensemble de cinq compétences de la pensée informatique.

La généralisation de motif est un processus en plusieurs étapes au cours duquel plusieurs facteurs peuvent faciliter ou au contraire entraver son déroulement.

2.3.5. IDENTIFICATION DU MOTIF

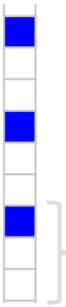
Dans leur article, Léonard et ses collègues (2022) proposent d'affiner le concept de motif avec l'idée notamment de pouvoir caractériser certaines des difficultés rencontrées en situation de résolution de problème de type itératif dans un contexte de programmation. Les auteurs font référence aux travaux de (Liljedahl, 2004) et définissent le **motif** comme « *une entité repérable au sein d'un ensemble car répétée à l'identique ou avec des variations prédictibles* ». Dans le cas de problème de programmation d'un dessin constitué d'une suite de motifs, il convient de commencer par définir ce que l'on appelle *motif visuel* (Léonard et al., 2022), c'est-à-dire cette entité repérable visuellement au sein du dessin.

A. MOTIF VISUEL

On définit le motif visuel comme ce qui est observable par l'élève, c'est-à-dire que seul l'aspect visuel est pris en considération. On distingue alors deux classes de situation selon le nombre de cases du motif : le motif est constitué d'**une seule case** ou il est constitué de **plusieurs cases**.

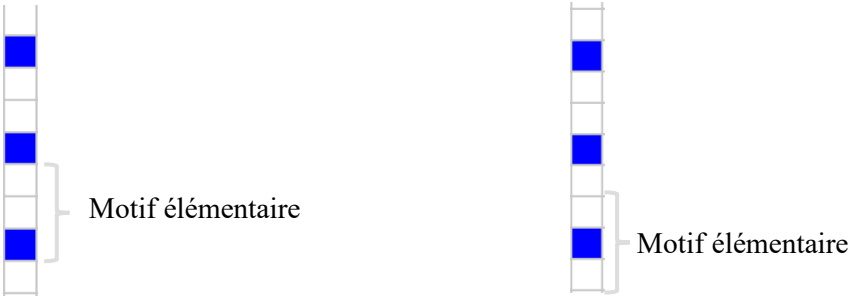
Lorsque le motif n'est constitué que d'une seule case, le taux de réussite aux problèmes de programmation de déplacement de type itératif est très élevé dès l'école élémentaire. Dans le cas où le motif s'étend sur plusieurs cases on peut distinguer plusieurs niveaux de complexité. En effet, le motif est constitué de **cases adjacentes** contenant ou non un **élément saillant**, qui selon l'environnement utilisé peut être un élément de décor ou la couleur de la case différente du fond. La présence ou non d'éléments saillants sur des cases adjacentes du motif ou n'appartenant pas au même motif permet de caractériser le niveau de complexité de l'identification du motif visuel.

Dans l'illustration de la typologie proposée, une case colorée est considérée comme un élément saillant[2].

<table>
<tr><td></td><td colspan="2">Les cases adjacentes identiques n'appartiennent pas au même motif.</td></tr>
<tr><td> 
Les cases adjacentes identiques appartiennent au même motif.
Motif</td><td> 
Les cases adjacentes de la couleur du fond appartiennent à des motifs différents.
Motif</td><td> 
Les cases adjacentes contenant le même élément saillant mais n'appartenant pas au même motif.
Motif</td></tr>
</table>

[2] Illustration correspondant à l'environnement Pixel'Art dans lequel le fond correspond à la grille, et une case est considérée comme un élément saillant lorsqu'elle est colorée.

S'il est bien entendu que le motif visuel ne concerne que ce qui est observable, il nous semble que le motif de la deuxième situation pourrait tout à fait être *vu* au moins de deux différentes façons :



Afin de rendre plus explicite cette caractérisation, j'apporte une précision par rapport à la typologie initiale, en reprenant le terme **motif élémentaire** (Liljedahl, 2004) défini comme étant *la plus petite unité de répétition qui permet de générer le pattern.* Cette précision s'avèrera utile au moment de la caractérisation des classes de situations en fonction du degré de correspondance entre le motif visuel et le motif algorithmique.

L'étude menée par Léonard et al. (2022) sur le taux de réussite des élèves aux problèmes dont le motif est sur plusieurs cases montre que la dernière catégorie est celle où l'identification visuelle du motif est la plus perturbée. On peut donc conclure que qu'un motif facilement isolé visuellement augmente les chances de réussite à la résolution de problème. Les éléments saillants sont des points de repère privilégiés lors de l'identification du *motif visuel*.

B. Classes de situations

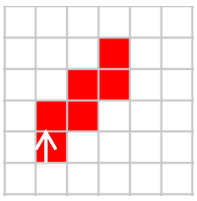
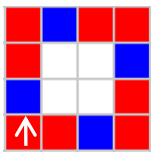
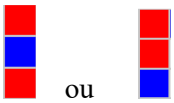
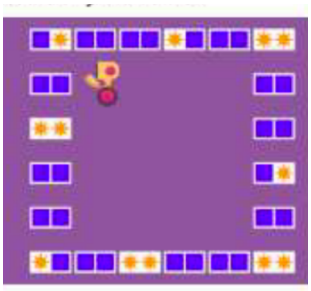
Léornard et al. (2022) proposent dans leur article une classification des situations en fonction du degré de correspondance entre le *motif visuel* et *motif algorithmique*. Le *motif algorithmique* étant définit comme le corps de la boucle. Dans leur étude, les auteurs se basent sur le taux de réussite des élèves à des problèmes de programmation de type itératifs. Les problèmes sont issus du concours Algorea et il s'agit plus particulièrement de programmer les déplacements d'un robot sur une grille afin d'atteindre un objectif (déposer une graine sur chaque tas de terre, ramasser tous les diamants, …). Les énoncés sont tous différents, et les scènes (grilles) peuvent incorporer des éléments de décors. La résolution des problèmes est plus ou moins difficile pour les élèves, notamment en raison du niveau d'abstraction nécessaire à la transposition du *motif visuel* en *motif algorithmique*.

Nous reprennons la typologie de Léornard et al. (2022) en transposant une nouvelle fois l'illustration : une case colorée est considérée comme un élément saillant, et la position et l'orientation de départ sont indiquées par une flèche.

Dans les deux premières classes, la correspondance entre les deux motifs est stricte. Chaque action de déplacement est repérable par la limite entre deux cases et les autres actions sont repérables par un élément saillant.

CLASSE 1	CLASSE 2
Déplacement selon une seule direction La case délimite le motif visuel Disposition linéaire du motif visuel élémentaire 	Déplacement selon une seule direction ou en déplacement absolu (N, S, E, O) Motif visuel linéaire de plusieurs cases. Disposition linéaire du motif visuel élémentaire 

Dans les classes suivantes, le motif visuel est constitué de plusieurs cases et des éléments saillants sur des cases adjacentes n'appartiennent au même motif. La correspondance entre le motif visuel et le motif algorithmique est partielle.

<table>
<tr><th>CLASSE 3</th><th>CLASSE 4</th><th>CLASSE 5</th></tr>
<tr><td>Déplacement relatif.
Disposition linéaire du motif visuel élémentaire.
  </td><td>Déplacement relatif.
Disposition cyclique du motif visuel élémentaire.
 
Dans ce cas, plusieurs motifs sont visibles :
 
ou</td><td>Déplacement relatif.
Perturbation visuelle du motif
 
Il y a en effet deux motifs : celui des dominos entièrement bleu (à ramasser) et celui des dominos bi-couleur</td></tr>
<tr><td>Conséquences sur le motif algorithmique :
- Plusieurs états possibles sur une même case.</td><td>Conséquences sur motif algorithmique :
- Plusieurs états possibles sur une même case.
- Plusieurs programmes possibles</td><td></td></tr>
</table>

Cette classification des situations permet d'établir **une gradation des difficultés** rencontrées par les élèves (Léonard et al., 2022) mais qu'en est-il du niveau d'acquisition des concepts informatiques mis en œuvre dans ce type d'activité ?

L'absence à l'heure actuelle de cadre théorique unificateur pour une didactique de l'informatique (Fluckiger, 2019; Declercq et al., 2022) invitent à réfléchir aux cadres théoriques mobilisables en didactique de l'informatique, et qui permettraient de modéliser les connaissances. La cadre de référence de la Théorie Anthropologique du Didactique offre des perspectives intéressantes (Jolivet et al., 2023). Il notamment est mobilisé par la recherche dans le domaine des Environnement Informatiques pour l'Apprentissage Humain (EIAH) et à l'origine du modèle didactique T4TEL (Chaachoua, 2010).

2.4. LA THEORIE ANTHROPOLOGIQUE DU DIDACTIQUE

La Théorie Anthropologique du Didactique (TAD) est un cadre de référence didactique développée par Yves Chevallard (1992, 1999).

La première notion en TAD est la notion d'**institution**, c'est-à-dire le dispositif social par rapport auquel on se situe. En effet, la TAD situe l'activité mathématiques, et donc l'activité d'étude en mathématiques donc l'ensemble des activités humaines et des institutions sociales (Chevallard, 1998).

Pour cela, On considère le **rapport au savoir** d'un **sujet**, dans la **position** qu'il occupe dans l'institution. Le **rapport personnel** de l'individu s'enrichit par les expériences, les différentes rencontres du savoir, et le **rapport institutionnel** est celui attendu par l'institution.

Tout savoir est donc le savoir d'une institution et il peut changer en passant d'une institution à une. C'est le rapport institutionnel qui dit comment doit vivre le savoir dans l'institution. Pour qu'un savoir puisse vivre dans une institution il doit donc s'adapter aux contraintes de l'institution.

Dans la TAD, il y a apprentissage lorsque le rapport personnel d'un sujet à un objet de savoir change et évolue vers le rapport institutionnel. Pour qu'un savoir puisse être appris ou enseigné, il est donc nécessaire de prendre en compte les contraintes institutionnelles, c'est ce que permet la transposition didactique.

2.4.1. TRANSPOSITION DIDACTIQUE

La transposition didactique (Figure 4) est une notion introduite en 1980 par Chevallard. Il s'agit du processus de transformation d'un **savoir savant**, *scientifique*, en un **savoir enseigné**, avec un l'écart plus ou moins grand selon l'institution dans laquelle on se place.

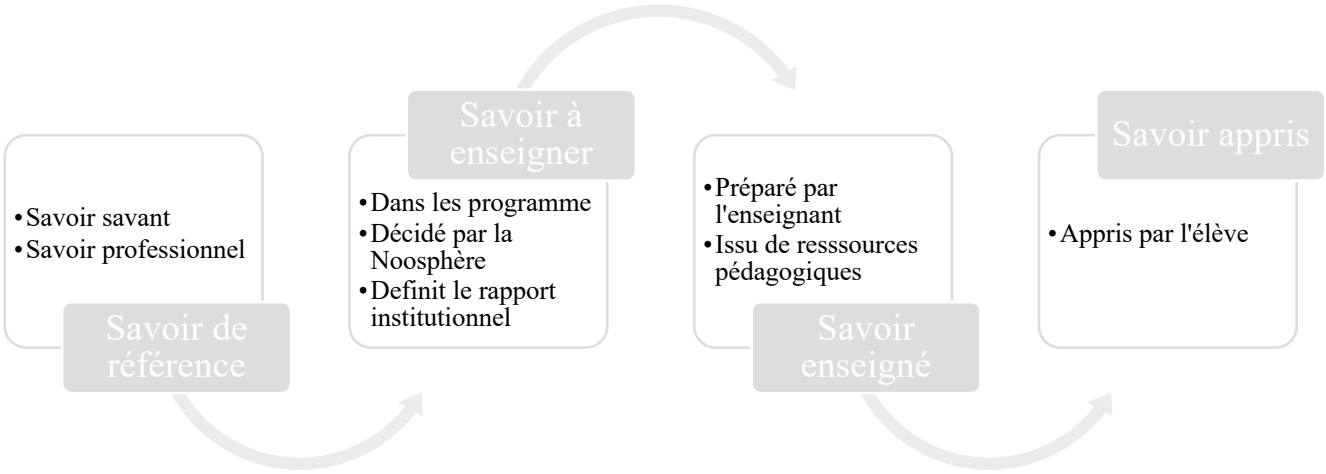


Figure 4 : Transposition didactique

Ce processus implique des choix, de la Noosphère[3], de l'enseignant, afin de rendre les savoirs accessibles aux apprenants et de permettre l'évolution de leur rapport personnel aux objets de savoirs institutionnels. Comment est-il possible caractériser cette évolution, de la préciser ?

2.4.2. PRAXEOLOGIE

La théorie anthropologique du didactique considère que toute activité humaine consiste à accomplir une **tâche** t d'un certain **type** T au moyen d'une **technique** τ, justifiée par une **technologie** θ, elle-même justifiée par une **théorie** Θ. Et donc, qu'elle peut être modélisée par un quadruplet $[T, \tau, \theta, \Theta]$ (Chevallard, 1992; Bosch & Chevallard, 1999) nommé **praxéologie ou organisation praxéologique**.

Praxis signifie « pratique » et renvoie au bloc $[T, \tau]$, le bloc du savoir-faire, *logos* qui signifie « raison » renvoie au bloc $[\theta, \Theta]$, celui du savoir.

[3] La noosphère est la sphère des gens qui pensent, qui réfléchissent sur l'enseignement, dans quelque registre qui soit.

2.4.3. PORTEE D'UNE TECHNIQUE

« **Une tâche**, et le **type de tâches** *parent*, s'exprime par un ***verbe suivi d'un complément*** : *balayer* la pièce, *développer* l'expression littérale donnée, *diviser* un entier par un autre, *saluer* un voisin, *lire* un mode d'emploi, *monter* l'escalier, *intégrer* la fonction $x \longmapsto xlnx\ entre\ x = 1\ et\ x = 2$, etc. » (Chevallard, 1998)

Pour *T* un type de tâches *donné*. Une **technique** relative à *T* précise une **manière d'accomplir**, de réaliser les tâches *t* de *T.* Il se peut qu'une technique ne réussit que sur une partie *P*(t) des tâches du type *T* auquel elle est relative : c'est ce que l'on appelle la **portée de la technique.**

« Ce qu'observe un chercheur dans une institution donnée ce sont des tâches : comment peut-il définir un type de tâches ? Ou encore, comment rattacher et organiser les tâches autour d'un même type de tâches ? » (Chaachoua, 2018).

2.4.4. GENERATEUR DE TYPES DE TACHES

Un premier niveau de modélisation du type de tâches est de regrouper les tâches autour d'un **genre de tâches** : un **verbe** comme calculer, développer... Puis on ajoute un **complément** afin de les discriminer par rapport aux objets communs sur lesquels porte l'action et par rapport aux moyens communs d'accomplir les tâches.

Le complément peut être défini selon différent **niveau de granularité**, du spécifique au générique. Ce niveau de granularité peut être spécifié grâce à la notion de **variable**.

Un générateur de type de tâches n'est pas un type de tâches. Il permet d'engendrer des types de tâches selon une **structuration hiérarchique** définie par des variables (Chaachoua, 2018).

Soit GT un générateur de type de tâches, on note :

GT = [Verbe d'action ; complément fixe ; Systèmes de variables]

Le verbe d'action permet de définir le genre de tâches, le complément défini le niveau de granularité du générateur. Le système de variables génératrices est défini par :

- une suite de variables
- les valeurs que peuvent prendre ces variables dans ce système

L'introduction des variables permet de structurer un ensemble de situations spécifiques d'une connaissance ou d'un savoir (Chaachoua & Bessot, 2019). Une variable permet donc de :

(1) Générer des sous-types de tâches
(2) Caractériser la portée des techniques

Ces deux *fonctions* des variables sont particulièrement utiles dans le cadre d'*analyse a priori* ou dans la construction de parcours d'apprentissage.

Le choix des valeurs des variables d'un générateur de type de tâches permettent de porter un triple point de vue (Chaachoua & Bessot, 2019) :

- **Épistémologique** : le découpage des valeurs est tel que le changement de valeur modifie l'éventail des techniques possibles du type de tâches.
- **Institutionnel** : les contraintes et conditions définies par l'institution dans laquelle on se place viennent restreindre les valeurs des variables d'un type de tâches relatif institutionnel.
- **Didactique** : une variable didactique est une variable au sein d'une institution, ses valeurs dépendent donc de celle-ci.

Une **variable didactique** est *potentiellement* à disposition de l'enseignant. Il s'agit d'un élément qui peuvent varier dans une situation et provoquer ou faire changer les stratégies, les procédures (le coût, la complexité, ...) des élèves (Brousseau, 1982).

Et si ses valeurs sont définies *a priori*, elles peuvent être enrichies *a posteriori* pour rendre compte des praxéologies personnelles des élèves (Croset & Chaachoua, 2016). Il s'agit de la troisième fonctions des variables d'un générateur de type de tâches :

(3) Rendre compte des praxéologies personnelles

3. PROBLEMATIQUE

Objet de recherches aussi bien du côté de l'algèbre que de l'algorithmique, notre revue de la littérature illustre le statut du pattern comme objet-frontière entre la pensée algébrique et la pensée algorithmique. La modélisation des types de raisonnement algébriques (Radford, 2006, 2008) et la structuration d'activité permettant un *développement idéal* du processus de généralisation (Squalli (2015) ; Vlassis (2017)) ouvrent ainsi des perspectives d'une possible transposition en algorithmique. La classification des situations, parce qu'elle permet d'établir une gradation des difficultés rencontres par les élèves (Léonard et al., 2022), apporte des éléments d'éclairage pour cette transposition.

Comment grâce aux apports de la recherche aussi bien du côté de l'algèbre que du côté de l'algorithmique et de la programmation pouvons-nous caractériser le raisonnement des élèves en situation de généralisation de pattern en informatique ? Et en quoi le cadre théorique de la Théorie Anthropologique du Didactique peut-il nous permettre modéliser les connaissances en jeu ?

4. METHODE

4.1. ANALYSE A PRIORI

4.1.1. GENERATEUR DE TYPE DE TACHES

Les problèmes de généralisation de motif sont des problèmes de programmation de type itératif. Afin de modéliser les connaissances en jeu dans la résolution de ces problèmes, nous définissons le générateur de types de tâches suivant :

GT : [Écrire un programme de type itératif, $V_{Flèche}$, V_{Souris} , $V_{Classe\ de\ situation}$, $V_{Nombre\ de\ répétitions}$, $V_{Imbrication}$]

Les **deux premières variables** : $V_{Flèche}$, V_{Souris} sont des variables descriptives de l'environnement informatique définies en prenant appui sur les travaux de Declercq & Tort (2018).

En effet, Declercq & Tort (2018) s'intéressent à l'impact de l'environnement d'apprentissage de la programmation sur l'activité des élèves et plus particulièrement à la **posture** qu'ils adoptent dans la résolution de problème de déplacement. Pour cela, ils mènent une analyse des postures observées lors de mise en activité des élèves à partir de situations de programmation avec Scratch (*Scratch - Imagine, Program, Share*, s. d.) issues des documents d'accompagnement des programmes. Ainsi ils identifient **quatre postures possibles chez l'élève** :

- **Utilisateur de télécommande** : L'élève effectue des manipulations directes mais ne programme pas. Par exemple, l'affectation d'évènements aux flèches du clavier (actions de déplacement) permet de guider un personnage, ou curseur, sans avoir à programmer.
- **Programmeur pas à pas** : L'élève réussit à écrire un programme simple mais ne développe pas de compétence d'anticipation.
- **Programmeur de télécommande** : L'élève est dans une posture de programmeur mais reporte la définition de la stratégie de résolution du problème, sur le futur utilisateur de la télécommande. Par exemple, la répétition d'un motif par déplacement de la position initiale du curseur à l'aide de la souris et non par répétitions des déplacements programmés.

- **Programmeur** : L'élève est dans le rôle du programmeur voulu par l'enseignant.

Les postures de télécommande sont étroitement liées aux fonctionnalités de Scratch (programmation d'évènement déclenché par le clavier, repositionnement du lutin, …) et permettent aux élèves de se détourner de l'activité de programmation.

Pixel'Art est conçu pour empêcher ces postures de *télécommande* et favoriser l'émergence de la posture de programmeur. Il offre la possibilité d'expérimenter la programmation d'un dessin de façon autonome grâce à un parcours d'activités proposé.

Ainsi, il est donc possible de caractériser deux **variables liées à l'environnement informatique** de la situation :

- $\mathbf{V_{Flèche}}$: L'environnement permet ou pas l'affectation d'évènements aux flèches du clavier. Les valeurs possibles sont : Oui et Non
- $\mathbf{V_{Souris}}$: La position initiale du curseur est ou pas déplaçable à l'aide de la souris. Les valeurs possibles sont : Oui et Non

Les **trois variables** suivantes : $\mathbf{V_{Classe\ de\ situation}}$, $\mathbf{V_{Nombre\ de\ répétitions}}$, $\mathbf{V_{Imbrication}}$ sont des variables didactiques de la situation. Des exemples d'instanciations seront données dans la section suivante.

- $\mathbf{V_{Classe\ de\ situation}}$ correspond à la classe de la situation comme définie par Léornard et al. (2022) en fonction du degré de correspondance entre le *motif visuel* et *motif algorithmique*. Les valeurs possibles sont 1 ; 2 ; 3 ; 4 ou 5 (Voir Classes de situations p.8)

- $\mathbf{V_{Nombre\ de\ répétitions}}$ correspond au nombre de répétition du motif. Les valeurs possibles sont des nombres entiers.

- $\mathbf{V_{Imbrication}}$ correspond à la présence d'un motif dans le motif et qui peut se traduire dans l'algorithme par l'imbrication de deux boucles. Les valeurs possibles sont : Oui et Non

En jouant sur les valeurs de variables, le générateur de types de tâches nous permet de générer cinq types de tâches et ainsi de produire une situation expérimentale.

4.1.2. SITUATION EXPERIMENTALE

Nous avons fait le choix de mener l'expérimentation sous l'environnement Pixel'Art afin de favoriser la posture de *programmeur* chez les élèves. La valeur des variables $\mathbf{V_{Flèche}}$, $\mathbf{V_{Souris}}$ est donc *Non* pour chacune.

A. VARIABLES DIDACTIQUES DE LA SITUATION

o LA CLASSE DE SITUATION

Les classes de situation 1 et 2 sont globalement bien réussies dès le CM1 (Léonard et al., 2022), les déplacements sur la grille sont selon une seule direction, ou en déplacement absolu ce qui limite les difficultés de programmation. En effet, le *motif visuel* correspond au *motif algorithmique*, tous les déplacements sont visibles.

L'identification du motif est la première étape du processus de généralisation. La 5ème classe de situations ne semble pas adaptée puisque l'identification du motif visuel est entravée. De même, la disposition cyclique du motif visuel élémentaire dans la 4ème classe de situation peut être une source de difficultés dans l'identification et l'expression du motif (Voir tableau des Classes de situations).

Nous choisissons donc de proposer un parcours d'activités de **classe 3**. Les erreurs possibles dans cette classe de situation sont liées à l'existence de plusieurs états possibles sur une même case.

$\mathbf{V_{Classe\ de\ situation}}$ est donc égale à 3 pour chaque sous type de tâches de la situation expérimentale.

o PROGRESSION DE LA SITUATION EXPERIMENTALE

Il ne suffit pas de placer les élèves dans une situation problème pour laquelle il ne possède pas encore d'instruments de solution pour que leur démarche de résolution débouche nécessairement sur la construction attendue (Laborde et al., 1985). En effet, lorsque la séquence d'instructions est identifiée et le nombre de répétition connu, il suffit alors d'écrire la séquence d'instructions le nombre de fois nécessaire. Toutefois, ce mode de planification a ses limites (Rogalski & Samurçay, 1986) :

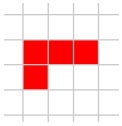
- lorsque le nombre de répétitions est assez grand, la succession d'action devient alors difficilement contrôlable ;

- lorsque le nombre de répétitions n'est pas connu à l'avance et dépend d'une condition réalisée par l'exécution des actions elles-mêmes

Pour amener à l'utilisation de la boucle, nous proposons de transposer la structuration des étapes de la généralisation algébrique (Squalli, 2015) pour construire une progression de la situation expérimentale :

TACHE 1

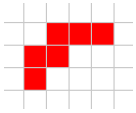
Écrire un programme de type itératif, $V_{Flèche = Non}$, $V_{Souris = Non}$, $V_{Classe de situation=3}$, $V_{Nombre de répétitions = 1}$, $V_{Imbrication=Non}$



Cette tâche peut sembler pauvre d'un point de vue algorithmique. Ceci dit, elle favorise la dévolution (Brousseau, 1982) de l'activité à l'élève. Elle permet aussi à l'élève de s'approprier le langage Pixel Art et ainsi de limiter l'apparition des erreurs de langage par la suite.

TACHE 2

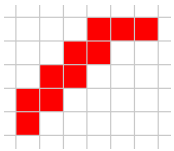
Écrire un programme de type itératif, $V_{Flèche = Non}$, $V_{Souris = Non}$, $V_{Classe de situation=3}$, $V_{Nombre de répétitions = 2}$, $V_{Imbrication=Non}$



Dans cette tâche, l'apparition de la boucle est possible mais n'est pas forcément un attendu. Il s'agit plutôt d'une étape favorisant l'émergence de la boucle aux étapes suivante avec une première exposition à la répétition du motif.

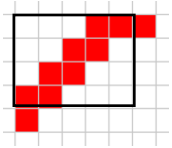
TACHE 3

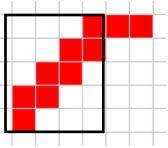
Écrire un programme de type itératif, $V_{Flèche = Non}$, $V_{Souris = Non}$, $V_{Classe de situation=3}$, $V_{Nombre de répétitions = 4}$, $V_{Imbrication=Non}$



Dans cette tâche, le dessin comporte un pattern visible avec 4 répétitions du motif. Une généralisation du motif algorithmique est attendue.

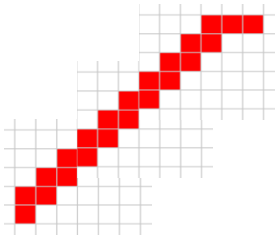
Si le motif visuel pensé dans l'élaboration de la tâche est la colonne de deux carrés rouges  , il est possible que les élèves identifient comme motif la ligne de carrés rouges :  . Il y a donc deux « visualisations » possibles du pattern :





TACHE 4

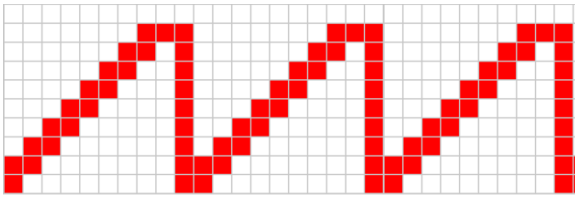
Écrire un programme de type itératif, $V_{Flèche = Non}$, $V_{Souris = Non}$, $V_{Classe de situation=3}$, $V_{Nombre de répétitions = 10}$, $V_{Imbrication=Non}$



Dans cette tâche, le dessin comporte un pattern visible avec 10 répétitions du motif. Une généralisation du motif algorithmique est attendue.

TACHE 5

Écrire un programme de type itératif, $V_{Flèche = Non}$, $V_{Souris = Non}$, $V_{Classe de situation=3}$, $V_{Nombre de répétitions = 10}$, $V_{Imbrication=Oui}$



Les problèmes impliquant des boucles imbriquées sont plus difficiles (Declercq & Tort, 2018). Grâce à cette dernière tâche, nous souhaitons observer les stratégies mise en œuvre par les élèves et caractériser leur niveau d'acquisition du schéma itératif, et par là leur capacité à généraliser l'utilisation de la boucle.

La tâche 5 offre donc la possibilité de caractériser la généralisation algorithmique à deux niveaux :

(1) Généralisation du motif algorithmique

(2) Généralisation de l'utilisation de la boucle pour la résolution de problème itératif

Les tâches 1 à 4 appartiennent aux sous types de tâches pour lesquels la valeur $V_{Nombre de répétitions}$ varie et prend les valeurs *1 ;2 ;4 ; 10* et celle de $V_{Imbrication}$ est *Non*. La tâche 5, elle, permet notamment de jouer sur la variable $V_{Imbrication}$ qui prend, pour ce sous type de tâches, la valeur *Oui*.

4.1.3. MODELE PRAXEOLOGIQUE DE REFERENCE

Afin de caractériser le raisonnement des élèves en situation de généralisation algorithmique, nous nous appuyons sur le modèle de Radford (2006, 2008) pour la généralisation algébrique, et proposons une typologie des raisonnements d'élèves dans la résolution d'un problème de généralisation.

Nous nous plaçons à un niveau « méta » (Margolinas, 1995) en caractérisant le niveau d'acquisition du concept de boucle et propose trois niveaux de généralisation en fonction du moment d'apparition de la boucle dans la production des élèves :

- **Programmation naïve :** L'élève met en œuvre une programmation naïve par une succession d'essais/erreurs
- **Programmation pas à pas :** L'élève met en œuvre une programmation pas à pas. Il programme une suite d'instructions et répétant l'écriture de la même séquence autant de fois que nécessaire.
- **Généralisation algorithmique** : L'élève a identifié le motif visuel élémentaire, l'a transposé algorithmiquement et il généralise le motif algorithmique à l'aide d'une boucle.

Les activités en algorithmique ou en programmation nécessite de penser le plan général et les schémas qui l'organise. (Rogalski & Samurçay, 1990) définissent la notion de *schéma* comme une structure utilisée dans le traitement de l'information pour atteindre des petits objectifs et celle de *plan* comme un ensemble organisés de schémas s'adaptant à l'organisation. Ces deux notions vont nous permettre d'analyser les activités algorithmiques des élèves.

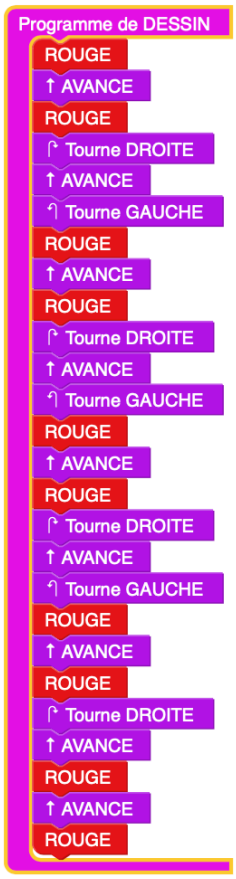
Les élèves peuvent faire le choix d'une démarche « ascendante », c'est-à-dire partir du schéma connu puis construire un plan, ou mettre en œuvre une démarche « descendante », partir d'un plan et spécifier les schémas. Lorsqu'un élève débute il ne dispose pas d'un répertoire de schémas suffisamment complet et adaptable pour planifier de façon adéquate (Lagrange & Guy, 2015).

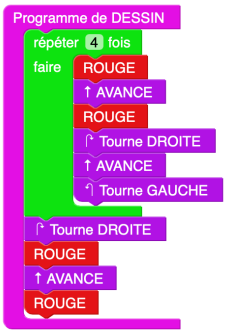
Dans le cas de la généralisation d'un motif en contexte de programmation, nous pouvons la caractériser de la façon suivante :

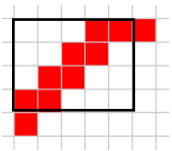
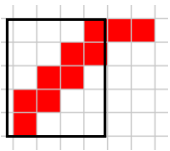
- **Schéma ascendant :** L'élève a identifié une régularité, construit le motif algorithmique puis l'intègre dans une boucle.
- **Schéma descendant :** L'élève a identifié une régularité, il crée une boucle puis construit le motif algorithme à intégrer dans cette boucle.

Dans le cas de la programmation d'un dessin sous Pixel'Art, le processus de généralisation du motif algorithme correspond à identifier le motif visuel, mettre en correspondance le motif et les actions, exprimer ce motif algorithmique avec le langage de Pixel'Art puis le généraliser à l'aide d'une boucle. Nous supposons que selon le niveau d'acquisition de la boucle, les élèves vont mettre en œuvre différentes techniques pour accomplir les cinq sous type de tâches de la situation expérimentale.

Nous proposons le modèle praxéologique théorique suivant pour le type de tâches générique : T « Ecrire un programme de type itératif » :

<table>
<tr><th>Programmation naïve</th><th>Programmation « pas à pas »</th><th colspan="2">Généralisation algorithmique</th></tr>
<tr><td>τ_1 :
- ajouter/supprimer des instructions
- exécuter le programme</td><td>τ_2 :
- identifier le motif visuel
- effectuer une correspondance « terme à terme » en répétant la programmation de la séquence d'instructions correspondant au motif algorithmique autant de fois que nécessaire
- exécuter le programme</td><td>τ_3 : Ascendante
- identifier le motif visuel
- programmer la séquence d'instructions correspondant au motif algorithmique
- intégrer le motif algorithmique dans une boucle
- modifier le nombre de répétition pour correspondre au pattern.
- exécuter le programme</td><td>$\tau_{3'}$: Descendante
- identifier le motif visuel
- créer une boucle
- construire la séquence d'instructions correspondant au motif algorithmique comme corps de l'itération
- modifier le nombre de répétition pour correspondre au pattern.
- exécuter le programme</td></tr>
<tr><td>θ_1 :
- Notion de séquence
- Déplacement relatif</td><td>θ_2:
- Notion de séquence
- Notion de motif
- Déplacement relatif</td><td colspan="2">θ_3:
- Notion de séquence
- Notion de motif
- Notion de boucle
- Déplacement relatif</td></tr>
<tr><td colspan="4">Illustration par des productions possibles pour la tâche 3</td></tr>
<tr><td>La programmation naïve se caractérise par une succession d'essais/erreurs qu'il est difficile de représenter de façon exhaustive.</td><td> 
</td><td colspan="2"> 





Le pattern est vu comme 4 colonnes ou comme 4 lignes</td></tr>
</table>

Les traces d'activité des élèves pour la résolution des tâches de la situation expérimentale vont nous permettre d'observer les techniques mises en œuvre par les élèves pour généraliser **le motif algorithmique** dans le cas d'un pattern de classe de situation 3 à l'aide d'une boucle.

4.1.4. PORTEE PRAGMATIQUE

La **portée théorique** d'une technique est l'ensemble des tâches où la technique permet d'accomplir une tâche quelconque de cet ensemble en dehors de considération des conditions de son exécution. On la note $P_{Th}(\tau)$. La **portée pragmatique** d'une technique est l'ensemble des tâches où la technique est **fiable** dans le sens où elle permet d'accomplir ces tâches avec peu de risque d'échec et à un coût raisonnable. On la note $P(\tau)$.

La progression de la situation expérimentale est pensée pour provoquer le recours à la boucle et donc de se confronter à la limite de la portée pragmatique de la technique τ_2 de la programmation pas à pas (Figure 5). Les cinq tâches amènent ainsi, a priori, les élèves à passer de l'utilisation de $\tau_1$1 à τ_2 puis τ_3.

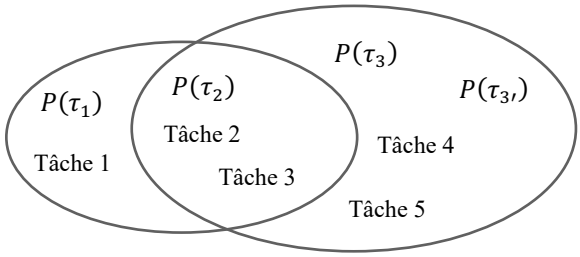


Figure 5 : Portées pragmatiques des techniques

4.2. PARTICIPANTS

Les deux expérimentations ont été menées avec une classe de 6ème de 26 élèves. Les élèves ont déjà mené deux projets Scratch dont les objectifs étaient la découverte de l'environnement, la programmation des déplacements, et d'évènement (envoi de messages entre deux lutins). Aucune notion algorithmique n'a été institutionnalisée et ils n'ont jamais travaillé sur l'environnement Pixel'Art.

4.3. MATERIEL

4.3.1. PIXEL'ART

Pixel'Art (*Pixel'Art*, s. d.) est un environnement informatique dédié à l'apprentissage de la programmation au cycle 3 et 4. Plus précisément, il s'agit d'un micro-monde qui propose aux élèves des **problèmes de programmation d'un dessin** par déplacement d'un curseur (symbolisé par une flèche) sur une grille et la colorisation de cases. Il a été conçu par Declercq, et programmé par Mauras, pour amener l'élève à être dans la posture du *programmeur* (Declercq & Tort, 2018).

Le **jeu d'instructions** mis à disposition des élèves (**Figure** 7) est réduit aux **déplacements relatifs** du curseur et à la **colorisation** de la case courante, à cela s'ajoutent **deux structures** : la séquence et la répétition bornée. Sa manipulation est assez simple : un bloc peut être créé par « cliquer-glisser » dans la **zone de programmation** et supprimé par « cliquer-glisser » dans la poubelle en bas à droite.

L'élève peut construire progressivement le programme. A chaque action « Dessiner », il exécute le programme et peut comparer le dessin obtenu à celui attendu. Lorsqu'il « efface » le dessin, le curseur est repositionné à sa position de départ. Les activités permettent de découvrir les instructions de déplacements, les notions de séquence et de répétitions bornées. Les modes « Voire la trace » et « Pas à pas » sont des retours instrumentaux et facilitent la correction d'erreurs de programmation. En effet, le mode « Voir la trace » permet d'observer la trace du curseur sur la grille et de suivre les positions et orientations successives. Le mode « Pas à pas » permet de suivre l'exécution de la séquence, instruction par instruction.

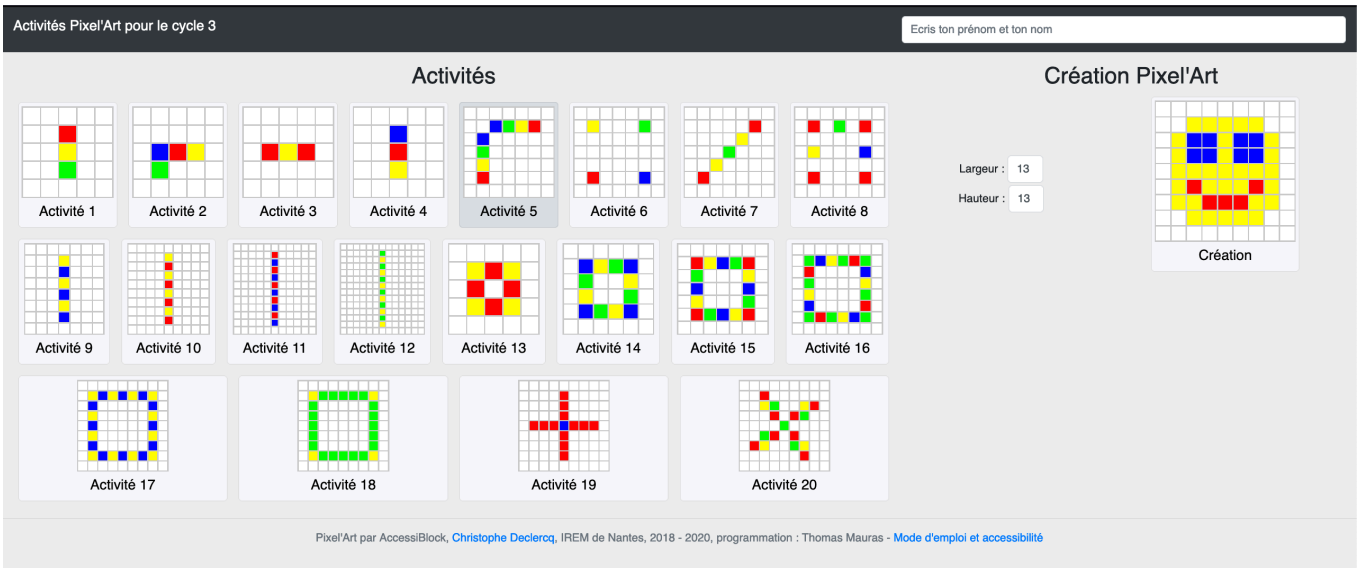


Figure 6 : Pixel'Art – Parcours d'activités

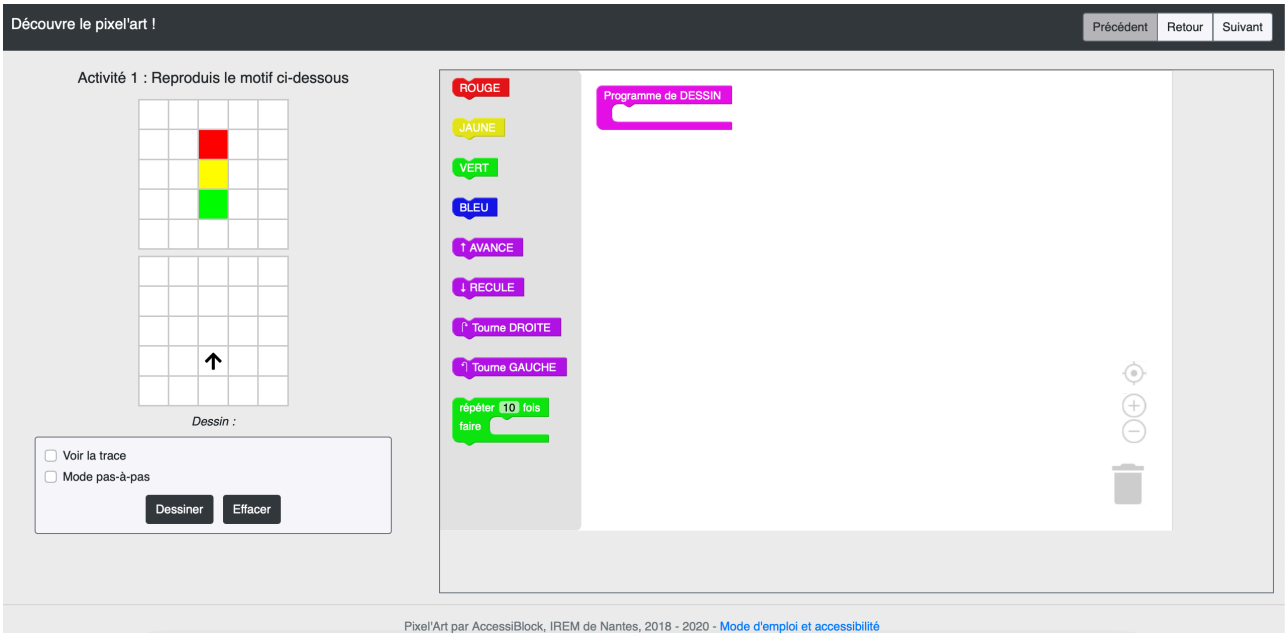


Figure 7 : Pixel'Art (Capture d'écran)

4.3.2. PARCOURS D'ACTIVITES PIXEL'ART

Dans le cadre de l'expérimentation, deux parcours d'activités sont proposés aux élèves :

- un parcours de **découverte** : https://beffarae.gricad-pages.univ-grenoble-alpes.fr/PixelArt3/
- un parcours **expérimental** : https://beffarae.gricad-pages.univ-grenoble-alpes.fr/PixelArt3/parcours1.html

A. DECOUVERTE DE L'ENVIRONNEMENT PIXEL'ART - PARCOURS 0

Afin que la prise en main de l'environnement Pixel Art n'interfère dans l'analyse du raisonnement des élèves, nous proposons aux élèves une séance de découverte de l'environnement. Pour cela, les élèves n'auront accès qu'aux quatre premières activités de programmation de dessin du parcours Pixel'Art.

Pour chacune d'elle, la tâche est de « Reproduire le motif » dans une grille vierge sur laquelle la position de départ du curseur est indiquée. Ces premières activités permettent aux élèves d'effectuer des premières manipulations de l'artefact et de s'approprier le langage de Pixel'Art.

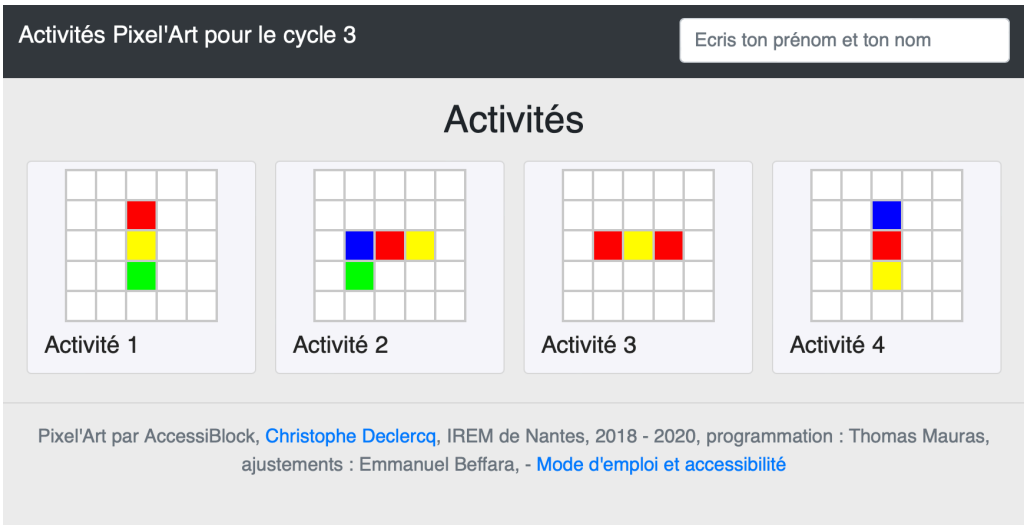


Figure 8 : Pixel'Art – Parcours 0

Ce « parcours 0 » permet aux élèves de découvrir la notion de séquence et les instructions élémentaires de Pixel Art et sans avoir besoin de recourir à structure de la boucle. Il n'y a pas de cases adjacentes ayant le même élément saillant, la même couleur, autrement dit, il n'y a pas de répétition et le dessin constitue le motif.

Les principales sources d'erreurs de programmation des dessins sont la découverte du langage et la position de départ du curseur qui change selon le problème.

B. PARCOURS EXPERIMENTAL

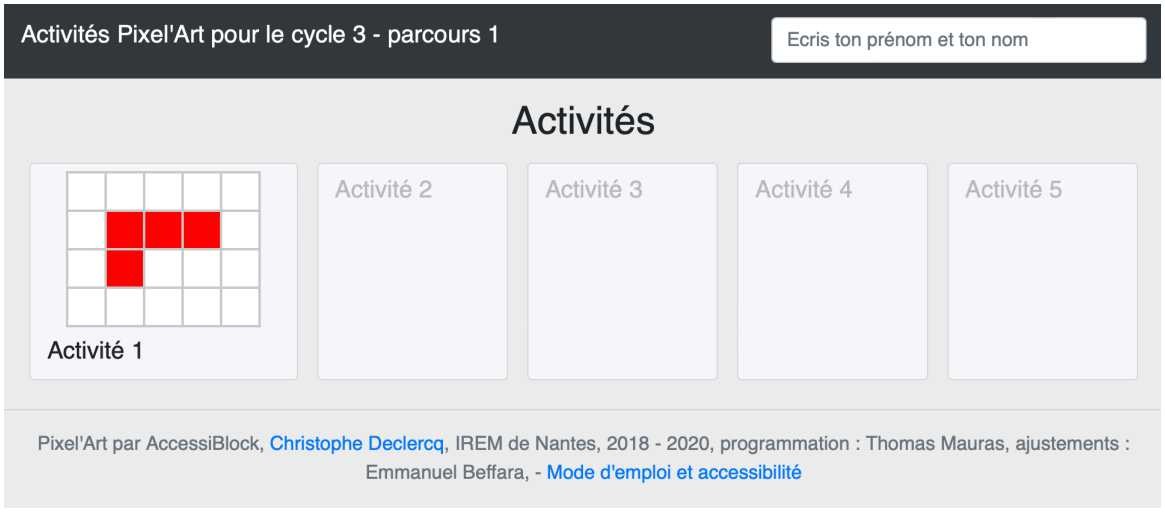


Figure 9 : Pixel'Art – Parcours expérimental

L'*interface* du parcours expérimental a été conçue de façon à ce que chaque tâche (« activité » pour les élèves), ne soit dévoilée que lorsque la précédente est réussie. L'idée étant de pas dévoiler toute de suite le pattern constitué par le parcours afin de ne pas influer sur le recours à la boucle.

4.3.3. INSTRUMENTATION DE PIXEL'ART

Une stratégie d'essai/erreur peut permettre à tous les élèves d'élaborer un programme, plus ou moins efficace, et de résoudre le problème initial. Il est possible d'évaluer le niveau l'acquisition de la notion de boucle avec la version

finale du programme et l'apparition ou non de la boucle dans celle-ci. Ceci-dit cette dernière version ne témoigne pas de la démarche mise en œuvre par l'élève pour y parvenir. Ce sont les manipulations effectuées pendant l'activité de programmation qui illustrent et permettent de caractériser le raisonnement mis en œuvre par l'élève pendant la résolution du problème.

L'instrumentation de Pixel'Art (*Pixel'Art*, s. d.) permet de mener une analyse descriptive de l'activité d'un élève à partir de deux sources :

- la **production** : le programme qui permet de résoudre le problème (reproduire un dessin ou une frise)
- la **trace d'activité** : les actions qui ont permis d'aboutir à la production.

A. HORODATAGE

L'enregistrement de l'activité est horodaté, il est ainsi possible de connaitre le temps que l'élève a passé sur chaque tâche et de suivre le passage d'une activité à une autre. Il permet aussi d'observer la rapidité de programmation ou au contraire le temps accordé au choix d'une instruction.

Enregistrement de l'activité

16:24:04 : Début activité 1
16:24:05 : Création d'un bloc dessin_debut
16:24:09 : Création d'un bloc dessin_vert
16:24:11 : Création d'un bloc dessin_avance
16:24:15 : Création d'un bloc dessin_jaune
16:24:18 : Création d'un bloc dessin_avance
16:24:21 : Création d'un bloc dessin_rouge
16:24:30 : Dessine OK Activité 1
16:24:35 : Début activité 2
16:24:36 : Création d'un bloc dessin_debut

Figure 10 : Trace d'activité Pixel'Art Initiale - Horodatage

B. ACTIONS DE PROGRAMMATION

Les actions de programmation visibles dans la trace d'activité sont :

- **Création d'un bloc dessin_debut** : correspond à la mise en place automatique de la structure de la séquence du dessin dans la zone de programmation

- **Création d'un bloc dessin_ *instruction*** : correspond à prendre une *instruction* parmi celles du jeu d'instructions mis à disposition et la faire glisser vers la zone de programmation.

- **Création d'un bloc controls_repeat** : correspond à prendre une structure de boucle et la faire glisser vers la zone de programmation.

- **Destruction d'un bloc** : correspond à suppression d'un bloc de la zone de programmation en le faisant glisser dans la poubelle en bas à droite de l'écran

- **Dessine** : correspond à l'exécution du programme.

- **Dessine OK Activité *N*** : correspond à l'exécution du programme et la réussite de l'activité *N*.

C. SUIVI DU PROGRAMME

Au cours de l'activité de programmation l'élève est amené à ajouter et/ou supprimer des instructions et dans le cas d'utilisation de la structure de boucle, à modifier le nombre de répétitions.

o AJOUT D'INSTRUCTION

Les actions **Création d'un bloc** permettent de suivre la chronologie des ajouts de blocs. Toutefois comme le montre l'exemple ci-dessous (figure 4), aucune précision n'est apportée sur **l'emplacement de l'instruction**.

Un bloc créé peut être placé dans ou hors de la structure de séquence. La trace ne précise ni le « lieu de dépôt » du bloc, ni à quel endroit est placé le bloc dans la séquence le cas échéant.

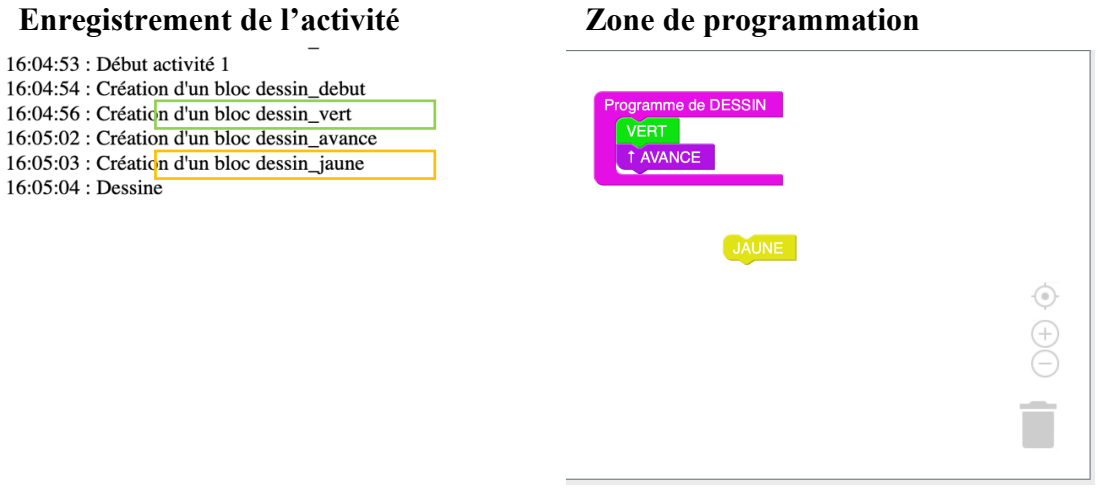


Figure 11 : Trace d'activité Pixel'Art Initiale – Ajout d'instruction

- **SUPPRESSION D'INSTRUCTION**

Pour supprimer un bloc de la séquence, il est possible soit de le faire « glisser » dans la poubelle située en bas à droite, soit de le « mettre de côté » dans la zone de programmation.

L'action **Destruction d'un bloc** ne permet pas de savoir quelle instruction a été supprimée (Figure 7).

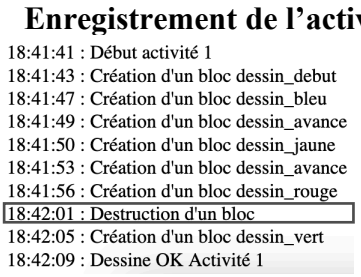


Figure 12 : Trace d'activité Pixel'Art Initiale – Suppression d'instruction (1)

Il est possible de supprimer une instruction de la séquence en la plaçant simplement dans la zone de programmation hors de la séquence. La trace ne permet pas d'observer cette action. C'est ce que l'exemple ci-dessous illustre : la suppression de la séquence des instructions « bleu » et « avance » n'apparait pas dans la trace d'activité.

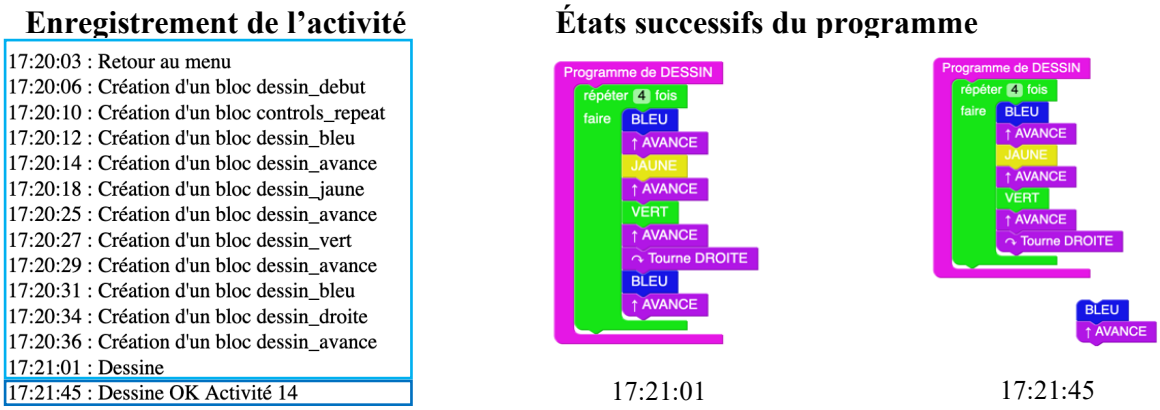


Figure 13 : Trace d'activité Pixel'Art Initiale – Suppression d'instruction (2)

Le programme exécuté à 17:21:01 ne permet pas d'obtenir le dessin attendu. Les instructions « bleu » et « avance » ont été supprimées de la séquence mais pas de la zone de programmation. Le programme permet d'obtenir le dessin attendu mais la trace ne nous donne pas accès à la correction du programme.

○ MODIFICATION DU NOMBRE DE REPETITION

Lors de la **Création d'un bloc controls_repeat**, le nombre de répétitions par défaut est 10. L'enregistrement de d'activité ne comporte pas de trace de la **modification de ce nombre** (figure 14).

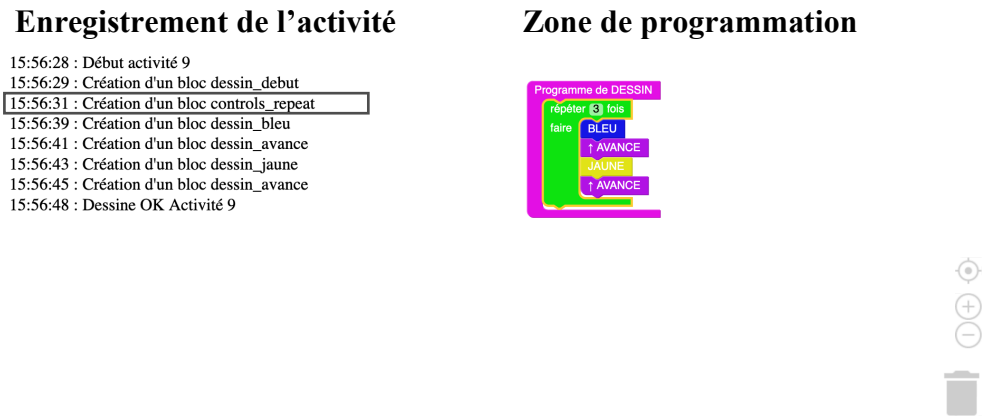


Figure 14 : Trace d'activité Pixel'Art Initiale – Création Boucle

Ces trois exemples illustrent en quoi la version initiale de l'instrumentation de Pixel'Art, si elle permet de suivre **les actions de programmation**, ne permet de connaitre la **version du programme exécutée** à partir uniquement de l'enregistrement de l'activité.

Enfin, l'utilisation des fonctions « Voir la trace » et « Mode pas à pas » n'apparaissent dans la trace d'activité. Il s'agit de fonctions d'étayages, des appuis à disposition des élèves dans l'anticipation de l'algorithme. Elles n'interviennent pas à proprement dit dans le processus de généralisation, mais permettent qu'il ne soit pas entravé par manque d'anticipation.

Figure 15 : Trace d'activité Pixel'Art Initiale – Fonctions d'étayage

Le lieu de « dépôt » ou le déplacement des blocs dans la zone de programmation ainsi que la modification du nombre d'itération de la structure de boucle sont deux points importants dans l'analyse de l'activité de programmation et dans le suivi de l'état du programme grâce à la trace de celle-ci.

La version mise à disposition (Declercq, 2020) de l'environnement d'apprentissage humain Pixel'Art nous offre la possibilité d'effectuer ses modification et de créer une progression d'activités spécifiques.

D. ENRICHISSEMENT DE LA TRACE D'ACTIVITE

L'analyse des traces d'activité des élèves en situation de généralisation de pattern sous l'environnement Pixel'Art doit permettre d'observer le moment d'apparition de la boucle et d'identifier les stratégies des élèves. Nous proposons donc plusieurs modifications afin de permettre une **lecture des états successifs du programme et de sa version définitive, la production de l'élève.**

○ AJOUT D'ACTION DE PROGRAMMATION

- **Modification du bloc** : correspond à la modification du nombre de répétitions de la structure de boucle

○ ÉTAT DU PROGRAMME ET DE LA ZONE DE PROGRAMMATION

Nous procédons à l'ajout d'une colonne « État du programme ». Grâce à un repérage du lieu de dépôt (dans ou hors de la structure de séquence du dessin) des instructions dans la zone de programmation, il est possible de connaitre l'état de la zone de programmation et par là celui du programme à chaque action de programmation.

Il est ainsi possible de suivre l'évolution du programme au fil de l'activité de programmation des élèves à partir de l'enregistrement de l'activité.

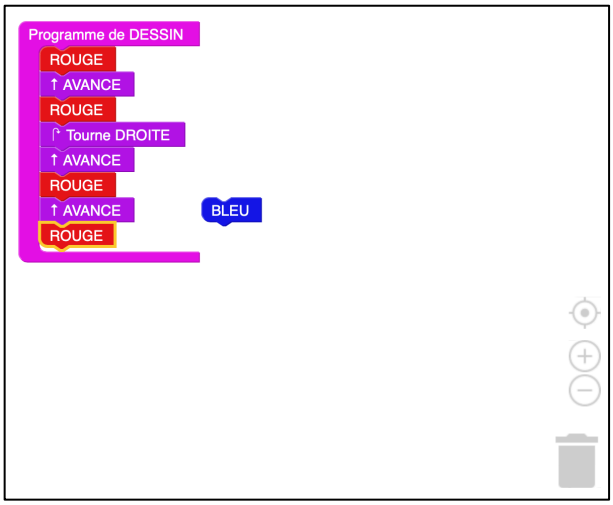


Figure 16 : Production de l'enregistrement d'activité (Annexe 2)

Pour illustrer les ajustements, nous mettons en annexe l'enregistrement de l'activité correspondant à la production ci-dessus (Figure 16) et en commentons un extrait (Figure 17).

18:35:28	Création d'un bloc dessin_bleu	[0,0] dessin_bleu [20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite [-151,169] dessin_bleu
18:35:29	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite [201.65097045898438,195] dessin_bleu
18:35:37	Création d'un bloc dessin_rouge	[0,0] dessin_rouge [-158,22] dessin_rouge [20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite [201.65097045898438,195] dessin_bleu
18:35:38	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_rouge [201.65097045898438,195] dessin_bleu
18:35:50	Création d'un bloc dessin_avance	[0,0] dessin_avance [20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_rouge [-148,213] dessin_avance [201.65097045898438,195] dessin_bleu
18:35:51	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge [201.65097045898438,195] dessin_bleu

Figure 17 : Extrait de l'enregistrement d'activité (Annexe 2)

Le repérage des instructions dans la zone de programmation nous permet de :

- savoir qu'un bloc est dans la séquence de programmation du dessin s'il n'est pas précédé de coordonnées. Par exemple, le bloc bleu a été créé mais n'est pas dans la séquence de programmation du dessin.

- connaitre l'ordre des instructions de la séquence de programmation du dessin. Par exemple, le bloc dessin_avance a été inséré entre deux instructions de la séquence.

L'enregistrement de l'activité nous donne accès aux actions de programmation qui ont permis aux élèves d'aboutir à leur production. Dans le cas d'une situation de généralisation algorithmique il est possible grâce à la trace de la caractériser encore plus finement le niveau de généralisation, notamment lors de la création de boucle.

5. RESULTATS

5.1. ANALYSE DES TRACES D'ACTIVITE PIXEL ART

5.1.1. ACQUISITION DE LA BOUCLE

Le tableau en Annexe 3 permet une description des productions des élèves à un niveau quantitatif.

Tout d'abord, l'enregistrement de l'activité nous permet d'observer que, bien qu'elle ne leur permette pas toujours de réussir la tâche, **100% des élèves ont recours au moins une fois à la boucle** au cours de la situation expérimentale.

La situation expérimentale a été pensée comme une progression de sous-types de tâches permettant ainsi de provoquer l'utilisation de la boucle en fonction de la valeur de $\mathbf{V_{Nombre\ de\ répétitions}}$. Il est donc intéressant de voir à partir de quelle tâche l'action **Création d'un bloc controls_repeat** apparait dans l'activité des élèves.

Tâche	Nombre d'élèves utilisant l'instruction « boucle » ?
1	3
2	1
3	8
4	12
5	2
Total	26

Figure 18 : Fréquences du niveau d'apparition de Création d'un bloc controls_repeat

Le nombre de répétitions de la tâche 4 semble bien influer sur le recours à la boucle par les élèves. En effet, près de la **moitié de la classe** créé une boucle pour cette tâche alors qu'ils ne l'avaient pas fait auparavant. Bien que le découpage ne soit pas précis, ce premier résultat montre le **rôle épistémologique** des valeurs de la variable.

L'utilisation de la boucle ne signifie pas pour autant la réussite de la tâche grâce à celle-ci. Le tableau (Annexe 3) nous permet d'observer des élèves qui utilisent la boucle à un moment donné dans la programmation du dessin puis finissent par l'**abandonner** ou alors sont mis **en échec** dans la résolution de la tâche. En effet, on note pour la 3ème tâche : 3 abandons de la boucle et 1 échec de l'activité après utilisation de la boucle et pour la 4ème tâche : 3 abandons de la boucle et 3 échecs de l'activité après utilisation de la boucle.

Si cette première analyse nous a permis d'observer un seuil du nombre de répétitions au-delà duquel le recours à la boucle était particulièrement visible, nous posons encore plusieurs questions :

Quelle(s) technique(s) appliquent les 15 élèves qui réussissent la tâche 4 avec itération ?
Quelle(s) technique(s) permet à deux élèves de n'utiliser la boucle qu'à la tâche 5 ?
Quelle(s) technique(s) appliquent les élèves qui réussissent la tâche 5 ?

Une analyse plus qualitative des enregistrements des traces d'activités est nécessaire pour apporter des éléments de réponses à ces questions.

5.1.2. GENERALISATION DU MOTIF

Face à un type de tâches T, l'institution attend d'un élève la mise en place d'une ou plusieurs techniques qui relèvent d'une organisation mathématique institutionnelle associées au type de tâches T. Le modèle praxéologique de référence (Modèle praxéologique de référence p.15) décrit ces organisations. Il est pour autant possible que le rapport personnel de l'élève à ce type de tâches ne soit non conforme au rapport institutionnel.

A. PRAXEOLOGIE PERSONNELLE

Cela se traduit par la mise en œuvre d'une technique soit, scientifiquement valide mais non adéquate institutionnellement soit, scientifiquement non valide (Nguyen et al., 2007). On appelle praxéologie personnelle le quadruplet d'organisation praxéologique constitué de quatre composantes suivantes (Croset, Chaachoua, 2016) :

- Un **type de tâches personnel** est l'ensemble des tâches que le sujet perçoit comme similaires, et qui provoque chez lui l'application d'une technique.

- Une **technique personnelle** permet de résoudre un seul type de tâches personnel : elle peut être correcte – attendue ou non dans l'institution, erronée. Elle doit présenter une certaine stabilité pour accomplir un type de tâches personnel dans le temps.

- Une **technologie** personnelle permet de justifier, de produire, de rendre intelligible, de contrôler et d'adapter une technique personnelle.

- Une **théorie personnelle** justifie la technologie personnelle.

L'analyse des traces d'activités nous donne accès aux techniques mises en œuvre par les élèves et donc éventuellement aux praxéologies personnelles, en particulier aux valeurs des situations auxquelles les élèves sont sensibles. Ces valeurs de variables ne correspondent pas nécessairement à notre analyse a priori qui repose sur les attentes institutionnelles

La tâche 1 est réussie sans échec par 8 élèves qui appliquent tous la technique τ_2. Les élèves qui ne réussissent pas la tâche 1 sans échec appliquent la **technique $\boldsymbol{\tau_1}$**. On peut notamment observer dans les traces d'activité une **alternance de création et destruction** de bloc ponctuée pas des exécutions du programme erroné. L'élève 5 (Annexe 3) réalise 25 essais/erreurs. Toutefois l'enregistrement de son activité de programmation (Annexe 4) nous invite à nuancer ce nombre et en proposer une interprétation. En effet, on peut observer plusieurs exécutions successives du programme, jusqu'à 6 à la suite.

Nous émettons l'hypothèse que l'élève partage une sorte de croyance du « bug » qui viendrait justifier que l'échec à l'activité ne vient pas de l'écriture du programme et que de l'exécuter plusieurs fois pourrait « debugger » l'ordinateur et permettre à l'élève de réussir l'activité. L'exécution successive du programme suite à un échec est appliquée par plusieurs élèves au cours de la 1ère tâche mais aussi au cours des suivantes.

Il semble donc que dans le cas d'une programmation naïve certains élèves mettent en œuvre une technique personnelle pour le sous type de tâches « Exécuter un programme » et qu'ils justifient par une technologie personnelle qui est la croyance du bug. Elle vient donc enrichir le modèle praxéologique.

Programmation naïve	
τ_1 : - ajouter/supprimer des instructions - exécuter le programme	$\tau_1{}'$: - ajouter/supprimer des instructions - exécuter le programme le programme plusieurs fois successives
θ_1 : - Notion de séquence - Notion de motif - Déplacement relatif	$\theta_1{}'$: - Notion de séquence - Notion de motif - Déplacement relatif - Notion de bug

La technique τ_2 est globalement appliquée par tous les élèves pour **la tâche 2** et elle permet à 5 élèves de la réussir sans échec.

Les enregistrements d'activités pour les tâches 3 et 4 sont marquées par une forte apparition de la boucle. Toutefois comme nous l'avons déjà indiqué, tous ne réussiront pas la tâche grâce à un programme contenant une boucle.

En effet, certains élèves abandonnent la boucle faute de trop d'échecs. Pendant l'expérimentation, un élève qui avait pourtant eu recours à la boucle m'explique :

E : « Oui, répéter plusieurs fois ça peut permettre d'aller plus vite, par exemple pour un programme que je vais répéter plein de fois, je vais utiliser répéter ».

E : « Bon pour une fois ce n'est pas très efficace »

P : « Pourtant pour cette activité je ne le vois pas dans ton programme »

E : « Mais là, je l'ai enlevé car ça ne fonctionnait pas très bien, et je ne le maîtrisais pas trop… »

Les traces d'activités montrent qu'une fois la boucle crée, les élèves y intègrent des instructions qu'ils pensent être constitutives du motif algorithmique puis ils enchainent les essais/erreur successifs en modifiant le corps de l'itération à chaque échec.

Cela met en évidence deux choses :

- L'importance de l'identification du motif et de sa transposition algorithmique
- Le retour à une programmation pas à pas lorsque la technique τ_3 n'est pas maitrisé, et ce même pour la tâche 4. Nous avions pourtant placé la tâche 4 hors de la portée pragmatique de τ_2.

Enfin, nous observons chez 6 élèves une technique que nous n'avions pas prévu *a priori*. Pour plus de lisibilité nous l'illustrons en (annexe 5) par un extrait de la trace d'activité simplifiée (sans les actions de déplacement non significatives) de l'élève 3.

On y observe la répétition d'une suite de créations d'instructions et exécution du programme. L'élève ainsi suit l'évolution de son programme tout en contrôlant visuellement chaque pas. Cela traduit la nécessité de l'élève de contrôler l'implémentation de la technique. C'est-à-dire que même si la technique est reconnue comme valide, des erreurs individuelles peuvent venir en obérer l'effectivité. (Castela, 2018)

Dans le cas d'une programmation pas à pas certains élèves mettent en œuvre une technique personnelle au type de tâches « Écrire un programme de type itératif » et qu'il justifie par une technologie personnelle qui est celle du contrôle de la technique. Elle vient donc enrichir le modèle praxéologique.

Programmation « pas à pas »	
τ_2 : - identifier le motif visuel - effectuer une correspondance « terme à terme » en répétant la programmation de la séquence d'instructions correspondant au motif algorithmique autant de fois que nécessaire - exécuter le programme	τ_2': - identifier le motif visuel - effectuer une correspondance « terme à terme » et programmer une séquence d'instructions - contrôler la technique par exécution de la séquence - répéter la programmation de la séquence d'instructions correspondant au motif algorithmique autant de fois que nécessaire et contrôler au fur et à mesure - exécuter le programme
θ_2: - Notion de séquence - Notion de motif - Déplacement relatif	θ_2: - Notion de séquence - Notion de motif - Déplacement relatif - Contrôle visuel

Enfin **les techniques τ_3 et τ_3'** sont bien appliquées pour la résolution de la tâche 5 et on les retrouve dans les traces d'activité.

La nécessité d'utiliser une boucle à la tâche 4 est perçu par l'ensemble des élèves. Pour autant, la mise en œuvre de la technique personnelle τ_2' nous rappelle les difficultés à déterminer le **corps de l'itération** et notamment l'**invariance de la séquence d'instructions** qui le compose, et modifie la portée des techniques (Figure 19).

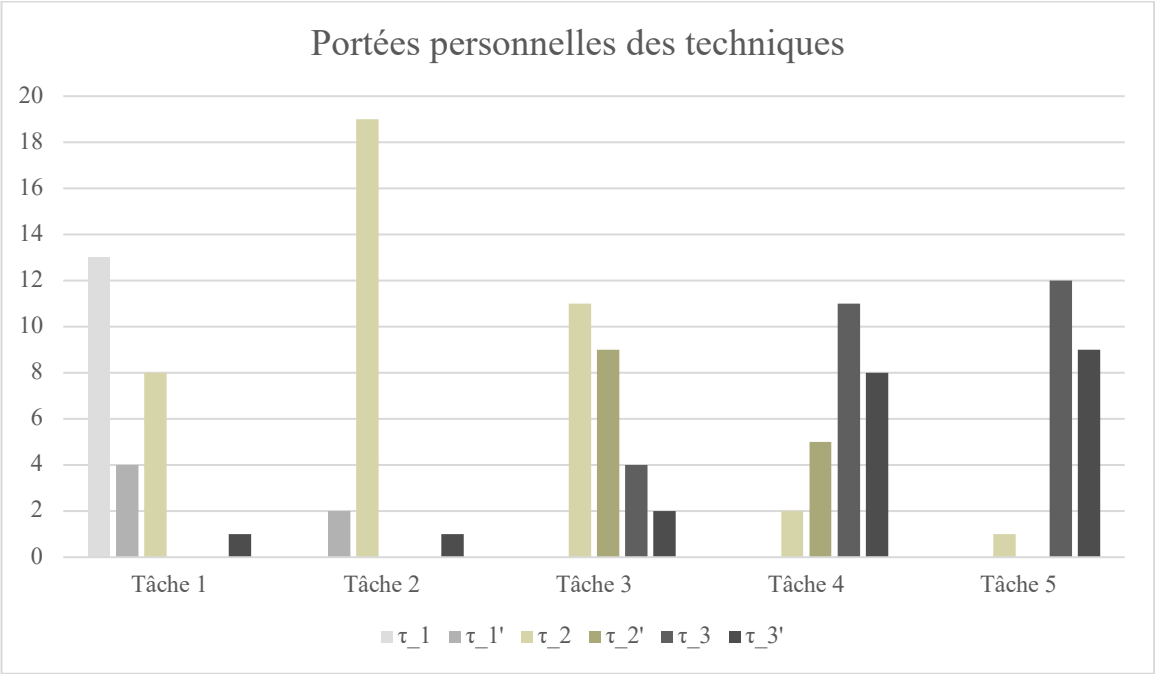


Figure 19 : Portées personnelles

Les praxéologies personnelles des élèves nous permettent d'enrichir le modèle praxéologique théorique de la généralisation de motif en contexte de programmation d'un dessin. Ainsi, nous proposons un modèle de connaissances algorithmiques qui permet de mieux caractériser le raisonnement des élèves en situation de généralisation de motif.

6. DISCUSSION ET PERSPECTIVES

Nous avons fait le choix de placer le pattern comme objet-frontière entre la pensée algébrique et la pensée algorithmique pour élaborer notre modélisation de connaissances algorithmiques. Les types de raisonnement algébriques identifiés par Radford (Radford, 2006, 2008) ont été modélisés par le quadruplet praxéologique afin de faciliter une transposition au raisonnement algorithmiques mis en œuvre en situation de généralisation d'un motif.

Les résultats de la recherche aussi bien du côté de l'algèbre (Squalli (2015) ; Vlassis (2017)) que du côté de l'algorithmique et de la programmation (Declercq & Tort, 2018 ; Léonard et al.,2022) nous ont permis d'identifier un générateur de types de tâches constitué de 5 variables fondamentales. Ce générateur a permis de concevoir une situation expérimentale permettant d'amener les élèves à la généralisation tout en prenant en considération les caractéristiques de la situation et de l'environnement. Et dont l'analyse *a posteriori* tend à monter que les situations ainsi engendrées sont pertinentes pour faire évoluer les techniques mobilisées par les élèves et donc à valider les variables retenues du générateur.

Le cadre théorique de la Théorie Anthropologique du Didactique a offert un cadre pour structurer les connaissances algorithmiques en organisations praxéologiques *a priori* enrichies par les praxéologies personnelles observées lors

d'une expérimentation dans une classe de 6e : de nouveaux ingrédients technologiques ont pu être identifiés, les portées des techniques réajustées.

Dans leur communication au colloque Didapro9, Blanvillain & Baumberger (2022) posent la question de la détection précoce des élèves en difficulté dans l'apprentissage de l'algorithmique et de la nécessité pour l'enseignant d'identifier les difficultés rencontrées lors de la recherche de solutions algorithmiques aux problèmes posés. Nous espérons ici avoir apporté quelques éléments de réponse.

L'enrichissement du modèle praxéologique *a priori* illustre à quel point dans la tâche de construction d'une boucle (Rogalski & Samurçay, 1986), la détermination du corps de l'itération et notamment l'invariance de la séquence d'instructions qui le compose, occupe une place primordiale. Cela nous invite à réfléchir aux « bonnes » pratiques (Brennan & Resnick, 2012) de programmation à développer chez les élèves mais aussi à la nécessité de les expliciter.

Enfin, l'identification du motif est un préalable au processus de généralisation aussi bien en algèbre qu'en algorithmique. L'utilisation de Pixel'Art pour l'élaboration de parcours d'apprentissage sur cette identification pourrait-elle venir soutenir le processus de généralisation de pattern en algèbre ? Cela reste une hypothèse qui mériterait d'être approfondie et testée empiriquement afin de déterminer son impact réel sur le processus de généralisation de pattern en algèbre.

Le pattern, objet à la fois des mathématiques et de l'informatique, nous permet de proposer une modélisation des connaissances algorithmiques grâce au cadre théorique de la Théorie Anthropologique du Didactique. Toutefois, il ne s'agit là que d'une petite partie des concepts et savoirs à enseigner par les enseignants de mathématiques et d'informatique. L'idée d'utiliser une science déjà développée et bien structurée semble être une piste pertinente pour approfondir notre compréhension de la discipline informatique, qui nécessite encore d'être adaptée à l'enseignement. En explorant d'autres objets-frontières nous pourrions ouvrir de nouvelles perspectives intéressantes.

BIBLIOGRAPHIE

Baron, G.-L., & Drot-Delange, B. (2016). L'informatique comme objet d'enseignement à l'école primaire française ? Mise en perspective historique. *Revue française de pédagogie*, *195*, 51-62. https://doi.org/10.4000/rfp.5032

Blanvillain, C., & Baumberger, B. (2022, mai). *Détection précoce des élèves rencontrant des difficultés d'apprentissage de l'algorithmique*. Didapro - DidaSTIC, Le Mans, France. http://hdl.handle.net/20.500.12162/5971

Bosch, M., & Chevallard, Y. (1999). La sensibilité de l'activité mathématique aux ostensifs : Objet d'étude et problématique. *Recherches en Didactique des Mathematiques*, *19*(1), 77-123.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*.

Bronner, A. (2015). Développement de la pensée algébrique avant la lettre—Apport des problèmes de généralisation et d'une analyse praxéologique. In L. Theis, *Pluralités culturelles et universalité des mathématiques : Enjeux et perspectives pour leur enseignement et leur apprentissage—Actes du colloque EMF2015 -GT3* (p. 247-264).

Brousseau, G. (1982). Ingénierie didactique : D'un problème à l'étude a priori d'une situation didactique. *Deuxième école d'été de didactique des mathématiques*.

Cai, J., & Knuth, E. (2011). *Early Algebraization : A Global Dialogue from Multiple Perspectives*. Springer Science & Business Media.

Castela, C. (2018). *Les praxéologies comme idiosyncrasies institutionnelles*. *22*(4), 86. https://hal.science/hal-03199902

Chaachoua, H. (2010). *La praxéologie comme modèle didactique pour la problématique EIAH. Etude de cas : La modélisation des connaissances des élèves* [Hdr, Université de Grenoble]. https://theses.hal.science/tel-00922383

Chaachoua, H. (2018). *T4TEL un cadre de référence didactique pour la conception des EIAH*. https://ardm.eu/wp. https://hal.science/hal-02009619

Chaachoua, H., & Bessot, A. (2019). La notion de variable dans le modèle praxéologique. *Educação Matemática Pesquisa : Revista do Programa de Estudos Pós-Graduados em Educação Matemática*, *21*. https://doi.org/10.23925/1983-3156.2019v21i4p234-247

Chevallard, Y. (1992). Concept Fondamentaux de la Didactique : Perspectives Apportees par une Approche Anthropologique. *Recherches en Didactique des Mathematiques*. https://cir.nii.ac.jp/crid/1570009750638097920

Chevallard, Y. (1998). Analyse des pratiques enseignantes et didactique des mathématiques : L'approche anthropologique. *Actes de l'UE de la Rochelle*, 91-118.

Chevallard, Y. (1999). L'analyse des pratiques enseignantes en theorie anthropologique du didactique. *Recherches en Didactique des Mathematiques*, *19*(2), 221-266.

Cogis, O., & Palaysi, J. (2021). *Algorithmes itératifs*. https://hal.science/hal-03501911

Croset, M.-C., & Chaachoua, H. (2016). Une réponse à la prise en compte de l'apprenant dans la TAD : La praxéologie personnelle. *Recherches en Didactique des Mathematiques*, *36(2)*, 161-196.

Declercq, C. (2021). Didactique de l'informatique : Une formation nécessaire [Pdf]. *Sticef*, *28*(3), 233-249. https://doi.org/10.23709/STICEF.28.3.8

Declercq, C. (2020, avril 15). *PixelArt3 · GitLab*. GitLab. https://gitlab.univ-nantes.fr/declercq-c/PixelArt3

Declercq, C., Lehuen, J., Leroux, P., Renault, V., & Sejourne, A. (2022). *L'informatique, objets d'enseignement et d'apprentissage. Quelles nouvelles perspectives pour la recherche ? Actes du colloque DIDAPRO 9*. 137 p. https://hal.science/hal-03697888

Declercq, C., & Tort, F. (2018, avril 3). *Organiser l'apprentissage de la programmation au cycle 3 avec des activités guidées et/ou créatives*. RJC EIAH 2018. https://hal.science/hal-01765408

Declercq, C., & Zeyringer, M. (2018, juin 27). *Analyse de l'activité des élèves dans l'environnement PixelArt pour l'apprentissage de la séquence et de la répétition au cycle 3*. ETIC3. https://hal.science/hal-01826065

Demonty, I., Fagnant, A., & Vlassis, J. (2015). Le développement de la pensée algébrique : Quelles différences entre les raisonnements mis en place par les élèves avant et après l'introduction de l'algèbre ? In L. Theis, *Pluralités culturelles et universalité des mathématiques : Enjeux et perspectives pour leur enseignement et leur apprentissage—Actes du colloque EMF2015 -GT3* (p. 265-279). https://orbi.uliege.be/handle/2268/192905

Dörfler, W. (1991). Forms and Means of Generalization in Mathematics. In A. J. Bishop, S. Mellin-Olsen, & J. Van Dormolen (Éds.), *Mathematical Knowledge : Its Growth Through Teaching* (Vol. 10, p. 61-85). Springer Netherlands. https://doi.org/10.1007/978-94-017-2195-0_4

Drot-Delange, B., & Tort, F. (2018, mars 27). *Résolution de défis et pensée informatique*. 10èmes rencontres scientifiques de l'ARDIST. https://hal.science/hal-01851772

Fluckiger, C. (2019). *Une approche didactique de l'informatique scolaire.*

Gaubert-Macon, C. (2022). Pratique de l'informatique aux cycles 3 et 4. *Rapport Inspection générale de l'Éducation, du Sport et de la Recherche (IGÉSR)*. https://www.education.gouv.fr/media/120388/download

Gouws, L. A., Bradshaw, K., & Wentworth, P. (2013). Computational thinking in educational activities : An evaluation of the educational game light-bot. *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, 10-15. https://doi.org/10.1145/2462476.2466518

Grover, S. (2017). *Assessing Algorithmic and Computational Thinking in K-12 : Lessons from a Middle School Classroom* (p. 269-288). https://doi.org/10.1007/978-3-319-52691-1_17

Grugeon, B. (1995). *Etude des rapports institutionnels et des rapports personnels des élèves à l'algèbre élémentaire dans la transition entre deux cycles d'enseignement : BEP et Première G.* [Phdthesis, Université de Paris 7 Denis diderot]. https://theses.hal.science/tel-01252058

Jolivet, S., Dechaux, E., Gobard, A.-C., & Wang, P. (2023). *Construction et exploitation d'un référentiel de types de tâches d'apprentissage de la programmation*. EIAH2023 : 11ème Conférence sur les Environnements Informatiques pour l'Apprentissage Humain.

Kaput, J. J. (2008). What is algebra ? In D. W. Carraher, M. L. Blanton, & J. J. Kaput, *Algebra in the Early Grades* (p. 5-18). Routledge.

Kieran, C. (1992). The learning and teaching of school algebra. In *Handbook of research on mathematics teaching and learning : A project of the National Council of Teachers of Mathematics* (p. 390-419). Macmillan Publishing Co, Inc.

Kieran, C. (2007). Learning and teaching algebra at the middle school through college levels : Building meaning for symbols and their manipulation. *Second handbook of research on mathematics teaching and learning*, *2*, 707-762.

Laborde, C., Mejias, B., & Balacheff, N. (1985). Genèse du concept d'itération : Une approche expérimentale. *Enfance*, *38*(2), 223-239. https://doi.org/10.3406/enfan.1985.2882

Laborne, C., Mejias, B., & Balachef, N. (1985). Genèse du concept d'itération : Une approche expérimentale. *Enfance*, *38*(2), 223-239. https://doi.org/10.3406/enfan.1985.2882

Lagrange, J.-B., & Guy, M.-N. (2015). Planification et connaissances mathématiques dans une situation d'apprentissage au lycée : L'algorithme de Kaprekar. *Petit x*, *97*, 45-70.

Larguier, M. (2015). Première rencontre avec l'algèbre. In *Pluralités culturelles et universalité des mathématiques : Enjeux et*

perspectives pour leur enseignement et leur apprentissage—Actes du colloque EMF2015 -GT3 (p. 313-333).

Lee, L. (1996). An Initiation into Algebraic Culture through Generalization Activities. In N. Bernarz, C. Kieran, & L. Lee (Éds.), *Approaches to Algebra* (p. 87-106). Springer Netherlands. https://doi.org/10.1007/978-94-009-1732-3_6

Léonard, M., Secq, Y., Peter, Y., & Fluckiger, C. (2022). *Pensée informatique : Approche didactique de l'identification de motifs*. https://lilloa.univ-lille.fr/handle/20.500.12210/75574

Li, F., Wang, X., He, X., Cheng, L., & Wang, Y. (2022). The effectiveness of unplugged activities and programming exercises in computational thinking education : A Meta-analysis. *Education and Information Technologies*, *27*(6), 7993-8013. https://doi.org/10.1007/s10639-022-10915-x

Liljedahl, P. (2004). Repeating pattern or number pattern : The distinction is blurred. *Focus on Learning Problems in Mathematics*, *26*, 24-42.

Margolinas, C. (1995). *La structuration du milieu et ses apports dans l'analyse a posteriori des situations*. 89.

Mason, J. (1996). Expressing Generality and Roots of Algebra. In N. Bernarz, C. Kieran, & L. Lee (Éds.), *Approaches to Algebra : Perspectives for Research and Teaching* (p. 65-86). Springer Netherlands. https://doi.org/10.1007/978-94-009-1732-3_5

Nguyen, A. Q., Chaachoua, H., & Comiti, C. (2007). De l'usage de la TAD pour l'analyse des erreurs. *Sociedad, Escuela y matematicas. Aportaciones de la teoria Antropologica de lo Didactico*, 621-639.

Peter, Y., Léonard, M., & Secq, Y. (2019, juin 4). *Reconnaissance de Motifs et Répétitions : Introduction à la Pensée Informatique*. Environnements Informatiques pour l'Apprentissage Humain. https://hal.science/hal-02151035

Piolti-Lamorthe, C., Roubin, S., & Trgalová, J. (2023). *Des patterns dans les classes !*

Pixel'Art. (s. d.). Consulté 20 mai 2023, à l'adresse https://declercq-c.univ-nantes.io/PixelArt3/

Radford, L. (2006). Algebraic thinking and the generalization of patterns : A semiotic perspective. *Proceedings of the Twenty Eighth Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*, *1*.

Radford, L. (2008). Iconicity and contraction : A semiotic investigation of forms of algebraic generalizations of patterns in different contexts. *ZDM*, *40*(1), 83-96. https://doi.org/10.1007/s11858-007-0061-0

Radford, L. (2014). The Progressive Development of Early Embodied Algebraic Thinking. *Mathematics Education Research Journal*, *26*(2), 257-277. https://doi.org/10.1007/s13394-013-0087-2

Rich, K. M., Binkowski, T. A., Strickland, C., & Franklin, D. (2018). Decomposition : A K-8 Computational Thinking Learning Trajectory. *Proceedings of the 2018 ACM Conference on International Computing Education Research*, 124-132. https://doi.org/10.1145/3230977.3230979

Rogalski, J., & Samurçay, R. (1986). Les Problèmes Cognitifs Rencontrés par des Elèves de l'Enseignement Secondaire dans l'Apprentissage de l'Informatique. *European Journal of Psychology of Education*, *1*(2), 97-110.

Rogalski, J., & Samurçay, R. (1990). Acquisition of Programming Knowledge and Skills. In *Psychology of Programming* (p. 157-174). Elsevier. https://doi.org/10.1016/B978-0-12-350772-3.50015-X

Scratch—Imagine, Program, Share. (s. d.). Consulté 25 juin 2023, à l'adresse https://scratch.mit.edu/

Selby, C., & Woollard, J. (2013). *Computational thinking : The developing definition* [Monograph]. University of Southampton (E-prints). https://eprints.soton.ac.uk/356481/

Squalli, H. (2000). *Une reconceptualisation du curriculum d'algèbre dans l'éducation de base* [Université Laval]. http://www.nlc-bnc.ca/obj/s4/f2/dsk1/tape3/PQDD_0017/NQ55825.pdf?is_thesis=1&oclc_number=1006925190

Squalli, H. (2002). Le développement de la pensée algébrique à l'école primaire : Un exemple de raisonnement à l'aide de concepts mathématiques. *Instantanés mathématiques*, *XXXIX*, 4, 5, 8 et 9.

Squalli, H. (2015). La généralisation algébrique comme abstraction d'invariants essentiels. In L. Theis, *Pluralités culturelles et universalité des mathématiques : Enjeux et perspectives pour leur enseignement et leur apprentissage—Actes du colloque EMF2015 -GT3* (p. 346-356).

Star, S. L., & Griesemer, J. R. (1989). Institutional Ecology, `Translations' and Boundary Objects : Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science*, *19*(3), 387-420. https://doi.org/10.1177/030631289019003001

Vergnaud, G. (1988). Long terme et court terme dans l'apprentissage de l'algèbre. In *Actes du premier colloque franco-allemand de didactique* (La Pensée Sauvage).

Vlassis, J., Demonty, I., & Squalli, H. (2017). Développer la pensée algébrique à travers une activité de généralisation basée sur des motifs (patterns) figuratifs. *Nouveaux cahiers de la recherche en éducation*, *20*(3), 131-155. https://doi.org/10.7202/1055731ar

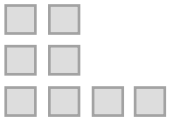
Wagner, S., & Kieran, C. (2018). *Research Issues in the Learning and Teaching of Algebra : The Research Agenda for Mathematics Education, Volume 4*. Routledge.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

ANNEXE 1

Activité « Post-it ! »

Voici une suite de dessins réalisés à l'aide de petits carrés :

			
Dessin n°1	Dessin n°2	Dessin n°3	Dessin n° 4

1. Continue la suite en dessinant le dessin n°4 dans la case vide ci-dessus.

2. Alexandre est un élève d'une autre classe. Il voudrait obtenir le dessin n°7, mais il n'a pas vu les premiers dessins.

    a. Explique-lui comment il doit faire.

    b. Finalement, combien de carré doit-il dessiner ?

3. Alexandre aimerait réaliser le dessin n°100 sur la fenêtre de sa classe avec des post-it. De combien de post-it aura-t-il besoin ?

4. Dans la classe d'Alexandre, il y a une boite contenant des papiers sur lesquels est chaque fois indiqué un nombre. Alexandre va choisir un papier au hasard dans la boite. Le nombre indiqué sera le numéro d'un dessin de la suite précédente.

    Écris-lui un message qui lui permettent de savoir comment il peut calculer le nombre post-it nécessaires pour réaliser le dessin.

ANNEXE 2

Enregistrement de l'activité

18:35:10	Début activité 1	
18:35:10	Création d'un bloc dessin_debut	[20,20] dessin_debut
18:35:10	Déplacement d'un bloc	[20,20] dessin_debut
18:35:13	Création d'un bloc dessin_rouge	[0,0] dessin_rouge [-152,16] dessin_rouge [20,20] dessin_debut
18:35:14	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge
18:35:15	Création d'un bloc dessin_avance	[0,0] dessin_avance [20,20] dessin_debut,dessin_rouge [-162,214] dessin_avance
18:35:16	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance
18:35:21	Création d'un bloc dessin_rouge	[0,0] dessin_rouge [-163,19] dessin_rouge [20,20] dessin_debut,dessin_rouge,dessin_avance
18:35:22	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge
18:35:25	Création d'un bloc dessin_droite	[0,0] dessin_droite [20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge [-72,238] dessin_droite
18:35:26	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite
18:35:28	Création d'un bloc dessin_bleu	[0,0] dessin_bleu [20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite [-151,169] dessin_bleu
18:35:29	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite [201.65097045898438,195] dessin_bleu
18:35:37	Création d'un bloc dessin_rouge	[0,0] dessin_rouge [-158,22] dessin_rouge [20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite [201.65097045898438,195] dessin_bleu
18:35:38	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_rouge [201.65097045898438,195] dessin_bleu
18:35:50	Création d'un bloc dessin_avance	[0,0] dessin_avance [20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_rouge [-148,213] dessin_avance [201.65097045898438,195] dessin_bleu
18:35:51	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge [201.65097045898438,195] dessin_bleu
18:35:51	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge [201.65097045898438,195] dessin_bleu
18:35:51	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge [201.65097045898438,195] dessin_bleu
18:36:00	Création d'un bloc dessin_avance	[0,0] dessin_avance [20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge [201.65097045898438,195] dessin_bleu [-160,222] dessin_avance
18:36:00	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance [201.65097045898438,195] dessin_bleu
18:36:03	Création d'un bloc dessin_rouge	[0,0] dessin_rouge [-157,23] dessin_rouge [20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance [201.65097045898438,195] dessin_bleu
18:36:04	Déplacement d'un bloc	[20,20] dessin_debut,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge [201.65097045898438,195] dessin_bleu
18:36:07	Dessine OK Activité 1	

ANNEXE 3

Traitement des traces d'activité de la situation expérimentale

Activité	Compte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	Total
1	échecs	0	0	15	1	25	16	7	20	3	1	12	12	0	0	0	4	1	1	1	17	9	15	60	0	0	0	220
1	échecs avec iteration	0	0	2	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	11
1	réussites	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
1	réussites avec iteration	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	échecs	0	0	32	2	26	26	12	28	2	0	3	12	1	18	2	10	12	141	68	23	39	9	51	0	17	0	534
2	échecs avec iteration	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	3
2	réussites	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
2	réussites avec iteration	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	échecs	3	8	9	13	93	25	7	67	0	6	1	12	44	108	10	12	8	59	13	59	23	10	44	5	8	11	658
3	échecs avec iteration	0	4	0	0	45	0	0	0	0	6	0	0	0	19	0	0	4	2	0	0	6	0	0	5	0	0	91
3	réussites	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
3	réussites avec iteration	0	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	5
4	échecs	7	32	6	32	29	29	20	32	4	6	16	14	6	25	40	13	29	84	34	29	0	17	21	0	30	21	576
4	échecs avec iteration	1	32	0	13	18	13	20	0	4	6	8	14	1	5	31	11	4	49	19	6	0	8	5	0	0	21	289
4	réussites	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	0	0	1	1	1	1	1	22
4	réussites avec iteration	0	1	0	0	0	1	1	0	1	1	1	1	1	0	1	1	1	0	1	0	0	1	1	1	0	1	16
5	échecs	56	10	30	38	0	28	16	30	1	19	39	26	37	0	30	32	55	0	25	0	0	33	9	29	69	14	626
5	échecs avec iteration	55	10	ces	29	0	27	16	27	1	18	39	8	36	0	30	32	13	0	25	0	0	32	9	29	51	14	501
5	réussites	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	6
5	réussites avec iteration	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	6

Utilisation de la boucle
2 réussites avec boucle
3 réussites avec boucle
4 réussites avec boucle
5 réussites avec boucle

Abandon de la boucle
Echec de l'activité
Activité non traitée

ANNEXE 4

Trace d'activité du compte 5 pour la Tâche 1

12:22:58	Début activité 1	
12:22:59	Création d'un bloc dessin_debut	[20,20] dessin_debut
12:22:59	Déplacement d'un bloc	[20,20] dessin_debut
12:23:05	Création d'un bloc dessin_avance	[0,0] dessin_avance [20,20] dessin_debut [-167,207] dessin_avance
12:23:06	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance
12:23:07	Création d'un bloc dessin_droite	[0,0] dessin_droite [20,20] dessin_debut,dessin_avance [-166,325] dessin_droite
12:23:09	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_droite
12:23:13	Création d'un bloc dessin_avance	[0,0] dessin_avance [20,20] dessin_debut,dessin_avance,dessin_droite [-168,214] dessin_avance
12:23:14	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_droite,dessin_avance
12:23:15	Création d'un bloc dessin_avance	[0,0] dessin_avance [20,20] dessin_debut,dessin_avance,dessin_droite,dessin_avance [-155,213] dessin_avance
12:23:16	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_droite,dessin_avance,dessin_avance
12:23:17	Dessine	
12:23:22	Dessine	
12:23:23	Dessine	
12:23:23	Dessine	
12:23:23	Dessine	
12:23:23	Dessine	
12:23:24	Dessine	
12:23:24	Dessine	
12:23:25	Dessine	
12:23:34	Déplacement d'un bloc	[0,0] dessin_avance [20,20] dessin_debut,dessin_avance,dessin_droite,dessin_avance [49,101] dessin_avance,dessin_avance
12:23:35	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_droite
12:23:35	Destruction d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_droite
12:23:36	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_droite [57,74] dessin_droite
12:23:37	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance
12:23:37	Destruction d'un bloc	[20,20] dessin_debut,dessin_avance
12:23:40	Création d'un bloc dessin_rouge	[-165,4] dessin_rouge [0,0] dessin_rouge [20,20] dessin_debut,dessin_avance
12:23:42	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge
12:23:43	Création d'un bloc dessin_droite	[0,0] dessin_droite [20,20] dessin_debut,dessin_avance,dessin_rouge [-157,319] dessin_droite
12:23:45	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite
12:23:46	Création d'un bloc dessin_avance	[0,0] dessin_avance [20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite [-148,209] dessin_avance
12:23:50	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite
12:23:50	Destruction d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite
12:23:51	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite [36,103] dessin_droite
12:23:52	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge
12:23:52	Destruction d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge
12:23:52	Création d'un bloc dessin_avance	[0,0] dessin_avance [20,20] dessin_debut,dessin_avance,dessin_rouge [-167,217] dessin_avance
12:23:53	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance
12:23:55	Création d'un bloc dessin_rouge	[0,0] dessin_rouge [-159,9] dessin_rouge [20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance
12:23:57	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:23:58	Création d'un bloc dessin_droite	[0,0] dessin_droite [20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge [-169,320] dessin_droite
12:23:59	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite
12:24:02	Création d'un bloc dessin_avance	[0,0] dessin_avance [20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite [-165,212] dessin_avance
12:24:02	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance
12:24:06	Création d'un bloc dessin_rouge	[0,0] dessin_rouge [-156,11] dessin_rouge [20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance
12:24:08	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge
12:24:09	Création d'un bloc dessin_avance	[0,0] dessin_avance [20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge [-140,222] dessin_avance
12:24:10	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance
12:24:11	Création d'un bloc dessin_rouge	dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance
12:24:13	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:15	Dessine	
12:24:24	Déplacement d'un bloc	dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:25	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:25	Déplacement d'un bloc	dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:27	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:27	Déplacement d'un bloc	dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:28	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge [184,114] dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:28	Déplacement d'un bloc	dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:29	Déplacement d'un bloc	dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:29	Déplacement d'un bloc	dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:30	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge [184,114] dessin_avance [154,252] dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:30	Destruction d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge [184,114] dessin_avance [154,252] dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:31	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge [154,252] dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:31	Destruction d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge [154,252] dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:33	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:34	Dessine	
12:24:41	Déplacement d'un bloc	[0,0] dessin_rouge [20,20] dessin_debut,dessin_avance,dessin_rouge [43,79] dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:42	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:44	Dessine	
12:24:45	Déplacement d'un bloc	[0,0] dessin_rouge [20,20] dessin_debut,dessin_avance [45,40] dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:46	Déplacement d'un bloc	[20,20] dessin_debut [62,231] dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:48	Déplacement d'un bloc	[20,20] dessin_debut [104,188] dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:49	Déplacement d'un bloc	dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:49	Déplacement d'un bloc	[20,20] dessin_debut [104,188] dessin_avance,dessin_rouge [74,340] dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:50	Déplacement d'un bloc	[20,20] dessin_debut [104,188] dessin_avance,dessin_rouge [107,219] dessin_rouge [74,340] dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:50	Déplacement d'un bloc	[20,20] dessin_debut [104,188] dessin_avance [131,245] dessin_rouge [74,340] dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:52	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance [131,245] dessin_rouge [74,340] dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:24:57	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge [74,340] dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:01	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:03	Dessine	
12:25:16	Déplacement d'un bloc	[0,0] dessin_rouge [20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge [46,151] dessin_rouge,dessin_avance,dessin_rouge
12:25:16	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite,dessin_avance [52,197] dessin_rouge,dessin_avance,dessin_rouge
12:25:17	Déplacement d'un bloc	dessin_rouge,dessin_avance,dessin_rouge
12:25:17	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite,dessin_avance [52,197] dessin_rouge,dessin_avance,dessin_rouge
12:25:18	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite,dessin_avance [43,126] dessin_avance [52,197] dessin_rouge,dessin_avance,dessin_rouge
12:25:18	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite [60,154] dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:18	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite [60,154] dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:20	Création d'un bloc dessin_rouge	dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:21	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite,dessin_rouge [60,154] dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:22	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite,dessin_rouge [77,180] dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:24	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite,dessin_rouge,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:25	Dessine	
12:25:26	Dessine	
12:25:27	Dessine	
12:25:27	Dessine	
12:25:28	Dessine	
12:25:28	Dessine	
12:25:28	Dessine	
12:25:31	Déplacement d'un bloc	dessin_rouge,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:32	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite [93,155] dessin_rouge,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:33	Déplacement d'un bloc	dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:33	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite [93,155] dessin_rouge [119,224] dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:35	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite [119,224] dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:35	Destruction d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite [119,224] dessin_avance,dessin_rouge,dessin_avance,dessin_rouge
12:25:37	Déplacement d'un bloc	[20,20] dessin_debut,dessin_avance,dessin_rouge,dessin_droite,dessin_avance,dessin_rouge,dessin_avance,dessin_rouge

ANNEXE 5

Extrait de la trace simplifiée de l'élève 8

11:53:36	Début activité 4
11:53:36	Création d'un bloc dessin_debut
11:53:55	Création d'un bloc dessin_rouge
11:54:10	Création d'un bloc dessin_avance
11:54:14	Dessine
11:54:24	Création d'un bloc dessin_rouge
11:54:30	Création d'un bloc dessin_droite
11:54:33	Dessine
11:54:37	Création d'un bloc dessin_avance
11:54:39	Dessine
11:54:41	Création d'un bloc dessin_rouge
11:54:46	Création d'un bloc dessin_rouge
11:54:48	Création d'un bloc dessin_avance
11:54:50	Création d'un bloc dessin_rouge
11:54:53	Dessine
11:55:08	Création d'un bloc dessin_gauche
11:55:11	Dessine
11:55:14	Création d'un bloc dessin_avance
11:55:15	Création d'un bloc dessin_rouge
11:55:20	Dessine
11:55:33	Création d'un bloc dessin_droite
11:55:45	Dessine
11:56:00	Création d'un bloc dessin_avance
11:56:03	Création d'un bloc dessin_rouge
11:56:07	Dessine
11:56:21	Création d'un bloc dessin_gauche
11:56:23	Création d'un bloc dessin_avance
11:56:25	Création d'un bloc dessin_rouge
11:56:28	Dessine
11:56:33	Création d'un bloc dessin_droite
11:56:38	Création d'un bloc dessin_avance
11:56:42	Création d'un bloc dessin_rouge
11:57:06	Dessine
11:57:08	Création d'un bloc dessin_gauche
11:57:11	Création d'un bloc dessin_avance
11:57:14	Création d'un bloc dessin_rouge
11:57:21	Dessine
11:57:27	Création d'un bloc dessin_droite
11:57:30	Création d'un bloc dessin_avance
11:57:33	Création d'un bloc dessin_rouge
11:57:46	Dessine
11:57:48	Création d'un bloc dessin_gauche
11:57:51	Création d'un bloc dessin_avance
11:57:54	Création d'un bloc dessin_rouge
11:57:58	Dessine
11:58:02	Création d'un bloc dessin_droite
11:58:05	Création d'un bloc dessin_avance
11:58:10	Création d'un bloc dessin_rouge
11:58:13	Dessine
11:58:15	Création d'un bloc dessin_droite
11:58:19	Création d'un bloc dessin_avance
11:58:22	Création d'un bloc dessin_rouge
11:58:25	Dessine
11:58:33	Création d'un bloc dessin_gauche
11:58:40	Dessine
11:58:54	Création d'un bloc dessin_droite
11:58:57	Création d'un bloc dessin_avance
11:59:03	Création d'un bloc dessin_rouge
11:59:07	Dessine
11:59:11	Création d'un bloc dessin_gauche
11:59:14	Création d'un bloc dessin_avance
11:59:17	Création d'un bloc dessin_rouge
11:59:40	Dessine
11:59:43	Création d'un bloc dessin_rouge
11:59:49	Dessine
12:00:13	Création d'un bloc dessin_gauche
12:00:18	Création d'un bloc dessin_droite
12:00:24	Création d'un bloc dessin_avance
12:00:28	Création d'un bloc dessin_rouge
12:00:33	Dessine
12:00:37	Création d'un bloc dessin_gauche
12:00:40	Création d'un bloc dessin_avance
12:00:43	Création d'un bloc dessin_rouge
12:00:47	Dessine
12:01:01	Création d'un bloc dessin_droite
12:01:06	Création d'un bloc dessin_avance
12:01:13	Création d'un bloc dessin_rouge
12:01:21	Dessine
12:01:27	Création d'un bloc dessin_gauche
12:01:31	Création d'un bloc dessin_avance
12:01:45	Création d'un bloc dessin_rouge
12:01:49	Dessine
12:01:53	Création d'un bloc dessin_avance
12:02:00	Création d'un bloc dessin_droite
12:02:04	Création d'un bloc dessin_avance
12:02:08	Dessine
12:02:15	Création d'un bloc dessin_rouge
12:02:29	Dessine
12:02:34	Création d'un bloc dessin_gauche
12:02:40	Création d'un bloc dessin_avance
12:03:05	Création d'un bloc dessin_rouge
12:03:25	Dessine
12:03:30	Création d'un bloc dessin_droite
12:03:44	Création d'un bloc dessin_rouge
12:03:49	Dessine
12:03:55	Création d'un bloc dessin_avance
12:04:00	Dessine
12:04:02	Création d'un bloc dessin_rouge
12:04:07	Dessine
12:04:11	Création d'un bloc dessin_droite
12:04:20	Dessine
12:04:29	Création d'un bloc dessin_avance
12:04:44	Dessine
12:04:47	Création d'un bloc dessin_rouge
12:05:03	Création d'un bloc dessin_rouge
12:05:09	Dessine OK Activité 4
12:05:15	Dessine OK Activité 4
12:05:31	Début activité 5

Université Grenoble Alpes

Institut national supérieur du professorat et de l'éducation Académie de Grenoble

Année universitaire 2022-2023

Master 2 *Métiers de l'enseignement, de l'éducation et de la formation*

Mention Pratique et Ingénierie de la Formation
Parcours : Didactique des sciences et du numérique

Titre du mémoire : Élaboration d'un modèle praxéologique de connaissances algorithmiques à partir d'un objet-frontière avec l'algèbre : Le pattern

Auteur : Gaëlle Walgenwitz

Résumé :

Ce mémoire explore le potentiel du pattern en tant qu'objet-frontière entre l'algèbre et l'algorithmique pour élaborer un modèle praxéologique des connaissances algorithmiques. Grâce aux résultats de recherches sur les types de raisonnement algébriques et en utilisant la Théorie Anthropologique du Didactique (TAD) comme référence théorique, nous proposons une organisation praxéologique de la généralisation de motif en contexte de programmation.

La conception et mise en œuvre d'une situation expérimentale a permis de mener une analyse des productions et des traces d'activités des élèves, d'enrichir le modèle de connaissances théorique. Ce travail souligne l'importance de l'explicitation des "bonnes" pratiques de programmation et met en lumière le rôle essentiel de l'identification du motif dans le processus de généralisation, tant en algèbre qu'en algorithmique.

Enfin, ce mémoire ouvre des perspectives pour l'enseignement de l'informatique, comme l'exploration d'autres objets-frontière afin d'enrichir l'enseignement de cette discipline.

Mots clés : pattern, pensée algébrique, pensée algorithmique, praxéologie

Summary :

This paper explores the potential of pattern as boundary objects between algebra and algorithmic to develop a praxiological model of algorithmic knowledge. Drawing on research on algebraic reasoning types and using the Anthropological Theory of Didactic (ATD) as a theoretical framework, we propose a praxiological organization of pattern generalization in a programming context. The design and implementation of an experimental situation allowed for the analysis of student productions and activity traces, enriching the theoretical knowledge model. This work emphasizes the importance of explicating "good" programming practices and highlights the essential role of pattern identification in the generalization process, both in algebra and algorithmics.

Lastly, this paper opens up perspectives for computer science education, such as exploring other boundary objects to enrich the teaching of this discipline.

Keywords: pattern, algebraic thinking, algorithmic thinking, praxeology