Contribution à l'analyse de la partie "pensée informatique" du projet de programme de mathématiques pour le cycle 4

Christophe Declercq, IREMI de la Réunion

Juin 2025

Description et analyse du projet de programme

La partie "pensée informatique" du projet de programme de mathématiques pour le cycle 4 annonce une "présentation de manière progressive tout au long du cycle de la programmation impérative par blocs". La présentation détaillée décline cette orientation, dont la principale différence par rapport au programme en vigueur est d'exclure la programmation par événements.

Cette évolution nous semble raisonnable pour plusieurs raisons :

- La programmation par événements bien qu'affichée officiellement dans le curriculum prescrit, n'a jamais véritablement été mise en oeuvre de manière généralisée par les enseignants de mathématiques au collège. Le curriculum effectif, analysé sur la base des sujets de programmation au diplôme national de brevet, accorde une place marginale à ce thème [1].
- La sémantique d'exécution en pseudo-parallélisme du langage Scratch a dérouté élèves et enseignants [2]. L'incompréhension de cette sémantique fondée à la fois sur l'exécution atomique des séquences d'instructions et le pseudo-parallélisme entre scripts avec commutation à chaque boucle, a laissé croire aux usagers de Scratch à une forme d'indéterminisme, favorisant ainsi la construction mentale chez l'élève - et parfois chez l'enseignant lui-même - de l'informatique comme une technique au résultat aléatoire.
- L'intégration de l'initiation à la programmation au cours de mathématiques aussi bien en cycle 4 qu'en seconde nécessite une continuité des apprentissages reposant sur une continuité et une progressivité des concepts enseignés. La rupture collège-lycée, si le programme de collège est débarrassé de la programmation événementielle, ne pose plus comme seule difficulté que la transition de la programmation par blocs à la programmation textuelle. Cette transition a été documentée et des instruments ont été proposés pour l'accompagner [5][6].

L'affirmation "La pensée informatique est présentée sous l'angle de l'algorithmique" est déclinée de manière pertinente dans le projet de programme. En particulier, la confusion existante dans les programmes en vigueur entre développement de la pensée algébrique et développement de la pensée informatique a disparu du projet de programme, ce qui constitue une avancée remarquable. La partie problématique sur les "programmes de calcul", qui constituent un outil pour l'acquisition de la pensée algébrique, mais un obstacle pour l'acquisition de la pensée informatique, a été remplacée par l'étude des "expressions informatiques" ce qui constitue un réel progrès pour l'acquisition des concepts fondamentaux en programmation. La programmation en Scratch de programmes de calcul [1] laissait croire en effet aux élèves qu'ils ne pouvaient écrire qu'une opération par instruction d'affectation et devaient décomposer les calculs en opérations élémentaires, comme en programmation assembleur ou à l'époque de la machine à registres d'Ada Lovelace.

Ces deux aspects - disparition de la programmation événementielle et des programmes de calcul - permettent de recentrer l'apprentissage de la pensée informatique au collège sur des bases conceptuelles solides et alignées sur les concepts enseignés ensuite au lycée. Cela permet aussi la définition de progressions raisonnables basées sur l'état de l'art [7] en didactique de l'informatique.

Analyse de la progression en terme de concepts, et de capacités

L'analyse du projet de programme permet d'identifier les concepts à enseigner : instruction, expression, boucle bornée, variable, conditionnelle, boucle non bornée et blocs, présentés dans la progression de la cinquième à la troisième dans cet ordre précis. Cet ordre est logique et cohérent. C'est d'ailleurs précisément l'ordre d'introduction de ces concepts dans le parcours Algorea [7]. La boucle bornée a pu être introduite au cycle 3. Les variables sont nécessaires avant l'introduction des conditionnelles - qui auraient peu de sens en ne portant que sur des constantes. Il parait raisonnable de n'introduire les boucles non bornées (appelées boucles conditionnelles dans le projet de programme) qu'après les conditionnelles. La disparition de la programmation évenementielle a pour effet bénéfique de cesser la confusion courante - observée chez les élèves mais aussi chez les enseignants débutants - entre le `si` et le `quand`. Les blocs n'ont pas de dépendance particulière par rapport aux autres concepts. Il est cependant raisonnable de positionner cet apprentissage en fin de progression pour deux raisons précises : les blocs sont la première construction nécessitant de mettre en oeuvre une compétence d'abstraction [4], compétence réputée difficile à aborder ; l'usage des blocs prend sens quand la complexité des programmes dépasse la taille de ce qu'un élève peut voir en un seul écran.

Concernant les capacités, elles sont ordonnancées de manière pertinente en proposant systématiquement d'évaluer des programmes contenant un nouveau concept, avant de proposer d'en modifier ou d'en écrire. Ceci est cohérent à la fois avec les préconisations de la méthode PRIMM [8] et avec l'approche par compétences que nous avons développé pour l'enseignement de la programmation au lycée [4]. Par exemple, les capacités "Représenter des formules sous la forme d'une expression informatique" et "Prévoir la valeur d'une expression informatique à l'exécution" sont bien distinguées dans le programme de cinquième. De même les capacités "Écrire un programme simple donné pour réaliser un objectif ou résoudre un problème" et "Modifier un programme donné pour changer son comportement" sont bien distinguées dans le programme de quatrième, permettant ainsi aux enseignants de prévoir des progressions raisonnables.

La question de l'introduction de la variable a fait l'objet de nombreux travaux depuis les années 1980 permettant de distinguer les usages de la variable. Ces travaux ont manifestement inspiré la progression proposée avec l'introduction en quatrième de variables "constantes" issues de lecture au clavier, puis de variables accumulateur permettant de compter en modifiant la variable. Un repère de progressivité est proposé concernant le nombre de variables d'un programme : une en quatrième puis plusieurs en troisième. Cette limitation drastique peut induire une difficulté en quatrième si cela amène à enregistrer le résultat d'un calcul dans la même variable que la donnée saisie. Une limitation à deux nous paraitrait plus raisonnable pour éviter cet écueil.

Analyse instrumentale et environnements d'apprentissage

La disparition de la programmation événementielle du programme de cycle 4 permet d'ouvrir considérablement le champ des environnements d'apprentissage pouvant être utilisés. L'utilisation de Scratch n'est plus justifié ni prescrit, ce qui sera probablement apprécié par les enseignants de mathématiques de collège - et leurs élèves - qui n'avaient pas adopté cet environnement dans leurs pratiques à cause de son aspect "logiciel de découverte de la programmation pour la petite enfance et les activités péri-scolaires" lié au contexte dans lequel le langage avait été créé [9].

L'environnement Algorea, déjà cité, permet d'aborder dans le cadre sémantique de la programmation impérative par blocs, l'ensemble des concepts au programme. Cet environnement informatique pour l'apprentissage propose une scénarisation sous forme de liste de défis avec des difficultés variables, permettant de différencier les apprentissages.

L'environnement Blockly (sur lequel repose aussi Algorea) permet de développer aisément des éditeurs de programme de type "bac à sable" ou "micro-monde" alliant une zone de saisie de programmes et une zone de visualisation de l'exécution. Au vu des exemples de programmes fournis dans le projet de programme, deux types de visualisations sont nécessaires :

- une visualisation d'une scène permettant des dessins en mode "tortue" avec les primitives "avancer", "pivoter à droite, à gauche"... Les représentations dans le plan et les angles étant aussi au programme de cinquième, il est possible de proposer des primitives incluant ces notions.
- une visualisation des valeurs des variables, de la console et d'un écran, pour les programmes effectuant des calculs arithmétiques ou logiques ou manipulant des textes à saisir ou à afficher.

De tels environnements existent déjà et peuvent être adaptés pour ne proposer pour chaque classe de la cinquième à

la troisième que les constructions du langage au programme de cette classe. Une convergence des termes employés en tant qu'intitulés de blocs est souhaitable pour permettre l'échange de ressources entre enseignants et l'élaboration de sujets d'examen. Les termes utilisés par Scratch peuvent être en partie retenus, même si certains termes ne font pas consensus.

Par exemple, l'intitulé `pivote à droite` utilisé en Algoréa pourrait être préféré à l'intitulé `tourne à droite` de Scratch, qui suggère à l'élève l'idée d'un mouvement de translation associé à la rotation.

De même, l'absence de correspondance explicite entre l'instruction `demande` et la variable `réponse` en Scratch pourrait inciter à proposer une instruction de type `met reponse à ...` suggérant explicitement le statut de variable pour `reponse` et utilisant une forme compatible avec l'écriture de l'affectation.

Synthèse de l'avis

En conclusion, le projet de programme de cycle 4 pour la partie "pensée informatique" proposé par le CSP le 4 juin 2025 constitue pour nous une avancée considérable d'un point de vue épistémologique et didactique permettant d'envisager à terme une meilleure appropriation par les enseignants de mathématiques de cycle 4 de cette partie du programme, un apprentissage fondamental amélioré de la programmation par les élèves sur la base d'un programme cohérent et réaliste, et une plus grande fluidité lors de la transition du collège au lycée au moment du passage de la programmation par blocs à la programmation Python.

Nous espérons ainsi voir ce projet de programme publié en l'état pour la partie "pensée informatique" et incitons à publier dès que possible des documents d'accompagnement intégrant des propositions d'usages de nouveaux environnements d'apprentissage de la programmation, cohérents et alignés avec les objectifs d'apprentissage du cycle, et inspirés des références didactiques proposées.

Références

- [1] Sylvie Alayrangues, Emmanuel Beffara, Sébastien Daniel, Christophe Declercq, Anne Héam, et al.. Une analyse des exercices d'algorithmique et de programmation du brevet 2017. 2018. ⟨hal-02077738v1⟩
- [2] Anne Héam. « La programmation événementielle avec Scratch : moins simple qu'il n'y paraît ». In : MathémaTICE (56 14 juin 2017). en ligne, http://revue.sesamath.net/spip.php?article998
- [3] Projet de programmes de mathématiques du cycle 4 - CSP, Education.gouv.fr, 4 mai 2025, en ligne, https://www.education.gouv.fr/media/227316/download
- [4] Sophie Chane-Lune, Sébastien Hoarau, Christophe Declercq. RCP un Référentiel de Compétences en Programmation, IREMI de La Réunion, en ligne, https://iremi974.gitlab.io/rcp/index.html
- [5] Christophe Declercq, Florence Nény. Block2Py, un éditeur de blocs pour l'apprentissage du langage Python. Didapro 8 – DidaSTIC, Feb 2020, Lille, France. ⟨hal-02526883⟩
- [6] Matthieu Branthôme. Conception et évaluation du jeu sérieux Pyrates: agencement du milieu didactique pour la transition Scratch-Python. Didapro 9 – DidaSTIC 9ème colloque francophone de didactique de l'informatique, May 2022, Le Mans, France. ⟨hal-03674705⟩
- [7] Marielle Léonard. Approche didactique et instrumentale de la pensée informatique : focus sur le concept de motif. Education. Université de Lille, 2024. Français. ⟨NNT : 2024ULILH034⟩. ⟨tel-04793189⟩
- [8] Sentance, S. and Waite, J. (2017). Primm : Exploring pedagogical approaches for teaching text-based programming in school. In Proceedings of the 12th Workshop on Primary and Secondary Com- puting Education, WiPSCE '17, page 113–114, New York, NY, USA. Association for Computing Machinery.
- [9] Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. ACM Transactions on Computing Education (TOCE), vol. 10, no. 4 (November 2010).