Bilan de 25 années d'enseignement du codage informatique dans le secondaire : recommandations et prospective.

Rupert Meurice de Dormale

fadagogo.com

rupert@fadagogo.com

RÉSUMÉ

Cet article attire l'attention de celles et ceux qui désirent concevoir des séquences d'apprentissage du codage ou choisir une de ces méthodes sur un certain nombre de passages délicats dans l'apprentissage du codage par des débutant.e.s absolu.e.s. Ces difficultés sont les sauts de lignes, la maîtrise de l'abstraction et la gestion des variables.

Les rôles de l'enseignant.e en classe sont abordés : animation du groupe, évaluation et aide aux plus faibles.

L'article suggère aussi une voie alternative pour former rapidement le personnel nécessaire à la généralisation de l'enseignement du codage.

ABSTRACT

Review of 25 years of teaching computer coding in secondary schools: recommendations and prospective.

This article draws the attention of those who wish to design coding learning sequences or choose one of these methods to a number of tricky passages in the learning of coding by absolute beginners. These difficulties are line breaks, mastering abstraction and managing variables.

The roles of the teacher in the classroom are discussed: leading the group, assessing and helping the weakest.

The article also suggests an alternative way to quickly train the staff needed to generalise the teaching of coding.

Mots-Clés : Codage, Programmation, Robotique virtuelle interactive, Algorithmique, Abstraction, fadagogo.com

Keywords : Coding, Programming, Interactive virtual robotics, Algorithms, Abstraction, fadagogo.com

Les années '80 ont vu un développement florissant de tout ce qui touche à la pédagogie de l'informatique, notamment l'apprentissage de ce qu'on appelle maintenant couramment le codage. J'ai enseigné le codage durant 25 années à plusieurs centaines d'élèves âgé.e.s de 16 à 18 ans dans l'enseignement secondaire général belge. Une méthode d'enseignement a été développée et améliorée par touches successives au fur-et-à-mesure des difficultés rencontrées chez les élèves. Cela m'a permis de passer d'un tiers d'élèves en difficulté en début de carrière à un taux très bas d'échecs. Une étape significative a été la mise au point en 1995 d'exercices de robotique virtuelle interactive. Cette approche a été un tournant dans les possibilités qu'ont eues les élèves de confronter leurs schémas mentaux avec le réel. Tout ce cheminement a abouti à une méthode d'apprentissage cohérente et efficace disponible sur l'Internet. J'y ferai référence de temps en temps au fil de cet article.

Je vous livre ici le résultat de ce que cette expérience m'a appris sous forme de recommandations et de prospective, voyez dans quelle mesure cela peut vous être utile. Comprenons bien que si je fais référence à ma méthode, il faut y voir la promotion de principes et d'un exemple plutôt que sa mise en avant comme la seule voie pertinente possible.

Les critères de choix

Comment différencier une bonne méthode d'apprentissage du codage d'une mauvaise ?

Tout simplement en examinant la façon dont les nouveaux concepts se « bousculent au portillon ». Les méthodes qui, dès la première minute, abordent un langage informatique, la manipulation de variables, de structures de contrôle... sont pédagogiquement peu recommandables.

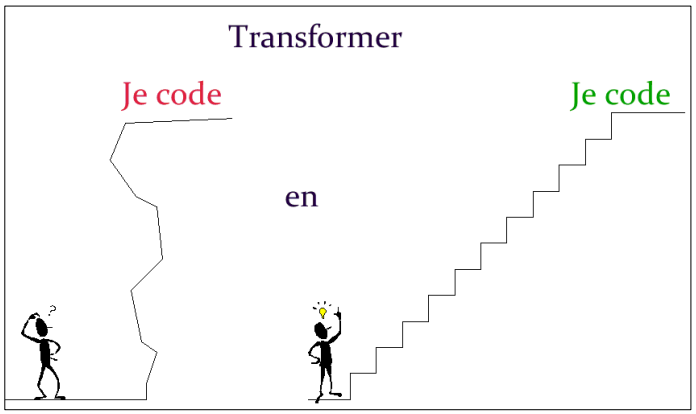

FIGURE 1 - Aborder les nouveaux concepts de façon progressive.

Une méthode d'apprentissage du codage n'est jamais facile à évaluer. Certains critères peuvent y aider :

- Le taux de réussite des élèves, mais cela demande du temps pour le mesurer ;
- La mention précise des objectifs à atteindre ;
- Le détail argumenté des méthodes pédagogiques mises en œuvre pour résoudre les difficultés des apprenants ;
- Les méthodes pour vérifier la maîtrise des acquis visés ;
- L'ergonomie, l'interactivité, les changements d'interfaces au cours de l'apprentissage…

Voyez les informations disponibles dans les méthodes que vous analysez et les critères qui vous importent le plus, sachant que toute méthode doit au moins apporter les compétences suivantes :

1. La séquence : maîtriser l'ordre dans lequel les instructions doivent se succéder ;
2. Les conditionnelles : maîtriser l'utilisation de tests conditionnels ;
3. Les expressions booléennes : pouvoir combiner plusieurs conditions entre elles ;
4. Les répétitives : maîtriser la répétition d'une ou de plusieurs instructions ;
5. Les variables : maîtriser les concepts d'affectation et de lecture des variables ;
6. Les annexes (fonctions) : la création et l'utilisation d'annexes au programme principal :
7. La mise en œuvre de tout cela en même temps ;
8. Et ceci, à notre avis, en pseudo-code. Un langage informatique pourra être introduit avec facilité quand tous ces concepts seront maîtrisés.

Différents points de cette liste vont être abordés dans les chapitres suivants afin d'attirer l'attention sur les difficultés rencontrées et les suggestions/recommandations qui peuvent être utiles.

Les sauts : se mettre « dans la peau » de l'ordinateur

Il est essentiel de déshumaniser l'ordinateur. Il ne voit rien, ne comprend rien, ne se souvient pas du passé ni ne peut anticiper l'avenir. Il ne connaît pas la finalité du programme qu'il exécute. C'est un automate : **l'ordinateur est un robot à exécuter des marches à suivre** (ou programmes).

On se focalise généralement sur ce que l'ordinateur est capable de réaliser au travers de ses différentes fonctions (lire au clavier, afficher à l'écran, calculer...), mais ce qu'il est aussi essentiel d'aborder est la façon dont l'ordinateur exécute son programme. À certains moments, il exécute les instructions ligne après ligne, à d'autres moments, il effectue des sauts en avant ou en arrière dans le programme sans qu'on comprenne toujours pourquoi ni comment.

Cette maîtrise des sauts de lignes est indispensable à tout programmeur, mais n'est pas abordée de façon explicite, systématique et rigoureuse lors de l'apprentissage, un peu comme si cela devait se manifester spontanément. Or, d'expérience, il s'avère qu'environ un tiers des élèves s'emmêlent dans ces sauts, ne sachant pas très bien d'où ils partent et surtout où ils arrivent. C'est une des trois grosses difficultés (avec l'abstraction et la gestion des variables) qui doivent attirer toute l'attention de l'enseignant.e.

Dans la méthode réalisée, c'est le premier point sur lequel l'attention est focalisée. Les sauts de lignes doivent devenir automatiques, tout comme le maniement du changement de vitesse pour un.e débutant.e en conduite automobile.

Les élèves doivent comprendre qu'il y a deux programmes dans l'ordinateur : le programme que j'ai réalisé moi-même et que je désire qu'il exécute, et le programme qui va permettre à l'ordinateur d'exécuter correctement mon programme.

L'essentiel de la tâche du programme interne de l'ordinateur est de déterminer : « **Quelle sera la prochaine ligne à exécuter ?** ».

Face à cette question, il y a trois possibilités :

1. Passer à la ligne suivante, ce sera le cas après une instruction simple et un FIN SI ;
2. Faire un saut « en avant » en passant un certain nombre de lignes ;
3. Faire un saut « en arrière » en recommençant un certain nombre de lignes…

… les points 2 et 3 étant en réponses au résultat d'un test effectué. Ce que devra être capable de faire un ordinateur est une espèce de jeu de l'oie où il fait des pas en avant ou en arrière en fonction de la case (de la ligne) sur laquelle il tombe.

En algorithmique de base, il existe **deux cas de « non-sauts »** (instructions élémentaires et FIN SI), **deux cas de sauts en avant** et **deux cas de sauts en arrière**.

Les 2 sauts « en avant » : le SI seul et SI avec un SINON

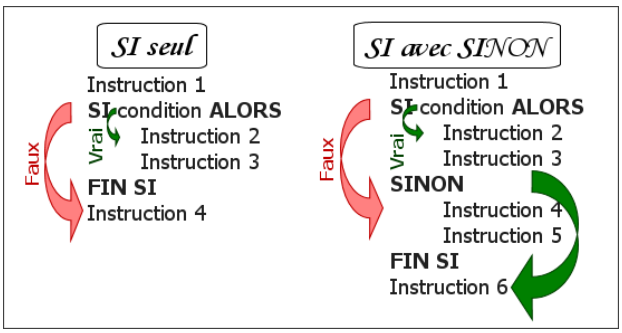

FIGURE 2 - Flux des instructions dans les conditionnelles.

Les 2 sauts « en arrière » : le RÉPÈTE et le TANT QUE

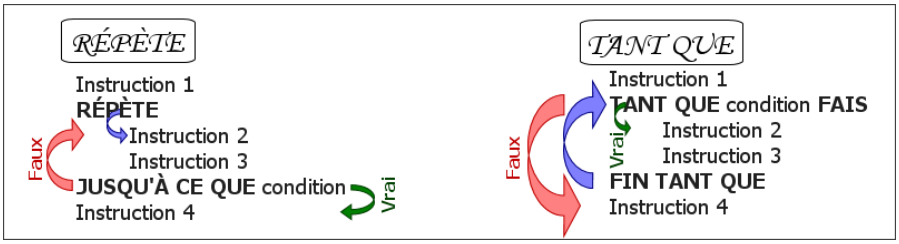

FIGURE 3 - Flux des instructions dans les répétitives.

Pour ce faire, l'ordinateur est capable de réaliser un certain nombre de tests dont le contenu n'a pas tellement d'importance à ce stade, la chose essentielle à savoir est que le résultat d'un test donne uniquement deux réponses possibles : VRAI ou FAUX.

Il est important de vérifier la bonne gestion des sauts pour chaque élève grâce à des questionnaires adaptés aux conditionnelles[1] et aux répétitives[2].

Comprendre et maîtriser une structure conditionnelle est une chose, pouvoir construire cette structure à partir d'un énoncé en est une autre. Il est possible d'aider les élèves à cela en leur présentant les « zones » d'une conditionnelle quand le résultat du test est VRAI ou FAUX.

Dans ce cas, le robot n'a que deux conditions à sa disposition : « Boule petite » et « Boule bleue ». Sur cette base, l'ordinateur doit distinguer la petite bleue, la petite jaune, la grosse bleue et la grosse jaune. Voici une manière de montrer comment construire une conditionnelles grâce aux « zones ».

FIGURE 4 – Construction d'une conditionnelle sur base des zones VRAI / FAUX.

Les opérateurs logiques

Pour les élèves en difficulté d'abstraction, une analogie des différents opérateurs logiques avec des circuits électriques peut aider à la compréhension de ces concepts[3].

La maîtrise du théorème de de Morgan est également importante, les négations de conditions complexes amenant parfois des situations difficiles à résoudre[4]. Cette loi simple aide efficacement.

Des tests[5] peuvent vérifier les acquis.

L'abstraction

Le manque d'abstraction chez les élèves est difficile à gérer car les explications que nous fournissons sont elles-mêmes… abstraites. Comme le précise cet article[6], il est indispensable de permettre à l'élève de confronter sa conception abstraite des tâches à accomplir avec le concret de leur exécution. Ceci est difficile dans un programme informatique car l'ordinateur est une boîte totalement noire.

La lecture de « L'apprentissage de l'abstraction » de Barth (1987) nous a conduit à définir sept principes sur lesquels nous pouvons nous baser pour permettre des allers-retours entre l'abstrait des schémas mentaux de l'élève et le concret de l'exécution des robots. Ceci a mené à la réalisation d'une série d'exercices de robotique virtuelle interactive en 1995.

---

1 Voir : https://fadagogo.com/algorithmique_et_programmation/html/ch03_test_sur_les_conditionnelles_01.html

2 Voir : https://fadagogo.com/algorithmique_et_programmation/html/ch05_test_sur_les_repetitives_01.html

3 Voir : https://fadagogo.com/automation/html/ch03_les_fonctions_logiques_de_base.html
et https://fadagogo.com/images_new/Expressions_booleennes.mp4

4 Voir : https://fadagogo.com/algorithmique_et_programmation/html/ch04_les_expressions_booleennes.html
et https://fadagogo.com/images_new/Theoreme_de_Morgan.mp4

5 Voir : https://fadagogo.com/algorithmique_et_programmation/html/ch04_test_sur_les_booleens_01.html

6 Voir : http://binaire.blog.lemonde.fr/2019/01/08/le-codage-est-un-langage-mais-pourquoi-et-comment/

1. Ne pas couper l'élève de ses ressources

Cet élève, même s'il ne peut abstraire facilement, connaît déjà des tas de « choses » qui sont essentiellement concrètes. Ce sont ces choses connues qui doivent servir de base à l'acquisition de nouveaux concepts. Si l'univers d'apprentissage dans lequel est placé l'élève est trop artificiel, celui-ci est coupé de ses ressources et il ne peut s'en servir.

2. Mettre l'élève devant une tâche clairement définie

La réussite et l'erreur sont complémentaires dans l'acquisition de nouveaux savoir-faire, à condition que l'univers dans lequel l'élève évolue, l'univers de référence, soit correctement défini et stable. La tâche à effectuer (ou à faire effectuer) devra donc être claire et précise. Ainsi, toute réussite et/ou échec se référera à la maîtrise du savoir-faire demandé, à l'exclusion de tout autre élément variable.

3. Permettre à l'élève d'explorer seul la tâche demandée

Il est capital que l'élève fasse, au moins dans un premier temps, ses découvertes seul. Il peut alors donner libre cours à ses hypothèses personnelles, alors qu'en groupe, l'élève se considérant comme faible risque de s'autocensurer.

4. Donner à l'élève le droit à l'erreur et à sa correction

Il est fondamental que l'élève sache qu'il peut se tromper, vérifier de lui-même, puis corriger son erreur. Ceci le décrispe, mais en plus lui permet de pousser l'exploration plus loin. Il fait ainsi appel à ses ressources selon des méthodes et modèles variés. L'exploration est plus poussée, multipliant les contre-exemples qui sont aussi importants que les réussites.

5. Permettre à l'élève de naviguer simultanément dans le concret et l'abstrait

La formation d'images mentales aide l'élève à gérer l'abstrait. Il faudra donc éduquer et stimuler son imagerie mentale en lui présentant, dans sa phase d'apprentissage, une navigation simultanée dans le concret et l'abstrait. En robotique pédagogique, « Grâce à la réalité des microrobots on donne du sens aux programmes, l'exécution d'une action sur le microrobot renvoie une image aux apprenants du programme qu'ils ont créé." (Leroux, 1996). À partir du moment où nous demandons à l'élève de réaliser lui-même son programme, il doit effectuer un grand nombre d'opérations, à savoir :

1. Imaginer les actions à réaliser par l'exécutant, quelles que soient les conditions de départ ;
2. Déterminer la structure de contrôle adéquate dont dépendra éventuellement chaque action ;
3. Formuler correctement la condition requise ;
4. Détecter les éventuelles imbrications de plusieurs structures de contrôle entre elles.

Il est donc indispensable que le début de l'apprentissage se fasse de façon rigoureuse :

- Présenter un univers clairement défini ;
- Permettre à l'élève de réaliser sa propre marche à suivre ;
- Permettre son exécution par le robot ;
- Effectuer un parallèle entre instruction exécutée et action effectuée ;
- Attirer l'attention sur les conditions testées et leurs résultats ;
- Attirer l'attention sur la ligne à laquelle va continuer le programme en fonction du résultat du test, ce point étant pour nous le plus important.

Après la réalisation de cette phase et la vérification de sa maîtrise, la phase de conception des véritables programmes informatiques en pseudo-code pourra commencer.

6. Faciliter la compréhension de son erreur par l'élève

Pour que l'erreur soit formative, elle doit être intégrée comme un contre-exemple (contraste) et donc être tout à fait comprise par l'élève. Si une structure de contrôle est mal utilisée, il doit visualiser que lorsque l'exécution de la marche à suivre arrive à ce niveau, les tâches réalisées "dérapent" et mènent à des actions indésirables.

7. Il faut une gradation correcte des difficultés

Apprendre, c'est comme monter un escalier. Il faut que toutes les marches s'y trouvent et aient une hauteur accessible. Les marches manquantes ou trop hautes bloquent l'apprentissage.

◊ ◊ ◊

Les bienfaits de la robotique pédagogique ne sont plus à démontrer. Celle-ci convient tout à fait à la réalisation de séquences d'apprentissage répondant aux prescriptions énoncées ci-dessus.

Nous avons cependant voulu opter pour des robots **virtuels**, et ceci pour plusieurs raisons :

- Moindre coût ;
- Moindre complexité de mise en œuvre ;
- Portabilité ;
- Et surtout une grande possibilité de variation des univers abordés ;

Cette diversité des univers abordables permet de concevoir des robots variés dédiés exactement aux concepts que l'on veut faire acquérir.

Par exemple, le robot loueur de voitures (qui vérifie les possessions du permis de conduire **ET** de la clé de voiture) introduit le théorème de de Morgan. Pour « forcer » l'utilisation de celui-ci, seulement deux SI sont permis et le SINON n'est pas disponible. Ceci oblige à la négation du ET.

Dans chaque exercice de robot, le système vérifie 23 points de syntaxe avant l'exécution du programme. Lors de l'exécution, le système peut repérer jusqu'à 33 erreurs fatales en signalant leur nature aussi explicitement que possible ainsi que la ligne à laquelle s'est produite l'erreur. Les programmes qui fonctionnent mais qui ne sont pas rationnels (trop d'instructions et/ou trop de tests) sont signalés comme tels à l'élève.

Les robots font l'objet d'une programmation par déplacement de blocs (à la Scratch) afin d'éviter les erreurs d'écriture des instructions et structures de contrôle qui sont rapidement démotivantes.

Nous ne saurions trop conseiller la mise en œuvre de ce genre d'environnement d'apprentissage.

Les variables

Ce domaine est sans doute celui pouvant donner le plus de schémas mentaux erronés chez les élèves. Ces erreurs de représentation sont énoncées de longue date par de nombreux auteurs.

Tenant compte de toutes ces constatations, une analogie a été recherchée afin d'introduire la notion de variable en robotique virtuelle. L'analogie qui nous a paru la plus pertinente est l'introduction de la bonne vieille ardoise (qui est à expliquer à nos virtuoses des tablettes et smartphones). Aucune valeur n'est accessible si rien n'est écrit sur l'ardoise. Inscrire une valeur nécessite qu'on efface l'ancienne. Lire une valeur revient à en prendre une copie.

Lorsqu'on passe au pseudo-code, ces ardoises sont remplacées par des planchettes sur lesquelles il est possible de coller des autocollants[7].

7 Voir : https://fadagogo.com/algorithmique_et_programmation/html/ch07_la_gestion_des_variables.html

Au moins un autocollant doit être présent pour pouvoir lire la variable. Le collage d'un nouvel autocollant masque définitivement l'autocollant précédent. Des autocollants de différents formats peuvent même suggérer les types des valeurs que les variables pourront contenir.

Cette matière nécessite quelques exercices afin de vérifier que les concepts soient bien acquis[8].

Le pseudo-code

Ce point à lui seul nécessiterait des pages d'explications[9]. Il s'agit de transformer la boîte noire qu'est l'ordinateur en une boîte vitrée dans laquelle toutes les opérations sont visibles étape par étape. Ceci permet de créer les images mentales chères au Pr Charles Duchâteau (CeFIS-UNamur).

L'avantage d'un tel système est que l'élève peut créer n'importe quel programme dans les limites du système (maximum 8 variables…) et voir son exécution pas à pas, chaque expression étant exécutée étape par étape. Un tel système facilite l'autocorrection de ses erreurs par l'élève.

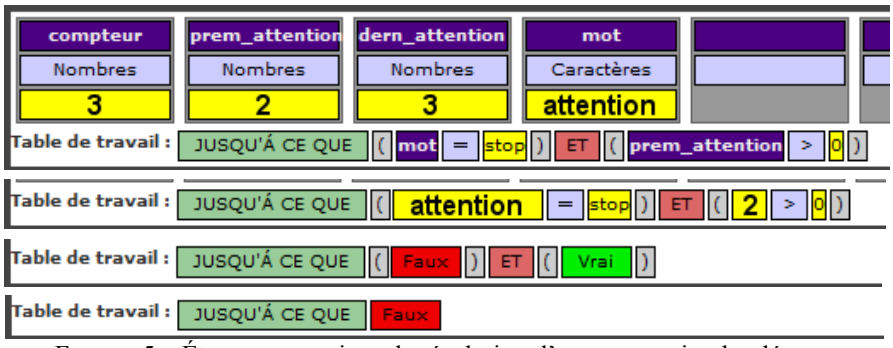

FIGURE 5 – Étapes successives de résolution d'une expression booléenne.

La dynamique des cours

L'intérêt d'une méthode en ligne est que l'enseignant.e est libéré.e du rôle de transmission du savoir. Il/elle peut alors utiliser ce temps pour gérer le groupe, jouer son rôle d'expert.e en évaluation et venir en aide aux plus faibles.

Par ailleurs, il se met rapidement en place des activités d'aide entre les élèves et la classe résonne bientôt comme une petite ruche où chacune et chacun interagit avec les autres.

Cette disponibilité en ligne permet aux plus motivé.e.s de continuer leur apprentissage en dehors de l'école ou aux élèves en difficulté de combler leur retard.

Un manuel d'accompagnement pédagogique et une fiche de suivi des élèves doivent être disponibles[10]. Ils permettent à l'enseignant.e d'avoir des conseils précieux sur l'aide à apporter dans tel ou tel cas et de suivre l'évolution de chacune et chacun en notant la date de réalisation de chaque exercice. Ceci permet de repérer rapidement les élèves à la traîne et de leur venir en aide.

Un tableau de bord[11] permet à chaque élève, grâce à un système de vignettes, de savoir à tout moment où il en est dans sa tâche. Un système de tutorat[12] permet à l'enseignant.e de consulter les programmes des élèves en différé mais sans toutefois pouvoir les modifier de façon durable. Ce système permet par exemple de projeter en classe un programme erroné de façon anonyme et de demander aux élèves de relever la ou les erreurs et de suggérer les corrections nécessaires.

8 Voir : https://fadagogo.com/algorithmique_et_programmation/html/ch07_test_sur_les_variables_01.html

9 Voir : https://fadagogo.com/algorithmique_et_programmation/html/ch08_programmer_en_pseudocode.html

10 Voir : https://fadagogo.com/pedagogie_codage/accompagnement_pedagogique.html

11 Voir : https://fadagogo.com/algorithmique_et_programmation/html/_informations_defis.html

12 Voir : https://fadagogo.com/algorithmique_et_programmation/html/_informations_tutorat.html

Cette activité est importante et développe les capacités de réflexion des élèves.

Prospectives[13]

La formation traditionnelle des futurs enseignant.e.s du codage en trois ou cinq ans envisagée dans différents pays n'est pas de nature à combler rapidement les postes nécessaires. À ce rythme, on n'ose imaginer en quelle année le cadre sera complet, sans tenir compte du problème de l'attractivité. Un jeune capable d'études en informatique va-t-il ou elle se destiner à l'enseignement alors que le privé recrute à grande échelle avec des salaires bien plus attractifs ?

Or il faut des enseignant.e.s **maintenant**, tout de suite. L'idée est de mettre en accès libre une méthode d'apprentissage du codage en ligne et de suggérer aux enseignant.e.s du cadre de s'essayer à cette discipline. Les plus intéressé.e.s pourraient alors rejoindre une formation qui leur permettrait d'acquérir les qualités d'enseignement du codage ou, à tout le moins, d'algorithmique.

Linguistes ou matheux ? À qui confier l'enseignement du codage ?

Il est généralement considéré que le codage est un domaine réservé aux matheux, ce qui est totalement faux. Il me semble néfaste de baser l'apprentissage du codage sur des exercices purement mathématiques. Cela fermerait cette discipline à un public que les maths enjouent peu. Le codage est essentiellement un problème de finesse du langage, de logique… Il doit rester ouvert à une grande diversité de domaines, quels que soient les enseignant.e.s qui s'en occupent.

Conclusions

Parmi les différentes activités informatiques, le codage est sans doute celle qui est la plus difficile à acquérir car elle nécessite la maîtrise de nombreux concepts. Les trois difficultés majeures sont la maîtrise des sauts de ligne, l'abstraction et la gestion des variables. L'article montre comment ces difficultés pourraient être, au moins en partie, aplanies dans l'intérêt de toutes et tous.

L'utilisation d'une méthode d'apprentissage en ligne libère l'enseignant.e de la tâche de transmission, lui permettant de jouer à plein ses fonctions d'animation du groupe et d'évaluation.

La formation des futur.e.s enseignant.e.s doit être repensée. Une décentralisation de la formation permettrait à des enseignant.e.s du cadre de combler rapidement les postes indispensables à la généralisation de l'apprentissage du codage ou, à tout le moins, de l'algorithmique.

Bibliographie

BARTH, B-M. (1987). L'apprentissage de l'abstraction, Paris, Retz,

CERVERA, D. et NONNON, P. (1993). Démarche de modélisation en simulation assistée par ordinateur pour l'apprentissage des concepts d'énergie des fluides, in DENIS, B. et BARON, G.-L. éd., Regards sur la robotique pédagogique, Actes du quatrième colloque international sur la robotique pédagogique, Université de Liège, Liège (B), pp. 187-195

DUCHÂTEAU, Ch. (1990). Images pour programmer, Louvain-la-Neuve (B), De Boeck-Wesmael

DUCHÂTEAU, Ch. (1993). Robotique-Informatique mêmes ébats, mêmes débats, mêmes combats ? in DENIS, B. et BARON, G.-L. Éd., Regards sur la robotique pédagogique, Actes du quatrième colloque international sur la robotique pédagogique, Liège (B), Université de Liège, pp. l0-33

HUDON, R. et NONNON, P. (1993). Environnement pédagogique informatisé pour la "visualisation" de systèmes technico-scientifiques, in DENIS, B. et BARON, G.-L. éd., Regards sur la robotique pédagogique, Actes du quatrième colloque international sur la robotique pédagogique, Liège (B), Université de Liège" pp.173-178

LEROUX, P. (1996). Intégration du pilotage de microrobots pédagogiques à un environnement de programmation, in INBMI éd., Les actes de la 5éme rencontre francophone sur la didactique de l'informatique, Tunis, INBMI, pp. 183-194 (Disponible via l'EPI (F), le SSIE (CH) ou le CeFIS (B))

13 Voir : https://fadagogo.com/pedagogie_codage/avis_aux_ministres.html