Accueil

La programmation et le développement de la pensée informatique

Par **Collaboration spéciale** - 19 janvier 2018

À l'automne 2017 a été dévoilée la stratégie numérique du Québec. Un des objectifs visés est d'assurer le développement des compétences numériques, entre autres par l'apprentissage de la programmation. Ce dossier se veut un outil pour faire connaissance avec le développement de la pensée informatique.

Un dossier conjoint de Carrefour éducation et d'École branchée

***par Adam Desjardins**, enseignant d'informatique,*
***Amy Tran**, enseignante d'informatique et de mathématique,*
***Marc-André Girard**, directeur du secondaire,*
Collège Beaubois

Sommaire du dossier

Introduction

Il en est question dans tous les milieux scolaires. On dit parfois programmation, codage ou robotique. On en fait même des compétitions! Ce sont là des composantes d'une compétence du 21e siècle : le développement de la pensée informatique.

Ce dossier, contrairement à ce qu'on pourrait penser, ne s'adresse pas qu'aux enseignants *geeks*, mais bien à tous ceux qui ont envie d'explorer ce à quoi le monde de l'éducation pourrait bien ressembler dans un avenir pas si lointain!

Codage, programmation et robotique

Codage et programmation

En éducation, ces deux termes ont tendance à être utilisés de manière équivalente, ce qui n'est traditionnellement pas le cas dans le domaine de l'informatique.

La **programmation** fait souvent référence au fait d'**analyser des problèmes et de les résoudre à l'aide d'algorithmes, puis de traduire ceux-ci dans l'un des nombreux langages de programmation existants**. Ceux qui occupent cette fonction sur le plan professionnel sont des *analystes-programmeurs*. La fonction de *programmeur* (sans la partie *analyse*) existait lors de l'arrivée des premiers ordinateurs dans les années 1950 et a presque disparu aujourd'hui. À noter que, de façon générale, les algorithmes sont des méthodes de résolution de problèmes à l'aide de différentes étapes de traitement d'un certain nombre de données.

De manière générale, lorsqu'on parle de **codage**, on fait référence au fait de **traduire en langage de programmation un algorithme existant**. Bref, la tâche de codage est plus une opération de traduction que de raisonnement en soi. Dans ce dossier et de façon générale en éducation, le terme programmation est privilégié

Que l'on parle de coder ou de programmer en milieu scolaire, l'important sur le plan pédagogique est ceci : **plus la tâche implique une grande portion d'analyse, plus on vise l'utilisation de la métacognition et le développement d'aptitudes en résolution de problèmes**. Et c'est justement là que se trouve la pertinence de la programmation en contexte scolaire! Nous y reviendrons.

Robotique

La robotique est l'une des façons d'appliquer des notions de programmation. Dans un contexte pédagogique, elle permet de concrétiser certaines notions qui seraient souvent beaucoup plus abstraites si elles étaient enseignées de manière formelle ou traditionnelle. De plus, certaines applications robotiques utilisées en milieu scolaire ne nécessitent peu ou pas de notions approfondies en programmation.

Enseigner les concepts de programmation grâce à la robotique constitue un bon point de départ et prépare les élèves à passer à un niveau de programmation plus formel.

Un aspect intéressant de la robotique est la variété de périphériques d'entrées (capteur de lumière, détecteur de distance, sonde à température, etc.) et de sorties (moteur, lumière, son, etc.) possibles. Il y a également un élément motivationnel pour l'élève dans le fait de voir directement et concrètement les résultats de sa programmation par l'entremise des actions d'un robot.

Historique

Vouloir rendre accessible l'apprentissage de la programmation est une idée qui ne date pas d'hier. Par exemple, en 1964, le langage BASIC a été inventé pour initier les étudiants universitaires et a longtemps été utilisé pour montrer les bases de la programmation.

L'idée d'enseigner formellement cette matière n'est pas nouvelle non plus. On n'a qu'à penser au langage Logo. Les premières versions de ce langage ont été conçues vers la fin des années 1960 et des expériences en contexte scolaire ont été menées dans les années 1970 dans quelques écoles des États-Unis. C'est toutefois dans les années 1980 que les tentatives d'implantation à plus grande échelle ont eu lieu, particulièrement en cinquième et sixième année du primaire, y compris au Québec.

Au Québec, il n'y a pas eu de déploiement à grande échelle d'un enseignement formel de la programmation, outre le programme de formation ISI (Introduction à la science de l'informatique) créé par le Ministère de l'Éducation du Québec en 1982. Ces cours optionnels étaient proposés aux élèves de quatrième et cinquième secondaire, mais ont pratiquement disparu des écoles. D'ailleurs, sans être la raison officielle de cette disparition, le coût des ordinateurs a longtemps constitué un frein à l'implantation des cours de programmation dans les écoles québécoises.

Pourquoi en parle-t-on maintenant?

Depuis plusieurs années, la croissance des entreprises dans le domaine des technologies de l'information et de la communication (TIC) crée une demande pour des travailleurs spécialisés dans tous les secteurs de l'informatique, particulièrement dans le développement de logiciels, de jeux vidéo et de solutions de communication pour les particuliers ou les entreprises. La pénurie de main-d'œuvre prévue par plusieurs analystes économiques dans le domaine des sciences, technologie, ingénierie et mathématiques (qu'on regroupe sous l'acronyme STIM, ou STEM en anglais) crée une pression sur les gouvernements et sur les systèmes d'éducation. En 2012, Microsoft a d'ailleurs suggéré différentes avenues pour renforcer la formation dans les domaines STIM, particulièrement dans celui de l'informatique. On peut en savoir plus en lisant *A National Talent Strategy: Ideas for Securing U.S. Competitiveness and Economic Growth*.

Que cette pénurie anticipée devienne réalité ou non, l'idée d'ajouter la programmation à la formation primaire et secondaire a déjà un impact. Par exemple, de grandes entreprises comme Google, Microsoft et Facebook financent

l'organisme à but non lucratif *Code.org*, dont l'un des principaux objectifs est de promouvoir l'apprentissage de la programmation dans les écoles.

Nonobstant le désir des grandes entreprises de voir leurs besoins de main-d'œuvre comblés, il faut reconnaître que la capacité de résoudre des problèmes et d'exprimer des solutions dans un langage informatique prend de plus en plus de place dans l'éventail des compétences les plus utiles sur le marché du travail.

À plus large échelle, c'est une prise de conscience collective qui est en train de s'opérer : il faut développer de nouvelles compétences chez les élèves. Par exemple, il devient de plus en plus important que les jeunes développent une compréhension des outils qu'ils utilisent quotidiennement pour en comprendre le fonctionnement, en maximiser l'utilisation et développer un regard critique face à cette même utilisation. En quelque sorte, il s'agit d'un niveau élevé de littératie numérique qui s'opère dans le développement de la pensée informatique.

Motivation, interdisciplinarité et littératie numérique

Les nouveaux besoins du marché de l'emploi coïncident avec d'autres facteurs humains qui constituent une opportunité pédagogique sans précédent. L'attrait des jeunes (et moins jeunes) pour les jeux vidéo en constitue un exemple frappant. En effet, la **conception de jeux par la programmation est un tremplin intéressant pour les individus, autant sur le plan motivationnel que pédagogique**. Ce sont des apprentissages qui détonnent avec ce que les élèves sont habitués à faire en classe. De plus, la possibilité de concrétiser certains contenus théoriques par la programmation plaît particulièrement aux garçons. Ces situations d'apprentissage se prêtent particulièrement bien à l'interdisciplinarité et à la transversalité. À lui seul, **l'apprentissage d'un langage de programmation permet le développement d'une série de compétences transposables dans plusieurs domaines**. Par exemple, les tâches informatiques s'intègrent naturellement à celles abordées dans les cours de mathématiques et de sciences. Nombre d'enseignants créatifs extrapolent même ces arrimages en français, en histoire et dans d'autres matières.

L'omniprésence du numérique dans le quotidien ne constitue-t-elle pas à elle seule une raison valable de se questionner sur ce qui se trouve derrière l'écran pour en comprendre le fonctionnement, et ainsi d'évacuer la conception magique que nous sommes portés à octroyer à ces machines? Justement, les cours de programmation pourront diminuer les effets d'un potentiel « fétichisme technologique » (attrait pour le « gadget ») et permettre de garder une distance critique face aux outils utilisés quotidiennement.

La progression des apprentissages en programmation

Avant même de choisir un outil pour enseigner la programmation, il convient de bien définir jusqu'où on désire accompagner les élèves dans leur apprentissage. Définir les différentes phases d'apprentissage de la programmation est une tâche colossale qui n'est pas encore achevée en ce qui concerne l'enseignement primaire et secondaire, la recherche étant encore à ses débuts dans ce domaine. Malgré cela, il est possible de définir quelques éléments d'apprentissage en consultant ce qui s'est déjà fait par le passé et ce qui semble fonctionner aujourd'hui.

Actuellement, les plateformes les plus populaires destinées à l'apprentissage de la programmation utilisent toutes plus ou moins les mêmes façons de faire : programmation séquentielle, procédurale et événementielle.

Programmation séquentielle	Programmation procédurale	Programmation événementielle
Grosso modo, la programmation séquentielle **implique la définition d'une « recette » que l'ordinateur doit suivre dans un ordre prédéfini**. Pour les	Dans la programmation procédurale, **on crée plusieurs « recettes », appelées routines ou procédures**. Cette façon de	En programmation événementielle, **les procédures peuvent être exécutées suivant certaines actions de l'utilisateur**.

élèves plus jeunes, c'est un bon point de départ parce qu'à ce niveau, les programmes sont essentiellement des recettes.	travailler étant très modulaire, elle permet de décomposer un problème complexe en plusieurs sous-problèmes plus simples.	Un clic de souris sur un bouton est un exemple d'événement qui peut déclencher l'exécution d'une procédure.

Il existe beaucoup d'autres paradigmes de programmation, comme la **programmation orientée objet** qui demande une plus grande capacité d'abstraction. Ici, l'objet représente un concept, une idée ou toute entité du monde physique, comme une voiture, une personne ou encore une page d'un livre. Il possède une structure interne et un comportement, et il sait interagir avec ses pairs. Il s'agit donc de représenter ces objets et leurs relations ; l'interaction entre les objets via leurs relations permet de concevoir et réaliser les fonctionnalités attendues, de mieux résoudre le ou les problèmes. Il est possible, avec certains élèves de la fin du secondaire, d'aborder ce style de programmation. De plus, il est même souhaitable que les élèves soient exposés bien avant à certains concepts inhérents à cette façon de programmer.

Afin de montrer le plus simplement possible quelle pourrait être la progression des apprentissages au niveau des structures de programmation, voici une série d'exemples en pseudocode illustrant les premières phases d'apprentissages :

Procédure séquentielle simple

C'est souvent par cette étape que l'on aborde la programmation. Un exercice typique consisterait à faire bouger un personnage dans un labyrinthe, ou encore faire un dessin. Un programme dessinant un triangle équilatéral pourrait ressembler à ceci :

```
Stylo en mode écriture
Avance de 10 pas
Tourne à 120 ° à gauche
Avance de 10 pas
Tourne à 120 ° à gauche
Avance de 10 pas
Relève le stylo
```

Structure répétitive simple

À cette étape, on pourrait imaginer que l'élève doive écrire un programme qui fait bouger une balle vers la droite en déplaçant celle-ci de 100 pixels (le pixel est une unité de mesure de la taille d'une image) au total. Le pseudocode pourrait ressembler à ceci :

```
Répète 100 fois {
   Avance la balle d'un pixel vers la droite
}
```

Lorsqu'on ajoute la structure de programmation répétitive, qui permet d'exécuter une ou plusieurs lignes de code un nombre déterminé de fois, on ne l'utilise effectivement que pour cette tâche. Toutefois, son intérêt se révélera beaucoup plus loin dans le processus d'apprentissage, par exemple, dans les traitements impliquant des variables indicées (comme x1, x2, x3… xn) et tout ce qui s'y apparente, notamment les chaînes de caractères et les tableaux.

Prise de décision

Ce nouvel élément permet à l'ordinateur de prendre une décision en fonction d'une information qui est variable. Par exemple, le fait de choisir au hasard la couleur des pièces que devra utiliser un joueur d'échecs :

Affecte une valeur aléatoire entre 1 et 2 à la variable x
Si x vaut 1 alors {
  le joueur a les pièces blanches
} sinon {
  le joueur a les pièces noires
}

Cette phase d'apprentissage comporte plusieurs cas de prise de décision : simple, binaire ou multiple. En voici différents exemples :

Cas simple : Si x alors a.
Cas binaire : Si x alors a, sinon b.
Cas multiple : Si x alors a, sinon si y alors b, sinon si z alors c, etc.

Pixabay

Structure répétitive avec condition

Cette structure permet de répéter des instructions tant qu'une certaine condition est remplie. Par exemple, un programme qui fait bouger un cheval vers la droite jusqu'à la ligne d'arrivée :

Tant que le cheval se situe avant la ligne d'arrivée {
  Avance le cheval vers la droite d'une unité
}

Afin de pouvoir résoudre des problèmes de différentes natures, l'élève doit pouvoir combiner ces structures de programmation. Cela nous amène à définir les stratégies pédagogiques permettant de développer de véritables compétences en programmation.

Quelques stratégies pédagogiques pour aborder la programmation en classe

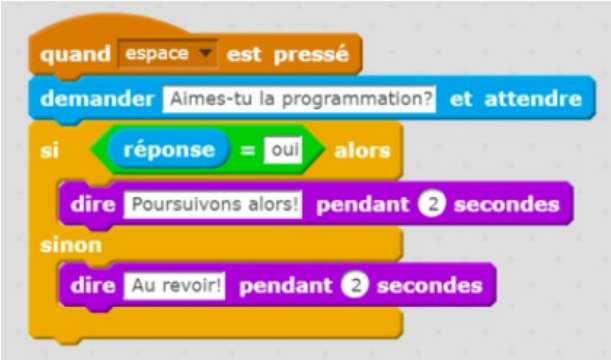
De plus en plus d'enseignants désirant intégrer la programmation dans leurs approches pédagogiques se butent au fait qu'il existe peu d'ouvrages didactiques à cette fin. Ainsi, voici quelques façons d'aborder la programmation, et ce, sans mettre en péril sa gestion de classe!

Le modelage

Dès le 1er cycle du primaire

Exemple d'exercice

Voici une commande qui permet au lutin de poser une question :



Maintenant, à votre tour de créer un questionnaire!

L'enseignant peut modeler l'activité en présentant un exemple de programmation et en invitant les élèves à faire de même, avec peu ou pas de divergence face au modèle imposé. Sur le plan cognitif, **les élèves sont en contact avec divers éléments liés à la logique, en développant des compétences de structuration des démarches et, finalement, en étant sensibilisés à l'importance d'une démarche rigoureuse**.

Cette méthode d'enseignement s'apparente à une liste d'instructions d'une recette : les élèves n'ont qu'à les suivre attentivement pour accomplir la tâche. À notre avis, cette approche est plutôt adaptée pour les jeunes du 1er cycle du primaire. C'est d'ailleurs ce que l'on retrouve le plus fréquemment dans nos écoles actuellement.

L'ordonnancement

Dès la fin du primaire

Exemple d'exercice

Voici 8 blocs de commandes. Créez un programme à l'aide de ces 8 blocs SEULEMENT. Tous les blocs doivent être utilisés et vous pouvez utiliser un bloc plus d'une fois. Laissez aller votre créativité!

Avec cette deuxième façon de faire, l'élève doit choisir une série d'instructions permettant d'accomplir une tâche parmi celles fournies au préalable par l'enseignant. Ce dernier présente la fonction d'une série de commandes et propose aux élèves de les mettre dans un ordre adéquat. Sur le plan cognitif, cette méthode d'enseignement

mobilise la créativité du jeune et permet de développer la prédiction et l'anticipation d'événements. C'est, en quelque sorte, la porte d'entrée du développement des compétences informatiques, soit la **capacité de penser de façon séquentielle pour parvenir à un résultat prévisible**. Aussi, selon notre expérience, cette façon de faire est plutôt motivante pour les élèves étant donné qu'ils peuvent aboutir à une multitude de résultats différents, ce qui leur confère un certain pouvoir dans leur propre démarche d'apprentissage.

À notre avis, cette approche est particulièrement pertinente pour les jeunes de la fin du primaire. Ces derniers sont alors plus autonomes et, en cas de blocage, ils ne craignent pas d'employer une démarche d'essai-erreur. De plus, étant donné que les programmes écrits sont uniques à chaque élève, une discussion sur la propriété intellectuelle est facilement envisageable suite aux activités pédagogiques, sans compter que des activités de débat ou de critique constructive peuvent également être tenues. Ce qui importe ici, c'est de permettre à l'élève de justifier ses choix et de les expliquer.

L'autorégulation

Au 1er cycle du secondaire

Parlant d'esprit critique, voici une approche un peu plus avancée où les élèves sont amenés à approfondir leurs connaissances des différentes commandes en programmation en analysant un programme déjà écrit et en y repérant des aspects-clés.

Comme mentionné précédemment, la tâche de codage se compare aisément à une tâche de traduction. D'une part, l'enseignant peut présenter un programme aux élèves et leur poser des questions précises quant aux différents résultats à prévoir. Cet exercice de verbalisation de l'abstraction permet en outre, d'une façon bien intéressante, de **développer les compétences analytiques des jeunes en décortiquant les blocs de programmation et en les traduisant dans une pensée plus formelle, prête à être réinvestie dans l'action**.

Exemple d'exercice
Voici un programme qui manipule la valeur de la variable *surprise*. Après exécution, quelle est la valeur de *surprise*? Expliquez le processus qui aboutit à ce résultat.

D'autre part, l'enseignant ou le tuteur peut présenter un programme comportant des erreurs techniques aux élèves et leur demander d'y remédier. Cet exercice est plus communément appelé une activité de *débogage*. Cognitivement, cette approche favorise le développement des compétences autorégulatrices, lesquelles s'apparentent à un raisonnement métacognitif. Au lieu de voir les erreurs comme étant des catastrophes, ce type d'exercice invite les élèves à apprendre et à tirer profit des erreurs pour les transformer en réussites. De plus, le

développement de nombreuses autres compétences est favorisé, comme la capacité de déconstruire un problème pour en identifier toutes les composantes et ainsi bâtir une solution originale et novatrice.

Cette approche peut être entamée chez les élèves du 1er cycle au secondaire et, fait intéressant, elle permet le développement de plusieurs compétences transposables dans les autres disciplines, notamment les mathématiques, les sciences et l'éthique.

Voler de ses propres ailes!

Finalement, l'enseignant propose aux élèves d'écrire leur propre programme pour accomplir une tâche. L'enseignant doit adopter un rôle d'observateur et de guide en invitant l'élève à réfléchir par lui-même à sa propre démarche de programmation : quels ont été mes erreurs et les obstacles rencontrés? Comment m'y suis-je pris pour les outrepasser?

Le rôle de l'enseignant consiste également à créer une communauté de partage des connaissances et des apprentissages au sein de la classe pour susciter la collaboration chez les élèves. Comment peuvent-ils s'aider mutuellement à surpasser les obstacles? Des conseils à offrir? Des expertises à partager?

Aussi, afin d'expliciter les processus métacognitifs de l'élève face à la tâche, il est recommandé d'adopter une méthodologie rigoureuse, telle que la tenue d'un journal de bord. Cet outil peut recueillir les traces de recherche, les plans de conception, un bilan de la progression à chaque cycle, une évaluation des pairs, des autoévaluations, etc.

Cette façon d'enseigner ne mise pas sur l'apprentissage des blocs de code spécifiquement, mais plutôt sur le développement d'une compétence issue de la pensée informatique dans une pleine autonomie.

La pensée informatique

Il se peut que vous ayez entendu parler de la *pensée informatique* (parfois appelée *pensée computationnelle*), ou bien de sa forme anglaise *computational thinking*. Peu la connaissent, mais plusieurs la pratiquent sans s'en rendre compte. En fait, en 2006, Jeannette Wing a inspiré un grand nombre d'éducateurs en déclarant ceci :

> *LA PENSÉE INFORMATIQUE EST UN ENSEMBLE D'ATTITUDES ET D'ACQUIS UNIVERSELLEMENT APPLICABLES QUE TOUS, ET PAS SEULEMENT LES INFORMATICIENS, DEVRAIENT APPRENDRE ET MAÎTRISER.*

En effet, la pensée informatique s'apparente à une compétence ou une *façon de faire* du 21e siècle. Le monde de l'informatique est souvent perçu comme un domaine abstrait et complexe. Par exemple, les discussions à propos d'intelligence artificielle et de voyages vers Mars exposent des réalités de plus en plus abstraites et difficiles à saisir. De plus en plus de problématiques nécessitent que l'être humain fasse appel à une machine afin de les résoudre. La pensée informatique, c'est donc « **une forme de raisonnement permettant de résoudre des problèmes complexes » en analysant une situation, en la décortiquant et en l'abordant de façon séquentielle**. C'est une manière d'aborder un problème étape par étape et de prévoir les incidences de chacun des gestes qui sera posé.

La pensée informatique n'est donc pas l'apanage de la science informatique et, bien au contraire, elle constitue un atout dans toutes les disciplines scolaires et dans toutes les sphères de la vie quotidienne.

Décomposer, structurer et reformuler

La pensée informatique contribue énormément au développement des compétences en résolution de problèmes. Quelle est la tâche à accomplir? Est-elle difficile? Si oui, à quel moment précisément devient-elle complexe? Évoluer dans un monde informatique favorise les habiletés à décomposer un problème, en restructurer les tâches et en reformuler les objectifs.

Les ordinateurs ne sont pas capables de penser par eux-mêmes, du moins, pas encore. Ainsi, toutes les choses merveilleuses qu'ils accomplissent viennent des commandes précises rigoureusement formulées par l'humain.

Autoévaluation – autorégulation

Même les programmeurs les plus reconnus sont en accord avec le propos suivant : *If your code works on your first trial, something is wrong*. (Traduction libre : Si ton code fonctionne du premier coup, c'est qu'il y a quelque chose qui ne va pas.) Il est tellement rare qu'un programme fonctionne du premier coup que les informaticiens sont particulièrement sceptiques quand cela arrive. Plusieurs attribuent cela au fait que les langages de programmation requièrent une très grande précision. En effet, les erreurs informatiques sont plutôt évidentes! En comparaison, dans une production écrite, l'oubli d'une lettre n'empêche pas la lecture du texte. Par contre, en programmation, l'oubli d'un seul symbole peut très bien faire en sorte que le programme ne fonctionne pas du tout. Les mises à l'essai sont fréquentes et l'aspect autorégulateur est incontournable. Les erreurs ne sont pas perçues négativement. Au contraire, elles sont considérées comme des itérations à l'établissement d'une solution optimale. L'apprentissage de la programmation requiert du temps, de la patience, et même, de la persévérance.

Environnements de développement et langages de programmation

En ce qui concerne le choix d'un langage de programmation aux fins d'enseignement, il faut distinguer plusieurs choses.

Langages de programmation vs environnements de développement

Le **langage de programmation** fait référence à la f**açon de coder : la syntaxe et le paradigme de programmation supportés**. Quant à lui, l'**environnement de développement** fait référence au **cadre logiciel dans lequel on utilisera un langage de programmation**.

Pixabay

Par exemple, l'environnement de développement Visual Studio permet de programmer dans plusieurs langages tels que C++, Python, C#, Visual Basic, etc. Par contre, on ne peut pas dire qu'on a programmé quelque chose en Visual Studio, mais on pourrait dire qu'on a programmé quelque chose dans Visual Studio en utilisant le langage C++. Dans le même ordre d'idées, on peut dire qu'on a programmé quelque chose avec Scratch, mais dans ce cas, on fait référence à la fois à l'environnement de développement et au langage de programmation.

Langages éducatifs vs « vrais » langages

Aussi, il faut distinguer les langages créés dans un but purement pédagogique de ceux qui permettent la création de véritables applications pouvant être distribuées. Par exemple, Scratch, Blockly et beaucoup d'autres langages visuels ont été développés spécifiquement pour l'apprentissage des bases de la programmation. Ils conviennent parfaitement aux élèves du primaire et du début secondaire. Par contre, plus les élèves développent des compétences dans ce domaine, plus ils voudront exploiter de véritables applications nécessitant des structures de programmation que n'offrent pas ces langages visuels.

2 facteurs à considérer dans le choix d'un langage

La syntaxe

La première chose à considérer dans le choix du langage de programmation est la syntaxe. Voici deux exemples de code où l'on additionne tous les nombres entre 1 et 10 pour affecter le résultat à une variable x :

for i=1 to 10 x=x+i	for (i=1; i<=10; i++) { x+=i;

next i	}

Le code de gauche est en langage BASIC alors que celui de droite est en C++.

Sans connaître la programmation dans ces deux langages, on peut constater que le code de gauche possède moins d'éléments syntaxiques que celui de droite et se rapproche un peu plus d'une langue naturelle : « pour i compte de 1 à 10 et répète : x est augmenté de i ». Le code écrit dans les langages visuels comme Scratch est beaucoup plus simple. La transition vers des langages formels sera plus facile si l'on tient compte de cette simplicité syntaxique.

L'environnement de développement

La deuxième chose à considérer est l'environnement de développement. Le fait de pouvoir créer une interface en déplaçant des objets sur une fenêtre accélère la programmation et permet d'obtenir des résultats plus rapidement, ce qui n'est pas négligeable sur le plan motivationnel. Il existe plusieurs solutions sur le marché, dont certaines sont gratuites lorsqu'elles sont utilisées dans un contexte éducatif. XOJO (www.xojo.com) en est un exemple intéressant. Il offre un environnement graphique et permet de coder dans un langage orienté objet dérivé du BASIC.

Peu importe la solution que l'on retient pour enseigner la programmation de manière formelle, il convient de minimiser le temps consacré à l'apprentissage de la syntaxe du langage et, conséquemment, minimiser le temps que les élèves auront à consacrer au débogage pour des erreurs syntaxiques. Tout cela afin de consacrer un maximum de temps à la logique de programmation qui permet de développer la pensée informatique.

Le petit guide de la robotique pédagogique

Il existe des dizaines d'outils pour enseigner la programmation dans un cadre robotique. En voici quelques exemples, présentés dans un contexte de progression des apprentissages.

Bee-bot et Blue-bot

Bee-bot est un robot qui peut se déplacer sur une surface. Pour le programmer, il suffit d'appuyer sur des flèches sous forme de boutons. *Blue-bot* offre les mêmes fonctions de base que *Bee-bot*, mais il est programmable en utilisant une application disponible sur iOS ou Android.

Les enfants d'âge préscolaire et du début du primaire pourront utiliser cet outil pour débuter l'apprentissage de la programmation séquentielle en faisant suivre au robot des chemins sur un tapis de jeu, ou pour trouver un chemin à travers des obstacles. Des ensembles contenant des activités et plusieurs robots sont disponibles. Il s'agit d'un bel outil pour introduire la pensée informatique.

Le prix de détail d'un *Bee-bot* est d'environ 140 $, alors qu'on devra débourser approximativement 190 $ pour le *Blue-bot*.

Cubelets

Ce sont des blocs cubiques magnétiques qu'on peut assembler pour construire une structure robotique. Chaque cube possède une fonction unique, comme un détecteur de lumière ou de distance, un moteur, un haut-parleur, une lumière, etc. Lorsqu'on les assemble, la structure peut accomplir une tâche déterminée par la somme des fonctions des blocs choisis. Il n'est pas nécessaire de les programmer : l'assemblage dicte le comportement de la structure et

s'apparente ainsi à un robot. Il est également possible de programmer les *Cubelets* en utilisant une application disponible sur iOS ou Android.

Les enfants du préscolaire et du primaire pourront apprendre des principes de robotique, mais également être exposés à une certaine forme de logique formelle, puisque des cubes permettent d'effectuer des inversions et des opérations minimum ou maximum. Ces dernières équivalentes aux trois opérateurs de base en logique : le « et », le « ou » et la négation. De plus, les Cubelets sont adaptés à cette clientèle et sont robustes.

On devra investir environ 400 $ pour un ensemble de 12 *Cubelets*, convenant à une équipe de deux élèves. D'autres ensembles sont disponibles, allant jusqu'à près de 6 000 $ pour la trousse de l'enseignant et 162 Cubelets, avec tous les accessoires nécessaires (chargeurs, câbles et adaptateurs *LEGO*).

Pixabay

LEGO Mindstorms EV3

LEGO Mindstorms EV3 est un ensemble d'éléments robotiques programmables offrant une immense variété de designs possibles. Il nécessite l'usage d'une application pour tablette Android ou iOS ou d'un logiciel pour ordinateur (Mac ou PC). La programmation est très visuelle, sous forme de blocs interconnectés en séquence.

LEGO a eu la brillante idée d'intégrer la robotique à ses légendaires blocs, ce qui permet d'initier les enfants à la pensée informatique tout en développant leur motricité fine.

L'âge minimum recommandé par *LEGO* pour cet ensemble est 10 ans, soit dès la quatrième ou cinquième année du primaire, notamment à cause des petites pièces. Mindstorms demeure une solution intéressante pour plusieurs années par la suite, y compris au début du secondaire. De nombreux concours de robotique utilisent *LEGO* Mindstorms et sont de plus en plus populaires au Québec comme ailleurs dans le monde. Un ensemble de base se détaille environ 430 $.

mBot

L'entreprise *Makeblock* offre plusieurs types de robots, dont certains portant le nom de mBot. L'aspect le plus intéressant de ce robot basé sur la technologie Arduino est qu'on peut le programmer en utilisant une version bonifiée de Scratch, un langage de programmation qui peut être enseigné dès le primaire.

Le mBot peut se déplacer et suivre un tracé sur une surface. Il possède également un détecteur de distance et peut émettre des sons et allumer des lumières DEL colorées. Le modèle *mBot Ranger* est tout à fait approprié pour les élèves du secondaire.

Le prix de cette gamme de robots, environ 130 $ pour le modèle le plus abordable, est une autre force de ce produit. Cela en fait un robot accessible autant pour les écoles que les familles.

Arduino

Bien qu'il en existe plusieurs variantes, il s'agit essentiellement d'un microcontrôleur électronique permettant d'analyser ou d'émettre des signaux électriques. On peut l'utiliser pour lire l'information provenant de capteurs, commander des moteurs, etc. On doit le programmer avec le langage C++, ce qui, pour l'enseignant, nécessite certaines connaissances de base en programmation.

Il est utilisé dans plusieurs programmes techniques du collégial, mais également avec les élèves du secondaire, particulièrement ceux de quatrième et cinquième secondaire. C'est un excellent outil pour enseigner la programmation de manière formelle tout en introduisant des éléments d'électronique et de robotique, surtout considérant son faible coût d'achat, qui est d'environ 30 $ pour un *Arduino* Uno. Il existe aussi des trousses éducatives offrant de nombreux projets pour un peu plus d'une centaine de dollars.

Raspberry Pi

Le *Raspberry Pi* est un petit ordinateur fonctionnant sous *Linux*, un système d'exploitation libre. Il occupe un très petit volume et offre plus de puissance de calcul qu'un *Arduino*. Ceci permet notamment de contrôler beaucoup de composantes ou d'appareils en même temps.

Le *Raspberry* Pi offre toutes les fonctionnalités de base d'un ordinateur : traitement de texte, tableur, logiciel de dessin, navigateur, etc. Il est également possible de programmer avec le langage Python.

Les possibilités offertes sont impressionnantes surtout si on considère son prix : il va de 60 $ à un peu plus d'une centaine de dollars pour une trousse contenant tout ce qu'il faut pour débuter avec le *Raspberry* Pi 3.

Remerciements et références

Nous tenons à remercier Brault & Bouthillier qui nous a permis de tester certains des produits mentionnés dans ce dossier.

Les produits mBot, Arduino Uno et Raspberry Pi ont été achetés chez Abra Electronics, à Montréal.

Références

A National Talent Strategy: Ideas for Securing U.S. Competitiveness and Economic Growth **:**
http://www.microsoft.com/en-us/news/download/presskits/citizenship/MSNTS.pdf

Activité de *débogage* : Brennan K., Balch C., Chung M. *Informatique créative*. Harvard Graduate School of Education, Creative Commons. En ligne :
https://files.inria.fr/mecsci/CreativeComputingFr/CreativeComputing20140806_FR_ttfrancais.pdf

Jeannette Wing: Wing J., « Computational Thinking », *Communications of the ACM*, March 2006, vol. 49,no 3.

Pensée informatique : Girard, M.-A. (2016). [Entrevue avec Alexandre Lepage] *La pensée informatique : accessible à tout le monde.* En ligne : www.ecolebranchee.com/2016/12/07/pensee-informatique-accessible/

Commentaires

Collaboration spéciale