L'enseignement de la **programmation** arrive dans les programmes scolaires

Cette année scolaire est un peu particulière car le code fait enfin son entrée officielle à l'école.

• Nicolas Decoster
Co-fondateur de **La Compagnie du Code** *et animateur*
Informaticien scientifique chez Magellium
@nnodot

Pourtant, et cela peut paraître surprenant, cela fait très longtemps que l'apprentissage du code a mis le pied à l'école. Par contre il n'y a que le pied qui est entré, pas plus… Et il en est même un peu ressorti par moments. En effet, beaucoup se souviennent des cours de LOGO (vous vous souvenez de la petite tortue à déplacer ?) ou des ateliers MO5 et TO7 de l'option informatique du bac (fin des années 80). Et depuis cette époque il n'y a pas eu grand-chose de plus, ou, du moins, rien de bien conséquent ; jusqu'à cette année, donc, qui voit une entrée franche de la programmation informatique dans les programmes de l'éducation nationale ! Il y a des notions de programmation en primaire et un peu plus au collège avec des concepts plus avancés qui apparaissent dans les programmes de mathématiques et de technologie. Par contre, il n'y a toujours pas de CAPES ou d'AGREG informatique qui garantirait un enseignement de qualité (bien que, autre nouveauté de cette année, il y ait maintenant une option informatique au CAPES de mathématiques), mais au moins cette présence officielle dans les programmes montre qu'il est jugé important que tous les enfants soient confrontés à la pensée informatique.

La question essentielle à se poser maintenant est : « quels moyens se donne l'éducation nationale pour permettre cet enseignement ? » Donc, il n'y a pas de CAPES ni d'AGREG pour l'instant (même si des acteurs reconnus comme la Société Française d'Informatique poussent pour qu'ils soient créés). Il faut donc former le corps enseignant actuel à ces nouvelles disciplines. Des professeurs de mathématiques et de technologies passionnés se sont déjà penchés sur la question et certains expérimentent de belles choses depuis quelques temps. Il y a une dynamique très positive. Mais cela ne fait pas tout. Il y a énormément de monde à former et cela va quand même être compliqué pour beaucoup d'enseignants de s'approprier cette nouvelle discipline si éloignée de leur expertise. En effet, l'informatique et les mathématiques sont deux disciplines complètement différentes, et, même s'il y a des ponts, un informaticien n'est pas forcément un mathématicien, et vice versa. Même chose pour les professeurs de technologie. Sans parler de la formation continue qui en est à ses balbutiements : les plans académiques de formation commencent à proposer quelques stages sur le sujet, des projets comme Class'Code (voir encadré) ou des ressources comme le livre "1, 2, 3… codez !" permettent aux enseignants qui en ont la volonté de se prendre en main pour s'auto-former. Certaines structures comme la Compagnie du Code proposent des accompagnements pour des établissements qui en ont la volonté. Mais tout cela n'est clairement pas suffisant pour un réel démarrage généralisé d'un enseignement de la programmation de qualité dans tous les établissements dès cette année. Espérons juste que les années de tâtonnements qui nous attendent ne dureront pas trop longtemps et que la montée en charge ne sera pas trop douloureuse pour un corps enseignant déjà bien malmené par ailleurs. Surtout, ajoutons qu'il est essentiel que cette tentative enfin un peu sérieuse d'introduction de la programmation à l'école ne retombe pas comme les précédentes. Mais soyons optimistes : beaucoup de voyants sont au vert et, même si ça pourrait être beaucoup mieux, il y a beaucoup de bonnes volontés qui œuvrent pour le succès de cette initiative. •

CLASS'CODE : Un projet dans lequel les développeurs peuvent aider les professionnel-le-s de l'éducation à initier nos enfants à la pensée informatique

Le projet Class'Code est un projet financé par le plan d'investissement d'avenir pour la mise en place d'une formation gratuite hybride à destination de tout pédagogue (enseignant ou animateur) non informaticien qui veut transmettre la pensée informatique et la programmation à des enfants. Hybride, parce qu'elle est composée d'une formation en ligne, et de temps de rencontre où les participants se retrouvent pour échanger ensemble avec l'aide de facilitateurs informaticiens qui, eux, ne sont pas pédagogues. Toute personne connaissant bien la programmation peut donc devenir facilitateur avec un minimum d'investissement : il faut juste se familiariser avec la formation (quelques heures), et, bien sûr, être présent aux temps de rencontre. D'après le site du projet : "Vous êtes un(e) professionnel(le) de l'informatique et vous désirez aider des débutants, partager votre expérience, votre culture scientifique et technique ? Sans pour autant tout savoir, vous êtes prêts à offrir un peu de votre expertise en informatique ? Bienvenue à Class'Code https://classcode.fr !" La formation est composée de cinq modules qui, au travers d'une pratique concrète de la programmation et d'activités débranchées, permettra de comprendre les concepts clés de l'informatique, et les enjeux de société qui y sont liés ; ceci pour être en capacité d'animer des ateliers sur ces sujets dès le premier module. Voici les cinq modules de la formation :

- 1 Module fondamental : découvrez la programmation créative ;
- 2 Module thématique : manipulez l'information ;
- 3 Module thématique : dirigez les robots ;
- 4 Module thématique : connectez le réseau ;
- 5 Module fondamental : le processus de création de A à Z.

Apprendre à programmer, pour quelle finalité ?

Cette question peut de prime abord sembler curieuse dans une telle publication. Il est vrai que jusqu'à récemment, hormis quelques périodes relativement limitées dans l'histoire de l'informatique (années 80 par exemple), apprendre à programmer était et reste principalement du fait des "geeks" et futurs développeurs de métier, une intersection large joignant ces deux catégories.

Youmna Ovazza
Fondatrice, Teen-Code (Teen-Code : stages et ateliers d'initiation à la programmation pour ados www.teen-code.com)

Aujourd'hui, alors que le monde entier se découvre un intérêt pour la programmation et pour son enseignement, sont-ce les mêmes motivations qui prévalent ? Celles des amateurs et novices reposent-elles sur les mêmes bases que celles des passionnés et professionnels ? Et sinon, comment les adresser ? Moi-même "non-geek" et initiée à la programmation sur le tard, dans un objectif d'éducation et d'information par rapport à un projet et non dans un but de reconversion professionnelle, je crois pouvoir avancer que les motivations de nous autres personnes ordinaires qui s'intéressent à la programmation ne sont pas tout à fait les mêmes que celles des experts du sujet ; et qu'il y a un grand malentendu sur le sujet, qui peut freiner l'accès à cet apprentissage, pourtant si riche, du plus grand nombre, si l'on ne prend pas davantage en considération les motivations des apprentis potentiels. Si l'on a certainement besoin des développeurs de métier et des passionnés pour transmettre leur savoir et leur enthousiasme, non, tout le monde :

- N'adore pas les robots ;
- Ne tombe pas en extase devant une imprimante 3D ;
- Ne joue pas à Minecraft, Pokémon Go ou World of Warcraft selon l'âge (ou pas :)) ;
- Ne rêve pas de faire le site de son CV en ligne en html/css !

Faut-il forcément apprendre tous les modes de cuisson de la viande pour se mettre à la cuisine ? Ne peut-on pas commencer directement par faire un gâteau, même s'il n'a pas 4 couches de crème et de génoise, un glaçage et s'il n'est pas moulé à la perfection ? Pourtant, au vu des cours et thèmes proposés pour s'initier à la programmation, il semblerait que seules les motivations "geek" prévalent : et on se demande, par exemple, pourquoi il y a toujours aussi peu de filles ? Mais aussi d'artistes, de littéraires, de sportifs, de parents, de personnes au foyer, etc. ? Pour qui en a fait l'expérience - mon propre témoignage - l'apprentissage de la programmation revêt de multiples intérêts, bien au-delà de la joie d'imprimer son porte-clé avec une imprimante 3D :

- Être enfin côté coulisses, et voir, même entrevoir, comment se fabrique une grande partie de notre monde actuel !
- Agir, au lieu de simplement consommer ;
- Comprendre par l'expérience ;
- Développer son autonomie de jugement par rapport aux nombreux enjeux du numérique ;
- Développer une plus grande efficacité dans les projets professionnels numériques ;
- Développer une meilleure collaboration avec ses collègues et partenaires développeurs ;

Ces motivations ont un champ d'application très large, en rapport avec tous les sujets et centres d'intérêt des personnes concernées ! Pourquoi ne pas partir justement de ces usages et centres d'intérêt, pour construire des cursus d'initiation et d'enseignement s'adressant au plus grand nombre, adaptés à cette logique de vulgarisation de qualité, et non une version "dégradée" d'un cursus professionnalisant avec toujours les mêmes langages et mêmes sujets ? Tout le monde a un téléphone dans sa poche, tout le monde peut être concerné un jour ou l'autre par le piratage et la cyber-sécurité, tout le monde utilise les réseaux sociaux... Tout comme la chimie appliquée à la cuisine, répondre à ces motivations requiert la rencontre de deux univers, de deux expertises : les programmeurs d'une part, les utilisateurs ou experts d'autres métiers, d'autre part.

Ce n'est qu'en joignant leurs forces et en mettant en commun leur créativité qu'on pourra véritablement adresser les motivations du plus grand nombre et intéresser de vastes populations. A l'heure où le numérique occupe une place si importante dans nos vies, ce ne serait pas du luxe que de se pencher un peu sur cette question qui touche autant à la citoyenneté qu'aux loisirs ou à l'efficacité économique ! •

INFORMATIQUE CRÉATIVE : *bien plus que l'apprentissage de la programmation*

• Nicolas Decoster
Co-fondateur de **La Compagnie du Code** *et animateur, Informaticien scientifique chez Magellium,* @nnodot

A l'origine de l'informatique créative, il y a l'idée d'utiliser l'ordinateur et la programmation comme prétextes pour créer des choses. L'apprentissage de la programmation n'est plus un objectif mais un moyen. On ne va pas apprendre à quelqu'un comment utiliser les boucles mais on va lui demander de faire une animation visuelle pour se présenter. N'étant plus une finalité, les apprentissages arrivent uniquement quand on en a besoin et l'accent est donné sur l'expérimentation et la découverte avec un maximum d'autonomie. Dans le projet que je suis en train de réaliser j'aimerais que mon personnage saute quand j'appuie sur la touche espace ? J'apprends par moi-même comment agencer des instructions d'événements, de déplacement et de contrôle pour donner l'illusion de ce mouvement. C'est ma motivation à réaliser un effet que j'ai décidé qui va me donner l'envie d'aller à la recherche du moyen pour le faire. Et, comme cela vient de mon désir, je retiendrai bien mieux ce que j'ai appris. La recherche de la manière de faire, va me pousser à chercher des informations en ligne ou auprès d'autres personnes, à me poser pour prendre du recul sur le problème en question et surtout à expérimenter.

La programmation est une activité qui se prête très bien à l'expérimentation. On pourrait presque dire, qu'en fait, l'expérimentation est l'essence même de la programmation. Le cycle de base de l'activité créatrice du développeur est : j'ai une idée, je réalise quelque chose, je teste, ça ne marche pas, j'analyse, je corrige, ça marche. Puis je passe à l'idée suivante. Cela permet aux enfants d'être dans le faire, de se familiariser avec le fait que c'est normal de se tromper et surtout de réussir à construire quelque chose et de trouver cela gratifiant. La programmation permet de très vite être confronté à cela.

Et, de plus c'est une opportunité pour certains enfants en difficulté de renouer avec l'apprentissage, le faire et la réussite. •

Où et comment coder en dehors de l'école ?

Il existe beaucoup de clubs d'informatique et de structures qui contribuent d'une manière ou d'une autre à la diffusion auprès des enfants et des ados, de la programmation, de l'informatique créative et de la pensée informatique. Il serait difficile de les nommer tous. Un premier point de départ est le site jecode.org qui contient une liste non exhaustive d'un grand nombre d'initiatives (http://jecode.org/initiatives). La typologie des structures est très riche : associations, centres de loisirs, entreprises, coopératives, établissements scolaires, éditeurs de logiciels et même des instituts de recherche impliqués dans la médiation scientifique comme l'Inria, qui est très actif. Il peut s'agir de structures dont la vocation est de proposer régulièrement des ateliers (comme Teen-Code ou la Compagnie du Code), ou bien des initiatives qui fournissent des ressources et qui organisent ponctuellement des événements sur ces sujets (comme Devoxx4Kids).

TEEN-CODE

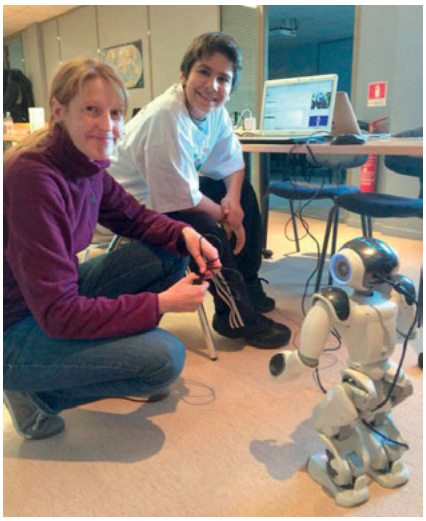
Teen-Code (http://www.teen-code.com/) initie les adolescents à partir de 13 ans à la programmation, à travers des stages de vacances et des ateliers thématiques, sur Paris. Teen-Code est une toute jeune société fondée par Youmna Ovazza, ex-Chief Digital Officer d'un groupe international de communication et spécialiste de marketing numérique, également formée à la programmation et mère de trois enfants. Teen-Code aspire à faire découvrir aux ados les coulisses de leurs usages quotidiens pour les rendre un peu plus acteurs et créateurs, et un peu moins simples consommateurs du numérique. La création d'applications mobiles sous Android est le contenu phare proposé par Teen-Code aujourd'hui, enrichi récemment de la création de jeux vidéo sous Unity, et bientôt d'autres sujets. Le parti pris est de traiter les ados comme des adultes en devenir ; les programmes sont donc conçus à partir des pratiques des professionnels et adaptés aux ados.

LA COMPAGNIE DU CODE

La Compagnie du Code (http://www.lacompagniedu-code.org/fr/) est une coopérative (plus précisément, une SCIC, pour société coopérative d'intérêt collectif) qui a pour vocation de transmettre la pensée informatique au plus grand nombre.
Elle est basée à Toulouse et organise des ateliers d'informatique créative pour les enfants sous forme de stages de vacances ou d'ateliers hebdomadaires, en extra et en péri scolaire. Elle est impliquée dans l'accompagnement des enseignants pour la prise en compte des nouveaux programmes de l'éducation nationale sur la programmation informatique. Elle anime également des ateliers et des enseignements pour les adultes pour démystifier le numérique et pour avoir un premier contact avec ce qu'est l'activité de développeur.
La CdC est portée par des informaticiennes et des informaticiens qui ont à cœur de transmettre leur goût de la création par la programmation et de démystifier le fonctionnement du monde numérique. Elle est impliquée dans l'accompagnement de l'introduction de la programmation dans les nouveaux programmes de l'éducation nationale en participant au Plan Académique de Formation, en intervenant dans des établissements et en étant partenaire du projet national Class'Code. Elle bénéficie du support d'étudiants du numérique (en particulier, des apprenants de Simplon.co et de Digital Campus).

DUCHESS FRANCE

Duchess France (http://www.duchess-france.org/) est une association destinée à valoriser et promouvoir les femmes développeuses, ou exerçant dans un domaine technique de l'informatique. Les actions de l'association visent à faire connaître ces métiers techniques notamment auprès des femmes et jeunes filles pour susciter de nouvelles vocations.



Duchess France publie des portraits de rôles modèles, intervient dans toute la France pour participer lors de forums et événements afin de parler du métier de développeuses à des collégiennes et lycéennes (Science Factor, Forum de la mixité, Science de l'ingénieur au féminin, HeForShe…), et organise plusieurs types d'ateliers : ateliers de prise de parole en public, ateliers mixtes de coaching "Call for Paper", ateliers techniques/de code, conférences, mais également ateliers d'apprentissage du code pour les enfants ainsi que les adultes débutants.

Autres structures

- Les Voyageurs du Code à Paris et dans certaines villes ;
- Magic Makers dans la région parisienne ;
- Tech Kids Academy à Paris et Saint-Germain-en-Laye ;
- Combustible à Toulouse ;
- Planète Sciences à Toulouse ;
- Alsace Digitale en Alsace ;
- Coding & Bricks dans le Nord Pas-de-Calais ;
- Ch'ti code liste les initiatives dans le Nord Pas-de-Calais ;
- Club Code France à Troyes ;
- Cod Cod Coding à Vandoeuvre lès Nancy ;
- Fantasticode à Gap ;
- Jeunes-Science Bordeaux à Bordeaux ;
- Geek School à Sophia Antipolis, Nice et Monaco ;
- Les Petits Hackers à Brest ;
- Savoirs Numériques Pour Tous à Grenoble ;
- MuseoMixLIM à Limoges ;
- Coding Gouters wiki qui liste les coding goûter de France ;
- Et pour bien finir, Devoxx4Kids France qui organise des ateliers dans toute la France.

Des **outils ludiques** pour tous les âges

Lorsque l'on pense à la programmation on se dit que son apprentissage va être très compliqué et semé d'embûches. Grâce aux outils existants, pour les enfants et adolescents, la programmation s'avère être un jeu d'enfant !

• Nicolas Decoster
Co-fondateur de **La Compagnie du Code** *et animateur*
Informaticien scientifique chez Magellium
@nnodot

Youmna Ovazza
Fondatrice,
Teen-Code
(Teen-Code : stages et ateliers d'initiation à la programmation pour ados
www.teen-code.com*)*

Aurélie Vache
Lead Developer chez
atchikservices
Duchess France Leader
@aurelievache

PROGRAMMATION PAR BLOCS

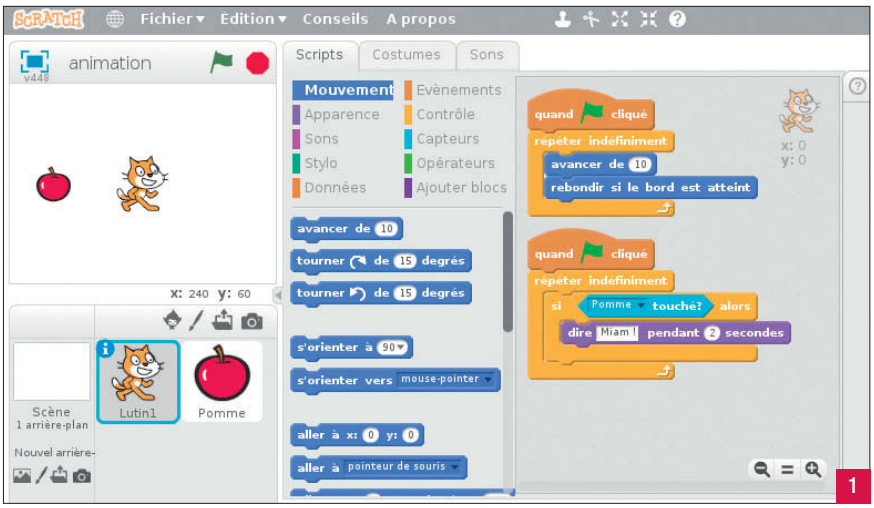
Lorsqu'on parle d'initiation à la programmation pour les enfants il est difficile de ne pas parler de Scratch. Il s'agit d'un outil de programmation visuelle par blocs qui a plus de dix ans. Il est développé par le MIT et remporte un très grand succès, auprès des petits et des grands, surtout depuis la version 2 qui est disponible en ligne sur un site communautaire qui promeut l'échange, le partage et le remix de projets (http://scratch.mit.edu). Sa syntaxe visuelle qui emboîte des blocs de textes comme des lego a inspiré de très nombreux outils similaires dont Snap! (un logiciel libre en ligne développé par Berkeley, une autre université aux USA) et Blockly (développé par Google). D'ailleurs le MIT et Google ont décidé de travailler ensemble pour réécrire Scratch, heuuu… from scratch ;-) entièrement en HTML et en JavaScript (alors que Scratch 2 est écrit en ActionScript et nécessite donc d'avoir un plugin flash pour fonctionner). Leur idée est de partir du moteur de rendu de Blockly et d'y brancher un moteur d'exécution Scratch. Le travail est en cours, affaire à suivre !

Scratch, la référence

Mais que peut-on faire avec Scratch ? En fait, il s'agit d'un langage tellement expressif que le mieux est d'en parler à l'aide d'un exemple. La figure ci-dessous montre l'interface de Scratch avec un exemple de programme. Si Captain Obvious était là, il dirait que ce programme fait se déplacer le personnage en permanence en le faisant rebondir sur les bords et en lui faisant dire "Miam !" dès qu'il touche une pomme. Voilà, un premier programme simple à faire mais qui produit déjà une première animation. [1]

L'interface de Scratch est basique. Il y a une scène à gauche, avec la liste des lutins en dessous, au milieu il y a les blocs utilisables regroupés par catégories et la partie programmation se trouve à droite. Pour programmer on glisse tout simplement les blocs avec l'aide de la souris et on les assemble comme des Lego. Nous n'allons pas rentrer dans les détails du fonctionnement de Scratch, il y a beaucoup de ressources en ligne pour cela et même de plus en plus de livres. Indiquons juste que Scratch permet d'utiliser beaucoup de concepts de base de la programmation : les séquences d'instructions, les exécutions conditionnelles, les boucles, les variables, les événements, les interactions avec l'utilisateur, l'affichage, l'encapsulation (on peut définir de nouveaux blocs), les messages, la création dynamique d'objets. Et, au final, les notions de base essentielles qu'utilisent les développeurs sont quasiment toutes là.

Cependant lorsque l'on commence à avoir une pratique un peu poussée de Scratch on atteint certaines limites : on ne peut pas définir de blocs qui retournent quelque chose, on ne peut pas attacher des informations aux messages que l'on envoie, on ne peut pas accéder aux informations ou aux actions des autres lutins, etc. Ces limitations qui sont réellement handicapantes lorsque l'on veut réaliser un projet un peu complexe, sont pleinement assumées par l'équipe de développement de Scratch : la volonté est de garder quelque chose de simple pour que cela reste un outil très accessible dédié à la découverte de la programmation, tout ce qui est un peu complexe à appréhender a été volontairement laissé de côté.



1

Snap! pour aller plus loin

Snap! reprend les principes et les fonctions de Scratch, et les deux outils sont très similaire d'aspect. Par contre Snap! offre plein de nouvelles possibilités qui pallient des manques, qui facilitent la vie du développeur, qui lui permettent d'explorer des concepts plus avancés ou qui lui ouvrent de nouvelles possibilités.

Au final, Scratch et Snap! sont tellement simples et amusants à utiliser que l'on pourrait penser qu'ils sont de simples jouets qui ne font qu'effleurer ce qu'est réellement la programmation. Mais cela est trompeur, au-delà de la simple découverte du code, il y a moyen de construire des choses ambitieuses et on peut aller très loin dans l'exploration de la programmation!

App Inventor

Curieusement méconnu en France, App Inventor (AI) est certainement une des meilleures

manières de s'initier à la programmation mobile, et à la programmation tout court, pour débutants. Conçu en 2009 par une équipe conjointe de Google et du MIT (Massachussets Institute of Technology) et depuis développé par ce dernier, App Inventor est un logiciel de programmation visuelle sous Android, (comme Scratch par exemple pour les enfants), basé quant à lui sur Blockly, open-source et accessible gratuitement en ligne en plusieurs langues.

Quelques chiffres clés pour en présenter l'impact international avant de détailler ses avantages : en 2015, la communauté d'utilisateurs de AI 3 était de millions d'utilisateurs issus de 195 pays différents. Plus de 100 000 utilisateurs actifs / semaine, ont construit à ce jour plus de 7 millions d'applis Android.

App Inventor, pour qui et pour quoi faire ?

Grâce à sa double interface de création, d'interface utilisateur d'une part, et de programmation d'autre part, AI permet de s'initier, bien au-delà de la programmation "pure", à l'ensemble de la logique qui préside à la création d'un projet : l'utilisateur est au cœur du projet, le design et l'ergonomie sont en permanence aussi importants et visibles que le code lui-même, et en cela, c'est un outil excellent pour s'initier à la programmation ou sensibiliser des profils non-techniques aux différentes dimensions qui y sont associées. [2]

App Inventor est également un outil qui permet la réalisation complète d'un projet, de zéro à l'installation d'une application fonctionnelle sur son téléphone ou sa tablette, au partage avec d'autres personnes voire à la publication sur le Google Play Store. C'est donc un outil qui n'est pas un simple bac à sable mais un véritable moyen de création d'une réalisation concrète qu'on transporte et manipule avec soi, dans sa poche, ce qui situe bien les enjeux de l'apprentissage en prise directe avec les usages quotidiens des utilisateurs.

Grâce à la grande variété des composants précodés déjà installés, AI se prête à de multiples sujets d'application, du dessin aux jeux vidéo ou à la photo en passant par la géolocalisation, la traduction, la gestion des SMS, les échanges d'information via Internet, etc. ce qui en fait un cadre flexible et ouvert aux centres d'intérêt du plus grand nombre, et notamment de tous les "non-geek" qui n'en utilisent pas moins un téléphone tous les jours !

Du cadre scolaire à la formation pour adultes, AI permet d'intégrer les bases de la logique et des bonnes pratiques de développement, en réalisant de vrais projets, sans y passer des mois : que demander de plus pour s'y mettre ?

PROGRAMMATION EN MODE TEXTE

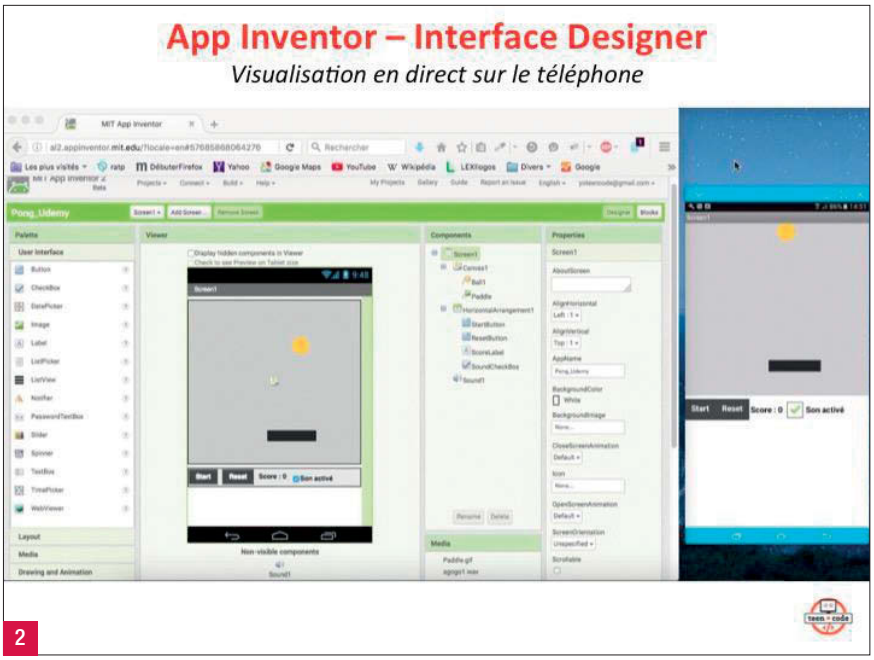
On l'a vu, les outils de programmation par bloc sont des candidats sérieux pour découvrir et approfondir la programmation et ses concepts. Cependant, actuellement le métier de développeur est essentiellement centré sur les langages de programmation en mode texte. De nos jours les logiciels, les applis, les sites Web font appel à des programmes écrits en C, C++, Java, JavaScript, Python, C#, bash, etc. L'attrait pour ces "vrais" langages de programmation par les enfants et les adolescents est donc tout naturel, ils ont envie d'entrer dans la cour des grands et de pratiquer les langages qui ont construit les logiciels qu'ils utilisent. Mais quels sont les langages textuels qui sont adaptés pour eux dans l'optique d'apprendre la programmation ? Et quels sont les avantages, les inconvénients et les spécificités par rapport à la programmation par blocs ?

En fait chaque langage a ses particularités qui fait qu'il sera plus ou moins adapté à l'apprentissage. Certains sont plus complexe, d'autres sont moins accessibles et d'autres sont des langages obligés pour un usage particulier. Passons en revue certain d'entre eux.

Tout d'abord le langage le plus disponible : JavaScript. Ce langage est présent partout, il suffit d'avoir un navigateur Web et vous pouvez faire du JavaScript. Vous ouvrez votre éditeur de texte préféré, vous mettez quelques lignes de code dans un fichier et vous l'exécutez avec votre navigateur. De plus, la plupart des navigateurs embarquent des outils de développement intégrés (console, débugger…). C'est le standard Web reconnu et beaucoup d'efforts sont fait pour disposer de moteurs performants : Google, Mozilla, Microsoft et Apple sont tous en compétition sur ce terrain. On dispose même de moteur JavaScript en dehors des navigateurs Web (le projet node.js est un moteur en ligne de commande). Cela fait que JavaScript est un langage avec lequel il faudra compter encore longtemps et sa communauté est l'une des plus bouillonnantes du moment.

Ensuite, parlons d'un vieux langage qui est le socle de beaucoup de choses : le langage C. Beaucoup de logiciels existants sont écrit en C, comme de grands pans du système Linux par exemple. C'est également un langage de prédilection pour de l'informatique embarquée car il est bas niveau et très performant. En particulier, c'est le langage qui est généralement utilisé pour programmer les cartes Arduino, et il existe tout un outillage pour s'y mettre facilement. Un avantage du langage C, qui peut être aussi un inconvénient, est qu'il est plus bas niveau que beaucoup d'autres, en particulier sur les problématiques de gestion de mémoire : c'est instructif sur le fonctionnement interne d'un programme mais c'est plus délicat à programmer. Ce langage à l'avantage, ou l'inconvénient ;-), d'imposer au développeur de se frotter aux joies de la compilation et de son lot d'erreurs.

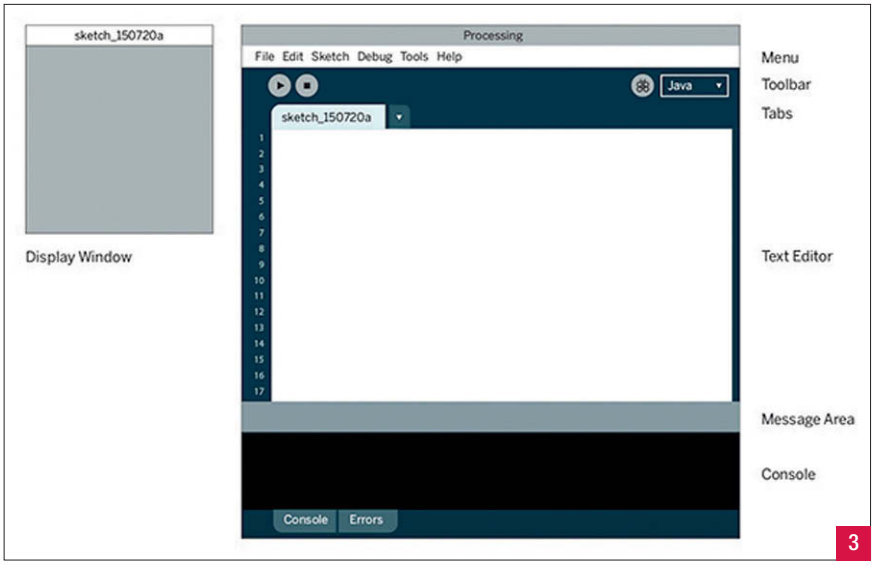
Un langage qui est un peu moins bas niveau est Python. C'est un langage interprété ce qui, en particulier, évite les problèmes de compilation. Il est très populaire et offre un écosystème très



2

riche. Quel que soit le domaine qui vous intéresse vous trouverez des modules python pour vous faciliter la vie : jeux, calcul scientifique, serveur Web, visualisation de données, communication avec du matériel, etc.

Enfin, parlons de Java, langage très utilisé en entreprise. C'est également un langage compilé (un peu comme le C mais avec quelques nuances), il est portable (comme JavaScript et Python) et dispose également d'une très grande communauté (mais peut-être avec un écosystème moins riche que pour Python). C'est également un langage très présent dans des problématiques d'actualité comme la Big Data. Java est le langage de la bibliothèque processing qui est très adapté à l'informatique créative et qui permet de découvrir la programmation de manière ludique (voir les figures ci-contre). [3]



Interface de développement de processing

```
void setup() {
  size(480, 120);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```

[4]

Exemple de code processing et le résultat associé

Rappelons également que le Java est utilisé pour programmer des applications mobiles sous Android.

Citons rapidement le C++ pour dire qu'il s'agit un langage très intéressant et très actif mais qui est plus difficile d'accès (on peut y venir après avoir appris le langage C), et la programmation *shell* (comme *bash*) qui est radicalement différente des autres mais qui présente un certain intérêt en tant que tel et qui est quasiment incontournable dès qu'on veut se plonger dans le fonctionnement des systèmes Unix (comme Linux). Il y a donc deux grandes grandes approches pour programmez et, donc, pour apprendre à programmer. Soit la programmation par bloc, soit la programmation en mode texte. Voyons les avantages et les inconvénients du code texte par rapport au code bloc.

Inconvénients :

- Le programmation textuelle est en général moins accessible, il faut souvent des outils à installer avant de pouvoir commencer à coder. Cependant, il y a de plus en plus de possibilités pour coder en ligne dans ces langages ;
- Souvent le résultat en mode texte est accessible moins rapidement qu'en mode bloc. Par exemple, avec Scratch ou Snap! on clique sur un bloc "avancer de 10" et le personnage avance tout de suite. Dans certains langages textuels il faut écrire le code, sauvegarder le fichier, éventuellement compiler, exécuter pour enfin voir le résultat ;
- Dans le même genre d'idée, en général il n'y a pas d'interactivité ou de *live coding*. Dans Scratch ou Snap!, pendant que le programme tourne on peut changer une valeur ou ajouter un bloc et l'exécution prend en compte la modification instantanément. Cela ouvre des possibilités d'explorations et d'expérimentations plus importantes. La plupart des langages textuels ne permettent pas cela (bien que cela commence à arriver, en particulier avec les développement Web en JavaScript) ;
- Pour développer en mode texte, même si ce n'est pas indispensable, c'est souvent mieux d'avoir un bon outillage et il peut parfois être complexe à mettre en place : il faut trouver le bon éditeur, le bon débogueur, le bon *linter*, etc. Mais on peut trouver quand même des choses bien intégrées, voire en ligne ;
- Avec un langage textuel on est confronté aux erreurs de syntaxe qui par définition ne sont pas présentes avec les blocs. Cependant un bon *linter* permet de limiter les dégâts ;
- Il en est de même avec les erreurs de compilation, c'est un frein supplémentaire, même si un bon outillage permet d'aider le développeur dans la recherche des problèmes.

Avantages :

- Comme nous l'avons dit, actuellement l'essentiel de la programmation se fait en mode texte. S'y mettre c'est se former à l'existant ;
- Un simple éditeur de texte est suffisant pour créer quelque chose, pas besoin d'une interface graphique ;
- Les langages textuels peuvent être plus puissants dans ce qu'ils permettent de faire ;
- Ils peuvent être plus bas niveau et donc offrir un contrôle plus fin de certaines choses (comme la gestion de la mémoire par exemple) ;
- Ils permettent de développer des choses complexes d'une certaine ampleur.

POUR LES PLUS PETITS

On a vu qu'il y avait des outils pour les adolescents, pour les enfants mais qu'en est-il pour les plus petits, peuvent-ils également être initiés au merveilleux monde de la programmation ?

L'initiation de la programmation n'est pas réservée qu'aux grands, les petits sont également gâtés. Avez-vous déjà entendu parler de Cubetto ou de Thymio, non ? Nous allons vous guider vers ces fabuleux outils.

Cubetto (à partir de 3 ans) [5]

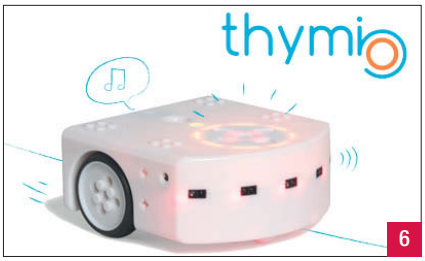
Cubetto, sous ce nom se cache un mignon petit robot en forme de cube. Une campagne Kickstarter a été lancé cette année et a été plus que financée, plus de 1 596 457 $

sur les 100 000 $ initiaux ont été réunis par 6 553 contributeurs ! Lorsque l'on sait que cette campagne de crowd-founding a été soutenue

par Randi Zuckerberg (sœur aînée de Mark Zuckerberg et entrepreneuse américaine) et Massimo Banzi (le co-fondateur d'Arduino), on se dit que ce succès est normal et mérité.
Le but de ce robot est d'apprendre les bases de la programmation à des enfants (filles et garçons) de trois ans et plus.

Comment ?

Le kit est composé d'un tapis de jeu, d'un livre d'instructions, d'une console de programmation et de 16 blocs d'instructions. Les blocs d'instructions permettront au robot d'avancer, de reculer, de tourner à gauche, de tourner à droite ou d'exécuter une fonction.
Si l'enfant désire que le robot aille d'un point A à un point B, il va devoir concevoir le programme en disposant les blocs d'instructions sur la console de programmation. Lorsque le bouton de démarrage est pressé, le robot se met en marche et suit les instructions sur le plateau de jeu. Un robot que l'on peut piloter et programmer en le touchant, sans être devant un écran, quelle façon ludique et sympathique d'initier les enfants ! :-) Lorsque l'on voit le petit cube en bois, on tombe immédiatement sous son charme, par contre ce coup de cœur aura un coût, il est disponible sur le site https://www.primotoys.com/ au prix de 225$, et 245$ pour la version avec 5 cartes et 5 storybooks.



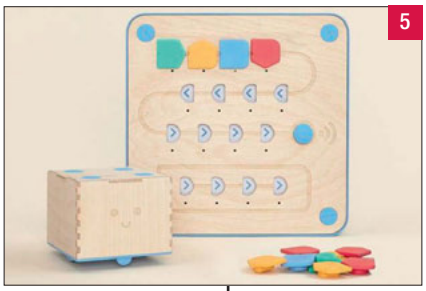
6

Thymio (de 4 à 15 ans) [6]

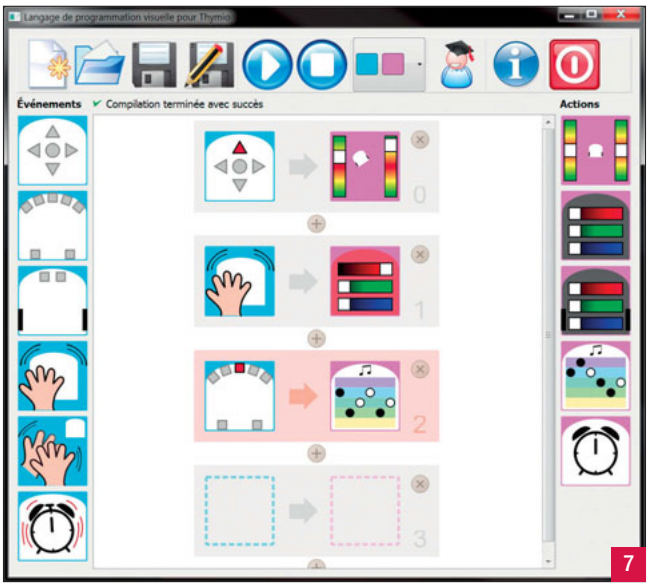
Thymio est un petit robot, d'origine suisse, qui permet de découvrir l'univers de la programmation, de la robotique et d'apprendre le langage des robots. Le robot est programmé avec six comportements : amical, explorateur, craintif, explorateur, obéissant et attentif. Grâce à ces comportements préprogrammés, l'enfant démarre son robot, appuie sur le bouton central permettant de sélectionner une couleur pour démarrer le comportement, il peut ainsi voir son petit robot rouler, bouger, changer de couleur, se comporter comme demandé.
Ensuite l'enfant peut passer à la seconde étape, programmer son robot !
Ce qu'il y a de super sympa avec Thymio c'est que l'apprentissage de la programmation et de la robotique avec ce robot s'adapte à l'âge et au niveau de l'enfant. En effet, plusieurs modes de programmations sont possibles : programmation visuelle, programmation blocky, programmation scratch et programmation texte.

Langage de Programmation Visuelle (VPL) pour Thymio [7]

Quoi de mieux que découvrir la programmation grâce à la programmation visuelle ? L'enfant n'a pas besoin de savoir lire, en glissant les pictos/les blocs d'images au centre de l'écran, l'enfant peut demander au robot d'exécuter des instructions. Il suffit d'appuyer sur Play (bouton avec un triangle) et le tour est joué. Programmer un Thymio avec cette interface c'est vraiment un jeu d'enfant.


5

"A playful programming language you can touch. Montessori approved, and LOGO Turtle inspired. Learn programming away from the screen."



7

Blockly [8]

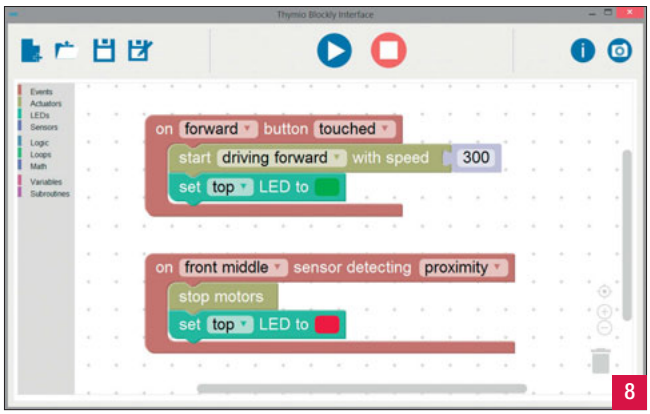
On connaît plus Scratch, au moins de nom, mais un autre langage de programmation par blocks existe. Développé par Google, *Blockly* (https://developers.google.com/blockly/), une bibliothèque logicielle JavaScript permettant de créer des environnements de développement utilisant un langage graphique, est le pont idéal entre la programmation visuelle et textuelle. En déplaçant les blocs d'instructions l'enfant/l'adolescent va pouvoir demander au robot ce qu'il souhaite qu'il fasse, et va ainsi s'initier à la programmation : aux boucles, aux conditions…
Avec Blockly l'enfant pourra utiliser 100% des capacités du robot sans écrire une seule ligne de code tout en apprenant la logique de l'informatique et de la robotique.

Scratch

Si vous ou votre enfant avez plus l'habitude de Scratch, il est également possible de programmer son robot Thymio en Scratch grâce à Asebascratch, une connexion logicielle entre Scratch 2 et le Thymio, qui permet aux programmes Scratch et à ses lutins d'interagir avec le robot.

Programmation texte avec Aseba Studio

En démarrant l'IDE on entre dans le vif du sujet, dans la programmation à part entière. Grâce à la programmation textuelle l'adolescent peut accéder à tous les capteurs du robot, vérifier l'état du thymio, créer des graphiques en temps réel, contrôler les LEDs, les moteurs, donner des instructions de plus en plus complexes au robot…
Si l'on veut par exemple faire dessiner notre Thymio, il suffit de mettre un stylo ou feutre dans le trou et d'exécuter le code suivant :

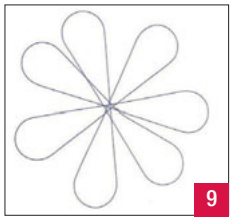


8

```
var itera = 0
timer.period[0] = 1000 # 1000ms = 1s

onevent timer0
  itera = itera + 1
  if itera==1 then
    motor.left.target = 230
    motor.right.target = -120
  end
  if itera==4 then
    motor.left.target = 80
    motor.right.target = 80
  end
  if itera==7 then
    itera = 0
  end
```

Le dessin qu'il en résultera sera le suivant : [9]
Vous pouvez essayer de modifier les valeurs des variables **motor.left.target** et **motor.right.target** et tester afin de voir quel sera le rendu.


9

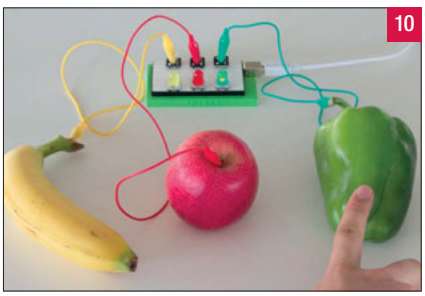
Autre cas d'utilisation, on met le robot sur une feuille blanche comprenant un chemin en noir, si l'on souhaite que le Thymio suive le tracé noir, vous pourrez utiliser le code suivant :

```
onevent startup
  mini=1023
  maxi=0
  mean=512
  vari=512
  ireg=0

  speed=100 #or another value

onevent prox
  p1=prox.ground.delta[1]
  callsub statistics
  call math.muldiv(ndev,SIGMA,p1-mean,vari)
  if abs(ndev)<SIGMA then
    preg=60*ndev/100
    ireg=ireg+10*preg/30
    motor.left.target=speed+(preg+ireg)
    motor.right.target=speed-(preg+ireg)
  else
    ireg=0
    motor.left.target=1*ndev/2
    motor.right.target=-1*ndev/2
  end

sub statistics
  call math.max(maxi,maxi,p1)
  call math.min(mini,mini,p1)
  if maxi-mini>400 then
    call math.muldiv(vari,45,maxi-mini,100)
    mean=(mini+maxi)/2
  end
```

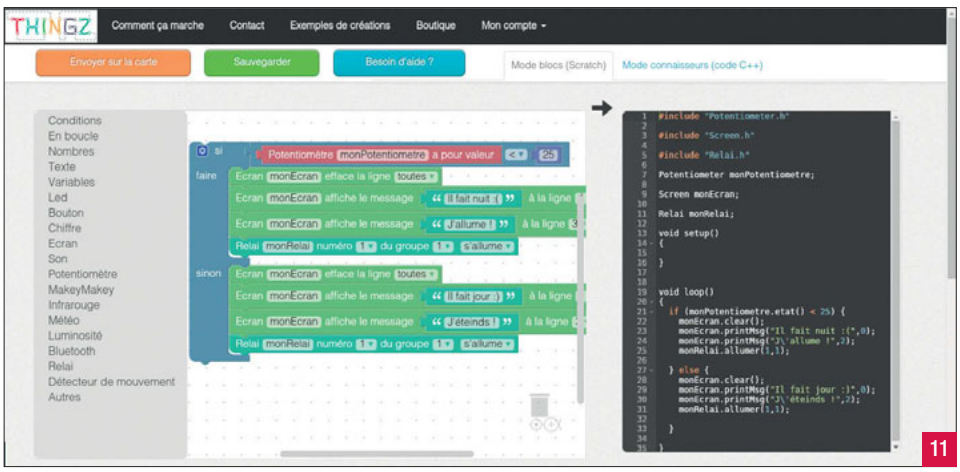

10

Exemples de montages Thingz et de modules disponibles, dont le module Bluetooth et le module simple connecteur qui permet de s'interfacer avec des… fruits et légumes ! comme avec Makey Makey.

Côté prix, il vous faudra débourser 179 CHF (soit environ 165€) pour la version sans-fil du Thymio, je vous invite à aller sur le site de Thymio (https://www.thymio.org/) pour découvrir les autres versions du petit robot ainsi que les nombreux packs.
Je vous avoue que le petit Thymio me tente bien car contrairement à Cubetto, même un adolescent pourra s'en amuser car il pourra programmer son robot en programmation blocks puis texte, donc cela peut être un investissement sur le long terme, et même pour nous en tant qu'adulte, développeur ou non, cela peut nous permettre de nous familiariser avec la robotique (et à l'IoT) en douceur grâce aux capteurs (de proximité, de température, de sol, microphone, accéléromètre…) que possède le robot.
Quelques classes ont testé l'initiation de la programmation à l'école grâce à ces robots, je vous invite à lire leur histoire, notamment celle dans l'académie de Strasbourg : http://tice67.site.ac-strasbourg.fr/wordpress/aspects-de-la-programmation-de-thymio-en-classe/. Je vous invite également à découvrir des exemples d'activités à faire avec un Thymio : http://www.netpublic.fr/2016/08/robots-et-code-pour-les-enfants-25-fiches-pour-mettre-en-place-des-activites-pedagogiques/
Si vous désirez des tutoriels, des exemples d'instructions à donner à votre Thymio, je vous invite à aller vers le Github de aseba-community : https://github.com/aseba-community

THINGZ

Pour apprendre à programmer avec un pied dans le monde réel, il y a Arduino. Avec ce système, on assemble des composants électroniques à une carte et on programme en langage C le comportement du montage. C'est très bien pour se construire des systèmes complets en se frottant aux concepts bas niveau de l'électronique et de la programmation, mais cela demande un investissement non négligeable avant de pouvoir créer un objet fonctionnel. Pour ceux qui veulent accéder à l'essence de la construction et de la programmation de ce type de dispositif mais de façon plus simple et ludique, il y a **Thingz**, qui permet de créer des objets en branchant des briques comme des LEGOs et d'apprendre la programmation pour devenir l'inventeur d'objets électroniques.
Le fonctionnement est vraiment très simple. Il y a une base dans laquelle on branche des modules : led, bouton, buzzer, mini écran, afficheur sept segments, Bluetooth, capteur de température ou de mouvement, simple connecteur (à la Makey Makey), etc. On branche la base sur un prise USB de l'ordinateur et en allant sur le site de Thingz on peut créer un programme soit en mode bloc, soit en mode texte. On peut d'ailleurs commencer avec les blocs, puis voir le code texte généré pour ensuite éventuellement le modifier. Cela peut être une très bonne manière de passer d'un mode de programmation à l'autre. [10 et 11]



11

Exemple d'un programme avec le code bloc et le code texte correspondant

ET LA ROBOTIQUE DANS TOUT ÇA ?

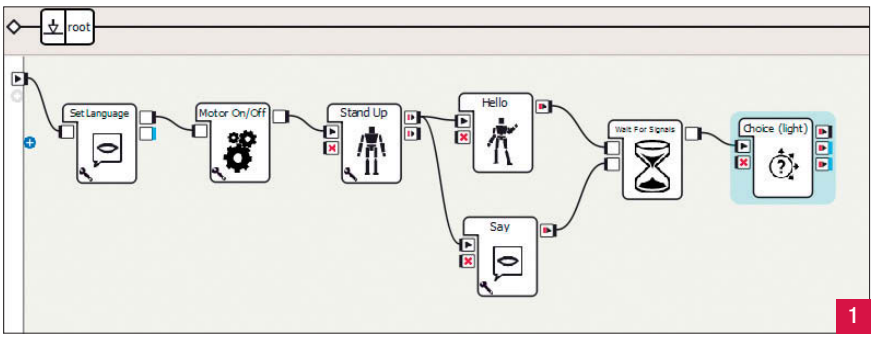
Lors de Devoxx France cette année j'ai découvert le célèbre robot Nao d'Aldebaran, enfin, de SoftBank Robotics Europe, et j'ai été bluffée. En fin d'année dernière, Blandine et Corinne, de Duchess France, ont animé un atelier Nao lors du RivieraJUG Devoxx4Kids 2015 à Sophia Antipolis, voici un retour de Blandine sur cette expérience :

Blandine Bourgois
Développeuse Java chez
ENOVACOM
Duchess France Leader
@bbourgois

C'est grâce à Corinne (Krych), une autre Duchess, que je me suis retrouvée dans l'aventure. Elle avait participé à l'édition de l'année précédente de Devoxx4Kids Sophia par le RivieraJUG et m'a proposé de rejoindre l'équipe.
J'ai la chance d'avoir un robot NAO chez moi mais faute de temps et de motivation (ce n'est pas forcément amusant toute seule) il reste souvent au placard. Je savais qu'il existait déjà des ressources pour faire un atelier pour enfant avec NAO (atelier Devoxx4Kids : http://www.devoxx4kids.org/materials/workshops/nao-robot/, atelier de Nicolas Rigaud : http://fr.slideshare.net/NicolasRigaud/nao-robot-workshop-for-kids-2-french-46241830), je me suis dit que c'était l'occasion de me lancer.

Le temps de préparation a été assez court : récupérer le TP, le tester et faire quelques modifications puis expliquer à Corinne le fonctionnement de chorégraphe (le logiciel permettant de programmer NAO). Comme c'était une 2ème édition, l'organisation était déjà rodée (salle, goûter, planning, inscription …).

Le jour J, je suis juste arrivée un peu plus tôt pour préparer les machines et c'était parti pour deux sessions de 1h30 avec 4 binômes d'enfant de plus de 10 ans pour 3 encadrants. Ludovic, le fils de Corinne, était venu nous prêter main forte. Le but étant que les enfants puissent faire plusieurs ateliers dans l'après-midi.



1

L'atelier NAO consiste à programmer le robot pour lui faire raconter une histoire. Le début était détaillé en pas à pas (histoire de prendre ses marques : mettre le robot en français, allumer les moteurs, tester les briques parler et question) puis les enfants pouvaient laisser libre court à leur imagination. Le TP était réalisé sur les PC et testé dans un simulateur puis chaque binôme pouvait lancer son programme sur le robot une fois fini.

Développer sur un robot peut paraître compliqué et rébarbatif. Heureusement, le robot NAO est fourni avec des briques de bases. Il suffit d'appeler la bonne API pour que le robot parle ou se lève. Ce qui est plus déroutant, c'est l'ordonnancement des actions. Le robot peut bouger et parler en même temps. Concrètement, dans Choregraphe, l'IDE, on retrouve toutes les actions de base dans des boîtes (ces boîtes appellent les méthodes des API des briques de base). Chaque boîte a des entrées (start et stop), des sorties et des paramétres. Pour créer un programme, il suffit de relier les boîtes entre elles. Le côté graphique permet une prise en main rapide et les boîtes de logique (if, wait for signals, switch …) une découverte en douceur de l'algorithme.

Dans l'exemple ci-dessous, le robot se lève puis parle pendant l'animation Hello. [1]
C'est un très bon souvenir et mon robot n'a pas souffert. Chaque binôme a fait un programme très différent. Certains n'ont même pas suivi la partie pas à pas pour partir à la découverte des possibilités du robot.

En bilan, c'était une après-midi intense ; on a initié 16 enfants. Mon Nao connaît maintenant quelques histoires farfelues, les enfants étaient très contents : ils nous ont même remerciés, et les parents ont apprécié les démos.

Play'n'Code : un jeu pour apprendre à coder

• Christelle PLISSONNEAU CEO

Early Birds Studio, une startup née d'étudiants d'Epitech, développe Play'n'Code : le jeu vidéo pour PC et Mac qui apprend aux enfants de 8 à 12 ans à coder en s'amusant ! Découvrons ensemble les dessous du projet.

Comment installer Play'n'Code ?

Depuis votre ordinateur, rendez-vous sur le site Internet de la startup https://early-birds-studio.fr et cliquez sur « Jouer à la démo ». Choisissez PC ou Mac, un installateur va se télécharger. Suivez les instructions pour installer le jeu sur votre ordinateur. Une fois le jeu ouvert, vous arrivez devant les portes de votre vaisseau spatial ! Pour y rentrer, vous devrez créer un compte famille. Cliquez sur « Créer un compte » vous emmènera sur le site Internet. Une fois vos informations remplies, un mail de confirmation vous sera envoyé. Après avoir validé votre email, vous pourrez revenir dans le jeu et vous connecter pour rentrer à bord de votre vaisseau ! A partir de là, vous pourrez laisser vos enfants prendre les commandes, le jeu est en pilote automatique. Il pourra créer une nouvelle partie et choisir son personnage pour commencer le jeu.

Play'n'Code, c'est quoi ?

Après avoir choisi un garçon ou une fille dans le menu principal, les enfants choisissent un nom de héros, saisissent leur âge, et le jeu démarre ! Malheureusement, pendant le chargement du jeu, le vaisseau s'est écrasé sur une planète inconnue. Les pièces du vaisseau se sont éparpillées aux quatre coins de cette planète, et le but est de les retrouver pour réparer le vaisseau et le faire rentrer chez lui. L'enfant va devoir programmer un petit script à la fin du jeu grâce à tout ce qu'il a appris au cours du jeu.

Pour récupérer les pièces du vaisseau, votre enfant va parcourir la planète sur laquelle il s'est écrasé, et parler aux différents personnages qui se trouvent sur son passage. Avec un peu de chance, ils auront des pièces du vaisseau ! Et surtout, il va découvrir la programmation en faisant des mini-jeux. TTY, le robot copain du joueur, volera à ses côtés tout au long de l'aventure pour lui expliquer les notions de programmation qui sont expliquées.

Qu'est-ce qu'on apprend dans ce jeu ?

L'équipe a créé un langage de programmation en français, sans ponctuation complexe telle que le point-virgule (pour fermer l'instruction) ou l'accolade (pour délimiter une fonction). Ce langage, proche du C ou du Lua, permet aux enfants de comprendre la logique du code, travailler l'algorithmie, tout en se rapprochant au maximum des langages existants sur le marché et qui sont en anglais. Un programme pédagogique a été créé autour de ce langage, et les notions que l'on retrouve dans tous les langages et tous les programmes, sont expliquées : variables, boucles, conditions et fonctions. Au début du jeu, les enfants apprennent à se servir des touches du clavier de l'ordinateur, la souris, et la notion de « programmation » leur est expliquée au moyen de mini-jeux. Toutes les notions sont expliquées en plusieurs étapes, qui sont elles-mêmes sous découpées selon le degré de difficulté de l'exercice. Les notions de code sont expliquées simplement au début, dans leur concept. Par exemple, une variable est une petite boîte dans laquelle on peut mettre des choses. Mais pas n'importe lesquelles : chaque conteneur ne peut accueillir qu'un type de contenu (ex : de l'eau dans un verre). Les syntaxes des notions sont ensuite expliquées, au moyen de glisser-déposer comme Scratch. Le but ici est d'afficher les premiers bouts de code liés aux notions que l'enfant a vues dans la première partie. Dans la 3e étape, les enfants sont amenés à réutiliser les mots-clés qu'ils ont vus dans les 2 premières étapes, et les écrire eux-mêmes. Des tutoriels sont là pour expliquer, et les enfants peuvent les regarder aussi souvent qu'ils veulent. La dernière étape arrive à la moitié du jeu. Rapidement, les enfants vont devoir écrire leur premier script ! Ils ont vu tout le programme pédagogique, et ils sont de plus en plus autonomes.

Organisez le jeu en sessions

Avec une quarantaine de mini-jeux, le jeu dure au moins 4 heures. L'équipe conseille d'organiser des sessions de 30 minutes pour ne pas noyer vos enfants.

Une application mobile « Compagnon » est disponible pour suivre la progression des enfants dans le jeu : temps total sur le jeu, temps passé par mini-jeu, explication des notions de programmation qu'ils ont vues… Une alerte est paramétrable depuis l'application mobile pour que le petit robot TTY déclenche un message dans le jeu : « Et si nous revenions plus tard ? Je commence à être fatigué ! ». •

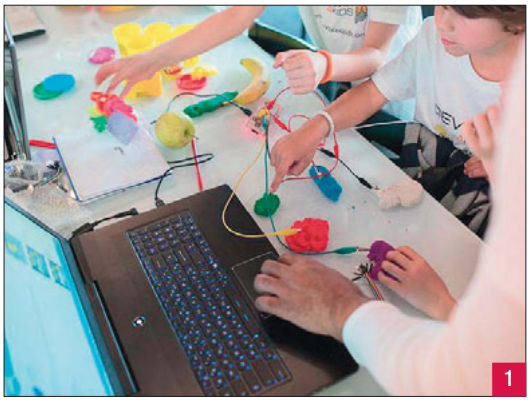
Quelques outils et matériels

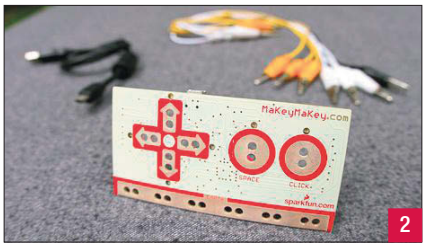
(liste non exhaustive)

MAKEY MAKEY

Même s'il ne s'agit pas directement de programmation, il y a un compagnon incontournable pour piloter vos créations. Nous voulons parler de Makey Makey, un prolongement tentaculaire de votre ordinateur. Makey Makey est un kit basé sur Arduino qui permet d'interfacer des objets conducteurs d'électricité (des fruits par exemple) avec l'ordinateur et de faire des applications amusantes. Le Makey Makey émule les 4 flèches du clavier et les 2 boutons de la souris. Le fonctionnement est simple : on branche le Makey Makey sur un port USB, on branche quelques fils du Makey Makey sur, disons, quelques bananes, et juste en touchant les fruits vous pouvez piloter votre jeu (ou tout autre logiciel en fait) ! Cela fonctionne par exemple aussi avec de la pâte à modeler. En réalité ce dispositif est vu par l'ordinateur comme un clavier et il affecte automatiquement chaque fil (et donc ce à quoi le fil est connecté) à une touche du clavier (les quatre flèches, la touche espace…). Voilà, aucune configuration, on branche et ça marche. On vous invite à aller regarder la vidéo de présentation de Makey Makey (https://www.youtube.com/watch?v=rfQqh7iCcOU) pour voir tout ce qu'on peut faire : un piano avec les marches d'un escalier, une fausse guitare en carton qui joue les sons de l'ordinateur, etc. On vous fait confiance pour trouver d'autres idées bien délirantes. [Fig.1, 2 et 3]

Si vous voulez aller plus loin avec Makey makey : http://magicmakers.fr/blog/comment-faire-un-petit-jeu-fun-avec-scratch-et-makey-makey


1



2

3

LEGO MINDSTORMS

(à partir de 10 ans)

Les Lego Mindstorms sont des Lego bien particuliers : équipés de plusieurs moteurs, de capteurs et d'une brique programmable, ceux-ci peuvent exécuter des programmes écrits à l'aide du logiciel fourni par LEGO. L'éditeur de Lego peut, par certains aspects faire penser à Scratch. On y assemble des briques pour construire notre programme, chaque brique correspondant à une instruction. De la même manière, les briques de même nature sont rassemblées ensemble pour nous aider à nous y retrouver.

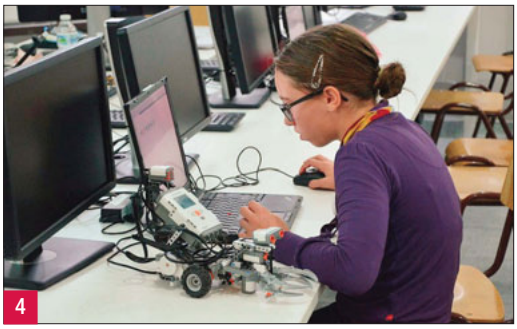
Par exemple, toutes les instructions concernant les moteurs seront regroupées ensemble, alors que tout ce qui concerne les boucles sera dans un autre groupe.

Pour nous aider, le kit propose plusieurs modèles de montages (véhicule, robot, araignée), chaque modèle ayant également plusieurs tutoriels intégrés.

Une fois le programme écrit, un câble va nous permettre de le charger dans la brique programmable et de le faire exécuter par celle-ci.

Le kit est fourni avec 3 moteurs : deux qui servent au déplacement de la structure, le troisième servant à faire fonctionner d'autres mécanismes montés par l'enfant comme un lance bille par exemple.

On peut ajouter à ces moteurs un détecteur de couleurs et un capteur d'ultrasons, ce qui peut par exemple nous permettre de faire se déplacer un véhicule sur un circuit et d'adapter son comportement en fonction des couleurs qu'il détecte sur ce circuit, ou encore de stopper lorsque son capteur ultrason rencontre un obstacle.


4

5

Deux versions sont disponibles :
la version "grand public" fonctionne avec des piles, la version "éducation" est un petit peu plus chère mais fonctionne sur batterie rechargeable, ce qui peut être un plus en atelier.

Ces Legos sont plutôt destinés aux jeunes de plus de 10 ans, le montage et l'interface pouvant paraître un peu complexes à la plupart des enfants plus jeunes. [Fig.4 et 5]

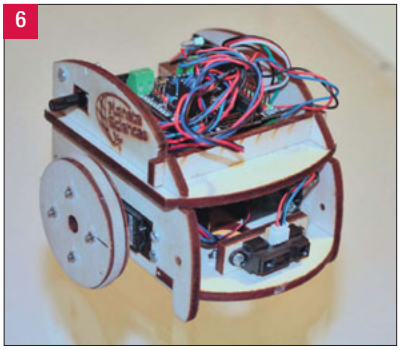
Si vous voulez aller plus loin, l'association Devoxx4Kids France met à disposition une documentation sur les Mindstorms permettant de créer des ateliers sur ce sujet : http://devoxx4kidsfr.github.io/materials/ateliers/mindstorm/index.html

BOUM'BOT, le robot du bricoleur

L'association Planète Sciences est particulièrement active dans la pédagogie en robotique notamment à travers l'organisation de concours : les Trophées de la Robotique, pour les moins de 18 ans, et la Coupe de Robotique, pour les adultes et en particulier les étudiants. Il y a quelques années, elle s'est lancée dans le développement d'un robot en kit pour initier les jeunes et moins jeunes à la programmation.

Boum'Bot — c'est son nom — est fabriqué uniquement à partir de matériel que l'on peut

acheter dans le commerce : carte Arduino UNO, shield et capteurs DFRobot, servomoteurs standards… Côté structure, le châssis est découpé au laser dans du contreplaqué et peut être réalisé dans n'importe quel FabLab.
Le robot dispose de 2 moteurs pour se déplacer (pilotables en vitesse) et d'un petit haut-parleur pour émettre des sons. Côté capteurs, il dispose d'un capteur de distance infrarouge et de deux capteurs de ligne au sol. L'ensemble des plans et des manuels est téléchargeable sur le site Web de l'association (http://www.planete-sciences.org/robot/boumbot).
La boutique Arobose propose un kit avec tous les éléments inclus au prix de 90€.
[Fig.6]



Côté programmation, on peut au choix programmer directement dans l'Arduino IDE en C++, ou alors en utilisant l'interface UniBot, développée par l'association, qui propose de programmer à l'aide de blocs qui s'emboîtent. Tous les concepts fondamentaux de la programmation peuvent être abordés : séquences d'instructions, tests et sélections, boucles "tant que" et variables. Des notions plus avancées comme l'asservissement et les contrôleurs PID peuvent être mises en pratique lors de concours de suivi de ligne par exemple ou des sorties de labyrinthe.
Le robot est utilisé régulièrement par l'association lors d'évènements comme la fête de la science ou en partenariat avec les écoles, collèges et lycées (voire même parfois à l'université !).
L'utilisation d'un matériel 100 % standard permet d'ajouter facilement des capteurs et autres actionneurs pour étendre les possibilités du robot, ou simplement des petites barres sur les côtés afin qu'il puisse attraper des objets au sol. Son look en bois et ses connexions apparentes et accessibles en font le robot idéal pour les bricoleurs ou les enseignants en technologie souhaitant l'adapter pour un projet scolaire. L'assemblage du kit nécessite seulement un tournevis et un peu de patience.
Présente dans plusieurs régions, l'association propose des formations et une aide technique dans différentes villes : Paris, Toulouse et Lyon.
Âge minimum pour le programmer : 8 ans
Âge recommandé : à partir de 10 ans.

CONCLUSION

Nous aurions aimé vous parler de bien d'autres choses encore telles que de la création de mods Minecraft, du monde merveilleux des Arduino, de l'initiation à la programmation sans écran avec le jeu de cartes The Bugs de Magik Square et bien plus encore mais pour cette fois-ci, toute bonne chose à une fin, et sera l'objet d'un autre article prochainement, avec en bonus track une revue de livres et des témoignages de programmeur.se.s en herbes. •

Vacances de Toussaint : un stage de **Kid Coding Robot** remporte un franc succès

• Olivier Pavie

C'est à Saint Raphaël, dans le Var, que l'espace de Coworking Point Sud a lancé il y a quelques mois une initiative de demi-journées de Kid Coding. Après le succès remporté, l'idée a été de proposer chaque après-midi pendant une semaine, un stage pour apprendre à programmer un robot en le faisant évoluer tous les jours. Des enfants heureux qui sont repartis avec leur robot sous le bras.

Louis Casa est l'instigateur et le dirigeant de l'espace de Coworking de Saint Raphaël, un homme de communication et d'évènementiel qui aime surfer sur tout ce qui touche aux nouvelles technologies. Julien Devincre est un des coworkers, dirigeant de sa société de services Technolan, informaticien, jeune papa, qui a su répondre à l'appel du pied de Louis pour se pencher sur cette proposition d'activité ouverte à tous les enfants de 8 ans à 15 ans. Certes, chaque enfant doit venir avec un ordinateur personnel pour pouvoir participer, et le stage a aussi un coût : 180 €, les enfants repartant avec leur robot.

Un important travail de préparation

« Préparer un stage de 5 après midi de Kid Coding est un véritable travail » nous confie Julien. « Mais la satisfaction du résultat est à la hauteur de l'investissement, les enfants ont passé une super semaine et il était étonnant de voir à quel point ils se concentraient ». Comment s'est préparé ce stage ? Julien a parcouru les sites, s'est documenté, le but était aussi de trouver un robot qui ne soit pas trop cher et soit capable d'exploiter des capteurs infrarouge, des capteurs ultrason, qu'il puisse disposer du WiFi pour arriver à être autonome. Mais l'essentiel étant aussi qu'il puisse être évolutif, à la fois dans les options qu'il pouvait embarquer mais aussi dans les modes de programmation qui pouvaient être exploités. Le mBot de Makeblock s'est imposé pour son rapport qualité/prix, sa possibilité d'être programmé en Scratch mais aussi sa carte électronique de type Arduino avec toutes les possibilités de connectiques déjà intégrées sur la carte en natif.

Se fixer des objectifs et essayer de les atteindre

Dans la préparation du Stage, Julien Devincre s'est aussi et bien évidemment penché sur le contenu éducatif. Comment arriver à motiver huit enfants et ne pas en laisser en route. « Avec les enfants de 8 à 10 ans, c'était parfois compliqué en matière de concentration, les pré-ados étaient vraiment toujours en réflexion et dans l'action. On a commencé la première demi-journée à détailler les pièces du robot et à les assembler : les enfants avaient hâte de passer à la programmation ! Dès le second jour on s'est attaqués à la programmation : le but, comprendre leur fonctionnement et commencer à initier un comportement en suivant une ligne noire tracée sur le sol ». L'imagination de Julien Devincre et de Louis Casa a permis de préparer un genre de Robot Park dans lequel tous les robots pouvaient se retrouver et s'essayer à mettre en pratique les premiers éléments permettant de déplacer le robot en utilisant le repère de la ligne noire. L'avantage du mBot, c'est qu'il peut travailler dans un mode où il exécute les commandes en même temps qu'elles sont entrées dans la machine, sans besoin de télécharger le programme dans l'Arduino ; « évidemment, c'est lents, ce n'est pas du vrai temps réel, mais pour le débogage et l'analyse du comportement des capteurs, c'est un mode de fonctionnement très pratique et très ludique ; ce n'est qu'une fois que le programme est téléchargé dans l'Arduino que le robot exécute rapidement l'algorithme » précise Julien.

1er jour, les enfants découvrent le robot et apprennent à le monter

Les enfants ont tous eu un "diplôme de fin de stage"

Les robots s'affutent à tâcher de suivre la ligne

Du côté des enfants

Victor est un des enfants qui ont fortement apprécié le stage. Il est prêt à revenir et à apprendre d'autres choses. Il ne connaissait rien à la programmation des robots et il avoue que maintenant il s'amuse à tester de nouvelles possibilités avec les capteurs, des finesses dans la programmation. « Ce qu'il y avait de réellement passionnant c'est la manière dont on a réfléchi au comportement du robot qui devait suivre cette ligne noire et qui après devait trouver un moyen de contourner un obstacle tout en retrouvant la ligne noire qui lui donne son cheminement. »

UNE EXPÉRIENCE À RENOUVELER

Que ce soit pour les organisateurs ou pour les enfants dans leur majorité, l'expérience est réussie. Plusieurs enfants pensent déjà à ce qu'ils vont essayer avec leur robot, voire attendent que les organisateurs proposent des demi-journées de kid coding robot en attendant peut-être un nouveau stage de programmation encore plus avancée. •