ACTIVITÉ Le ROBOT IDIOT

Objectifs :

- Découvrir les notions d'informatique à l'école : programme, langage de programmation, bug
- Savoir exécuter un programme composé d'instructions simples
- Savoir écrire un programme à partir de ces mêmes instructions en choisissant un langage de programmation

Pré-requis : Aucun Durée : 1h30

Modalités :

L'activité du robot idiot est une activité très documentée, les objectifs varient en fonction des auteurs. Le choix fait dans cette activité est de ne pas parler d'algorithme mais seulement de programme et langage de programmation.

Matériel :

- Tapis de jeu (draps peints, fichier modèle : « terrain_videA3 » ou « terrainA3cases ») ;
- Affiches pour les azimuts (fichier "azimuts") ;
- Objets à imprimer et étiquettes de programmation (fichier « cartes ») ;
- Fiches programmes (fichiers : « programmes », et pour la phase optionnelle 8bis "programmes_evenements")
- Par binôme d'élèves : plateau de jeu format A3 (fichier « terrain_videA3 » ou « terrainA3cases » ou « terrain_defis_objets »), fiche exemple de défis (dans le fichier « programmes») , fiche pour écrire un programme (fichier : "grille"), 1 playmobil et/ou un pion de jeu de dames
- Si disponible, un visualiseur pour projeter au tableau les programmes proposés par les élèves

Niveaux : cycle 1 (sauf phases 9 et 10), cycle 2, cycle 3.

On adaptera le vocabulaire, le questionnement proposé au niveau des élèves.

Remarque : les élèves présentant des troubles de la latéralité pourront bénéficier d'étayage spécifique (ex.: un point bleu sur la main gauche et un point rouge sur la main droite ou des bracelets de couleur et symboles imprimés en couleur sur les fiches de programme).

Synthèse du vocabulaire employé avec les élèves durant ces activités :

- une instruction (élémentaire) dicte l'action à réaliser par le robot idiot (déplacement pour passer d'un nœud (point) à un autre ou d'une case à une autre du tapis ; ramasser ; etc.).
- l'ensemble des instructions élémentaires va former un programme.
- langage de programmation : un ensemble de mots/codes qui permettent de construire des instructions à dicter au robot idiot.
- déplacement absolu : l'effet des instructions ne dépend pas de l'orientation du « mobile » qui les reçoit.
- déplacement relatif : l'effet des instructions dépend de l'orientation initiale du « mobile » qui les reçoit.
- bug d'un programme

Durée	Phase	Activités et consignes
10 à 20 min	**Phase 1 : Contextualisation :** **mise en route de l'activité « robot-idiot » et nécessité d'un langage commun**	Matériel support : **- Un tapis de jeu sur drap représentant le terrain** (fichier : terrain_videA3 ou terrainA3cases) Objectif : - se mettre d'accord sur un langage commun pour dicter des instructions au robot idiot - expliciter le codage utilisé (mots, dessins Nuage Soleil Montagne Mer (NSEO), flèches...) Modalité d'organisation pédagogique : binôme (cycle 2-3) et collectif Problème : Un point de départ étant donné sur le tapis de jeu, il faut créer un programme qui permette de rejoindre le point d'arrivée, en utilisant une série d'instructions (élémentaires). *Exemple : on peut dire « faire un pas vers .. », « faire un pas à droite » ; »* faire le tour du bosquet » n'est pas une instruction élémentaire. Déroulé : On propose aux élèves d'écrire (sur papier blanc) ou dicter un premier itinéraire « librement » et comparer le langage utilisé, en faisant valider le programme par un robot idiot sur le tapis de jeu. Par écrit, on pourra écrire le programme par deux, puis faire lire son programme par un autre binôme. Mise en commun : Le professeur notera sur un affichage collectif les mots utilisés. Si un langage relatif a été utilisé, il faudra souligner l'importance dans ce cas de l'orientation initiale du robot avant de réaliser l'instruction.
10 min	**Phase 2 : Découverte d'un langage de programmation de déplacements en mode absolu et appropriation des règles de ce langage.**	Matériel : - Affiches A3 pour représenter les azimuts (fichier « azimuts ») Objectifs : - S'approprier les instructions élémentaires - S'approprier les règles - Vérifier que tous les élèves ont compris les instructions ou mots du langage. Modalité d'organisation pédagogique : collectif Déroulé :

		- Présenter le langage à partir de ses instructions (4 dessins représentant les 4 azimuts). - Donner la règle du jeu : le processeur (l'animateur) lit le programme et les robots (les élèves) effectuent les actions correspondantes. Si le robot tombe à l'eau, s'il sort du drap ou s'il entre dans la forêt, alors il est éliminé. - Tous les élèves se mettent sur un drap, chacun sur un point. - Le professeur propose des instructions « Nuage », « Nuage », « Soleil », etc. et les élèves se déplacent tous en même temps. Ils sont éliminés au fur et à mesure si ils sortent du drap, entrent dans la forêt ou tombent à l'eau. On assiste à une « chorégraphie » de robots idiots. <u>Mise en commun :</u> noter le langage de programmation choisi (dessins des azimuts) ; insister sur le fait que le mot « avancer vers » est implicite dans l'instruction « nuage ».
15 min	**Phase 3 :** **Lire et exécuter un programme**	<u>Matériel :</u> - Un tapis de jeu sur drap représentant le terrain (fichier : terrain_videA3 ou terrainA3cases) - Les programmes (fichier « programmes » début 1ère page) <u>Objectifs :</u> - Identifier deux rôles dans la programmation : le processeur qui lit le programme ; le robot qui effectue les actions qui lui sont dictées. - Lire et exécuter un programme. - Exprimer en français ce que fait le programme **Choix du vocabulaire à expliquer** : **« L'ordinateur lit le programme et le robot effectue les actions correspondantes. »** <u>Modalités d'organisation pédagogique :</u> Deux choix possibles : – Par groupe (classe ou ½ classe), un processeur et un robot, les autres observent – Par trinôme avec un processeur, un robot et un observateur À chaque nouveau programme changer de processeur et de robot. Mise en commun collective. <u>Déroulé :</u> - Consigne pour les élèves : « pour chaque programme distribué, on va demander à un duo de venir sur le drap. L'un des élèves sera le processeur et lira le programme. L'autre sera le robot idiot et effectuera les

		actions correspondant aux instructions lues par le processeur ». - Les autres élèves sont autour du drap, vérifient que le robot suit bien les instructions, puis ils doivent expliquer en français à l'oral ou à l'écrit quel est l'objectif du programme (bien faire formuler aux élèves pour que dans la deuxième partie – travail en binôme ou en trinôme – ils comprennent ce qui est demandé). - Selon le nombre d'enfants et le niveau on peut tester tous les programmes ou seulement quelques-uns. *Programme 0 :* Le robot arrive entre le bosquet et le lac. Ne comporte pas d'erreur. *Programme 1 :* Le robot arrive sur le pont de gauche en passant par le pont de droite. Ne comporte pas d'erreur. On place des objets sur le terrain de jeu (voir fichier « terrain_defis_objets ») et on introduit une nouvelle instruction : **Main** pour « ramasser un objet aux pieds du robot ». *Programme 2 :* Le robot ramasse 2 objets au nord et au sud de l'arbre. Ne comporte pas d'erreur. Mise en commun : retour sur les affichages de début de séance ; le vocabulaire utilisé (programme, instruction) ; les implicites (”nuage” au lieu de “avance d'un pas vers le nuage”).
20 min	**Phase 4 : Écriture de programmes par les élèves**	Matériel : **- selon le cycle,** un tapis de jeu représentant le terrain ou un plateau de jeu plastifié (fichier « terrain_defis_objets ») - Un pion de jeu de dame par groupe (tant qu'on travaille en déplacements absolus, il est souhaitable que les pions à déplacer ne soient pas orientés, on évite donc les Playmobil par exemple) - Des défis (fichier : « programmes » , page 1) - Des fiches pour écrire un programme (fichier “grille”) Objectifs : – Identifier un troisième rôle dans la programmation : le programmeur qui écrit le programme ; – Écrire un programme en respectant un langage de programmation. – Exécuter un programme pas à pas. **Choix du vocabulaire à expliquer** : « **Le programmeur écrit le programme, l'ordinateur lit le programme et le robot effectue les actions correspondantes. »** Modalités d'organisation pédagogique: - Par groupe de 3 ou 4 - Le groupe d'élèves peut utiliser des cartes de programmation.

		Ces cartes peuvent être scratchées sur un support pour que le programme ne bouge pas même quand on se déplace ; elles peuvent aussi être disposées le long du tapis de jeu ou du plateau de jeu plastifié, de gauche à droite, ou bien encore le long du trajet, au moins pour la mise au point du programme. Écueils possibles du dernier exemple : redondance du codage si la carte est placée sur le tapis de jeu dans la direction à suivre ; risque de modifier le programme lorsqu'on ramasse les cartes. – Projeter les programmes écrits au tableau à l'aide d'un visualiseur, ou les recopier. Déroulé : - Choisir un des défis : e*xemples de défis :* *Emmener le robot au milieu du plan d'eau.* *Faire faire le tour du bosquet au robot et terminer sur le pont de gauche.* *Ramasser des objets selon un certain itinéraire* - Écrire le programme correspondant **dans le langage de programmation**. - Tester le programme avec un pion de dames sur le terrain de jeu. - Échanger les fiches avec un groupe voisin pour le faire valider. - Exécuter le programme du groupe voisin et écrire en français au dos de leur fiche l'objectif du programme qu'ils ont proposé. - Pour les plus groupes les plus avancés, inventer un programme (et le faire tester par le groupe voisin) Mise en commun : comparer plusieurs programmes écrits pour le même défi ; revenir sur l'importance de respecter les instructions élémentaires et la syntaxe pour le langage de programmation.
10 min	**Phase 5 : Rendre le langage plus expressif : proposer une nouvelle écriture**	Matériel et organisation pédagogique identique à la phase 4. - En complément, on distribuera les cartes comportant les instructions « x2 », « x3 », etc. Objectifs : - insister sur l'importance de la syntaxe dans un langage de programmation Déroulé : - lire et exécuter le *Programme 3*. - réécrire le programme 1 de la même manière que le programme 3. Mise en commun :

		En comparant les deux écritures du programme 1, on fait apparaître les éléments de conclusion suivants : - on s'aperçoit qu'on répète plusieurs fois la même instruction ; le langage est suffisant mais peu expressif, on le modifie pour permettre d'exprimer les répétitions; - l'écriture **direction x5** est plus lisible que 5 dessins identiques à la suite. Présenter la nouvelle écriture aux élèves. Il faut spécifier comment ces répétitions s'écrivent : une instruction *composée* peut être formée de 2 éléments : le déplacement à répéter (instruction *élémentaire*) et le nombre de répétitions. On s'aperçoit qu'un programme n'est pas qu'une simple suite d'instructions mais qu'il doit respecter une véritable grammaire. Attirer l'attention des élèves sur la difficulté de cette nouvelle façon d'écrire et la syntaxe à respecter : **direction x5** et non **5x direction (*).** Nous utilisons un **langage de programmation** qui, tout comme la langue française, a des mots ou **instructions élémentaires** et une grammaire ou **syntaxe** qu'il faut respecter pour se faire comprendre (on fait le parallèle avec les erreurs grammaticales comme « verbe verbe »). (*) Ce choix est lié au langage de programmation Bluebot, utilisé dans la continuité de cette activité.
15 min	**Phase 6 :** **Découvrir la notion de bug**	Matériel et organisation pédagogique identique à la phase 5 (fichier "programmes" page 2) Objectifs : - trouver le bug d'un programme. - définir la notion de bug dans un programme : un bug est un problème dans un programme. Déroulé : - lire *le Programme 4* et trouver son bug. **Ne pas prévenir les élèves qu'il y a un bug dans ce programme !** Ils doivent le découvrir par eux-mêmes. *Programme 4* : le robot finit dans le lac. - Puis réécrire le programme pour corriger ce bug. Pour corriger il faut savoir quelle était la tâche à accomplir (ici, arriver sur la presqu'île). Le professeur peut indiquer le but du programme. Correction : *programme 4bis*. - Mise en commun : s'interroger si la « faute » revient au robot, au lecteur du programme (processeur) ou

		au programmeur. Dans le cas présent, la syntaxe est respectée mais l'exécution mène à **un plantage logiciel ou matériel** (robot dans un obstacle). Le programme a **un bug**. *<u>Remarques sur la correction des erreurs :</u>* En langue française, si la syntaxe est incorrecte, on arrive à comprendre éventuellement le sens : nous idiot robot jouer... on peut deviner ce que la personne a voulu dire. Pour un langage de programmation, l'ordinateur qui lit le programme ne peut pas deviner. S'il y a une erreur de syntaxe dans le programme alors l'ordinateur ne peut rien faire. Lorsque le programme est correctement écrit mais produit un résultat indésirable (comme ici "faire tomber le robot à l'eau"), il devient très difficile de la corriger sans avoir plus d'informations, que ce soit pour un humain ou a fortiori pour une machine. Précisons que l'erreur ne vient pas de l'ordinateur mais du programmeur qui a écrit le programme. - lire le *Programme 5* et trouver son bug. Puis réécrire le programme pour corriger ce bug. - <u>Mise en commun :</u> *Programme 5 :* Le robot cherche à ramasser deux fleurs mais rate la seconde, il la dépasse d'un pas et ramasse "dans le vide". Ce programme comporte une erreur : il ne plante pas mais **il ne fait pas la tâche demandée**. Cette fois-ci il n'y a pas d'erreur de syntaxe ni de plantage. Le robot a bien exécuté le programme jusqu'au bout mais le programme n'a pas fait ce qu'on voulait. Là encore c'est le programmeur qui a fait une erreur. Comment peut-on corriger cette erreur ? Réponse : en échangeant les instructions "nuage" et "ramasser" à la fin du programme. Deux types de **bugs** ont donc été vus : - soit le programme plante (il met la machine dans un état d'erreur, voire il dégrade le matériel) - soit le programme réalise toute la suite d'instructions mais il ne résout pas le problème, n'atteint pas l'objectif recherché.
10 min	**Phase 7 : Bilan sur les notions et le langage**	Contrairement à un humain, un ordinateur ou une **machine** ne prend pas d'initiative et ne fait qu'**exécuter des programmes**. Un programme est une suite d'**instructions** précises et non ambiguës.

		Pour communiquer entre eux les humains utilisent par exemple le français. Cette langue est constituée de mots et de règles – la **grammaire** – pour fabriquer des phrases. Cette langue est riche, elle permet d'exprimer des sentiments, elle est subjective. Pour communiquer avec une machine on utilise un **langage de programmation**. Ce langage est constitué de mots, les **instructions élémentaires**. Ces mots sont moins nombreux que pour la langue française. Pour écrire un programme à partir de ces mots il faut respecter une grammaire : la **syntaxe** du programme. On peut distinguer 3 grands types d'erreurs de programmation : • le programme n'est pas compréhensible par la machine (**erreur de syntaxe**) ; • le programme conduit à un **plantage matériel** (robot qui fonce dans un obstacle...) ; • le programme **ne fait pas ce qu'on voudrait** qu'il fasse (itinéraire qui n'amène pas à la destination voulue). Dans les 2 derniers cas on a l'habitude de parler de "**bug**" mais il faut avoir conscience qu'il est presque toujours d'origine humaine : l'erreur vient du **programmeur**. Il est donc très important de **vérifier** les programmes. Il y a plusieurs méthodes pour vérifier un programme. Une méthode peut être de vérifier le programme pas à pas (dès que les programmes sont plus compliqués cette méthode s'avère rapidement insuffisante). Il existe des programmes permettant, dans une certaine mesure, de tester d'autres programmes ; cependant ces derniers laisseront toujours passer des erreurs sans les détecter. Une trace écrite peut être préparée collectivement sur le vocabulaire : programme, programmer, instruction, bug, langage de programmation, langue française, mot, ordinateur. (on pourra s'inspirer de la fiche support « lexique programmation cycle 1,2,3 » du Groupe Sciences Isère, en lien en fin de document)
15 min	**Phase 8 : Prolongement possible ou variantes proposées** **Changement de langage de programmation : déplacements relatifs**	En amont de cette phase, on pourra proposer une séance autour des itinéraires, de l'orientation dans la vie courante et mettre en avant le vocabulaire employé (tourner à droite, gauche, aller devant, derrière, au nord, au sud, etc…..). <u>Matériel :</u> – on peut revenir sur l'affichage collectif de la phase 1 – cartes de programmation en déplacements relatifs (fichier "cartes") – playmobil pour symboliser le robot avec une orientation – fichier "programmes" page 2

		Déroulé : - On enlève les azimuts : quel langage peut-on maintenant utiliser ? - Réécrire le programme d'un des défis (phase 4). Mise en commun : on adopte un langage différent pour exprimer nos programmes : à choisir parmi avancer, reculer, pivoter à gauche ou à droite, faire un pas de côté à gauche ou à droite. - Lire et exécuter le P*rogramme 6*. Le programme 6 a le même effet que le Programme 2 : le robot ramasse 2 objets au nord et au sud de l'arbre. Mise en commun On choisit ici le (un) jeu d'instructions compatible avec ce qu'un robot Blue-bot sera capable de faire par la suite : « avancer, reculer, pivoter à gauche ou à droite » - Écrire des programmes soi-même : les mêmes itinéraires que les défis de la phase 4. Mise en commun : on montre qu'on peut exprimer la même solution dans différents langages.
20 min	**Phase 9 (cycles 2 et 3 uniquement)** **Robot idiot événementiel ou comment permettre au robot de s'adapter à son environnement** **Un autre langage de programmation significativement plus expressif que les précédents**	Matériel : – idem phase 4 **+ cartes de programmation "quand..."** + carte « se déplacer dans une direction choisie au hasard» (fichier "cartes") – programmes et défis avec événements (fichier "programmes_evenements") Objectif : - découvrir la programmation événementielle - résoudre des défis simples à l'aide de scripts déclenchés par des événements Problème : On souhaite que notre robot adapte son comportement à différentes situations, par exemple ramasser toutes les fleurs le long d'un chemin donné, mais sans savoir à l'avance où sont situées les fleurs. *Remarque : la solution « avancer ; ramasser la fleur s'il y en a une ; avancer ; ramasser la fleur s'il y en a une ; etc. »* n'est pas satisfaisante (lourdeur, duplication de code ; et le chemin n'est plus lisible clairement). Modalités d'organisation pédagogique : par groupes de 3 puis 4, 5 selon le nombre de capteurs Déroulé : - On explique aux élèves les nouveaux rôles que l'on fait essayer sous forme de jeu de rôles sur le programme de la *situation 1* (fichier "programmes_evenements"). - On "sépare" le programme en 2 parties : la partie "itinéraire" et la partie "réaction à l'environnement". - Comme auparavant, deux élèves jouent toujours les rôles du processeur (qui lit l'itinéraire) et du robot

		(qui effectue les actions correspondantes). - Un nouveau rôle apparaît : un “capteur” qui observe si le robot vérifie une certaine condition (”passer sur une fleur” par exemple) et qui dispose d’un court programme (un **script**) à exécuter dans cette situation. - Une nouvelle règle pour le processeur : quand un capteur se manifeste, il doit lui céder la parole le temps que celui-ci exécute son script, puis reprendre là où il en était. - Ce partage du temps de parole peut être matérialisé par un objet (stylo, bâton, figurine) que se transmettent les participants afin de savoir qui dicte ses instructions au robot à tout instant. - Par la suite il y aura plusieurs “capteurs” dans un même groupe et pour un même robot ; le rôle du processeur, à l’inverse, sera amené à disparaître totalement. - On propose ensuite de résoudre différents problèmes à l’aide de la programmation événementielle. Par exemple : *Situation 2 :* le robot doit faire un tour sur lui-même lorsqu'il marche sur un trèfle. Le robot doit effectuer un pas de danse (un pas à gauche et un pas à droite) lorsqu'il marche sur une plume. *Situation 3* : le robot doit ramasser tous les objets du terrain, comme un aspirateur automatique. Pour cette dernière tâche, on autorise une nouvelle instruction qui signifie “se déplacer dans une direction choisie au hasard”. <u>Mise en commun :</u> On peut voir émerger au moins deux points de vigilance qui se posent lors des interruptions : – Le contrôleur du programme principal interrompu ne doit pas oublier où il en était (point de contrôle), par exemple en gardant son doigt sur la prochaine instruction qui lui reste à exécuter, jusqu’à ce qu’on lui rende la main. – Les scripts secondaires doivent être écrits de façon à repositionner le robot dans un état comparable à celui qu'il avait avant l'interruption (par exemple, sur le chemin qu'il suivait lorsqu’ils ont fait un écart).
25 min	**Phase 10 (cycles 2 et 3 uniquement) : programmer des déplacements à l’aide de scripts déclenchés par des événements**	<u>Matériel</u> : idem phase 9 + carte « le robot détecte un obstacle » (fichier « cartes ») <u>Objectif :</u> - programmer intégralement avec des scripts déclenchés par des événements (préparation à l’utilisation du robot Thymio). Plus on avance dans les défis de la phase 9, plus le risque augmente que le robot rencontre un obstacle en suivant un script. On introduit alors un capteur “le robot détecte un obstacle”. <u>Déroulé :</u>

		- Présenter les capteurs d'obstacles ( ). - *Situation 4 :* Adapter le programme de l'aspirateur (situation 3) pour lui éviter de foncer dans un obstacle. Proposer une écriture de ce nouveau programme. Mise en commun : à partir de 2 ou 3 programmes choisis dans la classe, on demandera de traduire chaque programme à l'oral en langue française. Insister sur la tâche demandée ; différencier l'action principale des actions liées aux capteurs. Pour les groupes avancés, on peut proposer un autre défi (qui peut utiliser des capteurs d'obstacles sur les côtés également) : - *Situation 5 :* avancer indéfiniment ; quand je vois un obstacle devant moi, je tourne à gauche. Écrire ce programme . L'itinéraire s'adapte complètement au terrain, mais le résultat n'est pas très satisfaisant : le robot finit rapidement par "tourner en rond". - *Situation 6 :* On suppose le robot placé initialement avec un obstacle à sa droite ; écrire une série de scripts permettant au robot de suivre le mur en continu. Le professeur peut demander d'écrire en premier lieu les instructions en langue française, puis de traduire ces instructions dans le langage de programmation commun. Mise en commun : à partir de 2 ou 3 programmes choisis dans la classe, on comparera les solutions en langue française, puis en langage de programmation.

Activités inspirées d'un travail de Marie Duflot-Kremer.
Ont participé à la rédaction de ce document : Maryline Althuser, Nathalie Brasset, Nathalie Penin, Anne Rasse, Jean-Marc Vincent, Benjamin Wack.

Sources et vidéos disponibles :
Page médiation scientifique de Marie Duflot : https://members.loria.fr/MDuflot/files/med/robot.html

Matériel à imprimer et lexique sur la programmation :
https://sciences-technologie-38.web.ac-grenoble.fr/article/stage-objets-techniques-cycle-123

Pour les sources qui suivent des objectifs qui ne sont pas les mêmes que ceux de l'activité que nous proposons :
https://pixees.fr/dis-maman-ou-papa-cest-quoi-un-algorithme-dans-ce-monde-numerique-%E2%80%A8/
http://images.math.cnrs.fr/Dis-maman-ou-papa-c-est-quoi-un-algorithme-dans-ce-monde-numerique

LEC Robot : http://www.ac-grenoble.fr/maths/ ressource n°593