Robotique virtuelle :
Environnement pour un apprentissage dynamique et interactif de l'algorithmique

Rupert Meurice de Dormale

CeFIS NAMUR (Belgique)

Table des matières

Résumé

Cette contribution décrit une concrétisation de la réflexion menée par Charles Duchâteau à propos de l'apprentissage de l'informatique dans « *Robotique- Informatique mêmes ébats, mêmes débats, mêmes combats ?* » in Actes du 4ème colloque international sur la Robotique pédagogique (Duchâteau , 1993). Une (re)lecture de ce texte permettra de comprendre la pertinence du parallèle effectué entre les savoir-faire développés par la robotique pédagogique et l'informatique et de constater l'intérêt de « *consacrer un certain temps à la programmation de "robots" (malheureusement) imaginaires ou abstraits* »dans l'apprentissage de l'Informatique.

Y a-t-il un quelconque avantage à ce que ces robots "*imaginaires ou abstraits"* deviennent virtuels ? C'est ce que va essayer de montrer cet article.

Après avoir rappelé quelques notions importantes, nous nous intéresserons a une méthode "traditionnelle" d'apprentissage de l'Informatique pratiquée durant une dizaine d'années avec des élèves en fin d'enseignement secondaire belge (16-18 ans). Nous examinerons ses faiblesses et dresserons un "cahier des charges" de ce vers quoi devrait tendre une méthode "améliorée". Nous verrons pourquoi ce cahier des charges nous a mené à la robotique virtuelle, son intérêt et la façon dont un environnement d'apprentissage a été implémenté. Enfin, après avoir confronté cette implémentation au cahier des charges préalablement dressé, nous examinerons son impact sur les élèves et les conclusions que nous pouvons en tirer.

Mots-clé : Algorithmique, Environnement d'apprentissage, Robotique virtuelle

Cet article présente une nouvelle méthode destinée à faciliter l'apprentissage de l'algorithmique et de la programmation.

Il nous est apparu normal de nous poser, au préalable, la question de savoir s'il était encore utile aujourd'hui d'apprendre à programmer, alors même que la publicité nous présente les ordinateurs comme des "outils" dont l'utilisation semble tellement simple. Le lecteur intéressé par l'approfondissement de cette question pourra se référer entre autres à Romainville, (1988), Duchâteau, (1990 et 1994).

Pour nous, l'algorithmique et l'informatique favorisent notamment l'acquisition d'une méthode de travail et de réflexion, le développement des facultés d'analyse, d'abstraction, d'anticipation et de logique. C'est dans ce but que nous introduisons ces disciplines pour nos élèves de 16-18 ans. Il est primordial de garder à l'esprit que notre objectif est d'apporter autant que possible à **tous nos élèves** ces capacités, et non de sélectionner les meilleurs afin de les orienter vers des études supérieures en informatique.

Programmation. algorithmes et marches à suivre, quelques rappels

Ce n'est généralement pas un problème de **faire** quelque chose, mais cela devient vite un problème de **faire faire** quelque chose par *un autre ! « Il s'agit, non d'être capable de mener à bien soi-même une tâche donnée, mais d'expliquer à* ***un autre*** *comment il doit s'y prendre pour exécuter cette tâche à notre place. En quelque sorte, programmer, c'est* ***faire faire****. Programmer, c'est donc rédiger une* ***marche à suivre*** *(un algorithme)* ***pour faire faire une tâche par un exécutant****..."* (Duchâteau, 1990)

"Programmation" et "algorithmique" sont pour nous synonymes. "Programme", "marche à suivre" et "algorithme" sont aussi des synonymes qui désignent un "texte" destiné à faire réaliser une tâche par un exécutant.

Les composants essentiels du "texte" d'une marche à suivre, programme ou algorithme sont :

- Une succession d'**instructions d'actions élémentaires**
- Un certain nombre de **structures de contrôle**

- *L'alternative* ou *conditionnelle*
- La *répétitive* ou *itérative*
- Le *branchement*
- L'appel de *procédure*
- La *séquence*.

La succession des instructions dans le texte (statique et figé) de la marche à suivre est évidente. Toutefois, lors de son exécution dynamique, les actions commandées par les instructions seront réalisées dans un ordre changeant et dépendant des conditions initiales. C'est ici que, pour nous, réside la difficulté essentielle de la programmation. Le rédacteur de l'algorithme doit se projeter dans un univers dynamique (anticipation, abstraction), imaginer toutes les possibilités de fonctionnement, prévoir les tests pour déterminer les conditions dans lesquelles ces fonctionnements auront lieu (analyse) et faire en sorte que l'action adéquate se fasse toujours au bon moment (logique).

Tournons-nous maintenant vers l'exécutant. Pour pouvoir le piloter, il faut que le programmeur en ait une connaissance précise. « *Que doit comporter la description d'un exécutant pour que sa programmation soit possible ?*

- *la description :*
  - *de l'exécutant*
  - *de son environnement de travail*
  - *de la tâche à effectuer*
- *la liste des actions dont l'exécutant est capable et des instructions correspondantes*
- *la liste des conditions que l'exécutant est capable de tester.* » (Duchâteau, 1993)

Programmer c'est donc concevoir et exprimer une marche à suivre, un programme ou algorithme en vue de faire réaliser, en différé, une tâche par un exécutant dont les capacités (limitées) sont parfaitement connues du programmeur.

L'exécutant peut être n'importe quel automate ou robot. Il peut être aussi « l'automate-ordinateur », qui ne se distingue pas particulièrement des autres exécutants. Le tout est d'en avoir une bonne description, de connaître les actions dont il est capable et la liste des conditions qu'il peut tester. Nous renvoyons le lecteur à Ch. Duchâteau (Duchâteau, 1993) pour une présentation métaphorique de l'exécutant-ordinateur.

Une méthode traditionnelle d'apprentissage de la programmation et ses limites

Nous estimons qu'il existe trois difficultés majeures à maîtriser pour accéder à la programmation d'un ordinateur

1. la maîtrise de l'utilisation des structures de contrôle (algorithmique)
2. la maîtrise des principes de fonctionnement des variables (affectation)
3. la fusion des deux points précédents et une bonne représentation métaphorique de l'exécutant afin de pouvoir programmer cet automate-ordinateur.

Voici la façon dont les différents points de matière étaient abordés et, pour chacun d'eux, la critique que nous pouvons en faire aujourd'hui.

Apprentissage de l'utilisation des structures de contrôle

Le cours commençait par l'explication au tableau noir du fonctionnement des trois structures de contrôle de base (séquence, alternative et répétitives). Un parallèle était fait entre ce qui était écrit dans le texte statique de la marche à suivre et ce qui pouvait se produire lors de son exécution dynamique.

Un très petit nombre d'exercices sur papier et au tableau devaient permettre aux élèves d'acquérir la compréhension de ces structures. Généralement, aucun élève ne signalait un quelconque problème à ce niveau. Et pourtant, il s'avérait que les difficultés apparaissant par la suite trouvaient leurs racines dans une mauvaise maîtrise de cette phase.

Critique : cette manière de procéder était trop abstraite. Seuls les élèves possédant déjà les facultés d'abstraction et de logique nécessaires pouvaient passer ce cap, mais pas les autres. C'est dans le but d'éviter que cette première étape soit déjà une sélection que nous avons développé le nouvel environnement d'apprentissage décrit au chapitre 4.

Apprentissage de la gestion des variables

La façon dont l'ordinateur gère les variables était un nouveau concept abstrait à maîtriser. Bien entendu, une analogie était réalisée. Les notions d'affectation et de lec- ture étaient vues, des exercices réalisés et, dans l'ensemble, tous les élèves arrivaient à maîtriser cette matière.

Critique : cette matière était souvent présentée comme un point de théorie et les exercices réalisés n'intégraient pas les connaissances algorithmiques vues précédemment. De plus, nous recommandons la plus grande vigilance quant à l'analogie utilisée.

Premiers programmes en pseudocode

Cette activité était la "fusion" des deux précédentes (ce qui nécessitait leur parfaite maîtrise). Toutefois, une difficulté supplémentaire se présentait l'automate-ordinateur n'était pas connu. Il fallait donc le présenter aux élèves et la bonne compréhension de son fonctionnement était un préalable de plus.

C'est à ce moment que les élèves exprimaient leurs premières difficultés. Environ un tiers d'entre eux comprenaient les programmes simples mais étaient totalement incapables de les refaire par eux-mêmes.

Critique : cette étape était trop dense. Elle nécessitait la maîtrise simultanée de trois savoirs et savoir-faire très différents. Une méthode plus progressive est envisagée dans l'environnement proposé.

La maîtrise de l'environnement de programmation

Pour pouvoir taper les programmes sur ordinateur, il fallait alors s'atteler à la syntaxe et à la manipulation de la machine (éditeur, compilateur, correction des (nombreuses) erreurs de syntaxe), sans compter les élèves qui frappaient à un ou deux doigts et auxquels il fallait une heure de cours pour taper un programme de 20 lignes.

Critique : tout ceci était néfaste et introduisait une trop grande cassure dans la progression de la matière. Tout y était présent pour distraire les élèves de l'algorithmique durant deux à trois semaines.

Enfin, la vérification sur ordinateur

L'élève pouvait enfin vérifier son programme sur ordinateur, voir s'il tournait correctement et, le cas échéant, comprendre ses erreurs. Or, dans cette phase, l'ordinateur ne lui apportait aucune explication. Seul l'enseignant pouvait expliquer (de façon abstraite) à l'élève son erreur et la façon de la corriger.

Critique : on rejoint ici un problème rencontré par de nombreux auteurs face à l'ordinateur ou aux systèmes "hermétiques" en général. Sougné propose « *l'affichage de l'évolution de l'état des entrées et sorties de l'interface alors que le programme s'exécute. Cette idée est à rapprocher de la lunette cognitive de Nonnon* » (Sougné, 1993). Hudon et Nonnon « *s'intéressent à l'utilisation d'un environnement pédagogique informatisé pour l'enseignement et l'apprentissage de concepts liés à des*

systèmes technologiques non visualisables. » Ils signalent que « *L'étude de systèmes aussi complexes et non visualisables ne peut se faire efficacement sans l'utilisation maximale d'une méthode qui incorporerait un système pour faciliter une appréhension plus visuelle et synthétique des interactions de variables.* " (Hudon et Nonnon, 1993).

C. Rellier et F. Sourdillat effectuent *"la mise au point d'expériences permettant à l'étudiant d'agir sur le système physique et d'en avoir une représentation graphique simultanée. Cette situation décrite par Nonnon sous le concept de lunette cognitive, doit aider l'étudiant à dans sa phase de conceptualisation du phénomène* » (Rellier et Sourdillat, 1993). Cervera et Nonnon portent leur « *attention sur les problèmes d'apprentissage des concepts et phénomènes qui, par nature, ne sont pas directement observables et qui comportent un certain degré de complexité. Ainsi, dans le but de rendre intelligibles les relations internes des variables qui régissent le déroulement des phénomènes, mais aussi de rendre compte des concepts qui sont mis à contribution, nous avons entrepris de modéliser les fonctions technologiques réalisées* » (Cervera et Nonnon, 1993). Duchâteau est également attentif à ce problème. « *L'environnement pour l'apprentissage de la programmation* ***Images pour programmer*** *comporte aussi un petit logiciel (START) qui permet de suivre, dans un environnement schématique correspondant à celui employé lors de la description de l'exécutant, le déroulement de l'exécution des premiers petits programmes conçus par les apprenants.* » (Duchâteau, 1993). START montre notamment l'évolution du contenu des différentes variables en parallèle à l'exécution du programme.

La manipulation de l'ordinateur n'est donc pas de nature à apporter l'aide nécessaire aux élèves en difficulté.

Conclusions

Nous nous étions fixé comme objectif de développer les capacités d'abstraction et de logique chez les élèves, et ceci par l'étude de la programmation. Or, l'analyse de la méthode montre qu'en permanence la faculté d'abstraction est sollicitée en tant que prérequis pour pouvoir maîtriser toutes les étapes du curriculum. La méthode est donc à revoir afin de permettre aux élèves ne possédant pas cette faculté d'avoir de réelles chances de faire leurs premiers pas dans l'étude de la programmation.

Cahier des charges pour l'amélioration de cette méthode « traditionnelle »

Après l'analyse effectuée au chapitre précédent, nous estimons que le problème crucial réside dans la maîtrise approfondie du maniement des structures de contrôle. De plus, il est nécessaire d'avoir une approche plus progressive et mieux intégrée de la notion de variable et de la métaphore de l'automate-ordinateur. Mais ceci est à placer sur un second plan.

La lecture de Barth (1987) nous a conduit à définir sept principes sur lesquels nous pourrions nous baser pour rendre le maniement des structures de contrôle accessible aux élèves ayant des difficultés d'abstraction.

Ne pas couper l'élève de ses ressources

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

Mettre l'élève devant une tâche clairement définie

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

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

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

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

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

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

La formation d'images mentales (visuelles ou auditives) aide l'élève à gérer l'abstrait. Il faudra donc éduquer et stimuler son imagerie mentale en lui présentant, dans sa phase d'apprentissage, une navigation simultanée dans le concret et l'abstrait. En robotique pédagogique, « *Grâce à la **réalité** des micro-robots on donne du sens aux programmes, l'exécution d'une action sur le micro-robot renvoie une image aux apprenants du programme qu'ils ont créé."* (Leroux, 1996) Oserions-nous dire que nous sommes en présence d'une" lunette cognitive inversée". Lorsqu'une machine (concrète) effectue des mesures, elle peut produire simultanément des graphiques portant une représentation conceptuelle (abstraite) des mesures effectuées. C'est le principe de lunettes cognitives de Nonnon. Dans le cas de l'exécution d'un programme en robotique pédagogique, c'est le programme (représentation abstraite de ce que devra être le comportement du robot) qui engendre les actions (concrètes) du robot. L'élève en présence d'un programme déjà réalisé naviguera simultanément dans le concret et l'abstrait en observant l'effet de ce programme sur le comportement du robot. A partir du moment où nous lui demandons de réaliser lui-même ce programme, nous entrons dans une phase nettement plus complexe au cours de laquelle il doit effectuer un grand nombre d'opérations, à savoir :

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

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

- ◊ présenter un univers clairement défini
- ◊ présenter une marche à suivre correcte **déjà réalisée**
- ◊ permettre son exécution par l'automate
- ◊ effectuer un parallèle entre *instruction exécutée* et *action effectuée*
- ◊ attirer l'attention sur les conditions testées et leurs résultats
- ◊ attirer l'attention sur **la ligne à laquelle va continuer le programme en fonction du résultat du test**, ce point étant pour nous le plus important.

Après la réalisation de cette phase et la vérification de sa maîtrise, la phase de conception des marches à suivre pourra commencer.

Faciliter la compréhension de son erreur par l'élève

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

Il faut une gradation correcte des difficultés

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

La robotique virtuelle : une solution et son implémentation

Les bienfaits de la robotique pédagogique ne sont plus à démontrer. Celle-ci convient tout à fait à la réalisation de séquences d'apprentissage répondant au « cahier des charges » réalisé ci-dessus.

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

- ◊ moindre coût
- ◊ moindre complexité de mise en œuvre
- ◊ grande possibilité de variation des univers abordés
- ◊ portabilité.

La virtualité pose cependant certaines questions. Sur quel pied faut-il placer cette virtualité? Il ne s'agit pas de la réalité, faut-il la classer pour autant d'office dans l'abstraction ou a-t-elle un statut intermédiaire puisqu'elle fait appel à une imagerie mentale déjà constituée et largement répandue (dessins animés, jeux vidéo,…). C'est "une réalité comme si on était à sa fenêtre" et qu'on voyait les choses se dérouler "de loin" . La vision du phénomène et son intégration sont les mêmes que pour un phénomène observé réellement, mais on ne peut pas toucher.

La virtualité ne nous paraît donc pas nécessiter l'un ou l'autre préalable que l'élève ne posséderait pas. À ce jour, aucun problème n'a été rencontré à ce niveau par les adolescents ou adultes ayant manipulé ce logiciel. La pratique de jeux électroniques se déroulant dans des univers virtuels est devenue banale.

La description de l'environnement développé montre qu'il a été tenu compte de tous les éléments énumérés au cahier des charges du chapitre 3.

Au niveau de l'ensemble du logiciel

Le logiciel aborde la matière selon un degré de complexité progressif et constant. La progression est la suivante :

1. alternatives, imbrication d'alternatives
2. opérateurs logiques (ET, OU, PAS)
3. répétitives, imbrication de répétitives
4. imbrication d'alternatives et de répétitives
5. les variables et les tours de main (Duchâteau 1990)
6. l'automate-ordinateur
7. les variables indicées
8. les procédures.

À chaque étape, avant que l'élève doive réaliser ses propres marches à suivre, quelques exemples d'algorithmes déjà rédigés sont présentés. Il se familiarise ainsi avec le nouveau niveau de difficulté abordé.

En outre, l'élève peut sauvegarder en l'état le travail auquel il est arrivé afin de pouvoir le retrouver et le compléter à la séance suivante. Il peut aussi, à sa meilleure convenance, imprimer tout ou partie de son travail.

En ce qui concerne chaque exercice

Chaque exercice est conçu sur le même canevas. Il est tenu compte d'un problème que plusieurs auteurs signalent, et notamment Leroux *"En dehors des difficultés de programmation attendues pour des néophytes en informatique, le principal blocage fut constaté au niveau de l'édition et de l'exécution de programmes. Pour faciliter la création de procédures, les usagers de LOGO disposent d'un éditeur pleine page. Le passage de la partie édition à la zone de commandes se fait normalement aisément. Certains stagiaires n'ont jamais pu assimiler cette différence. Ils confondaient sans cesse les deux zones, ils exécutaient des commandes dans l'éditeur et vice et versa. En fait cette transition cognitive entre deux mondes conceptuellement différents leur était inaccessible ils étaient perdus.* » (Leroux, 1996). Afin de réduire au maximum cette confusion, chaque exercice est séparé en deux pages très distinctes. La page d'édition (figure 1 en fin d'article) sans commande d'exécution à l'exception d'un *bouton* permettant d'accéder à la page d'exécution (figure 2 idem). Sur celle-ci ne figure aucune commande d'édition à l'exception du bouton *Correction* qui permet de revenir à la page d'édition. Nous constatons que la distinction entre ces deux niveaux d'intervention se crée rapidement.

Caractéristiques de la page d'édition (figure 1)

Sans entrer dans une description exhaustive, cette page comprend les éléments suivants :

◊ Un bouton pour accéder à l'énoncé. Celui-ci contient le détail précis de l'environnement de l'automate, des instructions correspondant aux actions qu'il peut réaliser, des conditions qu'il comprend et de la tâche exacte à réaliser

◊ Un bouton pour accéder à l'aide, soit « technique » (utilisation du logiciel), soit à une aide à la conception (voir ci-dessous)

◊ Un bouton de mise en page (indentation des structures de contrôle)

◊ Un bouton d'effacement du programme déjà réalisé (redémarrage à zéro)

◊ Un bouton pour passer à la phase d'exécution

◊ Un bouton de retour au menu

◊ les instructions d'actions élémentaires, conditions, opérateurs logiques et structures de contrôle nécessaires à la réalisation du programme

◊ une interface permettant à l'élève de réaliser rapidement et facilement le texte de son programme (système de glisser-poser, chaque élément du programme étant un module déplaçable). Les problèmes de syntaxe qui alourdissent le travail sont ainsi supprimés et l'élève peut focaliser son attention sur les points essentiels de son apprentissage

◊ bien que les problèmes de syntaxe soient gommés, il n'en subsiste pas moins une vérification à faire quant à la structuration correcte de la marche à suivre, notamment au niveau des structures de contrôle (utilisation d'un *Répète* sans son *Jusqu'à ce que,* oubli de la condition dans une structure de contrôle, utilisation de deux *Sinon* dans le même *Si,…*). Le logiciel signale l'erreur et la ligne à laquelle elle est constatée

◊ et enfin, il faut penser à "éduquer" l'élève à la démarche descendante et structurée. Une *aide à la conception* est donc disponible à tout moment. J'ai pu constater, en testant un logiciel précédent destiné à restructurer les modes opératoires dans l'application de règles (nomenclature des oxydes en chimie minérale), que certains élèves avaient des difficultés alors qu'ils connaissaient parfaitement la matière théorique. Le problème venait de ce qu'ils appliquaient mal les règles. En cas d'erreur, le programme leur présentait, toujours de la même façon et une à une, les questions à se poser afin d'arriver à la solution correcte. Au fur et à mesure des réponses données, un résumé schématisé du "mécanisme mental" à mettre en œuvre pour arriver à la bonne réponse se constituait. De façon très spectaculaire, il suffisait de quelques exécutions de cette aide pour que les modes opératoires corrects soient restaurés.

L'idée est d'utiliser la même stratégie : aborder chaque programme en posant, une à une, les questions dans un ordre adéquat. L'élève a le temps de réfléchir avant d'obtenir la réponse de l'ordinateur, et, en fonction de chaque réponse donnée, il voit une animation qui construit la marche à suivre de façon automatique. Toutefois, le programme ainsi construit n'est pas disponible pour l'élève. Celui-ci doit alors, après avoir quitté l'aide, reconstruire lui-même son propre programme (phase d'appropriation de la démarche). A la longue l'élève acquiert cette démarche et finit par se libérer d'un tel tutorat.

Caractéristiques de la page d'exécution (figure 2)

Cette page contient aussi plusieurs éléments

◊ L'automate prêt à fonctionner

◊ Plusieurs boutons pour déclencher l'exécution de la marche à suivre. Pour valider son programme, l'élève doit lui faire subir une série de tests. Pour chaque programme, deux à quatre tests doivent être exécutés. Leurs conditions initiales sont incluses dans le logiciel afin de détecter le maximum d'erreurs d'analyse possibles. Toutefois, il est signalé à l'élève que la dernière vérification revient au professeur.

◊ Il est indispensable que l'élève connaisse précisément le contexte dans lequel va démarrer l'automate. Les erreurs de l'élève peuvent provenir de ce qu'il n'ait pas imaginé certaines conditions de départ. Il est donc important qu'elles soient affichées afin que l'élève puisse, petit à petit, obtenir un programme qui fonctionne quelles que soient ces conditions de départ. Il apprendra ainsi à mieux analyser et "décortiquer" la situation avant de commencer à programmer.

◊ Un bouton pour obtenir de l'aide

◊ Un bouton pour retourner à la page d'édition et pouvoir corriger son programme

◊ Un bouton pour retourner au menu

◊ Un bouton pour stopper un éventuel bouclage sans fin

◊ Lors de l'exécution de la marche à suivre, un curseur signale avec précision l'instruction qui est exécutée et une zone d'affichage affiche en clair ce que l'automate est en train de faire.

◊ Lors de l'exécution d'une structure de contrôle, plusieurs choses se produisent. Premièrement, le résultat du test de la condition apparaît clairement afin d'attirer l'attention de l'élève sur le fait qu'une condition est testée. Ensuite, la zone d'affichage signale également le résultat de ce test, mais aussi (et surtout) la ligne à laquelle va se continuer le programme. Enfin, un graphisme très clair apparaît à l'écran pour bien montrer de quelle ligne à quelle autre ligne se fait le "saut" dans l'exécution de la marche à suivre. Pendant ce temps, des curseurs secondaires signalent les positions des éléments de la structure impliquée (par exemples les lignes où se trouvent le SI, le SINON et le FIN SI de la structure conditionnelle impliquée).

◊ En cas d'erreur, le programme s'arrête directement. Le curseur qui suit ligne à ligne l'exécution du programme se bloque sur la ligne incriminée afin de signaler précisément où se trouve l'instruction en cause. Une zone d'affichage signale également l'erreur.

◊ Deux types d'exécutions sont possibles : une exécution **automatique** permettant de faire se dérouler le programme sans intervention de l'élève (ceci peut servir à une vérification rapide); un autre type d'exécution, très important en cas d'erreur, est la réalisation **pas à pas** du programme. De cette façon, l'élève peut prendre connaissance, à son rythme, de tout ce qui se passe, tant dans l'évolution de l'automate que dans la succession des lignes d'instructions. Cette méthode est l'outil essentiel de remédiation en cas d'erreur difficilement compréhensible par l'élève.

La réalisation d'une marche à suivre

L'élève, par une simple action "glisser-poser" de la souris, peut tirer autant de fois qu'il le veut n'importe quel élément du cadre "Constituants de la marche à suivre" (figure 1) dans le grand cadre gris. Chaque élément reste "déplaçable" à volonté. S'il désire éliminer un élément, il lui suffit de le tirer n'importe où en dehors du cadre gris pour le faire disparaître.

Les avantages de cette façon de procéder sont multiples :

◊ Rapidité et simplicité de réalisation des marches à suivre

◊ Non disqualification des élèves peu habiles en dactylographie

◊ Facilité de construction des programmes. L'élève peut par exemple disposer o" il le désire les éléments constitutifs d'une structure de contrôle avant de la compléter. Cette méthode est une invitation à travailler par association d'idées, technique très utile dans la mise en œuvre de la démarche descendante et structurée. Cette pratique sera abondamment suggérée dans l'aide à la conception des marches à suivre

◊ Élimination des problèmes de syntaxe. L'élève n'est pas distrait par des détails de virgules, il peut focaliser toute son attention sur l'objectif poursuivi.

Deux difficultés peuvent cependant subsister :

◊ Des difficultés de manipulation de la souris. Très peu d'élèves manifestent ce genre de problème et ils s'adaptent assez rapidement.

◊ Des problèmes de sémantique. il faut bien comprendre le sens du contenu des éléments constitutifs de la marche à suivre (ceux-ci sont toutefois détaillés dans l'énoncé).

La réalisation d'un test

"Essayer un programme peut éventuellement montrer qu'il ne marche pas, jamais prouver qu'il est correct /" (Duchâteau, 1990)

La tâche du logiciel est ici très délicate. Il faut tester automatiquement le programme réalisé en essayant de repérer au maximum les erreurs qui pourraient s'y trouver. Pour chaque test, le logiciel introduit des conditions initiales différentes qui doivent révéler au maximum les erreurs de programmation. Un certain nombre de tests sont à réaliser (entre 2 et 4) pour que le programme de l'élève soit validé. En cas d'erreur, le bouton bleu correspondant au test effectué devient rouge. En cas de succès, il devient vert. L'élève sait que tous les boutons de tests doivent être verts avant d'appeler le professeur pour la validation finale.

Le logiciel tente aussi d'évaluer l'efficacité des programmes. Le critère retenu est l'utilisation du minimum possible de tests de conditions (absence de redondance).

Confrontation de l'environnement proposé au cahier des charges

Comparons l'environnement proposé au cahier des charges développé au chapitre 3.

Ne pas couper l'élève de ses ressources

Le fait que les robots soient virtuels nous permet d'imaginer des situations très différentes (plus variées qu'en LOGO par exemple). Nous veillerons à aborder des tâches qui soient relativement simples et parlantes aux élèves trier des enveloppes de couleurs différentes, charger des armoires sur un camion, arroser des plantes, lancer une pièce de monnaie en l'air jusqu'à obtenir Pile,…

Mettre l'élève devant une tâche clairement définie

Pour chaque énoncé, nous veillons à définir très précisément les quatre éléments essentiels pour que l'élève puisse réaliser sa marche à suivre : l'environnement de l'automate, sa tâche, les actions liées aux instructions et les conditions que l'automate peut tester.

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

L'élève peut concevoir seul ses marches à suivre. Il peut faire autant d'essais qu'il veut en vérifiant toutes les possibilités qui lui passent par la tête.

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

L'élève élabore des programmes aussi variés et compliqués qu'il le désire. L'automate les exécute et, en cas de problème, lui présente deux types d'erreurs :

- ◊ les erreurs « fatales », qui arrêtent le programme car les actions commandées sont impossibles à réaliser (partir dans deux directions différentes par exemple)
- ◊ les erreurs non fatales, qui ne bloquent pas le programme, mais qui font effectuer à l'automate des actions qui ne sont pas efficaces (regarder deux fois les indications,…). Dans ce cas, le programme continue, mais ces erreurs sont signalées dans la zone réservée à cet effet en fin d'exécution.

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

La mise en rapport de la progression dans l'exécution de la marche à suivre et de l'évolution de l'automate, présentée comme une "lunette cognitive inversée" est de nature à atteindre cet objectif. La phase de démonstration, au cours de laquelle l'élève demande l'exécution d'un programme qui est déjà réalisé pour lui, doit lui permettre de percevoir le fonctionnement des structures de contrôle, point le plus délicat de la matière.

Faciliter la compréhension de son erreur par l'élève

Lorsqu'une erreur intervient dans l'exécution du programme, plusieurs éléments sont présentés simultanément à l'élève afin que leur recoupement lui permette de comprendre pourquoi le problème apparaît et la façon de le résoudre.

Ces différents éléments sont :

- ◊ le curseur rouge en face de la ligne qui pose problème
- ◊ le texte signalant précisément de quelle erreur il s'agit
- ◊ l'observation de l'état du robot par rapport à ce qui est attendu

Il faut une gradation correcte des difficultés

Les exercices vont naturellement du plus simple au plus compliqué et ceci selon une gradation constante. Voir la liste des niveaux abordés ci-dessus

Résultats avec les élèves

Cette méthode est en cours d'expérimentation auprès d'élèves de 5e rénové (16-17 ans). Malgré le peu de recul concernant l'utilisation de ce logiciel, nous pouvons déjà dégager certains enseignements :

◊ Diminution du nombre des élèves ayant des problèmes dans le maniement des structures de contrôle

◊ Amélioration de la précision du dialogue. Avant, les élèves venaient signaler qu'ils ne savaient pas et, de plus, qu'ils ne savaient pas ce qu'ils ne savaient pas. Maintenant, les élèves posent des questions précises. Par exemple "dans quel cas utilise-t-on le RÉPÈTE ou le TANT QUE ?"

◊ Amélioration de la vitesse de remédiation. L'élève signale les exercices dans lesquels il a rencontré des difficultés. Le professeur peut lui demander de refaire ces exercices en exécution pas à pas. Le doigt est rapidement mis sur le problème et le professeur peut mener une remédiation plus efficace.

◊ Amélioration du contrôle des capacités. Le professeur peut exécuter une marche à suivre pas à pas et demander à l'élève de prévoir ce qui va se passer au pas suivant. Cette méthode est excellente pour développer et tester la connaissance des structures de contrôle.

Cet environnement montre de prime abord des avantages intéressants. Il nécessite toutefois une étude approfondie afin d'en dégager un bilan pédagogique plus complet.

Conclusions

Nous faisons appel à l'étude de la programmation dans le but de développer les capacités d'abstraction et de logique chez des élèves de 16-18 ans. Or, il s'avère que certaines méthodes d'apprentissage de la programmation nécessitent à un haut degré ces capacités en tant que préalables. L'étude du problème semble révéler que les difficultés apparaissent surtout au niveau de la manipulation des structures de contrôle (alternative et répétitives). La pertinence de mettre en œuvre des robots imaginaires ou abstraits a retenu toute notre attention et nous avons voulu pousser plus loin cette démarche en les rendant virtuels. Un logiciel ludique dans lequel l'élève doit réaliser les algorithmes mettant ces robots à l'œuvre a donc vu le jour. Dans une première phase, des algorithmes déjà réalisés sont présentés à l'élève afin de lui "montrer" notamment comment fonctionnent les différentes structures de contrôle. Ceci se fait en plaçant en parallèle et de façon simultanée l'exécution des instructions de la marche à suivre et les actions réalisées par le robot. Ensuite, l'élève doit réaliser ses propres algorithmes en réponse à des problèmes précis de difficulté croissante. La méthode, qui commence à être utilisée, semble dégager des résultats intéressants tant au niveau des élèves, qui progressent mieux et posent des questions plus précises en cas de problème, qu'au niveau des professeurs, qui disposent d'un outil concret d'apprentissage, de test de connaissance et d'aide pour la récupération de lacunes chez les élèves en difficulté. Ce logiciel semble donc une aide efficace pour aborder la programmation, que ce soit en tant qu'introduction à l'étude de l'informatique ou en tant que préparation à la robotique pédagogique.

Remerciements

Tous mes remerciements vont

◊ à Charles Duchâteau pour la lecture de cet article et pour la pertinence de ses remarques et suggestions
◊ à Marie-Cécile Allain pour ses conseils judicieux
◊ à Isabelle et Virginie pour leur difficulté à apprendre ce que je leur enseignais mal.

Note

Il convient cependant de rester prudent. Leroux signale des « *difficultés liées à la programmation par des publics dits de bas niveau de qualification* » qui montrent qu'en robotique pédagogique aussi, certains préalables sont indispensables. « *en fait, les difficultés rencontrées sont inhérentes à l'apprentissage de tout langage de programmation : non-respect de la syntaxe (mauvaise écriture de noms de commande, oubli d'espaces) mais très certainement aussi à des problèmes d'algorithmique (confusion dans l'emploi des instructions conditionnelles et de répétition) et d'analyse (mauvais enchaînement des commandes pour réaliser un mouvement). Ce sont des hypothèses que nous émettons en l'absence d'une étude fine de ces phénomènes* » (Leroux, 1996). L'auteur ajoute: « *il était nécessaire de créer un environnement de programmation dans lequel les contraintes syntaxiques des langages de programmation sont supprimées tout en conservant une richesse dans la structure des programmes. Il est clair qu'un tel environnement n'élimine pas les difficultés conceptuelles liées à la programmation, mais au contraire, permet de focaliser l'attention sur les notions fondamentales* ».

Bibliographie

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

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

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

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

DUCHÂTEAU, Ch. (1994). *Faut-il enseigner l'informatique à ses utilisateurs* ?, Communication au 4e colloque francophone sur la didactique de l'Informatique, Québec, *Formation-recherche en éducation no* 5.38, Publications du CeFIS, Namur (B), Facultés Notre-Dame de la Paix,

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

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

MEURICE de DORMALE, R. *Détection des lacunes et évaluation gérées par ordinateur,* non publié

RELLIER, C. et SOURDILLAT, F. (1993). *Evariste et la lunette cognitive,* in DENIS, B. et BARON, G.-L. éd., *Regards sur la robotique pédagogique, Actes du quatrième colloque international sur la robotique pédagogique,* Liège (B), Université de Liège, pp.179-186

ROMAINVILLE, M. (1988). *Une analyse critique de l'initiation à l'informatique : quels apprentissages et quels transferts?* in *Actes du colloque francophone sur la didactique de l'informatique,* Paris, EPI, pp.225-241

SOUGNE, J. (1993). *Les raisonnements temporels en robotique pédagogique,* in DENIS, B. et BARON, G.-L. éd., *Regards sur la robotique pédagogique, Actes du quatrième colloque international sur la robotique pédagogique,* Liège (B), Université de Liège, pp.85-93

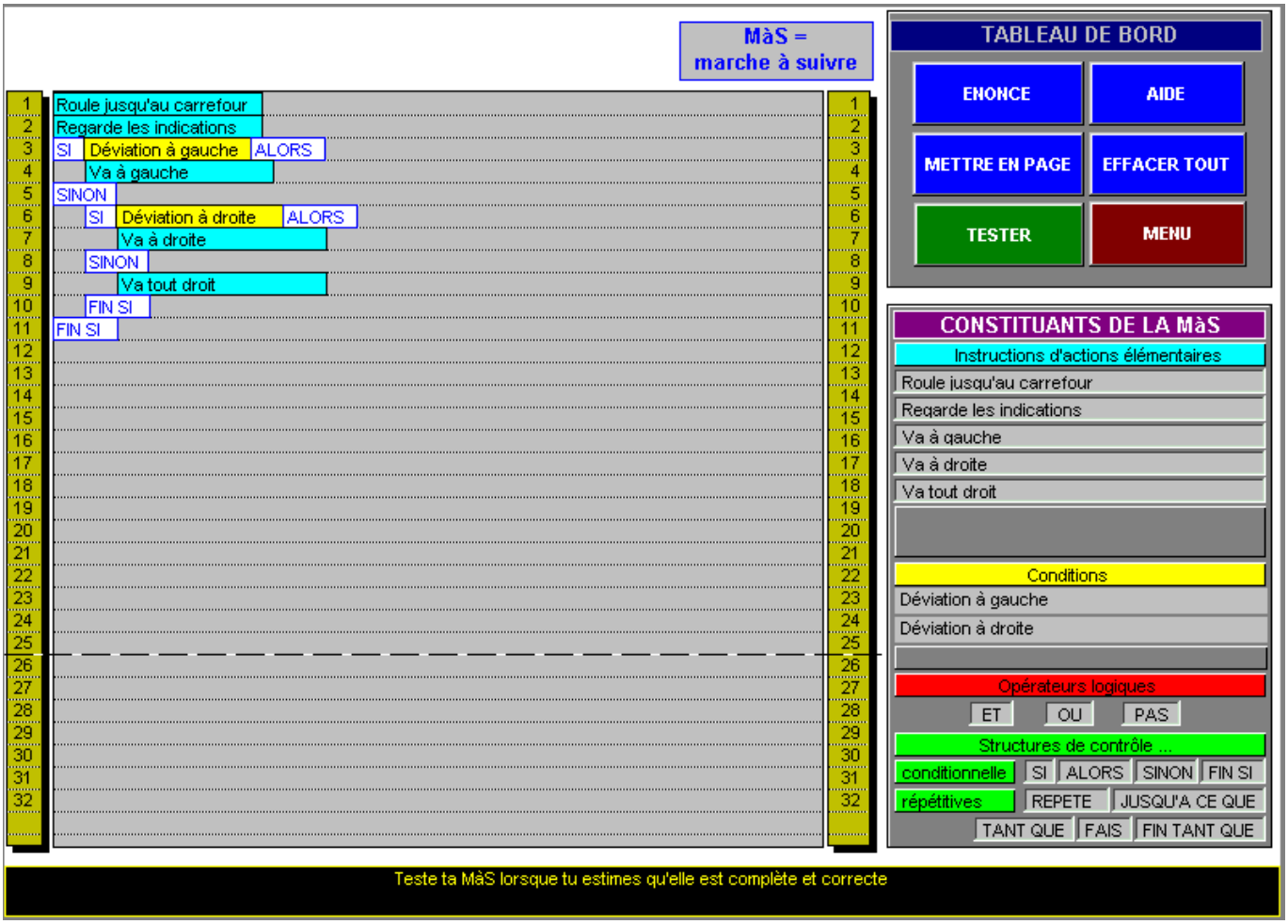

Figure 1

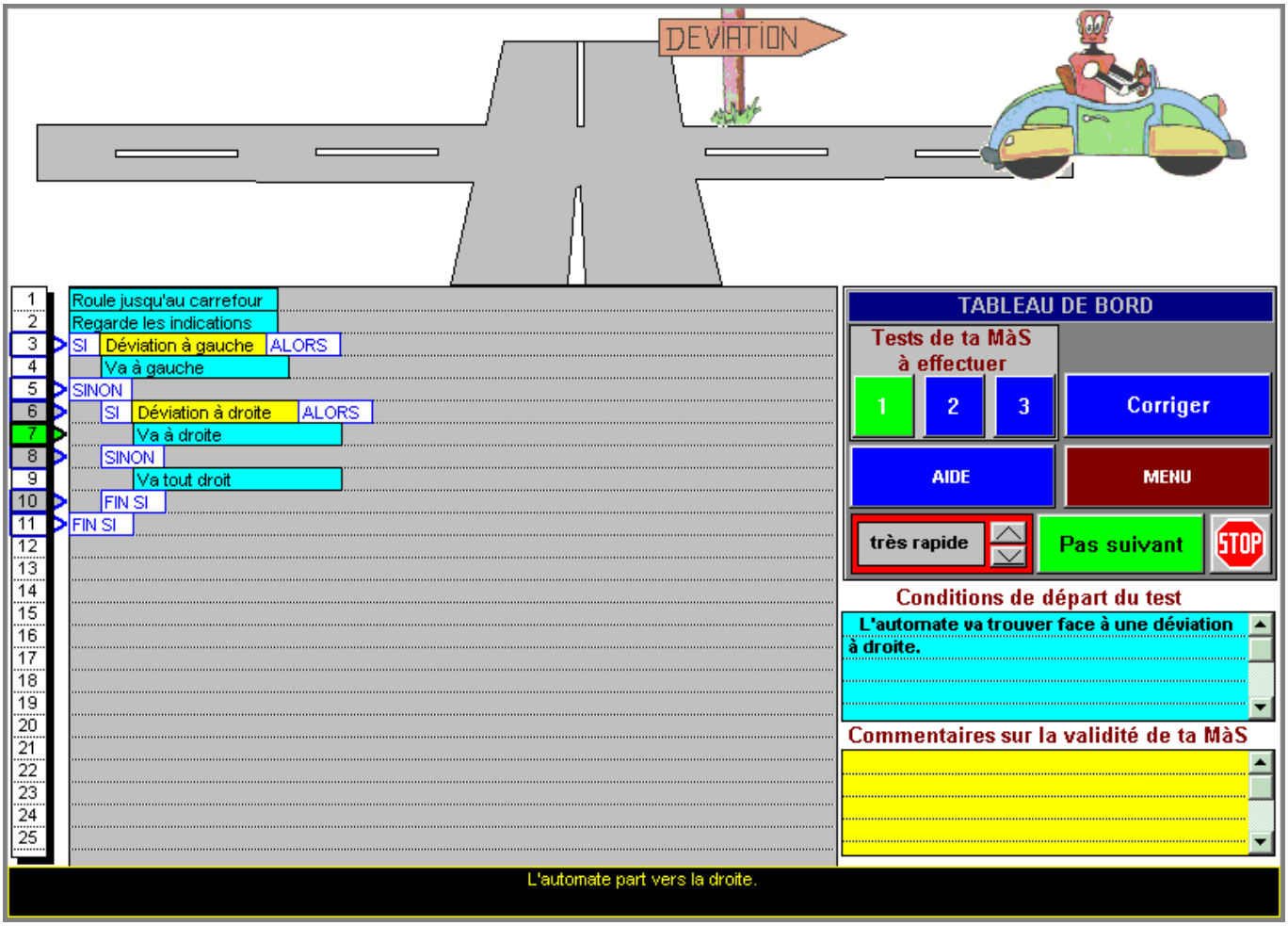

Figure 2

PS : les dessins des robots ont été réalisés par Jean-Baptiste Sonnet. Coordonnées sur demande.