DE L'ENSEIGNEMENT DE CONCEPTS INFORMATIQUES

Stéphane et Florence DUCASSE
École française de Berne

1. INTRODUCTION

Le cours de technologie en collège peut être un terrain propice à l'utilisation de nouvelles technologies lorsque celles-ci sont utilisées de manière adéquate. Cependant, mettre des ordinateurs dans les classes et l'accès à Internet ne sont que des solutions quantitatives alors que la solution est qualitative. Comme le faisait remarquer Alan Kay, le créateur de Smalltalk [BS96], ce n'est pas parce que l'on met un piano dans chaque classe de cours que l'on sollicite des vocations de pianistes, en particulier lorsque les enseignants ne sont pas eux-mêmes musiciens ou sensibilisés à la musique [Kay]. Cette triste mais évidente constatation trouve son écho dans les cours de technologie de collège.

Nous avons été confrontés au manque d'approche permettant aux élèves (1) d'avoir une attitude constructive et non consommatrice dans un contexte informatique et (2) d'utiliser l'informatique, non plus comme un outil au service d'autres matières mais, comme une science ayant ses propres concepts et possibilités d'enseignement. En effet, apprendre la bureautique (traitement de texte, tableur...) bien que pouvant être intéressant dans un premier temps, reste assez limité quand au support offert en terme de concepts enseignés, de créativité et de variations possibles. C'est pourquoi nous avons défini notre propre approche basée sur l'utilisation d'une tortue à la Logo pour présenter des *concepts de programmation*. Il faut noter que les approches à base de tortues ou autre animal informatiques après avoir été longtemps utilisées [WPcM79, Wer81, Men88], semblent être tombées en désuétude. Il est dommage de voir que de telles approches ont été abandonnées car notre expérience nous montre que les élèves sont très motivés, expérimentent beaucoup et assimilent des concepts non triviaux.

Dans cet article nous présentons comment nous avons utilisé une approche basée sur une tortue en Smalltalk afin d'enseigner des concepts informatiques de programmation. Nous abordons le contexte et les motivations qui nous ont poussé à définir ce cours. Nous présentons ensuite notre approche, et montrons un exemple d'interrogation, puis les résultats de l'apprentissage par les élèves. Nous expliquons aussi comment nous avons intégré l'utilisation d'un site web collaboratif comme médium pour la présentation des résultats. Nous finissons par une évaluation complète de l'approche et donnons des informations pratiques.

2. POURQUOI ? OU LES FAIBLESSES DES APPROCHES UTILISÉES

Nous discutons les différentes possibilités offertes dans le cadre d'un enseignement de technologie en collège.

- La bureautique, même guidée dans le cadre d'un journal de classe ou d'établissement, est une approche limitée car les élèves connaissent souvent très bien des programmes comme Word et sont souvent inintéressés. Il est difficile que chaque élève soit actif et partie prenante du projet. De plus, bien que les logiciels actuels soient très puissants les concepts restent élémentaires.
- L'utilisation de logiciels dits éducatifs est aussi limitée car ils sont souvent *trop* éducatifs et poussent à l'acquisition d'autres matières comme les mathématiques.
- L'utilisation d'internet reproduit le plus clair du temps un comportement de consommateurs et zappeurs devant une information inorganisée et de qualité diverse. De plus, la recherche d'information est une activité qui reste limitée et est toujours définie dans le cadre d'autres activités.
- Lego Mindstorm propose une alternative intéressante dans le sens où il permet aux élèves de développer leur propre robot. Cependant, cette approche a les désavantages suivants : elle a un coût élevé, l'environnement de programmation est assez limité (il est par exemple compliqué d'exprimer des concepts comme la conjonction (et) et le choix (ou)) et la qualité des briques logicielles pourraient être grandement améliorée.

Dans notre situation, la plupart des élèves avaient déjà suivi une année d'utilisation de Word avec lequel ils avaient effectué divers projets comme le journal de l'école et connaissaient parfaitement les possibilités offertes par le logiciel. De plus, ils se sont plaints ouvertement d'ennui et ont demandé à changer.

3. LA CONSTRUCTION DE PROGRAMMES

Notre approche comme de nombreuses approches utilise une tortue à la LOGO [Ad80, Wat84]. Cependant, les concepts enseignés sont des *concepts informatiques*. D'une manière générale, l'utilisation d'une tortue apporte les caractéristiques suivantes.

La responsabilité. L'élève est *responsable* des tortues qu'il crée. Il est responsable de créer les figures qui lui plaisent. Quand il réussit, il s'agit de son travail et non d'exercices fixes bien que des exercices soient proposés.

L'exploration. D'autre part, l'élève est libre et encouragé à expérimenter. Le nombre de variations est quasi illimité. L'aspect graphique de l'exécution des programmes aide à les débugger et à comprendre les erreurs commises et est une donnée importante de l'approche. L'élève peut être son propre correcteur et l'erreur est formatrice.

La créativité. Le résultat des programmes étant graphiques, la motivation de l'élève est plus grande. La beauté de certains graphiques comme les structures récursives motive la compréhension de concepts ardus.

Les points précédents même s'ils sont capitaux n'en sont pas pour autant originaux [Pap71]. L'originalité du travail présenté ici réside dans les points suivants :

Des concepts informatiques. Nous enseignons des *concepts informatiques* et non mathématiques. Nous voulons que les élèves acquièrent (1) des notions élémentaires telles que la notion d'état, les boucles, les variables, (2) une compréhension de la notion de processus ou algorithmes et (3) la notion d'abstraction, i.e. définition d'abstractions en termes d'abstractions.

Orientés par des problèmes. Un concept est toujours introduit en montrant qu'il est nécessaire et que l'on ne peut pas résoudre certains problèmes sans ce concept.

La formulation et validation d'hypothèses. Les nouveaux concepts sont introduits en demandant à l'élève de formuler des hypothèses quant aux résultats et à les vérifier à l'aide des tortues.

Minimaliste. Nous avons limité au maximum les concepts introduits. De plus, en nous greffant sur Smalltalk [AB88, BS96] nous n'avons pas défini de nouveaux éléments syntaxiques ou sémantiques, seulement limité les explications de certains concepts comme celui des classes que nous avons introduit comme la notion de fabrique d'objets.

Les contraintes

Lors de la conception de l'approche, nous avons défini les contraintes suivantes. Nous avons souhaité : (1) baser notre travail sur un *véritable* environnement non pas spécialisé pour un groupe d'élèves mais sur un système pouvant être utilisé dans de multiples projets, (2) utiliser un produit *gratuit* et *disponible* sur différentes plates-formes comme des Macintosh ou des PC. Avoir un *contrôle total*. Certaines approches semblent intéressantes comme Cocoa [SC99] ou StarLogo [Res94] mais nous voulions avoir le contrôle total de l'approche afin de pouvoir l'ajuster au retour des élèves. Notre approche par son minimalisme permet à toute personne comprenant la programmation objet de contrôler complètement l'environnement. Notre choix s'est porté sur Squeak, le nouvel environnement développé par Alan Kay et son équipe chez Disney [Squ]. Squeak est gratuit et « open source », il est disponsible pour plus d'une dizaine d'ordinateurs et de systèmes d'exploitation différents. Squeak est un Smalltalk qui intègre toutes les composantes multimédia : connexion avec le web, sons, midi, nombreux supports graphiques, modélisations 3D...

Audience

Notre principale audience est un public de collégiens ou de lycéens. Suivant le niveau les concepts plus difficiles tels que la récursivité ou la programmation objet peuvent être accentués. D'autre part, on peut considérer que ce cours pourrait être la base d'une introduction à la programmation suivie d'un cours élémentaire de programmation objets.

Stimuler les élèves : Le Pop Art

Afin de donner une visibilité aux élèves et de les stimuler, nous les avons laissé créer une galerie de leur graphes préférés. Après avoir programmé, ils peuvent capturer l'écran, le peindre à leur goût et

l'imprimer puis l'afficher. Nous avons insisté pour qu'ils capturent aussi le script ayant servi à la génération du graphe. Nous avons aussi commencé à utiliser un Wiki [1] afin de sortir du cadre de l'école comme présenté en 7.

4. LE COURS

Nous présentons les concepts que nous enseignons dans le cours. Le cours se décompose en trois parties (seule la première est écrite). La première partie « Une tortue comme un crayon » présente le concept essentiel de la programmation, la seconde « Une tortue comme un animal » présente les concepts spécifiques de la programmation objets et la dernière introduit la notion d'applications en faisant programmer un robot et son environnement.

Structure du cours et concepts abordés

Voici les chapitres qui composent la première partie du cours. Certains autres chapitres sont en cours de validation. Dans cette partie, nous utilisons et introduisons les concepts mathématiques suivants lorsqu'ils sont nécessaires : direction et repère dans le plan, angles, rotation absolue/relative, déplacement relatif.

Prise en main. Mise en place du système, premier contact avec une tortue.

Qu'est-ce qu'une tortue. Une tortue est un objet informatique qui est manipulable et possède un état. Une tortue a une direction, une couleur et une position dans un espace plan. Elle peut être manipulée en lui envoyant une instruction ou un message qu'elle exécute. Une tortue reçoit des messages et change son état en fonction, i.e. se déplace, se dirige ou change sa couleur – exemple : tracer un SOS graphique.

Angles. Présentation des méthodes permettant à une tortue de changer de direction de manière absolue et relative – exemple : un carré.

Boucles. Présentation de la nécessité de boucles. Une boucle ici est présentée comme un *itérateur*. Une boucle fait une séquence de messages un nombre donné de fois – exemple : carré et escalier.

Variables. Notion de variables pour la manipulation de nombres. Une variable est un nom *symbol* qui dénote une chose informatique.

1. Site internet permettant l'édition collaborative de pages.

Abstractions. Définition de nouvelles méthodes. Une méthode est présentée comme un moyen de donner un nom à une séquence de messages. Les méthodes n'ont pas d'arguments – exemple : la méthode qui dessine un carré de 100 pixels.

Composition de méthodes. La réutilisation et composition de méthodes sont mises en avant. En particulier, des éléments graphiques sont réutilisés dans différents graphiques – exemple : une boîte composée de plusieurs carrés.

Arguments. Le besoin d'arguments est montré, puis les arguments sont introduits – exemple : un carré dont on spécifie la taille.

Composition. La composition de méthodes avec arguments est abordée.

Nous sommes actuellement en train de définir les chapitres présentant les déplacements absolus, les conditions et les collections.

Remarque. Il faut noter que nous avons fait particulièrement attention, lors de l'introduction de ces concepts, à ce qu'ils représentent du code élégant et réaliste. En effet, certains auteurs dont les ouvrages sont certes très intéressants et furent une source d'inspiration, introduisent des procédures récursives bouclant indéfiniment – comme `POLY` dont la définition est montrée ci-après – est ne pouvant se terminer [Ad80, Wat84]. Nous pensons qu'il faut proscrire de telles pratiques si l'on veut apprendre aux élèves à écrire des méthodes viables [AD96].

```
TO POLY SIDE ANGLE
FORWARD SIDE
RIGHT ANGLE
POLY SIDE ANGLE
```

5. UN ENVIRONNEMENT MINIMALISTE

Nous avons choisi une approche résolument minimaliste : le support linguistique offert par Smalltalk est totalement utilisé, nous n'avons introduit aucun élément[2] syntaxique ou sémantique. D'autre part, l'environnement est réduit à sa plus simple expression : un éditeur de code et un interprète comme le montrent les figures 1 et 2.

2. Le lecteur doit être conscient que nous utilisons le terme méthode pour parler d'une abstraction (le nom d'un script) qui pourrait être une fonction dans les langages fonctionnels ou une procédure dans les langages procéduraux. Ici le terme méthode est dénué de la sémantique qu'il implique dans les langages à objets à savoir le polymorphisme.

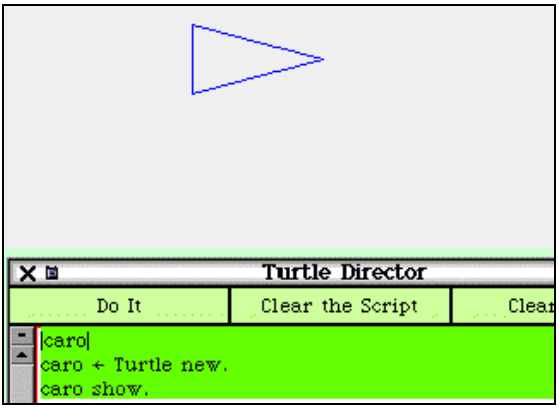

Figure 1. Un script est tapé puis exécuté, l'élève voit immédiatement le résultat.

Le langage des tortues

Voici la liste des méthodes que nous avons utilisées. Notez que ces méthodes ne sont introduites que lorsqu'elles s'avèrent nécessaires. Nous sommes en train d'évaluer s'il est intéressant d'introduire les méthodes `back:` qui fait reculer une tortue d'une certaine distance et `arcLeft:radius:` qui dessine un arc de cercle. Il faut aussi remarquer que l'implantation de la tortue reste assez simple et peut être très facilement modifiée ou adaptée à de nouveaux choix pédagogiques.

Nom	**Description**
`show`	Une tortue s'affiche à l'écran, elle est repérée par un triangle qui par défaut pointe à l'est. **Par défaut une tortue ne s'affiche pas.**
`hide`	Efface le triangle qui représente une tortue.
`trace`	Une tortue abaisse son stylo pour écrire, alors elle peut se déplacer en laissant une trace.
`noTrace`	Une tortue léve son stylo de l'écran, elle n'écrit plus.
`color: aColor`	Change la couleur laissée par une tortue
`go: aNumber`	Une tortue avance d'un certain nombre de pixels.
`north, east, west, south`	Ordonne à la tortue de pointer au nord, est, ouest ou sud.
`turnLeft: aNumber, turnRight: aNumber`	Ordonne à la tortue de tourner à droite ou à gauche d'un certain angle par rapport à la direction courante.
`n timesRepeat:[...]`	Répéte n fois une séquence de messages.
`pointTo: aPoint`	Une tortue s'oriente pour pointer vers un point donné.
`goAt: aPoint`	Une tortue se place à un point donné.

Deux scripts

Afin de donner un aperçu des scripts, nous montrons ici deux scripts, un dessinant un carré et un dessinant une spirale.

Le script carré. Le script du carré sans utilisation de boucles est proposé comme exercice lors du chapitre sur les angles au tout début du cours.

```
|caro|
caro:= Turtle new.
caro trace.
caro go: 100.
caro turnLeft:90.
caro go: 100.
caro turnLeft:90.
caro go: 100.
caro turnLeft:90.
caro go: 100.
caro turnLeft:90.
caro noTrace.
```

Le script spirale. Le script suivant génère une spirale. Il est proposé dans le chapitre qui combine les boucles et les variables. Il montre l'utilisation de la variable `longueur` dans le contexte d'une boucle. La nécessité de l'initialisation et de l'incrément de la variable sont alors des points primordiaux du chapitre.

```
|caro longueur|
caro:= Turtle new.
longueur:= 5.
caro trace.
200 timesRepeat: [caro go: longueur.
                  caro turnLeft: 91.
                  longueur:= longueur + 3]
```

L'environnement

La conception de l'environnement suit volontairement une présentation Smalltalkienne : l'environnement se compose d'un *interpréteur* de scripts (figure 2) et d'un *éditeur*. L'éditeur n'est introduit que lorsque la définition de méthode est abordée.

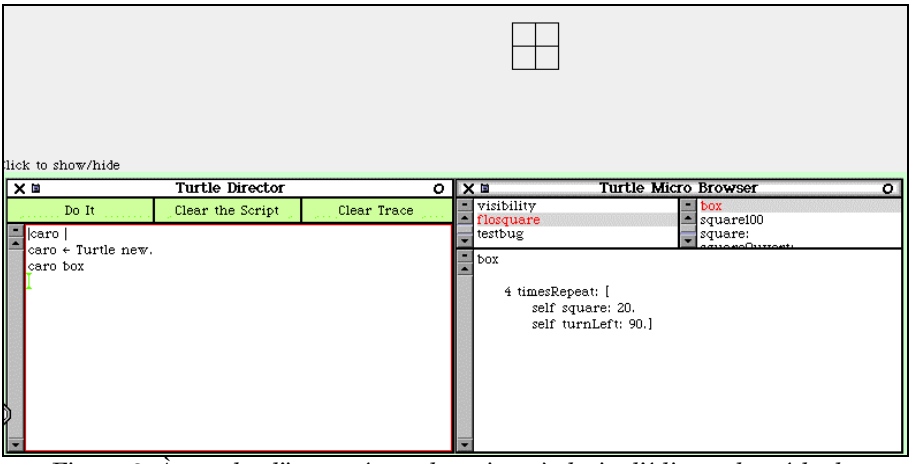

Figure 2. À gauche, l'interpréteur de scripts, à droite l'éditeur de méthodes et au-dessus le résultat de l'évaluation du script défini dans l'interprète.

6. ANALYSE DE L'ACQUISITION DE CONNAISSANCES

Dans cette partie, nous décrivons comment nous avons évalué si nos objectifs d'apprentissage ont été atteints et les erreurs que nous avons commises.

Contexte de l'expérience

Le cours que nous avons décrit a été donné aux élèves de 6ème et 5ème de l'École Française de Berne. Les effectifs des classes sont réduits : 12 en 5ème et 7 en 6ème. Les élèves ont 1h 30 de technologie par semaine. En général, un chapitre a été présenté sur une ou deux séances. La salle informatique est composée de 8 PC récents sur lesquels les élèves ont installé Squeak et l'environnement de la tortue. Les élèves travaillent par binôme. Cependant, notre expérience nous a montré que la pratique du binôme est source de stimulation plutôt qu'un frein.

Évaluation des acquisitions

Afin d'évaluer notre support de cours [DD00], nous avons posé aux élèves les questions suivantes. Il faut noter que nous ne leur avons jamais demandé d'apprendre ces concepts mais toujours montré en quoi ils étaient utiles pour diriger les tortues.

Qu'est-ce qu'une variable ? Qu'est-ce qu'une boucle ? Est-ce que le script suivant fonctionne ? Si non proposez une correction. Qu'est-ce qu'une méthode ?

```
|caro longueur|
caro:= Turtle new.
caro trace.
10 timesRepeat: [caro east.
                 caro go: longueur.
                 caro north.
                 caro go: 10.
                 longueur:= longueur + 10]
```

Voici quelques-unes de leurs réponses qui montrent que dans l'ensemble les concepts sont compris même si cette compréhension peut rester parfois vague.

Martin (Élève de 5ème). *Une boucle sert à répéter le nombre de fois que l'on veut la même chose. Une variable permet de créer des labyrinthes. Elle change des données. L'erreur est « longueur » n'est pas définie et l'ordinateur ne sait pas ce que c'est.*

Moncef (Élève de 5ème). *Une boucle permet de répéter un script sans avoir à réécrire le script le nombre de fois qu'on veut répéter. Il manque avant la boucle* `longueur:= 10`*. Une méthode ça sert à simplifier un long script en lui donnant un nom.*

Retour des élèves

Avant le cours, les machines n'étaient pas connectées à l'Internet et les élèves réclamaient que l'on puisse y avoir accès comme cela avait été le cas les années précédentes et en début d'année. Nous avions craint qu'une reconnection soit fatale à notre approche et que les élèves soient démotivés. A notre grande surprise, après avoir fait quelques recherches sur Internet dans le cadre d'autres activités, les élèves ont demandé à continuer à programmer les tortues et à aborder des concepts délicats comme celui de la définition d'abstraction et de composition de codes avec enthousiasme délaissant les navigateurs. *« Ça y est madame, j'ai trouvé la biographie d'Einstein. Est-ce que je peux retourner aux tortues »* sic. Bien entendu comme dans tout groupe d'élèves, certains ne sont jamais motivés.

Problèmes rencontrés

Le support de cours est dans sa forme actuelle bien trop verbeux pour des élèves. En effet, les élèves ne le lisent pas en général. En réaction à ce problème nous avons continué à définir le support dans sa forme verbeuse car notre public est aussi de possibles enseignants mais nous avons proposé aux élèves des fiches d'activité qui les poussent à lire le document d'une manière active à la recherche d'informations.

Dans une première version du cours, nous avions introduit les variables à l'aide de labyrinthes et de boucles. Cela a été une erreur car les élèves n'ont pas compris clairement le concept de variables et l'ont associé à celui des boucles. Dans le cours actuel, les variables sont présentées de manière indépendantes des autres concepts.

Dans une première version du cours nous avons utilisé le script suivant proposé par [Ad80, Wat84]. Dans la version proposée, la procédure est une récursion infinie ce que nous voulions proscrire, nous l'avions donc transformée à l'aide d'une boucle simple comme suit.

```
g1033 |caro angle|
caro:= Turtle new.
caro trace.
angle:= 20.
300 timesRepeat: [caro go: 30.
                  caro turnLeft: angle.
                  angle:= angle + 7]
```

Ce script et sa famille générée en changeant les paramètres produisent de magnifiques graphes mais posent le problème suivant. Il faut évaluer un nombre non facilement calculable de fois la boucle. Utiliser ce script a conduit les élèves a avoir une vue dénaturée de la notion de boucles. En effet, comme pour réaliser un joli dessin, il fallait boucler un grand nombre de fois aléatoires, ils ont perdu le sens de la répétition *finie*. Un élève a même écrit un carré avec 250 au lieu de 4 répétitions, c'est pourquoi nous avons enlevé ce script du cours.

7. WIKI ET TRAVAIL COLLABORATIF

De nos jours Internet semble être la solution à tous les problèmes aussi bien d'ordres économiques que d'enseignement, mais toute personne utilisant professionnellement Internet sait que comme tout médium, le meilleur comme le pire se côtoient. D'autre part, la profusion

d'informations non organisées rend leur utilisation assez aléatoire. Cependant, une utilisation intéressante est la notion de *Wiki*.

La Pop Gallerie sur le net

Dans le cadre de notre expérience à l'École Française de Berne, nous avons commencé à utiliser un Wiki pour constituer une galerie des graphes produits par les programmeurs de tortues. Un Wiki est un serveur de page html permettant une édition simplifiée et collaborative.

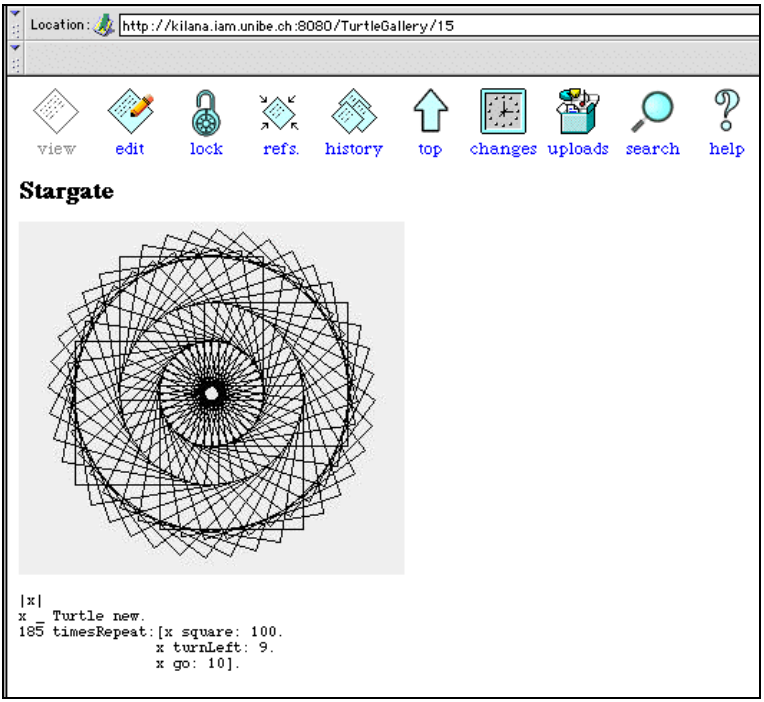

Figure 3. Une des pages créée par un des élèves en utilisant le Wiki.

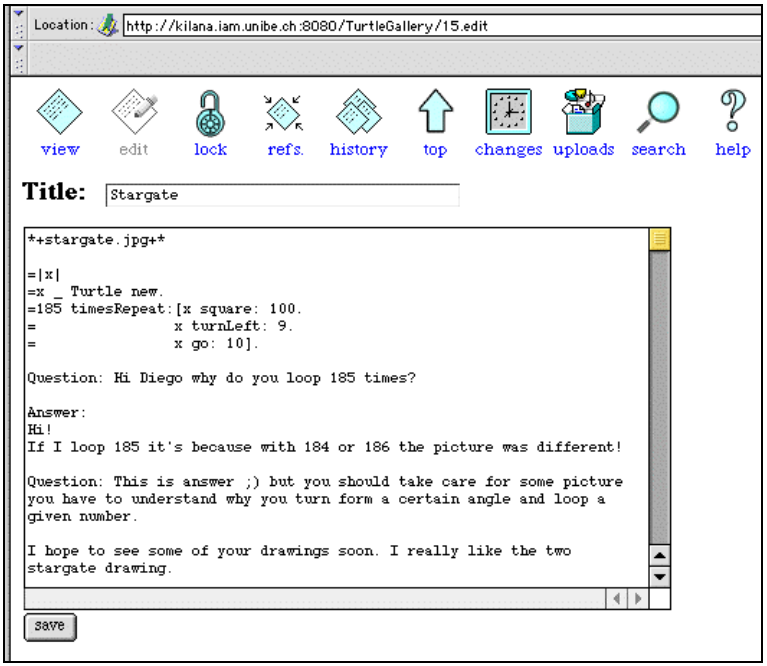

Figure 4. La même page en mode édition.

Les avantages sont : *Simplicité d'utilisation.* La description d'une page est loin d'être aussi complexe que HTML comme le montre la figure 4, donc il est très facile d'écrire des pages *lisibles* avec un Wiki. *Accessibilité.* Pour éditer ou créer de nouvelles pages, il est juste nécessaire d'avoir un navigateur. Aucun autre logiciel, à l'exception du Wiki n'est nécessaire. Les élèves peuvent donc modifier les pages d'où ils veulent, même de chez eux si le Wiki est public. *Favorise la collaboration.* Un Wiki est accessible et modifiable par différents utilisateurs en même temps. C'est donc un support simple dans le cadre de travail collaboratif. Notons qu'un Wiki propose divers niveaux de sécurité. *Gratuité.* Un wiki complet est fourni avec Squeak – un webserveur nommé Comanche est aussi disponible en open source mais est plus difficile à installer. Il n'est donc pas nécessaire d'utiliser des logiciels payants comme FrontPage.

8. ÉVALUATION

Au niveau pédagogique

Notre approche permet à l'élève de s'approprier un ordinateur non plus comme un outil de consommation mais comme un outil de création. En cela, elle remplit une connexion épistémologique importante. D'autre part, l'élaboration d'algorithmes et la décomposition de processus en processus plus simples sont des concepts favorisant l'abstraction que les élèves retrouvent en mathématiques. Permettre à l'élève de contrôler des tortues possède un aspect passionnant qui permet aussi à l'élève de démystifier l'ordinateur : un ordinateur n'est qu'une grosse machine à calculer qui se trompe quand l'élève programme des scripts erronés. S'il y a une erreur c'est l'erreur de l'élève, mais l'erreur n'est pas pénalisante car elle est une des bases de l'expérimentation. L'élève par son raisonnement et le support graphique doit être à même de corriger ses erreurs [Pap71]. L'approche étant basée sur la production de graphiques, les concepts mathématiques de repères, de déplacements dans l'espace, de rotations et d'angles sont nécessaires mais pas le sujet principal des cours donc appréhendés de manière non scolaire.

En conclusion. L'approche proposée, tout comme les approches basées sur Logo, permet à l'élève (1) d'être créatif et non consommateur, (2) de définir des processus en termes logiques et de les valider expérimentalement et (3) de décomposer logiquement des tâches et de les recomposer. L'élève ne subit plus l'informatique comme une autre façon d'aborder d'autres matières mais comme un domaine d'expérimentation et d'apprentissage. Il faut noter que bien que centré sur des concepts l'informatiques ce cours aide à l'apprentissage de concepts mathématiques : en effet, il est plus excitant pour un élève d'apprendre la notion de repère parce qu'il veut diriger une tortue que juste pour apprendre le concept de repère.

Au niveau de l'approche

Les concepts abordés reposant sur des concepts mathématiques élémentaires, il est relativement aisé à un professeur de physiques (notre cas) ou de mathématiques d'enseigner ces concepts. D'autre part, l'utilisation d'une technologie objet et de Smalltalk [BS96] en particulier possède les caractéristiques suivantes :

Antropomorphisme. Il est facile de présenter la tortue en disant que l'on crée une tortue et qu'on lui envoie des messages. D'autre part, plusieurs tortues peuvent être créées pour créer des graphes complexes.

Lisibilité. Avoir des messages clairs aide à la compréhension des concepts

```
caro north.
caro turnLeft: 125
```

est bien plus lisible que

```
tl 90
```

Adaptabilité. L'utilisation de Smalltalk nous permet de définir ou redéfinir les messages que les tortues comprennent. Ainsi créer une version française est réalisable en moins de 10 minutes comme le montre le code suivant.

```
Turtle>>tourneAGauche: angle
        self turnLeft: angle
```

Notons que ceci peut être réalisé par les *élèves eux-mêmes* ou un non spécialiste, ce qui fait toute la différence !!

9. INFORMATION

Recherche d'expérimentateurs. Notre support de cours va être utilisé dans le cadre d'une expérience similaire dans un collège suisse avec des élèves de 11 ans. Si notre expérience a éveillé votre curiosité et que vous souhaitez lire et nous donner votre avis ou si vous souhaitez utiliser notre support de cours et environnement n'hésitez pas à rentrer en contact avec nous (http://www.iam.unibe.ch/~ducasse/).

Liens. Voici quelques pages dont les informations seront certainement utiles.

Le support de cours décrit dans cet article :

http://www.iam.unibe.ch/~ducasse/WebPages/Teaching.html#turtle

Le Wiki utilisé dans notre cours :

http://kilana.unibe.ch:8080/TurtleGallery/

Squeak, le smalltalk open source de Disney :

http://www.squeak.org/

Des liens sur Squeak :

http://kilana.unibe.ch:8080/SmalltalkWiki/2

Stéphane et Florence DUCASSE
Université de Berne,
École Française de Berne.
ducasse@iam.unibe.ch

RÉFÉRENCES

[AB88] Michel Aubé and Danièle Bracke. En quoi consiste le langage smalltalk. *EPI*, (52) :212–216, December 1988.

[Ad80] Harold Abelson and Andrea diSessa. *Turtle Geometry*. MIT Press,1980.

[AD96] Laurent Arditi and Stéphane Ducasse. *La programmation : une approche fonctionnelle et récursive*. Eyrolles, Paris, 1996. 238 pages, ISBN : 2-212-08915-5.

[BS96] Xavier Briffault and Gérard Sabah. *Smalltalk : Programmation orientée objet et développement d'applications*. Eyrolles, 1996.

[DD00] Stéphane Ducasse and Florence Ducasse. Caro, *Dis-moi c'est quoi programmer ?*, 2000. Support de cours de Technologie, 150 pages, http://www.iam.unibe.ch/~ducasse/.

[Kay] Alan Kay. Revealing the elephant : The use and misuse of computers in education. 4(31) :22–28. http://toLearn.net/marketing/kay1.html.

[Men88] Patrick Mendelsohn. Schemes informatiques programmables : utiliser l'environnement logo pour construire des situations d'apprentissage de tracés graphiques. *Petit x*, pages 7–45, 1988.

[Pap71] Seymour Papert. Teaching children to be mathematicians vs. teaching about mathematics. Tech. Report Memo 249 – Logo Memo 4, MIT, AI Laboratory, 1971.

[Res94] M. Resnick. *Turtles, termites and traffic jams*. MIT Press, 1994.

[SC99] David Canfield Smith and Allen Cypher. Making programming easier for children. In *The Design of Children's Technology*, pages 201–221. Morgan Kaufmann Publishers, 1999.

[Squ] Squeak. http://www.squeak.org/.

[Wat84] Daniel Watt. *Learning With Apple Logo*. McGraw-Hill, 1984.

[Wer81] Harald Wertz. Some ideas on the educational use of computers. In *Proceedings of the Annual Conference of the ACM*, 1981.

[WPcM79] Harald Wertz, Daniel Perolat, and François Mathieu. *L'expérience d'Arc et Senans*. Technical report, Université Paris 8 (Vincennes), 1979.