DÉFENDRE AUJOURD'HUI L'ART DE PROGRAMMER...

Jean-Noël JOFFIN

L'abandon de l'option informatique et son remplacement timide par les APTIC, particulièrement mal définis et peu défendus, a pratiquement éliminé la programmation aujourd'hui dans les lycées sauf probablement dans quelques sections techniques où elle reste nécessaire *.

Cette situation est regrettable pour de nombreuses raisons que je tente de développer ci-dessous à partir de l'expérience d'un APTIC.

POURQUOI PROGRAMMER AUJOURD'HUI ?

On peut penser trouver pour tout problème posé une solution logicielle existante. S'il est inutile de fabriquer dans son coin un texteur, il n'empêche que l'on est loin d'avoir pour tout problème une solution. Programmer reste donc parfois utile pour créer rapidement une application. C'est d'ailleurs pour cela que nombre de personnes utilisent des outils comme EXCEL ou que de nombreux logiciels proposent la possibilité de créer ses macro-instructions.

Mais au niveau des lycées (et des étudiants), ce n'est pas l'aspect utilitaire qui me semble primordial, mais celui de la culture et celui de l'appropriation d'un outil tant pour la machine physique que la machine logique.

Aussi, le fait de programmer l'ordinateur doit conduire à la réalisation d'une application qui fonctionne mais peut aussi permettre de **démystifier une machine** qui peut commettre les erreurs que l'on a pas su éviter dans sa programmation...

Avant d'en arriver là il aura fallu à l'élève bien des efforts pour organiser ses idées, exprimer son projet et comprendre comment l'assemblage des mots d'une grammaire parfois obscure, l'agencement des différents objets utilisés pourra conduire au nirvana : qui n'a pas programmé ne sait pas le plaisir physique et intellectuel de voir enfin

tourner son programme... Mais c'est en voyant, même dans le virtuel de l'image informatique, la concrétisation de ses "rêves" qu'il comprendra qu'un ordinateur peut résoudre des problèmes. Une approche expérimentale, c'est-à-dire constituée d'essais de différentes méthodes de résolutions, renforce un caractère concret de la démarche, allant quelque peu contre les méthodes pédagogiques actuelles souvent très intellectuelles. Des notions abstraites de mathématiques trouvent ainsi une application pratique.

Il me semble donc clair qu'il y a encore, pour des élèves **volontaires**, une place pour la programmation, à condition de trouver un langage efficace et simple permettant de manipuler les objets des interfaces graphiques actuelles (Mac OS ou Windows), qui permette l'utilisation d'images et de sons de façon aussi simple que possible, et dont la puissance soit suffisante pour mener à bien des projets éventuellement complexes.

J'ai dans le cadre d'un APTIC essayé de mener à bien une expérience menée selon ces critères.

COMMENT AU TRAVERS D'UNE EXPÉRIENCE D'APTIC ?

Le langage utilisé est Hypertalk lié à Hypercard, langage ne fonctionnant que sur Macintosh.

Le choix Macintosh est ancien et peu conforme aux canons habituels. Le débat avec un collègue engagé dans l'option de seconde montre qu'il n'est pas sûr que l'on puisse trouver un langage de nature comparable, et donc utilisable dans le cadre défini, sous Windows. Mais nous ne sommes pas là pour déclencher ou entretenir des guerres de religions... et Hypercard existera un jour sur PC sous une forme probablement plus élaborée.

Hypertalk est un interpréteur qui se prête bien aux objectifs assignés. L'abord est simple. Le dessin point par point permet l'introduction et facilite la maîtrise de la machine, première étape inévitable avec des élèves qui, en terminale, n'ont pas souvenir, pour 80 % d'entre eux, d'avoir vu ou touché un ordinateur (sauf ceux qui viennent de STT). Un **dessin animé** permet de montrer la structure de cartes (écrans) communes à un fond : le corps d'un bonhomme est inclus dans le fond et ses membres dans des positions différentes dans les cartes. Un ordre simple oblige la machine à aller de carte en carte quand rien ne se passe. C'est le premier script de 3 lignes.

Viendront ensuite les **boutons**, faciles à créer et qui permettent d'aborder les notions de programmation événementielle. La création d'un petit piano virtuel illustre le propos.

Précisons ce qu'est cette programmation dans ce cas :

Un bouton est créé par un article de menu. Un double-clic sur le bouton permet de lui donner un nom et de lui imposer des contraintes de présentation (il sera rectangulaire, à coins ombrés, son nom caché...) .

Il contient un script (programme) qui peut réagir aux actions, en particulier au clic (ou *mouseup*) soit par exemple :

```
on mouseup          ("on" définit la procédure qui intercepte l'action mouseup)
    play flute "a"
end mouseup         ("end" termine la procédure mouseup)
```

Alors un clic sur le bouton provoque le son de la note La (A en anglais) avec comme instrument (son) la flûte.

Mais on peut faire plus fort : au lieu de jouer la note, on peut jouer le nom du bouton qui sera nommé a :

```
on mouseup
    get the short name of the target (récupérer le nom de la cible c'est-à-dire le nom du
    bouton)
    play flute it (it contient le résultat de la récupération)
end mouseup
```

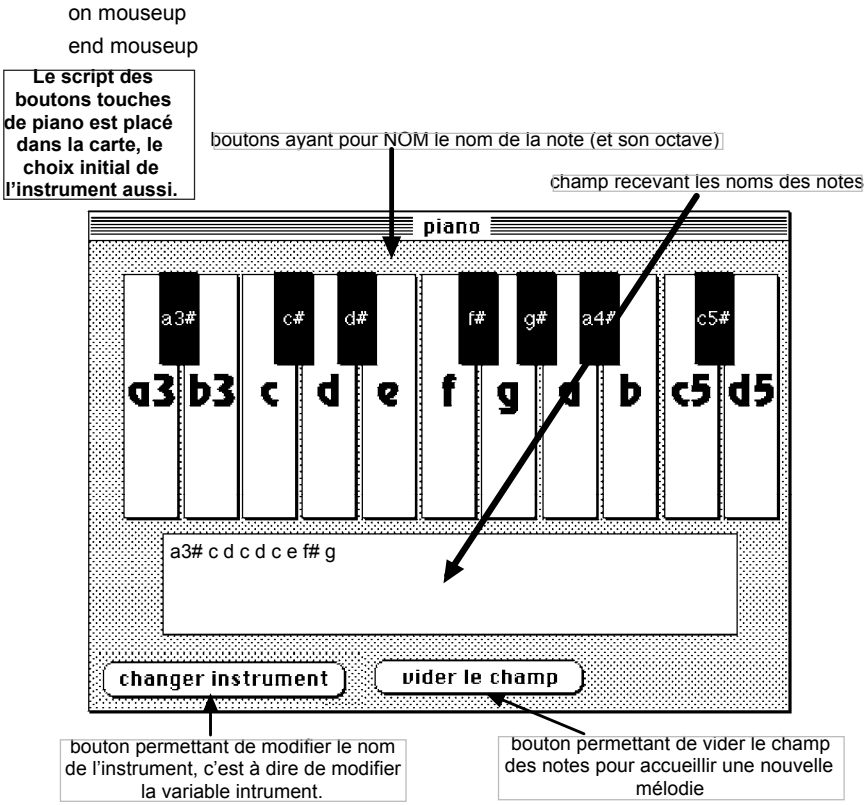
que l'on peut aussi écrire

```
on mouseup
    play flute the short name of the target
end mouseup
```

Et encore plus fort ? Le script peut être placé dans la carte car l'action mouseup est aussi envoyée à la carte qui peut donc réagir en testant au préalable s'il s'agit d'un bouton :

```
on mouseup
    get the name of the target     (récupérer le nom complet de la cible c'est-à-dire le
    nom du bouton)
    if there is "button" into it  (seuls les noms des boutons seront joués)
    then
    play flute the short name of the target
  end if
end mouseup
```

Pourquoi plus fort ? Tout simplement parce qu'en copiant le bouton puis en changeant de nom, on obtient la deuxième touche puis la troisième... Et tous ces boutons ne contiennent même pas de script ou du moins un script automatique avec seulement :



Exemple de réalisation plus complet
(a3 pour La *3° octave, a seul car 4° octave par défaut, a5 pour la 5° octave)*

Ainsi, pas à pas, sont mis en place les différents objets manipulés par Hypertalk, les scripts (ou programmes) nécessaires au fonctionnement. L'ensemble est plus complexe qu'il ne parait mais est tout à fait possible pour des élèves « normaux » de terminales littéraires, ES, STT, STI-électronique.

Il faudra alors ajouter les notions de boucles, de branchements conditionnels et de procédure. Le dessin programmé permet l'abord en douceur de notions qui restent complexes, avec la réalisation, entre

autres, d'une cible ou d'une rosace (création d'une procédure cercle et son utilisation).

A partir de là, s'impose l'idée de différenciation du travail des élèves : à chacun son projet.

Comme d'habitude, les idées ne sont pas légion. Il faut en fournir...

Voici quelques uns des projets développés dans l'année, illustrant les limites de l'imagination du professeur et des élèves :

- horloge analogique
- QCM
- jeu du pendu
- jeu du nombre mystérieux
- conjugaison des verbes du premier groupe
- jeu du morpion
- jeu de mots en désordre
- jeu de lettres mélangées pour trouver un ensemble de mots
- nombre de secondes avant l'an 2000

L'horloge analogique est un projet simple en apparence mais problématique pour les "non-matheux". Il s'agit de dessiner un cercle plein puis les aiguilles sous la forme de traits, ces aiguilles étant évidemment placées de façon à donner l'heure.

Une fois l'heure récupérée, de "délicats" calculs, avec d'horribles sinus et cosinus, permettent de déterminer les coordonnées des aiguilles. La mise à jour se fait en permanence quand l'ordinateur ne travaille pas.

Le jeu du nombre mystérieux est simple : il permet donc la mise en place d'une structure de jeu qui pourra ensuite être réutilisée. En effet, le jeu doit être initialisé, puis les réponses demandées à l'utilisateur et analysées par l'ordinateur. L'analyse inclut le test sur la réponse, le test sur le dépassement du nombre d'essais. Il doit permettre de rejouer. Sa construction pourra être naturellement récursive.

Explicitons un peu la structure :

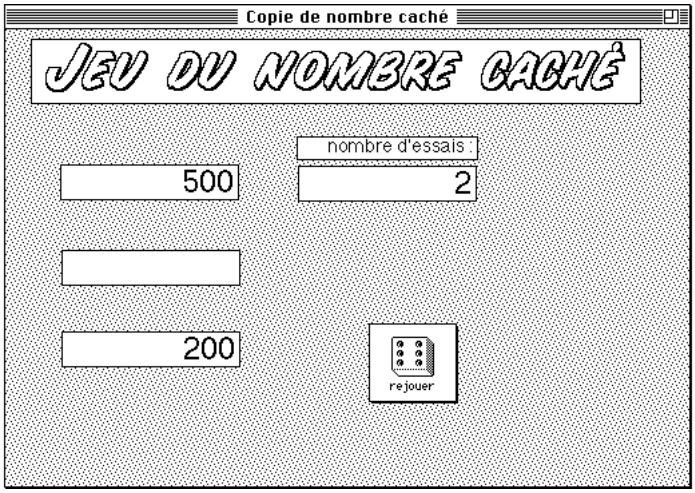
- un bouton servira au démarrage (qui peut être rendu automatique très simplement) :

```
on mouseup
    jeu random(1000),1      (random(1000) donnera le nombre à chercher et 1 permet
    de compter les essais)
end mouseup
```

- un script, par exemple placé dans la carte, définit jeu. (il est ici restreint à l'essentiel : quelques ajouts sont nécessaires pour les affichages)

```
on jeu nmyst, nessai        (deux paramètres...)
    if nessai > 10  (gestion du dépassement du nombre maximum de coups)
    then
            answer "perdu !!! le nombre était : "&nmyst
            exit jeu  (on arrête le jeu)
    end if

ask "Quel nombre ? : "    (acquisition au clavier de la réponse)
put it into n             (la réponse est placée dans it par la machine puis dans n par le
programme)
if n = nmyst      (test)
then   (cas gagné)
    answer "bravo"
    exit jeu  (on arrête le jeu)
else   (cas perdu puis différenciation si trop grand ou trop petit)
    if n > nmyst
    then
            answer "trop grand"
            jeu nmyst, nessai+1  (on rejoue avec un essai en plus)
    else
            answer "trop petit"
            jeu nmyst, nessai+1  (on rejoue avec un essai en plus)
            end if
    end if
end jeu
```



On peut alors concevoir le jeu du pendu sur le même modèle, s'ajoutant le dessin du pendu et une analyse plus fine. Ce jeu se prête bien à des ajouts complexes : saisie au clavier ou par clic ou par dialogue..., affichage des lettres jouées, ...

La conjugaison des verbes du premier groupe est un sujet passionnant pour (re)découvrir les douces joies du français. D'un premier abord le sujet semble simple puisque les terminaisons sont standardisées et le groupe censé être régulier... Mais voilà qu'arrive le verbe aller avec ses irrégularités et la bonne idée de commencer par une voyelle... Mais combien d'autres perfidies vont devoir être intégrées : épeler - appeler, jeter-peler, aboyer, etc. sans oublier les verbes transitifs...

EN GUISE DE CONCLUSION...

La confrontation lors de la commission examinant les différentes réalisations des APTIC académiques montre que cette façon d'envisager le travail n'est pas vraiment conforme aux objectifs, au demeurant fort flous, des APTIC. On retrouve de vieux relents de l'option informatique dans ma façon d'aborder le problème...

Il faut bien dire que la notion de projet réalisé en groupe n'est pas facile à mener : l'architecte risque d'être le seul à connaître l'ensemble de la structure et risque aussi de faire une grande partie du travail. C'est certainement pour cela qu'il m'est paru plus naturel de centrer l'APTIC sur l'apprentissage de l'algorithmique au travers d'un outil logiciel suffisamment puissant pour s'intéresser au travail sur la logique de la programmation sans pour autant négliger la présentation et l'efficacité. Il me semble que la nouvelle "option" informatique, plutôt que d'élaborer un programme fourre-tout, aurait pu être construite en prenant en compte de façon plus explicite l'intérêt de la programmation (l'algorithmique) pour les lycéens sans les différencier en « scientifiques » et « littéraires ». On pourra toujours dire que cette activité doit attendre l'université... mais vu le nombre de bidouilleurs sans formation on peut considérer que canaliser les tentatives n'est pas forcément inutile.

Jean-Noël JOFFIN

NB : si vous êtes intéressés par divers petits programmes développés dans le cadre de cet APTIC :

- laissez un message sur jnjoffin@ac-idf.jussieu.fr pour un envoi par internet
- ou envoyer un chèque de 20 FF ou l'équivalent en timbres à Jean-Noël Joffin - Lycée Paul Eluard - 93206 Saint--Denis Cedex pour recevoir la disquette correspondante. La somme réclamée couvre les frais d'envois, de disquette et d'enveloppe.

*** NDLR :** cette affirmation pourrait être quelque peu nuancée depuis le rétablissement de l'option informatique.