Programmation objet et tortues

http://revue.sesamath.net/spip.php?article853
patrick.raffinat@univ-pau.fr

A) Introduction

Comme je l'ai dit dans l'article, je ne m'intéresse pas ici à une hypothétique formation des élèves, mais à donner une culture générale aux enseignants en programmation objet.

En tant qu'enseignant d'informatique en IUT, je ne pense pas qu'elle soit nécessaire pour enseigner Scratch, mais comme tout le monde n'a pas la même opinion, j'ai décidé d'utiliser les tortues d'Alain Busser et de Florian Tobé pour vulgariser les concepts fondamentaux de la programmation objet.

La programmation objet s'appuie sur une vision du monde que tout un chacun met en pratique dans la vie courante : classer des objets (des animaux, des végétaux…) par catégories (des voitures, des tortues…). Chaque catégorie (appelée **classe** en informatique) a des caractéristiques (appelées **propriétés** ou attributs) et des actions (appelés **méthodes**) communes à tous les individus (appelés **objets**) qui en font partie :

- toutes les voitures ont un moteur, des roues, des portes comme propriétés ; elles peuvent effectuer des actions telles que démarrer (pas toujours hélas), rouler, klaxonner...
- toutes les tortues ont des yeux, des pattes, une carapace comme propriétés ; elles peuvent effectuer des actions telles qu'avancer (à la vitesse d'un escargot), manger des salades...

En tant qu'utilisateurs d'outils informatiques, vous utilisez sans le savoir des objets, par exemple des fenêtres qui ont comme propriétés une taille, une position, un bandeau, un contenu… Ces fenêtres répondent à des actions telles que déplacer, agrandir, iconiser...

B) Les tortues en programmation objet

Après cet indispensable préambule, revenons à nos moutons ou, plutôt, à nos tortues. Les tortues informatiques ont comme propriétés une abscisse, une ordonnée, une direction, un stylo multicolore à plusieurs billes… Elles peuvent effectuer des actions telles qu'avancer, reculer, tourner à gauche ou à droite, lever ou baisser le stylo...

Tout cela est bien beau me direz-vous, mais pourquoi se fatiguer à faire de la programmation objet alors qu'on s'en passe très bien quand on programme en langage Logo (ou dans un de ses nombreux descendants) ?

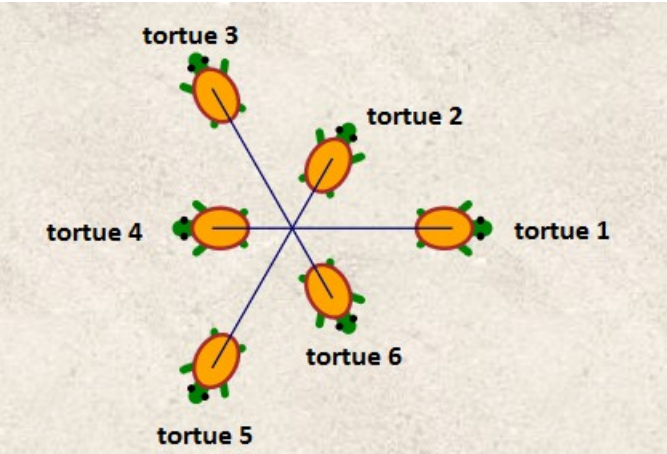
La réponse est simple : la programmation objet devient très utile à partir du moment où on veut manipuler plusieurs tortues. Mais pour ce qui est d'expliquer quelle utilité il y a de manipuler plusieurs tortues, ce n'est pas à moi de le faire et je laisse ce travail à Alain et à Florian.

Avant de passer à la pratique, je tiens à préciser qu'il y a deux niveaux de programmation en programmation objet :

- concevoir et programmer une ou plusieurs classes, ce qui est généralement difficile et nécessite donc une bonne maîtrise de la programmation objet.
- manipuler les objets d'une classe par programmation (ou avec une interface graphique), ce qui est plus simple et ne nécessite que des connaissances sommaires sur les concepts de la programmation objet.

C) Exemple

Mon objectif n'étant pas de former des experts, je me contenterai d'un petit exemple avec 6 tortues :



Voici le code correspondant, que je vais commenter :

```
effaceDessin();
// creation des 6 tortues, qu'on oriente tous les 60 degrés
tortues=[];      // container (facultatif) permettant de stocker les tortues
for(k=1; k<=6; k=k+1) {
        tortues[k] = new Tortue(k);    // création de la tortue de « dossard » k
        tortues[k].tg(60*(k-1));
}
// on fait avancer différemment les tortues paires et les tortues impaires
for(k=1; k<=5; k=k+2) {
        tortues[k].av(80);
        tortues[k+1].av(40);
}
// on affiche les propriétés de la tortue 2
Println("abscisse de la tortue 2 : " + tortues[2].x);
Println("ordonnee de la tortue 2 : " + tortues[2].y);
```

L'instruction new Tortue(k) crée une tortue en lui attribuant un « numéro de dossard » (k) dont l'utilité n'apparaît pas ici : en fait, Alain et Florian stockent leurs tortues dans un tableau interne nommé totos [1], qu'ils utilisent pour exécuter leurs programmes Blockly. Donc, le tableau tortues que j'ai créé ici ne sert qu'à rendre mon code plus compréhensible.

Les notations objets sont faciles à comprendre : le nom d'un objet (donc d'une tortue) est suivi d'un point, puis d'une propriété (x, y …) ou d'une méthode (av, tg...).

1 Si vous ne créez aucune tortue, vous verrez néanmoins une tortue (nommée totos[1]) à l'écran car Alain et Florian en créent une par défaut.

D) Compléments sur le vocabulaire objet

Vous rencontrerez peut-être des termes « barbares » lors de lectures ou de formations. Je les explique en évitant de les formaliser.

Propriétés publiques ou privées

Il n'y a aucun risque de faire afficher l'abscisse de la tortue 2 avec l'instruction Println(tortues[2].x). Mais qu'en serait-il si l'on voulait déplacer la tortue avec l'instruction tortues[2].x += 50 ?

Si vous essayez, vous ne verrez pas la tortue se déplacer. Pire, vous risquez ainsi de perturber l'exécution du programme s'il y a ensuite d'autres instructions puisqu'il n'y aura plus de cohérence entre la position physique de la tortue à l'écran et sa propriété « x ». Donc, je vous recommande de ne déplacer une tortue qu'avec les méthodes prévues pour : avancer, reculer, teleporter...

Pour que ce risque d'incohérence ne puisse arriver, de nombreux langages (pas Javascript à ma connaissance) permettent aux concepteurs de déclarer la propriété « x » comme **privée** : en d'autres termes, un programmeur utilisant la tortue tortues[2] ne pourra plus accéder directement à sa propriété « x » et devra passer par les méthodes prévues pour.

Propriétés d'instance ou de classe

Supposons que les tortues aient aussi comme propriété un « compte en salades », ce qui après tout n'est pas plus farfelu que de leur faire tenir un stylo. Pour le définir, il y a deux options :

- soit comme une propriété d'instance, ce qui signifie que chaque tortue a un compte en salades.
- soit comme une propriété de classe, ce qui signifie qu'il n'y a qu'un seul compte en salades commun à toutes les tortues.

Héritage

L'héritage est peut-être le concept clé qui explique le succès de la programmation objet : en effet, il permet de définir une classe à partir d'une autre (par exemple TortueEvoluee à partir de Tortue) en ne précisant que les propriétés supplémentaires (compte en salades) et les méthodes supplémentaires (manger ou engranger des salades).

L'héritage permet aussi de redéfinir certaines méthodes : par exemple, on peut envisager qu'une tortue évoluée avance en faisant des pointillés. Si rien n'est indiqué dans la définition de la classe TortueEvoluee, elle avancera comme n'importe quelle tortue avec un trait continu.