Apprendre l'informatique avec Javascool à travers la programmation de petits jeux

Benoît Crespin

Dernière mise à jour : 11 mars 2014

Version en ligne :
`http://javascool.gforge.inria.fr/documents/crespin-etal.pdf`

Table des matières

1 Préambule

1.1 Avertissement

Ce document est encore en développement, certaines parties sont en cours de rédaction. Toute remarque ou suggestion (ou correction des exercices !) est la bienvenue : `benoit.crespin@unilim.fr`

1.2 Qu'est-ce qu'un langage de programmation ?

Un langage de programmation peut être vu comme un moyen permettant à un programmeur humain de faire exécuter à un ordinateur des tâches complexes. Il s'agit donc, un peu comme une langue étrangère, d'un ensemble d'éléments (vocabulaire, syntaxe, grammaire, etc.) implantés dans l'ordinateur et que le programmeur doit apprendre à maîtriser pour arriver à ses fins. Accompagnant l'évolution matérielle des ordinateurs depuis les années 1950, il existe à présent des milliers de langages de programmation [1], en général créés pour répondre à des besoins précis et bénéficiant d'une audience internationale plus ou moins grande. On peut citer comme principaux langages largement utilisés actuellement : Java, C/C++, Python et Visual Basic.

1.2.1 Langages compilés vs interprétés

Une distinction usuelle entre les différents langages de programmation concerne l'exécution des programmes par l'ordinateur. Un programme écrit dans un langage **compilé** est transformé par un logiciel spécifique, le **compilateur**, en un ensemble d'instructions directement exécutables par la machine. Ce type de langage requiert donc une **phase de compilation** pour réaliser cette transformation avant de pouvoir exécuter le programme. A l'inverse, avec un langage **interprété** le programme est exécuté directement par la machine en transformant les instructions **à la volée** : aucune phase de compilation n'est nécessaire.

Cette différence essentielle fait que les langages compilés sont généralement préférés par les informaticiens, pour plusieurs raisons :

- lors de la phase de compilation, il est possible de détecter certaines erreurs dans le programme afin d'éviter l'exécution d'un programme erroné. Tant que ces erreurs ne sont pas corrigées l'exécution du programme est impossible.
- Si la compilation réussit, les instructions seront exécutées plus rapidement par la machine que le programme équivalent écrit dans un langage interprété.

Pourtant les choses ne sont pas forcément aussi simples, et de nombreux autres aspects sont à considérer concernant les différences entre langages. Le langage Java par exemple, que nous allons utiliser avec Javascool, peut être vu à la fois comme un langage compilé et interprété. Pour plus de détails sur le sujet consulter : `http://fr.wikipedia.org/wiki/Java_(langage)`

1.2.2 Javascool

Comme on peut le lire sur son site Web [2], **Java's Cool (alias Javascool) est un logiciel conçu pour l'apprentissage des bases de la programmation**. Il se base sur le logiciel Java, qui est le seul élément qui doit être installé sur la machine pour permettre à Javascool de fonctionner. Les avantages de Javascool pour l'apprentissage de l'algorithmique et de la programmation sont multiples :

- L'utilisation de la syntaxe de Java, elle-même proche de la syntaxe de nombreux langages tels que le C/C++, permet de donner aux élèves des bases qu'ils retrouveront fréquemment s'ils poursuivent des études dans le domaine Informatique.
- Cette syntaxe est également assez proche des *méta-langages* (ou "langages algorithmiques") traditionnellement utilisés pour enseigner l'algorithmique, par conséquent la traduction en langage Javascool d'un algorithme exprimé en méta-langage est plus simple qu'avec d'autres langages, même si ce document montre justement qu'il y a quelques pièges à éviter...
- Java étant multi-plateformes, Javascool peut fonctionner indifféremment sous Linux, Windows ou Mac

1. `http://en.wikipedia.org/wiki/Programming_language`
2. `http://javascool.gforge.inria.fr/`

– C'est un logiciel "tout-en-un", intégrant à la fois un éditeur pour écrire ses programmes et une fenêtre pour les exécuter
– Javascool est utilisé par de nombreux enseignants, et commence à proposer un nombre important de ressources pédagogiques accessibles sur son site web

On peut néanmoins lui trouver quelques inconvénients :

– Les messages d'erreurs ne sont pas toujours "parlants", et le logiciel ne propose pas réellement de fonctionnalités pour faciliter l'élimination des erreurs (ce qui est parfois vu comme un avantage étant donné que cela oblige à se poser plus de questions)
– Javascool contient des fonctionnalités permettant de simplifier l'apprentissage de la programmation en "cachant" certaines fonctionnalités du langage Java ; cela peut entraîner une confusion chez certains élèves lorsqu'ils sont confrontés plus tard dans leur cursus à des langages réellement utilisés dans le monde industriel ou académique.
– Java reste un langage relativement peu rapide par rapport à d'autres, ce qui n'empêche quand même pas de l'utiliser même pour résoudre des problèmes complexes.

Il n'en reste pas moins que tout langage de programmation a ses avantages et inconvénients, et le choix n'est pas facile parmi les milliers de langages utilisés actuellement sur la planète. Enseigner comment traduire un algorithme dans un langage relativement simple comme Javascool permet de toute à façon de préparer les élèves à apprendre plus tard comment passer facilement d'un langage de programmation à un autre et comment choisir le langage le plus approprié selon le type d'application envisagé, comme tout bon programmeur qui se respecte.

On peut aussi considérer Javascool comme un tremplin vers des plateformes de programmation professionnelles comme Eclipse [3] ou Netbeans [4].

1.3 Objectifs et structure de ce support

Ce document a pour objectif principal de former les enseignants pour leur permettre d'aider ensuite leurs élèves à utiliser Javascool, ce qui inclut :

– des notions basiques et avancées liées à la traduction d'un algorithme en "langage Javascool" (*ie Java simplifié)*
– *une description des outils de débogage et de mise au point des programmes*

On trouvera notamment ici des explications complémentaires sur certaines activités de base déjà proposées dans Javascool, d'autres idées d'activités à mener avec les élèves, et, dans certains cas, des notions liées à d'autres langages de programmation pour ceux qui souhaitent éventuellement aborder l'algorithmique sous d'autres formes. En revanche, de nombreuses activités proposées dans Javascool ne seront que peu abordées ici, mais elles sont en général suffisamment bien documentées.

La structure de ce document est la suivante :

– La section 2 donne les éléments de base permettant la prise en main de Javascool, pour aider à comprendre les grands principes et comment démarrer.
– La section 3 est une sorte de guide pour découvrir les ingrédients des algorithmes, c'est-à-dire la base qui permet de passer des algorithmes à la programmation.
– La section 4 est une analyse didactique des erreurs de compilation et d'exécution les plus courantes, c'est donc un contenu essentiel pour ne plus être "victime" des bugs.
– Les sections 5 et 6 permettent d'approfondir les éléments essentiels de programmation correspondant au programme ISN de l'enseignement de spécialité de TS.
– Enfin les sections 7 et 8 offrent la possibilité d'aller plus loin en explorant les structures usuelles de données de l'informatique au delà du programme ISN.

1.4 Fil rouge : programmer des jeux

Les livres ou cours de programmation illustrent souvent les notions abordées par une application complète, permettant de montrer l'intérêt de ces notions dans un cadre concret. Ici on tentera de mettre en pratique ces différents éléments à travers le développement de petits jeux. Ce choix d'une application ludique a de nombreux avantages : l'expérience du développement d'un jeu est souvent très motivante et permet de faire passer plus facilement concepts avancés et liens avec des domaines tels que

3. `http://www.eclipse.org/`
4. `http://netbeans.org/`

les mathématiques ou la physique. Programmer un jeu de type “casse-briques” par exemple implique de comprendre les notions de vélocité et de collision. On peut retrouver en fait dans les jeux un panorama complet des notions abordées en informatique :

- Algorithmique classique (comment fonctionne le jeu, que se passe-t-il si le joueur entre telle ou telle valeur)
- Structures de données (comment stocker et accéder efficacement aux données du jeu telles que les scores ou les positions des joueurs)
- Affichage, graphismes 2D ou 3D (comment afficher et éventuellement dessiner des données à l'écran)
- Son (comment générer des sons au cours du jeu)
- Aide à la décision (comment programmer un adversaire performant face au joueur humain)
- Réseaux (comment faire communiquer des joueurs au travers du jeu)
- etc.

1.5 Versions de Javascool

Les exemples donnés dans ce document utilisent la version 4 de Javascool, téléchargée le 29/02/2012 (5.9 Mo). Il est possible que certains ne fonctionnent pas avec des versions différentes (notamment des versions antérieures), ou se comportent différemment par rapport aux résultats attendus.

1.6 Pré-requis : des notions d'anglais

Cela peut paraître étonnant, mais un aspect parfois rebutant pour les élèves par rapport à la pratique de la programmation tient à ce que ce domaine est fortement lié, pour des raisons historiques, à la langue anglaise. Les mots-clefs employés par Javascool, comme dans la plupart des langages de programmation, sont en anglais et sont évidemment plus faciles à comprendre pour les élèves ayant une bonne maîtrise de cette langue. Par exemple, le concept de “programme principal” est immédiatement compréhensible pour qui sait que le terme anglo-saxon “main” se traduit par “principal” en français.

De même, les messages d'erreur sont parfois en anglais et peuvent apparaître très mystérieux pour les élèves. On s'efforcera dans ces documents d'apporter des précisions sur le sens des messages rencontrés le plus fréquemment mais il est important de faire passer auprès des élèves que la maîtrise de l'anglais, en informatique comme pour toute discipline scientifique, devient de plus en plus nécessaire.

2 Principes de base

2.1 La philosophie Javascool

Javascool est plus qu'un langage de programmation : comme mentionné plus haut c'est un logiciel complet intégrant un éditeur pour écrire ses programmes et une fenêtre pour les exécuter. De façon plus générale, Javascool se veut un support permettant d'apprendre à la fois l'algorithmique et sa traduction immédiate, en permettant au professeur de faciliter la vie des élèves grâce à la possibilité d'utiliser des “proglets” existants ou qu'il programmera lui-même. Une “proglet” est un ensemble de fonctionnalités écrites en Javascool et de documents expliquant comment les utiliser dans le cadre d'une activité, en passant par une interface graphique ou en écrivant du code Javascool[5].

Ainsi, on peut éventuellement proposer à l'élève d'utiliser une fonctionnalité (par exemple la fonction **load** permettant de charger une image en mémoire) en lui cachant les détails concernant la façon dont cette fonctionnalité est programmée.

Cette “philosophie” Javascool permet de simplifier la prise en main par l'élève de certains aspects techniques pour qu'il puisse se concentrer sur les aspects purement algorithmiques. Pour reprendre l'exemple de la manipulation d'images, l'élève pourra ainsi manipuler une image comme un tableau à deux dimensions dont les cases représentent les intensités des pixels, sans avoir à comprendre comment sont réellement codées les images au format JPG ou autre.

Nous nous concentrons dans ce document sur les aspects techniques de Javascool : la syntaxe du langage (similaire à celle du langage Java), les fonctionnalités du logiciel, les causes les plus fréquentes d'erreurs de traduction entre version algorithmique et version Javascool d'un programme. Les aspects plus

5. Les “educlets” sont des “proglets” restreintes à l'utilisation d'une interface graphique