Utiliser des proglets pour manipuler des objets numériques

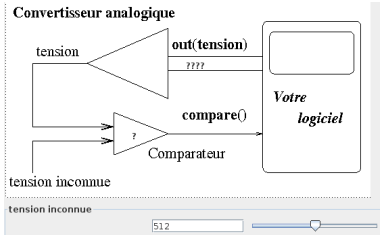
Thierry Viéville, Florian Dufour, Philippe Vienne, Françoise Tort

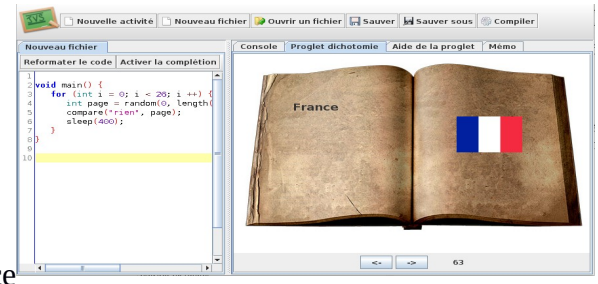
Utiliser des proglets pour manipuler des objets numériques. thierry.vieville@inria.fr

Apprendre l'Informatique et les Sciences du Numérique c'est au delà de cliquer sur les objets numériques, pouvoir soulever le capot et les paramétrer, programmer, changer, etc. Mais la *mise en œuvre* d'une activité de programmation est souvent très lourde, surtout si l'apprenant doit d'abord installer, s'interfacer avec les modules à utiliser. Le risque que le temps soit perdu à la mise en place, l'appropriation du contexte ou à des « pannes » est élevé. Pire, le but de l'activité risque d'être noyé dans l'accessoire. Le diable est dans les détails, voyons comment l'exorciser.

Une notion permet de mettre en place de manière relativement optimale une activité ISN, c'est la notion de « proglet ». Une «proglet» est une petite applette qui permet de s'initier à la programmation de manière ludique. Elle se présente sous forme d'une interface mettant en scène le/les concept(s) clé(s) à manipuler, sous forme visuelle et/ou sonore, à la fois pour mieux comprendre l'objet numérique et pour offrir une démarche expérimentale concrète. Une «proglet» est donc un objet numérique que l'on manipule en le programmant (en non en cliquant).

Exemples de proglet : Prenons le principe algorithmique abstrait probablement le plus élémentaire, le mécanisme de dichotomie (on ne le rappelle pas ici). Il peut se décliner par exemple pour trouver le zéro d'une fonction réelle continue monotone dans un intervalle, réaliser un convertisseur analogique-digital ou trouver un mot dans un dictionnaire : trois contextes très différents pour un même principe algorithmique.





Des proglets permettent d'expérimenter ce principe dans les contextes cités. Pour cela un panneau graphique minimal est programmé. Par exemple, à gauche, la proglet convertisseur est un dessin qui montre le schéma électrique les deux fonctions disponibles *out(double valeur)* pour sortir une valeur analogique à comparer et *int compare()* qui lit la valeur *1* ou *-1* sur le comparateur. Sur ce schéma, les valeurs électriques s'affichent au fil du programme de l'apprenant. Bien entendu, c'est une simulation et la tension électrique est choisie à la main au sud du panneau graphique, pour tester le programme dont le déroulement sera observable.

À droite, l'environnement complet d'une autre proglet est montré, mettant en œuvre un graphisme où on « tourne » les pages d'un livre, pour y rechercher un pays : panneau graphique, éditeur de code, document sur la proglet et texte de l'activité sont présentés dans le même environnement, c'est une jarre Java qui se lance sur un simple clic Web sur toutes les machines, rendant immédiat le démarrage et la prise en main de l'activité.

On a ainsi atteint les objectifs fixés : environnement convivial, ergonomie naturelle, documentation intégrée. Il n'y a plus qu'à se concentrer sur l'essentiel : le savoir et la pédagogie.

Spécification et mise en oeuvre d'une proglet : Pour spécifier une proglet il faut définir :
- quelques méta-données documentaires, pour que le grains soit moissonable sur le Web;

- la documentation : présentation du contexte, description des fonctions disponibles, script de l'activité proposées, liens vers des ressources de référence, ouverture du code source aussi;
- le code du *Panel* (le petit panneau graphique qui sera piloté), et le code des *Functions*, qui implémente les routines qui pourront être appellées, de manière modulaire, et réutilisable ;

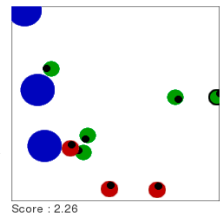
sans plus. Le framework intègre ces ingrédients et produit la plateforme à utiliser, avec le compilateur embarqué. Dans sa version initiale sous le nom de www.javascool.fr, une instanciation Java est mise à disposition avec désormais une 20taine de proglets (algorithmes mathématiques, manipulation d'images, de sons, de graphes, simulation de la bonne vieille torttue logo, activité autour d'une interface série, du cryptage RSA, du dessin pixelique, ..) dont plus d'un tiers crés par les enseignants ISN eux-mêmes. Tous les exercices du livre *Informatique et Sciences du Numérique. Spécialité ISN en Terminale S. Cap 11. (2012) Gilles Dowek et al.* sont disponibles dans une proglet spécifique et tous les exercices du programme d'algorithmique du programme de 2nd aussi.

Noter que nous parlons pas du langage. Peu importe le dialecte, pourvu qu'il implémente la base impérative. Le langage javascool est un sous langage de Java où la couche objet est masquée de la manière la plus simple du monde, il suffit de rajouter les imports idoines et la ligne «*public class XXX { ..»* pour en faire du Java. L'apprenant javascool va donc pouvoir passer *immédiatement* à un environnement professionnel Java ou C++, etc.. quand il sera prêt.

Conclusion et perspective.

Si cette simplification a eu son succès (javascool est une des deux grandes plateformes avec Python à avoir aider à la formation des enseignants ISN et compte probablement largement plus que 10^3 utilisateurs), le concept peut-être encore affiné et permettre de se tourner vers des activités encore plus intéressantes, profitant de l'arrivée de HTML5/JS et des frameworks associés (ce qui du reste tourne sur toutes les plateformes tablettes ou smartphone inclus). Avec ces « grains logiciels 3.0 » (nommés ainsi car ils sont conçus pour être des objets du Web3.0 avec des méta-données sémantiques, et des interfaces permettant d'y accéder à travers des Web service), on se propose de créer le même paradigme mais en tant que composant d'une page Web. L'apprenant est donc devant une activité de création de contenu pluri-média, qui comporte des éléments logiciels, la encore proposés packagés avec une activité à dérouler pour programmer la mise en fonctionnement des ingrédients proposés.

Sous le terme de carpasinivore, un contenu expérimental a été créé où des petits êtres minimalistes sont dans un environnement de survie et l'apprenant au lieu de jouer en cliquant doit programmer un « bot » qui une fois lancé dans le jeu ne doit sa survie qu'à la qualité de son algorithme. La sur-simplification permet à la fois l'exécution raisonnable du grain, sa prise en main rapide, mais surtout . . de soulever le capot. Une fois l'activité achevée et complètement soldée, l'apprenant va naturellement se dire : « ah tiens, mais comment il a été fait ce grain ? » ou « comment je pourrais le démonter/changer/etc.. ? ». Bonne nouvelle, le grain est simple, accessible et prêt à être « détourné ». L'utilisation de la plateforme github se prête naturellement à la création de ses branches, où chacune et chacun peut créer ses propres variantes.



On crée donc un chemin entre des activités complètement encadrées où il faut copier/coller des bouts de codes pour en observer le fonctionnement, puis remplir des codes « à trou », implémenter un principe algorithmique bien décrit, concevoir l'algorithme jusqu'à . . créer son propre objet numérique.