Applications

Éducation : EToys (Squeak pour OLPC), SqueakBot, BotsInc, Scratch, iStoa.net, Edusim, DrGeoII…

Multimédia : Sophie, OpenCroquet, Plopp…

Développement web : Seaside, Aida, Komanche, Swazoo…

Gestion de la persistance : base de données objets (Magma, GemStone), relationnelles (MySQL, PostgreSQL), mapping objet relationnel (Glorp…).

Etoys et DrGeo sur un OLPC

Glossaire

Image mémoire : L'environnement Smalltalk contient un entrepôt persistant d'objets, l'image. Elle contient le code de l'application (les classes et les méthodes), les objets qui constituent l'état de l'application et peut même contenir l'environnement de programmation pour inspecter et déboguer le programme pendant qu'il s'exécute.

Machine virtuelle : Une machine virtuelle est un programme qui est capable d'exécuter d'autres programmes. Cela permet de faciliter la portabilité des applications que l'on développe.

Réflexion : On dit qu'un langage est réflexif s'il dispose de mécanismes permettant d'inspecter et de modifier du code pendant l'exécution d'un programme.

Typage dynamique : Certains langages forcent le développeur à indiquer le type de chaque variable (entier, chaîne de caractères…). On appelle cela le typage statique. En typage dynamique, le développeur ne contraint pas ses variables à un type particulier.

Ouvrages

- Nombreux livres téléchargeables gratuitement
  `http://stephane.ducasse.free.fr/Books.html`
- Smalltalk en général
  Smalltalk with Style, Edward Klimas, Suzanne Skublics, David A. Thomas, Prentice Hall, 1996, gratuit ;
  Smalltalk by Example: the Developer's Guide, Alec Sharp, McGraw Hill Text, 1997, gratuit.
- Squeak en particulier
  Squeak by Example, Andrew P. Black, Stéphane Ducasse, Oscar Nierstrasz and Damien Pollet, Square Bracket Associates, 2007, libre et gratuit, traduction française 2008 ;
  Squeak, Xavier Briffault et Stéphane Ducasse, Eyrolles, 2001, français ;
  Powerful Ideas in the Classroom, using squeak to enhance math and science learning, BJ Allen-Conn et Kim Rose, traduit en français.

Manifestations

- Smalltalk Party : Journée francophone annuelle, organisée à Paris par les Smalltalkiens français.
  `http://community.ofset.org/wiki/SmalltalkParty`
- Conférence du groupe européen des utilisateurs de Smalltalk (ESUG). Elle réunit chaque année, depuis 1993, les Smalltalkiens industriels et académiques dans un pays d'Europe.
  `http://www.esug.org/conferences`
- Conférence annuelle, organisée en Amérique du nord par le STIC (`http://www.stic.st`), association qui réuni les grands acteurs industriels et éditeurs de Smalltalk.
  `http://www.smalltalksolutions.com/`

Internet

- Site officiel en anglais :
  `http://www.squeak.org`
- Wiki francophone :
  `http://community.ofset.org/wiki/Squeak`

Smalltalk

un langage de programmation
purement **orienté objet**
et un environnement **dynamique**

EUROPEAN SMALLTALK USER GROUP
esug
www.esug.org

Concepts importants de Smalltalk

Smalltalk est un langage *orienté objet*, à *typage dynamique* dont la syntaxe est minimale et peut s'apprendre en *quinze minutes*.

Sa principale force vient du fait qu'il soit *très cohérent* :
– tout est objet : les classes, les méthodes, les nombres, etc.
– très peu de règles et aucune exception.
Smalltalk fonctionne sur le principe d'une *machine virtuelle*. Le développement se fait dans une *image mémoire* dans laquelle se trouve l'ensemble des objets du système avec lesquels il est possible d'interagir.

Syntaxe Smalltalk

Mots réservés	
`nil`	objet indéfini (valeur par défaut des variables)
`true` et `false`	objets booléens
`self`	l'objet courant
`super`	l'objet courant dans le contexte de la super classe
`thisContext`	partie de la *pile d'exécution* correspondant à la méthode courante

Caractères réservés	
`:=` (ou ←)	affectation
`^` (ou ↑)	retour du résultat d'une méthode
`| var1 var2 var3 |`	déclaration de variables temporaires
`$a`	le caractère `a`
`#(abc 123)`	tableau contenant deux littéraux : le symbole `#abc` et le nombre `123`
`.` (point)	termine toute expression
`;`	cascade de messages
`[ ]`	bloc de code (c'est un objet !)
`"commentaire"`	
`'chaîne de caractères'`	

Envoi de messages

L'appel de méthode se fait par envoi de message. Le message se construit sur la base du langage naturel avec un sujet, un verbe et des compléments. Tout envoi de message retourne un objet. Tous les messages sont envoyés à un objet que l'on appelle le receveur du message. Il existe trois types de messages : unaire, binaire et à mots-clés.

Messages unaires. Un message unaire n'a pas d'argument.

```
array := Array new.
array size.
```

Le premier exemple crée et retourne une nouvelle instance de la classe Array en lui envoyant le message `new`. Le deuxième exemple demande la taille du tableau ce qui retourne `0`.

Messages binaires. Un message binaire ne prend qu'un argument, se nomme par un symbole et est souvent utilisé pour des opérations arithmétiques.

```
3 + 4.
'Hello', ' World'.
```

Le message `+` est envoyé à l'objet `3` avec comme paramètre `4`. Dans le second cas, le message `,` est envoyé à la chaîne de caractères `'Hello'` avec `' World'` en paramètre.

Messages à mots-clés. Un message à mots-clés peut prendre un ou plusieurs arguments. Les arguments sont insérés entre chaque mot-clé, après les deux-points.

```
'Smalltalk' allButFirst: 5.
3 to: 10 by: 2.
```

Le premier exemple appelle la méthode `allButFirst:` sur une chaîne de caractères et passe l'argument `5`. La méthode retourne la chaîne de caractères `'talk'`. Le deuxième exemple retourne une collection contenant les éléments `3`, `5`, `7` et `9`.

Bloc

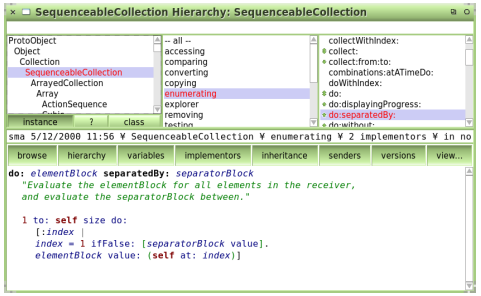
Les blocs sont des objets qui contiennent du code non exécuté immédiatement. Ils sont à la base des structures de contrôles comme les conditionnelles et les boucles.

```
#('Hello ' 'World')
       do: [:string | Transcript show: string].
```

Dans cet exemple, le message `do:` est envoyé à un tableau de chaînes de caractères avec un bloc en paramètre. Le bloc est évalué une fois pour chaque élément du tableau. Le paramètre `string` du bloc contient chaque élément du tableau, l'un après l'autre. Le résultat de l'expression complète est l'affichage des chaînes `'Hello '` puis `'World'` dans le transcript.

Environnement de développement

La plupart des implémentations de Smalltalk sont fournies avec un environnement de développement intégré qui permet de naviguer dans le code et d'interagir avec les objets. De nombreux outils sont disponibles, tous implémentés en Smalltalk grâce à une API de réflexion :
– un navigateur de classes et de méthodes ;
– des outils de refactorisation ;
– un inspecteur d'objets ;
– un débogueur ;
– etc.
L'environnement permet d'évaluer du code par un simple raccourci clavier et de voir immédiatement le résultat.



Le navigateur de code de Squeak

Implémentations

Il existe différentes implémentations de Smalltalk :

Squeak : implémentation libre, gratuite et multi-plateforme. Activement développée par une communauté internationale.

VisualWorks : implémentation propriétaire et multi-plateforme, disponible gratuitement pour une utilisation personnelle.

Gemstone : implémentation propriétaire qui intègre une base de données objet à hautes performances.

Et d'autres : GNU Smalltalk, Smalltalk/X, SyX, VA Smalltalk, Dolphin…