2

Visite guidée de quelques applications

Le chapitre précédent nous a permis de prendre un premier contact avec Squeak. Nous allons maintenant découvrir quelques-unes de ses possibilités, à travers une visite guidée de quelques applications de démonstration.

Squeak dans un navigateur Web : les « squeaklets »

Avant de passer à la description de ces applications, nous allons tout d'abord mettre en évidence une de leurs caractéristiques essentielles : tout programme développé en Squeak fonctionne à l'identique dans un navigateur Web, sans aucune modification.

Dans un premier temps, il est nécessaire d'installer un petit plug-in. Téléchargez le plug-in correspondant à votre système sur le Web (http://www.squeaklet.com/NPSqueak/download.html). Installez-le, en double cliquant sur l'installeur *SqueakPlugin*.

Dans un second temps, nous allons créer une page HTML qui permette l'ouverture d'un environnement Squeak fonctionnant dans un navigateur Web. Saisissez dans un éditeur de texte quelconque une page HTML contenant le code suivant :

```
<HTML>
   <b> Squeak fonctionne à l'identique dans un navigateur Web </b>
   <EMBED
      type="application/x-squeak-source"
      ALIGN="CENTER"
      WIDTH="700"
```

```
      HEIGHT="600">
   </EMBED>
</HTML>
```

Sauvegardez cette page, puis ouvrez-la dans un navigateur Web, quel qu'il soit. La zone grisée qui apparaît est un espace de travail Squeak vide. Cliquez dedans avec le bouton gauche, et vous retrouvez le menu principal, à partir duquel vous pouvez lancer les outils que nous avons vus précédemment (voir figure 2-1).

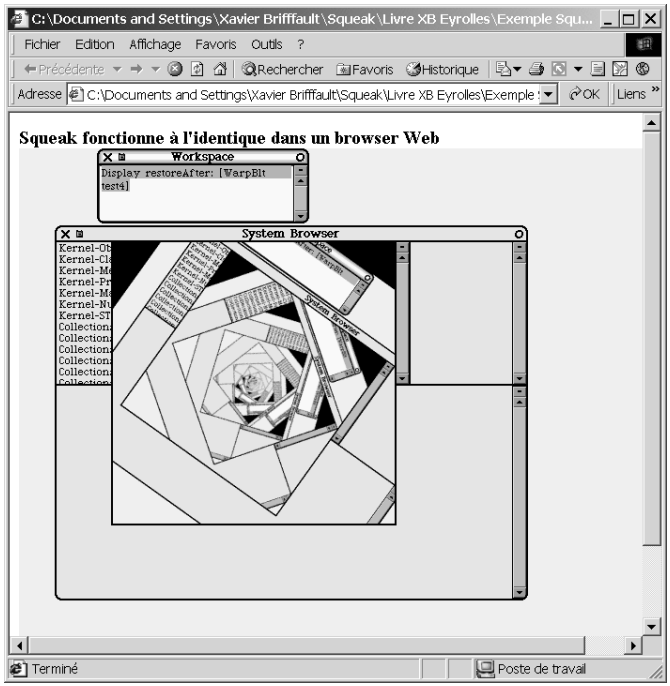


Figure 2–1
Squeak dans un navigateur Web

La création d'applets (ou squeaklets) est tout aussi simple. Créez un document HTML qui contienne le code suivant, en remplaçant l'URL du champ `src` (ligne 5) par une URL qui vous est propre :

```
<HTML>
   <b> Squeak fonctionne à l'identique dans un navigateur Web </b>
   <EMBED
      type="application/x-squeak-source"
      src ="http://m17.limsi.fr/Individu/xavier/maPremiereSqueaklet.sts"
      ALIGN="CENTER"
      WIDTH="700"
      HEIGHT="600">
   </EMBED>
</HTML>
```

Créez ensuite un fichier texte nommé "maPremiereSqueaklet.sts", dans lequel vous insérez l'un des exemples de code précédemment décrits dans le chapitre 1, par exemple :

```
Display restoreAfter: [WarpBlt test1].
```

Placez alors ce fichier à l'endroit qui correspond à l'URL spécifiée ligne 5 du fichier HTML, et ouvrez-le dans un navigateur Web. Le code du fichier `sts` s'exécute. Vous venez de créer votre première applet Squeak. Une applet Squeak peut contenir tout type de code, et fonctionne de façon identique dans un navigateur Web et dans un environnement graphique traditionnel. Ce fonctionnement est très intéressant, par rapport à d'autres langages où les développements pour navigateur Web sont différents des développements pour environnement graphique.

Applications multimédias en Squeak

Squeak offre, avec les milliers de classes qu'il propose, de nombreuses autres possibilités, que nous découvrirons en détail tout au long de ce livre. Avant d'en développer la technique, et de se lancer dans l'apprentissage du développement en Squeak, qui est l'objet de ce livre, nous allons proposer au lecteur de manipuler par lui-même quelques applications finales intéressantes et d'un emploi aisé afin de mieux percevoir les possibilités du langage. Plusieurs démonstrations sont fournies à cet effet dans la livraison standard de Squeak, et nous allons brièvement les présenter. Ces démonstrations sont accessibles à partir de l'interface principale de Squeak, en cliquant sur la fenêtre `The Worlds of Squeak`.

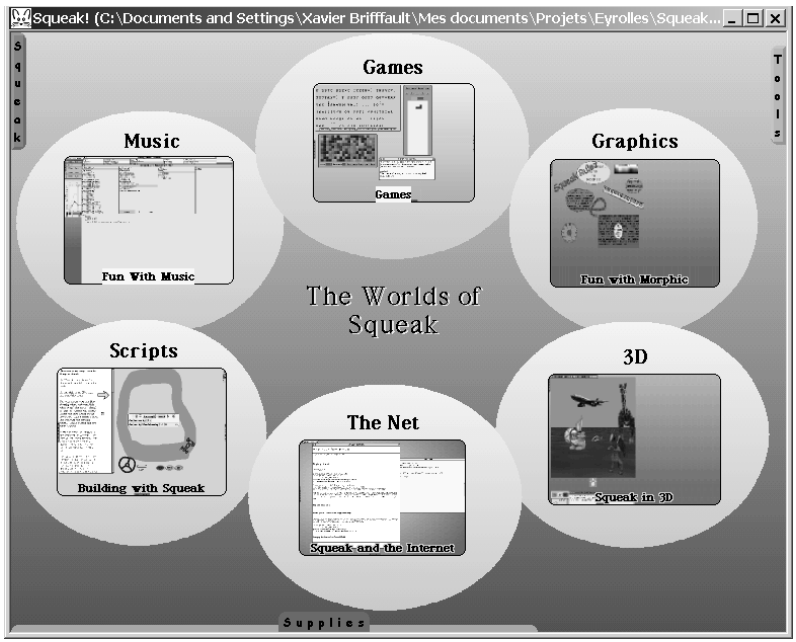


Figure 2–2
Accès aux démonstrations de Squeak

La fenêtre de la figure 2-2 qui apparaît alors présente six groupes de démonstrations différents : les applications musicales, les jeux, les applications graphiques, le graphisme 3D, l'Internet et les scripts. Pour entrer dans un « monde », il suffit de cliquer dessus. Pour en sortir, sélectionnez l'option <previous project> dans le menu bouton gauche.

Musique électronique, MIDI, et manipulation de sons

L'univers musical de Squeak est illustré figure 2-3. La première fenêtre contrôle la lecture de la partition polyphonique (plus précisément le « piano roll ») qui se trouve à droite. On y définit les instruments (des menus permettent de changer d'instrument), la vitesse et le volume de jeu, et on y trouve les boutons de contrôle de la lecture. Squeak peut lire et créer des fichiers au format MIDI, et piloter des instruments qui acceptent ce standard.

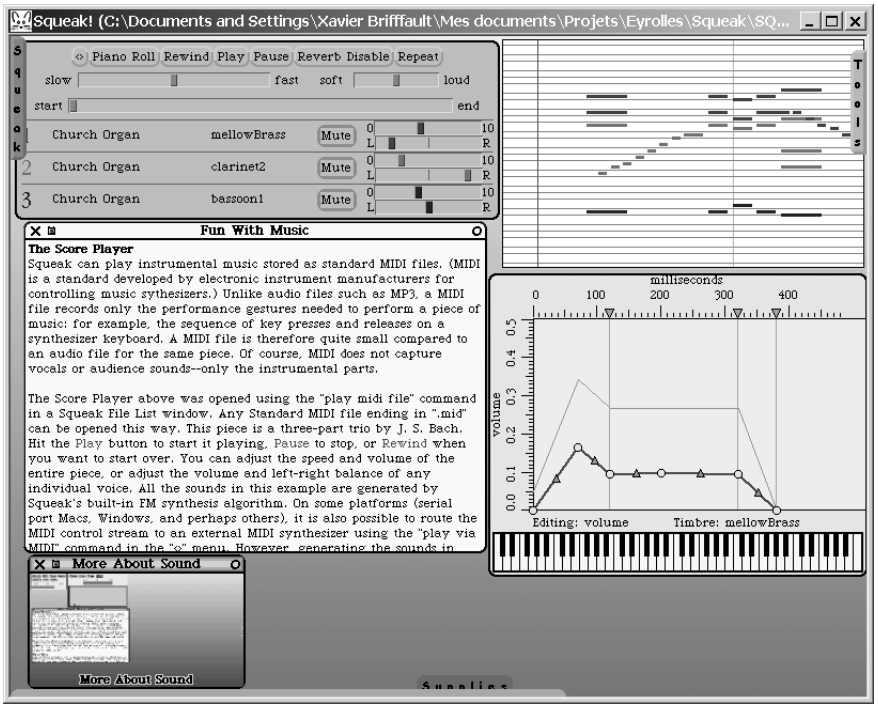


Figure 2–3
Quelques applications musicales de Squeak

Un éditeur d'enveloppe sonore permet de définir les paramètres des sons pour chaque type d'instrument (volume, attaque, tenue, décroissance, durée…). Il est accessible à partir du menu des choix d'instrument sur le Player MIDI.

La partition peut être éditée directement à partir d'un clavier (avec la durée de chaque note), obtenu au moyen du menu contextuel sur la partition elle-même.

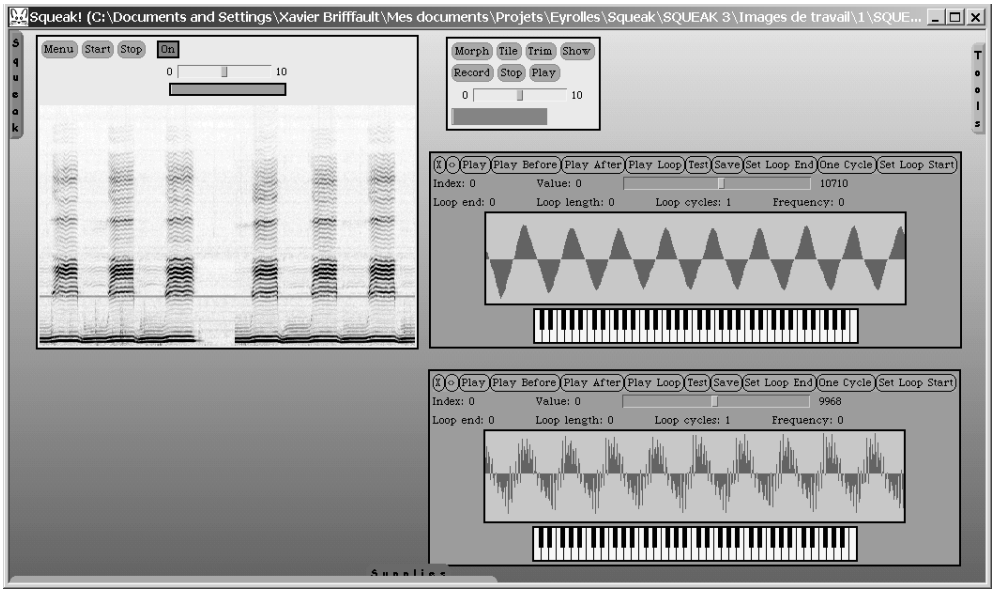


Figure 2–4

Enregistreur de sons, analyseur de spectre et visualisation de l'onde sonore

Implémentation

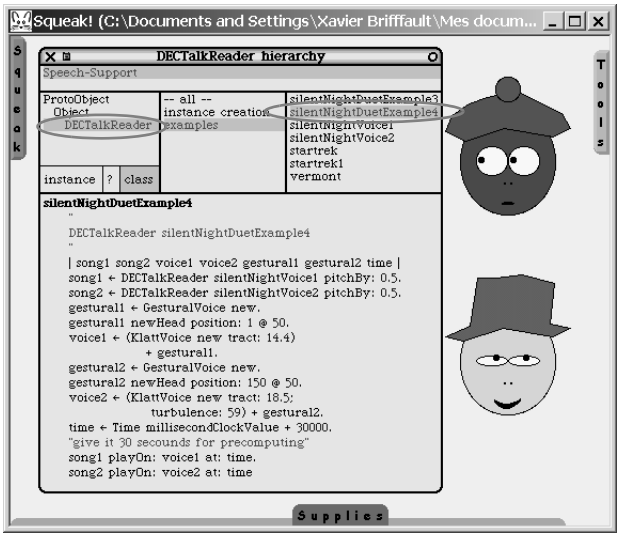
Les sons sont implémentés par des sous-classes de la classe `AbstractSound`, catégorie `Sound-Synthesis`. Les classes musicales de Squeak se trouvent pour la plupart dans les catégories de classes qui commencent par `Sound (Sound-Synthesis, Sound-Scores, Sound-Interface)`.

Les fichiers MIDI peuvent aussi être téléchargés sur le Web et joués directement. Vous pouvez par exemple essayer le code suivant :

```
MIDIFileReader playURLNamed:
   'http://squeak.cs.uiuc.edu/Squeak2.0/midi/wtellovr.mid'.
```

Parmi les autres outils de manipulation de sons de Squeak (cliquez sur la fenêtre `More About Sound`), on trouvera (voir figure 2-4) un enregistreur de sons (`RecordingControlsMorph`), un analyseur graphique de spectre (`SpectrumAnalyzerMorph`) et une interface graphique pour la visualisation d'ondes sonores (`WaveEditor`) ; le tout permettant d'enregistrer et de recomposer à loisir des sons. De nombreuses autres fonctionnalités de gestion des sons sont disponibles, parmi lesquelles on trouvera la gestion des codecs (compression/ décompression du son) ADPCM, GSM, MuLaw, Wavelet (tous implémentés par des sous-classes de la classe `SoundCodec`), mais aussi la gestion des standards AIFF, WAV…

Figure 2–5
Deux visages animés chantant « Silent night »



Synthèse et reconnaissance vocale, animation faciale

Un ensemble complet de classes qui permettent la synthèse vocale (*text to speech*) est fourni dans les applications préfixées par `Speech` (`Speech-TTS`, `Speech-Support`, `Speech-Klatt`, `Speech-Phonetics`, `Speech-Events`). La classe qui pilote la synthèse est `Speaker` (application `Speech-TTS`). Elle propose une méthode `say:` qui permet de faire prononcer au système une phrase passée en argument, par exemple :

```
Speaker bigMan say: 'Around midnight'.
```

Dans cet exemple, c'est la voix `bigMan`, créée par l'envoi de la méthode du même nom, qui est utilisée. Une vingtaine de voix différentes sont fournies en exemple en méthode de classe de `Speaker` (parmi lesquelles, `child`, `kid`, `man`, `woman`, `exorcist`…).

> **Méthodes de classe**
>
> Comme nous le verrons dans le chapitre sur le modèle objet de Squeak, tous les éléments de Squeak sont des objets. À l'instar des objets qu'elles définissent et contiennent, les classes disposent donc elles aussi de méthodes. Ces méthodes sont alors appelées méthodes de classe, par opposition aux méthodes d'instance.

La synthèse vocale utilise un modèle de synthèse à formants, selon une technique proposée dans les années 1980-90 (voir l'application `Speech-Klatt` pour plus de détails). Les éléments déterminants de la qualité de voix produite peuvent être déterminés par programmation.

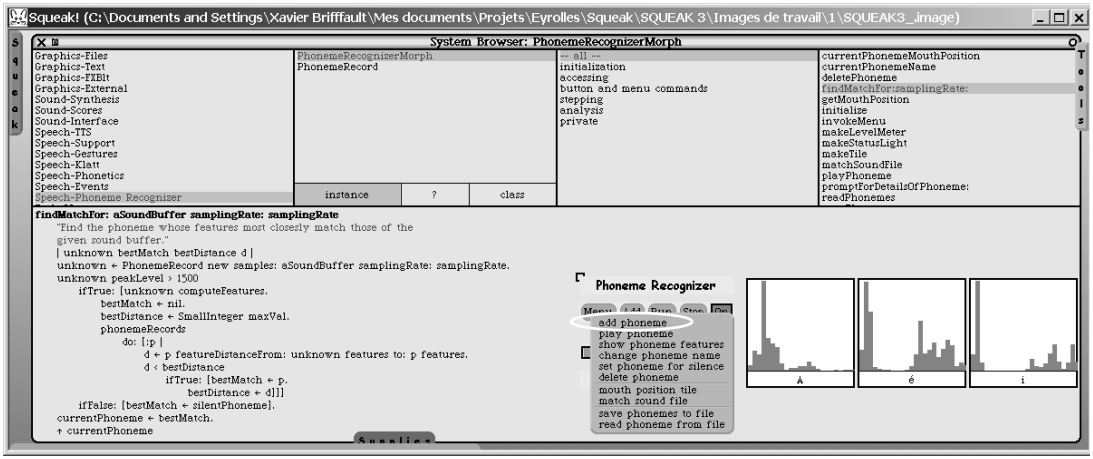


Figure 2–6

Le système de reconnaissance de phonèmes, quelques phonèmes, et un exemple de code

L'effrayante voix du film *L'Exorciste* est par exemple produite de la façon suivante :

```
(Speaker new pitch: 40.0;
           speed: 0.5;
           voice: (KlattVoice new tract: 10;
                                 diplophonia: 0.4;
                                 jitter: 0.3;
                                 shimmer: 0.5;
                                 turbulence: 50))
           say: 'Hello from Hell'
```

Il est également possible d'associer des visages animés à la synthèse vocale. La figure 2-5 illustre deux de ces figurines animées, en train de chanter *Silent night*. Les mouvements des lèvres sont synchronisés avec la mélodie. En arrière-plan, l'explorateur montre le code qui est utilisé pour l'animation. Les objets animés utilisés appartiennent à la classe `GesturalVoice`, de l'application `Speech-Gestures`.

Syntaxe

La portion de code que l'on vient de présenter illustre quelques nouveaux points de syntaxe, qui seront détaillés dans le chapitre 5. Dans l'exemple, `Speaker` et `KlattVoice` sont des classes. `pitch:`, `speed:`... sont des méthodes. Le point-virgule indique que les messages doivent être envoyés au même objet que le message précédent, et non à l'objet renvoyé par ce message.

Une reconnaissance vocale, très simplifiée, est également proposée sous la forme d'un système de reconnaissance de phonèmes (voir figure 2-6, `PhonemeRecognizerMorph new openInWorld`). Le système est capable, après enregistrement de phonèmes (o, e, a, è…) par l'utilisateur, de reconnaître ces derniers lorsqu'ils sont prononcés. Des positions spécifiques pour les lèvres des figures animées peuvent ensuite être associées aux sons enregistrés, ce qui permet une animation en temps réel, pilotée vocalement par l'utilisateur. Il ne s'agit évidemment pas d'un système de reconnaissance complet (*speech to text*), mais il offre déjà d'intéressantes possibilités.

Interfaces graphiques évoluées : Morphic

Les composants graphiques de Squeak sont appelés des morphs et ils ont d'étonnantes possibilités. La figure 2-7 comporte huit morphs amusants que nous allons décrire.

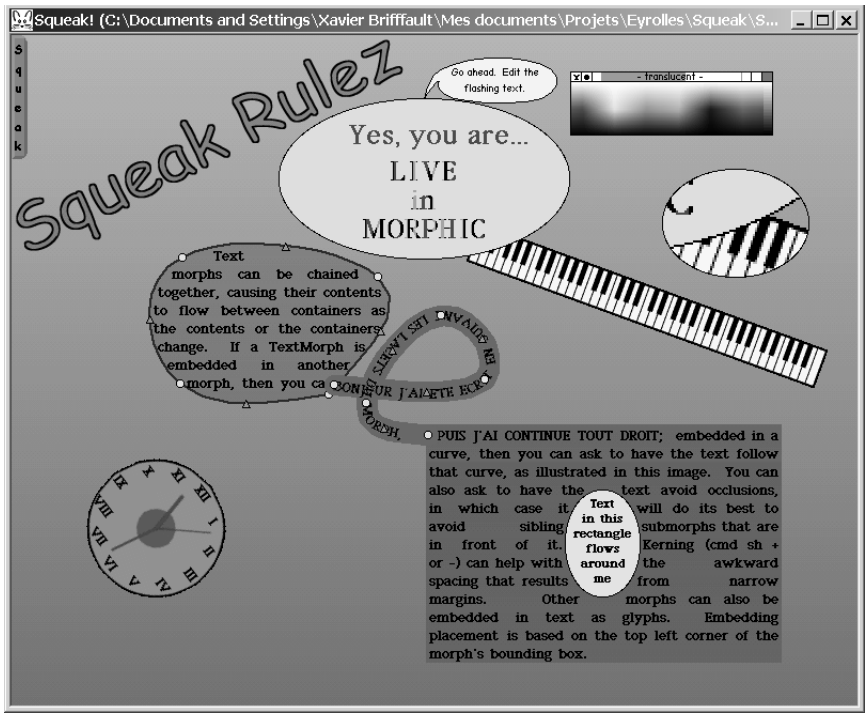


Figure 2–7
Quelques exemples de morphs

Le morph 1 (`MatrixTransformMorph`) a la particularité d'avoir un centre de rotation paramétrable. Le numéro 2 (`ColorPickerMorph`) est une palette de couleurs qui permet de modifier la coloration des objets sélectionnés. Le 3 (`EllipseMorph`) est un texte animé qui peut être édité sans cesser l'animation, tandis que le 4 est une simple loupe (`ScreeningMorph`). Le numéro 5 est un piano numérique (`PianoKeyboardMorph`) qui reste parfaitement utilisable malgré sa position peu orthodoxe.

Le sixième est une horloge à aiguille (`WatchMorph`), qui fonctionne quelle que soit la rotation qu'on lui impose. Le morph 7 est assez spectaculaire : le texte est affiché en suivant les méandres de la zone colorée, et peut aussi être édité ainsi (`CurveMorph`). Dans le morph 8 est illustré le contournement dynamique d'un ovale par du texte (`EllipseMorph` et `TextMorph`).

Nous reviendrons en détail sur les morphs dans le chapitre 9, « Le développement d'interfaces graphiques ».

Jeux de démonstration

Une dizaine de jeux de démonstration sont fournis avec Squeak, parmi lesquels on reconnaîtra Tétris, la réussite, les mots croisés (voir figure 2-8). Ces exemples sont surtout intéressants parce que le code y est toujours totalement accessible, lequel est contenu dans la catégorie `Morphic-Games` (l'implémentation complète de Tétris sera analysée en détail dans le chapitre 9, dans lequel le développement d'interfaces Morphic est traité).

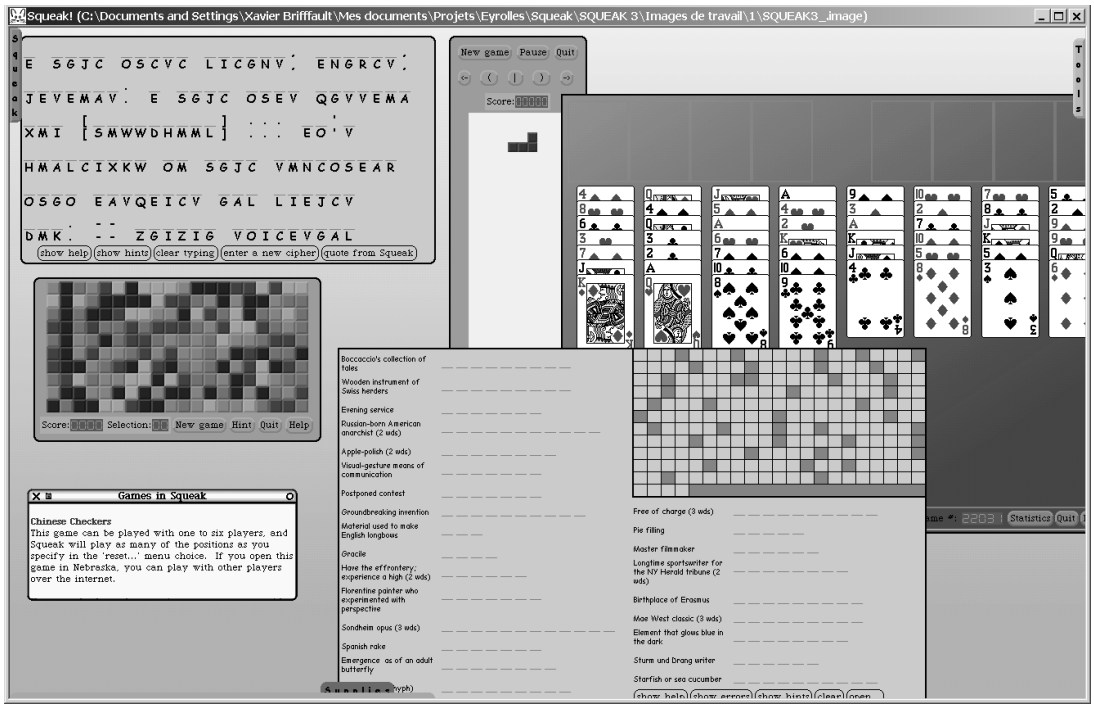


Figure 2–8

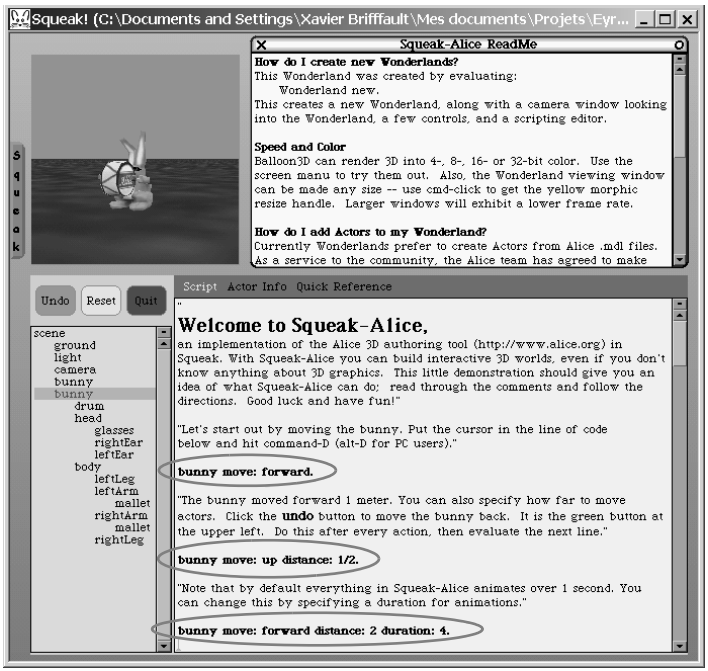
Quelques jeux de Squeak

Squeak et le graphisme 3D : Alice

Après cet aperçu des possibilités graphiques 2D de Squeak, nous allons maintenant considérer le framework graphique 3D, nommé Balloon, en regardant comment Alice, une application conçue pour apprendre aux novices à programmer, fonctionne (voir figure 2-9). Entièrement réalisé en Squeak, sans utilisation de librairies externes, Balloon permet la création d'objets 3D et leur visualisation en temps réel. Ce sont ces fonctionnalités qu'Alice utilise. Alice propose ainsi la possibilité de manipuler des objets 3D et leur animation avec un langage de scritp dedié. Les classes qui implémentent Balloon sont regroupées dans les catégories de classes commençant par `Balloon3D` (le nom du moteur graphique d'Alice). Ces catégories sont au nombre de trente, et comprennent aussi bien les interfaces graphiques que les objets et les algorithmes de calcul. La catégorie `Balloon3D-Demo morphs` contient trois classes d'exemples très simples.

Figure 2–9

Un exemple d'objet 3D créé par Alice



La fenêtre de visualisation graphique offre les possibilités de manipulation habituelles dans ce type d'environnement (déplacement de l'objet, de la caméra, zoom, travelling…), par manipulation directe sur l'image elle-même, ou en utilisant les flèches de contrôle.

Un autre élément intéressant d'Alice est son langage de script, très simple d'emploi, et qui permet d'animer les objets de la scène et de les lier aux actions de l'utilisateur. Ce langage de script utilise la syntaxe Squeak en y ajoutant quelques variables spécifiques. Par exemple, dans `bunny move: forward, bunny` référence le lapin, et `forward` indique un déplacement d'une unité vers l'avant. `move:` provoque donc le déplacement du lapin vers l'avant.

La distance et la durée peuvent être précisées. Par exemple :

```
bunny move: up distance: 0.5
bunny move: forward distance: 2 duration: 4.
```

Il est également possible de préciser le point de vue à partir duquel le déplacement est considéré, par exemple `bunny move: forward asSeenBy: camera`. La caméra est elle-même un objet, et peut être pilotée comme tout objet (`camera move: up`). Bien d'autres méthodes sont définies. Pour en prendre connaissance, on consulte la classe `WonderlandActor`.

Les objets Alice sont des objets composites, dont chaque élément peut être utilisé indépendamment (par exemple, le tambour, la tête, les pattes…). On peut ainsi manipuler les différents paramètres de chaque élément (couleur, taille…).

Par exemple :

```
bunny drum setColor: green.
bunny drum resize: 1/5.
bunny head resize: 1.5.
```

Chaque élément peut également être animé indépendamment :

```
bunny head turn: left turns: 1
```

ou

```
bunny head turn: left turns: 2 speed: 2
```

Des animations complexes peuvent également être définies en composant des actions élémentaires et en les affectant dans des variables. Par exemple,

```
saut := bunny move: up.  chute := bunny move: down
```

crée et exécute deux animations et les affecte dans les variables `saut` et `chute`. C'est une illustration intéressante de l'utilisation de la *réification* des actions : non seulement la méthode `move:` exécute le comportement spécifié, mais elle renvoie également un objet représentant l'action, qui va pouvoir être utilisé pour l'exécuter à nouveau, par exemple en la composant dans des actions plus complexes : `bond := w doInOrder: { saut. chute }.`

Terminologie

La réification est un procédé qui consiste à transformer en chose ce qui est mouvant. En programmation orientée objet, il peut s'agir, comme dans cet exemple, de la création d'un objet statique qui représente une action. Ce mécanisme est fréquemment utilisé dans Squeak ; les envois de messages, par exemple, peuvent être réifiés sur demande sous la forme d'objets `message`.

Dans cet exemple, `saut` et `chute` sont composés pour créer un `bond`. `w` est ici une variable dont la valeur est donnée par l'environnement qui contient un *Wonderland*, le monde dans lequel les objets Alice évoluent.

Le nombre de sauts à exécuter peut être déterminé de façon programmatique, comme ceci :

- `bond start` pour une seule exécution,
- `bond loop: 2` pour deux exécutions,
- `bond loop` pour une boucle sans fin.

Pour arrêter les bonds : `bond stopLooping`.

Les objets du *Wonderland* peuvent également être asservis aux actions de l'utilisateur. Par exemple, pour faire sauter le lapin dès que l'utilisateur clique sur le bouton droit de la souris :

```
bunny respondWith: [:event | bond start ] to: rightMouseClick.
```

Pour faire en sorte que le regard de Bunny soit toujours tourné vers la caméra :

```
bunny head pointAt: camera duration: eachFrame.
```

Il est également possible de lui faire suivre en temps réel les mouvements de la souris :

```
bunny head doEachFrame:
[bunny head
   pointAt: (camera transformScreenPointToScenePoint: (Sensor mousePoint)
      using: bunny)
   duration: rightNow].
```

Ce n'est là qu'un bref aperçu des possibilités d'Alice, que nous vous encourageons à explorer. Pour ce faire, de nombreux objets 3D déjà définis (au format mdl) sont disponibles à l'adresse ftp://st.cs.uiuc.edu/pub/Smalltalk/Squeak/alice/Objects.zip. Pour ajouter un nouvel objet dans un *Wonderland*, il suffit d'exécuter

```
w makeActorFrom: 'path/to/actor.mdl'
```

en remplaçant `path`, `to` et `actor` par les noms adéquats (quelques exemples sont présentés dans la figure 2-10).

Messagerie, client et serveur Web : les outils Internet

Les applications nommées `Celeste` et `Scamper` permettent respectivement la gestion des mails et l'accès aux pages HTML. Squeak propose également des classes (application `Network-IRC Chat`) pour l'accès aux *Internet Relay Chat* (IRC), et un framework complet pour la création de serveurs Web et de Swikis (application `Network-Pluggable Web Server`). Nous développerons en étude de cas dans la quatrième partie de ce livre des applications illustrant l'utilisation de ces bibliothèques.

Un Swiki (une implémentation en Squeak du concept de Wiki, créé par Ward Cunningham, http://c2.com/cgi/wiki?WelcomeVisitors) est un site collaboratif, qui peut être aisément modifié et administré à distance, grâce auquel un groupe de personnes peut accéder à de l'information et en créer sans aucune formation préalable. La plupart des sites d'information Squeak sont d'ailleurs supportés par des Swikis (voir par exemple http://minnow.cc.gatech.edu/squeak/1).

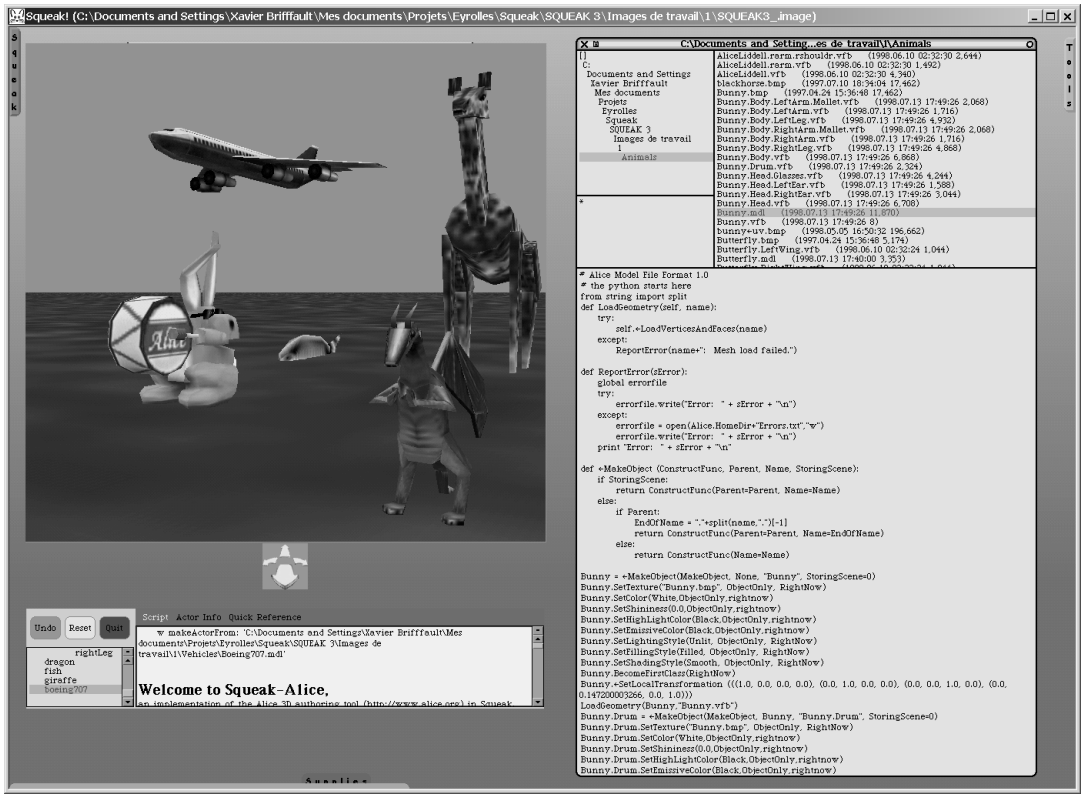


Figure 2–10

Exemples d'objets 3D et de script mdl

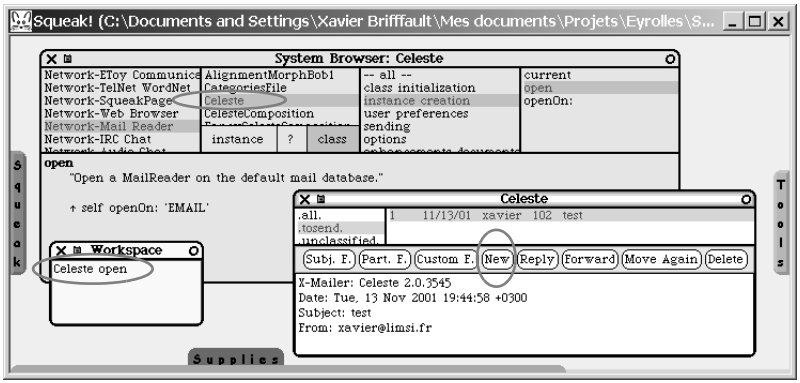
Gestion de mails (POP, SMTP) avec Celeste

> **Avertissement**
>
> Pour tester les exemples de cette section, il est nécessaire d'être connecté à Internet. Pour l'utilisation des mails, vous devez disposer d'un serveur POP et d'un serveur SMTP, permettant d'envoyer et de recevoir des mails.

Intéressons-nous tout d'abord aux facilités de gestion des mails. Squeak est fourni avec un framework de gestion des mails, sur lequel est construit Celeste, une application permettant de lire des e-mails et qui implémente une interface utilisateur, ainsi que les protocoles standards POP et SMTP. Ouvrez Celeste en évaluant le code suivant (voir figure 2-11) :

```
Celeste open 4
```

Figure 2–11
L'interface du gestionnaire de mail Celeste, et la catégorie correspondante



Celeste s'utilise de façon assez intuitive, comme tout autre client mail, et il n'est pas utile de le configurer par avance : les informations nécessaires (adresse de réponse, serveurs POP et SMTP…) ne sont demandées à l'utilisateur que lors de l'envoi ou de la réception de mails. Elles sont alors sauvegardées, et réutilisées lors des lancements ultérieurs. Pour écrire un mail, cliquez sur le bouton <New>. Vous pouvez organiser vos mails en utilisant des catégories, à l'aide du menu qui est associé à la zone des catégories.

L'interface de Celeste en fait un outil plaisant, mais ce n'est pas là son principal atout : avec l'apport supplémentaire de Squeak, il devient extrêmement facile de visualiser, de comprendre, d'utiliser et de réutiliser le code sous-jacent à l'interface. Pour vous en convaincre, nous allons inspecter les objets qui sous-tendent Celeste. Pour ce faire, à partir de la fenêtre Celeste (fenêtre 1, figure 2-12), ouvrez le menu de la fenêtre en cliquant sur le bouton du milieu (ou Alt+clic sur une souris à deux boutons). Sélectionnez l'option <debug>, puis l'option <inspect model> (fenêtre 2 et 3, figure 2-12). Un inspecteur s'ouvre alors. Il permet d'explorer l'objet représenté par l'interface (fenêtre 4, figure 2-12). On notera que cet objet est en fait une instance de la classe `Celeste`.

> **Terminologie**
>
> Dans la terminologie Squeak, l'objet qui est représenté par une interface est appelé son modèle, ce qui explique que l'option du menu est dénommée <inspect model>.

Il est possible d'inspecter chacune des variables d'instance de l'objet Celeste, en cliquant simplement sur le nom de la variable dans la liste affichée par l'inspecteur. L'inspection de la variable `mailDB` ouvre par exemple l'inspecteur 5 (voir figure 2-12). La fenêtre 6 est également un inspecteur sur la variable d'instance `messageFile` de l'objet `MailDB`. Ces inspections successives nous renseignent sur la classe et la structure des objets utilisés par Celeste. En utilisant un System Browser (option <open> <System Browser> du menu, ou raccourci clavier Alt b, voir la fenêtre 7, figure 2-12), on peut rapidement trouver la méthode utilisée pour provoquer l'envoi des messages qui sont en file d'attente. Cette méthode est `sendQueuedMail`, dont le code intégral est exposé dans la fenêtre 8.

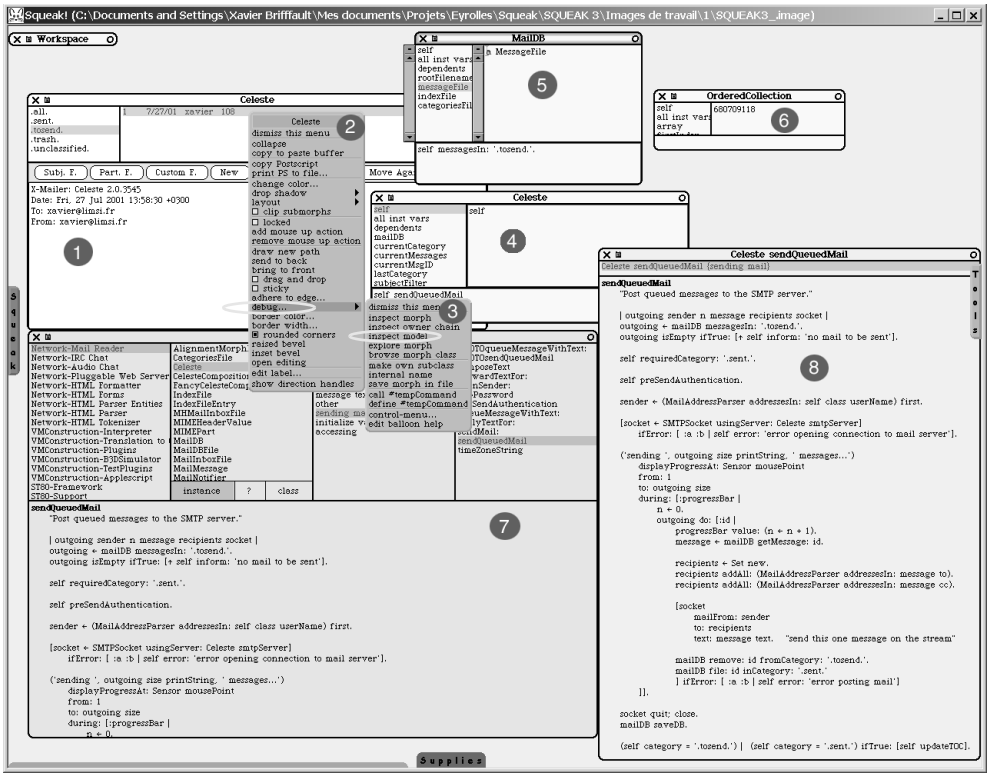


Figure 2–12

Utilisation programmatique de Celeste

Cette méthode peut être déclenchée, en évaluant `self sendQueuedMail` dans la zone inférieure de la fenêtre 4 par l'option <do it> du menu contextuel (ou le raccourci clavier Alt+d). Lors de cette évaluation, `self` prend comme valeur l'instance de Celeste inspectée, ce qui permet de lui envoyer un message.

> **Environnement de développement**
>
> L'environnement de Squeak permet de façon très simple, et à tout moment, d'accéder aux objets qui implémentent une interface utilisateur, et de les visualiser avec les outils adéquats. Cela facilite grandement la compréhension du code d'une application.

Ce petit parcours entremêlant interfaces, inspections d'objets, parcours du code et déclenchement de méthodes est un exemple typique de la facilité avec laquelle les différentes étapes d'un cycle de développement, traditionnellement très indépendantes, peuvent être simultanément mises en œuvre en Squeak.

La productivité de développement en est augmentée de façon très importante : plus de lourdes phases de compilation, d'édition de liens, de consultation de la documentation, de tâtonnements pour tenter de comprendre le comportement de méthodes dont le code source n'est pas accessible et dont la documentation est incomplète (ou fausse !).

Tout est immédiatement accessible, par simple navigation structurée dans le code, avec la facilité que procurent les interfaces graphiques de l'environnement de développement de Squeak. Nous verrons, au chapitre 10, consacré à l'environnement de développement, l'usage qui en est également fait pour le déverminage « incrémental ».

Méthode de développement

L'environnement de développement de Squeak, et les caractéristiques du langage, facilitent tant l'apprentissage direct par exploration/ manipulation que l'évaluation et le test fréquent du code, soit l'une des garanties de la qualité logicielle et de la productivité de développement.

Pour illustrer à nouveau ces facilités, intéressons-nous maintenant à un autre framework Web de Squeak, Scamper.

Le Web (HTML, HTTP) avec Scamper

Pour lancer Scamper, évaluez le code suivant (voir figure 2-13, fenêtre 1) :

```
Scamper open
```

Comme vous l'avez fait pour `Celeste`, inspectez le modèle de l'interface du navigateur Web ; vous obtenez la fenêtre 3. En inspectant la variable d'instance `recentDocuments`, nous obtenons la liste de ces documents (fenêtre 4). L'inspection d'un de ces documents nous montre alors comment sont implémentés les documents MIME (fenêtre 5).

Par exemple, nous pouvons voir la manière dont sont gérées les analyses (*parsing*) des documents HTML, tel celui vers lequel pointe la variable `content` de la fenêtre 5. Une rapide recherche dans le System Browser nous apprend qu'il existe une catégorie `Network-HTML Parser`.

Cette application contient, entre autres, une classe `HtmlParser`. L'examen des méthodes de classe de `HtmlParser` nous révèle une méthode de classe `parse`: (cliquez sur le bouton <class> du browser). Comme son nom le laisse supposer, cette méthode réalise l'analyse d'une chaîne HTML. Pour l'exécuter, saisissez dans la zone de saisie, le code suivant puis évaluez-le avec l'option <inspect> :

```
HtmlParser parse: self content readStream
```

Ce code déclenche l'analyse du source HTML et renvoie un objet Squeak instance de la classe `HTMLDocument` (1). Dans cet exemple, d'autres types d'objets sont obtenus comme résultat de l'analyse par Scamper du code HTML : `HtmlBody`, `OrderedCollection`… Vous pouvez explorer chacune des variables de chaque objet.

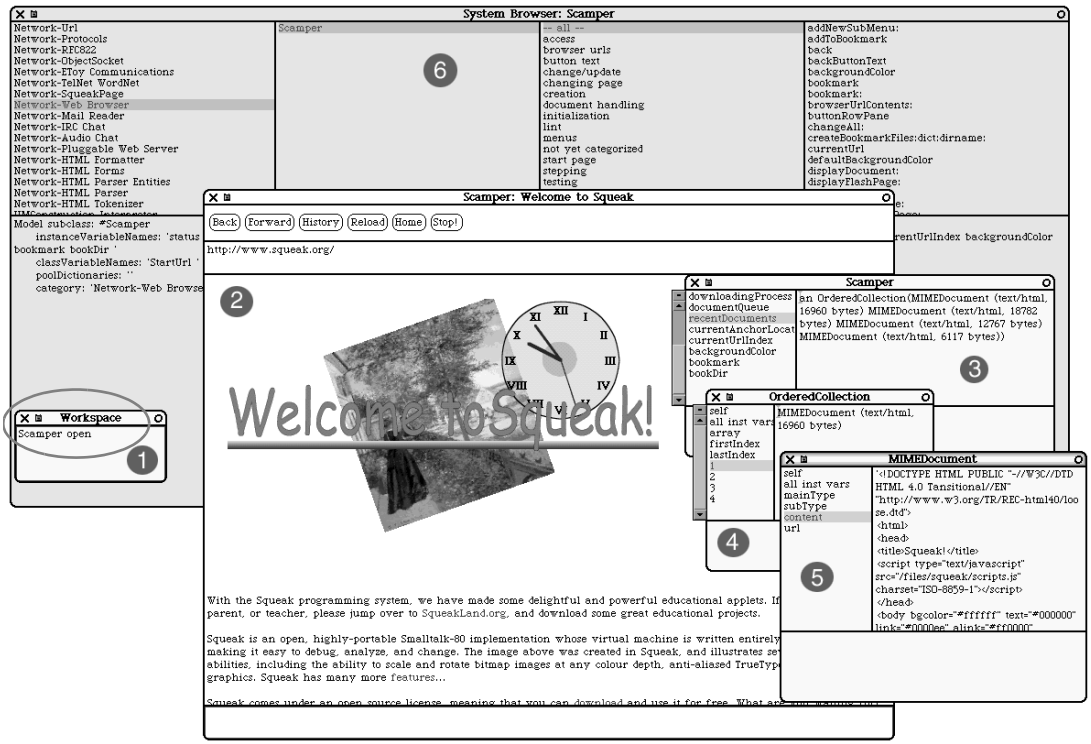


Figure 2–13

Utilisation programmatique de Scamper

En inspectant ensuite successivement les variables des objets obtenus (2, 3, 4), nous pouvons nous faire une idée de la structure interne d'un `HTMLDocument`. Cette structure prend la forme pour l'essentiel d'une liste qui en référence la tête et le corps.

Le corps (`HTMLBody`) du document est lui-même composé d'une séquence ordonnée (classe `OrderedCollection`) d'objets `HTMLEntity`. Il en existe toutes sortes – autant que de balises HTML ; en l'occurrence, il s'agit d'une `HTMLTable`.

Structure de données

Nous venons de mentionner la structure de données `OrderedCollection`. Les objets de cette classe contiennent une séquence ordonnée d'autres objets. Squeak propose une trentaine de classes qui implémentent les collections les plus utiles (tableau, liste chaînée, liste triée, liste de hachage, ensemble, dictionnaire, tableau creux...). Pour plus d'informations, explorez la catégorie `Kernel-Collections` à partir du browser système et reportez-vous au chapitre 6, « Les classes de collections ».

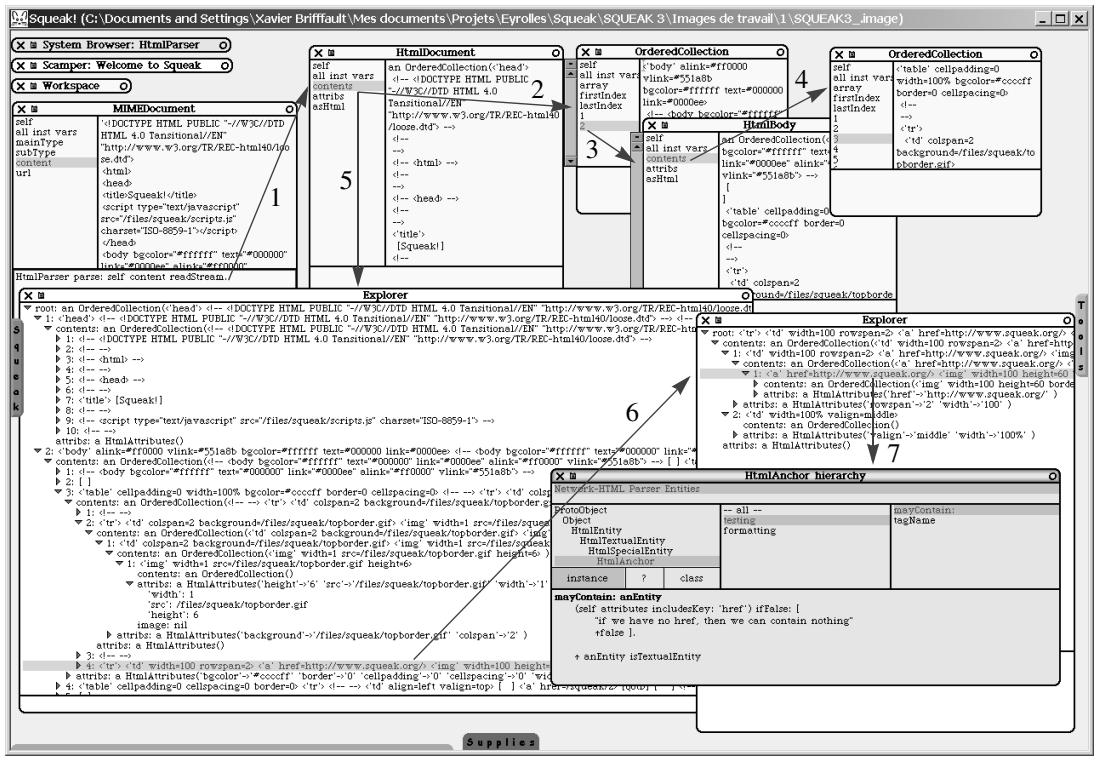


Figure 2–14

Découverte de la structure d'une analyse HTML

Les inspections successives 2, 3 et 4 permettent d'avoir accès à chaque objet dans une fenêtre autonome. Dans le cas des objets arborescents, comme c'est le cas du `HTMLDocument`, il peut être plus intéressant d'utiliser un explorateur hiérarchique qu'un inspecteur. L'explorateur hiérarchique est obtenu en sélectionnant l'option <explore> plutôt que <inspect> ou en tapant Alt+I (majuscule). L'explorateur obtenu montre très clairement la structure de l'objet (voir figure 2-14).

Il est évidemment possible d'appliquer cette exploration sur tout objet, quel que soit son niveau. En sélectionnant l'option <browse hierarchy> du menu contextuel, il est également possible d'accéder directement au browser sur la classe de l'objet sélectionné (fenêtre 7, il s'agit ici d'un explorateur hiérarchique).

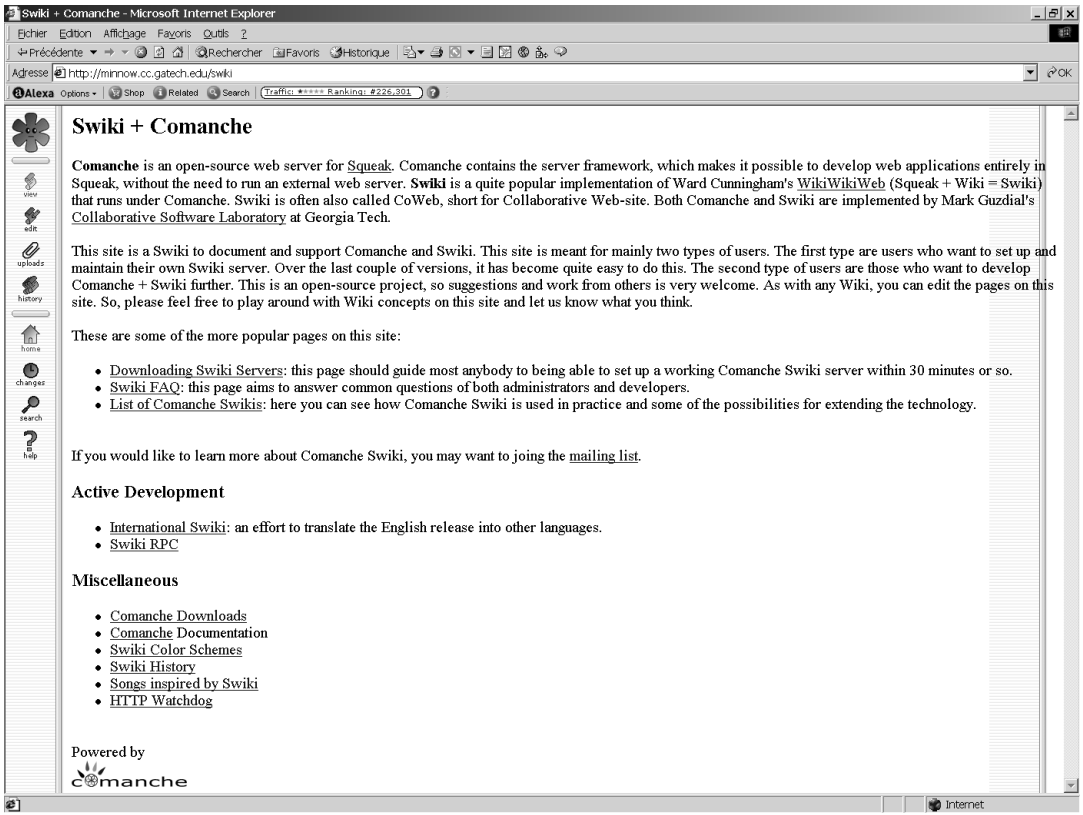


Figure 2–15

Exemple du Swiki de téléchargement de serveur Swiki

Création de serveurs Web et de Swikis

Une application complète de création de serveurs Web et de serveurs Swikis existe en Squeak. Pour configurer cette application, et pour pouvoir l'utiliser, il est nécessaire au préalable de télécharger un ensemble de fichiers additionnels, sous la forme d'une archive zip, à l'adresse http://minnow.cc.gatech.edu/swiki (voir figure 2-15). Cette page est elle-même servie par le serveur Web de Squeak et présentée sous la forme d'un Swiki.

Une fois que l'archive zip susmentionnée est téléchargée, son décompactage crée une hiérarchie de répertoires contenant les fichiers nécessaires au fonctionnement du serveur Web Squeak (voir figure 2-16).

Le répertoire de plus haut niveau (ComSwiki) contient une image et une machine virtuelle, spécialement configurées pour la gestion de ce serveur. Le lancement de cette image ouvre un environnement Squeak (voir figure 2-17).

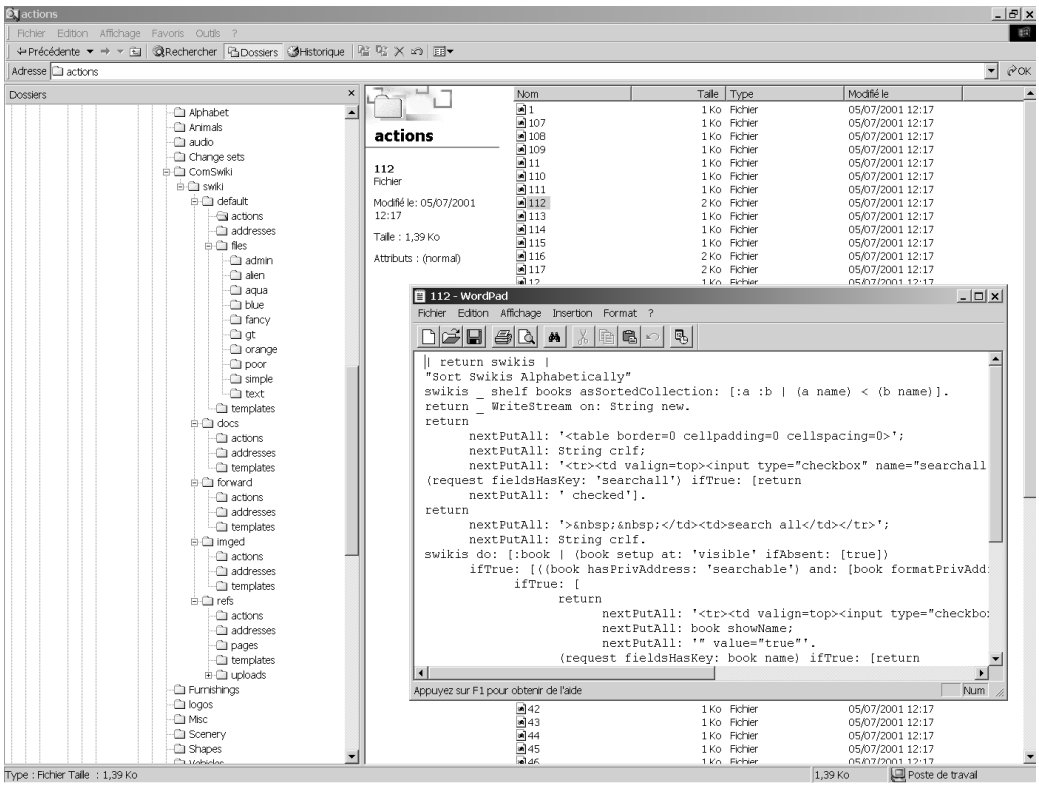


Figure 2–16

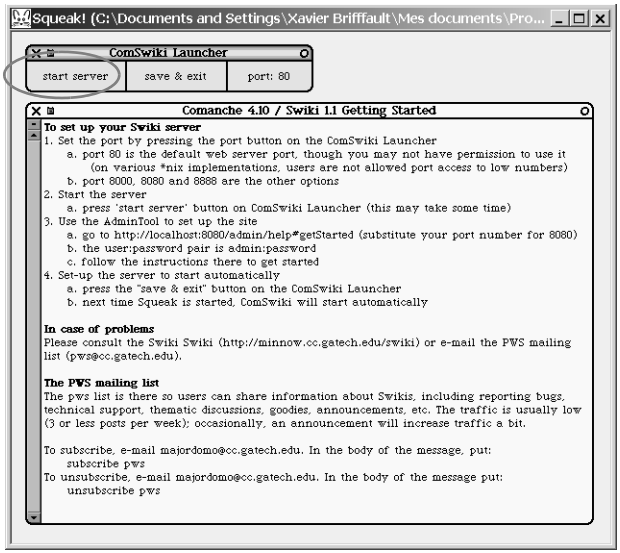
L'arborescence des fichiers créée pour le serveur Swiki, et un exemple de code

Qu'est ce qu'un Swiki ?

Un swiki est une implémentation en Squeak (la première a été faite par Mark Guzdial) du concept de Wiki (Swiki = Squeak + Wiki) créé par Ward Cunningham (http://c2.com/cgi/wiki?WelcomeVisitors). Il s'agit d'un outil de création, de parcours et de gestion d'un site Web collaboratif, qui peut être administré et modifié à partir d'un simple navigateur Web, par tout utilisateur disposant des droits nécessaires. Il s'agit d'un concept et d'outils simples, mais étonnamment puissants pour créer et exploiter des sites de partage d'informations dans des groupes collaboratifs. Un Swiki propose en particulier des fonctionnalités d'édition de pages, de recherche de tous les liens vers une page, d'upload de fichiers, d'historiques de modifications du site, de comparaison de versions, de recherche…De nombreuses implémentations différentes des Wiki sont disponibles (citons par exemple Zwiki, le wiki Zope – www.zope.org).

Pour plus de détails sur les concepts et les outils Wiki, nous renvoyons le lecteur au livre de Ward Cunningham, *Quick Collaboration on the Web : The Wiki Way.*

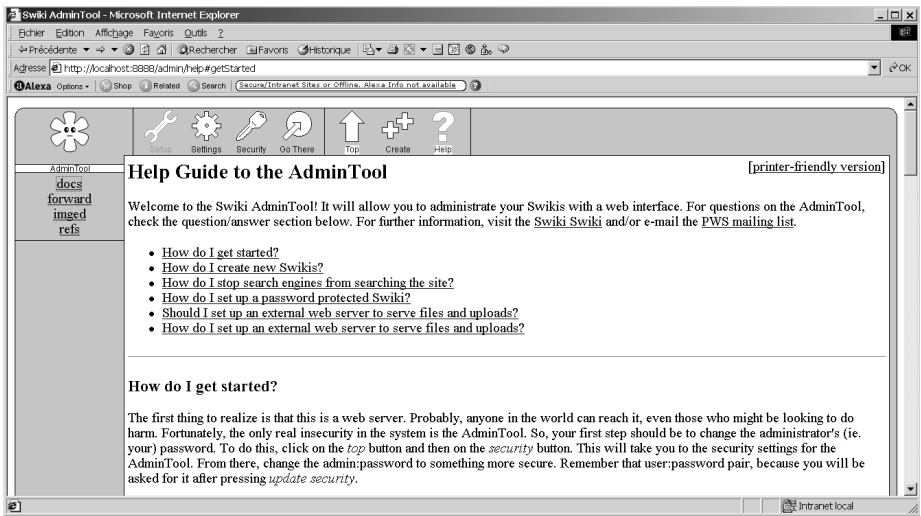
Figure 2–17
L'image ComSwiki



La petite fenêtre du haut contient trois boutons qui permettent le paramétrage du port à utiliser par le serveur, la sauvegarde de l'image et la sortie de l'application, et le démarrage/ l'arrêt du serveur ComSwiki.

Après démarrage de ce serveur, une connexion à l'adresse http://localhost:8888/admin/help#getStarted donne accès à la page d'accueil de l'outil d'administration de ComSwiki (voir figure 2-18).

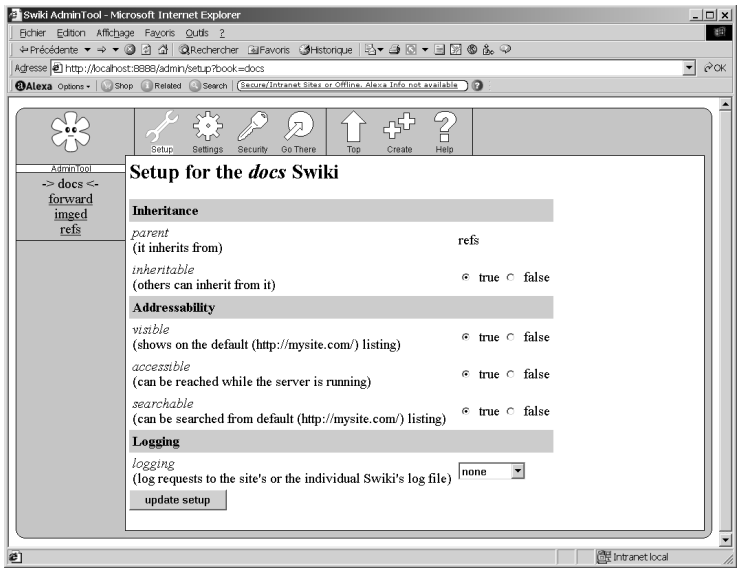
Figure 2–18
La page d'accueil de l'outil d'administration de ComSwiki



Si vous avez installé le serveur ComSwiki sur une autre machine que la machine locale, remplacez l'adresse de la machine (`localhost`) et le numéro du port (`8888`) par les valeurs qui correspondent à votre configuration. L'outil d'administration permet de créer des sites Swiki, et de définir leurs paramètres fondamentaux (sécurité, droits d'accès…).

On accède à ces différentes fonctionnalités au moyen des icônes qui figurent dans le sommet de la page. L'option <Setup> (voir figure 2-19) permet de paramétrer le comportement d'héritage des sites, la visibilité et la gestion du journal des requêtes.

Figure 2–19
Page d'accueil de l'option <Setup> de l'outil d'administration de site



L'option <Settings> (voir figure 2-20) permet de définir l'apparence du site, de spécifier l'emplacement où les fichiers ajoutés au site vont être stockés sur le serveur, les options de transformation du Swiki en site HTML traditionnel, le mot de passe administrateur, et les alertes à envoyer par mail lorsque des modifications surviennent sur le site.

L'option <Security> (voir figure 2-21) permet quant à elle la définition des droits d'accès et de modifications pour les utilisateurs. Les niveaux de sécurité proposés ne sont pas très sophistiqués[1], mais ils sont d'autant plus simples à utiliser.

L'option <Go There> permet d'accéder au site lui-même, ainsi qu'aux différentes fonctionnalités proposées par les sites Wiki (édition de pages, upload de fichiers, historique, comparaison de versions…).

Ces différentes fonctionnalités sont toutes accessibles à partir de la barre d'outils affichée dans le navigateur Web (voir figure 2-22).

1. Si l'on compare par exemple au paramétrage possible sur les serveurs Zope.

Figure 2–20

L'option <Settings> de l'outil d'administration du Swiki

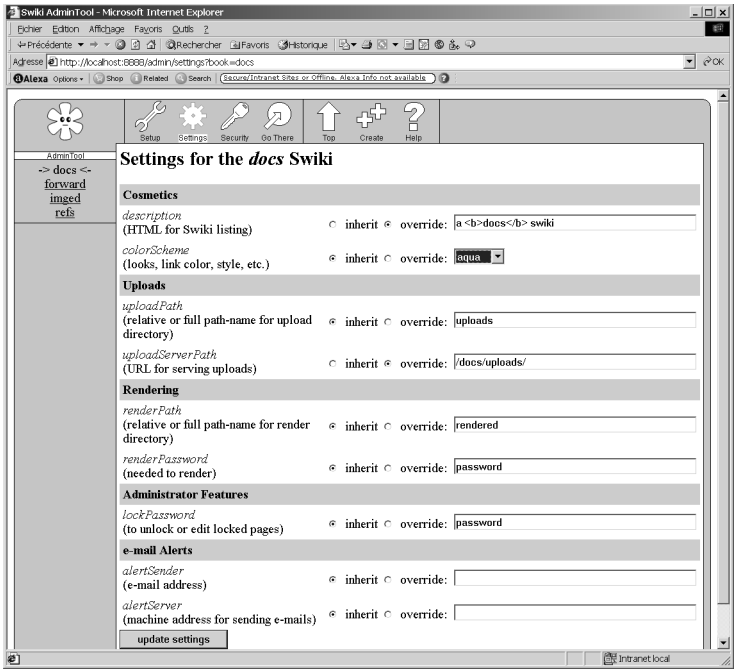
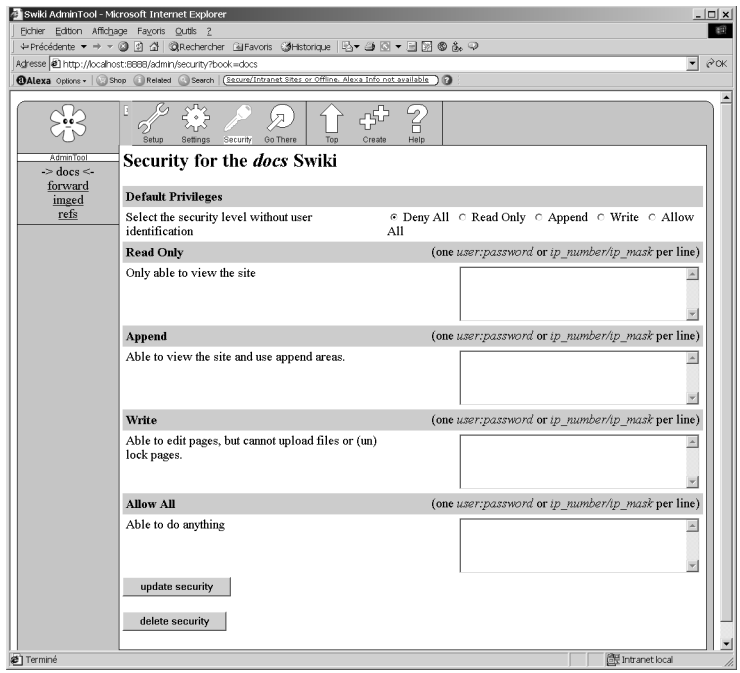


Figure 2–21

L'option <Security> de l'outil d'administration du Swiki



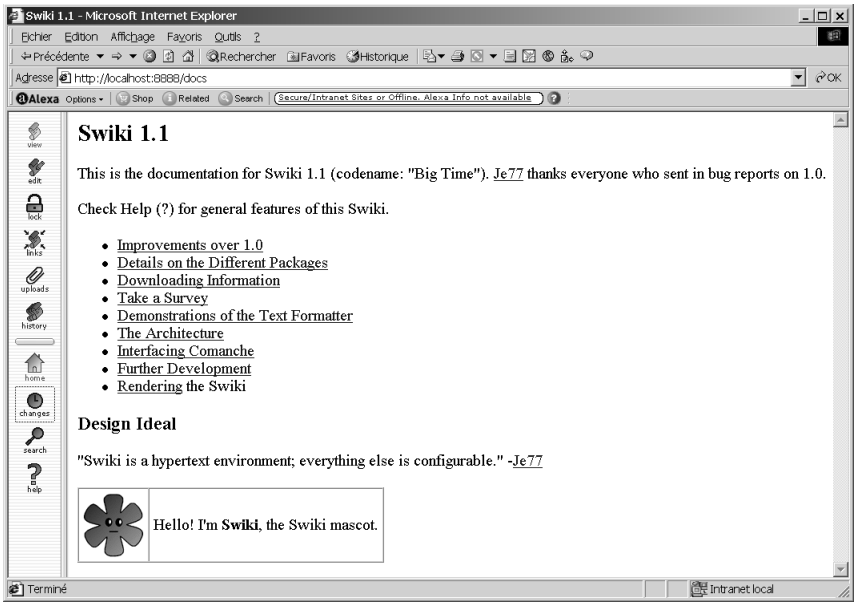


Figure 2–22
Un exemple de Swiki : le Swiki de présentation de la documentation

À l'instar des autres applications présentées jusqu'alors, `ComSwiki` est totalement programmé en Squeak et peut être modifié sans difficulté pour servir de base à d'autres développements. Les principales catégories regroupant les classes `ComSwiki` sont `Swiki-Comanche`, `Kom-kernel`, `Kom-demo`, `Kom-protocol http`, `Swiki-Formatting`, `Kom-HTML UI builder`, `Kom-Protocol abstract`, `Swiki-FileServer`, `Swiki-Structure`, `Swiki-Security`, `Swiki-Storage`. Nous détaillerons l'application `ComSwiki` dans le chapitre 12, consacré au développement d'applications Web.

Création de jeux interactifs et développement interactif d'interfaces : EToys

En combinant ses possibilités graphiques et de programmation de haut niveau, Squeak est en mesure de proposer un environnement de création d'objets graphiques (les EToys) pilotés par des scripts (à la LOGO), dont la programmation peut être assurée de façon traditionnelle, par des lignes de code, ou graphiquement en dessinant les objets et en combinant à la souris les actions disponibles. Une illustration de quelques éléments de cet environnement est donnée figure 2-23.

Les classes de cette application sont contenues dans les catégories `Morphic-Scripting`, `Morphic-Scripting Support` et `Morphic-Scripting Tiles`. Nous analyserons plus en détail le fonctionnement de l'application `EToys` dans le chapitre 11 consacré au développement d'interfaces graphiques en Squeak, section « Les interfaces de construction d'interfaces ».

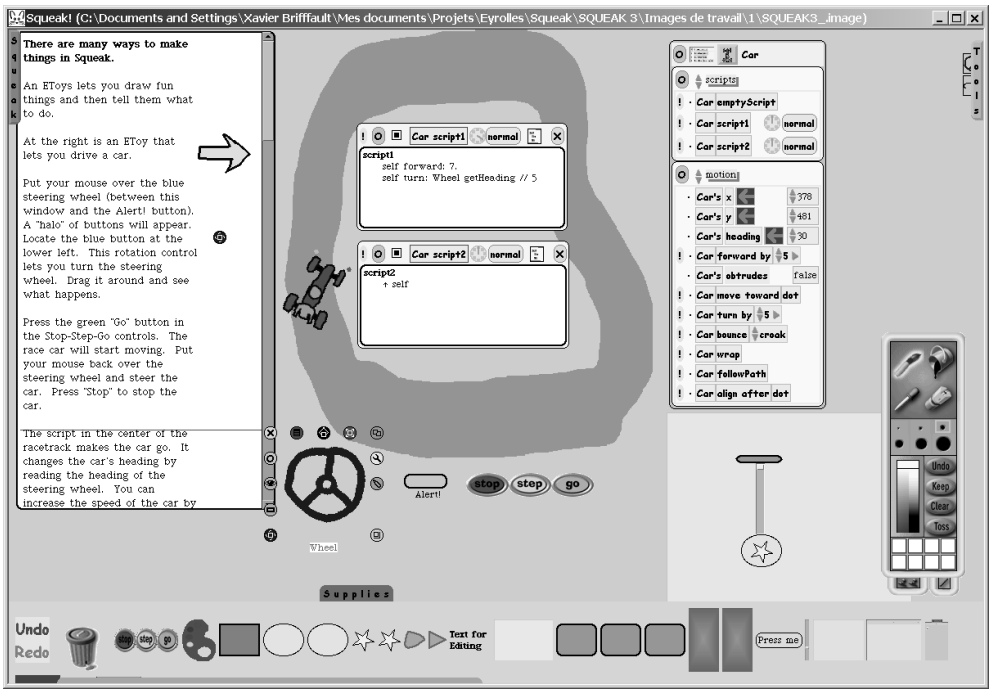


Figure 2–23

Un aperçu de l'environnement de création de EToys

Reconnaissance manuscrite avec Genie

Une application de reconnaissance de caractères, nommée Genie, est également proposée dans Squeak. Totalement paramétrable, Genie reconnaît les caractères tracés à l'aide de la souris, ou d'un stylo sur une tablette graphique ; des actions dans l'environnement peuvent être associées aux caractères reconnus.

Les classes de Genie sont regroupées dans les catégories de classes `Genie-Engine` (les classes de reconnaissance de gestes), `Genie-UI` (les interfaces utilisateurs utilisées pour le paramétrage du système de reconnaissance de gestes) et `Genie-Integration` (les classes d'intégration de Genie dans Morphic).

L'option <help> du menu <World> contient trois options spécifiques à Genie permettant de l'activer, d'inspecter et d'éditer le dictionnaire des formes, d'inspecter et d'éditer les propriétés d'affichage de Genie.

La classe `AGenieIntroduction` contient en commentaire une documentation complète sur le fonctionnement de Genie.

En résumé

Ce chapitre nous a permis d'aborder plusieurs applications de démonstration fournies avec l'environnement Squeak et d'affiner notre connaissance du langage et de l'environnement. Nous avons présenté la notion de squeaklet, et avons constaté que Squeak pouvait fonctionner sans difficulté dans un navigateur Web. En parcourant les applications multimédias, nous avons pu apprécier les outils de traitement des sons et de musique électronique, ainsi que les aspects plus particulièrement relatifs à la reconnaissance et à la synthèse de la parole, et leurs liens avec les animations faciales.

Nous nous sommes ensuite intéressé à l'environnement de développement d'interfaces Morphic, et nous avons pu constater que l'entité graphique de base de Squeak, le morph, avait des possibilités qui dépassaient largement celles des interfaces multi-fenêtres usuelles. Les capacités graphiques de Squeak ne se limitent d'ailleurs pas à la 2D, et Alice, un outil de visualisation et d'animation 3D, nous a permis de le constater.

Nous avons présenté les outils Internet et Web : la gestion des protocoles de courrier électronique, et l'outil qui permet de les gérer, Celeste, mais aussi la gestion des protocoles HTML et HTTP, et l'accès au Web avec Scamper. Enfin, nous avons évoqué la possibilité de créer des sites collaboratifs dynamiques au moyen des sites Swiki.