# 1 章 Squeak とは

この本は，Squeak を使い個性的で面白いプログラムを作りたいという人向けの入門書です。本書を読むことで，自力で Squeak のプログラミングができるようになります。

しかし，最初からプログラミングというわけにはいきません。1 章では，Squeak の魅力，インストールの仕方を解説します。その後，Squeak を立ち上げ，その世界へと入っていきます。

## 1.1 Squeak をはじめよう！

Squeak については，最近は SqueakToys と呼ばれる「タイルスクリプティング」環境が，話題になっています。テキストによるプログラムを一切書かずに，命令の書かれたタイルを貼り付けていくことで，自分で描いた絵を自由に動かしたりできます。これはこれで非常に楽しく，プログラミングが初めてという方には大変喜んでもらえます[1]。NHK の「未来への教室」という番組で紹介され，京都では実際に小学生たちを参加させワークショップが開かれたりもしています。プログラミングの楽しさをわかりやすい形で伝えるという点では，SqueakToys はまさに理想的なものといえるでしょう。

しかし，タイルスクリプティングのみが Squeak の魅力ではありません。タイルスクリプティングを動かしている「仕組み」といったものが存在します。それは "Smalltalk" というシンプルでダイナミックな性質を持つオブジェクト指向言語です。「Smalltalker(Smalltalk を自在に操る人)になる」ことこそが本書の提示する目標です。子供のころ機械のフタを開けて見ずにはいられなかった，そんな人であれば，本書のスタンスがわかっていただけることでしょう。

プログラミングを覚えたてのころ，「これさえあれば何でもできる」という熱い思いにと

[1] 興味のある方は，Thoru Yamamoto 著「スクイークであそぼう」(翔泳社刊)をご覧になるとよいでしょう。

らわれたことはなかったでしょうか。いつの間にか，そうした気持ちはなくなり，普通のプログラムを無難に書く日々にとらわれてしまっているのであれば，非常に悲しいことです。Smalltalk には，他のプログラミング言語に見られるような余計な制限はありません。想像力のおもむくまま，好きなことが好きなようにできます。ぜひこのユニークな環境をものにして, 本来のプログラミングの楽しさ, 力といったものを再発見していただければと思います。

### 1.1.1 なぜ Squeak か？

Smalltalk のフリーな実装は, 非常にたくさんあります。その中で Squeak を選ぶのには, いくつか理由があります。

■マルチプラットフォーム

Squeak は Windows，Mac，UNIX などの主要なプラットフォームで動作します。速度は遅くなりますが，Zaurus や iPAQ などの PDA でも動かせるくらいです[2]。Web ブラウザ上で動くプラグインもありますので，自分が作ったプログラムを広く使ってもらうには，まさに適しているといえるでしょう。

■オープンソース

Squeak のソースコードは，エンジン部分であるバーチャルマシンも含め，すべて公開されています。特定のベンダが, 重要な部分のソースコードを掌握しているようなことはなく, 本当にあらゆる部分のコードを見て，カスタマイズしていくことが可能です。

■コミュニティが熱い

Squeak は，商用の Smalltalk が持っていた，様々なしがらみを断ち切ったところからスタートしました[3]。そのため，自由な場所を求めていた往年の名 Smalltalker たちが，返り咲いたようなところがあります。熟年パワーおそるべしです。また，1996 年からインターネットを通じて広く公開されたこともあり，全く知らないところから単に面白そうだから飛び込んでいき, マスターとして認められるようになった人もたくさんいます。世代を超えて, 様々な人々が Cool なものを求めて楽しんでいるのです。

Squeak Foundation が管理するメーリングリスト Squeak-Dev[4]では，毎日 100 通近くの

---

2　PocketPostPet 上に Squeak を載せた例もあります。

3　詳しくは，http://www.mars.dti.ne.jp/~umejava/smalltalk/squeak/index.html をご覧ください。

4　http://lists.squeakfoundation.org/listinfo/squeak-dev

メールが飛び交っています。バグのレポートから，機能拡張，昔話や新しいアイデアなど，話題は尽きることがありません[5]。

■マルチメディアの扱いに長ける

Squeak が新たな Smalltalk の実装として世に出たとき, マルチメディアに関する機能が, 大幅に強化されました。少ないコードで，映像や音楽を非常に簡単に操ることができます。**図 1.1** は，アニメーションのキャラクタを表示させて，デュエットで「きよしこの夜」を歌わせるものですが，このようなプログラムが，わずか 10 数行で書けます。


図 1.1 Squeak で「きよしこの夜」を歌わせているところ

Squeak 以外の Smalltalk のフリーの実装は，多くはビジネスアプリケーションの方向に向いてしまっているので，これほど簡単に映像，音楽を扱うことはできません。これも Squeak が持つ楽しさの 1 つでしょう。

## 1.2 Squeak の入手とインストール

それでは，いよいよ Squeak を手に入れることにしましょう。基本的にはインターネット

[5] 日本人向けには Squeak-ja があります。http://www.smalltalk.jp/mailman/listinfo/squeak-ja

にアクセスし，必要なファイルをダウンロードして展開すればインストールは完了です。

本家のサイトは http://www.squeak.org なのですが，導入のしやすさから，本書では日本語版の Squeak を用いることにします[6]。

それでは，日本語版の Squeak が置いてあるサイトにアクセスすることにしましょう。

「日本語版 Squeak でプログラミングしよう」
http://swikis.ddo.jp/squeak

各プラットフォーム用に，必要なファイル群がアーカイブとなって置かれています。基本的には，ダウンロードして展開すればインストールは終了します。

以後は，Windows 版を入手したとして話を進めていきます。他のプラットフォームについては，適宜読み替えるようにしてください。展開すると，以下のようなファイルが出来上がります。

- Squeak.exe
- SqueakNihongo6Dev.image
- SqueakNihongo6Dev.changes
- SqueakV3.sources
- ImmWin32Plugin.dll

後に詳しく説明しますが，Squeak.exe というものが，Squeak を動かすためのエンジン―「バーチャルマシン」になります[7]。Squeak がマルチプラットフォームで動作するのは，このバーチャルマシンがプラットフォームの違いを吸収してくれるからです。

バーチャルマシンが動かす Squeak の本体は「イメージ」と呼ばれます。

---

[6] 大島さんや阿部さんといった，日本を代表する Squeaker らの手によって，多言語化，ローカライズがなされています。

[7] Mac では，Squeak 3.x.app(x はバージョン番号)，Linux では，squeak というファイルになっています。

SqueakNihongo6Dev.image というファイルがそれに該当します。これは，Windows であろうが Mac であろうが共通になります。

起動は，このイメージをバーチャルマシンに与えるだけです。ドラッグ＆ドロップで SqueakNihongo6Dev.image を Squeak.exe に重ねます。

UNIX 系の方は，展開されたディレクトリに移動して，

```
squeak SqueakNihongo6Dev.image
```

などとするとよいでしょう。

図 1.2 のような画面が立ち上がってきます。

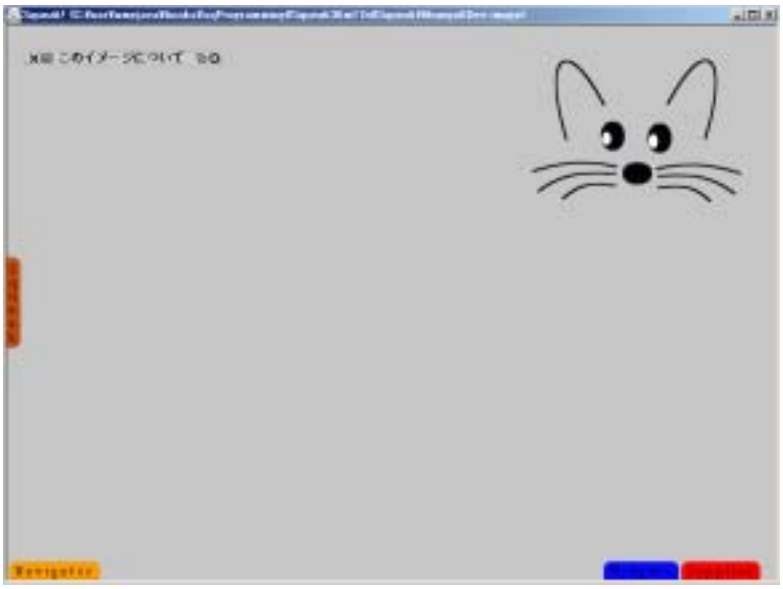

図 1.2 Squeak Nihongo6 開発版の起動画面

すぐにでも触ってみたいところですが，そこはぐっと押さえて，まず終了の仕方を習いましょう。車の運転もそうですが，ブレーキの仕方を知らないと，危なくて走らせるわけにはいきません。Squeak はそれこそなんでもできる環境なので，安全に終わらせるところから始めないと，環境そのものを破壊してしまう可能性もあるのです。

グレーの背景をクリックするとポップアップメニューが出てきますので, 一番下の "quit" を選びます(図 1.3)。セーブするかどうか聞いてきますが，まだ何もしていませんので保存の必要はありません。"No" を選んで終了します。

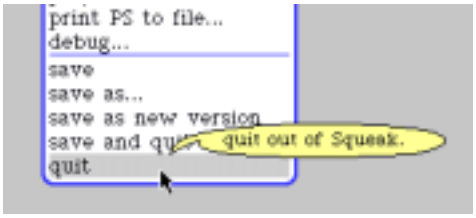


図 1.3 ポップアップメニューから "quit" を選ぶ

### 1.2.1 バーチャルマシンとイメージ

さて, あらためて「バーチャルマシン」と，「イメージ」について説明しましょう。Squeak は，バーチャルマシン(VM)上で，バーチャルイメージ(VI)が展開されて動作するという形式をとります。ここでの「イメージ」は，「画像」という意味ではなく，「生き写し，そのもの」といった意味です。イメージの中には，Squeak を構成するあらゆるものが封じ込められています。開発環境，クラスライブラリ，動作中のプログラムなどが，オブジェクトの無数の固まりとなって 1 つのバイナリファイルに収められているのです。

ノート PC などで「ハイバネーション機能」というものを使ったことがあるでしょう。これは RAM の内容をごっそりハードディスクに移して，PC を落としてしまうというものです。再開すると，メモリ内容は再び RAM 上に読み込まれ，電源を落とす前の状態で PC の画面がよみがえってきます。これと同じようなことが Squeak でも行われているのです。Squeak の世界は，普段はディスク上のファイルとしてフリーズドライされており，VM に引数として与えられることにより再び展開されて動き始めるのです。

セーブを行うと，新たなイメージが作られます。セーブ時の Squeak の状態が，そっくりそのまま, ある瞬間で写真を撮ったように保存されるので, イメージ保存は「スナップショット」とも呼ばれます(図 1.4)。

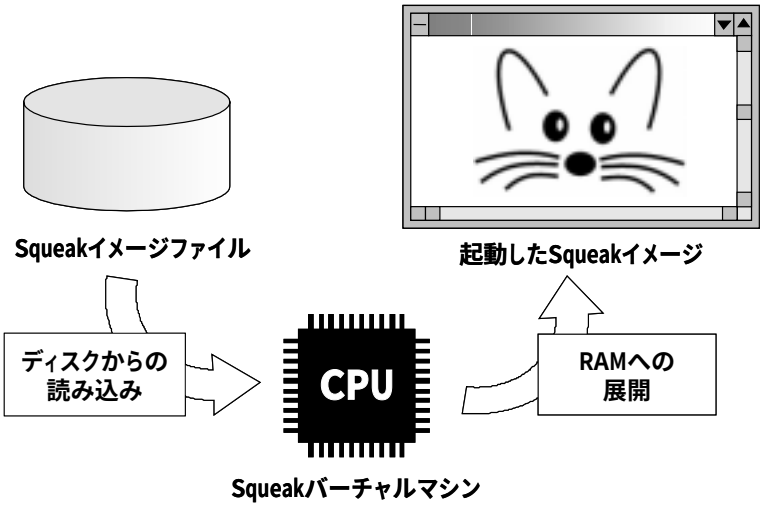


図 1.4 イメージの展開

スナップショットの様子を見るため，Squeak を再び起動し，いろいろと Squeak の画面をいじった後で，今後は別名で保存してみましょう(**図 1.5**)。


図 1.5 "save as..." で Squeak を別名保存する

新たな名前のイメージファイルができているはずです。これを起動すると，スナップショット時の状態が再現されることがわかるでしょう。

さて，このイメージファイルですが，各プラットフォームで，完全に共通のものが使えます。複数のプラットフォームを持っている人は，イメージファイルを別のプラットフォーム

に移して起動してみるのもよいでしょう。今回は，初心者向けにバーチャルマシンとイメージが両方とも含まれているバージョンの Squeak をダウンロードしたわけですが，実はイメージに関しては共通で使えるのです。そのため，本家の Squeak 開発用の FTP サイト(ftp://st.cs.uiuc.edu /Smalltalk/Squeak/)に行くと，個別でファイルがダウンロードできるようになっています。

### 1.2.2 ソースファイル

そのほかのファイルについても，少し見ていくことにしましょう。先ほど，イメージファイルには，あらゆるものが封じ込められているというような書き方をしました。しかし，実際にはイメージのサイズを節約するため，ソースコードに関しては別ファイル(ソースファイル)に保存されています。

Squeak 3.x の場合, SqueakV3.sources というファイルがソースファイルです(Squeak がバージョン 4.x に進んだときには SqueakV4.sources という名前になるでしょう)。バーチャルマシンと同じディレクトリにソースファイルがないと，Squeak は，起動時に「ソースファイルが参照できない」と文句を言ってきます。必ず同じ位置に置いておくようにしましょう。ソースファイルはあくまでソースコードを参照する開発時用のもので，単に Squeak を動かすだけであれば，必要ではありません。リソースの限定される PDA で Squeak を走らせるだけ(その中で開発はしない)といったようなときには，イメージファイルとバーチャルマシンだけあれば十分です。Web ブラウザ上でアプレット的に動かすときも同様です。

ソースファイルには，Squeak のクラスライブラリのソースコードがテキスト形式で収められています。イメージにはソースコードそのものは入っていません。イメージ側にあるのは，ソースポインタと呼ばれる，ソースファイルの特定位置を覚えているポインタなのです。そのため，ソースファイルが何らかの原因で壊れたりすると，イメージ側から正しい位置のソースコードを参照できなくなり，表示がおかしくなってしまうことがあります(変数名がすべて t1，t2 などという名前になります)。ソースファイルはテキスト形式ですが，決して通常のテキストエディタで編集などしてはいけません。改行コードが変わってしまう場合があるからです。FTP サイトからのダウンロード時や展開時にも，改行コードを変えてしまう親切すぎるツールが多いですから，イメージ上でうまくソースが表示できないような場合，まずこのことを疑ってみてください。

### 1.2.3 チェンジファイル

ソースファイルには Squeak のクラスライブラリのソースが格納されていました。これに対し，自分でプログラムを書いていったソースが書き込まれていくのがチェンジファイルです。チェンジファイルは一種のログであり，私たちが Squeak 上で行った，プログラムの追加, 変更, 削除, 重要な操作, などが逐一書き込まれていきます。チェンジファイルはイメージファイルとセットで作られます。イメージを別名で保存すると，各イメージファイルに対応した名前のチェンジファイルが自動的に出来上がります。

商用の Smalltalk では，チェンジファイルはインストール時に空になっているのが普通ですが, Squeak の場合は, 最初から付属してくるチェンジファイル(Squeak3.x.changes)に，既に情報が書き込まれています。これは，世界中の開発者が現行バージョンの Squeak に加えていった変更なのです。メジャーバージョンアップすると，変更は正式にソースファイルへと移されます。自分用の開発を行うときは，インストール時の状態と区別するために，イメージを別名で保存し，専用のチェンジファイルを作ります。

チェンジファイルはプログラミング中の重要な変更をすべて保存しているので，ソース管理のためには非常に重宝します。Squeak が不慮の事故で落ちてしまったときも，このファイルがあれば，落ちる直前の状態まで，復旧ができるようになっています[8]。

### 1.2.4 VM プラグイン

VM プラグインとは, VM を補助するためのプラットフォーム固有のファイル群です。DLL やシェアドライブラリの形で供給されます。これは VM の機能を拡張したり，変更したりするための仕組みです。例えば Windows 版の ImmWin32Plugin.dll は，IME を使った日本語入力機能を VM に組み込みます。

プラグインは，パフォーマンスチューニングなどにも使われることがあります。例えば暗号のエンコード，デコードなどで，高速な計算処理が必要なときなどは，その部分のみを C で書き，Squeak 側から呼び出すことができます。

[8] 詳しくは，9 章に書いてあります。

## 1.3 環境に親しむ

では, 一通り各ファイルが何のためのものなのか, 理解できましたので, そろそろ Squeak の世界を探検してみることにしましょう。Squeak を再び起動してみてください。

### 1.3.1 マウスによる操作

Squeak では，マウスによる操作を非常によく行います。そこで，まずマウスの使い方を説明しておきます。

使用するマウスは 3 つボタンが想定されています。ホイールのあるマウスの場合には，ホイールボタンを中ボタンとして使用します。1 つボタンである Mac などの場合は，キーを同時押しすることで代用します。

表の形でまとめると，ざっと以下のようになっています。位置については，マウスの設定により入れ替わっている場合があるかもしれません。

| ボタン | 通称 | 位置 | 同時押しキー | 主な機能 |
|---|---|---|---|---|
| 1 ボタン | レッドボタン | 左 | なし | 画面上のものを選択する |
| 2 ボタン | イエローボタン | 右 | Option + 1 ボタン | ポップアップメニューを出す |
| 3 ボタン | ブルーボタン | 真ん中 | Alt(Mac では Command) + 1 ボタン | Halo(ハロー)を出す |

色で呼ばれているのは，Smalltalk が専用機で動いていた頃のなごりだとか。実際にこの色でマウスボタンが塗られていたようです。

では，なにか画面上のものをマウスでいじってみることにしましょう。

まず目立つのは，キョロキョロしているネズミですね。Squeak のマークです(**図 1.6**)。

図 1.6 カーソルの方向を見つめるネズミ

レッドボタンでクリックして(通常の左クリック)つかんで動かすことができます。透明なのでちょっと面白いですね。このように，画面上のものを単純に「選択」するのにレッドボタンをよく使います。

では移動だけでなく，回転や拡大など，少し特殊な操作をしたくなったときにはどうすればよいのでしょう。そのようなときにはブルーボタンを使います。

ネズミをブルーボタンでクリックしてみてください。

カラフルな丸いマークがネズミの周りに現れます。これは Halo(ハロー)と呼ばれています[9]。それぞれのマーク(ハンドルといいます)をつかんだり, クリックしたりすることで, 様々な操作を行わせることができます(**図 1.7**)。


図 1.7 ハローに取り囲まれるネズミ

[9] Halo は「後光」という意味です。ネズミに後光が差したというわけですね。

青いハンドルは回転用です。これをつかんで動かすと，好きな方向に回転させることができます(**図 1.8**)。


図 1.8 青ハンドルを使ってネズミを回転させる

また，黄色のハンドルは，拡大，縮小用に使います(**図 1.9**)。


図 1.9 黄ハンドルを使ってネズミのサイズを変える

いろいろと楽しんでみてください。メニューやウィンドウすら回転させたりできます。

Squeak の中に表示されているものは，メニューやウィンドウも含めてすべてモーフと呼ばれています。モーフであれば，ハローを出して，様々な操作を加えることができます。

ハローの詳しい使い方については，7 章の Morphic 入門でゆっくりと見ていくことにします。今の段階では，基本的なハローの出し方がわかれば十分です[10]。

最後はイエローボタンです。これは開発ツール上でポップアップメニューを出すためのものです。プログラミングに慣れた段階で，嫌というほど使うことになります。

先ほどセーブ時に出したポップアップメニューは，例外的で，何もない所でレッドボタンを押すことで出てきました。これは Squeak 全体に対するメニューで，「ワールドメニュー」と呼ばれています。一方，開発ツール上では，イエローボタンでポップアップメニューを出すのです。

### 1.3.2 キーボードによる操作

キーボードは，皆さんが大好きなコーディングを行うためのものです。Squeak では多くのキーボードショートカットが用意されているので，慣れるに従い，効率の良い操作ができるようになります。ショートカットは，Alt（Mac では Command。以降，Mac の人は Command と読み替えてください）キーと組み合わせて使うものがほとんどです。

試しに，デスクトップ上で Alt + b を押してみましょう。開発ツールの 1 つである「システムブラウザ」が開きます。ブラウザの下の部分に適当にタイプしてみましょう（図 1.10）。

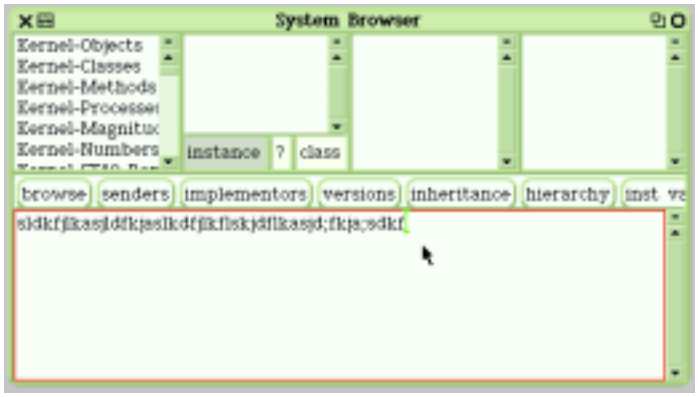


図 1.10 システムブラウザに文字列を書き込む

[10] 7 章にハローの操作がまとめてあります。

打ち込んだ文字列全体の選択は Alt + a になります。コピーは Alt + c，ペーストは Alt + v です。

では，今度はでたらめな文字列を消して，Display reverse と打ち込んでみましょう。その後 Alt + d を押します。画面がネガポジ反転になります。これは "do it"(評価)と呼ばれる操作で，書いた式を評価させるものです。

Squeak では，ある対象に向かって，メッセージを送ることで，実行が行われます。この場合は Display(Squeak 画面全体)に，reverse(反転しろ)というメッセージを送ったことになります(画面を戻したい場合は，あわてずにもう一度 Display reverse を "do it" しましょう)。

"do it" したうえで，さらに実行結果を隣に表示するのが "print it" です。改行して 3 + 4 と書き，Alt + p を押してみましょう。計算の結果として 7 がすぐ隣に表示されます。

さらに改行して #(a b c)と書き，今度は "inspect it" をしてみます。Alt + i を押しましょう。灰色の小さな窓が開きます。これは式の実行結果を見るための「インスペクタ」というツールです。まだ Smalltalk の文法を説明していませんが，配列の中身が見えているということがなんとなくわかるでしょう(図 1.11)。


図 1.11 インスペクタ

実はいちいち改行しなくとも，マウスで実行したい部分のみをハイライト(選択)すれば，その部分だけを実行することができます。上記を改行せずに試してみるとよいでしょう。

"do it"，"print it"，"inspect it" は，Smalltalk の式を実行するための 3 つの主要な操作です。これらは，ポップアップメニューからも起動できます。先ほど使わなかったイエローボタンを押してみると，それらしきメニューが出てくるのを確認できるでしょう(図 1.12)。


図 1.12 ブラウザのポップアップメニュー

Squeak で多少特別なキーとして, _(アンダースコア)があります。これは Squeak では ← として表示されます。←マークは変数に対する代入という意味で，コードの中では頻繁に使用します。^(キャレット)も特別なキーになります。これは ↑ として表示されます。他の言語でいうところの return，値を呼び出し側に返すということを意味します。後で矢印キーをキーボード中に探さなくともよいよう，これらのキーに関しては, 気をつけておきましょう。

また，コーディングに欠かせないショートカットとしては Control + t と Control + f があります。これを押すと，ifTrue:，ifFalse: という文字列がそれぞれ表示されます。分岐を書くときにはよく使うので，これも覚えておきましょう。

さて，ここで書いたコードは，いわば下書きで，システムにはまだ伝わっていません。伝えるためには Alt + s を押します。これは "accept" と呼ばれる操作です。ただし，今はでたらめな文字列を書いているだけなので，システムが受け入れてくれるはずもありません。文法をマスターしたあとで，このショートカットは頻繁に使うことになるでしょう。

そのほかのショートカットについては，おいおい覚えていくようにしましょう。

### 1.3.3 自分用のイメージを作る

さて，いざプログラミングをしていく前に，まず自由に試せる自分用のイメージファイルを作ることにします。デフォルトのSqueakイメージをそのまま使っていると，何かあって初期状態に戻したくなったときに困ります。デフォルトのイメージは，最初から始めるときのマスターとして取っておきましょう。

イメージファイルを新規に作るには，ワールドメニュー(ルートウィンドウのメニュー)を出し，"save as..."(別名で保存)を選ぶのでしたね。名前は好きなものでかまいません(拡張子の .image は自動的に付きます)。自分の名前を付けたり，何かアプリケーションを作っているということであれば，その名前を付けたりします。

### 1.3.4 デスクトップの色，フォントの変更

せっかく自分専用のイメージを作ったのですから，使いやすいようにカスタマイズしてみましょう。ここでは，デスクトップの背景の色，フォントを変更してみることにします。

デスクトップの色は，ワールドメニューの "appearance..." → "set desktop color..." で変更できます。カラー選択ツールが立ち上がるので，マウスで好きな色を選択すると，デスクトップの色が変化します。

フォントの変更は，"appearance..." → "system fonts..." で行います。いくつか候補がありますが，"default text font...", "list font...", "menu font..." あたりを変えておけばよいでしょう。

### 1.3.5 プロジェクトに入る

今度は，Squeakの中に「プロジェクト」を作ることにします。

「プロジェクト」とは，イメージの中をさらに仮想的に分割したようなものです。プロジェクトを使うと，デスクトップの状態や，ソースコードの変更などをプロジェクトごとに持てるようになるので，非常に便利です。

Squeakの初期画面では，既にウィンドウが立ち上がっており，このままでは自分が開いたウィンドウと区別がつかなってしまいます。そこで，新たなプロジェクトを作り，以後はその中で作業することにします。

ワールドメニューから，"open..." → "morphic project" を選択してください(**図 1.13**)。


図 1.13 新しく morphic プロジェクトを作る

さて, 選択すると, **図 1.14** のような小さなウィンドウが立ち上がります。最初はプロジェクトに名前が付いていませんので, 'Unnamed1' と表示されています。ウィンドウの下の部分を選択すると, 好きな名前を入れることができます。'MyProject' とでも付けておきましょう。


図 1.14 プロジェクトに名前を付ける

クリックして新規プロジェクトの中に入ると, まっさらなデスクトップが広がっています (**図 1.15**)。

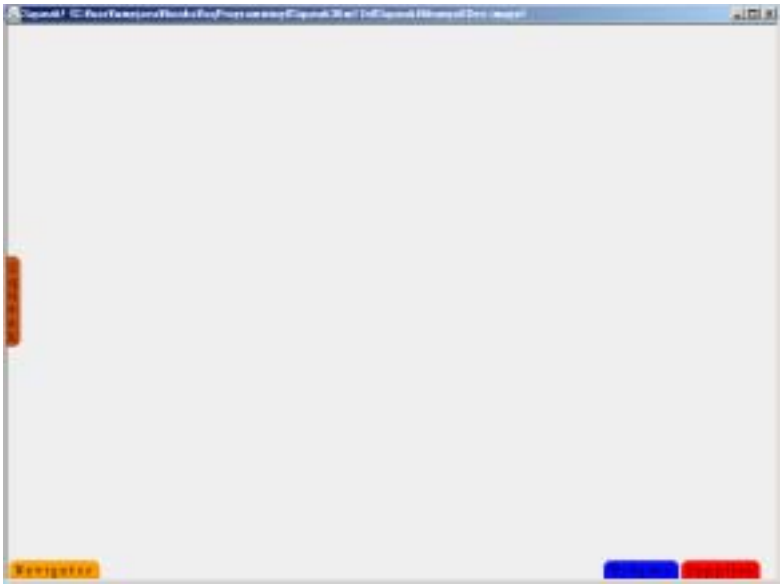

図 1.15 新しいプロジェクトに入る

プロジェクトは，個人の作業環境です。何か新しいことをすると決めたなら，そのつどプロジェクトを作って，そこに入って作業する癖をつけるとよいでしょう。

ワールドメニューの "jump to project..." を選ぶと，好きなプロジェクトにジャンプすることができます。単に，前のプロジェクトに戻りたい場合は，"previous project" を選択します。

## 1.4 開発ツールに慣れる

Squeak は，単なるプログラミング言語というよりは，ミニ OS と呼べるものです。ウィンドウやメニューを自ら表示しますし，プロセスのスケジューラすら自分で持っています。イメージには, コンパイラや, ブラウザといった開発ツールも含まれています。さらにディープになると，VM そのものを生み出すためのトランスレータまで用意されているのです。

Squeak でのプログラミングは，まずエディタでコードを書き，その後コンパイルしてできた実行形式を何らかの OS 上で動かすという，いわゆる普通のプログラミングスタイルとは，かなり違っています。最初に起動したときの環境を，プログラミングによって自分の思い描く世界へと変えていく「OS 育てゲーム」と考えたほうがしっくりきます。いわば Squeak は，皆さんのプログラミングでどのようにでも育っていく生き物なのです。

生物にたとえたのは，Squeak が非常にインタラクティブな環境だからです。こちらのちょっとした働きかけに対して，必ず何らかの反応を返してきます。この反応を観察して，次にどうすればいいのか考え，また別の働きかけをするという小さな繰り返しにより，プログラムができていくのです。

内蔵のツールは，Squeak とやり取りするための，重要なインタフェースです。プログラミングに慣れてきた人ほど，外部の汎用エディタで Squeak を書いて覚えようとする罠に，はまってしまいがちです。このようなやり方では，Squeak の良さはわからないでしょう。Squeak から十分なフィードバックが得られないのですから。

最初こそ奇妙に思えるかもしれませんが，Squeak は次第に自分になじんでくるようになります。ツールが使いにくかったとしたら, それをも変えていくのが正しいやり方です。もっとも最初はそんな高度なことはできないので，むしろ Squeak のほうに「いじめられる」感じも受けるでしょう。しかしそれを乗り越え，手なずけると，本当に「すべてをコントロールできる世界」がコンピュータ上に出来上がります。そのための手綱(たづな)が開発ツール群なのです。

### 1.4.1 実行の窓 ― ワークスペース

ワークスペースとは，ちょっとした Squeak プログラムを実行するための，簡単なメモ帳のようなものです。メモ帳と違うのは，インタラクティブなところです。様々な Smalltalk の式を書いて，実行させることができます。

ワールドメニューから "open..." → "workspace..." で開けることができます。**図 1.16** のような何の変哲もないウィンドウです。

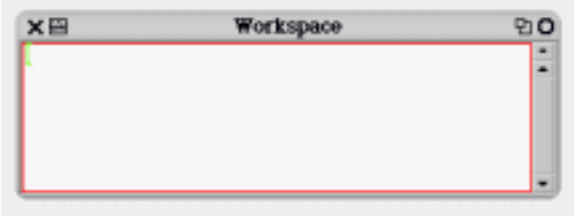


図 1.16 ワークスペース

では，Smalltalk の式を実行してみましょう。Squeak の中には，気軽に実行できるサンプルがたくさん入っているのです。サンプルを持っている対象に向かって，メッセージを送ることで，実行がスタートします。Pen example と打ち込んで，"do it" してみましょう(**図 1.17**)。

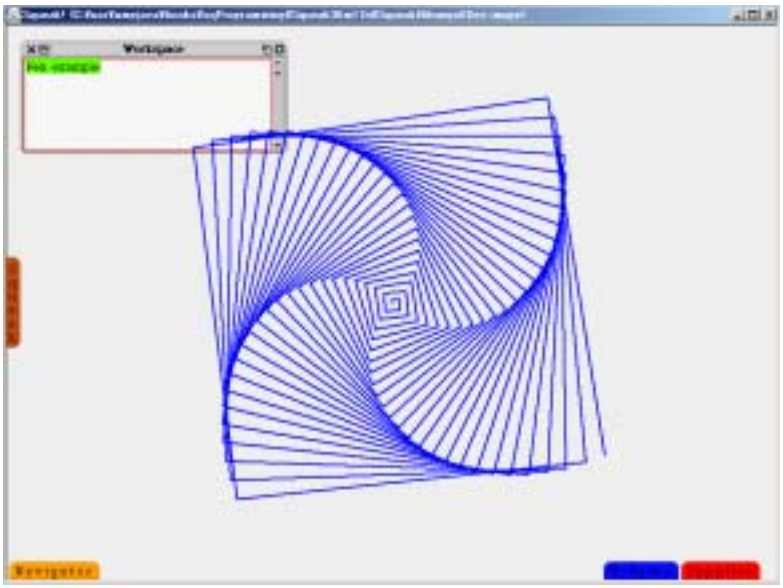

図 1.17 Pen example の実行結果

スクリーンにきれいな模様が描かれましたね(画面に直に描いているので，表示が残ったままになります。気になる方はワールドメニューから "restore display" で消してください)。

ワークスペースは変数を持つことができます。変数の代入は ← で行うのでしたね。では，ちょっと簡単なプログラムを書いてみましょう。

```
joe ← Morph new.
joe openInWorld.
joe flash.
joe delete.
```

わからないままに，1 行ずつ順番に "do it" してみましょう。1 行目が変数の代入で，joe という変数にモーフを新しく作って入れています。2 行目で，作った joe を表示させています。3 行目は，joe にピカピカ光れと命令しています。4 行目は，joe を画面から消します。ワークスペースでは，このように小さなプログラムを，結果を見ながら実行させていけるのです。

### 1.4.2 表示の窓 ― トランスクリプト

トランスクリプトとは，文字列を表示させるためのウィンドウです。他のプログラミング言語での標準出力のような使われ方をします。ワールドメニューから，"open…" → "transcript" を選択すると，オレンジがかったウィンドウが立ち上がります(**図 1.18**)。

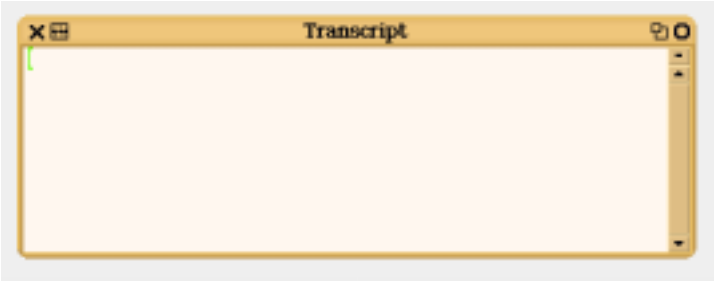


図 1.18 トランスクリプト

今度は，プログラミング入門で典型的な，Hello World を実行してみることにします。
ワークスペースに，以下のように打ち込んで "do it" してみてください(図 1.19)。

```
Transcript show: 'Hello World'
```

(これは「Transcript に，'Hello World' という文字列を表示せよ(show:)という命令を送っています)。

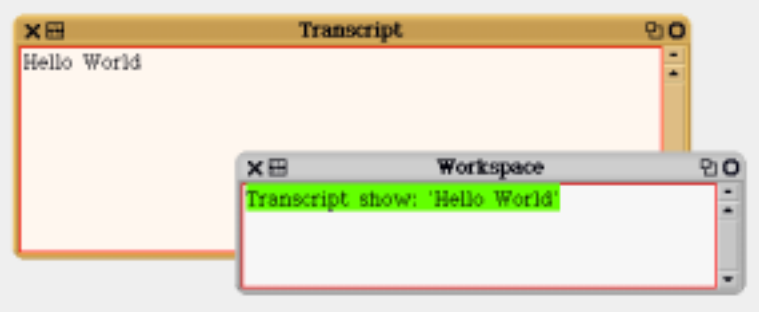


図 1.19 最初の Hello World プログラムの実行

表示が行われました。コードの隣に表示してしまう "print it" に対して，トランスクリプトは別ウィンドウとして開き，文字列でいっぱいになっても勝手にスクロールしてくれるので，次々に結果を見たいようなときに便利です。

### 1.4.3 修正の窓 — デバッガ

Squeak はインタラクティブな環境なので，多少のエラーならば，それを類推して，直してくれます。

例えば，Transcriptt show: 'Hello World' と t を続けて書いてしまい "do it" した場合，Squeak は，図 1.20 のような警告を出してきます。

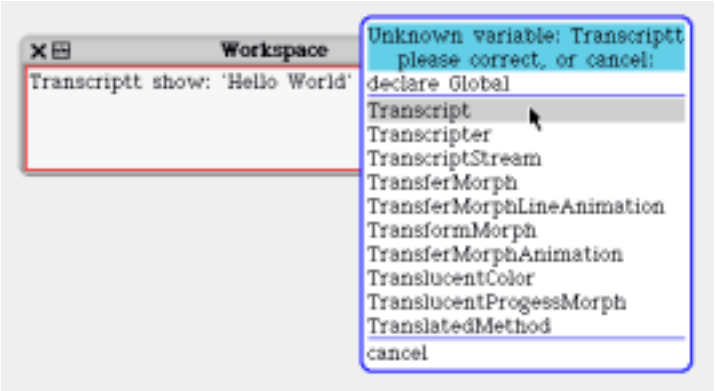


図 1.20 修正を促す Squeak

「Transcriptt？　定義されていない変数です。似たような候補があります。どれにしますか？　それともグローバル変数として新規に定義しますか？」と聞いているのです。これで Transcript をリストから選べば，修正が行われ，実行を続けていくことができます。

文法的なちょっとした誤りも指摘してくれます。例えば 'Hello World' の最後のシングルクォーテーション(')を，ダブルクォーテーション(")にしてしまったとします。この場合，Squeak は，「引用符があっていないですよ」と警告を出してきます(**図 1.21**)。

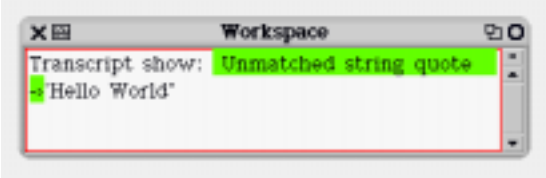


図 1.21 文法的な誤りの警告

この場合はバックスペースで警告を消して，修正できます。

もっとも，Squeak がとうてい対応できないような書き間違えのケースも起こり得ます。Transcript の先頭 T をわざと選ばずに "do it" してみましょう。文法が合っているので，"do it" によって実行を始めたものの，実行の途中でうまく進めなくなってしまいます。このようなとき Squeak は，「ここでエラーが起きました。続けますか？　中止しますか？　デバッグしますか？」というノーティファイア(警告用ダイアログ)を出してきます(**図 1.22**)。

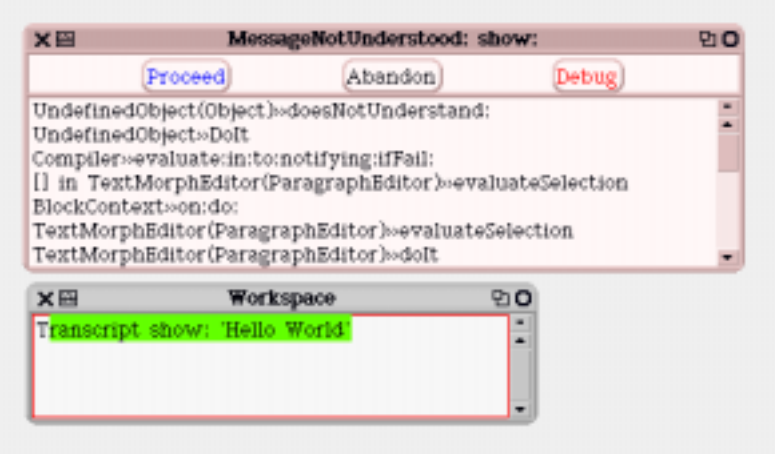


図 1.22 ノーティファイアの出現

初心者の方には，このピンクのウィンドウがたいそうショッキングに思え，これを見て Squeak をやめてしまうという人もいるほどです。しかしこれは，Squeak が次にどうすればよいかを聞いているだけなのです。

ここで "Debug" ボタンを押すと，デバッガが開き，デバッグを行っていくことができます(**図 1.23**)。

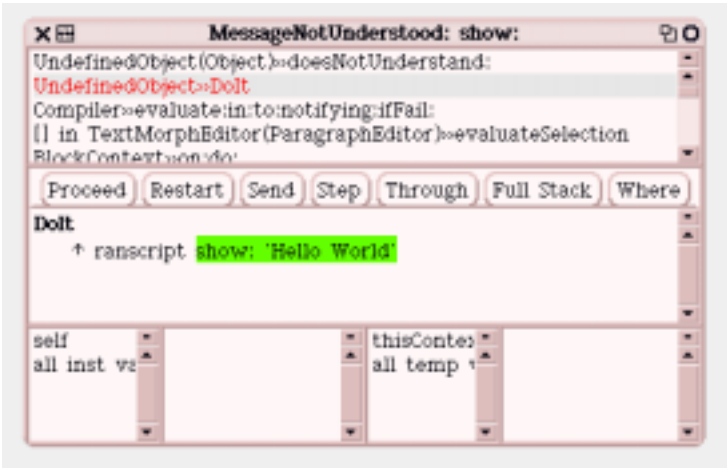


図 1.23 Squeak のインタラクティブなデバッガ

実はこのデバッガ，皆さんが今まで使ったことのあるデバッガの中でも，そして Squeak の開発ツール内でも類を見ないほどの強力な機能を備えています。どうすればいいかわからなくなったとき，あえて動かないコードを実行させ，デバッガの中で開発するという流儀(Just in Time Programming)も Smalltalk の中にはあるくらいです[11]。

今の段階では，「ショッキングピンクのウィンドウが開いても驚かない」ようにしていればよいでしょう。

参考までに簡単に説明すると，上のリストが実行スタックを表しています。下から上に向かって実行が積み重なっています。DoIt のあと，doesNotUnderstand: に至り，エラーが起こったというわけです。左下，右下はメッセージの受け取り側や引数の値を見るためのインスペクタになっています。

ここでは，デバッガは何もせずに閉じてしまいましょう(左上の×印を押します)。

### 1.4.4 探求，創造の窓 ― システムブラウザ

ワークスペースがちょっとしたプログラムを書いて試すためのものだとすると，本格的なプログラム開発は，どこで行われるのでしょうか？　答えは「システムブラウザ」です。Squeak でプログラミングを行うときに，最も使い込むことになるのがこのツールです。システムブラウザは，ワールドメニューからは "open..." → "browser" で立ち上がります(図 1.24)。

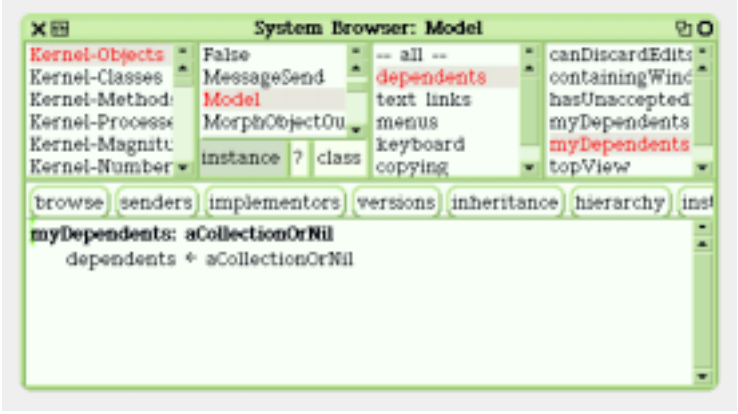


図 1.24 システムブラウザ

[11] Kent Beck という XP で有名な達人の考案したスタイルです。

システムブラウザとは，その名の通り，Squeak のシステムの中身をブラウズしていくためのものです。Squeak には，「クラス」というプログラミング部品が 1500 以上も格納されています。そのため自分の目的に合ったものがないか，探しながらプログラミングを行っていくのです。目的にかなったものがないときには自分で部品を作ることになりますが，その場合でも 0 からスタートするのではなく，既存のクラスをうまく利用できる仕組みがあります。

システムブラウザは，Squeak でプログラミングするには不可欠のツールですが，使いこなすためにはオブジェクト指向の知識が不可欠となります。詳しくは，後の章にゆずることにしましょう。