

4G-3

Rabbit: プログラマのプレゼンテーションツール

(Rabbit: Programmer's Presentation Tool)

須藤功平*
岩手大学†

西谷泰昭‡
岩手大学§

1 はじめに

使いやすいアプリケーションを開発するには、使用ユーザを限定し、そのユーザの利便性を意識するとよい。誰もが使いやすいように開発されたアプリケーションよりも、特定のユーザのために開発されたアプリケーションの方がそのユーザにとって使いやすい。

現在、プログラマのために開発されたプレゼンテーションツールは存在しない。また、プログラマに好まれそうな既存のプレゼンテーションツールではプログラマは満足しない。本稿では、プログラマのプレゼンテーションツール Rabbit を提案する。

ここで、プログラマとは以下のような人物であり、その代表的なモデルは筆者自身である。

- Visual Studio などの IDE よりもカスタマイズ可能な Emacs や Vim などのエディタを好む。
- マウス操作よりもキーボード操作を好む。
- 高度なカスタマイズ性を好む。

プレゼンテーションツールは大きく WYSIWYG 型とテキスト型のふたつに分けることができる。

WYSIWYG 型は OpenOffice.org の Impress, GNOME の Criawips, KDE の KPresenter, Apple の Keynote, Microsoft Office の PowerPoint のようにアプリケーション上でスライドの作成から表示までを行うタイプのプレゼンテーションツールである。WYSIWYG 型のプレゼンテーションツールではマウスを用いて GUI でスライドを作成するものが多い。

テキスト型は MagicPoint, Pyslide, prosper, less プレゼンテーションのようにスライド用のソースファイルはアプリケーションとは別の任意のエディタで作成し、そのソースを用いてアプリケーションがスライドの表示を行うタイプのプレゼンテーションツールである。

本稿ではプログラマのための以下の条件を満たすプレゼンテーションツールを提案する。

- テキスト型
- ソースと見栄えの分離
- 豊富なキーバインド
- 高いカスタマイズ性

2 テキスト型

プログラマは自分の手に馴染んだエディタを用いることを好む。そのため、スライド作成には WYSIWYG 型の編集インターフェイスをもったプレゼンテーションツールよりも自分の好きなエディタを用いるテキスト型を好む。また、既存の各種テキスト処理用ユーティリティやバージョン管理のしやすいテキストファイルを好む。よって、Rabbit ではテキスト型を採用する。

Rabbit がテキスト型を採用したことにより、プログラマは効率よくスライドを作成 / 校正 / 管理できるようになる。

まず、スライド作成の場合を考える。プログラマは愛用しているエディタがある。GUI でマウスクリック、テキスト入力、を繰り返してスライドを作成するよりも、使い慣れているエディタでスライドを作成した方が作業効率がよい。

次に、スライド校正 / 管理を考える。スライドのソースが単なるテキストであるため既存のツールをスライドの校正 / 管理に利用することができる。例えば、エディタの強力な編集機能はもちろん、grep によるテキスト検索や Subversion などによる版管理が行える。これらは強力に校正 / 管理を支援しスライド作成効率をあげるが、プログラマが日常的に使っているツール群であるため、学習コストがかからない。

テキスト型でのスライド校正作業は、ソースを更新し、スライドの表示結果を確認するという作業を繰り返す。

*Kouhei Sutou

†Iwate university

‡Yasuaki Nishitani

§Iwate university

返す。この繰り返しがスムーズに行えないと作業効率が格段に下がる。

Rabbit にはスライド表示中にソースを変更する機能がある。この機能はスライド校正中において非常に強力である。Rabbit を再起動しなくてもソースやテーマの変更が反映されるため、スライドの表示結果を素早く確認することができるため、作業効率が下らない。

3 ソースと見栄えの分離

テキスト型では、ソースにマークアップを行って、スライドの区切りや箇条書きなどを表現する。マークアップ言語の設計はプレゼンテーションツールによって大きく異なる。

Rabbit ではソースの見栄えからスライド表示が容易に想像できる人が可読なマークアップ言語を採用する。これにより以下のような利点がある。

- 実際にスライド表示をプレビューしなくてもスライド作成および校正が行える。
- ソースの編集が容易になる。

一方、ソースに詳細な指示を記述することが難しいという欠点がある。Rabbit では、ソースに見栄えのための詳細な記述はせずすむように、ソースと見栄えを別に指定する。これにより、テーマと呼んでいる見栄えの定義を複数のスライドで使いまわすことができる。

Rabbit にはスライド表示中にソースだけではなく、テーマを変更する機能もある。この機能も作業効率を下げることなくテーマ作成を行うことができるようになるため強力である。

4 豊富なキーバインド

プログラマはアプリケーションがキーボードによるインターフェイスを提供することを好む。

例えば、既存のプレゼンテーションツールのスライド移動は矢印キー、数字キー、スペース/バックスペースキーなどを用いる。これらは一般的なエディタやテキスト入力インターフェイスで提供されているキーを用いたわかりやすいインターフェイスである。しかしながら、プログラマが好んで使うエディタでは編集作業によく用いるカーソル移動のためにエディタ独自のキーバインドを提供していることが多い。例えば、Vim では h, j, k, l によるカーソル移動をサポー

トしているし、Emacs では C-b, C-n, C-p, C-f によるカーソル移動をサポートしている。プログラマにとって、これらのキーバインドでスライド移動ができた方が自然である。よって、Rabbit ではこれらのインターフェイスに加え、Emacs 風キーバインド、Vim 風キーバインドも提供する。

Rabbit はあらかじめプログラマの好む豊富なキーバインドを用意しているため、多くの場合、プログラマが望んだ動作をする。これによりプログラマの学習コストが小さくなる。

また、既存のプレゼンテーションツールではキーボードからはスライド移動、プレゼンテーション終了など、基本的なコマンドしか実行できない。Rabbit では、基本コマンドだけではなく、プレゼンテーションツールが提供するほとんどのコマンドをキーボードから実行可能にする。

このように、Rabbit は既存のプレゼンテーションツールに比べて、プログラマにとってより自然なインターフェイスを提供する。これは学習コストを下げ、作業効率を上げる。

5 高いカスタマイズ性

プログラマはアプリケーション自体をカスタマイズできることを好む。そこで、Rabbit を「本物の」プログラミング言語を用いてカスタマイズできるようにする。ベースとなる言語として動的な言語であるオブジェクト指向スクリプト言語 Ruby を採用する。

これにより、キーバインドの変更、テーマの定義、追加機能などを柔軟でかつ強力に行える。

テーマは「本物」のプログラミング言語で記述するため、単なる見栄えだけではなく、効果を表現することもできる。例えば、タイマーを表示したり、特定のスライドだけ特別に文字を大きくしたりすることができる。現在、もっともプログラマを引きつけているのはテーマの高いカスタマイズ性である。

6 まとめ

このように、Rabbit はプログラマの既存の資産を生かしてプログラマがより自然に使えるように作成されている。このため、Rabbit はプログラマにとって学習コストが低く、作業効率がよいプレゼンテーションツールであるといえる。

現在、Rabbit は <http://cozmixng.org/~kou/download/rabbit.tar.gz> で公開している。