Java アプレット / アプリケーションのための デモンストレーションヘルプシステム

A Demonstrational Help System for Java Applets/Applications

三浦 元喜[†] 田中 二郎[††]
Motoki MIURA Jiro TANAKA

†筑波大学 工学研究科
Doctoral Program in Engineering, University of Tsukuba
††筑波大学 電子情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

概 要

Java アプレットは，WWW ブラウザを用いることで誰にでも簡単に実行できる．しかし，その操作方法は文書で書かれることが多く，直感的に把握しにくい．本論文では，一般的な Java アプレットの操作を説明するデモンストレーションヘルプを作成するのに，実際の操作を用いて簡単に作成するための方法について述べる．本システムにおいては，低次元のイベント列から，操作の概要を表す文字列を生成するためのルールをまず準備する．それらのルールを用いることで，操作を行うだけで意味付けされたデモンストレーションが生成されるため，編集作業が容易になり，また開発者がヘルプ記述にかかる労力を軽減することができる．また，ユーザはデモンストレーションを眺めることでシステムの概要を把握することができる．

1 はじめに

ヘルプ機能は，アプリケーションの操作方法をユーザに示すために必要な機能である．従来は文章を主体とするヘルプが一般的であった．コマンド入力を主体とするアプリケーションでは文章によるヘルプでも問題はない．しかし，最近ではマウスやペンなどのポインティングデバイスを用いて操作するシステム，いわゆる GUI (Graphical User Interface) を用いたシステムが一般的である．このような GUI を用いたシステムにおいて，文章を主体とする操作説明はユーザが理解するのにかかる負荷が大きい．例えば，ユーザは「『ファイル』メニューの『開く』を選択」のような文を読み，アプリケーションの画面からそれらの「操作対象」を探して操作する必要があった．

文章の代わりに，アニメーションを用いてヘルプを提示する方法が提案されている [1] [2]が，実装のための手間がかかるため，いままで実際のシステムにはあまり適用されていない．

本論文では，一般的な Java アプレット / アプリケーションの操作を説明するデモンストレーションを，実際の操作から簡単に作成するための方法と，そのシステムについて述べる．Java アプレットは，Web ブラウザを用いれば誰でも実行できるので，ヘルプ機能の必要性は高い．その Web ページを初めて訪れる人にとってはアニメーションヘルプが用意されていればアプレットの概要を容易に把握できる．アニメーションヘルプは特に直接操作を多用したシステムの操作説明に有効であると考えられる．

本システムにおいては，特にヘルプ記述にかかる労力を軽減し，アプレットの開発者が簡単にヘルプを準備できることに注目している．そのため，アプレットやアプリケーション本体には変更を加えずに実現するという方針をとる．

2 デモンストレーションの記録形態

実際の操作から生成するデモンストレーションの記録形態として，動画像として記録するものと，システムが解釈できるイベントとして記録するものがある．ムービーファイルなどの動画像として記録した場合，再現するのに動画像専用のビューアを用いれば簡単に参照できる．しかし，システムの更新やメニュー構成の変化などのソフトウェアのバージョンアップに対応できない．また，ユーザのレベルに合わせて説明の粒度を調節することも難しい．我々は，ヘルプの編集のしやすさ，再生時の柔軟性などのメリットを考慮し，ヘルプとしてのデモンストレーションをイベントとして記録する方法を採用した．この手法による付加的なメリットとしては，一般にイベントとして記録されたものは動画像よりもデータサイズが少ないので，ネットワークから取得する Java アプレットのヘルプとしては適している．ヘルプを提示する際には，基本的には記録しておいたイベントを再現すればよい．

3 デモヘルプシステム “Jedemo”

このようなイベントの記録と再生の機能を一般的なアプレットに持たせるには，通常，個々のアプレットに対してそれらの機能を実装するしかなかった．我々は，この手間を省く方法として，「アプレットビューアモデル」という枠組みを提案し，一般のアプレットについてイベントの記録と再生を行うことができるシステム “Jedemo” を実装した．

3.1 記録と再生 ～アプレットの場合～

アプレットビューアモデルは，イベント管理機能を持たせようとするアプレット (以下，「対象アプレット」) が，イベント管理機構を持つアプレット (以下，「マネージャアプレット」) に貼り付けられて動作する機構である．図 1に，このアプレットビューアモデルの機構を示す．マネージャアプレットは，アプレットビューアの機能を持つアプレットであり，特殊なイベントリスナを対象アプレットの部品に登録することでイベントオブジェクトを取得できる．開発者は，マネージャアプレットを呼び出す際にパラメータとして対象アプレットのクラス名を指定するだけでイベントの記録と再生機能を追加することができる．

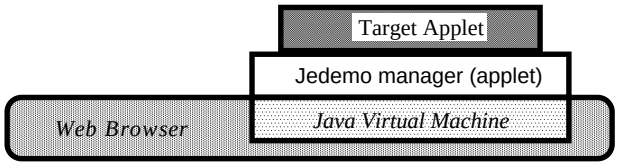

図 1 アプレットビューアモデル

3.2 記録と再生 ～アプリケーションの場合～

ヘルプの対象がアプリケーションの場合であっても，イベントを記録する方法についてはアプレットの場合と同じである．イベント記録専用のイベントリスナを各部品に登録することで行う．しかし，このイベントリスナを外部から登録する方法については，アプレットの場合と同じ方法を使うことができない．正確には，新しいウィンドウが生成され，画面に表示されたとき，その「ウィンドウが表示された」ことを瞬時に知る方法が用意されていない．そのため，新しいウィンドウ内の部品にイベントリスナを登録することが困難となる．このことは，特にアプリケーションの場合に問題となる．

アプリケーションの場合，ウィンドウを表示するのに主に Frame クラス (タイトル付きのウィンドウ) が用いられる．我々は，Frame クラスのクラスメソッド `getFrames()` を定期的に呼び出して，新規作成された Frame クラスのインスタンスを発見することで，ウィンドウ内の部品への参照を得る．

4 Jedemo による操作への意味付け

一般に操作を行うと，大量のイベントが生成される．これらのイベントの多くは「マウスを動かした」などの低レベルな意味を持つイベントである．デモンストレーションヘルプを編集するのに低レベルイベントで行なうのは効率が悪い．また低レベルイベント列の意味を理解していないと編集できない．

図 2　アイコンで表現されたイベント列

我々はイベント列をまとめた**コマンド**という概念を導入することで問題を解決した．1 つのコマンドは最低限の意味を持つイベント列である．コマンドには，その意味を表す**ラベル**を文字列として付加する．ラベルは，編集時のインデックスとして用いられるだけでなく，再生時にユーザに示す操作の概要として利用できる．

4.1　コマンドの生成

イベント列を解析して，コマンドを生成するために**コマンドルール**を用いる．コマンドルールは，対象アプリケーションの実装に基づき開発者があらかじめ準備する必要があるが，デモンストレーションのラベル付けを自動化できる．また，異なるアプリケーションでもたいていの操作は同じであるため，一度作成したコマンドルールの大部分は再利用可能である．

対象アプリケーションによって別々に用意しなければならないコマンドルールは，マウスのドラッグなどを多用したいわゆる「直接操作」の部分である．コマンドルールの定義は，実際のアプリケーションをまず操作し，その後，操作によって発生したイベント列を用いて行う (図 2)．

コマンドルールの例を図 3 に示す．この図では，

図 3　コマンドルールの例

4 つの定義されたコマンドルールが表示されている．アイコンで表示してあるものがイベント列のマッチングパターンである．アイコンの右の文字列がラベル生成ルールである．イベント列のマッチングパターンには，正規表現と同等の表現が扱える．図 3 の括弧で囲まれたイベント (ドラッグイベント) は，1 つ以上の任意のイベントにマッチする．

実際のイベント列からどのようにコマンドを生成するか述べる．まず，イベント列はセパレータによって，部分列に分解される．セパレータは通常，図 3 の一番上の列のように，「マウスカーソルが部品の上から出たときのイベント」「マウスカーソルが部品の上に入ったときのイベント」「マウスカーソルが動いたときのイベント」を指定しておく．これらのイベントを含まないように分解する．セパレータを用いてあらかじめコマンドの候補を絞りこむ理由は，イベント列の解析にかかる時間を短縮できることが挙げられる．分解したイベント列をコマンドルールによって 1 つずつ照合し，照合が成功したらラベル生成ルールによってラベル付けする．

ラベル生成ルールには，生成時の動的な情報をとりいれることができる．方法としては，動的な情報を挿入したい部分に，`@eventobj.method()` という記法で書く．`eventobj` の部分には，イベント指定子 (`add`, `remove`, `press`, `release`, `action` `etc.`) を用いて，イベントの発生元である部品 (オブジェクト) を特定する．`method()` では，その特定したオブジェクトのメソッドを指定することで，そのメソッドの返り値を挿入できる．例えば，`@action.getLabel()` は，アクションイベントの発生したボタンのラベル文字列に書き替えられる．

図 4に，実際に生成されたコマンドを実行している画面を示す．この例では，グラフ編集アプレット

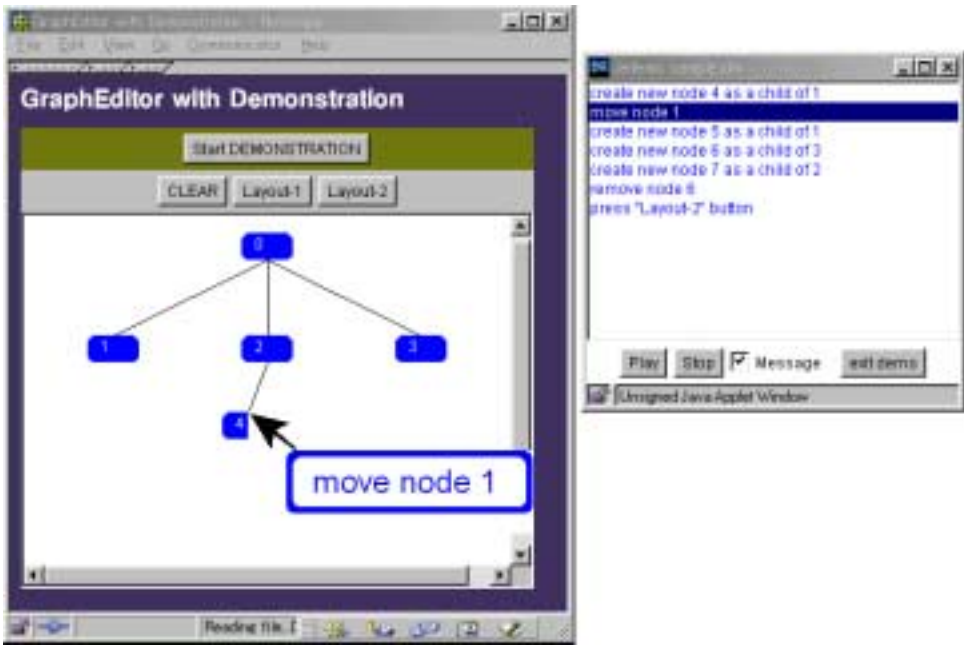

図 4　擬似マウスカーソルを用いたアプレットデモンストレーションの再生

のノード編集の方法を擬似マウスカーソルを動かしながらデモを実行している．コマンドのラベルを一覧表示したり，現在実行しているコマンドをハイライトして表示できる．

5　関連研究

操作を記録して，再利用しようとする試みは様々なアプリケーションやツールキットで実装されている．AppleScript [3]は，Macintosh 上の操作を再利用可能なスクリプトにする．ツールキット関連では X Window System からイベントを送る方法 [4]や，Tcl/Tk での実装 [5]がある．AIDE project [6]の AIDEWORKBENCH は，SmallTalk で作られたアプリケーションについてマクロ機能やアンドゥ機能を追加するものである．

Sun の JavaHelp [7]は，文章を主体としたヘルプのビューアである．全文検索機能などが充実しているがアニメーションヘルプの機能はない．

6　まとめ

一般の Java アプレット / アプリケーションについて，簡単にデモンストレーションヘルプを作成するシステム Jedemo について述べた．コマンドルールを用いて操作に意味付けすることで開発者の負担を軽減できる．また，バージョンの違いなどのシステムの変更にともなうヘルプの再構成も容易となる．Jedemo については `http://www.softlab.is.tsukuba.ac.jp/~miuramo/jedemo/` にて公開している．

参考文献

[1] Piyawadee Sukaviriya and James D. Foley. Coupling A UI Framework with Automatic Generation of Context-Sensitive Animated Help. In *Proceedings UIST '90*, pages 152–166, 1990.

[2] Krishna Bharat and Piyawadee Sukaviriya. Animating User Interfaces Using Animation Servers. In *Proceedings UIST '93*, pages 69–79, 1993.

[3] Apple Computer,Inc. Introduction to the Macintosh Family — Second Edition.

[4] Krishna Bharat, Piyawadee Sukaviriya, and Scott Hudson. Synthesized Interaction on the X Window System. Technical report, Graphics and Usability Center, Georgia Tech, USA, 1995.

[5] Charles Crowley. TkReplay: Record and Replay for Tk. In *USENIX Tcl/Tk Workshop Toronto*, pages 131–140, `http://www.cs.unm.edu/~crowley/papers/replay.tk95.html`, July 1996.

[6] Philippe P. Piernot and Marc P. Yvon. *The AIDE Project: An Application-Independent Demonstrational Environment*, chapter 18, pages 383–401. In "Watch What I Do : Programming by Demonstration," The MIT Press, 1993.

[7] Sun Microsystems Inc. JavaHelp Homepage, `http://java.sun.com/products/javahelp/`.