# 実世界 GUI による情報家電プログラミング

増井俊之

ソニーコンピュータサイエンス研究所

masui@acm.org

現在一般的な家電製品は、機器ごとに異なるリモコンが使用され、固定的な機能が割り当てられているのが普通であるが、将来の情報家電機器を扱う場合に、機器の数だけリモコンを用意し、機能の数だけボタンを用意するのは現実的でない。
実世界インタフェースに向いた入力装置「FieldMouse」を用いることにより、あらゆる場所において計算機上のグラフィカルユーザインタフェースと同じような操作で情報家電機器を操作することが可能である。本稿では、このような実世界 GUI の手法とビジュアルプログラミングの手法を組み合わせることにより、エンドユーザが情報家電機器を容易にプログラミングできる実世界指向プログラミング環境を構築することができることを示す。

# Real-World Programming by Real-World GUI

Toshiyuki MASUI

Sony Computer Science Laboratories, Inc.

masui@acm.org

Although more and more computing is performed away from desktop computers, most programs used in handheld computers, ubiquitous computers, and augmented-reality systems in the real world are still developed on desktop computers, and users of these systems cannot modify the behavior of the systems or make a new program for the systems without using desktop computers. Programs used in real-world environments should also be programmed in the real world, so we have developed a new programming paradigm, “Real-World Programming (RWP),” which enables users to make programs for handling real-world environments as well as data in computers. In this paper, we show how we can make simple real-world programs only by using a new device called the FieldMouse.

## 1 はじめに

近い将来、家庭内で使われる各種の家電製品は、ネットワークで結合された、いわゆる**情報家電**機器となると考えられる。現在一般的な家電製品は、機器ごとに異なるリモコンが使用され、固定的な機能が割り当てられているのが普通であるが、複雑な情報機器を扱う場合に、機器の数だけリモコンを用意し、機能の数だけボタンを用意するのは現実的でない。

我々は、**実世界インタフェース**に向いた入力装置 **FieldMouse** を提案している [2]。 FieldMouse を使うと、あらゆる場所において計算機上のグラフィカルユーザインタフェース (GUI) と同じような操作で情報家電機器を操作することができる [4]。本稿では、このような**実世界 GUI** の手法によってビジュアルプログラミングの手法を実世界 GUI に適用することによりエンドユーザが情報家電機器を容易にプログラミングできることを示す。

## 2 実世界 GUI

我々は、実世界インタフェースに向いた入力装置 FieldMouse を用いた「実世界 GUI」を提案している [2][4]。 FieldMouse はマウスや加速度センサのような相対移動検出装置とバーコードリーダのような ID 検出装置を一体化した装置であり、壁や紙の上に貼ったバーコードなどの ID を認識した後で FieldMouse を動かすことにより、従来の入力装置では難しかった各種の操作を簡単に指示することができる。 FieldMouse を用いることにより、不明瞭になりがちな情報家電機器のインタフェースを誰にも理解しやすくすることが可能である。

図 1はレーザスキャナつき PalmPilot[1]に傾きセンサを組みこんだ FieldMouse の例である。

図 1: レーザスキャナつき PalmPilot に傾きセンサを組みこんだ FieldMouse

[1]Symbol 社 SPT1500

### 実世界 GUI の例

図 2は紙の上に印刷された GUI 部品の例である。「Volume」と印刷されたバーコードを FieldMouse で認識した後左右に動かすことにより、相対移動量に応じて音量を制御することができる。また左下のバーコードを認識した後 FieldMouse を上方に動かすとビデオ 1 が選択され、下方に動かすとビデオ 2 が選択される。このように、バーコードを認識した後の FieldMouse の相対移動量や方向を活用することにより、普通の紙や壁を GUI 部品のように使うことができる。

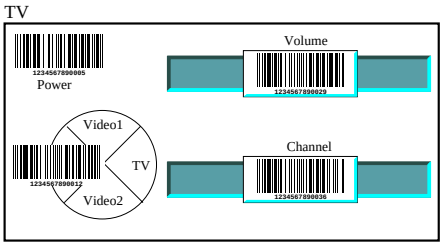


図 2: 紙上の GUI 部品の例

### リモコンの共通化

前述の例の場合、バーコードが印刷されていれば何でもリモコンとして使うことができるため、紙をコピーするなどによりどこでもリモコンを使うことができ、紛失を心配したりリモコンを探し回ったりする必要がなくなる。また、種類の異なるリモコンもひとつの FieldMouse で扱うことができるため、複数の機器を制御するために複数のリモコンを用意する必要がない。またリモコンの GUI 部品はどこに置いてもよいので、操作に最もふさわしい場所に実世界 GUI 部品を貼っておくことができる。たとえば、スピーカの音量を制御する部品をスピーカ本体に貼っておくことができるし、電話の横に貼っておけば電話中に周囲の音量を調整することができる。

### 直感的な具体的操作

たとえば音量を操作する場合、従来の回転型ボリュームに慣れた人は左右の回転操作を音量調節にマッピングすればよいし、スライドボリュームに慣れた人は移動量を音量調節にマッピングすればよい。

ビデオ A のデータをビデオ B にコピーしたい場合は、ビデオ A の前でデータをすくいとるような操作を行なってからビデオ B にデータを流し込むような操作を割り当てることができる。また、図 3のようなパネルの上で FieldMouse をデータの流れにそって動かすことにより直感的にデータの流れを指示することもできる。

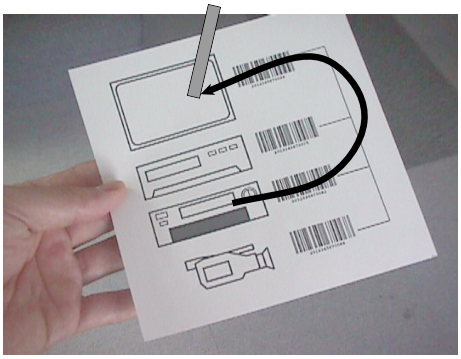

図 3: データの流れの指示

電気製品の操作は間接的でわかりにくいものが多いという問題が従来からしばしば指摘されている。例えば部屋の照明を操作する場合、どの照明がどのスイッチに対応しているのかわかりにくい場合が多いし、照明本体を直接操作することができない場合も多い。スピーカの音量を操作するためにはアンプで音量調節の操作をしなければならないが、スピーカを直接操作できてもよいはずである。

従来の家電製品ではこのように自明でないマッピングをユーザが把握しなければ使えないものが多いが、 FieldMouse による直接的操作が可能なインタフェースではこのような不自然なマッピングを最小限におさえることができる。

## 3 実世界指向プログラミング

アンプや計算機の制御パネルを使うかわりにスピーカの前で音量を制御したいと一般ユーザが思った場合など、スピーカで直接音量を制御するようなプログラムをユーザが簡単に作成できることが望ましい。

FieldMouse を使用する実世界 GUI のプログラムは計算機上のテキストエディタで作成することができるが、実世界の事物の処理のプログラミングは実世界の事物のみを使って行なうことができるようにした方がはるかに都合が良いと考えられる。

従来は、テキスト言語を使用して端末画面の GUI プログラムを作成するのが普通であったが、近年はグラフィカルな環境を用いて GUI プログラムを作成する手法が一般的になりつつある。図形を用いた、いわゆるビジュアルプログラミングが GUI のプログラミングに有用であるのと同じように、実世界 GUI のプログラミングを行なうためには実世界 GUI を用いることが自然であると考えられる。実世界の事物を最大限に活用して実世界指向インタフェースのプログラミングを行なう手法を**実世界指向プログラミング**と呼んでいるが [3]、 FieldMouse を用いることにより、ディスプレイやキーボードなどを使わずに実世界 GUI のプログラミングが可能になる。

### 実世界指向プログラミングの特長

実世界指向プログラミングには以下のような多くの利点がある。

#### プログラムと操作対象の整合性

従来のプログラミング言語は、計算機内のテキストとして表現されており、プログラムで扱う対象も計算機内のテキストや数であることがほとんどであった。このように、プログラムとその扱う対象の形式が近い場合は問題が少ないと考えられるが、形式が大きく異なる場合は整合性の問題が生じてしまう。例えばテキストベースのプログラミング言語で青い正方形を描く場合、

```
glColor3f(0.0,0.0,1.0);
glRectf(0,0,100,100);
```

などとプログラムしなければならないが、青い矩形と上記のテキストとのギャップは非常に大きい。一方グラフィカルエディタを使って正方形を描く操作をそのままプログラムとすることができれば理解がはるかに容易である。

ビジュアルな GUI のプログラミングシステムとして、いわゆる「インタフェースビルダ」が近年よく使われているが、この場合にはプログラムの表現と操作対象が近いために理解が容易になっている。ビジュアルな対象を扱うためにはビジュアルなプログラミングが適しているということになる。

同様に、実世界の対象を扱う実世界指向インタフェースのプログラミングを行なうには対象となる実世界の事物を使うことが最も適していると考えられる。

#### 具体性

実世界の事物をプログラミング要素として扱う場合、具体的な指定を行なうことが容易である。たとえば「ビデオからテレビに画像を転送する」という処理を指定する場合、テキストベースの計算機を使う場合は、あらかじめ各 AV 機器に名前をつけたうえで

```
% videocopy video1 TV1
```

などと指定しなければならないであろうが、

- ビデオの名前が “**video1**” であること
- テレビの名前が “**TV1**” であること
- ビデオ信号を転送するためのコマンドが “**videocopy**” であること

など多くのことを記憶して正しく指定する必要がある。実際にビデオからテレビにケーブルをつなぐ場合のようにこれらの機器を実際に使ってプログラミングを行なう場合にはこのような困難は発生しない。

#### 例示プログラミングとの整合性

実世界の事物を使って操作を指定する場合、具体的な処理は指定しやすいが、変数を含む処理や繰り返し / 条件分岐などを含む処理は指定しにくいという問題がある。しかし例示プログラミング (PBE, Programming by Example)[1] の

手法を用いれば、ある程度の抽象度をもつプログラムを自動的に生成することも可能である。たとえば何度もビデオからテレビに信号を送る操作を繰り返していた場合は、信号元としてビデオを指定した時点で送出先をシステムが推論することができるだろうし、毎日同じ操作をしていたような場合はその操作を自動実行させてしまうことも可能であろう。

### シンボルとして表現しにくい対象の指定

たとえば携帯端末において「駅のそばで電源を入れると時刻表を表示する」という機能をプログラムしたい場合、テキストベースのプログラムを使う場合は緯度 / 経度や電界強度などを数値化して比較するプログラムを書く必要があり指定が困難であるが、地図の上で領域を指定したり実際に駅に行って電界強度を計測しながらプログラムを書けば簡単に指定することができる。このように、他の表現手法が使いにくいものでも、実物またはその代理を使うことにより容易にプログラム中で用いることができる場合がある。

ビジュアルな GUI のプログラミング環境では、実際に画面上に矩形を描くことによりウィンドウを作成したり、その矩形へ矢印をひくことによりイベントの流れを表現したりすることが多い。たとえば、ビジュアルな GUI プログラミング環境である MAX では図 4のようなプログラムが使われる。ここでは、画面上のボタンが押されたときに計算が実行されて印刷されるというデータの流れが矩形と矢印で表現されている。 FieldMouse による実世界 GUI でも、同様の手法によりこのような実世界指向プログラミングを行なうことができる。

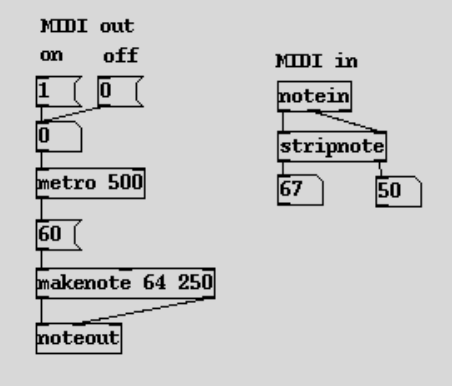


図 4: PD (MAX の後継システム) プログラム例

## 4 実世界指向プログラミングの実例

計算機を用いてプレゼンテーションを行なうプログラムを実世界指向プログラミングで作成する例を示す。

計算機を用いたプレゼンテーションでは、スライドの表示切り替え・ビデオの再生 / 停止・音量の調整などの各種の操作が必要になる。ビデオは計算機内の MPEG が使われることもあるし、ビデオデッキとテープが使われることもある。音量の調整は計算機上のコントロールパネルで行なわれることもあるし、アンプの音量つまみで制御されることもある。スクリーン上の画面の調整を行なうとき、 PC を操作すべきなのか、プロジェクタを操作すべきなのか、スクリーンを調整すべきなのか、判断に迷うことがある。画面を切り替えたりスピーカの音量を調整したりするという比較的単純な作業であるにもかかわらず、接続についてよく理解していないと要領よく操作することはできない。

実世界 GUI を使えば、ひとつの FieldMouse を使ってあらゆる機器を適切な場所で操作することが可能になる。たとえば、スクリーンを操作して次のスライドを表示したり、スピーカで直接音量を調整したり、写真をかざすことにより対応するビデオを再生したり停止したりすることができる。

テキストベースの言語を用いてこのようなプログラムを作成する場合は、例えば以下のようなプログラムが必要になる。

```
#define NEXTSLIDE 1234567
slideno = 1;
while(id = scan_id()){
  switch(id){
  case NEXTSLIDE:
    slideno++;
    show_slide(slideno);
    ...
  }
}
```

このようなプログラミング手法は、ウィンドウシステムの GUI プログラミングでよく用いられるが、エンドユーザにとって理解しやすいものではない。このようなプログラミングスタイルに慣れたプログラマにとってすら、**NEXTSLIDE** というシンボルと実際のスライドボタンの見栄えや操作とのマッピングを常に覚えておかなければならないので負担が大きい。

### 実世界指向プログラミングの実装

以下のような方針で実世界指向インタフェースのプログラミングを行なう。

- データフロー型のビジュアルプログラミングと同様の手法を採用

  MAX などと同じように、データや演算子のようなプログラミング要素 (モジュール) をリンクで結ぶことによりプログラムを表現し、計算実行はリンク上の文字列データの流れで表現する。

- バーコードの使用

  バーコードスキャナと傾きセンサを一体化した FieldMouse を使用し、バーコードを用いて実世界の事物の認識 / 操作を行なう。

- ID カードの使用

  演算子のように、実世界の事物で表現しにくいプログラミング要素は、名刺大の紙にバーコードを印刷した「ID カード」で表現する。

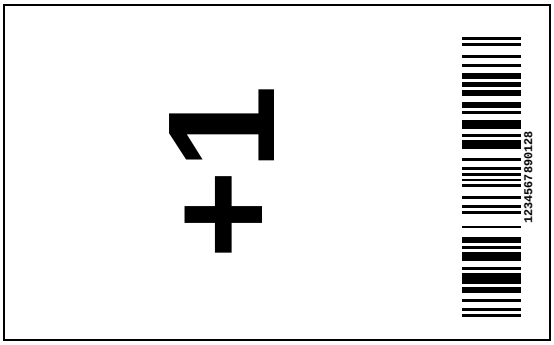


図 5: ID カードの例

- リンクの設定

  ソースとなるモジュールのバーコードからデータをすくい上げるジェスチャを行なってから、デスティネーションのバーコードにデータを流し込むジェスチャを行なうことにより、モジュール間にリンクを設定する。

モジュールがユーザのアクションを受け付ける場合は、ユーザがカードのバーコードをスキャンしたとき及びスキャン後 FieldMouse を回転した場合に、リンク先のモジュールに対して操作に対応した文字列を送付する。計算その他の処理を行なうモジュールの場合は、他モジュールから文字列を受け取った時点で、定義された処理を行なった後、リンク先のモジュールに計算結果の文字列を送付する。

### プログラミング要素の種類

スライドプレゼンテーションのための実世界指向プログラミングでは、以下のようなモジュールを使用する。

- 文字列モジュール

  文字列要素は、それに対してユーザがなんらかのアクションを起こしたときに文字列を他のモジュールに送出する。

- 計算モジュール

  計算モジュールは、四則演算や文字列演算などの各種の計算を行なう。たとえば文字列連結モジュールにふたつの文字列モジュールが接続されているとき、ユーザが片方の文字列にアクションを起こすと、それらが連結され、結果が他のモジュールに送出される。

- 連続値モジュール

  文字列モジュールは固定文字列を生成するが、連続値モジュールはユーザのアクションに応じて異なる値を連続的に出力する。

- 音声 / 画像モジュール

  音声 / 精神画 / 動画名の文字列を受け取り、対応するファイルを表示 / 再生する。

- 音量モジュール

  音量を受け取った数字に設定する。

### 簡単なプログラム例

前節で述べたような各種のプログラム要素を示す ID カードを使った実世界指向プログラミングの例を示す。

**文字列の表示**　文字列カードから文字列表示カードにリンクを設定することにより、 FieldMouse で文字列を選択したときにその文字列が表示されるようになる。

**画像の表示**　画像名を示す文字列カードから画像表示カードにリンクを設定しておくと、画像名カードを選択したときに画像名が表示カードに送られてその画像が表示される。

図 6のように、画像名を示す文字列カードにその画像を印刷しておけば、表示したい画像の印刷されたカードをスキャンするだけで画面にその画像が表示されるので理解しやすい。この手法は、動画や音楽の場合にも適用できる。

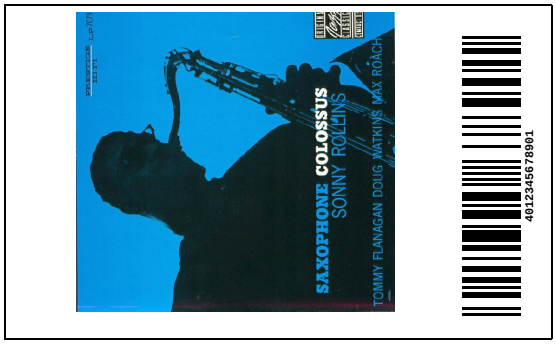


図 6: 画像 / 音楽カード

**音量制御**　数字を表現する文字列カードから音量調節カードにリンクを設定しておくと、選択した数字カードの値に音量が設定される。たとえば、文字列「0」を示す文字列カードに「Mute」と印刷しておけば、そのカードを FieldMouse で選択することにより音量をゼロにすることができる。

連続値カードから音量調節カードにリンクを貼ることにより、 FieldMouse の傾きにより音量を調節することができる。図 7のように、連続値カードに通常のボリュームのような絵を印刷しておけば、これを本物のボリュームと同じ感覚で使うことができる。

このようなカードをアンプ / スピーカ / 電話機など、音量調節が必要になりそうな場所に貼っておけば便利である。電話が鳴ったときにアンプの音量を下げるといったこともできる。

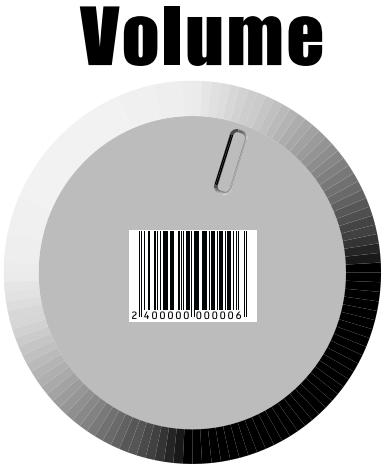


図 7: 音量調整カードの例

### プレゼンテーションプログラム

前述のような各種のカードを組み合わせてプログラミングすることにより、スライドやビデオを用いたプレゼンテーションをすべて実世界指向プログラミングすることができる。プレゼンテーションに必要なプログラムを図 8に示す。矩形が ID カードで表現されるモジュールを表現し、矩形間のリンクはジェスチャによって ID カード間に設定されるものである。

“+1”, “-1” などのカードを操作すると、“Σ” モジュールに加算結果が蓄積され、“Slide” という文字列と連結されて画像モジュールに送られることにより、“Slide1”, “Slide2”... などの画像ファイルが表示される。“Slide1” などのカードを操作すると直接そのスライドが表示される。

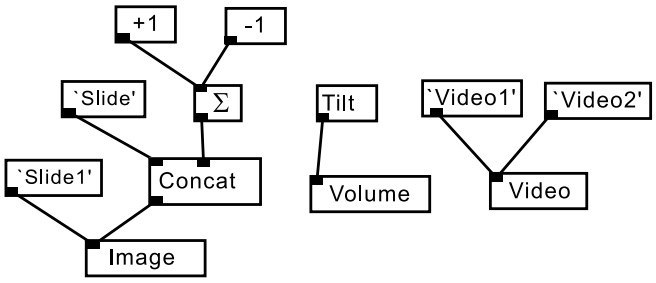


図 8: プレゼンテーションのためのプログラム

## 5 結論

FieldMouse を実世界指向インタフェースの入力装置として用いることにより、複雑になりがちな情報家電操作およびそのプログラミングをわかりやすくすることができることを示した。

今回はスライドを用いた計算機でのプレゼンテーションに限った応用を考えたためカードの種類はそれほど必要ないが、応用に応じて多くの種類のカードが必要になると考えられる。

実際には数字や文字列を示すカードが非常に沢山必要になってしまうので、これらについてはカード以外の文字列指示手法を使用した方が良いかもしれない。

GUI の歴史をふりかえると、時間をかけてボタン / スライダなどといったインタフェースの**イディオム**が共通知識化されてきたことと、 GUI を用いた GUI プログラム開発者環境が整ってきたことが普及の鍵となったようである。実世界 GUI に関してもイディオムの確立ととプログラミング環境の整備が重要であろう。

## 参考文献

[1] Allen Cypher, editor. *Watch What I Do – Programming by Demonstration*. The MIT Press, Cambridge, MA 02142, 1993.

[2] Itiro Siio, Toshiyuki Masui, and Kentaro Fukuchi. Real-world interaction using the FieldMouse. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'99)*, pages 113–119. ACM Press, November 1999.

[3] 増井 俊之. 実世界指向プログラミング. In 第 *40* 回冬のプログラミングシンポジウム予稿集, pages 19–25. 情報処理学会, January 1999.

[4] 増井 俊之, 椎尾 一郎, and 福地 健太郎. 紙 gui による情報家電制御. In 第 *59* 回情報処理学会全国大会 特別セッション *(1)* 講演論文集, pages 73–74, September 1999.