ビジュアル言語のすすめ

増井 俊之

事物を表現するには図が最適であることが多く、回路図 / 設計図 / 地図など、工学の世界はいろいろな「図」であふれていますが、計算機プログラムの表現には今のところ図よりもテキストの方がより多く使われているようです。

百聞は一見に若かないと古来伝えられているとおり、具体的事物を扱う計算機プログラムでは図的表現がテキスト表現に勝ることが多いと思われますし、抽象的な計算を行なう場合でも式や関係などを図で表現した方が都合が良い場合がよくあります。本物のプログラマはスクリーンエディタを使わないと信じられていた時代があったように、今でも本物のプログラマはビジュアル言語など使わないと思われているフシが無いでもありませんが、これは実用的なビジュアル言語がまだ少ないからであり、ビジュアル言語の未来は明るいと信じるに足る理由があります。

ビジュアル言語を選ぶ 7 つの理由

図的表現を使うビジュアル言語が優れている理由について具体的に見てみましょう。

見栄えが魅力的　プログラムに限らず「見た目」は非常に重要です。テキストベースで運用されていたときは評判を呼ばなかった World Wide Web が Mosaic により大成功した理由は、見栄えがよかったことに尽きるでしょう[1]。ビジュアル言語を使えば、プログラムを美しくわかりやすい形で表現することができる可能性があります。どんなに優れたテキストベースのプログラミング言語でも、エディタで編集しているところを 1 メートル先から見れば皆同じですが、見栄えが全然違えばそれだけで魅力的です。

色や形の素直な表現　テキストベースの言語では、色や形を含めてあらゆるものを文字列として表現しなければなりませんが、プログラム内でこれらをそのまま表現できるのであれば無理に文字列表現する必要はありません。

```
int color = 0x00ffff;
int pattern = 0xa5a5;
```

よりはまだ

```
Color color = ;
Pattern pattern = ;
```

[1]「絵をつけただけでなったよ 金持ちに」 ((C) 暦本純一 @ ソニー CSL)

の方がわかりやすいでしょう。

計算機の入出力画面のようにもともと 2 次元の対象を扱うときには、位置や大きさなどを数値で表現するよりも、プログラム内でも見たままの形で扱う方が便利です。

並列性　図的な記述は、1 次元のテキスト表現と異なり、解釈する順番が決まっていないのが普通です。同じ設計図面を使う場合でも、どの場所から作り始めるかには任意性があります。

並列事象をひとつの図で表現できるということは回路図やデータフローグラフのように信号の流れを表現するのに適していますし、解釈する順番に任意性があるため、「どういう手順で解釈するか」よりも「何を解釈するか」という本質を表現しやすいといえます。

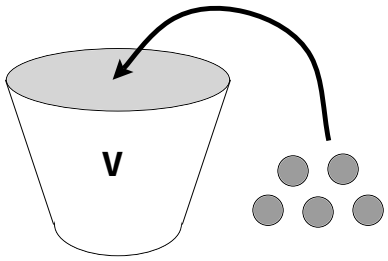
複雑な関係の記述　時間経過と事象の関係を表現するためには、時刻と事象の組を並べたりするよりも、時間の軸と事象の軸を 2 次元表現する方がはるかに理解しやすいのが普通です。世界で最も普及しているビジュアル言語のひとつである「楽譜」をはじめ、ノンリニア編集システムでもオーサリングシステムでも、時間を X 軸にした 2 次元表現が広く使われています。その他、状態遷移を表現したり、要素間の関連を表現したり、道順を表現したりするためには、図的表現が欠かせません。

一覧性　図的表現を拡大 / 縮小することにより、簡単に一部分を参照したり全体を見わたしたりすることができます。最近はパソコンでも連続的に高速に大きな図を拡大 / 縮小表示することが充分可能ですから、富豪的に巨大な図をプログラムとして使えばよいでしょう。

複数の視点　普通のカレンダは週と曜日で行列表現されています。時間は本来 1 次元ですからカレンダも 1 次元で表現してもよさそうなものですが、そのようにすると曜日の関係がわかりにくく非常に使いにくいものになってしまいます。カレンダでは、行 / 列表現することにより、時間経過の視点と曜日の視点がひとつの図で示されていることになります。この場合のように、図的表現を使うことにより、ひとつのデータを複数の視点から眺めることもできるようになります。

具体性　計算機のしくみやプログラミングを全く知らない人に対して

```
int v = 5;
```



の意味するところを説明するのはむずかしいでしょうが、「v の値段は 5 円である」とか上図のような表現であれば、もう少しわかりやすいかもしれません。プログラミングには抽象的な考えも必要ですが、具体的な図や操作により同じことが表現できる場合は、具体的な形 / 大きさ / 色 / 手順などで表現する方がわかりやすいのが普通です。多くのビジュアル言語では変数を「箱」で表現し、箱に入る矢印や箱から出る矢印によってデータの流れを表現します。こうすれば何か量をもつものが箱の中に有るという具体的実感がありますし、箱に名前をつける必要がないという利点もあります。

ビットマップ置換規則にもとづくビジュアル言語「Visulan」を使い、画面のビットマップの書き替え規則だけで「インベーダーゲーム」を作った例が本誌で以前紹介されています [3]。テキストベースのプログラム言語を使ってこのようなゲームを作ろうとすると、インベーダの形をテキストで表現したり、位置を変数で表現したり、実際のゲームの見栄えとプログラムとが相当違うものになってしまいますが、 Visulan では見栄えや動作例を具体的に示すだけで複雑なプログラムが作れるようになっています。具体的な書き替え規則の図を並べることによって子供にでもプログラミングを行なうことができるようにしたシステムもあります [1]。

表計算ソフトも非常に成功したビジュアル言語のひとつですが、表計算ソフトでは、変数のような抽象的な概念を使うかわりに具体的な位置をもつセルを計算に使うことにより、プログラム作成がより容易になっています。

ビジュアル言語の成功例

以上のようなビジュアル言語の特徴を活かすことにより、便利に使える言語やシステムが沢山作られています。少し例を見てみましょう。

インタフェースビルダ

最近のプログラミング環境は、作成するアプリケーションのグラフィカルユーザインタフェース (GUI) を作るための画面設計システムが一体となっているものが多いようです。画面設計システムはよく「インタフェースビルダ」と呼ばれますが、これはウィンドウシステムのツールキット部品を直接操作によりウィンドウ画面に配置しながらアプリケーションの見栄えを設計し、かつインタフェース部品を操作したときのアプリケーションの動作記述を支援するシステムのことで、 NeXT 社のワークステーションに登載された Interface Builder がはじまりです。 (非常に拡張された)Basic や C++ にインタフェースビルダ機能を付加した VisualBasic や Visual C++ などについては本特集で解説がありますし、最近の多くの Java の開発環境にはインタフェースビルダが付属しています。 Sun の JavaStudio のように、Java プログラム本体もビジュアルにプログラミングできるものもあります。

インタフェースビルダを使うと、 GUI 画面の配置や動作をマウスなどで直接編集できますし、ユーザの操作に対する反応に関するプログラムもある程度グラフィカルに作成することができるため、最近では GUI アプリケーションの作成に広く活用されています。インタフェースビルダは、「アプリケーションの見栄えをそのままプログラムできる」こと及び「インタフェース部品の具体的な実体をプログラミングに使用できる」という点でビジュアル言語として成功しているといえるでしょう。

MAX

本特集に解説がある MAX は、計算機音楽の演奏や研究の分野で広く使われているビジュアル言語であり、プログラミングについての詳しい知識が無い演奏家や作曲家でも計算機を音楽に活用できるようにした優れたシステムです。MAX では前述のビジュアル言語の特長の多くがうまく融合されています。非同期的に並列に発生する入出力信号の流れが具体的に図として表現されますし、インタフェースビルダの場合と同様に、音量などを制御するスライダなどの部品を見栄えそのまま画面上に配置することができます。変数や関数などは画面上の箱として具体的に表現されるので、プログラミングに通じていない音楽家でも容易に使うことができます。

インタフェースビルダや MAX は成功したビジュアル言語の例ですが、もちろん提案されているすべてのビジュアル言語や視覚化手法が成功しているわけではありません。テキスト言語に比べるとビジュアル言語のプログラムは作成や編集が面倒ですし、画面配置を行なうためのセンスや手間も必要ですから、特に前述のようなメリットを生かせない分野で無理に使う必要はないでしょう。しかし計算機上での作図 / 編集がテキスト入力 / 編集並に容易になれば状況は変わるかもしれません。 Pegasus[4] のような新しく使いやすい図形作図 / 編集システムがビジュアル言語をさらに身近にするかもしれせん。

テキスト言語のビジュアル言語性

テキスト処理の牙城のような UNIX プログラミング環境においても、ビジュアルな表現が実はかなり活用されています。

コメントやインデント　C では文間の空白が意味を持ちませんから、

```
for(i=0;i<10;i++){
        for(j=0;j<10;j++){
                func(i,j);
        }
}
```

と書くかわりに

```
for(i=0;i<10;
i++){for(j=0;j<10;j++){
func(i,j);}}
```

などと書いてもかまわないはずですが、最初のように意味のあるインデントをつけて書くのが普通になっています。この場合プログラマは、文の並びの順番のみでなく、行中の文の位置に意味をもたせることにより、ビジュアル言語の場合と同じように、複数の視点を視覚的に表現していることになります。目立つようにラベルや`case`を配置する手法も同様です。

パタン表現　UNIX ではパタンマッチを行なうために正規表現がよく使われますが、例えばパタン “`^.*abc.*$`” を指定して文字列 “`abc`” を探すような場合、パタンマッチが行末から行なわれるのか行頭から行なわれるのかは指定しているわけではありませんから、検索パタンを視覚的に指定していることになります。

このように、ビジュアルな表現を使いたいのだが、ツールが無いので渋々テキストベースのツールを使っているというケースは実際はかなり多いと思われます。

インタラクションとビジュアル言語

ユーザインタフェースのプログラミングにおいては、プログラムと実際の動きのギャップを小さくしたい / 見栄えを直接細かく指定したい / ユーザが自分でプログラミングできるようにしたい / 複雑な制御構造を楽にプログラミングしたい / といった要求が強いため、ビジュアル言語の研究が特に盛んに行なわれています。テキストベースのインタフェースが主流だった時代はテキストベースのプログラミングが主流だったように、 GUI が主流の時代だからこそビジュアル言語が主流となるべく特に望まれているのでしょう。将来の実世界インタフェースや Ubiquitous Computing の時代には「実世界プログラミング」が研究され、「実世界 Basic」とか「身振り C++」とかが売られるようになるかもしれません。

ビジュアル言語と日本人

IEEE の主催で毎年開催されているビジュアル言語に関するシンポジウム[2]では例年日本やヨーロッパからの発表が比較的多く、アメリカからの発表の割合が (他の会議に比べると) 少なくなっています。漢字という視覚的言語を使っているからかどうかはわかりませんが、日本人はアメリカ人よりも視覚表現に関する意識が高いのかもしれません。情報視覚化に関する様々な事例を集めた Tufte の “Envisioning Information” [2] では、最初のページの「伊勢神宮参拝案内図」における視覚化技法の解説に始まり、絵巻物 / 時刻表 / 地図 / 教科書の図など、日本における各種の視覚化事例が紹介されています[3]。世界で通用する日本発のビジュアル言語に期待したいものです。

[2] IEEE Symposium on Visual Languages.

[3] そういえばアメリカでは食品見本は無いし / 写真が載ってないレシピがほとんどだし / 教科書にグラフがほとんど載ってないそうだし / 地図や楽譜を読めない人も多そうだし / 視覚的表現に対する一般的意識が高くないのかもしれません。

参考文献

[1] Apple Computer, Inc. *Welcome to Cocoa – Internet Authoring For Kids*, February 1997. http://cocoa.apple.com/cocoa/.

[2] Edward R. Tufte. *Envisioning Information*. Graphics Press, 1990.

[3] 山本格也. ビジュアルプログラミングはどこへ行く? *bit*, Vol. 28, No. 7, pp. 12–17, July 1996.

[4] 五十嵐健夫. ?pegasus の紹介? *bit*, Vol. 29, No. ??, pp. ??–??, ?? 1997.

増井 俊之 (ソニーコンピュータサイエンス研究所)

おまけ: 判定！ビジュアル言語を使うべきか?