2003－SE－142 (1)
2003／5／30

プログラム動作可視化ツールにみるProgram Text Markup Languageの可能性

大木 幹雄†

あらまし
初等プログラミング教育ではプログラムがもつ基本的な概念やアルゴリズムをいかに理解させることが問題となる．本稿では，C プログラム中に簡単なスクリプトを注釈として記述するだけで，プログラムの動作を3次元空間内のオブジェクトの動きとして表現するプログラム動作可視化ツールPAVIの概要とプログラム概念の理解度に与えるPAVIの効果について述べる．プログラム中に記述するスクリプトは，プログラム動作表示に関するマークアップ言語とにしての機能をもつが，従来のJSP,ASP,PHPのようなHTML埋め込みプログラムの概念とは異なるイベントの監視と起動の考え方を基盤としている．そこで本稿では，このスクリプトの概念を一般化したPTML(Program Text Markup Language)の特徴と可能性についても同時に論じる．

A Proposal of Program Text Markup Language through Program Visualization Tool

Mikio OHKI†

Abstract
One of the most important issues in elementary programming education is the way to give students basic ideas and algorithms of a program. This paper describes an outline of the Program Action Visualization Tool that animates program behaviors as objects actions in a three dimensional space when the user writes simple scripts as annotations in the target C program. It also discusses the effects of the tool observed in the students' understanding levels of programming ideas. Although the scripts embedded in the target program play the role of a mark-up language for representing program behaviors, it is based on a unique concept of triggering and observing events, which is different from the concept of traditional embedded programs of HTML such as JSP, ASP, PHP. This paper also focuses on the features and potential abilities of the PTML (Program Text Markup Language) introduced to adjust the concept of the script.

1．まえがき

プログラミング能力には個人差があることは万人が認めるところであり，それがSPI(Synthetic Personality Inventory)試験によるプログラマの適性検査の根拠にもなっている．筆者が行った調査によると，プログラミング成績が優秀者な者とそうでない者との間では，SPI試験における回転図形パターン，回転図形状の同定等，空間図形のイメージ操作の問題で得点に有意な差があることが判明している．すなわち空間的なイメージ操作が巧みな者の方が，そうでない者に比較してプログラミング成績がよい．

本稿では，これらの調査結果をもとに，アルゴリズムの理解力が困難なプログラミング初心者の理解度を高めることを目的にして開発した「プログラム動作可視化ツールPAVI (Program Action Visualization Interpreter)」の概要と効果について述べる．同時に，独立実行するプログラムとコミュニケーションをとりながら可視化制御を行う注釈文（PAVIスクリプト）の考え方とPAVIスクリプトの意味を一般化したPTML(プログラムテキストマークアップ言語)の基本概念，およびその可能性について述べる．

†日本工業大学 情報工学科
Nippon Institute of Technology , Department of Computer and Information

2. プログラミング教育での可視化の必要性

プログラマとしての適正を計測する試験として考案されたSPI試験は，言語に関する問題と非言語に関する問題から構成される．前者には，関係する文の選択，文章の並び替え，文字列の分類，文章の比較等の問題があり，後者には，数値の分類，加算数の比較，回転させた図形パターンの同定，図形切断面の同定，回転させた図形形状の同定等の問題がある．

筆者は，プログラム動作(演算や制御フロー等)の理解度とSPI試験の間に相関が存在するかを検証するため，基本的なプログラミング言語仕様と概念（具体的には，代入演算，算術演算，論理演算，型変換，1次元配列，制御文，ループ・2重ループ，ポインタ等）を初めて学んだ大学1年次生17名を無作為抽出し，検証実験を実施した．

その結果，表 1 で示すとおりである．プログラム動作の理解度とSPI 試験の得点との間に，統計的に有意な相関関係があることが確認された（被験者集団はプログラミング経験がないことから，プログラミング経験による理解度向上効果は無視した）．なお，SPI試験問題は，表 1 にあるとおり代表的な３カテゴリの試験―「文の語順選択」，「算術計算の応用」，「回転図形状の同定」を用いた．

表１ プログラム動作の理解度とSPI試験の相関表（抜粋）

	文の語順選択	算術計算の応用	回転図形状の同定	SPI 合計点
代入	0.073	0.048	0.478*	0.123
論理演算	0.525**	0.338	0.255	0.429*
配列定義	0.222	0.098	0.410*	0.247
ループ制御	0.338	0.350	0.390*	0.320
配列アクセス	0.369	0.423*	0.250	0.426*
二重ループ	0.306	0.390*	0.299	0.463*
理解度計	0.326	0.297	0.389*	0.397*

(注 1)　*：ピアソン相関係数 5%の有意水準
(注 2) **:　ピアソン相関係数 1%の有意水準

表 1 中の「回転図形の同定」とは，図 1で示すとおり，左端の図形を 90 度回転させてできる図形を1～4 の図の中から１つ選択させる問題で，頭の中で図形をイメージとして回転させることから，この問題は「空間移動操作力」を判断する問題と考えることができる．表 1 は，空間移動操作力がプログラム動作の理解度と最も関係が深いことを示している．

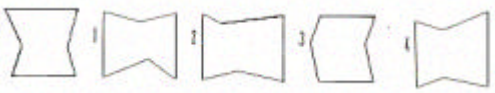

図 1　回転図形の同定問題例

ではプログラミング初心者の空間移動操作力を高めさせるには，プログラミング初心者にどのようなツールを提供すればいいのであろうか．表 1 をさらに見ると，空間移動操作力は，特にプログラミング動作の代入文，配列定義，ループ制御と有意な相関をもっている．これは逆に代入文，配列定義，ループ制御に絞って可視化して，プログラミング初心者の空間移動操作力を高めることができれば，プログラム動作の理解度を向上することが期待できることを示している．すなわち，図 2 で示すような関係によって，プログラム動作の理解度を高めることが期待できる．

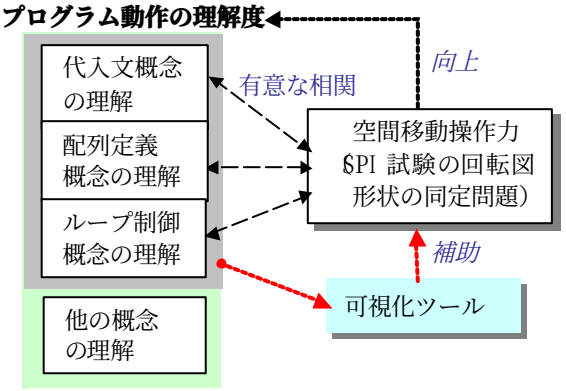

図 2　プログラム動作可視化の期待効果

3．プログラム動作可視化ツール PAVI の概要

3．1　可視化対象の演算

図 2 で示した期待効果が実際に生じるかを検証するため，プログラム中の次の演算に範囲を絞り，その動作の可視化した支援ツールがPAVIである[1]．C プログラムテキスト（本稿では，「ソースコード」に代わってこの用語を用いる）を読込み，それをインタプリトとして，プログラム中に記述されたデータに関する定義や演算を3 次元オブジェクトとその動きとしてアニメーション化する．

【PAVI で可視化対象とした演算】

(1)　代入演算
(2)　ポインタ演算

上記の 2つに演算を限定した理由は次の根拠による．

① プログラムの記述順序に最も影響を与えるものは，代入演算であり，代入の副作用が手順の理解を困難にしている．したがって，代入演算が評価される順序の把握をオブジェクトの動作として可視化することで，アルゴリズムのもつ手順の理解が容易になると期待できる．

② C 言語に特有のポインタ概念が初等プログラミング教育の 1 つの障害になっている．ポインタが指す内容を可視化することで，ポインタのもつ意味の理解が容易になるものと期待できる．

3．2　演算の表現と可視化範囲の指定方法

(1)　演算の 3 次元空間での表現方法

PAVI では，変数や配列，ポインタを 3 次元空間のオブジェクトとして表現し，図 3 で示すとおり，代入演算をオブジェクトの移動によって表現した．ポインタのもつオブジェクトへのポインティング操作は，ポインタから発する矢印で表現し，ポインタに対する演算は矢印を移動させることで

表現した．また変数や配列の要素，ポインタの値は，3 次元オブジェクトの高さによって表現している．

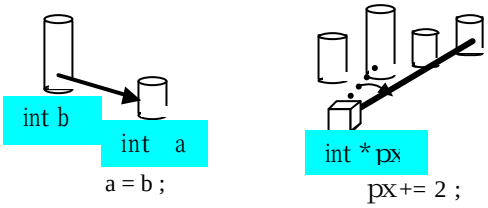

図 3　代入演算とポインタ演算の表現

(2)可視化する範囲の指定と動作表示の指定方法

プログラムの手順を理解するためには，プログラム全体の動作を可視化する必要はなく，注目したプログラムの範囲のみ可視化すればよいことが多い．そこで PAVI では，可視化する範囲を指定できるようにしている．また次のいずれかの方式でオブジェクトの代入演算を示す動作を選択できる．

① 順次動作による可視化方式

代入・ポインタ演算が出現する都度，それらをオブジェクトの動作として表現する方式である．

② 並行動作による可視化方式

指定された範囲にある代入・ポインタ演算を並列的に複数の動作として表現する方式である．表現する対象を増やすと注視力が弱まり，可視化の効果は薄れる．

3．3　PAVI スクリプトおよび類似ツール

(1) 可視化スクリプトの記述方法

PAVI では，可視化の対象となる変数やポインタを 3 次元オブジェクトとして表現する形式，可視化範囲，可視化動作の選択を，C プログラムテキスト中に特殊な注釈文/注釈行を記述することで指定する．注釈文/注釈行を表す記号に引き続く指定内容を以後「PAVI スクリプト」と呼ぶ．PAVI スクリプトは独自の構文規則をもち C プログラムの可視化に関するマークアップ言語として機能する．図 4 に具体的な PAVI スクリプトの記述例を示す．

```
//$ viewpoint=(0, -50, -50) farZ=500 nearZ=100
int data[10] =    { 24,10,8, 4,5,15,3,6,12,9}
 /*$ coord=(-20, 0,  -30) factor=3 color = 0xaa0000
        shape= cylinder width=2    $*/    ;
```

※C プログラムの配列定義 data[10] に対する属性指定
(単なる文字列 data[10] に対する属性指定ではない)

```
void quickSort(int array[], int l, int r);
main() {
    quickSort(data, 0, 9);
}
```

```
void quickSort(int data[], int l, int r) {
  int i, j, v;
  int swap /*$ coord=(20,0,-1) factor=3 color= 0xffffff
              shape= cube   width= 3   $*/   ;
  if (r> l)  {
      v = data[r];
      i = l - 1;
      j = r;
      do  {
          while(data[++i] < v);
          while(data[--j] > v);
          if (i < j)  {
              //$ traceBegin assign = line step 10
              swap = data[i];
              //$ parBegin
              data[i] = data[j];
              data[j] = swap;
              //$ parEnd   traceEnd
          }
      } while(i < j);
      //$ traceBegin assign = line step 10
      swap = data[i];
      //$ parBegin
      data[i] = data[r];
      data[r] = swap;
      //$ parEnd   traceEnd
      quickSort(data, l, i-1);
      quickSort(data, i+1, r);
  }
}
```

※ C プログラムの変数 swap の属性指定

3 次元表示空間とテキスト空間での移動表示

図 4　PAVI スクリプトの具体的な記述例

図 4 で示したテキストを PAVI ツールに入力し，3 次元空間内で可視化した事例が図 5～図 7 である．図 5 は，「ソースコードビューア」を示しており，PAVI がプログラムテキストをインタプリトし，3 次元空間でオブジェクトの動作で可視化したプログラムステップの現在行を反転表示している．図 6，図 7 は，指定した変数，配列，ポインタ等を 3 次元オブジェクトのアニメーション動作として表現する「アニメーションビューア」を示している．変数，配列，ポインタが 3 次元オブジェクトで表示されると共に，その識別名も名札プレートを用いて表示している．

(2) 類似ツールとの比較

PAVI の長所は，C コンパイラと互換性を保ちながら，PAVI スクリプトにしたがってプログラム動作が可視化できる点にある．図 4 で示したクイックソートプログラムの動きを可視化する特別なプログラムを組む必要はまったくない．アルゴリズムをアニメーション化するツールは，TANGO[5]

をはじめとして ANIM[6],CAT[7]等，数多く存在するが，これらは可視化用のライブラリ呼出のコードをプログラム中に記述しなければならない点で利便性を欠く．

(3) PAVI スクリプトの意味

図 4で示した PAVI スクリプトの具体的な意味は次のとおりである．/*$ $*/で囲まれた部分，または//$で始まる行は PAVI スクリプト記述部である．たとえば，変数，配列定義の後に記述された「coord=(-20, 0, -30) factor=3 color = 0xaa0000 shape= cylinder width=2 」で，3 次元空間にオブジェクトとして配置するときのx，y，z座標値，拡大率，色，形状，幅を指定する．traceBegin，traceEnd で囲んだ範囲に記述された代入・ポインタ演算は，インタプリタが評価を行う都度，図3で示したオブジェクトの動きとしてアニメーション化される．parBegin,parEnd で囲まれた範囲に記述された代入・ポインタ演算は，並列的な動きとしてアニメーション化される．traceBegin の後に続く「assign = line step 10」は，代入演算をきざみ 10 の直線的な動作で表現することを示している．

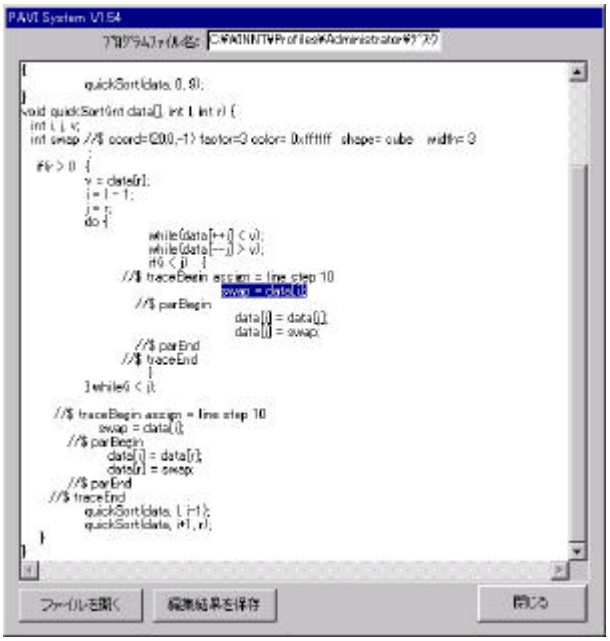

図 5 ソースコードビューア 反転表示した行位置が，3 次元空間表示中のプログラムステップを示している）

インタプリタは，traceBegin，traceEnd で囲まれた範囲に評価のコントロールがあるとき，マウスの右ボタンをクリックすると，次のプログラム行に評価を移す．このほか，右ボタンクリックが煩雑になる場合を考えて，自動実行モードが用意されている．図 6，図7で示したアニメーションビューア内に表示された変数や配列，ポインタを表現したオブジェクトは，値が変更されると直ちにその高さ（あるいは半径）に反映され再表示される．

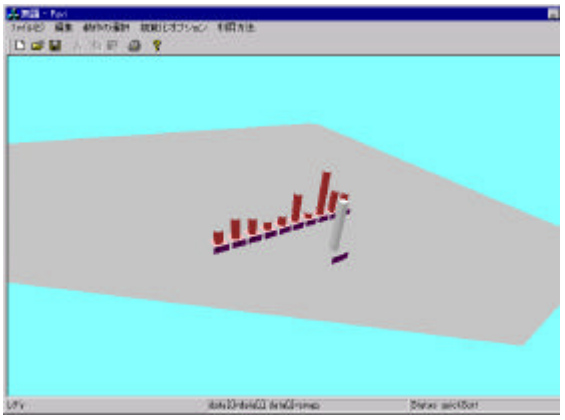

図 6 アニメーションビューア 図 4 プログラムの可視化例）

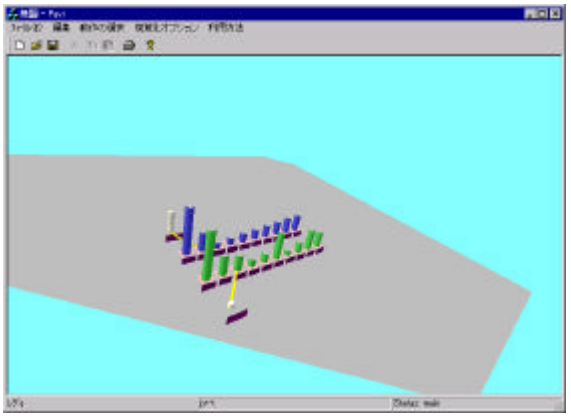

図 7 ポインタ演算を含むプログラムを可視化したアニメーションビューア（ポインタは黄線で示している）

(4) PAVI スクリプト言語仕様

PAVI スクリプトがもつ言語仕様は次のとおりである．

【オブジェクト可視化属性の指定】

①viewPoint: 3次元空間を見る視点位置．
②coordinate: データに対応するオブジェクトの 3 次元空間内での配置位置指定．
③color :オブジェクトの色指定．
④shape :オブジェクト形状（円柱，直方体，球）指定．
⑤height :オブジェクトの高さ上限値指定
⑥width :オブジェクトの幅上限値指定
⑦depth :オブジェクトの奥行き上限値指定
⑧factor :オブジェクトの拡大倍率指定

【可視化の範囲と動作指定】

① traceBegin,traceEnd: この指定の範囲内で行われる代入・ポインタ演算を逐次，オブジェクトの動

きとして可視化する.

② parBegin,parEnd: このしての範囲内で行われる代入・ポインタ演算を並列的にオブジェクトの動きとして可視化する.

③ trace wtyle: オブジェクトを始点から終点位置まで移動させる形態を指定する. 直線上, 円周上, 四角上の移動がある.

④ waiting event: 右クリックの都度, 次のプログラムステップの動作を実行するが, 右クリック以外のイベントを使用するときに指定する.

【その他の指定】

このほか, C言語と同様に, PAVI スクリプトの記述を容易にするため, 以下の指定を可能にしている.

①include 文: 取り込むべきファイルの指定

②define 文: 定義文

3. 4 表示画面と対話的な操作

PAVI は対話型可視化プログラムである. ステップ毎に評価される C プログラムは, 任意の時点で, 対話的に値や変数名,あるいは型名を確認することできる. 図 6, 図 7 の画面で, 直接, 3次元オブジェクトをマウスでポイントしながら, コントロールキーを押下すると, オブジェクトに対応する変数, 配列要素の型名や値がダイアログボックスで表示される. PAVI スクリプト記述によって初期配置されたオブジェクトの位置を, 可視化の途中で変更する必要が生じたとき, 3次元オブジェクトをマウスでポイントしながらシフトキーを押下するとダイアログボックスが表示され, 変更すべきオブジェクトの座標値を指定できる.

3. 5 PAVI の効果

前述の PAVI が実際にプログラミング初心者の空間移動操作力を高め, 引いてはプログラム動作やアルゴリズムがもつ意味の理解度を高めることができるか検証実験を行った結果は以下のとおりである. 実験は被験者17名を 2 グループに分け, PAVI を用いたときと用いないときの理解度を比較する評価テストを実施した 表 1 で示したプログラム基本的な動作の理解度に関して, 2グループ間には有意な差は存在しないことは確認済). また, 理解度の評価は, クイックソートのアルゴリズムを事前に解説して, そのプログラムソースコード中に出現する配列のインデックス操作, ブレーク文, 代入文・データ交換に関する空欄箇所を穴埋めする問題を解かせることで行った.

(1) プログラミング動作およびアルゴリズム理解度

2 グループ間の平均成績を t 検定すると, アルゴリズムの理解度に関して有意な差が得られた. PAVI を用いれば配列要素が並び替えられてゆく過程をアニメーションで確認できることから, 成績の向上は当然と言える.

(2) 動機付け効果

この他の定性的な効果として, PAVI を利用しないグループでは, 制限時間内で問題を解く努力を放棄する学生が 25%程度存在したが, PAVI を利用したグループでは存在しなかった.

以上の結果は, アルゴリズムのアニメーション化が, アルゴリズムのもつ動作の理解と, 問題を解く動機付けに有効に作用することを示している.

表 2 PAVI を用いたアルゴリズム理解度の差異

ｱﾙｺﾞﾘｽﾞﾑ問題	PAVI 利用 標準偏差	PAVI 未利用 標準偏差	t値
配列操作	5.63	6.43	3.35**
ブレーク	7.44	8.43	2.58*
代入文・データ交換	7.29	4.86	2.54*
合計	30.56	33.00	2.90*

(注) *：t 検定 5%の有意水準, **: t 検定 1%の有意水準

一方, PAVI で導入したスクリプトの概念は, 従来の HTML とは異なる概念にもとづいており, 3 次元ビジュアルプログラミング環境で用いる新しいスクリプト言語として発展させうる可能性を秘めている. 以下では PAVI スクリプトが果たす役割と特徴の観点から, この可能性について述べる.

4. マークアップ言語としての PAVI スクリプト

4. 1 PTML の定義

PAVI スクリプトは, プログラム動作の可視方法を記述したマークアップ言語として捉えることができる.

```
<%@ Language="VBScript" %>
<%
  Dim   Name
  Sub GetData
      :
  End Sub
%>
<HTML><BODY>
<TABL BORDER="1" >
<% If Request.QueryString("Type")="Student" Then %>
<TH >氏名</TH>
<TR><TD><%= Name %></TD>
</TR>              :
```

図 8 ASP によるサーバサイド・プログラム記述例

HTML 埋め込み言語である JSP(Java Server Pages), PHP(Hypertext Preprocessor),ASP(Active Server Pages)

では，クライアントサイドに表示するHTML テキスト中にタグの一種として，HTML テキストを生成するプログラムを記述する．たとえば，図 8 で示すような ASP を用いたプログラムを記述すると，サーバーサイドで<% %>で囲まれたスクリプトプログラムを評価して，HTML テキストを生成し，クライアントサイドに返信する．このとき，返信されるHTMLテキストとスクリプトとの実行空間は完全に独立しており，HTML テキストとスクリプト間で何らかの作用（＝コミュニケーション）をもつことはない．

一方，PAVI も同様にスクリプト部分を評価し，図 9 で示すとおり，スクリプトに記述された属性や制御指定にしたがって，プログラムテキストが意味する定義や動作を評価して，3 次元表示空間（アニメーションビューア）やテキスト空間（ソースコードビューア）に表示する．しかし表示対象となるテキストは，それ自体が動作するプログラムであり，プログラムの実行状態に依存して，スクリプト記述との間で密接な関連をもつ．

図 9 は，プログラムの変数に対して指定した PABI スクリプト表示属性がプログラムの実行状態に依存して，いかに 3 次元空間やテキスト空間と関連をもつかを示している．PAVI スクリプトによって変数や配列をスタック領域に確保する「領域確保イベント」や「代入演算イベント」が発生すると，PAVI はイベントハンドラが起動し，3 次元表示空間やテキスト表示空間に対して，オブジェクトの表示や移動，反転表示指令を送信する．このような「表示対象となるテキスト（＝プログラム）とコミュニケーションをもつ機能を内在したマークアップ言語」を **PTML（Program Text Markup Language）**と呼ぶことにする．

PTML の大きな特徴は，「分離独立して評価・実行される表示対象プログラムとスクリプトとが監視イベントによって密接に関連している」点にある．PTML スクリプトは表示対象プログラムの仮想エンジンで発生するイベントの中から監視すべきイベントを指定し，イベントが発生すると表示空間に送信する表示指令とパラメータ形式を決定する．

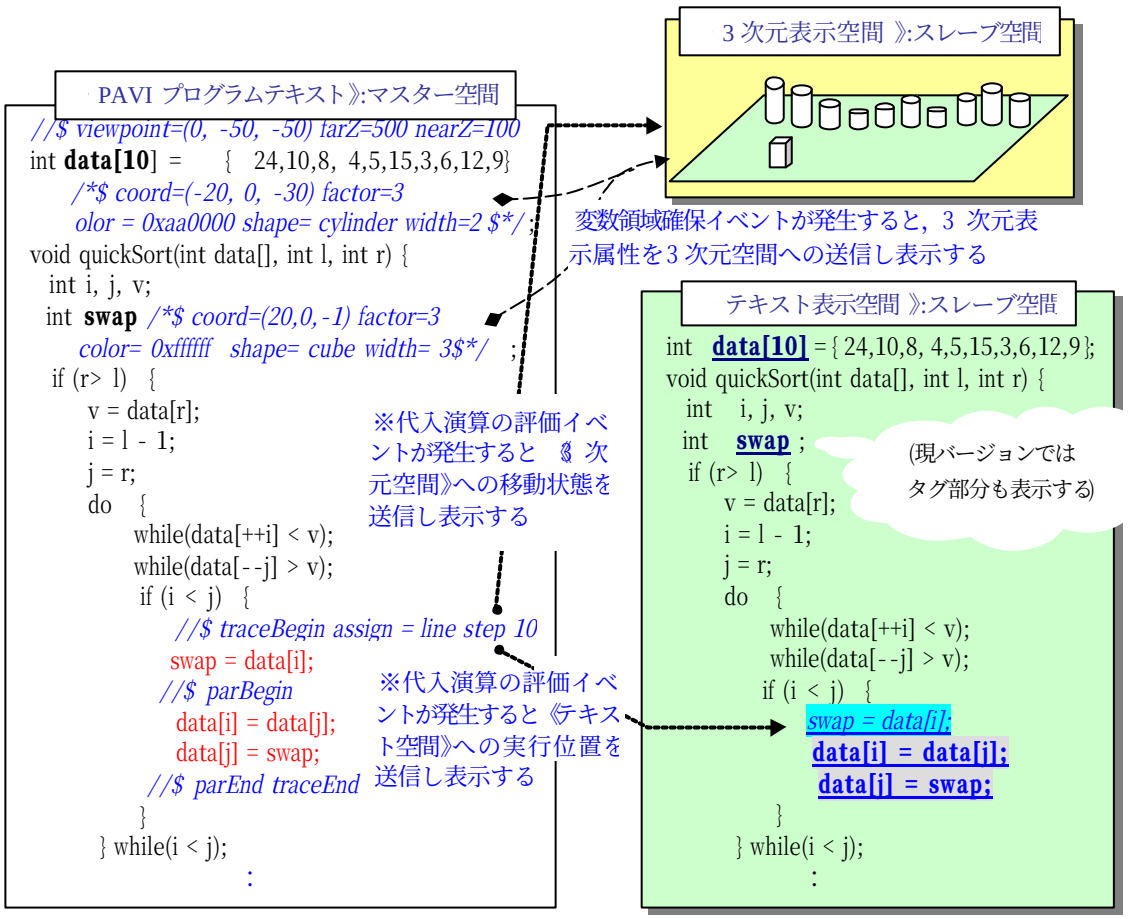

図 9 PTML としてのPAVI スクリプトの機能と役割

4. 2 処理形態から見たWebエンジンとの相違

JSP や ASP,PHP の HTML 埋め込み型サーバサイド・プログラム方式では，スクリプトで記述された手続きを HTML とは異なる実行空間で評価する（クライアントサイドに送信されたとしても同様）．クライアントサイドの Web エンジンは生成された HTML テキストをインタプリとして（スクリプトがあればそれも評価して），ホットスポットへのポインティングの監視やフィールド入力，サブミット指示等のイベントを監視して対応する処理を起動する．

一方，PTML 方式では動作するプログラムの実行状態を監視し，監視対象となるイベントが発生すると所定の表示空間へ表示指令を送る． PTML 方式では，スクリプトが埋め込まれ対象はプログラム言語のインタプリタが異なっていても，監視するイベントのインタフェースが一致しさえすれば適用できる長所をもつ. PTML 方式と従来の Web エンジンとの比較を図 9 で示す．図 9 では表示対象とする HTML テキストと Program テキストにおいて，スクリプトの関係が逆になっていることに注意する必要がある．

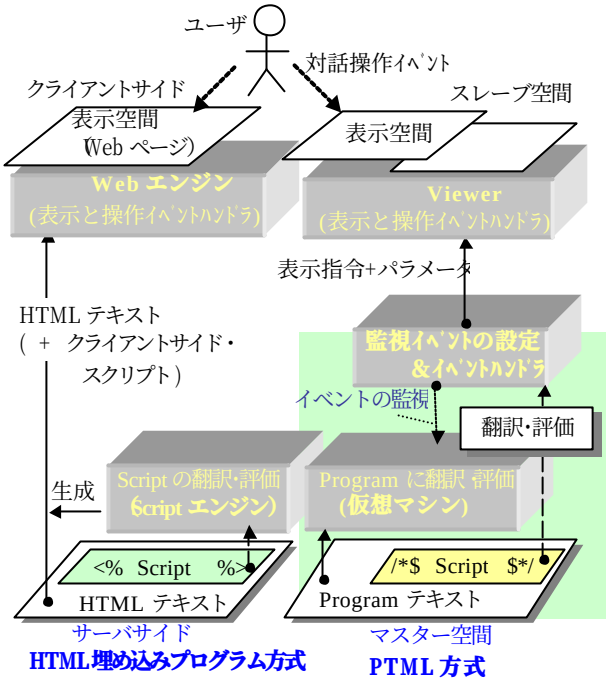

図 9　HTML 埋め込みプログラム方式とPTML 方式の比較

4. 3 PTML の評価と制御方法

PAVIでは,図 9 で示したアーキテクチャの概念に沿って，C プログラムインタプリタの構文解析モジュールと仮想エンジンモジュールに，次の機能をアタッチする方式で機能を実現している．具体的には次のとおりである．

(1) 構文解析モジュールへの機能追加

① スクリプト部の構文解析とバイトコード生成
スクリプト部を構文解析して PTML の処理特有のバイトコードを生成する．

② 各種の監視イベントを監視補助データの埋め込み
変数領域の確保イベントや代入演算イベント，ポインタ演算イベント等を監視するに必要な補助データをバイトコードへ埋め込む．

(2) 仮想マシンモジュールへの機能追加

① OpenGL を用いて，監視イベントが発生したとき，3 次元空間でオブジェクトを表示する．

② 監視イベントが発生したとき，テキスト空間で実行位置を表示する．

③ 3 次元表示空間でのキー&マウス操作イベントを監視し，ポインティングされたオブジェクトの情報を表示する．

これらの可視化に関する機能は,仮想マシンモジュールに可視化モジュール（Viewer）を追加することで行っている．

4. 3 PTML タグ体系の拡張と効果

現在の多くのビジュアルプログラミング支援環境は，GUI の定義に関しては優れた環境を提供している．しかしプログラミングに関しては，単にソーステキストの注釈や予約語の表示色を変化させる程度にしか過ぎない． プログラミング作業支援機能としては，貧弱な機能と言わざるを得ない．プログラムテキストを対象に PTML のタグ拡張すると，プログラムの実行状態と作用して，その状態を可視化する機能が容易に追加できる．その結果，次のようなプログラミング支援が可能になる．

(1) プログラム評価履歴の表示

PAVI の現バージョンではプログラムの評価状態を反映して，関連するプログラムテキストのみを選択して反転表示する．そこで「可視化の範囲と動作指定を指定するタグ」に，可視化の対象変数を含むステップの反転表示履歴を記録するパラメータを追加することで，プログラム評価履歴もテキスト表示空間で反転表示することが可能になる．その結果，どのステップが評価されたか，可視化データに依存した制御フローの追跡が容易に行えることになる．追跡の粒度を上げると，関数やブロック単位で追跡したものも表示できる（PAVI バージョン 2.0 で対応予定）．

(2) 視点依存のプログラム関連箇所の表示

PAVI バージョン 2.0（詳細は別稿で紹介する）では，可視化対象とする変数定義の有効範囲(=名前空間)に名称を与え，一定の検索条件を与えることで動的に選択できる．

そのため有効範囲を指定する働きをもつ関数名やブロックに対して複数の検索キーワードをタグとして付すことが可能になる．この機能を用いると，特定のキーワードに関連するプログラムテキストのみを表示することが可能になる．視点名に対応させたキーワードを上手に選択すれば，任意の視点で，相互に関連をもつ関数やブロックのみを表示できることになる．

(3) プログラム構成，設計書の参照

HTML の参照リンクの役割を分類すると次の5 つに大別できる．

①具体概念の提示
②抽象概念の提示（inherit）
③詳細概念の提示
④集約概念の提示(include)
⑤類似・反対概念の提示

たとえば，④集約概念参照リンクは，プログラムテキスト中に記述される従来の include 文や import 文に代表される構成上の要素を結びつける役割をもったリンクであり，①，⑤の参照リンクは名前のとおり，対象としたプログラムに関連した類似プログラム，具体事例を表示するリンクである．上記の 5 つのリンクの中で，特に②の抽象概念リンクは重要で，要約や概要を表示するリンクに相当する．

この役割をもつ参照リンクを，PTML のリンク属性に明示的に持ち込むことで，プログラムの理解性を高めることができる．具体的には，記述されたプログラム部分に該当するアルゴリズム仕様書やプログラム設計書は「抽象概念参照リンク」や「詳細化参照リンク」によって確認することができる(当然，逆の参照リンクが考えられるし，参照リンクを保守する DB の必要性が考えられるが，本稿では割愛する)．

5. PTML の将来性と課題

PTML は独立して動作するプログラム動作の可視化を簡易に行うために導入された考え方であるが，前述で述べた効果のほかに以下のような将来性を秘めている．

(1) プログラム動作の可視化の汎用システムへの発展

可視化するデータの構成を限定し，汎用的な可視化機能を提供するツールとして有名なものに AVS/EXPRESS[9]がある．メッシュ構造やジオメトリ構造をもったデータを読み込み，コンポーネントの組み合わせて，可視化する汎用パッケージである．同様に，PTML とインタフェースをもちうる仮想マシンに対して，PTML スクリプト中に可視化マクロ関数を組み合わせて記述できるようにできれば，複雑な動作をもつプログラムでも可視化の対象にすることができよう．

(2) スタイルシートによる複雑なアルゴリズムの可視化

PAVI の現バージョン 1.5 では，ソーティングアルゴリズムのようにアルゴリズムのもつ意味がデータ配列の値に反映される可視化プログラムに対してのみ有効な可視化機能を提供する．しかしネットワーク最短経路検索のアルゴリズムの可視化や B-TREE 索引構造の可視化等のようにデータが他のデータ構造へのポインタであるような複雑な関連をもつアルゴリズムの可視化に対しては，目下のところ無力である．その一つの解決策として，アルゴリズム毎に定義できるスタイルシートの導入や PTML スクリプト関数（および関数の組合せ）とアルゴリズム仕様との間で，明確な写像関係を定義可能にすることで，(1)で述べた汎用プログラム動作可視化ツールへの道が開けよう．

これらの将来性を加味して，PAVI のバージョンアップを今後，図って行きたい．

6. あとがき

本研究は，初等プログラミング言語教育でアルゴリズム理解の苦手な学生に如何に対処するかの問題から発している．その具体的な解決策を検討するために，情報工学科の学生に対して各種の評価実験を行った．協力していただいた学生諸君に感謝の意を表したい．

参考文献

[1] 大木幹雄,“プログラミング初等教育におけるプログラム動作可視化ツールの効果,” 情報処理学会ソフトウェア工学研究会 Winter Workshop In Kobe,pp.85-86(2003)

[2]平川正人, “マルチメディアソフトウェア工学, ” 共立出版 (1999)

[3] Brown, M., *Algorithm Animation*, MIT Press, Cambridge, 1988.

[4] J. Bazik, R. Tamassia, S. P. Reoss, and A. van Dam, “Software Visualization in Teaching at Brown University,” *Software Visualization*, J. Stasko, J. Domingue, M. H. Brown, and B. A. Price eds., MIT Press, 1998, pp. 383-398.

[5] J. Stasko, “TANGO: A Framework and System for Algorithm Animation”, *Computer*, Vol. 23, No. 9, Sep. 1990, pp. 213-236.

[6] J. Bentley and B. Kernighan, “A System for Algorithm Animation: Tutorial and User Manual,” *Computing Systems*, Vol. 4, No. 1 ,1991, pp. 5-30.

[7] M. Brown, “Zeus: A system for algorithm animation and multi-view editing,” *Proc. of IEEE Workshop on Visual Languages*, 1991, pp. 4-9.

[8] M. Brown and M. Najork, “Algorithm Animation using 3D Interactive Graphics,” *Proc. of IEEE Int’l. Sym. on Visual Languages*, 1996, pp. 266-275.

[9] http://www.avs.com/ Advanced Visual Systems