3 次元ビジュアルプログラミングシステム 3DPP の構築に向けて

Toward the Realization of 3-Dimentional Visual Programming System 3DPP

田中二郎 †

Jiro Tanaka

筑波大学 電子・情報工学系

Institute of Information Sciences and Electronics, University of Tsukuba

Abstract: We have been working for the design and implementation of the visual programming systems. Our initial system is called "PP," which stands for Pictorial Programming. After implemented PP, we continued to develop the series of PP-family, such as newPP, viewPP and PP-pad. In this paper, we propose 3DPP, which is the 3D extension of PP. Program elements are expressed as 3D objects in 3DPP. User can compose programs by combining these 3D objects. We have implemented "Claymore" which is the prototype of 3D modeling tools for our 3D visual programming. We have designed the program elements for 3DPP using "Claymore."

1 はじめに

近年のコンピュータの低価格化、パーソナル化により、コンピュータ上でのグラフィックスやアニメーションの表示が現実的となっている。

ビジュアルプログラミング (VP) という言葉は様々な文脈の中で使用され、その意味するところも様々である。VP を広義にとらえれば、プログラミングが何らかの意味で視覚的要素を含むなら VP であり、いわゆる Visual Basic、Visual C++ などもこれに含まれる。

一方、VP について、この言葉をもっと限定的に使用する立場も存在する。(狭義の)VP とは、従来の文字列や記号を中心としたインタフェースに代わって、アイコン、図形、アニメーションという人間がより理解しやすいビジュアルな表現を用いて人間とコンピュータの相互作用を行なう技術 (たとえば漫画プログラミング方式) を意味する [1, 2]。本論文では VP を狭義の意味で使用する。

1.1 VP のメリット

絵やアニメーションなどを用いてプログラムを表現したりその実行の様子を表示することができれば、見る人の持つ直感を視覚的に刺激することができる。さらに絵などによって得られる情報はテキストによって得られる情報よりも遥かに豊富である。つまりテキストとしてではなく絵として情報が表示することにより、実際の情報量としては同等のものを表示しても、はるかに豊かな情報をユーザに獲得させることができる。

VP のメリットの一つは、初心者用のエンドユーザコンピューティングにある。エンドユーザには、従来のようなコマンドベースの入出力に抵抗感がある。しかしながら、VP を使用することにより、抵抗なくシステムを操作することが出来る。また、初心者を対象とするだけでなく、熟練者にも VP は、有用と思われる。VP は人間に対して自然であり、VP をうまく用いれば「疲労度」や「誤り」の少ないヒューマンフレンドリなインタフェースが実現すると考えられる。

1.2 VP のデメリット

一方、プログラミングという作業は高度に知的な集中を要する作業であり、それにはテキストベースのプログラミングが適している。ビジュアルプログラミング技術はまだ稚拙な状態にあり、テキストベースのプログラミングから、VP への移行 (パラダイムシフト) が本当に起こるとは思えないという批判的な意見がある。

この他に、VP への批判的な意見としては以下のようなものがある。

1. 既存言語と比べ機能が不十分、オモチャのシステムしかできない

   一般に VP システムは試験的に作られることが多く、機能も基本機能しかサポートされない。ユーザの数も少なく、使えるマシンも限定されるので、やがてすたれてしまう。プログラミング言語の世界とは組み込み関数の数や、Unix 環境への親和性、ユーザの数、作られた既存ソフトウェアツールの数など、いわば総合力で勝敗の決まる世界である。

2. 図形はかさばり見づらい、大規模プログラムが書けない

   図形で書くとどうしてもかさばって量が多くなる。これは、通常、「普通の書物より漫画の方がわかりやすいがページ数が多くなる」のと同じ原理で

[1] 連絡先: 筑波大学 電子・情報工学系

〒 305-8573 茨城県つくば市天王台 1-1-1

Phone: 0298-53-5343, Fax: 0298-53-5206

Email: jiro@computer.org

ある。またこれに関連して、VP ではオモチャのプログラムは書けても大規模なプログラムは書けないという問題がある。

3. 入力の大変さ

   図形やアイコンには、 その抽象度によってさまざまなレベルがある。しかしながら、図形やアイコンの結合力の弱さから、プリミティブなアイコンを結合させて高度のアイコンを作ることが難しく、その結果図形やアイコンの入力に手間がかかることになる。また、絵心がないとプログラムを書けないと心配する人もいる。

こうした意見には一部もっともな点もあるが、我々は、そのようには考えない。今後のコンピュータ利用の広がりを考慮する時、 VP への移行 (パラダイムシフト) が起こる可能性は十分あると考える。

また、上述の VP のデメリットについては、本質的な問題点とは思われない。既存言語と比べ機能が不十分な点や、図形はかさばり見づらい点については、例えば、図形言語と既存のテキストベース言語を併用するアプローチ、魚眼表示などの表示方法、モジュール化についての工夫などを行なうことによって解決可能であるし、入力の大変さについても今後、研究の余地があると思われる。

2 VP システム PP

我々は、すでに 10 年以上前から、とくに狭義の VP に注目し、VP システム PP(Pictorial Programming) に関して、さまざまな研究 [3, 4, 5, 6, 7, 8, 9, 10] を行なってきた。我々は、ターゲット言語として高水準な宣言型プログラミング言語である並列論理型言語 GHC [11] を対象とし、ユーザがワークステーションから直接に図形を入力することによってプログラミングを行なうとともに、それを実行・デバッグする汎用の VP 環境を提供することを目指した。

ターゲット言語として高水準な宣言型プログラミング言語を想定した理由は、「VP は宣言的である」ので、宣言型言語と親和性がよいからである。また、宣言型プログラミング言語のプログラム実行はプログラム構造のリダクションによって表すことが可能であるので、静的なプログラム入力と動的なプログラム実行とのギャップが少ない。

従来の VP 研究においては、図形言語をテキストベースのプログラミング言語から完全に切り離した形式で提案することが多かった [12, 13]。しかしながらこれでは実用性からほど遠くなる。我々は図形言語と既存のテキストベース言語を併用するアプローチをとった。 すなわちシステムを既存のテキストベースの処理系に寄生させて実現し、図形言語と既存言語との共存を計った。

2.1 オリジナル PP

PP(Pictorial Programming) は、 1989 年から 1993 年にかけて、当初は論理型言語マシン専用ワークステーション PSI 上で、やがて汎用の Unix ワークステーション上で実装された [3, 4]。実装言語としては、 PSI 上のものは GHC と論理型言語 ESP を、 Unix ワークステーション上のものは SICStus Prolog を用いている。また、 GUI としては、 PSI 上のものについては、論理型言語 ESP に組み込みのものを、 Unix ワークステーション上のものについては、 SICStus Prolog に組み込まれた Interviews を用いている。

当初の研究は、素朴に「わかりにくい GHC も図形的に入力するようにしたらいいだろう」との動機でスタートした。我々が実現した PP の機能は、以下のようにまとめられる。

- 図形による入力

  ユーザは図形の組み合わせの形で、グラフィック・ビュー・ウインドウから、 GHC のプログラム節をグラフ構造の形で入力する。

- プログラム・コードの自動作成

  こうしたグラフ構造の形でプログラム節を入力すると、それに応じて、テキスト表現が、自動的に作成され、プログラム・ビュー・ウインドウに表示される。

- 図形表現の自動作成

  逆にテキスト表現をプログラム・ビュー・ウインドウから入力し、図形表現をグラフィック・ビュー・ウインドウに自動的に作成することも出来る。

- 修正機能

  テキスト表現や図形表現は、それをあとで修正できる。

- プログラムの実行機能

  入力されたプログラム・コードや図形表現は、それを図形的に実行することができる。

図 1、図 2に、汎用の Unix ワークステーション上での PP のプログラム例と実行例を示す。

まず、 PP を起動すると、ディスプレイ上に pp という名前を持ったメニューが表れる。メニューには Execute, Define, Load, Halt の四つのボタンがついている。図 1は *append* プログラムを図形的に入力した場合である。メニューの Define ボタンを押すと、 Clause Window が一つ開く。一つの Clause Window は、二つのサブウインドウからなっており、上方の 1/5 くらいがプログラム節のテキスト表現を表示するプログラム・

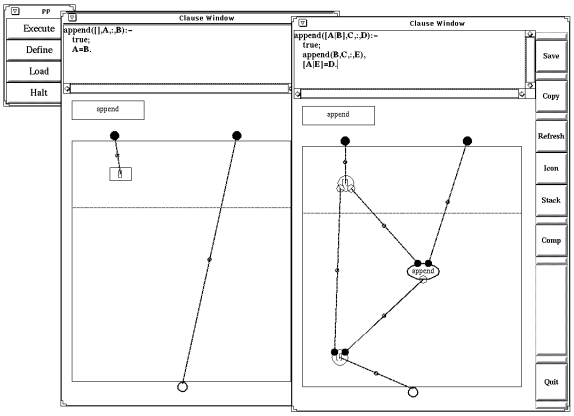

図 1: PP: プログラムの入力例

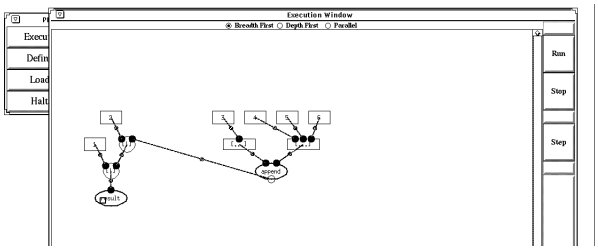

図 2: PP: プログラム実行のスナップショット

ビュー・ウインドウ、また残りの部分がプログラム節の図形表現を表示するグラフィック・ビュー・ウインドウである。

ユーザはこのグラフィック・ビュー・ウインドウから図形の形でプログラム節を入力する。 *append* 述語は二つのプログラム節からなるので、図 1では Clause Window が二つ開かれ、 2 つのプログラム節の定義が示されている。いずれも、グラフィック・ビュー・ウィンドウから図形の形でプログラム節を入力するとプログラム・ビュー・ウィンドウには対応するテキスト表現が自動合成されて表示される。

また、 PP は動的な機能として、ゴールの図形表現をそのまま実行する機能を持っている。それにはメニューの Execute ボタンを押し、 Execution Window を開き、そこに、実行すべきゴールを図形的に入力してやればよい (図 2参照)。この Execution Window を用いて、ユーザは作成したゴールの実行過程を直接ビジュアルに確認したり直接操作することが出来る。

2.2 newPP

オリジナル PP は、それなりに完成度の高いものではあったが、より完成度の高い VP をめざし、 1994 年から 1996 年にかけて新たに実装言語として Tcl/Tk を用

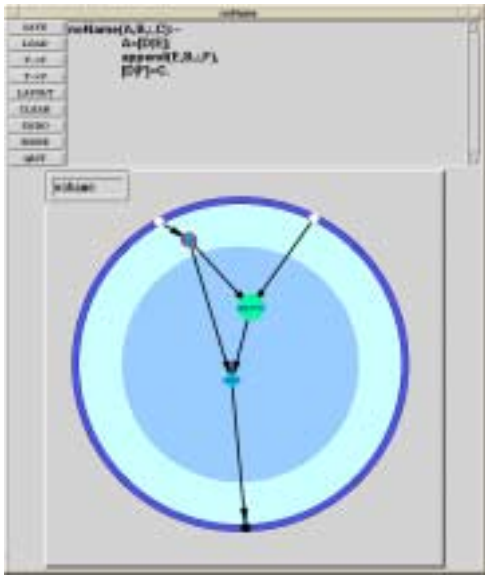

図 3: newPP によるプログラムの入力例

い、とくに PP におけるプログラムの入力の部分を作り直したのが newPP[5] である。 newPP によるプログラムの入力例を図 3に示す。

自然言語のキーボードによる入力は手書きと同程度か、手書きよりも高速でなければ誰も使わない。同様な意味で、 VP の入力についても入力方式の改善を計る必要がある。そこで、 newPP については、グラフ構造の生成をノードとエッジで別々に行なうのではなく一連のつながったノードを順に生成する「数珠つなぎ入力」、複数のノードからあるノードへのエッジを同時に生成する「同時入力」などの新しい入力方式の試験的な実装を行ない、ユーザがストレスなく入力を行なうための工夫を行なった。また、レイアウトの移動過程をアニメーションによりなめらかに表示するようにした。

さらには、この図形入力を突き詰めていくと間違えて入力した際に如何に前の状態に戻るかという Undo の問題にいきあたる。 newPP には、 Tree Undo と呼ぶ Undo 機構を試験的に実装し、他の Undo 機構との比較を行なった [6]。

2.3 viewPP

newPP は (静的な)PP のプログラム入力の部分について実装をし直したものであるが、 1996 年から 1997 年にかけて、 PP の動的なプログラム実行の可視化システム部分について Tcl/Tk を用い作り直したのが viewPP[7] である。 viewPP によるプログラムの実行の例を図 4に示す。

PP ではプログラム実行の様子をグラフ構造の変化で表現するが、 viewPP では、その際のグラフ構造の自動レイアウトとそのアニメーションにこだわった。当初

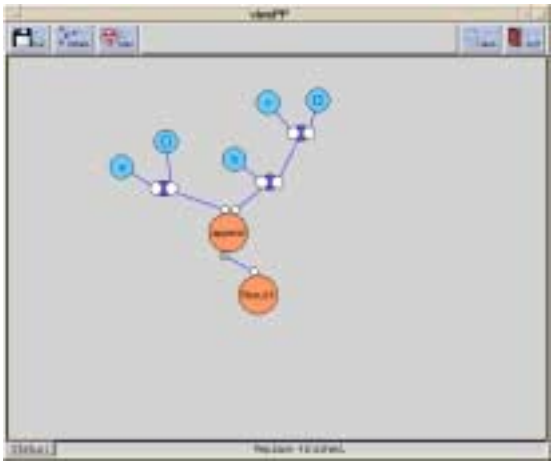

図 4: viewPP によるプログラムの実行例

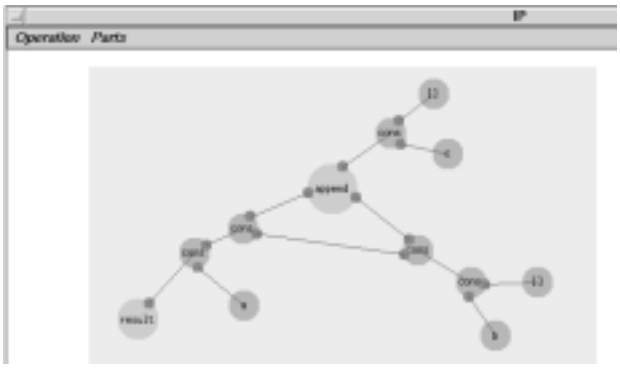

図 5: PP-pad によるプログラム実行の例

は、自動レイアウトにはグラフ描画アルゴリズムの採用が重要であるとの観点から、基本となる Eades のグラフ描画アルゴリズム [14] の発展形である鈴木のアルゴリズム [15] をさらに改良するなどの検討を行なった [8]。

viewPP におけるアニメーション表示にあたっては、従来のような「コマ割り」といった処理は行なわず、グラフのノードの再配置の過程をリアルタイムで逐次表示することにした。このため、描画アルゴリズムについては、単純で高速に計算できるアルゴリズムが望まれる。したがって、グラフ描画アルゴリズムは鈴木のアルゴリズムの改良などの複雑なものではなく、むしろ単純な Eades のアルゴリズムで良いことがわかり、結局 Eades のアルゴリズムにアニメーション向けの改良を行なった。また、 VP はかさばり大きなプログラムはかけないという問題への一つの解決策として、 Eades のアルゴリズムへ多視点遠近画法を viewPP に組み込んだ [7]。

2.4 PP-pad

オリジナル PP においては、静的なプログラムの入力と動的なプログラムの実行とは、同様の視覚化のフレームワークを用いているにもかかわらず両者を区別している。

我々は VP の検討を行なううちに、両者を区別すべきでなく、むしろ意図的に両者を単一環境の中に統合すべきであると考えるようになった。この単一環境のことを PP-field と呼ぶ。

我々は、 1997 年から 1998 年に、とくにこの単一環境の実現を目的として、 PP-pad を実装した [9, 10]。実装には北大で開発された IntelligentPad[16] を用いている。図 5に PP-pad によるプログラム実行の例を示す。

一人の料理人がたった一つのマナイタの上で次から次に料理を作り出すのと同様、たった一つの PP-field の中で、静的なプログラムの入力と動的なプログラムの実行の両方が行なわれる。ここでは、マナイタの上のプログラムの構造を、ユーザが好きなようにいじるという「直接操作」が強調される。プログラムのデータがそろうと自動的に計算が開始され、必要な時には評価を巻き戻すこともできる。評価例からプログラムを作成することもできる。このコンセプトは中野によって提案されたものである [17]。

3 3DPP

前章に述べたように、我々はいくつかの PP ファミリーの設計と実装を行なってきたが、こうした VP の発展形として、我々は PP を 3 次元に拡張した 3DPP の研究を 1998 年より開始している [18]。

3.1 3 次元 VP のメリット

3 次元視覚化については、すでに小池によるサーベイ [19] があるが、われわれが 3 次元に着目したのは以下の理由からである。

- 大規模プログラムへの対応

  VP のデメリットの項で述べたように、図形はかさばるので大規模プログラムに対応できないという問題点がある。この問題の解決策の一つとして 3 次元表示が考えられる。 2 次元表示では、例えば、縦方向と横方向に 5 個の要素を描画可能とすれば、 5 の 2 乗である 25 個の要素を描画可能である。しかしながら 3 次元を使えば、 5 の 3 乗である 125 個の要素が描画可能となり、記述能力が飛躍的に向上する。

- リアリティと直接操作

  実世界は 3 次元であり、 3 次元図形を用いることにより、よりリアルにプログラムを表現することができる。また、 3 次元図形を操作する時の操作

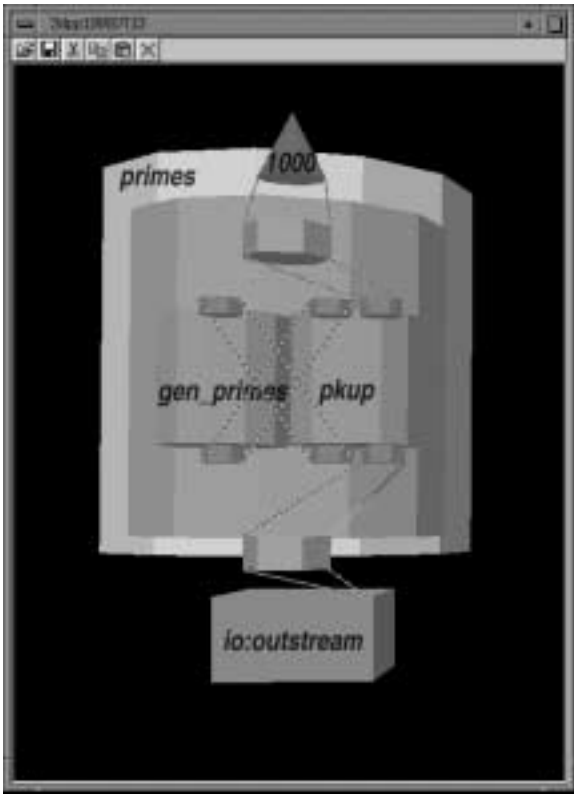

図 6: 3DPP のプログラムイメージ

感もリアルである。3 次元図形をいろいろ視点を変えて観察し、それに直接操作を行なえるようにすることによってよりプログラムをリアルに理解することが可能となる。

- レイアウトの自由度

  2 次元図形ではいくらレイアウトを工夫してもグラフ構造においてノードが重なったりエッジが交差したりすることが多々ある。 3 次元図形では自由度が一つ増えるため重なりや交差を避けることができる。

3.2 3 次元モデラ Claymore

3 次元環境を構築する上で避けて通ることができないのが、コンテンツ提供の問題である。コンテンツの製作において、もっとも基礎の部分にあたるのが 3 次元形状を作成するモデラ (図形エディタ) である。

3 次元モデラについては、これまでもさまざまなツールが作られており、商品も幾つか既に存在する。しかしながら、それらはほとんどが専門家を対象とした CAD ツールであり、 3 次元の入力に直接座標値指定や 3 面図による指定などを使うものが多く、操作はヒューマンフレンドリではなかった。

我々は、まず 3 次元 VP 用のモデラを作成することから研究を開始した。我々が目標としたは、 MacDraw の 3 次元版のようなお手軽モデラである。我々はすでに Claymore [20] と呼ぶ 3 次元モデラを試作している。Claymore は、 CAD に使われるような精密で複雑な図形を正確に描くのではなく、 MacDraw のように不正確でも直観的な方法で図形を描くことを目的としている。

Claymore では、ユーザは以下の手順でモデルを作成する.

1. 3 次元アイコンを用いて基本物体を生成する。 3 次元アイコンは常に地面の上に表示されていて、ユーザがクリックすることによって基本物体を生成する。
2. 切断および変形をすることによって、より複雑な形状をした部品を作成する。
3. 個々の部品にグループ化を行って部品同士を組み合わせる。
4. こうして作られた部品を、移動・回転によって配置する。
5. 作成したモデルのモデルデータを VRML 形式または Wavefront OBJ 形式にセーブする。

3.3 3DPP の実現

3DPP のプログラムイメージを図 6に示す。本システムでは、ゴールやアトムなどのプログラム要素を 3 次元図形で表現する。ユーザはこれらのプログラム要素を表わす 3 次元アイコンを使用して 3 次元図形を組合わせることでプログラミングをすることができる。図 6は、1000 番目の素数を io:outstream に出力するゴールを与えたところを示している。本体の primes ゴールは、gen_primes で素数を 2、 3、 5、 7 と生成し、 pkup がその素数を吸収していき、ちょうど 1000 番目の時にその時の素数を出力する。

我々は 3DPP の開発にあたって、 3DPP に必要な数多くのプログラム要素を、総て Claymore を使用して作成した。例えば 3DPP においてゴールは円柱の形状として表現される。また、プログラムの意味合いによって、ゴールには異なる数種類の色が付けられる。

実際に Claymore を使用して 3DPP のゴールの形状データを作成する際には、まず 3 次元アイコンによって円柱を生成し、切断機能によって円柱の上下部の角を削ることによって、ある程度の丸みを出す。次に、カラーリング機能によって適切な色を付ける。最後に、作成したゴールの形状データをセーブ機能によってファイルに保存する。 3DPP では Claymore によってセーブされたファイルをロードすることによって、ゴールの形状データを利用することが出来る。

Claymore で作成・編集された 3 次元図形は、単なる図形の集合であり そのままではプログラムとしての意味を持たない。一般に、 VP には図形の形状・配置の解析を行う空間パーサ (Spatial Parser) のモジュールがある。これはテキスト言語の処理系における字句解析器・構文解析器にあたる。我々の研究室ではすでに図形に意味や制約を与える空間パーサを持ったシステム “恵比寿” を開発している [21]。現段階では “恵比寿” に入力される構文要素は 2 次元図形であるが、今後 Claymore を用いて作成した 3 次元図形に意味や制約を与えるようなシステムを開発する予定である。

4 関連研究と本研究の特色

3 次元の VP の提案としては、すでに ToonTalk[22] Visulan[23] などがある。 ToonTalk は子供用のアニメーション環境、 Visulan はビットマップの書き換えによるプログラミングに研究の重点がおかれており、汎用の VP 環境をめざす 3DPP とは異なっている。

ToonTalk や Visulan が個々の VP 環境の構築を目指しているのに比べ、 3DPP の場合にはモデラや空間パーサなど、より一般的な VP 環境の構築ツールに研究の重点がおかれているところも特色である。

参考文献

[1] 田中二郎, 神田陽治編: インタフェース大作戦: グループウェアとビジュアルインタフェース, 共立出版, 1995.

[2] 田中二郎: ビジュアルプログラミング, ビジュアルインタフェース – ポスト GUI を目指して –, bit 別冊, pp. 65–78, 1996 年 2 月.

[3] 田中二郎: 並列論理型言語 GHC のビジュアル化の試み, インタラクティブシステムとソフトウェア I, 日本ソフトウェア科学会 WISS'93, 近代科学社, pp. 265–272, 1994.

[4] J. Tanaka: PP: Visual Programming System for Parallel Logic Programming Language GHC, *Proceedings of the IASTED International Conference Parallel and Distributed Computing and Networks* (PCDN'97), pp. 188–193, August 1997.

[5] 田中二郎, 後藤和貴, 馬場昭宏: ビジュアルプログラミングシステムにおける入力法の効率化, 日本ソフトウェア科学会第 12 回大会論文集, pp. 165–168, 1995 年 9 月.

[6] 馬場昭宏, 田中二郎: ビジュアルプログラミングシステムのための UNDO 機能, 日本ソフトウェア科学会第 13 回大会論文集, pp. 409–412, 1996 年 9 月.

[7] 南雲淳, 田中二郎: インタラクティブシステムのためのグラフ描画アルゴリズム, インタラクション'98 論文集, 情報処理学会, pp. 41–48, 1998 年 3 月.

[8] 中野勝次郎, 田中二郎: ビジュアルプログラミングシステムにおけるモデルの視覚化アルゴリズム, インタラクティブシステムとソフトウエア II, 日本ソフトウェア科学会 WISS'94, 近代科学社, pp. 205-214, 1994.

[9] 遠藤浩通, 田中二郎: パッドベースシステムによるビジュアルプログラミングシステムの構成, コンピュータ・ソフトウェア, 日本ソフトウェア科学会, Vol. 15, No. 1, pp. 50–53, 1998.

[10] H. Endoh and J. Tanaka: Integrating Data/Program Structure and their Visual Expressions in the Visual Programming System, *Proceedings of Asia Pacific Computer Human Interaction 1998*, IEEE Computer Society Press, pp. 453–458, July 1998.

[11] K. Ueda: *Guarded Horn Clauses*, ICOT Technical Report, TR-103, 1985.

[12] The Gunakara Sun Systems: *Prograph Tutorial*, The Gunakara Sun Systems, 1989.

[13] K.M. Kahn: Concurrent Constraint Programs to Parse and Animate Pictures of Concurrent Constraint Programs, *Proceedings of the International Conference on Fifth Generation Computer Systems 1992*, pp. 943–950, June 1992.

[14] P. Eades: A Heuristics for Graph Drawing, *Congressus Numeranitium*, Vol. 42, pp. 149–160, 1984.

[15] 鈴木和彦, 鎌田富久, 榎本彦衛: 単純無向グラフ自動描画アルゴリズム, コンピュータ・ソフトウェア, 日本ソフトウェア科学会, Vol. 12, No. 4, pp. 45–55, 1995.

[16] Y. Tanaka and T. Imataki: IntelligentPad: A Hypermedia System Allowing Functional Composition of Active Media Objects through Direct Manipulations, *Proceedings of IFIP 11th World Computer Congress*, pp.541–546, 1989.

[17] 中野勝次郎: ビジュアルプログラミングシステムにおける実行の視覚化, 筑波大学第三学群情報学類卒業研究論文, 1995 年 3 月.

[18] 宮城幸司, 大芝崇, 田中二郎: 三次元ビジュアル・プログラミング・システム 3D-PP, 日本ソフトウェア科学会第 15 回大会論文集, 1998 年 9 月.

[19] 小池英樹: インタラクティブ 3 次元情報視覚化, コンピュータ・ソフトウェア, 日本ソフトウェア科学会, Vol. 11, No. 6, pp. 20–31, 1994.

[20] J. Tanaka, H. Mitsunobu and T. Oshiba: Claymore: Three-Dimentional Modeling Tool, *Proceedings of 20th International Conference on Software Engineering*, IEEE Computer Society, Vol. 2, pp. 67–72, 1998.

[21] 馬場昭宏, 田中二郎: Spatial Parser Generator を持ったビジュアルシステム, 情報処理学会論文誌, Vol. 39, No. 5, 1998, pp. 1385–1394.

[22] K.M. Kahn: ToonTalk – An Animated Programming Environment for Children, *Journal of Visual Languages & Computing*, Vol. 7, No. 2, pp. 197–217, June 1996.

[23] 山本格也：ビットマップ型言語におけるモジュール機能, 情報処理学会論文誌, Vol. 38, No. 12, pp. 2544–2551, 1997.