筑波大学大学院修士課程

理工学研究科修士論文

# 三次元ビジュアルプログラミング環境の構築

宮　城　幸　司

平　成　11　年　2　月

筑波大学大学院修士課程

理工学研究科修士論文

# 三次元ビジュアルプログラミング環境の構築

宮　　城　　幸　　司

主任指導教官　電子・情報工学系　田中 二郎

# 目次

# 図一覧

# 要旨

近年のユーザインタフェース (UI) の研究を背景に、プログラミング環境においてもテキストベースの UI から、視覚的な表現を用いた UI によるビジュアルプログラミング (VP) の研究が進められている。我々は大規模プログラムへの対応、リアリティの向上、レイアウト自由度の向上の点で VP 環境の三次元化が有効と考え研究を進めている。本論文では二次元 VP におけるレイアウトとズーミングの手法 “Bubble-Gum” をベースに、三次元 VP の構築手法を提案する。 Bubble-Gum で採用していた力指向のアルゴリズムを改良し、気圧によってノードサイズを制御する手法や、中間層を省略しグラフを簡素化する手法等について述べる。これらにより三次元階層グラフのレイアウトやズーミングが可能となる。更には適量の情報をユーザに提供することが可能となる。

# 第 1 章

# 序論

近年のユーザインタフェースの研究により、計算機のインタフェースはキャラクタユーザインタフェース (CUI) からグラフィカルユーザインタフェース (GUI) へ移行してきた。さらに情報をテキストではなく図として表示することにより人間の情報に対する理解をより早く、より深くすることを目的とした情報視覚化の研究も進められている。近年、二次元空間上に表現されてきた情報を三次元空間上に表現する試みが行われている。しかし二次元で十分なものをわざわざ三次元にする必要はない。三次元情報視覚化のサーベイ [1] では、二次元では不可能であり三次元視覚化によって効果を上げた例を挙げている。階層データを三次元の木を表示する Cone Tree[2] は、三次元によって巨大階層構造の視覚化に成功している。三次元の壁上にデータを表示する Perspective Wall[3] は、透視投影図法を利用して局所的詳細と大局的概略を統合して表示することを可能にした。

情報視覚化の応用分野に**ビジュアルプログラミング (VP)** がある。 VP は、テキストを中心としたインタフェースから、アイコン、図形、アニメーションといったより理解しやすいビジュアルな表現を用いたインタフェースを用いたプログラミングである。我々は VP の三次元化について以下の点で着目している。

- プログラムの大規模化： VP は量がかさばる傾向にあり、大規模プログラムに対応できないという問題点がある。プログラムを三次元空間に配置すれば、一画面で扱えるプログラムの量は飛躍的に向上する。

- リアリティ：我々の住む現実世界は三次元空間である。プログラミング環境も三次元で構築したほうがより自然であり、リアルな表現が可能となる。

- レイアウトの自由度：二次元空間では、いくら見やすくレイアウトしようとしても、図形が交差したり重なったりすることが多々あるが、三次元では自由度

が一つ増えるため重なりや交差を避けることができる。

我々はこれらの理由から、二次元 VP である PP(Pictorial Programming)[4, 5] を三次元に拡張した **3D-PP**(Three-dimensional PP)[6] の研究を進めている。では我々の研究は VP 言語の分野ではどのようにクラス分けされるだろうか。VP 言語をクラス分けするシステム： A Classification System for Visual Programming Languages [7] では、

VPL: Visual Programming Languages

- VPL-I: VPL のためのツール、環境
- VPL-II: 言語の種類
- VPL-III: 言語の特徴
- VPL-IV: 言語の実装法
- VPL-V: 言語の目的
- VPL-VI: VPL の理論

に分類される。またそれぞれのカテゴリにはサブカテゴリが存在する。以下に我々の研究と関連のあるカテゴリのサブカテゴリを示す。

VPL-II: 言語の種類

A: パラダイム

1. 並列言語
2. 制約言語
3. データフロー言語
4. フォームベース、スプレッドシートベース言語
5. 関数型言語
6. 既存の手続き型言語
7. 論理型言語
8. マルチパラダイム言語
9. オブジェクト指向言語
10. Programming-by-demonstration 言語
11. ルールベース言語

B: 視覚的表示法

1. ダイアグラム言語
2. アイコン言語
3. 静的な絵の列に基づく言語

VPL-V: 言語の目的

A. 汎用言語

B. データベース言語

C. 画像処理言語

D. 科学計算視覚化言語

E. ユーザインタフェース生成言語

VPL-VI: VPL の理論

A. VPL の形式的定義

B. アイコン理論

C. 言語設計の諸問題

我々の研究は「VPL-II: 言語の種類」の中の「A: パラダイム」においては「7. 論理型言語」に、「B: 視覚的表示法」においては「1. ダイアグラム言語」に、「VPL-V: 言語の目的」においては「A. 汎用言語」に、「VPL-VI: VPL の理論」においては「C: 言語設計の諸問題」に分類される。我々は中でも特に「VPL-II-B: 視覚的表示法」において三次元グラフ (ダイアグラム) の表現手法とそれに対するインタラクションを研究課題の中心に据えている。

本論文では 3D-PP の開発にあたり、三次元 VP 環境構築のための要素技術、及び、構築論理を提案する。三次元環境の構築にあたりどのような表現手法が適しているのか、情報とのインタラクションを効率よく行うにはどうしたらよいかに焦点をあてる。まず第 2章では、 3D-PP の概要とその特徴について述べる。三次元では巨大な情報空間との対話が必要となる。我々は三次元情報に対するユーザの認知的負荷の軽減と操作性の向上が重要と考える。第 3章では、 VP 環境の構築に必要となる要素技術を上記の観点から述べる。第 4章では、二次元 VP において要素技術 (グラフの自動レイアウトと非線形ズーミングの技術) の改良による操作性の向上について述べる。第 5章では、二次元 VP での研究をもとに三次元 VP 環境の構築論理、及び、要素技術を提案する。

# 第 2 章

# 三次元 VP システム: 3D-PP

我々の研究グループでは、ユーザが図形を入力することによってプログラミングを行うとともに、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て PP の研究を行ってきた。 PP は既存のテキストベースの処理系に寄生させて構築することで実用性を高めたシステムである。テキストでも図形でも入力することができ相互に編集の結果が反映される。プログラム実行の様子はグラフ構造の変化をスナップショットの形で表現される。 newPP[8] では PP の図形入力の部分にユーザがストレスなく図形を入力するための手法を試験的に導入した。 viewPP[9] では、プログラム実行の様子を表現する部分に新たな手法を導入した。 viewPP は変化していくグラフを自動レイアウトによって再配置し、その過程を逐次表示することで実行のアニメーション表現を実現した。

二次元図形を使用した VP システムの研究には我々のグループの他にも数々の研究がなされてきた [10, 11]。しかし二次元 VP システムは、コンピュータの限られた画面サイズでは、一度に大規模なプログラムを扱えないという問題 (Small Screen Problem[12]) を指摘されてきた。さらに Burnett らは一歩進んで、有用な VP を構築するために、表示手法の問題の他にプログラミング言語の問題なども加えて Scaling-up Problem[13] をまとめている。

そこで現在、従来の視覚化手法である二次元グラフではなく、三次元による視覚化の新しい手法を取り入れた実用的な VP システム "**3D-PP**" の研究・開発に取り組んでいる。 3D-PP は並列論理型言語を VP システムの対象にしており、並列論理型言語の特徴であるゴールのリダクションの過程や論理変数の具体化に着目した表現手法を提案する。並列論理型言語の特徴である実行過程に着目し、その様子を素直に視覚化した。そのため従来の表現手法では理解しづらかった、並列論理型言語の振る舞いをユーザに提供できるようになった。ユーザは三次元アイコンを使用して図形を組合

せていくだけで実用的なプログラムを記述できる。実行の様子も、記述したビジュアルプログラムがアニメーションにより変化していく形で視覚化した。

## 2.1 三次元 VP システムのメリット

我々は三次元図形による視覚化によって、ユーザが享受できるメリットに以下のようなものがあると考えている。

### 2.1.1 大規模プログラムへの対応

ビジュアルプログラムは図形でプログラミングをするため量がかさばる傾向にあり、大規模プログラムに対応できないという問題点がある。その解決法として、 fisheye-view[14] などいくつかの手法が提案されている。それでも二次元図形では、一画面に表示できるプログラムの量に限界がある。プログラムを三次元空間に配置することにより、一画面で扱えるプログラムの量は飛躍的に向上する。例えば二次元空間では一画面に $10 \times 10$ で 100 個のノードを表示できるとすると、三次元空間では $10 \times 10 \times 10$ で 1000 個のノードを表示可能である。

### 2.1.2 リアルな表現力

我々の住む現実世界は三次元空間である。プログラミング環境も三次元で構築したほうがより自然であり、三次元空間上の位置関係によって情報の構造や意味を把握しやすい。三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。 3D-PP では、親ゴールとサブゴール群との関係を包含関係で表示するが、その様子を、まさに親ゴールの中にサブゴール群が存在する形で表現できる。また、それぞれのプログラム要素に三次元図形を割り当てるわけだが、同類の要素には上下方向から見た形を統一し、横から見ると各々の要素に合わせた形にするといった表現ができる。このような表現は二次元図形では難しい。また、実行の過程をアニメーションで表示する場合、二次元図形と三次元図形では表現力に多大な差が出てくる。

### 2.1.3 レイアウトの自由度

二次元図形ではいくら見やすくレイアウトしようと工夫しても、図形が交差したり重なったりすることが多々ある。しかし三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。

## 2.2 並列論理型言語と VP システム

VP システムの設計に際して、並列論理型言語 KL1[15] を視覚化の対象にした。プログラムの実行はゴールをサブゴールに書き換えることによって進められる。各々のゴールは、条件が満たされたものから順次書き換えられていく。ゴールは書き換えることのできないゴール “true” になった時点で書き換えを終了する。ゴールを “true” になるまで書き換えていく過程をリダクションと呼ぶ。並列論理型言語はプログラムが簡潔であるなど VP システムの対象としてふさわしい [16]。我々は、並列論理型言語の特徴を前面に押し出し、並列論理型言語を素直に視覚化した。よってユーザは対象言語のプログラム構造や実行過程を理解しやすく、可読性のよい VP システムということができる。

### 2.2.1 ゴールのリダクションの表現法

並列論理型言語は、手続き型言語と異なりどのゴールからリダクションされるかは処理系依存である。処理系は処理可能なゴールに対して、順次リダクションを試みる。リダクションされるか否かはガード部分に記述される条件を満たすデータが揃うか否かで決まる。データが揃わない場合は、そのゴールのリダクションを中断し、別の処理可能なゴールのリダクションを試みる。データが揃えばゴールがリダクションされ、サブゴールが処理可能なゴールとして生み出される。処理可能なゴールが無くなった時点で実行は終了する。リダクションの条件が満たされたものが複数ある場合は、それらが並列にリダクションされる。この様子を明示的に表現するために、各々のゴールを三次元空間に浮遊させて示した。

まず処理系は各ゴールに対してリダクションを試みるが、これをゴールの点滅表示で表現する。リダクションの条件が満たされるかどうかを、入力データと条件部分とを照合するアニメーションで表現する。条件が満たされない場合は、ゴールの点滅表示をやめ、他のゴールへのリダクションを試みる。リダクションを行えるデータがそろっている場合は、サブゴール群を生み出して消滅する。プログラムの実行は、このようにリダクションの繰り返しによって進んでいく。画面中にリダクション可能なゴールがなくなったとき、実行は終了する。

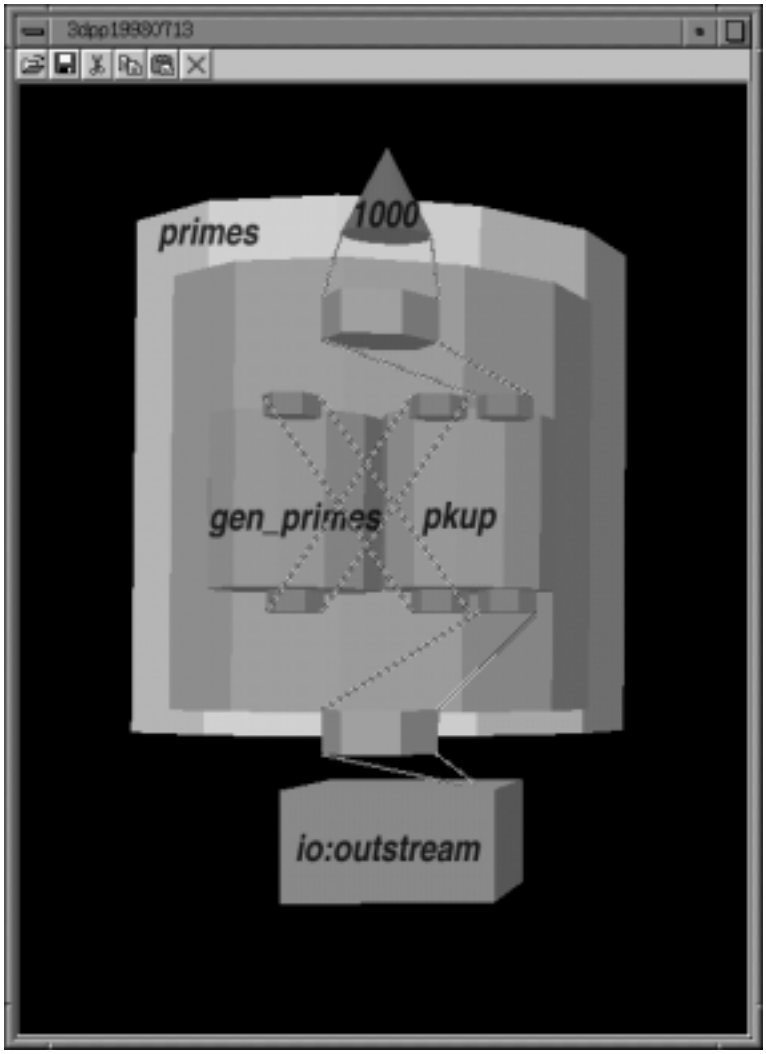


図 2.1: 3D-PP の画面イメージ

### 2.2.2 論理変数

各ゴールの関係は論理変数によって記述される。論理変数は手続き型言語でいう変数とは異なり、一度値が具体化すると二度と変化しない。論理変数が既に具体化されているのか否かを提示できれば、実行の状況をより詳しく観察できる。そこで具体化された論理変数と具体化されていない論理変数の表現を異なるものにした。このことによりテキストでは分かりにくかった論理変数の振る舞いを、明示的に表現することを可能とした。

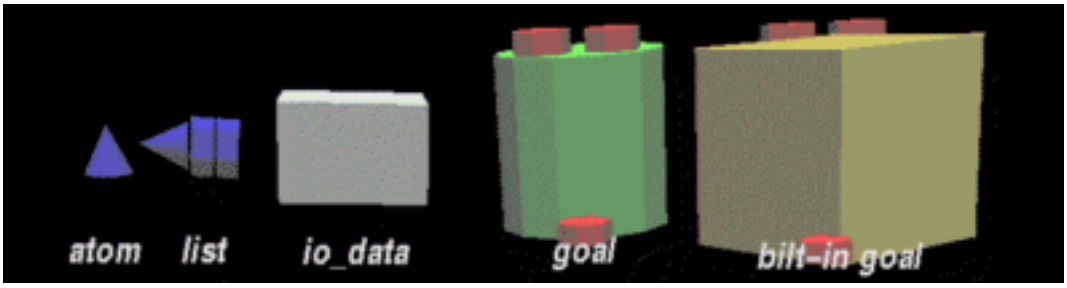


図 2.2: 3D-PP のプログラム要素

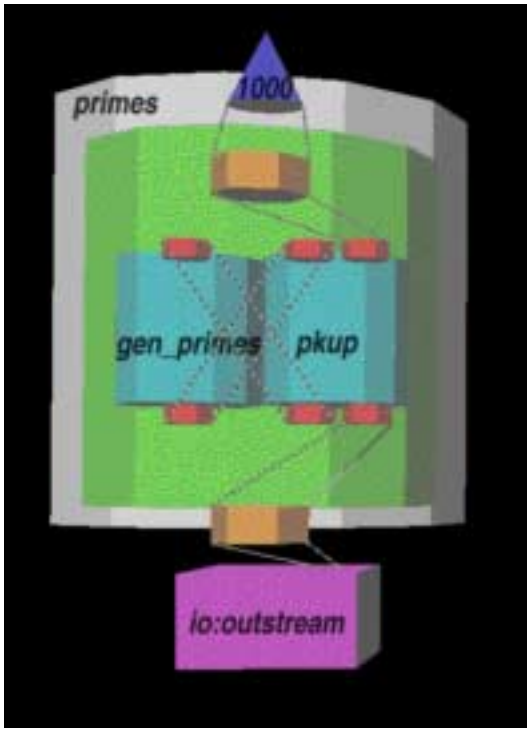


図 2.3: ビジュアルプログラム例

```
:-module main.
main:-primes(1000,LP),io:outstream([print(LP),nl]).
primes(Nth,LP):-gen_primes(AB,Ps),
pkup(Ps,Nth,AB,LP).
gen_primes(AB,Ps):-gen(AB,2,Ns)@lower_priority,
        sift(Ns,Ps).
gen(abort,_,Ns):-Ns=[].
alternatively.
gen(AB,N,Ns):-Ns=[N|Ns2],N2:=N+1,gen(AB,N2,Ns2).
sift([],Zs):-Zs=[].
sift([P|Xs],Zs):-Zs=[P|Zs2],filter(P,Xs,Ys),
sift(Ys,Zs2).
filter(_,[],Ys):-Ys=[].
filter(P,[X|Xs],Ys):-X mod P=\=0 |
        Ys=[X|Ys2],filter(P,Xs,Ys2).
filter(P,[X|Xs],Ys):-X mod P=:=0 |
        filter(P,Xs,Ys).
pkup([P|_], Nth,AB,LP):-Nth=:=1 |
        LP=P, AB=abort.
pkup([_|Ps],Nth,AB,LP):-Nth=\=1 |
        Nth2:=Nth-1,pkup(Ps,Nth2,AB,LP).
```

図 2.4: 左図に対応する KL1 コード

## 2.3 三次元 VP システム “3D-PP”

我々は、三次元 VP システム “**3D-PP**” を実装した。このシステムの画面イメージを図 2.1に示す。この図は 1000 番目の素数を計算し出力するプログラムを記述したところを表わしている (詳細は後述する)。

3D-PP では、ゴールやアトムなどのプログラム要素は三次元図形で表現される。プログラムを構成する三次元ノードのうち主なものを図 2.2に示す。左からアトム、リスト、入出力データ、ユーザ定義のゴール、組み込みのゴールである。ユーザは三次

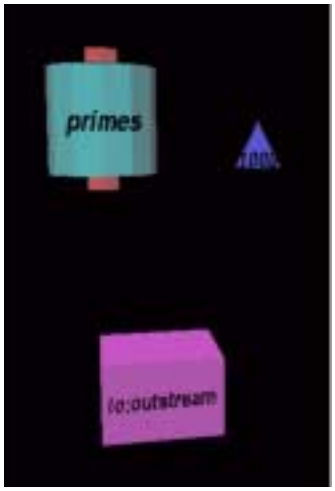


図 2.5: 3D-PP のプログラミング例 a

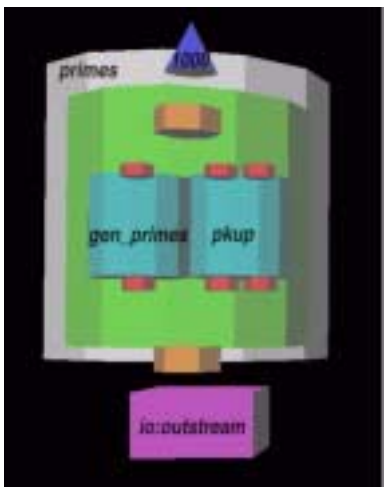


図 2.6: 3D-PP のプログラミング例 b

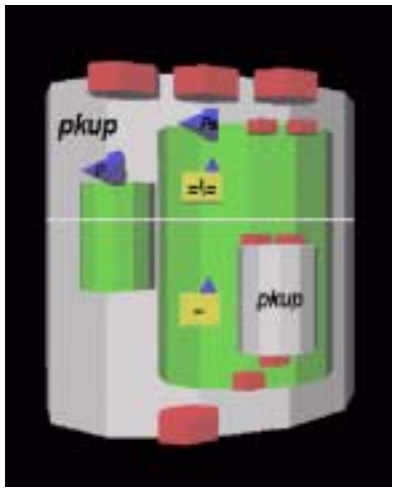


図 2.7: 3D-PP のプログラミング例 c

元アイコンを使用してこれらのプログラム要素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。

1000 番目の素数を計算し出力するビジュアルプログラム (図 2.3) に対応する KL1 のコードを図 2.4に示す。 primes は 1000 番目の素数を計算し出力するゴールを表わす。 gen_primes は素数列を生成し、 pkup がその素数列から 1000 番目の素数を取出し primes の出力に渡す。 primes が出力したデータを io:outstream が格納する。

ユーザは任意の階層までを画面に表示でき、図 2.1では gen_primes, pkup 以下の階層は表示させていない。さらに gen_primes, pkup の中身として gen, shift, pkup(再帰ゴール) などがある。

### 2.3.1 3D-PP の操作・実行例

では実際にビジュアルプログラミングの過程を 1000 番目の素数を求めるプログラム (図 2.3) を例にとって説明する。

#### 操作例

プログラム記述中のスナップショットを図 2.5, 2.6, 2.7に示す。まず、 1000, primes, io:outstream を画面上に作成する (図 2.5)。次に gen_primes, pkup を作成し primes の中に入れる (図 2.6)。随時各要素の引数を論理変数を表すエッジで結ぶ。同様に gen_primes, pkup の中身も作成していくことで、プログラムを完成することができる。図 2.7は定

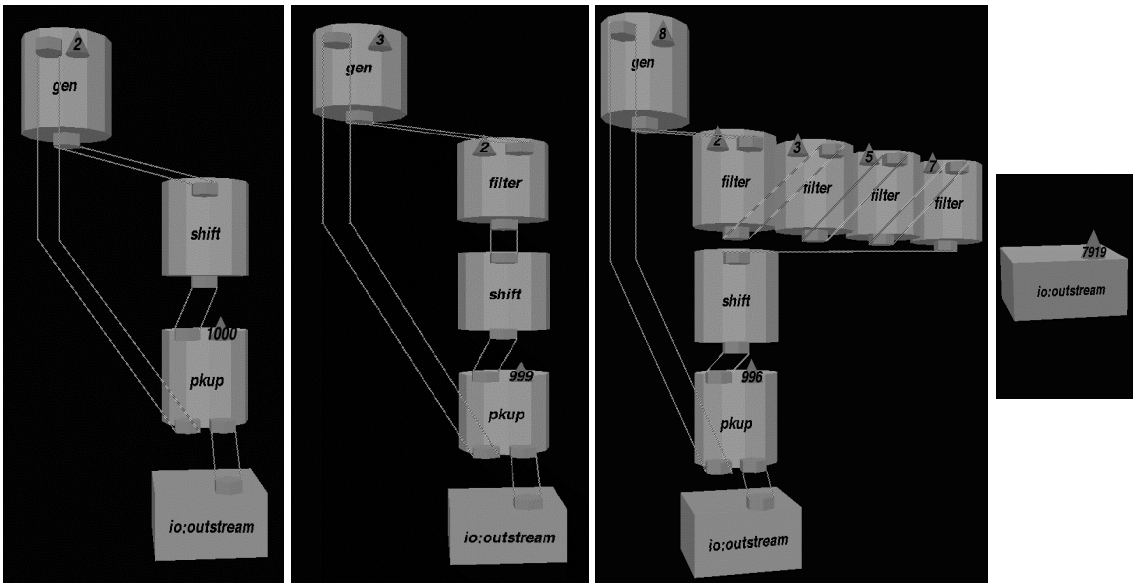


図 2.8: 実行イメージ

義節が複数存在するゴールの例で pkup を記述したところである。内側の大小 2 つの円柱が定義節を表している。左側の定義節は注目していないため縮小されている状態である。中央付近の水平な線はガードとボディの境界線で、上側がガード、下側がボディである。ガードにはその定義節が実行される条件が記述される。ガードに記述された条件が満たされとき、ボディに記述されたサブゴールに書き換えられる。右側の定義節には再度 pkup が記述されているが、再帰的なプログラムはこのように記述する。

### 実行例

図 2.3のプログラムの実行イメージを図 2.8に示す。図 2.8-a は、 gen_primes がリダクションして、 gen と shift が生成された時点のスナップショットである。 gen は自然数列を生成し、 shift はその数列から素数の倍数を除去してサブゴールに渡す。図 2.8-b は shift と pkup がリダクションした状態であり、自然数列から 2 の倍数を除去する filter が生成されている。素数が一つできたので、 pkup の数字が 999 に減っている。図 2.8-c は素数が 2, 3, 5, 7 と生成され、 pkup の数字が 996 になった状態を表わしている。 pkup の数字が 1 になった時点の素数が 1000 番目の素数として出力される。図 2.8-d は実行の最終段階を示しており、 1000 番目の素数が出力データとして io:outstream に格納された状態である。

### 2.3.2 3D-PP のシステム構成

本システムは「GUI 部」、「トランスレータ」、「KL1 エンジン」の 3 部構成になっている。「GUI 部」はユーザとのインタラクション全般を請け負う。プログラミングに関するあらゆる情報をユーザにビジュアルに提供する。ユーザは「GUI 部」においてプログラミング、デバッグを対話的に行う。「トランスレータ」はビジュアルプログラムと KL1 のプログラムコードとを相互に変換する。 KL1 でコーディングされた既存のプログラムを「トランスレータ」でビジュアルプログラムに変換し「GUI 部」に表示したり、逆に 3D-PP でプログラミングしたビジュアルプログラムを KL1 のプログラムに変換し KL1 の処理系で高速実行することも可能である。「KL1 エンジン」は 3D-PP で記述されたビジュアルプログラムの実行をトレースし、「GUI 部」に結果を返す。返された結果は「GUI 部」でアニメーション表現となってユーザに提示される。

3D-PP 内では、ビジュアルプログラムを本システム用に設計した内部表現の形で扱っている。 3 つのサブシステムは内部表現の形式でデータを交換する。例えばプログラム実行は、まず実行されるプログラムの内部表現が「KL1 エンジン」に送られ、次に「KL1 エンジン」によって作成された実行ログを内部表現として「GUI 部」に返される。

## 2.4 3D-PP に関連する研究

3D-PP に関連する三次元 VP 環境の研究として、 VisuaLinda[17], Cube[18], ToonTalk[19], 3D-Visulan[20], SAM[21] などがある。 VisuaLinda, Cube は我々と同様、並列論理型言語を視覚化の対象としている。 VisuaLinda は並列論理型言語 Linda の実行状態を三次元グラフィクスで視覚化したものである。 xy 平面に着目して見るとプロセスの数やプロセスと Linda サーバ間との関係を表わし、 yz 平面に着目して見ると各プロセスの時間変化と通信状況を表わすように三次元図を構成している。二種類の情報を 2 つの二次元図で視覚化しそれを三次元図に統合することで、一つの図から他大な情報を得ることができる点で興味深い。 VisuaLinda は実行状態のみの視覚化システムであるが、 3D-PP はプログラミングから実行までを視野にいれたシステムである。

ToonTalk は子供の教育用のシステムであり、親しみやすい子供や鳥、家、道具箱といったキャラクタを三次元図形で登場させ、それらを操作することでプログラムを学ぶことができる。

3D-Visulan は三次元ビットマップでプログラムを構成する。左側に使用前のビット

マップパターン、右側に使用後のビットマップパターンを描くことでプログラミングを行う。左側に描かれたビットマップとパターンが一致すると右側のビットマップに書き換えられることでプログラムの実行が進んでいく。プログラムはビットマップパターンの組であり、 3D-PP の三次元グラフによるプログラムとは異なった表現である。

SAM は三次元オブジェクトで表わされたプログラム要素を、アニメーションによって本物のように動作させることができるシステムである。プログラムはメッセージの交換によって通信する複数のエージェントによって構成されている。エージェントは球で表わされ、メッセージを受信すると実行されるルールはエージェントの中に包含されている。言語パラダイムはよく類似しており、視覚化手法も類似している。ただ 3D-PP のグラフは深い階層構造を持つ傾向にあり、それを適正に表現する工夫が必要である。

# 第 3 章

# 三次元 VP における要素技術

我々は三次元 VP の構築にあたり、情報に対するユーザの認知的負荷の軽減と操作性の向上が重要と考える。本章では、情報との効率的なインタラクションのための技術とプログラムの早く深く理解するための技術について述べる。

## 3.1 三次元インタラクション技術

三次元図形とのインタラクションデバイスとして二次元マウスと三次元用の特殊なデバイスがある。我々はワークステーションやパーソナルコンピュータなどの一般的な環境下での VP 環境の構築に興味がある。とすると二次元的にしか操作を伝えられないマウスで三次元図形を直接操作するための枠組が必要となる。インタラクションにはオブジェクトの位置、姿勢に対する操作 (Manipulation) とユーザ視点の制御 (Navigation) の二つがある。三次元の場合、 Manipulation ではオブジェクトの位置として絶対座標における x, y, z の 3 自由度と、オブジェクトの姿勢として絶対座標との傾きを指定する 3 自由度の計 6 自由度を二次元マウスで操作しなければならない。また、Navigation では最低でもカメラ位置として 3 自由度、参照している位置として 3 自由度の計 6 自由度を操作しなければならない。このように自由度が多数になるため、三次元 VP では二次元にも増してプログラムとのインタラクション技術の工夫が必要である。姿勢の制御にスライダなどの GUI 部品を用いる方法 [22] や、三次元空間の壁に正面図、側面図、立面図に相当する影を表示し影をマウスで操作する方法 [23] などがある。すべての自由度に対しての操作を許すと操作方法は複雑になり、かえって混乱する。暦本らは、オブジェクトの回転を z 軸を軸とした回転に制限するといったように一部の自由度を制限しても十分必要な情報は得られると述べている [24]。自由度を制限すれば二次元デバイスであるマウスでも操作方法が簡易になるであろう。

## 3.2 巨大な情報量の表示技術

一般にビジュアルプログラムは、図形を扱うためにテキストベースのプログラムに比べて量がかさばる傾向にある。特に三次元 VP は二次元 VP に比べ一画面上に表示される情報量が膨大であり、またそうでなければ三次元化した意味がない。現在の高々 20 インチ程度のディスプレイにすべての情報を二次元表示することは困難であり、 Small Screen Problem[12] と呼ばれる。巨大な情報をユーザの要求に合わせていかに適切に表示するかが問題となる。情報は巨大になればなる程複雑になるが、複雑なまま表示してはユーザがそこから有用な情報を得ることができない。ここではその対策手法として、情報の構造を明確に美しく見せる「グラフの自動レイアウト」と注目していない情報を縮小したり隠蔽してグラフを簡素化する「非線形ズーミング」について述べる。

### 3.2.1 自動レイアウト機能

一般に情報をグラフで表現することは理解や記憶を容易にする。しかし、グラフが有用であるか否かはレイアウト (ノードの配置やエッジの配線) の結果に強く依存し、数多くの研究がなされている [25]。良くレイアウトされたグラフはグラフの意味を素早く明確に伝えることができるが、良くないレイアウトは混乱や誤ちを招く可能性が高くする。グラフは巨大になればなる程複雑になるため、グラフの適切なレイアウトが重要となる。一般にノードの重複がないこと、無駄なスペースがないこと、エッジの交差が最小であることなどの制約を満たすレイアウトが望ましい。また、このような基準のみではなく、重要度や注目度の高いノードの周辺に故意にスペースを作る、関係の強いノード同士は近接させるといったレイアウト対象のグラフの構造に適した配置をするなど理解支援の研究もある [26]。

### 3.2.2 非線形ズーミング機能

三末らはグラフ表示手法への 5 つの要求として「詳細性」 (注目部分を詳細に表示しているか)、「全体性」 (情報の全体の概要を表示しているか)、「同時性」 (詳細性と全体性を同時に満たしているか)、「表示像の単一性」 (表示されている像は単一であるか)、「写像の適正」 (グラフの特性に適正な写像であるか) を挙げている [27]。5 つ要求全てを満たすことが望ましいが、一般的な画面スクロールやマルチウィンドウ方式では全てを満たせない。グラフ全体を線形に拡大する線形ズーミング手法を用いたものに Pad++[28] があるが、「同時性」を満たさない。そこで要求全てを満たす

ものとして注目されているのが非線形ズーミングである。詳細に表示したい領域だけをズーミングし、それ以外の領域を縮小するのでこう呼ばれる。情報の全体の概要といくつかの部分の詳細を分離せずに滑らかにつないで一画面に表示することができ、数多くの手法が提案されている [29]。注目していない部分は縮小されるだけでなく、情報の全体の構造が把握できる程度に抽象化されたり隠蔽される手法も提案されており、グラフの簡素化にも貢献している。

Leung らの非線形ズーミングの調査 [29] では、現在のほとんどの研究は Polyfocal Projection[30], Bifocal Display[31], Fisheye View[14] の子孫であると述べられている。非線形ズーミングは transformation function によってグラフをどう配置するかを決定し、 magnification function によってグラフの各領域がどの程度拡大 (縮小) されるかを決定する。 transformation function に自動レイアウトアルゴリズムを連携させる研究 [32, 33] や、ユーザの意図を予測することによって非線形ズーミングにおけるスムーズなインタラクションを提供する研究 (Hyper Mochi Sheet[34]) も発表されている。

## 3.3 アニメーション技術

アニメーション技術もまた、ユーザが情報をより早くより深く理解するために用いられ、主に以下のものがある。

- アルゴリズムアニメーション：一般に、プログラミング初心者にとって計算機アルゴリズムを理解するのは難しい。図を用いて読者の理解を助けようとしている教材があるが、静的な図ではアルゴリズムのダイナミックな動きを表現しきれない。これに対しアルゴリズムアニメーションは、各種アルゴリズムを図的に、かつアニメーションを利用して表示する手法である。例えばバブルソートプログラムでは，各データはその値に対応する高さを持った柱として表示され、プログラムの実行とともにデータ同士が交換されていく様子がダイナミックに表示される。ただし、提供される図やアニメーションは、視覚化の対象となるアルゴリズムに完全に依存するため汎用性が低い。異なるアルゴリズムには新たに図やアニメーションをデザインする必要がある。そのため初心者教育用や特定のアルゴリズム研究者用と用途が限られてしまう。

- プログラム実行のアニメーション：ビジュアルプログラムの実行をアニメーションで表現する。プログラムの実行を図の動的な変遷によって観察することがで

きるため、プログラムのふるまいやデータの流れを直観的に理解することができる。アルゴリズムアニメーションと異なり、その VP システムで作成されたビジュアルプログラムは全てアニメーションで表現できる。アルゴリズム毎に図やアニメーションをデザインする必要がない。

- グラフ変化のアニメーション： VP システムで作成されたグラフは、ユーザの操作を反映して自動レイアウト機能や非線形ズーミング機能などによって変形される。現在提示されているグラフから新しいグラフへ急激に変化すると、ユーザは両者の整合性をとるために認知的負荷を背負うことになる。適度な速度によるアニメーションはユーザの認知的負荷を軽減することができる。

# 第 4 章

# Bubble-Gum：二次元におけるレイアウトとズーミング

大規模なグラフを扱う VP 環境で必要となる要素技術として、非線形ズーミング機能、自動レイアウト機能とアニメーション表示を第 3章で挙げた。我々は、これらの技術がユーザの認知的負荷の軽減に貢献する点で注目している。**“Bubble-Gum”**は、二次元グラフを対象にしてこれらの技術について研究した成果であり、インタラクティブ性の向上と自然なアニメーションの両方を実現したシステムである [32]。

- インタラクティブ性：
  従来のアニメーション表示は新しいグラフを計算でもとめた後に計算前と計算後の差分を「コマ割り」して順次表示することで実現されてきた。そのためアニメーション前とアニメーション後の状態しか扱えず、全ての処理が終了するまでユーザの操作に答えることができない。例えばアニメーション中に「他のノードに関心が沸き焦点を移動したい」、「操作中にミスを犯したのでやり直したい」などの欲求が生じた時に、アニメーションが終了するまでは操作に答えることができない。しかも操作が反映される時には操作した時点からグラフが変化しており、意図した結果が得られないこともある。これではインタラクティブ性に欠ける。

- 自然なアニメーション：
  従来のシステムは非線形ズーミングと自動レイアウト機能が別々であったため、「コマ割り」の手法でしか自然なアニメーションを実現できなかった。ここでいう自然なアニメーションとはレイアウトやズーミングが施された最適グラフに向けて直接アニメーションすることである。従来は、例えば自動レイアウト機能により導出されたグラフに対して非線型ズーミングを適用し、目的のグラ

フを導出していた。そのため「コマ割り」の手法を用いずに計算の過程を表示すると、まず自動レイアウトの過程がアニメーション表示されて、次に非線形ズーミングの過程がアニメーション表示される。

我々は、インタラクティブ性の向上と自然なアニメーションの両方の実現に向け以下の方法を採用した。インタラクティブ性の向上には、最適グラフになるように微小変化を反復して計算する反復法を用いた。各ステップの微小変化を画面に反映することで、アニメーション表示をユーザに提示する。各ステップの計算コストは低いので即座にフィードバックを返すことができる。また各ステップでの状態を表す内部表現が存在するので、それに対してユーザの操作を施すことで即座に対応することができ、その内部表現に対して反復計算を再開することでアニメーションを続行する。自然なアニメーションの実現には両機能を融合した。両機能の要求を満たす微小変化の反復計算を可能にし、最適グラフまで自然にアニメーションをさせることができた。

## 4.1 自動レイアウト機能と非線形ズーミングの融合

我々は、自動レイアウト機能に非線形ズーミング機能を融合した新しいアルゴリズム**「多視点遠近スプリング・モデル」**を考案した。本アルゴリズムは無向グラフのレイアウトに有効なスプリング・モデル [35] をベースとしている。これは、各ノードの間に引力あるいは斥力を定義した力学系を与え、その力学系の釣り合いを反復法で求めることによって最適グラフを求めるものである。したがって我々の目指す高インタラクティブ性を満たすには、反復法を用いるスプリング・モデルは相性が良い。スプリング・モデルにおいてノードが受ける力には、エッジをバネに想定して計算される力と、エッジでつながれていないノード同士で働く力とがある。これらの力を本論文ではエッジ力とノード力と呼ぶことにする。エッジ力、ノード力は下式 (4.1, 4.2) で導出される。ただし $C_s, C_r$ は引力と斥力のバランスをとる定数である。また $d$ はノード間の実際の距離を表わし、 $d_0$ はバネの自然長 (エッジの理想長) を表わす。図 4.1にスプリング・モデルを示す。

$$f_s = C_s log \frac{d}{d_0} \tag{4.1}$$

$$f_r = C_r \frac{1}{d^2} \tag{4.2}$$

我々はスプリング・モデルに非線形ズーミングを導入するにあたり、4 つのポイントに留意した。注目度、バネの自然長、画面引力、そして収縮力である。注目度は各

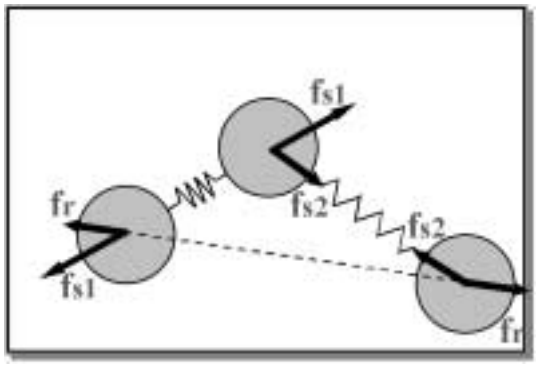


図 4.1: スプリング・モデル

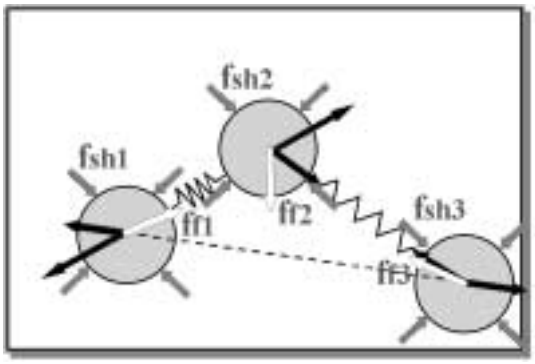


図 4.2: 多視点遠近スプリング・モデル

ノードに対するユーザの注目度を表し、ノードの大きさとして反映される。バネの自然長は、力学系が釣り合った時のノードの中心間の距離である。自然長の値はノードのサイズに応じて随時調整する必要がある。バネが固定長だと、ノードが大きい場合ノード同士が重なってしまうし、小さい場合は逆にノード間に隙間がたくさんできてしまう。画面引力は、画面の中心に向かって働く引力である。画面枠より大きなグラフの場合、枠に収まるようにこの力が働く。収縮力は、ノードを萎ませる力である。ノードのサイズを小さくして画面引力によって密集したノード同士の重なりを回避する。図 4.2に多視点遠近スプリング・モデルを示す。このアルゴリズムでは、エッジ力 $f_s$ とノード力 $f_r$ の他に、画面引力 $f_f$ と収縮力 $f_{sh}$ が働く。画面引力、収縮力は式 (4.3, 4.4) で表わされる。ただし、$C_{f1}$ は画面中心から距離に応じた引力の増減を表わす定数である。$d$ は中心からノードまでの距離であり、$d_f$ は中心から画面枠までの距離である。$C_{f2}$ は画面引力の大きさをコントロールする。$C_{sh}$ は画面引力と収縮力とのバランスをとる定数である。

$$f_f = -\frac{1}{C_{f1}(d - d_f)} + C_{f2} \tag{4.3}$$

$$f_{sh} = C_{sh} f_f \tag{4.4}$$

図 4.3, 4.4に図 4.1, 4.2の力関係が安定し、レイアウトが終了した状態を示した。多視点遠近スプリング・モデルでは、画面引力のおかげで画面中央に画面枠に収まるようにノードが配置される。また、収縮力のおかげでノードサイズが調整される。

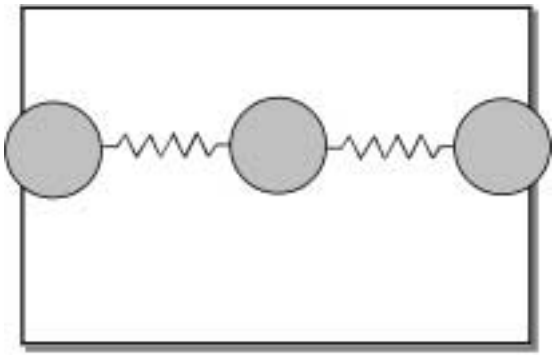

図 4.3: スプリング・モデル (レイアウト後)

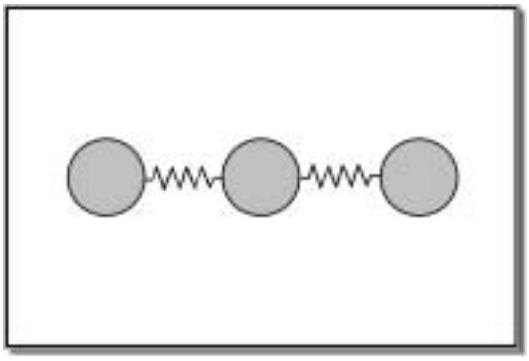

図 4.4: 多視点遠近スプリング・モデル (レイアウト後)

## 4.2 “Bubble-Gum” の実装

我々は多視点遠近スプリング・モデルを **“Bubble-Gum”** と命名して VP システム “viewPP” [9](図 4.5) に実装した。 “Bubble-Gum” のアルゴリズムを図 4.6に示す。このアルゴリズムを前述の 4 つのポイントに沿って説明する。

**注目度**　各ノードには「通常の大きさ」「注目時の大きさ」を内部表現に持たせる。ノードには、注目度に関して二つの状態がある。一つは注目時の状態で、「注目時の大きさ」がノードの理想サイズとなる。ユーザが注目度を変更した場合、このパラメータを変更することで対応する。もう一つは非注目時の状態で、「通常の大きさ」がノードの理想サイズとなる。ノードは現在のサイズから、理想サイズになるように微少変化をしていく。図 4.6の (1) では、注目度に関する処理：ノードの理想サイズの決定がなされる。

**バネの自然長**　バネの自然長は、両ノードの理想サイズを包含する円の直径の和に対する比で計算される。初期設定では、比を 100% としている。すなわち全く同じ大きさのノード同士の場合、ノードの間にもう 1 つ同じ大きさのノードが入る距離となる。図 4.6の (2) では、ノードの理想サイズに応じて比を保つようにバネの自然長が計算される。ちなみにこの間隔はユーザが指定することもできる。

**画面引力**　画面引力 $f_f$ の大きさは、グラフ全体の大きさに応じて柔軟に変更される。大きなグラフでは引力値を大きくしないと画面枠からはみ出てしまう。小さなグラフに対して大きな引力を与えると画面の中央にグラフが集まって画面を有効に使えない。

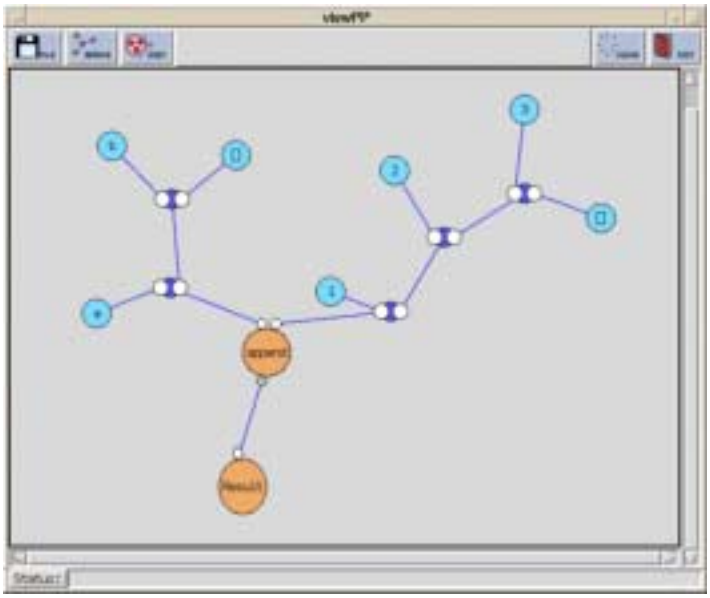

図 4.5: viewPP の画面


図 4.6: Bubble-Gum のアルゴリズム

適切な引力値の計算は以下のように行う。画面の縁付近に red ゾーン、その内側を環状に blue ゾーンを設定する。 red ゾーンにノードが存在する場合は、画面引力を強くする。 red ゾーンにノードがなく、 blue ゾーンにノードが存在する場合は、画面引力を変更しない。また、 red ゾーンにも blue ゾーンにもノードが存在しない場合は、画面引力を弱くする。画面引力の強弱は式 4.3の $C_{f2}$ でコントロールする。図 4.6の (3) ではグラフの全体サイズに応じて画面引力が計算される。そして図 4.6(4) でノードの位置計算に関わる合力の計算に (3) での画面引力が使用される。この合力は、画面引力の他にエッジ力、ノード力をかけ合わせた合力である。

**収縮力**　収縮力 $f_{sh}$ は、ノードの位置に対して働くノード力、エッジ力、画面引力とは異なり、ノードの大きさに対して働く。画面引力の大きさに比例して、ノードの大きさをコントロールする。画面引力が大きいということは、グラフ全体が大きいということであり、ノードを萎ませる度合と比例するからである。図 4.6の (5) で収縮力を計算し、理想サイズに収縮力を掛けてノードサイズを求める。

**実装上の工夫**　スプリング・モデルでは、エッジ力 $f_s$ についてはエッジにつながれたノードから受ける力を計算するが、ノード力 $f_r$ については総当たりで全ノードから受ける力を計算するため、計算量が膨大になる。本アルゴリズムでは、エッジ力 $f_s$ につ

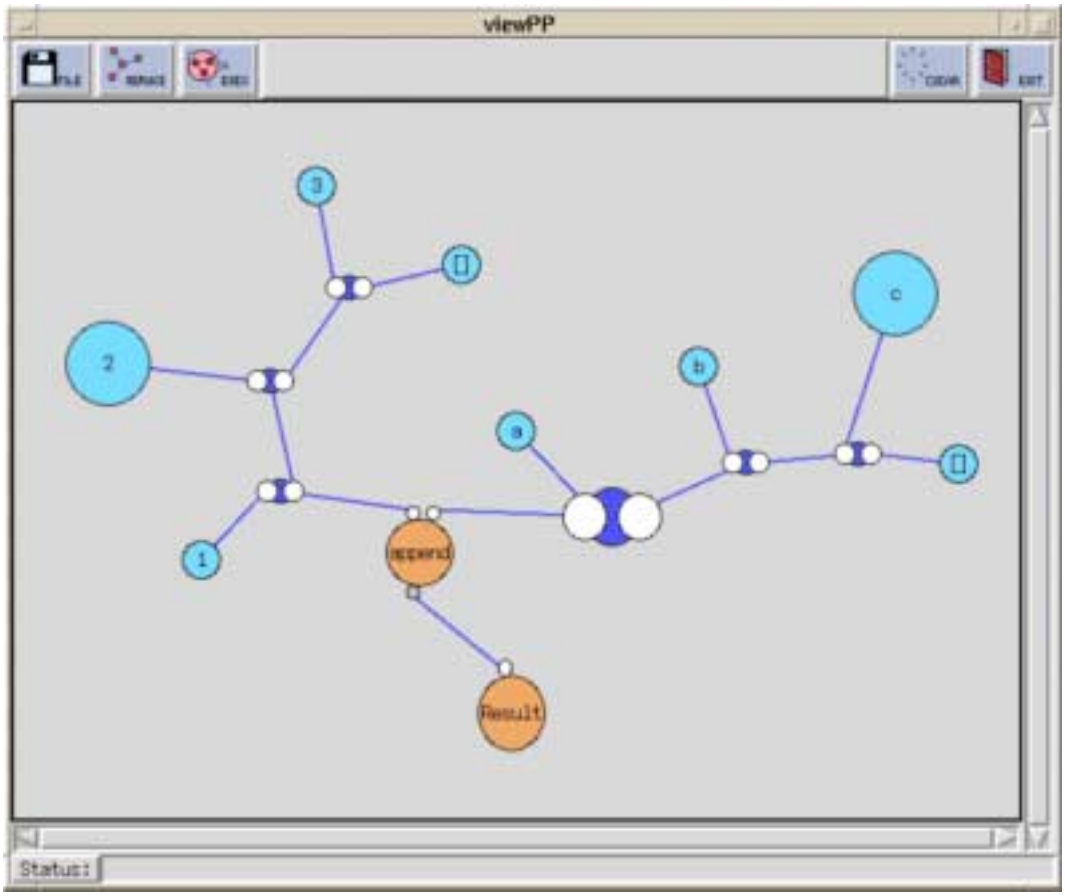


図 4.7: Bubble-Gum の実行画面

いてはスプリング・モデルと同様であるが、ノード力 $f_r$ については周辺の決められた範囲内のノードに対してのみ計算することで、計算コストを減少させている。図 4.6の (4) にこの工夫が適用されている。

また、反復計算で求められる微少変化においてノードに加わえられる力が大きくなると、1 ステップで移動する距離が大きくなり過ぎて滑らかなアニメーションにならない。本アルゴリズムでは、移動距離が閾値以上にならないように、移動距離を制限してアニメーションを滑らかに表現した。最も移動距離の大きなノードの移動距離が閾値以上の場合、移動距離を何%小さくすれば閾値以内になるのかを計算し、全ノードの移動距離に適用する。図 4.6の (6) でグラフの微少変化をアニメーションで表示する。移動距離の制限によるアニメーションのスムーズ化はここで行われる。

**反復計算**　本アルゴリズムは各ノードが安定状態になるまで (3) 〜 (6) を繰り返す。安定状態か否かは各ノードの受けた合力のうちの最大値が閾値を越えるか否かで決定

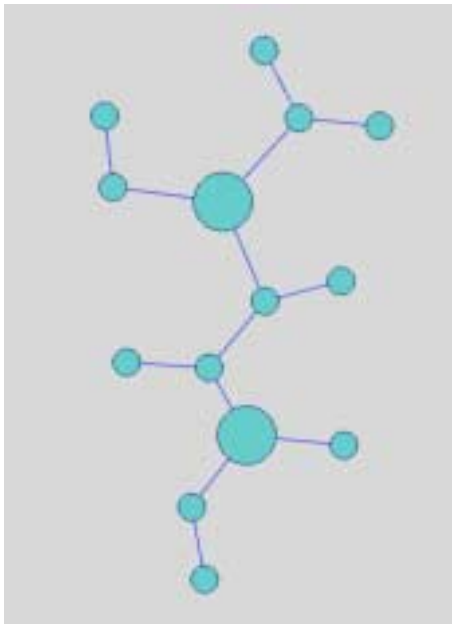

図 4.8: viewPP の画面

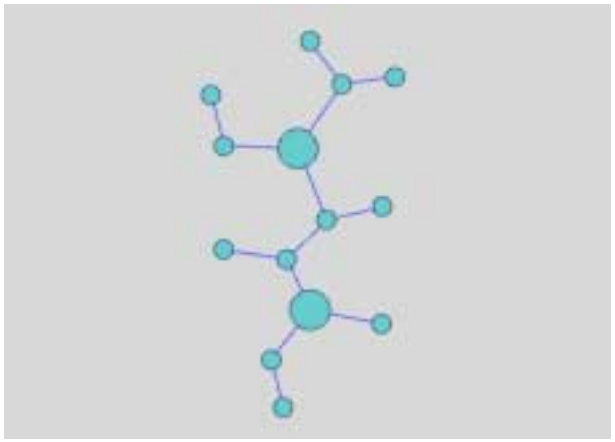

図 4.9: Bubble-Gum のアルゴリズム

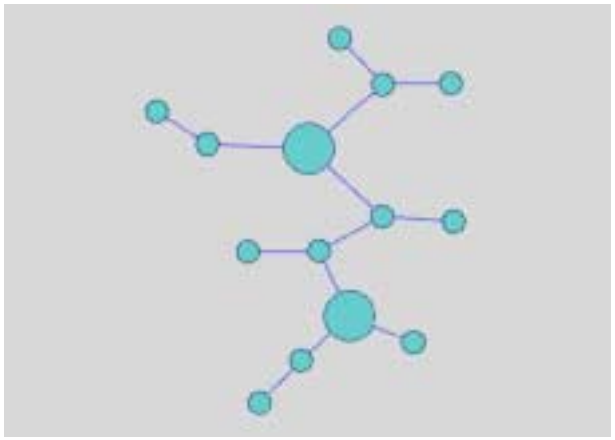

図 4.10: Bubble-Gum のアルゴリズム

される。“viewPP” において、“Bubble-Gum” を使用してレイアウトを行ったときのスナップショットを図 4.7に示す。この図は、アニメーション中の状態である。図 4.5と比較すると、ノードサイズが注目度に応じて拡大され、ノード位置は画面枠に収まるように配置されていることが分かる。

## 4.3 議論

画面引力と収縮力はグラフを画面の中央に画面に収まるように配置することを可能にした。この技術に対して、画面枠の中央にグラフ全体を移動させ、画面枠の大きさに収まるように一様に縮小すれば十分ではないかという指摘がある。ノード力、エッジ力の均衡を求める反復計算のステップ毎に画面枠に合わせる処理をすればアニメー

ションも可能である。この方法は画面枠の縦横比に近いグラフならば十分である。しかし図 4.8のように画面枠の形状との差が著しい図には適さない。図 4.9に示すように画面の領域を有効に使いきれないからである。その点 Bubble-Gum では全ての制御を力学系の均衡で求めるので、画面枠の形状に合わせた配置が可能である (図 4.10)。

## 4.4 Bubble-Gum に関連する研究

非線形ズーミングには、 Generalized fisheye views[14], The Continuous Zoom[38], 図的思考支援を目的とした図の多視点遠近画法 [27], Mochi sheet[39] などがある。これらは非線形ズーミングと自動レイアウト機能の融合におけるインタラクティブ性の向上について言及していない。また採用されているレイアウト・アルゴリズムが我々と異なっている。

スプリング・モデルに改良を加えた研究としてマグネティック・スプリング [40] と辺の長さに基く非線形ズーミング [33] がある。マグネティック・スプリングはエッジに対して磁力を与え、エッジが N 極を指すようにノードに力を加える。 3D-PP では論理変数を通してデータが流れるので、その流れる方向にエッジ (論理変数) の向きをそろえるとグラフが更に見易くなると予想される。この手法も力指向のアプローチであるため本研究との相性がよい。辺の長さに基く非線形ズーミングは、エッジの長さを調節することで注目ノードに近い程スペースを作る。ただし、ノード同士の力の均衡だけで解をもとめるので、画面枠からはみ出ようとするノードに対して対処できない。これに対し Bubble-Gum は、画面引力のおかげで画面の中央に、且つ、枠に収まるようにグラフを配置することができる。

# 第 5 章

# 三次元ビジュアルプログラミング環境の構築

我々は第 3章で述べた要素技術を基に三次元 VP 環境の構築にあたった。グラフレイアウト、非線形ズーミング、アニメーションに関して、 Bubble-Gum を三次元に拡張することで対応した。拡張された Bubble-Gum を “**3D-Bubble-Gum**” と呼ぶことにする。ただし、 3D-PP は包含関係をもつ階層構造を有するため、それへの対応が必要となる。また、扱うグラフは二次元から三次元になると同時に複雑化したので、ユーザの認知的負荷が高まる。いかに必要な情報を損わずに簡素化して見せるか、インタラクションを単純な操作で実現するかにも着目した。

図 5.1は、 3D-Bubble-Gum の画面イメージである。最も外側の箱 (以降「外箱」と呼ぶ) はグラフ全体を包含し、グラフはこの箱に収まるように配置されたりサイズを調整される。

## 5.1 3D-Bubble-Gum: Bubble-Gum の三次元化

Bubble-Gum における研究は、三次元 VP においても効率的な情報提示やインタラクティブ性の向上の点で有効であると考える。三次元 VP においてもインタラクティブ性は高い方がいい。 Bubble-Gum はグラフレイアウトも非線形ズーミングも力学系だけで構成されているため、階層構造を持たないグラフであれば三次元への拡張が容易である。二次元空間上で力学系の安定状態を求めていたのを三次元空間上で求めればよい。

## 5.2 階層グラフへの対応

3D-PP におけるビジュアルプログラムは、階層構造を持ったグラフで表現される。そのため、階層構造を持たないグラフを対象にしていた Bubble-Gum を階層グラフに

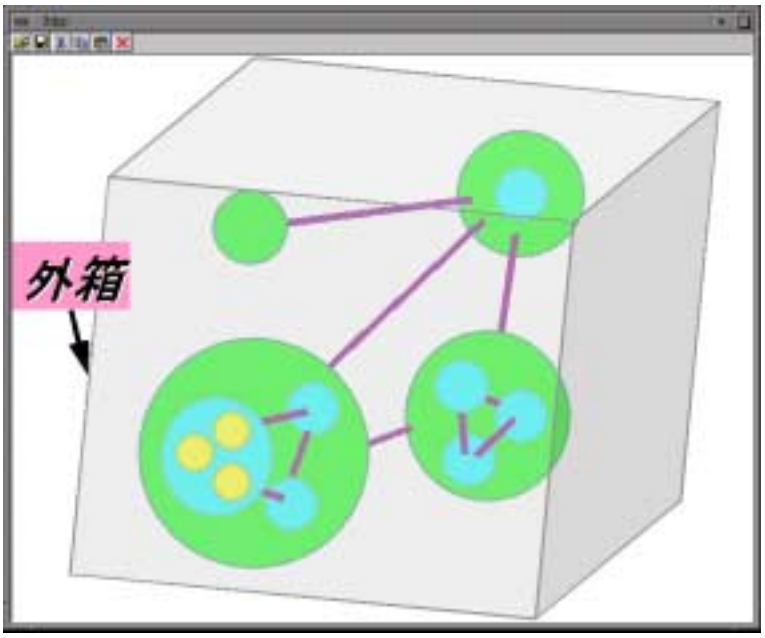


図 5.1: 3D-Bubble-Gum の画面イメージ

対応できるように拡張する方法を提案する。最上位の階層から深い層にいくに従って第 1 層, 第 2 層, …, 第 n 層と呼ぶことにすると、図 5.2に示すグラフでは第 1 層は外箱内の層でノード A, B, C が存在し、第 2 層は A, B 内の層でノード $a_1, a_2, a_3, b_1, b_2$ が存在するといった構造をしている。 Flowgraph Editor[36, 37] では、ノードのサイズを最深層 (第 n 層) から求めていき子ノードの合計サイズから親サイズを決定するといった形で求め、導かれたサイズのノードが重複しないようにレイアウトを行う手法を採用している。これに対し今回提案する方法は、力学系の釣り合いを求めることでノードの位置やサイズを求めるものである。

### 5.2.1 ノードの位置関係

Bubble-Gum では、ノードの位置をコントロールする力としてノード力、エッジ力、画面引力があった。階層グラフへの対応にあたり、ノードは画面枠のみに収まれば良いのではなく、それらを包含する親ノードが存在する場合は、親ノードからはみ出ないように配置する必要がある。この問題には、画面引力にあたる力を各親ノードの中にも働かせることで対応する。ただしこの力を親ノード内で働く時にも画面引力と呼ぶと整合性が悪いので、 3D-Bubble-Gum では**中心引力**と呼ぶことにする。

階層構造を持たないグラフでは、力学系の均衡を求める範囲がウィンドウ内に一つだけ存在し、その範囲内で釣り合いを求めれば良かった。これに対し階層グラフでは、

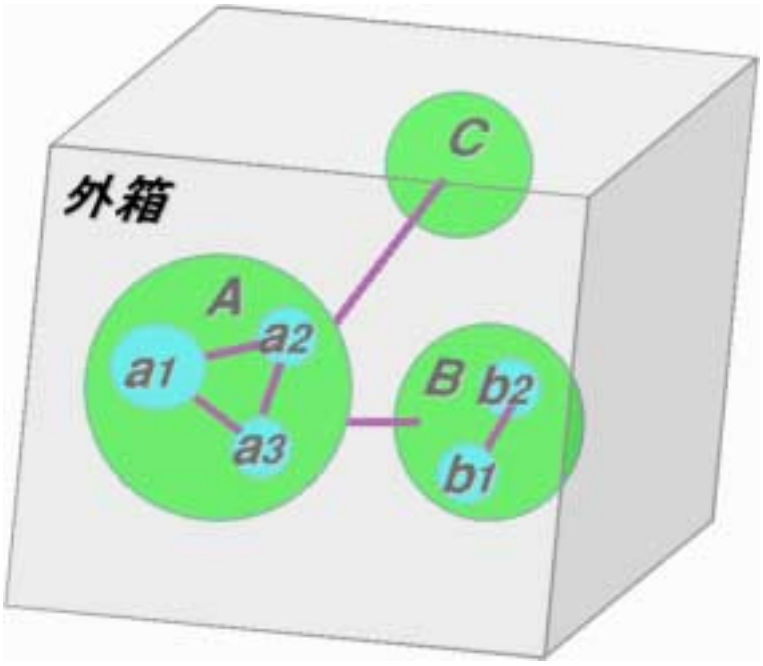


図 5.2: 三次元階層グラフ

親ノード毎に力学系の均衡を求める範囲を設定しその範囲毎に力学系の釣り合いを求める必要がある。図 5.2の例で説明すると、外箱内ではノード A, B, C の位置を、ノード A 内ではノード $a_1, a_2, a_3$ の位置をそれぞれの範囲で釣り合いを求めることで、全体のグラフのレイアウト結果を得ることができる。

### 5.2.2 ノードのサイズ

ノードのサイズはどのような制約条件を満たさなければならないであろうか。制約条件には以下のものが考えられる。

- 親となる物 (親ノードもしくは外箱) からはみ出ない大きさである
- 子ノード群を包含できる大きさである
- 外箱の大きさは画面内に収まり一定である
- 注目ノードは膨張し、非注目ノードは収縮する

これらの条件を満たすには親ノードサイズを柔軟に膨張・収縮をすることが求められる。親ノードは、子ノードが追加されれば大きくなり、削除されれば小さくなるといった具合に変化しないとはみ出したり、余分な空間ができるからである。また、非線形ズーミングによってノードは膨張・収縮するのにも対応しなければならない。

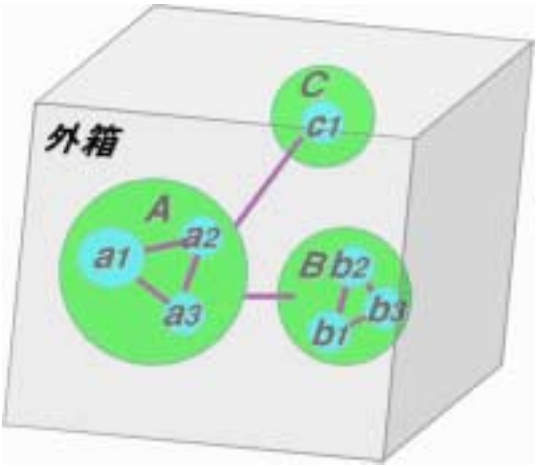


図 5.3: 気圧による制御 (前)

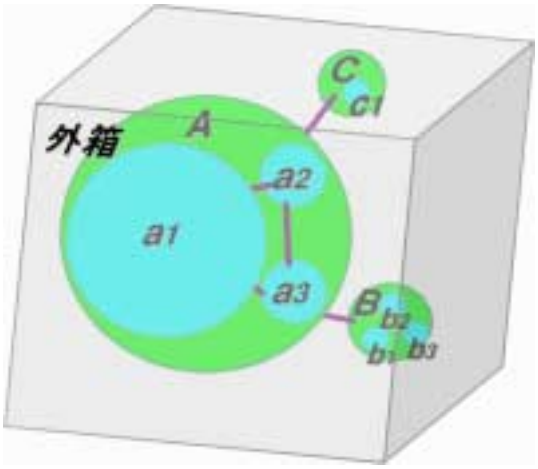


図 5.4: 気圧による制御 (後)

**気圧による制御**

我々はこのような条件を満たすような実世界の物体を探してみた。外箱を気体を満たして密閉した容器に見立て、各々のノードを伸縮自在な膜でできている風船としてその動きをイメージしたところ、十分に制約条件を満たすに足るものが想像された。密閉された容器 (外箱) 内に気体を注入すれば気圧が上昇して、風船 (ノード) は萎む。また風船に空気を注入すれば風船内の圧力が高まり風船が膨らむが、外の気圧によって容器からははみ出ない。風船の中にさらに風船を入れれば、階層グラフも扱える。例を図 5.3, 5.4を用いて説明する。図 5.3はノード a1 を膨らませる前の状態を、図 5.4は膨らませた後の状態を示している。第 2 層のノード a1 が膨らまされた場合、それを包含する第 1 層のノード A の中の空気圧が高まり、ノード a2, a3 は萎む。それに伴ないノード A は膨らむので、外箱内の気圧が高まり、ノード B, C が萎む。

絶対温度 $T$ の条件で分子のモル数 $n$ の気体を体積 $V$ の容器に閉じ込めると、気圧 $p$ は下式 5.1に示す気体の状態方程式で求められる。ここで比例定数 $R$ は気体定数である。式 5.1から気圧は分子数に比例し、容器の体積に反比例することが分かる。

$$p = \frac{RnT}{V} \tag{5.1}$$

これらから外箱やノードを容器として考え、各々の容器内の気圧の釣り合いでノードサイズを求めることにした。初期状態では、空の外箱、空のノードは気体を充満させ 1 気圧 $(atm)$ に保たれている。ノードを外箱の中に入れると、外箱内の気体の自由に動ける部分の体積はノードの体積分小さくなる。外箱の容積を $V_o$、中に入れられた

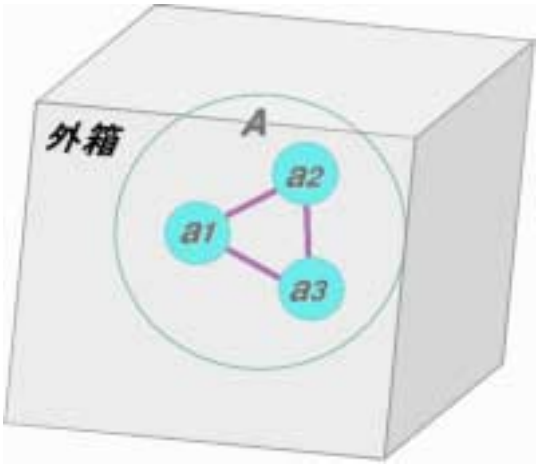


図 5.5: 子ノードのレイアウト結果 a

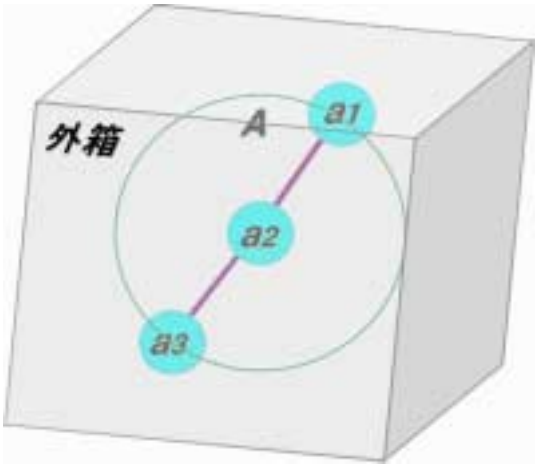


図 5.6: 子ノードのレイアウト結果 b

ノードの体積を $V_n$ とすると、外箱内の気圧を計算する際の容器の体積 $V$ は、

$$V = V_o - V_n$$

となる。よって $V$ が減少し外箱内の気圧は上昇する。また、外箱内の気圧が上昇すれば、ノードはその圧力を受けて萎むためノード内の圧力も上昇する。ノードのサイズは外箱内とノード内の気圧が釣り合った状態で安定する。

**子ノードのレイアウト結果が異なる場合**

親ノード (もしくは外箱) に対して子ノードが一つのみ存在する場合は上記の方法で十分である。ただし次に示すような例の場合、それだけでは親ノードのサイズを適切に制御できない。図 5.5はノード $a_1$ とノード $a_3$ の間にエッジが張られており、図 5.6は張られていない。エッジ一本の差でレイアウト結果にこのような差がでる。両図ともノード $a_1, a_2, a_3$ の体積 $V_1, V_2, V_3$ の合計は等しいため、ノード A 内の圧力 $p_A$ は、式 5.1に体積 $V = V_A - V_1 - V_2 - V_3$ を代入して計算され等しくなってしまう。しかし、レイアウトされたノード $a_1, a_2, a_3$ の包含するには図 5.6の方がノード A の体積を大きく (そのためにはノード A 内の気圧を高く) しなければならない。

我々はこの問題に対し、式 5.1における体積 $V$ を

$$V = V_p(\text{親ノードの体積}) - V_c(\text{子ノード群の体積の合計})$$

で求めるのではなく、

$$V = V_p(\text{親ノードの体積}) - V_i(\text{子ノード群を包含する球の体積})$$

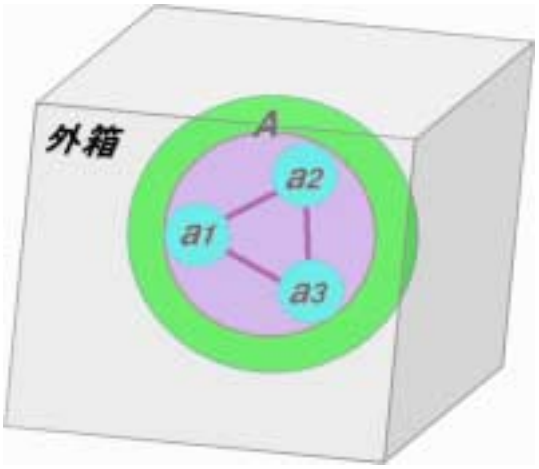


図 5.7: 包含球による気圧の制御 a

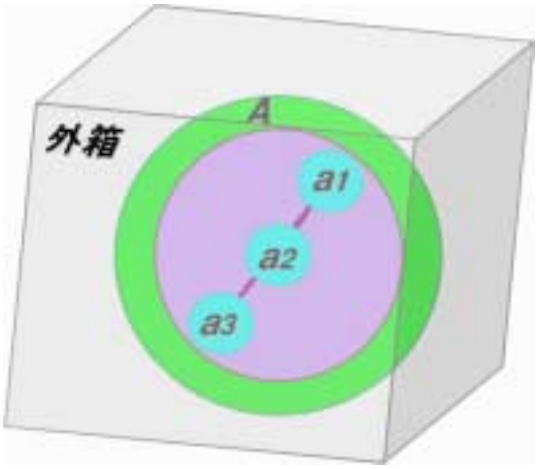


図 5.8: 包含球による気圧の制御 b

で求める。こうすれば、図 5.5, 5.6は、図 5.7, 5.8のように子ノード群を包含する球の体積を元に気圧を計算し、親ノードのサイズを適切に導くことができる。

### 気圧力の定義

以上のことを踏まえて、気体の状態方程式 5.1から気圧による圧力 $f_p$ を定義した (式 5.2)。

$$f_p = \frac{C_p n}{V} \qquad \left( \begin{array}{rcl} V & = & V_p - V_i \\ C_p & = & CRT \end{array} \right) \tag{5.2}$$

ただし、体積 $V$ は親ノードの体積 $V_p$ と子ノード群を包含する球の体積 $V_i$ から導かれる。比例定数 $C_p$ は気体定数 $R$ と絶対温度 $T$ に係数 $C$ を掛けたもので一定である。

### 中心引力の定義

中心引力は以下のように働く。親ノードの体積に対する子ノード群の占める体積が大きい場合、親ノードからはみ出ないように子ノードの位置を中心に寄せる。逆に親ノードの体積に対する子ノード群の占める体積がある程度小さい場合、子ノードを中心に寄せる力を緩める。親ノードの体積と子ノード群の占める体積は気圧力の強弱になって表れる。よって中心引力は気圧力の強さに応じて増減するように定義した (式 5.3)。気圧力 $f_p$ に定数 $C_{fp}$ を掛けた $C_{fp} f_p$ によって強さをコントロールされる。

$$f_c = -\frac{1}{C_{f1}(d - d_f)} + C_{fp} f_p \tag{5.3}$$

| Bubble-Gum | | 3D-Bubble-Gum | |
|---|---|---|---|
| エッジ力 | $f_s = C_s log\frac{d}{d_0}$ (4.1) | エッジ力 | 左に同じ |
| ノード力 | $f_r = C_r\frac{1}{d^2}$ (4.2) | ノード力 | 左に同じ |
| 画面引力 | $f_f = -\frac{1}{C_{f1}(d-d_f)} + C_{f2}$ (4.3) | 中心引力 | $f_c = -\frac{1}{C_{f1}(d-d_f)} + C_{fp}f_p$ (5.3) |
| 収縮力 | $f_{sh} = C_{sh}f_f$ (4.4) | 気圧力 | $f_p = \frac{C_p n}{V}$ (5.2) |

表 5.1: Bubble-Gum と 3D-Bubble-Gum の力

### 5.2.3 Bubble-Gum との力の比較

Bubble-Gum と 3D-Bubble-Gum で定義された力を表 5.1にまとめた。エッジ力、ノード力に関しては両手法とも同一の定義式で求められる。 Bubble-Gum における画面引力は 3D-Bubble-Gum では中心引力と呼ばれ、気圧力の強さで制御される。 Bubble-Gum では収縮力でノードのサイズを求めるが、 3D-Bubble-Gum では気圧力で求める。

## 5.3 グラフの簡素化

3D-PP におけるグラフ構造の特性を考慮して、グラフの簡素化に取り組まねばならない。我々は特に「階層構造」「半透明表示」に着目して簡素化を行った。

### 5.3.1 中間層の省略

深い階層を持つグラフを表示する場合、ある程度深い層のノードを見るためにそのノードを拡大するとそれを包含している祖先にあたるノードを通して見ることになる。その結果、幾重にも包含している祖先のノード群によって次のような問題が生じる。

- 注目ノードの周辺が密集して複雑になる
- 注目ノードを十分拡大できない
- 祖先ノード群を通して見るため、透明度が低くなり注目ノードをはっきり見れない

ここで我々は、注目ノードの属する階層を第 i 層とすると、第 2 層から第 (i-1) 層までを省略する手法を提案する。図 5.9, 5.10は、第 1 層と注目層の中間の層を表示したものと省略したものである。省略された中間層は一つの層にまとめられる。省略され

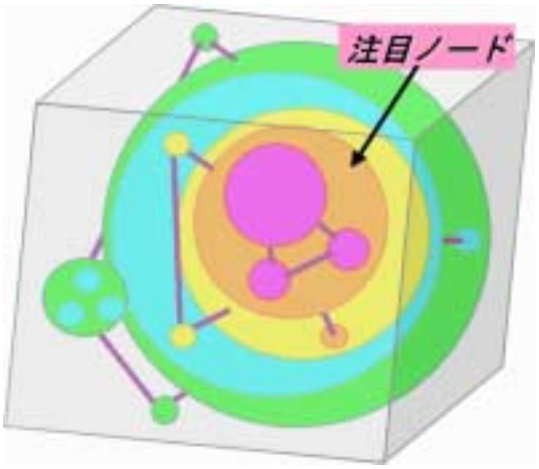


図 5.9: 中間層の省略 (前)

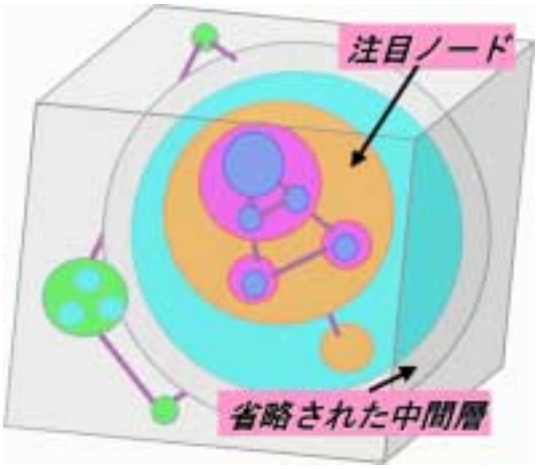


図 5.10: 中間層の省略 (後)

た層があることを示すために、まとめられた中間層は薄い霧で満たしたように描画する。中間層が省略されることによって、周辺はすっきり簡素化される。祖先ノードが占めていた領域は削減され、注目部分は十分拡大して詳細に見ることができる。そして注目ノードを見るために透かさなければならない層が減り、透明度が上昇するため、注目層をはっきり見ることができる。

### 5.3.2 半透明表示による簡素化

三次元空間で階層グラフを表示するには半透明表示が有効である [24]。三次元オブジェクトを Wire Frames で表現すると全ての情報は透けてしまうので、表示される図が複雑になってしまう。逆に不透明にすると中の情報が見えなくなる。半透明にすれば、両方の問題を解決することができる。第 1 層から第 n 層へ層が深くなるにつれて透明度を低くすることで、画面上に表示する情報量を削減できる。

これに対して注目ノードを拡大して中間層が省略される場合、省略しない場合と比較して注目層を高い透明度で表示できる。そのため、注目層よりも深い層の情報をより多く提供できる。

これらのことにより、注目ノードを拡大している場合でも、何も拡大していない場合でも表示する情報量をある程度一定に保ちながらユーザに提供することができる。

### 5.3.3 その他の簡素化

その他、グラフを簡素化する手法として、再帰的に繰り返し現れるノード (リカーシブゴール) の省略、数列のような同種のゴールの省略がある。これらの情報は規則

性があり、その情報の全容を表示するよりも規則性が分かるように表示したほうが理解しやすい。例えば自然数列ならば、 $1, 2, \cdots, n$ と表示したほうが解りやすい。また、3D-Bubble-Gum によって収縮したノードや、ある程度透明度が低くく中身を判別できないようなノードは、抽象化したオブジェクトに置き換えることによってすっきりしたグラフになると考えられる。

## 5.4 インタラクション技術

### 5.4.1 Manipulation：オブジェクトの位置・姿勢に対する操作

二次元マウスで三次元オブジェクトを操作するには、様々な工夫が必要である (第3章参照)。 Manipulation に必要な計 6 自由度の全てをユーザに操作させれば、細かい制御が可能な反面、操作が複雑になる。そこで、ユーザが操作できる自由度を、必要な情報を得るのに十分な程度に制限する。オブジェクトの移動では、ユーザの視線に垂直な平面上における二次元移動に限定する。奥行き方向の移動や微調整は自動レイアウト機能に任せるので、ユーザの操作はその程度で十分である。階層グラフでは、ノードの親子関係を築くのに子となるノードを親となるノードの中に入れなければならない。ノードの移動操作を二次元移動に制限しているので、この操作をドラッグ & ドロップ手法で実現する。子となるノードをドラッグし親となるノードに奥行き方向に重なる位置にドロップする。ノードの移動中は、移動ノードの透明度を最高位に上げることで移動ノードの影になってしまうノードもユーザに見えるようにし、ドロップ対象となるノードを指定しやすくする。姿勢に関してはオブジェクトの z 軸を軸とした回転に限定する。視線の移動ができるので、オブジェクトの全容は z 軸回転のみで十分把握できる。

### 5.4.2 Navigation：ユーザの視点の制御

視点位置を自由に操作することは、三次元空間を浮遊する感覚が得られ、三次元グラフィクスの魅力の一つになっている。しかし、三次元 VP 環境においてグラフの中に入っていってグラフの中を渡り歩くといった操作は向かない。グラフ全体の概要を捉えにくくなり、三次元空間で迷子になる可能性が高いからである (第 3章参照)。では三次元 VP 環境ではどのような視点の制御が適しているのであろうか。非線形ズーミングが可能な環境では、グラフ全体を周りから見回すだけで全体から詳細までの情報を得ることが可能である。 3D-Bubble-Gum では、グラフ全体は外箱の中に収容されているので、外箱を回転させることで視点の移動を代用できる。カメラの位置は固

定され、カメラと外箱との距離も固定する。これにより最低でも 6 自由度必要だった視点の制御を外箱の姿勢 3 自由度のみに限定でき、マウスのドラッグ方向とドラッグ距離だけで制御できる。

### 5.4.3　非線形ズーミングに関する操作

非線形ズーミングに関する操作には主に以下のものがある。

- 注目しているノードをユーザにより手動で徐々に膨張 (収縮)
- 注目したいノードを前回注目した時のサイズまで自動で膨張
- 注目されているノードを非注目時のサイズに自動で収縮

これらの操作を「手動膨張モード」、「自動膨張モード」などとマウスのモードを切り換えることで実現するのは繁雑である。またスライダ等の GUI を用いて操作するのは直観的ではなく解りにくい。操作対象となるノードを直接操作し、モードの切り換えなしでこれらの操作を実現したい。そこで  のようなアイコンをノードに装着し、これに対してのクリック動作でズーミングを実現する。この操作は風船に空気を注入している動作をイメージしている。アイコンは空気穴を想定している。△をクリックすると一定量の空気が注入されノードが膨張する。一回のクリック動作が空気入れで一回空気を注入する動作にあたる。手動で膨張させる場合は△を好みのサイズになるまで数回クリックする。逆に手動で収縮させる場合は▽を数回クリックする。また、自動で膨張させる場合は△を長くクリックすることでシステム側が「自動膨張」と判断して実行される。自動収縮は▽を長くクリックする。

## 5.5　3D-Bubble-Gum に関連する研究

Carpendale らは非線形ズーミングを二次元から三次元に拡張する研究を発表している [41]。二次元における非線形ズーミング手法を三次元に拡張して、非常に大規模な格子状のグラフを対象にそれへの適用を試みている。 3D-Bubble-Gum は深い階層を持つグラフを対象にしている点でこの研究と異なっている。

Information Cube[24] は、半透明を用いて三次元階層グラフを効率良くユーザに提示するシステムである。深い階層を有するグラフを、箱を入れ子状に包含していくことで表現した。この方法は深い階層構造を有するグラフの表示手法として向いていると述べている。また、半透明表示の有効性についても述べている。 3D-Bubble-Gum

は半透明表示を用いた階層グラフの描画手法に、力学系の釣り合いによるノードの位置やサイズの制御を導入した。このことで非線形ズーミングのメリットを三次元階層グラフでも享受できる。また Bubble-Gum における高いインタラクティブ性も受け継いでいる。

# 第 6 章

# 結論

我々は三次元グラフィクスによる VP 環境の構築の研究を進めており、試作システムとして 3D-PP を開発している。ビジュアルプログラムを表現する三次元グラフの構造や特性から、開発に必要な要素技術を挙げた。これらの技術から三次元 VP 環境の構築手法として、

1. 気圧によるノードサイズの制御
2. 中間層の省略
3. 半透明によるグラフの簡素化
4. 三次元グラフとのインタラクション手法

を提案した。 1. は複雑な階層グラフのレイアウトやズーミングを可能にする。 2., 3. はグラフを簡素化して適量の情報をユーザに提供する。 4. は複雑になりがちな三次元グラフの操作を自由度を制限することで簡易にした。

今後、実用的なプログラミング環境の構築を最終目標に研究を発展させていく予定である。それには、本論文の構築手法による三次元 VP システムの実装が課題となる。また、種々の大規模プログラムのプログラミングに使用し、表示手法やインタラクション手法の評価を行いたいと考えている。

# 謝辞

本研究を進めるにあたり終始丁寧に指導して下さいました田中二郎教授に心より感謝いたします。また筑波大学 電子・情報工学系 田中研究室のみなさんからはゼミ等を通して貴重なコメントを頂きました。特に 3D-PP の研究に共同で取り組んだ 3D-PP グループのメンバーである遠藤浩通さん、大芝崇さん、宮下貴史さん、神谷誠さんとは 3D-PP の研究・開発にあたり有益な議論をしました。 NEC C&C メディア研究所の古関義幸さん、富士ゼロックス 総合研究所の伊知地宏さんには、研究の進め方について貴重な助言を頂きました。さきがけ 21 の原田康徳さんは、力学系によるレイアウトとズーミングに関する議論につきあって下さいました。東京工業大学大学院情報理工学研究科の豊田正史さんは、参考文献の収集にあたって御協力下さいました。ここに感謝の意を表します。

# 参考文献


[1] 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.

[2] George Robertson, Jock D. Mackinlay, and Stuart K. Card: Cone Trees: Animated 3D Visualizations of Hierarchical Information, *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91)*, pp.189-194, 1991.

[3] J. D. Mackinlay, G. G. Robertson, and S. K. Card: The Perspective Wall: Detail and Context Smoothly Integrated, *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91)*, pp.173-179, 1991.

[4] 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, pp.265-272, 1993.

[5] 田中二郎: ビジュアル・プログラミング・システム PP, ビジュアルインタフェースの研究開発報告書, 財団法人 日本情報処理開発協会, March 1994.

[6] 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, pp.125-128, 1998.

[7] Margaret M. Burnett and Marla J. Baker: A Classification System for Visual Programming Languages, *Journal of Visual Languages and Computing* Vol.5 No.3 pp.287-300, 1994.

[8] 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.

[9] 南雲淳、田中二郎： "viewPP": グラフ構造とアニメーション表現に基づくプログラム実行の視覚化, 日本ソフトウェア科学会第 14 回大会論文集, pp. 17-20, 1997.

[10] Masashi Toyoda, Buntarou Shizuki, Shin Takahashi, Satoshi Matsuoka and Etsuya Shibayama: Supporting Design Patterns in a Visual Parallel Dataflow Programming Environment, *Proc. 1997 IEEE Symposium on Visual Languages*, 1997.

[11] P. T. Cox, F. R. Giles and T. Pietrzykowski：Prograph: A Step towards Liberating Programming from Textual Conditioning, *1989 IEEE Workshop on Visual Languages*, Rome, pp. 150-156, 1989.

[12] D. A. Henderson and S. K. Card: Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface, *ACM Transactions on Graphics*, Vol.5, No.3, pp.211-243, July 1986.

[13] Margaret Burnett, et al：Scaling Up Visual Programming Languages, *IEEE Computer*, Vol.28 No.3, pp.45–54, March 1995.

[14] George W. Furnas: Generalized fisheye views, *In Proceedings of ACM CHI'86 conference on Human Factors in Computing Systems*, pp. 16-23, Association for Computing Macinery, 1986.

[15] KLIC 講習会テキスト -KL1 言語編 -, 財団法人 新世代コンピュータ技術開発機構 作成, 財団法人 日本情報処理開発協会開発研究室 改訂, 1995.

[16] 財団法人 機械システム振興協会：並列システムのための視覚的インタフェースの構成に関する調査研究報告書, 財団法人 新世代コンピュータ技術開発機構, 1995.

[17] 高田 哲司, 小池 英樹：VisuaLinda: 並列言語 Linda のプログラムの実行状態の３次元視覚化, インタラクティブシステムとソフトウェア II: 日本ソフトウェア科学会 WISS'94, pp. 215-223, 1994.

[18] Marc-Alexander Najork：Programming in Three Dimentions, *Thesis for Ph.D. in Computer Science of the University of Illinois*, 1994.

[19] Ken Kahn: Seeing Systolic Computations in a Video Game World, *Proc. 1996 IEEE Symposium on Visual Languages*, pp.95-101, 1996.

[20] 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.

[21] Christian Geiger, Wolfgang Mueller, Waldemar Rosenbach: SAM - An Animated 3D Programming Languages, *Proc. 1998 IEEE Symposium on Visual Languages*, pp.228-235, 1998.

[22] Michael Chen, S.Joy Mountford, and Abigail Sellen: A Study in Interactive 3-D Rotation Using 2-D Control Devices, In *ACM SIGGRAPH Computer Graphics*, pp.121-129, 1988.

[23] K. P. Herndon, R. C. Zeleznik, D. C. Robbins, D. B. Conner, S. S. Snibbe, and van Dam: Interactive Shadows, *Proceedings of the ACM Symposium on User Interface Software and Technology*, pp.1-6, 1992.

[24] Jun Rekimoto and Mark Green: The Information Cube: Using Transparency in 3D Information Visualization, 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, pp.125-132, 1993.

[25] 杉山公造：グラフ自動描画法とその応用, 計測自動制御学会, pp.81-100, 1993.

[26] 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, pp.205-214, 1994.

[27] 三末和男、杉山公造：図的思考支援を目的とした図の多視点遠近画法について, 情報処理学会論文誌, Vol.32, No.8, pp.997-1005, 1991.

[28] Benjamin B. Bederson and James D. Hollan: Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics *Proceedings of the ACM Symposium on User Interface Software and Technology*, pp.17-26, 1994.

[29] Y. K. Leung and M. D. Apperley: A Review And Taxonomy of Distortion-Oriented Presentation Techniques, *ACM Transactions on Computer-Human Interaction*, vol.1, No.2, pp.126-160, 1994.

[30] Naftali Kadmon and Eli Shlomi: A polyfocal projection for statistical surfaces, *The Cartographic Journal*, 15(1), pp.36-41, June 1978.

[31] Robert Spence and Mark Apperley: Data-base navigation: An office environment for the professional, *Behaviour and Information Technology*, 1(1) pp.43-54, 1982.

[32] 宮城幸司, 田中二郎: Bubble-Gum： VPS のための多視点遠近画法と自動レイアウト機能の融合, インタラクティブシステムとソフトウェア VI, 日本ソフトウェア科学会 WISS'98, pp.187-192, 1998.

[33] 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, pp.41-48, 1998.

[34] 豊田正史、高橋伸、柴山悦哉: Hyper mochi sheet: 複数フォーカスズーミングエディタにおけるナビゲーションのためのフォーカス予測手法, インタラクティブシステムとソフトウェア V, 日本ソフトウェア科学会 WISS'97, pp.87-94, 1997.

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

[36] Koji Miyagi, Motoki Miura, Jiro Tanaka: A graph layout and a multi-focus perspective display in the flowgraph editor for the legal articles, Proceedings of the International Symposium on Future Software Technology, pp. 378-385, 1997.

[37] Koji Miyagi, Motoki Miura, Jiro Tanaka: Flowgraph Editor for Legal Articles, International Journal of Advanced Computational Intelligence, Vol.2 No.1, pp.34-42, 1998.

[38] Lyn Bartram, Albert Ho, John Dill, and Frank Henigman: The Continuous Zoom: A constrained Fisheye Technique for Viewing and Navigating Large Information Spaces, In *Proceedings of UIST '95*, pp. 207-215, November 1995.

[39] 豊田正史、志築文太郎、高橋伸、柴山悦哉： Mochi sheet: ズーミングとレイアウト編集機能の統合, インタラクション'97 論文集, pp. 79-86, 情報処理学会, February 1997.

[40] 三末和男 杉山公造：マグネティック・スプリング・モデルによるグラフ描画法について, 情報処理学会研究報告, ヒューマンインタフェース 55-3, pp.17-24, 1994.

[41] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia: Extending Distortion Viewing from 2D to 3D, *IEEE Computer Graphics and Applications*, Vol.17, No.4, pp.42-51, 1997.