アルゴリズム・アニメーション用ツールキットの構築

和田 裕二† 前田 宗則†‡

† (株) 富士通研究所国際情報社会科学研究所 ‡ 新情報処理開発機構

E-mail: † wada@iias.flab.fujitsu.co.jp

概要 アルゴリズムの動作を理解することを助けるために有効な手法の一つとして，アルゴリズム・アニメーション (可視化) という手法がある．この手法を用いる場合，アルゴリズムの動作を分かり易く表示することが重要である．しかし，アルゴリズムの動作を分かり易く表示するためには，表示のためのプログラムが複雑になる．これはアルゴリズム・アニメーションの持つ重要な問題点の一つである．

本研究では，アルゴリズム・アニメーションにおける高度な表示を支援するために，アルゴリズム・アニメーション用ツールキット **Gaea** を提案する．本稿では，**Gaea** の持つ機能のうち，画面上に表示されてる表示物の形状を変化させる変身機能と複数の表示物を画面上に同時に表示する並列描画機能について述べる．

A Proposal of Toolkit for the Algorithm Animation

Yuji Wada† Munenori Maeda†‡

† FUJITSU LABORATORIES, IIAS ‡ Real World Computing Partnership

Abstract Algorithm animation, or program visualization, has been proposed so far to make various kinds of algorithms more understandable. One of main issues of it is that animators, who describe program codes to visualize executions of algorithms, desire to describe the simpler program codes, while effective animations tend to require the more complicated them. Our approach to this issue is to make tools ,which allow animators to describe the simpler program codes even for constructing effective algorithm animations.

In this paper, we propose a tool kit for algorithm animations, called **Gaea** . **Gaea** provides high-level facilities such as metamorphosis and parallel display of objects. We also discuss about its implementation.

1 はじめに

一般に，アルゴリズムの理解・研究・開発・デバグなどを行うためには，そのアルゴリズムの振舞いを正確に把握するということが重要である．アルゴリズムの振舞いを理解することを助けるために有効な手法の一つとして，アルゴリズム・アニメーション(アルゴリズムの可視化)という手法がある．アルゴリズム・アニメーションとは，アルゴリズムが実行されている様子を動画(アニメーション)を用いて分かり易く画面上に表示することである．文章や図表などの静的な情報と比べて，アルゴリズム・アニメーションを用いる場合，処理装置によってアルゴリズムが実行されている様子をより視覚的に分かりやすく捕らえることができる．

アニメーションシステムに関しては，既に幾つかの研究が行なわれている[1, 2, 3, 4, 5]．これらの研究によって，アルゴリズムの実行の様子を分かりやすく表示することは一般には困難であるという点が指摘されている．

アルゴリズム・アニメーションを行なう場合には，いかに分かり易くアルゴリズムの実行の様子を表示することができるかということが鍵となる．これを実現するためには，かなり技巧的なアニメーションを行うことが必要である．そのため，アニメーションを行うためのコードが複雑になり，それを記述するプログラマ(アニメータ)に多大な労力を強いることになる．このことは，アニメーションの手法を用いる上での大きな問題点の一つであると考えられる．このような問題を解決するためには，アニメータが意図する表示を比較的簡単に実現できるための道具が必要である．

本研究では，画面上に表示される表示物を動画のように連続的に滑らかに移動させるといったような高度なアニメーションの実現を支援することを目指して，アルゴリズム・アニメーション用ツールキット **Gaea** を提案する．**Gaea** では，表示物全てを統一的に扱うための共通の枠組(インタフェース)を備えることにより，高度なアニメーションが支援できると考えている．現在，**Gaea** で提供する機能としては，表示物の形状の変化(変身機能)・複数の表示物の同時描画(並列描画機能)・個々の表示物による描画の部分的な巻戻し(ロールバック機能)などを考えている．本稿では，これらの機能のうち変身機能と並列描画機能について述べる．

2 アルゴリズム・アニメーション用ツールキット Gaea

本研究では，アニメータが意図した表示を自分自身で比較的簡単に実現することができるための道具として，アルゴリズム・アニメーション用ツールキット **Gaea** を提案する．アニメータは，このツールキットを用いて作成した表示物を画面上に配置することにより，自分が意図したアニメーションを作成する．つまり，アニメーションとして画面上に表示を行なう上でツールキットは基本的な構築部品となる．この部品の機能を高めることにより，高度なアニメーションをより容易に構築できると考えている．

Gaea を設計する上での問題点としては，

- 表示物としてどのような形状のものを選ぶのか？
- 表示物にどのような機能をもたせるのか？

などが考えられる．これらの点について，以下の3節・4節で述べる．

3 変身機能

この節では，**Gaea** の提供する変身機能について述べる．まず，変身機能を説明するための準備として，キャストについて述べる．次に，変身とはどのようなものなのかを定義する．最後に，変身機能を実現するために必要な描画情報の管理と履歴情報の管理について述べる．

3.1 キャスト

Gaea では，画面(描画領域)上に表示される全てのものを総称してキャストと呼ぶ．キャストは，画面上にその形状を表示するために必要な情報を属性として持つ．例えば，全てのキャストが持つ共通の属性としては形状・位置・大きさなどがある．更に，例えば形状が三角形のキャストの場合には，そのキャストは三頂点の位置・三辺の長さなどの形状に依存した固有の属性を持つ(図1)．このように，キャストは

- 全てのキャストに共通した属性
- 形状に依存した固有の属性

の二種類の属性を持つ．**Gaea** では，形状もキャストの持つ属性の一つであると考えて他の属性と同等に扱う

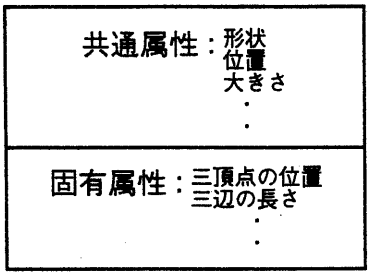

図 1: 三角形の持つ属性

ことにより，点・直線・円・多角形・文字列といった形状や属性の種類が異なる表示物をキャストとして統一的に扱っている．

現在 Gaea が提供するキャストの種類について，以下に簡単に述べる．

- 原始キャスト
  一つのキャストから構成され，内部に構造 (複数のキャスト間での位置関係など) を持たない表示物．
  - 幾何学的キャスト
    点・直線群・曲線群・円・多角形などの形状をした表示物．
  - 文字列キャスト
    アルファベット・数字などの文字列によってその形状が表現される表示物．
  - データキャスト
    ビットマップデータなどの何らかのフォーマットに従ったデータ列によってその形状が表現される表示物．
- 複合キャスト
  複数のキャスト (原始キャストや複合キャスト) によって構成され，内部に構造を持つ表示物．

アニメータは画面を構成するための部品としてこれらのキャストを用いることにより，意図する表示を比較的容易に実現することができると考えている．

3.2 変身とは

変身とは，ある形状をしたキャストをそれ以外の形状をしたキャストに動的に変形させることである．しかし，変身は形状だけを変更する単なる変形と異なり，そのキャストの振舞いをも変更する [6]．例えば，円（形状が円形のキャスト）が直線（形状が直線のキャスト）に変身する場合を考える（図 2）．円の場合，

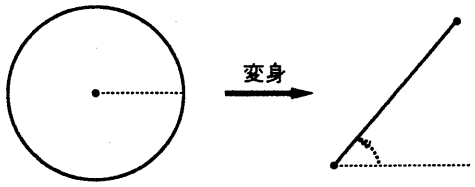

図 2: 円から直線への変身

- 半径・直径・中心点の位置など，円が持つ固有の属性を変更することができる．
- 円の中心点を中心に回転した場合，表示は変わらない．

といったような円という形状に依存した振舞いをする必要がある．一方，直線の場合，

- 始点の位置・終点の位置・傾きを変更することができる．
- 回転をすると，回転角に応じて表示が変わる．

という振舞いをする必要がある．そのため，キャストはその形状に応じて振舞いを変更するための枠組を持つ．

3.3 描画情報の管理

キャストは，自分の形状を画面上に表示するために必要な描画情報 (そのキャストの持つ属性) を管理している．つまり，キャストは自分に対して属性を変更する何らかの操作が行なわれた場合，自分が管理する描画情報を変更する．また，変身の際には，変身前の描画情報を変身後の描画情報にキャスト自身が自動的に変換する．例えば図 3に示すように四角形が文字列に変身する場合には，四角形を表示するために必要な描画情報 A を文字列を表示するために必要な描画情報 B にキャスト自身が自動的に変換する．この場合のキャスト自身とは，

変身前は四角形の形状をしていて，かつ変身後は文字列の形状をしているキャストのことである．描画情報の変

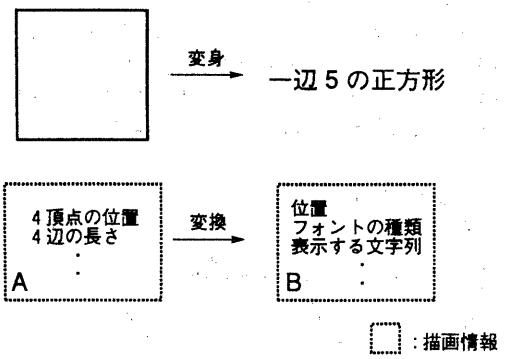

図 3: 描画情報の管理

換方法に関しては，基本的には何らかのデフォルトの規則 (変身戦略) を組込み，それに従って変換することを考えている．また，アニメータ自身が変身戦略を指定することを許すことにより，アニメータが自由に変身戦略をカスタマイズすることが可能となる．

3.4 履歴情報の管理

キャストは，自分に対して行なわれた操作の履歴情報も管理している．つまり，キャストは自分に対して何らかの操作が行なわれた場合，操作前の状態 (描画情報) と行われた操作を履歴情報として記録する．(但し，描画情報全てを記録するのではなく，その操作によって変更される描画情報だけを記録することにより，記録される履歴情報の量を減らす．) 履歴情報を用いて，キャストは以下のようなことを行なうことが可能である．

- 描画の再生・巻戻しを行なうことができる (図 4).

  履歴情報を最初の時点まで巻戻して表示を行なうことにより，描画の再生ができる．また，履歴情報を現在の時点から巻戻しながら表示を行なうことにより，描画の巻戻しができる．

- 変身前後での描画の一貫性を保持することができる (図 5).

  図 5に示されるように，円が文字列に変身する場合，キャストに対する操作の順序とキャストの形状

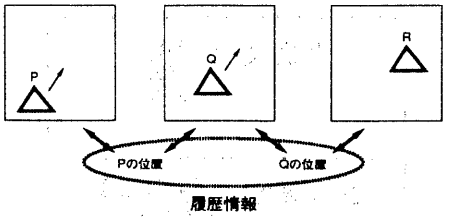

図 4: 描画の再生・巻戻し

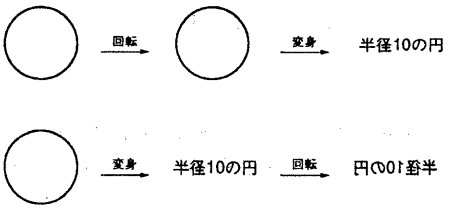

図 5: 描画の一貫性

によって描画結果が異なることが考えられる．図 5の上の場合，円を縦軸対称に 180 ° 回転させてから文字列に変身させている．この場合，円は縦軸対称に 180 ° 回転させても表示は変化しないため，単純に文字列に変身させてしまうと変身後の文字列は回転した文字列を表示しない．一方，図 5の下の場合，円を文字列に変身させてから縦軸対称に 180 ° 回転させている．この場合，文字列を縦軸対称に 180 ° 回転させると文字列が反転してしまう．このように，元は同じキャストであっても，変身の際に描画の一貫性を保つための仕組みが無ければ，操作の順序によって異なった表示が行なわれてしまう．変身の際に履歴情報を用いることによって，これを解決することも可能である．

また，キャストが管理する履歴情報に関しては以下のような操作を許すことにより，より柔軟な描画の再生・巻戻しを可能としている．

- 履歴情報のクリア

  その時点までに記録されている履歴情報をクリアする．これにより不要な履歴情報を減らすことができるが，その時点より前の時点における描画の再生・巻戻しは不可能となる．

- 履歴情報のオン/オフ

  履歴情報の記録を開始/中止する．これにより不要な履歴情報を減らすことができるが，履歴情報がオフの時点における描画の再生・巻戻しは不可能となる．

以上のように，キャスト自身が履歴情報を管理することにより，アニメータは描画の再生・巻戻しを容易に実現することができる．また，変身の際に履歴情報を用いてキャスト自身が自動的に描画の一貫性を保持することにより，アニメータに負担をかけること無く一貫性のある描画を得ることができる．

4 並列描画機能

この節では，Gaea の提供する並列描画機能について述べる．まず，並列描画とはどのようなものなのかを定義する．次に，並列描画機能を実現するために必要な行動戦略とその並列スケジューリングについて述べる．

4.1 並列描画とは

並列描画とは，複数のキャストを同時に画面上に表示することである．例えば，円・星形・文字列の 3 個のキャストの移動を並列描画する場合を考える (図 6)．こ

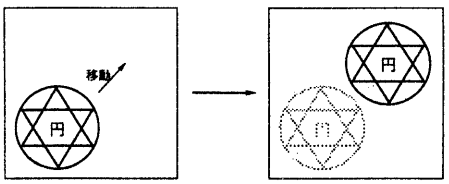

図 6: 並列描画

のような場合，通常は以下のような処理を繰り返すことになる．

(1) 円・星型・文字列を画面上から消去する．
(2) 円を少し移動させる．
(3) 星形を少し移動させる．
(4) 文字列を少し移動させる．
(5) 円・星型・文字列を画面上に表示する．

多数のキャストを同時に表示する場合には，このような記述は非常に煩雑であり，同じような操作を繰り返し記述することが頻繁に起きてしまう．このような点を解決するために，Gaea では並列描画機能を提供している．

並列描画機能を用いて複数のキャストを同時に表示するために，Gaea では複数のキャストをグループ化し複合キャストを作ることができる．複合キャストは，グループ化されているキャスト全てを管理する．つまり，複合キャストに行動戦略 (表示・消去・移動・点滅などの動作の指示) を与えることによって，グループ化されている全てのキャストに動作の指示を与えることができる．例えば図 6のような表示を並列描画機能を用いて行なう場合には，以下のように記述すれば良い．

(1) 円・星型・文字列を複合キャストとしてグループ化する．
(2) 複合キャストにグループの行動戦略を与える．
(3) 複合キャストに行動戦略の実行開始を指示する．

複合キャストが管理しているキャストに動作の指示を伝える際，複合キャストは管理しているキャスト全てを並列にスケジューリングする．つまり，図 6で述べたような繰り返しの処理を複合キャスト自身が行なう．また，複合キャストに対する行動戦略の与え方に関しても，幾つかの与え方が考えられる．行動戦略の与え方とキャストの並列スケジューリングに関しては，それぞれ 4.2節と 4.3節で述べる．

4.2 行動戦略

行動戦略とは，アニメータによって与えられるキャストに対する動作の指示 (表示・消去・移動・点滅など) のことである．アニメータがキャストに何らかの動作を行わせたい場合には，

- 行わせたい動作を直接指示する．例えば，" 位置 P に移動" といったように行うべき行動を直接指示する．このようにして動作が指示された時には，キャストは即座にその動作を行う．
- 行わせたい動作 (複数でも構わない) を，行動戦略として与える．例えば，" 位置 Q に移動" と" 点滅" の二つの行動戦略を与える．その後，行動戦略の実行開始を指示すると，キャストはそれらの行動戦略を実行する．

の二通りの方法でその動作を指示することができる．逆に，何も動作を行わせたくないキャストに対しては，アニメータは何も動作を指示する必要はない．

キャストは予め行動戦略が与えられている場合には，その行動戦略に従って表示を行なう．また，原始キャスト・複合キャストのどちらに対しても行動戦略を与えることが可能である．原始キャストの場合，そのキャスト自身が行動戦略を解釈し表示を行なう (行動戦略の解釈については，後で述べる)．複合キャストの場合，そのキャストは自分の管理しているキャストに対して行動戦略の並列スケジューリングを行う (4.3節)．更に，一つのキャストに対して複数の行動戦略を与えることが可能である．この場合，キャストはアニメータによって指定される以下のいずれかの基準に従ってとるべき行動を決定する．

- 行動戦略に優先度を付け，その優先度の高い行動戦略に従う．
- 複数の行動戦略を合成する (4.3節)．

  例えば図7に示すように，文字列に対して右に移動する (行動戦略 A)・上に移動する (行動戦略 B) の二つの行動戦略を与えた場合，右上に移動するというような行動戦略を合成する．

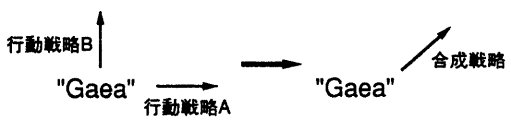

図 7: 行動戦略の合成

これによって，複合キャストにグループのデフォルトの行動戦略を与えグループに含まれる特定のキャストに個別の行動戦略を与えることが可能となり，グループ全体としてはデフォルトの行動戦略に従つつ特定のキャストは個別の行動戦略に従うといったような動作が容易に記述できる．

キャストに対する行動戦略の与え方に関しては，例えば以下のような与え方が考えられる．

(A-1) 単なる命令列 (表示・消去・移動・点滅などの命令) で与える．つまり，“表示⇒ 移動⇒ 点滅⇒ 消去” といったように，行動を順次具体的に指示する．

(A-2) 時系列に沿った命令列で与える．つまり，” ある時点から 10 単位時間表示し，次の時点から 10 単位時間点滅する” といったように，時間と行動を具体的に指示する．

(A-3) 移動の経路を指示するなどの場合には，方程式で与える．例えば，移動の軌跡を表現した運動方程式などを用いて移動経路を指示する．

どのような形式で行動戦略が与えられようともそれを解釈する場合には，キャストは与えられた行動戦略を” 時系列に沿った命令列の表 (時系列表)” に展開する．(A-1)~(A-3) のような形式で与えられた行動戦略を解釈する場合について以下に述べる．(A-2) に関しては，時系列に沿った命令列が行動戦略として与えられるため時系列表にするのは非常に簡単である．(A-1) に関しては，個々の命令にかかる時間を決めてやれば時系列表にすることが可能である．個々の命令にかかる時間はどの程度の滑らかさでアニメーションを表示するのかに依存するため，アニメーションの滑らかさをアニメータに指定してもらうことを考えている．(A-3) に関しては，例えば運動方程式の場合，運動方程式と初期値 (初期位置・初期速度・初期加速度) が分かれば次の時刻での位置・速度・加速度を計算できる．このようにして順次方程式を解いていくことにより，時系列表を得ることが可能である．また，時系列表の時間の粒度については (A-1) の場合と同様に，アニメータに指定してもらうことを考えている．

4.3 並列スケジューリング

複合キャストに対して行動戦略が与えられた場合，以下のような処理によって管理する複数のキャストを並列スケジューリングする．

(1) 複合キャストは，与えられた行動戦略を操作の時系列表に展開する．つまり，ある時間にあるキャストがするべき動作を表に展開する．この際，時間の指定の無い行動戦略については，以下のいずれかの規準に従って複合キャストがグループ全体の時間を管理する．

  - 一番時間のかかる動作に全体の時間を合わせる．例えば，キャスト A が移動するのに Ta 時間かかり，キャスト B が移動するのに Tb

時間かかるとする (ただし，Ta > Tb)．この場合，全体として Ta 時間で移動することになり，B がゆっくりと移動するように表示される．

- 一番時間のかからない動作に全体の時間を合わせる．上の例の場合を考えると，全体として Tb 時間で移動することになり，A が早く移動するように表示される．
- 各動作にかかる時間の平均時間に全体の時間を合わせる．同様に上の例の場合を考えると，全体として (Ta+Tb)/2 時間で移動することになり，A が少し早く移動し B がすこしゆっくり移動するように表示される．

(2) (1) で作られた時系列表に従い，管理する各キャストに動作の指示を与える．

また，行動戦略の合成は時系列表を合成することによって行なう．例えば図8に示すように，時系列表 A と時系列表 B を合成して時系列表 C を得る．この場合の合成方法は，表の対応する項目同士の単なる和となっている．

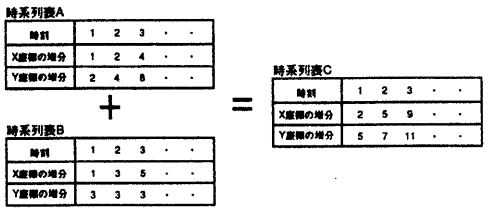

図 8: 時系列表の合成

5 おわりに

本稿では，アニメータの意図する表示を容易に実現するための道具として，アルゴリズム・アニメーション用ツールキット **Gaea** を提案し，**Gaea** の持つ機能である変身機能と並列描画機能について述べた．変身機能においては，描画情報の自動変換や変身戦略のカスタマイズを行なうことにより，様々な形式で変身の過程を描画できることを述べた．並列描画機能においては，複合キャストを用いることにより，複数のキャストを同じ行動戦略に従って表示できることを述べた．また，キャストに複数の行動戦略が与えられた場合キャスト自身がそれらを合成することにより，より複雑な行動戦略を簡単に与えることができることを述べた．

最後に，**Gaea** を用いた応用システムとして現在検討中であるアニメーションビルダ **Gigant** (Graphical Interface for Gaea ANimation Toolkit) について述べる．**Gigant** とは，以下のような機能を持つインタラクティブなアニメーション作成支援システムである．

- アニメータが記述したアニメーション・コードを解釈し，画面上に表示する．

  アニメーション・コードとは，**Gaea** を用いて画面上に表示を行うためのコマンド列である．**Gigant** は，アニメータが作成したアニメーション・コードを即座に解釈し，画面上に表示する (図 9)．また，

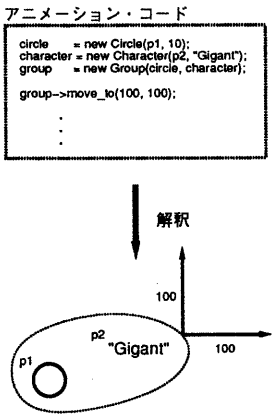

図 9: アニメーションとコードの解釈

  コマンドラインからのコード入力を許し，その結果を即座に画面上に反映させるインタラクティブな表示機能を持つ (図 10)．これによって，アニメーションのプロトタイプの試作が容易に行える環境を提供できると考えている．

- 画面上でアニメータが例示した動作を解釈し，その動作を行うアニメーション・コードを自動生成する．

  Gigant は，アニメータが画面上でマウスなどを用いて例示した動作を解釈し，その動作を行うアニ

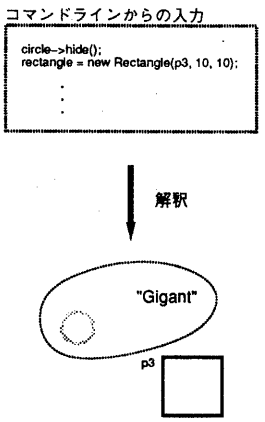

図 10: インタラクティブな表示

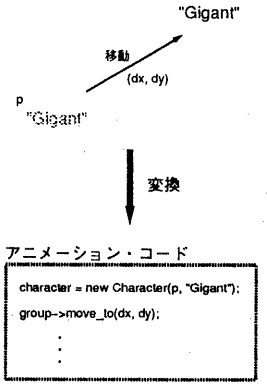

図 11: 例示によるアニメーション・コードの作成

メーション・コードを自動生成する (図 11). また, 描画ツールなどを用いて描かれた絵を解釈し, その絵を表示するキャストのコードに変換する. 具体的には, その絵を表示する新たなキャストのアニメーション・コードを生成することになる. これによって, 描画ツールによって描かれた絵を用いたアニメーションの作成が容易に行える. 更に, コードを記述するのが容易でないような動きをキャストにさせたい場合でも, その動きを例示することで容易にコード化することができる.

Gigant は **Gaea** が提供する機能を用いるための使い易いインタフェースを提供することにより, 視覚的な操作によるアニメーション・コードの作成支援を行うことを目指している. アニメータは, エディタなどを用いてアニメーションのためのコードを記述するのではなく, 描画ツールを用いて表示したいキャスト (の形状) を作成する. 更に, 作成したキャストをマウスなどを用いて実際に動かしてみることによって, そのキャストに行動戦略を与える.

今後の課題としては, **Gaea** のプロトタイプの実現及び機能の拡充, **Gigant** の詳細設計及びそのプロトタイプの実現, **Gaea** や **Gigant** を用いたアルゴリズム・アニメーションシステムの試作などを行なう予定である.

参考文献

[1] M.H. Brown. *Algorithm Animation.* The MIT Press, 1988.

[2] M.H. Brown. Exploring algorithms using balsa-ii. *IEEE Computer*, 21(5):14–36, 05 1988.

[3] M.H. Brown. Zeus: A system for algorithm animation and multiple-view editing. In *1991 IEEE Workshop on Visual Languages*, pages 10–27, 10 1991.

[4] K.M. Kahn. Concurrent constraint programs to parse and animate pictures of concurrent constraint programs. In *The International Conference on Fifth Generation Computer System 1992*, volume 2, pages 943–950, 06 1992.

[5] M. Maeda. Implementing a process oriented debugger with reflection and program transformation. In *The International Conference on Fifth Generation Computer System 1992*, volume 2, pages 961–968, 06 1992.

[6] 和田, 前田. アニメーションシステム構築のためのツールキットに関する考察. 日本ソフトウェア科学会第 9 回大会論文集, pages 209–212, 09 1992.