ビジュアルシステム生成系『恵比寿』における ジェスチャの実現

Realizing Gesture on Visual System Generator “Eviss”

山田 英仁 † 飯塚 和久 †† 田中 二郎 †††
Hideto YAMADA Kazuhisa IIZUKA Jiro TANAKA

† 筑波大学大学院 博士課程 システム情報工学研究科
Doctoral Program in Systems and Information Engineering, University of Tsukuba
†† 筑波大学大学院 博士課程 工学研究科
Doctoral Program in Engineering, University of Tsukuba
††† 筑波大学 電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba
{yama,iizuka,jiro}@iplab.is.tsukuba.ac.jp

本論文では，ビジュアルシステム生成系「恵比寿」でのジェスチャの実現について述べる.「恵比寿」は，図形文法にしたがった仕様から，ビジュアルシステムを生成する生成系である．本研究では，従来の「恵比寿」の図形文法に，マウスイベント，マウスの軌跡，ジェスチャを表すトークンを追加し，図形文法の枠組でジェスチャを記述できるようにした．これにより，従来の「恵比寿」におけるビジュアルシステムの仕様に加え，マウスの軌跡の取得方法や，ジェスチャに対する動作について記述することで，ジェスチャを扱うシステムを生成できるようになる．

1 はじめに

ジェスチャとは，一般的にはある事柄を表現するための身振り手振りのことを指す． 本研究では，マウスで描いた特定の軌跡に意味を持たせ，軌跡に応じてシステムの操作を行うマウスジェスチャを対象とする． ジェスチャには，マウスの右ボタンを押しながら左右へ動かす，図形の上でジグザグ線を描く，といった操作などがある． ジェスチャを用いることで，メニューを呼び出すことなくマウスのみで操作を行うことができる．

ジェスチャを用いたシステムを開発する場合，ジェスチャの登録や動作を，プログラムを書いて定義する必要がある． これに対し我々はビジュアルシステム生成系「恵比寿」[1, 2, 3] 上でジェスチャを実現する．しかし，従来の「恵比寿」には描かれたジェスチャを解析するための機構が用意されていないため，SATIN[4] のジェスチャ解析器を用いることでジェスチャの解析を自動的に行わせる． ユーザは，ジェスチャを扱えるシステムの開発を行うときにジェスチャに対する動作のみを考慮すればよく，ジェスチャの認識や解析は恵比寿と SATIN に行わせることができるので，システム開発の手間を軽減することができる．

2 恵比寿

「恵比寿」は，我々の研究室で開発しているビジュアルシステム生成系である．ここでいうビジュアルシステムとは，回路図や楽譜，数式など，図形どうしの関係に何らかの規則が定められているものを扱うシステムである． この図形どうしの規則を表すものを図形文法という． また，図形文法を自分で定義することで，ビジュアルシステムを生成することができる (図 1).

2.1 拡張 CMG

恵比寿では，図形文法として Constraint Multiset Grammars(CMG)[5] を拡張したものを用いている．拡張 CMG[1, 2, 3] とは，従来の CMG に対して，図形の書き換えなどを定義するアクションが追加されたものである．

拡張 CMG では，生成規則を以下のように定義する．

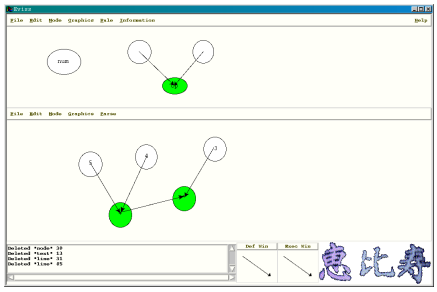

図 1: 恵比寿のスクリーンショット

```
P ::= P1,...,Pn where (
    制約
) {
    属性
} {
    アクション
}
```

この生成規則は，左辺の記号 P が右辺の記号 P_1,...,P_n から構成されていることを表している．

制約部分には，生成規則が適用されるための条件を記述し，属性部分には，左辺 P の属性を記述する．

アクション部分には，図形の書き換えなどを行うスクリプトを記述する．

2.2 恵比寿における処理の流れ

従来の恵比寿は，図形文法を入力する図形文法入力部と図形の入出力を行う図形入出力部，入力された図形を解析するパーサから構成される． ユーザはまず図形文法を入力し，図形間の規則を定義する． その後，図形入出力部に図形を入力すると，入力された図形のデータがパーサに送られ，図形文法にしたがって解析される． 解析結果にもとづいて図形の書き換えが行われ，その結果は図形入出力部に反映される．

2.3 ジェスチャを実装する際の処理の流れ

恵比寿でジェスチャを実現する際，マウスの軌跡の取得および描かれたジェスチャに対する動作の定義を図形文法で行いたい． そのために，図形文法の処理を行うパーサでマウスの軌跡や軌跡の解析結果を扱えるようにする． パーサでマウスの軌跡を扱うために，図形入出力部を改良してマウスイベントが発生したときにマウスの座標や押されたボタンなどをパーサに送るようにする． パーサには，マウスの座標が複数送られてくるので，これらから成るリストを生成，マウスの軌跡として扱う．

ジェスチャの軌跡を解析する解析部は SATIN のマウスの軌跡を解析する部分にあたる Recognizer を用いる． パーサで生成したマウスの軌跡を SATIN の Recognizer に送る． SATIN ではマウスの軌跡の解析を行い，マウスの軌跡がどのようなジェスチャを表すか，というデータをパーサへ返す． パーサでは，マウスの軌跡の解析結果および図形文法をもとに図形の書き換えを行い，その結果は図形入出力部に反映される．

2.4 新しいトークンの追加

現在の拡張 CMG にマウスの軌跡や解析結果を新しいトークンとして追加することで，ジェスチャの情報の記述を行う． 拡張 CMG におけるトークンと記号の関係は，オブジェクト指向の用語を用いると，記号がクラスであり，トークンがオブジェクトに相当する [1, 2]． また，マウスの軌跡の取得にはマウスイベントが必要なので，マウスイベントのトークンも追加する．

2.4.1 Mouse トークン

Mouse トークンは，図形入出力部で発生したマウスイベントを表す． Mouse トークンの属性は 6 つである．

```
Mouse(point pos, boolean leftbutton,
boolean middlebutton, boolean rightbutton,
string state, integer time)
```

pos にはマウスの座標が数値で表される． leftbutton, middlebutton, rightbutton はイベントが発生したとき，それぞれ左，中，右ボタンが押されていたかどうかを表し，boolean で表される． state はイベントが発生したときのマウスの状態を表す． マウスの状態には，ボタンが押された状態 (pressed)，離された状態 (released)，マウスが動いている状態 (move) がある．time は，イベントが発生したときの時間が数値で表される．

2.4.2 Stroke トークン

Stroke トークンは，ジェスチャを解析するために使用するストローク情報を表す．Stroke トークンの

属性は 2 つである．

```
Stroke(list pos_list, list time_list)
```

Stroke トークンの属性には，マウスの軌跡を表すのに必要なマウスの座標の集合 pos_list と，それぞれのマウスの座標に対する時間情報 time_list がある．

2.4.3 Gesture トークン

Gesture トークンは，ジェスチャの情報として必要なものを扱う． Gesture トークンの属性は 5 つである．

```
Gesture(string pattern, rectangle bounds,
integer length, point beginpos,
point endpos)
```

pattern にはジェスチャの意味を示す文字列が入る．pattern を用いることでジェスチャの区別を行う．

bounds にはバウンディングボックスの座標や大きさが入る． バウンディングボックスは矩形で表される． bounds は，ジェスチャがどの図形の上で描かれたかを判別するために使用する． なぜなら，ジェスチャの動作には，ブラウザの進む/ 戻るなど，それ単体で動くものもあるが，図形のカット，コピー，ペーストなど，ジェスチャの操作対象を指定する必要のある動作が存在するためである．

length には，ジェスチャの軌跡の長さの数値が入る． length を用いることで，視点の移動量を決める，といった操作を行うことができる．

beginpos と endpos は，それぞれジェスチャの描き始めの座標，描き終わりの座標が入る． beginpos と endpos を用いることで，グラフの 2 つの図形を結ぶ，といった操作を行いたいときに，ジェスチャが 2 つの図形にかかっているかを判別することができる．

3 ジェスチャの実現

3.1 マウスイベントからの軌跡の生成

マウスイベントから軌跡を生成するには，Mouse トークンを用いて，Stroke トークンの生成と更新を行う． Mouse トークンは図形入出力部からパーサにマウスイベントが送られてきたときに，パーサによって生成される．

最初に，Mouse トークンを構成要素としてマウスの軌跡 Stroke トークンを新しく作る CreateStroke を定義する． 制約部分では，マウスの座標の取得を開始する条件を記述する． アクション部分では createToken() メソッドを用いて Stroke トークンを生成する．createToken() は新しくトークンを生成するメソッドであり，新しく作られたトークンを返す．生成された Stroke トークンに対して，リストを作るメソッド createList() を用いてマウスの座標を Stroke トークンの属性 pos_list に追加している．Stroke トークンを生成した後は deleteToken() メソッドを用いて Mouse トークンを削除している．Mouse トークンが削除されるので，CreateStroke の構成要素が無くなり，制約が成り立たなくなる．このため，CreateStroke も消去される． このことにより，Stroke トークンが 2 度生成される，ということを防止する．

次に，マウスの軌跡の情報を更新する UpdateStroke を定義する．制約部分では，Stroke トークンを更新する条件を記述する． アクション部分では，新しく Stroke トークンを生成するようにする． 新しい Stroke トークンの属性 pos_list は，古い Stroke トークンの pos_list にマウスの座標を追加して生成される． 新しい Stroke トークンが作られた後は，古い Stroke トークンと Mouse トークンが deleteObject() によって削除される．

マウスの軌跡の生成の例として，マウスの右ボタンが押されたらマウスの座標の取得を開始し，マウスの右ドラッグ中に Stroke トークンの更新を行う操作を拡張 CMG を用いて以下に記述する．

```
C:CreateStroke ::= M:Mouse where (
    M.rightbutton == "true" &&
    M.state == "pressed"
) {} {
    S = createToken("Stroke")
    S.pos_list = createList(M.pos)
    deleteToken(M)
}

U:UpdateStroke ::= S:Stroke, M:Mouse
where (
    M.rightbutton == "true" &&
    M.state == "move"
) {} {
    N = createToken("Stroke")
    N.pos_list = addList(S.pos_list,
M.pos)
    deleteToken(M)
    deleteToken(S)
```

```
}
```

この例では，軌跡の取得の条件として，Mouse トークンの属性 rightbutton と state を調べている． この記述を変更することにより，様々な条件に対応することができる．

3.2 軌跡の情報の解析と Gesture トークンの生成

Stroke トークンと Mouse トークンを構成要素として Gesture トークンを生成する CreateGesture を定義する． 制約部分では，マウスの軌跡の取得を終了する条件を記述する． アクション部分では，createGesture() を用いてジェスチャ情報を生成する．createGesture() は，Stroke トークンを受け取って，それを解析してジェスチャ情報を生成し，Gesture トークンを返すメソッドである．createGesture() を呼び出したときに SATIN の Recognizer が呼ばれ，マウスの軌跡の解析を行う． マウスの軌跡の解析が終了した後は，認識結果をパーサに返す．パーサでは認識結果をもとに Gesture トークンを生成する． Gesture トークンを生成した後は，Mouse トークンと Stroke トークンを削除する．

以下に，マウスの右ボタンが離されたときに軌跡の取得を終了し，ジェスチャを生成する例を示す．

```
C:CreateGesture ::= S:Stroke, M:Mouse
where (
    M.rightbutton == "true" &&
    M.state == "released"
) {} {
    G = createGesture(S)
    deleteToken(M)
    deleteToken(S)
}
```

3.3 ジェスチャと動作との関連付け

ジェスチャと動作との関連付けを行うために，Gesture トークンを構成要素として CMG を記述する．描かれたジェスチャのパターンを表す Gesture トークンの属性 pattern を制約に使用することで，ジェスチャのパターンに応じた動作の関連付けを記述する．

まず，ジェスチャが描かれた場所に円を生成する CreateCircle の記述例を示す．制約部分ではジェスチャのパターンが”drawcircle” になっているかを調べる．アクション部分では円を表す Circle を生成する． Circle の中心座標は Gesture トークンのバウンディングボックスをもとに決定する．

```
C:CreateCircle ::= G:Gesture where (
    G.pattern == "drawcircle"
) {} {
    C = createToken("Circle")
    C.mid = (G.bounds_lu + G.bounds_rl) / 2
}
```

次に，円を削除する例を示す．制約部分ではジェスチャのパターンおよび図形とジェスチャのバウンディングボックスが触れているかどうかを調べる．touch() は 2 つの図形が触れているかを調べるメソッドである． 図形が触れている，というのは，互いの図形の一部分が重なっている状態のことである．アクション部分では，deleteToken() メソッドを用いて，Circle を削除している． 複数個の図形の上で図形消去のジェスチャが描かれた場合，DeleteCircle が複数個生成され，全ての図形が消去される．

```
D:DeleteCircle ::= G:Gesture, C:Circle
where (
    G.pattern == "delete" &&
    touch(G.bounds, C)
) {} {
    deleteToken(C)
}
```

4 まとめ

我々は，恵比寿でジェスチャを扱えるようにするため，恵比寿でジェスチャの情報，マウスの軌跡の情報，マウスイベントを扱えるようにした． また，拡張 CMG にマウスイベント，マウスの軌跡，ジェスチャを表すトークンを追加することで，拡張 CMG の枠組でジェスチャを記述できるようにした． マウスの軌跡の取得とジェスチャに対する動作について記述することで，ジェスチャを扱うシステムを自動的に生成できるようにした．

参考文献

[1] A. Baba and J. TAnaka. Eviss: A visual system having a spatial parser generator. In *Proceedings of Asia Pacific Computer Human Interaction 1998 (APCHI'98), IEEE Computer Society Press*, pp. 158–164, 1998.

[2] 馬場昭宏, 田中二郎. 「恵比寿」を用いたビジュアルシステムの作成. 情報処理学会論文誌, Vol.40, No.2, pp. 497–506, 1999.

[3] 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, pp. 1385–1394, 1998.

[4] Jason I. Hong and James A. Landay. SATIN: a toolkit for informal ink-based applications. In *UIST*, pp. 63–72, 2000.

[5] R. Helm, K. Marriott, and M. Odersky. Building visual language parsers. In *Conference proceeding on Human Factors in Computer Systems (CHI '91)*, pp. 105–112, 1991.