ATRテクニカルレポート表紙

［非公開］

TR-M-0038

議論の時間的構造と意味的構造の可視化

藤田　邦彦
Kunihiko FUJITA

西本　一志
Kazushi NISHIMOTO

角　康之
Yasuyuki SUMI

國籐　進
Susumu KUNIFUJI

間瀬　健二
Kenji MASE

1998. 3.31

ATR 知能映像通信研究所

議論の時間的構造と意味的構造の可視化
Meeting Support by Visualizing Discussion Structure and Semantics.

藤田邦彦[††]　西本一志[†]　角 康之[†]
國藤 進[‡]　間瀬健二[†]
Kunihiko FUJITA,　Kazushi NISHIMOTO,　Yasuyuki SUMI,
Susumu KUNIFUJI,　Kenji MASE

概要

本研究では，電子会議環境における議論構造の可視化手法を提案し，プロトタイプシステムを実装した．提案手法は，議論内容の変遷や全体構造の把握を容易にするための発言間の時間的・因果的な関係の可視化と，各意見間の意味的な関係の可視化を両立させるものである．時間的・因果的な関係は，各発言をノードとし，関係のある発言間をリンクで結んだ木構造で表現される．意味的な関係は，スプリングモデルを用いて内容の遠近に応じてマッピングされる．両者の関係は3次元空間内においてシームレスな融合を果たしており，ユーザは，両者の関係間の行き来を思考の連続性を保ちつつ行うことができる．

[†] (株) エイ・ティ・アール知能映像通信研究所
[††]北陸先端科学技術大学院大学 情報科学研究科

1　はじめに

本研究では，電子会議環境において，議論内容の変遷や全体構造の把握を容易にするため，意見間の時間的・因果的な関係と意味的な関係を両立して可視化するシステムを提案する．

会議においてはしばしば意見が錯綜し，全体の情勢が把握できず，議論の行方が判然としないことがある．このような状態においては，参加者は議論の内容の解析に手間取り，有効な意見やアイデアを出すのがおろそかになってしまうと考えられる．一連の意見がどのような関係で結び付いているかが提示されれば，議論内容の把握は容易になり，参加者は意見やアイデアの考察に集中できるだろう．

このようなニーズに応えて開発された一連の電子会議システムは，「構造的アプローチによるグループウェア」に分類される．これは，人々の間のインタラクションから，ある共通的なパターン構造を抽出してモデル化し，このモデルに基づいて構築されたグループウェアのことを指す．このような構造的アプローチによるグループウェアは，いわば意見間の時間的・因果的な関係を可視化するものであり，様々な局面で一定の評価を得てきたが，意思決定や発想支援などの機能を含むものは，これまでに開発されてこなかった．

一方，発想支援システムの研究の一環として，テキスト情報から得られたキーワード情報をもとに，テキスト間・キーワード間の類似関係を空間にマッピングするという手法が開発されてきた [田村 92] [杉本 94] [Sumi 97] [高杉 97] [斎藤 97] [野間 97]．この手法では，アイデアや知識の断片をテキスト情報として入力し，その空間での配置をユーザが観察することにより，何らかの分類的意味をくみとり，収束的または発散的な発想を得ることを意図している．

この手法を会議に適用すると，意見間の意味的関係が，空間上の配置から得られることになる．これを，意見間の時間的・因果的な関係とともに効果的に両立させた可視化を行うことにより，議論の内容や変遷の把握を容易にする支援のみならず，意思決定や発想までも支援できるシステムが構築できるとの仮説を立てた．

関連研究としては，2次元上での意見やアイデアのマッピングという観点と，会議の談話構造の解析という観点から，大きく2つに分けられる．

まず，2次元上での意見やアイデアのマッピングという観点で類似の研究として代表的なものに，[Sumi 97] や [高杉 97] があげられる．[Sumi 97] では，個人の持つ情報や視点をグループで共有するために，対話中の発言群で構成される情報空間の構造の可視化を行う環境 AIDE が紹介されている．AIDE では，双対尺度法を応用して対話空間の構造を可視化する．ユーザはツールから提示された対話空間を操作しながら，自分のアイデアを再認識したり再構成したりする．また [高杉 97] では，バネモデルを用いて同様のシステム DW(Dancing word) を構築している．両者とも，一つ一つの意見やアイデア，発言などを2次元上にオブジェクトとしてマッピングしており，どの意見やアイデア，発言同士が似ているか，2次元空間上のどの辺りの意見やアイデア，発言が出現していないかなどの情報が直感的に把握できるようになっている．このため，発散的思考（場合によっては収束的思考も）の支援には効果的であるが，議論の経過を把握するには，時系列的な構造を表示できないため適していない．

また，会議の談話構造の解析を目的としているという観点で類似の研究としては，[渡辺 92] があげられる．[渡辺 92] では，参加者の各意見を，誰のどの意見に対するものかを宣言してから意見し，意見を関係に基づいて線でつないで表示することで，意見の関係と議論の進展を視覚的に表示できるようにしている．しかしながら，[渡辺 92] では，意見間に関係がある場合，単に

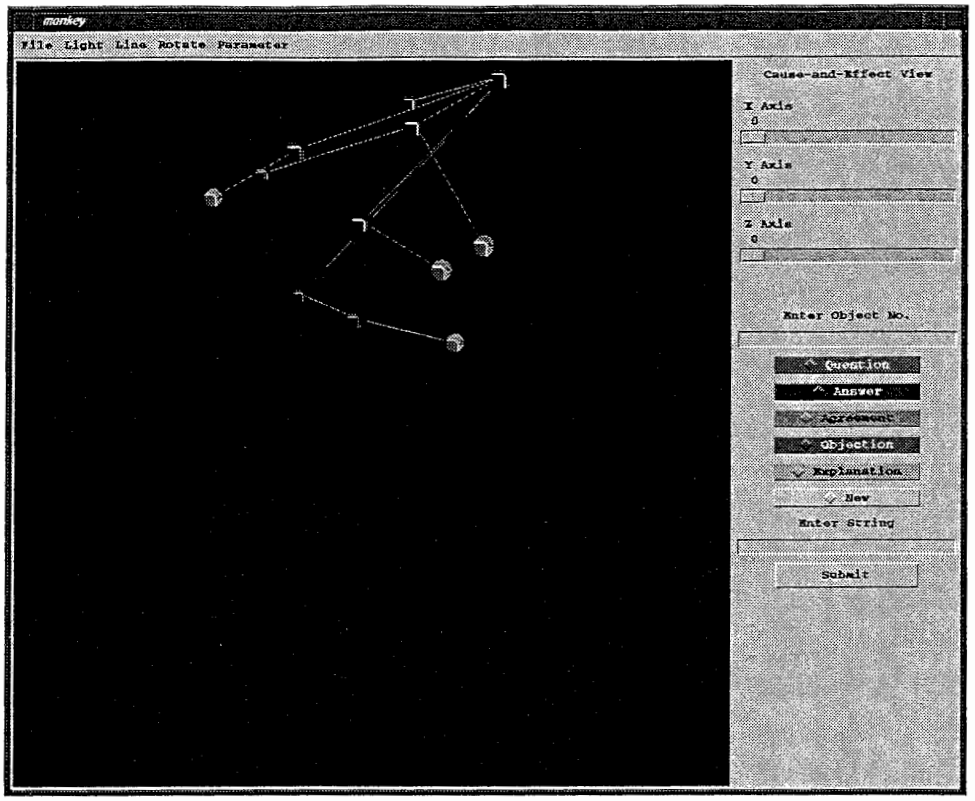

図 1: 時間的・因果的な関係の可視化

線で結んでいるだけである．本研究では [渡辺 92] よりさらに踏み込んで，各意見の関係性を図示することをもくろんでいる．

本報では，まず意見間の時間的・因果的な関係の可視化のモデルについて述べ，次に意見間の意味的な関係の可視化の手法について説明し，２つの可視化を効果的に両立させる方法について述べ，実験システムのプロトタイプを紹介し，評価方針について述べる．

なお，ここでは主として議論を収束に導く会議を想定している．これは，意見の関係性を入力させることが会議の発散的な過程には向かないことと，本システムが採用したバネモデルが，どの意見や発言同士が似ているか，そのまとまり具合を可視化するものであることに起因する．

2　時間的・因果的な関係の可視化

構造的アプローチによるグループウェアの代表的なものとして，gIBIS があげられる [阪田 92]. これは，H.Rittel によって開発された討論のモデルである IBIS モデルをベースにしている．IBIS モデルでは，発言の内容を Issue（問題），Position（立場），Argument（意見），Other（その他）の４つの要素に分け，これらの発言間の関係を，賛成，反対，応答，一般化，特殊化，提案，支持，その他，で表現している．

本研究では，発言の内容については特に分類せず，発言間の関係は，賛成と支持を同意に，一般化と特殊化を説明に，提案を新規議題で代表させた．これは，発言の内容は発言間の関係によりある程度は判明するであろうと考え，また関係の種類をできるだけ少なくすることにより，ユー

ザの混乱と操作の煩雑さを防ぐためである．よって，ここでは会議における意見間の関係は以下の６つで代表することにする．

1. 質問 (Question)
2. 応答 (Answer)
3. 同意 (Agreement)
4. 反対 (Objection)
5. 説明 (Explanation)
6. 新規議題 (New)

これらの関係性を用いて議論の構造を可視化する．

実装においては，各意見はオブジェクトとして表示され，ある意見とそれをフォローする意見の間にリンクが張られる形で，議論の可視化が行われる．ユーザは発言にあたっては，どの発言に対してどのような関係をもって発言するかを，発言の内容とともにシステムに入力する必要がある．これはユーザに多少煩雑な操作を強いるが，一方で，どの意見に対してどのような関係をもって発言するかが明確になるため，無駄な発言（チャット・あいづち）が減り，議論に有効な発言が促される．さらに，意見間のつながりが可視化されているため，複数の議論を並行して行うことも可能となる [渡辺 92]．

また，リンクが張られていない意見同士が全く関係がないというわけではない．本システムを使用する際には，極力，ある１つの意見に対するフォローは，それに関連する話題だけを述べるのが望ましく，多くの意見に対して，１つのフォローに全て含まれるのは推奨されない．なぜなら，仕様とシステムの制限により，リンクが張れるのは１つの意見に対してのみだからである．

実装にあたっては，関係はオブジェクトを色分けすることで可視化を行った．また，一つの発言から複数の発言にリンクが張られることを許した．このため，議論の全体構造は木構造として表現されることになる（図1参照）．

3　意味的な関係の可視化

アイデアや知識の断片を２次元空間上にマッピングした場合にどのような分布になるかをユーザに提示すると，発想を支援する効果があることが実証されている [田村 92] [杉本 94] [Sumi 97] [高杉 97] [斎藤 97] [野間 97]．本システムでは，この成果を電子会議に応用して，一つ一つの意見をアイデアや知識の断片とみなし，意見間の関係を示しながら，その分布を表示できるような機能を付加する．

オブジェクトの類似関係を空間にマッピングするには，大きくわけて２つの方法がある．１つは主成分抽出に基づくマッピング [Sumi 97] [杉本 94] であり，１つは力学的シミュレーションによるマッピング [田村 92] [高杉 97] [斎藤 97] [野間 97] である．前者の手法を用いる場合，各々の１対１のオブジェクト間の距離から，全体のオブジェクトの空間配置の最適な解を得られるが，オブジェクトが追加・削除されると，再計算が発生し空間配置が大きく変化する．一方，後者の手法を用いた場合，必ずしも前者の手法によって得られるような最適な空間配置が行われるとは

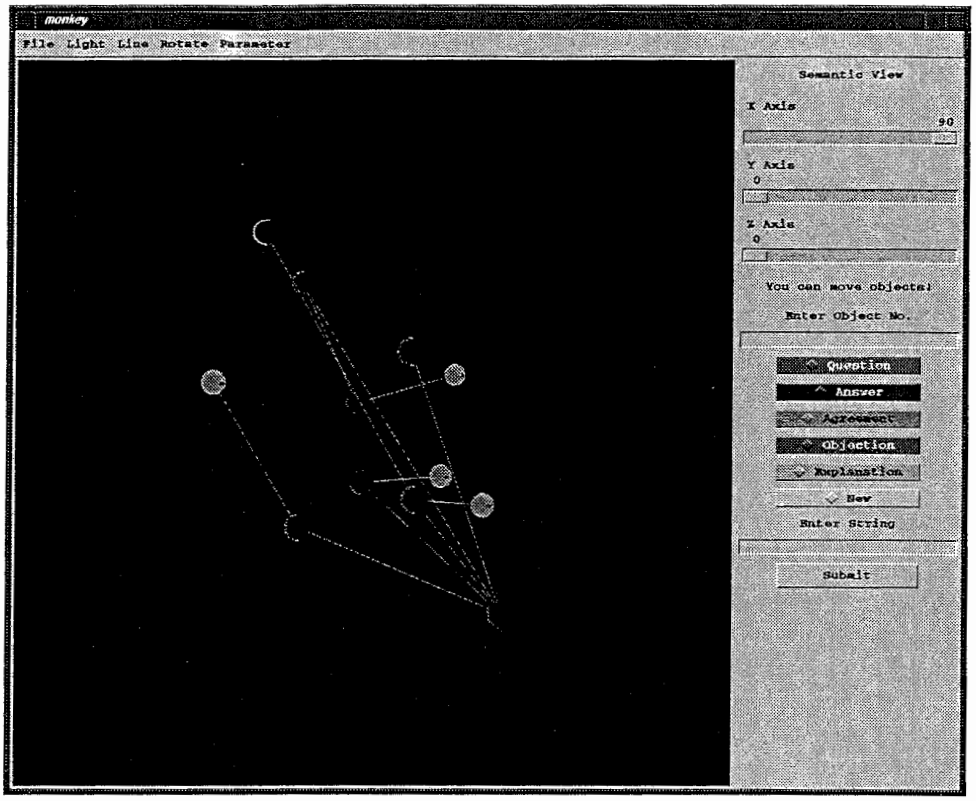

図 2: 意味的な関係の可視化

限らないが，オブジェクトの追加・削除などの操作に対しては，徐々に新しい空間配置に変化するため，ユーザは思考の連続性を保つことができる．本研究では，電子会議への応用を目指しており，オブジェクト（＝意見）の追加が頻繁に行われることを前提としているため，力学的シミュレーションによるマッピングを採用し，バネモデルを実験システムに実装することにした（図2参照*）．

バネモデルとは，オブジェクトとオブジェクトの間にバネが張られていると想定し，オブジェクト全体のバネによる引力ないし斥力のエネルギー量が最小になるよう配置を行うものである．このときの力学的数値（バネの自然長など）は，オブジェクト中のキーワードの共起性に基づいて計算される．具体的には，オブジェクトのテキスト情報から算出されたキーワードベクトル [渡部 91] を用いて決定される．以下で詳細を述べる．

3.1 類似度の算出

オブジェクトに含まれるテキスト情報に対し，「茶筌」[松本 97] を用いて形態素解析を行い，名詞と未定義語をキーワードとして採用する．キーワードの総数を n とし，各キーワードに番号を付与する．あるテキスト t の i 番目のキーワードの出現回数を W_i^t とすると，このテキスト t の

*画面中のオブジェクト間に張られている白い線は，バネではなく，前章で述べた，因果的関係のある意見同士に張られるリンクである．バネは，表示されないが全てのオブジェクト間に張られている．

キーワードベクトルは，

$$\mathbf{a}_t = (W_1^t, W_2^t, \cdots, W_i^t, \cdots, W_n^t)$$

となる．しかしこれでは各キーワードの重みを平坦にみており，各テキストに特徴的なキーワードの重みが浮かび上がらない．そこで，キーワードの各テキストにおける出現確率と全テキストにおける出現確率を考慮することによって重みをつけ直す．

テキスト t におけるキーワード i の出現確率 p_i^t は，以下のようにして算出される．

$$p_i^t = \frac{W_i^t}{\sum_k W_i^k}$$

また，テキスト集合全体におけるキーワード i の出現確率 p_i^{all} は，以下のようにして算出される．

$$p_i^{all} = \frac{\sum_k W_i^k}{\sum_k \sum_i W_i^k}$$

さらに，この p_i^t と p_i^{all} を用いて，テキスト t におけるキーワード i の重要度（p_i^{all} の p_i^t からのダイバージェンス）を，以下の式により計算する．

$$W_i^{t'} = p_i^t \log\left(\frac{p_i^t}{p_i^{all}}\right)$$

これにより，キーワード i の，全体における出現確率に対して，テキスト t における出現確率が高いほど，重みがつく．

$$\mathbf{a}'_t = (W_1^{t'}, W_2^{t'}, \cdots, W_i^{t'}, \cdots, W_n^{t'})$$

また，正規化されたキーワードベクトル $\mathbf{v}_t$ は，

$$\mathbf{v}_t = \frac{\mathbf{a}'_t}{|\mathbf{a}'_t|}$$

とする（ただし，$|\mathbf{a}'_t| = \sqrt{W_1^{t'2} + \cdots + W_n^{t'2}}$）．

正規化キーワードベクトル $\mathbf{v}_{t_1}, \mathbf{v}_{t_2}$ を持つテキスト t_1, t_2 間の類似度 R_{12} を，次のようにベクトルの内積で定義する．

$$R_{12} = \mathbf{v}_{t_1} \cdot \mathbf{v}_{t_2}$$

3.2 バネの理想距離の算出

オブジェクト i とオブジェクト j の間に張られるバネの理想距離 l_{ij} は，類似度 R_{ij} を用いて，以下のようにして算出される．

$$l_{ij} = m\frac{1}{R_{ij}}$$

理想距離調整数 m は任意の定数で，空間配置しようとする画面の大きさにより調整する．この式の通り，バネの理想距離は，オブジェクトに含まれるテキスト間の非類似度から求められる．非類似度は，類似していないほど大きな値をとる非負の実数で，類似度の逆数を用いる．これにより，類似度の大きいオブジェクト同士が近く，類似度の小さいオブジェクト同士が遠くに配置される．

3.3 オブジェクトの空間配置

オブジェクトを空間に配置するにあたって，オブジェクト i とオブジェクト j の間の理想距離（バネの自然長）を l_{ij}，実際のオブジェクト間の距離を d_{ij}，バネ定数を k としたとき，バネの張られたオブジェクト間に働く力を，

$$f_{ij} = k(d_{ij} - l_{ij})$$

で定義する．ただし，k は任意の正定数とする．ここで，f が正のとき引力となり，負のとき斥力となる．このとき，全体のエネルギーの総計 E は，

$$E = \sum_i \sum_j \frac{1}{2} k(d_{ij} - l_{ij})^2$$

となる．ある2つのオブジェクトの座標を (x_i, y_i)，(x_j, y_j) とすると，これらのオブジェクト間の距離 d_{ij} は，

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

となる．よって，これらのオブジェクト間に働くエネルギー E_{ij} は，

$$E_{ij} = \frac{1}{2} k \left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - l_{ij} \right)^2$$

である．これの x_i で偏微分したものを $\frac{\partial E_{ij}}{\partial x_i}$ とし，同様に y_i で偏微分したものを $\frac{\partial E_{ij}}{\partial y_i}$ とすると，それぞれ以下のようになる．

$$\frac{\partial E_{ij}}{\partial x_i} = k \left((x_i - x_j) - l_{ij} \frac{(x_i - x_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \right)$$

$$\frac{\partial E_{ij}}{\partial y_i} = k \left((y_i - y_j) - l_{ij} \frac{(y_i - y_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \right)$$

これを用いて，バネの力のベクトル $\mathbf{f}_{ij}$ は以下のようになる．

$$\mathbf{f}_{ij} = \left(\frac{\partial E_{ij}}{\partial x_i}, \frac{\partial E_{ij}}{\partial y_i} \right)$$

これは，2つのオブジェクト間にはられた1本のバネの力のベクトルである．本システムでは，全てのオブジェクト間に全結合の形でバネを張るため，オブジェクトが n 個存在する場合，1つのオブジェクトに張られるバネは $n-1$ 本である．オブジェクトに張られている全てのバネについて，それぞれバネの力のベクトルの総和は，そのオブジェクトに働く力の各座標成分に相当することになる．この計算を，各オブジェクトに対して行えば，全体のエネルギーの総計 E を最小にすることができる．つまり，任意の正定数 c を移動量調整数とし，

$$\delta_x = c \frac{\partial E_{ij}}{\partial x_i}$$

$$\delta_y = c \frac{\partial E_{ij}}{\partial y_i}$$

を移動量として定義する．cが大きければ，オブジェクトは速く移動し計算量も少ないが，動きが粗くなり，cが小さければ，オブジェクトの動きは滑らかになるが，遅く移動し計算量も多くなる．

実際のアルゴリズムは以下のようになる．

(n はオブジェクトの数)
(δ_x, δ_y は，各オブジェクトの移動量)
(δ'_x, δ'_y は，各オブジェクトの移動量の総和を求めるための変数)

```
begin
    for i = 1 to n do                          # 各頂点に対して処理を行う
    begin
        δ'x ← 0;
        δ'y ← 0;
        for オブジェクト i に設定されている各バネ do
        begin
            δx, δy を計算する
            δ'x += δx;
            δ'y += δy;
        end
        x ← x + δ'x                            # 新しい頂点の座標に更新する
        y ← y + δ'y                            # 新しい頂点の座標に更新する
    end
    全てのオブジェクトの重心を求め，重心が画面の中心にくるように座標を平行移動
end
```

これを繰り返し，各オブジェクトを配置していくことにより，全体のエネルギーの総計Eを減少させていくことができる．これは，オブジェクト間の実際の距離d_{ij}が理想距離l_{ij}に近づくような準最適解を求めることに相当する．実際に画面に表示するには，このままでは，画面からはみ出たり，特定の方向にオブジェクトが集まってしまう．そこで，座標を更新するごとにオブジェクト全体の重心をとり，それを表示画面の中央にくるように平行移動させる．それでも画面におさまらないオブジェクトは画面の端に置く．

4　2つの関係のシームレスな融合

時間的・因果的な関係と意味的な関係を表示する際，両者の関係を保ちつつ連続的に一方の観点から他方の観点に切り替える手法を考える．まず，このような手法の必要性について考察する．切り替えが断続的であり，一方の観点から他方の観点にスイッチするような切り替えでは，一方の観点で注目していたオブジェクトを，他方の観点に切り替わった時点で見失ってしまう．つまり，ユーザが着目したあるオブジェクトを中心とする思考の，観点間の連続性を保てないことになる．一方，切り替えが連続的であるならば，観点を切替える際も，注目していたオブジェクト

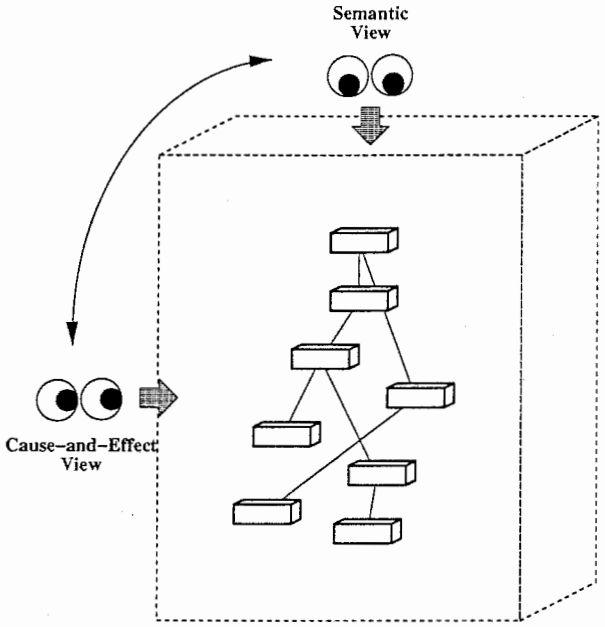

図 3: シームレスな可視化

を追随できるため，オブジェクトを中心とする思考の，観点間の連続性を保つことができる．

では，どのような融合方法が望ましいのか．たとえば単にオブジェクトがアニメーション的に移動することにより，連続的に 2 つの観点を切り替えるとするならば，その移動は特別な意味を持たないため，オブジェクトを見失わないという効果はあるものの，それ以外はユーザにとっては断続的な切り替えとほとんど変わりがないだろう．2 つの観点が相互に影響を及ぼしあいつつ，連続的な切り替えを行えるような方法だと，ユーザは，2 つの観点がどのように融合しているかを観察することにより，分類的意味を発見し，思考を刺激すると期待できる．

そこで本研究では，X-Z 平面上にマッピングされた意味的な関係に対し，Y 軸を時間軸として追加し，時間的・因果的な関係を可視化する方法を提案する（図 3参照）．この方法では，視線は常に原点方向を向いており，視点が X 軸を軸に 90 度回転することにより，時間的・因果的な関係の表示 (Cause-and-Effect View) から意味的な関係の表示 (Semantic View) に連続的に変化していく．

この方法を用いると，Cause-and-Effect View は，オブジェクトがマッピングされた 3 次元空間の X-Y 平面を表示することになるため，意味的な関係の X 軸方向でのオブジェクト間の距離は表示されることになる．また，意味的な関係の内，Z 軸方向でのオブジェクト間の距離についても，オブジェクトの大きさ（遠く（Z 軸負方向）にあるほど小さく，近く（Z 軸正方向）にあるほど大きく表示される）によって，ある程度は判明する．したがって Cause-and-Effect View は，単に意見の時間的・因果的関係を表示するのみならず，意味的な関係の配置が反映されたマッピングとなる．このため，意見の時間的・因果的な遠近を観察しつつ，同時に意味的な距離をみることができる．また，Semantic View では，意見間が時間的・因果的にリンクされているか否かが表示されるため，単なる意味的な類似度のみならず，時間的・因果的な関係をも知ることができる．

5　システム説明

ここでは，本システムがどのような機能を持っているかについて最初に説明し，次に，システムのモジュールの構成を述べる．

5.1　機能説明

5.1.1　オブジェクトの移動

機能　ユーザの選択したオブジェクトをマウスにより移動する．他のオブジェクトは選択されたオブジェクトの移動のたびに最適な位置が再計算され配置が行われる．

使い方　Semantic View のときに，あるオブジェクトの上にマウスカーソルを置き，左ボタンを押したまま移動すると，それにつれてオブジェクトが移動する．

5.1.2　オブジェクト回転

機能　5.1.1節で説明した「オブジェクトの移動」は，ユーザの選択したオブジェクトをマウスの移動によって移動させるというものであったが，本節の「オブジェクト回転」は，ユーザの選択したオブジェクトを自動的に回転移動させるというものである．

このような機能を付与したことにより，オブジェクト間の関係性がより直感的に把握できる．バネモデルだと，あるオブジェクトを動かすことにより，他のオブジェクトの動き（軌道や動きの速さなど）を観察することに意味がある．換言すると，動的な状態でのオブジェクト間の位置関係に意味を多く含むということである．これは，バネモデルによるオブジェクトマッピングが準最適解的であることに起因するものである．バネモデルでは，現在のマッピングにおいて，ある２つのオブジェクト間の距離が近いとき，それらの内容が本当は遠いにも関わらず，たまたま近くに配置されてしまうケースがあり得る．このような場合，片方のオブジェクトを動かし，もう一方のオブジェクトがそれを追いかけるような動きをするのか，それとも逆に離れるような動きをするのかを観察することにより，本当の内容の遠近がユーザに把握できる．

これを応用して，ユーザの使い方としては，着目した意見に対し，他の意見がどのような関係にあるのかを知りたいとき，着目した意見のオブジェクトを選択し，オブジェクト回転を行う，という方法がある．これにより，他のオブジェクトの動きが観察でき，静的な状態より直感的にオブジェクト間の関係が把握できる．

使い方　まず，ある一つのオブジェクトを選びマウスの左ボタンを押す．するとそのオブジェクトが赤の正方形に囲まれる．次にウィンドウの上段にあるメニューの一つに，“ Rotate” が設定されており，これを押すと図4のようなプルダウンメニューが現れる．オブジェクトを時計回りに回転させたい場合は “Clockwise” を，逆時計回りに回転させたい場合は “Anti-Clockwise” を選択する．すると，選択されたオブジェクトが画面の中心を中心に回転を始める．他のオブジェクトは，選択されたオブジェクトの移動のたびに最適な位置が再計算され配置を繰り返して行う．

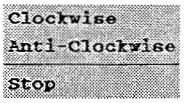

図 4: オブジェクト回転

図 5: 視点の回転移動

5.1.3 視点の回転移動

機能　視点の各軸に対する回転移動を行う．X 軸の回転角度が 0 度のときは，左の議論の構造を表示する画面は Cause-and-Effect View であり，X 軸の回転角度が 90 度のときは，Semantic View である．また，X 軸の回転角度が 0 度のときは，Y 軸を回転することにより，様々な角度から議論の時間的・因果的な関係を眺めることができる．また，X 軸の回転角度が 0 度のとき，Z 軸が 0 度なら議論は画面の上方から下方に向かって進むが，Z 軸が 90 度なら議論は左方から右方に向かって進む．Cause-and-Effect View と Semantic View の 2 つの観点間の移動のため，X 軸のみを回転させる場合がほとんどであろう．

使い方　本システムのウィンドウの向かって右方上段に，スライドバーが設定されている（図 5参照）．これを用いて視点を回転移動する．それぞれ “X Axis” （X 軸）,“Y Axis” （Y 軸）,“Z Axis” （Z 軸）を中心に視点が回転することを意味する．

5.1.4 発言の関係性指定

機能　2章で説明した，発言の関係性を指定する．

使い方　本システムのウィンドウの向かって右方中段に，エントリーボックスとラジオボタンが設定されている（図 6参照）．利用者は，発言しようとしている意見がどの意見に対するものなのか，その元の意見の番号をエントリーボックスに入力し，発言しようとしている意見が元の意見に対しどのような関係にあるかをラジオボタンによって指定する．ラジオボタンの色は，それぞれ，オブジェクトの色と対応している．

図 6: 発言の関係性指定

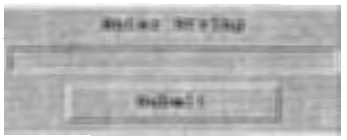

図 7: 発言

5.1.5 発言

機能 意見を発言する．これにより，発言が球体のオブジェクトとして左の議論の構造を表示する画面に追加される形で表示される．このとき，オブジェクトは Cause-and-Effect View ではオブジェクト群の最下部（Z 軸の回転が 0 度の場合）に追加される．発言の関係性の指定により，どの意見に対するフォローかがリンクという形で表現される．

使い方 本システムのウィンドウの向かって右方下段に，エントリーボックスとボタンが設定されている（図 7参照）．利用者は，エントリーボックスに発言しようとしている意見を入力した後，Submit ボタンを押す．

5.1.6 パラメータの設定

機能 バネモデルの主要なパラメータである，理想距離調整数，移動量調整数，バネ定数の 3 つを手動あるいは自動で設定する．各パラメータの意味については，3.2節，3.3節を参照されたい．

使い方 本システムのウィンドウの上段にあるメニューの一つに，“Parameter” が設定されており，これを押すと図 8のようなプルダウンメニューが現れる．手動で設定する場合，“Manual Setting” を選択し，エントリに希望の数値を入力する．自動で設定する場合，“Auto Setting” を選択する．この場合は，内部でオブジェクトの移動をシミュレートし，オブジェクトが中心に集まりすぎたり，逆に端に寄りすぎたりしないような値を計算し，パラメータを設定する．

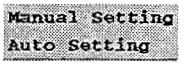

図 8: パラメータの設定

図 9: 光源の位置

5.1.7 光源の位置

機能 オブジェクトを照らす光源の位置を指定する．光源の位置によりオブジェクトが立体的でリアルに表現される．

使い方 本システムのウィンドウの上段にあるメニューの一つに，“Light” が設定されており，これを押すと図 9のようなプルダウンメニューが現れる．これは，光源の位置を設定するもので，例えば “X Positive” とあるのは，X 軸の正方向からの光源を意味する．

5.1.8 リンクの表示／非表示

機能 オブジェクト間に張られた，関係性を示すリンクを表示するか否かを指定する．

使い方 本システムのウィンドウの上段にあるメニューの一つに，“Line” が設定されており，これを押すと図 10のようなプルダウンメニューが現れる．これにより，リンクの表示／非表示を設定できる．

5.2 モジュール構成

図 11にモジュール構成を示す．あるユーザの発言は，一旦 Comunication Server[†]に対して送信され，会議に参加する全ユーザのクライアントホストの Coordinate Calculator に配信される．そこで，当該発言の配置が座標計算され，その座標データを Viewer に送り，表示がなされる．意味的な関係の配置については，その座標はクライアントで計算して表示する．ユーザは，Semantic View のときのみオブジェクトの移動などの操作を許される．したがって，ユーザによってオブジェクトの配置は異なる．

[†]1998 年 3 月 31 日の時点で未実装

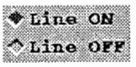

図 10: リンクの表示／非表示

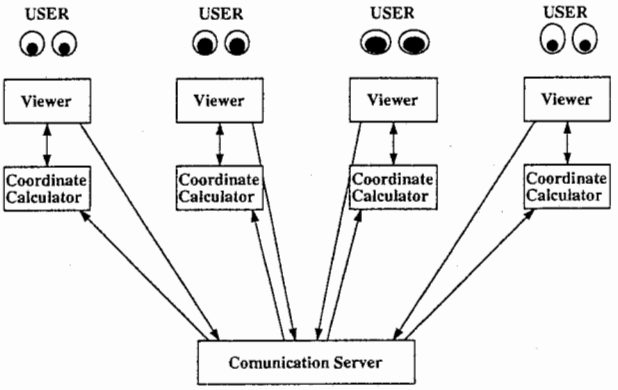

図 11: モジュールの構成

6 プログラム・ファイルの説明

本システムは，IRIX（シリコングラフィックス社の OS 環境）上で以下のツールを用いて構築した．したがって，本システムを起動の際には，予めこれらをインストールしておかなければならない．

OpenGL 3 次元グラフィックス用ライブラリ．

Tcl/Tk ユーザインターフェース構築のためのスクリプト言語．バージョンは，Tcl が 7.6，Tk が 4.2.

Togl OpenGL を Tk のウィジェットとして扱うためのインターフェース[‡]．

ChaSen 奈良先端科学技術大学院大学情報科学研究科自然言語処理学講座からリリースされているフリーの日本語形態素解析器[§]．

UNIX 上で，OpenGL の代わりに Mesa[¶]を用いても動作する筈であるが，1998 年 3 月 31 日の時点では未確認．

6.1 プログラムファイル（藤田作成分）

all.sh コンパイルとリンクのための C-Shell.

[‡]http://www.ssec.wisc.edu/ brianp/Togl.html 参照
[§]http://cactus.aist-nara.ac.jp/lab/nlt/chasen.html 参照
[¶]http://www.ssec.wisc.edu/ brianp/Mesa.html 参照

monkey.h プログラムで共通に使用する構造体や定数の設定を行っているヘッダファイル.

monkeyVector.c 1つのテキストを入力とし，キーワードベクトルを返す関数群. monkeyViewer.c から呼ばれる.

monkeyCoordinator.c 現在のオブジェクトの位置する座標とキーワードベクトルを元にして，各オブジェクトの座標を決定する関数群. monkeyViewer.c から呼ばれる.

monkeyViewer.c ユーザインターフェースの OpenGL ウィジェットの機能を定義する関数群.

monkeyViewer.tcl ユーザインターフェースの OpenGL ウィジェット以外の部分を定義する関数群.

filebox.tcl 出来合いの Tk ユーザインターフェースで，ファイルを選択する機能を持つ.

text.tcl オブジェクトに属するテキストを表示する.

6.2 ファイル

chasenResult ChaSen での解析の結果を納めるファイル.

linkedList 全テキストに出現するキーワードの一覧と，各テキストについて，そのキーワードが何回出現するかを記述したファイル.

relativeness テキスト同士の類似度を納めるファイル.

nodeDatabase 各オブジェクトのテキストや座標位置などの情報を保持するファイル.

tmp.txt オブジェクトに属するテキストを表示する際に，そのテキスト情報を納めるファイル.

6.3 プログラムファイル（外部提供分）

togl.c OpenGL を Tk のウィジェットとして扱うためのインターフェースを行うプログラム.

togl.h Togl を使ったプログラムで共通に使用する構造体や定数の設定を行っているヘッダファイル.

7 おわりに

本報では，電子会議環境において，議論内容の変遷や，議論の全体構造の把握を容易にし，ユーザの発想を支援するために，意見間の時間的・因果的な関係と意味的な関係を効果的に融合して可視化するシステムの構想についてまとめた．まず，意見間の時間的・因果的な関係を表示するモデルについて述べ，意見間の意味的な関係を表す手法としてバネモデルを定義し，2つの観点をシームレスに融合する方法について述べた．さらにこの提案を適用してプロトタイプシステムを試作した.

今後は以下の課題について考察し，プロトタイプシステムに反映していく.

システムの介入 会議に対してシステムが発言することにより介入を行う．このような支援はこれまでにもいくつか研究されており，ある程度の効果をあげてきた [西本 96] [Fujita 97]．本研究では，発言の関係の枠組を設けたことにより，システムによる支援の難易あるいは当否が明確になった．たとえば「同意」や「反論」はシステムが行うことは困難であろうと予測されるし妥当ではないと考えられる．一方，ある参加者の意見内に瀕出ではない言葉があった場合に，電子化辞書などの検索によりシステムが「説明」し，もし検索が失敗したら，その説明を求める「質問」をシステムが行う，という支援が考えられる．

さらに，ブレインストーミングなどのアイデアを出し合う会議では，新たな観点からのアイデアや，関連はあるが近すぎないアイデアなどが，参加者の発想を刺激することが知られている．そのようなアイデアを出せるような仕組みを考察し実装すれば，さらなる発想支援が行えるであろう．

facilitator 機能の付与 一定時間に意見が出ない場合に発言を促したり，同意か反対かが鮮明でない参加者に対して決断を求めたり，会議の終了予定時間が迫ったら結論をまとめる方向に誘導するなどの facilitator 機能をシステムに付与する．

参考文献

[田村 92] 田村淳：記号間の力学に基づく概念マップ生成システム SPRINGS，情報処理学会論文誌， Vol.33, No.4, pp465–470, (1992).

[杉本 94] 杉元雅則：複数他者の視点を可視化するシステムとその知的活動支援への応用に関する研究，東京大学大学院工学系研究科学位論文， (1997).

[Sumi 97] 角 康之, 西本 一志, 間瀬 健二：協同発想と情報共有を促進する対話支援環境における情報の個人化，電子情報通信学会論文誌, Vol.J80-D-I, No.7, pp.542-550, (1997).

[高杉 97] 高杉耕一：スプリングモデルを用いたアイデア触発のための思考支援システムの構築，北陸先端科学技術大学院大学情報科学研究科修士論文， (1997).

[斎藤 97] 斎藤一，柳川建久，前田隆：協調コミュニケーション環境における発想・理解支援機能の実現について， DiCoMo ワークショップ'97 論文集， pp.621–625, (1997).

[野間 97] 野間春生，角康之，宮里勉，間瀬健二： Haptic Interface による思考支援システム，第 13 回ヒューマン・インターフェース・シンポジウム論文集， (1997).

[阪田 92] 阪田史郎：グループウェアの実現技術，ソフト・リサーチ・センター， (1992).

[渡辺 92] 渡辺理：電子会議における発言間の関係についての一考察，情報処理学会研究報告, Vol.92, No.GW-3, pp.57–64, (1992).

[渡部 91] 渡部勇：発散的思考支援システム： Keyword Associator，計測自動制御学会合同シンポジウム論文集， pp.411–418, (1991).

[松本 97] 松本裕治，北内啓，山下達雄，今一修，今村友明：日本語形態素解析システム『茶筌』 version 1.0 使用説明書，NAIST Technical Report, NAIST-IS-TR97007, (1997).

[三末 95] 三末和男，杉山公造：思考支援システムの評価法およびD-ABDUCTORの評価実験について，第3回『発想支援ツール』シンポジウム講演論文集，(1995).

[西本 96] 西本一志，角康之，間瀬建二：一参加者として対話に加わる対話活性化エージェント，信学技報 TL96-7, (1996).

[Fujita 97] Kunihiko Fujita, Susumu Kunifuji : A Realization of a Reflection of Personal Information on Distributed Brainstorming Environment, Proc. of WWCA '97, pp.B-2-3-1 – B-2-3-16, (1997). (to appear in Springer-Verlag Lecture Notes in Computer Science, Vol.1274, pp.166–181, Edited by T. Masuda, Y. Masunaga, M. Tsukamoto : Worldwide Computing and its Applications '97.)

```
#
# all.sh is for compile and link of monkey.   by Fujita, Kunihiko  Nov.10 '97
#

echo "gcc -g -c -O -I/home/kunihiko/include -I/usr/local/include -I./libaux monk
eyViewer.c"
gcc -g -c -O -I/home/kunihiko/include -I/usr/local/include -I./libaux monkeyView
er.c

echo "gcc -g -c -O monkeyVector.c"
gcc -g -c -O monkeyVector.c

echo "gcc -g -c -O monkeyCoordinator.c"
gcc -g -c -O monkeyCoordinator.c

echo "gcc -g -c -O -I/home/kunihiko/include -I/usr/local/include  togl_jp.c"
gcc -g -c -O -I/home/kunihiko/include -I/usr/local/include  togl_jp.c

echo pe"gcc -g -L/home/kunihiko/lib -L/usr/local/lib -L./libaux monkeyViewer.o m
onkeyVector.o monkeyCoordinator.o togl_jp.o -ltk4.2jp -ltcl7.6jp -laux -lGLU -lG
L -L/usr/X11/lib -lX11 -lXmu -lXext -lm -lchasen -o monkey"
gcc -g -L/home/kunihiko/lib -L/usr/local/lib -L./libaux monkeyViewer.o monkeyVec
tor.o monkeyCoordinator.o togl_jp.o -ltk4.2jp -ltcl7.6jp -laux -lGLU -lGL -L/usr
/X11/lib -lX11 -lXmu -lXext -lm -lchasen -o monkey

echo "/bin/rm monkeyCoordinator.o"
/bin/rm monkeyCoordinator.o

echo "/bin/rm monkeyVector.o"
/bin/rm monkeyVector.o

echo "/bin/rm monkeyViewer.o"
/bin/rm monkeyViewer.o

echo "/bin/rm togl_jp.o"
/bin/rm togl_jp.o
```

```
/*
 * monkey.h by  Fujita, Kunihiko  Nov./12/1997
 */

/* 対応できるオブジェクトの数 */
#define OBJLIMIT 64

/* 円周率 */
#define PI 3.1415926535

/* 2乗を求める */
#define MULT2(X) ((X)*(X))

/* ofpsw での ortho, frustum, perspective 判定のため */
#define ORTHO 0
#define FRUSTUM 1
#define PERSPECTIVE 2

/* ON OFF */
#define OFF 0
#define ON 1

/* オブジェクト */
struct nodeDB{
  float coordinate[3];          /* XYZ座標 */
  char string[4096];             /* 文章（発言） */
  int color;               /* 色（オリジナルオブジェクトとの関係） */
  struct nodeDB *original;      /* オリジナルオブジェクトへのポインタ */
  int original_no;              /* オリジナルオブジェクトのナンバー */
  struct nodeDB *next;
};
typedef struct nodeDB ndb;

/* キーワード */
struct keywordDB{
  char  keyword[32];            /* キーワード */
  float score[OBJLIMIT];             /* 各テキストのKWのスコア */
  struct keywordDB *next;
};
typedef struct keywordDB kwdb;

/* Color default setting for using glColor3fv(). */
static float BLACKv[3]   = {0.0, 0.0, 0.0};
static float REDv[3]     = {1.0, 0.0, 0.0};
static float GREENv[3]   = {0.0, 1.0, 0.0};
static float BLUEv[3]    = {0.0, 0.0, 1.0};
static float YELLOWv[3]  = {1.0, 1.0, 0.0};
static float MAGENTAv[3] = {1.0, 0.0, 1.0};
static float CYANv[3]    = {0.0, 1.0, 1.0};
static float WHITEv[3]   = {1.0, 1.0, 1.0};
/* Color default setting for using alpha value. */
static float BLACKav[4]   = {0.0, 0.0, 0.0, 1.0};
static float REDav[4]     = {1.0, 0.0, 0.0, 1.0};
static float GREENav[4]   = {0.0, 1.0, 0.0, 1.0};
static float BLUEav[4]    = {0.0, 0.0, 1.0, 1.0};
static float YELLOWav[4]  = {1.0, 1.0, 0.0, 1.0};
static float MAGENTAav[4] = {1.0, 0.0, 1.0, 1.0};
static float CYANav[4]    = {0.0, 1.0, 1.0, 1.0};
static float WHITEav[4]   = {1.0, 1.0, 1.0, 1.0};
/* Color Index. */
static float colorIndex[8][4] = {
  {0.0, 0.0, 0.0, 1.0}, {1.0, 0.0, 1.0, 1.0},
  {0.0, 0.0, 1.0, 1.0}, {0.0, 1.0, 0.0, 1.0},
  {1.0, 0.0, 0.0, 1.0}, {0.0, 1.0, 1.0, 1.0},
  {1.0, 1.0, 0.0, 1.0}, {1.0, 1.0, 1.0, 1.0}
};
```

```
/*
 * This program is for spring model.  by Fujita, Kunihiko  Nov. 13 '97
 */

#include "monkey.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include <math.h>

/* 重み付け前のキーワードベクトル */
kwdb *kwstart;
/* 重み付け後のキーワードベクトル */
kwdb *wkwstart;
/* オブジェクトの総数 */
int num;
/* キーワードリンクトリストをクリアして再構築するのか追加するのかのスイッチ */
int clearswitch;


/* キーワードリンクトリストのノードを確保し引数セット */
kwdb *malloc_kwdb(char *kw)
{
  kwdb *info;
  int i = 0;

  if(!(info = (kwdb *)malloc(sizeof(kwdb)))){
    printf("\n out of memory");
    return;
  }
  strcpy(info->keyword, kw);
  for(i = 0; i < OBJLIMIT; i++)
    info->score[i] = 0.0;
  info->next = NULL;

  return info;
}


/* キーワードリンクトリストのload処理 */
void kwll_load(FILE *fp)
{
  char buf[2048], kwbuf[32];
  kwdb *info, *ftmp, *fbuf;
  int i = 0, j = 0, len = 0;

  /* 新規にロードするのなら，キーワードリンクトリストの領域を解放する */
  if(clearswitch){
    ftmp = kwstart;
    while(ftmp){
      fbuf = ftmp->next;
      free(ftmp);
      ftmp = fbuf;
    }
    kwstart = NULL;
  }

  if(fgets(buf, 2048, fp)){
    len = strlen(buf);
    while(j < len){
      while(!(buf[i] == ';')){
        kwbuf[i] = buf[j];
        i++;
        j++;
      }
      if(!kwstart){
        kwstart = malloc_kwdb(kwbuf);
        info = kwstart;
      } else{
        info->next = malloc_kwdb(kwbuf);
        info = info->next;
      }
      i = 0;
      j++;
    }
  }
}


/* スコアのload処理 */
score_load(FILE *fp)
{
  char buf[2048], kwbuf[32];
  kwdb *info;
  int i = 0, j = 0, k =0, len = 0;

  info = kwstart;

  k = 0;

  while(fgets(buf, 2048, fp)){
    len = strlen(buf);
    while(j < len){
      while(!(buf[i] == ';')){
        kwbuf[i] = buf[j];
        i++;
        j++;
      }
      info->score[k] = atof(kwbuf);
      info = info->next;
      i = 0;
      j++;
    }
    k++;
  }
```

```
}


/* キーワードとスコアを納めたファイルのloadを行う */
load_llist(FILE *fp)
{
  kwll_load(fp);
  score_load(fp);
}


/* キーワードとスコアのリンクトリストをファイルにsaveする */
save_llist(FILE *fp)
{
  kwdb *info;
  int i = 0;

  /* キーワードのsave処理 */
  info = kwstart;
  while(info){
    fprintf(fp, "%s;", info->keyword);
    info = info->next;
  }

  fprintf(fp, "\n");

  /* スコアのsave処理 */
  info = kwstart;
  while(i <= num){
    while(info){
      fprintf(fp, "%f;", info->score[i]);
      info = info->next;
    }
    i++;
    info = kwstart;
    fprintf(fp, "\n");
  }
}


/* 与えられたテキストからキーワードを抽出する */
int get_kw(char *buf, char *kwbuf)
{
  int wnum = 0, i = 0, len = 0;

  len = strlen(buf);

  while(wnum < len){
    if(!(strncmp(&(buf[wnum]), "名詞", 4)) ||
       !(strncmp(&(buf[wnum]), "未定義", 6))){
      while(!(buf[i] == ' ')){
        kwbuf[i] = buf[i];
        i++;
      }
      kwbuf[i] = '\0';
      return(1);
    } else{
      wnum++;
    }
  }
  return(0);
}


/* 入力テキストを形態素解析して，KWベクトル重み付け前を作成 */
make_table(char *txt)
{
  FILE *fp;
  char buf[256], kwbuf[32];
  kwdb *info, *prev;
  int flg = 0;

  if(!(fp = fopen("chasenResult", "w+"))){
    printf("Cannot open File:chasenResult.\n");
    return;
  }

  /* chasenによる形態素解析 */
  chasen_sparse(txt, fp);
  fseek(fp, 0, SEEK_SET);

  while(fgets(buf, 256, fp)){
    if(get_kw(buf, kwbuf)){
      if(!kwstart){
        kwstart = malloc_kwdb(kwbuf);
        ++(kwstart->score[0]);
      } else{
        info = kwstart;
        flg = 0;
        while(info){
          if(!(strcmp(info->keyword, kwbuf))){
            /* キーワードが合致したときの処理 */
            ++(info->score[num]);
            flg++;
          }
          prev = info;
          info = info->next;
        }
        /* キーワードが最後まで合致しなかったときの処理 */
        if(!flg){
          prev->next = malloc_kwdb(kwbuf);
          ++(prev->next->score[num]);
        }
      }
    }
  }
```

```
  fclose(fp);
}


/* ある1つのキーワードが全テキスト中で現れた回数 */
kwdb *sum_kw()
{
  kwdb *info, *sumstart, *suminfo;
  int i = 0;

  sumstart = NULL;
  info = kwstart;

  while(info){
    if(!sumstart){
      sumstart = malloc_kwdb(info->keyword);
      for(i = 0; i <= num; i++){
        sumstart->score[0] += info->score[i];
      }
      suminfo = sumstart;
    } else{
      suminfo->next = malloc_kwdb(info->keyword);
      suminfo = suminfo->next;
      for(i = 0; i <= num; i++){
        suminfo->score[0] += info->score[i];
      }
    }
    info = info->next;
  }

  return sumstart;
}


/* 全てのキーワードが全テキスト中で現れた回数 */
float sum_allkw(kwdb *sumstart)
{
  kwdb *suminfo;
  float sumall = 0;

  suminfo = sumstart;

  while(suminfo){
    sumall += suminfo->score[0];
    suminfo = suminfo->next;
  }

  return sumall;
}


/* ある1つのキーワードがある1つのテキスト中で現れる確率 */
kwdb *pit_kw(kwdb *sumstart)
{
  kwdb *info, *pitstart, *pitinfo, *suminfo;
   int i = 0;

  pitstart = NULL;
  info = kwstart;

  while(info){
    if(!pitstart){
      pitstart = malloc_kwdb(info->keyword);
      for(i = 0; i <= num; i++){
        pitstart->score[i] = info->score[i] / sumstart->score[0];
      }
      suminfo = sumstart;
      pitinfo = pitstart;
    } else{
      pitinfo->next = malloc_kwdb(info->keyword);
      pitinfo = pitinfo->next;
      suminfo = suminfo->next;
      for(i = 0; i <= num; i++){
        pitinfo->score[i] = info->score[i] / suminfo->score[0];
      }
    }
    info = info->next;
  }

  return pitstart;
}


/* ある1つのキーワードが全てのテキスト中で現れる確率 */
kwdb *pi_allkw(kwdb *sumstart, float sumall)
{
  kwdb *piallstart, *piallinfo, *suminfo;
  int i = 0;

  piallstart = NULL;
  suminfo = sumstart;

  while(suminfo){
    if(!piallstart){
      piallstart = malloc_kwdb(suminfo->keyword);
      for(i = 0; i <= num; i++){
        piallstart->score[0] = suminfo->score[0] / sumall;
      }
      piallinfo = piallstart;
    } else{
      piallinfo->next = malloc_kwdb(suminfo->keyword);
      piallinfo = piallinfo->next;
      for(i = 0; i <= num; i++){
        piallinfo->score[0] = suminfo->score[0] / sumall;
      }
```

```
    }
    suminfo = suminfo->next;
  }

  return piallstart;
}


/* ダイバージェンス計算 */
divergence_calc(kwdb *pitstart, kwdb *piallstart)
{
  kwdb *pitinfo, *piallinfo, *tmp;
  int i = 0;

  pitinfo = pitstart;
  piallinfo = piallstart;

  while(pitinfo){
    for(i = 0; i <= num; i++){
      /*
       * 出現確率が0のときは，そのままとする
       *  (logをとると無限小となり，動きがおかしい)
       */
      if(pitinfo->score[i])
        pitinfo->score[i] *= log((pitinfo->score[i] / piallinfo->score[0]) + 1);
    }
    pitinfo = pitinfo->next;
    piallinfo = piallinfo->next;
  }
}


/* スカラー値計算 */
float scalar_calc(kwdb *pitstart, int i)
{
  kwdb *pitinfo;
  float sum = 0;

  pitinfo = pitstart;

  while(pitinfo){
    sum += MULT2(pitinfo->score[i]);
    pitinfo = pitinfo->next;
  }

  return sqrt(sum);
}


/* 正規化計算 */
normalize_calc(kwdb *pitstart)
{
  kwdb *pitinfo;
  int i = 0;
  float tmpsum = 0;

  for(i = 0; i <= num; i++){
    pitinfo = pitstart;
    tmpsum = 0;
    tmpsum = scalar_calc(pitstart, i);
    while(pitinfo){
      pitinfo->score[i] /= tmpsum;
      pitinfo = pitinfo->next;
    }
  }
}


/* KWベクトル重み付けを行う */
weight_table()
{
  kwdb *sumstart, *pitstart, *piallstart;
  float sumall = 0;

  /* ある1つのキーワードが全テキスト中で現れた回数 */
  sumstart = sum_kw();

  /* 全てのキーワードが全テキスト中で現れた回数 */
  sumall = sum_allkw(sumstart);

  /* ある1つのキーワードがある1つのテキスト中で現れる確率 */
  pitstart = pit_kw(sumstart);

  /* ある1つのキーワードが全てのテキスト中で現れる確率 */
  piallstart = pi_allkw(sumstart, sumall);

  /* ダイバージェンス計算 */
  divergence_calc(pitstart, piallstart);

  /* 正規化計算 */
  normalize_calc(pitstart);

  wkwstart = pitstart;
}


/* テキスト間の類似度を求める */
relative_calc()
{
  FILE *fp;
  kwdb *info;
  float relative[num + 1][num + 1];
  int i = 0, j = 0;

  for(i = 0; i <= num; i++){
    for(j = 0; j <= num; j++){
```

```
      info = wkwstart;
      relative[i][j] = 0;
      /*
       * 高速化検討ポイント
       * i == j のときは，類似度は1である．
       * たとえば relative[0][1] と relative[1][0] は同じ値になる筈である．
       */
      while(info){
        relative[i][j] += (info->score[i] * info->score[j]);
        info = info->next;
      }
    }
  }

  /*
   * 中間ファイルに類似度を書き込む．
   * 中間ファイルを用いないことによる高速化を後に検討する．
   */

  if(!(fp = fopen("relativeness", "w+"))){
    printf("Cannot open File:relativeness.\n");
    return;
  }

  for(i = 0; i <= num; i++){
    for(j = 0; j <= num; j++){
      fprintf(fp, "%d;%d;%f\n", i, j, relative[i][j]);
    }
  }

  fclose(fp);
}


void kw_extract(char *txt, int getnum, int sw)
{
  FILE *fp;

  num = getnum;
  if((getnum == 0) && (sw == 1)){
    clearswitch = 1;
  } else{
    clearswitch = 0;
  }

  if(!(fp = fopen("linkedList", "w+"))){
    printf("Cannot open File:linkedList.\n");
    return;
  }

  /* KWベクトルファイルをload */
  load_llist(fp);

  /* 入力テキストを形態素解析して，KWベクトル重み付け前を作成 */
  make_table(txt);

  /* KWベクトル重み付けを行う */
  weight_table();

  /* テキスト間の類似度を求める */
  relative_calc();

  /* KWベクトルファイルにsave */
  fseek(fp, 0, SEEK_SET);
  save_llist(fp);
  fclose(fp);
}
```

```
/*
 * monkeyCoordinator.c is for spring model.  by Fujita, Kunihiko  Nov. 19 '97
 */

#include "monkey.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include <math.h>

/* 他ファイルの変数宣言 */
extern int num;
extern double viewLeft;
extern double viewRight;
extern double viewBottom;
extern double viewTop;

/* 類似度 */
float relative[OBJLIMIT][OBJLIMIT];
/* 理想距離調整数 */
/* nodeDatabase オリジナル idealAdjust = 42.0 */
/* nodeDatabase fj.soc.copyright_2 idealAdjust = 0.01 */
extern float idealAdjust;
/* 移動量調整数 */
/* nodeDatabase オリジナル moveAdjust = 1.0 */
/* nodeDatabase fj.soc.copyright_2 moveAdjust = 0.001 */
extern float moveAdjust;
/* バネ定数 */
/* nodeDatabase オリジナル springFixNum = 0.1 */
/* nodeDatabase fj.soc.copyright_2 springFixNum = 0.001 */
extern float springFixNum;
/* 上記のパラメータを自動で設定するか否かのスイッチ */
int autosw = OFF;

/* 類似度をファイルから読み込む */
void relative_get()
{
  FILE *fp;
  float tmprel = 0;
  int tmpi = 0, tmpj = 0;
  if(!(fp = fopen("relativeness", "r"))){
    printf("Cannot open File:relativeness.\n");
    return;
  }
  while(fscanf(fp, "%d;%d;%f\n", &tmpi, &tmpj, &tmprel) != EOF)
    relative[tmpi][tmpj] = tmprel;
  fclose(fp);
}


/* 配列の中で最大値を求める*/
float fmax(float *point)
{
  float tmpmax = 0;
  int i = 0;

  tmpmax = point[0];

  for(i = 0; i <= num; i++){
    if(tmpmax < point[i])
      tmpmax = point[i];
  }

  return tmpmax;
}


/* 配列の中で最小値を求める*/
float fmin(float *point)
{
  float tmpmin = 0;
  int i = 0;

  tmpmin = point[0];

  for(i = 0; i <= num; i++){
    if(tmpmin > point[i])
      tmpmin = point[i];
  }

  return tmpmin;
}


/* 座標計算を行う */
void coordinate_calc(ndb *lstart)
{
  float coordinateX[OBJLIMIT], coordinateZ[OBJLIMIT], newcoX[OBJLIMIT], newcoZ[O
BJLIMIT];
  float dij = 0, lij = 0, movexx = 0, movezz = 0;
  float maxx = 0, minx = 0, maxz = 0, minz = 0, shiftx = 0, shiftz = 0;
  int i = 0, j = 0, counter = 0;
  ndb *info;

  /* ノードDBの座標変数から計算用変数にコピー */
  counter = 0;
  info = lstart;
  while(info){
    coordinateX[counter] = info->coordinate[0];
    coordinateZ[counter] = info->coordinate[2];
    newcoX[counter] = 0;
    newcoZ[counter] = 0;
    counter++;
    info = info->next;
  }
```

```
/* 類似度をファイルから読み込む */
relative_get();

for(i = 0; i <= num; i++){
  movexx = 0;
  movezz = 0;
  for(j = 0; j <= num; j++){
    if(i != j){
      /* i-j間の実際の距離を求める */
      dij = sqrt(MULT2(coordinateX[i] - coordinateX[j])
                 + MULT2(coordinateZ[i] - coordinateZ[j]));
      /* i-j間の理想距離を求める
      lij = idealAdjust * 1 / relative[i][j]; */
      lij = idealAdjust * log(1 / relative[i][j]);
      /* 移動量（偏微分値）を求める partial */
      movexx += moveAdjust * springFixNum * ((coordinateX[i] - coordinateX[j])
-
                        ((lij / dij) * (coordinateX[i] - coordinateX[j])));
      movezz += moveAdjust * springFixNum * ((coordinateZ[i] - coordinateZ[j])
-
                        ((lij / dij) * (coordinateZ[i] - coordinateZ[j])));
    }
  }
  /*
   * 上の計算後に座標を更新するが,
   * その更新後の座標を用いて新たに他の頂点の座標を計算していくのか,
   * それとも,
   * 初期の座標を用いて全ての新たな座標を計算するのか.
   * ここでは，とりあえず後者の手法を用いている.
   * 準最適解なので，これで正しい筈.
   */
  newcoX[i] = coordinateX[i] + movexx;
  newcoZ[i] = coordinateZ[i] + movezz;
}

/* 重心を求めて，各頂点をシフトする */
maxx = fmax(newcoX);
minx = fmin(newcoX);
maxz = fmax(newcoZ);
minz = fmin(newcoZ);

/* 各頂点の平行移動量を算出 */
shiftx = (minx + maxx) / 2;
shiftz = (minz + maxz) / 2;

/* 重心分だけ各頂点の平行移動 */
for(i = 0; i <= num; i++){
  newcoX[i] -= shiftx;
  newcoZ[i] -= shiftz;
}

/* 計算用の変数からノードDBの座標変数にコピー */
counter = 0;
info = lstart;
while(info){
  if(newcoX[counter] > viewRight)
    newcoX[counter] = viewRight - 20;
  if(newcoX[counter] < viewLeft)
    newcoX[counter] = viewLeft + 20;
  if(newcoZ[counter] > viewTop)
    newcoZ[counter] = viewTop - 10;
  if(newcoZ[counter] < viewBottom)
    newcoZ[counter] = viewBottom + 10;
  info->coordinate[0] = newcoX[counter];
  info->coordinate[2] = newcoZ[counter];
  counter++;
  info = info->next;
}

}
```

```
/*
 * monkeyViewer.c   by FUJITA, kunihiko  Feb/7/97
 *
 * Togl - a Tk OpenGL widget
 * Copyright (C) 1996-1997  Brian Paul and Ben Bederson
 */

#include "togl.h"
#include "monkey.h"
#include "aux.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <GL/gl.h>
#include <GL/glu.h>

/*
 * The following variable is a special hack that is needed in order for
 * Sun shared libraries to be used for Tcl.
 */
extern int matherr();
int *tclDummyMathPtr = (int *)matherr;
/*
 * font setting
 */
static GLuint FontBase;

/* 他ファイルの関数宣言 */
extern void kw_extract(char *, int, int);
extern void coordinate_calc(ndb *);

/* 視界の回転角度の設定 */
static float xAngle = 0.0, yAngle = 0.0, zAngle = 0.0;
/* マウスポインタの位置 */
static float xPoint = 0.0, yPoint = 0.0, zPoint = 0.0;
/* 視体積の設定 */
GLdouble viewLeft = 0.0, viewRight = 0.0;
GLdouble viewBottom = 0.0, viewTop = 0.0;
GLdouble viewNearO = 0.0, viewFarO = 0.0;
GLdouble viewNearF = 0.0, viewFarF = 0.0;
GLdouble viewNearP = 0.0, viewFarP = 0.0;
GLdouble fovyP = 60.0;
/* ノードデータベースのリンクトリストの最初と最後のポインタ */
ndb *start = NULL;
ndb *last = NULL;
/* Bl-Motion時の操作対象のノードを指す */
ndb *opObj;
/* いくつのノードがDBに登録されているかのカウンタ */
static int objCtr = 0;
/* 表示できるオブジェクトの総数 y座標を決めるのに使用 */
int objAllNum = 20;
/* ノードを表すオブジェクトの中心から大きさ（球の半径） */
static float sradius = 12.0;
/* Ortho(0) か Frustum(1) か Perspective(2) のどちらを用いるかのスイッチ */
static int ofpsw = ORTHO;
/* lights setting */
int l0 = 1, l1 = 1, l2 = 0, l3 = 0, l4 = 0, l5 = 0;
/* 画面の縦横 */
int width;
int height;
/* 関係線を描くかどうかのスイッチ */
int linesw = 1;
/* 正方形で囲むオブジェクト for 内容表示 */
ndb *encObjC = NULL;
/* 正方形で囲むオブジェクト for 洗濯 */
ndb *encObjR = NULL;
/* 関数宣言 */
void enclose_object();
float calcRotate(float);
/* tmp */
int timersw = OFF;
/* 視点の座標 */
static float viewX = 0.0, viewY = 0.0, viewZ = 0.0;
/*
 * パラメータ
 *  （以下の3つのパラメータは，オブジェクトが広がり過ぎるときは値を小さくする）
 */
/* 理想距離調整数 */
float idealAdjust = 42.0;
/* 移動量調整数 */
float moveAdjust = 1.0;
/* バネ定数 */
float springFixNum = 0.1;



/* Set data in orderA to data in orderB. */
void orderSetF(float *varB, float *varA, int num)
{
  int i;

  for(i = 0; i < num; i++)
    varB[i] = varA[i];
}

/*
 * Togl widget create callback.  This is called by Tcl/Tk when the widget has
 * been realized.  Here's where one may do some one-time context setup or
 * initializations.
 */
void create_cb(struct Togl *togl)
{
  FontBase = Togl_LoadBitmapFont(togl, "k14");

  if(!FontBase){
```

```
    printf("Couldn't load font!\n");
    exit(1);
  }
}

/*
 * Togl widget reshape callback.  This is called by Tcl/Tk when the widget
 * has been resized.  Typically, we call glViewport and perhaps setup the
 * projection matrix.
 */
void reshape_cb(struct Togl *togl)
{
  int width = Togl_Width(togl);
  int height = Togl_Height(togl);
  float aspect = (float)width / (float)height;

  glViewport(0, 0, width, height);

  /* Set up projection transform */
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();

  viewLeft = -(width/2);
  viewRight = width/2;
  viewBottom = -(height/2);
  viewTop = height/2;
  if(width > height){
    viewNearO = -(width/2);
    viewFarO = width/2;
    viewNearF = width/2;
    viewFarF = viewNearF + width;
    viewNearP = sqrt(MULT2(width) - MULT2(width/2));
    viewFarP = viewNearP + width;
  } else{
    viewNearO = -(height/2);
    viewFarO = height/2;
    viewNearF = height/2;
    viewFarF = viewNearF + height;
    viewNearP = sqrt(MULT2(height) - MULT2(height/2));
    viewFarP = viewNearP + height;
  }

  if(ofpsw == ORTHO){
    glOrtho(viewLeft, viewRight, viewBottom, viewTop, viewNearO, viewFarO);
  } else if(ofpsw == FRUSTUM){
    glFrustum(viewLeft, viewRight, viewBottom, viewTop, viewNearF, viewFarF);
  } else{
    gluPerspective(fovyP, aspect, viewNearP, viewFarP);
  }

  /* Change back to model view transform for rendering */
  glMatrixMode(GL_MODELVIEW);
}


/* print string */
static void print_string(const char *s)
{
  int i = 0, cl, cu;
  GLuint *str;
  int numstr = strlen(s) + 1;

  str = (GLuint *)alloca(sizeof(GLuint) * numstr);

  while(*s){
    if((*s & 0x80) != 0){
      cu = *s++ & 0x7f;
      cl = *s++ & 0x7f;
      str[i++] = cu*256 + cl;
    } else{
      s++;
    }
  }

  glPushAttrib(GL_LIST_BIT);
  glListBase(FontBase);
  glCallLists(i, GL_UNSIGNED_INT, (GLuint *)str);
  glPopAttrib();
}

/* ORTHO における視点座標の計算 */
void pointOfView()
{
  /* 各軸回転時に考慮する半径 */
  float rX = 0.0, rY = 0.0, rZ = 0.0;
  /* XYZ座標 */
  float cXx = 0.0, cYx = 0.0, cZx = 0.0;
  float cXy = 0.0, cYy = 0.0, cZy = 0.0;
  float cXz = 0.0, cYz = 0.0, cZz = 0.0;
  /* 各軸回転角度（単位:ラジアン） */
  float aX = 0.0, aY = 0.0, aZ = 0.0;
  float radian = 180.0 / PI;
  /* Z軸回転時のイニシャル角度 */
  float aZinit = 0.0;

  /* 各軸回転角度の設定（単位:度→ラジアン） */
  aX = xAngle / radian;
  aY = yAngle / radian;
  aZ = zAngle / radian;

  /* X軸の回転 */
  rX = viewTop;
  cXx = 0.0;
  cYx = rX * sin(aX);
  cZx = rX * cos(aX);

  /* Y軸の回転 */
```

```
  rY = cZx;
  cXy = -(rY * sin(aY));
  cYy = cYx;
  cZy = rY * cos(aY);

  /* z軸の回転 */
  rZ = sqrt(MULT2(cXy) + MULT2(cYy));
  aZinit = calcRotate(cYy / rZ);
  cXz = -(rZ * cos(aZinit + aZ));
  cYz = rZ * sin(aZinit + aZ);
  cZz = cZy;

  viewX = cXz;
  viewY = cYz;
  viewZ = cZz;
}

/* 視点平面からオブジェクトの距離を求める */
float calcLengthPseudo(float coX, float coY, float coZ)
{
  float pseudoLength = 0.0;

  pseudoLength = (MULT2(viewX) - (viewX * coX) + MULT2(viewY) - (viewY * coY) +
MULT2(viewZ) - (viewZ * coZ)) / sqrt(MULT2(viewX) + MULT2(viewY) + MULT2(viewZ))
;

  return pseudoLength;
}

/* 視点からオブジェクトの距離を求める */
float calcLengthReal(float coX, float coY, float coZ)
{
  float realLength = 0.0;

  realLength = sqrt(MULT2(viewX - coX) + MULT2(viewY - coY) + MULT2(viewZ - coZ)
);

  return realLength;
}

/* オブジェクトの大きさ指数を求める */
float sizeIndex(float ol, float pl)
{
  float index = 0.0;

  index = 1.0 + ((ol - pl) / ol * 0.5);

  return index;
}


/* ノードをあらわすオブジェクトを描き，オブジェクト間に線分を描く */
void draw_object()
{
  /* counter */
  int k = 0;
  /* DB内で，描く対象のノード情報のある位置を指す */
  ndb *info;
  /* マテリアル放射光成分 */
  GLfloat mat_emission[4] = {0.0, 0.0, 0.0, 1.0};
  /* マテリアル鏡面反射光成分 */
  GLfloat mat_specular[4] = {1.0, 1.0, 1.0, 1.0};
  /* マテリアル鏡面指数 */
  GLfloat mat_shininess = 100.0;
  /* 視点からオブジェクトの距離 */
  float realLength = 0.0;
  /* 視点平面からオブジェクトの距離 */
  float pseudoLength = 0.0;
  /* 視点可動球面の半径 */
  float originalLength = viewTop;

  /* バッファーのクリアー*/
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  /* 鏡面反射光成分のセット */
  glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
  /* 鏡面指数のセット */
  glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);
  /* 放射光成分のセット */
  glMaterialfv(GL_FRONT, GL_EMISSION, mat_emission);

  info = start;

  while(info){
    glPushMatrix();
    /* 描画位置の移動 */
    glTranslatef(info->coordinate[0], info->coordinate[1], info->coordinate[2]);
    /* 視点からオブジェクトの距離を求める */
    realLength = calcLengthReal(info->coordinate[0], info->coordinate[1], info->
coordinate[2]);
    /* 視点平面からオブジェクトの距離を求める */
    pseudoLength = calcLengthPseudo(info->coordinate[0], info->coordinate[1], in
fo->coordinate[2]);
    /* オブジェクト球の拡散光の色の設定 */
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, colorIndex[info->color]);
    auxSolidSphere(sizeIndex(originalLength, pseudoLength) * sradius);
    glPopMatrix();

    /* 描いたオブジェクトが2個目以降 かつ
       linesw が真ならば,
       前のオブジェクトとの間に線分を描く */
    if(k > 0 && linesw){
      /* 線分の材質の環境&拡散光の色の設定 */
      glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, WHITEav);
      glBegin(GL_LINES);
      glVertex3fv(info->original->coordinate);
```

```
      glVertex3fv(info->coordinate);
      glEnd();
    }

    /* 選択されたオブジェクトがあるなら,
       その周りを正方形で囲む */
    if(encObjC || encObjR)
      enclose_object();

    glFlush();
    k++;
    info = info->next;
  }
}

/* 光源の設定 */
void light_init(void)
{
  /* 光源の位置 */
  GLfloat light_position0[] = {1.0, 0.0, 0.0, 0.0};
  GLfloat light_position1[] = {0.0, 1.0, 0.0, 0.0};
  GLfloat light_position2[] = {0.0, 0.0, 1.0, 0.0};
  GLfloat light_position3[] = {-1.0, 0.0, 0.0, 0.0};
  GLfloat light_position4[] = {0.0, -1.0, 0.0, 0.0};
  GLfloat light_position5[] = {0.0, 0.0, -1.0, 0.0};
  /* 環境光 */
  GLfloat light_ambient[] = {0.0, 0.0, 0.0, 1.0};
  /* 拡散光 */
  GLfloat light_diffuse[] = {1.0, 1.0, 1.0, 1.0};
  /* 鏡面光
  GLfloat light_specular[] = {1.0, 1.0, 1.0, 1.0}; */
  /* ライトモデルの環境光 */
  GLfloat lmodel_ambient[] = {0.4, 0.4, 0.4, 1.0};

  glEnable(GL_DEPTH_TEST);
  glDepthFunc(GL_LESS);

  /* 光源0の設定 */
  glLightfv(GL_LIGHT0, GL_POSITION, light_position0);
  glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
  glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
  /* 光源1の設定 */
  glLightfv(GL_LIGHT1, GL_POSITION, light_position1);
  glLightfv(GL_LIGHT1, GL_AMBIENT, light_ambient);
  glLightfv(GL_LIGHT1, GL_DIFFUSE, light_diffuse);
  /* 光源2の設定 */
  glLightfv(GL_LIGHT2, GL_POSITION, light_position2);
  glLightfv(GL_LIGHT2, GL_AMBIENT, light_ambient);
  glLightfv(GL_LIGHT2, GL_DIFFUSE, light_diffuse);
  /* 光源3の設定 */
  glLightfv(GL_LIGHT3, GL_POSITION, light_position3);
  glLightfv(GL_LIGHT3, GL_AMBIENT, light_ambient);
  glLightfv(GL_LIGHT3, GL_DIFFUSE, light_diffuse);
  /* 光源4の設定 */
  glLightfv(GL_LIGHT4, GL_POSITION, light_position4);
  glLightfv(GL_LIGHT4, GL_AMBIENT, light_ambient);
  glLightfv(GL_LIGHT4, GL_DIFFUSE, light_diffuse);
  /* 光源5の設定 */
  glLightfv(GL_LIGHT5, GL_POSITION, light_position5);
  glLightfv(GL_LIGHT5, GL_AMBIENT, light_ambient);
  glLightfv(GL_LIGHT5, GL_DIFFUSE, light_diffuse);
  /* 環境光の設定 */
  glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);

  glEnable(GL_LIGHTING);
  if(l0){
    glEnable(GL_LIGHT0);
  } else{
    glDisable(GL_LIGHT0);
  }
  if(l1){
    glEnable(GL_LIGHT1);
  } else{
    glDisable(GL_LIGHT1);
  }
  if(l2){
    glEnable(GL_LIGHT2);
  } else{
    glDisable(GL_LIGHT2);
  }
  if(l3){
    glEnable(GL_LIGHT3);
  } else{
    glDisable(GL_LIGHT3);
  }
  if(l4){
    glEnable(GL_LIGHT4);
  } else{
    glDisable(GL_LIGHT4);
  }
  if(l5){
    glEnable(GL_LIGHT5);
  } else{
    glDisable(GL_LIGHT5);
  }

}

/*
 * Togl widget display callback.  This is called by Tcl/Tk when the widget's
 * contents have to be redrawn.  Typically, we clear the color and depth
 * buffers, render our objects, then swap the front/back color buffers.
 */
void display_cb( struct Togl *togl )
{
  int width = Togl_Width(togl);
```

```
  int height = Togl_Height(togl);
  float backF;

  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  /* カメラの位置は原点，z軸の負の方向を向いている */
  glLoadIdentity();
  if(ofpsw == ORTHO){
    /*
     * ORTHOの場合，カメラの位置をz軸に沿って正（断じて負ではない）の方向に移動すると,
     * クリップされてしまうので移動しない
     */
    glTranslatef(0.0, 0.0, 0.0);
  } else if(ofpsw == FRUSTUM){
    /* カメラの移動量を計算 */
    if(width > height){
      backF = -width;
    } else{
      backF = -height;
    }
    /* カメラの位置をz軸に沿って正（断じて負ではない）の方向に移動，z軸の負の方向を向いている */
    glTranslatef(0.0, 0.0, backF);
  } else{
    /* テンポラリの移動量．あとでちゃんと考えて移動量を計算するようにする */
    glTranslatef(0.0, 0.0, -(viewNearP)-(width/2));
  }

  glRotatef(xAngle, 1.0, 0.0, 0.0);  /* Rotate by X, Y, and Z angles */
  glRotatef(yAngle, 0.0, 1.0, 0.0);
  glRotatef(zAngle, 0.0, 0.0, 1.0);

  glEnable(GL_DEPTH_TEST);

  light_init();

  draw_object();

  glDisable(GL_DEPTH_TEST);
  glLoadIdentity();
  Togl_SwapBuffers(togl);
}

int setXrot_cb(struct Togl *togl, int argc, char *argv[])
{
  Tcl_Interp *interp = Togl_Interp(togl);

  /* error checking */
  if(argc != 3){
    Tcl_SetResult(interp,
                  "wrong # args: should be \"pathName setXrot ?angle?\"",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  xAngle = atof(argv[2]);

  /* 視点座標の計算 */
  pointOfView();

  Togl_PostRedisplay(togl);

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

int setYrot_cb(struct Togl *togl, int argc, char *argv[])
{
  Tcl_Interp *interp = Togl_Interp(togl);

  /* error checking */
  if(argc != 3){
    Tcl_SetResult(interp,
                  "wrong # args: should be \"pathName setYrot ?angle?\"",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  yAngle = atof(argv[2]);

  /* 視点座標の計算 */
  pointOfView();

  Togl_PostRedisplay(togl);

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

int setZrot_cb(struct Togl *togl, int argc, char *argv[])
{
  Tcl_Interp *interp = Togl_Interp(togl);

  /* error checking */
  if(argc != 3){
    Tcl_SetResult(interp,
                  "wrong # args: should be \"pathName setZrot ?angle?\"",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  zAngle = atof(argv[2]);
```

```
  /* 視点座標の計算 */
  pointOfView();

  Togl_PostRedisplay(togl);

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

int setOmv_cb(struct Togl *togl, int argc, char *argv[])
{
  int width = Togl_Width(togl);
  int height = Togl_Height(togl);
  ndb *find();
  Tcl_Interp *interp = Togl_Interp(togl);

  /* error checking */
  if(argc != 4){
    Tcl_SetResult(interp,
                  "wrong # args: should be \"pathName setOmv ?coordinateX? ?coor
dinateY?\"",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  /*
   * マウスポインタの位置を計算
   * ピクセル単位の座標を当モデル空間の座標に変換
   * for ortho
   */
  xPoint = atof(argv[2]) * ((viewRight*2)/width);
  zPoint = atof(argv[3]) * ((viewTop*2)/height);

  /*
   * マウスポインタの位置に複数のノードが重なりあっている
   * 場合は一番上（y座標が一番大きいもの）を選択する
   */
  if(opObj){
    opObj->coordinate[0] = xPoint;
    opObj->coordinate[2] = zPoint;
    coordinate_calc(start);
    opObj->coordinate[0] = xPoint;
    opObj->coordinate[2] = zPoint;
    draw_object();
    Togl_PostRedisplay(togl);
  }else if((opObj = find())){
    opObj->coordinate[0] = xPoint;
    opObj->coordinate[2] = zPoint;
    coordinate_calc(start);
    opObj->coordinate[0] = xPoint;
    opObj->coordinate[2] = zPoint;
    draw_object();
    Togl_PostRedisplay(togl);
  }

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

/* tcl/tkから入力された座標，文字列などを，DBにセットする */
int setData_cb(struct Togl *togl, int argc, char *argv[])
{
  void dls_store();
  ndb *info, *tmp, *find_original();
  float viewHeight = viewTop - viewBottom;
  int sendnum = 0;
  Tcl_Interp *interp = Togl_Interp(togl);

  /* error checking */
  if(argc != 7){
    Tcl_SetResult(interp,
                  "wrong # args: should be \"pathName setData ?coordinateX? ?coo
rdinateZ? ?utterString? ?FollowNode? ?FollowKind?\"",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  if(!(info = (ndb *)malloc(sizeof(ndb)))){
    printf("Out of Memory!\n");
    return;
  }

  info->coordinate[0] = atof(argv[2]);
  info->coordinate[1] = (viewTop + sradius) - (viewHeight * objCtr / objAllNum);
  info->coordinate[2] = atof(argv[3]);
  strcpy(info->string, argv[4]);
  info->original = find_original(atoi(argv[5]));
  info->original_no = atoi(argv[5]);
  info->color = atoi(argv[6]);
  info->next = NULL;

  /* DBに登録 */
  dls_store(info);

  /* オブジェクト描画 */
  draw_object();
  Togl_PostRedisplay(togl);

  /* キーワードベクトル算出 */
  tmp = start;
  while(tmp){
    kw_extract(tmp->string, sendnum, 0);
    tmp = tmp->next;
    sendnum++;
```

```
  }

  /* 座標計算 */
  coordinate_calc(start);

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

/* DBに登録する．y座標で降順でソートされている */
void dls_store(ndb *i){

  objCtr++;

  if(objCtr == 1){
    i->next = NULL;
    start = i;
    last = i;
  } else{
    last->next = i;
    last = i;
  }
}

/* マウスポインタの位置から操作対象のノードを割り出す */
ndb *find(){
  ndb *info;

  info = start;

  while(info){
    if((xPoint - sradius < info->coordinate[0]) &&
       (xPoint + sradius > info->coordinate[0]) &&
       (zPoint - sradius < info->coordinate[2]) &&
       (zPoint + sradius > info->coordinate[2])){
      return info;
    }
    info = info->next;
  }
  return NULL;
}

/* フォローのオリジナルのノードのポインタを返す */
ndb *find_original(int num){
  ndb *info;
  int i = 1;

  info = start;

  while(info){
    if(num == i)
      return info;
    info = info->next;
    i++;
  }
  return NULL;
}

/* ButtonRelease-1時に，操作対象ノードをNULLにする */
int unsetOObj_cb(struct Togl *togl, int argc, char *argv[]){

  Tcl_Interp *interp = Togl_Interp(togl);

  /* error checking */
  if(argc != 2){
    Tcl_SetResult(interp,
                  "wrong # args: should be \"pathName unsetOObj\"",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  opObj = NULL;

  return TCL_OK;
}
/* オブジェクトの情報を save する */
void node_save(){
  ndb *info;
  FILE *fp;

  if(!(fp = fopen("nodeDatabase", "w"))){
    printf("cannnot open file\n");
    exit(0);
  }
  printf("\nsaving file\n");
  info = start;
  while(info){
    fprintf(fp,"%f %f %f %s %d %d\n",
            info->coordinate[0],info->coordinate[1],info->coordinate[2],
            info->string,info->color,info->original_no);
    info = info->next;
  }
  fclose(fp);
}

/* オブジェクトの情報を load する */
void node_load(char *openF){
  int size = 0;
  ndb *info = NULL, *tmp = NULL, *ftmp = NULL, *fbuf = NULL;
  FILE *fp;

  if(!(fp = fopen(openF, "r"))){
    printf("Cannnot Open File!\n");
    exit(0);
  }
```

```
  ftmp = start;
  while(ftmp){
    fbuf = ftmp->next;
    free(ftmp);
    ftmp = fbuf;
  }

  size = sizeof(ndb);
  if(!(start = (ndb *)malloc(size))){
    printf("Out of Memory!\n");
    return;
  }

  info = start;
  while(fscanf(fp,"%f %f %f %s %d %d\n",
               &info->coordinate[0], &info->coordinate[1], &info->coordinate[2],
               info->string, &info->color, &info->original_no) != EOF){
    info->original = NULL;
    info->next = NULL;
    info->original = find_original(info->original_no);
    if(!(info->next = (ndb *)malloc(size))){
      printf("Out of Memory!\n");
      return;
    }
    tmp = info;
    info = info->next;
    objCtr++;
  }
  free(info);
  tmp->next = NULL;
  last = tmp;
  fclose(fp);
}

/* オブジェクトの情報を save or load する */
int doSaveLoad_cb(struct Togl *togl, int argc, char *argv[])
{
  ndb *tmp, *test;
  int sendnum = 0, k = 0;

  Tcl_Interp *interp = Togl_Interp(togl);

  /* error checking */
  if(argc != 4){
    Tcl_SetResult(interp,
                  "wrong # args: should be \"pathName doSaveLoad ?save_or_load_s
w?\"",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  if(atoi(argv[2]) == 1){
    node_save();
  }else if(atoi(argv[2]) == 2){
    node_load(argv[3]);
  }

  /* キーワードベクトル算出 */
  tmp = start;
  while(tmp){
    kw_extract(tmp->string, sendnum, 1);
    tmp = tmp->next;
    sendnum++;
  }

  /* 座標計算 */
  coordinate_calc(start);

  /* オブジェクト描画 */
  draw_object();
  Togl_PostRedisplay(togl);

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

/* ライトの点灯／消灯をTkから行えるようにする */
int setLights_cb(struct Togl *togl, int argc, char *argv[])
{
  Tcl_Interp *interp = Togl_Interp(togl);

  /* error checking */
  if(argc != 8){
    Tcl_SetResult(interp,
                  "wrong # args for setLights",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  l0 = atoi(argv[2]);
  l1 = atoi(argv[3]);
  l2 = atoi(argv[4]);
  l3 = atoi(argv[5]);
  l4 = atoi(argv[6]);
  l5 = atoi(argv[7]);

  Togl_PostRedisplay(togl);

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

/* オブジェクト球を正方形で囲む */
```

```
void enclose_object()
{
  /* 内容表示 */
  if(encObjC){
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, CYANav);
    glMaterialfv(GL_FRONT, GL_EMISSION, CYANav);

    glBegin(GL_LINE_LOOP);
    glVertex3f(encObjC->coordinate[0] - sradius * 1.5,
               encObjC->coordinate[1],
               encObjC->coordinate[2] - sradius * 1.5);
    glVertex3f(encObjC->coordinate[0] + sradius * 1.5,
               encObjC->coordinate[1],
               encObjC->coordinate[2] - sradius * 1.5);
    glVertex3f(encObjC->coordinate[0] + sradius * 1.5,
               encObjC->coordinate[1],
               encObjC->coordinate[2] + sradius * 1.5);
    glVertex3f(encObjC->coordinate[0] - sradius * 1.5,
               encObjC->coordinate[1],
               encObjC->coordinate[2] + sradius * 1.5);
    glEnd();
  }

  /* 洗濯 */
  if(encObjR){
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, REDav);
    glMaterialfv(GL_FRONT, GL_EMISSION, REDav);

    glBegin(GL_LINE_LOOP);
    glVertex3f(encObjR->coordinate[0] - sradius * 1.5,
               encObjR->coordinate[1],
               encObjR->coordinate[2] - sradius * 1.5);
    glVertex3f(encObjR->coordinate[0] + sradius * 1.5,
               encObjR->coordinate[1],
               encObjR->coordinate[2] - sradius * 1.5);
    glVertex3f(encObjR->coordinate[0] + sradius * 1.5,
               encObjR->coordinate[1],
               encObjR->coordinate[2] + sradius * 1.5);
    glVertex3f(encObjR->coordinate[0] - sradius * 1.5,
               encObjR->coordinate[1],
               encObjR->coordinate[2] + sradius * 1.5);
    glEnd();
  }

  glMaterialfv(GL_FRONT, GL_EMISSION, BLACKav);
}

/* 内容表示のために選択されたオブジェクトを正方形で囲む */
int encloseObject_cb(struct Togl *togl, int argc, char *argv[])
{
  int width = Togl_Width(togl);
  int height = Togl_Height(togl);
  Tcl_Interp *interp = Togl_Interp(togl);
  ndb *indexedObj = NULL;

  /* error checking */
  if(argc != 4){
    Tcl_SetResult(interp,
                  "wrong # args for encloseObject",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  /*
   * マウスポインタの位置を計算
   * ピクセル単位の座標を当モデル空間の座標に変換
   * for ortho
   */
  xPoint = atof(argv[2]) * ((viewRight*2)/width);
  zPoint = atof(argv[3]) * ((viewTop*2)/height);

  if(indexedObj = find())
    encObjC = indexedObj;

  /* オブジェクト描画 */
  draw_object();
  Togl_PostRedisplay(togl);

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

/* Tcl/Tk のウィジェットを生成して，オブジェクトを生成する */
write_out(ndb *out)
{
  FILE *fp;

  /* 書出し処理 */
  if(!(fp = fopen("tmp.txt", "w"))){
    printf("cannnot open file\n");
    exit(0);
  }
  fprintf(fp, "%s", out->string);
  fflush(fp);
  system("text.tcl");
}

/* オブジェクトの含むテキストをみせる（ここではファイルに書き込む） */
int showText_cb(struct Togl *togl, int argc, char *argv[])
{
  int width = Togl_Width(togl);
  int height = Togl_Height(togl);
  Tcl_Interp *interp = Togl_Interp(togl);
  ndb *indexedObj = NULL;
  int pid;
```

```
  /* error checking */
  if(argc != 4){
    Tcl_SetResult(interp,
                  "wrong # args for showText",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  /*
   * マウスポインタの位置を計算
   * ピクセル単位の座標を当モデル空間の座標に変換
   * for ortho
   */
  xPoint = atof(argv[2]) * ((viewRight*2)/width);
  zPoint = atof(argv[3]) * ((viewTop*2)/height);

  /*
   * Get a child process.
   */
  if((pid = fork()) < 0){
    perror("fork");
    exit(1);
  }

  /*
   * The child process executes the code.
   */
  if(pid == 0){
    if(indexedObj = find()){
      write_out(indexedObj);
    }
    exit(1);
  }

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

/* 線を描くかどうかのスイッチをセットする */
int lineSwitch_cb(struct Togl *togl, int argc, char *argv[])
{
  Tcl_Interp *interp = Togl_Interp(togl);

  /* error checking */
  if(argc != 3){
    Tcl_SetResult(interp,
                  "wrong # args for lineSwitch",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  linesw = atoi(argv[2]);

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

/* 洗濯機能のオブジェクト選択 */
int rotateSet_cb(struct Togl *togl, int argc, char *argv[])
{
  Tcl_Interp *interp = Togl_Interp(togl);
  int width = Togl_Width(togl);
  int height = Togl_Height(togl);
  ndb *selectedObj = NULL;

  /* error checking */
  if(argc != 4){
    Tcl_SetResult(interp,
                  "wrong # args for rotateSet",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  /*
   * マウスポインタの位置を計算
   * ピクセル単位の座標を当モデル空間の座標に変換
   * for ortho
   */
  xPoint = atof(argv[2]) * ((viewRight*2)/width);
  zPoint = atof(argv[3]) * ((viewTop*2)/height);

  if(selectedObj = find())
    encObjR = selectedObj;

  /* オブジェクト描画 */
  draw_object();
  Togl_PostRedisplay(togl);

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

/* 角度を求める */
float calcRotate(float sinV)
{
  float sinX = 0.0;
  int i;

  if(sinV == 0.0){
    return(0.0);
  } else if(sinV == 1.0){
    return(PI / 2.0);
```

```
  } else{
    for(i = 0; i < 1000; i++){
      sinX = sin(2.0 * PI * i * 0.001);
      if(sinV < sinX)
        return(2.0 * PI * i * 0.001);
    }
  }
}

/* 洗濯する */
int rotateObj_cb(struct Togl *togl, int argc, char *argv[])
{
  Tcl_Interp *interp = Togl_Interp(togl);

  /* error checking */
  if(argc != 3){
    Tcl_SetResult(interp,
                  "wrong # args for rotateObj",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  /* オブジェクトが選択されていたら，洗濯を実行 */
  if(encObjR)
    timersw = atoi(argv[2]);

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}


/* 象限別にthetaの値を設定する */
float currentTheta(float x, float y, float theta)
{
  if(x >= 0.0){
    if(y >= 0.0){
      /* x = plus, y = plus */
      theta = theta;
    } else{
      /* x = plus, y = minus */
      theta = (2.0 * PI) - theta;
    }
  } else{
    if(y >= 0.0){
      /* x = minus, y = plus */
      theta = PI - theta;
    } else{
      /* x = minus, y = minus */
      theta = PI + theta;
    }
  }
  return theta;
}

/* タイマーを発生させる．ここでは洗濯に使用 */
void timer_cb(struct Togl *togl)
{
  float length = 0.0, theta = 0.0, metatheta = 0.0;
  float pitch = 100.0;
  float tmpX = 0.0, tmpZ = 0.0;

  if(timersw){
    length = sqrt(MULT2(encObjR->coordinate[0]) + MULT2(encObjR->coordinate[2]))
;
    /* sin(theta) の値を入力して，theta の値を得る（単位ラジアン） */
    theta = calcRotate(fabs(encObjR->coordinate[2]) / length);

    metatheta = currentTheta(encObjR->coordinate[0], encObjR->coordinate[2], the
ta);

    /* 時計回りのときの回転移動 */
    if(timersw == 1){
      if((metatheta + (2.0 * PI / pitch)) >= 2.0 * PI){
        metatheta = 2.0 * PI / pitch;
      } else{
        metatheta = metatheta + (2.0 * PI / pitch);
      }
    /* 反時計回りのときの回転移動 */
    } else if(timersw == 2){
      if((metatheta - (2.0 * PI / pitch)) <= 0.0){
        metatheta = 2.0 * PI - (2.0 * PI / pitch);
      } else{
        metatheta = metatheta - (2.0 * PI / pitch);
      }
    }

    tmpX = length * cos(metatheta);
    tmpZ = length * sin(metatheta);
    if(encObjR){
      encObjR->coordinate[0] = tmpX;
      encObjR->coordinate[2] = tmpZ;
      coordinate_calc(start);
      encObjR->coordinate[0] = tmpX;
      encObjR->coordinate[2] = tmpZ;
    }

    draw_object();
    Togl_PostRedisplay(togl);
  }
}

/* 正方形を消す */
int clearSquare_cb(struct Togl *togl, int argc, char *argv[])
{
```

```
  Tcl_Interp *interp = Togl_Interp(togl);

  /* error checking */
  if(argc != 3){
    Tcl_SetResult(interp,
                  "wrong # args for clearSquare",
                  TCL_STATIC);
    return TCL_ERROR;
  }

  if(atoi(argv[2]) == 1){
    encObjC = NULL;
  } else if(atoi(argv[2]) == 2){
    encObjR = NULL;
  }

  /* オブジェクト描画 */
  draw_object();
  Togl_PostRedisplay(togl);

  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

/* パラメータをシミュレーションして自動でセットする */
simParam()
{
  ndb *info;
  float savec[50][3];
  int i = 0, sw = 1, sww = 0, limit = 0, j = 0, mini = 0;

  /* 現時点での座標をセーブする */
  info = start;
  while(info){
    orderSetF(savec[i], info->coordinate, 3);
    info = info->next;
    i++;
  }

  /*
   * パラメータを初期値にセット.
   * 一番上のオブジェクトを動かす.
   * X,Zともに50ピクセル動かした時点で，全オブジェクトの座標をチェック.
   * 端に寄り過ぎてたら，パラメータを1/10にして，再度チャレンジ.
   * 中央に集まり過ぎてたら，パラメータを10倍にして，再度チャレンジ.
   */
  while(sw){
    sww = 0;
    info = start;
    info->coordinate[0] = info->coordinate[0] + 100.0;
    info->coordinate[2] = info->coordinate[0] + 100.0;
    coordinate_calc(start);
    while(info){
      if(info->coordinate[0] >  430.0 || info->coordinate[2] >  430.0 ||
         info->coordinate[0] < -430.0 || info->coordinate[2] < -430.0){
        sww = 1;
      }
      /* 小さくまとまってしまう病は，とりあえず対処しない
         else if(info->coordinate[0] <  50.0 || info->coordinate[2] <  50.0 ||
         info->coordinate[0] > -50.0 || info->coordinate[2] > -50.0){
         sww = 2;
         } */
      info = info->next;
    }
    if(sww == 1){
      /* 理想距離調整数もとりあえずいじらない
      idealAdjust -= 0.1;*/
      moveAdjust = moveAdjust / 1.1;
      /* 小さくまとまってしまう病は，とりあえず対処しない
         } else if(sww == 2){
         idealAdjust += 0.1;
         moveAdjust = moveAdjust * 1.1; */
    } else{
      sw = 0;
    }
    if(limit > 20){
      break;
    }
    limit++;
  }

  /* セーブした座標をリロードする */
  i = 0;
  info = start;
  while(info){
    orderSetF(info->coordinate, savec[i], 3);
    info = info->next;
    i++;
  }
printf("idealAdjust = %f\n", idealAdjust);
printf("moveAdjust = %f\n", moveAdjust);
printf("springFixNum = %f\n", springFixNum);

}

/* パラメータをセットする */
int setParameter_cb(struct Togl *togl, int argc, char *argv[])
{
  Tcl_Interp *interp = Togl_Interp(togl);

  /* error checking */
  if(argc != 6){
    Tcl_SetResult(interp,
                  "wrong # args for setParameter",
                  TCL_STATIC);
```

```
    return TCL_ERROR;
  }

  if(atoi(argv[2]) == ON){
    simParam();
  } else{
    if(atof(argv[3])){
      idealAdjust = atof(argv[3]);
    }
    if(atof(argv[4])){
      moveAdjust = atof(argv[4]);
    }
    if(atof(argv[5])){
      springFixNum = atof(argv[5]);
    }
  }


  /* Let result string equal value */
  strcpy(interp->result, argv[2]);
  return TCL_OK;
}

/*
 * Called by Tk_Main() to let me initialize the modules (Togl) I will need.
 */
int my_init(Tcl_Interp *interp)
{
  /*
   * Initialize Tcl, Tk, and the Togl widget module.
   */
  if(Tcl_Init(interp)  == TCL_ERROR) return TCL_ERROR;
  if(Tk_Init(interp)   == TCL_ERROR) return TCL_ERROR;
  if(Togl_Init(interp) == TCL_ERROR) return TCL_ERROR;

  /*
   * Specify the C callback functions for widget creation, display,
   * and reshape.
   */
  Togl_CreateFunc(create_cb);
  Togl_DisplayFunc(display_cb);
  Togl_ReshapeFunc(reshape_cb);
  Togl_TimerFunc( timer_cb );

  /*
   * Make a new Togl widget command so the Tcl code can set a C variable.
   */
  Togl_CreateCommand("setXrot", setXrot_cb);
  Togl_CreateCommand("setYrot", setYrot_cb);
  Togl_CreateCommand("setZrot", setZrot_cb);
  Togl_CreateCommand("setOmv",  setOmv_cb);
  Togl_CreateCommand("setData", setData_cb);
  Togl_CreateCommand("unsetOObj", unsetOObj_cb);
  Togl_CreateCommand("doSaveLoad", doSaveLoad_cb);
  Togl_CreateCommand("setLights", setLights_cb);
  Togl_CreateCommand("encloseObject", encloseObject_cb);
  Togl_CreateCommand("showText", showText_cb);
  Togl_CreateCommand("lineSwitch", lineSwitch_cb);
  Togl_CreateCommand("rotateSet", rotateSet_cb);
  Togl_CreateCommand("rotateObj", rotateObj_cb);
  Togl_CreateCommand("clearSquare", clearSquare_cb);
  Togl_CreateCommand("setParameter", setParameter_cb);

  /*
   * Specify a user-specific startup file to invoke if the application
   * is run interactively.  Typically the startup file is "~/.apprc"
   * where "app" is the name of the application.  If this line is deleted
   * then no user-specific startup file will be run under any conditions.
   */
#if (TCL_MAJOR_VERSION * 100 + TCL_MINOR_VERSION) >= 705
  Tcl_SetVar(interp, "tcl_rcFileName", "./monkeyViewer.tcl", TCL_GLOBAL_ONLY);
#else
  tcl_RcFileName = "./monkeyViewer.tcl";
#endif

  return TCL_OK;
}

int main(int argc, char *argv[])
{
  Tk_Main(argc, argv, my_init);
  return 0;
}
```

```
# monkey.tcl   by FUJITA, kunihiko  Oct/20/97

# Togl - a Tk OpenGL widget
# Copyright (C) 1996  Brian Paul and Ben Bederson

proc setup {} {
    wm title . "monkey"

#### kanji internalCode EUC

#### exec kinput2 &

####
#### font の定義
####
    set defaultfont -*-courier-bold-r-normal-*-16-*-*-*-*-*-*-*
    set menufont -*-screen-bold-r-normal-*-16-*-*-*-*-*-*-*
    set buttonfont -*-helvetica-bold-r-normal-*-16-*-*-*-*-*-*-*

####
#### widget の定義
####

#### フレームの定義
    frame .menu -relief raised -bd 2
    frame .f1
    frame .f2

#### メニュー部分の設定
    menubutton .menu.file -text "File" -menu .menu.file.m -font $defaultfont
    menubutton .menu.light -text "Light" -menu .menu.light.m -font $defaultfont
    menubutton .menu.line -text "Line" -menu .menu.line.m -font $defaultfont
    menubutton .menu.rotate -text "Rotate" -menu .menu.rotate.m -font $defaultfo
nt
    menubutton .menu.parameter -text "Parameter" -menu .menu.parameter.m -font $
defaultfont

    menu .menu.file.m -tearoff 0 -font $defaultfont
    .menu.file.m add command -label "Load" -command {fileSelection 2}
    .menu.file.m add command -label "Save" -command {save_load 1}
    .menu.file.m add separator
    .menu.file.m add command -label "Clear CYAN Square" -command {clearsq 1}
    .menu.file.m add command -label "Clear RED Square" -command {clearsq 2}
    .menu.file.m add separator
    .menu.file.m add command -label "Quit" -command exit

    menu .menu.light.m -font $defaultfont
    .menu.light.m add checkbutton -label "X Plus" -variable vrb0 \
            -command {setLight $vrb0 $vrb1 $vrb2 $vrb3 $vrb4 $vrb5}
    .menu.light.m add checkbutton -label "Y Plus" -variable vrb1 \
            -command {setLight $vrb0 $vrb1 $vrb2 $vrb3 $vrb4 $vrb5}
    .menu.light.m add checkbutton -label "Z Plus" -variable vrb2 \
            -command {setLight $vrb0 $vrb1 $vrb2 $vrb3 $vrb4 $vrb5}
    .menu.light.m add checkbutton -label "X Minus" -variable vrb3 \
            -command {setLight $vrb0 $vrb1 $vrb2 $vrb3 $vrb4 $vrb5}
    .menu.light.m add checkbutton -label "Y Minus" -variable vrb4 \
            -command {setLight $vrb0 $vrb1 $vrb2 $vrb3 $vrb4 $vrb5}
    .menu.light.m add checkbutton -label "Z Minus" -variable vrb5 \
            -command {setLight $vrb0 $vrb1 $vrb2 $vrb3 $vrb4 $vrb5}

    menu .menu.line.m -font $defaultfont
    .menu.line.m add radiobutton -label "Line ON" -variable lineline \
            -command {setLine 1}
    .menu.line.m add radiobutton -label "Line OFF" -variable lineline \
            -command {setLine 0}
    .menu.line.m add separator

    menu .menu.rotate.m -font $defaultfont
    .menu.rotate.m add command -label "Clockwise" -command {rotating 1}
    .menu.rotate.m add command -label "Anti-Clockwise" -command {rotating 2}
    .menu.rotate.m add separator
    .menu.rotate.m add command -label "Stop" -command {rotating 0}

    menu .menu.parameter.m -font $defaultfont
    .menu.parameter.m add command -label "Manual Setting" -command {paramset .di
alog 0}
    .menu.parameter.m add command -label "Auto Setting" -command {paramset .dial
og 1}

#### OpenGL画面の設定
    togl .f1.ogl -width 900 -height 900 -rgba true -double true -depth true

#### 操作スケール・ボタンなどの設定
    label .f2.l0 -font $defaultfont
    scale .f2.sx -label {X Axis} -font $defaultfont -from 0 -to 90 \
            -command {setAngle x} -orient horizontal -length 270
    scale .f2.sy -label {Y Axis} -font $defaultfont -from 0 -to 90 \
            -command {setAngle y} -orient horizontal -length 270
    scale .f2.sz -label {Z Axis} -font $defaultfont -from 0 -to 90 \
            -command {setAngle z} -orient horizontal -length 270
    label .f2.l1 -font $defaultfont
    label .f2.lnode -text "Enter Object No." -font $defaultfont
    entry .f2.enode
    radiobutton .f2.rbqu -text "Question" -font $defaultfont -fg white -bg magen
ta -width 15 \
            -relief raised -variable follow -value 1
    radiobutton .f2.rban -text "Answer" -font $defaultfont -fg white -bg blue -w
idth 15 \
            -relief raised -variable follow -value 2
    radiobutton .f2.rbag -text "Agreement" -font $defaultfont -bg green -width 1
5 \
            -relief raised -variable follow -value 3
    radiobutton .f2.rbob -text "Objection" -font $defaultfont -fg white -bg red
-width 15 \
            -relief raised -variable follow -value 4
    radiobutton .f2.rbex -text "Explanation" -font $defaultfont -bg cyan -width
```

```
15 \
            -relief raised -variable follow -value 5
    radiobutton .f2.rbne -text "New" -font $defaultfont -bg yellow -width 15 \
            -relief raised -variable follow -value 6
########## this is temporary setting start
    label .f2.ls4 -text "Enter String" -font $defaultfont
    entry .f2.es4
    button .f2.bsub -text "Submit" -font $defaultfont -command {submit_data $fol
low} -width 15

####
#### pack
####

#### フレーム .menu の pack
    pack .menu.file .menu.light .menu.line .menu.rotate .menu.parameter -side le
ft
    pack .menu -side top -fill x

#### フレーム .f1 (OpenGL画面のある方) の pack
    pack .f1.ogl -side left -padx 3 -pady 3 -fill both -expand t
    pack .f1 -side left -fill both -expand t

#### フレーム .f2 (スケール・ボタンなどのある方) の pack
    pack .f2.l0 -side top -padx 3 -pady 10 -fill x
    pack .f2.sx -side top -padx 3 -pady 3
    pack .f2.sy -side top -padx 3 -pady 3
    pack .f2.sz -side top -padx 3 -pady 3
    pack .f2.l1 -side top -padx 3 -pady 10 -fill x
    pack .f2.lnode -side top -padx 3 -pady 3 -fill x
    pack .f2.enode -side top -padx 3 -pady 3 -fill x
    pack .f2.rbqu -side top -padx 3 -pady 3
    pack .f2.rban -side top -padx 3 -pady 3
    pack .f2.rbag -side top -padx 3 -pady 3
    pack .f2.rbob -side top -padx 3 -pady 3
    pack .f2.rbex -side top -padx 3 -pady 3
    pack .f2.rbne -side top -padx 3 -pady 3
########## this is temporary packing start
    pack .f2.ls4 -side top -padx 3 -pady 3 -fill x
    pack .f2.es4 -side top -padx 3 -pady 3 -fill x
    pack .f2.bsub -side top -padx 3 -pady 3
########## this is temporary packing end

    pack .f2 -side right -fill both -expand t

#### チェックボタン"X Plus","Y Plus"を選択状態にする (X,Y軸正方向からの照明は最
初からつけておく)
    .menu.light.m invoke 1
    .menu.light.m invoke 2

#### ラジオボタン"Line ON"を選択状態にする
    .menu.line.m invoke 1
```

```
#### マウスボタン2が押された時のイベントのバインド
    bind .f1.ogl <ButtonPress-2> {
        press_event_B2 [lindex [%W config -width] 4] [lindex [%W config -height]
 4] %x %y
    }

#### マウスボタン3が押された時のイベントのバインド
    bind .f1.ogl <ButtonPress-3> {
        press_event_B3 [lindex [%W config -width] 4] [lindex [%W config -height]
 4] %x %y
    }

}

#### This is called when mouse button 1 is pressed and moved in
#### the OpenGL windows.
proc motion_event_B1 {width height x y} {

#### 画面の中心が(0,0)になるように変換
    .f1.ogl setOmv [expr $x - ($width / 2)] [expr $y - ($height / 2)]
}

#### This is called when mouse button 2 is pressed and moved in
#### the OpenGL windows.
proc press_event_B2 {width height x y} {

#### 画面の中心が(0,0)になるように変換
    .f1.ogl encloseObject [expr $x - ($width / 2)] [expr $y - ($height / 2)]
    .f1.ogl showText [expr $x - ($width / 2)] [expr $y - ($height / 2)]
}

#### This is called when mouse button 3 is pressed and moved in
#### the OpenGL windows.
proc press_event_B3 {width height x y} {

#### 画面の中心が(0,0)になるように変換
    .f1.ogl rotateSet [expr $x - ($width / 2)] [expr $y - ($height / 2)]
}

proc save_load {sw_sl} {

    .f1.ogl doSaveLoad $sw_sl  $sw_sl
}

#### This is called when SUBMIT button is pressed and data are
#### carried to C program.
proc submit_data {rbfollow} {
    .f1.ogl setData [.f2.ec1 get] [.f2.ec3 get] [.f2.es4 get] [.f2.enode get] $r
bfollow
    .f2.ec1 delete 0 end
    .f2.ec3 delete 0 end
    .f2.es4 delete 0 end
    .f2.enode delete 0 end
```

```
}

# This is called when a slider is changed.
proc setAngle {axis value} {
    global xAngle yAngle zAngle

    switch -exact $axis {
        x {.f1.ogl setXrot $value
            if { $value == "90" } {
                .f2.l0 configure -text "Semantic View"
                .f2.l1 configure -text "You can move objects!"
                bind .f1.ogl <B1-Motion> {
                    motion_event_B1 [lindex [%W config -width] 4] [lindex [%W co
nfig -height] 4] %x %y
                }
                bind .f1.ogl <ButtonRelease-1> {
                    .f1.ogl unsetOObj
                }
            } elseif { $value == "0" } {
                .f2.l0 configure -text "Cause-and-Effect View"
            } else {
                bind .f1.ogl <B1-Motion> {}
                .f2.l0 configure -text ""
                .f2.l1 configure -text ""
            }
        }
        y {.f1.ogl setYrot $value}
        z {.f1.ogl setZrot $value}
    }
}

# This is called when a check button is changed.
proc setLight {vrb0 vrb1 vrb2 vrb3 vrb4 vrb5} {

    .f1.ogl setLights $vrb0 $vrb1 $vrb2 $vrb3 $vrb4 $vrb5

}

# This is called when a line radiobutton is changed.
proc setLine {onoff} {

    .f1.ogl lineSwitch $onoff
}

# This is called when a rotate button is pressed.
proc rotating {num} {

    .f1.ogl rotateObj $num
}

# This is called when a rotate button is pressed.
proc clearsq {color} {
    .f1.ogl clearSquare $color
}

# This is called to set parameters.
proc paramsets {w ma} {

    if {$ma == "0"} {
        toplevel $w
        wm title $w "Dialog Message"
        frame $w.f1
        frame $w.f2
        frame $w.f3
        label $w.f1.la -text "Set Ideal Length Adjustment"
        entry $w.f1.en -relief sunken -width 40
        label $w.f2.la -text "Set Movement Adjustment"
        entry $w.f2.en -relief sunken -width 40
        label $w.f3.la -text "Set Spring Constant"
        entry $w.f3.en -relief sunken -width 40
        button $w.bu  -text "OK" -relief raised -width 40 \
                -command "mkWindow $w $ma"
        pack $w.f1.la -side left -padx 3 -pady 3
        pack $w.f1.en -side right -padx 3 -pady 3
        pack $w.f2.la -side left -padx 3 -pady 3
        pack $w.f2.en -side right -padx 3 -pady 3
        pack $w.f3.la -side left -padx 3 -pady 3
        pack $w.f3.en -side right -padx 3 -pady 3
        pack $w.f1 -side top -fill x
        pack $w.f2 -side top -fill x
        pack $w.f3 -side top -fill x
        pack $w.bu -side bottom -padx 3 -pady 3 -fill x
    } elseif {$ma == "1"} {
        .f1.ogl setParameter $ma 1 2 3
    }
}

proc mkWindow {ww maw} {

    .f1.ogl setParameter $maw [$ww.f1.en get] [$ww.f2.en get] [$ww.f3.en get]

    destroy $ww
}

#--------------------------------------
# filebox.tcl --
#
# This demonstration script prompts the user to select a file.
#
# SCCS: @(#) filebox.tcl 1.2 96/08/27 15:03:26

proc fileSelection {sw_sl} {

    set defaultfont -*-courier-bold-r-normal-*-16-*-*-*-*-*-*-*
    set w .filebox
```

```
    catch {destroy $w}
    toplevel $w
    wm title $w "File Selection Dialogs"
    wm iconname $w "filebox"
    positionWindow $w

    label $w.msg -font $defaultfont -wraplength 4i -justify left \
            -text "Enter a file name in the entry box or click on the \"Browse\"
 buttons to select a file name using the file selection dialog."
    pack $w.msg -side top
    frame $w.buttons
    pack $w.buttons -side bottom -fill x -pady 2m
    button $w.buttons.dismiss -text Dismiss -command "returnFile $w $sw_sl"
    pack $w.buttons.dismiss -side left -expand 1

    foreach i {open save} {
        set f [frame $w.$i]
        label $f.lab -text "Select a file to $i: " -anchor e
        entry $f.ent -width 20
        button $f.but -text "Browse ..." -command "fileDialog $w $f.ent $i"
        pack $f.lab -side left
        pack $f.ent -side left -expand yes -fill x
        pack $f.but -side left
        pack $f -fill x -padx 1c -pady 3
    }

#    if ![string compare $tcl_platform(platform) unix] {
#        checkbutton $w.strict -text "Use Motif Style Dialog" \
#                -variable tk_strictMotif -onvalue 1 -offvalue 0
#        pack $w.strict -anchor c
#    }

}

proc fileDialog {w ent operation} {
    #   Type names              Extension(s)    Mac File Type(s)
    #
    #---------------------------------------------------------
    set types {
        {"All files"            *}
        {"Text files"           {.txt .doc}     }
        {"Text files"           {}              TEXT}
        {"Tcl Scripts"          {.tcl}          TEXT}
        {"C Source Files"       {.c .h}         }
        {"All Source Files"     {.tcl .c .h}    }
        {"Image Files"          {.gif}          }
        {"Image Files"          {.jpeg .jpg}    }
        {"Image Files"          ""              {GIFF JPEG}}

    }
    if {$operation == "open"} {
        set file [tk_getOpenFile -filetypes $types -parent $w]
    } else {
        set file [tk_getSaveFile -filetypes $types -parent $w \
            -initialfile Untitled -defaultextension .txt]
    }
    if [string compare $file ""] {
        $ent delete 0 end
        $ent insert 0 $file
        $ent xview end
    }
}

# positionWindow --
# This procedure is invoked by most of the demos to position a
# new demo window.
#
# Arguments:
# w -           The name of the window to position.

proc positionWindow w {
    wm geometry $w +300+300
}

proc returnFile {w sw_sl} {

    .f1.ogl doSaveLoad $sw_sl [$w.open.ent get]

    destroy $w
}
#-------------------------------------

# Execution starts here!
setup
```

```
# filebox.tcl --
#
# This demonstration script prompts the user to select a file.
#
# SCCS: @(#) filebox.tcl 1.2 96/08/27 15:03:26

set w .filebox
catch {destroy $w}
toplevel $w
wm title $w "File Selection Dialogs"
wm iconname $w "filebox"
positionWindow $w

label $w.msg -font $font -wraplength 4i -justify left -text "Enter a file name i
n the entry box or click on the \"Browse\" buttons to select a file name using t
he file selection dialog."
pack $w.msg -side top

frame $w.buttons
pack $w.buttons -side bottom -fill x -pady 2m
button $w.buttons.dismiss -text Dismiss -command "destroy $w"
button $w.buttons.code -text "See Code" -command "showCode $w"
pack $w.buttons.dismiss $w.buttons.code -side left -expand 1

foreach i {open save} {
    set f [frame $w.$i]
    label $f.lab -text "Select a file to $i: " -anchor e
    entry $f.ent -width 20
    button $f.but -text "Browse ..." -command "fileDialog $w $f.ent $i"
    pack $f.lab -side left
    pack $f.ent -side left -expand yes -fill x
    pack $f.but -side left
    pack $f -fill x -padx 1c -pady 3
}

if ![string compare $tcl_platform(platform) unix] {
    checkbutton $w.strict -text "Use Motif Style Dialog" \
        -variable tk_strictMotif -onvalue 1 -offvalue 0
    pack $w.strict -anchor c
}

proc fileDialog {w ent operation} {
    #   Type names              Extension(s)    Mac File Type(s)
    #
    #---------------------------------------------------------
    set types {
        {"Text files"           {.txt .doc}     }
        {"Text files"           {}              TEXT}
        {"Tcl Scripts"          {.tcl}          TEXT}
        {"C Source Files"       {.c .h}         }
        {"All Source Files"     {.tcl .c .h}    }
        {"Image Files"          {.gif}          }
        {"Image Files"          {.jpeg .jpg}    }
        {"Image Files"          ""              {GIFF JPEG}}
        {"All files"            *}
    }
    if {$operation == "open"} {
        set file [tk_getOpenFile -filetypes $types -parent $w]
    } else {
        set file [tk_getSaveFile -filetypes $types -parent $w \
            -initialfile Untitled -defaultextension .txt]
    }
    if [string compare $file ""] {
        $ent delete 0 end
        $ent insert 0 $file
        $ent xview end
    }
}
```

```
#!/home/kunihiko/bin/wish4.2jp -f

text .text -width 40 -height 10 -relief sunken -bd 2 -yscrollcommand ".scroll se
t"
scrollbar .scroll -command ".text yview"
pack .scroll -side right -fill y
pack .text -side left

proc loadFile file {
    .text delete 1.0 end
    set f [open $file r]
    while {![eof $f]} {
        .text insert end [read $f 1000]
    }
    close $f
}

loadFile tmp.txt
```