2004－CG－117　(22)
2004／11／27

3 次元事象を可視化するための作図ツールの開発

山田　真也[†]　　白井　靖人[‡]

概要

3 次元空間中で発生する事象の説明において、図を用いることがある。これらの図の多くは幾何学的な図形によって構成され、事象をある方向から捉えた様子を表現したものとなっている。本来 3 次元の事象をこのように一面的な捉え方だけで提示しても、その事象を理解することは困難である。空間的に事象を表現し、多角的に見せることができれば理解が容易になると考えられる。このような場合に 3 次元の CG ソフトウェアの使用が有効であるが、現在の CG ソフトウェアは機能の高速化、高精細化の反面、3 次元事象を表現するためには必要のない機能が多く存在する。また、一般の CG ソフトウェアにはない特別な機能も必要となる。本研究では一般の人でも扱える作図ツールを開発し、作図と理解の両面から支援する。

Development of the drawing tool for visualizing 3-dimensional phenomenon

Shinya Yamada[†]　　Yasuto Shirai[‡]

ABSTRACT

A figure may be used for explanation of the phenomenon generated in 3-dimensional space. Many of these figures are constituted by the geometric figure, and it expresses a phenomenon from a certain direction. However, it is difficult to understand the phenomenon only by partial expression. If we can express and show a phenomenon spatially, we can understand easily. In such a case, 3-dimensional CG software is effective. The present CG software is fine and fast. However, Many functions are unnecessary in order to draw a 3-dimensional phenomenon. Moreover, A special function is also required. We will develop a drawing tool and it supports from both sides of drawing and understanding.

１．はじめに

数学や物理などで用いられる 3 次元の事象を 1 面的な図から空間的に把握するのは、見る側の空間把握能力を必要とし、一目見ただけでその事象を理解することは困難である。空間的に事象を表現し、多面的な角度から見せることができれば、理解が容易になると考えられる。

図 1 は透視投影と呼ばれる投影法を表した図である。このような図のほとんどは、幾何学図形で構成され、物体の名前などの情報も含んでいる。その構造を理解することは事象を把握する上で重要な要素となる。

[†]静岡大学院情報学研究科
Graduate School of Information, Shizuoka University
[‡]静岡大学情報学部
Faculty of Information, Shizuoka University

このような事象を図示する場合に図示する側としては、紙や黒板に図を描いたり、簡単なドローイングソフト等を用いて作図をする場合が多い。

しかし、事象をある方向からのみ捉えた図では必ずしも見る側にとって理解しやすい図であるとは限らない。また、作図する側にとっても空間的な図を 2 次元の平面に描くには熟練度を要する。このような場合に仮想的な 3 次元空間内に図を表現することができれば様々な角度から図を捉えることができ、理解も容易となる。そのような 3 次元空間内に図を描く方法として、3 次元 CG ソフトウェアの使用が有効な手段となる。

現在 CG ソフトは高精細化、高機能化が進んでいるが、その反面 3 次元事象を表現するためには必要のない余分な機能が多く存在する。これらの操作を習得するためには十分な時間をかけなければならない。また、本研究で開発

する作図ツールは一般の CG ソフトにはないような特別な機能も必要となる。

本研究では一般の人でも扱えるような作図ツールを開発し、作図する側にとっては 3 次元の CG ソフトを扱えるようになるために必要な時間や労力をかけることなく 3 次元事象の描写を可能にし、また図を見る側にとっては事象を多角的に捉えることによって理解しやすくする。

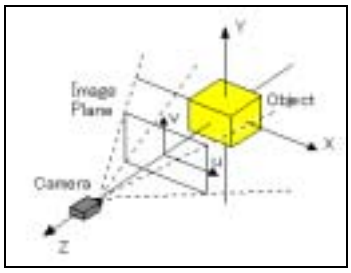


図 1：透視投影の図

２．機能要件

本研究で開発する作図ツールには一般の CG ソフトのような高精細なレンダリング機能やモデリングの際の多くの設定項目などは必要ではなく、最低限の作図機能があれば十分である。逆に 3 次元の事象を描くためには従来の 3 次元 CG ソフトウェアには備わっていない機能が必要となる。

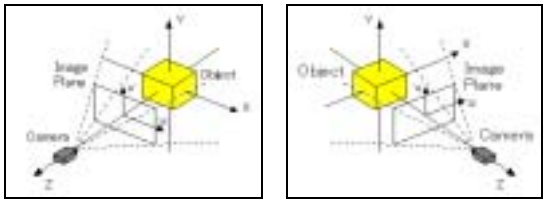
本研究で着目した機能要件について以下に紹介する。

2.1　ラベルの配置

モデリング操作によって作図された図中にある特定のオブジェクトに対してラベル付けを行うこと。図中に文字を加えることで、より分かりやすく図を表現することができる。

事象を表す場合に文字情報は重要な意味を持つと考えられる。通常の CG ソフトウェアでは文字や記号も一つのオブジェクトとして認識されているため、角度を変えて図形を表示した場合に文字オブジェクトも一緒に回転してしまい、角度によっては文字が読めなくなってしまう。

図 2 にはラベルを図中のオブジェクトに付加した場合の例を示している。例えば(a)から(b)のように図形を回転させたときも付加されたオブジェクトの付近に配置され、必ず投影面に向かって文字が前面を向いている必要がある。



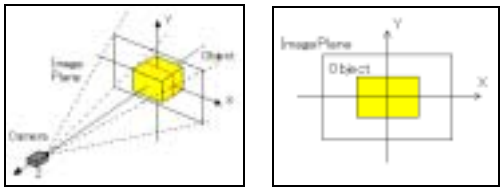
(a)**回転前**　　(b)**回転後**

図 2：ラベル配置の例

2.2　断面表示

断面として定義された平面図形とその他の平面及び立体図形との交差部分を表示することで、図形を切断したときの断面を見ることができるようにする。図 3 のように ImagePlane という面と、Object という立体が重なった場合に、面と立体との重なりあった部分を表示する。

この切断面を表示することで、Camera から見た ImagePlane に映る Object を図 b のように示すことができる。

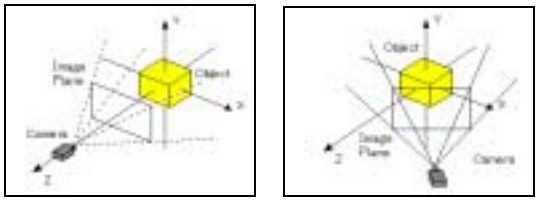


(a)全体図　　(b)投影面の様子

図 3：断面表示の例

2.3　図形同士の制約

平行・垂直などの制約を図形間に与え、制約を保つように図形同士の関係を維持する。制約を維持することで、ユーザがグラフィカルレイアウトの維持を気にする必要がなくなるため、多数の制御点の位置を一つずつ調整する必要もなくなり、事象の構造に対する理解も深まると考えられる。



(a)**移動前**　　(b)**移動後**

図 4：制約の維持

図 4 は、制約が与えられた場合に維持されるべき制約の図の一例である。例えば、図の Camera というオ

ブジェクトと ImagePlane というオブジェクトの間に Camera から伸びる放射状の破線が ImagePlane の四隅を通るという制約が与えられた時、(a)から(b)のように Camera オブジェクトを移動させると、両者間の制約が働き、Camera から ImagePlane の四隅に向かって伸びている放射線も Camera の変移に合わせて動き、またそれに伴い ImagePlane も動く。

この他にも、制約を与えることで事象の構造を変えることなくオブジェクトの変移が可能になり、事象に対する理解がより深まる。

３．研究方法

本研究で開発する作図ツールでは、作図方法に従来の 3 次元 CG ソフトウェアと同様に 3 面図を用いる。また、物体の形状表現にはワイヤフレームモデルを用いる。以下に物体を定義する方法について述べる。

3.1 定義方法

描画する図形はシーングラフによって定義される。シーングラフとは、3 次元仮想空間を概念的に表現するための構造のことで、DAG(directed-acyclic graph)を用いて表現される。DAG とはサイクル(繰返し)を持たない有向図(非周期有向図)である。

図 5 にシーングラフが定義された場合の構造を示す。このシーングラフの根ノードは実体を持たないが、定義された図形を子ノードとしてまとめた世界ノードを表す。仮想的な 3 次元空間内に図形を作図すると、その図形はシーングラフ中では世界ノードの子として定義される。図形を画面内に作図するたびにシーングラフ中の世界ノードにはオブジェクトのノードが子ノードとして増える。

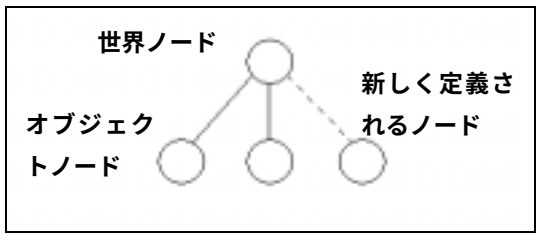


図 5：シーングラフ

オブジェクトノードはその図形の形状の情報、頂点の 3 次元座標データを保持している。それらの頂点を線分で結ぶことでオブジェクトを表現しているので、現在はワイヤフレームモデルでの表示となっている。

3.2 ラベル配置

付加されるラベルは設定される時点で付加するオブジェクトから一定距離離れた 3 次元配置座標を取得する。ラベルが付加されると図 6 のようにシーングラフ中に定義されたオブジェクトの子ノードとして定義される。このノードは配置された 3 次元座標と定義されたラベル名を保持する。このラベルノードはオブジェクトに対して付加されたラベルの数だけオブジェクトノードの子として定義される。

付加されるラベルはオブジェクトの名称や各部を表すものであり、文章などのような長い文字列が付加されると逆に図の邪魔になる。そのため、付加できる文字列の長さに制限を加え、長すぎるラベルを付加できなくする必要がある。

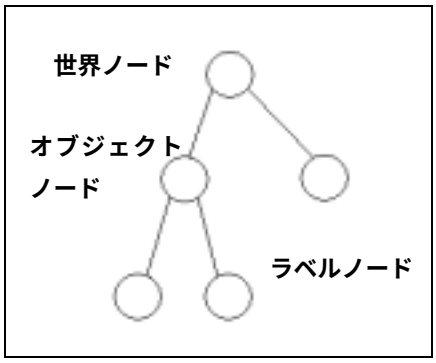


図 6：ラベルが定義された場合のシーングラフ

3.3 断面表示

断面が定義されると図 7 のようにシーングラフ中の世界ノードの子として定義される。すでに定義されているその他のオブジェクトノードは全て断面ノードの子ノードとなり、断面ノードは自分自身のコードの立体オブジェクトに対してのみ交差部分を求める。

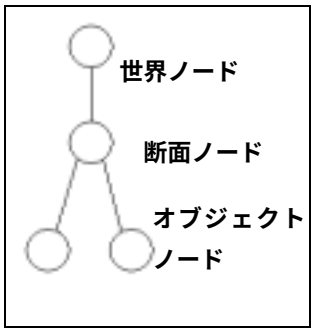


図 7：断面が定義された場合のシーングラフ

断面は断面上の一点と、その断面の法線ベクトルにより定義され、断面上の点、法線ベクトルの値を変えることにより移動・回転をさせることが可能となっている。

断面と多面体との交点は以下の手順により求められる。

1. 多面体の各面の稜線と切断面の交点を求める。
2. 各面の稜線と切断面の交点を結ぶ

以上の操作を全ての面に対して行うことで多面体の切断面を求める。

結局、多面体と切断面の交点を求めることは、それぞれの稜線と切断面との交点を求めることと等しいので、線分と平面の交点が求められればよい。

3.4 制約

制約は 3 次元仮想空間内に描画された複数のオブジェクト間に与えられ、一度制約が与えられると各オブジェクトは制約を維持するよう振舞う。

制約が与えられた場合のシーングラフを図 8 に示す。オブジェクト間に制約が与えられると、制約を与えたオブジェクト間の親として制約ノードが定義される。この制約ノードは自分の子であるオブジェクトノードに対してのみ、それぞれが与えられた制約を満たすように働きかける。

今、A、B というオブジェクト間に制約が与えられた場合、シーングラフ中のオブジェクトノード A、B の親ノードとして新たに制約ノードが定義される。この制約が与えられた状態で仮にオブジェクト A に変移が生じると、オブジェクトノード A は自らの親である制約ノードに対して変移を伝える。その変移を受け取った制約ノードはもう一方の子ノードであるオブジェクト B に対して、制約を維持するように働きかける。このようにして両者間の制約が保たれる。

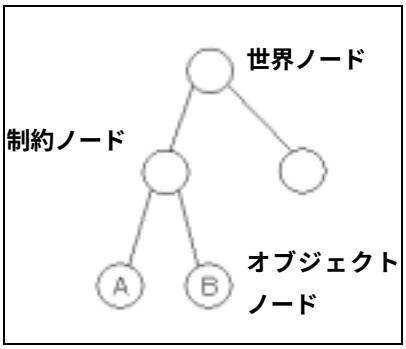


図 8：制約が与えられた場合のシーングラフ

４．実行結果

前項までの機能要件を満たす作図ツールを開発し、動作の確認を行った。

以下にモデリング操作の概要を示す。

4.1 描画画面

図 9 に実行画面を示す。実行中の画面は大きく分けてスクリーン部と操作部に分けられる。以下にスクリーン部と操作部の概要を示す。

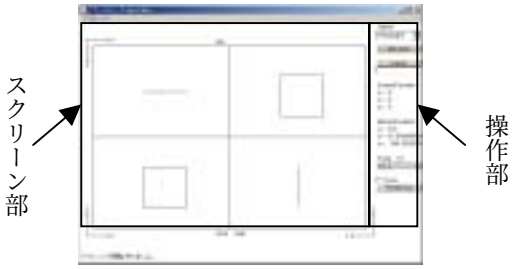


図 9：実行画面

・スクリーン部

3 次元空間を三面図表示している。左下が正面図、左上が上面図、右下が側面図となっている。右上の投影図は斜めからの斜投影面になっている。実際に図形を描き込むことができるのは正面図、上面図、側面図の 3 面である。この 3 面のうちのどれか 1 面に図形を描き込むことで、その図形が 3 次元空間上に定義され、他の投影面にも反映される。

・操作部

描画するオブジェクトの指定、編集、削除やラベルの設定や断面の表示などほとんどの操作を行う。以下に操作の概要を示す。

Object:新規生成するオブジェクトを選択する。
Label:ラベルの付加を行う。
Section:断面表示モードに入り、画面が断面の表示に切り替わる。
Deletion:描画されたオブジェクトの削除を行う。
Rotation:図形を回転するモードに切り替わる。
Scale:スクリーンの倍率を変更する。
Mouse:現在マウスが指している 3 次元空間上の座標を表示する。
Screen:投影面が置かれている座標を表示する。

4.2　新規オブジェクトの生成

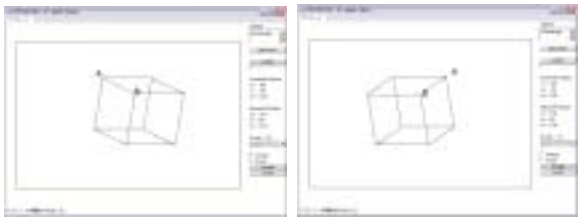
操作部の Object メニューから描画したいオブジェクトを選択し、描画画面上でマウスをドラッグすることで描画できる。オブジェクトを描画すると、オブジェクトは仮想的な 3 次元空間上に定義される。

現在､用意されているオブジェクトは線分､三角形、四角形、直方体、四角錐である。球面には対応していない。

4.3　図形の回転

前項の描画画面で実際に描画した図を回転させる画面。この画面でできることは図の回転のみで、図形の追加・編集・その他の作業を行うことはできない。マウスのドラッグを用いて回転を行う。

図 10 は図形を回転させた場合の画面である。図形の回転と共に頂点に付加されたラベルも移動していることが分かる。しかし、ラベルは設定された時点で配置される座標が決定しているので、文字同士の重なりやオブジェクト（線分）との重なりを回避することはできない。



(a)**回転前**　　(b)**回転後**

図 10：回転画面

4.4　ラベルの付加

描画されたオブジェクトに対してラベルを加える。図 11 にはオブジェクトにラベルを付加する様子を示している。(a)のように画面内に描画された図からラベルを付加したいオブジェクトを選択し、Label ボタンを押下する。その後その下にあるテキストボックスにラベル名を入力し、Enter を押すと、(b)のように選択されたオブジェクトにラベルが付加される。

付加されるラベルは、それぞれ 3 次元座標を取得し指定されたオブジェクトの周辺に表示される。

(a)**ラベル付加前**　　(b)**ラベル付加後**

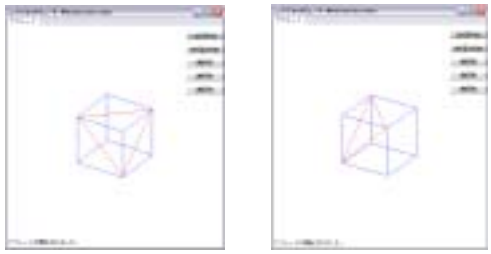
図 11：ラベルの付加

4.5　断面の表示

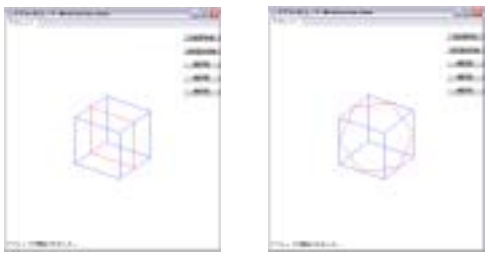
図形が画面内に描画された状態で Section ボタンを押すと、断面表示モードに切り替わる。

断面を表示している画面を図 12 に示す。Section モードでは描画された図形が青色の線で、切断面との交線が赤色の線で表示される。この状態では図形に対する操作は切断面の移動と図形の回転以外の操作は行えないようになっている。

(a)から(b)は図形を回転させた図、(a)から(c)、(d)はそれぞれ切断面を回転、平行移動させた図を示している。



(a)**回転前**　　(b)**回転後**



(c)**断面の回転**　　(d)**断面の移動**

図 12：断面表示

図形の回転については前に述べた図形の回転と同様の操作を行うことで可能となる。

切断面の移動は操作部のボタンによって行う。初期状態で断面は原点を含み、(1,1,1)を法線ベクトルとする平面となっている。切断面上の点を変更することで切断面の移動、法線ベクトルの値を変更することで面の回転を行う。

５．実装

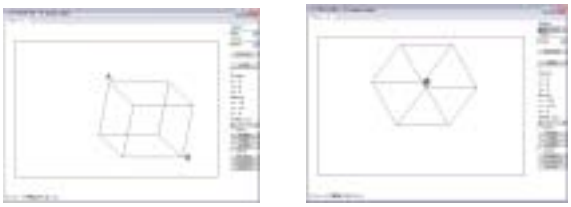
本研究の作図ツールプログラムは Java プログラム(Java2 Platform, Standard Edition v1.4.1)として実装されており、Java3D などは使用していない。今回の動作確認は Java アプレットを用いて行った。現段階の作図ツールはラベルの配置、断面表示については実装を完了しているが、制約の付加については未実装である。

６．考察

今回､形状表現にはワイヤフレームモデルを用いた。しかし､稜線表示だけでは表せる概念図に限界があり、複雑な図になると回転させながら見ても何を表しているか分かり辛いという場合が生じる可能性がある。今後、サーフェスモデルやソリッドモデルを利用し、陰面・陰線処理を行うことでより分かりやすい図形を描くことができると考えられる。

また、現段階の作図ツールには描画した図形を保存する機能が備わっていない。このような作図ツールには保存機能は必須の要素であるので、早急に対応する必要がある。

ラベルの配置に関して、図 13 の(b)のように図形を回転させる場合にラベルが重なりあう、または他の図形に邪魔されて見えなくなってしまうという問題がある。これを解決する方法としてはラベルが読めるように再配置することが考えられる。しかし、ユーザは通常図形を回転させながら図形がどのような構造になっているかを把握するため、ある角度からの図に対して文字を読むことができなくてもさほど気にはならない。逆にラベルの再配置を行うと、ラベルが指し示しているものが誤認されるなどユーザの混乱を招く恐れがある。



(a)**重ならない場合**　　(b)**重なった場合**

図 13：重なりあう例

また、切断面の表示については現在、定義される切断面は切断面上の点、法線ベクトルの値のみを保持しており、切断面の形状・範囲を指定することができない。そのため、切断面の延長上で図形と交差していない部分まで切断面が表示されることになる。また、切断面と交差する図形の面がちょうど一致した場合など、特殊な場合についても検討する必要がある。

現在切断面は凸多面体に対しては有効であるが、凹多面体に対しては切断面を表示することができない。事象を表現する上で凹多面体を用いることは十分に考えられるので、これにも対応していく。

前項で述べたように、現在制約を付加する部分についてはまだ未実装である。今後実装していくにあたって、オブジェクト間に制約を加える際には、加えられる制約がオブジェクト同士に対して有効かどうか、または与えられた制約に対しての両者の位置関係が必ず一意に求められるかどうかなどの判定を行い、条件が満たされない場合には制約を与えられないようにする必要がある。

最後に、できあがった作図ツールの提供方法について検討していく。現在、アプリケーションやアプレットなどの様々な方法で提供することが可能となっているが、本ツールはどのような形で提供していくかを検討していく。

参考文献

[1] “技術編　CG 標準テキストブック” CG-ARTS 協会

[2] 峯村　吉泰 “Java によるコンピュータグラフィクス” 森北出版株式会社

[3] 細部博史,「対話型 3 次元アプリケーションのための幾何制約解消法」, 情報処理学会論文誌, Vol. 44, No. 2, pp. 486–495, 2003.

[4] Hiroshi Hosobe, “A Modular Geometric Constraint Solver for User Interface Applications,” in *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology (UIST2001)*, CHI Letters, Vol. 3, Issue 2, pp. 91–100, ACM Press, 2001.