意味情報つきシーングラフによる仮想空間の統合化と表示方法

竹島　広人†　　富井　尚志‡

†横浜国立大学　大学院環境情報学府　情報メディア環境学専攻
‡横浜国立大学　大学院環境情報研究院

240-8501 横浜市保土ヶ谷区常盤台 79-7
E-mail: †take@arislab.dnj.ynu.ac.jp　‡tommy@ynu.ac.jp

あらまし　本研究では空間 DB と利用者との間のデータのやり取りとそのデータ構造について考え、以下の三つの課題の解決を目指す．一つ目は表示フォーマットの課題であり、状況に応じて適切なレンダリングツールに対応できるようなシーングラフを構築する．二つ目は意味情報の扱いについてであり、前述したシーングラフに形状データの他に意味情報も付加できるデータ構造を提案する．三つ目の課題は DB との連携であり、空間 DB 上に表現されたコミュニティ空間とエンドユーザに示されるシーングラフとの整合性を確保しなければならない．以上のことから本研究では、形状データの他に意味情報を記述できるようなシーングラフを提案し、それを用いてユーザと DB の間でデータのやり取りを行うシステムの構築を目的とする．また、XML を用いてシーングラフを記述する．

キーワード　空間 DB, XML,シーングラフ

Integration and Visualization Method of Virtual Space by Scene Graph with Semantic Information

Hiroto TAKESHIMA†　　Takashi TOMII‡

†‡Graduate School of Environment and Information Sciences, Yokohama National University

79-7 Tokiwadai, Hodogaya-ku, Yokohama 240-8501 Japan
E-mail:　†take@arislab.dnj.ynu.ac.jp　‡tommy@ynu.ac.jp

Abstract　In this paper, we consider about data communications between spatial database and end user, and the data structure, so we solve next three problems. First problem is about visualization method. We propose a scene graph that enable users to use multi-level rendering system. Second is about a management of semantic information. Therefore we propose the scene graph, which can express not only shape data but also semantic information. Third is about cooperation between DB and VR system. Because consistency between DB and the scene graph must be guaranteed, we design the scene graph as for general purpose using, and we construct a cyber community space by using the scene graph. In this paper, the scene graph is written in XML.

Keyword　Spatial DB，XML，Scene Graph

1. はじめに

近年、VR(バーチャルリアリティ)を用いたサイバーコミュニティの研究が数多くなされている[1]．ユーザはこの中でアバタという化身となり、仮想空間の中を自由に歩き回り他のユーザとコミュニケーションをとることができる[2]．しかし、ユーザが行えることはチャットやメールなど限られている．仮想空間をＤＢＭＳで管理すれば、ユーザは空間内での検索や、更新、シミュレーションなど多様な操作を行うことができる．このようなＶＲシステムとDBMSとのコラボレーションシステムとして VWDB[3][4]があげられる．これにより仮想空間を DB によって強力に管理することが可能となった．また、DB に蓄積されたオブジェクトに対するアクセス制御に関する研究も行われている[5].

さらに、現実世界の情報をＤＢＭＳで管理するシステムとしてRWDB[6]があげられる.

そこで本研究では、DB 上に表現された仮想空間を使ってそこに「コミュニティ」を構築することを考える．ユーザは、仮想空間の中を自由に動き回ることができ、他のユーザとコミュニケーションをとり、検索やシミュレーションなどを行うことができる．ここで以下の二つのことを考慮しなければならない．ひとつはプレゼンテーション(表示)である．状況によってユーザの求める詳細度は異なる．例えば、バーチャルモールなどでショッピングをしていて欲しい商品があった際、それをよりリアルに見たいという要求がある．このような場合を考慮して、状況に応じて詳細度を変えられることが望ましい．二つ目はコミュニケーションである．ユーザは仮想空間の中で複数のユーザとコミュニケーションをとることができるが、このとき現実空間と同じような感覚であることが望ましい．仮想空間にオブジェクト間の関係などの現実世界が持っている情報(意味情報)までも付加できれば、現実世界にいるのと同様の、もしくはそれ以上のことが実現可能となる．これらの二つのことを統合化しひとつのシステムとして構築するためには以下の段階を踏む.

- 外部(エンドユーザ)から見てどのようなデータのやり取りを行うか.
- 内部でどのような操作、実装をして実現するか.
- 概念的にどのような構造でモデル化するか.

本研究ではその中で特にユーザと DB との間のデータのやり取りとそのデータ構造について考え、以下の三つの課題の解決を目指す.

一つ目は表示フォーマットの課題である.上記のように、このようなコミュニティ空間では状況に応じて適切なレンダリングを行う必要がある．一般的にレンダリングの結果に対し写実性を求めれば求めるほど、多大な計算時間を要しウォークスルーのときのようなリアルタイムなレンダリングは難しくなる．そこで本研究では、ウォークスルーをするときはリアルタイムにレンダリングができる OpenGL を、よりリアルに見たいときはフォトリアリスティックなレンダリングができる 3ds max[7]を用いることとした．よって、このように複数のレンダラーを用いる場合、形状データはレンダラーに依存しないものでなくてはならない.

二つ目は意味情報の扱いについてである．我々は仮想空間に意味情報を付加することにより、仮想空間をより現実に近づけることを目的としている．よって、仮想空間を構成する要素として形状データだけでなく意味情報も含まれていなければならない．しかし、OpenGL や 3ds max などの既存の空間記述方式には意味情報を記述するような機能はない．そのため形状データのほかに意味情報も付加できるようなデータ構造を導入する.

三つ目の課題として DB との連携がある．コミュニティ空間の中では複数のユーザが自由に動き回っている．そのため、ウォークスルーの結果オリジナルとなる DB に更新を行う場合にはシーングラフとの整合性を確保しなければならない.

以上のことから本研究では、形状データの他に意味情報を付加できるようなデータ構造を提案し、それを用いてユーザと DB の間でデータのやり取りを行うシステムの構築を目的とする．また、このデータ構造を記述するために表現能力の高い XML を用いた.

以下 2 章では、我々の考えるコミュニティ空間について説明し、第 3 章では本研究の基本概念である ontology と mediator について述べる．また、第４章ではシーングラフに要求される構造について述べ、第５章でシーングラフの設計と実装について述べる．更に第 6 章で実装結果について述べ、最後にまとめと今後の課題について述べる.

2. 本研究におけるコミュニティ空間

本章では我々が提案するコミュニティ空間について説明する.

従来の仮想空間には現実世界が持っている意味情報は含まれていない．そこで我々は、従来の仮想空間に意味情報を付加することにより見た目だけでなく、空間そのものを現実世界に近づけることを目的としている．これによりユーザは、従来の仮想空間よりはるかに自由なことが行えるようになる.例えば、「このオブジェクトを収納できる場所を探せ」という検索を考える．このような検索を行う場合、「収納する」ということを認識しなければならない．これは従来の仮想空間では不可能なことである．また、別々の場所にいるユーザが仮想空間の中で現実に忠実に、あたかも皆がそこにいるかのように共同作業を行うことも可能となる．このように、現実世界ではできないようなこともできるようになる．仮想空間に現実世界が持つ意味情報をも付加できれば非常に有効であると考えられる.

図 1 に本研究のシステム構成を示す．本研究では DBMS に Microsoft 社の SQL Server 2000 を用いた．ユーザは DB に対して検索を行い、DB は検索結果をクライアントに返す．クライアントは受け取った結果からＸＭＬシーングラフを構築する．そして、そのＸＭＬシーングラフをレンダラーに応じて必要な形に変換してＶＲシステムに渡し仮想世界を表示させる.さらに、

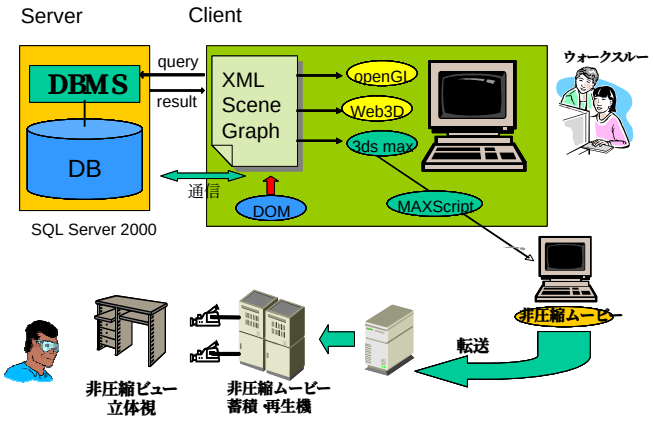

図 1　システム構成

非圧縮ムービー蓄積・再生機による仮想空間の立体視も視野に入れている．以下、このコミュニティ空間のことを高度コミュニティ空間と呼ぶ．

3. 高度コミュニティ空間における基本概念

この章では、高度コミュニティ空間における基本概念である ontology と mediator について述べる．

3.1. ontology

高度コミュニティ空間を実現するためには、仮想空間に意味を持たせなければならない．これに対して、ＶＲＭＬのような空間記述言語では、オブジェクトに対して注釈をつけることができる．しかし、意味情報はオブジェクトにのみ存在しているのではなくオブジェクトとオブジェクトの間にも存在している．このような意味情報すべてを、注釈をつけることだけで表現するのは困難である．そこで高度コミュニティ空間では、意味情報をオブジェクトから切り離し、意味情報自体をＤＢで管理する．ここで、ＤＢにデータを蓄積する際にはそのスキーマを考えなくてはならない．つまり、予めコミュニティ内に必要である意味情報を抽出し、意味情報間の関係を明示的に定義しなければならない．このような「概念化の明示的な規約」は ontology と呼ばれている[8]．本研究では、ontology はオブジェクトの「形状」に関する概念と、それが何に使われるかという「機能」に関する概念から構成されている[9].

3.2. mediator

ontology によって定義された意味情報を形状データと結びつけるために、本研究では mediator(仲介者)を用いることを提案している．オブジェクトの製作者一人一人が意味を勝手に決定すると意味に統一性がなくなり、意味に基づく検索が成り立たなくなる．そこで図 2 のように mediator と ontology の関係を統一しておけば、mediator を介することで意味に統一性をもたせ

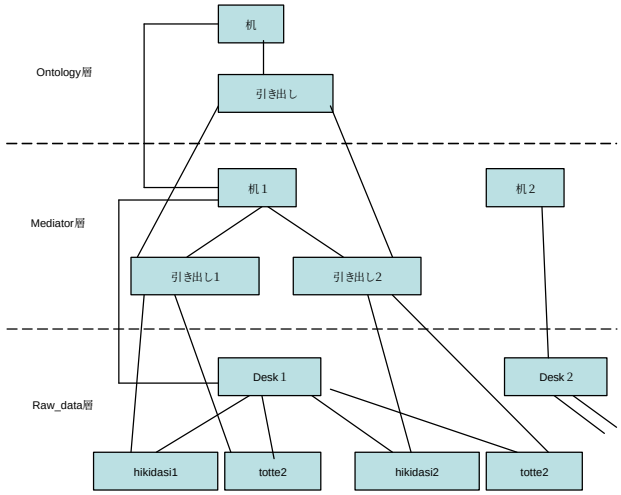

図 2　ontology-mediator-raw_data

ることができる．また、mediator は対応する形状データのポリゴンを近似した簡単なポリゴンデータを持っている．衝突検出などの空間検索の際にはこれを用いることにより計算量を減らすことができる．

3.3. スキーマ

高度コミュニティ空間におけるスキーマを図 3 に示す．本研究では対象とするコミュニティ空間をオフィスとした．前述したようにスキーマは ontology 層、mediator 層、物理層の 3 層に分かれている．更に ontology 層と mediator 層はそれぞれ「形状」と「機能」に分かれている．このスキーマの詳細な解説は文献[9]に譲る．

4. シーングラフに要求される構造

本章では、提案するシーングラフに要求される構造について説明する．

4.1. DB との対応付け

利用者からの空間に対する更新、検索はまずこのシーングラフに対して行われ、それがＤＢに反映される．そのため、シーングラフとＤＢの対応を保証する必要がある．そこで、それぞれのノードに id を持たせ、これを参照することによりシーングラフのどことＤＢのどこが対応しているかを明示的に示すこととした．

4.2. 形状データ

本研究では、状況に応じて複数のレンダラーを使い分けている．よって、形状データ部分はレンダラーに依存しないものでなくてはならない．既存のものでは、OpenGL では頂点リストと面ループを用いてオブジェクトが作られ、3ds max で提供されている MaxScript では基本図形にパラメータを与えることによりオブジ

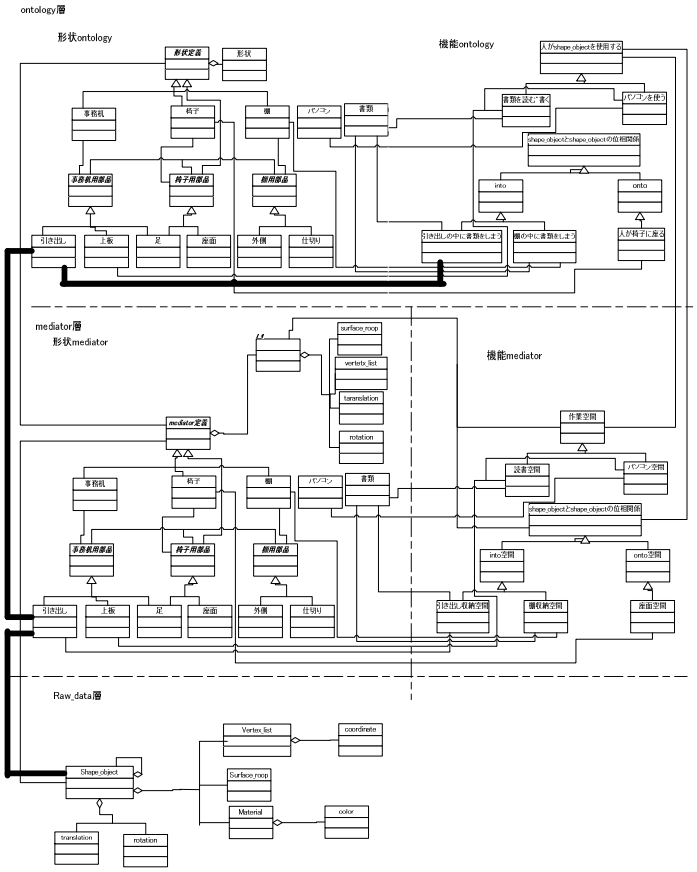

図 3　スキーマ

```
openGL
  GLdouble vertex[][3] = {
  { 0.0, 0.0, 0.0 },
   { 1.0, 0.0, 0.0 },
   { 1.0, 1.0, 0.0 },
        ⋮
  };
   int edge[][2] = {
  { 0, 1 },
   { 1, 2 },
   { 2, 3 },
        ⋮
  };
          ⋮
  glBegin(GL_LINES);
  for (i = 0; i < 12; i++) {
   glVertex3dv(vertex[edge[i][0]]);
  glVertex3dv(vertex[edge[i][1]]);
```

```
MaxScript

  op_a=gengon sides:6 radius:100 height:50”

 ”op_b=cylinder radius:50 height:60”

  op_c = boolObj.createBooleanObject op_a op_b 4 1”
```

図 4　OpenGL と 3ds max のデータ構造

ェクトが作られる．両者の具体的なデータ構造と特徴を図 4 に示す．本研究では、形状を表すデータとして基本要素である面ループと頂点リストを用いることとし、MaxScript でレンダリングする際には頂点リストと面ループから面を作りそれらを張り合わせることによってオブジェクトを作成することにした．

4.3. 意味情報の付加

本研究では、意味情報を付加することにより仮想空間を現実世界に近づけることを目標としている．そのためこのシーングラフには、意味情報が含まれていなければならない．そこで、このシーングラフにデータベースに管理されている意味データとのリンクを持たせることにより、仮想空間に意味情報を持たせることとした．これにより、「収納スペースを探せ」などの意味に基づく検索要求が来た場合、このリンクをたどりＤＢを参照することで検索を可能にする．例えば、「書類をしまう」という意味情報に基づいて形状オブジェクト「引出し」を探索することを考える．この場合、ＤＢ上のデータのつながりは図 3 のスキーマのクラス“引き出しの中にしまう”からクラス“shape_object”へのリンクパスをたどることにより表現できる．しかし、この構造は一般的に網構造であるため、ユーザ側のインターフェースでの取り扱いが難しい．そこで、ユーザ側には「ビュー」を提供するという観点から階層構造シーングラフを渡し、ＤＢ上のリンクの id によってその一意性を保証する．こうすることで、ＤＢは要素間が独立で様々な表現が可能な網(平坦)構造とし、ユーザ側にはそこから階層構造ビューを定義するということにした．

5. シーングラフの設計と実装

5.1. シーングラフの設計

本章では 3 章での考察に基づいて、以下の３つを満たし、3D レンダラーに依存しない汎用シーングラフを設計する．

1. DB との対応を示す id を埋め込むことができる．
2. 形状データを埋め込むことができる．
3. 意味情報とのリンクを埋め込むことができる．

以上の考察より、図 5 に設計したシーングラフを示す．以下、構成要素について述べる．(具体値については図 6 参照)

(a) id
　このノードにはＤＢＭＳによって割り振られた id が埋め込まれる．この id を参照することにより、ＤＢとシーングラフを一意に決めることができる．

(b) walkthrough_pass
　このタグには、OpenGL でウォークスルーした際のパスが書き込まれる。これを 3ds max に渡し、よりフォトリアリスティックに再現する．フォーマットは以下のように時点とその時点の視点座標および視点方向の組を列挙する．

　時間(秒)　X座標　Y座標　Z座標　X軸回転　Y軸回転　Z軸回転

(c) transform
　このタグには、オブジェクトの 3 次元位置

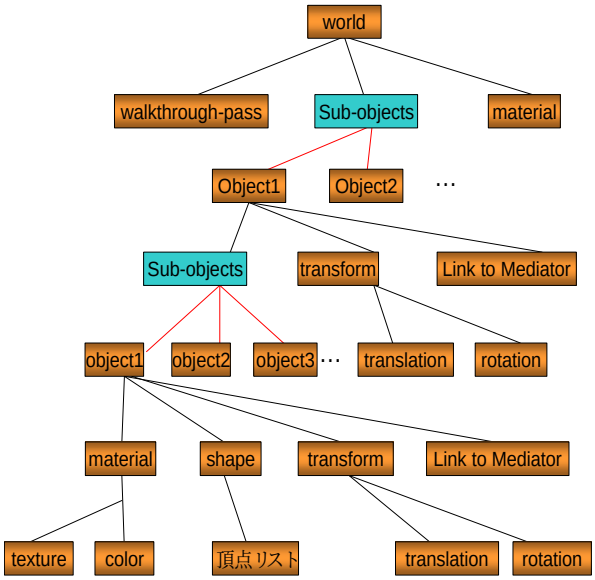

図 5　シーングラフの構造

```
<?xml version='1.0'?>

<world>
 <walkthrough_pass>                                                    (b)
  <pass>0.000000 -8.532304 0.000000 433.841873 0.000000 179.677315 0.000000</pass>
  <pass>0.015000 -8.314055 0.000000 433.443096 0.000000 179.450362 0.000000</pass>
                                  …
 </walkthrough_pass>
 <material>                                                            (e)
  <color>0,0,0</color>
 </material>
 <subobject>
  <object>
   <id>hikidasi@sita</id>    ・・(a)
   <transform>                                                         (c)
    <translation> 297.9 -76.9  102 </translation>
    <rotation> 0  0  0  0 </rotation>
   </transform>
   <LinktoMediator> 1003 </LinktoMediator>    ・・(f)
    <subobject>
     <object>
      <id>totte@shita</id>
      <transform>
       <translation>-1.198  12.32  265.5 </translation>
       <rotation> 0  0  0  0 </rotation>
      </transform>
      <LinktoMediator>-100 </LinktoMediator>
      <subobject>
       <object>
        <id>Line02_03</id>
        <shape>                                                        (d)
         <vertex_list>-33.47  0  5.729 ,-23.65  0  6.886 ,-23.65  0.0625   6.886 </vertex_list>
         <vertex_list>-33.47  0  5.729 ,-23.65  0.0625  6.886 ,-33.47  0.0625   5.729 </vertex_list>
                                  …
        </shape>
        <transform>
         <translation>-52.77  2.284 -7.972 </translation>
         <rotation> 1  0  0 -1.571 </rotation>
        </transform>
        <LinktoMediator>-100 </LinktoMediator>
        <material> 0.6039  0.8431  0.898 </material>
       </object>
       <object>
        <id>Loft01_03</id>
        <shape>
         <vertex_list>-48.1  0  10.65 ,-48.01  1.328  6.821 ,-48.19  0   6.823 </vertex_list>
         <vertex_list>-48.1  0  10.65 ,-47.92  1.328  10.65 ,-48.01  1.328   6.821 </vertex_list>
                                  …
        </shape>
        <transform>
         <translation>-1.717  1.43  0.45 </translation>
         <rotation> 0 -1  0 -3.142 </rotation>
        </transform>
        <LinktoMediator>-100 </LinktoMediator>
        <material> 0.6039  0.8431  0.898 </material>
       </object>
                …
```

図 6　XML document

(translation)と向き(rotation)が埋め込まれる．このタグは全てのオブジェクトに割り振られてり、それぞれ親オブジェクトに対する相対値とする．このように相対値とすることで、たとえば“机”を動かした場合“机”の相対座標のみを変えればよく、子ノードの座標を変える必要がなくなる．

(d) shape

一般に、表面形状データは頂点リストと面ループ群によって構成される[10]．このタグには、形状データとして頂点リストが埋め込まれる．子ノードとして vertex_list を持っており一つの vertex_list が一面に対応している．vertex_list の値として一面を構成する頂点リストを持ち、先頭から順番に点をつないでいけば面ができるようになっている．

(e) material

このタグには、マテリアル（表面のテクスチャ情報）が埋め込まれる．3ds max には多数のマテリアルが用意されているが、今回はよく使われる環境光、拡散反射、鏡面反射、発行係数、鏡面の強さ、反射、屈折の 8 種類をサポートするようにした．このうち、OpenGL がサポートするのは環境光、拡散反射、鏡面反射、発行係数、鏡面の強さの 5 種類である．

(f) LinktoMediator

このタグには、そのオブジェクトに相当する mediator ID が埋め込まれる．意味による検索時には、このＩＤをもとにＤＢをたどり対応する mediator を参照することができる．これにより意味による検索を可能にしている．

.

また、この構造に基づくインスタンスを図 7 に示す図 7 のシーングラフでは、次のような状態を表している．ひとつのワールドの中に二つの机(object 1 と object2)があり、それぞれ事務机、会議テーブルとする．事務机はひとつの引き出しと 4 本の足を持ち、天

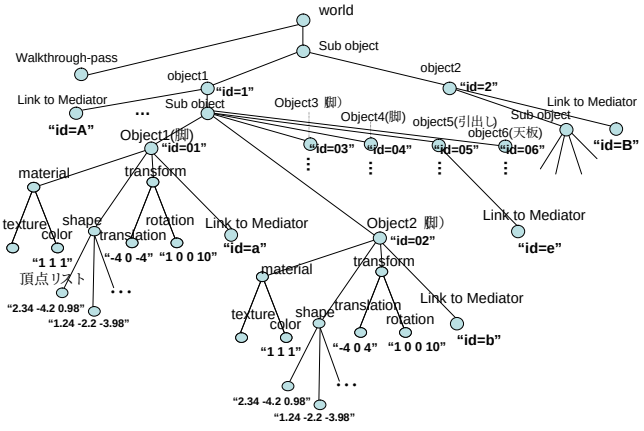

図 7　シーングラフのインスタンス

板、足共に直方体である．一方会議テーブルのほうは 4 本の足は持つが、引き出しは持たず、天板、足共に円柱である．ここで事務机の二つの足オブジェクトに注目すると、この二つは別の足であるにもかかわらずまったく同一の面ループと頂点リストを持っている．これは、二つの足は異なるインスタンスであるが形状がまったく同じであるためであり、つまり部品の再利用が可能であることを示している．これにより、ＤＢに蓄積するデータ量を減らすことができる．また、それぞれのノードには id がふられており、この id を用いて DB を参照することができる．また、それぞれの形状 object は mediator へのリンクを持っており、このリンクをたどることにより意味に基づく検索、更新が可能となっている．

5.2. シーングラフに対する操作

この節では、まず検索･更新が起きた際のシーングラフとＤＢとの連携について説明し、次にシーングラフを介した更新・検索を行う際、どのような手順を踏むかをスキーマ上で説明する．ここで対象とする世界に、前述した図 7 のシーングラフインスタンスを用いた．

5.2.1. 外部操作

ここではシーングラフがＤＢとどのように連携するかについて述べる．(図 8 参照)

まず、データが表示されるまでの流れを説明する．ユーザによってＤＢに対して複数の検索が出される．ＤＢは検索結果として複数のテーブルをクライアント側に返す．それらのテーブルはクライアント側でシーングラフ作成プログラムによりシーングラフに変換される．そしてそのシーングラフをレンダラーに応じて

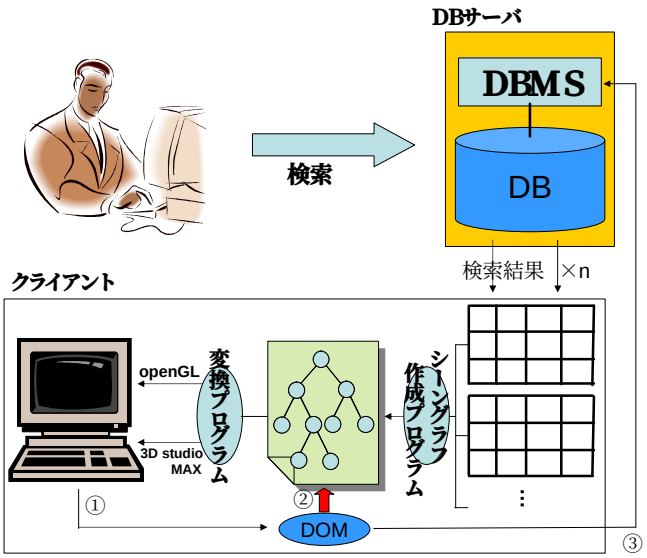

図 8 シーングラフとＤＢの連携

必要な形に変換し、表示機に渡す．本研究では、ＸＭＬ文書の変換に DOM を用い、ＸＭＬツールキットに ORACLE 社の XDK for C[11]を用いた．

次に、更新が起きた場合の流れを説明する．手順は以下の通りである．

1. ユーザから更新情報（形状 object id,更新された値）が送られる．（①）
2. シーングラフ内で変更されたところが書き換えられる．（②）
3. シーングラフ中から送られた形状 object id をもつオブジェクトに対応する mediator id を抜き出し、mediator id と更新された値を DB 側に送る．(③)
4. 送られてきた情報をもとに DB を更新する．
5. 更新された後、新たなシーングラフを他のユーザに送る．

5.2.2. 内部操作

ここでは、ＤＢ上でどのような手順で検索・更新が行われるか説明する．

まず、「書類を収納するスペースを探せ」という検索を考える．この手順は以下の通りである．(図９参照)

1. まず、ontology 層の機能部分から「書類を収納する」を見つける．(①、②)
2. 1.の結果をもとに③、④のテーブルをたどりその機能を有する形状概念を見つける．(引出し 1～3 と棚 1～3 が見つかる)
3. 2.の結果をもとにテーブル⑤より意味 ID を見つける．(0001～0003)
4. 3.の結果をもとにテーブル⑥より mediator ID を見つける．(e,g,i)
5. シーングラフの Link to Mediator 要素の id と 4.で得られた mediator ID を比較して一致するものを選ぶ．最終的に形状 object ID=05 の引出しが得られる．

よってシーングラフ内の object ID=05 が DB 内で「収納する」ことを表すことが確認できた．

次に、「事務机の引き出しを開く」という更新を考える．この場合には図 4 中の object タイプの値を参照し、そこからその形状 object に対応する mediator を参照する．最後にその mediator に対応する相対座標の値を更新することにより DB 内の「引き出しを開く」を実現する．具体的な手順は以下の通りになる．（図 9 参照）

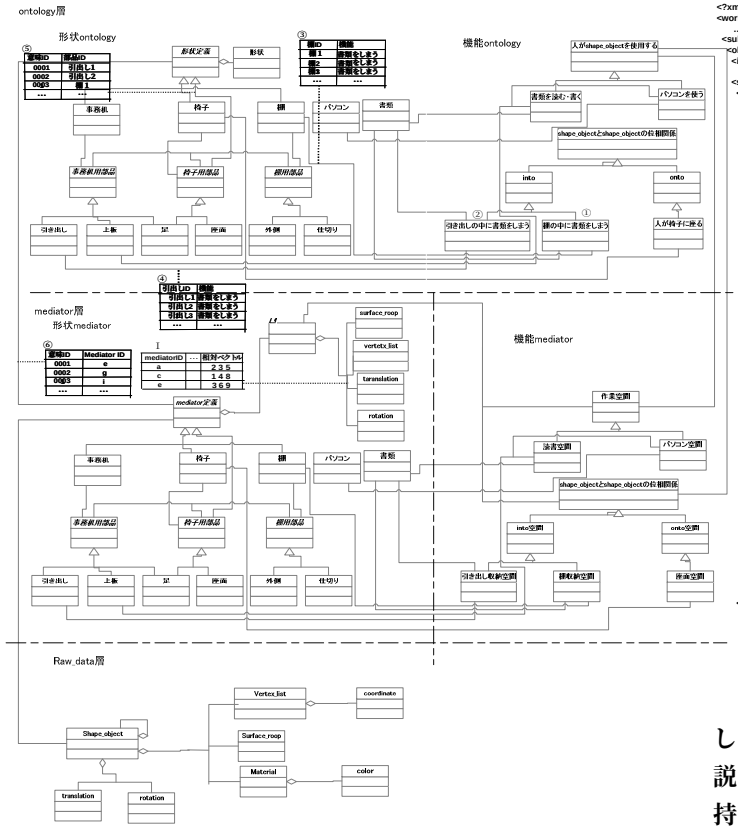

図９　検索・更新の手順

1. まず、シーングラフから事務机の mediator ID を見つける．(mediator ID=e が得られる)
2. mediator 層を参照し、mediator ID から対応する属性値を選び出す．（Ｉ）
3. 選択した属性値を、「開けた」あとの座標値に更新する．

以上のように「開ける」という意味操作を、DB の値を更新することで実現できることを示した．

6. システムの実装

この章では、実際に実装したシステムについて述べる.実装環境として、以下の装置を用いた．

実装ホスト：PC/AT 互換機(CPU:Pentium4 2.4 メモリ:512MB)
OS：Windows XP Professional
開発環境：Visual Studio .NET(VC++ .NET)
OpenGL1.1
Glut3.7.2
追加ソフトウェア：3ds max version5

6.1. シーングラフからOpenGL への変換

ここでは、シーングラフからデータを OpenGL に渡し表示する手順を述べる．まず,ノードを調べる手順を説明する．まず、ルートノードから調べ、子ノードを持っていないか調べる．持っていれば子ノードに対して、document でみて上から順番に子ノードがないか調べ、以後これを再帰する．よって,ノードをたどる順番は図 10 のようになる．ただし,vertex_list はポリゴンの数だけあるため,すべて調べていると時間がかかりすぎる．そのため、shape ノードの子ノードは調べない．このようにしてノードを調べていき,必要な情報を配列に入れて OpenGL 側に渡した．このようにした理由は、OpenGL がウォークスルーをリアルタイムに行うために予め頂点リストなどを display-list に登録しているためで,最初に全ての情報を OpenGL 側に渡す必要があるためである．

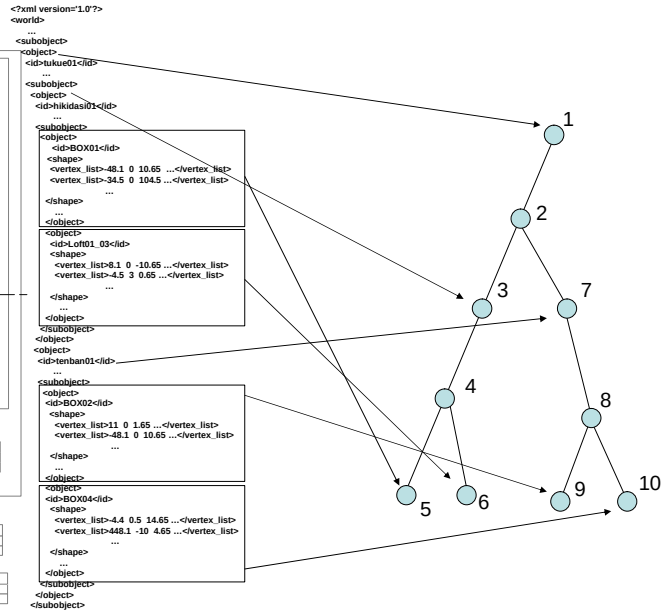

図 10　ノード探索の手順

以上の手順で実際に DB に格納されているデータを OpenGL で表示したものが図 11 ある．

6.2. シーングラフから3ds max への変換

ここでは、シーングラフから 3ds max にデータを渡す方法を述べる．データを渡す際にはファイルを用いた．このファイルを作成する際も上と同様のアルゴリズムでノードを探索し、必要な情報をファイルに書き込んだ．また、ファイルの読み込みからレンダリングまでの一連の流れを、MaxScript により自動化している．

図 12 に、3ds max によるシステム利用の様子を示す．

6.3. ウォークスルーの再現

ウォークスルーを再現する手順は次の通りである．まず、ウォークスルーが始まると OpenGL がウォークスルーパスを吐き出す．ウォークスルーが終わると吐

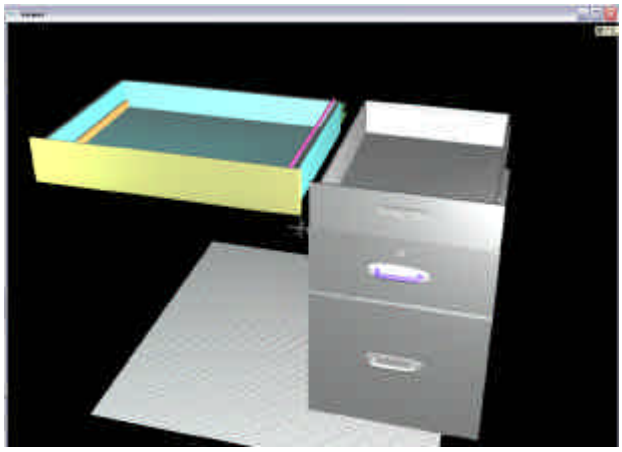

図 11　OpenGL による表示結果

図 12　3ds max によるシステム利用の様子

き出されたパスを配列に入れて返し、それをシーングラフに埋め込む．そのあとにシーングラフから 3ds max 用ファイルを作り、3ds max に渡す．そのファイルを MaxScript で読み込み、そのパスに従ってカメラを動かすことによりウォークスルーを再現している．(ここで、OpenGL と 3ds max では座標系が異なるため、カメラを動かす際には 3ds max の座標系に変換している)また、そのウォークスルーパスをＤＢに蓄積し、そのシーンをいつでも見ることができるようにすることも考えている．

実際に上の手順でウォークスルーの実験を行い、ウォークスルーが再現できていることを確認した．

7. まとめと今後の課題

我々が提案するコミュニティ空間を実現するために、シーングラフに要求される機能として、シーングラフと DB との対応が取れていること、意味情報が付加されていることがある．本稿ではシーングラフの各ノードに id をつけることにより、DB との対応がとれ、検索や更新が一意に行えることを示した．また、mediator へのリンクをシーングラフに持たせることにより、意味に基づく検索や更新が行えることを示した．更に、提案したシステムの実装を行った．

今後の課題としてはまず,ひとつのシステムとして完成させることを考えている．検索・更新の発行から，表示・ＤＢの更新までを完全自動化する必要がある．また、リアルタイムにインタラクションを取れるようにする必要もある．実用的なシステムを構築することが今後の課題といえる．

謝辞

本研究の一部は文部科学省科学研究費補助金(課題番号 40313473)および、財団法人横浜工業会研究助成による.また実装にあたり、「日本データベース学会・マイクロソフト株式会社共催 2002 年度データベース研究支援プログラム」の支援を得た.ここに記して謝意を表する．

文　献

[1] 松田晃一，上野比呂至，三宅貴浩“パーソナルエージェント指向仮想社会「PAW」の評価”電子情報通信学会論文誌 ,D- Ⅱ ,J82-D- Ⅱ ,No10, pp.1675-1683,Oct.1999

[2] C.Morningstar and F.R.Farmer ”The Lesson of Lucasfil’s Habitat” In CYBERSPACE:First Step,M.Benedikt,ed,The MIT Press,1991

[3] 渡辺智恵美，大杉あゆみ，佐藤こずえ，増永良文“仮想世界データベースシステムにおけるスキーマ・ドメイン定義言語の設計”2001 データ工学ワークショップ

[4] 渡辺知恵美，増永良文：仮想世界データベースシステムにおけるマルチモーダル問合せ言語の設計に向けて，情報処理学会データベースシステム研究会報告，Vol125，No.71, pp.289-296 2001

[5] 上浦真樹，衣田和也，田島敬史，田中克己“３次元空間データベースにおけるデータモデルとアクセス管理機構について”情報処理学会研究報告，96-DBS-109,pp.215-222,Jul,1996

[6] 富井尚志，小林みな子，有澤博“仮想 CG 空間へのマッピングによる現実シーンデータベースの設計”電子情報通信学会論文誌,D-1,Vol.J82-D-I, No.1, pp.211-222, Jan,1999

[7] http://www.discreet.com/products/3dsmax/

[8] 溝口理一郎“オントロジー研究の基礎と応用”人工知能学会誌,Vol.14,No６,Nov,1999

[9] 岡田直也、富井尚志“ontology を用いた空間形状データの意味情報モデリング”データ工学ワークショップ,Mar,2003

[10] 吉田典正，北嶋克寛　”ポリゴンモデルを対象とする視界に依存する適応度表示”　電子情報通信学会論文誌 ,D- Ⅱ ,Vol.J83-D-Ⅱ,NO.6,pp.1507-1515,Jun,2000

[11] http://technet.oracle.com/tech/xml/xdk_c/content.html