データベースシステム　125－11
（２００１．７．１７）

XMLを用いたマルチメディア構造化文書STOICの意味表現能力とその評価

越前谷健二 †　松田一章 ††　富井尚志 †††　有澤博 †††

† 横浜国立大学院 環境情報学府
†† 横浜国立大学大学院工学研究科
††† 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:{echizenn,cross, tommy, arisawa}@arislab.dnj.ynu.ac.jp

あらまし

本稿では、我々が住んでいる実世界の意味情報までをも含めた情報を配信することを目的として、これを実現するために必要な新しい情報キャリアとして時空間オブジェクト情報キャリア STOIC を提案する。また STOIC は情報キャリアとして配信されるため、データ構造が人間にとっても分かりやすくなければならない。そこで構造化文書によって表現することを考え、XML を用いて STOIC を表現する。また、このようにして設計・記述された STOIC の有用性を示すために、CG を用いた表示システムを実装する。

Semantics expression of MultiMedia structured document STOIC with XML and its estimation

Kenji Echizennya　Kazuaki Matsuda　Takashi Tomii　Hiroshi Arisawa

† Graduate School of Environment and Information Sciences, Yokohama National University
†† Division of Electrical and Computer Engineering, Faculty of Engineering, Yokohama National University
††† Faculty of Environment and Information Sciences, Yokohama National University
79-7, Tokiwadai, Hodogaya-ku, Yokohama 240-8501 JAPAN
E-mail:{echizenn, cross , tommy, arisawa}@arislab.dnj.ynu.ac.jp

Abstract

In this paper, for the purpose of sending the information even including the semantic information on the real world where we live, we propose Spatio Temporal Object Information Carrier STOIC as the novel information carrier in order to realize sending the information even including the semantic information. Since STOIC is sent as information carrier, its data structure must be intelligible not only for computer but also for human. Therefore, we have considered expressing STOIC by the structured document, we decide to express STOIC using XML. Moreover, we construct the visualization system as CG in order to show its usefulness.

1　はじめに

我々が住んでいる 3 次元空間と、その空間中の「もの」とその「振舞い」によって表現される実世界の情報は非常に様々な「意味」を持っている。本稿ではこのような実世界の情報を、計算機上で配信することを目的とした、情報の表現手法を提案し、それを評価した。

このような実世界の情報を計算機上で扱う技術として、コンピュータグラフィックス (CG) があげられる。しかし CG は視覚的な情報を計算機を用いて表現することを目的としている技術なので、配信される情報というものは、その 3 次元空間の空間的な情報、「物体」の 3 次元的な形状、また、ある時点における位置変化などの、視覚的な情報のみとなってしまい、実世界を忠実に表現しているとはいえない。例えば、実世界ではある時点のオブジェクト A と別の時点のオブジェクト A はオブジェクト A という関係において、同一であるが、CG では別々のオブジェクトとして表現してもよく、その同一性を保証しない。また、人間を表すオブジェクト A がある時点で物体 B をもちあげたことを表現するのに、実世界では、物体 B はオブジェクト A の手に従属して動くという制約を満たし動いているにもかかわらず、CG では、手と物体 B を別々に同じ方向へ動かし、視覚的には持っているように見えるが、そのデータには全くそのような意味的な情報は表現されていない。そこで我々は 3 次元空間と、その空間中の「物体」の 3 次元的な形状、また、その時点による位置変化などの視覚的に捉えることができるものだけでなく、その「物体」が持っている属性や、「物体」もしくはいくつかの「物体」の時間的変化によって表現される概念的なものなどの、オブジェクトやオブジェクト間の意味情報までをも含めた実世界を表現する手法を提案する。以下このような実世界のことを時空間と呼ぶ。

また、我々はこの時空間の情報を配信することを目的としているので、これを記述する際に以下の 2 つの条件を満たしていなければならないと考えた。

- データ構造が人間にとっても理解しやすい
- 実世界の情報を忠実、また、一般的に記述してある

まず、これらのデータは人間によって扱われるものなので、人間が見た時に分かりやすくなければならない。そのため、この情報を構造化されたドキュメントによって記述することにした。ここでいうドキュメントというのは、バイナリーやアプリケーション依存の文字・数値の羅列ではなく、テキスト化された文書のことである。構造化することによって、その構造自体に意味を付加し、人間にも計算機にもその内容が把握しやすくなる。また、2 つめとしてはだれがみてもおよそ一緒である情報の表現によって、表示したりする環境、アプリケーションに依存しなくなると考えた。

以上のような考えに基づいて、本稿では対象とする一つのシーンをその内容までをも含めて表現し、伝えることができるキャリアを時空間オブジェクト情報キャリア (Spatio Temporal Information Carrier:STOIC) を提案し、STOIC を用いたオブジェクトやオブジェクト間の意味情報までをも含めた実世界の情報を表現するシステムの構築を試みた。我々はこの STOIC をそのままデータベースに格納することを考えているのではない。データベースには、時空間を構成する要素がその素材に分けて様々な見方ができるような不偏的な形で記述されている [1][2]。STOIC はそれらの素材を組み合わせて再構成される、ある一つのシーンを記述し、配信するものである。この STOIC の実現手法として、その枠組を XML スキーマを用いて設計、実際のシーンをそれに従った XML 文書で記述した。XML を使用した理由としては、XML は自由に構造を定義できるマークアップ言語であり、その表現能力も高く、その構造やタグによっても意味を表現できるため、STOIC の構造を定義し、記述する際に有効である。様々な業界でデータ、もしくは構造化文書を記述する際のデファクトスタンダードになりつつある。また、特別なソフト・エディタが必要がないため、どんな計算機でもデータを見ることができる。さらに、様々な支援ツールや API が公開されているなどの理由があげられる。

本論文の構成は、以下のようになっている。第 2 章では、実世界の情報とは、第 3 章では、STOIC の説明と XML スキーマによる設計、また、その意味表現能力の評価、第 4 章では、STOIC の視覚化システムの実装とその有用性、第 5 章では、まとめ、について述べる。

2　時空間情報のとらえかた

2.1　時空間情報の表現と伝達

我々の目的は様々な「意味」を持った実世界のある時間的、空間的に閉じた一部分 (シーン) を記述し配信・通信することである。このシーンというものは 3 次元の空間であり、このような 3 次元的な空間のデータを扱う既存の技術として、CG が挙げられる。CG は 3 次元空間中の物体の位置や形状、または、時間的に変化する位置などを、視覚的に表現するための技術で、その空間中で表現される「意味」を表現することを目的としていない。例えば、ある時点のオブジェクト A と別の時点のオブジェクト A は実世界ではオブジェクト A であることで同一であるが、CG ではこれらが同一のものであることを保証しない。また、人間を表すオブジェクト A がある時点で物体 B をもちあげたことを表現するのに、実世界では、物体 B はオブジェクト A の手に従属して動くという制約を満たし動いているにもかかわらず、CG では、手と物体 B を別々に同じ方向へ動かし、視覚的には持っているように見えるが、そこには全くそのような意味的な情報は表現されていない。我々が考えている、様々な「意味」までも含めて表現するには CG の表現能力は不十分である。

また、インターネット・モバイル通信の普及にともない、映像や 3 次元コンテンツの配信が盛んに行なわれるようになった。配信するデータは不特定多数のユーザであるので、要求される内容も様々である。例えば、配信された情報を CG を用いて表示する場合、まずユーザの計算機環境の差によって同じ「球」を表示するにも、ある人は非常に高性能な計算機を使っているので、非常に詳細な球を表示したいと要求し、また別の人は性能があまりよくない計算機を使用しているので、計算機の演算能力に合わせた、詳細度をおとした球を表示させたいと要求することも考えられる。このような場合、配信する側がその要求に応えるために、同じシーンを表すいくつものコンテンツを、その詳細度を変えて用意しておくのは効率的ではない。さらに、ある人は高価なシミュレーションソフト用のデータ形式で、別の人は既存の CG 表示用言語用のデータ形式が欲しいというような、データ形式を指定した要求も考えられる。このように、概念的に同じシーン、同じオブジェクトなどを表現しているにもかかわらず、それぞれの要求するデータ形式や、表示形式のコンテンツを用意するというのもまた同様に効率的ではない。

以上のような問題点を解決するために、我々は、伝えたい情報というものは環境やアプリケーションによらず一つであって、その情報がどのようなデータ形式で処理されるかがアプリケーションによって決まるべきだと考える。また、そのデータ構造は計算機だけでなく、人間にとっても理解しやすくなければならないと考え、実世界の情報を忠実に、また一般的に表現するデータをテキストベースの構造化文書によって記述することにした。

2.2　時空間情報の意味表現

前節では我々が表現しようとしている実世界の情報は様々な意味を含んでおり、3 次元空間の見かけだけの情報のデータを扱う CG 技術では表現し切れないことを述べた。このような問題を解決するために単に CG を拡張していくといった方法ではなく、もう一度実世界というものを概念的にとらえ、新しい情報記述について考える必要がある。

まず、我々はおおよそ実世界を表現するために次のように考えた。まず、実世界というものをシーンに存在する「もの」とその時間的変化によって表現する。ここで、シーンに存在するオブジェクトはそのシーンで存在しなくなるということはなく、必ず存在し、また同一の「もの」である。シーンに存在する「もの」はそのシーンで起こる事象に対して影響を持たないオブジェクトと影響をもつオブジェクトに分けて考える。前者を「場」、後者を「オブジェクト」と呼ぶ。この「オブジェクト」の時間的変化を「変化」と呼び、この「変化」もしくは、「変化」間のつながりによって表現されるものを「イベント」と呼ぶ。実世界を表現するにはおおよそこれらの要素が必要だと考えた。

これらの要素に対してさらに同じような意味を持っているということを表現するために、タイプの同一性について考える。例えば、A という「オブジェクト」が人間である、と表現した場合、そのオブジェクト A は人間であるという意味を持ち、手や足などを持っているという情報を持つ。また、オブジェクト B を人間と表現した時、これらは同様の情報を持っているはずである。このタイプという概念をもちい

ることによって、同じカテゴリーに分けられているものは同じような情報を持っているという意味を表現できる。このタイプという概念を用いあらかじめ概念的に人間というタイプを定義しておくことで、CG では表現できなかった同一性などの意味をどのように表現するかについて述べる。例えば、CG を用いて人間を二人表示するデータを記述する場合を考える。CG では、一人目の人間を記述した後、二人目を記述する際に、その部品がほぼ同様なので同じデータで記述しようと考えたとする。このような場合、それらがどちらも人間であるので同じデータで記述したということは表現されず、それらが同じ人間を表現しているという「意味」がなくなってしまう。ここで、タイプという概念を用いると、タイプ同士の同一性から、それらが同じ人間であることを表現できる。次に、インスタンスの同一性に対して考える。例えば、ある時点のオブジェクト A と別の時点のオブジェクト A が同じオブジェクト A であるということを示すことは重要である。これは参照を用いてその同一性を示す。例えばサッカーボールがある時点でパンクしてまったく別の形状になったとしても、それは同一のサッカーボールである。この同じサッカーボールであることを示すために参照を行なう。

3 STOIC の導入と設計・構築

本章では、意味情報を含んだ実世界の情報を表現し伝達する時空間オブジェクト情報キャリア (Spatio Temporal Information Carrier:STOIC) を提案する。さらに、XML を用いた STOIC のを設計・構築するための、記述方式を説明する。

3.1 構造化文書 XML と STOIC

本節では我々が提案する STOIC を導入するために、XML を用いた STOIC の設計・構築を行なう。

XML は自由に構造を定義できるマークアップ言語であり、自由なタグ構造を定義できる。そのため前章で述べたような構造や要素名によって意味を表現する構造化文書の枠組を定義でき、さらにその定義した構造に従った構造化文書を記述できる。さらに、XML はその特徴として、強力なリンク機構があり、このリンク機構を用いることによって、様々な参照を行なうことができ、STOIC でのインスタンスの同一性を示す上で、非常に有効である。

また、様々な業界で交換・配布用データを記述する際のデファクトスタンダードになりつつあり、さらに、様々な支援ツールや API が公開されているなどの理由があげられる。以上のような観点から STOIC を設計・構築する上で、XML は非常に有効であると考えられる。

3.2 XML スキーマを用いた STOIC の設計・構築

本節では、前章までに述べた STOIC の枠組を XML スキーマを用いて設計し、その一シーンを XML インスタンスで記述する。

XML スキーマとは XML 文書の構造を定義する一手法で、他にも DTD など文書の構造を定義する手法はあるが、XML スキーマはその XML 文書の構造を定義する際、非常に様々な制約まで書くことができ、その表現能力も高い。

タイプの同一性を保証することで、そのインスタンス同士に同じ意味を持たすことができると述べた。それらが同じであるということを示すと述べた。XML スキーマでどのようにタイプを記述するかを説明するため、例として図 1に示すような人間タイプを XML スキーマによって記述する。これは人間を、頭や右

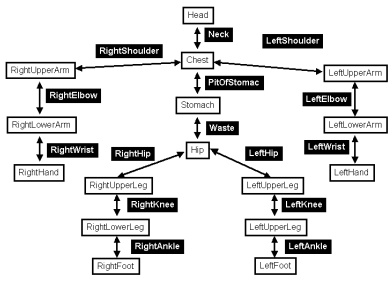

図 1: 人間タイプ

上腕などの部分 (パーツ) と、肘や腰などのパーツ同士のつながりを表現する関節 (ジョイント) の集合によって表現している。このように人間を表現するタイプを記述した XML スキーマを図 2 に示す。これは「オブジェクト」のタイプが人間タイプである時、その要素としては、人間の名前、人間のパーツ、人

```
<complexType name="human">
  <xsd:element name="HumanName"
                type="xsd:string"/>
  <xsd:element ref="HumanParts"/>
  <xsd:element ref="HumanStrcture"/>
</complexType>
<xsd:element name="HumanParts">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="Hip" type="parts"/>
      <xsd:element name="Stomac" type="parts"/>
      <xsd:element name="LeftUpperLeg"
                    type="parts"/>
          以下同様なパーツの定義なので省略
    </xsd:all>
  </xsd:complexType>
</xsd:element>
<xsd:complexType name="parts">
  <xsd:element ref="shape"/>
  <xsd:element ref="center"/>
</xsd:complexType>
<xsd:element name="HumanStructure">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="RightHip" type="joint"/>
      <xsd:element name="LeftHip" type="joint"/>
          以下同様なジョイントの定義なので省略
    </xsd:complexType>
  </xsd:all>
</xsd:element>
<xsd:complexType name="joint">
  <xsd:element ref="JointPosition"/>
  <xsd:element ref="connect"/>
</xsd:complexType>
```

図 2: XML スキーマの人間タイプの表現

間の構造を表すHumanName要素、HumanParts要素、HumanStructure要素を持つ。HumanParts要素は図1で示したパーツの集まりを要素として持つ。それらのパーツ要素は、その形状を表現するためのShape要素と、その中心がどこにあるかを示すためのcenter要素を持つ。HumanStructure要素はその子要素として、図1で示したジョイントの集まりを要素として持つ。それらのジョイント要素は、どの二つのパーツがつながっているかという参照と、そのパーツのつながっている位置を表すJointPosition要素と、connect要素をもつ。このようにXMLスキーマでタイプを定義することによって、例えばオブジェクトAとオブジェクトBは人間という意味で同一であるというタイプの同一性を示すことができる。

図2で記述したXMLスキーマを元にして、XMLインスタンスを記述することを考える。記述した例を図3に示す。

これは人間の構造・形状を、頭や胸などのパーツの集合と、それらを結び付けるためのジョイントの集合によって表現する際に、ジョイントがどのパーツとどのパーツを結んでいるかを、パーツ集合中の要素への参照を用いることによって表現している。このように参照を用いることによって、例えば、オブジェクトAの右肩とつながっている「右上腕」とオブジェクトAの右肘につながっている「右上腕」が同一であるというインスタンスの同一性を示せる。

3.3 XMLを用いたSTOICの記述能力の評価

これまでに時空間の情報を表現するためにはその視覚的な情報だけでなく、オブジェクトやオブジェクト間のつながりによって表現される意味情報が必要であることを述べた。そして、その枠組としてSTOICを提案し、XMLスキーマを用いて設計した。本節ではこのSTOICの記述能力の評価を行なう。

実世界の情報を表現するためにはオブジェクト同士の同一性や、オブジェクトの構造が持つ意味、シーンで起こった事象の意味、事象とオブジェクトの動きの関係が挙げられる。

STOICではXMLスキーマでのタイプの定義とインスタンス同士の参照を用いることによって、そのタイプの同一性、インスタンスレベルの同一性を表

現できる。オブジェクトの構造が持つ意味やシーンで起こった事象の意味、その事象とオブジェクトの動きの関係についてはさらなる考察が必要だと考えられ、今後の課題となる。

```
<object>
  <HumanName>A</HumanName>
  <HumanParts>
    <Chest>
      <shape>
        <Box>
          <width>0.4</width>
          <height>0.3</height>
          <length>0.2</length>
        </Box>
        <center>0 0 0</center>
      </shape>
      <center>0 0 0</center>
    </Chest>
    <LeftUpperArm>
      <shape>
        <Cylinder>
          <radius>0.1</radius>
          <height>0.35</height>
        </Cylinder>
        <center>0 0 0</center>
      </shape>
      <center>0 0 0</center>
    </LeftUpperArm>
          以下同様にパーツのインスタンスなので省略
  </HumanParts>
  <HumanStructure>
    <LeftShoulder>
      <JointPosition>2 0.1 0</JointPosition>
      <connect>
        <locator href=
          "origin().ancestor(1,JointPosition)"/>
        <locator href=
          "origin().ancestor(1,Chest)"/>
      </connect>
      <JointPosition>-0.1 0.17 0</JointPosition>
      <connect>
        <locator href=
          "origin().ancestor(1,JointPosition)"/>
        <locator href=
          "origin().ancestor(1,LeftUpperArm)"/>
      </connect>
    </LeftShoulder>
          以下同様にジョイントのインスタンスなので省
略
  </HumanStructure>
</object>
```

図 3: XML インスタンス

4 STOIC と表示システムの実装

4.1 三次元グラフィックスと STOIC

本節では STOIC を用いて表現されたシーンを三次元コンピュータグラフィックスを用いて視覚化する実装システム述べる。

前章では STOIC は XML スキーマによって表現し、そのインスタンスであるシーンは XML インスタンスとして記述した。XML という構造化文書でシーンを記述することにより、テキストエディタでその XML インスタンスを見ることでシーンの概要を理解することはできる。しかしながら、実際に XML インスタンスに表現されているシーンは三次元空間の視覚的情報を含み、文字で書いてあるだけではその空間を直観的に把握しにくい。よって、シーンの視覚的情報がユーザーに直観的に把握できるように、三次元コンピュータグラフィックスを用いてそのシーンの三次元空間を構築することが必要となる。

今日、三次元コンピュータグラフィックスを利用するための API は多種多様である。また、そのような API を用いて実装されているアプリケーションも非常に多い。このような既存のアプリケーションのファイルフォーマットに変換するコンバータを用意する事も有用である。その一つとして、我々はすでに VRML への変換を試みている。しかしながら、既存のアプリケーションを利用するだけでは、STOIC に記述されている意味・内容はコンバータを作成する際に用いられることはあってもそのアプリケーションで利用することはできない。そのため、STOIC に記述されている情報を利用するためのアプリケーションの枠組と STOIC 用 API の構築を必要とする。本稿では以上のような事を前提としつつ、三次元コンピュータグラフィックス API を用いて STOIC の視覚化 API の構築と STOIC Viewer の設計を行う。今回、三次元コンピュータグラフィックス API として Java3D を用いた。

XML と Java は Web 指向、オープン性、ベンダ中立という同じ立場を取っていることにより、最近、

両技術を利用した設計・開発が多く見られる。Sun microsystems は XML と Java は補間技術であるとし、JAXP,JAXB,JAXM といった XML 用 API を多々提供している。

本稿では、このような背景から XML を用いて設計された STOIC 用のアプリケーション作成、API の構築を行なうに当たって Java 言語を選び、Java 上での三次元コンピュータグラフィックス API として Java3D を利用する。OpenGL の Java バインディングとして GL4Java[6], JSparrow[7], Magician 等があるが、今後、統一的なものとなるかに疑問があるため利用を控える。

4.2 Java 言語を用いた STOIC アプリケーションの構築

本節では Java 言語を用いた STOIC アプリケーションの構築について述べる。STOIC アプリケーションのシステム構成を図 4に示す。このシステムを実現するための要素技術について順に説明する。

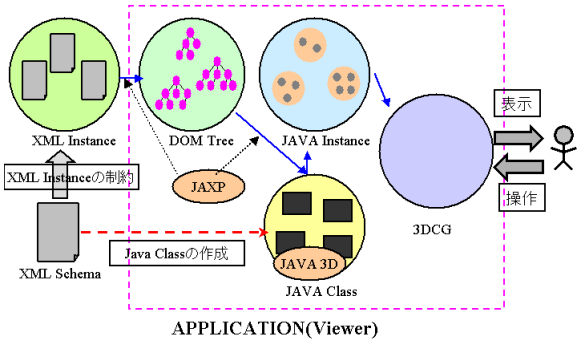

図 4: システム構成図

4.2.1 XML スキーマと Java クラス

STOIC を設計した XML スキーマを基に、その構造に沿って Java クラスを生成する。STOIC アプリケーションの設計はこの Java クラスを用いて行なわれる。

Java クラスの作成手順は以下のようになる。

1. 作成した XML スキーマを基に必要な Java クラスを想定し、そのクラスのプロパティを定義する。
2. XML スキーマで定義されるプロパティに accessor メソッド、mutator メソッドを与え、Java インターフェースを生成する。
3. インターフェース用のメンバ変数を定義するインプリメンテーション・クラスと、 インターフェース内の各メソッドのためのコードを生成する。

主に階層構造表現における親子はクラスのメンバとして定義する。また、STOIC の階層構造の leaf には数値や文字列だけでなく、グラフィックオブジェクトが配置される場合がある。グラフィックオブジェクトが定義されている場合は、Java3D クラスのインスタンスを親のメンバとする。

4.2.2 JAXP と DOM API

XML をプログラムで処理する場合、DOM や SAX といった XML プログラミングの標準モデルが定義されている。また、Sun microsystems が提供する JAXP を用いることによって DOM や SAX をベンダー非依存的な方法で使おうとするときに障害となるベンダー固有の作業が可能となる。

今回のアプリケーション作成では JAXP を用いて DOM Tree を構築し、Tree を隅無く巡回し、Node に対する操作を記述していく。Node に対する操作は主に前項で述べた Java クラスのインスタンスを生成することである。

4.2.3 Java3D とシーングラフ

Java3D はシーングラフをベースにした三次元グラフィック API であり、アプリケーションプログラミングを行なうには、シーングラフと呼ばれるオブジェクト・モデルを構築していくことになる。主に、以下のようなノード (DOM Tree の Node とは別のもの) の追加を行ないシーングラフを構築していく。

- TransformGroupNode

  移動、回転、拡大縮小など座標変換操作を適用する

- Shape3DNode

  物体の形状や頂点座標、色などの属性を管理する

- BehaviorNode

  物体の振る舞いを制御する

また、ViewPlatformNode により視点の移動や回転を制御できる。

4.3 STOIC Viewer

前節の構築システムに基づき、STOIC Viewer の実装を行なった。今回の Viewer では、STOIC インスタンスにおける物体の形状や構造のデータを使用し、シーングラフを構築し三次元グラフィックとして描画している。

5 まとめ

我々が住んでいる 3 次元空間と、その空間中の「もの」とその「振舞い」によって表現される実世界の情報は非常に様々な「意味」を持っていて、本稿では、そのような実世界の情報を計算機間でやりとりすることを目的としてこれを実現するためにSTOICを提案した。STOIC は情報キャリアなので、人間にも分かりやすいように、構造化された文書で記述することを考え、構造化文書を記述する一手法である XML を用いて STOIC を表現した。また、我々が提案・定義した STOIC の意味表現能力に対する評価を行ない、最後にその有用性を示すために、実際に CG を用いた表示システムを実装した。

参考文献

[1] H.Arisawa,H.Nagae,Y.Mochizuki:“Representation of Complex Objects in Semantic Data Model“AIS” and Implementation of Set Operators”,IEICE TRANSACTIONS,vol.E74,No.1,1991.

[2] H.Arisawa,T.Tomii,H.Yui,and H.Ishikawa: “Data Model and Architecture of Multimedia Database for Engineering Applications”, IEICE TRANS.INF. & SYST.,vol E78-D,No.11,1995.

[3] 松田一章, 越前谷健二, 富井尚志, 有澤 博: “XML を用いたシーンの時空間オブジェクト情報の記述” , 電子情報通信学会データ工学ワークショップ,2001

[4] XML 1.0(Second Edition):

[5] XML Schema:

Part0:Primer

Part1:Structures

Part2:Datatypes

[6] GL4Java: “http://www.jausoft.com/gl4java/”

[7] JSparrow: “http://www.pfu.co.jp/jsparrow/index.html/”