2007 年度 修士論文

空間情報処理プラットフォームによる
コミュニティ主導型開発の研究

Community Based Development on the Platform for Spatial Information Processing

上山 智士

Ueyama, Satoshi

東京大学大学院新領域創成科学研究科

社 会 文 化 環 境 学 専 攻

目次

第1章

序 論

近年、情報処理技術の発展に伴う情報爆発が起こり、大量の情報を整理するための手法が模索されている。空間情報を用いることは、大量の情報を管理するための有効な手段である。我々は、ある空間についての情報を必要とすることが多いからである。しかし、空間情報による情報管理を実用化するためには多くの人が空間情報を容易に処理できる環境が不可欠である。本章では、このような空間情報処理を取り巻く背景を解説し、本研究の目的を定義する。

1.1. 背景—空間情報処理の重要性

近年、情報処理技術の発展に伴い、膨大な量の情報が世界に溢れ、日々増加している。このような情報量の急速な増加は、情報爆発と呼ばれている[KITSUREGAWA2003]。情報爆発の中で情報を効率的に管理するための有効な切り口として、空間情報がある。空間情報とは、ある空間に関連する情報である。我々が情報を探すとき、多くの場合、空間という制約を設けることが多い。昼食を食べに行こうと思った場合、現実的に行くことができる店の情報のみが有用な情報であり、それ以外はノイズである。

現在、計算機に蓄積されている情報は、日付や名前、大きさなどを軸に管理されている。利用者は、情報を選別するために、日付や名前、大きさをキーに検索を行うことが一般的である。一方で、空間情報を軸とした情報管理は、まだ黎明期にあると言えよう。利用者が横浜に関する情報を検索する場合、現状では「横浜」という単語について検索を行っているに過ぎず、検索語や検索対象を地理的な意味のある語、つまり空間情報として扱ってはいない。空間情報を有効に活用できれば、情報の検索性は大いに向上し、我々は膨大な量の情報をより容易に扱えるだろう。

空間情報の単純な例としては、地図や航空写真が挙げられるが、空間情報は、地図や航空写真以外にも普遍的に存在するものである。卑近な例として、新聞記事を挙げよう。新聞記事には多くの場合、事象が起きた場所が含まれている。横浜で起きた交通事故に関する記事であれば、それは横浜という位置に関する空間情報であるといえる。個人の雑記や写真、あるいは歌曲も同様に空間情報である可能性がある。

空間情報を有効活用するためには、あらゆる人が容易に空間情報の整備を行える環境が必要である。単純なキーワードによる検索のためのデータであれば、単語や文章をテキストデータとして入力するだけで構築でき、実に簡便である。近年多くのサービスで導入されているタギング機能は、この簡便さの上に成立していると言えよう[ADAM2004]。空間情報も、より簡便にデータを構築できる環境が整備されれば、自ずと空間情報の整備が進むであろう。

1.2. 本研究の目的

本研究の最終的な目的は、空間情報処理の裾野を広げ、空間情報が活用される場

面を増やすことである。そこで、本論文ではまず、空間情報処理の現状を整理し、問題点を整理した。空間情報処理の現状が抱える問題点は、空間情報処理があくまで専門家が行う作業であり、一般の利用者との距離が遠いことである。空間情報処理に特化したソフトウェアである GIS は、利用者に多大なコストをかけて学習することを求めている。近年開発が進んでいる WebGIS は、利用するだけであれば従来の GIS に比べて容易であるが、データの構築や運用には高度な知識が必要である。

本研究では、一般の利用者でも容易に空間情報処理を行える環境の整備を目指した。この空間情報処理とは、単に既存の空間情報を利用するだけではなく、自ら空間情報を構築して発信することや、既存の空間情報を再利用し新たな情報やアプリケーションを作成することも含む。このように、利用者自らが作製者側に回ることで情報が情報を生み出すエコシステムは、まさに World Wide Web の発展に見られるものである。そこで、本研究ではまず、World Wide Web の成功の要因を分析し、空間情報処理の環境整備において必要な要件を整理した。続いて、それらの要件を満たす空間情報処理プラットフォームを構築し、その実装も作成した。最後に、プラットフォームの実装を利用してコンテンツの作成や再利用を行い、プラットフォームの有用性を確認した。

第2章

空間情報処理の現状と問題点

空間情報を処理するためのシステムは GIS と呼ばれ実用化されているが、GIS は専門性の高いツールであり、ごく限られた数の利用者が研究や業務に使用しているものである。本研究の最終的な目的は、空間情報処理の裾野を広げ、空間情報が活用される場面を増やすことである。そこで、本章では、GIS が抱える問題点を指摘し、それを解決するために、World Wide Web の成功を参考にしたアプローチを提示する。

2.1. 古典的な GIS

空間情報を処理するためには GIS と呼ばれるソフトウェアを利用することが一般的である[YAN2003]。本項で述べる GIS ソフトウェアは、利用者が自身の計算機に組み込んで利用する、デスクトップアプリケーションと呼ばれる形態のものである。GIS ソフトウェアは既に、一般市場向けの廉価な製品から、研究者や業務システム向けの高機能な製品まで多くの製品が存在する。図 1は、代表的な GIS ソフトウェアである ArcGIS の画面写真である。GIS ソフトウェアの利用者の目的は、収集したデータの解析、データベースとしての利用、視覚化など様々であり、GIS ソフトウェアはこれらの用途に必要な機能を統合している。GIS ソフトウェアは、様々な機能を統合することで熟練した利用者の要求に応える一方、専門的で一般の者には難解なツールとなっている。

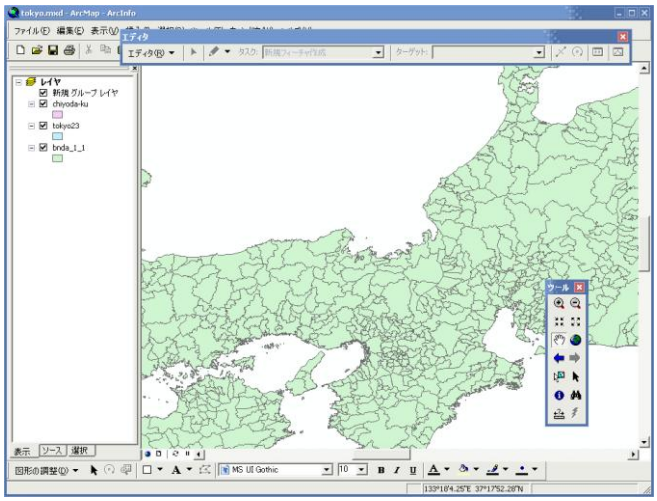

図 1 ArcGIS

2.2. WebGIS

他の分野のソフトウェアと同様に、Web への統合は GIS にとって重要なテーマである。WebGIS の代表的な例は、米 Google Inc.が提供するサービス「Google Maps」であろう。Google Maps は、一般利用者向けの地図閲覧サービスが出自であるが、利用者が独自の情報をオーバーレイする機能などを実装することにより、徐々に統合型 GIS としての側面を強めている。衛星写真やアドレスマッチング、オーバーレイ機能などの GIS 技術を大衆化させた Google Maps の功績は大きい。

しかし、Google Maps は、比較的高負荷の処理をスクリプティングで実装しているため、パフォーマンスの問題を抱えている。

Web 上で 3 次元 GIS を運用する試みもある。最近の研究では、Mark らによる研究および後藤らによる研究において、Web ベースの 3 次元 GIS を実装する試みが行われている[GOTO2004][MARK2003]。Mark らによる研究では、サーバからストリーミングされるデータを Java アプレットでレンダリングするシステムを実装している。携帯電話上の Java アプリケーションで 3 次元 GIS のレンダリングを行う試みもある[HAYASHIHARA2003]。米 Google Inc.が公開している「Google Earth」は、レンダリングのためのプログラムを計算機に組み込んで使用するため完全な WebGIS ではないが､データを利用者側の計算機に組み込まず、ネットワークからストリーミングしている。

WebGIS の利点は 2 つある。まず、ソフトウェアを利用者自身の計算機に組み込む必要がないことである。この利点は、GIS 以外の分野のソフトウェアが積極的に Web ベースへ移行している理由でもある。次に、利用者がデータの更新をする必要がないことである。サービスの運営者がホスト計算機上でデータを更新しさえすれば全利用者に新しいデータが行き渡るため、運営者が新しいデータを配布するコスト、および利用者がそれを受け取るコストを省くことができる。

一方で、問題もある。本項で述べた WebGIS は、サーバサイドに大掛かりなシステム必要とするアーキテクチャである。このようなアーキテクチャは、作成者側に参加するための障壁が高い。また、サーバが抱えているデータの流通、交換、加工などが進まない。Google Map は、この点についていくらかの改善をしたが、先に述べたとおり新たな問題も抱えている。Google Earth は、高い表現力を実現したものの、サーバ依存型のアーキテクチャを脱していない。

このように、WebGIS は、古典的な GIS と比べ、既存の情報を利用する目的には優れているが、データを作成するための障壁は高いままである。第 1 章で述べたように、本研究では、利用者が作成者にもなり、情報が情報を生むエコシステムを目指している。次節では、この目標を達成した例である World Wide Web に関する議論を行う。

2.3. World Wide Web のアーキテクチャ

World Wide Web は、世界で最も巨大なデータベースであると同時に、そのデー

タベースを利用するアプリケーションでもある。これほどまでに巨大なシステムが成功している理由を Roy は、その障壁の低さ（Low Entry-barrier）と拡張性（Extensibility）に見出している。障壁の低さについては、利用者、情報ソースの作者、およびアプリケーションの開発者それぞれの立場から分析している[ROY2004]。

まず、利用者からみた World Wide Web の特徴はユーザーインターフェイスである。ユーザーインターフェイスは、利用者にとってのアプリケーションの評価を決める大きな要素である。豊富なデータや機能も、粗悪なユーザーインターフェイスが原因で利用者の手元に届かなければ、それは利用者にとっては何の価値もない。Roy は、World Wide Web においてハイパーメディアがユーザーインターフェイスとして機能していることを指摘した。特に、単純なハイパーメディアの文書がリンクによって関連付けられている点を特筆している。複雑な情報を扱うためにユーザーインターフェイスも複雑化してしまう失敗は、ソフトウェア開発において良く見られる失敗であるが、World Wide Web では「ページ」という単位でごく一部の情報を扱い、それに関連する情報を別のページに分離し、リンクで利用者を案内する。デスクトップアプリケーションと World Wide Web のユーザーインターフェイスの比較を図 2に示す。

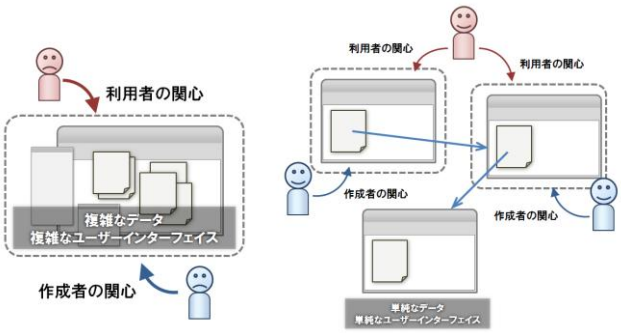

図 2 デスクトップアプリケーションと World Wide Web のユーザーインターフェイス

デスクトップアプリケーションは、複雑なデータを扱うために多機能化し、ユーザーインターフェイスも複雑化してしまう。利用者は複雑なユーザーインターフェイスに多くの学習コストをかけ、開発者も複雑化したシステムの保守に多大な労力をかける。一方 Web は、個々の開発者が単純な情報を Web ページという単純なユーザーインターフェイスと共に提供する。利用者は、一つ Web ページで一つの問

題を解決し、次のページへ移ることを繰り返せばよい。Roy が指摘した Web ページのユーザーインターフェイスの普遍性は、新たなページを利用するために必要な学習の量を低く抑えている。Roy は World Wide Web のコンテンツのフォーマットに対し「シンプルで既存の編集ツールで作成可能」であることを要求し、事実そのようになっている。 具体的にはどうだろうか。最も普及している商用 OS の Microsoft Windows には Notepad と呼ばれるエディタが付属している。Notepad は、エディタと呼ばれるソフトウェアの中では恐らく最も低機能なものの一つである。しかし、Notepad ですら小規模な コンテンツを作成する用途においては十分機能する。

拡張性とは何であろうか。GIS をはじめソフトウェアの多くはプラグイン機構により機能を拡張できる。これは拡張性の実現方法の一つであるが、World Wide Web の拡張性は、プラグインによる機能拡張とは異質なものである。World Wide Web の拡張性とは、ネットワーク上に分散した個々の要素が独立して交換、追加、削除可能ということである。World Wide Web の拡張性を実現している技術が URL とリンクである。URL は World Wide Web 上に存在する全ての情報に対して付される名前であり、リンクはその名前への参照である。この URL とリンクによる結合は、勝手な情報の改ざんや削除を許してしまう実に「ルーズな」結合である。一見欠点に見えるこのルーズさは、World Wide Web の無秩序な拡張を実現している。URL とリンクの機構は、単純な静的ページのみで World Wide Web が構成されていた時代には自然に機能していた。しかし、近年、動的生成のページやリッチなコンテンツの増加に伴い、URL とリンクが機能しないケースが目立ち、改めてその重要性が認識されている状況である。例えば、Google Maps のような DHTML を利用した地図の場合、ユーザが操作した後の状態を共有するためには、システムがその状態を表す URL を生成して利用者に提示する必要がある。開発者の技量不足などの問題でこのような機能が実装されていない場合は、操作後のページは URL を持たない「幻」になってしまう。また、動画を配信するサイトの場合、動画全体へのリンクは簡単であるが、動画の一部へのリンクを行いたい場合は、システムが動画の一部を表現する URL を生成する必要がある。このようなシステムの例としては、山本らが実験を行っている Synvie がある[YAMAMOTO2006]。

2.4. バザールモデル

Roy が指摘した World Wide Web の成功の理由は、バザールを促す仕掛けであったと言える。バザールとは、Eric が Linux の開発モデルを基に定義したソフトウェアの開発モデルである。Eric は、バザールの下では、大勢の開発者が、一見無秩序に開発を行っても、整合性の取れた信頼に足る巨大なシステムが完成することを示した。コンテンツ作成における「障壁の低さ」は、多くの開発者を取り込むことに寄与し、拡張性は、その多くの開発者が気兼ねなくリリースを行うために必要だった。ただし、Eric はバザールモデルによる開発のスタートアップについて条件をつけている。即ち、ごく初期にはバザールモデルをとらずにプロトタイプを完成させていることである。プロトタイプは、どんな書類よりも参加者にプロジェクトの目的をアピールする。また、貢献の結果が目に見えることが参加の動機になる。

第3章

空間情報処理プラットフォームの設計

前章では、現在の GIS が抱える問題点を指摘し、World Wide Web の成功を参考にして問題解決を図ることを提示した。本研究では、その具体例として、専門家でなくとも容易に理解できるシンプルなデータモデルを定義し、それを CSS、DOM、および JavaScript といった既存の技術で処理するというプラットフォームを構築した。本章では、このプラットフォームを構成するデータモデルおよびプログラミングモデルについて、設計の概観を述べた後、詳細な仕様を解説する。

3.1. 設計方針

本研究のプラットフォームは、World Wide Web のアーキテクチャに倣い設計されたデータモデルおよびプログラミングモデルで構成されている。本節では、その設計方針について解説する。

本研究のプラットフォームは、以下の 3 点を実現することを目標とした。

1. URL とリンクの保証
2. 低い障壁
3. 拡張性

URL とリンクの保証は、本研究のプラットフォームで記述されたコンテンツ同士、あるいはその他のコンテンツとの相互運用を実現するための要素である。URL とリンクの保証により、本研究のプラットフォームは、World Wide Web を拡張する部品として機能する。

低い障壁は、利用者がコンテンツの作成に参加することを容易にするための要素である。低い障壁の実現のために、本研究のプラットフォームでは、既に多くの開発者が親しんでいる技術を可能な限り再利用すうことにより、新しい技術の学習コストを減らすことを狙っている。また、新たに独自の仕様を決定する場合は、多機能であることよりも単純で理解しやすいことを優先している。

拡張性は、利用者が各自の目的に合わせて独自のデータやプログラムを組み込むことを可能にするための要素である。つまりこの拡張性とは、予期せぬ要素を取り込むことができるという寛容性である。

3.2. データモデル

本節では、本研究のプラットフォームで使用するデータモデルについて解説する。本節で開設するデータモデルは、ある地点や領域といった、単純な空間情報の組み合わせである。本節では、まずデータモデルの概観を示し、次に構成要素の解説をした後、最後に XML によるコーディング方法を解説する。

3.2.1. 概観

図 3に、本研究で定義したデータモデルの概観を示す。

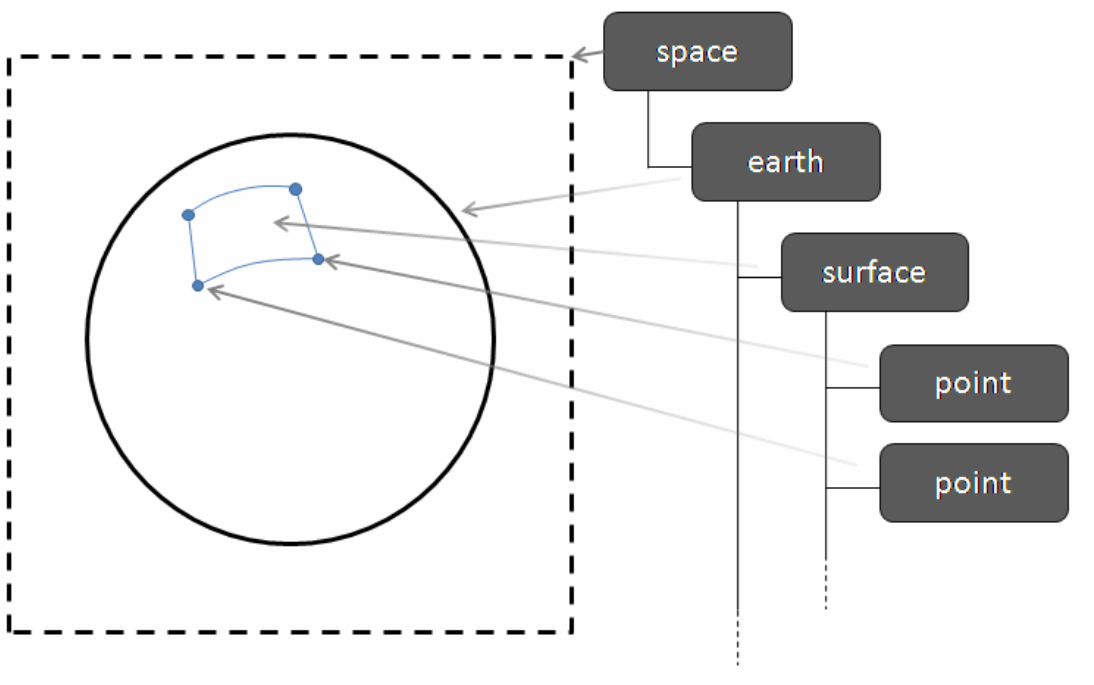

図 3 データモデルの概観

図 3の右部分は、本研究のデータモデルの各要素がツリーを構成している様子を表している。図 3の左部分は、右部分のツリーが表すデータを視覚化したものである。左部分についてもう少し補足する。図 3の左部分は、ある空間(space)の中に地球(earth)があり、地球の上に面(surface)や点(point)といった要素が存在している様子を表している。

"space"は、他の全ての要素のコンテナとなる要素である。現実の世界で全ての要素が「空間」の中にあるように、本研究のデータモデルの各要素も、space 要素の中になければならない。

earth 要素は、地球を表している。surface 要素は、地球上のある領域を表している。point は surface 内部のある点を表しており、上位の surface は、子要素の point 全てを包含するような領域となる。

本研究のデータモデルの構成要素を表 1にまとめた。

表 1 本研究のデータモデルの構成要素

要素名	説明
space	空間全体を表す要素であり、全ての要素のコンテナ
earth	地球全体を表す要素
surface	領域を表す要素
point	地点を表す要素

3.2.2. 記述方法

本研究のデータモデルを多くの人々に利用してもらうためには、データをどのように記述するかは重要な問題である。データの記述に際して、バイナリとして記述するかテキストとして記述するかという選択がある。本研究の目的のためには、ここではテキストを選択すべきである。なぜテキストデータが優れているか、バイナリとテキストの比較で議論しよう。同じデータをバイナリとテキストで記述した場合を図 4に示す。

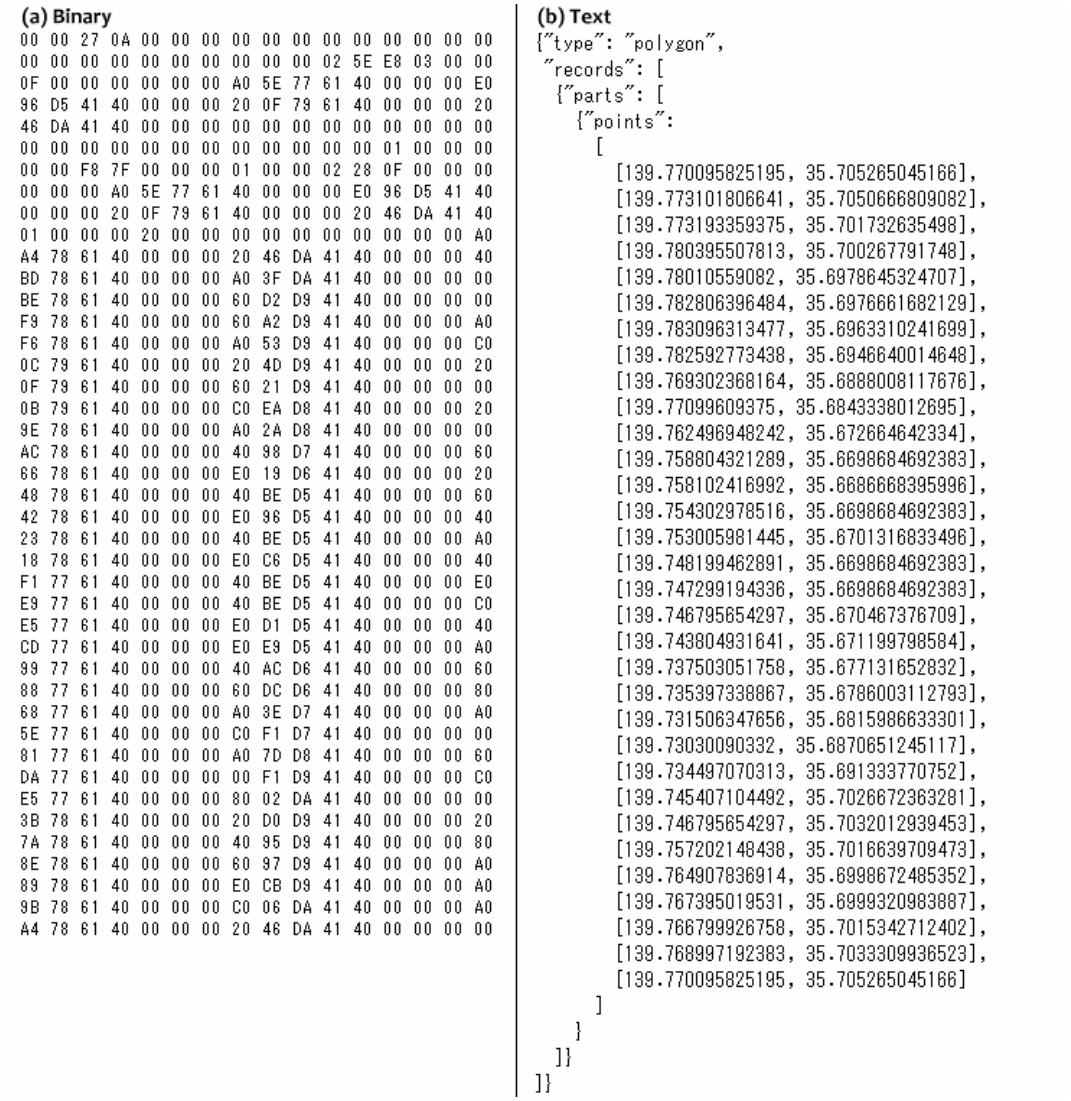

図 4 バイナリ記述とテキスト記述

図 4の(a)および(b)は、どちらも同じデータを記述している。(a)は、GIS 業界で事実上標準として利用されている Shape と呼ばれる形式である。(b)は、Web アプリケーションのデータ永続化や通信のために使用されている、JSON と呼ばれる形式である。Roy の言う「低い障壁」の観点から見れば、明らかに(b)が望ましい。(a)のようなバイナリ記述は、仕様書無しでは読み取ることが不可能である。一方(b) のようなテキストでは、データそのものが説明的である。例えば、"type": "polygon" は、このデータの種類がポリゴンであることを連想させる。優れたテキスト記述は、直感的な理解が可能である。ただし、当然ながらその陰には厳密な仕様が存在する。このような特徴により、テキスト記述は、「低い障壁」を実現

することができる。なぜなら、利用者は、入手したデータをテキストエディタで開くだけで、不完全ではあるが仕様書を入手し、開発環境を手に入れたことになるからである。以上のような理由により、本研究では、データモデルの記述を XML で行った。勿論、理解しやすい XML となるためには、データモデルそのものが明解であることが求められる。

幾何的な情報を XML で記述する点は、SVG に近い。ただし、本研究のデータモデルは、地理的な情報を直接記述でき、レンダラが地理的な情報を生かしたレンダリングを行う点が SVG との決定的な違いである。

ここで、本研究のデータモデルを XML で記述した具体的な例を示す。リスト 1 は、最も単純な UE データモデルの記述例である。

```
<ue:space id="space1">
    <ue:earth id="earth1">
            <ue:point lat="35.676880" lng="139.723450" />
            <ue:surface id="tokyo-area">
                    <ue:point lat="35.8" lng="139.6" class="rect-NW" />
                    <ue:point lat="35.8" lng="139.9" class="rect-NE" />
                    <ue:point lat="35.6" lng="139.9" class="rect-SE" />
                    <ue:point lat="35.6" lng="139.6" class="rect-SW" />
            </ue:surface>
    </ue:earth>
</ue:space>
```

リスト 1

リスト 1は、earth 要素直下、すなわち地球上のある地点とある面を一つずつ記述している。面を構成するために 4 つの点を追加しているので、各要素の子孫をすべて含めれば 5 個の点を記述している。各要素は、その属性に追加的な情報を持つことができる。point 要素は多くの場合、緯度と経度をそれぞれ lat 属性および lng 属性に保持する。id 属性や class 属性は多くの場合、アプリケーションの処理や意味的な情報の追加を目的として追加される。

実際に UE データモデルを記述する場合、XHTML 文書の内部に複合文書として記述することを推奨する。この方法には、XHTML の script 要素によるスクリプトの記述や style 要素によるスタイルシートの記述が可能になるという長所が

ある。リスト 1を XHTML の内部に記述した場合の例をリスト 2に示す。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html
xmlns="http://www.w3.org/1999/xhtml"
xmlns:ue="http://www.csis.u-tokyo.ac.jp/UE/"
xml:lang="ja" lang="ja">
<head>
<title>sample</title>
</head>

<body>
<div>
<ue:space id="space1">
    <ue:earth id="earth1">
        <ue:point lat="35.676880" lng="139.723450" />
        <ue:surface id="tokyo-area">
            <ue:point lat="35.8" lng="139.6" class="rect-NW" />
            <ue:point lat="35.8" lng="139.9" class="rect-NE" />
            <ue:point lat="35.6" lng="139.9" class="rect-SE" />
            <ue:point lat="35.6" lng="139.6" class="rect-SW" />
        </ue:surface>
    </ue:earth>
</ue:space>
</div>
</body>
</html>
```

リスト 2

リスト 2では、XHTML をデフォルト名前空間として、プレフィックス ue:を UE データモデルの要素の名前空間に割り当てている。リスト 2は、リスト 1の各要素にプレフィックス ue:を付加した上で内包している。

以上が本研究で定義したデータモデルの概要である。ここで、前章で述べた、作成者にとっての「低い障壁」が実現されているかどうか議論しよう。作成者が直面する第一の障壁は、新規文書の作成である。特定のソフトウェアに依存したデータ形式の場合、そのソフトウェアを入手するか、互換性のあるツールを入手

して書き出し処理を行う必要がある。本研究のデータモデルを用いて新たな文書を作成する場合は、空の XHTML 文書に space 要素および earth 要素を加えたものを雛形として情報を追記すればよい。これはテキストエディタで十分行える程度の作業である。雛形の具体例をリスト$list_template に示す。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html
xmlns="http://www.w3.org/1999/xhtml"
xmlns:ue="http://www.csis.u-tokyo.ac.jp/UE/"
xml:lang="ja" lang="ja">
<head><title></title></head>
<body>
<div>
<ue:space>
    <ue:earth></ue:earth>
</ue:space>
</div>
</body>
</html>
```

リスト 3

作成者は、リスト 3のような雛形を何らかの方法で用意すれば文書の作成を開始できる。

3.2.3. データモデルの拡張

前項で解説したように、本研究のデータモデルは XML で記述される。XML の処理系は認識できない要素を読み込んだ場合は異常終了せずに、単に無視して処理を継続する。よって、本研究のプラットフォームで定義していないデータを XML 中に記述することが可能である。記述するデータの例としては、利用者が独自に作成したアプリケーションのための情報や、microformats によるメタデータが考えられる。

3.2.4. 要素の URL

XML の要素は id 属性の指定により URL を持つことができる。URL"http://example.com/"で識別される文書中の要素の id 属性が"summary"であれば、その要素は"http://example.com/#summary"として識別される。本研究のデータモデルの各要素も同様に URL を持ち、他のドキュメントからリンク可能であることを保証している。

3.2.5. プレゼンテーション

UE データモデルは、XHTML 文書あるいはその他の XML 文書と同様に、スタイルシートを用いて視覚表現を記述することができる。本論文では、スタイルシートを Cascading Style Sheets(CSS)を用いて記述した。本研究で利用する CSS は、CSS Level 2 Revision 1(CSS2.1)の仕様を基本とし、CSS Level 3(CSS3)の仕様を一部取り入れている[W3C2007][W3C2006]。CSS2.1 および CSS3 は、どちらも正式に勧告されていない仕様であり、特に CSS3 は今後大きく変更される可能性がある。さらに、一部に独自の仕様を追加している。

CSS の記述方法は、HTML のそれとまったく同様である。リスト 4にもっとも単純な CSS の記述例を示す。

```
#space-1 {
    background: #fff;
}
```

リスト 4

リスト 4は、space-1 なる id に背景色を指定している。続いてセレクタに要素名を使う場合の記述例をリスト 5に示す。

```
@namespace ue url(http://www.csis.u-tokyo.ac.jp/UE/);
ue|point
{
    color: #f00;
}
```

リスト 5

リスト 5は、全ての point 要素の色を指定している。セレクタに要素名を使う場合は、@namespace 規則により名前空間を宣言し、接頭辞と共に要素名を指定する。接頭辞を使用することにより、本研究のデータモデルの要素が正確に選択される。

UE データモデルは、3 次元グラフィックスを用いて視覚化されることを想定している。そのため、スタイルシートに 3 次元グラフィックスを用いた視覚化に必要なプロパティをいくつか追加した。

(1) テセレーション関連のプロパティ

tessellation-type プロパティおよび tessellation-resolution プロパティは、面をレンダリングする際に用いられるテセレーションのパラメータを指定するためのプロパティである。このテセレーションとは、分割された平面で曲面を疑似的に表現する処理である。図 5にテセレーションの例を示す。

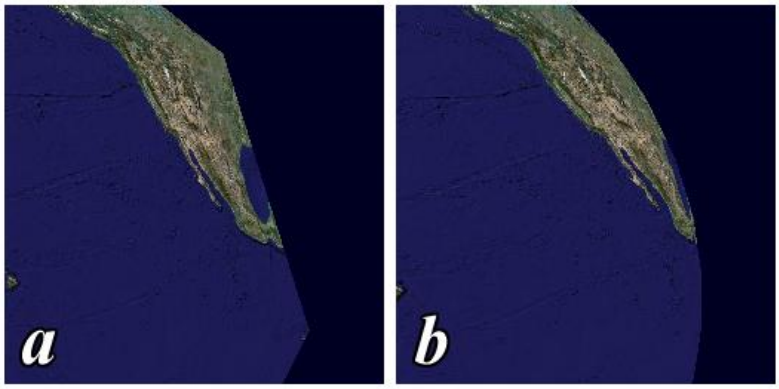

図 5 テセレーション

図 5の(a)は、テセレーションを行っていないため、本来曲面であるべき面が平面として見えている。図 5の(b)は、テセレーションを行って曲面を疑似的に表現している。tessellation-resolution プロパティは、テセレーション処理で曲面を分割する際の分割数を指定する。tessellation-type は、曲面を分割する方法を指定する。

(2) テクスチャ座標関連のプロパティ

tex-u プロパティおよび tex-v プロパティは、surface 要素に画像を割り当てる場合に画像のテクスチャ座標を指定するためのプロパティである。tex-u プロパティおよび tex-v プロパティを surface を構成する point 要素に指定することにより、領域の各頂点のテクスチャ座標が決定される。tex-u プロパティは U 座標、tex-v プロパティは V 座標として解釈される。テクスチャ座標と画素の関係を図 6に示す。

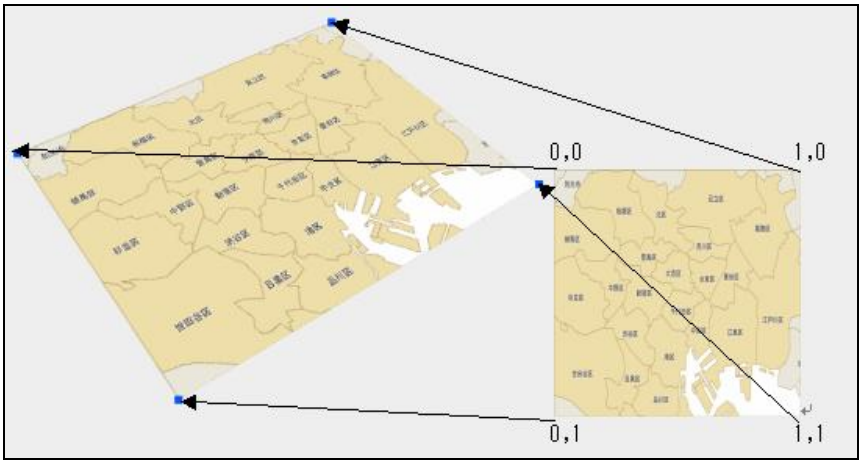

図 6 テクスチャ座標と画素

図 6に示すように、各頂点に tex-u プロパティおよび tex-v プロパティで指定したテクセルが割り当てられる。尚、surface 要素に割り当てる画像の指定するためのプロパティは、標準仕様に含まれている background-image プロパティである。tex-u プロパティ、tex- v プロパティおよび background-image プロパティを使用し、surface に航空写真を張り付けた例を図 7に示す。

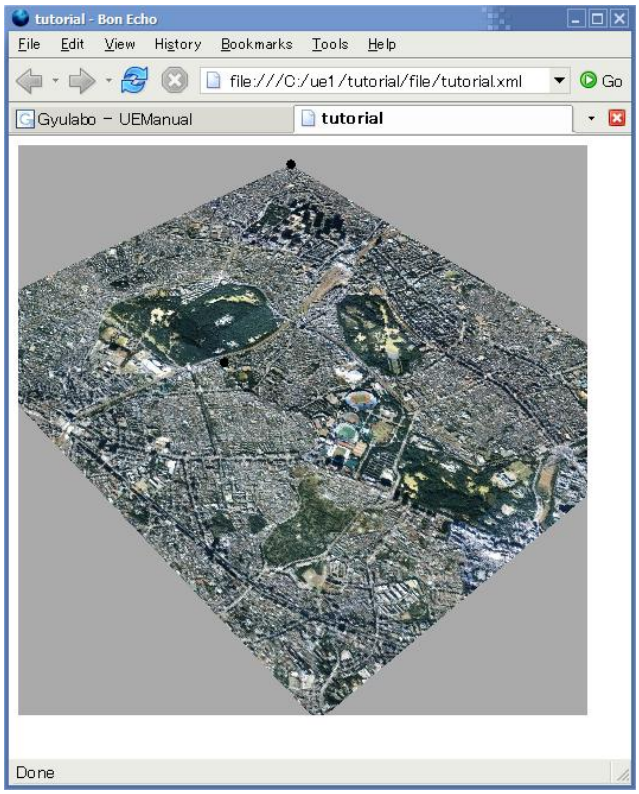

図 7 航空写真の貼り付けの例

(3) 位置関連のプロパティ

west プロパティおよび north プロパティは、要素が視覚化される位置を指定するためのプロパティである。west プロパティは要素の経度、 north プロパティは要素の緯度をそれぞれ指定する。リスト 6に west プロパティおよび north プロパティの記述例を示す。

```
north: attr(lat, deg);
west: attr(lng, deg);
```

リスト 6

リスト 6では、値を直接指定せずに、要素の属性から取得するよう指示している。前項で解説した point 要素の lat 属性および lng 属性をそれぞれ緯度、経度として正しく解釈するためには、リスト 6の指定が必須である。

(4) 座標変換関連のプロパティ

transform-3d プロパティは、視覚化された要素に対して座標変換を指定するためのプロパティである。transform-3d プロパティの値は、m44(m11, m12...)なる関数的記法で指定される。座標変換の指定の概念を図 8に示す。

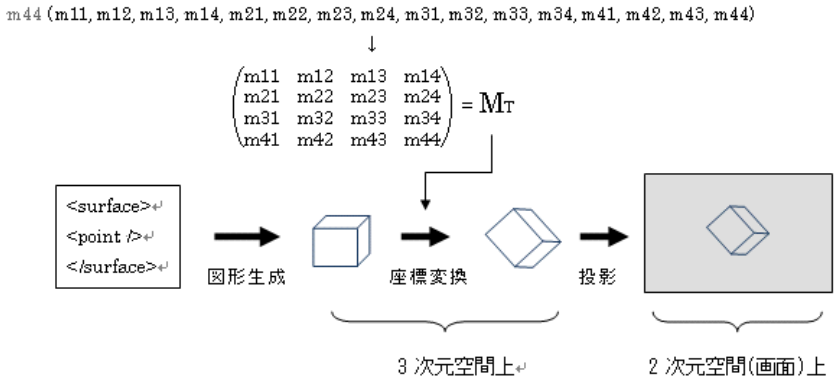

図 8 座標変換の指定

m44()内のパラメータは、第 1 行第 1 列、第 1 行第 2 列…の順に解釈され、4 行 4 列の行列を成す。レンダラは、画面への投影を行う直前に、この行列を適用して座標変換を行う。Y 軸を中心に 45 度の回転を行う変換は、リスト 7のようにコーディングされる。

```
transform-3d:
m44(0.7071067811865476,0,-0.7071067811865475,0,0,1,0,0,0.7071067811865475,0,0.7071067811865476,0,0,0,0,1);
```

リスト 7

transform-3d プロパティは、space 要素にも適用することができる。この場合、space 以下の空間全体に座標変換を行う結果となる。空間全体への座標変換は、視点の表現に適している。例えば、z 軸方向に 10m 移動する変換を space 要素に適用した場合は、視点が z 軸方向に- 10m 移動したようにレンダリングされる。また、space 要素は、特別な projection-matrix プロパティを持つ。projection-matrix プロパティに指定された座標変換は、画面上への投影に用いられる。

(5) スィープ関連のプロパティ

sweep プロパティは、surface の押し出しを指定するプロパティである。レンダラは、surface を sweep プロパティに指定された長さ分押し出してレンダリングする。sweep プロパティを指定していない状態の surface を図 9に、sweep プロパティを指定した状態の surface を図 10に示す。

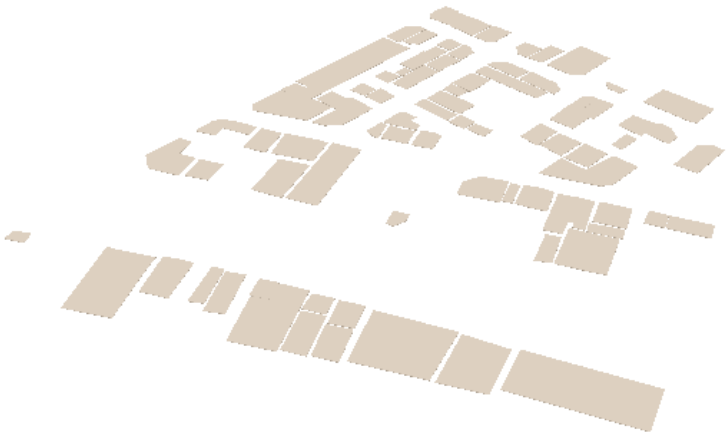

図 9 プロパティを指定していない状態の surface

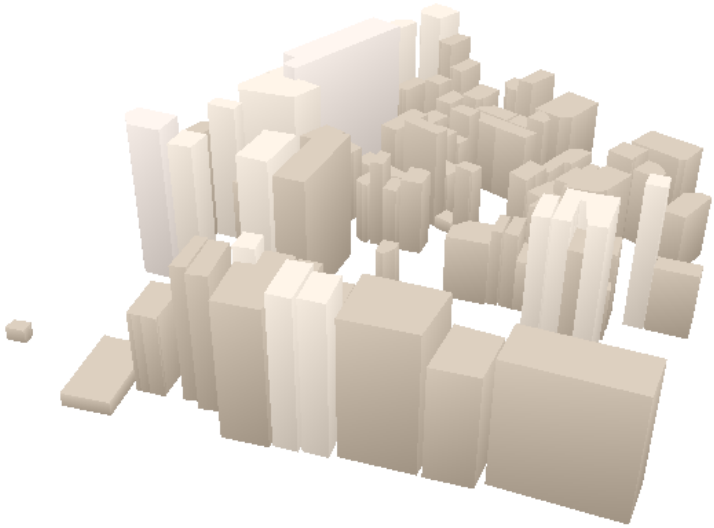

図 10 プロパティを指定した状態の surface

図 9と図 10は、同じデータを視覚化したものであるが、sweep プロパティの指定の有無により、レンダリング結果が大きく異なっていることがわかる。

(6) リスト関連のプロパティ

リスト関連のプロパティにもいくつか拡張がある。まず、list-style-type に指定可能な値を新たに定義した。list-style-type の値の拡張を表$list_type_val に示す。

表 2 list-style-type の値の拡張

プロパティ名	説明
rect-pillar	要素を直方体としてレンダリングする。

また、新しいプロパティをいくつか定義している。拡張プロパティを表$list_props に示す。

表 3 リストの拡張プロパティ

プロパティ名	説明
list-style-size	要素の視覚表現の大きさ(立体の場合は幅)
list-style-height	要素の視覚表現の高さ

拡張プロパティを利用した場合のレンダリング例を図 11に示す。

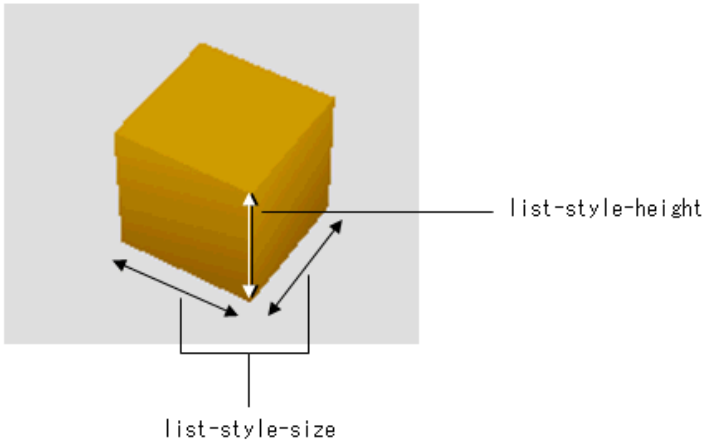

図 11 リストの拡張プロパティの利用例

図 11の例では、point 要素に対してリスト 8のようなスタイルを指定している。

```
color: #c90;
list-style-type: rect-pillar;
list-style-size: 90m;
list-style-height: 90m;
```

リスト 8

list-style-type: rect-pillar; の指定により直方体が生成され、その寸法に list-style-size プロパティおよび list-style-height プロパティの値が用いられている。また、color プロパティの値が直方体全体の色として用いられている。

3.3. プログラミングモデル

本研究のデータモデルおよびスタイルシートは、動的な操作を可能にするための API を備えている。スクリプトから API を利用することにより、開発者は、対話的なアプリケーションを容易に開発できる。本研究のデータモデルおよびスタイルシートが備えている API は、DOM API と全く同一である。DOM API プログラミングの経験があるプログラマーは、わずかな知識の拡張で、本研究の API によるプログラミングも行えるだろう。本研究のデータモデルおよびスタイルシートに対して動的な操作を行うことにより、アニメーションを高い自由度で行うことができ

る。アニメーションの典型的な利用例としては、時系列データの視覚化が挙げられる。さらに、利用者からの入力に対応して処理を行うことにより、対話的なアプリケーションを作成することができる。この対話的なアプリケーションの存在により、本研究のプラットフォームは、データを閲覧するだけでなく編集するための環境ともなる。

3.3.1. プログラミング言語

XML 文書やスタイルシートを動的に操作するためには、API を呼び出す処理を記述するためのプログラミング言語が必要である。このような目的のプログラムは、Javascript により記述され、Web ブラウザに組み込まれている Javascript 処理系で実行されることが多い。本論文でもプログラムを Javascript で記述している。無論、処理系がサポートしていれば他の言語による記述も可能である。

3.3.2. DOM API

DOM API は、XML 文書を構成する要素のツリーを操作するための API である。リスト 9に、DOM API を利用した基本的な操作の例として、要素の生成、挿入および削除を行うためのコード記述例を示す。

```
p = document.createElementNS("http://www.csis.u-tokyo.ac.jp/UE/", "point");
parent.appendChild(p);
```

リスト 9

リスト 9は、point 要素を生成し、parent の子要素として追加している。parent は、何らかの DOM ノードである。リスト 9において、名前空間や要素名は本研究のデータモデルに特有なものであるが、createElementNS や appendChild といった API 自体は標準の DOM API そのものである。

DOM API は、文書中の要素だけでなく、要素に結び付けられているスタイルシートの操作も可能である。リスト 10にスタイルシート操作の例を示す。

```
element.style.color = "#f00";
```

リスト 10

element が color プロパティの影響を受ける要素であれば、リスト 10を評価することによって element の視覚表現が変化するだろう。

3.3.3. DOM Events

DOM Events は、XML 文書に対して発生したイベントを利用してプログラミングを行うためのイベントモデルである。本研究のデータモデルの各要素は、DOM Events に定義されているイベントモデルに従う。DOM API と同様、イベントを利用したプログラムも標準の仕様に沿って記述可能である。例として、マウスボタン押下をハンドリングする例をリスト$ event_mouse に示す。

```
spaceObj.addEventListener("mousedown", onMouseDown, false);
```

リスト 11

リスト 11は、spaceObj が UE データモデルの space 要素であることを想定している。space 要素の視覚表現上でマウスボタンが押下された場合、onMouseDown という名前のルーチンが呼び出される。

3.4. 実装

本研究では、前節で解説したプラットフォームの実装を作成した。本研究では、Mozilla Firefox と呼ばれるウェブブラウザを改変する形態でプラットフォームを実装した。ただし、Mozilla Firefox という名称は、Mozilla Corporation 以外の第三者によるビルドには使用できず、代わりの名称が与えられる。本研究で使用したバージョンでは、代わりの名称は"Bon Echo"である。以後、Bon Echo を Mozilla Firefox と読み替えて差し支えない。

我々が Bon Echo に対して加えた改変は 4 点ある。1 つ目は、本研究のデータモデルの各要素をハンドリングする処理の実装である。2 つ目は、CSS に対する拡張の実装である。3 つ目は、イベントに対する拡張の実装である。そして 4 つ目は、本研究のデータモデルを視覚化するための描画処理の実装である。これらの改変点は、すべて Bon Echo の各コンポーネントのうち、コアにあたる Gecko に対するも

のである。図 12に本研究で作成した実装のシステム概観を示す。

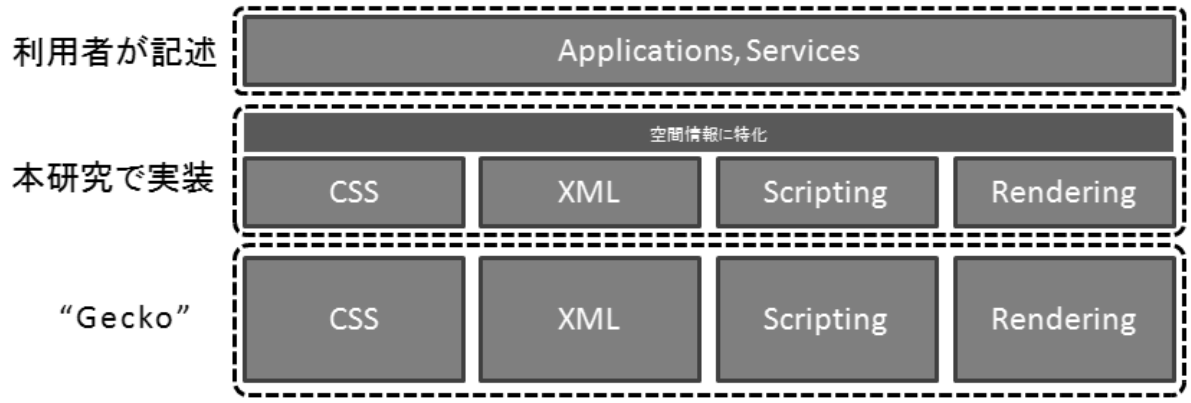

図 12 システム概観

本研究で作成した実装は、Gecko の CSS、XML、スクリプティング、およびレンダリングの実装を拡張子、空間情報に特化した機能を付加している。利用者は、これらの機能を利用してデータやプログラムを記述することで空間情報処理を行う。本研究では、この改変版の Gecko を Bon Echo に組み込んだ。

図 13は、改変された Bon Echo の起動時の画面写真である。図 14は、本研究のプラットフォームを用いて開発したアプリケーションの画面写真である。本研究で作成した実装は、基となった Bon Echo の機能を内包している。そのため、開発者は、Web サービスやスクリプト、ウェブページ等について既存のコンテンツを再利用できる。既存のコンテンツを再利用できることは、既に自ら作成したコンテンツを抱えている開発者たちの取り込みに有利である。

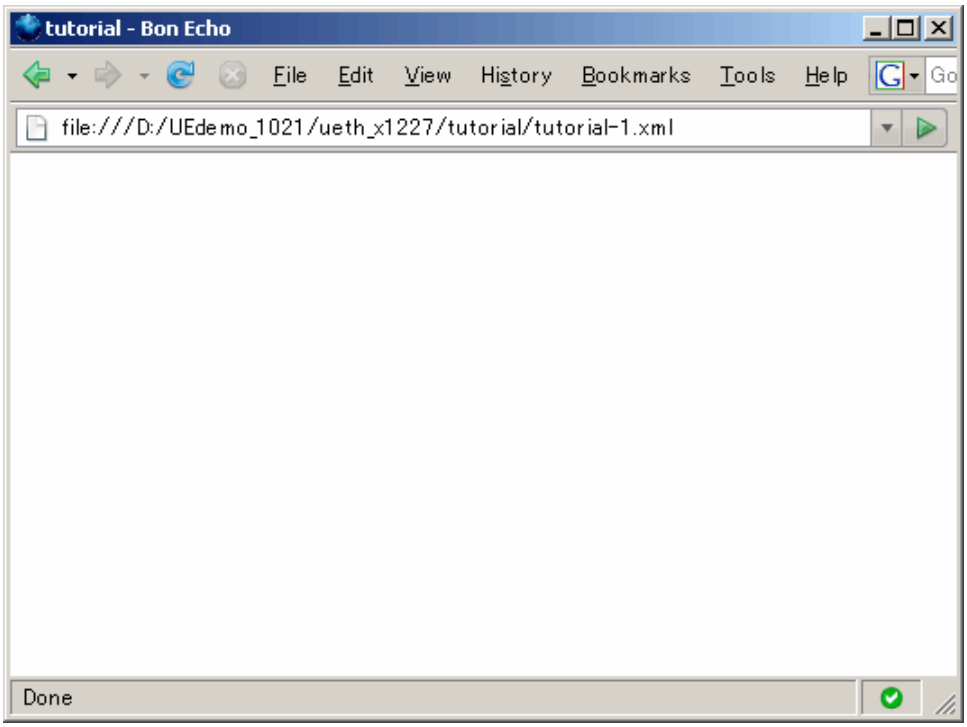

図 13 起動時の画面写真

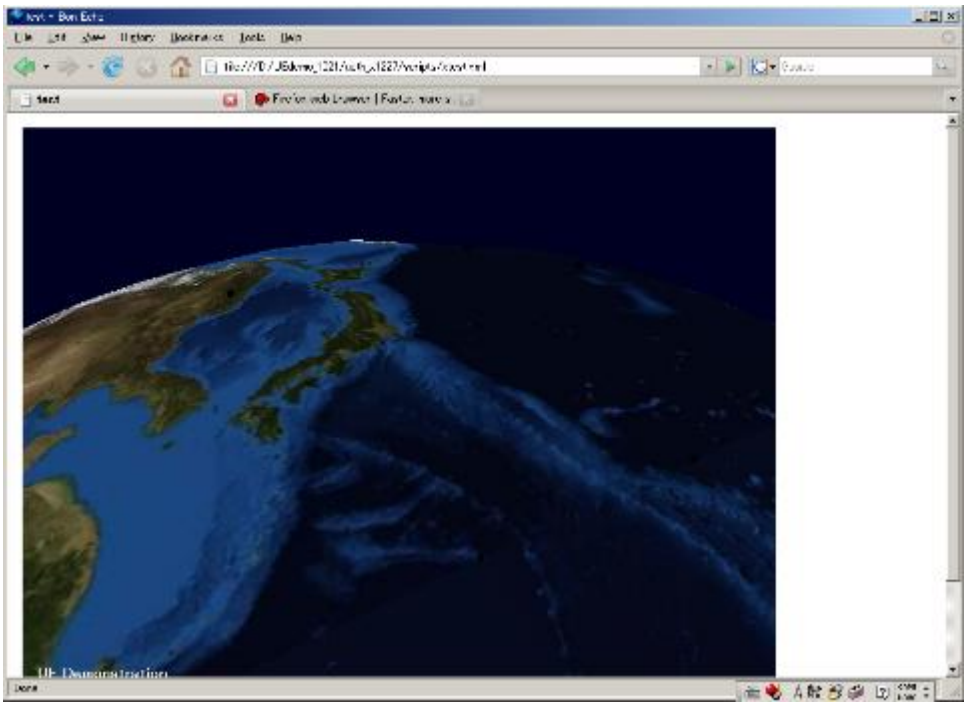

図 14 アプリケーションの画面写真

第4章

運用実験

本研究では、前章で解説したプラットフォームとその実装の有用性を確認するために、プラットフォームの実装を利用してコンテンツを開発し、さらにコンテンツを再利用する実験を行った。本章では、その実験の内容と結果について詳説する。

4.1. ライブラリの構築

本研究のプラットフォームは、極めて自由度の高いプラットフォームである。自由度の高すぎるシステムは、利用者が自身の目的に合った利用法を学習するまでのコストが高い。この問題に対する解決法は、適切な制約を設けることである。HTMLにおいても手法は見られる。現在の Web ブラウザは、全ての要素の視覚表現をスタイルシートで操作可能であり、文書の作成者が文書中の要素ひとつ毎に視覚表現を指定しながら文書を作成することも可能である。しかし、実際にそのような方法で文書を作成していては多くの知識と莫大な労力を必要とし、「低い障壁」という理想とは程遠いものになる。現実には、HTML の要素には予め既定のスタイルが割り当てられており、作成者が何もスタイルを指定しなくとも見出しや本文、引用部が適切に表現される。利用者は、既定のスタイルという制約の下で文書を記述することになるが、多くの場合それは利用者の生産性を上げる結果となる。

本研究のプラットフォームでは、以下の 2 つを提供し、適切な制約の下で利用者が迅速に学習できるよう配慮した。

1. スクリプトライブラリ
2. 既定スタイルシート

スクリプトライブラリは、利用者の操作による視点の移動や拡大、縮小などの機能を予め実装したものである。また、これらを利用するためのマニュアルも用意して利用者に提供した。このマニュアルは、本論文の付録に収録されている。

4.2. コンテンツの記述

実験では、空間情報を含むページを被験者に作成してもらった。図 15は、実験で作成したページの例である。実験では 12 個のページを 6 人で作成した。各ページとも、関西大学高槻キャンパス（大阪府高槻市）内のある地点の point 要素を持ち、その地点に関する説明を HTML で記述している。

図 15 実験で作成したページの例

4.3. コンテンツの再利用

本研究のデータモデルの特徴の一つは、利用者が独自の目的のためのデータを付加し、処理することができる拡張性であった。本節では拡張性の実験として、被験者が作成したページを再利用し、新しいコンテンツを作成する例を示す。

4.3.1. Yahoo! Pipes による統合 RSS の作成

まず、実験で作成したページを米 Yahoo! Inc.のサービス「Yahoo! Pipes」で加工する例を示す。当然ながら、Yahoo! Pipes は本研究のデータモデルに特有の処理を実装していないため、この実験では XML に関する汎用的な処理のみを行う。図 16に Yahoo! Pipes で加工を行っている様子を示す。

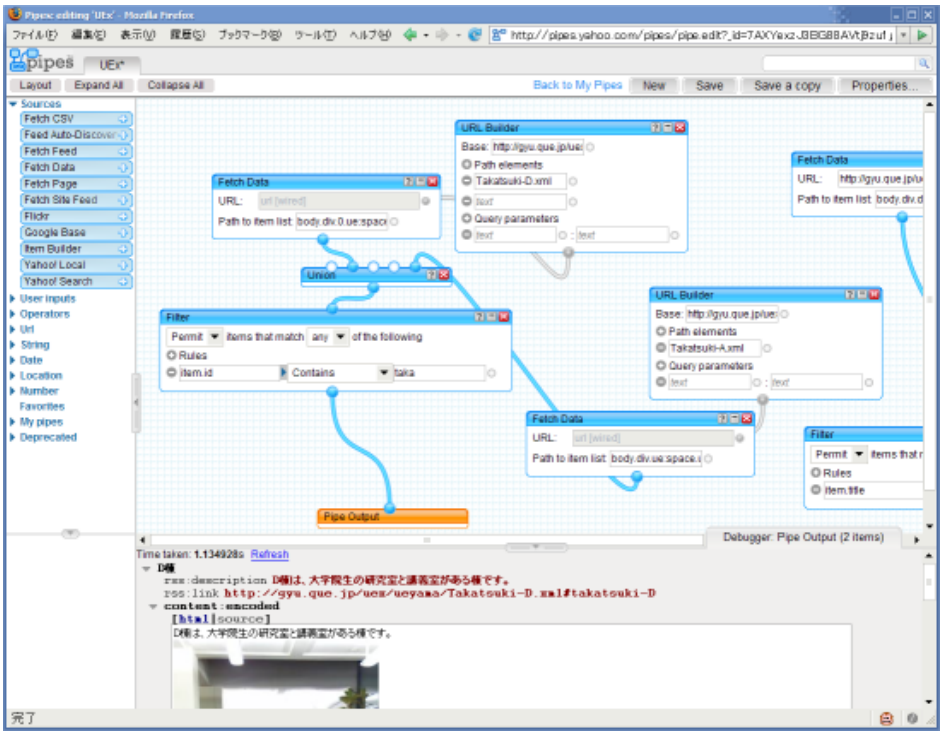

図 16 Yahoo! Pipes での加工

図 16に示した加工の結果、被験者が作成した複数のページから解説部分を抜き出した RSS が生成されている。RSS の生成手順を図 17に示す。

図 17 RSS の生成手順

4.3.2. Javascript アプリケーションによる案内図の生成

次に、Javascript により記述したアプリケーションを用いて被験者が作成したページを統合し、高槻キャンパスの案内図を生成する例を示す。 Yahoo! Pipes の例では、空間情報を利用せずに捨てていたが、この Javascript の例では空間情報も含めてデータを再利用する。

本節のアプリケーションは、被験者が作成した 12 個のページを取得して解析し、point を抜き出し、ひとつの earth 要素の上にコピーする。さらに、抜き出した point 要素のリストを作成する。このレンダリング結果は、図 18 生成されたに示すような案内図となる。本節のアプリケーションが生成する point 要素のリストをリスト 12に示す。

```
<ue:point lng="135.576" lat="34.8781" class="pagepoint c-1"/>
<ue:point lng="135.576" lat="34.8776" class="pagepoint c-2"/>
<ue:point lng="135.576" lat="34.8783" class="pagepoint c-3"/>
<ue:point lng="135.577" lat="34.8779" class="pagepoint c-4"/>
<ue:point lng="135.576" lat="34.877"   class="pagepoint c-5"/>
<ue:point lng="135.574" lat="34.8774" class="pagepoint c-6"/>
<ue:point lng="135.576" lat="34.8766" class="pagepoint c-7"/>
<ue:point lng="135.576" lat="34.8783" class="pagepoint c-8"/>
<ue:point lng="135.575" lat="34.8775" class="pagepoint c-9"/>
<ue:point lng="135.576" lat="34.877"   class="pagepoint c-10"/>
<ue:point lng="135.579" lat="34.8757" class="pagepoint c-11"/>
<ue:point lng="135.576" lat="34.8768" class="pagepoint c-12"/>
```

リスト 12

class 属性は元のページにあったものではなく、抜き出し処理の後で付加されたものである。これらの class 属性を利用して各点と凡例を色分けしている。各 point は図 18 生成されたで正しい位置にレンダリングされており、元のページに含まれていた空間情報を正しく解釈して再利用できたことがわかる。

4.4. まとめ

本章では、被験者にコンテンツを作成してもらい、それを再利用する実験を行った。再利用の実験結果から、本研究のデータモデルを直接利用しない場合、する場合のいずれも利用者が独自の目的に合わせてデータを加工し、再利用できることが

わかった。

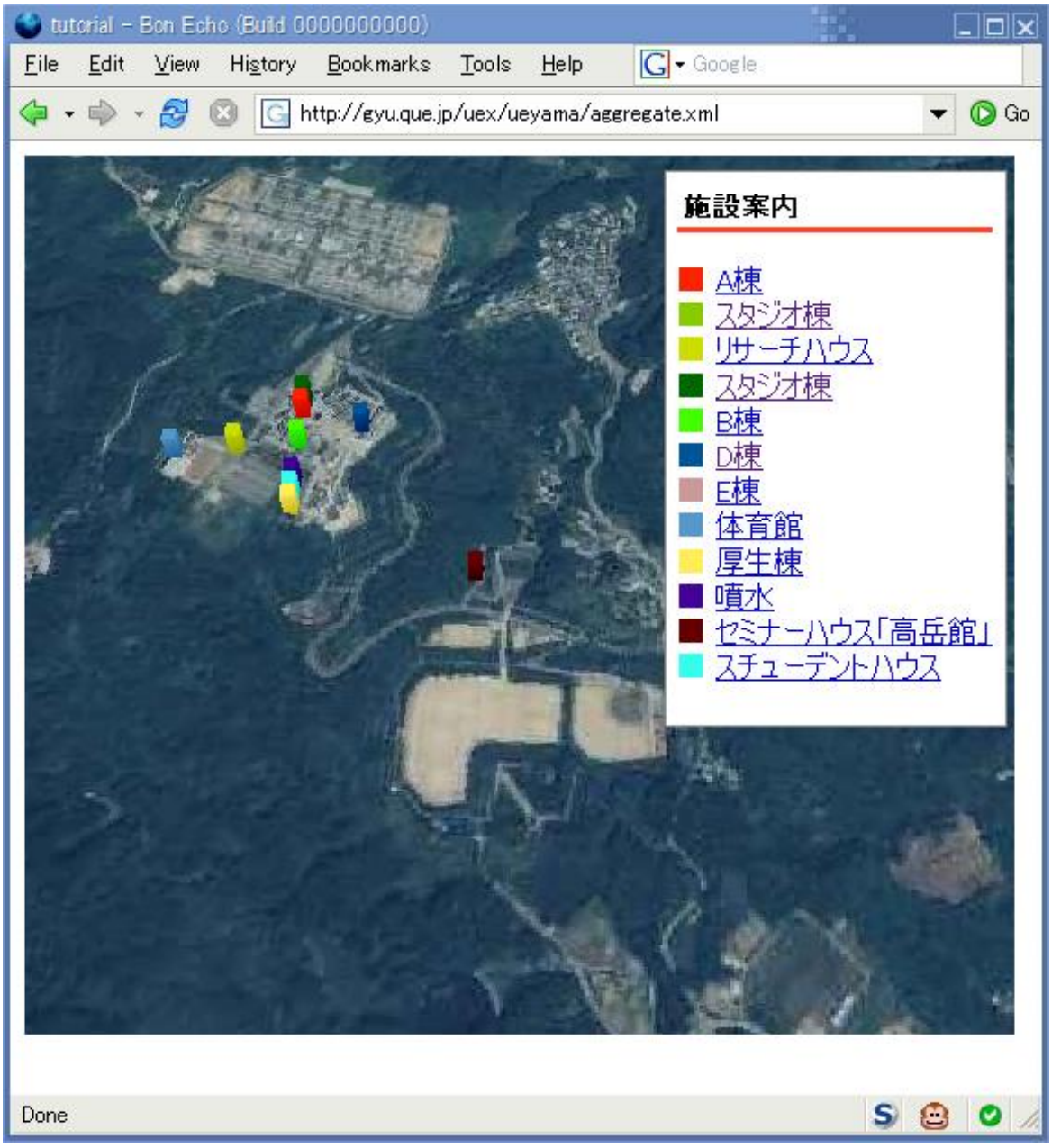

図 18 生成された案内図

第5章

結 論

本研究では、Roy による World Wide Web の成功に対する分析を参考にし、GIS が抱える問題点を解決することを提案した。また、その実例として、World Wide Web のアーキテクチャを取り入れた空間情報処理プラットフォームを構築し、その実装も行った。本章では、これらの取り組みの成果についてまとめ、今後の課題についても議論する。

5.1. 成果

本研究で構築した空間情報処理プラットフォームは、World Wide Web のアーキテクチャを取り入れ、コンテンツの利用者および作成者双方が、より容易に空間情報処理を行えることを目標としていた。具体的には、空間情報を記述するためのデータモデル、プレゼンテーションを記述するためのスタイルシート、およびプログラムを作成するためのプログラミングモデルの 3 つを定義した。これらは、XML、DOM、CSS といった World Wide Web の標準仕様を取り入れており、既存の技術を習得している開発者に受け入れられやすいものであった。

また、本研究では、プラットフォームを実装したシステムを作成し、その上でアプリケーションを構築する運用実験を行った。運用実験では、本研究のプラットフォームが World Wide Web のアーキテクチャを踏襲したコンテンツ作成方法及びユーザーインターフェイスを実現しており、従来の GIS より容易な空間情報処理を実現することを示した。

5.2. 課題

本研究のデータモデルは surface や point など物理的な情報を直接記述している。これは、レンダリング結果との対応がわかりやすく学習が容易という長所がある反面、意味的な情報を含んでいないため、面や点が何を意味しているかという情報を保存できないという短所がある。HTML において見出しや引用などの簡単な意味づけがあらかじめ用意されているように、空間情報においても池や山頂などといった、一般的な意味のある要素を用意すれば、より使いやすいデータモデルとなるであろう。それと同時に、点や線、面といった要素の物理的な形状をスタイルシートで制御できるようにし、要素と物理的形状を完全に分離することも必要だろう。

参考文献

1. [KITSUREGAWA2003] 喜連川 優: 特定領域研究「情報爆発(Info-plosion)」への新展開, 情報処理, 情報処理学会, Vol.48, No.8, pp. 917-919, 2007.
2. [YAN2003] 厳 網林: GIS の原理と応用, 日科技連出版社, 2003.
3. [ROY2004] Fielding, R.T., Architectural styles and the design of network-based software architectures [Ph.D. thesis]: Irvine, University of California, Available at http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm , October 2004.
4. [ADAM2004] Adam Mathes., Folksonomies; Cooperative Classification and Communication Through Shared Metadata. Computer Mediated Communication: LIS590CMC, December 2004.
5. [ERIC1999] RAYMOND, E. S., The cathedral and the bazaar, Available at http://catb.org/~esr/writings/cathedral-bazaar/ , 1999.
6. [GOTO2004] 後藤 寛, 上平好弘, 歌代和男, コーバー マーク, 小野寺久憲, 松崎康治, 野呂 治: ローコスト 3 次元 web GIS トータルシステムの構築と課題, 地理情報システム学会講演論文集, 地理情報システム学会, Vol. 13, pp. 303-306, 2004.
7. [MARK2003]コーバー マーク, カールソン マット, 歌代和男, 後藤 寛: ストリーミング技術を用いた 3 次元 webGIS 開発の意義と課題, 地理情報システム学会講演論文集 地理情報システム学会, Vol. 12, pp. 501-504, 2003.
8. [HAYASHIHARA2003] 林原尊敏, 越 雄一, 砥崎賢一:携帯電話向け三次元地理情報システムの研究, 地理情報システム学会講演論文集, 地理情報システム学会, Vol.12, pp. 509-512, 2003.
9. [YAMAMOTO2006] 山本大介, 清水敏之, 大平茂輝, 長尾確: Synvie: ブログの仕組みを利用したマルチメディアコンテンツ配信システム, 情報処理学会第 58 回グループウェアとネットワーク研究会, 情報処理学会, p13-18, 2006.
10. [W3C2007] W3C Recommendation, Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, Available at http://www.w3.org/TR/CSS21/ , 2007.
11. [W3C2006] W3C Working Draft, CSS3 Values and Units, Available at http://www.w3.org/TR/css3-values/ , 2006.

謝辞

本論文を書き上げるにあたり、柴崎先生および日本工営株式会社の今井龍一様にはお忙しいところ何度もご指導を頂きました。執筆経験の少ない私が本論文を完成まで漕ぎ着けることができたのは、ひとえに柴崎先生および今井様のご指導に縁るものであり、この場を以てお礼申し上げます。

また、副指導・副査を御担当頂いた有川先生、浅見先生には、本論文に関する貴重なご意見を頂き、論文の完成度を高めることができました。

最後に、本研究の実験にご協力いただきました関西大学総合情報学部の田中成典教授および田中研究室の学生の皆様、お忙しい時期に私の勝手にお付き合いいただきまして申し訳ございません。田中先生および学生の皆様の協力なしには、本論文は完成しませんでした。

附録

実験マニュアル

1. ソフトウェアの説明

本研究のソフトウェア(以下、UE)は、Mozilla Firefox(以下、Firefox) の一部を改変して実装されており、インストール方法および使用方法は、Firefox に準じます。Firefox の利用者であれば、次節および次々節を読み飛ばしても構いません。

2. インストール方法

アーカイブを適当なディレクトリに展開します

3. 起動方法

firefox.exe を実行します。ブラウザのウィンドウが表示されれば起動成功です。ウィンドウには Firefox で使用しているホームページが表示されます。

4. 空のドキュメントの作成

本節以降では、実際に UE の機能を使用してドキュメントを作成するための解説をします。まず、リスト 1 を tutorial.xml という名前で保存してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
    <html
        xmlns="http://www.w3.org/1999/xhtml"
        xml:lang="ja" lang="ja">
    <head>
        <title>tutorial</title>
    </head>

    <body>
    </body>
</html>
```

リスト 1

リスト 1 を UE で表示した様子を図 1 に示します。

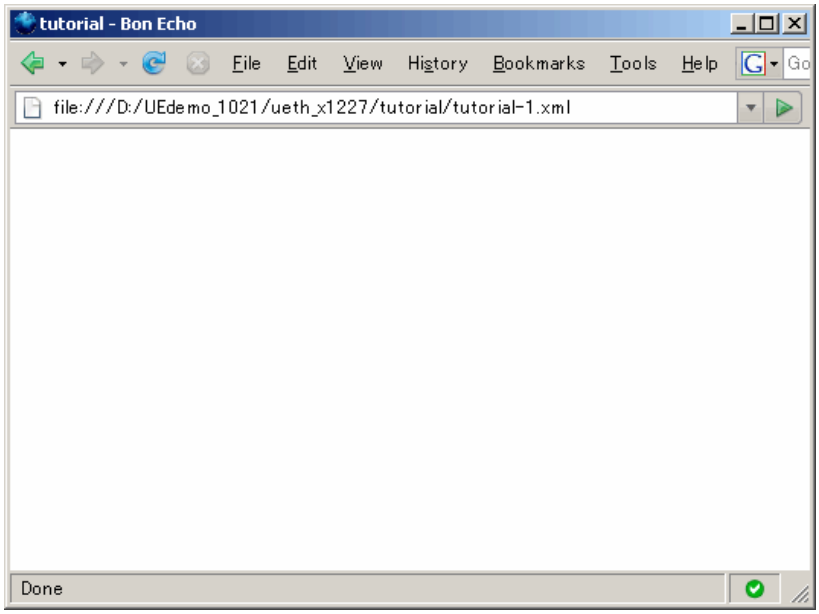

図 1

リスト 1 は、空の XHTML 文書であり、情報や機能を一切持っていませんが、UE の機能を利用するための土台となります。

5. 最低限の UE 要素の追加

本節では、前節で作成した文書に、空間情報を表す要素を追加します。リスト 1 にそれらを追加して得られるリスト 2 を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ue="http://www.csis.u-tokyo.ac.jp/UE/"
    xml:lang="ja" lang="ja">
<head>
    <title>tutorial</title>
    <style type="text/css">
            #space1 {
                    background: #aaa;
                    width: 480px;
```

```
                    height: 480px;
                    display: block;
                }
    </style>
  </head>

  <body>
    <div>
            <ue:space id="space1" style="background: #fff; width: 480px;
height: 480px;">
                    <ue:earth id="earth1">
                    </ue:earth>
            </ue:space>
    </div>
  </body>
  </html>
```

リスト 2

リスト 2 には UE 特有の要素がいくつか記述されています。5 行目では UE 特有の要素を使用するために名前空間を宣言しています。13 行目から 16 行 目では、ue: プレフィックスにより UE 特有の要素を記述しています。13 行目から 16 行目で記述している要素は以下の 2 つです。

- space
- earth

space 要素は、UE 特有の要素のルートであり、全ての要素のコンテナです。earth 要素は、後述する point(点)や surface(領 域)のコンテナです。これら 2 つの要素は、HTML 文書における html 要素や body 要素にあたり、文書の大枠を成す要素です。リスト 2 を UE で表示す ると、文書内に灰色の矩形がレンダリングされます。この灰色の矩形は、space 要素が占有している領域です。リスト 2 を UE で表示している様子を図 2 に 示します。

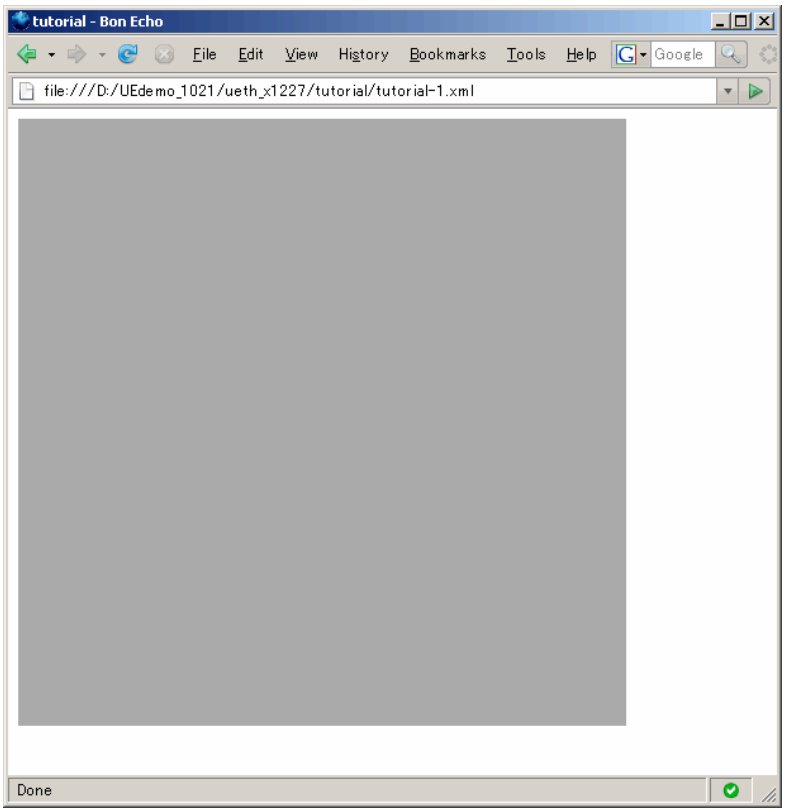

図 2

6. point 要素の追加

本節では、前節で追加した earth 要素の下にさらに要素を追加し、具体的な空間情報を記述します。リスト 2 に要素を追加して得られるリスト 3 を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ue="http://www.csis.u-tokyo.ac.jp/UE/"
    xml:lang="ja" lang="ja">
<head>
    <title>tutorial</title>
    <style type="text/css">
            #space1 {
                    background: #aaa;
                    width: 480px;
                    height: 480px;
                    display: block;
            }
```

```
    </style>
  </head>

  <body>
    <div>
        <ue:space id="space1">
            <ue:earth id="earth1">
                <ue:point lat="35.676880" lng="139.723450" />
            </ue:earth>
        </ue:space>
    </div>
  </body>
  </html>
```

リスト 3

リスト 3 は、新たに point 要素を含んでいます。point 要素は、ある地点を表す要素であり、その位置は lat 属性および lng 属性として記述されています。lat 属性は緯度、lng 属性は経度を表しています。

7. point 要素の視覚化

本節では、リスト 3 で追加された point 要素を視覚化し、画面上で表示します。リスト 3 で追加された point 要素を視覚化するためには、CSS を記述し、視覚化に必要な情報を UE に指示します。リスト 3 に CSS を記述して得られるリスト 4 を以下に示します。

```
  <?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  <html
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ue="http://www.csis.u-tokyo.ac.jp/UE/"
    xml:lang="ja" lang="ja">
  <head>
    <title>tutorial</title>
    <style type="text/css">
        #space1 {
            background: #aaa;
            width: 480px;
            height: 480px;
            display: block;
```

```
            }
            point {
                north: attr(lat, deg);
                west: attr(lng, deg);
            }
    </style>
</head>

<body>
    <div>
            <ue:space id="space1">
                <ue:earth id="earth1">
                    <ue:point lat="35.676880" lng="139.723450" />
                </ue:earth>
            </ue:space>
    </div>
</body>
</html>
```

リスト 4

リスト 4 に記述された CSS は、point 要素を描画する位置を指示しています。

```
north: attr(lat, deg);
west: attr(lng, deg);
```

の指示により、UE は、lat 属性を緯度、lng 属性を経度として解釈して描画の位置を決定します。つまり、指示内容によっては lat 属性および lng 属性とは無関係な位置に描画することも可能です。

8. スクリプトの追加

リスト 4 の CSS 記述により、point 要素は lat 属性および lng 属性で指示した位置に配置されました。しかし、リスト 4 を UE で表示しても point 要素を視認することはできません。point 要素を画面に表示するためには、画面に表示される領域を指示する必要があります。これは、カメラの 位置と向きを設定することに相当します。画面に表示される領域を指示するためには、CSS の独自拡張である transform-3d プロパティを使用します。transform-3d には座標変 換を表す 4 行 4 列の設定します。空間全体、つまり space 要素に座標変換を適用することにより、任意の領域を画面に表示することができます。しかし、座 標変換行列を作成し、それを記述する手間は大きいものです。そこで、スクリプトを利用して対話的に座標変換を行う方法を解説します。 リスト 4 にスクリプトを追加して得られるリスト 5 を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html
xmlns="http://www.w3.org/1999/xhtml"
xmlns:ue="http://www.csis.u-tokyo.ac.jp/UE/"
xml:lang="ja" lang="ja">
<head>
<title>tutorial</title>
<style type="text/css">
    #space1 {
            background: #aaa;
            width: 480px;
            height: 480px;
            display: block;
            -moz-user-select: none;
            cursor: -moz-grab;
    }

    point {
            north: attr(lat, deg);
            west: attr(lng, deg);
    }
</style>

    <script type="text/javascript" src="../scripts/UEBase.js"
charset="utf-8"></script>
    <script type="text/javascript" src="../scripts/UEBasicMath.js"
charset="utf-8"></script>
    <script type="text/javascript" src="../scripts/UEBasicGlobe.js"
charset="utf-8"></script>

    <script type="text/javascript">
function start()
{
var spaceElement = document.getElementById("space1");
var earthElement = document.getElementById("earth1");

var theEarth   = new UEBasicGlobe(earthElement);
var theViewer = new UEBasicViewer(spaceElement, theEarth);
}
```

```
    </script>
</head>

<body onload="start();">
<div>
    <ue:space id="space1">
            <ue:earth id="earth1">
                    <ue:point lat="35.676880" lng="139.723450" />
            </ue:earth>
    </ue:space>
</div>
</body>
</html>
```

リスト 5

UEBase.js、UEBasicMath.js、および UEBasicGlobe.js は、UE に付属するスクリプトです。これらは、スクリプ トによりアプリケーションを開発するための基本的な機能を実装したライブラリです。リスト 5 では、これらのライブラリを読み込み、以下に示す部分でライブ ラリの機能を呼び出しています。

```
var spaceElement = document.getElementById("space1");
var earthElement = document.getElementById("earth1");

var theEarth    = new UEBasicGlobe(earthElement);
var theViewer = new UEBasicViewer(spaceElement, theEarth);
```

このスクリプト片は、ライブラリに対して文書中の space 要素および earth 要素を利用するよう指示しています。"UEBasicViewer"は、ユーザーの入力を監視し、space 要素や earth 要素に対し適切な座標変換を行います。リスト 5 を UE で表示すると、point 要素を画面上で確認することができます。リスト 5 を表示している画面を図 3 に示します。

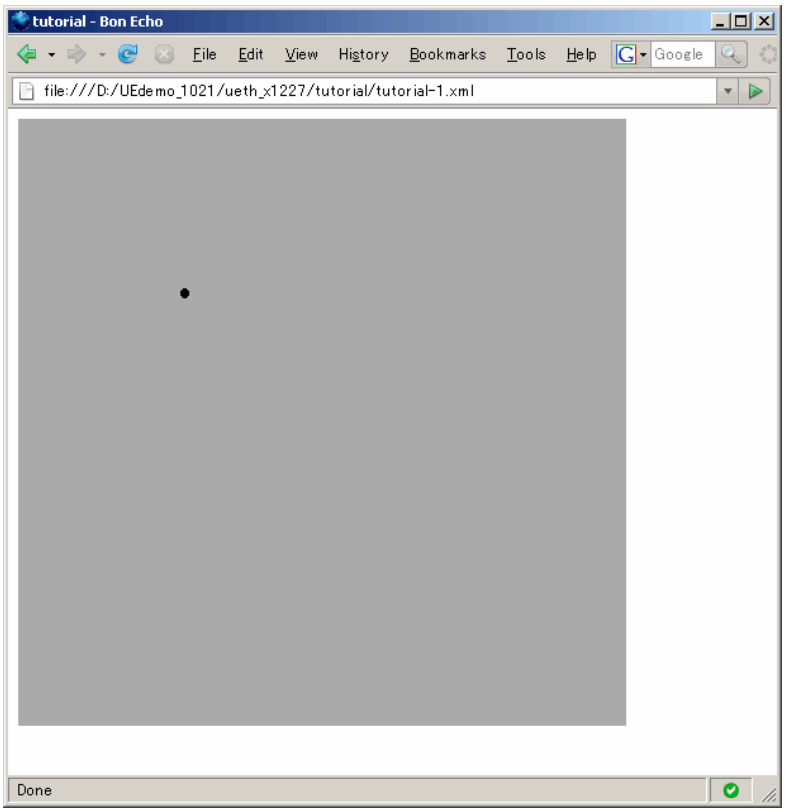

図 3

point 要素は、黒い小丸として表示されています。また、画面上をドラッグすることにより、黒い小丸が移動します。 "UEBasicViewer"は、ドラッグ操作を受け取ると、ちょうど地球儀を回すように earth 要素を回転させます。回転操作のイメージを図 4 に示 します。

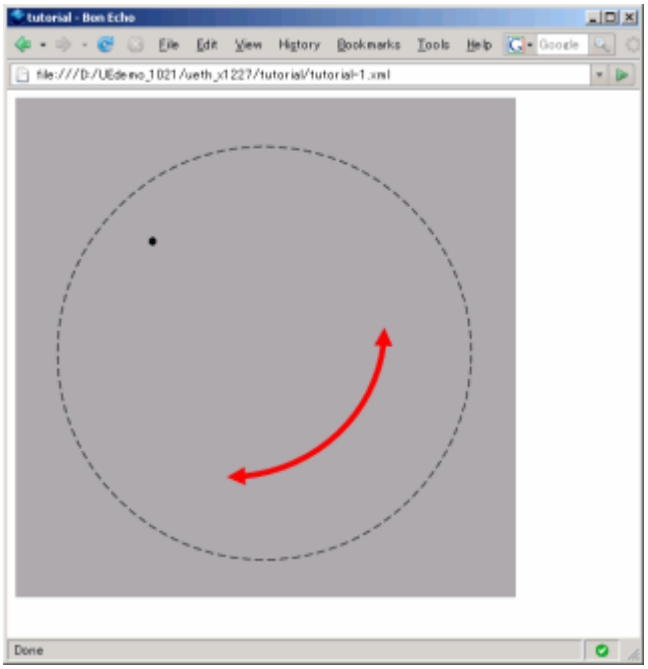

図 4

9. surface 要素の追加

本節では、リスト 5 に領域を表す surface 要素を追加し、視覚化します。リスト 5 に surface 要素を追加して得られるリスト 6 を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html
xmlns="http://www.w3.org/1999/xhtml"
xmlns:ue="http://www.csis.u-tokyo.ac.jp/UE/"
xml:lang="ja" lang="ja">
<head>
<title>tutorial</title>
<style type="text/css">
    #space1 {
            background: #aaa;
            width: 480px;
            height: 480px;
            display: block;
            -moz-user-select: none;
            cursor: -moz-grab;
    }

    point {
```

```
                north: attr(lat, deg);
                west: attr(lng, deg);
        }

        #tokyo-area {
                background-color: #def;
        }
    </style>

        <script type="text/javascript" src="../scripts/UEBase.js"
charset="utf-8"></script>
        <script type="text/javascript" src="../scripts/UEBasicMath.js"
charset="utf-8"></script>
        <script type="text/javascript" src="../scripts/UEBasicGlobe.js"
charset="utf-8"></script>

        <script type="text/javascript">
    function start()
    {
    var spaceElement = document.getElementById("space1");
    var earthElement = document.getElementById("earth1");

    var theEarth    = new UEBasicGlobe(earthElement);
    var theViewer = new UEBasicViewer(spaceElement, theEarth);
    }
        </script>
    </head>

    <body onload="start();">
    <div>
        <ue:space id="space1">
                <ue:earth id="earth1">
                        <ue:point lat="35.676880" lng="139.723450" />
                        <ue:surface id="tokyo-area">
                                <ue:point lat="35.8" lng="139.6" class="rect-NW"
/>
                                <ue:point lat="35.8" lng="139.9" class="rect-NE"
/>
                                <ue:point lat="35.6" lng="139.9" class="rect-SE"
/>
                                <ue:point lat="35.6" lng="139.6" class="rect-SW"
/>
                        </ue:surface>
```

```
            </ue:earth>
        </ue:space>
    </div>
    </body>
    </html>
```

リスト 6

前節までの作業で既に視覚化に必要な記述を行っているので、この surface 要素は直ちに画面上に表示されます。リスト 6 を UE で表示した様子を図 5 に示します。

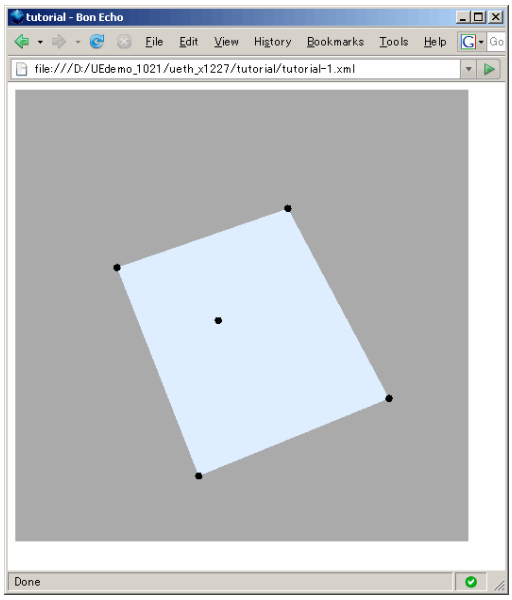

図 5

マウス操作を行い、リスト 4 で配置した point 要素に接近すると、surface 要素も視認することができます。

10. ライブラリの解説

本節では、ライブラリの動作について解説します。ライブラリの主な機能の一つである地球儀の回転がどのように実装されているか解説することにより、 UE でのプログラミングのヒントを示します。 地球儀の回転機能のフロントエンドは、UEBasicGlobe.js に実装されています。UEBasicGlobe.js では、DOM 標準のイベントハ ンドリング機能を使用してユーザのマウス操作を監視しています。マウス操作の監視を開始する処理は以下のように実装されています。

```
    spaceElement.addEventListener("mousedown",
this.onMouseDown.bind(this) , false);
    spaceElement.addEventListener("mousemove",
```

```
this.onMouseMove.bind(this) , false);
  spaceElement.addEventListener("mouseup",  this.onMouseUp.bind(this)     ,
false);
```

addEventListener は DOM イベントモデル標準の API であり、HTML などの要素と全く同じ方法でイベントの監視を行っていることがわかります。 続いて、マウス操作が行われた際に呼び出されるハンドラ側のコードを示します。まず、マウスのボタンが押下された際のハンドラを以下に示します。

```
  onMouseDown: function(e) {
      this.localPt = {};
      var sPt = {};

      if (e.button == 0)
            this.leftDrag = true;
      else if (e.button == 2)
            this.rightDrag = true;

      this.calcLocalPosition(e.clientX, e.clientY, this.localPt);
      this.calcNormalizedPosition(this.localPt.x, this.localPt.y, sPt);
      this.prevHand  =  UEBasicMath.computeGlobePick(this.currentViewProj,
this.earthWrapper.earthElement.radius, sPt.x, sPt.y);
      this.currentRotation.axis = null;
  },
```

5 行目から 8 行目にかけては実に素朴なコードです。ドラッグが開始されたことを示すために、左右ボタンいずれかのフラグをセットしています。10 行 目以降は、やや込み入ったコードです。まず、10 行目および 11 行目でマウスカーソルの座標を変換しています。変換後の座標は、画面の中心が(0, 0)で左上が(-1, 1)なる正規化された座標系です。 12 行目はおそらく最も重要な行です。12 行目の処理は、マウスカーソルが指している位置を地球儀の球面座標に変換します。 最後の 13 行目は、後の処理で使う変数の初期化です。 続いて、マウスカーソルが移動した際に呼び出されるハンドラの一部を以下に示します。マウスのボタンが押されている状態でこのハンドラが呼び出されると、 ドラッグの処理を行います。

```
  onMouseMove: function(e) {

      var pt = {};
      this.calcLocalPosition(e.clientX, e.clientY, pt);

      if (this.prevHand && this.leftDrag)
      {
            var sPt = {};
            this.calcNormalizedPosition(pt.x, pt.y, sPt);
```

```
            var vHand =
UEBasicMath.computeGlobePick(this.currentViewProj,
this.earthWrapper.earthElement.radius, sPt.x, sPt.y);

            if (vHand)
            {
                this.slerpPickedPoints(this.prevHand, vHand);
                this.prevHand = vHand;
            }
    }
  // 略
```

3 行目から 10 行目にかけて、マウスボタン押下時のハンドラと同様に、球面座標の計算を行っています。そして、14 行目では、前回求めた球面座標と今回求めた球面座標を引数として slerpPickedPoints というメソッドを呼び出しています。slerpPickedPoints の実装を以下に示します。

```
  slerpPickedPoints: function(p1, p2) {

    var N = UEBasicMath.crossProduct3(p2, p1);
    UEBasicMath.normalize3(p1);
    UEBasicMath.normalize3(p2);
    UEBasicMath.normalize3(N);

    var dp = UEBasicMath.dotProduct3(p1, p2);

    var angle = Math.acos(dp);

    this.currentRotation.angle = angle;
    this.currentRotation.axis = N;

    var m = UEBasicMath.createRotationAxisMatrix(N, angle);
    m.m41 = m.m42 = m.m43 = 0;

    var pThis = this;
    window.setTimeout(function(){
    pThis.earthWrapper.applyTransform(m);
    }, 1 );
  },
```

slerpPickedPoints メソッドは、パラメータとして渡された 2 本のベクトルの外積となるベクトルを軸として、2 本のベクトル間の角度だけ地球儀を回転させます。これは、地球儀をマウスのドラッグに追従して回転させる結果となります。

11. 画像の貼り付け

本節では、ラスタ画像を surface に貼り付けます。図 6 に示す画像は、新宿付近の航空写真です。この画像を surface に貼り付け、surface 全体を覆うように指定します。

surface に貼り付ける画像を指定する CSS を以下に示します。航空写真のファイルは shinjuku512.jpg に保存されているものとします。

```
#tokyo-area {
    background: #fff url(shinjuku512.jpg);
}
```

surface に貼り付ける画像を指定するためには、background プロパティを設定します。background プロパティは省略記法なので、この CSS は実際には background-image プロパティにファイル名を指定しています。

図 6

レンダリング結果を図 7 に示します。

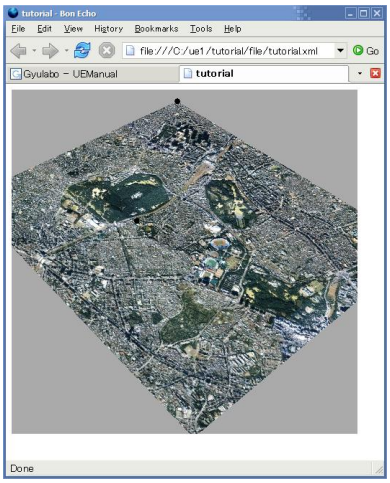

図 7