NINJAシステムとそのVRMLによる実現法

NINJAシステムとそのVRMLによる実現法

荒　屋　真　二　(情報工学科)

The NINJA system and its implementation with VRML

Shinji ARAYA (Department of Computer Science and Engineering)

Abstract

Three-dimensional virtual environment is a new prospective user interface between human and computer in the future Internet society. The Web3D consortium created VRML (Virtual Reality Modeling Language) as an international standard for the interactive 3D media for the WWW. This paper proposes a new 3D-map-based navigation interface for the virtual environment, and has implemented it as the NINJA system with the VRML. A virtual environment and 3D map are separately displayed in the same window to avoid the obstacle problem and implementation difficulties. An avatar movement in the 3D map is mutually linked with the user's movement in several manners from which users can freely select according to the situation. The files of 3D map interface are completely separated from the files of virtual worlds. This enables content creators concentrate on the virtual world without worrying about the 3D map, extending the applicability of the 3D-map-based navigation interface.

Key words: *virtual environment, navigation, user interface, 3D map, avatar, VRML, Web3D, WWW*

1．まえがき

インターネットの普及・進展に伴い，コンテンツ自体の多様化，複雑・大規模化，高機能化が急速に進んでいる。そのひとつの潮流として，これまで実験室や展示室などに限られていた仮想現実感技術をベースとした3次元仮想環境がインターネットに浸透しつつある[1]。3次元仮想環境は人間と現実世界とのインタラクションを自然な形で人間とコンピュータとのインタラクションに写像可能であり，新しいユーザインターフェイスとしての役割が期待できる。このようなインタラクティブWeb3Dのために制定された国際標準としてVRML (Virtual Reality Modeling Language) があり，現在も標準化作業が続けられている[2]。本研究の目的は3次元仮想環境をインターネット上で幅広く利用可能とするために，国際標準のVRMLと標準的I/Oデバイスを用いた仮想3次元空間ナビゲーションインターフェイスを確立することにある。

従来研究は実験室内での大掛りなシステム開発が主流であり，HMD (Head Mounted Display) や6自由度トラッカのほか，特殊で高価な専用I/Oデバイスの使用を前提にしていた。現行のVRMLブラウザのユーザインターフェイスはマウスとキーボードのみを前提としているが[1]，一般ユーザにとってナビゲーション操作は容易ではない。特にその機能は移動に限られており，世界全体の把握，現在地確認，ターゲット探索など経路計画支援機能は完全に欠落している。この機能は仮想世界が大規模複雑であるほど重要度を増す。経路計画支援に大きな効果を発揮するのはマップであ

平成13年10月31日受付

り，仮想環境においても2Dマップあるいは3Dマップを利用したインターフェイスが研究されてきた。両マップにはそれぞれ一長一短がある。本研究ではまだ未成熟であるが将来性が高いと考えられる3Dマップを取り上げる。3Dマップを用いた仮想環境ナビゲーションの研究はスタンドアロンで特殊なI/Oデバイスを用いたものが多く[4]－[6]，インターネットでの一般利用を考えていない。一方，VRMLを用いたナビゲーションに関する研究では3Dマップを扱ったものはやはり特殊なI/Oデバイスを用いており[7]，それ以外はマウスベースであるが2Dマップ[8]や特殊言語によるデータベース検索を利用している[9]，[10]。

本研究は3Dマップを用いた仮想環境ナビゲーションインターフェイスを提案するものであり，標準的なモニタとマウスのみの使用を前提とし，Web対応である点に第1の特徴がある。これは国際標準のVRMLと通常のWebブラウザだけでシステムの実装を行うことを意味する。VRMLを用いた数少ない研究は[7]，特殊なI/Oデバイスを使用しているだけでなく特殊なグラフィックボードのみで動作するシステムであり，Web対応にするのは容易でない。本研究の第2の特徴は，3Dマップを仮想世界の中のオブジェクトとした従来研究に対し[3]－[7]，3Dマップを仮想世界から完全に分離表示した点である。これによって仮想世界がマップによって部分的に遮蔽される問題，マップ画質低下の問題，実装上の問題などを回避できるようになった。仮想世界と3Dマップとの双方向連動は外部アプリケーションが行う。第3の特徴は，仮想世界におけるユーザの移動とマップ表示との連動方法を複数準備し，簡単に切替ができるマルチ連動マップとした点である。本来仮想空間移動を支援するための3Dマップにおいて，3Dマップという仮想空間における視点制御が新たな問題になっているのが現状といえる[6]。本研究では，仮想環境自体を3Dマップという明確な用途に利用する場合には，その典型的な利用形態に対応できる視点制御機能だけがあれば十分であると考える。これによってユーザのマップ操作の煩雑さを軽減した。第4の特徴は，ナビゲーションの対象である仮想世界のファイルと3Dマップを含むインターフェイスのファイルとを完全に分離した点である。これにより仮想世界クリエータは3Dマップのことを意識することなく制作に専念できる。また，インターフェイスのポータビリティも高まり，種々の仮想世界に対して簡単に応用できるようになる。3Dマップは2Dマップに較べてマップ自体を作成する手間がかからないという利点があるが，本研究はこれをさらに一歩進めたわけである。ファイルの分離を可能にするために，仮想世界と3Dマップの連動機能を自動的に作り出す外部アプリケーションを開発した。このプログラムは仮想世界にユーザの現在位置を計測するセンサやユーザの位置を制御するための視点などを追加する。また，同じ仮想世界を別のオブジェクトとしてHTML文書内に3Dマップとして埋め込み，ユーザと連動しかつマウスでドラッグ可能な3Dアバタを追加生成する。我々は以上の考え方に基づいた双方向マルチ連動3Dマップベース仮想環境ナビゲーションシステムNINJAを構築した。

本稿ではまず2章においてNINJAシステムの概要とその特徴について述べる。次に3章では3Dマップに表示される3Dアバタのモデリングについて，現在地のリアルタイム表示と空間ナビゲーションの観点から述べる。4章ではユーザの移動に伴う3Dマップの表示方法，すなわちユーザ視点とユーザの分身の視点との種々の連動法について述べる。5章ではVRMLによるNINJAシステムの実装法について述べる。6章では関連研究について，7章では本研究のまとめと今後の課題について述べる。

2．NINJAシステムとは

NINJAシステムは仮想3次元空間ナビゲーションのためのWeb対応マウスベース双方向連動3Dマップである。図1に示すように，仮想3次元空間を効率的に移動するためにユーザは分身の術を使って自分の分身を上空に放ち，分身の視点から見た画像（分身視

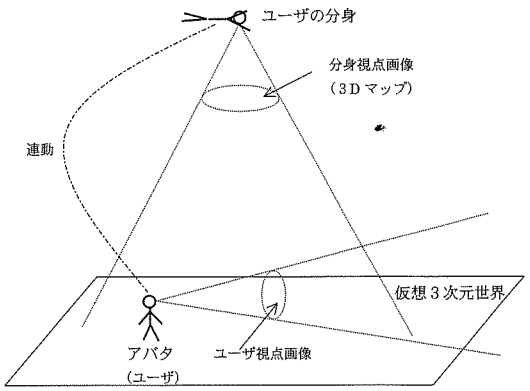

図1　NINJAシステムの基本原理

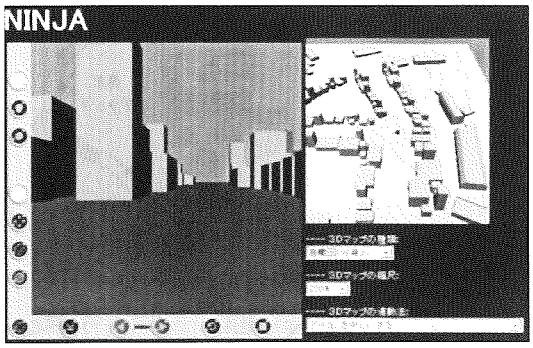

図２　NINJA システムの画面構成例

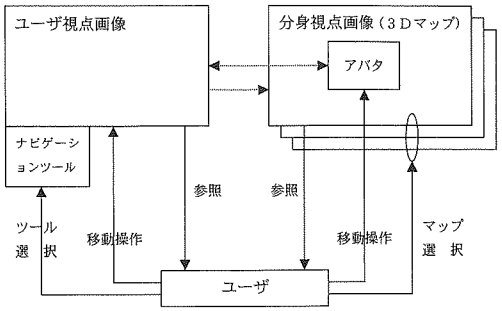

図３　NINJA システムの基本的利用形態

点画像）を３D マップとしてナビゲーションに利用する。NINJA システムの画面構成の１例を図２に示す。この例では HTML 文書にユーザ視点画像（左）とひとつの３D マップ（右）を埋め込み，３D マップの下に各種選択メニューを配置している。ユーザが VRML ブラウザ付属のナビゲーションツールにより仮想世界を移動すると，３D マップにはユーザ自身の位置と向きを示す３D アバタ（ユーザの化身）がリアルタイム表示される。逆に，３D マップ上のアバタをマウスドラッグして移動させることによってユーザ自身の位置や向きを制御することも可能である。すなわち，NINJA システムの３D マップは単に仮想世界の大局的把握と現在地表示だけでなく仮想世界移動のための入力手段にもなっている。

NINJA システムではユーザの分身をナビゲーションに有効と思われるところに複数放つことができる。複数の分身視点画像は切替表示にしても同時表示にしてもどちらでもよい。もちろんグラフィック性能によってフレームレートの点から同時表示可能な数は制限を受ける。複数の分身の動きをユーザが逐一指示するのは困難なので，それぞれの分身はユーザの動きに自動的に連動する。もちろんマウス操作により分身視点をきめ細かく制御することもできる。分身視点の連動方法もマップの使用目的に応じて複数準備し，ユーザが選択可能とする。通常分身視点はアバタが見えるように設定されるが，必ずしもそうする必要はない。また，状況によってはアバタがオブジェクトにさえぎられて見えなくなることも予想される。そのようなときのためにアバタは可変長の“アドバルーン”をあげることが可能である。以上のよう，NINJA システムはユーザを仮想の“忍者”に見立て，現実世界では不可能に近い忍術を使って各種ナビゲーション機能を実現している。

NINJA システムの基本的利用形態は，(1)マップ参照型移動，(2)マップ操作型移動，の二つに大別できる（図３）。マップ参照型移動では，VRML ブラウザのナビゲーションツールから移動モードを選択し，ユーザ視点画像内でのクリックやドラッグ操作により移動する。この場合３D マップは現実世界でのマップ利用時のように必要に応じて参照される。マップ操作型移動では，３D マップのなかから適当なマップを選択し，アバタをドラッグすることによってアバタの位置や向きを変え，ときどきユーザ視点画像がどうなっているかを参照する。アバタのドラッグ中にユーザ視点画像を主に参照し，３D マップを時々見ることも可能である。

３．３D アバタ

３．１　アバタの３D モデル

アバタの形状は，①ユーザの位置と向きが明瞭にわかるようにすること，②ユーザの位置と向きを個別に制御できるようにすること，③なるべくシンプルにすること，の３点から図４のように円筒と直方体を組み合わせてモデリングした。円筒はアバタの頭，直方体は鼻と考えてよい。円筒の重心がアバタの位置（正確にはユーザ視点の位置）に，鼻の向きが視線方向に対応する。

アバタのサイズは一定ではなく，３D マップの縮尺やアバタの高度に連動して自動的に変化し，マップ上での表示サイズが適当な大きさに維持される。例えば大規模な仮想都市で分身視点の高度が非常に高い場合にはアバタのサイズを大きくし，パソコンの内部に潜入するときなどには小さくする。このアバタ表示サイズのデフォルト値はあらかじめ指定しておくが，ユー

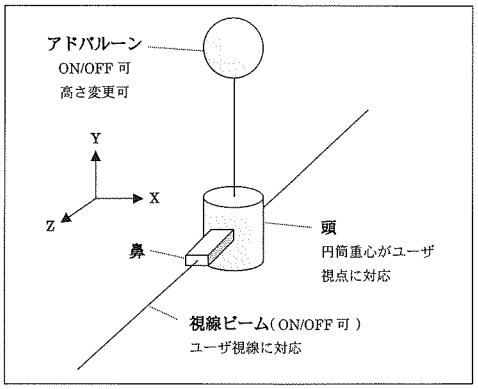

図4　アバタの3Dモデル

ザはナビゲーション中に状況に応じて自由に変更可能とした。

アバタの付属品として，視線ビーム，アドバルーン，仮想 GPS の三つを導入した。視線ビームはアバタの視線をビームとしてビジュアライズするもので，正面遠方のオブジェクトの把握に役に立つ。アドバルーンはアバタがオブジェクトの陰や建物内部に入ってユーザから見えなくなったときに使用するもので，大きさはアバタサイズと同じであるが高さを自由に変更できる。アドバルーンのドラッグによってアバタを移動させることもできる。仮想 GPS（Global Positioning System）はアバタの現在地の正確な3次元座標を知りたいときに使用する。

3．2　アバタドラッグによる空間移動

ユーザはアバタをマウスでドラッグすることにより3Dマップ上のアバタを移動させることができる。これに連動して仮想世界におけるユーザ自身の位置も変化する。すなわちユーザは自分の化身であるアバタを操作することにより，自分自身を移動させることができる。

アバタを上から見ているときには，頭をドラッグすることによりアバタを XZ 平面内で移動させることができる（図5a）。XZ 平面とは地面や床など仮想世界のベース平面に平行な平面を意味する。このときアバタの視線方向は変化しない。一定方向を見ながら移動するときに有用な制御方式である。例えば壁にかかった絵画を見ながら移動するような場合である。鼻をドラッグすることによりアバタの視線方向を Y 軸周りに回転させることができる（視線を左右方向に変更する）。このときアバタの位置は変化しない。

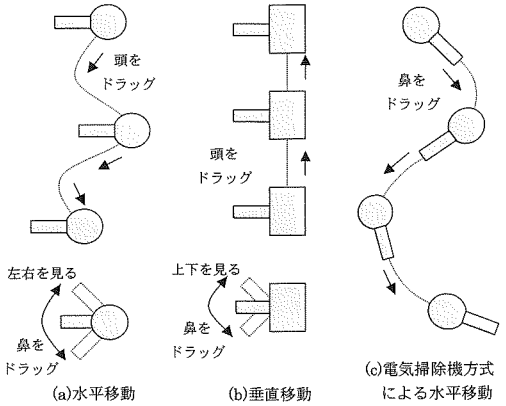

図5　アバタドラッグによるユーザ視点制御

アバタを横から見ているときには，頭をドラッグすることによりアバタを Y 軸にそって上下に移動させることができる（図5b）。Y 軸とは仮想世界のベース平面に垂直な座標軸である。これによってユーザ視点の高さを制御する。このときアバタの視線方向は変化しない。鼻をドラッグすることによりアバタの視線方向を X 軸周りに回転させることができる（視線を上下方向に変更する）。このときアバタの位置は変化しない。

アバタの位置と向きを同時に制御する機能として電気掃除機方式を導入した。これは電気掃除機のホースを引っ張ると本体の位置と向きを同時に変更できるように，アバタの鼻をドラッグすると鼻の向きだけでなく頭も一緒に付いてくるようにしたものである（図5c）。移動しながらいろいろな方向を見たいときに有用な視点制御方式である。

なお，アバタドラッグによって視線を Z 軸周りに回転させる（視線を視軸回りに回転させる）ことはできないようにしている。これはこの操作の用途が少ないことと，アバタが煩雑になるのを避けるためである。VRML ブラウザの“FLY＋ROLL”ツールによってユーザ視線を回転させることは可能であり，これによってユーザ視点画像は傾くが3Dマップのアバタは変化しない。

4．ユーザ視点と分身視点の連動

ここでは NINJA システムで主に使用する鳥瞰マップについて，ユーザ視点と分身視点との連動方法を中心に説明する。

4．1　鳥瞰マップと側面マップ

分身視点の位置と視線方向は3Dマップの性格を決定付ける。NINJA システムでは大きく2種類のマップを利用する。

(1) 鳥瞰マップ：仮想世界をその上空から眺望したシーンをマップとするもので，XZ 平面内でのナビゲーションに使用する。

(2) 側面マップ：仮想世界をその側面から眺望したシーンをマップとするもので，XY 平面内でのナビゲーションに使用する。

本稿では簡単のために分身の視線と仮想世界のベース平面とのなす角度を90度となるような鳥瞰マップを考える。少し斜めに見た場合もほぼ同様の議論が可能である。鳥瞰マップは仮想都市内のナビゲーションなどで有用とされるが[6]，このマップだけでは不十分な場合もある。例えば，ビルの屋上に立って下を見るなどの操作をするには高さ方向の制御と視線を上下に回転させる必要があり，側面マップが有用となる。移動対象の空間が3次元であるにもかかわらず従来研究の大部分がベース平面における2次元的移動のみを扱っている。

4．2　鳥瞰マップの種類

ユーザが移動すると鳥瞰マップ上のアバタも連動して移動する。この連動は一意に決まるが，ユーザ視点と分身視点の連動は一意には決まらない。ゆえに，分身視点をユーザ視点の移動にどのように連動させるかによって種々の特徴をもつ鳥瞰マップを構成できる。それらは二つの視点の位置連動法と視線連動法の組合せだけ存在する。位置連動法としては次の4種類を考えた。

① 分身視点を仮想世界中心点の真上に配置する（図6a）

② 分身視点をアバタの真上に配置する（図6b）

③ 分身視点をアバタ前方の真上に配置する(図6c)

④ 分身視点を，仮想世界とアバタの中点の真上に配置する（図6d）

①は位置連動の特殊な場合であり，仮想世界全体を眺望できる位置に分身視点を固定する方法である。仮想世界が小規模なときには良いが，大規模な場合にはマップが細かくて見づらくなる。②はアバタが常にマップの中心に来るように分身視点を連動させる方法で，マップの表示範囲がリアルタイムで更新される。③は②に較べアバタの進行方向領域のマップに占める表示割合を増加させたものである。後方よりも前方のマップのほうが移動に役立つことが多いという考えに基づく方法といえる。④はアバタの表示位置によって仮想世界の中心点がどちらの方向にあるのかを知ることができる。この方法は仮想世界が大規模のときにアバタを常に見えるようにしようとすると縮尺が小さくなりすぎる可能性がある。

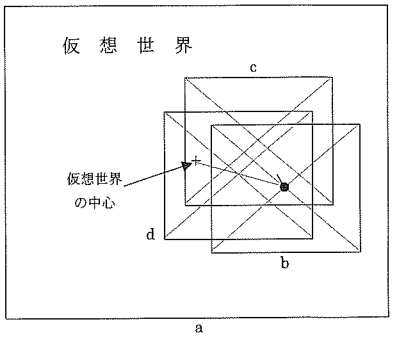

図6　ユーザ視点と分身視点の位置連動法

視線連動法としては従来からよく議論されている次の2種類を考えた[7]。

(a) 非回転マップ：仮想世界の北がマップの上にくるようにする。

(b) 回転マップ：ユーザの視線方向が常にマップの上を向くようにする。

非回転マップではユーザが向きを変えるとアバタの向きが変化し，マップの向きは変化しない。回転マップではユーザが向きを変えるとマップ自体が回転し，アバタは常にマップの上を向いている。

鳥瞰マップの有用性を大きく左右するもうひとつの要素としてマップの縮尺がある。仮想世界が大規模な場合，鳥瞰マップの縮尺を自由に変更できることが不可欠となる。縮尺が小さすぎるとマップが細かすぎて見づらく，アバタドラッグによる視点制御の精度も落ちる。逆に，縮尺が大きすぎると全体を見渡せなく，移動のためのアバタドラッグ量が増大する。マップの縮尺は分身視点の高度を変更することによって容易に変更可能である。ゆえに，NINJA システムでは VRML ブラウザの PAN を用いて鳥瞰マップの縮尺を連動的に変更可能であるが，初心者のためにドロップダウンメニューによりいくつかの縮尺の中から選択できるようにした。従来は6自由度トラッカを用いて3Dマップを操作していたため，縮尺のレンジが腕のストローク長によって大きく制限されていた[4]－[7]。大規模

な仮想世界に対して，腕をいっぱいに伸ばしたときに3Dマップが世界全体をカバーするようにすると腕を曲げたときの縮尺が小さ過ぎて使いものにならない。縮尺のレンジを大きくとるとわずかな腕の動きで縮尺が大きく変化し，極めて見づらいマップとなる。移動中はマップの縮尺を固定し，必要に応じて大きなレンジの縮尺の中から自由に選択できる本方式は使いやすく用途も広いと考えられる。

4．3　側面マップ

側面マップは仮想世界のオブジェクトの高さ情報を入手するため，あるいはユーザ視点の高さを制御するために使用される。単にこれだけであるならば垂直スライダーを用いてアバタの高さと向きだけを表示及び制御できればよい。これは単にアバタだけを側面から見たようなものである。仮想世界もアバタと同時に側面から見ることの利点はあまりなく，逆に欠点のほうが多い。例えば，オブジェクトが邪魔でアバタが見えない，オブジェクト同士が視界をさえぎる，特に大規模な仮想世界では周辺部は良いが内部が良くわからない。現実世界のマップでも側面マップはほとんど利用されていない。高層ビルやタワーなどが単独で存在する世界では側面マップが役に立つが，一般に利用頻度は少ないと考えられる。

4．4　マルチ連動マップ

NINJA システムでは上述した鳥瞰マップと側面マップ，位置連動法，視線連動法，縮尺の4項目に関して HTML 文書内のラジオボタンあるいはドロップダウンメニューによりユーザが切り替えできるようにした。このようなマルチ連動マップの利点はユーザのマップ操作が簡単になるということである。3Dマップの導入が仮想世界のナビゲーションに有用であるための条件は，3Dマップの操作が簡単であるとういうことである。さもなければ，視点制御問題を仮想世界から3Dマップに移しただけでなんら本質的な問題解決にはならないからである。3Dマップはナビゲーション対象の仮想世界と同様に3次元であり，3Dマップにおける視点制御は仮想世界における視点制御と本質的には変わらない。しかし，マップという明確な用途が存在する場合には，上述したような視点のなかから選択が可能であれば3Dマップにおける視点制御は十分であると考えられる。特に，ユーザ視点の変化に自動的に追従して分身視点が変化する連動機能はユーザのマップ操作の負担を大幅に軽減する。また，どのような移動目的のときにどのような視点連動法が最適であるかが明らかになっていない現時点ではユーザが自由に選択できることが望ましい。

ユーザとアバタとの連動は一意に決まるが，連動を実行・表示する時期によって逐次更新連動と一括更新連動に分けることができる。逐次更新連動では常にユーザとアバタを同じ状態に保つのに対し，一括更新連動はユーザ（アバタ）の移動中はアバタ（ユーザ）を固定しておき，移動が終了した時点でアバタ（ユーザ）をまとめて移動させる。一括更新連動では画面が変化するのはどちらかひとつだけであるのでユーザは注視できることと，計算負荷が小さくて済むという利点がある。一括更新は連続アニメーションと瞬時移動に分けることができる。

5．VRML による実装

NINJA システムは　WWW ブラウザとして Microsoft 社の Internet Explorer を，VRML ブラウザとして Parallel Graphics 社の Cortona VRML Client[1]を対象に開発した。これら両ブラウザを選んだ理由はその普及度，性能，ならびに将来性による。

5．1　基本的考え方

3Dマップインターフェイスを実装するときにまず考えられる方法は図7(a)に示すように VRML ブラウザ自体に3Dマップ機能を持たせることである。これは実装が簡単で高速処理が実現できるが，VRML ブラウザの汎用性が低下するだけでなく3Dマップを必要としない多くの応用で無駄になる。一方，図7(b)のように3Dマップを仮想世界のオブジェクトと同じシーングラフ内にひとつのインターフェイスオブジェクトとして実装する方法が提案されている[7]。この方法は仮想世界とインターフェイスが混在してしまうこと，新しい VRML ノードの導入や専用グラフィックボードと専用グラフィックソフトが必要になることなど，3Dマップの実装が難しくなり，また標準的な家庭用パソコンでは利用できない。

そこで本稿では，図8に示すような，新しいアーキテクチャでの実装を提案する。すなわち，仮想世界と3Dマップをひとつのシーングラフとするのではなく二つに分離し，それぞれに連動に必要なオブジェクトだけを追加する。具体的には，ひとつの HTML 文書

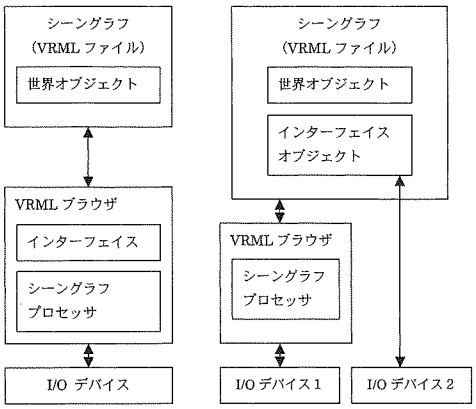

図7　従来のインターフェイス実装法

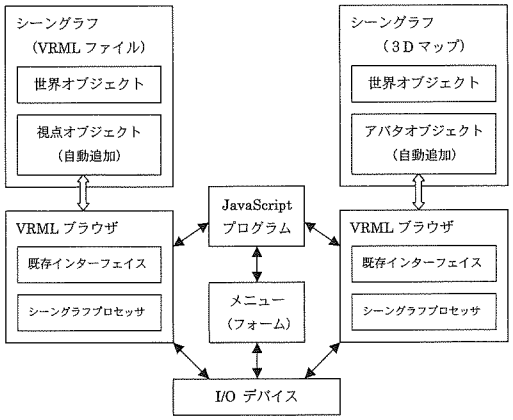

図8　提案するインターフェイス実装法

に二つの既存 VRML ブラウザを埋め込み，一方をユーザ視点画像表示専用に，もう一方を 3D マップ表示専用にし，両者の連携を外部アプリケーションがとるようにする。両者の連携をとるためには，それぞれのシーングラフに視点オブジェクトやアバタオブジェクトなどいくつかの VRML ノードを追加する必要がある。この初期化処理も仮想世界のファイルをロードした後にクライアント側で外部アプリケーションが実行する。

5．2　提案アーキテクチャの利点

第1に，提案アーキテクチャは VRML ブラウザ自体には手を加えないので，VRML ブラウザの汎用性とコンパクトさを損なわずに 3D マップを実現できる。第2に，3D マップをテクスチャとしてオブジェクト表面に貼り付けたりしないので，専用のグラフィックカードやグラフィックソフトを必要とせず，実装も簡単である。第3に，ユーザ視点画像と 3D マップを完全に分離表示するので，ユーザ視点画像が 3D マップに邪魔されることがない。従来はユーザが 3D マップを画面の邪魔にならないところに移動させたり，一時的に隠したりするといった余分な手間がかかっていた[7]。第4に，ナビゲーション対象である仮想世界の VRML ファイルに一切手を加えずに 3D マップを実現できることである。コンテンツクリエーターは 3D マップを意識することなく制作に専念でき，保守管理も効率的になる。第5に，HTML 文書に VRML ブラウザを埋め込むため HTML 自体の利点をフルに活用できる。例えば，インターフェイスの構成要素として各種のテキストベースメニューや画像などを HTML 文書に埋め込み，それと VRML ブラウザとを簡単に連携できる。システム構築にあたってテキスト情報の表示と操作は不可欠であり，HMD を使ったシステムの大きな問題点のひとつになっている[7]。

これまで VRML と外部アプリケーションとのインターフェイスとしては Java EAI きりなかったが[2]，Cortona VRML Client ではその代替として VRML Automation ドライバを提供した[11]。そこで今回の開発では外部スクリプト言語として動作安定性の点から Java アプレットの代わりに JavaScript を採用した。以下ではこのような動作環境のもとでの NINJA システムの VRML による実装法について述べる。

5．3　VRML ブラウザの HTML への埋め込み

NINJA システムでは一つの HTML ドキュメント内に二つの VRML ブラウザを〈OBJECT〉タグを用いて埋め込む。それらのオブジェクト ID を Cortona 1 及び Cortona 2 とし，前者をユーザ視点画像用に，後者を 3D マップ（分身視点画像）用に使用する。オブジェクト ID は〈SCRIPT〉タグ内で記述された JavaScript が二つの VRML ブラウザを識別する際に使用される。Cortona 1 及び Cortona 2 のクラス ID は同一である。Cortona を HTML に埋め込むときに，〈param〉タグによって読み込む VRML ファイル名やナビゲーションバーの有無など27種類のパラメータを指定できる。

JavaScript は VRML ブラウザと通信するときには“Cortona コントロールインターフェイス”を，VRML 世界と通信するときには Cortona コントロールの En-

gine プロパティ経由で“VRML Automation ドライバ”を使用する。これら２種類のインターフェイスを使用して JavaScript は埋め込まれた二つの VRML 世界同士の連携をとることができる。また，ボタンやチェックボックスなどのフォーム要素から VRML 世界にアクセスすることも可能になる。

５．４　コンテンツと３Ｄマップの初期化

コンテンツ（ナビゲーションの対象となる仮想世界を記述した VRML ファイル）がユーザのコンピュータにロードされ，HTML に Cortona1 及び Cortona2 として埋め込まれた時点で，HTML 内の JavaScript プログラムは NINJA システムとしての機能を実現するための初期化処理を行う。まず，Cortona1 のシーングラフ（ユーザ視点画像として使用するコンテンツ）には次の２種類のノードを追加する必要がある。

① ユーザの位置と向きを連続的に計測する接近センサノード
② ３Ｄマップからユーザの視点制御を行うための視点ノード

これらには JavaScript からアクセスできるように DEF によって名前をつけておく必要がある。一方，Cortona２のシーングラフ（３Ｄマップとして利用するコンテンツでユーザ視点画像に使用するものと同じもの）には次の４種類のオブジェクトを追加する必要がある。

① 平面センサのついたアバタの頭部オブジェクト
② 円筒センサのついたアバタの鼻オブジェクト
③ アバタ付属品(視線ビーム，アドバルーン，GPS)
④ 分身視点のための視点ノード

これらのオブジェクトは複数の VRML ノードからなり，いくつかのノードには JavaScript からアクセスできるように DEF によって名前が付けられている。

以上の初期化処理を行うにあたって注意すべきことは，VRML ファイルの読み込みが完了するのを待たないと JavaScript エラーが発生する点である。次のような〈SCRIPT〉タグを使った記述を HTML 内に置くと，VRML シーンの読み込みが成功したときに「記述１」が，失敗したときに「記述２」が実行される。この VRML ブラウザイベントハンドラを利用して初期化処理を開始する。

```
<SCRIPT for = “Cortona1” event = “OnSceneLoaded
        (Success)” language = “JavaScript”>
    if (Success){  …  記述１  …  }
    else{  …  記述２  …  }
</SCRIPT>
```

読み込んだ VRML ファイルに新たな VRML ノードを追加するには Engine オブジェクトの CreateVrmlFromString メソッドと RootNodes オブジェクトの Add メソッドを使用する。今，Cortona1 のシーングラフに世界全体をカバーする接近センサを ProSen という名前で追加するには次のようにする。

```
engine1 = document. Cortona1. Engine;
vrml1 = emgin1. CreateVrmlFromString
(“DEF ProSen ProximitySensor {size 999 999 999}”);
enginel. RootNodes. Add (vrml1);
```

視点ノードやアバタの追加も同様にして行うことができる。

仮想世界を JavaScript から監視制御するには，仮想世界を構成する VRML ノードのフィールドへの参照を取得する必要がある。例えば上述した Cortonal の接近センサ ProSen のイベント出力 position_changed への参照は次のようにして取得できる。

```
ProSenPosi = Cortonal. Engine. Nodes (‘ProSen’).
             Fields (‘position_changed’);
```

ユーザの位置の x，y，z 座標はそれぞれ ProSenPosi.x，ProSenPosi.y，ProSenPosi.z となる。

５．５　アバタによるユーザ視点情報表示

ユーザの移動に対して３Ｄマップのアバタをリアルタイムで連動させるためには，仮想世界に追加した接近センサのイベント出力を用いてアバタの位置と向きを制御する必要がある。これを実現する方法として，一定時間毎に接近センサの出力を読みとり，それを用いてアバタの状態を更新することが考えられる。これはシンプルで実装が簡単であるが無駄な処理が多い。なぜならば，常にユーザが移動しているとは限らないからである。接近センサの出力が変化したときだけアバタを更新するようにするためには，JavaScript が VRML イベントを検知できる必要がある。そのために Advise メソッドが準備されており，イベント出力が変化したときに実行する関数を定義できるようになっている。例えば，Cortona1 の接近センサ ProSen のイベント出力 position_changed にイベントハンドラ updateAvatarPosi をつけるには次のようになる。

```
<SCRIPT for = “Cortona1” event = “OnMouseEnter
        ()” language = “javascript”>
    ProSenPosi. Advise (updateAvatarPosi);
```

```
</SCRIPT>
```

ここで，マウスポインタが Cortona1 に入ったというブラウザイベント OnMouseEnter が発生したときに接近センサにイベントハンドラを付けているのは，ユーザが VRML ブラウザの Walk などのナビゲーションツールを使用しているときだけ行えばよいからである。

関数 updateAvatarPosi のなかでは，次のようにアバタ頭部の位置を更新する処理を記述する。

```
function updateAvatarPosi( ) {
  HEADPosi. x = ProSenPosi. x ;
   . . . . . .
}
```

ユーザの向きとアバタの向きを連動させる方法もここで述べた位置の連動方法とほぼ同様に実現できる。

5．6　アバタによるユーザ視点制御

アバタ頭部のドラッグによってユーザ視点の位置を制御するためには，アバタ頭部につけた PlaneSensor ノードの translation_changed イベント出力にイベントハンドラを付け，仮想世界のアクティブな Viewpoint ノードの position フィールドをリアルタイムに更新すればよい。ここで注意すべき実装上の問題点が三つある。以下ではその問題点と解決策について説明する。

(1)　視点の遊離

ファイルのなかに複数の視点ノードが記述されている場合，最初に現れる視点ノード VP 1 がアクティブになり，それがユーザの初期視点となる。この現在アクティブな視点をプログラム上の視点と呼ぶことにする。ユーザが Walk などによって移動すると，このプログラム上の視点 VP 1 とユーザの視点とが遊離した状態になる。このような状態でアバタをドラッグして VP 1 の position フィールドを変更しても，アバタの表示位置とユーザの位置の間にずれが生じてしまう。これは JavaScript によってアクティブな VP 1 のフィールド値を変更しても先に発生した遊離状態を維持したままユーザ視点が変化するためである。しかも JavaScript からユーザの視点を直接制御する手段は提供されていない。

そこで，もう一つの視点ノード VP 2 を準備しておく。VP 1 がアクティブなときには VP 2 は非アクティブなので VP 2 を現在のユーザ視点と一致させることは可能である。一致させたあとに VP 2 を JavaScript からアクティブにしてやると，プログラム上の視点 VP 2 とユーザの視点は完全に一致する。アバタをドラッグしたときに今度はこの VP 2 を更新してやればよい。このように VP 1 と VP 2 を交互にプログラム上の視点として利用することにより，ユーザの視点とアバタの表示位置を常に正しい状態に保つことが可能となる。

(2)　アバタの過剰移動

アバタをマウスでドラッグしたとき，平面センサの出力を用いてアバタ自体を移動させるとともに，対応する視点ノードのフィールド値を変更してユーザ視点も移動させる必要がある。ユーザ視点が変化するとそれを接近センサが検知し，そのイベントハンドラがアバタを再度移動させる。こうしてアバタは実際にドラッグした以上に移動してしまう。ゆえに，3D マップ上でアバタをドラッグするときには，すなわちマウスポインタが 3D マップ内に入ったときには次のようにして接近センサにつけたイベントハンドラを切断しておく必要がある。

```
ProSenPosi. Unadvise (updateAvatarPosi) ;
```

逆に，ユーザ視点画像内にマウスポインタが入ったときには切断したこのイベントハンドラを付け直す必要がある。

(3)　平面センサのオフセットのずれ

平面センサのオートオフセット機能を ON にしておくと，アバタをドラッグして XZ 平面内を移動させると，アバタの元の座標値（オフセット）にドラッグした量が加算されて，それがアバタの新しい座標値（新しいオフセット）になる。しかし，アバタをドラッグする前に Walk などでユーザが移動していた場合には，接近センサのイベントハンドラがアバタを直接移動させるので，平面センサのオフセットの値が実際とずれてしまう。ゆえに，3D マップ上でアバタをドラッグするときには，すなわちマウスポインタが 3D マップ内に入ったときには，まず平面センサのオフセットを現在のユーザ視点と同じように修正する必要がある。

以上述べてきた NINJA システムにおけるソフトウェアの基本構成と処理をまとめると図 9 のようになる。

6．関連研究

仮想 3 次元空間ナビゲーションに関する研究は，①小型モニタとマウスの使用を前提としたデスクトップ仮想環境のための移動インターフェイスと，②HMD や専用入出力装置の使用を前提とした没入型仮想環境

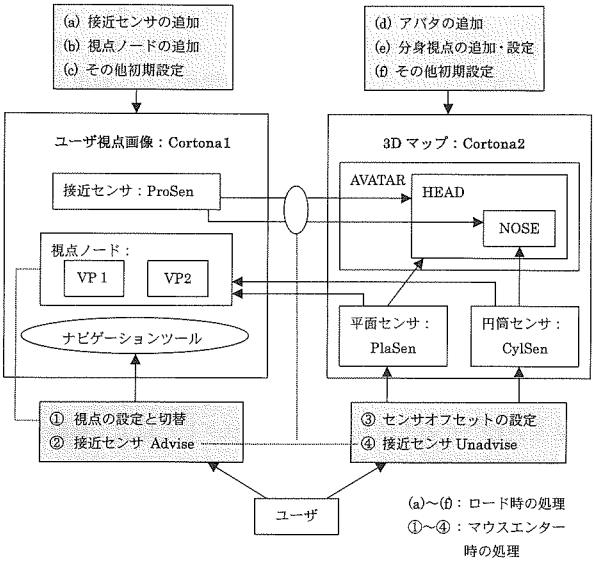

図 9　ソフトウェアの基本構成と処理

のための移動インターフェイスに大別できる。従来研究の大部分は後者に属するが，そのアイディアのなかには前者でも利用可能なものもある。一方，ナビゲーションに用いる情報の種類によって従来研究を分類すると，(1)ユーザ視点画像のみを用いて移動する研究，(2)仮想世界に関する文字情報を利用して移動する研究，(3)仮想世界の 2D マップを利用して移動する研究，(4)仮想世界の 3D マップを利用して移動する研究，の四つが考えられる。2D マップと 3D マップの研究には共通する点も多いが，以下では本研究と直接的に関連のある 3D マップを利用した従来研究について述べる。

3 次元仮想環境における視点制御のメタファとしては“eyeball in hand”，“scene in hand”，“flying vehicle control”の三つが有名である[3]。eyeball-in-hand は仮想環境を固定し，手に持った 6 自由度トラッカの移動・回転によってユーザ視点を移動・回転させる。scene-in-hand はユーザ視点を固定し，手に持った 6 自由度トッラカの移動・回転によって仮想世界自体を移動・回転させる。flying-vehicle-control は仮想環境を固定し，手に持った 6 自由度トラッカの移動・回転によってユーザの乗った仮想ビーグルの移動速度と回転速度を制御する。Ware らはこれら三つを実装し，種々のタスクに対する性能を比較評価した。そして，すべてのタスクに対して優れたメタファはなく，タスクに応じて使い分ける必要があると結論づけている。

仮想世界の 3D マップをナビゲーションへ利用する研究としては WIM（World In Miniature）が有名である[4]。これは仮想世界のミニチュアを仮想世界の一つのオブジェクトとして仮想世界の中に配置し，ユーザがこのミニチュアを 3D マップとして携帯しながら仮想世界をナビゲーションするものである。WIM のなかにはユーザの移動に連動するアバタが存在し，アバタをドラッグして移動することもできる。ユーザは 6 自由度トラッカ付の HMD を装着し，左手に 6 自由度トラッカ付のクリップボード，右手にボタンと 6 自由度トラッカ付のボールを持つ。HMD は仮想環境と WIM を表示する。クリップボードの面は WIM のベース平面と対応し，これを動かすことによって WIM をいろいろな視点から眺める。ボールは視線キャスティングによるオブジェクト選択に用いる。ゆえに WIM の操作に scene-in-hand 法を，オブジェクトの選択に eyeball-in-hand 法を応用している。

WIM ではユーザ視点画像のなかに 3D マップを表示していたが，ユーザ視点画像と 3D マップを連続的に切り替える方法が提案されている[5]。すなわち，ユーザが WIM 内のアバタを移動させて手を放すと，ユーザは WIM に入り込みアバタになってフルスケールの仮想環境内を移動し，移動が完了するとフルスケールの仮想環境は消えて新たな WIM が表示される。視点移動アニメーションをするときのユニークな実現法といえる。

VRML を用いているという意味で本研究に近い研究は，VRML ブラウザを HMD 対応にするためにナビゲーションインターフェイスを仮想環境のオブジェクトとして実現した MaPS である[7]。3D マップを仮想環境内に表示する点は WIM と同じであるが，ミニチュアをオブジェクトとして仮想世界に置いてはいない。その代わりに二つの視点を仮想世界に置き，ひとつをユーザ視点として画面全体に表示し，もうひとつを鳥瞰カメラとして仮想環境内のオブジェクト（マップツール）の表面にテクスチャ画像として貼り付けている。これを実装するために 3D グラフィックス専用のハードウェアとソフトウェアの機能を使用しており，VRML の標準の Script ノードでは実現できない。入力デバイスは 6 自由度トラッカ，圧力センサ，及びボタンのついた特殊なものであり，VRML の Script ノードが入力データを直接取り込む。

3D マップとして使用する鳥瞰カメラの制御をより容易にするための方式が提案されている[6]。このシステムではユーザ視点の制御はユーザの頭の向きと移動動作により行う。鳥瞰カメラの制御には WIM と同様に scene-in-hand 法を利用しているが，それ単独ではなくユーザの頭の動きにも連動させている。鳥瞰画像内にアバタは表示されるが，アバタをドラッグすることによって移動することはできない。このシステムも HMD と 2 個の 6 自由度トラッカを使用している。仮想都市においてランドマークを巡る実験により，提案手法と eyeball-in-hand 法及び scene-in-hand 法とを比較考察している。

7．あとがき

本研究では仮想空間内のユーザと 3D マップ内のアバタが双方向に連動するナビゲーションインターフェイスを完全に Web 対応化し，広く一般ユーザへの門戸を開いた。また，ユーザ視点画像の見やすさ向上のために 3D マップの分離表示を実現するとともに，3D マップの操作性を向上するためにユーザ視点とマップ表示のマルチ連動を提案・実現した。さらに，コンテンツ開発の効率化と 3D マップベースインターフェイスのポータビリティ改善のためにコンテンツとインターフェイスを分離実装する方法を確立した。本研究により一般ユーザ向けの 3 次元マップを用いた仮想環境が Web 上で構築可能となり，経路計画から移動までのナビゲーション全般にわたってユーザの意思決定と移動操作を簡単化・迅速化できるようになった。

NINJA システムに関して残された課題は，インターフェイスの部品化，実験的システム評価などである。3D マップ全般に関する本質的な重要課題として，仮想世界のオブジェクトに関する文字情報をいかに効率的に記述し，マップ上に見やすい形で重畳表示するかという問題がある。また，建物内部など内部を鳥瞰できないような仮想環境に対する 3D マップの実現法も重要である。2D マップの併用はこの問題に対するひとつの解決策であるが，3D マップは自動生成できるという 2D マップにはない利点を何とか生かせないだろうか。

従来研究では 3D マップを第 2 の視点，神様の視点，鳥瞰カメラ視点などと呼び，フルスケールの仮想世界をユーザ視点画像としていた。本研究でも 3D マップを分身視点画像と考え，3D マップに自分の化身であるアバタを登場させた。しかし，NINJA システムではユーザと分身を入れ替えて考えることもできる。つまり，ユーザ自身が仮想世界を全体的に見晴らすところに位置し，仮想世界を実際に動き回るのは分身にやらせ，詳細な画像情報をリアルタイムでユーザに送信させる。分身がユーザのアバタを見るのではなく，ユーザが分身のアバタを監視し，マウスで自由に移動させる。このように考えるほうがユーザ自身に主導権がありなじみやすいかもしれない。

参 考 文 献

[1] Cortona VRML Client: http://www.Parallelgraphics.com/products/cortona

[2] Web 3D コンソーシアム：http://www.web3d.org/

[3] Ware, C. and Osborne, S.：“Exploration and virtual camera control in virtual three dimensional environments”, Computer Graphics, ACM, vol.24, no.2, pp.175-183 (1990).

[4] Stoakley, R., Conway, M., and Pausch, R.：“Virtual reality on a WIM: Interactive worlds in miniature,” Proc. of CHI ’95, ACM, pp.265-272 (1995).

[5] Pausch, R. and Burnette, T.：“Navigation and locomotion in virtual worlds via flight into hand-held miniatures”, Proc. of SIGGRAPH ’95, ACM, pp.399-400 (1995).

[6] 深津，北村，正城，岸野：座標系対連動法による仮想環境内ナビゲーションのための鳥瞰カメラ視

点の直感的制御，信学論，Vol. J 83-D-II，No. 9，pp. 1905-1915（2000）.

[7] Edwards, J.D.M. and Hand, C.：“MaPS:movement and planning support for navigation in an immersive VRML browser”, Proc. of VRML '97, pp.65-73 (1997).

[8] Tae-Wook Kwon and Yoon-Chul Choy： “A new navigation method in 3 D VE (2 D map-based navigation)”, 6th International Conference on Virtual Systems and MultiMedia (VSMM 2000), pp.393-402 (2000).

[9] 水谷清美，高橋友一：インターネット上の3次元ナビゲーションにおけるインターフェイスの検討，信学論，Vol. J 81-D-Ⅱ，No. 5，pp. 925-932，1998年5月。

[10] Alex van Ballegooij and Anton Eliens: Navigation by query in virtual worlds, Web 3 D 2001 (2001).

[11] Parallel Graphics Cortona Software Developers Kit: http://www.parallelgraphics.com/products/sdk/