VR Juggler による 3 次元可視化

目 野　大 輔

神戸大学 工学部

陰 山　　聡

神戸大学　システム情報学研究科

1. はじめに

スーパーコンピュータを用いた大規模な 3 次元シミュレーションが日常的に行われるようになった今日、その出力数値データを効率的に可視化し、解析するための 3 次元な可視化手法を確立することは急務となっている。家庭用 3 次元 TV が急速に普及しつつある現状を考えれば、人間のもつ優れた 3 次元空間把握能力を積極的に活用した新しい 3 次元的データ可視化手法が、今後大きく発展することが期待できる。

しかしながら、家庭用 3 次元 TV のように、3 次元物体を単に両眼立体視を通じて受動的に見るだけでは、人間の持つ空間把握能力を十分に生かしているとは言えない。ルービックキューブで遊ぶとき、あるいは美術館で彫刻を鑑賞するときを想像すればわかるように、人間がある程度複雑な物体の立体構造を把握しようとする時には、その物体を単に立体視するだけでは不十分であり、その物体を手で回しながら、ためつすがめつ眺めたり、あるいは自分の足でその物体の周囲を歩きまわって眺め回すのが普通である。将来「3 次元美術番組」が登場し（既にあるかもしれないが)、そこで彫刻の名作が放送され、それを大画面の 3 次元 TV で見たとしても、その作品の良さを十分には味わえないであろう。洗練された 3 次元シミュレーションによって得られた膨大な数値データに隠された 3 次元構造を可視化という操作を通じて抽出しようとするときにも同様である。単なる立体視だけでなく、そのデータ空間の中を自由に歩きまわり、ためつすがめつ観察することで初めて人間の 3 次元把握能力をフルに活かした真の 3 次元可視化が可能となる。

バーチャルリアリティ（VR）はそのような、真の 3 次元可視化を可能にする技術である。バーチャルリアリティの基本要素として、慶応大学の舘教授は、等身大 3 次元空間、実時間相互作用、自己投射の 3 つを挙げている [1]。シミュレーションのデータをテレビのような小さな 2 次元面ではなく、自分よりも大きな 3 次元空間に表現し（等身大 3 次元空間)、そのデータから数学的に構成した可視化物体をあたかも実在のものかのように触り、移動させ、生成・消滅させ（実時間相互作用)、そのデータ空間にあたかも自分が入り込んでいるかのように感じる（自己投射）ことができれば、理想的な 3 次元データ可視化環境が構成されることは容易に想像できるであろう。

VR 技術を活用したそのような 3 次元可視化手法を、我々は約 10 年前から始め[2,3]、それ以来、自分たちのシミュレーション研究に活用してきた。その主な対象は、プラズマ物理や核融合、あるいは地球科学を中心とした流体系の 3 次元シミュレーションデータである。自分たちのシミュレーションデータを可視化するために必要な機能を自分たちの手で実装してきたので、そのデータ解析に役に立つソフトウェアができているのは当然と言えるかもしれないが、我々は常に自

分たちのソフトウェアの汎用性と応用性を意識しながら開発を続けてきた。そしてその手法やツールを VFIVE と名付けた可視化フレームワークに集約した[4-8]。長年の開発の結果、VFIVE は一般的な 3 次元の場のデータ、特に複雑な構造を持つベクトル場を同時に複数解析する必要のある場合に極めて効果的な VR 可視化ソフトウェアになったと考えている。当然のことながら、我々自身がこの VFIVE のヘビーユーザである。我々の専門である地球シミュレータを使った大規模な地磁気シミュレーションをこの VFIVE を駆使して VR 空間の中に「没入して」解析した結果、シミュレーションデータの電流構造に新しい 3 次元構造を発見することにも成功している [9,10]。

この VFIVE を中心とし、我々が開発してきた VR 可視化ソフトウェアは、全て CAVE 型と呼ばれる方式のバーチャルリアリティ装置用に作っている。CAVE は、イリノイ大学シカゴ校において開発されたバーチャルリティ装置で、一辺が 3m 程度の大きな正方形のスクリーンを複数枚組み合わせて、小さな部屋の床面と壁面に 3 次元画像を投映するものである [11]。むろん VR 装置なので、単なる 3 次元ディスプレイではなく、トラッキング技術によって目の位置と傾き、視線の方向等をリアルタイムに検出している。部屋の中を自由に歩き回り、目の前の仮想 3 次元物体を眺め回すことができる。（十分な解像度を備えた CAVE で見れば彫刻作品も鑑賞できるであろう。）

CAVE 型の VR 装置は、当初 3 面あるいは 4 面のスクリーンをもつ立方体形状であったが、その後、様々なバリエーションが出た。直方体形状の CAVE 装置もあれば、UC San Diego の StarCAVE [12] のように、床面が 6 角形で、壁面が垂直でないもある。さらには、スクリーンが 1 面しかない CAVE 装置もある。（本研究ではこの 1 面型 CAVE を用いた。）

スクリーンの形状は様々であるが、これらの CAVE 装置の特徴の一つとして、VR 空間を構成するための基本 API として CAVElib を使っているという点が挙げられる。CAVElib はもともとイリノイ大学シカゴ校で開発されたものであるが、現在では市販されている。CAVElib は、スクリーン間の画像の接続、立体視（液晶眼鏡とステレオ画像の同期）、トラッキングデータに基づいた射影計算（視点設定）の自動更新などを行うだけでなく、ワンドと呼ばれるコントローラを通じたバーチャルリアリティ空間との実時間相互作用を可能とするための関数群を提供している。

CAVElib は、現在 VR における業界標準的 API になっている優れた API であるが、有償であるため、我々のように自分たちが開発した VR 可視化ソフトを（無償で）世界的に普及させたいと考えている場合には、この点が障害となる。我々が開発した VFIVE は、現在、海洋研究開発機構のウェブサイトを通じて無償でダウンロードできるようにしているが、その基本 API として OpenGL と CAVElib を使っているために、CAVElib を使える環境でしか VFIVE を利用できないのは大変残念なことだと考えていた。

ところが、最近になっていくつかの無償 VR 用 API の開発と公開が進んできたために状況が変わってきた。そのなかでも我々が注目したのは VR Juggler と呼ばれる API である。これは CAVE のオリジナルの論文 [11] の筆頭著者である Cruz-Neira 教授が中心となって開発しているもので、LGPL ライセンスで無償提供され、現在 ver. 3.0 まで開発が進んでいる [13]。

VR Juggler は、開発者が個々の VR ハードウェアデバイスを意識しないで汎用的なプログラムを容易に書けるよう virtual platform (VP) という考え方を中心においている。VP とは、VR アプリケーションが、直接ハードウェアと通信することなく、kernerl とのみやりとりするというソフトウェアのデザインである。その結果、ユーザはステレオスクリーンやインプットデバイスなど、使用している VR のハードウェアを直接意識する必要がない。実際、アプリケーション

が動いている途中にスクリーンの一部を取り外したり、逆に面数を増やしたりすることも可能である。このような自由さは CAVElib にはない。

VR Juggler はこのように、従来の VR 用 API を大幅に改良したものであるということが最大の特徴であるが、本報告では、既存の VR 装置を CAVElib を使わずに利用するための無償の API としての面にのみ注目する。我々は将来、VFIVE を一から作り直すことを考えており、その際、バーチャルリアリティ用の API として、CAVElib に代わる無償の API を採用するつもりである。調査した結果、VR Juggler がその最有力候補と判断したので、これを我々の 1 面型 CAVE 装置にインストールし、簡単なバーチャルリアリティ可視化プログラムを作って API としての機能を試すことにした。

日本国内でも VR Juggler に対する関心は高まりつつあるようであるが、インストールが比較的難しく、また、参考となるような情報も少ないため、国内で VR Juggler の使用に成功した例はまだないようである。このような現状を鑑み、本報告において VR Juggler のインストールに必要となる情報をまとめ、あわせて、簡単な応用プログラムの開発について報告する。我々が調べた限り、VR Juggler のインストールについては、国外に英国の Numerical Algorithms Group によるレポート [14] があるだけのようである。

表 1 に今回の報告で我々が用いた 1 面型 CAVE 装置の仕様をまとめる。トラッキング情報は一度別の PC に入力され、Trackd の通信によって adt08 へと渡される。

スクリーン	3048 mm x 2441 mm
OpenGL	OpenGL version string: 1.4 (2.1 NVIDIA-1.6.24)
計算機	SGI Asterism adt08
CPU	AMD Opteron 2350 (2.0GHz 4 コア) x 2
メモリ	64GB
GPU	NVIDIA Quadro FX 4600
OS	SUSE Enterprise 10.2
トラッキング	Intersense IS900, Trackd

表 1　本報告で用いた 1 面型 CAVE 装置

2. VR Juggler のインストール

VR Juggler の公式サイト[15] からダウンロードできるのはバージョン 2.2 までである。今後は Google Code [16] に新バージョンが掲載されることになる。最新バージョンは 3.0.0-1 であるが、ここでは Google Code からダウンロードした VR Juggler-3.0.0-0 を例にとって説明する。

VR Juggler 本体をインストールする前に、依存ソフトウェア群（表 2）をインストールする必要がある。JDK と Doozer はマニュアルでは Optional と表記されているが、これらも入れておくほうが良い。JDK はコンフィグファイル記述用 GUI ツール vrjconfig のビルド、Doozer は

サンプルプログラムや自作のプログラムのビルドに必要だからである。

CppDom	C++用の Dom 解析ソフトウェア
SCons	ソフトウェアビルドツール
GMTL	汎用数学テンプレートライブラリ
Boost	C++のライブラリ
Flagpoll	ソフトウェア依存関係解決ツール
Python	devel が必要
Doozer	ビルド関係のツール
JDK	Java Development Kit

表２　インストールに必要なソフトウェア

パスを通し忘れないようして、付属（または公式サイト上）のマニュアルに従えばインストールは完了する。重要なパスとその例を表３に示す。

LD_LIBRARY_PATH	Boost へのパス。	/usr/local/lib
FLAGPOLL_PATH	FLAGPOLL へのパス。	/usr/local/lib/flagpoll :/usr/local/share/flagpoll
JDK_HOME	JDK へのパス。正しく通っていないと vrjconfig がビルドできない。	/usr/java/jdk1.6.0_21
DZR_BASE_DIR	Doozer へのパス。サンプルプログラムをビルドするのに必要。サンプルのビルドは VR Juggler のインストールとともに行われる。サンプルの Makefile を流用する場合にも必要。	/usr/local/share/Doozer

表３　重要なパス

我々の環境においては、インストール実行中に Boost のバージョンが不明と表示され、インストールが成功しなかった。この問題は、インストール前の configure 時に --with-boost-includes オプションで Boost のインストールされている場所（/usr/local/include）を指定することで回避した。

また、Boost.Filesystem が見つからないと表示されインストールに失敗する現象も発生した。これもインストール前の configure 時に、–with-boost オプションで libboostfilesystem のある場所（/usr/local）を指定することで回避することができた。

インストールの手順を以下にまとめる：

1. 解凍してできた VR Juggler フォルダの最上層で autogen.sh を実行する。
2. ビルド用のフォルダを生成する。mkdir build
3. フォルダ内に移動する。 cd build
4. コンフィグスクリプトを上記のオプションをつけて実行する。../configure.pl
5. ビルドする。gmake build
6. インストールする。gmake install

3. VR Juggler のコンフィギュレーション

インストールが完了していれば vrjconfig コマンドが使えるようになっている。実行すると Java による GUI ツールが立ち上がる。コンフィグファイルの拡張子は jconf である。jconf ファイルの中身は XML 文書だが、vrjconfig を使わずに書き上げるのは不可能に近い。サンプルの jconf ファイルは当環境では /usr/local/share/vrjuggler-3.0.0/data/configFiles にインストールされた。自作の jconf ファイルもそこに配置する。マニュアルを読み、サンプルの jconf ファイルを見比べた後、我々の CAVE 環境の jconf ファイルを作成する過程で重要であると思われた項目と説明、例を以下に挙げる。ここで説明していない設定項目は基本的にデフォルト値をそのまま使用している。

まず、スクリーン等のハードウェア関係の基本的な設定項目について説明する。既に CAVElib を利用している場合には、CAVElib 用のコンフィグファイル（.caverc ファイル）を参考にすると良い。我々の場合、トラッカー原点に対するスクリーンの四隅の座標設定は CAVElib のコンフィグファイルの設定値を参考にした。

今回我々が用いた CAVE 環境は 1 台のグラフィックスワークステーションと 1 枚のスクリーンのみの構成なので、ディスプレイシステムに関しては表 4 のような設定を行った。複数のスクリーンをもつ通常の CAVE 型 VR システムでは言うまでもなくこの部分をそれぞれの値に設定する必要がある。

Display System

Number of Pipes	出力する画面の数。	1
Pipe Identifire	どの画面に出力するか。-1 を指定すると環境変数の値を使用する。	-1

表 4　ハードウェア（GPU）関連の設定

ウィンドウの設定に関しては表 5 にまとめた。

VR アプリケーションを開発する上で不可欠なツールとして「シミュレータ」がある。これはアプリケーションプログラムを書いているときにそのテストのたびに毎回 CAVE 室内に表示・移動してテストする必要がないよう、モニタのウィンドウ上に CAVE 室での表示をシミュレートして表示するものである。キーボードとマウスを使って、視点とコントローラ（ワンド）の移動を模擬する。CAVElib には CAVE Simulator というシミュレータソフトが付属されているが、VR

Juggler にも同様なシミュレータがある。表 6 にそのシミュレータに関係した設定項目をまとめた。

CAVE 型 VR 装置で最も重要な設定はスクリーンの設定である。VR 表示を行う座標系におけるスクリーンの位置やステレオ表示の設定はここで行う。表 7 にそのスクリーンに関係する設定項目をまとめた。

Display Window

origin	ウィンドウを表示する位置。左下が原点。フルスクリーン時無効。	X:0 , Y:10
size	ウィンドウのサイズ。	Width:1400 , Height:1050
Pipe	どの画面（GPU の出力）に出力するか。	0
In Stereo	ステレオ表示するかどうか。	true
Use border	ウィンドウの枠を表示するかどうか。	true
Hide Mouse Pointer	マウスポインタを非表示にするかどうか。	true
Full Screen Window	フルスクリーンで表示するかどうか。	true
Always On Top	ウィンドウを常に最前面に表示するかどうか。	false
Use this Window?	このウィンドウを使うかどうか。	true
Keyboard/Mouse Input Handler	キーボードとマウスの入力元。	Keyboard/Mouse InputHandler0
Allow Mouse Locking	キーを押しながらマウスを操作する際にポインタをウィンドウ外に出さないようにするかどうか。	true

表５　出力する画面の設定

Simulator Viewport

origin	ウィンドウ内のどの位置に表示させるか。左下が原点。	X:0 , Y:0
size	ウィンドウ内でどのくらいの大きさで表示するか。割合で指定。	Width:1.0 , Height:1.0
View	どの目で見た画像を出力するか。	Stereo
User	ユーザの指定。	VRJugglerUser0
Use this Viewport	この画面を表示するかどうか。	false
Vertical Field of View	画角の指定。	80.0

表６　シミュレータの設定

Surface Viewport

origin	ウィンドウ内のどの位置に表示させるか。左下が原点。	X:0 , Y:0
size	ウィンドウ内でどのくらいの大きさで表示するか。割合で指定。	Width:1.0 , Height:1.0
View	どの目で見た画像を出力するか。	Stereo
lower_left_corner	スクリーンの左下の位置をユーザが使用するトラッカーの原点からの位置で指定。単位はメートル。	X:-1.524 , Y:0.0 , Z:0.0
lower_right_corner	スクリーンの右下の位置。	X:1.524 , Y:0.0 , Z:0.0
upper_right_corner	スクリーンの右上の位置。	X:1.524 , Y:2.441 , Z:0.0
upper_left_corner	スクリーンの左上の位置。	X:-1.524 , Y:2.441 , Z:0.0
User	ユーザの指定。	VRJugglerUser0
Use this viewport?	この画面を表示するかどうか。	true
Is Tracked	スクリーンが可動かどうか。	false
Tracker Proxy	可動スクリーンの場合のトラッカーの指定。	None
Auto Corner Update	可動スクリーンの場合、自動的にスクリーンの角の値を更新するかどうか。	No Update

表７　スクリーン等に映し出す際の設定。我々の環境で使用しているヘッドトラッキングの原点はスクリーンの下辺の中点である。

Input Manager

driver_path	入力機器のドライバへのパス。	/usr/local/lib/gadgeteer-2.0.0/drivers
driver	入力機器のドライバ名。	Trackd_drv

表８　入力デバイスの設定

VR Juggler では、インプットデバイスの管理を Gadgeteer と呼ばれるソフトウェアが行っている。標準的な CAVE 装置では、インプットデバイスからの入力信号や、トラッキングデータを Trackd と呼ばれるデーモンを介して受信するのが普通である。Trackd をそのまま使う場合には表 8 のように設定すればよい。Trackd 以外のデバイスドライバを使うことも可能である。

Gadgeteer は Virtual Reality Peripheral Network (VRPN) [17] にも対応しているので、様々な物理デバイスをネットワーク経由で利用することも可能である。各種のトラッキング装置や、力覚デバイス、Wii リモコン等にも対応している。有名なデバイスに関してはほとんどドライバが同梱されている。VRPN を利用するには別途インストールする必要があり、VR Juggler の../configure 時に--with-vrpn オプションでインストール先を指定した後 VR Juggler をビルドする。

最後のユーザ関係の主要な設定を表 9 に示す。

VR Juggler User

Head Position	ヘッドトラッキングの Proxy Alias の指定。	VJHead
User's eye separation	目の間隔。単位はメートル。	0.069

表 9　ユーザの設定

4. 入力デバイスの設定

入力デバイスの設定は、以下の手続きで行う：

1. Input Manager を設定する。（ドライバの存在する場所の指定）
2. Device Driver を設定する。（使用するドライバの設定）
3. Proxy を設定する。（ボタンやスティックの設定）
4. Proxy Alias を設定する。（プログラムで使用するための名前を設定）

設定が完了するとこれらの参照関係が Device Driver を根とする木構造となっているはずである。アプリケーションから使用できるのは 4 で設定する Proxy Alias のみなので必ずここまで設定しなければならない。

我々のシステムでは、入力デバイスとして、デジタルボタン 5 個、ジョイスティックが 1 つのコントローラ（ワンド）を使っている。ワンドの設定例を図 1 に示す。ジョイスティックの入力値は X 値と Y 値を独立に 2 つの入力として設定する必要がある。

図 2 にトラッキングデバイスの設定を、図 3 にはキーボードデバイスの設定を示す。ここでは、エスケープキーでプログラムを終了するように設定している。このような設定をしないままフルスクリーンでプログラムを実行すると、プログラムを終了させる手段がなくなる。

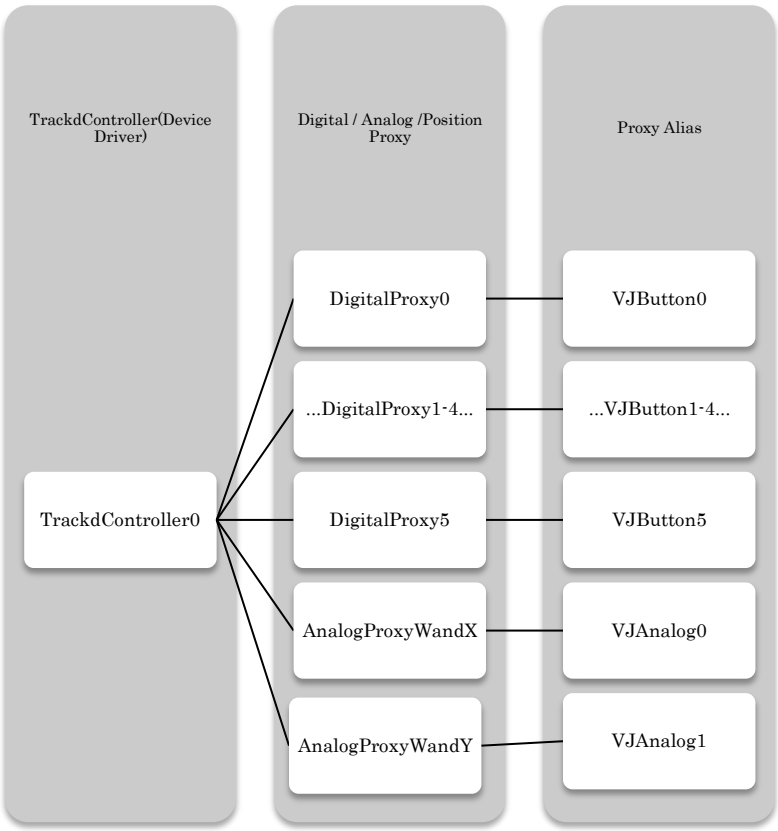

図１　コントローラのボタンやスティックの設定

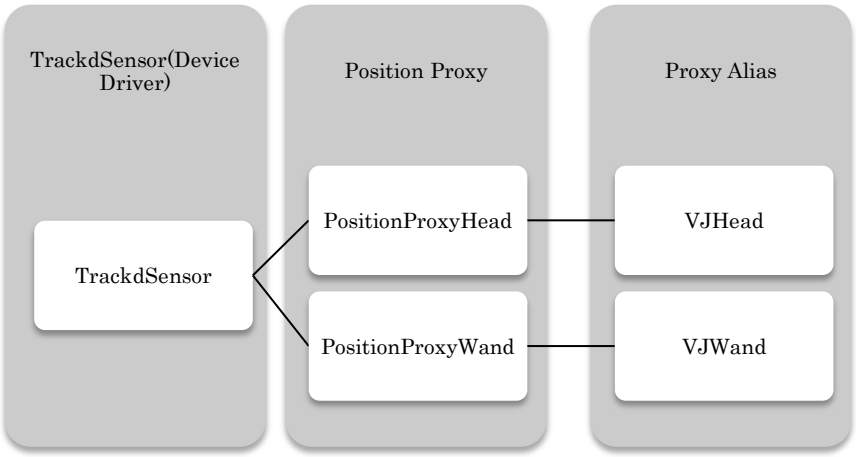

図２　トラッキングデバイスの設定

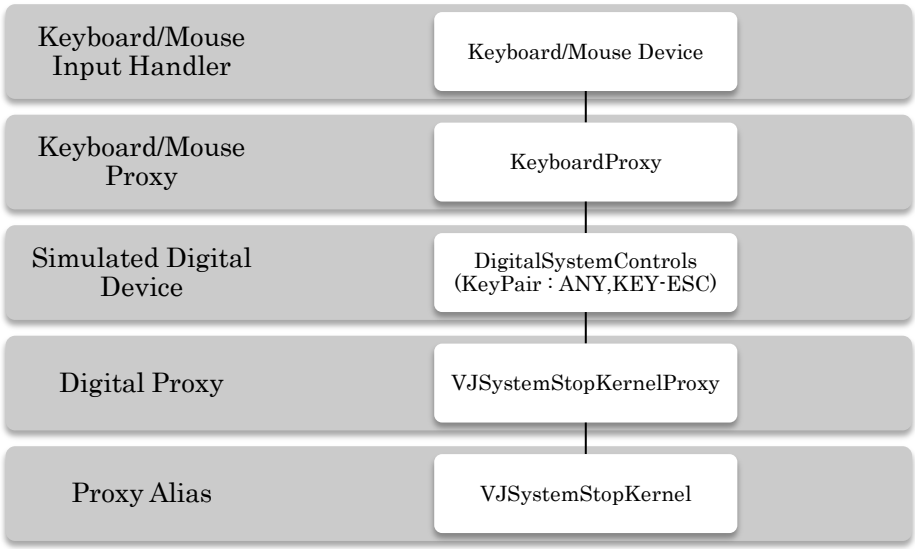

図３　キーボード（ESC キーで終了を有効化）の設定

以上の設定をこなせばそれぞれの環境固有のハードウェアを使用して VR Juggler が動作するはずである。サンプルプログラムもインストールと同時にビルドされているので、それを動かして確認することができる。サンプルプログラムはビルド時に作成したフォルダの中［18］にある。サンプルのソースはまた別の場所 [19] にある。

5. VR Juggler を利用した 3 次元データ可視化ソフトウェア開発の例

VR Juggler を API とした VR 可視化ソフトウェアが実際に 3 次元データの没入的な可視化に活用できることを確認するために、簡単な 3 次元可視化ソフトウェアの開発を行った。この可視化ソフト基本機能は、3 次元点列の座標データを読み取り、それを CAVE の VR 空間に表示するものである。テストとして海洋研究開発機構（JAMSTEC）が観測した地下構造［20］及び地震の発生位置（震源位置）データ（地殻構造探査データベース、機動的観測）[21,22] を用いた。

今回は VR Juggler の機能テストが主目的なので、データ可視化に関してはあまり凝ったことはせず、地下構造と震源の 3 次元分布を OpenGL による色つきの点列で可視化するにとどめた。インプットデバイスの管理には Trackd を使っている（図 4)。

ヘッド及びワンドのトラッキングも問題なく処理された。スクリーン前を歩きまわることで、仮想地下空間に没入し、震源の 3 次元分布を立体的に観察することが出来る。また、このプログラムには、ワンドのジョイスティックの角度データを使って、VR 空間中を平行移動する、いわゆるナビゲーション機能を組み込んでいるが、これも想定通りの結果が得られたので、インプットデバイスの処理は問題なく行われていることが確認できた。

今回のプログラム開発の経験をまとめると、VR プログラムの開発に関しては、CAVElib を用いた場合と VR Juggler を用いた場合の違いはほとんどないと言ってよい。CAVElib を使った VR アプリケーションの開発について我々は、1998 年にプログラミングガイド[23]を書いた。このようなガイドに従って、CAVElib プログラムを開発した経験のある開発者であれば、極めて自然に VR Juggler での開発に移行できるであろう。

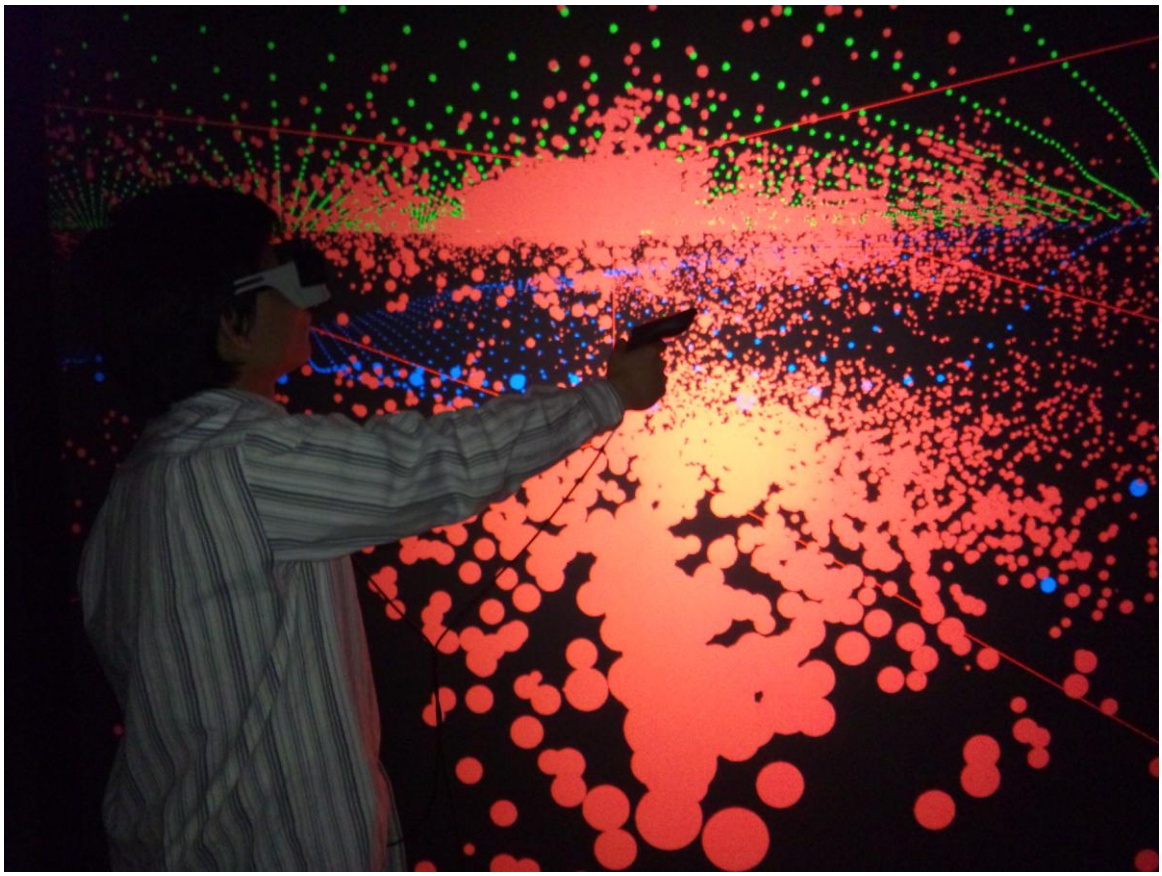

図４　VR Juggler を用いた 3 次元点列データ可視化ソフトウェア

6. おわりに

膨大な 3 次元数値データをいかに可視化処理するかという問題は、スーパーコンピューティングに関わる様々な分野の研究者にとって今後、大きな課題となるであろう。人間の持つ優れた 3 次元空間把握能力をフル活用することを可能とするバーチャルリアリティ（VR）は極めて洗練された技術レベルに達している。我々シミュレーション研究者がその最新技術を有効活用するためには、使い易く、安価で、業界標準的な基盤 API が不可欠である。VR Juggler はその有力候補である。

VR Juggler の最新版（ver.3）の開発については、最近、OpenGL の公式 WEB サイトのヘッドラインニュースでも紹介された。VR Juggler という言葉を目にする機会は今後ますます増えるであろう。本報告が VR Juggler だけでなく、VR 技術を活用した 3 次元可視化に興味を持っていただくきっかけになることを願う。

謝辞

本研究は、（財）ひょうご科学技術協会、（財）山田科学振興財団、（財）大川情報通信基金の助成を受けた。図 4 で示した 3 次元点列データ表示プログラムで用いたデータは海洋研究開発機構の金田義行博士と木戸ゆかり博士から提供されたものである。

参考文献

[1] 舘 暲, 第 18 回 3D&バーチャルリアリティ展、基調講演、2010 年 6 月, 東京

[2] A. Kageyama et al., Proc. ICNSP, 138 (1998)

[3] A. Kageyama, et al., 日本バーチャルリアリティ学会誌, 4, 717-722 (1999)

[4] A. Kageyama et al., Prog. Theor. Phys. Suppl. 138, 665 (2000)

[5] A. Kageyama and N. Ohno, Proc. ISSS-7, Kyoto, 133 (2005)

[6] N. Ohno et al., J. Plasma Phyics 72, 1069 (2006)

[7] N. Ohno and A. Kageyama, Phys. Earth Planet. Inter. 163, 305 (2007)

[8] 陰山 聡, 大野暢亮, プラズマ核融合学会誌, 84, 834-843 (2008)

[9] A. Kageyama et al., Nature, 454, 1106-1109 (2008)

[10] T. Miyagoshi et al., 463, 793-796 (2010)

[11] Cruz-Neira et al., Proc. SIGGRAPH '93, 135-142 (1993)

[12] Thomas A. DeFanti et al., Future Generation Computer Systems, 25, 169-178, (2008)

[13] Cruz-Neira et al., Proc. of the Virtual Reality 2001 Conf. , 89, (2001)

[14] http://www.nag.co.uk/IndustryArticles/vizwall_report.pdf

[15] http://www.vrjuggler.org/

[16] http://code.google.com/p/vrjuggler/

[17] http://www.cs.unc.edu/Research/vrpn/

[18] ..../vrjuggler-3.0.0-0-src/build/modules/vrjuggler/samples

[19] ..../vrjuggler-3.0.0-0-src/modules/vrjuggler/samples

[20] Baba, T. et al., Phys. Earth Planet. Inter., 132, 59-73 (2002)

[21] http://www.jamstec.go.jp/jamstec-j/IFREE_center/data/seismicity_all.html

[22] http://www.jamstec.go.jp/jamstec-j/IFREE_center/data/seismicity_data/2001_Kumano.html

[23] http://www.nifs.ac.jp/report/nifs-memo28.html