汎用 VR システムの活用研究 WG

成果報告書

（WG 活動期間：2018 年 3 月～2020 年 3 月）

2020 年 3 月

サイエンティフィック・システム研究会

汎用 VR システムの活用研究 WG

■商標について

記載されている製品名などの固有名詞は、各社/各機関の商標または登録商標です。

■著作権について

■会社名、商品名等について

会社名、商品名、商標、URL 等は発行時(2020 年 3 月)の情報です。

汎用 VR システムの活用研究 WG　成果報告書　目次

1. まえおき

1.1 本 WG と当報告書について

可視化は、実空間において見えているもののほかに、見えないものを視たり、見えない関係を探ったりすることで、人間に示唆や知見を与える。しかし、視る者の興味を強く引き付けるような画像がふんだんにゲームで使われ、また、可視化といえば計算機シミュレーションの結果を画像にすることを指すという限定的な見方があるなど、可視化はある特定の分野で先行しているのが現状である。

他方、Unity や Unreal Engine に代表されるゲーム用の開発エンジンは、Oculus Rift、HoloLens などの先進的なデバイスを広くサポートすることから VR (Virtual Reality) システム開発のプラットフォームとして注目を集めてきている。さらに HMD (ヘッドマウントディスプレイ)の VR 装置やゲーム用の入力／出力デバイスは､先進的であるにもかかわらず、CAVE 装置と比べ格段に安価であり、大学や研究機関に限らず、広く一般に普及し始めている。

ゲーム業界やシミュレーション研究で先行している可視化技術は､幅広く科学技術･教育分野に活用できる可能性を秘めている。

本 WG は、ゲーム分野で先行している、HMD をはじめとする VR 関連のハードウェアや、Unity、Unreal Engine などの統合開発環境を包含したゲームエンジンを、可視化を中心とした、シミュレーション研究、工学や生物学、医療、防災を含む人間社会科学など、様々な科学技術分野や教育現場などで活用することを目的に活動してきた。

課題として、VR 関連の開発用エンジン・先進的デバイスの関連書籍やインターネット上の情報が､ゲーム開発用の情報に限られており､ゲーム開発以外の分野に適用するための情報が不十分かつ、有益な情報にたどり着くには相当な時間が必要なところにある。

本報告書は、これら同様の課題に直面するであろう多くの研究者の手引きとなることを目的に、視覚を中心とした、VR システムの技術情報、可視化技術の現状及び活用ノウハウ、可視化の新しい活用方法について記載した。

1.2 活動概要

1.2.1　活動期間

2018 年 3 月～2020 年 3 月

1.2.2　WG メンバー

			氏名	期間（2019 年 12 月 20 日まで）
会員	担当幹事		井口　寧	北陸先端科学技術大学院大学
	推進委員	（まとめ役）	大谷　寛明	核融合科学研究所
			大野　暢亮	兵庫県立大学
			河村　拓馬	日本原子力研究機構
			北墓　如法	広島大学
			田村　祐一	甲南大学
			野田　茂穂	理化学研究所
			長谷川　恭子	立命館大学
			宮地　英生	東京都市大学
			安福　健祐	大阪大学
会員外			川原　慎太郎	海洋研究開発機構
賛助会員（富士通）	推進委員	（まとめ役）	畠中　靖浩	富士通
			荒川　拓也	富士通
			大日向　大地	富士通
			小笠　温滋	富士通
			島田　智史	富士通研究所（2019 年 3 月まで）
			寺西　広太郎	富士通デザイン
			七尾　典子	富士通
			西村　直正	富士通（2019 年 3 月まで）
			茂木　厚憲	富士通研究所
			山岡　鉄也	富士通デザイン

1.2.3 活動実績

- 第 1 回会合：2018 年 3 月 27 日
  - ・WG 活動の方向性/活動スケジュールの検討
  - ・委員及びゲストによる可視化研究の紹介　※参考資料
- 第 2 回会合：2018 年 7 月 3 日
  - ・WG 活動の方向性の議論
  - ・委員及びゲストによる可視化研究の紹介
- 第 3 回会合：2018 年 11 月 2 日
  - ・WG 活動の方向性の議論
  - ・委員及びゲストによる可視化研究の紹介
- 第 4 回会合：2019 年 2 月 22 日
  - ・WG 活動の方向性及び成果物の目次の議論
  - ・委員による可視化研究の紹介
- 第 5 回会合：2019 年 4 月 23 日
  - ・成果物の目次の議論
  - ・委員による可視化研究の紹介
- 第 6 回会合：2019 年 7 月 1 日
  - ・成果物の目次の議論
  - ・ゲストによる可視化研究の紹介
- 第 7 回会合：2019 年 10 月 10 日
  - ・成果物の内容の確認
  - ・委員による可視化研究の紹介
- 第 8 回会合：2019 年 12 月 20 日
  - ・成果物の内容の確認

2.　はじめに

2.1 可視化の歴史

2.1.1　リアルタイムレンダリング技術とハードウェアおよび開発環境の変遷

大阪大学　安福健祐

科学技術可視化に必要な要素技術として、特に VR を活用した可視化においては CG のリアルタイムレンダリングが重要な役割を担う。シミュレーション結果を可視化する際に、リアルタイムレンダリングによってユーザーがインタラクティブに視点操作や各種のパラメータの変更が可能となり、より直観的にデータの理解が促進される。ここでは VR を活用した可視化と深く関連する CG のリアルタイムレンダリング技術の歴史を紹介し、それを支えるハードウェア（GPU）の発展からゲームエンジンの普及までの変遷について概観する。

リアルタイムレンダリング技術は CG が誕生して以来、劇的な進歩を遂げてきた。CG はコンピュータが発明された同時期より軍事目的として研究開発がはじまり、1960 年代には CAD プログラムの先駆けとなる Sketchpad システムが開発されている。1970 年代に入ると 3D-CG の研究が本格化し、ユタ大学を中心に 3D-CG レンダリングの基礎理論が確立された。1980 年代に入ると、リアルタイムレンダリングを行うための専用のグラフィックス用ワークステーション（GWS）が市販されるようになる。当時の一般的なコンピュータではレンダリングに必要な幾何計算や照明計算を行うための演算性能が圧倒的に不足しており膨大な処理時間を要していたが、GWS は CPU とは別に「ジオメトリ・エンジン」と呼ばれる 3D-CG 処理専用のハードウェアを搭載することでリアルタイムレンダリングを実現した。ジオメトリ・エンジンはスタンフォード大学で研究開発がはじまり、その開発メンバーによってシリコングラフィックス社が創設される。シリコングラフィックス社といえばリアルタイムレンダリング用 API の OpenGL を開発した中心的存在だが、元々は「ジオメトリ・エンジン」を動作させるためのソフトウェアである IRIS GL というシステムがあり、その仕様をオープンにしたものが OpenGL である。現在ではジオメトリ・エンジンの機能を持つハードウェアのことを GPU と呼んでいるが、当時はまだそういう呼び方はされていなかった。

1990 年代になりリアルタイムレンダリング技術の応用分野が拡大していく。当初は軍事、科学技術分野が中心だったものが、映画やビデオゲームなどの娯楽産業に積極的に利用されるようになった。特に米国ハリウッドを中心とした SF 映画の特殊効果として 3D-CG が導入され、そのコンテンツ制作に GWS が利用された。またビデオゲーム分野においては、ゲームセンターに設置される業務用のゲーム機にジオメトリ・エンジンが搭載され、従来の 2D の画像表現から 3D ポリゴンをリアルタイムレンダリングする 3D ゲームが流行した（SEGA 社の「バーチャファイター」などが有名である）。さらに家庭用のゲーム機（SONY の PlayStation など）にもジオメトリ・エンジンが搭載されるようになり、量産効果による大幅なコストダウンが進んだ。

1990 年代はパーソナルコンピュータ（PC）分野でグラフィックスベースの OS が広く普及しはじめた頃で、グラフィックス処理専用の拡張ボードが販売され始めた。当初は 2 次元画

像の描画を高速に行うものだったが、1995 年にマイクロソフト社が発売し広く普及した Windows 95 からは DirectX と呼ばれるマルチメディア用 API によって 3D-CG のリアルタイムレンダリングを扱えるようになると、その処理を高速化する拡張ボードも登場した。このような状況で NVIDIA 社をはじめとするベンチャー企業は拡張ボード上の廉価なビデオチップを開発しており、1990 年後半になって 3D-CG への取り組みを強化していった。1999 年にリリースされた DirectX のバージョン 7 では、これまで GWS でしか実現出来なかったハードウェアによる幾何計算や照明計算機能が追加された(当時ハードウェア T&L と呼ばれた)。また、この時期から PC 業界でハードウェア T&L に対応したビデオチップのことを GPU（Graphics Processing Unit）と呼びはじめた。コストパフォーマンスの高い GPU を搭載した PC は、高価な GWS に次第に取って替わるようになる。また、当初 PC 用の GPU よりも家庭用ビデオゲーム用のカスタムチップ（SONY の PlayStation2 に搭載された Emotion Engine や Graphic Synthesizer 等）のほうが 3D-CG 処理性能は優れていたが、家庭用のゲーム機は新しいハードウェアが約 5 年おきにしか更新されないのに対し、PC 用の GPU は新製品が毎年登場するため､徐々に GPU の性能が家庭用ゲーム機を追い抜きその差が開いていった。現在では PC 用の GPU に最先端のリアルタイムレンダリング技術が実装されるようになり、最新の家庭用ゲーム機には PC とほぼ同じアーキテクチャの GPU が搭載されている。

GPU は 3D-CG 映像をリアルタイムに生成するための専用回路で構成されている。この専用回路を通して処理される計算過程のことを「グラフィックスパイプライン」と呼ぶ。GPU と呼ばれる前のビデオチップは、その一部の処理を CPU が行っていた。それがハードウェア T&L に対応した GPU になって、グラフィックスパイプラインのすべてを GPU が処理している（図 2.1.1-1 参照)。当初のグラフィックスパイプラインは、GPU のハードウェア仕様で決まられた計算しか出来ず、これを「固定機能パイプライン」と呼ばれた。固定機能では、最初にハードウェア T&L 機能を利用して 3 次元座標データを 2 次元座標に変換する幾何計算や頂点単位の陰影計算が行われる。次に面データを三角形ポリゴン化しピクセルデータに変換する。さらにピクセル単位の陰影処理やテクスチャマッピング等を行い、最終的にフレームバッファの各ピクセルの色が決定される。このような固定機能を使うため、プログラム側は API を介してグラフィックスコマンドやデータを転送する。この API の代表的なものが OpenGL と DirectX であった。

固定機能パイプラインは GPU の性能向上によって数年おきに刷新されていき、より高度な機能が増えれば、それを扱うための API（OpenGL や DirectX）も拡張されていった。新しい API が増える一方、使われなくなる API も出てくるが、互換性を確保するため、古い API に対応するパイプラインも残しておく必要があった。その結果 GPU の回路が複雑化、肥大化していくという問題が生じた。

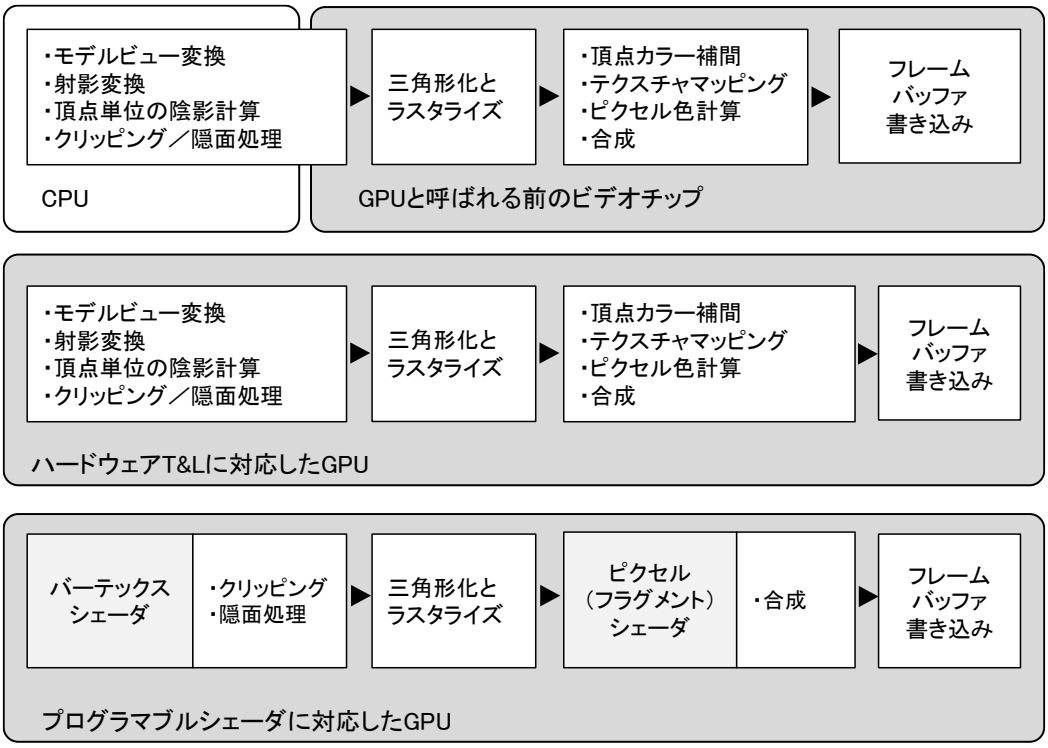


図 2.1.1-1　GPU のグラフィックスパイプラインの変化

これに対して、2001 年に発表された DirectX バージョン 8 と、それに対応した NVIDIA 社の GPU “GeForce 3”には、グラフィックスパイプラインで行う処理を自由にプログラムできる「プログラマブルシェーダ」という仕組みが搭載された。プログラマブルシェーダは汎用性を高めた演算プロセッサであり、グラフィックスパイプラインを固定機能としてではなく、ソフトウェアとして実装することによって新機能に対応し、回路の無駄を避けようとした。プログラマブルシェーダの導入により、座標変換や陰影計算をプログラマーが独自にカスタマイズする機能が提供される。それを実現するため、OpenGL には GLSL と呼ばれるシェーディング言語が追加され、多彩なグラフィックス表現が可能になった。具体的には、これまでハードウェア T&L で頂点単位の処理を行っていた部分を「バーテックスシェーダ」という形にプログラム可能なものにした。もう一つは、ピクセル単位の処理を行っていた部分を「フラグメントシェーダ」という形にプログラム可能なものにした。その後、シェーダにより汎用的な機能が付加されていくと、GPU をグラフィックス以外に応用して並列計算を行う GPU コンピューティングという手法が HPC 分野で普及する。また近年では、GPU の並列計算性能を機械学習に活用される場面も多い。さらに、GPU 本来のグラフィックス計算についても、従来はポリゴンベースのラスタライズ処理を前提としたものであったが、レイトレーシング専用のチップが搭載されるようになった。レイトレーシングによって正確な大域照明計算、物体の反射、屈折、陰影計算を行うにはリアルタイムで不可能な膨大な計算が必要

となるが、実用的にはモンテカルロ的な手法が用いられ、それによって画像に生じるノイズはAIの処理で除去するという方法が提案されている。

次にリアルタイムレンダリングを扱うソフトウェアの開発環境に着目すると、ビデオゲームの場合、その黎明期はタイトルごとにプログラムをフルスクラッチで開発しハードウェアに最適化することによって、PC よりも低い性能のハードウェアでも高品質なグラフィックス表現を可能にしていた。しかし、次第にハードウェアも高性能化し、ビデオゲーム開発の大規模化・複雑化する中で、コストと品質を維持していくため開発の効率化が求められた。特に2000年代以降は、3D-CGを活用したビデオゲームの巨大市場が形成され、リアルタイムレンダリング技術の研究開発が高度化したこと、複数の種類のゲーム機に同時に提供されるマルチプラットフォームのゲームタイトルが増えたことで、開発効率化の流れが加速した。当初は開発言語をアセンブラからC言語、C++に移行することで、ゲームで共通するグラフィックスプログラムやオーディオのプログラムをライブラリ化することからはじまる。特に欧米では、PCゲームを中心にFPS（First Person Shooting）と呼ばれる一人称視点のガンシューティングゲームが大流行し、FPSを効率よく開発するためのフレームワークへと展開した。現在は GUI ベースの統合開発環境に発展してゲームエンジンと呼ばれるようになり、FPS以外のゲーム開発にも応用できるようになっている。最新の PC と家庭用ゲーム機はそのアーキテクチャが同等になっていることから、PC ベースのゲームエンジンを利用して家庭用ゲームの開発が可能となる。ゲームエンジンの種類としては、ゲーム会社が独自に開発しているものから商用化されているもの、オープンソース化されているものまで様々である。その中でも、商用のゲームエンジンであるUnityとUnreal Engineは主な機能を無償で利用できるライセンス体系になったことで、アマチュアからプロのゲーム開発者まで幅広く利用されるに至っている。

ゲームエンジンの主な機能は、3D グラフィックスのデータ変換、リアルタイムレンダリング、アニメーション、衝突判定、物理演算から、オーディオ、AI、スクリプティング等、非常に幅広いが、全てリアルタイムの処理を前提とした設計になっているのが特徴である。また、UnityやUnreal EngineはPCからスマートフォン、タブレット、家庭用ゲーム機に至るまでビデオゲームが動作するほぼ全てのハードウェアに対応しており、ハードウェアごとに必要となるプログラム開発がゲームエンジン側でサポートされ最適化されている。そのような特性を活かし、ビデオゲーム開発以外の用途に利用するケースが増えており、科学技術可視化をはじめ、映像制作のプリビジュアライゼーション、インタラクティブアート、建築ビジュアライゼーション、バーチャルリアリティ（Virtual Reality, VR）によるアミューズメント用途から訓練シミュレータ等のインタラクティブ性を重視するアプリケーション開発で普及していった。特にVRに関しては、2016年以降、低価格で高性能な民生用のヘッドマウントディスプレイ（HMD）が販売され注目を集めたが、そのソフトウェア開発のほとんどはUnityかUnreal Engineが利用されているのが現状である。

2.1.2　科学技術可視化の歴史（一般論）

東京都市大学　宮地英生

1987 年 ACM　Computer Graphics 特別号 “Visualization in Scientific Computing”が出版された。ここには「可視化は見えないものを見えるようにする手法である」と定義され、科学技術の発展に可視化は欠かせないものと位置付けられた。当時、商用の可視化ソフトウェアは存在しなかったが、1990 年に入り数値計算用のワークステーションの普及に伴いデスクトップで利用できる可視化ソフトウェアの流通が始まった。

AVS（Advanced Visualization System）は Stellar 社が開発した無料の可視化ソフトウェアで、その後、AVS（Advanced Visual Systems Inc.）に開発が引き継がれ可視化ソフトウェアとして現在も開発が続いている。グラフィックス用ワークステーションの筆頭であった Silicon Graphics 社からは IRIS Explorer がリリースされ、同様に商用化されたが、現在は開発が停止している。スーパーコンピュータ用の可視化ソフトウェアとして Ensight（CEI 社）が開発され、現在は ANSAYS 社が開発を引き継いでいる。流体解析に特化した可視化ソフトウェアとしては Intelligent Light 社が Field View を開発、現在も開発を続けている。

1990 年半ばにはグラフィックスライブラリの標準が OpenGL または DirectX に固定されたこと、および、一般的な可視化機能の開発がひと段落したことから可視化ソフトウェアの開発が容易になり、可視化機能単独での汎用システムのビジネスが難しくなった。その結果、民間企業では、プリポスト処理、および、設計システムの中に可視化機能が統合されたシステム利用が普及し、研究機関では米国で開発された VTK（Visualization Tool Kit）をベースにした研究や開発が主流となった。

2000 年ころからは、大規模可視化の問題が顕在化してきた。スーパーコンピュータの高速化は CPU 単体の性能向上から並列化に移行した。90 年代後半は数値計算の並列化に応じて可視化も並列化が進んだが、対話処理を要する可視化作業は CPU 効率が低く超並列計算機の利用には適さなかった。その結果、研究者の計算機環境は、計算も可視化も 1 台で行うワークステーション環境から、再び、並列化された大型のサーバマシンを多人数で共有して計算しＰＣクライアントで可視化する分散環境に戻っていく。そこでは可視化の全てのステップで問題が生じた。まず、可視化用の PC クライアントへのデータ転送に尋常でない時間がかかり、ディスクに入れて宅配便で送る方が速い状況が生まれた。次に、たとえ、データを送ることができても PC 用のディスクの読取り速度が遅く、データをメモリに読込むだけで何時間もかかることも問題となった。そして、メモリ容量も CPU 速度も不足した。スーパーコンピュータと CPU のリソース差が広がりすぎたからである。

そのような技術課題に対応するため、米国では前述の VTK をベースに ParaView や VisIt という汎用の可視化ソフトウェアが、遠隔可視化・並列可視化の技術によって HPC システムの計算リソースを使ってスケーラブルに可視化を行うことを目指して開発された。また、それを改良してサーバ・クライアントの協調可視化やディスクベースでの可視化、対話操作を

効率化する自動可視化など、いくつかのアプローチで可視化の負荷軽減が研究された。しかし、スーパーコンピュータの性能向上は続きペタスケールからエクサスケール、さらに大規模化することが予想され、数値計算と同時にスーパーコンピュータ上で可視化を行う In-Situ 可視化が現在の主流となっている。それを前提として、効率的なパラメータ設定や画像より自由度の高い情報を書き出すことで対話的な可視化を残す方法が検討されている。

大規模可視化の課題には、ディスプレイの解像度不足も含まれる。その解決にはタイルディスプレイが使われるようになったが、最終的には人間が眼で見て判断する必要があり、表示できることと人間が認識できることの差異が課題として残った。現在は、可視化の対話処理のループには人間が含まれることから、認知・認識や心理の問題を含めた可視化の効果が研究されてきている。人工知能の適用も始まっている。

大規模可視化以外に、2000 年以降の科学技術可視化における新しいトピックスとして、GPU の性能向上とスーパーコンピューティングへの適用がある。GPU を数値計算のためのアクセラレータとして利用するコンセプト（GPGPU: General Purpose GPU）が広がり、ソフトウェア開発支援のための言語（CUDA 他）が普及し、2020 年現在、GPU はスーパーコンピューティングに不可欠な要素となっている。この動きがさらに GPU の高性能化を牽引し、2010 年代後半にリアルタイムレイトレーシングが可能になった。それを利用するため、Intel 社は自社 CPU を使う OSPRay、NVIDIA 社は自社 GPU を使う OptiX というライブラリを提供している。今後、科学技術可視化でもリアルタイムでフォトリアリスティックな表現が増えていくかもしれない。

2.2 データ可視化をするための現在の開発技術

富士通　小笠温滋

「国語＋百科」辞典の最高峰とされる広辞苑に「可視化」という言葉が初めて収録されたのは第７版（2018）である。その記述を以下に引用する。

可視化：目で見えるようにすること。映像などで示すこと。警察の取調べの～。

このように、可視化という言葉の意味する範囲は広いため、本節が対象とする「データ可視化」の意味する所についてもう少し詳しく定義する。

(1) （狭義の）データ可視化

情報可視化（Information Visualization）とも言う。企業に日々蓄積されるデータを目で見えるような形で整理、分析し、ビジネスに有用な情報を得ることを目的とする。可視化の表現形態としてグラフ（円グラフ、棒グラフ、折れ線グラフなど）を用いることが多く、主要なツールとして Microsoft 社の Excel が用いられる。近年はビッグデータに対応する BI(Business Intelligence)ツールやこれらを統合した ERP(Enterprise Resource Planning)が用いられる。情報可視化については本節の対象外とする。

(2) 思考の可視化

会議やインタビューなどでの会話を図や絵にして目でみえるようにしながら会話を続けていく際に行われる可視化。VF（Visual Facilitation）とも言われ、抽象的な概念などをイメージ化することで理解度を深め、問題解決に役立てる手法である。可視化の表現形態としてマインドマップ、図形チャートなどが用いられる。思考の可視化については本節の対象外とする。

(3) 科学的可視化

サイエンティフィック可視化とも言う。計算機シミュレーションの結果として得られる数値データをコンピュータ・グラフィックスの技術により目で見えるようにする。この概念は 1987 年に IEEE Computer Graphics and Applications で ViSC（Visualization in Scientific Computation）として最初に提唱された。

いろいろある可視化のうち、本節では「科学的可視化」に着目して、どのような技術で可視化が行われているかを説明する。

(1) 科学的可視化をするための現在の開発技術

科学的可視化が対象とするデータとして主に以下の二つがある。

① シミュレーションの結果として得られるデータ
② 実験や計測の結果として得られるデータ

後者に対しての可視化にあたり、補間やノイズ除去などのためにフィルタリングや統計処理などの前処理が必要な場合がある。前処理後のデータに対しては、シミュレーション結果データの可視化で用いられる手法やツールが適用できることが多い。

シミュレーションにもいろいろあるが、以下の三つに大別される。
① 連続型シミュレーション
物理、工学分野で多く行われているシミュレーションで、対象とする空間を格子に分割することで構成される要素間の関連性をモデル化した微分方程式を数値的に解くことで、時間経過と共に連続的に変化する状態量を求める。

② 離散型シミュレーション
物流、交通、通信などの待ち行列型システムに対して、システムの状態変化を起こす事象を離散的に発生させることで混雑現象を評価するためのシミュレーション。

③ エージェントシミュレーション
一定のルールに基づき自律的に振舞うエージェントの相互作用を分析することで、システム全体の振る舞いを分析するシミュレーション。

これらのうち、本節では、開発技術の蓄積が豊富で事例も多い連続型シミュレーションの結果データに対する可視化について着目し、詳しく説明する。

① 可視化の対象となるデータ
連続型シミュレーションの結果として得られるデータは、シミュレーションの対象とする空間内に切られた格子点上または要素上のスカラー値（温度、応力など）やベクトル値（速度など）である。可視化の対象となるデータは、空間内での格子や要素の位置と形状を決めている座標値と、その座標値上でのスカラー値やベクトル値である。

② シミュレーション結果データを可視化するまでの流れ

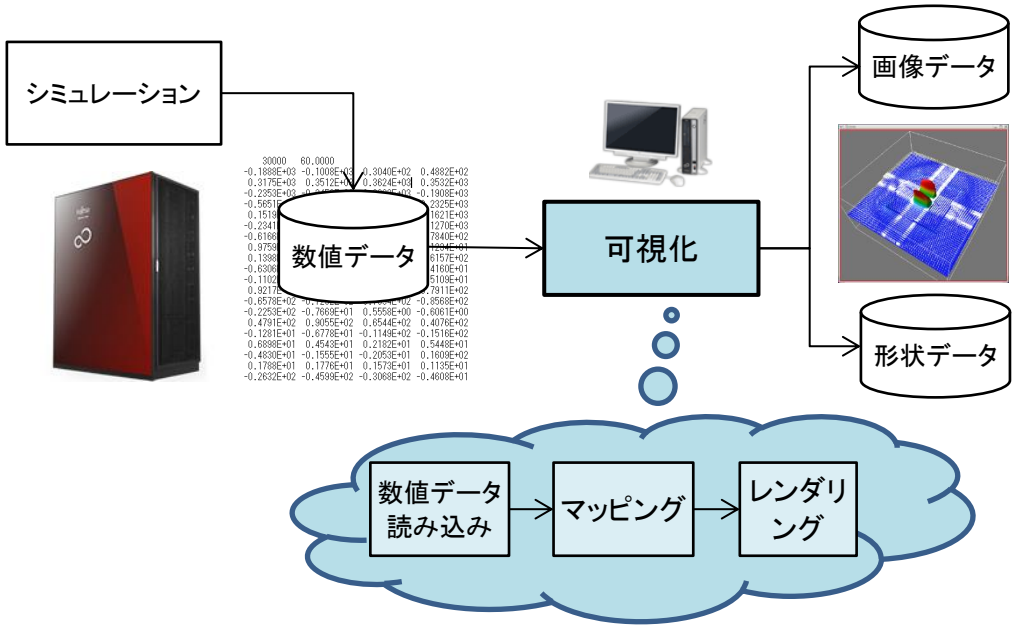


図 2.2-1 可視化処理の流れ

シミュレーションから出力された数値データを読み込み、画像データや形状データとして目でみえる形で出力するまでの流れを図 2.2-1 に示す。この一連の流れの中で「可視化」とされる処理は以下の三つの部分から成る。

(a) 数値データ読み込み

シミュレーション結果の数値データを読み込む処理。後述するように、シミュレーションの種類（流体解析／構造解析／計算化学）やプログラム（コード）の作成者、シミュレーションの実行環境（並列か逐次か、バイナリ（big／little エンディアン）か ASCII か）などによってデータのフォーマットが異なるので、これらの違いに対応した読み込みができる必要がある。

(b) マッピング

数値データを幾何形状に変換する。具体的にはシミュレーションの計算格子点を多角形（ポリゴン）でパッチし、シミュレーションの結果として得られた物理量（圧力、温度、速度など）に応じた色をポリゴンの属性値としてマップする。ポリゴンとしては後工程のレンダリングをハードウェアで効率的に行わせるために三角形を用いることが多い。

(c) レンダリング

マッピングで生成されたポリゴンを画像データ化することにより目で見えるようにする。他の CG ツールでレンダリングすることを想定して、画像データではなく例えば STL など汎用的なフォーマットの形状データとして出力することもある。画像データ化する場合の主な処理内容としては、3 次元の幾何形状を 2 次元画像に投影するためのビューイング変換、陰影付けのためのシェーディング、隠面処理などで、いわゆるスキャンコンバージョンとして総称される処理である。これら一連のレンダリング処理は、OpenGL などの API を呼び出すことによりグラフィックスボードのハードウェアで行うのが一般的である。Unity などのゲーム基盤は、可視化の一連の処理の流れのうち、レンダリングに対応する部分の機能しか備えていないことに留意されたい。

③ シミュレーションの種類と結果データ

可視化へのシミュレーション結果データの読み込みにあたり重要なのは、出力フォーマットがどうなっているかという点である。本節が対象とする連続型シミュレーションにも多くの種類があるが、シミュレーションの計算格子に着目すれば、表 2.2-1 に示すように、概ね、流体解析／構造解析／計算化学のシュミュレーションで用いられるフォーマットのいずれかに帰着する。

表 2.2-1 シミュレーションの種類と結果データ

種類		主なコード名称（※OSS）	計算格子	求める解
流体解析		FLUENT, STAR-CCM+, scFLOW, ※OpenFOAM	構造格子 非構造格子	格子点／接点／要素上のスカラー値（圧力等）、ベクトル値（速度等）
構造解析		LS-DYNA, Marc, NASTRAN	非構造格子	接点／要素上のスカラー値（応力等）、ベクトル値（変位等）
計算化学	分子動力学法	※GROMACS	離散データ	空間内の分子の位置
	分子軌道法	Gaussian	構造格子	格子点上の電子密度

④　シミュレーション結果データのフォーマット

シミュレーション結果データのフォーマットがどのようになっているかを、汎用可視化ソフト AVS の入力形式をもとに、計算格子毎の具体例を図 2.2-2 に示す。

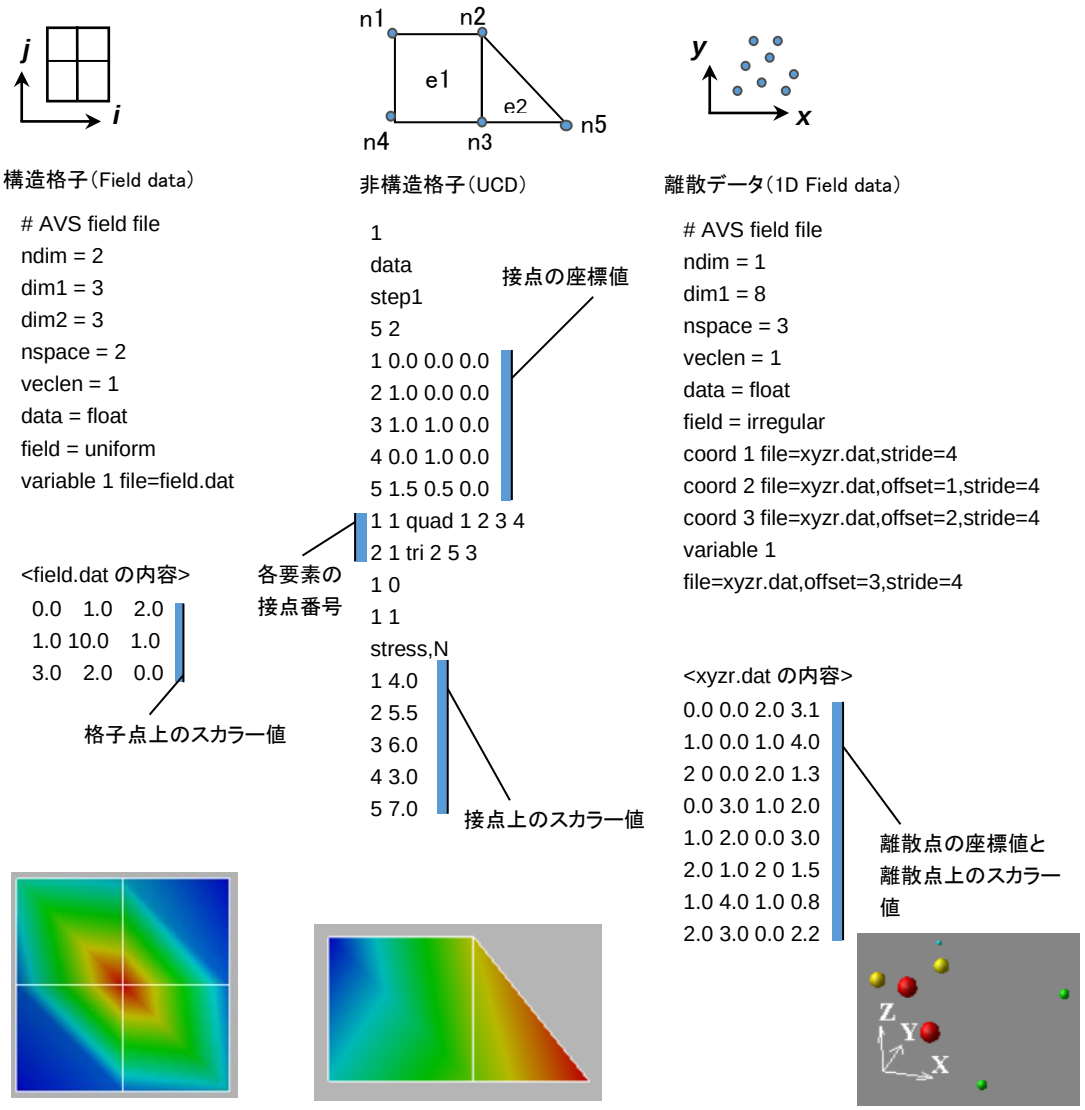


図 2.2-2　計算格子と解の具体例（AVS の入力形式）

このように、シミュレーション結果データには、計算格子点の座標値の並びと、シミュレーション結果として求められた解の並びが出力されている。商用のシミュレーションコードの多くはフォーマット非公開だが、上記の例と同様に座標値と解の並びが出力されている。

OSS のシミュレーションコードは下記のような汎用フォーマットで出力できるものが多い。フォーマットの詳細は Web で公開されている。

(a) VTK

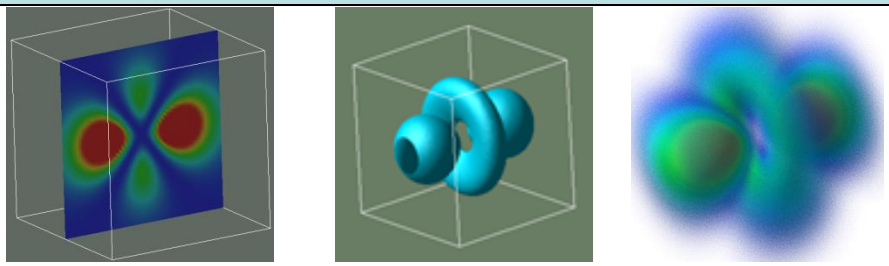
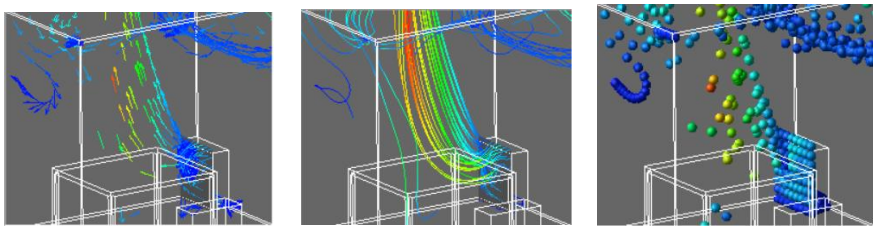
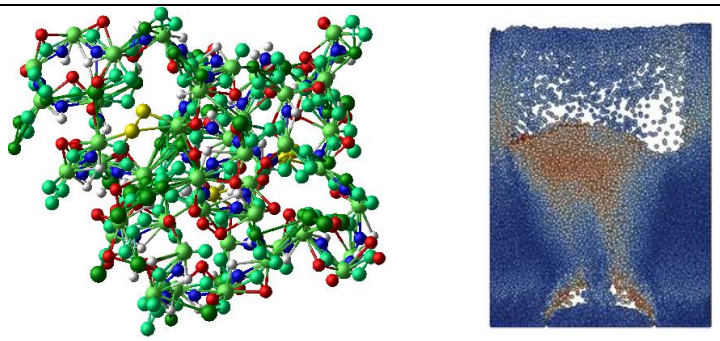
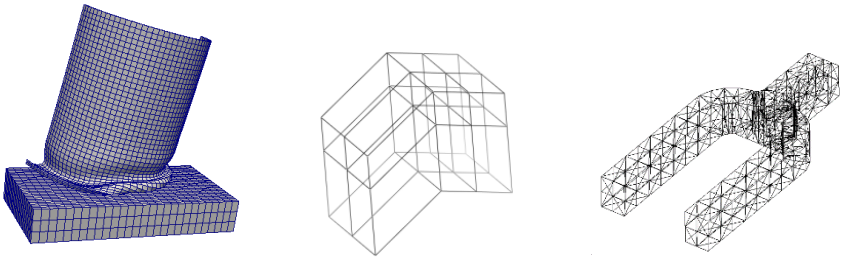
(b) PLOT3D

(c) CGNS

⑤　可視化手法

シミュレーション結果として得られたデータをどのように可視化したいかによって可視化手法を選ぶ。可視化の対象とするデータと、可視化手法の一覧を表 2.2-2 に示す。

表 2.2-2　可視化対象データと可視化手法

可視化対象	可視化手法	可視化例
スカラー値	コンター 等値面 ボリュームレンダリング	
ベクトル値	ベクトル図 流線 パーティクルトレース	
離散点	ボール＆スティック 粒子	
形状データ	メッシュ図	

⑥　可視化ツール

現在、多く利用されている代表的な可視化ツールの一覧を表 2.2-3 に示す。対象とするシミュレーションコード（ソルバ）の開発形態、データフォーマットの種類により、どの可視化ツールを選択するかを決める。

表 2.2-3　代表的な可視化ツール一覧

ソルバ	データフォーマット	可視化ツール（※OSS）
商用	独自（非公開）	ソルバ毎の専用ポストプロセサ EnSight（汎用ポストプロセサ）
OSS	独自（公開） 汎用（VTK, PLOT3D, CGNS 等）	FieldView（流体向け汎用） ※ParaView AVS
自作	可視化ツールの入力形式に合わせて出力	

⑦　可視化での VR の利用

可視化では 3 次元形状を正確に把握する目的で、両眼視差を利用した立体視表示は古くから行われてきた。商用の可視化ツールには偏光や液晶のメガネを用いて立体視表示する機能を標準で備えているものも多い。ただし立体視は VR（Virtual Reality）に必要な要素技術の一つにしか過ぎず、身体の動きなどの動作が仮想空間内に及ぼした変化を利用者に知覚させるフィードバックの仕組みを有すれば VR と定義される。

VR という言葉は 1989 年に VPL Research 社が発売した「データグローブ」という手袋型の入力デバイスの紹介で最初に使われたとされる。1990 年代に入り、主に軍事や産業分野での訓練を目的としたウェアラブルデバイスによる VR の利用が始まった。可視化においては当時、HMD などのウェアラブルデバイスでは充分な解像度が得られなかったため、人間の視野を覆う壁面に全方位映像を投影して没入空間を作る CAVE などの投影型 VR 装置が多く利用された。しかしシステムが大規模かつ複雑で初期導入費用だけでなく維持費もかかることから一部を除き使われなくなり、その後 10 年以上、可視化での VR 利用では大きな進展が無い時期が続く。

2016 年、主にゲームなどのエンターテイメント分野を対象として、高精度のセンサを備え、かつ、高精細な表示が可能な Oculus Rift などの HMD が個人でも購入可能な価格で相次いで発売されるようになった。これらの HMD の性能向上と低価格化はスマートフォン向けの小型で高精度のジャイロセンサや重力センサ、ディスプレイパネルの高精細化によりもたらされた。最近の HMD は 1990 年代のそれとは違い可視化の用途にも充分に堪える解像度を持っていることから、現在、これらのデバイスとそのアプリケーション開発基盤である Unity などのゲームエンジンを用いて、可視化での VR の利用を試みる動きが盛んになりつつある。

可視化での VR の利用においては以下のような効果が期待されている。

(a) 3D 形状の把握

計算化学での原子や分子の 3 次元構造の把握、流体解析や構造解析での 3 次元計算格子の形状が適切かどうかの確認、電磁界解析での多層プリント基板の配線の状況の把握など、特に 3D 空間内にワイヤーフレーム表示で可視化する場合、ワイヤーフレームを構成する線

分の前後関係は立体視をしないと正しく把握できないことが多い。このａについてはVRでなく立体視だけでも期待する効果は得られるが、利用者の身体の動きをレンダリング時のビューイング変換にフィードバックするヘッドトラッキングと呼ばれるVRの機能を有しているほうが、立体視が及ぼす利用者への疲労が少ないとされている。

(b) 没入視点での可視化

建築物や大型実験装置など、人間の身長と同等以上のサイズのモデルを対象としたシミュレーションに対する結果の可視化にあたり、VR による没入視点で可視化することで新たな知見が得られることが期待される。

(c) 高品位レンダリング

シミュレーション結果の可視化の目的の一つとして、成果公開やアウトリーチ活動など一般の人向けのプレゼンテーションがあり、その際にVR用のツールを使い上記のa,bの効果に加えて、ゲーエンジンの持つ高品位なレンダリングで見せることで、可視化結果を通じて研究成果に対する興味や理解をより深めてもらう効果が期待される。

上記のうち(a)(b)については、既に可視化ツール側でVR対応として、立体視並びにヘッドトラッキングによる視点移動の機能を備えたデバイスに直接表示できるものもある。VR対応していない可視化ツールであってもそのレンダリング処理部分がOpenGLで作成されているものであれば、Easy VR（有償）というOpenGLの表示内容をキャプチャーしてVRデバイスへ転送するツールを介在させることにより(a)(b)に示すレベルのVR対応は可能になる。

上記(c)については、Unity などのゲームエンジンが持つレンダリング機能の利用が期待されるが、可視化処理の内の数値データ読み込みとマッピングの部分をゲームエンジン上で動作するよう移植あるいは開発し直す必要があり容易には実現できない。また「3.1.1 ソフトウェア開発方法」で詳述するように、ゲームエンジン上でのデータの持ち方や描画の手順が、そもそもこれまで開発されてきた可視化ツールのそれとはかなり異なるという課題もある。

2.3 科学的可視化で何をやりたいのか？

下記、表 2.3-1 は様々な分野における科学的可視化の有効性を示したものである。

表 2.3-1　科学的可視化の有効性

項目	分野	概略
2.3.1	大規模シミュレーション向け In-Situ 可視化とその VR 化の課題（河村）	大規模シミュレーション向け In-Situ 可視化とその VR 化の課題を解決することが、原子力分野におけるシビアアクシデント対策の強化につながる。
2.3.2	数学分野での可視化（北基）	数学分野での可視化は、一つは数学の研究・数学そのものを進めることに寄与し、もう一つは数学教育・数学の学習や鑑賞に寄与する。ただ、研究には学習は欠かせないので、この二つは完全に分かれているわけではない。
2.3.3	シミュレーション研究の発展に寄与（大野）	大規模な 3 次元シミュレーション結果を可視化し、シミュレーション研究の発展に寄与する。
2.3.4	核融合プラズマ・プラズマ物理への活用（大谷）	複雑な構造の核融合炉におけるプラズマ現象を階層間の相関や様々な物理量間の相関を可視化できるようにすることにより、建設やメンテナンス作業にかかるコストの低減につながる。
2.3.5	可視化空間と実空間を繋ぐインターフェイス（田村）	可視化空間と実空間を繋ぐインターフェイスを構築することで、人はバーチャルな空間を主体的に体験できるようになる。
2.3.6	災害の可視化と仮想体験の共有（安福）	災害を見える化して多くの人と災害の仮想体験を共有し、今後の起こり得る災害に対する想像力を養うことが、災害時の避難誘導計画の立案や、リアルタイムの避難誘導に活用につながる。

2.3.1 大規模シミュレーション向け In-Situ 可視化とその VR 化の課題

日本原子力研究開発機構　河村拓馬

原子力分野ではシビアアクシデント対策として、原子炉圧力容器内部の流体計算や原子力施設のための耐震設計、あるいは都市圏に対する緊急時の汚染物質の拡散予測など、実験計測だけでは対応できない大規模な事象を高精度でシミュレーションすることが求められる。事象の規模だけでなく正確さまで求められるため、モデル化技術の向上に加えて、スーパーコンピュータを用いた高解像度のシミュレーションが必要とされている。計算機技術の発展によって、シミュレーションの規模はテラスケールからペタスケール、そしてエクサスケールへと成長しており、このような大規模シミュレーションから知見を抽出するための可視化技術が必要とされている。原子力分野における大規模シミュレーション向けの可視化技術には大きく分けて二つの課題がある。一つは、遠隔可視化技術の開発であり、もう一つは高精度化したシミュレーションに対するリアリティのある可視化である。

従来の遠隔可視化では、遠隔地のストレージ上に保存されたシミュレーション結果の生データを手元の PC に転送して可視化アプリケーションで可視化してきた。しかしシミュレーションが大規模化することでネットワーク帯域幅が不足し始め、データ転送に数時間から数日、あるいは数週間を要するようになった。同時に大規模なデータは PC のメモリ容量や処理性能の限界を超え、可視化が困難になってきた。そこで遠隔地にあるサーバ（ストレージが接続されたクラスタやスーパーコンピュータ）を利用して可視化処理を実行し、可視化用データを手元の PC に転送してクライアント（手元の PC）で観察する、クライアント・サーバ型可視化が利用されるようになってきた。さらに計算機の演算密度が増加することで、FLOPS とストレージへのデータ I/O 速度の差が広がり、エクサスケールシミュレーションでは結果データの出力自体が困難になってきた。この問題に対処するために、バッチ処理されるシミュレーションと同じ並列計算環境上で可視化処理も同時に行い可視化結果を出力する、In-Situ 可視化が重要視されるようになった。

モデル化技術の向上によるシミュレーションリアリティの増加によって、計算対象の原子炉建屋モデルはバネモデルから有限要素モデルになり、都市気流計算は多くの建築物を含む数キロ圏の範囲に拡大された。こうした状況は可視化に対してもリアリティのある可視化を要求し、従来可視化アプリによる遠くから全体を俯瞰するだけの可視化から、HMD 装置を用いて建屋や都市の中に没入する臨場感のある可視化が必要になった。そして上述の遠隔可視化技術と VR 技術を組み合わせ、スーパーコンピュータ中の計算空間を自由に歩き回る、クライアント・サーバ型あるいは In-Situ 型のウォークスルー可視化が必要とされている。

しかし、従来的な In-Situ 可視化の枠組みでは、ウォークスルー可視化の実現が困難である。ウォークスルー可視化では、カメラの視野角の範囲にあるデータを対話的に可視化処理することが必要になる。シミュレーションのバッチ処理実行時に可視化画像を配信することで対話的に In-Situ 可視化する技術[2.3.1-1]は開発されているが、可視化画像の生成は

各タイムステップに実行される。そのため、シミュレーションと独立した視点変更が難しく、自在にウォークスルーすることが困難である。これに加えて、従来的な In-Situ 可視化に用いられる可視化の並列処理性能が、シミュレーションを圧迫する。可視化用ポリゴンデータに基づく従来の可視化アルゴリズムでは、ステンシル計算で用いる領域分割が、可視化処理のためのデータ領域構成やデータ探索に伴う大域的通信を生じさせる[2.3.1-2]。そのため、並列数が増すほどに通信量が増大し、可視化処理のコストがシミュレーションのスケーラビリティを悪化させる。In-Situ 型のウォークスルーが可能な可視化技術の開発は、大規模シミュレーション向け可視化の分野における重要な課題である。

参考文献

[2.3.1-1] T. Tu, H. Yu, J. Bielak, O. Ghattas, J.C. L'opez, et. al.: "Analytics challenge - remote run-time steering of integrated terascale simulation and visualization", Proc. of SC2006 (2006) pp. 297-303.

[2.3.1-2] T. Peterka, H. Yu, R. Ross, K. L. Ma: "Parallel volume rendering on the IBM blue gene/p", Proc. of EGPGV'08 (2008) pp. 73-80, 2008

2.3.2　数学の可視化で何をやりたいのか

広島大学大学院教育学研究科　北臺如法

ここでは「数学の可視化で何をやりたいのか」を述べる。

これには大きく二つある。一つは数学の研究・数学そのものを進めることについて、もう一つは数学教育・数学の学習や鑑賞についてである。 ただ、研究には学習は欠かせないので、この二つは完全に分かれているわけではない。

まず、数学の研究や数学そのものを進めることについては、現代数学に現れる数学的対象を可視化し、そこから新しい情報・知見を得ることが重要な目標である。 VR 技術を使うことで 3 次元までであれば直接的に可視化ができるので、様々な数学の概念の定義や、理論に出てくる定理を実際に見てみることができる。4 次元以上となると直接的には見られないので様々な工夫が必要である。実際に図形を紙に描いて思考することに比べて、直接的に数学的な実験ができるとなると、それだけで考える手段が増える。（あまり知られていないが、数学には研究段階で実験をする分野も多い）これにより、行く行くは新しい数学を開拓できると非常に良い。

もう一つの数学教育・学習・鑑賞についてであるが、これは筆者のこれまでの教育経験から、可視化を取り入れることで数学の理解の補助にしたいという強い思いがある。 小学校算数から大学数学まで、数学には幾何学的な対象、図形が多く出現するにもかかわらず、学習段階で簡単に理解できない図形が出てくることがよくある。 特に空間図形（3 次元の図形）の理解が難しく、黒板にも教科書にも、2 次元に射影したものしか描かれないため、わかる人にしかわからない説明になりがちである。 そこをこれまで各教員が工夫してきたのであるが、この理解補助に数学的な可視化が有効に働けばという思いがある。後の事例集で筆者の取り組みを紹介する。

また、学習においてモチベーションは大切である。 既に数学が面白いと思っており、興味関心のある児童、生徒、学生、学習者だけではなく、 残念ながら数学に苦手意識を持っていたり嫌いになってしまう学習者も存在する。 学校数学によって、数学が嫌いになってしまった大人が少なくないというデータもある。 数学的可視化は、苦手意識を持った学習者の苦手意識をまず減らし、楽しいものだと思ってもらうことにも価値があるのではないかと考えている。 まず、数学を学ぶ玄関に入ってもらえば、その後の学習意欲が全く違うものになる。 この、自分の内から出てくるモチベーションを動機付けの理論では内発的動機付けという。 楽しそうだと感じ、学習意欲が湧き出すための手段として、可視化を用いることが考えられる。

最近では、ブラウン大学の ICERM での Illustrating Mathematics という特別な半年企画の中で 行われた研究集会 Illustrating Geometry and Topology （Sep 16 – 20, 2019） https://icerm.brown.edu/programs/sp-f19/w1/ が、数学（特に幾何学やトポロジー）、コンピュータ・グラフィックス、計算機科学の研究者のみならず、芸術のバックグランドを持つ研究者も含めて催され、様々な数学の可視化にとどまらない、その楽しみ、芸術との接点

も試みが多く行われた。 研究集会の様子の動画が Illustrating Geometry and Topology at ICERM - YouTube https://www.youtube.com/watch?v=llsXrD87vXs&list=PLTcTErKOWlcmsZC5QqDlYSefaXeW6-xGW のリストにあり、初日のダイジェストを見ることができる。 さらに上記の研究集会のページからも、参加者のサイトを訪問すると様々な試みをされていることがわかる。 これは、数学の鑑賞という点でも数学の可視化が注目されていることを表している。

数学は古来より可視化とは切っても切れない関係にあると言える。 幾何学はそもそも図形・かたちという数学的な対象を扱うものであり、元々見えるものが多い。 関数においては、そのグラフを考えるということは、関数という、対応関係というオブジェクトから、グラフという図形的・幾何学的オブジェクトを作り出すということであり、関数の可視化に他ならない。

しかし一方で数学は、目で見えないものや、仮に見えるものであったとしても対象を目に見る意味で見るのではなく、数学の持つ強力な論理の力で理解することもしてきた。 したがって、必ずしも可視化をすることで数学の理解に繋がるというわけではなく、逆にある数学が理解できたときにそれが可視化によるものとは限らないということには注意が必要である。 このことに注意をしながら、可視化によるメリットをうまく数学に活かしていきたい。

2.3.3　シミュレーション研究の発展に寄与

兵庫県立大学　大野暢亮

大規模な 3 次元シミュレーション結果を可視化し、シミュレーション研究の発展に寄与する。可視化の方法は、以下のとおりである。

(1) スパコン上でシミュレーションと同時に可視化を行う In-Situ 可視化

(2) CAVE を始めとするVR装置を利用する可視化

(2)のVR可視化は、複雑な3次元構造を把握することが容易になると言われている。

2.3.4　核融合プラズマ・プラズマ物理への活用

核融合科学研究所　大谷寛明

プラズマ現象では、空間的にも時間的にも複雑な構造を示し、また、電子やイオンといったミクロな階層から、磁場閉じ込め装置や太陽フレアといったマクロな階層までがかかわる。階層間の相関や様々な物理量間の相関を可視化できるようにしたい。

将来の核融合炉は多くの機器が取り付けられ、とても複雑な構造になる見込みである。この建設やメンテナンス作業の確認を実物のモックアップで行うとなれば、それだけで大変なコストとなる。これらの作業を行うロボットシステムも含めて可視化・検討ができる仕組みが必要である。

2.3.5　可視化空間と実空間を繋ぐインターフェイス

甲南大学　田村祐一

科学的可視化の「可視化」部分より、最近はどのように見せるか、体感させるかに興味がある。これまで様々な可視化手法を使って現実には直接見ることができないものを可視化してきたが、見るだけでは体験者は基本的に「傍観者」であった。そこで「傍観者」ではなく、可視化された空間、バーチャルな空間に主体的に体験者を関与させることに興味がある。そのために可視化空間と実空間を繋ぐインターフェイスについて様々な取り組みを進めている。

2.3.6　災害の可視化と仮想体験の共有

大阪大学　安福健祐

科学的可視化は見えない現象を見える化して深く理解するために行うことが多いが、私が専門としている災害の可視化というものを考えるとき、災害を見える化して多くの人と災害の仮想体験を共有し、今後の起こり得る災害に対する想像力を養ってもらうこともやりたいことの一つである。現在はマルチエージェントシステムによる避難シミュレーションによって避難状況を可視化し、群集の混雑を緩和する避難誘導計画の立案や、リアルタイムの避難誘導に活用できるのではないかと考えている。また、マルチエージェントシステムの対象として人間だけを扱うのではなく、それを応用した交通流の大規模可視化などにも取り組んでおり、可視化によって社会を構成するもの同士の相互作用により生じる複雑な現象の理解を促進することを目指している。

3. なぜゲームエンジンなのか？なぜUnityなのか？

核融合科学研究所　大谷寛明

3.1 Unityとは

可視化のプロセスでは、可視化の目的によって利用する可視化デバイスやプログラムが異なることが多い。例を以下に挙げる。

(1) 自らの計算・実験結果を3次元CGなどで確認をする場合、手元のPC等で汎用可視化プログラムを使用することが多い。

(2) 多人数での結果共有や議論、一般の方への説明をする場合、CompleXcopeなどの専用・大規模可視化装置で各VR・可視化装置専用の可視化プログラムを使用することが多い。

(3) 学会等での発表時や可視化結果共有の場合、ハンドヘルドデバイスでAndroidやiOS用のプログラムとPC用プログラムが別々の場合が多い。

それぞれの状況に合わせた可視化開発環境を別々に用いては大変な手間となるため、全ての場合において汎用的に利用可能な環境の構築が必要である。そのような開発環境の候補の一つがUnityのような汎用ゲームエンジンの活用である。汎用ゲームエンジンは様々OS上で実行可能で、Windows、Mac、iOS、Androidなどで実行可能で実行ファイルを出力可能である。また、PCの画面上だけでなく、ヘッドマウントディスプレイやzSpaceなど、様々なVR装置に出力することも可能である。

CAVEとUnityの橋渡しという側面で考えてみると、以下のようなことが可能である。

(1) Unityで作成したプログラムはMiddleVRというミドルウェアを組み込んでCAVEで表示することができる。

(2) CAVELibを使って作成されたVRプログラムは、ラッパープログラムCLCL[3.1-1]を組み込むことで、Oculus Rift、HTC VIVE、WindowsMRで表示することができる。

CAVEで開発された既存プログラムのHMDプログラムへの移植やUnityで開発されたプログラムのCAVEでの表示は容易に行うことができるようになっている。

このような可視化開発環境を使えば、共同研究や社会への発信をこれまで以上に強力に推し進めていけることが期待できる。

参考文献

[3.1-1] S.Kawahara and A.Kageyama: “Development of CAVELib Compatible Library for HMD-type VR Devices’’, Journal of Advanced Simulation in Science and Engineering, Vol.6 (2019), pp.234-248.

3.2 Unreal Engine との比較

現在、汎用ゲームエンジンとして広く活用されているのが、Unity と Unreal Engine である。ここでは、両者を比較することで、それぞれの得手不得手を考えたい（表 3.2-1)。なお、ここでの比較は、2018 年 7 月 3 日の WG 会合で行われた株式会社サイアメントの瀬尾拡史氏の講演資料を元にしている。

表 3.2-1　Unity と Unreal Engine の比較

言語	
Unity	C# C++でしか存在しないライブラリを使いたい場合、C#から呼び出せるように DLL を作ってそれを Unity から読み込むように設定する必要がある。 参考：Unity の Native Plugin を作って C++のコードやライブラリを使う http://yuki-koyama.hatenablog.com/entry/2015/11/18/192506
Unreal Engine	C++ Unreal Engine の機能をフル活用するためには独自の動的配列や型、命名規則などに従う必要があるが、設定すれば通常の C++や外部ライブラリも使える。 参考：鈴木　晃(著)、C++でつくる Unreal Engine アプリ開発 for Windows & macOS　～初歩からプラグイン開発まで～（(株)ラトルズ発行)。 OpenCV との連携方法なども具体的に記載されている。
スクリプト	
Unreal Engine	Blueprints（ブループリント ビジュアル スクリプティング システム） Unreal Engine 最大の特徴である。ビジュアルスクリプトでノードを繋ぐことで、スクリプトをゲームに視覚的に追加していく。ノード、イベント、関数、変数をワイヤーと接続して複雑なゲームプレイ要素を作成する。Unreal C++で書いた関数をノードとして Blueprints から呼び出すことも可能である入力、出力も複数可能（Blueprints に対応できる型は制限あり)。デバッグが視覚的に分かりやすく行うことができる。 参考：https://docs.unrealengine.com/ja/Engine/Blueprints/index.html
HoloLens への対応	
Unity	対応
Unreal Engine	非対応 Microsoft が公式に、Unreal Engine からの UWP 出力を可能にするソースコードを公開しているが、Unreal Engine としては非公式。
zSpace への対応	

Unity	zSpace による SDK あり
Unreal Engine	zSpace による SDK なし。
SteamVR、Oculus、Daydream、ARKit などへの対応	
Unity	どちらも対応。
Unreal Engine	
モバイル開発	
Unity	開発環境が充実しており、デバッグもしやすい。 デバイスから取得可能な値 ● タッチとスタイラスによる入力 ● 加速度計 ● ジャイロスコープ ● デバイスのカメラストリーム ● コンパス ● GPS ● ジョイスティックの名前と入力 エディタ上の画面を圧縮してデバイス上に送信可能 参考：https://docs.unity3d.com/jp/460/Manual/PlatformSpecific.html
Unreal Engine	バージョン 4.20（最新版は 4.23）からモバイル開発用の iPad 用補助アプリが Unreal Remote から Unreal Remote 2 となり、画質はあまり良くないが、Unity Remote と同じように、ゲーム画面がアプリ上に映るようになったので、デバッグ作業が楽になった（Mobile Device Preview)。 参考：https://docs.unrealengine.com/ja/Platforms/Mobile/index.html https://www.slideshare.net/EpicGamesJapan/ue4mobilestudy 一方、アプリ経由で取得出来るセンサ情報は Unreal Remote から変わっておらず、以下の値が取得可能。 ● タッチ ● 加速度計 ● ジャイロスコープ
GPGPU	
Unity	扱いやすい ComputeShader を簡単に使うことが出来、CPU 側のメモリを介さずに直接 GPU 上でいろいろといじることができる。 参考：IndieVisualLab（著）、Unity Graphics Programming Vol.1,2,3,（IndieVisualLab 発行）

Unreal Engine	困難 CUDA との連携で色々可能ではあるが、Unreal Engine を経由する場合、一旦 GPU 側から CPU 側に戻し、再び Unreal Engine の API を使って GPU 側に送る必要があるため、大きなボトルネックになる。 一応、FRHIComputeShader というクロスプラットフォームの API が Unreal Engine 側で用意されているようだが、情報が少なすぎて詳細は不明。
バージョン管理	
Unity	コード関係はテキスト化が可能である。 Unity 側の設定で、Asset Serialization オプションを Force Text にすることで、バイナリ形式で保存されているものを ASCII 形式で保存・管理出来るようになる。Git との相性が良い 画像ファイルなどはバイナリなので、Git LFS と適宜併用ができる。
Unreal Engine	Blueprints はバイナリ形式で保存されるのみ。 Blueprints は必ずバイナリ形式で保存されてしまうため、わずかに変更しただけでも Git 上では差分が取れない。 例えば、ノードの位置をわずかに動かした、といった場合でもファイルを丸ごと上書きすることになる。チーム開発が非常にやりづらい。 Unreal Engine エディタ側では Git 管理の差分を表示出来る機能があるが推奨されていない。 SVN または Perforce が推奨されている。

4.　標準的なワークフローにおけるノウハウ

4.1 ソフトウェア開発方法

東京都市大学　宮地英生

甲南大学　田村祐一

(1) 可視化のパイプライン

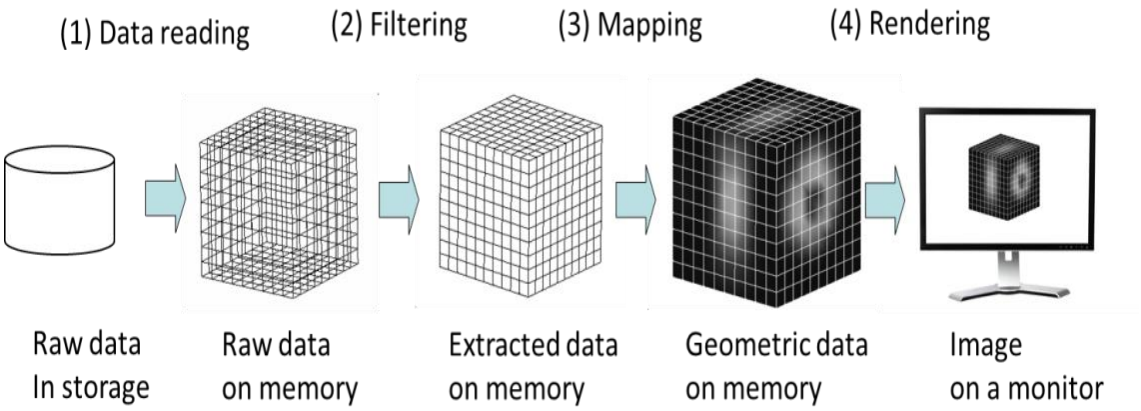


図 4.1-1　可視化のパイプライン

一般に数値計算結果が画像になるまで、上記のような 4 ステップのプロセスでデータが変換される（図 4.1-1)。

① データ読込み

② フィルタリング

③ マッピング

④ レンダリング

この後、画像を見ながら対話処理が行われ、研究者は数値計算結果の確認、数値計算結果からの発見、そして、結果の整理を行う。また、結果の整理が終わった後、その成果を一般に知らせるために可視化結果の画像が利用される。

結果の確認と整理には、決められたルーチンで可視化画像が生成されることが多く、ここではバッチ処理での可視化が行われる。

対話的な可視化処理が求められるのは試行錯誤の段階で、ここでは可視化のパラメータを変更し、いろいろな角度から結果を観察して「気づき」が得られると言われている。

最後に成果を知らせるための可視化では主張する点が明確になっているので、ある程度の演出が可能となる。また、その計算結果による設計変更が最終製品にとって有益であることを示すために AR/VR/MR (XR) が利用されることもある。この動向は顕著で、2012 年 Oculus Rift の開発に始まり、多種多様な HMD 装置、Kinect のようなモーションセンサ、デプス情報が取得できるカメラなど急速なコストパフォーマンスの改善が行われ VR の中にシミュレ

ーション、および可視化が組み込むことが必要とされている。また、Google Tango (ARCore) に代表される特徴点抽出から周囲の 3 次元情報を動的に取得する技術も普及し、AR も生活の中で実用的に利用できるようになってきた。

しかし、このような多くの先進技術のインターフェイスを開発することは困難である。そこで注目されているのが Unity、Unreal Engine といったゲームエンジンである。最新の VR/AR 機器を開発した会社は、それを広く利用してもらうため、これらのゲームエンジンでの利用を提供している。ゲームエンジンは高速・高品質のレンダリング能力に加え、ゲーム開発のための優れたインターフェイスを持っているのでグラフィックスライブラリの知識を持たない人でも容易にアプリケーションソフトウェアを開発することができる。

(2) Unity と可視化のパイプライン

Unity はゲームのシナリオを実現するエンジンであり、モデリングの機能は持っていない（図 4.1-2)。ゲームのキャラクタは Blender などのソフトウェアでモデリングされ、そのジオメトリやアマチュアなどのボーンモデルを持ったモーションも含めてゲームエンジンにインポートして利用される。

可視化のパイプラインではマッパー（可視化手法）が適用されたのちジオメトリのモデルが生成される。したがって、このタイミングで Unity にジオメトリ情報を受け渡し、レンダリング処理と VR/AR での利用をゲームエンジン側で担当すれば簡単にゲームエンジン上で可視化結果を見ることができる。しかし、ジオメトリデータを可視化ソフトウェアで生成し、ファイル経由でゲームエンジンに受け渡したのでは、可視化処理で重要な対話処理ができなくなってしまう。

また、可視化処理では画像化に際して、サーフェイスレンダリングとボリュームレンダリングの 2 種類のレンダリングを用いるが、医用画像で標準的に利用されるボリュームレンダリングはゲームエンジンに載せることが難しいと思われる。(というか、レンダラーまで新たにゲームエンジン上で開発したのでは、プラットフォームとしてゲームエンジンを利用する価値が低すぎるのではないか)

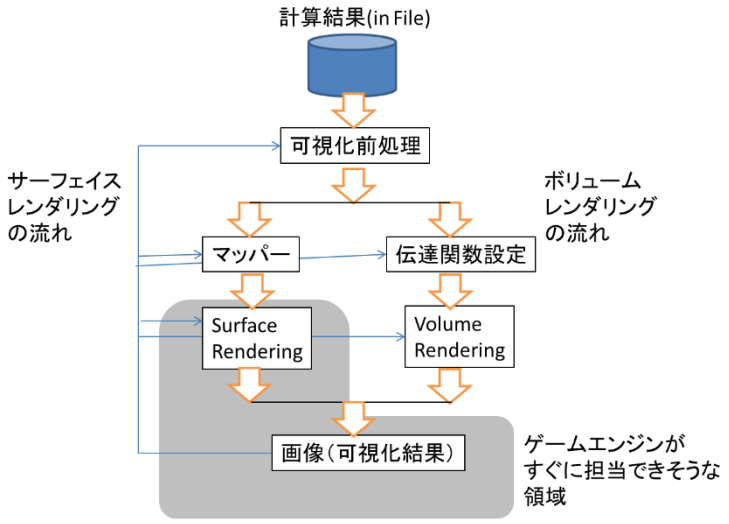


図 4.1-2　ゲームエンジンがサポートする可視化パイプライン

これらのことを考慮すると利用方法と、ゲームエンジンと可視化ソフトの統合方法は、次のようにまとめられる（表 4.1-1）。

表 4.1-1　目的別のゲームエンジンと可視化ソフトの連携方法

可視化処理方法	目的	ゲームエンジンと 可視化の連携方法
パッチ処理可視化	計算結果の確認 論文の可視化	連携不要
対話処理可視化	新しい発見や現象の考察のため	ゲームエンジンへの可視化機能の搭載通信による連携
プレゼンテーション可視化	製品や計算結果を知ってもらうため	可視化結果の受け渡し
シミュレーション統合環境	バーチャル体験に計算が必要な場合 実環境からシミュレーションへのリアルタイムフィードバック	可視化、シミュレーション、ゲームの統合

① 数値計算結果の確認と整理：ここでは従来型の可視化ソフトウェアがあれば十分である。

② 対話処理可視化：これを実現するにはゲームエンジンへの可視化機能の搭載、または、通信による連携が考えらえる。通信を利用する方が既存資産を活用できるが、ゲームエンジンの開発速度が速いので、可視化機能をゲームエンジンに搭載してしまうのが速くて便利と考える。

③ プレゼンテーション可視化：これにはジオメトリの情報、および、ジオメトリと動きの情報でゲームエンジンに渡すことが考えられる。しかし、これにはいくつかの課題がある。この点は後述する。

④ シミュレーション統合環境：ゲームエンジンがホロレンズや各種センサと接続されていることを考えると、ここからのリアルタイムデータをシミュレーションに渡し、可視化も含めた統合環境を作るのが未来的と考える。このときシミュレーションの種類によっては通信を用いた方が便利かもしれない。機械学習の結果を用いる場合も同様にサーバに紹介して結果を獲得する手法が面白いと思われる。

(3) ゲームエンジンと可視化ソフトの違い

高品質なレンダラーに可視化結果を受け渡す構想は古くから存在したが、いくつかの問題から普及しなかった。

① 高品質レンダリングにかからない線や点がある

レイトレーシングのようなレンダラーでは面しか扱わない。しかし、これはラインをパイプにすることで容易に解決できる。

② 色の持ち方が異なる

ゲームエンジンでは頂点に色を持つことが無い。マテリアルを定義してレンダリングするのが普通。すべてをテクスチャマッピングにするか、頂点カラーのレンダラーを入れるかの 2 択になる。

③ 時系列データの持ち方が異なる

これが最大の課題である。流体解析で利用するパーティクルトレースも時系列の等値面の変化も、数値計算では連続性がなく、頂点数が増減する。一方、ゲームエンジンにおけるアニメーションは「変形」が多く、頂点の増減が無い（場合が多い）。

(a) 外部デバイスとの接続

従来型の可視化と異なり、VR 技術を用いた可視化を行う場合、外部ハードウェアとの接続が必要となる場合も少なくない。その際ゲームエンジンは大きな助けとなる。従来は A と B というハードウェアを同時に使用したい場合、それぞれのハードウェアで用意された SDK を、一つのプログラムに組み込むことでシステムを構成していた（図 4.1-3）。もちろん、うまく組み合わせることができる場合も多いが、それぞれの SDK に競合が起こり、2 種類の

ハードウェアを同時に稼働させられないことも少なからず起こった。もちろん、ハードウェアの種類が多くなればなるほど、競合の可能性は高くなる。

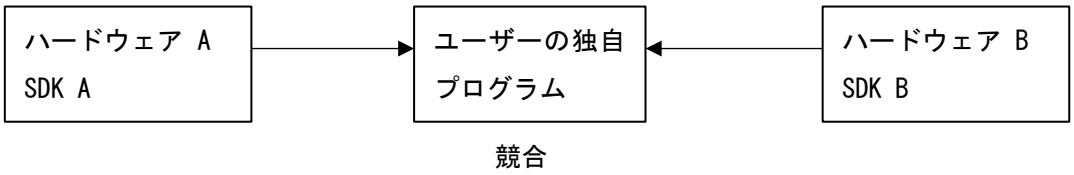


図 4.1-3　従来の開発方法

一方、ゲームエンジンを利用することで、競合が起こる場合が極めて少なくなる。それぞれのハードウェアとゲームエンジンの接続に関しては保証されており、ユーザーはゲームエンジンの中でプログラミングを行うだけなので、競合が起こる可能性が小さくなるという大きな利点がある（図 4.1-4)。

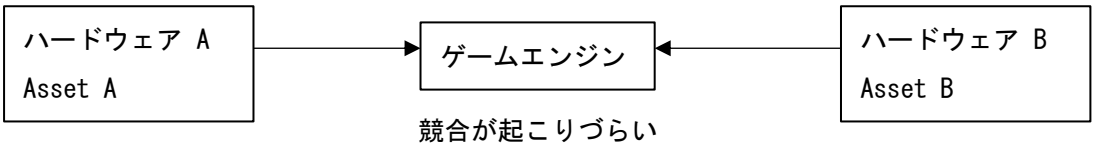


図 4.1-4　ゲームエンジンを使った開発方法

つまり、複数の特殊な（OS で一般的にサポートされていないという意味での特殊）デバイスを使用する場合、ゲームエンジンの利用を考えることは後のトラブルを大きく減らす効果がある。また、最近の VR コンテンツの多くはゲームエンジンで作られているものが多く、これに伴い VR に関連するハードウェアに Unity の Asset が用意されていることが一般的になってきている。

次に、開発を進めていくためにハードウェアだけでなく、外部ライブラリの使用も必要となる。近年、機械学習等で Python が広く使われるのはライブラリの充実度が高いからであるのと同様、ゲームエンジンにも様々なライブラリ、Asset を利用してプログラミングが可能である。例えば画像処理のスタンダードである OpenCV やフィジカルコンピューティングで利用する Arduino といったハードウェアとの接続も有料ではあるものの、Asset を購入することで、素早い開発が可能となる。ただ、科学技術計算ライブラリに関しては、C++などで記述されているものも多い。例えば、Unity のプログラミング言語は C#であるため、そのまま利用することは難しいが、使用時に制約はあるものの、C++で書かれたライブラリの DLL を利用することで利用可能である。

4.2 形状メッシュのインポート法や物理量のマッピング法など

富士通　大日向大地、小笠温滋

4.2.1 ParaView

本節では、ParaViewでの可視化結果をUnityにインポートするためのデータ変換を記す。

(1) VTKのデータ形式について

VTK（Visualization Tool Kit）のデータ形式について簡単に知っておく必要がある。VTKは科学技術計算での可視化ではデファクトスタンダードな可視化ライブラリで、ParaViewも内部ではVTKを使っている。

VTKは表4.2.1-1及び図4.2.1-1に示す形式（VTK形式）の格子データに対して様々な処理を行うフィルタを連接したパイプラインを組むことで可視化を行う。パイプラインで処理した結果もまたVTK形式である。例えば、構造格子の入力として閾値フィルタを適用すると、非構造格子のデータとして出力される。

また、これら複数の格子を複合する形式がある。よく使うものとして、Multi-Block DataSet （拡張子vtm）がある。粒子法などで用いる粒子データ専用の形式は無く、粒子データは点要素（vertex）のみからなる非構造格子で表現する。

表4.2.1-1　VTKで扱える格子形式

	形式名	拡張子	説明
構造格子系	Structured Grid	vts	曲線格子
	Rectilinear Grid	vtr	不等感覚直交格子
	Image	Vti	等間隔直交格子
非構造格子系	Unstructured Grid	Vtu	非構造格子
	Polygonal Mesh	Vtp	Line,Tri,Quad要素のみ（ポリゴン）

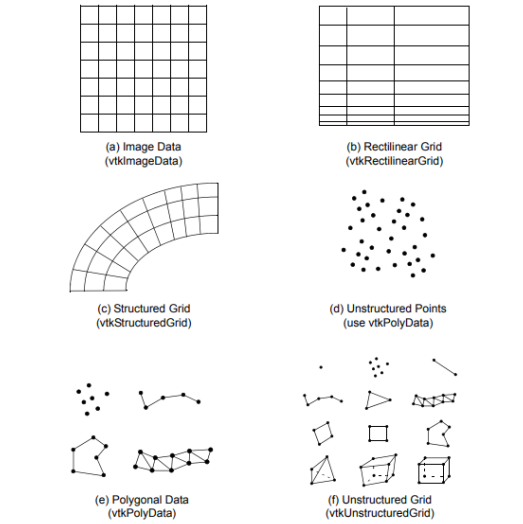


Figure 3–2 Dataset types found in **VTK**. Note that unstructured points can be represented by either polygonal data or unstructured grids, so are not explicitly represented in the system.

図 4.2.1-1　VTK で扱える格子形式（VTK ユーザーガイドから引用）

VTK の場合、物理量（速度、圧力、温度など）の格子上の定義位置は、「点」と「セル」の 2 種類がある（図 4.2.1-2）。点定義とセル定義は、数値処理によって相互に変換可能で、ParaView では、Point Data to Cell Data フィルタ、または Cell Data to Point Data フィルタを使う。この変換は数値処理（補間や平均）によるものなので、相互変換を繰り返すべきではない。

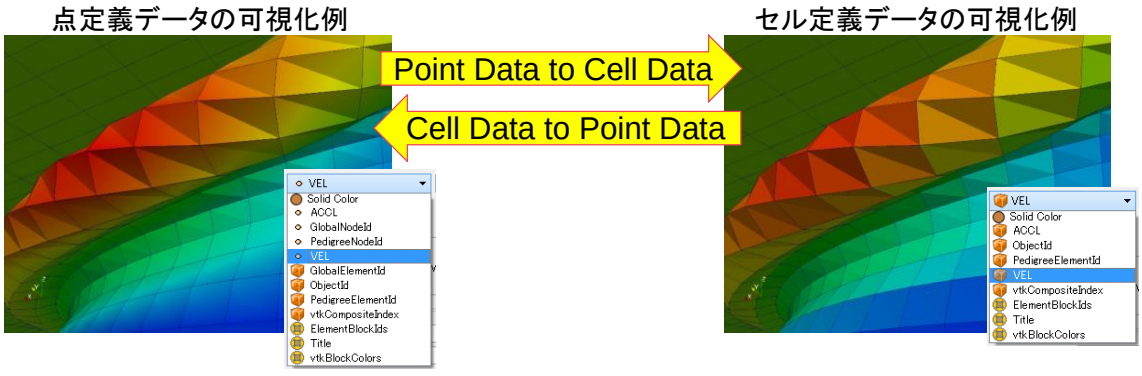


図 4.2.1-2　物理量定義位置の違いと変換

(2) ParaView で OBJ ファイルを出力する

ParaView 5.6 から OBJ ファイルを出力できるようになった。ただし、対象となるフィルタのアウトプットが Polygonal Mesh 形式またはその複合型の場合のみ有効である。Polygonal Mesh を生成する代表的なフィルタとしては表 4.2.1-2 のものがある。

表 4.2.1-2 Polygonal Mesh を生成する ParaView のフィルタ例

フィルタ名	機能
Extract Surface	格子（ブロック）の表面格子を抽出する
Slice	断面
ISO Surface	等値面
Stream Tracer	流線（Line 要素のみ＝太さが無い）

ParaView での可視化結果を OBJ 形式で出力しようとする場合、図 4.2.1-3 に示すように可視化パイプラインの最後のフィルタに ExtractSurface を用いると確実である。なお、Multi-Block の場合は、ブロック毎に OBJ ファイルが作られる。OBJ に出力するブロックを選択する場合は、Extract Block フィルタを用い、ブロックを統合する場合は、Merge Block フィルタを用いるとよい。

実際に可視化結果を OBJ 形式で出力するには、図 4.2.1-4 に示すように、File -> SaveData メニューから、出力先形式として Wavefront OBJ File Format (*.obj) を選ぶ。

なお、可視化元のデータが非定常データの場合は、タイムシリーズで出力することが可できる。また、この方法では色（マテリアル）は出力されないので、色情報つきの OBJ 形式としたい場合には、次項で説明する PLY 形式を経由した方法を用いるとよい。

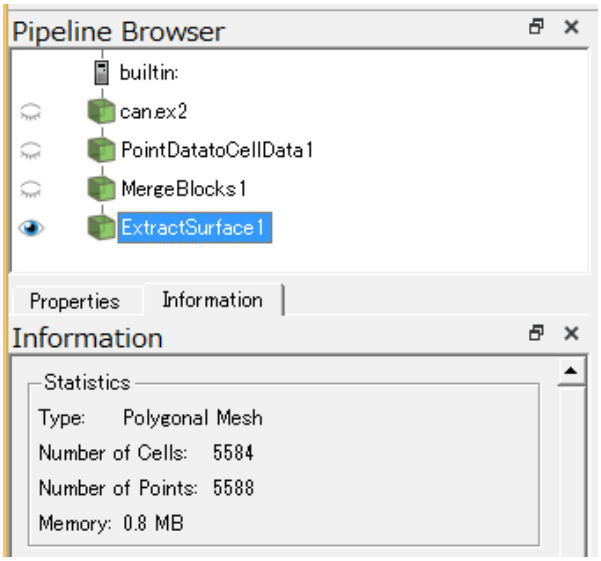


図 4.2.1-3　ParaView で ExtractSurface フィルタを使って Polygonal Mesh 形式を得る

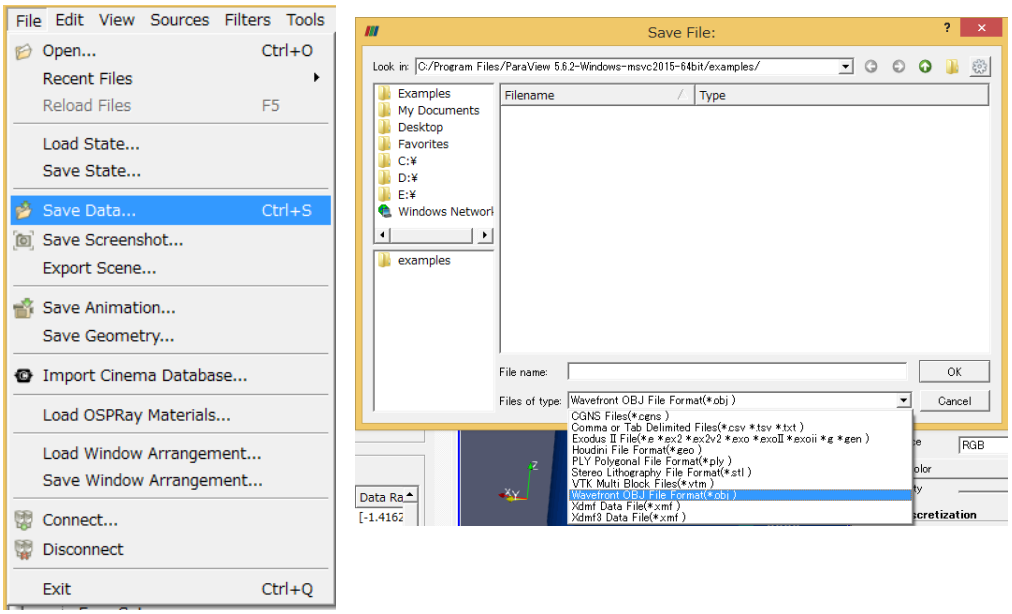


図 4.2.1-4　ParaView の可視化結果を OBJ 形式で出力する手順

(3) 色情報つきの OBJ ファイルを作成する

前述の方法では OBJ に色情報（マテリアル情報）は出力されない。色情報つきの OBJ ファイルを作るには、PLY 形式を経由した変換を行う方法がある。

① ParaView で、セル定義データに変換したうえで Polygonal File Format （.ply） に出力する

(a) Point Data to Cell Data フィルタを適用してセル定義データに変換

(b) Extract Surface フィルタで、表面ポリゴン（Polygonal Mesh）を抽出

(c) File -> Save Data… で PLY ファイルに出力する

（ここまでは OBJ 形式での出力と同様である）

(d) PLY 形式での保存時に図 4.2.1-5 のダイアログが出るので、必ず Enable Coloring をチェックする。

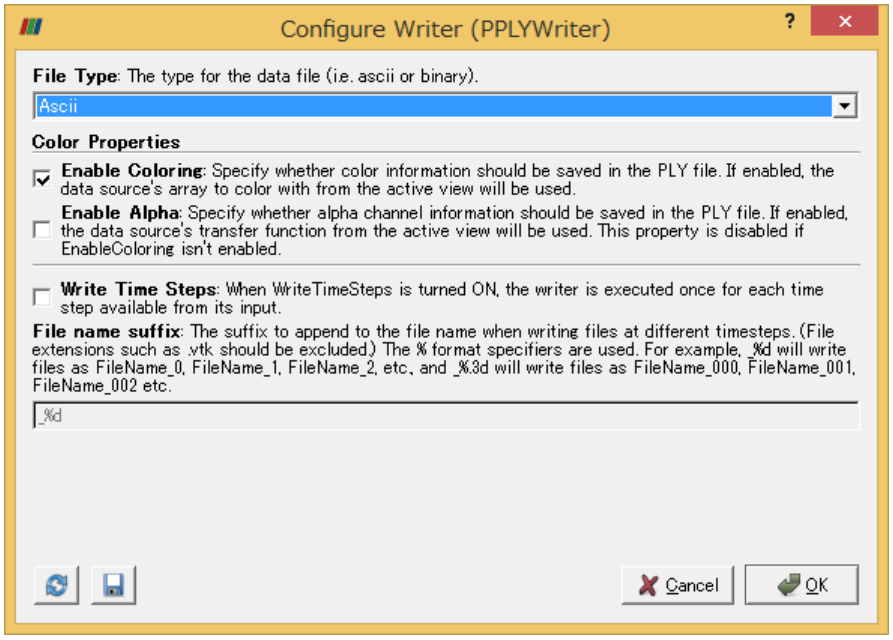


図 4.2.1-5　PLY 形式での出力方式の設定

② 汎用のメッシュデータ変換ツールである MeshLab を用いて、PLY 形式から OBJ 形式へ変換する。なお、ここではデータ形式の変換のみのため、MeshLab 上では特にデータの加工処理は不要である。

(a) File -> Import Mesh で PLY 形式のデータを読み込む（図 4.2.1-6）

(b) File -> Export Mesh As… で Wavefront Object （.obj） に出力する

(c) 図 4.2.1-7 の保存情報の選択ダイアログが現れるので、Face の Color を選択する（デフォルトで選択済み）

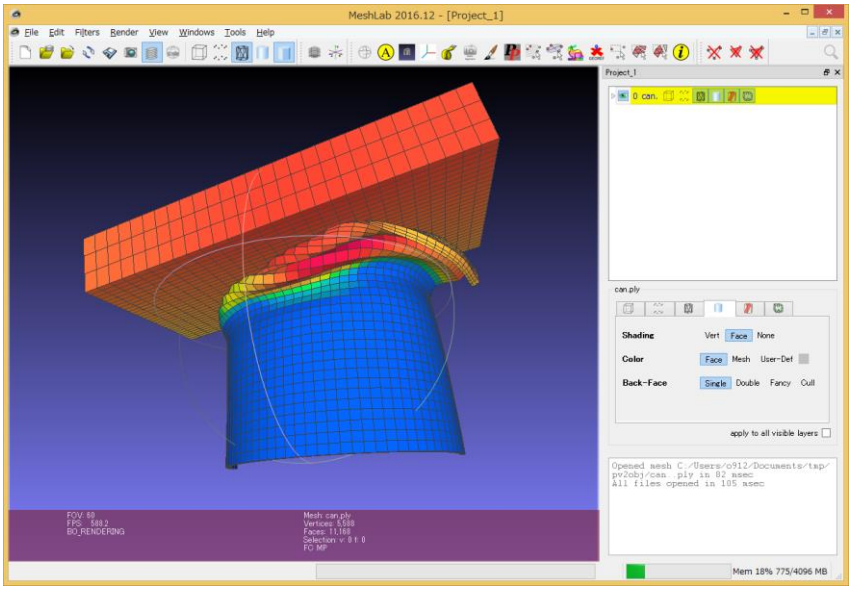


図 4.2.1-6　色付きの PLY データが MeshLab に読み込まれた

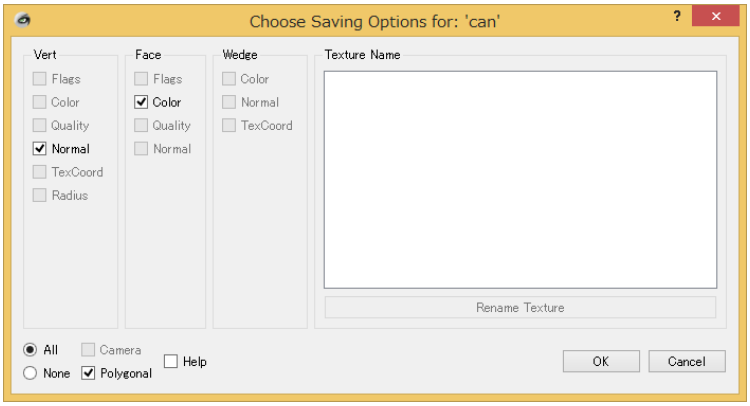


図 4.2.1-7　OBJ 形式保存時の保存情報の選択ダイアログ

③　色情報が記されたマテリアルファイル（.mtl）が一緒に出力される（図 4.2.1-8）

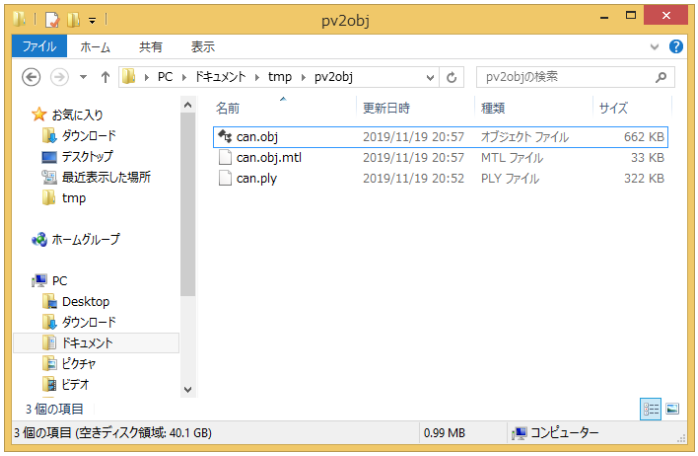


図 4.2.1-8　OBJ ファイルがマテリアルファイル（.mtl）と一緒に出力された

OBJ はマテリアル情報を面に持つので、面定義データが必要となる。MeshLab から出力されるマテリアルファイルには、色分解能ぶんのマテリアルが定義されている。例えば、ParaView での可視化が 256 色階調なら、256 マテリアルが定義される。これにより、Unity にインポートしたとき、図 4.2.1-9 に示すようにそのまま色を出せるようになる。

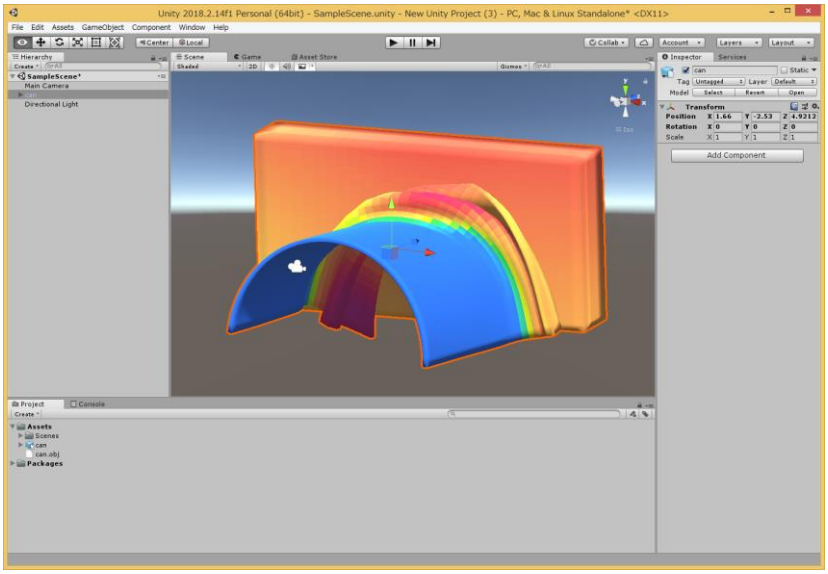


図 4.2.1-9　色情報つきの OBJ ファイルを Unity にインポートした結果

(4) 座標変換

ParaView で使われる座標系は CFD や計算科学などのシミュレーションで使われる、Y 軸が前後方向、Z 軸が上下方向の座標系であるのに対し、Unity は CG で使われる Y 軸が上下方向、Z 軸が前後方向の座標系を用いている。そのため、ParaView の可視化結果から作成した 3D オブジェクトを Unity にインポートする場合、座標変換が必要になる。

座標変換は、ParaView の可視化パイプラインで Transform フィルタを用いる方法（図 4.2.1-10）と、Unity にインポート後に行う方法（図 4.2.1-11）がある。いずれの場合も、X 軸で-90 度回転させることで変換を達成できる。

座標変換を ParaView で行うか、Unity で行うかはケースバイケースで選択することになる。たとえば、Unity に多くの同一モデルを多数配置する場合、それぞれについて座標変換を行うことは手間であるので、ParaView で事前に行ったほうがよい。

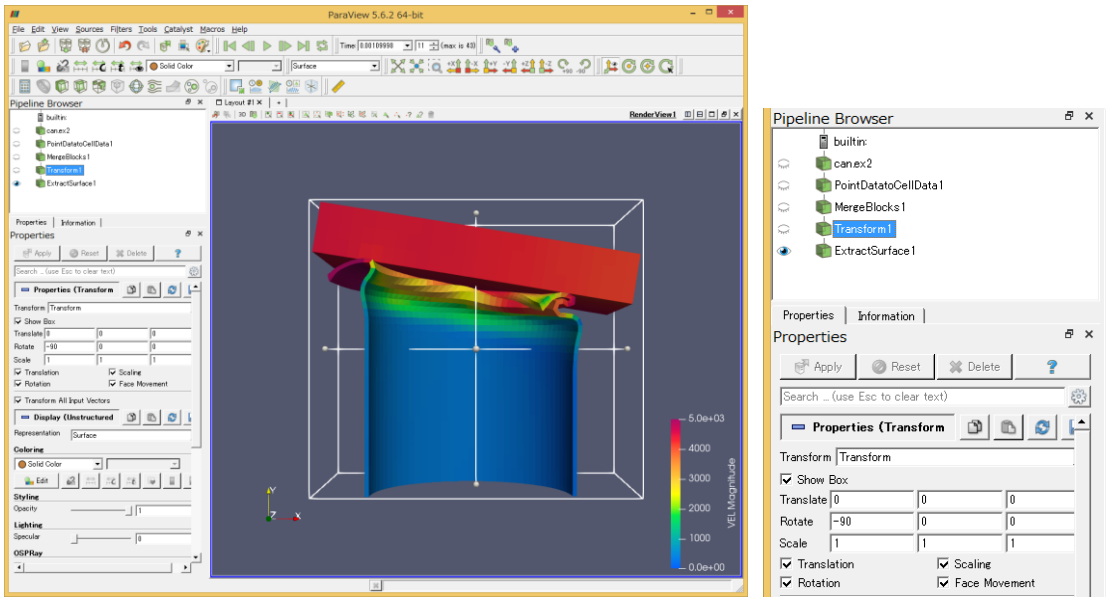


図 4.2.1-10　ParaView での座標変換

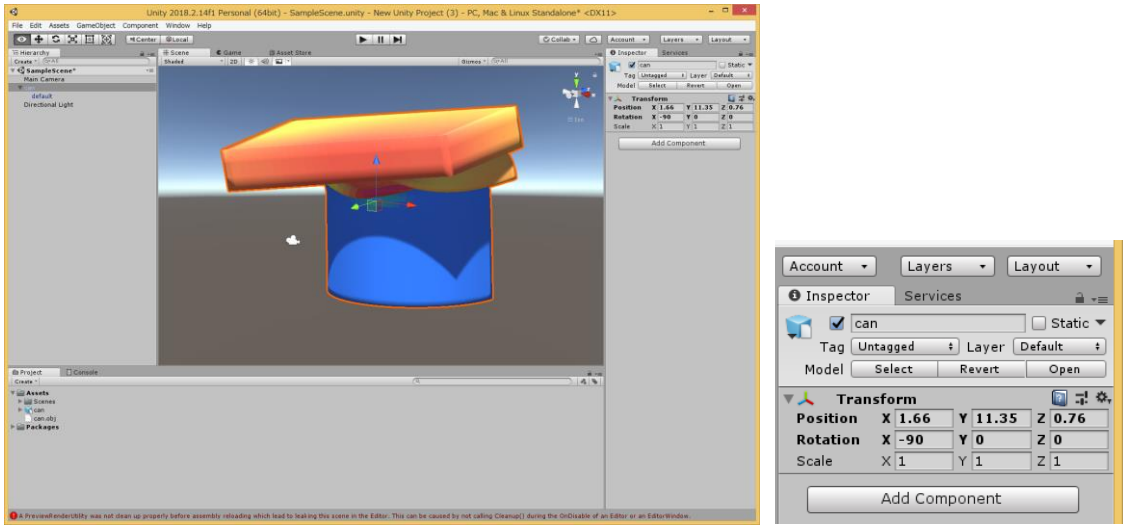


図 4.2.1-11　Unity での座標変換

4.2.2　AVS/Express

本節では、AVS/Express（以下 AVS と略）での可視化結果を Unity にインポートする方法を記す。

AVS には Unity で直接読み込める形式のデータを出力する機能が無いため、STL 形式でデータを出力し、これを Blender や MeshLab などの変換ツールでインポートし、DAE 形式でエクスポートしたファイルを Unity へ渡す流れになる（図 4.2.2-1）。

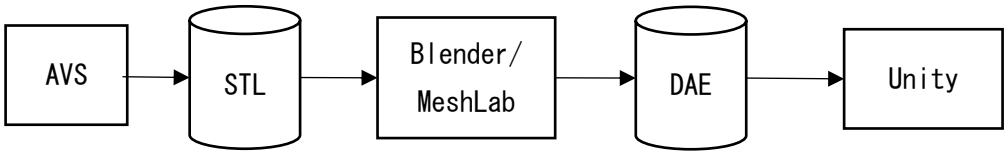


図 4.2.2-1　AVS から Unity へデータを渡す流れ

AVS で出力される STL 形式データはポリゴン形状のみで、物理量に応じてマップされる頂点カラーは出力されないことに留意されたい。AVS による UCD 形式の可視化結果の面データを STL 形式で出力する AVS ネットワークと可視化結果、並びに、出力された STL について MeshLab を介して Unity で表示した例を示す（図 4.2.2-2、図 4.2.2-3）。

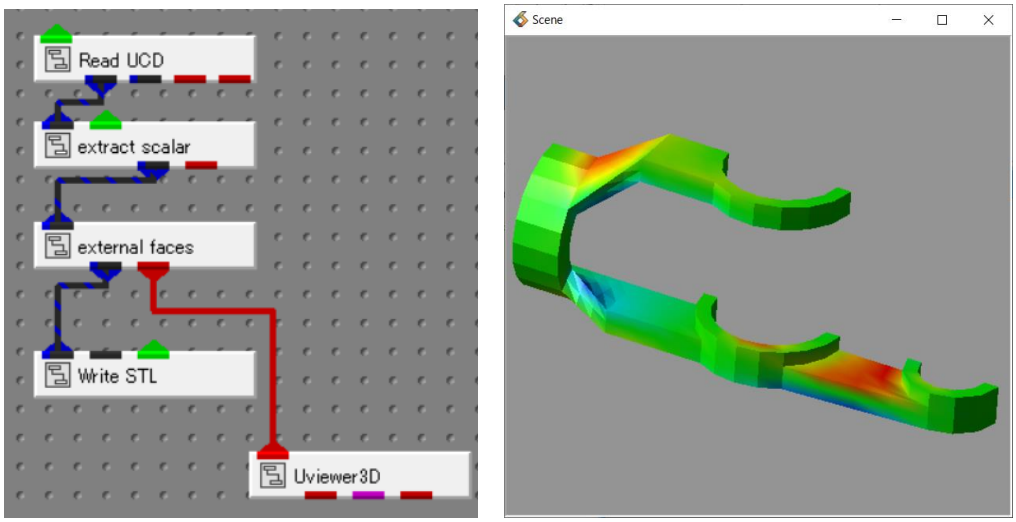


図 4.2.2-2　UCD の面データを STL で出力する AVS ネットワークと可視化例

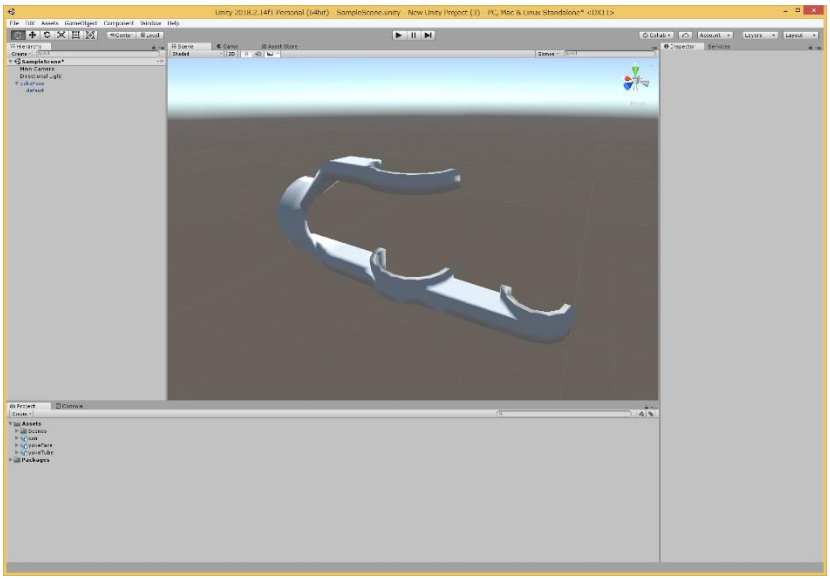


図 4.2.2-3　AVS の STL データの Unity による表示例

また、AVS で STL データとして出力できるのはポリゴン形状だけであるため、AVS で可視化したワイヤーフレームモデルを Unity に取り込みたい場合は、AVS の tube モジュールを用いてワイヤーフレームを構成する線分を円筒で表現すればよい。この AVS ネットワークと可視化結果、並びに、出力された STL について、MeshLab を介して Unity で表示した例を示す。(図 4.2.2-4、図 4.2.2-5)

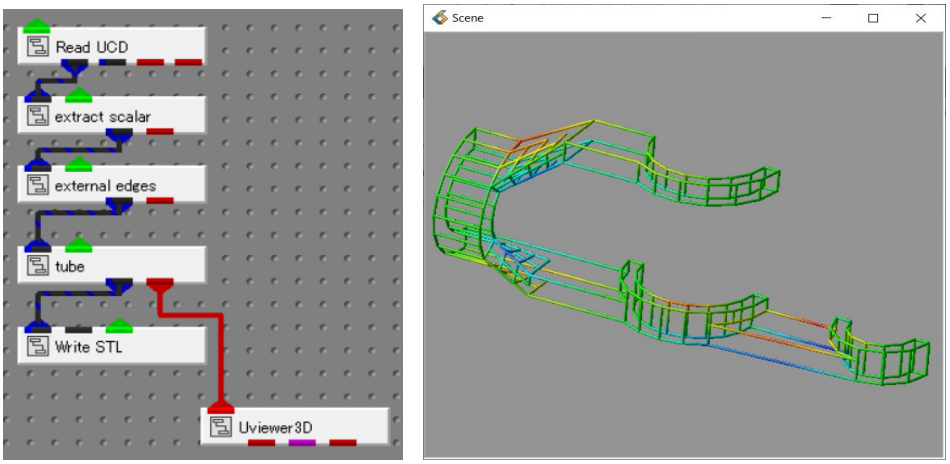


図 4.2.2-4　ワイヤーフレームを STL で出力する AVS ネットワークと可視化例

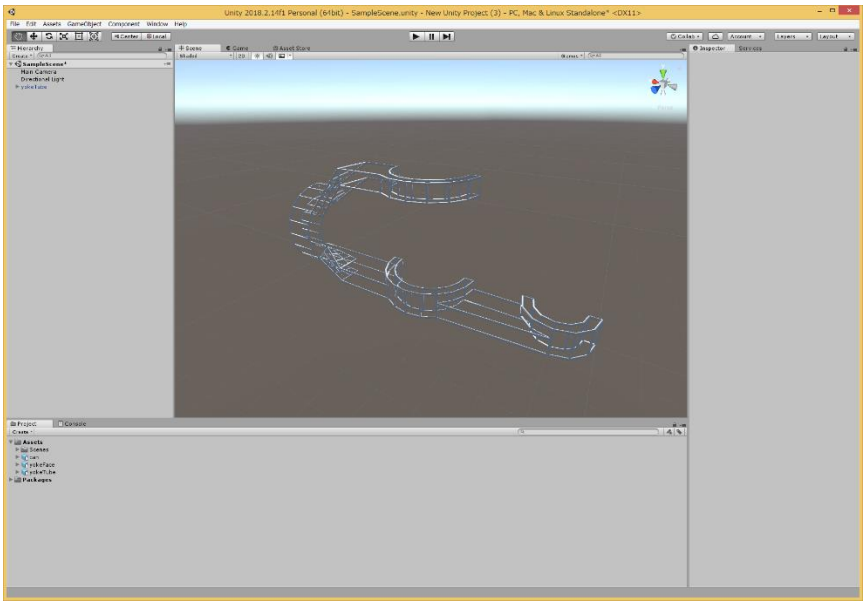


図 4.2.2-5　AVS のワイヤーフレームモデルの Unity による表示例

5.　新たな見せ方・新しい活用方法

ここまでは、計算機シミュレーションの結果を画像にする「オーソドックスな」可視化のノウハウを紹介した。しかし、可視化はシミュレーション研究のみならず、工学や生物学、医療、防災を含む人間社会科学など、様々な科学技術分野や教育現場などで活用されている。この章では以下の分野について、どのように可視化が活用されているかを紹介する（表 5-1)。

表 5-1　新たな見せ方・新しい活用方法

項番	分野	概略
5.1	文化財の可視化（長谷川）	有形文化財の保存手法の一つとして形状計測を行う事例が増えている。本節では大規模な点群データをポリゴン化なしに点群データのまま Unity 上で高速に可視化をおこなう。そのために、点群データに対する詳細度制御を提案し、シェーダを用いて高速可視化の実現を行った。
5.2	数学分野での可視化（北葁）	純粋数学や数学教育において VR を含めた可視化を用いて数学を理解しようとしたり、教育に役立てたり、新しい表現を得ようとする試みの国内外の動きを紹介する。
5.3	音と組み合わせた可視化（大野）	CAVE を利用した高齢者の認知に関する実験を紹介する。本実験では、主に CAVE の正面スクリーンを用いて、前方からこちらへ走行する自動車を表示する。このとき、臨場感を出すために、ステレオスピーカで、走行音を提示している。
5.4	運動機能計測での MR の利用（野田）	高齢者に対して歩行時の障害物回避動作を計測する際、実物の障害物に躓き転倒をするリスクが想定される。このリスクを回避するために MR を利用した表示システムの開発と適用について紹介する。

5.1 文化財の可視化

立命館大学　長谷川恭子

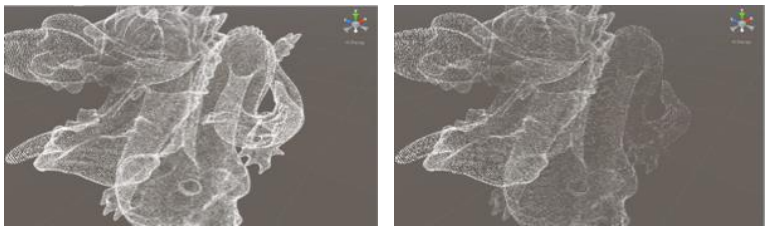
有形文化財を後世に残す取り組みの一つとして、有形文化財をデジタルデータとして保存して活用するデジタルアーカイブが世界各地で作成されている。近年では、デジタルデータとして保存するために、詳細な形状や色まで計測できる 3 次元計測技術を用いた文化財のデジタルデータ化が行われており、計測した 3 次元計測点群を使ったデジタルアーカイブが注目されている。一方で、文化財のデジタルアーカイブには、使用の容易さと機能の充実性から、ゲームエンジンを利用したものが多い。全天球カメラを用いて撮影された風景をゲームエンジンでヘッドマウントディスプレイを用いて体験させる事例は多く見られる。Unity は 3 次元計測点群を読み込み、ポリゴンのオブジェクトと同様に扱うことのできるプラグインが用意されていることもあり、その利用が増えている。しかしながら、3 次元計測点群は数億点にもなる大規模な点群になることが多いため、Unity 上でリアルタイムコンテンツを実現するには十分な描画速度を保つことが困難である。

本節では、大規模 3 次元計測点群を Unity に読み込んだ際のレンダリング処理の高速化実現のため、大規模 3 次元計測点群のための詳細度制御（Level of Detail; LOD）技術を提案する。LOD はカメラからの距離に応じてモデルの詳細度を変更する、CG の基礎技術の一つである。例えば、たくさんの高詳細度のポリゴンモデルを作成し、全てを同じ空間に配置した世界を、カメラを操作して眺めていくコンテンツを考える。その場合、カメラから十分離れているオブジェクトに関しては、どれだけ詳細なポリゴンモデルでも描画処理に時間がかかる割に、その詳細さはレンダリング結果には大きく影響しない。そのため、カメラから離れているポリゴンモデルに関しては、詳細度を下げたポリゴンモデルに入れ替えることで描画処理を高速化することができる。Unity の機能の中には LOD 機能が標準的に存在する。Unity の LOD 機能はそれぞれのオブジェクトごとに詳細度の違うモデルを用意してカメラから離れているものは詳細度の低いモデルに入れ替えることで LOD を実現している。点群に対しても、同じようにオブジェクトごとに詳細度の違う点群オブジェクトを用意し、カメラからの距離によって、詳細度の違うモデルを切り替えることで LOD を実現できる。しかし、点群に対して「一つのオブジェクト」という単位付けは困難であり、例えば数百点のまとまりを一つの点群オブジェクトとして定義しても、分割された点群オブジェクトごとに詳細度の制御を行うために欠損が描画結果に表れてしまうことが多い。 点群に関する LOD の提案には八分木を用いた手法があるが、ここではシェーダを用いて 3 次元計測点群に適した LOD を実装し、高速化を実現する。ジオメトリシェーダで描画確率によって点を減らす処理を行うことで描画しないという処理を実現している。また、ジオメトリシェーダで点を削減することで必要な点だけを次のシェーダ処理に渡すことができる。このように無駄な処理を省くことで、描画処理の高速化を実現している。

カメラ位置に依存した点群データの LOD を適用した結果を図 5.1-1 に示す。視点から点群までの距離に対して、詳細度制御を開始する距離 min とこれ以上は描画しない距離 max を

決め、min から max までの距離にある全ての点それぞれに対して描画確率を与え、カメラからの距離が遠いと低い描画確率、近いと高い描画確率となるようにした。これにより、カメラからの距離が近ければ描画される点数が多いため高詳細度になり、遠ければ低詳細度になる。これにより、シェーダで点群データに対する LOD 技術の実装ができる。

図 5.1-2 では、京都にある新町通りの 3 次元計測点群（126,861,748 点）を用いて、Unity に備わっているポリゴン用の LOD 機能を従来手法として、描画結果の比較と FPS の計測実験を行った。実験は街並みの中を通り過ぎるようにカメラを移動させ、その際に変化する fps と描画結果を計測するという手順で計測した。描画結果の比較を図 5.1-2 に示す。同図(a)の従来手法では点群の欠損があるが、同図(b)の提案手法では詳細度コントロールが正しく行えていることがわかる。また、LOD を適用したときの描画速度は安定して 60fps が実現できている。



(a) 全ての点群データ　　(b) LOD を使用した点群データ

図 5.1-1　点群データ高速描画のための LOD の適用

(a)従来の Unity の LOD　　(b)提案のカメラ位置に依存した LOD

図 5.1-2　LOD の適応の違いによる描画結果の比較

5.2 数学・数学教育分野での可視化について

広島大学大学院教育学研究科　北葔如法

2.3 で述べたように、筆者は数学・数学教育分野での可視化に興味を持つ。 このワーキンググループでの多くのメンバーの事例にあるシミュレーションのサイエンティフィック可視化で行われているスカラー場やベクトル場の可視化は、そのまま数学（ベクトル解析など）の学習に役立つと思われる。 ここではそれ以外の数学・数学教育の新たな見せ方・新しい活用方法を述べたい。

まず、数学分野での可視化について、現代数学の可視化として、1.3 で触れたブラウン大学の ICERM での Illustrating Mathematics 行われた研究集会 Illustrating Geometry and Topology (Sep 16 - 20, 2019) https://icerm.brown.edu/programs/sp-f19/w1/ という研究集会で紹介された事例を一つ紹介しよう。 Segerman らの研究 https://arxiv.org/abs/1702.04004, https://arxiv.org/abs/1702.04862 で、双曲空間を VR で実現し、 HMD を装着して動き回ることでホロノミーという幾何学的な情報を実感できるというものである。 数学的な現象を、実感することができうるというところが VR の面白さの一つであろう。 現代幾何学で現れる概念として、実際に我々の普通に生活しているユークリッド空間とは少し違う空間を VR で実現してしまうと、純粋数学的な現象が「体験」できてしまうという興味深い事例である。

また 2019 年 9 月 25 日に東京大学で開催された ミーティング「数学の可視化と Virtual Reality」では、純粋数学を中心に様々な方面から情報提供された。 中でも 九大数理の石井豊氏は「4 次元可視化プロジェクト」と題された講演の中で、 4 次元を認識出来るように可視化しようという試みを披露された。 4 次元を、2 次元（通常の網膜への射影）＋両眼視差（BD）1 次元＋運動視差（MP）1 次元と分解して 4 次元を知覚しようという意欲的な研究である。 これは可視化を通して人間の知覚そのものを拡張しようという新しい見せ方であり、非常に今後の進展が楽しみである。

数学教育分野における筆者による取り組みを第 7 節の事例集（7.2）に述べた。 3 次元の数学的な対象を可視化し、理解の補助にしようというものが主である。 単純に 3DCG のアニメーションというノンインタラクティブなものから、 iPad と Mac を連携したインタラクティブなもの、 AR、VR を活用するものと、 徐々にハードウェアの選択が多くなってきて可能性が広がってきた。

これらから言えるのは、特にインタラクティブなものや、AR、VR となると、 物質で模型を作っただけではできない表現が多々可能になるということである。 拡大縮小が自由であるし、インタラクションを作ることもできるし、ある意味で触ることができるし、 VR では構造を無視して実際に顔をオブジェクトの中に入れることもできる。模型ではできないことだ。 事例に挙げたものを、さらにそのようなインタラクティブに試行錯誤できるものに改良し、 これらのメリットをうまく利用して数学教育に役立てることを望んでいる。

数学の学びというのは古来より紙と鉛筆と思考であった。 ここに近年は計算機実験がプラ

スされている。 さらにこれに AR、 VR などによる可視化が加わり、 これまで、紙の上や 3DCG によって、仮に見えたとしてもよくわからなかったり理解がしづらかったり、 紙の上で理解できる人だけが理解し楽しんでいた状況が解放され、 より選択肢の多い、豊かな数学の学びが訪れることを願う。

5.3 音と組み合わせた可視化

兵庫県立大学　大野暢亮

CAVE を利用して、高齢者の交通事故に関する研究を兵庫県立大学環境人間学部、甲南大学知能情報学部と共同研究を行っている。高齢者の認知機能の低下を定量的に捉えることが目的である。

高齢者の交通事故は年々増加傾向にあり、原因の追究は急務であると言える。しかしながら、実地の実験を行うことは、危険を伴うことに加えて、全く同じ条件の下で実験を繰り返し行うことは不可能である。そこで、兵庫県立大学の所有する CAVE 型のバーチャルリアリティ装置を利用して、安全に、かつ同じ条件下で実験を行なっている。

実験の概要は、発表前であるので詳細はまだ記述できないが、あらかじめ参加者に説明した状況のもと、街並みを再現した VR 空間内で、複数の条件下で前方から自動車を走行させ調査を行う（図 5.3-1 参照)。参加者の作業は、映像を見てワンドのボタンを押すことだけである。

この実験を高齢者（前期および後期）と若年者を対象として行い、若年者と高齢者の結果を比較することで、認知力の低下を調査する。

VR 空間内を走行する自動車は、走行音を発している。遠くから近づくにつれ、走行音の音量が上がる。走行音を付加することにより、映像のみの場合よりも臨場感が大幅に増す。音の出力には、マルチプラットフォームの OpenAL ライブラリとステレオスピーカを利用している。本実験では、自動車が前方から走行してくるだけなので、3 次元的な音響装置までは必要としていないが、コンテンツによっては、3 次元音響が必要になると思われる。なおこの実験は、兵庫県立大学の倫理委員会の承認を得て行なっている。

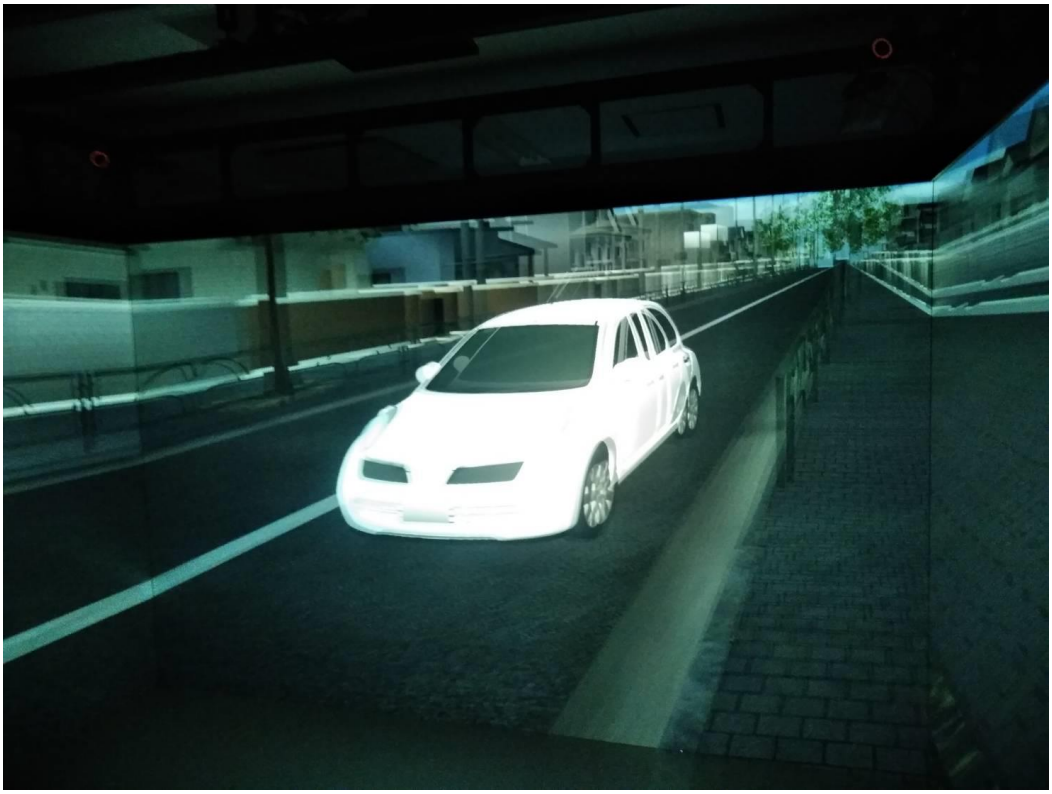


図 5.3-1　CAVE 内を走行する自動車

5.4 運動機能計測での MR の利用

理化学研究所　野田茂穂

高齢者を対象とした運動機能の計測として、人が歩行時に障害物を回避する動作を計測している。実在の障害物を設置すると、本当に躓いて怪我をする恐れがある。計測環境としてはこのような危険を回避する必要があり、課題が多い。そこで VR 技術を用いて、仮想的な障害物を被験者に表示することで躓く危険性を回避している。

高齢者を対象とした場合、HMD での VR 表示は違和感を伴い、歩くのが難しいという意見があり、Mixed Reality を利用し実空間の映像を表示することとしている。Mixed Reality のシステムとして、シースルー型とビデオスルー型が存在する。開発当時、シースルー型のシステムでは、上下方向の視野角が狭く、障害物の表示にはそぐわないため、ビデオスルー型のシステムを採用した。

図 5.4-1 にビデオスルー型 Mixed Reality 装置を示す。HMD デバイスとして Oculus Rift を使用し、ビデオスルーを実現するためのステレオビデオカメラとして Ovrvision Pro を HMD 前面に設置している。

図 5.4-1　ビデオスルー型 Mixed Reality 装置

表示システムのソフトウェア開発環境として Unity を使用している。Ovrvision Pro の開発環境も Unity むけにリリースされている。障害物の表示場所は、AR マーカを用いて表示位置を指示する。その後、歩行に邪魔となるため、AR マーカを使用せず、指示した位置に障害物を表示し続けるシステムとした。図 5.4-2 に表示例を示す。ビデオカメラで取得した映像に AR マーカで指示した障害物を表示し、床面に影を投影することで視覚的な違和感を少なくしている。

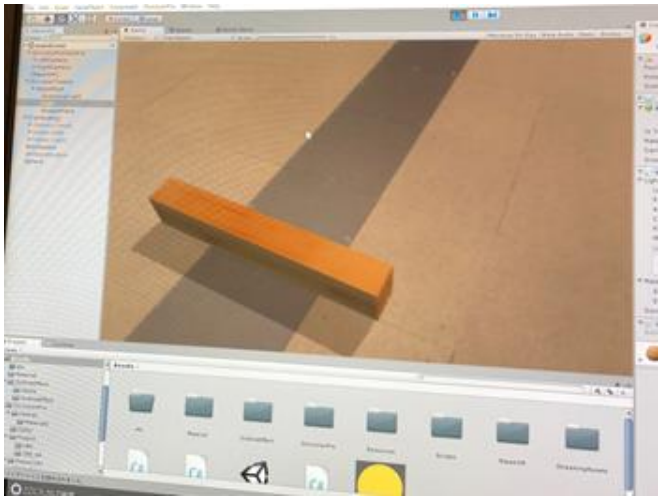


図 5.4-2　障害物の表示例

参考までに図 5.4-3 に計測環境を示す。Oculus Rift の位置センサを 4 台設置することで、5m 歩行の計測範囲を確保している。この位置センサは光学式モーションキャプチャと共存が可能である。

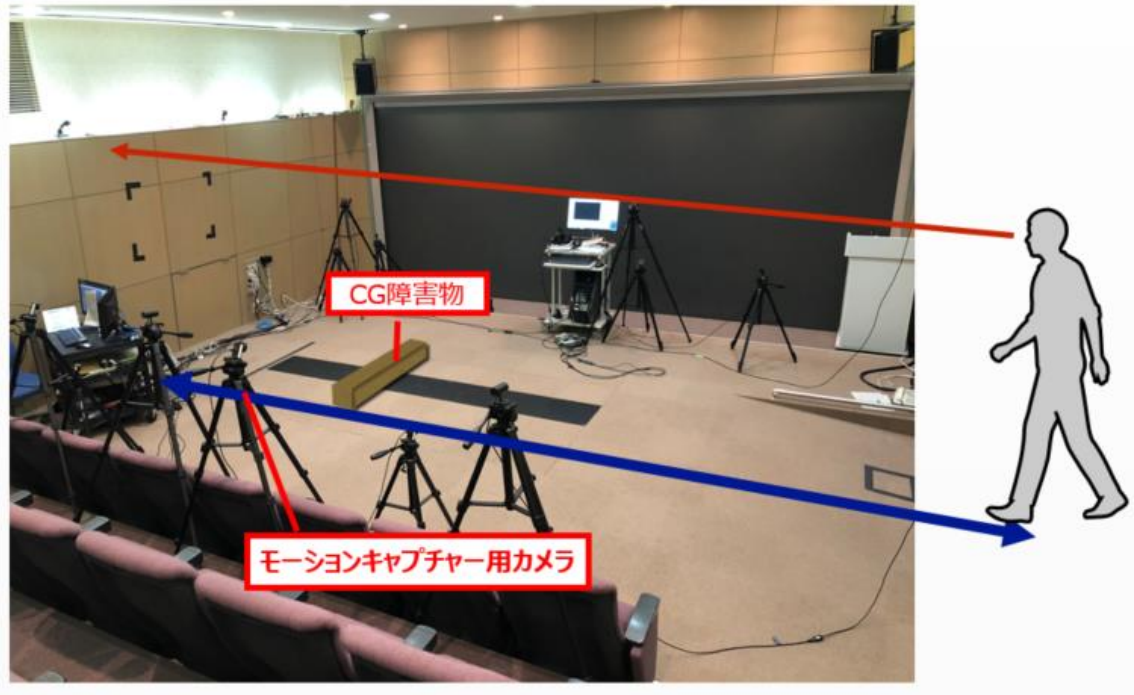


図 5.4-3　計測環境

6.　現状のデバイス紹介及び目的に合致したデバイスの選び方

6.1 概略

理化学研究所　野田茂穂

VR 表示装置である HMD システムは多くの商品がリリースされている。表 6.1-1 は 2016 年時点での比較的入手可能な VR 用 HMD のスペックと価格を調べたものである。手軽な HMD システムとして、ハコスコ VR と呼ばれているスマートフォンのディスプレイを利用したものがあるが、この表では対象外とした。

VR-HMD の性能要件として、解像度・画角視野角・FPS・重量・位置センサ形式を取り上げ、大体の入手価格を記載した。表示性能として、解像度・画角視野角・FPS が重要である。また、人の移動と同期をとった VR 空間の移動を行う場合、位置センサが重要となってくる。

Canon の MREAL は非常に高額であるが、HMD 本体だけでなく、表示システムであるソフトウェアも含んだ価格であり、CAD データなどを直接取り扱うことができる。

それ以外のものは、Unity や Unreal といった CG ソフトウェア作成環境を用いて開発する必要があり、その費用は含まれていない。

安価に入手可能なものでは、解像度が高く視野角も広い、Oculus Rift、 HTC VIVE が良いと判断できる。両者とも、ソフトウェアの開発環境として Unity、 Unreal が使用でき、開発環境のサポート能力も高い。注意すべきは、頻繁にアップデートされ、下位互換が保証されないことがあるため、最新のシステムにアップデートする時には事前に調査をしておくべきであろう。

Oculus、 Vive 共に HMD 内蔵の IMU にて視点方向の情報を取得することができ、HMD の位置は外部センサ（赤外線マーカなど）を使用する。この外部センサを複数配置することで、ルームスケール（5m * 5m 程度の空間）でのポジショントラッキングが可能となる。

Windows Mixed Reality Headset は Microsoft の規格に則り各社からリリースされており、視野角が劣るものの安価に利用できる。

Microsoft HoloLens、Metavision Meta2 はシースルー型の Mixed Reality を可能とするデバイスであるが、2019 年現在入手が困難である。

また、2019 年にリリースされた Varjo の“VR-1”は限られた表示エリアに対して人の目と同等な解像度の映像を実現している。今後もより高解像度なものがリリースされてくると思われるが、表示を担う PC のスペックへの要求が高くなり、最新のグラフィックカードの搭載が必須になると思われる。

表 6.1-1　VR 用 HMD のスペックと価格（2016 年時点）

メーカー	商品名	解像度	画角 視野角	FPS	重量	価格	位置センサ	用途
Canon	MREAL	1920*1200	68*60*40	54	1040g	900 万円から	vicon 使用	VR/MR
Microsoft	HoloLens		30*17.5			50 万円	IMU	MR HMD
Oculus	Rift	2160*1200	110	90	440g	10 万円（5 万円）	IMU＋位置センサ	VR HMD
HTC	VIVE	2160*1200	110	90	620g	10 万円	IMU+位置センサ	VR HMD
Acer, HP, ASUS, Dell, Lenovo	Windows Mixed Reality Headset	2880*1440	95	90		4 万円程度	インサイド方式位置	VR HMD
SONY	PlayStation VR	1920*1080	100	120, 90	610g			VR HMD
Metavision	Meta 2	2560*1440	90	60	500g	20 万円		MR HMD
Vrvana	Totem	2540*1440	120	90			IMU センサ	MR HMD
AMD	Sulon Q	2560*1440	110	90				VR/MR/AR
Razer	OSVR2	2160*1200	110	90				VR HMD
Magicleap	不明	不明	不明	不明	不明	不明		MR HMD
FOVE	FOVE 0	2560*1440	100	60	520g	$599.00	IMU + 赤外線ポジショントラッキング Camera sensor	VR HMD
Immerex	VRG-9020	1920*1080			200g			VR HMD

なお、最新の情報は、Mogura VR　https://www.moguravr.com　に比較的良質な情報が記載されている。

6.2 主要な HMD の特徴（と目的に合致した選択方法）

国立研究開発法人海洋研究開発機構　川原慎太郎

現在発売されている HMD は、PC 接続型のものと、スタンドアロン型の 2 種類に大別される。使用するデバイスを選択する際には、それらのハードウェア的な機能面だけでなく、目的や利用のシーンといった用途を考慮する必要がある。以下、現在主要な HMD についてタイプ別に紹介する。

6.2.1　PC 接続型 HMD

その名の通り、PC に接続して使用するタイプの HMD である。アプリケーションの実行は HMD を接続した PC 上で行われ、PC には高いハードウェアスペック（特にグラフィックス性能）が要求される。また、HMD の種類によって必要となる機器接続用の外部端子数や種類（HDMI/Display Port、USB2.0/3.0）が異なるため、HMD と併せて PC を購入する場合には注意が必要である。OS については、Microsoft Windows しか選択肢は無いのが現状である。PC および HMD の間の接続は基本的に有線接続となるため、現実空間内での歩行を伴う移動が可能な範囲はそのケーブル長に（外部センサが必要な HMD の場合はセンサが検出可能な範囲にも）依存する。以下、主要な機器について述べるとともに、それらのハードウェアスペックについては本項の最後に表 6.2.1-1 としてまとめた。また、各 HMD とそれらに対応するコントローラの外観について図 6.2.1-1 に掲載する。本図中には各コントローラを持った際の様子も併せて掲載した。機種によりその形状が大きく異なり、それぞれに特徴があることがよくわかるだろう。ジョイスティックやタッチパッドの有無、使用可能なボタンの数等も機種毎に異なるので、機種選定時に確認しておくとよいだろう。

6.2.1.1 Oculus 社製 HMD（Rift, Rift S）

現在の VR ブームの先駆けとなった 2013 年の Development Kit 1（DK1）、2014 年の Development Kit 2（DK2）は開発者向け HMD であったが、2016 年にコンシューマ向けに一般発売されたのが Rift である。DK1 ではヘッドセット内蔵センサによる 3DoF でのヘッドトラッキングのみ、DK2 ではヘッドセット内蔵センサと外部カメラによるアウトサイドイン方式を組み合わせた 6DoF でのヘッドトラッキングと進化を続け、Rift では外部カメラによる 6DoF での手持ちコントローラ（Oculus Touch）2 台のトラッキングも可能となった。2019 年 5 月に発売された Rift S では、ヘッドセット本体に内蔵されたカメラによるインサイドアウト方式でのトラッキングにより、Rift では必要だった外部センサカメラは不要となっている。本稿執筆時において、Rift S の発売により Rift の販売は既に終了しているため、現在購入可能な PC 接続型の Oculus 社製 HMD は Rift S のみとなる。Rift、Rift S ともに、アプリケーションは専用のプラットフォームソフトウェア（Oculus App）を介して実行される。

6.2.1.2 HTC 社製 HMD（VIVE, VIVE Pro）

2016 年に HTC 社が発売した PC 接続型 HMD が VIVE である。ヘッドセット内蔵センサと外部センサでのアウトサイドイン方式による、ヘッドセットおよび手持ちコントローラ 2 台の 6DoF でのトラッキングが可能となっている。2018 年に VIVE のアップデート版として発売された VIVE Pro では高解像度化が、2019 年に発売された VIVE Pro Eye ではアイトラッキング機能が追加された。2019 年 10 月に発売される VIVE COSMOS では外部センサが不要となり、ヘッドセット内蔵カメラによるインサイドアウト方式での 6DOF トラッキングが可能となる。VIVE、VIVE Pro ともに、アプリケーションは専用のプラットフォームソフトウェア（SteamVR）を介して実行される。

6.2.1.3 Windows Mixed Reality ヘッドセット

特定のベンダー製の HMD を指すものではなく、Microsoft の作成したハードウェア基準（Windows Mixed Reality 規格）に適合する HMD 全般を指すものである。“Mixed Reality”と名称には入っているが、MR 機能の無い通常の VR ヘッドセットである。HP、DELL、Acer、ASUS、富士通、Lenovo、Samsung の各社が同規格にて HMD を発売しており、いずれの製品もヘッドセット内蔵センサと内蔵カメラでのインサイドアウト方式による、ヘッドセットおよび手持ちコントローラ 2 台の 6DoF でのトラッキングが可能な点は共通である。また、各社製ともコントローラは同一のものであるが、ヘッドセットについては装着感等においてベンダー毎に特徴が異なるようである。アプリケーションは専用のプラットフォームソフトウェア（Mixed Reality Portal）を介して実行される。

(a) Rift

(b) Rift S

(c) VIVE

(d) VIVE Pro

(e) Window Mixed Reality ヘッドセット

図 6.2.1-1　主要な PC 接続型 HMD と各機器に対応したコントローラの外観

表 6.2.1-1　主要な PC 接続型 HMD のハードウェアスペック

機器名称	Rift	Rift S	VIVE	VIVE Pro	Windows MR
片眼解像度	1080×1200	1280×1440	1440×1440	1440×1600	1080×1200
リフレッシュレート	90Hz	80Hz	90Hz	90Hz	90Hz
視野角	110°	110°	110°	110°	110°
トラッキング	6DoF	6DoF	6DoF	6DoF	6DoF
PC との接続	HDMI×1 USB3.0×3	HDMI×1 USB3.0×1	HDMI×1 USB3.0×1	DP×1 USB-C 3.0	HDMI×1 USB3.0×1
外部電源	不要	不要	要	要	不要
外部センサ	要	不要	要	要	不要

6.2.2 スタンドアロン型 HMD

前述した PC 接続型 HMD とは異なり、機器単体での使用が可能な HMD である。ヘッドセットに内蔵された OS（Android）上にアプリケーションをインストールし、機器装着時にコントローラ操作によりアプリケーションを選択、実行する。スタンドアロン型 HMD は機器単体での使用が可能なため手軽に扱えるという利点があるが、内蔵された Android での処理となるため、高負荷のアプリケーションや、大容量データを扱うには不向きであると言える。また、現在主要な PC 接続型 HMD がヘッドセット、コントローラともに 6DoF であるのに対し、ヘッドセットのみでも 6DoF に対応した機器が少ないという点もアプリケーション開発の際には考慮する必要がある。主要なスタンドアロン型 HMD のハードウェアスペックについて表 6.2.2-1 にまとめるとともに、それらの外観について図 6.2.2-1 に示す。

表 6.2.2-1　主要なスタンドアロン型 HMD のハードウェアスペック

機器名称	Oculus Go	Oculus Quest	VIVE Focus
片眼解像度	1280×1440	1280×1440	1440×1600
リフレッシュレート	60/72Hz	80Hz	75Hz
視野角	95°	95°	110°
トラッキング	3DoF	6DoF	6DoF

(a) Oculus Go　　(b) Oculus Quest　　(c) VIVE Focus

図 6.2.2-1　主要なスタンドアロン型 HMD の外観

7.　コンテンツを作成する際の課題

7.1 表現したいコンテンツの性質を考慮した作成する際の課題

甲南大学　田村祐一

7.1.1　コンテンツ表現手法

VR を利用した可視化コンテンツを作成するにあたり，どの程度の情報・体験をユーザーに対して提示するかを決定することが最も重要である。例えば、一般的な 2 次元グラフによる表現で十分な情報について、過剰な可視化手法を使用することは、体験のクオリティを下げる可能性、特別な機器を必要とすることで、情報共有が難しくなる等の問題が生じる。そこで、本節ではコンテンツ表現に使用可能な手法を自由度と没入感の観点から考えることとする。図 7.1.1-1 に VR を利用した可視化手法から見た従来手法の位置づけを示す。

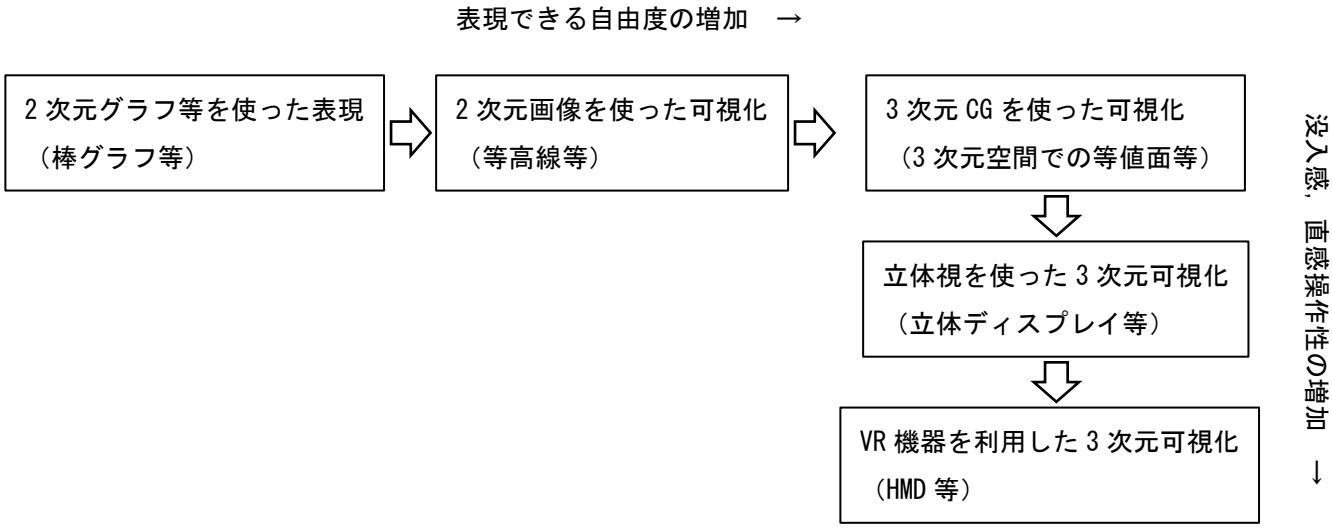


図 7.1.1-1　VR を利用した 3 次元可視化と従来手法の関係

VR 可視化は没入感が高く、直感的に操作が可能ということもあり、広く用いられるようになってきているが、数値情報等の可視化を考えた場合には、表現できる自由度自体は変わらないことに注意が必要である。同様に単純な立体視に関しても表現可能な自由度は同じである。一方で、視点を自由に変えられることや奥行き知覚が容易であることなどから、直感的に構造を把握することに適している。逆に、提示できるデバイスが現時点ではまだ広く普及していないという問題がある。これらのことを考慮にいれ、コンテンツ作成者は適切な可視化手法を選択することが求められる。

7.1.2　コンテンツとのインターフェイスの構築

次に、作成したコンテンツについて、ユーザーにどのように見せるかについて検討する必要がある。図 7.1.2-1 にコンテンツとユーザーとのインターフェイスについてまとめる。ユーザーがコンテンツに対して変更できる情報が少ないインターフェイスから多いものについて並べている。コンテンツをユーザーに提示するのみの場合、特別なインターフェイスの設計は必要ない。次に、最低限の操作機能を付加する場合、HMD 等を装着せず、従来型の入力デバイスを利用する場合は問題が生じないが、HMD 等を利用する場合には、十分に直感的なインターフェイスの設計が求められる。特に、2 次元画面での操作をそのまま流用することはユーザー体験の質を下げることになる。どのようなインターフェイスがよいかについての知見は、2 次元でのインターフェイスと比較して少なく、現在のところ様々なものが提案されているところである。最後にバーチャルな対象物と直接的な相互作用を検討する場合についてである。このタイプのインターフェイスは図 7.1.2-1 における 3 次元 CG、立体視、VR のシステムでのみ基本的に利用されることとなる。直接的な手の操作を利用する一つの利点として、自らの手の大きさとバーチャルな物体との比較や、手を伸ばしたときの距離感により、バーチャルな空間に表現されている物体の大きさを無意識に把握することで、厳密に理解できるなどの効果が考えられる。一方、現状の VR 機器で正確な大きさを提示することが難しく、例えば HMD を利用する際には個人差のある瞳孔間距離（IPD）を適切に設定することが不可欠である。

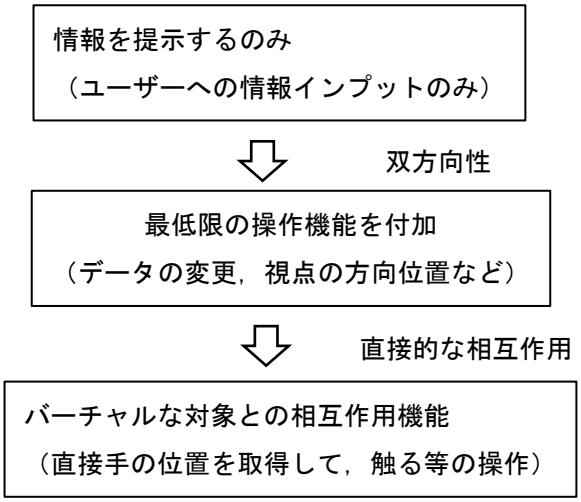


図 7.1.2-1　表示するコンテンツと利用するインターフェイスの関係

7.2 Unity で可視化コンテンツを作成するときの課題

東京都市大学　宮地英生

可視化ソフトウェアとゲームエンジンは開発思想が異なるので 4.1　ソフトウェア開発方法で述べたように二つのソフトウェアの連携には課題が生じる。ここではゲームエンジン Unity にデータフロー型の可視化ソフトウェア AVS/Express に近いフレームワークを構築し、そこに可視化アセットを C#で記述することを試みた。以下、その実装方法と課題について述べる。

7.2.1　頂点の生成と色の割り当て

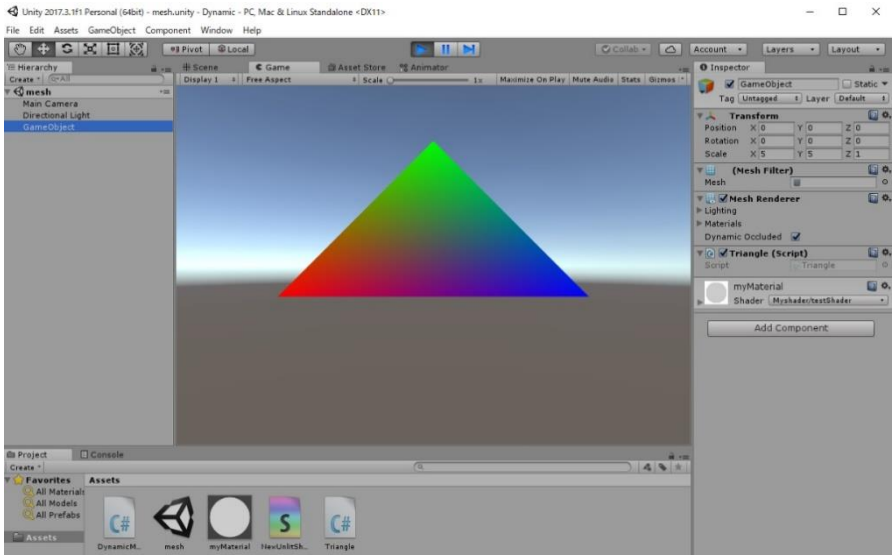


図 7.2.1-1　3 頂点に RGB が割り当てられた三角形の表示

可視化のマッパーでは動的なメッシュ生成と色の割り付けが必要となる。図 7.2.1-1 は三頂点に RGB の色を割り当てた三角形を表示した例である[7.2-1, 7.2-2, 7.2-3]。
以下に作成の手順を図 7.2.1-2 に示す。

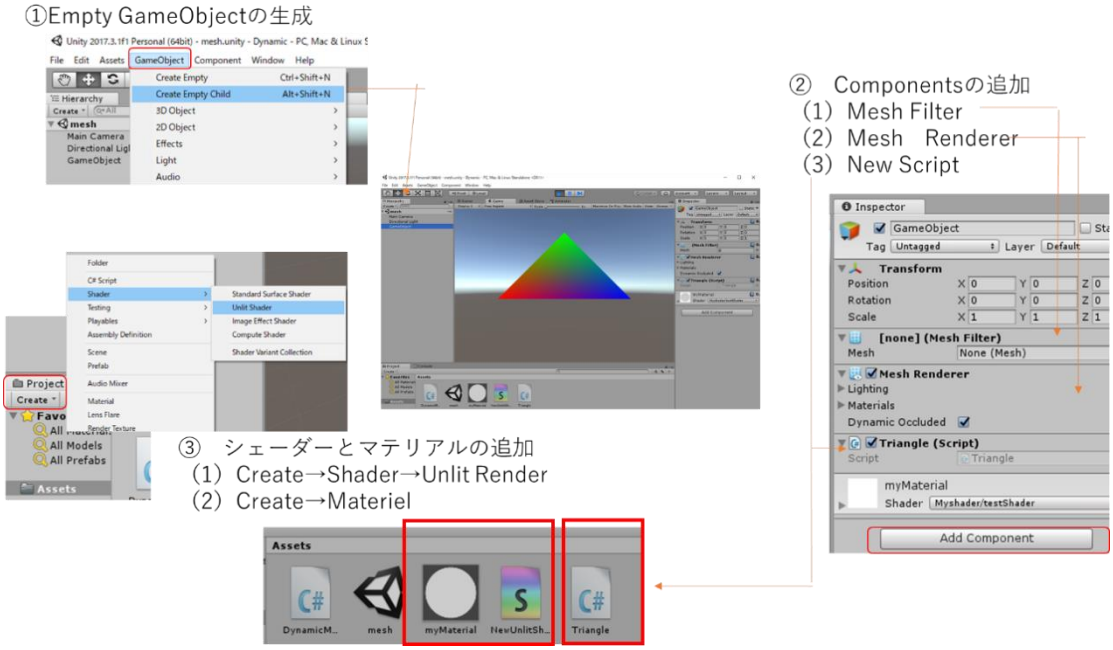


図 7.2.1-2　作成手順

①　空の GameObject の作成

Unity の上部のメニュー　GameObject→Create Empty を選択する。すると空の GameObject が Scene に追加される。(左側の Hierarchy に GameObject が追加され、右側の Inspector にも追加される。この時、Inspector には Object の名前と Transform のコンポーネントしか無い。)

②　Components の追加

Component は、Inspector に GameObject のメニューが表示されているとき、その一番下側にある“Add Component”メニューから追加する。(Unity の左側の Heirarchy で選択した Object のメニューが Inspector に表示される。)

(a) Mesh Filter の追加

(b) Mesh Renderer の追加

Add Component をクリックして表示されるポップアップメニューの上側にある Search 領域にカーソルを入れて文字を入力すると追加する component の候補がメニュー上にリストされる。図 7.2.1-3 は“mesh”と入力したところである。

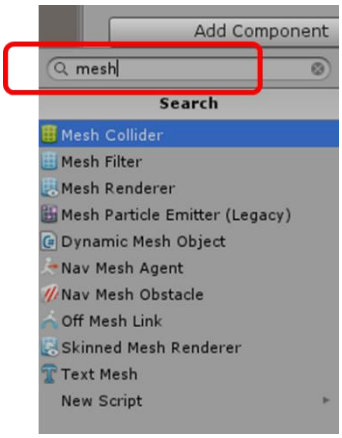


図 7.2.1-3　Component の追加

ここで“Mesh Filter”を選択して追加、さらに同じ操作で“Mesh Renderer”も追加する。追加された Component は図 7.2.1-2（右側）のように Inspector 上に表示される。この二つは、図 7.2.1-1 の三角形を表示するためには、単に追加するだけで良い。

(c) New Script

プログラム（Script）の追加は、Add Component で“New Script”を選択して追加する。NewScript を選択した後、Script 名を入力する。

ここでは“Triangle”と入力した（図 7.2.1-4）。これは Script の名前、つまり、関数名になる。ここでは、この Script に三角形の頂点座標の生成と、各頂点への色の割り付けを記述する。

そのプログラムの詳細は後述する。

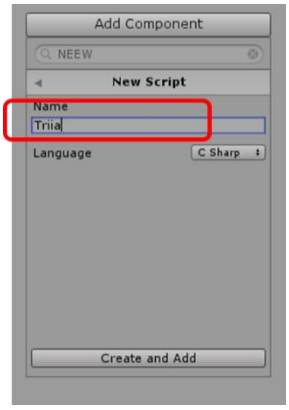


図 7.2.1-4　NewScript の追加

③　Shader と Material の追加

この二つは Unity の左下の project の中の Create メニューから追加する（図 7.2.1-2）。

(a) Unlit Render

これは Create→Shader の中から“Unlit Shader”を選択することで Asset（フォルダ）の中に NewUnlitShader が追加される。これは Script なので書き変えが必要である。その詳細は後述する。

(b) Material

Create→Material を選択すると Asset（フォルダ）の中に“New Material”が追加される。

名前は適当な名前にしておくと良い。図 7.2.1-1 では“MyMaterial”という名前にしている。この Material は①で生成した GameObject に追加する。この追加は“MyMaterial”のアイコンを Hierarchy の GameObject の上にドラッグする。追加が成功すると Inspector のメ

ニューに Material（ここでは“MyMaterial”）が追加される（図 7.2.1-5)。

これで準備は完了である。次に二つの Script について記述する。

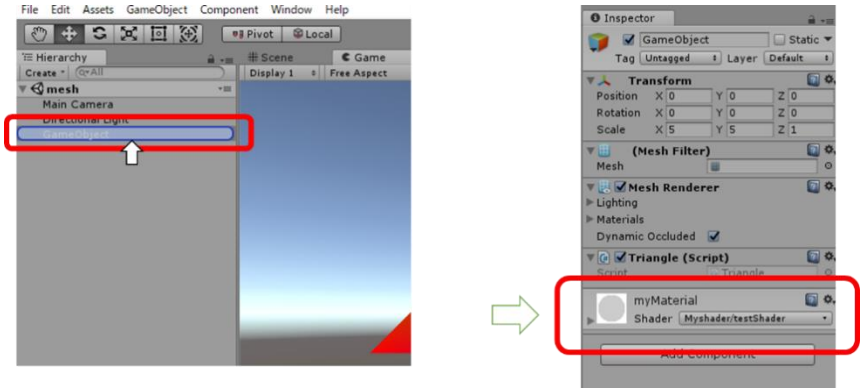


図 7.2.1-5　Meterial の GameObject への追加

7.2.2　Script の記述

一般的な Script の記述方法は Unity のマニュアル他を参照のこと。頂点生成と Color 割り付けプログラム（Script：Triangle）を付録 7.2-1 に添付する。プログラムは Start 関数に記述しているので、この記述は Unity の Play が押された直後に 1 回だけ実行される。3 頂点の XYZ 座標値、コネクティビティ（0, 1, 2）および頂点の色が設定されている。

(1) シェーダプログラム（Script）

プログラムを付録 7.2-2 に添付する。新しい Unlit Shader を作成すると付録 2 のような初期コードが生成される。それを書き換えることになる。

1 行目の Shader “Myshader/testShader” は、シェーダの名前で自由に名前を付けることができるが、これは Material で Shader を選択するときに使われる。

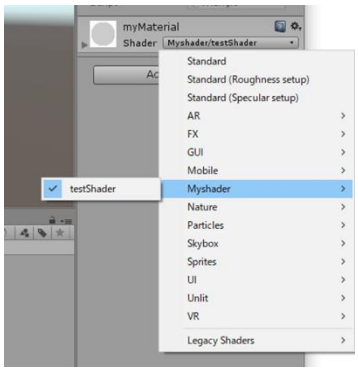


図 7.2.2-1　Shader の選択

Shader は Material で選択されることになっておりそのデフォルトは“Standard”である。

この Standard の Shader が頂点カラーを扱わないので新しい Shader を追加して Material で選択しなければならない。

Myshader/testShader の“/”はメニューの階層として扱われるようだ。Script の詳細は参考文献[7.2-

1]に記述されている。

7.2.3　コンター図の表示

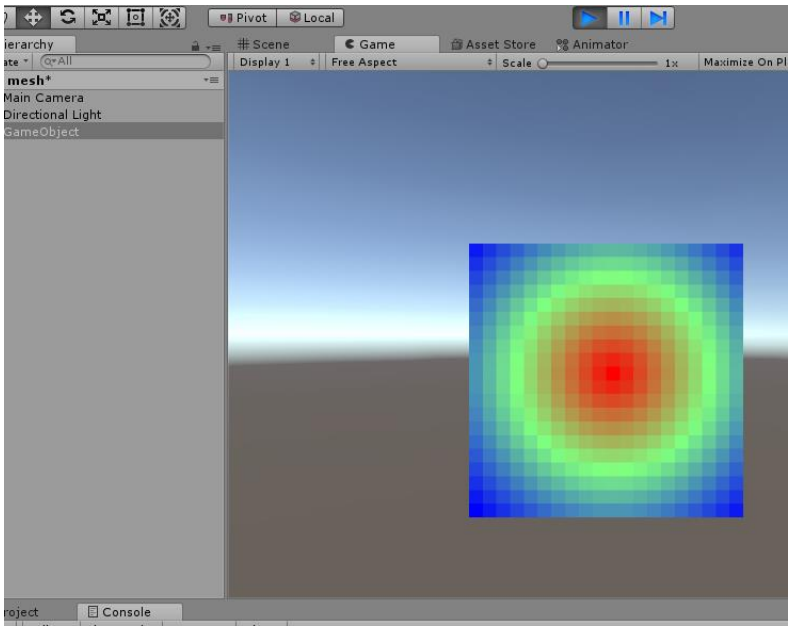


図 7.2.3-1　コンター表示プロトタイプ

頂点カラーの表示ができたので、あとは C#で可視化処理を追加していけば Unity 上に可視化 Asset を追加することができると思われる。図 7.2.3-1 は断面コンターの表示例である。

7.2.4 Unity 上での可視化 Prefab キットの設計と課題[7.2-4]

Unity は Asset をインスタンスするときに階層構造を取ることができる。

この仕組みを用いてデータフロー型の可視化システムのフレームワークを構築した。

図 7.2.4-1 に可視化 Prefab キットの例を示す。このうちプロトタイプとして、“Read File”、“Bounds”、 “Orthoslice”、“isosurface” の四つの Prefab を実装した。これらを用いた可視化例を図 7.2.4-2 に示す。

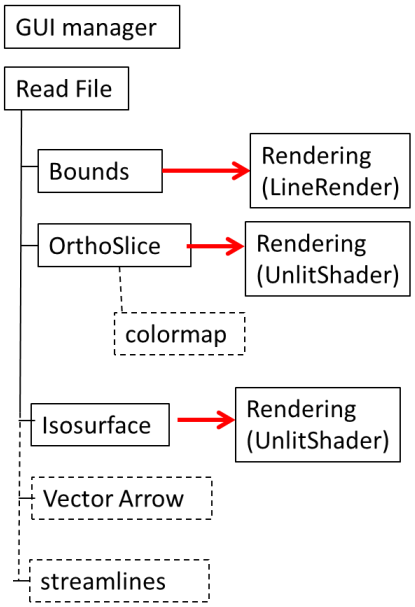


図 7.2.4-1　データフロー型可視化 Prefab キット設計例

これは AVS/Express のモジュール実装を念頭に置いて設計しているが、AVS/Express と異なるいくつかの事項がある。

(1) 親を二つ以上持つことができない

AVS のモジュール Streamline や isosurface では入力を二つ以上持つ場合があるが、階層構造では親を複数持つことができない。そこで Colormap のような付帯的なモジュールはマッパーモジュールの下側に配置するよう設計した。(図 7.2.4-1 では OrthoSlice の子として colormap が配置されている)

(2) Coroutine の実行

流体解析のパーティクルアニメーションのように自主的に動作するモジュールは Coroutine という仕組みを持っている。Unity では常に Prefab が実行常態にあるので、原則、全ての Prefab が常に実行状態にある。稼働を継続することはできるが、二つ以上の Prefab が動くとき、実行制御と同様に複雑さが生じる。

(3) Prefab の実行制御と実行順序

この階層上にデータフロー型の制御をマッピングした。親が更新したとき全ての子供に対する Active フラグを 1 にセットし、それを監視している子オブジェクトが実行を行い、Active フラグを 0 にリセットすることで実現した

(4) GUI の配置

GUI は Unity エディタ上ではキャンバスと呼ばれる 2 次元のパネルに配置し、そこにボタンやスライダーを配置できる。しかし、VR 環境で 2 次元 GUI は利用できない。そこで 3 次元のプレーンオブジェクトをカメラの直前に配置し、そこにテキストを表示し、C#のスクリプトでボタンの機能を実装した。VR 環境ではデモンストレーションが主たる目的と考え冗長な操作であってもゲームパッドのような周辺機器からボタンやスティックだけで操作できるようなシンプルな GUI を指向した。

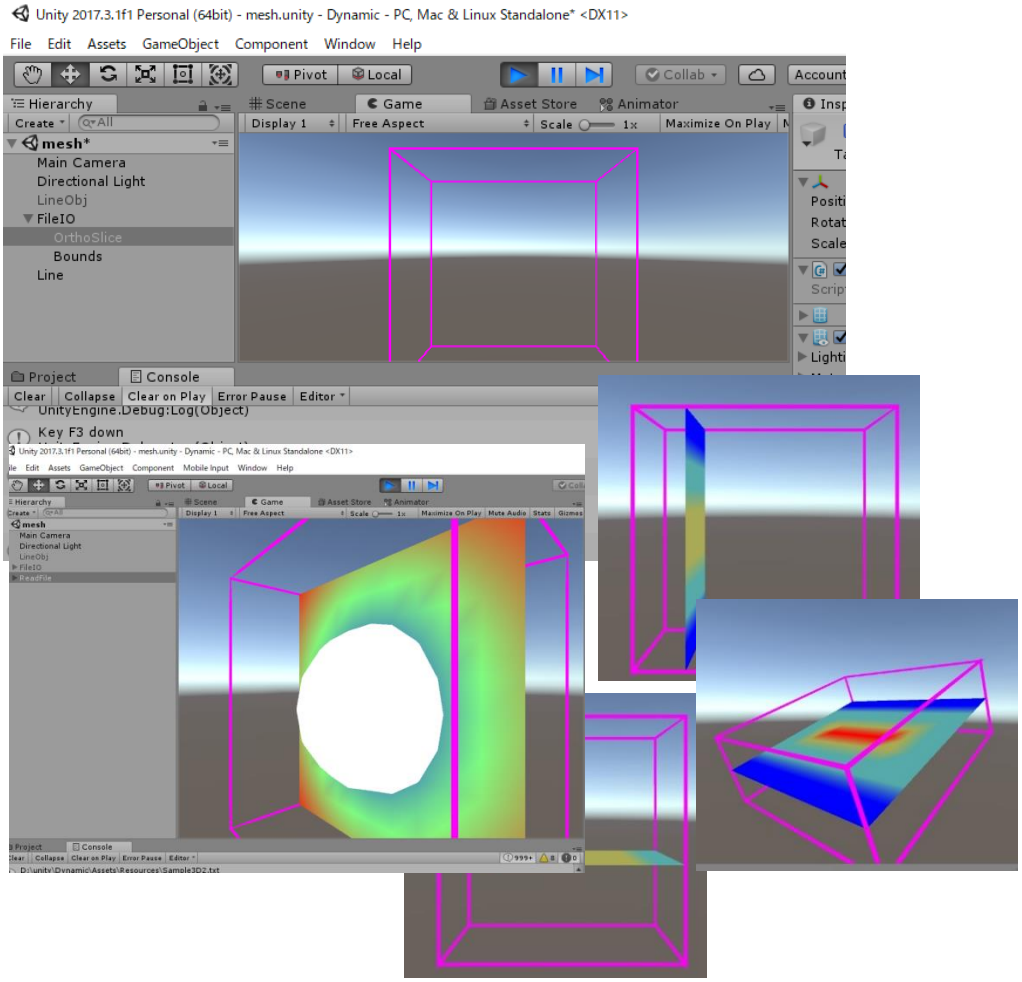


図 7.2.4-2　可視化 Prefab プロタイプによる可視化例

図 7.2.4-3 に GUI の実装例を示す。

①　可視化アセットの階層構造

Unity エディタ上では、アセットの階層構造は Hierarchy に可視化されている。VR 上での実行時には見えない。

② パラメータを操作する可視化アセットの選択

可視化アセットは図 7.2.4-1 の GUI Manager で管理されている。指定されたキー（例えば、左矢印キー、を押すことで対象となる可視化アセットのリストのカレントフラグが 1 インクリメントされる。リストの最後に到達すると一番前に戻る。一方向にすることで一つのキーで操作することができる。

③ 各可視化アセットのパラメータの選択

可視化アセットのパラメータの選択は、各アセットが管理する。2 種類以上のパラメータがある場合、プロトタイプでは②と同様の方式で操作対象となるパラメータを選択するように実装している。パラメータの操作は、値を上下する場合は別の二つのキーを割り当てている（ここは一方向で操作するのは適当でない）。

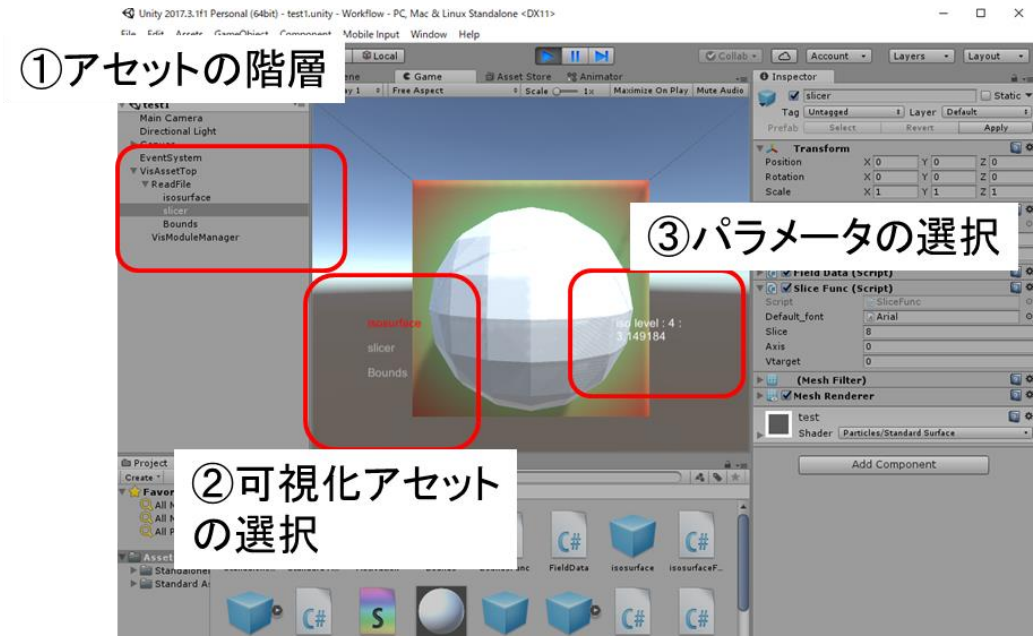


図 7.2.4-3 可視化 Prefab プロタイプの GUI 設計

参考文献

[7.2-1] はじめに測度ゼロの抹茶チョコ：“Unity で頂点色を表示するシェーダを作る”、2018 年 8 月 29 日、https://matcha-choco010.net/

[7.2-2] 渋谷ほととぎす通信：“Unity 動的にメッシュを生成するシリーズ”、2018 年 12 月 6 日、https://www.shibuya24.info/entry/dynamid_mesh

[7.2-3] 渋谷ほととぎす通信：“Unity 動的に生成したメッシュに対して頂点カラーを反映する”、2015 年 11 月 30 日、https://www.shibuya24.info/entry/2015/11/30/090000

[7.2-4] 宮地英生、川原慎太郎：“ゲームエンジン Unity 上での Visualization Assets の開発”、第 88 回 CAVE 研究会（東海大学高輪校舎）,2019.11.1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Triangle : MonoBehaviour {

    // Use this for initialization
    void Start () {
        var mesh = new Mesh ();

        var vertices = new List<Vector3> {
            new Vector3 (-1, 0, 0),
            new Vector3 (0, 1, 0),
            new Vector3 (1, 0, 0),
        };
        mesh.SetVertices (vertices);

        var colors = new List<Color> {
            new Color (1, 0, 0),
            new Color (0, 1, 0),
            new Color (0, 0, 1),
        };

        var triangles = new List<int> { 0, 1, 2 };
        mesh.SetTriangles (triangles, 0);
        mesh.SetColors (colors);

        var meshFilter = GetComponent<MeshFilter> ();
        meshFilter.mesh = mesh;
    }

    // Update is called once per frame
    void Update () {

    }
}
```

付録 7.2.2-1. 動的に頂点を生成して色を割り付ける Script 例

```
Shader "Myshader/testShader"
{
    SubShader
    {

        Pass
        {
            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            #include "UnityCG.cginc"

            struct appdata
            {
                float4 vertex : POSITION;
                float4 color : COLOR;
            };

            struct v2f
            {
                float4 vertex : SV_POSITION;
                float4 color : COLOR;
            };

            void vert (in appdata v, out v2f o)
            {
                o.vertex = UnityObjectToClipPos(v.vertex);
                o.color = v.color;
            }

            void frag (in v2f i, out float4 col : SV_Target)
            {
                col = i.color;
            }
            ENDCG
        }
    }
}
```

付録 7.2.2-2. 頂点 Color を有効にするシェーダ例

7.3 HMD 上での独自レンダラーの開発

日本原子力研究開発機構　河村拓馬

並列分散環境で実行される大規模シミュレーションの可視化では、様々なボトルネックが原因で従来的な可視化手法の適用が困難であり（1.3 節）、大規模向けの並列可視化手法が開発されている。並列可視化手法ではレンダラーも従来とは異なるものが採用されている。統合開発環境である Unity や UE4 だけでなく、既存の可視化ソフトである ParaView や Visit も主要な HMD への画像出力をサポートしているが、これらのアプリケーションには従来的な可視化手法のレンダラーが実装されている。HMD で大規模シミュレーション向け可視化をするには、並列可視化手法に合わせたレンダラーを実装する必要がある。この節では、大規模可視化向けに開発された粒子ベースのボリュームレンダラーを商用 HMD である Oculus Rift に移植する際に立ちはだかる実装上の課題とその解決策について記述する。

粒子ベースボリュームレンダリング（Particle-Based Volume Rendering, PBVR）は計算結果のボリュームデータを可視化用の十分サイズの小さい粒子データに変換し、それを画像面に投影してボリュームレンダリング画像を生成する可視化手法である[7.3-1]。PBVR は可視化ライブラリ KVS （https://github.com/naohisas/KVS）を用いて開発されている。Oculus Rift への移植に関して、フレームバッファ上でのボリュームレンダリング画像の生成までは KVS の機能を利用できるが、両眼用の画像の生成とその取り込みは Oculus SDK を利用して開発する必要がある。

KVS はシミュレーション結果データや医療用データ等の 3 次元データ向け可視化アプリケーションを簡単に開発するための C++クラスライブラリである。Windows、Linux、MacOS 上で動作し、ウィンドウの生成やデバイスへの入出力に GLUT を利用している。KVS では PBVR の他に、ポリゴンによる等値面、ラインによる流線、グリフによるベクトル場の矢印表示等、様々な可視化要素を利用した可視化に対応しており、KVS を用いてプログラミングをすることで統合的な可視化が可能になる。

(1) VR 対応機能開発の要件

KVS に実装されたクラスは、マウス・キーボードによるユーザーからの入力と、通常の 2D ディスプレイにおけるウィンドウシステムを使った出力を前提としている。一方、Oculus Rift は没入型 VR 装置であるため、マウスやキーボードといった通常の入力装置の使用は困難であり、Oculus Touch と Oculus Rift 本体のセンサによる入力のみを前提とする必要がある。また、出力に関しても 1 回の描画フレーム中に左右両眼に 1 回ずつ、計 2 回のレンダリングを行う必要がある。これらの制約から、以下のような問題が発生した。

① 視点移動に関する操作は、Oculus Rift 本体の内蔵センサによる位置情報に同期させる必要があるが、KVS にはその機能はない。

② 拡大縮小などの描画オブジェクトに対する操作を、Oculus Touch のスティックやボタンで行う機能が KVS にはない。

③ KVS が提供する対話操作のための GUI への操作機能は、マウスカーソルやキーボードフォーカスを前提としたものであり、Oculus Touch による操作が KVS では実装されていない。

④ 左右両眼へのレンダリングを 1 回の描画フレームで実行することを前提としたレンダリング管理機能は KVS では実装されていない。KVS が提供するレンダラーは通常の 2D ウィンドウへの描画を前提としているため、1 回の描画処理で左右両眼に 1 回ずつ（計 2 回）のレンダリングを行う場合に不具合が発生する可能性がある。特に確率的レンダラーでは内部で独自のフレームバッファを使用しているが、Oculus Rift のような 1 回の描画フレームで、カメラ位置を変えて複数回レンダリングを行うような実装では、描画フレーム内で複数のフレームバッファを使用する必要があるため、レンダラーにイブのフレームバッファ切り替え機構と衝突して、問題が生じた。

これらの問題を解消するため、以下の機能を開発した。

① Oculus Rift 本体の内蔵センサによる位置情報をリアルタイムに取得し、視点移動を行う機能。

② Oculus Touch のスティックやボタンの状態をリアルタイムに取得し、拡大縮小などの描画オブジェクトに対する操作を行う機能。

③ Oculus Touch のスティックやボタンの状態をリアルタイムに取得し、対話操作のための GUI に対する操作を行う機能。

④ 左右両眼へのレンダリングを 1 回の描画フレームで実行することを前提としたレンダリング管理機能。そして、1 回の描画処理で左右両眼に 1 回ずつ（計 2 回）のレンダリングを行うことを考慮したレンダリング機能。

(2) 開発時に発生した問題と解決策

実装時に発生した問題と解決法について 1 例を記述しておく。ボリュームレンダリング機能の実装時にフレームバッファに関する問題が発生した。図 7.3-1 に示す失敗例では両眼のフレームバッファによる表示領域をまたいでいる、レンダリングされたフレームバッファの右端にかかっていると思われるクリッピングが発生している、激しいちらつきがある、Oculus HMD 側には全く表示されない、といった状況が確認された。

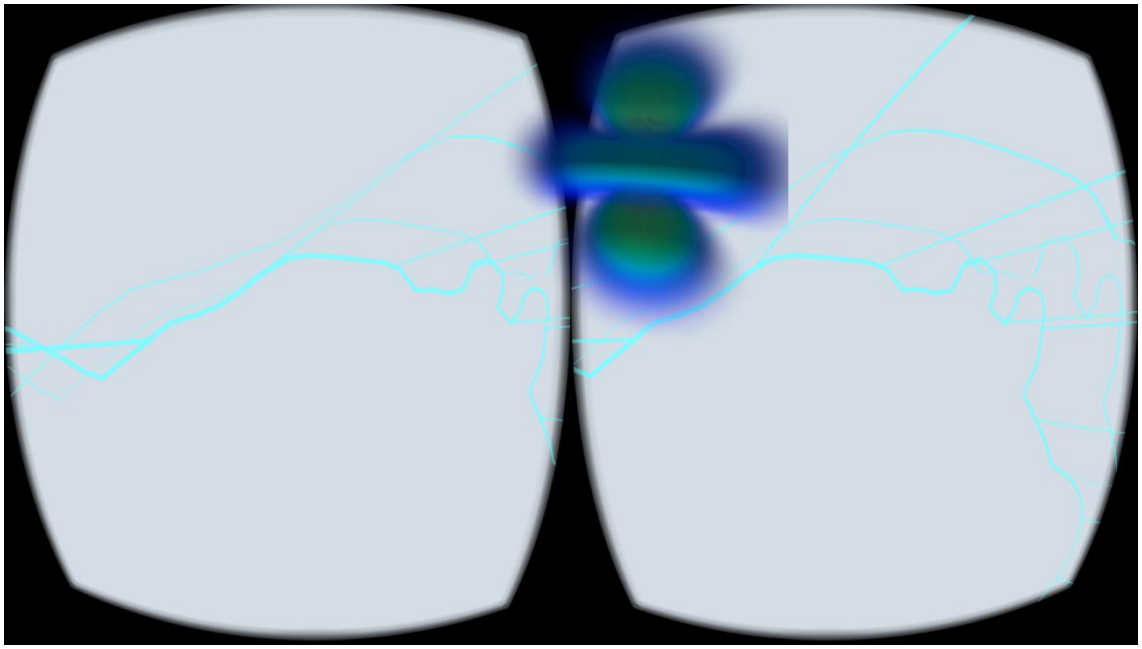


図 7.3-1　Oculus 上のボリュームレンダリング（失敗）

この問題を解消するため、RayCastingRenderer クラス内でフレームバッファを切り替える際に GuardedBinder を使用するように修正した。GuardedBinder とは、フレームバッファをバインドする際に、直前にバインドされていたフレームバッファを記憶しておき、バインドを解除する際に記憶していたフレームバッファへのパイントを回復する機能を持つクラスである。GuardedBinder を使用することで、表 7.3-1 のように挙動が変化した。

表 7.3-1　GuardedBinder 使用による挙動の変化

	適用前の挙動	適用後の挙動
Oculus HMD への表示	［×］何も表示されない	［△］左目だけ表示される
GLUT ウィンドウの表示	［△］ 左目だけ表示される	［△］変化なし （左目だけ表示される）
GLUT ウィンドウ表示のちらつき	［×］ちらつきあり	［〇］ちらつきなし
フレームバッファのサイズ	［×］おかしい	［〇］正しくなった

左目画像しか表示されない問題については、左目・右目それぞれに個別のフレームバッファを作成する手法への修正を行った。フレームバッファ及びそれに付随するカラーテクスチャとデプステクスチャをまとめて管理するため、HUD 実装の際に使用した。OculusTextureBuffer クラスを使用した。なお、OculusTextureBuffer クラスは Oculus SDK のサンプル（OculusRoomTiny（GL））で 3D 画像を描画するために使われていた実装を一部修

正したものである。また、従来の実装では、1 枚のフレームバッファの左半分と右半分がそれぞれ左目用・右目用の描画領域となるように、ビューポートの原点とサイズを設定していたが、フレームバッファを 2 枚にしたため、原点とサイズの設定を変更した。ここまでの変更を行うことで、図 7.3-2 に示すとおり、RayCastingRenderer が正常に描画されるようになった。

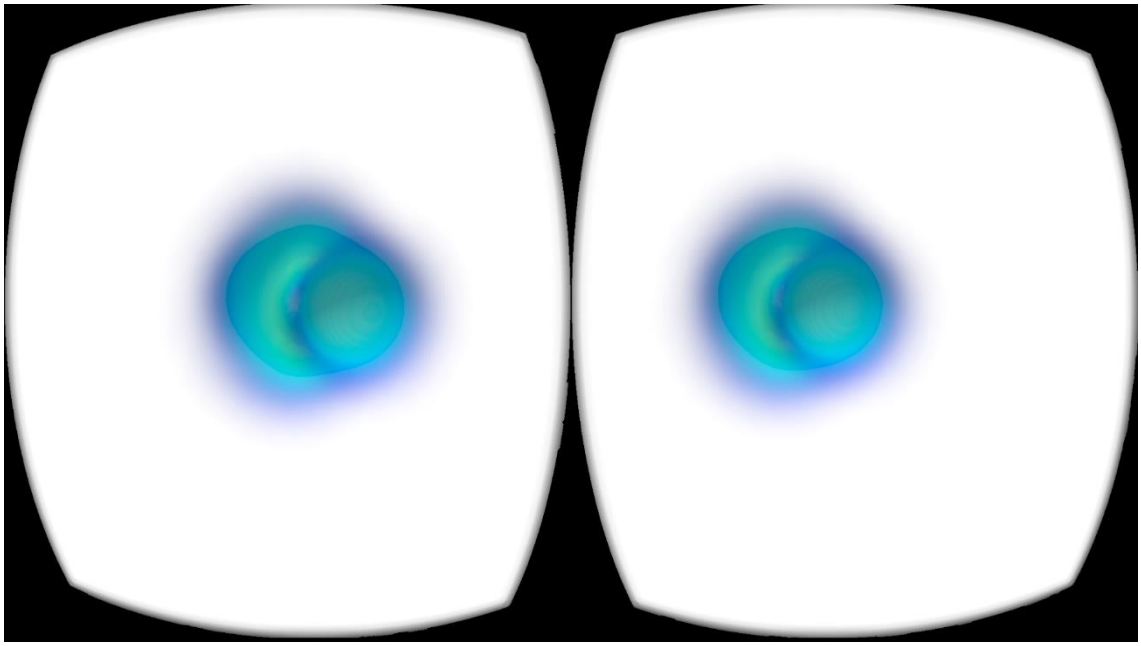


図 7.3-2　Oculus 上のボリュームレンダリング（成功）

参考文献

[7.3-1] Takuma Kawamura, Naohisa Sakamoto, Koji Koyamada: “A High Quality Sampling Technique for Particle-based Volume Rendering”, IEEE Visualization(Poster) (2009).

8.　可視化の事例集

8.1 計測点群データの可視化

立命館大学　長谷川恭子

近年、レーザ計測装置を用いることや、写真ことで容易に点群データが取得できるようになっており、VR 空間として、現在の風景を点群として取得して描画する事例も増えている。Unity では、点群データに対応したアセットが多くある。例えば、Kinect で取得したデプスマップを取得するプラグイン（https://www.microsoft.com/en-us/download/details.aspx?id=44561）では、ポリゴン化してリアルタイム表示するものである。また、点群をポリゴン化せずに描画できるアセットとしては、“Point Cloud Viewer & Tools for Unity”や“Pcx - Point Cloud Importer/Renderer for Unity”等がある。これらのアセットは点群データのファイルフォーマットとして、Stanford Triangle Format として知られている PLY 形式が指定されている。 また、Unreal Engine でも点群の読み込みおよび描画が可能なプラグインとして“Point Cloud Plugin”（https://pointcloudplugin.com/ ）が公開されている。以上のように、ゲームエンジンでは従来不得手であった、点群データの取り組みが行われはじめ、このようなアセットやプラグインを用いることで、図 8.1-1 のような 1 億点以上の計測データの扱いが容易になってきている。

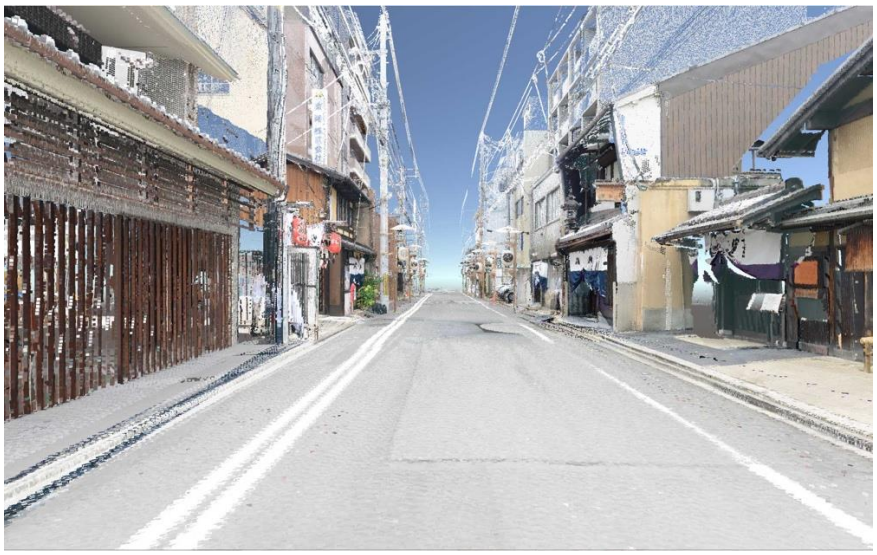


図 8.1-1　計測点群の可視化

8.2 数学と可視化

広島大学大学院教育学研究科　北葔如法

筆者が大学の工学部、教育学部、大学院教育学研究科の数学の授業や教材開発で実践してきた事例を紹介する。 主に数学教育を改善したいという動機の実践である。

8.2.1　3DCG アニメーション

初期の試みはベクトル解析という内容の授業の空間曲線の速度ベクトルや加速度ベクトル、捩率などや、線積分の考え方を可視化して解説する 3DCG アニメーションを作成したというものである［8.2-1］(図 8.2.1-1)。

ベクトル解析は 3 次元空間の数学であり、特に何をやっているかわからなくなる学生が多くなりがちな内容である。

フリーの 3DCG 作成ソフトウェア POV-Ray で制作した。 当時としては、これで 3 次元の表示ができるし動きもあるので、ある程度わかりやすくなり、学生アンケートの結果も良好であった。 しかしインタラクティブではないアニメーション動画を作っただけで、ユーザー側で試行錯誤できるものではなかった。

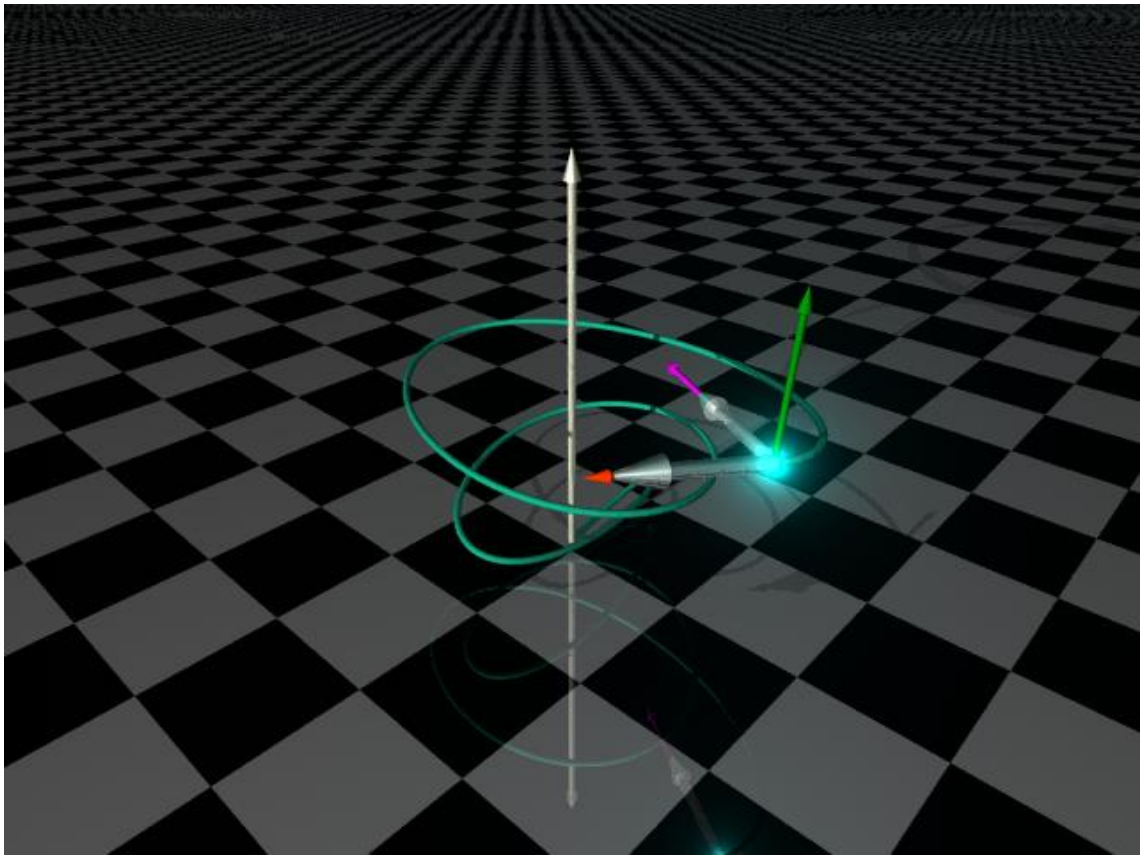


図 8.2.1-1　ベクトル解析の 3DCG の動画の一コマ
曲線の接線ベクトル、主・従法線ベクトルを表示したアニメーション

8.2.2　iPad と Mac を通信するインタラクティブな空間曲面教材

同じくベクトル解析において、二つのパラメータ（例えば (u,v)）で（つまりベクトル値2変数関数で）空間内の曲面を表現するということを解説するインタラクティブなソフトウェアを作成した。 uv 平面は平らな平面であり、これを当時発売されたばかりの iPad 上に表示し、タッチした点（u,v）に対応する空間の曲面上の点が離れたパソコンで表示されるというものである。 iPad と Mac は無線 LAN で通信し、マルチタッチの全ての座標を Mac に送信して表示した。 これにより、指を iPad 上で u 方向、v 方向に動かしたり、丸を描いてみたりすると、曲面上のどの点に対応しているのか理解しやすいインタラクティブな可視化ができた（図 8.2.2-1)。 Mac を教室のプロジェクタにつなぎ、曲面を教室の前面スクリーンに大きく表示しておき、iPad を教室中持ち歩きながら次々に学生に操作させると、対応する点をスクリーン上において全員で確認しながら体験することができ、教室中に一体感ができてよい。

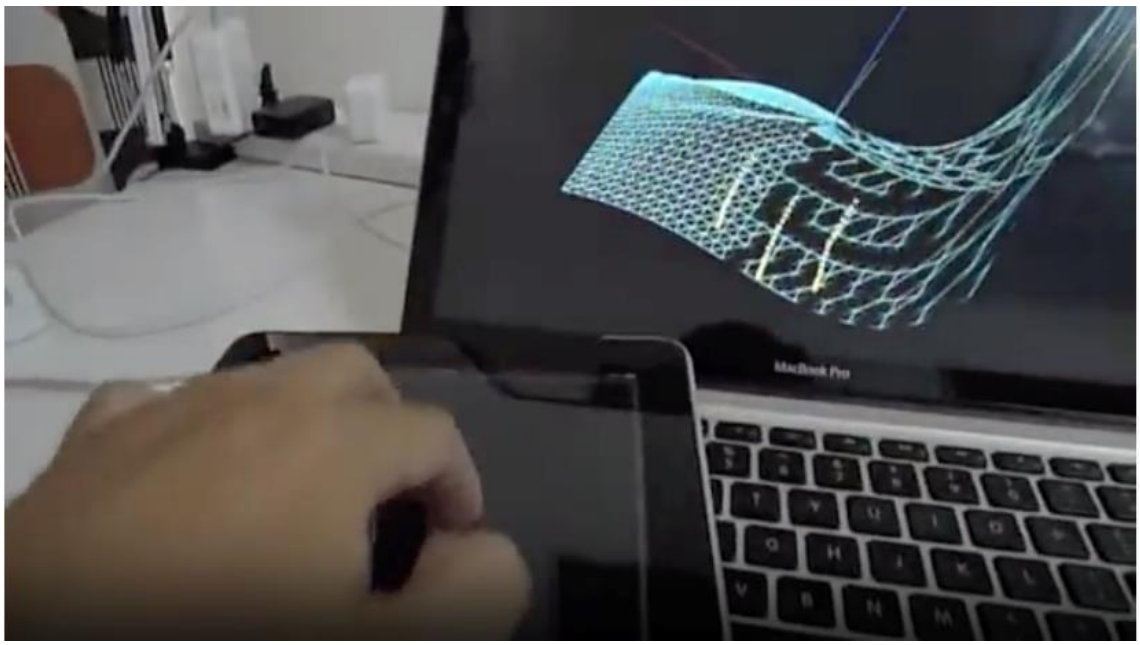


図 8.2.2-1　iPad 上を 3 本指で触りながら Mac の画面の対応する曲面上の点を示しているところ。指の動きに合わせて曲面上に黄色の 3 本の曲線が現れる。

8.2.3　数学と AR with Unity + Vuforia

次に AR 教材を Unity で作成した事例を紹介しよう。

まず微分積分学における多変数関数の微分法についての AR 教材を作成した。 2 変数関数のグラフは概ね 3 次元空間内の曲面をなす。 やはり 3 次元のオブジェクトであるため、紙の教科書ではよくわかる学生にしかわからないという状況が起きがちである。

あらかじめいくつか 2 変数関数を用意しておき、一つを選び、カメラで写した空間にその関数のグラフである曲面を AR 技術で表示し、 グラフにタッチするとその点での二つの偏微分を線分の傾きで表示する。 また、いくつかの条件付き極値問題の可視化を同時表示するようにした。 これで、偏微分、接平面、極値、条件付き極値といった、2 変数関数の微分法で通常学習する概念の多くについて、手を動かして把握することができる。

また、同じく AR 技術を用いて、射影幾何の教材として「3 次元空間内にある無限に長い平行な 2 直線を見る」という AR 教材も作成した（図 8.2.3-1)。3 次元空間内にある無限に長い平行な 2 直線が、遠くでどう見えるかを体験することは、射影幾何のアイディアの重要な手がかりとなる。

図 8.2.3-1　3 次元空間内にある無限に長い平行な 2 直線を見ているところ

これらの作成には Unity に Vuforia という AR フレームワークを組み合わせて実装した。Vuforia は 2012 年ごろに筆者が知った AR フレームワークで、当時は Qualcomm が開発していた。現在は PTC に委譲されているようだ (2015)。 当初から Unity のフレームワークが存在し、Unity に組み込むことは用意であった。さらに Unity 2017.2 で Unity 本体に統合され、導入がより簡単になった。

Unity に Vuforia を使用方法を述べておく。

(1) Vuforia にユーザー登録をする。
(2) License Manager https://developer.vuforia.com/vui/develop/licenses でライセンスキーを取得しておく。
(3) Unity で Vuforia を入れておく。
(4) Create → Vuforia → AR Camera
(5) Open Vuforia Configuraion でライセンスキーを入れる。
(6) Create → Vuforia → Image
(7) Image Target Behaviour の中の Data Set を TargetName に設定。その下の Image Target プロパティも変わる。
(8) ImageTarget の子として他のオブジェクトを入れる。
(9) Directional light は ImageTarget の外にあればよい。

最新版では不要だが Unity 2017.2 以前で使用するには、以下のとおりにすればよい。

(1) Vuforia ユーザー登録する。(Vuforia のサイトで独自マーカも作れる)

(2) https://developer.vuforia.com/downloads/sdk から SDK ダウンロードする 。(.unitypackage がある)
(3) Unity に import する。
(4) まず Main Camera を削除する。
(5) import した中にある ARCamera を配置する。
(6) サイトの License Manager でライセンスキーを出し、ARCamera の App Lisence Key のところに入力する。(2016 年ごろから)
(7) Load Data Set TargetName と Active にチェックを入れる。
(8) (同じプレハブから) ImageTarget を配置。Image Target Behavior の中の Data Set を TargetName に設定。その下の Image Target プロパティも変わる。
(9) ImageTarget の子として他のオブジェクトを入れる。
(10)Directional light は ImageTarget の外にあればよい。

2 変数関数のグラフとなる曲面として自由な曲面を Unity で作り出す方法を述べる。Mesh オブジェクトに、頂点のデータと三角形のデータ (頂点のインデックスで指定する) と法線と UV のデータ (.vertices, .triangles, .normals, .uv) を計算して与えて、MeshFilter オブジェクト の .sharedMesh プロパティにその Mesh オブジェクトを設定すればよい。

注意として、以前筆者がはまってしまったポイントを述べる。

AR に限らないことではあるが、Camera は near に注意する。Unity の画面でオブジェクトを置いたときには見えていても、実機でテストするとカメラの位置が異なり、near でカットされてしまってオブジェクトが見えていないことがある。インポートや設定のミスかもしれないと気付かず永遠に調べてしまったことがあった。

また、2 変数関数のグラフを表示するアプリは UnityScript という JavaScript ベースの言語で書いていたため、現在の最新版の Unity ではサポートされなくなっており、動かなくなってしまった。 継続的に使用可能にしたり、改良をしていくには、作りっぱなしにせず Unity やその周辺のソフトウェアの互換性を定期的にチェックしておく必要があると教訓を得た。UnityScript サポートが終了する前に移行期間に変換ツールで C# に変換することもできたようだったが (未確認) 時既に遅しであった。株式会社ユニティーの安原裕二氏へ質問しても現在は変換する方法はおそらくないとのことであった。現在仕方なく C# で少しずつ書き直している途中である。

AR アプリを作成する他のやり方を付記しておく。 iOS では Apple 公式の ARKit というフレームワークが用意されている。 かなり強力にハードウェアと連携しており、カメラの映像に吸い付くように滑らかに表示される。 また、マーカレスといって、あらかじめ覚えさせた画像が必要ない形式でも AR コンテンツが作成でき、 水平面、垂直面 (垂直面は iOS 12 から) を自動で認識してくれる。

Android は Google が ARCore というフレームワークを用意している。

ちなみに、グラフをARで日常風景に重ねて表示するだけであれば、GeoGebra ARというアプリが使える。元のGeoGebraは平面図形や空間図形の幾何学の作図や関数のグラフ、アニメーション、自動定理証明などができるオープンソースの数学ソフトウェアである。

8.2.4 数学とVR with Unity or WebVR

数学へのVRの試みを紹介する。

まず、比較的安価なVRデバイスVIVE FocusをUnityで使う方法を述べる。HTCがスタンドアロンのHMDであるVIVE Focusを日本で昨年秋に法人向けに販売開始した。スタンドアロンのHMDはOculus Goが発売されていたが、VIVE Focusは6 degrees of freedom（6DoF）となっており、頭の回転（3次元）だけでなく頭の移動（3次元）が取れるため、より数学では3次元の図形を観察するのに適していていると言える。

VIVE Focusの中身はAndroid端末であり、次のように通常 Unity で Android アプリを開発する手順で概ね開発ができる。VR対応するための WaveSDK というフレームを導入することだけが特別である。

(1) Android Studio を入手する。
(2) ViveFocusの developer → “Become a Developer” → “Material Download” → WaveSDK
(3) Unityの Build setting で 、Android に “Switch Platform”。
(4) JDKはJava SE 8を入れた。
(5) wavevr.unitypackage の WaveVR というオブジェクトが Aseets/WaveVR/Prefabs/ に入っているので、これをシーンにドラッグアンドドロップする。
(6) コントローラは同じフォルダの ControllerLoader。これも放り込む。
(7) 注意点は以下のとおり。

① 取得可能なイベントのサンプルは、sample の Hello VR の GoEvent.cs というスクリプトなどに入っている。
② VIVE Focus は Android 端末なので、Setting の Developer のところで、Developer、USB Debugging を on にする。

しかしここで 数学の教員を目指す人・数学の教員はほとんどプログラミングが素人であるという問題を考えよう。実際に上記の VIVE Focus をもっと数学教育分野で活用したいと考えるとき、教育現場での開発にAndroidアプリを作成しなければならないというのはなかなか実行可能性に乏しい。そこで、プログラミング初心者にもVR開発を、ということで、よりプログラミングの知識を少なく導入できる WebVR という技術に注目している。

WebVR はいわゆるウェブブラウザでVR表現をするAPIの規格集であり、現在は VR だけでなく AR, MR を統合する XR を扱うという拡大された WebXR という規格が 2019年11月20日現在、W3C Editor’s Draft, 19 November 2019 まで出ている。 https://immersive-web.github.io/webxr/

WebVR, WebXR ではウェブブラウザでページを見ることで VR 体験をするものであるから、基本的には HTML と JavaScript を書けばよい。 つまり、簡単な VR 教材ならば、テキストファイルを少し書くだけで作成可能である。

加えて、表示の仕組み自体は OpenGL をウェブブラウザで表現する WebGL の上に構築されており、既に存在する WebGL を扱うグラフィックライブラリを利用することでさらに簡単に作成できる。 three.js https://threejs.org/ や A-Frame https://bulan.co/swings/webvr-a-frame/ が挙げられる。

ここからは、学生に three.js を用いて数学の教材を作る技術も入門させながら、数学教育の教材作成を行うという大学院 1 年生向けの授業の実践を紹介する。

three.js の基本的な使い方は次の通りである。

(1) html で three.js 本体の JavaScript を読み込む。
(2) (以下 JavaScript で) シーンを作る。
(3) カメラを置く。
(4) 光源を置く。
(5) オブジェクトを置く。
(6) レンダリング する。(アニメーションループで)

あとは WebVR 対応にするには、冒頭で three.js をロードした直後に three.js のソースツリーの中にある vr/WebVR.js もロードすれば良い。

授業内で学生から出た問題の一例を示す。

まず問題は「太さの等しい中身のつまった二つの円柱が直角に交わっている。その共通部分の体積を求めよ」というものである。 大学入試でもよく出題される定積分の問題である。(日本女子大、大阪大、東工大、ほか類題多数)

体積を求める前に、この形が想像できるだろうか。

注意点は、円柱の共通部分の形が具体的に想像できなくても計算はできることである。形が想像できなくて特に悪いわけではない。 むしろ絵がなくても計算ができることも数学の良さである。

しかし、これを想像できても損はなく、むしろこのような空間図形の苦手意識を、VR の助けを用いることで軽減したり、むしろ楽しいものだと思えるように変えることができればとてもよいと考える。

作例として、ヒントの図形 (図 8.2.4-1) を出すということを考えた。 ヒントの図形は、単純に二つの円柱を交わらせただけのものである。それがその共通部分のヒントになっている。 学習者がこのヒントの図形を様々な角度から立体的に観察することを想定している。

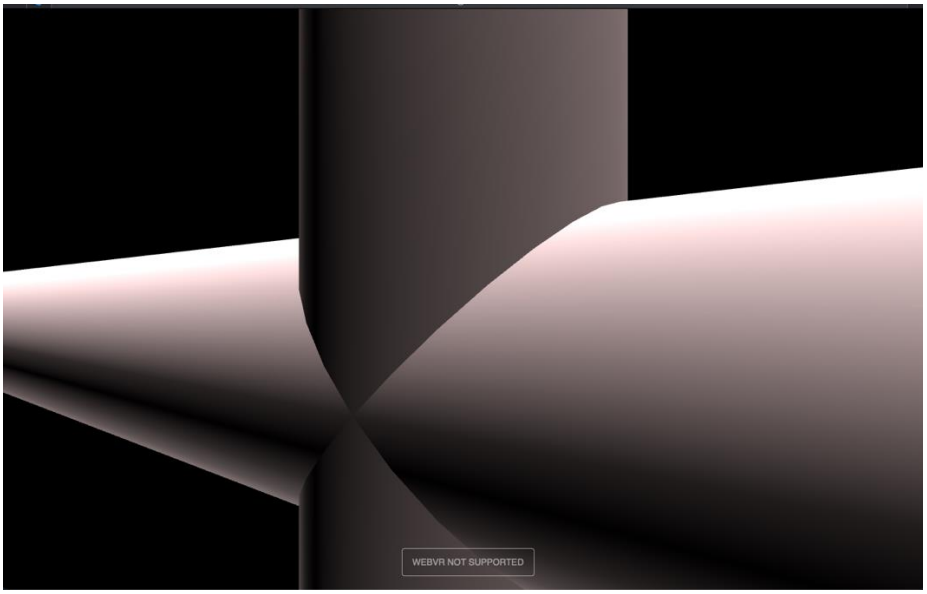


図 8.2.4-1　ヒントの図形

（VR 表示ではないブラウザの表示。実際にはこれを WebVR により HMD で立体視する。）

(1) これが想像の助けになるか。
(2) 助けにならなくても何か得るものがあるか
(3) 楽しさを感じるか
(4) 苦手意識が減るか

などといったことをこれから効果測定し、教育効果を測っていくことを考えている。 空間図形に苦手意識がある人は多く、他の数学が得意な人でも存在する。 苦手意識を克服するツール開発の可能性もあるのである。

8.2.5　数学・数学教育小まとめ

数学の研究における可視化・VR 活用はまだ始まったばかりであるが、世界中で注目されている。

数学教育に有効かどうかはまだまだ検証が必要であり、 効果測定の方法論が確立されていないことが課題である。

AR 教材は使う側は手軽であるしタブレットやスマートフォンで気軽に実施することができる。 VR 教材については安価なスタンドアロンの HMD が入手しやすくなってきたので、高スペックのマシンと HMD を用意しなくてもできるようになってきた。これにより教育現場でも従来よりも導入しやすくはなっている。 しかし依然として HMD を装着している人にしか対象物が見えないという欠点はある。 通常、授業なら数十人、セミナーのような少人数の学習環境でも通常数人はいる。 また、WebVR, WebXR といった技術を使うことで、Unity のような大がかりなものを使わない教材作成もこれからは並行して推進することも良いの

ではないかと考える。

参考文献

[8.2-1] 北臺如法，伊藤浩行：“工学系数学教育における動画教材の導入–教育 GP「工学教育を支える「数学力」養成プログラム」の取組として–”，平成 22 年度 工学・工業教育研究講演会 講演論文集 (2010)，pp.68-69.

8.3 CAVE 向け興味領域（ROI）抽出機能と詳細度（LOD）自動調節機能

兵庫県立大学　大野暢亮

大規模な 3 次元データを CAVE 装置で対話的に解析するために、CAVE 用汎用可視化ソフト VFIVE（神戸大学 陰山教授が開発）に興味領域（ROI）抽出機能と詳細度（LOD）自動調節機能を実装した[8.3-1,8.3-2]。この機能により、例えば、スカラーデータの可視化では、常にほぼ一定の大きさのデータを扱うだけで良くなり、可視化に要する時間（等値面生成等の CPU への負荷）と表示速度（ポリゴン表示等の GPU への負荷）をおおよそ一定に保つことができ、大規模データの VR 空間内での対話的な可視化が可能となった。機能の詳細は、参考文献[8.3-1],[8.3-2]に記されている。図 8.3-1～3 は、マントル対流シミュレーションのデータを本機能を利用して CAVE で可視化している様子である。

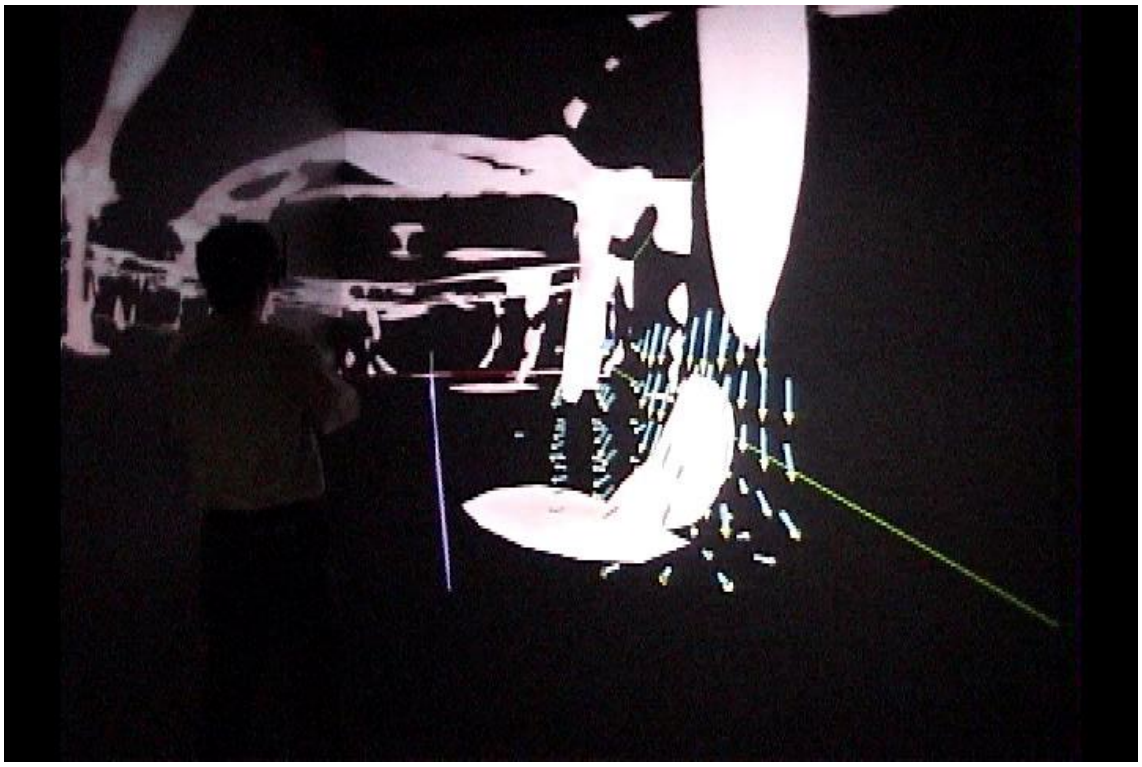


図 8.3-1　データを概観

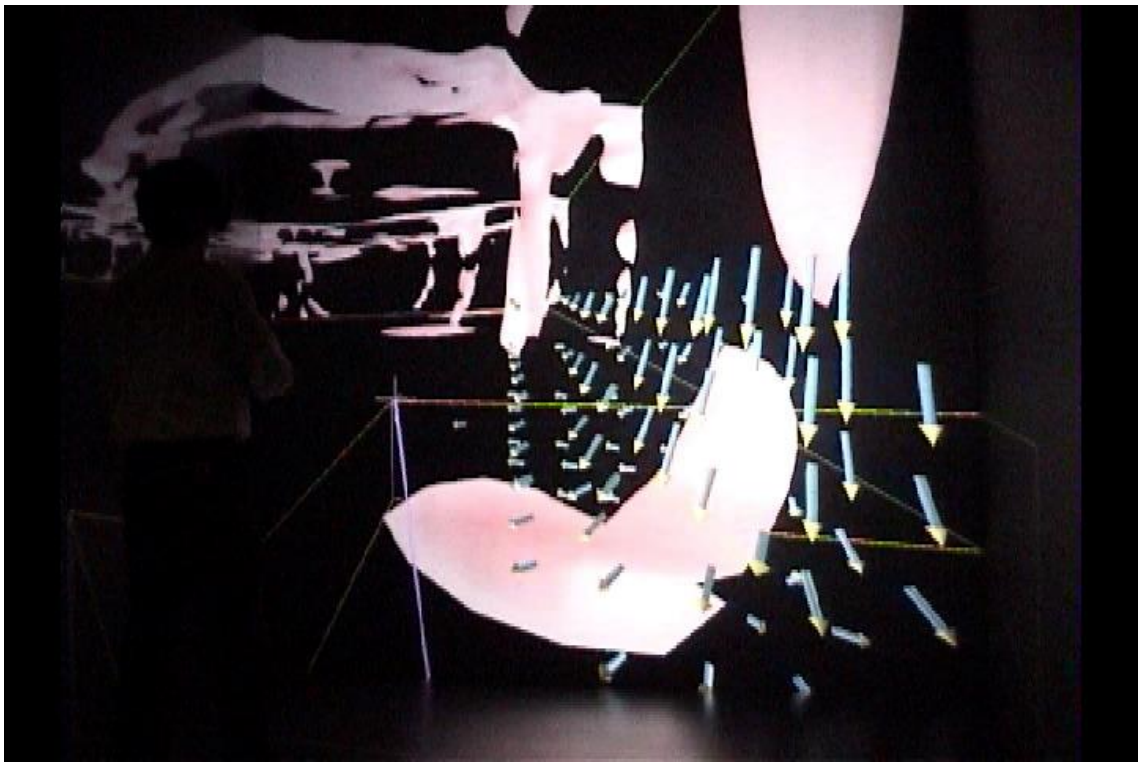


図 8.3-2　詳細に見たいところ（ROI）を選択

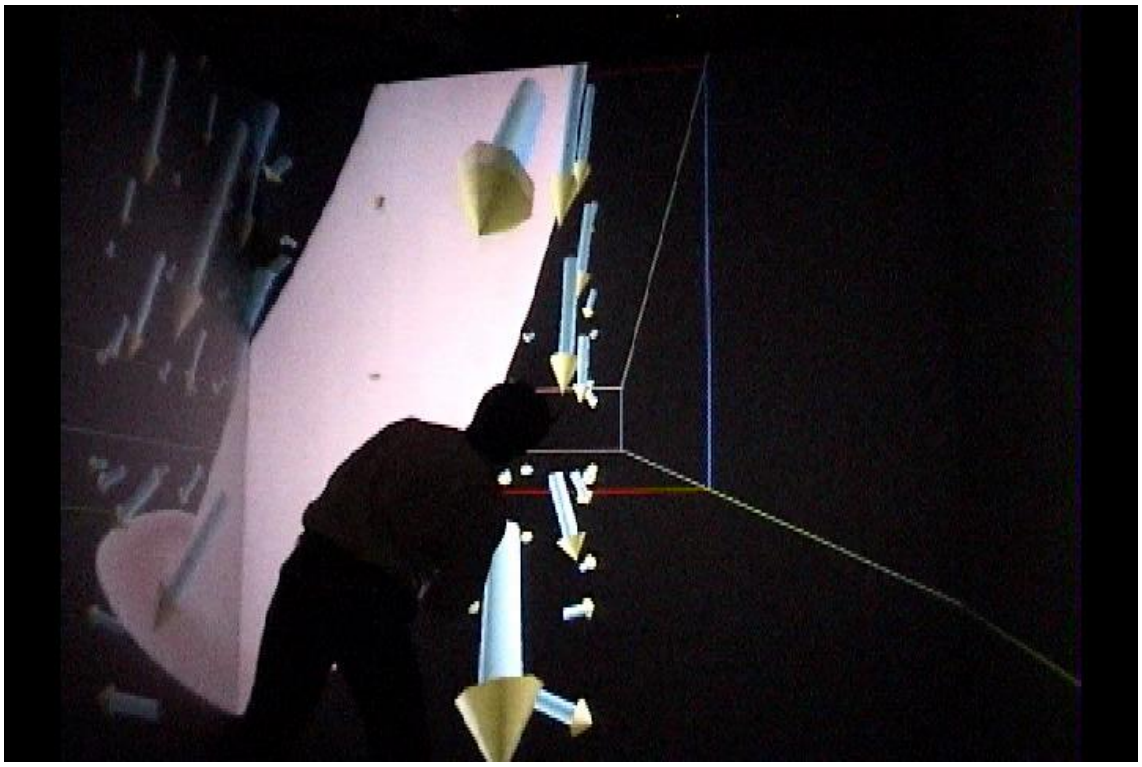


図 8.3-3　選択された ROI を詳細に観察

※データは、愛媛大学 理学部 亀山真典教授から提供されたものである。

参考文献

[8.3-1] N.Ohno, A .Kageyama: “Region-of-Interest Visualization by CAVE VR System with Automatic Control of Level-of-Detail”, Computer Physics Communications, Vol.181 (2010) pp.720-725.

[8.3-2] 大野暢亮: “大規模シミュレーション結果の CAVE 装置による対話的可視化 -興味領域の抽出と自動詳細度設定-”, 可視化情報学会誌, Vol.39, No.152, (2019) pp.3-7.

8.4 4D シアターと 4DVisualizer

理化学研究所　野田茂穂

4D シアターとはシミュレーションや実験で取得した 4 次元（空間＋時間）データを可視化する事を目的に整備したフラットウオール型の立体視表示システムである。1999 年に導入し、現在はプロジェククターのリプレースを重ね第 3 世代を運用している。幅 6m 高さ 2.5m の大型スクリーンを床面から隙間なく立ち上げることにより、自然な空間的奥行きを感じることができる立体視表示を実現している。（図 8.4-1）また、リアプロジェクションシステムを用いることでプレゼンターの影がスクリーンに映り込むことを回避している（図 8.4-2)。どの座席からでも表示が見えるよう、階段式座席（40 席）を用意している。

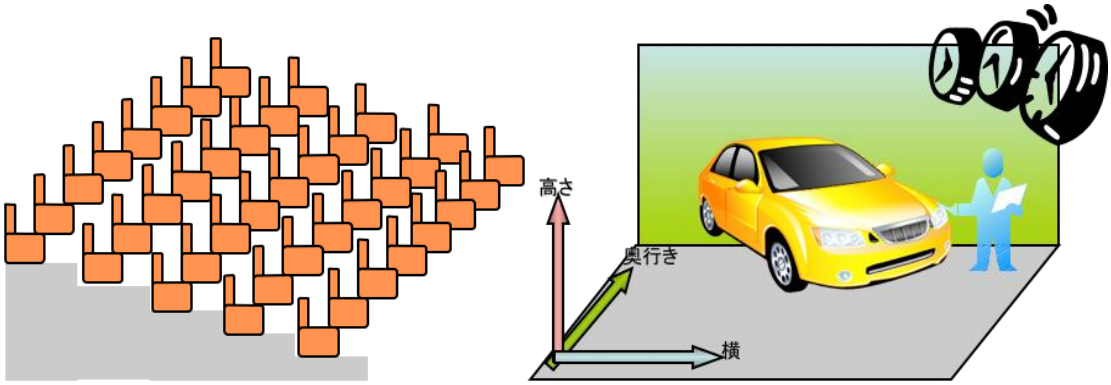


図 8.4-1　4D シアター概略図

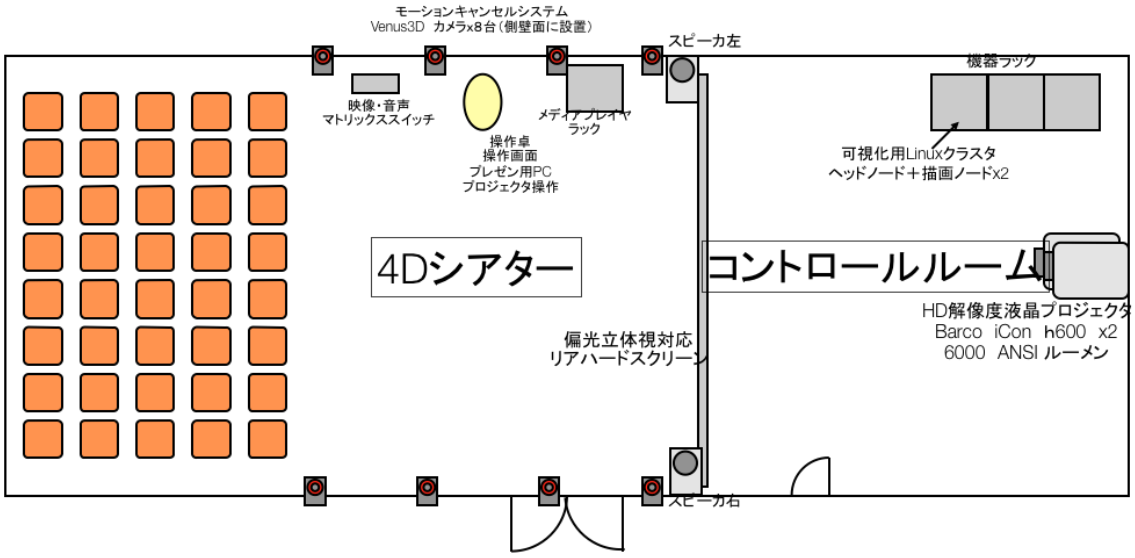


図 8.4-2　4D シアター平面図

4D シアターで映像を表示するソフトウェアとして、4DVisualizer と呼んでいる専用のソ

フトウェアを開発し利用している。4DVisualizer では 3 次元表示での視点変更に加えて表示時刻の変更を容易にすることで 4 次元データのインタラクティブな可視化を可能としている。

図 8.4-3 は野球ボールの空力シミュレーションの映像であり、時間変化をインタラクティブに操作することによりボールの縫い目がボール後流に及ぼす影響がよく理解できる。図 8.4.-4 は予めシミュレーションで求めた野球ボールの空力特性からボールの軌跡を表示し、より具体的に空力の影響を知るために利用しているコンテンツである。

図 8.4-3　野球ボール周りの空力シミュレーション

図 8.4.-4　野球場でのボール軌跡の表示

このほかにも分子動力学の計算結果や物体の断層画面を重ね合わせて
作成したボリュームデータの表示が可能となっている。

8.5 物体の大きさ知覚の差異に関する研究―HMD を使った VR 空間と現実空間での比較―

甲南大学　田村祐一

8.5.1　はじめに

(1) 背景

近年、現実空間においてコスト面、危険性等あらゆる要因で実行が困難、また不可能な実験を行うことのできる HMD (Head Mounted Display) を用いた VR 実験の需要が高まっている。また、実際に触れないものを触ってもらう用途などでも利用も期待されている。しかし、VR 実験と現実空間での実験において、同じ結果を得るためには、物体の大きさ、色、物体の影の広さ等、様々な環境での知覚を現実空間と同じにする必要がある。近年、安価な HMD が手に入るようになり、広く用いられるようになってきたが、それらの機器を使用したときの物体の知覚に関する研究は十分に行われていない。特に、HMD を用いた VR 空間と現実空間で物体の大きさの知覚にどのような差異がみられるかについては、最も重要であるにもかかわらず、十分な研究例がない。そこで本研究では、HMD と現実空間での大きさ知覚について実験を行った。

(2) 先行研究

物体の大きさ知覚の差異に関する研究として、現実空間と鏡の中の空間での物体の大きさ知覚を比較する研究[8.5-1]がある。この研究では被験者に背後に設置した物体を写した平面鏡の中の虚像と被験者の距離を示してもらう実験で、平面鏡なしよりも小さく知覚しているという結果が出ている。

このように現実空間と別の空間の知覚を検証する研究はあるが、本研究では現実空間と HMD を使用した VR 空間での大きさ知覚の差異に注目する。

8.5.2　実験方法

(1) 実験概要

本研究では VR 空間と現実空間での大きさ知覚の差異を明らかにするために以下の方法で実験を行った。

(2) 実験方法

現実空間と VR 空間に同一の物体を置き、双方が被験者から同じ距離になるよう VR 空間内の物体を移動させるタスクを行ってもらった。両眼の輻輳による奥行きの手がかりは約 20m まで有効とされている[8.5-2]が、実験するスペースの関係で今回は 1m、1.5m、2m、2.5m、3m の比較的近い距離について、現実空間と VR 空間での奥行き知覚、物体の大きさ知覚に差があるかを調べた。

(3) 現実空間での実験環境

図 8.5-1 に現実空間での実験の様子を示す。現実空間での実験には知覚させる物体に一般的に大きさになじみのあるサッカーボールの 5 号球を用い、被験者のつま先からサッカーボールの中心までの距離を認識させる。

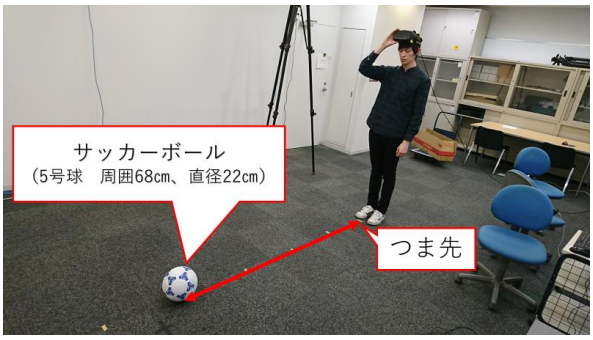


図 8.5-1　現実空間での実験の様子

(4) VR 空間での実験環境

図 8.5-2 に VR 空間のシステムを示す。VR 空間での実験環境の構築にはゲームエンジン Unity を用いた。現実空間の実験スペースと同じ 10m×10m＝100 ㎡の空間を作成し、カーペットの色、壁の色も現実空間とほぼ同じにした。被験者は HMD を装着し、被験者のつま先（図 8.5-2 における VR 空間内の黒点）から CG で描画された物体（サッカーボール）を現実空間での実験で知覚した場所にキーボードを使って移動させる。被験者が最終的に現実空間と同じとした距離を測定、記録する。

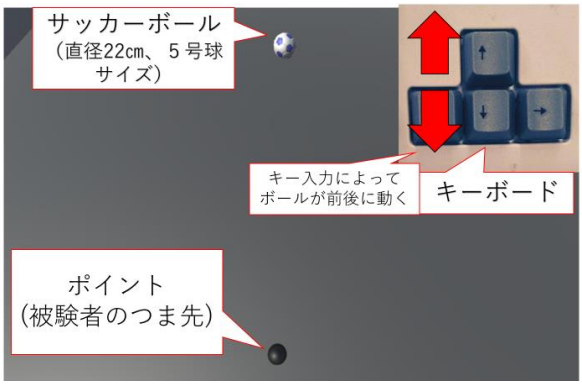


図 8.5-2　VR 空間内の実験環境

(5) 実験手順

実験は以下Ⅰ, Ⅱのプロセスを異なる距離に対して繰り返して行う。

① 現実空間で被験者に物体との距離を伝え、距離感を認識させる。

② 現実空間で知覚した位置に VR 空間上のボールを移動させ、物体の中心から被験者のつま先の距離を測定する。

距離は 3m、2.5m、2m、1.5m、1m の順に提示する。

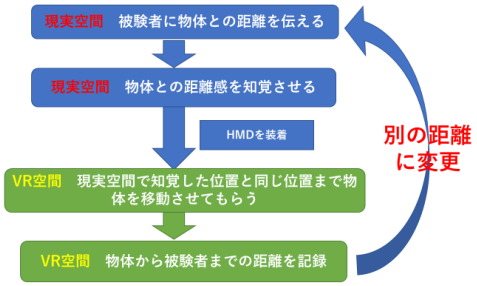


図 8.5-3　実験手順

8.5.3　結果と考察

21、22 歳の男女を含めた大学生 7 名を対象に上記の実験を行った。VR 空間で被験者が知覚した、　物体との距離を図 8.5-4、表 8.5-1 に示す。結果を見ると中央値、平均値ともに現実空間における実際の物体との距離より値が大きい、つまり、すべての距離において被験者が物体を小さく知覚していることがわかった。唯一、被験者⑦が 3m 以外のすべての距離で物体を大きく知覚したが、他の被験者の傾向から例外と考えられる。

仮想空間で小さく知覚する原因は不明だが、関連研究でも示した先行研究[8.5-1]で示されている鏡の中での知覚と同じ傾向の結果が得られた。

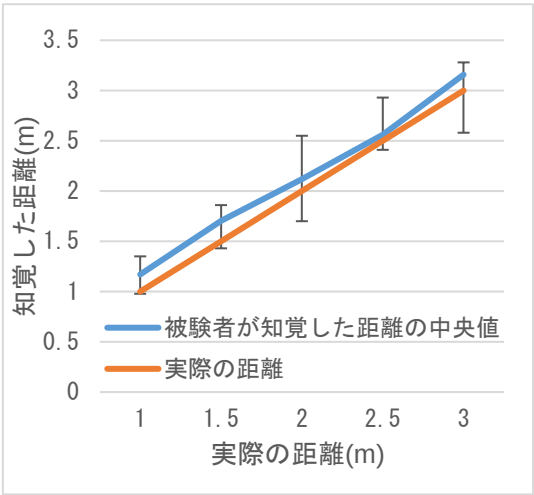


図 8.5-4　被験者と物体の距離

表 8.5-1　仮想空間で被験者が知覚した距離

現実空間での物体との距離	被験者①	被験者②	被験者③	被験者④	被験者⑤	被験者⑥	被験者⑦	平均	標準偏差
1	1.28	1.18	1.17	1.35	1.09	1.168	0.978	1.174	0.1120
1.5	1.703	1.86	1.7	1.85	1.556	1.821	1.43	1.703	0.1494
2	2	2.55	2.334	2.245	1.92	2.117	1.7	2.124	0.2611
2.5	2.52	2.93	2.59	2.681	2.41	2.562	2.42	2.588	0.1651
3	3.28	3.11	3.19	2.74	2.58	3.218	3.158	3.089	0.2485

8.5.4　まとめ

本研究では HMD で表現された VR 空間と現実空間で物体の大きさ知覚に差異が生じるかを確認する実験を行った。実験結果から HMD で表現された VR 空間内の仮想物体のほうが現実空間より物体を小さく知覚している可能性があることがわかった。HMD で表現された物体が、VR 空間で小さく知覚する原因は不明だが、今後 VR 実験、VR 研修を行う場合は大きさの知覚に差異があることを理解した上で利用する必要がある。

参考文献

[8.5-1] Mirror vision: Perceived size and perceived distance of virtual images -ATSUKI HIGASHIYAMA

[8.5-2]3 次元映像の基礎（Ohmsha）

-泉　武博　（監修）NHK 放送技術研究所　（編）

8.6 臨場感浜辺可視化、半透明ディスプレイへの表示、
モーションキャプチャデータの可視化

東京都市大学　宮地英生

8.6.1　CAVE™タイプの没入感システムでの臨場感浜辺可視化[8.6.1-1]

図 8.6.1-1 は海岸での実写画像を CAVE システムに表示した例である。ここでは視差画像は使っていないが、3 次元の物体（足元から遠方に伸びて、適当な距離で垂直な壁になるような）にテクスチャで貼り付けることで立体視している。

また、パノラマ撮影をすると円弧になる水平線を水平になるよう前処理で画像補正している。

図 8.6.1-1　浜辺臨場感システムの表示
※中央大学の樫山研究室の装置を用いている。

8.6.2　半透明ディスプレイへの表示例

図 8.6.2-1 のようにアクリル板を四角錘の形にしてディスプレイの上に配置することで立体的な表示ディスプレイを構築することができる[8.6.2-1]。

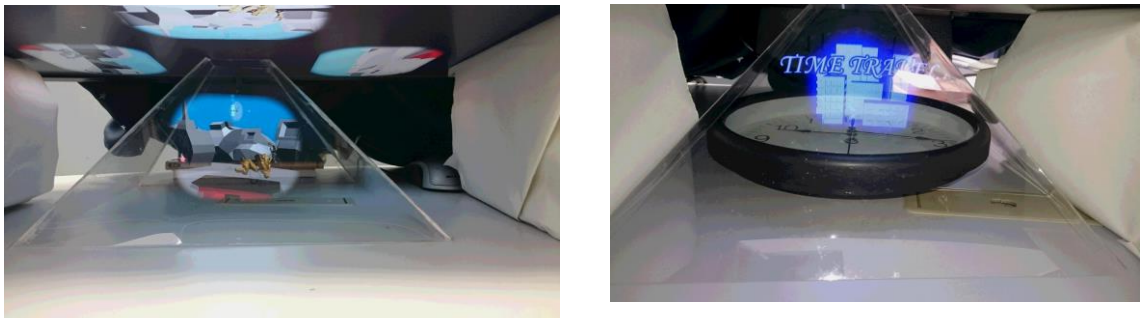


図 8.6.2-1　アクリル板を用いた 3 次元的な“ホロディスプレイ”

この表示装置を用いて、どの程度奥行き感が得られるかを簡易的に試験したが、視差を用いていないので物体が空中にあるときの奥行は表現することができなかった。しかし、オクルージョンがある場合、相対的な前後関係が解り相応の奥行き感が生じる。このとき投影方法は現実には見ることの無い斜投影であっても奥行に対する違和感は少ない (図 8.6.2-2)。

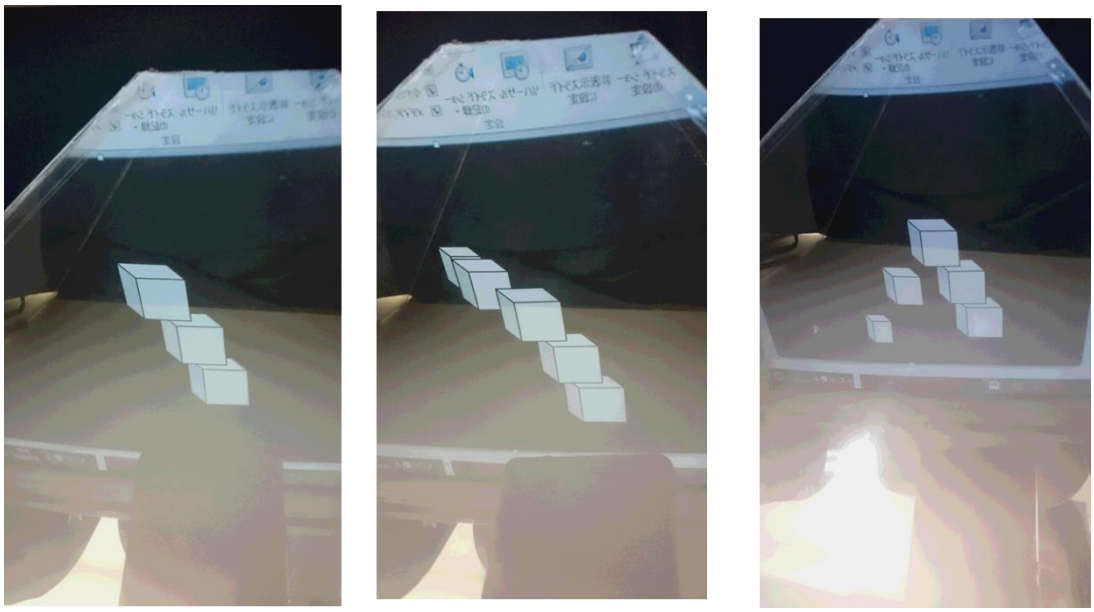


図 8.6.2-2　斜投影の立方体でもオクルージョンにより奥行き感が得られる例

8.6.3　モーションキャプチャデータの可視化例[8.6.3-1]

図 8.6.3-1 および図 8.6.3-2 にテニススイングの情報をモーションキャプチャで取得して可視化した結果を示す。図 8.6.3-1（左）のように人体には両肩、両腰、および、肘、手の甲に、ラケットには 3 点のマーカを付け初心者と経験者にテニススイング（素振り）をしてもらい、その挙動の違いを調べた。

図 8.6.3-1（右）は、取得データを 3 次元で可視化したものである。

図 8.6.3-2 は二つのスイングを比較するための可視化である。二つの時刻を合わせるためにスイングが最高速になるときを基準とし、空間的に合わせるためには腰のラインが一致するようにした。

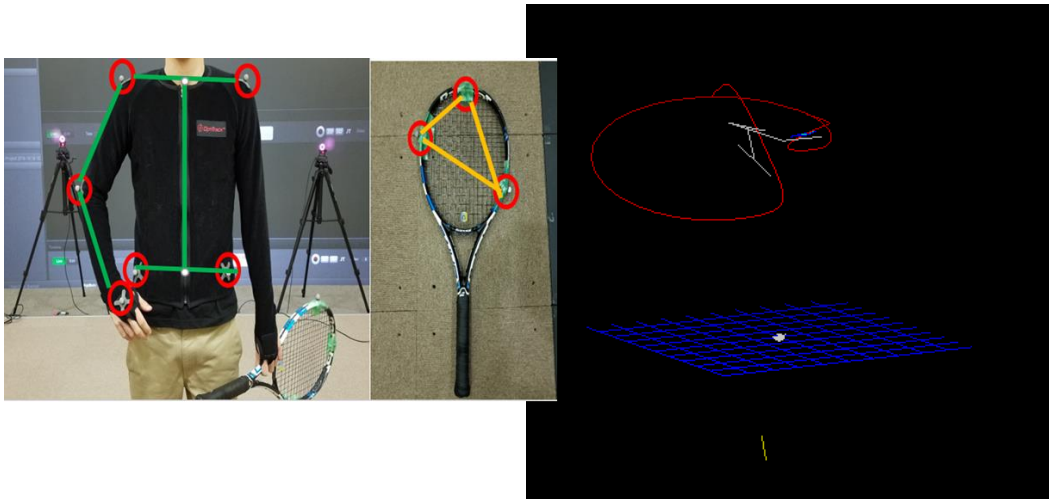


図 8.6.3-1　モーションキャプチャによるテニススイングの情報取得と 3 次元可視化

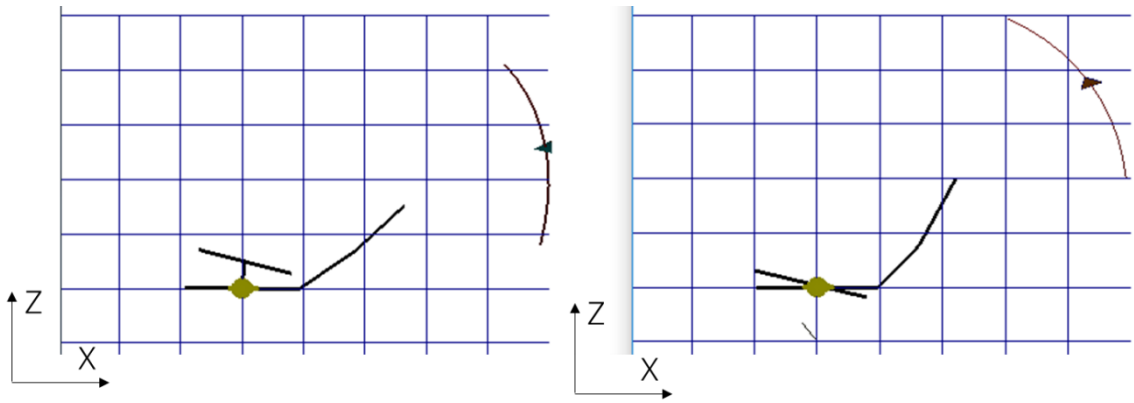


図 8.6.3-2　テニススイングの初心者と経験者のスイングの違い

参考文献

[8.6.1-1] 宮地英生、川原慎太郎、樫山和男：“VR を用いた浜辺臨場感システムの開発”、CAVE 研究会、2017 年 3 月

[8.6.2-1] 根岸菜摘, 吉田孟弘, 宮地英生: “ペッパーズゴーストを用いた拡張現実の教育利用” 第 21 回日本バーチャルリアリティ学会大会（講演番号 12E-02)、2016 年 9 月（筑波）

[8.6.2-2] 宮地英生:“半透明ディスプレイ表示装置への可視化表示”、可視化情報全国大会（日立）,2016

[8.6.3-1]森優也：”テニス習熟度から発生するフォアハンドストロークの相違点”、2018 年度東京都市大学メディア情報学部情報システム学科卒業論文、2018.3

8.7 航空交通流のインタラクティブ可視化

大阪大学　安福健祐

京コンピュータの後継機ポスト「京」の開発事業（フラグシップ 2020 プロジェクト）では、新たに取り組むチャレンジングな課題として萌芽的課題 4 テーマが設定され、2016 年に 8 つの課題が決定された。大阪大学サイバーメディアセンターでは、萌芽的課題(2)「複数の社会経済現象の相互作用のモデル構築とその応用研究」の一つである「堅牢な輸送システムモデルの構築と社会システムにおける最適化の実現」（課題責任者：東京理科大学 藤井孝藏教授、研究期間：2016 年 8 月 1 日～2020 年 3 月 31 日）において、可視化プログラムの研究開発を行っており、その事例を紹介する。

本課題で対象としている輸送システムとは、現時点では航空機を対象としている。近年そのシステムは複雑化し、混雑や乗客のトラブル、気象の影響などに対して脆弱となり、日常的に遅延が生じている。本課題は、個別の遅延やトラブルへの効果的な対応の策定を越えて、一定規模の地域や国内全体（さらに将来には国際社会）の大規模輸送手段を一つのシステムと捉え、相互作用を考慮した上で全体最適を実現、加えてトラブル時への対処が容易な堅牢性を有する運行方式によって安全性と効率性という相反する要求を両立する手法の確立を目指している。

本プロジェクトにおける可視化プログラムの特徴は、大規模な航空交通流データを高いフレームレートを維持しながらリアルタイムレンダリングし、インタラクティブな操作によって必要なデータを絞り込み、詳細に分析を行うことである。開発環境は C++をベースにグラフィックスライブラリとして OpenGL、GLFW、ImGui、NFD （Native File Dialog）を利用している。以下、可視化プログラムを用いた分析のワークフローを説明する。

8.7.1　可視化プログラムの分析ワークフロー

可視化プログラムを起動すると、画面にはメインメニューと 3D 地球儀モデルが表示される。3D 地球儀モデルはマウス操作によって任意軸方向の回転、拡大縮小が行える。また 3 次元空間内のカメラを自由に回転して傾けることもでき、自由な角度から 3D 地球儀モデルを表示できる。カメラの持つパラメータとしては投影方法（透視投影または平行投影）、焦点距離がありメインメニューから変更ができる。3D 地球儀モデルの表示オプションとして空港名と位置、ウェイポイント名と位置の表示・非表示をメインメニューから選択できる。

分析はまずメインメニューから航空交通流データのファイルを読み込む。ファイルは CSV 形式で CARATS Open Data[8.7-1]のフォーマットをベースとしており、時刻に対する便名と航空機の型式および緯度、経度、高度の位置情報からなる。データ読み込み時には全航空機の軌跡データが時系列で生成されるが、コンピュータのメモリ量に応じてそのタイムステップを 1 分単位、2 分単位、4 分単位の 3 段階で指定できる（デフォルトは 2 分単位）。また読み込みデータに指定されたタイムステップのデータが含まれない場合は線形補間が行われる。データが読み込まれると、先頭データの時刻における航空機の位置データから 3D 地

球儀モデル上に 3D 航空機モデルが表示され、指定された時間単位で位置データを更新する（初期設定は画面が 1 フレーム進むごとに 1 分経過する。フレームレートが 60 fps の場合は 60 倍の再生速度となる)。ファイルが読み込まれるとサブメニューが表示されており、時系列データの範囲内でのタイムラインがあり、再生、停止、早送り、巻き戻し、任意の時刻への移動が可能となる。前述のマウスの操作によって自由なカメラ視点で航空交通流を表示できる。カメラを傾ければ航空機同士の高低差が視認できる。初期画面は 3D 地球儀モデル全体が表示されているが、画面を拡大して羽田空港周辺を表示すれば、空港内の道路や建物も衛星写真で表示でき、その上空の航空交通流を確認できる。マクロな視点からミクロな視点へ連続的な切り替わるときには、3D 航空機モデルのスケールは動的に変化させて航空機の状況を把握しやすいようにし、空港レベルのミクロな視点にくると 3D 航空機モデルが実スケールで表示される。このようにして 3D-CG をベースに航空機の高低差を確認できるとともに、各航空機を常に視認しやすいスケールを自動調整することで直観的な航空交通流の理解を支援する。

可視化プログラムには航空交通流を詳細に分析する主な機能として各航空機の航跡表示がある。表示オプションとしては、データに含まれている全航跡の表示と、表示中の時刻から遡った「任意の時間単位の長さ」での航跡表示がある。航跡を表示することで、ベクタリングやホールディングの発生状況が把握できる。特に時間単位の航跡表示は、近接する航空機同士の近接状態などを視覚的かつ定量的に表現することができる（図 8.7.1-1)。

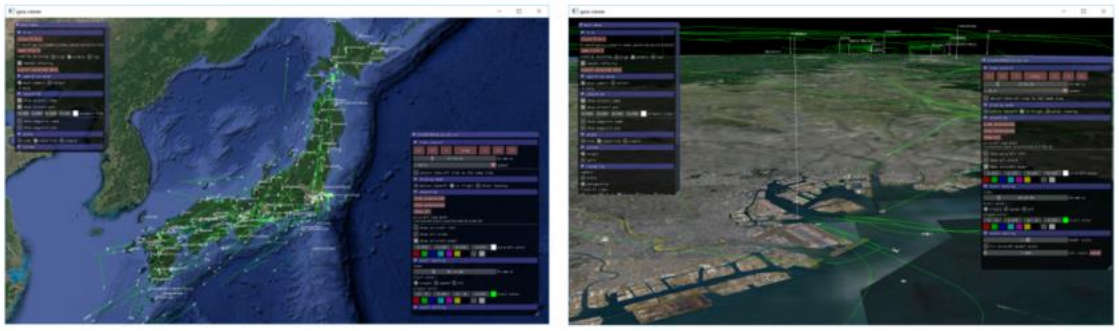


図 8.7.1-1　可視化プログラムの画面

大規模な航空交通流データから特定の航空機を抽出して表示するため、可視化プログラムでは任意の航空機の選択機能と表示・非表示機能がある。航空機の表示・非表示は、大きく「出発前」「フライト中」「到着後」の 3 パターンで切り替えることができる。その中からさらにマウスによって選択した航空機の表示・非表示が設定できる。選択操作は、画面のある範囲をマウスでドラッグすることで、その矩形範囲内にある航空機に対し、「選択」「追加選択」「選択解除」を行う。この選択操作の有用な使い方として、特定の空港の出発機、到着機のみを表示することが挙げられる。例えば、羽田空港への到着機のみを表示したい場合、タイムラインを最後まで移動させて、「到着後」の全航空機を表示させる。次に羽田空港に

到着した航空機をマウスにより選択した後、表示する航空機を「フライト中」に変更することで、羽田空港への到着機のみが表示される。CARATS Open Data 自体には、出発空港、到着空港の情報が含まれていないが、可視化プログラムの GUI 操作によって画面上で容易に航空機をフィルタリングすることができる。表示している航空機のデータのみを保存しておきたい場合は選択した航空機のみファイルの出力も可能である（図 8.7.1-2（左側））。

可視化プログラム上で行うことができる視覚的な分析方法として、シミュレーションデータと実データの比較、シミュレーション上での最適化前後の比較がある。そのために可視化プログラムは二つのファイルを同時に開くことができる。二つのファイルを開くと、3D 航空機モデルは一つの 3D 地球儀モデル上に重なって表示されるが、ファイルごとに別々のサブメニューが表示されている。サブメニューには、タイムライン、航空機と航跡の表示・非表示およびカラー設定があり、色分けして表示することで、一つの画面で二つのデータの軌跡を比較することができる。メインメニューのほうには、共通のタイムラインが表示されており、こちらで操作をすると二つのデータを同時にコントロールすることができる。さらに画面は左右に 2 分割することができ、それぞれの画面に二つのデータを表示して比較する機能がある（図 8.7.1-2（右側））。オプションによって、左右のカメラの位置を別々に操作するか、同時に操作するかを選択できる。画面分割状態で一つのファイルを開いたときは、左右に同じデータが表示されることになり、異なる視点から同じデータを見て分析するという使い方もできる。

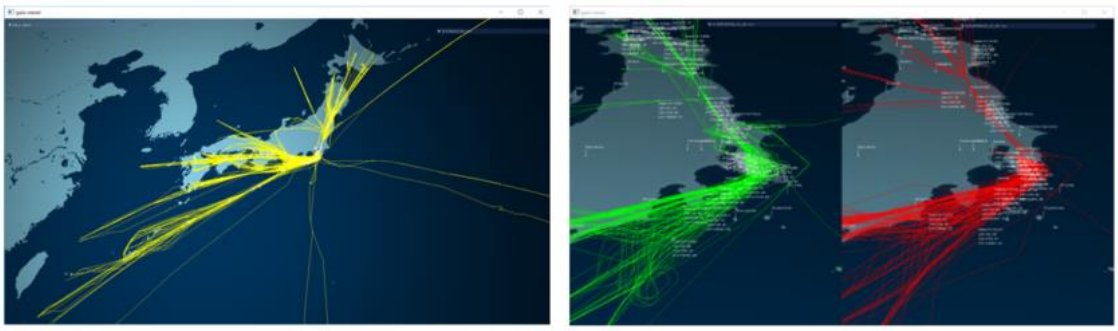


図 8.7.1-2　可視化プログラムの機能
（左：到着空港によるフィルタリング、右：二つのファイルの比較）

8.7.2　タイルドディスプレイによる大規模可視化

航空交通流を大画面で高精細に可視化するための表示装置としてタイルドディスプレイを用いる。タイルドディスプレイとは、複数のディスプレイを格子状に配置して一つの高解像度ディスプレイを構築する技術である。今回使用するシステムは、水平 150 度程度の超広視野を有する 1920×1080（フル HD）50 インチプロジェクションモジュール 24 面（縦 4 面、横 6 面）およびバックエンドで可視化処理を行う画像処理用 PC 6 台と制御用 PC 2 台で構成されており、24 面大型立体表示システムと呼んでいる（図 8.7.2-1）。画像処理用 PC1 台

から縦 4 面のディスプレイ用の画像を生成し、より没入感を与えるためアクティブ方式のステレオ立体視表示に対応する。描画処理では PC 間の同期処理が必要となり、CAVELib[8.7-2]とよばれるライブラリを利用する。

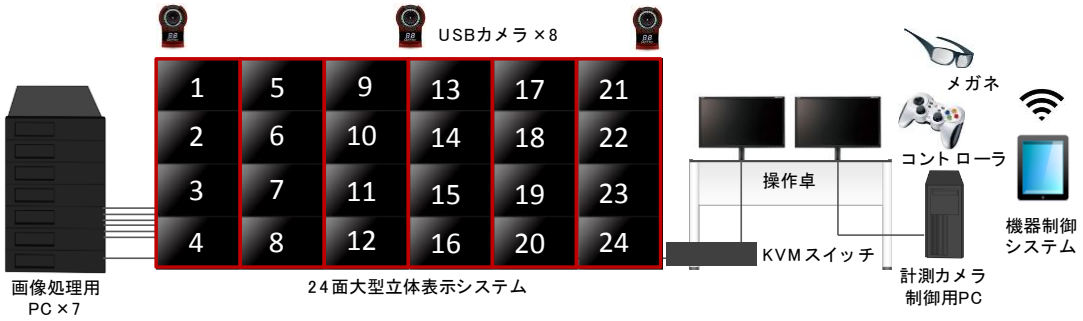


図 8.7.2-1　24 面大型立体表示システムの構成

図 8.7.2-1 の 24 面大型立体表示システムを用いて CARATS オープンデータの航空機軌跡を表示した結果を図 8.7.2-2 に示す。シミュレーション結果は、11,520×4,320 ピクセルのドットバイドットで秒間 60 フレームのレンダリングが行われている。一般的なディスプレイよりも大型で 12K という高解像度のタイルドディスプレイウォールを用いることで、画面に日本全体を表示しても、視野全体で個々の航空機の動きを直観的に把握することができる。また、ステレオ立体視によって平面図の地図を表示しても、航空機の高低差を立体視によって確認できる。このように 24 面大型立体表示システムを用いることで、シミュレーションの専門家、航空管制の専門家など、多人数で航空交通流の可視化結果を確認しながら、データの検証から航空機遅延の原因をデータ探求するための環境を構築している。

図 8.7.2-2　24 面大型立体表示システムによる可視化

参考文献

[8.7-1] CARATS Open Data: http://www.mlit.go.jp/koku/koku_fr13_000006.html

[8.7-2] CAVELib: https://www.mechdyne.com/software.aspx?name=CAVELib

8.8 海洋研究開発機構における VR 可視化

国立研究開発法人海洋研究開発機構　川原慎太郎

海洋科学技術センター（現 海洋研究開発機構）では、CAVE[8.8-1]型バーチャルリアリティ装置“BRAVE”を 2003 年に導入した（図 8.8-1）。2002 年に運用を開始したスーパーコンピュータ「地球シミュレータ」上でのシミュレーションにより出力されたデータには、複雑な 3 次元構造を有する種々の現象が含まれており、これらを立体映像として直感的に理解し、データ解析の一助とすることを目的として導入したものである。2018 年度末をもってその運用は既に終了しているが、当機構における VR 可視化のこれまでとこれからとして、ここで御紹介させて頂く。

図 8.8-1　CAVE 型バーチャルリアリティ装置「BRAVE」の外観

本装置は、一辺 3 メートルの正方スクリーンを壁 3 面、床 1 面の計 4 面となるように立方体状に配置したものであり、各面には対応するプロジェクタから背面投影（壁面）または直接投影（床面）により、時分割方式で立体映像が投影される。スクリーン上に投影された時分割方式の映像の分離には液晶シャッター眼鏡を用いる。スクリーン上に投影された左眼用・右眼用映像の切り替えタイミングと、液晶シャッターの開閉タイミングを同期させることにより、眼鏡装着者の左眼では左眼用映像のみが、右眼では右眼用映像のみが視認可能となる。液晶シャッター眼鏡および手持ちコントローラ（図 8.8-2）には再帰性反射塗料を塗布したマーカが取り付けられており、複数の赤外線カメラで撮影することで、それぞれの 3 次元位置および傾きをリアルタイムで検出することが可能となっている。特に、液晶シャッター眼鏡に取り付けたマーカでの頭部位置の検出結果は、眼鏡装着者の頭部位置および視線方向から適切な視差映像をリアルタイムに生成するために用いられる。本装置は、複数人での同時利用も可能ではあるが、立体映像を正しく融像できるのは、マーカを取り付けた

液晶シャッター眼鏡（マスター眼鏡）を装着した一人のみとなる。

図 8.8-2　BRAVE 用液晶シャッター眼鏡と手持ちコントローラ
2 台ある液晶シャッター眼鏡の内、マーカが取り付けられたものがマスター眼鏡となる。

図 8.8-3 は CAVE 型バーチャルリアリティ装置用可視化ソフトウェア“VFIVE”[8.8-2]を使ったシミュレーションデータの可視化の様子である[8.8-6]。VFIVE には入力データ（スカラー場、ベクトル場）に対する主要な可視化アルゴリズムが実装されており、メニューパネルからの可視化アルゴリズムの選択や可視化パラメータの変更といった各種操作を、装置外に出ることなく手持ちコントローラを用いてインタラクティブに行うことができる。本図中では計算領域中のベクトル場に対して、コントローラを用いて流跡線の描画開始点（3 次元位置）を決定している。

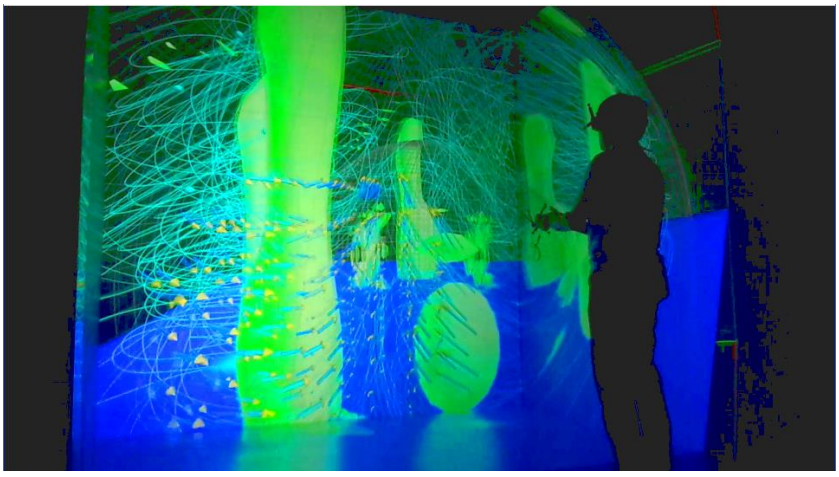


図 8.8-3　BRAVE 上で動作する VFIVE

VFIVE を含め、本装置で動作するソフトウェアの多くは、商用の CAVE 用アプリケーション開発ライブラリ“CAVELib”[8.8-3]を用いて独自に開発したものである。C++プログラム

中で使用することで、2-D および 3-D グラフィックス API である OpenGL を用いた描画結果を CAVE 装置上に投影することができる。

このような既開発のソフトウェア資産を、本報告書中でも紹介した安価なヘッドマウントディスプレイ（HMD）でも活用することを考えた場合、その選択肢は二つである。一つはゲームエンジンを用い、元アプリケーションのエッセンスを取り込みつつ移植する方法と、もう一つは HMD ベンダーの提供する C++用ライブラリを用いることで、極力元のソースコードを活かしつつ移植する方法である。HMD 用アプリケーションを新規に開発する場合、本報告書でまとめたように Unity や Unreal Engine といったゲームエンジンを用いるのが現在の主流であることは言うまでもない。しかしながら、当機構のように CAVELib を用いて記述された C++プログラム（以下 CAVELib プログラム）を多数運用している場合、それら全てをゲームエンジンベースで個別に書き直すことは容易では無い。一方、主要な HMD ベンダーは、ゲームエンジン用アセットと併せて C++用ライブラリも公開しており、こちらを用いる方が CAVELib プログラムの移植への親和性は高いと考えた。主要な PC 接続タイプの HMD と、HMD ベンダーの提供する C++用ライブラリ（以下 HMD 用ライブラリ）を表 8.8-1 に示す。

表 8.8-1　主要な PC-HMD における C++用ライブラリの対応状況

PC-HMD の種類	C++用ライブラリ
Oculus Rift / S （/DK2）	Oculus SDK / OpenVR
HTC VIVE / VIVE Pro	OpenVR
Windows MR	OpenVR

現在、HMD 用ライブラリとしては、Oculus 社の提供する Oculus SDK[8.8-4]と、Valve 社の提供する OpenVR[8.8-5]の二つが主流である。Oculus SDK が Oculus 社製 HMD 専用であるのに対し、OpenVR はネイティブ対応の HTC 社製 HMD だけでなく、Oculus 社製 HMD および Windows Mixed Reality 規格のヘッドセット（Windows MR）にも対応機器を拡大し、現在主要な HMD 全てに対応した HMD 用ライブラリとなっている。当機構では、これらの HMD 用ライブラリを用いて CAVELib の内部動作をエミュレートする互換ライブラリ“CLCL（CAVELib Compatible Library for HMD)”を開発した[8.8-6]。本ライブラリは、CAVELib の主要な関数群について、その内部を HMD 用ライブラリや OpenGL ツールキット等で再実装することにより、CAVELib と同名の関数コールで同等の機能を再現するものである。HMD 用ライブラリが提供する機能は、HMD の初期化やトラッキング情報の取得など、デバイス制御に関わる部分のみとなるため、グラフィックス制御などそれ以外の部分については GLFW や GLEW といった OpenGL ツールキットなど、フリーの C++ライブラリを用いることにより CAVELib 関数の挙動を再現できるよう、その内部を再実装している。現在、CLCL には二つのバージョンがあり、Oculus SDK を使って開発した Oculus 社製 HMD 専用のバージョンと、OpenVR を使

用して開発した、表 8.8-1 掲載の全ての HMD で動作するバージョンを並行して開発している。私見ではあるが、HMD として Oculus 社製 HMD を使用する場合には、Oculus SDK 版 CLCL を用いた方が良いと思われる。図 8.8-4 は、CLCL を用いてビルドした VFIVE の HMD での実行の様子である[8.8-6]。VFIVE の特徴である手持ちコントローラを使った可視化操作を、主要な HMD それぞれにおいて 6 自由度の手持ちコントローラを使って BRAVE での操作時と同様に行えていることがわかる。

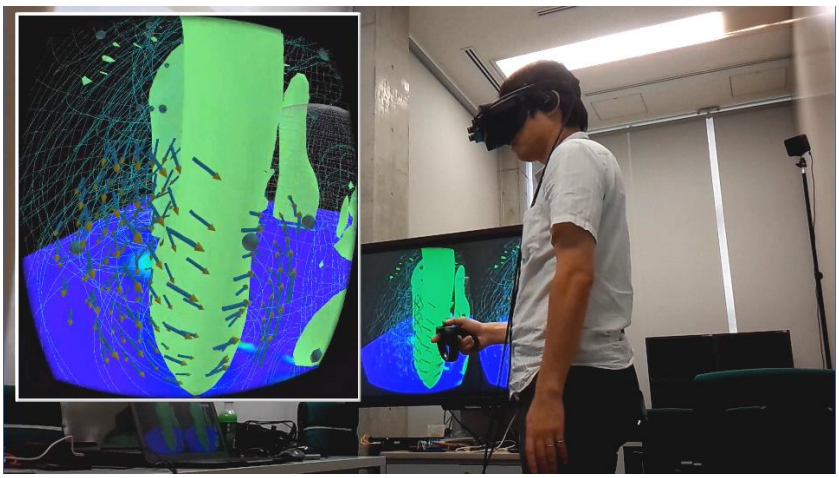


図 8.8-4　HMD で動作する VFIVE

また、CLCL では独自機能として、HMD 前面にステレオカメラを装着することにより、そこで取得した外部映像をアプリケーションの描画する 3-D CG と併せて HMD 内に表示するビデオスルー機能を試験的に実装している。現在対応しているステレオカメラは、しのびや製“Ovrvision”および“Ovrvision Pro”[8.8-7]と、StereoLabs 社製“ZED Mini”[8.8-8]の 3 機種である。これらの内 ZED Mini については OculusSDK 版 CLCL のみでの対応となる。Ovrvision/Ovrvision Pro を使用する場合、カメラで取得した実映像を背景とし、その上に 3-D CG をオーバーレイ表示することができる。一方、ZED Mini を使用する場合、ベンダー提供の SDK に含まれる、視差画像からのデプスマップ算出機能を用いて、3-D CG と実映像を MR (Mixed Reality) 表示することができる。図 8.8-5 は ZED Mini を装着した Rift、図 8.8-6 はその際の MR 表示の様子である[8.8-6]。

図 8.8-5　ZED Mini を装着した Rift

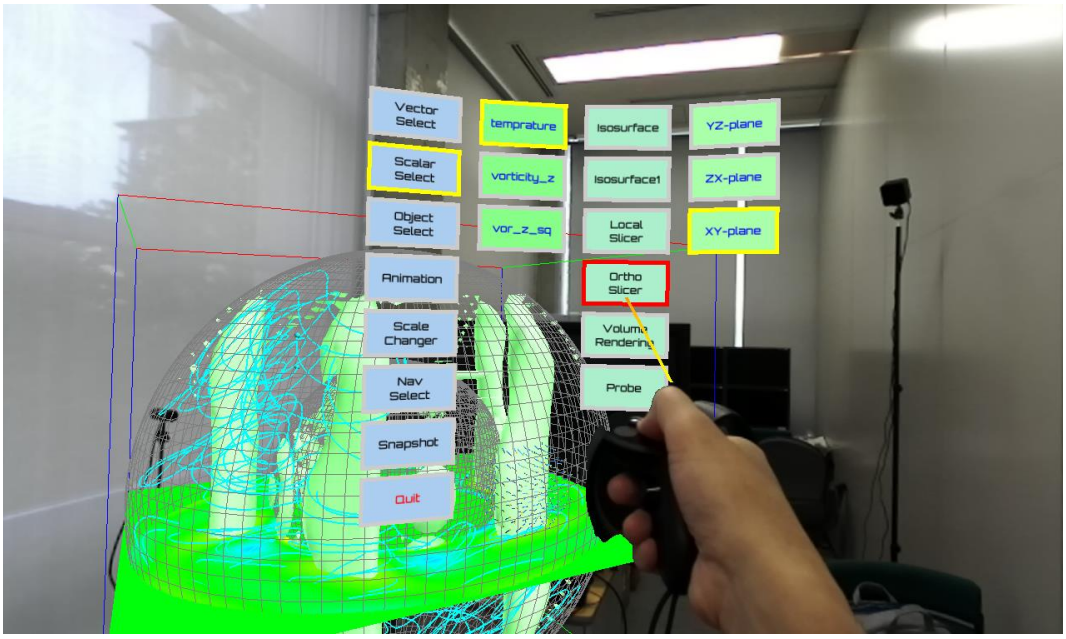


図 8.8-6　ZED Mini 装着時の MR 表示の様子

当機構で開発した VFIVE 以外の CAVELib プログラムだけでなく、他機関で開発された CAVELib プログラムについても、概ね元のアプリケーションと同様の動作を HMD でも再現できることを確認しているが、ここでの紹介は割愛させて頂く。CLCL を用いた移植にあたり、元の CAVELib プログラムからの変更箇所はいずれもわずか数行と軽微なものであった。現在、CLCL についてはソースコードを GitHub にて公開している[8.8-9]。また、VFIVE についても当機構にてソースコードを公開しており[8.8-10]、HMD 用 VFIVE のコンパイル方法についても CLCL の公開ページで詳述しているので参考になれば幸いである。

参考文献

[8.8-1] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, J. C. Hart: The CAVE: Audio Visual Experience Automatic Virtual Environment, Communications of the ACM, 35:6 (1992), 64-72.

[8.8-2] A. Kageyama, Y. Tamura, T. Sato: Visualization of Vector Field by Virtual Reality, Progress of Theoretical Physics Supplement, 138 (2000), 665-673.

[8.8-3] CAVELib: https://www.mechdyne.com/software.aspx?name=CAVELib

[8.8-4] Oculus SDK for Windows: https://developer.oculus.com/downloads/package/oculussdk-for-windows/

[8.8-5] OpenVR: https://github.com/ValveSoftware/openvr/

[8.8-6] S. Kawahara and A. Kageyama: Development of CAVELib Compatible Library for HMD-type VR Devices, Journal of Advanced Simulation in Science and Engineering, Vol.6, No.1, pp.234-248 (2019)

[8.8-7] Ovrvision/Ovrvision Pro: http://ovrvision.com/

[8.8-8] ZED Mini: https://www.stereolabs.com/zed-mini/

[8.8-9] CLCL: https://github.com/kawaharas/CLCL

[8.8-10]VFIVE: http://www.jamstec.go.jp/ceist/aeird/avcrg/vfive.ja.html

8.9 PBVR を利用した In-Situ 可視化フレームワーク

日本原子力研究開発機構　河村拓馬

原子力分野のアプリケーションに対する In-Situ ウォークスルー可視化に向けて、PBVR を利用した In-Situ 可視化フレームワークである“In-Situ PBVR”が開発されている。In-Situ PBVR は商用 HMD である Oculus Rift に対する機能開発が行われている。図 8.9-1 に In-Situ PBVR フレームワークの概要を示す。この手法では、In-Situ 環境下で結果データを十分小さな可視化用粒子データに圧縮し、対話処理可能なノード上で起動されたプログラムがストレージ上の粒子データを集約し、クライアントとなる PC に転送する。この In-Situ PBVR はポリゴンベースの従来的な並列可視化手法と比べて以下の点で独自性がある。

(1) 転送した粒子データにより、シミュレーション実行時に自在な視点変更
(2) インターネット回線であっても転送可能な数十 MB のサイズの粒子データ
(3) 並列化された粒子生成処理のストロングスケーリング性能
(4) ファイルベースの対話的 In-Situ 可視化制御

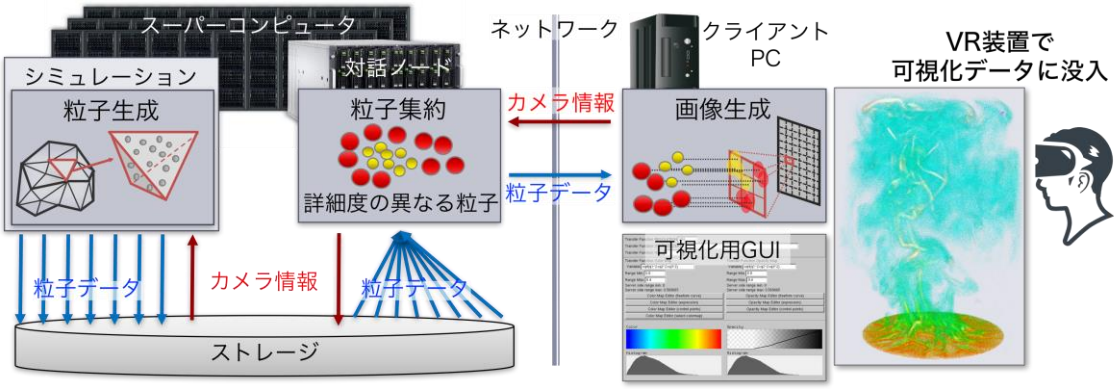


図 8.9-1　In-Situ PBVR フレームワークの概要

In-Situ PBVR を最新のメニーコア CPU である Xeron Phi 7250 (KNL) が搭載された東大/筑波のスーパーコンピュータ Oakforest-PACS (OFP) に移植し、SIMD 最適化を実施した。燃料溶融複雑系解析コード JUPITER に結合し、OFP の約 10 万コアまでを使用して、強スケーリング試験を実施、ソルバ及び最適化前後の In-Situ PBVR の性能を測定した。JUPITER は 3 次元の領域分割された構造格子上の MPI/OpenMP ハイブリッド並列モデルによって非圧縮流体モデルと Volume of Fluid 法に基づく多相多成分熱流動解析を処理する。典型的なシミュレーション時間は約 300 万タイムステップであり、本来は 1000 ステップ毎に可視化されていが、本実験では毎タイムステップに In-Situ PBVR による可視化を行なった。問題規模は 240x240x1920（約 1 億格子）に固定し、可視化用粒子は約 1000 万粒子（約 250MB)、画面解像度は 1024x1024 を使用した。本実験において、1 ノードあたり 64 コアを

使用し、スレッド数を 16、プロセス数を 4 に固定した。

可視化結果を図 8.9-2 に、強スケーリング試験の結果を図 8.9-3 に示す。In-Situ PBVR は約 10 万コアまでの強スケーリングを達成した。In-Situ PBVR の性能は SIMD 最適化により 10 倍以上高速化し、その可視化コストはソルバの 1%未満に抑えられた。この結果から、将来のエクサスケールシミュレーションに対する In-Situ PBVR の適用が期待される。

開発した機能を利用して、JUPITER による圧力容器下部ペデスタルへの燃料溶融物（ウラン）の落着の計算を可視化した。図 8.9-4 は、圧力容器内部のコンクリート壁を灰色に設定し、ウランの形状に温度分布を色でマッピングし、断面をとって可視化した結果である。ペデスタル落下後のウランは温度が下がるが、くぼみに流入して溜まることによって再臨界し、再び温度が上がっていることが確認できる。

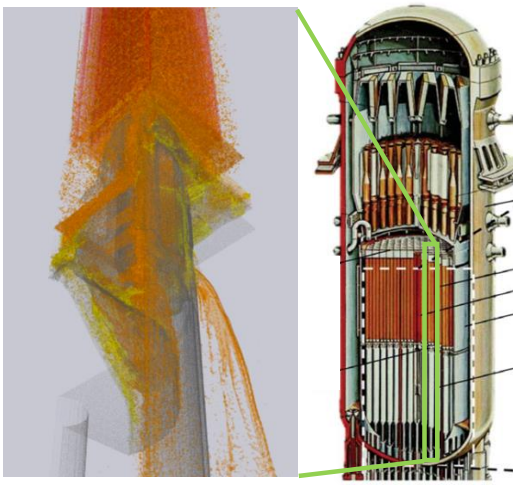


図 8.9-2　左：約 1 億格子の熱流動計算（圧力容器内部の制御棒における燃料溶融）の可視化結果、右：圧力容器の模式図

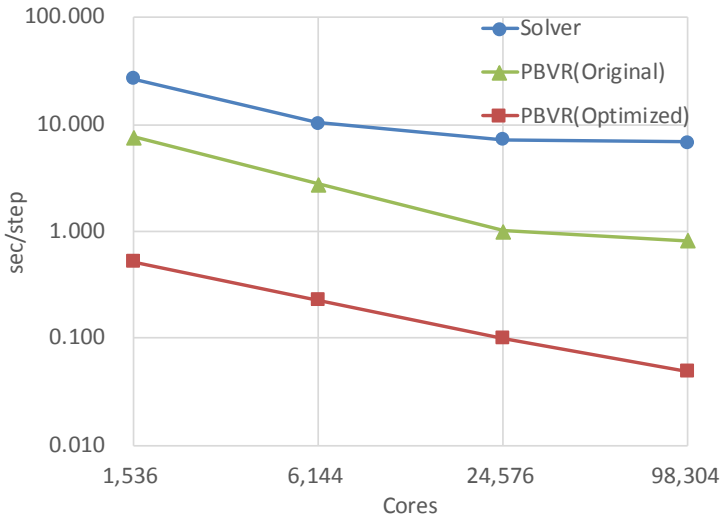


図 8.9-3　JUPITER（Solver）、In-Situ PBVR（PBVR）における強スケーリング試験の結果

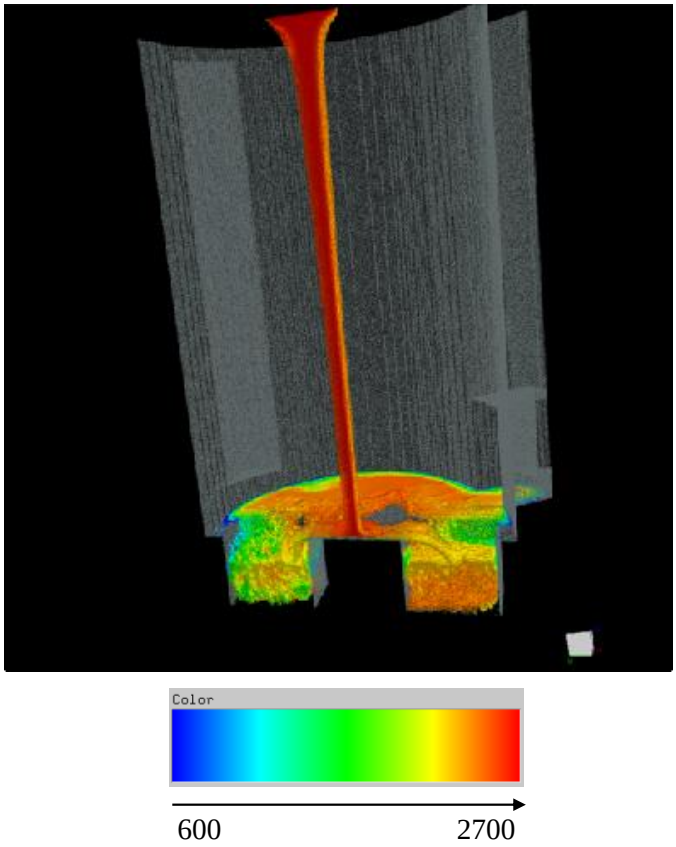


図 8.9-4　上：圧力容器内壁（灰色）と圧力容器下部に落着する燃料溶融物の可視化結果、下：燃料溶融物に割り当てられた温度のカラーマップ

8.10　核融合科学研究所における VR 可視化

核融合科学研究所　大谷寛明

（核融合科学研究所 30 年史及び[8.10-1]より抜粋し、加筆・修正した）

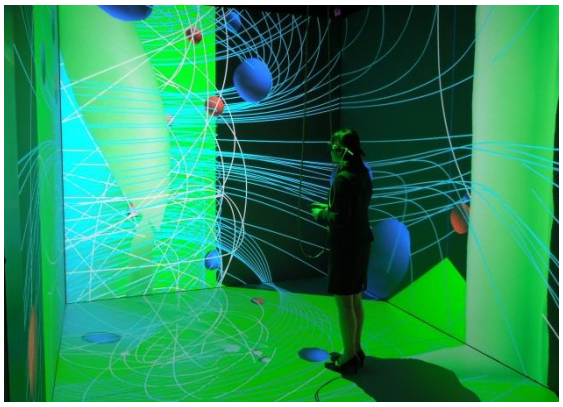


図 8.10-1　CompleXcope による磁力線の再結合のシミュレーション結果の可視化。青線と白線はそれぞれ磁力線とイオンの粒子軌道を，赤球と青球は磁力線の出発点と終点を，緑の面はプラズマ圧力の等値面を表す。

核融合科学研究所は図 8.10-1 のような没入型バーチャルリアリティ（VR）装置“CompleXcope（コンプレックスコープ)”を 1997 年に導入した。科学的な可視化を行うことを目的として日本で最初に導入された装置である。この装置は、ヘッドマウントディスプレイ（HMD）と違って、立体映像が投影された大きなスクリーンで部屋を囲っている。そのため、多人数で一緒にひとつの VR 空間に入ることができるので、同時にひとつのモノを見ながら議論を行うことができる。大学共同利用機関として共同研究を進めることに非常に役立っている。導入以来、シミュレーションデータの可視化だけではなく、医学分野や心理学分野への応用なども行ってきた。ここでは、シミュレーションと実験装置データの同時可視化と時系列データの VR 可視化の研究、及び実験データと計算データを融合した VR 空間での表示、ヘリカル型原型炉のデータの VR 装置を使った可視化について紹介する。

複雑な振る舞いを示す高温プラズマの中で、3 次元的に複雑な構造を示す磁場や荷電粒子の運動の様子を様々な視野方向から観測してより直観的に現象を探索するため、VR 可視化研究を推進している。また、核融合研では将来の核融合炉の設計研究を行っており、炉の組み立てやメンテナンスの工程を VR 空間で確認して、設計研究への貢献も行っている。導入している可視化ソフトウェアを表 8.10-1 にまとめる。

表 8.10-1　CompleXcope で利用可能な可視化ソフトウェア

ソフトウェア名		目的	備考
CAVELib		CAVE システムで稼働するプログラムを作成するためのライブラリ。陰山聡氏と佐藤哲也氏によるプログラミングガイドがある[8.10-2]。	
CAVELib プログラム	VFIVE	CAVE システムでシミュレーション結果などを解析するための汎用 VR 可視化ツール。ダウンロード先は[8.10-4]。	[8.10-3]
	Virtual LHD	LHD の平衡プラズマを CAVE システムで可視化するためのプログラム。磁力線追跡、プラズマ等圧面、粒子軌道追跡、ダスト粒子の実験結果を表示することができる。また、粒子軌道追跡では立体音響装置を使って飛行する粒子からドップラー効果を伴って音を出すこともできる。	[8.10-5, 8.10-6, 8.10-7, 8.10-8]
AVS/ExpressMPE		AVS/Express の Multiple Pipeline Edtion。	[8.10-9, 8.10-10]
EasyVR、FusionVR		市販の CAD ソフトウェアの可視化データを VR 装置に表示するためのソフトウェアおよび二つ以上のソフトウェアによる可視化データを一つの VR 空間に統合して VR 装置に表示するためのソフトウェア。EasyVR にはオブジェクトを VR 空間でつかんだり、衝突判定を行う機能などがある。	[8.10-11]
Unity			
MiddleVR		Unity で開発した可視化ソフトウェアを CAVE システムで表示するためのミドルウェア。	[8.10-12]
CLCL		CAVELib を使って開発された CAVE 用 VR ソフトウェアを HMD で稼働するためのラッパーライブラリ。	[8.10-13]
VirDSE		旭エレクトロニクス社製の可視化ソフトウェア。ネイティブな 3 次元 CAD データを直接取り込んで、高精細な描画をリアルタイムで行う。質感評価や視認性確認、組付け検証などを行う機能がある。VR 装置に表示することもできる。	[8.10-14]

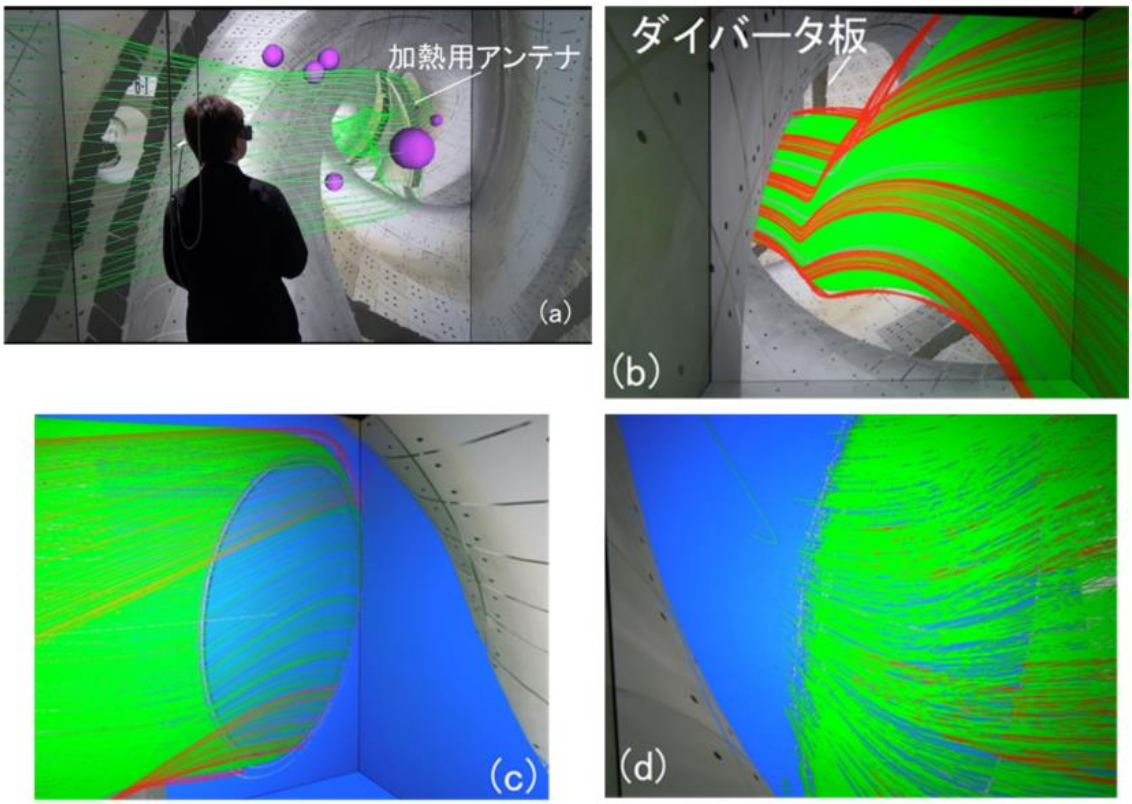


図 8.10-2　大型ヘリカル装置（LHD）プラズマのシミュレーション結果を実験装置の中に表示。プラズマを加熱するためのアンテナやダイバータ板が図(a)と(b)に表示されている。図(a)のマゼンタの球はプラズマに閉じ込められた粒子である。緑と赤の線はそれぞれ 1 本の磁力線を表している。青い面の上には磁力線のポアンカレ断面を表示している。図(c)の緑の磁力線は閉じた磁気面を、赤の磁力線は磁気島を形成している。それらの 3 次元的な構造を図(b)のように見ることができる。図(d)ではストカスティック領域を形成していることがわかる。

図 8.10-2 は、核融合科学研究所　大型ヘリカル装置（LHD）の平衡プラズマのシミュレーション結果を LHD 真空容器の内部に表示した結果である[8.10-6, 8.10-7]。真空容器の内部及び LHD 周辺装置は Unity を使ってレンダリングを行っている。その際に実験装置の設計で使われているデータを使っているので、プラズマを加熱するためのアンテナやダイバータ板の配置、真空排気装置や NBI 加熱装置も忠実に再現している（図 8.10-3)。

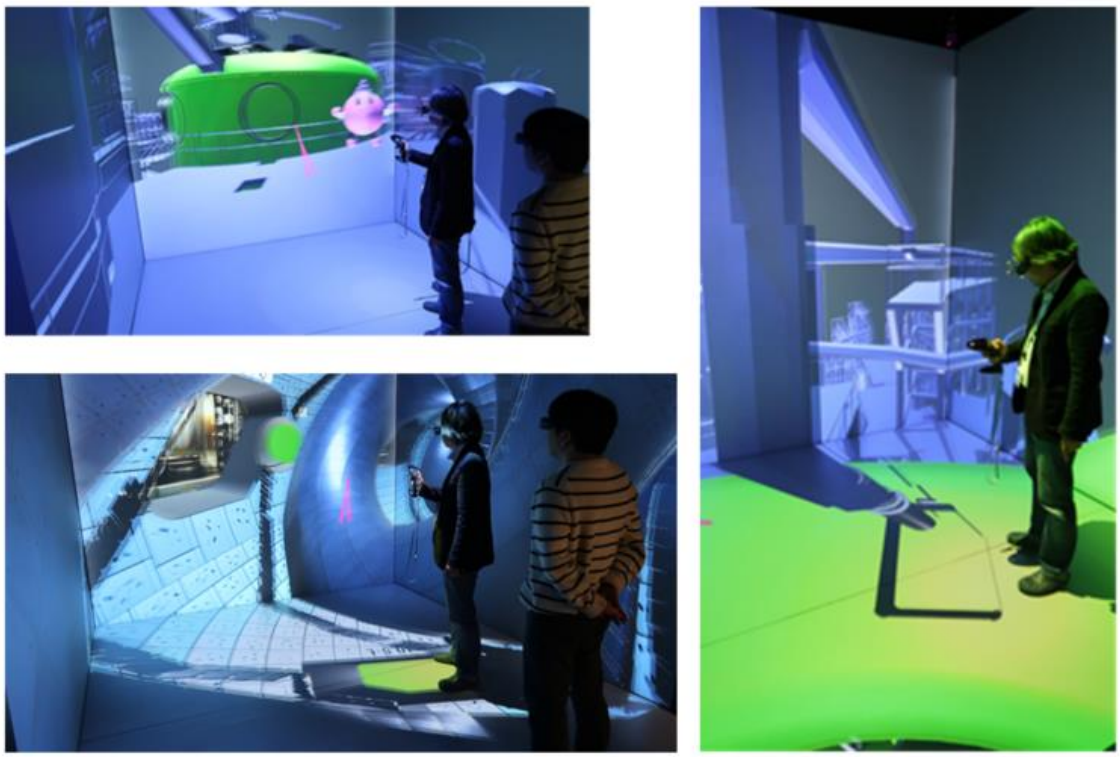


図 8.10-3　CAD で作成された LHD 周辺装置を Unity に取り込み、EasyVR の FusionVR オプションを使って、Virtual LHD とともに CompleXcope に表示している。

真空容器の中にプラズマのシミュレーション結果を表示することで、プラズマと実験装置の空間的な関係を調べたり、複雑な磁場の構造を直観的に調べたりすることができる。図 8.10-2 では、磁気面や磁気島、ストカスティック領域の様子が描かれている。図 8.10-1 は、磁力線の再結合に関するシミュレーションの結果で、時々刻々変化するプラズマの様子を、その時間変化を含めて VR 空間にムービーで再現した様子の一コマである[8.10-15]。この研究により、磁力線再結合とイオンの複雑な軌道との関係を直接調べることができた。

LHD の実験では、装置内部の容器壁などからはがれたダスト粒子がプラズマ中に侵入して、プラズマの長時間維持を妨げる場合がある。ダスト粒子がプラズマに及ぼす効果と影響を詳しく調べるためには、ダスト粒子が磁場中をどのように移動しているかを理解しなければならない。ダスト粒子はプラズマ中で光るため、LHD 実験では、この光を 2 台の高速カメラで撮影してダスト粒子の軌道を観測している。一方、磁場の構造は計算によって求めている。これらの二つのデータを重ねて表示すれば、磁場の中でのダスト粒子の振る舞いが分かる。LHD 実験におけるダスト粒子の観測データと計算によって求めた磁力線のデータを融合して 3 次元空間に表示するためのシステムを構築した[8.10-8]。開発したシステムを使うことで（図 8.10-4)、観測者自身がプラズマの中に入ってダスト粒子の軌道や磁力線をあらゆる方向から眺めながら、その立体構造を目の当たりにできる。訓練を積んだ専門の研究者

だけなく初学者にも容易にその立体構造が理解できるようになった。

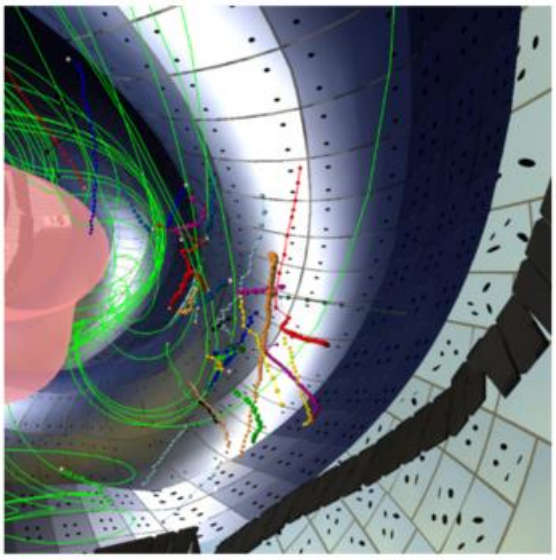


図 8.10-4　実験データとシミュレーションデータを同時に表示する 3 次元バーチャルリアリティを用いて、ダスト粒子とプラズマ中の磁力線との関係を VR 空間に表示した例。実際の設計データに基づいて再現した真空容器内部に、ダスト粒子の軌道データを、計算結果であるプラズマの磁力線（緑線）と等圧面（マゼンダ）と同時に表示している。複数のダスト粒子の軌道は異なる色で表現している。

研究所では、将来の核融合発電炉を実現するために、その原型となる「ヘリカル型原型炉」の設計研究を進めている。原型炉では多くの機器が取り付けられ、とても複雑な構造になる見込みである。そのため、原型炉設計では建設時の組み立て工程や稼働開始後のメンテナンス手順などを考慮する必要がある。この時、部品の取り付けや取り外し、移動に利用するロボットアームの設計、また、それを動かす手順も検討する必要がある。これら原型炉そのものとロボットアーム、そのまわりのメンテナンス作業を行う場所をまとめた総合的な設計研究が進められている。このような検討では、これまで設計用のソフトウェアを使って、通常のパソコンのディスプレイのような 2 次元ディスプレイに表示された情報を基に行ってきた。しかし、この方法では、本来 3 次元の情報を 2 次元に投影するために奥行き情報が失われてしまい、部品の立体構造や 3 次元的な位置関係の把握は難しくなる。そのため、部品やロボットアームの動きを検討しながら検討結果を設計に反映することは大変難しく、この問題を解決できる新たなシステムの開発が求められていた。そこで、研究所では、CompleXcope を使ってロボットアームを含めたヘリカル型原型炉の設計データを 3 次元 VR 空間に投影して、炉内部品の位置関係やロボットアームの動きについて、3 次元で確認できるシステムを新たに構築した[8.10-16]。このシステムでは、まず、原型炉の設計データを VR 空間に投影し、自分自身が原型炉の中に立ったり、歩いて視点を変えたりするなどして、

部品の位置関係をあらゆる方向から確認できるようにした（図 8.10-5）。次に、ロボットアームを含めたデータを投影し、ロボットアームによる部品の取り付け・取り外しや移動を確認できるようにし、さらに、自分自身の「手」を VR 空間の中に投影することで、VR 空間内の「手」で部品をつかんで動かしたりすることもできるようにした（図 8.10-5）。

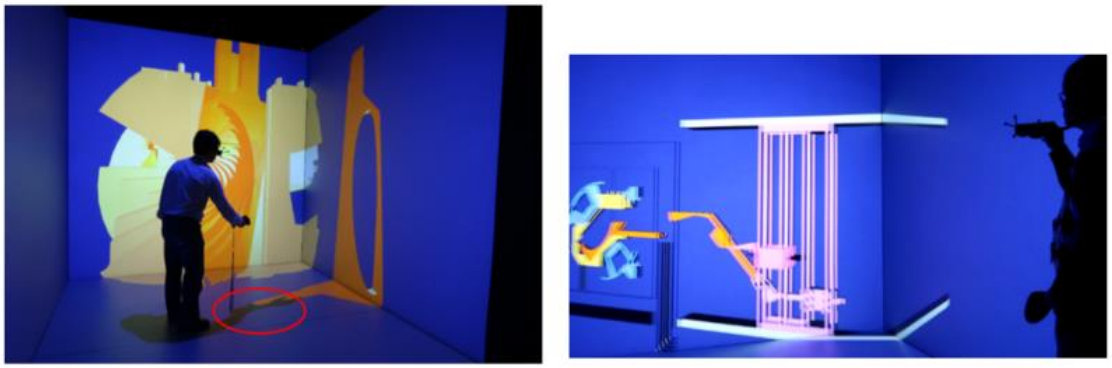


図 8.10-5　原型炉の設計データを、CompleXcope を用いて、VR 空間に投影している様子（左図）と原型炉設計データとロボットアームを VR 空間に投影している様子（右図）。VR 空間中に投影された自分自身の「手」（床面スクリーン上の赤で囲まれたところに手が映っている。操作している人には自分の「手」は浮かんで見えている）で部品をつかんで動かしている。下図ではブランケット（オレンジ色）をロボットアーム（ピンク色）がつかみ、右方向へ運んでいる。

VR 技術はあらゆる分野で活躍する可能性を秘めている。医療教育として MRI（磁気共鳴画像）の画像を可視化して手術の手順等の決定を支援するシステムや、仮想空間の中で歩行訓練をするなどのリハビリテーションへの応用、スポーツの動作を可視化して動作分析を支援することで技術向上を図るシステム、自然災害などでの緊急時の様子を疑似体験するシステムなど、様々な分野で研究が進められている。ゲーム業界を含め、様々な分野の研究者と協力して、VR 装置を用いた可視化研究を更に推進し、それを通して、核融合発電の実現に貢献していきたいと考える。

参考文献

[8.10-1] 大谷寛明：NIFS ニュース No.211 及び No.242、2016 年 4 月 5 日プレスリリースより

[8.10-2] 陰山　聡, 佐藤哲也, NIFS-MEMO, 28, (1998).

[8.10-3] A. Kageyama, Y. Tamura, T. Sato, Trans. Virtual Reality Soc. Japan, 4, 717 (1999).

[8.10-4] http://www.jamstec.go.jp/esc/research/Perception/vfive.ja.html

[8.10-5] A. Kageyama, T. Hayashi, R. Horiuchi, K. Watanabe and T. Sato, Proceedings of 16th International Conference on the Numerical Simulation of Plasmas, 138, (1998).

[8.10-6] H.Ohtani, et al: Plasma Fusion Research, 6 (2011) 2406027 (4pages).

[8.10-7] H.Ohtani, et al: Proceeding of International conference on Simulation Technology (JSST 2012)(2012) 394-398 (4pages).

[8.10-8] H.Ohtani, M.Shoji, N.Ohno, Y.Suzuki, S.Ishiguro, A.Kageyama and Y.Tamura: Contrib. Plasma Phys. Vol.56, No.6-8, (2016) 692-697.

[8.10-9] https://www.cybernet.co.jp/avs/products/avsexpress/

[8.10-10] https://www.cybernet.co.jp/avs/products/mpe/

[8.10-11] https://www.fiatlux.co.jp/product/virtual/easyvr/index.html

[8.10-12] https://www.middlevr.com/home/

[8.10-13] S.Kawahara and A.Kageyama: Journal of Advanced Simulation in Science and Engineering, Vol.6 (2019), pp.234-248.

[8.10-14] http://www.aec.co.jp/solution/mm/products/virdse/index.html

[8.10-15] N.Ohno, H.Ohtani, et al.: Plasma Fusion Research, 7 (2012) 1401001 (6pages).

[8.10-16] Hiroaki Ohtani, Seiji Ishiguro: Proceeding of the 36th JSST Annual International Conference on Simulation Technology, (2017) 194.

8.11　製品設置空間の事前検証のための MR 利用

富士通デザイン株式会社　山岡鉄也

(1) 初めに

大型サーバや製造機械などの大型製品は、その設置に於いて課題となることとして設置空間の確保の問題がある。この設置空間とは、実際に製品を設置した際に占める空間とその製品を取り扱う際に必要な動作が可能な空間を意味するが、実空間上でそれを検討する場合、平面的な占有面積は検討しやすいが、3 次元的な空間の占有面積となると計測が難しくなる。さらにそれを使った動作に必要な空間となると計測は非常に難易度が上がると考えられる。富士通デザインでは XR の技術を使ってこの課題の解決方法を見出すべく、実空間とその製品の原寸大の 3D モデルを同時に表示し、さらにその製品データに操作を加えることができる方法を検討した。

(2) 手法の検討

現実世界と仮想世界を同時に体験する方法として AR／MR と手法が最適と考えらえるが、そのもっとも手軽なスマホで実現する AR の場合は、片手がふさがるので作業検討には向かないと推測できる。一方、両手が空く HMD であれば両手を自由に動かせるので作業検討にも対応ができると推測ができた。現在、AR/MR デバイスの主流である Microsoft HoloLens での実現の検討を開始してみた。しかし、二つの課題があるため断念した。一つは視野角の問題である。HoloLens の視野角は 30° ×17.5° （表 6.1-1）程度しかないため、手が届く程度に対象に近づくと操作対象となる製品の全容が確認できないことが分かった。二つ目は HoloLens が認識する手の動きは動作としてとらえておらず、マウスタップのようなジェスチャとしてのみ有効であるため両手を使った作業を再現するのには向かないと判断した。そこで、さらに調査を進めて行くと HoloLens がパススルー型 MR デバイスであるのに対し、ビデオスルー型 MR と呼ばれるデバイスがあることがわかった。ZED Mini（図 8.11-1）は HMD に装着することで VR 用の HMD をビデオスルー型 MR に改造できるデバイスである。このデバイスは左右二つのカメラで撮影した外界の景色を HMD の片目ずつそれぞれに見せることで、液晶を通した映像ながら立体視として視認することを可能にしている。さらに HoloLens と比較して視野角は広く、手が届くまで近づいても十分に操作対象を認識できる範囲を見ることが出来る性能を有している。富士通デザインではこの ZED Mini と HTC VIVE を組み合わせてビデオスルー型 MR デバイスを構築した。

図 8.11-1　ZED Mini を装着した HTC VIVE

(3) 設置空間の事前検証を可能にするソリューションとしてのビデオスルー型 MR

設置検討をする製品の候補として富士通製品内で、簡単に移設して確認できないものとして PRIMERGY サーバ（以下サーバ）を検討の対象として選定した。富士通デザイン内のハードウェアデザインチームからデザイン用 3DCAD データを借用することで、製品の 3D モデルの寸法を取得した。デザイン用の 3D モデルデータは内部の詳細な構造は含まれていないので、通常の製品 CAD データと比較して非常に軽量であり、インタラクティブなコンテンツ用として非常に向いているといえるが、さらに外部から見えない箇所や MR に不要な箇所のデータは手作業で外し、HMD 内で描画をする際の負荷軽減を行った。しかし、この 3D モデルは CAD データのため、外観はリアルに描画されていない。そのため、マテリアルやテクスチャ設定などを Blender と Photoshop で行い、描画される製品がよりリアルに視認できるよう CG 的な外観の加工修正を加えた。さらに本検討は作業の再現を目指しているため、メンテナンス用のドアを開けられるようにデータのパーツ分解を施しさらに挿さっている HDD を取り出すといった作業までを再現可能にするために、Unity によるインタラクション開発を行った（図 8.11-2)。

図 8.11-2　AR／MR による製品の設置空間確認の検証のため
原寸のサーバを出現させるアプリを Unity で制作

ZED Mini と VIVE のビデオスルー型 MR による製品設置と作業範囲検証を制作したアプリで体験すると、実空間内に重畳された CG の安定感がよいことが分かった。VIVE はアウトサイドイン方式で実空間に設置したベースステーションで空間認識しているため、自分が装着しているカメラで位置補足するインサイドアウト方式より、CG の位置がブレにくい。合わせて ZED Mini の特徴でもある CG より距離的に手前にあるものは実写を CG の手前に描画する機能により、自分の手が MR 内に見えており、対象の製品との距離感をより体感度が高く体験できることが分かった。VIVE の機能上、コントローラを握ったかたちの手が見えてしまうのだが（図 8.11-3)、それでもサーバラックの扉を開けるときの動作を再現できるので、扉が開く際の回転動作範囲やそれを引く腕の動く範囲などを確認するに十分といえる。（図 8.11-4）

図 8.11-3　VIVE のコントローラを握った状態の手が CG の手前に表示された様子

図 8.11-4　CG で表現されたサーバの操作を再現している様子

この検討の応用として、様々な活用が考えられるが一例として、リフォーム時のキッチンなどのビルドイン型の家具の設置検討が考えられる。最近のキッチンはシステム化されており、様々なオプションの組み合わせが可能になっているが、そのためショールームなどではすべてを展示しきれない現状にある。さらに、それを設置先の住宅で検討しようとすることは非常に難しく、現状では図面や絵から想像して決定するような手段が取られている。これでは購入者が納得して購入しているとは言い難い状況である。しかしこの課題は、当ソリューションで解決が期待できる。例えば、現状のインテリアについてカメラを通して見ながら、新たに設置するキッチンを同時に CG で表示することで、それぞれのマッチングの確認をしたり、高さや幅の再現も原寸で可能なので、キッチンを使う際の使い勝手の確認も可能になると想像することができる。(図 8.11-5)

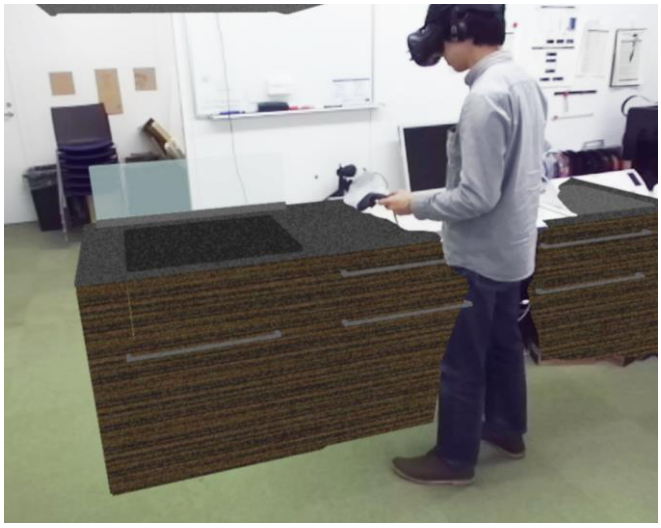


図 8.11-5　キッチンの設置検討用への活用が考えられる

(4) これから解決すべき課題

現状は、コントローラを使った腕の動きの再現とコントローラのボタン操作による押す、引くなどの指の動きの再現のみだが、作業検討という以上は、より細かい動作の検討も実現していく必要がある。そのため、コントローラに替えてグローブ型のデバイスの利用を検討する予定である（図 8.11-6)。最近発売されているグローブ型デバイスでは 10 本の指の動きをセンシングして細かな作業を再現するだけでなく、指先への触感のフィードバックを実現しているものも多い。それらを活用することで設置検証のみならず、その場で指先を使うような細かな作業性までを検証することが可能になると考える。さらにもしキッチンのようなインテリアの検討まで利用範囲を考慮すると、実空間の明るさを CG 内に反映することも必要になるだろう。たとえば、ダイニングの昼と夜でのキッチンの見え方の確認などさ

らに用途が広がること期待できる。この様に、当検討が様々な製品検討への活用へ資することを願う。

図 8.11-6　VR/MR 用グローブデバイス（MANUS VR）

9. まとめ

本WGでは､データ可視化を行っている研究者､VR装置を使って実験を行っている研究者、VR装置などで稼働する可視化表現法やそのソフトウェアを研究開発している研究者、VR技術の開発を行っている研究者など、技術開発から応用までの幅広い分野の研究者が集まり、ゲーム開発エンジン Unity を科学技術分野や教育分野で活用するためのノウハウや課題について議論を進めるとともに、広く、VR 可視化技術・可視化情報技術の開発、それによる研究の現状評価と研究の方向性について議論を行った。

Unityをはじめとするゲーム開発エンジンは、高速・高品質のレンダリング能力やゲーム開発のための優れたインターフェイスを持ち、また、豊富なアセットが用意されており、さらに､多くの先進技術による入出力デバイスのインターフェイスを備えている。このため、可視化アプリケーションを容易に開発することができる。他方､数値計算結果からの発見や結果の整理には対話的可視化処理が必要で、そのような可視化処理に Unity を連携させようとすると、Unityへの可視化機能の搭載や通信が必要であること、Unityと可視化ソフトウェアでは色や時系列データの持ち方が異なるなど､課題が浮き彫りとなった。その課題の解決方法として、データフロー型のフレームワークをUnityに構築する方法やParaViewでの可視化結果をUnityにインポートするためのデータ変換方法が提案された。

本報告書では､可視化の歴史､データ可視化の開発技術の現状､ゲーム開発エンジンUnityを用いた標準的なワークフローやデータのインポート・マッピング法､現状のデバイスの紹介などを載せ､さらに新しい可視化手法や活用方法､コンテンツ開発の際の具体的な課題及びその解決方法例、WG 参加研究者による可視化事例をまとめた。これから可視化を行う研究者にとって何かしらのヒントが本報告書の中で見つけられると自負している。

本報告書での議論がさらに発展し、国や分野を超えた汎用VRシステム利活用に関する検討の一助となることを期待する。最後に、2年間の WG 活動で講師としてご発表いただいた先生方、ご協力いただいた皆様に感謝いたします。

（汎用VRシステムの活用研究WG　まとめ役　大谷寛明、畠中靖浩）

参考資料　可視化技術の歴史と VR 技術
　　宇宙航空研究開発機構　藤野敦志氏

可視化技術の歴史とVR技術

JAXA 藤野敦志