建築物向けVRアプリケーションのためのソフトウェアアーキテクチャに関する考察

M2018SE008 北川智久
指導教員：沢田篤史

1 はじめに

VR用ヘッドマウントディスプレイ（以下，HMDを呼ぶ）とPCの低価格化や，スタンドアロン式HMDの発売によりVR（仮想現実）はより身近なものとなっている．また，VR技術の発展に伴い，VRを利用したサービスが数多く登場している．例えば，アミューズメント施設[1]，軍事訓練，自宅でのVRスポーツ，商品のプレゼンテーションなどへの利用例が挙げられる．本研究ではVRを利用したサービスのうち，商業的な利用方法の中で最も利用されており，かつ前述のとおり非常に効果的である商品プレゼンテーションサービスの一つの建築物向けVRアプリケーションに着目する[2]．

建築物向けVRアプリケーションは，主に物件の内装や外装を利用者がVR空間上で確認する目的で利用される．このサービスのメリットは，利用者が直接物件のある場所に物理的に移動しなくてもVR空間上で物件を確認できることにより大幅に移動や時間のコストを削減できること，またモデルルームの運営コストなどを削減でき，利益率の向上を期待できることである．

建築物向けVRアプリケーションのアプリケーションアーキテクチャについては充分に整備が進んでいない．VRアプリケーション向けのアーキテクチャは既に提案されている[3] [4]が，建築物向けVRアプリケーションは通常のVRアプリケーションとは異なる特有の要求がある．特有の機能や特性を以下に示す．

- 激しい方向転換が行われることを想定すべき．
- 移動中に前方以外を見ることは少ない．
- 部屋の内覧とアバター以外のオブジェクトは必要ではない．

これらの要求に対応するためのコストが必要で，開発効率に悪影響が出る．したがってこのVRアプリケーションのアーキテクチャでは建築物向けVRアプリケーションの開発に対応するには不十分であり，この特有の要求に対応し開発効率の問題を解決する必要がある．

本研究では，開発効率が悪化している問題を解決するために建築物向けVRアプリケーション特有の要求に対応したソフトウェアアーキテクチャを提案することで，建築物向けVRアプリケーションの開発支援を行う．

まず，既存のVRアプリケーションアーキテクチャについて考察し，建築物向けVRアプリケーションの開発に適するようにアーキテクチャを再構築する．次に建築物向けVRアプリケーション特有の機能や特性について調査する．そしてこれらがVRアプリケーションアーキテクチャのどのモジュールに関わるのかを検討し，再構築したアーキテクチャに追加することで再設計する．

既存のVRアプリケーションアーキテクチャと本アーキテクチャを比較した場合，後者は描画処理の負荷が低いので実行時のフレームレートが高く，よりVR酔いを起こしにくいアプリケーションを作成することができる．また，建築物向けVRアプリケーションの開発に関わるモジュールのみ存在しているのでアーキテクチャがよりシンプルとなり，建築物向けVRアプリケーションの開発に不慣れな場合でも利用しやすいアーキテクチャとなったと考える．さらに，MVVMモデルに基づいたアーキテクチャ設計により，ソフトウェアの煩雑さを解消し，HMDの座標や向きのデータを迅速に検知し，各モジュールを高速に更新することが可能となる．

2 建築物向けVRアプリケーション開発における課題

2.1 VRアプリケーションの概要

VR(Virtual Reality)は，VR体験用のゴーグルやHMDを装着することで現実に近い世界に没入する感覚が得られる体験をすることのできる技術である．実際に肉体が移動することで仮想空間内でも移動することができ，また専用のコントローラを用いた移動や操作も可能である．

2.2 建築物向けVRアプリケーションの概要と課題

本研究で対象とする建築物向けVRアプリケーションは，一般的に不動産業界で用いられているVRアプリケーションである[4]．具体的には，利用者がHMDを装着し，コントローラを用いたり実際に移動して仮想空間内の物件を歩きまわるものを想定している．建築物向けVRアプリケーションの課題を考察する際に，まず建築物向けVRアプリケーション特有の機能や特性について調査した[4]．特有の機能や特性を以下に示す．

- 激しい方向転換が行われることを想定すべき．（利用者には物件の様々な場所を確認したいという要求があるので，視点は360度様々な方向を向く可能性があるから）
- 移動中に前方以外を見ることは少ない．（移動中に背後を確認するシチュエーションが無いから）
- 部屋の内覧とアバター以外のオブジェクトは必要ではない．

通常のVRアプリケーションと異なり建築物向けVRアプリケーションは特有の機能や要求が多く，また特殊であることが問題である．建築物向けVRアプリケーションでは激しい方向転換を行った際にフレームレートが大幅に下がり，VR酔いを引き起こす可能性がある[5]．VR酔いとはVR体験の際に気分が悪くなる症状のことで，VR酔いが起こる原因としてはフレームレートの不足，自身の視覚情報と平衡感覚のずれなどがあり，VRコンテン

ツ開発者はアプリケーションを開発する際に VR 酔いの原因になりうるものは極力排除し，VR 酔いを起こしにくくする必要がある．

建築物向け VR アプリケーションの場合，激しい方向転換をした際でも VR 酔いを起こさないようにフレームレートを高く維持するなど，特有の機能や要求に対応する必要がある．

2.3 関連研究

2.3.1 VR アプリケーションアーキテクチャ

目的の達成のため，基盤となる VR アプリケーションアーキテクチャについて考察した．既に提案されている VR アプリケーションのアーキテクチャ[2] を図 1 に示す．

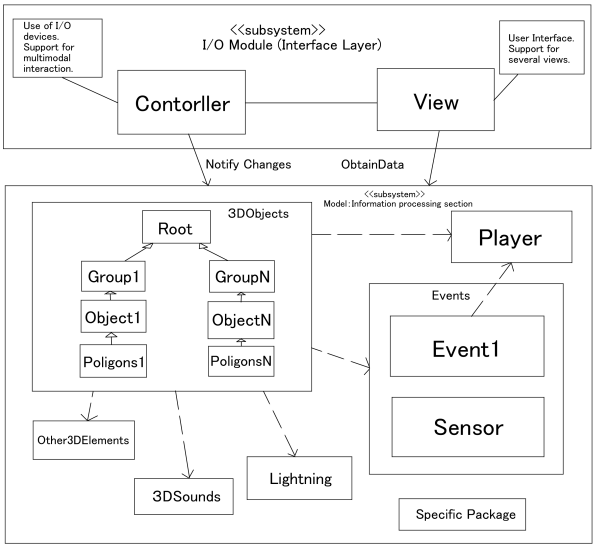

図 1 既存の VR アプリケーションアーキテクチャ

このアーキテクチャは MVC(Model-View-Controller) モデルに基づいて構築されている．

Model では 3D オブジェクト，アバター，ライト，その他のオブジェクトによって形成されている VR 空間内の仮想シーンに関わるデータを扱う．プレイヤーのアバター制御は *Player* で，家具や床などの 3D オブジェクトの管理は *3DObject* で，光源の種類やオンオフの管理は *Lightning* で行い，*Events* ではオブジェクトの移動や回転，センサーの数値変更などをイベントとして受け取り，各モジュールにイベントを送り必要な処理を行う．

View では HMD に描画される画像を随時更新する役割を担っている.Controller のデータ変更を監視し，画像を更新する．

Controller ではユーザからの入力や Model からのイベントを受け取り，Model や View にイベントを送る役割を担っている．また，DeviceType 属性によりさまざまなデバイスに対応できる．

2.3.2 問題点

建築物向け VR アプリケーションは，不動産物件の内覧，外覧を VR 空間上で再現するために家具などのオブジェクト，光源，テクスチャ，自身のアバターなどのデータを扱う．これらのデータを全て統合したものをレンダリングした空間を利用者が VR 空間上で歩きまわる．基盤となる VR アプリケーションアーキテクチャは，これらのデータ全てを MVC モデルの Model で扱い，Model が肥大化している．これはソフトウェアを複雑にしソースコードが煩雑になる原因であり，解決しなければならない問題である．

また，建築物向け VR アプリケーションなど，特定の用途で利用する際に十分なモジュールが定義されていないことも問題として挙げられる．さらに，入出力に関するモジュールが少なく，このアーキテクチャを利用して開発を行うには不十分であることも問題である．

3 建築物向け VR アプリケーションアーキテクチャ

3.1 想定する建築物向け VR アプリケーションの概要

本研究で想定した建築物向け VR アプリケーションは，一般的に不動産業界で利用されているものである．ユースケースを図 2 に示す．

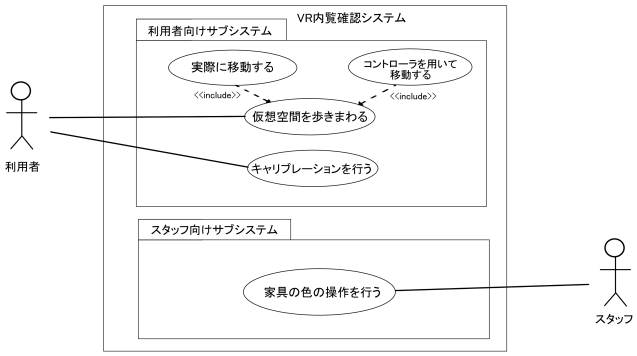

図 2 建築物向け VR アプリケーションのユースケース

ユーザは HMD を装着した後，仮想空間のアバターと現実の体の体格や目線を合わせるためにキャリブレーションを行い，コントローラや現実での移動などで仮想空間内の物件内を移動する．スタッフはユーザの見ている視界を別の PC で確認し，ユーザからの要望などがあれば家具の色変更などの操作を行う．

3.2 アーキテクチャの再構築

汎用の VR アプリケーションアーキテクチャの問題を解決するために，MVC モデルに基づいて設計されたアーキテクチャを MVVM モデルに基づいて再構築する．MVVM モデルとは，アプリケーションソフトウェアを Model-View-ViewModel の 3 つに分割し，設計・実装する方法である．MVVM モデルに基づいてアーキテクチャを再構築する理由は以下の二つである．

- MVVM モデルは，View と ViewModel が双方向データバインディングで結びついており，データの変更を検知した際に迅速に View を変更することが可能である.VR アプリケーションにおいて視点の変更は

頻繁に起こり，その変更に応じて HMD に映る映像を視点に合わせたものに変更する必要があるので，MVVM の双方向データバインディングは VR アプリケーションにとって効果的であるから．

- 基盤となるアーキテクチャでは MVC モデルの Model に多くのデータを扱わせていたが，それらのデータの一部を MVVM モデルの ViewModel に扱わせることで Model の肥大化を防ぐことができるから．

3.3 特有の機能や特性をアーキテクチャに追加

建築物向け VR アプリケーション特有の機能や特性をモジュールとしてアーキテクチャに追加することで，アーキテクチャを再設計する．

VR アプリケーションでは，高いフレームレートを維持し VR 酔いを起こす可能性を下げる必要がある．高いフレームレートを維持するにあたって必要であることは，描画する必要がないものを極力描画せず，描画負荷を軽減することである．建築物向け VR アプリケーションにおいてユーザは移動中は前方以外を見ることが少ないという特性から，移動中の視点とは逆の方向にあるもののオブジェクトを描画せず，描画負荷を軽減することで高いフレームレートの維持に貢献することができる．

また，部屋の内覧とアバター以外のオブジェクトは存在する必要が無いので，実行時にそれらのオブジェクトを非表示にすることで描画負荷を軽減することができる．

これら二つを高いフレームレートを維持するためのモジュールとして実装する．

3.4 建築物向け VR アプリケーションアーキテクチャ

既存の汎用 VR アプリケーションを基盤に，建築物向け VR アプリケーションアーキテクチャを再設計した．以下にアーキテクチャの静的構造と動的構造を示す．

3.4.1 静的構造

設計方針に基づき，まずアーキテクチャを MVVM モデルに基づいて再構築する．次に建築物向け VR アプリケーション特有の機能や要求をモジュールとして追加することでアーキテクチャを再設計する．再設計した建築物向け VR アプリケーションの静的構造を図 3 に示す．

既存のアーキテクチャは Model と比べて Controller と View のモジュールが少なく，抽象度にばらつきがあるという問題があった．これを解決するために MVVM で再構築する際には View，ViewModel をより詳細にし，抽象度を下げた．

Model にはイベントの送受信処理を行うモジュール（*Event1*，*Sensor*）が定義されていたが，MVVM の場合イベントの送受信処理を行う機能を有しているのは View か ViewModel であるので，そこに定義すべきである．したがって，イベントの送受信処理機能を持つモジュール（*Device*，*RoomViewConstructor*）を View と ViewModel に定義した．

本研究では，ViewModel に UI の描画に必要な情報を保持する機能を，View にイベント送受信機能と入出力情報の管理を，Model にビジネスロジックとオブジェクト

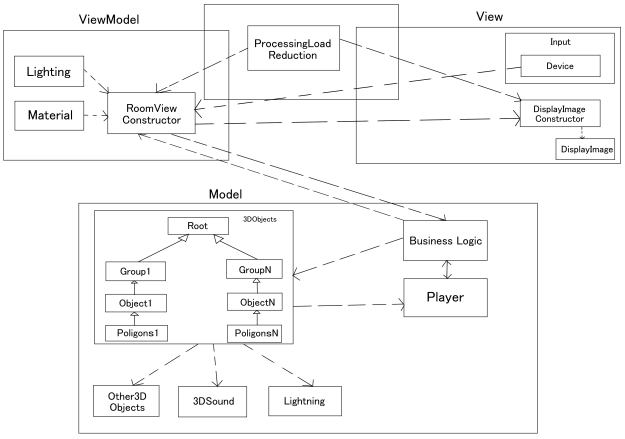

図 3 建築物向け VR アプリケーションアーキテクチャの静的構造

データの管理のみを集中させることで様々な機能のコードが各モジュールに分散することを防ぎ，コードの煩雑さを軽減し保守性を高めている．

建築物向け VR アプリケーションの処理負荷軽減機能を横断的関心事として扱い，処理負荷軽減機能をもつモジュール（*ProcessingLoadReduction*）を定義した．単一モジュールとして定義することでモジュール間の独立性を高める．横断的関心事とは，オブジェクト指向や機能指向など，システムに対する支配的な関心事に基づいて分割された複数のモジュール間に横断的に関心があり，支配的な関心事によるモジュール分割では 1 つのモジュールにまとめにくい関心事のことである．処理負荷軽減機能にはユーザが移動している際に背後のオブジェクトを非表示にする機能と，内覧確認の際に不要なをオブジェクトを非表示にする機能を採用した．

3.4.2 動的構造

再設計した建築物向け VR アプリケーションの動的構造を図 4 に示す．

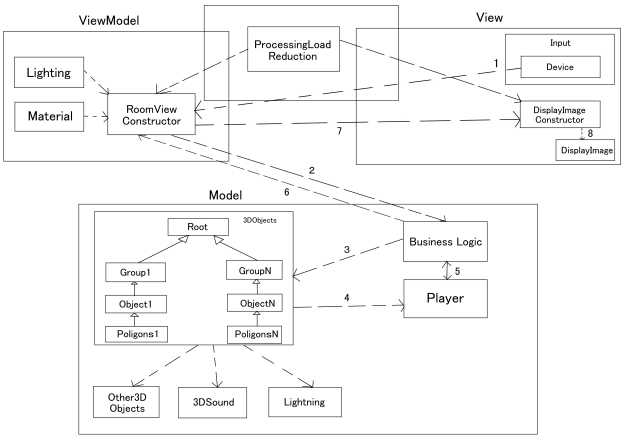

図 4 建築物向け VR アプリケーションアーキテクチャの動的構造

HMD の向きや座標，コントローラからの入力を *Device* が受け取り，イベントとして *RoomViewConstructor* にメッセージを送る．*RoomViewConstructor* は仮想空間内の物件の内覧やアバターをレンダリングする前の情報を持っており，HMD やコントローラからの入力データによってアバターの位置や角度が変化した場合 HMD に出力する画像を角度や位置に応じて変更すべきである．

BusinessLogic では受け取ったメッセージに基づいて仮想空間内のオブジェクトの位置を変更やライティングの変更，音声の変更などを *3DObjects*，*Other3DObjects*，*3DSounds*，*Lightning* に依頼する．*3DObjects*，*Other3DObjects* では各オブジェクトの位置や回転，拡縮のパラメタを変更する．*3DSounds* では音声のオンオフ，音源の変更を行う．*Lightning* では光源のオンオフ，種類の変更を行う．

BusinessLogic は変更後のオブジェクトデータなどを *RoomViewConstructor* に送る．*RoomViewConstructor* は変更後のオブジェクトデータの材質の情報の適用を *Material* に依頼し，*Lighting* では光源を仮想空間内に適用する．*RoomViewConstructor* はレンダリングするための仮想空間の情報を持ち，*RoomViewConstructor* は *DisplayImageConstructor* に仮想空間のレンダリングを依頼する．*DisplayImage* ではレンダリング結果の画像を HMD の片目ずつそれぞれのディスプレイに出力することでユーザは仮想空間の立体視が可能になる．

4　考察

既存の VR アプリケーションアーキテクチャは Model，View，Controller 間でモジュールの数の差が大きく，特に Model が肥大化していた．これはソフトウェアを複雑にしソースコードが煩雑になる原因である．本アーキテクチャでは Model，View，ViewModel 間のモジュール数の差を極力減らし，肥大化を防ぎソフトウェアの煩雑さを解消した．

また，View と ViewModel を双方向データバインディングで結びつかせることで，HMD の座標や向きの数値の変更を迅速に検知し，各モジュールを高速に更新することが可能となる．

DisplayImageConstructor，*Player* や *RoomViewConstructor* など，建築物向け VR アプリケーションの開発のためのモジュール存在しているのでアーキテクチャがよりシンプルとなり，建築物向け VR アプリケーションの開発に不慣れな場合でも利用しやすいアーキテクチャとなったと考える．

本アーキテクチャを利用してアプリケーションを開発する場合，既存のアーキテクチャを利用する場合に比べ描画処理の負荷を軽減するモジュールが存在するので，実行時のフレームレートが高く，より VR 酔いを起こしにくいアプリケーションを開発することができる．

開発に用いるゲームエンジンである Unity では，VR アプリケーションでのユーザの視野を表すオブジェクトである Camera オブジェクトが存在し，移動中にこの Camera オブジェクトの視野外にあるオブジェクトを非表示にするスクリプトを Camera オブジェクトにアタッチすることでユーザの視点とは逆方向のオブジェクトを非表示にする機能を実現することができる．また，物件の内覧を見る場面になった際に指定したオブジェクトを非表示にするスクリプトを記述することでユーザの視点とは逆方向のオブジェクトを非表示にする機能を実現できる．

また,Unity でのアプリケーション開発において,MVP モデルに基づいて開発する場合も多いが,MVP モデルで設計する場合 Model が肥大化する傾向にある。したがって,MVP モデルに基づいてアーキテクチャを設計する必要性は低く, MVVM モデルに基づいた設計は適切である．

5　おわりに

建築物向け VR アプリケーションのアプリケーションアーキテクチャは整備が進んでおらず，建築物向け VR アプリケーションには激しい方向転換に対応するべきであることや不必要なオブジェクトが存在することなど特有の機能や要求が多いので，既存のアーキテクチャで対応するには不十分であり，開発効率に悪影響が出る．

そこで，本研究では特有の機能や要求に対応したソフトウェアアーキテクチャを提案することで，建築物向け VR アプリケーションの開発支援を行った．本アーキテクチャを利用することで，建築物向け VR アプリケーション特有の機能や要求に対応することができ，実行時のフレームレートを高く維持し，より VR 酔いを起こしにくいアプリケーションを開発することができると考える．また，MVVM モデルに基づいた再構築によりソフトウェアの煩雑さを解消した．

今後の課題は，本アーキテクチャを利用しアプリケーションを開発した際に開発効率が改善されたことのデータを取得することである．また，より多くの開発環境に対応した汎用性の高いアーキテクチャとするために，多くの企業で開発に用いられているゲームエンジンである Unity 以外のエンジンでの開発にも対応できるようにアーキテクチャを再構築すべきである．

参考文献

[1] 相川 達也,“ エンターテイメントにおける VR 技術を用いた没入感の創造 ”，早稲田大学理工学術院基幹理工学研究科 修士論文，2015.

[2] Rafael Capilla ，Margarita Martinez,“ Software Architectures for Designing Virtual Reality Applications ”，EWSA，p .135-147 2004.

[3] Lanshan Zhang, Linhui Sun, Wendong Wang, and Jiangchuan Liu,“ Unlocking the Door to Mobile Social VR: Architecture, Experiments and Challenges ”，IEEE Network，p .160-165，2018.

[4] Jennifer Whyte, ,“ Industrial applications of virtual reality in architecture and construction ”，ITcon Vol.8，p .43-50，2003.

[5] 藤木 卓，市川 幸子，寺嶋 浩介，小清水 貴子,“ VR コンテンツの精度が現実感と酔いに与える影響 ”，日本教育工学会論文誌，p .73-76，2012.