データベースを核とした ゲームエンジンと三次元 CAD によるデザイン手法の試行

○戸田　勇登[*1]　高橋　洋祐[*2]
加戸　啓太[*3]　平沢　岳人[*4]

キーワード：Virtual Reality　データベース　ゲームエンジン　三次元 CAD

1. はじめに

設計から施工に至る各段階において三次元モデルが用いられることが一般的になりつつある。BIM（Building Information Modeling）のコンセプトのように、仮想空間上に建築の三次元モデルを構築することで、例えば空間の構成をより適切に把握することができる。従来の模型による表現と比較すれば、設計変更などへの対応が容易であるという優位性を挙げることもできる。

三次元モデルのビジュアライゼーション手段として、多くの建築向け三次元 CAD（以下、建築 CAD）には、レイトレーシングなどによる太陽光や照明のシミュレーションを行い、写実的なパース画像を生成するレンダリング機能が用意されている（図 1）。レンダリングには相応の時間を要するものの、コンピュータの負荷や操作に対する応答性を重視し簡易的な表現に留まる建築 CAD の操作画面（図 2）では確認しにくい、採光や材質による空間の変化を確認することもでき、プレゼンテーションのみならず設計段階においてもこの機能が使われることもある。また、近年では、GRAPHISOFT 社の BIMx や REDSTACK 社の FUZOR など、バーチャルリアリティ（Virtual Reality、以下 VR）体験を提供するソフトウェアも用いられるようになった。これらの写実性はレンダリングによる画像と操作画面の中間ほどであるといえるが、建築 CAD との連携を前提としておりスムーズに VR シーンを構築可能なため、これによるプレゼンテーションを目にする機会も増えた。

一方、コンピュータゲームなどの分野では、ゲームエンジンと呼ばれるソフトウェアを用い VR シーンの構築が行われている。ユーザーの操作に応じたインタラクションが前提であるゲームにおいては、写実的なシーンをリアルタイムで生成することが必須である。1/60 秒程度でレンダリング画像を逐次出力することが求められるが、コンピュータの性能向上を背景に、光伝搬現象のシミュレート物理ベースで行うレンダリング方式なども導入されており、写実性に富んだ VR シーンを構築できる。物理ベースレンダリングなどの新しい技術の意欲的な導入、ヘッドマウントディスプレイなどのハードウェアのサポートを始め、活発な開発コミュニティーもゲームエンジンの特長として挙げられる。ヘッドマウントディスプレイを用いた没入型の VR 体験は注目度も高く、建築設計段階での検討における活用[1)]や、施工管理者向けの教育システムとしての活用[2)]も報告されている。

さて、建築 CAD で作成したデータを、ゲームエンジンによって VR シーンとするにはいくつかの手順が必要である。建築 CAD から OBJ 形式でエクスポートし、それをゲームエンジンにインポートというのが一般的な手順であるが、そのままでは材質の質感や光環境の再現に問題があるため、マテリアルや光源を設定しなおすといった手間を要する。建築 CAD 上での更新があった場合は再度同様の手順を行うことになる。もちろんこれらはゲームエンジンに関する技術・知識がなければ行うことはできない。

そこで本研究では建築 CAD とゲームエンジンをデータベースによって接続し、ゲームエンジンによる表現豊かな VR シーンをより活用するための手法を提案する。スムーズな連携を実現することで、VR 体験と設計を並行したデザイン手法が実現できる。また、VR シーンを用いたプレゼンテーションにクライアントの要求を即座に反映させることなどもでき、様々な発展が期待できると考えられる。

図 1　レイトレースレンダリング

図 2　CAD 画面

2. データベースを介した CAD とゲームエンジンの連携

2.1. システム構成

本研究では図 3 のように建築 CAD とゲームエンジンをデータベースによって接続し連携を試みる。データベースのスキーマは三次元モデルの相互編集を想定して設計しているが、本報告では建築 CAD からゲームエンジンへのモデルデータの送信のみを報告する。建築 CAD には BIM の導入された建築 CAD である ArchiCAD[3)]、ゲームエンジンにはオープンソースで拡張性の高い UnrealEngine4 (UE4)[4)]、データベース管理システムには PostgreSQL[5)]を使用した。データベースに格納するモデルデータは、最も汎用性が高く、多くのゲームエンジンでも標準的に使用さ

れており、建築 CAD のエクスポート形式としてもサポートされていることが多い三角形ポリゴンメッシュとした。

図 3　システムの構成

2.2 データベースの設計

建築 CAD からゲームエンジンへモデルデータを渡すにあたっては、前述の一般的なファイルのインポート・エクスポートによる場合に課題となる：

・建築 CAD での三次元モデル更新の反映

・マテリアルや光環境などの読み替えの手間

に着目しつつ、データベースのスキーマを図 4 のように設計した。part テーブルは 1 部材を 1 レコードとして管理し、部材毎の更新情報をタイムスタンプとして記録することでモデルが更新された際は差分のみ読み込むよう工夫した。プライマリーキーとして ArchiCAD でのオブジェクト GUID を登録することで、一意に管理を可能とした。prop テーブルには部材情報のうち、オブジェクトカテゴリーによって付随の有無が変わるパラメータを持たせた。例としては柱の縦横寸法などを記録し、ゲームエンジン側からでもモデルをメッシュ化した際に失われた元の形状のパラメータにアクセスできるように配慮した。mesh テーブルは幾何形状を管理するためのテーブルであるが、part テーブルと別に設けこれを part テーブルから参照するような構成とした。例えば曲面を多用しメッシュの頂点数が多い椅子が、同形状でありながら異なる位置に複数置かれていても、無駄なく処理を行えるという利点がある。なお、ArchiCAD 上では、部材形状の同異は管理されていないため、メッシュの幾何形状から生成した hash 値を比較することで判別している。mesh_section テーブルでは、マテリアルとメッシュとの対応(3.1 節で述べる)、mesh_triangle 及び mesh_vertex テーブルにはメッシュの頂点情報と接続情報を格納した。material テーブルには CAD 側及びゲームエンジン側でのマテリアル名の対応を格納し、マテリアルの読み替えを自動化した。

3. 三次元モデル連携の実装

3.1. ArchiCAD での実装

ArchiCAD から PostgreSQL へのアクセスは、開発者機能を用いてアドオンとして実装した。ArchiCAD のアドオンでは、モデルデータのメッシュ情報に対してアクセスすることができ、これにより頂点位置を抽出した。しかし、ArchiCAD におけるメッシュ情報と UE4 におけるメッシュ情報はともに境界表現(B-Rep)を基本としているものの、作法の相違点が多いことが分かった。以下に作法の違いとその対応について簡単に説明する。なおデータベースに格納するには、どちらかの作法に合わせる必要があるが、本研究では UE4 に合わせる方針を取った。データベースとの情報のやり取りにおいてリアルタイム性が求められる UE4 において、処理コストの増加を避けるためである。

まず、メッシュの法線情報に関して、ArchiCAD は面法線を基本とし、曲面を有するメッシュのみ頂点法線を持つのに対し、UE4 では全て頂点法線である点が異なっていた。形状を正確に抑える必要のある建築 CAD ではハードエッジ、写実性に重視したゲームエンジンではスムーズエッジという考えによるものだと思われる。ArchiCAD でハードエッジとなっている頂点は複製を行い、異なる頂点法線を持つ複数の頂点として出力した。

また、ArchiCAD でのメッシュの構造は、親の Element に対して複数の Body と呼ばれる子要素として格納されている。Body は Element の建築的な意味での構成要素や閉じた形状を単位に分割されている場合が多く、1 つの Body に複数のマテリアルが割り当てられいていることもある。一方、ゲームエンジンでは通例的に、マテリアルごとに対

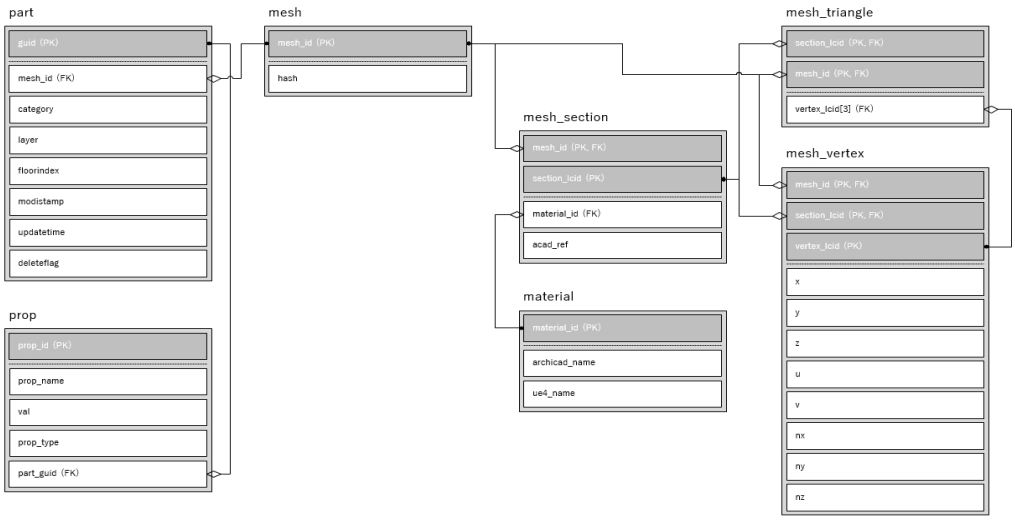

図 4　データベースのスキーマ

してメッシュの子要素を分割する方針が取られている。このため、ArchiCAD から出力する際にマテリアルごとに Section が分けられるよう、メッシュをマテリアル順に再ソートした後にデータベースへと格納した。図 5 に例示するように、ある机は、ArchiCAD においては天板メッシュと、脚 1 本毎のメッシュが 5 脚分の計 6Body で構成されているが、UE4 では天板の木材部と脚の鉄部と脚接地部分の黒い樹脂部の 3Section となっている。

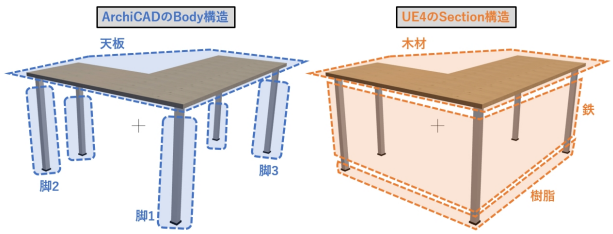

図 5　机でのメッシュ構造の違い

3.2. UnrealEngine4 での実装

UE4 から PostgreSQL へのアクセスは、UE4 のプラグイン作成機能を用いて実装した。UE4 ではブループリントと呼ばれるビジュアルプログラミング手法が推奨されており、処理を意味するコンポーネントは C++の関数として作成することができるため、データベースへの各手続きをコンポーネント化してゲームエンジンからの接続を容易に行えるように実装した(図 6)。データベースから取得したメッシュデータは、RuntimeMeshComponent[6]プラグインによってメッシュ化し描画を行う。このプラグインは、メッシュの頂点座標、頂点法線、頂点 UV、頂点接続情報などをもとに動的にメッシュ生成を行うことをサポートするものである。

ゲームエンジンには、構築した VR シーン（ゲーム）を実行形式として書き出すパッケージ化という機能が備わっている。パッケージ化を行うことで、端的に言えば特別なソフトウェアを用意せずとも VR シーンの再生を行うことができる。本研究では、データベースとの連携を行い、VR シーンを構築するタイミングについて、パッケージ化した状態での行う実装と、ゲームエンジンのエディタ上（パッケージ化をする前）で行う実装の二通りについてシステムを試作した。

前者では、ゲームエンジンに関しての知識が無くても、パッケージ化されたファイルを再生するのみで、VR シーンを利用することができる。具体的には、再生開始時にデータベースへの接続、メッシュ生成が行われ、VR シーンが構築される。また、再生中は常にデータベースに更新の有無を問い合わせているため、建築 CAD から送信された更新情報があれば、VR シーンも更新される。VR シーンの美麗さについては、全オブジェクトが可動的な物となるためにライトビルドと呼ばれる事前計算機能を活用できないという欠点がある。NVIDIA 社の開発のリアルタイム大域グローバルイルミネーション技術 VXGI[7]を導入し、物体同士での光の反射による影の影響や鏡面反射をリアルタイムで計算するなど工夫を行っている。

後者では、ゲームエンジンのエディタ上でデータベースへの接続、メッシュ生成が行われ、VR シーンの作成が行われる。データベースへの問い合わせによりスムーズに建築 CAD 上での更新を反映できる点は前者と同様である。ゲームエンジンの知識・技術を要するものの、例えば通行人の追加やドア等の開閉システムの追加、あるいは描画表現の工夫など、ゲームエンジンに備わっている様々な機能を自由に追加しながら、VR シーンを構築できるという特長がある。パッケージ化前であるので、光環境やそれによる陰影を事前計算することができ、最終的な VR シーンの美麗さはこちらのほうが期待できる。

4. 実装結果

ArchiCAD を開発している GRAPHISOFT 社が、プロジェクト例として配布している RC 造集合住宅のデータ[8]を使用し、実験を行った。UE4 でのマテリアルには、UE4 のマーケットプレイスで提供されている物を一部編集し使用した。物理ベースのマテリアルであるため、ArchiCAD で使用されているマテリアルと比較して非常に実写的なレンダリングが行えていることが見て取れる(図 7)。

図 8 に ArchiCAD の編集画面、同付属の BIMx による VR シーン、本システムにおいて UE4 でパッケージ化後にデータベースへ接続し生成した VR シーンを示す。本システムによる VR シーンでは、材質の質感として、ガラスや金属面への映り込みなどの表現がなされ、また、テクスチャの反復に起因する違和感が解消されており、主観的な評価に留まるが、建築空間の VR 体験に資する VR シーンが得られていると思われる。

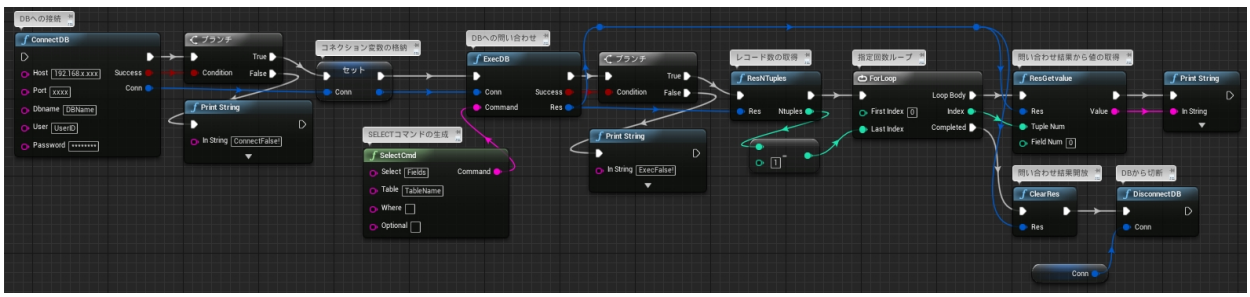

図 6　ブループリントによるデータベースへの接続例

図 7　マテリアルの置き換え例

5. まとめと展望

3.2 節で既述した通り、UE4 からデータベースを介した ArchiCAD へのアクセスはパッケージ化前後の両方で実装したが、特に前者の実装において、本研究で用いた建築モデルは集合住宅形態であり部屋数が多く、オブジェクト約 3000 個が可動性を持つこととなり、かつライトに関しても可動であるため、描画フレームレートが 10～30fps 程度まで落ち込んでしまった。VXGI の設定やメッシュの生成システムの設定を見直すことで多少の改善は見込めるが、VR に求められる 90fps を達成するにはハードウェアの更新やシェーダーアルゴリズムの改善も必須である。

本研究では ArchiCAD から UE4 への一方的なモデルの移行に留まっているが、UE4 でモデルに対して行った変更を ArchiCAD へ反映させることも想定しデータベースのスキーマを設計している。ArchiCAD のオブジェクトが持つ変更可能なパラメータ群をデータベースの prop テーブルで格納し、UE4 側からこの値を変更し、ArchiCAD へ更新リクエストを行うようなかたちで、モデルに対する操作を行う機能を追加することを計画している(図 9)。

図 9　UE4 でのオブジェクト形状の変更の方法

[参考文献]

1) 竹澤拓晃、大西康伸、本間里見『BIM と VR ツールの連携による建築設計における外構デザイン検討に関する研究』日本建築学会大会(中国)学術公演梗概集、2016.8
2) 積木製作による「BIM データを活用した教育 VR システム」 http://tsumikiseisaku.com/vrox/result/obayashi.html
3) GRAPHISOFT「ARCHICAD | グラフィソフトジャパン」 http://www.graphisoft.co.jp/archicad/
4) EPIC GAMES「UNREAL ENGINE」 https://www.unrealengine.com
5) PostgreSQL Global Development Group「PostgreSQL」 https://www.postgresql.org/
6) Koderz「RuntimeMeshComponent」 https://github.com/Koderz/RuntimeMeshComponent
7) NvPhysX「NVIDIA GameWorks UnrealEngine-VXGI」 https://github.com/NvPhysX/UnrealEngine/tree/VXGI
8) GRAPHISOFT「ARCHICAD サンプルプロジェクト」 http://www.graphisoft.co.jp/download/sampleproject/

＊1　千葉大学大学院 工学研究科 博士前期課程
＊2　千葉大学大学院 融合理工学府 博士前期課程
＊3　千葉大学大学院 工学研究院 助教
＊4　千葉大学大学院 工学研究院 教授

図 8　ArchiCAD(上段)と BIMx(中段)と UE4(下段)での描画の違い