令和 6 年度 卒業論文

Web 技術を活用した軽量な 3D エディタの開発

一関工業高等専門学校
未来創造工学科 情報・ソフトウェア系 水津研究室
伊藤瞭汰

2025 年 2 月 14 日

Abstract

Developing a Lightweight 3D Editor with Web Technologies

This research addresses the growing need for accessible 3D editors that function seamlessly across a wide range of devices, including mobile devices and high-performance PCs. By enhancing the Web3D Viewer, a web-based 3D model viewer developed in a prior study, this project aimed to improve performance, display quality, and user experience. The previous Web3D Viewer, while offering cross-platform support and offline functionality via WebGL, JavaScript, and PWA technology, suffered from performance bottlenecks when handling large 3D models, displayed incorrect textures on specific models, and lacked advanced editing capabilities.

To remove these restrictions, this research implemented several key improvements. Level of Detail (LOD) and instancing were incorporated to enhance rendering performance, particularly for complex scenes. Draco compression was supported to reduce model file sizes and loading times. Environment lighting was properly configured to ensure accurate material display, resolving previous texture issues. Additionally, a progress bar was added to provide users with feedback during model loading.

These enhancements resulted in significant performance gains, especially for large models, leading to smoother operation on mobile devices. While testing confirmed improved loading times and frame rates compared to the previous version, challenges persist in loading very large models on resource-constrained mobile devices. Furthermore, loading speeds have yet to match those of established desktop applications like Blender.

Future work will prioritize further optimization for mobile platforms, explore automatic file compression techniques, and integrate more intuitive modeling features. Ultimately, this research seeks to lower the barrier to entry for 3DCG technology, promoting its wider adoption in fields such as education, research, and industry.

目次

1. はじめに

近年，3DCG（3 次元コンピュータグラフィックス）は，映画やゲームといった映像分野に留まらず，VR（仮想現実），AR（拡張現実），医療分野でのデータ可視化[1]，建築設計，都市計画シミュレーションなど，多様な領域で活用されており，その需要は高まる一方である．また，令和 5 年度からの高等学校学習指導要領の改訂により，コンテンツ制作に関する技術として，3DCG が扱われることとなった．[2]しかし，大規模な 3DCG データを扱う際には，端末の性能，特にメモリ容量や GPU 性能に関する制約が存在する．現在広く利用されている多くの 3DCG 作成ツールは，快適な動作のために一定水準以上のハードウェアを要求する．そのため，3DCG の利用が一般化しつつある現在でも，既存のツールを使いこなすには，ユーザーが自身の端末環境を考慮する必要があり，3DCG の活用，学習，またはその分野への参入には一定の困難が伴う．

このような背景から，特定の端末性能に依存せず，より利用しやすい 3DCG 作成ツールの必要性が高まっている．本研究では，この課題を解決すべく，端末に制約されず，軽量で，基本的なモデリング機能を備えた 3D エディタの開発を目的とする．先行研究[3]においては，上記の目的のもと，Web 技術を活用した 3D データビューワの開発が行われた．本研究では，先行研究で作成された 3D ビューワに存在する課題，特にパフォーマンスの問題と機能不足を解消し，最終的な目的の達成を目指す.

先行研究では，特定の環境下，特にモバイル端末で大規模な 3D モデルを読み込んだ際にフレームレートが低下するという問題が残されていた．また，機能面では，3D データの表示，共有，注釈機能が実装されているものの，より高度な編集機能は不足している．本研究ではこれらの課題のうち，大規模モデルを扱う際のパフォーマンス改善を最優先事項とし，モバイル端末でもスムーズに動作することを目指す.

2. システム改善の構想

本章では，先行研究で開発された Web3D ビューワ(以下，Web3D ビューワ)と代表的な Web ベースの 3D エディタの比較分析，および Web3D ビューワの現状分析を通じて，本研究で取り組むべき課題を明確にし，その解決に向けた改善目標と基本方針を示す.

2-1. 既存システムの分析と課題の明確化

Web3D ビューワは, Babylon.js[4]を用いて 3D グラフィックスを描画する Web アプリケーションであり，3D データの閲覧，共有，注釈機能を備えている．また，PWA（Progressive Web Apps）技術を採用することで，Web アプリケーションでありながらネイティブアプリケーションと同等のユーザー体験を提供し, アプリケーション自体のキャッシュによりオフラインでの動作も実現している．さらに，.obj 形式，.gltf/.glb 形式，.stl 形式の読み込みに対応しており，.gltf/.glb 形式ではアニメーションの再生もサポートする．

Web3D ビューワの現状を把握するため，表 1 に示す類似ツール(Blender[5]，Spline[6]，Vectary[7])との機能比較を行う．比較要件として，クロスプラットフォーム対応，オフラインでの動作，3D データの編集機能，3D データのインポート，アニメーションの読み込み機能，Web サイトへの埋め込み，注釈機能，ファイル容量制限とした．

表 1 Web3D ビューワと類似ツールの比較

要件	Web3Dビューワー	Blender	Spline	Vectary
クロスプラットフォーム	○	○	○	○
オフラインでの動作	○	○	×	×
3Dデータの編集	△	○	○	○
3Dデータのインポート	○	○	○	○
アニメーションの読み込み	○	○	○	○
3Dデータの共有	○	×	○	○
Webサイトへ3Dデータの埋め込み	×	×	○	○
注釈・コメント機能	○	×	○	○
ファイル容量制限	なし	なし	60MB	50万ポリゴン

この表から，Web3D ビューワはファイル容量制限がなく，オフライン動作が可能

である点において，他の Web アプリケーション（Spline, Vectary）とは異なる特徴を持つことがわかる．また，データの共有が可能である点は，高機能なデスクトップアプリケーションである Blender にはない利点である．しかし，3D データの編集機能に関しては，他のツールと比較して限定的であり，今後の改善が期待される．

次に，先行研究で実施された検証結果を分析し，Web3D ビューワの現状の問題点を整理する．表 2 に，先行研究で実施された，各アプリケーションでの 3D モデル読み込み時間に関する検証結果を示す．表 3 と図 1，2，3 に，この検証に用いられた 3D モデルの概要を示す．

表 2 各データ読み込み時にかかる時間(秒)

モデル	Web3Dビューワ	Windows3Dビューワ	Blender
①	0.86	0.556	0.48
②	0.722	1.31	0.34
③	26.76	21.85	18.724

表 3 検証に用いられた 3D モデル

ラベル	モデル	容量(MB)
DS1	様々な材料情報が含まれる3Dデータ	4.683
DS2	アニメーションが含まれる3Dデータ	0.118
DS3	東京都渋谷区周辺の建物や 地形情報が含まれる3Dデータ	319.828

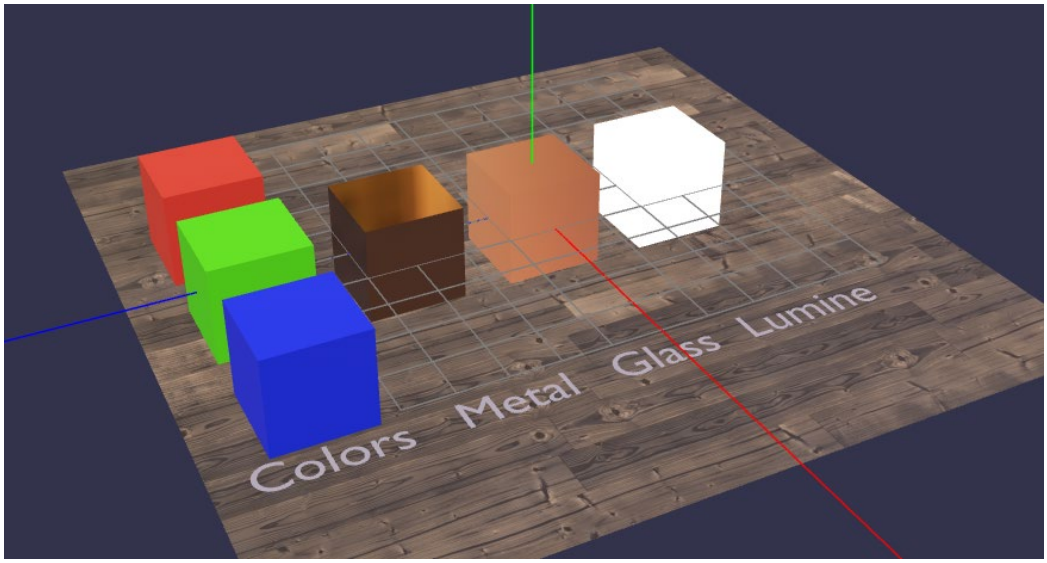

図 1 DS1 - 様々な材料情報が含まれる 3D データ

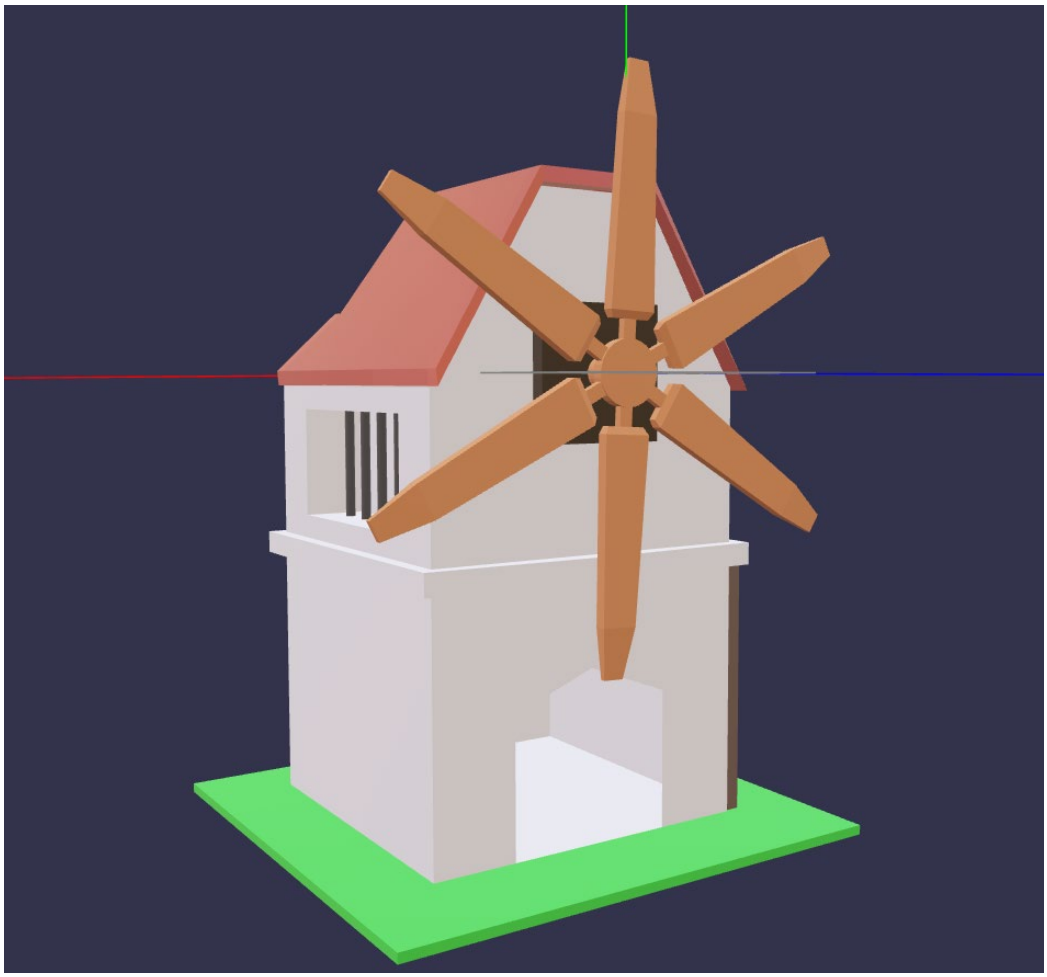

図 2 DS2 - アニメーションが含まれる 3D データ

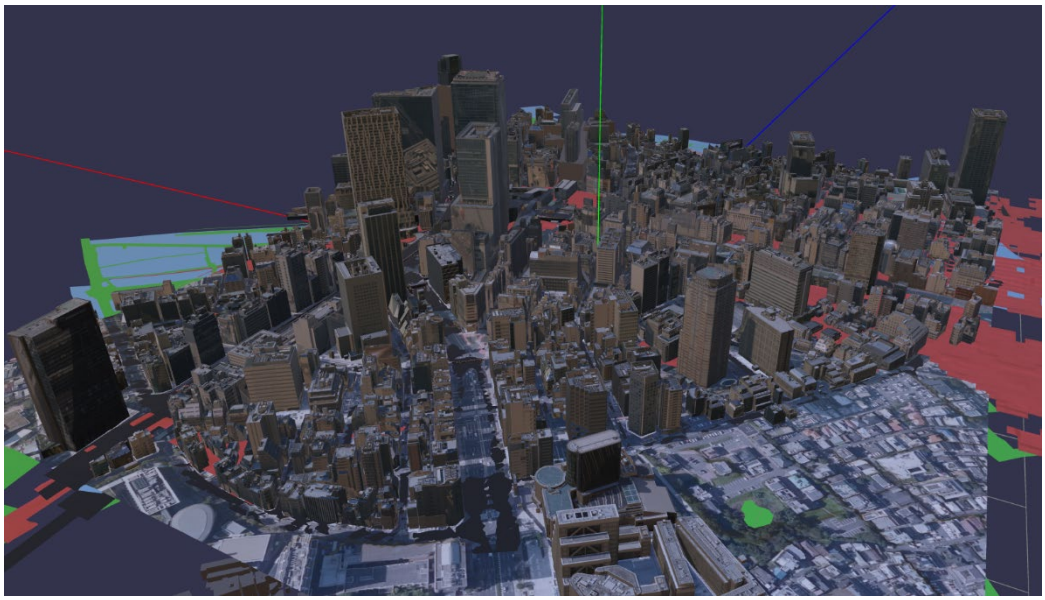

図 3 DS3 - 東京都渋谷区の建物や地形情報が含まれる 3D データ

表 2 の結果から，Web3D ビューワは，他のアプリケーションと比較して，特に大規模モデル(DS3)の読み込みに時間を要するものの，実用上は問題ないレベルであることが確認できる．

また，表 4 に，先行研究で実施された，異なる端末での Web3D ビューワのフレームレートに関する検証の結果を示す．この検証で使用した端末を表 5 に示す．

表 4 端末ごとのフレームレートの検証結果(fps)

	①	②	③
PC1	60	60	6.77
PC2	60	60	40.7
PC3	60	60	6.93
PC4	60	60	54.1
MB1	49.87	59.97	-
MB2	59.9	59.9	-

表 5 検証に使用した端末一覧

ラベル	CPU/SoC	GPU	RAM(GB)	OS	画面解像度
PC1	AMD Ryzen5 3600	RTX2060	16	Windows 10	Full HD
PC2	Intel Core i5 8600K	RTX 2070	32	Windows 10	Full HD
PC3	AMD Ryzen5 5600	RTX 2070	16	Windows 10	Full HD
PC4	Apple M1 Pro	-	16	macOS 13.1	3024 x 1964
MB1	Apple A14 Bionic	-	4	iPad OS 16.1	2360 x 1640
MB2	Snapdragon 855	-	8	Android 12	2280 x 1080

表 4 の結果から，Web3D ビューワは，特定の環境，特にモバイル端末（MB1, MB2）で大規模モデル（DS3）を読み込んだ際に，フレームレートが著しく低下するか，読み込み自体が失敗することが明らかとなった．また，一部のモデルでは，テクスチャが正常に表示されない問題も確認された．

以上の分析から，Web3D ビューワには，以下の課題が存在すると結論づける．

- 大規模モデル読み込み時のパフォーマンス低下
- 画像マテリアル付きデータの表示の問題
- 3D データの編集機能の不足

2-2. 改善目標と基本方針

本研究では，上記の課題を解決し，「端末性能に制約されず，軽量で，基本的な機能を備えた 3D エディタ」の実現を目指す．具体的には，大規模モデル読み込み時のフレームレートを向上による安定した動作の実現, 特定のモデルで発生していたテクスチャ表示の問題の解消，細部の改善を通じた快適な利用環境の提供の 3 つを改善目標とする．

これらの目標を達成するため，本研究では，主にモデルの最適化，ファイルサイズの削減, マテリアル表示問題の修正という 3 つの技術的アプローチを採用する．モデルの最適化では，LOD(Level of Detail)やインスタンシングの技術を用いて，描画不可を軽減する．ファイルサイズの削減では，Draco 圧縮に対応し，ロード時間の短縮のために

圧縮を可能にする．マテリアル表示の修正では，環境光と環境テクスチャの設定を見直し，マテリアルが正しく表示されるように修正を行う．

なお，これらの改善手法の詳細は，「4. 具体的な改善手法」にて詳述する．

3. 使用する技術の詳細

本章では，前節の構想を達成するために採用した主要な技術について，概要と具体的な実装方法について述べる．

本研究では，主に以下のライブラリを使用した(表 6)．

表 6 使用したライブラリ(抜粋)

名称	バージョン	用途
babylon.js	7.40.0	3D描画
chakra-ui/react	2.3.7	UI構築
supabase/supabase-.js	2.9.6	データベース
next	12.2.5	フレームワーク
react	18.2.0	フレームワーク

3-1. LOD：Level of Detail

本研究では，描画負荷を軽減し，パフォーマンスを向上させるため，LOD (Level of Detail) 技術を採用した．LOD は，カメラからの距離に応じて 3D オブジェクトの精細度を動的に変化させる手法である．具体的には，カメラに近いオブジェクトは高精細なモデルで表示し，遠いオブジェクトは頂点数を削減した低精細なモデルで表示することが可能である．

Babylon.js では，LOD を容易に実装するための専用メソッドが提供されている．Web3D ビューワでは，addLODLevel メソッドを用いて LOD を実装した．このメソッドにより，メッシュに対して複数の精細度レベルを設定し，カメラから各メッシュまでの距離に応じて表示するモデルを動的に切り替える事ができる．

3-2. Draco 圧縮

大規模な 3D モデルのファイルサイズを削減し, ロード時間を短縮するため Draco 圧縮[8]技術を採用した．Draco 圧縮は，Google が開発したオープンソースの 3D モデル圧縮ライブラリであり，頂点情報，インデックス情報，テクスチャ座標などのジオメトリデータを効率的に圧縮する事ができる．圧縮されたデータは，専用のプラグインまたはデコーダーを用いて解凍する必要がある．

Web3D ビューワでは，Babylon.js で提供されている機能を利用して，Draco 圧縮

されたファイルを読み込み，WebAssembly ベースの高速なデコーダーを用いてクライアントサイドで解凍する．これにより，ファイルサイズを大幅に削減し，ネットワーク経由のデータ転送量が減少し，ロード時間の短縮が見込まれる．

3-3. インスタンシング

本研究では，同一形状のオブジェクトを複数描画する際の描画負荷を低減するために，インスタンシング技術を採用した．インスタンシングは，基本となるジオメトリデータをメモリ上で共有し，各オブジェクトには位置，回転，スケールなどの属性情報のみを持たせることで，GPU 上のメモリ使用量を削減できるだけでなく，描画のためのデータ転送量や，CPU から GPU への描画命令の発行回数（ドローコール数）を削減する手法である．

Web3D ビューワでは，主に Babylon.js の Mesh クラスの createInstance メソッドなどを用いてインスタンシングを実装した．

4. 具体的な改善手法

本章では，前章で述べた改善目標を達成するために具体的にどのような手法を用いてWeb3D ビューワの改善を行ったのかを詳述する．

4.1. 実装アーキテクチャ

図 4 に，改善後の Web3D ビューワのアーキテクチャを示す．

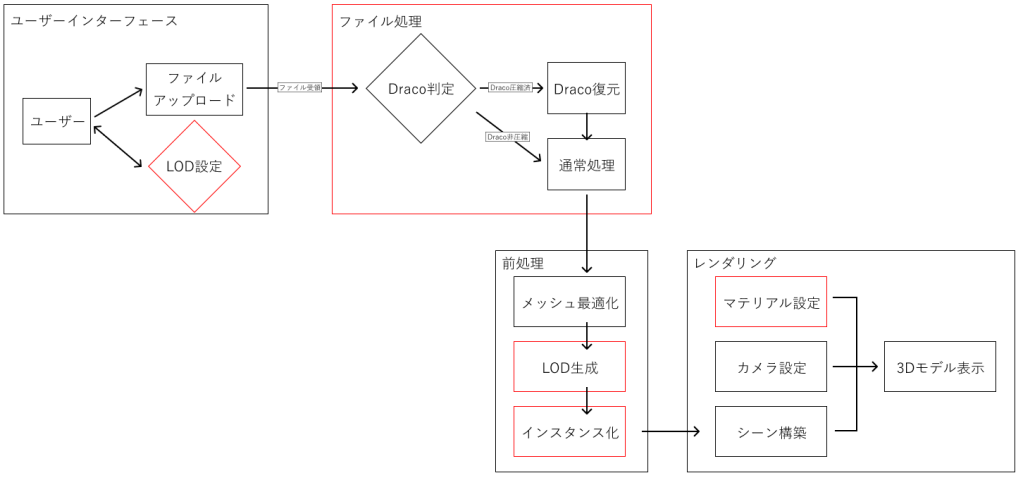

図 4: 改善後の Web3D ビューワのアーキテクチャ

本研究では，既存の Web3D ビューワの基本構造は維持しつつ，パフォーマンスとロード時間に影響を与える特定のモジュールに対して，重点的に改善と拡張を行った．具体的には，ファイルローダー，読み込まれたデータをシーンに反映させるモジュール，そして，新たにメッシュ最適化モジュールを追加した(図 4 の赤枠部)．ファイルローダーでは，既存の機能に加え，Draco 圧縮ファイルの読み込み，及びプログレスバー表示機能を追加した．また，シーン管理モジュールでは環境光と環境テクスチャの設定処理を追加し，マテリアルの表示問題を解消した．メッシュ最適化モジュールでは，LOD とインスタンシングの適用を行う．

4-2. 軽量化・高速化のための具体的な実装

本研究では，Web3D ビューワの軽量化と高速化を実現するため，主に以下の 4 つの改善を行った．

4-2-1.LOD の適用

Web3D ビューワでは，ユーザーがアップロードしたモデルに対して，自動的に 3 段階の LOD を生成する機能を実装した．具体的には，以下の手順で LOD を適用する．

1. 元のモデルを高精細モデルとする．
2. 頂点数を削減した低精細モデルを中間生成する．（現状では，元のモデルのメッシュを 1/3 に削減）
3. Babylon.js の addLODLevel メソッドを用いて，高精細モデルと低精細モデルそれぞれに，カメラからの距離に応じた表示・非表示の閾値を設定する．

なお，閾値は環境やモデルの大きさによって表示結果が異なるので，ユーザーがパラメータとして調整できるように設計した．

4-2-2.インスタンシングの実装

Web3D ビューワには，シーン内にジオメトリとマテリアルが同一のメッシュが存在するかを検査する機能を備えている．同一のメッシュが複数存在する場合は，インスタンスとして生成し直す．これにより，ドローコール数が削減し，多数の同一オブジェクトを含むシーンの描画パフォーマンスが向上する．

4-2-3.ファイル読み込み方法の改善

ファイル読み込み方法に関しては，主に Draco 圧縮への対応と，プログレスバーの表示を行った．Draco 圧縮に関しては，Babylon.js の DracoCompression モジュールを用いて，事前に Draco 圧縮を適用した.glb ファイルの読み込みを可能とした．これにより，ファイルサイズの大幅な削減が期待できる．ただし，ユーザーが事前にファイルを Draco 圧縮しておく必要がある．プログレスバーの表示では，モデルのロード中および Draco 圧縮モデルのデコード中にプログレスバーを表示し，ユーザーに進行状況を視覚的に示すようにした．

4-2-4.環境光と環境テクスチャの設定

先行研究の Web3D ビューワでは，特定のモデル（モデル③）において，画像を用いたマテリアルが正しく表示されない問題が発生していた（図 5）．

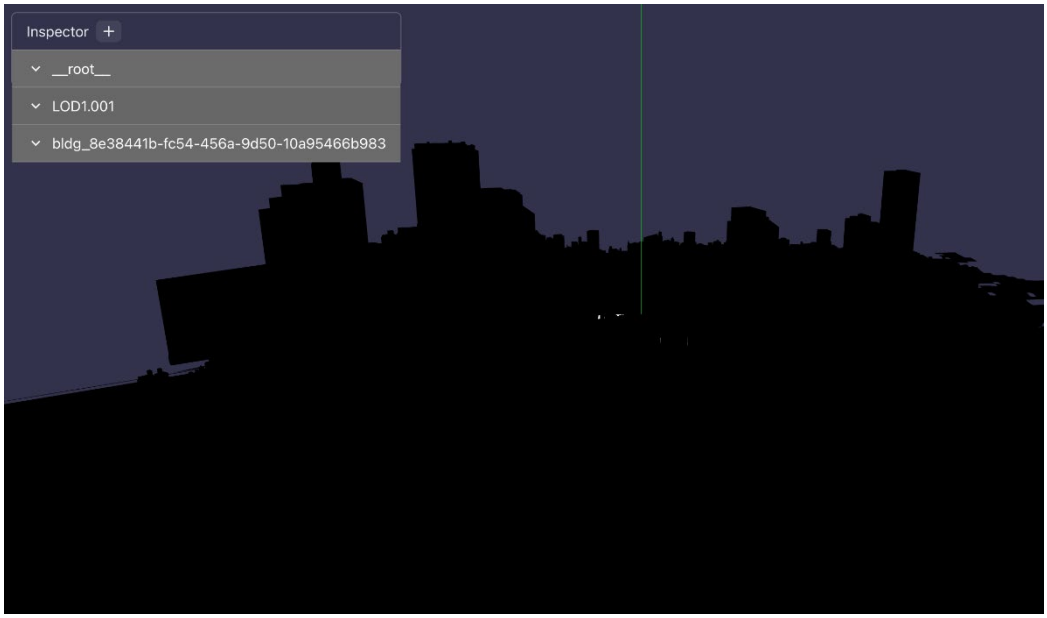

図 5 改善前の Web3D ビューワにおける画像付きモデルの表示

この問題は，PBR マテリアルが環境からの光を適切に受け取るための環境光と環境テクスチャの設定が不足していたこと，および，ロードされたモデルの PBR マテリアルに対して，シーンの環境テクスチャが適用されていなかったことが原因であった．本研究では，シーンにデフォルトの環境光と環境テクスチャを作成するとともに，各メッシュの PBR マテリアルの反射テクスチャに，作成した環境テクスチャを適用することで，この問題を解消した（図 6）．

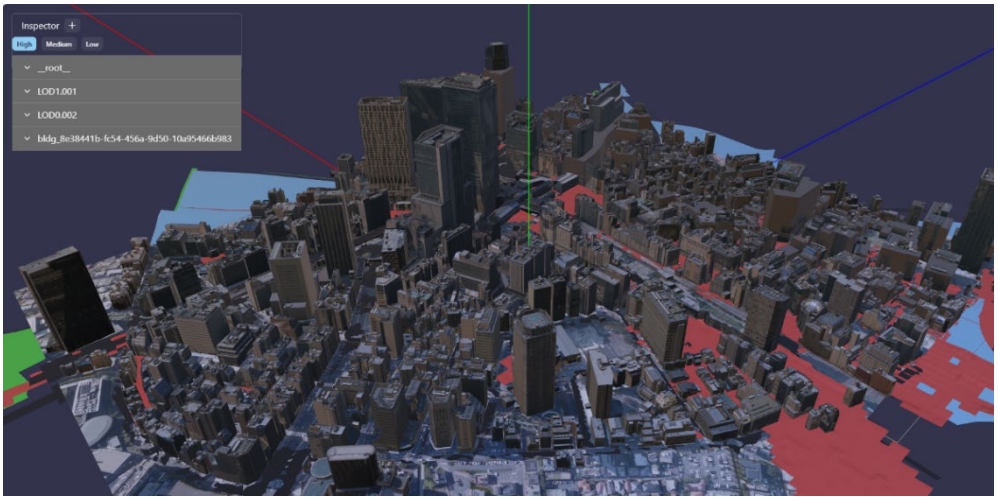

図 6 改善後の Web3D ビューワにおける画像付きモデルの表示

5. 検証

本研究では，改善後の Web3D ビューワの性能を評価するため，インポート速度とフレームレートに関する 2 つの検証を行った．

検証には，表 5 に示す端末を使用し，Web3D ビューワには同一 LAN に接続された別の端末でホスティングし，検証端末から Google Chrome のシークレットモードでアクセスした．検証端末では，Google Chrome 以外のアプリケーションはすべて終了し，ハードウェアアクセラレーションを ON に設定した．

5-1.検証 1：アプリケーションごとのインポート速度の比較

本検証では，Web3D ビューワと，表 1 に示した類似アプリケーション(Blender, Spline，Vectary)との間で，3D モデルのインポート速度を比較した．

検証には，表 7 に示す端末とアプリケーション，表 8 に示す 3 つの 3D モデルを使用した．ここで，モデル①，モデル②は先行研究で使用されたモデル（図 1，2），モデル③は先行研究の検証で使用された都市データをより広範囲に拡大したものである．

表 7 検証に使用する端末

項目	詳細
CPU	Intel® Core™ i7-12700
GPU	RTX 3070
RAM(GB)	32
OS	Windows 10
Google Chrome	132.0.6834.160
Blender	4.1.1
Spline	11-12-2024
Vectary	2025年2月8日時点での最新版

表 8 検証に使用する 3D モデル

ラベル	モデル	容量(MB)	出典
①	様々な材料情報が含まれる3Dデータ	4.683	先行研究
②	アニメーションが含まれる3Dデータ	0.118	先行研究
③	東京都渋谷区周辺の建物や 地形情報が含まれる3Dデータ	735.578	Plateau[9]

5-1-1.検証方法

各アプリケーションで，以下の手順で 3D モデルを読み込み，読み込み時間を測定した.

1. アプリケーションを起動し，3D データをインポートするためのファイルダイアログを開く．
2. ファイルを選択すると同時に，タイマーで測定を開始する．
3. 3D モデルのオブジェクト及びマテリアルが完全に表示された時点で計測を停止する．
4. アプリケーションを終了する．
5. 1.から 4.の手順を 5 回繰り返す.

上記の手順を，表 6 に示す各モデルで実施し，平均読み込み時間を算出した．

5-1-2.検証結果

検証結果を表 9 および図 7 に示す.

表 9 各アプリケーションの平均読み込み時間(秒)

モデル	Web3Dビューワ	Blender	Spline	Vectary
①	1.16	1.00	1.05	1.71
②	0.99	0.85	0.63	1.43
③	33.35	20.42	-	-

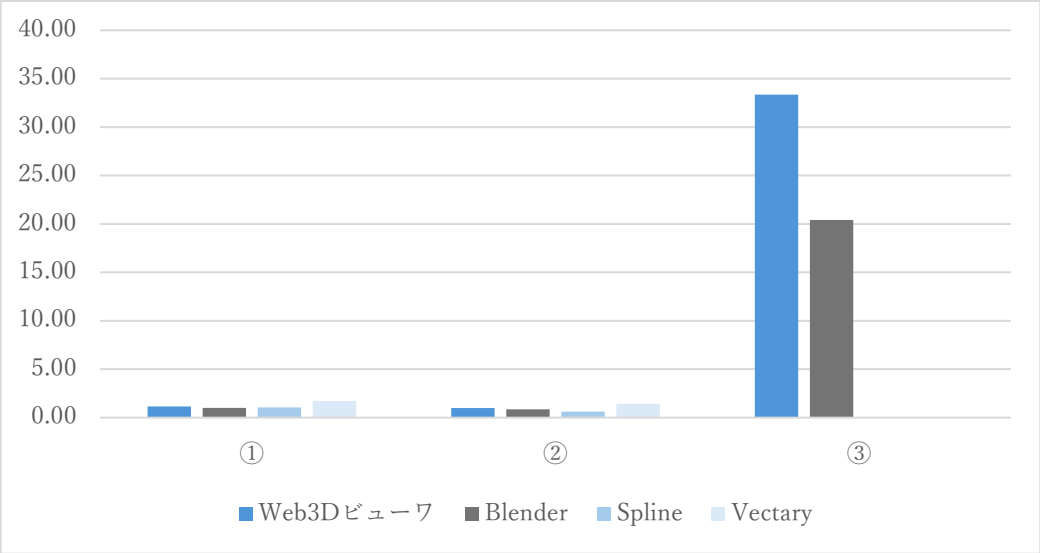

図 7 各アプリケーションの平均読み込み時間

図 7 より, 改善後の Web3D ビューワは Blender を除くアプリケーションと同等の速度で 3D モデルを読み込めることがわかる．また，Spline と Vectary では読み込みができなかった大規模モデル(モデル③)についても，Web3D ビューワでは平均 33.35 秒で読み込むことができた．

5-2. 検証 2：各端末によるインポート時間およびフレームレートの比較

本検証では，改善前後の Web3D ビューワの性能を比較するとともに，異なる端末での動作の違いを調査した．特に，本研究で改善を行ったインポート速度とフレーム

レートに着目した.

5-2-1.検証方法

表 10 に示す各端末で，以下の手順で Web3D ビューワ(改善前および改善後)の動作を検証した.

1. Web3D ビューワ（改善前または改善後）で，5-1-1.の手順 1.から 4.と同様の手順で 3D モデルを読み込む.
2. 3D モデルが描画された後，2 分間，以下の操作をランダムに繰り返す.
   ・視点移動(パン，チルト，回転)
   ・オブジェクト移動
   ・オブジェクトの拡大・縮小
3. 操作中，Chrome DevTools[9]の FPS メーターを用いて，フレームレートを継続的に監視し，最大値と最小値を記録する.
4. Web3D ビューワを終了する.
5. 1.から 4.の手順を 5 回繰り返す.

上記の手順を，表 8 に示す各 3D モデル，各端末，改善前後の Web3D ビューワで実施し，平均読み込み時間，フレームレートの最大値と最小値を記録した.

表 10 検証 2 に使用した端末一覧

ラベル	CPU/SoC	GPU	RAM(GB)	OS	画面解像度	リフレッシュレート(fps)
PC1	AMD Ryzen5 3600	RTX 2060	16	Windows 10	1920 x 1080	60
PC2	AMD Ryzen 5 8400F	RTX 4060	32	Windows 11 Pro	1920 x 1080	60
PC3	Intel® Core™ i7-12700	RTX 3070	32	Windows 10	1920 x 1080	144
PC4	Apple M1	-	8	macOS 15.3	1440 x 900	60
MB1	Apple A14 Bionic	-	4	iPad OS 18.0.1	2360 x 1640	60
MB2	Apple A15 Bionic	-	8	iOS 18.2.1	2532 x 1170	60

5-2-2.検証結果

検証結果を表 11 および図 8 に示す．なお，表中の「*」は，テクスチャが正常に表示されなかったことを示し，「-」はモデルが表示できなかったことを示す.

表 11 端末ごとの Web3D ビューワの動作検証結果

端末	ラベル	改善前読み込み時間(秒)	改善後読み込み時間(秒)	改善後フレームレート(fps)
PC1	①	1.43	0.85	45.6 - 60.0
	②	1.15	0.76	45.0 - 60.0
	③	48.79*	46.09	45 - 60
PC2	①	0.93	0.91	45 - 60
	②	0.85	0.85	45.8 - 60
	③	34.75*	33.66	45 - 60
PC3	①	1.16	0.99	139.2 - 144
	②	1.08	0.90	129.5 - 144
	③	13.93*	35.86	129.6 - 144
PC4	①	1.66	1.16	45.6 - 60
	②	1.37	0.99	45.6 - 60
	③	31.45*	33.35	54.0 - 60
MB1	①	2.25	1.85	58 - 60
	②	2.15	1.68	59 - 60
	③	-	-	-
MB2	①	2.21	1.93	53 - 60
	②	1.90	1.79	58 - 60
	③	-	-	-

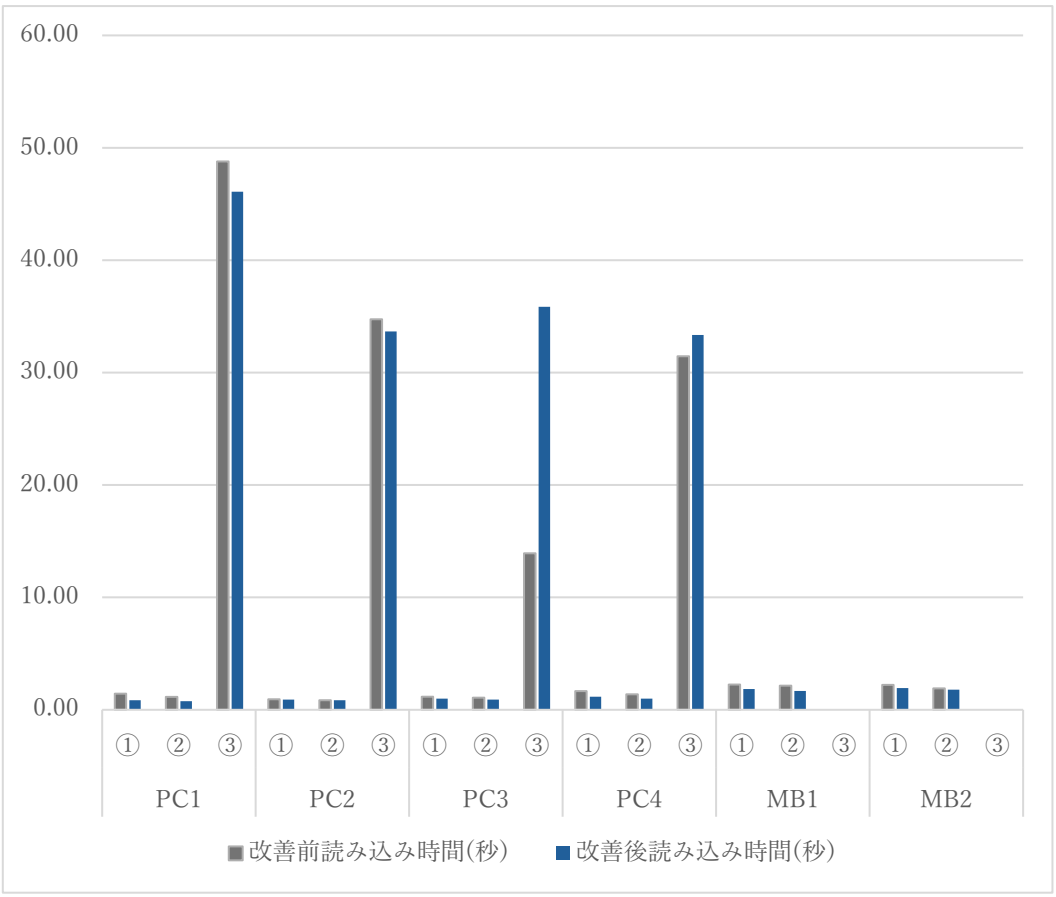

図 8 端末ごとの Web3D ビューワの動作検証結果

表 11 及び図 8 より，改善後の Web3D ビューワは，改善前と比較して，正常に読み込みができたデータでは平均読み込み時間が平均で 18.7%減された．また，フレームレートは多くの端末で 45FPS から 60FPS で安定しており，特に大規模データ(モデル③)を扱った際に，先行研究と比較して大幅に改善されたことがわかる．しかし，モバイル端末(MB1，MB2)では，依然として大規模モデルの読み込みができない問題が残ることが確認された．

6. 考察・今後の展望

本研究では，Web3D ビューワの軽量化と利用環境の拡大を目指し，主に読み込み時間とフレームレートの改善に取り組んだ．検証の結果，改善後の Web3D ビューワは，先行研究と比較して，3D モデルのインポート速度が向上し，特に大規模モデル（モデル③）を扱う際のフレームレートが安定したことが確認された．

具体的には，検証 1 において，Web3D ビューワは，他の Web アプリケーションと同等の速度で 3D モデルを読み込めることが示された．しかし，Blender と比較すると，大規模モデル（モデル③）の読み込みに時間を要することも明らかになった．これは，Blender が内部で画像圧縮処理を行うことで，ビューポート表示を最適化しているためと推測される．したがって，Babylon.js を用いて Blender と同等以上の性能を実現するには，モデルのロード前に Basis Universal などを用いた KTX2 形式への圧縮変換を検討するなど，データ戦略の見直しが必要であると考える．

検証 2 においては，改善前後の Web3D ビューワを比較し，読み込み時間が平均で約 20% 短縮した．また，先行研究と比較して大規模モデルを扱う際のフレームレートも大幅に向上した．しかし，モバイル端末（MB1, MB2）では，大規模モデル（モデル③）の読み込み中にページがリロードされる問題が依然として残る．これは，ブラウザが Web3D ビューワに割り当てるメモリ容量が不足しているためと考えられる．このことから，モバイル端末での利用を前提とする場合は，さらなる軽量化に加えて，マテリアル表示の ON/OFF 機能の実装や，より徹底的なメモリ管理方法の導入が必要であると考えた．

今後の展望としては，まず，モバイル端末でのパフォーマンス改善のための機能追加が必要である．具体的には，モデルの分割表示機能，テクスチャ解像度の動的変更機能，不要なオブジェクトの非表示化などを導入し，メモリ消費量を抑制することが必要であると考えられる．

さらに，3D エディタとしての機能拡充も重要な課題である．現状では，基本的な 3D データの表示，共有，注釈機能のビューワとしての機能が実装されているが，さらに実用的な 3D エディタとして利用されるためには，より高度なモデリング機能や，アニメーション編集機能などの追加が望まれる．また，3DCG の初学者でも容易に操作できるよう，3DCG の概念を学ぶチュートリアルやサンプルデータの提供，学習支援機能の充実も検討すべきである．これらの取り組みを通じて，本システムが，より幅広いユーザー層にとって 3DCG を扱うための身近なツールとなることが期待される．

謝辞

本研究の開発を進めるに当たり 1 年間ご指導いただきました水津俊介先生に深く感謝を申し上げます．また，当研究員の一員であり，本研究に多大なるご助言をいただきました川村樹君，平野俊君，藤原捷羽君，三浦優真君に感謝の意を表します．

参考文献

[1]. Shuguang Yuan, H.C. Stephan Chan, Zhenquan Hu：”Implementing WebGL and HTML5 in Macromolecular Visualization and Modern Computer-Aided Drug Design”, 2017

[2]. 文部科学省：【情報編】高等学校学習指導要領(平成 30 年告示)解説 https://www.mext.go.jp/content/000166115.pdf, pp.152 – 160

[3]. 長田智也, “Web 技術を活用した 3D データビューワの開発”, 一関工業高等専門学校 (2023)

[4]. Babylon.js: Powerful, Beautiful, Simple, Open – Web-Based 3D At Its Best https://www.babylonjs.com/, (最終閲覧日：2025 年 2 月 9 日)

[5]. Home of the Blender project – Free and Open 3D Creation Software https://www.blender.org/, (最終閲覧日：2025 年 2 月 9 日)

[6]. Spline – 3D Design tool in the browser with real-time collaboration https://spline.design/, (最終閲覧日：2025 年 2 月 9 日)

[7]. Vectary – Build interactive 3D and AR solutions online https://www.vectary.com/, (最終閲覧日：2025 年 2 月 9 日)

[8]. Draco 3D Graphics Compression https://google.github.io/draco/, (最終閲覧日：2025 年 2 月 9 日)

[9]. Plateau[プラトー] | 国土交通省 https://www.mlit.go.jp/plateau/, (最終閲覧日：2025 年 2 月 9 日)

[10]. Chrome DevTools | Chrome for Developers https://developer.chrome.com/docs/devtools, (最終閲覧日：2025 年 2 月 9 日)