簡単！軽くてカラフルな Voxelized Game Engine

Last Updated Date: 2025-12-3
Date Created: 2025-12-15
武井陽靜 (Yousei Takei)
_@tsei.jp

Proof of Concept

デモ: glre.dev

ボクセル
- 世界で最も人気なゲームで使用
- データ効率と実装簡易性を両立

TypeScript/Rust/glre
- ブラウザ実行と高性能計算

WebAssembly/WebGL/WebGPU
- クロスプラットフォームな 3D 処理基盤

Install

```
> npm i voxelized-js
```

Repository

github.com/tseijp/voxelizer

Homepage

voxelizer.tsei.jp

1.1 簡単！すぐに実装できる

Web 標準技術によるライブラリ化で開発障壁を除去

現状のコンテンツの課題 （as-is）

Unity 前提のライブラリ
- 複雑な環境構築
- プロプライエタリ
- Web 対応が困難

専用データ形式
- Unity 専用アセット
- バイナリ形式
- 変換ツール必須

改修ポイント （to-be）

ライブラリとして配布
- npm install のみ
- オープンソース
- ブラウザ直接実行

標準データ形式
- 画像データ
- CDN 配信対応
- キャッシュ効率

1.2 簡単！座標系

3 つの異なる座標系を画像ピクセル基準に統合し、座標変換の複雑性を排除

地図難しい 😭

```
- PLATEAU 座標系 (EPSG:6677 - JGD2011 / 日本平面直角座標系第 IX 系)
    - 原点は日本中央部に位置し、偽東距/偽北距を持つ
    - X 軸は東向き（正の値は東方向に増加）
    - Y 軸は北向き（正の値は北方向に増加）
    - 単位は原点からのメートル
    - 東京エリアのモデルは通常 X 値が約-5000m、Y 値が約 48000m
- ウェブメルカトルタイルシステム
    - ズームレベル 0-18 のグローバルタイリングスキーム
    - レベル 18 は北緯 35° で約 125m/タイルを提供
    - タイル座標は東方向（X）と南方向（Y）に増加
    - 各タイルは 256×256 ピクセルを含む
    - ビューア/ワールド座標系: +X=東, -Z=北, +Y=上。
- 重要な変換要件: これらのシステム間の変換には以下が必要です：
    1. PLATEAU メートル座標を地理座標（緯度/経度）に変換
    2. 地理座標をウェブメルカトルタイル座標に投影
    3. テクスチャマッピング用のタイル内の正確なピクセルオフセットを計算
    4. 目に見える位置ずれを防ぐためサブメートル精度を維持
```

1.3 簡単！シンプルなデータ構造

三角形ベースの複雑なデータから 0/1 配列への転換により、演算を最適化

3D モデル （as-is）

.obj + .mtl + 画像

三角形の集合:
- 衝突判定: $O(n^2)$
- 全三角形を検査
- AABB 等で最適化が必要

レンダリング:
- 数万回の draw call
- 複雑な material 管理

ボクセルデータ構造 （to-be）

3 次元の 0/1 配列

1 ブロック = 0 or 1:
- 衝突判定: O(1)
- 座標から直接判定
- 整数計算のみ

レンダリング:
- 1 回の draw call
- ビット演算で最適化可能

2.1 軽い！描画処理

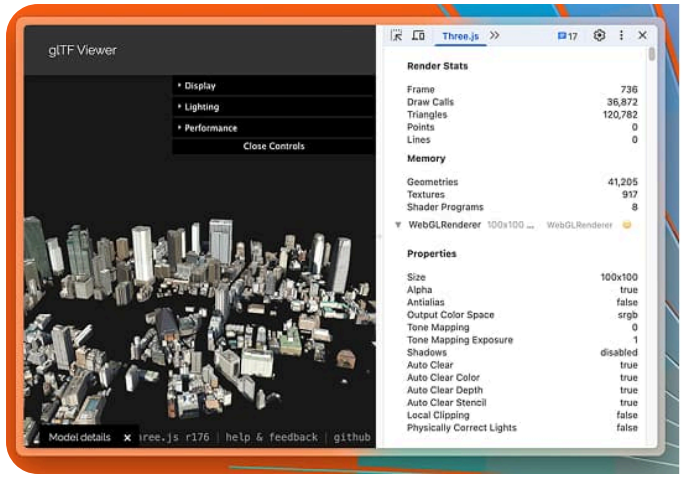
数万回の draw call を 1 回に集約、120fps 描画を実現

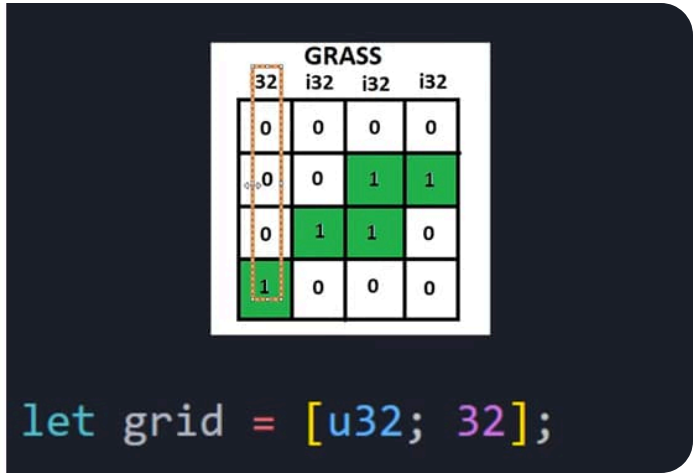
重すぎてブラウザで動作不可 （as-is）

数十〜数百 MB の 3D Model → 0fps
複数 Material → 数万回 の draw call (36,872)
複数 Texture → CPU/GPU 間転送過多
複雑な形状 → メモリ不足

ビット演算可能！1 回の draw call （to-be）

最適化された処理 → 120fps
統合 Material → 1 回の draw call
1 つの Texture → 最小限データ転送
ビット演算 → インスタンシング描画





参考: https://www.youtube.com/watch?v=qnGoGq7DWMc

2.2 軽い！軽量な地形データ

3D モデルから画像データへの変換により、部分配信とキャッシュ活用で高速起動を実現

重量な 3D モデルファイル（as-is）

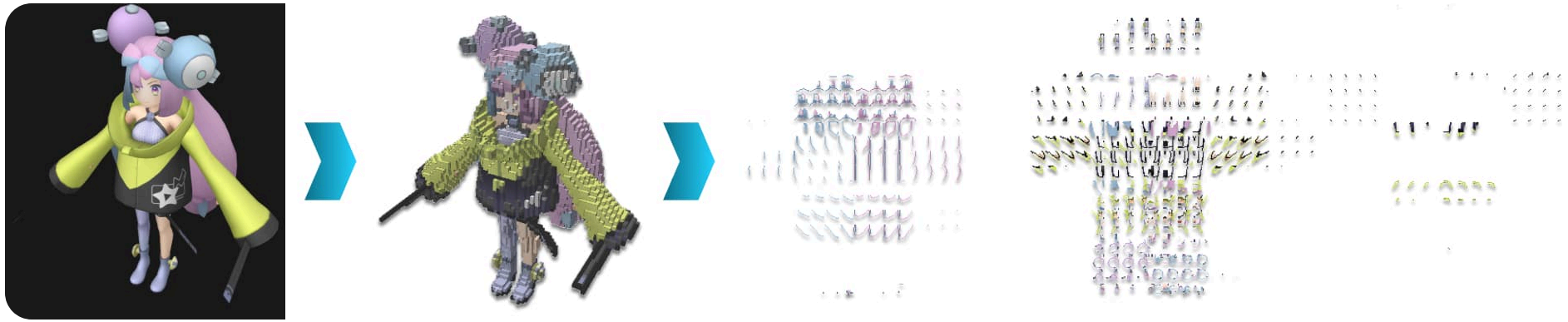
fbx / obj + mtl + 複数テクスチャ

- ファイルサイズ: 数十〜数百 MB
- ロード時間: 数分
- 複雑なアセット管理
- 荒い見た目になるまでデータ削減必要

画像データとして配信 （to-be）

PNG 画像 （256×256×256 voxel → 4096×4096 pixel）

- 1 区画 = 1 画像ファイル
- タイル分割し部分的データ取得
- ブラウザキャッシュ活用
- CDN 配信対応 (Cloudflare R2)



2.3 軽い！軽量な処理コード

1000 行以内の依存なしコードにより、保守コスト削減と高速起動を両立

一般的なゲームエンジン （as-is）

Unity/Three.js/Babylon.js
- 数百 MB ～数 GB のエンジン
- 複雑な依存関係
- 定期的なアップデート対応
- 高い学習コスト

我々のソリューション （to-be）

最小限コード （1000 行以内）
- WebGL API 以外依存なし
- アップデート対応不要
- 高速初期レンダリング
- 管理コスト最小化

保守性のメリット

- 処理が軽量 → パフォーマンス向上
- 管理コスト削減 → 開発効率向上
- 依存関係なし → 技術的負債回避
- 初期読込み最小化 → UX 向上

3.1 カラフル！航空写真の色データ

PLATEAU LOD2 の地域制約を航空写真との組み合わせで解決し、全国対応可能

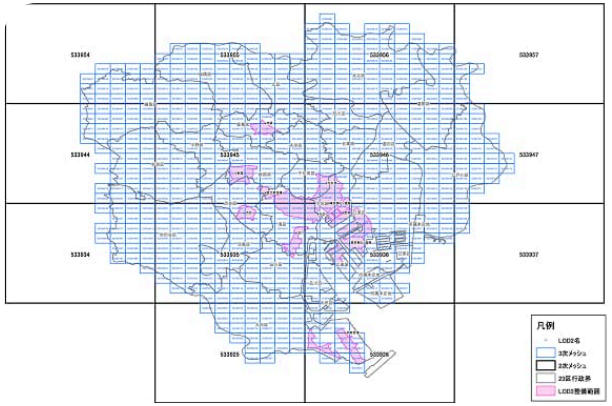
3D 都市モデル Open Data の制約 （as-is）

LOD1 （全国対応）
- 色データなし
- 単純な直方体

LOD2 （限定地域）
- 色データあり（一部の駅周辺等のみ）

我々のソリューション: 航空写真活用 （to-be）

国土地理院航空写真
- 全国対応の高解像度画像
- PLATEAU 建物形状 + 航空写真色
- 上空視点に最適化された色再現



3.2 カラフル！航空写真による綺麗な見た目

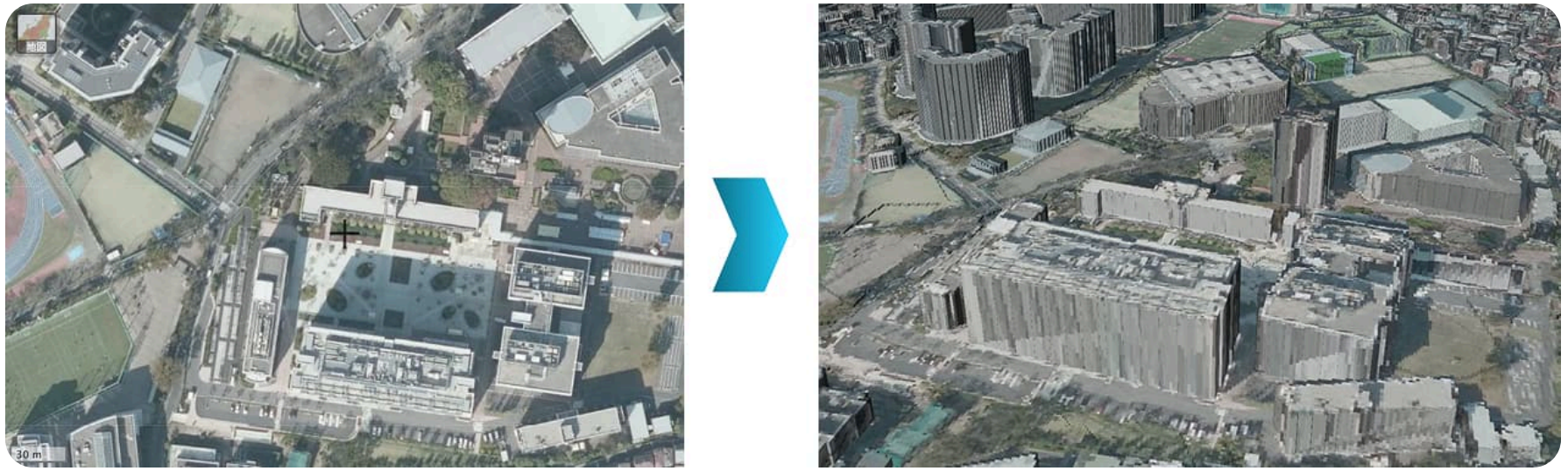
航空写真のノイズを 3D 形状情報により意味のある影に変換、計算コストなしで高品質描画

航空写真単体の問題 （as-is）

撮影時の建物影がノイズとして混入
→ なぜ影があるのか理解困難
→ 不調和な印象

3D 建物形状との組み合わせ効果 （to-be）

建物の高さ情報により影の理由が明確
→ 影の理由が納得できる
→ 計算コストなしで高品質



参考: https://maps.gsi.go.jp/#18/35.662652/139.634084/&base=ort&ls=ort&disp=1&vs=c1g1j0h0k0l0u0t0z0r0s0m0f0

3.3 カラフル！ボクセル形式はプレーヤーによるデータ操作が簡単

都市データを活用した多様な Web 体験を提供

利点:

- 整数座標 → 精密な配置
- O(1)衝突判定 → リアルタイム編集
- 統一データ構造 → シンプルな実装
- ネットワーク配信 → マルチプレイ対応

ユースケース:

- 都市計画シミュレーション
- 建築プロトタイピング
- 教育コンテンツ作成
- バーチャル観光体験

まとめ

Thank You！

実現した価値提案

- **簡単**: Web 標準技術で動作可能
  - ライブラリ化による開発障壁除去
  - 座標系統合による実装簡易化
  - シンプルデータ構造による軽量な計算
- **軽い**: Web 最適化された描画エンジン
  - 1 draw call による描画効率化
  - 画像配信による高速起動
  - 1000 行未満のコード
- **カラフル**: 全国対応の高品質表示
  - 航空写真による全域色付け
  - 影効果による調和向上
  - 256^3 種類ブロック

技術革新のポイント

- **データ変換**: 3D Model → Voxel → Image の変換
- **座標統合**: 複数座標系をピクセル基準に統一
- **描画最適化**: ビット演算とインスタンス描画
- **配信効率**: 分割による部分配信とキャッシュ活用