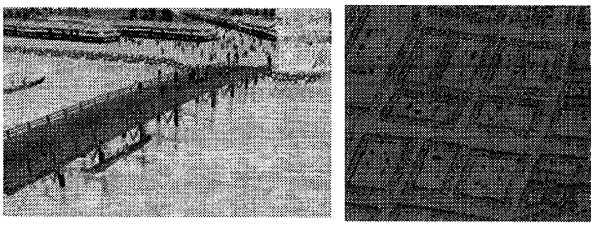
Occlusion-Cullingによる大規模な江戸の町並みの高速表示

安 ベヌア友章† 高橋 時市郎†
†東京電機大学大学院工学研究科

1. まえがき

我々は，江戸東京博物館と共同で，江戸時代の東京の町並みを3次元CGによって復元し，その町並みをインタラクティブにウォークスルーすることを目指した「江戸の町並み復元プロジェクト」を進めている．現在までに復元された町並みは，日本橋を中心とした2キロ四方に渡っている[1](図 1(a))．本プロジェクトでは，このような大規模の町並みをリアルタイムでレンダリングするためのレンダリングシステム，「江戸レンダラ」の開発も行ってきた[2]．以前制作した町並みのデータセットは，図 1(b)に示すように，区画の内部を持たず，裏長屋がぎっしり詰まった当時の区画よりも極端に粗いモデリングであった．しかし，最近，その区画の内部を埋める手法が提案され[3]，「江戸レンダラ」においてもより，建物の密集度の高い町並みのWalkthroughに対応する必要が生じた．

本稿では「江戸レンダラ」において，より広範囲に密に家屋が並ぶ町並みをレンダリングするために実装したOcclusion-Cullingとその評価について報告する．



(a) 日本橋 (b) 区画の俯瞰

図 1 復元された江戸の町並み

2. 江戸レンダラでのレンダリング処理

江戸レンダラは，広範囲にわたる江戸の町並みを効率的にレンダリングするために，複数の手法を組み合わせてレンダリングの処理を最適化する．Occlusion-Cullingを導入する以前のレンダラは，先ず，Octreeによって保存されている町並みのデータセットに対してFrustum-Cullingを行う．次に，Culling Testに合格したオブジェクト群に対してInstancing処理を施してから描画処理を行っていた．本稿では，Culling処理をさらに強化するためにFrustum-Cullingの直後にOcclusion-Cullingを追加する．すなわち，図 2に示す流れでレンダリング処理を行うように拡張した．

Occlusion-Cullingは描画するオブジェクトの内，レンダリングしても他のオブジェクトによって隠されるために，実際には画面に表示されないオブジェクトを事前に検知し，そのようなオブジェクトを早期にレンダリングのプロセスから除去する手法である．Occlusion-Cullingの実装には前処理を必要とする場合もあるが，江戸レンダラでは実行時の処理のみでOcclusion-Cullingを行う．

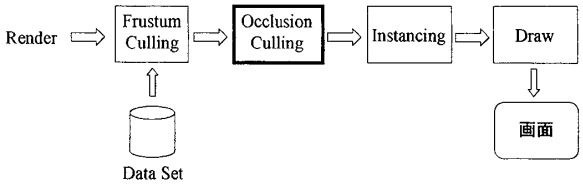


図 2 レンダリング処理の流れ

3. Occlusion-Cullingの実装

江戸レンダラにおけるOcclusion-Cullingのアルゴリズムは次の方針に従うように実装されている．

1. 画面上で，より大きく表示されるオブジェクトから先にOcclusion Testとレンダリングを行う．
2. 画面上で，互いに近い位置にあるオブジェクトはまとめてOcclusion Testをする．

この方針は大きいオブジェクトは他のオブジェクトを隠す可能性が大きいこと，互いに近いオブジェクトは同じオブジェクトによって隠される可能性が大きいこと，を根拠としている．

3.1. オブジェクトのソート

江戸レンダラのOcclusion-Cullingでは，オブジェクトをスクリーン上での大きさと位置でソートする為，スクリーンを，Quadtreeを用いて分割する(図 3)．各オブジェクトはOcclusion Testの前にBounding Volumeをスクリーン座標系へ透視投影変換し，Bounding Rectangle (BR)を得る．そして，そのBounding Rectangleを包含するノードの中で最も小さなノードに保存される．

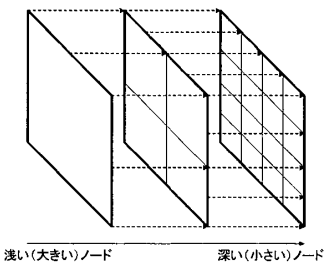


図 3 スクリーンのQuadtree化

このQuadtree内では，オブジェクトは大きければ浅いノードに，小さければ深いノードに保存される．それだけでなく，画面上での位置が近いオブジェクトとは，同じノードか親子関係にあるノードに保存される利点もある．

Hi-speed Rendering by Occlusion-Culling for Cyber Scapes of 'Edo'
†Tomoaki. Benoît Yasu, Tokiichiro Takahashi
Graduate School of Engineering, Tokyo Denki University

3.2. Occlusion Testとレンダリング

Occlusion Test にはグラフィックスプロセッサでサポートされている Occlusion Query を利用する. Occlusion Query を利用すると, 画面に対してポリゴンをレンダリングした際に, 実際にいくつのピクセルがレンダリングされるかを, 厳密に測定することが出来る. ただし, この測定には, 実際にレンダリングするのと同程度の時間が必要になる.

江戸レンダラでは Occlusion Test に Quadtree のノードをちょうど覆う形の Bounding Rectangle を用いる. 浅いノードから順にそのノードで最も画面に近いノードを隠すことの出来る深度の Bounding Rectangle で Occlusion Test を行う. もし, この Bounding Rectangle がレンダリングされなければ, このノードに保管されているオブジェクトのレンダリングを省略する. レンダリングされるのであれば, 保管されているオブジェクトを実際に画面へレンダリングする.

Occlusion Testでは, 親ノードの結果を再利用することで, 子ノードでのOcclusion Queryを省略できる場合がある. 子ノードのBounding Rectangleの占める領域は, 必ず親ノードの領域に収まる. そのため, 子ノードの画面上での深度が親ノードの深度よりも深い場合には, Occlusion Queryを使うまでもなく, 子ノードのオブジェクトが表示される可能性がないことが分かる. 図 4は, 以上の処理を示した擬似コードである.

```
(ルートノードから開始)
while true
  for 現在の深度のノードを一つずつ処理する
    bool isVisible;
    if このノードは親ノードよりも画面の奥にある
      isVisible = false;
    else
      int pixelCount = OcclusionQuery(ノードの BR)
      if (pixelCount > 0)
        isVisible = true;
      else
        isVisible = false;
    if isVisible == true
      Render(ノードのオブジェクト)
  探索するノードの深度を一つ深くする
```

図 4 Occlusion Test とレンダリングの擬似コード

4. 評価実験

Occlusion-Cullingの評価実験として, 表 1に示した測定環境において, 実際に江戸の町並みをWalkthroughし, Occlusion-Cullingの効果を測定した.

表 1 測定環境

OS	Windows Vista Home Premium
CPU	Intel CORE2 Extreme Q6850
Memory	2045M byte
Graphics Card	NVIDIA GeForce 8800 GTX

測定に用いる町並みは, Occlusion-Cullingの目的である建物が密集した区画から成る町並みを想定し, 図 5のように故意に区画の中に家屋を敷き詰めたデータセットを用いた.

Walkthroughでは, 大通りや路地をいくつか曲がりながら江戸の町を散策する. 図 6に示したグラフはWalkthrough開始から終了までの描画オブジェクト数の推移を記録したグラフである. OC前はFrustum-Culling直後のオブジェクト数, OC後はOcclusion-Cullingの結果, 実際に描画されたオブジェクト数である. Occlusion-Culling前には最大 3 万以上あるオブジェクトをOcclusion-Cullingを行うことで最大でも 5 千程度, 平均では 2 千程度にまで抑えることが出来ており, 高速化に大きく寄与している.

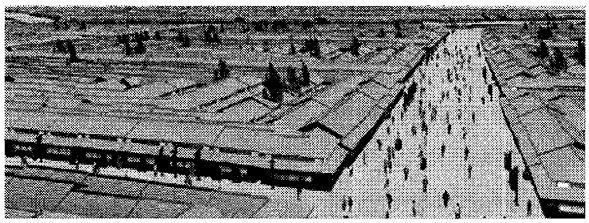


図 5 測定に用いた町並み

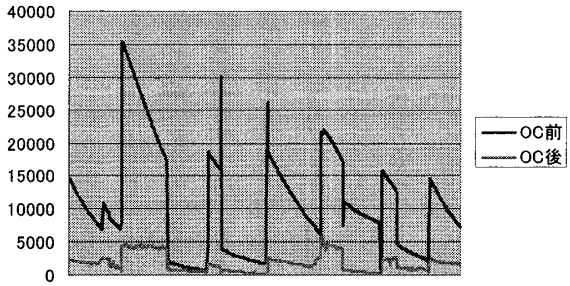


図 6 Walkthrough 時の描画オブジェクト数

表 2に, 同じ経路のWalkthroughで, Occlusion-Cullingの有無による平均FPSの違いを示す. Occlusion-Cullingを導入することによって, 平均 2.45 倍の高速化を実現した.

表 2 Occlusion-Culling(OC)の有無による平均 FPS

	OC 有	OC 無	比
FPS	4.04	1.65	2.45

5. むすび

大規模かつ建物の密集度の高い町並みを高速にレンダリングすることを目的として Occlusion-Culling を実装した. 評価実験の結果, 本手法によって, FPS にして約 2.5 倍の高速化に成功した.

参考文献

[1] 勝村他, "3DCG による歴史的町並み復元のための家屋生成手法,"画像電子学会誌, Vol.36, No.4, pp.382-389, 2007.

[2] 安他, "大規模な町並みのレンダリングに用いる実時間レンダラの開発," 電子情報通信学会技術報告, 画像工学研究会, IE2007-231, Vol.107, No.486, pp.73-78, 2008.

[3] 木村他, "奥行き伸縮モデルによる江戸後期の町並み復元," 日本バーチャルリアリティ学会第 13 回大会論文集, 2B3-6, pp.501-502, 2008.