2005－CG　118　(2)
2005／2／7

ポリゴン間のプライオリティ決定のための円環型データ構造

福重 真一[†]　　鈴木宏正[†]

与えられたポリゴンの集合からポリゴンの優先度の順に円環状に連なったデータ構造を構成し、任意視点から見たポリゴンの前後関係を効率的に求めるアルゴリズムを提案する.あらかじめポリゴンモデルの周囲に有限個の view cell を配し、各 cell から見たポリゴンの優先順位を単一の円環構造（Priority Circle と呼ぶ）として保持しておく．これにより、従来ならば BSP 法などのようにツリー構造上を再帰的に探索する必要があったポリゴンの前後判定を、円環列を一巡するだけの一回のパスで行うことが可能となり、レンダリング時における計算時間を格段に短縮することが出来る.

A Cyclic Data Structure for Determining the Face's Priority Order

Shinichi Fukushige,[†]　Hiromasa Suzuki[†]

This paper describes a depth sorting algorithm using a cyclic priority data structure. This priority cycle is constructed by face's obstruction relations seen from view cells distributed around 3D model. Compared with BSP tree method, priority cycle algorithm can search visibility order quickly without recursively traversing the tree structure.

1. はじめに

平面ポリゴンによって構成された 3 次元モデルを画面に表示するには可視面、不可視面の判定を行い不可視面を消去する、いわゆる陰面消去が必要である.

今日、特にリアルタイムレンダリングにおいてはグラフィックスハードウェアによる z バッファ法を用いた陰面消去処理が一般的ではあるが、半透明なポリゴンを多数含むシーンにはそのままでは適用できない．そこで、CPU によるポリゴンのソーティングを行い、視点から見て奥にあるポリゴンから順にアルファブレンディングを施しながら描画を行う必要がある.

街並や建築物などのようにシーンのほとんどが静的であることを仮定した場合には BSP（Binary Space Partitioning）法や Octree 法といった、ポリゴンを前処理としてツリー状に構造化することで高速かつ安定に前後判定を行う方法が多く提案されている．しかし、これらの方法では無意味なポリゴンの切断が多数発生し、それに伴いツリーが出力ポリゴン数に比例して巨大化してしまう、という問題がある．また、ポリゴンのソーティングにはツリーのすべてのノードにおける面の前後判定が不可欠なためツリーの大きさに比例した処理時間がかかってしまう.

† 東京大学 大学院 工学系研究科
Graduate School of Engineering,
The University of Tokyo

そこで本研究ではツリー構造ではなく、よりシンプルな円環状の構造を用いてポリゴン間の視点に対する前後関係を効率的に表現する方法を提案する．あらかじめ有限個の view cell を視点が通過する領域にポリゴンモデルを囲むように配し、それら各セル内部から見たときのポリゴンの優先順位を単一の円環構造として保持しておく．これによりモデルの描画時において、ポリゴン間の前後判定を行わずに、移動する視点位置に対するポリゴンの前後関係を円環列を一巡するだけの一回のパスで求めることが可能となる.

第２節では関連研究について述べ、第３節で本手法を提案するにあたっての準備事項について述る．第４節で本手法の概略とその特徴を述べた後、第５節で具体的なアルゴリズムの解説を行い、第６節で本手法と BSP 法とを比較した実験結果を紹介する．最後に第７節で今後解決すべき課題について述べる.

2. 関連研究

静的なポリゴンモデルに対して視点位置のみが移動する場合、前処理としてポリゴンデータを構

造化しておくことで視点に対する面の前後関係を効率的に求めることが出来る．特にツリー状に構造化する方法はそのシンプルな形態と再帰的なアルゴリズムを適用可能なことから多くの手法が提案されている．

BSP 法は Schmacker らが提案した手法を基に Fuch らが更に発展させたものであり、適当に面を選びながらその面によって空間を階層的に順次 2 分割していき、すべての面をツリー構造として保持する、というものである [1) 7) 2)]．面の前後判定はこの BSP ツリーをルートとなる面からたどっていくことで高速に行うことが出来る．出力ポリゴン数 N に対して、ソーティングに要する時間は $O(N)$ である．

しかし、この BSP ツリーを構築していくには、新しいポリゴンをツリーに挿入する度にその面を含む平面で他のポリゴンを順次分割する必要がある為、無意味なポリゴンの切断が多数発生してしまう（理論上は入力ポリゴン数 N に対して最悪の場合出力ポリゴン数が $O(N^3)$ のオーダーで増加する）[7)]、という問題がある．ツリーの各ノードではそのノード面の表裏判定が行われるが、ツリーの巨大化がそのままフレーム毎の判定回数の増大につながり、結果として処理時間が増加する．また、BSP 法ではツリーの形状がアルゴリズムのパフォーマンスを大きく左右する為、最適なツリー構造を構築するためにノード面を最適に選択してゆかなければならないが、このような面を自動的に選択することは極めて難しい．

その為、今日まで BSP 法のさまざまな改良が試みられてきた．Chen らはポリゴンの裏面消去を有効にすることで、面の切断回数および前後判定が必要なノード数を減らしソーティングを高速化させる、という Feudal Priority アルゴリズムを提案した [3)]．しかし、ツリーを再帰的に探索しながらポリゴンの前後判定を行うことに変わりはない．

また、James らは BSP ツリーの葉に対してポリゴン間の優先順位の情報を効率的に与えておくことでモデルの描画時にソーティングに要する時間を入力ポリゴン数 n 出力ポリゴン数 N に対して $O(\log_{4/3} n)$ にまで減少させた Priori Face Determination Tree アルゴリズムを提案した [6)]．しかし、同時に必要なメモリ容量が $O(N1.5^{\log_{4/3} n-1})$ のオーダーで増大してしまうため、大規模なモデルには適用し難い、という問題がある．

また、Morer らや Dur らはモデルの形状を建築物や閉曲面のような特定のものに限定することで BSP 法をより高速化するアルゴリズムを提案している [4) 5) 8)]．

3. 準備

3.1 構成面

モデルを構成するポリゴンは平面を複数の線分によって切り取った単連結な多角形とし、凸多角形、凹多角形のいずれでもよいものとする．またポリゴンには裏面、表面を設定し、裏面は不可視とする．

裏面を可視にする必要がある場合は２つの同形のポリゴンの裏面どうしを張り合わせたものとして扱う．

3.2 ポリゴン間の遮蔽関係

3 次元空間上に互いに干渉していないポリゴン A、B および視点 v が与えられているとする．v より発せられ A、B を共に表側から貫通する光線 $\mathbf{r}$（v を起点とする半直線）が存在し、かつ v に対して常に $\mathbf{r}\cap$A の点が $\mathbf{r}\cap$B の点よりも近くにあるとき、A は B を遮蔽しているとし、その関係を $A \prec_v B$ と表す．また、A，B 間に遮蔽関係が無い場合は $A =_v B$ と表す．

3.3 Priority List

視点 v に対し、ポリゴンを遮蔽関係に従って前から後ろの順に並べたリストを Priority List と呼び P^v と表す．すなわち、視点 v およびリスト内の i 番目の要素 p_i、および j 番目の要素 p_j に対して「$p_i \prec_v p_j$ ならば $i < j$」という関係が常に成り立つものを Priority List とする．ただし、一つの視点に対して Priority List は必ずしも一通りではない．

3.4 ポリゴン間の優先順位関係

視点位置に依存しないポリゴン間の優先順位関係について以下のように定義する．

ポリゴン p の乗る平面によって空間を二分割し、ポリゴンの表側半空間を $F_s(p)$、裏側半空間を $B_s(p)$ とする．

ここで

$$p_1 \cap F_s(p_2) \neq \phi \text{ かつ } p_2 \cap B_s(p_1) \neq \phi \qquad (1)$$

であるとき、p_1 と p_2 のいずれも裏面からは不可視である、という性質から、図１に示すようにそれらの遮蔽関係はあらゆる視点位置に対して $p_1 \prec_v p_2$ または $p_1 =_v p_2$ のいずれかとなり、視点位置に関係なく p_1 は p_2 よりも常に優先順位が高い、とすることが出来る．

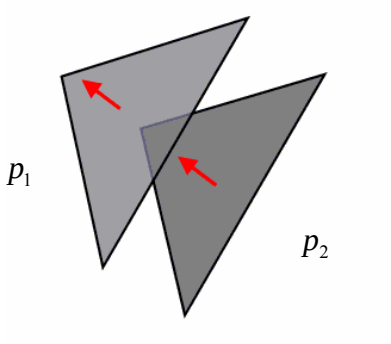

図 1．視点に依存しない優先順位関係

$p_1 \cap F_s(p_2) = \phi$ かつ $p_2 \cap F_s(p_1) = \phi$
または

$p_1 \cap B_s(p_2) = \phi$ かつ $p_2 \cap B_s(p_1) = \phi$ (2)

であるとき、p_1 と p_2 には遮蔽関係が無く、常に $p_1 =_v p_2$ となるので、その優先関係は同等である．

$p_1 \cap F_s(p_2) \neq \phi$ かつ $p_1 \cap B_s(p_2) \neq \phi$ かつ
$p_2 \cap F_s(p_1) \neq \phi$ かつ $p_2 \cap B_s(p_1) \neq \phi$ (3)

であるとき、p_1 と p_2 の優先関係は巡廻し、優先関係が定まらない．

そこで、ポリゴン p_2 の乗る平面と p_1 の交線によって p_1 を p_1' と p_1'' とに切断する、という処置を施しておく．これにより、2つのポリゴン p_1、p_2 間の優先関係の巡廻が解消し、新たに p_1' と p_2、p_1'' と p_2 とに固定の優先関係を設定することが可能となる．ただし p_1' と p_1'' とは同一平面上にあるので優先関係はない．

3.5 view cell の設定

あらかじめ視点が通る領域を細かくグリッド状に分割し、これらを view cell と呼ぶことにする．ここで、各 view cell をその中心位置 v_k で代表させ、cell 内部に視点 v があるとき、v から見た Priority List P^v は cell の中心位置 v_k を視点として見た時の P^{v_k} によって表すものとする．

すなわち、連続的に移動する視点位置を有限個の v_k で近似することになるので、表された P^{v_k} は厳密な前後関係ではないが、本研究で提案するアルゴリズムの適用例としては回転やズーミングによるモデルの閲覧を想定しており、表示に十分なだけの密度の cell を配置することで遜色のない表示結果を得るものと考える．

本研究では、回転とズーミングによるモデルの閲覧に適した視点空間として極座標空間を用い、view cell の中心を以下のように配する．（原点をモデルの中心とする）

$$\begin{aligned} x &= r_n^{\,2} \sin\theta_m \cos\varphi_l \\ y &= r_n^{\,2} \sin\theta_m \sin\varphi_l \\ z &= r_n^{\,2} \cos\theta_m \end{aligned} \qquad (4)$$

ただし $r_n = r_0 + n\,d_r$, $\quad n = 0,1,\ldots,N$
$\theta_m = \theta_0 + m\,d_\theta$, $\quad m = 0,1,\ldots,M$
$\varphi_l = \varphi_0 + l\,d_\varphi$, $\quad l = 0,1,\ldots,L$

view cell は r,θ,φ 空間をレギュラーなグリッド状に分割したものと言える．それぞれの軸方向における cell の幅 d_r, d_θ, d_φ および cell 数はモデルの複雑さやポリゴン数に応じて定められる．

また、r_n が二乗されているのは、モデルに近い位置ほど視点の変化に対するポリゴンの前後関係の変化が大きく、view cell の密度を大きくする必要がある為である．

実際にモデルの周囲に約 150 万個の視点を配した例を図 2 に示す．

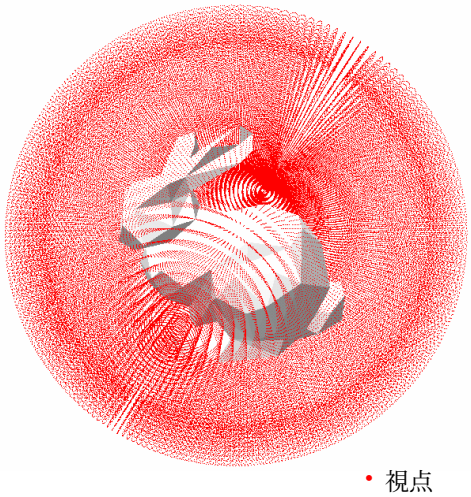

図 2．モデルとその周囲に配置された視点

4. Priority Circle

4.1 円環列

要素 $A_1, A_2, \ldots, A_N$ を円環状に並べたものを円環列と呼び $\langle A_1, A_2, \ldots, A_N \rangle$ と表す．ただし

$\langle A_1, A_2, \ldots, A_N \rangle$
$= \langle A_2, A_3, \ldots, A_1 \rangle$
……
$= \langle A_N, A_1, \ldots, A_{N-2}, A_{N-1} \rangle$ である．

また $\langle A_1, A_2, \ldots, A_N \rangle$ の並びを順方向としたとき、$\langle A_N, A_{N-1}, \ldots, A_1 \rangle$ の並びを逆方向とする．

本手法の特長は、与えられたポリゴンの集合に対して有限回の面切断を施すことによって面の優

先度の順に円環状に連なったデータ構造を構成し（これを Priority Circle と呼ぶ）、任意視点からの面の前後関係を効率的に求める、というものである．各 view cell に対して、Priority Circle 上の一つのポリゴンが対応し、そのポリゴンを先頭として円環列から順方向にポリゴンを取り出したものが、その cell 内の視点における Priority List となるように Priority Circle を構築する．

4.2 遮蔽関係が巡回する場合

視点 v が与えられたとき、v から見た遮蔽関係が図３のように巡廻し、P^v が存在しない場合がある．このとき、巡廻するポリゴンから適当な 2 面を選択し切断することで巡廻状態が解消し Priority List が求まる．

図３．巡廻する遮蔽関係

4.3 Priority Circle

視点 v に対して任意のポリゴンの組 p_i、p_j は

$p_i \prec_{v_p} p_j$　(5)　または

$p_j \prec_{v_p} p_i$　(6)　または

$p_i =_{v_p} p_j$　(7)

のいずれかの関係を持つ．

このうち(5)と(6)場合について、p_i、p_j に起点 s^v を加えた３つの要素から成る円環列を以下のように定める．

$p_i \prec_v p_j$ ならば、$\langle s^v, p_i, p_j \rangle$　(8)

$p_j \prec_v p_i$ ならば、$\langle s^v, p_j, p_i \rangle$　(9)

ここで、起点とは図４に示すように円環列から Priority List を求める時にリストの先頭を指し示す為に仮に挿入されるもので、円環列 $\langle s^v, p_i, p_j \rangle$ は s^v を先頭として要素を順方向に抽出したリスト $\{ s^v, p_i, p_j \}$ が v から見た Priority List であることを表している．

この３つの要素から成る円環列は視点 v から見て遮蔽関係があるポリゴンの組すべてについて定められ、単位円環列と呼ぶ．それら単位円環列の集合を Φ^v とする．

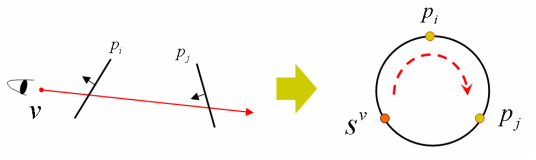

図４．単位円環列

Φ^v をすべて内部に含むように全ポリゴンおよび s^v から成る円環列 $C = \langle s^v, p_1, \cdots, p_N \rangle$ を構成したとき、起点 s^v を先頭として円環列 C から順方向にポリゴンを取り出したリスト $P^v = \{ s^v, p_1, \cdots, p_N \}$ は視点 v に対する Priority List となる．4.2 の処理により v における Priority List が必ず存在することになるので、Φ^v をすべて含むような C を構成することが可能であることが保証される．

次に、別の視点 $\hat{v}$ から見たときのポリゴンの遮蔽関係からも単位円環列の集合 $\Phi^{\hat{v}}$ が求まる．

このとき、Φ^v と $\Phi^{\hat{v}}$ とを同時に満たすように C の要素の並べ替えを行い、新しく円環列 C' を構成することが出来る．

すなわち起点 s^v を先頭として円環列 C' から順方向にポリゴンを取り出したリストが視点 v に対する Priority List となり、起点 $s^{\hat{v}}$ を先頭として円環列 C' から順方向にポリゴンを取り出したリストが視点 $\hat{v}$ に対する Priority List となるように C' を作ることが出来る．

以上の性質は、次のように導かれる．

Φ^v、$\Phi^{\hat{v}}$ の各要素となる単位円環列のうち、起点 s^v、$s^{\hat{v}}$ を除いた２つのポリゴン p_k、p_l が共通する単位円環列どうしに着目する．

3.4 の処理により p_k と p_l の間には視点位置に関係のない固定された優先順位関係が存在している．仮に p_l よりも p_k の優先順位が高いとすると、視点 v および $\hat{v}$ に対して

$p_k \prec_v p_l$ または $p_k =_v p_l$　(10)

$p_k \prec_{\hat{v}} p_l$ または $p_k =_{\hat{v}} p_l$　(11)

の関係が常に成り立つ．このとき、それぞれの単位円環列が $\langle s^v, p_k, p_l \rangle$、$\langle s^{\hat{v}}, p_k, p_l \rangle$ となり、互いに s^v と $s^{\hat{v}}$ を入れ替えただけの同一構造のものであることが分かる．Φ^v と $\Phi^{\hat{v}}$ のすべての要素が単一の円環列 C' に同時に含まれるには、Φ^v、$\Phi^{\hat{v}}$ に共通する要素間に同一の構造があることが保証されていればよい．

この性質を利用することによって Φ^v と $\Phi^{\hat{v}}$ とが互いに干渉することなく単一の円環列上に共に

含まれるようにすべてのポリゴンから成る円環列 C' を構成することが可能であることを示すことが出来る.

これを更に、モデルの周囲に配された有限個の視点すべてに適用する。(ただし、3つ以上のすべての視点における単位円環列の集合を同時にすべて含むような単一の円環列を構成することが可能である、ということは現時点では完全に証明するには至っていない.)

このような円環列を Priority Circle とする.

すべての視点位置から見たポリゴンの遮蔽関係は一つの Priority Circle として保持される. 各視点には Priority Circle の一つの起点が対応し、その起点を先頭として Priority Circle から順方向にポリゴンを取り出したリストがその視点における Priority List である.

よって、与えられた視点位置に対応する Priority Circle 上の起点が求まれば、ポリゴン間の前後判定を必要とせずに、その視点位置におけるポリゴンの順位がそのまま Priority Circle の並び順として即座に求まる.

5. Priority Circle の構築

Priority Circle の構築方法の概略について述べる.

与えられたポリゴンの集合 M からポリゴン p を一つずつ取り出し、円環列 C へ挿入していく.

このとき、視点 v_k をすべてチェックし、ポリゴンどうしの遮蔽関係に巡廻が発生する場合は p の切断を行い、巡廻を解消しておく.

その後、各 v_k から見た遮蔽関係を満たすように p を挿入できる C 上の範囲を求める。すなわち Φ^{v_k} のうち p を含む単位円環列を参照しながら、C への挿入位置を絞り込んでいく。ただし、既に C に含まれている単位円環列と新しく挿入する p とで円環列の並びに矛盾が発生する場合があり、このとき、矛盾に関わる C の要素を M に戻す、という処理を行う。また、各 v_k は常に対応する C 上の起点へのポインタを持ち、C が変わる度にポインタを更新する.

最終的に M 内のポリゴンがすべて C に挿入された時点で処理を終わる。

6. 実験

本手法をポリゴン数が1万面以上のモデルに適用し、移動する視点に対する優先順位判定の速度、および描画精度を、現在、最も広く用いられているデプスソーティングアルゴリズムである BSP 法と比較した.

実験環境は以下の通りである.

CPU	AMD Celeron 1GHz
CPU キャッシュ	1024KB
メインメモリ	256MB

表1. 実験環境

図5はポンプの一部品であり 14462 面のポリゴンから成る、図6の車モデルは車体および車体とは切り離された4つのタイヤから成り、ポリゴン数は 10093 面である.

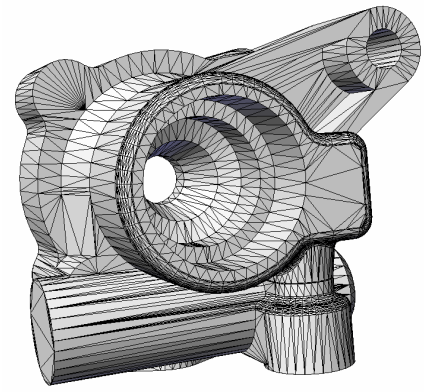

図5. PUMP

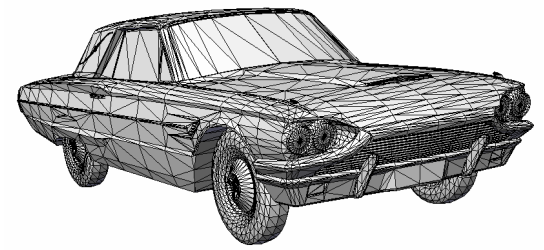

図6. CAR

これら2つのモデルに対して、本手法および BSP 法を適用したときの、切断面数、view cell の数(本手法のみ)、必要となるメモリ容量、そしてソーティングに要した処理時間を表2, 3に示す.

なお、ソーティングに要した時間は視点位置を遷移させながら 10 万フレームを描画し、ソーティングと描画にかかった時間のうち、描画を除いた1フレームあたりの平均時間から算出している. 描画時間そのものはハードウェアの性能に大きく依存し、かつ手法による差が出ないために今回の実験では考察から外している.

	本手法	BSP 法
ポリゴン数	14462 面	
切断面数	211 面	1793 面
view cell 数	1500000	
メモリコスト	7MB	1MB
ソート時間	0.043msec	6.5msec

表2．PUMP

	本手法	BSP 法
ポリゴン数	10093 面	
切断面数	128 面	820 面
view cell 数	1500000	
メモリコスト	7MB	1MB
ソート時間	0.029msec	4.1msec

表3．CAR

本手法は視点が内部にある view cell さえ特定できれば、その view cell が指し示す起点を先頭に円環列から順にポリゴンを取り出したものがそのままポリゴンの優先順位となるため、描画時間を除いたポリゴンのソートにのみ要する時間を比較すれば、視点位置が変わるたびにツリー上を再帰的に探索する必要がある BSP 法よりも 100 倍以上高速であることが分かる.

ただし、すべての view cell に Priority Circle 上の起点を指し示すポインタを持たせる必要がある為、必要となるメモリ容量についてはツリー構造のみを保持すればよい BSP より数倍の容量が必要であった.

モデルのポリゴン数が $2^{32} = 4294967296$ 面以内であればポインタを 4byte で表現し、それにビューセルの数をかけ合わせたものと Priority Circle の構造を保持するための配列分（16byte×ポリゴン数）を合わせたものが必要なメモリ容量と推定できる.

14712 ポリゴンから成る PUMP であれば

4byte × 1500000 viewcells

+ 16byte × 14462polygons = 6.24MB

がメモリコストとなる、と推定されることから、ほぼ計算通りの結果となっていることが分かる.

また、本手法では本来連続的に移動する視点を有限個の視点位置によって近似的に表現しているが、十分な密度をもって view cell を配することでほぼ正確な表示結果を得た.

7. 展望

本研究では極座標空間においてレギュラーな view cell をモデルの周囲に配置することで、視点から見たポリゴン間の優先順位を、その視点を内部に含む cell から見た Priority List とし、単一の円環構造を用いて高速に求められることを示した.

しかし、回転やズーミングによるモデルの単純な閲覧といった用途のみならず、例えば街並みや建築物内部のウォークスルーのような応用を考えたとき、視点が通過する領域を特定して効率よく view cell を配する必要があり、また view cell の集合に対して構造化を行うなどして、視点がどの view cell に含まれるかを求める処理がなるべく複雑にならないようにしなければならない。またモデル内のポリゴンの密度分布やモデルの複雑度に応じた view cell の配置を行い、また、より効率的に view cell を探索するアルゴリズムを開発する必要がある.

参考文献

1) I. E. Sutherland, R. F. Sproull, R. A. Schumacker, “A characterization of ten hidden-surface algorithms,” ACM Computing Surveys 6(1), 1974, pp. 1-55.
2) H. Fuchs, Z. M. Kedem. “Predeterminig Visibility Priority in 3D Scenes,” SIGGRAPH '79 Proc, Volume 13, pages 175-181, July 1979.
3) H. Chen, W. Wang, “The Feudal Priority Algorithm on Hidden-Surface Removal,” ACM SIGGRAPH 96, August 1996, pp.55-64.
4) A. Dür, S. Leimgruber, “A practical list-priority algorithm for 3D polygons,” Journal of Graphics Tools, 8(4):25-36, 2003.
5) P. K. Agarwal, T. M. Murali, J. S. Vitter, “Practical Techniques for Constructing Binary Space Partitions for Orthogonal Rectangles,” Symposium on Computational Geometry,1997.
6) A. James, A. M. Day, "The priority face determination tree for hidden surface removal," Computer Graphics Forum, 17(1):55-71, 1998.
7) H. Fuchs, Z. M. Kedem, and B. F. Naylor. “On visible surface generation by a priori tree structures,” SIGGRAPH '80 Proceedings, vol. 14, pages 124--133, July 1980.
8) P. Morer, A. M. Garcia-Alonso, and J. Flaquer. “Optimization of a Priority List Algorithm for 3-D Rendering of Buildings,” Computer Graphics Forum, vol. 14, number 4, pages 217-227, 1995.