G-107

空間データ構造再構築を用いた 透過性織布のリアルタイムアニメーション

Real-time animation of transparent woven cloth using spatial data structure reconstruction

松本達也†　飛谷謙介†　岡本拓也†　長田典子†
Tatsuya Matsumoto　Kensuke Tobitani　Takuya Okamoto　Noriko Nagata

1. はじめに

近年，計算機の性能向上に伴いコンピュータグラフィックス技術は著しい発展を遂げ，一般社会にも CG が深く浸透するようになった．その結果，より高品質かつ高精細な質感をインタラクティブな環境で表現することが求められるようになった．質感の表現性を高めるには，物体の正確な反射・透過特性を考慮し，その特性に適したモデル式やレンダリングアルゴリズムを用いる必要がある．またレンダリングを高速化することで質感表現をリアルタイムに行うことが可能になる．

CG のシーン中で，頻繁に用いられる素材に布素材がある．布素材は固有の反射・透過特性を持ち，なおかつ，変形することによって光の経路が複雑に変わり，視覚的に大きな変化をもたらす．これまでに我々は織布の高品質な CG 表現のため，双方向反射分布関数(BRDF: Bidirectional Reflectance Distribution Function)や双方向透過分布関数(BTDF: Bidirectional Transmittance Distribution Function)の計測を行い，物理ベースモデルである Woven cloth microfacet BSDF モデルを提案した[1]．また，計測した BTDF データからラスタライズ法を用いることで，リアルタイムに揺れ動く織布の質感を表現することのできるレンダリングアルゴリズムを提案した[2]．しかし，この手法では実際に光の挙動をシミュレートせず，物体の前後関係から描写を行うため，正確な透過特性を表現することが困難であった．

そこで本研究では， Woven cloth microfacet BSDF モデルをレイトレーシング法に適用することで，光の挙動をシミュレートし織布固有の透過特性を正確に表現する．また，アニメーション生成のリアルタイム化を図るため，GPU による並列処理を行い，空間データ構造として kd-tree[3]を用いることで，レイトレーシングを高速化する．

2. 先行研究

質感表現を高めるには，物体個々の特性に適した BRDF や BTDF の近似モデルを適用し，その目的にあったレンダリングアルゴリズムを導入する必要がある．織布の質感をリアルタイムに表現した研究例として，Sattler ら[4]や Claustres らの手法[5]がある．これらの織布の研究ではラスタライズ法を用いることでリアルタイムレンダリングを実現している．しかし質感は反射特性のみを考慮し，固有の透過特性は扱っていない．一方で，透過特性を持つ物体のリアルタイムレンダリングを試みた研究に葉の透過特性に着目した Ralf らの手法[6]や，すりガラスのような荒い表面を持つ物体の反射・透過特性を取り扱った Rousiers らの手法[7]などがある．しかし，いずれの研究においても表現対象として織布を扱ったものはない．

†関西学院大学　大学院理工学研究科，
Graduate School of Science and Technology, Kwansei Gakuin University

そこで，我々はこれまでに，織布特有の反射・透過特性に着目した Woven cloth microfacet BSDF モデルを提案した．また BTDF の計測データから，ラスタライズ法を用いることでリアルタイムに織布の質感を表現する手法を提案した．しかし，ラスタライズ法は実際にレイの挙動をシミュレートせず，三次元中のプリミティブを二次元画像に変換することで画像を生成するため，複雑な反射や透過を表現することが困難であった．

光の経路を追跡し画像を生成することで，複雑な反射・透過を表現できるレンダリングアルゴリズムにレイトレーシング法がある．レイトレーシングの計算負荷は非常に高いが，近年では様々な高速化手法が提案され，レイトレーシングをリアルタイム化する試みは数多くなされている．高速化手法は主にハードウェアによる手法[8]と，空間データ構造を用いる手法がある．空間データ構造による高速化手法は数多く存在するが，構築速度やトラバーサル効率の観点から Bounding Volume Hierarchy (BVH) [9]や kd-tree が一般的に用いられている．これらのデータ構造は構築速度・トラバーサル効率の更なる改善が行われており，Wald らは kd-tree を $O(N \log N)$ で構築を行うアルゴリズムを提案した[10]．Popov らや Shevstov らはマルチコア CPU により kd-tree の構築を並列的に行うアルゴリズムを提案し[11] [12]，Zhou らは GPU により大規模な並列処理を行うことで，リアルタイムに kd-tree を構築することに成功した[13]．また，Zhou らによって提案された手法はフォトンマッピング法において再近傍フォトンを求める際に必要となる k-nearest neighbor (KNN) search への適用も可能である．

以上を踏まえて本研究では，光の経路から画像を生成するレイトレーシング法を採用し Woven cloth microfacet BSDF モデルを適用することで織布の透過特性を正確に表現する．その際， GPU による並列処理と空間データ構造による走査の改善を行うことで，リアルタイムに織布のアニメーションを生成する．

3. 織布の質感表現

本章では，織布の透過特性を表現するために用いる Woven cloth microfacet BSDF モデルについて述べる．

3.1 Woven cloth microfacet BSDF モデル

我々これまでに，計測データと Phong モデル，Ward モデル，Ashikhmin-Shirly モデル，Cook-Torrance モデルについて BTDF 用のパラメータに変更したものに加え，Woven Cloth HGF モデル，および Microfacet BTDF モデルである GGX モデルの計 6 つの近似モデルとの比較を行った[1]．その結果，織布の BTDF として GGX モデルのフィッティング精度が最も高いことを確認した． そこで GGX モデル

を基に，織布の質感表現のための近似モデルである Woven cloth microfacet BSDF モデルを提案した．

Woven cloth microfacet BSDF モデル式は式(1)のように与えられる．

$$I_{out} = k_a I_{direct} + 1/2(k_t I_t f_{GGX} \exp(-\tau)\rho) + 1/2(k_t I_p f_{GGX} \exp(-\tau)\rho) \quad (1)$$

ただし，L_{out} は始点方向へ透過する放射輝度，k_a は環境係数，L_{direct} は糸にあたらず透過する放射輝度（直接透過光），L_t は糸の内部で屈折し透過する放射輝度（拡散透過光）L_p は糸の内部で屈折せず透過する放射輝度（指向性透過光），k_t は直接透過係数，$\exp(-\tau)$ は吸収係数，ρ は糸の密度である．

また n_1，n_2 をそれぞれの透過する媒質の屈折率，θ_{lh} を光源ベクトルとハーフベクトルのなす角，θ_{vh} を視点ベクトルとハーフベクトルのなす角，θ_{nl} が法線ベクトルと光源ベクトルのなす角度，θ_{vn} が視点ベクトルと法線ベクトルのなす角度としたとき，f_{GGX} は式(2)のように与えられる．

$$f_{GGX} = \frac{\cos\theta_{lh}\cos\theta_{vh}}{\cos\theta_{nl}\cos\theta_{vn}} \frac{{n_2}^2(1-F)GD_{GGX}}{(n_1\cos\theta_{lh} + n_2\cos\theta_{vh})^2} \quad (2)$$

幾何減衰率 G は反射光が物体表面で減衰する現象であるセルフシャドウリングやセルフマスキングの割合を表し，θ_{nh} を法線ベクトルと視点ベクトルのなす角，θ_{nv} を法線ベクトルとハーフベクトルのなす角としたとき式(3)のように表現される．

$$G = \min(1, \frac{2\cos\theta_{nh}\cos\theta_{nv}}{\cos\theta_{vh}}, \frac{2\cos\theta_{nh}\cos\theta_{nl}}{\cos\theta_{vh}}) \quad (3)$$

フレネル反射 F を表す際は Schlick によって導かれた近似式(4)を用いる．

$$F(\theta) \approx F_0 + (1-F_0)(1-\cos\theta)^5 \quad (4)$$

分布関数 D_{GGX} は物体表面の微小面が任意の正面を向いている割合を荒合し，微小面の荒さを表すパラメータを r としたとき，式(5)のように表現される．

$$D_{GGX} = \frac{r^2}{\pi\cos\theta_{lh}^4(r^2 + \tan\theta_{lh}^4)^2} \quad (5)$$

図 1 に Woven cloth microfacet BSDF モデルによるオフラインでのレンダリング結果を示す．カーテンのドレープによって引き起こされる，織布特有の透過特性を表現できていることが判断でき，Woven cloth microfacet BSDF モデルの有用性を確認することができる．

図 1　Woven Cloth Microfacet BSDF モデルを適用した織布のシーン

4. レイトレーシングのリアルタイム化

本章では，一般的なレイトレーシング法を紹介した後，リアルタイムレンダリングを実現するために，今回実装するハードウェアによる手法と空間データ構造を用いた手法について述べる．

4.1 レイトレーシング

レイトレーシングでは，画素からスクリーンに向かうレイを生成し，レイとプリミティブの交差判定を行う．鏡面物体や透過物体と交差が起こった際は新たに反射光・透過光の二つのレイを二次レイとして生成する．この処理は二次レイが光源に届くか，背景に届くか，もしくは非反射・透過物体に到達するまで繰り返し行われ，出力する画素の色が決定される．この処理をすべての画素に対して行うことで最終的に画像を生成する．レイトレーシングはオブジェクトの探索を行う走査に処理の大半を占めるため，走査の高速化がレイトレーシング全体の高速化となる．

4.2 GPU ベースのレイトレーシングアルゴリズム

代表的なハードウェアによる高速化手法に並列コンピューティングがあり，GPU は入力された異なるデータそれぞれに対し同一の計算処理を行うことが可能である（ストリームプロセッシング）．したがって，大量の画素それぞれに同様の計算処理を行うレイトレーシングはストリーミングプロセッシングとの相性が良いとされている．以下に GPU レイトレーシングのアルゴリズムについて述べる．

1. 画素数分の視線レイを生成し，GPU メモリ上に残るデータを破棄する．
2. 与えられたレイからシーンを走査し交差判定を行う．その際衝突したプリミティブの情報を視線追跡反射・透過回数毎に保持する．
3. 反射・透過回数がある一定以下なら 4 へ．そうでなければ 5 へ向かう．
4. 2 の交差点を始点とする透過レイ・反射レイをそれぞれ画素数分生成．2 に戻る．
5. 2 のプリミティブの情報を元に，画素数分のシェーディングを反射・透過回数ごとに行う．
6. 5 で求められたシェーディング結果を反射率・透過率を元に足し合わせ，最終的な出力とする．

図 2　ラスタライズ法による透過感表現

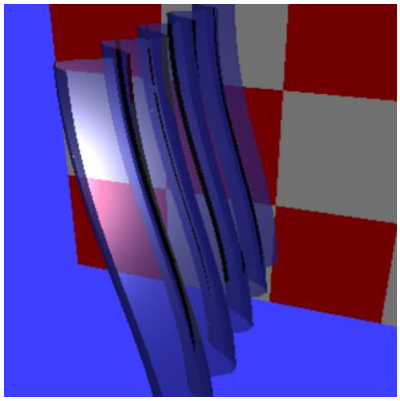


図 3　レイトレーシング法による透過感表現

シェーディングは反射・透過の二次レイが生成されるたびに行い，視線ベクトルと交差する物体の色は RGB 値を持つ光源からの光を，その物体の反射率などの性質に合わせ減衰し求める．光の反射は BRDF の近似モデルを，透過率は BTDF の近似モデルを用いて求め，各画素の色とする．織布固有の透過感は三章で述べた Woven cloth micro facet BSDF モデルを適用することで表現する．

4.2.1 実行結果

図 2 にラスタライズ法により織布の透過特性を表現したレンダリング結果を，図 3 にレイトレーシング法に woven cloth microfacet BSDF モデルを適用した結果を示す．レイトレーシング法を用いることで光の挙動を実際にシミュレートし高品質な画像を出力することに成功した．また，ラスタライズ法では表現することのできなかった，カーテンのドレープが光を遮ることによって生じるセルフシャドウを表現できていることが確認できる．

4.3 空間データ構造による走査の改善

単純なレイトレーシングでは，配列に含まれるすべてのプリミティブとレイの交差判定を行うため，その数に比例した計算時間がかかる．そこで空間データ構造を用いることで，レイが通過する付近の物体とのみ交差判定を行い，走査を最小限に抑えることが可能になる．空間分割アルゴリズムはそれぞれに長所と短所があり，実装する内容に応じてそれらを使い分ける必要があるが，Havran は 12 種類の空間データ構造のアルゴリズムを比較した結果, kd-tree が平均的に高いパフォーマンスを示したと報告した[17]．したがって本研究では， kd-tree を GPU レイトレーシングに適用することで更なるパフォーマンス改善を目指す．

4.3.1 空間データ構造による走査の改善

基本的な kd-tree は Bentley[3]に提案された二分木構造を持つもので，空間を三次元空間の軸に対して平行に分割し深さ優先探索（DFS: depth-first search）の順にノードを生成する．kd-tree は各ノード内に位置するプリミティブを記録することでシーンを管理するため，オブジェクトを移動・変形させた場合はその都度，再構築する必要がある．しかし，kd-tree を構築する際は，分割面を決定する関数 SAH (surface area heuristic)の計算に時間を要するため，フレーム毎に再構築を行うことでリアルタイム性を損なっていた．

SAH はノード走査のコストを表す定数を C_{ts}，ノードを位置 x で分割した際の左子ノードのコストを $C_L(x)$，右子ノードコストを $C_R(x)$，左子ノードの表面積を $A_L(x)$，右子ノードの表面積を $A_R(x)$，空間全体の表面積を A としたとき式(6)で与えられる．

$$SAH(x) = C_{ts} + \frac{C_L(x)A_L(x)}{A} + \frac{C_R(x)A_R(x)}{A} \qquad (6)$$

そこで GPU により並列処理を行いリアルタイムで kd-tree の構築を行う Zhou らの手法を用い織布のアニメーション生成を図る．

4.3.2 リアルタイム kd-tree 構築アルゴリズム

Zhou らの手法では，従来までの DFS による kd-tree の構築とは異なり幅優先探索（BFS: Breadth first search）に従って，同レベルのノードを並列的に生成する．ノードの分割面を決定する際は含まれている三角形の個数に応じてノードを分類し，それぞれに異なった処理を行う．含まれる三角形が閾値を超える場合は large node とし，SAH によるコスト計算は行わずノードを図 4(b)のように Axis-Aligned Bounding Box (AABB)の最も長い辺の中点で分割する．また，ノードに含まれる三角形を取り囲む AABB の各辺がノード自体の AABB と比較したときに，一定以上プリミティブの含まれない空間があれば，図 4(a)のようにその空間を切り出す．

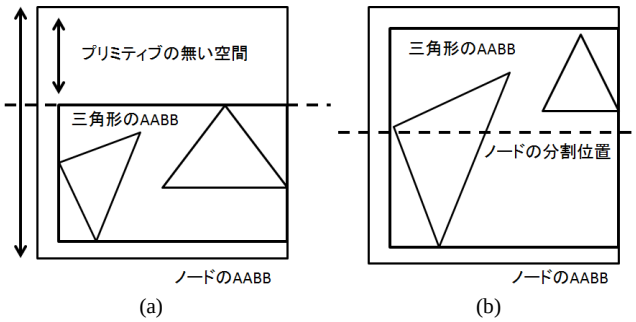
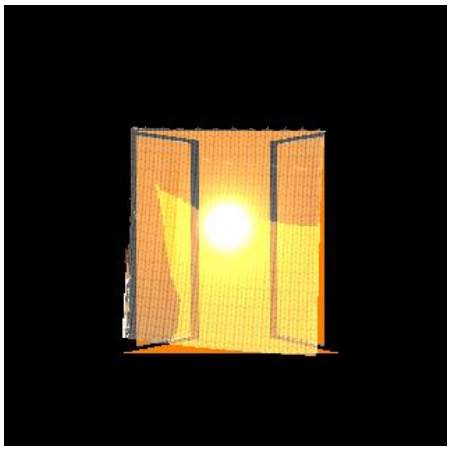
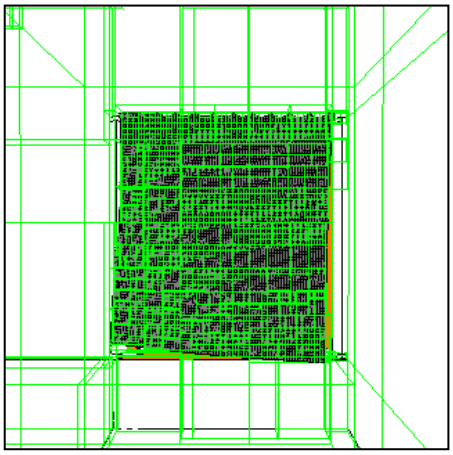


図 5　large node での分割

プリミティブの数が閾値以下の場合は Small node に分類し，ノードに含まれるプリミティブ個々を取り囲むバウンディングボックスの辺を分割面の候補としてあらか

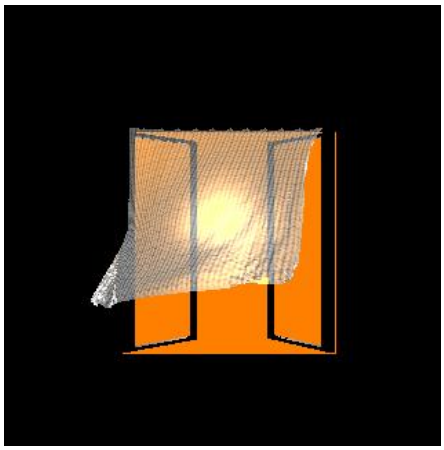


(a) 初期状態の織布のレンダリング画像

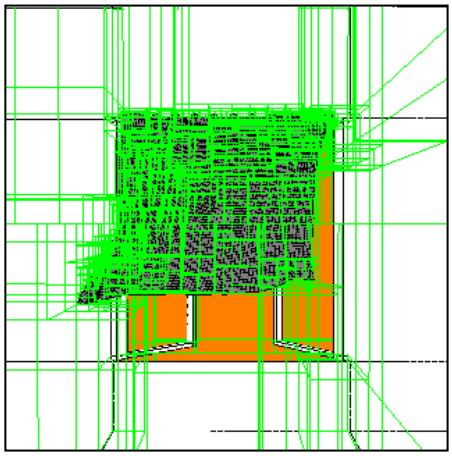


(b) 初期状態のシーンにおける kd-tree

図 6　結果画像：初期状態



(a) 変形後の織布のレンダリング画像



(b) 変形後のシーンにおける kd-tree

図 7　結果画像：変形後

じめ記録する．またノードに含まれるプリミティブはビットマスクに記録し， SAH を計算することで分割平面を求める．分割を繰り返し，現在のノードよりも算出したコストが悪化した場合は新たな分割を中止し，これを葉ノードとする．

4.3.3 kd-tree 視覚化ツール

直感的にシーン中の状況を判断するために本研究では OpenGL を用いて，シーン中の情報を取得しグラフィックスとして出力するツールを作成する．これにより kd-tree の再構築が正しく行われているのかを容易に判断することが可能となる．視覚化する kd-tree のデータはノードの分割を行う際に各ノードの AABB の情報を取得し，光源位置や座標軸に関する情報はレイトレーシングのプロセスで個別に取得される．最終的な画像は Z バッファ法を用いて別画面に出力される．出力の際にシェーディングは行わず，光源は黄色，kd-tree の AABB は緑色，座標軸はグレー，プリミティブは各プリミティブの拡散反射係数を元にした色で描画を行う．またプリミティブの表示を行う際は視認性を向上するため，各プリミティブのエッジを黒く強調する．

5. 透過性織布のアニメーション生成

本章では実際に織布のアニメーションを生成することで，提案手法の有効性を検証する．レンダリングの際は，点光源を窓の外側に配置し，カーテンを動かすための風を，窓面に対して垂直方向に発生させた． 図 5(a)と図 6(a)にレンダリング結果を，図 5(b)と図 6(b)に kd-tree の様子を示す．

5.1 アニメーション生成における並列環境の構築

表 1 実行環境と実行速度

CPU	Intel(R) Core(TM) i7-2600 3.40GHz
メモリ	8.00GB RAM
GPU	NVIDIA Quadro 4000：2.0 GB
OS	Microsoft Windows 7 Professional, 64-bit
ポリゴン数	12786
解像度	320×320
追跡レベル	4
fps	4.65

プログラム言語には C++，CPU と GPU の並列処理環境は NVIDIA が提供する統合開発環境である CUDA を用いて構築する．GPU からの最終的な出力である色情報は，一度 CPU に渡され OpenGL を用いて表示される．画面の出力やキー検出等の入出力管理にのみ OpenGL を利用し，モデルデータの読み取りやカメラ移動などは C++を用いて実装する．部屋のモデルは実行時に OBJ ファイルを読み取ることで生成し，織布のアニメーションはクロスシ

ミュレーションを行い，フレーム毎に各プリミティブの頂点座標を計算することで生成する．レンダリングに使用した PC の詳細と，実行速度を表 1 に示す．

5.2 考察

光の挙動を実際にシミュレートするレイトレーシング法に Woven cloth microfacet BSDF モデルを適用することで，より高品質な画像を出力することに成功した．また，シーン中の織布の動きに伴い,kd-tree の再構築を行うことで，透過性織布のアニメーションをリアルタイムに生成することを確認した．これにより織布の形状が変化することによって生じる光の複雑な経路の変化を表現することができ，更なる質感表現の向上につながったといえる．

また，kd-tree 視覚化ツールに関しても，実際にシーン中の情報をグラフィクスとして表示することによって，理解することの難しかった空間分割の様子や光源座標を容易に確認することができるようになり有用性を確認できた．

6. まとめ

本研究では高品質な織布の質感を表現するために，レイトレーシング法に物理ベースモデルである Woven cloth microfacet BSDF モデルを適用し織布固有の透過特性を表現した．また GPU による並列処理によりレイトレーシングの高速化を図り，kd-tree の再構築を行うことでアニメーションをリアルタイムに生成した．これによりラスタライズ法では表現することのできなかった，光の複雑な経路の変化を表現することができ有用性を確認できた．kd-tree 視覚化ツールに関しても，シーン中の情報を容易に判断することが可能となり，有用性を確認できた．

今回実装したレイトレーサーでは局所的な照明効果のみに限定されるため，光源からの光の影響しか受けず全体的に暗い印象を受けてしまう．したがって今後は，GI を考慮したアニメーションを生成するためにフォトンマッピング法を導入する予定である．

参考文献

1) 飛谷謙介,石田適志,野村周平,長田典子: Microfacet BSDF モデルを用いた織布の CG 表現—カーテンアニメーションカタログの制作—. 精密工学会誌, Vol.79, No.11, pp.1165-1170 (2013).
2) S, Nomura. A, Ishida. E Ishigo. T Okamoto, Y, Mizushima. & N, Nagata.: Lace curtain modeling and rendering of woven cloth using microfacet BSDF: production of a catalog of curtain animations. ACM SIGGRAPH 2011: posters, Modeling, 71 (2011).
3) J,L,Bentley.: Multidimensional binary search trees used for associative searching. Communications of the ACM, Vol.18, No.9, pp.509-517 (1975).
4) M, Sattler. R, Sarlette. & R, Klein.: Effcient and Realistic Visualization of Cloth. Institute of Computer Science II, University of Bonn, Germany Eurographics Symposium on Rendering (2003).
5) L, Claustres. L, Barthe & M, Paulin.: Wavelet Encoding of BRDFs for Real-Time Rendering. Graphics Interface Conference 2007 28-30 May, pp.169-176 (2007).
6) R, Habel. A, Kusternig. & M, Wimmer.: Physically Based Real-Time Translucency for Leaves. Eurographics Symposium on Rendering 2007 pp.253-263 (2007).
7) C, Rousiers. A, Bousseau. K Subr. N, Holzschuch. & R, Ramamoorthi.: Real-Time Rendering of Rough Refraction. IEEE Trans. Visualization and Computer Graphics, Vo1.8, No.10 (2014).
8) T, J, Purcell. I, Buck. W, R, Mark. and P, Hanrahan.: Ray tracing on programmable graphics hardware. In ACM SIGGRAPH 2005 Courses, pp. 268 (2005).
9) I, Wald et al.,: Ray tracing deformable scenes using dynamic bounding volume hierarchies, ACM Trans. Graph, 2007, Vol 26, No.1, Article 6 (2007).
10) I, Wald. and V, Havran.: On building fast kd-trees for ray tracing, and on doing that in O(NlogN). In Proceedings of IEEE Symposium on Interactive Ray Tracing, pp.61-69 (2006).
11) S, Popov. J, Gunther. H.-P, Seidel. and P, Slusallek.: Experiences with streaming construction of SAH KDtrees. In IEEE Symposium on Interactive Ray Tracing, pp.89-94 (2006).
12) M, Shevtsov. A, Soupikov. and A, Kapustin.: Highly. Parallel Fast KD-tree Construction for Interactive Ray Tracing of Dynamic Scenes. Computer Graphics Forum, Vol. 26, No.3, pp. 395-404 (2007).
13) K, Zhou. Q, Hou. R, Wang. and G, Baining.: Real-time KD-tree Construction on Graphics Hardware. ACM SIGGRAPH Asia (2008).
14) V, Havran.: Heuristic Ray Shooting Algorithms. Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague (2000).